

## 8 THE TABULAR SOLUTION



### MODULE OBJECTIVE

Tabular models are in-memory databases in Analysis Services. In this module we will explore the tabular model, create a tabular project, deploy and query the solution using Data Analytic Expressions (DAX).

### MODULE TOPICS

## MODULE TOPICS

- Understanding the Tabular Model
- Creating a Tabular Project
- Deploying
- Browsing the Model
- Querying the Solution and Understanding DAX
- Maintaining and Optimizing

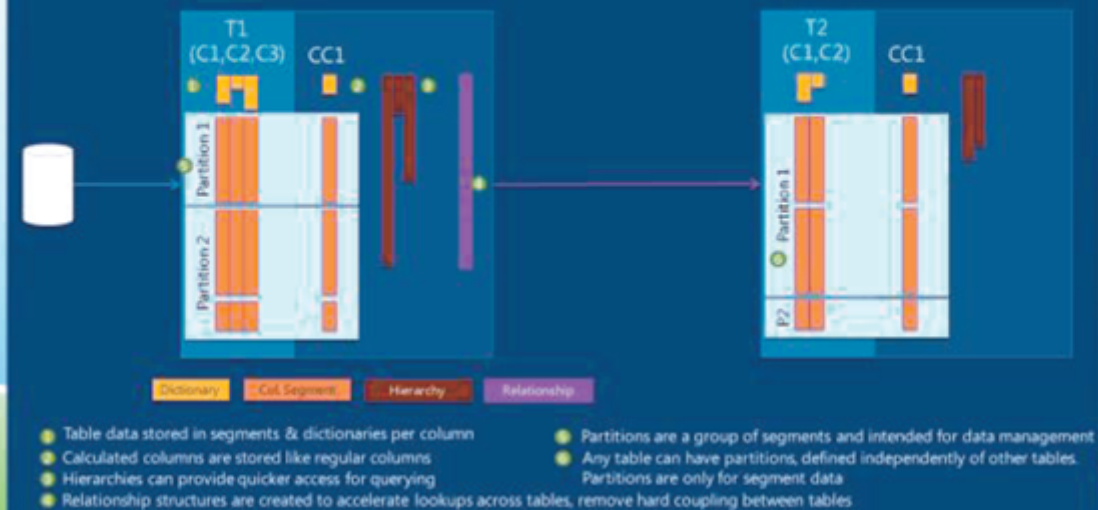


## UNDERSTANDING THE TABULAR MODEL

This docu

### UNDERSTANDING THE TABULAR MODEL

#### VertiPaq Storage



Tabular models are designed by using SQL Server Data Tools (SSDT), and a project in SSDT maps onto a database in Analysis Services. After you have finished designing a project in SSDT, it must be deployed to an instance of Analysis Services, which means SSDT executes a number of commands to create a new database in Analysis Services or alters the structure of an existing database. SQL Server Management Studio can also be used to write queries against databases. Databases

are made up of one or more tables of data. A table in the tabular model is very similar to a table in the relational database world. A table in tabular is usually loaded from a single table in a relational database or from the results of a SQL SELECT statement. A table has a fixed number of columns that are defined at design time and can have a variable number of rows, depending on the amount of data that is loaded. Each column has a fixed type, so for example, a single column could contain only integers, only text, or only decimal values. Loading data into a table is referred to as processing that table. It is also possible to define relationships between tables at design time.

Unlike in SQL, it is not possible to define relationships at query time; all queries must use these preexisting relationships. However, relationships between tables can be marked as active or inactive, and at query time it is possible to choose which relationships between tables are actually used. It is also possible to simulate the effect of relationships that do not exist inside queries and calculations. All relationships are one-to-many relationships and must involve just one column from each of two tables. It is not possible to define relationships that are explicitly one-to-one or many-to-many, although it is certainly possible to achieve the same effect by writing queries and calculations in a particular way. It is also not possible to design relationships that are based on more than one column from a table or recursive relationships that join a table to itself. The tabular model uses a purely memory-based engine and stores only a copy of its data on disk so that no data is lost if the service is restarted. Whereas the multidimensional model, like most relational database engines, stores its data in a row-based format, the tabular model uses a column-oriented database called the xVelocity in-memory analytics engine, which in most cases offers significant query performance improvements.

## XVELOCITY ENGINE AND VERTIPAQ STORAGE

The xVelocity engine is designed with three principles in mind:

- ▶ Performance, Performance, Performance!
- ▶ Query performance is much more important than processing performance.
- ▶ Accommodate changes without forcing reload, if possible.

When loading data, it goes through various stages:

- ▶ **Encoding**—everything is converted to integers, even strings (as it is the fastest data type).
  - **Value**—one just has to do math to get to the answer. As an example, 1.5, 2.5, and 3.5 will be multiplied by 10 to get a whole number. You could then subtract 15 to get to a smaller number that has to be stored (DATA ID).
  - **Hash**—used for strings. Useful for very sparse value distribution. DATA ID is the index allocated to the string when it is inserted in the table for the first time. It requires decompression for computation. One can't do math on hash encoding.

Encoding is done automatically to conserve space. On the first load, a sample set of data will be checked to determine the best type of encoding applicable for that column.

- ▶ **Compression (per segment)**—calculated columns are only addressed after the compression phase.

Tabular models use the xVelocity analytics engine, which provides in-memory data processing, or DirectQuery, which passes queries to the source database to leverage its query processing capabilities. The benefits of columnar databases and in-memory data processing go hand in hand. Columnar databases achieve higher compression than traditional storage, typically 10x compression depending on the data cardinality. Data cardinality focuses on characterizing the data distribution within a single column. High data cardinality means that the data values within a column are highly unique (for example, customer number). Low data cardinality means that the data values within a column can repeat (for example, gender and marital status). The lower the data cardinality, the higher the compression, which means that more data can fit into memory at a given time. As data modelers, it is important to understand the cardinality of your data to determine which datasets are best suited for your tabular model as well as the associated memory requirements to support the model.

## QUERYING PERFORMANCE

When a user queries a tabular model, the engine performs memory scans to retrieve the data and to calculate aggregations on the fly without the need of disk I/O processing. This approach can provide very high query performance without requiring special tuning and aggregation management.

The best and easiest way to optimize query performance for tabular models is to maximize available memory. From a scalability perspective, data volume is mostly limited by physical memory. It is highly recommended that you provide sufficient memory to contain all of the data in your tabular model. In scenarios where memory is constrained, the in-memory engine also provides basic paging support according to physical memory. In addition, there are server-side configuration settings that allow IT to more finely manage the memory available to tabular models.

## CREATING A TABULAR PROJECT



### ► Created with SQL Server Data Tools

- **Analysis Services Tabular Project** creates a new, empty project for designing a tabular model.
- **Import from PowerPivot** enables you to import a model created by using PowerPivot into a new SSDT project.

- **Import from Server (Tabular)** enables you to point to a model that has already been deployed to Analysis Services and import its metadata into a new project.

