

OFFICIAL MICROSOFT LEARNING PRODUCT

20768B

Developing SQL Data Models

MCT USE ONLY. STUDENT USE PROHIBITED

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2017 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

Product Number: 20768B

Part Number (if applicable): X21-31973

Released: 03/2017

MICROSOFT LICENSE TERMS MICROSOFT INSTRUCTOR-LED COURSEWARE

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to your use of the content accompanying this agreement which includes the media on which you received it, if any. These license terms also apply to Trainer Content and any updates and supplements for the Licensed Content unless other terms accompany those items. If so, those terms apply.

**BY ACCESSING, DOWNLOADING OR USING THE LICENSED CONTENT, YOU ACCEPT THESE TERMS.
IF YOU DO NOT ACCEPT THEM, DO NOT ACCESS, DOWNLOAD OR USE THE LICENSED CONTENT.**

If you comply with these license terms, you have the rights below for each license you acquire.

1. DEFINITIONS.

- a. "Authorized Learning Center" means a Microsoft IT Academy Program Member, Microsoft Learning Competency Member, or such other entity as Microsoft may designate from time to time.
- b. "Authorized Training Session" means the instructor-led training class using Microsoft Instructor-Led Courseware conducted by a Trainer at or through an Authorized Learning Center.
- c. "Classroom Device" means one (1) dedicated, secure computer that an Authorized Learning Center owns or controls that is located at an Authorized Learning Center's training facilities that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
- d. "End User" means an individual who is (i) duly enrolled in and attending an Authorized Training Session or Private Training Session, (ii) an employee of a MPN Member, or (iii) a Microsoft full-time employee.
- e. "Licensed Content" means the content accompanying this agreement which may include the Microsoft Instructor-Led Courseware or Trainer Content.
- f. "Microsoft Certified Trainer" or "MCT" means an individual who is (i) engaged to teach a training session to End Users on behalf of an Authorized Learning Center or MPN Member, and (ii) currently certified as a Microsoft Certified Trainer under the Microsoft Certification Program.
- g. "Microsoft Instructor-Led Courseware" means the Microsoft-branded instructor-led training course that educates IT professionals and developers on Microsoft technologies. A Microsoft Instructor-Led Courseware title may be branded as MOC, Microsoft Dynamics or Microsoft Business Group courseware.
- h. "Microsoft IT Academy Program Member" means an active member of the Microsoft IT Academy Program.
- i. "Microsoft Learning Competency Member" means an active member of the Microsoft Partner Network program in good standing that currently holds the Learning Competency status.
- j. "MOC" means the "Official Microsoft Learning Product" instructor-led courseware known as Microsoft Official Course that educates IT professionals and developers on Microsoft technologies.
- k. "MPN Member" means an active Microsoft Partner Network program member in good standing.

MCT USE ONLY. STUDENT USE PROHIBITED

- l. "Personal Device" means one (1) personal computer, device, workstation or other digital electronic device that you personally own or control that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
- m. "Private Training Session" means the instructor-led training classes provided by MPN Members for corporate customers to teach a predefined learning objective using Microsoft Instructor-Led Courseware. These classes are not advertised or promoted to the general public and class attendance is restricted to individuals employed by or contracted by the corporate customer.
- n. "Trainer" means (i) an academically accredited educator engaged by a Microsoft IT Academy Program Member to teach an Authorized Training Session, and/or (ii) a MCT.
- o. "Trainer Content" means the trainer version of the Microsoft Instructor-Led Courseware and additional supplemental content designated solely for Trainers' use to teach a training session using the Microsoft Instructor-Led Courseware. Trainer Content may include Microsoft PowerPoint presentations, trainer preparation guide, train the trainer materials, Microsoft One Note packs, classroom setup guide and Pre-release course feedback form. To clarify, Trainer Content does not include any software, virtual hard disks or virtual machines.

2. **USE RIGHTS.** The Licensed Content is licensed not sold. The Licensed Content is licensed on a **one copy per user basis**, such that you must acquire a license for each individual that accesses or uses the Licensed Content.

2.1 Below are five separate sets of use rights. Only one set of rights apply to you.

a. **If you are a Microsoft IT Academy Program Member:**

- i. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
- ii. For each license you acquire on behalf of an End User or Trainer, you may either:
 - 1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User who is enrolled in the Authorized Training Session, and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
 - 2. provide one (1) End User with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
 - 3. provide one (1) Trainer with the unique redemption code and instructions on how they can access one (1) Trainer Content,**provided you comply with the following:**
- iii. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
- iv. you will ensure each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,
- v. you will ensure that each End User provided with the hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
- vi. you will ensure that each Trainer teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,

- vii. you will only use qualified Trainers who have in-depth knowledge of and experience with the Microsoft technology that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Authorized Training Sessions,
- viii. you will only deliver a maximum of 15 hours of training per week for each Authorized Training Session that uses a MOC title, and
- ix. you acknowledge that Trainers that are not MCTs will not have access to all of the trainer resources for the Microsoft Instructor-Led Courseware.

b. If you are a Microsoft Learning Competency Member:

- i. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
- ii. For each license you acquire on behalf of an End User or Trainer, you may either:
 - 1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Authorized Training Session and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware provided, **or**
 - 2. provide one (1) End User attending the Authorized Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
 - 3. you will provide one (1) Trainer with the unique redemption code and instructions on how they can access one (1) Trainer Content,
provided you comply with the following:
- iii. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
- iv. you will ensure that each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,
- v. you will ensure that each End User provided with a hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
- vi. you will ensure that each Trainer teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,
- vii. you will only use qualified Trainers who hold the applicable Microsoft Certification credential that is the subject of the Microsoft Instructor-Led Courseware being taught for your Authorized Training Sessions,
- viii. you will only use qualified MCTs who also hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Authorized Training Sessions using MOC,
- ix. you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and
- x. you will only provide access to the Trainer Content to Trainers.

c. If you are a MPN Member:

- i. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
- ii. For each license you acquire on behalf of an End User or Trainer, you may either:
 1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Private Training Session, and only immediately prior to the commencement of the Private Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
 2. provide one (1) End User who is attending the Private Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
 3. you will provide one (1) Trainer who is teaching the Private Training Session with the unique redemption code and instructions on how they can access one (1) Trainer Content,
provided you comply with the following:
- iii. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
- iv. you will ensure that each End User attending an Private Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Private Training Session,
- v. you will ensure that each End User provided with a hard copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
- vi. you will ensure that each Trainer teaching an Private Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Private Training Session,
- vii. you will only use qualified Trainers who hold the applicable Microsoft Certification credential that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Private Training Sessions,
- viii. you will only use qualified MCTs who hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Private Training Sessions using MOC,
- ix. you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and
- x. you will only provide access to the Trainer Content to Trainers.

d. If you are an End User:

For each license you acquire, you may use the Microsoft Instructor-Led Courseware solely for your personal training use. If the Microsoft Instructor-Led Courseware is in digital format, you may access the Microsoft Instructor-Led Courseware online using the unique redemption code provided to you by the training provider and install and use one (1) copy of the Microsoft Instructor-Led Courseware on up to three (3) Personal Devices. You may also print one (1) copy of the Microsoft Instructor-Led Courseware. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

e. If you are a Trainer.

- i. For each license you acquire, you may install and use one (1) copy of the Trainer Content in the form provided to you on one (1) Personal Device solely to prepare and deliver an Authorized Training Session or Private Training Session, and install one (1) additional copy on another Personal Device as a backup copy, which may be used only to reinstall the Trainer Content. You may not install or use a copy of the Trainer Content on a device you do not own or control. You may also print one (1) copy of the Trainer Content solely to prepare for and deliver an Authorized Training Session or Private Training Session.

- ii. You may customize the written portions of the Trainer Content that are logically associated with instruction of a training session in accordance with the most recent version of the MCT agreement. If you elect to exercise the foregoing rights, you agree to comply with the following: (i) customizations may only be used for teaching Authorized Training Sessions and Private Training Sessions, and (ii) all customizations will comply with this agreement. For clarity, any use of “*customize*” refers only to changing the order of slides and content, and/or not using all the slides or content, it does not mean changing or modifying any slide or content.

2.2 **Separation of Components.** The Licensed Content is licensed as a single unit and you may not separate their components and install them on different devices.

2.3 **Redistribution of Licensed Content.** Except as expressly provided in the use rights above, you may not distribute any Licensed Content or any portion thereof (including any permitted modifications) to any third parties without the express written permission of Microsoft.

2.4 **Third Party Notices.** The Licensed Content may include third party code that Microsoft, not the third party, licenses to you under this agreement. Notices, if any, for the third party code are included for your information only.

2.5 **Additional Terms.** Some Licensed Content may contain components with additional terms, conditions, and licenses regarding its use. Any non-conflicting terms in those conditions and licenses also apply to your use of that respective component and supplements the terms described in this agreement.

3. **LICENSED CONTENT BASED ON PRE-RELEASE TECHNOLOGY.** If the Licensed Content’s subject matter is based on a pre-release version of Microsoft technology (“**Pre-release**”), then in addition to the other provisions in this agreement, these terms also apply:
- a. **Pre-Release Licensed Content.** This Licensed Content subject matter is on the Pre-release version of the Microsoft technology. The technology may not work the way a final version of the technology will and we may change the technology for the final version. We also may not release a final version. Licensed Content based on the final version of the technology may not contain the same information as the Licensed Content based on the Pre-release version. Microsoft is under no obligation to provide you with any further content, including any Licensed Content based on the final version of the technology.
 - b. **Feedback.** If you agree to give feedback about the Licensed Content to Microsoft, either directly or through its third party designee, you give to Microsoft without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft technology, Microsoft product, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its technology, technologies, or products to third parties because we include your feedback in them. These rights survive this agreement.
 - c. **Pre-release Term.** If you are an Microsoft IT Academy Program Member, Microsoft Learning Competency Member, MPN Member or Trainer, you will cease using all copies of the Licensed Content on the Pre-release technology upon (i) the date which Microsoft informs you is the end date for using the Licensed Content on the Pre-release technology, or (ii) sixty (60) days after the commercial release of the technology that is the subject of the Licensed Content, whichever is earliest (“**Pre-release term**”). Upon expiration or termination of the Pre-release term, you will irretrievably delete and destroy all copies of the Licensed Content in your possession or under your control.

4. **SCOPE OF LICENSE.** The Licensed Content is licensed, not sold. This agreement only gives you some rights to use the Licensed Content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the Licensed Content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the Licensed Content that only allows you to use it in certain ways. Except as expressly permitted in this agreement, you may not:
 - access or allow any individual to access the Licensed Content if they have not acquired a valid license for the Licensed Content,
 - alter, remove or obscure any copyright or other protective notices (including watermarks), branding or identifications contained in the Licensed Content,
 - modify or create a derivative work of any Licensed Content,
 - publicly display, or make the Licensed Content available for others to access or use,
 - copy, print, install, sell, publish, transmit, lend, adapt, reuse, link to or post, make available or distribute the Licensed Content to any third party,
 - work around any technical limitations in the Licensed Content, or
 - reverse engineer, decompile, remove or otherwise thwart any protections or disassemble the Licensed Content except and only to the extent that applicable law expressly permits, despite this limitation.

5. **RESERVATION OF RIGHTS AND OWNERSHIP.** Microsoft reserves all rights not expressly granted to you in this agreement. The Licensed Content is protected by copyright and other intellectual property laws and treaties. Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the Licensed Content.

6. **EXPORT RESTRICTIONS.** The Licensed Content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the Licensed Content. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.

7. **SUPPORT SERVICES.** Because the Licensed Content is “as is”, we may not provide support services for it.

8. **TERMINATION.** Without prejudice to any other rights, Microsoft may terminate this agreement if you fail to comply with the terms and conditions of this agreement. Upon termination of this agreement for any reason, you will immediately stop all use of and delete and destroy all copies of the Licensed Content in your possession or under your control.

9. **LINKS TO THIRD PARTY SITES.** You may link to third party sites through the use of the Licensed Content. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.

10. **ENTIRE AGREEMENT.** This agreement, and any additional terms for the Trainer Content, updates and supplements are the entire agreement for the Licensed Content, updates and supplements.

11. **APPLICABLE LAW.**
 - a. United States. If you acquired the Licensed Content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

b. Outside the United States. If you acquired the Licensed Content in any other country, the laws of that country apply.

- 12. LEGAL EFFECT.** This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the Licensed Content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.
- 13. DISCLAIMER OF WARRANTY. THE LICENSED CONTENT IS LICENSED "AS-IS" AND "AS AVAILABLE." YOU BEAR THE RISK OF USING IT. MICROSOFT AND ITS RESPECTIVE AFFILIATES GIVES NO EXPRESS WARRANTIES, GUARANTEES, OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT AND ITS RESPECTIVE AFFILIATES EXCLUDES ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**
- 14. LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT, ITS RESPECTIVE AFFILIATES AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO US\$5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

This limitation applies to

- anything related to the Licensed Content, services, content (including code) on third party Internet sites or third-party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this Licensed Content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection des consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit local, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 \$ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence, aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers; et.
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Revised July 2013

Welcome!

Thank you for taking our training! We've worked together with our Microsoft Certified Partners for Learning Solutions and our Microsoft IT Academies to bring you a world-class learning experience—whether you're a professional looking to advance your skills or a student preparing for a career in IT.

- **Microsoft Certified Trainers and Instructors**—Your instructor is a technical and instructional expert who meets ongoing certification requirements. And, if instructors are delivering training at one of our Certified Partners for Learning Solutions, they are also evaluated throughout the year by students and by Microsoft.
- **Certification Exam Benefits**—After training, consider taking a Microsoft Certification exam. Microsoft Certifications validate your skills on Microsoft technologies and can help differentiate you when finding a job or boosting your career. In fact, independent research by IDC concluded that 75% of managers believe certifications are important to team performance¹. Ask your instructor about Microsoft Certification exam promotions and discounts that may be available to you.
- **Customer Satisfaction Guarantee**—Our Certified Partners for Learning Solutions offer a satisfaction guarantee and we hold them accountable for it. At the end of class, please complete an evaluation of today's experience. We value your feedback!

We wish you a great learning experience and ongoing success in your career!

Sincerely,

Microsoft Learning
www.microsoft.com/learning

Microsoft | Learning

¹ IDC, Value of Certification: Team Certification and Organizational Performance, November 2006

Acknowledgements

Microsoft Learning would like to acknowledge and thank the following for their contribution towards developing this title. Their effort at various stages in the development has ensured that you have a good classroom experience.

Alistair Matthews – Content Developer

Alistair Matthews is an experienced consultant and technical author who has worked with a wide variety of Microsoft technologies over his nineteen-year career. Recent projects have included revising SQL Server 2014 courseware, authorizing Azure courseware, and developing SharePoint apps, all for different groups within Microsoft. He designed and wrote the Microsoft Official Curriculum course number 20486 "Developing ASP.NET 4.5 MVC Web Applications" and several modules in the SharePoint developers' courses numbers 20488 and 20489. He has developed content for Microsoft Press books, TechNet, and MSDN and has written e-learning courseware and examination questions. He lives in Cornwall in the United Kingdom.

Ed Harper – Content Developer

Ed Harper is a database developer specializing in Microsoft SQL Server. Ed has worked with SQL Server since 1999, and has developed and designed transaction-processing and reporting systems for cloud security, telecommunications, and financial services companies.

Jamie Newman – Content Developer

Jamie Newman became an IT trainer in 1997, first for an IT training company and later for a university, where he became involved in developing courses as well as training them. He began to specialize in databases and eventually moved into database consultancy. In recent years he has specialized in SQL Server and has set up multi user systems that are accessed nationwide. Despite now being more involved with development work, Jamie still likes to deliver IT training courses when the opportunity arises!

John Daisley – Content Developer

John Daisley is a mixed vendor Business Intelligence and Data Warehousing contractor with a wealth of data warehousing and Microsoft SQL Server database administration experience. Having worked in the Business Intelligence arena for over a decade, John has extensive experience of implementing business intelligence and database solutions using a wide range of technologies and across a wide range of industries including airlines, engineering, financial services, and manufacturing.

Martin Ellis – Content Developer

Martin is a technical author, consultant, and SQL Server subject matter expert. As a Microsoft Certified Trainer (MCT), he has extensive experience of delivering Microsoft training courses on technologies including SQL Server and Windows Server. He has authored and co-authored numerous Microsoft courses for the SQL Server 2008 R2, SQL Server 2012, SQL Server 2014, and SQL Server 2016 curriculums, and is the author of several white papers and other technical documents. Martin also provides consultancy services for organizations in the planning, implementation, optimization, and maintenance of SQL Server solutions.

Simon Butler – Content Developer

Simon Butler FISTC is a highly-experienced Senior Technical Writer with nearly 30 years' experience in the profession. He has written training materials and other information products for several high-profile clients. He is a Fellow of the Institute of Scientific and Technical Communicators (ISTC), the UK professional body for Technical Writers/Authors. To gain this, his skills, experience and knowledge have been judged and assessed by the Membership Panel. He is also a Past President of the Institute and has been a tutor on the ISTC Open Learning course in Technical Communication techniques. His writing skills are augmented by extensive technical skills gained within the computing and electronics fields.

Geoff Allix – Technical Reviewer

Geoff Allix is a Microsoft SQL Server subject matter expert and professional content developer at Content Master—a division of CM Group Ltd. As a Microsoft Certified Trainer, Geoff has delivered training courses on SQL Server since version 6.5. Geoff is a Microsoft Certified IT Professional for SQL Server and has extensive experience in designing and implementing database and BI solutions on SQL Server technologies, and has provided consultancy services to organizations seeking to implement and optimize database solutions.

Lin Joyner – Technical Reviewer

Lin is an experienced Microsoft SQL Server developer and administrator. She has worked with SQL Server since version 6.0 and previously as a Microsoft Certified Trainer, delivered training courses across the UK. Lin has a wide breadth of knowledge across SQL Server technologies, including BI and Reporting Services. Lin also designs and authors SQL Server and .NET development training materials. She has been writing instructional content for Microsoft for over 15 years.

Contents

Module 1: Introduction to Business Intelligence and Data Modeling

Module Overview	1-1
Lesson 1: Overview of Business Intelligence and Data Models	1-2
Lesson 2: Microsoft Business Intelligence Platform	1-7
Lab: Exploring a Business Intelligence Solution	1-13
Module Review and Takeaways	1-17

Module 2: Creating Multidimensional Databases

Module Overview	2-1
Lesson 1: Introduction to Multidimensional Analysis	2-2
Lesson 2: Data Sources and Data Source Views	2-8
Lesson 3: Cubes	2-12
Lesson 4: Overview of Cube Security	2-17
Lesson 5: Configuring SSAS	2-23
Lesson 6: Monitoring SSAS	2-28
Lab: Creating a Multidimensional Database	2-34
Module Review and Takeaways	2-39

Module 3: Working with Cubes and Dimensions

Module Overview	3-1
Lesson 1: Configuring Dimensions	3-2
Lesson 2: Defining Attribute Hierarchies	3-6
Lesson 3: Implementing Sorting and Grouping for Attributes	3-12
Lesson 4: Slowly Changing Dimensions	3-15
Lab: Working with Cubes and Dimensions	3-18
Module Review and Takeaways	3-23

Module 4: Working with Measures and Measure Groups

Module Overview	4-1
Lesson 1: Working with Measures	4-2
Lesson 2: Working with Measure Groups	4-6
Lab: Configuring Measures and Measure Groups	4-14
Module Review and Takeaways	4-18

Module 5: Introduction to MDX

Module Overview	5-1
Lesson 1: MDX Fundamentals	5-2
Lesson 2: Adding Calculations to a Cube	5-10
Lesson 3: Using MDX to Query a Cube	5-17
Lab: Using MDX	5-23
Module Review and Takeaways	5-27

Module 6: Customizing Cube Functionality

Module Overview	6-1
Lesson 1: Implementing Key Performance Indicators	6-2
Lesson 2: Implementing Actions	6-8
Lesson 3: Implementing Perspectives	6-11
Lesson 4: Implementing Translations	6-13
Lab: Customizing a Cube	6-15
Module Review and Takeaways	6-19

Module 7: Implementing a Tabular Data Model by Using SQL Server Analysis Services

Module Overview	7-1
Lesson 1: Introduction to Tabular Data Models in SQL Server Analysis Services	7-2
Lesson 2: Creating a Tabular Data Model	7-7
Lesson 3: Using a SQL Server Analysis Services Tabular Model in an Enterprise BI Solution	7-17
Lab: Implementing a Tabular Data Model in SQL Server Analysis Services	7-25
Module Review and Takeaways	7-35

Module 8: Introduction to Data Analysis Expressions (DAX)

Module Overview	8-1
Lesson 1: DAX Fundamentals	8-2
Lesson 2: Using DAX to Create Calculated Columns and Measures in a Tabular Data Model	8-12
Lab: Using DAX to Enhance a Tabular Data Model	8-24
Module Review and Takeaways	8-30

Module 9: Performing Predictive Analysis with Data Mining

Module Overview	9-1
Lesson 1: Overview of Data Mining	9-2
Lesson 2: Creating a Custom Data Mining Solution	9-7
Lesson 3: Validating a Data Mining Model	9-15
Lesson 4: Connecting to and Consuming a Data Mining Model	9-20
Lab: Using Data Mining to Support a Marketing Campaign	9-23
Module Review and Takeaways	9-28

Lab Answer Keys

Module 1 Lab: Exploring a Business Intelligence Solution	L01-1
Module 2 Lab: Creating a Multidimensional Database	L02-1
Module 3 Lab: Working with Cubes and Dimensions	L03-1
Module 4 Lab: Configuring Measures and Measure Groups	L04-1
Module 5 Lab: Using MDX	L05-1
Module 6 Lab: Customizing a Cube	L06-1
Module 7 Lab: Implementing a Tabular Data Model in SQL Server Analysis Services	L07-1
Module 8 Lab: Using DAX to Enhance a Tabular Data Model	L08-1
Module 9 Lab: Using Data Mining to Support a Marketing Campaign	L09-1

About This Course

This section provides a brief description of the course, audience, suggested prerequisites, and course objectives.

Course Description



Note: This release ('B') MOC version of course 20768B has been developed on the RTM version of SQL Server 2016.

The focus of this 3-day instructor-led course is on creating managed enterprise BI solutions. It describes how to implement both multidimensional and tabular data models and how to create cubes, dimensions, measures, and measure groups.

Audience

The primary audience for this course are database professionals who need to fulfil BI Developer role to create enterprise BI solutions. Primary responsibilities will include:

- Implementing multidimensional databases by using SQL Server Analysis Services
- Creating tabular semantic data models for analysis by using SQL Server Analysis Services

The secondary audiences for this course are 'power' information workers.

Student Prerequisites

In addition to their professional experience, students who attend this training should already have the following technical knowledge:

- At least 2 years' experience of working with relational databases, including:
 - Designing a normalized database
 - Creating tables and relationships
 - Querying with Transact-SQL
 - Some basic knowledge of data warehouse schema topology (including star and snowflake schemas)
 - Some exposure to basic programming constructs (such as looping and branching)
 - An awareness of key business priorities such as revenue, profitability, and financial accounting is desirable

Course Objectives

After completing this course, students will be able to:

- Describe the components, architecture, and nature of a BI solution.
- Create a multidimensional database with analysis services.
- Implement dimensions in a cube.
- Implement measures and measure groups in a cube.
- Use MDX syntax.
- Customize a cube.

- Implement a tabular database.
- Use DAX to query a tabular model.
- Use data mining for predictive analysis.

Course Outline

The course outline is as follows:

- Module 1: 'Introduction to Business Intelligence and data modelling' introduces key BI concepts and the Microsoft BI product suite
- Module 2: 'Creating multidimensional databases' describes the steps required to create a multidimensional database with analysis services.
- Module 3: 'Working with cubes and dimensions' describes how to implement dimensions in a cube.
- Module 4: 'Working with measures and measure groups' describes how to implement measures and measure groups in a cube.
- Module 5: 'Introduction to MDX' describes the MDX syntax and how to use MDX.
- Module 6: 'Customizing cube functionality' describes how to customize a cube.
- Module 7: 'Implementing a tabular data model by using analysis services' describes how to implement a tabular data model in PowerPivot.
- Module 8: 'Introduction to Data Analysis Expression (DAX)' describes how to use DAX to create measures and calculated columns in a tabular data model.
- Module 9: 'Performing predictive analysis with data mining' describes how to use data mining for predictive analysis.

Course Materials

The following materials are included with your kit:

- **Course Handbook:** a succinct classroom learning guide that provides the critical technical information in a crisp, tightly-focused format, which is essential for an effective in-class learning experience.
 - **Lessons:** guide you through the learning objectives and provide the key points that are critical to the success of the in-class learning experience.
 - **Labs:** provide a real-world, hands-on platform for you to apply the knowledge and skills learned in the module.
 - **Module Reviews and Takeaways:** provide on-the-job reference material to boost knowledge and skills retention.
 - **Lab Answer Keys:** provide step-by-step lab solution guidance.



Additional Reading: Course Companion Content on the
<http://www.microsoft.com/learning/en/us/companion-moc.aspx> **Site:** searchable, easy-to-browse digital content with integrated premium online resources that supplement the Course Handbook.

- **Modules:** include companion content, such as questions and answers, detailed demo steps and additional reading links, for each lesson. Additionally, they include Lab Review questions and answers and Module Reviews and Takeaways sections, which contain the review questions and answers, best practices, common issues and troubleshooting tips with answers, and real-world issues and scenarios with answers.
- **Resources:** include well-categorized additional resources that give you immediate access to the most current premium content on TechNet, MSDN®, or Microsoft® Press®.



Additional Reading: Student Course files on the

<http://www.microsoft.com/learning/en/us/companion-moc.aspx> **Site:** includes the Allfiles.exe, a self-extracting executable file that contains all required files for the labs and demonstrations.

- **Course evaluation:** at the end of the course, you will have the opportunity to complete an online evaluation to provide feedback on the course, training facility, and instructor.
- To provide additional comments or feedback on the course, send email to mcsprt@microsoft.com. To inquire about the Microsoft Certification Program, send an email to mcphelp@microsoft.com.

Virtual Machine Environment

This section provides the information for setting up the classroom environment to support the business scenario of the course.

Virtual Machine Configuration

In this course, you will use Microsoft® Hyper-V® to perform the labs.



Note: At the end of each lab, you must revert the virtual machines to a snapshot. You can find the instructions for this procedure at the end of each lab

The following table shows the role of each virtual machine that is used in this course:

Virtual machine	Role
20768B-MIA-DC	MIA-DC1 is a domain controller.
20768B-MIA-SQL	MIA-SQL has SQL Server 2016 installed
MSL-TMG1	TMG1 is used to access the internet

Software Configuration

The following software is installed on the virtual machines:

- Windows Server 2012 R2
- SQL2016
- Microsoft Office 2016
- SharePoint 2013SP1

Course Files

The files associated with the labs in this course are located in the D:\Labfiles folder on the 20768B-MIA-SQL virtual machine.

Classroom Setup

Each classroom computer will have the same virtual machine configured in the same way.

Course Hardware Level

To ensure a satisfactory student experience, Microsoft Learning requires a minimum equipment configuration for trainer and student computers in all Microsoft Learning Partner classrooms in which Official Microsoft Learning Product courseware is taught.

- Intel Virtualization Technology (Intel VT) or AMD Virtualization (AMD-V) processor
- Dual 120-gigabyte (GB) hard disks 7200 RPM Serial ATA (SATA) or better
- 16 GB of random access memory (RAM)
- DVD drive
- Network adapter
- Super VGA (SVGA) 17-inch monitor
- Microsoft mouse or compatible pointing device
- Sound card with amplified speakers

Additionally, the instructor's computer must be connected to a projection display device that supports SVGA 1024×768 pixels, 16-bit colors.

Module 1

Introduction to Business Intelligence and Data Modeling

Contents:

Module Overview	1-1
Lesson 1: Overview of Business Intelligence and Data Models	1-2
Lesson 2: Microsoft Business Intelligence Platform	1-7
Lab: Exploring a Business Intelligence Solution	1-13
Module Review and Takeaways	1-17

Module Overview

Business intelligence (BI) is a fundamental part of everyday business activity for modern organizations, enabling them to understand trends and plan future activities more effectively. A great deal of BI depends on data models, which expose data in a meaningful way, and enable much faster analysis of complex data.

Objectives

After completing this module, you will be able to:

- Describe business intelligence and data modeling.
- Describe the technologies that make up the Microsoft business intelligence platform.

Lesson 1

Overview of Business Intelligence and Data Models

The term “business intelligence” is commonly used to refer to a broad range of activities and technologies that enable organizations to better understand their data, and to use it to gain insights that can benefit the business. Many organizations use data models as the basis of their BI activities because they can improve usability and offer other benefits such as improved performance. Any BI project is a substantial undertaking and requires careful planning. To ensure that projects are successful and achieve the required return on investment (ROI), it is important to have an understanding of the key project roles in this type of project.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe key business intelligence scenarios.
- Describe current trends in business intelligence.
- Describe the purpose of data models, and their benefits.
- Explain the components of business intelligence projects and the key project roles.

Business Intelligence Scenarios

Businesses today have access to rapidly increasing volumes of data, which business users must use to make ever-faster decisions in day-to-day business operations. A typical BI solution enables end users, such as data analysts and other information workers, to produce reports and associated documentation, to view this kind of information, and to perform sophisticated data analysis. Decision makers can use the information in reports, and the insights that are obtained from data analysis, to better understand past performance and to guide them in their ongoing planning.

- Reporting
 - The communication of information gained from business intelligence
- Analysis
 - The interpretation of the business data that a business intelligence solution delivers

Reporting

Reporting is the communication of information that is gained from BI. Most organizations rely on reports to summarize business performance and activities. Consequently, most BI solutions include an element that generates reports. Typical examples are financial and management reports that include cash flow, profit and loss, a balance sheet and open orders. A retail business might require stock inventory reports, whereas a technical support call center might need a report that shows call log data. In some scenarios, users might need to view reports interactively in a web browser or custom application. In other instances, the requirement might be to send reports as email attachments in specific formats, such as Microsoft® Excel® workbooks or Microsoft Word documents. In many cases, the reports might need to be printed, for example, to send a physical report to customers or shareholders. When you are planning a reporting solution, you must consider the reports that are required, the audiences for those reports, and how they will be delivered. Microsoft SQL Server® 2016 includes Reporting Services, a powerful reporting technology that supports a wide range of reporting scenarios.



Note: This course does not describe SQL Server 2016 Reporting Services in detail. For more information about using Reporting Services, you can attend course number 20770-2A: *Analyzing Data with SQL Server Reporting Services*.

Analysis

Analysis is the interpretation of business data that the BI solution delivers. For business analysts in particular, performing analysis is a discrete activity that involves using specialist analytical tools to examine data in analytical models. For others, analysis is simply a part of everyday work and involves using reports or dashboards as a basis for business decision making. In general, when you are planning a BI solution, you should consider the following kinds of analytical requirements:

- **Interactive analysis.** Some BI solutions must support interactive “slice and dice” analysis in business tools such as Excel or specialist data analysis tools. The resulting information can then be published as a report.
- **Dashboards and scorecards.** Commonly, analytical data can be summarized in a dashboard or scorecard and embedded into business applications or portals, such as Microsoft SharePoint® Server sites. These solutions might provide some interactivity to enable users to “drill down” into specific details, or they may simply show important key performance indicators (KPIs).
- **Data mining.** Most analysis and reporting concerns historical data, but a BI solution can also support predictive analysis by using historical data to determine trends and patterns.

Trends in Business Intelligence

There have been many changes in the area of BI in recent years, and the traditional model of a complex, on-site, IT-managed BI solution is no longer the only option for organizations.

Broader Adoption of Business Intelligence

BI was once the preserve of large companies that had the financial resources to fund complex projects, and the in-house knowledge to plan, implement, and maintain the required technologies.

As BI technologies have become much less expensive and more straightforward to implement, more and more organizations have started to use them. Furthermore, the availability of BI solutions in cloud services such as Microsoft Azure® has lowered the barrier to entry for smaller organizations by removing the need to purchase and maintain expensive hardware.

Self-Service Business Intelligence

Traditionally, information workers have had to rely on IT departments and technical users not only to manage the infrastructure, but also to oversee the creation of reports. However, in recent years, the trend has been to move away from this “IT-managed model” toward a “self-service BI model,” which enables information workers to directly create and manage the reports that they need. Business users are increasingly familiar with technology and adept at using it to perform their jobs. Although complex report authoring and delivery remains a task that is best suited to IT and BI professionals, many organizations are adopting self-service reporting solutions that take advantage of the available business data and IT

- Broader adoption of business intelligence
 - Lower barrier to entry enables more organizations to implement business intelligence solutions
- Self-service business intelligence
 - Creation and management of reports by information workers
- Out-of-the-box solutions
 - Quicker to implement, providing a faster ROI

sophistication of users. This self-service reporting approach is of significant benefit in scenarios where businesses need to:

- Empower information workers to get the information that they need to make informed decisions without waiting for an IT professional to create a report.
- Supplement standard reports that the IT department create with custom reports for specific job roles or scenarios.
- Reduce IT workload by minimizing requests for custom reports, enabling IT professionals to focus on more strategic and operational tasks.
- Promote collaboration—when information workers can easily share the reports and other documents that they create, collaboration becomes much easier.

For self-service reporting to be effective, some initial work must be done to design and implement a suitable reporting infrastructure. After that is in place, users can benefit from the ability to customize the reports that they use without placing an additional burden on the IT department.

Out-of-the-Box Solutions

Another significant trend in BI is the availability of out-of-the-box solutions, which are all-in-one offerings that can accelerate the time that it takes to plan and implement a BI solution. Reducing planning and implementation times makes it possible to achieve a much faster ROI, but the upfront cost of these systems can be a barrier for some organizations.

Introduction to Data Models

Most corporate BI solutions include analytical data models that provide information workers with a way to “slice and dice” data warehouse measures by aggregating them across the dimensions. Often, these analytical models are referred to as “cubes.” Technically, a cube is a specific organization of measures and dimensions in a multidimensional data model, but the word “cube” is commonly used as a generic term referring to any data model that enables users to aggregate measures by business entities.

- Data models are also referred to as cubes
- Benefits of data models include:
 - They present data in a more user-friendly way
 - You can change the underlying database without affecting the model
 - Data models can include custom logic that adds business value
 - Data models aggregate data for improved performance
 - Well-established standards enable the use of a wide range of tools

Benefits of Analytical Data Models

It is possible to create BI solutions that support reporting and analysis directly from tables or views in a database or data warehouse. However, in most scenarios, creating a separate analytical data model layer results in the following benefits:

- The data model abstracts the underlying data warehouse tables, which enables you to create models that reflect how business users perceive the business entities and measures, regardless of the data warehouse table schema. If necessary, you can modify or expand the underlying data warehouse without affecting the data model that is used by business users for analysis.
- The data model reflects the users’ view of the business, so data analysis is easier for information workers who have little or no understanding of database schema design. You can use meaningful names for tables and fields, and define hierarchies based on attributes in dimension tables that make the data more intuitive for business users.

- You can add custom logic to a data model that increases business value when analyzing the data. For example, you can define KPIs that make it easier to compare actual business measures with targets.
- Although the SQL Server database engine can provide extremely fast query performance, a data warehouse typically contains a massive volume of data. Most analyses involve aggregating measures across multiple dimensions, so the processing overhead for complex queries can result in unacceptably slow response times, especially when many users access the data concurrently. A data model typically preaggregates the data, which provides vastly superior performance for analytical queries.
- Data models are a common feature in BI solutions and there are well-established standards. By creating a data model, you can expose analytical data through a standard interface to be consumed by client applications such as Excel or third-party analytical tools.



Note: In addition to using data models as a source for reporting and analysis, you can also create analytical reports directly from the data warehouse or a departmental data mart. This enables you to express queries by using Transact-SQL, which may be more familiar to a report developer than a data modeling query language such as Multidimensional Expressions (MDX) or Data Analysis Expressions (DAX).

Business Intelligence Projects

A data model is only part of a BI project. Other key components of a BI project might include:

- **A data warehouse.** A data warehouse is a database that acts as the central repository for the data that you use in analysis and reporting. Typically, a data warehouse has a specialized schema, such as a star or snowflake schema, and contains two types of data. There is “fact” data, which is typically the numerical data—such as sales data—that you want to analyze, and “dimension” data—such as date or customer data—which you use to filter the fact data.
- **Extract, Transform, and Load (ETL) processes.** ETL is the process of extracting data from data sources, applying transformations to convert the data into the required format, and then loading it into a data store such as a data warehouse. Source data can come from many sources, and might be in various formats, and it is usually necessary to perform regular data loads to keep a data warehouse up to date.
- **Reporting and analysis tools.** There are many different tools that you can use to create reports and analyze data, including Excel, Reporting Services, and Report Builder. The tools that end users will require can have a big influence on the design of the overall solution.

- Other components of a BI project might include:
 - A data warehouse
 - ETL processes
 - Reporting and analysis tools
- Project roles include:
 - Project manager
 - BI solution architect
 - Data modeler
 - Database administrator
 - Infrastructure specialist
 - ETL developer
 - Report developer
 - Business users
 - Testers

Project Roles

Implementing a BI project can be a complex process, and typically requires the involvement of people from across the business. Most BI projects include some or all of the roles that are described below (note that the project role names might differ; it is the function of the role that is important):

- **A project manager.** Coordinates project tasks and schedules, ensuring that the project is completed on time and within budget.
- **A BI solution architect.** Has overall responsibility for the technical design of the data warehousing solution.
- **A data modeler.** Designs the data warehouse schema and analytical data models.
- **A database administrator.** Designs the physical architecture and configuration of the data warehouse database. Database administrators who have responsibility for data sources that are used in the data warehousing solution must also be involved in the project to provide access to the data sources that the ETL process uses.
- **An infrastructure specialist.** Implements the server and network infrastructure for the data warehousing solution.
- **An ETL developer.** Builds the ETL workflow for the data warehousing solution.
- **A report developer.** Creates the reporting elements of the BI solution.
- **Business users.** Provide requirements and help to prioritize the business questions that the data warehousing solution will answer. Often, the team includes a business analyst as a full-time member to help interpret the business questions and ensure that the solution design meets users' needs.
- **Testers.** Verify the business and operational functionality of the solution as it is developed.



Note: The list in this topic is not exhaustive, but represents roles that must be performed. In some cases, multiple roles may be performed by one person, although, in general, you should avoid having testers validate their own development work.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? A data model, which is created by using a technology such as SQL Server 2016 Analysis Services, is a required component of all BI projects.	

Lesson 2

Microsoft Business Intelligence Platform

Microsoft products are used to provide IT infrastructure for organizations all over the world. Therefore, it makes sense for many of these organizations to consider using the Microsoft BI platform to benefit from the close integration and common infrastructure capabilities of the various products that can help to deliver an enterprise BI solution. As a Microsoft BI professional, you need to know which products can be used to implement the various elements of a BI solution, and how those products integrate with each other.

Lesson Objectives

After completing this lesson, you will be able to describe:

- The role of SQL Server in a BI solution.
- The role of SharePoint Server in a BI solution.
- The role of Microsoft Office applications in a BI solution.
- The types of analytical data models that SQL Server 2016 Analysis Services supports.
- The installation options for SQL Server 2016 Analysis Services.
- Some considerations for SharePoint integration.

SQL Server

SQL Server 2016 provides the core data services for a BI solution. These services include:

- The SQL Server database engine, which is used for application databases, operations databases, and the data warehouse throughout the BI solution.
- SQL Server Integration Services, which is used as the primary platform for ETL processes.
- Data Quality Services, which provides data cleansing and matching capabilities.
- Master Data Services, which provides master data management capabilities.
- SQL Server Analysis Services, which provides a storage and query processing engine for multidimensional and tabular data models.
- SQL Server Reporting Services, which provides a platform for publishing and delivering reports that users can consume through a native web-based interface or have delivered by subscriptions.

- SQL Server 2016 includes core data services:
 - Database engine
 - SQL Server Integration Services
 - Data Quality Services
 - Master Data Services
 - SQL Server Analysis Services
 - SQL Server Reporting Services
- SQL Server 2016 Editions include:
 - SQL Server 2016 Enterprise Edition
 - SQL Server 2016 Business Intelligence Edition
 - SQL Server 2016 Standard Edition

SQL Server 2016 Editions

SQL Server 2016 is available in the following core editions:

- **SQL Server 2016 Enterprise.** You should use this edition for data warehouses and BI solutions that require advanced SQL Server Integration Services features, such as fuzzy logic and change data capture (CDC) components.

- **SQL Server 2016 Business Intelligence.** You should use this edition for servers that are hosting SQL Server Integration Services, Data Quality Services, and Master Data Services. You should also use this edition for solutions in SQL Server Reporting Services and SQL Server Analysis Services that require more than 16 processor cores, or if you need to support tabular data models, PowerPivot for SharePoint, Power View for SharePoint, or advanced data mining.
- **SQL Server 2016 Standard.** You can use this edition for solutions that require basic reporting in SQL Server Reporting Services, multidimensional models in SQL Server Analysis Services, and basic data mining.



Note: SQL Server 2016 is also available in Web and Express editions, but these are generally not appropriate for BI solutions.

SharePoint Server

SharePoint Server 2013 provides enterprise information-sharing services through collaborative websites. SharePoint Server provides the following BI capabilities:

- **Excel Services.** Users can view and interact with Excel workbooks that are shared in a SharePoint document library through a web browser. This includes workbooks that use data connections to query data in a data warehouse or SQL Server Analysis Services data model.
- **PowerPivot Services.** Users can share and interact with Excel workbooks that contain a PowerPivot tabular data model. This enables business users to create and share their own analytical data models.
- **Integration with SQL Server Reporting Services.** You can deliver and manage reports and data alerts through SharePoint document libraries instead of the native Report Manager interface that SQL Server Reporting Services provides.
- **Power View.** Power View is an interactive data visualization technology through which users can graphically explore a data model in a web browser.
- **PerformancePoint Services.** PerformancePoint Services enables BI developers to create dashboards and scorecards that deliver KPIs and reports through a SharePoint site.

- SharePoint Server 2013 enables enterprise information-sharing services
 - Excel Services
 - PowerPivot Services
 - Integration with SQL Server Reporting Services
 - Power View
 - PerformancePoint Services

Office Applications

Microsoft Office 2016 and Microsoft Office 365™ provide productivity applications that business users can use to consume and interact with BI data. These applications include:

- **Microsoft Excel.** Excel is the most commonly used data analysis tool in the world, and can be used to:
 - Import data from a data warehouse and use it to create charts and reports.
 - Create interactive PivotTables and PivotCharts from analytical data models in SQL Server Analysis Services or PowerPivot.
 - Create PowerPivot workbooks that contain tabular data models.
 - Create Power View visualizations from tabular models.
- **Microsoft Word.** Word is a document-authoring tool. In a BI scenario, users can export SQL Server Reporting Services reports in Word format and use the editing and reviewing tools in Word to enhance them.
- **Microsoft PowerPoint®.** PowerPoint is a widely used presentation tool. Users can save Power View visualizations as PowerPoint presentations, and present business data in a dynamic, interactive format.
- **Microsoft Visio®.** Visio is a diagramming tool that can be used to document database schemas and to visualize data mining analyses.

- Microsoft Office 2016 and Office 365 productivity applications used for business intelligence:
 - Microsoft Excel
 - Charts and reports
 - PivotTables and PivotCharts
 - PowerPivot workbooks that contain tabular data models
 - Power View visualizations from tabular models
 - Microsoft Word
 - Enhance SQL Server Reporting Services reports
 - Microsoft PowerPoint
 - Use Power View visualizations in presentations
 - Microsoft Visio
 - Document database schemas
 - Visualize data mining analyses

Analytical Data Models

SQL Server 2016 includes two types of data model, multidimensional data models and tabular data models:

- **Multidimensional data models.** Multidimensional data models have been supported in every version of SQL Server Analysis Services since the release of SQL Server 7.0. You can use a multidimensional data model to create a SQL Server Analysis Services database that contains one or more cubes, each providing aggregations of measures in measure groups across multiple dimensions.
- **Tabular data models.** Tabular data models were first introduced in Excel with PowerPivot in SQL Server 2008 R2 and extended to SQL Server Analysis Services in SQL Server 2012. For a user who is performing analysis, tabular models provide similar functionality to a multidimensional model. In many cases, the two models are indistinguishable from one another. However, for BI developers, tabular models are often faster and easier to create because they do not require as much online analytical processing (OLAP) modeling knowledge as multidimensional models. Tabular data models

- Multidimensional data models
- Tabular data models
- Considerations for choosing a data model include:
 - Hardware
 - Calculations
 - Many-to-many relationships
 - Data sources
 - Available skills
 - Client tools

are based on relationships between multiple tables of data, so they are much easier to understand for professionals who are used to working with relational databases.



Note: In SQL Server 2016, analytical data models are sometimes also called BI Semantic Models.

Selecting a Data Model

Many factors are involved in selecting a data model for a specific BI scenario. As you learn about the data models in this course, you will understand more about the differences between the two types of model, and when you might choose to use them. In general terms, the following factors will influence the choice of data model:

- **Hardware.** Tabular data models use in-memory storage, so they require relatively large amounts of memory.
- **Calculations.** Data models usually include calculations—you can implement these in tabular data models by using the DAX language and in multidimensional models by using the MDX language. Some calculations are easier to implement in DAX, others in MDX.
- **Many-to-many relationships.** Tabular data models do not support the creation of many-to-many relationships, although you can use DAX to model this type of relationship.
- **Data sources.** Tabular data models support a wider range of data sources than multidimensional data models, including text files and Excel.
- **Available skills.** As with many technical decisions, a frequent deciding factor is the knowledge and skills of the developers who will implement the data model.
- **Client tools.** Most Microsoft tools can consume data from either type of data model. However, although you can use Power View in Excel against tabular models, you can only use Power View against multidimensional models through a SharePoint portal that uses the Microsoft SQL Server 2016 Reporting Services add-in for SharePoint Server.

Installation Options for SQL Server 2016 Analysis Services

SQL Server Analysis Services is a feature of SQL Server 2016 that you install separately from the SQL Server database engine. When you install SQL Server Analysis Services, you must select the mode in which the SQL Server Analysis Services instance will run. There are three installation modes:

- **Multidimensional and data mining mode.** This mode enables the creation of multidimensional data models, and additionally includes data mining functionality. Data mining is the process of analyzing data to identify trends, relationships, and patterns, and using this to predict future behavior.
- **Tabular mode.** Tabular mode enables the creation of tabular data models.

- Analysis Services has three installation modes:
 - Multidimensional and data mining mode, for creating multidimensional data models and performing data mining analysis
 - Tabular mode, for creating tabular data models
 - PowerPivot mode, for SharePoint integration
- To use multiple Analysis Services modes, install each mode as a separate instance

- **PowerPivot mode.** PowerPivot mode, which is also known as SharePoint integrated mode, integrates SQL Server Analysis Services with a SharePoint server instance to make features of SQL Server Analysis Services available through a SharePoint portal.

You cannot install two modes of SQL Server Analysis Services as part of the same instance, and you cannot switch modes after installation. For example, if you need to create both tabular data models and multidimensional data models, you should install two separate instances, each of which uses the appropriate mode. As with instances of the database engine, it is possible to install multiple instances of SQL Server Analysis Services on the same server running Windows Server®.

Considerations for SharePoint Integration

SharePoint Server 2013 enables the sharing and collaboration of reports, and is an important component of many BI solutions. By installing SQL Server Analysis Services in PowerPivot mode, you can take advantage of SharePoint to provide multiple benefits, including:

- A single location for PowerPivot reports, which users access through the easy-to-use PowerPivot Gallery.
- Reports remain within the scope of management—for example, by being included in backup runs. The PowerPivot Management Dashboard provides a single interface that administrators can use to manage the PowerPivot mode deployment—for example, by monitoring resource usage, how often reports are used, and data refresh successes and failures.

- SharePoint architecture is tiered:
 - Web front-end tier
 - Application tier
 - Data tier
- Available topologies for SharePoint deployments include:
 - Single server
 - Scale-out
 - High availability
- Use the SQL Server 2016 installation media to install Analysis Services in PowerPivot mode
 - Install the Power Pivot for SharePoint add-in to add extra functionality to SharePoint farm servers

SharePoint Architecture

When planning SharePoint integration, there are three tiers of the SharePoint architecture to consider:

- **Web front-end tier.** One or more servers that accept requests for a SharePoint application or service, and direct the request to the appropriate application server.
- **Application tier.** One or more servers that host the service applications in the SharePoint Server infrastructure.
- **Data tier.** A SQL Server instance that hosts SharePoint databases. It is common to use clustering to ensure high availability of the SharePoint databases.

SharePoint Business Intelligence Topologies

By changing where you host each tier, you can deploy SharePoint in one of three topologies:

- **Single server.** In this topology, you host all of the SharePoint Server architecture layers on a single server running Windows Server. The main benefit of this model is that it minimizes the licensing cost for the solution. Typically, you use this type of configuration in development environments or training environments because it is the easiest way to set up a SharePoint farm. However, because all three layers run on the same server running Windows Server and share the same hardware, there can be increased contention for resources. If this affects the performance of business reports, consider implementing a scale-out solution.

- **Scale-out.** In a scale-out topology, you separate each of the SharePoint Server architecture layers onto different servers running Windows Server. Scaling out a SharePoint farm distributes the workload across multiple servers, which reduces resource contention and improves throughput. However, this topology comes with an additional licensing cost because it uses multiple servers. Furthermore, it requires greater preparation to ensure that security works seamlessly across the SharePoint farm. This topology offers no resilience if one of the servers shuts down. You should consider a scale-out topology if performance is important, but there is no requirement for resilience.
- **High availability.** The high-availability topology involves separating the SharePoint architecture layers across servers running Windows Server, and then duplicating each layer on another server. This is the most expensive topology to implement, but it provides load balancing and high availability across the entire SharePoint farm. The infrastructure preparation is similar to that of creating a scale-out architecture, and also requires a network load balancer to distribute incoming requests to the first available web front-end server.

Installing SQL Server Analysis Services in PowerPivot Mode

After setting up the SharePoint 2013 farm, you can use the SQL Server 2016 installation media to install PowerPivot for SharePoint on the application servers in the farm. This creates a PowerPivot mode instance of SQL Server Analysis Services on the application server, which provides a workspace for working with PowerPivot files. After the installation is complete, you can download the PowerPivot for SharePoint add-in file **spPowerPivot.msi**, and install it on each server in the SharePoint Server farm. The PowerPivot for SharePoint add-in installs data providers, the PowerPivot configuration tool, the PowerPivot Gallery, and components that are required for data refresh.

Check Your Knowledge

Question	
<p>You are part of a team that is in the early stages of planning a BI project. You have identified that a key requirement for the project will be the ability to use historical data to make predictions about future scenarios, which the marketing department will use to help to create new campaigns. Which type of SQL Server Analysis Services instance will you need to include so that you can meet this requirement?</p>	
<p>Select the correct answer.</p>	
<input type="checkbox"/>	An instance of SQL Server Analysis Services in tabular mode.
<input type="checkbox"/>	An instance of SQL Server Analysis Services in multidimensional and data mining mode.
<input type="checkbox"/>	An instance of SQL Server Analysis Services in PowerPivot mode.

Lab: Exploring a Business Intelligence Solution

Scenario

Adventure Works Cycles is a multinational corporation that manufactures and sells bicycles and cycling accessories. The company sells its products through an international network of resellers and, in recent years, has developed a direct sales channel through an e-commerce website.

The company is financially sound and has a strong order book, but sales volumes have remained relatively static for the past few years. Senior management is under pressure from shareholders to develop a growth strategy that will drive increased revenue and profit. Management believes that a key factor in its growth strategy is investment in technology that improves collaboration between the various divisions of the company, and enables them to track and share key business performance metrics.

Objectives

After completing this lab, you will have:

- Explored a data warehouse by using a database diagram and by issuing Transact-SQL queries.
- Performed basic data analysis by using Excel with a data model as a data source.

Estimated Time: 45 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Exploring a Data Warehouse

Scenario

Adventure Works Cycles has a data warehouse containing data that relates to Internet and reseller sales. This data warehouse will form the foundation for an enterprise BI solution to include SQL Server Analysis Services data models and a SQL Server Reporting Services solution for reports.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Explore a Data Warehouse Schema
3. Query a Data Warehouse

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab01\Starter folder as Administrator.

► Task 2: Explore a Data Warehouse Schema

1. Use SQL Server Management Studio to connect to the **MIA-SQL** SQL Server instance.
2. Create a database diagram for the **AdventureWorksDW** database, and note the following details about its schema:
 - The **FactInternetSales** table stores details of sales orders that are made through the Adventure Works website. The table is related to the **DimCustomer** table, which contains details of the customers who have placed orders.

- The **FactResellerSales** table stores details of sales orders that are made to resellers. The table is related to the **DimReseller** table, which contains details of the resellers who have placed orders.
 - The **DimDate** table stores values for dates, including details of the calendar and fiscal periods in which individual dates occur.
 - The **FactInternetSales** and **FactResellerSales** tables are related to the **DimDate** table by multiple fields. These fields represent the order date (the date when the order was placed), the due date (the date when the order was expected to be in stock), and the ship date (the date when the order was shipped to the customer).
 - Both the **DimCustomer** and **DimReseller** tables are related to the **DimGeography** table, which stores details of geographical locations. The **DimSalesTerritory** table is also related to **DimGeography**.
 - The **DimProduct** table contains details of products, and is related to the **DimProductSubcategory** table, which is in turn related to the **DimProductCategory** table.
 - Many tables include multiple language values for the same data value, for example, the **DimDate** table stores the English, French, and Spanish words for each month name.
 - The **DimEmployee** table is related to itself to represent the fact that each employee has a manager, who is also an employee.
3. Save the diagram as **AdventureWorksDW Schema**.

► Task 3: Query a Data Warehouse

1. Open the **Query DW.sql** script file in the D:\Labfiles\Lab01\Starter folder.
2. Execute the query under the comment **Internet sales by year and month**, and note the following details:
 - The data warehouse contains historical Internet sales orders from July 2005 to July 2008. Reseller sales in the data warehouse are also recorded for this time period.
 - You can use the **MonthNumberOfYear** column in the **DimDate** table to sort month names into chronological order—without this field, it would be difficult (though not impossible) for reporting clients to sort months other than alphabetically. A similar field named **DayNumberOfWeek** can be used to sort weekday names into chronological order.
3. Execute the query under the comment **Geographical reseller sales**, and note the following details:
 - In 2005, Adventure Works only sold to resellers in the United States and Canada.
 - In 2006, this was expanded to include France and the United Kingdom.
 - In 2007, resellers in Australia and Germany were added.
 - By contrast, Adventure Works has sold directly to Internet customers in all of these regions since 2005.
4. Execute the query under the comment **Sales by product category**, and note the following details:
 - Adventure Works sells four categories of product: Accessories, Bikes, Clothing, and Components.
 - Components are only sold to resellers, not to Internet customers.
 - Accessories were not sold to Internet customers until 2007.

Results: After completing this exercise, you will have:

Used a database diagram to explore a data warehouse schema.

Used Transact-SQL queries to explore the data in a data warehouse.

Exercise 2: Exploring a Data Model

Scenario

Having examined the data warehouse, you will now explore a SQL Server Analysis Services data model that is based on the data warehouse tables.

The main tasks for this exercise are as follows:

1. View a SQL Server Analysis Services Database
2. Create a PivotTable in Excel
3. Filter the PivotTable

► Task 1: View a SQL Server Analysis Services Database

1. In SQL Server Management Studio, connect to the **MIA-SQL** instance of SQL Server Analysis Services.
2. Explore the **Adventure Works OLAP** database in the MIA-SQL SQL Server Analysis Services server. Note that it contains a data source named **Adventure Works Data Warehouse**, which connects to the **AdventureWorksDW** database that you explored in the previous exercise.
3. Browse the cube named **Internet Sales**, to view the **Sales Amount** measure by **Order Date – Fiscal Year**. Note that in Adventure Works, fiscal years run from July to June, so fiscal year 2006 represents the period from July 2005 to June 2006.
4. Close SQL Server Management Studio when you are finished.

► Task 2: Create a PivotTable in Excel

1. Create a new blank Excel workbook.
2. Import data from the **MIA-SQL** SQL Server Analysis Services server:
 - Select the **Internet Sales** cube in the **Adventure Works OLAP** database.
 - Import the data into a PivotTable report.
3. In the PivotTable, display the **Sales Amount** measure. Change the settings of this value field to format it by using the **Accounting** format.
4. Add the **Order Date.Calendar Date** hierarchy to the rows of the PivotTable, and explore the data by drilling down from years to semesters, semesters to quarters, quarters to months, and months to days.
5. Save the workbook as **Sales.xlsx** in the D:\Labfiles\Lab01\Starter folder.

► Task 3: Filter the PivotTable

1. Add a slicer based on the **English Product Category Name** field in the **Product Category** hierarchy to the PivotTable that you created in the previous task:
 - Note that the **Components** category is disabled because there are no sales of components to Internet customers.

2. Use the filter to show only sales for the **Accessories** category:
 - Note that no accessories were sold in 2005 or 2006.
3. Clear the filter, and then save and close the workbook.

Results: After completing this exercise, you will have used Excel to explore an analytical data model built on the data warehouse.

Question: Which tools have you used to view BI data? How do these tools differ and what are the advantages and disadvantages of these tools?

Module Review and Takeaways

In this module, you have learned about the common elements of a BI solution and how Microsoft software products can help you to implement them. You have also learned about the key roles and scoping tasks in a BI project.

Review Question(s)

Question: In your experience, what are the factors that affect the success or failure of a BI project?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 2

Creating Multidimensional Databases

Contents:

Module Overview	2-1
Lesson 1: Introduction to Multidimensional Analysis	2-2
Lesson 2: Data Sources and Data Source Views	2-8
Lesson 3: Cubes	2-12
Lesson 4: Overview of Cube Security	2-17
Lesson 5: Configuring SSAS	2-23
Lesson 6: Monitoring SSAS	2-28
Lab: Creating a Multidimensional Database	2-34
Module Review and Takeaways	2-39

Module Overview

You can use Microsoft® SQL Server® Analysis Services (SSAS) to build on Online Analytical Processing (OLAP) solutions that help you to quickly and efficiently obtain accurate business intelligence (BI) that improves business decision making—and helps to drive growth. SSAS multidimensional databases, which are also known as OLAP cubes, have several benefits including:

- Fast query response times because of a preaggregation of data and calculations. Queries against cubes are generally much quicker than the equivalent queries against relational databases.
- Multidimensional structures are logical and straightforward, and also provide an intuitive navigation path for end users that is easy to understand.
- Accessibility from a wide range of tools, such as Microsoft Excel®, Power View, and third-party tools.

This module introduces multidimensional databases and the core components of an OLAP cube.

Objectives

After completing this module, you will be able to:

- Describe the core components of a multidimensional database solution.
- Create data sources and data source views.
- Create and browse a multidimensional cube.
- Explain how to implement security in a multidimensional database.
- Configure memory and storage to optimize Analysis Services instances.
- Monitor Analysis Services for performance issues.

Lesson 1

Introduction to Multidimensional Analysis

Multidimensional analysis using OLAP involves the creation of OLAP cubes, data sources, data source views, dimensions, measures, and measure groups. This lesson explains the business problem that multidimensional data models address, and describes data sources, data source views, and the core components of a cube: dimensions, measures, and measure groups.

Lesson Objectives

After completing this lesson, you will be able to:

- Identify the business problems that multidimensional solutions can address.
- Describe the core concepts of multidimensional analysis.
- Describe the role of data sources in a multidimensional database.
- Describe the role of data source views in a multidimensional database.
- Describe the role of dimensions in a multidimensional database.
- Describe the role of measures and measure groups in a multidimensional database.
- Describe the role of cubes in a multidimensional database.

The Business Problem

Most database systems are designed for Online Transaction Processing (OLTP). In an OLTP system, many transactions—updates, inserts, and deletions—can happen per second and data is stored at the most granular level possible. This level of detail is required and is useful for many operations. However, in other scenarios, an OLTP system is not always the most appropriate solution.

For example, the fictional organization Adventure Works sells bicycles and associated accessories. The Adventure Works OLTP database records every item from every sale, and these updates to the database need to happen almost instantaneously. However, problems might emerge in situations that involve issuing complex queries, such as when the trading manager requires a broad overview of sales for the past month by product category. The Transact-SQL query needed to generate this data would involve filtering all sales to find only those that occurred in the past month, grouping them by product category, and then calculating the total sales by product group. Not only is this query potentially very slow, it will cause resource contention on the server; it also requires the locking of tables or other objects to prevent transactions simultaneously attempting to change the data that the query is reading. Contention and locking can negatively affect database performance, and compromise the database's primary role as a fast, transactional system.

- Most databases are designed for OLTP, and are not optimal for reporting and analyzing data
- Aggregated spanning queries are slow to generate
- Large queries can cause performance issues for transactional workloads
- Relational data models are not intuitive for business users

When considering using an OLTP system as a direct source of data for reporting and analysis, bear in mind the following points:

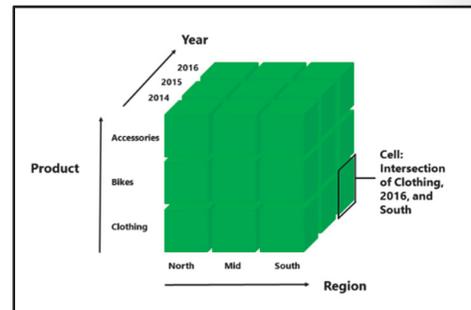
- Aggregated spanning queries are slow to generate.
- Large queries can cause performance issues for transactional workloads.
- Normalized relational data models are not intuitive for business users.

Core Concepts for Multidimensional Data Analysis

The multidimensional data model can appear to be quite conceptually challenging when you first encounter it, especially if you are used to working with relational databases. However, the basic concepts are easy to grasp if you have a clear understanding of the terminology.

Cubes, Measures, and Dimensions

In a multidimensional database, you create a structure called a cube to store measures, which are discrete, aggregated, numerical values, such as sales revenue totals or manufacturing costs. The cube stores measures in individual cells, and has different dimensions by which you can analyze the data. The dimensions are analogous to the height, width, and depth dimensions of a real, physical cube, except that they represent values such as date, region, or product instead. Each cell is a location in the cube that represents the intersection of the dimensions of the cube for a given set of dimension values. For example, you might select to analyze the sales data for the product category **Clothing** in the **South** region, in the year **2016**. The cell in the cube that represents the intersection of the dimensions for the values **Clothing**, **South**, and **2016** is easily located and contains the required value, already aggregated. The fact that data is preaggregated in a cube means that values do not have to be calculated when a user issues a query against the cube—this is much less resource intensive, and enables faster query response times. Cubes in multidimensional databases nearly always have many more dimensions than the three dimensions in a real, physical cube. This means that using the concept of a three-dimensional cube to understand multidimensional cubes is not always intuitive!



Drilling Down, Slicing, and Dicing

You can use dimensions to *drill down* through different layers of data by using a hierarchy. For example, you can initially view sales by year, then drill down to sales by month, and then by day. You can also filter the data by using *slicing* and *dicing* dimension values:

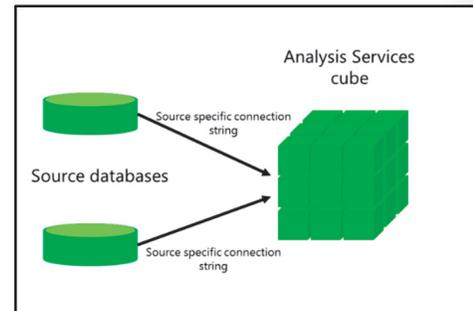
- **Slicing** involves using a member of one dimension to provide a slice of the cube. For example, you could slice a cube using a particular product, and view all sales of that product across all dates and customers.
- **Dicing** involves providing values for every dimension to locate a single value for a cube. For example, you could look for sales of a particular product, on a particular day, to a particular customer.

Aggregations

One key difference between cubes and relational systems—and between cubes and tabular data models—is the processing of the cube. The aggregations of measures are calculated when the cube is processed, not when the query runs. Consequently, when business users run queries, response times are extremely fast, because the complex processing is already complete.

Data Sources

A data source is an object that provides an SSAS database that has the information required to connect to a data source for the cubes it contains. Analysis Services can access data from one or more data source. Typically, the source database for an OLAP cube is a data warehouse or a departmental data mart, in which the tables have been denormalized to reflect a star or snowflake schema of fact and dimension tables. However, SSAS can use virtually any database as a data source, and abstract the underlying data model by using a data source view.



At a minimum, a data source includes an identifier, a name, and a connection string. The connection string used to access the source data includes the following information:

- The type of provider, for example MSOLAP or OLE DB.
- Other properties that the provider supports or requires, such as the initial catalog (database) to connect to, or the server name. The actual property settings for particular data source objects vary according to the provider.



For more information about connection string properties in Analysis Services, see the article [Connection String Properties \(Analysis Services\) in the SQL Server 2016 Technical Documentation](http://aka.ms/H5sor4):

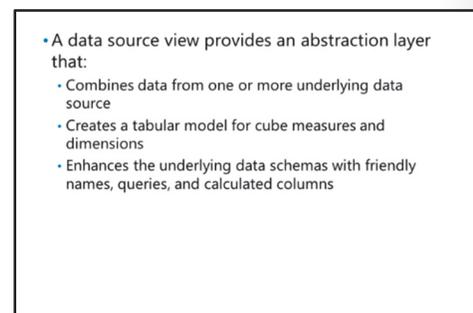
<http://aka.ms/H5sor4>

Data Source Views

A data source view abstracts the underlying data source and defines the specific tables that should be retrieved from the data source. It can also change the presentation of those tables and their columns to make it more user-friendly for the end users who will work with the data. For example, the column names in database tables might be meaningful to database administrators and developers who use the database regularly, but they might be difficult to read for other users who are not as familiar with the database. For example, a column name such as **Orderlineitems** lacks space characters and capitalization, making it relatively difficult to understand. In a data source view, you can represent columns by using names that are easier to read.

A data source view contains the following items:

- A name and a description.
- A definition of any subset of the schema retrieved from one or more data sources, which could involve the whole schema, including the following:
 - Table names



- Column names
- Data types
- Nullability
- Column lengths
- Primary keys
- Primary key/foreign key relationships
- Enhancements to the schema from the underlying data sources, including the following:
 - Friendly names for tables, views, and columns.
 - Named queries that return columns from one or more data sources (that show as tables in the schema).
 - Named calculations that return columns from a data source (that show as columns in tables or views).
 - Logical primary keys (needed if a primary key is not defined in the underlying table, or is not included in the view or named query).
 - Logical primary key/foreign key relationships between tables, views, and named queries.

You can also create multiple diagrams showing different subsets of the data source view. Unlike different data source views, different diagrams reference the same schema. Therefore, any changes you make in one diagram apply to all the others in the data source view. Diagrams are useful when you create multiple cubes within the same solution.

Dimensions

As described earlier, dimensions provide the context for the facts stored in the OLAP cube. Dimensions include key values and hierarchies.

Key Values

Each dimension includes a key attribute that matches values in the fact table. The key attribute also enables SSAS to join records to related data in the fact table, in the same way that tables use columns to match values in Transact-SQL statements. For example, a dimension called

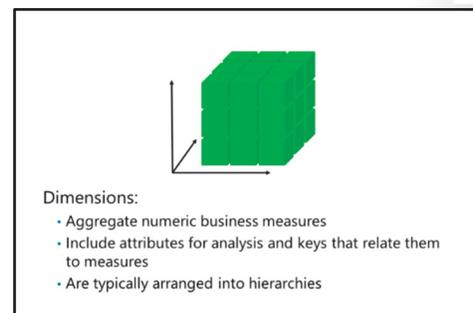
Product Category might include a

ProductCategoryID key value, which also exists as

a column in the fact table. Matching the values in the dimension and the fact table enables Analysis Services to aggregate the measure values correctly, according to the dimension that the user specifies.

