Module 5: Modeling Data

Contents:

Module overview

Microsoft® Power BI is making its mark in the self-service BI world—because it can quickly create visually stunning, interactive reports, and dashboards. Power BI provides a straightforward way to combine data from a wide range of sources into a single dataset, and work with that data to create cohesive reports. This module goes behind the scenes of the visualizations, and explores the techniques and features on offer to shape and enhance your data. With automatic relationship creation, a vast

library of DAX functions, and the ability to add calculated columns, tables, and measures quickly, you will see how Power BI creates attractive reports, while helping you find hidden insights into data.

Objectives

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

- Describe relationships between data tables.
- Understand the DAX syntax and use DAX functions to enhance your dataset.
- Create calculated columns, calculated tables, and measures.

Lesson 1: Relationships

This lesson explores the relationships between the tables in your data, why they are important, and how to create them.

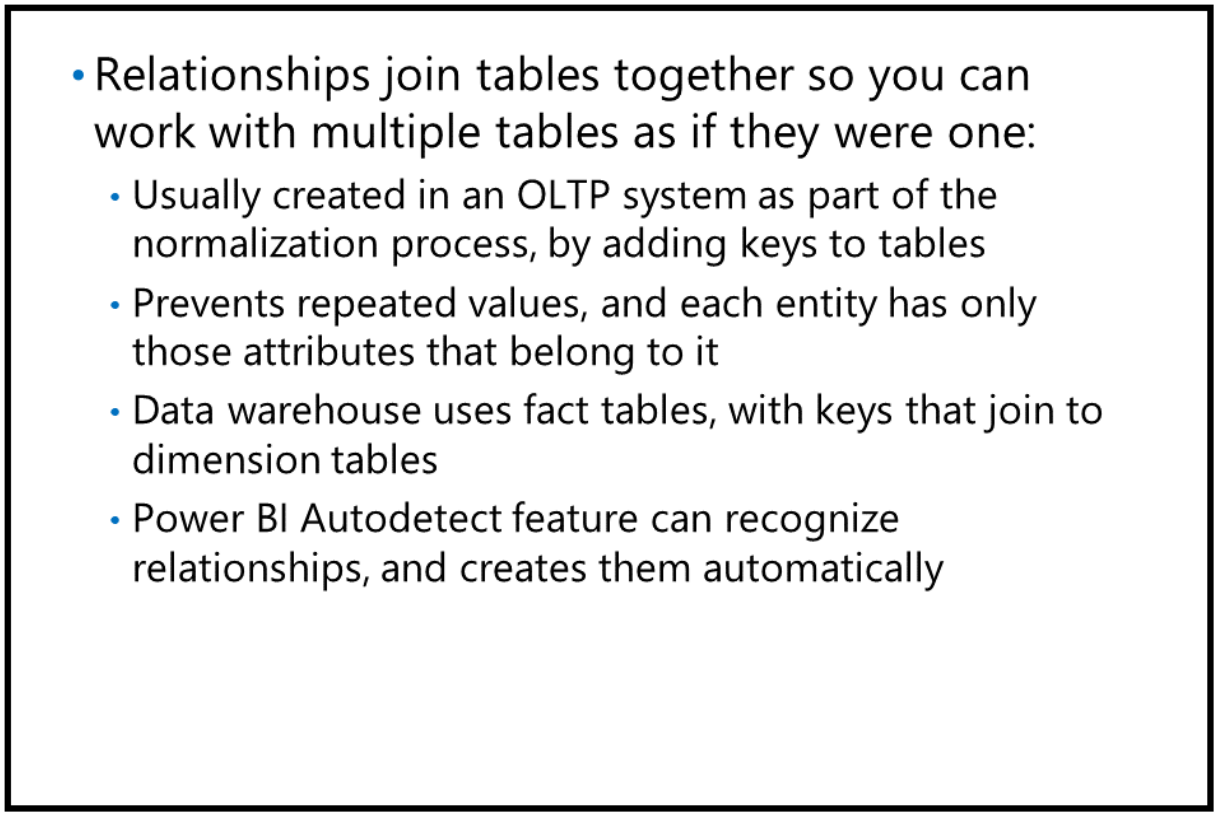
Lesson objectives

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

- Describe the purpose of relationships between tables.
- View relationships in the Power BI Desktop Power Query Editor.

- Create relationships using the Power Query Editor.
- Understand cardinality.
- Choose the correct cross filter direction in your relationships.

What are relationships?



Relationships exist to join tables together so that you can work with them as if they were one. If you are familiar with relational databases, such as Microsoft SQL Server®, or data warehouse databases such as SQL Server Analysis Services

(SSAS), you will understand the concept of relationships in Power BI, as this is much the same.

Relationships in an OLTP system

Relationships are usually created in an online transactional processing (OLTP), or relational database, as part of a normalization process. Normalization works at various levels, or forms, depending on how close to official normalization rules you want to adhere. Two of the main purposes of normalization are eliminating repeated data, and only including columns in a table, or entity, that are a direct attribute of that entity. For example, you would store your list of customers in a table with one row for each customer. Your Sales table would have a link back to the Customers table, using a key column such as CustomerID. This prevents you from repeating all the customer data, such as contact name, address, postcode, and so on, each time a customer places an order. Furthermore, when a customer updates their details, you only need to update one record, keeping your data consistent. The link from the Sales table to the Customer table using the CustomerID key is a relationship.

Relationships in a data warehouse

If you have worked with a data warehouse database, you know that a fact table connects to the dimension tables using keys. Although data stored in a star schema in a data warehouse is structured differently to data stored in an OLTP, or relational database system, the keys in both designs create relationships by joining tables together.

Table relationships in Power BI

The following table shows rows from the SalesOrderDetail table. Each row contains an order for a product. The SalesOrderID column values in this case are identical, so these rows are part of the same order. There is also a ProductID column in the table, linking the SalesOrderDetail table to the Product table.

SalesOrderID	ProductID	OrderQty	UnitPrice	LineTotal
43659	714	3	28.8404	86.521200
43659	716	1	8.8404	28.840400
43659	709	6	5.70	34.200000
43659	712	4	5.1865	10.373000
43659	711	2	20.1865	80.746000

The SalesOrderDetail table is related to the SalesOrderHeader table, shown next. There is one row in the SalesOrderHeader table for each order, though this order might comprise multiple rows in the SalesOrderDetail table. The CustomerID column links to the Customers table.

SalesOrderID	OrderDate	ShipDate	CustomerID
43659	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	29825

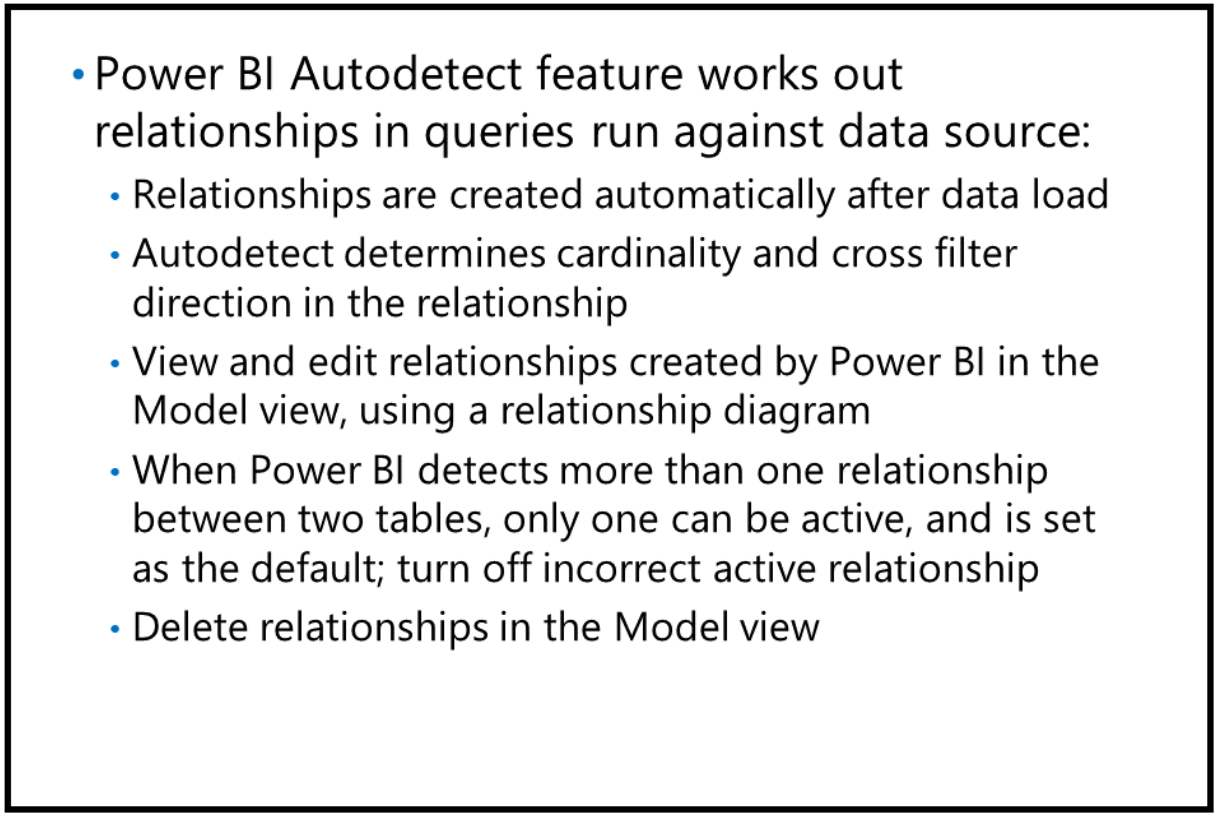
Traversing the tables using the relationships, the SalesOrderDetail table is related to the SalesOrderHeader table, and the Product table. In turn, the SalesOrderHeader

table is related to the SalesOrderDetail table, in addition to the Customers table. You can use these relationships to view the four tables as one, so you see all the products ordered by a customer as if they are in one table. This is useful for aggregating data across tables in visualizations.

