DAX

Intro to DAX

DAX (Data Analysis eXperssion Language) is robust development language. DAX is used to get the most out of your Tabular and Power BI Desktop models

You can learn the basics of DAX fairly easily. Once you get experienced in DAX, you may grow to appreciate DAX's flexibility in solving complex analytic and reporting challenges.

Why is DAX so important?

It’s easy to create a new Power BI Desktop file and import some data into it. You can even create reports that show valuable insights without using any DAX formulas at all. But what if you need to:

- Analyze growth percentage across product categories and for different date ranges?
- You need to calculate year-over-year growth compared to market trends?

DAX formulas provide this capability. You can use DAX to perform any kind of calculations and 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.

DATA TYPES in DAX

- You can import data into a model from many different data sources that might support different data types.
- When you import data into a model, the data is converted to one of the tabular model data types.
- When the model data is used in a calculation, the data is then converted to a DAX data type for the duration and output of the calculation.
- When you create a DAX formula, the terms used in the formula will automatically determine the value data type returned.

Data type in model	Data type in DAX	Description
Whole Number	A 64 bit (eight-bytes) integer value [1, 2]	Numbers that have no decimal places. Integers can be positive or negative numbers, but must be whole numbers between -9,223,372,036,854,775,808 (-2^63) and 9,223,372,036,854,775,807 (2^63-1).
Decimal Number	A 64 bit (eight-bytes) real number [1, 2]	Real numbers are numbers that can have decimal places. Real numbers cover a wide range of values: Negative values from -1.79E +308 through -2.23E -308 Zero Positive values from 2.23E -308 through 1.79E + 308 However, the number of significant digits is limited to 17 decimal digits.

Boolean	Boolean	Either a True or False value.
Text	String	A Unicode character data string. Can be strings, numbers or dates represented in a text format.
Date	Date/time	Dates and times in an accepted date-time representation. Valid dates are all dates after March 1, 1900.
Currency	Currency	Currency data type allows values between -922,337,203,685,477.5808 to 922,337,203,685,477.5807 with four decimal digits of fixed precision.
N/A	Blank	A blank is a data type in DAX that represents and replaces SQL nulls. You can create a blank by using the BLANK function, and test for blanks by using the logical function, ISBLANK.

Operators in DAX

The Data Analysis Expression (DAX) language uses operators to create expressions that compare values, perform arithmetic calculations, or work with strings.

Types of operators:

There are four different types of calculation operators:

- Arithmetic
- Comparison
- Text concatenation
- Logical

Arithmetic operators

To perform basic mathematical operations such as addition, subtraction, or multiplication; combine numbers.

ARITHMETIC OPERATOR	MEANING	EXAMPLE
+ (plus sign)	Addition	3+3
– (minus sign)	Subtraction or sign	3–1–1
* (asterisk)	Multiplication	3*3
/ (forward slash)	Division	3/3
^ (caret)	Exponentiation	16^4

Comparison operators

You can compare two values with the following operators. When two values are compared by using these operators, the result is a logical value, either TRUE or FALSE.

Comparison operator	Meaning	Example
=	Equal to	[Region] = "USA"
==	Strict equal to	[Region] == "USA"
>	Greater than	[Sales Date] > "Jan 2009"
<	Less than	[Sales Date] < "Jan 1 2009"
>=	Greater than or equal to	[Amount] >= 20000
<=	Less than or equal to	[Amount] <= 100
<>	Not equal to	[Region] <> "USA"

Text concatenation operator

Use the ampersand **(&)** to join, or concatenate, two or more text strings to produce a single piece of text.

Text operator	Meaning	Example
& (ampersand)	Connects, or concatenates, two values to produce one continuous text value	[Region] & ", " & [City]