Dimension Hierarchies

Analysis Services dimensions are hierarchical, not flat. For example, a **Time** dimension typically has years, months, and days; a **Geography** dimension typically includes country, state, and town. A dimension can have more than one hierarchy, meaning you can drill down in different ways. For example, one hierarchy might include year, month, and day, and another might include year, week, and day.



Dimension hierarchies can also be unbalanced and ragged. An *unbalanced dimension hierarchy* does not have the same number of levels in every branch. For example, in an **Employee** dimension, the Sales Director reports to the Chief Executive Officer (CEO) and has three levels of staff below her. The Executive Assistant also reports to the CEO but has no levels below him. A *ragged dimension hierarchy* is a particular type of unbalanced dimension where there are missing or duplicated levels in the hierarchy. For example, in a **Geography** dimension hierarchy that contains Continent, Country or Region, State, and City, the city state of Monaco would either have the values of Europe, Monaco, Monaco, Monaco—or the values of Europe, Monaco, NULL, NULL.

Measures and Measure Groups

Measures are the facts aggregated at each intersection of dimension members. They are typically numeric and additive, and are aggregated by using a SUM calculation, although not all measures meet these criteria. For example, you might use a COUNT aggregation to create a measure that shows the number of orders placed.

Every cube includes at least one measure, but most cubes contain many more to reflect the different values by which an organization needs to assess its performance. The values in measures are frequently derived from a column in a table in a source database, such as a fact table in a data warehouse. However, you can also create measures as calculated members by using the Multidimensional Expressions (MDX) language. You can use this to apply calculations to measure values or use multiple columns from a fact table in a measure expression.

- A measure is a column that contains data that can be aggregated
- Usually a column in a fact table
- Can be defined by using a measure expression
- Measures are grouped into measure groups, based on their underlying fact tables



Note: You will learn more about MDX in the *Introduction to MDX* module in this course.

Measure Groups

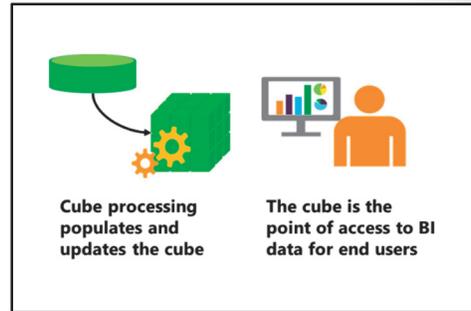
Measures are grouped, based on their underlying fact tables. Measure groups are used to define the relationship between dimensions and measures. When a dimension or a measure group is added to the cube, SSAS will examine the relationship that exists between the fact table and the dimension table in the data source and automatically set the **Measure Group** information in the **Dimension Usage** tab of the Cube Designer. You can then modify this to your requirements.

Cubes

A cube is the unit that defines dimensions and measures and is the user's point of access to the multidimensional data. The term "cube" is used because it describes the multidimensional nature of the data, as opposed to the one-dimensional or two-dimensional types of relational tables.

However, almost all cubes have more than three dimensions. You can use cubes to bring together all the functionality of dimensions and measures to display them in a user-friendly interface, such as an Excel PivotTable table. You can use the PivotTable interface with SSAS to display aggregated measure data, and slice and dice it by using dimension members.

After creating a cube, you must process it. This populates the cube with data and creates the measures and calculations specified in the cube design. Periodically, you should also process cubes to bring the data they contain up to date, and to reflect any changes. Cubes also represent an administrative unit within Analysis Services, and include settings that define storage, aggregation levels, and processing frequency for the cube.



Categorize Activity

Match the definition to the correct Analysis Services component.

Items	
1	An object that filters a data source and specifies a subset of the tables to include.
2	An aggregated numerical value, such as a sales total.
3	An object that contains data, such as product or customer information, that provides context for data analysis.

Category 1	Category 2	Category 3
Data source view	Measure	Dimension

MCT USE ONLY. STUDENT USE PROHIBITED

Lesson 2

Data Sources and Data Source Views

OLAP and data mining projects in SSAS are designed based on a logical data model of related tables, views, and queries from one or more data sources. Data sources are defined as data source objects in Analysis Services, and logical data models are defined as data source views. You use data source views to filter the data from a data source so that the cube includes only a subset of the data that populates a large data warehouse.

Lesson Objectives

After completing this lesson, you will be able to:

- Create a data source.
- Create a data source view.
- Describe how to modify a data source view.

Creating a Data Source

You can use the Data Source Wizard in Visual Studio® SQL Server Data Tools to define one or more data sources for an Analysis Services project. Data sources use a Connection object that defines the connection string. In the first step of the Data Source Wizard, you must define the Connection object, or use an existing Connection object.

The default provider for a new connection is the Native OLE DB\SQL Server Native Client 11.0 provider. Other supported providers include:

- .NET providers for SQLClient and OracleClient
- Native OLE DB providers for:
 - Analysis Services
 - Oracle
 - SQL Server Native Client 10.0

If you have an existing data source defined in an SSAS database or project, and want to create a new data source object that connects to the same underlying data source, you can copy the properties of the first data source object into a new version.

- Use the Data Source Wizard in Visual Studio SQL Server Data Tools to create data sources
- Data source definitions include Connection objects that support a range of data providers

Creating a Data Source View

Data source views provide an abstraction layer over a data source, which reduces the complexity of the schema that you will use in Analysis Services. You can create a data source view by using the Data Source View Wizard in SQL Server Data Tools. When creating a view, you must specify the following:

- The name of the data source that the view will use.
- The tables or views that you want to include in the data source view.
- A name for the data source view.

- Provides an abstraction layer over the data source to reduce schema complexity
- Is defined using the Data Source View Wizard
- Enables simple binding from cubes and dimensions

SSAS design tools use data source views to maintain a cache of relational metadata, and take advantage of some of the annotations within a data source view. Advantages of this approach include:

- A data source view makes available only tables that OLAP and data mining objects require, by describing a subset of tables and views in a data source.
- A data source view handles the layout of tables, filters, SQL expressions, relationships, and other complexities of the schema. Therefore, a data source view enables simple bindings by SSAS cubes, dimensions, and mining models to the tables and columns in the data source view.

Working with Data Source Views

After you have created a data source view, you can modify it to make the data more user-friendly. Often, underlying data sources use naming conventions to help developers understand the schema. Also, the underlying data does not typically store additional formatting information. By using the Data Source View Designer, you can make modifications to a data source view without modifying the underlying source data or database schemas. You can then present user-friendly names and apply formatting without making any changes to the source database systems. For example, you can:

- Use the **FriendlyName** property to specify a column name from a table or view that is easier for users to understand—or is more relevant to the subject area.
- Create a named query and named calculations that mean you can extend the relational schema of existing tables in a data source view, without modifying the underlying data source.
- Create logical primary keys and relationships for improved performance.
- Create diagrams to reduce the visual clutter when you only want to view a subset of the tables in the data source view.

- By modifying data source views, you can achieve the following benefits:
 - Create user-friendly names
 - Use named queries
 - Create relationships to improve performance
 - Create diagrams to view subsets of the tables in a data source view

Demonstration: Creating a Data Source and a Data Source View

In this demonstration, you will see how to:

- Create a new Analysis Services project.
- Create a data source.
- Create a data source view.
- Modify a data source view.

Demonstration Steps

Create an Analysis Services Project

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod02 folder, run **Setup.cmd** as Administrator.
3. Start Visual Studio, and then click **New Project**.
4. In the **New Project** dialog box, in the **Business Intelligence** templates section, create a new **Analysis Services Multidimensional and Data Mining Project** named **Demo** in the D:\Demofiles\Mod02 folder.

Create a Data Source

1. In Solution Explorer, right-click **Data Sources**, and then click **New Data Source**.
2. On the **Welcome to the Data Source Wizard** page, click **Next**.
3. On the **Select how to define the connection** page, click **New**.
4. In the **Server name** field, type **localhost**, in the **Select or enter a database name** list, click **AdventureWorksDW**, click **OK**, and then click **Next**.
5. On the **Impersonation Information** page, click **Use a Specific Windows user name and password**, enter the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **Next**.
6. In the **Data source name** field, type **Adventure Works Demo DS**, and then click **Finish**.

Create a Data Source View

1. In Solution Explorer, right-click **Data Source Views**, and then click **New Data Source View**.
2. On the **Welcome to the Data Source View Wizard** page, click **Next**.
3. On the **Select a Data Source** page, ensure that **Adventure Works Demo DS** is selected, and then click **Next**.
4. On the **Select Tables and Views** page, select the **DimDate**, **DimProduct**, **DimProductCategory**, **DimProductSubcategory**, **DimReseller**, **DimSalesTerritory** and **FactResellerSales** tables. Add these tables to **Included objects**, and then click **Next**.
5. On the **Completing the Wizard** page, in the **Name** field, type **Adventure Works Demo DSV**, and then click **Finish**.

Modify a Data Source View

1. If it is not already open, in Solution Explorer, right-click **Adventure Works Demo DSV**, and then click **Open**.
2. In Data Source Designer, click the table name of the **FactResellerSales** table and press F4.
3. In the Properties pane, change the **FriendlyName** property to **Reseller Sales**.
4. Repeat the previous steps to rename the **DimSalesTerritory** table to **Sales Territory**.
5. Right-click the **DimProduct** table, and then click **New Named Calculation**.
6. In the **Column name** field, type **Product Size**.
7. In the **Expression** field, type **[Size] + ' ' + [SizeUnitMeasureCode]**, and then click **OK**.
8. Right-click the **DimProduct** table and click **Explore Data**, then scroll down until you see a value in the **Product Size** column (the last column in the table) that shows the size and measurement unit.
9. Close the Explore DimProduct Table window then, on the **File** menu, click **Save All**.
10. Keep Visual Studio open. You will return to it in a later demonstration.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? You are investigating data sources for a planned BI project that includes Analysis Services multidimensional cubes. You have identified an Oracle database that contains important data, but a colleague says that you cannot import data from an Oracle database into an Analysis Services cube.	

Lesson 3

Cubes

A cube is a multidimensional structure containing dimensions and measures. Dimensions define the structure of the cube and measures provide the numerical values of interest to the end user. As a logical structure, a cube enables a client application to retrieve values as if cells in the cube define every possible summarized value. Cubes consume the data that you make available to them through data sources and data source views.

Lesson Objectives

This lesson describes how to create and modify cubes, and includes some considerations for cube design. After completing this lesson, you will be able to:

- Create and configure a multidimensional cube.
- Describe the considerations for using time dimensions in a cube.
- Browse a multidimensional cube.

Creating a Cube

To create a cube in Analysis Services, you can use the Cube Wizard in an Analysis Services project in SQL Server Data Tools. You can launch the wizard by right-clicking the **Cubes** node in a project in Solution Explorer, and then clicking **New Cube**.

The Cube Wizard guides you through the process of creating a cube, and includes various configuration options:

- You can create a cube by choosing the relevant data source and data source view, and then selecting the tables from the view that you want to include in the cube.
- You can specify the measures from the tables that you want to use.
- When you create the cube, you can add existing dimensions, or create new dimensions that structure the cube.
- You can also create dimensions separately, by using the Dimension Wizard, and then add them to a cube.
- Instead of using an existing data source and data source view, you can build the cube without using a data source and subsequently generate the database schema. Another option is to create the cube, and enable SSAS to create the underlying schema supporting the cube.
- You can use the Cube Wizard to automatically build attributes and hierarchies, or you can choose to define them in the Cube Designer later.

- Create cubes by using the Cube Wizard
 - Use existing data sources and data source views
 - Specify the tables and measures to include
 - Specify existing dimensions to include
 - Alternatively, create an empty cube
 - Add the cube schema later
 - Use the Dimension Designer to create dimensions and add them to the cube
- Modify and configure cubes by using the Cube Designer

Options for modifying cubes

Although the Cube Wizard creates the cube, many of its properties are available for configuration in the Cube Designer interface in SQL Server Data Tools. You can use the Cube Designer to:

- View the cube's properties and its objects.
- Browse cube data.
- Modify cube structure, dimension usage, and calculations.
- Create key performance indicators (KPIs) and actions.
- Configure cube partitioning and storage.
- Create aggregations.
- Create perspectives on cube data.
- Add translations to localize cube data.

Time Dimensions

One of the most important and commonly-used types of dimension is the time dimension, which includes time and date values that users can use to filter the data in the cube. Although most cubes include a time dimension, some cubes use different factors instead—such as a unit of work or project—to delineate data.

Hierarchies in Time Dimensions

Like many dimensions, time dimensions include hierarchical data; for example, days roll up into months, and months roll up into years. Depending on the business requirements of the solution, a time dimension might include multiple hierarchies. For example, in addition to the days, months, and years hierarchy, you might include a hierarchy with levels for days, weeks, and years. Together, these two hierarchies would enable users to explore the data by weeks or by months. However, a single hierarchy that included years, months, weeks, and days would not work, because weeks do not fit into months in a regular way. You can create multiple time dimensions, or just a single time dimension that has multiple hierarchies. In the earlier example, you would typically create one time dimension with multiple hierarchies but, if you require a time dimension for a different purpose, such as fiscal time—which often differs from calendar time—you could create it as a separate dimension.

Time Dimension logic in Analysis Services

Analysis Services includes built-in logic for time dimensions that puts the correct days into the correct months and the correct months into the correct quarter. Time dimensions are the only dimension type that can include this type of logic, because time data behaves in a consistent way across all databases. For example, Analysis Services understands how months roll up into years, and can apply this logic in any database. However, Analysis Services has no way of understanding the logic that controls which product should be assigned to each product group for any given database, because this will differ between databases.

- Time dimensions are included in the majority of cubes
- Time dimensions include hierarchies such as year/month/day
- Analysis Services includes built-in logic for time dimensions
- Time dimensions store data in one of two ways:
 - On the server
 - In the data source

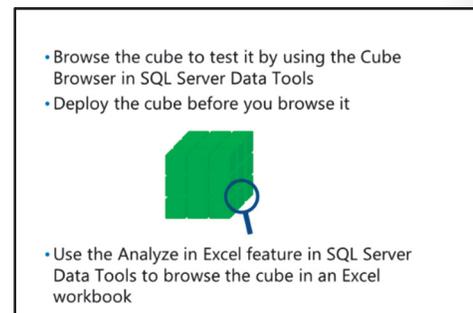
Types of Time Dimensions in Analysis Services

Creating a time dimension involves creating a table that contains the time data. The dimension is then created, based on this table. You can use the Dimension Wizard to create a time dimension—this wizard enables you to create a time dimension in two ways:

- **Server time dimensions.** Server time dimensions are based on tables stored on the Analysis Services server. A server time dimension avoids the need for joins in the data source, and can therefore be faster to process. You can use server time dimensions if security settings prevent you from creating tables in the data source.
- **Time dimension tables.** Time dimensions are based on tables that are created and stored in the data source. Many time dimensions need to include additional data—for example, corporate time information such as quarters, which do not follow calendar quarters or public holidays. A time dimension table gives you greater control so you can, for example, add detail to the time members, which allows you to store additional properties for each date.

Browsing a Cube

After you create a cube, users can access it and analyze the data that it contains by using a tool such as Excel. To ensure that the cube functions as expected before allowing users to access it, you can use the Cube Browser in SQL Server Data Tools. You can use the Cube Browser to browse the cube, and to see the measures that it contains aggregated by its dimensions. Being able to browse the cube from within the development environment is very useful, because it means you can test the cube without switching between a client application and SQL Server Data Tools. By using the Cube Browser, you can change the security context and browse the cube as another user, which helps to ensure that any security features and perspectives that you have configured are functioning correctly.



 **Note:** Before you can browse cube data in the Cube Browser tab, you must deploy the database and cube. You can then click the Browser tab from within the Cube Designer to browse the cube.

Analyze in Excel

You can also use the Analyze in Excel option, in the Cube Browser in SQL Server Data Tools, to open an Excel workbook and use it to browse data in the cube. The Analyze in Excel feature automatically creates the data connection from Excel to the cube, so there is no need to configure it manually.

Demonstration: Creating and Browsing a Cube

In this demonstration, you will see how to:

- Create a cube.
- Deploy and browse a cube.
- Customize cube measures and dimensions.

Demonstration Steps

Create a Cube

1. Ensure that you have completed the previous demonstration in this module.
2. Maximize Visual Studio, which should be open with the **Demo** project loaded.
3. In Solution Explorer, right-click **Cubes**, and click **New Cube**.
4. On the **Welcome to the Cube Wizard** page, click **Next**.
5. On the **Select Creation Method** page, ensure that **Use existing tables** is selected, and click **Next**.
6. On the **Select Measure Group Tables** page, click **Suggest** then note that the wizard identifies **DimReseller** and **Reseller Sales** as potential measure group tables. Clear **DimReseller** so that only **Reseller Sales** is selected, and then click **Next**.
7. On the **Select Measures** page, clear the **Reseller Sales** check box to clear all selections, select the **Total Product Cost** and **Sales Amount** measures, and then click **Next**.
8. On the **Select New Dimensions** page, clear the **Reseller Sales** dimension, and then click **Next**.
9. On the **Completing the Wizard** page, change the cube name to **SalesDemo** and click **Finish**. The cube is created and opened in the Cube Designer.

Deploy and Browse a Cube

1. In Solution Explorer, right-click **Demo** and click **Deploy**. If prompted, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** user.
2. After the deployment completes, in the Cube Designer, click the **Browser** tab.
3. In the Metadata pane, expand **Measures** and expand **Reseller Sales**. Note that the wizard has created the measures you selected.
4. Drag the **Sales Amount** measure to the **Drag levels or measures here to add to the query** area. The total sales amount for all reseller sales is shown.
5. In the Metadata pane, note that the wizard has created dimensions for the **DimReseller** and **DimProduct** tables. It has also determined that there are three relationships defined between the **Reseller Sales** table and the **DimDate** table, and so has a dimension for each related column (**Due Date**, **Order Date**, and **Ship Date**).
6. Expand each dimension and note that the wizard has only created attributes for key columns. You will need to modify the dimensions to add meaningful attributes for analysis.
7. Drag the **Product Key** attribute from the **Dim Product** dimension to the left of the **Sales Amount** value in the grid. The browser now shows the aggregated sales amount totals for each product key.

Customize Measures and Dimensions

1. In the Cube Designer, click the **Cube Structure** tab, and then, in the Measures pane, expand **Reseller Sales**.
2. Right-click the **Total Product Cost** measure and click **Rename**, then change the name of the measure to **Cost**.
3. In the Dimensions pane, right-click **Dim Reseller**, click **Rename** and change the name of the dimension to **Reseller**.
4. Repeat the previous step to rename **Dim Product** to **Product**.
5. Right-click the **Product** dimension and click **Edit Dimension**. This opens the Dimension Designer for the **Dim Product** dimension (the dimension is named **Product** in the **SalesDemo** cube, but still named **Dim Product** in the database).
6. In the Data Source View pane, in the **DimProduct** table, drag the **EnglishProductName** field to the Attributes pane.
7. In the Attributes pane, right-click **English Product Name** and click **Rename**, then rename the attribute to **Product Name**.
8. In Solution Explorer, right-click **Demo** and click **Deploy**. If prompted, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** user.
9. After the deployment completes, click the **SalesDemo.cube [Design]** tab to return to the Cube Designer, and then click the **Browser** tab.
10. On the **Browser** tab, click the **Reconnect** button in the upper left corner. If the grid contains any data from previous queries, right-click in the grid and click **Clear Grid**.
11. In the Metadata pane, expand **Measures** and **Reseller Sales**, and drag **Cost** to the **Drag levels or measures here to add to the query** area.
12. Expand the **Product** dimension, and then drag **Product Name** to the left of the **Cost** value. The total cost associated with sales for each product is shown.
13. On the **File** menu, click **Save All**.
14. Keep Visual Studio open. You will use it again in a later demonstration.

Check Your Knowledge

Question	
You have created a new cube by using SQL Server Data Tools, and you want to test it by browsing the cube by using a PivotTable in Excel. What is the easiest way to do this?	
Select the correct answer.	
<input type="checkbox"/>	Open Excel, connect to the cube, create a PivotTable, and then add measures and attribute values to it.
<input type="checkbox"/>	In the Cube Browser window in SQL Server Data Tools, use the Analyze in Excel options to open an Excel workbook, and then add measures and attribute values to it.

Lesson 4

Overview of Cube Security

After you install an instance of SQL Server Analysis Services, only members of the server role have server-wide permissions to perform any task within that instance. This is because, by default, no other users have access permissions to the objects in the instance. Members of the SSAS server role can grant other users access to server and database objects by using Microsoft SQL Server Management Studio, SQL Server Data Tools, or an XML for Analysis (XMLA) script. This lesson describes the SSAS security model, and explains how to use roles and permissions to implement security.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the SSAS security model.
- Describe the use of roles in SSAS.
- Describe the use of permissions in SSAS.

The Analysis Services Security Model

SSAS enables users to connect by using only Windows® accounts—unlike a SQL Server database engine instance. An Analysis Services instance does not include the ability to create logins. In a domain environment, Active Directory user accounts are authenticated by using the Kerberos protocol. After successful authentication, users can access the resources in Analysis Services, providing that they have been authorized to do so. Administrators can authorize users by assigning permissions that allow them access to specific resources, such as cubes and dimensions. To manage resource access in Analysis Services, you create roles, assign the required permissions to the roles, and then assign user accounts to the appropriate roles. User accounts inherit any permissions assigned to the roles that they belong to.

- Analysis Services takes advantage of Windows authentication to authenticate connections
- Assigns permissions based on role membership
- Permissions can be applied to databases, data sources, dimensions, individual dimension members, and individual cells

- SSAS has a single fixed server role for administrators—this is called the **server administrator role**. Members of this role have full permissions to perform any action in the instance.
- Administrators can create database roles and grant access permissions to these roles to control user access and database administrator rights. Role permissions are database specific.
- You can grant database role permissions for Analysis Services databases, cubes, dimensions, dimension members, individual cells within a cube, mining structures, mining models, data sources, and stored procedures.

Working with Roles

Roles are a key part of Analysis Services security. You can use the built-in **server administrator role** and user-created database roles to assign rights and permissions to users.

The Server Administrator Role

The **server administrator role** is a fixed role that has administrative rights over an entire Analysis Services instance—you cannot change permissions on this role and you cannot create any additional server roles. Members of the **server administrator role** can access all Analysis Services databases and objects on that instance. To give lower-level administrative rights, you should consider adding users to a database role, and granting the relevant permissions to that role. You can use SQL Server Management Studio to add members to the server role on the **Security** page of the **Analysis Server Properties** dialog box.

- Manage access to Analysis Services resources by using roles
 - Server administrator role
 - Used to assign server-wide rights and permissions
 - The server role is the only role at server level
 - You cannot change server role permissions
 - Database roles
 - Used to assign more limited rights and permissions
 - Created by administrators

Database Roles

To assign permissions to users who do not require server-wide administrative rights, you can create database roles. You should first decide what levels of access are required within your organization, including basic users, power users, managers, and department administrators. After you have defined the roles, you can create them, and then add users to the roles. Consider the following points when planning database roles and permissions:

- When assigning permissions, you should always follow the principle of least permission—you should never give a role greater permissions than it requires to enable users to perform their tasks.
- Remove groups with duplicated permissions and, where possible, consolidate groups with similar permissions, providing this does not give any sensitive permission to another group.

You can create database roles by using SQL Server Management Studio or by using SQL Server Data Tools:

- In SQL Server Management Studio, in Object Explorer, expand the database, right-click the **Roles** folder, and then click **New Role**.
- In Visual Studio, in Solution Explorer, right-click the **Roles** folder and then click **New Role**. You can also add a role by clicking the **Project** menu, and then clicking **New Role**.

Assigning Permissions

Administrators assign permissions to roles, and these permissions determine the actions that a user can perform. You can set permissions for Analysis Services objects including databases, data sources, cubes, cells, dimensions, and dimension data.

- Define permissions including:
 - Database permissions
 - Data source permissions
 - Cube permissions
 - Cell permissions
 - Dimension permissions

Database Permissions

Assigning permissions to roles in an Analysis Services database gives users administrative rights in the database—it does not enable them to view or work with the data in the database. To enable users to browse dimensions and measures, you need to assign permissions on cubes and other objects. The permissions you can assign on Analysis Services databases include:

- **Full control (Administrator).** This gives users the ability to perform all administrative tasks, such as processing databases and database objects, and creating and managing roles.
- **Process database.** This enables users to process the database. Processing a database populates it with data from the data source, and updates the database with new data.
- **Read definition.** You can assign this to roles that also have the **Process database** permission to enable users to process a database, by using tools such as SQL Server Data Tools. Without the **Read definition** permission, the **Process database** permission only enables user accounts to process databases by using scripting.

Data Source Permissions

In most cases, you do not need to assign permissions to enable users to use data sources, because the permissions that you assign on the cube and other objects are sufficient. However, some data mining operations, such as performing predictions based on a mining model, do require data source access. In this situation, you can give users the **Read** permission on the relevant data source.

Cube Permissions

To access data within the database, users must either be members of a role with permissions to access the cube, or members of a role with administrative rights. You can assign the following permissions for a cube:

- **Read.** This enables users to read the data in a cube. This permission is usually sufficient for the majority of users, who will only need to analyze data and not update it.
- **Read/Write.** This enables users to read and change the data. Note that updates that users perform are not written back to the source database—instead, the changes are written to a writeback table.

By default, a role with permissions for the cube can access all objects in the cube, including:

- The individual cells within the cube.
- The dimensions and dimension members that the cube contains.



Note: Permissions to access a data source are not required for a user to access cube data.

Cell Permissions

Cell permissions control access to measure data in a cube at the individual cell level. A role that has **Read** or **Read/Write** permissions on a cube can, by default, access all individual cells in the cube, because the permissions are inherited from the cube level. By applying cell permissions, you can override the default permissions to control access at a more granular level. Although the default is that cell permissions apply to all cube cells, if you grant access to one or more cells for a role, that role is denied access to all other cells—until you grant further permissions. You can use an MDX statement to specify the range of cells to which the permissions apply.



Note: When a user attempts to access cells for which they do not have permission, the query returns the value **#N/A** for that cell.

Note that assigning permissions to roles at the cell level cannot raise the permission level that is assigned to the role at the cube level. For example, if a role has **Read** permission on a cube and is assigned **Read/Write** permission on a cell, the effective permission that they have on the cell is only **Read**—the **Write** portion of the permission is not applied.

In addition to the **Read** and **Read/Write** permissions for cells, you can also assign the **Read-Contingent** permission. This permission conditionally enables access to a calculated cell, but users can only read the cell if they also have permissions on all of the cells that the calculation references. For example, imagine a role called **Sales Manager** that has the following access rights:

- **Read** access to the **Sales Value** measure, but no access to the **Product Cost** measure.
- **Read** access to the **Profit** calculated measure, which is defined as **Sales Value** minus **Product Cost**.

When granted **Read** permission on the **Profit** measure, **Sales Manager** role members can see the results of the **Profit** measure. Because they can also view the results of the **Sales Value** measure, they could calculate the value of the **Product Cost** measure, even though they have no direct access to it. Applying **Read-Contingent** access to the **Profit** measure means that users can only see the results of the **Profit** measure if they also have permission to view both the **Sales Value** and the **Product Cost** measures.

Permissions for Dimensions and Dimension Data

Dimension permissions control administrative actions, such as processing the dimension. Dimension permissions do not control access to the data that a dimension contains. However, you can define permissions on dimension attribute members and measures to limit access to dimension data.

Demonstration: Implementing Cube Security

In this demonstration, you will see how to:

- Create a role and assign permissions.
- Test permissions.

Demonstration Steps

Create a Role

1. Ensure you have completed the previous demonstration in this module. Visual Studio should be open with the **Demo** project loaded.

2. In Solution Explorer, right-click **Roles**, and click **New Role**. In Solution Explorer, right-click the **Role.role** icon that has been added, click **Rename**, and change the name to **Restricted User.role**. When prompted to change the object name, click **Yes**.
3. In the role designer, on the **General** tab, note that by default this role has no database permissions.
4. On the **Membership** tab, note that this is where you can add Windows users and groups to the role.
5. On the **Data Sources** tab, note that by default the role has no access to any data sources.
6. On the **Cubes** tab, change the **Access** value for **SalesDemo** to **Read**. This allows the role to access the cube and read its data.
7. On the **Cell Data** tab, note that this is where you can specify MDX statements that define cells in the cube that you want to permit or deny this role to access.
8. On the **Dimensions** tab, note that the role has **Read** access to all dimensions in the database. In the **Select Dimension Set** list at the top of the page, click **SalesDemo cube dimensions** and note that these permissions are inherited by the dimensions in the **Demo Cube** cube.
9. On the **Dimension Data** tab, in the **Dimension** list, under **SalesDemo**, select **Measures Dimension** and click **OK**. On the **Basic** tab, clear the check box for the **Cost** measure.
10. In the **Dimension** list, under **SalesDemo**, select **Product** and click **OK**. On the **Basic** tab, in the **Attribute hierarchy** list, select **Product Name**.
11. Select **Deselect all members** then scroll to the bottom of the list of product names and select all of the products that begin with the word **Touring**.
12. On the **File** menu, click **Save All**.

Test Permissions

1. In Solution Explorer, right-click **Demo** and click **Deploy**. If prompted, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** user.
2. When deployment is complete, in Solution Explorer, double-click **SalesDemo.cube** to return to the Cube Designer.
3. On the **Browser** tab, click the **Reconnect** button at the top left. If the grid contains any data from previous queries, right-click in the grid and click **Clear Grid**.
4. In the Metadata pane, expand **Measures** and **Reseller Sales**, and drag **Sales Amount** and **Cost** to the **Drag levels or measures here to add to the query** area. Expand the **Product** dimension and drag **Product Name** to the left of the measure values. The browser shows the cost and sales amount for all products.
5. Right-click in the grid and click **Clear Grid**.
6. In the top left area of the browser window, click the **Change User** button. In the **Security Context – Demo Cube** dialog box, select **Roles**, and in the drop-down list, select **Restricted User** and click **OK**. Click **OK** again to return to the browser window.
7. Note that the grid has been cleared then, in the Metadata pane, expand **Measures** and **Reseller Sales**, and drag **Sales Amount** to the **Drag levels or measures here to add to the query** area. Note that members of this role cannot access the **Cost** measure.
8. Expand the **Product** dimension and drag **Product Name** to the left of the measure values. The browser only shows the sales amount for the **Touring** products.
9. On the **File** menu, click **Save All**, and then close Visual Studio.

Check Your Knowledge

Question	
<p>You wish to assign permissions in Analysis Services to meet the following criteria: User A needs access to the entire cube to browse and analyze the data. User B needs access to the entire cube to browse and analyze the data, and also needs to perform predictive analysis against a data mining model.</p> <p>How should you configure these users' roles and permissions?</p>	
Select the correct answer.	
	Create a single role and assign permissions that enable access to the cube and the data source. Assign both user accounts to this role.
	Create a single role and assign permissions that enable access to the cube. Assign both user accounts to this role.
	Create a single role and assign permissions that enable access to the data source. Assign both user accounts to this role.
	Create two roles, named Role X and Role Y. Assign permissions that enable access to the cube on Role X and assign permissions that enable access to the cube and the data source on Role Y. Assign User A to Role X and User B to Role Y.
	Create two roles, named Role X and Role Y. Assign permissions that enable access to the cube on Role X and assign permissions that enable access to the cube and the data source on Role Y. Assign User A to Role Y and User B to Role X.

Lesson 5

Configuring SSAS

It is important to understand how SSAS allocates and manages memory to ensure that memory usage and limits are optimized for the server that is running SSAS. Different configurations, such as multi-instance, sharing with a SQL Server relational instance, or single-instance configurations, might require some adjustment of the memory configuration settings. In addition, it is important to understand the optimal storage configuration for Analysis Services, because this has specific characteristics that are different to other SQL Server workloads.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe memory configuration settings.
- Describe NUMA support settings.
- Describe common disk configurations.

Configuring SSAS Memory Limits

SSAS allocates an initial amount of memory (40 to 50 MB) when the service starts. This is the case even if the Analysis Services instance is idle. As query or processing workloads increase, this amount of memory is typically increased, based on certain threshold limits set on the SSAS instance.

- Analysis Services has memory threshold settings
 - LowMemoryLimit
 - VertiPaqMemoryLimit
 - TotalMemoryLimit
 - HardMemoryLimit
- Configure limits based on scenario
 - Single dedicated server
 - Multiple SSAS instances
 - SSAS and SQL DB instances

The threshold limits are outlined in the following table:

Setting	Description
LowMemoryLimit	This limit is used by multidimensional instances, and is the lowest limit at which the SSAS instance will begin to release memory allocated to infrequently used objects.
VertiPaqMemoryLimit	This limit is used by tabular instances, and is the lowest threshold at which the SSAS instance begins releasing memory allocated to infrequently used objects. If memory paging to disk is enabled, this value also defines the level of memory consumption at which memory paging should start. The default value is 60.
TotalMemoryLimit	This is an upper threshold at which the SSAS instance begins releasing memory more aggressively to make room for currently executing requests and new high-priority requests. This limit applies to both multidimensional and tabular instances. The default value is 80 percent of either physical memory or virtual address space, whichever is smaller.

Setting	Description
HardMemoryLimit	This threshold defines a level at which the SSAS instance starts to reject requests completely due to memory usage. The default value for this setting is 0, which automatically places the threshold midway between the TotalMemoryLimit value and either the physical memory or virtual address space, whichever is smaller.

Setting the above properties with a value between 1 and 100 will interpret the values as a percentage of system memory (or virtual address space). Setting the properties with a number larger than 100 will interpret the value as a fixed number of bytes. For example, setting the value of TotalMemoryLimit to 85 will mean that SSAS will release memory more aggressively when 85 percent of memory is in use, but setting the value to 12884901888 would mean that SSAS would start releasing memory more aggressively after 12 GB of memory was in use.

In addition, for tabular instances, you can set the VertiPaqPagingPolicy property to 0 to disable memory paging to disk. This will generate an out-of-memory error if no more memory is available. Setting the value to 1 enables paging to disk.



Note: Several other server properties can affect memory behavior, but you should not alter others than those outlined here without seeking guidance from Microsoft Support.

Typical SSAS Deployments

A few common deployment scenarios are worth considering when deciding whether to adjust the memory limits values for an Analysis Services instance.

Single Server, Single SSAS Instance

With a single server configuration that is dedicated to a single instance of SSAS, the default memory limits configuration does not often require any adjustment until the server has 64 GB RAM or more. Higher memory configurations can leave a large amount of memory reserved for the operating system, which is unnecessary.

Single Server, Multiple SSAS Instances

SSAS attempts to increase the allocated memory for an instance up to the HardMemoryLimit value, but does not release this memory back to the operating system or to other applications on the same server. With multiple SSAS instances sharing the same server, you should configure memory limits carefully so that one SSAS instance does not restrict memory from other instances. This means that you should take the value of physical memory in the server, subtract an amount to reserve for the operating system, and divide the remaining value between all the SSAS instances on the server to set the HardMemoryLimit value for each instance (typically as a static value in bytes rather than a percentage value). You then set the TotalMemoryLimit and LowMemoryLimit values for each instance in line with the HardMemoryLimit for that instance. Note that memory does not need to be divided up evenly between each instance. For example, you might have a production SSAS instance and a development instance on the same server. The development instance might have a small value set for HardMemoryLimit, so that there is plenty of memory available for the production instance which would have a high HardMemoryLimit value.

Single Server, SQL Server DB Instance and SSAS Instance

Although the SQL Server service can grow and shrink memory usage, Analysis Services cannot reduce memory that has been allocated. This means that, to leave memory available for the SQL Server database instance, it is important to adjust the memory limit values for Analysis Services. For example, on a server with 64 GB RAM, you might set the HardMemoryLimit value to 28 GB, to leave the remaining 36 GB for the operating system and for the SQL Server database instance. In addition, for a configuration such as this, you might also consider configuring the minimum server memory property on the SQL DB instance, to ensure that the SQL Server database instance has sufficient memory for processing.

For more information, see *Memory Properties* in the SQL Server 2016 Technical Documentation:

 **Memory Properties**

<https://aka.ms/wc4yue>

NUMA Configurations

Non-Uniform Memory Access (NUMA) is a mechanism designed to improve performance on multiprocessor systems. NUMA architecture allocates specific areas of memory to individual processors in a multiprocessor system, so that processors experience less contention over access to a large shared memory.

Analysis Services multidimensional instances are NUMA node aware. You can use the PerNumaNode setting to specify NUMA node behavior for multidimensional instances in the following ways:

- NUMA improves performance on multiprocessor systems
- SSAS multidimensional instances NUMA aware
 - Change NUMA settings with PerNumaNode property
- SSAS tabular instances NUMA aware from SP1
 - NUMA behavior automatic with four-plus NUMA nodes
- GroupAffinity property provides advanced performance tuning on systems with 64-plus logical processors

PerNumaNode value	Behavior
-1	This is the default setting and enables SSAS to automatically adjust NUMA node behavior. On systems with less than four NUMA nodes, the SSAS instance will behave in the same way as setting PerNumaNode=0. On systems with four or more NUMA nodes, the instance will behave in the same way as setting PerNumaNode=1.
0	The SSAS instance will ignore NUMA nodes and create one IOProcess thread pool to be used by all logical processors.
1	The SSAS instance creates one IOProcess thread pool for each NUMA node. This improves coordinated access to memory.
2	The SSAS instance creates one IOProcess thread pool for each logical processor. It is intended for highly scaled hardware, with intensive Analysis Services workloads.

As of SQL Server 2016 SP1 and later, Analysis Services tabular instances are NUMA node aware. NUMA node awareness and changes to behavior are automatically activated on systems with four or more NUMA nodes, where a separate job queue is maintained for each NUMA node. On systems with only two NUMA nodes, the marginal benefit usually doesn't overcome the overhead of managing NUMA.

MCT USE ONLY. STUDENT USE PROHIBITED

For more information, see *Improving Analysis Services Performance and Scalability with SQL Server 2016 Service Pack 1* on the Analysis Services Team Blog:



Improving Analysis Services Performance and Scalability with SQL Server 2016 Service Pack 1

<https://aka.ms/k6wf45>

Processor Affinity

You can also use the GroupAffinity property to control which Analysis Services thread pools can use which logical processors in the system. This means a more granular control of workloads for advanced performance tuning on large, multicore systems. Customizing GroupAffinity on systems that have less than 64 logical processors might not provide tangible benefits and may even be detrimental to Analysis Services performance.

For more information, see *Thread Pool Properties* in the SQL Server 2016 Technical Documentation:



Thread Pool Properties

<https://aka.ms/n3y2ug>

Storage Configuration

Based on a typical Analysis Services workload, the I/O generated is largely random reads during operational periods. Of course, this depends on how frequently the cube is being updated, but cube updates are periodical and often occur outside of operational hours. This means that disk configurations that favor read operations are typically desirable.

In principle, most variants of RAID storage are suitable from a performance perspective. However, production environments should also consider requirements for resiliency. It is recommended that RAID 1, RAID 5, or another RAID variant that offers some form of redundancy, is used to protect the cube data from disk error or loss.

In choosing the type of disk, the decision rests on the balance between cost and speed: SATA disks offer the lowest performance, but are the cheapest; SAS disks offer good speed at a reasonable cost; and SSDs offer the fastest performance, but at significant cost.

In addition, there are a few general recommendations for Analysis Services storage configuration:

- **Configure a single volume.** It is not normally considered a good practice to separate any of the data files for the cube onto a separate volume. Configuration of a single, large volume for your cubes will typically ease management overhead and allow the largest I/O bandwidth to the storage hardware.
- **Exclude Analysis Services folders from virus scanners.** Ensure that the Data, TempDir, and backup folders are not being scanned by antivirus software. There are no executable files in these folders.
- **Defragment the data folder.** Over time, the files in cube data folders can become very fragmented. Periodically, you should defragment the cube volume during a maintenance period to keep performance at expected levels.

- Mostly random read operations
- RAID configuration typically:
 - RAID 1
 - RAID 5
- Balance cost and speed:
 - SATA
 - SAS
 - SSD
- General recommendations:
 - Single disk volume
 - Exclude data folders from antivirus
 - Defrag the data folder

For more information, see the *Microsoft SQL Server Analysis Services Multidimensional Performance and Operations Guide* on the Microsoft Download Center.



Microsoft SQL Server Analysis Services Multidimensional Performance and Operations Guide

<https://aka.ms/q0gr28>

Check Your Knowledge

Question	
You are configuring an Analysis Services instance to coexist on the same server as a SQL Server relational database instance. You want to ensure that Analysis Services leaves enough memory for the relational instance to be always running. What setting should you configure?	
Select the correct answer.	
<input type="checkbox"/>	Set the HardMemoryLimit value to 0.
<input type="checkbox"/>	Set the TotalMemoryLimit value to 0.
<input type="checkbox"/>	Set the HardMemoryLimit value to 95.
<input type="checkbox"/>	Set the HardMemoryLimit value to 45.
<input type="checkbox"/>	Set the LowMemoryLimit value to 45.

Lesson 6

Monitoring SSAS

Monitoring an Analysis Services multidimensional instance for performance will give you important information regarding instance and cube performance. Of course, instance performance will be affected by the availability of hardware resources, such as processor or memory. However, cube performance can also be greatly affected by the way queries are written and processed.

Lesson Objectives

After completing this lesson, you will be able to:

- Select Analysis Services instance counters to monitor.
- Use tools to monitor Analysis Services.
- Select counters to monitor cube and query performance.

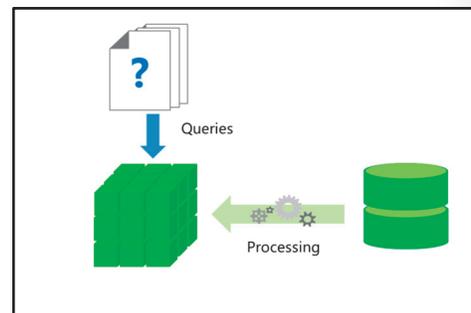
Considerations for Analysis Services Performance

Before attempting to identify and analyze the performance of an Analysis Services instance, it is important to understand the workloads and the available tools.

Analysis Services Workloads

Analysis Services databases do not have the same workload characteristics as relational database instances, and must typically support the following two key workloads:

- **Queries.** Users submit queries to retrieve data from Analysis Services (typically through various applications). Both multidimensional and tabular instances must perform two tasks:
 - **Data retrieval.** The storage engine extracts the required data from the data model.
 - **Calculations.** The formula engine performs the required calculations to output the data.
- **Cube Processing.** The data held in the cube is typically updated at a predetermined interval, to allow queries to represent more up-to-date information or changes to the data model. Cube processing can be a resource intensive task, so it should only be performed as frequently as the data needs updating for fresh reports or analysis output.



Monitoring Analysis Services

Performance monitor has a large number of counters available for monitoring Analysis Services that can be used for real-time monitoring and troubleshooting, or for creating performance baselines for future comparison.

Analysis Services Performance Counters

Each Analysis Services instance on the server will have a dedicated set of performance counters. A default instance will use the **MSAS13:Memory** object for the memory counters and the **MSOLAP\$SSAS2:Memory** object will have the memory counters for the instance named **SSAS2**. The following counters can help to provide an overview of Analysis Services resource utilization:

Object	Counter(s)	What the counter shows
MSAS13:Memory	Memory Usage KB	The quantity of memory used by Analysis Services.
MSAS13:Connection	Current Connections	The number of open connections to Analysis Services.
MSAS13:Proc Aggregations	Temp file bytes written	The amount of data written to the temp file during processing operations. If the cube and hardware are well balanced, this will be near 0.
MSAS13:Proc Indexes	Current Partitions Rows/sec	Measures the speed and concurrency of index processing.
MSAS13:Processing	Rows read/sec Rows written/sec	The speed of processing data.
Memory	Available Kbytes Page Faults/sec	The amount of physical memory available to the server and the frequency of memory paging to disk.
System	File Read bytes/sec File Read Operations/sec	File system read activity, including reads from the file system cache.
Network Interface	Bytes Received/sec Bytes Sent/sec	Quantity of network traffic.
TCPv4 and TCPv6	Segments/sec Segments Retransmitted/sec	Stability of network connections.

For performance troubleshooting, you might also include counters from the MSAS13:Threads, MSAS13:MDX and MSAS13:Storage Engine Query objects. For I/O troubleshooting, the Memory and Physical Disk objects contain some useful extra counters.

Analysis Services Monitoring Tools

SQL Server Profiler enables you to create, capture and replay event data (called a trace) to examine behavioral and performance characteristics of Analysis Services. For example, you might perform a trace to determine how long certain actions performed by analysis services last, when a specific query is made.

SQL Server Profiler works by capturing event information that occurs while Analysis Services is running. Events are certain types of action, such as:

- Login connections or disconnections.
- Transact-SQL statements.
- The start or end of a stored procedure.
- The start or end of a SQL batch.
- An error written to the log.

These events are grouped into classes. To capture only the information that you need, you can select which classes of event to capture in the trace. Examples of event classes for Analysis Services include:

- Audit Login
- Audit Logout
- Query Begin
- Query End
- Query Subcube

SQL Server Profiler also gives you the ability to select how much information about each kind of event you capture. Examples of these DataColumns include:

- ApplicationName
- StartTime
- Duration
- DatabaseName
- NYUserName

When you create a trace for SQL Server Profiler, and select the event classes and DataColumns for the trace, you can also save the trace as a template. The template is saved as a file with a .tdf extension. This enables you to rerun the trace at a later time to capture the same kind of information, or transfer the file to a different server to run the same trace there—such as from a development server to a production server.

For more information, see *SQL Server Profiler* in the SQL Server 2016 Technical Documentation:



SQL Server Profiler

<https://aka.ms/mab11d>

- SQL Server Profiler
 - NOT deprecated for Analysis Services
 - Create from scratch or use template
- Extended Events (XEEvents)
 - Graphical interface in SSMS 2016
 - Create from scratch or use template
 - Cover more classes of event than SQL Server Profiler
 - Lower overhead than SQL Server Profiler
- Dynamic Management Views (DMVs)
 - Executed by SELECT statement
 - Displays current system operations and server health

 **Note:** Note that SQL Server Profiler has been deprecated for use with the SQL Server database engine, and will be removed from a future version of the product. However, SQL Server Profiler is *not* deprecated for use with Analysis Services in SQL Server 2016.

Extended Events

Analysis Services 2016 also includes a graphical user interface to create and manage Extended Events (sometimes called XEvents) within SQL Server for capturing detailed event data. Extended Events can capture information similar to SQL Server Profiler; however, Extended Events are considered as superior because they can capture more information and perform additional tasks—and they have less overhead on the system. For example, it is reported that 20,000 events/sec on a 2ghz CPU with 1 GB of RAM takes less than 2 percent of the CPU. This means that, on a modern server, there should be little impact on performance.

Extended Events are constructed in SQL Server Management Studio (SSMS), in a similar way to SQL Server Profiler traces, starting by selecting the types of event to capture (these include equivalent SQL Server Profiler events), and the associated information about the event, called event fields. Extended Event data can be saved to a file for later analysis, or sent to a memory buffer for real-time monitoring. You can also use SSMS to output your Extended Event configuration as a script, which can be useful to reproduce the Extended Event easily on other servers, or to reproduce the capture data after you delete the Extended Event. Deleting the session object is currently the only way to stop the Extended Event from capturing data.

For more information, see *Using Extended Events with SQL Server Analysis Services 2016 CTP 2.3* on the Analysis Services Blog:

 **Using Extended Events with SQL Server Analysis Services 2016 CTP 2.3**

<https://aka.ms/vbnih8>

Dynamic Management Views

Analysis Services also provides Dynamic Management Views (DMVs) to review current server state information. DMVs operate as a set of queries designed to return metadata and monitoring information about the Analysis Services instance.

Most DMV queries are executed by using a **SELECT** statement on the **\$System** schema with an XML/A schema rowset. These include:

- **\$system.discover_commands** that lists all currently running commands on the server.
- **\$system.discover_connections** that lists current open connections.
- **\$system.discover_sessions** that lists all sessions on the server. You can use it to determine the commands that were executed and in which connections.
- **\$system.discover_memoryusage** that lists all memory allocations in the server.
- **\$system.discover_locks** that lists currently held and requested locks.

For more information, see *Use Dynamic Management Views (DMVs) to Monitor Analysis Services* in the SQL Server 2016 Technical Documentation:

 **Use Dynamic Management Views (DMVs) to Monitor Analysis Services**

<https://aka.ms/vx9kt2>

Monitoring Query Performance

In addition to examining server resource issues, it is also important to recognize that query construction can affect query performance. Identifying query performance issues can be challenging, but monitoring the following counters may help identify elements that can be improved:

- Query performance counters:
 - MDX: Total cells calculated
 - MDX: Number of calculation covers
 - MDX: Total Sonar subcubes
 - MDX: Total recomputes
 - MDX: Total NON EMPTY unoptimized
 - MDX: Total NON EMPTY for calculated members

Counter	What to look for
MDX: Total cells calculated	If this number is high, queries might be performing cell-by-cell calculations instead of block-oriented calculations.
MDX: Number of calculation covers	The total number of active and cached evaluation nodes built by MDX execution plans. If this number is high, queries might be performing cell-by-cell calculations instead of block-oriented calculations.
MDX: Total Sonar subcubes	The total number of subcubes that query optimizer generated. If this number is high, queries might be performing cell-by-cell calculations instead of block-oriented calculations.
MDX: Total recomputes	The number of cells recomputed due to error. Non-zero values indicate recalculations to fix errors, again possibly on a cell-by-cell basis, which are using additional processor time.
MDX: Total NON EMPTY unoptimized	The number of times an unoptimized NON EMPTY algorithm is used.
MDX: Total NON EMPTY for calculated members	The total number of times a NON EMPTY algorithm was looping over calculated members.

For more information, see the *Analysis Services MOLAP Performance Guide for SQL Server 2012 and 2014* in the SQL Server 2014 Technical Documentation.



Analysis Services MOLAP Performance Guide for SQL Server 2012 and 2014

<https://aka.ms/fkz91s>

Check Your Knowledge

Question	
Which performance counter can indicate that the server is struggling to perform well during cube processing?	
Select the correct answer.	
<input type="checkbox"/>	MSAS13: Memory
<input type="checkbox"/>	MSAS13: Proc Aggregations
<input type="checkbox"/>	MDX: Total recomputes
<input type="checkbox"/>	MSAS13: Processing
<input type="checkbox"/>	MSAS13: Connection

Lab: Creating a Multidimensional Database

Scenario

Business analysts at Adventure Works Cycles have requested an analytical solution that enables them to “slice and dice” business data to analyze Internet sales. To accomplish this, you will create and test a multidimensional database that includes an OLAP cube.

Objectives

After completing this lab, you will have:

- Created a data source.
- Created and modified a data source view.
- Created and modified a cube.
- Added a dimension to a cube.

Estimated Time: 45 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Creating a Data Source

Scenario

Information workers want to analyze Internet sales data in Excel. To support this, you plan to implement a multidimensional database. First, you need to create a connection and data source to the data warehouse.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create an Analysis Services Project
3. Create a Data Source

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run Setup.cmd in the D:\Labfiles\Lab02\Starter folder as Administrator. Note that the script may take up to several minutes to complete.

► Task 2: Create an Analysis Services Project

1. Use SQL Server Data Tools to create a new Analysis Services Multidimensional and Data Mining project.
2. Name the project **Adventure Works OLAP**, and then save it in the D:\Labfiles\Lab02\Starter folder.

► Task 3: Create a Data Source

1. Use the **Data Source Wizard** in Visual Studio to create a data source for the **AdventureWorksDW** database on the local server.
2. Use Windows authentication to connect to the data source.

3. Configure the data source to use the **ADVENTUREWORKS\ServiceAcct** account with the password **Pa\$\$w0rd**.
4. Name the data source **Adventure Works Data Warehouse**.

Results: After completing this exercise, you will have:

Created an Analysis Services project by using SQL Server Data Tools.

Created a data source.

Exercise 2: Creating and Configuring a Data Source View

Scenario

The **AdventureWorksDW** database includes a large amount of data not required by your users, and the tables use naming conventions that are confusing to business analysts. You need to define which tables you require, and then change their names to be more user-friendly. Users have also requested that they can retrieve full names of customers, in addition to their first, middle, and last names.

The main tasks for this exercise are as follows:

1. Create a Data Source View
2. Configure a Data Source View

► Task 1: Create a Data Source View

1. Use the Data Source View Wizard in Visual Studio to create a data source view that references the **Adventure Works Data Warehouse** data source that you created in the previous task.
2. Include the following tables:
 - **DimCustomer**
 - **DimDate**
 - **DimGeography**
 - **DimProduct**
 - **DimProductCategory**
 - **DimProductSubCategory**
 - **FactInternetSales**
3. Name the data source view **Adventure Works DSV**.

► Task 2: Configure a Data Source View

1. Change the **FriendlyName** property of the tables in the Adventure Works DSV data source view to remove the “Dim” or “Fact” prefix and add spaces between words where appropriate.
 - For example, the **FactInternetSales** table should have the friendly name **Internet Sales**, and the **DimProductCategory** table should have the friendly name **Product Category**.

2. Add a calculation named **Full Name** to the **Customer** table. Use the following MDX expression:

```
CASE
WHEN MiddleName IS NULL THEN
  FirstName + ' ' + LastName
ELSE
  FirstName + ' ' + MiddleName + ' ' + LastName
END
```

Results: After completing this exercise, you will have:

Created a data source view.

Configured a data source view.

Exercise 3: Creating and Configuring a Cube

Scenario

Now you have created a data source view, you are ready to create a cube that enables business users to perform multidimensional analysis on the data. You will then modify the cube to give the measures user-friendly names to make analysis simpler. You will also add extra attributes to dimensions, to provide users with multiple ways to aggregate the measures.

You need to modify dimension attributes used for dates in your cube to ensure uniqueness across temporal periods. For example, the month of January is not unique because it occurs in every year. You must modify the month attribute so that its key is based on both month and year, and then specify which of these key columns should be used when displaying the attribute name.

The main tasks for this exercise are as follows:

1. Create a Cube
2. Configure Measures
3. Configure Dimensions
4. Browse the Cube

► Task 1: Create a Cube

1. Use the Cube Wizard in SQL Server Data Tools to create a new cube from existing tables.
2. Use the **Internet Sales** table as the measure group, and select the following measures:
 - **Order Quantity**
 - **Total Product Cost**
 - **Sales Amount**
 - **Internet Sales Count**
3. Create dimensions for all tables other than **Internet Sales**.
4. Name the cube **Sales**.

► Task 2: Configure Measures

1. Change the name of the **Order Quantity** measure to **Internet Order Quantity**.
2. Change the name of the **Total Product Cost** measure to **Internet Cost**.
3. Change the name of the **Sales Amount** measure to **Internet Revenue**.

4. Save the cube.

► Task 3: Configure Dimensions

1. Modify the **Customer** dimension by using Dimension Designer.
 - Add the **City**, **StateProvinceName**, **EnglishCountryRegionName**, and **PostalCode** attributes from the **Geography** table to the **Geography** dimension.
 - Add the **CustomerAlternateKey**, **Title**, **FirstName**, **MiddleName**, **LastName**, and **Full Name** attributes from the **Customer** table to the **Customer** dimension.
2. Modify the **Product** dimension by using Dimension Designer.
 - Add the **ProductAlternateKey**, **EnglishProductName**, and **ListPrice** attributes from the **Product** table to the dimension.
 - Add the **EnglishProductSubcategoryName** attribute from the **ProductSubcategory** table to the dimension.
 - Add the **EnglishProductCategoryName** attribute from the **ProductCategory** table to the dimension.
3. Modify the **Date** dimension by using Dimension Designer.
 - Add the **EnglishMonthName**, **MonthNumberOfYear**, **CalendarQuarter**, **CalendarYear**, and **CalendarSemester** attributes from the **Date** table to the dimension.
 - Rename the **English Month Name** attribute to **Month**.

► Task 4: Browse the Cube

1. Deploy the **Adventure Works OLAP** project, entering the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd** if prompted.
2. Use the Cube Browser in Visual Studio to view the **Sales** cube, and view the **Internet Revenue** and **Internet Sales Count** measures to the data area by **Order Date.Calendar Year**.

Results: After completing this exercise, you will have:

Created a cube.

Edited measures in the cube.

Edited dimensions in the cube.

Browsed the cube.

Exercise 4: Adding a Dimension to the Cube

Scenario

After you have created the cube with the wizard, a business analyst has requested the ability to view Internet sales by sales territory. To accommodate this, you must create a dimension in the database, and add it to the cube.

The main tasks for this exercise are as follows:

1. Edit the Data Source View
2. Create a Dimension
3. Add a Dimension to a Cube
4. Analyze a Cube by Using Excel

► Task 1: Edit the Data Source View

1. Edit the Adventure Works DSV data source view and add the **DimSalesTerritory** table.
2. Change the friendly name of the **DimSalesTerritory** table to **Sales Territory**.
3. Save the data source view.

► Task 2: Create a Dimension

1. Add a dimension to the database based on the **Sales Territory** table you added in the previous task.
2. Use **SalesTerritoryKey** as the key column and the name column.
3. Add the **SalesTerritoryRegion**, **SalesTerritoryCountry**, and **SalesTerritoryGroup** attributes and enable browsing for each of them.

► Task 3: Add a Dimension to a Cube

1. Add the **Sales Territory** dimension to the **Sales** cube.
2. Verify that the **Sales Territory** dimension is used by the **Internet Sales** measure group through a relationship based on the **Sales Territory Key** attribute.
3. Save the cube.

► Task 4: Analyze a Cube by Using Excel

1. Deploy the **Adventure Works OLAP** project, entering the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd** if prompted.
2. Analyze the cube in Excel, and view the **Internet Revenue** measure by the **Sales Territory Group**, **Sales Territory Country**, and **Sales Territory Region** attributes of the **Sales Territory** dimension.
3. Close Excel without saving the workbook.
4. Save the project and close SQL Server Data Tools.

Results: After completing this exercise, you will have:

Edited the data source view.

Created a dimension.

Added a dimension to the cube.

Browsed the cube by using Excel.

Question: In the final exercise, in which you added a dimension to the cube, you edited the data source view to include the table that the dimension was based on. Why was it not necessary to also edit the data source object?

Module Review and Takeaways

In this module, you learned about:

- The core components of a multidimensional database solution.
- Data sources and data source views.
- Creating and browsing multidimensional cubes.
- How to implement security in a multidimensional database.

Review Question(s)

Question: What should you consider when assigning friendly names for objects in a multidimensional database?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 3

Working with Cubes and Dimensions

Contents:

Module Overview	3-1
Lesson 1: Configuring Dimensions	3-2
Lesson 2: Defining Attribute Hierarchies	3-6
Lesson 3: Implementing Sorting and Grouping for Attributes	3-12
Lesson 4: Slowly Changing Dimensions	3-15
Lab: Working with Cubes and Dimensions	3-18
Module Review and Takeaways	3-23

Module Overview

Dimensions are a fundamental component of cubes. Facts, or measures, have the values that interest you, but dimensions provide the business context by which you aggregate these measures. Dimensions organize your data by category and show results grouped by these categories, such as product, customer, or month.

Microsoft® SQL Server® Analysis Services dimensions contain attributes that correspond to columns in dimension tables. These attributes appear as attribute hierarchies and can be organized into user-defined hierarchies, or defined as parent-child hierarchies, based on columns in the underlying dimension table. You use hierarchies to organize measures contained in a cube.

Objectives

After completing this module, you will be able to:

- Configure dimensions.
- Define attribute hierarchies.
- Sort and group attributes.
- Configure Slowly Changing Dimensions (SCDs).

Lesson 1

Configuring Dimensions

All Analysis Services dimensions are groups of attributes based on columns from tables or views in a data source view. Dimensions can exist independently of a cube—you can use dimensions in multiple cubes, use them multiple times in a single cube, and link them between Analysis Services instances. A database dimension exists independently of a cube; an instance of a database dimension within a cube is called a cube dimension.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the concept of a dimension.
- Edit dimensions with Dimension Designer.
- Describe the dimension storage options.
- Describe the use of dimension attributes.
- Define the columns used to connect dimensions to the fact table, to display to users, and to use in Multidimensional Expressions (MDX) calculations.

Dimension Concepts

Dimensions form the contexts for facts, and define the aspects of the business by which you aggregate facts. For example, **Product** and **Time** dimensions might provide context to the **Sales Revenue** measure.

Dimensions are collections of attributes from tables or views. You can use dimensions in multiple cubes and link to them from remote instances.

Attributes add meaning to dimensions. Each column in the dimension table can provide an attribute that has a piece of information about the dimension member. For example, a **Retail Store** dimension might have **Manager** and **Postal Code** attributes.

The **Key** attribute is typically the primary key of the dimension table. The **Key** attribute is the column in the dimension table that links to the fact table. The **Name** attribute provides a friendly name for the dimension, enabling you to display an alternative, more relevant, name than the source table name-to-end users.

Typically, you arrange attributes into hierarchies that define the drill-down paths through aggregations. Each layer of the hierarchy is called a level. For example, a **Time** dimension might have a hierarchy with **Day**, **Month**, and **Year** levels, but you could define other hierarchies that have attributes such as Week, Quarter, and Century.

By default, every attribute defined within a dimension is part of a two-level hierarchy known as an attribute hierarchy. The top level is named **All**; the second is the **Leaf** level that contains values for each individual member of the attribute. While they can help in browsing, the data within an attribute

Dimensions:

- Provide context for facts
- Are collections of attributes from a table
- Are typically arranged as a hierarchy

hierarchy can typically be too large and difficult to navigate. Attribute hierarchies are often replaced with user defined hierarchies where you can define more levels and make data more manageable. For example, a **Date** attribute hierarchy might have an **All** level and a second level containing all the dates within the dimension. To make this more manageable, you could replace the **Date** attribute hierarchy with a user-defined hierarchy named **Calendar Year** that contains **Year**, **Quarter** and **Month** levels, making the data easier to navigate.

Analysis Services can add dimension intelligence to some dimensions. For example, time dimensions have a fixed structure that can be used by Analysis Services. It knows which months are in each quarter, which days are in each month, and so on.

The Dimension Designer

Use the Dimension Designer to edit the attributes, levels, hierarchies, and translations of a dimension, and to browse the dimension. The Dimension Designer includes four tabs:

- Use the **Dimension Structure** tab to view and edit the attributes, levels, and hierarchies of the dimension.
- Use the **Attribute Relationships** tab to create, modify, or delete the attribute relationships of the selected dimension.
- Use the **Translations** tab to view and edit the multilanguage translations for the attributes, levels, and hierarchies.
- Use the **Browser** tab to browse members of each hierarchy in the dimension. You can only browse members after you have deployed the solution.

- Use the **Dimension Structure** tab to view and edit the attributes, levels, and hierarchies of the dimension
- Use the **Attribute Relationships** tab to create, modify or delete the attribute relationships of the selected dimension
- Use the **Translations** tab to view and edit the multilanguage translations for the attributes, levels, and hierarchies
- Use the **Browser** tab to browse members of each hierarchy in the dimension; you can only browse members after you have deployed the solution

Configuring Dimension Storage

The two dimension storage modes available in Analysis Services are Multidimensional Online Analytical Processing (MOLAP) and Relational Online Analytical Processing (ROLAP). These storage modes define where, and in what structure type, the dimension data is stored:

- Data for a dimension that uses MOLAP is stored in a multidimensional structure in the instance of Analysis Services. This multidimensional structure is created and populated when the dimension is processed. MOLAP dimensions provide better query performance than ROLAP dimensions.
- Data for a dimension that uses ROLAP is stored in tables used to define the dimension. The ROLAP storage mode can be used to support large dimensions without duplicating large amounts of data, but at the expense of query performance. Because the dimension relies directly on the tables in the

- MOLAP
 - Data is stored in a multidimensional structure that is created and populated when the dimension is processed
 - Optimized for fastest query performance
- ROLAP
 - Data is stored in a relational format and aggregated when queried
 - Optimized for minimal processing overhead and real-time querying of the data in the relational source

data source view used to define the dimension, the ROLAP storage mode also supports real-time ROLAP.

Configuring Dimension Attributes

The Cube Wizard and Dimension Wizard create attributes for a dimension. Other wizards and elements of the Analysis Services user interface may further modify these attributes. These default settings are sufficient in most situations, but you can use Dimension Designer to edit the attributes.

You can remove attributes from the dimension by right-clicking the attribute and clicking **Delete**. Removing attributes from a dimension does not affect the data source view, which can be used by multiple dimensions without forcing each dimension to use all the same attributes.

You can rename an attribute to provide a more meaningful or user-friendly name than the dimension table. Do this by right-clicking the attribute in the **Dimension Structure** tab of the Dimension Designer. You can also set the **Name** property of the attribute in the Properties window or edit it directly if the Attributes pane is in the grid view.

Having many attributes for a dimension can confuse users. You can organize attributes into display folders to simplify browsing. Using folders only affects the way client applications display the dimension and has no other effects on hierarchies or attributes. After you have created display folders, deploy the solution and reconnect to see the results on the **Browser** tab.

Attributes can be used to create hierarchies. For example, a **Product** dimension might have **Product Category**, **Product Sub-Category**, and **Product Name** attributes. Hierarchies are covered in more detail in the next lesson.

Attributes are used to add extra detail to dimension members and do not need to be the basis for a hierarchy method. Therefore, in the previous example, you might have **Product Description** and **Product Image** attributes. These attributes are not used for a hierarchy, nor to sort or group members, but are often needed to provide additional detail to dimension members.

Attribute Column Bindings

To control output from attributes, you can define the column that uniquely identifies attribute values, the column that users see, and an optional value column you can use for MDX calculations.

- The **KeyColumn** is the column, or columns, that uniquely identify members for each attribute value in the fact table. This is often the primary key of the dimension table, and is used when you choose to order the hierarchy by key. In some cases, a member must be identified by a combination of column values.

- Dimension attributes can be set with a wizard or the Dimension Designer
- Attributes can be renamed and organized into folders
- Attributes can be used to provide detail or form hierarchies

- Key Column
 - Uniquely identifies members
- Name Column
 - The value displayed to the user
- Value Column
 - The value returned by an MDX function

For example, in a geography dimension, the **City** value "Paris" might not be unique, as there might be a row for Paris, France and another row for Paris, Texas, in the United States. You can either use the primary key column of the dimension table to identify the correct "Paris", or a multicolumn key that includes the city, state, and country or region to identify members uniquely in your dimension. This is particularly important if you plan to include an attribute in a hierarchy.

- The **NameColumn** provides the value that a user will see and can give more useful information than the key column. You can define a name column as a calculated column you have created in a data source view. For example, an **Employee** dimension might have an **EmployeeID** key column, but you might want the name of the employee to be displayed. To do this, you can create a calculated column based on first and last name in the data source view, and then use this as the name column.
- The **ValueColumn** is returned by the MDX **MemberValue** function. This means you can create calculations based on a value other than the name or key. For example, a time dimension is stored in a dimension table with one row for each day. The key column is a SMALLINT and the name column is a CHAR(11) that displays the day, month, and year. The value column contains the date as a SMALLDATETIME. This allows calculations to use the true date value rather than converting it from string values, but also enables the key column to hold a smaller value. A key column value occurs in every row of the fact table, so reducing its size makes the fact table smaller and makes joins between the fact table and the time dimension faster.