## DEPLOYING



### DIRECTLY FROM SQL SERVER DATA TOOLS

Deploying directly from SQL Server Data Tools (SSDT), deploys the model according to the Deployment Server configuration of your database. Visual Studio 2010 configurations, unfortunately, are not supported for tabular projects. You can just change deployment server properties (such as server and database name) every time you want to switch from a development to a production environment. A workaround to that is using a custom MSBuild task, as described at <http://blogs.msdn.com/b/cathyk/archive/2011/08/10/deploying-tabularprojects-using-a-custom-msbuild-task.aspx>.<sup>vi</sup> This approach can work if you have access to all the servers from your development workstation, and you have administrative rights on every Analysis Services instance to execute the deployment operation by using MSBuild.

### DEPLOYMENT WIZARD



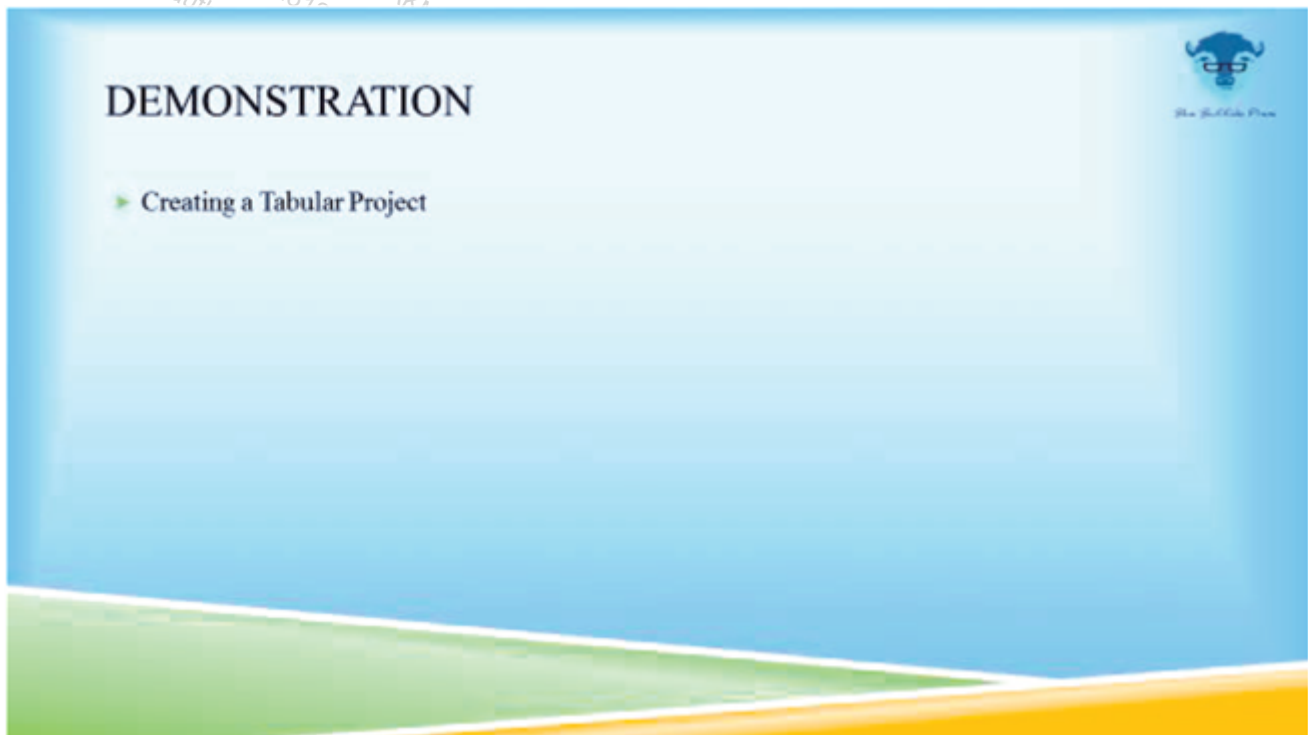
By using the Analysis Services Deployment Wizard, you can generate a deployment XMLA script that you can forward to an administrator who can execute the script directly in SSMS, by ASCMD, or by scheduling it in SQL Server Agent, provided he or she has the required administrative rights.

## SYNCHRONIZE DATABASE WIZARD

Lastly, you can deploy using the Synchronize Database Wizard or by executing the XMLA script that this wizard can generate. This wizard copies the content of a database from the source server that you select to a target server. The server on which the wizard has been selected will be the target server and will receive the Synchronize command in an XMLA script. This option can be useful to move a database deployed on a development or test server to a production server, and is also useful when you want to duplicate the tabular database in a server farm that is part of a scale-out architecture.

## DEMONSTRATION

### VIDEO: CREATING A TABULAR PROJECT





## EXERCISE A.1: CREATING A TABULAR PROJECT

Objective: in this exercise we will create and configure a tabular project.

- A.1.1 Navigate down to the taskbar, right-click **SQL Server Data Tools** icon, right-click the new **SQL Server Data Tools** icon showing, and then click **Run as administrator**.
- A.1.2 In the **User Account Control** dialog box, click **Yes**.
- A.1.3 When **Microsoft Visual Studio** opens, click **New Project....**
- A.1.4 When the **New Project** dialog box opens, navigate to the left side of the dialog box, in the **Installed Templates** section, verify **Business Intelligence** is selected, then move to the middle of the dialog box, and click to select **Analysis Services Tabular Project**.
- A.1.5 Click **Browse....**
- A.1.6 In the **Project Location** dialog box, navigate to **C: | Lab Files | Student**.
- A.1.7 Inside the **Student** folder, create a new folder, and name it **08 The Tabular Solution**.
- A.1.8 Open the **08 The Tabular Solution** folder.



- A.1.9 Click **Select Folder**.
- A.1.10 Back in the **New Project** dialog box, navigate to the **Name** text box and change the name to FirstTabular.
- A.1.11 Click **OK**.
- A.1.12 When **Tabular model designer** dialog box opens, review the current settings.
- A.1.13 Navigate to the **Workspace server** setting, use the corresponding drop-down arrow, and click to select **localhost\TABULAR**.
- A.1.14 Click **Test Connection**.
- A.1.15 In the **Tabular model designer** dialog box advising **Test connection succeeded**, click **OK**.
- A.1.16 Move to the **Compatibility level** setting, use the corresponding drop-down arrow, and click to select **SQL Server 2012 SP 1 (1103)**.
- A.1.17 Click **OK**. Review the results.
- A.1.18 Navigate to **Solution Explorer** pane on the right, right-click **Model.bim** and click **Rename**.
- A.1.19 Enter InternetSales.bim.
- A.1.20 Click into a blank spot in the design area, and review the results.
- A.1.21 Navigate up to the menu and click **Model | Import From Data Source....**
- A.1.22 When the **Connect to a Data Source** dialog box opens, scroll through and review the options available.
- A.1.23 Leave **Microsoft SQL Server** selected, and click **Next**.
- A.1.24 When the **Connect to a Microsoft SQL Server Database** dialog box opens, review the settings.
- A.1.25 Navigate to the **Server name** setting, use the corresponding drop-down arrow, and click to select **QUICK**.
- A.1.26 Move down to the **Database name** setting, use the corresponding drop-

down arrow, and click to select **AdventureWorksDW2012**.

