Contents

Tutorials for SQL Server 2014

Tutorial: Writing Transact-SQL Statements +

Data Mining Tutorials

- Basic Data Mining Tutorial
  - Lesson 1: Preparing the Analysis Services Database
    - Creating an Analysis Services Project
    - Creating a Data Source
    - Creating a Data Source View
  - Lesson 2: Building a Targeted Mailing Structure
    - Creating a Targeted Mailing Mining Model Structure
    - Specifying the Data Type and Content Type
    - Specifying a Testing Data Set for the Structure
  - Lesson 3: Adding and Processing Models
    - Adding New Models to the Targeted Mailing Structure
    - Processing Models in the Targeted Mailing Structure
  - Lesson 4: Exploring the Targeted Mailing Models
    - Exploring the Decision Tree Model
    - Exploring the Clustering Model
    - Exploring the Naive Bayes Model
  - Lesson 5: Testing Models
    - Testing Accuracy with Lift Charts
    - Testing a Filtered Model
  - Lesson 6: Creating and Working with Predictions
    - Creating Predictions
    - Using Drillthrough on Structure Data
- Intermediate Data Mining Tutorial
  - Lesson 1: Creating the Intermediate Data Mining Solution
    - Creating a Solution and Data Source
  - Lesson 2: Building a Forecasting Scenario

Adding a Data Source View for Forecasting
Understanding the Requirements for a Time Series Model
Creating a Forecasting Structure and Model
Modifying the Forecasting Structure
Customizing and Processing the Forecasting Model
Exploring the Forecasting Model
Creating Time Series Predictions
Advanced Time Series Predictions
Time Series Predictions using Updated Data
Time Series Predictions using Replacement Data
Comparing Predictions for Forecasting Models
Lesson 3: Building a Market Basket Scenario
Adding a Data Source View with Nested Tables
Creating a Market Basket Structure and Model
Modifying and Processing the Market Basket Model
Exploring the Market Basket Models
Filtering a Nested Table in a Mining Model
Predicting Associations
Lesson 4: Building a Sequence Clustering Scenario
Creating a Sequence Clustering Mining Model Structure
Processing the Sequence Clustering Model
Exploring the Sequence Clustering Model
Creating a Related Sequence Clustering Model
Creating Predictions on a Sequence Clustering Model
Lesson 5: Building Neural Network and Logistic Regression Models
Adding a Data Source View for Call Center Data
Creating a Neural Network Structure and Model
Exploring the Call Center Model
Adding a Logistic Regression Model to the Call Center Structure
Creating Predictions for the Call Center Models (Intermediate Data Mining Tutorial)
Creating and Querying Data Mining Models with DMX: Tutorials
Bike Buyer DMX Tutorial

Lesson 1: Creating the Bike Buyer Mining Structure
Lesson 2: Adding Mining Models to the Bike Buyer Mining Structure
Lesson 3: Processing the Bike Buyer Mining Structure
Lesson 4: Browsing the Bike Buyer Mining Models
Lesson 5: Executing Prediction Queries
Market Basket DMX Tutorial
Lesson 1: Creating the Market Basket Mining Structure
Lesson 2: Adding Mining Models to the Market Basket Mining Structure
Lesson 3: Processing the Market Basket Mining Structure
Lesson 4: Executing Market Basket Predictions
Time Series Prediction DMX Tutorial
Lesson 1: Creating a Time Series Mining Model and Mining Structure
Lesson 2: Adding Mining Models to the Time Series Mining Structure
Lesson 3: Processing the Time Series Structure and Models
Lesson 4: Creating Time Series Predictions Using DMX
Lesson 5: Extending the Time Series Model
Reporting Services Tutorials
Create a Basic Table Report (SSRS Tutorial)
Lesson 1: Creating a Report Server Project
Lesson 2: Specifying Connection Information
Lesson 3: Defining a Dataset for the Table Report
Lesson 4: Adding a Table to the Report
Lesson 5: Formatting a Report
Lesson 6: Adding Grouping and Totals
Create a Data-Driven Subscription (SSRS Tutorial)
Lesson 1: Creating a Sample Subscriber Database
Lesson 2: Modifying the Report Data Source Properties
Lesson 3: Defining a Data-Driven Subscription
Create a Drillthrough (RDLC) Report with Parameters using ReportViewer (SSRS Tutorial)
Lesson 1: Create a New Web Site
Lesson 2: Define a Data Connection and Data Table for Parent Report
Lesson 3: Design the Parent Report using the Report Wizard

Lesson 4: Define a Data Connection and Data Table for Child Report
Lesson 5: Design the Child Report using the Report Wizard
Lesson 6: Add a ReportViewer Control to the Application
Lesson 7: Add Drillthrough Action on Parent Report
Lesson 8: Create a Data Filter
Lesson 9: Build and Run the Application
Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)
Lesson 1: Creating the Web Service Client Project
Lesson 2: Adding a Web Reference
Lesson 3: Accessing the Web Service
Lesson 4: Running the Application (VB/VC#)
Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)
Lesson 1: Create the RDL Schema Visual Studio Project
Lesson 2: Generate Classes from the RDL Schema using the xsd Tool
Lesson 3: Load a Report Definition from the Report Server
Lesson 4: Update the Report Definition Programmatically
Lesson 5: Publish the Report Definition to the Report Server
Lesson 6: Run the RDL Schema Application (VB/C#)
Tutorials (Report Builder)
Prerequisites for Tutorials
Alternative Ways to Get a Data Connection (Report Builder)
Tutorial: Creating a Basic Table Report
Tutorial: Creating a Matrix Report
Tutorial: Creating a Free Form Report
Tutorial: Format Text (Report Builder)
Tutorial: Add a Column Chart to Your Report (Report Builder)
Tutorial: Add a Pie Chart to Your Report (Report Builder)
Tutorial: Add a Bar Chart to Your Report (Report Builder)
Tutorial: Add a Sparkline to Your Report (Report Builder)
Tutorial: Adding a KPI to Your Report
Tutorial: Map Report
Tutorial: Add a Parameter to Your Report (Report Builder)

Tutorial: Creating Drillthrough and Main Reports
Tutorial: Introducing Expressions
Enterprise Information Management Tutorials
Enterprise Information Management using SSIS, MDS, and DQS Together [Tutorial]
Lesson 1: Creating the Suppliers DQS Knowledge Base
Task 1: Creating a Knowledge Base and Domains
Task 2: Adding Domain Values Manually
Task 3: Importing Domain Values from an Excel File
Task 4: Setting Domain Rules
Task 5: Setting Term Based Relationships
Task 6: Setting Synonyms
Task 7: Creating a Composite Domain
Task 8: Creating a Composite Domain Rule
Task 9: Configuring a Reference Data Service
Task 10: Configuring Composite Domain to Use Reference Data Service
Task 11: Publishing the Knowledge Base
Task 12: Discovering Knowledge (Knowledge Discovery)
Lesson 2: Cleansing Supplier Data using the Suppliers Knowledge Base
Task 1: Creating a Data Quality Project
Task 2: Mapping Excel Columns to DQS Domains
Task 3: Cleansing Data against the Suppliers Knowledge Base
Task 4: Managing and Viewing Results
Task 5: Exporting Cleansing Results to an Excel File
Task 6: Importing Values from the Cleanse Supplier List Project
Lesson 3: Matching Data to Remove Duplicates from Supplier List
Task 1: Defining a Matching Policy
Task 2: Testing and Publishing the Matching Policy
Task 3: Creating and Running a Data Quality Project for Matching
Task 4: Exporting the Results from Matching Activity to an Excel File
Lesson 4: Storing Supplier Data in MDS
Task 1: Creating Suppliers Model using Master Data Manager
Task 2: Uploading Supplier Data to MDS using MDS Add-in for Excel

Task 3: Verifying the Data in Master Data Manager
Task 4 (Optional): Combining, Matching, and Publishing New Set of Data
Task 5: Creating a Domain-Based Attribute from Excel
Task 6: Verify that the Domain-Based Attribute is Created using Master Data Manager
Task 7: Viewing Updates Made using Master Data Manager in Excel
Task 8: Adding a New Value for State Entity in Excel
Task 9: Creating a Derived Hierarchy using Master Data Manager
Lesson 5: Automating the Cleansing and Matching using SSIS
Task 1 (Prerequisite): Removing Supplier Data in MDS
Task 2 (Optional): Creating a MDS Subscription View using Master Data Manager
Task 3 (Optional): Reviewing the Subscription Views
Task 4: Creating an SSIS Project using SQL Server Data Tools
Task 5: Adding Data Flow Task
Task 6: Adding Excel Source to the Data Flow
Task 7: Adding DQS Cleansing Transform to the Data Flow
Task 8: Adding Conditional Split Transform to Split Cleansing Output
Task 9: Adding Union All Transform to Combine Correct and Corrected Records
Task 10: Adding Fuzzy Group Transform to Identify Duplicates
Task 11: Adding Conditional Split Transform to Filter Duplicates
Task 12: Adding Derived Column Transform to Add Columns Required by MDS
Task 13: Adding OLE DB Destination to Write Data to MDS Staging Table
Task 14: Adding Execute SQL Task to Control Flow to Run the Stored Procedure for MDS
Task 15: Building and Running the SSIS Project
Task 16: Verifying with Master Data Manager
Task 17: Reviewing DQS Cleansing Project Created by the SSIS package
Conclusion

Tutorials for SQL Server 2014

2/11/2019 • 2 minutes to read • Edit Online

SQL Server 2014 introduces many new technologies and features.

There are plans to provide additional tutorials in the future. Meanwhile, you can continue to learn from the SQL Server 2008 R2 tutorials. In some cases, you might find minor differences between the user interface in SQL Server 2014 and SQL Server 2008 R2 and there might be subtle differences in tutorial prerequisites. If a tutorial uses one of the Adventure Works databases, you should download and use the SQL Server 2014 version.

If you are using a SQL Server 2014 server, you have the SQL Server 2014 data providers installed. In order for the tutorials to work, you will either need the SQL Server 2014 sample databases that work with those providers, or you need the SQL Server 2008 R2 data providers that work with the SQL Server 2008 R2 version of the sample databases.

- To download SQL Server 2014 samples to use with tutorials, go to Samples for SQL Server on CodePlex.
- To download a PowerPivot workbook and Power View reports to use with Power View tutorials, go to Image Files for HelloWorldPicnic Sample Model in PowerPivot and Power View.

> **NOTE**
> Power View is a feature of SQL Server 2014Reporting Services Add-in for MicrosoftSharePoint Server 2010 Enterprise Edition.

- To download SQL Server 2014 versions of Adventure Works databases, go to Adventure Works 2014 Sample Databases on CodePlex.
- To download SQL Server 2012 versions of Adventure Works databases, go to Adventure Works for SQL Server 2012 on CodePlex.
- To see a list of SQL Server 2008 R2 tutorials, go to Tutorials in MSDN library.

In This Section

Tutorials are available for these SQL Server 2014 technologies.

TECHNOLOGY	DESCRIPTION
Analysis Services Tutorials (SSAS)	Learn how to develop and deploy Analysis Services tabular or multidimensional models. Also learn how to use tools such as PowerPivot to analyze the data in the models.
Reporting Services Tutorials (SSRS)	Learn how to create basic reports and subscriptions.
Integration Services Tutorials	Learn how to create and deploy SSIS packages.
Database Engine Tutorials	Learn how to connect to an instance of the database engine and then build and run Transact-SQL statements.
Replication Tutorials	Learn how to set up and run replication topologies using SQL Server Management Studio.

TECHNOLOGY	DESCRIPTION
Enterprise Information Management Tutorials	Learn how to manage information in an enterprise using Microsoft EIM technologies.

Tutorial: Writing Transact-SQL Statements

12/3/2018 • 2 minutes to read • Edit Online

Welcome to the Writing Transact-SQL Statements tutorial. This tutorial is intended for users who are new to writing SQL statements. It will help new users get started by reviewing some basic statements for creating tables and inserting data. This tutorial uses Transact-SQL, the Microsoft implementation of the SQL standard. This tutorial is intended as a brief introduction to the Transact-SQL language and not as a replacement for a Transact-SQL class. The statements in this tutorial are intentionally simple, and are not meant to represent the complexity found in a typical production database.

> **NOTE**
>
> Novice users of databases will usually find it easier to work with SQL Server by using SQL Server Management Studio, instead of writing Transact-SQL statements.

Finding More Information

To find more information about any specific statement, either search for the statement by name in SQL Server Books Online, or use the Contents to browse the 1,800 language elements listed alphabetically under Transact-SQL Reference (Database Engine). Another good strategy for finding information is to search for key words that are related to the subject matter you are interested in. For example, if you want to know how to return a part of a date (such as the month), search the index for **dates [SQL Server]**, and then select **dateparts**. This takes you to the topic DATEPART (Transact-SQL). As another example, to find out how to work with strings, search for **string functions**. This takes you to the topic String Functions (Transact-SQL).

What You Will Learn

This tutorial shows you how to create a database, create a table in the database, insert data into the table, update the data, read the data, delete the data, and then delete the table. You will create views and stored procedures and configure a user to the database and the data.

This tutorial is divided into three lessons:

Lesson 1: Creating Database Objects
In this lesson, you create a database, create a table in the database, insert data into the table, update the data, and read the data.

Lesson 2: Configuring Permissions on Database Objects
In this lesson, you create a login and user. You will also create a view and a stored procedure, and then grant the user permission to the stored procedure.

Lesson 3: Deleting Database Objects
In this lesson, you remove access to data, delete data from a table, delete the table, and then delete the database.

Requirements

To complete this tutorial, you do not have to know the SQL language, but you should understand basic database concepts such as tables. During this tutorial, you will create a database and create a Windows user. These tasks require a high level of permissions; therefore, you should log in to the computer as an administrator.

Your system must have the following installed:

- Any edition of SQL Server.
- Either SQL Server Management Studio or Management Studio Express.
- Internet Explorer 6 or later.

> **NOTE**
>
> When you review the tutorials, we recommend that you add the **Next** and **Previous** buttons to the document viewer toolbar.

Data Mining Tutorials (Analysis Services)

12/13/2018 • 2 minutes to read • Edit Online

Microsoft SQL Server Analysis Services makes it easy to create sophisticated data mining solutions. The tools in Analysis Services help you design, create, and manage data mining models that use either relational or cube data. You can manage client access to data mining models and create prediction queries from multiple clients.

The step-by-step tutorials in the following list will help you learn how to get the most out of Analysis Services, so that you can perform advanced analysis to solve business problems that are beyond the reach of traditional business intelligence methods.

In This Section

- Basic Data Mining Tutorial

  This tutorial walks you through a targeted mailing scenario. It demonstrates how to use the data mining algorithms, mining model viewers, and data mining tools that are included in Analysis Services. You will build three data mining models to answer practical business questions while learning data mining concepts and tools.

- Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

  This tutorial contains a collection of lessons that introduce more advanced data mining concepts and techniques. The scenarios include these model types:

  - forecasting
  - market basket analysis
  - neural networks and logistic regression
  - sequence clustering

  The lessons are independent and can be done in any order, but you should have a basic knowledge of how to build data sources.

  Advanced concepts covered in these lessons include the use of nested tables, cross-prediction, custom data source views and named queries, and filtering in data mining queries. You will also gain proficiency in using the prediction query tools that are included in Analysis Services.

Reference

Data Mining Algorithms (Analysis Services - Data Mining)

Data Mining Extensions (DMX) Reference

Related Sections

Data Mining Tools

Logical Architecture (Analysis Services - Data Mining)

Logical Architecture (Analysis Services - Multidimensional Data)

Data Mining Projects

See Also

Data Mining Solutions
Microsoft SQL Server Data Mining resources
Creating and Querying Data Mining Models with DMX: Tutorials (Analysis Services - Data Mining)

Basic Data Mining Tutorial

2/11/2019 • 3 minutes to read • Edit Online

Welcome to the Microsoft Analysis Services Basic Data Mining Tutorial. Microsoft SQL Server provides an integrated environment for creating data mining models and making predictions. In this tutorial, you will complete a scenario for a targeted mailing campaign in which you use machine learning to analyze and predict customer purchasing behavior. The tutorial demonstrates how to use three of the most important data mining algorithms: clustering, decision trees, and Naive Bayes. You will also learn how to analyze your findings using the mining model viewers, and to create predictions and accuracy charts using the data mining tools that are included in Microsoft SQL Server Analysis Services. The fictitious company, Adventure Works Cycles, is used for all examples.

When you are comfortable using the data mining tools, we recommend that you also complete the Intermediate Data Mining Tutorial (Analysis Services - Data Mining). The lessons demonstrate how to use forecasting, market basket analysis, time series, association models, nested tables, and sequence clustering.

Tutorial Scenario

In this tutorial, you are an employee of Adventure Works Cycles who has been tasked with learning more about the company's customers based on historical purchases, and then using that historical data to make predictions that can be used in marketing. The company has never done data mining before, so you must create a new database specifically for data mining and set up several data mining models.

What You Will Learn

This tutorial teaches you how to create and work with several different types of machine learning methods. You will also learn how to create a copy of a mining model, and apply a filter to the input data to get different results. Afterwards you can compare the results of both models using a lift chart. Finally, you will use drillthrough to retrieve additional data from the underlying mining structure.

Microsoft Analysis Services Data Mining includes the following features that help you easily develop and compare multiple predictive models and then take actions on the results :

- *Holdout Test Sets* -When you create a mining structure, you can now divide the data in the mining structure into training and testing sets. This lets you test models on similar data sets, and compare the accuracy of related models.
- *Mining model filters* -You can now attach filters to a mining model, and apply the filter during both training and testing. This lets you easily build related models on different subsets of the data.
- *Drillthrough to Structure Cases and Structure Columns* - You can now easily move from the general patterns in the mining model to actionable detail in the data source.

This tutorial is divided into the following lessons:

Lesson 1: Preparing the Analysis Services Database (Basic Data Mining Tutorial)
In this lesson, you will learn how to create a new Analysis Services database, add a data source and data source view, and prepare the new database to be used with data mining.

Lesson 2: Building a Targeted Mailing Structure (Basic Data Mining Tutorial)
In this lesson, you will learn how to create a mining model structure that can be used as part of a targeted mailing scenario.

Lesson 3: Adding and Processing Models
In this lesson you will learn how to add models to a structure. The models you create are built with the following algorithms:

- Microsoft Decision Trees
- Microsoft Clustering
- Microsoft Naive Bayes

Lesson 4: Exploring the Targeted Mailing Models (Basic Data Mining Tutorial)
In this lesson you will learn how to explore and interpret the findings of each model using the Viewers.

Lesson 5: Testing Models (Basic Data Mining Tutorial)
In this lesson, you make a copy of one of the targeted mailing models, add a mining model filter to restrict the training data to a particular set of customers, and then assess the viability of the model.

Lesson 6: Creating and Working with Predictions (Basic Data Mining Tutorial)
In this final lesson of the Basic Data Mining Tutorial, you use the model to predict which customers are most likely to purchase a bike. You then drill through to the underlying cases to obtain contact information.

Requirements

Make sure that the following are installed:

- Microsoft SQL Server 2014
- Microsoft SQL Server Analysis Services in multidimensional mode
- The **AdventureWorksDW2012** database.

To enhance security, the sample databases are not installed with SQL Server. To install the official databases for Microsoft SQL Server, visit the Microsoft SQL Sample Databases page and select SQL Server 2014.

> **NOTE**
> When you are working through a tutorial, you might find it easier to move back and forth between the steps if you add the **Next topic** and **Previous topic** buttons to the document viewer toolbar.

See Also

Data Mining Solutions
Mining Model Tasks and How-tos
Creating and Querying Data Mining Models with DMX: Tutorials (Analysis Services - Data Mining)

Lesson 1: Preparing the Analysis Services Database (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

You are a new employee of Adventure Works Cycles who has been tasked with designing a business intelligence application in SQL Server 2014. Adventure Works Cycles hopes to leverage your Analysis Services data mining experience to discover interesting and actionable information about people who have purchased bicycles. They then want you to predict which prospective customers are most likely to purchase a bicycle in the future.

Designing this application in SQL Server starts with the creation in SQL Server Data Tools (SSDT) of a SQL Server Analysis Services project based on the Analysis Services project template for multidimensional modeling and data mining. After you create an Analysis Services project, you define one or more data sources. Then, you define a view of the metadata, called a *data source view*, from selected tables and views from the data sources.

In this lesson, you will create an Analysis Services project, define a single data source, and add a subset of tables to a data source view. This lesson includes the following tasks:

Creating an Analysis Services Project (Basic Data Mining Tutorial)

Creating a Data Source (Basic Data Mining Tutorial)

Creating a Data Source View (Basic Data Mining Tutorial)

First Task in Lesson

Creating an Analysis Services Project (Basic Data Mining Tutorial)

Next Lesson

Lesson 2: Building a Targeted Mailing Structure (Basic Data Mining Tutorial)

See Also

Data Source Views in Multidimensional Models
Data Sources Supported (SSAS Multidimensional)
Build Analysis Services Projects (SSDT)
Creating an Analysis Services Project

Creating an Analysis Services Project (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Each Microsoft SQL Server Analysis Services project defines the objects in a single Analysis Services database. An Analysis Services database can contains many different types of objects

- Multidimensional models (cubes)
- Mining structures and mining models
- Supporting objects such as data sources, data source views, and custom assemblies

Note that you **do not** require a cube to do data mining. If you need to perform data mining on an existing cube, you should add the data mining models to the same project you used to build the cube. However, for most purposes you can build your models on relational data sources, such as a data warehouse, and get better performance if a cube is not involved.

In this tutorial you will use a relational data warehouse, **AdventureWorksDW2012**, as the data source. You will deploy all your data mining objects to an Analysis Services database named `BasicDataMining`, used just for data mining.

By default, Analysis Services uses the **localhost** instance for new projects. If you are using a named instance or a different server, you must first create and open the project and then change the instance name.

For more information about Analysis Services projects, see Creating an Analysis Services Project.

To create an Analysis Services project

1. Open SQL Server Data Tools (SSDT).
2. On the **File** menu, point to **New**, and then select **Project**.
3. Verify that **Business Intelligence Projects** is selected in the **Project types** pane.
4. In the **Templates** pane, select **Analysis Services Multidimensional and Data Mining Project**.
5. In the **Name** box, name the new project `BasicDataMining`.
6. Click **OK**.

To change the instance where data mining objects are stored

1. In SQL Server Data Tools (SSDT), on the **Project** menu, select **Properties**.
2. On the left side of the **Property Pages** pane, under **Configuration Properties**, click **Deployment**.
3. On the right side of the **Property Pages** pane, under **Target**, verify that the **Server** name is **localhost**. If you are using a different instance, type the name of the instance. Click **OK**.

Next Task in Lesson

Creating a Data Source (Basic Data Mining Tutorial)

See Also

Build Analysis Services Projects (SSDT)
Create an Analysis Services Project (SSDT)

Creating a Data Source (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

A *data source* is a data connection that is saved and managed in your project and deployed to your Microsoft SQL Server Analysis Services database. The data source contains the names of the server and database where your source data resides, in addition to any other required connection properties.

> **IMPORTANT**
>
> The name of the database is **AdventureWorksDW2012**. If you have not already installed this database, see the Microsoft SQL Sample Databases page.

To create a data source

1. In **Solution Explorer**, right-click the **Data Sources** folder and select **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** to add a connection to the **AdventureWorksDW2012** database.
4. In the **Provider** list in **Connection Manager**, select **Native OLE DB\SQL Server Native Client 11.0**.
5. In the **Server name** box, type or select the name of the server on which you installed **AdventureWorksDW2012**.

   For example, type **localhost** if the database is hosted on the local server.
6. In the **Log onto the server** group, select **Use Windows Authentication**.

   > **IMPORTANT**
   >
   > Whenever possible, implementers should use Windows Authentication, as it provides a more secure authentication method than SQL Server Authentication. However, SQL Server Authentication is provided for backward compatibility. For more information about authentication methods, see Database Engine Configuration - Account Provisioning.
7. In the **Select or enter a database name** list, select **AdventureWorksDW2012** and then click **OK**.
8. Click **Next**.
9. On the **Impersonation Information** page, click **Use the service account**, and then click **Next**.

   On the **Completing the Wizard** page, notice that, by default, the data source is named Adventure Works DW 2012.
10. Click **Finish**.

    The new data source, Adventure Works DW 2012, appears in the **Data Sources** folder in Solution Explorer.

Next Task in Lesson

Creating a Data Source View (Basic Data Mining Tutorial)

Previous Task in Lesson

Creating an Analysis Services Project (Basic Data Mining Tutorial)

See Also

Create a Data Source (SSAS Multidimensional)
Defining a Data Source
Set Impersonation Options (SSAS - Multidimensional)

Creating a Data Source View (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

A data source view is built on a data source and defines a subset of the data, which you can then use in your mining structures. You can also use the data source view to add columns, create calculated columns and aggregates, and add named views. By using data source views, you can select the data that relates to your project, establish relationships between tables, and modify the structure of the data, without modifying the original data source. For more information, see Data Source Views in Multidimensional Models.

To create a data source view

1. In **Solution Explorer**, right-click **Data Source Views**, and select **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, under **Relational data sources**, select the Adventure Works DW 2012 data source that you created in the last task. Click **Next**.

   > **NOTE**
   > If you want to create a data source, right-click **Data Sources** and then click **New Data Source** to start the Data Source Wizard.

4. On the **Select Tables and Views** page, select the following objects, and then click the right arrow to include them in the new data source view:
   - **ProspectiveBuyer (dbo)** - table of prospective bike buyers
   - **vTargetMail (dbo)** - view of historical data about past bike buyers
5. Click **Next**.
6. On the **Completing the Wizard** page, by default the data source view is named Adventure Works DW 2012. Change the name to `Targeted Mailing`, and then click **Finish**.

   The new data source view opens in the **Targeted Mailing.dsv [Design]** tab.

Previous Task in Lesson

Creating a Data Source (Basic Data Mining Tutorial)

Next Lesson

Lesson 2: Building a Targeted Mailing Structure (Basic Data Mining Tutorial)

See Also

Defining a Data Source View (Analysis Services)

Lesson 2: Building a Targeted Mailing Structure (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

The Marketing department of Adventure Works Cycles wants to increase sales by targeting specific customers for a mailing campaign. The company's database contains a list of past customers and a list of potential new customers. By investigating the attributes of previous customers, the company hopes to discover patterns that they can then apply to potential customers. For example, they might use past trends to predict which potential customers are most likely to purchase a bike from Adventure Works Cycles, or create customer segments for future marketing campaigns.

In this lesson you will use the **Data Mining Wizard** to create the targeted mailing structure. After you complete the tasks in this lesson, you will have a mining structure with a single model. Because there are many steps and important concepts involved in creating a structure, we have separated this process into the following three tasks:

Creating a Targeted Mailing Mining Model Structure (Basic Data Mining Tutorial)

Specifying the Data Type and Content Type (Basic Data Mining Tutorial)

Specifying a Testing Data Set for the Structure (Basic Data Mining Tutorial)

First Task in Lesson

Creating a Targeted Mailing Mining Model Structure (Basic Data Mining Tutorial)

Previous Lesson

Lesson 1: Preparing the Analysis Services Database (Basic Data Mining Tutorial)

Next Lesson

Lesson 3: Adding and Processing Models

See Also

Create the Data Mining Structure (Data Mining Wizard)
Create a Relational Mining Structure

Creating a Targeted Mailing Mining Model Structure (Basic Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

The first step in creating a targeted mailing scenario is to use the Data Mining Wizard in SQL Server Data Tools (SSDT) to create a new mining structure and decision tree mining model.

In this task you will set up a new mining structure, and add an initial mining model based on the Microsoft Decision Trees algorithm. To create the structure, you will first select tables and views and then identify which columns will be used for training and which for testing.

To create a mining structure for the targeted mailing scenario

1. In Solution Explorer, right-click **Mining Structures** and select **New Mining Structure** to start the Data Mining Wizard.
2. On the **Welcome to the Data Mining Wizard** page, click **Next**.
3. On the **Select the Definition Method** page, verify that **From existing relational database or data warehouse** is selected, and then click **Next**.
4. On the **Create the Data Mining Structure** page, under **Which data mining technique do you want to use?**, select **Microsoft Decision Trees**.

   > **NOTE**
   > If you get a warning that no data mining algorithms can be found, the project properties might not be configured correctly. This warning occurs when the project attempts to retrieve a list of data mining algorithms from the Analysis Services server and cannot find the server. By default, SQL Server Data Tools will use **localhost** as the server. If you are using a different instance, or a named instance, you must change the project properties. For more information, see Creating an Analysis Services Project (Basic Data Mining Tutorial).

5. Click **Next**.
6. On the **Select Data Source View** page, in the **Available data source views** pane, select **Targeted Mailing**. You can click **Browse** to view the tables in the data source view and then click **Close** to return to the wizard.
7. Click **Next**.
8. On the **Specify Table Types** page, select the check box in the **Case** column for vTargetMail to use it as the case table, and then click **Next**. You will use the ProspectiveBuyer table later for testing; ignore it for now.
9. On the **Specify the Training Data** page, you will identify at least one predictable column, one key column, and one input column for your model. Select the check box in the **Predictable** column in the **BikeBuyer** row.

   > **NOTE**
   > Notice the warning at the bottom of the window. You will not be able to navigate to the next page until you select at least one **Input** and one **Predictable** column.

10. Click **Suggest** to open the **Suggest Related Columns** dialog box.

    The **Suggest** button is enabled whenever at least one predictable attribute has been selected. The **Suggest Related Columns** dialog box lists the columns that are most closely related to the predictable column, and orders the attributes by their correlation with the predictable attribute. Columns with a significant correlation (confidence greater than 95%) are automatically selected to be included in the model.

    Review the suggestions, and then click **Cancel** toignore the suggestions.

    > **NOTE**
    > If you click **OK**, all listed suggestions will be marked as input columns in the wizard. If you agree with only some of the suggestions, you must change the values manually.

11. Verify that the check box in the **Key** column is selected in the **CustomerKey** row.

    > **NOTE**
    > If the source table from the data source view indicates a key, the Data Mining Wizard automatically chooses that column as a key for the model.

12. Select the check boxes in the **Input** column in the following rows. You can check multiple columns by highlighting a range of cells and pressing CTRL while selecting a check box.

    - **Age**
    - **CommuteDistance**
    - **EnglishEducation**
    - **EnglishOccupation**
    - **Gender**
    - **GeographyKey**
    - **HouseOwnerFlag**
    - **MaritalStatus**
    - **NumberCarsOwned**
    - **NumberChildrenAtHome**
    - **Region**
    - **TotalChildren**
    - **YearlyIncome**

13. On the far left column of the page, select the check boxes in the following rows.

    - **AddressLine1**
    - **AddressLine2**
    - **DateFirstPurchase**
    - **EmailAddress**
    - **FirstName**

- **LastName**

Ensure that these rows have checks only in the left column. These columns will be added to your structure but will not be included in the model. However, after the model is built, they will be available for drillthrough and testing. For more information about drillthrough, see Drillthrough Queries (Data Mining)

14. Click **Next**.

Next Task in Lesson

Specifying the Data Type and Content Type (Basic Data Mining Tutorial)

See Also

Specify Table Types (Data Mining Wizard)
Data Mining Designer
Microsoft Decision Trees Algorithm

Specifying the Data Type and Content Type (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Now that you have selected which columns to use for building your structure and training your models, make any necessary changes to the default data and content types that are set by the wizard.

Review and modify content type and data type for each column

1. On the **Specify Columns' Content and Data Type** page, click **Detect** to run an algorithm that determines the default data and content types for each column.

2. Review the entries in the **Content Type** and **Data Type** columns and change them if necessary, to make sure that the settings are the same as those listed in the following table.

   Typically, the wizard will detect numbers and assign an appropriate numeric data type, but there are many scenarios where you might want to handle a number as text instead. For example, the **GeographyKey** should be handled as text, because it would be inappropriate to perform mathematical operations on this identifier.

COLUMN	CONTENT TYPE	DATA TYPE
Address Line1	**Discrete**	**Text**
Address Line2	**Discrete**	**Text**
Age	**Continuous**	**Long**
Bike Buyer	**Discrete**	**Long**
Commute Distance	**Discrete**	**Text**
CustomerKey	**Key**	**Long**
DateLastPurchase	**Continuous**	**Date**
Email Address	**Discrete**	**Text**
English Education	**Discrete**	**Text**
English Occupation	**Discrete**	**Text**
FirstName	**Discrete**	**Text**
Gender	**Discrete**	**Text**
Geography Key	**Discrete**	**Text**
House Owner Flag	**Discrete**	**Text**

COLUMN	CONTENT TYPE	DATA TYPE
Last Name	**Discrete**	**Text**
Marital Status	**Discrete**	**Text**
Number Cars Owned	**Discrete**	**Long**
Number Children At Home	**Discrete**	**Long**
Region	**Discrete**	**Text**
Total Children	**Discrete**	**Long**
Yearly Income	**Continuous**	**Double**

3. Click **Next**.

Next Task in Lesson

Specifying a Testing Data Set for the Structure (Basic Data Mining Tutorial)

Previous Task in Lesson

Creating a Targeted Mailing Mining Model Structure (Basic Data Mining Tutorial)

See Also

Content Types (Data Mining)
Data Types (Data Mining)

Specifying a Testing Data Set for the Structure (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

In the final few screens of the Data Mining Wizard you will split your data into a testing set and a training set. You will then name your structure and enable drillthrough on the model.

Specifying a Testing Set

Separating data into training and testing sets when you create a mining structure makes it possible to easily assess the accuracy of the mining models that you create later. For more information on testing sets, see Training and Testing Data Sets.

To specify the testing set

1. On the **Create Testing Set** page, for **Percentage of data for testing**, leave the default value of `30`.
2. For **Maximum number of cases in testing data set**, type `1000`.
3. Click **Next**.

Specifying Drillthrough

Drillthrough can be enabled on models and on structures. The checkbox in this dialog box enables drillthrough on the named model. After the model has been processed, you will be able to retrieve detailed information from the training data that were used to create the model.

If the underlying mining structure has also been configured to allow drillthrough, you can retrieve detailed information from both the model cases and the mining structure, including columns that were not included in the mining model. For more information, see Drillthrough Queries (Data Mining).

To name the model and structure and specify drillthrough

1. On the **Completing the Wizard** page, in **Mining structure name**, type `Targeted Mailing`.
2. In **Mining model name**, type `TM_Decision_Tree`.
3. Select the **Allow drill through** check box.
4. Review the **Preview** pane. Notice that only those columns selected as **Key**, **Input** or **Predictable** are shown. The other columns you selected (e.g., AddressLine1) are not used for building the model but will be available in the underlying structure, and can be queried after the model is processed and deployed.
5. Click **Finish**.

Previous Task in Lesson

Specifying the Data Type and Content Type (Basic Data Mining Tutorial)

Next Lesson

Lesson 3: Adding and Processing Models

See Also

Enable Drillthrough for a Mining Model
Drillthrough Queries (Data Mining)
Specify the Training Data (Data Mining Wizard)

Lesson 3: Adding and Processing Models

2/11/2019 • 2 minutes to read • Edit Online

The mining structure that you created in the previous lesson contains a single mining model that is based on the Microsoft Decision Trees algorithm. You can use this model to identify customers for the targeted mailing campaign. However, to ensure that your analysis is thorough, it is a common practice to create related models using different algorithms and compare their results. That way you can get different insights as well. Therefore, you will create two additional models, then process and deploy the models.

In this lesson, you will create a set of mining models that will suggest the most likely customers from a list of potential customers.

To complete the tasks in this lesson, you will use the Microsoft Clustering Algorithm and the Microsoft Naive Bayes Algorithm.

This lesson contains the following tasks:

Adding New Models to the Targeted Mailing Structure (Basic Data Mining Tutorial)

Processing Models in the Targeted Mailing Structure (Basic Data Mining Tutorial)

First Task in Lesson

Adding New Models to the Targeted Mailing Structure (Basic Data Mining Tutorial)

Previous Lesson

Lesson 2: Building a Targeted Mailing Structure (Basic Data Mining Tutorial)

Next Lesson

Lesson 4: Exploring the Targeted Mailing Models (Basic Data Mining Tutorial)

See Also

Add Mining Models to a Structure (Analysis Services - Data Mining)

Adding New Models to the Targeted Mailing Structure (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

In this task, you will define two additional models by using the **Mining Models** tab of Data Mining Designer. You will use the Microsoft Clustering and Microsoft Naive Bayes algorithms to create the models. These two algorithms are selected because of their ability to predict a discrete value (i.e., bike purchase). For more information about these algorithms, see Microsoft Clustering Algorithm and Microsoft Naive Bayes Algorithm

To create a clustering mining model

1. Switch to the **Mining Models** tab in Data Mining Designer in SQL Server Data Tools (SSDT).

   Notice that the designer displays two columns, one for the mining structure and one for the `TM_Decision_Tree` mining model, which you created in the previous lesson.

2. Right-click the **Structure** column and select **New Mining Model**.

3. In the **New Mining Model** dialog box, in **Model name**, type `TM_Clustering`.

4. In **Algorithm name**, select **Microsoft Clustering**.

5. Click **OK**.

The new model now appears in the **Mining Models** tab of Data Mining Designer. This model, built with the Microsoft Clustering algorithm, groups customers with similar characteristics into clusters and predicts bike buying for each cluster. Although you can modify the column usage and properties for the new model, no changes to the `TM_Clustering` model are necessary for this tutorial.

To create a Naive Bayes mining model

1. In the **Mining Models** tab of Data Mining Designer, right-clickthe **Structure** column, and select **New Mining Model**.

2. In the **New Mining Model** dialog box, under **Model name**, type `TM_NaiveBayes`.

3. In **Algorithm name**, select **Microsoft Naive Bayes**, then click **OK**.

   A message appears stating that the Microsoft Naive Bayes algorithm does not support the **Age** and **Yearly Income** columns, which are continuous.

4. Click **Yes** to acknowledge the message and continue.

A new model appears in the **Mining Models** tab of Data Mining Designer. Although you can modify the column usage and properties for all the models in this tab, no changes to the `TM_NaiveBayes` model are necessary for this tutorial.

Next Task in Lesson

Processing Models in the Targeted Mailing Structure (Basic Data Mining Tutorial)

See Also

Add Mining Models to a Structure (Analysis Services - Data Mining)
Data Mining Designer

Processing Models in the Targeted Mailing Structure (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Before you can browse or work with the mining models that you have created, you must deploy the Analysis Services project and process the mining structure and mining models.

- *Deploying* sends the project to a server and creates any objects in that project on the server.
- *Processing* populates Analysis Services objects with data from relational data sources.

Models cannot be used until they have been deployed and processed. Also, when you make any changes to the model, such as adding new data, you must redeploy and reprocess the models.

Ensuring Consistency with HoldoutSeed

When you deploy a project and process the structure and models, individual rows in your data structure are assigned to either the training set or testing set based on a numerical seed value. By default, the numerical seed value is computed based on attributes of the data structure. However, if you ever change some aspects of your model, the seed value would change, leading to subtly different results. Therefore, in order to ensure that your results are the same as described here, we will arbitrarily assign a fixed *holdout seed* of `12`. The holdout seed is used to initialize the sampling algorithm, and ensures that the data is partitioned in roughly the same way for all mining structures and their models.

This value does not affect the number of cases in the training set; it simply ensures that the same partitioning method will be used each time you build the model.

For more information on holdout seed, see Training and Testing Data Sets.

To set the Holdout Seed

1. Click on the **Mining Structure** tab or the **Mining Models** tab in Data Mining Designer in SQL Server Data Tools (SSDT).

   Targeted Mailing MiningStructure displays in the **Properties** pane.

2. Ensure that the **Properties** pane is open by pressing **F4**.
3. Ensure that **CacheMode** is set to **KeepTrainingCases**.
4. Enter `12` for **HoldoutSeed**.

Deploying and Processing the Models

In Data Mining Designer, you can decide which objects to process, depending on the scope of changes you've made to your model or the underlying data:

For this task, because the data and the models are new, you will process the structure and all the models at the same time.

To deploy the project and process all the mining models

1. In the **Mining Model** menu, select **Process Mining Structure and All Models**.

   If you made changes to the structure, you will be prompted to build and deploy the project before processing the models. Click **Yes**.

2. Click **Run** in the **Processing Mining Structure - Targeted Mailing** dialog box.

   The **Process Progress** dialog box opens to display the details of model processing. Model processing might take some time, depending on your computer.

3. Click **Close** in the **Process Progress** dialog box after the models have completed processing.

4. Click **Close** in the **Processing Mining Structure - <structure>** dialog box.

Previous Task in Lesson

Adding New Models to the Targeted Mailing Structure (Basic Data Mining Tutorial)

Next Lesson

Lesson 4: Exploring the Targeted Mailing Models (Basic Data Mining Tutorial)

See Also

Processing Requirements and Considerations (Data Mining)

Lesson 4: Exploring the Targeted Mailing Models (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

After the models in your project have been processed, you can explore them to look for interesting trends. Because patterns can be complex and difficult simply by looking at numbers, SQL Server Data Mining provides some visual tools that help you investigate the data and understand the rules and relationships that the algorithms have discovered within the data. You can also use a variety of accuracy tests to validate your dataset or discover which model performs best before you deploy it.

When you use SQL Server Data Tools (SSDT) to explore your models, each model you created is listed in the **Mining Model Viewer** tab in Data Mining Designer. You can use the viewers to explore the models. These viewers are also available in SQL Server Management Studio.

Each algorithm that you used to build a model in Analysis Services returns a different type of result. Therefore, Analysis Services provides custom viewers for each type of machine learning model.

If you want to get into details, Analysis Services also provides an HTML viewer, called the **Generic Content Tree Viewer**, that displays detailed information about the model data and any patterns that were found, in a semi-tabular format. For more information, see Browse a Model Using the Microsoft Generic Content Tree Viewer.

In this lesson you will look at the results from your three models. Each model type is based on a different algorithm and provides different insights into the data.

- The Decision Tree model tells you about factors that influence bike buying.
- The Clustering model groups your customers by attributes that include their bike buying behavior and other selected attributes.
- The Naive Bayes model enables you to explore the relationship between different attributes.

See the following topics to learn more about each of the mining model viewers.

- Exploring the Decision Tree Model (Basic Data Mining Tutorial)
- Exploring the Clustering Model (Basic Data Mining Tutorial)
- Exploring the Naive Bayes Model (Basic Data Mining Tutorial)

All three models can be viewed using the **Generic Content Tree Viewer**, to extract formulas, data values, and so forth.

First Task in Lesson

Exploring the Decision Tree Model (Basic Data Mining Tutorial)

Previous Lesson

Lesson 3: Adding and Processing Models

Next Lesson

Lesson 5: Testing Models (Basic Data Mining Tutorial)

See Also

Mining Model Viewer Tasks and How-tos
Data Mining Model Viewers

Exploring the Decision Tree Model (Basic Data Mining Tutorial)

2/11/2019 • 4 minutes to read • Edit Online

The Microsoft Decision Trees algorithm predicts which columns influence the decision to purchase a bike based upon the remaining columns in the training set.

Decision Tree Tab

On the **Decision Tree** tab, you can view decision trees for every predictable attribute in the dataset.

In this case, the model predicts only one column, Bike Buyer, so there is only one tree to view. If there were more trees, you could use the **Tree** box to choose another tree.

As you view the `TM_Decision_Tree` model in the Decision Tree viewer, you can see the most important attributes at the left side of the chart. "Most important" means that these attributes have the greatest influence on the outcome. Attributes further down the tree (to the right of the chart) have less of an effect.

In this example, age is the single most important factor in predicting bike buying. The model groups customers by age, and then shows the next more important attribute for each age group. For example, in the group of customers aged 34 to 40, the number of cars owned is the strongest predictor after age.

To explore the model in the Decision Tree tab

1. Select the **Mining Model Viewer** tab in **Data Mining Designer**.

   By default, the designer opens to the first model that was added to the structure -- in this case, `TM_Decision_Tree`.

2. Use the magnifying glass buttons to adjust the size of the tree display.

   By default, the Microsoft Tree Viewer shows only the first three levels of the tree. If the tree contains fewer than three levels, the viewer shows only the existing levels. You can view more levels by using the **Show Level** slider or the **Default Expansion** list.

3. Slide **Show Level** to the fourth bar.

4. Change the **Background** value to `1`.

   By changing the **Background** setting, you can quickly see the number of cases in each node that have the target value of `1` for [Bike Buyer]. Remember that in this particular scenario, each case represents a customer. The value `1` indicates that the customer previously purchased a bike; the value **0** indicates that the customer has not purchased a bike. The darker the shading of the node, the higher the percentage of cases in the node that have the target value.

5. Place your cursor over the node labeled **All**. An tooltip will display the following information:

   - Total number of cases
   - Number of non bike buyer cases
   - Number of bike buyer cases
   - Number of cases with missing values for [Bike Buyer]

   Alternately, place your cursor over any node in the tree to see the condition that is required to reach that

node from the node that comes before it. You can also view this same information in the **Mining Legend**.

6. Click on the node for **Age >=34 and < 41**. The histogram is displayed as a thin horizontal bar across the node and represents the distribution of customers in this age range who previously did (pink) and did not (blue) purchase a bike. The Viewer shows us that customers between the ages of 34 and 40 with one or no cars are likely to purchase a bike. Taking it one step further, we find that the likelihood to purchase a bike increases if the customer is actually age 38 to 40.

Because you enabled drillthrough when you created the structure and model, you can retrieve detailed information from the model cases and mining structure, including those columns that were not included in the mining model (e.g., emailAddress, FirstName).

For more information, see Drillthrough Queries (Data Mining).

To drill through to case data

1. Right-click a node, and select **Drill Through** then **Model Columns Only**.

   The details for each training case are displayed in spreadsheet format. These details come from the vTargetMail view that you selected as the case table when building the mining structure.

2. Right-click a node, and select **Drill Through** then **Model and Structure Columns**.

   The same spreadsheet displays with the structure columns appended to the end.

Dependency Network Tab

The **Dependency Network** tab displays the relationships between the attributes that contribute to the predictive ability of the mining model. The Dependency Network viewer reinforces our findings that Age and Region are important factors in predicting bike buying.

To explore the model in the Dependency Network tab

1. Click the `Bike Buyer` node to identify its dependencies.

   The center node for the dependency network, `Bike Buyer`, represents the predictable attribute in the mining model. The graph highlights any connected nodes that have an effect on the predictable attribute.

2. Adjust the **All Links** slider to identify the most influential attribute.

   As you drag down the slider, attributes that have only a weak effect on the [Bike Buyer] column are removed from the graph. By adjusting the slider, you can discover that Age and Region are the greatest factors in predicting whether someone is a bike buyer.

Related Tasks

See these topics to explore the data using the other kinds of models.

- Exploring the Clustering Model (Basic Data Mining Tutorial)
- Exploring the Naive Bayes Model (Basic Data Mining Tutorial)

Next Task in Lesson

Exploring the Clustering Model (Basic Data Mining Tutorial)

See Also

Mining Model Viewer Tasks and How-tos
Decision Tree Tab (Mining Model Viewer)
Dependency Network Tab (Mining Model Viewer)
Browse a Model Using the Microsoft Tree Viewer

Exploring the Clustering Model (Basic Data Mining Tutorial)

2/11/2019 • 5 minutes to read • Edit Online

The Microsoft Clustering algorithm groups cases into clusters that contain similar characteristics. These groupings are useful for exploring data, identifying anomalies in the data, and creating predictions.

The Microsoft Cluster Viewer provides the following tabs for use in exploring clustering mining models:

Cluster Diagram Tab

The Cluster Diagram tab displays all the clusters that are in a mining model. The lines between the clusters represent "closeness" and are shaded based on how similar the clusters are. The actual color of each cluster represents the frequency of the variable and the state in the cluster.

To explore the model in the Cluster Diagram tab

1. Use the **Mining Model** list at the top of the **Mining Model Viewer** tab to switch to the `TM_Clustering` model.

2. In the **Viewer** list, select **Microsoft Cluster Viewer**.

3. In the **Shading Variable** box, select **Bike Buyer**.

   The default variable is **Population**, but you can change this to any attribute in the model, to discover which clusters contain members that have the attributes you want.

4. Select **1** in the **State** box to explore those cases where a bike was purchased.

   The **Density** legend describes the density of the attribute state pair selected in the Shading Variable and the State. In this example it tells us that the clusterwith the darkest shading has the highest percentage of bike buyers.

5. Pause your mouse over the cluster with the darkest shading.

   A tooltip displays the percentage of cases that have the attribute, `Bike Buyer = 1`.

6. Select the cluster that has the highest density, right-click the cluster, select **Rename Cluster** and type **Bike Buyers High** for later identification. Click **OK**.

7. Find the cluster that has the lightest shading (and the lowest density). Right-click the cluster, select **Rename Cluster** and type **Bike Buyers Low**. Click **OK**.

8. Click the **Bike Buyers High** cluster and drag it to an area of the pane that will give you a clear view of its connections to the other clusters.

   When you select a cluster, the lines that connect this cluster to other clusters are highlighted, so that you can easily see all the relationships for this cluster. When the cluster is not selected, you can tell by the darkness of the lines how strong the relationships are amongst all the clusters in the diagram. If the shading is light or nonexistent, the clusters are not very similar.

9. Use the slider to the left of the network, to filter out the weaker links and find the clusters with the closest relationships. The Adventure Works Cycles marketing department might want to combine similar clusters together when determining the best method for delivering the targeted mailing.

Cluster Profiles Tab

The **Cluster Profiles** tab provides an overall view of the `TM_Clustering` model. The **Cluster Profiles** tab contains a column for each cluster in the model. The first column lists the attributes that are associated with at least one cluster. The rest of the viewer contains the distribution of the states of an attribute for each cluster. The distribution of a discrete variable is shown as a colored bar with the maximum number of bars displayed in the **Histogram bars** list. Continuous attributes are displayed with a diamond chart, which represents the mean and standard deviation in each cluster.

To explore the model in the Cluster Profiles tab

1. Set **Histogram** bars to **5**.

   In our model, 5 is the maximum number of states for any one variable.

2. If the **Mining Legend** blocks the display of the **Attribute profiles**, move it out of the way.

3. Select the **Bike Buyers High** column and drag it to the right of the **Population** column.

4. Select the **Bike Buyers Low** column and drag it to the right of the **Bike Buyers High** column.

5. Click the **Bike Buyers High** column.

   The **Variables** column is sorted in order of importance for that cluster. Scroll through the column and review characteristics of the Bike Buyer High cluster. For example, they are more likely to have a short commute.

6. Double-click the **Age** cell in the **Bike Buyers High** column.

   The **Mining Legend** displays a more detailed view and you can see the age range of these customers as well as the mean age.

7. Right-click the **Bike Buyers Low** column and select **Hide Column**.

Cluster Characteristics Tab

With the **Cluster Characteristics** tab, you can examine in more detail the characteristics that make up a cluster. Instead of comparing the characteristics of all of the clusters (as in the Cluster Profiles tab), you can explore one cluster at a time. For example, if you select **Bike Buyers High** from the **Cluster** list, you can see the characteristics of the customers in this cluster. Though the display is different from the Cluster Profiles viewer, the findings are the same.

> **NOTE**
>
> Unless you set an initial value for **holdoutseed**, results will vary each time you process the model. For more information, see HoldoutSeed Element

Cluster Discrimination Tab

With the **Cluster Discrimination** tab, you can explore the characteristics that distinguish one cluster from another. After you select two clusters, one from the **Cluster 1** list, and one from the **Cluster 2** list, the viewer calculates the differences between the clusters and displays a list of the attributes that distinguish the clusters most.

To explore the model in the Cluster Discrimination tab

1. In the **Cluster 1** box, select **Bike Buyers High**.

2. In the **Cluster 2** box, select **Bike Buyers Low**.

3. Click **Variables** to sort alphabetically.

   Some of the more substantial differences among the customers in the **Bike Buyers Low** and **Bike Buyers High** clusters include age, car ownership, number of children, and region.

Related Tasks

See the following topics to explore the other mining models.

- Exploring the Decision Tree Model (Basic Data Mining Tutorial)
- Exploring the Naive Bayes Model (Basic Data Mining Tutorial)

Next Task in Lesson

Exploring the Naive Bayes Model (Basic Data Mining Tutorial)

Previous Task in Lesson

Exploring the Decision Tree Model (Basic Data Mining Tutorial)

See Also

Browse a Model Using the Microsoft Cluster Viewer
Cluster Discrimination Tab (Mining Model Viewer)
Cluster Profiles Tab (Mining Model Viewer)
Cluster Characteristics Tab (Mining Model Viewer)
Cluster Diagram Tab (Mining Model Viewer)

Exploring the Naive Bayes Model (Basic Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

The Microsoft Naive Bayes algorithm provides several methods for displaying the interaction between bike buying and the input attributes.

The Microsoft Naive Bayes Viewer provides the following tabs for use in exploring Naive Bayes mining models:

Dependency Network

The **Dependency Network** tab works in the same way as the **Dependency Network** tab for the Microsoft Tree Viewer. Each node in the viewer represents an attribute, and the lines between nodes represent relationships. In the viewer, you can see all the attributes that affect the state of the predictable attribute, Bike Buyer.

To explore the model in the Dependency Network tab

1. Use the **Mining Model** list at the top of the **Mining Model Viewer** tab to switch to the `TM_NaiveBayes` model.
2. Use the **Viewer** list to switch to **Microsoft Naive Bayes Viewer**.
3. Click the `Bike Buyer` node to identify its dependencies.

   The pink shading indicates that all of the attributes have an effect on bike buying.
4. Adjust the slider to identify the most influential attribute.

   As you lower the slider, only the attributes that have the greatest effect on the [Bike Buyer] column remain. By adjusting the slider, you can discover that a few of the most influential attributes are: number of cars owned, commute distance, and total number of children.

Attribute Profiles

The **Attribute Profiles** tab describes how different states of the input attributes affect the outcome of the predictable attribute.

To explore the model in the Attribute Profiles tab

1. In the **Predictable** box, verify that `Bike Buyer` is selected.
2. If the **Mining Legend** is blocking display of the **Attribute profiles**, move it out of the way.
3. In the **Histogram** bars box, select **5**.

   In our model, 5 is the maximum number of states for any one variable.

   The attributes that affect the state of this predictable attribute are listed together with the values of each state of the input attributes and their distributions in each state of the predictable attribute.
4. In the **Attributes** column, find **Number Cars Owned**. Notice the differences in the histograms for bike buyers (column labeled 1) and non-buyers (column labeled 0). A person with zero or one car is much more likely to buy a bike.
5. Double-click the **Number Cars Owned** cell in the bike buyer (column labeled 1) column.

   The **Mining Legend** displays a more detailed view.

Attribute Characteristics

With the **Attribute Characteristics** tab, you can select an attribute and value to see how frequently values for other attributes appear in the selected value cases.

To explore the model in the Attribute Characteristics tab

1. In the **Attribute** list, verify that `Bike Buyer` is selected.
2. Set the **Value** to **1**.

   In the viewer, you will see that customers who have no children at home, short commutes, and live in the North America region are more likely to buy a bike.

Attribute Discrimination

With the **Attribute Discrimination** tab, you can investigate the relationship between two discrete values of bike buying and other attribute values. Because the `TM_NaiveBayes` model has only two states, 1 and 0, you do not have to make any changes to the viewer.

In the viewer, you can see that people who do not own cars tend to buy bicycles, and people who own two cars tend not to buy bicycles.

Related Tasks

See the following topics to explore the other mining models.

- Exploring the Decision Tree Model (Basic Data Mining Tutorial)
- Exploring the Clustering Model (Basic Data Mining Tutorial)

Next Lesson

Lesson 5: Testing Models (Basic Data Mining Tutorial)

Previous Task in Lesson

Exploring the Clustering Model (Basic Data Mining Tutorial)

See Also

Browse a Model Using the Microsoft Naive Bayes Viewer
Attribute Discrimination Tab (Mining Model Viewer)
Attribute Profiles Tab (Mining Model Viewer)
Attribute Characteristics Tab (Mining Model Viewer)
Dependency Network Tab (Mining Model Viewer)

Lesson 5: Testing Models (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Now that you have processed the model by using the targeted mailing scenario training set, you will test your models against the testing set. Validation is an important step in the data mining process. Knowing how well your targeted mailing mining models perform against real data is important before you deploy the models into a production environment.

Because the data in the testing set already contains known values for bike buying, it is easy to determine whether the model's predictions are correct. The model that performs the best will be used by the Adventure Works Cycles marketing department to identify the customers for their targeted mailing campaign.

In this lesson you will validate your models using multiple methods:

1. You'll make predictions against the testing set to see how accurate the model is on known results. You'll use a *lift chart* to measure its effectiveness.

   Testing Accuracy with Lift Charts (Basic Data Mining Tutorial)

2. You will test your models on a filtered subset of the data. You can compare multiple models in the same lift chart.

   Testing a Filtered Model (Basic Data Mining Tutorial)

For more information about how model validation in general, see Data Mining Concepts.

First Task in Lesson

Testing Accuracy with Lift Charts (Basic Data Mining Tutorial)

Previous Lesson

Lesson 4: Exploring the Targeted Mailing Models (Basic Data Mining Tutorial)

Next Lesson

Lesson 6: Creating and Working with Predictions (Basic Data Mining Tutorial)

See Also

Lift Chart Tab (Mining Accuracy Chart View)
Lift Chart (Analysis Services - Data Mining)
Testing and Validation (Data Mining)
Classification Matrix Tab (Mining Accuracy Chart View)
Classification Matrix (Analysis Services - Data Mining)

Testing Accuracy with Lift Charts (Basic Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

On the **Mining Accuracy Chart** tab of Data Mining Designer, you can calculate how well each of your models makes predictions, and compare the results of each model directly against the results of the other models. This method of comparison is referred to as a *lift chart*. Typically, the predictive accuracy of a mining model is measured by either lift or classification accuracy. For this tutorial we will use the lift chart only.

In this topic, you will perform the following tasks:

- Choose the Input Data
- Configure Accuracy Chart Parameters

Choosing the Input Data

The first step in testing the accuracy of your mining models is to select the data source that you will use for testing. You will test how well the models perform against your testing data and then you will use them with external data.

To select the data set

1. Switch to the **Mining Accuracy Chart** tab in Data Mining Designer in SQL Server Data Tools (SSDT) and select the **Input Selection** tab.
2. In the **Select data set to be used for Accuracy Chart** group box, select **Use mining structure test cases**. This is the testing data that you set aside when you created the mining structure.

   For more information on the other options, see Choose an Accuracy Chart Type and Set Chart Options.

Setting Accuracy Chart Parameters

To create an accuracy chart, you must define three things:

- Which models should you include in the accuracy chart?
- Which predictable attribute do you want to measure? Some models might have multiple targets, but each chart can measure only one outcome at a time.

  To use a column as the **Predictable Column Name** in an accuracy chart, the columns must have the usage type of `Predict` or `Predict Only`. Also, the content type of the target column must be either `Discrete` or `Discretized`. In other words, you cannot measure accuracy against continuous numeric outputs using the lift chart.
- Do you want to measure the model's general accuracy, or its accuracy in predicting a particular value (such as [Bike Buyer] = 'Yes')

To generate the lift chart

1. On the **Input Selection** tab of Data Mining Designer, under **Select predictable mining model columns to show in the lift chart**, select the checkbox for **Synchronize Prediction Columns and Values**.
2. In the **Predictable Column Name** column, verify that **Bike Buyer** is selected for each model.
3. In the **Show** column, select each of the models.

By default, all the models in the mining structure are selected. You can decide not to include a model, but for this tutorial leave all the models selected.

4. In the **Predict Value** column, select **1**. The same value is automatically filled in for each model that has the same predictable column.

5. Select the **Lift Chart** tab.

   When you click the tab, a prediction query is executed to get predictions for the test data, and the results are compared against the known values. The results are plotted on the graph.

   If you specified a particular target outcome using the **Predict Value** option, the lift chart plots the results of random guesses and the results of an ideal model.

   - The random guess line shows how accurate the model would be without using any data to inform its predictions: that is, a 50-50 split between two outcomes. The lift chart helps you visualize how much better your model performs in comparison to a random guess.
   - The ideal model line represents the upper bound of accuracy. It shows you the maximum possible benefit you could achieve if your model always predicted accurately.

   The mining models you created will usually fall between these two extremes. Any improvement from the random guess is considered to be *lift*.

6. Use the legend to locate the colored lines representing the Ideal Model and the Random Guess Model.

   You'll notice that the `TM_Decision_Tree` model provides the greatest lift, outperforming both the Clustering and Naive Bayes models.

For an in-depth explanation of a lift chart similar to the one created in this lesson, see Lift Chart (Analysis Services - Data Mining).

Next Task in Lesson

Testing a Filtered Model (Basic Data Mining Tutorial)

See Also

Lift Chart (Analysis Services - Data Mining)
Lift Chart Tab (Mining Accuracy Chart View)

Testing a Filtered Model (Basic Data Mining Tutorial)

2/11/2019 • 4 minutes to read • Edit Online

Now that you have determined that the `TM_Decision_Tree` model is the most accurate, you will customize the model to better suit the needs of the Adventure Works Cycles targeted mailing campaign. Specifically, the Marketing department would like to know if there are any differences between male and female customers. The information could help them decide which magazines to use for advertising and which products to feature in their mailings.

Using Filters

Filtering enables you to easily create models built on subsets of your data. The filter is applied only to the model and does not change the underlying data source.

In this lesson, you will create a model that is filtered on gender, to predict the characteristics that most influence buying behavior in males and female.

First you will make a copy of the `TM_Decision_Tree` model.

To copy the Decision Tree Model

1. In SQL Server Data Tools (SSDT), in Solution Explorer, select **BasicDataMining**.
2. Click the **Mining Models** tab.
3. Right click the `TM_Decision_Tree` model, and select **New Mining Model.**
4. In the **Model name** field, type `TM_Decision_Tree_Male`.
5. Click **OK**.

Next, create a filter to select customers for the model based on their gender.

To create a case filter on a mining model

1. Right-click the `TM_Decision_Tree_Male` mining model to open the shortcut menu.

   -- or --

   Select the model. On the **Mining Model** menu, select **Set Model Filter**.
2. In the **Model Filter** dialog box, click the top row in the grid, in the **Mining Structure Column** text box.

   The drop-down list displays only the names of the columns in that table.
3. In the Mining Structure Column text box, select **Gender**.

   The icon at the left side of the text box changes to indicate that the selected item is a table or a column.
4. Click the **Operator** text box and select the equal (=) operator from the list.
5. Click the **Value** text box, and type **M**.
6. Click the next row in the grid.
7. Click **OK** to close the **Model Filter** dialog box.

   The filter displays in the **Properties** window. Alternately, you can launch the **Model Filter** dialog from the **Properties** window.

8. Repeat the above steps, but this time name the model `TM_Decision_Tree_Female` and type **F** in the **Value** text box.

Process the Filtered Models

Models cannot be used until they have been deployed and processed. For more information on processing models, see Processing Models in the Targeted Mailing Structure (Basic Data Mining Tutorial).

To process the filtered model

1. Right-click the `TM_Decision_Tree_Male` model and select **Process Mining Structure and all Model**s
2. Click **Run** to process the new models.
3. After processing is complete, click **Close** on both processing windows.

   You now have two new models displayed in the **Mining Models** tab.

Evaluate the Results

View the results and assess the accuracy of the filtered models in much the same way as you did for the previous three models. For more information, see:

Exploring the Decision Tree Model (Basic Data Mining Tutorial)

Testing Accuracy with Lift Charts (Basic Data Mining Tutorial)

To explore the filtered models

1. Select the **Mining Model Viewer** tab in **Data Mining Designer**.
2. In the Mining Model box, select `TM_Decision_Tree_Male`.
3. Slide **Show Level** to `3`.
4. Change the **Background** value to `1`.
5. Place your cursor over the node labeled **All** to see the number of bike buyers versus non-bike buyers.
6. Repeat steps 1 - 5 for `TM_Decision_Tree_Female`.
7. Explore the results for the `TM_Decision_Tree` and the models filtered for gender. Compared to all bike buyers, male and female bike buyers share some of the same characteristics as the unfiltered bike buyers but all three have interesting differences as well. This is useful information that Adventure Works Cycles can use to develop their marketing campaign.

To test the lift of the filtered models

1. Switch to the **Mining Accuracy Chart** tab in Data Mining Designer in SQL Server Data Tools (SSDT) and select the **Input Selection** tab.
2. In the **Select data set to be used for Accuracy Chart** group box, select **Use mining structure test cases**.
3. On the **Input Selection** tab of Data Mining Designer, under **Select predictable mining model columns to show in the lift chart**, select the checkbox for **Synchronize Prediction Columns and Values**.
4. In the **Predictable Column Name** column, verify that **Bike Buyer** is selected for each model.
5. In the **Show** column, select each of the models.
6. In the **Predict Value** column, select `1`.
7. Select the **Lift Chart** tab to display the lift chart.

You will now notice that all three Decision Tree models provide significant lift compared to the random guess model, and also outperform the Clustering and Naive-Bayes models.

Related Tasks

For more information on filters, see Filters for Mining Models (Analysis Services - Data Mining).

For an example of how to apply filters to nested tables, see Intermediate Data Mining Tutorial (Analysis Services - Data Mining).

Previous Task in Lesson

Testing Accuracy with Lift Charts (Basic Data Mining Tutorial)

Next Lesson

Lesson 6: Creating and Working with Predictions (Basic Data Mining Tutorial)

See Also

Intermediate Data Mining Tutorial (Analysis Services - Data Mining)
Mining Model Tasks and How-tos
Delete a Filter from a Mining Model
Filters for Mining Models (Analysis Services - Data Mining)

Lesson 6: Creating and Working with Predictions (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

You have trained, tested, and explored the data mining models you created. Now you are ready to use the models to identify the people most likely to respond to the new targeted mailing campaign.

In this lesson you will create a query to predict which customers are most likely to purchase a bike. You will also retrieve the *probability* that the prediction is correct, so the marketing department can decide whether to you can decide whether to use the prediction or not.

Once you have identified customers with a high probability of purchasing a bike, you will drill through to the details of the cases in the mining model to retrieve names and contact information for these customers.

This lesson contains the following topics:

Creating Predictions (Basic Data Mining Tutorial)

Using Drillthrough on Structure Data (Basic Data Mining Tutorial)

Next Lesson

Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

Previous Lesson

Lesson 5: Testing Models (Basic Data Mining Tutorial)

Next Task in Lesson

Creating Predictions (Basic Data Mining Tutorial)

See Also

Mining Model Content for Decision Tree Models (Analysis Services - Data Mining)
Create a Prediction Query Using the Prediction Query Builder

Creating Predictions (Basic Data Mining Tutorial)

2/11/2019 • 6 minutes to read • Edit Online

After you have tested the accuracy of your mining models and decided that you are satisfied with the results, you can then generate predictions by using the Prediction Query Builder on the **Mining Model Prediction** tab in the Data Mining Designer.

The Prediction Query Builder has three views. With the **Design** and **Query** views, you can build and examine your query. You can then run the query and view the results in the **Result** view.

All prediction queries use DMX, which is short for the Data Mining Extensions (DMX) language. DMX has syntax like that of T-SQL but is used for queries against data mining objects. Although DMX syntax is not complicated, using a query builder like this one, or the one in the SQL Server Data Mining Add-Ins for Office, makes it much easier to select inputs and build expressions, so we highly recommend that you learn the basics.

Creating the Query

The first step in creating a prediction query is to select a mining model and input table.

To select a model and input table

1. On the **Mining Model Prediction** tab of Data Mining Designer, in the **Mining Model** box, click **Select Model**.
2. In the **Select Mining Model** dialog box, navigate through the tree to the **Targeted Mailing** structure, expand the structure, select `TM_Decision_Tree`, and then click **OK**.
3. In the **Select Input Table(s)** box, click **Select Case Table**.
4. In the **Select Table** dialog box, in the **Data Source** list, select the data source view Adventure Works DW Multidimensional 2012.
5. In **Table/View Name**, select the **ProspectiveBuyer (dbo)** table, and then click **OK**.

   The `ProspectiveBuyer` table most closely resembles the **vTargetMail** case table.

Mapping the Columns

After you select the input table, Prediction Query Builder creates a default mapping between the mining model and the input table, based on the names of the columns. At least one column from the structure must match a column in the external data.

> **IMPORTANT**
>
> The data that you use to determine the accuracy of the models must contain a column that can be mapped to the predictable column. If such a column does not exist, you can create one with empty values, but it must have the same data type as the predictable column.

To map the inputs to the model

1. Right-click the lines connecting the **Mining Model** window to the **Select Input Table** window, and select **Modify Connections**.

   Notice that not every column is mapped. We will add mappings for several **Table Columns**. We will also generate a new birth date column based on the current date column, so that the columns match better.

2. Under **Table Column**, click the `Bike Buyer` cell and select ProspectiveBuyer.Unknown from the dropdown.

   This maps the predictable column, [Bike Buyer], to an input table column.

3. Click **OK**.

4. In **Solution Explorer**, right-click the **Targeted Mailing** data source view and select **View Designer**.

5. Right-click the table, ProspectiveBuyer, and select **New Named Calculation**.

6. In the **Create Named Calculation** dialog box, for **Column name**, type `calcAge`.

7. For **Description**, type **Calculate age based on birthdate**.

8. In the **Expression** box, type `DATEDIFF(YYYY,[BirthDate],getdate())` and then click **OK**.

   Because the input table has no **Age** column corresponding to the one in the model, you can use this expression to calculate customer age from the BirthDate column in the input table. Since **Age** was identified as the most influential column for predicting bike buying, it must exist in both the model and in the input table.

9. In Data Mining Designer, select the **Mining Model Prediction** tab and re-open the **Modify Connections** window.

10. Under **Table Column**, click the **Age** cell and select ProspectiveBuyer.calcAge from the dropdown.

    > **WARNING**
    > If you do not see the column in the list, you might have to refresh the definition of the data source view that is loaded in the designer. To do this, from the **File** menu, select **Save all**, and then close and re-open the project in the designer.

11. Click **OK**.

Designing the Prediction Query

1. The first button on the toolbar of the **Mining Model Prediction** tab is the **Switch to design view / Switch to result view / Switch to query view** button. Click the down arrow on this button, and select **Design**.

2. In the grid on the **Mining Model Prediction** tab, click the cell in the first empty row in the **Source** column, and then select **Prediction Function**.

3. In the **Prediction Function** row, in the **Field** column, select `PredictProbability`.

   In the **Alias** column of the same row, type **Probability of result**.

4. From the **Mining Model** window above, select and drag [Bike Buyer] into the **Criteria/Argument** cell.

   When you let go, [TM_Decision_Tree].[Bike Buyer] appears in the **Criteria/Argument** cell.

   This specifies the target column for the `PredictProbability` function. For more information about functions, see Data Mining Extensions (DMX) Function Reference.

5. Click the next empty row in the **Source** column, and then select TM_Decision_Tree mining model**.**

6. In the `TM_Decision_Tree` row, in the **Field** column, select `Bike Buyer`.

7. In the `TM_Decision_Tree` row, in the **Criteria/Argument** column, type `=1`.

8. Click the next empty row in the **Source** column, and then select **ProspectiveBuyer table**.

9. In the `ProspectiveBuyer` row, in the **Field** column, select **ProspectiveBuyerKey**.

   This adds the unique identifier to the prediction query so that you can identify who is and who is not likely to buy a bicycle.

10. Add five more rows to the grid. For each row, select **ProspectiveBuyer table** as the **Source** and then add the following columns in the **Field** cells:

    - calcAge
    - LastName
    - FirstName
    - AddressLine1
    - AddressLine2

Finally, run the query and browse the results.

The **Prediction Query Builder** also includes these controls:

- **Show** check box

  Lets you remove clauses from the query without having to delete them from the designer. This can be useful when you are working with complex queries and want to preserve syntax without having to copy and paste DMX into the window.

- **Group**

  Inserts an opening (left) parentheses at the beginning of the selected line, or inserts a closing (right) parenthesis at the end of the current line.

- **AND/OR**

  Inserts the `AND` operator or the `OR` operator immediately after the current function or column.

To run the query and view results

1. In the **Mining Model Prediction** tab, select the **Result** button.

2. After the query runs and the results are displayed, you can review the results.

   The **Mining Model Prediction** tab displays contact information for potential customers who are likely to be bike buyers. The **Probability of result** column indicates the probability of the prediction being correct. You can use these results to determine which potential customers to target for the mailing.

3. At this point, you can save the results. You have three options:

   - Right-click a row of data in the results, and select **Copy** to save just that value (and the column heading) to the Clipboard.
   - Right-click any row in the results, and select **Copy All** to copy the entire result set, including column headings, to the Clipboard.
   - Click **Save query result** to save the results directly to a database as follows:

     a. In the **Save Data Mining Query Result** dialog box, select a data source, or define a new data source.

     b. Type a name for the table that will contain the query results.

     c. Use the option, **Add to DSV**, to create the table and add it to an existing data source view. This is useful if you want to keep all related tables for a model-such as training data,

prediction source data, and query results-in the same data source view.

d. Use the option, **Overwrite if exists**, to update an existing table with the latest results.

   You must use the option to overwrite the table if you have added any columns to the prediction query, changed the names or data types of any columns in the prediction query, or if you have run any ALTER statements on the destination table.

   Also, if multiple columns have the same name (for example, the default column name **Expression**) you must create an alias for the columns with duplicate names, or an error will be raised when the designer tries to save the results to SQL Server. The reason is that SQL Server does not allow multiple columns to have the same name.

   For more information, see Save Data Mining Query Result Dialog Box (Mining Model Prediction View).

Next Task in Lesson

Using Drillthrough on Structure Data (Basic Data Mining Tutorial)

See Also

Create a Prediction Query Using the Prediction Query Builder

Using Drillthrough on Structure Data (Basic Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

As part of their advertising campaign, Adventure Works Cycles is sending a mailer to potential customers in the 34-40 age demographic. The marketing department has decided that they would also like to send the mailer to the customers who purchased bikes from Adventure Works Cycles more than five years ago. In this lesson you will identify customers with older bikes and retrieve their contact information. This information is not included in the model, but is included in the structure. To retrieve the contact information you will first ensure that drillthrough is enabled for the structure and then you will use drillthrough to reveal the names and addresses of the targeted customers.

To enable drillthrough on a mining model

1. In SQL Server Data Tools (SSDT), on the **Mining Models** tab of Data Mining Designer, right-click the **TM_Decision_Tree** model, and select **Properties**.
2. In the Properties windows, click **AllowDrillthrough**, and select **True**.
3. In the Mining Models tab, right-click the model, and select **Process Model**.

For more information, see Drillthrough Queries (Data Mining)

To view drillthrough data from a mining model

1. In Data Mining Designer, click the **Mining Model Viewer** tab.
2. Select the **TM_Decision_Tree** model from the **Mining Model** list.
3. Change the **Background** value to `1`. By doing this, you show only the part of the model that is related to customer who bought bikes.
4. Select the Microsoft Tree viewer from the **Viewer** list. This will force the viewer to refresh with the new filter conditions. Then, locate the **Age >=34 and <41** node and right-click the node.
5. Select **Drill Through**, and then select **Model and Structure Columns** to open the **Drill Through** window.
6. Scroll to the **Structure.Date First Purchase** column to view the purchase dates for the older bikes.
7. To copy the data to the Clipboard, right-click any row in the table, and select **Copy All**.

Congratulations, you have completed the basic data mining tutorial. Now that you are comfortable using the data mining tools, we recommend that you also complete the intermediate data mining tutorial, which demonstrates how to create models for forecasting, market basket analysis, and sequence clustering.

Previous Task in Lesson

Creating Predictions (Basic Data Mining Tutorial)

See Also

Create a Prediction Query Using the Prediction Query Builder

Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

2/11/2019 • 3 minutes to read • Edit Online

Microsoft Analysis Services provides an integrated environment for creating and working with data mining models. You can easily bind to data sources, create and test multiple models on the same data, and deploy models for use in predictive analysis.

In the Basic Data Mining Tutorial, you learned how to use SQL Server Data Tools (SSDT) to create a data mining solution, and you built three models to support a targeted mailing campaign for analyzing customer purchasing behavior and for targeting potential buyers.

This intermediate tutorial builds on that experience and introduces several new scenarios, including common business requirements such as forecasting and market basket analysis. You will learn how to create a time series model, an association model, and a sequence clustering model. Finally, you will learn how to use neural network to explore correlations in data and to use logistic regression for predictions.

The lessons are independent and can be completed separately.

To complete the following tutorials, you should to be familiar with the data mining tools and with the mining model viewers that were introduced in the Basic Data Mining Tutorial.

All scenarios use the **AdventureWorksDW2012** data source, but you will create different data source views for different scenarios. You can do the lessons in any order as long as you create the data source first.

Lesson Scenarios

After your success with the targeted mailing campaign, you have been asked to apply your knowledge of data mining to develop several new models for use in business planning. These include the following tasks:

- **Forecasting:** You will create a *time series* model, to forecast the sales of products in different regions around the world. You will develop individual models for each region and learn how to use *cross-prediction*.
- **Market basket analysis:** You will create an *association model*, to analyze groupings of products that are purchased during visits to the Adventure Works Cycles e-commerce site. Based on this market basket model, you can recommend products to customers.
- **Sequence analysis:** You build a *sequence clustering model*, to analyze the order in which customers buy products. Based on this model, you can plan changes in Web site design or new product offerings.
- **Factor analysis:** You use a *neural network* model to explore the possible causes of poor service quality in call center data. Based on the insights from the preliminary model, you will create a *logistic regression model* to predict strategies for improving customer experience.

What You Will Learn

This tutorial teaches you how to create and work with several types of data mining algorithms. This tutorial is divided into the following lessons:

Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial)
In this lesson, you will create a new project based on the **AdventureWorksDW2012** database, to support

several new data sources views and many more mining models.

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)
In this lesson, you will create a mining model that can be used as part of a forecasting scenario. You will also explore mining models that are built with the Microsoft Time Series algorithm.

You will build models for individual regions, and then build a general model that can be used for cross-prediction.

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)
In this lesson, you will add a new data source view and learn how to work with nested tables and keys. Based on this data, you will create a mining model that can be used as part of a market basket scenario. You will also explore mining models that are built with the Microsoft Association algorithm.

Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)
In this lesson, you will create a mining model that can be used as part of a sequence clustering scenario. You will also learn how to explore mining models that are built with the Microsoft Sequence Clustering algorithm.

Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)
In this lesson, you will create several related mining models, using the Microsoft Neural Network and Microsoft Logistic Regression algorithms. You will also learn to work with data source views to explore data underlying the models.

Requirements

Make sure that the following are installed:

- Microsoft SQL Server 2014
- Microsoft SQL Server Analysis Services
- SQL Server with the **AdventureWorksDW2012** database.

By default, the sample databases are not installed, to enhance security. To install the official databases for Microsoft SQL Server, visit the Microsoft SQL Sample Databases page and select the appropriate version of the sample database.

See Also

Basic Data Mining Tutorial
Bike Buyer DMX Tutorial
Market Basket DMX Tutorial

Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

In the Basic Data Mining tutorial, you created an Analysis Services project that contains a simple data mining solution based on the new **AdventureWorksDW2012** database.

For this tutorial, you will create a separate Analysis Services project by using SQL Server Data Tools (SSDT). You will create a Analysis Services data source that uses **AdventureWorksDW2012**, and add several new data source views to that data source, to support the scenarios and model types.

This lesson consists of the following task:

- Creating a Solution and Data Source (Intermediate Data Mining Tutorial)

Next Step

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

All Lessons

Lesson 1: Creating the Intermediate Data Mining Solution

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)

Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)

See Also

Basic Data Mining Tutorial
Creating and Querying Data Mining Models with DMX: Tutorials (Analysis Services - Data Mining)

Creating a Solution and Data Source (Intermediate Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

To work with data mining, you must first create a project in SQL Server Data Tools (SSDT) using the template, **Analysis Services Multidimensional and Data Mining Project**. When you open the template, it loads into the designer all the schemas that you might need for data mining: data sources, mining structures and mining models, and even cubes if your mining structure uses multidimensional data.

When you create the project, your solution is stored as a local file until the solution is deployed. When you deploy the solution, Analysis Services looks for the Analysis Services server specified in the project properties, and creates a new Analysis Services database with the same name as the project. By default, Analysis Services uses the **localhost** instance for new projects. If you are using a named instance, or if you specified a different name for the default instance, you must change the deployment database property of the project to the location where you want to create your data mining objects.

For more information about Analysis Services projects, see Create an Analysis Services Project (SSDT).

To create a new Analysis Services project for this tutorial

1. Open SQL Server Data Tools (SSDT).
2. On the **File** menu, point to **New**, and then click **Project**.
3. Select **Analysis Services Multidimensional and Data Mining Project** from the **Installed Templates** pane.
4. In the **Name** box, name the new project **DM Intermediate**.
5. Click **OK**.

To change the instance where data mining objects are stored (optional)

1. In SQL Server Data Tools (SSDT), on the **Project** menu, click **Properties**.
2. In the left side of the **Property Pages** pane, click **Deployment**.
3. Verify that the **Server** name is **localhost**. If you are using a different instance, type the name of the instance. If you are using a named instance of Analysis Services, type the machine name and then the instance name. Click **OK**.

To change the deployment properties for a project (optional)

1. In Solution Explorer, right-click the project, and then select **Properties**.

   -- or --

   In SQL Server Data Tools (SSDT), on the **Project** menu, select **Properties**.
2. In the left side of the **Property Pages** pane, click **Deployment**.

   In the **Options** pane, select **Deployment Mode**, and set the options to **Deploy All** to overwrite, or to **Deploy Changes Only** to update objects or add objects.

Creating a Data Source

In the Basic Data Mining Tutorial, you created a *data source* that stores connection information for the

AdventureWorksDW2012 database. Follow the same steps to create the **AdventureWorksDW2012** data source in this solution.

To create a data source

- Creating a Data Source (Basic Data Mining Tutorial)

A single data source can support multiple data source views, and each data source view can have multiple tables. However, because the data source and data source view are deployed to your Microsoft SQL Server Analysis Services database together with the data mining models that you create, as a best practice you should include in each data source view only those tables that are required for each data mining model or group of models.

In the following lessons, you will add data source views to support each of the new scenarios. Only the market basket and sequence clustering lessons use the same data source view; otherwise, each scenario uses a different data source view, so the lessons are independent of each other and can be completed separately.

SCENARIO	DATA INCLUDED IN THE DATA SOURCE VIEW
Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)	Monthly sales reports for bicycle models in different regions, collected as a single view.
Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)	A table containing a list of customer orders, and a nested table showing the individual purchases for each customer.
Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)	The same data that is used for the market basket analysis, with the addition of an identifier that shows the order in which items were purchased.
Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)	A single table containing some preliminary performance tracking data from a call center.

Next Lesson

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

See Also

Data Mining Projects
Data Source Views in Multidimensional Models

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

As the sales analyst for Adventure Works Cycles, you have been asked to forecast the sales of products for the next year. In particular, you have been asked to compare forecasts for the different regions and product lines. Additionally, you have been asked to determine whether sales of different products vary depending on the time of the year.

To find the requested information, in this lesson you will summarize the company's sales data at the monthly level, and you will also summarize sales figures by three regions: Europe, North America, and the Pacific.

After you complete the tasks in this lesson, you will be able to answer the following questions:

- How do the sales of different bike models change over time?
- Are there differences between the patterns for sales in the three regions?
- Can we forecast sales peaks?

The lesson can be completed in two parts:

- Part One introduces the basics of how to create and use a time series model.
- Part Two walks you through creation of a general time series model, based on all regions. You can use this general model for *cross-prediction*.

To complete the tasks in this lesson, which are listed below, you will use the **AdventureWorksDW2012** data source that you created in Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial).

> **WARNING**
> The dates in the Adventure Works Cycles sample database have been updated for this release. If you use an earlier version of Adventure Works Cycles, you can build the model following these steps, but you might see different results.

Creating a Simple Forecasting Model

- Adding a Data Source View for Forecasting (Intermediate Data Mining Tutorial)
- Creating a Forecasting Structure and Model (Intermediate Data Mining Tutorial)
- Modifying the Forecasting Structure (Intermediate Data Mining Tutorial)
- Customizing and Processing the Forecasting Model (Intermediate Data Mining Tutorial)
- Exploring the Forecasting Model (Intermediate Data Mining Tutorial)
- Creating Time Series Predictions (Intermediate Data Mining Tutorial)

Creating a General Forecasting Model for Cross-Prediction

- Advanced Time Series Predictions (Intermediate Data Mining Tutorial)
- Time Series Predictions using Updated Data (Intermediate Data Mining Tutorial)

- Time Series Predictions using Replacement Data (Intermediate Data Mining Tutorial)
- Comparing Predictions for Forecasting Models (Intermediate Data Mining Tutorial)

Next Task in Lesson

Adding a Data Source View for Forecasting (Intermediate Data Mining Tutorial)

Understanding the Requirements for a Time Series Model (Intermediate Data Mining Tutorial)

All Lessons

Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial)

Lesson 2: Forecasting Scenario (Intermediate Data Mining Tutorial)

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)

Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)

See Also

Basic Data Mining Tutorial
Intermediate Data Mining Tutorial (Analysis Services - Data Mining)
Microsoft Time Series Algorithm

Adding a Data Source View for Forecasting (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add a data source view that will be used for the forecasting scenario. A forecasting model requires that the data contains a column that can be used to identify steps in a time series. If you plan to analyze multiple series of data, all series must end on the same date or time step.

To add a data source view

1. In Solution Explorer, right-click **Data Source Views**, and then select **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, under **Relational data sources**, select the **AdventureWorksDW2012** data source. Click **Next**.

   > **NOTE**
   > If you do not have this data source, you can find the steps to create the data source in the Basic Data Mining Tutorial.

4. On the **Select Tables and Views** page, select the table, vTimeSeries (dbo), and then click the right arrow to add it to the data source view.
5. Click **Next**.
6. On the **Completing the Wizard** page, by default the data source view is named Adventure Works DW Multidimensional 2012. Change the name to **SalesByRegion**, and then click **Finish**.

   Data Source View Designer opens and the **SalesByRegion** data source view appears.

Working with the Data Source View

After you have created the data source view, you can explore the data in the following ways:

- Right-click the table vTimeSeries in the designer, and select **Explore Data** to open the selected table in a grid.
- Click **Sampling options** and then use the **Data Exploration Options** dialog box to change the sampling method. Click **Refresh** to load data in the table using the new option settings. For example, you could specify the number of rows to output in the sample, or choose the top n rows.
- Right-click the table vTimeSeries and select **Properties** to assign a new name to the table. You can also select individual columns from the data source view, and the modify the column properties.
- Click anywhere in the data source view design area to create a new query and assign a name to it, to create relationships between tables, or to change the layout of the design area.
- Right-click a table and select **New Named Calculation** to create derived columns, including aggregations. You can also add new tables and views from the data source in this view.

In the next task, you will explore the time series data and determine the best column to use as the time series identifier. You will also learn how to handle gaps in time series data.

Next Task in Lesson

Understanding the Requirements for a Time Series Model (Intermediate Data Mining Tutorial)

See Also

Microsoft Time Series Algorithm

Understanding the Requirements for a Time Series Model (Intermediate Data Mining Tutorial)

2/11/2019 • 4 minutes to read • Edit Online

When you are preparing data for use in a forecasting model, you must ensure that your data contains a column that can be used to identify the steps in the time series. That column will be designated as the `Key Time` column. Because it is a key, the column must contain unique numeric values.

Choosing the right unit for the `Key Time` column is an important part of analysis. For example, suppose your sales data is refreshed on a minute by minute basis. You would not necessarily use minutes as the unit for the time series; you might find it more meaningful to roll up sales data by the day, week, or even month. If you are unsure which unit of time to use, you can create a new data source view for each aggregation, and build related models, to see if different trends emerge at each level of aggregation.

For this tutorial, sales data is collected on a daily basis in the transactional sales database, but for data mining, the data has been pre-aggregated by the month, using a view.

Additionally, it is desirable for analysis that the data have as few gaps as possible. If you plan to analyze multiple series of data, all series should preferably start and end on the same date. If the data has gaps, but the gaps are not at the beginning or end of a series, you can use the MISSING_VALUE_SUBSTITUTION parameter to fill in the series. Analysis Services also provides several options for replacing missing data with values, such as using means or constants.

> **WARNING**
>
> The PivotChart and PivotTable tools that were included in earlier versions of the data source view designer are no longer provided. We recommend that you identify gaps in time series data beforehand, by using tools such as the Data Profiler included in Integration Services.

To identify the time key for the forecasting model

1. In the pane, **SalesByRegion.dsv [Design]**, right-click the table vTimeSeries, and then select **Explore Data**.

   A new tab opens, titled **Explore vTimeSeries Table**.

2. On the **Table** tab, review the data that is used in the TimeIndex and Reporting Date columns.

   Both are sequences with unique values and can both be used as the time series key; however, the data types of the columns are different. The Microsoft Time Series algorithm does not require a `datetime` data type, only that the values be distinct and ordered. Therefore, either column can be used as the time key for the forecasting model.

3. In the data source view design surface, select the column, Reporting Date and select **Properties**. Next, click the column TimeIndex and select **Properties**.

   The field TimeIndex has the data type System.Int32, whereas the field Reporting Date has the data type System.DateTime. Many data warehouses convert date/time values to integers and use the integer column as the key, to improve indexing performance. However, if you use this column, the Microsoft Time Series algorithm will make predictions using future values such as 201014, 201014, and so forth. Because you want to represent your sales data forecast by using calendar dates, you will use the Reporting Date column as the unique series identifier.

To set the key in the data source view

1. In the pane **SalesByRegion.dsv**, select the vTimeSeries table.
2. Right-click the column, Reporting Date, and select **Set Logical Primary Key**.

Handling Missing Data (Optional)

If any series has missing data, you might get an error when you try to process the model. You have several ways to work around missing data:

- You can have Analysis Services fill in missing values, either by calculating a mean, or by using a previous value. You do this by setting the MISSING_VALUE_SUBSTITUTION parameter on the mining model. For more information about this parameter, see Microsoft Time Series Algorithm Technical Reference. For information about how to change parameters on an existing mining model, see View or Change Algorithm Parameters.
- You can alter the data source or filter the underlying view to eliminate ragged series or to replace values. You can do this in the relational data source, or you can modify the data source view by creating custom named queries or named calculations. For more information, see Data Source Views in Multidimensional Models. A later task in this lesson provides an example of how to build both a named query and a custom calculation.

For this scenario, some data is missing at the beginning of one series: that is, there is no data for the T1000 product line until July 2007. Otherwise, all series end on the same date, and there are no missing values.

The requirement of the Microsoft Time Series algorithm is that any series that you include in a single model should have the same **ending** point. Because the T1000 bicycle model was introduced in 2007, the data for this series starts later than for other bicycle models, but the series ends on the same date; therefore the data is usable.

To close the data source view designer

- Right-click the tab, **Explore vTimeSeries Table**, and select **Close**.

Next Task in Lesson

Creating a Forecasting Structure and Model (Intermediate Data Mining Tutorial)

See Also

Microsoft Time Series Algorithm

Creating a Forecasting Structure and Model (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Next, you will use the Data Mining Wizard to create a new mining structure and mining model based on the data source view that you just created. In this task you will specify that the mining model should use the Microsoft Time Series algorithm.

To create a forecasting mining structure

1. In Solution Explorer in SQL Server Data Tools (SSDT), right-click **Mining Structures** and select **New Mining Structure**.
2. On the **Welcome to the Data Mining Wizard** page, click **Next**.
3. On the **Select the Definition Method** page, verify that **From existing relational database or data warehouse** is selected, and then click **Next**.
4. On the **Create the Data Mining Structure** page, under **Which data mining technique do you want to use?**, select **Microsoft Time Series**, and then click **Next**.
5. On the **Select Data Source View** page, under **Available data source views**, select **SalesByRegion**.
6. Click **Next**.
7. On the **Specify Table Types** page, ensure that the check box in the **Case** column for the vTimeSeries table is selected, and then click **Next**.
8. On the **Specify the Training Data** page, select the check boxes in the **Key** column for the ModelRegion and ReportingDate columns.

   ReportingDate should be selected by default, because you specified this column as the logical primary key when you created the data source view. By adding ModelRegion as a second key, you are telling the algorithm to create a separate time series for each combination of model and region listed in this field.
9. Select the check boxes in the **Input** and **Predictable** columns for the Quantity, column, and then click **Next**.

   By selecting **Predictable**, you indicate that you want to create forecasts on the data in this column. However, because you want to base the forecasts on past data, you must also add the column as an input.
10. On the page **Specify Columns' Content and Data Type**, review the selections.

    The ModelRegion column is designated as a **Key** column and the ReportingDate column is automatically designated as a **Key Time** column. You can have only one of each type of key.
11. Click **Next**.
12. On the **Completing the Wizard** page, for **Mining structure name**, type `Forecasting`.

    > **NOTE**
    > The option to enable drillthrough is not available for time series models.
13. In **Mining model name**, type `Forecasting`, and then click **Finish**.

Data Mining Designer opens to display the `Forecasting` mining structure that you just created.

Next Task in Lesson

Modifying the Forecasting Structure (Intermediate Data Mining Tutorial)

See Also

Data Mining Designer
Microsoft Time Series Algorithm

Modifying the Forecasting Structure (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

The mining structure that you created in the previous task contains a single forecasting model. Before you process and explore the model, you must change its structure slightly and modify one of its properties.

