What is DAX?

DAX is a collection of functions, operators, and constants that can be used in a formula, or expression, to calculate and return one or more values. Stated more simply, DAX helps you create new information from data already in your model.

Why is DAX so important?

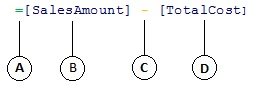
It’s easy to create a workbook and import some data into it. You can even create PivotTables or PivotCharts that display important information without using any DAX formulas. But, what if you need to analyze critical sales data across several product categories and for different date ranges? Or, you need combine important inventory data from several tables in different data sources? DAX formulas provide this capability and many other important capabilities as well. Learning how to create effective DAX formulas will help you get the most out of your data. When you get the information you need, you can begin to solve real business problems that affect your bottom line. This is Business Intelligence, and DAX will help you get there.

Prerequisites

You might already be familiar with creating formulas in Microsoft Excel. That knowledge will be helpful in understanding DAX, but even if you have no experience with Excel formulas, the concepts described here will help you get started creating DAX formulas and solving real-world BI problems right away.

### 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, 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)



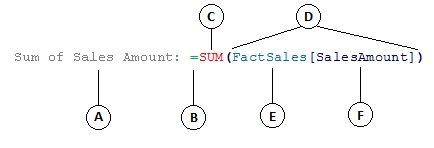
This formula’s syntax includes the following elements:

1. 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.
2. 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.
3. The subtraction (-) mathematical operator.
4. 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.*

Let’s take a look at another type of formula, one that is used in a measure:



This formula includes the following syntax elements:

1. The measure name Sum of Sales Amount. Formulas for measures can include the measure name, followed by a colon, followed by the calculation formula.
2. The equals sign operator (=) indicates the beginning of the calculation formula. When calculated, it will return a result.
3. The function SUM adds up all of the numbers in the [SalesAmount] column. You will learn more about functions later.
4. Parenthesis () surround one or more arguments. All functions require at least one argument. An argument passes a value to a function.
5. The referenced table FactSales.
6. The referenced column [SalesAmount] in the FactSales table. With this argument, the SUM function knows on which column to aggregate a SUM.

You can read this formula as:

*For the measure* *named Sum of Sales Amount, calculate (=) the SUM of values in the [* *SalesAmount* *] column in the FactSales* *table.*

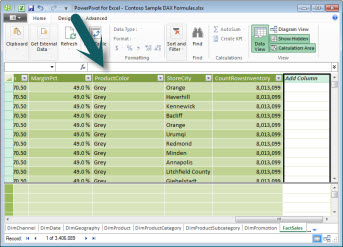
When placed into the Values drop zone in a PivotTable Field List, this measure calculates and returns values defined by each cell in the PivotTable, for example, Cell Phones in the USA.

Notice there are a few things different about this formula compared to the formula we used for the Margin calculated column. In particular, we introduced a function, SUM. Functions are pre-written formulas that make it easier to do complex calculations and manipulations with numbers, dates, time, text, and more. You will learn more about functions later.

Unlike the Margin calculated column earlier, you see the column [SalesAmount] was preceded by the table FactSales in which the column belongs. This is known as a fully qualified column name in that it includes the column name preceded by the table name. Columns referenced in the same table do not require the table name be included in the formula. This can make long formulas that reference many columns shorter and easier to read. However, it is good practice to always include the table name in your measure formulas, even when in the same table.

#### Task: Create a simple formula for a calculated column

1. If you are not already in the Power Pivot window, in Excel, on the Power Pivot ribbon, click Power Pivot**Window**.
2. In the Power Pivot window, click the **FactSales** table (tab).
3. Scroll to the right-most column, and then in the column header, click **Add Column**.
4. Click in the formula bar along the top of the model designer window.



Your cursor now appears in the formula bar. The formula bar is where you can type a formula for a calculated column or a calculated field.

Let’s take a moment to look at the three buttons to the left of the formula bar.

Formula bar

When the cursor is active in the formula bar, those three buttons become active. The leftmost button, the **X**, is simply a cancel button. Go ahead and click it. Your cursor no longer appears in the formula bar, and the cancel button and checkmark button no longer appear. Go ahead and click in the formula bar again. The cancel button and the checkmark button now reappear. This means you are ready to start entering a formula.

The checkmark button is the check formula button. It doesn’t do much until you’ve entered a formula. We’ll come back to it in a little bit.