A.1.27 Click **Test Connection**.

A.1.28 In the **Tabular Model Designer** dialog box advising **Test connection succeeded**, click **OK**.

A.1.29 Back in the **Connect to a Microsoft SQL Server Database** dialog box, click **Next**.

A.1.30 In the **Impersonation Information** dialog box, review the settings.

A.1.31 Navigate to the **User Name** text box and enter Student.

A.1.32 Move to the **Password** text box and enter Passw0rd.



*In Passw0rd the 0 is numeric.*

A.1.33 Click **Next**.

A.1.34 When the **Choose How to Import the Data** dialog box opens, review the options available

A.1.35 Click the radio button to **Write a query that will specify the data to import**.

A.1.36 Click **Next**.

A.1.37 When the **Specify a SQL Query** dialog box opens, review the options available.

A.1.38 Click **Design....**

A.1.39 When the query designer opens, review the options available.

A.1.40 Close the query designer.

A.1.41 Back in the **Specify a SQL Query** dialog box, click **Back**.

A.1.42 Back in the **Choose How to Import the Data** dialog box, click the radio button to **Select from a list of tables and views to choose the data to import**.

- A.1.43 Click **Next**.
- A.1.44 In the **Select Tables and Views** dialog box, review the options available.
- A.1.45 Move to the **Tables and Views** section and place a check in the **FactInternetSales** check box.
- A.1.46 Click **Select Related Tables**.
- A.1.47 Review the results noticing **6 related tables were selected**.
- A.1.48 Clear the check from the **DimCurrency** check box.
- A.1.49 Move to the **DimCustomer** row, click into the corresponding **Friendly Name** cell and change the name to Customer. Click off of the cell and review the results.
- A.1.50 Move to the **DimDate** row, click into the corresponding **Friendly Name** cell and change the name to Date. Click off of the cell and review the results.
- A.1.51 Move to the **DimProduct** row, click into the corresponding **Friendly Name** cell and change the name to Product. Click off of the cell and review the results.
- A.1.52 Clear the check from the **DimPromotion** check box.
- A.1.53 Clear the check from the **DimSalesTerritory** check box.
- A.1.54 Click **Preview & Filter**.
- A.1.55 When the **Preview Selected Table** dialog box opens, review the results.
- A.1.56 Navigate to the **SalesTerritoryAlternateKey** column and clear the corresponding check from the check box.
- A.1.57 Move to the **SalesTerritoryImage** column and clear the corresponding check from the check box.
- A.1.58 Click **OK**.
- A.1.59 Notice you now see **Applied filters** in the **Filter Details** column of the **DimSalesTerritory** row.

- A.1.60 Click to select **FactInternetSales** row.
- A.1.61 Click **Preview & Filter**.
- A.1.62 When the **Preview Selected Table** dialog box opens, review the results.
- A.1.63 Navigate to the **DueDateKey** column and clear the corresponding check from the check box.
- A.1.64 Move to the **ShipDateKey** column and clear the corresponding check from the check box.
- A.1.65 Navigate to the **PromotionKey** column and clear the corresponding check from the check box.
- A.1.66 Move to the **CurrencyKey** column and clear the corresponding check from the check box.
- A.1.67 Navigate to the **SalesTerritoryKey** column and clear the corresponding check from the check box.
- A.1.68 Move to the **SalesOrderNumber** column and clear the corresponding check from the check box.
- A.1.69 Navigate to the **SalesOrderLineNumber** column and clear the corresponding check from the check box.
- A.1.70 Move to the **RevisionNumber** column and clear the corresponding check from the check box.
- A.1.71 Navigate to the **CarrierTrackingNumber** column and clear the corresponding check from the check box.
- A.1.72 Move to the **CustomerPONumber** column and clear the corresponding check from the check box.
- A.1.73 Navigate to the **DueDate** column and clear the corresponding check from the check box.
- A.1.74 Move to the **ShipDate** column and clear the corresponding check from the check box.
- A.1.75 Navigate to the **TotalProductCost** column, use the provided drop-down

arrow, and review the options available.

A.1.76 Click **Cancel**.

A.1.77 Click **OK**.

A.1.78 Click **Finish**.

A.1.79 In the **Importing** dialog box, review the status.

A.1.80 Upon **Success**, review the results.

A.1.81 Click **Details** link.

A.1.82 When the **Details** dialog box opens, review the results.

A.1.83 Click **OK**.

A.1.84 Click **Close**. Review the results.

A.1.85 Notice the tabs along the bottom of the interface.




A.1.86 Click **Date** tab, and review the results.

A.1.87 Click **Product** tab, and review the results.

A.1.88 Click **DimSalesTerritory** tab, and review the results.

A.1.89 Click **FactInternetSales** tab, and review the results.

A.1.90 Navigate to the lower-right and click  (**Diagram**). Review the results.

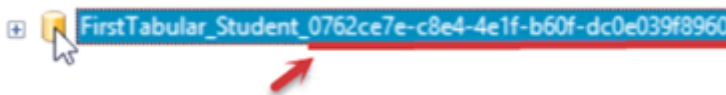
A.1.91 Notice **DimSalesTerritory** table is not linked via relationship to any of the other tables.

A.1.92 Right-click **DimSalesTerritory** table, and then click **Delete**.

A.1.93 In the **Confirm** dialog box asking **Do you want to permanently delete this table**, click **Delete from Model**. Review the results.


A.1.94 Minimize **Microsoft Visual Studio**.

- A.1.95 Navigate to the taskbar and start **SQL Server Management Studio**.
- A.1.96 When **SQL Server Management Studio** opens, navigate to the **Connect to Server** dialog box, move to the **Server type** setting, use the corresponding drop-down arrow, and click **Analysis Services**.
- A.1.97 Locate the **Server name** setting, use the corresponding drop-down arrow, and click to select **Quick\TABULAR**.
- A.1.98 Click **Connect**.
- A.1.99 Navigate to the **Object Explorer** pane on the left, and review the results noticing the icon identifying the connection.
- A.1.100 Expand **Databases**.
- A.1.101 Review the results noticing **FirstTabular\_Student\_...** database is listed, and contains a guid.