Modifying the Mining Structure

You can change the mining structure by using the **Mining Structure** tab of Data Mining Designer. When you created the model with the Data Mining Wizard, you used three columns: ReportingDate, ModelRegion, and Quantity. However, the **Forecasting** table also contains an Amount column, which you can use to forecast the amount of sales. By using the **Mining Structure** tab, you can add this column from the data source view to the mining structure.

To add the Amount column to the Forecasting mining structure

1. On the **Mining Structure** tab of Data Mining Designer, in the **Data Source View** pane, select the Amount column in the vTimeSeries table.

2. Drag the Amount column from the **Data Source View** pane into the list of columns for the **Forecasting** structure.

   The Amount column is now included in the **Forecasting** mining structure.

Modifying the Columns in the Mining Model

Because you added a new column to the structure, you must define how the model will use the column. You can specify how the column will be used on the **Mining Models** tab of Data Mining Designer.

The **Mining Models** tab lists the columns that the mining structure contains in the **Structure** column of the grid, and lists the columns that the mining model contains in the column that has the name of the model, in this case **Forecasting**. Click the names of the columns to make modifications. In the **Forecasting** mining model, the Amount column is used as an input column and is also used to forecast future sales. Therefore, you must set the properties of the column so that it can be used as both an input column and a predictable column.

> **NOTE**
>
> In the **Mining Models** tab, you can also create new models based on the same structure, and you can adjust the algorithm and column properties for each model. However, you must process the model before these changes take effect.

To define how the Amount column will be used

1. In the **Forecasting** column of the grid on the **Mining Models** tab, click the cell in the Amount row.

2. Select **Predict** from the list.

   The Amount column is now both an input column and a predictable column.

You can also change the properties of individual columns by selecting the column and opening the **Properties** window. To open the **Properties** window, right-click the column name, and then select **Properties**. If you change a property within the column for an individual model, you can change the properties only for that model. However, when you change a property within the **Structure** column, the change affects every model that is associated with

the structure. Whenever you make changes to the model or structure, you must reprocess to see the effects.

Next Task in Lesson

Customizing and Processing the Forecasting Model (Intermediate Data Mining Tutorial)

See Also

Mining Structures (Analysis Services - Data Mining)
Mining Models (Analysis Services - Data Mining)

Customizing and Processing the Forecasting Model (Intermediate Data Mining Tutorial)

2/11/2019 • 4 minutes to read • Edit Online

The Microsoft Time Series algorithm provides parameters that affect how a model is created, and how time data is analyzed. Changing these properties can significantly affect how the mining model makes predictions.

For this task in the tutorial, you will perform the following tasks to modify the model:

1. You will customize the way your model handles time periods by adding a new value for the *PERIODICITY_HINT* parameter.
2. You will learn about two other important parameters for the Microsoft Time Series algorithm: FORECAST_METHOD, which lets you control the method used for forecasting, and PREDICTION_SMOOTHING, which lets you customize the blend of long-term and short-term predictions.
3. Optionally, you will tell the algorithm how you want missing values to be imputed.
4. After all the changes have been made, you will deploy and process the model.

Setting Time Series Parameters

Periodicity Hints

The *PERIODICITY_HINT* parameter provides the algorithm with information about additional time periods that you expect to see in the data. By default, time series models will try to automatically detect a pattern in the data. However, if you already know the expected time cycle, providing a periodicity hint can potentially improve the accuracy of the model. However, if you provide the wrong periodicity hint, it can decrease accuracy; therefore, if you are not sure what value should be used, it is best to use the default.

For example, the view used for this model aggregates sales data from Adventure Works DW Multidimensional 2012 on a monthly basis. Therefore each time slice used by the model represents one month, and all predictions will also be in terms of months. Since there are 12 months in a year and you expect that sales patterns more or less repeat on a yearly basis, you will set the *PERIODICITY_HINT* parameter to `12`, to indicate that 12 time slices (months) constitute one complete sales cycle.

Forecasting Method

The *FORECAST_METHOD* parameter controls whether the time series algorithm is optimized for short-term or long-term predictions. By default, the *FORECAST_METHOD* parameter is set to MIXED, which means that two different algorithms are blended and balanced to provide good results for both short-term and long-term prediction.

However, if you know that you want to use a particular algorithm, you can change the value to either ARIMA or ARTXP.

Weighting Long-Term vs. Short-Term Predictions

You can also customize the way that long-term and short-term predictions are combined by using the PREDICTION_SMOOTHING parameter. By default, this parameter is set to 0.5, which generally provides the best balance for overall accuracy.

To change the algorithm parameters

1. On the **Mining Models** tab, right-click **Forecasting**, and select **Set Algorithm Parameters**.

2. In the `PERIODICITY_HINT` row of the **Algorithm Parameters** dialog box, click the **Value** column, then type `{12}`, including the braces.

   By default, the algorithm will also add the value {1}.

3. In the `FORECAST_METHOD` row, verify that the **Value** text box is either blank or set to `MIXED`. If a different value has been entered, type `MIXED` to change the parameter back to the default value.

4. In the **PREDICTION_SMOOTHING** row, verify that the **Value** text box is either blank or set to 0.5. If a different value has been entered, click **Value** and type `0.5` to change the parameter back to the default value.

   > **NOTE**
   > The PREDICTION_SMOOTHING parameter is available only in SQL Server Enterprise. Therefore, you cannot view or change the value of the PREDICTION_SMOOTHING parameter in SQL Server Standard. However, the default behavior is to use both algorithms and weight them equally.

5. Click **OK**.

Handling Missing Data (Optional)

In many cases, your sales data might have gaps that are filled with nulls, or a store might have failed to meet the reporting deadline, leaving an empty cell at the end of the series. In such scenarios, Analysis Services raises the following error and will not process the model.

"Error (Data mining): Time stamps not synchronized starting with series <series name>, of the mining model, <model name>. All time series must end at the same time mark and cannot have arbitrarily missing data points. Setting the MISSING_VALUE_SUBSTITUTION parameter to Previous or to a numeric constant will automatically patch missing data points where possible."

To avoid this error, you can specify that Analysis Services automatically provide new values to fill in the gaps by using any one of the following methods:

- Using an average value. The mean is calculated by using all valid values in the same data series.
- Using the previous value. You can substitute previous values for multiple missing cells, but you cannot fill starting values.
- Using a constant value that you supply.

To specify that gaps be filled by averaging values

1. On the **Mining Models** tab, right-click the **Forecasting** column, and select **Set Algorithm Parameters**.

2. In the **Algorithm Parameters** dialog box, in the **MISSING_VALUE_SUBSTITUTION** row, click the **Value** column, and type `Mean`.

Build the Model

To use the model, you must deploy it to a server, and process the model by running the training data through the algorithm.

To process the forecasting model

1. On the **Mining Model** menu of SQL Server Data Tools, select **Process Mining Structure and All Models**.

2. At the warning asking whether you want to build and deploy the project, click **Yes**.

3. In the **Process Mining Structure - Forecasting** dialog box, click **Run**.

   The **Process Progress** dialog box opens to display information about model processing. Model processing may take some time.

4. After processing is complete, click **Close** to exit the **Process Progress** dialog box.

5. Click **Close** again to exit the **Process Mining Structure - Forecasting** dialog box.

Next Task in Lesson

Exploring the Forecasting Model (Intermediate Data Mining Tutorial)

See Also

Microsoft Time Series Algorithm Technical Reference
Microsoft Time Series Algorithm
Processing Requirements and Considerations (Data Mining)

Exploring the Forecasting Model (Intermediate Data Mining Tutorial)

2/11/2019 • 9 minutes to read • Edit Online

Now that you have built the forecasting mining model, you can explore the results by using the **Mining Model Viewer** tab of Data Mining Designer. The Microsoft Time Series Viewer contains two tabs: **Charts** and **Model**.

Additionally, you can use the Microsoft Generic Tree Viewer with all models. Each view presents a slightly different picture of the information in the time series model.

- Charts Tab
- Model Tab
- Microsoft Generic Content Viewer

Charts Tab

The **Charts** tab of the Microsoft Time Series Viewer graphically shows you each of the series, including historical data and predictions. Each line in the time series graph represents a unique combination of product, region, and predictable attribute.

The legend on the right side of the viewer lists the time series that available, based on the selections in the drop-down list. You can select and clear the check boxes in the legend to control which time series displays in the graph.

You can also change the display options, such as the colors used for each time series, or whether values are displayed at points in the chart.

To select a time series

1. Click the **Charts** tab of the **Mining Model Viewer** tab, if it is not visible.
2. Click the drop-down list to the right of the chart view, and select all the check boxes. Click **OK**.

   The chart should now contain 24 different series lines.
3. In the check boxes to the right of the chart, clear the boxes to temporarily hide the lines for all series that are based on Amount.

   Now, clear the check boxes related to the R750 and R250 bicycles.

   The chart now contains just the following six series lines, so that can you more easily compare trends for the M200 and T1000 bicycles.

   - **M200 Europe: Quantity**
   - **M200 North America: Quantity**
   - **M200 Pacific: Quantity**
   - **T1000 Europe: Quantity**
   - **T1000 North America: Quantity**
   - **T1000 Pacific: Quantity**

The chart that is displayed in this viewer includes both historical and predicted data. Predicted data is shaded to differentiate it from historical data. To make it easier to compare different series, you can also change the colors associated with each line in the graph. For more information, see Change the Colors Used in the Data Mining Viewer.

From the trend lines, you can see that total sales for all regions are generally increasing, with a peak every 12 months in December. From the chart, you can also see that the data for the T1000 bicycle starts much later than the data for the other product series. That is because it is a newer product, but because this series is based on much less data, the predictions might not be as accurate.

By default, five prediction steps are shown for each time series, displayed as dotted lines. You can change this value to view more or fewer predictions. You can also graphically view the standard deviation for the predictions by adding error bars to the chart.

To change prediction and display options in the Chart view

1. Try changing the value for **Prediction Steps** gradually, increasing it from **5** to **10**, and then back to **6**.

   When the historical data has large fluctuations, the fluctuations tend to be repeated or even amplified as you increase the number of predictions. You probably need to do some research at this point, to understand the cause of the big increase in the historical data and then decide whether to accept these results, seek some kind of correction in the source data, or apply some kind of smoothing in the model.

2. Select the **Show Deviations** check box.

   This option displays the estimated error for each predicted value.

3. Note the scale of the X-axis. The changes over both historical and predicted data are always expressed as a percentage, but the actual values are adjusted automatically to fit all values onto the graph. Therefore you need to be careful when comparing models to not rely on visuals alone. To get the exact value, or the percentage increase and value for predictions, pause the mouse over the dotted line or solid lines, or click the lines to view the values in the **Mining Legend**.

   Tip: If the **Mining Legend** is not visible, switch to **Model** view, right-click any node, and select **Show Legend**.

From looking at these trends, you are concerned about the lack of data for some of the series, and wonder if you might get more reliable predictions by averaging sales by model, or perhaps averaging sales by region. You will explore this approach in a later lesson in this tutorial.

Back to Top

Model Tab

The **Model** tab of the Microsoft Time Series Viewer in Data Mining Designer lets you view the forecasting model in the form of a tree graph.

First, notice that because your data describes two different measures (Amount and Quantity) for sales of multiple product lines (T1000, etc.) in three different regions (Europe, North America, and Pacific), the model that you built actually contains 24 different trees, each tree representing a model of the sales patterns for a different combination of region, product, and predictable attribute.

You can choose which combination of product line, region, and sales metric you want to view by selecting a series from the **Tree** dropdown list on the **Model** tab.

So what can you learn from viewing the model as a tree? As an example, let's compare two models, one that has several levels in the tree, and one that has a single node.

- When a tree graph contains a single node, it means the trend found in the model is mostly homogenous over time. You can use this single node, labeled **All**, to view the formula that describes the relationship between the input variables and the outcome.

- When a tree graph for a time series has multiple branches, it means the time series that was detected is too complex to be represented as a single equation. Instead, the tree graph might contain multiple branches, each branch labeled with the conditions that caused the tree to *split*. When the tree splits, each branch represents a different segment of time, inside which the trend can be described as a single equation.

  For example, if you look at the chart graph and see a sudden jump in sales volume starting sometime in September and continuing through a year-end holiday, you can switch to the **Model** view to see the exact date where the trend changed. The branches in the tree that represent "before September" and "after September" would contain different formulas: one formula that mathematically describes the sales trends up to the split, and another formula that describes sales trends for September through the year-end holiday.

To explore the decision tree for a time series model

1. In the **Tree** list on the **Model** tab of the viewer, select the **T1000 Europe: Amount** series.

   Click the node labeled **All**.

   For an **All** node, the ToolTip that appears includes information such as, the number of cases in the entire series, and time series equations derived from analysis of the data.

2. If the **Mining Legend** is not visible, right-click the node and select **Show Legend**.

   The **Mining Legend** provides much the same information that is in the Tooltip. If any of your independent variables are discrete, you will also see a histogram that shows the distribution of variables in the node.

3. Now select a different time series to view. Using the **Tree** list on the **Model** tab of the viewer, select the **M200 North America: Amount** series.

   The tree graph now contains an **All** node and two child nodes. By looking at the labels on the child nodes, you can understand at what point the trend line changed.

   For each child node, the description in the **Mining Legend** also includes the count of cases in each branch of the tree.

The following list describes some additional features in the tree viewer:

- You can change the variable that is represented in the chart by using the **Background** control. By default, nodes that are darker contain more cases, because the value of **Background** is set to **Population**. To see

just how many cases there are in a node, pause the mouse over a node and view the ToolTip that appears, or click the node and view the numbers in the **Node Legend** window.

- The regression formula for the node can also be viewed in the ToolTip, or by clicking the node. If you have created a mixed model, you can see two formulas, one for ARTXP (in the leaf nodes) and one for ARIMA (in the root node of the tree).
- The little diamonds are used in nodes that represent continuous numbers. The range of the attributes is shown in the bar on which the diamond rests. The diamond is centered on the mean for the node, and the width of the diamond represents the variance of the attribute at that node.

Back to Top

(Optional) Generic Content Tree Viewer

In addition to the custom viewer for time series, Analysis Services provides the **MicrosoftGeneric Content Tree Viewer** for use with all data mining models. This viewer provides some advantages:

- **Microsoft Time Series Viewer**: This view merges the results of the two algorithms. Although you can view each series separately, you cannot determine how the results of each algorithm were combined. Also, in this view, the Tooltips and Mining Legend show only the most important statistics.
- **Generic Content Tree Viewer**: Lets you browse and view all of the data series that were used in the model at one time, and if you have created a mixed model, both the ARIMA and ARTXP trees are displayed in the same graph.

  You can use this viewer to get all the statistics from both algorithms, as well as distributions of the values.

  Recommended for expert users of data mining who want to know more about the ARIMA and ARTXP analyses.

To view details for a particular data series in the generic content viewer

1. In the **Mining Model Viewer** tab, select **Microsoft Generic Content Tree Viewer** from the **Viewer** drop-down list.
2. In the **Node Caption** pane, click the topmost (All) node.
3. In the **Node Details** pane, view the value for ATTRIBUTE_NAME.

   This value shows you which series, or combination of product and region, is contained in this node. In the AdventureWorks example, the topmost node is for the M200 Europe series.
4. In the **Node Caption** pane, locate the first node that has child nodes.

   If a series node has children, the tree view that appears on the **Model** tab of the Microsoft Time Series Viewer will also have a branching structure.
5. Expand the node and click one of the child nodes.

   The NODE_DESCRIPTION column of the schema contains the condition that caused the tree to split.
6. In the **Node Caption** pane, click the topmost ARIMA node, and expand the node until all child nodes are visible.
7. In the **Node Details** pane, view the value for ATTRIBUTE_NAME.

   This value tells you which time series is contained in this node. The topmost node in the ARIMA section should match the topmost node in the (All) section. In the AdventureWorks example, this node contains the ARIMA analysis for the series, M200 Europe.

For more information, see Mining Model Content for Time Series Models (Analysis Services - Data Mining).

Back to Top

Next Task in Lesson

Creating Time Series Predictions (Intermediate Data Mining Tutorial)

See Also

Time Series Model Query Examples
Microsoft Time Series Algorithm Technical Reference

Creating Time Series Predictions (Intermediate Data Mining Tutorial)

2/11/2019 • 5 minutes to read • Edit Online

In the previous tasks in this lesson, you created a time series model and explored the results. By default, Analysis Services always creates a set of five (5) predictions for a time series model and displays the predicted values as part of the forecasting chart. However, you can also create forecasts by building Data Mining Extensions (DMX) prediction queries.

In this task, you will create a prediction query that generates the same predictions that you saw in the viewer. This task assumes that you have already completed the lessons in the Basic Data Mining Tutorial and are familiar with how to use Prediction Query Builder. You will now learn how to create queries specific to time series models.

Creating Time Series Predictions

Typically, the first step in creating a prediction query is to select a mining model and input table. However, a time series model does not require additional input for a regular prediction. Therefore, you do not need to specify a new source of data when making predictions, unless you are adding data to the model or replacing the data.

For this lesson, you must specify the number of prediction steps. You can specify the series name, to get a prediction for a particular combination of a product and a region.

To select a model and input table

1. On the **Mining Model Prediction** tab of the Data Mining Designer, in the **Mining Model** box, click **Select Model**.
2. In the **Select Mining Model** dialog box, expand the Forecasting structure, select the **Forecasting** model from the list, and then click **OK**.
3. Ignore the **Select Input Table(s)** box.

   > **NOTE**
   >
   > For a time series model, you do not need to specify a separate input unless you are doing cross-prediction.

4. In the **Source** column, in the grid on the **Mining Model Prediction** tab, click the cell in the first empty row, and then select **Forecasting mining model**.
5. In the **Field** column, select **Model Region**.

   This action adds the series identifier to the prediction query to indicate the combination of model and region to which the prediction applies.

6. Click the next empty row in the **Source** column, and then select **Prediction Function**.
7. In the **Field** column, select **PredictTimeSeries**.

> **NOTE**
> You can also use the `Predict` function with time series models. However, by default, the Predict function creates only one prediction for each series. Therefore, to specify multiple prediction steps, you must use the **PredictTimeSeries** function.

8. In the **Mining Model** pane, select the mining model column, **Amount.** Drag Amount to the **Criteria/Arguments** box for the **PredictTimeSeries** function that you added earlier.
9. Click the **Criteria/Arguments** box, and type a comma, followed by **5**, after the field name.

   The text in the **Criteria/Arguments** box should now display the following:

   `[Forecasting].[Amount],5`
10. In the **Alias** column, type `PredictAmount`.
11. Click the next empty row in the **Source** column, and then select **Prediction Function** again.
12. In the **Field** column, select **PredictTimeSeries**.
13. In the **Mining Model** pane, select the column Quantity, and then drag it into the **Criteria/Arguments** box for the second **PredictTimeSeries** function.
14. Click the **Criteria/Arguments** box, and type a comma, followed by **5**, after the field name.

    The text in the **Criteria/Arguments** box should now display the following:

    `[Forecasting].[ Quantity],5`
15. In the **Alias** column, type `PredictQuantity`.
16. Click **Switch to query result view**.

    The results of the query are displayed in tabular format.

Remember that you created three different types of results in the query builder, one that uses values from a column, and two that get predicted values from a prediction function. Therefore, the results of the query contain three separate columns. The first column contains the list of product and region combinations. The second and third columns each contain a nested table of prediction results. Each nested table contains the time step and predicted values, such as the following table:

Example results (amounts are truncated to two decimal places):

M200 Europe PredictAmount

$TIME	AMOUNT
7/25/2008	99978.00
8/25/2008	145575.07
9/25/2008	116835.19
10/25/2008	116537.38
11/25/2008	107760.55

M200 Europe PredictQuantity

$TIME	QUANTITY
7/25/2008	52
8/25/2008	67
9/25/2008	58
10/25/2008	57
11/25/2008	54

M200 North America - PredictAmount

$TIME	AMOUNT
7/25/2008	348533.93
8/25/2008	340097.98
9/25/2008	257986.19
10/25/2008	374658.24
11/25/2008	379241.44

M200 North America - PredictQuantity

$TIME	QUANTITY
7/25/2008	272
8/25/2008	152
9/25/2008	250
10/25/2008	181
11/25/2008	290

WARNING

The dates that are used in the sample database have changed for this release. If you are using an earlier version of the sample data, you might see different results.

Saving the Prediction Results

You have several different options for using the prediction results. You can flatten the results, copy the data from the Results view, and paste it into an Excel worksheet or other file.

To simplify the process of saving results, Data Mining Designer also provides the ability to save the data to a data source view. The functionality for saving results to a data source view is available only in SQL Server Data Tools

(SSDT). The results can only be stored in a flattened format.

To flatten the results in the Results pane

1. In the Prediction Query Builder, click **Switch to query design view**.

   The view changes to allow manual editing of the DMX query text.

2. Type the `FLATTENED` keyword after the `SELECT` keyword. The complete query text should be as follows:

```
SELECT FLATTENED
  [Forecasting].[Model Region],
  (PredictTimeSeries([Forecasting].[Amount],5)) as [PredictAmount],
  (PredictTimeSeries([Forecasting].[Quantity],5)) as [PredictQuantity]
FROM
  [Forecasting]
```

3. Optionally, you can type a clause to restrict the results, such as the following example:

```
SELECT FLATTENED
  [Forecasting].[Model Region],
  (PredictTimeSeries([Forecasting].[Amount],5)) as [PredictAmount],
  (PredictTimeSeries([Forecasting].[Quantity],5)) as [PredictQuantity]
FROM
  [Forecasting]
WHERE [Forecasting].[Model Region] = 'M200 North America'
OR [Forecasting].[Model Region] = 'M200 Europe'
```

4. Click **Switch to query result view**.

To export prediction query results

1. Click **Save query results**.

2. In the **Save Data Mining Query Result** dialog box, for **Data Source**, select **AdventureWorksDW2012**. You can also create a data source if you want to save the data to a different relational database.

3. In the **Table Name** column, type a new temporary table name, such as **Test Predictions**.

4. Click **Save**.

> **NOTE**
> To view the table that you created, create a connection to the database engine of the instance where you saved the data, and create a query.

Conclusion

You have learned how to build a basic time series model, interpret the forecasts, and create predictions.

The remaining tasks in this tutorial are optional, and describe advanced time series predictions. If you decide to go on, you will learn how to add new data to your model and create predictions on the extended series. You will also learn how to perform cross-prediction, by using the trend in the model but replacing the data with a new series of data.

Next Lesson

Advanced Time Series Predictions (Intermediate Data Mining Tutorial)

See Also

Time Series Model Query Examples

Advanced Time Series Predictions (Intermediate Data Mining Tutorial)

2/11/2019 • 5 minutes to read • Edit Online

You saw from exploring the forecasting model that although sales in most of the regions follow a similar pattern, some regions and some models, such as the M200 model in the Pacific region, exhibit very different trends. This does not surprise you, as you know that differences among regions are common and can be caused by many factors, including marketing promotions, inaccurate reporting, or geopolitical events.

However, your users are asking for a model that can be applied worldwide. Therefore, to minimize the effect of individual factors on projections, you decide to build a model that is based on aggregated measures of worldwide sales. You can then use this model to make predictions for each individual region.

In this task, you will build all the data sources that you need to perform the advanced prediction tasks. You will create two data source views for use as inputs to the prediction query, and one data source view to use in building a new model.

Steps

1. Prepare the extended sales data (for prediction)
2. Prepare the aggregated data (for building the model)
3. Prepare the series data (for cross-prediction)
4. Predict using EXTEND
5. Create the cross-prediction model
6. Predict using REPLACE
7. Review the new predictions

Creating the New Extended Sales Data

To update your sales data, you will need to get the latest sales figures. Of particular interest are the data just in from the Pacific region, which launched a regional sales promotion to call attention to the new stores and raise awareness of their products.

For this scenario, we'll assume that the data has been imported from an Excel workbook that contains just three months of new data for a couple of regions. You'll create a table for the data using a Transact-SQL script, and then define a data source view to use for prediction.

Create the table with new sales data

1. In a Transact-SQL query window, execute the following statement to add the sales data to the AdventureWorksDW database (or any other database).

```
USE [database name];
GO
IF OBJECT_ID ([dbo].[NewSalesData]) IS NOT NULL
    DROP TABLE [dbo].[NewSalesData];
GO
CREATE TABLE [dbo].[NewSalesData]([Series] [nvarchar](255) NULL,
[NewDate] [datetime] NULL,
[NewQty] [float] NULL,
[NewAmount] [money] NULL) ON [PRIMARY]

GO
```

2. Insert the new values using the following script.

```
INSERT INTO [NewSalesData]
(Series,NewDate,NewQty,NewAmount)
VALUES('T1000 Pacific', '7/25/08', 55, '$130,170.22'),
('T1000 Pacific', '8/25/08', 50, '$114,435.36 '),
('T1000 Pacific', '9/25/08', 50, '$117,296.24 '),
('T1000 Europe', '7/25/08', 37, '$88,210.00 '),
('T1000 Europe', '8/25/08', 41, '$97,746.00 '),
('T1000 Europe', '9/25/08', 37, '$88,210.00 '),
('T1000 North America', '7/25/08', 69, '$164,500.00 '),
('T1000 North America', '8/25/08', 66, '$157,348.00 '),
('T1000 North America', '9/25/08', 58, '$138,276.00 '),
('M200 Pacific', '7/25/08', 65, '$149,824.35'),
('M200 Pacific', '8/25/08', 54,  '$124,619.46'),
('M200 Pacific', '9/25/08', 61, '$141,143.39'),
('M200 Europe', '7/25/08', 75, '$173,026.00'),
('M200 Europe', '8/25/08', 76, '$175,212.00'),
('M200 Europe', '9/25/08', 84, '$193,731.00'),
('M200 North America', '7/25/08', 94, '$216,916.00'),
('M200 North America', '8/25/08', 94, '$216,891.00'),
('M200 North America', '9/25/08', 91,'$209,943.00');
```

WARNING

The quotation marks are used with the currency values to prevent problems with the comma separator and the currency symbol. You could also pass in the currency values in this format: `130170.22`

Note that the dates used in the sample database have changed for this release. If you are using an earlier edition of AdventureWorks, you might need to adjust the inserted dates accordingly.

Create a data source view using the new sales data

1. In **Solution Explorer**, right-click **Data Source Views**, and then select **New Data Source View**.

2. In the Data Source View wizard, make the following selections:

   Data Source: Adventure Works DW Multidimensional 2012

   Select Tables and Views: Select the table that you just created, NewSalesData.

3. Click **Finish**.

4. In the Data Source View design surface, right-click NewSalesData, and then select **Explore Data** to verify the data.

> **WARNING**
> You will use this data for prediction only, so it does not matter that the data is incomplete.

Creating the Data for the Cross-Prediction Model

The data that was used in the original forecasting model was already grouped somewhat by the view vTimeSeries, which collapsed several bike models into a smaller number of categories, and merged results from individual countries into regions. To create a model that can be used for world-wide projections, you will create some additional simple aggregations directly in the Data Source View Designer. The new data source view will contain just a sum and an average of the sales of all products for all regions.

After you have created the data source used for the model, you must create a new data source view to use for prediction. For example, if you want to predict sales for Europe using the new worldwide model, you must feed in data for the Europe region only. So you will set up a new data source view that filters the original data, and change the filter condition for each set of prediction queries.

To create the model data using a custom data source view

1. In **Solution Explorer**, right-click **Data Source Views**, and then select **New Data Source View**.
2. On the welcome page of the wizard, click **Next**.
3. On the **Select Data Source** page, select Adventure Works DW Multidimensional 2012, and then click **Next**.
4. In the page, **Select Tables and Views**, do not add any tables-just click **Next**.
5. On the page, **Completing the Wizard**, type the name `AllRegions`, and then click **Finish**.
6. Next, right-click the blank data source view design surface, and then select **New Named Query**.
7. In the **Create Named Query** dialog box, for **Name**, type `AllRegions`, and for **Description**, type **Sum and average of sales for all models and regions**.
8. In the SQL text pane, type the following statement and then click OK:

```
SELECT ReportingDate,
SUM([Quantity]) as SumQty, AVG([Quantity]) as AvgQty,
SUM([Amount]) AS SumAmt, AVG([Amount]) AS AvgAmt,
'All Regions' as [Region]
FROM dbo.vTimeSeries
GROUP BY ReportingDate
```

9. Right-click the `AllRegions` table, and then select **Explore Data**.

To create the series data for cross-prediction

1. In **Solution Explorer**, right-click **Data Source Views**, and then select **New Data Source View**.
2. In the Data Source View wizard, make the following selections:

   Data Source: Adventure Works DW Multidimensional 2012

   Select Tables and Views: Do not select any tables

   Name: `T1000 Pacific Region`
3. Click **Finish**.
4. Right-click the empty design surface for **T1000 Pacific Region.dsv**, and then select **New Named Query**.

The **Create Named Query** dialog box appears. Retype the name, and then add the following description:

Name: `T1000 Pacific Region`

Description: **Filter** `vTimeSeries` **by region and model**

5. In the text pane, type the following query, and then click OK:

```
SELECT ReportingDate, ModelRegion, Quantity, Amount
FROM dbo.vTimeSeries
WHERE (ModelRegion = N'T1000 Pacific')
```

> **NOTE**
>
> Since you will need to create predictions for each series separately, you might want to copy the query text and save it to a text file so that you can re-use it for the other data series.

6. In the Data Source View design surface, right-click T1000 Pacific, and then select **Explore Data** to verify that the data is filtered correctly.

   You will use this data as the input to the model when creating cross-prediction queries.

Next Task in Lesson

Time Series Predictions using Updated Data (Intermediate Data Mining Tutorial)

See Also

Microsoft Time Series Algorithm
Microsoft Time Series Algorithm Technical Reference
Data Source Views in Multidimensional Models

Time Series Predictions using Updated Data (Intermediate Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

Creating Predictions using the Extended Sales Data

In this lesson, you will create a prediction query that adds the new sales data to the model. By extending the model with new data, you can get up-to-date predictions that include the newest data points.

Creating time series predictions that use new data is easy: you simply add the parameter EXTEND_MODEL_CASES to the PredictTimeSeries (DMX) function, specify the source of the new data, and specify how many predictions you want to get.

> **WARNING**
>
> The parameter EXTEND_MODEL_CASES is optional; by default the model is extended any time that you create a time series prediction query by joining new data as inputs.

To build the prediction query and add new data

1. If the model is not already open, double-click the Forecasting structure, and in Data Mining Designer, click the **Mining Model Prediction** tab.
2. In the **Mining Model** pane, the model Forecasting should already be selected. If it is not selected, click **Select Model**, and then select the model, Forecasting.
3. In the **Select Input Table(s)** pane, click **Select Case Table**.
4. In the **Select Table** dialog box, select the data source, Adventure Works DW Multidimensional 2012.

   From the list of data source views, select NewSalesData and then click **OK**.
5. Right-click the surface of the design area and select **Modify Connections**.
6. Using the **Modify Mapping** dialog box, map the columns in the model to the columns in the external data as follows:
   - Map the ReportingDate column in the mining model to the NewDate column in the input data.
   - Map the Amount column in the mining model to the NewAmount column in the input data.
   - Map the Quantity column in the mining model to the NewQty column in the input data.
   - Map the ModelRegion column in the mining model to the Series column in the input data.
7. Now you will build the prediction query.

   First, add a column to the prediction query to output the series the prediction applies to.

   a. In the grid, click the first empty row, under **Source**, and then select Forecasting.

   b. In the **Field** column, select Model Region and for **Alias**, type `Model Region`.
8. Next, add and edit the prediction function.

   a. Click an empty row, and under **Source**, select **Prediction Function**.

b. For **Field**, select **PredictTimeSeries**.

c. For **Alias**, type **Predicted Values**.

d. Drag the field Quantity from the **Mining Model** pane into the **Criteria/Argument** column.

e. In the **Criteria/Argument** column, after the field name, type the following text:
5,EXTEND_MODEL_CASES

The complete text of the **Criteria/Argument** text box should be as follows:
`[Forecasting].[Quantity],5,EXTEND_MODEL_CASES`

9. Click **Results** and review the results.

The predictions begin in July (the first time slice after the end of the original data) and end at November (the fifth time slice after the end of the original data).

You can see that to use this type of prediction query effectively, you need to know when the old data ends, as well as how many time slices there are in the new data.

For example, in this model, the original data series ended in June, and the data is for the months of July, August, and September.

Predictions that use EXTEND_MODEL_CASES always begin at the end of the original data series. Therefore, if you want to get only the predictions for the unknown months, you need to specify the starting point and the end point for prediction. Both values are specified as a number of time slices starting at the end of the old data.

The following procedure demonstrates how to do this.

Change the start and end points of the predictions

1. In Prediction Query Builder, click **Query** to switch to DMX view.

2. Locate the DMX statement that contains the PredictTimeSeries function and change it as follows:

```
PredictTimeSeries([Forecasting 12].[Quantity],4,6,EXTEND_MODEL_CASES)
```

3. Click **Results** and review the results.

Now the predictions begin at October (the fourth time slice, counting from the end of the original data) and end at December (the sixth time slice, counting from the end of the original data).

Next Task in Lesson

Time Series Predictions using Replacement Data (Intermediate Data Mining Tutorial)

See Also

Microsoft Time Series Algorithm Technical Reference
Mining Model Content for Time Series Models (Analysis Services - Data Mining)

Time Series Predictions using Replacement Data (Intermediate Data Mining Tutorial)

2/11/2019 • 5 minutes to read • Edit Online

In this task, you will build a new model based on worldwide sales data. Then, you will create a prediction query that applies the worldwide sales model to one of the individual regions.

Building a General Model

Remember that your analysis of the results of the original mining model revealed big differences between regions and between product lines. For example, sales in North America were strong for the M200 model, while sales of the T1000 model did not do as well. However, the analysis is complicated by the fact that some series didn't have much data, or data started at a different point in time. Some data was also missing.

To address some of the data quality issues, you decide to merge the data from sales around the world, and use that set of general sales trends to build a model that can be applied to predict future sales in any region.

When you create predictions, you will use the pattern that is generated by training on worldwide sales data, but you will replace the historical data points with the sales data for each individual region. That way, the shape of the trend is preserved but the predicted values are aligned with the historical sales figures for each region and model.

Performing Cross-Prediction with a Time Series Model

The process of using data from one series to predict trends in another series is called cross-prediction. You can use cross-prediction in many scenarios: for example, you might decide that television sales are a good predictor of overall economic activity, and apply a model trained on television sales to general economic data.

In SQL Server Data Mining, you perform cross-prediction by using the parameter REPLACE_MODEL_CASES within the arguments to the function, PredictTimeSeries (DMX).

In the next task, you will learn how to use REPLACE_MODEL_CASES. You will use the merged world sales data to build a model, and then create a prediction query that maps the general model to the replacement data.

It is assumed that you are familiar with how to build data mining models by now, and so the instructions for building the model has been simplified.

To build a mining structure and mining model using the aggregated data

1. In **Solution Explorer**, right-click **Mining Structures**, and then select **New Mining Structure** to start the Data Mining Wizard.

2. In the Data Mining Wizard, make the following selections:

   - Algorithm: Microsoft Time Series

   - Use the data source that you built earlier in this advanced lesson as the source for the model. See Advanced Time Series Predictions (Intermediate Data Mining Tutorial).

     Data source view: `AllRegions`

   - Choose the following columns for the series key and time key:

     Key time: ReportingDate

     Key: Region

   - Choose the following columns for `Input` and `Predict`:

     SumQty

     SumAmt

     AvgAmt

     AvgQty

   - For **Mining structure name**, type: `All Regions`

   - For **Mining model name**, type: `All Regions`

3. Process the new structure and the new model.

To build the prediction query and map the replacement data

1. If the model is not already open, double-click the AllRegions structure, and in Data Mining Designer, click the **Mining Model Prediction** tab.

2. In the **Mining Model** pane, the model AllRegions should already be selected. If it is not selected, click **Select Model**, and then select the model, AllRegions.

3. In the **Select Input Table(s)** pane, click **Select Case Table**.

4. In the **Select Table** dialog box, change the data source to T1000 Pacific Region, and then click **OK**.

5. Right-click the join line between the mining model and the input data and select **Modify Connections**. Map the data in the data source view to the model as follows:

   a. Verify that the ReportingDate column in the mining model is mapped to the ReportingDate column in the input data.

   b. In the **Modify Mapping** dialog box, in the row for the model column AvgQty, click under **Table Column** and then select T1000 Pacific.Quantity. Click **OK**.

      This step maps the column you created in the model for predicting average quantity to the actual data from the T1000 series for sales quantity.

   c. Do not map the column Region in the model to any input column.

Because the model aggregated the data across all series, there is no match for the series values such as T1000 Pacific, and an error is raised when the prediction query runs.

6. Now you will build the prediction query.

   First, add a column to the results that outputs the AllRegions label from the model together with the predictions. This way you know that the results were based on the general model.

   a. In the grid, click the first empty row, under **Source**, and then select AllRegions mining model.

   b. For **Field**, select Region.

   c. For **Alias**, type **Model Used**.

7. Next, add another label to the results, so that you can see which series the prediction is for.

   a. Click an empty row, and under **Source**, select **Custom Expression**.

   b. In the **Alias** column, type **ModelRegion**.

   c. In the **Criteria/Argument** column, type `'T1000 Pacific'`.

8. Now you will set up the cross-prediction function.

   a. Click an empty row, and under **Source**, select **Prediction Function**.

   b. In the **Field** column, select **PredictTimeSeries**.

   c. For **Alias**, type **Predicted Values**.

   d. Drag the field AvgQty from the **Mining Model** pane into the **Criteria/Argument** column by using the drag and drop operation.

   e. In the **Criteria/Argument** column, after the field name, type the following text: `,5, REPLACE_MODEL_CASES`

      The complete text of the **Criteria/Argument** text box should be as follows: `[AllRegions].[AvgQty],5,REPLACE_MODEL_CASES`

9. Click **Results**.

Creating the Cross-Prediction Query in DMX

You might have noticed a problem with cross-prediction: namely, that to apply the general model to a different data series, such as the T1000 product model in the North America region, you must create a different query for each series, so that you can map the each set of inputs to the model.

However, rather than building the query in the designer, you can switch to DMX view and edit the DMX statement that you created. For example, the following DMX statement represents the query that you just built:

```
SELECT
      ([All Regions].[Region]) as [Model Used],
      ('T-1000 Pacific') as [ModelRegion],
      (PredictTimeSeries([All Regions].[Avg Qty],5, REPLACE_MODEL_CASES)) as [Predicted Quantity]
     FROM [All Regions]
PREDICTION JOIN
    OPENQUERY([Adventure Works DW2003R2], 'SELECT [ReportingDate] FROM
      (
       SELECT  ReportingDate, ModelRegion, Quantity, Amount
       FROM dbo.vTimeSeries
       WHERE (ModelRegion = N''T1000 Pacific'')
       ) as [T1000 Pacific]    ')
    AS t
ON
[All Regions].[Reporting Date] = t.[ReportingDate]
AND
[All Regions].[Avg Qty] = t.[Quantity]
```

To apply this to a different model, you simply edit the query statement to replace the filter condition and to update the labels associated with each result.

For example, if you change the filter conditions and column labels by replacing 'Pacific' with 'North America', you will get predictions for the T1000 product in North America, based on the patterns in the general model.

Next Task in Lesson

Comparing Predictions for Forecasting Models (Intermediate Data Mining Tutorial)

See Also

Time Series Model Query Examples
PredictTimeSeries (DMX)

Comparing Predictions for Forecasting Models (Intermediate Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

In the previous steps of this tutorial, you created multiple time series models:

- Predictions for each combination of region and model, based only on data for the individual model and region.
- Predictions for each region, based on updated data.
- Predictions for all models on a worldwide basis, based on aggregated data.
- Predictions for the M200 model in the North America region, based on the aggregated model.

To summarize the features for time series predictions, you will review the changes to see how the use of the options to extend or replace data affected forecasting results.

EXTEND_MODEL_CASES

REPLACE_MODEL_CASES

Comparing the Original Results with Results after Adding Data

Let's look at the data for just the M200 product line in the Pacific region, to see how updating the model with new data affects the results. Remember that the original data series ended in June 2004, and we obtained new data for July, August, and September.

- The first column shows the new data that was added.
- The second column shows the forecast for July and later based on the original data series.
- The third column shows the forecast based on the extended data.

M200 PACIFIC	UPDATED REAL SALES DATA	FORECAST BEFORE DATA WAS ADDED	EXTENDED PREDICTION
7-25-2008	**65**	32	**65**
8-25-2008	**54**	37	**54**
9-25-2008	**61**	32	**61**
10-25-2008	No data	36	32
11-25-2008	No data	31	41
12-25-2008	No data	34	32

You will note that the forecasts using the extended data (shown here in bold) repeat the real data points exactly. The repetition is by design. As long as there are real data points to use, the prediction query will return the actual values, and output new prediction values only after the new actual data points have been used up.

In general, the algorithm weights the changes in the new data more strongly than data from the beginning of the model data. However, in this case, the new sales figures represent an increase of only 20-30 percent over the previous period, so there only was a slight uptick in projected sales, after which the sales projections drop again, more in line with the trend in the months before the new data.

Comparing the Original and Cross-Prediction Results

Remember that the original mining model revealed big differences between regions and between product lines. For example, sales for the M200 model were very strong, while sales for the T1000 model were fairly low across all regions. Moreover, some series didn't have much data. Series were ragged, meaning they didn't have the same starting point.

So how did the predictions change when you made your projections based on the general model, which was based on world-wide sales, rather than the original data sets? To assure yourself that you have not lost any information or skewed the predictions, you can save the results to a table, join the table of predictions to the table of historical data, and then graph the two sets of historical data and predictions.

The following diagram is based on just one product line, the M200. The graph compares the predictions from the initial mining model against the predictions using the aggregated mining model.

From this diagram, you can see that the aggregated mining model preserves the overall range and trends in values while minimizing the fluctuations in the individual data series.

Conclusion

You have learned how to create and to customize a time series model that can be used for forecasting.

You have learned to update your time series models without having to reprocess them, by adding new data and creating predictions using the parameter, EXTEND_MODEL_CASES.

You have learned to create models that can be used for cross-prediction, by using the REPLACE_MODEL_CASES parameter and applying the model to a different data series.

See Also

Intermediate Data Mining Tutorial (Analysis Services - Data Mining)
Time Series Model Query Examples

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

The marketing department of Adventure Works Cycles wants to improve the company Web site to promote cross-selling. As part of the site update, they would like the ability to predict products that a customer might want to purchase, based on the other products that are already in the customer's online shopping basket. The marketing department also wants to understand customer purchasing behavior better, so that they can design the Web site so that the items that tend to be purchased together appear together. They have learned that data mining is especially useful for this kind of *market basket analysis* and have asked you to develop a data mining model.

After you complete the tasks in this lesson, you will have a mining model that shows groups of items from historical customer transactions. Additionally, you can use the mining model to predict additional items that a customer may want to purchase.

To complete the tasks in this lesson, you will use the solution and data source that you created in the first lesson of the Intermediate Data Mining Tutorial (Analysis Services - Data Mining). You will modify this solution by adding a data source view that contains tables about the customer, including a nested table of customer purchases. You will then build a mining model that uses the Microsoft Association Rules algorithm, which is suited to market basket scenarios.

This lesson contains the following topics:

- Adding a Data Source View with Nested Tables (Intermediate Data Mining Tutorial)
- Creating a Market Basket Structure and Model (Intermediate Data Mining Tutorial)
- Modifying and Processing the Market Basket Model (Intermediate Data Mining Tutorial)
- Exploring the Market Basket Models (Intermediate Data Mining Tutorial)
- Filtering a Nested Table in a Mining Model (Intermediate Data Mining Tutorial)
- Predicting Associations (Intermediate Data Mining Tutorial)

Next Task in Lesson

Adding a Data Source View with Nested Tables (Intermediate Data Mining Tutorial)

All Lessons

Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial)

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

Lesson 3: Market Basket Scenario (Intermediate Data Mining Tutorial)

Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)

See Also

Basic Data Mining Tutorial
Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)
Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

Adding a Data Source View with Nested Tables (Intermediate Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

To create a market basket model, you must use a data source view that supports associative data. This data source view will also be used for the sequence clustering scenario.

This data source view is different from others that you may have worked with because it contains a *nested table*. A *nested table* is a table that contains multiple rows of information about a single row in the case table. For example, if your model analyzes the purchasing behavior of customers, you would typically use a table that has a unique row for each customer as the case table. However, each customer might make multiple purchases, and you might want to analyze the sequence of purchases, or products that are frequently purchased together. To logically represent these purchases in your model, you add another table to the data source view that lists the purchases for each customer.

This nested purchases table is related to the customer table by a many-to-one relationship. The nested table might contain many rows for each customer, each row containing a single product that was purchased, perhaps with additional information about the order that the purchases were made, the price at the time of the order, or any promotions that applied. You can use the information in the nested table as inputs to the model, or as the predictable attribute.

In this lesson, you do the following tasks:

- You add a data source view to the Adventure Works DW Multidimensional 2012 data source.
- You add the case and nested tables to this view.
- You specify the many-to-one relationship between the case and nested table.

> **NOTE**
>
> . It is important that you follow the described procedure exactly, to correctly specify the relationship between the case table and the nested table and to avoid errors when you process the model.

- You define how the columns of data are used in the model.

For more information about working with case and nested tables, and how to choose a nested table key, see Nested Tables (Analysis Services - Data Mining).

To add a data source view

1. In Solution Explorer, right-click **Data Source Views**, and then select **New Data Source View**.

   The Data Source View Wizard opens.

2. On the **Welcome to the Data Source View Wizard** page, click **Next**.

3. On the **Select a Data Source** page, under **Relational data sources**, select the Adventure Works DW Multidimensional 2012 data source that you created in the Basic Data Mining Tutorial. Click **Next**.

4. On the **Select Tables and Views** page, select the following tables, and then click the right arrow to include them in the new data source view:

   - `vAssocSeqOrders`

- `vAssocSeqLineItems`

5. Click **Next**.

6. On the **Completing the Wizard** page, by default the data source view is named Adventure Works DW Multidimensional 2012. Change the name to `Orders`, and then click **Finish**.

   Data Source View Designer opens and the `Orders` data source view appears.

To create a relationship between tables

1. In Data Source View Designer, position the two tables so that the tables are aligned horizontally, with the vAssocSeqLineItems table on the left side and the vAssocSeqOrders table on the right side.

2. Select the **OrderNumber** column in the vAssocSeqLineItems table.

3. Drag the column to the vAssocSeqOrders table, and put it on the **OrderNumber** column.

   > **IMPORTANT**
   >
   > Make sure to drag the **OrderNumber** column from the vAssocSeqLineItems nested table, which represents the many side of the join, to the vAssocSeqOrders case table, which represents the one side of the join.

   A new *many-to-one relationship* now exists between the vAssocSeqLineItems and vAssocSeqOrders tables. If you have joined the tables correctly, the data source view should appear as follows:

Next Task in Lesson

Creating a Market Basket Structure and Model (Intermediate Data Mining Tutorial)

See Also

Intermediate Data Mining Tutorial (Analysis Services - Data Mining)
Mining Structures (Analysis Services - Data Mining)
Mining Models (Analysis Services - Data Mining)

Creating a Market Basket Structure and Model (Intermediate Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

Now that you have created a data source view, you will use the Data Mining Wizard to create a new mining structure. In this task, you will create a mining structure and a mining model that is based on the Microsoft Association algorithm.

> **NOTE**
>
> If you encounter an error stating that vAssocSeqLineItems cannot be used as a nested table, return to the previous task in the lesson, and be sure to create the many-to-one join by dragging from the vAssocSeqLineItems table (the many side) to the vAssocSeqOrders table (the one side). You can also edit the relationship between the tables by right-clicking the join line.

To create an association mining structure

1. In Solution Explorer in SQL Server Data Tools (SSDT), right-click **Mining Structures** and select **New Mining Structure** to open the Data Mining Wizard.

2. On the **Welcome to the Data Mining Wizard** page, click **Next**.

3. On the **Select the Definition Method** page, verify that **From existing relational database or data warehouse** is selected, and then click **Next**.

4. On the **Create the Data Mining Structure** page, under **Which data mining technique do you want to use?**, select **Microsoft Association Rules** from the list, and then click **Next**. The **Select Data Source View** page appears.

5. Select **Orders**under **Available data source views**, and then click **Next**.

6. On the **Specify Table Types** page, in the row for the vAssocSeqLineItems table, select the **Nested** check box, and in the row for the nested table vAssocSeqOrders, select the **Case** check box. Click **Next**.

7. On the **Specify the Training Data** page, clear any boxes that might be checked. Set the key for the case table, vAssocSeqOrders, by selecting the **Key** check box next to OrderNumber.

   Because the purpose of the market basket analysis is to determine which products are included in a single transaction, you do not have to use the **CustomerKey** field.

8. Set the key for the nested table, vAssocSeqLineItems, by selecting the **Key** check box next to Model. The **Input** check box is also automatically selected when you do this. Select the **Predictable** check box for `Model` as well.

   In a market basket model, you do not care about the sequence of products in the shopping basket, and therefore you should not include **LineNumber** as a key for the nested table. You would use **LineNumber** as a key only in a model where the sequence is important. You will create a model that uses the Microsoft Sequence Clustering algorithm in Lesson 4.

9. Select the check box to the left of IncomeGroup and Region,but do not make any other selections. Checking the leftmost column adds the columns to the structure for later reference, but the columns will not be used in the model. Your selections should look like the following:

10. Click **Next**.

11. On the **Specify Columns' Content and Data Type**page, review the selections, which should be as shown in the following table, and then click **Next**.

COLUMNS	CONTENT TYPE	DATA TYPE
IncomeGroup	Discrete	Text
Order Number	Key	Text
Region	Discrete	Text
vAssocSeqLineItems		
Model	Key	Text

12. On the **Create testing set** page, the default value for the option **Percentage of data for testing** is 30 percent. Change this to **0**. Click **Next**.

> **NOTE**
> Analysis Services provides different charts for measuring model accuracy. However, some accuracy chart types, such as the lift chart and cross-validation report, are designed for classification and estimation. They are not supported for associative prediction.

13. On the **Completing the Wizard** page, in **Mining structure name**, type `Association`.

14. In **Mining model name**, type `Association`.

15. Select the option **Allow drill through**, and then click **Finish**.

    Data Mining Designer opens to display the `Association` mining structure that you just created.

Next Task in Lesson

Modifying and Processing the Market Basket Model (Intermediate Data Mining Tutorial)

See Also

Microsoft Association Algorithm
Content Types (Data Mining)

Modifying and Processing the Market Basket Model (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Before you process the association mining model that you created, you must change the default values of two of the parameters: *Support* and *Probability*.

- *Support* defines the percentage of cases in which a rule must exist before it is considered valid. You will specify that a rule must be found in at least 1 percent of cases.
- *Probability* defines how likely an association must be before it is considered valid. You will consider any association with a probability of at least 10 percent.

For more information about the effects of increasing or decreasing support and probability, see Microsoft Association Algorithm Technical Reference.

After you have defined the structure and parameters for the **Association** mining model, you will process the model.

To adjust the parameters of the Association model

1. Open the **Mining Models** tab of Data Mining Designer.
2. Right-click the **Association** column in the grid in the designer and select **Set Algorithm Parameters to open the Algorithm Parameters** dialog box.
3. In the **Value** column of the **Algorithm Parameters** dialog box, set the following parameters:

   MINIMUM_PROBABILITY = 0.1

   MINIMUM_SUPPORT = 0.01
4. Click **OK**.

To process the mining model

1. On the **Mining Model** menu of SQL Server Data Tools (SSDT), select **Process Mining Structure and All Models.**
2. At the warning asking whether you want to build and deploy the project, click **Yes**.

   The **Process Mining Structure - Association** dialog box opens.
3. Click **Run**.

   The **Process Progress** dialog box opens to display information about model processing. Processing of the new structure and model might take some time.
4. After processing is complete, click **Close** to exit the **Process Progress** dialog box.
5. Click **Close** again to exit the **Process Mining Structure - Association** dialog box.

Next Task in Lesson

Exploring the Market Basket Models (Intermediate Data Mining Tutorial)

See Also

Processing Requirements and Considerations (Data Mining)

Exploring the Market Basket Models (Intermediate Data Mining Tutorial)

2/11/2019 • 9 minutes to read • Edit Online

Now that you have built the `Association` model, you can explore it by using the Microsoft Association Viewer in the **Mining Model Viewer** tab of Data Mining Designer. This tutorial walks you through using the viewer to explore relationships between items. The viewer helps you see at a glance which products tend to appear together, and get a general idea of the emerging patterns.

The Microsoft Association Viewer contains three tabs: **Rules**, **Itemsets**, and **Dependency Network**. Because each tab reveals a slightly different view of the data, when you are exploring a model, you will typically switch back and forth between the different panes several times as you pursue insights.

- Dependency Network tab
- Itemsets tab
- Rules tab
- Generic Content View

For this tutorial, you will start on the **Dependency Network** tab, and then use the **Rules** tab and **Itemsets** tab to deepen your understanding of the relationships revealed in the viewer. You will also use the **Microsoft Generic Content Tree Viewer** to retrieve detailed statistics for individual rules or itemsets.

Dependency Network Tab

With the **Dependency Network** tab, you can investigate the interaction of the different items in the model. Each node in the viewer represents an item, while the lines between them represent rules. By selecting a node, you can see which other nodes predict the selected item, or which items the current item predicts. In some cases, there is a two-way association between items, meaning that they often appear in the same transaction. You can refer to the color legend at the bottom of the tab to determine the direction of the association.

A line connecting two items means that these items are likely to appear in a transaction together. In other words, customers are likely to buy these items together. The slider is associated with the probability of the rule. Move the slider up or down to filter out weak associations, meaning rules with low probability.

The dependency network graph shows pairwise rules, which can be represented logically as A->B, meaning if Product A is purchased, then Product B is likely. The graph cannot show rules of the type AB->C. If you move the slider to show all rules but still do not see any lines in the graph, it means that there were no pairwise rules that met the criteria of the algorithm parameters.

You can also find nodes by name, by typing the first letters of the attribute name. For more information, see Find Node Dialog Box (Mining Model Viewer).

To open the Association mode in the Microsoft Assocaition Rules Viewer

1. In **Solution Explorer**, double-click the Association structure.
2. In Data Mining Designer, click the **Mining Model Viewer** tab.
3. Select Association from the list of mining models in the **Mining Model** dropdown list.

To navigate the dependency graph and locate specific nodes

1. In the **Mining Model Viewer** tab, click the **Dependency Network** tab.

2. Click **Zoom In** several times, until you can easily view the labels for each node.

   By default, the graph displays with all nodes visible. In a complex model, there may be many nodes, making each node quite small.

3. Click the **+** sign in the lower right-hand corner of the viewer and hold down the mouse button to pan around the graph.

4. On the left side of the viewer, drag the slider down, moving it from **All Links** (the default) to the bottom of the slider control.

5. The viewer updates the graph to now show only the strongest association, between the Touring Tire and Touring Tire Tube items.

6. Click the node labeled **Touring Tire Tube = Existing**.

   The graph is updated to highlight only items that are strongly related to this item. Note the direction of the arrow between the two items.

7. On the left side of the viewer, drag the slider up again, moving it from the bottom to around the middle.

   Note the changes in the arrow that connects the two items.

8. Select **Show attribute name only** from the dropdown list at the top of the Dependency Network pane.

   The text labels in the graph are updated to show only the model name.

Back to Top

Itemsets Tab

Next, you will learn more about the rules and itemsets generated by the model for the Touring Tire and Touring Tire Tube products. The **Itemsets** tab displays three important pieces of information that relate to the itemsets that the Microsoft Association algorithm discovers:

- **Support:** The number of transactions in which the itemset occurs.
- **Size:** The number of items in the itemset.
- **Items:** A list of the items included in each itemset.

Depending on how the algorithm parameters are set, the algorithm might generate many itemsets. Each itemset that is returned in the viewer represents transactions in which the item was sold. By using the controls at the top of the **Itemsets** tab, you can filter the viewer to show only the itemsets that contain a specified minimum support and itemset size.

If you are working with a different mining model and no itemsets are listed, it is because no itemsets met the criteria of the algorithm parameters. In such a scenario, you can change the algorithm parameters to allow itemsets that have lower support.

To filter the itemsets that are shown in the viewer by name

1. Click the **Itemsets** tab of the viewer.

2. In the **Filter Itemset** box, type `Touring Tire`, and then click outside the box.

   The filter returns all items that contain this string.

3. In the **Show** list, select **Show attribute name only**.

4. Select the **Show long name** check box.

The list of itemsets is updated to show only the itemsets that contain the string Touring Tire. The long name of the itemset includes the name of the table that contains the attribute and value for each item.

5. Clear the **Show long name** check box.

   The list of itemsets is updated to show only the short name.

The values in the **Support** column indicate the number of transactions for each itemset. A transaction for an itemset means a purchase that included all the items in the itemset.

By default, the viewer lists the itemsets in descending order by support. You can click on the column headers to sort by a different column, such as the itemset size or name. If you are interested in learning more about the individual transactions that are included in an itemset, you can drill through from the itemsets to the individual cases. The structure columns in the drillthrough results are the customer's income level and customer ID, which were not used in the model.

To view details for an itemset

1. In the list of itemsets, click the **Itemset** column heading to sort by name.
2. Locate the item, `Touring Tire` (with no second item).
3. Right-click the item, `Touring Tire`, select **Drill Through**, and then select **Model and Structure Columns**.

   The **Drill Through** dialog box displays the individual transactions used as support for this itemset.
4. Expand the nested table, vAssocSeqLineItems, to view the actual list of purchases in the transaction.

To filter itemsets by support or size

1. Clear any text that might be in the **Filter Itemset** box. You cannot use a text filter together with a numeric filter.
2. In the **Minimum support** box, type 100, and then click the background of the viewer.

   The list of itemsets is updated to show only itemsets with support of at least 100.

Back to Top

Rules Tab

The **Rules** tab displays the following information that is related to the rules that the algorithm finds.

- **Probability:** The *likelihood* of a rule, defined as the probability of the right-hand item given the left-hand side item.
- **Importance:** A measure of the usefulness of a rule. A greater value means a better rule.

  Importance is provided to help you gauge the usefulness of a rule, because probability alone can be misleading. For example, if every transaction contains a water bottle--perhaps the water bottle is added to each customer's cart automatically as part of a promotion--the model would create a rule predicting that water bottle has a probability of 1. Based on probability alone, this rule is very accurate, but it does not provide useful information.
- **Rule:** The definition of the rule. For a market basket model, a rule describes a specific combination of items.

Each rule can be used to predict the presence of an item in a transaction based on the presence of other items. Just like in the **Itemsets** tab, you can filter the rules so that only the most interesting rules are shown. If you are working with a mining model that does not have any rules, you might want to change the algorithm parameters to lower the probability threshold for rules.

To see only rules that include the Mountain-200 bicycle

1. In the **Mining Model Viewer** tab, click the **Rules** tab.
2. In the **Filter Rule** box, enter `Mountain-200`.

   Clear the **Show long name** check box.
3. From the **Show** list, select **Show attribute name only**.

   The viewer will then display only the rules that contain the words "`Mountain-200`". The probability of the rule tells you how likely it is that when someone buys a `Mountain-200` bicycle, that person will also buy the other listed product.

The rules are ordered by probability in descending order, but you can click the column headings to change the sort order. If you are interested in finding out more details about a particular rule, you can use drillthrough to view the supporting cases.

To view cases that support a particular rule

1. In the **Rules** tab, right-click the rule that you want to view.
2. Select **Drill Through**, and then select **Model Columns Only**, or **Model and Structure Columns**.

   The **Drill Through** dialog box provides a summary of the rule at the top of the pane, and a list of all cases that were used as supporting data for the rule.

Back to Top

Generic Content Tree Viewer

This viewer can be used for all models, regardless of the algorithm or model type. The **Microsoft Generic Content Tree Viewer** is available from the **Viewer** drop-down list.

A content tree is a representation of a mining model as a series of nodes, where each node represents learned knowledge about some subset of the data. The node can contain a pattern, a set of rules, a cluster, or the definition of a range of dates that share some characteristics. The exact content of the node differs depending on the algorithm and the type of the predictable attribute, but the general representation of the content is the same. You can expand each node to see increasing levels of detail, and copy the content of any node to the Clipboard.

To view details about the rule by using the content viewer

1. In the **Mining Model Viewer** tab, select **Microsoft Generic Content Tree Viewer** from the **Viewer** list.
2. In the Node Caption pane, scroll to the bottom of the list, and click the last node.

   The viewer shows itemsets first and rules next, but does not group them. The easiest way to find a specific node is to create a content query. For more information, see Association Model Query Examples.
3. In the Node Details pane, review the value for NODE_TYPE and NODE_DESCRIPTION.

   A node type of 8 is a rule, and a node type of 7 is an itemset. For a rule, the value of NODE_DESCRIPTION tells you the conditions that make up the rule. For an itemset, the value of NODE_DESCRIPTION tells you the items included in the itemset.

You can also create a content query to obtain detailed statistics about the rules. For more information about mining model content and how to interpret it, see Mining Model Content for Association Models (Analysis Services - Data Mining).

Back to Top

Next Task in Lesson

Filtering a Nested Table in a Mining Model (Intermediate Data Mining Tutorial)

See Also

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)
Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)
Microsoft Association Algorithm
Microsoft Association Algorithm Technical Reference

Filtering a Nested Table in a Mining Model (Intermediate Data Mining Tutorial)

2/11/2019 • 4 minutes to read • Edit Online

After you have created and explored the model, you decide that you want to focus on a subset of the customer data. For example, you might want to analyze only the baskets that contain a specific item, or to analyze the demographics of customers who have not purchased anything in a certain period.

Analysis Services provides the ability to filter the data that is used in a mining model. This feature is useful because you do not need to set up a new data source view to use different data. In the Basic Data Mining Tutorial, you learned how to filter data from a flat table by applying conditions to the case table. In this task, you create a filter that applies to a nested table.

Filters on Nested vs. Case Tables

If your data source view contains a case table and a nested table, like the data source view used in the Association model, you can filter on values from the case table, the presence or absence of a value in the nested table, or some combination of both.

In this task, you will first make a copy of the Association model and then add the IncomeGroup and Region attributes to the new related model, so that you can filter on those attributes in the case table.

To create and modify a copy of the Association model

1. In the **Mining Models** tab of SQL Server Data Tools (SSDT), right-click the `Association` model, and select **New Mining Model**.
2. For **Model Name**, type `Association Filtered`. For **Algorithm Name**, select **Microsoft Association Rules**. Click **OK**.
3. In the column for the Association Filtered model, click the IncomeGroup row and change the value from **Ignore** to **Input**.

Next, you will create a filter on the case table in the new association model. The filter will pass to the model only the customers in the target region or with the target income level. Then, you will add a second set of filter conditions to specify that the model uses only customers whose shopping baskets contained at least one item.

To add a filter to a mining model

1. In the **Mining Models** tab, right-click the model Association Filtered, and select **Set Model Filter**.
2. In the **Model Filter** dialog box, click the top row in the grid, in the **Mining Structure Column** text box.
3. In the **Mining Structure Column** text box, select IncomeGroup.

   The icon at the left side of the text box changes to indicate that the selected item is a column.
4. Click the **Operator** text box and select the **=** operator from the list.
5. Click the **Value** text box, and type `High` in the box.
6. Click the next row in the grid.
7. Click the **AND/OR** text box in the next row of the grid and select **OR**.
8. In the **Mining Structure Column** text box, select IncomeGroup. In the **Value** text box, type `Moderate`.

The filter condition that you created is automatically added to the **Expression** text box, and should appears as follows:

```
[IncomeGroup] = 'High' OR [IncomeGroup] = 'Moderate'
```

9. Click the next row in the grid, leaving the operator as the default, **AND**.
10. For **Operator**, leave the default value, **Contains**. Click the **Value** text box.
11. In the **Filter** dialog box, in the first row under **Mining Structure Column**, select `Model`.
12. For **Operator**, select **IS NOT NULL**. Leave the **Value** text box blank. Click **OK**.

    The filter condition in the **Expression** text box of the **Model Filter** dialog box is automatically updated to include the new condition on the nested table. The completed expression is as follows:

    ```
    [IncomeGroup] = 'High' OR [IncomeGroup] = 'Moderate' AND EXISTS SELECT * FROM [vAssocSeqLineItems] WHERE
    [Model] <> NULL).
    ```

13. Click **OK**. ``

To enable drillthrough and to process the filtered model

1. In the **Mining Models** tab, right-click the `Association Filtered` model, and select **Properties**.
2. Change the **AllowDrillThrough** property to **True**.
3. Right-click the `Association Filtered` mining model, and select **Process Model**.
4. Click **Yes** in the error message to deploy the new model to the Analysis Services database.
5. In the **Process Mining Structure** dialog box, click **Run**.
6. When processing is complete click **Close** to exit the **Process Progress** dialog box, and click **Close** again to exit the **Process Mining Structure** dialog box.

You can verify by using the Microsoft Generic Content Tree viewer and looking at the value for NODE_SUPPORT that the filtered model contains fewer cases than the original model.

Remarks

The nested table filter that you just created checks only for the presence of at least one row in the nested table; however, you can also create filter conditions that check for the presence of specific products. For example, you could create the following filter:

```
[IncomeGroup] = 'High' AND
 EXISTS (SELECT * FROM [<nested table name>] WHERE [Model] = 'Water Bottle' )
```

This statement means that you are restricting the customers from the case table to only those who have purchased a water bottle. However, because the number of nested table attributes is potentially unlimited, Analysis Services does not supply a list of possible values from which to select. Instead, you must type the exact value.

You can click **Edit Query** to manually change the filter expression. However, if you change any part of a filter expression manually, the grid will be disabled and thereafter you must work with the filter expression in text edit mode only. To restore grid editing mode, you must clear the filter expression and start over.

> **WARNING**
>
> You cannot use the LIKE operator in a nested table filter.

Next Task in Lesson

Predicting Associations (Intermediate Data Mining Tutorial)

See Also

Model Filter Syntax and Examples (Analysis Services - Data Mining)
Filters for Mining Models (Analysis Services - Data Mining)

Predicting Associations (Intermediate Data Mining Tutorial)

2/11/2019 • 9 minutes to read • Edit Online

After the models have been processed, you can use the information about associations stored in the model to create predictions. In the final task of this lesson, you learn how to build prediction queries against the association models that you created. This lesson assumes that you are familiar with how to use the Prediction Query Builder and want to learn how to build prediction queries against association models. For more information how to use Prediction Query Builder, see Data Mining Query Interfaces.

Creating a Singleton Prediction Query

Prediction queries on an association model can be very useful:

- Recommend items to a customer, based on prior or related purchases
- Find related events.
- Identify relationships in or across sets of transactions.

To build a prediction query, you first select the association model you want to use, and then you specify the input data. Inputs can come from an external data source, such as a list of values, or you can build a singleton query and provide values as you go.

For this scenario, you will first create some singleton prediction queries, to get an idea of how prediction works. Then you will create a query for batch predictions that you could use for making recommendations based on a customer's current purchases.

To create a prediction query on an association model

1. Click the **Mining Model Prediction** tab of Data Mining Designer.
2. In the **Mining Model** pane, click **Select Model**. (You can skip this step and the next step if the correct model is already selected.)
3. In the **Select Mining Model** dialog box, expand the node that represents the mining structure **Association**, and select the model **Association**. Click **OK**.

   For now, you can ignore the input pane.
4. In the grid, click the empty cell under **Source** and select **Prediction Function.** In the cell under **Field**, select `PredictAssociation`.

   You can also use the **Predict** function to predict associations. If you do, be sure to choose the version of the **Predict** function that takes a table column as argument.
5. In the **Mining Model** pane, select the nested table `vAssocSeqLineItems`, and drag it into the grid, to the **Criteria/Argument** box for the `PredictAssociation` function.

   Dragging and dropping table and column names lets you build complex statements without syntax errors. However, it replaces the current contents of the cell, which include other optional arguments for the `PredictAssociation` function. To view the other arguments, you can temporarily add a second instance of the function to the grid for reference.
6. Click the **Criteria/Argument** box and type the following text after the table name: `,3`

The complete text in the **Criteria/Argument** box should be as follows:

```
[Association].[v Assoc Seq Line Items],3
```

7. Click the **Results** button in the upper corner of the Prediction Query Builder.

The expected results contain a single column with the heading **Expression**. The **Expression** column contains a nested table with a single column and the following three rows. Because you did not specify an input value, these predictions represent the most likely product associations for the model as a whole.

MODEL
Women's Mountain Shorts
Water Bottle
Touring-3000

Next, you will use the **Singleton Query Input** pane to specify a product as input to the query, and view the products that are most likely associated with that item.

To create a singleton prediction query with nested table inputs

1. Click the **Design** button in the corner of the Prediction Query Builder to switch back to the query building grid.
2. On the **Mining Model** menu, select **Singleton Query**.
3. In the **Mining Model** dialog box, select the **Association** model.
4. In the grid, click the empty cell under **Source** and select **Prediction Function.** In the cell under **Field**, select `PredictAssociation`.
5. In the **Mining Model** pane, select the nested table `vAssocSeqLineItems`, and drag it into the grid, to the **Criteria/Argument** box for the `PredictAssociation` function. Type `,3` after the nested table name just as in the previous procedure.
6. In the **Singleton Query Input** dialog box, click the **Value** box next to **vAssoc Seq Line Items**, and then click the **(...)** button.
7. In the **Nested Table Input** dialog box, select `Touring Tire` in the **Key column** pane, and then click **Add**.
8. Click the **Results** button.

The results now show the predictions for products that are most likely associated with the Touring Tire.

MODEL
Touring Tire Tube
Sport-100
Water Bottle

However, you already know from exploring the model that the Touring Tire Tube is frequently purchased with the Touring Tire; you are more interested in knowing what products you can recommend to customers who purchase these items together. You will change the query so that it predicts related products based on two items in the basket. You will also modify the query to add the probability for each predicted product.

To add inputs and probabilities to the singleton prediction query

1. Click the **Design** button in the corner of the Prediction Query Builder to switch back to the query building grid.

2. In the **Singleton Query Input** dialog box, click the **Value** box next to **vAssoc Seq Line Items**, and then click the **(...)** button.

3. In the **Key column** pane, select `Touring Tire`, and then click **Add**.

4. In the grid, click the empty cell under **Source** and select **Prediction Function.** In the cell under **Field**, select `PredictAssociation`.

5. In the **Mining Model** pane, select the nested table `vAssocSeqLineItems`, and drag it into the grid, to the **Criteria/Argument** box for the `PredictAssociation` function. Type `,3` after the nested table name just as in the previous procedure.

6. In the **Nested Table Input** dialog box, select `Touring Tire Tube` in the **Key column** pane, and then click **Add**.

7. In the grid, in the row for the `PredictAssociation` function, click the **Criteria/Argument** box, and change the arguments to add the argument, INCLUDE_STATISTICS.

   The complete text in the **Criteria/Argument** box should be as follows:

   ```
   [Association].[v Assoc Seq Line Items], INCLUDE_STATISTICS, 3
   ```

8. Click the **Results** button.

The results in the nested table now change to show the predictions, together with support and probability. For more information about how to interpret these values, see Mining Model Content for Association Models (Analysis Services - Data Mining).

MODEL	$SUPPORT	$PROBABILITY	$ADJUSTEDPROBABILITY
Sport-100	4334	0.291...	0.252...
Water Bottle	2866	0.192...	0.175...
Patch Kit	2113	0.142...	0.132

Working with Results

When there are many nested tables in the results, you might want to flatten the results for easier viewing. To do this, you can manually modify the query and add the `FLATTENED` keyword.

To flatten nested rowsets in a prediction query

1. Click the **SQL** button in the corner of the Prediction Query Builder.

   The grid changes to an open pane where you can view and modify the DMX statement that was created by the Prediction Query Builder.

2. After the `SELECT` keyword, type `FLATTENED`.

   The complete text of the query should be as follows:

```
SELECT FLATTENED
  PredictAssociation([Association].[v Assoc Seq Line Items],INCLUDE_STATISTICS,3)
FROM
  [Association]
NATURAL PREDICTION JOIN
(SELECT (SELECT 'Touring Tire' AS [Model]
  UNION SELECT 'Touring Tire Tube' AS [Model]) AS [v Assoc Seq Line Items]) AS t
```

3. Click the **Results** button in the upper corner of the Prediction Query Builder.

Note that after you have manually edited a query, you will not be able to switch back to Design view without losing the changes. If you wish to save the query, you can copy the DMX statement that you created manually to a text file. When you change back to Design view, the query is reverted to the last version that was valid in Design view.

Creating Multiple Predictions

Suppose you want to know the best predictions for individual customers, based on past purchases. You can use external data as input to the prediction query, such as tables containing the customer ID and the most recent product purchases. The requirements are that the data tables be already defined as an Analysis Services data source view; moreover, the input data must contain case and nested tables like those used in the model. They need not have the same names, but the structure must be similar. For the purpose of this tutorial, you will use the original tables on which the model was trained.

To change the input method for the prediction query

1. In the **Mining Model** menu, select **Singleton Query** again, to clear the check mark.
2. An error message appears warning that your singleton query will be lost. Click **Yes**.

   The name of the input dialog box changes to **Select Input Table(s)**.

Because you are interested in creating a prediction query that provides Customer ID and a list of products as input, you will add the customer table as the case table, and the purchases table as the nested table. Then you will add prediction functions to create recommendations.

To create a prediction query using nested table inputs

1. In the Mining Model pane, select the Association Filtered model.
2. In the **Select Input Table(s)** dialog box, click **Select Case Table**.
3. In the **Select Table** dialog box, for **Data Source**, select AdventureWorksDW2008. In the **Table/View Name** list, select vAssocSeqOrders, and then click **OK**.

   The table vAssocSeqOrders is added to the pane.
4. In the **Select Input Table(s)** dialog box, click **Select Nested Table**.
5. In the **Select Table** dialog box, for **Data Source**, select AdventureWorksDW2008. In the **Table/View name** list, select vAssocSeqLineItems, and then click **OK**.

   The table vAssocSeqLineItems is added to the pane.
6. In the **Specify Nested Join** dialog box, drag the OrderNumber field from the case table and drop it onto the OrderNumber field in the nested table.

   You can also click **Add Relationship** and create the relationship by selecting columns from a list.
7. In the **Specify Relationship** dialog box, verify that the OrderNumber fields are mapped correctly, and then click **OK**.

8. Click **OK** to close the **Specify Nested Join** dialog box.

   The case and nested tables are updated in the design pane to show the joins connecting the external data columns to the columns in the model. If the relationships are wrong, you can right-click the join line and select **Modify Connections** to edit the column mapping, or you can right-click the join line and select **Delete** to remove the relationship completely.

9. Add a new row to the grid. For **Source**, select **vAssocSeqOrders table**. For **Field**, select CustomerKey.

10. Add a new row to the grid. For **Source**, select **vAssocSeqOrders table**. For **Field**, select Region.

11. Add a new row to the grid. For **Source**, select **Prediction Function**, and for **Field**, select `PredictAssociation`.

12. Drag vAssocSeqLineItems, into the **Criteria/Argument** box of the `PredictAssociation` row. Click at the end of the **Criteria/Argument** box and then type the following text: `INCLUDE_STATISTICS,3`

    The complete text in the **Criteria/Argument** box should be:
    `[Association].[v Assoc Seq Line Items], INCLUDE_STATISTICS, 3`

13. Click the **Result** button to view the predictions for each customer.

You might try creating a similar prediction query on the multiple models, to see whether filtering changes the prediction results. For more information about creating predictions and other types of queries, see Association Model Query Examples.

See Also

Mining Model Content for Association Models (Analysis Services - Data Mining)
PredictAssociation (DMX)
Create a Prediction Query Using the Prediction Query Builder

Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

The marketing department of Adventure Works Cycles wants to understand how customers move through the Adventure Works Cycles Web site. The company suspects that there is a pattern to the order in which customers put products into their shopping baskets. They want to analyze the order of purchase sequences to learn how customers add related items to their baskets. They can then use this information to streamline the flow of the Web site so that it leads customers to purchase additional products.

After you complete the tasks in this lesson, you will have created a mining model that uses the Microsoft Sequence Clustering algorithm to predict the next item that customers will put into their shopping baskets. You will experiment with two versions of the model: one that analyzes only the order of products in the basket, and one that contains some additional customer demographics for clustering. Finally, you will use the models to create predictions that you can use to recommend products to customers.

To complete the tasks in the lesson, you will use the market basket mining structure that you created in Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial). This lesson contains the following tasks:

- Creating a Sequence Clustering Mining Model Structure (Intermediate Data Mining Tutorial)
- Processing the Sequence Clustering Model
- Exploring the Sequence Clustering Model (Intermediate Data Mining Tutorial)
- Creating a Related Sequence Clustering Model (Intermediate Data Mining Tutorial)
- Creating Predictions on a Sequence Clustering Model (Intermediate Data Mining Tutorial)

Next Task in Lesson

Creating a Sequence Clustering Mining Model Structure (Intermediate Data Mining Tutorial)

All Lessons

Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial)

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)

Lesson 4: Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)

See Also

Basic Data Mining Tutorial
Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

Creating a Sequence Clustering Mining Model Structure (Intermediate Data Mining Tutorial)

2/11/2019 • 3 minutes to read • Edit Online

The first step in creating a sequence clustering mining model is to use the Data Mining Wizard to create a new mining structure and a mining model based on the Microsoft Sequence Clustering algorithm.

You will use the same data source view that you used for the market basket analysis, but you will add a column that contains the `sequence` identifier. In this scenario, the sequence means the order in which the customer added items to the shopping basket.

You will also add some columns that are used in one of the models to group customers by demographics.

To create a sequence clustering structure and model

1. In Solution Explorer in SQL Server Data Tools (SSDT), right-click **Mining Structures** and select **New Mining Structure**.

2. On the **Welcome to the Data Mining Wizard** page, click **Next**.

3. On the **Select the Definition Method** page, verify that **From existing relational database or data warehouse** is selected, and then click **Next**.

4. On the **Create the Data Mining Structure** page, verify that the option **Create mining structure with a mining model** is selected. Next, click the dropdown list for the option, **Which data mining technique do you want to use?**, and select **Microsoft Sequence Clustering**. Click **Next**.

   The **Select Data Source View** page appears. Under **Available data source views**, select `Orders`.

   Orders is the same data source view that you used for the market basket scenario. If you have not created this data source view, see Adding a Data Source View with Nested Tables (Intermediate Data Mining Tutorial).

5. Click **Next**.

6. On the **Specify Table Types** page, select the **Case** check box next to the **vAssocSeqOrders** table, and select the **Nested** check box next to the **vAssocSeqLineItems** table. Click **Next**.

   > **NOTE**
   >
   > If an error occurs when you select the **Case** or **Nested** check boxes, it may be that the join in the data source view is not correct. The nested table, **vAssocSeqLineItems**, must be connected to the case table, **vAssocSeqOrders,** by a many-to-one join. You can edit the relationship by right-clicking on the join line and then reversing the direction of the join. For more information, see Create or Edit Relationship Dialog Box (Analysis Services - Multidimensional Data).

7. On the **Specify the Training Data** page, choose the columns for use in the model by selecting a check box as follows:

   - **IncomeGroup** Select the **Input** check box.

     This column contains interesting information about the customers that you can use for clustering. You will use it in the first model and then ignore it in the second model.

   - **OrderNumber** Select the `Key` check box.

This field will be used as the identifier for the case table, or `Key`. In general, you should never use the key field of the case table as an input, because the key contains unique values that are not useful for clustering.

- **Region** Select the **Input** check box.

  This column contains interesting information about the customers that you can use for clustering. You will use it in the first model and then ignore it in the second model.

- **LineNumber** Select the `Key` and **Input** check boxes.

  The **LineNumber** field will be used as the identifier for the nested table, or `Sequence Key`. The key for a nested table must always be used for input.

- **Model** Select the **Input** and **Predictable** check boxes.

Verify that the selections are correct, and then click **Next**.

8. On the **Specify Columns' Content and Data Type** page, verify that the grid contains the columns, content types, and data types shown in the following table, and then click **Next**.

TABLES/COLUMNS	CONTENT TYPE	DATA TYPE
IncomeGroup	Discrete	Text
OrderNumber	Key	Text
Region	Discrete	Text
vAssocSeqLineItems		
Line Number	Key Sequence	Long
Model	Discrete	Text

9. On the **Create Testing Set** page, change the **Percentage of data for testing** to 20, and then click **Next**.
10. On the **Completing the Wizard** page, for the **Mining structure name**, type `Sequence Clustering with Region`.
11. For the **Mining model name**, type `Sequence Clustering with Region`.
12. Check the **Allow drill through** box, and then click **Finish**.

Next Task in Lesson

Processing the Sequence Clustering Model

See Also

Data Mining Designer
Microsoft Sequence Clustering Algorithm

Processing the Sequence Clustering Model

2/11/2019 • 2 minutes to read • Edit Online

After you create a new mining structure, you must deploy the changes that you made to the data mining solution, and then process the structure. After processing of both the new structure and the mining model is complete, you can browse the mining model.

Processing is always required when you create a new data mining structure. However, if you add a new mining model to an existing structure, you can process just the mining model. In this scenario, because you have created a new mining structure and a new mining model, you must process both.

To process the mining structure and model

1. On the **Mining Model** menu of SQL Server Data Tools (SSDT), select **Process Mining Structure and All Models**.
2. At the warning asking whether you want to build and deploy the project, click **Yes**.
3. In the **Process Mining Structure - Sequence Clustering with Region** dialog box, click **Run**.

   The **Process Progress** dialog box opens to display information about model processing. Processing of the new structure and model might take some time.
4. After processing is complete, click **Close** to exit the **Process Progress** dialog box.
5. Click **Close** again to exit the **Process Mining Structure - Sequence Clustering with Region** dialog box.

Next Task in Lesson

Exploring the Sequence Clustering Model (Intermediate Data Mining Tutorial)

See Also

Data Mining Designer
Microsoft Sequence Clustering Algorithm
Processing Requirements and Considerations (Data Mining)

Exploring the Sequence Clustering Model (Intermediate Data Mining Tutorial)

2/11/2019 • 13 minutes to read • Edit Online

Now that you have built the **Sequence Clustering with Region** model, you can explore it by using the Microsoft Sequence Clustering Viewer in the **Mining Model Viewer** tab of Data Mining Designer. The Microsoft Sequence Cluster Viewer contains five tabs: **Cluster Diagram**, **Cluster Profiles**, **Cluster Characteristics**, **ClusterDiscrimination**, and **State Transitions**. For more information about how to use this viewer, see Browse a Model Using the Microsoft Sequence Cluster Viewer.

- Cluster Diagram tab
- Cluster Profiles tab
- Cluster Characteristics tab
- Cluster Discrimination tab
- State Transitions tab
- Generic Content View

Cluster Diagram Tab

The **Cluster Diagram** tab graphically displays the clusters that the algorithm discovered in the database. The layout in the diagram represents the relationships of the clusters, with similar clusters grouped close together. By default, the shade of each node represents the density of all cases in the cluster: the darker the shade of the node, the more cases it contains. You can change the meaning of the shading of the nodes so that it represents support, within each cluster, for an attribute and a state.

You can also rename the clusters, to make it easier to identify and work with target clusters. For this tutorial, you will rename the cluster that has the highest percentage of customers from the Pacific region, and the cluster that has the most cases overall.

> **NOTE**
>
> The cases that are assigned to specific clusters might change when you reprocess the model, depending on the data and the model parameters. Also, if you rename clusters, the names will be lost when you reprocess the mining model.

To change the attribute used for highlighting clusters

1. In the **Shading Variable** list, select **Model**.
2. Select **Cycling Cap** in the **State** list.

   The diagram updates to show the concentration of the selected product in each of the clusters. The cluster that has the darkest shading contains the highest density of cycling caps. You can change the shading variable to use any state of any input column.
3. In the **Shading Variable** list, select **Population**.

   When you change the shading variable to population, the diagram updates to compare the clusters by size. The cluster that has the darkest shading contains more cases than the other clusters.

To rename nodes in the model

1. Change **Shading Variable** to `Region`, and set **State** to **Pacific**.
2. Highlight the darkest node in the graph.
3. Right-click this cluster and select **Rename Cluster.**
4. Type the name**Pacific Cluster.**
5. Change the value of **Shading Variable** to **Population**.
6. In the updated graph, locate the darkest cluster, which should be the largest cluster. If you cannot tell by the shading which cluster is largest, pause the mouse over each cluster and view the ToolTip, and then choose the cluster that contains the most cases.
7. Right-click this cluster and select **Rename Cluster**. Type the new name, `Largest Cluster`.

You can drill through from the node that represents the cluster to view details of the cases that are in each cluster. This can be useful if you want to take action on the results of your analysis, such as sending e-mail to a customer. You can also browse the other attributes of the cases that you included in the structure but did not use in the model, such as Region and IncomeGroup. For more information about drilling through from mining models to the underlying cases, see Drillthrough Queries (Data Mining).

To drill through to details from the Cluster diagram

1. Right-click `Pacific Cluster`, select **Drill Through**, and then select **Model and Structure columns**.

   The **Drill Through** dialog box opens. Columns that are not used in the model but that are available for querying are prefixed with **Structure**.

   You can see that this cluster contains mostly customers from the Pacific region, with only a few customers from other regions.

2. Click the plus sign in the nested column v Assoc Seq Line Items to view the sequence of items in a particular customer order.
3. Close the **Drill Through** dialog box.

> **NOTE**
> The **Play** button enables you to requery the data; however, requerying does not change the data that is displayed, unless the model has been dynamically updated in the background by some other process.

Back to Top

Cluster Profiles Tab

The **Cluster Profiles** tab displays the sequences that are in each cluster. The clusters are listed in individual columns to the right of the **States** column.

In the viewer, the **Model** row describes the overall distribution of items in a cluster, and the **Model.samples** row contains sequences of the items. Each line of the color sequences in each cell of the **Model.samples** row represents the behavior of a randomly selected user in the cluster.

Each color in an individual sequence histogram represents a product model. The Mining Legend shows you the sequences of products by using both color-coding and the product model names. If you have added other columns to the model for clustering, such as Region or Income Group, the viewer will contain an additional row for each column that shows the distribution of these values within each cluster.

To view the sequences that are most common in a cluster

1. Right-click the **Model** row in the column for the cluster `Largest Cluster`, and select **Show Legend**.

   The **Color** column contains a shaded bar that indicates the frequency of items found in sequences. Each item is represented by a different color. The **Meaning** column lists the product model names for each color. The **Distribution** column tells you the percentage of cases that contained this item in a sequence.

2. Close the **Mining Legend**.

3. Right-click the **Model.samples** row in the column with the heading, **Population,** and select **Show Legend**.

4. Scan the list of sequences in the overall model `.`

   The Mining Legend lists the most common sequences first, so you can see that Mountain Tire Tube is the first item in many sequences. This means that a customer is very likely to put the Mountain Tire Tube in the shopping basket first.

To drill through to cases from the cluster viewer

1. Scroll down in the Attribute pane until you find the row for the `Region` attribute.

   The row contains a histogram for each cluster in the model, plus one additional histogram for **Population**, meaning the entire set of cases used in the model. A histogram is a bar with different colors in it, where each color represents an attribute, and the size of the colored section for that attribute represents the percentage of cases with that attribute.

2. Compare the histograms for the clusters that you renamed `Pacific Cluster` and `Largest Cluster`. Each cluster appears in a different column.

   Both look like solid colors, but the colors are different.

3. In the `Region` row, pause the mouse over the colored histogram for `Largest Cluster`.

   The ToolTip displays values that show the actual percentages of cases from each region.

4. Right-click the colored histogram in the `Region` row for `Pacific Cluster`, select **Drill Through**, and then select **Model Columns Only**.

5. Move the scroll bar to review all of the customers in this cluster.

   Again, from drilling through to the details you can see that the cluster contains mostly orders from the Pacific region but also a few from the North America and Europe regions.

6. Close the **Drill Through** dialog box.

Back to Top

Cluster Characteristics Tab

The **Cluster Characteristics** tab summarizes the transitions between states in a cluster by displaying bars that visually represent the importance of the attribute value for the selected cluster. The **Variables** column tells you what the model found to be important for the selected cluster or population: either a particular value or the relationship between values, known as *transition*. The **Values** column provides more detail about the value or transition, and the **Probability** column visually represents the weight of this attribute or transition.

To view the important attributes for a cluster

1. In the **Cluster** dropdown list, select `Pacific Cluster`.

   The list updates to show the characteristics of the cluster that you renamed `Pacific Cluster`. In this cluster, the most important characteristic is `Region`.

2. Pause the mouse over the shaded bar in the row for `Region`.

   The probability of the value being Pacific is very high. For more information about how to interpret these values, see Microsoft Sequence Clustering Algorithm Technical Reference.

3. Look through the list of characteristics for the cluster until you find the first transition row.

4. A transition row contains the text Transition in the **Variables** column, and some combination of sequential attribute values in the **Value** column. The sequence can also contain starting points and missing values.

   For example, suppose the transition has the value, [Start] -> Road Tire Tube. This means that customers in this cluster frequently put the Road Tire Tube in their shopping basket first. This might signify that the product is a popular item that customers seek out first, or it might only indicate that the product is easy to find on the purchasing site.

5. Scroll through the list until you find the first transition that does not have **[Start]** or **missing** in it.

   For example, suppose you find the transition, **Touring Tire, Touring Tire Tube**. This means that customers in this cluster frequently purchased these items together, in exactly this order.

6. Pause the mouse over the shaded bar for this transition.

   The probability of this transition is displayed as a percentage.

7. In the **Cluster** dropdown list, select **Population (All)**.

   The list of attributes updates to show the characteristics of all orders used to create the model. In this mining model, the most important characteristic for distinguishing between clusters is `Region`, with a value of **North America**.

After reviewing these tasks, you realize two things. The first is that you need a lot of data to obtain a meaningful number of combinations. For example, the sequences with the highest probabilities are likely to include a **[Start]** or **Missing** state.

The second is that there is a strong clustering effect on attributes for `Region`, which makes it more difficult to see the groups of sequences. Therefore, you decide to create another model that uses sequences only, and does not include the columns for region or income.

Back to Top

Cluster Discrimination Tab

The **Cluster Discrimination** tab helps you compare two clusters, to determine which attributes distinguish a particular cluster from another cluster. The tab contains four columns: **Variables**, **Values**, **Cluster 1**, and **Cluster 2**. You can choose any cluster to use as **Cluster 1** and **Cluster 2**.

The **Variables** column tells you the name of the attribute, which can either be a column name or combination of column name and the word **transition**. The **Values** column shows the exact value of the attribute or the transition. The shaded bars in the columns for **Cluster 1** and **Cluster 2** indicate the strength of the attribute in the clusters that you are comparing. The longer the bar, the more the cluster is likely to include cases with that attribute.

To compare two clusters by using the Cluster Discrimination tab

1. In the **Cluster Discrimination** tab, for **Cluster 1**, select `Pacific Cluster`.

   By default, the selection for **Cluster 2** changes to **Complement of Pacific Cluster**.

   The top attribute that distinguishes `Pacific Cluster` from all other cases is the region. Region is such a strong attribute for clustering that it obscures other attributes. To avoid this effect, try comparing several of the smaller clusters to each other. When you do so, the list of attributes changes and might include more

transitions between models.

2. Locate a transition row, and pause the mouse over the shaded bar.

   The items in the **Values** column can include both states and transitions. The shading for each item indicates the discrimination score. To learn more about the meaning of different scores, see Mining Model Content for Sequence Clustering Models (Analysis Services - Data Mining).

Back to Top

State Transitions Tab

On the **State Transitions** tab, you can select a cluster and browse through its state transitions. If you select **Population (All)** from the cluster drop-down list, the diagram shows the distribution of states for the whole mining model.

Each node in the graph represents a state, or possible value, of the sequences that you are trying to analyze. The background color of the nodes represents the frequency of that state. Lines connect some states, indicating a transition between states. You can move the slider up or down to change the probability threshold for the transitions. Numbers are associated with some nodes, indicating the probability of that state.

To explore the relationships in the State Transition tab

1. In the **State Transitions** tab of the Mining Model viewer, select `Pacific Cluster` from the list of clusters. Ensure that the **Show Edge Labels** option is selected.

   The graph updates to show the transitions that are most common in this cluster.

2. Click any node that is connected by a line to another node.

   The graph is updated and highlights the related nodes. The numeric value next to the line indicates the probability of the transition.

3. Raise the slider up to **All Links**, to increase the number of transitions included in the graph.

4. Select **Population (All)** from **Cluster**.

   Note that when you load a different cluster, the graph resets to the default display settings, so the slider control is reset to the middle position.

5. Click the darkest node in the graph, which should be **Sport-100**.

   Note that there are no lines connecting this product to other products.

6. Raise the slider up one step, to increase the number of transitions included in the graph. Do not go all the way to **All Links** yet.

   The graph is updated by adding several more transitions to the graph, but none that include the Sport-100 model.

7. Move the slider control all the way to **All Links**. Click the Sport-100 node if it is not already selected.

   The graph updates to show many transitions that include the Sport-100 product. The direction of the arrow on the connecting line tells you whether the Sport-100 item was selected as the first item or the second item in the pair.