Autodetect feature

When you import data into Power BI, the Autodetect feature operates in the background, and works out the relationships in your dataset. It also automatically sets the cardinality and cross filter direction, both of which are covered as topics in a later lesson. For much of the time, Power BI makes a good guess, correctly identifying related tables and creating the relationships for you. In this scenario, you might not have to do any further work to establish relationships between the tables.

Viewing relationships



When you import data into Power BI, queries are run against the data source to copy the data required to fulfill your modeling requirements for the dataset. While these queries are running, Power BI observes them to determine if there are relationships between the tables. After the data has finished loading, you view and manage the relationships that Power BI has created for you.

Power BI Desktop comprises three main views: Report, Data, and Model. You view the tables and column names in Report view, and add fields to visualizations. In Data view, you apply extensive modeling and formatting to the data, and you view the values in the tables. The Model view shows the tables and columns linking together those tables that are related. In the views pane, click **Model** to open a diagrammatic

view of the relationships in your model. The relationships appear the same in the Model view regardless of whether they have been created manually or by Power BI. All tables are included, even if they are not related to any others.

You can see information about the relationships, just by looking at the relationship diagram. Each relationship is represented by a line that joins the two tables. The arrow icon on the line indicates the cross filter direction of the relationship—either one arrow for **Single**, pointing in the direction of the filter, or two arrows when the cross filter direction is set to **Both**. At the end of each relationship line, where it joins to either table, another icon represents the cardinality. A star icon (*****) represents **Many**, and a **1** represents **One**, for either a Many to One (*:1), One to One (1:1), or One to Many (1:*) relationship. When you click a relationship line, the related columns in either table are highlighted with a black border, for quick identification.

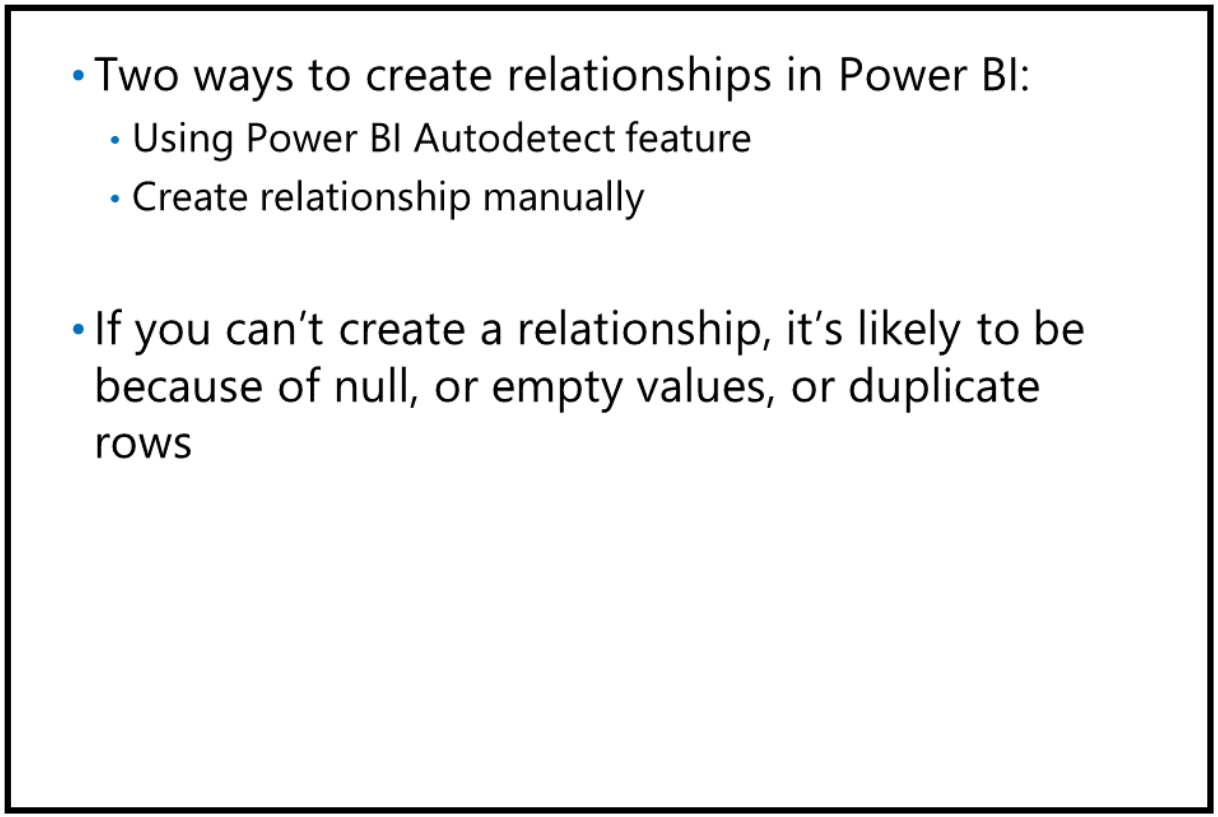
Editing relationships

When a relationship line has focus, it is highlighted in yellow. Double-click the line to open the **Edit relationship** dialog box. You can also click **Manage Relationships** from the **Relationships** group on the **Home** tab, to view the **Manage relationships** dialog box. From the **Manage relationships** dialog box, you create new relationships, run the Autodetect feature, and then edit and delete existing relationships. In the **Manage relationships** dialog box, double-click a relationship to open the **Edit relationship** dialog box. This opens the same view as double-clicking a relationship line. In the **Edit relationship** dialog box, you can change the related table and column, switch cardinality between Many to One (*:1), One to One (1:1), or One to Many (1:*), and toggle the cross filter direction between **Single**, or **Both**. You can also turn on or off the **Make this relationship active** option. When the Power BI

Autodetect feature runs, it sometimes finds more than one relationship between two tables. In this case, only one of the relationships is set to active, and this becomes the default relationship. You can use this setting when the active relationship is incorrect.

You can also delete relationships. Click the relationship line that joins two tables so it is highlighted yellow. Right-click the relationship line, and then click **Delete**.

Creating relationships



There are two ways to create relationships in Power BI. You use the Autodetect feature and Power BI works out the relationships for you, or you create them manually.

Creating relationships using Autodetect

When data is imported into the model, Power BI automatically creates relationships. If you then create calculated tables, or use Enter Data to add new tables, relationships will not exist. You can run the Autodetect feature from the **Home** tab: in the **Relationships** group, click **Manage Relationships**, and then in the **Manage relationships** dialog box, click **Autodetect**. Power BI runs the Autodetect feature to look for new relationships and by default, also determines the cardinality, cross filter direction, and active relationships. However, bear in mind that the Autodetect feature is a best guess, and might need adjusting after it runs.