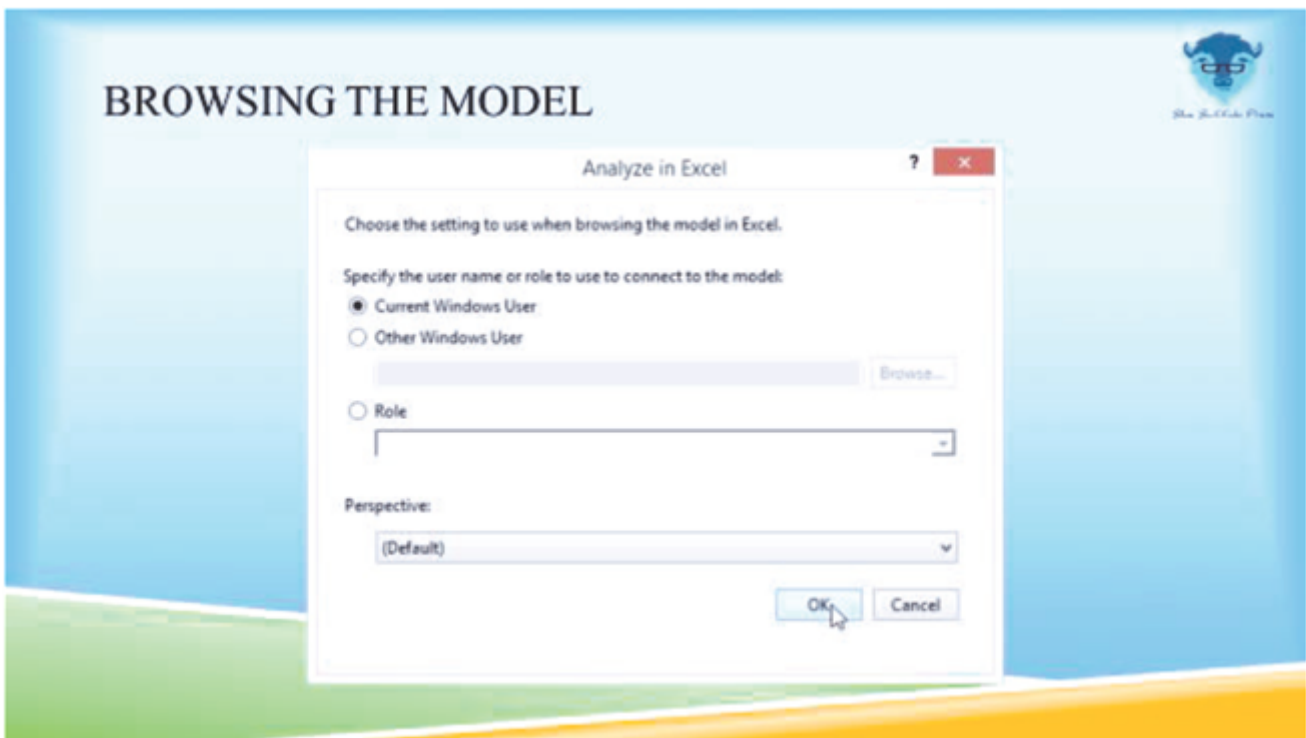


- A.1.102 Switch back to **Microsoft Visual Studio**.
- A.1.103 Navigate to the **Solution Explorer** pane on the right, right-click the **FirstTabular** project, then click **Build**.
- A.1.104 Notice in the lower-left you see **Build succeeded**.
- A.1.105 Move back to the **Solution Explorer** pane on the right, right-click the **FirstTabular** project, then click **Deploy**.
- A.1.106 When the **Deploy** dialog box opens, review the status.
- A.1.107 Upon **Success**, review the results, and then click **Close**.
- A.1.108 Notice in the lower-left you see **Deploy succeeded**.
- A.1.109 Switch to **SQL Server Management Studio**.
- A.1.110 Navigate to **Object Explorer** pane on the left, right-click **Databases** folder, and click **Refresh**.



- A.1.111 Review the results noticing you now see a new database listed (**FirstTabular**). Also notice the new database does not contain a guid.
- A.1.112 Switch back to **Microsoft Visual Studio**.
- A.1.113 Navigate up to the toolbar and click  (**Save All**).
- A.1.114 Close **Microsoft Visual Studio**.
- A.1.115 Switch to **SQL Server Management Studio**.
- A.1.116 Back in **SQL Server Management Studio**, navigate to the **Object Explorer** pane on the left, right-click the **Databases** folder within the **Analysis Server** connection, and click **Refresh**.
- A.1.117 Review the results noticing **FirstTabular** is now the only one listed.
- A.1.118 Expand **FirstTabular** database. Review the results.
- A.1.119 Expand **Tables** folder. Review the results.
- A.1.120 Right-click **Customer** table, and review the options available noticing there is no Browse listed.
- A.1.121 Close all open windows.

## BROWSING THE MODEL

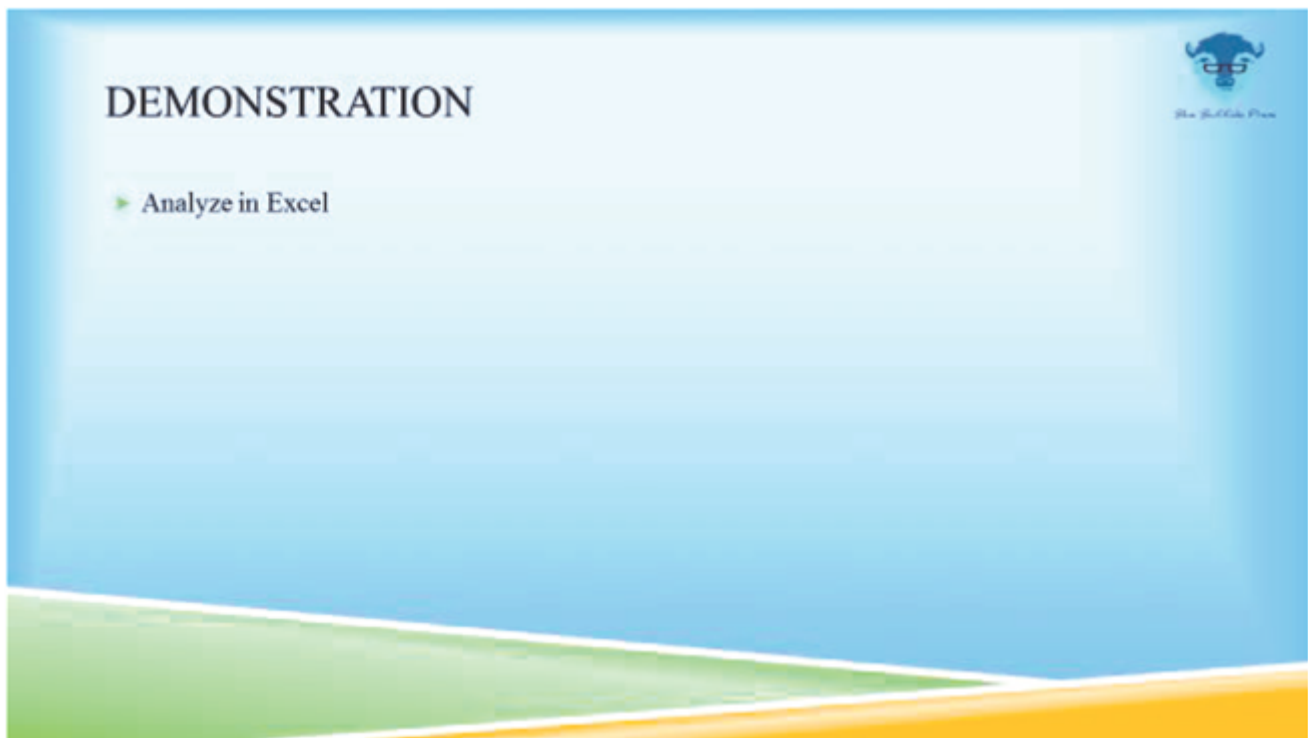


## ANALYZE IN EXCEL

The SSMS Browse window and the SSDT tabular project window both include the Analyze in Excel button on their respective menus. The button launches Excel and allows you to browse the tabular data from within in a pivot table. When you first click the Analyze in Excel button, you are able to choose which perspective to open in Excel. Once you've designated a perspective, click OK. Excel starts and opens to a new pivot table that automatically connects to the SSAS tabular data source. On the right-side of the interface you'll find the PivotTable Field List pane, which contains a list of all your database objects, including tables, columns, and measures. To create a pivot table, drag the fields and measures you want to include in your report to the COLUMNS, ROWS, and VALUES areas. The data will then be displayed in the pivot table. You can also create filters by dragging fields to the FILTERS area.