8. Clicking the node for Touring Tire and move the slider control back down to the middle position.

   At first, there are many transition lines connecting Touring Tire to other products, but when you raise the probability threshold, the less likely transitions are eliminated from the graph, leaving just the transition, Touring Tire > Touring Tire Tube. This transition means that if a customer puts a Touring Tire into the shopping basket, there is a strong probability that the customer will next put a Touring Tire Tube into the

basket.

Back to Top

Generic Content Tree Viewer

This viewer can be used for all models, regardless of the algorithm or model type. The **MicrosoftGeneric Content Tree Viewer** is available from the **Viewer** drop-down list.

A content tree is a representation of any mining model as a series of nodes, wherein each node represents learned knowledge about the training data. The node can contain a pattern, a set of rules, a cluster, or the definition of a range of dates that share some attributes. The exact content of the node differs depending on the algorithm and the predictable attribute, but the general representation of the content is the same.

You can expand each node to see increasing levels of detail, and copy the content of any node to the Clipboard. For more information, see Browse a Model Using the Microsoft Generic Content Tree Viewer.

To view details for a sequence clustering model by using the Generic Content Tree Viewer

1. In the **Mining Model Viewer** tab, click the **Viewer** list, and select **Microsoft Generic Content Tree viewer**.

2. In the **Node Caption** pane, click `Pacific Cluster (1)`.

   The name for this node contains both the friendly name that you assigned to the cluster and the underlying node ID. You can use the node IDs to drill down into additional detail in the model.

3. Expand the first child node, named **Sequence level for cluster 1**.

   The sequence level node for a cluster contains details about the states and transitions that are included in that cluster. You can use these details, available in the NODE_DISTRIBUTION column, to explore the sequences and the states for each cluster or for the model as a while.

4. Continue to expand nodes and view the details in the HTML viewer pane.

For more information about the mining model content, and how to use the details in the viewer, see Mining Model Content for Sequence Clustering Models (Analysis Services - Data Mining).

Back to Top

Next Task in Lesson

Creating a Related Sequence Clustering Model (Intermediate Data Mining Tutorial)

See Also

Microsoft Sequence Clustering Algorithm
Sequence Clustering Model Query Examples

Creating a Related Sequence Clustering Model (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Through your exploration of the sequence clustering model, you learned that other attributes such as Region or Income have a strong effect on the models; therefore, to understand the sequences better, you will create a related sequence clustering model and remove the attributes related to customer demographics.

In this task, you will create a copy of the regional sequence clustering model, and then remove from the model any columns that are not directly related to the sequences.

The new model will contain all the same columns as the mining model on which it is based. However, you do not need to remove the columns from the mining structure, only specify that the new mining model ignore the columns.

To make a copy of the sequence clustering model

1. In SQL Server Data Tools (SSDT), in the Data Mining Designer, click the **Mining Models** tab.
2. Right-click the model you want to copy, and select **New Mining Model**.
3. In the **New Mining Model** dialog box, type a model name, and select Microsoft `Sequence Clustering`.

   For this tutorial, type the name `Sequence Clustering`.
4. Click **OK**.

To remove columns from the mining model

1. In the **Mining Model** tab, in the column for the new model named Sequence Clustering, click the row for the **Income Group** attribute, and select **Ignore**.
2. Repeat this step for the attribute **Region**.
3. Click the plus sign next to the table name, **v Assoc Seq Line Items**, to expand the table and view the columns from the nested table.

   The new model should have only the following columns:

   Order NumberKey

   Line Number Key

   Model Predict

To process the new sequence clustering model

1. In the **Mining Model** tab, right-click the new model named `Sequence Clustering`, and select **Process Model**.

   Because the new simplified mining model is based on a structure that has already been processed, you do not need to reprocess the structure. You can process just the new mining model.
2. Click **Yes** to deploy the updated data mining project to the server.
3. In the **Process Mining Model** dialog box, click **Run**.
4. Click **Close** to close the **Process Progress** dialog box, and then click **Close** again in the **Process Mining**

Model dialog box.

Next Task in Lesson

Creating Predictions on a Sequence Clustering Model (Intermediate Data Mining Tutorial)

See Also

Processing Requirements and Considerations (Data Mining)

Creating Predictions on a Sequence Clustering Model (Intermediate Data Mining Tutorial)

2/11/2019 • 10 minutes to read • Edit Online

After you understand the sequence clustering model better by browsing it in the viewer, you can create prediction queries by using Prediction Query Builder on the **Mining Model Prediction** tab in Data Mining Designer. To create a prediction, you first select the sequence clustering model, and then select the input data. For inputs, you can use either an external data source, or you can build a singleton query and provide values in a dialog box.

This lesson assumes that you are already familiar with how to use the prediction query builder and want to learn how to build queries that are specific to a sequence clustering model. For general information about how to use Prediction Query Builder, see Data Mining Query Interfaces or the section of the Basic Data Mining tutorial, Creating Predictions (Basic Data Mining Tutorial).

Creating Predictions on the Regional Model

For this scenario, you will first create some singleton prediction queries, to get an idea of how predictions might be different by region.

To create a singleton query on a sequence clustering model

1. Click the **Mining Model Prediction** tab of Data Mining Designer.
2. In the **Mining Model** column menu, select **Singleton Query**.

   The **Mining Model** pane and **Singleton Query Input** pane appear.
3. In the **Mining Model** pane, click **Select Model**. (You can skip this step if the sequence clustering mode is already selected.)

   The **Select Mining Model** dialog box opens.
4. Expand the node that represents the mining structure **Sequence Clustering with Region**, and select the model **Sequence Clustering with Region**. Click **OK**. For now, ignore the input pane; you will specify the inputs after you have set up the prediction functions.
5. In the grid, click the empty cell under **Source** and select **Prediction Function.** In the cell under **Field**, select **PredictSequence**.

   > **NOTE**
   >
   > You can also use the **Predict** function. If you do, be sure to choose the version of the **Predict** function that takes a table column as argument..

6. In the **Mining Model** pane, select the nested table `v Assoc Seq Line Items`, and drag it into the grid, to the **Criteria/Argument** box for the **PredictSequence** function.

   Dragging and dropping table and column names enables you to build complex statements without syntax errors. However, it replaces the current contents of the cell, which include other optional arguments for the **PredictSequence** function. To view the other arguments, you can temporarily add a second instance of the function to the grid for reference.
7. Click the **Result** button in the upper corner of the Prediction Query Builder.

The expected results contain a single column with the heading **Expression**. The **Expression** column contains a nested table with three columns as follows:

$SEQUENCE	LINE NUMBER	MODEL
1		Mountain-200

What do these results mean? Remember that you did not specify any inputs. Therefore, the prediction is made against the entire population of cases, and Analysis Services returns the most likely prediction overall.

Adding Inputs to a Singleton Prediction Query

Until now, you have not specified any inputs. In the next task, you will use the **Singleton Query Input** pane to specify some inputs to the query. First, you will use [Region] as an input to the regional sequence clustering model, to determine whether the predicted sequences are the same for all regions. You will then learn how to modify the query to add the probability for each prediction, and flatten the results to make them easier to view.

To generate predictions for a specific customer group

1. Click the **Design** button in the upper left-hand corner of the Prediction Query Builder to switch back to the query building grid.
2. In the **Singleton Query Input** dialog box, click the **Value** box for `Region`, and select **Europe**.
3. Click the **Result** button to view predictions for customers in Europe.
4. Click the **Design** button in the upper left-hand corner of the Prediction Query Builder to switch back to the query building grid.
5. In the **Singleton Query Input** dialog box, click the **Value** box for `Region`, and select **North America**.
6. Click the **Result** button to view predictions for customers in North America.

Adding Probabilities by Using a Custom Expression

To output the probability for each prediction is slightly more complicated, because the probability is an attribute of the prediction and is output as a nested table. If you are familiar with Data Mining Extensions (DMX), you can easily alter the query to add a sub-select statement on the nested table. However, you can also create a sub-select statement in the Prediction Query Builder by adding a custom expression.

To output probabilities for a predicted sequence by using a custom expression

1. Click the **Design** button in the upper left-hand corner of the Prediction Query Builder to switch back to the query building grid.
2. In the grid, under **Source**, click a new row, and select **Custom Expression**.
3. Leave the box under **Field** blank.
4. For **Alias**, type `t`.
5. In the **Criteria/Argument** box, type the complete sub-select statement as shown in the following code sample. Be sure to include the starting and ending parentheses.

```
(SELECT PredictProbability([Model]) FROM PredictSequence([Sequence Clustering with Region].[v Assoc Seq
Line Items]))
```

6. Click the **Result** button to view predictions for customers in Europe.

The results now contain two nested tables, one with the prediction, and one with the probability for the prediction. If the query does not work, you can switch to query design view and review the complete query statement, which should be as follows:

```
SELECT
  PredictSequence([Sequence Clustering with Region].[v Assoc Seq Line Items]),
  ( (SELECT PredictProbability([Model]) FROM PredictSequence([Sequence Clustering with Region].[v Assoc Seq
Line Items]))) as [t]
FROM
  [Sequence Clustering with Region]
NATURAL PREDICTION JOIN
(SELECT 'Europe' AS [Region]) AS t
```

Working with Results

When there are many nested tables in the results, you might want to flatten the results for easier viewing. To do this, you can manually modify the query and add the `FLATTENED` keyword.

To flatten nested rowsets in a prediction query

1. Click the **Query** button in the corner of the Prediction Query Builder.

   The grid changes to an open pane where you can view and modify the DMX statement that was created by the Prediction Query Builder.

2. After the `SELECT` keyword, type `FLATTENED`.

   The complete text of the query should be similar to the following:

   ```
   SELECT FLATTENED
     PredictSequence([Sequence Clustering with Region].[v Assoc Seq Line Items]),
     ( (SELECT PredictProbability([Model]) FROM PredictSequence([Sequence Clustering with Region].[v Assoc
   Seq Line Items]))) as [t]
   FROM
     [Sequence Clustering with Region]
   NATURAL PREDICTION JOIN
   (SELECT 'Europe' AS [Region]) AS t
   ```

3. Click the **Results** button in the upper corner of the Prediction Query Builder.

After you have manually edited a query, you will not be able to switch back to Design view without losing the changes. You can, however, save the DMX statement that you created manually to a text file, and then change back to Design view. When you do so, the query is reverted to the last version that was valid in Design view.

Creating Predictions on the Related Model

The previous examples used a case table column, Region, as the input to the singleton prediction query, because you were interested in knowing whether the model had found any differences between regions. However, after exploring the model, you decided that the differences are not strong enough to justify customizing product recommendations by region. What you are really interested in predicting is the items that customers select. Therefore, in the queries that follow, you will use the sequence clustering model that does not include Region, to generate recommendations for all customers.

Using Nested Table Columns as Input

First you will create a singleton prediction query that takes a single item as input and returns the next most likely item. To get a prediction of this kind, you have to use a nested table column as the input value. This is because the attribute that you are predicting, Model, is part of a nested table. Analysis Services provides the **Nested Table Input** dialog box to help you easily create prediction queries on nested table attributes, by using the Prediction Query Builder.

To use a nested table as input to a prediction

1. Click the **Design** button in the upper left-hand corner of the Prediction Query Builder to switch back to the query building grid.

2. In the **Singleton Query Input** dialog box, click the **Value** box for `Region`, and select the empty row to clear the input for this field.
3. In the **Singleton Query Input** dialog box, click the **Value** box for `vAssocSeqLineItems`, and then click the (...) button.
4. In the **Nested Table Input** dialog box, click **Add**.
5. In the new row, click the box under `Model`, and select Touring Tire from the list. Click **OK**.
6. Click the **Result** button to view the predictions.

The model recommends the following next items for all customers who choose the Touring Tire as the first item. You already know from exploring the model that customers frequently purchase the products Touring Tire and Touring Tire Tube together, so these recommendations look good.

$SEQUENCE	LINE NUMBER	MODEL
1		Touring Tire Tube
2		Sport-100
3		Long-Sleeve Logo Jersey

Creating a Bulk Prediction Query using Nested Table Inputs

Now that you are satisfied that the model creates the kind of predictions that you can use in making recommendations, you will create a prediction query that is mapped to an external data source. That data source will provide values representing current products. Because you are interested in creating a prediction query that provides Customer ID and a list of products as input, you will add the customer table as the case table, and the purchases table as the nested table. Then you will add prediction functions as you did previously to create recommendations.

This is the same procedure that you use to create predictions for the market basket scenario in Lesson 3; however, in a sequence clustering model predictions also need the order as input.

To create a prediction query using nested table inputs

1. In the **Mining Model** pane, select the Sequence Clustering model, if it is not already selected.
2. In the **Select Input Table(s)** dialog box, click **Select Case Table**.
3. In the **Select Table** dialog box, for Data Source, select Orders. In the **Table/View Name** list, select vAssocSeqOrders, and then click **OK**.
4. In the **Select Input Table(s)** dialog box, click **Select Nested Table**.
5. In the **Select Table** dialog box, for **Data Source**, select Orders. In the **Table/View name** list, select vAssocSeqLineItems, and then click **OK**.

   Analysis Services will try to detect relationships and create them automatically if the data types match and the column names are similar. If the relationships that it creates are wrong, you can right-click the join line and select **Modify Connections** to edit the column mapping, or you can right-click the join line and select **Delete** to remove the relationship completely. In this case, because the tables were already joined in the data source view, those relationships are automatically added to the design pane.
6. Add a new row to the grid. For **Source**, select vAssocSeqOrders, and for **Field**, select CustomerKey.
7. Add a new row to the grid. For **Source**, select **Prediction Function**, and for **Field**, select **PredictSequence**.

8. Drag vAssocSeqLineItems, into the **Criteria/Argument** box. Click at the end of the **Criteria/Argument** box and then type the following arguments: `2`.

   The complete text in the **Criteria/Argument** box should be:
   `[Sequence Clustering].[v Assoc Seq Line Items],2`

9. Click the **Result** button to view the predictions for each customer.

You have completed the tutorial on sequence clustering models.

Next Steps

If you have finished all the sections in the Intermediate Data Mining Tutorial (Analysis Services - Data Mining), the next step might be to learn to use Data Mining Extensions (DMX) statements to build models and generate predictions. For more information, see Creating and Querying Data Mining Models with DMX: Tutorials (Analysis Services - Data Mining).

If you are familiar with programming concepts, you can also use Analysis Management Objects (AMO) to programmatically work with data mining objects. For more information, see AMO Data Mining Classes.

See Also

Sequence Clustering Model Query Examples
Mining Model Content for Sequence Clustering Models (Analysis Services - Data Mining)

Lesson 5: Building Neural Network and Logistic Regression Models (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

The Operations department of Adventure Works is engaged in a project to improve customer satisfaction with their call center. They hired a vendor to manage the call center and to report metrics on call center effectiveness, and have asked you to analyze some preliminary data provided by the vendor. They want to know if there are any interesting findings. In particular, they would like to know if the data suggests any staffing problems with staffing or ways to improve customer satisfaction.

The data set is small and covers only a 30-day period in the operation of the call center. The data tracks the number of new and experienced operators in each shift, the number of incoming calls, the number of orders as well as issues that must be resolved, and the average time a customer waits for someone to respond to a call. The data also includes a service quality metric based on *abandon rate*, which is an indicator of customer frustration.

Because you do not have any prior expectations about what the data will show, you decide to use a neural network model to explore possible correlations. Neural network models are often used for exploration because they can analyze complex relationships between many inputs and outputs.

What You Will Learn

In this lesson, you will use the neural network algorithm to build a model that you and the Operations team can use to understand the trends in the data. As part of this lesson, you will try to answer the following questions:

- What factors affect customer satisfaction?
- What can the call center do to improve service quality?

Based on the results, you will then build a logistic regression model that you can use for predictions. The predictions will be used by the Operations team as an aid in planning call center operation.

This lesson contains the following topics:

- Adding a Data Source View for Call Center Data (Intermediate Data Mining Tutorial)
- Creating a Neural Network Structure and Model (Intermediate Data Mining Tutorial)
- Exploring the Call Center Model (Intermediate Data Mining Tutorial)
- Adding a Logistic Regression Model to the Call Center Structure (Intermediate Data Mining Tutorial)
- Creating Predictions for the Call Center Models (Intermediate Data Mining Tutorial)

Next Task in Lesson

Adding a Data Source View for Call Center Data (Intermediate Data Mining Tutorial)

All Lessons

Lesson 1: Creating the Intermediate Data Mining Solution (Intermediate Data Mining Tutorial)

Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)

Lesson 4: Building a Sequence Clustering Scenario (Intermediate Data Mining Tutorial)

Lesson 5: Neural Network and Logistic Regression Scenario (Intermediate Data Mining Tutorial)

See Also

Basic Data Mining Tutorial
Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

Adding a Data Source View for Call Center Data (Intermediate Data Mining Tutorial)

2/11/2019 • 4 minutes to read • Edit Online

In this task, you add a data source view that will be used to access the call center data. The same data will be used to build both the initial neural network model for exploration, and the logistic regression model that you will use to make recommendations.

You will also use the Data Source View Designer to add a column for the day of the week. That is because, although the source data tracks call center data by dates, your experience tells you that there are recurring patterns both in terms of call volume and service quality, depending on whether the day is a weekend or a weekday.

Procedures

To add a data source view

1. In **Solution Explorer**, right-click **Data Source Views**, and select **New Data Source View**.

   The Data Source View Wizard opens.

2. On the **Welcome to the Data Source View Wizard** page, click **Next**.

3. On the **Select a Data Source** page, under **Relational data sources**, select the Adventure Works DW Multidimensional 2012 data source. If you do not have this data source, see Basic Data Mining Tutorial. Click **Next**.

4. On the **Select Tables and Views** page, select the following table and then click the right arrow to add it to the data source view:

   - **FactCallCenter (dbo)**
   - **DimDate**

5. Click **Next**.

6. On the **Completing the Wizard** page, by default the data source view is named Adventure Works DW Multidimensional 2012. Change the name to **CallCenter**, and then click **Finish**.

   Data Source View Designer opens to display the **CallCenter** data source view.

7. Right-click inside the Data Source View pane, and select **Add/Remove Tables**. Select the table, **DimDate** and click **OK**.

   A relationship should be automatically added between the `DateKey` columns in each table. You will use this relationship to get the column, **EnglishDayNameOfWeek**, from the **DimDate** table and use it in your model.

8. In the Data Source View designer, right-click the table, **FactCallCenter**, and select **New Named Calculation**.

   In the **Create Named Calculation** dialog box, type the following values:

Column name	DayOfWeek

Description	Get day of week from DimDate table
Expression	`(SELECT EnglishDayNameOfWeek AS DayOfWeek FROM DimDate where FactCallCenter.DateKey = DimDate.DateKey)`

To verify that the expression creates the data you need, right-click the table **FactCallCenter**, and then select **Explore Data**.

9. Take a minute to review the data that is available, so that you can understand how it is used in data mining:

COLUMN NAME	CONTAINS
FactCallCenterID	An arbitrary key created when the data was imported to the data warehouse. This column identifies unique records and should be used as the case key for the data mining model.
DateKey	The date of the call center operation, expressed as an integer. Integer date keys are often used in data warehouses, but you might want to obtain the date in date/time format if you were going to group by date values. Note that dates are not unique because the vendor provides a separate report for each shift in each day of operation.
WageType	Indicates whether the day was a weekday, a weekend, or a holiday. It is possible that there is a difference in quality of customer service on weekends vs. weekdays so you will use this column as an input.
Shift	Indicates the shift for which calls are recorded. This call center divides the working day into four shifts: AM, PM1, PM2, and Midnight. It is possible that the shift influences the quality of customer service so you will use this as an input.
LevelOneOperators	Indicates the number of Level 1 operators on duty. Call center employees start at Level 1 so these employees are less experienced.
LevelTwoOperators	Indicates the number of Level 2 operators on duty. An employee must log a certain number of service hours to qualify as a Level 2 operator.
TotalOperators	The total number of operators present during the shift.
Calls	Number of calls received during the shift.
AutomaticResponses	The number of calls that were handled entirely by automated call processing (Interactive Voice Response, or IVR).

COLUMN NAME	CONTAINS
Orders	The number of orders that resulted from calls.
IssuesRaised	The number of issues requiring follow-up that were generated by calls.
AverageTimePerIssue	The average time required to respond to an incoming call.
ServiceGrade	A metric that indicates the general quality of service, measured as the *abandon rate* for the entire shift. The higher the abandon rate, the more likely it is that customers are dissatisfied and that potential orders are being lost.

Note that the data includes four different columns that are based on a single date column: `WageType`, **DayOfWeek**, `Shift`, and `DateKey`. Ordinarily in data mining it is not a good idea to use multiple columns that are derived from the same data, as the values correlate with each other too strongly and can obscure other patterns.

However, we will not use `DateKey` in the model because it contains too many unique values. There is no direct relationship between `Shift` and **DayOfWeek**, and `WageType` and **DayOfWeek** are only partly related. If you were worried about collinearity, you could create the structure using all of the available columns, and then ignore different columns in each model and test the effect.

Next Task in Lesson

Creating a Neural Network Structure and Model (Intermediate Data Mining Tutorial)

See Also

Data Source Views in Multidimensional Models

Creating a Neural Network Structure and Model (Intermediate Data Mining Tutorial)

2/11/2019 • 11 minutes to read • Edit Online

To create a data mining model, you must first use the Data Mining Wizard to create a new mining structure based on the new data source view. In this task you will use the wizard to create a mining structure, and at the same time create an associated mining model that is based on the Microsoft Neural Network algorithm.

Because neural networks are extremely flexible and can analyze many combinations of inputs and outputs, you should experiment with several ways of processing the data to get the best results. For example, you might want to customize the way that the numerical target for service quality is *binned*, or grouped, to target specific business requirements. To do this, you will add a new column to the mining structure that groups numerical data in a different way, and then create a model that uses the new column. You will use these mining models to do some exploration.

Finally, when you have learned from the neural network model which factors have the greatest impact for your business question, you will build a separate model for prediction and scoring. You will use the Microsoft Logistic Regression algorithm, which is based on the neural networks model but is optimized for finding a solution based on specific inputs.

Steps

Create the default mining structure and model

Use discretization to bin the predictable column

Copy the column and change the discretization method for a different model

Create an alias for the predictable column so that you can compare models

Process all models

Create the Default Call Center Structure

1. In Solution Explorer in SQL Server Data Tools (SSDT), right-click **Mining Structures** and select **New Mining Structure**.

2. On the **Welcome to the Data Mining Wizard** page, click **Next**.

3. On the **Select the Definition Method** page, verify that **From existing relational database or data warehouse** is selected, and then click **Next**.

4. On the **Create the Data Mining Structure** page, verify that the option **Create mining structure with a mining model** is selected.

5. Click the dropdown list for the option **Which data mining technique do you want to use?**, then select **Microsoft Neural Networks**.

   Because the logistic regression models are based on the neural networks, you can reuse the same structure and add a new mining model.

6. Click **Next**.

   The **Select Data Source View** page appears.

7. Under **Available data source views**, select `Call Center`, and click **Next**.
8. On the **Specify Table Types** page, select the **Case** check box next to the **FactCallCenter** table. Do not select anything for **DimDate**. Click **Next**.
9. On the **Specify the Training Data** page, select **Key** next to the column **FactCallCenterID.**
10. Select the `Predict` and **Input** check boxes.
11. Select the **Key**, **Input**, and `Predict` check boxes as shown in the following table:

TABLES/COLUMNS	KEY/INPUT/PREDICT
AutomaticResponses	Input
AverageTimePerIssue	Input/Predict
Calls	Input
DateKey	Do not use
DayOfWeek	Input
FactCallCenterID	Key
IssuesRaised	Input
LevelOneOperators	Input/Predict
LevelTwoOperators	Input
Orders	Input/Predict
ServiceGrade	Input/Predict
Shift	Input
TotalOperators	Do not use
WageType	Input

Note that multiple predictable columns have been selected. One of the strengths of the neural network algorithm is that it can analyze all possible combinations of input and output attributes. You wouldn't want to do this for a large data set, as it could exponentially increase processing time..

12. On the **Specify Columns' Content and Data Type** page, verify that the grid contains the columns, content types, and data types as shown in the following table, and then click **Next**.

COLUMNS	CONTENT TYPE	DATA TYPES
AutomaticResponses	Continuous	Long
AverageTimePerIssue	Continuous	Long

COLUMNS	CONTENT TYPE	DATA TYPES
Calls	Continuous	Long
DayOfWeek	Discrete	Text
FactCallCenterID	Key	Long
IssuesRaised	Continuous	Long
LevelOneOperators	Continuous	Long
LevelTwoOperators	Continuous	Long
Orders	Continuous	Long
ServiceGrade	Continuous	Double
Shift	Discrete	Text
WageType	Discrete	Text

13. On the **Create testing set** page, clear the text box for the option, **Percentage of data for testing**. Click **Next**.
14. On the **Completing the Wizard** page, for the **Mining structure name**, type `Call Center`.
15. For the **Mining model name**, type `Call Center Default NN`, and then click **Finish**.

    The **Allow drill through** box is disabled because you cannot drill through to data with neural network models.
16. In Solution Explorer, right-click the name of the data mining structure that you just created, and select **Process**.

Use Discretization to Bin the Target Column

By default, when you create a neural network model that has a numeric predictable attribute, the Microsoft Neural Network algorithm treats the attribute as a continuous number. For example, the ServiceGrade attribute is a number that theoretically ranges from 0.00 (all calls are answered) to 1.00 (all callers hang up). In this data set, the values have the following distribution:

As a result, when you process the model the outputs might be grouped differently than you expect. For example, if you use clustering to identify the best groups of values, the algorithm divides the values in ServiceGrade into ranges such as this one: 0.0748051948 - 0.09716216215. Although this grouping is mathematically accurate, such ranges might not be as meaningful to business users.

In this step, to make the result more intuitive, you'll group the numerical values differently, creating copies of the numerical data column.

How Discretization Works

Analysis Services provides a variety of methods for binning or processing numerical data. The following table illustrates the differences between the results when the output attribute ServiceGrade has been processed three different ways:

- Treating it as a continuous number.
- Having the algorithm use clustering to identify the best arrangement of values.
- Specifying that the numbers be binned by the Equal Areas method.

Default model (continuous)

VALUE	SUPPORT
Missing	0
0.09875	120

Binned by clustering

VALUE	SUPPORT
< 0.0748051948	34
0.0748051948 - 0.09716216215	27
0.09716216215 - 0.13297297295	39
0.13297297295 - 0.167499999975	10
>= 0.167499999975	10

Binned by equal areas

VALUE	SUPPORT
< 0.07	26
0.07 - 0.00	22
0.09 - 0.11	36
>= 0.12	36

> **NOTE**
> You can obtain these statistics from the marginal statistics node of the model, after all the data has been processed. For more information about the marginal statistics node, see Mining Model Content for Neural Network Models (Analysis Services - Data Mining).

In this table, the VALUE column shows you how the number for ServiceGrade has been handled. The SUPPORT column shows you how many cases had that value, or that fell in that range.

- **Use continuous numbers (default)**

  If you used the default method, the algorithm would compute outcomes for 120 distinct values, the mean value of which is 0.09875. You can also see the number of missing values.

- **Bin by clustering**

  When you let the Microsoft Clustering algorithm determine the optional grouping of values, the algorithm would group the values for ServiceGrade into five (5) ranges. The number of cases in each range is not evenly distributed, as you can see from the support column.

- **Bin by equal areas**

  When you choose this method, the algorithm forces the values into buckets of equal size, which in turn changes the upper and lower bounds of each range. You can specify the number of buckets, but you want to avoid having two few values in any bucket.

For more information about binning options, see Discretization Methods (Data Mining).

Alternatively, rather than using the numeric values, you could add a separate derived column that classifies the service grades into predefined target ranges, such as **Best** (ServiceGrade <= 0.05), **Acceptable** (0.10 >

ServiceGrade > 0.05), and **Poor** (ServiceGrade >= 0.10).

Create a Copy of a Column and Change the Discretization Method

You'll make a copy of the mining column that contains the target attribute, ServiceGrade and change the way the numbers are grouped. You can create multiple copies of any column in a mining structure, including the predictable attribute.

For this tutorial, you will use the Equal Areas method of discretization, and specify four buckets. The groupings that result from this method are fairly close to the target values of interest to your business users.

To create a customized copy of a column in the mining structure

1. In Solution Explorer, double-click the mining structure that you just created.
2. In the Mining Structure tab, click **Add a mining structure column**.
3. In the **Select column** dialog box, select ServiceGrade from the list in **Source column**, then click **OK**.

   A new column is added to the list of mining structure columns. By default, the new mining column has the same name as the existing column, with a numerical postfix: for example, ServiceGrade 1. You can change the name of this column to be more descriptive.

   You will also specify the discretization method.
4. Right-click ServiceGrade 1 and select **Properties**.
5. In the **Properties** window, locate the **Name** property, and change the name to **Service Grade Binned** .
6. A dialog box appears asking whether you want to make the same change to the name of all related mining model columns. Click **No**.
7. In the **Properties** window, locate the section **Data Type** and expand it if necessary.
8. Change the value of the property `Content` from `Continuous` to `Discretized`.

   The following properties are now available. Change the values of the properties as shown in the following table:

PROPERTY	DEFAULT VALUE	NEW VALUE
`DiscretizationMethod`	`Continuous`	`EqualAreas`
`DiscretizationBucketCount`	No value	4

> **NOTE**
> The default value of DiscretizationBucketCount is actually 0, which means that the algorithm automatically determines the optimum number of buckets. Therefore, if you want to reset the value of this property to its default, type 0.

9. In Data Mining Designer, click the **Mining Models** tab.

   Notice that when you add a copy of a mining structure column, the usage flag for the copy is automatically set to `Ignore`. Usually, when you add a copy of a column to a mining structure, you would not use the copy for analysis together with the original column, or the algorithm will find a strong correlation between the two columns that might obscure other relationships.

Add a New Mining Model to the Mining Structure

Now that you have created a new grouping for the target attribute, you need to add a new mining model that uses the discretized column. When you are done, the CallCenter mining structure will have two mining models:

- The mining model, Call Center Default NN, handles the ServiceGrade values as a continuous range.
- You will create a new mining model, Call Center Binned NN, that uses as its target outcomes the values of the ServiceGrade column, distributed into four buckets of equal size.

To add a mining model based on the new discretized column

1. In Solution Explorer, right-click the mining structure that you just created, and select **Open**.
2. Click the **Mining Models** tab.
3. Click **Create a related mining model**.
4. In the **New Mining Model** dialog box, for **Model name**, type `Call Center Binned NN`. In the **Algorithm name** dropdown list, select **Microsoft Neural Network**.
5. In the list of columns contained in the new mining model, locate ServiceGrade, and change the usage from `Predict` to `Ignore`.
6. Similarly, locate ServiceGrade Binned, and change the usage from `Ignore` to `Predict`.

Create an Alias for the Target Column

Ordinarily you cannot compare mining models that use different predictable attributes. However, you can create an alias for a mining model column. That is, you can rename the column, ServiceGrade Binned, within the mining model so that it has the same name as the original column. You can then directly compare these two models in an accuracy chart, even though the data is discretized differently.

To add an alias for a mining structure column in a mining model

1. In the **Mining Models** tab, under **Structure**, select ServiceGrade Binned.

   Note that the **Properties** window displays the properties of the object, ScalarMiningStructure column.

2. Under the column for the mining model, ServiceGrade Binned NN, click the cell corresponding to the column ServiceGrade Binned.

   Note that now the **Properties** window displays the properties for the object, MiningModelColumn.

3. Locate the **Name** property, and change the value to `ServiceGrade`.
4. Locate the **Description** property and type **Temporary column alias**.

   The **Properties** window should contain the following information:

PROPERTY	VALUE
Description	Temporary column alias
ID	ServiceGrade Binned
Modeling Flags	
Name	Service Grade
SourceColumn ID	Service Grade 1

PROPERTY	VALUE
Usage	Predict

5. Click anywhere in the **Mining Model** tab.

   The grid is updated to show the new temporary column alias, `ServiceGrade`, beside the column usage. The grid containing the mining structure and two mining models should look like the following:

STRUCTURE	CALL CENTER DEFAULT NN	CALL CENTER BINNED NN
	Microsoft Neural Network	Microsoft Neural Network
AutomaticResponses	Input	Input
AverageTimePerIssue	Predict	Predict
Calls	Input	Input
DayOfWeek	Input	Input
FactCallCenterID	Key	Key
IssuesRaised	Input	Input
LevelOneOperators	Input	Input
LevelTwoOperators	Input	Input
Orders	Input	Input
ServceGrade Binned	Ignore	Predict (ServiceGrade)
ServiceGrade	Predict	Ignore
Shift	Input	Input
Total Operators	Input	Input
WageType	Input	Input

Process All Models

Finally, to ensure that the models you have created can be easily compared, you will set the seed parameter for both the default and binned models. Setting a seed value guarantees that each model starts processing the data from the same point.

> **NOTE**
> If you do not specify a numeric value for the seed parameter, SQL Server Analysis Services will generate a seed based on the name of the model. Because the models always have different names, you must set a seed value to ensure that they process data in the same order.

To specify the seed and process the models

1. In the **Mining Model** tab, right-click the column for the model named Call Center - LR, and select **Set Algorithm Parameters**.

2. In the row for the HOLDOUT_SEED parameter, click the empty cell under **Value**, and type `1`. Click **OK**. Repeat this step for each model associated with the structure.

> **NOTE**
> The value that you choose as the seed does not matter, as long as you use the same seed for all related models.

3. In the **Mining Models** menu, select **Process Mining Structure and All Models**. Click **Yes** to deploy the updated data mining project to the server.

4. In the **Process Mining Model** dialog box, click **Run**.

5. Click **Close** to close the **Process Progress** dialog box, and then click **Close** again in the **Process Mining Model** dialog box.

Now that you have created the two related mining models, you will explore the data to discover relationships in the data.

Next Task in Lesson

Exploring the Call Center Model (Intermediate Data Mining Tutorial)

See Also

Mining Structures (Analysis Services - Data Mining)

Exploring the Call Center Model (Intermediate Data Mining Tutorial)

2/11/2019 • 7 minutes to read • Edit Online

Now that you have built the exploratory model, you can use it to learn more about your data by using the following tools provided in SQL Server Data Tools (SSDT).

- Microsoft Neural Network Viewer **:** This viewer is available in the **Mining Model Viewer** tab of Data Mining Designer, and is designed to help you experiment with interactions in the data.
- Microsoft Generic Content Tree Viewer **:** This standard viewer provides in-depth detail about the patterns and statistics discovered by the algorithm when it generated the model.

Microsoft Neural Network Viewer

The viewer has three panes - **Input**, **Output**, and **Variables**.

By using the **Output** pane, you can select different values for the predictable attribute, or dependent variable. If your model contains multiple predictable attributes, you can select the attribute from the **Output Attribute** list.

The **Variables** pane compares the two outcomes that you chose in terms of contributing attributes, or variables. The colored bars visually represent how strongly the variable affects the target outcomes. You can also view lift scores for the variables. A lift score is calculated differently depending on which mining model type you are using, but generally tells you the improvement in the model when using this attribute for prediction.

The **Input** pane lets you add influencers to the model to try out various what-if scenarios.

Using the Output Pane

In this initial model, you are interested in seeing how various factors affect the grade of service. To do this, you can select Service Grade from the list of output attributes, and then compare different levels of service by selecting ranges from the dropdown lists for **Value 1** and **Value 2**.

To compare lowest and highest service grades

1. For **Value 1**, select the range with the lowest values. For example, the range 0-0-0.7 represents the lowest abandon rates, and therefore the best level of service.

   > **NOTE**
   >
   > The exact values in this range may vary depending on how you configured the model.

2. For **Value 2**, select the range with the highest values. For example, the range with the value >=0.12 represents the highest abandon rates, and therefore the worst service grade. In other words, 12% of the customers who phoned during this shift hung up before speaking to a representative.

   The contents of the **Variables** pane are updated to compare attributes that contribute to the outcome values. Therefore, the left column shows you the attributes that are associated with the best grade of service, and the right column shows you the attributes associated with the worst grade of service.

Using the Variables Pane

In this model, it appears that `Average Time Per Issue` is an important factor. This variable indicates the average time that it takes for a call to be answered, regardless of call type.

To view and copy probability and lift scores for an attribute

1. In the **Variables** pane, pause the mouse over the colored bar in the first row.

   This colored bar shows you how strongly `Average Time Per Issue` contributes toward the service grade. The tooltip shows an overall score, probabilities, and lift scores for each combination of a variable and a target outcome.

2. In the **Variables** pane, right-click any colored bar and select **Copy**.

3. In an Excel worksheet, right-click any cell and select **Paste**.

   The report is pasted as an HTML table, and shows only the scores for each bar.

4. In a different Excel worksheet, right-click any cell and select **Paste Special**.

   The report is pasted as text format, and includes the related statistics described in the next section.

Using the Input Pane

Suppose that you are interested in looking at the effect of a particular factor, such as the shift, or number of operators. You can select a particular variable by using the **Input** pane, and the **Variables** pane is automatically updated to compare the two previously selected groups given the specified variable.

To review the effect on service grade by changing input attributes

1. In the **Input** pane, for **attribute**, select Shift.

2. For **Value**, select **AM**.

   The **Variables** pane updates to show the impact on the model when the shift is **AM**. All other selections remain the same - you are still comparing the lowest and highest service grades.

3. For **Value**, select **PM1**.

   The **Variables** pane updates to show the impact on the model when the shift changes.

4. In the **Input** pane, click the next blank row under **Attribute**, and select Calls. For **Value**, select the range that indicates the greatest number of calls.

   A new input condition is added to the list. The **Variables** pane updates to show the impact on the model for a particular shift when the call volume is highest.

5. Continue to change the values for Shift and Calls to find any interesting correlations between shift, call volume, and service grade.

   > **NOTE**
   >
   > To clear the **Input** pane so that you can use different attributes, click **Refresh viewer content**.

Interpreting the Statistics Provided in the Viewer

Longer waiting times are a strong predictor of a high abandon rate, meaning a poor service grade. This may seem an obvious conclusion; however, the mining model provides you with some additional statistical data to help you interpret these trends.

- **Score**: Value that indicates the overall importance of this variable for discriminating between outcomes. The higher the score, the stronger the effect the variable has on the outcome.
- **Probability of value 1**: Percentage that represents the probability of this value for this outcome.
- **Probability of value 2**: Percentage that represents the probability of this value for this outcome.
- **Lift for Value 1** and **Lift for Value 2**: Scores that represents the impact of using this particular variable for

predicting the Value 1 and Value 2 outcomes. The higher the score, the better the variable is at predicting the outcomes.

The following table contains some example values for the top influencers. For example, the **Probability of value 1** is 60.6% and **Probability of value 2** is 8.30%, meaning that when the Average Time Per Issue was in the range of 44-70 minutes, 60.6% of cases were in the shift with the highest service grades (Value 1), and 8.30% of cases were in the shift with the worse service grades (Value 2).

From this information, you can draw some conclusions. Shorter call response time (the range of 44-70) strongly influences better service grade (the range 0.00-0.07). The score (92.35) tells you that this variable is very important.

However, as you look down the list of contributing factors, you see some other factors with effects that are more subtle and more difficult to interpret. For example, shift appears to influence service, but the lift scores and the relative probabilities indicate that shift is not a major factor.

ATTRIBUTE	VALUE	FAVORS < 0.07	FAVORS >= 0.12
Average Time Per Issue	89.087 - 120.000		Score: 100 Probability of Value1: 4.45 % Probability of Value2: 51.94 % Lift for Value1: 0.19 Lift for Value2: 1.94
Average Time Per Issue	44.000 - 70.597	Score: 92.35 Probability of Value1: 60.06 % Probability of Value2: 8.30 % Lift for Value1: 2.61 Lift for Value2: 0.31	

Back to Top

Microsoft Generic Content Tree Viewer

This viewer can be used to view even more detailed information created by the algorithm when the model is processed. The **MicrosoftGeneric Content Tree Viewer** represents the mining model as a series of nodes, wherein each node represents learned knowledge about the training data. This viewer can be used with all models, but the contents of the nodes are different depending in the model type.

For neural network models or logistic regression models, you might find the `marginal statistics node` particularly useful. This node contains derived statistics about the distribution of values in your data. This information can be useful if you want to get a summary of the data without having to write many T-SQL queries. The chart of binning values in the previous topic was derived from the marginal statistics node.

To obtain a summary of data values from the mining model

1. In Data Mining Designer, in the **Mining Model Viewer** tab, select <mining model name>.

2. From the **Viewer** list, select **Microsoft Generic Content Tree Viewer**.

The view of the mining model refreshes to show a node hierarchy in the left-hand pane and an HTML table in the right-hand pane.

3. In the **Node Caption** pane, click the node that has the name 10000000000000000.

   The topmost node in any model is always the model root node. In a neural network or logistic regression model, the node immediately under that is the marginal statistics node.

4. In the **Node Details** pane, scroll down until you find the row, NODE_DISTRIBUTION.

5. Scroll down through the NODE_DISTRIBUTION table to view the distribution of values as calculated by the neural network algorithm.

To use this data in a report, you could select and then copy the information for specific rows, or you can use the following Data Mining Extensions (DMX) query to extract the complete contents of the node.

```
SELECT *
FROM [Call Center EQ4].CONTENT
WHERE NODE_NAME = '10000000000000000'
```

You can also use the node hierarchy and the details in the NODE_DISTRIBUTION table to traverse individual paths in the neural network and view statistics from the hidden layer. For more information, see Neural Network Model Query Examples.

Back to Top

Next Task in Lesson

Adding a Logistic Regression Model to the Call Center Structure (Intermediate Data Mining Tutorial)

See Also

Mining Model Content for Neural Network Models (Analysis Services - Data Mining)
Neural Network Model Query Examples
Microsoft Neural Network Algorithm Technical Reference
Change the Discretization of a Column in a Mining Model

Adding a Logistic Regression Model to the Call Center Structure (Intermediate Data Mining Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

In addition to analyzing the factors that might affect call center operations, you were also asked to provide some specific recommendations on how the staff can improve service quality. In this task, you will use the same mining structure that you used to build the exploratory model and add a mining model that will be used for creating predictions.

In Analysis Services, a logistic regression model is based on the neural networks algorithm, and therefore provides the same flexibility and power as a neural network model. However, logistic regression is particularly well-suited for predicting binary outcomes.

For this scenario, you will use the same mining structure that you used for the neural network model. However, you will customize the new model to target your business questions. You are interested in improving service quality and determining how many experienced operators you need, so you will set up your model to predict those values.

To ensure that all the models based on the call center data are as similar as possible, you will use the same seed value as before. Setting the seed parameter ensures that the model processes the data from the same starting point, and minimizes variations caused by artifacts in the data.

To add a new mining model to the call center mining structure

1. In SQL Server Data Tools (SSDT), in Solution Explorer, right-click the mining structure, **Call Center Binned**, and select **Open Designer**.
2. In Data Mining Designer, click the **Mining Models** tab.
3. Click **Create a related mining model**.
4. In the **New Mining Model** dialog box, for **Model name**, type `Call Center - LR`. For **Algorithm name**, select **Microsoft Logistic Regression**.
5. Click **OK**.

   The new mining model is displayed in the **Mining Models** tab.

To customize the logistic regression model

1. In the column for the new mining model, `Call Center - LR`, leave Fact CallCenter ID as the key.
2. Change the value of ServiceGrade and Level Two Operators to **Predict**.

   These columns will be used both as input and for prediction. In essence, you are creating two separate models on the same data: one that predicts the number of operators, and one that predicts the service grade.
3. Change all other columns to **Input**.

To specify the seed and process the models

1. In the **Mining Model** tab, right-click the column for the model named Call Center - LR, and select **Set Algorithm Parameters**.
2. In the row for the HOLDOUT_SEED parameter, click the empty cell under **Value**, and type `1`. Click **OK**.

> **NOTE**
> The value that you choose as the seed does not matter, as long as you use the same seed for all related models.

3. In the **Mining Models** menu, select **Process Mining Structure and All Models**. Click **Yes** to deploy the updated data mining project to the server.
4. In the **Process Mining Model** dialog box, click **Run**.
5. Click **Close** to close the **Process Progress** dialog box, and then click **Close** again in the **Process Mining Model** dialog box.

Next Task in Lesson

Creating Predictions for the Call Center Models (Intermediate Data Mining Tutorial)

See Also

Processing Requirements and Considerations (Data Mining)

Creating Predictions for the Call Center Models (Intermediate Data Mining Tutorial)

2/11/2019 • 9 minutes to read • Edit Online

Now that you have learned something about the interactions between shifts, the number of operators, calls, and service grade, you are ready to create some prediction queries that can be used in business analysis and planning. You will first create some predictions on the exploratory model to test some assumptions. Next, you will create bulk predictions by using the logistic regression model.

This lesson assumes that you are already familiar with the concept of prediction queries.

Creating Predictions using the Neural Network Model

The following example demonstrates how to make a singleton prediction using the neural network model that was created for exploration. Singleton predictions are a good way to try out different values to see the effect in the model. In this scenario, you will predict the service grade for the midnight shift (no day of the week specified) if six experienced operators are on duty.

To create a singleton query by using the neural network model

1. In SQL Server Data Tools (SSDT), open the solution that contains the model that you want to use.
2. In Data Mining Designer, click the **Mining Model Prediction** tab.
3. In the **Mining Model** pane, click **Select Model**.
4. The **Select Mining Model** dialog box shows a list of mining structures. Expand the mining structure to view a list of mining models associated with that structure.
5. Expand the mining structure Call Center Default, and select the neural network model, Call Center - LR.
6. From the **Mining Model** menu, select **Singleton Query**.

   The **Singleton Query Input** dialog box appears, with columns mapped to the columns in the mining model.
7. In the **Singleton Query Input** dialog box, click the row for Shift, and then select *midnight*.
8. Click the row for Lvl 2 Operators, and type `6`.
9. In the bottom half of the **Mining Model Prediction** tab, click the first row in the grid.
10. In the **Source** column, click the down arrow, and select **Prediction function**. In the **Field** column, select **PredictHistogram**.

    A list of arguments that you can use with this prediction function automatically appears in the **Criteria/Arguments** box.
11. Drag the ServiceGrade column from the list of columns in the **Mining Model** pane to the **Criteria/Arguments** box.

    The name of the column is automatically inserted as the argument. You can choose any predictable attribute column to drag into this text box.
12. Click the button **Switch to query results view**, in the upper corner of the Prediction Query Builder.

The expected results contain the possible predicted values for each service grade given these inputs, together with support and probability values for each prediction. You can return to design view at any time and change the inputs, or add more inputs.

Creating Predictions by using a Logistic Regression Model

If you already know the attributes that are relevant to the business problem, you can use a logistic regression model to predict the effect of making changes in some attributes. Logistic regression is a statistical method that is commonly used to make predictions based on changes in independent variables: for example, it is used in financial scoring, to predict customer behavior based on customer demographics.

In this task, you will learn how to create a data source that will be used for predictions, and then make predictions to help answer several business questions.

Generating Data used for Bulk Prediction

There are many ways to provide input data: for example, you might import staffing levels from a spreadsheet, and run that data through the model to predict service quality for the next month.

In this lesson, you will use the Data Source View designer to create a named query. This named query is a custom Transact-SQL statement that for each shift on the schedule calculates the maximum number of operators on staff, the minimum calls received, and the average number of issues that are generated. You will then join that data to a mining model to make predictions about a series of upcoming dates.

To generate input data for a bulk prediction query

1. In Solution Explorer, right-click **Data Source Views**, and then select **New Data Source View**.
2. In the Data Source View wizard, select Adventure Works DW Multidimensional 2012 as the data source, and then click **Next**.
3. On the **Select Tables and Views** page, click **Next** without selecting any tables.
4. On the **Completing the Wizard** page, type the name, `Shifts`.

   This name will appear in Solution Explorer as the name of the data source view.
5. Right-click the empty design pane, then select **New Named Query**.
6. In the **Create Named Query** dialog box, for **Name**, type `Shifts for Call Center`.

   This name will appear in Data Source View designer only as the name of the named query.
7. Paste the following query statement into the SQL text pane in the lower half of the dialog box.

   ```
   SELECT DISTINCT WageType, Shift,
   AVG(Orders) as AvgOrders, MIN(Orders) as MinOrders, MAX(Orders) as MaxOrders,
   AVG(Calls) as AvgCalls, MIN(Calls) as MinCalls, MAX(Calls) as MaxCalls,
   AVG(LevelTwoOperators) as AvgOperators, MIN(LevelTwoOperators) as MinOperators, MAX(LevelTwoOperators)
   as MaxOperators,
   AVG(IssuesRaised) as AvgIssues, MIN(IssuesRaised) as MinIssues, MAX(IssuesRaised) as MaxIssues
   FROM dbo.FactCallCenter
   GROUP BY Shift, WageType
   ```
8. In the design pane, right-click the table, Shifts for Call Center, and select **Explore Data** to preview the data as returned by the T-SQL query.
9. Right-click the tab, **Shifts.dsv (Design),** and then click **Save** to save the new data source view definition.

Predicting Service Metrics for Each Shift

Now that you have generated some values for each shift, you will use those values as input to the logistic

regression model that you built, to generate some predictions that can be used in business planning.

To use the new DSV as input to a prediction query

1. In Data Mining Designer, click the **Mining Model Prediction** tab.
2. In the **Mining Model** pane, click **Select Model**, and choose Call Center - LR from the list of available models.
3. From the **Mining Model** menu, clear the option, **Singleton Query**. A warning tells you that the singleton query inputs will be lost. Click **OK**.

   The **Singleton Query Input** dialog box is replaced with the **Select Input Table(s)** dialog box.
4. Click **Select Case Table**.
5. In the **Select Table** dialog box, selectShifts from the list of data sources. In the **Table/View name** list, select Shifts for Call Center (it might be automatically selected), and then click **OK.**

   The **Mining Model Prediction** design surface is updated to show mappings that are created based on the names and data types of columns in the input data and in the model.
6. Right-click one of the join lines, and then select **Modify Connections**.

   In this dialog box, you can see exactly which columns are mapped and which are not. The mining model contains columns for Calls, Orders, IssuesRaised, and LvlTwoOperators, which you can map to any of the aggregates that you created based on these columns in the source data. In this scenario, you will map to the averages.
7. Click the empty cell next to LevelTwoOperators, and select **Shifts for Call Center.AvgOperators**.
8. Click the empty cell next to Calls, select **Shifts for Call Center.AvgCalls**. and then click **OK**.

To create the predictions for each shift

1. In the grid at the bottom half of the **Prediction Query Builder**, click the empty cell under **Source,** and then select Shifts for Call Center.
2. In the empty cell under **Field**, select Shift.
3. Click the next empty line in the grid and repeat the procedure described above to add another row for WageType.
4. Click the next empty line in the grid. In the **Source** column, select **Prediction Function**. In the **Field** column, select **Predict**.
5. Drag the column ServiceGrade from the **Mining Model** pane down to the grid, and into the **Criteria/Argument** cell. In the **Alias** field, type **Predicted Service Grade**.
6. Click the next empty line in the grid. In the **Source** column, select **Prediction Function**. In the **Field** column, select **PredictProbability**.
7. Drag the column ServiceGrade from the **Mining Model** pane down to the grid, and into the **Criteria/Argument** cell. In the **Alias** field, type **Probability**.
8. Click **Switch to query result view** to view the predictions.

The following table shows sample results for each shift.

SHIFT	WAGETYPE	PREDICTED SERVICE GRADE	PROBABILITY
AM	holiday	0.165	0.377520666

SHIFT	WAGETYPE	PREDICTED SERVICE GRADE	PROBABILITY
midnight	holiday	0.105	0.364105573
PM1	holiday	0.165	0.40056055
PM2	holiday	0.165	0.338532973
AM	weekday	0.165	0.370847617
midnight	weekday	0.08	0.352999173
PM1	weekday	0.165	0.317419177
PM2	weekday	0.105	0.311672027

Predicting the Effect of Reduced Response Time on Service Grade

You generated some average values for each shift, and used those values as input to the logistic regression model. However, given that the business objective is to keep abandon rate within the range 0.00-0.05, the results are not encouraging.

Therefore, based on the original model, which showed a strong influence of response time on service grade, the Operations team decides to run some predictions to assess whether reducing the average time for responding to calls might improve service quality. For example, if you cut the call response time to 90 percent or even to 80 percent of the current call response time, what would happen to service grade values?

It is easy to create a data source view (DSV) that calculates the average response times for each shift, and then add columns that calculate 80% or 90% of the average response time. You can then use the DSV as input to the model.

Although the exact steps are not shown here, the following table compares the effects on service grade when you reduce response times to 80% or to 90% of current response times.

From these results, you might conclude that on targeted shifts you should reduce the response time to 90 percent of the current rate in order to improve service quality.

SHIFT, WAGE, AND DAY	PREDICTED SERVICE QUALITY WITH CURRENT AVERAGE RESPONSE TIME	PREDICTED SERVICE QUALITY WITH 90 PERCENT REDUCTION IN RESPONSE TIME	PREDICTED SERVICE QUALITY WITH 80 PERCENT REDUCTION IN RESPONSE TIME
Holiday AM	0.165	0.05	0.05
Holiday PM1	0.05	0.05	0.05
Holiday Midnight	0.165	0.05	0.05

There are a variety of other prediction queries that you can create on this model. For example, you could predict how many operators are required to meet a certain service level or to respond to a certain number of incoming calls. Because you can include multiple outputs in a logistic regression model, it is easy to experiment with different independent variables and outcomes without having to create many separate models.

Remarks

The Data Mining Add-Ins for Excel 2007 provide logistic regression wizards that make it easy to answer complex questions, such as how many Level Two Operators would be required to improve service grade to a target level for a specific shift. The data mining add-ins are a free download, and include wizards that are based on the neural

network or logistic regression algorithms. For more information, see the following links:

- SQL Server 2005 Data Mining Add-Ins for Office 2007: Goal Seek and What If Scenario Analysis
- SQL Server 2008 Data Mining Add-Ins for Office 2007: Goal Seek Scenario Analysis, What If Scenario Analysis, and Prediction Calculator

Conclusion

You have learned to create, customize, and interpret mining models that are based on the Microsoft Neural Network algorithm and the Microsoft Logistic Regression algorithm. These model types are sophisticated and permit almost infinite variety in analysis, and therefore can be complex and difficult to master.

However, these algorithms can iterate through many combinations of factors and automatically identify the strongest correlations, providing statistical support for insights that would be very difficult to discover through manual exploration of data using Transact-SQL or even PowerPivot.

See Also

Logistic Regression Model Query Examples
Microsoft Logistic Regression Algorithm
Microsoft Neural Network Algorithm
Neural Network Model Query Examples

Creating and Querying Data Mining Models with DMX: Tutorials (Analysis Services - Data Mining)

2/11/2019 • 2 minutes to read • Edit Online

After you have created a data mining solution by using Microsoft SQL Server Analysis Services, you can create queries against the data mining models to predict trends, retrieve patterns in the data, and measure the accuracy of the mining models.

The step-by-step tutorials in the following list will help you learn how to build and run data mining queries by using Analysis Services so that you can get the most from your data.

In This Section

- Bike Buyer DMX Tutorial

  This tutorial walks you through the creation of a new mining structure and mining models by using the Data Mining Extensions (DMX) language, and explains how to create DMX prediction queries.

- Market Basket DMX Tutorial

  This tutorial uses a typical market basket scenario, where you find associations between the products that customers purchase together. This tutorial also demonstrates how to use nested tables when you create a mining structure. You build and train a model based on this structure, and then create predictions using DMX.

- Time Series Prediction DMX Tutorial

  This tutorial creates a forecasting model to illustrate the use of the CREATE MODEL (DMX) statement. You then add related models and customize the behavior of each by changing the parameters of the Microsoft Time Series algorithm. Finally you create predictions and update the predictions with new data. The ability to update a time series while making predictions was added in SQL Server 2008.

Reference

Data Mining Algorithms (Analysis Services - Data Mining)

Data Mining Extensions (DMX) Reference

Related Sections

- Basic Data Mining Tutorial

  This tutorial introduces basic concepts, such as how to create a project and how to build mining structures and mining models.

- Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

  This tutorial contains a number of independent lessons, each introducing you to a different model type. Each lesson walks you through the process of creating a model, exploring the model, and then customizing the model and creating prediction queries.

See Also

Data Mining Solutions
Data Mining Tools
Data Mining Projects

Bike Buyer DMX Tutorial

2/11/2019 • 5 minutes to read • Edit Online

In this tutorial, you will learn how create, train, and explore mining models by using the Data Mining Extensions (DMX) query language. You will then use these mining models to create predictions that determine whether a customer will purchase a bicycle.

The mining models will be created from the data contained in the **AdventureWorksDW2012** sample database, which stores data for the fictitious company Adventure Works Cycles. Adventure Works Cycles is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European, and Asian commercial markets. Its base operation is located in Bothell, Washington, with 290 employees, and it has several regional sales teams located throughout their international market base.

Tutorial Scenario

Adventure Works Cycles has decided to extend their data analysis by creating a custom application that uses data mining functionality. Their goal for the custom application is to be able to:

- Take as input specific characteristics about a potential customer and predict whether they will buy a bicycle.
- Take as input a list of potential customers, as well as characteristics about the customers, and predict which ones will buy a bicycle.

In the first case, customer data is provided by a customer registration page, and in the second case, a list of potential customers is provided by the Adventure Works Cycles marketing department.

In addition, the marketing department has asked for the ability to group existing customers into categories based on characteristics such as where they live, the number of children they have, and their commute distance. They want to see whether these clusters can be used to help target specific kinds of customers. This will require an additional mining model.

Microsoft SQL Server Analysis Services provides several tools that can be used to accomplish these tasks:

- The DMX query language
- The Microsoft Decision Trees Algorithm and the Microsoft Clustering Algorithm
- Query Editor in SQL Server Management Studio

Data Mining Extensions (DMX) is a query language provided by Analysis Services that you can use to create and work with mining models. The Microsoft Decision Trees algorithm creates models that can be used to predict whether someone will purchase a bicycle. The resulting model can take an individual customer or a table of customers as an input. The Microsoft Clustering algorithm can create groupings of customers based on shared characteristics. The goal of this tutorial is to provide the DMX scripts that will be used in the custom application.

For more information: Data Mining Solutions

Mining Structure and Mining Models

Before you begin to create DMX statements, it is important to understand the main objects that Analysis Services uses to create mining models. The mining structure is a data structure that defines the data domain from which mining models are built. A single mining structure can contain multiple mining models that share the same domain. A mining model applies a mining model algorithm to the data, which is represented by a mining

structure.

The building blocks of the mining structure are the mining structure columns, which describe the data that the data source contains. These columns contain information such as data type, content type, and how the data is distributed.

Mining models must contain the key column described in the mining structure, as well as a subset of the remaining columns. The mining model defines the usage for each column and defines the algorithm that is used to create the mining model. For example, in DMX you can specify that a column is a Key column or a PREDICT column. If a column is left unspecified, it is assumed to be an input column.

In DMX, there are two ways to create mining models. You can either create the mining structure and associated mining model together by using the CREATE MINING MODEL statement, or you can first create a mining structure by using the CREATE MINING STRUCTURE statement, and then add a mining model to the structure by using the ALTER STRUCTURE statement. These methods are described in the following table.

CREATE MINING MODEL
Use this statement to create a mining structure and associated mining model together using the same name. The mining model name is appended with "Structure" to differentiate it from the mining structure. This statement is useful if you are creating a mining structure that will contain a single mining model.

For more information, see CREATE MINING MODEL (DMX).

ALTER MINING STRUCTURE
Use this statement to add a mining model to a mining structure that already exists on the server. This statement is useful if you want to create a mining structure that contains several different mining models. There are several reasons that you would want to add more than one mining model in a single mining structure. For example, you might create several mining models that use different algorithms to see which algorithm works best. You might create several mining models that use the same algorithm, but with a parameter set differently for each mining model to find the best setting for the parameter.

For more information, see ALTER MINING STRUCTURE (DMX).

Because you will create a mining structure that contains several mining models, you will use the second method in this tutorial.

For More Information

Data Mining Extensions (DMX) Reference, Understanding the DMX Select Statement, Structure and Usage of DMX Prediction Queries

What You Will Learn

This tutorial is divided into the following lessons:

Lesson 1: Creating the Bike Buyer Mining Structure
In this lesson, you will learn how to use the `CREATE` statement to create mining structures.

Lesson 2: Adding Mining Models to the Bike Buyer Mining Structure
In this lesson, you will learn how to use the `ALTER` statement to add mining models to a mining structure.

Lesson 3: Processing the Bike Buyer Mining Structure
In this lesson you will learn how to use the `INSERT INTO` statement to process mining structures and their associated mining models.

Lesson 4: Browsing the Bike Buyer Mining Models
In this lesson, you will learn how to use the `SELECT` statement to explore the content of the mining models.

Lesson 5: Executing Prediction Queries

In this lesson, you will learn how to use the `PREDICTION JOIN` statement to create predictions against mining models.

Requirements

Before doing this tutorial, make sure that the following are installed:

- Microsoft SQL Server
- Microsoft SQL Server 2005 Analysis Services (SSAS), SQL Server 2008 Analysis Services (SSAS), SQL Server 2014 Analysis Services (SSAS), or SQL Server Analysis Services
- The **AdventureWorksDW2012** database. By default, the sample databases are not installed, to enhance security. To install official sample databases for Microsoft SQL Server, visit the Microsoft SQL Sample Databases page and select the databases that you want to install..

> **NOTE**
> When you review tutorials, we recommend that you add **Next topic** and **Previous topic** buttons to the document viewer toolbar.

See Also

Market Basket DMX Tutorial
Basic Data Mining Tutorial

Lesson 1: Creating the Bike Buyer Mining Structure

2/11/2019 • 4 minutes to read • Edit Online

In this lesson, you will create a mining structure that allows you to predict whether a potential customer of Adventure Works Cycles will purchase a bicycle. If you are unfamiliar with mining structures and their role in data mining, see Mining Structures (Analysis Services - Data Mining).

The Bike Buyer mining structure that you will create in this lesson supports adding mining models based on the Microsoft Clustering AlgorithmMicrosoft Decision Trees Algorithm. In later lessons, you will use the clustering mining models to explore the different ways in which customers can be grouped, and will use decision tree mining models to predict whether or not a potential customer will purchase a bicycle.

CREATE MINING STRUCTURE Statement

To create a mining structure, you use the CREATE MINING STRUCTURE (DMX) statement. The code in the statement can be broken into the following parts:

- Naming the structure.
- Defining the key column.
- Defining the mining columns.
- Defining an optional testing data set.

The following is a generic example of the CREATE MINING STRUCTURE statement:

```
CREATE MINING STRUCTURE [<mining structure name>]
(
    <key column>,
    <mining structure columns>
)
WITH HOLDOUT (<holdout specifier>)
```

The first line of the code defines the name of the structure:

```
CREATE MINING STRUCTURE [<mining structure name>]
```

For information about naming an object in Data Mining Extensions (DMX), see Identifiers (DMX).

The next line of the code defines the key column for the mining structure, which uniquely identifies an entity in the source data:

```
<key column>,
```

In the mining structure you will create, the customer identifier, `CustomerKey`, defines an entity in the source data.

The next line of the code is used to define the mining columns that will be used by the mining models associated with the mining structure:

```
<mining structure columns>
```

You can use the DISCRETIZE function within <mining structure columns> to discretize continuous columns by using the following syntax:

```
DISCRETIZE(<method>,<number of buckets>)
```

For more information about discretizing columns, see Discretization Methods (Data Mining). For more information about the types of mining structure columns that you can define, see Mining Structure Columns.

The final line of the code defines an optional partition in the mining structure:

```
WITH HOLDOUT (<holdout specifier>)
```

You specify some portion of the data to use for testing mining models that are related to the structure, and the remaining data is used for training the models. By default, Analysis Services creates a test data set that contains 30 percent of all case data. You will add the specification that the test data set should contain 30 percent of the cases up to a maximum of 1000 cases. If 30 percent of the cases is less than 1000, the test data set will contain the smaller amount.

Lesson Tasks

You will perform the following tasks in this lesson:

- Create a new blank query.
- Alter the query to create the mining structure.
- Execute the query.

Creating the Query

The first step is to connect to an instance of Analysis Services and create a new DMX query in SQL Server Management Studio.

To create a new DMX query in SQL Server Management Studio

1. Open SQL Server Management Studio.
2. In the **Connect to Server** dialog box, for **Server type**, select **Analysis Services**. In **Server name**, type `LocalHost`, or type the name of the instance of Analysis Services that you want to connect to for this lesson. Click **Connect**.
3. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX** to open the **Query Editor** and a new, blank query.

Altering the Query

The next step is to modify the CREATE MINING STRUCTURE statement described above to create the Bike Buyer mining structure.

To customize the CREATE MINING STRUCTURE statement

1. In the Query Editor, copy the generic example of the CREATE MINING STRUCTURE statement into the blank query.
2. Replace the following:

   ```
   [<mining structure>]
   ```

   with:

```
[Bike Buyer]
```

3. Replace the following:

```
<key column>
```

with:

```
CustomerKey LONG KEY
```

4. Replace the following:

```
<mining structure columns>
```

with:

```
[Age] LONG DISCRETIZED(Automatic,10),
[Bike Buyer] LONG DISCRETE,
[Commute Distance] TEXT DISCRETE,
[Education] TEXT DISCRETE,
[Gender] TEXT DISCRETE,
[House Owner Flag] TEXT DISCRETE,
[Marital Status] TEXT DISCRETE,
[Number Cars Owned] LONG DISCRETE,
[Number Children At Home] LONG DISCRETE,
[Occupation] TEXT DISCRETE,
[Region] TEXT DISCRETE,
[Total Children]LONG DISCRETE,
[Yearly Income] DOUBLE CONTINUOUS
```

5. Replace the following:

```
WITH HOLDOUT (holdout specifier>)
```

with:

```
WITH HOLDOUT (30 PERCENT or 1000 CASES)
```

The complete mining structure statement should now be as follows:

```
CREATE MINING STRUCTURE [Bike Buyer]
(
   [Customer Key] LONG KEY,
   [Age]LONG DISCRETIZED(Automatic,10),
   [Bike Buyer] LONG DISCRETE,
   [Commute Distance] TEXT DISCRETE,
   [Education] TEXT DISCRETE,
   [Gender] TEXT DISCRETE,
   [House Owner Flag] TEXT DISCRETE,
   [Marital Status] TEXT DISCRETE,
   [Number Cars Owned]LONG DISCRETE,
   [Number Children At Home]LONG DISCRETE,
   [Occupation] TEXT DISCRETE,
   [Region] TEXT DISCRETE,
   [Total Children]LONG DISCRETE,
   [Yearly Income] DOUBLE CONTINUOUS
)
WITH HOLDOUT (30 PERCENT or 1000 CASES)
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.
7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Bike Buyer Structure.dmx`.

Executing the Query

The final step is to execute the query. After a query is created and saved, it needs to be executed. That is, the statement needs to be run in order to create the mining structure on the server. For more information about executing queries in Query Editor, see Database Engine Query Editor (SQL Server Management Studio).

To execute the query

1. In Query Editor, on the toolbar, click **Execute**.

   The status of the query is displayed in the **Messages** tab at the bottom of Query Editor after the statement finishes executing. Messages should display:

   ```
   Executing the query
   Execution complete
   ```

   A new structure named **Bike Buyer** now exists on the server.

In the next lesson, you will add mining models to the structure you just created.

Next Lesson

Lesson 2: Adding Mining Models to the Bike Buyer Mining Structure

Lesson 2: Adding Mining Models to the Bike Buyer Mining Structure

2/11/2019 • 5 minutes to read • Edit Online

In this lesson, you will add two mining models to the Bike Buyer mining structure that you created Lesson 1: Creating the Bike Buyer Mining Structure. These mining models will allow you to explore the data using one model, and to create predictions using another.

To explore how potential customers can be categorized by their characteristics, you will create a mining model based on the Microsoft Clustering Algorithm. In a later lesson, you will explore how this algorithm finds clusters of customers who share similar characteristics. For example, you might find that certain customers tend to live close to each other, commute by bicycle, and have similar education backgrounds. You can use these clusters to better understand how different customers are related, and to use the information to create a marketing strategy that targets specific customers.

To predict whether a potential customer is likely to buy a bicycle, you will create a mining model based on the Microsoft Decision Trees Algorithm. This algorithm looks through the information that is associated with each potential customer, and finds characteristics that are useful in predicting if they will buy a bicycle. It then compares the values of the characteristics of previous bike buyers against new potential customers to determine whether the new potential customers are likely to buy a bicycle.

ALTER MINING STRUCTURE Statement

In order to add a mining model to the mining structure, you use the ALTER MINING STRUCTURE (DMX) statement. The code in the statement can be broken into the following parts:

- Identifying the mining structure
- Naming the mining model
- Defining the key column
- Defining the input and predictable columns
- Identifying the algorithm and parameter changes

The following is a generic example of the ALTER MINING MODEL statement:

```
ALTER MINING STRUCTURE [<mining structure name>]
ADD MINING MODEL [<mining model name>]
(
    [<key column>],
    <mining model columns>,
) USING <algorithm name>( <algorithm parameters> )
WITH FILTER (<expression>)
```

The first line of the code identifies the existing mining structure to which the mining models will be added:

```
ALTER MINING STRUCTURE [<mining structure name>]
```

The next line of the code names the mining model that will be added to the mining structure:

```
ADD MINING MODEL [<mining model name>]
```

For information about naming an object in DMX, see Identifiers (DMX).

The next lines of the code define columns from the mining structure that will be used by the mining model:

```
[<key column>],
<mining model columns>
```

You can only use columns that already exist in the mining structure, and the first column in the list must be the key column from the mining structure.

The next line of the code defines the mining algorithm that generates the mining model and the algorithm parameters that you can set on the algorithm:

```
) USING <algorithm name>( <algorithm parameters> )
```

For more information about the algorithm parameters that you can adjust, see Microsoft Decision Trees Algorithm and Microsoft Clustering Algorithm.

You can specify that a column in the mining model be used for prediction by using the following syntax:

```
<mining model column> PREDICT
```

The final line of the code, which is optional, defines a filter that is applied when training and testing the model. For more information about how to apply filters to mining models, see Filters for Mining Models (Analysis Services - Data Mining).

Lesson Tasks

You will perform the following tasks in this lesson:

- Add a decision tree mining model to the Bike Buyer structure by using the Microsoft Decision Trees algorithm
- Add a clustering mining model to the Bike Buyer structure by using the Microsoft Clustering algorithm
- Because you want to see results for all cases, you will not yet add a filter to either model.

Adding a Decision Tree Mining Model to the Structure

The first step is to add a mining model based on the Microsoft Decision Trees algorithm.

To add a decision tree mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX** to open Query Editor and a new, blank query.
2. Copy the generic example of the ALTER MINING STRUCTURE statement into the blank query.
3. Replace the following:

   ```
   <mining structure name>
   ```

   with:

```
[Bike Buyer]
```

4. Replace the following:

```
<mining model name>
```

with:

```
Decision Tree
```

5. Replace the following:

```
<mining model columns>,
```

with:

```
(
   CustomerKey,
   [Age],
   [Bike Buyer] PREDICT,
   [Commute Distance],
   [Education],
   [Gender],
   [House Owner Flag],
   [Marital Status],
   [Number Cars Owned],
   [Number Children At Home],
   [Occupation],
   [Region],
   [Total Children],
   [Yearly Income]
```

In this case, the `[Bike Buyer]` column has been designated as the PREDICT column.

6. Replace the following:

```
USING <algorithm name>( <algorithm parameters> )
```

with:

```
Using Microsoft_Decision_Trees
WITH DRILLTHROUGH
```

The WITH DRILLTHROUGH statement allows you to explore the cases that were used to build the mining model.

The resulting statement should now be as follows:

```
ALTER MINING STRUCTURE [Bike Buyer]
ADD MINING MODEL [Decision Tree]
(
   CustomerKey,
   [Age],
   [Bike Buyer] PREDICT,
   [Commute Distance],
   [Education],
   [Gender],
   [House Owner Flag],
   [Marital Status],
   [Number Cars Owned],
   [Number Children At Home],
   [Occupation],
   [Region],
   [Total Children],
   [Yearly Income]
) USING Microsoft_Decision_Trees
WITH DRILLTHROUGH
```

7. On the **File** menu, click **Save DMXQuery1.dmx As**.
8. In the **Save As** dialog box, browse to the appropriate folder, and name the file `DT_Model.dmx`.
9. On the toolbar, click the **Execute** button.

Adding a Clustering Mining Model to the Structure

You can now add a mining model to the Bike Buyer mining structure based on the Microsoft Clustering algorithm. Because the clustering mining model will use all the columns defined in the mining structure, you can use a shortcut to add the model to the structure by omitting the definition of the mining columns.

To add a Clustering mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX** to open Query Editor opens and a new, blank query.
2. Copy the generic example of the ALTER MINING STRUCTURE statement into the blank query.
3. Replace the following:

   ```
   <mining structure name>
   ```

   with:

   ```
   [Bike Buyer]
   ```

4. Replace the following:

   ```
   <mining model>
   ```

   with:

   ```
   Clustering Model
   ```

5. Delete the following:

```
(
    [<key column>],
    <mining model columns>,
)
```

6. Replace the following:

```
USING <algorithm name>( <algorithm parameters> )
```

with:

```
USING Microsoft_Clustering
```

The complete statement should now be as follows:

```
ALTER MINING STRUCTURE [Bike Buyer]
ADD MINING MODEL [Clustering]
USING Microsoft_Clustering
```

7. On the **File** menu, click **Save DMXQuery1.dmx As**.
8. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Clustering_Model.dmx`.
9. On the toolbar, click the **Execute** button.

In the next lesson, you will process the models and the mining structure.

Next Lesson

Lesson 3: Processing the Bike Buyer Mining Structure

Lesson 3: Processing the Bike Buyer Mining Structure

2/11/2019 • 3 minutes to read • Edit Online

In this lesson, you will use the INSERT INTO statement and the vTargetMail view from the **AdventureWorksDW2012** sample database to process the mining structures and mining models that you created in Lesson 1: Creating the Bike Buyer Mining Structure and Lesson 2: Adding Mining Models to the Bike Buyer Mining Structure.

When you process a mining structure, Analysis Services reads the source data and builds the structures that support mining models. When you process a mining model, the data defined by the mining structure is passed through the data mining algorithm that you choose. The algorithm searches for trends and patterns, and then stores this information in the mining model. The mining model, therefore, does not contain the actual source data, but instead contains the information that was discovered by the algorithm. For more information about processing mining models, see Processing Requirements and Considerations (Data Mining).

You need to reprocess a mining structure only if you change a structure column or change the source data. If you add a mining model to a mining structure that has already been processed, you can use the INSERT INTO MINING MODEL statement to train the new mining model.

Train Structure Template

In order to train the mining structure and its associated mining models, use the INSERT INTO (DMX) statement. The code in the statement can be broken into the following parts:

- Identifying the mining structure
- Listing the columns in the mining structure
- Defining the training data

The following is a generic example of the INSERT INTO statement:

```
INSERT INTO MINING STRUCTURE [<mining structure name>]
(
   <mining structure columns>
)
OPENQUERY([<datasource>],'<SELECT statement>')
```

The first line of the code identifies the mining structure that you will train:

```
INSERT INTO MINING STRUCTURE [<mining structure name>]
```

The next line of the code specifies the columns that are defined by the mining structure. You must list each column in the mining structure, and each column must map to a column contained within the source query data.

```
(
   <mining structure columns>
)
```

The final line of the code defines the data that will be used to train the mining structure:

```
OPENQUERY([<datasource>],'<SELECT statement>')
```

In this lesson, you use `OPENQUERY` to define the source data. For information about other methods of defining the source query, see <source data query>.

Lesson Tasks

You will perform the following task in this lesson:

- Process the Bike Buyer mining structure

Processing the Predictive Mining Structure

To process the mining structure by using INSERT INTO

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the INSERT INTO statement into the blank query.

3. Replace the following:

   ```
   [<mining structure name>]
   ```

   with:

   ```
   Bike Buyer
   ```

4. Replace the following:

   ```
   <mining structure columns>
   ```

   with:

   ```
   [Customer Key],
   [Age],
   [Bike Buyer],
   [Commute Distance],
   [Education],
   [Gender],
   [House Owner Flag],
   [Marital Status],
   [Number Cars Owned],
   [Number Children At Home],
   [Occupation],
   [Region],
   [Total Children],
   [Yearly Income]
   ```

5. Replace the following:

   ```
   OPENQUERY([<datasource>],'<SELECT statement>')
   ```

   with:

```
OPENQUERY([Adventure Works DW],
   'SELECT CustomerKey, Age, BikeBuyer,
         CommuteDistance,EnglishEducation,
         Gender,HouseOwnerFlag,MaritalStatus,
         NumberCarsOwned,NumberChildrenAtHome,
         EnglishOccupation,Region,TotalChildren,
         YearlyIncome
    FROM dbo.vTargetMail')
```

The OPENQUERY statement references the Adventure Works DW Multidimensional 2012 data source to access the view vTargetMail. The view contains the source data that will be used to train the mining models.

The complete statement should now be as follows:

```
INSERT INTO MINING STRUCTURE [Bike Buyer]
(
   [Customer Key],
   [Age],
   [Bike Buyer],
   [Commute Distance],
   [Education],
   [Gender],
   [House Owner Flag],
   [Marital Status],
   [Number Cars Owned],
   [Number Children At Home],
   [Occupation],
   [Region],
   [Total Children],
   [Yearly Income]
)
OPENQUERY([Adventure Works DW],
   'SELECT CustomerKey, Age, BikeBuyer,
         CommuteDistance,EnglishEducation,
         Gender,HouseOwnerFlag,MaritalStatus,
         NumberCarsOwned,NumberChildrenAtHome,
         EnglishOccupation,Region,TotalChildren,
         YearlyIncome
    FROM dbo.vTargetMail')
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.
7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Process Bike Buyer Structure.dmx`.
8. On the toolbar, click the **Execute** button.

In the next lesson, you will explore content in the mining models you added to the mining structure in this lesson.

Next Lesson

Lesson 4: Browsing the Bike Buyer Mining Models

Lesson 4: Browsing the Bike Buyer Mining Models

2/11/2019 • 5 minutes to read • Edit Online

In this lesson, you will use the SELECT (DMX) statement to explore the content in the decision tree and clustering mining models that you created in Lesson 2: Adding Mining Models to the Predictive Mining Structure.

The columns contained in a mining model are not the columns defined by the mining structure, but instead are a specific set of columns that describe the trends and patterns that are found by the algorithm. These mining model columns are described in the DMSCHEMA_MINING_MODEL_CONTENT Rowset schema rowset. For example, the MODEL_NAME column in the content schema rowset contains the name of the mining model. For a clustering mining model, the NODE_CAPTION column contains the name of each cluster, and the NODE_DESCRIPTION column contains a description of the characteristics of each cluster. You can browse these columns by using the SELECT FROM <model>.CONTENT statement in DMX. You can also use this statement to explore the data that was used to create the mining model. Drillthrough must be enabled on the mining structure in order to use this statement. For more information about the statement, see SELECT FROM <model>.CASES (DMX).

You can also return all the states of a discrete column by using the SELECT DISTINCT statement. For example, if you perform this operation on a gender column, the query will return `male` and `female`.

Lesson Tasks

You will perform the following tasks in this lesson:

- Explore the content contained within the mining models
- Return the cases from the source data that was used to train the mining models
- Explore the different states available for a specific discrete column

Returning the Content of a Mining Model

In this lesson, you use the SELECT FROM <model>.CONTENT (DMX) statement to return the contents of the clustering model.

The following is a generic example of the SELECT FROM <model>.CONTENT statement:

```
SELECT <select list> FROM [<mining model>].CONTENT
WHERE <where clause>
```

The first line of the code defines the columns to return from the mining model content, and the mining model they are associated with:

```
SELECT <select list> FROM [<mining model].CONTENT
```

The .CONTENT clause next to the name of the mining model specifies that you are returning content from the mining model. For more information about the columns contained in the mining model, see DMSCHEMA_MINING_MODEL_CONTENT Rowset.

You can optionally use the final line of the code to filter the results returned by the statement:

```
WHERE <where clause>
```

For example, if you want to restrict the results of the query to only the clusters that contain a high number of cases, you can add the following WHERE clause to the SELECT statement:

```
WHERE NODE_SUPPORT > 100
```

For more information about using the WHERE statement, see SELECT (DMX).

To return the content of the clustering mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the SELECT FROM <model>.CONTENT statement into the blank query.

3. Replace the following:

   ```
   <select list>
   ```

   with:

   ```
   *
   ```

   You can also replace * with a list of any of the columns contained within the DMSCHEMA_MINING_MODEL_CONTENT Rowset.

4. Replace the following:

   ```
   [<mining model>]
   ```

   with:

   ```
   [Clustering]
   ```

   The complete statement should now be as follows:

   ```
   SELECT * FROM [Clustering].CONTENT
   ```

5. On the **File** menu, click **Save DMXQuery1.dmx As**.

6. In the **Save As** dialog box, browse to the appropriate folder, and name the file `SELECT_CONTENT.dmx`.

7. On the toolbar, click the **Execute** button.

   The query returns the content of the mining model.

Use Drillthrough

The next step is to use the drillthrough statement to return a sampling of the cases that were used to train the decision tree mining model. In this lesson, you use the SELECT FROM <model>.CASES (DMX) statement to return the contents of the decision tree model.

The following is a generic example of the SELECT FROM <model>.CASES statement:

```
SELECT <select list>
FROM [<mining model>].CASES
WHERE IsInNode('<node id>')
```

The first line of the code defines the columns to return from the source data, and the mining model they are contained within:

```
SELECT <select list> FROM [<mining model>].CASES
```

The .CASES clause specifies that you are performing a drillthrough query. In order to use drillthrough you must enable drillthrough when you create the mining model.

The final line of the code is optional and specifies the node in the mining model that you are requesting cases from:

```
WHERE IsInNode('<node id>')
```

For more information about using the WHERE statement with IsInNode, see SELECT FROM <model>.CASES (DMX).

To return the cases that were used to train the mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the SELECT FROM <model>.CASES statement into the blank query.

3. Replace the following:

   ```
   <select list>
   ```

   with:

   ```
   *
   ```

   You can also replace * with a list of any of the columns contained within the source data (such as [Bike Buyer]).

4. Replace the following:

   ```
   [<mining model>]
   ```

   with:

   ```
   [Decision Tree]
   ```

   The complete statement should now be as follows:

   ```
   SELECT *
   FROM [Decision Tree].CASES
   ```

5. On the **File** menu, click **Save DMXQuery1.dmx As**.
6. In the **Save As** dialog box, browse to the appropriate folder, and name the file `SELECT_DRILLTHROUGH.dmx`.
7. On the toolbar, click the **Execute** button.

   The query returns the source data that was used to train the decision tree mining model.

Return the States of a Discrete Mining Model Column

The next step is to use the SELECT DISTINCT statement to return the different possible states in the specified mining model column.

The following is a generic example of the SELECT DISTINCT statement:

```
SELECT DISTINCT [<column>]
FROM [<mining model>]
```

The first line of the code defines the mining model columns for which the states are returned:

```
SELECT DISTINCT [<column>]
```

You must include DISTINCT in order to return all of the states of the column. If you exclude DISTINCT, then the full statement becomes a shortcut for a prediction and returns the most likely state of the specified column. For more information, see SELECT (DMX).

To return the states of a discrete column

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.
2. Copy the generic example of the SELECT Distinct statement into the blank query.
3. Replace the following:

   ```
   [<column,name>
   ```

   with:

   ```
   [Bike Buyer]
   ```
4. Replace the following:

   ```
   [<mining model>]
   ```

   with:

   ```
   [Decision Tree]
   ```

   The complete statement should now be as follows:

   ```
   SELECT DISTINCT [Bike Buyer]
   FROM [Decision Tree]
   ```

5. On the **File** menu, click **Save DMXQuery1.dmx As**.
6. In the **Save As** dialog box, browse to the appropriate folder, and name the file `SELECT_DISCRETE.dmx`.
7. On the toolbar, click the **Execute** button.

   The query returns the possible states of the Bike Buyer column.

In the next lesson, you will predict whether potential customers will be bike buyers by using the decision tree mining model.

Next Lesson

Lesson 5: Executing Prediction Queries

Lesson 5: Executing Prediction Queries

2/11/2019 • 7 minutes to read • Edit Online

In this lesson, you will use the SELECT FROM <model> PREDICTION JOIN (DMX) form of the SELECT statement to create two different types of predictions based on the decision tree model you created in Lesson 2: Adding Mining Models to the Association Mining Structure. These prediction types are defined below.

Singleton Query
Use a singleton query to provide ad hoc values when making predictions. For example, you can determine whether a single customer is likely to be a bike buyer, by passing inputs to the query such as the commute distance, the area code, or the number of children of the customer. The singleton query returns a value that indicates how likely the person is to purchase a bicycle based on those inputs.

Batch Query
Use a batch query to determine who in a table of potential customers is likely to purchase a bicycle. For example, if your marketing department provides you with a list of customers and customer attributes, then you can use a batch prediction to determine who from the table is likely to purchase a bicycle.

The SELECT FROM <model> PREDICTION JOIN (DMX) form of the SELECT statement contains three parts:

- A list of the mining model columns and prediction functions that are returned in the results. The results can also contain input columns from the source data.
- The source query defining the data that is being used to create a prediction. For example, in a batch query this could be a list of customers.
- A mapping between the mining model columns and the source data. If these names match, then you can use NATURAL syntax and leave out the column mappings.

You can further enhance the query by using prediction functions. Prediction functions provide additional information, such as the probability of a prediction occurring, and provide support for the prediction in the training dataset. For more information about prediction functions, see Functions (DMX).

The predictions in this tutorial are based on the ProspectiveBuyer table in the **AdventureWorksDW2012** sample database. The ProspectiveBuyer table contains a list of potential customers and their associated characteristics. The customers in this table are independent of the customers that were used to create the decision tree mining model.

You can also create predictions by using the prediction query builder in SQL Server Data Tools (SSDT).

Lesson Tasks

You will perform the following tasks in this lesson:

- Create a singleton query to determine whether a specific customer is likely to purchase a bicycle.
- Create a batch query to determine which customers, listed in a table of customers, are likely to purchase a bicycle.

Singleton Query

The first step is to use the SELECT FROM <model> PREDICTION JOIN (DMX) in a singleton prediction query. The following is a generic example of the singleton statement:

```
SELECT <select list> FROM [<mining model name>]
NATURAL PREDICTION JOIN
(SELECT '<value>' AS [<column>], ...)
AS [<input alias>]
```

The first line of the code defines the columns from the mining model that the query should return, and specifies the mining model that is used to generate the prediction:

```
SELECT <select list> FROM [<mining model name>]
```

The next lines of the code define the characteristics of the customer that you use to create a prediction:

```
NATURAL PREDICTION JOIN
(SELECT '<value>' AS [<column>], ...)
AS [<input alias>]
ORDER BY <expression>
```

If you specify NATURAL PREDICTION JOIN, the server matches each column from the model to a column from the input, based on column names. If column names do not match, the columns are ignored.

To create a singleton prediction query

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the singleton statement into the blank query.

3. Replace the following:

   ```
   <select list>
   ```

   with:

   ```
   [Bike Buyer] AS Buyer, PredictHistogram([Bike Buyer]) AS Statistics
   ```

   The AS statement is used to alias columns returned by the query. The PredictHistogram function returns statistics about the prediction, including the probability and the support. For more information about the functions that can be used in a prediction statement, see Functions (DMX).