Click the **Fx** button. You’ll see that a new dialog box appears; the Insert Function dialog box. The Insert Function dialog box is the easiest way to get started entering a DAX formula. We’ll add a function to a formula when we create a measure a little later, but for now, you do not need to add a function to your calculated column formula. Go ahead and close the Insert Function dialog box.

1. In the formula bar, type an equals sign =, then type an opening bracket [. You’ll see a small window appears with all of the columns in the FactSales table. This is IntelliSense in action.

Because calculated columns are always created in the active table you are in, there is no need to precede the column name with the table name. Go ahead and scroll down and then double-click [SalesQuantity]. You can also scroll to the column name you want, and then press Tab.

Your cursor is now active to the right of **[SalesQuantity]**.

1. Type a space, and then type a subtraction operator - (a minus sign), and then type another space.
2. Now, type another opening bracket [. This time, select the **[ReturnQuantity]** column, and then press Enter.

If you get an error, look carefully at your syntax. If necessary, compare it to the formula in the Margin calculated column described earlier.

After you press Enter to complete the formula, the word **Calculating** appears in the status bar along the bottom of the Power Pivot window. It goes quickly, even though you just calculated new values for more than three million rows.

1. Right click the column header and rename the column, NetSales.

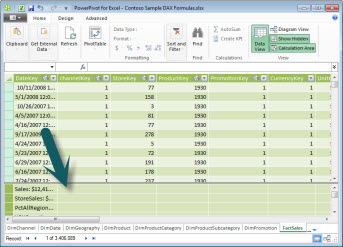
That’s it! You just created a simple yet very powerful DAX formula. For each row in the FactSales table, the NetSales formula calculates a value by subtracting the value in the [ReturnQuantity] column from the value in the [SalesQuantity] column. Notice how we just said “For each row”. This is a glimpse of another very important concept in DAX; row context. You will learn more about row context later.

Something really important to understand when typing an operator into a DAX formula is the data type in the arguments you are using. For example, if you were to type the following formula, = 1 & 2, the value returned would be a text value of “12”. This is because the ampersand (&) operator is for text concatenation. DAX interprets this formula to read: Calculate a result by taking the value 1 as text and add value 2 as text. Now, if you were to type = 1 + 2, DAX reads this formula as: Calculate a result by taking the numeric value 1 and adding the numeric value 2. The result is of course “3”, a numeric value. DAX calculates resultant values depending on the operator in the formula, not based on the data type of columns used in the argument. Data types in DAX are very important, but outside the scope of this Quick Start. To learn more about data types and operators in DAX formulas, see the DAX Reference (http://go.microsoft.com/fwlink/?LinkId=239769&clcid=0x409) in Books Online.

Let’s try another. This time, you will create a measure by typing the formula and by using IntelliSense. Don’t worry too much if you don’t fully understand the formula. The important thing here is to learn how to create a formula using several elements together in correct syntax.

#### Task: Create a measure formula

1. In the **FactSales** table, click in any empty cell in the Calculation Area. This is the area of empty cells just below a table in the Power Pivot window.



1. In the formula bar, type the name Previous Quarter Sales:.
2. Type an equals sign = to begin the calculation formula.
3. Type the first few letters CAL, and then double-click the function you want to use. In this formula, you want to use the **CALCULATE** function.
4. Type an opening parenthesis ( to begin the arguments to be passed to the CALCULATE function.

Notice after typing the opening parenthesis, IntelliSense shows you the arguments required for the CALCULATE function. You will learn about arguments in a little bit.

1. Type the first few letters of the **FactSales** table, and then in the dropdown list, double-click **FactSales[Sales]**.
2. Type a comma (,) to specify the first filter, then type, PRE, and then double-click the **PREVIOUSQUARTER**function.

After selecting the PREVIOUSQUARTER function, another opening parenthesis appears, indicating another argument is required; this time, for the PREVIOUSQUARTER function.

1. Type the first few letters Dim, and then double-click **DimDate[DateKey]**.
2. Close both the argument being passed to the PREVIOUSQUARTER function and the CALCULATE function by typing two closing parentheses )).

Your formula should now look like this:

Previous Quarter Sales:=CALCULATE(FactSales[Sales], PREVIOUSQUARTER(DimDate[DateKey]))

1. Click the check formula button on the formula bar to validate the formula. If you get an error, verify each element of the syntax.