Check Your Knowledge

Question	
MOLAP dimensions provide better performance than ROLAP but which of the following is a reason for choosing ROLAP?	
Select the correct answer.	
<input type="checkbox"/>	ROLAP is easier to implement than MOLAP.
<input type="checkbox"/>	When real-time OLAP is required.
<input type="checkbox"/>	Querying ROLAP generally involves simpler queries.
<input type="checkbox"/>	ROLAP is more efficient with small dimensions.

Lesson 2

Defining Attribute Hierarchies

Hierarchies define the multilevel structure of a dimension, and the relationships between the levels can affect query and processing performance.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe hierarchies.
- Distinguish parent-child from balanced hierarchies and describe when each type should be used.
- Describe ragged hierarchies and the options you have when configuring them.
- Describe considerations for using hierarchies.
- Create attribute relationships.

What Are Hierarchies?

Attributes are exposed to users through attribute hierarchies. An attribute hierarchy in a dimension includes an optional **All** level and distinct members of the attribute. For example, a **Customer** dimension might include a **Name** attribute hierarchy with two levels: the **All** level and a second level with a member for each name. Attribute hierarchies define the space of a cube, which you can think of as the multidimensional space created by the product of its attribute hierarchies. In addition to being containers for attribute hierarchies, dimensions can also include user hierarchies as a navigational convenience, but these do not affect cube space.

Defining relationships between hierarchy levels enables Analysis Services to establish more useful aggregations that, in turn, increase query performance. This can save memory during processing and increase performance—which is often important with large or complex cubes.

When arranging attributes into user-defined hierarchies, you define relationships between hierarchy levels. The levels are connected in a many-to-one or one-to-one relationship, referred to as a natural relationship. For example, in a **Calendar Time** hierarchy, a **Day** level is related to the **Month** level, the **Month** level to the **Quarter** level, and so on.

A natural hierarchy is composed of attributes where each is a member property of the attribute below. For example, a **Customer Geography** hierarchy with the levels **Country-Region**, **State-Province**, **City**, **ZIP Code** and **Customer**, is a natural hierarchy because of the relationships between the attributes. By contrast, a **Marital Status-Gender** hierarchy containing the levels **Marital Status** and **Gender** is non-natural, because marital status and gender do not have a hierarchical relationship to each other.

For performance reasons, natural hierarchies are preferred and the Dimension Designer will warn you if you create a hierarchy that is non-natural. You can ignore this warning when it makes sense to use a non-natural hierarchy. For example, it is plausible that you would want to drill down from **marital status** to **gender**.

- Dimensions are typically arranged into hierarchies
- Attributes in natural hierarchies have a relationship (for example Year, Quarter, Month)
- Attributes in non-natural hierarchies do not have a relationship (for example Marital Status-Gender)

Parent-Child Hierarchies

A parent-child hierarchy exists when you have a self-referencing dimension table. For example, an Employee dimension might have a Manager attribute. With the exception of the most senior employee, all other employees have a position in the hierarchy defined by the identities of their managers, and those they manage.

A parent-child hierarchy is an unbalanced hierarchy. This imbalance occurs when the number of levels from leaf to root is different for individual leaf members. For example, some managers might have three levels of employee working for them, while others have just two.

You define parent-child hierarchies, not by the order in which attributes are added, but by the relationships in the dimension table.

Parent-child dimensions are “changing dimensions”, which means that rows can be modified without the need for processing. They are also the only dimension type you can write-enable.

Parent-child dimensions can be read-only and are classified as changing dimensions because the hierarchy structure is calculated at run time. This can be beneficial if the dimension records change frequently, but there are performance considerations. Because the parent-child hierarchy is calculated at run time, no aggregates are calculated when the cube is processed. For example, the total sales for each sales manager, and every employee who reports to them, are only calculated when the query is run.

This trade-off between a versatile changing dimension and query performance must be considered and, ideally, the effects tested. If you do not want to use a parent-child dimension, you can create traditional dimensions. For example, an employee could list their level managers up through the hierarchy. This would allow the dimension aggregates to be calculated during processing, but with a loss of flexibility.

You should also consider what to do with non-leaf data. In the example used here, the sales manager can also make sales. If this is the case, and you have data in the fact table associated to non-leaf members, you must change the dimension's Members With Data property to non-leaf data visible or non-leaf data hidden. Otherwise, processing the cube fails. If you use non-leaf data visible, the parent—in this case the sales manager—will also appear in the level below and seem to report to himself or herself. If you use non-leaf data hidden, the sales manager will not appear in the level below and, should you perform a sum on all employee sales, the value will not include the sales manager.

For more information on parent-child dimensions, see MSDN:

 **Parent-Child Dimensions**

<http://aka.ms/Whhuwv>

- Parent-child hierarchies:
 - Exist when you have self-referencing dimension tables
 - Are unbalanced hierarchies
 - Are changing dimensions
 - Have aggregates that are calculated at run time

Ragged Hierarchies

To the user, ragged and parent-child hierarchies often seem similar. They both have a different number of levels in separate parts of the hierarchy, but these are formed from different dimension table columns, rather than from a self-referencing relationship. For example, in a Location dimension, you might have Location, Region, State and Country levels, but the State level might only be used for the United States, Canada, and Australia. All other countries might skip the State level.

The **HideMemberIf** property makes a regular hierarchy ragged. There are five possible values for this property:

- **Never** – this value creates a regular hierarchy.
- **OnlyChildWithNoName** – this value hides a level member when it is an only child and is null or a zero-length string.
- **OnlyChildWithParentName** – this value hides a level member when it is an only child with the same name as its parent.
- **NoName** – this value hides a level member when it is null or a zero-length string.
- **ParentName** – this value hides a level member when it has the same name as its parent.

You can use a ragged hierarchy as an alternative to a parent-child hierarchy. The aggregates are calculated when the cube is processed, improving query performance. A ragged hierarchy is not, however, a changing dimension, so the flexibility of parent-child dimensions is lost.

- Some hierarchy members might not have the same number of levels
- Use **HideMemberIf** property to control ragged hierarchy behavior

Using Hierarchies

You typically create user-defined hierarchies to allow users to drill up and drill down through the data, and to allow Analysis Services to create meaningful aggregations of the measures.

There are several important hierarchy properties:

- The **IsAggregatable** property defines whether an **All** level is created. In some scenarios, an **All** level is irrelevant. For example, the **Time** dimension in a solution holds data from January 12, 1995, because earlier data is too costly to convert and gives minimal business benefit. This range of dates is not relevant to the business and does not need to be aggregated.
- The **AttributeHierarchyOrdered** property defines whether the hierarchy is ordered. If this is set to **False** and you do not query the attribute hierarchy, you will reduce processing time. If you only use an attribute hierarchy to order another attribute hierarchy, it is not queried.

- Is an **All** level relevant?
- Is the hierarchy ordered?
- Should indexes be created for the hierarchy?
- Do aggregations need to be created for the hierarchy?
- Should the hierarchy always be visible?

- The **AttributeHierarchyOptimizedState** can be set to **NotOptimized** to prevent Analysis Services from creating indexes on the attribute hierarchy. This will reduce processing time but increase query time, and should only be used when you do not query the attribute hierarchy.
- Hiding and disabling attribute hierarchies can improve performance and focus data for users. For example:
- If you set the **AttributeHierarchyEnabled** property to **False**, the hierarchy is disabled and Analysis Services only creates the attribute hierarchy as a member property. This is useful if the attribute is providing detail, but you do not want to use it as a level for aggregation.
- If you set the **AttributeHierarchyVisible** property to **False**, the attribute is only visible from user-defined hierarchies. This is useful if there are large numbers of distinct values in the attribute hierarchy that would cause confusion and not add any benefit.

Attribute Relationships

Attributes are always related directly or indirectly to the key attribute. Initially all attributes are directly related to the key attribute and this might be the relationship you require. For example, a **Product** dimension contains **ProductKey**, **Name**, **Weight**, **ProductionCost**, and **RecommendedSalesCost**. In this dimension, you do not group by any attribute, and you only need the **All** level and one other level containing every member.

In most dimensions, there are levels with relationships between them. For example, a **Customer** dimension containing attributes for

CustomerKey, **Name**, **Address**, **City**, **Region**, and **Country**. **Name** and **Address** attributes relate directly to the **CustomerKey**, but relationships have to be created for the other attributes. You form a relationship, from **Country** to **Region**, from **Region** to **City**, and from **City** to **CustomerKey**; this supports levels within the data and is used to create aggregates to improve query performance.

Attribute relationships are straightforward to create, using the Attribute Relationships tab of the Dimension Designer. You can right-click a blank space to create a dimension relationship, and then choose the source and related attributes. Each source attribute can only have one related attribute. For example, a month can only have one quarter; although a quarter can have several months. In this relationship, the month is the source attribute and the quarter is the related attribute.

- Attributes are related directly or indirectly to the key attribute
- Most dimensions have levels and relationships between the levels
- Attribute relationships are straightforward to create in Dimension Designer

Demonstration: Creating a Parent-Child Hierarchy

In this demonstration, you will see how to:

- Create a parent-child hierarchy.
- Name the levels in a parent-child hierarchy.

Demonstration Steps

Create a Parent-Child Hierarchy

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. In the D:\Demofiles\Mod03 folder, run **Setup.cmd** as Administrator.
2. Start Visual Studio and open **Demo.sln** in the D:\Demofiles\Mod03 folder.
3. In Solution Explorer, expand **Data Source Views** and double-click **Adventure Works Demo DSV.dsv**.
4. On the **Data Source View** menu, click **Add/Remove Tables**. In the **Add/Remove Tables** dialog box, add the **DimEmployee** table to the **Included objects** list and click **OK**.
5. In the data source view diagram, right-click the **DimEmployee** table and click **New Named Calculation**. In the **Create Named Calculation** dialog box, create a named calculation with the column name **Full Name** and the expression **[FirstName] + ' ' + [LastName]**, and then click **OK**.
6. In Solution Explorer, right-click **Dimensions** and click **New Dimension**.
7. On the **Welcome to the Dimension Wizard** page, click **Next**.
8. On the **Select Creation Method** page, select **Use an existing table** and click **Next**.
9. On the **Specify Source Information** page:
 - In the **Main table** box, select **DimEmployee**.
 - In the **Key columns** list, ensure that **EmployeeKey** is selected.
 - In the **Name column** box, select **Full Name**.
 - Click **Next**.
10. On the **Select Related Tables** page, clear the **Sales Territory** check box, and click **Next**.
11. On the **Select Dimension Attributes** page, click **Next**.
12. On the **Completing the Wizard** page, in the **Name** box, type **Employee** and click **Finish**.
13. Leave Visual Studio open for the next demonstration.

Name the Levels in a Parent-Child Hierarchy

1. In the **Employee.dim** Dimension Designer, on the **Dimension Structure** tab, in the **Attributes** section, click **Parent Employee Key**.
2. In the Properties pane, note that the **Usage** property for this attribute has been set to **Parent**.
3. In the **NamingTemplate** property value, click the ellipses (...).
4. In the **Level Naming Template** dialog box, in the first blank **Name** cell, type **Executive**.
5. In the next blank **Name** cell, type **Middle Management**.
6. In the next blank **Name** cell, type **Senior Employee**.

7. In the next blank **Name** cell, type **Junior Employee** then click **OK**.
8. In the Properties pane, change the **MembersWithData** property value to **NonLeafDataHidden**.
9. In the Attributes pane, click **Employee** dimension, and set the **UnknownMember** property to **None**. Select the **Employee Key** attribute, expand its **KeyColumns** property, expand the **DimEmployee.EmployeeKey** column, and set the **NullProcessing** property to **Automatic**.
10. In Solution Explorer, right-click **Demo** and click **Deploy**. If prompted for a password, type **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** user, and then click **OK**.
11. When deployment has completed, in the **Employee.dim** Dimension Designer, in the **Browser** tab, expand the employees in the hierarchy. Note that, when you select an employee, the hierarchy level name is displayed at the top of the browser area.
12. Leave Visual Studio open for the next demo.

Question: Excluding organizational charts, where else might you use parent-child hierarchies?

Lesson 3

Implementing Sorting and Grouping for Attributes

A cube can be difficult to navigate if it has numerous attributes and attribute hierarchies. In Analysis Services, you can sort by the member name, the member key, or by a related attribute. You can also group attributes. A member group is a system-generated collection of consecutive dimension members. To improve usability in Analysis Services, attribute members can be formed into member groups through a process called discretization.

Lesson Objectives

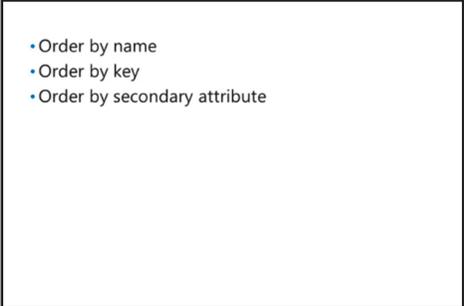
This lesson describes sorting and grouping attributes. After completing this lesson, you will be able to:

- Sort attributes.
- Group attributes.

Sorting Attributes

You can choose any attribute by which to sort a hierarchy. This attribute can be the key, the name, or any other secondary attribute in the dimension table.

- If you order by name, Analysis Services will put the members in alphanumeric order.
- If you order by key, you can specify one or more key columns and use these to sort the members. This allows you to sort dates by quarters. With a single key, the order of quarters would be Q1, Q2, Q3, and Q4 with all years aggregated. For example, Q1 would aggregate all Q1 data from every year. If you add the Year column to the key, the order would be Q1 2000, Q2 2000, Q3 2000, Q4 2000, Q1 2001, Q2 2001, Q3 2001, and so on.
- You can also order by a secondary attribute, which could be a standard or calculated column created in the data source view. For example, you might want to sort a Course dimension in which the course name runs from "Course 1" to "Course 450". The course key is provided by the course vendor and is not a relevant sort order. If you sort by name, the order is "Course 1", "Course 11", "Course 111", "Course 2", "Course 22", and so on. Therefore, you create a new column in the dimension table with the course number as a numeric field. This can then be used to sort data in the correct order.

- 
- Order by name
 - Order by key
 - Order by secondary attribute

Grouping Attributes

Some hierarchies have no natural levels, so Analysis Services only creates All and Leaf levels. This can make the cube difficult to navigate if there are many members in the hierarchy. You can use grouping to organize members into groups to simplify cube browsing. For example, customers could be grouped into income brackets rather than showing individual incomes. You can then drill down into the groups to show the detail as needed.

- Group hierarchies that have no natural levels
- Group by equal numbers or cluster the members
- Define the number of groups
- Define a naming template

To enable grouping, you must set the **DiscretizationMethod** property of the attribute. This can be set to:

- **Equal Areas** – divides members into groups of equal numbers.
- **Clusters** – uses a clustering algorithm to group members based on the training data; this can form useful groups, but has a higher processing cost.

To specify the number of groups, you must set the **DiscretizationBucketCount** property. The default is based on the square root of the number of distinct members.

To specify a naming template, you must set the **Format** option for the **NameColumn** property of an attribute. The default-naming template displays the first and last group members in the format "January – March". You can create your own naming templates to modify group names. Because naming templates are based on the members they contain, the names of groups can change as members are added or removed.

Demonstration: Using Sorting and Grouping

In this demonstration, you will see how to use Sorting and Grouping.

Demonstration Steps

Use Sorting

1. In Visual Studio, in Solution Explorer, expand **Dimensions**, and then double-click **Dim Date.dim**.
2. In the **Browser** tab, ensure **Date** is selected in the **Hierarchy** drop-down and explore the **Date** dimension, noting that the month numbers are not in numeric order.
3. On the **Dimension Structure** tab, in the **Attributes** section, click **Month Number Of Year**.
4. In the Properties pane, change the **OrderBy** property to **Key**.
5. On the **Dimension** menu, click **Process**. If you receive a message about out-of-date content, click **Yes**, and then click **Yes** to confirm you want to continue with deployment. If you are prompted for a password, type **Pa\$\$w0rd** and click **OK**.
6. In the **Process Dimension – Dim Date** dialog, click **Run**, and when the build process has completed, in the **Process Progress** dialog, click **Close**.
7. In the **Process Dimension – Dim Date** dialog, click **Close**.
8. In the **Browser** tab, ensure **Date** is selected in the **Hierarchy** menu and then, on the **Dimension** menu, click **Reconnect**.

9. Browse the **Date** dimension noting the month numbers are now sorted correctly.

Use Grouping

1. In Solution Explorer, expand **Dimensions**, and then double-click **Dim Reseller.dim**.
2. On the **Browser** tab, ensure **Resellers** is selected in the **Hierarchy** drop-down, and then browse the **Resellers** hierarchy, noting the data is presented as one long list of reseller IDs. To enable simpler location of a specific retailer ID, these will be grouped.
3. On the **Dimension Structure** tab, in the Attributes pane, click **Reseller Key**, in the Properties pane change the **DiscretizationMethod** property to **Automatic**, and then change the **DiscretizationBucketCount** to **10**.
4. On the **Dimension** menu, click **Process**. If you receive a message about out-of-date content, click **Yes**, and then click **Yes** to confirm you want to continue with deployment. If you are prompted for a password, type **Pa\$\$w0rd** and click **OK**.
5. In the **Process Dimension – Dim Reseller** dialog, click **Run**, and when the build process has completed, in the **Process Progress** dialog, click **Close**.
6. In the **Process Dimension – Dim Reseller** dialog, click **Close**.
7. On the **Browser** tab, ensure **Resellers** is selected in the **Hierarchy** drop-down and then, on the **Dimension** menu, click **Reconnect**.
8. Browse the Reseller dimension noting Reseller IDs are now grouped.

Question: When might you need to use a secondary attribute for sorting data?

Lesson 4

Slowly Changing Dimensions

In many dimension tables, the stored data never changes. For example, if a sale takes place in Japan, that location is fixed. However, in other dimensions, some attributes may change. For example, the sales representative who made that sale in Japan might move departments, offices, or work for a different manager. With changing dimensions like this one, you should consider carefully how a cube stores attribute values. For example, is it important to store old versions of attributes? In this lesson, you will see how to design and implement dimensions that might change over time.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe Type 1, Type 2, and Type 3 slowly changing dimensions.
- Explain the operations that the Slowly Changing Dimension Transformation can perform on incoming rows during a data load.
- Use the Slowly Changing Dimension Wizard to configure the Slowly Changing Dimension Transformation.

What Are Slowly Changing Dimensions?

Many dimensions have values that do not change. For example, if a customer purchases a product at noon on December 12, that time does not change, regardless of later events. However, if the sales representative who helped the customer worked in the Seattle office, that representative might later move to the Paris office. Changes such as this to the values of a dimension attribute might affect analyses based on your cube. You should ensure that your dimension design considers this. In this case, you might want the sale to remain credited to the Seattle office even after the representative moves to Paris. This would require that both the old office and the new office are stored for the representative, together with the date when that person moved.

Any dimension that is subject to such change is termed a Slowly Changing Dimension (SCD). You can take three different approaches to their design:

- **Type 1.** These changes are the simplest type of SCD to implement. Attribute values are updated directly in the existing dimension table row and no history is maintained. This makes Type 1 changes suitable for attributes that are used to provide drill-through details, but unsuitable for analytical slicers or hierarchy members where historic comparisons must reflect the attribute values as they were at the time of the fact event.
- **Type 2.** These changes involve the creation of a fresh version of the dimension entity in the form of a new row. Typically, a bit column in the dimension table is used as a flag to indicate which version of the dimension row is the current one. Additionally, datetime columns are often used to indicate the start and end of the period for which a version of the row was (or is) current. Maintaining start and end dates makes it easier to assign the appropriate foreign key value to fact rows as they are loaded

Slowly Changing Dimension Types:

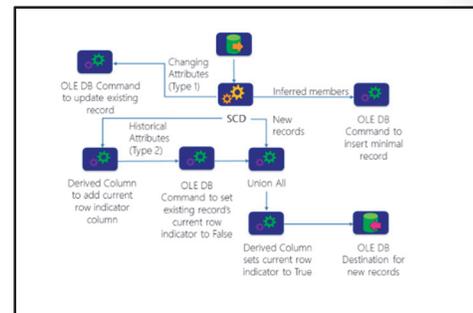
- **Type 1:** old values are overwritten
- **Type 2:** new values are written as new rows in the dimension table
- **Type 3:** new values are written in columns in the existing row

so that they are related to the version of the dimension entity that was current at the time the fact occurred.

- **Type 3.** These changes are rarely used. In a Type 3 change, the previous value (or sometimes a complete history of previous values) is maintained in the dimension table row. This requires modifying the dimension table schema to accommodate new values for each tracked attribute, and can result in a complex dimension table that is difficult to manage.

The Slowly Changing Dimension Transformation

The Slowly Changing Dimension Transformation applies the changes required by Type 1 and Type 2 SCDs during data loads. The logic required for Type 2 SCDs in particular can become complex—you should use this transformation to handle row matching, detect incoming rows that require changes, and to insert new records that are correctly linked to old values. The Slowly Changing Dimension Transformation also handles fixed attributes, which should never be changed by incoming values, and inferred members, which are members of a dimension table that do not yet exist but are referenced by data in a fact table.



 **Note:** Type 3 SCDs are rarely used and not supported by the Slowly Changing Dimension Transformation or the Slowly Changing Dimension Wizard.

The Slowly Changing Dimension Transformation has a workflow of operations that includes the following:

- A path to insert new dimension records in an SCD.
- A path to update dimension records where a changing attribute has been modified and no historical dimension data is required. This is an implementation of a Type 1 change.
- A path to insert a new record for dimension members where a historic attribute has been modified. This path also updates the current record indicator. This is an implementation of a Type 2 SCD.
- A path to insert minimal records for inferred members in the source data.

For more information about the Slowly Changing Dimension Transformation, see:

 **Slowly Changing Dimension Transformation**

<https://aka.ms/m4r621>

The Slowly Changing Dimension Wizard

Use the Slowly Changing Dimension Wizard to configure the Slowly Changing Dimension Transformation. This can be a complex task but the wizard guides you through the configuration in a straightforward manner. The wizard enables you to configure:

- The columns containing keys that can be used to look up existing dimension records in the data warehouse.
- The non-key columns that are fixed, changing, or historic attributes.
- Whether or not changes to a fixed column should produce an error or be ignored.
- The column in the dimension table that should be used to indicate the current version of a dimension member for which historic attributes have changed over time.
- Whether or not the staged data includes inferred members for which a minimal record should be inserted. Inferred members are identified based on the value of a specified column in the source.

Use the Slowly Changing Dimension Wizard to configure the Transformation and set:

- Key columns for matching old records to new records
- Non-key columns that are fixed, changing, or historic attributes
- Whether changes to a fixed column should produce an error
- The column that indicates the current version or a dimension member
- Whether to include inferred members



Best Practice: When you design and select key columns in dimension tables, ensure that they are unchanging. For example, in a product database, a product number or part number is often used as a primary key because it is unique to the product. However, a product number or part number can change and so should be an attribute of an SCD. If you want to store both old and new product numbers for later analysis, you need a Type 2 SCD and two rows with different product numbers. Ensure that the Slowly Changing Dimension Transformation can identify these records as relating to the same product by using a different identity value for the primary key in the dimension table.

Question: You are implementing a Customer dimension in a cube. Should you implement this as a fixed dimension, a Type 1 SCD, or a Type 2 SCD?

Lab: Working with Cubes and Dimensions

Scenario

You have created a cube and tested it with a small set of business users whose feedback says that the cube shows potential but is difficult to navigate intuitively. To resolve this problem, you plan to refine the cube's dimensions to include the attributes by which users want to aggregate data, and hierarchies that make it easy to do so at multiple levels.

Objectives

After completing this lab, you will be able to:

- Implement dimensions in a cube.
- Configure sorting and grouping of dimensions.

Estimated Time: 90 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTURWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Configuring Dimensions

Scenario

Several users at Adventure Works Cycles want to be able to analyze data across time periods. You need to configure the Date dimension as a Time dimension so that Analysis Services can apply temporal calculations to values.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Remove Unused Attributes
3. Add Dimension Intelligence

► Task 1: Prepare the Lab Environment

1. Ensure the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab03\Starter folder as Administrator.

► Task 2: Remove Unused Attributes

1. Open the **Adventure Works OLAP.sln** solution in the D:\Labfiles\Lab03\Starter folder in Visual Studio.
2. Open the **Customer** dimension in the Dimension Designer and note that many attributes have been added to allow business users to aggregate measures in many different ways. However, users have complained that some of these attributes are unnecessary and that they should be removed to make browsing the cube simpler.
3. Delete the **Commute Distance**, **Number Cars Owned**, and **Number Children At Home** attributes from the **Customer** dimension.
4. Similar feedback has been received about the **Product** dimension, so delete the **Days To Manufacture** and **Safety Stock Level** attributes from the **Product** dimension.

► Task 3: Add Dimension Intelligence

1. Use the Business Intelligence Wizard on the **Date** dimension to define dimension intelligence.
2. Specify that the dimension is a **Time** dimension.
3. Map the dimension attribute columns as follows:
 - **Year:** Calendar Year
 - **Half Year:** Calendar Semester
 - **Quarter:** Calendar Quarter
 - **Month:** Full Date Alternate Key
 - **Date:** Month

Results: After this exercise, unused attributes in the Customer and Product dimensions will have been removed; time intelligence will have been added to the Date dimension.

Exercise 2: Defining Relationships and Hierarchies

Scenario

Business users need to view aggregations at different levels. Specifically, analysts want to view business results for product categories, and then drill down to see details for subcategories and individual products. Analysts also want to view customers by gender and then, for each gender, view business results based on marital status.

The main tasks for this exercise are as follows:

1. Create a Natural Hierarchy
2. Create a Non-Natural Hierarchy
3. Create a Hierarchy with Attribute Relationships

► Task 1: Create a Natural Hierarchy

1. Edit the **Product.dim** dimension to create a hierarchy that contains the following attributes:
 - English Product Category Name
 - English Product Subcategory Name
 - English Product Name

Note: if a warning is displayed, notifying you that attribute relationships do not exist and performance may be decreased, ignore it. You will see how to use attribute relationships to optimize a hierarchy later in this lab.

2. Rename the hierarchy to **Categorized Products**.
3. Rename the hierarchy levels as follows:
 - Category
 - Subcategory
 - Product
4. Set the **AttributeHierarchyVisible** property of all attributes to **False** so that the **Categorized Products** hierarchy is the only way to browse the **Product dimension**.

5. Process the dimension when you have finished.
 - If you are prompted to redeploy the project, do so.
 - If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** account.
6. Browse the **Categorized Products** hierarchy in the dimension browser.
7. Save the dimension.

► Task 2: Create a Non-Natural Hierarchy

1. In the **Customer** dimension, create a hierarchy named **Gender-Marital Status** that includes the following attributes:
 - Gender
 - Marital Status

Note: if a warning is displayed, saying that attribute relationships do not exist and that performance may be decreased, ignore it. You will see how to use attribute relationships to optimize a hierarchy later in this lab.

2. Process the dimension when you have finished:
 - If you are prompted to redeploy the project, do so.
 - If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** account.
3. Browse the **Gender-Marital Status** hierarchy in the dimension browser. Note that gender can be **F**, **M**, or **Unknown**, and that marital status can be **M** or **S**.
4. Save the dimension when you have finished.

► Task 3: Create a Hierarchy with Attribute Relationships

Configure Attribute Column Bindings

1. In the Dimension Designer for the **Date** dimension, modify the **Calendar Semester** attribute so that a semester is uniquely identified by both the calendar year and semester, but displayed using only the calendar semester value. To do this:
 - In the properties for the **Calendar Semester** attribute, set the **KeyColumns** property so that the key columns are **CalendarYear** followed by **CalendarSemester**.
 - Set the **NameColumn** property so that the source column value is **CalendarSemester**.
 - Set the **ValueColumn** property so that the source column value is **CalendarSemester**.
2. Modify the **Calendar Quarter** attribute so that a quarter is uniquely identified by both the calendar year and the calendar quarter, but displayed using only the calendar quarter value. To do this:
 - In the properties of the **Calendar Quarter** attribute, set the **KeyColumns** property so that the key columns are **CalendarYear** followed by **CalendarQuarter**.
 - Set the **NameColumn** property so that the source column value is **CalendarQuarter**.
 - Set the **ValueColumn** property so that the source column value is **CalendarQuarter**.

3. Modify the **Month** attribute so that a month is uniquely identified by both the calendar year and the month number of year, but displayed using the month name value. To do this:
 - Set the **KeyColumns** property so that the key columns are **Calendar Year** followed by **MonthNumberOfYear**.
 - Set the **NameColumn** property so that the source column value is **EnglishMonthName**.
 - Set the **ValueColumn** property so that the source column value is **EnglishMonthName**.

Create Attribute Relationships

1. In the **Date** dimension, create the following attribute relationships:
 - **Full Date Alternate Key > Month** (Rigid)
 - **Month > Calendar Quarter** (Rigid)
 - **Calendar Quarter > Calendar Semester** (Rigid)
 - **Calendar Semester > Calendar Year** (Rigid)
2. Save the dimension when you have finished.

Create a Hierarchy

1. Modify the **Date** dimension structure to create a new hierarchy named **Calendar Date** that contains the following levels:
 - Calendar Year
 - Calendar Semester
 - Calendar Quarter
 - Month
 - Full Date Alternate Key (renamed to **Day**)
2. Set the **AttributeHierarchyVisible** property of the attributes that are included in the **Calendar Date** hierarchy to **False** so that they can only be used by browsing the hierarchy.
3. Process the dimension:
 - If you are prompted to redeploy the project, do so.
 - If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** account.

Results: After this exercise, you should have created a Categorized Products hierarchy and a Gender-Marital Status hierarchy.

Exercise 3: Sorting and Grouping Dimension Attributes

Scenario

Having configured a Date dimension, it has been reported that months are being sorted alphabetically rather than chronologically. In this lab, you will configure the date dimension so that months are sorted chronologically.

The main tasks for this exercise are as follows:

1. Sort Attribute Members
2. Group Attribute Members

► Task 1: Sort Attribute Members

1. Browse the **Calendar Date** hierarchy in the dimension browser. Note that months are displayed in alphabetical, rather than chronological order.
2. Modify the **Month** attribute to set its **OrderBy** property to **Key**. The key for this attribute consists of the **Calendar Year** and **Month Number of Year** columns, so sorting by key should ensure that months are displayed in chronological order.
3. Reprocess the dimension and verify that the months are now displayed in the correct order.

► Task 2: Group Attribute Members

1. Open the **Customer** dimension in Dimension Designer, and explore the data in the **Customer** table. Notice the range of values for the **YearlyIncome** column.
2. Process the dimension and browse the **Yearly Income** hierarchy.
 - If you are prompted to redeploy the project, do so.
 - If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for the **ADVENTUREWORKS\ServiceAcct** account.
3. Browse the dimension and view the **Yearly Income** hierarchy. Note that this hierarchy has no structure.
4. Modify the following properties of the **Yearly Income** attribute:
 - **DiscretizationMethod**: Automatic
 - **DiscretizationBucketCount**: 5
 - **OrderBy**: Key
5. Reprocess the dimension and browse the **Yearly Income** hierarchy to verify that the values are now grouped into five income bands, with a sixth band for customers whose income is unknown.

Results: You will have a multidimensional project with enhanced dimensions.

Module Review and Takeaways

Dimensions are a fundamental component of cubes. Facts or measures have the values that interest you, but dimensions provide the business context by which you aggregate these measures. Dimensions organize your data by category and show results grouped by these categories—such as product, customer, or month.

Microsoft SQL Server Analysis Services dimensions contain attributes that correspond to columns in dimension tables. These attributes appear as attribute hierarchies and can be organized into user-defined hierarchies, or defined as parent-child hierarchies, based on columns in the underlying dimension table. Hierarchies are used to organize measures contained in a cube.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 4

Working with Measures and Measure Groups

Contents:

Module Overview	4-1
Lesson 1: Working with Measures	4-2
Lesson 2: Working with Measure Groups	4-6
Lab: Configuring Measures and Measure Groups	4-14
Module Review and Takeaways	4-18

Module Overview

Measures and measure groups specify the data values that your cube can aggregate to provide summary values for analysis. This module describes measures and measure groups. It also explains how you can use them to define fact tables and associate dimensions with measures.

Objectives

After completing this module, you will be able to:

- Configure measures.
- Configure measure groups.

Lesson 1

Working with Measures

This lesson explains how to work with measures, including configuring how measures are displayed and aggregated, and information about measure properties.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the concept of a measure.
- List the properties of measures.
- Configure measure formatting.
- Choose appropriate aggregation functions for required additivity.

Introducing Measures

A measure represents a column that contains quantifiable data, usually numeric, that you can aggregate. Typically, measures occur as a column in the fact table and are the values that the business user finds most interesting. Measures are organized into measure groups and are distinct from dimensions.

For example, in a retail cube that has dimensions for stores, products, time, and salespeople, the measures might include sales value and units. These measures are then aggregated to provide total sales units and value for each store, product, day, and salesperson.

To provide specific aggregated details, you might select a store from the stores dimension, a product from the products dimension, a day from the time dimension, a salesperson from the salesperson dimension and, from the measures dimension, either sales units or sales value.

You create measures as part of the Cube Wizard or on the **Cube Structure** tab of the Cube Designer.

Calculated Measures

You can create calculated measures based on other measures in the fact table. You use a “measure expression” to define the value of a calculated measure, based on a fact table column as modified by a Multidimensional Expressions (MDX) statement. You should consider the performance impact of using calculated measures because they are worked out at run time. If a calculated measure is infrequently used and is quite simple—for example, the sum of two measures—it will have little performance impact on queries, but might reduce processing times. If a measure is frequently used and involves a complex calculation, it might cause unacceptable query times.

Measures from Attribute Columns

You can use attribute columns from dimension tables to define measures. However, these are typically semiadditive or nonadditive, which means that they cannot be fully aggregated across all dimensions. You will learn more about semiadditive and nonadditive measures later in this lesson.

- Represents a column containing quantifiable data
- Usually from a fact table
- Organized into measure groups
- Can be aggregated
- Typically numeric and aggregated
- Calculated measures
- Measures from attribute columns
- Aggregation function
- Granularity

Aggregation Function

The aggregation behavior of each measure is determined by the aggregation function that is associated with the measure. You will learn more about aggregations later in this lesson.

Granularity

You should consider the granularity of measures, which refers to how specific something is. For example, an hour is more granular than a day, and a day is more granular than a week. You cannot make a measure more granular than the highest level of granularity in the dimension tables, but you can make it less so. For example, if you have a time dimension that stores data at the levels of day, month, quarter, and year, you cannot store sales measures at a granularity of hours, but you can store them at a granularity of months. As you will see later in this module, you can specify the granularity of a measure group in relation to a specific dimension.

Properties of Measures

Measures have properties that enable you to define how the measures function and to control how the measures appear to users. You can configure measures on the **Cube Structure** tab of the Cube Designer.

Measures inherit certain properties from the measure group of which they are a member, unless those properties are overridden at the measure level.

Measure properties determine:

- How measures are aggregated.
- The data type of the column in the underlying fact table to which the measure is bound.
- The column in the data source view to which the measure is bound.
- The description of the measure, which might be exposed in client applications.
- The name that is displayed to the user.
- The folder in which the measure will appear when users connect to the cube.
- The display format of the measure.
- The unique identifier (ID) of the measure; note that this property is read-only.
- Any MDX expressions that define the measure.
- The visibility of the measure.

Measure properties determine:

- How measures are aggregated
- The column in the data source view to which the measure is bound
- The name that is displayed
- The folder in which the measure will appear when users connect to the cube
- The display format
- Any MDX expressions that define the measure
- The visibility of the measure

Configuring How Measures Are Displayed

To determine how values of a measure are displayed to users, by using the Properties window of the measure in the Cube Designer, you can configure the measure property **FormatString**.

Typical **FormatString** values include **Standard** or **Currency**, but, although a list of display formats is provided, you can specify many additional formats that are not in the list by specifying any named or user-defined format that is valid in Microsoft® Visual Basic®.

The table below shows some examples of formats that can be specified.

Display Format Value	Example Output
Standard	123456.78
Currency	\$123,456.78
0	123457
\$#,#.00	\$123,456.78
#,#0.000	123,456.789

The last two rows in the above table show user-defined formats that have been entered, rather than chosen from the list. In this example, the regional setting in Control Panel on the client computer is English (United States).



Note: The currency format will display the currency symbol of the client's location.

Therefore, the value may be incorrect and, in most scenarios, it is more appropriate to employ a user-defined **FormatString** property.

Aggregation Functions

Aggregation functions determine the function that is used to aggregate the measure. Before applying an aggregation function to a measure, you should consider how the measure will aggregate along all of the dimensions in the measure group that contains the measure. This is called the additivity of the measure, of which there are three types:

- **An additive measure.** An additive measure, also called a fully additive measure, can be aggregated along all of the dimensions included in the measure group that contains the measure, without restriction. Additive measures are the most commonly used in SQL Server Analysis Services.

- Select or enter **Formatstring** property
- Named or user-defined format
- User-defined format for currency

- Additivity of measures:
 - Additive
 - Semiadditive
 - Nonadditive
- All aggregate functions are additive, semiadditive, or nonadditive

- **A semiadditive measure.** A semiadditive measure can be aggregated along some, but not all, dimensions that are included in the measure group that contains the measure. An example of a semiadditive measure is a units-in-stock value for warehouse goods. This would be additive by region to reveal the total units in stock, but would not be additive by time because, if you add last week's stock to this week's stock, the resulting value could be twice as much stock as you actually have.
- **A nonadditive measure.** A nonadditive measure cannot be aggregated along any dimension in the measure group that contains the measure. Instead, you must individually calculate the measure for each cube cell that represents the measure. For example, a calculated measure that returns a percentage, such as profit margin, cannot be aggregated from the percentage values of child members in any dimension.

By default, the **AggregateFunction** property is set to **Sum**. You can change the **AggregateFunction** property for a measure in the Properties window of the measure in the Cube Designer. All aggregation functions are additive, semiadditive, or nonadditive.

Commonly used aggregation functions include:

- The **Sum** function. This is additive and calculates the sum of all values for every child member. **Sum** is the default function.
- The **Count** function. This is additive and calculates the quantity of child members.
- The **Min** function. This is semiadditive and calculates the lowest value for each child member.
- The **Max** function. This is semiadditive and calculates the highest value for each child member.
- The **DistinctCount** function. This is nonadditive and calculates the count of all unique child members.
- The **None** function. This is nonadditive, performs no aggregation, and supplies values directly from the fact table.

Other semiadditive aggregation functions include:

- The **ByAccount** function. This calculates aggregation according to the aggregation function that is assigned to the account type for a member in an account dimension.
- The **AverageOfChildren** function. This calculates the average of values for all non-empty child members.
- The **FirstChild** function. This retrieves the value of the first child member.
- The **LastChild** function. This retrieves the value of the last child member.
- The **FirstNonEmpty** function. This retrieves the value of the first non-empty child member.
- The **LastNonEmpty** function. This retrieves the value of the last non-empty child member.

Question: Give an example of scenarios where you would use an additive measure, a semiadditive measure, and a nonadditive measure.

Lesson 2

Working with Measure Groups

You use measure groups to associate dimensions with measures. Properties that you can configure at the measure group level include:

- **Partitions.** These are containers for a portion of the measure group data.
- **Aggregations.** These are precalculated summaries of data from leaf cells that improve query response times.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe measure groups.
- List the properties of measure groups.
- Describe the relationships between measure groups and dimensions.
- Describe aggregations and their effect on performance.
- Explain partitions and measure group storage.
- Explain how to configure measure group storage.

Introducing Measure Groups

In a cube, measures are sorted into measure groups by their underlying fact tables. You use measure groups to associate dimensions with the measures in the fact table.

A measure group defines:

- The dimensions by which the measures can be aggregated. This is achieved by creating relationships between measure groups and dimensions in the cube. For example, you might relate an **Internet Sales** measure group to **Product**, **Date**, and **Customer** dimensions. In the same cube, you might relate a **Reseller Sales** measure group to **Product**, **Date**, and **Reseller** dimensions. Measures in both groups can be aggregated by **product** and **date**, **Internet Sales** measures by customer, and **Reseller Sales** measures by reseller.
- The granularity at which the measures are related to the dimensions. For example, a cube might include a **Reseller Sales** measure group that is related to the **Date** dimension based on a key for an individual day (because each sale is associated with the specific date on which it occurred), and a **Sales Quota** measure group related to the **Date** dimension based on the year (because sales quotas are set for each year, not for individual days, weeks, or months).
- How measure aggregations should be stored. For example, you might optimize cube performance by storing preaggregated values for some or all of the possible aggregations for measures in a measure group and its related dimensions.

- Measure groups are collections of related measures
- A measure group defines:
 - The dimensions by which the measures can be aggregated
 - The granularity at which the measures are related to the dimensions
 - How measure aggregations should be stored
- By default, a measure group is created for each fact table

When you create a cube by using the Cube Wizard in SQL Server Data Tools, you define one or more measure groups. By default, the Cube Wizard will create one measure group per fact table. You can add and configure measure groups for existing cubes by using the Cube Designer.



Note: The **DistinctCount** aggregation looks for the number of distinct values in a fact table column. For example, you might want to know how many sales staff sold a particular product. This query is very resource-intensive, so SQL Server Analysis Services creates a separate group for **DistinctCount** measures.

Measure Group Properties

Measure group properties determine behaviors for the entire measure group and set default behaviors for certain properties of measures within a measure group. You can access the properties of a measure group from the Properties window of the measure group in the Cube Designer. Measure group properties include:

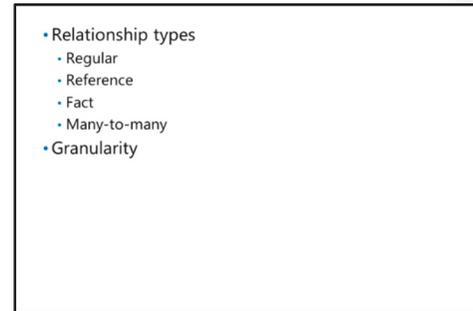
- **AggregationPrefix.** This specifies the common prefix that is used for all aggregation names of the measure group.
- **DataAggregation.** This determines whether SQL Server Analysis Services can aggregate persisted data or cached data for the measure group. The default value is **DataAndCacheAggregatable**, which will aggregate persisted and cached data. You can change this to **CacheAggregatable** or **DataAggregatable**. This can be useful on systems that have limited RAM because the other two settings will attempt to keep the aggregations in memory to improve performance.
- **ErrorConfiguration.** This provides configurable error-handling settings for handling of duplicate keys, unknown keys, null keys, error limits, action upon error detection, and the error log file.
- **EstimatedRows.** This specifies the estimated number of rows in the fact table. You use this when you design aggregations. The value is static and should be kept approximately accurate, either manually or by updating the count in the Aggregation Design Wizard.
- **EstimatedSize.** This specifies the estimated byte size of the measure group. You use this when you design aggregations.
- **ID.** This specifies the identifier of the object.
- **IgnoreUnrelatedDimensions.** This determines the output when dimensions are used that are not related to the measure group. If this is set to **True**, the output will show the top-level aggregation for the dimension. If this is set to **False**, the output will be blank for the dimension.
- **ProactiveCaching.** This defines how data is reprocessed when there are changes in the underlying data source. You will learn more about proactive caching later in this module.
- **ProcessingMode.** This indicates whether indexing and aggregating should occur during or after processing. You can use lazy processing to enable SQL Server Analysis Services to choose a time when workload is low, but you will not know when the indexing workload will occur, or when indexing has taken place and queries are at peak performance.

- AggregationPrefix
- DataAggregation
- ErrorConfiguration
- EstimatedRows and EstimatedSize
- ID
- IgnoreUnrelatedDimensions
- ProactiveCaching
- ProcessingMode and Processing Priority
- StorageLocation and StorageMode
- Type

- **ProcessingPriority.** This determines the processing priority of the cube during background operations, such as lazy aggregations and indexing.
- **StorageLocation.** This is the file system storage location for the measure group. If none is specified, the location is inherited from the cube that contains the measure group.
- **StorageMode.** This determines the storage mode for the measure group. You will learn more about storage modes later in this module.
- **Type.** This specifies the type of the measure group.

Relationships Between Measure Groups and Dimensions

A relationship between a dimension and a measure group consists of the dimension and fact tables participating in the relationship and a granularity attribute, which specifies the granularity of the dimension in the particular measure group. This relationship is configured on the **Dimension Usage** tab of the Cube Designer.



 **Note:** When a measure group is updated, dimension relationship information is unavailable until the cube is processed.

Relationship Types

There are four types of relationship between dimension and fact tables:

- **Regular.** A regular dimension relationship represents the relationship between dimension tables and a fact table in a traditional star schema design. The key column for the dimension is joined directly to the fact table.
- **Reference.** A reference dimension relationship represents the relationship between dimension tables and a fact table in a snowflake schema design. The key column for the dimension is joined indirectly to the fact table.
- **Fact.** If a dimension consists entirely of values in a fact table and does not have a separate dimension table, it has a fact dimension relationship.
- **Many-to-many.** Typically, every dimension member relates to many facts, but all of those relate to just one member in each dimension. This is a standard one-to-many relationship. You might also have many-to-many relationships when, for example, a customer has multiple reasons for making a purchase, each reason having numerous customers. In a relational database, this relationship would be achieved through an intermediate table. In SQL Server Analysis Services, a similar technique is used by joining the dimension to an intermediate fact table, by joining the intermediate fact table to an intermediate dimension, and by joining the intermediate dimension to the fact table.

 **Note:** Foreign-key relationships between all tables that are involved in a many-to-many relationship must exist in the underlying data source view.

Granularity

By default, a measure group will use the highest level of granularity. For example, measure groups are often related to a date dimension, in which the highest level of granularity is day level. This is suitable for some measure groups, but in other measure groups, the data in the fact table might not have that level of detail, in which case you will need to change the granularity to another level, such as month or quarter. To change the granularity of a particular relationship, you can use the **Granularity attribute** list.

Demonstration: Defining Relationships Between Dimensions and Measure Groups

In this demonstration, you will see how to define a referenced relationship.

Demonstration Steps

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. In the D:\Demofiles\Mod04 folder, run **Setup.cmd** as Administrator.
2. Start SQL Server Data Tools, and then open **Demo.sln** in the D:\Demofiles\Mod04 folder.
3. On the **Build** menu, click **Deploy Solution**. If you are prompted, specify the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**.
4. Wait for the **Deployment Completed Successfully** message in the Deployment Progress – Demo window, and then close the Deployment Progress – Demo window.
5. In Solution Explorer, double-click **SalesDemo.cube**, and on the **Browser** tab, expand **Measures**, expand **Reseller Sales**, and then drag the **Revenue** measure to the query results area. Expand the **Geography** dimension, and then drag **Country-Region** to the left of the **Revenue** value. Note that the values for each region are the same. The aggregation is incorrect.
6. In Solution Explorer, expand dimensions, right-click **Geography.dim**, and then click **View Designer**. Note that this dimension is based on the **Geography** table and has a **Country-Region** attribute.
7. In Solution Explorer, right-click **Reseller.dim**, and then click **View Designer**. Note that this dimension is based on the **DimReseller** table, which includes a **GeographyKey** attribute that relates it to the **Geography** table.
8. Click the tab for the **SalesDemo** cube, and on the **Cube Structure** tab, in the Data Source View pane, note that there is no direct relationship between the **Reseller Sales** fact table and the **Geography** dimension table.
9. On the **Dimension Usage** tab, click the intersection of the **Reseller Sales** measure group and the **Geography** dimension, and then click the ellipsis (...) button.
10. In the **Select relationship** type list, select **Referenced**.
11. In the **Intermediate dimension** list, select **Reseller**.
12. In the **Reference dimension attribute** list, select **Geography Key**, in the **Intermediate dimension attribute** list, select **Geography Key**, and then click **OK**.
13. On the **Dimension Usage** tab, in the **Dimensions** list, right-click **Geography**, and then click **Rename**.
14. Change the name of this cube dimension to **Reseller Geography**, and then press Enter to make this name change take effect.

15. On the **Build** menu, click **Deploy Solution**. If you are prompted, specify the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**.
16. Wait for the **Deployment Completed Successfully** message in the Deployment Progress – Demo window, and then close the Deployment Progress – Demo window.
17. On the **Browser** tab for the **SalesDemo** cube, click the **Reconnect** button.
18. Right-click the query results area, and then click **Clear Grid**.
19. Expand **Measures**, expand **Reseller Sales**, and then drag the **Revenue** measure to the query results area. Expand the **Reseller Geography** dimension, and then drag **Country-Region** to the left of the **Revenue** value. Note that the values for each region are now correct.

Aggregations

Aggregations are precalculated summaries of leaf cell data that improve query response time by preparing the answers before the questions are asked.

A simple **Aggregation** object consists of:

- Basic information that includes the name of the aggregation, the ID, annotations, and a description.
- A collection of **AggregationDimension** objects that contain the list of granularity attributes for the dimension.

- Precalculated summaries of leaf data
- Give multidimensional cubes their performance benefits
- Typically provide the best performance when only a proportion are precalculated
- Set aggregations initially by using the Aggregation Design Wizard
- Improve at a later time by using the Usage-Based Optimization Wizard

Aggregations give multidimensional cubes their performance benefits. They precalculate the answer to queries at processing time so that, when the query is run, performance is hugely improved. If you consider that a query might have to aggregate thousands of individual values to produce one summarized output value, of which there may be hundreds, it becomes clear that precalculating these results will have enormous benefits.

It might seem appropriate to precalculate every possible result, but some aggregations provide little benefit. SQL Server Analysis Services incorporates a sophisticated algorithm to select aggregations for precalculation so that other aggregations can be quickly computed from the precalculated values. For example, if SQL Server Analysis Services stores aggregates at a month level, it takes little processing to generate quarter data at query run time. The Aggregation Design Wizard gives options for you to specify storage and percentage constraints on the algorithm to achieve a satisfactory trade-off between query response time and storage requirements. Typically, there is little benefit from preprocessing more than 30 percent of aggregations.

After using the Aggregation Design Wizard, you can further improve aggregation design for your specific system by using SQL Server Analysis Services with normal workloads for some time, and then running the Usage-Based Optimization Wizard. This wizard is almost identical to the Aggregation Design Wizard, but bases optimization on queries that are submitted to SQL Server Analysis Services and should, therefore, give better query response with less processing time.

To use the Aggregation Design Wizard or the Usage-Based Optimization Wizard for a measure group, in the Cube Designer, on the **Aggregations** tab, right-click the measure, and then click **Design Aggregations** or **Usage Based Optimization**.

Partitions

A partition is a container for all, or a portion, of the measure group data. When a measure group is created, it will have one partition. You can then add further partitions to the measure group.

SQL Server Analysis Services Enterprise Edition uses partitions to manage and store data and aggregations for a measure group in a cube. By splitting the data into multiple partitions, there are performance benefits:

- Can split a measure group into multiple partitions
- Performance benefits:
 - Incremental update
 - Store frequently used data differently
 - Distribute across multiple drives
- Partition a measure group horizontally or vertically

- You can speed up cube updates by using an incremental update. An incremental update is where one partition is updated rather than the whole measure group. For example, if you had a different partition for each month, and are only updating the most recent month, you only need to process the latest partition, which reduces processing time.
- You can store frequently used data in a different way from aggregated data. This means you can use higher levels of aggregation and a faster storage mode to improve query times for the frequently used data, while reducing processing times and storage requirements for older data.
- Partitions enable the source data and aggregate data of a cube to be distributed across multiple hard drives and among multiple server computers. You can use this to greatly improve performance on large cubes.



Note: When you incrementally update a partition, a temporary partition is created that has an identical structure to the source partition. The temporary partition is then merged with the original partition. There are possible scenarios that could affect data integrity when you incrementally update individual partitions. For example, if you had a different partition for each month, and the date of a record in the source data changed to or from the most recent month, the data may either disappear or be duplicated. You should consider these scenarios and plan accordingly.

A simple **Partition** object consists of:

- Basic information that includes the name of the partition, in addition to the storage and processing modes.
- A slicing definition that is an MDX expression that specifies a tuple or a set.
- An aggregation design that is a collection of aggregation definitions that you can share across multiple partitions.

Typically, you can partition a measure group horizontally or vertically.

In a horizontally partitioned measure group, each partition is based on a separate fact table that can be from different data sources.

A vertically partitioned measure group is based on a single table, with each partition centered on a source system query that filters data for the partition.

Configuring Measure Group Storage

There are three storage modes in SQL Server Analysis Services: relational online analytical processing (ROLAP), multidimensional online analytical processing (MOLAP), and hybrid online analytical processing (HOLAP).

ROLAP

ROLAP storage stores the aggregations in indexed relational data, along with the source data. It does not use multidimensional storage, so ROLAP data is slower to query and process than MOLAP or HOLAP, but it enables real-time access to data and uses less storage space.

MOLAP

MOLAP is the most commonly used storage mode. It stores both the aggregations and a copy of the source data in the multidimensional cube. MOLAP gives the greatest performance, but requires more storage space due to the duplication. There is latency when you use MOLAP storage because the cube data is refreshed only when the cube is processed, so changes from the data source are only updated periodically.

You can use proactive caching with MOLAP storage. When proactive caching is used, the server is notified when changes are made to the data source, and then the changes are incorporated. While the cache is rebuilt with new data, you can choose to send queries to ROLAP data, which is up to date, but slower, or to the original MOLAP storage, which is faster, but will not have the new data.

If you only need daily updates, you can set processing to occur every 24 hours. You can schedule processing for when the cube is rarely used, for example, at night—however, some systems are in use 24 hours a day, seven days a week.

For maximum performance, you can use MOLAP without proactive caching or scheduling. This is best suited to cubes that do not need the most up-to-date data. When objects are updated, some downtime will be required. You can minimize this by processing the cube to a staging server, and using database synchronization to copy the processed data to the production server.

HOLAP

HOLAP storage stores aggregations in the multidimensional cube and leaves source data in the relational database. This can provide a good compromise when leaf-level data is rarely accessed. If leaf-level data is frequently accessed, MOLAP would provide improved performance.



Note: You should also consider tabular data model storage, particularly as an alternative to ROLAP and HOLAP. The tabular data model is covered in more detail in Module 7 of this course, “Implementing a Tabular Data Model by Using Analysis Services.”

Storage Settings

You can configure storage at partition level by using the **Storage Settings** dialog box. This enables you to maximize efficiency by using configurations that are most suitable for each partition.

The **Storage Settings** dialog box contains the following standard storage configurations, which are combinations of storage mode and caching options:

- Storage modes
 - ROLAP
 - MOLAP
 - HOLAP
- Storage settings
 - Partition level
 - Standard storage configurations
 - Apply to existing partitions, new partition defaults, or new measure group defaults

- **Real Time ROLAP.** Use to set to ROLAP storage mode.
- **Real Time HOLAP.** Use to set to HOLAP storage mode.
- **Low Latency MOLAP.** Use to set to MOLAP storage mode with proactive caching and a latency target of 30 minutes after first change—queries revert to ROLAP during update.
- **Medium Latency MOLAP.** Use to set to MOLAP storage mode with proactive caching and a latency target of four hours after first change—queries revert to ROLAP during update.
- **Automatic MOLAP.** Use to set to MOLAP storage mode with proactive caching and a latency target of two hours after first change—during update, queries use MOLAP data that was cached before update started.
- **Scheduled MOLAP.** Use to set to MOLAP storage mode with updates scheduled every 24 hours.
- **MOLAP.** Use to set MOLAP without proactive caching or scheduled updates.

You can use the **Storage Settings** dialog box to apply these settings to existing partitions, default new partitions in a measure, and default new measure groups within a cube.

- To configure storage for an existing partition, in the Cube Designer, on the **Partitions** tab, expand the measure group, right-click the partition, and then click **Storage Settings**. This displays the **Storage Settings** dialog box for the selected partition.
- To configure default storage settings for new partitions that are added to a measure group, in the Cube Designer, on the **Partitions** tab, expand the measure group, and then click the **Storage Settings** link for that measure group. This displays the **Storage Settings** dialog box for the selected measure group.
- To configure default storage settings for new measure groups that are added to a cube, in the Cube Designer, on the **Cube Structure** tab, in either the Measures or Dimensions pane, click the cube object. In the Properties window, click the ellipsis (...) button for the **ProactiveCaching** property setting. This displays the **Storage Settings** dialog box for the selected cube.

Check Your Knowledge

Question	
You are configuring the storage settings for a measure group. The data in the group does not need to be updated regularly. However, the data must be available at all times, and queries that access the data must execute with the best possible performance. What is the most suitable storage setting to apply to the measure group?	
Select the correct answer.	
<input type="checkbox"/>	Real Time ROLAP
<input type="checkbox"/>	Real Time HOLAP
<input type="checkbox"/>	MOLAP
<input type="checkbox"/>	Automatic MOLAP
<input type="checkbox"/>	Medium Latency MOLAP

Lab: Configuring Measures and Measure Groups

Scenario

Business users are analyzing Internet sales by using the multidimensional cube that you have created, but they also need to analyze reseller sales. You have reseller sales data in the data warehouse on which the cube is based, so you plan to add a second measure group that contains reseller measures.

Users have also requested that you remove some measures that they do not need to use when analyzing data, and also ensure that the remaining measures are clearly named.

Users specifically need to analyze reseller sales by product, so you must ensure that the cube supports the required relationship between the new reseller sales measure group and the **Product** dimension.

Finally, the data center administrator is concerned about the disk space that the cube uses, but your users are worried about performance when they analyze Internet sales. You must optimize the storage and aggregations of the cube to balance these concerns.

Objectives

After completing this lab, you will be able to:

- Configure measures.
- Define a regular relationship.
- Configure measure group storage.

Estimated Time: 60 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Configuring Measures

Scenario

You are refining the online analytical processing (OLAP) cube for your company. Users can currently view measures for Internet sales, but you have additional reseller sales data in your data warehouse that they would like to analyze. When you add the reseller sales measures, your users want you to remove any measures that are not required for business analysis, and ensure that measures in the cube are clearly named.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create a Measure Group
3. Modify Measure Groups

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab04\Starter folder as Administrator.

► Task 2: Create a Measure Group

1. Start SQL Server Data Tools and open the **Adventure Works OLAP.sln** solution in the D:\Labfiles\Lab04\Starter folder.
2. In the **Adventure Works DSV** data source view, add the **FactResellerSales** table, together with all related tables.
3. In the **Sales** cube, add a new measure group named **Reseller Sales** based on the **FactResellerSales** table.
4. Review the names of the measures in the **Reseller Sales** measure group. Note that, when the **Reseller Sales** measure group was set up, measures were created for all of the numerical fields in the **FactResellerSales** table.

► Task 3: Modify Measure Groups

1. Delete the following measures from the cube:
 - Revision Number
 - Unit Price
 - Extended Amount
 - Unit Price Discount Pct
 - Discount Amount
 - Product Standard Cost
 - Tax Amt
 - Freight

Tip: click the **Show Measures Grid** icon to view all of the measures in the cube as a grid. In this view, you can select several measures at the same time by holding down the Ctrl key.

2. Rename the following measures:
 - Order Quantity (rename to **Reseller Order Quantity**).
 - Total Product Cost (rename to **Reseller Cost**).
 - Sales Amount (rename to **Reseller Revenue**).
 - Fact Reseller Sales Count (rename to **Reseller Sales Count**).

Results: After this exercise, you should have created a new measure group for the **FactResellerSales** table, removed unrequired measures, and renamed measures.

Exercise 2: Defining a Regular Relationship

Scenario

Your users have specifically stated that they must be able to analyze reseller sales by product, so you must create the required relationships to support aggregating reseller sales measures by the **Product** dimension.

The main tasks for this exercise are as follows:

1. View Existing Dimensions for Measure Groups
2. Create a Dimension

► Task 1: View Existing Dimensions for Measure Groups

1. Deploy the Adventure Works OLAP solution, typing the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd** if you are prompted.
2. Browse the **Sales** cube and review the measures and dimensions that are available by selecting each measure group:
 - When the **Internet Sales** measure group is selected, the **Customer** dimension should be listed.
 - When the **Reseller Sales** measure group is selected, there is no **Customer** dimension, because reseller sales are sold to resellers, which are defined in a different dimension table.

► Task 2: Create a Dimension

1. Add a dimension named **Reseller** based on the **DimReseller** table to the solution:
 - Do not include any related tables.
 - Include only the **Reseller Key**, **Business Type**, and **Reseller Name** attributes.
2. On the **Dimension Usage** tab of the Cube Designer for the **Sales** cube, verify that a regular relationship has been defined between the **Reseller Sales** measure group and the **Reseller** dimension.
3. Deploy the Adventure Works OLAP solution, and then browse the cube, verifying that you can view reseller revenue by attributes of the **Reseller** dimension.

Results: After this exercise, you should have added a **Reseller** dimension that uses a regular relationship with the **Reseller Sales** measure group to enable you to analyze reseller sales data.

Exercise 3: Configuring Measure Group Storage

Scenario

Users have requested that the cube is optimized to improve performance when analyzing Internet sales. However, you must balance this need for optimization against the space that is required to store aggregations.

The main tasks for this exercise are as follows:

1. Configure Proactive Caching
2. Design Aggregations

► Task 1: Configure Proactive Caching

1. Configure proactive caching for the **Internet Sales** measure group in the **Sales** cube. Specify that the **Automatic MOLAP** setting should be used.
2. Configure proactive caching for the **Reseller Sales** measure group in the **Sales** cube. Specify that the **Automatic MOLAP** setting should be used.

► Task 2: Design Aggregations

1. Use the Aggregation Design Wizard to design aggregations for the **Internet Sales** measure group in the **Sales** cube:
 - Set all aggregation usage to default before starting the configuration.
 - Use the wizard to count the objects in the measure group.
 - Generate aggregations until the performance gain reaches 30 percent.
 - Name the aggregation that you have generated as **InternetSalesAgg**.
2. Use the Aggregation Design Wizard to design aggregations for the **Reseller Sales** measure group, using the same settings as the **Internet Sales** measure group.
3. Deploy the **Adventure Works OLAP** solution.
4. Use SQL Server Management Studio to connect to the **localhost** instance of SQL Server Analysis Services, and then verify that the aggregation designs were deployed by using the **Sales** cube in the **Adventure Works OLAP** database.

Results: After this exercise, you should have configured proactive caching and defined the storage mode aggregations for the **Internet Sales** and **Reseller Sales** measure groups.

Module Review and Takeaways

In this module, you have learned how to create and configure measure groups in a cube, and define relationships between measure groups and dimensions.

Module 5

Introduction to MDX

Contents:

Module Overview	5-1
Lesson 1: MDX Fundamentals	5-2
Lesson 2: Adding Calculations to a Cube	5-10
Lesson 3: Using MDX to Query a Cube	5-17
Lab: Using MDX	5-23
Module Review and Takeaways	5-27

Module Overview

When you have a relational database that contains structured data in rows and columns, you can use Transact-SQL to write simple and effective queries to return the data that interests you. However, Transact-SQL is not designed to handle data that has more than two dimensions, such as that found in an online analytical processing (OLAP) cube. Instead, you can use the Multidimensional Expressions (MDX) language both to query multidimensional data and to return data in multiple dimensions. In this module, you will see how to use MDX to add features to a cube and display data in client applications such as SQL Server® Reporting Services or Microsoft® Excel®.

Objectives

After completing this module, you will be able to:

- Write a simple MDX query that establishes cube context and specifies both query and slicer axes.
- Use MDX to add calculated members, named sets, and scoped assignments to cubes.
- Use MDX from SQL Server Management Studio, Excel, SQL Server Reporting Services, and custom code to access and display data.

Lesson 1

MDX Fundamentals

A table in Microsoft SQL Server consists of two-dimensional data that is organized into columns and rows. An OLAP cube can have many more dimensions and multiple members of each dimension. The Transact-SQL language that is used to query tabular data is not ideal for more than two dimensions, so SQL Server Analysis Services includes the MDX query language. By using MDX, you can write queries that are easy to read and understand, but execute against multidimensional cube data. In this lesson, you will learn the essential features of the MDX language.

Lesson Objectives

After completing this lesson, you will be able to:

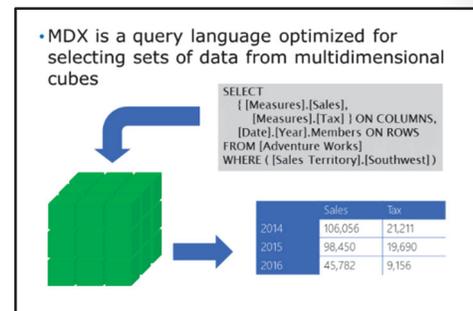
- Describe how MDX differs from Transact-SQL and how it enables developers to query multidimensional data more easily.
- Describe the structure of a simple MDX SELECT query.
- Use SELECT and WHERE to specify query and slicer axes in an MDX query.
- Use FROM, CREATE SUBCUBE, LOOKUPCUBE, and FILTER to establish the cube context for a query.
- Use SQL Server Management Studio to issue simple MDX queries.

What Is MDX?

Most databases store data in two-dimensional tables that consist of rows and columns. Transact-SQL is designed to formulate queries against this two-dimensional data and is highly efficient and readable. However, in an OLAP cube, data that has three or more dimensions is frequently stored. Very complex Transact-SQL would be required to query this data and it would be difficult for developers to read and understand such queries. Instead of using Transact-SQL, you can use MDX to query OLAP cubes. In MDX, it is simple to drill down into multiple dimensions and hierarchies, and select pertinent data from many dimensions.

In some ways, MDX is similar to Transact-SQL. For example, SELECT statements and WHERE clauses are common. However, these familiar elements have a different meaning and accept a different syntax in MDX.

An MDX query usually selects a subset of all of the data in the cube and this subset frequently has a smaller number of dimensions. For example, it is common to use MDX to return two-dimensional table-like data that can be displayed in an Excel spreadsheet. However, you can return any number of dimensions in your result set, up to the number of dimensions in the cube itself.



Note: There is also a hard limit of 128 query axes in an MDX SELECT statement.

An MDX query that returns a single value is called an MDX expression. A query that returns a large set of data, such as a table or a cube, is called an MDX statement. The following terms are also frequently used in MDX documentation:

- **Cell.** A cell is the space at the intersection between the specified member of the measures dimension and a specified member of one or more other dimensions. You can think of a cell as containing a single piece of data.
- **Tuple.** A tuple uniquely identifies a slice of data from a cube. It is defined as a combination of dimension members from different hierarchies. For example, a tuple might be all of the customers that are located in North America.
- **Set.** A set is a collection of tuples that defines multiple cells from the same dimensions. For example, a set might be all of the customers that are located in North America and all of the customers that are located in Africa.

MDX is used in many scenarios. For example, you can use it to select data for display in Excel and SQL Server Reporting Services. You can also use it in the definition of a cube, for example, to define a calculated member.

Basic MDX Query Syntax

The most common form of MDX query is a SELECT statement. This is used to return a subset of the data in the cube for display or for use in calculations. A SELECT statement frequently includes the following elements:

- **A SELECT clause.** The SELECT clause specifies what will appear along each axis of the result set. For example, it might say that specific years will appear on the columns and sales amounts will appear on the rows. The values that appear in each cell will be the intersections of those years and amounts.
- **A FROM clause.** The FROM clause specifies the cube to execute the query against.
- **A WHERE clause.** The WHERE clause is optional. You can use it to restrict the data that is returned. For example, it might specify that only data for the South West region is returned.