## DEMONSTRATION

### VIDEO: ANALYZE IN EXCEL




## EXERCISE B.1: ANALYZE IN EXCEL


Objective: in this exercise we will analyze data using Microsoft Excel.

B.1.1 Navigate down to the taskbar, right-click **SQL Server Data Tools** icon, right-click the new **SQL Server Data Tools** icon showing, and then click **Run as administrator**.

B.1.2 In the **User Account Control** dialog box, click **Yes**.

- B.1.3 When **Microsoft Visual Studio** opens, navigate to the **Recent Project** section, and click to open **FirstTabular**.
- B.1.4 When **FirstTabular** opens, review the results.
- B.1.5 Move up to the toolbar, and click  (**Analyze in Excel**).
- B.1.6 In the **Analyze in Excel** dialog box, review the settings and options available.
- B.1.7 Click **OK**.
- B.1.8 Notice **Microsoft Excel** opened.
- B.1.9 Switch to **Microsoft Excel**.
- B.1.10 When **Microsoft Excel** opens, review the results.
- B.1.11 Navigate to the **PivotTable Fields** pane on the right, and scroll through reviewing the options available.
- B.1.12 Move to the **Show fields** setting, use the corresponding drop-down arrow, and click to select **Product**. Review the results.
- B.1.13 Staying within the **PivotTable Fields** pane, place a check in the **Color** check box.
- B.1.14 Review the results noticing you now see **Color** in the **ROWS** section below.
- B.1.15 Hover your cursor onto the **Color** check box, use the provided drop-down arrow, and clear the check from the **(Select All)** check box.
- B.1.16 Place a check in the **Black**, **Blue**, and **Grey** check boxes. (*Only those 3 should be selected*)
- B.1.17 Click **OK**. Review the results.
- B.1.18 Switch back to **Microsoft Visual Studio**.
- B.1.19 Locate the **SalesAmount** column, and click into the first empty cell at the bottom of the column.

TotalProductCost	SalesAmount	TaxAmt
\$1.87	\$4.99	\$0.40

- B.1.20 Navigate up to the toolbar and click  (**Sum**). Review the results noticing the resulting value is labeled as **Sum of SalesAmount**.
- B.1.21 Switch back to **Microsoft Excel**.
- B.1.22 Back in **Excel**, click **DATA** tab.
- B.1.23 Locate the **Connections** section, use the drop-down arrow beneath **Refresh All**, and click **Refresh All**.
- B.1.24 Navigate to the **PivotTable Fields** pane on the right, move to the **Show fields** setting, use the corresponding drop-down arrow, and click to select **FactInternetSales**. Review the results.
- B.1.25 Staying within the **PivotTable Fields** pane, place a check in the **Sum of SalesAmount** check box. Review the results.
- B.1.26 Navigate up to the ribbon and click the **POWERPIVOT** tab.
- B.1.27 Locate the **Data Model** grouping and click **Manage**.
- B.1.28 When **PowerPivot for Excel-Book 1** opens, review the results noticing it is empty.
- B.1.29 Close **PowerPivot for Excel-Book 1**.
- B.1.30 Close **Microsoft Excel**.
- B.1.31 In the **Microsoft Excel** dialog box asking if you **Want to save your changes**, click **Don't Save**.

# QUERYING THE SOLUTION AND UNDERSTANDING DAX



DAX is the expression language that is utilized in BISM tabular as well as in Excel PowerPivot. It is an extension of the Excel formula language and as such will be familiar to Excel users. In BISM tabular the end user will use Excel to view the data unlike in multidimensional mode where they can browse the cube in SSMS.

The syntax of DAX formulas is very similar to that of Excel formulas, using a combination of functions, operators, and values. Where DAX formulas differ from Excel formulas is that DAX functions work with tables and columns, not ranges, and let you do sophisticated lookups to related values and related tables. With DAX formulas, you can create aggregations that would ordinarily require in-depth knowledge of relational database schemas or OLAP concepts. Moreover, because calculations in DAX formulas utilize the highly optimized in-memory engine, you can rapidly look up and calculate values across very large columns or tables. DAX provides functions that have the same functionality and names as the Excel functions that you might already be familiar with. However, the functions have been modified to use DAX data types and to work with tables and columns. In addition, DAX provides many specialized functions for specific purposes, such as lookups based on relationships, the ability to iterate over a table to perform recursive calculations, and calculations utilizing time intelligence.



## DAX formulas major differences from Excel

- ▶ A DAX function always references a complete column or a table. If you want to use only particular values from a table or column, you can add filters to the formula.
- ▶ If you want to customize calculations on a row-by-row basis, PowerPivot provides functions that let you use the current row value or a related value to perform calculations that vary by context.
- ▶ DAX includes a type of function that returns a table as its result, rather than a single value. These functions can be used to provide input to other functions, thus calculating values for entire tables or columns.
- ▶ Some DAX functions provide time intelligence, which lets you create calculations using meaningful ranges of dates, and compare the results across parallel periods.

### Syntax

Look at DAX formula syntax. Syntax includes the various elements that make up a formula, or more simply, how the formula is written. For example, let's look at a simple DAX formula used to create new data (values) for each row in a calculated column, named Margin, in a FactSales table (formula text colors are for illustrative purposes only).

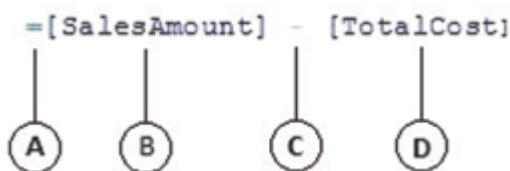


Figure 8.1

This formula's syntax includes the following elements:

- A. The equals sign operator (`=`) indicates the beginning of the formula, and when this formula is calculated it will return a result or value. All formulas that calculate a value will begin with an equals sign.

- B. The referenced column [SalesAmount] contains the values we want to subtract from. A column reference in a formula is always surrounded by brackets ([]). Unlike Excel formulas which reference a cell, a DAX formula always references a column.
- C. The subtraction (-) mathematical operator.
- D. The referenced column [TotalCost] contains the values we want to subtract from values in the [SalesAmount] column.