You did it! You just created a measure using DAX, and not an easy one at that. What this formula will do is calculate the total sales for the previous quarter, depending on the filters applied in a PivotTable or PivotChart.

You were just introduced to several important aspects of DAX formulas. First, this formula included two functions. Notice the PREVIOUSQUARTER function is nested as an argument passed to the [CALCULATE](https://support.office.com/en-us/article/calculate-19654bc2-aa88-4f6c-a0b9-6fa7a59c4432)function. DAX formulas can contain up to 64 nested functions. It is unlikely a formula would ever contain so many nested functions. In fact, such a formula would be very difficult to create and debug, and it probably wouldn’t be very fast either.

In this formula, you also used filters. Filters narrow down what will be calculated. In this case, you selected one filter as an argument, which is actually another function. You will learn more about filters later.

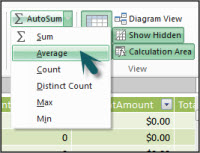
Finally, you used the CALCULATE function. This is one of the most powerful functions in DAX. As you author data models and create more complex formulas, you will likely use this function many times. Discussing the CALCULATE function is outside the scope of this QuickStart, but as your knowledge of DAX grows, pay special attention to this one.

### Extra Credit

You may be asking: ‘What is the simplest DAX formula I can create?’ Well, the answer to that is ‘the formula you don’t have to’. And, that is exactly what you can do by using a standard aggregation function in a measure. Almost any data model needs to filter and calculate on aggregated data. For example, the SUM function in the Sum of Sales Amount measure you saw earlier is used to add up all the numbers in a particular column. DAX includes several other functions that aggregate values as well. You can automatically create formulas using standard aggregations by using the AutoSum feature.

#### Extra Credit Task: Create a measure formula by using the AutoSum feature

1. In the FactSales table, scroll to the ReturnQuantity column, and then click on the column header to select the whole column.
2. On the **Home**tab, on the ribbon, in the **Calculations** group, click the **AutoSum**button.



﻿Click the down arrow next to **AutoSum**, and then click **Average**(notice the other standard aggregation functions you can use, too).

Immediately, a new measure is created with the name Average of ReturnQuantity: followed by the formula =AVERAGE([ReturnQuantity]).

Now wasn’t that easy? Of course, not all formulas you create will be so simple. But, by using the AutoSum feature, you can create quick and easy formulas by using standard aggregation calculations.

This should give you a fairly good understanding of syntax used in DAX formulas. You were also introduced to some really cool features like IntelliSense and AutoSum to help you create quick, easy, and accurate formulas. Of course there is a lot more you can learn about syntax. A good place to learn more is the DAX Reference or SQL Books Online.

#### Syntax QuickQuiz

1. What does this button on the formula bar do?  
   Function button
2. What always surrounds a column name in a DAX formula?
3. How would you write a formula for the following:  
   *In the DimProduct table, for each row in the UnitMargin calculated column, calculate a value by subtracting values in the UnitCost column from values in the UnitPrice column*?

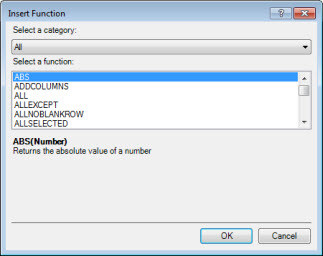
Answers are provided at the end of this topic.

### Functions

Functions are predefined formulas that perform calculations by using specific values, called arguments, in a particular order or structure. Arguments can be other functions, another formula, column references, numbers, text, logical values such as TRUE or FALSE, or constants.

DAX includes the following categories of functions: Date and Time, Information, Logical, Mathematical, Statistical, Text, and Time Intelligence Functions. If you are familiar with functions in Excel formulas, many of the functions in DAX will appear similar to you; however, DAX functions are unique in the following ways:

* 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 need to customize calculations on a row-by-row basis, DAX provides functions that let you use the current row value or a related value as a kind of argument, to perform calculations that vary by context. You will learn more about context later.
* DAX includes many functions that return a table rather than a value. The table is not displayed, but is used to provide input to other functions. For example, you can retrieve a table and then count the distinct values in it, or calculate dynamic sums across filtered tables or columns.
* DAX includes a variety of time intelligence functions. These functions let you define or select date ranges, and perform dynamic calculations based on them. For example, you can compare sums across parallel periods.

