

DAX Language Quick Reference Guide



Compiled by

EXcelerator BI
Power BI Solutions for Business

This DAX functions quick reference guide has been prepared by Matt Allington from <http://exceleratorbi.com.au> and contains a list of all the current DAX functions in a summarised and easy to use format. You can print the document and/or use the hyperlinks/search features for PDF documents to find the DAX function you are looking for.

Tip: If you are looking for a function name in this document you can

- Navigate to the [DAX Functions List](#) wherein the functions are arranged in chronological order, locate the function and click on it to go to the function syntax and description.
- Use search, type the function name followed by an open bracket. E.g. instead of searching for VALUES you should search **VALUES(**.

Note: This document is a supplement and is not intended to replace the more detailed Microsoft documentation that is available online. When looking for online documentation it is best to do a web search from your favourite search engine by specifying the function name followed by the word DAX i.e. **"FunctionName DAX"**.

Table of Contents

DAX Aggregation Functions (Aggregators).....	3
DAX Date and Time Functions	5
DAX Filter Functions	8
DAX Financial Functions	16
DAX Information Functions	31
DAX Logical Functions.....	36
DAX Math and Trig Functions	39
DAX Other Functions	44
DAX Other Special Functions (X-Functions / Iterators)	46
DAX Other Special Functions (Argument Functions).....	49
DAX Parent and Child Functions	51
DAX Query Functions.....	52
DAX Statistical Functions	56
DAX Text Functions.....	60
DAX Time Intelligence Functions	64
DAX Functions List	73

More Great Ways to Learn



Power BI (DAX)
Online Training

<https://xbi.com.au/swspbi>



Advanced DAX
Online Training

<https://xbi.com.au/swadax>



Power BI for the Business Analyst
Online Training

<https://xbi.com.au/pbilon>



Become a Data Master
Online Training

<https://xbi.com.au/swpqa>



Power BI Live Online
Instructor-Led Training

<https://xbi.com.au/swpbi4ba>



Learn to Write DAX
in Power BI/Excel

<https://xbi.com.au/daxbundle>

DAX Aggregation Functions (Aggregators)

DAX **Aggregation Functions** (called **aggregators** for short) take a column or a table as the argument and aggregate the values.

Function	Notes
APPROXIMATEDISTINCTCOUNT(column)	Returns the approximate number of rows that contain distinct values in a column. This function can query large amounts of data with potentially better performance than DISTINCTCOUNT , with slight deviation from the exact result.
AVERAGE(column)	Returns the average (arithmetic mean) of all the numbers in a column in the current filter context. This function is the equivalent of adding up the values in the column and then dividing by the number of rows.
AVERAGEA(column)	The AVERAGEA function takes a column and averages the numbers in it, but also handles non-numeric data types according to the following rules: <ul style="list-style-type: none"> • Values that evaluates to TRUE count as 1. • Values that evaluate to FALSE count as 0 (zero). • Values that contain non-numeric text count as 0 (zero). • Empty text ("") counts as 0 (zero).
COUNT(column)	Counts numbers only in the current filter context.
COUNTA(column)	Counts text values as well as numbers in the current filter context.
COUNTBLANK(column)	Counts the number of blank cells in a column in the current filter context.
COUNTROWS(table)	The COUNTROWS function counts the number of rows in the specified table, or in a table defined by an expression in the current filter context.
DISTINCTCOUNT(column)	Counts each value in a column once and only once in the current filter context.
DISTINCTCOUNTNOBLANK(column)	Counts the number of distinct values in a column. Unlike DISTINCTCOUNT function, DISTINCTCOUNTNOBLANK does not include the BLANK value.
MAX(column) MAX(expression1, expression2)	The MAX function takes a column or two expressions that return numeric values as argument(s) and returns the largest numeric value. Ignores logical values and text in the current filter context. Can also find the MAX of a text column.
MAXA(column)	Like MAX , however also considers Dates and Logical values, such as TRUE and FALSE . Rows that evaluate to TRUE count as 1; rows that evaluate to FALSE count as 0 (zero)

Function	Notes
MIN(column) MIN(expression1, expression2)	The MIN function takes a column or two expressions that return numeric values as argument(s) and returns the smallest numeric value. Ignores logical values and text in the current filter context.
MINA(column)	Like MIN , however also considers Dates and Logical Values. Rows that evaluate to TRUE count as 1; rows that evaluate to FALSE count as 0 (zero)
PRODUCT(column)	Multiplies all the values in a column together. Why you would want to? I have no idea.
SUM(column)	Adds all the numbers in a column in the current filter context.
TOPN(n_value, tablename, orderByexpression1, [Order], [orderByexpression2, [Order]], ...)	Returns a table containing the top N rows. Order by expression is typically a measure that you want to rank on.

DAX Date and Time Functions

You can use [DAX Date and Time Functions](#) in the calculations based on dates and time. DAX Date and Time Functions are like the Excel date and time functions but use a **datetime** data type and can take values from a column as an argument.

Function	Notes
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.
CALENDARAUTO ([end month of fiscal year])	Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is calculated automatically based on data in the model.
DATE (year , month , day)	Returns the specified date in datetime format. <ul style="list-style-type: none"> Dates beginning with March 1, 1900 are supported. If the year value is between 0 and 1899, the value is added to 1900 to produce the final value. The value of the year argument can include one to four digits. You should use four digits for the year argument whenever possible to prevent unwanted results.
DATEDIFF (start date , end date , interval)	Returns the count of interval boundaries crossed between two dates. The interval to use when comparing dates can be one of the following: <ul style="list-style-type: none"> SECOND MINUTE HOUR DAY WEEK MONTH QUARTER YEAR If start_date is larger than end_date an error is returned.
DATEVALUE (date text)	Converts a date in the form of text to a date in datetime format.
DAY (date)	Returns the day of the month, a number from 1 to 31.
EDATE (start date , months)	Returns the date that is the indicated number of months before or after the start date. You can use EDATE to calculate maturity dates or due dates that fall on the same day of the month as the date of issue. You can also use EDATE to find the same date prior month or prior year.