4. Replace the following:

   ```
   [<mining model>]
   ```

   with:

   ```
   [Decision Tree]
   ```

5. Replace the following:

   ```
   (SELECT '<value>' AS [<column name>], ...)  AS t
   ```

   with:

```
(SELECT 35 AS [Age],
  '5-10 Miles' AS [Commute Distance],
  '1' AS [House Owner Flag],
  2 AS [Number Cars Owned],
  2 AS [Total Children]) AS t
```

The complete statement should now be as follows:

```
SELECT
   [Bike Buyer] AS Buyer,
   PredictHistogram([Bike Buyer]) AS Statistics
FROM
   [Decision Tree]
NATURAL PREDICTION JOIN
(SELECT 35 AS [Age],
   '5-10 Miles' AS [Commute Distance],
   '1' AS [House Owner Flag],
   2 AS [Number Cars Owned],
   2 AS [Total Children]) AS t
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.
7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Singleton_Query.dmx`.
8. On the toolbar, click the **Execute** button.

   The query returns a prediction about whether a customer with the specified characteristics will purchase a bicycle, as well as statistics about that prediction.

Batch Query

The next step is to use the SELECT FROM <model> PREDICTION JOIN (DMX) in a batch prediction query. The following is a generic example of a batch statement:

```
SELECT TOP <number> <select list>
FROM [<mining model name>]
PREDICTION JOIN
OPENQUERY([<datasource>],'<SELECT statement>')
  AS [<input alias>]
ON <on clause, mapping,>
WHERE <where clause, boolean expression,>
ORDER BY <expression>
```

As in the singleton query, the first two lines of the code define the columns from mining model that the query returns, as well as the name of the mining model that is used to generate the prediction. The TOP <number> statement specifies that the query will only return the number or the results specified by <number>.

The next lines of the code define the source data that the predictions are based on:

```
OPENQUERY([<datasource>],'<SELECT statement>')
  AS [<input alias>]
```

You have several options for the method of retrieving the source data, but in this tutorial, you will use OPENQUERY. For more information about the options available, see <source data query>.

The next line defines the mapping between the source columns in the mining model and the columns in the source data:

```
ON <column mappings>
```

The WHERE clause filters the results returned by the prediction query:

```
WHERE <where clause, boolean expression,>
```

The last and optional line of the code specifies the column that the results will be ordered by:

```
ORDER BY <expression> [DESC|ASC]
```

Use ORDER BY in combination with the TOP <number> statement, to filter the results that are returned. For example, in this prediction you will return the top ten bike buyers, ordered by the probability of the prediction being correct. You can use [DESC|ASC] syntax to control the order in which the results are displayed.

To create a batch prediction query

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the batch statement into the blank query.

3. Replace the following:

   ```
   <select list>
   ```

   with:

   ```
   SELECT
     TOP 10
     t.[LastName],
     t.[FirstName],
     [Decision Tree].[Bike Buyer],
     PredictProbability([Bike Buyer])
   ```

   The TOP 10 clause specifies that only the top ten results will be returned by the query. The ORDER BY statement in this query orders the results by the probability of the prediction being correct, so only the ten most likely results will be returned.

4. Replace the following placeholder:

   ```
   [<mining model>]
   ```

   With the name of the model:

   ```
   [Decision Tree]
   ```

5. Replace the following generic OPENQUERY statement:

   ```
   OPENQUERY([<datasource>],'<SELECT statement>')
   ```

   With a statement that references the current Adventureworks data warehouse, such as:

```
OPENQUERY([Adventure Works DW 2014],
  'SELECT
    [LastName],
    [FirstName],
    [MaritalStatus],
    [Gender],
    [YearlyIncome],
    [TotalChildren],
    [NumberChildrenAtHome],
    [Education],
    [Occupation],
    [HouseOwnerFlag],
    [NumberCarsOwned]
  FROM
    [dbo].[ProspectiveBuyer]
  ') AS t
```

6. Replace the following generic syntax:

```
<ON clause, mapping,>
WHERE <where clause, boolean expression,>
ORDER BY <expression>
```

With the column mappings needed for this model and input data set:

```
[Decision Tree].[Marital Status] = t.[MaritalStatus] AND
  [Decision Tree].[Gender] = t.[Gender] AND
  [Decision Tree].[Yearly Income] = t.[YearlyIncome] AND
  [Decision Tree].[Total Children] = t.[TotalChildren] AND
  [Decision Tree].[Number Children At Home] = t.[NumberChildrenAtHome] AND
  [Decision Tree].[Education] = t.[Education] AND
  [Decision Tree].[Occupation] = t.[Occupation] AND
  [Decision Tree].[House Owner Flag] = t.[HouseOwnerFlag] AND
  [Decision Tree].[Number Cars Owned] = t.[NumberCarsOwned]
WHERE [Decision Tree].[Bike Buyer] =1
ORDER BY PredictProbability([Bike Buyer]) DESC
```

Specify `DESC` in order to list the results with the highest probability first.

The complete statement should now be as follows:

```
SELECT
  TOP 10
  t.[LastName],
  t.[FirstName],
  [Decision Tree].[Bike Buyer],
  PredictProbability([Bike Buyer])
FROM
  [Decision Tree]
PREDICTION JOIN
  OPENQUERY([Adventure Works DW 2014],
    'SELECT
      [LastName],
      [FirstName],
      [MaritalStatus],
      [Gender],
      [YearlyIncome],
      [TotalChildren],
      [NumberChildrenAtHome],
      [Education],
      [Occupation],
      [HouseOwnerFlag],
      [NumberCarsOwned]
    FROM
      [dbo].[ProspectiveBuyer]
    ') AS t
ON
  [Decision Tree].[Marital Status] = t.[MaritalStatus] AND
  [Decision Tree].[Gender] = t.[Gender] AND
  [Decision Tree].[Yearly Income] = t.[YearlyIncome] AND
  [Decision Tree].[Total Children] = t.[TotalChildren] AND
  [Decision Tree].[Number Children At Home] = t.[NumberChildrenAtHome] AND
  [Decision Tree].[Education] = t.[Education] AND
  [Decision Tree].[Occupation] = t.[Occupation] AND
  [Decision Tree].[House Owner Flag] = t.[HouseOwnerFlag] AND
  [Decision Tree].[Number Cars Owned] = t.[NumberCarsOwned]
WHERE [Decision Tree].[Bike Buyer] =1
ORDER BY PredictProbability([Bike Buyer]) DESC
```

7. On the **File** menu, click **Save DMXQuery1.dmx As**.
8. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Batch_Prediction.dmx`.
9. On the toolbar, click the **Execute** button.

   The query returns a table containing customer names, a prediction of whether each customer will purchase a bicycle, and the probability of the prediction.

This is the last step in the Bike Buyer tutorial. You now have a set of mining models that you can use to explore similarities between you customers and predict whether potential customers will purchase a bicycle.

To learn how to use DMX in a Market Basket scenario, see Market Basket DMX Tutorial.

Market Basket DMX Tutorial

2/11/2019 • 5 minutes to read • Edit Online

In this tutorial, you will learn how to create, train, and explore mining models by using the Data Mining Extensions (DMX) query language. You will then use these mining models to create predictions that describe which products tend to be purchased at the same time.

The mining models will be created from the data contained in the **AdventureWorksDW2012** sample database, which stores data for the fictitious company Adventure Works Cycles. Adventure Works Cycles is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European, and Asian commercial markets. Its base operation is located in Bothell, Washington, with 290 employees, and it has several regional sales teams are located throughout their international market base.

Tutorial Scenario

Adventure Works Cycles has decided to create a custom application that employs data mining functionality to predict what types of products their customers tend to purchase at the same time. The goal for the custom application is to be able to specify a set of products, and predict what additional products will be purchased with the specified products. Adventure Works Cycles will then use this information to add a "suggest" feature to their website, and also to better organize the way that they present information to their customers.

Microsoft SQL Server Analysis Services provides several tools that can be used to accomplish this task:

- The DMX query language
- The Microsoft Association Algorithm
- Query Editor in SQL Server Management Studio

Data Mining Extensions (DMX) is a query language provided by Analysis Services that you can use to create and work with mining models. The Microsoft Association algorithm creates models that can predict the products that are likely to be purchased together.

The goal of this tutorial is to provide the DMX queries that will be used in the custom application.

For more information: Data Mining Solutions

Mining Structure and Mining Models

Before you begin to create DMX statements, it is important to understand the main objects that Analysis Services uses to create mining models. The *mining structure* is a data structure that defines the data domain from which mining models are built. A single mining structure can contain multiple *mining models* that share the same domain. A mining model applies a mining model algorithm to the data, which is represented by a mining structure.

The building blocks of the mining structure are the mining structure columns, which describe the data that the data source contains. These columns contain information such as data type, content type, and how the data is distributed.

Mining models must contain the key column described in the mining structure, as well as a subset of the remaining columns. The mining model defines the usage for each column and defines the algorithm that is used to create the mining model. For example, in DMX you can specify that a column is a Key column or a PREDICT

column. If a column is left unspecified, it is assumed to be an input column.

In DMX, there are two ways to create mining models. You can either create the mining structure and associated mining model together by using the `CREATE MINING MODEL` statement, or you can first create a mining structure by using the `CREATE MINING STRUCTURE` statement, and then add a mining model to the structure by using the `ALTER STRUCTURE` statement. These methods are described below.

`CREATE MINING MODEL`
Use this statement to create a mining structure and associated mining model together using the same name. The mining model name is appended with "Structure" to differentiate it from the mining structure.

This statement is useful if you are creating a mining structure that will contain a single mining model.

For more information, see CREATE MINING MODEL (DMX).

CREATE MINING STRUCTURE
Use this statement to create a new mining structure without any models.

When you use CREATE MINING STRUCTURE, you can also create a holdout data set that can be used for testing any models that are based on the same mining structure.

For more information, see CREATE MINING STRUCTURE (DMX).

`ALTER MINING STRUCTURE`
Use this statement to add a mining model to a mining structure that already exists on the server.

There are several reasons that you would want to add more than one mining model in a single mining structure. For example, you might create several mining models using different algorithms to see which one works best. Alternatively, you might create several mining models using the same algorithm, but with a parameter set differently for each mining model to find the best setting for that parameter.

For more information, see ALTER MINING STRUCTURE (DMX).

Because you will create a mining structure that contains several mining models, you will use the second method in this tutorial.

For More Information

Data Mining Extensions (DMX) Reference, Understanding the DMX Select Statement, Structure and Usage of DMX Prediction Queries

What You Will Learn

This tutorial is divided into the following lessons:

Lesson 1: Creating the Market Basket Mining Structure
In this lesson, you will learn how to use the `CREATE` statement to create mining structures.

Lesson 2: Adding Mining Models to the Market Basket Mining Structure
In this lesson, you will learn how to use the `ALTER` statement to add mining models to a mining structure.

Lesson 3: Processing the Market Basket Mining Structure
In this lesson, you will learn how to use the `INSERT INTO` statement to process mining structures and their associated mining models.

Lesson 4: Executing Market Basket Predictions
In this lesson, you will learn how to use the `PREDICTION JOIN` statement to create predictions against mining models.

Requirements

Before doing this tutorial, make sure that the following are installed:

- Microsoft SQL Server
- Microsoft SQL Server Analysis Services
- The **AdventureWorksDW2012** database

By default, the sample databases are not installed, to enhance security. To install the official sample databases for Microsoft SQL Server, go to http://www.CodePlex.com/MSFTDBProdSamples or on the Microsoft SQL Server Samples and Community Projects home page in the section Microsoft SQL Server Product Samples. Click **Databases**, then click the **Releases** tab and select the databases that you want.

> **NOTE**
>
> When you review tutorials, we recommend that you add **Next topic** and **Previous topic** buttons to the document viewer toolbar.

See Also

Bike Buyer DMX Tutorial
Basic Data Mining Tutorial
Lesson 3: Building a Market Basket Scenario (Intermediate Data Mining Tutorial)

Lesson 1: Creating the Market Basket Mining Structure

2/11/2019 • 4 minutes to read • Edit Online

In this lesson, you will create a mining structure that allows you to predict what Adventure Works Cycles products a customer tends to purchase at the same time. If you are unfamiliar with mining structures and their role in data mining, see Mining Structures (Analysis Services - Data Mining).

The association mining structure that you will create in this lesson supports adding mining models based on the Microsoft Association Algorithm. In later lessons, you will use the mining models to predict the type of products a customer tends to purchase at the same time, which is called a market basket analysis. For example, you may find that customers tend to buy mountain bikes, bike tires, and helmets at the same time.

In this lesson, the mining structure is defined by using nested tables. Nested tables are used because the data domain that will be defined by the structure is contained within two different source tables. For more information on nested tables, see Nested Tables (Analysis Services - Data Mining).

CREATE MINING STRUCTURE Statement

In order to create a mining structure containing a nested table, you use the CREATE MINING STRUCTURE (DMX) statement. The code in the statement can be broken into the following parts:

- Naming the structure
- Defining the key column
- Defining the mining columns
- Defining the nested table columns

The following is a generic example of the CREATE MINING STRUCTURE statement:

```
CREATE MINING STRUCTURE [<Mining Structure Name>]
(
   <key column>,
   <mining structure columns>,
   <table columns>
   (  <nested key column>,
      <nested mining structure columns> )
)
```

The first line of the code defines the name of the structure:

```
CREATE MINING STRUCTURE [Mining Structure Name]
```

For information about naming an object in DMX, see Identifiers (DMX).

The next line of the code defines the key column for the mining structure, which uniquely identifies an entity in the source data:

```
<key column>
```

The next line of the code is used to define the mining columns that will be used by the mining models associated with the mining structure:

```
<mining structure columns>
```

The next lines of the code define the nested table columns:

```
<table columns>
(  <nested key column>,
   <nested mining structure columns> )
```

For information about the types of mining structure columns that you can define, see Mining Structure Columns.

> **NOTE**
> By default, SQL Server Data Tools (SSDT) creates a 30 percent holdout data set for each mining structure; however, when you use DMX to create a mining structure, you must manually add the holdout data set, if desired.

Lesson Tasks

You will perform the following tasks in this lesson:

- Create a new blank query
- Alter the query to create the mining structure
- Execute the query

Creating the Query

The first step is to connect to an instance of Analysis Services and create a new DMX query in SQL Server Management Studio.

To create a new DMX query in SQL Server Management Studio

1. Open SQL Server Management Studio.
2. In the **Connect to Server** dialog box, for **Server type**, select **Analysis Services**. In **Server name**, type `LocalHost`, or the name of the instance of Analysis Services that you want to connect to for this lesson. Click **Connect**.
3. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

Altering the Query

The next step is to modify the CREATE MINING STRUCTURE statement described above to create the Market Basket mining structure.

To customize the CREATE MINING STRUCTURE statement

1. In Query Editor, copy the generic example of the CREATE MINING STRUCTURE statement into the blank query.

2. Replace the following:

```
[mining structure name]
```

with:

```
[Market Basket]
```

3. Replace the following:

```
<key column>
```

with:

```
OrderNumber TEXT KEY
```

4. Replace the following:

```
<table columns>
(  <nested key column>,
   <nested mining structure columns> )
```

with:

```
[Products] TABLE (
    [Model] TEXT KEY
)
```

The TEXT KEY language specifies that the Model column is the key column for the nested table.

The complete mining structure statement should now be as follows:

```
CREATE MINING STRUCTURE [Market Basket] (
    OrderNumber TEXT KEY,
    [Products] TABLE (
        [Model] TEXT KEY
    )
)
```

5. On the **File** menu, click **Save DMXQuery1.dmx As**.

6. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Market Basket Structure.dmx`.

Executing the Query

The final step is to execute the query. After a query is created and saved, it needs to be executed (that is, the statement needs to be run) in order to create the mining structure on the server. For more information about executing queries in Query Editor, see Database Engine Query Editor (SQL Server Management Studio).

To execute the query

- In Query Editor, on the toolbar, click **Execute**.

  The status of the query is displayed in the **Messages** tab at the bottom of Query Editor after the statement

finishes executing. Messages should display:

```
Executing the query
Execution complete
```

A new structure named **Market Basket** now exists on the server.

In the next lesson, you will add mining models to the Market Basket mining structure you just created.

Next Lesson

Lesson 2: Adding Mining Models to the Market Basket Mining Structure

Lesson 2: Adding Mining Models to the Market Basket Mining Structure

2/11/2019 • 5 minutes to read • Edit Online

In this lesson, you will add two mining models to the Market Basket mining structure that you created in Lesson 1: Creating the Market Basket Mining Structure. These mining models will allow you to create predictions.

To predict the types of products that customers tend to purchase at the same time, you will create two mining models using the Microsoft Association Algorithm and two different values for the *MINIMUM_PROBABILTY* parameter.

MINIMUM_PROBABILTY is a Microsoft Association algorithm parameter that helps to determine the number of rules that a mining model will contain by specifying the minimum probability that a rule must have. For example, setting this value to 0.4 specifies that a rule can be generated only if the combination of products that the rule describes has at least a forty percent probability of occurring.

You will view the effect of changing the *MINIMUM_PROBABILTY* parameter in a later lesson.

ALTER MINING STRUCTURE Statement

To add a mining model that contains a nested table to a mining structure, you use the ALTER MINING STRUCTURE (DMX) statement. The code in the statement can be broken into the following parts:

- Identifying the mining structure
- Naming the mining model
- Defining the key column
- Defining the input and predictable columns
- Defining the nested table columns
- Identifying the algorithm and parameter changes

The following is a generic example of the `ALTER MINING STRUCTURE` statement that adds a mining model to a structure that includes nested table columns:

```
ALTER MINING STRUCTURE [<Mining Structure Name>]
ADD MINING MODEL [<Mining Model Name>]
(
    [<key column>],
    <mining model column> <usage>,
    <table columns>
    (  [<nested key column>],
       <nested mining model columns> )
) USING <algorithm>( <algorithm parameters> )
```

The first line of the code identifies the existing mining structure to which the mining model will be added:

```
ALTER MINING STRUCTURE [<mining structure name>]
```

The next line of the code names the mining model that will be added to the mining structure:

```
ADD MINING MODEL [<mining model name>]
```

For information about naming an object in Data Mining Extensions (DMX), see Identifiers (DMX).

The next lines of the code define the columns in the mining structure that will be used by the mining model:

```
[<key column>],
<mining model columns> <usage>,
```

You can only use columns that already exist in the mining structure.

The first column in the list of mining model columns must be the key column in the mining structure. However, you do not have to type `KEY` after the key column to specify usage. That is because you have already defined the column as a key when you created the mining structure.

The remaining lines specify the usage of the columns in the new mining model. You can specify that a column in the mining model will be used for prediction by using the following syntax:

```
<column name> PREDICT,
```

If you do not specify usage, you do not have to include a data mining structure column in the list. All columns that are used by the referenced data mining structure are automatically available for use by the mining models that are based on that structure. However, the model will not use the columns for training unless you specify the usage.

The last line in the code defines the algorithm and algorithm parameters that will be used to generate the mining model.

```
) USING <algorithm>( <algorithm parameters> )
```

Lesson Tasks

You will perform the following tasks in this lesson:

- Add an association mining model to the structure using the default probability
- Add an association mining model to the structure using a modified probability

Adding an Association Mining Model to the Structure Using the Default MINIMUM_PROBABILITY

The first task is to add a new mining model to the Market Basket mining structure based on the Microsoft Association algorithm using the default value for *MINIMUM_PROBABILITY*.

To add an Association mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

   > **NOTE**
   > To create a DMX query against a specific Analysis Services database, right-click the database instead of the instance.

2. Copy the generic example of the `ALTER MINING STRUCTURE` statement into the blank query.
3. Replace the following:

```
<mining structure name>
```

with:

```
[Market Basket]
```

4. Replace the following:

```
<mining model name>
```

with:

```
[Default Association]
```

5. Replace the following:

```
[<key column>],
<mining model columns>,
<table columns>
(  [<nested key column>],
   <nested mining model columns> )
```

with:

```
OrderNumber,
    [Products] PREDICT (
        [Model]
    )
```

In this case, the `[Products]` table has been designated as the predictable column `.` Also, the `[Model]` column is included in the list of nested table columns because it is the key column of the nested table.

> **NOTE**
> Remember that a nested key is different from a case key. A case key is a unique identifier of the case, whereas the nested key is an attribute that you want to model.

6. Replace the following:

```
USING <algorithm>( <algorithm parameters> )
```

with:

```
Using Microsoft_Association_Rules
```

The resulting statement should now be as follows:

```
ALTER MINING STRUCTURE [Market Basket]
ADD MINING MODEL [Default Association]
(
    OrderNumber,
    [Products] PREDICT (
        [Model]
    )
)
Using Microsoft_Association_Rules
```

7. On the **File** menu, click **Save DMXQuery1.dmx As**.
8. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Default_Association_Model.dmx`.
9. On the toolbar, click the **Execute** button.

Adding an Association Mining Model to the Structure Changing the Default MINIMUM_PROBABILITY

The next task is to add a new mining model to the Market Basket mining structure based on the Microsoft Association algorithm, and change the default value for MINIMUM_PROBABILITY to 0.01. Changing the parameter will cause the Microsoft Association algorithm to create more rules.

To add an Association mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the `ALTER MINING STRUCTURE` statement into the blank query.
3. Replace the following:

   ```
   <mining structure name>
   ```

   with:

   ```
   Market Basket
   ```

4. Replace the following:

   ```
   <mining model name>
   ```

   with:

   ```
   [Modified Association]
   ```

5. Replace the following:

   ```
   <mining model columns>,
   <table columns>
   (  [<nested key column>],
      <nested mining model columns> )
   ```

with:

```
OrderNumber,
[Products] PREDICT (
        [Model]
    )
```

In this case, the `[Products]` table has been designated as the predictable column. Also, the `[MODEL]` column is included in the list because it is the key column in the nested table.

6. Replace the following:

```
USING <algorithm>( <algorithm parameters> )
```

with:

```
USING Microsoft_Association_Rules (Minimum_Probability = 0.1)
```

The resulting statement should now be as follows:

```
ALTER MINING STRUCTURE [Market Basket]
ADD MINING MODEL [Modified Assocation]
(
    OrderNumber,
    [Products] PREDICT (
        [Model]
    )
)
USING Microsoft_Association_Rules (Minimum_Probability = 0.1)
```

7. On the **File** menu, click **Save DMXQuery1.dmx As**.

8. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Modified Association_Model.dmx`.

9. On the toolbar, click the **Execute** button.

In this next lesson you will process the Market Basket mining structure together with its associated mining models.

Next Lesson

Lesson 3: Processing the Market Basket Mining Structure

Lesson 3: Processing the Market Basket Mining Structure

2/11/2019 • 4 minutes to read • Edit Online

In this lesson, you will use the INSERT INTO (DMX) statement and the vAssocSeqLineItems and vAssocSeqOrders from the **AdventureWorksDW2012** sample database to process the mining structures and mining models that you created in Lesson 1: Creating the Market Basket Mining Structure and Lesson 2: Adding Mining Models to the Market Basket Mining Structure.

When you process a mining structure, Analysis Services reads the source data and builds the structures that support mining models. When you process a mining model, the data defined by the mining structure is passed through the data mining algorithm that you chose. The algorithm searches for trends and patterns, and then stores this information in the mining model. The mining model, therefore, does not contain the actual source data, but instead contains the information that was discovered by the algorithm. For more information about processing mining models, see Processing Requirements and Considerations (Data Mining).

You only have to reprocess a mining structure if you change a structure column or change the source data. If you add a mining model to a mining structure that has already been processed, you can use the `INSERT INTO MINING MODEL` statement to train the new mining model on the existing data.

Because the Market Basket mining structure contains a nested table, you will have to define the mining columns to be trained using the nested table structure, and use the `SHAPE` command to define the queries that pull the training data from the source tables.

INSERT INTO Statement

In order to train the Market Basket mining structure and its associated mining models, use the INSERT INTO (DMX) statement. The code in the statement can be broken into the following parts.

- Identifying the mining structure
- Listing the columns in the mining structure
- Defining the training data using `SHAPE`

The following is a generic example of the `INSERT INTO` statement:

```
INSERT INTO MINING STRUCTURE [<mining structure name>]
(
   <mining structure columns>
   [<nested table>]
   ( SKIP, <skipped column> )
)
SHAPE {
  OPENQUERY([<datasource>],'<SELECT statement>') }
APPEND
(
  {OPENQUERY([<datasource>],'<nested SELECT statement>')
}
RELATE [<case key>] TO [<foreign key>]
) AS [<nested table>]
```

The first line of the code identifies the mining structure that you will train:

```
INSERT INTO MINING STRUCTURE [<mining structure name>]
```

The next lines of the code specify the columns that are defined by the mining structure. You must list each column in the mining structure, and each column must map to a column contained within the source query data. You can use `SKIP` to ignore columns that exist in the source data but do not exist in the mining structure. For more information about how to use `SKIP`, see INSERT INTO (DMX).

```
(
   <mining structure columns>
   [<nested table>]
   ( SKIP, <skipped column> )
)
```

The final lines of the code define the data that will be used to train the mining structure. Because the source data is contained within two tables, you will use `SHAPE` to relate the tables.

```
SHAPE {
  OPENQUERY([<datasource>],'<SELECT statement>') }
APPEND
(
  {OPENQUERY([<datasource>],''<nested SELECT statement>'')
}
RELATE [<case key>] TO [<foreign key>]
) AS [<nested table>]
```

In this lesson, you use `OPENQUERY` to define the source data. For information about other methods of defining a query on the source data, see <source data query>.

Lesson Tasks

You will perform the following task in this lesson:

- Process the Market Basket mining structure

Processing the Market Basket Mining Structure

To process the mining structure by using INSERT INTO

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the INSERT INTO statement into the blank query.

3. Replace the following:

   ```
   [<mining structure>]
   ```

   with:

   ```
   Market Basket
   ```

4. Replace the following:

```
<mining structure columns>
[<nested table>]
( SKIP, <skipped column> )
```

with:

```
[OrderNumber],
[Products]
(SKIP, [Model])
```

In the statement, `Products` refers to the Products table defined by the SHAPE statement. `SKIP` is used to ignore the Model column, which exists in the source data as a key, but is not used by the mining structure.

5. Replace the following:

```
SHAPE {
  OPENQUERY([<datasource>],'<SELECT statement>') }
APPEND
(
  {OPENQUERY([<datasource>],'<nested SELECT statement>')
}
RELATE [<case key>] TO [<foreign key>]
) AS [<nested table>]
```

with:

```
SHAPE {
  OPENQUERY([Adventure Works DW],'SELECT OrderNumber
            FROM vAssocSeqOrders ORDER BY OrderNumber')}
APPEND
(
  {OPENQUERY([Adventure Works DW],'SELECT OrderNumber, Model FROM
    dbo.vAssocSeqLineItems ORDER BY OrderNumber, Model')
}
RELATE OrderNumber to OrderNumber
) AS [Products]
```

The source query references the **AdventureWorksDW2012** data source defined in the **AdventureWorksDW2012** sample project. It uses this data source to access the vAssocSeqLineItems and vAssocSeqOrders views. These views contain the source data that will be used to train the mining model. If you have not created this project or these views, see Basic Data Mining Tutorial.

Within the `SHAPE` command, you will use `OPENQUERY` to define two queries. The first query defines the parent table, and the second query defines the nested table. The two tables are related using the OrderNumber column, which exists in both tables.

The complete statement should now be as follows:

```
INSERT INTO MINING STRUCTURE [Market Basket]
(
   [OrderNumber],[Products] (SKIP, [Model])
)
SHAPE {
  OPENQUERY([Adventure Works DW],'SELECT OrderNumber
            FROM vAssocSeqOrders ORDER BY OrderNumber')}
APPEND
(
  {OPENQUERY([Adventure Works DW],'SELECT OrderNumber, Model FROM
    dbo.vAssocSeqLineItems ORDER BY OrderNumber, Model')
}
RELATE OrderNumber to OrderNumber
) AS [Products]
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.
7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Process Market Basket.dmx`.
8. On the toolbar, click the **Execute** button.

After the query has finished running, you can view the patterns and itemsets that were found, view associations, or filter by itemset, probability, or importance. To view this information, in SQL Server Management Studio, right-click the name of the data model, and then click **Browse**.

In the next lesson, you will create several predictions based on the mining models that you added to the Market Basket structure.

Next Lesson

Lesson 4: Executing Market Basket Predictions

Lesson 4: Executing Market Basket Predictions

2/11/2019 • 6 minutes to read • Edit Online

In this lesson, you will use the DMX `SELECT` statement to create predictions based on the association models you created in Lesson 2: Adding Mining Models to the Market Basket Mining Structure. A prediction query is created by using the DMX `SELECT` statement and adding a `PREDICTION JOIN` clause. For more information about the syntax of a prediction join, see SELECT FROM <model> PREDICTION JOIN (DMX).

The **SELECT FROM <model> PREDICTION JOIN** form of the `SELECT` statement contains three parts:

- A list of the mining model columns and prediction functions that are returned in the result set. This list can also contain input columns from the source data.
- A source query that defines the data that is being used to create a prediction. For example, if you are creating many predictions in a batch, the source query could retrieve a list of customers.
- A mapping between the mining model columns and the source data. If the columns names match, you can use the `NATURAL PREDICTION JOIN` syntax and omit the column mappings.

You can enhance the query by using prediction functions. Prediction functions provide additional information, such as the probability of a prediction occurring, or the support for a prediction in the training dataset. For more information about prediction functions, see Functions (DMX).

You can also use the prediction query builder in SQL Server Data Tools (SSDT) to create prediction queries.

Singleton PREDICTION JOIN Statement

The first step is to create a singleton query, by using the **SELECT FROM <model> PREDICTION JOIN** syntax and supplying a single set of values as input. The following is a generic example of the singleton statement:

```
SELECT <select list>
    FROM [<mining model>]
[NATURAL] PREDICTION JOIN
(SELECT '<value>' AS [<column>],
    (SELECT 'value' AS [<nested column>] UNION
        SELECT 'value' AS [<nested column>] ...)
    AS [<nested table>])
AS [<input alias>]
```

The first line of the code defines the columns from the mining model that the query returns, and specifies the name of the mining model used to generate the prediction:

```
SELECT <select list> FROM [<mining model>]
```

The next line of the code indicates the operation to perform. Because you will specify values for each of the columns and type the column names exactly so as to match the model, you can use the `NATURAL PREDICTION JOIN` syntax. However, if the column names were different, you would have to specify mappings between the columns in the model and the columns in the new data by adding an `ON` clause.

```
[NATURAL] PREDICTION JOIN
```

The next lines of the code define the products in the shopping cart that will be used to predict additional products that a customer will add:

```
(SELECT '<value>' AS [<column>],
    (SELECT 'value' AS [<nested column>] UNION
        SELECT 'value' AS [<nested column>] ...)
    AS [<nested table>])
```

Lesson Tasks

You will perform the following tasks in this lesson:

- Create a query that predicts what other items a customer will likely purchase, based on items already existing in their shopping cart. You will create this query by using the mining model with the default *MINIMUM_PROBABILITY*.
- Create a query that predicts what other items a customer will likely purchase based on items already existing in their shopping cart. This query is based on a different model, in which *MINIMUM_PROBABILITY* has been set to 0.01. Because the default value for *MINIMUM_PROBABILITY* in association models is 0.3, the query on this model should return more possible items than the query on the default model.

Create a Prediction by Using a Model with the Default MINIMUM_PROBABILITY

To create an association query

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX** to open the Query Editor.
2. Copy the generic example of the `PREDICTION JOIN` statement into the blank query.
3. Replace the following:

   ```
   <select list>
   ```

   with:

   ```
   PREDICT([Default Association].[Products],INCLUDE_STATISTICS,3)
   ```

   You could just include the column name [Products], but by using the Predict (DMX) function, you can limit the number of products that are returned by the algorithm to three. You can also use `INCLUDE_STATISTICS`, which returns the support, probability, and adjusted probability for each product. These statistics help you rate the accuracy of the prediction.
4. Replace the following:

   ```
   [<mining model>]
   ```

   with:

   ```
   [Default Association]
   ```
5. Replace the following:

```
(SELECT '<value>' AS [<column>],
    (SELECT 'value' AS [<nested column>] UNION
        SELECT 'value' AS [<nested column>] ...)
    AS [<nested table>])
```

with:

```
(SELECT (SELECT 'Mountain Bottle Cage' AS [Model]
  UNION SELECT 'Mountain Tire Tube' AS [Model]
  UNION SELECT 'Mountain-200' AS [Model]) AS [Products]) AS t
```

This statement uses the `UNION` statement to specify three products that must be included in the shopping cart together with the predicted products. The Model column in the `SELECT` statement corresponds to the model column that is contained in the nested products table.

The complete statement should now be as follows:

```
SELECT
  PREDICT([Default Association].[Products],INCLUDE_STATISTICS,3)
From
  [Default Association]
NATURAL PREDICTION JOIN
(SELECT (SELECT 'Mountain Bottle Cage' AS [Model]
  UNION SELECT 'Mountain Tire Tube' AS [Model]
  UNION SELECT 'Mountain-200' AS [Model]) AS [Products]) AS t
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.
7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Association Prediction.dmx`.
8. On the toolbar, click the **Execute** button.

   The query returns a table that contains three products: HL Mountain Tire, Fender Set - Mountain, and ML Mountain Tire. The table lists these returned products in order of probability. The returned product that is most likely to be included in the same shopping cart as the three products specified in the query appears at the top of the table. The two products that follow are the next most likely to be included in the shopping cart. The table also contains statistics describing the accuracy of the prediction.

Create a Prediction by Using a Model with a MINIMUM_PROBABILITY of 0.01

To create an association query

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX** to open the Query Editor.
2. Copy the generic example of the `PREDICTION JOIN` statement into the blank query.
3. Replace the following:

```
<select list>
```

with:

```
PREDICT([Modified Association].[Products],INCLUDE_STATISTICS,3)
```

4. Replace the following:

```
[<mining model>]
```

with:

```
[Modified Association]
```

5. Replace the following:

```
(SELECT '<value>' AS [<column>],
    (SELECT 'value' AS [<nested column>] UNION
        SELECT 'value' AS [<nested column>] ...)
    AS [<nested table>])
```

with:

```
(SELECT (SELECT 'Mountain Bottle Cage' AS [Model]
  UNION SELECT 'Mountain Tire Tube' AS [Model]
  UNION SELECT 'Mountain-200' AS [Model]) AS [Products]) AS t
```

This statement uses the `UNION` statement to specify three products that must be included in the shopping cart together with the predicted products. The `[Model]` column in the `SELECT` statement corresponds to the column in the nested products table.

The complete statement should now be as follows:

```
SELECT
  PREDICT([Modified Association].[Products],INCLUDE_STATISTICS,3)
From
  [Modified Association]
NATURAL PREDICTION JOIN
(SELECT (SELECT 'Mountain Bottle Cage' AS [Model]
  UNION SELECT 'Mountain Tire Tube' AS [Model]
  UNION SELECT 'Mountain-200' AS [Model]) AS [Products]) AS t
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.

7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Modified Association Prediction.dmx`.

8. On the toolbar, click the **Execute** button.

   The query returns a table that contains three products: HL Mountain Tire, Water Bottle, and Fender Set - Mountain. The table lists these products in order of probability. The product that appears at the top of the table is the product that is most likely to be included in the same shopping cart as the three products specified in the query. The remaining products are the next most likely to be included in the shopping cart. The table also contains statistics that describe the accuracy of the prediction.

   You can see from the results of this query that the value of the *MINIMUM_PROBABILITY* parameter affects the results returned by the query.

This is the last step in the Market Basket tutorial. You now have a set of models that you can use to predict the products that customers might purchase at the same time.

To learn how to use DMX in another predictive scenario, see Bike Buyer DMX Tutorial.

See Also

Association Model Query Examples
Data Mining Query Interfaces

Time Series Prediction DMX Tutorial

2/11/2019 • 2 minutes to read • Edit Online

In this tutorial, you will learn how to create a time series mining structure, create three custom time series mining models, and then make predictions by using those models.

The mining models are based on the data contained in the **AdventureWorksDW2012** sample database, which stores data for the fictitious company Adventure Works Cycles. Adventure Works Cycles is a large, multinational manufacturing company.

Tutorial Scenario

Adventure Works Cycles has decided to use data mining to generate sales projections. They have already built some regional forecasting models; for more information, see Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial). However, the Sales Department needs to be able to periodically update the data mining model with new sales data. They also want to customize the models to provide different projections.

Microsoft SQL Server Analysis Services provides several tools that can be used to accomplish this task:

- The Data Mining Extensions (DMX) query language
- The Microsoft Time Series Algorithm
- Query Editor in SQL Server Management Studio

The Microsoft Time Series algorithm creates models that can be used for prediction of time-related data. Data Mining Extensions (DMX) is a query language provided by Analysis Services that you can use to create mining models and prediction queries.

What You Will Learn

This tutorial assumes that you are already familiar with the objects that Analysis Services uses to create mining models. If you have not previously created a mining structure or mining model by using DMX, see Bike Buyer DMX Tutorial.

This tutorial is divided into the following lessons:

Lesson 1: Creating a Time Series Mining Model and Mining Structure
In this lesson, you will learn how to use the `CREATE MINING MODEL` statement to add a new forecasting model and a related mining model.

Lesson 2: Adding Mining Models to the Time Series Mining Structure
In this lesson, you will learn how to use the ALTER MINING STRUCTURE statement to add new mining models to the time series structure. You will also learn how to customize the algorithm used for analyzing a time series.

Lesson 3: Processing the Time Series Structure and Models
In this lesson, you will learn how to train the models by using the `INSERT INTO` statement and populating the structure with data from the **AdventureWorksDW2012** database.

Lesson 4: Creating Time Series Predictions Using DMX
In this lesson, you will learn how to create time series predictions.

Lesson 5: Extending the Time Series Model
In this lesson, you will learn how to use the `EXTEND_MODEL_CASES` parameter to update the model with new data

when you make predictions.

Requirements

Before doing this tutorial, make sure that the following are installed:

- Microsoft SQL Server
- Microsoft SQL Server Analysis Services
- The **AdventureWorksDW2012** database

By default, the sample databases are not installed, to enhance security. To install the official sample databases for Microsoft SQL Server, go to http://www.CodePlex.com/MSFTDBProdSamples or on the Microsoft SQL Server Samples and Community Projects home page in the section Microsoft SQL Server Product Samples. Click **Databases**, then click the **Releases** tab and select the databases that you want.

> **NOTE**
>
> When you review tutorials, we recommend that you add **Next topic** and **Previous topic** buttons to the document viewer toolbar.

See Also

Basic Data Mining Tutorial
Intermediate Data Mining Tutorial (Analysis Services - Data Mining)

Lesson 1: Creating a Time Series Mining Model and Mining Structure

2/11/2019 • 4 minutes to read • Edit Online

In this lesson, you will create a mining model that allows you to predict values over time, based on historical data. When you create the model, the underlying structure will be generated automatically and can be used as the basis for additional mining models.

This lesson assumes that you are familiar with forecasting models and with the requirements of the Microsoft Time Series algorithm. For more information, see Microsoft Time Series Algorithm.

CREATE MINING MODEL Statement

In order to create a mining model directly and automatically generate the underlying mining structure, you use the CREATE MINING MODEL (DMX) statement. The code in the statement can be broken into the following parts:

- Naming the model
- Defining the time stamp
- Defining the optional series key column
- Defining the predictable attribute or attributes

The following is a generic example of the CREATE MINING MODEL statement:

```
CREATE MINING MODEL [<Mining Structure Name>]
(
   <key columns>,
   <predictable attribute columns>
)
USING <algorithm name>([parameter list])
WITH DRILLTHROUGH
```

The first line of the code defines the name of the mining model:

```
CREATE MINING MODEL [Mining Model Name]
```

Analysis Services automatically generates a name for the underlying structure, by appending "_structure" to the model name, which ensures that the structure name is unique from the model name. For information about naming an object in DMX, see Identifiers (DMX).

The next line of the code defines the key column for the mining model, which in the case of a time series model uniquely identifies a time step in the source data. The time step is identified with the `KEY TIME` keywords after the column name and data types. If the time series model has a separate series key, it is identified by using the `KEY` keyword.

```
<key columns>
```

The next line of the code is used to define the columns in the model that will be predicted. You can have multiple predictable attributes in a single mining model. When there are multiple predictable attributes, the Microsoft Time

Series algorithm generates a separate analysis for each series:

```
<predictable attribute columns>
```

Lesson Tasks

You will perform the following tasks in this lesson:

- Create a new blank query
- Alter the query to create the mining model
- Execute the query

Creating the Query

The first step is to connect to an instance of Analysis Services and create a new DMX query in SQL Server Management Studio.

To create a new DMX query in SQL Server Management Studio

1. Open SQL Server Management Studio.
2. In the **Connect to Server** dialog box, for **Server type**, select **Analysis Services**. In **Server name**, type `LocalHost`, or the name of the instance of Analysis Services that you want to connect to for this lesson. Click **Connect**.
3. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

Altering the Query

The next step is to modify the CREATE MINING MODEL statement to create the mining model used for forecasting, together with its underlying mining structure.

To customize the CREATE MINING MODEL statement

1. In Query Editor, copy the generic example of the CREATE MINING MODEL statement into the blank query.
2. Replace the following:

   ```
   [mining model name]
   ```

   with:

   ```
   [Forecasting_MIXED]
   ```

3. Replace the following:

   ```
   <key columns>
   ```

   with:

   ```
   [Reporting Date] DATE KEY TIME,
   [Model Region] TEXT KEY
   ```

The `TIME KEY` keyword indicates that the ReportingDate column contains the time step values used to order the values. Time steps can be dates and times, integers, or any ordered data type, so long as the values are unique and the data is sorted.

The `TEXT` and `KEY` keywords indicate that the ModelRegion column contains an additional series key. You can have only one series key, and the values in the column must be distinct.

4. Replace the following:

```
< predictable attribute columns> )
```

with:

```
[Quantity] LONG CONTINUOUS PREDICT,
[Amount] DOUBLE CONTINUOUS PREDICT
)
```

5. Replace the following:

```
USING <algorithm name>([parameter list])
WITH DRILLTHROUGH
```

with:

```
USING Microsoft_Time_Series(AUTO_DETECT_PERIODICITY = 0.8, FORECAST_METHOD = 'MIXED')
WITH DRILLTHROUGH
```

The algorithm parameter, `AUTO_DETECT_PERIODICITY` = 0.8, indicates that you want the algorithm to detect cycles in the data. Setting this value closer to 1 favors the discovery of many patterns but can slow processing.

The algorithm parameter, `FORECAST_METHOD`, indicates whether you want the data to be analyzed using ARTXP, ARIMA, or a mixture of both.

The keyword, `WITH DRILLTHROUGH`, specify that you want to be able to view detailed statistics in the source data after the model is complete. You must add this clause if you want to browse the model by using the Microsoft Time Series Viewer. It is not required for prediction.

The complete statement should now be as follows:

```
CREATE MINING MODEL [Forecasting_MIXED]
     (
    [Reporting Date] DATE KEY TIME,
    [Model Region] TEXT KEY,
    [Quantity] LONG CONTINUOUS PREDICT,
    [Amount] DOUBLE CONTINUOUS PREDICT
    )
USING Microsoft_Time_Series (AUTO_DETECT_PERIODICITY = 0.8, FORECAST_METHOD = 'MIXED')
WITH DRILLTHROUGH
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.

7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Forecasting_MIXED.dmx`.

Executing the Query

The final step is to execute the query. After a query is created and saved, it needs to be executed to create the mining model and its mining structure on the server. For more information about executing queries in Query Editor, see Database Engine Query Editor (SQL Server Management Studio).

To execute the query

- In Query Editor, on the toolbar, click **Execute**.

  The status of the query is displayed in the **Messages** tab at the bottom of Query Editor after the statement finishes executing. Messages should display:

  ```
  Executing the query
  Execution complete
  ```

  A new structure named **Forecasting_MIXED_Structure** now exists on the server, together with the related mining model **Forecasting_MIXED**.

In the next lesson, you will add a mining model to the **Forecasting_MIXED** mining structure that you just created.

Next Lesson

Lesson 2: Adding Mining Models to the Time Series Mining Structure

See Also

Mining Model Content for Time Series Models (Analysis Services - Data Mining)
Microsoft Time Series Algorithm Technical Reference

Lesson 2: Adding Mining Models to the Time Series Mining Structure

2/11/2019 • 4 minutes to read • Edit Online

In this lesson, you will add a new mining model to the mining structure that you just created in Lesson 1: Creating a Time Series Mining Model and Mining Structure.

ALTER MINING STRUCTURE Statement

In order to add a new mining model to an existing mining structure, you use the ALTER MINING STRUCTURE (DMX) statement. The code in the statement can be broken into the following parts:

- Identifying the mining structure
- Naming the mining model
- Defining the key column
- Defining the predictable columns
- Specifying the algorithm and any parameter changes

The following is a generic example of the ALTER MINING STRUCTURE statement:

```
ALTER MINING STRUCTURE [<mining structure name>]
ADD MINING MODEL [<mining model name>]
   ([<key columns>],
    <mining model columns>
   )
USING <algorithm name>([<algorithm parameters>])
[WITH DRILLTHROUGH]
```

The first line of the code identifies the existing mining structure to which the mining models will be added:

```
ALTER MINING STRUCTURE [<mining structure name>]
```

The next line of the code names the mining model that will be added to the mining structure:

```
ADD MINING MODEL [<mining model name>]
```

For information about naming an object in DMX, see Identifiers (DMX).

The next lines of the code define columns from the mining structure that will be used by the mining model:

```
[<key columns>],
<mining model columns>
```

You can only use columns that already exist in the mining structure, and the first column in the list must be the key column from the mining structure.

The next lines of the code defines the mining algorithm that generates the mining model and the algorithm

parameters that you can set on the algorithm, and specify whether you can drill down from the mining model into view detailed data in the training cases:

```
USING <algorithm name>([<algorithm parameters>])
WITH DRILLTHROUGH
```

For more information about the algorithm parameters that you can adjust, see Microsoft Time Series Algorithm Technical Reference.

You can specify that a column in the mining model be used for prediction by using the following syntax:

```
<mining model column> PREDICT
```

Lesson Tasks

You will perform the following tasks in this lesson:

- Add a new time series mining model to the structure.
- Change the algorithm parameters to use a different method of analysis and prediction

Adding an ARIMA Time Series Model to the Structure

The first step is to add a new forecasting mining model to the existing structure. By default, the Microsoft Time Series algorithm creates time series mining models by using two algorithms, ARIMA and ARTXP, and blending the results. However, you can specify a single algorithm to use, or you can specify the exact blend of algorithms. In this step, you will add a new model that uses only the ARIMA algorithm. This algorithm is optimized for long-term prediction.

To add an ARIMA time series mining model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX** to open Query Editor and a new, blank query.
2. Copy the generic example of the ALTER MINING STRUCTURE statement into the blank query.
3. Replace the following:

   ```
   <mining structure name>
   ```

   with:

   ```
   [Forecasting_MIXED_Structure]
   ```

4. Replace the following:

   ```
   <mining model name>
   ```

   with:

   ```
   Forecasting_ARIMA
   ```

5. Replace the following:

```
<key columns>,
```

with:

```
[ReportingDate],
[ModelRegion]
```

Note that you do not need to repeat any of the date type or content type information that you provided in the CREATE MINING MODEL statement, because this information is already stored in the mining structure.

6. Replace the following:

```
<mining model columns>
```

with:

```
([Quantity] PREDICT,
[Amount] PREDICT
)
```

7. Replace the following:

```
USING <algorithm name>([<algorithm parameters>])
[WITH DRILLTHROUGH]
```

with:

```
USING Microsoft_Time_Series (AUTO_DETECT_PERIODICITY = .08, FORECAST_METHOD = 'ARIMA')
WITH DRILLTHROUGH
```

The resulting statement should now be as follows:

```
ALTER MINING STRUCTURE [Forecasting_MIXED_Structure]
ADD MINING MODEL [Forecasting_ARIMA]
   (
   ([ReportingDate],
    [ModelRegion],
    ([Quantity] PREDICT,
    [Amount] PREDICT
   )
USING Microsoft_Time_Series (AUTO_DETECT_PERIODICITY = .08, FORECAST_METHOD = 'ARIMA')
WITH DRILLTHROUGH
```

8. On the **File** menu, click **Save DMXQuery1.dmx As**.
9. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Forecasting_ARIMA.dmx`.
10. On the toolbar, click the **Execute** button.

Adding an ARTXP Time Series Model to the Structure

The ARTXP algorithm was the default time series algorithm in SQL Server 2005 and is optimized for short-term prediction. To compare predictions by using all three time series algorithms, you will add one more model that is

based on the ARTXP algorithm.

To add an ARTXP time series mining model

1. Copy the following code into a blank query window.

   Note that you do not need to change anything except the name of the new mining model, and the value of the FORECAST_METHOD parameter.

```
ALTER MINING STRUCTURE [Forecasting_MIXED_Structure]
ADD MINING MODEL [Forecasting_ARTXP]
   (
   ([ReportingDate],
    [ModelRegion],
    ([Quantity] PREDICT,
    [Amount] PREDICT
   )
USING Microsoft_Time_Series (AUTO_DETECT_PERIODICITY = .08, FORECAST_METHOD = 'ARTXP')
WITH DRILLTHROUGH
```

2. On the **File** menu, click **Save DMXQuery1.dmx As**.

3. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Forecasting_ARTXP.dmx`.

4. On the toolbar, click the **Execute** button.

In the next lesson, you will process all of the models and the mining structure.

Next Lesson

Lesson 3: Processing the Time Series Structure and Models

See Also

Microsoft Time Series Algorithm
Microsoft Time Series Algorithm Technical Reference

Lesson 3: Processing the Time Series Structure and Models

2/11/2019 • 3 minutes to read • Edit Online

In this lesson, you will use the INSERT INTO (DMX) statement to process the time series mining structures and mining models that you created.

When you process a mining structure, Analysis Services reads the source data and builds the structures that support mining models. You always have to process a mining model and structure when you first create it. If you specify the mining structure when using INSERT INTO, the statement processes the mining structure and all its associated mining models.

When you add a mining model to a mining structure that has already been processed, you can use the `INSERT INTO MINING MODEL` statement to process just the new mining model by using the existing data.

For more information about processing mining models, see Processing Requirements and Considerations (Data Mining).

INSERT INTO Statement

In order to train the time series mining structure and all its associated mining models, use the INSERT INTO (DMX) statement. The code in the statement can be broken into the following parts.

- Identifying the mining structure
- Listing the columns in the mining structure
- Defining the training data

The following is a generic example of the `INSERT INTO` statement:

```
INSERT INTO MINING STRUCTURE [<mining structure name>]
(
   <mining structure columns>
)
OPENQUERY (<source data definition>)
```

The first line of the code identifies the mining structure that you will train:

```
INSERT INTO MINING STRUCTURE [<mining structure name>]
```

The next lines of the code specify the columns that are defined by the mining structure. You must list each column in the mining structure, and each column must map to a column contained within the source query data.

```
(
   <mining structure columns>
)
```

The final lines of the code define the data that will be used to train the mining structure.

```
OPENQUERY (<source data definition>)
```

In this lesson, you use `OPENQUERY` to define the source data. For more information about other methods of defining a query on the source data, see <source data query>.

Lesson Tasks

You will perform the following task in this lesson:

- Process the mining structure Forecasting_MIXED_Structure
- Process the related mining models Forecasting_MIXED, Forecasting_ARIMA, and Forecasting_ARTXP

Processing the Time Series Mining Structure

To process the mining structure and related mining models by using INSERT INTO

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the INSERT INTO statement into the blank query.

3. Replace the following:

   ```
   [<mining structure>]
   ```

   with:

   ```
   Forecasting_MIXED_Structure
   ```

4. Replace the following:

   ```
   <mining structure columns>
   ```

   with:

   ```
   [ReportingDate],
   [ModelRegion]
   ```

5. Replace the following:

   ```
   OPENQUERY(<source data definition>)
   ```

   with:

   ```
   OPENQUERY([Adventure Works DW 2008R2],'SELECT [ReportingDate], [ModelRegion], [Quantity], [Amount]
   FROM vTimeSeries ORDER BY [ReportingDate]')
   ```

   The source query references the **AdventureWorksDW2012** data source defined in the IntermediateTutorial sample project. It uses this data source to access the view vTimeSeries. This view contains the source data that will be used to train the mining model. If you are not familiar with this project or this views, seeLesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial).

The complete statement should now be as follows:

```
INSERT INTO MINING STRUCTURE [Forecasting_MIXED_Structure]
(
   [ReportingDate],[ModelRegion],[Quantity],[Amount])
)
OPENQUERY(
[Adventure Works DW 2008R2],
'SELECT [ReportingDate],[ModelRegion],[Quantity],[Amount] FROM vTimeSeries ORDER BY [ReportingDate]'
)
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.
7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `ProcessForecastingAll.dmx`.
8. On the toolbar, click the **Execute** button.

After the query has finished running, you can create predictions by using the processed mining models. In the next lesson, you will create several predictions based on the mining models that you created.

Next Lesson

Lesson 4: Creating Time Series Predictions Using DMX

See Also

Processing Requirements and Considerations (Data Mining)
<source data query>
OPENQUERY (DMX)

Lesson 4: Creating Time Series Predictions Using DMX

2/11/2019 • 3 minutes to read • Edit Online

In this lesson and the following lesson, you will use Data Mining Extensions (DMX) to create different types of predictions based on the time series models that you created in Lesson 1: Creating a Time Series Mining Model and Mining Structure and Lesson 2: Adding Mining Models to the Time Series Mining Structure.

With a time series model, you have many options for making predictions:

- Use the existing patterns and data in the mining model
- Use the existing patterns in the mining model but supply new data
- Add new data to the model or update the model.

The syntax for making these prediction types is summarized below:

Default time series prediction
Use PredictTimeSeries (DMX) to return the specified number of predictions from the trained mining model.

For example, see PredictTimeSeries (DMX) or Time Series Model Query Examples.

EXTEND_MODEL_CASES
Use PredictTimeSeries (DMX) with the EXTEND_MODEL_CASES argument to add new data, extend the series, and create predictions based on the updated mining model.

This tutorial contains an example of how to use EXTEND_MODEL_CASES.

REPLACE_MODEL_CASES
Use PredictTimeSeries (DMX) with the REPLACE_MODEL_CASES argument to replace the original data with a new data series, and then create predictions based on applying the patterns in the mining model to the new data series.

For an example of how to use REPLACE_MODEL_CASES, see Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial).

Lesson Tasks

You will perform the following tasks in this lesson:

- Create a query to get the default predictions based on existing data.

In the following lesson you will perform the following related tasks:

- Create a query to supply new data and get updated predictions.

In addition to creating queries manually by using DMX, you can also create predictions by using the prediction query builder in SQL Server Data Tools (SSDT).

Simple Time Series Prediction Query

The first step is to use the `SELECT FROM` statement together with the `PredictTimeSeries` function to create time series predictions. Time series models support a simplified syntax for creating predictions: you do not need to

supply any inputs, but only have to specify the number of predictions to create. The following is a generic example of the statement you will use:

```
SELECT <select list>
FROM [<mining model name>]
WHERE [<criteria>]
```

The select list can contain columns from the model, such as the name of the product line that you are creating the predictions for, or prediction functions, such as Lag (DMX) or PredictTimeSeries (DMX), which are specifically for time series mining models.

To create a simple time series prediction query

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the statement into the blank query.

3. Replace the following:

   ```
   <select list>
   ```

   with:

   ```
   [Forecasting_MIXED].[ModelRegion],
   PredictTimeSeries([Forecasting_MIXED].[Quantity],6) AS PredictQty,
   PredictTimeSeries ([Forecasting_MIXED].[Amount],6) AS PredictAmt
   ```

   The first line retrieves a value from the mining model that identifies the series.

   The second and third lines use the `PredictTimeSeries` function. Each line predicts a different attribute, `[Quantity]` or `[Amount]`. The numbers after the names of the predictable attributes specify the number of time steps to predict.

   The `AS` clause is used to provide a name for the column that is returned by each prediction function. If you do not supply an alias, by default both columns are returned with the label, `Expression`.

4. Replace the following:

   ```
   [<mining model>]
   ```

   with:

   ```
   [Forecasting_MIXED]
   ```

5. Replace the following:

   ```
   WHERE [criteria>]
   ```

   with:

   ```
   WHERE [ModelRegion] = 'M200 Europe' OR
   [ModelRegion] = 'M200 Pacific'
   ```

The complete statement should now be as follows:

```
SELECT
[Forecasting_MIXED].[ModelRegion],
PredictTimeSeries([Forecasting_MIXED].[Quantity],6) AS PredictQty,
PredictTimeSeries ([Forecasting_MIXED].[Amount],6) AS PredictAmt
FROM
[Forecasting_MIXED]
WHERE [ModelRegion] = 'M200 Europe' OR
[ModelRegion] = 'M200 Pacific'
```

6. On the **File** menu, click **Save DMXQuery1.dmx As**.

7. In the **Save As** dialog box, browse to the appropriate folder, and name the file `SimpleTimeSeriesPrediction.dmx`.

8. On the toolbar, click the **Execute** button.

   The query returns 6 predictions for each of the two combinations of product and region that are specified in the `WHERE` clause.

In the next lesson, you will create a query that supplies new data to the model, and compare the results of that prediction with the one you just created.

Next Task in Lesson

Lesson 5: Extending the Time Series Model

See Also

PredictTimeSeries (DMX)
Lag (DMX)
Time Series Model Query Examples
Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial)

Lesson 5: Extending the Time Series Model

2/11/2019 • 6 minutes to read • Edit Online

In SQL Server 2014 Enterprise, you can add new data to a time series model and automatically incorporate the new data into the model. You add new data to a time series mining model in one of two ways:

- Use a PREDICTION JOIN to join data in an external source to the training data.
- Use a singleton prediction query to provide data one slice at a time.

For example, assume that you trained the mining model on existing sales data some months ago. When you get new sales, you might want to update the sales predictions to incorporate the new data. You can do this in one step, by supplying the new sales figures as input data and generating new predictions based on the composite data set.

Making Predictions with EXTEND_MODEL_CASES

The following are generic examples of a time series prediction using EXTEND_MODEL_CASES. The first example enables you to specify the number of predictions starting from the last time step of the original model:

```
SELECT [<model columns>,] PredictTimeSeries(<table column reference>, n, EXTEND_MODEL_CASES)
FROM <mining model>
PREDICTION JOIN <source query>
[WHERE <criteria>]
```

The second example enables you to specify the time step where predictions should start, and where they should end. This option is important when you extend the model cases because, by default, the time steps used for prediction queries always start at the end of the original series.

```
SELECT [<model columns>,] PredictTimeSeries(<table column reference>, n-start, n-end, EXTEND_MODEL_CASES)
FROM <mining model>
PREDICTION JOIN <source query>
[WHERE <criteria>}
```

In this tutorial, you will create both kinds of queries.

To create a singleton prediction query on a time series model

1. In **Object Explorer**, right-click the instance of Analysis Services, point to **New Query**, and then click **DMX**.

   Query Editor opens and contains a new, blank query.

2. Copy the generic example of the singleton statement into the blank query.

3. Replace the following:

   ```
   SELECT [<model columns>,] PredictTimeSeries(<table column reference>, n, EXTEND_MODEL_CASES)
   ```

   with:

   ```
   SELECT [Model Region],
   PredictTimeSeries([Quantity],6, EXTEND_MODEL_CASES) AS PredictQty
   ```

   The first line retrieves a value from the model that identifies the series.

The second line contains the prediction function, which gets 6 predictions for Quantity. An alias, `PredictQty`, is assigned to the prediction result column to make it easier to understand the results.

4. Replace the following:

```
FROM <mining model>
```

with:

```
FROM [Forecasting_MIXED]
```

5. Replace the following:

```
PREDICTION JOIN <source query>
```

with:

```
NATURAL PREDICTION JOIN
(
   SELECT 1 AS [Reporting Date],
   '10' AS [Quantity],
   'M200 Europe' AS [Model Region]
   UNION SELECT
   2 AS [Reporting Date],
   15 AS [Quantity]),
   'M200 Europe' AS [Model Region]
) AS t
```

6. Replace the following:

```
[WHERE <criteria>]
```

with:

```
WHERE [ModelRegion] = 'M200 Europe' OR
[ModelRegion] = 'M200 Pacific'
```

The complete statement should now be as follows:

```
SELECT [Model Region],
PredictTimeSeries([Quantity],6, EXTEND_MODEL_CASES) AS PredictQty
FROM
   [Forecasting_MIXED]
NATURAL PREDICTION JOIN
   SELECT 1 AS [ReportingDate],
  '10' AS [Quantity],
  'M200 Europe' AS [ModelRegion]
UNION SELECT
  2 AS [ReportingDate],
  15 AS [Quantity]),
  'M200 Europe' AS [ModelRegion]
) AS t
WHERE [ModelRegion] = 'M200 Europe' OR
[ModelRegion] = 'M200 Pacific'
```

7. On the **File** menu, click **Save DMXQuery1.dmx As**.
8. In the **Save As** dialog box, browse to the appropriate folder, and name the file `Singleton_TimeSeries_Query.dmx`.
9. On the toolbar, click the **Execute** button.

   The query returns predictions of sales quantity for the M200 bicycle in the Europe and Pacific regions.

Understanding Prediction Start with EXTEND_MODEL_CASES

Now that you have created predictions based on the original model, and with new data, you can compare the results to see how updating the sales data affects the predictions. Before you do so, review the code that you just created, and notice the following:

- You supplied new data for only the Europe region.
- You supplied only two months' worth of new data.

The following table shows how the new values supplied for M200 Europe affect predictions. You did not provide any new data for the M200 product in the Pacific region, but this series is presented for comparison:

Product and Region: M200 Europe

		Existing model (`PredictTimeSeries`)	Model with updated sales data (`PredictTimeSeries` with `EXTEND_MODEL_CASES`)
M200 Europe	7/25/2008 12:00:00 AM	77	10
M200 Europe	8/25/2008 12:00:00 AM	64	15
M200 Europe	9/25/2008 12:00:00 AM	59	72
M200 Europe	10/25/2008 12:00:00 AM	56	69
M200 Europe	11/25/2008 12:00:00 AM	56	68
M200 Europe	12/25/2008 12:00:00 AM	74	89

Product and Region: M200 Pacific

		Existing model (`PredictTimeSeries`)	Model with updated sales data (`PredictTimeSeries` with `EXTEND_MODEL_CASES`)
M200 Pacific	7/25/2008 12:00:00 AM	41	41
M200 Pacific	8/25/2008 12:00:00 AM	44	44
M200 Pacific	9/25/2008 12:00:00 AM	38	38
M200 Pacific	10/25/2008 12:00:00 AM	41	41

M200 Pacific	11/25/2008 12:00:00 AM	36	36
M200 Pacific	12/25/2008 12:00:00 AM	39	39

From these results, you can see two things:

- The first two predictions for the M200 Europe series are exactly the same as the new data you supplied. By design, Analysis Services returns the actual new data points instead of making a prediction. That is because when you extend the model cases, the time steps used for prediction queries always start at the end of the original series. Therefore, if you add two new data points, the first two predictions returned overlap with the new data.
- After all the new data points are used up, Analysis Services makes predictions based on the updated model. Therefore, starting in September 2005, you can see the difference between predictions for M200 Europe from the original model, in the left-hand column, and the model that uses EXTEND_MODEL_CASES, in the right-hand column. The predictions are different because the model has been updated with the new data.

Using Start and End Time Steps to Control Predictions

When you extend a model, the new data is always attached to the end of the series. However, for the purpose of prediction, the time slices used for prediction queries start at the end of the original series. If you want to obtain only the new predictions when you add the new data, you must specify the starting point as a number of time slices. For example, if you are adding two new data points and want to make four new predictions, you would do the following:

- Create a PREDICTION JOIN on a time series model, and specify two months of new data.
- Request predictions for four time slices, where the starting point is 3, and the ending point is time slice 6.

In other words, if your new data contains n time slices, and you request predictions for time steps 1 through n, the predictions will coincide with the same period as the new data. To get new predictions for a time periods not covered by your data, you must either start predictions at the time slice n+1 after the new data series, or make sure that you request additional time slices.

> **NOTE**
> You cannot make historical predictions when you add new data.

The following example shows the DMX statement that lets you get only the new predictions for the two series in the previous example.

```
SELECT [Model Region],
PredictTimeSeries([Quantity],3,6, EXTEND_MODEL_CASES) AS PredictQty
FROM
   [Forecasting_MIXED]
NATURAL PREDICTION JOIN
   SELECT 1 AS [ReportingDate],
  '10' AS [Quantity],
  'M200 Europe' AS [ModelRegion]
UNION SELECT
  2 AS [ReportingDate],
  15 AS [Quantity]),
  'M200 Europe' AS [ModelRegion]
) AS t
WHERE [ModelRegion] = 'M200 Europe'
```

The prediction results start at time slice 3, which is after the 2 months of new data that you supplied.

Product and Region: M200 Europe

Model with updated data (PredictTimeSeries with EXTEND_MODEL_CASES)

M200 Europe	9/25/2008 12:00:00 AM	72
M200 Europe	10/25/2008 12:00:00 AM	69
M200 Europe	11/25/2008 12:00:00 AM	68
M200 Europe	12/25/2008 12:00:00 AM	89

Making Predictions with REPLACE_MODEL_CASES

Replacing the model cases is useful when you want to train a model on one set of cases and then apply that model to a different data series. A detailed walkthrough of this scenario is presented in Lesson 2: Building a Forecasting Scenario (Intermediate Data Mining Tutorial).

See Also

Time Series Model Query Examples
PredictTimeSeries (DMX)

Reporting Services Tutorials (SSRS)

2/14/2019 • 2 minutes to read • Edit Online

This section contains tutorials for SQL Server 2012 Reporting Services.

Create a Data-Driven Subscription (SSRS Tutorial)
Reporting Services provides data-driven subscriptions so that you can customize the distribution of a report based on dynamic list of subscribers that will receive the report. Data-driven subscriptions are typically created and maintained by report server administrators. The ability to create data-driven subscriptions requires expertise in building queries, knowledge of data sources that contain subscriber data, and elevated permissions on a report server.

Create a Basic Table Report (SSRS Tutorial)
Follow the steps in this tutorial to learn how to create your first report. This tutorial shows you how to work with SQL Server Data Tools (SSDT) to create a data connection (data source), define a simple query (dataset), and configure a data region to contain your data.

Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)
Follow the steps in this tutorial to learn how to access the SQL Server 2014 Reporting Services (SSRS) Web services from an application created with Microsoft Visual Basic or Microsoft Visual C#.

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)
Follow the steps in this tutorial to learn how to use the XML Schema Definition Tool (Xsd.exe) to generate classes that allow you to serialize and deserialize report definition files (.rdl and .rdlc) with the Microsoft .NET Framework.

Create a Drillthrough (RDLC) Report with Parameters using ReportViewer (SSRS Tutorial)
Follow the steps in this tutorial to learn how to create a drillthrough report with parameters and a filter using the ReportViewer control.

See Also

Adventure Works for SQL Server 2012
Reporting Services Samples on CodePlex (https://go.microsoft.com/fwlink/?LinkId=198283)
TechNet WIKI: SQL Server 2012 Samples (https://go.microsoft.com/fwlink/?linkID=220734)
Picnic Dataset (Sample Data) (https://go.microsoft.com/fwlink/?linkID=219108)

Create a Basic Table Report (SSRS Tutorial)

2/14/2019 • 2 minutes to read • Edit Online

This tutorial is designed to help you create a basic table report based on the **AdventureWorks2012** database using Report Designer. You can also use Report Builder or the Report Wizard to create reports. In this tutorial, you will create a report project, set up connection information, define a query, add a Table data region, group and total some fields, and preview the report.

> **NOTE**
> To complete this tutorial, you must be running Reporting Services in native mode. If you are running Reporting Services in SharePoint integrated mode, the steps that use report server URLs do not work. For more information about Reporting Services modes, see Reporting Services Report Server.

Requirements

Your system must have the following installed to use this tutorial:

- Microsoft SQL Server 2014 database engine.
- The **AdventureWorks2012** database. For more information, see Adventure Works for SQL Server 2012 (Adventure Works for SQL Server 2012) (https://go.microsoft.com/fwlink/?LinkId=245471).. For more information about support for SQL Server sample databases and sample code for SQL Server Express, see Databases and Samples Overview on the CodePlex Web site.
- SQL Server 2014 Reporting Services (SSRS).
- SQL Server SQL Server Data Tools (SSDT).

  > **NOTE**
  > SQL Server Data Tools is not supported on Itanium-based computers. However, support for SQL Server Data Tools is available for x64-based computers. If the SQL Server sample databases have been deployed on an Itanium-based computer, use BI Development Studio on either an x86-based or x64-based computer to modify and run the samples.

You must also have read-only permissions to retrieve data from the **AdventureWorks2012** database.

Tasks

Lesson 1: Creating a Report Server Project (Reporting Services)

Lesson 2: Specifying Connection Information (Reporting Services)

Lesson 3: Defining a Dataset for the Table Report (Reporting Services)

Lesson 4: Adding a Table to the Report (Reporting Services)

Lesson 5: Formatting a Report (Reporting Services)

Lesson 6: Adding Grouping and Totals (Reporting Services)

> **NOTE**
> When reviewing tutorials, we recommend that you add **Next** and **Previous** buttons to the document viewer toolbar. For more information, see.

See Also

Reporting Services Tutorials (SSRS)

Lesson 1: Creating a Report Server Project (Reporting Services)

2/14/2019 • 2 minutes to read • Edit Online

To create a report in SQL Server, you must first create a report server project where you will save your report definition (.rdl) file and any other resource files that you need for your report. Then you will create the actual report definition file, define a data source for your report, define a dataset, and define the report layout. When you run the report, the actual data is retrieved and combined with the layout, and then rendered on your screen, from where you can export it, print it, or save it.

In this lesson, you will learn how to create a report server project in SQL Server Data Tools (SSDT). A report server project is used to create reports that run on a report server.

To create a report server project

1. Click **Start**, point to **All Programs**, point to **Microsoft SQL Server 2014**, and then click **SQL Server Data Tools**. If this is the first time you have opened SQL Server Data Tools, click **Business Intelligence Settings** for the default environment settings.
2. On the **File** menu, point to **New**, and then click **Project**.
3. In the **Installed Templates** list, click **Business Intelligence**.
4. Click **Report Server Project**.
5. In **Name**, type **Tutorial**.
6. Click **OK** to create the project.

   The Tutorial project is displayed in Solution Explorer.

To create a new report definition file

1. In Solution Explorer, right-click **Reports**, point to **Add**, and click **New Item**.

   > **NOTE**
   >
   > If the **Solution Explorer** window is not visible, from the **View** menu, click **Solution Explorer**.

2. In the **Add New Item** dialog box, under **Templates**, click **Report**.
3. In **Name**, type **Sales Orders.rdl** and then click **Add**.

   Report Designer opens and displays the new .rdl file in Design view.

Report Designer is a Reporting Services component that runs in SQL Server Data Tools (SSDT). It has two views: **Design** and **Preview**. Click each tab to change views.

You define your data in the **Report Data** pane. You define your report layout in **Design** view. You can run the report and see what it looks like in **Preview** view.

Next Task

You have successfully created a report project called "Tutorial" and added a report definition (.rdl) file to the report project. Next, you will specify a data source to use for the report. See Lesson 2: Specifying Connection Information

(Reporting Services).

See Also

Create a Basic Table Report (SSRS Tutorial)

Lesson 2: Specifying Connection Information (Reporting Services)

2/14/2019 • 2 minutes to read • Edit Online

After you add a report to the Tutorial project, you need to define a *data source*, which is connection information the report uses to access data from either a relational database, multidimensional database, or other resource.

In this lesson, you will use the **AdventureWorks2012** sample database as your data source. This tutorial assumes that this database is located in a default instance of SQL Server Database Engine that is installed on your local computer.

To set up a connection

1. In the **Report Data** pane, click **New** and then click **Data Source...**.

   > **NOTE**
   > If the **Report Data** pane is not visible, from the **View** menu, click **Report Data**.

2. In **Name**, type **AdventureWorks2012**.
3. Make sure **Embedded connection** is selected.
4. In **Type**, select **Microsoft SQL Server**.
5. In **Connection string**, type the following:

   ```
   Data source=localhost; initial catalog=AdventureWorks2012
   ```

   This connection string assumes that SQL Server Data Tools (SSDT), the report server, and the **AdventureWorks2012** database are all installed on the local computer and that you have permission to log on to the **AdventureWorks2012** database.

   > **NOTE**
   > If you are using SQL Server Express with Advanced Services or a named instance, the connection string must include instance information:
   >
   > ```
   > Data source=localhost\SQLEXPRESS; initial catalog=AdventureWorks2012
   > ```
   >
   > For more information about connection strings, see Data Connections, Data Sources, and Connection Strings in Reporting Services and Data Source Properties Dialog Box, General.

6. Click **Credentials** in the left pane and click **Use Windows Authentication (integrated security)**.
7. Click **OK**. data source AdventureWorks2012 is added to the **Report Data** pane.

Next Task

You have successfully defined a connection to the **AdventureWorks2012** sample database. Next, you will create the report. See Lesson 3: Defining a Dataset for the Table Report (Reporting Services).

See Also

Data Connections, Data Sources, and Connection Strings in Reporting Services

Lesson 3: Defining a Dataset for the Table Report (Reporting Services)

2/14/2019 • 2 minutes to read • Edit Online

After you define the data source, you need to define a dataset. In Reporting Services, data that you use in reports is contained in a *dataset*. A dataset includes a pointer to a data source and a query to be used by the report, as well as calculated fields and variables.

You can use the query designer in Report Designer to design the query. For this tutorial, you will create a query that retrieves sales order information from the **AdventureWorks20122008** database.

To define a Transact-SQL query for report data

1. In the **Report Data** pane, click **New**, and then click **Dataset...**. The **Dataset Properties** dialog box opens.
2. In the **Name** box, type **AdventureWorksDataset**.
3. Click **Use a dataset embedded in my report**.
4. Make sure the name of your data source, AdventureWorks2012, is in the **Data source** text box, and that the **Query type** is **Text**.
5. Type, or copy and paste, the following Transact-SQL query into the **Query** box.

```
SELECT
   soh.OrderDate AS [Date],
   soh.SalesOrderNumber AS [Order],
   pps.Name AS Subcat, pp.Name as Product,
   SUM(sd.OrderQty) AS Qty,
   SUM(sd.LineTotal) AS LineTotal
FROM Sales.SalesPerson sp
   INNER JOIN Sales.SalesOrderHeader AS soh
      ON sp.BusinessEntityID = soh.SalesPersonID
   INNER JOIN Sales.SalesOrderDetail AS sd
      ON sd.SalesOrderID = soh.SalesOrderID
   INNER JOIN Production.Product AS pp
      ON sd.ProductID = pp.ProductID
   INNER JOIN Production.ProductSubcategory AS pps
      ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
   INNER JOIN Production.ProductCategory AS ppc
      ON ppc.ProductCategoryID = pps.ProductCategoryID
GROUP BY ppc.Name, soh.OrderDate, soh.SalesOrderNumber, pps.Name, pp.Name,
   soh.SalesPersonID
HAVING ppc.Name = 'Clothing'
```

6. (Optional) Click the **Query Designer** button. The query is displayed in the text-based query designer. You can toggle to the graphical query designer by clicking **Edit As Text**. View the results of the query by clicking the run **(!)** button on the query designer toolbar.

   You see the data from six fields from four different tables in the **AdventureWorks2012** database. The query makes use of Transact-SQL functionality such as aliases. For example, the SalesOrderHeader table is called soh.

   Click **OK** to exit the query designer.

7. Click **OK** to exit the **Dataset Properties** dialog box.

Your **AdventureWorksDataset** dataset and fields appear in the Report Data pane.

Next Task

You have successfully specified a query that retrieves data for your report. Next, you will create the report layout. See Lesson 4: Adding a Table to the Report (Reporting Services).

See Also

Query Design Tools in Report Designer SQL Server Data Tools (SSRS)
SQL Server Connection Type (SSRS)
Tutorial: Writing Transact-SQL Statements

Lesson 4: Adding a Table to the Report (Reporting Services)

2/14/2019 • 2 minutes to read • Edit Online

After the dataset is defined, you can start designing the report. You create a report layout by dragging and dropping data regions, text boxes, images, and other items that you want to include in your report to the design surface.

Items that contain repeated rows of data from underlying datasets are called *data regions*. A basic report will have only one data region, but you can add more, for example, if you want to add a chart to your tabular report. After you add a data region, you can add fields to the data region.

To add a Table data region and fields to a report layout

1. In the **Toolbox**, click **Table**, and then click on the design surface and drag the mouse. Report Designer draws a table data region with three columns in the center of the design surface.

   > **NOTE**
   >
   > The **Toolbox** may appear as a tab on the left side of the **Report Data** pane. To open the **Toolbox**, move the pointer over the **Toolbox** tab. If the **Toolbox** is not visible, from the **View** menu, click **Toolbox**.

2. In the **Report Data** pane, expand the **AdventureWorksDataset** dataset to display the fields.

3. Drag the Date field from the **Report Data** pane to the first column in the table.

   When you drop the field into the first column, two things happen. First, the data cell will display the field name, known as the *field expression*, in brackets: `[Date]`. Second, a column header value is automatically added to Header row, just above the field expression. By default, the column is the name of the field. You can select the Header row text and type a new name.

4. Drag the Order field from the **Report Data** pane to the second column in the table.

5. Drag the Product field from the **Report Data** pane to the third column in the table.

6. Drag the Qty field to the right edge of the third column until you get a vertical cursor and the mouse pointer has a plus sign [+]. When you release the mouse button, a fourth column is created for `[Qty]`.

7. Add the LineTotal field in the same way, creating a fifth column.

   > **NOTE**
   >
   > The column header is Line Total. Report Designer automatically creates a friendly name for the column by splitting LineTotal into two words.

   The following diagram shows a table data region that has been populated with these fields: Date, Order, Product, Qty, and Line Total.

Preview Your Report

Previewing a report enables you to view the rendered report without having to first publish it to a report server. You will probably want to preview your report frequently during design time. Previewing the report will also run validation on the design and data connections so you can correct errors and issues before publishing the report to a report server.

To preview a report

- Click the **Preview** tab. Report Designer runs the report and displays it in Preview view.

  The following diagram shows part of the report in Preview view.

Date	Order	Product	Qty	Line Total
7/1/2005 12:00:00 AM	SO43659	AWC Logo Cap	2	10.373000
7/1/2005 12:00:00 AM	SO43659	Long-Sleeve Logo Jersey, M	3	86.521200
7/1/2005 12:00:00 AM	SO43659	Long-Sleeve Logo Jersey, XL	1	28.840400

  Notice that the currency (in the Line Total column) has six places after the decimal, and the date has includes a time stamp. You're going to fix that formatting in the next lesson.

> **NOTE**
>
> On the **File** menu, click **Save All** to save the report.

Next Steps

You have successfully added a Table data region to your report, added fields to the data region, and previewed your report. Next, you will format column headers and date and currency values. See Lesson 5: Formatting a Report (Reporting Services).

See Also

Tables (Report Builder and SSRS)
Dataset Fields Collection (Report Builder and SSRS)

Lesson 5: Formatting a Report (Reporting Services)

2/14/2019 • 2 minutes to read • Edit Online

Now that you've added a data region and some fields to the Sales Orders report, you can format the date and currency fields and the column headers.

In this topic:

- Format the Date
- Format the Currency
- Change Text Style and Column Widths

Format the Date

The Date field displays date and time information by default. You can format it to display only the date.

To format a date field

1. Click the **Design** tab.
2. Right-click the cell with the `[Date]` field expression and then click **Text Box Properties**.
3. Click **Number**, and then in the **Category** field, select `Date`.
4. In the **Type** box, select **January 31, 2000**.
5. Click **OK**.
6. Preview the report to see the change to the `[Date]` field and then change back to design view.

Format the Currency

The LineTotal field displays a general number. Format it to display the number as currency.

To format a currency field

1. Right-click the cell with the `[LineTotal]` field expression and then click **Text Box Properties**.
2. Click **Number**, and in the **Category** field, select **Currency**.
3. If your regional setting is English (United States), the defaults should be:
   - **Decimal places: 2**
   - **Negative numbers: ($12345.00)**
   - **Symbol: $ English (United States)**
4. Select **Use 1000 separator (,)**.

   If the sample text is:**$12,345.00**, then your settings are correct.
5. Click **OK**.
6. Preview the report to see the change to the `[LineTotal]` field and then change back to design view.

Change Text Style and Column Widths

You can also change the formatting of the header row to differentiate it from the rows of data in the report. Lastly, you will adjust the widths of the columns.

To format header rows and table columns

1. Click the table so that column and row handles appear above and next to the table.

	Date	Order	Product	Qty	Line Total
≡	[Date]	[Order]	[Product]	[Qty]	[LineTotal]

   The gray bars along the top and side of the table are the column and row handles.

2. Point to the line between column handles so that the cursor changes into a double arrow. Drag the columns to the size you want.

3. Select the row containing column header labels and from the **Format** menu, point to **Font** and then click **Bold**.

4. To preview your report, click the **Preview** tab. It should look something like this:

Date	Order	Product	Qty	Line Total
July 01, 2005	SO43659	AWC Logo Cap	2	$10.37
July 01, 2005	SO43659	Long-Sleeve Logo Jersey, M	3	$86.52
July 01, 2005	SO43659	Long-Sleeve Logo Jersey, XL	1	$28.84

5. On the **File** menu, click **Save All** to save the report.

Next Steps

You have successfully formatted column headers and date and currency values. Next, you will add grouping and totals to your report. See Lesson 6: Adding Grouping and Totals (Reporting Services).

See Also

Formatting Numbers and Dates (Report Builder and SSRS)
Rendering Behaviors (Report Builder and SSRS)

Lesson 6: Adding Grouping and Totals (Reporting Services)

2/14/2019 • 4 minutes to read • Edit Online

Add grouping and totals to your report to organize and summarize your data.

For information about adding running totals to reports, see this curation on curah.microsoft.com: Adding totals to Reporting Services (SSRS) reports.

In this topic:

- To group data in a report
- To add totals to a report
- To add a daily total to a report
- To add a grand total to a report
- To Publish the Report to the Report Server (Optional)

To group data in a report

1. Click the **Design** tab.
2. If you do not see the **Row Groups** pane , right-click the design surface and click **view** and then click **Grouping**.
3. From the **Report Data** pane, drag the `Date` field to the **Row Groups** pane. Place it above the row called **(Details)**.

   Note that the row handle now has a bracket in it, to show a group. The table now also has two Date columns -- one on either side of a vertical dotted line.

4. From the **Report Data** pane, drag the `Order` field to the **Row Groups** pane. Place it below Date and above **(Details)**.

   Note that the row handle now has two brackets in it, to show two groups. The table now has two `Order` columns, too.

5. Delete the original Date and Order columns to the **right** of the double line. This removes this individual record values so that only the group value is displayed. Select the column handles for the two columns, right-click and click **Delete Columns**.

You can format the column headers and date again.

6. Switch to the **Preview** tab to preview the report. It should look similar to the following illustration:

Date	Order	Product	Qty	Line Total
July 01, 2005	SO43659	AWC Logo Cap	2	$10.37
		Long-Sleeve Logo Jersey, M	3	$86.52
		Long-Sleeve Logo Jersey, XL	1	$28.84
		Mountain Bike Socks, M	6	$34.20
	SO43661	AWC Logo Cap	4	$20.75
		Long-Sleeve Logo Jersey, L	4	$115.36
		Long-Sleeve Logo Jersey, XL	2	$57.68

To add totals to a report

1. Switch to Design view.
2. Right-click the data region cell that contains the field `[LineTotal]`, and click **Add Total**.

   This adds a row with a sum of the dollar amount for each order.
3. Right-click the cell that contains the field `[Qty]`, and click **Add Total**.

   This adds a sum of the quantity for each order to the totals row.
4. In the empty cell to the left of `Sum[Qty]`, type the label "**Order Total**".
5. You can add a background color to the totals row. Select the two sum cells and the label cell.
6. On the **Format** menu, click **Background Color**, click **Light Gray**, and click **OK**.

To add a daily total to a report

1. Right-click the Order cell, point to **Add Total**, and click **After**.

   This adds a new row containing sums of the quantity and dollar amount for each day, and the label "**Total**" in the Order column.
2. Type the word **Daily** before the word **Total** in the same cell, so it reads **Daily Total**.
3. Select the **Daily Total** cell, the two **Sum** cells and the empty cell between them.

4. On the **Format** menu, click **Background Color**, click **Orange**, and click **OK**.

To add a grand total to a report

1. Right-click the Date cell, point to **Add Total**, and click **After**.

   This adds a new row containing sums of the quantity and dollar amount for the entire report, and the **Total** label in the `Date` column.

2. Type the word **Grand** before the word **Total** in the same cell, so it reads **Grand Total**.

3. Select the **Grand Total** cell, the two **Sum** cells and the empty cells between them.

4. On the **Format** menu, click **Background Color**, click **Light Blue**, and click **OK**.

5. Click Preview.

   The last page should look something like this:

	S071950	Women's Mountain Shorts, L	6	$251.96
		Women's Mountain Shorts, S	3	$125.98
		Order Total	9	$377.95
	S071951	Women's Mountain Shorts, L	3	$125.98
		Order Total	3	$125.98
	S071952	Women's Mountain Shorts, L	15	$548.55
		Women's Mountain Shorts, M	3	$125.98
		Women's Mountain Shorts, S	3	$125.98
		Order Total	21	$800.51
	Daily Total		2999	$83,642.49
Grand Total			84589	$1,780,769.91

To Publish the Report to the Report Server (Optional)

1. An optional step is to publish the completed report to the native mode report server so you can view the report from Report Manager.

2. On the toolbar click **Project** and then click **tutorial Properties...**

3. In the **TargetServerURL** type the name of the name of your report server, for example **http://<servername>/reportserver**

4. Click **OK**

5. On the toolbar click **Build** and then click **Deploy tutorial**.

   If you see a message similar to the following in the output window, it indicates a successful deployment.

   > ------ Build started: Project: tutorial, Configuration: Debug ------Skipping 'Sales Orders.rdl'. Item is up to date.Build complete -- 0 errors, 0 warnings------ Deploy started: Project: tutorial, Configuration: Debug ------Deploying to http://<server name>/reportserverDeploying report '/tutorial/Sales Orders'.Deploy complete -- 0 errors, 0 warnings========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ========== Deploy: 1 succeeded, 0 failed, 0 skipped ==========

   If you see an error message similar to the following, verify you have permissions on the report server and you have started SQL Server Data Tools with administrator privileges.

   > "The permissions granted to user 'XXXXXXXX\<your user name>' are insufficient for performing this operation"

6. Start Report Manager with administrator privileges, for example, right-click the icon for Internet Explorer and click **Run as administrator**.

   Browse to the Report Manager URL, for example: `http://<server name>/reports`.

7. Browse to the folder that contains the report and click the name of the report `Sales Orders` to view the rendered report in the browser.

Next Steps

You have successfully completed the Creating a Basic Table Report tutorial.

See Also

Filter, Group, and Sort Data (Report Builder and SSRS)

Create a Data-Driven Subscription (SSRS Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

Reporting Services provides data-driven subscriptions so that you can customize the distribution of a report based on dynamic subscriber data. Data-driven subscriptions are intended for the following kinds of scenarios:

- Distributing reports to a large recipient pool whose membership may change from one distribution to the next. For example, distribute a monthly report to all current customers.
- Distributing reports to a specific group of recipients based on predefined criteria. For example, send a sales performance report to the top ten sales managers in an organization.

What You Will Learn

This tutorial shows you how to use data-driven subscriptions using a simple example to illustrate the concepts.

This tutorial is divided into three lessons:

Lesson 1: Creating a Sample Subscriber Database
In this lesson you will learn how to create a local SQL Server database that contains subscriber information.

Lesson 2: Modifying the Report Data Source Properties
In this lesson, you will learn how to modify report data source properties so that the report can run unattended. Unattended processing requires stored credentials. You will also modify the report dataset to include a parameter that is supplied by the subscriber data.

Lesson 3: Defining a Data-Driven Subscription
In this lesson you will learn how to define a data-driven subscription. This lesson guides you through each page in the Data-Driven Subscription Wizard.

Requirements

Data-driven subscriptions are typically created and maintained by report server administrators. The ability to create data-driven subscriptions requires expertise in building queries, knowledge of data sources that contain subscriber data, and elevated permissions on a report server.

The tutorial will use the report created in the tutorial Create a Basic Table Report (SSRS Tutorial) and data from **AdventureWorks2012**

Your system must have the following installed to use this tutorial:

- An edition of SQL Server that supports data-driven subscriptions. For more information, see Editions and Components of SQL Server 2014.
- The report server must be running in native mode. The user interface described in this tutorial is based on a native mode report server. Subscriptions are supported on SharePoint mode report servers but the user interface will be different than what is described in this tutorial.
- SQL Server Agent service must be running.
- A report that includes parameters. This tutorial assumes the sample report, `Sales Orders` you create using the tutorial Create a Basic Table Report (SSRS Tutorial).
- The **AdventureWorks2012** sample database, which provides data to the sample report.

- A role assignment that includes the Manage all subscriptions task on the sample report. This task is required for defining a data-driven subscription. If you are an administrator on the computer, the default role assignment for local administrators provides the permissions necessary for creating data-driven subscriptions. For more information, see Granting Permissions on a Native Mode Report Server.
- A shared folder for which you have write permissions. The shared folder must be accessible over a network connection.

Estimated time to complete the tutorial: 30 minutes. An additional 30 minutes if you have not completed the basic report tutorial.

See Also

Data-Driven Subscriptions
Create a Basic Table Report (SSRS Tutorial)

Lesson 1: Creating a Sample Subscriber Database

2/11/2019 • 2 minutes to read • Edit Online

Before you can define a data-driven subscription, you must have a data source that provides subscription data. In this step, you will create a small database to store the subscription data used in this tutorial. Later, when the subscription is processed, the report server retrieves this data and uses it to customize report output, delivery options, and report presentation format.

This lesson assumes you are using Management Studio to create a SQL Server 2014 database.

To create a sample Subscriber database

1. Start Management Studio, and open a connection to a Database Engine.
2. Right-click on Databases, select **New Database...**.
3. In the New Database dialog box, in Database Name, type *Subscribers*. Click **OK**.
4. Click the **New Query** button on the toolbar.
5. Copy the following Transact-SQL statements into the empty query:

```
Use Subscribers
CREATE TABLE [dbo].[OrderInfo] (
    [SubscriptionID] [int] NOT NULL PRIMARY KEY ,
    [Order] [nvarchar] (20) NOT NULL,
    [FileType] [bit],
    [Format] [nvarchar] (20) NOT NULL ,
) ON [PRIMARY]
GO

INSERT INTO [dbo].[OrderInfo] (SubscriptionID, [Order], FileType, Format)
VALUES ('1', 'so43659', '1', 'IMAGE')
INSERT INTO [dbo].[OrderInfo] (SubscriptionID, [Order], FileType, Format)
VALUES ('2', 'so43664', '1', 'MHTML')
INSERT INTO [dbo].[OrderInfo] (SubscriptionID, [Order], FileType, Format)
VALUES ('3', 'so43668', '1', 'PDF')
INSERT INTO [dbo].[OrderInfo] (SubscriptionID, [Order], FileType, Format)
VALUES ('4', 'so71949', '1', 'Excel')
GO
```

6. Click **! Execute** on the toolbar.
7. Use a SELECT statement to verify that you have three rows of data. For example: `select * from OrderInfo`

Next Steps

You have successfully created the subscription data that will drive report distribution and vary the report output for each subscriber. Next, you will modify the data source properties of the report you will distribute to subscribers. Modifying the data source properties is done to prepare the report for data-driven subscription delivery. You will also modify the report design to include a parameter that the subscription will use with the subscriber data. Lesson 2: Modifying the Report Data Source Properties.

See Also

Create a Data-Driven Subscription (SSRS Tutorial)

Create a Database

Create a Basic Table Report (SSRS Tutorial)

Lesson 2: Modifying the Report Data Source Properties

2/11/2019 • 3 minutes to read • Edit Online

In this lesson, you will use Report Manager to select a report that will be delivered to recipients. The data-driven subscription that you will define will distribute the **Sales Order** report created in the tutorial Create a Basic Table Report (SSRS Tutorial). In the steps that follow, you will modify the data source connection information used by the report to get data. Only reports that use **stored credentials** to access a report data source can be distributed through a data-driven subscription. Stored credentials are necessary for unattended report processing.

You will also modify the dataset and report to use a parameter to filter the report on the `[Order]` so the subscription can output different instances of the report for specific orders and rendering formats.

In this topic:

- To Modify the Data Source Properties
- To Modify the AdventureWorksDataset
- To Add a Report Parameter and Republish the Report
- To Re-deploy the Report

To Modify the Data Source Properties

1. Start Report Manager (SSRS Native Mode) with administrator privileges, for example, right-click the icon for Internet Explorer and click **Run as administrator**.
2. Browse to the folder containing the **Sales Orders** report and in the context menu of the report, click **Manage**.

3. Click the **Data Sources** tab.
4. For **Connection Type**, select **Microsoft SQL Server**.
5. The custom data source connection string will be the following and it assumes that the sample database is on a local database server:

```
Data source=localhost; initial catalog=AdventureWorks2012
```

6. Click **Credentials stored securely in the report server**.
7. Type your user name (use the format *domain\user*) and password. If you do not have permission to access the **AdventureWorks2012** database, specify a login that does.
8. Click **Use as windows credentials when connecting to the data source**, and then click **OK**. If you are not using a domain account (for example, if you are using a SQL Server login), do not click this checkbox.
9. Click **Test Connection** to verify you can connect to the data source.
10. Click **Apply**.
11. View the report to verify that the report runs with the credentials you specified. To view the report, click the **View** tab. Note that once the report is open, you must select an Employee name and then click the **View Report** button to view the report.

To Modify the AdventureWorksDataset

1. Open the Sales Orders report in SQL Server Data Tools
2. Right-click the dataset `AdventureWorksDataset` and click **Dataset Properties**.
3. Add the statement `WHERE (UPPER(SalesOrderNumber) =UPPER(@OrderNumber) or @OrderNumber IS NULL)` before the `Group By` statement. The full query syntax is the following:

```
SELECT soh.OrderDate AS Date, soh.SalesOrderNumber AS [Order], pps.Name AS Subcat, pp.Name AS Product,
SUM(sd.OrderQty) AS Qty, SUM(sd.LineTotal)  AS LineTotal
FROM Sales.SalesPerson AS sp INNER JOIN
  Sales.SalesOrderHeader AS soh ON sp.BusinessEntityID = soh.SalesPersonID INNER JOIN
   Sales.SalesOrderDetail AS sd ON sd.SalesOrderID = soh.SalesOrderID INNER JOIN
   Production.Product AS pp ON sd.ProductID = pp.ProductID
INNER JOIN
   Production.ProductSubcategory AS pps ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
INNER JOIN
    Production.ProductCategory AS ppc ON ppc.ProductCategoryID = pps.ProductCategoryID

WHERE (UPPER(SalesOrderNumber) =UPPER(@OrderNumber) or  @OrderNumber IS NULL)

GROUP BY ppc.Name, soh.OrderDate, soh.SalesOrderNumber, pps.Name, pp.Name, soh.SalesPersonID
HAVING (ppc.Name = 'Clothing')
```

4. Click **OK**

To Add a Report Parameter and Republish the Report

1. In the **Report Data** pane click **New** and then click **Parameter...**
2. In **Name**, type `OrderNumber`.
3. In **Prompt**, type `OrderNumber`.
4. Select **Allow blank value ("")**.
5. Select **Allow null value**.
6. Click **OK**. The parameter will be added to the **Report Data pane** and it will look like the following image:

7. Click the **Preview** tab to run the report.Note the parameter input box at the top of the report. You can either:

   - Click View Report to see the full report without using a parameter.
   - Unselect the Null option and type an order number, for example so71949 to view only the one order in the report.

8. Re-deploy the report so the subscription configuration in the next lesson can utilize the changes you made in this lesson. For more information on the project properties used in the table tutorial, see section 'To Publish the Report to the Report Server (Optional)' of Lesson 6: Adding Grouping and Totals (Reporting Services).

To Re-deploy the Report

1. Re-deploy the report so the subscription configuration in the next lesson can utilize the changes you made in this lesson. For more information on the project properties used in the table tutorial, see section 'To Publish the Report to the Report Server (Optional)' of Lesson 6: Adding Grouping and Totals (Reporting Services).

2. On the toolbar click **Build** and then click **Deploy tutorial**.

Next Steps

You successfully configured the report to get data using stored credentials. Next, you specify the subscription using the Data-Driven Subscription pages in Report Manager. See Lesson 3: Defining a Data-Driven Subscription.

See Also

Manage Report Data Sources
Specify Credential and Connection Information for Report Data Sources
Create a Data-Driven Subscription (SSRS Tutorial)
Create a Basic Table Report (SSRS Tutorial)

Lesson 3: Defining a Data-Driven Subscription

2/11/2019 • 3 minutes to read • Edit Online

In this lesson, you use the data-driven subscription pages to connect to a subscription data source, build a query that retrieves subscription data, and map the result set to report and delivery options.

> **NOTE**
> Before you start, verify that SQL Server Agent service is running. If it is not running, you cannot save the subscription.

This lesson assumes you completed Lesson 1 and Lesson 2 and that the report data source uses stored credentials. For more information, see Lesson 2: Modifying the Report Data Source Properties

In this topic:

- Start the Data-Driven Subscription Wizard
- Step 1 - Define a description
- Step 2 - Define a Connection to the Subscriber Data Source
- Step 3 - Define a Query to Retrieve Subscriber data
- Step 4 - Set Delivery Options
- Step 5 - Configure a Parameter Value to Very Report Output
- Step 6 - To Schedule a Subscription

Start the Data-Driven Subscription Wizard

1. In Report Manager, click **Home**, and navigate to the folder containing the **Sales Orders** report.
2. In the context menu of the report, click **Manage**, and then click the **Subscriptions** tab.
3. Click **New Data-driven Subscription**. If you do not see this button, you do not have Content Manager permissions.

Step 1 - Define a description

1. Type **Sales Order delivery** in description.
2. Select **Windows FileShare** for **Specify how recipients are notified**.
3. Select **Specify for this subscription only**, and then click **Next**.

Step 2 - Define a Connection to the Subscriber Data Source

1. Select **Microsoft SQL Server** as the data source type.
2. In Connection string, type the following connection string:

```
data source=localhost; initial catalog=Subscribers
```

> **NOTE**
> Subscribers is the database you created in lesson 1.

3. Click **Credentials stored securely in the report server**.
4. In **User Name** and **Password**, type your domain user name and password. Include both the domain and user account when specifying **User Name**.

> **NOTE**
> Credentials used to connect to a subscriber data source are not passed back to Management Studio. If you modify the subscription later, you must retype the password used to connect to the data source.