When trying to understand how to read a DAX formula, it is often helpful to break down each of the elements into a language you think and speak every day. For example, you can read this formula as:

In the FactSales table, for each row in the Margin calculated column, calculate (=) a value by subtracting (-) values in the [TotalCost] column from values in the [SalesAmount] column.

The QR codes below will take you to the corresponding references for your convenience.

- PowerPivot: DAX: Operator Reference<sup>vii</sup>
- DAX Function Reference<sup>viii</sup>
- DAX Cheat Sheet<sup>ix</sup>



PowerPivot:  
DAX: Operator  
Reference



DAX Function  
Reference



DAX Cheat  
Sheet

## DEMONSTRATION

### VIDEO: CALCULATED MEASURES AND CALCULATED FIELDS




## EXERCISE C.1: CALCULATED MEASURES AND CALCULATED FIELDS

Objective: in this exercise we will create a calculated measure and a calculated column.

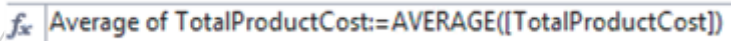
C.1.1 Switch back to **Microsoft Visual Studio**.

C.1.2 Locate the **TotalProductCost** column, and click into the first empty cell at the bottom of the column.

C.1.3 Navigate up to the toolbar, click the drop-down arrow next to  (**Sum**), and click **Average**.

C.1.4 Review the results noticing the resulting value is labeled as **Average of TotalProductCost**.

C.1.5 Review the resulting code in the function bar.

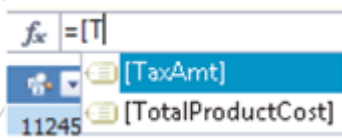


fx Average of TotalProductCost:=AVERAGE([TotalProductCost])

C.1.6 Scroll to the far right and click into the first empty cell of the **Add Column** column.

C.1.7 Move to the function bar and enter =[T.

C.1.8 IntelliSense displays all options available beginning with **T**. Double-click **[TaxAmt]**. Review the results.



fx =[TaxAmt]

C.1.9 Move back to the function bar, place your cursor at the end of the existing formula, and enter +[F.

C.1.10 IntelliSense displays all options available. Double-click **[Freight]**. Review the results.




fx =[TaxAmt]+[Freight]

C.1.11 Press **Enter**. Review the results.

C.1.12 Right-click the header cell in **CalculatedColumn1** column and click **Rename Column**.

C.1.13 Enter Tax And Freight, then press **Enter**. Review the results.

C.1.14 Move up to the toolbar, and click  (**Analyze in Excel**).

C.1.15 In the **Analyze in Excel** dialog box, review the settings and options available.

C.1.16 Click **OK**.

C.1.17 Notice **Microsoft Excel** opened.

C.1.18 Switch to **Microsoft Excel**.

C.1.19 When **Microsoft Excel** opens, review the results.

C.1.20 Navigate to the **PivotTable Fields** pane on the right, and scroll through reviewing the options available.

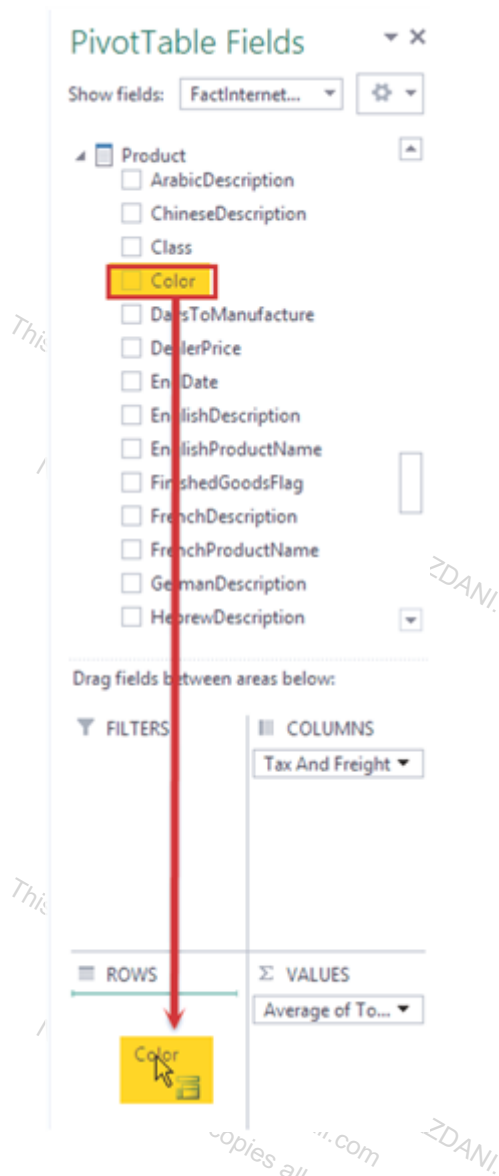
C.1.21 Move to the **Show fields** setting, use the corresponding drop-down arrow, and click to select **FactInternetSales**. Review the results.

C.1.22 Staying within the **PivotTable Fields** pane, place a check in the **Average of TotalProductCost** check box. Review the results.

C.1.23 Place a check in the **Tax And Freight** check box. Review the results.

C.1.24 Navigate down to the **ROWS** box, and using drag-and-drop, take **Tax And Freight** and drop it into the **COLUMNS** box. Review the results.

C.1.25 Locate the **Color** field, and using drag-and-drop, drop it into the **ROWS** box. Review the results.



C.1.26 Close **Microsoft Excel**.

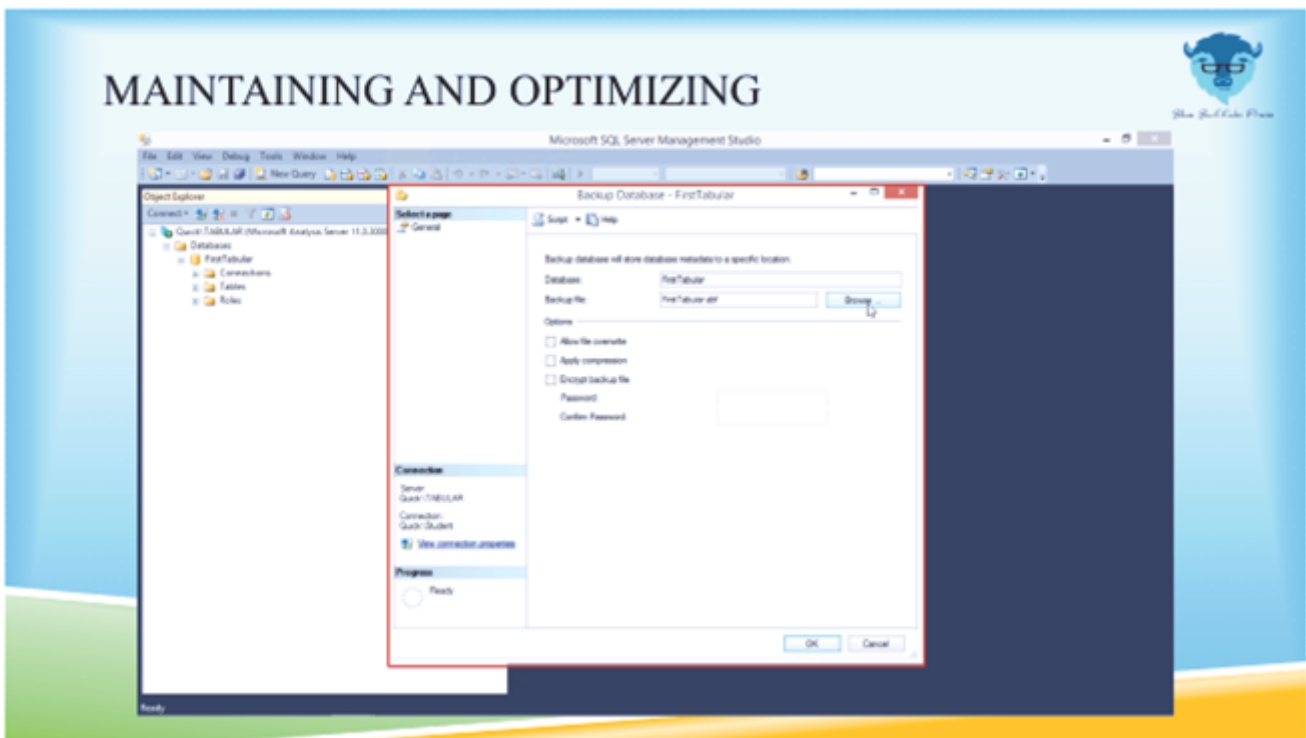
C.1.27 In the **Microsoft Excel** dialog box asking if you **Want to save your changes**, click **Don't Save**.