Function	Notes
EOMONTH (start date , months)	Returns the date in datetime format of the last day of the month, before or after a specified number of months. You can use EOMONTH to calculate maturity dates or due dates that fall on the last day of the month.
 HOUR (datetime value)	Returns the hour as a number from 0 (12:00 A.M.) to 23 (11:00 P.M.).
 MINUTE (datetime value)	Returns the minute as a number from 0 to 59, given a date and time value.
 MONTH (datetime value)	Returns the month as a number from 1 (January) to 12 (December).
 NETWORKDAYS (start date , end date [, <weekend>, <holidays>])	Returns the number of whole workdays between two dates (inclusive). weekend indicates the weekend days and they are not included in the calculation of the number of days. weekend can be omitted or can be a number between 1 and 17. if omitted or if it is 1 then the weekend is Saturday & Sunday. the numbers from 2 to 17 represent different combinations of days in the week that are the weekend. holidays is a table with a column of one or more dates that are to be excluded from the working day calendar.
 NOW ()	Returns the current date and time in datetime format. The NOW function is useful when you need to display the current date and time on a report or calculate a value based on the current date and time, and have that value updated each time you open the workbook.
 QUARTER (datetime value)	Returns the quarter as a number from 1 to 4. 1 (Jan – Mar) ... 4 (Oct – Dec) If the input value is BLANK, the output value is also BLANK.
 SECOND (datetime value)	Returns the seconds of a time value, as a number from 0 to 59.
 TIME (hour , minute , second)	Converts hours, minutes, and seconds given as numbers to a time in datetime format.
 TIMEVALUE (time text)	Converts a time in text format to a time in datetime format.
 TODAY ()	Returns the current date in datetime format. The TODAY function is useful when you need to have the current date displayed on a report. It is also useful for calculating intervals.

Function	Notes
UTCNOW()	Returns the current UTC date and time.
UTCTODAY()	Returns the current UTC date.
WEEKDAY(date, return type)	Returns a number from 1 to 7 identifying the day of the week of a date. <ul style="list-style-type: none"> If return type is 1, and the week begins on Sunday (1) and ends on Saturday (7). If return type is 2, the week begins on Monday (1) and ends on Sunday (7). If return type is 3, the week begins on Monday (0) and ends on Sunday (6).
WEEKNUM(date, return type)	Returns the week number for the given date in a year according to the return_type value. The week number indicates where the week falls numerically within a year. If return type is 1, week begins on Sunday. Weekdays are numbered 1 through 7. If return type is 2, week begins on Monday. Weekdays are numbered 1 through 7.
YEAR(datetime value)	Returns the year of a date as a four-digit integer in the range 1900-9999.
YEARFRAC(start_date, end_date, [basis])	Calculates the fraction of the year represented by the number of whole days between two dates. Use the YEARFRAC function to identify the proportion of a whole year's benefits or obligations to assign to a specific term. Basis (optional) is the type of day count basis to use: 0 - US (NASD) 30/360 1 - Actual/actual 2 - Actual/360 3 - Actual/365 4 - European 30/360

DAX Filter Functions

[DAX Filter Functions](#) are very different to Excel functions. They are used to (typically) return filtered tables that can be used in your data model. These new “virtual” tables retain lineage with the physical data model and hence they can “filter” the physical data model on the fly. Lookup functions work by using tables and relationships between them. Filtering functions let you manipulate data context to create dynamic calculations.

DAX **FILTER** and **VALUES** functions are the most complex and powerful functions.

Function	Notes
ADDMISSINGITEMS (showAllColumn[, showAllColumn] ..., table, groupingColumn[, groupingColumn] ... [, filterTable] ...)	<p>Adds combinations of items from multiple columns to a table if they do not already exist. The determination of which item combinations to add is based on referencing source columns which contain all the possible values for the columns.</p> <p>To determine the combinations of items from different columns to evaluate: AutoExist is applied for columns within the same table while CrossJoin is applied across different tables.</p>
ADDMISSINGITEMS (showAllColumn[, showAllColumn] ..., table, [ROLLUPISSUBTOTAL(groupingColumn[, isSubtotal_columnName] [, groupingColumn][, isSubtotal_columnName] ...)], [, filterTable] ...)	<p>The ADDMISSINGITEMS function will return BLANK values for the IsSubtotal columns of blank rows it adds.</p> <p>See also: The argument functions – ROLLUPISSUBTOTAL and ROLLUPGROUP.</p>

Function	Notes
ALL([TableOrColumn] [, TableOrColumn] ...)	<p>Updated in Power BI Desktop to include ALL().</p> <p>Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied in the current context.</p> <p>If Table is used as the argument, this function is useful for clearing filters and creating calculations on all the rows in a table.</p> <p>If specific Columns are used as the arguments, this function removes all filters from the specified columns in the table and all other filters on other columns in the table still apply.</p> <p>This is useful when you want to remove the context filters for one or more specific columns and to keep all other context filters.</p> <p>ALL() removes all the filters everywhere. ALL() can only be used to clear filters but not to return a table.</p>
ALLCROSSFILTERED(table)	<p>Clear all filters which are applied to a table.</p> <p>ALLCROSSFILTERED can only be used to clear filters but not to return a table.</p>
ALLEXCEPT(table, column [, column] ...)	<p>Removes all context filters in the table except filters that are applied to the specified columns.</p> <p>This is convenient to use when you want to remove the filters on many, but not all, columns in a table.</p>
ALLNOBLANKROW(table column)	<p>When the passed parameter was a table, returns all rows but the blank row from the parent table of a relationship and disregards any context filters that might exist.</p> <p>When the passed parameter was a column, returns all distinct values of the column but the blank row, and disregards any context filters that might exist.</p>

Function	Notes
ALLSELECTED ([tableName columnName])	<p>The ALLSELECTED function gets the context that represents all rows and columns in the query, while keeping explicit filters and contexts other than row and column filters. This function can be used to obtain visual totals in queries.</p> <p>Keeps filters on Rows and Columns in a pivot table while keeping the filters on slicers and other explicit filters.</p> <p>This function is different from ALL because it retains all filters explicitly set within the query, and it retains all context filters other than row and column filters.</p>
CALCULATE (expression [, filter1] [, filter2] ...)	<p>Returns a scalar value.</p> <p>Modifies the initial filter context prior to calculating the expression. It can do this in 2 ways, by applying the new filters specified in the CALCULATE function and/or by converting an existing row context into an equivalent filter context aka context transition.</p> <p>expression is same as a measure.</p> <p>Filters can be:</p> <ul style="list-style-type: none"> • Boolean filter expressions • Table filter expressions • Filter modification functions <p>Boolean filter expressions can contain an aggregation function that returns a scalar value.</p> <p>When there are multiple filters, they can be evaluated by using AND (&&) meaning all conditions must be TRUE, or by OR (), meaning either condition can be true.</p>