5. Select **Use as windows credentials when connecting to the data source**, and then click **Next**.

Step 3 - Define a Query to Retrieve Subscriber data

1. In the query box, type the following query:

```
Select * from OrderInfo
```

2. Specify a time-out of 30 seconds.
3. Click **Validate**, and then click **Next**.

Step 4 - Set Delivery Options

1. For **File name**, select **Get the value from the database**. Select the field **Order**.
2. For **Path**, select **Specify a static value**. In Setting Value, type the name of a public file share for which you have write permissions (for example, `\\mycomputer\public\myreports`).
3. For **Render Format**, select **Get the value from the database**. Select **Format**.
4. For **Write mode**, select **Specify a static value** and select **AutoIncrement**.
5. For **File Extension**, select **Specify a static value** and select **True**.
6. For **User name**, select **Specify a static value**. Type your domain user account. Enter it in this format: `<domain>\<account>`. The user account needs to have permissions to the path you configured in the previous steps.
7. For **Password**, select **Specify a static value**. Type your password. Be sure that you type the password carefully. The wizard does not validate the password.
8. Click **Next.**

Step 5 - Configure a Parameter Value to Very Report Output

1. For **OrderNumber**, select **Get the value from the database**. In Value, select **Order**. Click **Next.**

Step 6 - To Schedule a Subscription

1. Click **On a schedule created for this subscription**, and then click **Next**.

2. In **Schedule Details**, click **Once**.

3. Specify a start time that is a few minutes ahead of the current time.

4. Click **Finish**.

Next Steps

When the subscription runs, four report files will be delivered to the file share you specified, one for each order in the *Subscribers* data source. Each delivery should be unique in terms of data (the data should be order-specific), rendering format, and file format. You can open each report from the shared folder to verify that each version is customized based on the subscription options you defined.

The subscription page in Report Manager will contain the **Last Run** date and **Status** of the subscription.

> **NOTE**
>
> Refresh the page after the subscription runs to see the updated information.

This step concludes the tutorial "Defining a Data-Driven Subscription". To learn more about other Reporting Services tutorials, see Reporting Services Tutorials (SSRS).

See Also

Create a Data-Driven Subscription (SSRS Tutorial)
Subscriptions and Delivery (Reporting Services)
Data-Driven Subscriptions
Create, Modify, and Delete a Data-Driven Subscription
Use an External Data Source for Subscriber Data (Data-Driven Subscription)

Create a Drillthrough (RDLC) Report with Parameters using ReportViewer (SSRS Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

A drillthrough report is a report that a user opens by clicking a link within another report. Drillthrough reports commonly contain details about an item that is contained in an original summary report. This tutorial will walk you through the following lessons of creating a drillthrough report with parameters and a query, in local mode reporting.

Requirements

To use this walkthrough, you must have access to the **AdventureWorks2008** sample database. The query used in this walkthrough will also work with **AdventureWorks2012** database. For more information about how to get the **AdventureWorks2008** sample database, see Walkthrough: Installing the AdventureWorks Database for Microsoft Visual Studio 2010.

This walkthrough assumes that you are familiar with Transaction-SQL queries and ADO.NET DataSet and DataTable objects.

Use Visual Studio 2010, or Visual Studio 2012, and the ASP.NET website template to create an ASP.NET webpage with a ReportViewer control. The control is configured to view a report that you create. For this walkthrough, you create the application in Microsoft Visual C#.

Tasks

Lesson 1: Create a New Web Site
Lesson 2: Define a Data Connection and Data Table for Parent Report
Lesson 3: Design the Parent Report using the Report Wizard
Lesson 4: Define a Data Connection and Data Table for Child Report
Lesson 5: Design the Child Report using the Report Wizard
Lesson 6: Add a ReportViewer Control to the Application
Lesson 7: Add Drillthrough Action on Parent Report
Lesson 8: Create a Data Filter
Lesson 9: Build and Run the Application

See Also

Reporting Services Tutorials (SSRS)
Design Reports with Report Designer (SSRS)

Lesson 1: Create a New Web Site

2/11/2019 • 2 minutes to read • Edit Online

In this lesson you'll learn how to create a new website project using the ASP.NET website template for Visual C#.

To create a new website

1. Open Microsoft Visual Studio 2010 or Microsoft Visual Studio 2012.
2. On the **File** menu, point to **New**, and select **Web Site**.
3. In the **New Web Site** dialog box, in the **Installed Templates** pane, select **Visual C#** and then choose **ASP.NET web site**.
4. In the **Location** box, specify a project directory and click **OK**.

The website project opens.

Next Task

You've successfully created a new website project. Next, you will create a data connection and a data table for the parent report. See Lesson 2: Define a Data Connection and Data Table for Parent Report.

Lesson 2: Define a Data Connection and Data Table for Parent Report

2/11/2019 • 2 minutes to read • Edit Online

After you create a new website project using the ASP.NET website template for Visual C#, your next step is to create a data connection and a data table for the parent report. In this tutorial the data connection is to the AdventureWorks2008 database. You also have the option of connecting to the AdventureWorks2012 database.

To define a data connection and Data Table by adding a DataSet (for parent report)

1. On the **Website** menu, select **Add New Item**.
2. In the **Add New Item** dialog box, select **DataSet** and click **Add**. When prompted you should add the item to the **App_Code** folder by clicking **Yes**.

   This adds a new XSD file **DataSet1.xsd** to the project and opens the DataSet Designer.
3. From the Toolbox window, drag a **TableAdapter** control to the design surface. This launches the **TableAdapter** Configuration Wizard.
4. On the **Choose Your Data Connection** page, click **New Connection**.
5. If this is the first time you've created a data source in Visual Studio, you will see the **Choose Data Source** page. In the **Data Source** box, select **Microsoft SQL Server**.
6. In the **Add Connection** dialog box, perform the following steps:
   a. In the **Server name** box, enter the server where the **AdventureWorks2008** database is located.

      The default SQL Server Express instance is **(local)\sqlexpress**.

   b. In the **Log on to the server** section, select the option that provides you access to the data. **Use Windows Authentication** is the default.

   c. From the **Select or enter a database name** drop-down list, click **AdventureWorks2008**.

   d. Click **OK**, and then click **Next**.
7. If you selected **Use SQL Server Authentication** in the Step 6 (b), select the option whether to include the sensitive data in the string or set the information in your application code.
8. On the **Save the Connection String to the Application Configuration File** page, type in the name for the connection string or accept the default **AdventureWorks2008ConnectionString**. Click **Next**.
9. On the **Choose a Command Type** page, select **Use SQL Statements**, and then click **Next**.
10. On the **Enter a SQL Statement** page, enter the following Transact-SQL query to retrieve data from the **AdventureWorks2008** database, and then click **Next**.

    ```
    SELECT ProductID, Name, ProductNumber, SafetyStockLevel, ReorderPoint FROM  Production.Product Order By
    ProductID
    ```

    You can also create the query by clicking **Query Builder**, and then verify the query by clicking **Execute Query**. If the query does not return the expected data, you might be using an earlier version of AdventureWorks. For more information about installing the **AdventureWorks2008** version of

AdventureWorks, see Walkthrough: Installing the AdventureWorks Database.

11. On the **Choose Methods to Generate** page, be sure to uncheck **Create methods to send updates directly to the database (GenerateDBDirectMethods)**, and then click **Finish**.

> **WARNING**
>
> Be sure to uncheck Create

You have now completed configuring the ADO.NET DataTable object as the data source for your report. On the DataSet Designer page in Visual Studio, you should see the DataTable object you added, listing the columns specified in the query. DataSet1 contains the data from the Product table, based on the query.

12. Save the file.

13. To preview the data, click **Preview Data** on the **Data** menu, and then click **Preview**.

Next Task

You have successfully created a data connection and a data table for the parent report. Next, you will design the parent report using the Report Wizard.

Lesson 3: Design the Parent Report using the Report Wizard

2/11/2019 • 2 minutes to read • Edit Online

After you create a data connection and a data table for the parent report, your next step is to design the parent report using the Report Wizard in Report Designer. For more information about Report Designer, see Design Reports with Report Designer (SSRS).

To design the parent report using the Report Wizard

1. Make sure that the top-level website is selected in **Solution Explorer**.
2. Right-click on the website and select **Add New Item**.
3. In the **Add New Item** dialog box, select **Report Wizard**, enter a name for the report file, and then click **Add**.

   This launches the Report Wizard.
4. On the **Dataset Properties** page, in the **Data source** box, select the **DataSet1** you created in Lesson 2: Define a Data Connection and Data Table for Parent Report.
   The **Available datasets** box is automatically updated with the **DataTable** you created above.
5. Click **Next**.
6. In the **Arrange Fields** page do the following:
   a. Drag **ProductID**, **Name**, **ProductNumber**, **SafetyStockLevel**, and **ReorderLevel** from **Available fields** to the **Values** box.
   b. Click the arrow next to **Sum(ProductID)**, **Sum(SafetyStockLevel)**, **Sum(ReorderLevel)** and clear the **Sum** selection.
7. Click **Next** twice, then click **Finish** to close the **Report Wizard**.

   You've now created the .rdlc file. The file opens in Report Designer. The tablix you designed is now displayed in the design surface.
8. Save the .rdlc file.

Next Task

You have successfully designed the parent report using the Report Wizard. Next, you will create a data connection and a data table for the child report.

Lesson 4: Define a Data Connection and Data Table for Child Report

2/11/2019 • 2 minutes to read • Edit Online

After you design the parent report, you next step is to create a data connection and a data table for the child report. In this tutorial the data connection is to the AdventureWorks2008 database. You also have the option of connecting to the AdventureWorks2012 database.

To define a data connection and DataTable by adding a DataSet (for child report)

1. On the **Website** menu, click **Add New Item**.
2. In the **Add New Item** dialog box, click **DataSet** and then click **Add**. When prompted, you should add the item to the **App_Code** folder by clicking **Yes**.

   This adds a new XSD file **DataSet2.xsd** to the project and opens the DataSet Designer.
3. From the Toolbox window, drag a **TableAdapter** control to the design surface. This launches the **TableAdapter** Configuration Wizard.
4. On the **Choose Your Data Connection** page, click **New Connection**.
5. In the **Add Connection** dialog box, perform the following steps:

   a. In the **Server name** box, enter the server where the **AdventureWorks2008** database is located.

      The default SQL Server Express instance is **(local)\sqlexpress**.

   b. In the **Log on to the server** section, select the option that provides you access to the data. **Use Windows Authentication** is the default.

   c. From the **Select or enter a database name** drop-down list, click **AdventureWorks2008**.

   d. Click **OK**, and then click **Next**.
6. If you selected **Use SQL Server Authentication** in Step 5 (b), select the option whether to include the sensitive data in the string or set the information in your application code.
7. On the **Save the Connection String to the Application Configuration File** page, type in the name for the connection string or accept the default **AdventureWorks2008ConnectionString**. Click **Next**.
8. On the **Choose a Command Type** page, select **Use SQL Statements**, and then click **Next**.
9. On the **Enter a SQL Statement** page, enter the following Transact-SQL query to retrieve data from the **AdventureWorks2008** database, and then click **Next**.

   ```
   SELECT PurchaseOrderID, PurchaseOrderDetailID, OrderQty, ProductID, ReceivedQty, RejectedQty,
   StockedQty FROM Purchasing.PurchaseOrderDetail
   ```

   You can also create the query by clicking **Query Builder**, and then verify the query by clicking **Execute Query** button. If the query does not return the expected data, you might be using an earlier version of AdventureWorks. For more information about installing the **AdventureWorks2008** version of AdventureWorks, see Walkthrough: Installing the AdventureWorks Database.
10. On the **Choose Methods to Generate** page, uncheck **Create methods to send updates directly to the**

database (GenerateDBDirectMethods), and then click **Finish**.

You have now completed configuring the ADO.NET DataTable as data source for your report. On the DataSet Designer page in Visual Studio, you should see the **DataTable** you added, listing the columns specified in the query. DataSet2 contains the data from the PurhcaseOrderDetail table, based on the query.

11. Save the file.
12. To preview the data, click **Preview Data** on the **Data** menu, and then click **Preview**.

Next Task

You have successfully created a data connection and data table for the child report. Next, you will design the child report using the Report Wizard.

Lesson 5: Design the Child Report using the Report Wizard

2/11/2019 • 2 minutes to read • Edit Online

After you create a data connection and data table for the child report, your next step is to design the child report using the Report Wizard in Report Designer. For more information about Report Designer, see Design Reports with Report Designer (SSRS).

To design the child report using the Report Wizard

1. Make sure that the top-level website is selected in **Solution Explorer**.
2. Right-click on the website and select **Add New Item**.
3. In the **Add New Item** dialog box, click **Report Wizard**, enter a name for the report file, and then click **Add**.

   This launches the Report Wizard.
4. In the **Dataset Properties** page, in the **Data source** box, click **DataSet2**.

   The **Available datasets** box is automatically updated with the DataTable you created.
5. Click **Next**.
6. In the **Arrange Fields** page do the following:

   a. Drag **ProductID**, **PurchaseOrderID**, **PurchaseOrderDetailID**, **OrderQty**, **ReceivedQty**, **RejectedQty**, and **StockedQty** from **Available Fields** to the **Values** box.

   b. Click the arrow next to **Sum(ProductID)**, **Sum(PurchaseOrderID)**, **Sum(PurchaseOrderDetailID)**, **Sum(OrderQty)**, **Sum(ReceivedQty)**, **Sum(RejectedQty)**, and **Sum(StockedQty)** and clear the **Sum** selection.
7. Click **Next** twice, then click **Finish** to close the **Report Wizard**.

   You've now created the .rdlc file. The file opens in Report Designer. The tablix you designed is now displayed in the design surface.
8. With the .rdlc file open, add a parameter by doing the following:

   a. Click **Parameters** in the **Report Data** pane, and then click **Add Parameters**.

   b. Enter **productid** in the **Name** box.

   c. Confirm that **Integer** is selected in the **Data Type** list box.

   d. Click **OK**.
9. Save the .rdlc file.

Next Task

You have successfully designed the child report by using the Report Wizard. Next, you will add a ReportViewer control to the website application.

Lesson 6: Add a ReportViewer Control to the Application

2/11/2019 • 2 minutes to read • Edit Online

After you design the child report by using the Report Wizard, your next step is to add a ReportViewer control to the website application.

To add a ReportViewer control to the application

1. In **Solution Explorer**, right-click **Default.aspx**, and then click **View Designer**.
2. From the **AJAX Extensions** group in the **Toolbox** window, drag a **ScriptManager** control to the design surface.
3. From the **Reporting** group, drag a **ReportViewer** control to the design surface below the **ScriptManager** control.
4. Open the **ReportViewer Tasks** window by clicking the arrow in the top right-hand corner of the **ReportViewer** control.
5. In the **Choose Report** box, select Parent report you created.

   When you select a report, instances of data sources used in the report are created automatically. Code is generated to instantiate each DataTable (and its DataSet container). An ObjectDataSource control is added to the design surface, corresponding to each data source used in the report. This data source control is configured automatically.

   If you're using Microsoft Visual Studio 2012, make sure that the ObjectDataSource control is bound with DataSet1 that is fully qualified with the project namespace, if the fully qualified name is listed in the **Choose your business object** drop-down list box (for example, Projectnamespace.DataSet1TableAdapters.ProductTableAdapter). You access the list box by right-clicking ObjectDataSource, and then clicking **Configure Data Source**.
6. On the Build menu, click Build website.

   The report is compiled and any errors such as a syntax error in a report expression appear in the **Error List** area. Click **Error List** at the bottom of the Visual Studio window to display the **Error List** area.

Next Task

You have successfully added a ReportViewer control to the website application. Next, you will add a drillthrough action on the parent report.

Lesson 7: Add Drillthrough Action on Parent Report

2/11/2019 • 2 minutes to read • Edit Online

After you add a ReportViewer control to the website application, your next step is to add a drillthrough action on the parent report.

To add drillthrough action on the parent report

1. Go to the parent report.
2. Click the textbox that holds the value of **Name**.
3. Right click the textbox, and then click **Text Box Properties**.
4. Go to the **Action** tab, and then select the **Go to report** option.
5. Enter the name of the child report in the **Specify a report** section.
6. Click **Add** under **Use these parameters to run the report** section.
7. Type **productid** in the **name** box, and then click **ProductID** in the **Value** drop-down list.
8. Click **Ok** to finish.

Next Task

You have successfully added a drillthrough action on the parent report. Next, you will create a data filter for the data table that you defined for the child report.

Lesson 8: Create a Data Filter

2/11/2019 • 7 minutes to read • Edit Online

After you add a drillthrough action on the parent report, your next step is to create a data filter for the data table that you defined for the child report.

You can create a table-based filter **or** a query filter for the drillthrough report. This lesson provides instructions for both options.

Table-Based Filter

You need to complete the following tasks to implement a table-based filter.

- Add a filter expression to the tablix in the child report.
- Create a function that selects unfiltered data from the `PurchaseOrderDetail` table.
- Add an event handler that binds the `PurchaseOrderDetail` DataTable to the child report.

To add a filter expression to the tablix in the child report

1. Open the child report.
2. Select a column heading in the tablix, right-click the gray cell that appears above the column heading, and then click **Tablix Properties**.
3. Click on the **Filters** page, and then click **Add**.
4. In the **Expression** filed, click `ProductID` from the drop-down list. This is the column to which you apply the filter.
5. Click the equal (=) operator in the **Operator** drop-down list.
6. Click the expression button next to the **Value** field, click **Parameters** in the **Category** area, and then double-click `productid` in the **Values** area. The **Set expression for: Value** field should now contain expression similar to **=Parameters!productid.Value**.
7. Click **OK,** and **OK** again in the **Tablix Properties** dialog box.
8. Save the .rdlc file.

To create a function that selects unfiltered data from the PurchaseOrdeDetail table

1. In Solution Explorer, expand Default.aspx, and then double click Default.aspx.cs.
2. Create a new function that accepts a parameter, `productid`, of type Integer and returns a `datatable` object, and does the following.
   a. Creates an instance of the dataset, `DataSet2`, which was created in Step 2 of Lesson 4: Define a Data Connection and Data Table for Child Report.
   b. Create a connection to the SqlServer database to execute the query defined in **Lesson 4: Define a Data Connection and DataTable for Child Report**.
   c. The query will return unfiltered data.
   d. Fill the DataSet instance with the unfiltered data by executing the query.
   e. Return the `PurchaseOrderDetail` DataTable.

The function will look similar to the one below, (This is just for your reference. You can follow any pattern that you want, to fetch the necessary data for child report).

```
/// <summary>
    /// Function to query PurchaseOrderDetail table, fetch the
    /// unfiltered data and bind it with the Child report
    /// </summary>
    /// <returns>A dataTable of type PurchaseOrderDetail</returns>
    private DataTable GetPurchaseOrderDetail()
    {
        try
        {
            //Create the instance for the typed dataset, DataSet2 which will
            //hold the [PurchaseOrderDetail] table details.
            //The dataset was created as part of the tutorial in Step 4.
            DataSet2 ds = new DataSet2();

            //Create a SQL Connection to the AdventureWorks2008 database using Windows
Authentication.
            using (SqlConnection sqlconn = new SqlConnection("Data Source=.;Initial
Catalog=Adventureworks;Integrated Security=SSPI"))
            {
                //Building the dynamic query with the parameter ProductID.
                SqlDataAdapter adap = new SqlDataAdapter("SELECT PurchaseOrderID,
PurchaseOrderDetailID, OrderQty, ProductID, ReceivedQty, RejectedQty, StockedQty FROM
Purchasing.PurchaseOrderDetail ", sqlconn);
                //Executing the QUERY and fill the dataset with the PurchaseOrderDetail table
data.
                adap.Fill(ds, "PurchaseOrderDetail");
            }

            //Return the PurchaseOrderDetail table for the Report Data Source.
            return ds.PurchaseOrderDetail;
        }
        catch
        {
            throw;
        }
    }
```

To add an event handler that binds the PurchaseOrderDetail DataTable to the child report

1. Open Default.aspx.
2. Right click the ReportViewer control, and then click **Properties.**
3. On the **Properties** page, click the **Events** icon.
4. Double click the **Drillthrough** event.

   This will add an event handler section in the code, which will look similar to the below block.

```
protected void ReportViewer1_Drillthrough(object sender,
Microsoft.Reporting.WebForms.DrillthroughEventArgs e)
{
}
```

5. Complete the event handler. It should include the following functionalty.
   a. Fetch the child report object reference from the *DrillthroughEventArgs* parameter.
   b. Call the function, `GetPurchaseOrderDetail`
   c. Bind the `PurchaseOrderDetail` DataTable with the report's corresponding data source.

The completed event handler code will look similar to the following.

```
protected void ReportViewer1_Drillthrough(object sender,
Microsoft.Reporting.WebForms.DrillthroughEventArgs e)
    {
        try
        {
             //Get the instance of the Target report, in our case the JumpReport.rdlc.
            LocalReport report = (LocalReport)e.Report;

             //Binding the DataTable to the Child report dataset.
            //The name DataSet1 which can be located from,
            //Go to Design view of Child.rdlc, Click View menu -> Report Data
            //You'll see this name under DataSet2.
            report.DataSources.Add(new ReportDataSource("DataSet1", GetPurchaseOrderDetail()));
        }
        catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
    }
```

6. Save the file.

Query Filter

You need to complete the following tasks to implement a query filter.

- Create a function that selected filtered data from the `PurchaseOrderDetail` table.
- Add an event handler that retrieves parameter values and binds the `PurchaseOrdeDetail` DataTable to the child report.

To create a function that selects filtered data from the PurchaseOrderDetail table

1. In Solution Explorer, expand Default.aspx, and then double click Default.aspx.cs.
2. Create a new function that accepts a parameter, `productid`, of type Integer and returns a `datatable` object and does the following.
   a. Creates an instance of the dataset, `DataSet2`, which was created in Step 2 of Lesson 4: Define a Data Connection and Data Table for Child Report.
   b. Create a connection to the SqlServer database to execute the query defined **Lesson 4: Define a Data Connection and DataTable for Child Report**.
   c. The query will include a parameter, `productid`, to make sure the data returned is filtered based on the `ProductID` selected in the parent report.
   d. Fill the DataSet instance with the filtered data by executing the query.
   e. Return the `PurchaseOrderDetail` DataTable.

   The function will look similar to the one below, (This is just for your reference. You can follow any pattern that you want, to fetch the necessary data for child report).

```
/// <summary>
    /// Function to query PurchaseOrderDetail table and filter the
    /// data for a specific ProductID selected in the Parent report.
    /// </summary>
    /// <param name="productid">Parameter passed from the Parent report to filter data.</param>
    /// <returns>A dataTable of type PurchaseOrderDetail</returns>
    private DataTable GetPurchaseOrderDetail(int productid)
    {
        try
        {
            //Create the instance for the typed dataset, DataSet2 which will
            //hold the [PurchaseOrderDetail] table details.
            //The dataset was created as part of the tutorial in Step 4.
            DataSet2 ds = new DataSet2();

            //Create a SQL Connection to the AdventureWorks2008 database using Windows
Authentication.
            using (SqlConnection sqlconn = new SqlConnection("Data Source=.;Initial
Catalog=Adventureworks;Integrated Security=SSPI"))
            {
                //Building the dynamic query with the parameter ProductID.
                SqlDataAdapter adap = new SqlDataAdapter("SELECT PurchaseOrderID,
PurchaseOrderDetailID, OrderQty, ProductID, ReceivedQty, RejectedQty, StockedQty FROM
Purchasing.PurchaseOrderDetail where ProductID = " + productid, sqlconn);
                //Executing the QUERY and fill the dataset with the PurchaseOrderDetail table
data.
                adap.Fill(ds, "PurchaseOrderDetail");
            }

            //Return the PurchaseOrderDetail table for the Report Data Source.
            return ds.PurchaseOrderDetail;
        }
        catch
        {
            throw;
        }
    }
```

To add an event handler that retrieves parameter values and binds the PurchaseOrdeDetail DataTable to the child report

1. Open Default.aspx.
2. Right-click the ReportViewer control, and then click **Properties**.
3. On the **Properties** pane, click the **Events** icon.
4. Double click the **Drillthrough** event.

   This will add an event handler section in the code that will look similar to the following.

   ```
   protected void ReportViewer1_Drillthrough(object sender,
   Microsoft.Reporting.WebForms.DrillthroughEventArgs e)
   {
   }
   ```

5. Complete the event handler. It should include the following functionality.

   a. Fetch the Child report object reference from the *DrillthroughEventArgs* parameter.

   b. Get the child report parameter list from the child report object fetched.

   c. Iterate through the parameter's collection and retrieve the value for the parameter, `ProductID`, passed from the parent report.

d. Call the function, `GetPurchaseOrderDetail`, and pass the value for parameter `ProductID`.

e. Bind the `PurchaseOrderDetail` DataTable with the Report's corresponding data source.

The completed event handler code will look similar to the following.

```
protected void ReportViewer1_Drillthrough(object sender,
Microsoft.Reporting.WebForms.DrillthroughEventArgs e)
    {
        try
        {
            //Variable to store the parameter value passed from the MainReport.
            int productid = 0;

            //Get the instance of the Target report, in our case the JumpReport.rdlc.
            LocalReport report = (LocalReport)e.Report;

            //Get all the parameters passed from the main report to the target report.
            //OriginalParametersToDrillthrough actually returns a Generic list of
            //type ReportParameter.
            IList<ReportParameter> list = report.OriginalParametersToDrillthrough;

            //Parse through each parameters to fetch the values passed along with them.
            foreach (ReportParameter param in list)
            {
                //Since we know the report has only one parameter and it is not a multivalued,
                //we can directly fetch the first value from the Values array.
                productid = Convert.ToInt32(param.Values[0].ToString());
            }

            //Binding the DataTable to the Child report dataset.
            //The name DataSet1 which can be located from,
            //Go to Design view of Child.rdlc, Click View menu -> Report Data
            //You'll see this name under DataSet2.
            report.DataSources.Add(new ReportDataSource("DataSet1",
GetPurchaseOrderDetail(productid)));
        }
        catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
    }
```

6. Save the file.

Next Task

You have successfully created a data filter for the data table that you defined for the child report. Next, you will build and run the website application.

Lesson 9: Build and Run the Application

2/11/2019 • 2 minutes to read • Edit Online

After you create a data filter for the data table, your next step is to build and run the website application.

To build and run the application

1. Press **CTRL+F5** to run the Default.aspx page without debugging, or press F5 to run the page with debugging.

   As part of the build process, the report is compiled and any errors found (such as a syntax error in an expression used in the report) are added to the **Task List** that is located at the bottom of the Visual Studio window.

   The webpage appears in the browser. The ReportViewer control displays the report. You can use the toolbar to browse through the report, zoom, and export the report to Excel.

2. Hover the mouse over any of the rows under **Name** column. The mouse cursor will display a Hand symbol.

3. Click a value in the **Name** column. The child report is shown with the corresponding filtered data.

4. Click the icon, **Go back to parent report**, in the **ReportViewer** tool bar to navigate back to the **Parent** report.

5. Close the browser to exit.

Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

The following tutorial shows you how to access the Report Server Web service from an application created with Microsoft Visual Basic or Microsoft Visual C#.

What You Will Learn

During the course of this tutorial, you will complete the following:

- Create a client application using the Microsoft Visual Studio 2010 Console Application project template.
- Add a Web reference for the Report Server Web service.
- Write code to access the Web service.
- Run the console application in debug mode.

Requirements

To complete the tutorial, you must have the following:

- Microsoft SQL Server 2012 Reporting Services.
- Microsoft Visual Studio 2010 or a similar Microsoft .NET Framework-compatible development tool.
- Sufficient permissions to be able to access the SQL Server Report Server Web service on the computer where your report server is located.
- A report installed on your report server. This tutorial uses the sample report, Company Sales. For more information about sample reports, see SQL Server Reporting Services Product Samples.

> **NOTE**
>
> The samples are not installed automatically during setup, but you can install them at any time. For information about samples, see SQL Server Product Samples.

Estimated time to complete the tutorial: 60 minutes

Tasks

Lesson 1: Creating the Web Service Client Project

Lesson 2: Adding a Web Reference

Lesson 3: Accessing the Web Service

Lesson 4: Running the Application (VB-VC#)

Lesson 1: Creating the Web Service Client Project

2/11/2019 • 2 minutes to read • Edit Online

For this walkthrough, you will create a simple console application that accesses the Report Server Web service. This walkthrough assumes you are developing in Visual Studio.

To create a console application

1. On the **File** menu, point to **New**, and then click **Project** to open the **New Project** dialog box.
2. In the left pane, under **Installed Templates**, click either **Visual Basic** or the **Visual C#** node, and select a category of project types from the expanded list.
3. Choose the **Console Application** project type.
4. In the **Name** box, enter a name for your project. Type the name `GetPropertiesSample`.
5. In the **Location** box, enter the path where you want to save your project, or click **Browse** to navigate to the folder.
6. Click **OK**. A collapsed view of your project appears in Solution Explorer.

   In Solution Explorer, expand the project node. A file with the default name of Program.cs (Module1.vb for Microsoft Visual Basic) has been added to your project.

See Also

Lesson 2: Adding a Web Reference
Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)

Lesson 2: Adding a Web Reference

2/11/2019 • 2 minutes to read • Edit Online

Web service discovery is the process by which a client locates a Web service and obtains its service description. The process of Web service discovery in Visual Studio involves interrogating a Web site following a predetermined algorithm. The goal of the process is to locate the service description, which is an XML document that uses the Web Services Description Language (WSDL).

The service description describes what services are available and how to interact with those services. Without a service description, it is impossible to programmatically interact with a Web service.

Your application must have a means to communicate with the Web service and to locate it at run time. Adding a Web reference to your project for the Web service does this by generating a proxy class that interfaces with the Web service and provides a local representation of the Web service. For more information, see "How to: Generate an XML Web Service Proxy" in your Visual Studio documentation.

To add a Web reference

1. On the **Project** menu, click **Add Service Reference**.
2. In the **Add Service Reference** dialog box, click **Advanced**.
3. In the **Service Reference Settings** dialog box, click **Add Web Reference**.
4. In the **URL** box of the **Add Web Reference** dialog box, type the URL to obtain the service description of the Report Server Web service, such as http://localhost/reportserver/reportservice2010.asmx. Then click the **Go** button to retrieve information about the Web service.

   - or -

   If the Report Server Web service exists on the local machine, click the **Web services on the local machine** link in the browser pane. Then click the link for the ReportService2010 Web service from the list provided.
5. In the **Web reference name** box, rename the Web reference to ReportService2010, which is the namespace you will use for this Web reference.
6. Click **Add Reference** to add a Web reference for the target Web service.

   Visual Studio downloads the service description and generates a proxy class to interface between your application and the Report Server Web service. You will also need to add a reference to the Services namespace for your Web reference to work.
7. On the Project menu, click **Add Reference**.
8. In the **Add Reference** dialog box, in the **.NET** tab, select **System.Web.Services**, then click **OK**.

For more information, see Accessing the SOAP API.

See Also

Report Server Web Service
Lesson 3: Accessing the Web Service
Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)

Lesson 3: Accessing the Web Service

2/11/2019 • 2 minutes to read • Edit Online

After you add a reference for the Report Server Web service to your project, the next step is to create an instance of the Web service's proxy class. You can then access the methods of the Web service by calling the methods in the proxy class. When your application calls these methods, the proxy class code generated by Visual Studio handles the communications between your application and the Web service.

First, you will create an instance of the Web service's proxy class, ReportingService2010. Next, you will make a call to the Web service's GetProperties method using the proxy class. You will use the call to retrieve the name and description of one of the sample reports, Company Sales.

> **NOTE**
>
> When accessing a Web service running on SQL Server Express with Advanced Services, you must append "$SQLExpress" to the "ReportServer" path. For example:
>
> `http://<Server Name>/reportserver$sqlexpress/reportservice2010.asmx"`

To access the Web service

1. You must first add the namespace to the Program.cs file (Module1.vb in Visual Basic) by adding a `using` (`Imports` in Visual Basic) directive to the code file. If you use this directive, you do not need to fully qualify the types in the namespace.

2. To do this, add the following code to the beginning of your code file:

```
Imports System
Imports GetPropertiesSample.ReportService2010
```

```
using System;
using GetPropertiesSample.ReportService2010;
```

3. Once you have entered the namespace directive to your code file, enter the following code in the Main method of your console application. Make sure to change the name of your server when setting the **Url** property of the web service instance:

```
Sub Main()
   Dim rs As New ReportingService2010
   rs.Credentials = System.Net.CredentialCache.DefaultCredentials
   rs.Url = "http://<Server Name>/reportserver/reportservice2010.asmx"

   Dim name As New [Property]
   name.Name = "Name"

   Dim description As New [Property]
   description.Name = "Description"

   Dim properties(1) As [Property]
   properties(0) = name
   properties(1) = description

   Try
      Dim returnProperties As [Property]() = rs.GetProperties( _
         "/AdventureWorks 2012 Sample Reports/Company Sales 2012", properties)

      Dim p As [Property]
      For Each p In returnProperties
          Console.WriteLine((p.Name + ": " + p.Value))
      Next p

   Catch e As Exception
      Console.WriteLine(e.Message)
   End Try
End Sub
```

```
static void Main(string[] args)
{
   ReportingService2010 rs = new ReportingService2010();
   rs.Credentials = System.Net.CredentialCache.DefaultCredentials;
   rs.Url = "http://<Server Name>/reportserver/reportservice2010.asmx";

   Property name = new Property();
   name.Name = "Name";

   Property description = new Property();
   description.Name = "Description";

   Property[] properties = new Property[2];
   properties[0] = name;
   properties[1] = description;

   try
   {
      Property[] returnProperties = rs.GetProperties(
      "/AdventureWorks 2012 Sample Reports/Company Sales 2012",properties);

      foreach (Property p in returnProperties)
      {
         Console.WriteLine(p.Name + ": " + p.Value);
      }
   }

   catch (Exception e)
   {
      Console.WriteLine(e.Message);
   }
}
```

4. Save the solution.

The walkthrough sample code uses the GetProperties method of the Web service to retrieve properties of the sample report, Company Sales 2012. The GetProperties method takes two arguments: the name of the report for which you want to retrieve property information and an array of **Property[]** objects that contains the names of properties whose values you want to retrieve. The method also returns an array of **Property[]** objects that contains the names and values of the properties specified in the properties argument.

> **NOTE**
>
> If you supply an empty **Property[]** array for the properties argument, all available properties are returned.

In the previous sample, the code uses the GetProperties method to return the name and description of the sample report, Company Sales 2012. The code then uses a `foreach` loop to write the properties and values to the console.

For more information about creating and using a proxy class for the Report Server Web service, see Creating the Web Service Proxy.

See Also

Lesson 4: Running the Application (VB-VC#)
Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)

Lesson 4: Running the Application (VB-VC#)

2/11/2019 • 2 minutes to read • Edit Online

Visual Studio offers several methods to build and run a console application from the IDE, such as:

- Start (with Debugging)
- Start without Debugging

To build and run the GetPropertiesSample

1. From the **Debug** menu, click **Start Without Debugging**. This ensures that the console window remains open after the program has finished executing.

   The application prints the following output to the console:

   ```
   Name: Company Sales
   Description: Adventure Works sales by quarter and product category. This report
   illustrates the use of a matrix data region that provides drilldown from summary
   data into detail data by showing and hiding rows. This report also illustrates
   the use of background images.
   Press any key to continue...
   ```

2. Press any key to close GetPropertiesSample.

See Also

GetProperties
Accessing the Report Server Web Service Using Visual Basic or Visual C# (SSRS Tutorial)

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)

2/11/2019 • 2 minutes to read • Edit Online

This tutorial illustrates how to use the XML Schema Definition Tool (Xsd.exe) to generate classes that allow you to serialize and deserialize report definition files (.rdl and .rdlc) with the Microsoft .NET Framework XmlSerializer class.

What You Will Learn

During the course of this tutorial, you will complete the following activities:

- Create an application using the Microsoft Visual Studio Console Application project template.
- Generate classes from the Report Definition Language (RDL) schema using the **xsd** tool.
- Connect to a report server and retrieve a report definition.
- Write code to update the report definition file.
- Save the updated report definition back to the report server.
- Run the RDL Schema Application (VB/C#).

> **NOTE**
> The code samples provided in this tutorial might fail for reports having no description. The failure is because the description property does not exist for the reports with description not specified.

Requirements

To complete the tutorial, you must have the following:

- Microsoft **Microsoft SQL Server 2014** Reporting Services.
- Microsoft Visual Studio 2010.
- Sufficient permissions to be able to access and publish reports to the Report Server Web service on the computer where your report server is located.
- The **AdventureWorks2012** sample database installed to an instance of SQL Server.
- A report installed on your report server. This tutorial uses the sample report, Company Sales 2012. For more information about sample reports, see SQL Server Reporting Services Product Samples.

> **NOTE**
> The samples are not installed automatically during setup, but you can install them at any time. For information about samples, see SQL Server Product Samples.

Estimated time to complete the tutorial: 30 minutes

Tasks

Lesson 1: Create the RDL Schema Visual Studio Project

Lesson 2: Generate Classes from the RDL Schema using the xsd Tool

Lesson 3: Load a Report Definition from the Report Server

Lesson 4: Update the Report Definition Programmatically

Lesson 5: Publish the Report Definition to the Report Server

Lesson 6: Run the RDL Schema Application (VB-C#)

See Also

Report Definition Language (SSRS)

Lesson 1: Create the RDL Schema Visual Studio Project

2/11/2019 • 3 minutes to read • Edit Online

For this tutorial, you will create a simple console application. This tutorial assumes you are developing in Microsoft Visual Studio 2010.

> **NOTE**
>
> When accessing the Report Server Web service running on SQL Server Express with Advanced Services, you must append "_SQLExpress" to the "ReportServer" path. For example:
>
> `http://myserver/reportserver_sqlexpress/reportservice2010.asmx"`

To create the web service proxy

1. From the **Start** menu, select **All Programs**, then Microsoft Visual Studio, then **Visual Studio Tools**, and then **Visual Studio 2010 Command Prompt**.

2. In the command prompt window, run the following command if you are using C#:

   ```
   wsdl /language:CS /n:"ReportService2010" http://<Server Name>/reportserver/reportservice2010.asmx?wsdl
   ```

   If you are using Visual Basic, then run the following command:

   ```
   wsdl /language:VB /n:"ReportService2010" http://<Server Name>/reportserver/reportservice2010.asmx?wsdl
   ```

   This command generates a .cs or .vb file. You will add this file to your application.

To create a console application

1. On the **File** menu, point to **New**, and then click **Project** to open the **New Project** dialog box.

2. In the left pane, under **Installed Templates**, click either **Visual Basic** or the **Visual C#** node, and select a category of project types from the expanded list.

3. Choose the **Console Application** project type.

4. In the **Name** box, enter a name for your project. Type the name `SampleRDLSchema`.

5. In the **Location** box, type the path where you want to save your project, or click **Browse** to navigate to the folder.

6. Click **OK**. A collapsed view of your project appears in Solution Explorer.

7. On the **Project** menu, click **Add Existing Item**.

8. Navigate to the location for the .cs or .vb file you generated, then select the file, and then click **Add**.

   You will also need to add a reference to the Services namespace for your Web reference to work.

9. On the Project menu, click **Add Reference**.

   In the **Add Reference** dialog box, in the **.NET** tab, select **System.Web.Services**, then click **OK**.

For more information about how to connect to the Report Server Web service, see Building Applications Using the Web Service and the .NET Framework.

10. In Solution Explorer, expand the project node. You will see a code file with the default name of Program.cs (Module1.vb for Visual Basic) has been added to your project.

11. Open the Program.cs (Module1.vb for Visual Basic) file and replace the code with the following:

    The following code provides the method stubs we will use to implement the Load, Update and Save functionality.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;
using ReportService2010;

namespace SampleRDLSchema
{
    class ReportUpdater
    {
        ReportingService2010 _reportService;

        static void Main(string[] args)
        {
            ReportUpdater reportUpdater = new ReportUpdater();
            reportUpdater.UpdateReport();
        }

        private void UpdateReport()
        {
            try
            {
                // Set up the Report Service connection
                _reportService = new ReportingService2010();
                _reportService.Credentials =
                    System.Net.CredentialCache.DefaultCredentials;
                _reportService.Url =
                   "http://<Server Name>/reportserver/" +
                                   "reportservice2010.asmx";

                // Call the methods to update a report definition
                LoadReportDefinition();
                UpdateReportDefinition();
                PublishReportDefinition();
            }
            catch (Exception ex)
            {
                System.Console.WriteLine(ex.Message);
            }
        }

        private void LoadReportDefinition()
        {
            //Lesson 3: Load a report definition from the report
            //          catalog
        }

        private void UpdateReportDefinition()
        {
            //Lesson 4: Update the report definition using the
            //          classes generated from the RDL Schema
        }

        private void PublishReportDefinition()
        {
            //Lesson 5: Publish the updated report definition back
            //          to the report catalog
        }
    }
}
```

```
Imports System
Imports System.Collections.Generic
Imports System.IO
Imports System.Text
Imports System.Xml
Imports System.Xml.Serialization
Imports ReportService2010

Namespace SampleRDLSchema

    Module ReportUpdater

        Private m_reportService As ReportingService2010

        Public Sub Main(ByVal args As String())

            Try
                'Set up the Report Service connection
                m_reportService = New ReportingService2010
                m_reportService.Credentials = _
                    System.Net.CredentialCache.DefaultCredentials
                m_reportService.Url = _
                    "http:// <Server Name>/reportserver/" & _
                           "reportservice2010.asmx"

                'Call the methods to update a report definition
                LoadReportDefinition()
                UpdateReportDefinition()
                PublishReportDefinition()
            Catch ex As Exception
                System.Console.WriteLine(ex.Message)
            End Try

        End Sub

        Private Sub LoadReportDefinition()
            'Lesson 3: Load a report definition from the report
            '          catalog
        End Sub

        Private Sub UpdateReportDefinition()
            'Lesson 4: Update the report definition using the
            '          classes generated from the RDL Schema
        End Sub

        Private Sub PublishReportDefinition()
            'Lesson 5: Publish the updated report definition back
            '          to the report catalog
        End Sub

    End Module

End Namespace
```

Next Lesson

In the next lesson, you will use the XML Schema Definition Tool (Xsd.exe) to generate classes from the RDL schema and include them in your project. See Lesson 2: Generate Classes from the RDL Schema using the xsd Tool.

See Also

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)

Lesson 2: Generate Classes from the RDL Schema using the xsd Tool

2/11/2019 • 2 minutes to read • Edit Online

After you have created your Visual Studio project, the next step is to retrieve a local copy of the report definition schema and run the XML Schema Definition Tool (Xsd.exe).

To generate the RDL classes

1. Open an instance of Microsoft Internet Explorer (or equivalent Web browser) and navigate to the following URL:

   ```
   https://schemas.microsoft.com/sqlserver/reporting/2010/01/reportdefinition/ReportDefinition.xsd
   ```

2. Once the RDL schema has been opened in the browser, browse to the **File** menu, and select **Save As**.

3. Browse to the location where you created your Visual Studio project and save the schema with the file name ReportDefinition.xsd.

4. After the file has been saved, open an instance of the Visual Studio 2010 command prompt. To open an instance of the command prompt, click the Start menu, point to **All Programs**, point to **Microsoft Visual Studio 2010**, point to **Visual Studio Tools** and click **Visual Studio Command Prompt (2010)**.

5. Change the current path to the location where you saved the ReportDefinition.xsd file:

   ```
   CD\<ReportDefinition.xsd Path>
   ```

6. Generate the ReportDefinition.cs file that contains the classes for the RDL schema with the following command:

   ```
   xsd /c /n:SampleRDLSchema ReportDefinition.xsd
   ```

   To generate a ReportDefinition.vb file use this command:

   ```
   xsd /c /l:VB /n:SampleRDLSchema ReportDefinition.xsd
   ```

7. Add ReportDefinition.xsd your project. From the **Project** menu, click **Add Existing Item**. Browse to the location of the ReportDefinition.xsd file, select ReportDefinition.xsd, and click **Add**.

   > **NOTE**
   >
   > After you have added the ReportDefinition.xsd file to the project you will notice in **Solution Explorer** that the ReportDefinition.cs (.vb) file is not there. To display the file, click the expand/collapse button next to the ReportDefinition.xsd file.

Next Lesson

In the next lesson, you will write code to load a report definition from a report server using the classes you generated from the RDL schema. See Lesson 3: Load a Report Definition from the Report Server.

See Also

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)
Report Definition Language (SSRS)

Lesson 3: Load a Report Definition from the Report Server

2/11/2019 • 2 minutes to read • Edit Online

After you have created your project and generated the classes from the RDL schema, you are ready to load a report definition from the report server.

To load a report definition

1. Add a private field at the top of the `ReportUpdater` class (module if you are using Visual Basic) for the `Report` class. This field will be used to maintain a reference to report that is loaded from the report server for the life of the application.

```
private Report _report;
```

```
Private m_report As Report
```

2. Replace the code for the `LoadReportDefinition()` method in the Program.cs file (Module1.vb for Visual Basic) with the following code:

```
private void LoadReportDefinition()
{
    System.Console.WriteLine("Loading Report Definition");

    string reportPath =
        "/AdventureWorks 2012 Sample Reports/Company Sales 2012";

    // Retrieve the report definition
    // from the report server
    byte[] bytes =
        _reportService.GetItemDefinition(reportPath);

    if (bytes != null)
    {
        XmlSerializer serializer =
            new XmlSerializer(typeof(Report));

        // Load the report bytes into a memory stream
        using (MemoryStream stream = new MemoryStream(bytes))
        {
            // Deserialize the report stream to an
            // instance of the Report class
            _report = (Report)serializer.Deserialize(stream);
        }
    }
}
```

```
Private Sub LoadReportDefinition()

    System.Console.WriteLine("Loading Report Definition")

    Dim reportPath As String = _
        "/AdventureWorks 2012 Sample Reports/Company Sales 2012"

    'Retrieve the report definition
    'from the report server
    Dim bytes As Byte() = _
        m_reportService.GetItemDefinition(reportPath)

    If Not (bytes Is Nothing) Then

        Dim serializer As XmlSerializer = _
            New XmlSerializer(GetType(Report))

        'Load the report bytes into a memory stream
        Using stream As MemoryStream = New MemoryStream(bytes)

            'Deserialize the report stream to an
            'instance of the Report class
            m_report = CType(serializer.Deserialize(stream), _
                             Report)

        End Using

    End If

End Sub
```

Next Lesson

In the next lesson, you will write code to update the report definition that was loaded from the report server. See Lesson 4: Update the Report Definition Programmatically.

See Also

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)
Report Definition Language (SSRS)

Lesson 4: Update the Report Definition Programmatically

2/11/2019 • 2 minutes to read • Edit Online

Now that the report definition has been loaded from the report server and you have a reference to it using the report field, you need to update the report definition. For this example, you will update the `Description` property for the report.

To update the report definition

1. Replace the code for the UpdateReportDefinition() method in the Program.cs file (Module1.vb for Visual Basic) with the following code:

```
private void UpdateReportDefinition()
{
    System.Console.WriteLine("Updating Report Definition");

    // Create a list of the Items Choices for the Report. The
    // ItemsChoiceType118 enum represents all the properties
    // available in the report and the ItemsElementName
    // represents the properties that exist in the current
    // instance of the report.
    List<ItemsChoiceType118> _reportItems =
        new List<ItemsChoiceType118>(_report.ItemsElementName);

    // Locate the index for the Description property
    int index = _reportItems.IndexOf(
        ItemsChoiceType118.Description);

    // The Description item is of type StringLocIDType, so
    // cast the item type first and then assign new value.
    System.Console.WriteLine("- Old Description: " +
        ((StringLocIDType)_report.Items[index]).Value );

    // Update the Description for the Report
    ((StringLocIDType)_report.Items[index]).Value =
        "New Report Description";

    System.Console.WriteLine("- New Description: " +
        ((StringLocIDType)_report.Items[index]).Value );
}
```

```
Private Sub UpdateReportDefinition()

    System.Console.WriteLine("Updating Report Definition")

    'Create a list of the Items Choices for the Report. The
    'ItemsChoiceType118 enum represents all the properties
    'available in the report and the ItemsElementName
    'represents the properties that exist in the current
    'instance of the report.
    Dim reportItems As List(Of ItemsChoiceType118) = _
        New List(Of ItemsChoiceType118)(m_report.ItemsElementName)

    'Locate the index for the Description property
    Dim index As Integer = _
        reportItems.IndexOf(ItemsChoiceType118.Description)

    'The Description item is of type StringLocIDType, so
    'cast the item type first and then assign new value.
    System.Console.WriteLine("- Old Description: " & _
        DirectCast(m_report.Items(index), StringLocIDType).Value)

    'Update the Description for the Report
    DirectCast(m_report.Items(index), StringLocIDType).Value = _
        "New Report Description"

    System.Console.WriteLine("- New Description: " & _
        DirectCast(m_report.Items(index), StringLocIDType).Value)

End Sub
```

Next Lesson

In the next lesson, you will save the updated report definition back to the report server. See Lesson 5: Publish the Report Definition to the Report Server.

See Also

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)

Lesson 5: Publish the Report Definition to the Report Server

2/11/2019 • 2 minutes to read • Edit Online

The last step for updating the report definition is to publish it back to the report server.

To publish the report to the report catalog

1. Replace the code for the `PublishReportDefinition()` method in your Program.cs file (Module1.vb for Visual Basic) with the following code:

```
private void PublishReportDefinition()
{
    System.Console.WriteLine("Publishing Report Definition");

    string reportPath =
        "/AdventureWorks 2012 Sample Reports/Company Sales 2012";

    XmlSerializer serializer =
        new XmlSerializer(typeof(Report));

    using (MemoryStream stream = new MemoryStream())
    {
        // Serialize the report into the MemoryStream
        serializer.Serialize(stream, _report);
        stream.Position = 0;

        byte[] bytes = stream.ToArray();

        // Update the report on the report server
        Warning[] warnings =
            _reportService.SetItemDefinition(reportPath, bytes, null);
    }
}
```

```
Private Sub PublishReportDefinition()

    System.Console.WriteLine("Publishing Report Definition")

    Dim reportPath As String = _
        "/AdventureWorks 2012 Sample Reports/Company Sales 2012"
    Dim serializer As XmlSerializer = _
        New XmlSerializer(GetType(Report))

    Using stream As MemoryStream = New MemoryStream

        'Serialize the report into the MemoryStream
        serializer.Serialize(stream, m_report)
        stream.Position = 0

        'Update the report on the report server
        Dim bytes As Byte() = stream.ToArray
        Dim warnings As Warning() = _
            m_reportService.SetItemDefinition(reportPath, bytes, Nothing)

    End Using

End Sub
```

Next Lesson

In the next lesson, you will compile and run the `SampleRDLSchema` application. See Lesson 6: Run the RDL Schema Application (VB-C#).

See Also

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)
Report Definition Language (SSRS)

Lesson 6: Run the RDL Schema Application (VB-C#)

2/11/2019 • 2 minutes to read • Edit Online

Visual Studio offers two methods to build and run a console application from the integrated development environment (IDE):

- Start (with Debugging)
- Start without Debugging

To build and run the SampleRDLSchema application

1. From the **Debug** menu, click **Start Without Debugging**. This ensures that the console window remains open after the program has finished executing.

   The application prints the following output to the console:

   ```
   Loading Report Definition
   Updating Report Definition
   - Old Description: <Old Report Description>
   - New Description: <New Report Description>
   Publishing Report Definition
   ```

2. Press any key to close the console.

   > **NOTE**
   >
   > Any errors that occur are written to the console.

3. When the sample application is finished running an updated copy of the report will be saved to the report server.

See Also

Updating Reports Using Classes Generated from the RDL Schema (SSRS Tutorial)
Report Definition Language (SSRS)

Tutorials (Report Builder)

2/14/2019 • 2 minutes to read • Edit Online

Use the following Report Builder tutorials to learn how to create basic reports.

Prerequisites for Tutorials (Report Builder)

To use these tutorials, you must have read-only access to a SQL Server 2014 database and permissions to access a SQL Server 2014 Reporting Services (SSRS) report server.

For help with these prerequisites, contact your report server administrator.

Alternative Ways to Get a Data Connection (Report Builder)

Most of these tutorials use embedded data sources. If you have access to shared data sources you can use them instead. This topic provides the steps for using shared data sources.

Tutorial: Creating a Basic Table Report (Report Builder)

Follow the steps in this tutorial to learn how to create your first report. This tutorial shows you how to work with Report Builder to create a data source connection, define a simple query, build a layout to contain your data, format your report, and add grouping and totals.

Tutorial: Creating a Matrix Report (Report Builder)

Follow the steps in this tutorial to learn how to add and configure a matrix. You will use the Table or Matrix Wizard to create the report data source, dataset, and layout, and then enhance the matrix within the **Design** view of Report Builder.

Tutorial: Creating a Free Form Report (Report Builder)

Follow the steps in this tutorial to learn how to create a free-form report from scratch. This report resembles a newsletter.

Tutorial: Format Text (Report Builder)

Follow the steps in this tutorial to learn some of the many ways you can format text in your reports.

Tutorial: Add a Column Chart to Your Report (Report Builder)

Follow the steps in this tutorial to learn how to add a column chart with a moving average to a report.

Tutorial: Add a Pie Chart to Your Report (Report Builder)

Follow the steps in this tutorial to learn how to add a pie chart to a report.

Tutorial: Add a Bar Chart to Your Report (Report Builder)

Follow the steps in this tutorial to learn how to add a bar chart to a report.

Tutorial: Add a Sparkline to Your Report (Report Builder)

Follow the steps in this tutorial to learn how to create a matrix and then add a set of sparkline charts to the matrix.

Tutorial: Adding a KPI to Your Report (Report Builder)

Follow the steps in this tutorial to learn how to add a key performance indicator (KPI) to a report.

Tutorial: Map Report (Report Builder)

Follow the steps in this tutorial to learn how to add a map to a report.

Tutorial: Add a Parameter to Your Report (Report Builder)

Follow the steps in this tutorial to learn how to customize the appearance and content of a report by using parameters.

Tutorial: Creating Drillthrough and Main Reports (Report Builder)
Follow the steps in this tutorial to learn how to create reports for a drillthrough scenario based on an Analysis Services cube. You will create the main report and enable it for drillthrough and the report that is the target of the drillthrough action.

Tutorial: Introducing Expressions
Follow the steps in this tutorial to learn how to use expressions to concatenate, calculate, and lookup field values and show them a report. You will also learn how to conditionally display different images and indicator states using expressions.

See Also

Report Design View (Report Builder)
Report Builder in SQL Server 2014
Tutorial: Create a Quick Chart Report Offline (Report Builder)

Prerequisites for Tutorials (Report Builder)

2/14/2019 • 2 minutes to read • Edit Online

The Report Builder tutorials expect that you can view and save reports on a report server or SharePoint site that is integrated with a report server. For data, all tutorials use literal queries that must be processed by an instance of SQL Server 2014.

If you do not have access to a report server or site or to a data source, you can learn about Report Builder by building an offline report. See Tutorial: Create a Quick Chart Report Offline (Report Builder).

Requirements

You must have the following prerequisites to complete Report Builder tutorials:

- Access to Microsoft SQL Server 2014 Report Builder. You can run Report Builder by using the stand-alone version of Report Builder or the ClickOnce version, available from Report Manager or a SharePoint site. Only the first step, how to open Report Builder, is different for ClickOnce versions.

  To use Report Manager, open Report Manager and click **Report Builder**. By default, the URL for Report Manager is http://<*servername*>/reports.

  To use a SharePoint site, navigate to the site, click the Documents tab, click New Document, and from the drop-down list, click Report Builder Report. For example, http://<servername>/sites/mySite/reports. The SharePoint administrator must enable the Report Builder Report feature for each document library.

- The URL to a Microsoft SQL Server 2014 Reporting Services (SSRS) report server or a SharePoint site that is integrated with a SQL Server 2014 Reporting Services (SSRS) report server. You must have permission to save and view reports, shared data sources, shared datasets, report parts, and models. By default, the URL for a report server is http://<servername>/reportserver. By default, the URL for a SharePoint site is http://<sitename> or http://<server>/site.

- The name of a Microsoft SQL Server 2014 instance and credentials sufficient for read-only access to any database. The dataset queries in the tutorials use literal data, but each query must be processed by a SQL Server 2014 instance to return the metadata that is required for a report dataset. For example, the following connection string specifies only a server: `data source=<servername>`. You must have read access to the default database that is assigned to you by the system administrator who grants you permission to access the server. You can also specify a database, as shown in the following connection string: `data source=<servername>;initial catalog=<database>`.

- For the tutorial that includes a map, the report server must be configured to support the Bing maps as a background. For more information, see Plan for Map Report Support in Reporting Services documentation in SQL Server Books Online on msdn.microsoft.com.

- The tutorial, Tutorial: Creating Drillthrough and Main Reports (Report Builder), uses the Contoso business intelligence demo dataset. This dataset is comprised of the ContosoDW data warehouse and the Contoso_Retail online analytical processing (OLAP) database. The reports you will create in this tutorial retrieve report data from the Contoso Sales cube. The Contoso_Retail OLAP database can be downloaded from Microsoft Download Center. You need only download the file ContosoBIdemoABF.exe. It contains the OLAP database.

  The other file, ContosoBIdemoBAK.exe, is for the ContosoDW data warehouse, which is not used in this tutorial.

The Web site includes instructions extracting and restoring the ContosoRetail.abf backup file to the Contoso_Retail OLAP database.

You must have access to an instance of Analysis Services on which to install the OLAP database.

The report server administrator must grant you the necessary permissions on the report server, configure SQL Server 2014 Reporting Services (SSRS) folder locations, and configure Report Builder default options. For more information, see Install, Uninstall, and Report Builder Support.

See Also

Tutorials (Report Builder)

Alternative Ways to Get a Data Connection (Report Builder)

2/11/2019 • 2 minutes to read • Edit Online

A data connection contains the information to connect to an external data source such as a SQL Server database. Usually, you get the connection information and the type of credentials to use from the data source owner.

To specify a data connection, you can use a shared data source from the report server or create an embedded data source that is used only in a specific report.

In most tutorials you use embedded data sources, but if you have access to shared data sources, then you can use them instead.

Getting a Data Connection From a Shared Data Source

If the report server has available shared data source that you have permissions to use, you can use them instead of an embedded data source. The following procedures tell how to locate the shared data sources and provide any credentials needed to use them.

To use a shared data source, you must browse to a report server and select one. Usually, you get the report server URL from the report server administrator.

To specify a data connection from a list of shared data sources

1. On the **Choose a dataset** page, select **Create a dataset**, and then click **Next**. The **Choose a connection to a data source** page opens.
2. From the list of data sources, select a data source that you have permission to access.
3. To verify that you can connect to the data source, click **Test Connection**. The message "Connection created successfully" appears. Click **OK**.
4. Click **Next**.

   If necessary, enter your credentials. To save the credentials locally, select **Save password with connection**. If you not select this option, you will be prompted for credentials every time that you run the report
5. Click **OK**.

To specify a data connection by browsing to a shared data source on a report server

1. On the **Choose a dataset** page, select **Create a dataset**, and then click **Next**. The **Choose a connection to a data source** page opens.
2. Click **Browse**. The **Select Data Source** dialog box opens.
3. From the **Look in** drop-down list, select **Recent Sites and Servers**. In the data source pane, click the URL for your server, and then click **Open**.

   The list of data sources or models appears.
4. Alternatively, in **Name**, type the URL to the report server. Click **Open**.

   Report Builder connects to the report server and loads the data sources that are available at the root folder.
5. Navigate to a folder that contains a data source that you have sufficient permissions to connect to, select

the data source, and then click **Open**.

You are back on the **Choose a connection to a data source** page.

6. To verify that you can connect to the data source, click **Test Connection**.

   The message "Connection created successfully" appears. Click **OK**.

7. Click **Next**.

8. If you are prompted for a user name and password, enter your credentials. To save the credentials locally, select **Save password with connection**.

9. Click **OK**.

See Also

Add Data to a Report (Report Builder and SSRS)
Tutorials (Report Builder)

Tutorial: Creating a Basic Table Report (Report Builder)

2/14/2019 • 13 minutes to read • Edit Online

This tutorial teaches you to create a basic table report based on sample sales data. The following illustration shows the report you will create.

Product Sales

Sales Date	Subcategory	Product	Quantity	Sales
January 05, 2009	Accessories	Carrying Case	68	$9,224.60
		Mini Battery Charger	44	$1,056.00
			112	$10,980.60
	Digital	Slim Digital	44	$8,357.80
			44	$8,357.80
			156	$19,338.40
January 11, 2009	Accessories	Lens Adapter	17	$1,147.50
			17	$1,147.50
	Digital	Advanced Digital	39	$7,234.50
			39	$7,234.50
	Total		56	$8,382.00
Total			576	$106,920.40

What You Will Learn

In this tutorial, you will learn how to do the following:

1. Create a New Report from Getting Started
    a. Specify a Data Connection in the Table Wizard
    b. Create a Query in the Table Wizard
    c. Organize Data into Groups in the Table Wizard
    d. Add Subtotal and Total Rows in the Table Wizard
    e. Choose a Style in the Table Wizard
2. Format Data as Currency
3. Format Data as Date
4. Change Column Widths
5. Add a Report Title
6. Save the Report
7. Export the Report

Estimated time to complete this tutorial: 20 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a New Report from Getting Started

Create a table report from the **Getting Started** dialog box. There are two modes: report design and shared dataset design. In report design mode, you specify data in the Report Data pane and the report layout on the design surface. In shared dataset design mode, you create dataset queries to share with others. In this tutorial, you will be using report design mode.

To create a new report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box opens.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, from the **Report Builder** button, click **New**.

2. In the left pane, verify that **New Report** is selected.

3. In the right pane, verify that **Table or Matrix Wizard** is selected.

1a. Specify a Data Connection in the Table Wizard

A data connection contains the information to connect to an external data source such as a SQL Server database. Usually, you get the connection information and the type of credentials to use from the data source owner. To specify a data connection, you can use a shared data source from the report server or create an embedded data source that is used only in this report.

In this tutorial, you will use an embedded data source. To learn more about using a shared data sources, see Alternative Ways to Get a Data Connection (Report Builder).

To create an embedded data source

1. On the **Choose a dataset** page, select **Create a dataset**, and then click **Next**. The **Choose a connection to a data source** page opens.

2. Click **New**. The **Data Source Properties** dialog box opens.

3. In **Name**, type **Product Sales** a name for the data source.

4. In **Select a connection type**, verify that **Microsoft SQL Server** is selected.

5. In **Connection string**, type the following text, where <*servername*> is the name of an instance of SQL Server:

   ```
   Data Source=<servername>
   ```

   Because you will use a query that contains the data instead of retrieving the data from a database, the connection string does not include the database name. For more information, see Prerequisites for Tutorials (Report Builder).

6. Click **Credentials**. Enter the credentials that you need to access the external data source.

7. Click **OK**.

   You are back on the **Choose a connection to a data source** page.

8. To verify that you can connect to the data source, click **Test Connection**.

   The message "Connection created successfully" appears.

9. Click **OK**.

10. Click **Next**.

1b. Create a Query in the Table Wizard

In a report, you can use a shared dataset that has a predefined query, or you can create an embedded dataset for use only in your report. In this tutorial, you will create an embedded dataset.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a query

1. On the **Design a query** page, the relational query designer is open. For this tutorial, you will use the text-based query designer.

   Click **Edit As Text**. The text-based query designer displays a query pane and a results pane.

2. Paste the following Transact-SQL query into the **Query** box.

```
SELECT CAST('2009-01-05' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Carrying Case' as Product, CAST(9924.60 AS money) AS Sales, 68 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Tripod' as Product, CAST(1350.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-11' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Lens Adapter' as Product, CAST(1147.50 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Mini Battery Charger' as Product, CAST(1056.00 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Accessories' as Subcategory,
   'Telephoto Conversion Lens' as Product, CAST(1380.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,'Accessories' as Subcategory,
   'USB Cable' as Product, CAST(780.00 AS money) AS Sales, 26 as Quantity
UNION SELECT CAST('2009-01-08' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(3798.00 AS money) AS Sales, 9 as Quantity
UNION SELECT CAST('2009-01-09' AS date) as SalesDate, 'Camcorders' as Subcategory,
   'Business Videographer' as Product, CAST(10400.00 AS money) AS Sales, 13 as Quantity
UNION SELECT CAST('2009-01-10' AS date) as SalesDate, 'Camcorders' as Subcategory,
   'Social Videographer' as Product, CAST(3000.00 AS money) AS Sales, 60 as Quantity
UNION SELECT CAST('2009-01-11' AS date) as SalesDate,  'Digital' as Subcategory,
   'Advanced Digital' as Product, CAST(7234.50 AS money) AS Sales, 39 as Quantity
UNION SELECT CAST('2009-01-07' AS date) as SalesDate,  'Digital' as Subcategory,
   'Compact Digital' as Product, CAST(10836.00 AS money) AS Sales, 84 as Quantity
UNION SELECT CAST('2009-01-08' AS date) as SalesDate,  'Digital' as Subcategory,
   'Consumer Digital' as Product, CAST(2550.00 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Digital' as Subcategory,
   'Slim Digital' as Product, CAST(8357.80 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2009-01-09' AS date) as SalesDate, 'Digital SLR' as Subcategory,
   'SLR Camera 35mm' as Product, CAST(18530.00 AS money) AS Sales, 34 as Quantity
UNION SELECT CAST('2009-01-07' AS date) as SalesDate, 'Digital SLR' as Subcategory,
   'SLR Camera' as Product, CAST(26576.00 AS money) AS Sales, 88 as Quantity
```

3. On the query designer toolbar, click **Run** (**!**).

   The query runs and displays the result set for the fields SalesDate, Subcategory, Product, Sales, and Quantity.

In the result set, the column headings are based on the names in the query. In the dataset, the column headings become the field names, and are saved in the report. After you complete the wizard, you can use the Report Data pane to view the collection of dataset fields.

4. Click **Next**.

1c. Organize Data into Groups in the Table Wizard

When you select fields to group on, you design a table that has rows and columns that display detail data and aggregated data.

To organize data into groups

1. On the **Arrange fields** page, drag Product to **Values**.
2. Drag Quantity to **Values** and place below Product.

   Quantity is automatically aggregated by the Sum function, the default aggregate for numeric fields. The value is [Sum(Quantity)].

   You can open the drop-down list to view the other aggregate functions available. Do not change the aggregate function.

3. Drag Sales to **Values** and place below [Sum(Quantity)].

   Sales is aggregated by the Sum function. The value is [Sum(Sales)].

   Steps 1, 2, and 3 specify the data to display in the table.

4. Drag SalesDate to **Row groups**.
5. Drag Subcategory to **Row groups** and place below SalesDate.

   Steps 4 and 5 organize the values for the fields first by date, and then by product subcategory for that date.

6. Click **Next**.

1d. Add Subtotal and Total Rows in the Table Wizard

After you create groups, you can add and format rows on which to display aggregate values for the fields. You can choose whether to show all the data or to let a user expand and collapse grouped data interactively.

To add subtotals and totals

1. On the **Choose the layout** page, under **Options**, verify that **Show subtotals and grand totals** is selected.
2. Verify that **Blocked, subtotal below** is selected.

   The wizard Preview pane displays a table with five rows. When you run the report, each row will display in the following way:

   a. The first row will repeat once for the table to show column headings.

   b. The second row will repeat once for each line item in the sales order and display the product name, order quantity, and line total.

   c. The third row will repeat once for each sales order to display subtotals per order.

   d. The fourth row will repeat once for each order date to display the subtotals per day.

   e. The fifth row will repeat once for the table to display the grand totals.

3. Clear the option **Expand/collapse groups**. In this tutorial, the report you create does not use the

drilldown feature that lets a user expand a parent group hierarchy to display child group rows and detail rows.

4. Click **Next**.

1e. Choose a Style in the Table Wizard

A style specifies a font style, a set of colors, and a border style.

To specify a table style

1. On the **Choose a Style** page, in the Styles pane, select Ocean.

   The Preview pane displays a sample of the table with that style.

2. Optionally, click the other styles to see the sample with them applied.

3. Click **Finish**.

The table is added to the design surface. The table has 5 columns and 5 rows. The Row Groups pane shows three row groups: SalesDate, Subcategory, and Details. Detail data is all the data that is retrieved by the dataset query.

2. Format Data as Currency

By default, the summary data for the Sales field displays a general number. Format it to display the number as currency. Toggle **Placeholder Styles** to display formatted text boxes and placeholder text as sample values.

To format a currency field

1. Click **Design** to switch to design view.

2. Click the cell in the second row (under the column headings row) in the Sales column and drag down to select all cells that contain `[Sum(Sales)]`.

3. On the **Home** tab, in the **Number** group, click the **Currency** button. The cells change to show the formatted currency.

   If your regional setting is English (United States), the default sample text is [**$12,345.00**]. If you do not see an example currency value, click **Placeholder Styles** in the **Numbers** group, and then click **Sample Values**.

4. Click **Run** to preview your report.

The summary values for Sales display as currency.

3. Format Data as Date

By default, the SalesDate field displays both date and time information. You can format them to display only the date.

To format a date field as the default format

1. Click **Design** to return to design view.

2. Click the cell that contains `[SalesDate]`.

3. On the Ribbon, on the **Home** tab, in the **Number** group, from the drop-down list, select **Date**.

   The cell displays the example date **[1/31/2000]**. If you do not see an example date, click **Placeholder Styles** in the **Numbers** group, and then click **Sample Values**.

4. Click **Run** to preview the report.

The SalesDate values display in the default date format.

To change the date format to a custom format

1. Click **Design** to return to design view.
2. Click the cell that contains `[SalesDate]`.
3. On the **Home** tab, in the **Number** group, click the dialog box launcher.

   The launcher is the small arrow in the right-hand corner of the group. The **Text Box Properties** dialog box opens.
4. In the Category pane, verify that **Date** is selected.
5. In the **Type** pane, select **January 31, 2000**.
6. Click **OK**.

   The cell displays the example date **[January 31, 2000]**.
7. Click **Run** to preview your report.

The SalesDate value displays with the name of the month instead of the number for the month.

4. Change Column Widths

By default, each cell in a table contains a text box. A text box expands vertically to accommodate text when the page is rendered. In the rendered report, each row expands to the height of the tallest rendered text box in the row. The height of the row on the design surface has no affect on the height of the row in the rendered report.

To reduce the amount of vertical space each row takes, expand the column width to accommodate the expected contents of the text boxes in the column on one line.

To change the width of table columns

1. Click **Design** to return to design view.
2. Click the table so that column and row handles appear above and next to the table.

   The gray bars along the top and side of the table are the column and row handles.
3. Point to the line between column handles so that the cursor changes into a double arrow. Drag the columns to the size you want. For example, expand the column for Product so that the product name displays on one line.
4. Click **Run** to preview your report.

5. Add a Report Title

A report title appears at the top of the report. You can place the report title in a report header or if the report does not use one, in a text box at the top of the report body. In this tutorial, you will use the text box that is automatically placed at the top of the report body.

The text can be further enhanced by applying different font styles, sizes, and colors to phrases and individual characters of the text. For more information, see Format Text in a Text Box (Report Builder and SSRS).

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Product Sales**, and then click outside the text box.
3. Right-click the text box that contains **Product Sales** and click **Text Box Properties**.
4. In the **Text Box Properties** dialog box, click **Font**.

5. In the **Size** list, select **18pt**.
6. In the **Color** list, select **Cornflower Blue**.
7. Select **Bold**.
8. Click **OK**.

6. Save the Report

Save the report to a report server or your computer. If you do not save the report to the report server, a number of Reporting Services features such as report parts and subreports are not available.

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.
2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you see the contents of the report folder that the report server administrator specified as the default location for reports.
4. In **Name**, replace the default name with **Product Sales**.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report on your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop**, **My Documents**, or **My computer**, and browse to the folder where you want to save the report.
3. In **Name**, replace the default name with **Product Sales**.
4. Click **Save**.

7. Export the Report

Reports can be exported to different formats such Microsoft Excel and comma separated value (CSV). For more information, see Exporting Reports (Report Builder and SSRS).

In this tutorial, you will export the report to Excel and set a property on the report to provide a custom name for the workbook tab.

To specify the workbook tab name

1. Click **Design** to return to design view.
2. Click anywhere outside the report.
3. .In the Properties pane, locate the InitialPageName property and type **Product Sales Excel**.

> **NOTE**
>
> If the Properties pane is not visible, click the View tab on the ribbon, and then click **Properties**.

To export a report to Excel

1. Click **Run** to preview the report.
2. .On the ribbon, click **Export**, and then click **Excel**.

   The **Save As** dialog box opens.
3. Browse to the **Documents** folder.
4. In the **File name** text box, type **Product Sales Excel**.
5. Verify that the file type is **Excel Workbook**.
6. Click **Save**.

To view the report in Excel

1. Open the **Documents** folder and double click **Product Sales Excel.xlsx**.
2. Verify that the name of the workbook tab is **Product Sales Excel**.

Next Steps

This concludes the walkthrough for how to create a basic table report. For more information about tables, see Tables, Matrices, and Lists (Report Builder and SSRS).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Creating a Matrix Report (Report Builder)

2/14/2019 • 15 minutes to read • Edit Online

This tutorial teaches you how to create a basic matrix report based on sample sales data. The matrix has nested row and column groups and an adjacent column group. You will also learn how to format columns and rotate text. The following illustration shows a report similar to the one you will create.

Sales by Territory, Subcategory, and Day

		⊟ Accessories						⊞ Digital		Total	
		Carrying Case		Lens Adapter		Tripod					
	Sales Date	Sales	QTY	Sales	QTY	Sales	QTY	Sales	QTY	Sales	QTY
⊟ Central	1/5/2009	$16,996.60	68	$1,147.50	17	$1,350.00	18	$17,409.10	117	$36,903.20	220
	1/6/2009	$13,497.30	54	$887.50	13	$1,200.00	16	$13,747.40	82	$29,332.20	165
	Total	$30,493.90	122	$2,035.00	30	$2,550.00	34	$31,156.50	199	$66,235.40	385
⊞ North	Total	$25,744.85	103	$1,350.00	20	$3,825.00	51	$35,471.50	223	$66,391.35	397
⊞ South	Total	$19,496.10	78	$2,632.50	39	$2,550.00	34	$35,176.80	233	$59,855.40	384
Total		$75,734.85	303	$6,017.50	89	$8,925.00	119	$101,804.80	655	$192,482.15	1166

2/23/2010 2:58:45 PM 1/1

An enhanced version of the report you will create in this tutorial is available as a sample SQL Server 2014 Report Builder report. For more information about downloading this sample report and others, see Report Builder sample reports.

What You Will Learn

In this tutorial, you will learn how to:

1. Create a Matrix Report and Dataset from the New Table or Matrix Wizard
2. Organize Data and Choose Layout and Style from the New Table or Matrix Wizard
3. Format Data
4. Add Adjacent Column Group
5. Change Column Widths
6. Merge Matrix Cells
7. Add a Report Header and Report Title
8. Save the Report

Other Optional Step

1. Rotate Text Box 270 Degrees

Estimated time to complete this tutorial: 20 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Matrix Report and Dataset from the New Table or Matrix Wizard

From the **Getting Started** dialog box in Report Builder, choose a shared data source, create an embedded dataset, and then display the data in a matrix.

> **NOTE**
> In this tutorial, the query already contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a new matrix

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   > **NOTE**
   > The **Getting Started** dialog box should appear. If it doesn't, from the Report Builder button, click **New**.

2. In the left pane, verify that **New Report** is selected.
3. In the right pane, click **Table or Matrix Wizard**.
4. On the **Choose a dataset** page, click **Create a dataset**.
5. Click **Next**.
6. On the **Choose a connection to a data source** page, select an existing data source or browse to the report server, and then select a data source. If no data source is available or you do not have access to a report server, you can use an embedded data source instead. For more information about creating an embedded data source, see Tutorial: Creating a Basic Table Report (Report Builder).
7. Click **Next**.
8. On the **Design a query** page, click **Edit as Text**.
9. Copy and paste the following query into the query pane:

```
SELECT CAST('2009-01-05' AS date) as SalesDate, 'Central' as Territory, 'Accessories' as
Subcategory,'Carrying Case' as Product, CAST(16996.60 AS money) AS Sales, 68 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'North' as Territory, 'Accessories' as
Subcategory, 'Carrying Case' as Product, CAST(13747.25 AS money) AS Sales, 55 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'South' as Territory, 'Accessories' as
Subcategory,'Carrying Case' as Product, CAST(9248.15 AS money) As Sales, 37 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Central' as Territory, 'Accessories' as
Subcategory,'Tripod' as Product, CAST(1350.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'North' as Territory, 'Accessories' as
Subcategory,'Tripod' as Product, CAST(1800.00 AS money) AS Sales, 24 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'South' as Territory, 'Accessories' as
Subcategory,'Tripod' as Product, CAST(1125.00 AS money) AS Sales, 15 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Central' as Territory, 'Accessories' as
Subcategory,'Lens Adapter' as Product, CAST(1147.50 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'North' as Territory, 'Accessories' as
Subcategory,  'Lens Adapter' as Product, CAST(742.50 AS money) AS Sales, 11 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'South' as Territory, 'Accessories' as
Subcategory,'Lens Adapter' as Product, CAST(1417.50 AS money) AS Sales, 21 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Central' as Territory, 'Accessories' as
Subcategory, 'Carrying Case' as Product, CAST(13497.30 AS money) AS Sales, 54 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'North' as Territory, 'Accessories' as
Subcategory, 'Carrying Case' as Product, CAST(11997.60 AS money) AS Sales, 48 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'South' as Territory, 'Accessories' as
Subcategory, 'Carrying Case' as Product, CAST(10247.95 AS money) As Sales, 41 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Central' as Territory, 'Accessories' as
Subcategory, 'Tripod' as Product, CAST(1200.00 AS money) AS Sales, 16 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'North' as Territory, 'Accessories' as
Subcategory,'Tripod' as Product, CAST(2025.00 AS money) AS Sales, 27 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'South' as Territory, 'Accessories' as
Subcategory,'Tripod' as Product, CAST(1425.00 AS money) AS Sales, 19 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Central' as Territory, 'Accessories' as
Subcategory,'Lens Adapter' as Product, CAST(887.50 AS money) AS Sales, 13 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'North' as Territory, 'Accessories' as
Subcategory, 'Lens Adapter' as Product, CAST(607.50 AS money) AS Sales, 9 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'South' as Territory, 'Accessories' as
Subcategory,'Lens Adapter' as Product, CAST(1215.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Central' as Territory, 'Digital' as
Subcategory,'Compact Digital' as Product, CAST(10191.00 AS money) AS Sales, 79 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'North' as Territory, 'Digital' as Subcategory,
'Compact Digital' as Product, CAST(8772.00 AS money) AS Sales, 68 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'South' as Territory, 'Digital' as Subcategory,
'Compact Digital' as Product, CAST(10578.00 AS money) AS Sales, 82 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Central' as Territory,'Digital' as Subcategory,
'Slim Digital' as Product, CAST(7218.10 AS money) AS Sales, 38 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'North' as Territory,'Digital' as Subcategory,
'Slim Digital' as Product, CAST(8357.80 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'South' as Territory,'Digital' as
Subcategory,'Slim Digital' as Product, CAST(9307.55 AS money) AS Sales, 49 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Central' as Territory, 'Digital' as
Subcategory,'Compact Digital' as Product, CAST(3870.00 AS money) AS Sales, 30 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'North' as Territory, 'Digital' as
Subcategory,'Compact Digital' as Product, CAST(5805.00 AS money) AS Sales, 45 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'South' as Territory, 'Digital' as Subcategory,
'Compact Digital' as Product, CAST(8643.00 AS money) AS Sales, 67 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Central' as Territory, 'Digital' as Subcategory,
'Slim Digital' as Product, CAST(9877.40 AS money) AS Sales, 52 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'North' as Territory, 'Digital' as Subcategory,
'Slim Digital' as Product, CAST(12536.70 AS money) AS Sales, 66 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'South' as Territory, 'Digital' as Subcategory,
'Slim Digital' as Product, CAST(6648.25 AS money) AS Sales, 35 as Quantity
```

10. Click **Next**.

2. Organize Data and Choose Layout and Style from the New Table or

Matrix Wizard

Use the wizard to provide a starting design on which to display data. The preview pane in the wizard helps you to visualize the result of grouping data before you complete the matrix design.

To organize data into groups and choose a layout and style

1. On the **Arrange fields** page, drag Territory from **Available fields** to **Row groups**.
2. Drag SalesDate to **Row groups** and place it below Territory.

   The order in which fields are listed in **Row groups** defines the group hierarchy. Steps 1 and 2 organize the values of the fields first by territory, and then by sales date.
3. Drag Subcategory to **Column groups**.
4. Drag Product to **Column groups,** and then place below Subcategory.

   The order in which fields are listed in **Column groups** defines the group hierarchy.

   Steps 3 and 4 organize the values for the fields first by subcategory, and then by product.
5. Drag Sales to **Values**.

   Sales is summarized with the Sum function, the default function to summarize numeric fields.
6. Drag Quantity to **Values**.

   Quantity is summarized with the Sum function.

   Steps 5 and 6 specify the data to display in the matrix data cells.
7. Click **Next**.
8. On the Choose the Layout page, under **Options**, verify that **Show subtotals and grand totals** is selected.
9. Verify that **Blocked, subtotal below** is selected.
10. Verify the option **Expand/collapse groups** is selected.
11. Click **Next**.
12. On the Choose a Style page, in the Styles pane, select **Slate**.
13. Click **Finish**.

    The matrix is added to the design surface. The Row Groups pane shows two row groups: Territory and SalesDate. The Column Groups pane shows two column groups: Subcategory and Product. Detail data is all the data that is retrieved by the dataset query.
14. Click **Run** to preview the report.

For each product that is sold on a specific date, the matrix shows the subcategory to which the product belongs and the territory of the sales.

3. Format Data

By default, the summary data for the Sales field displays a general number and the SalesDate field displays both date and time information. Format the Sales field to display the number as currency and the SalesDate field to display only the date. Toggle **Placeholder Styles** to display formatted text boxes and placeholder text as sample values.

To format fields

1. Click **Design** to switch to design view.

2. Press the Ctrl key, and then select the nine cells that contain `[Sum(Sales)]`.

3. On the **Home** tab, in the **Number** group, click **Currency**. The cells changeto show the formatted currency.

   If your regional setting is English (United States), the default sample text is [**$12,345.00**]. If you do not see an example currency value, click **Placeholder Styles** in the **Numbers** group, and then click **Sample Values**.

4. Click the cell that contains `[SalesDate]`.

5. In the **Number** group, from the drop-down list, select **Date**.

   The cell displays the example date **[1/31/2000]**. If you do not see an example date, click **Placeholder Styles** in the **Numbers** group, and then click **Sample Values**.

6. Click **Run** to preview your report.

The date values display only dates and the sales values display as currency.

4. Add Adjacent Column Group

You can nest row and column groups in parent child relationships or adjacent in sibling relationships.

Add a column group that is adjacent to the Subcategory column group, copy cells to populate the new column group, and then use an expression to create the value of the column group header.

To add an adjacent column group

1. Click **Design** to return to design view.

2. Right-click the cell that contains `[Subcategory]`, point to **Add Group**, and then click **Adjacent Right**.

   The **Tablix Group** dialog box opens.

3. In the **Group By** list, select SalesDate, and then click **OK**.

   A new column group is added to the left of the Subcategory column group.

4. Right-click the cell in the new column group that contains `[SalesDate],` and then click **Expression**.

5. Copy the following expression to expression box.

   ```
   =WeekdayName(DatePart("w",Fields!SalesDate.Value))
   ```

   This expression extracts the weekday name from the sales date. For more information, see Expressions (Report Builder and SSRS).

6. Right-click the cell in the Subcategory column group that contains Total, and then click **Copy**.

7. Right-click the cell immediately below the cell that contains the expression you created in step 5 and click **Paste**.

8. Press the Ctrl key.

9. In the Subcategory group, click the Sales column header and the three cells below it, right-click, and then click **Copy**.

10. Paste the four cells into the four empty cells in the new column group.

11. Click **Run** to preview the report.

The report includes columns named Monday and Tuesday. The dataset contains only data for these two days.

> **NOTE**
> If the data included other days, the report would include columns for them as well. Each column has the column header, `Sales`, and sales totals by territory.

5. Change Column Widths

A report that includes a matrix typically expands horizontally as well as vertically when it runs. Controlling horizontal expansion is particularly important if you plan to export the report to formats such as Microsoft Word or Adobe PDF that are used for printed reports. If the report expands horizontally across multiple pages, the printed report is difficult to understand. To minimize horizontal expansion, you can resize columns to be only the width necessary to display the data without wrapping. You can also rename columns so that their titles fit the width needed to display the data.

To rename and resize the columns

1. Click **Design** to return to design view.
2. Select the text in the furthest Quantity column to the left, and then type **QTY**.

   The column title is now QTY.
3. Repeat step 2 for the other columns named Quantity. There are two of them.
4. Click the matrix so that column and row handles appear above and next to the matrix.

   The gray bars along the top and side of the table are the column and row handles.
5. To resize the furthest QTY column to the left, point to the line between column handles so that the cursor changes into a double arrow. Drag the column towards the left until it is 1/2 inch wide.

   A column width of 1/2 inch is adequate to display the quantity.
6. Repeat step 5 for the other columns named QTY.
7. Click **Run** to preview your report.

The columns in the report that contains quantities are now named QTY and the columns are narrower.

6. Merge Matrix Cells

The corner area is in the upper left corner of the matrix. Depending on the number of row and column groups in the matrix, the number of cells in the corner area varies. The matrix, built in this tutorial, has four cells in its corner area. The cells are arranged in two rows and two columns, reflecting the depth of row and column group hierarchies. The four cells are not used in this report and you will merge them into one.

To merge matrix cells

1. Click **Design** to return to design view.
2. Click the matrix so that column and row handles appear above and next to the matrix.
3. Press the Ctrl key, and then select the four corner cells.
4. Right-click the cells and then click **Merge Cells**.
5. Right-click the corner cell, and then click **Text Box Properties**.
6. Click the **Fill** tab.
7. Click the (***fx***) button for **Fill color**.

8. Copy and paste the following expression in the expression box.