C.1.28 Close **Microsoft Visual Studio**.

C.1.29 In the **Microsoft Visual Studio** dialog box asking do you want to **Save changes to the following items**, click **Yes**.

## MAINTAINING AND OPTIMIZING





## BACKUP AND RESTORE

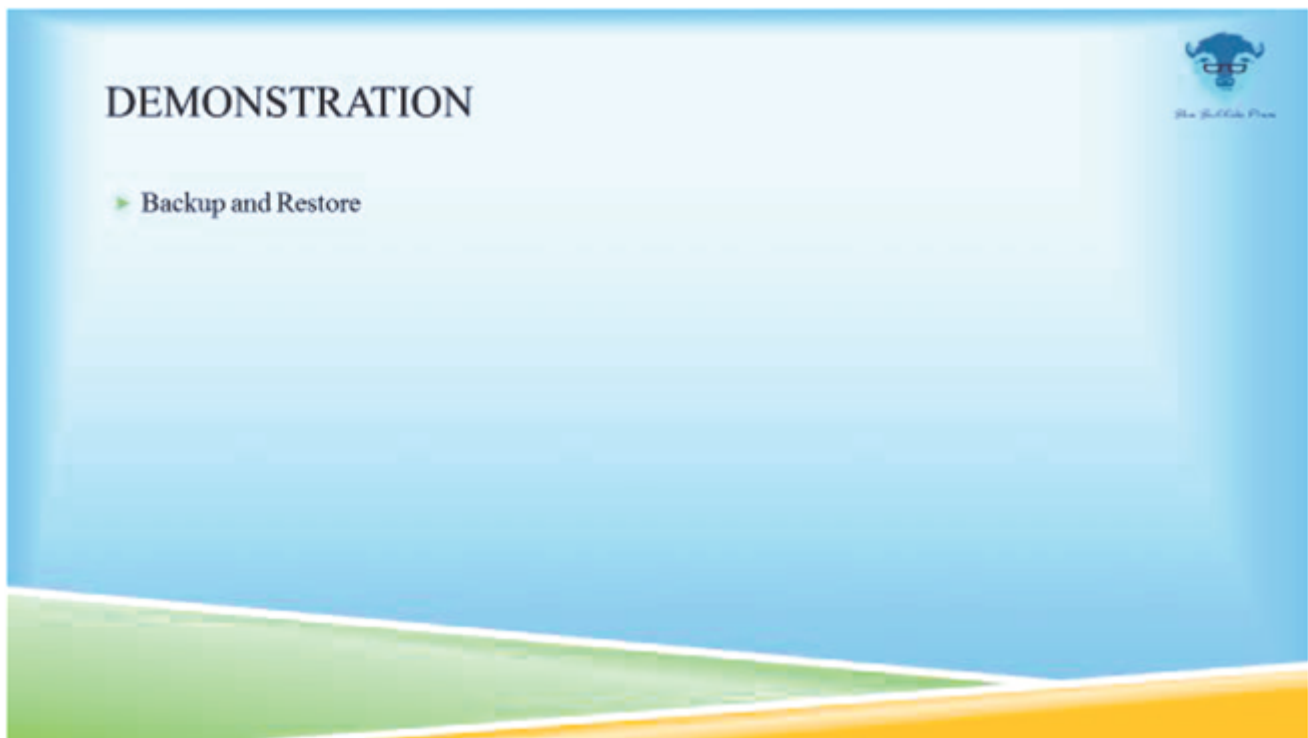
SSAS includes backup and restore so that you can recover a database and its objects from a particular point in time. Additionally, backup and restore is a valid technique for migrating databases to upgraded servers, moving databases between servers, or deploying a database to a production server. For data recovery purposes, if you do not already have a backup plan and your data is valuable, you should design and implement a plan as soon as possible.

The backup and restore commands are completed on a deployed Analysis Services database. For a full backup that includes source data, you'll need to back up the database which contains detail data.

Distinctively, if you are using ROLAP or DirectQuery database storage, detail data is stored in an external SQL Server relational database that is distinct from the Analysis Services database. Otherwise, if all objects are tabular or multidimensional, the Analysis Services backup will include both the metadata and source data.

## DEMONSTRATION

### VIDEO: BACKUP AND RESTORE



## EXERCISE D.1: BACKUP AND RESTORE

Objective: in this exercise we will backup and restore the FirstTabular database we created.

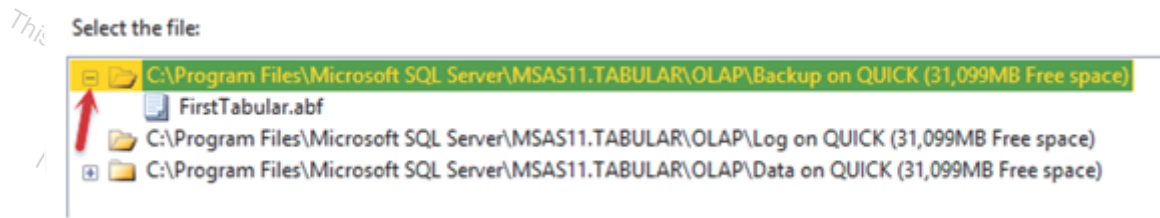
D.1.1 Navigate to the taskbar and start **SQL Server Management Studio**.

D.1.2 When **SQL Server Management Studio** opens, navigate to the **Connect to Server** dialog box, move to the **Server type** setting, use the corresponding drop-down arrow, and click **Analysis Services**.