Function	Notes
CALCULATE (expression , filter1 , filter2, ...)	<p>Returns a table of values.</p> <p>Modifies the filter context prior to returning a table of values. It can do this in 2 ways, by applying the new filters specified in the CALCULATE function and/or by converting an existing row context into an equivalent filter context aka context transition.</p> <p>expression must be a table in the data model or a function that returns a table.</p> <p>Filters can be:</p> <ul style="list-style-type: none"> • Boolean filter expressions • Table filter expressions • Filter modification functions <p>Boolean filter expressions can contain an aggregation function that returns a scalar value.</p> <p>When there are multiple filters, they can be evaluated by using AND (&&) meaning all conditions must be TRUE.</p>
CROSSFILTER (columnName1 , columnName2 , direction)	<p>Sets the cross-filtering direction for the indicated relationship, for the duration of the query. It does not return any value.</p> <p>You can use CROSSFILTER inside CALCULATE.</p> <p>The cross-filter direction to be used is determined by the argument – direction.</p> <p>None - No cross-filtering occurs along this relationship</p> <p>Both - Filters on either side filter the other</p> <p>OneWay - Filters on the one or lookup side of the side of the relationship filter the many side.</p> <p>OneWay_LeftFiltersRight - Filters on the side of columnName1 filter the side of columnName2.</p> <p>OneWay_RightFiltersLeft - Filters on the side of columnName2 filter the side of columnName1.</p>
DISTINCT (column)	<p>Returns a 1 column table of all the distinct values in the current filter context. If there are BLANKS then they will be ignored. If you want to return a BLANK as well, then use VALUES function instead.</p>
DISTINCT (table)	<p>Returns a table by removing duplicate rows from another table or expression.</p> <p>The returned table will contain unique (or distinct) rows.</p>

Function	Notes
EARLIER (column [, number])	Used to access a previous row context when more than 1 row context exists in the function. EARLIER is useful for nested calculations where you want to use a certain value as an input and produce calculations based on that input.
EARLIEST (column)	As above, but returns the absolute first row context.
FILTER (table, filter)	Returns a table containing only the filtered rows. You can use FILTER to use only specific data in calculations. FILTER is not used independently, but as a function that is embedded in other functions such as CALCULATE .
FILTERS (columnName)	Returns a table containing the list of values that are directly applied as filters.
HASONEFILTER (columnName)	Returns TRUE when the number of directly filtered values on columnName is one; otherwise returns FALSE. Used to check if there is one and only one filter on a column in the current filter context.
HASONEVALUE (columnName)	Returns TRUE when the context for columnName has been filtered down to one distinct value only. Otherwise is FALSE. Used to check if there is one and only one value visible in a column in the current filter context.
INDEX (position [, relation] [, orderBy] [, blanks] [, partitionBy])	Returns a row at an absolute position, specified by the position parameter, within the specified partition, sorted by the specified order. If the current partition can't be deduced to a single partition, multiple rows may be returned.
ISCROSSFILTERED (columnName)	Returns TRUE when columnName or another column in the same or related table is being filtered. Otherwise returns FALSE. Used to check if there is an indirect filter on a column in the current filter context.

Function	Notes
ISFILTERED(columnName)	<p>Returns TRUE when columnName is being filtered directly. If there is no filter on the column or if the filtering happens because a different column in the same table or in a related table is being filtered, then the function returns FALSE.</p> <p>Used to check if there the column is filtered at all in the current filter context.</p>
KEEPFILTERS(expression)	<p>You use KEEPFILTERS within the context CALCULATE and CALCULATETABLE functions, to override the standard behaviour of those functions.</p> <p>CALCULATE filters replace the current context, while KEEPFILTERS adds filters to the current context.</p>
OFFSET(delta [, relation] [, orderBy] [, blanks] [, partitionBy])	<p>Returns a single row that is positioned either before or after the current row within the same table, by a given offset.</p> <p>If the current row cannot be deduced to a single row, multiple rows may be returned.</p>
RELATED(column)	<p>Returns a related value from another table. A single value that is related to the current row.</p> <p>Forces a row context to follow the relationship to a related table and return that value. Can only be used on the many side of the relationship.</p>
RELATEDTABLE(tableName)	<p>Returns a table of values from the many side of the relationship.</p> <p>Forces a row context to follow the relationship to a related table and return that value. Can only be used on the one side of the relationship.</p>
REMOVEFILTERS([table column[, column[, column[,...]]]])	<p>Clears filters from the specified table or columns.</p> <p>REMOVEFILTERS can only be used to clear filters but not to return a table.</p> <p>So it can only be used within CALCULATE.</p>

Function	Notes
SELECTEDVALUE (columnName [, alternateResult])	<p>It can be used as a substitute for IF(HASONEVALUE()) and will return a scalar value if there is one and only 1 value selected in the current filter context.</p> <p>Otherwise returns alternateResult.</p> <p>If alternateResult is omitted, the default value is BLANK().</p>
SUBSTITUTEWITHINDEX (table , indexColumnName , indexColumnsTable , orderBy_expression [, order] [, orderBy_expression [, order]] ...)	<p>Returns a table which represents a left semijoin of the two tables supplied as arguments. The semijoin is performed by using common columns, determined by common column names and common data type. The columns being joined on are replaced with a single column in the returned table which is of type integer and contains an index. The index is a reference into the right join table given a specified sort order.</p> <p>Columns in the right/second table supplied which do not exist in the left/first table supplied are not included in the returned table and are not used to join on.</p> <p>The index starts at 0 (0-based) and is incremented by one for each additional row in the right/second join table supplied. The index is based on the sort order specified for the right/second join table.</p>
TREATAS (table_expression , column1 [, column2] [, column3] ...)	<p>Returns a table that contains all the rows in column(s) that are also in table_expression.</p> <p>The number of columns specified must be equal to the number of columns in the table expression, and be in the same order.</p> <p>Use when a relationship does not exist between the tables.</p>
USERELATIONSHIP (columnName1 , columnName2)	<p>You can have more than 1 relationship between 2 tables in DAX, but only 1 can be active at a time. Use this function inside CALCULATE to use the inactive relationship instead of the active one.</p>
VALUES (TableNameOrColumnName)	<p>Returns a table consisting of a single column of unique values in the current filter context. If there are blanks in the list a blank will be returned.</p> <p>If you want to exclude the blank then use DISTINCT instead.</p>

Function	Notes
WINDOW (from [, from_type], to [, to_type] [, relation] [, orderBy] [, blanks] [, partitionBy])	Returns multiple rows which are positioned within the given interval.

