Building calculated columns

Click link to watch a video about calculated columns.

Prerequisites

To use the model shown later in this lesson click link to download **Contoso Sales Sample for Power BI Desktop** file, and unpack it to **C:\PBExams**.

Start **Power BI Desktop** and open **Contoso Sales Sample for Power BI Desktop.**

It is recommended to perform the steps described in this lesson in student's own workstation.

Calculated columns in Power BI using DAX

Click link to watch a video about DAX.

In this lesson, you will learn how to create **calculated columns** in Power BI using **DAX**. The building of calculated columns is a great way of extending the analytical capability of Power BI and by the end of this chapter, you will feel very comfortable with creating new columns through DAX. The writing of calculated columns logically occurs after the data model has been developed.

DAX stands for **Data Analysis Expressions**. DAX is Microsoft's new(ish) language which allows you to return results from data stored using the **xVelocity** database engine, which, unlike for most databases, stores data in columns rather than rows. You can program in DAX within **Power BI** (Microsoft's flagship BI tool), **PowerPivot** (an Excel add-in which allows you to create pivot tables based on multiple tables) and **Analysis Services Tabular Model** (the successor to SSAS Multi-Dimensional, which allows you to share data models and implement security).

Calculated columns are stored in the table in which they are assigned, and the values are static until the data is refreshed.

There are three ways you can use DAX formulas in Power BI:

1. **Calculated Tables** - These calculations will add an additional table to the report based on a formula.
2. **Calculated Columns** - These calculations will add an additional column to a table based on a formula. These columns are treated like any other field in the table.
3. **Measures** - These calculations will add a summary or aggregated measure to a table based on a formula.

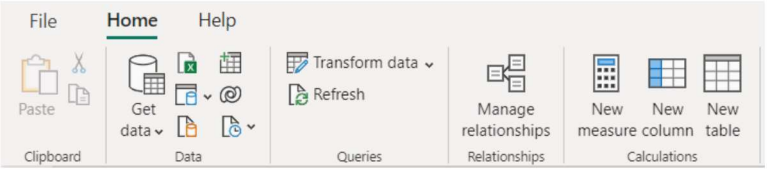
The main difference between these three types of calculations is in their context, and the outputs they produce.

Click link to watch a video about DAC calculation types.

Before you continue you need to first know that Power BI Desktop provides the **IntelliSense** feature. IntelliSense will help you out a lot when writing code, as you will discover very soon. This built-in functionality will autocomplete your code as you go and will also help you explore and discover new functions in the DAX language. In order to take advantage of IntelliSense, you simply need to start typing in the formula bar. Now you are ready to start writing DAX!

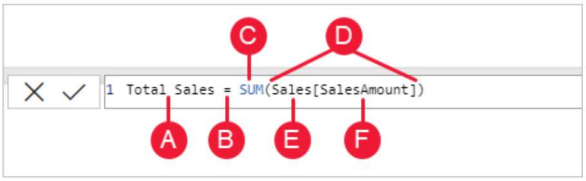
When you create a data model in Power BI Desktop, you can extend a table by creating new columns. The content of the columns is defined by a **DAX** expression evaluated row by row. The DAX formula does not contain the column name and starts with the assignment symbol (**=**).

To add any one of these types of calculations to a model, navigate to the **Modeling** tab of the ribbon. Here you will find three choices for adding either a new **measure**, **calculated column**, or **table**. Alternatively, you can right-click a table in the Fields pane, and you will get the option to add a new measure or calculated column in the drop-down menu.



DAX Syntax

Before you create your own formulas, let's take a 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, here's a simple DAX formula for a measure:



This formula includes the following syntax elements:

A. The measure name, **Total Sales**.

B. The equals sign operator (=), which indicates the beginning of the formula. When calculated, it will return a result.

C. The DAX function **SUM**, which adds up all of the numbers in the **Sales[SalesAmount]** column.

D. Parenthesis **()**, which surround an expression that contains one or more arguments. Most functions require at least one argument. An argument passes a value to a function.

E. The referenced table, **Sales**.

F. The referenced column, **[SalesAmount]**, in the Sales table. With this argument, the SUM function knows on which column to aggregate a SUM.

It's important your formulas have the correct syntax. In most cases, if the syntax isn't correct, a syntax error is returned. In other cases, the syntax might be correct, but the values returned might not be what you're expecting.