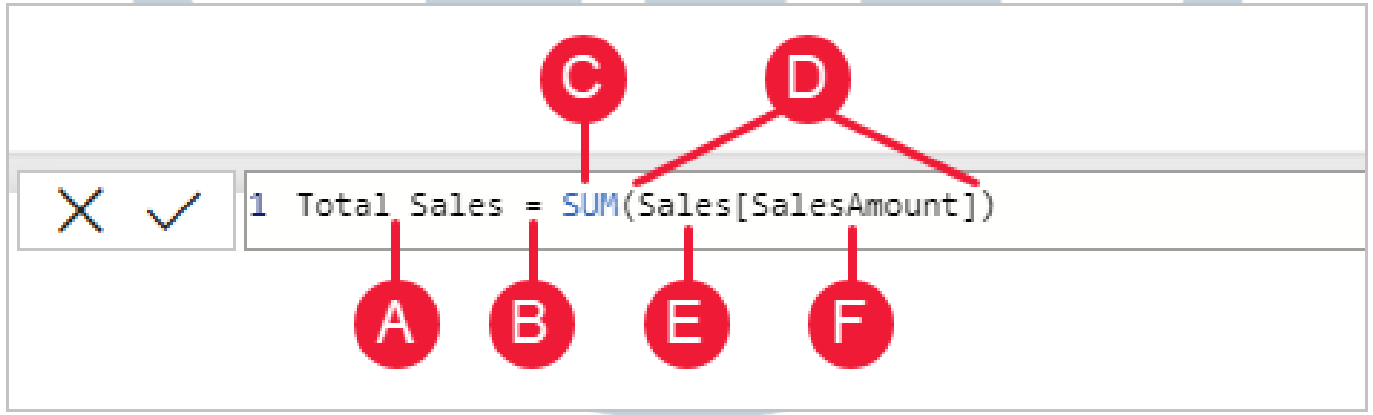
Logical operators

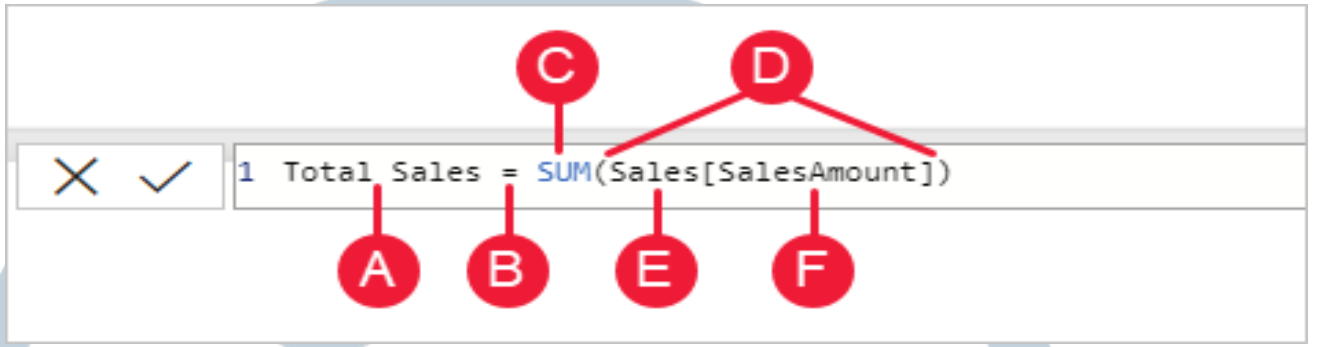
Use logical operators (&&) and (||) to combine expressions to produce a single result.

Text operator	Meaning	Examples
&& (double ampersand)	Creates an AND condition between two expressions that each have a Boolean result. If both expressions return TRUE, the combination of the expressions also returns TRUE; otherwise the combination returns FALSE.	([Region] = "France") && ([BikeBuyer] = "yes"))
|| (double pipe symbol)	Creates an OR condition between two logical expressions. If either expression returns TRUE, the result is TRUE; only when both expressions are FALSE is the result FALSE.	(([Region] = "France") || ([BikeBuyer] = "yes"))
IN	Creates a logical OR condition between each row being compared to a table. Note: the table constructor syntax uses curly braces.	'Product'[Color] IN { "Red", "Blue", "Black" }

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:





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. You'll learn more about functions later.

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.

What are Formulas, Functions and Arguments

Formula: A formula is basically a mathematical expression which follows BODMAS Rule.

Function: A function is a predefined formula that performs calculations using specific values in a particular order.

Arguments: Arguments are the values that functions use to perform calculations. There are two types of arguments, one is required and other is optional

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**

Aggregate Functions

Aggregation functions calculate a (scalar) value such as sum, average, minimum, or maximum for all rows in a column or table as defined by the expression.

- MIN

```
MIN(<column>)
```

Returns the smallest value in a column, or between two scalar expressions.

- MAX

```
MAX(<column>)
```

Returns the largest value in a column, or between two scalar expressions.

- AVERAGE

```
AVERAGE(<column>)
```

Returns the average (arithmetic mean) of all the numbers in a column.

- SUM

```
SUM(<column>)
```

Adds all the numbers in a column.

- SUMX

```
SUMX(<table>, <expression>)
```

Returns the sum of an expression evaluated for each row in a table.