DAX Financial Functions

[DAX Financial Functions](#) are used in formulas that perform financial calculations. These functions are similar to Excel financial functions.

Function	Notes
ACCRINT (issue , first_interest , settlement , rate , par , frequency [, basis [, calc_method]])	<p>Returns the accrued interest for a security that pays periodic interest.</p> <p>Formula used:</p> $ACCRINT = \text{par} \times \frac{\text{rate}}{\text{frequency}} \times \sum_{i=1}^{NC} \frac{A_i}{NL_i}$ <p>where:</p> <p>A_i = number of accrued days for the i^{th} quasi-coupon period within odd period.</p> <p>NC = number of quasi-coupon periods that fit in odd period. If this number contains a fraction, raise it to the next whole number.</p> <p>NL_i = normal length in days of the quasi-coupon period within odd period.</p>
ACCRINTM (issue , maturity , rate , par [, basis])	<p>Returns the accrued interest for a security that pays interest at maturity.</p> <p>Formula used:</p> $ACCRINTM = \text{par} \times \text{rate} \times \frac{A}{D}$ <p>where:</p> <p>A = Number of accrued days counted according to a monthly basis. For interest at maturity items, the number of days from the issue date to the maturity date is used.</p> <p>D = Annual Year Basis.</p>
AMORDEGRC (cost , date_purchased , first_period , salvage , period , rate [, basis])	<p>Returns the depreciation for each accounting period.</p> <p>This function is provided for the French accounting system.</p> <p>If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account.</p> <p>The function is similar to AMORLINC, except that a depreciation coefficient is applied in the calculation depending on the life of the assets.</p>

Function	Notes
AMORLINC (cost, date_purchased, first_period, salvage, period, rate [, basis])	<p>Returns the depreciation for each accounting period.</p> <p>This function is provided for the French accounting system.</p> <p>If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account.</p>
COUPDAYBS (settlement, maturity, frequency [, basis])	<p>Returns the number of days from the beginning of a coupon period until its settlement date.</p>
COUPDAYS (settlement, maturity, frequency [, basis])	<p>Returns the number of days in the coupon period that contains the settlement date.</p>
COUPDAYSNC (settlement, maturity, frequency [, basis])	<p>Returns the number of days from the settlement date to the next coupon date.</p>

Function	Notes
COUPNCD (settlement , maturity , frequency [, basis])	Returns the next coupon date after the settlement date.
COUPNUM (settlement , maturity , frequency [, basis])	Returns the number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole coupon.
COUPPCD (settlement , maturity , frequency [, basis])	Returns the previous coupon date before the settlement date.
CUMIPMT (rate , nper , pv , start_period , end_period , type)	Returns the cumulative interest paid on a loan between start_period and end_period. The units used for specifying rate and nper should be consistent.

Function	Notes
CUMPRINC (rate, nper, pv, start_period, end_period, type)	<p>Returns the cumulative principal paid on a loan between start_period and end_period.</p> <p>The units used for specifying rate and nper should be consistent.</p>
DB (cost, salvage, life, period [, month])	<p>Returns the depreciation of an asset for a specified period using the fixed-declining balance method.</p> <p>Formulas used:</p> <p>DB for a period</p> $(cost - \text{total depreciation from prior periods}) \times rate$ <p>where</p> <ul style="list-style-type: none"> • $rate = 1 - ((\frac{salvage}{cost})^{(\frac{1}{life})})$ • rate is rounded to 3 decimal places. <p>DB for the first period</p> $\frac{cost \times rate \times month}{12}$ <p>DB for the last period</p> $\frac{(cost - \text{total depreciation from prior periods}) \times rate \times (12 - month)}{12}$
DDB (cost, salvage, life, period [, factor])	<p>Returns the depreciation of an asset for a specified period using the double-declining balance method or some other method you specify.</p> <p>The parameter factor specifies the rate at which the balance declines. If factor is omitted, it is assumed to be 2 (the double-declining balance method).</p> <p>Formula used:</p> $\frac{\text{Min}((cost - \text{total depreciation from prior periods}) \times (\frac{factor}{life}), (cost - salvage - \text{total depreciation from prior periods}))}{1}$ <p>If you want to switch to the straight-line depreciation method when depreciation is greater than the declining balance calculation use the VDB function.</p>

Function	Notes
DISC (settlement , maturity , pr , redemption [, basis])	<p>Returns the discount rate for a security.</p> <p>Formula used:</p> $DISC = \frac{\text{redemption} - \text{par}}{\text{redemption}} \times \frac{B}{DSM}$ <p>where:</p> <ul style="list-style-type: none"> B = number of days in a year, depending on the year basis. DSM = number of days between settlement and maturity.
DOLLARDE (fractional_dollar , fraction)	<p>Converts a dollar price expressed as an integer part and a fraction part, separated by a decimal symbol (e.g. 1.02), into a dollar price expressed as a decimal number.</p> <p>The parameter fraction is the integer to divide the fraction part of the fractional_dollar. The choice of integer depends on the dollar precision you want.</p>
DOLLARFR (decimal_dollar , fraction)	<p>Converts a dollar price expressed as a decimal number into a dollar price expressed as an integer part and a fraction part, , separated by a decimal symbol (e.g. 1.02.)</p>
DURATION (settlement , maturity , coupon , yld , frequency [, basis])	<p>Returns the Macauley duration for an assumed par value of \$100.</p> <p>Duration is defined as the weighted average of the present value of cash flows and is used as a measure of a bond price's response to changes in yield.</p>
EFFECT (nominal_rate , npery)	<p>Returns the effective annual interest rate, given the nominal annual interest rate and the number of compounding periods per year.</p> <p>The parameter npery is used to specify the number of compounding periods per year</p> <p>Formula used:</p> $EFFECT = \left(1 + \frac{\text{nominal_rate}}{\text{npery}}\right)^{\text{npery}} - 1$