- D.1.3 Locate the **Server name** setting, use the corresponding drop-down arrow, and click to select **QUICK\TABULAR**.
- D.1.4 Click **Connect**.
- D.1.5 Navigate to the **Object Explorer** pane on the left, and review the results noticing the icon identifying the connection.
- D.1.6 Expand **Databases** folder and notice **FirstTabular** is listed.
- D.1.7 Right-click **FirstTabular** database and click **Back Up....**
- D.1.8 In the **Backup Database – FirstTabular** dialog box, navigate to the **Encrypt backup file** option, and clear the check box.
- D.1.9 Navigate to the **Apply compression** option, and clear the check box.
- D.1.10 Click **Browse....**
- D.1.11 In the **Save File As** dialog box, review the current settings.
- D.1.12 Click **Cancel**.
- D.1.13 Back in the **Backup Database – FirstTabular** dialog box, review the other settings, and then click **OK**.
- D.1.14 Navigate to **Object Explorer** pane on the left, right-click **FirstTabular** database, and click **Delete**.
- D.1.15 In the **Delete Objects** dialog box, review the settings.
- D.1.16 Click **OK**.
- D.1.17 Move back to the **Object Explorer** pane and notice **FirstTabular** database is no longer listed.
- D.1.18 Right-click **Databases** folder, and click **Restore....**
- D.1.19 When **Restore Database** dialog box opens, review the current settings and options available.
- D.1.20 Locate the **Restore Source** section, move to the **Backup file** setting and click the corresponding **Browse....**

D.1.21 When the **Locate Database Files** dialog box opens, review the settings and options available.

D.1.22 Expand the first folder (**C:\Program Files\Microsoft SQL Server\MSAS11.TABULAR\OLAP\Backup on QUICK...**)



D.1.23 Click to select **FirstTabular.abf**.

D.1.24 Click **OK**.

D.1.25 Back in the **Restore Database** dialog box, review the remaining settings, and then click **OK**.

D.1.26 Move back to the **Object Explorer** pane on the left, right-click **Databases** folder, and click **Refresh**.

D.1.27 Expand **Databases** folder and notice **FirstTabular** is listed.

D.1.28 Expand **FirstTabular** database.

D.1.29 Expand **Tables** folder. Review the results.

D.1.30 Close all open windows.

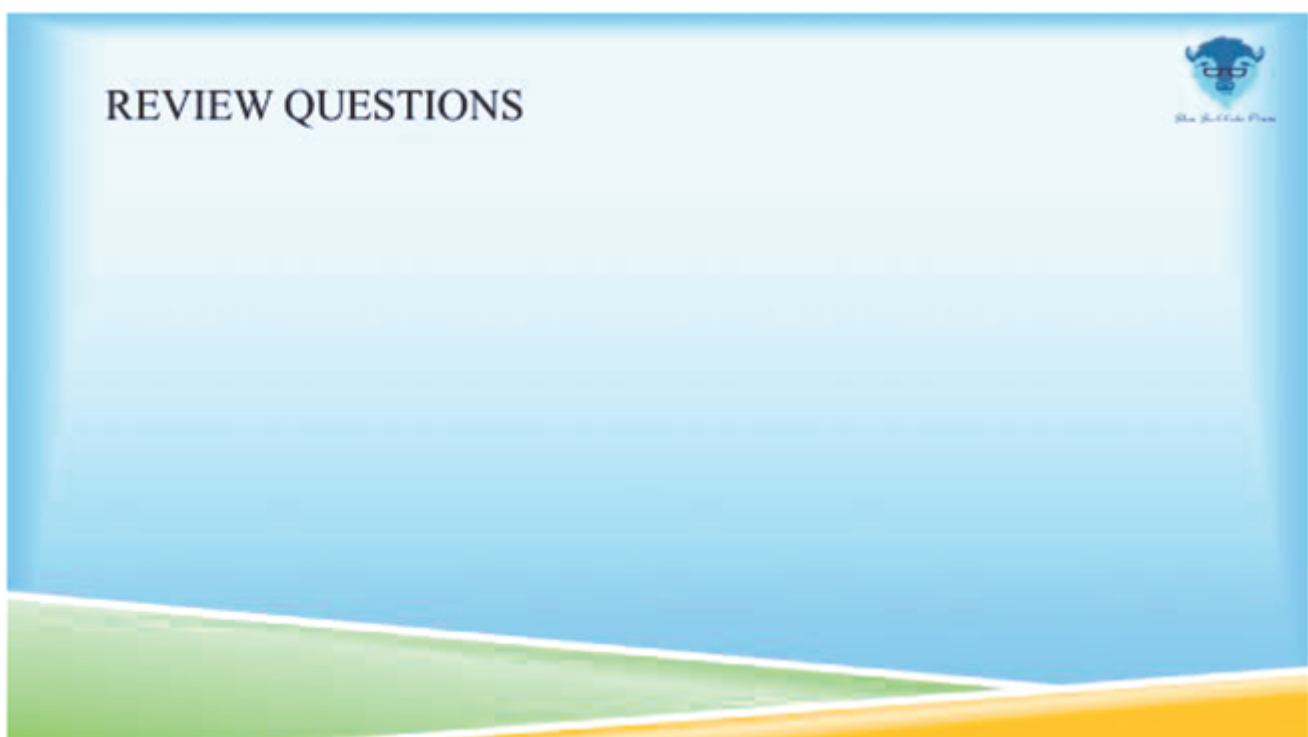
## MODULE REVIEW



## MODULE OBJECTIVE

Tabular models are in-memory databases in Analysis Services. In this module we will explore the tabular model, create a tabular project, deploy and query the solution using Data Analytic Expressions (DAX).

## REVIEW QUESTIONS



1. \_\_\_\_\_ models are in-memory databases in Analysis Services.
2. After you have finished designing a project in SSDT, it must be \_\_\_\_\_ to an instance of  
  
Analysis Services, which means SSDT executes a number of commands to create a new database in Analysis Services or alters the structure of an existing database.
3. The tabular model uses a column-oriented database called the \_\_\_\_\_ in-memory analytics  
  
engine, which in most cases offers significant query performance improvements.
4. True or False: High data cardinality means that the data values within a column can repeat.
5. The SSMS Browse window and the SSDT tabular project window both include the \_\_\_\_\_  
  
\_\_\_\_\_ button on their respective menus.

## REVIEW QUESTIONS ANSWERED





1. \_\_\_\_\_ models are in-memory databases in Analysis Services.
  - a. Tabular
2. After you have finished designing a project in SSDT, it must be \_\_\_\_\_ to an instance of Analysis Services, which means SSDT executes a number of commands to create a new database in Analysis Services or alters the structure of an existing database.
  - a. Deployed
3. The tabular model uses a column-oriented database called the \_\_\_\_\_ in-memory analytics engine, which in most cases offers significant query performance improvements.
  - a. xVelocity
4. True or False: High data cardinality means that the data values within a column can repeat.
  - a. False - Low data cardinality means that the data values within a column can repeat.
5. The SSMS Browse window and the SSDT tabular project window both include the \_\_\_\_\_ button on their respective menus.
  - a. Analyze in Excel