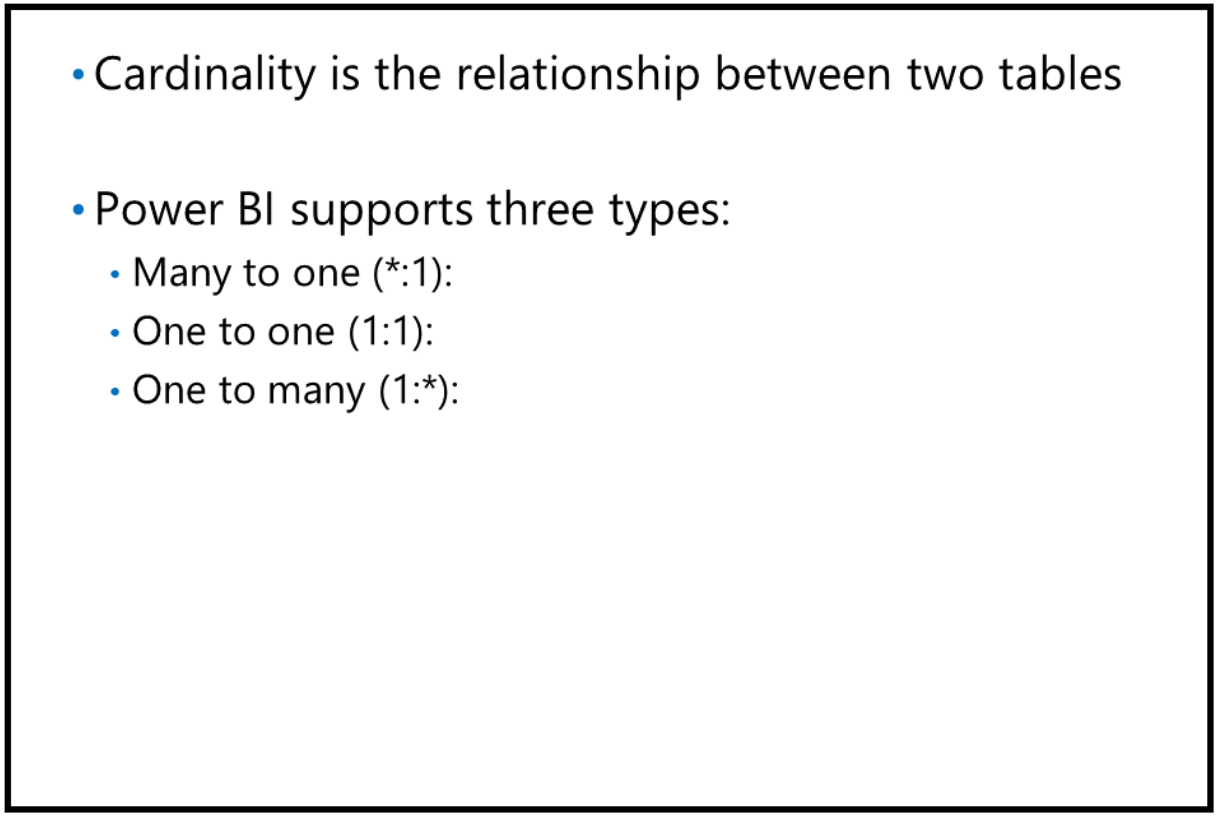
Creating relationships manually

The quickest way to create a relationship between two tables is to drag the column from the first table, to the related column in the second table where you want to join. If the data is valid for creating a relationship, the columns are connected. You can also click **Manage Relationships** from the **Relationships** group on the **Home** tab. This opens the **Manage relationships** dialog box. Click **New** to open the **Create relationship** dialog box. Select the first table in the upper table list—this displays a preview of the table. Click the column you want to use in the relationship, and then select the table to join to in the lower table list. This displays a preview of the second table. Again, click the column where you wish to join. Power BI automatically

determines the cardinality and cross filter direction of the relationship. This is usually correct, so unless future data is likely to change this, click **OK** to create the relationship. Otherwise, change the cardinality and cross filter direction settings, and then click **OK**. Click **Close** to close the **Manage relationships** dialog box.

You might find that you cannot create a relationship between your tables. This might be due to columns with null, or empty, values or duplicate data. You remove rows with null or blank values by using the filter in the query tab, or replace them with valid data, including “NULL”. Removing rows might affect calculations, yet using NULL can create artificial relationships. If you use the latter approach, make sure you include appropriate filters in your visualizations.

Cardinality



In data modeling, cardinality refers to the relationship that one table has with another. In Power BI modeling, the cardinality can be one of the following three types:

1. **Many to one (*:1)**: Many to one is the default type in Power BI, and generally the most common. Many to one means that one table can have more than one instance of the value used in the column to join to the other table. The other table would have only one value. For example, your Sales table has many instances of the CustomerID because the customer has placed multiple orders, and joins to the Customers table using CustomerID. The Customers table has one instance of the CustomerID, or rather one row for each customer. This is also common for lookup tables, where you might have a list of states, or

countries. Each state or country is listed only once, but the instance exists multiple times in the Customers (or other) table.

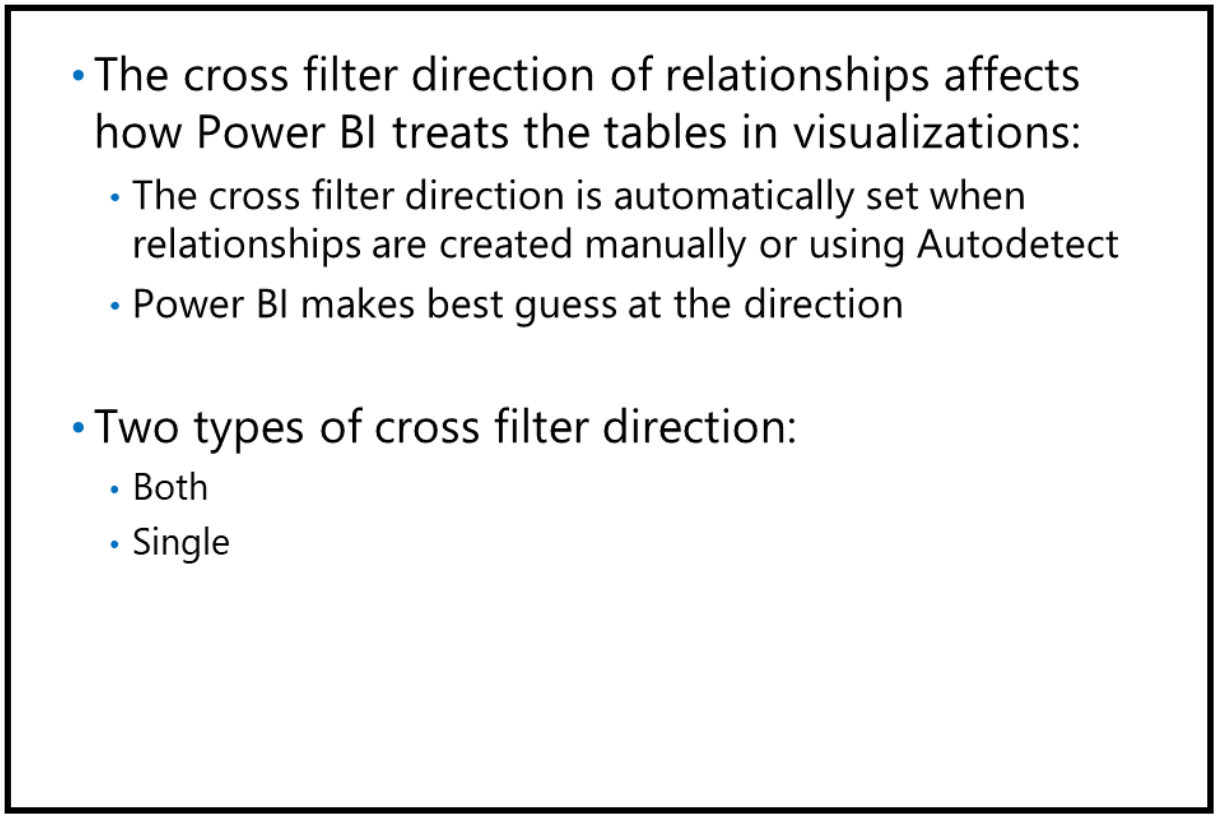
2. **One to one (1:1)**: In a one to one relationship, both tables in the relationship have one instance of a value. In relational database systems, one to one is not as common as many to one, but one of its uses can be to split up larger tables. For example, you might have an Employees table with an EmployeeID column and other columns for the employee name, address, date of birth, phone number, and salary. This data is used frequently by the Human Resources department. You have another table called EmployeeAdditionalDetails, with a row for each employee, and an EmployeeID column to join to, from Employees. The EmployeeAdditionalDetails table contains less used fields such as next of kin, number of dependents, training information, and qualifications. This would be a one to one relationship.

3. **One to many (1:*)**: This is the same as many to one, except the position of the tables is reversed in the relationship. In this case, you could have your Customers table with one row for each customer, related to many orders in the Sales table.

Having relationships between your tables prevents the need to flatten the tables, or combine them together into a single table, before importing the data into the model. Power BI uses an Autodetect feature to work out the cardinality of relationships, for both those you create manually, and those it has created automatically. You change the cardinality by clicking **Manage Relationships** from the **Home** tab. In the **Manage relationships** dialog box, double-click a relationship, or click **Edit**, to open the **Edit**

relationship dialog box and select from the **Cardinality** list to change it. Click **OK**, and then click **Close**. In the Model view, double-click a relationship in the diagram to open the **Edit relationship** dialog box. Change the cardinality, and click **OK**.

Cross filter direction



The cross filter direction of the relationships in your dataset affects how Power BI treats the tables in visualizations in your reports. When you manually create a relationship, or the Autodetect feature generates the relationship for you, Power BI makes a best guess at the cross filter direction. The direction can be Both, or Single:

- **Both**: Both is the most common, and the default. When you apply filtering, the two tables are considered as one table for aggregating data in a visualization. The Both cross filter direction is ideal for a table that is related to numerous lookup tables, such as a fact table in a star schema. For example, a FactInternetSales table is surrounded by the related lookup tables, such as DimCustomer, DimCurrency, DimDate, DimProduct, DimPromotion, and DimSalesTerritory. The layout of the tables in the Model view might reflect this as a snowflake shape. In this example, there is a mix of cross filter direction types. However, if you have a lookup table that is related to more than one (non-lookup) table, you might want to set the cross filter direction to Single. For example, if you have two tables with values for aggregating, that are unrelated, but both reference a Country lookup table, then set the cross filter direction to Single. This prevents aggregations from including data that is not actually connected. The FactInternetSales table has a many to one relationship with DimCustomer, using a cross filter direction of Both. With this, you can use both tables as one in your visualizations. The DimCurrency table is also related to the FactInternetSales table with a many to one relationship, but this has a Single cross filter direction, preventing any other tables that use this lookup from inclusion in aggregations.
- **Single**: With a Single cross filter direction, the filters in related tables operate on the table where the values are aggregated. If you have imported data from Power Pivot for Excel® 2013 or earlier, all relationships have Single cross filter direction.