Sometimes it is difficult to know which functions you might need to use in a formula. Power Pivot, and the tabular model designer in SQL Server Data Tools, include the Insert Function feature, a dialog box that helps you select functions by category and provides short descriptions for each function.  
  


Let’s create a new formula which includes a function you will select by using the Insert Function feature:

#### Task: Add a function to a formula by using Insert Function

1. In the FactSales table, scroll to the right-most column, and then in the column header, click **Add Column**.
2. In the formula bar, type an equals sign, =.
3. Click the **Insert Function** button. Insert Function This opens the **Insert Function** dialog box.
4. In the **Insert Function** dialog box, click the **Select a category** list box. By default, **All**is selected, and all of the functions in the **All**category are listed below. That’s a lot of functions, so you will want to filter the functions to make it easier to locate the type of function you are looking for.
5. For this formula, you want to return some data that already exists in another table. For that, you are going to use a function in the Filter category. Go ahead and click the **Filter**category, and then in **Select a function**, scroll down and double-click the RELATED function. Click **Ok**to close the **Insert Function**dialog box.
6. Use IntelliSense to help you find and select the DimChannel[ChannelName] column.
7. Close the formula and then press Enter.
8. After you press Enter to complete the formula, the word Calculating appears in the status bar along the bottom of the Power Pivot window. Now you will see that you just created a new column in the FactSales table with channel information from the DimChannel table.
9. Rename the column Channel.

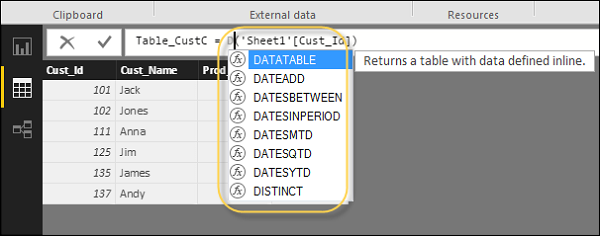
Your formula should look like this: =RELATED(DimChannel[ChannelName])

## DAX Functions

In Power BI, you can use different function types to analyze data, and create new columns and measures. It includes functions from different categories such as −

* Aggregate
* Text
* Date
* Logical
* Counting
* Information

Power BI provides an easy way to see the list of all functions. When you start typing your function in the formula bar, you can see the list of all functions starting with that alphabet.



### Aggregate Functions

DAX has a number of aggregate functions.

* MIN
* MAX
* Average
* SUM
* SUMX

### Counting Functions

Other counting functions in DAX include −

* DISTINCTCOUNT
* COUNT
* COUNTA
* COUNTROWS
* COUNTBLANK

### Logical Functions

Following are the collection of Logical functions −

* AND
* OR
* NOT
* IF
* IFERROR

### TEXT Functions

* REPLACE
* SEARCH
* UPPER
* FIXED
* CONCATENATE

### DATE Functions

* DATE
* HOUR
* WEEKDAY
* NOW
* EOMONTH

### INFORMATION Functions

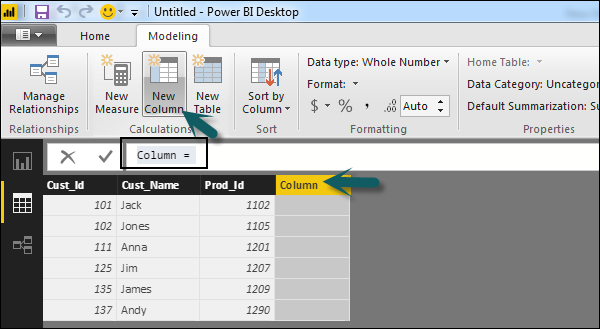
* ISBLANK
* ISNUMBER
* ISTEXT
* ISNONTEXT
* ISERROR

## DAX Calculation Types

In Power BI, you can create two primary calculations using DAX −

* Calculated columns
* Calculated measures

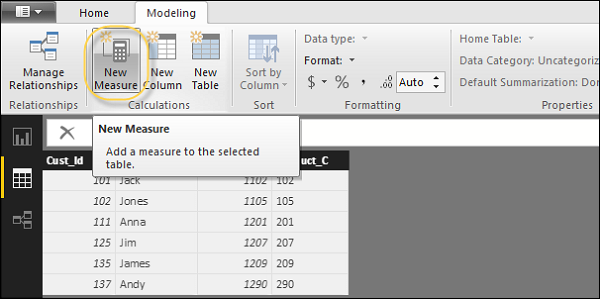
When you navigate to the Modeling tab, you can see a New Column option at the top of the screen. This also opens the formula bar where you can enter DAX formula to perform the calculation. DAX - Data Analysis Expression is a powerful language used in Excel to perform calculations. You can also rename the column by changing the Column text in the formula bar.