```
#96a4b2
```

   This is the RGB hex value for a gray blue color used in the Slate style.

9. Click **OK**.

10. Click **Run** to preview your report.

The upper corner matrix is a single cell and has the same color as the row group and column group cells.

7. Add a Report Header and Report Title

A report title appears at the top of the report. You can place the report title in a report header or if the report does not use one, in a text box at the top of the report body. In this tutorial, you will remove the text box at the top of the report and add a title to the header.

To add a report header and report title

1. Click **Design** to return to design view.

2. Click the text box at the top of the report body that contains **Click to add title**, and then press the Delete key.

3. On the **Insert** tab of the ribbon, click **Header** and then click **Add Header**.

   A header is added to the top of the report body.

4. On the **Insert** tab, click **Text Box**, and then drag a text box inside the report header. Make the text box about 6 inches long and 3/4 inch tall and place it on the left side of the report header.

5. In the text box, type **Sales by Territory, Subcategory, and Day**.

6. Select the text you typed, right-click, and then click **Text Properties**.

   > **NOTE**
   >
   > To format characters at the same time, they must be contiguous.

7. In the **Text Properties** dialog box, click **Font**.

8. In the **Font** list, select **Times New Roman**; in **Size** select **24 pt**, in **Color** select **Maroon**, and in **Style** select **Italic**.

9. Click **OK**.

10. Click **Run** to preview the report.

The report includes a report title in the report header.

8. Save the Report

You can save reports to a report server, SharePoint library, or your computer.

In this tutorial, save the report to a report server. If you do not have access to a report server, save the report to your computer.

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.

2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you will see the contents of the report folder that the report server administrator specified as the default report location.
4. In **Name**, replace the default name with **SalesByTerritorySubcategory**.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report on your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop**, **My Documents**, or **My computer**, and then browse to the folder where you want to save the report.
3. In **Name**, replace the default name with **SalesByTerritorySubcategory**.
4. Click **Save**.

9. (Optional) Rotate Text Box 270 Degrees

A report with matrices can expand horizontally and vertically when it runs. By rotating text boxes vertically, or 270 degrees, you can save horizontal space. The rendered report is then narrower and if exported to a format such as Microsoft Word, will be more likely to fit on a printed page.

A text box can also display text as horizontal, vertical (top to bottom). For more information, see Text Boxes (Report Builder and SSRS).

To rotate text box 270 degrees

1. Click **Design** to return to design view.
2. Click the cell that contains `[Territory].`
3. In the Properties pane, locate the WritingMode property and in its drop-down list, select **Rotate270**.

   If the Properties pane is not open, click the **View** tab of the ribbon, and then select **Properties**.
4. Verify that the CanGrow property is set to `True`.
5. Resize the Territory column to be 1/2 inch wide and delete the column title.
6. Click **Run** to preview your report.

The territory name is written vertically, bottom to top. The height of the Territory row group varies by the length of the territory name.

Next Steps

This concludes the tutorial for how to create a matrix report. For more information about matrices, see Tables, Matrices, and Lists (Report Builder and SSRS), Matrices (Report Builder and SSRS), Tablix Data Region Areas (Report Builder and SSRS), and Tablix Data Region Cells, Rows, and Columns (Report Builder) and SSRS

See Also

Tutorials (Report Builder)

Tutorial: Creating a Free Form Report (Report Builder)

2/14/2019 • 19 minutes to read • Edit Online

This tutorial teaches you how to create an SSRS free form report that resembles a forms letter. You can arrange report items to create a form with text boxes, images and other data regions.

The report you create in this tutorial is based on sample sales data that is included in the tutorial. The report groups information by territory and displays the name of the sales manager for the territory as well as detailed and summary sales information. You will use the list data region as the foundation for the free form report, and then add a decorative panel with an image, static text with data inserted, a table to show detailed information, and optionally, pie and column charts to display summary information.

What You Will Learn

In this tutorial, you will learn how to do the following:

- Create a Blank Report, Data Source, and Dataset
- Add and Configure a List
- Add Graphics
- Add Free Form Text
- Add a Table to Show Details
- Format Data
- Save the Report

Other Optional Steps

- Add a Line to Separate Areas of Report
- Add Summary Data Visualization

Estimated time to complete this tutorial: 20 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Blank Report, Data Source, and Dataset

> **NOTE**
> In this tutorial, the query contains the data values so that the report does not need an external data source. The use of this type of internal data is great for learning purposes, but the approach makes the query long. .

To create a blank report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

> **NOTE**
>
> The **Getting Started** dialog box should appear. If it does not, from the Report Builder button, click **New**.

2. In the left pane of the **Getting Started** dialog box, verify that **New Report** is selected.
3. In the right pane, click **Blank Report**.

To create a new data source

1. In the Report Data pane, click **New**, and then click **Data Source**.
2. In the `Name` box, type: **ListDataSource**
3. Click **Use a connection embedded in my report**.
4. Verify that the connection type is Microsoft SQL Server, and then in the **Connection string** box type: **Data Source = <servername>**

   <servername>, for example Report001, specifies a computer on which an instance of the SQL Server Database Engine is installed. Because the report data is not extracted from a SQL Server database, you need not include the name of a database. The default database on the specified server is used to parse the query.
5. Click **Credentials**, and enter the credentials needed to connect to the instance of the SQL Server Database Engine.
6. Click **OK**.

To create a new dataset

1. In the Report Data pane, click **New**, and then click **Dataset**.
2. In the `Name` box, type: **ListDataset.**
3. Click **Use a dataset embedded in my report**, and verify that the data source is **ListDataSource**.
4. Verify that the **Text** query type is selected, and then click **Query Designer**.
5. Click **Edit as Text**.
6. Copy and paste the following query into the query pane:

```
SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as Territory,
'Accessories' as Subcategory,'Carrying Case' as Product, CAST(16996.60 AS money) AS Sales, 68 as
Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(13747.25 AS money) AS Sales, 55 as
Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Carrying Case' as Product, CAST(9248.15 AS money) As Sales, 37 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory,'Tripod' as Product, CAST(1350.00 AS money) AS Sales, 18 as
Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(1800.00 AS money) AS Sales, 24 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(1125.00 AS money) AS Sales, 15 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(1147.50 AS money) AS Sales, 17
as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory,  'Lens Adapter' as Product, CAST(742.50 AS money) AS Sales, 11 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(1417.50 AS money) AS Sales, 21 as Quantity
```

```
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(13497.30 AS money) AS Sales,
54 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(11997.60 AS money) AS Sales, 48 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(10247.95 AS money) As Sales, 41 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory, 'Tripod' as Product, CAST(1200.00 AS money) AS Sales, 16 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(2025.00 AS money) AS Sales, 27 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(1425.00 AS money) AS Sales, 19 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(887.50 AS money) AS Sales, 13 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory, 'Lens Adapter' as Product, CAST(607.50 AS money) AS Sales, 9 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(1215.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Lauren Johnson' as FullName,'Central' as
Territory, 'Digital' as Subcategory,'Compact Digital' as Product, CAST(10191.00 AS money) AS Sales, 79
as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Warren Pal' as FullName,'North' as Territory,
'Digital' as Subcategory, 'Compact Digital' as Product, CAST(8772.00 AS money) AS Sales, 68 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Fernando Ross' as FullName,'South' as Territory,
'Digital' as Subcategory, 'Compact Digital' as Product, CAST(10578.00 AS money) AS Sales, 82 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory,'Digital' as Subcategory, 'Slim Digital' as Product, CAST(7218.10 AS money) AS Sales, 38 as
Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as
Territory,'Digital' as Subcategory, 'Slim Digital' as Product, CAST(8357.80 AS money) AS Sales, 44 as
Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as
Territory,'Digital' as Subcategory,'Slim Digital' as Product, CAST(9307.55 AS money) AS Sales, 49 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Lauren Johnson' as FullName,'Central' as
Territory, 'Digital' as Subcategory,'Compact Digital' as Product, CAST(3870.00 AS money) AS Sales, 30 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Warren Pal' as FullName,'North' as Territory,
'Digital' as Subcategory,'Compact Digital' as Product, CAST(5805.00 AS money) AS Sales, 45 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Fernando Ross' as FullName,'South' as Territory,
'Digital' as Subcategory, 'Compact Digital' as Product, CAST(8643.00 AS money) AS Sales, 67 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Digital' as Subcategory, 'Slim Digital' as Product, CAST(9877.40 AS money) AS Sales, 52 as
Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Digital' as Subcategory, 'Slim Digital' as Product, CAST(12536.70 AS money) AS Sales, 66 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Digital' as Subcategory, 'Slim Digital' as Product, CAST(6648.25 AS money) AS Sales, 35 as Quantity
```

7. Click Run icon to run the query.

   The query results are the data available to display in your report.

SalesDate	FullName	Territory	Subcategory	Product	Sales	Quantity
1/5/2009 12:00:...	Fernando Ross	South	Accessories	Carrying Case	9248.1500	37
1/5/2009 12:00:...	Fernando Ross	South	Accessories	Lens Adapter	1417.5000	21
1/5/2009 12:00:...	Fernando Ross	South	Accessories	Tripod	1125.0000	15
1/5/2009 12:00:...	Fernando Ross	South	Digital	Compact Digital	10578.000	82
1/5/2009 12:00:...	Fernando Ross	South	Digital	Slim Digital	9307.5500	49
1/5/2009 12:00:...	Lauren Johnson	Central	Accessories	Carrying Case	16996.6000	68
1/5/2009 12:00:...	Lauren Johnson	Central	Accessories	Lens Adapter	1147.5000	17
1/5/2009 12:00:...	Lauren Johnson	Central	Accessories	Tripod	1350.0000	18
1/5/2009 12:00:...	Lauren Johnson	Central	Accessories	Compact Digital	10191.0000	79
1/5/2009 12:00:...	Lauren Johnson	Central	Accessories	Slim Digital	7218.1000	38
1/5/2009 12:00:...	Warren Pal	North	Accessories	Carrying Case	13747.2500	55
1/5/2009 12:00:...	Warren Pal	North	Accessories	Lens Adapter	742.5000	11
1/5/2009 12:00:...	Warren Pal	North	Accessories	Tripod	1800.0000	24
1/5/2009 12:00:...	Warren Pal	North	Accessories	Compact Digital	8772.0000	68

8. Click **OK**.

2. Add and Configure a List

Reporting Services provides three data region templates: table, matrix, and list. These templates are all based on a tablix data region.

In this tutorial, you will use a list to display the sales information for sales territories in a report that resembles a newsletter. The information is grouped by territory. You will add a new row group that groups data by territory, and then delete the built-in Details row group. The list template is ideal for creating free form reports. For more information, see Lists (Report Builder and SSRS).

> **NOTE**
> This report uses the paper size Letter (8.5 X11) and 1 inch margins. A report page taller than 9 inches or wider than 6 1/2 inches might generate blank pages.

To add a list

1. On the **Insert** tab of the ribbon, in the **Data Regions** area, click **List** and then drag the list inside the report body. Make the list 7 inches tall and 6.25 inches wide.

2. Click inside the list, right-click the top of the list, and then click **Tablix Properties**.

3. In the **Dataset name** drop-down list, select **ListDataset**.

4. Click **OK**.

5. Right-click inside the list, and then click **Rectangle Properties**.

6. On the **General** tab, select the **Add a page break after** checkbox.

7. Click **OK**.

To add a new row group and to delete the Details group

1. In the Row Groups pane, right-click the Details group, point to **Add Group**, and then click **Parent Group**.

2. In the drop-down list, select `[Territory].`

3. Click **OK**.

   A column is added to the list. The column contains the cell `[Territory].`

4. Right-click the Territory column in the list, and then click **Delete Columns**.

5. Click **Delete Columns only**.

6. In the Row Groups pane, right-click the **Details** group, and then click **Delete Group**.

7. Click **Delete Group only**.

8. Click **OK**.

3. Add Graphics

One of the advantages of using a list data region is that you can add report items such as rectangles and text boxes anywhere, instead of being limited to a tabular layout. You will enhance the appearance of the report by adding a graphic (a rectangle filled with a color).

To add graphic elements to the report

1. On the **Insert** tab of the ribbon, click **Rectangle**,and then drag a rectangle to the upper left corner of the list. Make the rectangle 7 inches tall and 1 inch wide.

2. Right-click the rectangle, and then click **Rectangle Properties**.

3. Click the **Fill** tab.

4. In the **Fill color** drop-down list, click **More Colors**, and then select the **DarkGray** color.

5. Click **OK**.

6. Click **Run** to preview the report.

The left side of the report now has vertical graphic that consists of a dark gray rectangle.

4. Add Free Form Text

A text box contains static text that is repeated on each report page as well as data fields.

To add text to the report

1. Click **Design** to return to design view.

2. On the **Insert** tab of the ribbon, click **Text Box**, and then drag a text box to the upper left corner of the list, but inside of the rectangle you added previously. Make the text box about 3 inches tall and 5 inches wide.

3. Place the cursor in the upper part of the text box, and then type: **Newsletter for** .

> **NOTE**
>
> Be sure to include the extra space after the word "for". The space separates the text and the field that you will add in the next step.

4. Drag the Territory field to the text box and place it after the text you typed in step 3.

5. Select all text, right-click, and then click **Text Properties**.

6. Click the **Font** tab.

7. In the **Font** list, select **Times New Roman**; in **Size** select **20 pt**, in **Color** select **Red**.

8. Click **OK**.

9. Place the cursor below the text you typed in step 3 and type: **Hello** .

> **NOTE**
>
> Be sure to include the extra space after the word "Hello". The space separates the text and the field that you will add in the next step.

10. Drag the FullName field to the text box and place it after the text you typed in step 9, and then type a comma (,).

11. Select the text you added in steps 9 and 10, right-click, and then click **Text Properties**.

12. Click the **Font** tab.

13. In the **Font** list, select **Times New Roman**; in **Size** select **16 pt**, in **Color** select **Black** color.

14. Click **OK**.

15. Place the cursor below the text you added in steps 9 through 13, and then copy and paste the following "greeked" text:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed dolor in ipsum pulvinar egestas. Sed
sed lacus at leo ornare ultricies. Vivamus velit risus, euismod nec sodales gravida, gravida in dui.
Etiam ullamcorper elit vitae justo fermentum ut ullamcorper augue sodales. Ut placerat, nisl quis
feugiat adipiscing, nibh est aliquet est, mollis faucibus mauris lectus quis arcu. In mollis tincidunt
lacinia. In vitae erat ut lorem tincidunt luctus. Curabitur et magna nunc, sit amet adipiscing nisi.
Nulla rhoncus elementum orci nec tincidunt. Aliquam imperdiet cursus erat vel tincidunt. Donec et neque
ac urna rutrum sodales. In id purus et nisl dignissim dapibus. Sed rhoncus metus at felis feugiat eu
tempor dolor vehicula. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam faucibus
consectetur diam eu pellentesque.
Nulla facilisi. Proin ligula enim, porta ut tincidunt id, adipiscing sit amet eros. Ut purus sem,
bibendum et vulputate sit amet, facilisis eget magna. Sed aliquam erat non erat eleifend hendrerit. Ut a
ligula est, sit amet eleifend enim. Ut et nisl enim, sit amet adipiscing augue. Vivamus eu arcu ac
libero posuere elementum. Integer condimentum bibendum venenatis. Integer odio tellus, feugiat in
pellentesque semper, interdum nec sem. Sed cursus euismod sem, ut elementum sapien placerat vel.
```

16. Select the text you added in steps 15, right-click, and then click **Text Properties**.

17. Click the **Font** tab.

18. In the **Font** list, select **Arial**; in **Size** select **10 pt**, in **Color** select **Black**.

19. Click **OK**.

20. Place the cursor below the text you pasted in step 15, and then type: **Congratulations on your total sales of** .

> **NOTE**
>
> Be sure to include the extra space after the word "of". The space separates the text and the field that you will add in the next step.

21. Drag the Sales field to the text box, place it after the text you typed in step 20, and then type an exclamation mark (!).

22. Highlight the Sales field, right-click the field, and then click **Expression**.

23. In the expression box, change the expression to include the Sum function as follows:

```
=Sum(Fields!Sales.value)
```

24. Click **OK**.

25. Select the text you added in steps 20 through 23, right-click, and then click **Text Properties**.

26. Click the **Font** tab.

27. In the **Font** list, select **Times New Roman**; in **Size** select **16 pt**, in **Color** select **Red**.

28. Click **OK**.

29. Select `[Sum(Sales)]` and on the **Home** tab, in the **Number** group, click the **Currency** button.

30. Right-click the text box with the "Click to add title" text, and then click **Delete**.

31. Select the list box and using the arrow keys, move it to the top of the page.

32. Click **Run** to preview the report.

The report displays static text and each report page includes data that pertains to a territory. Sales are formatted as currency.

5. Add a Table to Show Sales Details

Use the New Table and Matrix Wizard to add a table to the free form report. After you complete the wizard, you will manually add a row for totals.

To add a table

1. On the **Insert** tab of the ribbon, in the **Data Regions** area, click **Table**, and then click **Table Wizard**.

2. On the Choose a dataset page, click **ListDataset**.

3. Click **Next**.

4. On the **Arrange fields** page, drag the Product field from **Available fields** to **Values.**

5. Repeat step 4, for SalesDate, Quantity, and Sales. Place SalesDate below Product, Quantity below SalesDate, and Sales below SalesDate.

6. Click **Next**.

7. On the **Choose the layout** page, view the layout of the table.

   The table is very simple. It consists of five columns and has no row or column groups. Because it has no groups, the layout options related to groups, are not available. You will manually update the table to include a total later in the tutorial.

8. Click **Next**.

9. On the **Choose a Style** page, in the **Styles** pane, select **Slate**.

10. Click **Finish**.

11. Drag the table to below the text box that you added in lesson 4.

> **NOTE**
>
> Make sure the table is inside the list.

12. Confirmed that the table is selected, then in the Row Group pane right-click Details, point to **Add Total**, and then click **After**.

13. Click **Run** to preview the report.

The report displays a table with sales details and totals.

Nullam nec hendrerit tortor. Morbi ac vestibulum erat. Nunc imperdiet nibh in erat luctus, at hendrerit purus pharetra. Maecenas vitae metus quam. Proin ut eleifend enim. Sed congue neque vel semper venenatis. Donec molestie ultricies quam, sed rutrum magna laoreet in. Proin est mi, molestie at lobortis vel, iaculis eu diam. Curabitur volutpat varius nunc fermentum mattis. Etiam odio massa, pellentesque et enim nec, placerat porta enim. Morbi ut est sed arcu pulvinar fringilla. Fusce vulputate sit amet mauris a sollicitudin. Maecenas pulvinar diam eros, at sodales erat ornare a.

Congratulations on your total sales of $66,235.40!

Product	Sales Date	Quantity	Sales
Carrying Case	1/5/2009 12:00:00 AM	68	16996.6000
Lens Adapter	1/5/2009 12:00:00 AM	17	1147.5000
Tripod	1/5/2009 12:00:00 AM	18	1350.0000
Compact Digital	1/5/2009 12:00:00 AM	79	10191.0000
Slim Digital	1/5/2009 12:00:00 AM	38	7218.1000
Carrying Case	1/6/20 09 12:00:00 AM	54	13497.3000
Lens Adapter	1/6/2009 12:00:00 AM	13	887.5000
Tripod	1/6/2009 12:00:00 AM	16	1200.0000
Compact Digital	1/6/2009 12:00:00 AM	30	3870.0000
Slim Digital	1/6/2009 12:00:00 AM	52	9877.4000
		385	66235.4000

7/11/2014 4:45:50 PM

6. Format Data

Format numeric data as currency and dates as day and time only.

To format fields table

1. Click **Design** to switch to design view.
2. Click the table cells that contain `[Sum(SalesSales)]` and on the **Home** tab, in the **Number** group, click the **Currency** button.

3. Click the cell that contains `[SalesDate]` and in the **Number** group, from the drop-down list, select **Date**.

4. Click **Run** to preview the report.

The report now displays formatted data and is easier to read.

Congratulations on your total sales of $66,235.40!

Product	Sales Date	Quantity	Sales
Carrying Case	1/5/2009	68	$16,996.60
Lens Adapter	1/5/2009	17	$1,147.50
Tripod	1/5/2009	18	$1,350.00
Compact Digital	1/5/2009	79	$10,1919.00
Slim Digital	1/5/2009	38	$7,218.10
Carrying Case	1/6/2009	54	$13,497.30
Lens Adapter	1/6/2009	13	$887.50
Tripod	1/6/2009	16	$1,200.00
Compact Digital	1/6/2009	30	$3,870.00
Slim Digital	1/6/2009	52	$9,877.50
		385	$66,235.40

7/11/2014 5:01:38 PM

7. Save the Report

You can save reports to a report server, SharePoint library, or your computer. You can also export the report to a variety of formats such as Word and PDF, by running the report and selecting the format from the **Export** menu.

In this tutorial, save the report to a report server. If you do not have access to a report server, save the report to your computer.

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.
2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you see the contents of the report folder that the report server administrator specified as the default location for reports.
4. In `Name`, replace the default name with **SalesInformationByTerritory**.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report on your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop**, **My Documents**, or **My computer**, and then browse to the folder where you want to save the report.
3. In `Name`, replace the default name with **SalesInformationByTerritory**.
4. Click **Save**.

8. (Optional) Add a Line to Separate Areas of the Report

Add a line to separate the editorial and details areas of the report.

To add a line

1. Click **Design** to return to design view.
2. On the **Insert** tab of the ribbon, in the **Report Items** area, click **Line.**
3. Draw a line below the free form text box you added in lesson 4.
4. Click the line.
5. Click the **Home** tab.
6. In the **Border** area, for width select **4 1/2** pt and for color, for color select **Red**.

Newsletter for Central
Hello Lauren Johnson,

Nullam nec hendrerit tortor. Morbi ac vestibulum erat. Nunc imperdiet nibh in erat luctus, at hendrerit purus pharetra. Maecenas vitae metus quam. Proin ut eleifend enim. Sed congue neque vel semper venenatis. Donec molestie ultricies quam, sed rutrum magna laoreet in. Proin est mi, molestie at lobortis vel, iaculis eu diam. Curabitur volutpat varius nunc fermentum mattis. Etiam odio massa, pellentesque et enim nec, placerat porta enim. Morbi ut est sed arcu pulvinar fringilla. Fusce vulputate sit amet mauris a sollicitudin. Maecenas pulvinar diam eros, at sodales erat ornare a.

Congratulations on your total sales of $66,235.40!

Product	Sales Date	Quantity	Sales
Carrying Case	1/5/2009	68	$16,996.60
Lens Adapter	1/5/2009	17	$1,147.50
Tripod	1/5/2009	18	$1,350.00
Compact Digital	1/5/2009	79	$10,1919.00
Slim Digital	1/5/2009	38	$7,218.10
Carrying Case	1/6/2009	54	$13,497.30
Lens Adapter	1/6/2009	13	$887.50
Tripod	1/6/2009	16	$1,200.00
Compact Digital	1/6/2009	30	$3,870.00
Slim Digital	1/6/2009	52	$9,877.50
		385	$66,235.40

9. (Optional) Add Summary Data Visualization

Rectangles help you control how the report renders. Place a pie and column chart inside a rectangle to ensure that the report renders the way you want.

To add a rectangle

1. Click **Design** to return to design view.
2. On the **Insert** tab of the ribbon, in the **Report Items** area click **Rectangle**, and then drag the rectangle inside the list, to the right of the table. Make the rectangle 2 inches wide and 4 inches tall.
3. Align the tops of the rectangle and the table.

To add a pie chart

1. On the **Insert** tab of the ribbon, in the **Data Visualizations** area, click **Chart,** and then click **Chart Wizard**.
2. On the Choose a dataset page, click **ListDataset**, and then click **Next**.
3. Click **Pie**, and then click **Next**.
4. On the arrange chart fields page, drag Product to **Categories**.
5. Drag Quantity to **Values**, and then click **Next**.
6. On the **Choose a Style** page, in the **Styles** pane, select **Slate**.
7. Click **Finish**.

8. Resize the chart that appears in the upper left corner of the report, to be 1 1/2 inches tall and 2 inches wide.

9. Drag the chart inside the rectangle.

10. Right-click the chart title, and then click **Title Properties**.

11. In the **Chart Title Properties** dialog box, in Title text, type: **Product Quantities Sold**.

12. Click the **Font** tab, and in the **Size** list, click **10pt**.

13. Click **OK**.

To add a column chart

1. On the **Insert** tab of the ribbon, in the **Data Visualizations** area, click **Chart,** and then click **Chart Wizard**.

2. On the **Choose a dataset** page, click **ListDataset**, and then click **Next**.

3. Click **Column**, and then click **Next**.

4. On the arrange chart fields page, drag the Product field to **Categories**.

5. Drag Sales to **Values,** and then click **Next**.

   Values display on the vertical axis.

6. On the **Choose a Style** page, in the **Styles** pane, select **Slate**.

7. Click **Finish**.

   A column chart is added to the upper left corner of the report.

8. Resize the chart to be 2 inches wide and 2 inches tall.

9. Drag the chart inside the rectangle, below the pie chart.

10. Right-click the chart title and then click **Title Properties**.
11. In the **Chart Title Properties** dialog box, in Title text, type: **Product Sales**.
12. Click the **Font** tab, and in the **Size** list click **10pt**, and then click **OK**.
13. In the column chart, right click the vertical axis, and then deselect **Show Axis Title**.
14. Repeat step 13 for the horizontal axis.
15. Right click the legend, and then click **Delete Legend**.

> **NOTE**
>
> Removing axis titles and the legend makes the chart more readable when it is a small size.

To verify the charts are inside the rectangle

1. Click the rectangle you added earlier in this lesson.

   In the Properties pane, the `Name` property displays the name of the rectangle.

2. Click the pie chart.

3. In the **Properties** pane, verify that the `Parent` property contains the name of the rectangle.

4. Click the column chart and repeat steps 2 and 3.

> **NOTE**
>
> If the charts are not inside the rectangle, the rendered report does not display the charts together.

To make the charts the same size

1. Click the pie chart, press the Ctrl key, and then click the column chart.

2. With both charts selected, right-click, point to **Layout**, and then click **Make Same Width**.

> **NOTE**
>
> The item you click first determines the width of all the selected items.

3. Click **Run** to preview the report.

The report now displays summary sales data in pie and column charts.

Product	Sales Date	Quantity	Sales
Carrying Case	1/5/2009	68	$16,996.60
Lens Adapter	1/5/2009	17	$1,147.50
Tripod	1/5/2009	18	$1,350.00
Compact Digital	1/5/2009	79	$10,1919.00
Slim Digital	1/5/2009	38	$7,218.10
Carrying Case	1/6/2009	54	$13,497.30
Lens Adapter	1/6/2009	13	$887.50
Tripod	1/6/2009	16	$1,200.00
Compact Digital	1/6/2009	30	$3,870.00
Slim Digital	1/6/2009	52	$9,877.50
		385	$66,235.40

More Information

For more information about lists, see Tables, Matrices, and Lists (Report Builder and SSRS), Lists (Report Builder and SSRS), Tablix Data Region Areas (Report Builder and SSRS), and Tablix Data Region Cells, Rows, and Columns (Report Builder) and SSRS.

For more information about query designers, see Query Designers (Report Builder) and Text-based Query Designer User Interface (Report Builder).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Format Text (Report Builder)

2/14/2019 • 15 minutes to read • Edit Online

In this tutorial, you can practice formatting text in various ways. After you set up the blank report with the data source and dataset, you can pick and choose the steps that you want to explore.

The following illustration shows a report similar to the one you will create.

Territory	Link Text	Product	Sales
Central	Installing Report Builder 3.0	Carrying Case	$30,493.90
		Compact Digital	$14,061.00
		Lens Adapter	$2,035.00
		Slim Digital	$17,095.50
		Tripod	$2,550.00
		Total	$66,235.40
	Total		$66,235.40
North	Getting Started with Report Builder 3.0	Carrying Case	$24,744.85
		Compact Digital	$14,577.00
		Lens Adapter	$1,350.00
		Slim Digital	$20,894.50
		Tripod	$3,825.00
		Total	$66,391.35
	Total		$66,391.35
South	What Is New in Report Builder 3.0	Carrying Case	$19,496.10
		Compact Digital	$19,221.00
		Lens Adapter	$2,632.50
		Slim Digital	$15,955.80
		Tripod	$2,550.00
		Total	$59,855.40
	Total		$59,855.40
Total			$192,482.15

Limitations of cascading style sheets

Only a basic set of **cascading style sheet (CSS)** attributes are supported:

- Text-align, text-indent
- font-family, font-size
- padding, padding-bottom, padding-top, padding-right, padding-left
- font-weight

In one step, you make a mistake on purpose so you can see why it is a mistake. Then you correct the mistake to achieve the desired effect.

An enhanced version of the report you create in this tutorial is available as a sample SQL Server 2014 Report Builder report. For more information about downloading this sample report and others, see Report Builder sample reports.

What You Will Learn

Set Up the Report

1. Create a Blank Report with a Data Source and Dataset
2. Add a Field to the Report Design Surface (Incorrectly, then Correctly)
3. Add a Table to the Report Design Surface

Pick and Choose

Add a Hyperlink to the Report

Rotate Text in the Report

Displaying Text with HTML Formatting

Format Currency

Save the Report

Estimated time to complete this tutorial: 20 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

Create a Blank Report with a Data Source and Dataset

To create a blank report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2014Report Builder**, and then click **Report Builder**.

   > **NOTE**
   > The **Getting Started** dialog box should appear. If it does not, from the Report Builder button, click **New**.

2. In the left pane of the **Getting Started** dialog box, verify that **New Report** is selected.

3. In the right pane, click **Blank Report**.

To create a data source

1. In the Report Data pane, click **New**, and then click **Data Source**.

2. In the **Name** box, type: **TextDataSource**

3. Click **Use a connection embedded in my report**.

4. Verify that the connection type is Microsoft SQL Server, and then in the **Connection string** box type: **Data Source = <servername>**

   > **NOTE**
   > The expression <servername>, for example Report001, specifies a computer on which an instance of the SQL Server Database Engine is installed. This tutorial does not need specific data; it just needs a connection to a SQL Server 2014 database. If you already have a data source connection listed under **Data Source Connections**, you can select it and go to the next procedure, "To create a dataset." For more information, see Alternative Ways to Get a Data Connection (Report Builder).

5. Click **OK**.

To create a dataset

1. In the Report Data pane, click **New**, and then click **Dataset**.

2. Verify that the data source is **TextDataSource**.

3. In the **Name** box, type: **TextDataset.**

4. Verify that the **Text** query type is selected, and then click **Query Designer**.

5. Click **Edit as Text**.

6. Paste the following query into the query pane:

```
SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as Territory,
'Accessories' as Subcategory,'Carrying Case' as Product, CAST(16996.60 AS money) AS Sales, 68 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
```

```
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(13747.25 AS money) AS Sales, 55 as
Quantity, 'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?
LinkId=160556' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Carrying Case' as Product, CAST(9248.15 AS money) As Sales, 37 as
Quantity, 'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS
URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory,'Tripod' as Product, CAST(1350.00 AS money) AS Sales, 18 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(1800.00 AS money) AS Sales, 24 as Quantity,
'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=160556' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(1125.00 AS money) AS Sales, 15 as Quantity, 'What
is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(1147.50 AS money) AS Sales, 17
as Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882'
AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory,  'Lens Adapter' as Product, CAST(742.50 AS money) AS Sales, 11 as
Quantity, 'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?
LinkId=160556' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(1417.50 AS money) AS Sales, 21 as Quantity,
'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(13497.30 AS money) AS Sales,
54 as Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?
LinkId=154882' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(11997.60 AS money) AS Sales, 48 as
Quantity, 'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?
LinkId=160556' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory, 'Carrying Case' as Product, CAST(10247.95 AS money) As Sales, 41 as
Quantity, 'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS
URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory, 'Tripod' as Product, CAST(1200.00 AS money) AS Sales, 16 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(2025.00 AS money) AS Sales, 27 as Quantity,
'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=160556' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Tripod' as Product, CAST(1425.00 AS money) AS Sales, 19 as Quantity, 'What
is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(887.50 AS money) AS Sales, 13 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Accessories' as Subcategory, 'Lens Adapter' as Product, CAST(607.50 AS money) AS Sales, 9 as Quantity,
'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=160556' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Accessories' as Subcategory,'Lens Adapter' as Product, CAST(1215.00 AS money) AS Sales, 18 as Quantity,
'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Lauren Johnson' as FullName,'Central' as
Territory, 'Digital' as Subcategory,'Compact Digital' as Product, CAST(10191.00 AS money) AS Sales, 79
as Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882'
AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Warren Pal' as FullName,'North' as Territory,
'Digital' as Subcategory, 'Compact Digital' as Product, CAST(8772.00 AS money) AS Sales, 68 as Quantity,
'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=160556' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate,  'Fernando Ross' as FullName,'South' as Territory,
```

```
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Digital' as Subcategory, 'Compact Digital' as Product, CAST(10578.00 AS money) AS Sales, 82 as
Quantity, 'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS
URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory,'Digital' as Subcategory, 'Slim Digital' as Product, CAST(7218.10 AS money) AS Sales, 38 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as
Territory,'Digital' as Subcategory, 'Slim Digital' as Product, CAST(8357.80 AS money) AS Sales, 44 as
Quantity, 'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?
LinkId=160556' AS URL
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as
Territory,'Digital' as Subcategory,'Slim Digital' as Product, CAST(9307.55 AS money) AS Sales, 49 as
Quantity, 'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS
URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Lauren Johnson' as FullName,'Central' as
Territory, 'Digital' as Subcategory,'Compact Digital' as Product, CAST(3870.00 AS money) AS Sales, 30 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Warren Pal' as FullName,'North' as Territory,
'Digital' as Subcategory,'Compact Digital' as Product, CAST(5805.00 AS money) AS Sales, 45 as Quantity,
'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=160556' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Fernando Ross' as FullName,'South' as Territory,
'Digital' as Subcategory, 'Compact Digital' as Product, CAST(8643.00 AS money) AS Sales, 67 as Quantity,
'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Lauren Johnson' as FullName,'Central' as
Territory, 'Digital' as Subcategory, 'Slim Digital' as Product, CAST(9877.40 AS money) AS Sales, 52 as
Quantity, 'Installing Report Builder' as LinkText, 'https://go.microsoft.com/fwlink/?LinkId=154882' AS
URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Warren Pal' as FullName,'North' as Territory,
'Digital' as Subcategory, 'Slim Digital' as Product, CAST(12536.70 AS money) AS Sales, 66 as Quantity,
'Getting Started with Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=160556' AS URL
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Fernando Ross' as FullName,'South' as Territory,
'Digital' as Subcategory, 'Slim Digital' as Product, CAST(6648.25 AS money) AS Sales, 35 as Quantity,
'What is New in Report Builder' as Link, 'https://go.microsoft.com/fwlink/?LinkId=165064' AS URL
```

7. Click Run (**!**) to run the query.

   The query results are the data available to display in your report.

8. Click **OK**.

Add a Field to the Report Design Surface

If you want a field from your dataset to appear in a report, your first impulse may be to drag it directly to the design surface. This exercise points out why that doesn't work and what to do instead.

To add a field to the report (and get the wrong result)

1. Drag the **FullName** field from the Report Data pane to the design surface.

   Report Builder creates a text box with an expression in it, represented as <Expr>.

2. Click **Run**.

   Note that there is just one record, **Fernando Ross**, which is alphabetically the first record in the query. The field does not repeat to show the other records in that field.

3. Click **Design** to return to design view.

4. Select the expression <Expr> in the text box.

5. In the Properties pane, for the **Value** property, you see the following (if you don't see the Properties pane, on the **View** tab, check **Properties**):

```
=First(Fields!FullName.Value, "TextDataSet")
```

The `First` function is designed to retrieve only the first value in a field, and that is what it has done.

Dragging the field directly to the design surface created a text box. Text boxes by themselves are not data regions, so they do not display data from a report dataset. Text boxes in data regions, such as tables, matrices, and lists, do display data.

6. Select the text box (if you have the expression selected, press ESC to select the text box), and press the DELETE key.

To add a field to the report (and get the right result)

1. On the **Insert** tab of the ribbon, in the **Data Regions** area, click **List**. Click the design surface, and then drag to create a box that about two inches wide and one inch tall.
2. Drag the **FullName** field from the Report Data pane to the list box.

   This time Report Builder creates a text box with the expression `[FullName]` in it.
3. Click **Run**.

   Note that this time the box repeats to show all the records in the query.
4. Click **Design** to return to design view.
5. Select the expression in the text box.
6. In the Properties pane, for the **Value** property, you see the following:

   ```
   =Fields!FullName.Value
   ```

   By dragging the text box to the list data region, you display the data that is in the dataset.
7. Select the list box and press the DELETE key.

Add a Table to the Report Design Surface

Create this table so that you'll have a place to put hyperlinks and rotated text.

To add a table to the report

1. On the **Insert** menu, click **Table**, and then click **Table Wizard**.
2. On the **Choose a dataset** page of the New Table or Matrix wizard, click **Choose an existing dataset in this report or a shared dataset**, and click **TextDataset (in this Report)**, and then click **Next**.
3. On the **Arrange fields** page, drag the **Territory**, **LinkText**, and **Product** fields to **Row groups**, drag the **Sales** field to **Values**, and then click **Next**.
4. On the **Choose the layout** page, clear the **Expand/collapse groups** check box so you can see the whole table, and then click **Next**.
5. On the **Choose a style** page, click **Slate**, and then click **Finish**.
6. Drag the table so it is below the title block.
7. Click **Run**.

   The table looks OK, but it has two Total rows. The **LinkText** field doesn't need a Total row.
8. Click **Design** to return to design view.

9. Right-click the text box that contains `[LinkText]`, and click **Split Cells**.
10. Select the empty cell below the `[LinkText]` cell, and then hold down the SHIFT key and select the two cells to its right: the **Total** cell in the **Product** column and the `[Sum(Sales)]` cell in the **Sales** column.
11. With those three cells selected, right-click one of those cells and click **Delete Row**.
12. Click **Run**.

Add a Hyperlink to the Report

In this section, you add a hyperlink to text in the table from the previous section.

To add a hyperlink to the report

1. Click **Design** to return to design view.
2. Right-click in the cell containing `[LinkText]`, and click **Text Box Properties**.
3. In the **Text Box Properties** box, click **Action**.
4. Click **Go to URL**.
5. In the **Select URL** box, click **[URL]**, and then click **OK**.
6. Note that the text does not look any different. You need to make it look like link text.
7. Select `[LinkText]`.
8. In the **Font** section of the **Home** tab, click the **Underline** button, and then click the drop-down arrow next to the **Color** button, and click **Blue**.
9. Click **Run**.

   The text now looks like a link.
10. Click a link. If the computer is connected to the Internet, a browser will open to a Report Builder Help topic.

Rotate Text in the Report

In this section, you rotate some of the text in the table from the previous sections.

To rotate text

1. Click **Design** to return to design view.
2. Click in the cell containing `[Territory].`
3. On the **Home** tab in the **Font** section, click the **Bold** button.
4. If the Properties pane is not open, on the **View** tab, select the **Properties** check box.
5. Locate the WritingMode property in the Properties pane.

   > **NOTE**
   > When the properties in the Properties pane are organized into categories, WritingMode is in the **Localization** category. Be sure you have selected the cell and not the text. WritingMode is a property of the text box, not of the text.

6. In the list box, click **Rotate270**.
7. On the **Home** tab in the **Paragraph** section, click the **Middle** and **Center** buttons to locate the text in the

center of the cell both vertically and horizontally.

8. Click Run (**!**).

Now the text in the `[Territory]` cell runs vertically from the bottom to the top of the cells.

Displaying Text with HTML Formatting

To display text formatted as HTML

1. Click **Design** to switch to design view.

2. On the **Insert** tab, click **Text Box**, and then on the design surface, click and drag to create a text box under the table, about four inches wide and three inches tall.

3. Copy this text and paste it into the text box:

```
<h4>Limitations of cascading style sheet attributes</h4>
      <p>Only a basic set of <b>cascading style sheet (CSS)</b> attributes are defined:</p>
      <ul><li>
          text-align, text-indent
        </li><li>
          font-family, font-size
        </li><li>
          color
        </li><li>
          padding, padding-bottom, padding-top, padding-right, padding-left
        </li><li>
          font-weight
        </li></ul>
```

4. Select all of the text in the text box.

   This is a property of the text, not the text box, so in one text box you could have a mixture of plain text and text that uses HTML tags as styles.

5. Right-click all of the selected text and click **Text Properties**.

6. On the **General** page, under **Markup type**, click **HTML - Interpret HTML tags as styles**.

7. Click **OK**.

8. Click Run (**!**) to preview the report.

The text in the text box is displayed as a heading, paragraph, and bulleted list.

Format Currency

To format numbers as currency

1. Click **Design** to switch to design view.

2. Click the top table cell that contains `[Sum(Sales)]`, hold down the SHIFT key, and click the bottom table cell that contains `[Sum(Sales)]`.

3. On the **Home** tab, in the **Number** group, click the **Currency** button.

4. (Optional) On the **Home** tab, in the **Number** group, click the **Placeholder Styles** button and click **Sample Values** to see how the numbers will be formatted.

5. (Optional) On the **Home** tab, in the **Number** group, click the **Decrease Decimals** button twice to display dollar figures with no cents.

6. Click Run (**!**) to preview the report.

The report now displays formatted data and is easier to read.

Save the Report

You can save reports to a report server, SharePoint library, or your computer.

In this tutorial, save the report to a report server. If you do not have access to a report server, save the report to your computer.

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.
2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you see the contents of the report folder that the report server administrator specified as the default location for reports.
4. In **Name**, replace the default name with a name of your choosing.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report on your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop**, **My Documents**, or **My computer**, and then browse to the folder where you want to save the report.
3. In **Name**, replace the default name with a name of your choosing.
4. Click **Save**.

Next Steps

There are many ways to format text in Report Builder Tutorial: Creating a Free Form Report (Report Builder) contains more examples.

See Also

Tutorials (Report Builder)
Formatting Report Items (Report Builder and SSRS)
Report Builder in SQL Server 2014

Tutorial: Add a Column Chart to Your Report (Report Builder)

2/14/2019 • 8 minutes to read • Edit Online

A column chart displays a series as a set of vertical bars that are grouped by category. A column chart can be useful to:

- Show data changes over a period of time.
- Compare the relative value of multiple series.
- Display a moving average to show trends.

The following illustration shows the column chart you will create, with a moving average.

What You Will Learn

In this tutorial you will learn how to do the following:

1. Create a Chart from the Chart Wizard
2. Choose the Chart Type
3. Format and Label the Horizontal Axis
4. Move the Legend
5. Title the Chart
6. Format and Label the Vertical Axis
7. Add a Moving Average
8. Add a Report Title
9. Save the Report

> **NOTE**
> In this tutorial, the steps for the wizard are consolidated into one procedure. For step-by-step instructions about how to browse to a report server, choose a data source, and create a dataset, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

Estimated time to complete this tutorial: 15 minutes.

Requirements

For information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Chart Report from the Chart Wizard

From the **Getting Started** dialog box, use the Chart Wizard to create an embedded dataset, choose a shared data source, and create a column chart.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a new chart report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box appears.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, from the **Report Builder** button, click **New**.

2. In the left pane, verify that **New Report** is selected.

3. In the right pane, click **Chart Wizard**.

4. On the **Choose a dataset page**, click **Create a dataset**, and then click **Next**.

5. On the **Choose a connection to a data source** page, select an existing data source or browse to the report server and select a data source, and then click **Next**. You may need to enter a user name and password.

   > **NOTE**
   > The data source you choose is unimportant, as long as you have adequate permissions. You will not be getting data from the data source. For more information, see Alternative Ways to Get a Data Connection (Report Builder).

6. On the **Design a query** page, click **Edit as Text**.

7. Paste the following query into the query pane:

```
SELECT CAST('2009-01-01' AS date) AS SalesDate, CAST(54995.21 AS money) AS Sales
UNION SELECT CAST('2009-01-05' AS date) AS SalesDate, CAST(64499.04 AS money) AS Sales
UNION SELECT CAST('2009-02-11' AS date) AS SalesDate, CAST(37821.79 AS money) AS Sales
UNION SELECT CAST('2009-03-18' AS date) AS SalesDate, CAST(53633.08 AS money) AS Sales
UNION SELECT CAST('2009-04-23' AS date) AS SalesDate, CAST(24019.3 AS money) AS Sales
UNION SELECT CAST('2009-05-01' AS date) AS SalesDate, CAST(93245.5 AS money) AS Sales
UNION SELECT CAST('2009-06-06' AS date) AS SalesDate, CAST(55288.0 AS money) AS Sales
UNION SELECT CAST('2009-06-16' AS date) AS SalesDate, CAST(68733.5 AS money) AS Sales
UNION SELECT CAST('2009-07-16' AS date) AS SalesDate, CAST(24750.85 AS money) AS Sales
UNION SELECT CAST('2009-08-23' AS date) AS SalesDate, CAST(43452.3 AS money) AS Sales
UNION SELECT CAST('2009-09-24' AS date) AS SalesDate, CAST(58656. AS money) AS Sales
UNION SELECT CAST('2009-10-15' AS date) AS SalesDate, CAST(44583. AS money) AS Sales
UNION SELECT CAST('2009-11-21' AS date) AS SalesDate, CAST(81568. AS money) AS Sales
UNION SELECT CAST('2009-12-15' AS date) AS SalesDate, CAST(45973. AS money) AS Sales
UNION SELECT CAST('2009-12-26' AS date) AS SalesDate, CAST(96357. AS money) AS Sales
UNION SELECT CAST('2009-12-31' AS date) AS SalesDate, CAST(81946. AS money) AS Sales
```

8. (Optional) Click the Run button (**!**) to see the data your chart will be based on.
9. Click **Next**.

2. Choose the Chart Type

You can choose from a variety of predefined chart types.

To add a column chart

1. On the **Choose a chart type** page, the column chart is the default chart type. Click **Next**.
2. On the **Arrange chart fields** page, drag the SalesDate field to **Categories**. Categories display on the horizontal axis.
3. Drag the Sales field to **Values**. The **Values** box displays Sum(Sales) because the sum of the sales total value is aggregated for each date. Values display on the vertical axis.
4. Click **Next**.
5. On the **Choose a Style** page, in the Styles box, select a style.

   A style specifies a font style, a set of colors, and a border style. When you select a style, the Preview pane displays a sample of the chart with that style.
6. Click **Finish**.

   The chart is added to the design surface.
7. Click the chart to display the chart handles. Drag the bottom-right corner of the chart to increase the size of the chart. Note that the report design surface increases in size to accommodate the chart size.
8. Click **Run** to preview the report.

3. Format and Label the Horizontal Axis

By default, the horizontal axis displays values in a general format that is automatically scaled to fit the size of the chart.

To format a date on the horizontal axis

1. Switch to report design view.
2. Right-click the horizontal axis, and then click **Horizontal Axis Properties**.
3. Click **Number**.

4. In **Category**, select **Date**.
5. In the **Type** box, select **31 Jan 2000**.
6. Click **OK**.
7. On the Home tab, click **Run** to preview the report.

The date displays in the date format that you selected. Notice that the chart does not label every category on the horizontal axis. By default, only labels that fit next to the axis are included.

You can customize the label display by rotating the labels and specifying the interval.

To rotate the axis labels and change the display interval along the horizontal axis

1. Switch to report design view.
2. Right-click the horizontal axis title, and then click **Show Axis Title** to remove the title. Because the horizontal axis displays dates, the title is not needed.
3. Right-click the horizontal axis and then click **Horizontal Axis Properties**.
4. In the **Axis Options** page under **Axis range and interval**, type **3** for **Interval**. The chart will display every third date.
5. Click **Labels**.
6. In **Change axis label auto-fit options**, select **Disable auto-fit**.
7. In **Label rotation angle**, select **-90**.
8. Click **OK**.

   The sample text for the horizontal axis rotates by 90 degrees.
9. Click **Run** to preview the report.

On the chart, the labels are rotated and the label for every third date is shown.

4. Move the Legend

The legend is automatically created from category and series data.

To move the legend below the chart area of a column chart

1. Switch to report design view.
2. Right-click the legend on the chart, and then click **Legend Properties**.
3. For **Layout and Position**, select a different position. For example, set the position to the middle bottom option.

   When the legend is placed at the top or bottom of a chart, the layout of the legend changes from vertical to horizontal. You can select a different layout from the **Layout** drop-down list.
4. Click **OK**.
5. (Optional) Because there is only one category in this tutorial, the legend is not needed. To remove the legend, right-click the legend and then click **Delete Legend**.
6. Click **Run** to preview the report.

5. Title the Chart

To change the chart title above the chart area

1. Switch to report design view.
2. Select the words **Chart Title** at the top of the chart, and then type the following text: **Store Sales Order Totals**.
3. Click **Run** to preview the report.

6. Format and Label the Vertical Axis

By default, the vertical axis displays values in a general format that is automatically scaled to fit the size of the chart.

To format as currency the numbers on the vertical axis

1. Switch to report design view.
2. Double-click the labels on the vertical axis along the side of the chart to select them.
3. On the ribbon, on the **Home** tab, in the **Number** group, click the **Currency** button. The axis labels change to show the currency format.
4. On the ribbon, on the **Home** tab, in the **Number** group, click the **Decrease Decimal** button two times, to show the number rounded to the nearest dollar.
5. Right-click the vertical axis and click **Vertical Axis Properties**.
6. Click **Number**. Note that **Currency** is already selected in the **Category** box, and **Decimal places** is already **0** (zero).
7. In the **Show Values in** box, click **Thousands**.
8. Click **OK**.
9. Right-click the vertical axis title along the side of the chart and click **Axis Title Properties**.
10. Replace the text in the **Title text** field with the following text: **Sales Total (in Thousands)**. You can also specify a variety of options related to how the title is formatted.
11. Click **OK**.
12. Click **Run** to preview the report.

7. Add a Moving Average

To add a moving average

1. Switch to report design view.
2. Double-click the chart to display the **Chart Data** pane.
3. Right-click the **[Sum(Sales)]** field that is in the **Values** area, and then click **Add Calculated Series**.
4. In **Formula**, verify that **Moving average** is selected.
5. In **Set Formula Parameters**, for **Period**, select **4**.
6. Click **Border**.
7. In **Line width**, select **3pt**.
8. Click **OK**.
9. Click **Run** to preview the report.

The chart displays a line that shows the moving average for total sales by date, averaged over every four dates.

8. Add a Report Title

To add a report title

1. Switch to report design view.
2. On the design surface, click **Click to add title**.
3. Type **Sales Chart**, press ENTER, and then type **January to December 2009**, so it looks like this:

   Sales Chart

   January to December 2009
4. Select **Sales Chart**, and click the **Bold** button in the **Font** section on the **Home** tab of the ribbon.
5. Select **January to December 2009**, and in the **Font** section on the **Home** tab, set the font size to **10**.
6. (Optional) You may need to make the **Title** text box taller to accommodate the two lines of text by pulling down on the double-headed arrows when you click in the middle of the bottom edge.

   This title will appear at the top of the report. When there is no page header defined, items at the top of the report body are the equivalent of a report header.
7. Click **Run** to preview the report.

9. Save the Report

To save the report

1. Switch to report design view.
2. From the Report Builder button, click **Save As**.
3. In **Name**, type **Sales Order Column Chart**.
4. Click **Save**.

Next Steps

You have successfully completed the Adding a Column Chart to Your Report tutorial. To learn more about charts, see Charts (Report Builder and SSRS) and Sparklines and Data Bars (Report Builder and SSRS).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Add a Pie Chart to Your Report (Report Builder)

2/14/2019 • 7 minutes to read • Edit Online

Pie charts and doughnut charts display data as a proportion of the whole. Pie charts are most commonly used to make comparisons between groups. Pie and doughnut charts, along with pyramid and funnel charts, constitute a group of charts known as shape charts. Shape charts have no axes. When a numeric field is dropped on a shape chart, the chart calculates the percentage of each value to the total.

If there are too many data points on a pie chart, your data point labels might be too crowded to read. In that case, consider using a line chart. Consider using pie charts only after you have aggregated your data into a few data points.

The following illustration shows the pie chart you will create.

What You Will Learn

In this tutorial, you will learn how to:

1. Create a Pie Chart from the Chart Wizard
2. Choose the Chart Type
3. Display Percentages in Each Slice
4. Combine Small Slices into One Slice
5. Customize the Drawing Effect
6. Add a Report Title
7. Save the Report

> **NOTE**
> In this tutorial, the steps for the wizard are consolidated into two procedures. For step-by-step instructions about how to browse to a report server, add a data source, and add a dataset, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

Estimated time to complete this tutorial: 10 minutes

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Pie Chart from the Chart Wizard

From the Getting Started dialog box, use the Chart Wizard to create an embedded dataset, choose a shared data source, and create a pie chart.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a new chart report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   The Getting Started dialog box appears.

   > **NOTE**
   > If the Getting Started dialog box does not appear, from the Report Builder button, click **New**.

2. In the left pane, verify that **New Report** is selected.
3. In the right pane, click **Chart Wizard**.
4. On the **Choose a dataset** page, click **Create a dataset**, and then click **Next**.
5. On the **Choose a connection to a data source** page, select an existing data source or browse to the report server and select a data source, and then click **Next**. You may need to enter a user name and password.

   > **NOTE**
   > The data source you choose is unimportant, as long as you have adequate permissions. You will not be getting data from the data source. For more information, see Alternative Ways to Get a Data Connection (Report Builder).

6. On the **Design a Query** page, click **Edit as Text**.
7. Paste the following query into the query pane:

```
SELECT 'Advanced Digital Camera' AS Product, CAST(254995.21 AS money) AS Sales
UNION SELECT 'Slim Digital Camera' AS Product, CAST(164499.04 AS money) AS Sales
UNION SELECT 'SLR Digital Camera' AS Product, CAST(782176.79 AS money) AS Sales
UNION SELECT 'Lens Adapter' AS Product, CAST(36333.08 AS money) AS Sales
UNION SELECT 'Macro Zoom Lens' AS Product, CAST(40199.3 AS money) AS Sales
UNION SELECT 'USB Cable' AS Product, CAST(53245.5 AS money) AS Sales
UNION SELECT 'Independent Filmmaker Camcorder' AS Product, CAST(452288.0 AS money) AS Sales
UNION SELECT 'Full Frame Digital Camera' AS Product, CAST(247250.85 AS money) AS Sales
```

8. (Optional) Click the Run button (**!**) to see the data your chart will be based on.

9. Click **Next**.

2. Choose the Chart Type

You can choose from a variety of predefined chart types.

To add a pie chart

1. On the **Choose a chart type** page, click **Pie**, and then click **Next**. The **Arrange chart fields** page opens.

   On the **Arrange chart fields** page, drag the Product field to the **Categories** pane. Categories define the number of slices in the pie chart. In this example, there will be eight slices, one for each product.

2. Drag the Sales field to the **Values** pane. Sales represents the sales amount for the subcategory. The **Values** pane displays `[Sum(Sales)]` because the chart displays the aggregate for each product.

3. Click **Next**.

4. On the **Choose a style** page, in the Styles pane, select a style.

   A style specifies a font style, a set of colors, and a border style. When you select a style, the Preview pane displays a sample of the chart with that style.

5. Click **Finish**.

   The chart is added to the design surface.

6. Click the chart to display the chart handles. Drag the bottom-right corner of the chart to increase the size of the chart. Note that the report design surface increases in size to accommodate the chart size.

7. Click **Run** to preview the report.

The report displays the pie chart with eight slices, one for each product. The size of each slice represents the sales for that product. Three of the slices are quite thin.

3. Display Percentages in Each Slice

On each slice of the pie, you can display a percentage for this slice compared to the whole pie.

To display percentages in each slice of the pie chart

1. Switch to report design view.

2. Right-click the pie chart and click **Show Data Labels**. The data labels appear on the chart.

3. Right-click a label, and then click **Series Label Properties**.

4. In Label data, from the drop-down box, select **#PERCENT**.

   To display values as percentages, the UseValueAsLabel property must be false. If you are prompted to set this value in the **Confirm Action** dialog, click **Yes**.

5. (Optional) To specify how many decimal places the label shows, type `#PERCENT{Pn}` where *n* is the number of decimal places to display. For example, to display no decimal places, type `#PERCENT{P0}`.

> **NOTE**
>
> **Number Format** in the **Series Label Properties** dialog box has no effect when you format percentages. This formats the labels as percentages, but does not calculate the percentage of the pie that each slice represents.

6. Click **OK**.
7. Click **Run** to preview the report.

The report displays the percentage of the whole for each pie slice.

4. Combine Small Slices into One Slice

Three of the slices in the pie are quite small. You can combine multiple small slices into one larger slice that represents all of them.

To combine any slices on the pie chart smaller than 5 percent into one slice

1. Switch to report design view.
2. On the **View** tab, in the **Show/Hide** group, select **Properties**.
3. On the design surface, click on any slice of the pie chart. The properties for the series are displayed in the Properties pane.
4. In the **General** section, expand the **CustomAttributes** node.
5. Set the **CollectedStyle** property to **SingleSlice**.
6. Verify that the **CollectedThreshold** property is set to 5.
7. Verify that the **CollectedThresholdUsePercent** property is set to **True**.
8. On the Ribbon, on the **Home** tab, click **Run** to preview the report.

In the legend, the category "Other" now exists. The new pie slice combines all the slices that were under 5% into one slice that is 6% of the whole pie.

5. Customize the Drawing Effect

In the Chart Wizard, the default style for a pie chart is Ocean, which features a Concave drawing effect. You can change that after you run the wizard.

To add a drawing effect to the pie chart

1. Switch to report design view.
2. If the Properties pane is not already opened, on the **View** tab, select **Properties**.
3. Double-click the pie chart itself. The series properties for the pie chart are shown in the Properties pane.
4. In the Properties pane, expand the **CustomAttributes** node.
5. Set the **PieDrawingStyle** to **SoftEdge**.

> **NOTE**
>
> Drawing effects and three-dimensional effects are exclusive options. If a chart has three-dimensional effects applied, **PieDrawingStyle** is not available on the Properties pane.

6. Click **Run** to preview the report.

The following illustration shows the pie chart with the soft edge effect.

6. Add a Report Title

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Camera and Camcorder Sales**, press ENTER, and then type **As a Percentage of Total Sales**, so it looks like this:

   Camera and Camcorder Sales

   As a Percentage of Total Sales
3. Select **Camera and Camcorder Sales**, and click the **Bold** button from the **Font** section of the **Home** tab of the ribbon.
4. Select **As a Percentage of Total Sales**, and in the **Font** section on the **Home** tab, set the font size to **10**.
5. (Optional) You may need to make the Title text box taller to accommodate the two lines of text.

   This title will appear at the top of the report. When there is no page header defined, items at the top of the report body are the equivalent of a report header.
6. Click **Run** to preview the report.

7. Save the Report

To save the report

1. Switch to report design view.
2. From the Report Builder button, click **Save As**.

3. In **Name**, type **Sales Pie Chart**.

4. Click **Save**.

Your report is saved on the report server.

Next Steps

You have successfully completed the Adding a Pie Chart to Your Report tutorial. To learn more about charts, see Charts (Report Builder and SSRS) and Sparklines and Data Bars (Report Builder and SSRS).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Add a Bar Chart to Your Report (Report Builder)

2/14/2019 • 10 minutes to read • Edit Online

A bar chart displays category data horizontally. This can help to:

- Improve readability of long category names.
- Improve understandability of times plotted as values.
- Compare the relative value of multiple series.

The following illustration shows the bar chart that you will create, with sales for 2008 and 2009 for the top five salespeople, in alphabetical order.

What You Will Learn

In this tutorial you will learn how to do the following:

1. Create a Chart from the Chart Wizard
2. Choose the Chart Type
3. Display all Category Values on the Vertical Axis
4. Modify the Display of Names on the Vertical Axis
5. Move the Legend
6. Move the Chart Title

7. Format and Label the Horizontal Axis
8. Add a Filter to Display the Top Five Values
9. Add a Report Title
10. Save the Report

> **NOTE**
> In this tutorial, the steps for the wizard are consolidated into one procedure. For step-by-step instructions about how to browse to a report server, create a dataset, and choose a data source, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

Estimated time to complete this tutorial: 15 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Chart Report from the Chart Wizard

From the **Getting Started** dialog box, create an embedded dataset, choose a shared data source, and create a bar chart by using the Chart Wizard.

> **NOTE**
> In this tutorial, the query contains the data values so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a new chart report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box appears.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, click the Report Builder button, and then click **New**.

2. In the left pane, verify that **New Report** is selected.
3. In the right pane, click **Chart Wizard**.
4. On the **Choose a dataset** page, click **Create a dataset**, and then click **Next**.
5. On the **Choose a connection to a data source** page, select an existing data source or browse to the report server and select a data source, and then click **Next**. You may need to enter a user name and password.

   > **NOTE**
   > The data source you choose is unimportant, as long as you have adequate permissions. You will not be getting data from the data source. For more information, see Alternative Ways to Get a Data Connection (Report Builder).

6. On the **Design a query** page, click **Edit as Text**.

7. Paste the following query into the query pane:

```
SELECT 'Luis' as FirstName, 'Alverca' as LastName, CAST(170000.00 AS money) AS SalesYear2009,
CAST(150000. AS money) AS SalesYear2008
UNION SELECT 'Jeffrey' as FirstName, 'Zeng' as LastName, CAST(210000. AS money) AS SalesYear2009,
CAST(190000. AS money) AS SalesYear2008
UNION SELECT 'Houman' as FirstName, 'Pournasseh' as LastName, CAST(150000. AS money) AS SalesYear2009,
CAST(180000. AS money) AS SalesYear2008
UNION SELECT 'Robin' as FirstName, 'Wood' as LastName, CAST(75000. AS money) AS SalesYear2009,
CAST(175000. AS money) AS SalesYear2008
UNION SELECT 'Daniela' as FirstName, 'Guaita' as LastName,  CAST(170000. AS money) AS SalesYear2009,
CAST(175000. AS money) AS SalesYear2008
UNION SELECT 'John' as FirstName, 'Yokim' as LastName, CAST(160000. AS money) AS SalesYear2009,
CAST(195000. AS money) AS SalesYear2008
UNION SELECT 'Delphine' as FirstName, 'Ribaute' as LastName, CAST(180000. AS money) AS SalesYear2009,
CAST(205000. AS money) AS SalesYear2008
UNION SELECT 'Robert' as FirstName, 'Hernady' as LastName, CAST(140000. AS money) AS SalesYear2009,
CAST(180000. AS money) AS SalesYear2008
UNION SELECT 'Tanja' as FirstName, 'Plate' as LastName, CAST(150000. AS money) AS SalesYear2009,
CAST(160000. AS money) AS SalesYear2008
UNION SELECT 'David' as FirstName, 'Bradley' as LastName, CAST(210000. AS money) AS SalesYear2009,
CAST(180000. AS money) AS SalesYear2008
UNION SELECT 'Michal' as FirstName, 'Jaworski' as LastName, CAST(175000. AS money) AS SalesYear2009,
CAST(220000. AS money) AS SalesYear2008
UNION SELECT 'Chris' as FirstName, 'Ashton' as LastName, CAST(195000. AS money) AS SalesYear2009,
CAST(205000. AS money) AS SalesYear2008
UNION SELECT 'Pongsiri' as FirstName, 'Hirunyanitiwatna' as LastName, CAST(175000. AS money) AS
SalesYear2009, CAST(215000. AS money) AS SalesYear2008
UNION SELECT 'Brian' as FirstName, 'Burke' as LastName, CAST(187000. AS money) AS SalesYear2009,
CAST(207000. AS money) AS SalesYear2008
```

8. (Optional) Click the Run button (**!**) to see the data your chart will be based on.

9. Click **Next**.

2. Choose the Chart Type

You can choose from a variety of predefined chart types.

To add a column chart

1. On the **Choose a chart type** page, the column chart is the default chart type.

2. Click **Bar**, and then click **Next**.

   On the **Arrange chart fields** page, there are four fields in the **Available fields** pane: FirstName, LastName, SalesYear2009, and SalesYear2008.

3. Drag LastName to the Categories pane.

4. Drag SalesYear2009 to the Values pane. SalesYear2009 represents the sales amount for each salesperson for the year 2009. The Values pane displays `[Sum(SalesYear2009)]` because the chart displays the aggregate for each product.

5. Drag SalesYear2008 to the Values pane under SalesYear2009. SalesYear2008 represents the sales amount for each salesperson for the year 2008.

6. Click **Next**.

7. On the **Choose a style** page, in the Styles pane, select a style.

   A style specifies a font style, a set of colors, and a border style. When you select a style, the Preview pane

displays a sample of the chart with that style.

8. Click **Finish**.

   The chart is added to the design surface.

9. Click the chart to display the chart handles. Drag the bottom-right corner of the chart to increase the size of the chart.

10. Click **Run** to preview the report.

The report displays the bar chart for sales for each sales person for the years 2008 and 2009. The length of the bar corresponds to the sales total.

3. Modify the Display of Names on the Vertical Axis

By default, only some of the values on the vertical axis appear. You can change the chart to display all categories.

To display all sales persons along the category axis of a bar chart

1. Switch to report design view.
2. Right-click the vertical axis, and then click **Vertical Axis Properties**.
3. Under **Axis range and interval**, in the **Interval** box, type **1**.
4. Click **OK**.
5. Right-click the vertical **Axis Title** and clear the **Show Axis Title** check box.
6. Click **Run** to preview the report.

> **NOTE**
> If you cannot read the salesperson names on the vertical axis, you can make your chart taller or change the formatting options for the axis labels.

Display Last Name and First Name on Vertical Axis

You can change the category expression to include last name followed by first name of each sales person.

To change the category expression

1. Switch to report design view.
2. Double-click the chart to display the **Chart Data** pane.
3. In the **Category Groups** area, right-click [LastName], and then click **Category Group Properties**.
4. In Label, click the expression (Fx) button.
5. Type the following expression: `=Fields!LastName.Value & ", " & Fields!FirstName.Value`

   This expression concatenates the last name, a comma, and the first name.

6. Click **OK**.
7. Click **OK**.
8. Click **Run** to preview the report.

If the first names do not appear when you run the report, you can refresh the data manually. While still in preview mode, on the **Run** tab in the **Navigation** group, click **Refresh**.

> **NOTE**
> If you cannot read the salesperson names on the vertical axis, you can make your chart taller or change the formatting options for the axis labels.

4. Change the Sort Order for Names on the Vertical Axis

When you sort the data on a chart, you are changing the order of values on the category axis.

To sort the names in alphabetical order on the bar chart

1. Switch to report design view.
2. Double-click the chart to display the **Chart Data** pane.
3. In the **Category Groups** area, right-click [LastName], and then click **Category Group Properties**.
4. Click **Sorting**. The **Change sorting options** page displays a list of sort expressions. By default, this list has one sort expression that is the same as the original category group expression.
5. In Sort by, click the expression (**Fx**) button.
6. Type the following expression: `=Fields!LastName.Value & ", " & Fields!FirstName.Value`
7. Click **OK**.
8. Back on the **Category Group Properties** page, in the **Order** drop-down list, select **Z to A**. This selects reverse alphabetical order so that the names appear in order from top to bottom.
9. Click **OK**.
10. Click **Run** to preview the report.

The names on the horizontal axis are sorted in reverse order, with **Alerca** at the top and **Zeng** at the bottom.

5. Move the Legend

To improve the readability of the chart values, you might want to move the chart legend. For example, in a bar chart where bars are shown horizontally, you can change the position of the legend so that it is above or below the chart area. This gives more horizontal space to the bars.

To display the legend below the chart area of a bar chart

1. Switch to report design view.
2. Right-click the legend on the chart.
3. Select **Legend Properties**.
4. For **Legend position**, select a different position. For example, set the position to the middle bottom option.

   When the legend is placed at the top or bottom of a chart, the layout of the legend changes from vertical to horizontal. You can select a different layout from the **Layout** drop-down list.
5. Click **OK**.
6. Click **Run** to preview the report.

6. Title the Chart

To change the chart title above the chart area of a bar chart

1. Switch to report design view.

2. Select the words **Chart Title** at the top of the chart, and then type the following text: **Sales for 2008 and 2009**.
3. Click anywhere outside the text.
4. Click **Run** to preview the report.

7. Format and Label the Horizontal Axis

By default, the horizontal axis displays values in a general format that is automatically scaled to fit the size of the chart.

To format the numbers on the horizontal axis

1. Switch to report design view.
2. Click the horizontal axis along the bottom of the chart to select it.

   On the ribbon, on the **Home** tab, in the **Number** group, click the **Currency** button. The horizontal axis labels change to currency.
3. (Optional) Remove the decimal digits. Near the **Currency** button, click the **Decrease Decimal** button twice.
4. Right-click the horizontal axis, and click **Horizontal Axis Properties**.
5. On the **Number** tab, select **Show values in Thousands.**
6. Click **OK**.
7. Right-click **Axis Title** and click **Axis Title Properties**.
8. In the **Title text** box, type **Sales in thousands** and click **OK**.
9. Click **Run** to preview the report.

The report displays the sales amount on the horizontal axis as currency in thousands, and has no decimal digits.

8. Add a Filter to Display the Top Five Values

You can add a filter to the chart to specify which data from the dataset to include or exclude in the chart.

To add a filter and display the top five values

1. Switch to report design view.
2. Double-click the chart to display the **Chart Data** pane.
3. In the **Category Groups** area, right-click the [LastName] field, and then click **Category Group Properties**.
4. Click **Filters**. The **Change filters** page can display a list of filter expressions. By default, this list is empty.
5. Click **Add**. A new blank filter appears.
6. In **Expression**, type **[Sum(SalesYear2009)]**. This creates the underlying expression `=Sum(Fields!SalesYear2009.Value)`, which you can see if you click the **fx** button.
7. Verify that the data type is **Text**.
8. In **Operator**, select **Top N** from the drop-down list.
9. In **Value**, type the following expression: **=5**
10. Click **OK**.
11. Click **Run** to preview the report.

If the results are not filtered when you run the report, you can refresh the data manually. On the **Run** tab in the **Navigation** group, click **Refresh**.

The chart shows the top five salesperson names from the 2009 sales data.

9. Add a Report Title

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Sales Bar Chart**, press ENTER, and then type **Top Five Sellers for 2009**, so it looks like this:

   Sales Bar Chart

   Top Five Sellers for 2009
3. Select **Sales Bar Chart**, and click the **Bold** button.
4. Select **Top Five Sellers for 2009**, and in the **Font** section on the **Home** tab, set the font size to **10**.
5. (Optional) You may need to make the Title text box taller to accommodate the two lines of text.

   This title will appear at the top of the report. When there is no page header defined, items at the top of the report body are the equivalent of a report header.
6. Click **Run** to preview the report.

10. Save the Report

To save the report

1. Switch to report design view.
2. From the **Report Builder** button, click **Save As**.
3. In **Name**, type **Sales Bar Chart**.
4. Click **Save**.

Your report is saved on the report server.

Next Steps

You have successfully completed the Adding a Bar Chart to Your Report tutorial. To learn more about charts, see Charts (Report Builder and SSRS) and Sparklines and Data Bars (Report Builder and SSRS).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Add a Sparkline to Your Report (Report Builder)

2/14/2019 • 12 minutes to read • Edit Online

In this tutorial, you create a basic table report based on sample sales data, and then add a sparkline chart to a cell in the table.

An enhanced version of the report you create in this tutorial is available as a sample SQL Server 2014 Report Builder report. For more information about downloading this sample report and others, see Report Builder sample reports. The following illustration shows the sample report similar to the one that you will create.

The video How to: Create a Sparkline in a Table (Report Builder Video) illustrates how to create a similar report with sparklines.

What You Will Learn

In this tutorial, you will learn how to do the following:

1. Create a Report with a Table
2. Create a Query in the Table or Matrix Wizard
3. Add a Sparkline to the Table
4. Align the Sparklines Vertically and Horizontally

Other Optional Steps

5. Format Data as Currency
6. Format Data as Dates
7. Change Column Widths
8. Add a Report Title
9. Save the Report

Estimated time to complete this tutorial: 30 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Report with a Table

To create a report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box opens.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, from the **Report Builder** button, click **New**.

2. In the left pane, verify that **New Report** is selected.
3. In the right pane, click **Table or Matrix Wizard**.
4. On the **Choose a dataset** page, select **Create a dataset**, and then click **Next**. The **Choose a connection to a data source** page opens.

   > **NOTE**
   > This tutorial does not need specific data; it just needs a connection to a SQL Server 2014 database. If you already have a data source connection listed under **Data Source Connections**, you can select it and go to step 10. For more information, see Alternative Ways to Get a Data Connection (Report Builder).

5. Click **New**. The **Data Source Properties** dialog box opens.
6. In **Name**, type **Product Sales**, a name for the data source.
7. In **Select a connection type**, verify that **Microsoft SQL Server** is selected.
8. In **Connection string**, type the following text:

   Data Source=<servername>

   The expression <servername>, for example Report001, specifies a computer on which an instance of the SQL Server Database Engine is installed. Because the report data is not extracted from a SQL Server database, you need not include the name of a database. The default database on the specified server is used to parse the query.

9. Click **Credentials**. Enter the credentials that you need to access the external data source.
10. Click **OK**.

    You are back on the **Choose a connection to a data source** page.

11. To verify that you can connect to the data source, click **Test Connection**.

    The message "Connection created successfully" appears.

12. Click **OK**.
13. Click **Next**.

2. Create a Query in the Table Wizard

In a report, you can use a shared dataset that has a predefined query, or you can create an embedded dataset for use only in your report. In this tutorial, you will create an embedded dataset.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a query

1. On the **Design a query** page, the relational query designer is open. For this tutorial, you will use the text-based query designer.

2. Click **Edit As Text**. The text-based query designer displays a query pane and a results pane.

3. Paste the following Transact-SQL query into the **Query** box.

```
SELECT CAST('2010-01-04' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Carrying Case' as Product, CAST(16996.60 AS money) AS Sales, 68 as Quantity
UNION SELECT CAST('2010-01-05' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Carrying Case' as Product, CAST(1350.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2010-01-10' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Carrying Case' as Product, CAST(1147.50 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2010-01-04' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(1056.00 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2010-01-05' AS date) as SalesDate,  'Accessories' as Subcategory,
   'Slim Digital' as Product, CAST(1380.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2010-01-05' AS date) as SalesDate,'Accessories' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(780.00 AS money) AS Sales, 26 as Quantity
UNION SELECT CAST('2010-01-07' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(3798.00 AS money) AS Sales, 9 as Quantity
UNION SELECT CAST('2010-01-08' AS date) as SalesDate, 'Camcorders' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(10400.00 AS money) AS Sales, 13 as Quantity
UNION SELECT CAST('2010-01-09' AS date) as SalesDate, 'Camcorders' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(3000.00 AS money) AS Sales, 60 as Quantity
UNION SELECT CAST('2010-01-10' AS date) as SalesDate,  'Digital' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(7234.50 AS money) AS Sales, 39 as Quantity
UNION SELECT CAST('2010-01-06' AS date) as SalesDate,  'Digital' as Subcategory,
   'Carrying Case' as Product, CAST(10836.00 AS money) AS Sales, 84 as Quantity
UNION SELECT CAST('2010-01-07' AS date) as SalesDate,  'Digital' as Subcategory,
   'Slim Digital' as Product, CAST(2550.00 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2010-01-04' AS date) as SalesDate, 'Digital' as Subcategory,
   'Slim Digital' as Product, CAST(8357.80 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2010-01-08' AS date) as SalesDate, 'Digital SLR' as Subcategory,
   'Slim Digital' as Product, CAST(18530.00 AS money) AS Sales, 34 as Quantity
UNION SELECT CAST('2010-01-06' AS date) as SalesDate, 'Digital SLR' as Subcategory,
   'Slim Digital' as Product, CAST(26576.00 AS money) AS Sales, 88 as Quantity
```

4. On the query designer toolbar, click Run (**!**).

   The query runs and displays the result set for the fields **SalesDate**, **Subcategory**, **Product**, **Sales**, and **Quantity**.

5. Click **Next**.

6. On the **Arrange fields** page, drag **Sales** to **Values**.

   Sales is aggregated by the Sum function. The value is [Sum(Sales)].

7. Drag **Product** to **Row groups**.

8. Drag **SalesDate** to **Column groups**.

9. Click **Next**.

10. On the **Choose the layout** page, under **Options**, verify that **Show subtotals and grand totals** is selected.

The wizard Preview pane displays a table with three rows. When you run the report, each row will display in the following way:

a. The first row will appear once for the table to show column headings.

b. The second row will repeat once for each product and display the product name, total per day, and line total.

c. The third row will appear once for the table to display the grand totals.

11. Click **Next**.

12. On the **Choose a Style** page, in the **Styles** pane, select **Slate**.

    The Preview pane displays a sample of the table with that style.

13. Click **Finish**.

14. The table is added to the design surface. The table has three columns and three rows.

    Look in the Grouping pane. If you can't see the Grouping pane, on the **View** menu, click **Grouping**. The Row Groups pane shows one row group: **Product**. The Column Groups pane shows one column group: **SalesDate**. Detail data is all the data that is retrieved by the dataset query.

15. Click **Run** to preview the report.

3. Add a Sparkline

To add a sparkline chart to a table

1. Click **Design** to return to design view.

2. Select the Total column in your table.

3. Right-click, point to **Insert Column**, and then click **Left**.

4. In the new column, right-click in the [Product] row, point to the **Insert** ribbon tab, and then click **Sparkline**.

5. Make sure the first sparkline in the **Column** row is selected and then click **OK**.

6. Click the sparkline to show the Chart Data pane.

7. Click the plus (+) sign in the Values box, and then click **Sales**.

   The values in the **Sales** field are now the values for the sparkline.

8. Click the plus (+) sign in the Category Groups box, and then click **SalesDate**.

9. Click **Run** to preview your report.

   Note that there are sparkline charts in each row of the table, but they're not correct. The bars in the charts don't line up with each other. There are only four bars in the second row of data, so the bars are wider than the bars in the first row, which has six. You can't compare values for each product per day. They need to line up with each other.

   Also note that for each row, the tallest bar for that row is the height of the row. This is misleading, too, because the largest values for each row are not equal: the largest value for Budget Movie-Maker is $10,400, but the largest value for Slim Digital is $26,576-more than twice as large. And yet the largest bars in those two rows are about the same height. That also needs to be made to scale with the other sparklines.

4. Align the Sparklines Vertically and Horizontally

The sparklines are hard to read when they don't all use the same measurements. Both the horizontal and vertical axes for each need to match the rest.

To set alignment for the sparklines in the table

1. Click **Design** to return to design view.
2. Right-click the sparkline and click **Vertical Axis Properties**.
3. Check the **Align axes in** check box.

   Tablix1 is showing in the list. That is the only option. This sets the height of the bars in each sparkline relative to the others.
4. Click **OK**.
5. Right-click the sparkline and click **Horizontal Axis Properties**.
6. Check the **Align axes in** check box.

   Tablix1 is showing in the list. That is the only option. This sets the width of the bars in each sparkline relative to the others. If some sparklines have fewer bars than others, then those sparklines will have blank spaces for the missing data.
7. Click **OK**.
8. Click **Run** to preview your report again.

Note that all the bars are now aligned with the bars in the other rows.

5. (Optional) Format Data as Currency

By default, the summary data for the **Sales** field displays a general number. Format it to display the number as currency. Toggle **Placeholder Styles** to display formatted text boxes and placeholder text as sample values.

To format a currency field

1. Click **Design** to switch to design view.
2. Click the cell in the second row (under the column headings row) in the **SalesDate** column and drag to select all cells that contain `[Sum(Sales)]`.
3. On the **Home** tab, in the **Number** group, click the **Currency** button. The cells change to show the formatted currency.

   If your regional setting is English (United States), the default sample text is [**$12,345.00**]. If you do not see an example currency value, click **Placeholder Styles** in the **Numbers** group, and then click **Sample Values**.
4. Click **Run** to preview your report.

The summary values for **Sales** display as currency.

6. (Optional) Format Data as Dates

By default, the **SalesDate** field displays both date and time information. You can format them to display only the date.

To format a date field as the default format

1. Click **Design** to return to design view.
2. Click the cell that contains `[SalesDate]`.
3. On the Ribbon, on the **Home** tab, in the **Number** group, from the drop-down list, select **Date**.

   The cell displays the example date **[1/31/2000]**. If you do not see an example date, click **Placeholder Styles** in the **Numbers** group, and then click **Sample Values**.
4. Click **Run** to preview the report.

The **SalesDate** values display in the default date format.

7. (Optional) Change Column Widths

By default, each cell in a table contains a text box. A text box expands vertically to accommodate text when the page is rendered. In the rendered report, each row expands to the height of the tallest rendered text box in the row. The height of the row on the design surface has no affect on the height of the row in the rendered report.

To reduce the amount of vertical space each row takes, expand the column width to accommodate the expected contents of the text boxes in the column on one line.

To change the width of columns

1. Click **Design** to return to design view.
2. Click the table so that column and row handles appear above and next to the table.

   The gray bars along the top and side of the table are the column and row handles.
3. Point to the line between column handles so that the cursor changes into a double arrow. Drag the columns to the size you want. For example, expand the column for **Product** so that the product name displays on one line.
4. Click **Run** to preview your report.

8. (Optional) Add a Report Title

A report title appears at the top of the report. You can place the report title in a report header or if the report does not use one, in a text box at the top of the report body. In this tutorial, you will use the text box that is automatically placed at the top of the report body.

The text can be further enhanced by applying different font styles, sizes, and colors to phrases and individual characters of the text. For more information, see Format Text in a Text Box (Report Builder and SSRS).

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Product Sales**, and then click outside the text box.
3. Right-click the text box that contains **Product Sales** and click **Text Box Properties**.
4. In the **Text Box Properties** dialog box, click **Font**.
5. In the **Size** list, select **18pt**.

6. In the **Color** list, select **Maroon**.
7. Select **Bold**.
8. Click **OK**.

9. Save the Report

Save the report to a report server or your computer. If you do not save the report to the report server, a number of Reporting Services features such as report parts and subreports are not available.

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.
2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you see the contents of the report folder that the report server administrator specified as the default location for reports.
4. In **Name**, replace the default name with **Product Sales**.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report on your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop**, **My Documents**, or **My computer**, and browse to the folder where you want to save the report.
3. In **Name**, replace the default name with **Product Sales**.
4. Click **Save**.

Next Steps

This concludes the tutorial for creating a table report with sparkline charts. For more information about sparklines, see Sparklines and Data Bars (Report Builder and SSRS).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Adding a KPI to Your Report (Report Builder)

2/14/2019 • 12 minutes to read • Edit Online

A key performance indicator (KPI) is a measurable value that has business significance. This tutorial teaches you how to include a (KPI) in a report. In this scenario, the sales summary by product subcategories is the KPI. The current state of the KPI is shown by using colors, gauges, and indicators.

The following illustration shows the report that you will create.

Product Sales KPI

Sales Date	Subcategory	Product	Quantity	Sales	KPI	KPI
January 05, 2009	Accessories	Carrying Case	68	$16,996.00		
		Mini Battery Charger	44	$1,056.00		
			112	$18,052.60		
	Digital	Slim Digital	44	$8,357.80		
			44	$8,357.80		
			156	$26,410.40		
January 11, 2009	Accessories	Lens Adapter	17	$1,147.50		
			17	$1,147.50		
	Digital	Advanced Digital	39	$7,234.50		
			39	$7,234.50		
	Total		56	$8,382.00		
Total			576	$113,992.40		

What You Will Learn

In this tutorial, you will learn to add a KPI by setting the background color of table cells based on cell value and add and configure a gauge and an indicator. You also learn to write the expression that sets the background color of the table cells.

This tutorial contains the following procedures:

1. Create a Table Report and Dataset from the Table or Matrix Wizard
2. Organize Data, Choose Layout, and Style from the Table or Matrix Wizard
3. Use Background Colors to Display a KPI
4. Display a KPI by Using a Gauge
5. Display a KPI by Using an Indicator
6. Add a Report Title
7. Save the Report

> **NOTE**
>
> In this tutorial, the steps for the wizard are consolidated into two procedures: one to create the dataset and one to create a table. For step-by-step instructions about how to browse to a report server, choose a data source, create a dataset, and run the wizard, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

Estimated time to complete this tutorial: 15 minutes.

Requirements

For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Table Report and Dataset from the Table or Matrix Wizard

From the **Getting Started** dialog box, choose a shared data source, create an embedded dataset, and display the data in a table.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a new table

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2012 Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box appears.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, from the Report Builder button, click **New**.

2. In the left pane, verify that **New Report** is selected.
3. In the right pane, click **Table or Matrix Wizard**.
4. On the Choose a dataset page, click **Create a dataset**.
5. Click **Next**.
6. On the **Choose a connection to a data source** page, select an existing data source or browse to the report server and select a data source. If there no data source is available or you do not have access to a report server, you can use an embedded data source instead. For more information, see Tutorial: Creating a Basic Table Report (Report Builder).
7. Click **Next**.
8. On the **Design a query** page, click **Edit as Text**.
9. Copy and paste the following query into the query pane:

```
SELECT CAST('2009-01-05' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Carrying Case' as Product, CAST(16996.60 AS money) AS Sales, 68 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Tripod' as Product, CAST(1350.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-11' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Lens Adapter' as Product, CAST(1147.50 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Mini Battery Charger' as Product, CAST(1056.00 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,  'Accessories' as Subcategory,
   'Telephoto Conversion Lens' as Product, CAST(1380.00 AS money) AS Sales, 18 as Quantity
UNION SELECT CAST('2009-01-06' AS date) as SalesDate,'Accessories' as Subcategory,
   'USB Cable' as Product, CAST(780.00 AS money) AS Sales, 26 as Quantity
UNION SELECT CAST('2009-01-08' AS date) as SalesDate, 'Accessories' as Subcategory,
   'Budget Movie-Maker' as Product, CAST(3798.00 AS money) AS Sales, 9 as Quantity
UNION SELECT CAST('2009-01-09' AS date) as SalesDate, 'Camcorders' as Subcategory,
   'Business Videographer' as Product, CAST(10400.00 AS money) AS Sales, 13 as Quantity
UNION SELECT CAST('2009-01-10' AS date) as SalesDate, 'Camcorders' as Subcategory,
   'Social Videographer' as Product, CAST(3000.00 AS money) AS Sales, 60 as Quantity
UNION SELECT CAST('2009-01-11' AS date) as SalesDate,  'Digital' as Subcategory,
   'Advanced Digital' as Product, CAST(7234.50 AS money) AS Sales, 39 as Quantity
UNION SELECT CAST('2009-01-07' AS date) as SalesDate,  'Digital' as Subcategory,
   'Compact Digital' as Product, CAST(10836.00 AS money) AS Sales, 84 as Quantity
UNION SELECT CAST('2009-01-08' AS date) as SalesDate,  'Digital' as Subcategory,
   'Consumer Digital' as Product, CAST(2550.00 AS money) AS Sales, 17 as Quantity
UNION SELECT CAST('2009-01-05' AS date) as SalesDate, 'Digital' as Subcategory,
   'Slim Digital' as Product, CAST(8357.80 AS money) AS Sales, 44 as Quantity
UNION SELECT CAST('2009-01-09' AS date) as SalesDate, 'Digital SLR' as Subcategory,
   'SLR Camera 35mm' as Product, CAST(18530.00 AS money) AS Sales, 34 as Quantity
UNION SELECT CAST('2009-01-07' AS date) as SalesDate, 'Digital SLR' as Subcategory,
   'SLR Camera' as Product, CAST(26576.00 AS money) AS Sales, 88 as Quantity
```

10. Click **Next**.

2. Organize Data, Choose Layout, and Style from the Table or Matrix Wizard

Use the wizard to provide a starting design on which to display data. The preview pane in the wizard helps you to visualize the result of grouping data before you complete the table or matrix design.

To organize data into groups, choose a layout and a style

1. On the Arrange fields page, drag Product to **Values**.
2. Drag Quantity to **Values** and place below Product.

   Quantity is summarized with the Sum function, the default function to summarize numeric fields.
3. Drag Sales to **Values** and place below Quantity.

   Steps 1, 2, and 3 specify the data to display in the table.
4. Drag SalesDate to **Row groups**.
5. Drag Subcategory to **Row groups** and place below SalesDate.

   Steps 4 and 5 organize the values for the fields first by date, and then by all sales for that date.
6. Click **Next**.

   When you run the report, the table displays each date, all orders for each date, and all products, quantities, and sales totals for each order.
7. On the Choose the Layout page, under **Options**, verify that **Show subtotals and grand totals** is selected.

8. Verify that **Blocked, subtotal below** is selected.
9. Clear the option **Expand/collapse groups**.

   In this tutorial, the report you create does not use the drilldown feature that lets a user expand a parent group hierarchy to display child group rows and detail rows.
10. Click **Next**.
11. On the Choose a Style page, in the Styles pane, select a style.

    The illustration of the completed report shows the report using the Ocean style.
12. Click **Finish**.

    The table is added to the design surface. The table has five columns and five rows. The Row Groups pane shows three row groups: SalesDate, Subcategory, and Details. Detail data is all the data that is retrieved by the dataset query.
13. Click **Run** to preview the report.

For each product that is sold on a specific date, the table displays the product name, the quantity sold, and the sales total. The data is organized first by sales date and then by subcategory.

3. Use Background Colors to Display a KPI

Background colors can be set to an expression that is evaluated when you run the report.

To display the present state of a KPI by using background colors

1. In the table, right-click two cells down from the `[Sum(Sales)]` cell (the subtotal row that displays the sales for a subcategory), and then click **Text Box Properties**.
2. In **Fill**, click the **fx** button next to the **Fill color** option and enter the following expression in the **Set expression for: BackgroundColor** field:

```
=IIF(Sum(Fields!Sales.Value) >= 5000 ,"Lime", IIF(Sum(Fields!Sales.Value) < 2500, "Red","Yellow"))
```

This changes the background color to green, using the shade of green named "Lime", for each cell that contains an aggregated sum for `[Sum(Sales)]` that is greater than or equal to 5000. Values of `[Sum(Sales)]` between 2500 and 5000 are colored yellow. Values less than 2500 are colored red.

1. Click **OK**.
2. Click **Run** to preview the report.

In the subtotal row that displays the sales for a subcategory, the background color of the cell is red, yellow, or green depending on value of the sales sum.

4. Display a KPI by Using a Gauge

A gauge depicts a single value in a dataset. This tutorial uses a horizontal linear gauge because its shape and simplicity makes it easy to read, even in when it is a small size and used within a table cell. For more information, see Gauges (Report Builder and SSRS).

To display the present state of a KPI using a gauge

1. Switch to Design view.
2. In the table, right-click the column handler for the cell that you changed in the previous procedure, point to **Insert Column**, and then click **Right**. A new column is added to the table.
3. Type **KPI** in the column heading.

4. On the **Insert** tab, in the **Data Regions** group, click **Gauge**, and then click the design surface outside the table. The **Select Gauge Type** dialog box appears.
5. Click **Linear**. The first linear gauge type, **Horizontal**, is selected.
6. Click **OK**.

   A gauge is added to the design surface.
7. From the Report Data pane, drag Sales to the gauge. When you drag Sales across the gauge, the Gauge Data pane opens.
8. Drop Sales in the **Values** list.

   When you drop the field onto the gauge, the field is aggregated by using the built-in Sum function.
9. Right-click the pointer in the gauge and click **Pointer Properties**.
10. In **Pointer Type**, select **Bar**. This changes the pointer from a marker to a bar that will be more visible when the gauge is added to the table.
11. Click **Pointer Fill**. In **Secondary Color,** pick **Yellow**. The gradient fill pattern will change from white to yellow.
12. Right-click the scale in the gauge and click **Scale Properties**.
13. Set the **Maximum** option to 25000.

    > **NOTE**
    > Instead of a constant such as 25000, you can use an expression to dynamically calculate the value of the **Maximum** option. The expression would use the aggregate of aggregate feature and look similar to the expression `=Max(Sum(Fields!Sales.value), "Tablix1")`.
14. Drag the gauge inside the table into the third cell in the subtotal row that displays the sales for a subcategory of the column that you inserted.

    > **NOTE**
    > You might have to resize the column so that the horizontal linear gauge fits into the cell. To resize the column, click a column header and use the handles to resize the cells horizontally and vertically.
15. Click **Run** to preview the report.

    The horizontal length of the bar in the gauge changes depending on the value of the KPI.
16. (Optional) Add a maximum pin to handle overflow so that any value over the scale maximum always points to the maximum pin:

    a. Open the Properties pane.

    b. Click the scale. The properties for the linear scale are displayed in the Properties pane.

    c. In the **Scale Pins** category, expand the **MaximumPin** node.

    d. Set the **Enable** property to `True`. A pin appears after the maximum value on the scale.

    e. Set **BorderColor** to `Lime`.

17. Click **Run** to preview the report.

5. Display a KPI by Using an Indicator

Indicators are small simple gauges that communicate data values at a glance. Because of their size and simplicity, indicators are often used in tables and matrices. For more information, see Indicators (Report Builder and SSRS).

To display the present state of a KPI using an indicator

1. Switch to Design view.
2. In the table, right-click the column handler for the cell that you changed in the previous procedure, point to **Insert Column**, and then click **Right**. A new column is added to the table.
3. Type **KPI** in the column heading.
4. Click the cell for the subcategory subtotal.
5. On the **Insert** tab, in the **Data Regions** group, double-click **Indicator.**

   The **Select Indicator Type** dialog box opens.
6. Click **Shapes**. The first shape type, **3 Traffic Lights (Unrimmed),** is selected.

   You will use this indicator in the tutorial.
7. Click **OK**.

   The indicator is added to the design surface.
8. Right-click the indicator and click **Indicator Properties**.
9. Click **Values and States**.
10. In the Value drop-down list, select **[Sum(Sales)]**, but do not change any other options.

    By default, data synchronization occurs across the data region and you see the value **Tablix1**, the name of the table data region in the report, in the **Synchronization scope** box.

    In this report, you can also change the scope of an indicator placed in the cell of the subcategory subtotal to synchronize across the SalesDate field.
11. Click **Run** to preview the report.

6. Add a Report Title

A report title appears at the top of the report. You can place the report title in a report header or if the report does not use one, in a text box at the top of the report body. You will use the text box that is automatically placed at the top of the report body.

The text can be further enhanced by applying different font styles, sizes, and colors to phrases and individual characters of the text. For more information, see Format Text in a Text Box (Report Builder and SSRS).

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Product Sales KPI**, and then click outside the text box.
3. Optionally, right-click the text box that contains **Product Sales KPI**, click **Text Box Properties**, and then on the Font tab select the different font styles, sizes and colors.
4. Click **Run** to preview the report.

7. Save the Report

Save the report to a report server or your computer. If you do not save the report to the report server, a number of Reporting Services features such as report parts and subreports are not available.

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.
2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you see the contents of the report folder that the report server administrator specified as the default location for reports.
4. In **Name**, replace the default name with **Product Sales KPI**.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report on your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop**, **My Documents**, or **My computer**, and browse to the folder where you want to save the report.

> **NOTE**
>
> If you do not have access to a report server, click **Desktop**, **My Documents**, or **My computer** and save the report to your computer.

1. In **Name**, replace the default name with **Product Sales KPI**.
2. Click **Save**.

Next Steps

You have successfully completed the Adding a KPI to Your Report tutorial. For more information, see Gauges (Report Builder) Indicators (Report Builder and SSRS).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Map Report (Report Builder)

2/14/2019 • 21 minutes to read • Edit Online

This tutorial is designed to help you learn about the map features you can use to display report data against a geographic background.

Maps are based on spatial data that typically consists of points, lines, and polygons. For example, a polygon can represent the outline of a county, a line can represent a road, and a point can represent the location of a city. Each type of spatial data is displayed on a separate map layer as a set of map elements.

To vary the appearance of map elements, you specify a field that has values that match the map elements with analytical data from a dataset. You can also define rules that vary color, size, or other properties based on ranges of data.

In this tutorial, you will build a map report that displays store locations in New York state counties.

What You Will Learn

In this tutorial you will learn how to do the following:

1. Create a Map with a Polygon Layer from the Map Wizard
2. Add a Map Point Layer to Display Store Locations
3. Add a Map Line Layer to Display a Route
4. Add a Bing Maps Tile Background
5. Make a Layer Transparent
6. Vary County Color Based on Sales
    a. Build a Relationship between Spatial and Analytical Data
    b. Specify Color Rules for Polygons
    c. Format the Data in the Color Scale as Currency
    d. Create a New Legend
    e. Associate Legend and Color Rules
    f. Clear Color from Counties with No Data
7. Add a Custom Point
8. Center the Map View
9. Add a Report Title
10. Save the Report

> **NOTE**
> In this tutorial, the steps for the wizard are consolidated into two procedures: one to create the dataset and one to create a table. For step-by-step instructions about how to browse to a report server, choose a data source, create a dataset, and run the wizard, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

Estimated time to complete this tutorial: 30 minutes.

Requirements

For information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Map with a Polygon Layer from the Map Wizard

Add a map to your report from the map gallery. The map has one layer that displays the counties in New York state. The shape of each county is a polygon based on spatial data that is embedded in the map from the map gallery.

To add a map with the map wizard in a new report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2014Report Builder**, and then click **Report Builder**.

   The Getting Started dialog box appears.

   > **NOTE**
   > If the Getting Started dialog box does not appear, from the Report Builder button, click **New**.

2. In the left pane, verify that **Report** is selected.
3. In the right pane, click **Map Wizard**.
4. Click **Create**.
5. **Choose a source of spatial data** page, verify that **Map gallery** is selected.
6. In the Map Gallery pane, expand **States by County** under **USA**, and click **New York**.

   The Map Preview pane displays the New York county map.

7. Click **Next**.
8. On the **Choose spatial data and map view options** page, accept the defaults. By default, map elements from a map gallery will automatically be embedded in the report definition.
9. Click **Next**.
10. On the **Choose map visualization** page, verify **Basic Map** is selected, and click **Next**.
11. On the **Choose color theme and data visualization** page, select the **Display labels** option.
12. If it is selected, clear the **Single color map** option.
13. From the **Data field** drop-down list, click #COUNTYNAME. The Map Preview pane in the wizard displays the following items:

    - A title with the text **Map Title**.
    - A map that displays counties in New York where each county is a different color and the county name

appears wherever it fits over the county area.

- A legend that contains a title and a list of items 1 through 5.
- A color scale that contains values 0 to 160 and no color.
- A distance scale that displays kilometers (km) and miles (mi).

14. Click **Finish**.

    The map is added to the design surface.

15. Click the map to select it and display the **Map Layers Pane**. The **Map Layers Pane** shows one polygon layer of layer type **Embedded**. Each county is an embedded map element on this layer.

    > **NOTE**
    > If you do not see the **Map Layers** pane, it might be displayed outside your current view. Use the scroll bar at the bottom of the Design view window to change your view. Alternatively, in the **View** tab, clear the **Properties** or the **Report Data** option to provide more design surface area.

16. Right-click the map title, and then click **Title Properties**.
17. Replace the title text with **Sales by Store**.
18. Click **OK**.
19. Preview the report.

The rendered report displays the map title, the map, and the distance scale. The counties are on a map polygon layer. Each county is a polygon that varies by color from a color palette, but the colors are not associated with any data. The distance scale displays distances in both kilometers and miles.

The map legend and color scale do not yet appear because there is no analytical data associated with each county. You will add analytical data later in this tutorial.

2. Add a Map Point Layer to Display Store Locations

Use the map layer wizard to add a point layer that displays the locations of stores.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To add a point layer based on a SQL Server spatial query

1. Switch to Design view.
2. Double-click the map to display the **Map Layer** pane. On the toolbar, click the **New layer wizard** button .
3. On the **Choose a source of spatial data** page, select **SQL Server spatial query**, and click **Next**.
4. On the **Choose a dataset with SQL Server spatial data** page, click **Add a new dataset with SQL Server spatial data**, and then click **Next**.
5. On the **Choose a connection to a SQL Server spatial data source** page, select an existing data source or browse to the report server and select a data source.