You manually change the cross filter direction by clicking **Manage Relationships** from the **Home** tab. In the **Manage relationships** dialog box, double-click a

relationship, or click **Edit**, to open the **Edit relationship** dialog box, and select **Both**, or **Single** in the **Cross filter direction** list. Click **OK**, and then click **Close**. Alternatively, in the Model view, double-click a relationship in the diagram to open the **Edit relationship** dialog box.

> **Note:** The direction of the cross filter is displayed as an arrow for Single, or a double arrow for Both on the relationship line. The Single arrow points in the direction of the filter.

Demonstration: Working with relationships in Power BI

In this demonstration, you will see how to:

- Import a data extract into Power BI.
- View and edit the relationships created automatically.
- Add new relationships.

Check your knowledge

Select the best answer

Which of the following statements is false?

The Power BI Autodetect feature works out the cardinality of the relationship between two tables.

When querying the data source, Power BI automatically determines the relationships, and creates them.

The Sales table is related to the Customer table using the CustomerID column. There are many orders in the Sales table for each customer, and one row in the Customers table for each customer. This is a many to one relationship.

The Employees table has one row for each employee, and is related to the EmployeeAdditionalDetails table using the EmployeeID column. There is one instance of each employee in the EmployeeAdditionalDetails table. This is a one to one relationship.

After Power BI automatically creates a relationship, you cannot change the cardinality or cross filter direction options.

Check answer Show solution Reset

Lesson 2: DAX queries

In this lesson, you will learn about DAX, the syntax structure, and how to use functions.

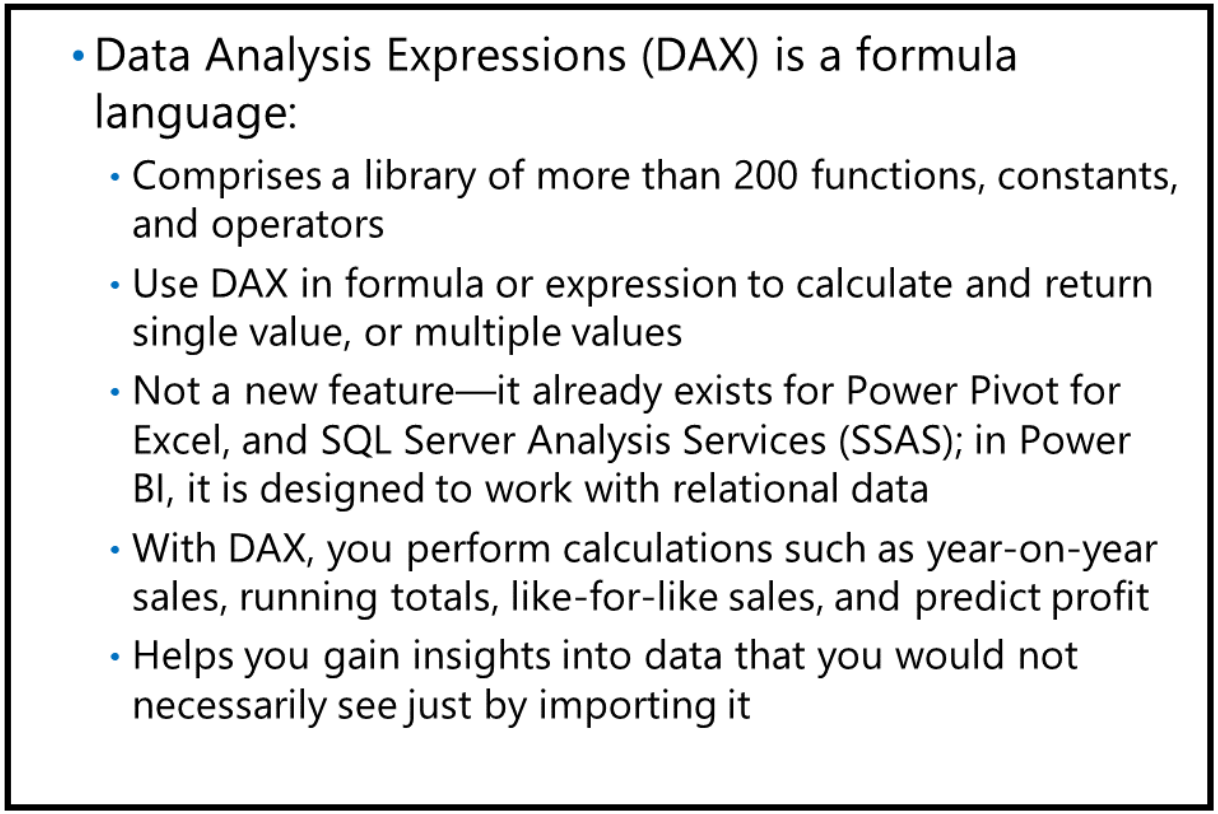
Lesson objectives

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

- Describe DAX and what it is used for.
- Understand the DAX syntax so you can create queries.
- Write DAX queries using functions.

- Understand the importance of context when using DAX.

What is DAX?



Data Analysis Expressions (DAX) is a formula language that comprises a library of more than 200 functions, constants, and operators. You use DAX in a formula or expression, to calculate and return a single value, or multiple values. DAX is not new —you may have used it in Power Pivot for Excel or SQL Server Analysis Services (SSAS). If you have used Excel formulas, you will discover some similarity; however, DAX functions are designed specifically to work with relational data, which is what

you work with in your Power BI datasets. DAX is commonly used in calculated columns and measures, both of which are covered in more detail in the next lesson.

Why use DAX?

You import your data into Power BI Desktop and can begin creating reports straightaway. However, while this certainly presents your data visually, and facilitates interaction using the drill-through feature, what if you want to include year-on-year sales growth, or running totals based on monthly sales, or perhaps predict profit for next year? With DAX formulas, you can do this, and they can help you find the insights you want to extract from your data to make it more useful. For example, you might want to compare sales so far this year, like-for-like with last year. If the current month is May, you only want to compare that part of the previous year. DAX provides a function for this, as shown in the following code. This is not something that is easy to do without DAX:

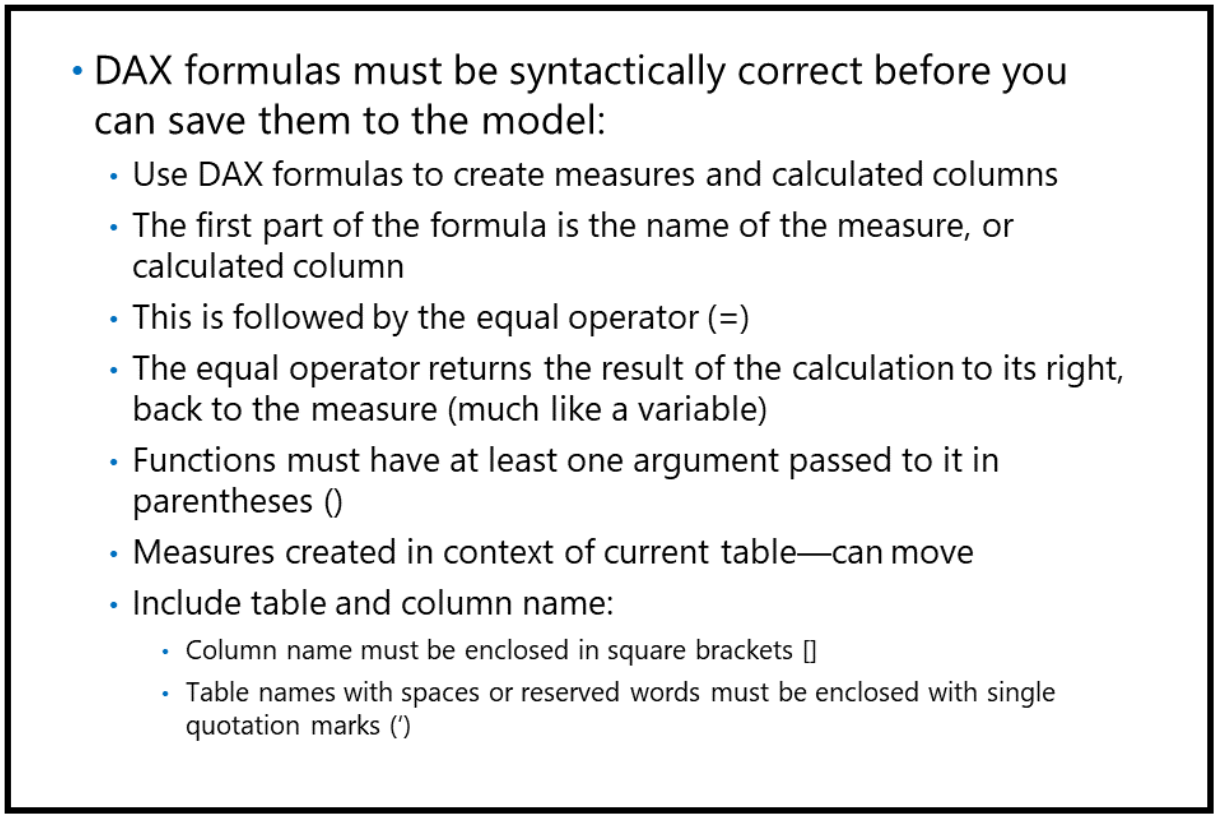
Calculate Sales for the Same Time Period Last Year

The following DAX formula returns the sales from last year, using the sales dates for the current year, to provide a like-for-like comparison:

```
Last Year Sales = CALCULATE ([Total Sales],
SAMEPERIODLASTYEAR('Date'[Date]))
```

The key to understanding and using DAX is learning the concepts of the syntax for structuring your formulas, the functions you use to make calculations, and context. These concepts are covered in detail in the remainder of this lesson.