- Navigating hierarchies
- Using square brackets
- Avoiding ambiguity

```
SELECT
{
  [Date].[Calendar Year].[2015].[Q1],
  [Date].[Calendar Year].[2016].[Q1]
} ON COLUMNS
[Measures].[Sales Amount] ON ROWS
FROM [Adventure Works]
WHERE ( [Customers].[Tailspin Toys] )
```

Navigating Hierarchies

In a cube, the members of dimensions and measures are hierarchical. For example, in a time dimension, you might want to see data for 2016, for May 2016, or for May 1, 2016. There might also be parallel hierarchies. For example, you may want to see data for the 2015–2016 fiscal year instead of the calendar year. In MDX, you navigate these hierarchies by using dots. For example, to specify May 2016, you might use the term “[2016].[Q1].[May]”. The levels in the hierarchy depend on the structure of the cube that you are executing queries against.

Using Square Brackets

You need not always place square brackets around the name of an object in MDX. However, there are three cases where square brackets are required:

- When the object name contains a space or another special character.
- When the object name matches an MDX keyword, such as SELECT or FROM.

- When the object name begins with a numeric character, such as the name of a year.

Many MDX developers always use square brackets as a convention to avoid errors and ambiguity.

Avoiding Ambiguity

In a cube, many members might share the same name. For example, in a time dimension, **Q1** might refer to the first quarter of 2015 or the first quarter of 2016, or any other year in a single dimension. Similarly, the name **Tailspin Toys** may appear in both a **Suppliers** dimension and a **Customers** dimension. It is important to ensure that you specify objects or ranges of objects without any ambiguity to ensure that the data displayed is what you intended.

To eliminate these ambiguities, ensure that you qualify each object name to uniquely identify it. For example, to specify the first quarter of 2015, use **[2015].[Q1]**. To specify Tailspin Toys in the **Customers** dimension, use **[Customers].[Tailspin Toys]**, and so on. The precise qualification that you use will depend on the structure and hierarchy levels in your cube.

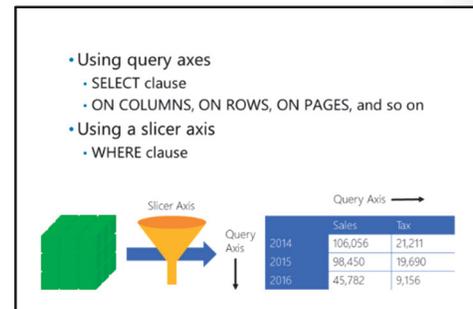
If an object is identified ambiguously, MDX does not return an error, but returns the first occurrence of the object in the database. For example, if there were members named **Q1** for every year from 2011, **[2011].[Q1]** would be returned because it is the first occurrence. Even if this is what you intended, it is best to uniquely identify a member for the benefit of other query writers.

Query and Slicer Axes

In an MDX query, query axes specify the members that appear in the edges of the result set. The slicer axis filters the result set. By using a combination of these axes, it is simple to drill down into very specific summaries of the data in your cube.

Using Query Axes

If you want a query that returns two-dimensional, tabular data, you must define what appears along the top, and down the side, of that table. These are the query axes. If your query is intended to return data in more than two dimensions, you must specify more than two query axes—one for every dimension of the result set.



Note: It is very common to specify two query axes because the tabular data that is produced can be easily read, for example, in a tabular report.

In the SELECT query, use the ON keyword to specify the axis on which to display a set of members.

In the following example, the query axes are specified by using the aliases COLUMNS, ROWS, and PAGES:

Using Axis Aliases

```
SELECT
  {[Date].[Calendar Year].[2015], [Date].[Calendar Year].[2016]} ON COLUMNS,
  {[Measures].[Sales Amount]} ON ROWS,
  {[Customers].[Tailspin Toys], [Customers].[Wingtip Toys]} ON PAGES
FROM [Adventure Works Cube]
```

In a SELECT query, you can specify up to 128 query axes. The first five of these axes have the aliases COLUMNS, ROWS, PAGES, SECTIONS, and CHAPTERS. Instead of using these aliases, you can use an ordinal number, starting from 0 for the columns.

In the following example, the query axes are specified by using the ordinal numbers:

Using Axis Ordinal Numbers

```
SELECT
    {[Date].[Calendar Year].[2015], [Date].[Calendar Year].[2016]} ON 0,
    {[Measures].[Sales Amount]} ON 1,
    {[Customers].[Tailspin Toys], [Customers].[Wingtip Toys]} ON 2
FROM [Adventure Works Cube]
```

 **Note:** You cannot skip axes in a SELECT query. For example, you cannot specify axes 0 and 2 without specifying axis 1. Similarly, you cannot use a PAGES axis without using both a COLUMNS axis and a ROWS axis.

In the above example, the 0 axis will include a 2015 column and a 2016 column. To return a column for every calendar year in the **Date** dimension, you could use the **Members** function. For example, use **[Date].[Calendar Year].Members**.

Using a Slicer Axis

A slicer axis filters the data that the SELECT expression returns. It is defined by the WHERE clause in an MDX SELECT query and it specifies a condition that must be met by all of the data that is summarized in the cells of the result set.

The following example is the same as the previous queries except that sales amounts will only be summed if they were made by the sales representative, Nicholas Rose:

Using a Slicer Axis

```
SELECT
    {[Date].[Calendar Year].[2015], [Date].[Calendar Year].[2016]} ON COLUMNS,
    {[Measures].[Sales Amount]} ON ROWS,
    {[Customers].[Tailspin Toys], [Customers].[Wingtip Toys]} ON PAGES
FROM [Adventure Works Cube]
WHERE [Sales Reps].[Name].[Nicholas Rose]
```

A slicer axis does not filter the members that appear along the edges of the result set. For example, it does not affect the rows that are returned. Instead, it filters the cube cells that will be summed (or otherwise aggregated) to generate the result cells.

Establishing Cube Context

Before you execute a query, you must define what the data query will run against. This definition is known as the cube context. Often, you establish the cube context simply by specifying the OLAP cube, but, in this topic, you will also see methods that you can use to narrow or broaden that context.

Using the FROM Clause

In MDX, you specify the cube to execute a query against by using the FROM clause.

In the following example, the cube context is the entire **Adventure Works** cube:

Using the FROM Clause

```
SELECT
  {[Date].[Calendar Year].[2015], [Date].[Calendar Year].[2016]} ON COLUMNS,
  {[Measures].[Sales Amount]} ON ROWS
FROM [Adventure Works]
```

- Using the FROM clause
- Creating a subcube
- Using the LOOKUPCUBE function
- Using the FILTER function

Creating a Subcube

Sometimes, you might be interested in executing multiple MDX queries against a small portion of the cube. For example, you might want to investigate several aspects of sales performance in North America during 2015. In such cases, you can, of course, specify query and slicer axes individually for each query, but it is more efficient to create a subcube of the data that interests you, and then execute queries against it.

Use the CREATE SUBCUBE statement to restrict a cube in this way. You pass to this statement an MDX expression that defines the content of the subcube.

In this example, a subcube is created that includes sales amounts for all calendar years and all customers. Other dimensions in the original cube are left out of the subcube.

Creating a Subcube

```
CREATE SUBCUBE [Adventure Works] AS
SELECT
  {[Date].[Calendar Year].Members} ON COLUMNS,
  {[Measures].[Sales Amount]} ON ROWS,
  {[Customers].Members} ON PAGES
FROM [Adventure Works]
```



Note: In the previous example, the subcube shares the same name as the original cube (“Adventure Works”). All subsequent MDX queries that have “Adventure Works” in the FROM clause will execute against the subcube until the session closes, or until you issue a DROP SUBCUBE statement.

Note that some members of a cube might be present in the subcube even though you did not explicitly specify them in the CREATE SUBCUBE statement. For example, the subcube will include all of the descendants and ascendants of the members that you specified.

Also, be careful to understand aggregation in subcubes. For example, consider a subcube that includes the US states, Washington and Oregon, plus the country, the United States. The reported sales for the United States member will be the sum of sales in Washington and Oregon, but sales from other states will not be added, because those states are not included in the subcube.

Using the LOOKUPCUBE Function

In addition to narrowing the cube context by using subcubes, you can broaden the cube context by using the LOOKUPCUBE function. This function enables you to include members from cubes other than the one that you specify in the FROM clause. All cubes in your query must be in the same database.

In the following example, the "Sales Amount" measure from the "Adworks Archive" cube is added as a new measure member named "Archive Sales Amount." This is displayed on its own row.

Using the LOOKUPCUBE Function

```
WITH MEMBER [Measures].[Archive Sales Amount] AS
LOOKUPCUBE("Adworks Archive", "[Measures].[Sales Amount]")
SELECT
    { [Date].[Calendar Year].Members } ON COLUMNS,
    { [Measures].[Sales Amount], [Measures].[Archive Sales Amount] } ON ROWS,
FROM [Adventure Works]
```



Best Practice: Although they are useful, queries that use the LOOKUPCUBE function to integrate two cubes are likely to perform less well than queries against a single cube. If you find that you use LOOKUPCUBE often, consider redesigning your cubes to include all of the relevant data.

Using the FILTER Function

The FILTER function returns a set of members where a specific condition is true. For example, if you had a dimension member that consisted of all of the years since 1980, you could use a FILTER function to return only the years after 2010. You can use FILTER functions in several places in an MDX query. To help to establish cube context, you can use FILTER functions to select members to place on query axes.

In the following example, the FILTER function removes all calendar years before 2010 from the result set:

Use FILTER in Query Axes

```
SELECT
    { FILTER( [Date].[Calendar Year].Members, [Date].[Calendar Year]>2009 ) } ON COLUMNS,
    { [Measures].[Sales Amount] } ON ROWS,
FROM [Adventure Works]
```

Demonstration: Querying a Cube

In this demonstration, you will see how to write and execute MDX queries in SQL Server Management Studio.

Demonstration Steps

Deploy and Query a Cube

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. In the D:\Demofiles\Mod05\Starter folder, run **Setup.cmd** as Administrator.
2. On the Start screen, type **SQL Server Data Tools 2015**, and then press Enter.
3. In SQL Server Data Tools, on the **File** menu, point to **Open**, and then click **Project/Solution**.
4. In the **Open Project** dialog box, browse to the **D:\Demofiles\Mod05\Starter** folder, and then double-click **Adventure Works OLAP.sln**.
5. On the **Build** menu, click **Deploy Solution**. If you are prompted, specify the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **OK**.
6. Wait for the **Deployment Completed Successfully** message in the Deployment Progress – Adventure Works OLAP window, and then close the Deployment Progress – Adventure Works OLAP window.
7. Close Microsoft Visual Studio®.
8. Start SQL Server Management Studio.
9. In the **Connect to Server** dialog box, in the **Server type** drop-down list, select **Analysis Services**.
10. In the **Server name** text box, select **MIA-SQL**, and then click **Connect**.
11. In **Object Explorer**, expand **Databases**.
12. Right-click **Adventure Works OLAP**, point to **New Query**, and then click **MDX**.
13. In the query pane, type the following code:

```
SELECT
{
[Measures].[Internet Sales Count],
[Measures].[Reseller Sales Count]
} ON COLUMNS,
{
[Order Date].[Calendar Date].[Calendar Year]
} ON ROWS
FROM [Sales]
```

14. Click **Execute**, and then examine the results.
15. To specify a slicer axis, add the following line of code at the previous query:

```
WHERE [Customer].[State Province Name].&[Ohio]
```

16. Click **Execute**, and then examine the results.
17. Close SQL Server Management Studio, without saving any changes.

Check Your Knowledge

Question	
You want to run 10 queries against the Adventure Works cube, but you are only interested in summarizing results from 2015 and 2016. Which of the following techniques should you use to execute queries most efficiently?	
Select the correct answer.	
<input type="checkbox"/>	Specify the 2015 and 2016 members in each SELECT query.
<input type="checkbox"/>	Use the FILTER function in each query to display only the 2015 and 2016 members.
<input type="checkbox"/>	Use the LOOKUPCUBE function.
<input type="checkbox"/>	Create a subcube that includes only the 2015 and 2016 members.
<input type="checkbox"/>	Specify a slicer axis by using the WHERE clause.

Lesson 2

Adding Calculations to a Cube

In previous modules, you have seen how to create dimensions and measures and use them to define the data in a cube. However, not all of the members of a cube must come from the underlying data source, such as a data warehouse. In this lesson, you will see how to create additional features in a cube, including calculated members, named sets, and other script commands. All of these features are added to the cube by using the **Calculations** tab of the Cube Designer. You will also see some MDX functions that can be very helpful when defining script commands.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe calculated members and explain how they can be used to enhance the information in a cube.
- Define named sets and explain how they can make it easier to write MDX queries.
- Describe how scoped assignments limit the cells to which a script action applies.
- Use MDX functions to locate the family members for an object, such as a parent or the first child.
- List other commonly used MDX functions and choose a function for a given calculation.
- Add calculated members to a cube by using SQL Server Data Tools.

Calculated Members

A calculated member is a cube member whose value is calculated when a user queries the cube (in other words, at run time) by using an MDX expression that you define as part of the cube. Desktop applications and MDX clients can access the calculated member just as they access other members of dimensions or measures. A calculated member can be defined as a combination of cube data, arithmetic operations, numbers, and functions.

For example, if your cube includes the measures for [Sale Price] and [Production Cost], you could define a member named [Profit] as follows:

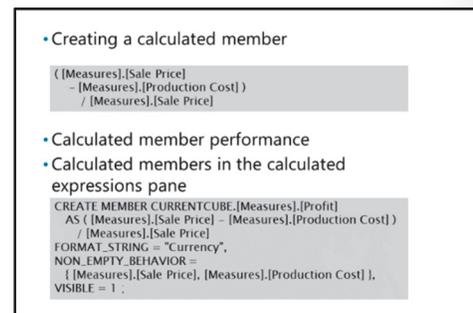
An Example of a Calculated Member Expression

```
( [Measures].[Sale Price] - [Measures].[Production Cost] ) / [Measures].[Sale Price]
```

Creating a Calculated Member

Use the Cube Designer to add a calculated member to a cube. On the **Calculations** tab, in the Calculation Expressions pane, the default CALCULATE command specifies that the measures in the cube should be aggregated according to their defined aggregate function. To define a new calculated member, leave the default script in place and click the **New Calculated Member** button. In addition to the MDX expression for the new calculated member, you must fix the following properties:

- **Name.** Specify a descriptive and unique name to identify the new member. Ensure that the name is meaningful to users who will be browsing the cube.



- **Parent hierarchy.** Specify the dimension that the member appears within. If you place the calculated member in the **Measures** hierarchy, it is known as a calculated measure.
- **Format string.** You can select how the value is displayed. For example, you could select **Currency** or **Percent**.
- **Visibility.** If you want users to be able to see the calculated member as they browse the cube, select **True**.
- **Non-empty behavior.** This is an important property for performance. See the next section for details.

Calculated Member Performance

The values of calculated members are not stored with the cube, so they do not increase the disk space that a cube consumes. However, because calculated members are executed at run time, they can increase demands on the processing resources of your database server. If you use many complex calculated members, they might have a negative impact on query performance.

One way to help to reduce the processing demands of a calculated member is to carefully specify a **Non-empty behavior** value. If this value is blank, SQL Server Analysis Services must evaluate the full expression to determine whether the calculated member is empty. If, by contrast, you specify one or more members in the **Non-empty behavior** property, SQL Server Analysis Services trusts that when all of these members are empty, the calculated member is empty. This means that SQL Server Analysis Services can skip this calculation. In the expression example above, you could add the [Sale Price] and [Production Cost] to the **Non-empty behavior** property, because when they are empty, [Profit] is empty.

Calculated Members in the Calculation Expressions Pane

When you complete the **New Calculated Member** form, a new CREATE MEMBER MDX command appears in the Calculation Expressions pane.

The following example shows the code that the **Calculated Expressions** page displays when you add a calculated member:

Calculated Member Example

```
CREATE MEMBER CURRENTCUBE.[Measures].[Profit]
  AS ( [Measures].[Sale Price] - [Measures].[Production Cost] ) / [Measures].[Sale
Price]
  FORMAT_STRING = "Currency",
  NON_EMPTY_BEHAVIOR = { [Measures].[Sale Price], [Measures].[Production Cost] },
  VISIBLE = 1 ;
```

Named Sets

A named set is an MDX expression that returns a set of dimension members. Named sets can be defined and saved as part of the cube. Named sets enable other MDX expressions—for example, expressions that are issued by clients or custom applications—to easily access a common set of data.

Named sets are used for two purposes:

- To enable clients to easily access a commonly used set of members.
- To enable clients to easily access a set of members with a complex definition.

By defining a named set, you reduce the amount and complexity of MDX code that must be created in client applications. As for calculated members, you can use the **Calculations** tab of the Cube Designer to create named sets.

The following example creates a set named **Core Products** that includes only products from the category **Bikes**.

Creating a Named Set

```
CREATE SET CURRENTCUBE.[Core Products]
AS [Product Category].[English Product Category Name].&[Bikes] ;
```

- Named sets are MDX expressions that return sets of members
 - For commonly used sets
 - To hide complex set definitions from clients

```
CREATE SET CURRENTCUBE.[Core Products]
AS [Product Category]
.[English Product Category Name]
.&[Bikes] ;
```

Scoped Assignments

When you use the **New calculated member** button or the **New named set** button on the **Calculations** tab of the Cube Designer, you add script to the definition of the cube that creates that member or set, according to the options that you select. You can also use the **New script command** button to add any other kind of MDX script to the definition of the cube.

A common script to add to the cube definition is a scoped assignment. This is a way to limit an action to apply to a subcube, instead of the default scope, which is the entire cube. Use the SCOPE statement to begin your scoped assignment and define the subcube. All subsequent operations in your script will apply to that subcube only until the END SCOPE statement, or until another SCOPE statement.

You can nest SCOPE statements. For example, you could scope the script to provide only results from 2014, take some actions, and then scope the script further to provide only results from January 2014, and then take some more specific actions.

Within the scoped assignment, use the THIS function to refer to all members of the current subcube.

- Default scope for calculations is the entire cube
- You can restrict calculations to a subcube by using a scoped assignment
- Use the THIS function to refer to the members of the current scope

```
SCOPE
([Date],[Fiscal Year].&[2016],
 [Date],[Fiscal],[Fiscal Quarter],Members,
 [Measures].[Sales Amount Quota]) ;
THIS = PARALLELPERIOD
([Date],[Fiscal],[Fiscal Year], 1,
 [Date],[Fiscal],CurrentMember) * 1.1 ;
END SCOPE
```

In the following example, a SCOPE statement limits the current scope to the fiscal quarters in 2016 and the **Sales Amount Quota** measure. The THIS statement is used to apply a calculation to the members of that scope by multiplying the quotas from the previous year, which are obtained by using the PARALLELPERIOD function.

Using the SCOPE Statement

```
SCOPE
(
  [Date].[Fiscal Year].&[2016],
  [Date].[Fiscal].[Fiscal Quarter].Members,
  [Measures].[Sales Amount Quota]
);

THIS = PARALLELPERIOD
(
  [Date].[Fiscal].[Fiscal Year], 1,
  [Date].[Fiscal].CurrentMember
) * 1.1 ;

END SCOPE
```

 **Note:** When you are troubleshooting scoped assignments, it is helpful to see which cells are in the current scope. You can use the BACK_COLOR cell property to highlight the cells in the current scope.

MDX Functions for Family Members

As you have seen, in an OLAP cube, the members of dimensions and measures are organized into hierarchies. When you write MDX expressions and scripts, it is often helpful to access members that are related to the current member. For example, to assign a sales target to a quarter, you might obtain the sales target for the year and then divide it by four. The year is the parent of the quarter.

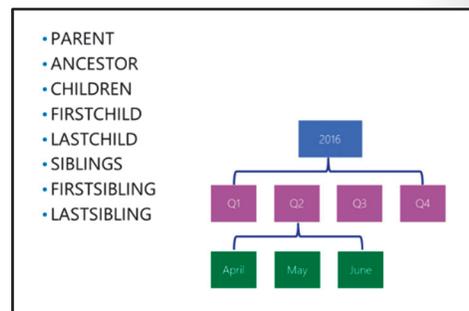
The following functions are used to return various family members according to their relationship to the current member:

- **PARENT.** This function returns the member one level above the current member. Each member has only one parent. For example, the following expression returns the parent year for the current quarter.

```
[Date].[Year].[Quarters].CurrentMember.Parent.Name
```

- **ANCESTOR.** This function returns the member a specified number of levels above the current member. When you specify one level above, ANCESTOR returns the same member as PARENT. However, you can use ANCESTOR to obtain members multiple levels above the current member in a single operation. ANCESTOR returns a single member. For example, the following expression returns the year for the current month.

```
Ancestor([Date].[Year].[Quarters].[Month].CurrentMember, 2).Name
```



- **CHILDREN.** This function returns all of the members, one level below the current member, that have the current member as their parent.
- **FIRSTCHILD.** This function returns a single member, one level below the current member, that appears first in the database.
- **LASTCHILD.** This function returns a single member, one level below the current member, that appears last in the database.
- **SIBLINGS.** This function returns all of the members that have the same parent as the current member. The members returned include the current member.
- **FIRSTSIBLING.** This function returns the sibling that appears first in the database.
- **LASTSIBLING.** This function returns the sibling that appears last in the database.

Other Common MDX Functions

Other MDX functions that are often useful in cube calculations include the following:

- **CURRENTMEMBER.** This function returns the member currently being operated on when your code iterates through a set of hierarchy members.
- **NAME.** This function returns the name of a cube object. The object might be a dimension, a hierarchy, a level, or a member.
- **DIMENSION.** This function returns the dimension that contains an object. The object may be a hierarchy, a level, or a member.
- **HIERARCHY.** This function returns the hierarchy that contains an object. The object might be a level or a member.
- **LEVEL.** This function returns the level that contains a member.
- **PARALLELPERIOD.** This time-based function returns a member from the previous period with the same relative expression. For example, you could use this function to find a month one year (or four quarters) earlier than another member, as in the following example:

```
PARALLELPERIOD([Date].[Calendar].[Calendar Quarter], 4,
[Date].[Calendar].[Month].[October 2016])
```

- **NONEMPTY.** This function returns only the members of a set that are not empty. This can help to improve performance by ensuring that a complex calculation is not performed when a key member is empty. For example, the following code returns a set of all of the customers whose **Sales Amount** value was not empty on January 1, 2016:

```
NONEMPTY([Customer].Members,
{ [Date].[Calendar].[Date].&[20160101], [Measures].[Sales Amount] })
```

- CURRENTMEMBER
- NAME
- DIMENSION
- HIERARCHY
- LEVEL
- PARALLELPERIOD
- NONEMPTY

Demonstration: Adding a Calculated Member to a Cube

In this demonstration, you will see how to add a simple calculated member and then use it in an MDX query.

Demonstration Steps

Add a Calculated Member

1. Ensure that you have completed the previous demonstration in this module.
2. On the Start screen, type **SQL Server Data Tools 2015**, and then press Enter.
3. In SQL Server Data Tools, on the **File** menu, point to **Open**, and then click **Project/Solution**.
4. In the **Open Project** dialog box, browse to the **D:\Demofiles\Mod05\Starter** folder, and then double-click **Adventure Works OLAP.sln**.
5. In Solution Explorer, under **Cubes**, double-click **Sales.cube**.
6. In the Cube Designer, click the **Calculations** tab.
7. Right-click in the Script Organizer pane, and then click **New Calculated Member**.
8. In the **Name** text box, type **[Total Sales Count]**.
9. In the **Parent Hierarchy** list, ensure that **Measures** is selected.
10. In the **Expression** text box, type the following MDX expression:

```
[Measures].[Internet Sales Count] + [Measures].[Reseller Sales Count]
```

11. In the **Format string** drop-down list, select **"0"**.
12. In the **Non-empty behavior** list, select **Internet Sales Count** and **Reseller Sales Count**, and then click **OK**.
13. On the **File** menu, click **Save All**.
14. On the **Build** menu, click **Deploy Solution**. If you are prompted to authenticate, use the password **Pa\$\$w0rd**, and then click **OK**.
15. When the deployment is complete, close Visual Studio.

Query a Calculated Member

1. Open SQL Server Management Studio.
2. In the **Connect to Server** dialog box, in the **Server type** drop-down list, click **Analysis Services**.
3. In the **Server name** drop-down list, click **MIA-SQL**, and then click **Connect**.
4. In Object Explorer, expand **Databases**.
5. Right-click **Adventure Works OLAP**, click **New Query**, and then click **MDX**.
6. Type the following code in the query pane:

```
SELECT
{
[Measures].[Internet Sales Count],
[Measures].[Reseller Sales Count],
[Measures].[Total Sales Count]
} ON COLUMNS,
[Product].[Categorized Products].[Category] ON ROWS
FROM [Sales]
```

7. Execute the query, and then examine the results.
8. Close SQL Server Management Studio, without saving any changes.

Categorize Activity

Categorize each item into the appropriate category. Indicate your answer by writing the category number to the right of each item.

Items	
1	PARENT
2	PARALLELPERIOD
3	CHILDREN
4	NAME
5	SIBLINGS
6	CURRENTMEMBER
7	FIRSTSIBLING
8	HIERARCHY
9	LASTCHILD
10	LEVEL
11	ANCESTOR
12	NONEMPTY

Category 1	Category 2
Family MDX functions	Non-family MDX functions

Lesson 3

Using MDX to Query a Cube

Multidimensional data, such as that in an OLAP cube, is often accessed and displayed from Excel in PivotTables and PivotCharts. It might also be a data source in SQL Server Reporting Services. In this lesson, you will see how to connect to a cube in SQL Server Analysis Services from these and other tools. After you have connected to SQL Server Analysis Services, you can issue MDX queries by writing them manually, or by using user-friendly tools.

Lesson Objectives

After completing this lesson, you will be able to:

- Create connection strings to connect to a specific cube in SQL Server Analysis Services.
- Create MDX queries by using the tools in Excel and SQL Server Reporting Services.
- Advise developers about object models to use for access to SQL Server Analysis Services.
- Query a SQL Server Analysis Services cube from Excel.

Using and Testing MDX Queries

MDX is a query language that enables users to obtain interesting data quickly from an OLAP cube. Like Transact-SQL, a typical user has little or no expertise in MDX, but uses a client application that has an easily understood user interface to formulate queries and view results. Typical scenarios in which MDX is used include the following:

- **From Excel.** Excel includes an MDX query designer that can be used to populate PivotTables, PivotCharts, and other displays.
- **From SQL Server Reporting Services.** You can use Report Builder or Report Designer to build reports that are based on MDX queries, just as you can for Transact-SQL queries.
- **From PerformancePoint Dashboards.** In Microsoft SharePoint® sites, you can use PerformancePoint dashboards to display business performance metrics and compare them against targets. OLAP cubes can be used as sources for these metrics.
- **From a custom application.** A developer can write code that uses programmatic interfaces such as XMLA or ADOMD.NET to send MDX queries to SQL Server Analysis Services. Such code can include desktop applications, mobile device apps, and many other kinds of custom solution.

Excel, SQL Server Reporting Services, and custom applications will be examined in more detail later in this module.

- Common MDX scenarios
- Connection and authentication
- Using SQL Server Management Studio to test MDX queries

Connection and Authentication

Regardless of the client that you use to connect to SQL Server Analysis Services, you must always connect and authenticate. To connect, you must provide several values to populate a connection string. For SQL Server Analysis Services, the connection string often includes:

- **A data source.** This specifies the instance of SQL Server Analysis Services. You can use an instance name, a fully qualified domain name, an IP address, or a port number to identify the instance.
- **An initial catalog.** This specifies the SQL Server Analysis Services database to connect to.
- **A provider.** This optional value specifies the version of SQL Server Analysis Services. In SQL Server 2016 and 2012, use "MSOLAP.5" for the provider.
- **A cube.** This specifies the cube to connect to.
- **An application name.** By using an application name in your connection string, you identify the client application. This application name appears in trace logs and can help to identify a problem application in troubleshooting.
- **A timeout.** This specifies how long the client waits for a response to a query before it raises an error.
- **Character encoding.** This specifies whether strings use UTF-8 or UTF-16 to encode characters in MDX queries.
- **An MDX compatibility value.** If this value is set to 0 or 1, missing members in ragged hierarchies are displayed by placeholders. For example, Excel sets this property to 1 and displays an empty cell for missing members. If this value is set to 2, placeholders are not returned.

SQL Server Analysis Services does not support SQL Server Authentication, so you must use Windows® integrated authentication. Depending on the client, this may be Kerberos or NTLM authentication. You can use the following connection string properties to control authentication:

- **EffectiveUserName.** Use this setting to impersonate a different user for the duration of the connection. Specify the user account in domain\username format. Note that the connection is made as the currently logged-on user, who must have administrative permission for SQL Server Analysis Services.
- **Use Encryption for Data.** When this value is true, all queries and results are encrypted on the network between the client and the server.
- **User ID.** Use this account to connect to SQL Server Analysis Services. If this value is not specified, the client uses the currently logged-on Windows user account. This setting must be used with **Password**.
- **Password.** Use this to authenticate with the **User ID** setting.

Using SQL Server Management Studio to Test MDX Queries

SQL Server Management Studio is not designed as an end-user tool, but you can use it to connect to SQL Server Analysis Services and issue MDX queries. Often, SQL Server Management Studio is used to develop and test queries that are used in reports or PivotTables when you are confident that they execute and perform as expected.

To connect to SQL Server Analysis Services in SQL Server Management Studio:

1. Start SQL Server Management Studio.
2. In the **Connect to Server** dialog box, in the **Server type** drop-down list, select **Analysis Services**.
3. On the **Login** tab, in the **Server name** text box, type the name or fully qualified domain name of the SQL Server Analysis Services server. To connect to an instance other than the default instance, use the Server\Instance format.
4. Click **Connect**.

 **Note:** You can only use Windows authentication to connect to SQL Server Analysis Services, not SQL Server Authentication. Your current Windows account must be authorized to access the database and cube.

Using MDX from Excel and SQL Server Reporting Services

PivotTables and PivotCharts in Excel are ideal tools to analyze and display multidimensional data. Excel can connect to SQL Server Analysis Services and includes an MDX Query Designer that you can use to determine what OLAP cube data to import into Excel.

Connecting to SQL Server Analysis Services from Excel

Take the following steps to connect to SQL Server Analysis Services from Excel:

1. Open an existing or new Excel workbook.
2. On the ribbon, click the **Data** tab.
3. Under **Get External Data**, click **From Other Sources**, and then click **From Analysis Services**.
4. In the Data Connection Wizard, in the **Server name** text box, type the name of the Analysis Services server to connect to, and then click **Next**.
5. From the drop-down list, select the SQL Server Analysis Services database.
6. Select the cube from the list, and then click **Finish**.

 **Note:** The precise steps for this procedure might vary depending on your version of Excel.

For more information about connecting Excel to SQL Server Analysis Services, see the following article:

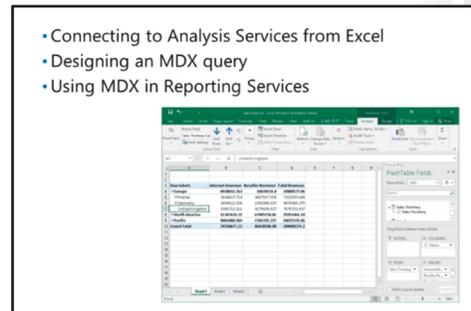
 **Connect to or import data from SQL Server Analysis Services**

<http://aka.ms/Bnxvey>

Designing an MDX Query

When you have a connection to SQL Server Analysis Services, you can use the MDX Query Designer to import data from the cube into Excel. The MDX Query Designer can be used in two modes:

- **Design mode.** This mode is for those who prefer to construct MDX queries by graphical means. The user interface includes the following elements:
 - **Select Cube button.** Use this button to select the current cube.
 - **Metadata pane.** This pane displays dimensions, measures, and key performance indicators (KPIs) in the current cube. You can drag members from this pane onto the data pane to formulate a query.
 - **Calculated Members pane.** This pane displays calculated members in the current cube. As for the Metadata pane, you can drag members from this pane onto the data pane.



- **Filter pane.** Use this pane to create filters from dimensions and measures to limit the data that is returned.
- **Data pane.** This is where the results for the current query are displayed.
- **Query mode.** This mode is for those who prefer to construct queries by writing MDX text. The user interface includes the following elements:
 - **Select Cube button.** This is the same button that appears in Design mode.
 - **Metadata pane.** As for Design mode, this pane displays dimensions, measures, and KPIs. In Query mode, however, it also displays MDX functions and template queries, which you can drag onto the query pane.
 - **Query pane.** This pane displays the current query as MDX query text. You can formulate queries by typing them manually into this pane, by dragging elements from the Metadata pane, or by a combination of both operations.
 - **Data pane.** This pane displays the results for the current query as it is displayed in the query pane.

For more information on how to use the MDX Query Designer in Excel, see the following article:



Analysis Services MDX Query Designer (Power Pivot)

<http://aka.ms/Qaexrf>

Using MDX in SQL Server Reporting Services

In SQL Server Reporting Services, you can use an OLAP cube that is hosted in SQL Server Analysis Services as a source of data to display in a report. To do this, you must create a data source in SQL Server Reporting Services of the type **Microsoft SQL Server Analysis Services**.

In SQL Server Reporting Services, data sources can either be embedded, which means that they can only be used in a single report, or shared, which means that they can be used in many different reports. The following steps explain how to create a shared data source in SQL Server Analysis Services by using Report Manager:

1. In Report Manager, on the **Contents** page, click **New Data Source**.
2. Type a descriptive name for the data source.
3. In the **Data source type** drop-down list, select **Microsoft SQL Server Analysis Services**.
4. In the **Connection string** text box, type a connection string for the instance of SQL Server Analysis Services that you want to query. See the previous topic for details of the properties that you should use.
5. Under **Connect using**, specify how SQL Server Reporting Services will authenticate with SQL Server Analysis Services. Remember that you can only use Windows authentication:
 - If you select **Windows Integrated Security**, the current Windows account will be used to authenticate.
 - If you select **Credentials supplied by the user running the report**, the user will be prompted for the credentials of a Windows account.
 - If you select **Credentials stored securely in the report server**, the same user account will always be used to authenticate with SQL Server Analysis Services, regardless of the user who accesses the report.

- If you select **Credentials are not required**, SQL Server Reporting Services will authenticate anonymously with SQL Server Analysis Services. You must ensure that anonymous access is enabled in SQL Server Analysis Services and also that the unattended execution account is configured on the report server.
6. Use the **Test Connection** button to validate your settings.
 7. Click **OK** to create the shared data source.

When you have created a data source, you can use the MDX Query Designer to formulate queries. This tool is very similar to the Excel MDX Query Designer.

For more information about creating MDX queries against SQL Server Analysis Services data in SQL Server Reporting Services, see the following article:

 **Analysis Services Connection Type for MDX (SSRS)**

<http://aka.ms/K3v2bm>

Using MDX from Custom Code

SQL Server Reporting Services and Excel are versatile tools that you can use to display data from OLAP cubes in many scenarios. However, sometimes you might want to issue MDX queries and display results in your own custom application. Developers have a choice of the following programmatic interfaces, all of which can be called from managed code:

- **ADOMD.NET**. This is the multidimensional version of the popular ADO.NET object model. It makes it easy for you to issue MDX queries and process multidimensional results in managed code.
- **SQL Server Analysis Services OLE DB provider**. If you prefer to use the OLE DB standard to access data, you can use this provider to access a cube.
- **XMLA**. This is an industry-standard protocol based on SOAP and XML—a lower-level protocol that might require more custom code than ADOMD.NET. However, it is supported by any programming platform that has support for SOAP and XML.
- **AMO**. Analysis Management Objects (AMO) is a library of managed objects that you can use to configure SQL Server Analysis Services. You cannot use this library to query a cube—instead, you can use it to create cubes, dimensions, measures, and other objects from custom code.

- ADOMD.NET
- SQL Server Analysis Services OLE DB provider
- XMLA
- AMO

For more information about programmatic interfaces for multidimensional data, see the following article:

 **Programmatic Interfaces**

<http://aka.ms/Cr0cp5>

Demonstration: Querying a Cube from Excel

In this demonstration, you will see how to use Excel to import data from a SQL Server Analysis Services cube into a PivotTable, by formulating an MDX query.

Demonstration Steps

Connect to SQL Server Analysis Services

1. On the taskbar, click **Excel 2016**.
2. On the Welcome screen, click **Blank workbook**.
3. On the ribbon, click the **Data** tab.
4. Click **Get External Data**, click **From Other Sources**, and then click **From Analysis Services**.
5. In the **Server name** text box, type **MIA-SQL**, and then click **Next**.
6. Ensure that the **Adventure Works OLAP** database and the **Sales** cube are selected, and then click **Next**.
7. Click **Finish**.
8. In the **Import Data** dialog box, ensure that **PivotTable Report** is selected, and then click **OK**.

Formulate an MDX Query in Excel

1. In the **PivotTable Fields** box, under **Internet Sales**, select **Internet Sales Count**.
2. Under **Reseller Sales**, select **Reseller Sales Count**.
3. Under **Values**, select **Total Sales Count**.
4. Under **Order Date**, select **Order Date.Calendar Date**.
5. In the **COLUMNS** box, drag **Order Date.Calendar Date** to the **ROWS** box.
6. Examine the data in the PivotTable.
7. Close all open windows, without saving any changes.

Check Your Knowledge

Question	
<p>You are creating a connection string for a data source in SQL Server Reporting Services. You want to ensure that ragged hierarchies include placeholders for missing members. Which of the following properties should you use for this setting?</p>	
<p>Select the correct answer.</p>	
<input type="checkbox"/>	Data source
<input type="checkbox"/>	Initial catalog
<input type="checkbox"/>	Provider
<input type="checkbox"/>	Character encoding
<input type="checkbox"/>	MDX compatibility

Lab: Using MDX

Scenario

Currently, the AdventureWorks SQL Server Analysis Services cubes contain large amounts of vital information. However, some of that data needs further calculation to produce the exact information required. You plan to use MDX calculations to produce this information.

You want to add several important measures to a cube, but these measures must be calculated from other, existing measures. These include:

- The total revenue from both reseller sales and Internet sales.
- The percentage profit from reseller sales.
- The percentage profit from Internet sales.

You intend to add these measures to the cube by calculating them from existing measures. You also want to test the cube by using MDX queries in SQL Server Management Studio. Finally, you want to create an Excel worksheet that draws data from the **Sales** cube into a worksheet and into a PivotTable.

Objectives

After completing this lab, you will be able to:

- Use form view and script view in the Cube Designer to add calculated members.
- Write and execute MDX queries in SQL Server Management Studio and Excel.

Estimated Time: 45 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Adding Calculated Members

Scenario

You are refining an OLAP cube that summarizes sales data from your data warehouse. You have been asked to add important calculated members to the measures in the cube. You will add a **Total Revenue** member and two profit members—one for Internet sales and one for reseller sales.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Add a Calculated Member by Using Form View
3. Add Calculated Members by Using Script View

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab05\Starter folder as Administrator.

► Task 2: Add a Calculated Member by Using Form View

1. Start SQL Server Data Tools, and then open the **Adventure Works OLAP.sln** solution in the D:\Labfiles\Lab05\Starter folder.

2. In the **Sales.cube** cube, use the **Calculations** tab of the Cube Designer to add a new calculated member. Use the following information:
 - Name: [Total Revenue]
 - Parent Hierarchy: Measures
 - Expression: Add [Internet Revenue] to [Reseller Revenue]
 - Format String: "Currency"
 - Non-Empty Behavior: [Internet Revenue] and [Reseller Revenue]

► **Task 3: Add Calculated Members by Using Script View**

1. On the **Calculations** tab of the Cube Designer, in script view, type MDX code to create a new calculated member. Use the following information:
 - Name: [Internet Profit]
 - Expression: Take [Internet Cost] away from [Internet Revenue] and then divide the result by [Internet Cost]
 - Format String: "Percent"
 - Non-Empty Behavior: [Internet Revenue] and [Internet Cost]
 - Visible: 1
 - Associated Measure Group: 'Internet Sales'
2. Type MDX code to create a second new calculated member. Use the following information:
 - Name: [Reseller Profit]
 - Expression: Take [Reseller Cost] away from [Reseller Revenue] and then divide the result by [Reseller Cost]
 - Format String: "Percent"
 - Non-Empty Behavior: [Reseller Revenue] and [Reseller Cost]
 - Visible: 1
 - Associated Measure Group: 'Reseller Sales'
3. Save your changes, deploy the solution, and then close Visual Studio.

Results: After this exercise, you should have added three calculated members to the **Sales** cube and deployed the cube to the MIA-SQL instance of SQL Server Analysis Services.

Exercise 2: Querying a Cube by Using MDX

Scenario

You want to test the new calculated members that you added to the cube by using SQL Server Management Studio to issue MDX queries. You also want to create an Excel workbook that imports data from the **Sales** cube into both a worksheet and a PivotTable.

The main tasks for this exercise are as follows:

1. Test Total Revenue in SQL Server Management Studio
2. Test Profit Calculations in SQL Server Management Studio
3. Use Excel to Execute MDX Queries
4. Use an MDX Query in a PivotTable

► Task 1: Test Total Revenue in SQL Server Management Studio

1. Start SQL Server Management Studio, and then connect to the MIA-SQL instance of SQL Server Analysis Services.
2. Write an MDX query against the [Sales] cube. Return [Internet Revenue], [Reseller Revenue], and [Total Revenue] on columns, and all of the members of [Product].[Categorized Products] on rows. Examine the results to ensure that the total is correct for each member.

► Task 2: Test Profit Calculations in SQL Server Management Studio

1. In SQL Server Management Studio, create and execute a new MDX query against the **Adventure Works OLAP** cube. Use the following information:
 - Columns: [Internet Cost], [Internet Revenue], and [Internet Profit]
 - Rows: All of the members of [Customer].[City]
2. Add a slicer axis to the MDX query so that it displays only results from New York State. Execute your modified query, and then examine the results.
3. Create and execute a second new query. Use the following information:
 - Columns: [Reseller Cost], [Reseller Revenue], and [Reseller Profit]
 - Rows: All of the members of [Sales Territory].[Sales Territory]
4. Close SQL Server Management Studio, without saving any changes.

► Task 3: Use Excel to Execute MDX Queries

- In Excel 2016, add a new SQL Server Analysis Services query that loads data from the **Sales** cube in the **Adventure Works OLAP** database. Display the following columns:
 - Internet Cost
 - Internet Revenue
 - Internet Profit
 - Product Category
 - Product Subcategory

► Task 4: Use an MDX Query in a PivotTable

1. In Excel, use the Get External Data tool to connect to SQL Server Analysis Services to import sales data into a new PivotTable. Use the following information:
 - Server name: MIA-SQL
 - Database: Adventure Works OLAP
 - Cube: Sales
 - Import the data into a new worksheet
2. In the new PivotTable, choose the **Internet Revenue**, **Reseller Revenue**, and **Total Revenue** measures for columns, and the **Sales Territory** dimension for rows.
3. In the worksheet, determine the **Total Revenue** value for the United Kingdom.
4. Close Excel, without saving any changes.

Results: After this exercise, you should have created several test MDX queries in SQL Server Management Studio and an Excel spreadsheet that connects to the **Sales** OLAP cube.

Module Review and Takeaways

MDX is a sophisticated query language that is much like Transact-SQL, but designed specifically for executing queries against, and returning, multidimensional data such as that in an OLAP cube. In this module, you have seen how to specify query axes by using the SELECT statement, how to specify a slicer axis by using the WHERE clause, and how to establish cube context by using the FROM clause. You have learnt how advanced features such as calculated members and named sets can be added to a cube by using MDX queries in the Cube Designer. You have also seen how to display cube data in client applications such as SQL Server Reporting Services and Excel.

Review Question(s)

Question: Users often want to aggregate data from Europe and North America in their reports and spreadsheets. Users who have good MDX skills have been able to do this by editing MDX queries in their client programs. How can you help users who have no MDX skills to display this data?

Question: You are testing the **Sales** cube by writing MDX queries in SQL Server Management Studio. You would like to compare sales for 2015 with data from the **Archived Sales** cube, which includes sales data from 2008 to 2010. How can you add this data to your query?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 6

Customizing Cube Functionality

Contents:

Module Overview	6-1
Lesson 1: Implementing Key Performance Indicators	6-2
Lesson 2: Implementing Actions	6-8
Lesson 3: Implementing Perspectives	6-11
Lesson 4: Implementing Translations	6-13
Lab: Customizing a Cube	6-15
Module Review and Takeaways	6-19

Module Overview

In this module, you will learn how to customize cube functionality by using several technologies that are available in SQL Server Analysis Services. By customizing cubes, you can improve the user experience, making cubes easier to use.

You will learn about key performance indicators (KPIs) and how they can be used to obtain a quick and accurate summary of business progress. You will find out about actions, and how they enable end users to go beyond traditional analysis to initiate solutions, and to discover problems and deficiencies. This module also covers perspectives and how they enable users to see a simplified version of a cube, so that users can focus on the most relevant data. Finally, you will learn about translations, and how they can be used to translate various elements of a cube, and enable users to view and understand cube and dimension data.

Objectives

After completing this module, you will be able to:

- Implement KPIs
- Implement actions
- Implement perspectives
- Implement translations

Lesson 1

Implementing Key Performance Indicators

In business, a KPI is a quantifiable measurement that is used to understand how well, or badly, an activity is going. KPIs are mostly evaluated over time. KPIs may be specific to a department, or generic for the whole organization.

Generic KPI examples might include gross profit or sales turnover measured monthly. Department-specific KPIs might include staff turnover, and the number of people who have been trained in a month for the human resources department.

KPIs are often viewed together in a business scorecard to give a summary of the health of a business.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe KPIs
- List the elements of a KPI
- Browse a KPI

Introducing KPIs

In SQL Server Analysis Services, a KPI is a group of calculations that are used to evaluate a business activity. KPIs are associated with a measure group in a cube, and can be compared against a business goal to provide trend information over time.

A multidimensional expression in the Multidimensional Expressions (MDX) language can be used to return a KPI status value between -1 (worst) and +1 (best), which is then passed to a client application. A typical graphical representation might be to create a performance dashboard that has traffic light indicators. In this example, a value between -1.0 and -0.34 might display a red traffic light, a value between -0.33 and +0.33 an amber traffic light, and a value between +0.34 and +1.0 a green traffic light. As you will see later in the module, a KPI status value is a value compared against a goal. You monitor KPI status values over time and they provide a trend.

In practice, business rules will determine how to assign the KPI status values. An unsuitable MDX formula will result in a department, or the company, receiving incorrect feedback on their progress.



Best Practice: Getting a KPI measurement right is a business decision rather than an IT decision. A KPI must genuinely reflect an outcome that the business wants to achieve. Care must be taken that the KPI does not encourage damaging behavior. For example, measuring sales turnover by quarter could result in salespeople booking work before client contracts have been signed, or agreeing sales at reduced margins. A KPI that measures repeat purchases, on the other hand, might be a better indicator of customer satisfaction and future business success.

- A key performance indicator (KPI) is a quantifiable business measure
- They are calculated from data held within a cube
- MDX expressions can be used to query cube data
- KPIs are often used on performance dashboards, represented by graphics such as traffic lights
- Designing KPIs is a business-led task
- KPIs in SQL Server Analysis Services can be used by different client applications

An advantage of calculating KPIs by using SQL Server Analysis Services is that the data is held centrally, and can be used by different client applications. When KPIs are calculated locally, they might be calculated by using different datasets, and could be difficult to compare.

SQL Server Analysis Services includes functionality that helps you to define KPI calculations in a multidimensional cube.

For more information about KPIs, see the following article on MSDN:



Key Performance Indicators (KPIs) in Multidimensional Models

<http://aka.ms/Fs3xyp>

Elements of a KPI

KPIs in SQL Server Analysis Services consist of several properties, including:

- Value
- Goal
- Status, and status indicator
- Trend, and trend indicator
- Display folder
- Parent KPI
- Weight
- Current time member

- KPI value: The measure or calculation that represents the performance indicator
- Goal value: The target for the activity
- Status: An MDX expression that compares the KPI value against the goal, and returns a value between -1 and +1
- Trend: An MDX expression that shows how the KPI value compares against the goal over time

Value

A value is the result of a calculation that is defined by using an MDX expression. This is the actual value of the KPI and represents the current position.

Goal

A goal expression is a value, or an MDX expression that resolves to a value, that defines the target for the measure that the value expression describes. For example, a retailer might have a margin goal expression that has a value of 0.6 (or 60 percent) for accessories, and 0.3 (30 percent) for all other categories.

Margin goal of 60 percent for accessories, and 30 percent for all other categories.

MDX Expression

```
Case
  When [Product].[Category].CurrentMember Is
    [Product].[Category].[Accessories]
  Then .60
  Else .30
End
```

Status

A status expression evaluates the current status of the KPI value, compared to the goal. The result is returned within the range of -1 to +1, where -1 is the worst and +1 is the best. The following MDX expression returns a value of 1 if the KPI value is greater than, or equal to, the goal value. It returns a value of -1 if the KPI value is less than 0.5; otherwise, it returns 0. This means that the status is 1 if the goal has been reached, 0 if the value is more than halfway toward the goal, and -1 if the value is less than halfway toward the goal.

The KPI value is compared to the Goal value to return a Status value.

MDX Expression

```
Case
  When KpiValue( "Gross Profit Margin" ) /
    KpiGoal ( "Gross Profit Margin" ) >= 1
  Then 1
  When KpiValue( "Gross Profit Margin" ) /
    KpiGoal ( "Gross Profit Margin" ) < 0.5
  Then -1

  Else 0
End
```

Trend

A trend expression is used to evaluate the current trend of the value expression compared to the goal expression. The trend expression helps a business user to understand whether a KPI value is getting better or worse over time, relative to the goal expression.

You can also associate graphical devices with a trend expression to help business users to quickly understand the trend.

The following code returns a value of 1 if the gross profit margin is increasing over the last year, -1 if it is decreasing, and 0 if it is the same.

KPI Trend Expression

```
Case
  When KpiValue( "Gross Profit Margin" ) >
    (KpiValue( "Gross Profit Margin" ),
     ParallelPeriod
      ( [Date].[Fiscal Time].[Fiscal Year], 1,
        [Date].[Fiscal Time].CurrentMember ))
  Then 1
  When KpiValue( "Gross Profit Margin" ) <
    (KpiValue( "Gross Profit Margin" ),
     ParallelPeriod
      ( [Date].[Fiscal Time].[Fiscal Year], 1,
        [Date].[Fiscal Time].CurrentMember ))
  Then -1
  Else 0
End
```

Additional properties of the KPI include:

- **Display folder.** This is where the KPI will appear if you are browsing the cube.
- **Parent KPI.** This property defines the parent of the current KPI. The browser displays the KPI as a child while allowing the parent to access the values of the child. This feature enables you to have KPIs that are based on other KPIs.
- **Weight.** You can apply a weight to adjust the importance of a child KPI against its siblings.

- **Current time member.** This is an MDX expression that defines the current time member for the KPI.

Browsing KPIs

KPIs in SQL Server Analysis Services are server-based, so they can be used by different client applications. The business logic is within SQL Server Analysis Services, and stored in the cube. In addition, client applications can use MDX expressions with the cube data to retrieve parts of the KPI—such as the value or goal—to use in expressions, statements, and scripts.

Third-party client applications might not display the indicators that you have defined. You should verify that the KPI displays correctly.

The following code example uses MDX functions to return the KPI value and goal for the Reseller Margin KPI of the Adventure Works database for descendants of three members of the Calendar Year attribute hierarchy.

Reseller profit and goal returned by using MDX functions.

Reseller Margin KPI

```
SELECT
{
    KPIValue("Reseller Margin"),
    KPIGoal("Reseller Margin")
} ON COLUMNS,
{
    [Order Date].[Calendar Date].[Calendar Year].&[2006],
    [Order Date].[Calendar Date].[Calendar Year].&[2007],
    [Order Date].[Calendar Date].[Calendar Year].&[2008]
} ON ROWS
FROM [Sales];
```

- Use MDX to retrieve KPI values
- Client applications often include their own graphical indicators for KPIs

Demonstration: Creating a KPI

In this demonstration, you will see how to:

- Create a KPI.
- Browse a KPI in the Cube Designer.
- Browse a KPI in Excel.

Demonstration Steps

Create a KPI

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. In the D:\Demofiles\Mod06 folder, run **Setup.cmd** as Administrator. In the User Account Control window click **Yes**.

2. Start Microsoft® Visual Studio®, and then in the D:\Demofiles\Mod06 folder, open the **Adventure Works OLAP.sln** solution.
3. On the **Build** menu, click **Deploy Solution**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**.
4. After deployment is complete, close the Deployment Progress... window.
5. In Solution Explorer, double-click **Sales.cube** to open it in the Cube Designer, and then click the **Calculations** tab.
6. In the Script Organizer, select **[Reseller Profit]**, and then note that this calculated member calculates reseller gross margin as a percentage of revenue.
7. In the Cube Designer, click the **KPIs** tab, and then, on the **Cube** menu, click **New KPI**.
8. In the **Name** box, type **Reseller Margin**, and then in the **Associated measure group** list, select **Reseller Sales**.
9. In the Calculation Tools pane, on the **Metadata** tab, expand **Measures**, expand **Reseller Sales**, and then drag the **Reseller Profit** measure to the **Value Expression** box.
10. In the **Goal Expression** box, type **0.1** (this sets a gross margin target of 10 percent).
11. Check that **Gauge** is selected in the **Status indicator** list, and then type the following MDX expression in the **Status expression** box (you can copy this from New_KPI_Demo.txt in the demo folder):

```
Case
  When
    KpiValue("Reseller Profit")/KpiGoal("Reseller Profit")>=.95
  Then 1
  When
    KpiValue("Reseller Profit")/KpiGoal("Reseller Profit")<.95
    And
    KpiValue("Reseller Profit")/KpiGoal("Reseller Profit")>=.5
  Then 0
  Else -1
End
```

12. Check that **Standard arrow** is selected in the **Trend indicator** list, and then type the following expression in the **Trend expression** box (you can copy this from New_KPI_Demo.txt in the demo folder):

```
Case
  When IsEmpty(ParallelPeriod([Order Date].[Calendar Date].[Calendar Year],
    1, [Order Date].[Calendar Date].CurrentMember))
  Then 0
  When [Measures].[Reseller Profit] = (ParallelPeriod([Order Date].[Calendar
Date].[Calendar Year],
    1, [Order Date].[Calendar Date].CurrentMember), [Measures].[Reseller Profit])
  Then 0
  When [Measures].[Reseller Profit] >
    (ParallelPeriod([Order Date].[Calendar Date].[Calendar Year],
    1, [Order Date].[Calendar Date].CurrentMember), [Measures].[Reseller Profit])
  Then 1
  Else -1
End
```

13. On the **File** menu, click **Save All**.

Browse a KPI in the Cube Designer

1. On the **Build** menu, click **Deploy Solution**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**.
2. After deployment is complete, close the Deployment Progress... window.
3. In the cube browser, click the **KPIs** tab, and then on the **Cube** menu, point to **Show KPIs in**, and then click **Browser**. The KPI is shown based on the total overall margin.
4. In the top pane of the KPI browser, in the **Dimension** list, select **Sales Territory**, in the **Hierarchy** list, select **Sales Territory**, in the **Operator** list, select **Equal**, and then in the **Filter Expression** list, expand **All**, select **North America**, and then click **OK**.
5. Click anywhere in the KPI browser pane to update the values for the Reseller Margin KPI. The updated values indicate the overall margin for sales in North America.
6. Amend the KPI criteria again. In the top pane of the KPI browser, in the **Dimension** list, select **Order Date**, in the **Hierarchy** list, click **Calendar Date**, in the **Operator** list, select **Equal**, and then in the **Filter Expression** list, expand **All**, select **2006**, and then click **OK**.
7. Click anywhere in the KPI browser pane to update the values for the Reseller Margin KPI. The updated values indicate the overall Reseller margin for sales in 2006.

Browse a KPI in Excel

1. In the Cube Designer, click the **Browser** tab, and then on the **Cube** menu, click **Analyze in Excel**. If a security notice is displayed, click **Enable**.
2. In Microsoft Excel®, in the PivotTable Fields pane, under **Reseller Sales**, select **Reseller Profit**. The overall gross margin is shown in the PivotTable.
3. In the PivotTable Fields pane, under **Order Date**, drag **Order Date.Calendar Date** to the **Rows** area. The gross margin for each year is shown.
4. In the PivotTable Fields pane, expand **KPIs**, expand **Reseller Margin**, and then select **Goal**. The Reseller Margin goal is displayed in the PivotTable.
5. Close Excel without saving the workbook.
6. Close Visual Studio.

Question: What KPIs are used in your business? Do you have KPIs in your department? Does the KPI generally have a positive effect on employee behavior? Can you think of any KPIs that have had unwanted results?

Lesson 2

Implementing Actions

In SQL Server Analysis Services, an action is a stored MDX statement that can be presented to and employed by client applications. An action enables SQL Server Analysis Services to interact with other applications, providing information that can be used to perform a task, such as opening a web page or displaying data. The action is defined on the server and used by a client application.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe actions
- List the types of actions
- Implement actions

Introducing Actions

An action is an MDX statement that is saved as part of a cube definition, and can be used by client applications. Actions are stored on the server, and perform some action that is required on the client side. For example, an action might be to drill through and return a rowset, or to display a web page in a browser. The action is stored in the SQL Server Analysis Services database, so it is important to test whether the client application supports the action and can correctly interpret the events of the action.

A simple action object consists of basic information, the target, a condition to limit the action scope, and the type. The target is the location in the cube where the action is to occur, and consists of a target type and target object.

The target type could be level members, cells, hierarchy, hierarchy members, or others. The target object is specific to the target type. For example, if the target type is hierarchy, the target object is any one of the defined hierarchies in the cube.

A condition is implemented as a Boolean MDX statement that evaluates to true or false. If the statement returns true, the action is performed; if it returns false, the action is not performed.

By saving and reusing actions, end users can go beyond traditional analysis, which finishes with data presentation. Actions enable users to look for solutions to issues that they discover as they work with the data. Actions enable a business intelligence application to go beyond the cube.

You can be flexible when you create actions. For example, an action can launch an application, retrieve information from a database, or open a web page in a browser.

You can configure actions to be triggered from almost any part of a cube, including dimensions, levels, members, and cells, or to create multiple actions for the same portion of a cube.

- Actions are stored statements on the server that respond to client-side requests
- There are various types of actions, including drill-through and reporting actions
- Actions enable users to interact with data

Types of Actions

SQL Server Analysis Services supports different types of actions:

- **CommandLine.** Executes a command at the command prompt.
- **Dataset.** Returns a dataset.
- **Drillthrough.** Returns an expression that the client application then uses to return a rowset.
- **Html.** Executes some HTML script.
- **Proprietary.** Uses a proprietary interface to perform an action.
- **Report.** Returns a report based on a parameterized URL.
- **Rowset.** Returns a rowset.
- **Statement.** Runs an OLE DB command.
- **URL.** Displays a web page.

- Types of actions include:
 - CommandLine
 - Dataset
 - Drillthrough
 - Html
 - Proprietary
 - Report
 - Rowset
 - Statement
 - URL

Typically, actions pass an attribute of the current object, enabling them to be dynamic. For example, you could create a URL action that displays a customer’s location on a map. To create this action, open Microsoft Bing® Maps, search for a postal code, and then click **Share**. Bing will generate a URL that contains the postal code, which you can then remove and append to the original URL.

For more information about actions in SQL Server Analysis Services, see the following article on MSDN:

 **Actions (Analysis Services - Multidimensional Data)**

<http://aka.ms/d2i6vg>

Creating Actions for a Cube

Use the **Actions** tab of the Cube Designer to build actions and specify the following details for standard actions:

1. **Name.** Select a name that identifies the action.
2. **Action Target.** Select the object to which the action is attached. For **Target type**, select from the following objects:
 - Attribute members
 - Cells
 - Cube
 - Dimension members
 - Hierarchy
 - Hierarchy members
 - Level

Standard Actions	Drill-through Actions	Reporting Actions
Name	Name	Name
Target type	Target type	Target type
Target object	Target object	Target object
Condition	Condition	Condition
Action type <ul style="list-style-type: none"> • Dataset • Proprietary • Rowset • Statement • URL 	Drill-through columns <ul style="list-style-type: none"> • Dimensions • Return columns 	Report settings <ul style="list-style-type: none"> • Server Name • Server Path • Report Format
Action expressions	Additional properties	

MCT USE ONLY. STUDENT USE PROHIBITED

- Level members
3. **Action Content.** Select the type of action from the following:
- **Dataset.** This retrieves a dataset.
 - **Proprietary.** This performs an operation by using an interface other than those listed here.
 - **Rowset.** This retrieves a rowset.
 - **Statement.** This runs an OLE DB command.
 - **URL.** This displays a variable page in an Internet browser.
4. **Additional Properties.** Select additional action properties from the following:
- **Invocation.** This specifies how the action is run. Interactive, the default, specifies that the action is run when a user accesses an object. The possible settings are Batch, Interactive, and On Open.
 - **Application.** This describes the application of the action.
 - **Description.** This describes the action.
 - **Caption.** This provides a caption that is displayed for the action. If the caption is generated dynamically by an MDX statement, specify **True** for **Caption is MDX**.
 - **Caption is MDX.** This indicates that the caption is an MDX expression that, after it is executed, will generate the text to be displayed.

To create a new drill-through action, on the **Cube** menu, click **New Drillthrough Action**, and then specify the drill-through columns that are required.

To create a reporting action, on the **Cube** menu, click **New Reporting Action**, and then specify the **Server Name**, **Server Path**, and **Report Format** values.

 **Note:** For **Server Path**, specify the path to the report on the instance of SQL Server Reporting Services. For example, *AdventureWorks/YearlyInternetSales*. For **Report Format**, select **HTML5**, **HTML3**, **Excel**, or **PDF** based on the client tool that your users are most likely to use to access your report.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? Actions are client-side processes that return data to the server.	

Lesson 3

Implementing Perspectives

Cubes can be complex for users to explore in SQL Server Analysis Services.

A single cube can represent the contents of a complete data warehouse, with multiple measure groups that represent multiple fact tables, and multiple dimensions based on multiple dimension tables. Although cubes are powerful, they can be daunting. A user might only need to interact with a small part of the cube to satisfy their business intelligence and reporting requirements.

In SQL Server Analysis Services, you can use a perspective to reduce the perceived complexity of a cube.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe perspectives
- Implement perspectives

Introducing Perspectives

A perspective is a view of a cube, helping users to focus on a specific aspect of cube data.

Although cubes are more intuitive to navigate than relational databases, they can become large if they include many measure groups, measures, and dimensions. However, perspectives do not cause any additional processing overhead, so you can create them for each view of the data that users need.

A simple perspective object consists of basic information, dimensions, measure groups, calculations, KPIs, and actions. Each of these properties defines a subset of the cube as follows:

- Basic information includes the name and default measure of the perspective.
- The dimensions are a subset of the cube dimensions.
- The measure groups are a subset of the cube measure groups.
- The calculations are a subset of the cube calculations.
- The KPIs are a subset of the cube KPIs.
- The actions are a subset of the cube actions.

Objects in a cube that are not visible to the user through a perspective can still be directly referenced and retrieved by using XML for Analysis (XMLA), MDX, or Data Mining Extensions (DMX) statements.

Perspectives do not restrict access to objects in a cube and should not be used as such. Instead, use perspectives to provide a better experience for users as they work with cube data.

For more information about perspectives, see the following article on MSDN:

- A perspective is a view onto a cube
- It defines a subset of the cube's data
- Perspectives reduce the perceived complexity of a cube
- Perspectives can display or hide cube objects, including dimensions, attributes, and more
- The user must already have permissions for the objects in the cube



Perspectives

<http://aka.ms/K7c015>

Defining Perspectives in the Cube Designer

The first column of the **Perspectives** tab in the Cube Designer is **Cube Objects**, which lists all of the objects in the cube. You can add a perspective to the **Perspectives** tab by:

- Clicking **New Perspective** on the **Cube** menu.
- Clicking the **New Perspective** button on the toolbar.
- Right-clicking anywhere in the pane, and then clicking **New Perspective** on the shortcut menu.

- Create a new perspective from the **Perspectives** tab
- Give the perspective a descriptive name
- Select the measures, hierarchies, attributes, KPIs, and other objects to be included

Naming a Perspective

When you create a perspective, the name defaults to Perspective. Additional perspectives default to Perspective 1, Perspective 2, and so on. You should change the name to something meaningful that describes the purpose of the perspective to business users then select the measures, dimension attributes and hierarchies, KPIs, named sets, and calculated members to be included in the perspective.

Removing a Perspective

You can remove a perspective by:

- Clicking any cell in the column for the perspective that you want to delete. Then, on the **Cube** menu, click **Delete Perspective**.
- Clicking the **Delete Perspective** button on the toolbar.
- Right-clicking any cell in the perspective that you want to delete, and then clicking **Delete Perspective** on the shortcut menu.

Check Your Knowledge

Question	
How many perspectives can each cube contain?	
Select the correct answer.	
<input type="checkbox"/>	Each cube can contain no more than one perspective.
<input type="checkbox"/>	Each cube can contain many perspectives.
<input type="checkbox"/>	Each cube must contain one perspective for each user.
<input type="checkbox"/>	The number of perspectives depends on the security rules of the cube.
<input type="checkbox"/>	Cubes do not support perspectives in SQL Server Analysis Services.

Lesson 4

Implementing Translations

Multilanguage support in SQL Server Analysis Services is accomplished by using language translations. In this lesson, you will be introduced to the concepts that are involved in using translations, and how to implement cube and dimension translations.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe translations
- Implement cube translations
- Implement dimension translations

Introducing Translations

A translation is a mechanism to change the displayed labels and captions from one language to another.

Translating elements of a cube into a different language enables more users to view and understand the cube's metadata. Translations are available for all objects in SQL Server Analysis Services, and can also be used to display an alternative column. Therefore, if you have data in multiple languages, a different column could be displayed for an attribute in each language.

A simple translation object is composed of a language ID number—an integer with the language ID—and a translated caption that is the translated text.

In SQL Server Analysis Services, a cube translation is a language-specific representation of the name of a cube object, such as a caption or display folder. There is no automatic translation functionality, so you must provide text for each label in each language.

Translations provide server support for applications that can support multiple languages. For others, the default language is displayed.

The collation and language information for the client computer is stored in a locale identifier (LCID). The following process takes place when a client connects to the cube:

- On connection, the client passes the LCID to the instance of SQL Server Analysis Services.
- The instance uses the LCID to determine which set of translations to use when providing metadata for objects in SQL Server Analysis Services to each business user.
- If an object in SQL Server Analysis Services does not contain the specified translation, the default language is used to deliver content to the client.

- Translations are a mechanism to display localized labels and captions
- Each translation is defined as a pair of values:
 - A string with the translated text
 - A number with the language ID
- There are two types of translations:
 - **Cube translation.** A language-specific representation of the name of a cube object such as a display folder
 - **Dimension translation.** A language-specific representation of the name of a dimension or one of its members

For more information about implementing translations, see the following article on MSDN:



Translation support in Analysis Services

<http://aka.ms/Divre7>

Implementing Cube Translations

In a cube, you can specify language translations for the following objects:

- Measure groups and measures
- Dimensions
- KPIs
- Named sets
- Calculated members

To create a cube translation, click the **New Translation** button on the toolbar of the **Translations** tab in the Cube Designer. To remove a translation, select it, and then click **Delete Translation**.

