Dax Expressions

Dax Measure

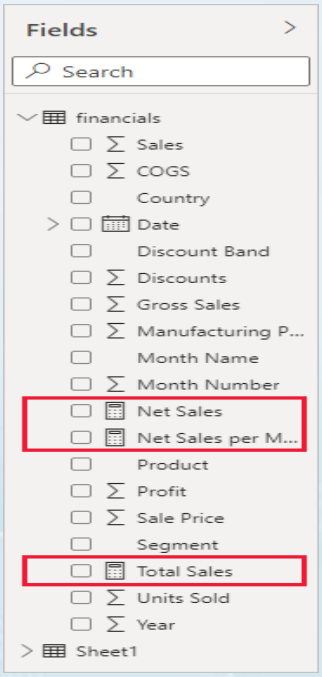
A measure is a formula that is created specifically for use in a PivotTable (or PivotChart) that uses Power Pivot data. Measures can be based on standard aggregation functions, such as COUNT or SUM, or you can define your own formula by using DAX. A measure is used in the **Values** area of a PivotTable. If you want to place calculated results in a different area of a PivotTable, use a calculated column instead.

When you define a formula for an explicit measure, nothing happens until you add the measure into a PivotTable. When you add the measure, the formula is evaluated for each cell in the **Values** area of the PivotTable. Because a result is created for each combination of row and column headers, the result for the measure can be different in each cell.

The definition of the measure that you create is saved with its source data table. It appears in the PivotTable Fields list and is available to all users of the workbook.

Understanding measures

In Power BI Desktop, measures are created and displayed in Report View, Data View, or Model View. Measures you create yourself appear in the **Fields** list with a calculator icon. You can name measures whatever you want, and add them to a new or existing visualization just like any other field.



Data Analysis Expressions

Measures calculate a result from an expression formula. When you create your own measures, you'll use the Data Analysis Expressions (DAX) formula language. DAX includes a library of over 200 functions, operators, and constructs. Its library provides immense flexibility in creating measures to calculate results for just about any data analysis need.

DAX formulas are a lot like Excel formulas. DAX even has many of the same functions as Excel, such like DATE, SUM, and LEFT.

But the DAX functions are meant to work with relational data like we have in Power BI Desktop.

SUM
Let's start with SUM. Most of you know what SUM is, if you've ever used Excel or some other software programs, SUM comes quite naturally. We understand SUM adds up all the values in a column, and because of this the SUM syntax requires a column as input.

SUM(<column>)

In order to see how this function operates, let's create a new measure. Let's say that we want to calculate the value of all our individual items sold. We will call our measure SUM and we'll SUM the column 'Total Sales' from our 'Sales' table. SUM is going to look at the 'Total Sales' column in the 'Sales' table and sum all the values together.

SUM = SUM(Sales[Total Sales])

Let's use a matrix visual to display the measure 'SUM' by 'ITEMNAME'. To do this, we add 'ITEMNAME' as the row value and 'SUM' as our values in Power BI. Here we can see that SUM does exactly what we expect it to do. It will sum the values for each 'ITEMNAME' in the table together and return us a Total at the bottom. This total is the Total value for 'SUM' of all the 'ITEMNAMES'.

ITEMNAME	SUM
13W Mini Fluorescent Bulb	£649 935
2300 Series Posture Chair	£310 420
50W/12V Halogen Bulb	£1 287 780
Answering Machine	£57 505
Bulletin Board	£225 110
Desk Calendar Pad	£1 047 755
Desk Note Book	£523 584
Dry-erase White Board Markers	£442 111
Exec. Credenza 72in X 20in	£50 239
Fluorescent Desk Lamp	£2 143 058
Halogen Desk Light	£1 076 616
Hanging File Folder	£107 505
High Back Arm Tilter	£37 021
Image 1500 Series Desk Accessories	£341 297
Kings 5000 Series Desk Accessories	£30 685
Krugg 220 Arm Tilter-Brown	£12 885
Personal Digital Assistant	£192 092
Total	**£8 535 598**

SUMX

Let's create a new measure now to see how the function SUMX operates. The first notable difference for SUMX is that you need to provide the table name and an expression, so SUMX returns the sum of an expression evaluated for each row in the table. The term <table> contains the rows for which the expression will be evaluated, and the term <expression> evaluate each row of the table

SUMX(<table>, <expression>)

In other words, let's say we want to determine the Total sales for each unit, we will have to choose the 'Sales' table, and, in the 'Sales' table, we have two values, quantity of units sold 'QTYNET' and the unit price 'Unit Price' that must be multiplied to determine the Total sales for each unit.

SUMX = SUMX(Sales, Sales[QTYNET]*Sales[Unit Price])

DATESBETWEEN

Returns a table that contains a column of dates that begins with a specified start date and continues until a specified end date.

This function is suited to pass as a filter to the Calculate function. Use it to filter an expression by a custom date range.

```
DATESBETWEEN(<Dates>, <StartDate>, <EndDate>)
```

Syntax

Example

```
Customers LTD =
CALCULATE(
    DISTINCTCOUNT(Sales[CustomerKey]),
    DATESBETWEEN(
        'Date'[Date],
        BLANK(),
```

ALLSELECTED

Removes context filters from columns and rows in the current query, while retaining all other context filters or explicit filters.

The ALLSELECTED function gets the context that represents all rows and columns in the query, while keeping explicit filters and contexts other than row and column filters. This function can be used to obtain visual totals in queries.

Syntax

```
ALLSELECTED([<tableName> | <columnName>
[, <columnName>[, <columnName>[,...]]] )
```

KEEPFILTERS

Modifies how filters are applied while evaluating a CALCULATE or CALCULATETABLE function.

Syntax

```
KEEPFILTERS(<expression>)
```