# DAX cheatsheet

## Introduction

As you discovered earlier in this lesson, DAX, or data analysis expression, is a programming language that provides a library of functions, operators, and constants for creating additional information about a data model.

Mastering DAX requires understanding its fundamentals, like the syntax, data types, operators, and how to refer to columns and tables.

This reading offers an overview of these fundamental elements of DAX alongside specific examples. Through careful study of this material, you'll develop your understanding of DAX.

## DAX syntax

A DAX formula always starts with an equal operator (**=**). After the equals sign, you can provide any expression that evaluates to a scalar or can be converted to a scalar. Let us understand the syntax with an example by breaking down the following DAX measure formula:

```
Total Sales = SUM (Sales[Sales Amount])
```

- **Total Sales:** The name of the calculated column.
- **Equal operator:** Indicates the beginning of the DAX formula.
- **SUM**: An aggregation function of DAX
- **Parenthesis ( ):** Grouping of arguments
- **Sales**: Table to be referenced
- **Square brackets [Sales Amount]**: The square brackets contain the referenced column, which is also the argument. The value of this argument must be passed to the function.

## Data types in DAX

DAX can perform computations on different data types, which include the following:

- Text (Binary): "Hello, world".
- Decimal (Float): 1.23.

- Whole number (Integer): 123.
- Boolean: TRUE or FALSE.
- Date (Date/Time): DATE(2023,5,11).
- Currency: A fixed decimal number.

**DAX Operators**

DAX formulas rely on operators to perform arithmetic calculations, compare values, work with strings, or test conditions.

Here is an overview of some commonly used operators in DAX:

| Operator Type | Symbol | Application | Example |
|---|---|---|---|
| **Parenthesis** | **()** | Grouping of arguments and precedence order | **(5+7) * 5** |
| **Arithmetic** | **+** | Addition | **5 + 3 = 8** |
| | **-** | Subtraction | **9 – 5 = 4** |
| | ***** | Multiplication | **3 * 9 = 27** |
| | **/** | Division | **18/3 = 6** |
| | **^** | Exponentiation | **16^4 = 65536** |
| **Logical** | **&&** | AND condition between two Boolean expressions | **[Region] = "USA" && [Quantity] > 5** |
| | **\|\|** | OR condition between two Boolean expressions | **[Region] = "USA" \|\| [Quantity] > 5** |
| **Comparison** | **=** | Equal to | **[Region] = "USA"** |
| | **<>** | Not equal to | **Region] <> "USA"** |
| | **>** | Greater than | **[Quantity] > 5** |
| | **>=** | Greater than or equal to | **[Quantity] >= 10** |
| | **<** | Less than | **[Quantity] < 5** |
| | **<=** | Less than or equal to | **[Quantity] <= 5** |

|  |  |  |  |
|---|---|---|---|
| **Text Concatenation** | **&** | Concatenation of strings | **[Region] & ", " & [City]** |

**DAX Functions**

A function is a named formula within an expression. Most functions have required and optional arguments, also called parameters, as input. When the function is executed, a value is returned.

DAX includes functions to perform calculations using dates and times, create conditional values, work with strings, perform lookups based on relationships, and iterate over a table to perform recursive calculations. Some of the most used classes of DAX functions are given below.

**Text Functions**

You can use these functions to return part of a string, search for text within a string or concatenate string values to create a new column.

With the **CONCATENATE** function, you can join two text strings into one text string. For example, you can combine an employee's first name and last name into a new column by defining a DAX formula as follows:

1

**Full Name =**

**CONCATENATE (Employees[FirstName], CONCATENATE ( " ", Employees[LastName] ) )**

The **LEFT** function returns the leftmost characters from a text value. For example, you could create a column that shortens the names of the months for better visualization purposes. The **LEFT** function allows you to create a column with only the first three letters of each month as follows:

1

**Short Name = LEFT (Date[Month], 3)**

**Date/Time Functions**

These functions in DAX are like date and time functions in Microsoft Excel. However, DAX functions are based on the datetime data types used by Microsoft SQL Server.