- You can specify translations for the following objects:
 - Measure groups and measures
 - Dimensions
 - KPIs
 - Named sets
 - Calculated members

Implementing Dimension Translations

You can specify translations for the following objects in a dimension:

- Dimension
- Attributes
- Hierarchies and levels

To create a dimension translation, click the **New Translation** button on the toolbar of the **Translations** tab in the dimension designer. To remove a translation, select the translation, and then click **Delete Translation**.

- You can specify translations for the following objects in a dimension:
 - Dimension
 - Attributes
 - Hierarchies and levels

In addition to specifying translated values for object names, you can also translate attribute members by specifying another column in the dimension table that contains translated values. Select **New Caption Column** to display the **Attribute Data Translation** dialog box, and then define a new caption column when you modify an attribute in the **Translation Details** grid. For example, you could specify a caption column of **French Month Name** for the **Month** attribute of the **Time** dimension. You can use the **Edit Caption Column** and **Delete Caption Column** buttons to modify or delete caption columns.

Question: What do you think are the advantages and disadvantages of implementing translations in a cube?

Lab: Customizing a Cube

Scenario

Information workers in the sales department at Adventure Works Cycles use the cube that you have created to analyze sales data. Several users have asked to be able to view aggregated sales figures in Excel, and then drill through to see details of specific orders for a given aggregation.

Some users are complaining that the cube is too complex for their needs. These users typically only want to analyze sales amount by customer, and would like to view the measures and dimensions that are needed to support this requirement. Finally, the company employs several senior sales managers who would like to view cube data and metadata in French, their first language.

Objectives

After completing this lab, you will be able to:

- Implement an action
- Implement a perspective
- Implement a translation

Estimated Time: 45 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Implementing an Action

Scenario

Information workers in the sales department at Adventure Works Cycles are using the cube that you have created to analyze sales data. Several users want to view aggregated sales figures in Excel, and then quickly drill through to see details of specific orders for a given aggregation. For example, when the users view the sales total for a product category in a specific year, they need to quickly generate a second Excel worksheet that shows details of the customer, location, date, and individual product for each sale in that year.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create a Drill-Through Action
3. Browse a Drill-Through Action

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab06\Starter folder as Administrator.

► Task 2: Create a Drill-Through Action

1. Use Visual Studio to open and deploy the **Adventure Works OLAP.sln** solution in the D:\Labfiles\Lab06\Starter folder.

2. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**.
3. Open **Sales.cube**, and then add a **drillthrough action** named **Internet Sales Details**.
4. The action should use the **Internet Sales** measure group members.
5. The action should return the specified drill-through columns from the following dimensions:
 - a. **Customer (City and Full Name)**
 - b. **Order Date (Full Date Alternate Key)**
 - c. **Product (English Product Name)**
6. Set additional properties so that the drill-through action has the caption **Drillthrough to Order Details**.
7. Save the solution when you have finished.

► Task 3: Browse a Drill-Through Action

1. Deploy the **Adventure Works OLAP** solution. If you are prompted, enter the password **Pa\$\$w0rd**.
2. In the Cube Designer, select the **Browser** tab, and then on the **Cube** menu, click **Analyze in Excel**.
3. Choose **PivotTable Fields**:
 - a. **Internet Revenue** measure
 - b. **Order Date.Calendar Date**
 - c. **Product.Categorized Products**
4. Right-click the **sales** amount for **Bikes** in **2007**, point to **Additional Actions**, and then click **Drillthrough to Order Details** to test the drill-through action.
5. When you have finished, close Excel without saving the workbook.

Results: After this exercise, you should have a new drill-through action in the solution.

Exercise 2: Implementing Perspectives

Scenario

Some of the users in the sales department are complaining that the cube is too complex for their needs. They explain that they typically need to analyze sales amount by customer, and want to view the measures and dimensions that are required to support this.

The main tasks for this exercise are as follows:

1. Create a Perspective
2. Browse a Perspective

► Task 1: Create a Perspective

1. Add a perspective named **Reseller Sales** to the **Sales** cube. The perspective should include only cube objects that are relevant to reseller sales:
 1. Remove all measures for **Reseller Sales**.
 2. Remove the **Reseller** dimension.

3. Remove the **Reseller Profit** calculated member.
4. Add a perspective named **Internet Sales**. The perspective should include only cube objects that are relevant to reseller sales:
 1. Remove the **Internet Sales** measure group.
 2. Remove the **Customer** dimension.
 3. Remove the **Internet Sales Details** action.
 4. Remove the **Internet Profit** calculated member.
 5. Save the solution.

► Task 2: Browse a Perspective

1. In Visual Studio, on the **Build** menu, click **Deploy Solution** to deploy the **Adventure Works OLAP** solution.
2. Select the **Browser** tab on the Sales cube.
3. Click the ellipsis (...) above the Metadata pane to select the **Internet Sales** perspective.
4. In the Metadata pane:
 - a. Expand **Internet Sales**, and then drag **Internet Revenue** to the query results pane.
 - b. Expand the **Customer** dimension, and then drag **Yearly Income** to the query results area.
5. Analyze the cube in Excel to view the **Reseller Sales** perspective.
6. In Excel, in the PivotTable Fields pane, select **Reseller Revenue** from **Reseller Sales** and **Business Type** from **Reseller**.
7. When you have finished browsing the perspectives, close Excel without saving the workbook.

Results: After this exercise, you should have defined a new perspective in the solution.

Exercise 3: Implementing a Translation

Scenario

Adventure Works Cycles employs two French sales managers. They have asked to be able to view cube data, in addition to metadata, in their native language.

In this lab exercise, you will add the final enhancement to the revenue information cube by specifying French translations.

The main tasks for this exercise are as follows:

1. Create Dimension Translations
2. Create Cube Translations
3. Browse Translations

► Task 1: Create Dimension Translations

1. Using the dimension designer for the Date dimension, create a French (France) translation. This should include translated captions for the following items:
 - a. The Calendar Date hierarchy—translation **Date du Calendrier**.

- b. The **Calendar Year** level—translation **Année**.
- c. The **Calendar Semester** level—translation **Semestre**.
- d. The **Calendar Quarter** level—translation **Trimestre**.
- e. The **Month** level—translation **Mois**.
- f. The **Day** level—translation **Journée**.



Note: To type é, press Alt+130. Check that Num Lock is on. Otherwise, type the translation without the accents.

2. Configure the **Month** attribute to use **FrenchMonthName** as a translation column.
3. To display hidden attributes, click the **Show All Attributes** toolbar icon.
4. To configure attribute translation columns, click the France (French) language column for the attribute. Select **FrenchMonthName**.
5. Save the Date dimension when you have finished.

► Task 2: Create Cube Translations

1. Using the Cube Designer for the **Sales** cube, create a new **French (France)** translation that has the following caption translations:
 - a. The **Internet Sales** measure group—translation **Ventes d'Internet**.
 - b. The **Internet Revenue** measure—translation **Revenu d'Internet**.
 - c. The **Order Date** dimension—translation **Date de Vente**.
2. Save the cube when you have finished.

► Task 3: Browse Translations

1. Deploy the solution. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**.
2. Use the **Browser** tab of the Cube Designer to browse the cube, reconnecting if necessary.
3. In the **Language** list, select **French (France)**.
4. View the **Revenu d'Internet** measure by the **Date de Vente.Date du Calendrier** hierarchy.
5. Verify that the French translations are used for the captions and month attribute values.

Results: After this exercise, you should have French data and metadata in the cube.

Question: When you apply the translation feature to a cube, what appears in the specified language?

Question: What does a drill-through action do?

Module Review and Takeaways

In this module, you have learned how to use KPIs, actions, perspectives, and translations. These features are designed to help you to make it easier for users to work with cube data.

Review Question(s)

Question: Can you think of scenarios in your own business where the use of the cube enhancements discussed in this module would help business users?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 7

Implementing a Tabular Data Model by Using SQL Server Analysis Services

Contents:

Module Overview	7-1
Lesson 1: Introduction to Tabular Data Models in SQL Server Analysis Services	7-2
Lesson 2: Creating a Tabular Data Model	7-7
Lesson 3: Using a SQL Server Analysis Services Tabular Model in an Enterprise BI Solution	7-17
Lab: Implementing a Tabular Data Model in SQL Server Analysis Services	7-25
Module Review and Takeaways	7-35

Module Overview

You can install SQL Server® Analysis Services in tabular mode and create tabular data models that information workers can access by using tools such as Microsoft® Excel® and Power View.

This module describes tabular data models in SQL Server Analysis Services and explains how to develop a tabular data model by using SQL Server Data Tools.

Objectives

After completing this module, you will be able to:

- Describe considerations for tabular databases.
- Create a tabular data model.
- Use a tabular model in SQL Server Analysis Services.

Lesson 1

Introduction to Tabular Data Models in SQL Server Analysis Services

This lesson describes tabular data models in SQL Server Analysis Services and explains how you can use SQL Server Data Tools to create a tabular data model.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain tabular data models in SQL Server Analysis Services.
- Describe the options for creating a tabular data model in SQL Server Analysis Services.
- Explain the benefits of using SQL Server Data Tools to develop a tabular data model.
- Describe the purpose of the workspace database.

What Is a Tabular Data Model?

Tabular data models are in-memory databases that use the xVelocity storage and processing engine. The xVelocity engine uses column-based storage and sophisticated compression algorithms to deliver very fast query response times, even with large datasets that contain millions of rows of data. This makes the xVelocity engine ideal for the complex queries that data analysis and reporting tools typically generate.

Tabular data models expose data in a relational format, so the developer interacts with tables and relationships instead of dimensions and cubes.

Tabular data models are quick and easy to create compared to multidimensional data models, although they lack some of the advanced features of multidimensional models. Tabular models are suitable for anything from a personal desktop business intelligence (BI) application that has been developed in Excel, to departmental or larger solutions, depending upon the complexity of the application.

Tabular models offer two principal advantages over multidimensional models:

- The relational model is widely understood and relatively intuitive, so the barrier to entry for relational database developers and information workers wanting to develop analysis solutions, is lower for tabular models than for multidimensional models. This helps companies to minimize costs by taking advantage of their existing internal expertise to create data analysis solutions.
- Tabular data models are generally simpler in design than multidimensional models, so companies can achieve a faster time to deployment for BI applications.

Creators of tabular models can use the Data Analysis Expressions (DAX) language to create measures and calculated columns, and to implement security. DAX is similar to the formulas that are used in Excel workbooks, so information workers who already use Excel should find it relatively easy to learn and use.

- An in-memory database that uses xVelocity storage and processing engine
- Exposes data in a relational format
- Easier to learn for IT professionals who have relational database experience
- Offers a faster deployment time due to simplicity
- Uses DAX
- Can be created by using the PowerPivot add-in in Excel, or SQL Server Data Tools

Tabular data models are generally easy to create because their structure enables developers to interact directly with data, tables, and relationships without having to create complex additional structures, such as dimensions and cubes. Tabular data models help to reduce development effort, enabling developers to get applications into production more quickly, and making it possible for companies to take advantage of a wider pool of talent to develop solutions in SQL Server Analysis Services. Relational models are generally more widely understood than their multidimensional equivalents, so in-house relational database developers should quickly adapt to creating tabular data models with minimal training.

You can create tabular data models by using the PowerPivot add-in in Excel, or by using SQL Server Data Tools. You can use SQL Server Management Studio to manage tabular data model databases in SQL Server Analysis Services after deploying them.



Note: This module focuses on tabular data models in databases in SQL Server Analysis Services, which are typically created by BI developers as part of a managed enterprise BI solution. Information workers can use the PowerPivot add-in to create their own tabular data models in Excel workbooks for personal analysis or to share with colleagues in Microsoft SharePoint® Server. If these PowerPivot data models become important to the business in the future, they can be imported into a tabular database in SQL Server Analysis Services and managed centrally by IT. For more information about PowerPivot, you can attend course number 20778A, *Analyzing Data with Power BI*.

Options for Creating a Tabular Data Model Project in SQL Server Analysis Services

A tabular data model in SQL Server Analysis Services is a database that is stored on an instance of SQL Server Analysis Services, running in tabular mode. You can install an instance of SQL Server Analysis Services in multidimensional, PowerPivot for SharePoint, or tabular mode. To create tabular databases, you must install at least one instance of SQL Server Analysis Services that runs in tabular mode.

- Install SSAS instance that runs in tabular mode
- Create a tabular data model project by using:
 - SQL Server Data Tools for BI templates:
 - Analysis Services Tabular Project
 - Import from PowerPivot
 - Import from Server (Tabular)
 - SQL Server Management Studio:
 - Restore from PowerPivot
- Key features in Analysis Services tabular databases
 - DirectQuery mode
 - Row-level security
 - Partitions



Note: You can switch the mode of an instance of SQL Server Analysis Services after installation, but only if you have not yet created a database on that instance.

You can create a tabular data model project for SQL Server Analysis Services by using the SQL Server Data Tools project templates. Each of the three tabular data model templates enables you to create a project in a different way:

- **Analysis Services Tabular Project.** This template creates a new, empty tabular data model. You must import data and metadata from your data sources to populate the model.
- **Import from PowerPivot.** This template creates a new tabular data model by using the tabular data model that is embedded in a PowerPivot for Excel workbook. The Import from PowerPivot option extracts the data and metadata from the specified workbook and uses this to populate the new model.

- **Import from Server (Tabular).** This template creates a new tabular data model from an existing tabular data model in SQL Server Analysis Services. It extracts the data and metadata from the specified database and uses these to populate the new model.

Before you create a tabular data model project, you must ensure that there is an instance of SQL Server Analysis Services running in tabular mode that is available for the project to use during development. The instance of SQL Server Analysis Services can be local or located on a network. A local instance will typically provide better performance during the development phase, and makes it possible to work with the model offline. However, a network instance offers easier collaboration if several developers are involved, particularly if the project is integrated with Team Foundation Server.

You can also create a tabular data model by using the **Restore from PowerPivot** option in SQL Server Management Studio. This method enables you to create a tabular data model database directly from a PowerPivot for Excel workbook, without having to use SQL Server Data Tools. Like the **Import from PowerPivot** template in SQL Server Data Tools, the **Restore from PowerPivot** option extracts the data and metadata from the specified PowerPivot for Excel workbook and uses this to populate the tabular data model database.

Key Features

The key features that you can configure in a tabular database in SQL Server Analysis Services are:

- DirectQuery mode, which passes queries directly to the underlying data source instead of using the in-memory storage engine.
- Row-level security, which enables you to implement security at a much more granular level.
- Partitions, which enable you to manage large tables more efficiently.

Using SQL Server Data Tools to Develop a Tabular Data Model in SQL Server Analysis Services

SQL Server Data Tools is based on Microsoft Visual Studio®. It is a comprehensive single development environment for all of your BI projects, including projects in SQL Server Reporting Services, and multidimensional and tabular projects in SQL Server Analysis Services. It supports a range of features to enable you to manage the full project development life cycle, including:

- Templates to create projects.
- Source control, if it is used in conjunction with Team Foundation Server or other third-party source control plug-ins.
- Debugging tools.
- Building and deployment directly to test and production servers.

When you create a tabular data model in SQL Server Data Tools, the tabular model designer opens automatically. It consists of several windows, including:

- **Solution Explorer.** This window displays the project and its contents. By default, the contents are the **References** object and the **Model.bim** file. You can set the properties of the project, such as the deployment server or the query mode, from this window.

- Single development environment for BI developers that supports a range of features:
 - Templates to create projects
 - Integrated project life cycle management and source control
 - Debugging tools
 - Building and deployment directly to test and production servers
- Tabular Model Designer:
 - Data view and diagram view
 - Measure grid
 - Table Import Wizard

- **Designer window.** This window displays a visual representation of the model, offering two different views:
 - **Data view.** This view displays one table at a time, showing the columns and the data that the table contains. You can select the table that you want to view by clicking the appropriate tab. The data view contains the measure grid, which you can use to create measures and key performance indicators (KPIs). You can also create calculated columns in the data view by adding a DAX expression to the formula bar.
 - **Diagram view.** This view presents the tables in a schema-like diagram. It shows the columns and hierarchies in each table, and the relationships between the tables. You can also use this view to create and manage hierarchies.
- **Properties window.** This window displays the properties of the object that you select.
- **Error list.** This window displays errors and other messages that relate to the model.
- **Output window.** This window displays status information that relates to builds and deployments.

The tabular model designer also contains extra menu items, including **Model**, **Table**, and **Column**.

After creating a tabular data model in SQL Server Data Tools, you can import the necessary data and metadata by using the Table Import Wizard. This creates all of the metadata structures that underpin a tabular data model. You do not need to create data source views, dimensions, or cubes. You will see how to use the Table Import Wizard later in this module.

The Workspace Database

Creating a tabular data model project by using one of the templates in SQL Server Data Tools automatically adds a new database to the tabular instance of SQL Server Analysis Services that is configured as the workspace server. Every tabular data model project has its own exclusive workspace database. You can view the workspace database on the SQL Server Analysis Services server, using SQL Server Management Studio. Workspace databases use a naming convention that combines the project name, the name of the user who created it, and a system-generated GUID in the following format:

`<project name>_<user name>_<GUID>`

This naming format ensures that every workspace database has a unique name.

The workspace database is an in-memory database that contains all of the data and metadata for the project while it is being developed. All of the changes that you make to the model in SQL Server Data Tools, such as creating hierarchies or adding linked tables, result in updates to the workspace database. Deploying the model creates a new database—a renamed copy of the workspace database—on the designated instance of SQL Server Analysis Services.

Local and Remote Workspace Databases

Ideally, the workspace database should be located on the same computer where the developer creates the model because this is likely to result in the best performance. You can use a remote instance of SQL Server Analysis Services to host the workspace database, but this configuration has the following limitations:

- You cannot create tabular data model projects by using the Import from PowerPivot template.

- Created on SSAS instance when tabular data model project is created
- Contains all data and metadata of project
- Can use local or remote SSAS instance
- Configure the workspace database:
 - **Workspace Server**
 - **Workspace Retention**

- You cannot use the **Backup to disk** option in the **Data Backup** property.
- You might experience slower performance because of the latency introduced by using a remote server.

Configuring the Workspace Database

The **Properties** window of the **Model.bim** file in SQL Server Data Tools enables you to configure the workspace database.

- The **Workspace Server** property enables you to specify the instance of SQL Server Analysis Services that hosts the workspace database.
- The **Workspace Retention** property defines what happens to the in-memory workspace database when you close a tabular data model project. The default **Workspace Retention** setting is **Unload from memory**, which removes the database from memory and stores it on disk. This setting frees up memory, but the model takes longer to load when you reopen the project. You can also configure the **Workspace Retention** setting to keep the workspace database both in memory and on disk when you close a project, or to delete it completely. If you choose this last option, the workspace database is recreated every time you open the project.



Note: The default host for new workspace databases is defined in the **Default workspace server** setting on the **Workspace Database** page. To access the **Workspace Database** page, on the **Tools** menu, click **Options**, and then, in the **Options** dialog box, expand **Analysis Services Tabular Designers**, and click **Workspace Database**.

Question: What are the advantages and disadvantages of tabular models when compared to multidimensional models?

Lesson 2

Creating a Tabular Data Model

A tabular data model consists of multiple tables, usually linked by using relationships. Columns in the tables can be used as attributes and measures that enable business users to aggregate key business values. You can organize the attributes by which values are aggregated into hierarchies to enable drill-down and drill-up analysis.

Lesson Objectives

After completing this lesson, you will be able to:

- Import tables into a tabular data model project.
- Define measures in a tabular data model.
- Manage relationships between tables in a tabular data model.
- Configure columns in a tabular data model.
- Create hierarchies in a tabular data model.

Importing Tables

The first step in creating a tabular data model is to import tables of data from one or more sources. The available data sources for a tabular model include:

- An instance of the SQL Server database engine.
- SQL Server Analysis Services cubes.
- Microsoft Azure® SQL Database.
- Microsoft Excel.
- Microsoft Access®.
- Text files.
- SQL Server Reporting Services reports.
- SharePoint lists.
- Data feeds.
- Databases running on Oracle, Teradata, Sybase, Informix, and IBM DB2 database management systems.
- OLE DB and Open Database Connectivity (ODBC) providers that enable imports from any OLE DB or ODBC source.

- Create data source connections for a wide range of connection options including common third-party databases
- Filter out columns that are not required for analysis:
 - Improves performance
 - Simplifies user experience
- Provide table aliases for ease of use

When importing from a database, you can filter tables to exclude columns that are not required for your analysis. You should always aim to only import the necessary data into the data model to ensure optimal performance and simplify the end-user experience. You should also provide a friendly name, or alias, for the tables that you import. Tables in databases do not usually have easy-to-understand names, so providing a friendly name will make the data easier for the end user to interpret.

You can import tables by using the Table Import Wizard in SQL Server Data Tools. To access the Table Import Wizard, on the **Model** menu, click **Import From Data Source**.

Defining Measures

Measures are numeric values in your data model that users can aggregate to analyze business performance. Unlike traditional multidimensional data models, in which measures can only be defined as part of a measure group—usually based on a fact table—tabular data models enable you to define measures for any numeric column in any table.

Measures are aggregated across multiple dimensions within the data model, and typically provide a summary value for core business metrics.

For example, a data model that includes a sales order table will typically define measures that sum sales revenue, cost, and order quantities by attributes of business entities that are defined in other tables, such as customers, products, and stores.

You can define measures in the Measure Grid area of the tabular model designer in SQL Server Data Tools. Measures are usually based on a simple DAX function that is applied to a numeric column. For example, a measure named **Revenue** could be defined as the sum of values in the **SalesAmount** column by using the DAX expression **Revenue:=Sum([SalesAmount])**.

Sum is the most common aggregate function that is used to define measures, but you can also use **Average**, **Count**, **Max**, **Min**, and many others. You can also define more complex measures that reference multiple columns, or use custom algorithms.

- Measures are numeric aggregations of business metrics
- They are usually based on a simple DAX function that aggregates a numeric column value

```
Revenue:=Sum([SalesAmount])
```

Demonstration: Creating a Tabular Data Model Project

In this demonstration, you will see how to:

- Create a tabular data model project.
- Import tables into a tabular model.
- Define measures in a data model.
- Test the data model in Excel.

Demonstration Steps

Create a Tabular Data Model Project

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. In the D:\Demofiles\Mod07 folder, run **Setup.cmd** as Administrator.
2. Start SQL Server Data Tools 2015, and then on the **File** menu, point to **New**, and then click **Project**.
3. In the **New Project** dialog box, type the following values, and then click **OK**:
 - **Project Template:** Analysis Services Tabular Project
 - **Name:** TabularDemo
 - **Location:** D:\Demofiles\Mod07

4. If the **Tabular model designer** dialog box is displayed, type the following values, and then click **OK**:
 - **Workspace server:** localhost\SQL2
 - **Compatibility level:** SQL Server 2016 RTM (1200)
5. On the **Tools** menu, click **Options**, and then in the **Options** dialog box, expand **Analysis Services Tabular Designers**, and then click **Workspace Database**. Note the name of the default workspace server that is used to host the data model during development, and then click **Cancel**.

Import Tables into a Tabular Model

1. On the **Model** menu, click **Import From Data Source**.
2. In the Table Import Wizard, on the **Connect to a Data Source** page, select **Microsoft SQL Server**, and then click **Next**.
3. On the **Connect to a Microsoft SQL Server Database** page, type the following values, and then click **Next**:
 - **Friendly connection name:** AdventureWorksDW
 - **Server name:** MIA-SQL
 - **Log on to the server:** Use Windows Authentication
 - **Database name:** AdventureWorksDW
4. On the **Impersonation Information** page, type the following values, and then click **Next**:
 - **User Name:** ADVENTUREWORKS\ServiceAcct
 - **Password:** Pa\$\$w0rd
5. On the **Choose How to Import the Data** page, ensure that **Select from a list of tables and views to choose the data to import** is selected, and then click **Next**.
6. On the **Select Tables and Views** page, select the following source tables, changing the **Friendly Name** value as indicated in parentheses:
 - **DimCustomer** (Customer)
 - **DimDate** (Date)
 - **DimGeography** (Geography)
 - **DimProduct** (Product)
 - **DimProductCategory** (Product Category)
 - **DimProductSubcategory** (Product Subcategory)
 - **FactInternetSales** (Internet Sales)
7. Select the row for the **DimCustomer** table, click **Preview & Filter**, clear the check box at the top of the following columns, and then click **OK**:
 - Title
 - MiddleName
 - NameStyle
 - Suffix
 - SpanishEducation
 - FrenchEducation

- SpanishOccupation
 - FrenchOccupation
8. Select the row for the **DimDate** table, click **Preview & Filter**, clear the check box at the top of the following columns, and then click **OK**:
 - SpanishDayNameOfWeek
 - FrenchDayNameOfWeek
 - SpanishMonthName
 - FrenchMonthName
 - CalendarSemester
 - FiscalSemester
 9. When you have selected and filtered the tables, in the **Table Import Wizard** dialog box, click **Finish**, and then wait for the data to be imported. When the data has been imported successfully, click **Close**.
 10. Click each of the tabs in the model designer to see the data that has been imported into each table.

Define Measures in a Data Model

1. Click the **Internet Sales** tab, and then click the first empty cell in the measure grid under the **SalesAmount** column.
2. On the **Column** menu, point to **AutoSum**, and then click **Sum**.
3. In the formula bar, modify the expression that has been generated to change the name of the measure to **Revenue**, as shown in the following example:

```
Revenue:=SUM([SalesAmount])
```

4. Widen the **SalesAmount** column, and then note that the measure is calculated and displayed as a currency value. The formatting has been inherited from the column.
5. Select the cell that contains the **Revenue** measure, and then press F4 to display the properties pane. View the **Format** property, which is set to **Currency**.
6. Create a second measure named **Cost** in the measure grid under the **TotalProductCost** column based on the following expression, and format the **Cost** measure as currency:

```
Cost:=Sum([TotalProductCost])
```

7. On the **File** menu, click **Save All**.

Test the Data Model in Excel

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and then click **OK**.
3. When Excel opens, note that the measures that you defined are displayed under an **Internet Sales** measure group in the PivotTable Fields pane. The **Internet Sales** table is also shown in this pane, and contains the columns in the table, including those on which the aggregated measures are based. Some users might find this confusing so, in a future refinement of the data model, the columns that are not useful for analysis should be hidden.
4. Select the **Revenue** measure so that it is summarized in the PivotTable.

5. In the **Date** table, select **CalendarYear** so that the revenue in the PivotTable is aggregated by year. Note that there is no indication in the **Date** table that tells the user if the values that are displayed represent the order date, the due date, or the ship date that is associated with the order.
6. In the **Date** table, select **EnglishMonthName** so that the revenue is further broken down into monthly totals. Note that the months are shown in alphabetical order instead of chronological order.
7. Close Excel without saving the workbook.
8. Keep SQL Server Data Tools open for the next demonstration.

Managing Relationships

When you import data from a table in a relational database, you can choose to automatically import additional tables that have a relationship with that table. The Table Import Wizard identifies related tables by analyzing their foreign-key relationships. However, data analysts frequently need to analyze data that comes from multiple sources and for which explicit relationships are not defined. You can manually create, edit, and delete relationships as required in tabular data model projects.

When you create a relationship, you must specify the two tables that you want to relate and the common column that contains the key values. Relationships between tables in tabular data models are one-to-many, so, when you create a relationship, you must specify the columns and tables for each end of it. To create a relationship, in the diagram view of the data model designer, drag the column in one of the tables in the relationship to the column in the other table in the relationship.

Tabular data models support multiple relationships between tables. For example, date and fact tables in data warehouses often have multiple relationships such as order date, due date, and delivery date. However, only one relationship is marked as being active between the tables. When you view multiple relationships in the diagram view of the data model designer, the active relationship shows as an unbroken line, and inactive relationships appear as dotted lines.

If you need to implement a "role-playing dimension", in which multiple relationships between a fact table and a dimension table must be active concurrently, you must import the dimension table multiple times, assigning a different name each time, and create individual relationships between the fact table and each copy of the dimension table. For example, if you needed to implement order date, due date, and delivery date role-playing dimensions based on a date dimension table, you would need to import the date dimension table three times, assigning an appropriate name to each copy of the table. You would then create three individual relationships between each of the keys that were used for order date, due date, and delivery date in the fact table, and the appropriate copy of the date dimension table.

- Automatically recognize relationships based on foreign keys
- Manually create relationships when they are not explicitly defined
- Only one relationship between two tables can be active
- To define role-playing dimensions, import the same table multiple times

Configuring Columns

The columns in the tables that you import into a tabular data model become the attributes by which analysts can aggregate measures for analysis. For example, a **Customer** table might include **City** and **Gender** columns, enabling analysts to aggregate numerical measures such as revenue by city, or order quantity by customer gender.

Specifying Column Data Types

When you have imported the tables for your data model, you can specify the data type of each column in the **Properties** window. For example, if the **Customer** table includes a **DateOfBirth** column, you can specify that it contains date values.

Creating Calculated Columns

In addition to columns in the source table, you can define additional attributes by creating calculated columns. A calculated column is based on a DAX formula, which uses a similar syntax to a formula in Excel. For example, in the **Customer** table that was described previously, you could concatenate the **FirstName** and **LastName** columns to create a calculated column named **FullName**, based on the DAX expression `= [FirstName] & " " & [LastName]`, or create a calculated column named **YearOfBirth**, based on the DAX expression `= YEAR([DateOfBirth])`.

Specifying Sort Order for Columns

You can define sort orders for columns, including sorting a column based on the value in another column in the same table. This approach is most commonly used in time dimension tables that contain numeric values—such as a **MonthOfYear** column that has values between 1 and 12—and text values, such as a **MonthName** column that has values from “January” to “December.” In this example, you can specify that the **MonthName** column should be sorted based on the corresponding **MonthOfYear** value. You can define a sort order for a column in the **Sort by Column** dialog box, by clicking the column heading in the data view of the data model designer, and then, on the **Column** menu, pointing to **Sort**, and clicking **Sort by Column**.

Hiding and Renaming Columns

By default, all columns in all tables are visible when the data model is browsed in a client tool such as Excel. This includes columns that are not useful for aggregation, such as those on which you have based measures, and columns that are encapsulated in hierarches. You can make the data model simpler for users to browse by hiding columns that are not useful as attributes for aggregation. You can also rename the columns that you want to show in the client tool to a more user-friendly name. To hide or rename a column in the diagram view of the data model designer, maximize the table that contains the column, right-click the column, and then click **Hide from Client Tools** or **Rename**.

- Specify column data types
- Create calculated columns based on DAX expressions
- Specify sort orders for columns
- Hide and rename columns

Demonstration: Managing Relationships and Columns

In this demonstration, you will see how to:

- Manage relationships.
- Rename and hide columns.
- Configure column sort order.
- Review your changes in Excel.

Demonstration Steps

Manage Relationships

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Data Tools, with the Model.bim pane visible, on the **Model** menu, point to **Model View**, and then click **Diagram View**. The tables are shown as a schema diagram, with lines between them to denote relationships. Note that there are three relationships between the **Internet Sales** table and the **Date** table.
3. Double-click the solid line between the **Internet Sales** and **Date** tables. Note the columns that are used to define this relationship, and that this is an active relationship, and then click **Cancel**.
4. Double-click each of the dotted lines between the **Internet Sales** and **Date** tables, and then click **Cancel**. In each case, note the columns that are used to define these relationships and that these are inactive relationships, and then click **Cancel**.
5. Right-click each dotted relationship line in turn, and then click **Delete**. When you are prompted to permanently delete the relationship from the model, click **Delete from Model**.
6. Double-click the remaining solid line between the **Internet Sales** and **Date** tables.
7. In the **Edit Relationship** dialog box, in the **Columns** list for the **Internet Sales** table, select **OrderDateKey** if it is not already selected, and then click **OK**. This ensures that the active relationship is based on the order date, not the shipping date or due date.
8. Right-click the **Date** table title bar, click **Rename**, and then rename the table to **Order Date**.
9. On the **Model** menu, click **Existing Connections**.
10. In the **Existing Connections** dialog box, ensure that the **AdventureWorksDW** connection is selected, and then click **Open**. If you are prompted for impersonation credentials, type the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **OK**.
11. On the **Choose How to Import the Data** page, ensure that **Select from a list of tables and views to choose the data to import** is selected, and then click **Next**.
12. On the **Select Tables and Views** page, select the **DimDate** table, and then change the **Friendly Name** value to **Ship Date**.
13. Select the row for the **DimDate** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - SpanishDayNameOfWeek
 - FrenchDayNameOfWeek
 - SpanishMonthName
 - FrenchMonthName
 - CalendarSemester

- FiscalSemester
14. Click **Finish**, and then wait for the table to be imported. When it has been imported successfully, click **Close**.
 15. Arrange the diagram so that you can see the **Ship Date** and **Internet Sales** tables, and then drag the **ShipDateKey** column from the **Internet Sales** table to the **DateKey** column in the **Ship Date** table to create the relationship. If the **Ship Date** table does not appear, refresh the page.

Rename and Hide Columns

1. Click the title bar of the **Order Date** table, and then click its **Maximize** icon.
2. In the maximized **Order Date** table, click the **DateKey** column, hold down CTRL, click the **DayNumberOfWeek** and **MonthNumberOfYear** columns to select them, right-click any of the selected columns, and then click **Hide from Client Tools**.
3. In the maximized **Order Date** table, right-click the **FullDateAlternateKey** column, click **Rename**, and then rename the column to **Date**.
4. Repeat the previous step to rename the following columns:
 - EnglishDayNameOfWeek (rename to **Weekday**)
 - EnglishMonthName (rename to **Month**)
5. Click the **Restore** icon for the **Order Date** table.
6. Maximize the **Internet Sales** table, hide all columns other than the **Revenue** and **Cost** measures that you created in the previous exercise, and then restore the **Internet Sales** table.

Configure Column Sort Order

1. On the **Model** menu, point to **Model View**, and then click **Data View**.
2. On the **Order Date** tab, click the **Weekday** column heading. In the **Column** menu, point to **Sort**, and then click **Sort by Column**.
3. In the **Sort By Column** dialog box, in the **Sort** column, ensure that **Weekday** is selected, in the **By** column, select **DayNumberOfWeek**, and then click **OK**.
4. Click the **Month** column heading. In the **Column** menu, point to **Sort**, and then click **Sort by Column**.
5. In the **Sort By Column** dialog box, in the **Sort** column, ensure that **Month** is selected, in the **By** column, select **MonthNumberOfYear**, and then click **OK**.
6. On the **File** menu, click **Save All**.

Review Your Changes in Excel

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and then click **OK**.
3. When Excel opens, note that the measures that you defined are still displayed under an **Internet Sales** measure group in the PivotTable Fields pane, but that the **Internet Sales** table is no longer shown here because all other columns have been hidden.
4. Select the **Revenue** measure so that it is summarized in the PivotTable.
5. In the **Order Date** table, select **CalendarYear** so that the revenue in the PivotTable is aggregated by year based on the order date.

6. In the **Order Date** table, select **Month** so that the revenue is further broken down into monthly totals. Note that the months are now shown in chronological order.
7. In the **Order Date** table, select **CalendarQuarter**, and then note that the quarter is shown under each month.
8. In the PivotTable Fields pane, in the **Rows** area, drag **CalendarQuarter** so that it is above **Month**, and the PivotTable shows the correct hierarchy for year, quarter, and month. This user experience could be improved by enabling users to select a hierarchy that automatically enables them to drill down to the correct levels.
9. Close Excel without saving the workbook.
10. Keep SQL Server Data Tools open for the next demonstration.

Creating Hierarchies

A hierarchy is a collection of attributes from a single table and it is organized as parent and child nodes. You create hierarchies to simplify the end-user experience and enable drill-up and drill-down aggregations. For example, you might have **Product Category, Subcategory, and Product Name** columns in a table. These values represent a natural hierarchy in which you can organize data, enabling users to view aggregations such as total sales revenue at the Product Category, Product Subcategory, and Product Name levels. Another common hierarchy is Year, Quarter, and Month in a time dimension table.

- Hierarchies enable drill-up and drill-down aggregations
- Common examples:
 - Product Category, Product Subcategory, and Product Name
 - Year, Quarter, Month, and Date
- Create hierarchies in the diagram view

You can manage hierarchies for data in a table by maximizing the table in the diagram view of the data model designer. To create a hierarchy, on the title bar of the maximized table, click the **Create Hierarchy** icon. To add a column to a hierarchy, in the maximized table, right-click the column, point to **Add to Hierarchy**, and then click the name of the hierarchy.

Demonstration: Creating a Hierarchy

In this demonstration, you will see how to:

- Create a hierarchy.
- Browse the hierarchy in Excel.

Demonstration Steps

Create a Hierarchy

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Data Tools, with the Model.bim pane visible, on the **Model** menu, point to **Model View**, and then click **Diagram View**.
3. Maximize the **Order Date** table, and then on its title bar, click the **Create Hierarchy** icon. Name the new hierarchy **Calendar Date**.
4. Drag the **CalendarYear** column to the **Calendar Date** hierarchy.

5. Drag the **CalendarQuarter** column to the **Calendar Date** hierarchy.
6. Drag the **Month** column to the **Calendar Date** hierarchy.
7. Drag the **Date** column to the **Calendar Date** hierarchy.
8. Restore the **Order Date** table, and then on the **File** menu, click **Save All**.

Browse the Hierarchy in Excel

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and then click **OK**.
3. When Excel opens, select the **Revenue** measure so that it is summarized in the PivotTable.
4. In the **Order Date** table, select **Calendar Date** so that the revenue in the PivotTable is aggregated by year.
5. In the PivotTable, expand the years to show the revenue by quarter, expand the quarters to view revenue by month, and then expand the months to view revenue for individual dates.
6. Close Excel without saving the workbook.
7. Keep SQL Server Data Tools open for the next demonstration.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
<p>You are implementing an order date and a ship date role-playing dimension based on a date dimension table. You can achieve this by importing the fact table and the date dimension table once, and then creating a relationship between the order date key in the fact table and the date key in the date dimension table, and also creating a relationship between the ship date key in the fact table and the date key in the date dimension table.</p>	

Lesson 3

Using a SQL Server Analysis Services Tabular Model in an Enterprise BI Solution

SQL Server Analysis Services includes tabular data model capabilities that can be useful in an enterprise BI scenario. It enables you to define perspectives as subsets of the data model for specific user groups, and partition the tables in the data model to improve data refresh processing performance. There is also a choice of query modes to help you to optimize the right balance of query and processing performance, and the ability to define security restrictions on a data model.

Lesson Objectives

After completing this lesson, you will be able to:

- Create perspectives in a tabular data model.
- Create partitions in a tabular data model.
- Process tabular databases, tables, and partitions.
- Implement DirectQuery mode in a tabular data model.
- Implement security in a tabular data model.
- Deploy a tabular data model.

Perspectives

Tabular data models can contain large numbers of tables and measures, which can make it difficult for users to work with the data in a client tool such as Excel. You can simplify the user experience by creating perspectives that display only a subset of the tables, columns, and measures in a model. For example, the tables in your tabular data model might contain many different types of data, such as human resources, sales, products, promotions, and financial data. However, users who want to analyze sales performance will probably not need to view the financial or human resources tables, so you could create a perspective where these are explicitly excluded. When users connect to the model with this perspective, they will not see the excluded tables in their client application.

- Provide a simplified view of complex data models
- Select tables, measures, and columns to include in perspective
- Not a method of implementing security

Creating Perspectives

You can create a perspective in a tabular data model in SQL Server Analysis Services by using the **Perspectives** dialog box in SQL Server Data Tools. To open this dialog box, on the **Model** menu, click **Perspectives**.

When you create a perspective, you must provide a name for it and specify the tables, measures, and columns to include. When users connect to the model, they will be able to select the perspective that they want to use. You should ensure that the names that you give to perspectives are descriptive and easy to understand. To speed up the process, you can also create a perspective by copying an existing similar one.

Perspectives and Security

You cannot use perspectives to secure tabular data models. You cannot define permissions on a perspective, so users who have the right to view the tabular data model will be able to use any perspective that is defined in the model. In addition, if a user does not have permissions to view certain tables, they cannot circumvent this by using a perspective. If you need to secure your tabular data models, you should create roles and define permissions for the required objects. You will learn about security for tabular data models later in this module.

Partitions

Partitions enable you to manage large tables more efficiently. You can divide a table into logical units that you can load or reload individually. For example, you could use a large table called **Sales** in your tabular data model with a data warehouse fact table, which is updated daily, as the source. In this example, the data would quickly go out of date.

You could refresh the data in **Sales**, but this operation is time-consuming because it reloads all of the rows from the source table. To keep **Sales** synchronized with the source table more efficiently, you could create partitions that divide up the table, perhaps by using the month or year of the order date. Then, you would only need to regularly refresh the partition that contains the most recent data, updating the other partitions only when required.

- Divide tables into logical partitions
- Separate frequently changing data from static data
- Create partitions in workspace and deployed databases

 **Note:** Partitions in tabular data models are not intended to improve query performance in the way that partitions in other types of databases can. Partitioning can, however, improve the performance of data model processing by enabling you to process only partitions that contain new or updated data.

Creating Partitions

Every table that you import into a tabular data model has a single partition that is associated with it by default. This enables you to refresh each table separately. During the development stage, you can create a new partition in a tabular data model by performing the following steps:

1. In SQL Server Data Tools, in the tabular model designer, on the **Table** menu, and then click **Partitions**.
2. In the **Partition Manager** dialog box, select the table for which you want to create a partition, and then click **New**. You can also create a partition by copying an existing one.
3. To select the rows to include in the partition, use the Table Preview pane, or use the Query Editor pane to define a Transact-SQL query. The partitions that you create in this way are added to the workspace database for the project.

For tabular data models that are deployed to an instance of SQL Server Analysis Services, you can create and manage partitions by using SQL Server Management Studio.

Using Partitions in the Development Phase

When you are creating a tabular data model by using a source database that has large tables, you should consider implementing partitions to enable you to work with a reduced sample data set. Using a sample data set during development is more efficient because it reduces data load times and enables better query performance. You can create, delete, edit, and merge partitions in the workspace database as required. When you deploy the model, you can import all of the data and employ a different partitioning design that is appropriate to the deployed database.

Analysis Services Processing Types

Processing populates the tables and partitions in a tabular data model by importing data from the source database. You can choose to process individual tables or partitions, or all of the tables and partitions in a model together. Depending on the option that you choose, processing rebuilds relationships and hierarchies in addition to recalculating calculated columns and measures.

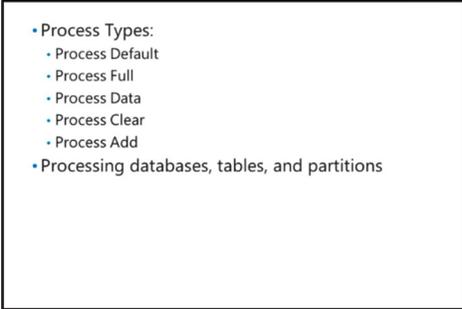
The following processing types are available:

- **Process Default.** This option detects the current state of a table or partition and performs the necessary actions to ensure that it is in a fully processed state. When you first create a table or partition, this option loads it and creates all hierarchies, relationships, measures, and calculated columns.
- **Process Full.** This option fully processes a table or partition. It removes existing data and repopulates partitions, before recreating hierarchies, relationships, measures, and calculated columns.
- **Process Data.** This option only processes data—it does not recreate hierarchies, relationships, measures, and calculated columns.
- **Process Clear.** This option deletes data from a table or partition.
- **Process Add.** This option updates tables or partitions by only adding new data.

You can process partitions by using the **Process Partitions** dialog box. To access the **Process Partitions** dialog box in SQL Server Data Tools, on the **Model** menu, point to **Process**, and then click **Process Partitions**. To access the **Process Partition(s)** dialog box in SQL Server Management Studio, right-click the table that you need to update, point to **Partitions**, and then click **Process**.

You can also use SQL Server Management studio to process tabular databases, tables, or partitions. For example, to process a database, right-click the database, and then click **Process Database**, select a processing type, and then click **OK**.

 **Note:** In SQL Server 2016 Analysis Services, when you process a table or database with more than one partition, the partitions are processed in parallel in order to maximize performance. By using the TMSL Sequence command, you can specify sequential processing, in which each partition is processed in turn but this is beyond the scope of this course.

- 
- Process Types:
 - Process Default
 - Process Full
 - Process Data
 - Process Clear
 - Process Add
 - Processing databases, tables, and partitions

For more information about the Sequence command, see:



Sequence command (TMSL)

<https://aka.ms/ogen8x>

DirectQuery Mode

The xVelocity engine provides in-memory storage and data processing for very fast query handling, and is appropriate for most tabular data models in SQL Server Analysis Services. However, you can also configure a tabular data model to bypass the xVelocity engine and route queries directly to the underlying data source by using DirectQuery mode. This option might be suitable in the following circumstances:

- Bypass xVelocity in-memory storage
- Retrieve data directly from a relational database
- Take advantage of powerful server hardware
- Appropriate for very large data sets
- Some feature limitations to consider

- For tabular data models that have very large data sets that take a long time to load into memory, and which might fail to load if there is insufficient memory available. Using DirectQuery removes the need to load data into memory.
- For tabular data models that need to contain up-to-date information and for which frequent processing is impractical. DirectQuery retrieves source data directly, providing access to the most recent data.
- For organizations that have invested in high-specification data warehouse hardware that can deliver excellent performance. DirectQuery enables companies to take advantage of their hardware performance while still providing access to data through a tabular data model. In this situation, you should test thoroughly to determine whether you can achieve the best performance by relying on the data warehouse hardware or by using the xVelocity engine.

Designing Tabular Data Models for Use with DirectQuery

When you design tabular data models for which you intend to enable DirectQuery mode, there are several points to consider:

- DirectQuery only supports a single data source, which must be a SQL Server relational database, an Oracle relational database, or a Teradata relational database.
- You cannot use certain DAX functions, such as those for time intelligence, in the model. Others, such as certain statistical functions, might behave differently in DirectQuery mode.
- There are limitations to the Multidimensional Expressions (MDX) that you can use in DirectQuery mode.

For further details about the limitations of features that you can use in DirectQuery mode, see the “Feature limitations in DirectQuery mode” section of the following page:



DirectQuery Mode (SSAS Tabular)

<http://aka.ms/M6m6uz>

Enabling DirectQuery Mode

To enable DirectQuery for a database in SQL Server Management Studio, configure the **Default Mode** setting on the **Model** page of database properties. To enable DirectQuery in SQL Server Data Tools, configure the **DirectQuery Mode** setting in the properties of the **Model.bim** file.

Security for Tabular Data Models

You can configure security in a tabular data model by creating database roles, and then applying permissions and filters to them. You should use roles to gather together Windows® user accounts and groups that have the same data access requirements, and then apply the permissions to the role. You can create and manage roles in a tabular data model in the development phase by using the **Role Manager** dialog box in SQL Server Data Tools. To access this dialog box, on the **Model** menu, click **Roles**.

- Create roles to group users according to security requirements
- Apply database-level permissions to roles:
 - Read
 - Read and Process
 - Process
 - Administrator
 - None
- Use DAX to create filters that define row-level security

Database Permissions

You can use Role Manager to define database permissions for roles. You can define permissions that allow five levels of access to a tabular data model database. They are:

- **Read.** This permission gives role members read access to the data, but they cannot see the model in SQL Server Management Studio.
- **Read and Process.** This permission gives role members read access to the database and allows them to process data, but they cannot see the model in SQL Server Management Studio.
- **Process.** This permission gives role members the ability to process data in the database.
- **Administrator.** This permission gives role members rights to read, process, and modify the data.
- **None.** This permission prevents role members from accessing the data.

You can also define database permissions for roles by using SQL Server Management Studio.

Row-Level Security

You can create DAX expressions that filter the rows that role members can see. DAX filters are expressions that evaluate to TRUE or FALSE. Rows that evaluate to TRUE are visible to role members. For example, the expression in the following code example filters the **Region** column in the **Sales Territory** table:

```
= 'Sales Territory' [Region] = "France"
```

The rows that contain the value **France** evaluate to TRUE, so they are visible to role members. Rows that contain other values in the **Region** column are not visible.

You can only define DAX filters for roles that have the **Read** and **Read and Process** database permissions. Roles that have **None** or **Process** permissions cannot view any data, so filters are irrelevant. Roles that have the **Administrator** permission can view all data. You can test the database permissions and row-level filters that you define by using the **Analyze in Excel** option in SQL Server Data Tools, and connecting as a specific role.

Deploying a Tabular Data Model

When you have completed the development phase for a tabular data model, you can deploy it to an instance of SQL Server Analysis Services that is running in tabular mode. This enables information workers and data analysts to connect to the data model by using client tools such as Excel and Power View. Deploying a tabular data model copies the data and metadata in the workspace database and uses it to create a new database that has the same name as the project by default. You can deploy to the same server that you used to develop the model, but, more often, you will deploy to a different instance of SQL Server Analysis Services, such as a test or production server.

- Copies data and metadata in workspace database to new database on SQL Server Analysis Services instance
- Enables connections from client tools
- Deployment options
- Deployment server
- BI semantic model connection

You can control this process by using the **Deployment Options** and **Deployment Server** settings in the project properties in SQL Server Data Tools. To access the project properties, in Solution Explorer, right-click the project, and then click **Properties**.

- There are two options available in the **Deployment Options** settings:
- **Processing Option.** Use this setting to define how the new database is processed. There are three options:
 - **Default.** When you select this option, SQL Server Analysis Services detects the state of the database and processes the data and metadata accordingly.
 - **Full.** When you select this option, SQL Server Analysis Services processes all data and metadata.
 - **Do Not Process.** When you select this option, SQL Server Analysis Services deploys only the metadata.
- **Transactional Deployment.** By default, SQL Server Analysis Services will deploy objects even if processing for those objects fails. You can use this option to force deployment to fail if processing fails.
- There are four options available in the **Deployment Server** settings:
- **Server.** Use this option to specify the name of the deployment server.
- **Edition.** Use this option to specify the edition of the instance of SQL Server Analysis Services on the deployment server.
- **Database.** Use this option to specify the name of the database on the deployment server.
- **Cube Name.** Use this option to specify the name that client tools see when they connect to the model.

BI Semantic Model Connection

Clients can connect directly to a tabular data model in SQL Server Analysis Services by using tools such as Excel. However, you can make the connection process easier for users by creating a BI semantic model connection in the PowerPivot Gallery. A BI semantic model connection enables users to connect to a tabular database in SQL Server Analysis Services or a PowerPivot workbook by using Excel or Power View with a single click—and without needing to create connections manually.

For further details about how to create a BI semantic model connection, see *Create a BI Semantic Model Connection to a Tabular Model Database* in MDSN:



Create a BI Semantic Model Connection to a Tabular Model Database

<http://aka.ms/jc2xy0>

Demonstration: Deploying a Tabular Data Model

In this demonstration, you will see how to:

- Deploy a tabular data model project.
- Use a tabular SQL Server Analysis Services database in Excel.

Demonstration Steps

Deploy a Tabular Data Model Project

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Data Tools, in Solution Explorer, right-click the **TabularDemo** project, and then click **Properties**.
3. In the **TabularDemo Property Pages** dialog box, type the following values, and then click **OK**:
 - **Server:** localhost\SQL2
 - **Database:** DemoDB
 - **Cube Name:** Internet Sales
4. On the **Build** menu, click **Deploy TabularDemo**. If you are prompted for impersonation credentials, type the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **OK**.
5. In the **Deploy** dialog box, when deployment has completed, click **Close**.
6. Close SQL Server Data Tools.

Use a Tabular SQL Server Analysis Services Database in Excel

1. Start Excel, and then create a new blank document.
2. On the **Data** tab, in **Get External Data** area, in the **From Other Sources** drop-down list, select **From Analysis Services**.
3. In the **Data Connection Wizard** dialog box, in the **Server name** box, type **MIA-SQL\SQL2**, ensure that **Use Windows Authentication** is selected, and then click **Next**.
4. On the **Select Database and Table** page, ensure that the **DemoDB** database and the **Internet Sales** cube are selected, and then click **Next**.
5. On the **Save Data Connection File and Finish** page, click **Finish**.
6. In the **Import Data** dialog box, ensure that the **Existing Worksheet** option is selected, and then click **OK**.
7. In the PivotTable Fields pane, under **Internet Sales**, select **Revenue**.
8. In the PivotTable Fields pane, under **Order Date**, select the **Calendar Date** hierarchy.
9. In the PivotTable Fields pane, under **Geography**, drag **EnglishCountryRegionName** to the **Columns** area.

10. Explore the data in the PivotTable. When you have finished, close Excel without saving the workbook.

Check Your Knowledge

Question	
Which of the following statements regarding DirectQuery mode is correct?	
Select the correct answer.	
<input type="checkbox"/>	A tabular data model that uses DirectQuery mode will always have a better performance than a tabular data model that uses in-memory storage.
<input type="checkbox"/>	A tabular data model that uses in-memory storage will always have a better performance than a tabular data model that uses DirectQuery mode.
<input type="checkbox"/>	A tabular data model that uses DirectQuery mode will always contain up-to-date information.
<input type="checkbox"/>	A tabular data model that uses DirectQuery mode will only support a SQL Server relational database data source.
<input type="checkbox"/>	You cannot use MDX in a tabular data model that uses DirectQuery mode.

Lab: Implementing a Tabular Data Model in SQL Server Analysis Services

Scenario

Adventure Works needs an enterprise data model for reseller sales analysis, and you have decided to implement it by using a tabular model.

Objectives

After completing this lab, you will be able to:

- Create a tabular data model project in SQL Server Analysis Services.
- Configure columns and relationships.
- Deploy a tabular data model project in SQL Server Analysis Services.

Estimated Time: 60 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Creating a Tabular Data Model Project in SQL Server Analysis Services

Scenario

Adventure Works Cycles has a data warehouse that contains fact and dimension tables for sales. You plan to use the tables that relate to reseller sales as the basis of your data model.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create a Tabular Project in SQL Server Analysis Services
3. Import Tables into the Data Model
4. Create Measures
5. Test the Model

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab07\Starter folder as Administrator.

► Task 2: Create a Tabular Project in SQL Server Analysis Services

1. Use SQL Server Data Tools to create a tabular project in SQL Server Analysis Services named **AWSalesTab** in the D:\Labfiles\Lab07\Starter folder.
2. Use the **localhost\SQL2** instance of SQL Server Analysis Services as the workspace server, and set the compatibility level of the project to **SQL Server 2016 RTM (1200)**.

► Task 3: Import Tables into the Data Model

1. Open the **Model.bim** model, and then, using the impersonation user name **ADVENTUREWORKS\ServiceAcct** and the impersonation password **Pa\$\$w0rd**, import the following tables from the **AdventureWorksDW** database on the **MIA-SQL** instance of SQL Server (use the friendly names in parentheses):
 - **DimDate** (Date)
 - **DimEmployee** (Employee)
 - **DimGeography** (Geography)
 - **DimProduct** (Product)
 - **DimProductCategory** (Product Category)
 - **DimProductSubcategory** (Product Subcategory)
 - **DimReseller** (Reseller)
 - **FactResellerSales** (Reseller Sales)
2. Filter the following tables to remove the columns listed below:
 - **DimDate**
 - SpanishDayNameOfWeek
 - FrenchDayNameOfWeek
 - DayNumberOfYear
 - WeekNumberOfYear
 - SpanishMonthName
 - FrenchMonthName
 - CalendarSemester
 - FiscalSemester
 - **DimEmployee**
 - SalesTerritoryKey
 - NameStyle
 - Title
 - HireDate
 - BirthDate
 - LoginID
 - EmailAddress
 - Phone
 - MaritalStatus
 - EmergencyContactName
 - EmergencyContactPhone
 - SalariedFlag
 - Gender

- PayFrequency
- Baserate
- VacationHours
- SickLeaveHours
- CurrentFlag
- SalesPersonFlag
- StartDate
- EndDate
- Status
- **DimGeography**
 - SpanishCountryRegionName
 - FrenchCountryRegionName
 - IpAddressLocator
- **DimProduct**
 - WeightUnitMeasureCode
 - SizeUnitMeasureCode
 - SpanishProductName
 - FrenchProductName
 - FinishedGoodsFlag
 - SafetyStockLevel
 - ReorderPoint
 - DaysToManufacture
 - ProductLine
 - DealerPrice
 - Class
 - Style
 - ModelName
 - FrenchDescription
 - ChineseDescription
 - ArabicDescription
 - HebrewDescription
 - ThaiDescription
 - GermanDescription
 - JapaneseDescription
 - TurkishDescription
 - StartDate

- EndDate
- Status
- **DimProductCategory**
 - SpanishProductCategoryName
 - FrenchProductCategoryName
- **DimProductSubcategory**
 - SpanishProductSubcategoryName
 - FrenchProductSubcategoryName
- **DimReseller**
 - OrderFrequency
 - OrderMonth
 - FirstOrderYear
 - LastOrderYear
 - ProductLine
 - AddressLine1
 - AddressLine2
 - AnnualSales
 - BankName
 - MinPaymentType
 - MinPaymentAmount
 - AnnualRevenue
 - YearOpened
- **FactResellerSales**
 - DueDateKey
 - PromotionKey
 - CurrencyKey
 - SalesTerritoryKey
 - RevisionNumber
 - CarrierTrackingNumber
 - CustomerPONumber
 - DueDate

► **Task 4: Create Measures**

1. In the model that has been created, edit the **Reseller Sales** table to add the following measures:
 - `Quantity:=Sum([OrderQuantity])`
 - `Cost:=Sum([TotalProductCost])`
 - `Revenue:=Sum([SalesAmount])`

2. Format the **Quantity** measure as a whole number that includes a thousand separator, and ensure that the **Cost** and **Revenue** measures are formatted as currency values.

► Task 5: Test the Model

1. Analyze the model that you have created in Excel.
2. Note the following problems with the model:
 - Aggregated measures from the **Reseller Sales** table are easily confused with the columns on which they are based.
 - Column names are not user-friendly.
 - Months in the **Date** table are sorted alphabetically, not chronologically.
 - The relationship between **Reseller Sales** and **Date** is ambiguous, because sales orders include an order date and a ship date.
3. You will fix these problems in the next exercise.

Results: After this exercise, you should have created a tabular data model project.

Exercise 2: Configuring Columns and Relationships

Scenario

You have created an initial tabular data model, but have identified some problems. The column names are not user-friendly, some of the relationships are ambiguous, and some date attributes are being sorted alphabetically rather than chronologically. You are now going to address these issues.

The main tasks for this exercise are as follows:

1. Configure Relationships
2. Rename and Hide Columns
3. Configure Column Sort Order
4. Create Hierarchies
5. Test the Model

► Task 1: Configure Relationships

1. View the existing relationships between the **Reseller Sales** and **Date** tables.
2. Make the active relationship the one based on the order date, delete all other relationships between these tables, and then rename the **Date** table to **Order Date**.
3. Open the existing connection to the **AdventureWorksDW** database, and then import the **DimDate** table again:
 - Use the **ADVENTUREWORKS\ServiceAcct** account with the password **Pa\$\$w0rd** if you are prompted for impersonation credentials.
 - Use the friendly name **Ship Date** for the **DimDate** table.
 - Filter the table to delete the following columns:
 - SpanishDayNameOfWeek
 - FrenchDayNameOfWeek
 - DayNumberOfYear

- WeekNumberOfYear
 - SpanishMonthName
 - FrenchMonthName
 - CalendarSemester
 - FiscalSemester
4. Create a relationship between the **Reseller Sales** and **Ship Date** tables based on the **ShipDateKey** and **DateKey** columns.

► **Task 2: Rename and Hide Columns**

1. In the **Geography** table, hide the **GeographyKey** and **SalesTerritoryKey** columns, and then rename the columns as shown in the following table:

Column	New name
StateProvinceCode	State or Province Code
StateProvinceName	State or Province
CountryRegionCode	Country or Region Code
EnglishCountryRegionName	Country or Region
PostalCode	Postal Code

2. In the **Reseller** table, hide the **ResellerKey**, **GeographyKey**, and **ResellerAlternateKey** columns, and then rename the columns as shown in the following table:

Column	New name
BusinessType	Business Type
ResellerName	Reseller Name
NumberEmployees	Employees

3. In the **Employee** table, hide the **EmployeeKey**, **ParentEmployeeKey**, **EmployeeNationalIDAlternateKey**, and **ParentEmployeeNationalIDAlternateKey** columns, and then rename the columns as shown in the following table:

Column	New name
FirstName	First Name
LastName	Last Name
MiddleName	Middle Name
DepartmentName	Department

4. In the **Order Date** table, hide the **DateKey**, **DayNumberOfWeek**, and **MonthNumberOfYear** columns, and then rename the columns as shown in the following table:

Column	New name
FullDateAlternateKey	Date
EnglishDayNameOfWeek	Weekday
DayNumberOfMonth	Day of Month
EnglishMonthName	Month
CalendarQuarter	Calendar Quarter
CalendarYear	Calendar Year
FiscalQuarter	Fiscal Quarter
FiscalYear	Fiscal Year

5. In the **Ship Date** table, hide the **DateKey**, **DayNumberOfWeek**, and **MonthNumberOfYear** columns, and then rename the columns as shown in the following table:

Column	New name
FullDateAlternateKey	Date
EnglishDayNameOfWeek	Weekday
DayNumberOfMonth	Day of Month
EnglishMonthName	Month
CalendarQuarter	Calendar Quarter
CalendarYear	Calendar Year
FiscalQuarter	Fiscal Quarter
FiscalYear	Fiscal Year

6. In the **Product** table, hide the **ProductKey**, **ProductAlternateKey**, and **ProductSubcategoryKey** columns, and then rename the columns as shown in the following table:

Column	New name
EnglishProductName	Product Name
StandardCost	Standard Cost
ListPrice	List Price
SizeRange	Size Range
EnglishDescription	Description

7. In the **Product Subcategory** table, hide the **ProductSubcategoryKey**, **ProductSubcategoryAlternateKey**, and **ProductCategoryKey** columns, and then rename the **EnglishProductSubcategoryName** column to **Subcategory**.
8. In the **Product Category** table, hide the **ProductCategoryKey** and **ProductCategoryAlternateKey** columns, and then rename the **EnglishProductCategoryName** column to **Category**.
9. In the **Reseller Sales** table, hide all columns other than the **Quantity**, **Cost**, and **Revenue** measures that you created in the previous exercise.

► Task 3: Configure Column Sort Order

- In the **Ship Date** table, configure the sort orders of the following columns:
 - The **Weekday** column should be sorted by the **DayNumberOfWeek** column.
 - The **Month** column should be sorted by the **MonthNumberOfYear** column.
- In the **Order Date** table, configure the sort orders of the following columns:
 - The **Weekday** column should be sorted by the **DayNumberOfWeek** column.
 - The **Month** column should be sorted by the **MonthNumberOfYear** column.

► Task 4: Create Hierarchies

- In the **Geography** table, create a hierarchy named **Location** that includes the following columns:
 - Country or Region
 - State or Province
 - City
 - Postal Code
- In the **Order Date** table, create a hierarchy named **Calendar Date** that contains the following fields:
 - Calendar Year
 - Calendar Quarter
 - Month
 - Day of Month

3. In the **Order Date** table, create a second hierarchy named **Fiscal Date** that contains the following fields:
 - Fiscal Year
 - Fiscal Quarter
 - Month
 - Day of Month

► **Task 5: Test the Model**

1. View the data model in Excel, and then verify that the changes that you have made are reflected in the PivotTable.
2. When you have finished, close Excel without saving the workbook.

Results: After completing this exercise, you should have a tabular data model that includes renamed columns, custom sort orders, and specific relationships between tables.

Exercise 3: Deploying a Tabular Data Model Project in SQL Server Analysis Services

Scenario

You have configured the columns, relationships, and hierarchies of your tabular data model. Now you must improve the processing performance by separating the more recent data and the older data into partitions. You must then set up a perspective for an analysis team who are interested in breaking down sales only by product dimensions and not any other dimensions. Next, you must set up a security role for users who must only have access to data that relates to bike sales in the United States. Finally, you must deploy the tabular model to the server so that users can connect to it by using client tools.

The main tasks for this exercise are as follows:

1. Separate Data into Partitions
2. Create a Perspective
3. Create a Security Role
4. Deploy the Reseller Sales Project
5. Use the Deployed Tabular Database

► **Task 1: Separate Data into Partitions**

1. Create a new partition in the **Reseller Sales** table.
2. Rename the two partitions in the **Reseller Sales** table to **Before 2008** and **Since 2008**.
3. Set the **Before 2008** partition to only show records in the **Reseller Sales** table that have an order date before 2008.
4. Set the **Since 2008** partition to only show records in the **Reseller Sales** table that have an order date that is greater than or equal to January 1, 2008.
5. Use SQL Server Data Tools to process both partitions, using **Process Full** mode.

► Task 2: Create a Perspective

1. Create a perspective called **Products Analysis** that only shows the **Reseller Sales** table, the **Product Category** table, the **Product Subcategory** table, and the **Products** table.
2. View the data model in Excel by using the **Products Analysis** perspective. Note that only the tables that you specified for the perspective are visible.
3. When you have finished, close Excel without saving the workbook.

► Task 3: Create a Security Role

1. Create a security role called **US Bike Sales**, with **Read and Process** permission.
2. Set a row filter, so that the new role only shows reseller sales where the value of the **Category** column in the **Product Category** table is **Bikes**, and the value of the **Country or Region Code** column in the **Geography** table is **US**.
3. View the data model in Excel by using the **US Bike Sales** role. Note that the reseller data is only showing where the location is the United States and product category is Bikes.
4. When you have finished, close Excel without saving the workbook.

► Task 4: Deploy the Reseller Sales Project

1. Configure the project for deployment:
 - The project should be deployed to the **localhost\SQL2** instance of SQL Server Analysis Services.
 - The deployed database should be named **AdventureWorksTab**.
 - The deployed cube should be named **Reseller Sales**.
2. Build and deploy the project, and then close SQL Server Data Tools.

► Task 5: Use the Deployed Tabular Database

1. Create a blank workbook in Excel, and then import the data model from SQL Server Analysis Services:
 - Use the **From Analysis Services** source on the **Data** tab of the ribbon.
 - Connect to the **MIA-SQL\SQL2** instance of SQL Server Analysis Services by using Windows authentication.
 - Select the **Reseller Sales** cube in the **AdventureWorksTab** database.
 - Import the data into a PivotTable in the workbook.
2. Explore the data in the PivotTable. When you have finished, close Excel without saving the workbook.

Results: After this exercise, you should have deployed the tabular data model project.

Module Review and Takeaways

In this module, you have learned how to develop and deploy a tabular data model for SQL Server Analysis Services by using SQL Server Data Tools.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 8

Introduction to Data Analysis Expressions (DAX)

Contents:

Module Overview	8-1
Lesson 1: DAX Fundamentals	8-2
Lesson 2: Using DAX to Create Calculated Columns and Measures in a Tabular Data Model	8-12
Lab: Using DAX to Enhance a Tabular Data Model	8-24
Module Review and Takeaways	8-30

Module Overview

You can extend Microsoft® SQL Server® Analysis Services tabular data models by using the Data Analysis Expressions (DAX) language. DAX is a highly flexible language that enables you to create measures and calculated columns for use in PivotTable tables and PivotChart charts.

Objectives

This module explains the fundamentals of the DAX language. It also explains how you can use DAX to create calculated columns and measures, and how you can use them in your tabular data models.

After completing this module, you will be able to:

- Describe the fundamentals of DAX.
- Use DAX to create calculated columns and measures.

Lesson 1

DAX Fundamentals

To use DAX effectively, you should understand the components of the language and its capabilities. This lesson explains the fundamentals of DAX and provides an overview of its functionality.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe DAX and how it is used.
- Describe the different types of DAX functions.
- Describe the syntax for writing DAX formulas.
- Describe aggregations.
- Explain why context is important when writing DAX formulas.
- Describe the basics of DAX queries.