Syntax



The DAX formulas that you write must be syntactically correct; otherwise, Power BI gives you a syntax error message. Therefore, it is important to understand how to structure your expressions. The following code shows an example of a typical formula you might use in Power BI to create a measure:

DAX Formula to Calculate Total Sales

The following DAX formula adds together the values in the LineTotal column of the InternetSales table, and returns a measure named Total Sales:

```
Total Sales = SUM(InternetSales[LineTotal])
```

The first part of the formula is Total Sales. This example uses a measure, but it could be a calculated column, and you can rename both in the Report view. The name can contain spaces, in addition to symbols such as the percentage sign (%). The name of the measure is followed by the equal operator (=). The equal operator returns the value of the calculation to the right of it, to the measure, in much the same way as you assign values to a variable. This example uses the SUM function, and adds up all values in the argument you pass to it in the parenthesis (). An argument passes a value to the function, and all functions must have at least one argument. In this case, the argument is the LineTotal column in the InternetSales table.

When you write DAX formula, Power BI creates the new measure in the context of the current table. However, this is completely flexible, and you can move the measure to whichever table you want. Select the measure in the Fields pane, and then select the **Modeling** tab. In the Properties group, click **Home Table**, and then select the table where you want to move the measure. If you create the above example in the InternetSales table, you can move it elsewhere without affecting the formula. Because you passed the table and column name as the argument, this creates

independence—the function knows exactly which values to operate on, regardless of its home table.

> **Note:** When you refer to a column in a formula, and include the table name, this is known as a "fully qualified column name". You exclude the table name when the measure refers to a column in the same table in which it also resides; however, it's good practice to include it. While this can lengthen formulas that reference many columns, it provides clarity and the reassurance that you are referencing the correct columns—you can also create measures that span multiple tables, and move them as required.

If your table name contains spaces, reserved keywords, or disallowed characters, enclose the name using single quotation marks. Table names containing characters outside of the ANSI alphanumeric character range will also need enclosing with single quotation marks. The column name is always encased with square brackets; for example, [LineTotal].

You type DAX formula into the formula bar. There are two buttons to the left of the bar, a cross (X) icon, and a tick icon. The cross icon cancels the measure, and removes any work without saving. The tick icon validates your syntax, and enters your new measure into the model.

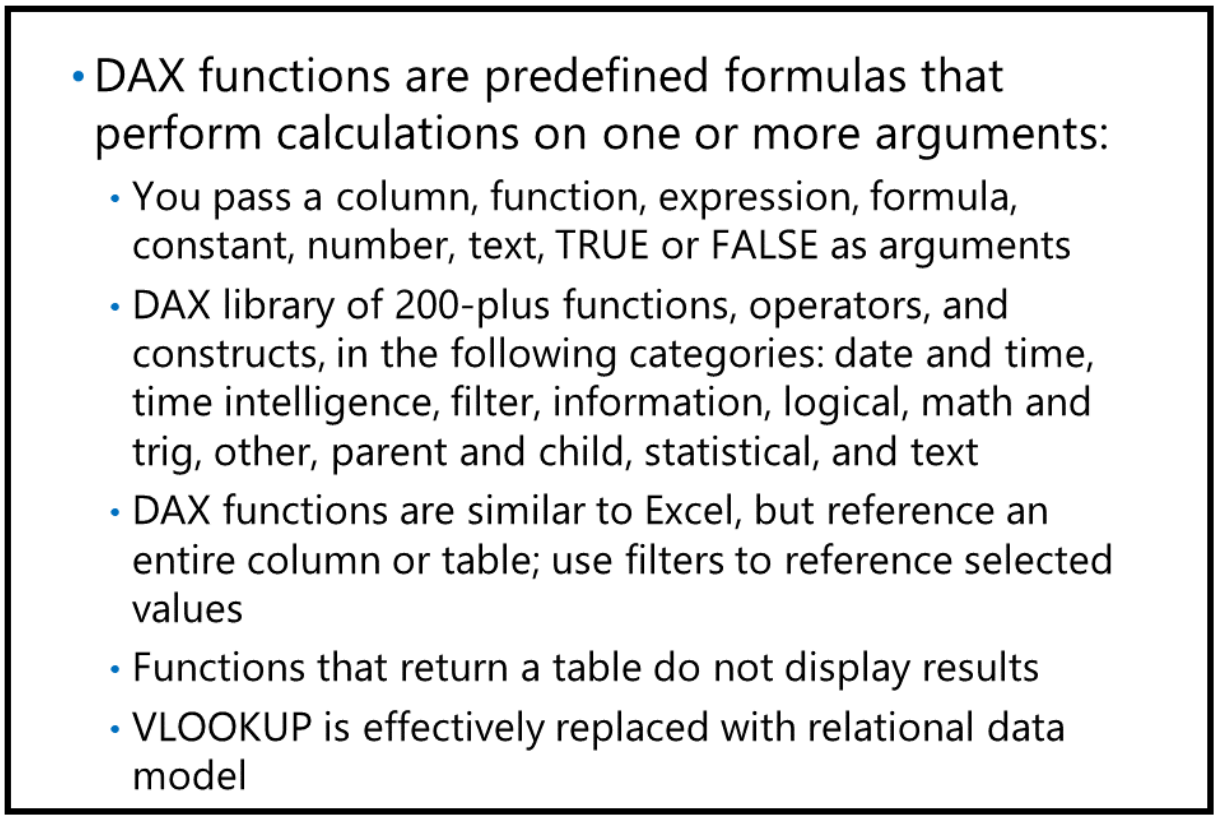
If you are already familiar with Power BI, you might know that numerical fields are automatically calculated, and wonder why you would want to create the above measure, because Power BI will sum this for you. By adding this measure, you use it as an argument for another formula, meaning you can create all the calculations you

require within your dataset. For more information on DAX syntax, see *DAX syntax* in Microsoft Docs:

DAX syntax

http://aka.ms/tl7369

Functions



Functions are predefined formulas that perform calculations on one or more arguments. As you learned in the previous topic, you can pass a column as an

argument, in addition to using other functions, expressions, formulas, constants, numbers, text, and TRUE or FALSE values. The DAX library of more than 200 functions, operators, and constructs, is segmented into the following 10 categories:

- **Date and time**: similar to the date and time functions used in Excel, but based on the datetime data types used by Microsoft SQL Server. Date and time functions include DATEDIFF, DAY, EOMONTH, NOW, WEEKDAY, WEEKNUM, and YEAR.
- **Time-intelligence**: with these functions, you create calculations using date and time ranges combined with aggregations. This is useful for building comparisons across time periods. Time intelligence functions include CLOSINGBALANCEMONTH, DATEADD, NEXTQUARTER, NEXTYEAR, PREVIOUSMONTH, SAMEPERIODLASTYEAR, and TOTALYTD.
- **Filter**: with filter functions, you return specific data types, look up values in related tables, or filter by related values. The functions work by using tables and the relationships between them. Filter functions include CALCULATE, FILTER, ISFILTERED, RELATED, RELATEDTABLE, and VALUES.
- **Information**: information functions evaluate a table or column provided as an argument to another function, and inform you if the value matches the expected type. Information functions include ISBLANK, ISERROR, ISEVEN, ISNUMBER, ISTEXT, LOOKUPVALUE, and USERNAME.
- **Logical**: these functions return information about the value in your expression. Logical functions include FALSE, IF, IFERROR, NOT, OR, and TRUE.
- **Math and trig**: similar to the mathematical and trigonometric functions in Excel,

math and trig functions perform a wide variety of calculations. Functions include ABS, ASIN, CEILING, CURRENCY, DEGREES, EVEN, FLOOR, ODD, PI, ROUND, ROUNDDOWN, ROUNDUP, SQRT, SUM, and TRUNC.

- **Other**: these functions are unique and do not fall into any of the other categories. They include EXCEPT, GROUPBY, INTERSECT, NATURALINNERJOIN, UNION, and VAR.

- **Parent and child**: parent and child functions work on data that is presented in a parent/child hierarchy in the data model. Parent and child functions include PATH, PATHCONTAINS, PATHITEM, PATHINREVERSE, and PATHLENGTH.