6. Click **Next**.

7. On the Design a Query page, click **Edit as Text**.

8. Paste the following text in the query pane:

```
Select 114 as StoreKey, 'Contoso Albany Store' as StoreName, 1125 as SellingArea, 'Albany' as City,
'Albany' as County,
 CAST(1000000 as money) as Sales, CAST('POINT(-73.7472924218681 42.6564617079878)' as geography) AS
SpatialLocation
UNION ALL SELECT 115 AS StoreKey, 'Contoso New York No.1 Store' AS  StoreName, 500 as SellingArea, 'New
York' AS City, 'New York City' as County,
 CAST('2000000' as money) as Sales, CAST('POINT(-73.9922069374483 40.7549638237402)' as geography) AS
SpatialLocation
UNION ALL Select 116 as StoreKey, 'Contoso Rochester No.1 Store' as StoreName, 462 as SellingArea,
'Rochester' as City,  'Monroe' as County,
 CAST(3000000 as money) as Sales, CAST('POINT(-77.624041566786 43.1547066024338)' as geography)  AS
SpatialLocation
UNION ALL Select 117 as StoreKey, 'Contoso New York No.2 Store' as StoreName, 700 as SellingArea, 'New
York' as City,'New York City' as County,
  CAST(4000000 as money) as Sales, CAST('POINT(-73.9712488 40.7830603)' as geography) AS SpatialLocation
UNION ALL Select 118 as StoreKey, 'Contoso Syracuse Store' as StoreName, 680 as SellingArea, 'Syracuse'
as City, 'Onondaga' as County,
 CAST(5000000 as money) as Sales, CAST('POINT(-76.1349120532546 43.0610223535974)' as geography) AS
SpatialLocation
UNION ALL Select 120 as StoreKey, 'Contoso Plattsburgh Store' as StoreName, 560 as SellingArea,
'Plattsburgh' as City,  'Clinton' as County,
 CAST(6000000 as money) as Sales, CAST('POINT(-73.4728622833178 44.7028831413324)' as geography) AS
SpatialLocation
UNION ALL Select 121 as StoreKey, 'Contoso Brooklyn Store' as StoreName, 1125 as SellingArea, 'Brooklyn'
as City, 'New York City' as County,
 CAST(7000000 as money) as Sales, CAST('POINT (-73.9638533447143 40.6785123489351)' as geography) AS
SpatialLocation
UNION ALL Select 122 as StoreKey, 'Contoso Oswego Store' as StoreName, 500 as SellingArea, 'Oswego' as
City, 'Oswego' as County,
 CAST(8000000 as money) as Sales, CAST('POINT(-76.4602850815536 43.4353224527794)' as geography) AS
SpatialLocation
UNION ALL Select 123 as StoreKey, 'Contoso Ithaca Store' as StoreName, 460 as SellingArea, 'Ithaca' as
City, 'Tompkins' as County,
 CAST(9000000 as money) as Sales, CAST('POINT(-76.5001866085881 42.4310489934743)' as geography) AS
SpatialLocation
UNION ALL Select 124 as StoreKey, 'Contoso Rochester No.2 Store' as StoreName, 700 as SellingArea,
'Rochester' as City, 'Monroe' as County,
 CAST(100000 as money) as Sales, CAST('POINT(-77.6240415667866 43.1547066024338)' as geography) AS
SpatialLocation
UNION ALL Select 125 as StoreKey, 'Contoso Queens Store' as StoreName, 700 as SellingArea,'Queens' as
City, 'New York City' as County,
 CAST(500000 as money) as Sales, CAST('POINT(-73.7930979029883 40.7152781765927)' as geography) AS
SpatialLocation
UNION ALL Select 126 as StoreKey, 'Contoso Elmira Store' as StoreName, 680 as SellingArea, 'Elmira' as
City, 'Chemung' as County,
 CAST(800000 as money) as Sales, CAST('POINT(-76.7397414783301 42.0736492742663)' as geography) AS
SpatialLocation
UNION ALL Select 127 as StoreKey, 'Contoso Poestenkill Store' as StoreName, 455 as SellingArea,
'Poestenkill' as City, 'Rensselaer' as County,
CAST(1500000 as money) as Sales, CAST('POINT(-73.5626737425063 42.6940551238618)' as geography) AS
SpatialLocation
```

9. On the query designer toolbar, click **Run** (**!**).

   The result set displays seven columns: StoreKey, StoreName, SellingArea, City, County, Sales, and SpatialLocation. This data represents a set of stores in New York State that sell consumer goods. Each row in the result set contains a store identifier, store name, the area available for product display, the city and county where the store is located, the sales total, and the spatial location in longitude and latitude. The

display area ranges from 455 square feet to 1125 square feet.

10. Click **Next**.

    The report dataset named DataSet1 is created for you. After you complete the wizard, you can use the Report Data to its field collection.

11. On the **Choose a spatial data and map view options** page, verify that the **Spatial field** is `SpatialLocation` and that the **Layer type** is **Point**. Accept the other defaults on this page.

    The map view displays circles that mark the location of each store.

12. Click **Next**.

13. Specify a map type that displays markers that vary by analytic data. On the Choose map visualization page, click **Analytical Marker Map**, and then click **Next**.

14. On the **Choose the analytical dataset** page, click DataSet1. This dataset contains both analytical data and spatial data that will be displayed on the new point layer.

15. Click **Next**.

16. On the **Choose color theme and data visualization** page, clear the **Use marker colors to visualize data** option, and then select the option **Use marker types to visualize data**.

17. In **Data field**, select `[Sum(SellingArea)]` to vary marker types by the size of the area a store sets aside to display the products.

18. Click **Finish**.

    The map layer is added to the report. The legend displays marker types based on SellingArea values.

    Double-click the map to display the **Map Layer** pane. The **Map Layer** pane displays a new layer, PointLayer1, with spatial data source type **DataRegion**.

19. Add a legend title. Right-click the legend title, and then click **Legend Title Properties**.

20. Delete the title, and type **Display Area (Square Feet)**.

21. Click **OK**.

22. View the default values that were set by the wizard. In the **Map Layers Pane**, right-click the point layer, and then click **Marker Type Rule**.

    On the **General** tab, the markers are listed in the order in which they appear in the legend. On the **Distribution** tab, the number of subranges is 5. On the **Legend** tab, the legend text is set to display the start and end value in each range.

23. Click **OK**.

24. Preview the report.

The map displays the locations of stores in New York state. The marker type for each store is based on the display area. Five ranges of display area were automatically calculated for you.

3. Add a Map Line Layer to Display a Route

Use the map layer wizard to add a map layer that displays a route between two stores. In this tutorial, the path is created from three store locations. In a business application, the path might be the best route between stores.

To add a line layer to map

1. Switch to Design view.

2. Double-click the map to display the **Map Layer** pane. On the toolbar, click **New layer wizard**.
3. On the **Choose a source of spatial data** page, select **SQL Server spatial query** and click **Next**.
4. On the **Choose a dataset with SQL Server spatial data** page, click **Add a new dataset with SQL Server spatial data** and click **Next**.
5. On the **Choose a connection to a SQL Server spatial data source**, select DataSource1, the data source that you created in the first procedure.
6. Click **Next**.
7. On the **Design a Query** page, click **Edit as Text**. The query designer switches to text-based mode.
8. Paste the following text in the query pane:

```
SELECT N'Path' AS Name, CAST('LINESTRING(
   -76.5001866085881 42.4310489934743,
   -76.4602850815536 43.4353224527794,
   -73.4728622833178 44.7028831413324)' AS geography) as Route
```

9. Click **Next**.

   A path appears on the map that connects three stores.

10. On the **Choose spatial data and map view options** page, verify that the **Spatial field** is **Route** and that the **Layer type** is **Line**. Accept the other defaults.

    The map view displays a path from a store in the northern part of New York state to a store in the southern part of New York state.

11. Click **Next**.
12. On the **Choose map visualization** page, click **Basic Line Map**, and then click **Next**.
13. On the **Choose color theme and data visualization**, select the option **Single color map**. The path appears as a single color based on the selected theme.
14. Click **Finish**.

The map displays a new line layer with spatial data source type **DataSet**. In this example, the spatial data comes from a dataset but no analytical data is associated with the line.

4. Add a Bing Maps Tile Background

Add a map layer that displays a Bing Maps tile background.

To add a Virtual Earth tile background

1. Switch to Design view.
2. Double-click the map to display the **Map Layer** pane. On the toolbar, click **Add Layer**.
3. From the drop-down list, click **Tile Layer**.

   The last layer in the **Map Layer** pane is TileLayer1. By default, the tile layer displays the road map style.

> **NOTE**
> In the wizard, you can also add a tile layer on the **Choose spatial data and map view options** page. To do this, select **Add a Bing Maps background for this map view**. In a rendered report, the tile background displays Bing Maps tiles for the current map viewport center and zoom level.

4. Click the down arrow on TileLayer1, and then click **Tile Properties**.
5. In **Type**, select **Aerial**. The aerial view does not contain text.
6. Click **OK**.

5. Make a Layer Transparent

To let the items on one layer show through another layer, you can adjust the order of layers and the transparency of each layer to get the effect that you want.

To set the transparency of a layer

1. Switch to Design view.
2. Double-click the map to display the **Map Layer** pane.
3. Click the down arrow on PolygonLayer1, and then click **Layer Data**. The **Map Polygon Layer Properties** dialog box opens.
4. Click **Visibility**.
5. In **Transparency (%)**, type **30**.
6. Click **OK**.

The design surface displays the counties as semi-transparent.

6. Vary County Color Based on Sales

Each county on the polygon layer has a different color because the report processor automatically assigns a color value from the color palette based on the theme that you chose on the last page of the map wizard.

In the following steps, specify a color rule to associate specific colors with a range of store sales for each county. The colors red-yellow-green indicate relative high-middle-low sales. Format the color scale to show currency. Display the annual sales ranges in a new legend. For counties that do not contain stores, use no color to show that there is no associated data.

6a. Build a Relationship between Spatial and Analytical Data

To vary the county shapes by color based on analytical data, you first must associate the analytical data with the spatial data. In this tutorial, you will use the county name to match on.

To build a relationship between spatial data and analytical data

1. Switch to Design view.
2. Double-click the map to display the **Map Layer** pane.
3. Click the down arrow on PolygonLayer1, and then click **Layer Data**. The **Map Polygon Layer Properties** dialog box opens.
4. Click **Analytical data**.
5. From the drop-down list, select DataSet1. This dataset was created by the wizard when you specified the spatial data query for the counties.

6. In **Fields to match on**, click **Add**. A new row is added.
7. In **From spatial dataset**, from the drop-down list, click COUNTYNAME.
8. In **From analytical dataset**, from the drop-down list, click [County].
9. Click **OK**.
10. Preview the report.

By specifying a match field from the spatial data source and from the analytical dataset, you enable the report processor to group analytical data based on the map elements. A data-bound map element has a successful match for the values that you specified.

Each county that contains a store has a color that is based on the color palette for the style that you chose in the wizard.

6b. Specify Color Rules for Polygons

To create a rule that varies the color of each county based store sales, you must specify the range values, the number of divisions within that range that you want to display, and the colors to use.

To specify color rules for all polygons that have associated data

1. Switch to Design view.
2. Click the down arrow on PolygonLayer1, and then click **Polygon Color Rule**. The **Map Color Rules Properties** dialog box opens. Notice that the color rule option **Visualize data by using color palette** is selected. This option was set by the wizard.
3. Select **Visualize data by using color ranges**. The palette option is replaced by start color, middle color, and end color options.
4. Define range values for sales per county. In **Data field**, from the drop-down list, select `[Sum(Sales)]`.
5. To change the format to display currency in thousands, change the expression to the following: `=Sum(Fields!Sales.Value)/1000`
6. Change **Start color** to **Red**.
7. Change **End color** to **Green**.

   Red represents low sales values, **Yellow** represents middle sales values, and **Green** represents high sales values. The report processor calculates a range of colors based on these values and the options that you choose on the **Distribution** page.
8. Click **Distribution**.
9. Verify that the distribution type is **Optimal**. For the expression from step 5, optimal distribution divides the values into subranges that balance the number of items in each range and the span for each range.
10. Accept the default values for other options on this page. When you select the optimal distribution type, the number of subranges are calculated when the report runs.
11. Click **Legend**.
12. In **Color scale options**, verify that **Show in color scale** is selected.
13. In **Show in this legend**, from the drop-down list, select the blank line. For now, you will show the color ranges only in the color scale.
14. Click **OK**.

The color scale displays five colors: red, orange, yellow, yellow green, and green. Each color represents a sales

range that is automatically calculated based on the sales by county.

6c. Format the Data in the Color Scale as Currency

By default, data has a general format. You can apply custom formats.

To set the format for the color scale

1. Right-click the color scale, and then click **Color Scale Properties**.
2. Click **Number**.
3. In **Category**, click **Currency**.
4. In **Decimal places**, type **0**. This format specifies no decimal places for currency.
5. Click **OK**.
6. Preview the report.

The color scale displays annual sales in currency format for each range.

6d. Create a New Legend

By default, all rules display in the first legend. To improve the display for a map, you can add legends.

To change the default display, there are two steps: create a new legend and then associate the rule results for a map layer with the new legend.

To create a new legend

1. Switch to Design view.
2. Right-click the map outside the viewport, and then click **Add Legend**. A new legend is added to the map at a default location.
3. Right-click the legend, and then click **Legend Properties**.
4. In **Position options**, click the location that specifies where you want the legend to appear relative to the viewport. The map on the design surface changes to show the effect of your selections.
5. Click **OK**.
6. Click **Title** on the legend to select the legend title.
7. Click **Title** again to enter insert mode for the text. Replace **Title** by **Sales (Thousands)**, and then click outside the text.

The legend expands to display the title.

6e. Associate Legend and Color Rules

Each legend can display one or more sets of rule results.

To associate a legend with color rules

1. Double-click the map to display the **Map Layer** pane.
2. Click the down arrow on PolygonLayer1, and then click **Polygon Color Rule**. The **Map Color Rules Properties** dialog box opens.
3. Click **Legend**.
4. In **Color scale options**, clear **Show in color scale**.
5. In **Legend options**, from the drop-down list, select Legend2. The legend text option appears. By default, legend text is formatted with a general .NET Framework format string. The 0 in N0 specifies no decimal digits.

6. In **Legend text**, use the following format to specify currency with no decimal digits:
   `#FROMVALUE {C0} - #TOVALUE {C0}`

7. Click **OK**.

   On the design surface, the legend displays the color ranges with sample data formatted as currency.

8. Preview the report.

The counties that have associated stores and sales display according to the color rules. Counties that have no sales have no color.

6f. Change Color for Counties with No Data

You can set the default display options for all map elements on a layer. Color rules take precedence over these display options.

To set the display properties for all elements on a layer

1. Switch to Design view.

2. Double-click the map to display the **Map Layer** pane.

3. Click the down arrow on PolygonLayer1, and then click **Polygon Properties**. The **Map Polygon Properties** dialog box opens. Display options set in this dialog box apply to all polygons on the layer before rule-based display options are applied.

4. Click **Fill**.

5. Verify that the fill style is **Solid.** Gradients and patterns apply to all colors.

6. In **Color**, click the down arrow, and then click **Light Steel Blue**.

7. Click **OK**.

8. Preview the report.

Counties that have no associated data display as blue. Only counties that have associated analytical data appear in the **Red** through **Green** colors from the color rules that you specified.

7. Add a Custom Point

To represent a new store that has not yet been built, specify a point and use the **PushPin** marker type.

To add a custom point

1. Switch to Design view.

2. Double-click the map to display the **Map Layer** pane. On the toolbar, click **Add Layer**, and then click **Point Layer**.

   A new point layer is added to the map. By default, the point layer has spatial data type **Embedded**.

3. Click the down arrow on PointLayer2, and then click **Add Point**.

4. Move the pointer over the map viewport. The cursor changes to crosshairs.

5. Click the location on the map where you want to add a point. In this tutorial, click a location in a county next to the start of the route. A point marked by a circle is added to the layer at the location where you clicked. By default, the point is selected.

6. Right-click the point you added, and then click **Embedded Point Properties**.

7. Select the option **Override point options for this layer**. Additional pages appear in the dialog box. Values that you set here take precedence over display options for the layer or for color rules.

8. Click **Marker**.
9. For **Marker type**, select **Star**.
10. Click **OK**.
11. Preview the report.

The new point you added appears as a **Star**.

To add a label for the custom point

1. Switch to Design view.
2. Right-click the point you just added, and then click **Embedded Point Properties**.
3. Click **Labels**.
4. In **Label text**, type **New Store**.
5. In **Placement**, click **Top**.
6. Click **OK**.
7. Preview the report.

The label appears above the store location.

Center the Map View

Change the map viewport center and zoom level.

To change the viewport

1. Right-click the map viewport, and then click **Viewport Properties**.
2. Click **Center and Zoom**.
3. Verify that the option **Set a view center and zoom level** is selected.
4. Click **OK**.
5. Left-click the map viewport, and drag the center of the viewport to the location that you want.
6. Use the mouse wheel to change the zoom level of the viewport.
7. Preview the report.

In Design view, the map on the display surface and the view is based on sample data. In the rendered report, the map view is centered on the view that you specified.

Add a Report Title

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Sales in New York Stores** and then click outside the text box.

This title will appear at the top of the report. Items at the top of the report body when there is no page header defined are the equivalent of a report header.

Save the Report

To save the report

1. Switch to Design view.
2. From the Report Builder button, click **Save As**.
3. In **Name**, type **Store Sales in New York**.

Click **Save**.

Next Steps

This concludes the walkthrough for how to add a map to your report.

For more information, see Maps (Report Builder and SSRS) and the blog entry Cartographic Adjustment of Spatial Data for SQL Server Reporting Services on blogs.msdn.com.

For more tutorials, see Tutorials (Report Builder).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014
Map Wizard and Map Layer Wizard (Report Builder and SSRS)
Vary Polygon, Line, and Point Display by Rules and Analytical Data (Report Builder and SSRS)

Tutorial: Add a Parameter to Your Report (Report Builder)

2/14/2019 • 16 minutes to read • Edit Online

Add a parameter to your report to let users filter report data from the data source or in the report. Report parameters are created automatically for each query parameter that you include in a dataset query. The parameter data type determines how it appears on the report view toolbar.

What You Will Learn

In this tutorial you will learn how to do the following:

1. Create a Matrix Report and Dataset from the Table or Matrix Wizard
2. Organize Data, Choose Layout, and Style from the Table or Matrix Wizard
3. Add a Query Parameter to Create a Report Parameter
4. Change Default Data Type and Other Properties for a Report Parameter
   a. Add a Dataset to Provide Available Values and Display Names
   b. Specify Available Values to Create a Drop-List of Values
   c. Specify Default Values so the Report Runs Automatically
   d. Look up a Value from a Dataset that has Name/Value Pairs
5. Display the Selected Parameter Value in the Report
6. Use the Report Parameter in a Filter
7. Change the Report Parameter to Accept Multiple Values
8. Add a Boolean Parameter for Conditional Visibility
9. Add a Report Title
10. Save the Report

> **NOTE**
> In this tutorial, the steps for the wizard are consolidated into one procedure. For step-by-step instructions about how to browse to a report server, choose a data source, and create a dataset, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

Estimated time to complete this tutorial: 25 minutes.

Requirements

For information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Matrix Report and Dataset from the Table or Matrix Wizard

Create a matrix report, a data source, and a dataset.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

To create a new matrix report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2014Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box appears.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, from the **Report Builder** button, click **New**.

2. In the left pane, verify that **Report** is selected.
3. In the right pane, click **Table or Matrix Wizard**.
4. Click **Create**.
5. On the **Choose a dataset** page, click **Create a dataset**.
6. Click **Next**.
7. On the **Choose a connection to a data source** page, select a data source that is type **SQL Server**. Select a data source from the list or browse to the report server to select one.
8. Click **Next**.
9. On the **Design a query** page, click **Edit as Text**.
10. Paste the following query into the query pane:

```
;WITH CTE (StoreID, Subcategory, Quantity)
AS (
SELECT 200 AS StoreID, 'Digital SLR Cameras' AS Subcategory, 2002 AS Quantity
UNION SELECT  200 AS StoreID, 'Camcorders' AS Subcategory, 1954 AS Quantity
UNION SELECT  200 AS StoreID, 'Accessories' AS Subcategory, 1895 AS Quantity
UNION SELECT  199 AS StoreID, 'Digital Cameras' AS Subcategory, 1849 AS Quantity
UNION SELECT  306 AS StoreID, 'Digital SLR Cameras' AS Subcategory, 1579 AS Quantity
UNION SELECT  306 AS StoreID, 'Camcorders' AS Subcategory, 1561 AS Quantity
UNION SELECT  306 AS StoreID, 'Digital Cameras' AS Subcategory, 1553 AS Quantity
UNION SELECT  306 AS StoreID, 'Accessories' AS Subcategory, 1534 AS Quantity
UNION SELECT 307 AS StoreID, 'Accessories' AS Subcategory, 1755 AS Quantity
UNION SELECT 307 AS StoreID, 'Camcorders' AS Subcategory, 1631 AS Quantity
UNION SELECT 307 AS StoreID, 'Digital SLR Cameras' AS Subcategory, 1772 AS Quantity)
SELECT StoreID, Subcategory, Quantity
FROM CTE
```

This query combines the results of several Transact-SQL SELECT statements inside a common table expression to specify values that are based on simplified data from the Contoso sample database. The Contoso sales data represents international sales data for consumer goods. This tutorial uses sales data for cameras. The subcategories represent digital cameras, digital single lens reflex (SLR) cameras, camcorders, and accessories.

The query specifies column names that include a store identifier, a sales item subcategory, and the quantity ordered for sales orders from three stores. In this query, the store name is not part of the result set. Later in this tutorial, you will look up the name of the store that corresponds to the store identifier from a separate dataset.

This query does not contain query parameters. You will add query parameters later in this tutorial.

11. On the query designer toolbar, click **Run** (**!**). The result set displays 11 rows of data that show the quantity of items sold for each subcategory for four stores, and includes the following columns: StoreID, Subcategory, Quantity.
12. Click **Next**.

2. Organize Data, Choose Layout, and Style from the Table or Matrix Wizard

Use the wizard to provide a starting design on which to display data. The preview pane in the wizard helps you to visualize the result of grouping data before you complete the table or matrix design.

To organize data into groups

1. On the **Arrange fields** page, drag Subcategory to **Row groups**.
2. Drag StoreID to **Column groups**.
3. Drag Quantity to **Values**.

   You have organized the quantity sold values in rows grouped by subcategory. There will be one column for each store.
4. Click **Next**.
5. On the **Choose a Layout** page, under **Options**, verify that **Show subtotals and grand totals** is selected.

   When you run the report, the last column will show the total quantity of each subcategory for all stores, and the last row will show the total quantity for all subcategories for each store.
6. Click **Next**.

7. On the **Choose a Style** page, in the Styles pane, select a style.
8. Click **Finish**.

   The matrix is added to the design surface. The matrix displays three columns and three rows. The contents of the cells in the first row are Subcategory, [StoreID], and Total. The contents of the cells in the second row contain expressions that represent the subcategory, the quantity of items sold for each store, and the total quantity for each subcategory for all stores. The cells in the final row display the grand total for each store.
9. Click in the matrix, hover over the edge of the first column, grab the handle, and expand the column width.
10. Click **Run** to preview the report.

The report runs on the report server and displays the title and the time the report processing occurred.

In this scenario, the column headings display the store identifier but not the store name. Later, you will add an expression to look up the store name in a dataset that contains store identifier/store name pairs.

3. Add a Query Parameter to Create a Report Parameter

When you add a query parameter to a query, Report Builder automatically creates a single-valued report parameter with default properties for name, prompt, and data type.

To add a query parameter

1. Switch to Design view.
2. In the Report Data pane, expand the **Datasets** folder, right-click **DataSet1**, and then click **Query**.
3. Add the following Transact-SQL `WHERE` clause as the last line in the query:

   ```
   WHERE StoreID = (@StoreID)
   ```

   The `WHERE` clause limits the retrieved data to the store identifier that is specified by the query parameter *@StoreID*.
4. On the query designer toolbar, click **Run** (**!**). The **Define Query Parameters** dialog box opens and prompts for a value for the query parameter *@StoreID*.
5. In **Parameter Value**, type **200**.
6. Click **OK**.

   The result set displays the quantities sold for Accessories, Camcorders, and Digital SLR Cameras for the store identifier **200**.
7. Click **OK**.
8. In the Report Data pane, expand the **Parameters** folder.

Notice that there is now a report parameter named *@StoreID*. By default, the parameter has data type **Text**. Because the store identifier is an integer, you will change the data type to Integer in the next procedure.

4. Change Default Data Type and Other Properties for a Report Parameter

After a report parameter is created, you can adjust the default values for properties.

To change the default data type for a report parameter

1. In the Report Data pane under the **Parameters** node, right-click *@StoreID*, and then click **Parameter**

Properties.

2. In Prompt, type **Store identifier?** This text appears on the report viewer toolbar when you run the report.
3. In **Data type**, from the drop-down list, select **Integer**.
4. Accept the remaining default values in the dialog box.
5. Click **OK**.
6. Preview the report. The report viewer displays the prompt for *@StoreID*.
7. On the report viewer toolbar, next to Store ID, type **200**, and then click **View Report**.

4a. Add a Dataset to Provide Available Values and Display Names

To ensure a user can type only valid values for a parameter, you can create a drop-down list of values to choose from. The values can come from a dataset or from a list that you specify. Available values must be supplied from a dataset that has a query that does not contain a reference to the parameter.

To create a dataset for valid values for a parameter

1. Switch to Design view.
2. In the Report Data pane, right-click the **Datasets** folder, and then click **Add Dataset**.
3. In **Name**, type **Stores**.
4. Select the **Use a dataset embedded in my report** option.
5. In **Data source**, from the drop-down list, choose the data source you created in the first procedure.
6. In **Query type**, verify that **Text** is selected.
7. In **Query**, paste the following text:

```
SELECT 200 AS StoreID, 'Contoso Catalog Store' as StoreName
UNION SELECT 199 AS StoreID, 'Contoso North America Online Store' as StoreName
UNION SELECT 307 AS StoreID, 'Contoso Asia Online Store' as StoreName
UNION SELECT 306 AS StoreID, 'Contoso Europe Online Store' as StoreName
```

8. Click **OK**.

   The Report Data pane displays the fields StoreID and StoreName under the **Stores** dataset node.

4b. Specify Available Values to Create a Drop-down List of Values

After you create a dataset to provide available values, you must change the report properties to specify which dataset and which field to use to populate the drop-down list of valid values on the reportviewer toolbar.

To provide available values for a parameter from a dataset

1. In the Report Data pane, right-click the parameter *@StoreID*, and then click **Parameter Properties**.
2. Click **Available Values**, and then click **Get values from a query**.
3. In **Dataset**, from the drop-down list, click **Stores**.
4. In **Value field**, from the drop-down list, click StoreID.
5. In **Label field**, from the drop-down list, click StoreName. The label field specifies the display name for the value.
6. Click **General**.

7. In Prompt, type **Store name?**

   The user will now select from a list of store names instead of store identifiers. Note that the parameter data type remains **Integer** because the parameter is based on the store identifier, not the store name.

8. Click **OK**.

9. Preview the report.

   In the report viewer toolbar, the parameter text box is now a drop-down list that displays **<Select a Value>**.

10. From the drop-down list, select Contoso Catalog Store, and then click **View Report**.

The report displays the quantity sold for Accessories, Camcorders, and Digital SLR Cameras for the store identifier **200**.

4c. Specify Default Values so the Report Runs Automatically

You can specify a default value for each parameter so the report runs automatically.

To specify a default value from a dataset

1. Switch to Design view.

2. In the Report Data pane, right-click *@StoreID*, and then click **Parameter Properties**.

3. Click **Default Values**, and then click **Get values from a query**.

4. In **Dataset**, from the drop-down list, click **Stores**.

5. In **Value field**, from the drop-down list, click StoreID.

6. Click **OK**.

7. Preview the report.

For *@StoreID*, the report viewer displays the value "Contoso North America Online Store". This is the first value from the result set for the dataset **Stores**. The report displays the quantity sold for Digital Cameras for store identifier **199**.

To specify a custom default value

1. Switch to Design view.

2. In the Report Data pane, right-click *@StoreID*, and then click **Parameter Properties**.

3. Click **Default Values**, and click **Specify values**, and then click **Add**. A new value row is added.

4. In **Value**, type **200**.

5. Click **OK**.

6. Preview the report.

For *@StoreID*, the report viewer displays the value "Contoso Catalog Store". This is the display name for store identifier **200**. The report displays the quantity sold for Accessories, Camcorders, and Digital SLR Cameras for the store identifier **200**.

4d. Look up a Value from a Dataset that has Name/Value Pairs

A dataset might contain both the identifier and the corresponding name field. When you only have an identifier, you can look up the corresponding name in a dataset that you created that includes name/value pairs.

To look up a value from a dataset

1. Switch to Design view.
2. On the design surface, in the matrix, in the first row column header, right-click `[StoreID]` and then click **Expression**.
3. In the expression pane, delete all text except the beginning `equals` (=).
4. In **Category**, expand **Common Functions**, and click **Miscellaneous**. The Item pane displays a set of functions.
5. In Item, double-click **Lookup**. The expression pane displays `=Lookup(`. The Example pane displays an example of Lookup syntax.
6. Type the following expression:
   `=Lookup(Fields!StoreID.Value,Fields!StoreID.Value,Fields!StoreName.Value,"Stores")`

   The Lookup function takes the value for StoreID, looks it up in the "Stores" dataset, and returns the StoreName value.
7. Click **OK**.

   The store column header contains the display text for a complex expression: **<<Expr>>**.
8. Preview the report.

The text box at the top of each page displays the store name instead of the store identifier.

5. Display the Selected Parameter Value in the Report

When a user has questions about a report, it helps to know which parameter values they chose. You can preserve user-selected values for each parameter in the report. One way is to display the parameters in a text box in the page footer.

To display the selected parameter value and label on a page footer

1. Switch to Design view.
2. Right-click the page footer, point to **Insert**, and then click **Text Box**. Drag the text box next to the text box with the time stamp. Grab the side handle of the text box and expand the width.
3. From the Report Data pane, drag the parameter *@StoreID* to the text box. The text box displays `[@StoreID]`.
4. To display the parameter label, click in the text box until the insert cursor appears after the existing expression, type a space, and then drag another copy of the parameter from the Report Data pane to the text box. The text box displays `[@StoreID] [@StoreID]`.
5. Right-click the first expression and click **Expression**. The **Expression** dialog box opens. Replace the text `Value` by `Label`.
6. Click **OK**.

   The text displays: `[@StoreID.Label] [@StoreID]`.
7. Preview the report.

6. Use the Report Parameter in a Filter

Filters help control which data to use in a report after it is retrieved from an external data source. To let a user help control the data they want to see, you can include the report parameter in a filter for the matrix.

To specify a parameter in a matrix filter

1. Switch to Design view.
2. Right-click a row or column header handle on the matrix, and then click **Tablix Properties**.
3. Click **Filters**, and then click **Add**. A new filter row appears.
4. In **Expression**, from the drop-down list, select the dataset field StoreID. The data type displays **Integer**. When the expression value is a dataset field, the data type is set automatically.
5. In **Operator**, verify that `equals` (=) is selected.
6. In **Value**, type `[@StoreID]`. `[@StoreID]` is the simple expression syntax that represents `=Parameters!StoreID.Value`.
7. Click **OK**.
8. Preview the report.

   The matrix displays data only for "Contoso Catalog Store".
9. On the report viewer toolbar, for **Store name?**, select **Contoso Asia Online Store**, and then click **View Report**.

The matrix displays data that corresponds to the store that you selected.

7. Change the Report Parameter to Accept Multiple Values

To change a parameter from single to multivalued, you must change the query, and all expressions that contain a reference to the parameter, including filters. A multivalued parameter is an array of values. In a dataset query, query syntax must test for inclusion of one value in a set of values. In a report expression, expression syntax must access an array of values instead of an individual value.

To change a parameter from single to multivalued

1. Switch to Design view.
2. In the Report Data pane, right-click *@StoreID*, and then click **Parameter Properties**.
3. Select **Allow multiple values**.
4. Click **OK**.
5. In the Report Data pane, expand the **Datasets** folder, right-click **DataSet1**, and then click **Query**.
6. Change `equals` (=) to `IN` in the Transact-SQL `WHERE` clause in the last line in the query:

   ```
   WHERE StoreID IN (@StoreID)
   ```

   The `IN` operator tests a value for inclusion in a set of values.
7. Click **OK**.
8. Right-click a row or column header handle on the matrix, and then click **Tablix Properties**.
9. Click **Filters**.
10. In **Operator**, select **In**.
11. Click **OK**.
12. In the text box that displays the parameter in the page footer, delete all text.
13. Right-click the text box, and then click **Expression**. Type the following expression:

```
=Join(Parameters!StoreID.Label, ", ")
```

This expression concatenates all store names that the user selected.

14. Click **OK**.
15. Click in the text box in front of the expression that you just created, and then type the following: Parameter Values Selected:.
16. Preview the report.
17. Click the drop-down list next to Store Name?

    Each valid value appears next to a check box.

18. Click **Select All**, and then click **View Report**.

    The report displays the quantity sold for all subcategories for all stores.

19. From the drop-down list, click **Select All** to clear the list, click "Contoso Catalog Store" and "Contoso Asia Online Store", and then click **View Report**.

8. Add a Boolean Parameter for Conditional Visibility

To add a boolean parameter

1. On the design surface, in the Report Data pane, right-click **Parameters**, and click **Add Parameter**.
2. In **Name**, type ShowSelections.
3. In **Prompt**, type Show selections?
4. In **Data type**, from the drop-down list, click **Boolean**.
5. Click **Default Values**.
6. Click **Specify value**, and then click **Add**.
7. In **Value**, type **False**.
8. Click **OK**.

To set visibility based on a boolean parameter

1. On the design surface, right-click the text box in the page footer that displays the parameter values, and then click **Text Box Properties**.
2. Click **Visibility**.
3. Select the option **Show or hide based on an expression**, and then click the expression button **Fx**.
4. Type the following expression: `=Not Parameters!ShowSelections.Value`

   The text box Visibility option is controlled by the property Hidden. Apply the `Not` operator so that when the parameter is selected, the Hidden property is false, and the text box will be displayed.

5. Click **OK**.
6. Click **OK**.
7. Preview the report.

   The text box that displays the parameter choices does not appear.

8. In the report viewer toolbar, next to **Show selections**, click `True`.

9. Preview the report.

The text box in the page footer displays all the store names that you selected.

9. Add a Report Title

To add a report title

1. On the design surface, click **Click to add title**.

2. Type Parameterized Product Sales, and then click outside the text box.

10. Save the Report

To save the report on a report server

1. From the **Report Builder** button, click **Save As**.

2. Click **Recent Sites and Servers**.

3. Select or type the name of the report server where you have permission to save reports.

   The message **Connecting to report server appears**. When the connection is complete, you see the contents of the report folder that the report server administrator specified as the default location for reports.

4. In **Name**, replace the default name with Parameterized Sales Report.

5. Click **Save**.

The report is saved to the report server. The report server that you are connected to appears in the status bar at the bottom of the window.

Next Steps

This concludes the walkthrough for how to add a parameter to your report. To learn more about parameters, see Report Parameters (Report Builder and Report Designer).

See Also

Tutorials (Report Builder)
Report Builder in SQL Server 2014

Tutorial: Creating Drillthrough and Main Reports (Report Builder)

2/14/2019 • 21 minutes to read • Edit Online

This tutorial teaches you how to create two kinds of reports: a drillthrough report and a main report. The sample sales data used in these reports is retrieved from an Analysis Services cube. The following illustration shows the reports you will create.

The following illustration shows how the field value, Games and Toys, in the main report displays in the drillthrough report's title. The data in the drillthrough pertains to the Games and Toys product category.

What You Will Learn

In the drillthrough report you will learn how to:

1. Create a Drillthrough Matrix Report and Dataset from the Table or Matrix Wizard
    a. Specify a Data Connection
    b. Create an MDX Query
    c. Organize Data into Groups Style
    d. Add Subtotals and Totals
    e. Choose a Style
2. Format Data as Currency
3. Add columns to Show Sales Values in Sparklines
4. Add Report Title with Product Category Name
5. Update Parameter Properties
6. Save the Report to a SharePoint Library

In the main report you will learn how to:

1. Create the Main Matrix Report and Dataset from the Table or Matrix Wizard
    a. Specify a Data Connection
    b. Create an MDX Query
    c. Organize Data into Groups
    d. Add Subtotals and Totals
    e. Choose a Style
2. Remove the Grand Total Row
3. Configure Text Box Action for Drillthrough
4. Replace Numeric Values with Indicators
5. Update Parameter Properties
6. Add a Report Title
7. Save the Report to a SharePoint Library
8. Run the Main and Drillthrough Reports

Estimated time to complete this tutorial: 30 minutes.

Requirements

This tutorial requires access to the Contoso Sales cube. This requirement applies to both the drillthrough and the main reports. For more information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Drillthrough Report from the Table or Matrix Wizard

From the Getting Started dialog box, create a matrix report by using the **Table or Matrix Wizard**. There are two

modes available in the wizard: report design and shared dataset design. In this tutorial, you will use the report design mode.

To create a new report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2014 Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box opens. If it does not appear, from the **Report Builder** button, click **New**.

2. In the left pane, verify that **New Report** is selected.

3. In the right pane, verify that **Table or Matrix Wizard** is selected.

1a. Specify a Data Connection

A data connection contains the information necessary to connect to an external data source such as an Analysis Services cube or a SQL Server database. To specify a data connection, you can use a shared data source from the report server or create an embedded data source that is used only in this report. In this tutorial, you will use an embedded data source. To learn more about using a shared data source, see Alternative Ways to Get a Data Connection (Report Builder).

To create an embedded data source

1. On the **Choose a dataset** page, select **Create a dataset**, and then click **Next**. The **Choose a connection to a data source** page opens.

2. Click **New**. The **Data Source Properties** dialog box opens.

3. In **Name**, type **Online and Reseller Sales Detail** as the name for the data source.

4. In **Select a connection type**, select **Microsoft SQL Server Analysis Services**, and then click **Build**.

5. In **Data source**, verify that the data source is **Microsoft SQL Server Analysis Services (AdomdClient)**.

6. In **Server name**, type the name of a server where an instance of Analysis Services is installed.

7. In **Select or enter a database name**, select the Contoso cube.

8. Click **OK**.

9. Verify that **Connection string** contains the following syntax:

   ```
   Data Source=<servername>; Initial Catalog = Contoso
   ```

   The `<servername>` is the name of an instance of SQL Server with Analysis Services installed.

10. Click **Credentials type**.

    > **NOTE**
    > Depending on how permissions are configured on the data source, you might need to change the default authentication options. For more information, see Security (Report Builder).

11. Click **OK**.

    The **Choose a connection to a data source** page appears.

12. To verify that you can connect to the data source, click **Test Connection**.

    The message **Connection created successfully** appears.

13. Click **OK**.

14. Click **Next**.

1b. Create an MDX Query

In a report, you can use a shared dataset that has a predefined query, or you can create an embedded dataset for use only in your report. In this tutorial, you will create an embedded dataset.

To create query filters

1. On the **Design a query** page, in the Metadata pane, click the button **(...)**.

2. In the **Cube Selection** dialog box, click Sales, and then click **OK**.

> **TIP**
> If you do not want to build the MDX query manually, click the icon, toggle the query designer to Query mode, paste the completed MDX to the query designer, and then proceed to step 6 in To create the dataset.

```
SELECT NON EMPTY { [Measures].[Sales Amount], [Measures].[Sales Return Amount] } ON COLUMNS, NON EMPTY
{ ([Channel].[Channel Name].[Channel Name].ALLMEMBERS * [Product].[Product Category Name].[Product
Category Name].ALLMEMBERS * [Product].[Product Subcategory Name].[Product Subcategory Name].ALLMEMBERS
) } DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS FROM ( SELECT ( { [Date].[Calendar
Year].&[2009] } ) ON COLUMNS FROM ( SELECT ( { [Sales Territory].[Sales Territory Group].&[North
America] } ) ON COLUMNS FROM ( SELECT ( STRTOSET(@ProductProductCategoryName, CONSTRAINED) ) ON COLUMNS
FROM ( SELECT ( { [Channel].[Channel Name].&[2], [Channel].[Channel Name].&[4] } ) ON COLUMNS FROM
[Sales]))) WHERE ( [Sales Territory].[Sales Territory Group].&[North America], [Date].[Calendar
Year].&[2009] ) CELL PROPERTIES VALUE, BACK_COLOR, FORE_COLOR, FORMATTED_VALUE, FORMAT_STRING,
FONT_NAME, FONT_SIZE, FONT_FLAGS
```

3. In the Measure Group pane, expand Channel, and then drag Channel Name to the **Hierarchy** column in the filter pane.

   The dimension name, Channel, is automatically added to the **Dimension** column. Do not change the **Dimension** or **Operator** columns.

4. To open the **Filter Expression** list, click the down arrow in the **Filter Expression** column.

5. In the filter expression list, expand **All Channel**, click **Online**, click **Reseller**, and then click **OK**.

   The query now includes a filter to include only these channels: Online and Reseller.

6. Expand the Sales Territory dimension, and then drag Sales Territory Group to the **Hierarchy** column (below **Channel Name**).

7. Open the **Filter Expression** list, expand **All Sales Territory**, click **North America**, and then click **OK**.

   The query now has a filter to include only sales in North America.

8. In the Measure Group pane, expand Date, and then drag Calendar Year to the **Hierarchy** column in the filter pane.

   The dimension name, Date, is automatically added to the **Dimension** column. Do not change the **Dimension** or **Operator** columns.

9. To open the **Filter Expression** list, click the down arrow in the **Filter Expression** column.

10. In the filter expression list, expand **All Date**, click **Year 2009**, and then click **OK**.

    The query now includes a filter to include only the calendar year 2009.

To create the parameter

1. Expand the Product dimension, and then drag the Product Category Name member to the **Hierarchy** column below **Calendar Year**.
2. Open the **Filter Expression** list, click **All Products**, and then click **OK**.
3. Click the **Parameter** checkbox. The query now includes the parameter ProductProductCategoryName.

> **NOTE**
> The parameter contains the names of product categories. When you click a product category name in the main report, its name is passed to the drillthrough report by using this parameter.

To create the dataset

1. From the Channel dimension, drag Channel Name to the data pane.
2. From the Product dimension, drag Product Category Name to the data pane, and then place it to the right of Channel Name.
3. From the Product dimension, drag Product Subcategory Name to the data pane, and then place it to the right of Product Category Name.
4. In the Metadata pane, expand **Measure**, and then expand Sales.
5. Drag the Sales Amount measure to the data pane, and then place it to the right of Product Subcategory Name.
6. On the query designer toolbar, click **Run (!)**.
7. Click **Next**.

1c. Organize Data into Groups

When you select the fields on which to group the data, you design a matrix with rows and columns that displays detail and aggregated data.

To organize data into groups

1. To switch to design view, click **Design**.
2. On the **Arrange fields** page, drag Product_Subcategory_Name to **Row groups**.

> **NOTE**
> The spaces in the names are replaced with underscores (_). For example Product Category Name is Product_Category_Name.

3. Drag Channel_Name to **Column groups**.
4. Drag Sales_Amount to **Values**.

   Sales_Amount is automatically aggregated by the Sum function, the default aggregate for numeric fields. The value is `[Sum(Sales_Amount)]`.

   To view the other aggregate functions available, open the drop-down list (do not change the aggregate function).

5. Drag Sales_Return_Amount to **Values**, and then place it below `[Sum(Sales_Amount)]`.

   Steps 4 and 5 specify the data to display in the matrix.

6. Click **Next**.

1d. Add Subtotals and Totals

After you create groups, you can add and format rows where the aggregate values for the fields will display. You can also choose whether to show all the data or to let a user expand and collapse grouped data interactively.

To add subtotals and totals

1. On the **Choose the layout** page, under **Options**, verify that **Show subtotals and grand totals** is selected.

   The wizard Preview pane displays a matrix with four rows.

2. Click **Next**.

1e. Choose a Style

A style specifies a font style, a set of colors, and a border style.

To specify a style

1. On the **Choose a Style** page, in the Styles pane, select Slate.

2. Click **Finish**.

   The table is added to the design surface.

3. To preview the report, click **Run (!)**.

2. Format Data as Currency

Apply currency formatting to the sales amount fields in the drillthrough report.

To format data as currency

1. To switch to design view, click **Design**.

2. To select and format multiple cells at one time, press the Ctrl key, and then select the cells that contain the numeric sales data.

3. On the **Home** tab, in the **Number** group, click **Currency**.

3. Add Columns to Show Sales Values in Sparklines

Instead of showing sales and sales returns as currency values, the report shows the values in a sparkline.

To add sparklines to columns

1. To switch to design view, click **Design**.

2. In the Total group of the matrix, right-click the **Sales Amount** column, click **Insert Column**, and then click **Right**.

   An empty column is added to the right of **Sales Amount**.

3. On the ribbon, click **Rectangle**, and then click the empty cell to the right of the `[Sum(Sales_Amount)]` cell in the [Product_Subcategory] row group.

4. On the ribbon, click the **Sparkline** icon, and then click the cell where the rectangle was added.

5. In the **Select Sparkline Type** dialog box, verify that **Column** type is selected.

6. Click **OK**.

7. Right-click the sparkline.
8. In the Chart Data pane, click the **Add field** icon, and then click Sales_Amount.
9. Right-click the `Sales_Return_Amount` column, and then add a column to the right of it.
10. Repeat steps 2 through 6.
11. Right-click the sparkline.
12. In the Chart Data pane, click the **Add field** icon, and then click Sales_Return_Amount.
13. To preview the report, click **Run**.

4. Add Report Title with Product Category Name

A report title appears at the top of the report. You can place the report title in a report header or, if the report does not use one, in a text box at the top of the report body. In this tutorial, you will use the text box that is automatically placed at the top of the report body.

To add a report title

1. To switch to design view, click **Design**.
2. On the design surface, click **Click to add title**.
3. Type **Sales and Returns for Category:**.
4. Right-click, and then click **Create Placeholder**.
5. Click the **(fx)** button to the right of the **Value** list.
6. In the **Expression** dialog box, in the Category pane, click **Dataset**, and then in the **Values** list double-click `First(Product_Category_Name)`.

   The **Expression** box contains the following expression:

   ```
   =First(Fields!Product_Category_Name.Value, "DataSet1")
   ```

7. To preview the report, click **Run**.

The report title includes the name of the first product category. Later, after you run this report as a drillthrough report, the product category name will dynamically change to reflect the name of the product category that was clicked in the main report.

5. Update Parameter Properties

By default parameters are visible, which is not appropriate for this report. You will update the parameter properties for the drillthrough report.

To hide a parameter

1. In the Report Data pane, expand **Parameters**.
2. Right-click @ProductProductCategoryName, and then click **Parameter Properties**.

   > **NOTE**
   > The @ character next to the name indicates that this is a parameter.

3. On the **General** tab, click **Hidden**.

4. In the **Prompt** box, type **Product Category**.

> **NOTE**
> Because the parameter is hidden, this prompt is never used.

5. Optionally, click **Available Values** and **Default Values** and review their options. Do not change any options on these tabs.

6. Click **OK**.

6. Save the Report to a SharePoint Library

You can save the report to a SharePoint library, report server, or your computer. If you save the report to your computer, a number of Reporting Services features such as report parts and subreports are not available. In this tutorial, you will save the report to a SharePoint library.

To save the report

1. From the Report Builder button, click **Save**. The **Save As Report** dialog box opens.

> **NOTE**
> If you are resaving a report, it is automatically resaved to its previous location. To change the location, use the **Save As** option.

2. To show a list of recently used report servers and SharePoint sites, click **Recent Sites and Servers**.

3. Select or type the name of the SharePoint site where you have permission to save reports.

   The URL of the SharePoint library has the following syntax:

```
Http://<ServerName>/<Sites>/
```

4. Click **Save**.

   Recent Sites and Servers lists the libraries on the SharePoint site.

5. Navigate to the library where you will save the report.

6. In the **Name** box, replace the default name with **ResellerVSOnlineDrillthrough**.

> **NOTE**
> You will save the main report to the same location. If you want to save the main and drillthrough reports to different sites or libraries, you must update the path of the **Go to report** action in the main report.

7. Click **Save**.

1. Create a New Report from the Table or Matrix Wizard

From the **Getting Started** dialog box, create a matrix report by using the **Table or Matrix Wizard**.

To create a new report

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2014 Report Builder**, and then click **Report Builder**.

2. In the **Getting Started** dialog box, verify that **New Report** is selected, and then click **Table or Matrix Wizard**.

1a. Specify a Data Connection

You will add an embedded data source to the main report.

To create an embedded data source

1. On the **Choose a dataset** page, select **Create a dataset**, and then click **Next**.
2. Click **New**.
3. In **Name**, type **Online and Reseller Sales Main** as the name for the data source.
4. In **Select a connection type**, select **Microsoft SQL Server Analysis Services**, and then click **Build**.
5. In **Data source**, verify that the data source is **Microsoft SQL Server Analysis Services (AdomdClient)**.
6. In **Server name**, type the name of a server where an instance of MicrosoftAnalysis Services is installed.
7. In **Select or enter a database name**, select the Contoso cube.
8. Click **OK**.
9. Verify that the **Connection string** contains the following syntax:

   ```
   Data Source=<servername>; Initial Catalog = Contoso
   ```

10. Click **Credentials type**.

    Depending on how permissions are configured on the data source, you might need to change the default authentication.

11. Click **OK**.
12. To verify that you can connect to the data source, click **Test Connection**.
13. Click **OK**.
14. Click **Next**.

1b. Create an MDX Query

Next, create an embedded dataset. To do so, you will use the query designer to create filters, parameters, and calculated members as well as the dataset itself.

To create query filters

1. On the **Design a query** page, in the Metadata pane, in the cube section, click the ellipsis **(...)**.
2. In the **Cube Selection** dialog box, click Sales, and then click **OK**.

> **TIP**
> If you do not want to build the MDX query manually, click the icon, toggle the query designer to Query mode, paste the completed MDX to the query designer, and then proceed to step 5 in To create the dataset.

```
WITH MEMBER [Measures].[Net QTY] AS [Measures].[Sales Quantity] -[Measures].[Sales Return Quantity]
MEMBER [Measures].[Net Sales] AS [Measures].[Sales Amount] - [Measures].[Sales Return Amount] SELECT
NON EMPTY { [Measures].[Net QTY], [Measures].[Net Sales] } ON COLUMNS, NON EMPTY { ([Channel].[Channel
Name].[Channel Name].ALLMEMBERS * [Product].[Product Category Name].[Product Category Name].ALLMEMBERS
) } DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS FROM ( SELECT ( { [Date].[Calendar
Year].&[2009] } ) ON COLUMNS FROM ( SELECT ( STRTOSET(@ProductProductCategoryName, CONSTRAINED) ) ON
COLUMNS FROM ( SELECT ( { [Sales Territory].[Sales Territory Group].&[North America] } ) ON COLUMNS
FROM ( SELECT ( { [Channel].[Channel Name].&[2], [Channel].[Channel Name].&[4] } ) ON COLUMNS FROM
[Sales]))) WHERE ( [Sales Territory].[Sales Territory Group].&[North America], [Date].[Calendar
Year].&[2009] ) CELL PROPERTIES VALUE, BACK_COLOR, FORE_COLOR, FORMATTED_VALUE, FORMAT_STRING,
FONT_NAME, FONT_SIZE, FONT_FLAGSQuery text: Code.
```

3. In the Measure Group pane, expand Channel, and then drag Channel Name to the **Hierarchy** column in the filter pane.

   The dimension name, Channel, is automatically added to the **Dimension** column. Do not change the **Dimension** or **Operator** columns.

4. To open the **Filter Expression** list, click the down arrow in the **Filter Expression** column.

5. In the filter expression list, expand **All Channel**, click **Online** and **Reseller**, and then click **OK**.

   The query now includes a filter to include only these channels: Online and Reseller.

6. Expand the Sales Territory dimension, and then drag Sales Territory Group to the **Hierarchy** column, below **Channel Name**.

7. Open the **Filter Expression** list, expand **All Sales Territory**, click **North America**, and then click **OK**.

   The query now has a filter to include only sales in North America.

8. In the Measure Group pane, expand Date, and drag Calendar Year to the **Hierarchy** column in the filter pane.

   The dimension name, Date, is automatically added to the **Dimension** column. Do not change the **Dimension** or **Operator** columns.

9. To open the **Filter Expression** list, click the down arrow in the **Filter Expression** column.

10. In the filter expression list, expand **All Date**, click **Year 2009**, and then click **OK**.

    The query now includes a filter to include only the calendar year 2009.

To create the parameter

1. Expand the Product dimension, and then drag the Product Category Name member to the **Hierarchy** column below **Sales Territory Group**.

2. Open the **Filter Expression** list, click **All Products**, and then click **OK**.

3. Click the **Parameter** checkbox. The query now includes the parameter ProductProductCategoryName.

To create calculated members

1. Place the cursor inside the Calculated Members pane, right-click, and then click **New Calculated Member**.

2. In the Metadata pane, expand **Measures** and then expand Sales.

3. Drag the Sales Quantity measure to the **Expression** box, type the subtraction character (-), and then drag the Sales Return Quantity measure to the **Expression** box; place it after the subtraction character.

   The following code shows the expression:

```
[Measures].[Sales Quantity] - [Measures].[Sales Return Quantity]
```

4. In the Name box, type **Net QTY**, and then click **OK**.

   The Calculated Members pane lists the **Net QTY** calculated member.

5. Right-click **Calculated Members**, and then click **New Calculated Member**.

6. In the Metadata pane, expand **Measures**, and then expand Sales.

7. Drag the Sales Amount measure to the **Expression** box, type the subtraction character (-), and then drag the Sales Return Amount measure to the **Expression** box; place it after the subtraction character.

   The following code shows the expression:

```
[Measures].[Sales Amount] - [Measures].[Sales Return Amount]
```

8. In the **Name** box, type **Net Sales**, and then click **OK**.The Calculated Members pane lists the **Net Sales** calculated member.

To create the dataset

1. From the Channel dimension, drag Channel Name to the data pane.

2. From the Product dimension, drag Product Category Name to the data pane, and then place it to the right of Channel Name.

3. From **Calculated Members**, drag `Net QTY` to the data pane, and then place it to the right of Product Category Name.

4. From Calculated Members, drag Net Sales to the data pane, and then place it to the right of `Net QTY`.

5. On the query designer toolbar, click **Run (!)**.

   Review the query result set.

6. Click **Next**.

1c. Organize Data into Groups

When you select the fields on which to group data, you design a matrix with rows and columns that displays detail and aggregated data.

To organize data into groups

1. On the **Arrange fields** page, drag Product_Category_Name to **Row groups**.

2. Drag Channel_Name to **Column groups**.

3. Drag `Net_QTY` to **Values**.

   `Net_QTY` is automatically aggregated by the Sum function, the default aggregate for numeric fields. The value is `[Sum(Net_QTY)]`.

   To view the other aggregate functions available, open the drop-down list. Do not change the aggregate function.

4. Drag `Net_Sales_Return` to **Values** and then place it below `[Sum(Net_QTY)]`.

   Steps 3 and 4 specify the data to display in the matrix.

1d. Add Subtotals and Totals

You can show subtotals and grand totals in reports. The data in the main report displays as an indicator; you will remove the grand total after you complete the wizard.

To add subtotals and grand totals

1. On the **Choose the layout** page, under **Options**, verify that **Show subtotals and grand totals** is selected.

   The wizard Preview pane displays a matrix with four rows. When you run the report, each row will display in the following way: The first row is the column group, the second row contains the column headings, the third row contains the product category data (`[Sum(Net_ QTY)]` and `[Sum(Net_Sales)]`, and the fourth row contains the totals.

2. Click **Next**.

1e. Choose a Style

Apply the Slate style to the report. This is the same style that the drillthrough report uses.

To specify a style

1. On the **Choose a Style** page, in the Styles pane, select Slate.
2. Click **Finish**.
3. To preview the report, click **Run**.

2. Remove the Grand Total Row

The data values are shown as indictor states, including the column group totals. Remove the row that displays the grand total.

To remove the grand total row

1. To switch to design view, click **Design**.
2. Click the Total row (the last row in the matrix), right-click, and then click **Delete Rows**.
3. To preview the report, click **Run**.

3. Configure Text Box Action for Drillthrough

To enable the drillthrough, specify an action on a text box in the main report.

To enable an action

1. To switch to design view, click **Design**.
2. Right-click the cell that contains Product_Category_Name, and then click **Text Box Properties**.
3. Click the **Action** tab.
4. Select **Go to report.**
5. In **Specify a report**, click **Browse**, and then locate the drillthrough report named ResellerVSOnlineDrillthrough.
6. To add a parameter to run the drillthrough report, click **Add**.
7. In the **Name** list, select ProductProductCategoryName.
8. In **Value**, type `[Product_Category_Name.UniqueName]`.

Product_Category_Name is a field in the dataset.

> **IMPORTANT**
> You must include the `UniqueName` property because the drillthrough action requires a unique value.

9. Click **OK**.

To format the drillthrough field

1. Right-click the cell that contains the `Product_Category_Name`, and then click **Text Box Properties**.
2. Click the **Font** tab.
3. In the **Effects** list, select **Underline**.
4. In the **Color** list, select **Blue**.
5. Click **OK**.
6. To preview your report, click **Run**.

The product category names are in the common link format (blue and underlined).

4. Replace Numeric Values with Indicators

Use indicators to show the state of quantities and sales for Online and Reseller channels.

To add an indicator for Net QTY values

1. To switch to design view, click **Design**.
2. On the ribbon, click the **Rectangle** icon, and then click in the `[Sum(Net QTY)]` cell in the `[Product_Category_Name]` row group in the `Channel_Name` column group.
3. On the ribbon, click the **Indicator** icon, and then click inside the rectangle. The **Select Indicator Type** dialog box opens with the **Directional** indicator selected.
4. Click the **3 Signs** type, and then click **OK**.
5. Right-click the indicator and in the Gauge Data pane, click the down arrow next to **(Unspecified)**. Select `Net_QTY`.
6. Repeat steps 2 through 5 for the `[Sum(Net QTY)]` cell in the `[Product_Category_Name]` row group within **Total**.

To add an indicator for Net Sales values

1. On the ribbon, click the **Rectangle** icon, and then click inside the `[Sum(Net_Sales)]` cell in the `[Product_Category_Name]` row group in the `Channel_Name` column group.
2. On the ribbon, click the **Indicator** icon, and then click inside the rectangle.
3. Click the **3 Signs** type, and then click **OK**.
4. Right-click the indicator and in the Gauge Data pane, click the down arrow next to **(Unspecified)**. Select `Net_Sales`.
5. Repeat steps 1 through 4 for the `[Sum(Net_Sales)]` cell in the `[Product_Category_Name]` row group within **Total**.
6. To preview your report, click **Run**.

5. Update Parameter Properties

By default, parameters are visible, which is not appropriate for this report. You will update the parameter properties to make the parameter internal.

To make the parameter internal

1. In the Report Data pane, expand **Parameters**.
2. Right-click `@ProductProductCategoryName,` and then click **Parameter Properties**.
3. On the **General** tab, click **Internal**.
4. Optionally, click the **Available Values** and **Default Values** tabs and review their options. Do not change any options on these tabs.
5. Click **OK**.

6. Add a Report Title

Add a title to the main report.

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **2009 Product Category Sales: Online and Reseller Category:**.
3. Select the text you typed.
4. On the **Home** tab of the ribbon, in the Font group, select the **Times New Roman** font, **16pt** size, and the **Bold** and **Italic** styles.
5. To preview your report, click **Run**.

7. Save the Main Report to a SharePoint Library

Save the main report to a SharePoint library.

To save the report

1. To switch to design view, click **Design**.
2. From the Report Builder button, click **Save**.
3. Optionally, to show a list of recently used report servers and SharePoint sites, click **Recent Sites and Servers**.
4. Select or type the name of the SharePoint site where you have permission to save reports. The URL of the SharePoint library has the following syntax:

   ```
   Http://<ServerName>/<Sites>/
   ```

5. Navigate to the library where you want to save the report.
6. In **Name**, replace the default name with **ResellerVSOnlineMain**.

> **IMPORTANT**
>
> Save the main report to the same location where you saved the drillthrough report. To save the main and drillthrough reports to different sites or libraries, confirm that the **Go to report** action in the main report, points to the correct location of the drillthrough report.

7. Click **Save**.

8. Run the Main and Drillthrough Reports

Run the main report, and then click values in the product category column to run the drillthrough report.

To run the reports

1. Open the SharePoint library where the reports are saved.
2. Double-click ResellerVSOnlineMain.

   The report runs and displays product category sales information.
3. Click the **Games and Toys** link in the column that contains product category names.

   The drillthrough report runs, displaying only the values for the Games and Toys product category.
4. To return to the main report, click the Internet Explorer back button.
5. Optionally, explore other product categories by clicking their names.

See Also

Tutorials (Report Builder)

Tutorial: Introducing Expressions

2/14/2019 • 20 minutes to read • Edit Online

Expressions help you create powerful and flexible reports. This tutorial teaches you to create and implement expressions that use common functions and operators. You will use the **Expression** dialog box to write expressions that concatenate name values, look up values in a separate dataset, display different pictures based on field values, and so on.

The report is a barred report with alternating row colors in white and a color. The report includes a parameter for selecting the color of the non-white rows.

The following illustration shows a report similar to the one you will create.

Sales Comparison Summary

Name	Gender	State Province	Country Region	Last Purchase	Days Ago	YTD Purchase	+or-AVG
A Martin		Saxony	Germany	11/19/2010	55	29097.6000	
A Nath		Alaska	United States	10/13/2010	92	607.5000	
B Sanchez		North Dakota	United States	9/17/2010	118	6191.0000	
B Sh			Germany	5/10/2008	978	0.0000	
				8/27/2010	139	8772.0000	
O Romeo		Texas		8/30/2010	44	3470.0000	
R Roberts		Washington	United State				
R Caron		NW Territories	Canada	4/29/2010			
W Pal		New South Wales	Australia	7/3/2010	194	747.2500	
Y Sharma		Micronesia	Micronesia	8/23/2010	143	3247.9500	

1/13/2011 3:28:44 PM

What You Will Learn

In this tutorial you will learn how to do the following:

1. Create a Table Report and Dataset from the Table or Matrix Wizard
2. Update Default Names of the Data Source and Dataset
3. Display First Name, Initial, and Last Name
4. Use Images to Display Gender
5. Look Up CountryRegion Name
6. Count Days Since Last Purchase
7. Use an Indicator to Show Sales Comparison
8. Make the Report a "Green Bar" Report

Other Optional Steps

- Format Date Column
- Add a Report Title
- Save the Report

Estimated time to complete this tutorial: 30 minutes.

Requirements

For information about requirements, see Prerequisites for Tutorials (Report Builder).

1. Create a Table Report and Dataset from the Table or Matrix Wizard

Create a table report, a data source, and a dataset. When you lay out the table, you will include only a few fields. After you complete the wizard you will manually add columns. The wizard makes it easy for you to lay out the table and apply a style.

> **NOTE**
> In this tutorial, the query contains the data values, so that it does not need an external data source. This makes the query quite long. In a business environment, a query would not contain the data. This is for learning purposes only.

> **NOTE**
> In this tutorial, the steps for the wizard are consolidated into one procedure. For step-by-step instructions about how to browse to a report server, choose a data source, and create a dataset, see the first tutorial in this series: Tutorial: Creating a Basic Table Report (Report Builder).

To create a new table report

1. Click **Start**, point to **Programs**, click **Microsoft SQL Server 2014Report Builder**, and then click **Report Builder**.

   The **Getting Started** dialog box appears.

   > **NOTE**
   > If the **Getting Started** dialog box does not appear, from the **Report Builder** button, click **New**.

   > **NOTE**
   > If you prefer using the ClickOnce version of Report Builder, open Report Manager and click **Report Builder**, or go to a SharePoint site on which Reporting Services content types such as reports are enabled and click **Report Builder Report** on the **New Document** menu on the **Documents** tab of a shared documents library.

2. In the left pane, verify that **New Report** is selected.
3. In the right pane, click **Table or Matrix Wizard**.
4. On the **Choose a dataset** page, click **Create a dataset**.
5. Click **Next**.
6. On the **Choose a connection to a data source** page, select a data source that is type **SQL Server**. Select a data source from the list or browse to the report server to select one.
7. Click **Next**.
8. On the **Design a query** page, click **Edit as Text**.
9. Paste the following query into the query pane:

```
SELECT 'Lauren' AS FirstName,'Johnson' AS LastName, 'American Samoa' AS StateProvince, 1 AS
CountryRegionID,'Unknown' AS Gender, CAST(9996.60 AS money) AS YTDPurchase, CAST('2010-6-10' AS date) AS
LastPurchase
UNION SELECT'Warren' AS FirstName, 'Bal' AS LastName, 'New South Wales' AS StateProvince, 2 AS
```

```
UNION SELECT 'Warren' AS FirstName, 'Pai' AS LastName, 'New South Wales' AS StateProvince, 2 AS
CountryRegionID, 'Male' AS Gender, CAST(5747.25 AS money) AS YTDPurchase, CAST('2010-7-3' AS date) AS
LastPurchase
UNION SELECT 'Fernando' AS FirstName, 'Ross' AS LastName, 'Alberta' AS StateProvince, 3 AS
CountryRegionID, 'Male' AS Gender, CAST(9248.15 AS money) AS YTDPurchase, CAST('2010-10-17' AS date) AS
LastPurchase
UNION SELECT 'Rob' AS FirstName, 'Caron' AS LastName, 'Northwest Territories' AS StateProvince, 3 AS
CountryRegionID, 'Male' AS Gender, CAST(742.50 AS money) AS YTDPurchase, CAST('2010-4-29' AS date) AS
LastPurchase
UNION SELECT 'James' AS FirstName, 'Bailey' AS LastName, 'British Columbia' AS StateProvince, 3 AS
CountryRegionID, 'Male' AS Gender, CAST(1147.50 AS money) AS YTDPurchase, CAST('2010-6-15' AS date) AS
LastPurchase
UNION SELECT  'Bridget' AS FirstName, 'She' AS LastName, 'Hamburg' AS StateProvince, 4 AS
CountryRegionID, 'Female' AS Gender, CAST(7497.30 AS money) AS YTDPurchase, CAST('2010-5-10' AS date) AS
LastPurchase
UNION SELECT 'Alexander' AS FirstName, 'Martin' AS LastName, 'Saxony' AS StateProvince, 4 AS
CountryRegionID, 'Male' AS Gender, CAST(2997.60 AS money) AS YTDPurchase, CAST('2010-11-19' AS date) AS
LastPurchase
UNION SELECT 'Yolanda' AS FirstName, 'Sharma' AS LastName ,'Micronesia' AS StateProvince, 5 AS
CountryRegionID, 'Female' AS Gender, CAST(3247.95 AS money) AS YTDPurchase, CAST('2010-8-23' AS date) AS
LastPurchase
UNION SELECT 'Marc' AS FirstName, 'Zimmerman' AS LastName, 'Moselle' AS StateProvince, 6 AS
CountryRegionID, 'Male' AS Gender, CAST(1200.00 AS money) AS YTDPurchase, CAST('2010-11-16' AS date) AS
LastPurchase
UNION SELECT 'Katherine' AS FirstName, 'Abel' AS LastName, 'Moselle' AS StateProvince, 6 AS
CountryRegionID, 'Female' AS Gender, CAST(2025.00 AS money) AS YTDPurchase, CAST('2010-12-1' AS date) AS
LastPurchase
UNION SELECT 'Nicolas' as FirstName, 'Anand' AS LastName, 'Seine (Paris)' AS StateProvince, 6 AS
CountryRegionID, 'Male' AS Gender, CAST(1425.00 AS money) AS YTDPurchase, CAST('2010-12-11' AS date) AS
LastPurchase
UNION SELECT 'James' AS FirstName, 'Peters' AS LastName, 'England' AS StateProvince, 12 AS
CountryRegionID, 'Male' AS Gender, CAST(887.50 AS money) AS YTDPurchase, CAST('2010-8-15' AS date) AS
LastPurchase
UNION SELECT 'Alison' AS FirstName, 'Nath' AS LastName, 'Alaska' AS StateProvince, 7 AS CountryRegionID,
'Female' AS Gender, CAST(607.50 AS money) AS YTDPurchase, CAST('2010-10-13' AS date) AS LastPurchase
UNION SELECT 'Grace' AS FirstName, 'Patterson' AS LastName, 'Kansas' AS StateProvince, 7 AS
CountryRegionID, 'Female' AS Gender, CAST(1215.00 AS money) AS YTDPurchase, CAST('2010-10-18' AS date)
AS LastPurchase
UNION SELECT 'Bobby' AS FirstName, 'Sanchez' AS LastName, 'North Dakota' AS StateProvince, 7 AS
CountryRegionID, 'Female' AS Gender, CAST(6191.00 AS money) AS YTDPurchase, CAST('2010-9-17' AS date) AS
LastPurchase
UNION SELECT 'Charles' AS FirstName, 'Reed' AS LastName, 'Nebraska' AS StateProvince, 7 AS
CountryRegionID, 'Male' AS Gender, CAST(8772.00 AS money) AS YTDPurchase, CAST('2010-8-27' AS date) AS
LastPurchase
UNION SELECT 'Orlando' AS FirstName, 'Romeo' AS LastName, 'Texas' AS StateProvince, 7 AS
CountryRegionID, 'Male' AS Gender, CAST(8578.00 AS money) AS YTDPurchase, CAST('2010-7-29' AS date) AS
LastPurchase
UNION SELECT 'Cynthia' AS FirstName, 'Randall' AS LastName, 'Utah' AS StateProvince, 7 AS
CountryRegionID, 'Female' AS Gender, CAST(7218.10 AS money) AS YTDPurchase, CAST('2010-1-11' AS date) AS
LastPurchase
UNION SELECT 'Rebecca' AS FirstName, 'Roberts' AS LastName, 'Washington' AS StateProvince, 7 AS
CountryRegionID, 'Female' AS Gender, CAST(8357.80 AS money) AS YTDPurchase, CAST('2010-10-28' AS date)
AS LastPurchase
UNION SELECT 'Cristian' AS FirstName, 'Petulescu' AS LastName, 'Wisconsin' AS StateProvince, 7 AS
CountryRegionID, 'Male' AS Gender, CAST(3470.00 AS money) AS YTDPurchase, CAST('2010-11-30' AS date) AS
LastPurchase
UNION SELECT 'Cynthia' AS FirstName, 'Randall' AS LastName, 'Utah' AS StateProvince, 7 AS
CountryRegionID, 'Female' AS Gender, CAST(7218.10 AS money) AS YTDPurchase, CAST('2010-1-11' AS date) AS
LastPurchase
UNION SELECT 'Rebecca' AS FirstName, 'Roberts' AS LastName, 'Washington' AS StateProvince, 7 AS
CountryRegionID, 'Female' AS Gender, CAST(8357.80 AS money) AS YTDPurchase, CAST('2010-10-28' AS date)
AS LastPurchase
UNION SELECT 'Cristian' AS FirstName, 'Petulescu' AS LastName, 'Wisconsin' AS StateProvince, 7 AS
CountryRegionID, 'Male' AS Gender, CAST(3470.00 AS money) AS YTDPurchase, CAST('2010-11-30' AS date) AS
LastPurchase
```

The query specifies column names that include a birth date, first name, last name, state or province, country/region identifier, gender, and year-to-date purchases.

10. On the query designer toolbar, click **Run** (**!**). The result set displays 20 rows of data and includes the following columns: FirstName, LastName, StateProvince, CountryRegionID, Gender, YTDPurchase, and LastPurchase.

11. Click **Next**.

12. On the **Arrange fields** page, drag the following fields, in the specified order, from the **Available Fields** list to the **Values** list.

    - StateProvince
    - CountryRegionID
    - LastPurchase
    - YTDPurchase

    Because the CountryRegionID and YTDPurchase contain numeric data, the SUM aggregate is applied to them by default.

    > **NOTE**
    > The FirstName and LastName fields are not included. You will add them in a later step.

13. In the **Values** list, right-click `CountryRegionID` and click the **Sum** option.

    Sum is no longer applied to CountryRegionID.

14. In the **Values** list, right-click **YTDPurchase** and click the **Sum** option.

    Sum is no longer applied to YTDPurchase.

15. Click **Next**.

16. On the **Choose the layout** page, click **Next**.

17. On the **Choose a style** page, click **Slate**, and then click **Finish**.

2. Update Default Names of the Data Source and Dataset

To update the default name of the data source

1. In the Report Data pane, expand **Data Sources**.

2. Right-click **DataSource1** and click **Data Source Properties.**

3. In the **Name** box, type **ExpressionsDataSource**

4. Click **OK**.

To update the default name of the dataset

1. In the Report Data pane, expand **Datasets**.

2. Right-click **DataSet1** and click **Dataset Properties.**

3. In the **Name** box, type **Expressions**

4. Click **OK**.

3. Display First Name, Initial, and Last Name

Use the **Left** function and the **Concatenate** (**&**) operator in an expression that evaluates to a name that includes

an initial and a last name. You can build the expression step by step or skip ahead in the procedure and copy/paste the expression from the tutorial into the **Expression** dialog box.

To add the Name column

1. Right-click the **StateProvince** column, point to **Insert Column**, and then click **Left**.

   A new column is added to the left of the **StateProvince** column.

2. Click the title of the new column and type **Name**

3. Right-click the data cell for the **Name** column and click **Expression**.

4. In the **Expression** dialog box, expand **Common Functions**, and then click **Text**.

5. In the **Item** list, double-click **Left**.

   The **Left** function is added to the expression.

6. In the **Category** list, click **Fields (Expressions)**.

7. In the **Values** list, double-click **FirstName**.

8. Type **, 1)**

   This expression extracts one character from the **FirstName** value, counting from the left.

9. Type **&" "&**

10. In the **Values** list, double-click **LastName**.

    The completed expression: `=Left(Fields!FirstName.Value, 1) &" "& Fields!LastName.Value`

11. Click **OK**.

12. Click **Run** to preview the report.

4. Use Images to Display Gender

Use images to show the gender of a person, and identify unknown gender values by using a third image. You will add to the report three hidden images and a new column to display the images, and then determine the image that appears in the column based on the value of the Gender field.

To apply a color to the table cell that contains the image when you make the report a barred report, you will add a rectangle and then add the image to the rectangle. You need to use a rectangle because you can apply a background color to a rectangle, but not to an image.

The tutorial uses images that are installed with Windows, but you can use any images available to you. You will use embedded images, and they do not need to be installed on your local computer or the report server.

To add images to the report body

1. Click **Design** to return to design view.

2. On the **Insert** tab of the ribbon, click **Image** and then click in the report body, below the table.

   The **Image Properties** dialog box opens.

3. Click **Import** and navigate to C:\Users\Public\Public Pictures\Sample Pictures.

4. Click Penguins.JPG and click **Open**.

   In the **Image Properties** dialog box, click **Visibility** and then click the **Hide** option.

5. Click **OK**.

6. Repeat steps 2 through 5, but choose Koala.JPG.
7. Repeat steps 2 through 5, but choose Tulips.JPG.

To add the Gender column

1. Right-click the **Name** column, point to **Insert Column**, and then click **Right**.

   A new column is added to the right of the **Name** column.

2. Click the title of the new column and type **Gender**

To add a rectangle

- On the **Insert** tab of the ribbon, click **Rectangle** and then click in the data cell of the **Gender** column.

  A rectangle is added to the cell.

To add an image to the rectangle

1. Right-click in the rectangle, point to **Insert**, and then click **Image**.
2. In the **Image Properties** dialog box, click the down arrow beside **Use this image**, and select one of the images you added, for example, Penguins.JPG.
3. Click **OK**.

To use images to show gender

1. Right-click the image in the data cell in the **Gender** column and click **Image Properties**.
2. In the **Image Properties** dialog box, click the expression **fx** button next to the **Use this image** text box.
3. In the **Expression** dialog box, expand **Common Functions** and click **Program Flow**.
4. In the **Item** list, double-click **Switch**.
5. In the **Category** list, click **Fields (Expressions)**.
6. In the **Values** list, double-click **Gender**.
7. Type **="Male", "Koala",**
8. In the **Values** list, double-click **Gender**.
9. Type **="Female", "Penguins",**
10. In the **Values** list, double-click **Gender**.
11. Type **="Unknown", "Tulips")**

    The completed expression:

    ```
    =Switch(Fields!Gender.Value ="Male", "Koala",Fields!Gender.Value ="Female","Penguins",Fields!Gender.Value
    ="Unknown","Tulips")
    ```

12. Click **OK**.
13. Click **OK** again to close the **Image Properties** dialog box.
14. Click **Run** to preview the report.

5. Look Up CountryRegion Name

Create the CountryRegion dataset and use the **Lookup** function to display the name of a country/region instead of the identifier of the country/region.

To create the CountryRegion dataset

1. Click **Design** to return to design view.

2. In the Report Data pane, click **New** and then click **Dataset**.
3. Click **Use a dataset embedded in my report**.
4. In the **Data source** list, select ExpressionsDataSource.
5. In the **Name** box, type **CountryRegion**
6. Verify that the **Text** query type is selected and click **Query Designer**.
7. Click **Edit as Text**.
8. Copy and paste the following query into the query pane:

```
SELECT 1 AS ID, 'American Samoa' AS CountryRegion
UNION SELECT 2 AS CountryRegionID, 'Australia' AS CountryRegion
UNION SELECT 3 AS ID, 'Canada' AS CountryRegion
UNION SELECT 4 AS ID, 'Germany' AS CountryRegion
UNION SELECT 5 AS ID, 'Micronesia' AS CountryRegion
UNION SELECT 6 AS ID, 'France' AS CountryRegion
UNION SELECT 7 AS ID, 'United States' AS CountryRegion
UNION SELECT 8 AS ID, 'Brazil' AS CountryRegion
UNION SELECT 9 AS ID, 'Mexico' AS CountryRegion
UNION SELECT 10 AS ID, 'Japan' AS CountryRegion
UNION SELECT 10 AS ID, 'Australia' AS CountryRegion
UNION SELECT 12 AS ID, 'United Kingdom' AS CountryRegion
```

9. Click **Run** (**!**) to run the query.

   The query results are the country/region identifiers and names.

10. Click **OK**.
11. Click **OK** again to close the **Dataset Properties** dialog box.

To look up values in the CountryRegion dataset

1. Click the **Country Region ID** column title and delete the text: ID.
2. Right-click the data cell for the **Country Region** column and click **Expression**.
3. Delete the expression except the initial equal (=) sign.

   The remaining expression is: `=`

4. In the **Expression** dialog box, expand **Common Functions** and click **Miscellaneous**.
5. In the **Item** list, double-click **Lookup**.
6. In the **Category** list, click **Fields (Expressions)**.
7. In the **Values** list, double-click `CountryRegionID`.
8. If the cursor is not already immediately after `CountryRegionID.Value`, place it there.
9. Delete the right parenthesis and then type **,Fields!ID.value, Fields!CountryRegion.value, "CountryRegion")**

   The completed expression:
   `=Lookup(Fields!CountryRegionID.Value,Fields!ID.value, Fields!CountryRegion.value, "CountryRegion")`

   The syntax of the **Lookup** function specifies a lookup between CountryRegionID and ID in the CountryRegion dataset that returns the CountryRegion value, which is also in the CountryRegion dataset.

10. Click **OK**.

11. Click **Run** to preview the report.

6. Count Days Since Last Purchase

Add a column and then use the **Now** function or the `ExecutionTime` built-in global variable to calculate the number of days from today since a person's last purchases.

To add the Days Ago column

1. Click **Design** to return to design view.
2. Right-click the **Last Purchase** column, point to **Insert Column**, and then click **Right**.

   A new column is added to the right of the **Last Purchase** column.
3. In the column header, type **Days Ago**
4. Right-click the data cell for the **Days Ago** column and click **Expression**.
5. In the **Expression** dialog box, expand **Common Functions**, and then click **Date & Time**.
6. In the **Item** list, double-click **DateDiff**.
7. If the cursor is not already immediately after `DateDiff(`, place it there.
8. Type **"d",**
9. In the **Category** list, click **Fields (Expressions)**.
10. In the **Values** list, double-click **LastPurchase**.
11. If the cursor is not already immediately after `Fields!LastPurchase.Value`, place it there.
12. Type **,**
13. In the **Category** list, click **Date & Time** again.
14. In the **Item** list, double-click **Now**.

> **WARNING**
> In production reports you should not use the **Now** function in expressions that are evaluated multiple times as the report renders (for example, in the detail rows of a report). The value of **Now** changes from row to row and the different values affect the evaluation results of expressions, which leads to results that are subtly inconsistent. Instead, you should use the `ExecutionTime` global variable that Reporting Services provides.

15. If the cursor is not already immediately after `Now(`, place it there.
16. Delete the left parenthesis and then type **)**

    The completed expression: `=DateDiff("d", Fields!LastPurchase.Value, Now)`
17. Click **OK**.

7. Use an Indicator to Show Sales Comparison

Add a new column and use an indicator to show whether a person's year-to-date (YTD) purchases are above or below the average YTD purchases. The **Round** function removes decimals from values.

The configuration of the indicator and its states requires many steps. If you want to, in the "To configure the indicator" procedure, you can skip ahead and copy/paste the completed expressions from this tutorial into the **Expression** dialog box.

To add the + or - AVG Sales column

1. Right-click the **YTD Purchase** column, point to **Insert Column**, and then click **Right**.

   A new column is added to the right of the **YTD Purchase** column.

2. Click the title of the column and type **+ or - AVG Sales**

To add an indicator

1. On the **Insert** tab of the ribbon, click **Indicator**, and then click the data cell for the **+ or - AVG Sales** column.

   The **Select Indicator Type** dialog box opens.

2. In the **Directional** group of icon sets, click the set of three gray arrows.

3. Click **OK**.

To configure the indicator

1. Right-click the indicator, click **Indicator Properties**, and then click **Value and States**.

2. Click the expression **fx** button next to the **Value** text box.

3. In the **Expression** dialog box, expand **Common Functions**, and then click **Math**.

4. In the **Item** list, double-click **Round**.

5. In the **Category** list, click **Fields (Expressions)**.

6. In the **Values** list, double-click **YTDPurchase**.

7. If the cursor is not already immediately after `Fields!YTDPurchase.Value`, place it there.

8. Type **-**

9. Expand **Common Functions** again and click **Aggregate**.

10. In the **Item** list, double-click **Avg**.

11. In the **Category** list, click **Fields (Expressions)**.

12. In the **Values** list, double-click **YTDPurchase**.

13. If the cursor is not already immediately after `Fields!YTDPurchase.Value`, place it there.

14. Type **, "Expressions"))**

    The completed expression:
    `=Round(Fields!YTDPurchase.Value - Avg(Fields!YTDPurchase.Value, "Expressions"))`

15. Click **OK**.

16. In the **States Measurement Unit** box, select **Numeric**.

17. In the row with the down-pointing arrow, click the **fx** button to the right of the text box for the **Start** value.

18. In the **Expression** dialog box, expand **Common Functions**, and then click **Math**.

19. In the **Item** list, double-click **Round**.

20. In the **Category** list, click **Fields (Expressions)**.

21. In the **Values** list, double-click **YTDPurchase**.

22. If the cursor is not already immediately after `Fields!YTDPurchase.Value`, place it there.

23. Type **-**
24. Expand **Common Functions** again and click **Aggregate**.
25. In the **Item** list, double-click **Avg**.
26. In the **Category** list, click **Fields (Expressions)**.
27. In the **Values** list, double-click **YTDPurchase**.
28. If the cursor is not already immediately after `Fields!YTDPurchase.Value`, place it there.
29. Type **, "Expressions")) < 0**

    The completed expression:
    ```
    =Round(Fields!YTDPurchase.Value - Avg(Fields!YTDPurchase.Value, "Expressions")) < 0
    ```
30. Click **OK**.
31. In the text box for the **End** value, type **0**
32. Click the row with the horizontal-pointing arrow and click **Delete**.
33. In the row with the up-pointing arrow, in the **Start** box, type **0**
34. Click the **fx** button to the right of the text box for the **End** value.
35. In the **Expression** dialog box, create the expression:
    ```
    =Round(Fields!YTDPurchase.Value - Avg(Fields!YTDPurchase.Value, "Expressions")) >0
    ```
36. Click **OK**.
37. Click **OK** again to close the **Indicator properties** dialog box.
38. Click **Run** to preview the report.

8. Make the Report a "Green Bar" Report

Use a parameter to specify the color to apply to alternating rows in the report, making it a barred report.

To add a parameter

1. Click **Design** to return to design view.
2. In the **Report Data** pane, right-click **Parameters** and click **Add Parameter**.

   The **Report Parameter Properties** dialog box opens.
3. In **Prompt**, type **Choose color**
4. In **Name**, type **RowColor**
5. In the left pane, click **Available Values**.
6. Click **Specify values**.
7. Click **Add**.
8. In the **Label** box, type: **Yellow**
9. In the **Value** box, type **Yellow**
10. Click **Add**.
11. In the **Label** box, type **Green**

12. In the **Value** box, type **PaleGreen**
13. Click **Add**.
14. In the **Label** box, type **Blue**
15. In the **Value** box, type **LightBlue**
16. Click **Add**.
17. In the **Label** box, type **Pink**
18. In the **Value** box, type **Pink**
19. Click **OK**.

To apply alternating colors to detail rows

1. Click the **View** tab on the ribbon and verify that **Properties** is selected.
2. Click the data cell for the **Name** column and press the Shift key.
3. One by one, click the other cells in the row.
4. In the Properties pane, click **BackgroundColor**.

   If your Properties pane lists properties by category, you will find the **BackgroundColor** under the **Fill** category.
5. Click the down arrow and then click **Expression**.
6. In the **Expression** dialog box, expand **Common Functions**, and then click **Program Flow**.
7. In the **Item** list, double-click **IIf**.
8. Expand **Common Functions** and click **Aggregate**.
9. In the **Item** list, double-click **RunningValue**.
10. In the **Category** list, click **Fields (Expressions)**.
11. In the **Values** list, double-click **FirstName**.
12. If the cursor is not already immediately after `Fields!FirstName.Value`, place it there and type **,**
13. Expand **Common Functions** and click **Aggregate**.
14. In the **Item** list, double-click **Count**.
15. If the cursor is not already immediately after `Count(`, place it there.
16. Delete the left parenthesis and then type **,"Expressions")**

    > **NOTE**
    >
    > Expressions is the name of the dataset in which to count data rows.

17. Expand **Operators** and click **Arithmetic**.
18. In the **Item** list, double-click **Mod**.
19. If the cursor is not already immediately after `Mod`, place it there.
20. Type **2 =0,**

> **IMPORTANT**
>
> Be sure you include a space before you type the number 2.

21. Click **Parameters** and in the **Values** list, double-click **RowColor**.
22. If the cursor is not already immediately after `Parameters!RowColor.Value`, place it there.
23. Type **, "White")**

    The completed expression:

    ```
    =IIf(RunningValue(Fields!FirstName.Value,Count, "Expressions") Mod 2 =0, Parameters!RowColor.Value,
    "White")
    ```

24. Click **OK**.

Run the Report

1. If not on the **Home** tab, click **Home** to return to design view.
2. Click **Run**.
3. In the **Choose color** drop-down list, select the color of the non-white bars on the report.
4. Click **View Report**.

   The report renders and alternating rows have the background that you chose.

(optional) Format Date Column

Format the **Last Purchase** column, which contains dates.

To format date column

1. Click **Design** to return to design view.
2. Right-click the data cell for the **Last Purchase** column and click **Text Box Properties**.
3. In the **Text Box Properties** dialog box, click **Number**, click **Date**, and then click the type ***1/31/2000**.
4. Click **OK**.

(optional) Add a Report Title

Add a title to the report.

To add a report title

1. On the design surface, click **Click to add title**.
2. Type **Sales Comparison Summary**, and then click outside the text box.
3. Right-click the text box that contains **Sales Comparison Summary** and click **Text Box Properties**.
4. In the **Text Box Properties** dialog box, click **Font**.
5. In the **Size** list, select **18pt**.
6. In the **Color** list, select **Gray**.
7. Select **Bold** and **Italic**.
8. Click **OK**.

(optional) Save the Report

You can save reports to a report server, SharePoint library, or your computer. For more information, see Saving Reports (Report Builder).

In this tutorial, save the report to a report server. If you do not have access to a report server, save the report to your computer.

To save the report to a report server

1. From the **Report Builder** button, click **Save As**.
2. Click **Recent Sites and Servers**.
3. Select or type the name of the report server where you have permission to save reports.

   The message "Connecting to report server" appears. When the connection is complete, you will see the contents of the report folder that the report server administrator specified as the default report location.
4. In **Name**, replace the default name with **Sales Comparison Summary**.
5. Click **Save**.

The report is saved to the report server. The name of report server that you are connected to appears in the status bar at the bottom of the window.

To save the report to your computer

1. From the **Report Builder** button, click **Save As**.
2. Click **Desktop** , **My Documents**, or **My computer**, and then browse to the folder where you want to save the report.
3. In **Name**, replace the default name with **Sales Comparison Summary**.
4. Click **Save**.

See Also

Expressions (Report Builder and SSRS)
Expression Examples (Report Builder and SSRS)
Indicators (Report Builder and SSRS)
Images, Text Boxes, Rectangles, and Lines (Report Builder and SSRS)
Tables (Report Builder and SSRS)
Add Data to a Report (Report Builder and SSRS)

Enterprise Information Management Tutorials

2/11/2019 • 2 minutes to read • Edit Online

This section contains tutorials for managing information in an enterprise by using Enterprise Information Management (EIM) technologies that are included in SQL Server 2012. Enterprise Integration Management (EIM) provides a portfolio of solutions that enable organizations to trust the credibility and consistency of their data so they can make critical business decisions. SQL Server 2012 has the following technologies that help you implement EIM solutions in your enterprise.

- SQL Server Integration Services (SSIS). SSIS provides a powerful, extensible platform for integrating data from various sources in a comprehensive extract, transform, and load (ETL) solution that supports business workflows, a data warehouse, or master data management.
- SQL Server Data Quality Services (DQS). DQS enables you to cleanse, match, standardize, and enrich data, so you can deliver trusted information for business intelligence, a data warehouse, and transaction processing workloads.
- SQL Server Master Data Services (MDS). MDS provides a central data hub that ensures that the integrity of information and consistency of data is constant across different applications.

Enterprise Information Management using SSIS, MDS, and DQS Together [Tutorial]
In this tutorial, you learn how to use SSIS, MDS, and DQS together to implement a sample Enterprise Information Management (EIM) solution. First, you use DQS to create a knowledgebase with the knowledge about the supplier data (metadata), cleanse the data in an Excel file against the knowledge base, and match the data to identify and remove duplicates in the data. Next, you use the MDS Add-in for Excel to upload the cleansed and matched data to MDS. Then, you automate the whole process by using an SSIS solution.

See Also

Enterprise Information Management - Microsoft SQL Server

Enterprise Information Management using SSIS, MDS, and DQS Together [Tutorial]

2/11/2019 • 4 minutes to read • Edit Online

Managing information in an enterprise typically involves integrating data from across the enterprise and beyond, cleansing the data, matching the data to remove any duplicates, standardizing the data, enriching the data, making the data conform to legal and compliance requirements, and then storing the data in a centralized location with all the necessary security settings.

SQL Server 2014 provides all the components needed for an effective Enterprise Information Management (EIM) solution in a single product. Key components that help you build an EIM solution are:

- SQL Server Integration Services
- SQL Server Data Quality Services
- SQL Server Master Data Services

SQL Server Integration Services (SSIS) provides a powerful, extensible platform for integrating data from various sources in a comprehensive extract, transform, and load (ETL) solution that supports business workflows, a data warehouse, or master data management. See Integration Services Overview topic for a quick overview and typical uses of SSIS.

SQL Server Data Quality Services (DQS) enables you to cleanse, match, standardize, and enrich data, so you can deliver trusted information for business intelligence, a data warehouse, and transaction processing workloads. See Introducing Data Quality Services topic for the business need for DQS and how DQS answers the need.

SQL Server Master Data Services (MDS) provides a central data hub that ensures that the integrity of information and consistency of data is constant across different applications. See Master Data Services Overview topic for brief descriptions of important features of MDS.

See Cleansing and Matching Master Data using EIM Technologies whitepapers for a comprehensive guidance on implementing an EIM solution using these Microsoft EIM technologies together and watch Enterprise Information Management (EIM): Bringing together SSIS, DQS, and MDS video for a cool demonstration of an EIM scenario.

In this tutorial, you learn how to use SSIS, MDS, and DQS together to implement a sample Enterprise Information Management (EIM) solution. First, you use DQS to create a knowledgebase that contains knowledge about the data (metadata), cleanse the data in an Excel file by using the knowledge base, and match the data to identify and remove duplicates in the data. Next, you use the MDS Add-in for Excel to upload the cleansed and matched data to MDS. Then, you automate the whole process by using an SSIS solution. The SSIS solution in this tutorial reads the input data from an Excel file, but you can extend it to read from various sources such as Oracle, Teradata, DB2, and Windows Azure SQL Database.

Prerequisites

1. Microsoft SQL Server 2012 with the following components installed.
    a. Integration Services (SSIS)
    b. Master Data Services (MDS)
    c. Data Quality Services (DQS)

d. SQL Server Data Tools

See SQL Server 2012 Installation Guide for details about installing the product.