In the following example, we have created a new column: Product Code (Product\_C), which is derived from the last 3 characters of Prod\_Id column. Following is the formula −

Product\_C = RIGHT( Sheet1[Prod\_Id],3)

To create a calculated measure, navigate to New Measure tab under Modeling. This will add a new object under the Fields tab with the name Measure.

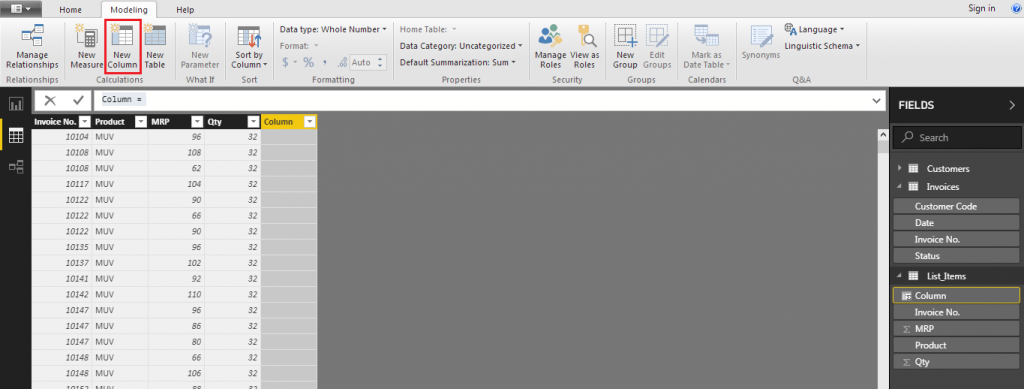




You can write DAX formula to calculate the value of the new measure, as we did for the new calculated column.

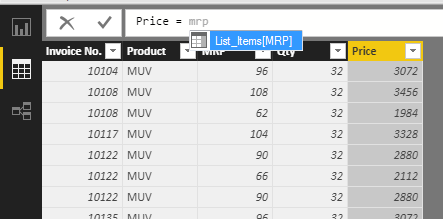
### Calculated Columns

As the name may suggest, you can create new columns based on the data as needed. For example, there is no ‘net price’ column available in the *List\_Items*table. Only MRP & quantity are available. Let’s use DAX to calculate a new column. This will demonstrate how simple it is to get started with DAX. To do this, in the data view, click on the List\_Items table. Then to add a new calculated column, go to the Modeling Tab and select New Column. A new blank column will be created and your cursor will be in the DAX editor (similar to the formula bar in Excel).



New Calculated Column

Here, whatever is before the = will become the name of the column (you can rename the column later on as well). The calculation or, more appropriately, expression will go after the equal sign. Let’s name the column ‘Price’. After the equal sign, type mrp and you should see an automatic dropdown showing the table and column name like this:



IntelliSense in Power BI

Pressing tab will type it out for you! Next, multiply by the quantity column and press enter. Once done, your ‘formula’ or expression should look like this:

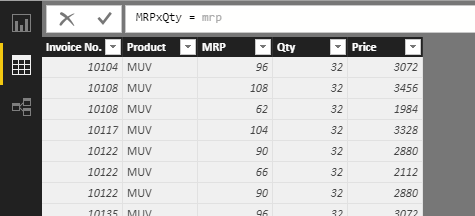
Price = List\_Items[MRP]\*List\_Items[Qty]

In the data, each row will now have the respective calculated price.

Calculated columns are easy to understand and use, but take up more storage space on disk, especially if there are more than a few billion rows of data. This is because all the rows will have to be calculated every time the data is refreshed.