Function	Notes
FV(rate, nper, pmt [, pv[, type]])	<p>Calculates the future value of an investment based on a constant interest rate. You can use FV with either periodic, constant payments, and/or a single lump sum payment.</p> <p>Units used for specifying rate and nper should be consistent.</p>
INTRATE(settlement, maturity, investment, redemption [, basis])	<p>Returns the interest rate for a fully invested security.</p> <p>Formula used:</p> $\text{INTRATE} = \frac{\text{redemption} - \text{investment}}{\text{investment}} \times \frac{B}{\text{DIM}}$ <p>where</p> <ul style="list-style-type: none"> • B = number of days in a year, depending on the year basis. • DIM = number of days from settlement to maturity.
IPMT(rate, per, nper, pv [, fv[, type]])	<p>Returns the interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.</p> <p>Units used for specifying rate and nper should be consistent.</p>
ISPMT(rate, per, nper, pv)	<p>Calculates the interest paid (or received) for the specified period of a loan (or investment) with even principal payments.</p>

Function	Notes
MDURATION (settlement , maturity , coupon , yld , frequency [, basis])	<p>Returns the modified Macauley duration for a security with an assumed par value of \$100.</p> <p>Formula used:</p> $MDURATION = \frac{DURATION}{1 + \left(\frac{Market\ yield}{Coupon\ payments\ per\ year} \right)}$
NOMINAL (effect_rate , npery)	<p>Returns the nominal annual interest rate, given the effective rate and the number of compounding periods per year.</p> <p>The relationship between NOMINAL and EFFECT is shown in the following equation:</p> $EFFECT = \left(1 + \frac{nominal_rate}{npery} \right)^{npery} - 1$
NPER (rate , pmt , pv [, fv[, type]])	<p>Returns the number of periods for an investment based on periodic, constant payments and a constant interest rate.</p>

Function	Notes
ODDFPRICE (settlement , maturity , issue , first_coupon , rate , yld , redemption , frequency [, basis])	<p>Returns the price per \$100 face value of a security having an odd (short or long) first period.</p> <p>Formulas used:</p> <p>Odd short first coupon:</p> $\text{ODDFPRICE} = \left[\frac{\text{redemption}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(N - 1 + \frac{\text{DSC}}{E}\right)}} \right] + \left[\frac{100 \times \frac{\text{rate}}{\text{frequency}} \times \frac{\text{DFC}}{E}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(\frac{\text{DSC}}{E}\right)}} \right] +$ $\left[\sum_{k=2}^N \frac{100 \times \frac{\text{rate}}{\text{frequency}}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(k - 1 + \frac{\text{DSC}}{E}\right)}} \right] - \left[100 \times \frac{\text{rate}}{\text{frequency}} \times \frac{A}{E} \right]$ <p>where:</p> <ul style="list-style-type: none"> A = number of days from the beginning of the coupon period to the settlement date (accrued days). DSC = number of days from the settlement to the next coupon date. DFC = number of days from the beginning of the odd first coupon to the first coupon date. E = number of days in the coupon period. N = number of coupons payable between the settlement date and the redemption date. (If this number contains a fraction, it is raised to the next whole number.) <p>Odd long first coupon:</p> $\text{ODDFPRICE} = \left[\frac{\text{redemption}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(N + N_q + \frac{\text{DSC}}{E}\right)}} \right] + \left[\frac{100 \times \frac{\text{rate}}{\text{frequency}} \times \left[\sum_{i=1}^{NC} \frac{\text{DC}_i}{\text{NL}_i} \right]}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(N_q + \frac{\text{DSC}}{E}\right)}} \right] +$ $\left[\sum_{k=1}^N \frac{100 \times \frac{\text{rate}}{\text{frequency}}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(k - N_q + \frac{\text{DSC}}{E}\right)}} \right] - \left[100 \times \frac{\text{rate}}{\text{frequency}} \times \sum_{i=1}^{NC} \frac{A_i}{\text{NL}_i} \right]$ <p>where:</p> <ul style="list-style-type: none"> A_i = number of days from the beginning of the ith, or last, quasi-coupon period within odd period. DC_i = number of days from dated date (or issue date) to first quasi-coupon (i = 1) or number of days in quasi-coupon (i = 2, ..., i = NC). DSC = number of days from settlement to next coupon date. E = number of days in coupon period. N = number of coupons payable between the first real coupon date and redemption date. (If this number contains a fraction, it is raised to the next whole number.) NC = number of quasi-coupon periods that fit in odd period. (If this number contains a fraction, it is raised to the next whole number.) NL_i = normal length in days of the full ith, or last, quasi-coupon period within odd period. N_q = number of whole quasi-coupon periods between settlement date and first coupon.

Function	Notes
ODDFYIELD (settlement , maturity , issue , first_coupon , rate , pr , redemption , frequency [, basis])	<p>Returns the yield of a security that has an odd (short or long) first period.</p> <p>ODDFYIELD is calculated using an iterative method.</p> <p>It uses the Newton method based on the formula used for the function ODDFPRICE.</p> <p>The yield is changed through 100 iterations until the estimated price with the given yield is close to the price.</p>
ODDLPRICE (settlement , maturity , last_interest , rate , yld , redemption , frequency [, basis])	<p>Returns the price per \$100 face value of a security having an odd (short or long) last coupon period.</p>