Counting Functions

Counting functions in DAX include –

- DISTINCTCOUNT

```
DISTINCTCOUNT(<column>)
```

Counts the number of distinct values in a column.

- COUNT

```
COUNT(<column>)
```

Counts the number of cells in a column that contain non-blank values.

- COUNTA

```
COUNTA(<column>)
```

Counts the number of cells in a column that are not empty.

- COUNTROWS

```
COUNTROWS([<table>])
```

counts the number of rows in the specified table, or in a table defined by an expression.

- COUNTBLANK

```
COUNTBLANK(<column>)
```

Counts the number of blank cells in a column.

Logical Functions

Following are the collection of Logical functions –

- AND

```
AND(<logical1>,<logical2>)
```

Checks whether both arguments are TRUE and returns TRUE if both arguments are TRUE. Otherwise returns false.

- OR

```
OR(<logical1>,<logical2>)
```

Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.

- NOT

```
NOT(<logical>)
```

Changes FALSE to TRUE, or TRUE to FALSE.

- IF

```
IF(<logical_test>, <value_if_true>[, <value_if_false>])
```

Checks a condition, and returns one value when it's TRUE, otherwise it returns a second value.

- IFERROR

```
IFERROR(value, value_if_error)
```

Evaluates an expression and returns a specified value if the expression returns an error; otherwise returns the value of the expression itself.

TEXT Functions

- REPLACE

```
REPLACE(<old_text>, <start_num>, <num_chars>, <new_text>)
```

REPLACE replaces part of a text string, based on the number of characters you specify, with a different text string.

- SEARCH

```
SEARCH(<find_text>, <within_text>[, [<start_num>][, <NotFoundValue>]])
```

Returns the number of the character at which a specific character or text string is first found, reading left to right. Search is case-insensitive and accent sensitive.

- UPPER

```
UPPER (<text>)
```

Converts a text string to all uppercase letters.

- FIXED

```
FIXED(<number>, <decimals>, <no_commas>)
```

Rounds a number to the specified number of decimals and returns the result as text. You can specify that the result be returned with or without commas.

- CONCATENATE

```
CONCATENATE(<text1>, <text2>)
```

Joins two text strings into one text string.

DATE Functions

These functions are used to carry out calculations on the date and time values

Comparing Data over Time

Year-to-Date

Prior Year

Year Over Year

Requirements

Date Table

One Row for Every Date (No gaps)

Span Range of Possible Dates

Some important DATE functions

- DATE

```
DATE(<year>, <month>, <day>)
```

Returns the specified date in **datetime** format.

- WEEKDAY

```
WEEKDAY(<date>, <return_type>)
```

Returns a number from 1 to 7 identifying the day of the week of a date. By default the day ranges from 1 (Sunday) to 7 (Saturday).

- DATESBETWEEN

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

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

- TOTALYTD

```
TOTALYTD(<expression>,<dates>[,<filter>][,<year_end_date>])
```

Evaluates the year-to-date value of the **expression** in the current context.

- SAMEPERIODLASTYEA

```
SAMEPERIODLASTYEAR(<dates>)
```

Returns a table that contains a column of dates shifted one year back in time from the dates in the specified **dates** column, in the current context.

- DATESYTD

```
DATESYTD(<dates> [,<year_end_date>])
```

Returns a table that contains a column of the dates for the year to date, in the current context.

- DATEADD

```
= DATEADD(DateTime[DateKey],-1,year)
```

Returns a table that contains a column of dates, shifted either forward or backward in time by the specified number of intervals from the dates in the current context.

- CALENDAR

```
= CALENDAR (DATE (2015, 1, 1), DATE (2021, 12, 31))
```

Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.

- DATEADD

```
DATEADD(<dates>,<number_of_intervals>,<interval>)
```

Returns a table that contains a column of dates, shifted either forward or backward in time by the specified number of intervals from the dates in the current context.

- CALENDAR

```
CALENDAR(<start_date>, <end_date>)
```

Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.

INFORMATION Functions

- ISBLANK

```
ISBLANK(<value>)
```

Checks whether a value is blank and returns TRUE or FALSE.

- ISNUMBER

```
ISNUMBER(<value>)
```

Checks whether a value is a number and returns TRUE or FALSE.

- ISTEXT

```
ISTEXT(<value>)
```

Checks if a value is text and returns TRUE or FALSE.

- ISNONTEXT

```
ISNONTEXT(<value>)
```

Checks if a value is not text (blank cells are not text), and returns TRUE or FALSE.

