Excel DAX - Operators

DAX is a formula language comprising of functions, operators, and values that can be used in a formula or expression, to calculate and return one or more values.

You can use **DAX operators** to compare values, perform arithmetic calculations, and concatenate strings. In this chapter, you will learn about DAX operators and how to use them.

Types of DAX Operators

DAX supports the following types of operators −

- DAX Arithmetic Operators
- DAX Comparison Operators
- DAX Text Concatenation Operator
- DAX Logical Operators

DAX Operator Precedence Order

You can have a DAX formula with many DAX operators combining several values or expressions. In such a case, the final result will depend on the order in which the operations are performed. DAX provides you with the default operator precedence order and also ways of overriding the default precedence order.

DAX default operator precedence is listed in the following table.

Precedence Order	Operator(s)	Operation
1	^	Exponentiation
2	–	Sign
3	* and /	Multiplication and Division
4	!	NOT
5	+ and –	Addition and Subtraction
6	&	Concatenation
7	=, <, >, <=, >= and <>	Equal to, Less than, Greater than, Less than or equal to, Greater than or equal to and Not equal to

DAX Expression Syntax

You need to first understand the DAX expression syntax and how the expression evaluation is done with the operands and operators.

- All expressions always begin with an equal sign (=). The equal sign indicates that the succeeding characters constitute an expression.
- To the right of the equal sign, you will have the operands connected by the DAX operators. For example, = 5 + 4 > 5.

  = 5 * 6 - 3.
- Expressions are always read from left to right, and the calculations are done in that sequence, based on the DAX operator precedence given in the previous section.
- If the DAX operators have equal precedence value, they are evaluated from the left to right. For example, =5*6/10. Both * and / have same the precedent order. Hence, the expression is evaluated as 30/10 = 3.
- If the DAX operators in the expression have different precedence values, then they are evaluated in the precedence order from the left to right.
  - = 5 + 4 > 7. Default precedence is + first and > next. Hence, the expression is calculated from the left to right. - 5 + 4 is calculated first resulting in 9 and then 9 > 5 is evaluated that results in TRUE.
  - = 5 * 6 - 3. Default precedence is * first and - next. Hence, the expression is calculated from the left to right. - 5 * 6 is calculated first resulting in 30 and then 30 - 3 is calculated that results in 27.
  - = 2 * 5 - 6 * 3. Default precedence is * first, * next and then -. Hence, the expression evaluates as 10 – 18 and then as -8. Note, that it is not 10 - 6 resulting in 4 and then 4*3 that is 12.

Using Parentheses to Control DAX Calculation Order

You can change the DAX default operator precedence order by using parentheses, grouping the operands and the operators to control the calculation sequence.

For example, = 5 * 6 - 3 evaluates to 27 with the DAX default operator precedence order. If you use parenthesis to group the operands and operators as = 5 * (6 - 3), then 6 - 3 is calculated first resulting in 3 and then 5 * 3 is calculated which results in 15.

= 2 * 5 - 6 * 3 evaluates to -8 with the DAX default operator precedence order. If you use parenthesis to group the operands and operators as = 2 * (5 - 6) * 3, then 5 - 6 is calculated first resulting in -1 and then 2 * (-1) * 3 is calculated which results in -6.

As you can see, with the same operands and operators, different results are possible by the way you group them. Hence, when you use the DAX operators in the DAX formulas, you should pay attention to how the computation sequence is to be.

Differences Between Excel and DAX

Though DAX has similarities with Excel formulas, there are certain significant differences between the two.

- DAX is more powerful than Excel because of its underlying memory resident computation engine.
- DAX supports more data types than Excel.
- DAX provides additional advanced features of a relational database, Data Model, including richer support for date and time types.

In some cases, the results of calculations or the behavior of functions in DAX may not be the same as in Excel. This is due to the differences in the following −

- Data type casting
- Data types

Difference in Data Type Casting

In DAX, when you have an expression =value1 operator value2, the two operands value1 and value2 should be of the same data type. If the data types are different, DAX will convert them first to a common data type implicitly. Refer to the chapter – DAX Syntax for details.

For example, you have to compare two operands of different data types, say a number resulting from a formula, such as =[Amount] * 0.08 and an integer. The first number can be a decimal number with many decimal places, whereas the second number is an integer. Then DAX handles it as follows −

- First, DAX will convert both the operands to real numbers using the largest numeric format that can store both kinds of numbers.
- Next, DAX will compare the two real numbers.

In contrast, Excel tries to compare values of different data types without first coercing them to a common data type. For this reason, you might find different results in DAX and in Excel for the same comparison expression.

Difference in Data Types

The operator precedence order in DAX and Excel is the same. However, the operator percent (%) and data ranges that Excel supports are not supported by DAX. Moreover, DAX supports table as a data type, which is not the case in Excel.

Further, in Excel formulas, you can refer to a single cell, or an array or a range of cells. In DAX formulas, you cannot refer to any of these. The DAX formula references to data should be by tables, columns, calculated fields, and calculated columns.

If you copy formulas from Excel and paste them in DAX, ensure the correctness of the DAX formula as DAX syntax is different from Excel formula syntax. Also, even if a function has the same name in DAX and Excel, its parameters might be different and the result of the function can also be different.

You will learn more about all these in the subsequent chapters.