Function	Notes
ODDLYIELD (settlement , maturity , last_interest , rate , pr , redemption , frequency [, basis])	<p>Returns the yield of a security that has an odd (short or long) last period.</p> <p>Formula used:</p> $\text{ODDLYIELD} = \frac{\left[\text{redemption} + \left(\left(\sum_{i=1}^{\text{NC}} \frac{\text{DC}_i}{\text{NL}_i} \right) \times \frac{100 \times \text{rate}}{\text{frequency}} \right) - \left(\text{par} + \left(\left(\sum_{i=1}^{\text{NC}} \frac{\text{A}_i}{\text{NL}_i} \right) \times \frac{100 \times \text{rate}}{\text{frequency}} \right) \right) \right]}{\text{par} + \left(\left(\sum_{i=1}^{\text{NC}} \frac{\text{A}_i}{\text{NL}_i} \right) \times \frac{100 \times \text{rate}}{\text{frequency}} \right)} \times \left[\frac{\text{frequency}}{\left(\sum_{i=1}^{\text{NC}} \frac{\text{DSC}_i}{\text{NL}_i} \right)} \right]$ <p>where:</p> <ul style="list-style-type: none"> • A_i = number of accrued days for the i^{th}, or last, quasi-coupon period within odd period counting forward from last interest date before redemption. • DC_i = number of days counted in the i^{th}, or last, quasi-coupon period as delimited by the length of the actual coupon period. • NC = number of quasi-coupon periods that fit in odd period; if this number contains a fraction it will be raised to the next whole number. • NL_i = normal length in days of the i^{th}, or last, quasi-coupon period within odd coupon period.
PDURATION (rate , pv , fv)	<p>Returns the number of periods required by an investment to reach a specified value.</p> <p>Formula used:</p> $\text{PDURATION} = \frac{\log(\text{fv}) - \log(\text{pv})}{\log(1 + \text{rate})}$
PMT (rate , nper , pv [, fv[, type]])	<p>Calculates the payment for a loan based on constant payments and a constant interest rate.</p>
PPMT (rate , per , nper , pv [, fv[, type]])	<p>Returns the payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate.</p>

Function	Notes
PRICE(settlement, maturity, rate, yld, redemption, frequency [, basis])	<p>Returns the price per \$100 face value of a security that pays periodic interest.</p> <p>Formulas used:</p> <ul style="list-style-type: none"> N is the number of coupons payable between the settlement date and redemption date When N > 1 $PRICE = \left[\frac{\text{redemption}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(N - 1 + \frac{\text{DSC}}{E}\right)}} \right] + \left[\sum_{k=1}^N \frac{100 \times \frac{\text{rate}}{\text{frequency}}}{\left(1 + \frac{\text{yld}}{\text{frequency}}\right)^{\left(k - 1 + \frac{\text{DSC}}{E}\right)}} \right] - \left[100 \times \frac{\text{rate}}{\text{frequency}} \times \frac{A}{E} \right]$ <ul style="list-style-type: none"> When N = 1 $DSR = E - A$ $T1 = 100 \times \frac{\text{rate}}{\text{frequency}} + \text{redemption}$ $T2 = \frac{\text{yld}}{\text{frequency}} \times \frac{DSR}{E} + 1$ $T3 = 100 \times \frac{\text{rate}}{\text{frequency}} \times \frac{A}{E}$ $PRICE = \frac{T1}{T2} - T3$ <p>where</p> <ul style="list-style-type: none"> E = number of days in coupon period in which the settlement date falls. A = number of days from beginning of coupon period to settlement date.
PRICEDISC(settlement, maturity, discount, redemption [, basis])	<p>Returns the price per \$100 face value of a discounted security.</p> <p>Formula used:</p> $PRICEDISC = \text{redemption} - \text{discount} \times \text{redemption} \times \frac{DSM}{B}$ <p>where</p> <ul style="list-style-type: none"> B = number of days in year, depending on year basis. DSM = number of days from settlement to maturity.

Function	Notes
PRICEMAT (settlement , maturity , issue , rate , yld [, basis])	<p>Returns the price per \$100 face value of a security that pays interest at maturity.</p> <p>Formula used:</p> $\text{PRICEMAT} = \frac{100 + \left(\frac{\text{DIM}}{\text{B}} \times \text{rate} \times 100\right)}{1 + \left(\frac{\text{DSM}}{\text{B}} \times \text{yld}\right)} - \left(\frac{\text{A}}{\text{B}} \times \text{rate} \times 100\right)$ <p>where:</p> <ul style="list-style-type: none"> B = number of days in year, depending on year basis. DSM = number of days from settlement to maturity. DIM = number of days from issue to maturity. A = number of days from issue to settlement.
PV (rate , nper , pmt [, fv[, type]])	<p>Calculates the present value of a loan or an investment, based on a constant interest rate.</p> <p>Formulas used:</p> <p>If rate is not 0, then:</p> $\text{pv} \times (1 + \text{rate})^{\text{nper}} + \text{pmt}(1 + \text{rate} \times \text{type}) \times \left(\frac{(1 + \text{rate})^{\text{nper}} - 1}{\text{rate}}\right) + \text{fv} = 0$ <p>If rate is 0, then:</p> $(\text{pmt} \times \text{nper}) + \text{pv} + \text{fv} = 0$
RATE (nper , pmt , pv [, fv[, type[, guess]]])	<p>Returns the interest rate per period of an annuity.</p> <p>RATE is calculated by iteration and can have zero or more solutions.</p> <p>If the successive results of RATE do not converge to within 0.0000001 after 20 iterations, an error is returned.</p>
RECEIVED (settlement , maturity , investment , discount [, basis])	<p>Returns the amount received at maturity for a fully invested security.</p> <p>Formula used:</p> $\text{RECEIVED} = \frac{\text{investment}}{1 - (\text{discount} \times \frac{\text{DIM}}{\text{B}})}$ <p>where</p> <ul style="list-style-type: none"> B = number of days in a year, depending on the year basis. DIM = number of days from issue to maturity.

Function	Notes
RRI (nper, pv, fv)	Returns an equivalent interest rate for the growth of an investment. Formula used: $\left(\frac{fv}{pv} \right)^{\left(\frac{1}{nper} \right)} - 1$
SLN (cost, salvage, life)	Returns the straight-line depreciation of an asset for one period.
SYD (cost, salvage, life, per)	Returns the sum-of-years' digits depreciation of an asset for a specified period.
TBILLEQ (settlement, maturity, discount)	Returns the bond-equivalent yield for a Treasury bill. Formula used: $TBILLEQ = \frac{365 \times \text{discount}}{360 - (\text{discount} \times \text{DSM})}$ where: <ul style="list-style-type: none"> DSM is the number of days between settlement and maturity computed according to the 360 days per year basis.
TBILLPRICE (settlement, maturity, discount)	Returns the price per \$100 face value for a Treasury bill. Formula used: $TBILLPRICE = 100 \times \left(1 - \frac{\text{discount} \times \text{DSM}}{360} \right)$ where: <ul style="list-style-type: none"> DSM = number of days from settlement to maturity, excluding any maturity date that is more than one calendar year after the settlement date.