A DAX function is an inbuilt function provided in the DAX language to enable you to perform various actions on the data in the tables in your Data Model. DAX functions enable you to perform commonly used data calculations on the Data Model. Some of the DAX functions have same names and functionality as that of Excel functions but have been modified to use DAX data types and to work with tables and columns.

DAX functions can also be nested inside each other to perform multiple operations efficiently. This can save a lot of time when writing DAX formulas. For example, it is often useful to have multiple nested IF statements or to use the IFERROR function to wrap around another function, so that any errors in the formula are represented by the value you specify.

Some of the most common DAX functions used in reports are:

1. Simple calculations: COUNT, DISTINCTCOUNT, SUM, AVERAGE, MIN, MAX.
2. SUMMARISE: Returns a table typically used to further apply aggregations over different groupings.
3. CALCULATE: Performs an aggregation along with one or more filters. When you specify more than one filter, the function will perform the calculation where all filters are true.
4. IF: Based on a logical condition, it will return a different value for if it is true or false. This is similar to the CASE WHEN operation in SQL.
5. IFERROR: Looks for any errors for an inner function and returns a specified result
6. ISBLANK: Checks if the rows in a column are blank and returns true or false. Useful to use in conjunction with other functions like IF.

7. EOMONTH: Returns the last day of the month of a given date (column reference in a date format) for as many months in the past or the future.
8. DATEDIFF: returns the difference between 2 dates (both as column references in date formats) in days, months, quarters, years, etc

Click link to watch a video about DAX functions.

DAX formulas in Power BI are dynamic, and they change according to the context in which they were created. It's important to have an understanding of how contexts work in DAX, as it can help save you a lot of headaches when you run into confusing errors in your formulas.

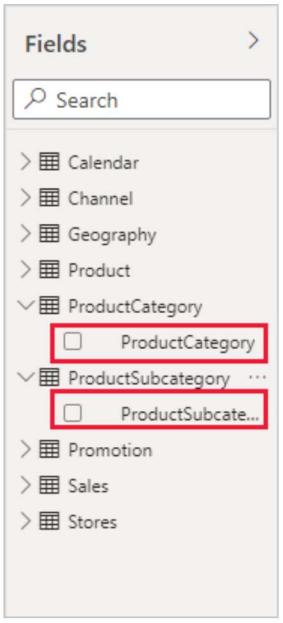
There are two main types of context in DAX: row context and filter context.

- **Row Context** refers to just "the current row" across all columns of a table and also extends to all columns in related tables too. This type of context lets the DAX formula know which rows to use for a specific formula.
- **Filter context** is applied on top of a row context and refers to a subset of rows or columns that are specified as filters in the report. Filters can be applied in a few ways:

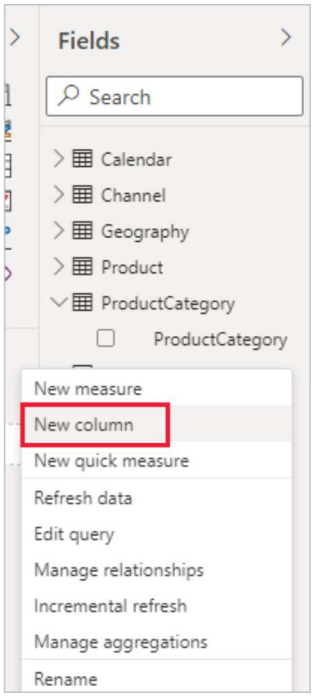
    - Directly in a DAX formula
    - Using the filters pane
    - Using a slicer visual
    - Through the fields that make up a visual (such as the rows and columns in a matrix).