Overview of DAX

The DAX language consists of a library of functions, operators, and constants—you can use these to create formulas that define the calculated columns and measures to extend the functionality of tabular data models. DAX was introduced in the first version of the PowerPivot for Excel® add-in, as the language for writing the formulas that define business logic in PowerPivot for Excel workbooks. In SQL Server 2016, you can use DAX in tabular data models in SQL Server Analysis Services, in addition to those you create in PowerPivot for Excel workbooks, and in Power BI Desktop.

- Formula-based language for building business logic and queries in tabular data models:
 - Create columns and measures based on complex custom calculations
 - Create columns that reference data in related tables
 - Define parent-child hierarchies
 - Create Key Performance Indicators (KPIs)
- Syntax is similar to Microsoft Excel formulas

Typical Uses of DAX

You can use DAX to extend tabular data models beyond the basic aggregation of numerical columns, based on tables and hierarchies. With DAX, you can:

- Create columns and measures based on complex custom calculations.
- Create columns that reference data in related tables.
- Define parent-child hierarchies.
- Create Key Performance Indicators (KPIs).

DAX Syntax

DAX is based on the Microsoft Excel formula syntax and uses a range of functions, operators, and values. However, unlike formulas in Excel, DAX does not work on individual cells or ranges of cells. Instead, you use it to work with the relational columns and tables that you import into a tabular data model. DAX also includes more advanced functionality provided by Excel formulas. The similarity between DAX and Excel formulas minimizes the learning time for information workers who want to learn to use DAX, enabling faster adoption of the language and reducing the need for training.

For links to the DAX documentation, see the topic *Data Analysis Expressions (DAX) Reference* in the SQL Server 2016 Technical Documentation:

 **Data Analysis Expressions (DAX) Reference**

<http://aka.ms/Sxj8h6>

DAX Functions

DAX provides a range of functions you can use to build formulas. Many of these functions are the same as or similar to those used to build formulas in Excel. DAX, however, includes many additional functions that help you to create much more complex and sophisticated reports than you can by using only Excel functions. For example, DAX includes functions that can navigate table relationships, take account of context to calculate values in PivotTable tables, or compare time periods to produce the desired result. A DAX formula can contain multiple functions that can be nested within other functions (up to 64 levels deep).

- Text functions
- Information functions
- Filter and value functions
- Logical functions
- Mathematical and trigonometric functions
- Statistical and aggregation functions
- Date and time functions
- Time intelligence functions
- Navigation and Lookup functions

 **Note:** You should avoid having too many levels of nesting in your formulas because it makes them difficult to understand and troubleshoot.

DAX includes several categories of functions, including:

- **Text.** These functions include CONCATENATE, TRIM, and BLANK. They are based on the Excel string functions, and enable you to manipulate strings.
- **Information.** These functions include PATH, ISBLANK, and ISERROR. They enable you to test row values to see whether they match a type of value, such as an error or a null, and to handle relationships in a table. This category also includes functions such as USERNAME, which you can use to implement row-level security in SQL Server Analysis Services tabular data models.
- **Filter and value.** These functions include CALCULATE, ALL, and FILTER. You can use them to manipulate data in a variety of ways, such as modifying the context for calculations, or using the relationships between tables to return specific results. There is no direct equivalent of these functions in Excel.
- **Logical.** These functions include TRUE, IF, SWITCH, and NOT. With these functions, you can perform logical tests on expressions to return information about the state of the tested values, and to use conditional logic.
- **Mathematical and trigonometric.** These functions include ROUND, FLOOR, and POWER. They are similar to the mathematical and trigonometric functions in Excel.
- **Statistical and aggregation.** These functions include SUM, COUNT, RANKX, and DISTINCTCOUNT. They are similar to aggregation functions in Excel.
- **Date and time.** These functions include DATE, MONTH, and YEAR. They are equivalent to date and time functions in Excel.

- **Time intelligence.** These functions include PARALLELPERIOD, TOTALYTD, and OPENINGBALANCEYEAR. You can use these functions to work with columns using the Date data type in a more sophisticated way; for example, by comparing specific business measures across time periods.
- **Navigation and Lookup.** DAX is designed to work with multiple tables and columns in a tabular data model. It provides functions such as RELATEDTABLE, USERELATIONSHIP, CROSSJOIN, and LOOKUPVALUE, which you can use to combine data from multiple tables. You can also use functions such as PATH and PATHITEM to define navigable hierarchies of column values in a single table.

For complete documentation about the functions supported by DAX, see the topic *DAX Function Reference* in the SQL Server 2016 Technical Documentation:



DAX Function Reference

<http://aka.ms/wqmicc>

DAX Syntax and Data Types

DAX is syntactically similar to the Excel formula language, so users already skilled in this area should be able to start creating DAX formulas quickly. The following list summarizes basic DAX syntax and conventions:

- All DAX formulas must start with an equal sign (=), followed by an expression.
- Expressions can contain functions, operators, constants, and references to columns in tables. Object names that are referenced in expressions, such as table and column names, are case insensitive and can include spaces. You cannot use leading or trailing spaces, characters invalid in the names of SQL Server Analysis Services objects, or control characters in table, column, or measure names. Arithmetic, comparison, text concatenation, and logical operators are largely the same as for formulas in Excel, and mostly work in the same way.
- DAX functions use tables and columns as inputs. To avoid ambiguity, you should use fully qualified names when referencing columns. A fully qualified name uniquely identifies a column because it includes the name of the table to which the column belongs. Use brackets around the column names and single straight quotation marks around table names that contain spaces; for example, **'Order Date'[Calendar Year]**. You can use unqualified names to refer to columns in the same table. For example, when used in the context of the **Order Date** table, the **'Order Date'[Calendar Year]** column can be referenced as **[Calendar Year]**.
- When you are writing and referencing one measure to another, you should enclose the measure name in brackets; for example, **[Total Profit]**.

- **DAX Syntax**
 - DAX formulas start with the equal sign (=) followed by an expression
 - Expressions can contain functions, operators, constants, and references to columns
 - Column references:
 - **'table name'[column name]** – a fully qualified name
 - **[column name]** – an unqualified name
 - Measure names must be enclosed in brackets
 - Table names containing spaces must be enclosed in single quotation marks

Although DAX formulas are similar to Excel formulas, there are two important differences:

- Excel formulas can reference ranges of cells or individual cells. DAX formulas can only act on complete tables or columns of data. To work with a subset of rows in a table or column, you must use a filter function to restrict the data.
- The data types supported by DAX formulas are not identical to the data types recognized by Excel formulas.

For more information about DAX syntax, see the topic *DAX Syntax Reference* in the SQL Server 2016 Technical Documentation:



DAX Syntax Reference

<http://aka.ms/Pujroo>

Operators

DAX operators are very similar to operators in Excel formulas. The following operators are supported by DAX:

- Arithmetic:
 - +. Addition
 - -. Subtraction
 - *. Multiplication
 - /. Division
 - ^. Exponentiation
- Comparison:
 - =. Equal
 - <. Less than
 - >. Greater than
 - <=. Less than or equal to
 - >=. Greater than or equal to
 - <>. Not equal
- Text concatenation:
 - &. Concatenate text
- Logical:
 - &&. Logical AND
 - ||. Logical OR

For more information about DAX operators, see the topic *DAX Operator Reference* in the SQL Server 2016 Technical Documentation:



DAX Operator Reference

<http://aka.ms/Veqptn>

Data Types

DAX will implicitly convert data types where possible, so there is no need to explicitly convert a column to a different data type. For example, if you enter a date as a string and use that value in a date function, DAX will attempt to convert the data type to Date. There are no DAX functions for explicitly converting data types.

DAX uses the following data types:

- I8 (eight-byte integer).
- R8 (eight-byte real number).

- Boolean. TRUE or FALSE.
- String.
- Date/time.
- Currency.
- BLANK. Represents a blank or missing value or an empty cell in an Excel worksheet. You can test for blank values by using the ISBLANK function.
- Table. Used by functions that either require a table as input or return a table.



Note: BLANK and NULL are not the same thing. Handling of BLANK data types varies, depending upon the operation that you perform. For example, $\text{BLANK} + 5 = 5$, but $\text{BLANK} * 5 = \text{BLANK}$. However, in the SQL Server Database Engine, both $\text{NULL} + 5$ and $\text{NULL} * 5$ result in a NULL value. Database Engine NULL values are implicitly converted to BLANK values when you import data into a tabular data model.

Aggregations

Information workers are typically interested in finding answers to questions such as: "How much revenue did the company generate through its Internet sales channel in April?" or: "What was the best-selling product in France last year?" To answer these kinds of questions, you must group and aggregate data. You group data by using values that give meaning and context, such as Product Category or Sales Region. These values are usually stored in the tables you import from data sources. You can aggregate data by using DAX aggregation functions. The following are commonly used aggregation functions:

- Summarize underlying detailed data for analysis
- Use automatic aggregation for simple calculations:
 - SUM
 - COUNT
 - MIN
 - MAX
 - AVERAGE
- Create a measure for more complex aggregations

SUM

Adds up the values in a numeric column, providing a single total for that column.

For example, the following code totals all the values in the **Sales Amount** column in the **Reseller Sales** table:

DAX SUM

```
=SUM('Reseller Sales'[Sales Amount])
```

SUMX

Filters and then adds up the values in a numeric column, returning a single total for that filtered column.

For example, the following code returns the total of the **Freight** column in the **Internet Sales** table, but the calculation only includes rows for which the **SalesTerritoryKey** value is 5:

DAX SUMX

```
=SUMX(FILTER('Internet Sales', 'Internet Sales'[SalesTerritoryKey]=5),[Freight])
```

COUNT

Returns a count of the number of rows that are not blank for columns containing either numbers or dates. COUNTA is similar, but returns a count of the number of rows that are not blank for columns containing non-numeric values.

For example, the following code returns a count of the rows in the **Order Date** column in the **Reseller Sales** table:

DAX COUNT

```
=COUNT('Reseller Sales'[Order Date])
```

COUNTX

Filters and then counts the number of values that are not blank in a column and returns a single count for that filtered column. The COUNTX function works in the same way as COUNTA, but only counts numeric values. The COUNTBLANK function returns the number of blank values in a column, and the COUNTROWS function counts the number of rows in a table.

For example, the following code returns the number of rows in the **Phone** column in the **Reseller** table, where the value in the **Status** column is set to **"Active"**:

DAX COUNTX

```
=COUNTX(FILTER('Reseller', [Status]="Active"), [Phone])
```

MIN and MAX

Return the minimum and maximum values in a numeric column. The MAXA and MINA functions return the maximum and minimum numeric values in a column, but they also handle non-numeric logical values, such as TRUE and FALSE, which evaluate to 1 and 0 respectively.

The following code example returns the maximum value for the Sales Amount column in the Reseller Sales table:

DAX MAX

```
=MAX('Reseller Sales'[Sales Amount])
```

MAXX and MINX

Return the maximum and minimum values in a column in a table filtered by using an expression.

For example, the following code returns the minimum value for the **Freight** column in the Internet **Sales** table for which the corresponding **SalesTerritoryKey** value is **5**:

DAX MINX

```
=MINX(FILTER('Internet Sales', [SalesTerritoryKey] = 5), [Freight])
```

AVERAGE

Returns the mean of all values in a numeric column. The AVERAGEA function does the same thing as AVERAGE, but also handles non-numeric data types.

For example, the following code returns the average value in the **Total Product Cost** column in the **Reseller Sales** table:

DAX AVERAGE

```
=AVERAGE('Reseller Sales'[Total Product Cost])
```

AVERAGEX

Enables you to supply an expression to obtain the average of values that are not neatly contained in a single column.

For example, the following code sums the **Freight** and **TaxAmt** values for each row in the **Internet Sales** table, and then calculates the average value:

DAX AVERAGEX

```
=AVERAGEX('Internet Sales', 'Internet Sales'[Freight]+ 'Internet Sales'[TaxAmt])
```

Using Aggregations

When you add a column to the **Values** area in the **PowerPivot Field List** in a PowerPivot for Excel worksheet, column values are automatically summed and the total added to the PivotTable table or PivotChart chart that you are using. If you need to summarize the data differently, you can select another type of aggregation, such as COUNT, MIN, or AVERAGE, from a list. The DAX formula is then automatically created for you. When you need to create aggregations more complex than this, you can create a measure.

For a complete list of statistical and aggregate functions in DAX, see the topic *Statistical Functions (DAX)* in the SQL Server 2016 Technical Documentation:



Statistical Functions (DAX)

<http://aka.ms/Ltbpp9>

Context

When you create a DAX measure or calculated column, you define a field you can use in a PivotTable or a PivotChart. However, the specific values appearing in any given PivotTable or PivotChart will depend on the context, such as the columns and rows added to a PivotTable or a PivotChart, or the slicers that are enabled at the time. For example, if you create a PivotTable and add only the **Sum of Sales Amount** measure to the **Values** area and nothing to the **Row Labels** area, you will see a single value representing the total sales amount for all categories, years, and countries in the **Reseller Sales** channel. If you then add **Calendar Year** to the **Row Labels** area, you will see multiple values, each one a total for a given year. Even though the same measure is used in both cases, the displayed value changes to reflect the context. This context-aware behavior makes it possible to perform dynamic analysis of data quickly and easily, without needing to reconfigure PivotTables or PivotCharts whenever you add a new measure.

It is important to be aware of context when you create measures and calculated columns. If you do not take context into account, you might find your reports do not aggregate data in the way you intended, and are inaccurate and misleading.

Every DAX expression has two contexts; a row context, and a filter context.

- A DAX measure or calculated column defines a field that you can use in a PivotTable table or a PivotChart chart
- The exact values that appear in PivotTable tables and PivotChart charts vary with context:
 - Row context
 - Filter context
 - Query context

Row Context

The row context of a DAX expression corresponds to a single row in the source data. Row context is used when you use functions that operate on two or more columns in the same row.

For example, the formula in the following code references the **First Name** and **Last Name** columns in the **Employee** table:

DAX Row Context

```
=CONCATENATE([First Name], CONCATENATE(" ", [Last Name]))
```

The calculated column evaluates the expression for each row. It uses the row context to obtain the values in the **First Name** and **Last Name** columns for each row, and then uses these values to create a new value for each row by concatenating them. Row context is important when using any function that evaluates an expression for each row in a table, such as SUMX.

Filter Context

The filter context of a DAX expression is made up of all the filters that are applied to the data when the expression is evaluated. Filter context is most obviously seen in PivotTables; when you add columns to the Rows or Columns area of a PivotTable, aggregated columns in the Values are split according to the row and column values. The row and column values define the filter context for each value cell. PivotTable slicers also affect the filter context by excluding some values from the PivotTable.

Filter context is important in DAX because you can use DAX functions to modify the row and query contexts by creating filters in formulas. You can use the FILTER function to produce a subset of rows on which you can perform a calculation. For example, you might filter by specifying a product category, then aggregate the sales amount for that category only. You can override filters by using the ALL and ALLEXCEPT functions. ALL forces all filters to be ignored, and ALLEXCEPT helps you to selectively override them by specifying particular columns to retain in the filter context—all other columns are ignored.

Query Context

When you add a measure to a PivotTable, the xVelocity engine evaluates the measure value for each cell. The evaluation takes account of the query context to calculate these values for a given cell. Query context is determined by the row labels, column labels, filters, and slicers that apply to the PivotTable table. Removing or adding a new slicer, row label, or column label, will change the filter context; the PowerPivot engine must take account of the new context that is performing the calculation for each row in the query.

For more information about context, see the section *Context in DAX Formulas* in the topic *Understanding DAX in Tabular Models (SSAS Tabular)* in the SQL Server 2016 Technical Documentation:

 **Understanding DAX in Tabular Models (SSAS Tabular) - Context in DAX Formulas**

http://aka.ms/Jx680b#Anchor_6

DAX Queries

DAX includes extensions that help you to write queries that retrieve data. Client reporting tools, such as Power View, use DAX to query tabular data models in SQL Server Analysis Services. It is unlikely that you will need to write DAX queries manually, but a basic understanding can help with troubleshooting.

You can retrieve data from tables by using the EVALUATE keyword, which is functionally equivalent to the SELECT statement in SQL Server.

The following code example returns all the rows from the **Reseller Sales** table:

Simple EVALUATE

```
EVALUATE('Reseller Sales')
```

Queries can include FILTER to return a reduced result set, similar to the way in which a WHERE clause returns a reduced result set in SQL Server.

The following code example returns all the rows from the **Reseller Sales** table where the **OrderDateKey** value is greater than 20040101:

FILTER and EVALUATE

```
EVALUATE(FILTER('Reseller Sales', 'Reseller Sales' [OrderDateKey]>20040101))
```

You can also order result sets.

The following code example returns all the rows from the **Reseller Sales** table where the **OrderDateKey** value is greater than 20040101. The results are ordered by **Sales Amount**:

FILTER and EVALUATE with ORDER BY

```
EVALUATE(FILTER('Reseller Sales', 'Reseller Sales' [OrderDateKey]>20040101))
ORDER BY 'Reseller Sales' [Sales Amount] ASC
```

For more information about DAX queries, see the topic *DAX Queries* in the SQL Server 2016 Technical Documentation:



DAX Queries

<http://aka.ms/Rnt1dc>

- Client applications, such as the Power View reporting tool, issue DAX queries
- You can write queries manually by using the DAX query language
- You can filter, order, and summarize results

```
EVALUATE(FILTER('Reseller Sales', 'Reseller Sales'
[OrderDateKey]>20040101))
```

Check Your Knowledge

Question	
When a DAX expression is evaluated, which contexts apply?	
Select the correct answer.	
<input type="checkbox"/>	The row context.
<input type="checkbox"/>	The filter context.
<input type="checkbox"/>	Both the row context and the filter context.

Lesson 2

Using DAX to Create Calculated Columns and Measures in a Tabular Data Model

DAX creates calculated columns and measures that you can then use in PivotTable tables and PivotChart charts in PowerPivot for Excel workbooks. This lesson shows you how to create DAX calculated columns and measures using a variety of functions, including time intelligence functions and functions that manipulate table relationships. This lesson also shows you how to create KPIs with DAX.

Lesson Objectives

At the end of this lesson, you will be able to:

- Use calculated columns.
- Use measures.
- Use DAX KPIs.
- Work with relationships between tables.
- Understand time intelligence functions.
- Work with hierarchical data.

Calculated Columns

A calculated column is a named column that you populate by using a DAX formula. The formula will usually refer to other columns, either in the same table or in a different, related table, but you can also create calculated columns based on measures, or on other calculated columns.

The following code example creates a column that calculates profit by subtracting the value in the **Total Product Cost** column from the value in the **Sales Amount** column. You could use this DAX expression to create a calculated column named **Sales Profit**.

A DAX Expression for a Calculated Column

```
=[Sales Amount] - [Total Product Cost]
```

You can create calculated columns in a tabular data model just as you can in an Excel workbook. In the **Data View** window of a data model designer, select the table to which you want to add the column, and then click **Add Column**. You can then name the column and type the DAX formula in the formula bar, which will determine the contents of the column. The xVelocity engine performs the calculation immediately to populate the column. By default, the engine recalculates columns automatically if the source data changes.



Note: The formula used to create a calculated column applies to every row. You cannot apply the formula only to selected data ranges, as you can in Excel worksheets.

- Named columns that are populated by using a DAX formula
- A value is calculated for each row in the table when the calculated column is created

When creating calculated columns, be aware of the following considerations:

- Calculated columns calculate a value for every row in a table. This can take a long time when tables contain a large number of rows.
- Changing the names of columns referenced by a calculated column will cause an error.

For more information about DAX calculated columns, see the topic *Calculated Columns (SSAS Tabular)* in the SQL Server 2016 Technical Documentation:



Calculated Columns (SSAS Tabular)

<http://aka.ms/She68f>

Measures

A measure is a named formula that can encapsulate complex business logic, usually by aggregating one or more numerical values.

The following example defines a measure named **Profit** that calculates the sum of the **Sales Profit** column:

A DAX Expression for a Measure

```
Profit:=SUM([Sales Profit])
```

Measures are defined in the Measures Grid, in Data View, of the data model designer. In client tools, measures are associated with the table in which they have been defined. If you need to create a measure that is not associated with an existing table, you can paste an Excel worksheet containing only a placeholder column header value into the data model as a new, empty table. You can then define measures in the measure grid for the empty table.

For more information about DAX measures, see the topic *Measures (SSAS Tabular)* in the SQL Server 2016 Technical Documentation:



Measures (SSAS Tabular)

<http://aka.ms/K55jub>

- Named formulas that can contain sophisticated business logic
- Usually aggregating numerical values
- Defined in the Measure Grid of a table
- Client tools associate measures with the tables in which they are defined

KPIs

KPIs are used to compare measures with a given goal or target value to assess business performance. For example, a business might set a profit margin goal of 15 percent, and use a KPI to indicate how actual sales results compare.

In tabular data models, KPIs are defined based on measures, and can consist of the following elements:

- **Base Value:** the actual value being assessed, calculated by a measure.

- Used to compare measures against target values
- Definition includes:
 - Base Value: the actual measure result
 - Target Value: the goal for the measure
 - Status Thresholds: indicators to show how the base value compares to the target value

- **Target Value:** the goal against which the base value is compared. This can be an absolute value or calculated by a measure.
- **Status Thresholds:** the status is an indicator of the base value's performance against the target value. You can set thresholds that establish the status, based on the percentage of the target value achieved by the base value. For example, you could use a red indicator if the base value is below 20 percent of the target value; a yellow indicator if it is between 20 and 80 percent; and a green indicator if it is above 80 percent.

For more information about working with DAX KPIs, see the topic *KPIs (SSAS Tabular)* in the SQL Server 2016 Technical Documentation:



KPIs (SSAS Tabular)

<http://aka.ms/Cle5ry>

Demonstration: Creating and Using Measures and Calculated Columns

In this demonstration, you will see how to:

- Create calculated columns.
- Create measures.
- Create a KPI.
- Analyze calculated columns and measures.

Demonstration Steps

Create Calculated Columns

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and log on to the 20768B-MIA-SQL virtual machine as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. Then, in the D:\Demofiles\Mod08 folder, right-click **Setup.cmd** and click **Run as administrator**. When prompted, click **Yes**.
2. Start Visual Studio and open the **TabularDemo.sln** solution in the D:\Demofiles\Mod08 folder. If the **Tabular model designer** dialog box is displayed, in the **Workspace server** list, specify **localhost\SQL2** and then click **OK**. Then in Solution Explorer, double-click **Model.bim** to open the data model designer. If you are prompted to run a script on the server, click **Yes**.
3. On the **Model** menu, point to **Process**, and click **Process All**. If you are prompted for impersonation credentials, specify the user name **ADVENTUREWORKS\ServiceAcct** with the password **Pa\$\$w0rd** and click **OK**. When all of the tables have been processed, click **Close**.
4. On the **Customer** tab, in the first empty column after the existing columns, double-click **Add Column** and name the new column **Full Name**.
5. With the **Full Name** column selected, in the formula bar, enter the following DAX expression:

```
=CONCATENATE([FirstName], " " & [LastName])
```

Wait for the table to finish updating, and note the calculated values in the new column.

6. Click the **FirstName** column heading, hold **Shift** and click the **LastName** column heading, and then right-click either of the selected column headings and click **Hide from Client Tools**.
7. On the **Internet Sales** tab, in the first empty column after the existing columns, double-click **Add Column** and name the new column **SalesProfit**.

- With the **SalesProfit** column selected, in the formula bar, enter the following DAX expression:

```
=[SalesAmount]-[TotalProductCost]
```

Wait for the table to finish updating, and note the calculated values in the new column.

Create Measures

- On the **Internet Sales** tab, click the first empty cell in the measure grid under the **SalesProfit** column.
- In the formula bar, enter the following DAX expression:

```
Profit:=SUM([SalesProfit])
```

- Select the cell containing the Profit measure, and press F4 to display the properties pane. Then set the **Format** property to **Currency**.
- Click the empty cell in the measure grid under the **Profit** measure.
- In the formula bar, enter the following DAX expression:

```
Margin:=[Profit]/[Revenue]
```

- Select the cell containing the **Margin** measure, and press F4 to display the properties pane. Then set the **Format** property to **Percentage**.
- Right-click the **SalesProfit** column header and click **Hide from Client Tools**.
- On the **File** menu, click **Save All**.

Create a KPI

- On the **Internet Sales** tab, click the empty cell in the measure grid under the **Margin** measure.
- In the formula bar, enter the following DAX expression:

```
Target Margin:=(SUM('Product'[ListPrice]) - SUM('Product'[StandardCost])) / SUM('Product'[ListPrice])
```

- Select the cell containing the **Margin** measure, and press F4 to display the properties pane. Then set the **Format** property to **Percentage**.
- On the **Internet Sales** tab, right-click the cell in the measure grid containing the **Margin** measure and click **Create KPI**.
- In the **Key Performance Indicator (KPI)** dialog box, note that the **KPI base measure (value)** is defined by the **Margin** measure. Then, under **Define target value**, ensure that **Measure** is selected and select **Target Margin**.
- Set the first status threshold to **65%** and the second to **90%**.
- Note the default icon style, and click **OK**.
- On the **File** menu, click **Save All**.

Analyze Columns and Measures

- On the **Model** menu, click **Analyze in Excel**.
- In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.

3. When **Excel** opens, in the **Internet Sales** table, select the **Profit** and **Margin** measures so that they are summarized in the **PivotTable**.
4. In the **Customer** table, select **Full Name** so that the PivotTable shows profit and margin by customer.
5. In the PivotTable Fields pane, expand **KPIs**, expand **Margin**, and select **Status** so that an indicator shows the KPI status for each customer.
6. Close Excel without saving the workbook.
7. Keep Visual Studio open for the next demonstration.

Multiple Relationships

Most data models consist of multiple tables that are related, based on common key values. There are many cases where a DAX expression needs to combine data from related tables, and DAX provides some functions to support this requirement.

Looking Up Related Data Values

DAX provides a LOOKUPVALUE function you can use to return a column value from a table, based on a specified value for another column.

For example, if a **Customer** table contains a **Registration Date** column, and a **Date** table contains **Full Date** and **Fiscal Year** columns, you could use the following expression in a calculated column in the **Customer** table to look up the fiscal year in which the customer registered:

DAX LOOKUPVALUE

```
LOOKUPVALUE('Date'[Fiscal Year], [Full Date], [Registration Date])
```

The LOOKUPVALUE does not require an existing relationship between tables; it simply looks up a column value based on the specified value for another column in the same table.

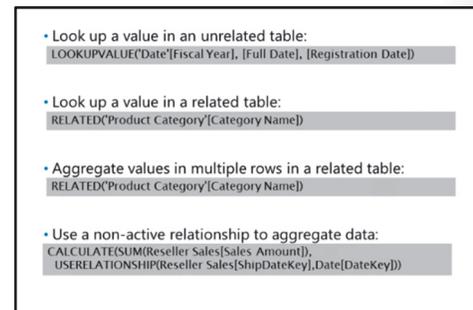
If a relationship is defined between two tables, a more effective way to look up related values is to use the RELATED function.

For example, if a **Product** table is related to a **Product Category** table, based on a common **CategoryKey** column, you could use the following expression to return the **Category Name** value from the **Category** table into a calculated column in the **Product** table:

DAX RELATED

```
RELATED('Product Category'[Category Name])
```

Note that the RELATED function does not require that you specify a lookup value. It automatically uses the active relationship between the tables to find the related value. The RELATED function works well when you need to look up a single value from the “many” side of a one-to-many relationship. In the example above, each category exists once in the **Product Category** table but many times in the **Product** table.



For more information about the RELATED function, see the topic *RELATED Function (DAX)* in the SQL Server 2016 Technical Documentation:

 **RELATED Function (DAX)**

<http://aka.ms/Vram4o>

Occasionally, you might want to aggregate many related values from the “one” side of the relationship.

For example, in the Product Category table, you might want to create a calculated column named **Product Count** containing the total number of products in each category. To accomplish this task, you can use the RELATEDTABLE function shown in the following example:

DAX RELATEDTABLE

```
COUNTX(RELATEDTABLE('Product'), [ProductKey])
```

For more information about the RELATEDTABLE function, see the topic *RELATEDTABLE Function (DAX)* in the SQL Server 2016 Technical Documentation:

 **RELATEDTABLE Function (DAX)**

<http://aka.ms/Ey4h2a>

Referencing Tables with Multiple Relationships

Tabular data models support multiple relationships between tables. This is useful if, for example, the **Date** and **Sales** tables have multiple relationships to support different dates, such as order date, due date, and delivery date. Multiple relationships appear in the diagram view of the data model designer, with the active relationship shown as an unbroken line. Non-active relationships appear as dotted lines.

Only one relationship can be active and this will automatically be used when performing calculations using data from the two tables. You can use the USERRELATIONSHIP function when you are writing DAX formulas to override this behavior and force PowerPivot to use the relationship that you specify. For example, if the default relationship between your fact table and date table is based on the **Order Date** column—but you need to aggregate data by shipping date—you can write a DAX measure containing the USERRELATIONSHIP function to achieve this.

 **Note:** USERRELATIONSHIP does not create a relationship. The relationship must already exist for you to use it.

For example, the following code calculates the sum of the **Sales Amount** column based on shipping date:

DAX USERRELATIONSHIP

```
=CALCULATE(SUM(Reseller Sales[Sales Amount]), USERRELATIONSHIP('Reseller Sales' [ShipDateKey], 'Date' [DateKey]))
```

 **Note:** You can only use USERRELATIONSHIP with other functions that accept a filter as an argument, such as CALCULATE in the previous example.

For more information about the USERELATIONSHIP function, see the topic *USERELATIONSHIP Function (DAX)* in the SQL Server 2016 Technical Documentation:

 **USERELATIONSHIP Function (DAX)**

<http://aka.ms/Vdbozm>

For more information about working with relationships in tabular models, see the topic *Relationships (SSAS Tabular)* in the SQL Server 2016 Technical Documentation:

 **Relationships (SSAS Tabular)**

<http://aka.ms/Guu6au>

Demonstration: Using a Specific Relationship in a Measure

In this demonstration, you will see how to:

- Retrieve a value from an unrelated table.
- Retrieve a value from a related table.
- Aggregate related values.
- Analyze related data.

Demonstration Steps

Retrieve a Value from an Unrelated Table

1. Ensure that you have completed the previous demonstration.
2. In Visual Studio, with the Model.bim pane open in data view, on the **Customer** tab, after the existing columns, double-click the **Add Column** header and enter the column name **First Fiscal Year**.
3. With the **First Fiscal Year** column selected, in the formula bar, enter the following DAX expression:

```
=LOOKUPVALUE('Order Date'[FiscalYear], [Date], [DateFirstPurchase])
```

Wait for the table to finish updating, and note the calculated values in the new column.

Retrieve a Value from a Related Table

1. On the **Customer** tab, after the existing columns, double-click the **Add Column** header and enter the column name **Country-Region**.
2. With the **Country-Region** column selected, in the formula bar, enter the following DAX expression:

```
=RELATED('Geography'[CountryRegionCode])
```

Wait for the table to finish updating, and note the calculated values in the new column.

Aggregate Related Values

1. On the **Customer** tab, after the existing columns, double-click the **Add Column** header and enter the column name **Order Count**.
2. With the **Order Count** column selected, in the formula bar, enter the following DAX expression:

```
=COUNTAX(RELATEDTABLE('Internet Sales'), [SalesOrderNumber])
```

Wait for the table to finish updating, and note the calculated values in the new column.

3. On the **File** menu, click **Save All**.

Analyze Related Data

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.
3. When Excel opens, in the **Internet Sales** table, select the **Revenue** measure.
4. In the **Customer** table, select **First Fiscal Year** so that the PivotTable shows sales revenue by customer, based on the first year in which customers made a purchase.
5. In the **Customer** table, clear **First Fiscal Year** and select **Country-Region** so that the PivotTable shows sales revenue by customer country or region.
6. In the **Customer** table, clear **Country-Region** and select **Order Count** so that the PivotTable shows sales revenue by customer, based on the number of orders placed.
7. Close Excel without saving the workbook.
8. Leave Visual Studio open for the next demonstration.

Introduction to Time Intelligence

You can use time intelligence functions to compare data from a time period against equivalent data from another time period. For example, you might want to compare sales for the first quarter of this year against sales for the first quarter of last year.

To support time intelligence functionality, you should ensure that your data and tables meet the following criteria:

- The model should have a separate table that contains only date information.
- The date table should have a continuous range of dates without gaps.
- The date table column that uses the date data type should use “day” as the lowest level of granularity.

- Compare data from one time period against equivalent data from a different time period
- The tabular model should include a separate table that contains only date information
 - The date table should have a continuous range of dates without gaps
 - The column in the date table that uses the date data type should use “day” as the lowest level of granularity
 - Mark the table as Date Table to use time intelligence functions against that table

Relationships between a table containing a measure and a date table do not need to be based on columns that use the date data type to support time intelligence functionality. Instead, you can base these relationships on columns containing integer values. This is useful because it is likely that the databases you use as sources for analysis will contain fact and date tables that are related by columns using the integer data type—particularly if the source database is a data warehouse. For example, in the **AdventureWorksDW** database, the **FactResellerSales** table relates to the **DimDate** table on the **DateKey** column. To handle time intelligence correctly when integers are used in relationships, you must manually mark the date table as **Date Table** and specify the column in the table that contains the date values. This column must use the date data type.

Time Intelligence Functions

DAX provides a range of time intelligence functions that support common business analysis scenarios. These include (among many others):

- NEXTMONTH, NEXTQUARTER, NEXTYEAR. These functions return a table that contains a range of dates representing the time period after the current context.
- PREVIOUSMONTH, PREVIOUSQUARTER, PREVIOUSYEAR. These functions return a table that contains a range of dates representing the time period preceding the current context.
- SAMEPERIODLASTYEAR, PARALLELPERIOD. You can use these functions to compare measure values across the same period in different timespans; for example, to compare sales for a month in the current year with sales in the same month in the previous year.
- TOTALMTD, TOTALQTD, TOTALYTD. These functions calculate a running total for the corresponding time period. For example, you could use TOTALQTD to calculate sales for the quarter so far.

For more information about DAX time intelligence functions, see the topic *Time Intelligence Functions (DAX)* in the SQL Server 2016 Technical Documentation:



Time Intelligence Functions (DAX)

<http://aka.ms/Vwsifx>

Demonstration: Using Time Intelligence

In this demonstration, you will see how to:

- Mark a table as a date table.
- Create a measure that uses time intelligence.
- Analyze data using time intelligence.

Demonstration Steps

Mark a Table As a Date Table

1. Ensure you have completed the previous demonstration.
2. In Visual Studio on 20768B-MIA-SQL, with the Model.bim pane open in data view, click the **Order Date** tab.
3. On the **Table** menu, point to **Date** and click the **Mark As Date Table**.
4. In the **Mark As Date Table** dialog box, select the **Date** column, and click **OK**.

Create a Measure That Uses Time Intelligence

1. In the model designer, click the **Internet Sales** tab.
2. Click the first empty cell in the measure grid, then in the formula bar, enter the following DAX expression:

```
Previous Year Revenue:=CALCULATE(SUM([SalesAmount]), SAMEPERIODLASTYEAR('Order Date'[Date]))
```

3. Select the cell containing the **Previous Year Revenue** measure, and press F4 to display the properties pane. Then set the **Format** property to **Currency**.
4. On the **File** menu, click **Save All**.

Analyze Data Using Time Intelligence

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.
3. When Excel opens, in the **Internet Sales** table, select the **Revenue** and **Previous Year Revenue** measures.
4. In the **Order Date** table, drag **Calendar Date** to the **Rows** area so that the PivotTable shows sales revenue and previous year revenue for each year.
5. Expand 2007 and 2008, and note that the revenue for each quarter in 2007 is shown as the previous year revenue for the same quarters in 2008.
6. Close Excel without saving the workbook.
7. Close Visual Studio.

Implementing Parent-Child Hierarchies

Unlike in MDX, there is no inherent support for a parent-child hierarchy within a tabular data model. To define a parent-child hierarchy, you must use DAX expressions including functions that can be used to specify the path of parent and child IDs, and the distinct levels you want to support within the hierarchy. Understanding the functions available is important because parent-child hierarchies can be commonplace within an analytical data model. To create a parent-child hierarchy in a tabular data model, you must use the **PATH**, **LOOKUPVALUE**, and **PATHITEM** DAX functions.

- Requires calculated measures based on DAX functions
 - **PATH**
 - **LOOKUPVALUE**
 - **PATHITEM**
- Create levels in a table
- Create the hierarchy in the data model

The **PATH** Function

The **PATH** function returns a value to denote the entire parent values related to the key value on which the **PATH** statement is based. You can create a calculated measure that stores the value of the path item. For example, you could create an additional column named **EmployeeLevels** that stores this value in an **Employee** table.

The **PATH** function has the following syntax:

DAX PATH Syntax

```
PATH(<id_columnName>, <parent_columnName>)
```

PATH takes the following parameters:

- **<id_columnName>** refers to the key column in a table. For example, in an **Employees** table, the key column might be **EmployeeID**.
- **<parent_columnName>** refers to the parent column for a key column in a table. For example, in the **Employees** table, this might be **ManagerID**.

There are other DAX functions similar to PATHITEM that are used to work with other aspects of hierarchical data, including PATHITEMREVERSE, PATHLENGTH, and PATHCONTAINS. For more information about all the DAX hierarchy functions, see the topic *Parent and Child Functions (DAX)* in the SQL Server 2016 Technical Documentation:



Parent and Child Functions (DAX)

<http://aka.ms/Jqr010>

The LOOKUPVALUE and PATHITEM Functions

To return a specific value and more meaningful information within a parent-child hierarchy from the PATH function, you can use the LOOKUPVALUE and PATHITEM functions together. The LOOKUPVALUE function returns the value to display and PATHITEM determines the level from which the value should be returned. For example, instead of returning an **EmployeeID** and a **ManagerID** in a parent-child hierarchy, both functions could be used together to return the **Name** attribute of an employee.

The LOOKUPVALUE function has the following syntax:

DAX LOOKUPVALUE Syntax

```
LOOKUPVALUE( <result_columnName>, <search_columnName>, <search_value>[,  
<search_columnName>, <search_value>]...)
```

LOOKUPVALUE takes the following parameters:

- <result_columnName> is the name of an existing column that contains the value you want to return.
- <search_columnName> is the name of an existing column on which the lookup is performed.
- <search_value> provides a filter for the LOOKUPVALUE function; this can include a string literal value or another function, such as PATHITEM, to filter the data.
- Optionally, additional <search_columnName> and <search_value> parameters can be defined.

The PATHITEM function has the following syntax:

DAX PATHITEM Syntax

```
PATHITEM(<path>, <position>[, <type>])
```

PATHITEM takes the following parameters:

- <path> refers to a column that contains the results of the PATH function. In the example of an **Employees** table, this could be a column named **EmployeeLevels**.
- <position> is an integer expression referring to the position of the item to be returned.
- <type> is an optional parameter that can be used to determine the data type that the result should be returned in. A value of 0 is text—which is the default—and a value of 1 is an integer data type.

You can use the LOOKUPVALUE and PATHITEM functions together to create a parent-child hierarchy.

An additional calculated measure can be created in a calculated column that contains the following code to populate the **Name** value for an employee's immediate manager:

Employee's Immediate Manager

```
LOOKUPVALUE ([Name], [EmployeeId], PATHITEM ([EmployeeLevels], 1))
```

The calculated measure to create the second level of the employee hierarchy in a second calculated column is shown in the following code example:

Second Level Manager

```
LOOKUPVALUE ([Name], [EmployeeId], PATHITEM ([EmployeeLevels], 2))
```

Design Considerations

When you design parent-child hierarchies in analytical data models, consider the following points:

- Ensure that the parent and child keys are of compatible data types.
- For the best query and processing performance, ensure that a self-join relationship exists between the parent key and the child.
- Because the manager levels are hard coded, you may need to change the data model if the organizational structure changes—to allow more levels of management than the model defines.

For more information about working with parent-child hierarchies, see the topic *Understanding Functions for Parent-Child Hierarchies in DAX* in the SQL Server 2016 Technical Documentation:



Understanding Functions for Parent-Child Hierarchies in DAX

<http://aka.ms/Dqba35>

Check Your Knowledge

Question	
You are creating a calculated column in a table, based on the values in a column in another table. The two tables have a one-to-many relationship, based on a key column. Which DAX function should you use to perform the lookup in the most efficient way?	
Select the correct answer.	
<input type="checkbox"/>	USERRELATIONSHIP
<input type="checkbox"/>	RELATEDTABLE
<input type="checkbox"/>	RELATED
<input type="checkbox"/>	LOOKUPVALUE

Lab: Using DAX to Enhance a Tabular Data Model

Scenario

A senior business analyst at Adventure Works Cycles wants to perform some detailed analysis of the data in a tabular data model. To accomplish this, you will need to use DAX to enhance the data model.

Objectives

At the end of this lab, you will be able to:

- Use DAX to create calculated columns.
- Use DAX to create measures.
- Create a KPI that uses a DAX expression for the target value.
- Use DAX to implement a parent-child hierarchy.

Estimated Time: 75 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTURWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Creating Calculated Columns

Scenario

You have the raw data that you require, but this data would be more useful if it could be adapted by using calculations.

To improve the presentation of data, you want to concatenate your employees' first name, middle name, and last name fields. You want to make profit more visible by creating a calculated value that subtracts a product's cost from its selling price. Furthermore, you want to create a calculation to improve the categorization of products.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Concatenate Text Values
3. Calculate a Numeric Value
4. Show Related Values
5. View Calculated Columns in Excel

► Task 1: Prepare the Lab Environment

1. Ensure the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the D:\Labfiles\Lab08\Starter folder as Administrator.

► Task 2: Concatenate Text Values

1. Use Visual Studio to open the **AWSalesTab.sln** solution in the D:\Labfiles\Lab08\Starter folder, and view the data model designer. If the **Tabular model designer** dialog box is displayed, in the **Workspace server** list, select **localhost\SQL2**, and then click **OK**.
2. View the model, running a script in the server if prompted, then process the model using the impersonation credentials **ADVENTUREWORKS\ServiceAcct** with the password **Pa\$\$w0rd**.

3. In the **Employee** table, create a custom column named **Employee Name** that concatenates the first name, middle name (if there is one), and last name of the employee. You can use the following DAX expression to do this:

```
=[First Name] & IF(ISBLANK([Middle Name]), "", CONCATENATE(" ", [Middle Name])) &
CONCATENATE(" ", [Last Name])
```

4. After you have defined the **Employee Name** column, hide the **First Name**, **Middle Name**, and **Last Name** columns from client tools and save the data model.

► Task 3: Calculate a Numeric Value

1. In the **Reseller Sales** table, create a custom column named **SalesProfit** that calculates profit by subtracting the total product cost from the sales amount. You can use the following DAX expression to do this:

```
=[SalesAmount]-[TotalProductCost]
```

2. After you have defined the **SalesProfit** column, confirm that its data type has been automatically detected as **Currency**, and then save the model.

► Task 4: Show Related Values

1. In the **Product** table, create a column named **Subcategory** that returns the related **Subcategory** column from the **Product Subcategory** table. You can use the following DAX expression to do this:

```
=RELATED('Product Subcategory'[Subcategory])
```

Note: Some products at the beginning of the table are uncategorized.

2. In the **Product** table, create a column named **Category** that returns the related **Category** column from the **Product Category** table. You can use the following DAX expression to do this:

```
=RELATED('Product Category'[Category])
```

Note: Some products at the beginning of the table are uncategorized.

3. In the **Product** table, create a hierarchy named **Categorized Products** that includes the following columns:
 - Category
 - Subcategory
 - Product Name
4. Hide all columns in the **Product Subcategory** and **Product Category** tables from client tools.
5. Save the data model.

► Task 5: View Calculated Columns in Excel

1. Analyze the model you have created in Excel; view the **Revenue** measure by the **Categorized Products** hierarchy on rows, and the **Employee Name** column on columns.
2. Verify that the calculated columns work as expected, and then close Excel without saving the workbook. Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have a calculated column named **Employee Name** in the **Employee** table, a calculated measure named **SalesProfit** in the **Reseller Sales** table, and a hierarchy named **Categorized Products** in the **Product** table.

Exercise 2: Creating Measures

Scenario

Business users have requested the ability to view the following aggregated measures:

- Profit
- Margin
- Previous Year Revenue

The main tasks for this exercise are as follows:

1. Create a Measure That Aggregates a Column
2. Create a Measure That References Other Measures
3. Create a Measure That Uses Time Intelligence
4. View Measures in Excel

► Task 1: Create a Measure That Aggregates a Column

1. In the **Reseller Sales** table, create a measure named **Profit** that sums the values in the **SalesProfit** column. You can use the following DAX expression to do this:

```
Profit:=SUM([SalesProfit])
```

2. Format the **Profit** measure as **Currency**.
3. Hide the **SalesProfit** column from client tools.
4. Save the data model.

► Task 2: Create a Measure That References Other Measures

1. In the **Reseller Sales** table, create a measure named **Margin** that divides the **Profit** measure by **Revenue**. You can use the following DAX expression to do this:

```
Margin:=[Profit]/[Revenue]
```

2. Format the **Margin** measure as **Percentage**.
3. Save the data model.

► Task 3: Create a Measure That Uses Time Intelligence

1. Mark the **Order Date** table as a date table, specifying the **Date** column as the table's unique identifier.
2. In the **Reseller Sales** table, create a measure named **Previous Year Revenue** that uses the **SAMEPERIODELASTYEAR** function to calculate the sum of the **SalesAmount** column for the equivalent time period in the previous year. You can use the following DAX expression to do this:

```
Previous Year Revenue:=CALCULATE([Revenue], SAMEPERIODELASTYEAR('Order Date'[Date]))
```

3. Format the **Previous Year Revenue** measure as **Currency**.
4. Save the data model.

► Task 4: View Measures in Excel

1. Analyze the model you have created in Excel; view the **Revenue, Profit, Margin, and Previous Year Revenue** measures by the **Calendar Date** hierarchy on rows.
2. Verify that the measures you created all work as expected, and then close Excel without saving the workbook. Leave Visual Studio open for the next exercise.

Results: At the end of this exercise, the **Reseller Sales** table should contain the following measures:

Profit

Margin

Previous Year Revenue

Exercise 3: Creating a KPI

Scenario

Adventure Works wants to achieve year-on-year revenue growth of 20 percent. To track this goal, you plan to create a KPI.

The main tasks for this exercise are as follows:

1. Create a Measure to Calculate a KPI Goal
2. Create a KPI
3. View a KPI in Excel

► Task 1: Create a Measure to Calculate a KPI Goal

1. In the **Reseller Sales** table, create a measure named **Revenue Goal** that multiplies the **Previous Year Revenue** measure by 1.2. You can use the following DAX expression to do this:

```
Revenue Goal:=[Previous Year Revenue] * 1.2
```

2. Format the **Revenue Goal** measure as **Currency** and hide it from client tools.
3. Save the data model.

► Task 2: Create a KPI

1. In the **Reseller Sales** table, create a KPI based on the **Revenue** measure.
2. Configure the KPI to use the **Revenue** measure as the target value, and set status thresholds at 75% and 95%.
3. When you have created the KPI, save the data model.

► Task 3: View a KPI in Excel

1. Analyze the model you have created in Excel; view the **Value, Goal, and Status** of the **Revenue Goal** KPI by the **Fiscal Date** hierarchy on rows.
2. Verify that the KPI indicates the status appropriately for the quarters and months in 2008, and then close Excel without saving the workbook. Leave Visual Studio open for the next exercise.

Results: At the end of this exercise, the **Reseller Sales** table should include a measure named **Revenue Goal** and a KPI based on the **Revenue** measure.

Exercise 4: Implementing a Parent-Child Hierarchy

Scenario

The CEO at Adventure Works wants to view sales results aggregated by each sales manager, and then drill down to view individual employee sales results.

The main tasks for this exercise are as follows:

1. Create a Path Column
2. Create a Column for Each Hierarchy Level
3. Use the Calculated Columns in a Hierarchy
4. View a Parent-Child Hierarchy in Excel

► Task 1: Create a Path Column

1. In the **Employee** table, add a column named **Path** that uses the PATH function to define the recursive chain of parent employee keys. You can use the following DAX expression to do this:

```
=PATH([EmployeeKey], [ParentEmployeeKey])
```

2. Save the data model.

► Task 2: Create a Column for Each Hierarchy Level

1. In the **Employee** table, add the following calculated columns:

- Level1

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 1, 1))
```

- Level2

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 2, 1))
```

- Level3

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 3, 1))
```

- Level4

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 4, 1))
```

2. Save the data model.

► Task 3: Use the Calculated Columns in a Hierarchy

1. Create a hierarchy named **Employee Hierarchy** in the **Employee** table.
2. Add the **Level1**, **Level2**, **Level3**, and **Level4** columns to the hierarchy, in that order.

► Task 4: View a Parent-Child Hierarchy in Excel

1. Analyze the model you have created in Excel; view the **Revenue** measure by **Employee Hierarchy**.
2. Expand **Employee Hierarchy** to view sales totals for managers and their subordinates. Note that the personal revenue for sales managers is shown with a blank employee name under the total for that sales manager.
3. When you have finished, close Excel without saving the workbook, and close Visual Studio.

Results: At the end of this exercise, the **Employee** table should include a hierarchy named **Employee Hierarchy**.

Check Your Knowledge

Question	
Which DAX time intelligence function would you use to calculate a measure based on values for the end of the next quarter?	
Select the correct answer.	
<input type="checkbox"/>	ENDOFQUARTER
<input type="checkbox"/>	NEXTQUARTER
<input type="checkbox"/>	OPENINGBALANCEQUARTER
<input type="checkbox"/>	PREVIOUSQUARTER

Module Review and Takeaways

In this module, you have learned how to use DAX to enhance tabular data models.

Review Question(s)

Question: Will time intelligence functionality be useful to you? What kinds of time-based analyses might you want to perform? Which are the relevant time intelligence functions that you might use to achieve this?

Module 9

Performing Predictive Analysis with Data Mining

Contents:

Module Overview	9-1
Lesson 1: Overview of Data Mining	9-2
Lesson 2: Creating a Custom Data Mining Solution	9-7
Lesson 3: Validating a Data Mining Model	9-15
Lesson 4: Connecting to and Consuming a Data Mining Model	9-20
Lab: Using Data Mining to Support a Marketing Campaign	9-23
Module Review and Takeaways	9-28

Module Overview

With data mining, you can use the data you own to gain insights that could help you make intelligent decisions about your business. Microsoft® SQL Server® 2016 Analysis Services includes data mining tools that you can use to identify patterns in your data, helping you to determine why particular things happen and to predict what might occur in the future. This module introduces data mining, describes how to create a data mining solution; how to validate data mining models; and how to incorporate data mining results into Reporting Services reports.

Objectives

After completing this module, you will be able to:

- Describe the key data mining concepts.
- Create a data mining solution.
- Validate data mining models.
- Use data mining data in a report.

Lesson 1

Overview of Data Mining

Data mining is a type of data analysis that involves using statistical models to reveal connections and correlations in large sets of data that would otherwise be very difficult, or even impossible, to identify.

This lesson explains the purpose of data mining, and describes the components of a data mining solution, in addition to the algorithms used to build prediction models.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the purpose of data mining.
- Describe the components of an Analysis Services data mining solution.
- Describe the different types of data mining algorithms you can use in Analysis Services data mining solutions.

What Is Data Mining?

Exploration of the data in a data warehouse has the potential to reveal patterns and trends that can be useful to organizations in many ways. These include being able to predict customer behavior, or to provide customers with product recommendations in real time as they shop online. However, the size and complexity of data in large data warehouses can make it very difficult to derive useful information from raw data by using standard data analysis tools and techniques. Data mining is the statistical analysis of large volumes of data that would be very difficult to analyze manually. With data mining, business users can overcome the problems associated with analyzing large and complex data sets to access useful and actionable information.

- Analysis of large data sets to reveal hidden patterns and trends
- Data mining algorithms
 - Different types of statistical analyses for different scenarios, for example:
 - Predict future behavior
 - Identify correlation between data values
- Data mining has a range of applications in business, for example:
 - Sales forecasting
 - Targeted advertising
 - Making recommendations
 - Risk assessment

Data Mining Algorithms

Data mining involves using algorithms to search through data to extract patterns and trends. For example, a retail organization might want to discover if customers in large urban areas are more likely to buy goods in a high price range, compared to customers living in more rural locations. The organization could use data mining to determine any correlations between location and price; how strong the relationships might be; or how multiple correlating factors affect outcomes when they are considered together. There are various data mining algorithms you can use, depending on the type of questions that you need to be answered.

Data mining can help you to identify trends and patterns that may not be immediately obvious. You can use data mining to predict unknown values, based on statistics and patterns in previously analyzed sets of data. This is useful when you are trying to predict and plan for future events.

Data Mining Scenarios

Examples of where businesses find value in data mining include:

- **Forecasting.** Companies can use data mining to analyze sales patterns to determine, for example, when a particular product will sell or how stores will profit over time.
- **Targeted advertising.** Organizations can create target marketing and advertising campaigns by identifying the factors that best predict the customers who are most likely to purchase a given product, and then targeting the campaign at those individuals.
- **Recommendations.** Companies can recommend merchandise to customers as they shop online by analyzing their purchasing histories and previous sales of products.
- **Risk assessment.** Insurance companies can assess the likelihood of a claim being fraudulent by employing data mining algorithms that use the outcomes of previous claims to create a weighting for each factor affecting the new claim. Credit ratings agencies can use financial and customer history data to predict which individuals are most likely to default on a loan.

For a general introduction to data mining on SQL Server, see the topic *Data Mining Concepts* in the SQL Server 2016 Technical Documentation:

 **Data Mining Concepts**

<http://aka.ms/dz1swm>

Data Mining Concepts

The basic unit of data in data mining is a case. Data mining involves processing cases through an analytical model. To begin working with data mining, you must identify one or more data sources from where your cases will be drawn. You then create a data mining structure—which defines cases through metadata—and one or more data mining models, which hold statistical patterns found in the case data.

- **Data sources**
 - Any valid OLE DB or .Net data provider
- **Data mining structure**
 - Data source view (DSV)
 - Case table and mining structure columns
 - Specifies training set and testing set
- **Case table**
 - Aggregations of source data used by mining models
 - Content type defines relationship between column values
- **Data mining model**
 - Uses a single mining algorithm
 - Includes columns from mining structure

 **Note:** To create a data mining solution in SQL Server Analysis Services, you must install Analysis Services in Multidimensional and Data Mining mode. Tabular mode and PowerPivot for Microsoft SharePoint® mode do not support data mining. You can install multiple instances of Analysis Services, running in different modes, on the same server.

Data Sources

The source data analyzed by a data mining solution is not stored in the solution—instead, the solution contains bindings to the data sources. When the data mining solution is processed, an aggregated summary of the source data is generated; the aggregations may be discarded after processing, or stored in the solution for use in later operations.

You can use an Analysis Services data source view (DSV) to provide your source data definition; a DSV may contain bindings to, and define relationships between, multiple data sources. Data sources are not restricted to OLAP and relational data in SQL Server; Analysis Services supports many data sources for which an OLE DB or .Net data provider is available.

For a list of data sources supported by Analysis Services for use in data mining, see the topic *Supported Data Sources (SSAS Multidimensional)* in the SQL Server 2016 Technical Documentation:



Supported Data Sources (SSAS Multidimensional)

<http://aka.ms/Yy2cps>

Data Mining Structure

The data mining structure is the central component of a data mining solution. It contains definitions for the data from which mining models in the solution are built, including:

- The DSV which defines the source data.
- The case table and the mining structure columns that, in turn, express the data and content types.
- A training set definition.
- A testing set definition.

Different portions of the case table data may be used to train and test data mining models.

For more information about mining structures, see the topic *Mining Structures (Analysis Services - Data Mining)* in the SQL Server 2016 Technical Documentation:



Mining Structures (Analysis Services - Data Mining)

<http://aka.ms/Otbw5s>

Case Table

The case table stores aggregations from source data; the aggregations are used by the data mining models.

Cases may be simple, in which circumstances each row in the case table corresponds to a case—for example, if each case relates to a customer. When cases are complex, two or more case tables are connected with a one-to-many relationship; these are referred to as nested cases, or nested tables.

When you specify the case table, you define the data type and content type for each column. The content type defines how the values in a column relate to one another. Defining content types makes data mining models more accurate as, when a content type is specified, the mining model is less likely to make poor inferences about the statistical relationship between data values.

Some examples of content types include:

- **Discrete.** These are values that are not part of a sequence. Discrete columns contain data that has a finite number of values with no continuum between them. Examples of discrete data include number of children, phone number, or gender. Discrete values can be numeric or non-numeric.
- **Continuous.** These represent sequences of numeric data on a scale. Examples of continuous data include temperature and weight.
- **Cyclical.** These represent data that is organized into limited, ordered sets that repeat. Examples of cyclical data include numbered days of the week or numbered months of the year.

For a full list of content types, see the topic *Content Types (Data Mining)* in the SQL Server 2016 Technical Documentation:



Content Types (Data Mining)

<http://aka.ms/Fpba6j>

Data Mining Model

A data mining structure may be referenced by one or more data mining models. Each data mining model uses a data mining algorithm, which you specify when you create the model. The mining model uses the algorithm to analyze data from the data mining structure. When you create a data mining model, you define the columns from the data mining structure, and specify a usage value for each column. Columns from the data mining structure for which you do not specify a usage value are not included in the model. The available usage values are:

- **Key.** This indicates a key column containing values that identify each row uniquely.
- **Input.** This indicates that the model should use this column to help forecast values for the predictable column.
- **Predict.** This indicates the column for which you want to predict values in the mining mode. Its value is also used as an input.
- **Predict Only.** This indicates the column for which you want to predict values in the mining mode. Its value is not used as an input.

For example, if you want to predict the amount of money each customer is likely to spend on a supermarket website, you could use **CustomerID** as the key column; **CustomerSpend** as the predictable column; and various others, containing data such as address, age, and number of children, as input columns.

For more information on data mining models, see the topic *Mining Models (Analysis Services - Data Mining)* in the SQL Server 2016 Technical Documentation:

 **Mining Models (Analysis Services - Data Mining)**

<http://aka.ms/ipkyve>

Data Mining Algorithms

Data mining algorithms provide rules for the analysis of data. Analysis Services provides nine algorithms that you can use to create data mining models. You can also add further algorithms using third-party plugins if necessary.

Widely-used data mining algorithms fall into the following broad categories:

- **Classification algorithms.** These predict one or more discrete variables based on other attributes. Microsoft Decision Trees is an example that might be used to forecast whether a customer will purchase a particular product. Microsoft Neural Network and Microsoft Naïve Bayes are other classification algorithms.

- Classification algorithms:
 - Microsoft Decision Trees
 - Microsoft Neural Network
 - Microsoft Naive Bayes
- Regression algorithms:
 - Microsoft Time Series
 - Microsoft Linear Regression
 - Microsoft Logistic Regression
- Segmentation or clustering algorithms:
 - Microsoft Clustering
- Association algorithms:
 - Microsoft Association
- Sequence analysis algorithms:
 - Microsoft Sequence Clustering

The best algorithm for your purposes depends on a number of factors, such as the volume of data or the specific types of column being analyzed. For example, the Naïve Bayes algorithm can use discrete columns (such as **City**) to classify data but does not support continuous columns that may be grouped into ranges (such as **Age**). If you need to classify data into groups based on ranges of continuous values, the Decision Trees algorithm may be a better choice.

- **Regression algorithms.** These predict one or more continuous variables, such as profit or loss. The Microsoft Time Series algorithm is an example that might be used to determine a retail store's seasonal sales for the coming year. Microsoft Linear Regression and Microsoft Logistic Regression can also predict continuous variables.
- **Segmentation or clustering algorithms.** These divide data into groups or clusters of items that have similar properties. Microsoft Clustering, for example, might be used to divide customers into groups with similar purchasing habits or preferences.
- **Association algorithms.** These find correlation between different attributes in a data set. The Microsoft Association algorithm is an example that might be used to describe which items—such as products being purchased together—are likely to appear in a transaction.
- **Sequence analysis algorithms.** Sequence analysis finds common sequences in data. The Microsoft Sequence Clustering algorithm is an example that might be used to find common web clickthrough paths, or the order of placing items in a cart.

For more information about the data mining algorithms included in Analysis Services, see the topic *Data Mining Algorithms (Analysis Services - Data Mining)* in the SQL Server 2016 Technical Documentation:



Data Mining Algorithms (Analysis Services - Data Mining)

<http://aka.ms/Nfg2r8>

For more information about the plugin architecture for adding third-party data mining algorithms, see the topic *Plugin Algorithms* in the SQL Server 2016 Technical Documentation:



Plugin Algorithms

<http://aka.ms/Hndiao>

Sequencing Activity

Put the following components of a data mining solution in order by numbering each to indicate the correct sequence from source to target.

	Steps
	Source data
	Analysis Services data source view
	Data Mining Structure
	Data Mining Model

Lesson 2

Creating a Custom Data Mining Solution

To produce a data mining solution in Analysis Services, you first create a model that describes your business problem. The model is then trained by running your data through algorithms that generate a mathematical model. You can then either visually explore the mining model or create prediction queries against it. Analysis Services data mining structures can use datasets from both relational and multidimensional databases, and there is a set of algorithms you can use to investigate that data in various ways.

This lesson describes how to create a data mining structure and data mining model. The lesson also describes how to edit data models, and introduces the Data Mining Extensions (DMX) language, which you can use to work with data mining in Analysis Services.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the tools used to create a data mining structure.
- Describe the tools used to modify a data mining structure.
- View a data mining model.
- Describe and work with DMX.

Data Mining Tools

In Visual Studio® SQL Server Data Tools (SSDT), you can use the Data Mining Wizard to create new data mining structures and models. You can then use the Data Mining Designer in SSDT to configure the structure. You can also use the Data Mining Client for Excel® add-in to create and edit data mining structures and models.

Data Mining Wizard

You can use the Data Mining Wizard to define the data mining structure and, optionally, to create the first data mining model for the structure. The Data Mining Wizard starts automatically when you create a new data mining structure. When you run the Data Mining Wizard, you can configure the data source view, the case table, the Key, Input and Predictable columns for the training data, the column data types and content types, and the proportion of the data to use for the testing set. If you choose to create a data mining model at the same time as a data mining structure, you must also specify the mining technique for the model to use.

Data Mining Designer

You can use the Data Mining Designer to configure the data mining structure and data mining model that you created with the Data Mining Wizard. For example, you can use the Data Mining Designer to add new data mining models, train, browse, and compare models, and to create predictions based on data mining models.

- 
- SQL Server Data Tools
 - Data Mining Wizard
 - Data Mining Designer
 - Data Mining Client for Excel

Data Mining Client for Excel

You can use the Data Mining Client for Excel add-in to create data mining structures and models. You can also use the add-in to edit models and structures you created by using SQL Server Data Tools. The add-in includes dedicated wizards for creating the more commonly-used data mining models, which make the process much simpler for inexperienced users.

For more information about the Data Mining Wizard, see the topic *Data Mining Wizard (Analysis Services - Data Mining)* in the SQL Server 2016 Technical Documentation:



Data Mining Wizard (Analysis Services - Data Mining)

<http://aka.ms/Hw9h83>

Demonstration: Using the Data Mining Wizard

In this demonstration, you will see how to:

- Create a data mining project in Visual Studio.
- Create a data mining structure and a data mining model using the Data Mining Wizard.

Demonstration Steps

Create a Data Mining Project in Visual Studio

Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.

In the D:\Demofiles\Mod09 folder, run **Setup.cmd** as Administrator.

In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

Start Visual Studio and on the **File** menu, point to **New**, and then click **Project**.

In the **New Project** dialog box, expand **Business Intelligence**, then click **Analysis Services**, then click **Analysis Services Multidimensional and Data Mining Project**. Type **Mine AW** in the **Name** box, then type **D:\Demofiles\Mod09** in the **Location** box, then click **OK**.

In Solution Explorer, right-click **Data Sources**, and then click **New Data Source**.

In the **Data Source Wizard**, on the Welcome to the Data Source Wizard page, click **Next**, and then on the Select How to Define the Connection page, click **New**.

In the **Connection Manager** dialog box, in the **Server name** field, type **MIA-SQL**, in the **Select or enter a database name** drop-down list, click **AdventureWorksDW**, and then click **OK**.

In the **Data Source Wizard**, on the Select How to Define the Connection page, click **Next**. On the Impersonation Information page, click **Use a specific Windows user name and password**, then type the following credentials and click **Next**:

- **User name:** ADVENTUREWORKS\ServiceAcct
- **Password:** Pa\$\$w0rd

On the Completing the Wizard page, click **Finish**.

In Solution Explorer, right-click **Data Source Views**, and then click **New Data Source View**.

In the **Data Source View Wizard**, on the Welcome to the Data Source View Wizard page, click **Next**. On the Select a Data Source page, ensure that **Adventure Works DW** is selected, and then click **Next**.

On the Select Tables and Views page, in the **Available objects** list, click **ProspectiveBuyer (dbo)**, hold the **Ctrl** key and click **vTargetMail (dbo)**, and click the > button to move the selected objects to the **Included objects** list. Then click **Next**.

On the Completing the Wizard page, in the **Name** field, type **AW DW View**, and then click **Finish**.

Create a Data Mining Structure and a Data Mining Model Using the Data Mining Wizard

- In Solution Explorer, right-click **Mining Structures**, click **New Mining Structure**, and then in the **Data Mining Wizard**, on the Welcome to the Data Mining Wizard page, click **Next**.

On the Select the Definition Method page, ensure that **From existing relational database or data warehouse** is selected, and then click **Next**.

On the Create the Data Mining Structure page, ensure that **Create mining structure with mining model is selected**, ensure that **Microsoft Decision Trees** is selected, and then click **Next**.

On the Select a Data Source View page, select **AW DW View** and click **Next**.

On the Specify Table Types page, in the **vTargetMail** row, select the check box in the **Case** column, and then click **Next**.

On the Specify the Training Data page, in the **Mining Model Structure** table, select the following columns and click **Next**:

- **BikeBuyer**: Predictable
- **CustomerKey**: Key
- **All other columns**: Input

On the Specify Columns' Content and Data Type page, click **Detect** and review the content type and data types found. Ensure that the content type of the **Bike Buyer** column is identified as **Discrete**. Then click **Next**.

On the Create Testing Set page, note that the **Percentage of data for testing** value is 30 percent, and then click **Next**.

On the Completing the Wizard page, in the **Mining structure name** box, type **Purchase Prediction**, in the **Mining model** name field type **Purchase Decision Tree**, and then click **Finish**.

On the **Build** menu, click **Deploy Mine AW**. When deployment completes, close the Deployment Progress – Mine AW window.

Leave Visual Studio open for the next demonstration.

Modifying Data Mining Structures and Models

There are two ways to make changes to data mining structures and models.

Data Mining Designer

You can use the Data Mining Designer in Visual Studio to edit data mining structures, add and edit data mining models, test the accuracy of data models, and view model data.

- Visual Studio
 - Data Mining Designer
 - Data Mining Client for Excel

To add models to an existing data mining structure, use the following procedure:

- In Visual Studio, on the **Mining Models** tab, click **Create a related mining model**, or right-click an existing mining model, and then click **New Mining Model**.

In the **New Mining Model** dialog box, enter a model name and select an algorithm for the new model.

