

DAX SYNTAX

MEASURE NAME

- Note:** Measures are always surrounded in brackets (i.e. **[Total Quantity]**) when referenced in formulas, so spaces are OK

Total Quantity: =SUM(Transactions[quantity])

Referenced
TABLE NAME

Referenced
COLUMN NAME

FUNCTION NAME

- Calculated columns don't always use functions, but measures do:
 - In a **Calculated Column**, =Transactions[quantity] returns the value from the quantity column in each row (since it evaluates one row at a time)
 - In a **Measure**, =Transactions[quantity] will return an **error** since Power BI doesn't know how to translate that as a single value (you need some sort of aggregation)

Note: This is a "fully qualified" column, since it's preceded by the table name -- table names with spaces must be surrounded by **single quotes**:

- Without a space: Transactions[quantity]
- With a space: 'Transactions Table'[quantity]



PRO TIP:

For **column** references, use the fully qualified name (i.e. **Table[Column]**)
For **measure** references, just use the measure name (i.e. **[Measure]**)

*Copyright 2018, Excel Maven & Maven Analytics, LLC

DAX OPERATORS

Arithmetic Operator	Meaning	Example
+	Addition	2 + 7
-	Subtraction	5 - 3
*	Multiplication	2 * 6
/	Division	4 / 2
^	Exponent	2 ^ 5

Comparison Operator	Meaning	Example
=	Equal to	[City]="Boston"
>	Greater than	[Quantity]>10
<	Less than	[Quantity]<10
>=	Greater than or equal to	[Unit_Price]>=2.5
<=	Less than or equal to	[Unit_Price]<=2.5
<>	Not equal to	[Country]<>"Mexico"

Pay attention to these!

Text/Logical Operator	Meaning	Example
&	Concatenates two values to produce one text string	[City] & " " & [State]
&&	Create an AND condition between two logical expressions	(([State]="MA") && ([Quantity]>10)
(double pipe)	Create an OR condition between two logical expressions	(([State]="MA") ([State]="CT")
IN	Creates a logical OR condition based on a given list (using curly brackets)	'Store Lookup'[State] IN { "MA", "CT", "NY" }

*Head to www.msdn.microsoft.com for more information about DAX syntax, operators, troubleshooting, etc

*Copyright 2018, Excel Maven & Maven Analytics, LLC

COMMON FUNCTION CATEGORIES

MATH & STATS Functions

Basic **aggregation** functions as well as “**iterators**” evaluated at the row-level

Common Examples:

- SUM
- AVERAGE
- MAX/MIN
- DIVIDE
- COUNT/COUNTA
- COUNTROWS
- DISTINCTCOUNT

Iterator Functions:

- SUMX
- AVERAGEX
- MAXX/MINX
- RANKX
- COUNTX

LOGICAL Functions

Functions for returning information about values in a given **conditional expression**

Common Examples:

- IF
- IFERROR
- AND
- OR
- NOT
- SWITCH
- TRUE
- FALSE

TEXT Functions

Functions to manipulate **text strings** or **control formats** for dates, times or numbers

Common Examples:

- CONCATENATE
- FORMAT
- LEFT/MID/RIGHT
- UPPER/LOWER
- PROPER
- LEN
- SEARCH/FIND
- REPLACE
- REPT
- SUBSTITUTE
- TRIM
- UNICHAR

FILTER Functions

Lookup functions based on related tables and **filtering** functions for dynamic calculations

Common Examples:

- CALCULATE
- FILTER
- ALL
- ALLEXCEPT
- RELATED
- RELATEDTABLE
- DISTINCT
- VALUES
- EARLIER/EARLIEST
- HASONESVALUE
- HASONEFILTER
- ISFILTERED
- USERRELATIONSHIP

DATE & TIME Functions

Basic **date and time** functions as well as advanced **time intelligence** operations

Common Examples:

- DATEDIFF
- YEARFRAC
- YEAR/MONTH/DAY
- HOUR/MINUTE/SECOND
- TODAY/NOW
- WEEKDAY/WEEKNUM

Time Intelligence Functions:

- DATESYTD
- DATESQTD
- DATESMTD
- DATEADD
- DATESINPERIOD

***Note:** This is NOT a comprehensive list (does not include trigonometry functions, parent/child functions, information functions, or other less common functions)

*Copyright 2018, Excel Maven & Maven Analytics, LLC

BASIC DATE & TIME FUNCTIONS

DAY/MONTH/ YEAR()

Returns the day of the month (1-31), month of the year (1-12), or year of a given date

=**DAY/MONTH/YEAR**(Date)

HOUR/MINUTE/ SECOND()

Returns the hour (0-23), minute (0-59), or second (0-59) of a given datetime value

=**HOUR/MINUTE/SECOND**(Datetime)

TODAY/NOW()

Returns the current date or exact time

=**TODAY/NOW**()

WEEKDAY/ WEEKNUM()

Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year

=**WEEKDAY/WEEKNUM**(Date, [ReturnType])

EOMONTH()

Returns the date of the last day of the month, +/- a specified number of months

=**EOMONTH**(StartDate, Months)

DATEDIFF()

Returns the difference between two dates, based on a selected interval

=**DATEDIFF**(Date1, Date2, Interval)

*Copyright 2018, Excel Maven & Maven Analytics, LLC

BASIC LOGICAL FUNCTIONS (IF/AND/OR)

IF()

Checks if a given condition is met, and returns one value if the condition is TRUE, and another if the condition is FALSE

=**IF**(LogicalTest, ResultIfTrue, [ResultIfFalse])

IFERROR()

Evaluates an expression and returns a specified value if the expression returns an error, otherwise returns the expression itself

=**IFERROR**(Value, ValueIfError)

AND()

Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE, otherwise returns FALSE

=**AND**(Logical1, Logical2)

OR()

Checks whether one of the arguments is TRUE to return TRUE, and returns FALSE if both arguments are FALSE

=**OR**(Logical1, Logical2)

Note: Use the **&&** and **||** operators if you want to include more than two conditions!

*Copyright 2018, Excel Maven & Maven Analytics, LLC

TEXT FUNCTIONS

LEN()

Returns the number of characters in a string

=**LEN**(Text)

Note: Use the **&** operator as a shortcut, or to combine more than two strings!

CONCATENATE()

Joins two text strings into one

=**CONCATENATE**(Text1, Text2)

**LEFT/MID/
RIGHT()**

Returns a number of characters from the start/middle/end of a text string

=**LEFT/RIGHT**(Text, [NumChars])
=**MID**(Text, StartPosition, NumChars)

**UPPER/LOWER/
PROPER()**

Converts letters in a string to upper/lower/proper case

=**UPPER/LOWER/PROPER**(Text)

SUBSTITUTE()

Replaces an instance of existing text with new text in a string

=**SUBSTITUTE**(Text, OldText, NewText, [InstanceNumber])

SEARCH()

Returns the position where a specified string or character is found, reading left to right

=**SEARCH**(FindText, WithinText, [StartPosition], [NotFoundValue])

*Copyright 2018, Excel Maven & Maven Analytics, LLC

RELATED

RELATED()

Returns related values in each row of a table based on relationships with other tables

=RELATED(ColumnName)



The column that contains the values you want to retrieve

Examples:

- Product_Lookup[ProductName]
- Territory_Lookup[Country]

HEY THIS IS IMPORTANT!

RELATED works almost *exactly* like a **VLOOKUP** function – it uses the relationship between tables (defined by primary and foreign keys) to pull values from one table into a new column of another. Since this function requires row context, it can only be used as a **calculated column** or as part of an **iterator function** that cycles through all rows in a table (*FILTER, SUMX, MAXX, etc*)



PRO TIP:

Avoid using **RELATED** to create redundant calculated columns unless you absolutely need them, since those extra columns increase file size; instead, use **RELATED** within a measure like **FILTER** or **SUMX**

*Copyright 2018, Excel Maven & Maven Analytics, LLC

BASIC MATH & STATS FUNCTIONS

SUM()

Evaluates the sum of a column

=SUM(ColumnName)

AVERAGE()

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

=AVERAGE(ColumnName)

MAX()

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

=MAX(ColumnName) or =MAX(Scalar1, [Scalar2])

MIN()

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

=MIN(ColumnName) or =MIN(Scalar1, [Scalar2])

DIVIDE()

Performs division and returns the alternate result (or blank) if div/0

=DIVIDE(Numerator, Denominator, [AlternateResult])

*Copyright 2018, Excel Maven & Maven Analytics, LLC

COUNT, COUNTA, DISTINCTCOUNT & COUNTROWS

COUNT()

Counts the number of cells in a column that contain numbers

=**COUNT**(ColumnName)

COUNTA()

Counts the number of non-empty cells in a column (numerical and non-numerical)

=**COUNTA**(ColumnName)

DISTINCTCOUNT()

Counts the number of distinct or unique values in a column

=**DISTINCTCOUNT**(ColumnName)

COUNTROWS()

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

=**COUNTROWS**(Table)

*Copyright 2018, Excel Maven & Maven Analytics, LLC

CALCULATE

CALCULATE()

Evaluates a given expression or formula under a set of defined filters

=**CALCULATE**(Expression, [Filter1], [Filter2],...)