Create a calculated column with values from related tables

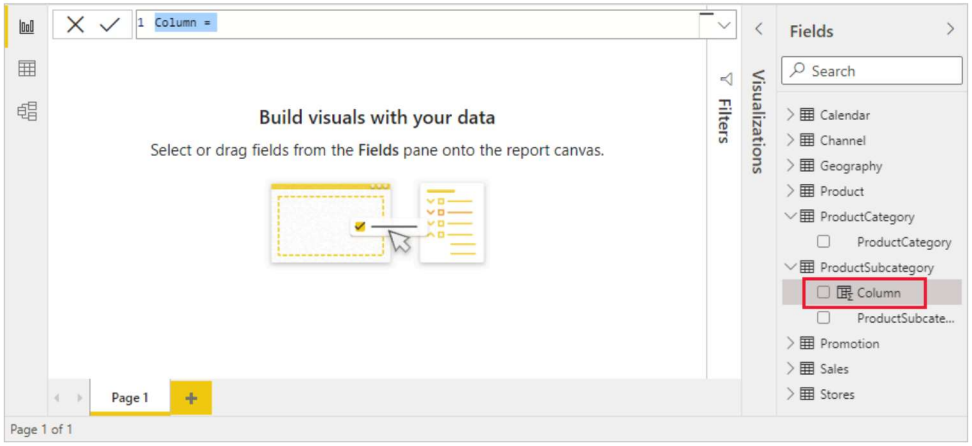
In your Sales Report, you want to display product categories and subcategories as single values, like "Cell phones – Accessories", "Cell phones – Smartphones & PDAs", and so on. There's no field in the **Fields** list that gives you that data, but there's a **ProductCategory** field and a **ProductSubcategory** field, each in its own table. You can create a calculated column that combines values from these two columns. DAX formulas can use the full power of the model you already have, including relationships between different tables that already exist.



1. To create your new column in the **ProductSubcategory** table, right-click or select the ellipsis **...** next to **ProductSubcategory** in the **Fields** pane, and choose **New column** from the menu.



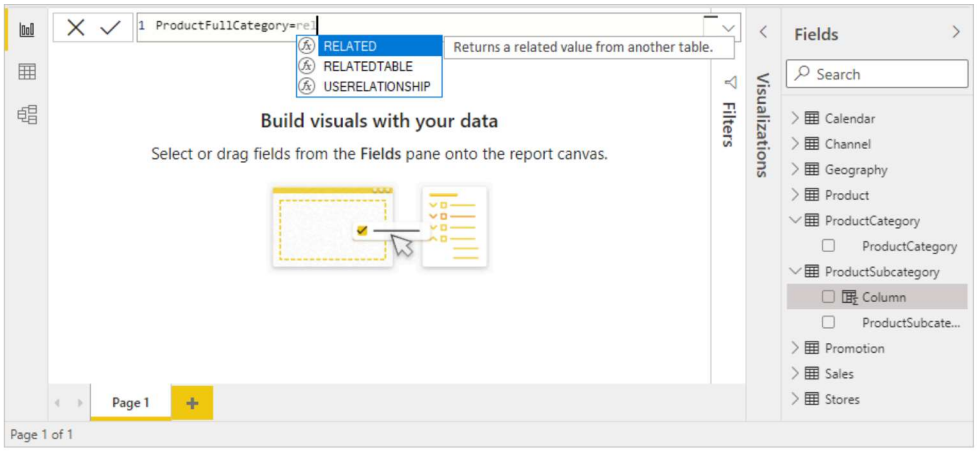
When you choose **New column**, the **Formula bar** appears along the top of the Report canvas, ready for you to name your column and enter a DAX formula.



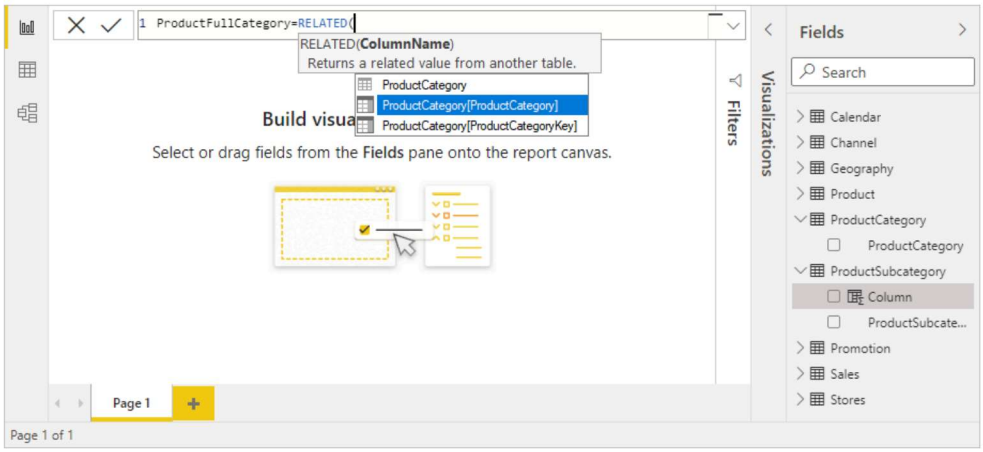
2. By default, a new calculated column is named **Column**. If you don't rename it, new columns will be named **Column 2**, **Column 3**, and so on. You want your column to be more identifiable, so while the **Column** name is already highlighted in the formula bar, rename it by typing **ProductFullCategory**, and then type an equals (**=**) sign.