Data Mining Client for Excel

You can also use the Data Mining Client for Excel add-in to edit data mining structures and models. When you use the **Add Model to Structure Wizard** in the Data Mining Client for Excel add-in to edit an existing structure by adding a new model, you can select and configure a data mining algorithm. You can use the **Manage Mining Structures and Models** dialog box to rename, delete, clear, process, export, and import structures and models.

For more information about using the Data Mining Designer, see the topic *Data Mining Designer* in the SQL Server 2016 Technical Documentation:



Data Mining Designer

<http://aka.ms/Qe2sgh>

For more information about using the Data Mining Client for Excel to work with data mining structures, see the topic *Advanced Modeling (Data Mining Add-ins for Excel)* in the SQL Server 2016 Technical Documentation:



Advanced Modeling (Data Mining Add-ins for Excel)

<http://aka.ms/Bkdytk>

Demonstration: Using the Data Mining Designer

In this demonstration, you will see how to modify a data mining structure with SSDT, by adding a new data model.

Demonstration Steps

- Ensure you have completed the previous demonstrations in this module.

On the 20768B-MIA-SQL virtual machine, in Visual Studio Solution Explorer, expand **Mining Structures** and double-click **Purchase Prediction.dmm**. In the Purchase Prediction.dmm [Design] window, click **Mining Models**. Right-click anywhere in the **Mining Models** tab, then click **New Mining Model**.

In the **New Mining Model** dialog box, type **Purchase Naive Bayes** in the **Model name** box. In the Algorithm name box, select **Microsoft Naive Bayes**, then click **OK**.

In the **Microsoft Visual Studio** dialog box, read the information message about content types unsupported by the Microsoft Naive Bayes algorithm, then click **Yes**.

In the **Mining Models** tab, in the **Purchase Naive Bayes** column, in the **Name Style** row, select **Ignore** to exclude the column from the model.

On the **Build** menu, click **Deploy Mine AW**. When deployment completes, close the Deployment Progress – Mine AW window.

Leave Visual Studio open for the next demonstration.

Viewing a Data Mining Model

From a design perspective, the results of a data mining model can be viewed in two ways.

Mining Model Viewer

You can view the results of data mining analyses in Visual Studio or SQL Server Management Studio (SSMS) by using the Mining Model Viewer. The viewing options available vary depending on the type of model you want to view; for example, you can use the Microsoft Tree Viewer to browse a decision tree model. The Microsoft Tree Viewer includes the Decision Tree and Dependency Network tabs so you can view results in different graphical formats.

In the Mining Model Viewer, you can refresh data model views and, depending on the viewer used, perform other tasks, such as filter items, or adjust thresholds.

- Mining Model Viewer
 - Visual Studio
 - SQL Server Management Studio
 - Data Mining Client for Excel

Browse Models by Using the Data Mining Client for Excel Add-In

You can also browse mining model results by using the Data Mining Client for Excel add-in. To browse a model, click the Browse in the Model Usage area of the ribbon and then specify the model you want to view. As with the Mining Model Viewer, the way results are displayed varies depending on the type of model.

For more information about browsing data mining models using the Data Mining Client for Excel, see the topic *Browsing Models in Excel (SQL Server Data Mining Add-ins)* in the SQL Server 2016 Technical Documentation:

 **Browsing Models in Excel (SQL Server Data Mining Add-ins)**

<http://aka.ms/l2cclr>

Demonstration: Using the Data Mining Model Viewer

In this demonstration, you will see how to view the results of a data mining model in SSMS.

Demonstration Steps

- Ensure you have completed the previous demonstrations in this module.

On the 20768B-MIA-SQL virtual machine, start SQL Server Management Studio. In the **Connect to Server** dialog box, select **Analysis Services** in the **Server Type** box, then type **MIA-SQL** in the **Server name** box, then click **Connect**.

In Object Explorer expand **Databases**, expand **Mine AW**, then expand **Mining Structures**. Right-click **Purchase Prediction**, then click **Browse**.

In the **Mining Model** drop-down list, ensure that **Purchase Naive Bayes** is selected, and on the **Dependency Network** tab, move the slider gradually from **All Links** to **Strongest Links** to see which factors are the strongest predictors that a customer will purchase a bike.

On the **Attribute Profiles** tab, view the color-coded indicators of the values for each column when compared to customers with a **Bike Buyer** value of **1** or **0**.

On the **Attribute Characteristics** tab, in the **Attribute** drop-down list, ensure that **Bike Buyer** is selected, and in the **Value** drop-down list, select **1**. Then view the probability for each other column value when the **Bike Buyer** value is **1**.

On the **Attribute Discrimination** tab, in the **Attribute** drop-down list, ensure that **Bike Buyer** is selected. In the **Value 1** drop-down list, select **1**, and in the **Value 2** drop-down list, select **0**. Then note how values for all other columns favor a particular **Bike Buyer** value.

In the **Mining Model** drop-down list, select **Purchase Decision Tree**, and on the **Decision Tree** tab, in the **Background** drop-down list, select **1**. Then view the decision tree to see how the other column values influence a value of **1** for **Bike Buyer**.

Close the Purchase Prediction [Browse] window, and leave SQL Server Management Studio open for the next demonstration.

What Is DMX?

Analysis Services includes the DMX query language, which you can use to work with data mining models. DMX is an extension of Transact-SQL, and can be used to create the structure of new data mining models, to train these models, and to browse, manage, and use them to generate predictions.

You can use the result of any DMX query as the basis of a report. By taking advantage of the parameterization and formatting features available in Reporting Services, you can deliver data mining results in a format that is easy to consume.

DMX is composed of Data Definition Language (DDL) statements, Data Manipulation Language (DML) statements, and a set of functions and operators.

DDL Statements

You use DDL statements to create and delete data mining structures and models, and to import and export data mining structures and models. DDL statements include CREATE, ALTER, DROP, IMPORT, EXPORT, and SELECT INTO statements.

The following code example creates a new mining model that uses the Microsoft Naïve Bayes algorithm. The **Bike Buyer** column is defined as the predictable attribute.

DMX DDL

```
CREATE MINING MODEL [NBSample]
(
    CustomerKey LONG KEY,
    Gender TEXT DISCRETE,
    [Number Cars Owned] LONG DISCRETE,
    [Bike Buyer] LONG DISCRETE PREDICT
)
USING Microsoft_Naive_Bayes
```

- Data Definition Language statements
 - CREATE
 - ALTER
 - DROP
 - IMPORT
 - EXPORT
 - SELECT INTO
- Data Manipulation Language statements
 - SELECT
 - PREDICTION JOIN
 - INSERT INTO

DML Statements

You use DML statements to train and browse mining models, and create predictions against them. DML statements include SELECT statements, SELECT statements with PREDICTION JOIN clauses, and INSERT INTO statements for training data mining models.

The following code example returns the midpoint, minimum age, and maximum age for all the values in the **Age** column:

DMX DML

```
SELECT DISTINCT [Age] AS [Midpoint Age],
    RangeMin([Age]) AS [Minimum Age],
    RangeMax([Age]) AS [Maximum Age]
FROM [TM Decision Tree]
```

To help you build DMX prediction queries, SSMS includes a query builder tool and DMX templates you can use as the basis for new queries. In the SQL Server Data Tools environment, you access the query builder tool from the Mining Model Prediction tab of Data Mining Designer.

For more information about DMX, see the topic *Data Mining Extensions (DMX) Reference* in the SQL Server 2016 Technical Documentation:



Data Mining Extensions (DMX) Reference

<http://aka.ms/Kapx7e>

Demonstration: Querying a Mining Model Using DMX

In this demonstration, you will see how to query a data mining model using DMX and the query builder tool.

Demonstration Steps

- In SSMS Object Explorer, right-click the **Purchase Prediction** mining structure and click **Build Prediction Query**.

In the query designer, in the Mining Model pane, click **Select Model**. Then in the **Select Mining Model** dialog box, expand **Purchase Prediction**, click **Purchase Naive Bayes**, and click **OK**.

In the Select Input Table(s) pane, click **Select Case Table**. Then in the **Select Table** dialog box, click **ProspectiveBuyer (dbo)**, and click **OK**.

Under the Mining Model pane, in the **Source** column, select **ProspectiveBuyer** table, and then in the **Field** column, select **EmailAddress**.

Under the row you just added, in the **Source** column, click **Purchase Naive Bayes** mining model, in the **Field** column, click **Bike Buyer**, and in the **Criteria/Argument** column, type **=1**.

Under the row you just added, in the **Source** column, click **Prediction Function**, in the **Field** column, click **PredictProbability**, in the **Alias** column, type **Purchase Probability**, and then drag the Bike Buyer column from the Purchase Naive Bayes model in the Mining Model pane to the **Criteria/Argument** column so it contains the value **[Purchase Naive Bayes].[Bike Buyer]**.

On the **Mining Model** menu, click **Query** to view the DMX code that has been generated.

On the **Mining Model** menu, click **Result** to view the query results. The query returns the email address of every prospective customer who is predicted to buy a bike, along with the probability (expressed as a percentage in fraction format) that this forecast is accurate.

Close the Purchase Prediction [Query] window, and leave SQL Server Management Studio open for the next demonstration.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? All the data mining algorithms implemented by default in Analysis Services can process all content types.	

Lesson 3

Validating a Data Mining Model

Validation is the process of assessing how well your mining models perform against real data. It is important to validate your mining models by understanding their quality and characteristics before you deploy them into a production environment.

This lesson provides an overview of data mining model validation and describes the criteria for validating models. It also introduces the tools for model validation that are provided in SQL Server Data Tools and the Data Mining Client for Excel add-in.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain the need for model validation.
- Use the available accuracy charts to validate data mining models.

Overview of Data Mining Validation

Data mining algorithms use statistical models to predict the outcome of defined scenarios. However, if you use multiple algorithms to analyze a single data set, the results will not be identical. This is because each algorithm uses different techniques to analyze data. Consequently, when you analyze data using different algorithms, you need to test the results to discover which algorithm is the most accurate for the given scenario. This process is known as validation.

When you create a data model, you use a set of data called the training set to build the model. You also define a proportion of the data to use to test the model—this is known as the testing set. To test the validity of a data mining model, you create prediction queries to use against the testing set, and then use validation tools to compare the results of the mining model to this data. You can view the results of validation tests in accuracy charts, which compare predicted values with actual values.

Other factors can affect the validity of a data model, including the quality of the data and the usefulness of the results to the business.

Accuracy

Data mining models correlate a defined outcome, such as the amount of money a customer will spend, with input criteria, such as customer age, location, number of items purchased, and so on. The accuracy of these correlations can be affected by missing, approximate, or incorrect data. It is important to assess the accuracy of a model so you know how much you can trust the results produced. If you are aware in advance that a certain percentage of the data contains inaccuracies, you can decide to accept this degree of inaccuracy in the models you create. Data that qualifies as highly accurate is not necessarily reliable or useful. For example, you might test a model that predicts future sales for a store, based on previous performance. The model might be very accurate because the correlations are strong. However, if the method of calculating profitability used in the model is incorrect, the data produced is neither reliable nor useful.

- Validate data mining models to assess accuracy, reliability, and usefulness
- Accuracy:
 - Strength of correlations between inputs and predictable outcomes
 - Accurate data mining models are not always reliable or useful
- Reliability:
 - Consistency of the data mining model when used with multiple data sets
- Usefulness:
 - Does the data mining model produce information that is useful to the business?

Reliability

To be regarded as reliable, a data mining model must perform consistently. To assess reliability, you need to test the data model by using multiple, similar data sets. For each data set, the model should produce similar results. For example, you could test a model that predicts future sales performance for a store by using it against data from multiple stores. If one store calculates profitability differently, this will show up in the testing results.

Usefulness

A data model might be both accurate and reliable, but this does not necessarily mean that the results it produces are useful to the business. For example, your model might identify a strong correlation between the profitability of a store and its location, but this might not be useful because it doesn't explain why that correlation exists.

For more information about validation of data mining models, see the topic *Testing and Validation (Data Mining)* in the SQL Server 2016 Technical Documentation:



Testing and Validation (Data Mining)

<http://aka.ms/I516n8>

Accuracy Charts

Visual Studio, the Data Mining Add-In for Excel, and SQL Server Management Studio include tools for validating data mining models.

Visual Studio

You can use the Mining Accuracy Chart tab in the Data Mining Designer, in the SSDT, to create charts and reports to help you assess the validity of data models.

- Validation tools are available in:
 - Visual Studio
 - Data Mining Add-Ins for Excel
 - SQL Server Management Studio
- Lift chart: compare accuracy of different models
- Scatter plot: compare predicted results with actual results
- Profit chart: estimated profit gained by using a model
- Classification matrix: count true and false positives
- Cross-validation report: analyze data set by partitions

- **Lift chart.** This is a graphical representation of the difference between the results produced by a data mining model and those that random guessing would generate. The difference between these two sets of results is called the lift. You can compare multiple models simultaneously by using a lift chart, which means you can easily identify the best model. For example, if you are creating a mailing campaign, you can create a lift chart to identify the model that best predicts the customers most likely to respond. Typically, a lift chart is at its best when used with continuous data values.
- **Scatter plot.** This is a graphical representation of the accuracy of model predictions. Each point on the graph represents a prediction; predicted values are displayed along the Y axis, actual values are displayed along the X axis. The accuracy of each prediction can be determined from its proximity to the ideal prediction line, which can be visualized as a line running at 45 degrees between the X axis and the Y axis. For example, in a mailing campaign, in a model predicting sales by day, you could use a scatter plot to compare the predicted and actual values of sales. A scatter plot can only be used with continuous numeric values; it cannot be used with time series data.

- **Profit chart.** This is a graph that displays the estimated profit or loss identified by the model for a particular scenario. When you create a profit chart, you specify values for the number of rows to use in the assessment, the overall cost of the scenario, the cost per row, and the anticipated revenue per row. For example, in a mailing campaign, you could specify to use 5,000 customer rows, the total cost of the campaign, the cost per customer, and the revenue per customer. These values are used to calculate the campaign's profitability.
- **Classification matrix.** This uses testing data to count and display the number of true and false positives that a data mining model identifies for a given predicted value. For example, imagine a Customers table that includes a **BikeBuyer** column containing values of 1 and 0. The value 1 indicates that the customer purchased a bike, and 0 that they did not. A classification matrix will use the input columns to calculate whether a customer will be a bike buyer or not, and then compare these values to the real values in the data. The matrix will display the results, showing the number of correct and incorrect predictions for each value. Typically, a classification matrix is more accurate when used with discrete data values.
- **Cross-validation report.** Lift charts, profit charts, and classification matrices are all types of chart that assess and display the accuracy of models in different ways. A cross-validation report is a different way of validating a data mining model. To perform cross-validation, Analysis Services first divides a data set into partitions. It then uses a model to analyze one of the partitions, treating this as the training set. The other partitions are used to validate the results of this analysis. The process is then repeated several times, using a different partition as the training set on each occasion.

SQL Server Management Studio

You can use SQL Server Management Studio to connect to an Analysis Services database containing a data mining structure, and perform the following validation tasks for models:

- Create a lift chart.
- Create a scatter plot.
- Create a profit chart.
- Create a classification matrix.
- Create a cross-validation report.

Data Mining Client for Excel

You can use the Data Mining Client for Excel to validate data mining models using a set of wizards, which you can access from the **Accuracy and Validation** area of the **Data Mining** tab on the ribbon.

- The **Accuracy Chart Wizard** enables you to create a lift chart or a scatter plot.
- The **Classification Matrix Wizard** enables you to create a classification matrix.
- The **Profit Chart Wizard** enables you to generate a profit chart.
- The **Cross-Validation Wizard** enables you to perform cross-validation.

For more information about using the Data Mining Client for Excel to carry out data model validation, see the *Test, Query, and Validate Models* section of the topic *Data Mining Client for Excel (SQL Server Data Mining Add-ins)* in the SQL Server 2016 Technical Documentation:

 **Data Mining Client for Excel (SQL Server Data Mining Add-ins) - Test, Query, and Validate Models**

http://aka.ms/Dlgmem#Anchor_2

Demonstration: Viewing Accuracy Charts

In this demonstration, you will see how to view:

- A Lift Chart.
- A Profit Chart.
- A Classification Matrix.
- A Cross-Validation Report.

Demonstration Steps

Create a Lift Chart

- Ensure you have completed the previous demonstrations in this module.

In SSMS Object Explorer, right-click the **Purchase Prediction** mining structure and click **View Lift Chart**.

On the **Input Selection** tab, note that both mining models are selected with the **Bike Buyer** column as predictable, and that the test cases defined in the models themselves will be used for the validation.

Click the **Lift Chart** tab, and view the lift chart, which compares accuracy for the two mining models you have created against an ideal model by plotting the number of correct predictions against the number of cases in the overall sample. As the number of cases increases, the ideal model maintains an accuracy of 100 percent. However, the models you have created tend to become less accurate the more cases there are.

Review the scores in the Mining Legend pane to see which of your models is the most accurate for this test data.

Create a Profit Chart

- In the **Chart type** drop-down list, select **Profit Chart**.

In the **Profit Chart Settings** dialog box, enter the following values to reflect a marketing campaign you are planning, and then click **OK**:

- **Population:** 20,000 (this is the number of potential customers you plan to contact).
- **Fixed cost:** 1,000 (this is the fixed cost of your marketing campaign).
- **Individual cost:** 3 (this is the cost associated with contacting each customer).
- **Revenue per individual:** 10 (this is the amount you expect a customer to spend if they respond positively to the campaign).

Review the chart and the **Mining Legend** pane to evaluate which mining model is likely to generate the most profitable marketing campaign based on the test data.

Create a Classification Matrix

- Click the **Classification Matrix** tab.

Review the matrix, noting that for each model it shows the number of times the model predicted a **Bike Buyer** value of **1** or **0** on rows, with columns for the actual value of the **Bike Buyer** column in the test data.

Create a Cross Validation Report

- Click the **Cross Validation** tab.

Enter the following values, and click **Get Results**:

- **Fold Count:** 5 (this is the number of partitions used to group the data for analysis).
- **Max Cases:** 5 (this is the number of cases to be analyzed).
- **Target Attribute:** Bike Buyer (this is the predictable column to be evaluated).
- **Target State:** 1 (this is the desired value for the target attribute).
- **Target Threshold:** 0.1 (this is a value between 0 and 1 that indicates the level of accuracy required for a prediction to be considered correct).

View the resulting report, and note that for each mining model, the results include the following:

- Classifications for true positives, false positives, true negatives, and false negatives.
- The likely lift gained by using the model.

Close SQL Server Management Studio without saving any changes.

Check Your Knowledge

Question	
Which data model accuracy chart type would you use to compare the accuracy of several data models?	
Select the correct answer.	
<input type="checkbox"/>	Scatter plot
<input type="checkbox"/>	Classification matrix
<input type="checkbox"/>	Lift chart
<input type="checkbox"/>	Cross-validation report
<input type="checkbox"/>	Profit chart

Lesson 4

Connecting to and Consuming a Data Mining Model

Results produced by data mining models need to be displayed in useful and comprehensible ways, for the benefit of data and business analysts.

This lesson describes the options for viewing data mining results and how you can use Reporting Services reports to render them in a user-friendly format.

Lesson Objectives

After completing this lesson, you will be able to:

- Use Reporting Services to display data mining results.

Creating a Connection to a Data Mining Model from a Reporting Services Report

You have seen how to use DMX queries to generate predictive information from a data mining model. To provide a way of viewing data mining results that is easier for business users to consume, you can create a SQL Server Reporting Services (SSRS) report that uses DMX to query the data mining mode and provide predictive information. SSRS reports offer a wide range of display and formatting options that enable you to show results in the most intuitive way. For convenience, you can also incorporate the results of multiple models into a single report.

Data mining models store data in an Analysis Services database. To connect to an Analysis Services database and create a data mining report, you create an Analysis Services connection in an SSRS project, in SSDT. You can then use Query Builder to create the query that retrieves the data to use in the report.

For more information about using DMX to query data models in an SSRS report, see the topic *Retrieve Data from a Data Mining Model (DMX) (SSRS)* in the SQL Server 2016 Technical Documentation:



Retrieve Data from a Data Mining Model (DMX) (SSRS)

<http://aka.ms/E2h6fz>

- Make data mining results accessible to non-technical users
- Wide range of options for formatting and displaying data
- Include the results from multiple models in a single report

Demonstration: Using Data Mining Results in an SSRS Report

In this demonstration, you will see how to include data mining results in an SSRS report.

Demonstration Steps

- Ensure you have completed the previous demonstrations in this module.

In Visual Studio, on the **File** menu, point to **New** and click **Project**.

In the **New Project** dialog box, expand **Business Intelligence**, select **Reporting Services**, select **Report Server Project Wizard**. In the **Name** field, type **Purchase Prediction**, in the **Location** field, type **D:\Demofiles\Mod09**, and then click **OK**.

In the **Report Wizard**, on the Welcome to the Report Wizard page, click **Next**.

On the Select the Data Source page, select **New data source**, change the **Name** to **AWMine**, in the **Type** drop-down list, select **Microsoft SQL Server Analysis Services**, and click **Edit**.

In the **Connection Properties** dialog box, in the **Server name** field, type **MIA-SQL**, in the **Select or enter a database name** drop-down list, click **Mine AW**, and then click **OK**.

On the Select the Data Source page, click **Next**.

On the Design the Query page, click **Query Builder**.

In **Query Designer**, in the Mining Model pane, click **Select Model**. Then in the **Select Mining Model** dialog box, expand **Purchase Prediction**, click **Purchase Naive Bayes**, and click **OK**.

In the Select Input Table(s) pane, click **Select Case Table**. Then in the **Select Table** dialog box, click **ProspectiveBuyer (dbo)**, and click **OK**.

Under the Mining Model pane, in the **Source** column, select **ProspectiveBuyer table**, and then in the **Field** column, select **EmailAddress**.

Under the row you just added, in the **Source** column, click **Purchase Naive Bayes mining model**, in the **Field** column, click **Bike Buyer**, and in the **Criteria/Argument** column, type **=1**.

Under the row you just added, in the **Source** column, click **Prediction Function**, in the **Field** column, click **PredictProbability**, in the **Alias** column, type **Purchase Probability**, and then drag the Bike Buyer column from the Purchase Naive Bayes model in the Mining Model pane to the **Criteria/Argument** column so it contains the value **[Purchase Naive Bayes].[Bike Buyer]**, then click **OK**.

On the Design the Query page, click **Next**.

On the Select the Report Type page, verify that **Tabular** is selected, then click **Next**.

On the Design the Table page, in the **Available fields** box, click **EmailAddress**, then click **Details >**, then in the **Available fields** box, click **Purchase_Probability**, then click **Details >**. Click **Next**.

If the Choose the Table Style page appears, click **Next**.

If the Choose the Deployment Location page appears, click **Next**.

On the Completing the Wizard page, type **Likely Purchase Email Report** in the **Report name** box, then click **Finish**.

In the report designer, on the **Design** tab, click the table.

In the Row Groups pane, right-click the **(table1_Details_Group)** drop-down list and click **Group Properties**.

On the Sorting page of the **Group Properties** dialog box, click **Add**, in the **Sort by** drop-down list, select **[Purchase_Probability]**, and in the Order drop-down list, click **Z to A**. Then click **OK**.

In the **Purchase Probability** column, right-click **[Purchase Probability]** and click **Text Box Properties**. Then in the **Text Box Properties** dialog box, click **Number**, select **Percentage**, and click **OK**.

Click the **Preview** tab to review the report.

Close Visual Studio.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? You can include the results of multiple mining models in a single SSRS report.	

Lab: Using Data Mining to Support a Marketing Campaign

Scenario

The marketing department at Adventure Works Cycles is planning a direct mail campaign. To maximize its effectiveness, you have been asked to create a report that uses data mining techniques to identify the subset of potential customers who are most likely to purchase a bike.

Objectives

After completing this lab, you will be able to:

- Create a data mining structure and a data mining model, then add a data mining model to a data mining structure.
- Explore a data mining model.
- Validate data mining models.
- Use a data mining model as a data source for a report.

Estimated Time: 60 minutes

Virtual machine: **20768B-MIA-SQL**

User name: **ADVENTURWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Creating a Data Mining Structure and Models

Scenario

To support the marketing campaign, you want to create a data mining model that will help you to identify the individuals most likely to buy a bike. You will use SQL Server Data Tools to create the structure and model, and then deploy the model to Analysis Services. When this is complete, you will add a second data model to the structure.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create a Data Mining Project
3. Create a Data Mining Structure and a Data Mining Model
4. Add a Data Mining Model to a Data Mining Structure

► Task 1: Prepare the Lab Environment

- Start the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.

Run **Setup.cmd** in the D:\Labfiles\Lab09\Starter folder as Administrator.

► Task 2: Create a Data Mining Project

- Use Visual Studio to create an Analysis Services multidimensional and data mining project named **AW Data Mining**. Save the project in the D:\Labfiles\Lab09\Starter folder.

Create a data source named **Adventure Works DW** that connects to the **AdventureWorksDW** database on the **MIA-SQL** Database Engine instance. Use the Service Account.

Use the **Adventure Works DW** data source to create a data source view named **Adventure Works DW DM View** that includes the **ProspectiveBuyer** table and the **vTargetMail** view.

► Task 3: Create a Data Mining Structure and a Data Mining Model

- Use the Data Mining Wizard to create a data mining structure from existing data that uses the **Microsoft Decision Trees** algorithm.

Specify the **vTargetMail** view from the **Adventure Works DW DM View** data view as the case table.

Configure the columns in the case table as follows:

- **BikeBuyer**: Predictable
- **CustomerKey**: Key

All other columns: Input.

Have the wizard detect the content and data types of the columns. Ensure that the content type of the **Bike Buyer** column is identified as **Discrete**, and change the **Yearly Income** column to **Discrete**.

Create a testing set with 30 percent of the data.

Name the data mining structure **Purchase Prediction**, and name the data mining model **Purchase Decision Tree**.

Deploy the completed model to the **MIA-SQL** Analysis Services instance.

► Task 4: Add a Data Mining Model to a Data Mining Structure

- Use Visual Studio to add a new mining model to the **Purchase Prediction** mining structure. The model should use the **Microsoft Naive Bayes** mining algorithm.

Call the model **Purchase – Bayes**, and amend the model definition to ignore the **Name Style** column.

Deploy the completed model to the **MIA-SQL** Analysis Services instance.

Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have a data mining structure with two data models.

Exercise 2: Explore a Data Mining Model

Scenario

Now that the data mining structure has been deployed, you will use the Visual Studio Mining Model Viewer to explore and review data models.

The main tasks for this exercise are as follows:

1. Review Data Models

► Task 1: Review Data Models

- In Visual Studio, use the Mining Model Viewer to explore the output of the **Purchase Decision Tree** and **Purchase – Bayes** models in the **Purchase Prediction** mining structure. Notice that different viewers are available for data models based on different data mining algorithms.

Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have reviewed the output of data mining models using Visual Studio.

Exercise 3: Validating Data Mining Models

Scenario

Now you have created the data mining models, you want to validate them to see which is the most accurate for your data. To do this, you will use Visual Studio.

The main tasks for this exercise are as follows:

1. Review a Lift Chart
2. Review a Profit Chart
3. Review a Classification Matrix
4. Review a Cross-Validation Report

► Task 1: Review a Lift Chart

- In Visual Studio, review a lift chart for the data models in the **Purchase Prediction** mining structure. Which of the models shows greater accuracy?

► Task 2: Review a Profit Chart

- Review a profit chart for the data models in the **Purchase Prediction** mining structure. Use the following values to configure your profit chart:
 - **Population:** 5,000
 - **Fixed cost:** 500
 - **Individual cost:** 1
 - **Revenue per individual:** 150

► Task 3: Review a Classification Matrix

- Review a classification matrix for the data models in the **Purchase Prediction** mining structure.

► Task 4: Review a Cross-Validation Report

- Review a cross-validation report for the data models in the **Purchase Prediction** mining structure. Use the following values to configure your cross-validation matrix:
 - **Fold Count:** 5
 - **Max Cases:** 500
 - **Target Attribute:** Bike buyer
 - **Target State:** 1
 - **Target Threshold:** 0.7

Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have two validated data mining models in your data mining structure.

Exercise 4: Using a Data Mining Model in a Report

Scenario

Now you have validated the data mining models, you want to create a report that contains the list of potential bike purchasers, listed from “most likely to purchase” to “least likely”. You will create a Reporting Services report that uses data from the AW Data Mining database, and then format the report.

The main tasks for this exercise are as follows:

1. Create a Report

► Task 1: Create a Report

1. Use Visual Studio to create a new **Report Server Project Wizard** project named **Promotion Targeting** in the D:\Labfiles\Lab09\Starter folder.
2. Create a new data source for the **AW Data Mining** database in the **MIA-SQL** instance of Analysis Services.
3. Use the query builder to create a query that uses the **Purchase - Bayes** model and the **ProspectiveBuyer(dbo)** case table. Configure the query to return the following fields:

Source	Field	Alias	Criteria/Argument
ProspectiveBuyer table	LastName		
ProspectiveBuyer table	Address Line 1		
ProspectiveBuyer table	City		
Purchase – Bayes mining model	Bike Buyer		=1
Prediction Function	PredictProbability	Purchase Probability	[Purchase - Bayes].[Bike Buyer]

4. Create a tabular report with all fields in the details section, format it using any style, and name it **Potential Bike Buyers**.
5. Sort the report from Z to A by the **Purchase Probability** column and change the format of the **Purchase Probability** column to percentage.
6. Preview the **Potential Bike Buyers** report, and modify its formatting until you are happy with it.
7. When you have finished working on the report, close Visual Studio.

Results: After this exercise, you should have an SSRS report that predicts bike purchasers.

Check Your Knowledge

Question	
Which type of data mining algorithm would you use to identify sequences in your data?	
Select the correct answer.	
<input type="checkbox"/>	A regression algorithm, such as the Microsoft Time Series algorithm.
<input type="checkbox"/>	A sequence analysis algorithm, such as the Microsoft Sequence Clustering algorithm.
<input type="checkbox"/>	A classification algorithm, such as the Microsoft Decision Trees algorithm.
<input type="checkbox"/>	A clustering algorithm, such as the Microsoft Clustering algorithm.
<input type="checkbox"/>	An association algorithm, such as the Microsoft Association algorithm.

Module Review and Takeaways

In this module, you have learned how to use the data mining capability of SQL Server Analysis Services for predictive analysis.

Review Question(s)

Question: How have you seen, or can you envisage, data mining being used in organizations where you have worked?

Course Evaluation

Course Evaluation

- Your evaluation of this course will help Microsoft understand the quality of your learning experience.
- Please work with your training provider to access the course evaluation form.
- Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

Your evaluation of this course will help Microsoft understand the quality of your learning experience.

Please work with your training provider to access the course evaluation form.

Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 1: Introduction to Business Intelligence and Data Modeling

Lab: Exploring a Business Intelligence Solution

Exercise 1: Exploring a Data Warehouse

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab01\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.

► Task 2: Explore a Data Warehouse Schema

1. Start SQL Server Management Studio and connect to the **MIA-SQL** instance of the database engine. Use Windows® authentication.
2. In Object Explorer, expand **Databases**, and then expand **AdventureWorksDW**.
3. Right-click **Database Diagrams**, and then click **New Database Diagram**.
4. In the **Add Table** dialog box, click **DimAccount**, hold down Shift and click **FactSurveyResponse**, click **Add**, and then click **Close**.
5. Explore the diagram, noting the following details:
 - The **FactInternetSales** table stores details of sales orders that are made through the Adventure Works website. The table is related to the **DimCustomer** table, which contains details of the customers who have placed orders.
 - The **FactResellerSales** table stores details of sales orders that are made to resellers. The table is related to the **DimReseller** table, which contains details of the resellers who have placed orders.
 - The **DimDate** table stores values for dates, including details of the calendar and fiscal periods in which individual dates occur.
 - The **FactInternetSales** and **FactResellerSales** tables are related to the **DimDate** table by multiple fields. These fields represent the order date (the date when the order was placed), the due date (the date when the order was expected to be in stock), and the ship date (the date when the order was shipped to the customer).
 - Both the **DimCustomer** and **DimReseller** tables are related to the **DimGeography** table, which stores details of geographical locations. The **DimSalesTerritory** table is also related to **DimGeography**.
 - The **DimProduct** table contains details of products, and is related to the **DimProductSubcategory** table, which is in turn related to the **DimProductCategory** table.
 - Many tables include multiple language values for the same data value, for example, the **DimDate** table stores the English, French, and Spanish words for each month name.

- The **DimEmployee** table is related to itself to represent the fact that each employee has a manager, who is also an employee.
6. On the **File** menu, click **Save Diagram_0**, and then save the diagram as **AdventureWorksDW Schema**.
- **Task 3: Query a Data Warehouse**
1. Click **File**, click **Open**, click **File**, browse to the D:\Labfiles\Lab01\Starter folder, and then double-click **Query DW.sql**.
 2. Select the Transact-SQL code under the comment **Internet sales by year and month**, and then click **Execute**.
 3. View the results of the query, and note the following details:
 - The data warehouse contains historical Internet sales orders from July 2005 to July 2008. Reseller sales in the data warehouse are also recorded for this time period.
 - You can use the **MonthNumberOfYear** column in the **DimDate** table to sort month names into chronological order—without this field, it would be difficult (though not impossible) for reporting clients to sort months other than alphabetically. A similar field named **DayNumberOfWeek** can be used to sort weekday names into chronological order.
 4. Select the Transact-SQL code under the comment **Geographical reseller sales**, and then click **Execute**.
 5. View the results of the query, and note the following details:
 - In 2005, Adventure Works only sold to resellers in the United States and Canada.
 - In 2006, this was expanded to include France and the United Kingdom.
 - In 2007, resellers in Australia and Germany were added.
 - By contrast, Adventure Works has sold directly to Internet customers in all of these regions since 2005.
 6. Select the Transact-SQL code under the comment **Sales by product category**, and then click **Execute**.
 7. View the results of the query, and note the following details:
 - Adventure Works sells four categories of product: Accessories, Bikes, Clothing, and Components.
 - Components are only sold to resellers, not to Internet customers.
 - Accessories were not sold to Internet customers until 2007.

Results: After completing this exercise, you will have:

Used a database diagram to explore a data warehouse schema.

Used Transact-SQL queries to explore the data in a data warehouse.

Exercise 2: Exploring a Data Model

► Task 1: View a SQL Server Analysis Services Database

1. In SQL Server Management Studio, in Object Explorer, click **Connect**, and then click **Analysis Services**.
2. In the **Connect to Server** dialog box, in the **Server name** field, type **MIA-SQL**, and then click **Connect**.
3. In Object Explorer, under the **MIA-SQL** Microsoft Analysis Server node, expand **Databases**, and then expand **Adventure Works OLAP**.
4. Under the **Adventure Works OLAP** database, expand **Data Sources**. Note that the SQL Server Analysis Services database includes a data source called **Adventure Works Data Warehouse**, which connects to the **AdventureWorksDW** database that you explored in the previous exercise.
5. Under the **Adventure Works OLAP** database, expand **Cubes**. Right-click **Internet Sales**, and then click **Browse**.
6. In the cube browser window, in the Metadata pane, expand **Measures**, expand **Internet Sales Order**, and then drag the **Sales Amount** measure onto the empty query pane. The total Internet sales amount is displayed.
7. In the Metadata pane, expand **Order Date**, and then drag **Order Date.Fiscal Year** to the query pane. The sales revenue is now aggregated by fiscal year. In Adventure Works, fiscal years run from July to June, so fiscal year 2006 represents the period from July 2005 to June 2006.
8. Close SQL Server Management Studio without saving any files.

► Task 2: Create a PivotTable in Excel

1. On the taskbar, click **Excel 2016**, and then create a new blank workbook.
2. On the **Data** tab, in the **Get External Data** area or list, in the **From Other Sources** list, click **From Analysis Services**.
3. In the **Data Connection Wizard** dialog box, in the **Server name** box, type **MIA-SQL**, ensure that **Use Windows Authentication** is selected, and then click **Next**.
4. On the **Select Database and Table** page, in the **Select the database that contains the data you want** drop-down list, select the **Adventure Works OLAP** database, in the **Connect to a specific cube or table** field, click **Internet Sales**, and then click **Next**.
5. On the **Save Data Connection File and Finish** page, click **Finish**.
6. In the **Import Data** dialog box, select **PivotTable Report**, ensure that **Existing worksheet** is selected with the cell reference =**\$A\$1**, and then click **OK**.
7. In the PivotTable Fields pane, under **Internet Sales Order**, select **Sales Amount**. The total revenue from Internet sales is displayed in the PivotTable on the worksheet.
8. In the PivotTable Fields pane, in the **Values** area, in the drop-down list for the **Sales Amount** field, click **Value Field Settings**.
9. In the **Value Field Settings** dialog box, click **Number Format**.
10. In the **Format Cells** dialog box, select **Accounting**, click **OK**, and then, in the **Value Field Settings** dialog box, click **OK**. The Internet sales total in the PivotTable on the worksheet is formatted as a currency value.

11. In the PivotTable Fields pane, under **Order Date**, select **Order Date.Calendar Date**. The total calendar years for the sales are displayed as columns.
12. In the PivotTable Fields pane, drag **Order Date.Calendar Date** from the **Columns** area to the **Rows** area. The years are now displayed on rows.
13. In the PivotTable, expand **2006**. The calendar semesters for the year are shown. Expand **H1CY2006** to reveal the first two quarters, expand **Q1 CY 2006** to reveal the first three months, and then expand **January** to reveal daily sales totals in January 2006.
14. Save the workbook as **Sales.xlsx** in the D:\Labfiles\Lab01\Starter folder.

► **Task 3: Filter the PivotTable**

1. In Excel, in the **Sales.xlsx** workbook, click any cell in the PivotTable on the **Sheet1** worksheet. On the ribbon, on the **Analyze** tab, click **Insert Slicer**.
2. In the **Insert Slicers** dialog box, under **Product**, in the **Product Category** hierarchy, select **English Product Category Name**, and then click **OK**.
3. Move the slicer so that you can see the PivotTable. Note that the **Components** category is disabled because there are no sales of components to Internet customers.
4. In the slicer, click **Accessories**. The PivotTable is filtered to show only sales of accessories. Note that no accessories were sold in 2005 or 2006.
5. In the slicer, click the **Clear Filter** icon to remove the filter.
6. Save the workbook, and then close Excel.

Results: After completing this exercise, you will have used Excel to explore an analytical data model built on the data warehouse.

Module 2: Creating Multidimensional Databases

Lab: Creating a Multidimensional Database

Exercise 1: Creating a Data Source

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab02\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish. Note that the script may take up to several minutes to complete.

► Task 2: Create an Analysis Services Project

1. On the taskbar, click **Visual Studio 2015**.
2. In Visual Studio, under **Start**, click **New Project**, in the **New Project** dialog box, in the **Templates** list, click **Business Intelligence**, in the main window, click **Analysis Services Multidimensional and Data Mining Project**, and then in the **Name** box, type **Adventure Works OLAP**.
3. Click **Browse**, browse to the D:\Labfiles\Lab02\Starter folder, click **Select Folder**, and then click **OK**.

► Task 3: Create a Data Source

1. In Solution Explorer, right-click the **Data Sources** folder, and then click **New Data Source**.
2. In the **Data Source Wizard**, on the **Welcome to the Data Source Wizard** page, click **Next**.
3. On the **Select how to define the connection** page, click **New**.
4. In the **Connection Manager** dialog box, in the **Server name** box, type **localhost**.
5. In the **Log on to the server** area, ensure that **Use Windows Authentication** is selected.
6. In the **Connect to a database** area, in the **Select or enter a database name** box, click **AdventureWorksDW**, and then click **OK**.
7. On the **Select how to define the connection** page, click **Next**.
8. On the **Impersonation Information** page, select **Use a specific Windows user name and password**.
9. In the **User name** box, type **ADVENTUREWORKS\ServiceAcct**.
10. In the **Password** box, type **Pa\$\$w0rd**, and then click **Next**.
11. On the **Completing the Wizard** page, in the **Data source name** field, change the value to **Adventure Works Data Warehouse**, and then click **Finish**.

Results: After completing this exercise, you will have:

Created an Analysis Services project by using SQL Server Data Tools.

Created a data source.

Exercise 2: Creating and Configuring a Data Source View

► Task 1: Create a Data Source View

1. In Solution Explorer, right-click the **Data Source Views** folder, and then click **New Data Source View**.
2. On the **Welcome to the Data Source View Wizard** page, click **Next**.
3. On the **Select a Data Source** page, verify that the **Adventure Works Data Warehouse** data source is selected, and then click **Next**.
4. In the **Available objects** list, click **DimCustomer (dbo)**, and then hold down the Ctrl key and click **DimDate (dbo)**, **DimGeography (dbo)**, **DimProduct (dbo)**, **DimProductCategory (dbo)**, **DimProductSubcategory (dbo)**, and **FactInternetSales (dbo)**. Click the right arrow (>) button to add the selected tables to the **Included objects** list, and then click **Next**.
5. On the **Completing the Wizard** page, in the **Name** field, type **Adventure Works DSV**, and then click **Finish**. The Data Source View Designer opens automatically.

► Task 2: Configure a Data Source View

1. In the diagram, click the title bar of the **FactInternetSales** table and press F4.
2. In the Properties pane, change the **FriendlyName** property to **Internet Sales**, and then press Enter.
3. Click the **DimCustomer** table, change its **FriendlyName** property to **Customer**, and then press Enter.
4. Repeat the previous step to edit the **FriendlyName** property of the following tables (**Table name: FriendlyName**):
 - **DimDate: Date**
 - **DimGeography: Geography**
 - **DimProduct: Product**
 - **DimProductCategory: Product Category**
 - **DimProductSubcategory:**
 - **Product Subcategory**
5. In the diagram, right-click the **Customer** table, and then click **New Named Calculation**.
6. In the **Create Named Calculation** dialog box, in the **Column** name box, type **Full Name**.
7. In the **Expression** box, type the text in the following code example:

```
CASE
WHEN MiddleName IS NULL THEN
  FirstName + ' ' + LastName
ELSE
  FirstName + ' ' + MiddleName + ' ' + LastName
END
```

8. In the **Create Named Calculation** dialog box, click **OK**.
9. On the **File** menu, click **Save All**.

Results: After completing this exercise, you will have:

Created a data source view.

Configured a data source view.

Exercise 3: Creating and Configuring a Cube

► Task 1: Create a Cube

1. In Solution Explorer, right-click the **Cubes** folder, and then click **New Cube**.
2. On the **Welcome to the Cube Wizard** page, click **Next**.
3. On the **Select Creation Method** page, verify that **Use existing tables** is selected, and then click **Next**.
4. On the **Select Measure Group Tables** page, click **Suggest**.
5. Note that the wizard selects the **Internet Sales** table as the measure group table, and then click **Next**.
6. On the **Select Measures** page, clear every check box except **Order Quantity**, **Total Product Cost**, **Sales Amount**, and **Internet Sales Count**, and then click **Next**.
7. On the **Select New Dimensions** page, clear the **Internet Sales** check box, and then click **Next**.
8. On the **Completing the Wizard** page, change the **Cube name** to **Sales**, and then click **Finish**. The Cube Designer opens automatically.

► Task 2: Configure Measures

1. In Solution Explorer, click **Sales.cube**. In the Measures pane of the Cube Designer, expand the **Internet Sales** measure group, right-click the **Order Quantity** measure, and then click **Rename**.
2. Rename **Order Quantity to Internet Order Quantity**, and then press Enter.
3. In the Measures pane, right-click the **Total Product Cost** measure, and then click **Rename**.
4. Rename **Total Product Cost** to **Internet Cost**, and then press Enter.
5. In the Measures pane, right-click **Sales Amount**, and then click **Rename**.
6. Rename **Sales Amount** to **Internet Revenue**, and then press Enter.
7. On the **File** menu, click **Save All**.

► Task 3: Configure Dimensions

1. In Solution Explorer, under **Dimensions**, right-click **Customer.dim**, and then click **View Designer**.
2. In the Data Source View pane, in the **Geography** table, click **City**.
3. Hold down the Ctrl key, click **StateProvinceName**, **EnglishCountryRegionName**, and **PostalCode**, and then drag the selected columns to the Attributes pane.
4. In the Data Source View pane, in the **Customer** table, click **CustomerAlternateKey**.
5. Hold down the Ctrl key, click **Title**, **FirstName**, **MiddleName**, **LastName**, and **Full Name**.
6. Drag the selected columns to the Attributes pane.
7. On the **File** menu, click **Save All** then close the **Customer.dim** Dimension Designer.
8. In Solution Explorer, under **Dimensions**, right-click **Product.dim**, and then click **View Designer**.
9. In the Data Source View pane, in the **Product** table, click **ProductAlternateKey**. Hold down the Ctrl key, click **EnglishProductName** and **ListPrice**, and then drag the selected columns to the Attributes pane.
10. In the Data Source View pane, in the **Product Subcategory** table, click **EnglishProductSubcategoryName**, and drag the selected column to the Attributes pane.

11. In the Data Source View pane, in the **Product Category** table, click **EnglishProductCategoryName**, and drag the selected column to the Attributes pane.
12. On the **File** menu, click **Save All** then close the **Product.dim** Dimension Designer.
13. In Solution Explorer, under **Dimensions**, right-click **Date.dim**, and then click **View Designer**.
14. In the Data Source View pane, in the **Date** table, click **FullDateAlternateKey**. Hold down the Ctrl key, click **EnglishMonthName**, **MonthNumberOfYear**, **CalendarQuarter**, **CalendarYear**, and **CalendarSemester** and drag the selected columns to the Attributes pane.
15. In the Attributes pane, right-click **English Month Name** and click **Rename**. Change the attribute name to **Month**.
16. On the **File** menu, click **Save All**, and then close the **Date.dim** designer window.

► **Task 4: Browse the Cube**

1. In Solution Explorer, right-click the **Adventure Works OLAP** solution, and then click **Deploy**. If an **Account Password** dialog box appears, in the **Password** field type **Pa\$\$w0rd**, and then click **OK**. Wait for the **Deploy Completed Successfully** message in the **Deployment Progress – Adventure Works OLAP** dialog box, and then close the **Deployment Progress – Adventure Works OLAP** dialog box.
2. In the **Sales.cube** Cube Designer, click the **Browser** tab.

Tip: Click the **Auto Hide** icon on the various Visual Studio panes to make it easier to see the entire Cube Browser window.
3. In the Measure Group pane, expand **Measures**, expand **Internet Sales**, and then drag the **Internet Revenue** measure to the **Drag levels or measures here to add to the query** area.
4. In the Measure Group pane, drag **Internet Sales Count** to the right of the **Internet Revenue** column.
5. In the Measure Group pane, expand the **Order Date** dimension, and drag the **Order Date.Calendar Year** attribute to the left of the **Internet Revenue** column. The Cube Browser shows historical sales amounts and counts for multiple years.
6. In Visual Studio, on the **File** menu, click **Save All**.
7. Keep Visual Studio open for the next exercise.

Results: After completing this exercise, you will have:

Created a cube.

Edited measures in the cube.

Edited dimensions in the cube.

Browsed the cube.

Exercise 4: Adding a Dimension to the Cube

► Task 1: Edit the Data Source View

1. In Visual Studio, in the **Adventure Works OLAP** project, in Solution Explorer, under **Data Source Views**, double-click **Adventure Works DSV.dsv** to open it. In the **Data Source View** menu, click **Add/Remove Tables**.
2. In the **Add/Remove Tables** dialog box, in the **Available objects** list, click **DimSalesTerritory**, click the right arrow (>) button, and then click **OK**.
3. In the diagram, click the title bar of the **DimSalesTerritory** table and press F4.
4. In the Properties pane, change the **FriendlyName** property to **Sales Territory**, and then press Enter.
5. On the **File** menu, click **Save All**.

► Task 2: Create a Dimension

1. In Solution Explorer, right-click the **Dimensions** folder and click **New Dimension**.
2. In the Dimension Wizard, on the **Welcome to the Dimension Wizard** page, click **Next**.
3. On the **Select Creation Method** page, select **Use an existing table**, and click **Next**.
4. On the **Specify Source Information** page, ensure that the **Adventure Works DSV** data source view is selected. In the **Main table** list, select **Sales Territory**. Verify that **SalesTerritoryKey** is selected as the key column and the name column, and then click **Next**.
5. On the **Select Dimension Attributes** page, select **SalesTerritoryRegion**, **SalesTerritoryCountry**, and **SalesTerritoryGroup**, ensure that the **Enable Browsing** check box is selected for each of them, and then click **Next**.
6. On the **Completing the Wizard** page, click **Finish**. The new dimension is opened in the Dimension Designer.

► Task 3: Add a Dimension to a Cube

1. In Solution Explorer, under **Cubes**, double-click **Sales.cube** to open it in the Cube Designer, and then click the **Cube Structure** tab.
2. On the **Cube** menu, click **Add Cube Dimension**.
3. In the **Add Cube Dimension** dialog box, click **Sales Territory**, and then click **OK**.
4. Click the **Dimension Usage** tab, verify that the value **Sales Territory** appears in the **Dimensions** list, and then next to it, in the **Measure Groups** list, the value **Sales Territory Key** appears. This confirms that the **Sales Territory** dimension is related to the **Internet Sales** measure group using the **Sales Territory Key** attribute.
5. On the **File** menu, click **Save All**.

► Task 4: Analyze a Cube by Using Excel

1. In Solution Explorer, right-click the **Adventure Works OLAP** solution, and then click **Deploy**. If an **Account Password** dialog box appears, in the **Password** field type **Pa\$\$w0rd** and click **OK**. Wait for the **Deploy Completed Successfully** message in the **Deployment Progress – Adventure Works OLAP** dialog box, and then close the **Deployment Progress – Adventure Works OLAP** dialog box.
2. When deployment has completed successfully, in the **Sales.cube** Cube Designer, on the **Browser** tab, click **Reconnect**.
3. On the **Cube** menu, click **Analyze in Excel**. If you are prompted to enable data connections, click **Enable**.

4. In the PivotTable Fields pane, beneath the **Internet Sales** measure, select **Internet Revenue**.
5. In the PivotTable Fields pane, under the **Sales Territory** dimension, select **Sales Territory Group**. Sales for each territory group are shown.
6. In the PivotTable Fields pane, under the **Sales Territory** dimension, select **Sales Territory Country**. Sales within each sales territory group are broken down for each country.
7. In the PivotTable Fields pane, under the **Sales Territory** dimension, select **Sales Territory Region**. Sales within each sales territory are broken down for each region. Note that some countries do not include sales regions, so the country is shown at both the country and region levels.
8. Close Excel without saving the workbook.
9. In Visual Studio, on the **File** menu, click **Save All**, and then close Visual Studio.

Results: After completing this exercise, you will have:

Edited the data source view.

Created a dimension.

Added a dimension to the cube.

Browsed the cube by using Excel.

Module 3: Working with Cubes and Dimensions

Lab: Working with Cubes and Dimensions

Exercise 1: Configuring Dimensions

► Task 1: Prepare the Lab Environment

1. Ensure the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab03\Starter folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog, click **Yes**, and wait for the script to finish.

► Task 2: Remove Unused Attributes

1. Start Visual Studio and open **Adventure Works OLAP.sln** in the D:\Labfiles\Lab03\Starter folder.
2. In Solution Explorer, expand **Dimensions** and double-click the **Customer.dim** dimension. Notice that many attributes have been added to allow business users to aggregate measures in many different ways. However, users have complained that some of these attributes are unnecessary and that they should be removed to make browsing the cube simpler.
3. In the Attributes pane, click **Commute Distance**, press the Ctrl key, and click **Number Cars Owned** and **Number Children At Home**, right-click any of the highlighted attributes, and then click **Delete**. In the **Delete Objects** dialog box, click **OK**.
4. On the **File** menu, click **Save All**, then close the **Customer.dim** Dimension Designer.
5. In Solution Explorer, double-click the **Product.dim** dimension. Again, users have requested that you remove some unnecessary attributes from this dimension.
6. In the Attributes pane, click **Days To Manufacture**, press the Ctrl key, and click **Safety Stock Level**. Press Delete and, in the **Delete Objects** dialog box, click **OK**.
7. On the **File** menu, click **Save All**, then close the **Product.dim** Dimension Designer.

► Task 3: Add Dimension Intelligence

1. In Solution Explorer, right-click **Date.dim**, and then click **Add Business Intelligence**.
2. On the **Welcome to the Business Intelligence Wizard** page, click **Next**.
3. On the **Choose Enhancement** page, click **Define dimension intelligence**, and then click **Next**.
4. On the **Define Dimension Intelligence** page, in the **Dimension type** field, click **Time**.
5. In the **Dimension attributes** table, select the **Include** check box for the following attribute types, and select the corresponding item in the **Dimension Attribute** column:
 - **Year**: Calendar Year
 - **Half Year**: Calendar Semester
 - **Quarter**: Calendar Quarter
 - **Month**: Full Date Alternate Key
 - **Date**: Month
6. Click **next**, and then click **Finish**.
7. Leave Visual Studio open for the next lab.

Results: After this exercise, unused attributes in the Customer and Product dimensions will have been removed; time intelligence will have been added to the Date dimension.

Exercise 2: Defining Relationships and Hierarchies

► Task 1: Create a Natural Hierarchy

1. In Solution Explorer, double-click the **Product.dim** dimension to open it in the Dimension Designer. Note that this dimension includes attributes from three related tables (**Product**, **Product Subcategory**, and **Product Category**).
2. On the **Dimension Structure** tab, in the Attributes pane, drag **English Product Category Name** into an empty area of the Hierarchies pane. This creates a new hierarchy named **Hierarchy**.
3. In the Attributes pane, drag **English Product Subcategory Name** to the **<new level>** area beneath **English Product Category Name** in the hierarchy.
4. In the Attributes pane, drag **English Product Name** to the **<new level>** area beneath **English Product Subcategory Name** in the hierarchy.

Note: if a warning is displayed, notifying you that attribute relationships do not exist and performance may be decreased, ignore it. You will see how to use attribute relationships to optimize a hierarchy later in this lab.

5. Right-click **Hierarchy** and click **Rename**. Then rename the hierarchy to **Categorized Products**.
6. In the **Categorized Products** hierarchy, right-click **English Product Category Name** and click **Rename**. Rename the hierarchy level to **Category**.
7. In the **Categorized Products** hierarchy, right-click **English Product Subcategory Name** and click **Rename** then rename the hierarchy level to **Subcategory**.
8. In the **Categorized Products** hierarchy, right-click **English Product Name** and click **Rename** then rename the hierarchy level to **Product**.
9. In the Attributes pane, select all the attributes by clicking the first one, and then holding Shift and clicking the last one. In the Properties pane, set the **AttributeHierarchyVisible** property to **False**. This hides the individual attributes, making the **Categorized Products** hierarchy the only way to browse the **Product** dimension.
10. On the **Dimension** menu, click **Process**. If you are prompted to build and deploy the project first, click **Yes**. If you are prompted to enter a password, type **Pa\$\$w0rd** and click **OK**.
11. In the **Process Dimension-Product** dialog box, click **Run** and wait for the dimension to be processed. Click **Close**, and click **Close** again to close the **Process Dimension-Product** dialog box.
12. On the **Browser** tab, ensure that the **Categorized Products** hierarchy is selected and expand the **All** level to display the categories.
13. Expand the categories and subcategories to view individual products.
14. On the **File** menu, click **Save All**.

► Task 2: Create a Non-Natural Hierarchy

1. In Solution Explorer, double-click the **Customer.dim** dimension to open it.
2. On the **Dimension Structure** tab, in the Attributes pane, drag **Gender** into an empty area of the Hierarchies pane. This creates a new hierarchy named **Hierarchy**.
3. Right-click **Hierarchy** and click **Rename**, then rename the hierarchy to **Gender-Marital Status**.

- In the Attributes pane, drag **Marital Status** to the **<new level>** area beneath Gender in the **Gender-Marital Status** hierarchy.

Note: if a warning is displayed, saying that attribute relationships do not exist and that performance may be decreased, ignore it. You will see how to use attribute relationships to optimize a hierarchy later in this lab.

- On the **Dimension** menu, click **Process**. If you are prompted to build and deploy the project first, click **Yes**, and if you are prompted to confirm the database will be overwritten, click **Yes**.
- In the **Process Dimension- Customer** dialog box, click **Run** and wait for the dimension to be processed. Click **Close**, and click **Close** again to close the **Process Dimension-Customer** dialog box.
- On the **Browser** tab, ensure that the **Gender-Marital Status** hierarchy is selected and expand the **All** level to display the **gender** level. Note that the gender can be **F**, **M**, or **Unknown**.
- Expand **F** and **M**, and note that the **marital status** level value can be **M** or **S**.
- On the **File** menu, click **Save All**.

► Task 3: Create a Hierarchy with Attribute Relationships

Configure Attribute Column Bindings

- In Solution Explorer, double-click the **Date.dim** dimension to open it in the Dimension Designer.
- In the Properties pane, note that the **Type** property for this dimension has been set to **Time**. This occurred when you added dimension intelligence in the previous exercise.
- In the Attributes pane, right-click **Calendar Semester**, and click **Properties**.
- In the Properties pane, scroll down to the **Source** section, click the **KeyColumns** field, and then click the ellipses (...) button.
- In the **Key Columns** dialog box, in the **Available Columns** table, click **CalendarYear**, and then click the right arrow (>) icon.
- Click the up arrow icon to move **CalendarYear** above **CalendarSemester**, and then click **OK**.
- In the Properties pane, click the **NameColumn** field, and then click the ellipses (...) button.
- In the **Name Column** dialog box, in the **Source column** field, click **CalendarSemester**, and then click **OK**.
- In the Properties pane, click the **ValueColumn** field, and then click the ellipses (...) button.
- In the **Value Column** dialog box, in the **Source column** field, click **CalendarSemester**, and then click **OK**.
- In the Attributes pane, click **Calendar Quarter**.
- In the Properties pane, scroll down to the **Source** section, click the **KeyColumns** field, and then click the ellipses (...) button.
- In the **Key Columns** dialog box, in the **Available Columns** table, click **CalendarYear**, and then click the right arrow (>) icon.
- Click the up arrow icon to move **CalendarYear** above **CalendarQuarter**, and then click **OK**.
- In the Properties pane, click the **NameColumn** field, and then click the ellipses (...) button.
- In the **Name Column** dialog box, in the **Source column** field, click **CalendarQuarter**, and then click **OK**.
- In the Properties pane, click the **ValueColumn** field, and then click the ellipses (...) button.

18. In the **Value Column** dialog box, in the **Source column** field, click **CalendarQuarter**, and then click **OK**.
19. In the Attributes pane, click **Month**, and in the Properties pane, scroll down to the **Source** section, click the **KeyColumns** field, and then click the ellipses (...) button.
20. In the **Key Columns** dialog box, in the **Key Columns** table, click **EnglishMonthName**, and then click the left arrow (<) icon.
21. In the **Available Columns** table, click **CalendarYear**, and then click the right arrow (>) icon.
22. In the **Available Columns** table, click **MonthNumberOfYear**, then click the right arrow (>) icon. Ensure that **CalendarYear** appears above **MonthNumberOfYear**, and then click **OK**.
23. In the Properties pane, click the **NameColumn** field, and then click the ellipses (...) button.
24. In the **Name Column** dialog box, in the **Source column** field, click **EnglishMonthName**, and then click **OK**.
25. In the Properties pane, click the **ValueColumn** field, and then click the ellipses (...) button.
26. In the **Value Column** dialog box, in the **Source column** field, click **EnglishMonthName**, and then click **OK**.
27. On the **File** menu, click **Save All**.

Create Attribute Relationships

1. In the **Date.dim** Dimension Designer, click the **Attributes Relationships** tab.
2. In the diagram pane, right-click an empty space, and then click **New Attribute Relationship**.
3. In the **Create Attribute Relationship** dialog box, in the **Source Attribute** section, in the **Name** field, click **Full Date Alternate Key**. Then, in the **Related Attribute** section, in the **Name** field, click **Month**.
4. In the **Relationship type** field click **Rigid (will not change over time)**, because a particular date will always be in the same month, and then click **OK**.
5. Repeat the previous three steps to create the following relationships:
 - **Month > Calendar Quarter** (Rigid)
 - **Calendar Quarter > Calendar Semester** (Rigid)
 - **Calendar Semester > Calendar Year** (Rigid)
6. On the **File** menu, click **Save All**.

Create a Hierarchy

1. In the **Date.dim** Dimension Designer, on the **Dimension Structure** tab, in the Attributes pane, drag **Calendar Year** into the Hierarchies pane. A new hierarchy named **Hierarchy** is created.
2. Right-click **Hierarchy** and click **Rename** then rename the hierarchy to **Calendar Date**.
3. In the Attributes pane, drag the following attributes one-by-one to the **<new level>** area in the **Calendar Date** hierarchy:
 - Calendar Semester
 - Calendar Quarter
 - Month
 - Full Date Alternate Key

4. In the **Calendar Date** hierarchy, right-click **Full Date Alternate Key** and click **Rename**. Then rename the hierarchy level to **Day**.
5. In the Attributes pane, select **Calendar Year**, press the Ctrl key, and click **Calendar Semester**, **Calendar Quarter**, **Month**, and **Full Date Alternate Key** to select all of these attributes.
6. In the Properties pane, change the **AttributeHierarchyVisible** property to **False**. This defines these attributes as member properties rather than hierarchies in their own right, so that users can only browse them through the **Calendar Date** hierarchy.
7. On the **Dimension** menu, click **Process**. If you are prompted to build and deploy the project first, click **Yes**, and if you are prompted to overwrite the database click **Yes**.
8. In the **Process Dimension-Date** dialog box, click **Run** and wait for the dimension to be processed. Click **Close**, and click **Close** again to close the **Process Dimension-Date** dialog box.

Results: After this exercise, you should have created a Categorized Products hierarchy and a Gender-Marital Status hierarchy.

Exercise 3: Sorting and Grouping Dimension Attributes

► Task 1: Sort Attribute Members

1. In the **Browser** tab, in the **Hierarchy** drop-down, ensure **Calendar Date** is selected.
2. Expand **All**, expand **2005**, and then expand **1**. Note that months are displayed in alphabetical order instead of chronological order.
3. On the **Dimension Structure** tab, in the Attributes pane, right-click the **Month** attribute and click **Properties**.
4. In the Properties pane, in the **OrderBy** box, select **Key**.
5. On the **Dimension** menu, click **Process**. If you are prompted to build and deploy the project first, click **Yes**. If you are prompted to overwrite the database, click **Yes**. In the **Process Dimension-Date** dialog, click **Run**.
6. When processing is complete, click **Close** and then, in the **Process Dimension-Date** dialog, click **Close**.
7. In the **Browser** tab, in the **Hierarchy** drop-down, ensure **Calendar Date** is selected.
8. In the **Dimension** menu, click **Refresh**. Note that the months are now correctly sorted in chronological order.

► Task 2: Group Attribute Members

1. In Solution Explorer, double-click the **Customer.dim** dimension to open it in the Dimension Designer.
2. On the **Dimension Structure** tab, in the Data Source View pane, right-click the **Customer** table, and then click **Explore Data**. Notice the range of values for the **YearlyIncome** column, and then close the **Explore Customer Table** window.
3. On the **Browser** tab, change the **Hierarchy** field to **Yearly Income**.
4. Expand **All**, and notice that the data is unstructured.
5. On the **Dimension Structure** tab, in the Attributes pane, click **Yearly Income**.

6. In the Properties pane, in the **DiscretizationMethod** box, click **Automatic**, in the **DiscretizationBucketCount** box type **5** and, in the **OrderBy** list, select **Key**.
7. On the **Dimension** menu, click **Process**. If you are prompted to build and deploy the project first, click **Yes**, and if you are prompted to overwrite the database, click **Yes**.
8. In the **Process Dimension-Customer** dialog, click **Run** and then, when processing is complete, click **Close**. Click **Close** in the **Process Dimension-Customer** dialog.
9. On the **Browser** tab, click **Reconnect** and verify that the **yearly income** is grouped into five ranges, with a sixth member for unknown values.

Results: You will have a multidimensional project with enhanced dimensions.

Module 4: Working with Measures and Measure Groups

Lab: Configuring Measures and Measure Groups

Exercise 1: Configuring Measures

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab04\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.

► Task 2: Create a Measure Group

1. Start SQL Server Data Tools and open the **Adventure Works OLAP.sln** solution in the D:\Labfiles\Lab04\Starter folder.
2. In Solution Explorer, right-click **Adventure Works DSV.dsv**, and then click **View Designer**.
3. On the **Data Source View** menu, click **Add/Remove Tables**.
4. In the **Add/Remove Tables** dialog box, in the **Available objects** list, select **FactResellerSales (dbo)**, and then click the > button to add the selected table to the **Included objects** list.
5. Click **Add Related Tables** to add the tables that are related to **FactResellerSales**, and then click **OK**. Note that the tables have been added to the data source view, and then, on the **File** menu, click **Save All**.
6. Close the **Adventure Works DSV.dsv** designer.
7. In Solution Explorer, right-click **Sales.cube**, and then click **View Designer**.
8. On the **Cube** menu, click **New Measure Group**, and then in the **New Measure Group** dialog box, select **FactResellerSales**, and then click **OK**.
9. In the Measures pane, right-click the **Fact Reseller Sales** measure group, click **Rename**, and then change the name of the measure group to **Reseller Sales**.
10. Expand the **Reseller Sales** measure group, and then review the names of the measures that it contains. Note that when the **Reseller Sales** measure group was set up, measures were created for all of the numerical fields in the **FactResellerSales** table.

► Task 3: Modify Measure Groups

1. On the **Cube** menu, point to **Show Measures In**, and then click **Grid** to view all of the measures in the cube as a grid. Note the aggregations that are used to summarize the measures when they are analyzed across dimensions.
2. Click the row for the **Revision Number** measure, and then hold down the Ctrl key and click the rows for the following measures to select them all:
 - Unit Price
 - Extended Amount

- Unit Price Discount Pct
 - Discount Amount
 - Product Standard Cost
 - Tax Amt
 - Freight
3. Click the **Delete** button, and then in the **Delete Objects** dialog box, click **OK** to remove these measures.
 4. Right-click the **Order Quantity** measure, click **Rename**, and then change the name of the measure to **Reseller Order Quantity**.
 5. Repeat the previous step to rename the following measures:
 - Total Product Cost (rename to **Reseller Cost**).
 - Sales Amount (rename to **Reseller Revenue**).
 - Fact Reseller Sales Count (rename to **Reseller Sales Count**).
 6. On the **Cube** menu, point to **Show Measures In**, click **Tree** to view measures in the cube as a tree, and then, on the **File** menu, click **Save All**.

Results: After this exercise, you should have created a new measure group for the **FactResellerSales** table, removed unrequired measures, and renamed measures.

Exercise 2: Defining a Regular Relationship

► Task 1: View Existing Dimensions for Measure Groups

1. On the **Build** menu, click **Deploy Solution**. If you are prompted for an account password, type **Pa\$\$w0rd**, and then click **OK**.
2. Wait for the **Deployment Completed Successfully** message in the Deployment Progress – Adventure Works OLAP window, and then close the Deployment Progress – Adventure Works OLAP window.
3. In Solution Explorer, right-click **Sales.cube**, and then click **Browse**.
4. In the Metadata pane, in the **Measure Group** drop-down list, select **Internet Sales**. Notice that there is a **Customer** dimension.
5. In the **Measure Group** drop-down list, select **Reseller Sales** and notice that there is no **Customer** dimension, because reseller sales are sold to resellers, which are defined in a different dimension table.