- **Statistical**: statistical functions are used to perform aggregations, such as SUM, MIN, MAX, and AVERAGE. With DAX, you can filter a column prior to aggregating, and create aggregations based on related tables. Further functions include COUNT, COUNTBLANK, COUNTROWS, CROSSJOIN, MEDIAN, ROW, SIN, TAN, and TOPN.

- **Text**: text functions operate on string values. You can use them to search for text within a string; return a substring; format dates, times, and numbers; concatenate strings. Text functions include CONCATENATE, FIND, LEFT, LEN, LOWER, REPLACE, RIGHT, SEARCH, TRIM, and UPPER.

For a full list of DAX functions, and examples of how to use each function, see *DAX function reference* in Microsoft Docs:

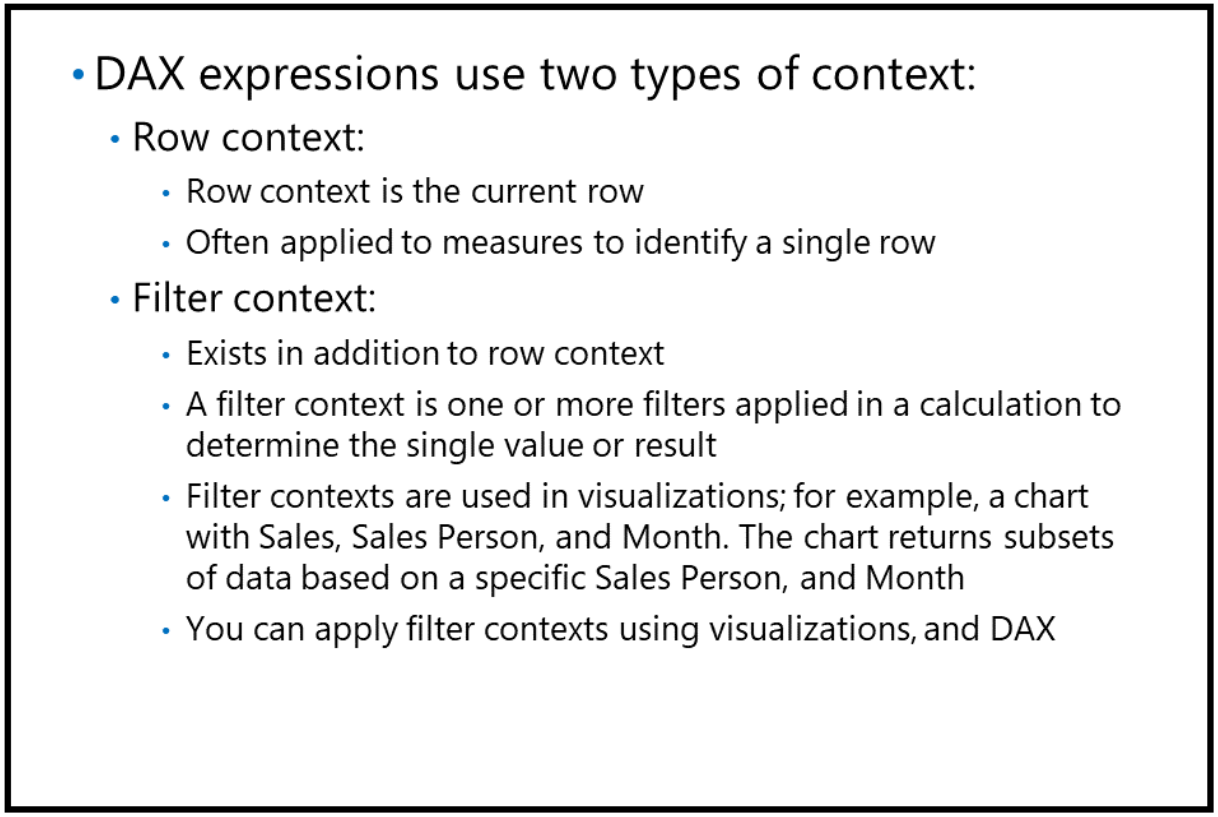
DAX function reference

http://aka.ms/lrf8p9

If you have been using Excel functions, DAX functions might look familiar. However, DAX functions differ in the following ways:

- DAX functions reference an entire column or a table. To use selected values from a table or column, you include filters in your formula.
- If you want to customize a calculation to work on a row-by-row basis, use functions to utilize the current row value, or related value as an argument.
- If you use one of the DAX functions that returns a table, rather than a single value, the table is not displayed—it's used to provide input for another function. For example, return a table and count the values, count distinct values, or filter columns and aggregate the values.
- With the time intelligence functions, you define or select date ranges, and then perform calculations on them.
- Instead of using a VLOOKUP, as you would in Excel, DAX functions accept a column or table as a reference. In Power BI, you work on a relational data model, so finding values in another table is straightforward because you can create relationships, and might not actually need a formula.

Context



Context is an important concept to understand if you want to write expressions that return the results you expect. In DAX, there are two types of context: row context, and filter context:

- **Row context**: you can think of row context as the current row. When a formula includes a function that uses filters to identify a single row in a table, this is considered row context. The function applies a row context to each row in the table to which the filter is applied. This type of context is often applied to measures.

- **Filter context**: filter context exists in addition to row context. A filter context is one

or more filters applied in a calculation, which determine a single value or result. You use a filter context to reduce the values that are included in a calculation. The filter specifies the row context, and also a particular value or filter, in that row context. Filter contexts select subsets of data. If you have a visualization in your report that includes Sales, Sales Person, and Month, the filter context works on subsets of data to return Sales by a specific Sales Person and Year. You can apply filter context by using filters this way in your reports, or by using DAX.

The following measure demonstrates how row context and filter context operate on the calculation in the formula. A new measure is created and named UK Sales. The CALCULATE function evaluates the expression in brackets, in a context set by the filters. The first argument in the expression is the measure [Total Sales], which has the formula, Total Sales = SUM(Sales[Revenue]). The comma separates the first argument from the filter argument. In this formula, the referenced column [Country], in the Customers table, sets the row context. Each row in the Country column specifies a country, such as France, Germany, UK, or US. This code filters on the UK, providing the filter context.

Using Row Context and Filter Context in a Measure

The following code is an example of a measure with the Country column in the Customers table as the row context, and the UK value as the filter context:

```
UK Sales = CALCULATE([Total Sales], Customers[Country] = "UK")
```

This formula uses Total Sales, and applies a filter of UK, so only the sum of UK sales is returned in the result. DAX is powerful in its ability to reference a selected value from a related table.

Demonstration: Using row and filter context in DAX formulas

In this demonstration, you will see how row and filter context works with measures.

Check your knowledge

Select the best answer

You want to concatenate and manipulate columns containing string data. Which of the following functions will not be compatible for working with text?

- CONCATENATE
- MEDIAN
- REPLACE
- TRIM
- UPPER

Check answer Show solution Reset

Lesson 3: Calculations and measures

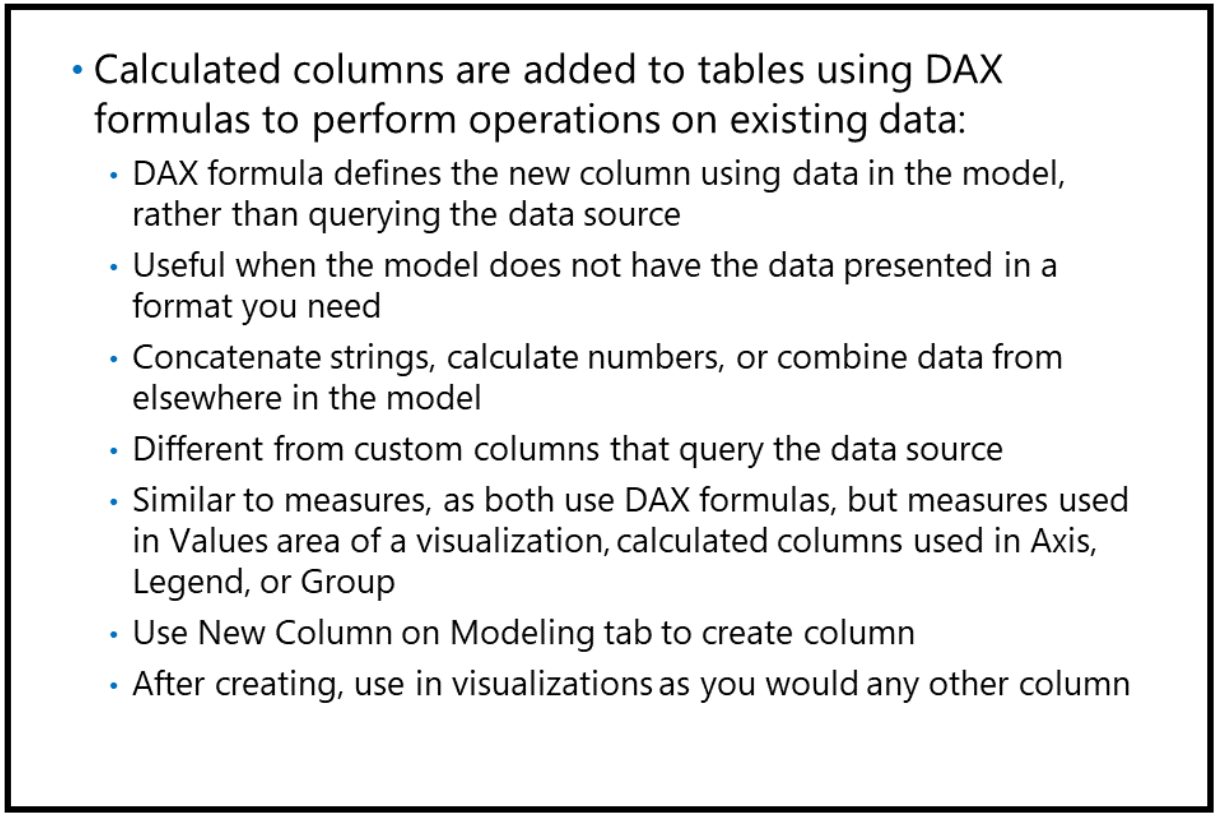
In this lesson, you will see how to manipulate your data using calculated columns and calculated tables, and learn how measures provide additional insights into your data.