- ISERROR

```
ISERROR(<value>)
```

Checks whether a value is an error, and returns TRUE or FALSE.

Other Important Functions

- CALCULATE

```
CALCULATE(<expression>[, <filter1> [, <filter2> [, …]]])
```

Evaluates an expression in a modified filter context.

- Related

```
RELATED(<column>)
```

Returns a related value from another table.

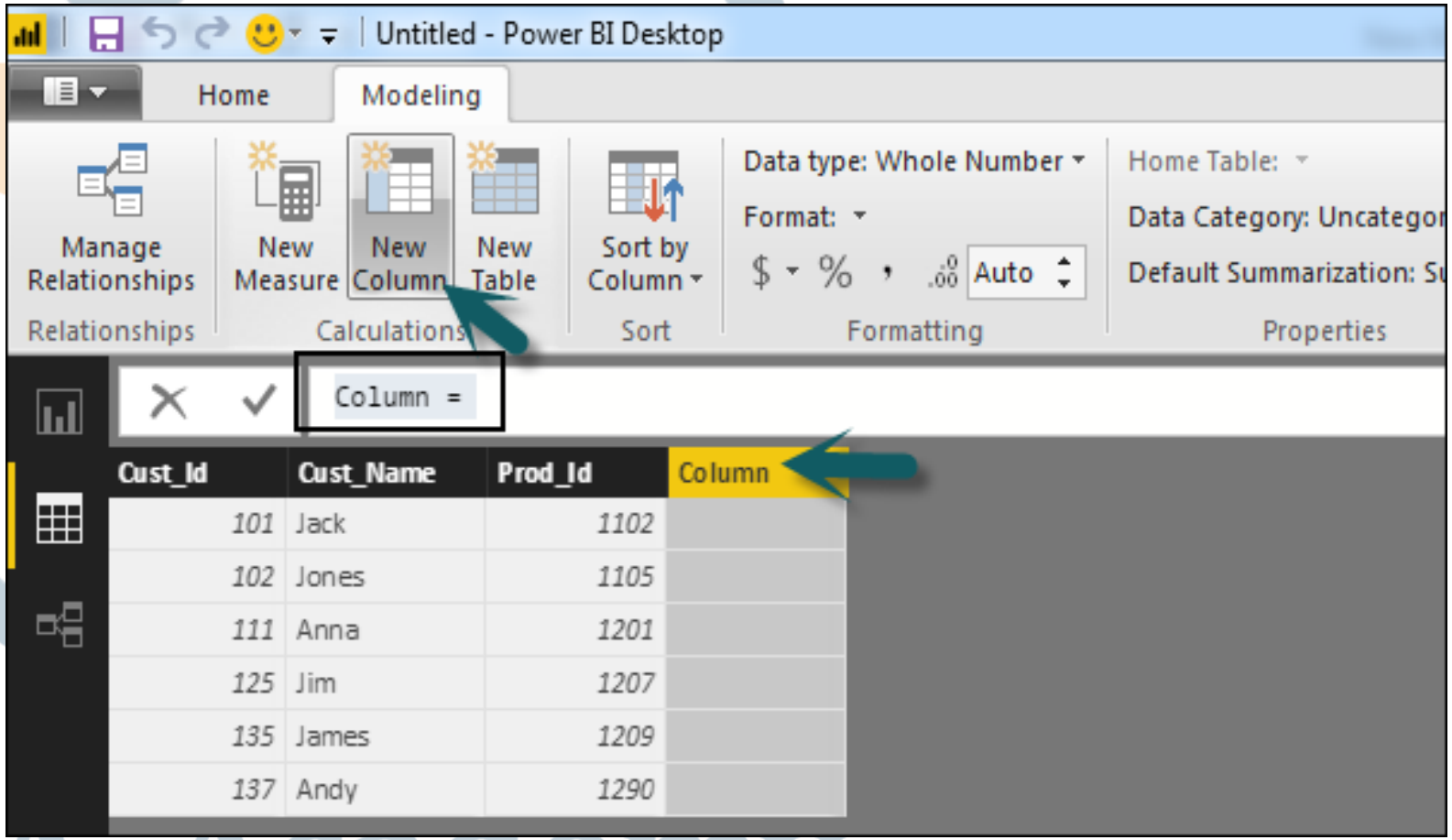
DAX Calculation Types

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

- Calculated columns
- Calculated measures
- Calculated Table

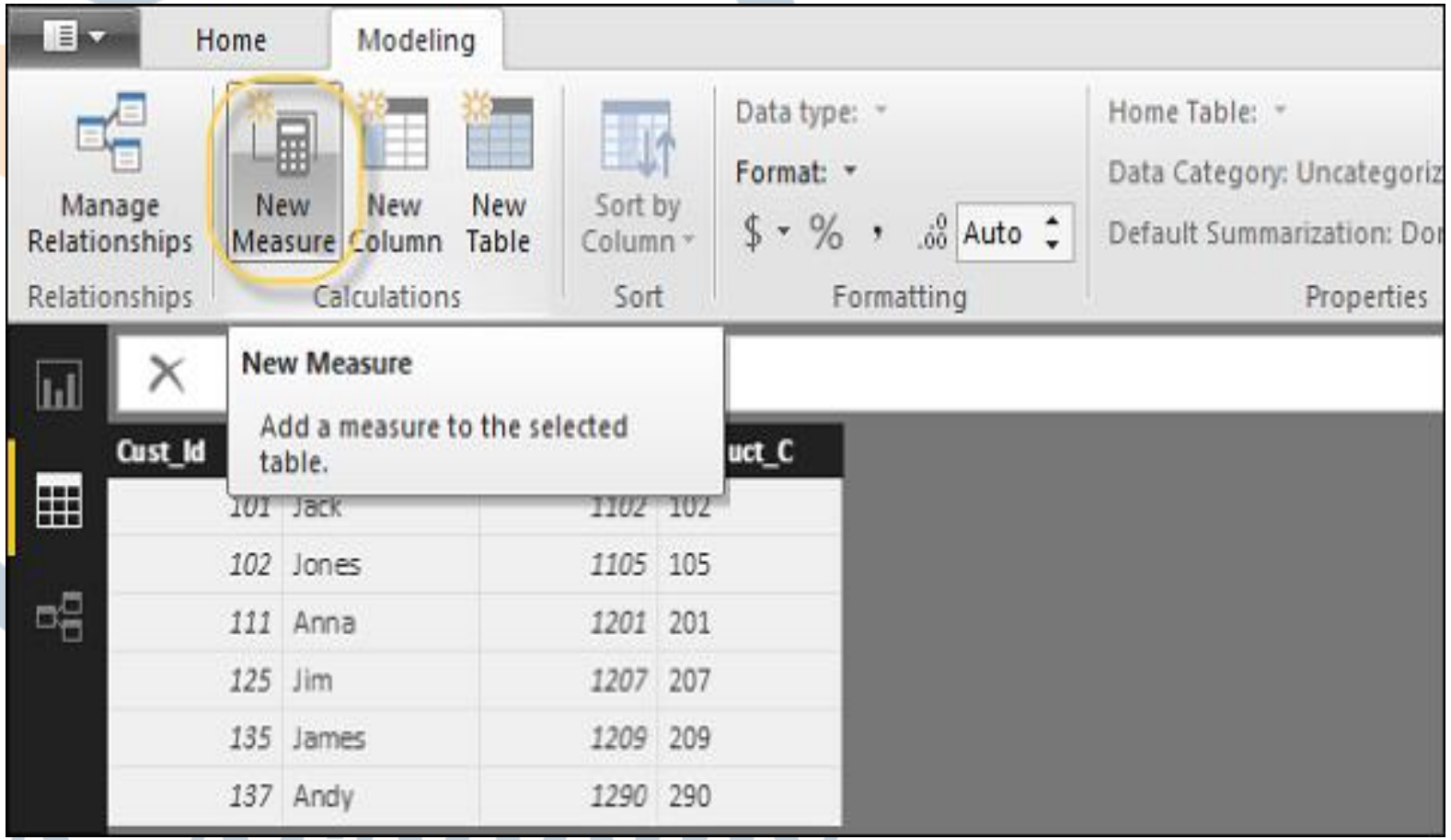
Calculated columns

- Modeling tab -> New Column
- This open the formula bar where you can enter DAX formula to perform the calculation.
- You can also rename the column by changing the Column text in the formula bar.



Calculated measures

- 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.



Calculated Tables

- Most of the time, you create tables by importing data into your model from an external data source.
- But *calculated tables* let you add new tables based on data you've already loaded into the model.
- Instead of querying and loading values into your new table's columns from a data source, you create a DAX formula to define the table's values.
- Just like other Power BI Desktop tables, calculated tables can have relationships with other tables.