► Task 2: Create a Dimension

1. In Solution Explorer, right-click **Dimensions**, and then click **New Dimension**.
2. In the Dimension Wizard, on the welcome page, click **Next**.
3. On the **Select Creation Method** page, ensure that **Use an existing table** is selected, and then click **Next**.

4. On the **Specify Source Information** page, in the **Main table** list, select **DimReseller**, and then click **Next**.
5. On the **Select Related Tables** page, clear **Geography and Sales Territory**, and then click **Next**.
6. On the **Select Dimension Attributes** page, select only the following attributes, and then click **Next**:
 - **Reseller Key**
 - **Business Type**
 - **Reseller Name**
7. On the **Completing the Wizard** page, in the **Name** box, type **Reseller**, and then click **Finish**.
8. In the **Sales.cube** designer, click the **Dimension Usage** tab.
9. On the **Cube** menu, click **Add Cube Dimension**.
10. In the **Add Cube Dimension** dialog box, select **Reseller**, and then click **OK**.
11. On the **Dimension Usage** tab, click the **Reseller Key** cell at the intersection of the **Reseller Sales** measure group and the **Reseller** dimension, and then click the ellipsis (...) button.
12. In the **Define Relationship** dialog box, note that a **Regular** relationship type has been detected, and then click **Cancel**.
13. On the **File** menu, click **Save All**.
14. On the **Build** menu, click **Deploy Solution**. If you are prompted for an account password, type **Pa\$\$w0rd**, and then click **OK**.
15. Wait for the **Deployment Completed Successfully** message in the Deployment Progress – Adventure Works OLAP window, and then close the Deployment Progress – Adventure Works OLAP window.
16. In Solution Explorer, right-click **Sales.cube**, and then click **Browse**.
17. On the **Cube** menu, click **Reconnect**.
18. In the Metadata pane, in the **Measure Group** drop-down list, select **Reseller Sales**. Notice that there is now a **Reseller** dimension.
19. Expand **Measures**, expand **Reseller Sales**, and then drag **Reseller Revenue** to the query results area.
20. Expand **Reseller**, and then drag **Business Type** to the left of the **Reseller Revenue** column to see reseller sales revenue broken down by business type.

Results: After this exercise, you should have added a **Reseller** dimension that uses a regular relationship with the **Reseller Sales** measure group to enable you to analyze reseller sales data.

Exercise 3: Configuring Measure Group Storage

► Task 1: Configure Proactive Caching

1. In the Cube Designer for the **Sales** cube, on the **Cube Structure** tab, in the Measures pane, right-click **Internet Sales**, and then click **Properties**.
2. In the Properties window, click **ProactiveCaching**, and then click the ellipsis (...) button.

3. Under **Standard setting**, review each of the storage setting options. When you have finished, drag the slider to **Automatic MOLAP**, and then click **OK**.
4. Repeat the previous steps to configure the **Reseller Sales** measure group for **Automatic MOLAP** proactive caching.

► Task 2: Design Aggregations

1. In the Cube Designer for the **Sales** cube, click the **Aggregations** tab.
2. Right-click **Internet Sales (0 Aggregation Designs)**, and then click **Design Aggregations**.
3. On the **Welcome to the Aggregation Design Wizard** page, click **Next**.
4. On the **Review Aggregation Usage** page, click **Set All to Default**, and then click **Next**.
5. On the **Specify Object Counts** page, click **Count**. When the count process has completed, click **Next**.
6. On the **Set Aggregations Options** page, select **Performance gain reaches**, ensure that the value is set to **30%**, and then click **Start**.
7. When the wizard has designed aggregations for a performance gain of at least 30 percent, click **Next**.
8. On the **Completing the Wizard** page, change the name of the aggregation to **InternetSalesAgg**, select **Save the aggregations but do not process them**, and then click **Finish**.
9. Repeat the previous steps to design **Reseller Sales** aggregations for a performance gain of 30 percent. Name the aggregation design **ResellerSalesAgg**.
10. On the **File** menu, click **Save All**.
11. On the **Build** menu, click **Deploy Solution**. If you are prompted for an account password, type **Pa\$\$w0rd**, and then click **OK**.
12. Wait for the **Deployment Completed Successfully** message in the Deployment Progress – Adventure Works OLAP window, and then close the Deployment Progress – Adventure Works OLAP window.
13. Close SQL Server Data Tools.
14. Start SQL Server Management Studio. When you are prompted, in the **Connect to Server** dialog box, specify the following settings, and then click **Connect**:
 - **Server type**: Analysis Services
 - **Server name**: localhost
15. In Object Explorer, expand **Databases**, expand **Adventure Works OLAP**, expand **Cubes**, expand **Sales**, expand **Measure Groups**, expand **Internet Sales**, and then expand **Aggregation Designs**.
16. Verify that the **InternetSalesAgg** aggregation design has been deployed with the cube.
17. Expand **Reseller Sales** and its **Aggregation Designs** folder to verify that the **ResellerSalesAgg** aggregation design has been deployed with the cube.
18. Close SQL Server Management Studio.

Results: After this exercise, you should have configured proactive caching and defined the storage mode aggregations for the **Internet Sales** and **Reseller Sales** measure groups.

Module 5: Introduction to MDX

Lab: Using MDX

Exercise 1: Adding Calculated Members

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab05\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.

► Task 2: Add a Calculated Member by Using Form View

1. On the Start screen, type **SQL Server Data Tools 2015**, and then press Enter.
2. On the **File** menu, point to **Open**, click **Project/solution**, browse to the D:\Labfiles\Lab05\Starter folder, and then double-click **Adventure Works OLAP.sln** to open it.
3. In Solution Explorer, right-click **Sales.cube**, and then click **View Designer**.
4. On the **Cube** menu, point to **View**, and then click **Calculations**.
5. On the **Cube** menu, click **New Calculated Member**.
6. In the **Name** text box, type **[Total Revenue]**.
7. In the **Parent hierarchy** box, ensure that **Measures** is selected.
8. In the **Expression** text box, type the following MDX expression:

```
[Measures].[Internet Revenue] + [Measures].[Reseller Revenue]
```
9. In the **Format string** drop-down list, select **"Currency"**.
10. In the **Non-empty behavior** list, select **Internet Revenue** and **Reseller Revenue**, and then click **OK**.
11. On the **File** menu, click **Save selected items**.
12. Above the form, click the **Script View** button, and then check the MDX text that has been added to the cube.

► Task 3: Add Calculated Members by Using Script View

1. In the Script Editor pane, place the cursor after the last semicolon, and then press Enter.
2. Type the following code:

```
CREATE MEMBER CURRENTCUBE.[Measures].[Internet Profit]
  AS ([Measures].[Internet Revenue] - [Measures].[Internet Cost]) /
  [Measures].[Internet Cost],
  FORMAT_STRING = "Percent",
  NON_EMPTY_BEHAVIOR = { [Internet Revenue], [Internet Cost] },
  VISIBLE = 1,
  ASSOCIATED_MEASURE_GROUP = 'Internet Sales' ;
```

3. Press Enter, and then type the following code:

```
CREATE MEMBER CURRENTCUBE.[Measures].[Reseller Profit]
AS ([Measures].[Reseller Revenue] - [Measures].[Reseller Cost]) /
[Measures].[Reseller Cost],
FORMAT_STRING = "Percent",
NON_EMPTY_BEHAVIOR = { [Reseller Revenue], [Reseller Cost] },
VISIBLE = 1,
ASSOCIATED_MEASURE_GROUP = 'Reseller Sales' ;
```

4. On the **File** menu, click **Save selected items**.
5. On the **Build** menu, click **Deploy Solution**. If you are prompted to overwrite the database, click **Yes**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**, and then click **OK**.
6. When the deployment has completed successfully, close Visual Studio.

Results: After this exercise, you should have added three calculated members to the **Sales** cube and deployed the cube to the MIA-SQL instance of SQL Server Analysis Services.

Exercise 2: Querying a Cube by Using MDX

► Task 1: Test Total Revenue in SQL Server Management Studio

1. Start SQL Server Management Studio.
2. In the **Connect to Server** dialog box, in the **Server type** drop-down list, select **Analysis Services**.
3. In the **Server name** drop-down list, select **MIA-SQL**, and then click **Connect**.
4. In Object Explorer, expand **Databases**.
5. Right-click **Adventure Works OLAP**, point to **New Query**, and then click **MDX**.
6. In the query pane, type the following code:

```
SELECT
{
  [Measures].[Internet Revenue],
  [Measures].[Reseller Revenue],
  [Measures].[Total Revenue]
} ON COLUMNS,
[Product].[Categorized Products].Members ON ROWS
FROM [Sales]
```

7. Click **Execute**, and then examine the results.

► Task 2: Test Profit Calculations in SQL Server Management Studio

1. In SQL Server Management Studio, in Object Explorer, right-click **Adventure Works OLAP**, point to **New Query**, and then click **MDX**.
2. In the query pane, type the following code:

```
SELECT
{
[Measures].[Internet Cost],
[Measures].[Internet Revenue],
[Measures].[Internet Profit]
} ON COLUMNS,
[Customer].[City].Members ON ROWS
FROM [Sales]
```

3. Click **Execute**, and then examine the results.
4. In the query pane, after the existing code, add the following line of code:

```
WHERE ([Customer].[State Province Name].&[New York])
```

5. Click **Execute**, and then examine the results.
6. In SQL Server Management Studio, in Object Explorer, right-click **Adventure Works OLAP**, point to **New Query**, and then click **MDX**.
7. In the query pane, type the following code:

```
SELECT
{
[Measures].[Reseller Cost],
[Measures].[Reseller Revenue],
[Measures].[Reseller Profit]
} ON COLUMNS,
[Sales Territory].[Sales Territory].Members ON ROWS
FROM [Sales]
```

8. Click **Execute**, and then examine the results.
9. Close SQL Server Management Studio, without saving any changes.

► Task 3: Use Excel to Execute MDX Queries

1. On the taskbar, click **Excel 2016**.
2. On the home page, click **Blank workbook**.
3. On the ribbon, click the **Data** tab.
4. In the **Get & Transform** group, click **New Query**, point to **From Database**, and then click **From SQL Server Analysis Services Database**.
5. In the **Microsoft SQL Server Analysis Services Database** dialog box, in the **Server** box, type **MIA-SQL**, and then click **OK**.
6. In the **Access Microsoft SQL Server Analysis Services** dialog box, click **Connect**.
7. In the Navigator pane, expand **Adventure Works OLAP**, expand the **Sales** folder, expand the **Sales** cube, and then expand the **Internet Sales** measure group.
8. Select **Internet Cost**, **Internet Revenue**, and **Internet Profit**.
9. Expand **Product**, and then expand **Categorized Products**.

10. Select **Category** and **Subcategory**, and then click **Load**.

► **Task 4: Use an MDX Query in a PivotTable**

1. In Excel, click **Sheet 1**.
2. On the ribbon, click the **Data** tab.
3. Click **Get External Data**, click **From Other Sources**, and then click **From Analysis Services**.
4. In the **Data Connection Wizard** dialog box, on the **Connect to Database Server** page, in the **Server name** text box, type **MIA-SQL**, and then click **Next**.
5. On the **Select Database and Table** page, ensure that the **Adventure Works OLAP** database and the **Sales** cube are selected, and then click **Next**.
6. On the **Save Data Connection File and Finish** page, click **Finish**.
7. In the **Import Data** dialog box, click **New worksheet**, and then click **OK**.
8. In the PivotTable Fields pane, select **Internet Revenue**, **Reseller Revenue**, and **Total Revenue**.
9. Under **Sales Territory**, select **Sales Territory**.
10. In the worksheet, expand **Europe**, and note the **Total Revenue** value for the **United Kingdom**.
11. Close Excel, without saving any changes.

Results: After this exercise, you should have created several test MDX queries in SQL Server Management Studio and an Excel spreadsheet that connects to the **Sales** OLAP cube.

Module 6: Customizing Cube Functionality

Lab: Customizing a Cube

Exercise 1: Implementing an Action

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab06\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.

► Task 2: Create a Drill-Through Action

1. Start Visual Studio, on the File menu, point to **Open**, and then click **Project/Solution**. Browse to D:\Labfiles\Lab06\Starter folder and open the **Adventure Works OLAP.sln** solution.
2. On the **Build** menu, click **Deploy Solution**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**, and then click **OK**.
3. When deployment is complete, close the Deployment Progress... window.
4. In Solution Explorer, double-click **Sales.cube** to open it in the Cube Designer.
5. In the Cube Designer, click the **Actions** tab.
6. On the **Cube** menu, click **New Drillthrough Action**.
7. In the **Name** box, change the name of this action to **Internet Sales Details**.
8. In the **Measure group members** list, select **Internet Sales**.
9. In the **Drillthrough Columns** box, in the **Select Dimensions** list, select **Customer**, in the **Return Columns** list, select **City** and **Full Name**, and then click **OK**.
10. In the **Drillthrough Columns** box, in the **Select Dimensions** list, click **Order Date**, in the **Return Columns** list, select **Full Date Alternate Key**, and then click **OK**.
11. In the **Drillthrough Columns** box, in the **Select Dimensions** list, click **Product**, in the **Return Columns** list, select **English Product Name**, and then click **OK**.
12. Expand **Additional Properties**, and then in the **Caption** box, type **Drillthrough to Order Details**.
13. On the **File** menu, click **Save All**.

► Task 3: Browse a Drill-Through Action

1. In Visual Studio, on the **Build** menu, click **Deploy Solution**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**, and then click **OK**.
2. Close the Deployment Progress... window, and then in the Cube Designer, click the **Browser** tab.
3. On the **Cube** menu, click **Analyze in Excel**. If a security notice is displayed, click **Enable**.
4. In Excel, in the PivotTable Fields pane, under **Internet Sales**, select **Internet Revenue**.
5. In the PivotTable Fields pane, under **Order Date**, select **Order Date.Calendar Date**.
6. In the PivotTable Fields pane, under **Product**, select **Categorized Products**.

7. In the PivotTable, right-click the sales amount for **Bikes** in **2007**, point to **Additional Actions**, and then click **Drillthrough to Order Details**.
8. View the new worksheet that is generated by the drill-through action, noting that it shows the city, customer name, and product name for each sale of a bike in 2007.
9. Close Excel without saving the workbook.
10. Keep Visual Studio open for the next exercise.

Results: After this exercise, you should have a new drill-through action in the solution.

Exercise 2: Implementing Perspectives

► Task 1: Create a Perspective

1. In Visual Studio, in the **Sales.cube** Cube Designer, click the **Perspectives** tab.
2. On the **Cube** menu, click **New Perspective**.
3. Change the name of the new perspective. Clear the first row of the first **Perspective Name** column, and then type **Reseller Sales**.
4. In the same column, clear the check box for the following objects:
 - a. The **Reseller Sales** measure group (this clears all of the measures in this measure group).
 - b. The **Reseller** dimension.
 - c. The **Reseller Profit** calculated member.
5. Create a new perspective. On the **Cube** menu, click **New Perspective**.
6. Change the name of the new perspective. Clear the first row of the second **Perspective Name** column, and then type **Internet Sales**.
7. In the same column, clear the check box for the following objects:
 - a. The **Internet Sales** measure group (this clears all of the measures in this measure group).
 - b. The **Customer** dimension.
 - c. The **Internet Sales Details** action.
 - d. The **Internet Profit** calculated member.
8. On the **File** menu, click **Save All**.

► Task 2: Browse a Perspective

1. In Visual Studio, on the **Build** menu, click **Deploy Solution**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**, and then click **OK**.
2. When deployment has successfully completed, close the Deployment Progress... window.
3. In the Cube Designer, click the **Browser** tab, and then click the **Reconnect** icon.
4. In the cube selection area above the Metadata pane, click the ellipsis (...), select **Internet Sales**, and then click **OK**.
5. In the Metadata pane, expand **Measures**, and then note that only the **Internet Sales** measure group is shown in this perspective.

6. Expand **Internet Sales**, and then drag the **Internet Revenue** measure to the query results area.
7. In the Metadata pane, expand the **Customer** dimension, and then drag the **Yearly Income** hierarchy to the left of the **Internet Revenue** figure. This displays Internet revenue for customers, broken down by income band.
8. On the **Cube** menu, click **Analyze in Excel**.
9. In the **Analyze in Excel** dialog box, select the **Reseller Sales** perspective, and then click **OK**. If a security notice is displayed, click **Enable**.
10. In Excel, in the PivotTable Fields pane, note that only the **Reseller Sales** measures are shown.
11. In the PivotTable Fields pane, under **Reseller Sales**, select **Reseller Revenue**.
12. In the PivotTable Fields pane, under **Reseller**, select **Business Type**. The PivotTable shows revenue for reseller sales, broken down by business type.
13. Close Excel without saving the workbook.

Results: After this exercise, you should have defined a new perspective in the solution.

Exercise 3: Implementing a Translation

► Task 1: Create Dimension Translations

1. In Visual Studio, in Solution Explorer, in the Dimensions folder, double-click **Date.dim** to open it in the dimension designer, and then click the **Translations** tab.
 2. On the **Dimension** menu, click **New Translation**.
 3. In the **Select Language** dialog box, select **French (France)**, and then click **OK**.
 4. In the row for the **Calendar Date** hierarchy, type **Date du Calendrier** in the French (France) translation column.
 5. In the row for the **Calendar Year** level, type **Année** in the French (France) translation column.
-  **Note:** To type é, press Alt+130 by using the number pad. Ensure that Num Lock is on. If this does not work, type the captions without accents.
6. In the row for the **Calendar Semester** level, type **Semestre** in the French (France) translation column.
 7. In the row for the **Calendar Quarter** level, type **Trimestre** in the French (France) translation column.
 8. In the row for the **Month** level, type **Mois** in the French (France) translation column.
 9. In the row for the **Day** level, type **Journée** in the French (France) translation column.
 10. On the **Translation** tab toolbar, click **Show All Attributes**. This reveals the attributes that are hidden in the dimension.
 11. Click the **French (France)** translation row for the **Month** attribute, and then click the **[.]** button to open the **Attribute Data Translation** dialog box.
 12. In the **Translation columns** list, click **FrenchMonthName**, and then click **OK**.
 13. On the **File** menu, click **Save All**.
 14. Close the Date.dim dimension designer.

► Task 2: Create Cube Translations

1. In Solution Explorer, double-click **Sales.cube** to view it in the Cube Designer.
2. In the Cube Designer, click the **Translations** tab.
3. On the **Cube** menu, click **New Translation**, and then, in the **Select Language** dialog box, click **French (France)**, and then click **OK**.
4. In the row for the **Internet Sales** measure group, in the French (France) translation column, type **Ventes d'Internet**.
5. In the row for the **Internet Revenue** measure, in the French (France) translation column, type **Revenu d'Internet**.
6. In the row for the **Order Date** dimension, in the French (France) translation column, type **Date de Vente**.
7. On the **File** menu, click **Save All**.

► Task 3: Browse Translations

1. In Visual Studio, on the **Build** menu, click **Deploy Solution**. If you are prompted for impersonation credentials, enter the password **Pa\$\$w0rd** for **ADVENTUREWORKS\ServiceAcct**, and then click **OK**.
2. When deployment is complete, close the Deployment Progress... window.
3. In the Cube Designer, click the **Browser** tab, and then click the **Reconnect** icon.
4. On the toolbar, in the **Language** list, select **French (France)**.
5. In the Metadata pane, expand **Measures**, expand **Ventes d'Internet**, and then drag **Revenu d'Internet** to the query results area.
6. In the Metadata pane, expand the **Date de Vente** dimension, and then drag the **Date de Vente.Date du Calendrier** hierarchy to the left of **Revenu d'Internet**. Note that the captions are displayed in French, and that the month names in the **Mois** column are also in French.
7. Close Visual Studio.

Results: After this exercise, you should have French data and metadata in the cube.

Module 7: Implementing a Tabular Data Model by Using SQL Server Analysis Services

Lab: Implementing a Tabular Data Model in SQL Server Analysis Services

Exercise 1: Creating a Tabular Data Model Project in SQL Server Analysis Services

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab07\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.

► Task 2: Create a Tabular Project in SQL Server Analysis Services

1. Start SQL Server Data Tools 2015. On the **File** menu, point to **New**, and then click **Project**.
2. In the **New Project** dialog box, type the following values, and then click **OK**:
 - **Project Template:** Analysis Services Tabular Project
 - **Name:** AWSalesTab
 - **Location:** D:\Labfiles\Lab07\Starter
3. If the **Tabular model designer** dialog box is displayed, in the **Workspace server** list, select **localhost\SQL2**, in the **Compatibility level** box, select **SQL Server 2016 RTM (1200)**, and then click **OK**.

► Task 3: Import Tables into the Data Model

1. In Solution Explorer, double-click **Model.bim** to open the model.
2. On the **Model** menu, click **Import from Data Source**.
3. In the **Table Import Wizard** dialog box, on the **Connect to a Data Source** page, select **Microsoft SQL Server**, and then click **Next**.
4. On the **Connect to a Microsoft SQL Server Database** page, type the following values, and then click **Next**:
 - **Friendly connection name:** AdventureWorksDW
 - **Server name:** MIA-SQL
 - **Log on to the server:** Use Windows Authentication
 - **Database name:** AdventureWorksDW
5. On the **Impersonation Information** page, in the **User Name** box, type **ADVENTUREWORKS\ServiceAcct**, in the **Password** box, type **Pa\$\$w0rd**, and then click **Next**.
6. On the **Choose How to Import the Data** page, ensure that **Select from a list of tables and views to choose the data to import** is selected, and then click **Next**.

7. On the **Select Tables and Views** page, select the following source tables, changing the **Friendly Name** value as indicated in parentheses:
 - **DimDate** (Date)
 - **DimEmployee** (Employee)
 - **DimGeography** (Geography)
 - **DimProduct** (Product)
 - **DimProductCategory** (Product Category)
 - **DimProductSubcategory** (Product Subcategory)
 - **DimReseller** (Reseller)
 - **FactResellerSales** (Reseller Sales)
8. Select the row for the **DimDate** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - SpanishDayNameOfWeek
 - FrenchDayNameOfWeek
 - DayNumberOfYear
 - WeekNumberOfYear
 - SpanishMonthName
 - FrenchMonthName
 - CalendarSemester
 - FiscalSemester
9. Select the row for the **DimEmployee** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - SalesTerritoryKey
 - NameStyle
 - Title
 - HireDate
 - BirthDate
 - LoginID
 - EmailAddress
 - Phone
 - MaritalStatus
 - EmergencyContactName
 - EmergencyContactPhone
 - SalariedFlag
 - Gender
 - PayFrequency
 - Baserate

-
- VacationHours
 - SickLeaveHours
 - CurrentFlag
 - SalesPersonFlag
 - StartDate
 - EndDate
 - Status
10. Select the row for the **DimGeography** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
- SpanishCountryRegionName
 - FrenchCountryRegionName
 - IpAddressLocator
11. Select the row for the **DimProduct** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
- WeightUnitMeasureCode
 - SizeUnitMeasureCode
 - SpanishProductName
 - FrenchProductName
 - FinishedGoodsFlag
 - SafetyStockLevel
 - ReorderPoint
 - DaysToManufacture
 - ProductLine
 - DealerPrice
 - Class
 - Style
 - ModelName
 - FrenchDescription
 - ChineseDescription
 - ArabicDescription
 - HebrewDescription
 - ThaiDescription
 - GermanDescription
 - JapaneseDescription
 - TurkishDescription
 - StartDate

- EndDate
 - Status
12. Select the row for the **DimProductCategory** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - SpanishProductCategoryName
 - FrenchProductCategoryName
 13. Select the row for the **DimProductSubcategory** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - SpanishProductSubcategoryName
 - FrenchProductSubcategoryName
 14. Select the row for the **DimReseller** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - OrderFrequency
 - OrderMonth
 - FirstOrderYear
 - LastOrderYear
 - ProductLine
 - AddressLine1
 - AddressLine2
 - AnnualSales
 - BankName
 - MinPaymentType
 - MinPaymentAmount
 - AnnualRevenue
 - YearOpened
 15. Select the row for the **FactResellerSales** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - DueDateKey
 - PromotionKey
 - CurrencyKey
 - SalesTerritoryKey
 - RevisionNumber
 - CarrierTrackingNumber
 - CustomerPONumber
 - DueDate
 16. When you have selected and filtered the tables, in the **Table Import Wizard** dialog box, click **Finish**, and then wait for the data to be imported. When the data has been imported successfully, click **Close**.

► Task 4: Create Measures

1. In the Model.bim pane, on the **Reseller Sales** tab, click the first empty cell in the grid under the **OrderQuantity** column.
2. On the **Column** menu, point to **AutoSum**, click **Sum**, and then, in the formula bar, modify the expression that has been generated to name the measure **Quantity**, as shown in the following example:

```
Quantity:=Sum([OrderQuantity])
```

3. Widen the **OrderQuantity** column to see the calculated **Quantity** measure, and then click the cell in the measure grid that contains the calculated value.
4. Press F4 to view the Properties pane, and then set the **Format** property to **Whole Number**, and the **Show Thousand Separator** property to **True**.
5. Click the first empty cell under the **TotalProductCost** column.
6. On the **Column** menu, point to **AutoSum**, click **Sum**, and then modify the expression that is generated to name the measure **Cost**, as shown in the following example:

```
Cost:=Sum([TotalProductCost])
```

7. Widen the **TotalProductCost** column to see the calculated **Cost** measure.
8. Select the cell that contains the calculated **Cost** measure, and then, in the Properties pane, ensure that the **Format** property is set to **Currency**.
9. Click the first empty cell under the **SalesAmount** column.
10. On the **Column** menu, point to **AutoSum**, click **Sum**, and then modify the expression that is generated to name the measure **Revenue**, as shown in the following example:

```
Revenue:=Sum([SalesAmount])
```

11. Widen the **SalesAmount** column to see the calculated **Revenue** measure.
12. Select the cell that contains the calculated **Revenue** measure, and then, in the Properties pane, ensure that the **Format** property is set to **Currency**.

► Task 5: Test the Model

1. In SQL Server Data Tools, with the Model.bim pane open, on the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and then click **OK**.
3. When Excel opens, note that the measures that you defined are displayed under a **Reseller Sales** measure group in the PivotTable Fields pane. The **Reseller Sales** table is also shown in this pane, and contains the columns in the table on which the measures are based, which some users might find confusing. You will fix this problem in the next exercise.
4. Select the **Revenue** measure, and then note that it is displayed in the PivotTable. In the **Geography** table, select **EnglishCountryRegionName**. The revenue is shown for each country or region, but the column name that is used to display this is not user-friendly.

5. Clear the **EnglishCountryRegionName** field. In the **Date** table, select **CalendarYear** and **EnglishMonthName**. The revenue is shown for each year, and each month. However, the months are listed in alphabetical rather than chronological order. In addition, the **Reseller Sales** table contains two date keys (**OrderDateKey** and **ShipDateKey**), but it is unclear which of these dates is used to determine the year and month. You will fix these problems in the next exercise.
6. Close Excel without saving the workbook.

Results: After this exercise, you should have created a tabular data model project.

Exercise 2: Configuring Columns and Relationships

► Task 1: Configure Relationships

1. In SQL Server Data Tools, with the Model.bim pane visible, on the **Model** menu, point to **Model View**, and then click **Diagram View**. The tables are shown as a schema diagram, with lines between them to denote relationships. Note that there are two relationships between the **Reseller Sales** table and the **Date** table.
2. Double-click the solid line between the **Reseller Sales** and **Date** tables, note the columns that are used to define the relationship, and that this is an active relationship, and then click **Cancel**.
3. Double-click the dotted line between the **Reseller Sales** and **Date** tables, note the columns that are used to define the relationship, and that this is an inactive relationship, and then click **Cancel**.
4. If the inactive relationship that you noted above is based on the **OrderDateKey** column, right-click the dotted line, and then click **Mark as Active** so that the active relationship is based on the **OrderDateKey** column.
5. Right-click the dotted relationship line (which should now represent the relationship that is based on the **ShipDateKey** column), and then click **Delete**. When you are prompted to delete the relationship from the model, click **Delete from Model**.
6. Right-click the **Date** table title bar, click **Rename**, and then rename the table to **Order Date**.
7. On the **Model** menu, click **Existing Connections**.
8. In the **Existing Connections** dialog box, ensure that the **AdventureWorksDW** connection is selected, and then click **Open**. If you are prompted for impersonation credentials, type the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **OK**.
9. On the **Choose How to Import the Data** page, ensure that **Select from a list of tables and views to choose the data to import** is selected, and then click **Next**.
10. On the **Select Tables and Views** page, select the **DimDate** table, and then change the **Friendly Name** value to **Ship Date**.
11. Select the row for the **DimDate** table, click **Preview & Filter**, clear the following columns, and then click **OK**:
 - SpanishDayNameOfWeek
 - FrenchDayNameOfWeek
 - DayNumberOfYear
 - WeekNumberOfYear
 - SpanishMonthName

- FrenchMonthName
 - CalendarSemester
 - FiscalSemester
12. Click **Finish**, and then wait for the table to be imported. When it has been imported successfully, click **Close**.
 13. Arrange the diagram so that you can see the **Ship Date** and **Reseller Sales** tables, and then drag the **ShipDateKey** column from the **Reseller Sales** table to the **DateKey** column in the **Ship Date** table to create the relationship.

► **Task 2: Rename and Hide Columns**

1. In SQL Server Data Tools, with the Model.bim pane visible, in the diagram, click the title bar of the **Geography** table, and then click its **Maximize** icon.
2. In the maximized **Geography** table, click the **GeographyKey** column, hold down CTRL, and then click the **SalesTerritoryKey** column to select both columns. Right-click either of the selected columns, and then click **Hide from Client Tools**.
3. In the maximized **Geography** table, right-click the **StateProvinceCode** column, click **Rename**, and then rename the column to **State or Province Code**.
4. Repeat the previous step to rename the following columns in the **Geography** table:

Column	New name
StateProvinceName	State or Province
CountryRegionCode	Country or Region Code
EnglishCountryRegionName	Country or Region
PostalCode	Postal Code

5. Click the **Restore** icon for the **Geography** table.
6. Click the title bar of the **Reseller** table, and then click its **Maximize** icon.
7. In the maximized **Reseller** table, click the **ResellerKey** column, hold down CTRL, and then click the **GeographyKey** column and the **ResellerAlternateKey** column to select the columns. Right-click any of the selected columns, and then click **Hide from Client Tools**.
8. In the maximized **Reseller** table, right-click the **BusinessType** column, click **Rename**, and then rename the column to **Business Type**.
9. Repeat the previous step to rename the following columns in the **Reseller** table:

Column	New name
ResellerName	Reseller Name
NumberEmployees	Employees

10. Click the **Restore** icon for the **Reseller** table, click the title bar of the **Employee** table, and then click its **Maximize** icon.

11. In the maximized **Employee** table, click the **EmployeeKey** column, hold down CTRL, and then click the **ParentEmployeeKey** column, the **EmployeeNationalIDAlternateKey** column, and the **ParentEmployeeNationalIDAlternateKey** column to select the columns. Right-click any of the selected columns, and then click **Hide from Client Tools**.
12. In the maximized **Employee** table, right-click the **FirstName** column, click **Rename**, and then rename the column to **First Name**.
13. Repeat the previous step to rename the following columns in the **Employee** table:

Column	New name
LastName	Last Name
MiddleName	Middle Name
DepartmentName	Department

14. Click the **Restore** icon for the **Employee** table, click the title bar of the **Order Date** table, and then click its **Maximize** icon.
15. In the maximized **Order Date** table, click the **DateKey** column, hold down CTRL, and then click the **DayNumberOfWeek** column and the **MonthNumberOfYear** column to select the columns. Right-click any of the selected columns, and then click **Hide from Client Tools**.
16. In the maximized **Order Date** table, right-click the **FullDateAlternateKey** column, click **Rename**, and then rename the column to **Date**.
17. Repeat the previous step to rename the following columns in the **Order Date** table:

Column	New name
EnglishDayNameOfWeek	Weekday
DayNumberOfMonth	Day of Month
EnglishMonthName	Month
CalendarQuarter	Calendar Quarter
CalendarYear	Calendar Year
FiscalQuarter	Fiscal Quarter
FiscalYear	Fiscal Year

18. Click the **Restore** icon for the **Order Date** table, click the title bar of the **Ship Date** table, and then click its **Maximize** icon.
19. In the maximized **Ship Date** table, click the **DateKey** column, hold down CTRL, and then click the **DayNumberOfWeek** column and the **MonthNumberOfYear** column to select the columns. Right-click any of the selected columns, and then click **Hide from Client Tools**.

20. In the maximized **Ship Date** table, right-click the **FullDateAlternateKey** column, click **Rename**, and then rename the column to **Date**.
21. Repeat the previous step to rename the following columns in the **Ship Date** table:

Column	New name
EnglishDayNameOfWeek	Weekday
DayNumberOfMonth	Day of Month
EnglishMonthName	Month
CalendarQuarter	Calendar Quarter
CalendarYear	Calendar Year
FiscalQuarter	Fiscal Quarter
FiscalYear	Fiscal Year

22. Click the **Restore** icon for the **Ship Date** table, click the title bar of the **Product** table, and then click its **Maximize** icon.
23. In the maximized **Product** table, click the **ProductKey** column, hold down CTRL, and then click the **ProductAlternateKey** column and the **ProductSubcategoryKey** column to select the columns. Right-click any of the selected columns, and then click **Hide from Client Tools**.
24. In the maximized **Product** table, right-click the **EnglishProductName** column, click **Rename**, and then rename the column to **Product Name**.
25. Repeat the previous step to rename the following columns in the **Product** table:

Column	New name
StandardCost	Standard Cost
ListPrice	List Price
SizeRange	Size Range
EnglishDescription	Description

26. Click the **Restore** icon for the **Product** table, click the title bar of the **Product Subcategory** table, and then click its **Maximize** icon.
27. In the maximized **Product Subcategory** table, click the **ProductSubcategoryKey** column, hold down CTRL, and then click the **ProductSubcategoryAlternateKey** column and the **ProductCategoryKey** column to select the columns. Right-click any of the selected columns, and then click **Hide from Client Tools**.
28. In the maximized **Product Subcategory** table, right-click the **EnglishProductSubcategoryName** column, click **Rename**, and then rename the column to **Subcategory**.

29. Click the **Restore** icon for the **Product Subcategory** table, click the title bar of the **Product Category** table, and then click its **Maximize** icon.
30. In the maximized **Product Category** table, click the **ProductCategoryKey** column, hold down CTRL, and then click the **ProductCategoryAlternateKey** column to select both columns. Right-click either of the selected columns, and then click **Hide from Client Tools**.
31. In the maximized **Product Category** table, right-click the **EnglishProductCategoryName** column, click **Rename**, and then rename the column to **Category**.
32. Click the **Restore** icon for the **Product Category** table, click the title bar of the **Reseller Sales** table, and then click its **Maximize** icon.
33. In the maximized **Reseller Sales** table, click the **ProductKey** column, hold down SHIFT, and then click the **ShipDate** column to select all columns other than the **Quantity**, **Cost**, and **Revenue** measures that you created in the previous exercise. Right-click any of the selected columns, and then click **Hide from Client Tools**.
34. Click the **Restore** icon for the **Reseller Sales** table.

► Task 3: Configure Column Sort Order

1. In SQL Server Data Tools, with the Model.bim pane open in the diagram view, on the **Model** menu, point to **Model View**, and then click **Data View**.
2. On the **Ship Date** tab, click the column heading for the **Weekday** column.
3. On the **Column** menu, point to **Sort**, and then click **Sort by Column**.
4. In the **Sort By Column** dialog box, in the **Sort** column, ensure that **Weekday** is selected, in the **By** column, select **DayNumberOfWeek**, and then click **OK**.
5. On the **Ship Date** tab, click the column heading for the **Month** column.
6. On the **Column** menu, point to **Sort**, and then click **Sort by Column**.
7. In the **Sort By Column** dialog box, in the **Sort** column, ensure that **Month** is selected, in the **By** column, select **MonthNumberOfYear**, and then click **OK**.
8. On the **Order Date** tab, click the column heading for the **Weekday** column.
9. On the **Column** menu, point to **Sort**, and then click **Sort by Column**.
10. In the **Sort By Column** dialog box, in the **Sort** column, ensure that **Weekday** is selected, in the **By** column, select **DayNumberOfWeek**, and then click **OK**.
11. On the **Order Date** tab, click the column heading for the **Month** column.
12. On the **Column** menu, point to **Sort**, and then click **Sort by Column**.
13. In the **Sort By Column** dialog box, in the **Sort** column, ensure that **Month** is selected, in the **By** column, select **MonthNumberOfYear**, and then click **OK**.

► Task 4: Create Hierarchies

1. In SQL Server Data Tools, with the Model.bim pane visible, on the **Model** menu, point to **Model View**, and then click **Diagram View**.
2. In the diagram, click the title bar of the **Geography** table, and then click its **Maximize** icon.
3. In the title bar of the maximized **Geography** table, click the **Create Hierarchy** icon, and then rename the new hierarchy to **Location**.

4. In the maximized **Geography** table, drag the **Country or Region** column onto the **Location** hierarchy, drag the **State or Province** column onto the **Location** hierarchy, drag the **City** column onto the **Location** hierarchy, and then drag the **Postal Code** column onto the **Location** hierarchy.
5. Click the **Restore** icon for the **Geography** table, click the title bar of the **Order Date** table, and then click its **Maximize** icon.
6. In the title bar of the maximized **Order Date** table, click the **Create Hierarchy** icon, and then rename the new hierarchy to **Calendar Date**.
7. In the maximized **Order Date** table, drag the following columns onto the **Calendar Date** hierarchy, in the same order:
 - Calendar Year
 - Calendar Quarter
 - Month
 - Day of Month
8. In the title bar of the maximized **Order Date** table, click the **Create Hierarchy** icon, and then rename the new hierarchy to **Fiscal Date**.
9. In the maximized **Order Date** table, drag the following columns onto the **Fiscal Date** hierarchy, in the same order:
 - Fiscal Year
 - Fiscal Quarter
 - Month
 - Day of Month
10. Click the **Restore** icon for the **Order Date** table.

► Task 5: Test the Model

1. In SQL Server Data Tools, with the Model.bim pane open, on the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and then click **OK**.
3. In Excel, in the PivotTable Fields pane, select **Revenue**. The total revenue is shown in the PivotTable.
4. In the **Order Date** table, select the **Calendar Date** hierarchy. The revenue for each calendar year is shown.
5. In the PivotTable, expand **2007** to reveal the quarterly revenue, and then expand quarter **3** to show the monthly revenue. Note that the months are displayed in chronological order.
6. Close Excel without saving the workbook.

Results: After completing this exercise, you should have a tabular data model that includes renamed columns, custom sort orders, and specific relationships between tables.

Exercise 3: Deploying a Tabular Data Model Project in SQL Server Analysis Services

► Task 1: Separate Data into Partitions

1. In SQL Server Data Tools, with the Model.bim pane open in the diagram view, on the **Model** menu, point to **Model View**, and then click **Data View**.
2. On the **Reseller Sales** tab, click the **Table** menu, and then click **Partitions**.
3. In the **Partition Manager** dialog box, ensure that the **Reseller Sales** table is selected, and then click **New**.
4. In the **Partition Name:** box, change the text to **Since 2008**.
5. In the **Partition Name** column, click **Reseller Sales**, and then in the **Partition Name:** box, change the text to **Before 2008**.
6. Click the **Query Editor** icon, and then change the text in the **SQL Statement** box to the following SQL statement:

```
SELECT * FROM FactResellerSales WHERE OrderDateKey<20080101
```

7. In the **Partition Name** column, click **Since 2008**, click the **Query Editor** icon, change the text in the **SQL Statement** box to the following SQL statement, and then click **OK**:

```
SELECT * FROM FactResellerSales WHERE OrderDateKey>=20080101
```

8. On the **Model** menu, point to **Process**, and then click **Process Partitions**.
9. In the **Process Partitions** dialog box, for **Mode**, select **Process Full**, in the **Process** column, select **Before 2008** and **Since 2008**, and then click **OK**. If you are prompted for impersonation credentials, type the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **OK**.
10. Wait until the processing is complete, and then in the **Data Processing** dialog box, observe how many rows were transferred for each partition, and then click **Close**.

► Task 2: Create a Perspective

1. On the **Model** menu, point to **Perspectives**, and then click **Create and Manage**.
2. In the **Perspectives** dialog box, click **New Perspective**, and then type **Products Analysis**.
3. In the **Products Analysis** column, select **Product**, **Product Category**, **Product Subcategory**, **Reseller Sales**, and then click **OK**.
4. On the **Model** menu, click **Analyze in Excel**.
5. In the **Analyze in Excel** dialog box, for **Perspective**, select **Products Analysis**, ensure that **Current Windows User** is selected, and then click **OK**.
6. In Excel, in the PivotTable Fields pane, note that only the tables that you specified for the perspective are visible.
7. Under **Reseller Sales**, select **Revenue**.
8. In the **Product Category** table, click **Category**.
9. Close Excel without saving the workbook.

► Task 3: Create a Security Role

1. On the **Model** menu, click **Roles**.
2. In the **Role Manager** dialog box, click **New**.
3. In the **Name** column, type **US Bike Sales**, and then in the **Permissions** column, select **Read and Process**.
4. In the **DAX Filter** column, for the **Geography** table, type the DAX expression as shown below:

```
= 'Geography' [Country or Region Code] = "US"
```

5. In the **DAX Filter** column, for the **Product Category** table, type the DAX expression as shown below, and then click **OK**:

```
= 'Product Category' [Category] = "Bikes"
```

6. On the **Model** menu, click **Analyze in Excel**.
7. In the **Analyze in Excel** dialog box, click **Role**, and then select the drop-down list below **Role**.
8. Select **US Bike Sales**, and then click **OK**.
9. In the **Analyze in Excel** dialog box, ensure that the **(Default)** perspective is specified, and then click **OK**.
10. In Excel, in the PivotTable Fields pane, under **Reseller Sales**, select **Revenue**.
11. In the **Geography** table, select **Location**.
12. In the **Product Category** table, select **Category**.
13. In the **Product Subcategory** table, select **Subcategory**.
14. Note that reseller data is only showing where **Location** is **United States** and **Category** is **Bikes**.
15. Close Excel without saving the workbook.

► Task 4: Deploy the Reseller Sales Project

1. In SQL Server Data Tools, in Solution Explorer, right-click the **AWSalesTab** project, and then click **Properties**.
2. In the **AWSalesTab Property Pages** dialog box, in the **Deployment Server** section, verify that the **Server** value is **localhost\SQL2**.
3. Change the **Database** property to **AdventureWorksTab**, change the **Cube Name** property to **Reseller Sales**, and then click **OK**.
4. On the **Build** menu, click **Deploy AWSalesTab**. If you are prompted for impersonation credentials, type the user name **ADVENTUREWORKS\ServiceAcct** and the password **Pa\$\$w0rd**, and then click **OK**.
5. In the **Deploy** dialog box, when deployment has completed, click **Close**.
6. Close SQL Server Data Tools.

► Task 5: Use the Deployed Tabular Database

1. Start Excel, and then create a new blank document.
2. On the **Data** tab, in the **Get External Data** area, in the **From Other Sources** drop-down list, select **From Analysis Services**.

3. In the **Data Connection Wizard** dialog box, in the **Server name** box, type **MIA-SQL\SQL2**, ensure that **Use Windows Authentication** is selected, and then click **Next**.
4. On the **Select Database and Table** page, ensure that the **AdventureWorksTab** database and the **Reseller Sales** cube are selected, and then click **Next**.
5. On the **Save Data Connection File and Finish** page, click **Finish**.
6. In the **Import Data** dialog box, ensure that the **Existing Worksheet** option is selected, and then click **OK**.
7. In the PivotTable Fields pane, under **Reseller Sales**, select **Revenue**.
8. In the **Geography** table, select the **Location** hierarchy.
9. Drag the **Fiscal Date** hierarchy in the **Order Date** table to the **Columns** area of the PivotTable Fields pane.
10. Explore the data in the PivotTable. When you have finished, close Excel without saving the workbook.

Results: After this exercise, you should have deployed the tabular data model project.

Module 8: Introduction to Data Analysis Expressions (DAX)

Lab: Using DAX to Enhance a Tabular Data Model

Exercise 1: Creating Calculated Columns

► Task 1: Prepare the Lab Environment

1. Ensure the 20768B-MIA-DC and 20768B-MIA-SQL virtual machines are both running, and then log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab08\Starter folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

► Task 2: Concatenate Text Values

1. Start Visual Studio and open the **AWSalesTab.sln** solution in the D:\Labfiles\Lab08\Starter folder. If the **Tabular model designer** dialog box is displayed, in the **Workspace server list**, select **localhost\SQL2**, and then click **OK**.
2. If the **Tabular Model Designer** dialog box appears, prompting you to run a script on the server, click **Yes**.
3. In Solution Explorer, double-click **Model.bim** to open the data model designer.
4. On the **Model** menu, point to **Process**, and click **Process All**. If the **Impersonation Credentials** dialog box appears, use the following credentials, and then click **OK**:
 - User name: **ADVENTUREWORKS\ServiceAcct**
 - Password: **Pa\$\$w0rd**
5. In the **Data Processing** dialog box, when all of the tables have been processed, click **Close**.
6. On the **Employee** tab, after the existing columns, double-click the **Add Column** header, type **Employee Name**, and then press Enter.
7. Click the **Employee Name** column, click the formula bar, and type the following DAX expression, and then press Enter:

```
=[First Name] & IF(ISBLANK([Middle Name]), "", CONCATENATE(" ", [Middle Name])) &  
CONCATENATE(" ", [Last Name])
```

- Verify that the new column shows the full employee name (including middle name, if present).
8. Click the **First Name** column header, and then hold **Shift** and click the **Middle Name** column header to select the **First Name**, **Middle Name**, and **Last Name** columns. Then right-click any of the selected columns and click **Hide from Client Tools**.
 9. On the **File** menu, click **Save All**.

► Task 3: Calculate a Numeric Value

1. On the **Reseller Sales** tab, in the first empty column to the right of the populated columns, double-click **Add Column**, type **SalesProfit**, and then press Enter.
2. Click the **SalesProfit** column, then click the formula bar, and type the following DAX expression, and then press Enter:

```
=[SalesAmount]-[TotalProductCost]
```

Wait for the table to finish updating, and note the calculated values in the new column.

3. Press F4 and view the Properties pane to verify that the **Data Format** property for this column has been set to **Currency**.

► Task 4: Show Related Values

1. On the **Product** tab, after the existing columns, double-click **Add Column**, type **Subcategory**, and then press Enter.
2. Click **Subcategory**, click the formula bar and type the following DAX expression, and then press Enter:

```
=RELATED('Product Subcategory'[Subcategory])
```

Scroll down to verify that products with a subcategory are shown with the subcategory name. Note that some products at the beginning of the table are not members of a subcategory, so have a blank value in the new column.

3. After the **Subcategory** column, double-click **Add Column**, type **Category**, then press Enter.
4. Click **Category**, click the formula bar, and type the following DAX expression, and then press Enter:

```
=RELATED('Product Category'[Category])
```

Scroll down to verify that products with a category are shown with the category name. Note that some products at the beginning of the table are not members of a category, so have a blank value in the new column.

5. On the **Model** menu, point to **Model View**, and then click **Diagram View**.
6. In the diagram, click the title bar of the **Product** table, then click its **Maximize** icon.
7. On the title bar of the maximized **Product** table, click the **Create Hierarchy** icon, type **Categorized Products**, and then press Enter.
8. Drag the following columns from the column list of the **Product** table to the **Categorized Products** hierarchy:
 - Category
 - Subcategory
 - Product Name
9. On the **Product** table title bar, click the **Restore** icon.
10. On the **Model** menu, point to **Model View**, and then click **Data View**.
11. On the **Product Subcategory** tab, right-click the **Subcategory** column—which is the only visible column—and click **Hide from Client Tools**.

12. In the **Product Category** table, right-click the **Category** column—which is the only visible column—and click **Hide from Client Tools**.
13. On the **File** menu, click **Save All**.

► **Task 5: View Calculated Columns in Excel**

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.
3. When Excel opens, in the **PivotTable Fields** section, under **Reseller Sales**, select the **Revenue** measure.
4. Under **Product**, select **Categorized Products** so that the PivotTable shows sales revenue for each product category.
5. Under **Employee**, drag the **Employee Name** to the **Columns** area so that the PivotTable shows sales revenue for each employee.
6. Expand product categories and subcategories to verify that the hierarchy shows subcategory and product names.
7. Close Excel without saving the workbook.
8. Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have a calculated column named **Employee Name** in the **Employee** table, a calculated measure named **SalesProfit** in the **Reseller Sales** table, and a hierarchy named **Categorized Products** in the **Product** table.

Exercise 2: Creating Measures

► **Task 1: Create a Measure That Aggregates a Column**

1. In Visual Studio, in the data model designer, on the **Reseller Sales** tab, click the first empty cell in the measure grid under the **SalesProfit** column.
2. In the formula bar, type the following DAX expression, and then press Enter:

```
Profit:=SUM([SalesProfit])
```

3. Select the cell containing the **Profit** measure. In the Properties pane, set the **Format** property to **Currency**.
4. Right-click the column header for the **SalesProfit** column and click **Hide from Client Tools**.
5. On the **File** menu, click **Save All**.

► **Task 2: Create a Measure That References Other Measures**

1. On the **Reseller Sales** tab, click the empty cell in the measure grid under the **Profit** measure.
2. In the formula bar, type the following DAX expression, and then press Enter:

```
Margin:=[Profit]/[Revenue]
```

3. Select the cell containing the **Margin** measure, and in the Properties pane, set **Format** to **Percentage**.
4. On the **File** menu, click **Save All**.

► **Task 3: Create a Measure That Uses Time Intelligence**

1. In the data model designer, click the **Order Date** tab.
2. On the **Table** menu, point to **Date**, and then click **Mark As Date Table**.
3. In the **Mark as Date Table** dialog box, in the **Date** list, ensure is **Date** selected, and then click **OK**.
4. On the **Reseller Sales** tab, click the empty cell in the measure grid under the **Revenue** measure.
5. In the formula bar, type the following DAX expression, and then press Enter:

```
Previous Year Revenue:=CALCULATE([Revenue], SAMEPERIODLASTYEAR('Order Date'[Date]))
```

6. Select the cell containing the **Previous Year Revenue** measure, and in the Properties pane, set the **Format** property to **Currency**.
7. On the **File** menu, click **Save All**.

► **Task 4: View Measures in Excel**

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.
3. When Excel opens, in the **PivotTable Fields** section, under **Reseller Sales** table, select the **Revenue**, **Profit**, **Margin**, and **Previous Year Revenue** measures.
4. Under **Order Date**, drag the **Calendar Date** hierarchy to the **Rows** area so that the PivotTable shows revenue, profit, margin, and previous year revenue for each year.
5. Expand 2006 and 2007, and note that the revenue for each quarter in 2006 is shown as the previous year revenue for the same quarters in 2007.
6. Close Excel without saving the workbook. Leave Visual Studio open for the next exercise.

Results: At the end of this exercise, the **Reseller Sales** table should contain the following measures:

Profit

Margin

Previous Year Revenue

Exercise 3: Creating a KPI

► Task 1: Create a Measure to Calculate a KPI Goal

1. In Visual Studio, in the data model designer, on the **Reseller Sales** tab, click the empty cell in the measure grid under the **Previous Year Revenue** measure.
2. In the formula bar, type the following DAX expression, and then press Enter:

```
Revenue Goal:=[Previous Year Revenue] * 1.2
```

3. Select the cell containing the **Revenue Goal** measure, and in the Properties pane, set the **Format** property to **Currency**.
4. Right-click the cell containing the **Revenue Goal** measure, and click **Hide from Client Tools**.
5. On the **File** menu, click **Save All**.

► Task 2: Create a KPI

1. On the **Reseller Sales** tab, right-click the cell in the measure grid containing the **Revenue** measure, and click **Create KPI**.
2. In the **Key Performance Indicator (KPI)** dialog box, note that the **KPI base measure (value)** is defined by the **Revenue** measure. Under **Target**, ensure that **Measure** is selected, and in the list, select the **Revenue** measure.
3. On the red, amber, and green bar, drag the first status threshold slider to **75%** and the second to **95%**. Note the default icon style, and click **OK**. Notice that the **Revenue** measure gains an icon in the measure grid to indicate that it is the basis for a KPI.
4. On the **File** menu, click **Save All**.

► Task 3: View a KPI in Excel

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.
3. In Excel, in the **PivotTable Fields** pane, expand **KPIs**, expand **Revenue Goal**, and then select the **Value**, **Goal**, and **Status** check boxes.
4. Under **Order Date**, drag the **Fiscal Date** hierarchy to the **Rows** area so that the PivotTable shows the KPI data for each year.
5. Expand 2008 and its quarters and months to view the status of the KPI for each month.
6. Close Excel without saving the workbook. Leave Visual Studio open for the next exercise.

Results: At the end of this exercise, the **Reseller Sales** table should include a measure named **Revenue Goal** and a KPI based on the **Revenue** measure.

Exercise 4: Implementing a Parent-Child Hierarchy

► Task 1: Create a Path Column

1. In Visual Studio, in the data model designer, click the **Employee** tab.
2. After the existing columns, double-click **Add Column**, type **Path**, and then press Enter.
3. Click **Path**, click the formula bar, type the following DAX expression, and then press Enter:

```
=PATH([EmployeeKey], [ParentEmployeeKey])
```

Wait for the table to finish updating, and note the calculated values in the new column.

4. On the **File** menu, click **Save All**.

► Task 2: Create a Column for Each Hierarchy Level

1. In the **Employee** table, in the first empty column to the right of the populated columns, double-click **Add Column**, type **Level1**, then press Enter.
2. Click **Level1**, click the formula bar, type the following DAX expression, and then press Enter:

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 1, 1))
```

3. In the first empty column to the right, double-click **Add Column**, type **Level2**, then press Enter.
4. Click **Level2**, click the formula bar, type the following DAX expression, and then press Enter:

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 2, 1))
```

5. In the first empty column to the right, double-click **Add Column**, type **Level3**, then press Enter.
6. Click **Level3**, click the formula bar, type the following DAX expression, and then press Enter:

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 3, 1))
```

7. In the first empty column to the right, double-click **Add Column**, type **Level4**, then press Enter.
8. Click **Level4**, click the formula bar, type the following DAX expression, and then press Enter:

```
=LOOKUPVALUE ([Employee Name], [EmployeeKey], PATHITEM ([Path], 4, 1))
```

9. On the **File** menu, click **Save All**.

► Task 3: Use the Calculated Columns in a Hierarchy

1. On the **Model** menu, point to **Model View**, and then click **Diagram View**.
2. Click the **Employee** table, and then click the **Maximize** button.
3. In the maximized **Employee** table, click **Create Hierarchy**, type **Employee Hierarchy**, and then press Enter.
4. Drag the following columns from the column list of the **Employee** table to the **Employee Hierarchy**, in the following order:
 - a. Level1
 - b. Level2
 - c. Level3
 - d. Level4

5. Click the **Path** column, hold **Shift** and click the **Level4** column. Then right-click the selected columns and click **Hide from Client Tools**.
6. In the title bar, click **Restore**.
7. On the **File** menu, click **Save All**.

► **Task 4: View a Parent-Child Hierarchy in Excel**

1. On the **Model** menu, click **Analyze in Excel**.
2. In the **Analyze in Excel** dialog box, ensure that **Current Windows User** is selected and that the **(Default)** perspective is specified, and click **OK**.
3. In Excel, in the **PivotTable Fields** section, under **Reseller Sales**, select the **Revenue** measure.
4. Under **Employee**, select **Employee Hierarchy** so that the PivotTable shows revenue for the top-level employee.
5. Expand the employee hierarchy to view sales totals for managers and their subordinates. Note that the personal revenue for sales managers is shown with a blank employee name under the total for that sales manager.
6. Close Excel without saving the workbook.
7. Close Visual Studio.

Results: At the end of this exercise, the **Employee** table should include a hierarchy named **Employee Hierarchy**.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 9: Performing Predictive Analysis with Data Mining

Lab: Using Data Mining to Support a Marketing Campaign

Exercise 1: Creating a Data Mining Structure and Models

► Task 1: Prepare the Lab Environment

1. Ensure that the 20768B-MIA-DC, and 20768B-MIA-SQL virtual machines are running.
2. Log on to 20768B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
3. In the D:\Labfiles\Lab09\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
4. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

► Task 2: Create a Data Mining Project

1. Start Visual Studio, and on the **File** menu, point to **New**, and click **Project**.
2. In the **New Project** dialog box, expand **Business Intelligence**, then click **Analysis Services**, then click **Analysis Services Multidimensional and Data Mining Project**. Type **AW Data Mining** in the **Name** box, then type **D:\Labfiles\Lab09\Starter** in the **Location** box, and then click **OK**.
3. In Solution Explorer, right-click **Data Sources**, and then click **New Data Source**.
4. In the **Data Source Wizard**, on the Welcome to the Data Source Wizard page, click **Next**.
5. On the Select how to define the connection page, click **New**. Then, in the **Connection Manager** dialog box, in the **Server name** field, type **MIA-SQL**, in the **Select or enter a database name** drop-down list, select **AdventureWorksDW**, and click **OK**.
6. In the **Data Source Wizard**, on the Select how to define the connection page, click **Next**.
7. On the Impersonation Information page, click **Use the service account**, and then click **Next**.
8. On the Completing the Wizard page, click **Finish**.
9. In Solution Explorer, right-click **Data Source Views**, and then click **New Data Source View**.
10. In the **Data Source View Wizard**, on the Welcome to the Data Source View Wizard page, click **Next**.
11. On the Select a Data Source page, ensure that **Adventure Works DW** is selected, and then click **Next**.
12. On the Select Tables and Views page, in the **Available objects** list, click **ProspectiveBuyer (dbo)**, hold the **Ctrl** key. Click **vTargetMail (dbo)** and click the > button to move the selected objects to the **Included objects** list, and then click **Next**.
13. On the Completing the Wizard page, in the **Name** field, type **Adventure Works DW DM View**, and then click **Finish**.

► Task 3: Create a Data Mining Structure and a Data Mining Model

1. In Solution Explorer, right-click **Mining Structures**, click **New Mining Structure**.
2. In the **Data Mining Wizard**, on the Welcome to the Data Mining Wizard page, click **Next**.
3. On the Select the Definition Method page, ensure that **From existing relational database or data warehouse** is selected, and then click **Next**.

4. On the Create the Data Mining Structure page, ensure that **Create mining structure with mining model** is selected and select the **Microsoft Decision Trees** data mining technique. Then click **Next**.
5. On the Select a Data Source View page, select **Adventure Works DW DM View** and click **Next**.
6. On the Specify Table Types page, in the **vTargetMail** row, select the check box in the **Case** column, and then click **Next**.
7. On the Specify the Training Data page, in the **Mining model structure** table, in the **BikeBuyer** row, select the check box in the **Predictable** column. In the **CustomerKey** row, select the check box in the **Key** column; for all other rows, select the **Input** check box, and then click **Next**.
8. On the Specify Columns' Content and Data Type page, click **Detect**. Ensure that the content type of the **Bike Buyer** column is identified as **Discrete**, and change the **Yearly Income** column to **Discrete**. Then click **Next**.
9. On the Create Testing Set page, note that the **Percentage of data for testing** value is 30 percent, and then click **Next**.
10. On the Completing the Wizard page, in the **Mining structure name** field, type **Purchase Prediction**. In the **Mining model name** field, type **Purchase Decision Tree**, and then click **Finish**.
11. On the menu bar, click **Build**, and then click **Deploy AW Data Mining**.
12. When deployment is complete, close the **Deployment Progress - AW Data Mining** window.

► **Task 4: Add a Data Mining Model to a Data Mining Structure**

1. In Visual Studio Solution Explorer, expand **Mining Structures** and double-click **Purchase Prediction.dmm**.
2. In the Purchase Prediction.dmm [Design] window, click **Mining Models**. Right-click anywhere in the **Mining Models** tab, then click **New Mining Model**.
3. In the **New Mining Model** dialog box, in the **Model name** box, type **Purchase - Bayes**. Select **Microsoft Naive Bayes** in the **Algorithm name** box, and then click **OK**.
4. In the **Microsoft Visual Studio** dialog box, read the information message about content types unsupported by the Microsoft Naïve Bayes algorithm, then click **Yes**.
5. In the **Mining Models** tab, in the **Purchase - Bayes** column, in the **Name Style** row, select **Ignore** to exclude the column from the model.
6. On the menu bar, click **Build**, and then click **Deploy AW Data Mining**.
7. When deployment is complete, close the Deployment Progress - AW Data Mining window.
8. Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have a data mining structure with two data models.

Exercise 2: Explore a Data Mining Model

► Task 1: Review Data Models

1. In Solution Explorer expand **Mining Structures**, right-click **Purchase Prediction.dmm**, and then click **Browse**.
2. In the **Mining Model** drop-down list, ensure that **Purchase Decision Tree** is selected, and on the **Dependency Network** tab, move the slider gradually from **All Links** to **Strongest Links** to see which factors are the strongest predictors that a customer will purchase a bike. You might need to move the Mining Legend window to be able to see the **Dependency Network** tab.
3. On the **Decision Tree** tab, in the **Background** drop-down list, select **1**. Then view the decision tree to see how the other column values influence a value of **1** for **Bike Buyer**.
4. In the **Mining Model** drop-down list, select **Purchase - Bayes**, and on the **Attribute Profiles** tab, view the color-coded indicators of the values for each column when compared to customers with a **Bike Buyer** value of **1** or **0**.
5. On the **Attribute Characteristics** tab, in the **Attribute** drop-down list, ensure that **Bike Buyer** is selected, and in the **Value** drop-down list, select **1**. Then view the probability for each of the other column values when the **Bike Buyer** value is **1**.
6. On the **Attribute Discrimination** tab, in the **Attribute** drop-down list, ensure that **Bike Buyer** is selected. In the **Value 1** drop-down list, select **1**, and in the **Value 2** drop-down list, select **0**. Then note how values for all other columns favor a particular **Bike Buyer** value.
7. Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have reviewed the output of data mining models using Visual Studio.

Exercise 3: Validating Data Mining Models

► Task 1: Review a Lift Chart

1. In Visual Studio, in the Purchase Prediction.dmm [Design] pane, click **Mining Accuracy Chart**.
2. On the **Input Selection** tab, verify that the **Show** box is selected for both the **Purchase Decision Tree** and **Purchase - Bayes** models, and that **Use mining model test cases** is selected in the **Select data set to be used for Accuracy Chart** section.
3. On the **Lift Chart** tab, review the lift chart, and the detailed data in the Mining Legend window.
The **Purchase Decision Tree** model is closer to the **Ideal Model**, and can therefore be considered more accurate.

► Task 2: Review a Profit Chart

1. On the **Lift Chart** tab, in the **Chart type** drop-down list, select **Profit Chart**. You might need to move the Mining Legend window to see the **Chart type** list.
2. In the **Profit Chart Settings** dialog box, enter the following values to reflect a marketing campaign you are planning, and then click **OK**:
 - **Population:** 5,000 (this is the number of potential customers you plan to contact).
 - **Fixed cost:** 500 (this is the fixed cost of your marketing campaign).

- **Individual cost:** 1 (this is the cost associated with contacting each customer).
 - **Revenue per individual:** 150 (this is the amount you expect a customer to spend if they respond positively to the campaign).
3. Review the chart and the Mining Legend window to evaluate which mining model is likely to generate the most profitable marketing campaign based on the test data.

► Task 3: Review a Classification Matrix

1. Click the **Classification Matrix** tab.
2. Review the matrix, noting that for each model it shows the number of times the model predicted a **Bike Buyer** value of **1** or **0** on rows, with columns for the actual value of the **Bike Buyer** column in the test data.

► Task 4: Review a Cross-Validation Report

1. Click the **Cross Validation** tab.
2. Enter the following values, and click **Get Results**:
 - **Fold Count:** 5 (this is the number of partitions used to group the data for analysis).
 - **Max Cases:** 500 (this is the number of cases to be analyzed).
 - **Target Attribute:** Bike Buyer (this is the predictable column to be evaluated).
 - **Target State:** 1 (this is the desired value for the target attribute).
 - **Target Threshold:** 0.7 (this is a value between 0 and 1 that indicates the level of accuracy required for a prediction to be considered correct).
3. View the resulting report, and note that for each mining model, the results include the following:
 - Classifications for true positives, false positives, true negatives, and false negatives.
 - The likely lift gained by using the model.
4. Leave Visual Studio open for the next exercise.

Results: After this exercise, you should have two validated data mining models in your data mining structure.

Exercise 4: Using a Data Mining Model in a Report

► Task 1: Create a Report

1. In Visual Studio, on the **File** menu, point to **New** and click **Project**.
2. In the **New project** dialog box, in the Templates pane, expand **Business Intelligence**, then select **Report Server Project Wizard**. In the **Name** field, type **Promotion Targeting**, in the **Location** field, browse to the D:\Labfiles\Lab09\Starter folder and click **Select Folder**, and then click **OK**.
3. In the **Report Wizard**, on the Welcome to the Report Wizard page, click **Next**.
4. On the Select the Data Source page, select **New data source**, change the **Name** to **AWDataMining**. In the **Type** drop-down list, select **Microsoft SQL Server Analysis Services**, and click **Edit**.
5. In the **Connection Properties** dialog box, in the **Server** name field, type **MIA-SQL**, in the **Select or enter a database name** drop-down list, click **AW Data Mining**, and then click **OK**.

6. On the Select the Data Source page, click **Next**.
7. On the Design the Query page, click **Query Builder**.
8. In **Query Designer**, in the Mining Model pane, click **Select Model**. Then in the **Select Mining Model** dialog box, expand **Purchase Prediction**, click **Purchase - Bayes**, and click **OK**.
9. In the Select Input Table(s) pane, click **Select Case Table**. Then in the **Select Table** dialog box, click **ProspectiveBuyer (dbo)**, and click **OK**.
10. In the table at the bottom of the query designer, in the **Source** column, select **ProspectiveBuyer table**, and then in the **Field** column, in the drop-down list, click **FirstName**.
11. Repeat step 10 to add five more rows to the table by using the settings in the following table, and then click **OK**:

Source	Field	Alias	Criteria/Argument
ProspectiveBuyer table	LastName		
ProspectiveBuyer table	Address Line 1		
ProspectiveBuyer table	City		
Purchase – Bayes mining model	Bike Buyer		=1
Prediction Function	PredictProbability	Purchase Probability	[Purchase - Bayes].[Bike Buyer]

12. On the Design the Query page, click **Next**.
13. On the Select the Report Type page, select **Tabular**, and click **Next**.
14. On the Design the Table page, move all the columns into the **Details** area, and click **Next**.
15. On the Choose the Table Style page, select any style and click **Next**.
16. On the Choose the Deployment Location page, click **Next**.
17. On the Completing the Wizard page, change the **Report name** to **Potential Bike Buyers** and click **Finish**.
18. In the Row Groups pane, click the **(table1_Details_Group)** drop-down list and click **Group Properties**.
19. On the **Sorting** tab, click **Add**, in the **Sort by** drop-down list, click **[Purchase_Probability]**, and in the **Order** drop-down list, click **Z to A**. Then click **OK**.
20. In the report design pane, in the **Purchase Probability** column, right-click **[Purchase Probability]** and click **Text Box Properties**. Then in the **Text Box Properties** dialog box, click **Number**, select **Percentage**, and click **OK**.
21. Click the **Preview** tab to review the **Potential Bike Buyers** report.
22. When you have finished working on the report, close Visual Studio.

Results: After this exercise, you should have an SSRS report that predicts bike purchasers.

MCT USE ONLY. STUDENT USE PROHIBITED