2. Configure MDS using Master Data Services Configuration Manager

   Use the Configuration Manager to create and configure a Master Data Services database. After you create the MDS database, create a web application for MDS in a web site (for example: http://localhost/MDS) and associate the MDS database with the MDS web application. Note that, to create an MDS web application, you should have IIS installed on your computer. See Web Application Requirements (Master Data Services) and Database Requirements (Master Data Services) for details about the prerequisites for configuring MDS database and web application.

3. Install and Configure DQS using Data Quality Server Installer. Click **Start**, click **All Programs**, click **Microsoft SQL Server 2014**, click **Data Quality Services**, and then click **Data Quality Server Installer**.

4. Microsoft Excel 2010 (32-bit is preferred).

5. Install **Master Data Services Add-in for Excel** (32-bit or 64-bit based on the version of Excel you have on your computer) from here. To find the version of Excel installed on your computer, run **Excel**, click **File** on menu bar and click **Help** to see the version in the right pane. Note that you need to install Visual Studio 2010 Tools for Office Runtime before installing the Excel Add-in.

6. (Optional) Create an account with Windows Azure Marketplace. One of the tasks in the tutorial requires you to have an **Azure Marketplace** (originally named **Data Market**) account. You can skip this task if you want and proceed with the next task.

7. Download the Suppliers.xls file from Microsoft Download Center.

8. DQS does not allow you to export the cleansing or matching results to an Excel file if you are using **64-bit version of Excel**. This issue is a known issue. To work around the issue, do the following:

   a. Run **DQLInstaller.exe -upgrade**. If you installed the default instance of SQL Server, the DQSInstaller.exe file is available at C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\Binn. Double-click the DQSInstaller.exe file.

   b. In **Master Data Services Configuration Manager**, click **Select Database**, select existing **MDS** database, and then click **Upgrade**.

Lessons

LESSON	BRIEF DESCRIPTION	ESTIMATED TIME TO COMPLETE (IN MINUTES).
Lesson 1: Creating the Suppliers DQS Knowledge Base	In this lesson, you create a DQS knowledge base named **Suppliers**.	60
Lesson 2: Cleansing Supplier Data using the Suppliers Knowledge Base	In this lesson, you create and run a DQS project to cleanse the supplier data in an Excel file by using the **Suppliers** Knowledge Base you created in the first lesson.	45
Lesson 3: Matching Data to Remove Duplicates from Supplier List	In this lesson, you create a DQS project to perform matching activity to identify and remove duplicates from the cleansed suppler list.	45

LESSON	BRIEF DESCRIPTION	ESTIMATED TIME TO COMPLETE (IN MINUTES).
Lesson 4: Storing Supplier Data in MDS	In this lesson, you upload the cleansed and matched supplier data to Master Data Services (MDS) by using the **MDS Add-in for Excel**.	45
Lesson 5: Automating the Cleansing and Matching using SSIS	In this lesson, you create an SSIS solution that cleanses input data by using DQS, matches the cleansed data to remove duplicates, and stores the cleansed and matched data on MDS in an automated manner.	75

Next Steps

To begin the tutorial, continue to the first lesson: Lesson 1: Creating the Suppliers DQS Knowledge Base.

Lesson 1: Creating the Suppliers DQS Knowledge Base

2/11/2019 • 3 minutes to read • Edit Online

In this lesson, you create a DQS knowledge base named **Suppliers** with the knowledge (metadata) about supplier data. You use the knowledge base to perform the cleansing and matching activities on input supplier data. The cleansing activity identifies incorrect/invalid data, corrects the incorrect data, proposes corrections/suggestions, standardizes the data, and enriches the data with more information. The matching activity compares data and identifies similar records (but slightly different) in the data that helps you remove duplicates on the data.

You can use both interactive and computer-assisted processes to create, build, and manage a knowledge base. Knowledge in a knowledge base is maintained in domains, each of which is specific to a data field in the data that you want to cleanse and/or match.

In this lesson, you perform the following tasks to create the **Suppliers** knowledge base:

- Create a DQS knowledge base named **Suppliers**. You can create a knowledge base in several ways. You can build a knowledge base from scratch or build it based on an existing knowledge base or by importing a DQS file (.dqs) that contains a prebuilt and exported knowledge base, or by performing a knowledge discovery activity on sample data. In this tutorial, you create the knowledge base from scratch.
- Create domains in the **Suppliers** knowledge base that you use for cleansing data, and matching data to identify duplicates. create domains for data fields that you want to use in cleansing and matching activities, not for all the data fields in the data.
- Add values to a domain by adding values manually, importing values from an excel file, by performing a knowledge discovery activity on sample data, and by importing project values from a cleansing project. You can also import domain values by importing a DQS file that contains domain properties and values, which you do not perform in the tutorial.
- Set rules for a domain. A domain rule is a condition that is used by DQS to validate, correct, and standardize domain values.
- Set term-based relationships for a domain. A term-based relationship enables you to make a correction to a term that is part of a value in a domain. For example, in the value **Contoso Inc., Inc.** is a term that can be defined as Incorporated. This helps in standardizing the data as well as in identifying duplicates. For example, **Contoso Inc.** and **Contoso Incorporated** can be considered duplicates.
- Specify synonyms in domain values. You can set two or more values as synonyms and set one of them as a leading value, which replaces its synonym values during a cleansing activity to standardize the data.
- Create a composite domain named Address Validation that comprises Address line, City, State, and Zip domains. A composite domain is a domain that consists of one or more single domains. It lets you create a rule that involves multiple domains. For example, you can define a rule: if City is Los Angeles, State must be CA, where City and State are two separate domains.
- Configure and use a reference data service. The Reference Data Service feature in Data Quality Services (DQS) enables you to subscribe to third-party reference data providers, and to cleanse and enrich your business data by validating it against their high-quality data. You can use services from leading DQS providers from within DQS to standardize, correct, or enrich your data during the cleansing process. In this tutorial, you learn how to configure your DQS environment to use a reference data service on Windows Azure Marketplace and use the service associated with the Address Validation composite domain to cleanse

address data.

- Publish the knowledge base so that the knowledge base can be used in cleansing and matching activities.

Next Step

Task 1: Creating a Knowledge Base and Domains

Task 1: Creating a Knowledge Base and Domains

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create the **Suppliers** knowledge base and create domains that is used for cleansing data and matching data to remove duplicates.

1. Launch **Data Quality Client**. Click **Start**, point to **All Programs**, click **Microsoft SQL Server 2012**, click **Data Quality Services**, and then click **Data Quality Client**.

2. In the **Connect to Server** dialog box, select the database server instance on which the DQS is installed, and click **Connect**.

3. In the Data Quality Client home page, in the **Knowledge Base Management** pane, click **New Knowledge Base**.

4. Enter **Suppliers** for **Name** of the knowledge base.

5. Confirm that **Create Knowledge Base from** field is set to **None** since you are creating the **Suppliers** knowledge base from scratch.

6. Confirm that **Domain Management** is selected for the **Activity** and click **Next**. The Domain Management activity lets you create and manage domains in the knowledge base.

7. In the **Domain Management** window, click **Create a domain** toolbar button to create a domain.

8. In the **Create Domain** dialog box, type **Supplier ID** for the **Domain Name**, and click **OK**.

9. Repeat previous step to create the following domains with all the default settings. To keep the tutorial simple, you set the **Data Type** of all the domains as **String**. The other allowed data types are: Integer, Decimal, and Date. When the **Use Leading Values** option is selected (default), all synonyms are replaced with the leading value of the synonym group in the output. Setting **Normalize String** option (default) removes any special characters in the domain values. The **Format Output to** option lets you select the formatting that is applied when the data values in the domain are output. Select **Enable Speller** (default) to run Speller on all string values when populating the domain. The **Language** setting specifies which language version of the **Speller** you want to apply. Select **Disable Syntax Error Algorithms** to populate the domain without checking string values for syntax errors. See Create a Domain topic in the MSDN library for more details.

   - Supplier Name
   - Contact Email
   - Address Line
   - City
   - State
   - Country
   - Zip

Next Step

Task 2: Adding Domain Values Manually

Task 2: Adding Domain Values Manually

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add a value for the **Country** domain manually. See Change Domain Values topic for more details about the fields on this page.

1. Click **Country** domain in the **Domain list**.
2. In the right pane, switch to the **Domain Values** tab.
3. Click **Add new domain value** button on the toolbar in the right pane.

4. Type **United States** for the **Value** field and press **ENTER**. You can see that, by default, the **Type** is set to **Correct** (green check). The Type can be set to **Error** (red cross) or **Invalid** (orange triangle), and a correct value can be entered in the **Correct To** field.

Next Step

Task 3: Importing Domain Values from an Excel File

Task 3: Importing Domain Values from an Excel File

2/11/2019 • 2 minutes to read • Edit Online

In this task, you import values for the **State** domain from a worksheet of an Excel file.

1. Click **State** domain in the **Domain list**.
2. Ensure that the **Domain Values** tab is active in the right pane.
3. In the right pane, from the toolbar, click **down arrow** next to the **Import Values** button, and click **Import Valid Values from Excel**.

4. Click **Browse**, select **Suppliers.xls**, and click **Open**.
5. Select **StatesToImport$** for the **Worksheet**.

6. Click **OK** to close the **Import Domain Values** dialog box. You should see all the names of states you imported in the list. Notice that **Show Only New** option is automatically selected after importing. When you import values and you don't see the old values in the list, it is because this option is automatically enabled after importing. To see all the values, clear the check box. If you import the same set of values again, none of the values are imported as they already exist in the domain.

Next Step

Task 4: Setting Domain Rules

Task 4: Setting Domain Rules

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a rule for the **Contact Email** domain to verify if the email address ends with **@adventure-works.com**. See Creating a Domain Rule topic for more details on the page.

1. Click **Contact Email** in the **Domain list**.
2. Switch to the **Domain Rules** tab in the right pane.

3. In the right pane, click **Add a new domain rule** button on the toolbar (see the image) to add a rule.
4. Type **Email Validation** for the **rule name** and press **ENTER**. The **Active** check box should be checked by default. This control allows you to deactivate a rule temporarily.
5. In the **Build a Rule** pane, click **down arrow**, and select **Value ends with**.
6. Type **@adventure-works.com** in the text box and press **TAB**. You can add more conditions by clicking **Add a new condition to the selected clause** toolbar button in the **Build a Rule** pane.

7. Click **Run the selected domain rule on test data** button on the toolbar in the right pane to test the rule against sample data.

8. In the **Test Domain Rule** dialog box, click **Adds a new testing term for the domain rule** button on the toolbar.

9. Type **frank7@adventure-works.com** (a valid value) in the **Contact Email** column.
10. Repeat previous two steps to add **joe2@adventure-work.com** (an invalid value with no 's').
11. Click the last button (**Test the domain rule on all the terms**) on the toolbar to test the input data against the rule.

12. Notice that the first entry is shown as a valid item and the second one as an invalid item.

13. Click **Close** to close the **Test Domain Rule** dialog box.

Next Step

Task 5: Setting Term Based Relationships

Task 5: Setting Term Based Relationships

2/11/2019 • 2 minutes to read • Edit Online

In this task, you define a few term-based relations for values for the **Supplier Name** domain. A term-based relation enables you to make a correction to a term that is part of a value in a domain. It enables multiple values that are identical except for the spelling of a common part of them to be considered identical synonyms. For example, **Inc.** can be corrected to **Incorporated**. DQS uses these relations in the knowledge discovery, cleansing, or matching processes. See Create Term-based Relations for more details.

1. Select **Supplier Name** in the **Domain list**.
2. Switch to the **Term-Based Relationships** tab in the right pane.
3. Click **Add new relation** button on the toolbar to add a relation to the table.
4. Type **Co.** for the **Value** field and **Company** for the **Correct To** field.
5. Repeat the previous two steps for the following values:

VALUE	CORRECT TO
Corp.	Corporation
Inc.	Incorporated

Next Step

Task 6: Setting Synonyms

Task 6: Setting Synonyms

2/11/2019 • 2 minutes to read • Edit Online

In this task, you set two domain values, **USA** and **United States**, of the **Country** domain as synonyms with **United States** as the leading value. Since the **Use Leading Values** option was selected when creating the **Country** domain, any **USA** values for the **Country** domain will be output as **United States** (as United States is the leading value). See Change Domain Values for more details.

1. Select **Country** from the list of domains.
2. Switch to the **Domain Values** tab.
3. Click **Add new domain value** button on the toolbar.
4. Type **USA** for the value and press **ENTER**.
5. Multiselect **United States** and **USA** using CTRL or SHIFT keys, right-click the selected items, and then click **Set as Synonyms**. DQS groups these values and designate one of the values as the leading value that the other values are replaced with.

6. Notice that **United States** is set as the leading value. If you want USA to be the leading value, you can right-click on USA and select **Set as Leading** option. For this tutorial, we use **United States** as the leading value.

Next Step

Task 7: Creating a Composite Domain

Task 7: Creating a Composite Domain

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a composite domain, **Address Validation**, which comprises **Address Line**, **City**, **State**, and **Zip** domains. A composite domain lets you define a cross-domain rule that involves multiple domains in a rule. There are other advantages to a composite domain such as being able to parse a field value into multiple domains. For example, a value for a Full Name field can be parsed into separate First Name, Middle Name, and Last Name domains. In this tutorial, you only define a cross-domain rule. See Managing a Composite Domain for more details.

1. In the left pane, click **Create a composite domain** button on the toolbar.

2. Enter **Address Validation** for the **Composite Domain Name**.

3. From the domain list select **Address Line**, **City**, **State**, and **Zip** and click **right arrow** to add them to the **Domains in Composite Domain** list.

4. Click **OK** to close the dialog box.

Next Step

Task 8: Creating a Composite Domain Rule

Task 8: Creating a Composite Domain Rule

10/2/2018 • 2 minutes to read • Edit Online

In this task, you create a rule for the **Address Validation** composite domain. You define a cross-domain rule: if **City** is **Los Angeles**, **State** must be **CA** where **City** and **State** are two domains.

1. In the right pane, switch to the **CD Rules** tab.
2. Click **Add a new domain rule** from the toolbar.
3. Type **City-State Rule** for **Name** and press **ENTER**.
4. In the **Build a Rule** pane, select **City** in the domain list, and select condition **Value is equal to** and type **Los Angeles** for the value.
5. In the **Then** pane, select **State** in the domain list, and select **Value is equal to**, type **CA** for the value, and press **TAB**.

6. Click **Close** button at the bottom of the page to switch to the main page of DQS Client. You will publish the knowledge base in the next lesson. Notice that the knowledge base is in locked state (lock icon).

Next Step

Task 9: Configuring a Reference Data Service

Task 9: Configuring a Reference Data Service

2/11/2019 • 2 minutes to read • Edit Online

In this task, you configure DQS to use a Reference Data Service on Windows Azure Marketplace. In the next task, you will configure the **Address Validation** domain to use this service. At runtime, during cleansing activity, DQS passes the values of domains in the **Address Validation** domain to the service for cleansing. See Configure DQS to Use Reference Data for more details.

1. In the main page of **DQS Client**, in the **Administration** pane, click **Configuration**.
2. Ensure that **Reference Data** tab is active.
3. In the **Network Settings** area, type appropriate values in the **Proxy Server** and **Port** fields if you need to use a proxy server to connect to Internet.
4. Type your **Windows Azure Marketplace Account Key** for the **DataMarket Account ID** field.

5. Click **Validate** button next to the text box to validate the account ID.
6. Click **OK** on the message box.
7. Click **Close** at the bottom of the page to switch to the main page of DQS Client.

Next Task

Task 10: Configuring Composite Domain to Use Reference Data Service

Task 10: Configuring Composite Domain to Use Reference Data Service

2/11/2019 • 2 minutes to read • Edit Online

In this task, you configure the **Address Validation** composite domain to use the **Melissa Data - Address Check** service. At runtime, during cleansing activity, DQS passes the values of domains in the Address Validation domain to the service for cleansing. See Map Domain/Composite Domain to Reference Data for more details.

1. In the main page of **DQS Client**, click **Suppliers (Domain Management)** under **Recent Knowledge Bases** to launch the **Domain Management** page.

2. Select the **Address Validation** composite domain in the **list of domains**.

3. In the right pane, switch to the **Reference Data** tab.

4. Click **Browse** button on the toolbar.

5. On the **Online Reference Data Providers Catalog** dialog box, select **check box** next to **Melissa Data - Address Check**.

6. In the right pane, in the **Schema** section, map **Address Line** domain to the **Address Line (M)** schema item by using the drop-down list.

7. Click **Add Schema Entry (+)** button on the toolbar to create an entry in the list.

8. Map the following DQS domains by using the drop-down lists as shown in the following picture.

9. Click **OK** to close the dialog box.

Next Step

Task 11: Publishing the Knowledge Base

Task 11: Publishing the Knowledge Base

2/11/2019 • 2 minutes to read • Edit Online

In this task, you publish the knowledge base. A published knowledgebase can be used for cleansing or matching activity in data quality project.

1. Click **Finish** button at the bottom of the window.
2. Click **Publish** in the **SQL Server Data Quality Services** dialog box.
3. Click **OK** to close the message box.

Next Step

Task 12: Discovering Knowledge (Knowledge Discovery)

Task 12: Discovering Knowledge (Knowledge Discovery)

2/11/2019 • 2 minutes to read • Edit Online

In this task, you perform the **Knowledge Discovery** activity on **Supplier ID** and **Supplier Name** domains. In this scenario, the knowledge discovery process mainly imports values for these two domains.

In this tutorial, you started building knowledge base from scratch. You can also start creating a knowledge base by performing a knowledge discovery activity. When you click **Create a Knowledge Base** in the main page, DQS client takes you to a page with **Domain Management** activity selected for the activity. You can change the **activity** to **Knowledge Discovery** and then in the next page you can create domains as part of the knowledge discovery process. See Perform Knowledge Discovery for more details.

1. In the main page of DQS Client, in the **Recent Knowledge Base** section, click **right-arrow** next to the **Suppliers** knowledge base and click **Knowledge Discovery**. Alternatively, you can click **Open Knowledge Base**, select **Suppliers** from the **list of knowledge bases**, select **Knowledge Discovery** as **activity** and click **Next**.

2. Select **Excel File** for **Data Source**.

3. Click **Browse**, navigate and select **Suppliers.xls**, and click **Open**.

4. Select **Suppliers for Discovery** for **Worksheet**.

5. In the **Mappings** section, map **SupplierID** column from the **Excel** file to the **Supplier ID** domain and **Supplier Name** column to the **Supplier Name** domain by using **drop-down lists**. The Excel file has sample data for the **Supplier ID** and **Supplier Name** domains. In the discovery process, you can select the domains for which you want to discover the values. You can create domains on this page and then map the source columns to those domains. It is not uncommon to create domains during knowledge discovery activity instead of creating domains during domain management activity.

6. Click **Next** to switch to the **Discover** page.

7. On the **Discover** page, click **Start** to start the discovery process. Discovery is performed on the columns **SupplierID** and **Supplier Name** in the **Suppliers.xls** file. The **Supplier ID** and **Supplier Name** domains should be populated with the knowledge drawn from the discovery.

8. After the analysis has completed, review the **Source Statistics** in the **Profiler tab** at the bottom of the page. Notice that 10 new records with total 20 values (**SupplierID** and **Supplier Name** values from the **Excel worksheet**) were discovered. You will also see how many of the values are new, unique, new and unique, and valid. In the list box to the right, you can see more details for each domain involved in the discovery process. If you hover the mouse over the status bar in the Completeness column, you can see if there are any missing values in the columns in the source.

9. Click **Next** to switch to the **Manage Domain Values** page.

10. In the **Manage Domain Values** page, click **Supplier Name** domain from the list of domains.

11. In the right pane, right-click **Lazy Country Storex** (notice 'x' at the end), and select **Lazy Country Store**. DQS suggests this change after running the spell checker on the domain. By default, speller is enabled on the domains you create.

12. In the domain values list, confirm that the value **Lazy Country Storex** is set as an error (red **X** mark) with **Lazy Country Store** as the correction and also the **Lazy Country Store** is also added as a valid value.

13. Click **Finish**.

14. On **SQL Server Data Quality Services** dialog box, click **Publish**.

15. Click **OK** on the success message box.

    You have completed the first lesson of the tutorial.

Next Step

Lesson 2: Cleansing Supplier Data using the Suppliers Knowledge Base

Lesson 2: Cleansing Supplier Data using the Suppliers Knowledge Base

2/11/2019 • 2 minutes to read • Edit Online

In this lesson, you cleanse the supplier data in an Excel file by using the **Suppliers** knowledge base you have created in the first lesson. Data cleansing in DQS includes a **computer-assisted process** that analyzes how data conforms to the knowledge in a knowledge base, and an **interactive process** that enables you to review and modify results from the computer-assisted process. The data cleansing feature identifies incorrect data in your data source and then corrects or suggests corrections for the incorrect data. It also standardizes and enriches customer data by using domain values, leading values for synonyms, domain rules, term-based relations, and reference data. You can interactively approve or reject changes proposed by the computer-assisted process. See Data Cleansing for more details.

The computer-assisted process uses the following threshold values that you can configure using the Configuration option on the DQS Client main page.

- **Min score for suggestions:** The minimum score or confidence level that is used by DQS for suggesting replacement for a value.
- **Min score for auto corrections:** The minimum score or confidence level that is used by DQS for automatically correcting a value.

See Configure Threshold Values for Cleansing and Matching for details on how to configure these settings.

In this lesson, you perform the following tasks to cleanse the input data by using the Suppliers knowledge base.

1. Create a Data Quality Project for Cleansing, select the Suppliers knowledge base as the knowledge base to use to analyze and cleanse the source data in an Excel file, and select the Cleansing activity.
2. Map the Excel columns that you want to cleanse to appropriate DQS domains/composite domains in the knowledge base.
3. Run the computer-assisted cleansing activity. The computer-assisted process displays data quality information in the Data Quality Client that you can use to cleanse the data interactively.
4. View and manage the results of the cleansing activity. You can review the values that the computer-assisted process finds to be correct, incorrect but corrected, incorrect with a suggested change, or invalid. You can interactively approve or reject changes, correcting or overriding the suggestion from the computer-assisted process by using the Correct To field.
5. Export the results from the cleansing process to an Excel file.
6. Import the values from the cleansing project into domains to augment the knowledge in the knowledge base with new rules, values, corrections etc...

Next Step

Task 1: Creating a Data Quality Project

Task 1: Creating a Data Quality Project

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a Data Quality Project for cleansing the supplier data in an Excel file against the Suppliers knowledge base you created earlier in this tutorial.

1. In the **Data Quality Project** pane on the main page, click **New Data Quality Project**.

2. Type **Cleanse Supplier List** for the **name** of project.

3. Select **Suppliers** for the **Use Knowledge Base** field. You will be cleansing input supplier data against the Suppliers knowledge base you created earlier in this tutorial.

4. Ensure that **Cleansing** is selected as the **activity** at the bottom of the right pane and click **Next**.

Next Step

Task 2: Mapping Excel Columns to DQS Domains

Task 2: Mapping Excel Columns to DQS Domains

2/11/2019 • 2 minutes to read • Edit Online

1. In the **Map** page, select **Excel File** for **Data Source**.
2. Click **Browse**, select **Suppliers.xlsx**, and click **Open**.
3. Select **IncomingSuppliers$** for the **Worksheet**.
4. Map columns as shown in the following table and screenshot. When creating mappings for the **State** domain, click **Add a column mapping** button on the toolbar located just above the list.

> **TIP**
> You are not using **Supplier ID** column/domain for cleansing. You will use the **Supplier ID** domain later in the matching activity.

EXCEL COLUMN	DQS DOMAIN
Supplier Name	Supplier Name
ContactEmailAddress	Contact Email
Address Line	Address Line
City	City
State	State
Country (Click **+(Add a column mapping)** toolbar to add a row)	Country
Zip Code	Zip

5. As you have mapped all the individual domains within the **Address Validation** composite domain, the composite domain automatically participates in the cleansing process. Click **View/Select Composite Domains** button to see that the **Address Validation** composite domain is automatically selected, and then click **OK**.

6. Click **Next** to switch to the **Cleanse** page.

Next Step

Task 3: Cleansing Data against the Suppliers Knowledge Base

Task 3: Cleansing Data against the Suppliers Knowledge Base

2/11/2019 • 2 minutes to read • Edit Online

In this task, you run the computer-assisted cleansing process. DQS uses advanced algorithms and confidence levels based on the threshold values specified to analyze the data against the selected knowledge base, and then cleanse it. See Cleansing Data Using DQS (Internal) Knowledge for more details.

1. Click **Start** to start the computer-assisted cleansing process.

2. When the cleansing process is completed, review **statistics** in the **Profiler** tab. The Source Statistics provide the number of records processed, number of records that are found to be correct, number of records that DQS corrects, number of records that have changes suggested by DQS, and the number of records that are invalid. In the list box to the right, you can see the corrected values, suggested values, and the completeness (the extent to which the data is present) and accuracy (the extent to which the data can be used for intended purposes) of values for each domain involved in the cleansing process.

3. Click **Next** to switch to **Manage and View Results** page.

Next Step

Task 4: Managing and Viewing Results

Task 4: Manaing and Viewing Results

2/11/2019 • 4 minutes to read • Edit Online

In this task, you review the results of computer-assisted cleansing and also perform interactive cleansing on the supplier data. See Interactive Cleansing Stage for more details.

1. Select **Contact Email** domain from the list of domains.
2. Switch to the **Invalid** tab in the right pane. Notice that two email addresses that were missing character 's' at the end. These two emails that were found to be invalid by the domain rule that requires all email addresses to end with **@adventure-works.com** (with 's'). DQS uses the domain rule while cleansing to determine whether an email is a valid one or not. This tab displays the domain values that were either marked as invalid in the knowledge base or failed a domain rule. In this case, these values failed the domain rule (Email Validation).
3. In the **Correct To** column, type the right email address that end with **@adventure-works.com** (with 's').

4. Click **Approve** for both the records to approve both the changes. When you approve, the records move to the **Corrected** tab. Instead of approving each item separately, you can approve all the changes at once using the **Approve all terms** toolbar button.
5. Switch to the **New** tab in the right pane. The values on this tab are the values for which DQS does not have enough information in the knowledge base yet to determine whether the values are correct. Therefore, it cannot change or suggest changes to the domain values.
6. Review the values to confirm that all the emails end with **@adventure-works.com** and click **Approve all terms** on the toolbar. The approved values from this tab move to the **Correct** tab.
7. Select the **Country** domain from the list of domains.
8. Switch to the **Corrected** tab in the right pane and notice that **United State** value is automatically corrected to the **United States** with 's' at the end. This rule is not a rule you defined for the **Country** domain, but DQS is **83%** confident that the correct value is **United States**. The **Approve** button is automatically selected for all the **Corrected** items. You can override this behavior and reject a change.
9. Notice that **USA** is corrected to **United States** because they are synonyms and **United States** is the leading (preferred) value.

10. Notice that the **Approve** button is already selected for these corrected values. This behavior is the default

for the corrected values. You can reject a change and when you do so, the value moves to the **Invalid** tab.

11. Select **Supplier Name** from the list of domains.

12. Switch to the **Corrected** tab in the right pane.

Value	# Records	Correct to	Confidence	Reason	Approve	Reject
A. Datum Corp.	1	A. Datum Corporation	100%	Term based relation		
Lazy Country Storex	1	Lazy Country Store	100%	Domain value		
Let's Stop N Shopx	1	Let's Stop N Shop	86%	DQS cleansing		
LILA Supermarketx	1	LILA Supermarket	86%	DQS cleansing		
Richter Supermarkt	1	Richter Supermarket	88%	DQS cleansing		

    a. Notice that **A. Datum Corp.** is corrected to **A. Datum Corporation** and the **Reason** is set to **Term based relation. A. Datum Corporation** is a known domain value to DQS because it was discovered during the knowledge discovery process. Therefore, DQS is **100% confident** about this correction.

    b. Notice that **Lazy Country Storex** is corrected to **Lazy Country Store**, **Confidence Level** is set to **100%**, and the **Reason** is set to **Domain Value**. During the knowledge discovery process, you set **Lazy Country Storex** as an error with **Lazy Country Store** as the **correction**, so DQS is **100% confident** about making this correction.

    c. DQS is not familiar with the other values in the list, but it found the corrections for these values using the **Spell Checker** and proposes the appropriate corrections. DQS is **not 100%** confident about these corrections, but the confidence level is above 80%, which is the threshold level for making corrections, so DQS proposes the corrections.

13. Notice that the **Approve** is automatically enabled for all the values. You can override the corrected value or reject the change as appropriate. By default the **Approve** button is selected for all the values on the **Corrected** tab.

14. Switch to the **New** tab.

15. Notice that **Corp.** is corrected to **Corporation**, **Co.** is corrected to **Company**, and **Inc.** is corrected to **Incorporated**. For example, **Consolidate Inc.** is corrected to **Consolidate Incorporated** and **Consolidated Co.** is corrected to **Consolidated Company**, and **Frabrikam Corp.** is corrected to **Fabrikam Corporation**. You can see that **term-based relation** is mentioned as the reason. These changes are proposed by using the term-based relations you defined during the domain management activity. You can change the **Correct To** values manually here.

16. Scroll the list to see **Hunxgry Coyote Store** with a red squiggly line. Right-click on it and click **Hungy Coyote Store** (with no 'x'). The **Correct To** column should be automatically populated with **Hungry Coyote Store**. You can also manually type a value in the Correct To column.

17. Click **Approve all terms** from the toolbar. The domain values with the **Correct To** value specified move to the **Corrected** tab and the new values with no associated **Correct To** values move to the **Correct** tab.

18. Select the **Address Validation** composite domain from the domain list.

19. In the right pane, switch to the **Correct** tab. You should see the addresses that are found to be correct by the **Melissa Data - Address Check** DQS service on the **Azure Marketplace**.

20. Switch to the **Corrected** tab.

21. Notice that **State** for the record that has **City** as **Los Angeles** is set to **CA** now. Notice in the **Reason** field is that **Corrected by Rule 'City-State Rule'**.

22. Notice that the **Approve** radio button is already selected for this item in the list. This is the default behavior for items on the **Corrected** tab.

23. Switch to the **Suggested** tab. Review the changes suggested by the **Melissa Data - Address Check** service.

24. **Click Approve all terms** on the toolbar button and click **OK** on the **Confirmation** message box.

25. Click **Next** to switch to the **Export** page.

Next Step

Task 5: Exporting Cleansing Results to an Excel File

Task 5: Exporting Cleansing Results to an Excel File

2/11/2019 • 2 minutes to read • Edit Online

In this task, you export results from the cleansing activity to an Excel file. See Export Stage topic for more details.

1. In the right pane, select **Excel** for the **Destination Type**.
2. Click **Browse**, specify the output file name as **Cleansed Supplier List.xls**, and then click **Open**.
3. Select **Data Only** for the **Output** format to export just the cleansed data. The second option, **Data and Cleansing Info**, lets you export cleansing activity details along with the cleansed data. The **Standardize Format** option lets you apply any output formats you define on a domain to the values of that domain. You have not defined an output format on any domain in the tutorial.

4. Click **Export** to export the data. Do not click **Finish** yet.
5. Click **Close** on the **Exporting** dialog box.
6. Click **Finish** to finish the activity. If you forgot to export results before clicking **Finish**, click **Open Data Quality Project** in the main page of **DQS Client**, select **Cleanse Supplier List** from the list of projects, and click **Next** at the bottom of the screen to get to the **Export** stage of cleansing process again. You can also switch to **Manage and View Results** tab by clicking **Back** button.
7. Open the **Cleansed Supplier List.xls** and do the following:
   a. Ensure that there are no email addresses that end with adventure-work.com (without character 's') by searching for adventure-work.com in the worksheet.
   b. See that there is no **USA** value in the **Country** column.
   c. Search for **Los Angeles** and see that the **State** is set to **CA**.
   d. Confirm that there are no terms **Co.**, **Corp.**, and **Inc.**.
   e. Delete the **Address Validation** column from the spreadsheet and save the excel file. This additional column corresponds to the Address Validation composite domain.

Next Step

Task 6: Importing Values from the Cleanse Supplier List Project

Task 6: Importing Values from the Cleanse Supplier List Project

2/11/2019 • 2 minutes to read • Edit Online

In this task, you import the data quality knowledge gathered during the cleansing process. See Importing Cleansing Project Values into a Domain topic for more details. You also export the knowledge base into a DQS file before publishing the updated **Suppliers** knowledge base.

1. In the main page of **DQS Client**, click **right-arrow** next to **Suppliers** under **Recent Knowledge Bases** and click **Domain Management**.
2. Click **Contact Email** in the list of domains, and switch to the **Domain Values** tab in the right pane.
3. Click **down arrow** next to the **Import Values** icon on the toolbar and click **Import Project Values**.

4. In the **Import Project Values** dialog box, select the **Cleanse Supplier List** project, and click **OK**.
5. Notice that all the emails are imported along with the two corrections you did during interactive cleansing. Scroll to see the two corrections.

VALUE	CORRECT TO
bobby0@adventure-work.com	bobby0@adventure-works.com
tad0@adventure-work.com	tad0@adventure-works.com

6. Repeat the previous step of importing project values for the **Country** domain and notice that a new entry is added for correcting **United State** to **United States** (with 's').

VALUE	CORRECT TO
United State	United States

7. To see the old domain values, clear **Show Only New** checkbox.
8. Repeat the previous step of importing project values for the **Supplier Name** domain. By default, after importing, you will only see the new values. To see all the values, clear **Show Only New** check box. You have enriched the **Suppliers** knowledge base with what you learned from the cleansing activity. The stronger the knowledge base is, the better the cleansing results are.

> **NOTE**
> It is not possible import values for a composite domain.

9. Click **Export Knowledge Base** icon on the toolbar and then click **Export Knowledge Base**.

10. Navigate to the Tutorial folder, type **Suppliers.dqs** for the **file name**, and click **Save**. You can use this DQS file to create a new knowledge base based on it.
11. Click **OK** to close the **Export Knowledge Base - Suppliers** message box.
12. Click **Finish** to finish the activity.
13. Click **Publish**.
14. Click **OK** on the message box.

Next Step

Lesson 3: Matching Data to Remove Duplicates from Supplier List

Lesson 3: Matching Data to Remove Duplicates from Supplier List

2/11/2019 • 2 minutes to read • Edit Online

You prepare the knowledge base for performing matching activity by creating a matching policy in the knowledge base. There can be only one matching policy in a knowledge base. A matching policy consists of one or more matching rules. A rule identifies the domains that are involved in the matching process, and specifies the weight that each domain value carries in the matching judgment. You specify in the rule whether domain values have to be an exact match or can be similar, and to what degree of similarity. You also specify whether a domain match is a prerequisite for the matching process. You can test each rule separately and test the entire policy against sample data. The testing process displays records whose matching scores are greater than the **Min record score** threshold specified in the DQS configuration in a cluster (group). You can continue to tweak the rules in the policy until you are satisfied.

After defining the policy, you create a Data Quality Project to run the matching activity. The matching project applies the matching rules in the matching policy to the data source to be assessed. This process assesses the likelihood that any two rows are matches. When DQS performs the matching analysis, it creates clusters of records that DQS considers matches. DQS randomly identifies one of the records as a pivot record. You can verify and reject any record that is not an appropriate match for the cluster. See Create a Matching Policy topic for more details.

In this lesson, you perform a matching activity to remove duplicates from the supplier list. First, you create a matching policy with one rule to identify duplicates in the supplier list and publish the policy to the knowledge base. Next, you create and run a data quality project for matching. Finally, you export the results from the matching activity to an Excel file that you use later in uploading data to Master Data Services (MDS).

Next Step

Task 1: Defining a Matching Policy

Task 1: Defining a Matching Policy

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a matching policy with one rule in it. The rule will have one prerequisite: **Supplier ID**, which means that the Supplier IDs must match before using the other domains in the rule. The rule uses two other domains: **Supplier Name** with **Similarity** value set to **70%** and **Contact Email** with **Similarity** value set to **30%**.

1. In the main page of **DQS Client**, click **right-arrow** next to **Suppliers** knowledge base, and select **Matching Policy**.

2. On the **Map** page, select **Excel File** for **Data Source**.

3. Click **Browse**, ensure that filter is set to **Excel Workbook**, and select **Cleansed Supplier List.xls** file that you exported after you perform the cleansing activity.

> **NOTE**
>
> At the end of this activity, you cannot export results because this activity is primarily focused on defining a matching policy. You will create a Data Quality Project for the Matching activity and run it to remove duplicates from the supplier list by using this matching policy in the next lesson.

4. Map **SupplierID** column to **Supplier ID** domain, **Supplier Name** column to **Supplier Name** domain, **ContactEmailAddress** column to **Contact Email** domain. You only need to map source columns to domains that you want to use in defining the matching policy. In this case, you are making the Supplier ID, Supplier Name, and Contact Email domains available for the matching policy activity.

5. Click **Next** to move to the **Matching Policy** page where you will be defining a matching policy with one rule in it.

6. Click **Create a matching rule** button on the toolbar to create a rule in the policy.

7. In the **Rule Details** pane on the right, enter **Remove Duplicate Suppliers** for the **Rule name**.

8. Click **Add a new domain element** in the toolbar in the right pane.

9. Select **Supplier ID** for the **domain** and select the **Prerequisite** check box. Notice that **Similarity** is automatically set to **Exact**. By setting **Supplier ID** as the **Prerequisite**, you specify that the values for this field in the two records must return a 100% match, else the records are not considered a match and the other clauses in the rule are disregarded.

10. Click **Add a new domain element** from the toolbar again.
11. Select **Supplier Name** domain, select **Similar** for **Similarity**, and Type **70** for the **Weight**. Here, you are specifying that supplier names do not need to be identical but can be similar for the records to be considered as a match. The weight indicates the contribution of this field's score to the overall matching score.
12. Repeat previous two steps to add **Contact Email** domain with **30** for the **Weight**.
13. Notice that the **min matching score** is set to **80%**, which is the value you see in the **General** tab of the **Configuration** page of **DQS Administration**. You can only increase this score above this threshold value here.
14. Notice that **Overlapping Clusters** option is selected. With this option, a record can show up in multiple clusters. If you change the setting to Non-Overlapping Clusters, the clusters that have common records are combined into one single cluster.
15. The **Start** button on this page allows you to test each rule in the policy separately, whereas, the Start button in the next page allows you to test entire policy (all the rules in the policy).
16. Click **Next** to switch to the **Matching Results** page.

Next Step

Task 2: Testing and Publishing the Matching Policy

Task 2: Testing and Publishing the Matching Policy

2/11/2019 • 2 minutes to read • Edit Online

In this task, you test and publish the **Remove Duplicate Suppliers** matching policy.

1. In the **Matching Results** page, click **Start** to test the entire policy. In your case, you have only rule in the policy, so the results from testing the rule and the policy should be the same.

2. Review all the matched records and their matching score in the list box. A record that has a **Green** icon associated with it is a duplicate of the pivot record that precedes it. Here are couple of examples:

   a. The record with **Record ID: 1000005** is a match of the record with **Record Id: 1000004** with **Score: 100%** because both the records have the same values for **SupplierID (prerequisite)**, **Supplier Name**, and **ContactEmailAddress columns**. DQS randomly picks a record as the pivot record for a cluster.

   b. The record **1000023** is a match of the record **1000022** with the matching score: 93% because the two records have the same values for **SupplierID (prerequisite)** and **Supplier Name** columns, but different values for the **ContactEmailAddress** column.

   c. Scroll to the bottom of the list to see two records with records IDs: **1000051** and **1000052**. Record **1000052** is considered a match with matching score **91%** because the two records have the same values for the **SupplierID** and **ContactEmailAddress** columns, but different values for the **Supplier Name** column.

Filter: Matched

Rule	Record Id	Cluster	Score	SupplierID	Supplier Name	ContactEmailAddress
	1000004	1000004		S5	Richter Supermarket	sachinl1@adventure-works.com
	1000005	1000004	100%	S5	Richter Supermarket	sachinl1@adventure-works.com
	1000006	1000006		S6	Tortuga Market	karin0@adventure-works.com
	1000007	1000006	100%	S6	Tortuga Market	karin0@adventure-works.com
	1000017	1000017		S16	Trail's Head Gourmet Provs	kelly1@adventure-works.com
	1000018	1000017	100%	S16	Trail's Head Gourmet Provs	kelly1@adventure-works.com
	1000019	1000017	100%	S16	Trail's Head Gourmet Provs	kelly1@adventure-works.com
	1000020	1000020		S19	Trailer Head	carole1@adventure-works.com
	1000021	1000020	100%	S19	Trailer Head	carole1@adventure-works.com
	1000022	1000022		S21	Hungry Coyote Store	gary7@adventure-works.com
	1000023	1000022	93%	S21	Hungry Coyote Store	tad0@adventure-works.com

3. Right-click on any matched record (with green icon) and click **View Details** to see more details about the matching such as contribution of each field score to the overall matching score.

4. Click **Close** to close the **Matching Score Details** dialog box.

5. Click **Matching Results** tab at the bottom of the page. This tab gives you detail such as number of matched records, number of unmatched records, number of clusters with matched records, the average cluster size, minimum cluster size, and maximum cluster size. See Create a Matching Policy for more details. You cannot export results from this activity. You are just defining a matching policy by using the sample data to test rules and the policy against the sample data.

6. Click **Finish** to finish creating the matching policy.

> **NOTE**
>
> You have defined the matching policy here; therefore you cannot export results to an output file. You basically used a sample input file, created rules, and tested the rules and policy against the sample data with the goal of defining the policy.

7. In the SQL Server Data Quality Services dialog box, click **Publish** and click **OK** on the message box. Now, the matching policy you defined is published into the **Suppliers** Knowledge Base. You can use the knowledge base to run the matching process against an input file to identify and remove duplicates.

Next Step

Task 3: Creating and Running a Data Quality Project for Matching

Task 3: Creating and Running a Data Quality Project for Matching

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a Data Quality Project for the matching activity and run the matching process on cleansed supplier data to remove any duplicates in the data.

1. On the main page of **DQS Client**, click **New Data Quality Project**.
2. Type **Remove Supplier Duplicates** from the **Name of the project**.
3. Select **Suppliers** from the list of KBs for the **Use Knowledge Base** field. You have created a matching policy in this knowledge base in the previous lesson.
4. Select **Matching** from the **list of activities** from the bottom-right pane.

5. Click **Next**.
6. In the **Map** page, select **Excel File** for **Data Source**.
7. Click **Browse** and select **Cleansed Supplier List.xls**, which is the output file from the cleansing activity.
8. Map **SupplierID** source column to the **Supplier ID** domain, **Supplier Name** column to **Supplier Name** domain, and **ContactEmailAddress** column to **Contact Email** domain.
9. Click **Next** to switch to the **Matching** page.
10. Click **Start** to start the matching process. You should see results similar to those from the previous task because you used the same input file for defining the matching policy.
11. Review all the matched records and their matching score in the list box. The results should be same as the ones you saw in the previous task. See the steps in the previous task to analyze the results from this matching activity.
12. Click **Next** to switch to the **Export** page.

Next Step

Task 4: Exporting the Results from Matching Activity to an Excel File

Task 4: Exporting the Results from Matching Activity to an Excel File

2/11/2019 • 2 minutes to read • Edit Online

In this task, you export the results from the matching activity to an Excel file.

1. In the **Export** page, select **Excel File** for the **Destination Type**.
2. Select **Survivorship Results** option. In the survivorship process, DQS determines a survivor record for each cluster based on the **Survivorship Rule** you selected.
3. Click **Browse** and navigate to the folder where you want to store the output file.
4. Type **Cleansed and Matched Suppliers.xls** for the name and click **Open**.
5. Confirm that **Pivot Record** is selected for the **Survivorship Rule**. When you select this option, the pivot record for each cluster is picked for the output from a cluster. The other options for the Survivorship Rule are:
   a. **Most complete record:** Survivor record is the one with the largest number of populated fields.
   b. **Longest record:** Survivor record is the one with the largest number of terms in source fields.
   c. **Most complete and longest record:** Survivor record is the one with the largest number of populated fields, and has the largest number of terms in each field.

6. Click **Export** to export the results to an Excel file.
7. Click **Close** to close the **Matching Export** dialog box.
8. Click **Finish** to finish the matching activity.
9. Open the **Cleansed and Matched Suppliers.xlsx** file and confirm that you do not see any duplicates (SupplierID).

Now, you have supplier data that has been cleansed and matched to remove duplicates.

Next Step

Lesson 4: Storing Supplier Data in MDS

Lesson 4: Storing Supplier Data in MDS

2/11/2019 • 3 minutes to read • Edit Online

Master Data Services (MDS) is the SQL Server solution for master data management. Master data management (MDM) describes the efforts made by an organization to discover and define nontransactional lists of data.

Models are the highest level of organization in Master Data Services and organize the structure of your master data. Your MDS implementation can have one or many models where each model groups similar data. In general, master data can be categorized in one of four ways: people, place, things, or concepts. For example, you can create a Product model to contain product-related data or a Customer model to contain customer-related data. See Models (Master Data Services) for more details.

A model can contain one or more entities. Each entity has attributes (columns) and members (rows). Each row contains the master data. In this lesson, you create a Suppliers model with two entities named Supplier and State. The Supplier entity will have the following attributes: Code, Name, Contact First Name, Contact Last Name, Contact Email Address, Address Line, City, State, Zip, and Country. See Attributes (Master Data Services) for more details about attributes in general. The Code and Name attributes correspond to the SupplierID and Supplier Name columns in the Cleansed and Matched Suppliers Excel file.

A domain-based attribute is an attribute with values that are populated by members of another entity. Domain-based attributes prevent users from entering attribute values that are not valid. An attribute value can be selected only from the drop-down list that is populated by another entity. In this tutorial, the State attribute of the Supplier entity is a domain-based attribute with values from the State entity. You can only change the value of the State attribute of the Supplier entity to one of the values in the State entity. See Domain-Based Attributes for more details.

A derived hierarchy in MDS is derived from the domain-based attribute relationship in the model. In this tutorial, you create a derived hierarchy between the Supplier entity and the State entity. After you create the derived hierarchy, you will see a list of states in the Browser of Master Data Manager. When you click a state in the list, you see the suppliers in that state in the right pane. You will be creating a derived hierarchy later based on this relationship. See Derived Hierarchies for more details.

You built a knowledge base in DQS and used it to cleanse and match supplier data and stored the results in the Cleansed and Matched Supplier Data.xls file. In this lesson, you upload the cleansed and matched data into MDS. DQS only contains knowledge about the data (metadata) whereas MDS stores the data itself (master set). For example: DQS may have knowledge about several suppliers but MDS only maintains the suppliers that a company uses.

In this lesson, you perform the following tasks:

1. Create the **Suppliers** model in **MDS** by using the **Master Data Manager Web Application**.
2. Open **Cleansed and Matched Supplier Data.xls** in Excel and use the **MDS Add-in for Excel** to create an entity named **Supplier** and upload the data to MDS.
3. Verify that the data is created in MDS by using the **Master Data Manager**.
4. Create an entity named **State** and update the **State** attribute of **Supplier** entity to be a domain-based attribute depending on the **State** entity. You do this all using the **MDS Add-in for Excel**.
5. Verify that the domain-based attribute is created by using **Master Data Manager** and update the values for the **Name** attribute of the **State** entity.
6. View the updates you made using **Master Data Manager** in **Excel**.

7. Load values from the **State** entity into **Excel** and add a value, and verify the addition by using **Master Data Manager**.

8. Create and use a derived hierarchy by using the domain-based attribute relationship between the **Supplier** entity and the **State** entity (the State attribute of the Supplier entity is of type State entity) by using **Master Data Manager**.

Next Step

Task 1: Creating Suppliers Model using Master Data Manager

Task 1: Creating Suppliers Model using Master Data Manager

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a model named **Suppliers** in MDS using **Master Data Manager**.

1. Navigate to http://localhost/mds to launch **Master Data Manager**. Replace the URL if you have configured Web Application with a different name or on a different a web site.

2. Click **System Administration** in the **Administrative Tasks** section.

3. If you do not see the **Add Model** page, hover mouse over **Manage** on the menu bar, click **Models**, and then click **Add Model (+)** toolbar button to create a model.

4. Enter **Suppliers** for **Model name**.

5. Clear **Create entity with same name as model** option. You will create an entity later using the **MDS Add-in for Excel**.

6. Click **Save Model** button on the toolbar.

Next Step

Task 2: Uploading Supplier Data to MDS using MDS Add-in for Excel

Task 2: Uploading Supplier Data to MDS using MDS Add-in for Excel

2/11/2019 • 2 minutes to read • Edit Online

In this task, you publish the cleansed and supplier data to **MDS** using the **MDS Add-in for Excel**. You create an entity named **Supplier** in the **Suppliers** model you created in the previous lesson. The entity will have an attribute for each column in the Excel file. The Code and Name attributes of the Supplier entity correspond to the **SupplierID** and **Supplier Name** columns in Excel.

1. Open **Cleansed and Matched Suppliers.xls** in **Excel**.
2. Press **CTRL+A** to select entire data. It is **important** that you select the entire data in the spreadsheet.
3. Click **Master Data** on the menu bar.
4. Click **Create Entity** button on the ribbon.

5. In **Manage Connections** dialog box, if you do not see the connection to **local MDS server** under **Existing connections**, do the following:
   a. Select **Create a new connection**, and click **New** button.
   b. In the **Add New Connection** dialog box, type **Local MDS Server** for **Description** and **http://localhost/MDS** for **MDS server address**, and click **OK** to close the dialog box.
6. In **Manage Connections** dialog box, select **Local MDS Server** (http://localhost/MDS), click **Test** to test the connection. Click **OK** on the message box.
7. Click **Connect** to connect to the MDS server.

8. In the **Create Entity** dialog box, select **Suppliers** for the **Model**.

9. Ensure that **VERSION_1** is selected for **Version**.

10. Enter **Supplier** for **New entity name**.

11. Select **SupplierID** for **the column that contains a unique identifier** field (you can also generate a code automatically). You are essentially mapping the **SupplierID** column in **Excel** to the **Code** attribute of **Supplier** entity.

12. Select **Supplier Name** for **the column that contains names** field. You are essentially mapping the **Supplier Name** column in **Excel** to the **Name** attribute of the **Supplier** entity. The **Code** and **Name** attributes are mandatory attributes for an entity in MDS.

13. Click **OK** to create the entity on MDS, publish the master data to the entity, and close **Create Entity** dialog box.

14. Now, you should see a new sheet titled **Supplier**, which is the name of the entity, added to your Excel spreadsheet and at the top of the worksheet you should see that the worksheet is connected to the MDS server. Notice that the original worksheet (titled **Sheet1**) still exists.

15. Keep **Excel** open.

Next Task

Task 3: Verifying the Data in Master Data Manager

Task 3: Verifying the Data in Master Data Manager

2/11/2019 • 2 minutes to read • Edit Online

In this task, you verify that the **Supplier** entity is created on **MDS** using **Master Data Manager Web Application**.

1. If **Master Data Manager** is already open, click **SQL Server 2012 Master Data Services** at the top to navigate to the home page. Otherwise, navigate to http://localhost/mds to launch **Master Data Manager**.

2. Select **Suppliers** for **Model**, and click **Explorer**.

3. Review the data stored on MDS. If you do not see the data, confirm that you selected **Suppliers** for the **Model** on the home page before launching **Explorer**. You can add to or delete from the supplier list by using **Add Member** and **Delete Member** buttons on the toolbar.

Next Step

Task 4 (Optional): Combining, Matching, and Publishing New Set of Data

Task 4 (Optional): Combining, Matching, and Publishing New Set of Data

2/11/2019 • 4 minutes to read • Edit Online

Over time, you will want to add more data to the MDS repository. Before adding data, it can be useful to compare the new data to the data that's already managed in MDS, to ensure that you are not adding duplicate or inaccurate data. In the Master Data Services Add-in for Excel, you can combine data from two worksheets and the compare the data to identify and remove duplicates before publishing the data to MDS. The matching feature of MDS Excel Add-in uses the DQS matching functionality to identify matches in the data. In this task, you will combine data from a two worksheets into one and then perform the matching activity to identify and remove duplicates before publishing to MDS. See Data Quality Matching in the MDS Add-in for Excel and Combine Data topics for more details.

1. Launch new instance of **Excel**. Click **Start**, point to **Run**, type **Excel**, and click **OK**.
2. Switch to the **Master Data** tab by clicking **Master Data** on the menu bar.
3. Click **Connect** on the ribbon in the **Connect and Load** group to connect to the **MDS server**. You have configured this connection earlier in this lesson.

4. You should see the **Master Data Explorer** pane to the right. If you do not see the Master Data Explorer, click **Show Explorer** button on the ribbon.
5. In the **Master Data Explorer** Window, select **Suppliers** in the drop-down list for the **Model**. You should see that the model has one entity: **Supplier**.

6. Double-click **Supplier** in the entity list to load the entity members into the Excel worksheet.
7. Click **Sheet2** at the bottom to switch to the **Sheet2** tab. If you do not see **Sheet2**, add a new worksheet.
8. Open **Suppliers.xls** file (the original input file that is included in the tutorial files) and copy all (three) rows from the **CombineAndCleanse** worksheet to **Sheet2**.
9. Switch back to the **Supplier** sheet in the **Book 1 - Microsoft Excel** (not the **Cleansed and Matched Supplier List** Excel) that is connected to **MDS**.
10. Click **Master Data** on the menu bar.
11. Click **Combine Data** on the ribbon. You will see the **Combine Data** dialog box.
12. In the **Combine Data** dialog box, click the button next to **Range to combine with MDS data** text box as

shown in the following image.

13. You should see the shrunken dialog box now. Now, click **Sheet2** to switch to the **Sheet2** tab that has the new supplier data with 4 rows (including one header row).

14. In the **Sheet2**, select **all rows including the header row** (even if they seem to be already selected). You should see the **Range to combine with MDS data** is automatically updated.

15. Switch back to the **Suppliers** tab without closing the **Combine Data** dialog box.

16. Click the **button** next to the **text box**. You should see that the dialog box is expanded now. You should see all the mappings between columns of the **Supplier** MDS **entity** to **Excel** columns are automatically populated.

17. Ensure that **Code** entity column is mapped to the **SupplierID** column in the worksheet and **Zip Code** entity column is mapped to the **Zip Code** column in the worksheet.

18. On the **Combine Data** dialog box, click **Combine**.

19. Confirm that three data rows are added to the bottom of the worksheet and they should be color coded.

Northwind Traders	98052	Jay	Hamlin	jay0@adventure-works.com	9825 Willows Road NE	Redmond	WA	USA	External
Litware Incorporated	84043	Jamie	Reding	jamie0@adventure-works.com	3400 North Ashton Blvd	Lehi	UT	USA	External
A. Datum Corporation	30082	Frank	Miller	frank7@adventure-works.com	1125 Sanctuary Parkway	Alpharetta	GA	USA	External

20. Click **Math Data** on the ribbon to identify duplicates. This feature uses the matching functionality of DQS.

21. In the **Match Data** dialog box, select **Suppliers** for **DQS Knowledge Base**.

22. Map worksheet columns to domains as shown in the following table.

WORKSHEET COLUMN	DOMAIN
Code (you uploaded Supplier ID as the Code for the Supplier entity in MDS)	Supplier ID

WORKSHEET COLUMN	DOMAIN
Name (you uploaded Supplier Name as the Name for the Supplier entity to MDS)	Supplier Name
ContactEmailAddress	ContactEmail

23. Select **Prerequisite** for the **Code** column mapping.
24. Enter **70%** as the **weight** for **Supplier Name** and **30%** as the **weight** for **Contact Email** as shown in the image.
25. Click **OK**.
26. The matching process should identify one duplicate for the supplier with **Code: S1**.

1	MDS Connection: Local MDS Ser	MDS Connection: Local MDS Server(http://localhost/MDS)		Model: Suppliers	Version: VERSION_1	Entity: Supplier	Retrieved: 10/17/2012 4:01:08 PM						
2	CLUSTER_ID	Name	Code	Contact First I	Conta	ContactE	Address Lin	City	State	Zip Coc	Countr	SOURC	
3	1000000	A. Datum Corporation	S1	Frank	Miller	frank7@adv	1125 Queensg	Smyrna	GA	30082	United Sta	MDS	
4	1000000	A. Datum Corporation	S1	Frank	Miller	frank7@adv	1125 Sanctuan	Alpharett	GA	30082	USA	External	

27. Select the **duplicate row (orange)**, right-click, and click **Delete** to delete the row.
28. Delete the **CLUSTER_ID** column since you don't need it anymore.
29. Click **Publish** to publish the other two new records with **Codes S66** and **S57** to MDS.
30. In the **Publish and Annotate** dialog box, add an **annotation**, and click **Publish**.
31. Switch to the **Master Data Manager Web application**.
32. On the home page, ensure that **Suppliers** is selected for the **Model**, and click **Explorer**. If you already have the **Explorer** open, refresh the internet browser.
33. **Sort** the list by **Code** and look for records with **S57** and **S66** as codes. You can also use the **Filter** button on the toolbar to search for a specific record in the list.
34. Now, close **Book1 - Microsoft Excel** window without saving the file.

Next Step

Task 5: Creating a Domain-Based Attribute from Excel

Task 5: Creating a Domain-Based Attribute from Excel

2/11/2019 • 2 minutes to read • Edit Online

In this task, you convert the **State** attribute of the **Supplier** entity as a **domain-based attribute**. After you configure the State attribute to be a domain-based one and publish it to MDS, a new entity named **State** will be created on MDS server with all the values in the column and the **State** attribute of the **Supplier** entity will be populated with values from the **State** entity. Now, the **Suppliers** model should have two entities: **Supplier** and **State** where the **State** attribute of the **Supplier** entity is a domain-based attribute that depends on **State** entity.

1. Switch to **Excel** window that has **Cleansed and Matched Suppliers.xlsx** open.
2. Click **Refresh** button on the ribbon to get the latest updates from MDS. You should see the two more records if you have performed the optional **Task 4**.
3. Click column name **State** (cell **I1**) in the **header row**.

4. Click **Attribute Properties** on the ribbon.
5. In the **Attribute Properties** dialog box, select **Constrained list (Domain-based)** for the **Attribute type**.
6. Type **State** for the **New entity name** and click **OK**.

7. Now, in Excel, you should see **down arrow** when you click any value in the **State** column. You can change the value by using the drop-down list if you need.

Next Step

Task 6: Verify that the Domain-Based Attribute is Created using Master Data Manager

Task 6: Verify that the Domain-Based Attribute is Created using Master Data Manager

2/11/2019 • 2 minutes to read • Edit Online

In this task, you verify that the **State** entity is created in **MDS** and the **State** attribute of the **Supplier** entity is a domain-based attribute that depends on the **State** entity by using **Master Data Manager**.

1. Switch to the **Master Data Manger** web application.
2. Click **SQL Server 2012 Master Data Services** at the top to get to the home page.
3. Ensure that **Suppliers** model is selected and click **Explorer**. You could refresh the page if you already had **Explorer** open.
4. Hover your mouse over **Entities** on the menu bar and notice that now there are two entities: **Supplier** and **State**.

5. Click **State** if the entity is not open already.
6. Select **GA** from the list.
7. In the **Details** pane to the right, change the **Name** to **Georgia** in the **right pane**, and click **OK**.
8. Repeat the previous steps for other states.

CODE	NAME
CA	California
CO	Colorado
IL	Illinois
DC	District of Columbia
FL	Florida
AL	Alabama
KY	Kentucky
MA	Massachusetts
AZ	Arizona

CODE	NAME
MI	Michigan
MN	Minnesota
NJ	New Jersey
NV	Nevada
NY	New York
OH	Ohio
OK	Oklahoma
OR	Oregon
PA	Pennsylvania
SC	South Carolina
KS	Kansas
TN	Tennessee
TX	Texas
UT	Utah
VA	Virginia
WA	Washington
WI	Wisconsin
HI	Hawaii
MD	Maryland
CT	Connecticut

9. Select any of the state entries and click **View Transactions** from the Toolbar. You should see the transaction for the update you just made is in the list of transactions.
10. Hover the mouse over **Entities** menu and click **Supplier**.
11. Now, notice that a value for the **State** field can be changed in the **Details** pane by using the drop-down list. You can also see that, in the list to the left and in the drop-down list in the **Details** pane, code is displayed first and then the name in curly braces. You can also change any other value in the **Details** pane.

Next Step

Task 7: Viewing Updates Made using Master Data Manager in Excel

Task 7: Viewing Updates Made using Master Data Manager in Excel

2/11/2019 • 2 minutes to read • Edit Online

In this task, you verify that you see the updated data from MDS in Excel.

1. Now, switch to the excel window that has **Cleansed and Matched Suppliers** spreadsheet open.
2. Click **Refresh** button on the ribbon.

3. Notice that **names** show up (California, New York etc...) for the **State** field along with their **codes**.

Next Step

Task 8: Adding a New Value for State Entity in Excel

Task 8: Adding a New Value for State Entity in Excel

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add a value for the State entity in Excel and publish the change to the MDS server.

1. Add a **work sheet** in Excel by clicking the new tab at the bottom.

2. In **Excel**, click the **Master Data** tab on the menu, and then click **Show Explorer** on the ribbon.

3. In the **Master Data Explorer**, select **Suppliers** for **Model**. You should see two entities: **Supplier** and **State** in the entity list.

4. Double-click **State** in the list. All the members of the **State** entity from MDS should be displayed in the worksheet.

5. Now, add a row at the end with the following values: **North Carolina** for **Name** and **NC** for **Code**. The color coding differentiates any new/updated records from the other records.

Massachusetts	MA
Maryland	MD
Michigan	MI
Minnesota	MN
New Jersey	NJ
Nevada	NV
New York	NY
Ohio	OH
Oklahoma	OK
Oregon	OR
Pennsylvania	PA
South Carolina	SC
Tennsesse	TN
Texas	TX
Utah	UT
Virginia	VA
Washington	WA
Wisconsin	WI
North Carolina	NC

6. Click **Publish** on the ribbon to publish the change to MDS.

7. On the **Publish and Annotate** dialog box, notice that the **Use same annotation for all changes** is selected. You can enter a single annotation for all the changes here.

8. Select **Review changes and provide annotations individually** option to provide annotation for each change (in this case, only one).

9. Click **Publish** to publish data to MDS.

10. Notice that **color coding** for the row with **North Carolina** as the **State** is same as other records now.

11. **Optional:** Verify that the new member (NC) is added to the **State** entity by using the **Explorer** in **Master Data Manager**.

12. In Excel, right-click the **State** worksheet at the bottom, and click **Delete** to delete the worksheet. Deleting the worksheet does not delete any data from the MDS server.

Next Step

Task 9: Creating a Derived Hierarchy using Master Data Manager

Task 9: Creating a Derived Hierarchy using Master Data Manager

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a derived hierarchy by using Master Data Manager. This derived hierarchy is derived from the domain-based attribute relationships between the **Supplier** and **State** entities.

1. Switch to the main page of **Master Data Manager** by clicking **SQL Server 2012 Master Data Services** at the top of the page.
2. Click **System Administration** in the **Administrative Tasks** section.
3. Hover the mouse over **Manage** on the menu bar, and click **Derived Hierarchies**.

4. Click **Add Derived Hierarchy (+)** button on the toolbar.

5. Type **SuppliersInState** for the **Derived hierarchy name**.
6. Click **Save** button on the toolbar.

7. Drag **Supplier** from **Available Levels: SuppliersInState** to **Current Levels: SuppliersInState**.

8. Drag **State** from **Available Levels: SuppliersInState** to **Current Levels: SuppliersInState**. The screen should have **Current Levels** as shown in the following picture.

9. In the **Preview** window, expand **NY { New York}** and you should see one supplier in that state as shown in the preceding image.

10. Switch to the main page of **Master Data Manager** by clicking **SQL Server 2012 Master Data Services** at the top of the page.

11. Click **Explorer**.

12. Hover the mouse over **Hierarchies** and click **Derived:SuppliersInState**.

13. Click on any **state** node in the **tree view** and you should see the suppliers in that state in the right pane.

Next Step

Lesson 5: Automating the Cleansing and Matching using SSIS

Lesson 5: Automating the Cleansing and Matching using SSIS

2/11/2019 • 2 minutes to read • Edit Online

In Lesson 1, you built the Suppliers knowledge base and used it to cleanse data in Lesson 2 and match data in Lesson 3 using the tool **DQS Client**. In a real world scenario, you may have to pull data from a source that DQS does not support or you want to automate the cleansing and matching process without having to use the **DQS Client** tool. SQL Server Integration Services (SSIS) has components that you can use to integrate data from various heterogeneous sources and a **DQS Cleansing Transform** component to invoke the cleansing functionality exposed by DQS. Currently, DQS does not expose matching functionality for SSIS to use, but you can use the **Fuzzy Grouping Transform** to identify duplicates in the data.

You can upload data to MDS by using the **Entity-based Staging feature**. When you create an entity in MDS, corresponding staging tables and stored procedures are automatically created. For example, when you created the Supplier entity, the **stg.supplier_Leaf** table and the **stg.udp_Supplier_Leaf** stored procedure were automatically created. You use the staging tables and procedures to create, update, and delete entity members. In this lesson, you create new entity members for the Supplier Entity. To load data into the MDS server, the SSIS package first loads the data into the staging table stg.supplier_Leaf and then triggers the associated stored procedure stg.udp_Supplier_Leaf. See Importing Data for more details.

In this lesson, you perform the following tasks:

1. Remove supplier data in MDS (if you have gone through previous four lessons). The SSIS package you create in this lesson uploads the data to MDS automatically. Earlier, you uploaded the cleansed and matched supplier data to MDS server manually using the DQS Client.
2. Create a subscription view on the Supplier entity to expose data in the entity to other applications. This action creates a SQL view that you will verify using SQL Server Management Studio. You will not be consuming this view in this version of the tutorial.
3. Create and run an SSIS project by using **SQL Server Data Tools**. The project uses **Data Cleansing** transform to submit a cleansing request to the DQS server. DQS does not expose the matching functionality yet, so you will use **Fuzzy Grouping** transform to identify duplicates.
4. Verify that the data is created in MDS by using Master Data Manger.
5. Review the results from DQS cleansing project created by the SSIS package and optionally perform interactive cleansing to build the knowledge base further.

Next Step

Task 1 (Prerequisite): Removing Supplier Data in MDS

Task 1 (Prerequisite): Removing Supplier Data in MDS

2/11/2019 • 2 minutes to read • Edit Online

In this task, you remove the supplier data stored in MDS. You uploaded the data manually using **MDS Excel Add-in** in the previous lesson. The SSIS package you create in this lesson will automatically upload the data to MDS for you. Therefore, before testing the SSIS package, you need to remove the supplier data from MDS, remove the derived hierarchy, remove supplier and state entities, and create the supplier entity with no data.

1. Launch **Master Data Manager** by navigating to **http://localhost/MDS** or the website and application you specified when configuring MDS. If you kept the **Master Data Manager** open, click **SQL Server 2012 Master Data Services** at the top to switch to the **home page**.
2. Click **System Administration** in the **Administrative Tasks** section.
3. Hover the mouse over **Manage** on the menu and click **Derived Hierarchies**. You need to delete the derived hierarchy **SuppliersInState** before deleting the entities in the **Suppliers** model.
4. Select **SuppliersInState** from the **Derived Hierarchy** list and click **X (Delete)** button on the toolbar.
5. Click **OK** to confirm deletion.
6. Hover the mouse over **Manage** on the menu and click **Entities**.
7. Click **Supplier** and click **Delete (X)** button on toolbar to delete the entity. Click **OK** on message boxes.
8. Repeat the previous step to delete **State** entity.
9. Don't close **Master Data Manager**.
10. Switch to the Excel window that has **Cleansed and Matched Suppliers.xls** file open. Switch to the **Sheet1** tab at the bottom.
11. Select only the **first row with headers**. Don't select any other row. You want to create the entities based on the Excel columns but don't want to upload any data. Therefore you select only the first row with the headers.
12. Click **Master Data** on the menu bar.
13. Click **Create Entity** from the ribbon.
14. In **Manage Connections** dialog box, if you do not see the connection to **local MDS server** under **Existing connections**, do the following:
    a. Select **Create a new connection**, and click **New** button.
    b. In the Add New Connection dialog box, type **Local MDS Server** for **Description** and **http://localhost/MDS** for **MDS server address**, and click **OK** to close the dialog box.
15. In **Manage Connections** dialog box, select **Local MDS Server** (http://localhost/MDS), click **Test** to test the connection. Click **OK** on the message box.
16. Click **Connect** to make a connection to the MDS server.
17. In the **Create Entity** dialog box, do the following:
    a. Confirm that **Range** is set to **$1:$1**.
    b. Select **Suppliers** for **Model**.

c. Select **VERSION_1** for **Version**.

d. Type **Supplier** for **New entity name**.

e. Select **SupplierID** for **Code**.

f. Select **Supplier Name** for **Name**.

g. Click **OK** to create the entity and close the dialog box.

18. Close **Excel** and **do not save** the file.

19. In **Master Data Manager**, refresh the internet browser and confirm that **Supplier** entity is displayed in the list.

20. Switch to the **home page** by clicking **SQL Server 2012 Master Data Services** at the top.

21. Confirm that **Suppliers** model is selected for **Model** and **VERSION_1** is selected for **Version**.

22. Click **Explorer**. Notice that the **Supplier** entity with all the attributes is created with **no values**.

Next Step

Task 2 (Optional): Creating a MDS Subscription View using Master Data Manager

Task 2 (Optional): Creating a MDS Subscription View using Master Data Manager

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create a subscription view to expose the **Supplier** entity in the **Suppliers** model to other applications. You do not consume this view in the current version of the tutorial.

1. Switch to the main page of **Master Data Manager** (http://localhost/MDS) by clicking **SQL Server 2012 Master Data Services** at the top.
2. Click **Integration Management**.
3. Click **Create Views** on the menu bar.

4. Click **+ (Plus)** icon on the toolbar to create a subscription view.
5. In the **Create Subscription View** pane, type **Suppliers** for **Subscription view name**.
6. Select **Suppliers** for **Model**.
7. Select **VERSION_1** for **Version**.
8. Select **Supplier** for **Entity**.
9. Select **Leaf members** for **Format**.

10. Click **Save** on the toolbar to save the subscription view. This action creates a view in SQL Server named **Suppliers**. You can verify this using SQL Server Management Studio (SSMS).

Next Step

Task 3 (Optional): Reviewing the Subscription Views

Task 3 (Optional): Reviewing the Subscription Views

2/11/2019 • 2 minutes to read • Edit Online

In this task, you confirm that the SQL views are created by using SQL Server Management Studio.

1. Launch **SQL Server Management Studio**. Click the **Start** button, click **All Programs**, click **Microsoft SQL Server 2012**, and then click **SQL Server Management Studio**.
2. In the **Connect to Server** window, set **Server Type** to **Database Engine**, type the **server name** (or select **(local)**, and select appropriate **authentication**, and click **Connect**.
3. In the **Object Explorer** pane, expand **Databases**, expand **MDS**, and then expand **Views**.
4. Confirm that you see the **mdm.Suppliers** view in the list.

Next Step

Task 4: Creating an SSIS Project using SQL Server Data Tools

Task 4: Creating an SSIS Project using SQL Server Data Tools

2/11/2019 • 2 minutes to read • Edit Online

In this task, you create an SSIS project by using **SQL Server Data Tools** to automate cleansing and matching supplier data.

1. Launch **SQL Server Data Tools**. Click Start, point to **All Programs**, expand **Microsoft SQL Server 2012**, and click **SQL Server Data Tools**.

2. On the **File** menu, point to **New**, and click **Project**.

3. Expand **Business Intelligence** in the **Installed Templates** pane, and select **Integration Services**.

4. Select **Integration Services Project** in the **list of project types**.

5. Type **CleanseAndCurateSuppliers** for **Name** and click **OK**.

6. In **Solution Explorer** window, right-click **Package.dtsx** and select **Rename**. If you don't see **Solution Explorer** window, click **View** on the menu bar and click **Solution Explorer**.

7. Type **CleanseAndCurate.dtsx** and press **ENTER**. Make sure that the **extension** remains **.dtsx**.

Next Step

Task 5: Adding Data Flow Task

Task 5: Adding Data Flow Task

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add a Data Flow Task to the control flow of SSIS package.

1. Drag and drop **Data Flow Task** from **SSIS Toolbox** to the **Control Flow** tab in the SSIS Designer. If you do not see the **SSIS Toolbox**, click anywhere in the **Control Flow** tab, click **SSIS** on the menu bar, and click **SSIS Toolbox**.

2. Right-click the **Data Flow Task** in the **Control Flow** tab and click **Rename**.

3. Type **Receive, Cleanse, Match, and Curate Supplier Data** and press **ENTER**.

4. Double-click the **Data Flow Task** to switch to the **Data Flow** tab.

Next Step

Task 6: Adding Excel Source to the Data Flow

Task 6: Adding Excel Source to the Data Flow

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add an Excel Source to the data flow to read supplier data from the source Excel file. The Excel Source extracts data from worksheets or ranges in Microsoft Excel workbooks. See Excel Source topic for more details.

1. Drag-drop **Excel Source** from **Other Sources** in **SSIS Toolbox** to the **Data Flow** tab.
2. Right-click on **Excel Source** in the **Data Flow** tab, and click **Rename**.
3. Type **Read Supplier Data from Excel File** and press **ENTER**.
4. Double-click **Read Supplier Data from Excel File** to launch the **Excel Source Editor** dialog box.
5. In the **Excel Source Editor** dialog box, click **New** to create an Excel connection.
6. In the **Excel Connection Manager** dialog box, click **Browse**, and then select the **Suppliers.xls** file in the **EIM Tutorial** folder. Confirm that **Microsoft Excel 97-2003** is selected in the **Excel Version** box and then click **OK**.

7. In the **Excel Source Editor** dialog box, select **IncomingSuppliers$** in the **Name of the Excel sheet** list box.

8. Click **Preview** to preview the data in Excel file.
9. Click **OK** to close the dialog box.

10. Drag-drop **DQS Cleansing** transform in **Other Transforms** on the **SSIS Toolbox** to the **Data Flow** tab under **Read Supplier Data from Excel File**. The DQS Cleansing transformation uses Data Quality Services (DQS) to correct data by applying approved rules in the knowledge base. This transform, at runtime, creates a DQS cleansing project on the DQS server. See DQS Cleansing Transformation topic for more details.

Next Step

Task 7: Adding DQS Cleansing Transform to the Data Flow

Task 7: Adding DQS Cleansing Transform to the Data Flow

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add DQS Cleansing Transform to the data flow to cleanse the input supplier data by using DQS. See **DQS Cleansing Transform** for more details about the transform.

1. Right-click **DQS Cleansing** in the **Data Flow** tab, and click **Rename**. Type **Cleanse Supplier Data**, and press **ENTER**.
2. Select **Read Supplier Data from Excel File**; drag the blue connector to **Cleanse Supplier Data**. The components are now connected.

3. Double-click **Cleanse Supplier Data**.
4. In the **DQS Cleansing Transformation Editor**, click **New** next to the **Data Quality Connection Manager drop-down list**.
5. In the **DQS Cleansing Connection Manager** dialog box, type **(local)** or **period** (.) to connect to the local server. This lesson assumes that you have DQS installed on a local server.
6. Click **Test Connection** to test the connection to DQS server.
7. Click **OK** to close the dialog box.
8. Select **Suppliers** for the **Data Quality Knowledge Base**.

9. Switch to the **Mapping** tab at the top.

10. From **Available Input Columns**, select **Supplier Name**, **ContactEmailAddress**, **Address Line**, **City**, **State**, **Country**, and **Zip Code** by selecting the check boxes.

11. In the bottom pane, map these columns by using drop-down lists in the **Domain** column:

COLUMN	DOMAIN
Supplier Name	Supplier Name
ContactEmailAddress	Contact Email
Address Line	Address Line
City	City
State	State
Country	Country
Zip Code	Zip

12. Click **OK** to close the **DQS Cleansing Transformation Editor** dialog box.

Next Step

Task 8: Adding Conditional Split Transform to Split Cleansing Output

Task 8: Adding Conditional Split Transform to Split Cleansing Output

2/11/2019 • 2 minutes to read • Edit Online

In this transform, you add a Conditional Split Transform to the data flow. The Conditional Split transformation can route rows to different outputs based on the content of the data. For this tutorial, you use the **Record Status** output column from the DQS Cleansing transform. You will upload only correct or corrected records to MDS server in this tutorial. Therefore you check if the **Record Status** is **Correct** or **Corrected**, and combine the records before uploading the records to MDS.

1. Drag-drop **Conditional Split Transform** from **Common** section in the **SSIS Toolbox** to the **Data Flow** tab under **Cleanse Supplier Data**.
2. Right-click **Conditional Split**, and click **Rename**. Type **Pick Correct and Corrected Records** and press **ENTER**.
3. Connect **Cleanse Supplier Data** and **Pick Correct and Corrected Records** using the blue connector.

4. Double-click **Pick Correct and Corrected Records** in the **Data Flow** tab.
5. Change the **Default Output Name** at the bottom of the screen to **Correct**.
6. Expand **Columns** in the **top-left pane**.

7. Drag-drop **Record Status** to the **Condition** column.
8. Type **=="Corrected"** next to **[Record Status]** for the **Condition** column.
9. Click **Case 1** in the **Output Name Column**, and change the name to **Corrected**.
10. Click **OK** to close the **Conditional Split Transformation Editor** dialog box.

Next Step

Task 9: Adding Union All Transform to Combine Correct and Corrected Records

Task 9: Adding Union All Transform to Combine Correct and Corrected Records

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add the Union All Transform to the data flow. The Union All transformation combines multiple inputs into one output. In your scenario, it combine both Correct and Corrected records into one stream.

1. Drag-drop **Union All** Transform from **Common** section of the **SSIS Toolbox** to the **Data Flow** tab and place it under **Pick Correct and Corrected Records**.

2. Right-click **Union All** Transform in the **Data Flow** tab, and click **Rename**. Type **Combine Correct and Corrected Records**, and press **ENTER**.

3. Connect **Pick Correct and Corrected Records** to **Combine Correct and Corrected Records** in the **Data Flow** tab by using the blue connector. You should see the **Input Output Selection** dialog box.

4. In the **Input Output** dialog box, select **Correct** for **Output** and click **OK**.

5. Move the connector titled **Correct** to the left by dragging and dropping the dot at the end of the connector to left.

6. If you select **Pick Correct and Corrected Records** transform, you should see another blue connector. Drag that blue connector to **Combine Correct and Corrected Records**.

7. This **connector** should be titled **Corrected**. Since you have only two conditions **Correct** and **Corrected**, and one condition was already used, the **Input Output Selection** dialog box is not displayed this time. If the connectors overlap, move one to left and the other one to right by dragging the connector to left or right.

Next Step

Task 10: Adding Fuzzy Group Transform to Identify Duplicates

Task 10: Adding Fuzzy Group Transform to Identify Duplicates

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add a Fuzzy Group Transform to the data flow. The Fuzzy Group transformation can help identify duplicates in the source data. See Fuzzy Grouping Transformation for more details.

1. Drag-drop **Fuzzy Group** transform in **Other Transforms** on the **SSIS Toolbox** to the **Data Flow** tab under **Combine Correct and Corrected Records**.
2. Right-click **Fuzzy Group** Transform in the **Data Flow** tab, and click **Rename**. Type **Group Suppliers with matching IDs** and press **ENTER**.
3. Connect **Combine Correct and Corrected Records** to **Group Suppliers with matching IDs** using the blue connector.

4. Double-click **Group Suppliers with matching IDs**.
5. In the **Fuzzy Group Transformation Editor**, click **New** next to **OLE DB Connection Manager drop-down list** to launch **Configure OLE DB Connection Manager** dialog box.
6. In the dialog box, click **New** to launch **Connection Manager** dialog box.
7. Type **(local)** or **period** (.) for the Server name.
8. Select **MDS** for **Select or enter a database name** field. You will use the MDS database as the temporary storage for the **Fuzzy Group Transform**. The **Fuzzy Grouping** transformation requires a connection to an instance of SQL Server to create the temporary SQL Server tables that the transformation algorithm requires to do its work. You can create a database or use another existing database for this purpose.
9. Click **Test Connection** to test the connection and click **OK** on the message box.
10. In the **Connection Manager** dialog box, click **OK**.
11. Select **(local).MDS** (or **localhost.MDS**) from the **list of Data Connections** and click **OK**.
12. In the **Fuzzy Grouping Transformation Editor**, confirm that **(local).MDS** or **localhost.MDS** is selected for the **OLE DB Connection Manager**.
13. Switch to the **Columns** tab.
14. Select (check box) **SupplierID_Output** from the list of **Available Input Columns**. To configure the transformation, select the input columns to use when identifying duplicates. To keep it simple, you only use the SupplierID in this step.

15. Click **OK** to close the **Fuzzy Group Transformation Editor**.

Next Step

Task 11: Adding Conditional Split Transform to Filter Duplicates

Task 11: Adding Conditional Split Transform to Filter Duplicates

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add the Conditional Split Transform to the data flow. This transform helps you filter duplicates from the incoming record set. The Fuzzy Group transform groups the records that it finds to be matches and picks one of the records as a pivot record. All the records in a group have the same _key_out value. The pivot record in the group has _key_in same as the _key_out value. The other records in the group have different values for _key_in and _key_out. Therefore, when you filter using the condition _key_in==_key_out, you only get the pivot row in the group.

1. Drag-drop **Conditional Split** Transform from **Common** section in the **SSIS Toolbox** to the **Data Flow** tab.
2. Right-click **Conditional Split Transform** in the **Data Flow** tab, and click **Rename**. Type **Filter Duplicates** and press **ENTER**.
3. Connect **Group Suppliers with Matching IDs** to **Filter Duplicates**.
4. Double-click **Filter Duplicates** to launch the **Conditional Split Transform Editor** dialog box.
5. Expand **Columns** in the top-left pane.
6. Drag-drop **_key_in** to the **Condition** column.
7. Type == (equals to) next to **_key_in** and drag-drop **_key_out**.
8. Click **Case 1** in the **Output Name** column, type **Unique Records**, and press **ENTER**.

9. Click **OK** to close the **Conditional Split Transformation Editor** dialog box.

Next Step

Task 12: Adding Derived Column Transform to Add Columns Required by MDS

Task 12: Adding Derived Column Transform to Add Columns Required by MDS

2/11/2019 • 2 minutes to read • Edit Online

In this task, you add the Derive Column Transform to the data flow. You add two derived columns, **ImportType** and **BatchTag**, to the records passed to this transform. You should add these columns before uploading the data to staging tables in MDS. These two are required columns for the staging tables in MDS. See Leaf Member Staging Tables for more details.

1. Drag-drop **Derived Column transform** from **Common** section in the **SSIS Toolbox** to the **Data Flow** tab.
2. Right-click **Derived Column** Transform in the **Data Flow** tab, and click **Rename**. Type **Add Columns Required by MDS** and press **ENTER**.
3. Connect **Filter Duplicates** to **Add Columns Required by MDS** using the blue connector. You should see the **Input Output Selection** dialog box.
4. In the **Input Output Selection** dialog box, select **Unique Records**, and click **OK**.

5. Click **SSIS** on the menu bar and click **Variables**.
6. In the **Variables** window, click **Add Variable** button on the toolbar.

7. Type **ImportType** for the **Name** and **2** for the **value**. You specify the value as 2 because you want to add new members to an entity in MDS. For details about this parameter, see Leaf Member Staging Table.
8. Click **Add Variable** toolbar button again.
9. Type **BatchTag** for the **Name**, select **String** for the **Data type**, and **EIMBatch** for the **Value**. **BatchTag** is just a unique name for the batch you will be submitting to MDS.
10. In the **Data Flow** tab, double-click **Add Columns Required by MDS**.

11. In the **Derived Column Transformation Editor** dialog box, in the **list box in the bottom pane**, type **ImportType** for the **Derived Column Name**.

12. Expand **Variables and Parameters** in the top-left pane, drag-drop **User::ImportType** to the **Expression** column.

13. Type **BatchTag** in the next row for the **Derived Column Name**.

14. Drag-drop **User::BatchTag** from **Variables and Parameters** to the **Expression** column.

15. Click **OK** to close the **Derived Column Transformation** dialog box.

Next Step

Task 13: Adding OLE DB Destination to Write Data to MDS Staging Table

Task 13: Adding OLE DB Destination to Write Data to MDS Staging Table

2/11/2019 • 2 minutes to read • Edit Online

Now that you have added **ImportType** and **BatchTag** values to all records, you are ready to send them over to MDS for staging. In this task, you use the OLE DB Destination to write the data into **stg.supplier_Leaf** staging table.

1. Drag **OLE DB Destination** from **Other Destinations** section in the **SSIS Toolbox** to the **Data Flow** tab and drop it under **Add Columns Required by MDS**.
2. Right-click **OLE DB Destination** in the **Data Flow** tab, and click **Rename**. Type **Write Supplier Data to MDS Staging Table** and press **ENTER**.
3. Connect the **Add Columns Required by MDS** to **Write Supplier Data to MDS Staging Table** using the blue connector.
4. Double-click **Write Supplier Data to MDS Staging Table** in the **Data Flow** tab.
5. In the **OLE DB Destination Editor** Dialog box, make sure that **(local).MDS** (or **localhost.MDS**) is selected for the **OLE DB Connection Manager** field.
6. Select **stg.Supplier_Leaf** table from the list of **Name of the table or the view**.

7. Switch to the **Mappings** page by clicking **Mapping** on the menu on left.
8. Map **input** and **destination** columns as shown in the following table.

Input Column	Destination Column
ImportType	ImportType
<ignore>	ImportStatus_ID
<ignore>	Batch_ID
BatchTag	BatchTag
<ignore>	ErrorCode
SupplierID	Code
Supplier Name_Output	Name
<ignore>	NewCode
Contact First Name	Contact First Name
Contact Last Name	Contact Last Name
ContactEmailAddress_Output	ContactEmailAddress
Address Line_Output	Address Line
City_Output	City
State_Output	State
Zip Code_Output	Zip Code
Country_Output	Country

9. Confirm that you are using **_Output** columns for Input Columns, not the **_Status** or **_Source** columns. **_Output** columns contain the output values from DQS Cleansing.

10. Click **OK** to close the **OLE DB Destination Editor** dialog box.

11. The data flow should like the following image.

Next Step

Task 14: Adding Execute SQL Task to Control Flow to Run the Stored Procedure for MDS

Task 14: Adding Execute SQL Task to Control Flow to Run the Stored Procedure for MDS

2/11/2019 • 2 minutes to read • Edit Online

After loading data into the staging tables of MDS, you run a stored procedure associated with that table to load the data from staging into the appropriate tables in the MDS database. This stored procedure has two required parameters that you need to pass: LogFlag and VersionName. LogFlag specifies whether transactions are logged during the staging process and VersionName represents the version of the model. See Staged Stored Procedure topic for more details.

In this task, you add the Execute SQL Task to the control flow to invoke the stored procedure to load the staged data into appropriate MDS tables.

1. Now, switch to the **Control Flow** tab.
2. Drag-drop **Execute SQL Task** from the **SSIS Toolbox** to the **Control Flow** tab.
3. Right-click **Execute SQL Task** in the **Control Flow** tab, and click **Rename**. Type **Trigger Stored Procedure to Load Data into MDS** and press **ENTER**.
4. Connect **Receive, Cleanse, Match, and Curate Supplier Data** to **Trigger Stored Procedure to Load Data** using the green connector.

5. Using the **Variables** window, add two new variables with the following settings. If you do not see the **Variables** window, click **SSIS** on the menu bar and click **Variables**.

NAME	DATA TYPE	VALUE
LogFlag	Int32	1
VersionName	String	VERSION_1

6. Double-click **Trigger Stored Procedure to Load Data into MDS**.
7. In the **Execute SQL Task Editor** dialog box, select **(local).MDS** (or **localhost.MDS**) for **Connection**.

8. Type **EXEC [stg].[udp_Supplier_Leaf] ?, ?, ?** for **SQL Statement**. You can verify the name by using SQL Server Management Studio.

9. Click **Parameter Mapping** from the menu on left.

10. In the **Parameter Mapping** page, click **Add** to add a mapping. Maximize the window and resize columns so that you can see values in drop-down lists properly.

11. Select **User::VersionName** from the drop-down list for the **Variable Name**.

12. Select **NVARCHAR** for **Data Type**.

13. Type **0** (zero) for **Parameter Name**.

14. Repeat the previous four steps to add two more variables.

VARIABLE NAME	DATA TYPE (IMPORTANT)	PARAMETER NAME
User::LogFlag	LONG	1
User::BatchTag	NVARCHAR	2

15. Click **OK** to close the **Execute SQL Editor** dialog box.

Next Step

Task 15: Building and Running the SSIS Project

Task 15: Building and Running the SSIS Project

2/11/2019 • 2 minutes to read • Edit Online

In this task, you build and run the SSIS project. If you have the 64-bit version of Excel 2010 installed on your computer, you should set the value of **Run64BitRuntime** to **False** for the Excel source to work.

1. In the **Solution Explorer** window, click **Project** on the menu, and click **CleanseAndCurateSuppliers Properties**.

2. In the **Properties** dialog box, expand **Configuration Properties** on left, and click **Debugging**.

3. Set **Run64BitRuntime** to **False**.

4. Click **OK** to close the **Properties** dialog box.

5. Click **Build** on menu bar and click **Build CleanseAndCurateSuppliers**. Make sure that there are no build errors.

6. Click **Debug** on the menu bar and click **Start Debugging**.

7. Review messages in the **Progress** window and verify that package executed and ended successfully.

8. Click **Debug** on menu bar and click **Stop Debugging** to stop the debugging session. If the package fails, you should enable data viewers and see how the data flows between components.

Next Step

Task 16: Verifying with Master Data Manager

Task 16: Verifying with Master Data Manager

2/11/2019 • 2 minutes to read • Edit Online

In this task, you check the status of the batch job submitted by the SSIS package and verify that the data was uploaded to MDS server by using Master Data Manager.

1. Launch **Master Data Manager** (http://localhost/MDS). If it is already open, click **Microsoft SQL Server Master Data Services** at the top to switch to the **home page**.

2. Click **Integration Management**.

3. Notice that there is a batch with named **EIMBatch** that you submitted in the list. Click **Import Data** on the menu bar if you do not see the following screen.

4. Switch back to the home page by click **SQL Server 2012 Master Data Services** at the top.

5. Make sure that **Suppliers** model is selected for **Model** and **VERSION_1** is selected for **Version**, and click **Explorer**.

6. You can see the data SSIS package imported into MDS. The data should be cleansed and have no duplicates **Code** values (Note: **SupplierID** column in Excel corresponds to **Code** attribute of Supplier entity in MDS).

Next Step

Task 17: Reviewing DQS Cleansing Project Created by the SSIS package

Task 17: Reviewing DQS Cleansing Project Created by the SSIS package

2/11/2019 • 2 minutes to read • Edit Online

In this task, you open the DQS project created by the SSIS package in DQS Client, review the results from the cleansing process, and optionally perform interactive cleansing and export the results.

1. Launch **Data Quality Client**.
2. Click **Activity Monitoring** in the **Administration** pane.
3. Sort the list based on **Activity Start Time** to see the latest record.
4. Notice that you see a name of the project in the following format: **CleanseAndCurate.Cleanse Supplier Data.GUID**.

5. Notice that the value in the **Is Active** field is **Active**.
6. Click **Profiler** tab in the bottom pane to see profiler statistics for the Cleansing activity that the SSIS package performed.
7. Click **Close** to close the **Administration** screen.
8. In the main page of **DQS Client**, click **Open Data Quality Project** in the **Data Quality Projects** pane.
9. In the list of projects, select the project created by SSIS DQS Cleansing component. The name of the project should be in format: **CleanseAndCurate.Cleanse Supplier Data.GUID (in red color)**. You may need to sort the list based on **Date Created** column and look for the latest record.
10. Click **Next**.
11. The **Manage and View Results** page should be familiar to you from the interactive cleansing you did earlier in this tutorial.
12. Review the cleansing results. You can also perform interactive cleansing and export results to an Excel file or to a database in the next page.
13. Click **Next**. In this **Export** page, you can export results to an excel file, CSV file, or to a SQL database.
14. Click **Finish** to finish the activity.
15. In the main page of **DQS Client**, click **Activity Monitoring** in the **Administration** pane.
16. Notice that the value of **IsActive** field for the project is **Ended** now.

Next Step

Conclusion

Conclusion

2/11/2019 • 2 minutes to read • Edit Online

In this tutorial, you have learned how to use SQL Server Integration Services (SSIS), Master Data Services (MDS), and Data Quality Services (DQS) together to implement a sample Enterprise Information Management (EIM) solution. First, you used the Data Quality Client tool to create a DQS knowledge base with the knowledge about suppliers, cleansed the input supplier data in an excel file against the knowledge base, and then matched the supplier data by using a matching policy in the knowledge base to identify and remove duplicates in the data. Next, by using the MDS Add-in for Excel, you stored the cleansed and matched supplier list in MDS. Finally, you automated the whole process of receiving input data, cleansing and matching the data, and storing the master data in MDS by creating an SSIS solution.

For more information:

Enterprise Information Management (EIM): Bringing together SSIS, DQS, and MDS (Video)