- **NOW()**: The NOW function displays the current date and time on a worksheet or calculates a value based on the current date and time. It updates the value each time you open the worksheet.

- **YEAR(<date>)**: Returns the year of a date as a four-digit integer from the date column. You can add a column for a year from your date table.
- **MONTH(<date>)**: Returns the month of a date as a number (1 - 12).

**Logical Functions**

These functions evaluate logical conditions and return true or false values.

- **IF(<logical_test>, <value_if_true>, <value_if_false>):** Returns one value if a condition is true and another value if it is false.
- **AND(<logical1>, <logical2>):** Returns TRUE if all its arguments are TRUE; returns FALSE if one or more argument is FALSE.
- **OR(<logical1>, <logical2>):** Returns TRUE if any argument is TRUE; returns FALSE if all arguments are FALSE.
- **NOT(<logical>):** Reverses the logic of its argument.

**Aggregation Functions**

These functions perform aggregations. Commonly these functions create sums and averages and find minimum and maximum values. You can also filter a column in DAX based on related tables, before creating aggregations. Common aggregation functions are:

- **COUNT(<column>):** Returns the count, or total, of all rows in a column.
- **SUM(<column>):** Returns the sum of all values in a column.
- **AVERAGE(<column>):** Returns the average of all values in a column.
- **MIN(<column>):** Returns the smallest value in a column.
- **MAX(<column>):** Returns the largest value in a column.

**Time Intelligence Functions**

These functions create calculations using built-in knowledge of calendars and dates. You can build meaningful comparisons across comparable periods for sales, inventory, and so on using time and date ranges combined with aggregations or calculations.

- **SAMEPERIODLASTYEAR(<dates>):** Returns a parallel period calculated against the dates provided.
- **DATESYTD(<dates>):** Returns dates from the beginning of the year until the last date in the dates column provided.

- **DATESMTD(<dates>):** Returns dates from the beginning of the month until the last date in the provided dates column.
- **DATESQTD(<dates>):** Returns dates from the beginning of the quarter until the last date in the provided dates column.
- **EDATE(<start_date>, <months>):** Returns the date that is a specified number of months before or after the start date.

**Statistical Functions**

These functions calculate values related to statistical distributions and probability, such as standard deviation and number of permutations. Common statistical functions are:

- **STDEV.P(<ColumnName>):** Returns the standard deviation of the entire population.
- **MEDIAN(<column>):** Returns the median of numbers in a column.
- **RANKX(<table>, <expression>[, <value>[, <order>[, <ties>]]]):** Returns the ranking of a number in a list of numbers for each row in the table argument.

**Relational Functions**

These functions are for managing and utilizing relationships between tables. For example, you can specify a particular relationship to be used in a calculation.

- **RELATED(<column>):** Returns a related value from another table.
- **RELATEDTABLE:** Evaluates a table expression in a context modified by the given filters.

**Information Functions**

These functions look at a table or column provided as an argument to another function and determine if the value matches the expected type.

- **ISBLANK(<expression>):** Checks if a value is blank and returns TRUE or FALSE.
- **CONTAINS(<table>, <column>, <value>):** Checks if the values in a column already exist in another column and returns a value of TRUE or FALSE.

**Points to remember:**

- DAX is not case-sensitive but distinguishes between blanks and zeros.

- Use comments to explain your code. You can use **//** for a single-line comment and **/* ... */** for a multi-line comment.
- Remember, many DAX functions require an existing relationship between tables, so ensure your data model is set up correctly.
- Study row and filter context, as they are fundamental to understanding and using DAX effectively.

**Conclusion**

Remember, practice makes perfect. Spend time learning and experimenting with different DAX functions and syntaxes using sample datasets to improve your understanding and proficiency. For more guidance, you can consult the Microsoft Learn guide to the basics of DAX.