### Measures

Measures allow you to perform a calculation, without actually adding to the data. This is very helpful for reports; where the price can be shown, without needing a whole new column to store it in. Just one major difference is that measures have to be told exactly what to calculate on. If you type in a measure to calculate MRP \* Qty; you would **not** get a dropdown with the column names:



Measures without Aggregators – No IntelliSense

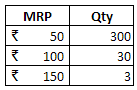
This is indicating that such a calculation is not accepted at all. This is because in order to calculate, we need to first specify what needs to be calculated –

1] Total of the MRP column **\*** Total of Qty column

OR

2] the MRP value in each row **\*** the corresponding Qty value in each row

This difference will be easier to grasp if you try to do both these calculations on this:



The answer to the first calculation will be ₹99,900 and the answer to the second will be ₹18,450!

Calculated columns just assumed and automatically used the second calculation. Using measures, you can calculate for each row by using:

MRPxQty = SUM**X**(List\_Items,List\_Items[MRP]\*List\_Items[Qty])

The ‘X’ after the sum means this calculation will be carried out for each row. This difference is officially called the Row Context.

### Tables

DAX functions in tabular model simply return entire tables, not just a value or a column of values. For example, to get a list of all the cities the company has customers in, use the function:

CitiesTouched = DISTINCT(Customers[City])

This will create table called ‘CitiesTouched’ and list out all the distinct or unique values in the City column in the Customers Table. This table can then be used independently as just like any other table. It will also be visible with the other tables in the tables section of Power BI.

### A word on Filters

Filters essentially “hide” any rows that do not fit a given criteria. An calculations after filtering out data will be applied only on rows that match the criteria. This is similar to what happens in Excel with the subtotal or aggregate functions. By default, they ignore any data that is filtered out. In DAX, most functions by default ignore rows that are filtered out. This is officially called the Filter Context. There are a few exceptions that can operate outside of the filtered data (see the *calculate* & *all*functions below).

## Power BI DAX Functions

Some related function in DAX are as follows. Since a lot of them are available in Excel, they function just like you would expect them to. Will elaborate more where necessary. Like in Excel, the required syntax / arguments / format function in DAX will be displayed in the application itself.

### Average

Find the average from a given set of values.  
**Example**– find out the average sales amount across all orders.  
AvgSales = AVERAGE(List\_Items[Price])

### Max

Find the maximum from a given set of values.  
**Example** – Find out the highest order.  
HighestSale = MAX(List\_Items[Price])

### Min

Find the minimum from a given set of values.  
**Example** – Find out the lowest order.  
LowestSale = MIN(List\_Items[Price])

### Count

Count any numerical data.  
**Example** – Count number of invoices generated.  
InvoiceVolume = COUNT(Invoices[Invoice No.])

### CountRows

Count the number of rows.  
**Example** – Count the number of rows in the ‘CitiesTouched’ table.  
RowsOfCitiesTouched = COUNTROWS(CitiesTouched)

### CountA

Count any kind of data; except blanks.  
Example – Count the customer codes to get the number customers the company has catered to.  
CustCount = COUNTA(Customers[Customer Code])

### Concatenate

Is used to join values in calculated columns. Use ConcatenateX if using in measures.  
**Example** – Concatenate the Product names and MRPs will give a unique code for all the price points at which each product is sold. This is helpful because the price of the product determines variant in our specific data model.  
ProMrp = CONCATENATE(List\_Items[Product],List\_Items[MRP])

### TotalYTD

Calculates the sum from the start of current Year To a given Date. It uses calendar year, not financial year. Calculates for each row, i.e. will return single values.  
**Example** – Calculate running / cumulative totals for the price column.  
CumiSales = TOTALYTD(SUM(List\_Items[Price]),Invoices[Date])

### Distinct

Returns unique values as a table. Using this in a calculated column will not work as it returns a whole table.  
**Example** – As used above, in the tables section.

### Filter

The DAX filter function returns a Table based on a criteria. Can be used to create a Sub- Table. Using this in a calculated column will not work as it returns a whole table.  
**Example** – Create a new table showing customers only from the USA.  
CustUSA = FILTER(Customers,Customers[Country]="USA")

### Calculate

The Calculate function in DAX is used to circumvent all existing filters applied to any table and calculate. It can also add new filters before calculating.  
**Example** – Create a new table showing total sales for each product. Filter the data to show only sales from a single country (marked red). Add a new column to show percentage of that country’s sales vis a vis total sales.  
PercentOFGlobal = SUM(List\_Items[Price]) / CALCULATE(SUM(List\_Items[Price]),ALL(List\_Items[Product]),ALL(Customers[Country]))

### All

Returns everything. Ignores filters.  
**Example** – Used with the calculate function above**.**