Function	Notes
TBILLYIELD (settlement , maturity , pr)	<p>Returns the yield for a Treasury bill.</p> <p>Formula used:</p> $TBILLYIELD = \frac{100 - pr}{pr} \times \frac{360}{DSM}$ <p>where:</p> <ul style="list-style-type: none"> DSM = number of days from settlement to maturity, excluding any maturity date that is more than one calendar year after the settlement date.
VDB (cost , salvage , life , start_period , end_period [, factor [, no_switch]])	<p>Returns the depreciation of an asset for any period you specify, including partial periods, using the double-declining balance method or some other method you specify.</p> <p>VDB stands for variable declining balance.</p> <p>The parameter factor is the rate at which the balance declines.</p> <p>If factor is omitted, it is assumed to be 2 (the double-declining balance method).</p>
XIRR (table , values , dates [, guess] [, alternateResult])	<p>Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic.</p> <p>alternateResult is a value returned in place of an error when a solution cannot be determined.</p> <p>Formula used:</p> $\sum_{j=1}^N \frac{P_j}{(1+rate)^{\frac{d_j-d_1}{365}}}$ <p>Where P_j is the j-th payment, d_j is the j-th payment date, and d_1 is the first payment date</p>
XNPV (table , values , dates , rate)	<p>Returns the present value for a schedule of cash flows that is not necessarily periodic.</p> <p>Formula used:</p> $\sum_{j=1}^N \frac{P_j}{(1+rate)^{\frac{d_j-d_1}{365}}}$ <p>Where P_j is the j-th payment, d_j is the j-th payment date, and d_1 is the first payment date</p>

Function	Notes
YIELD(settlement, maturity, rate, pr, redemption, frequency [, basis])	<p>Returns the yield on a security that pays periodic interest. Use YIELD to calculate bond yield.</p> <p>Formula used:</p> <ul style="list-style-type: none"> If there is one coupon period or less until redemption $\text{YIELD} = \frac{\left(\frac{\text{redemption}}{100} + \frac{\text{rate}}{\text{frequency}}\right) - \left(\frac{\text{par}}{100} + \left(\frac{A}{E} \times \frac{\text{rate}}{\text{frequency}}\right)\right)}{\frac{\text{par}}{100} + \left(\frac{A}{E} \times \frac{\text{rate}}{\text{frequency}}\right)} \times \frac{\text{frequency} \times E}{\text{DSR}}$ <p>where:</p> <ul style="list-style-type: none"> A = number of days from the beginning of the coupon period to the settlement date (accrued days). DSR = number of days from the settlement date to the redemption date. E = number of days in the coupon period. If there is more than one coupon period until redemption <ul style="list-style-type: none"> YIELD is calculated through a hundred iterations. The resolution uses the Newton method, based on the formula used for the function PRICE. The yield is changed until the estimated price given the yield is close to price.
YIELDDISC(settlement, maturity, pr, redemption [, basis])	<p>Returns the annual yield for a discounted security.</p>
YIELDMAT(settlement, maturity, issue, rate, pr [, basis])	<p>Returns the annual yield of a security that pays interest at maturity.</p>

DAX Information Functions

[DAX Information Functions](#) provide required information based on the given argument.

Function	Notes
CONTAINS (table , columnName, value [, columnName, value] ...)	Returns TRUE if each specified value is contained in the corresponding columnName. Otherwise, the function returns FALSE .
CONTAINSROW (table , scalar expression1 [, scalar expression2, ...])	Returns TRUE if a row of values exists or contained in a table. Otherwise returns FALSE . Note: The IN operator and the CONTAINSROW function are functionally the same. Only the syntax is different.
CUSTOMDATA()	Returns the content of the CustomData property in the connection string. Returns Blank, if CustomData property was not defined at connection time.
scalar expression IN table OR (scalar expression1, scalar expression2, ...) IN table	IN Operator: Returns TRUE if a row of values exists or contained in a table. Otherwise returns FALSE . Note: The IN operator and the CONTAINSROW function are functionally the same. Only the syntax is different.

Function	Notes
ISAFter (scalar expression , scalar expression [, sort order] [, scalar expression , scalar expression [, sort order]], ...)	Returns TRUE for a row that meets all of the condition parameters. This function takes a variable number of triples, the first two parameters in a triple are the expressions to be compared and the third is the sort order - ascending (default) or descending. Based on the sort order, the first parameter is compared with the second parameter. If the sort order is ascending, the comparison to be done is first parameter greater than or equal to second parameter. If the sort order is descending, the comparison to be done is first parameter less than or equal to second parameter. This function is similar to ISONORAFTER. The difference is ISAFter returns true for a row only after values sorted strictly after the filter values whereas ISONORAFTER returns true for values sorted on or after the filter values. This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.
ISBLANK (value)	Returns TRUE if the value is blank. Otherwise, returns FALSE .
ISEMPTY (table_expression)	Returns TRUE if the table is empty (has no rows), Returns FALSE otherwise.
ISERROR (value)	Returns TRUE if the value is an Error. Otherwise, returns FALSE .
ISEVEN (number)	Returns TRUE if number is even, Returns FALSE if number is odd. If number is nonnumeric, ISEVEN returns the #VALUE! error value.
ISINScope (column)	Returns TRUE when the specified column is the level in a hierarchy of levels.
ISLOGICAL (value)	Returns TRUE if the value is a logical value (TRUE OR FALSE). Otherwise, returns FALSE .
ISNONTEXT (value)	Returns TRUE if the value is not text or blank. (Blank cells are not text). Returns FALSE if the value is text. An empty string is considered as text.
ISNUMBER (value)	Returns TRUE if the value is numeric. Otherwise, returns FALSE .