Lesson objectives

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

- Add calculated columns to your tables.
- Create a new calculated table within your dataset.
- Add measures to your queries to deliver insights into your data.

Calculated columns



Calculated columns are added to your tables by applying DAX formulas to your existing data. The DAX formula defines the values in the new column, rather than querying the data source to create the column. Calculated columns are useful when the data source does not contain data presented in a format that you want. You can concatenate strings or multiple numbers together, combining data from anywhere in the model, to create a calculated column.

Calculated columns differ from custom columns, because they use data that already resides in the model. They are similar to measures, as both measures and calculated columns use a DAX formula, but the difference is in how they are used. Generally, measures are used in the Values area of a visualization, to calculate the results

based on other columns used in the Axis, Legend, or Group area of the visualization. Calculated columns are used for the fields you want to add to the Axis, Legend, or Group.

In Power BI Desktop, you use the **New Column** button on the **Modeling** tab to create a calculated column, or right-click the table name in the FIELDS pane and select **New column**. This opens the formula bar where you can type your DAX formula, and press Enter to create it. By default, Power BI names the new column as Column, but you can change this by typing in a new name. The following example creates a new column called Full Name, concatenating existing fields together.

Create a Calculated Column Using Existing Data

The following code example concatenates the First Name and Last Name fields into a new calculated column called Full Name:

```
Full Name = [First Name] & " " & [Last Name]
```

The above code does not include the table names, so these are classed as nonqualified column names. The columns exist within the Customers table, so they do not have to be qualified. In a small dataset, with no possibility of duplicate names in other tables, this is less of an issue, but it is considered good practice to include the table name for clarity. If you referred to a column in another table, then you must fully qualify the column. The following example uses the RELATED function to look up a value in another table.

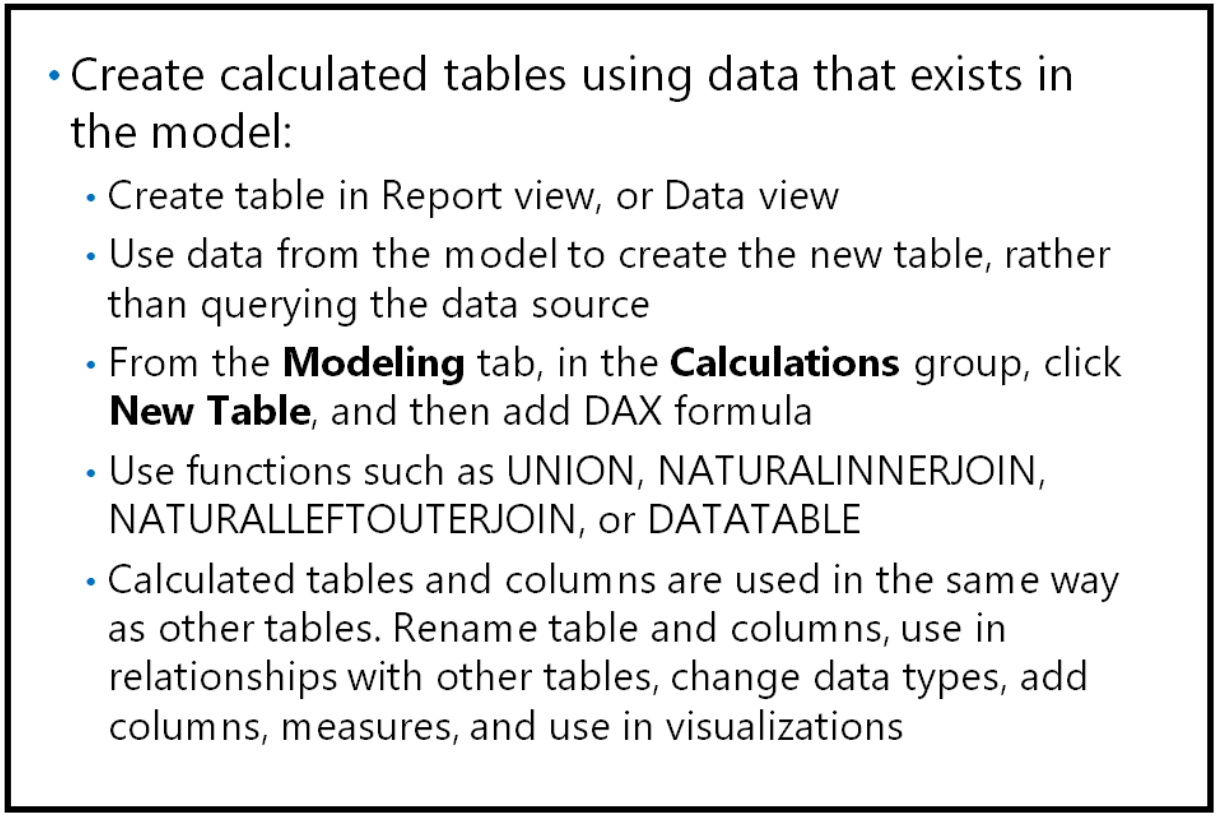
Create a Calculated Column Using the RELATED Function

The following code example returns the related Region value for the City column in the Customers table:

```
Location = RELATED(Countries[Region]) & “, “ & [City]
```

After creating a calculated column, it appears in the FIELDS pane and behaves in the same way as other columns. However, you can identify calculated columns by the icon next to the name. Calculated columns can have any name you want, and added to visualizations in exactly the same way as other columns.

Calculated tables



Like calculated columns, calculated tables are created using data that already exists in the model—you use a DAX formula to define the values in the table. Tables are created in both the Report view, and the Data view in Power BI Desktop. Calculated tables work well for intermediate calculations, and data that you want to be stored in the model, rather than calculated when the data source is queried.

To create a calculated table, open Power BI Desktop and the report with the dataset where you want to add the table. Click **Modeling**, and then in the **Calculations** group, click **New Table**. The formula bar opens and, by default, is populated with Table =. You can overwrite the word Table to give your table a better name. Write your DAX formula to the right of the equal sign, which creates your table. For

example, you could use a union, inner, left, or cross join function in your DAX to create the table. The following example creates a calculated table using the UNION function:

Combine Existing Tables with UNION to Create Calculated Table

The following code combines the existing NorthAmericanSales and EuropeanSales tables into one, to create a calculated table named Global Sales:

```
Global Sales = UNION (NorthAmericanSales, EuropeanSales)
```

When using the UNION function to combine two tables into one new calculated table, the tables must have the same number of columns. The columns are combined on their position in the table, so make sure the column order matches between the two tables. UNION includes duplicate rows that exist in both tables. If you want to remove duplicate rows, open Power Query Editor, and from the **Reduce Rows** group on the **Home** tab, click **Remove Duplicates**. The new table has the same column names as the first table, so in the preceding example, the UNION would take the names of the columns in the NorthAmericanSales table. The order of the columns is also taken from the first table, and related tables are not included in the union.

While UNION appends rows from one table to another, you merge columns using one of the join functions. You can use NATURALINNERJOIN, or NATURALLEFTOUTERJOIN, to merge the columns of two tables that have a related column. The following example joins the Customers table to the Sales table on the

CustomerID column, which is included in both tables. The columns from the Sales table are added to the right of the Customers table columns, to create the Customer Sales table.

Create Calculated Table Using the NATURALINNERJOIN Function

The DAX function uses the NATURALINNERJOIN function to create a new table called Customer Sales, which adds the columns from the Sales table to the columns from the Customers table:

```
Customer Sales = NATURALINNERJOIN (Customers, Sales)
```

In the preceding example, only rows with matching values in both tables are added to the new calculated table. To include all rows in the Customers table, regardless of a match in the Sales table, use NATURALLEFTOUTERJOIN instead. When using a join, the columns you are joining on must have the same data type.