3. You want the values in your new column to start with the name in the **ProductCategory** field. Because this column is in a different but related table, you can use the RELATED function to help you get it.

   After the equals sign, type **r**. A dropdown suggestion list shows all of the DAX functions beginning with the letter R. Selecting each function shows a description of its effect. As you type, the suggestion list scales closer to the function you need. Select **RELATED**, and then press **Enter**.



An opening parenthesis appears, along with another suggestion list of the related columns you can pass to the RELATED function, with descriptions and details of expected parameters.

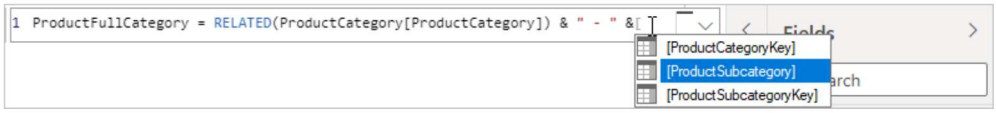


4. You want the **ProductCategory** column from the **ProductCategory** table. Select **ProductCategory[ProductCategory]**, press **Enter**, and then type a closing parenthesis.

5. You want dashes and spaces to separate the **ProductCategories** and **ProductSubcategories** in the new values, so after the closing parenthesis of the first expression, type a space, ampersand (**&**), double-quote (**"**), space, dash (**-**), another space, another double-quote, and another ampersand. Your formula should now look like this:

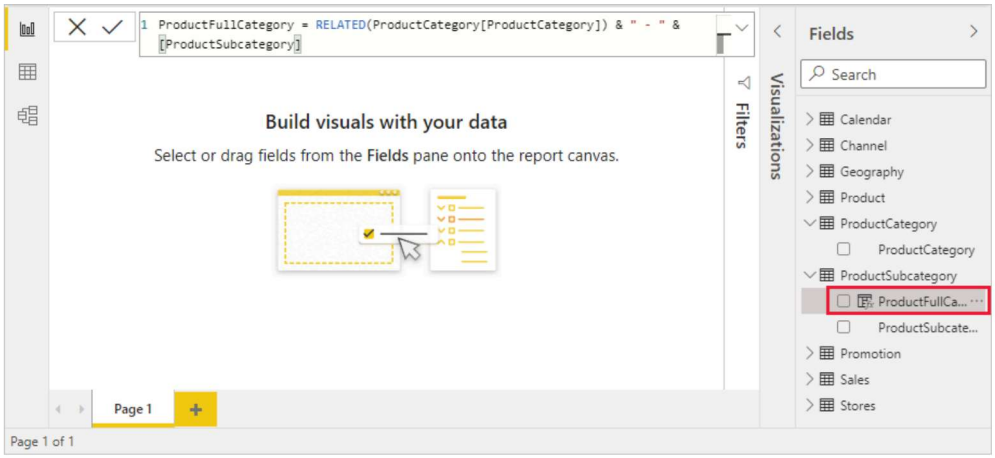
```
ProductFullCategory = RELATED(ProductCategory[ProductCategory]) & " - " &
```

6. Enter an opening bracket (**[**), and then select the **[ProductSubcategory]** column to finish the formula.



You didn't need to use another RELATED function to call the **ProductSubcategory** table in the second expression, because you're creating the calculated column in this table. You can enter **[ProductSubcategory]** with the table name prefix (fully qualified) or without (non-qualified).

7. Complete the formula by pressing **Enter** or selecting the checkmark in the formula bar. The formula validates, and the **ProductFullCategory** column name appears in the **ProductSubcategory** table in the **Fields** pane.



Creating calculated columns is described in detail in Exercise 11.

Creating calculated measures

◄ Lesson 4 Quiz

Siirry...

Exercise 11 - Building calculated columns ►