Name of an existing measure, or a DAX formula for a valid measure

Examples:

- [Total Orders]
- SUM>Returns_Data[ReturnQuantity])

List of simple Boolean (True/False) filter expressions (**note:** these require simple, fixed values; you cannot create filters based on measures)

Examples:

- Territory_Lookup[Country] = "USA"
- Calendar[Year] > 1998






PRO TIP:

CALCULATE works just like **SUMIF** or **COUNTIF** in Excel, except it can evaluate measures based on ANY sort of calculation (not just a sum, count, etc); it may help to think of it like "CALCULATEIF"

*Copyright 2018, Excel Maven & Maven Analytics, LLC

CALCULATE (EXAMPLE)

  Bike Returns = CALCULATE([Total Returns], Products[CategoryName] = "Bikes") 

CategoryName	Total Returns	Bike Returns
Accessories	1,115	342
Bikes	342	342
Clothing	267	342
Components		342
Total	1,724	342

Here we've defined a new measure named "**Bike Returns**", which evaluates the "**Total Returns**" measure when the **CategoryName** in the **Products** table equals "**Bikes**"

Wait, why do we see the **same repeating values** when we view a matrix with different categories on rows?

Shouldn't these cells have different filter contexts for **Accessories**, **Clothing**, **Components**, etc?



HEY THIS IS IMPORTANT!

CALCULATE **modifies** and **overrides** any competing filter context!

In this example, the "Clothing" row has filter context of CategoryName = "**Clothing**" (defined by the row label) **and** CategoryName= "**Bikes**" (defined by the CALCULATE function)

Both cannot be true at the same time, so the "**Clothing**" filter is overwritten and the "**Bikes**" filter (from CALCULATE) takes priority

*Copyright 2018, Excel Maven & Maven Analytics, LLC

ALL

ALL()

Returns all rows in a table, or all values in a column, ignoring any filters that have been applied

=ALL(Table or ColumnName, [ColumnName1], [ColumnName2],...)

The table or column that you want to clear filters on

Examples:

- Transactions
- Products[ProductCategory]

List of columns that you want to clear filters on (optional)

Notes:

- If your first parameter is a table, you can't specify additional columns
- All columns must include the table name, and come from the same table

Examples:

- Customer_Lookup[CustomerCity], Customer_Lookup[CustomerCountry]
- Products[ProductName]



PRO TIP:

Instead of adding filter context, ALL **removes it**. This is often used when you need unfiltered values that won't react to changes in filter context (i.e. **% of Total**, where the denominator needs to remain fixed)

*Copyright 2018, Excel Maven & Maven Analytics, LLC

FILTER

FILTER()

Returns a table that represents a subset of another table or expression

=**FILTER**(Table, FilterExpression)

Table to be filtered

Examples:

- Territory_Lookup
- Customer_Lookup

A Boolean (True/False) filter expression to be evaluated for each row of the table

Examples:

- Territory_Lookup[Country] = "USA"
- Calendar[Year] = 1998
- Products[Price] > [Overall Avg Price]

HEY THIS IS IMPORTANT!

FILTER is used to add new filter context, and can handle **more complex filter expressions** than CALCULATE (by referencing measures, for example)

Since FILTER returns an entire table, it's almost always used as an *input* to other functions, like CALCULATE or SUMX



PRO TIP:

Since FILTER iterates through each row in a table, it can be slow and processor-intensive; don't use FILTER if a CALCULATE function will accomplish the same thing

*Copyright 2018, Excel Maven & Maven Analytics, LLC

ITERATOR ("X") FUNCTIONS

Iterator (or "X") functions allow you to loop through the same calculation or expression on *each row of a table*, and then apply some sort of aggregation to the results (SUM, MAX, etc)

=**SUMX**(Table, Expression)

Aggregation to apply to calculated rows*

Examples:

- SUMX
- COUNTX
- AVERAGEX
- RANKX
- MAXX/MINX

Table in which the expression will be evaluated

Examples:

- Sales
- FILTER(Sales, RELATED(Products[Category])="Clothing")

Expression to be evaluated for each row of the given table

Examples:

- [Total Orders]
- Sales[RetailPrice] * Sales[Quantity]



PRO TIP:

Imagine the function **adding a temporary new column** to the table, calculating the value in each row (based on the expression) and then applying the aggregation to that new column (like SUMPRODUCT)

*In this example we're looking at SUMX, but other "X" functions follow a similar syntax

*Copyright 2018, Excel Maven & Maven Analytics, LLC

TIME INTELLIGENCE FORMULAS

Time Intelligence functions allow you to easily calculate common time comparisons:

Performance
To-Date

=**CALCULATE**(Measure, **DATESYTD**(Calendar[Date]))

→ Use **DATESQTD** for Quarters or **DATESMTD** for Months

Previous
Period

=**CALCULATE**(Measure, **DATEADD**(Calendar[Date], -1, **MONTH**))

Select an interval (**DAY**, **MONTH**, **QUARTER**, or **YEAR**) and the
of intervals to compare (i.e. previous month, rolling 10-day)

Running
Total

=**CALCULATE**(Measure, **DATESINPERIOD**(Calendar[Date], **MAX**(Calendar[Date]), -10, **DAY**))



PRO TIP:

To calculate a **moving average**, use the running total calculation above and divide by the number of intervals