Use the DATATABLE function in your DAX formula to create a new table, set the data types of the columns, and insert data. It is best to create your calculated tables in the Data view as you can view the new table immediately. The following code creates a Countries table, and adds values to the table:

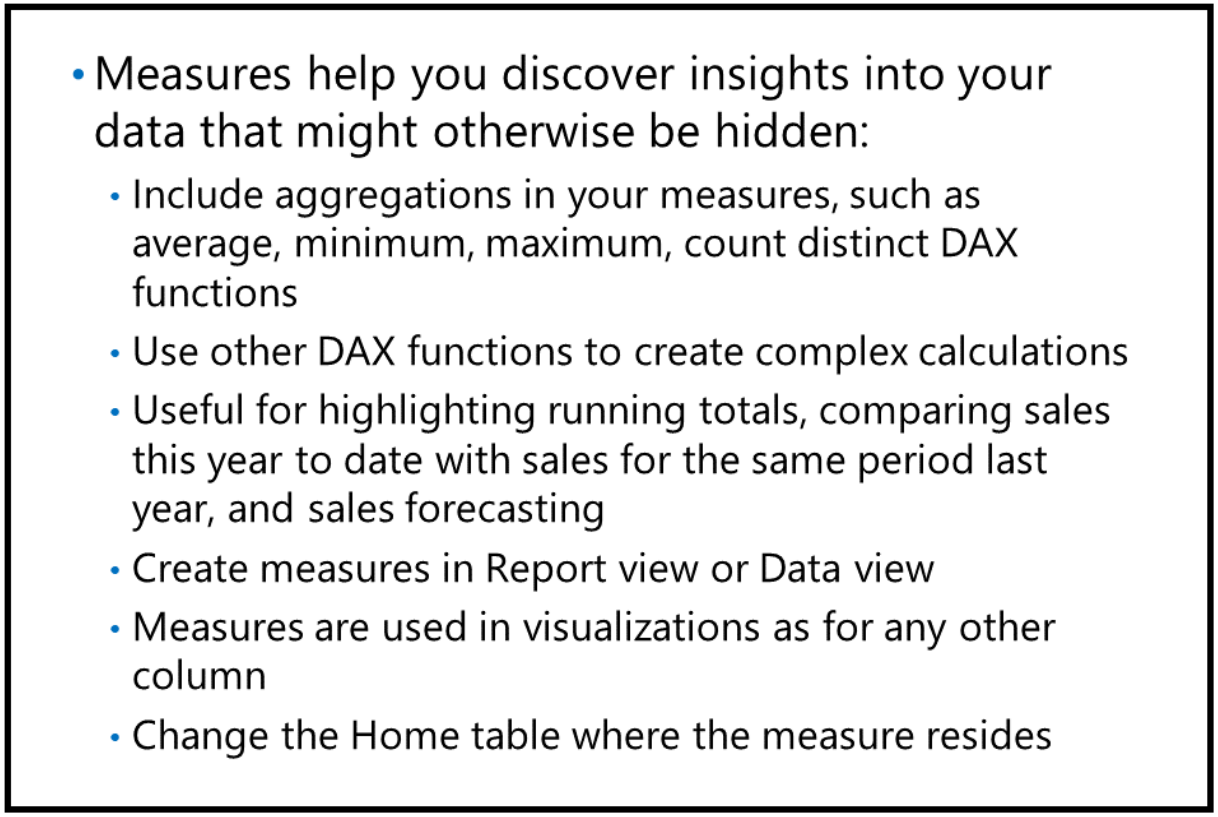
Use the DATATABLE Function to Create a New Table

The following code example creates a new table using the DATATABLE function. Use it to define the column names and data types, and enter values into the table:

```
Countries = DATATABLE
(
   "Country", STRING,
                "Code", STRING,
  {
          {"United States", "US"},
                       {"United Kingdom", "UK"},
                       {"France", "FR"},
                       {"Germany", "DE"},
                       {"Spain", "ES"}
                 }
           )
```

After you create a calculated table, you use it in exactly the same way as any other table that exists in the model, including for in relationships. You give the table and column names any name you like, and format them as you would with a standard table. You then use the columns in your visualizations alongside columns from other tables. You can also add calculated columns and measures to visualizations.

Measures



Power BI measures help you discover insights in your data that might otherwise be hidden. You use measures to answer questions about your data. Some common examples would be using aggregations such as average, minimum, maximum, count distinct, or more complex calculations that use a DAX function. The values in your measures will update and change alongside a data refresh, so your reports always display up-to-date figures.

Measures are created using DAX formulas, and with an extensive library of functions, operators, and constructs, there is scope to create all the measures you require. Measures are useful for creating running totals, or comparing sales for a partial year to sales over the same time the previous year. You can also predict sales by

multiplying current year sales against a target percentage for growth, resulting in an expected sales target.

In Power BI Desktop, you create measures in Report view or Data view, and they appear in the FIELDS pane. To create a new measure in Report view or Data view, right-click the table in the FIELDS pane, and click **New measure**. Alternatively, from the **Modeling** tab, in the **Calculations** group, click **New Measure**. It is generally easier to work in the Data view, because you see the values of the data in the table to which you want to add the measure. The following example creates a measure named **YTD Sales**. Using the TOTALYTD function, the SalesAmount column in the FactInternetSales table is aggregated using SUM, and the dates for the current year.

Create a Measure Calculate Year to Date Sales

The following code creates a measure called YTD Sales. It uses the TOTALYTD function to calculate the year to date sales:

```
YTD Sales = TOTALYTD(SUM(FactInternetSales[SalesAmount]),
DimDate[FullDateAlternateKey])
```

After creating measures, you add them to visualizations in your report, as you would any other column. If you have a visualization showing Last Year's Sales, you could create a new measure to calculate sales for the coming year, based on a predicted growth percentage. The following example creates a measure that multiplies sales for last year by 1.05, or 5 percent:

Create a Measure to Predict Sales for the Coming Year

The following code creates a measure to predict sales for last year based on a 5 percent increase:

```
Sales Forecast = SUM('Sales'[LY Sales]) * 1.05
```

You can change the table in which the measure resides. In the Fields pane, click the measure you want to move, and highlight it. From the **Modeling** tab, in the **Properties** group, click **Home Table**, and select the table.

Demonstration: Creating calculated columns and measures with DAX

In this demonstration, you will see how to:

- Create calculated columns.
- Add a new table.
- Create a new measure.

Check your knowledge

Select the best answer

Which of the following DAX functions is not suitable for creating a calculated table?

UNION

SUM

CROSSJOIN

NATURALINNERJOIN

NATURALLEFTOUTERJOIN

Check answer Show solution Reset

Lab: Modeling data

Scenario

Adventure Works employees are increasingly frustrated by the time it takes to implement managed BI services. The existing managed BI infrastructure, including a data warehouse, enterprise data models, and reports and dashboards, are valued sources of decision-making information. However, users increasingly want to explore relationships with other, currently unmanaged data. It takes too long for the IT department to include these requirements into the corporate BI solution.

As a BI professional, you need to explore ways in which Adventure Works can empower business users to augment their managed enterprise BI solution with self-service BI.

Objectives

After completing this lab, you will be able to:

- View and create relationships in your dataset.
- Add a calculated column to a table.

> **Note:** Because of updates to Microsoft Power BI, the lab steps for this course change frequently. Microsoft Learning regularly updates the lab steps, so they are not available in this manual − but you can access them on GitHub.

Lab setup

Estimated time: 60 minutes

Virtual machine: **20778C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

All the lab steps are contained in 20778C_LAB_05.md.

Exercise 1: Create relationships

Scenario

The data in your organization spreads across a number of sources. To begin with, you will import data extracts from Excel worksheets. The data should be related, so you will examine the relationships that Power BI detects automatically. Because the sales data is an extract, Power BI might not detect all of the relationships, or create them correctly, so you will have to configure them.

The main tasks for this exercise are as follows:

1. Preparing the environment.
2. Automatic relationships.
3. Manual relationships.

Result: At the end of this exercise, you will have a dataset combining data from two Excel worksheets, with relationships between the tables.

Exercise 2: Calculations

Scenario

Having created the required relationships in your dataset, you feel that you might benefit from some additional data that doesn't currently exist. You will add calculated

columns to the tables in your dataset, to fill in the gaps.

The main task for this exercise is as follows:

- Add a calculated column.

Result: At the end of this exercise, you will have calculated columns added to the tables in your dataset.

Review question(s)

Check your knowledge

Discovery

Discuss the functions covered in this topic, or use the link provided in the Functions topic of the DAX Queries lesson to look online at the DAX function reference. How many of these have you already used? Have you used the equivalent functions in Excel? Which functions can you use for creating columns and measures in your organizational datasets?

Show solution Reset

Check your knowledge

Discovery

Look at the dataset you used in the labs. How else can you use DAX formulas to add additional columns or create new measures? Do you think there are any gaps in the data that you could fill using DAX?

Show solution Reset

Module review and takeaways

Microsoft Power BI is making its mark in the self-service BI world—because it can quickly create visually stunning, interactive reports and dashboards. Power BI provides a straightforward way to combine data from a wide range of sources into a single dataset, and then work with that data to create cohesive reports. This module went behind the scenes of the visualizations, and explored the techniques and features on offer to shape and enhance your data. With automatic relationship creation, a vast library of DAX functions, and the ability to add calculated columns, tables, and measures quickly, you have seen how Power BI creates attractive reports, while helping you find hidden insights into your data.