Function	Notes
ISODD(number)	<p>Returns TRUE if number is odd.</p> <p>Returns FALSE if number is even.</p> <p>If number is nonnumeric, ISODD returns the #VALUE! error value.</p>
ISONORAFTER(scalar expression, scalar expression [, sort order] [, scalar expression, scalar expression [, sort order]], ...)	<p>This function takes a variable number of triples, the first two parameters in a triple are the expressions to be compared and the third is the sort order - ascending (default) or descending.</p> <p>Based on the sort order, the first parameter is compared with the second parameter.</p> <p>If the sort order is ascending, the comparison to be done is first parameter greater than or equal to second parameter.</p> <p>If the sort order is descending, the comparison to be done is first parameter less than or equal to second parameter.</p> <p>This function is similar to ISAFter. The difference is ISONORAFTER returns true for values sorted on or after the filter values, where ISAFter returns true for values sorted strictly after the filter values.</p>
ISSELECTEDMEASURE(M1, M2, ...)	<p>Returns TRUE if the measure that is currently being used in a calculation is one of those specified in the list of parameters.</p>
ISSUBTOTAL(column)	<p>Creates another column in a SUMMARIZE expression that returns TRUE if the row contains subtotal values for the column given as argument, otherwise returns FALSE.</p> <p>This function can only be used in the expression of a SUMMARIZE function.</p> <p>This function must be preceded by the name of the Boolean column.</p>
ISTEXT(value)	<p>Returns TRUE if the value is text</p> <p>Otherwise, returns FALSE.</p> <p>Empty string is text.</p> <p>Blank is not text.</p>

Function	Notes
LOOKUPVALUE(result_columnName, search_columnName, search_value [, search_columnName, search_value], ... [, <alternateResult>])	<p>Updated in Power BI Desktop to include alternateResult.</p> <p>Returns the value of result_column at the row where all pairs of search_column and search_value have a match.</p> <p>If only some of the criteria match, a BLANK or alternateResult (if given) is returned.</p> <p>If multiple rows match the search values and in all cases result_column values are identical, then that value is returned. Otherwise, an error or alternateResult (if given) is returned.</p>
SELECTEDMEASURE()	<p>Used by expressions for calculation items to reference the measure that is in context.</p> <p>Can only be referenced in the expression for a calculation item.</p>
SELECTEDMEASUREFORMATSTRING()	<p>Used by expressions for calculation items to retrieve the format string of the measure that is in context.</p> <p>Returns a string holding the format string of the measure that is currently in context when the calculation item is evaluated.</p> <p>This function can only be referenced in expressions for calculation items in calculation groups. It is designed to be used by the Format String Expression property of calculation items.</p>
SELECTEDMEASURENAME()	<p>Used by expressions for calculation items to determine the measure that is in context by name.</p> <p>Returns a string value holding the name of the measure that is currently in context when the calculation item is evaluated.</p> <p>Can only be referenced in the expression for a calculation item.</p> <p>This function is often used for debugging purposes when authoring calculation groups.</p>
USERCULTURE()	<p>Returns the locale for the current user as a string.</p> <p>Locale is determined by the operating system, browser settings, or Power BI service.</p> <p>Note: This function is currently supported in Power BI Premium per capacity, Power BI Premium per user, and Power BI Embedded only.</p>
USERNAME()	<p>Returns the username from the credentials given to the system at connection time. If this is a Power BI Desktop or Excel file it will be in the format Domain\Username. If it is in the Power BI Service, it will be in the format Username@domain.com</p>

Function	Notes
USEROBJECTID()	Returns the current user's Object ID from Azure AD for Power BI or Azure Analysis Services models or SID for SQL Server Analysis Services models.
USERPRINCIPALNAME()	Returns the user principal name at connection time.

DAX Logical Functions

[DAX Logical Functions](#) return values based on the conditional results.

Function	Notes
AND (logical_value, logical_value)	Checks whether both arguments are TRUE , and returns TRUE if both arguments are TRUE. Otherwise returns FALSE . If you have more than two arguments, use && (double ampersand) as an alternative and you can have as many "&&" as you like.
BITAND (number, number)	Returns a bitwise AND of two numbers. number is any scalar expression that returns number. If not an integer, it is truncated. Supports both positive and negative numbers.
BITLSHIFT (number, shift_amount)	Returns a number (integer) shifted left by the specified number of bits. Both number, shift_amount can be any DAX expression that returns an integer expression. If shift_amount is negative, it will shift in the opposite direction. If absolute value of shift_amount is larger than 64, there will be no error but will result in overflow/underflow.
BITOR (number, number)	Returns a bitwise OR of two numbers. number is any scalar expression that returns number. If not an integer, it is truncated. Supports both positive and negative numbers.
BITRSHIFT (number, shift_amount)	Returns a number (integer) shifted right by the specified number of bits. Both number, shift_amount can be any DAX expression that returns an integer expression. If shift_amount is negative, it will shift in the opposite direction. If absolute value of shift_amount is larger than 64, there will be no error but will result in overflow/underflow.
BITXOR (number, number)	Returns a bitwise XOR of two numbers. number is any scalar expression that returns number. If not an integer, it is truncated. Supports both positive and negative numbers.

Function	Notes
COALESCE (expression, expression [, expression] ...)	<p>Returns the scalar value coming from the first expression that does not evaluate to BLANK. If all expressions evaluate to BLANK, BLANK is returned.</p> <p>Input expressions may be of different data types that return a scalar expression.</p> <p>COALESCE can be used to convert BLANK values of totals to 0.</p> <p>COALESCE simplifies the code that otherwise requires an IF condition. i.e. instead of writing <code>IF(ISBLANK(Result), 0, Result)</code>, we can write <code>COALESCE(Result, 0)</code></p>
FALSE ()	Returns the logical value FALSE .
IF (logical test, value_if_true [, value_if_false])	<p>Checks if the condition provided as the first argument is met. Returns one value if the condition is TRUE and returns another value if the condition is FALSE.</p> <p>Returns blank, if the condition is FALSE and value_if_false is omitted.</p>
IF.EAGER (logical test, value_if_true [, value_if_false])	<p>Evaluates both the expressions value_if_true and value_if_false.</p> <p>Then checks if the condition provided as the first argument (logical_test) is met. Returns value_if_true if the condition is TRUE and returns value_if_false if the condition is FALSE.</p> <p>The difference between IF and IF.Eager is that</p> <ul style="list-style-type: none"> • IF evaluates the condition first and then selects the branch expression to evaluate based on the result of the condition. • IF-EAGER evaluates both branch expressions first and then evaluates the condition. The return value is based on the condition.
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.
NOT (logical_value)	Changes FALSE to TRUE , or TRUE to FALSE .
OR (logical_value, logical_value)	<p>Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.</p> <p>If you have more than two arguments, use <code> </code> (double pipe) as an alternative and you can have as many <code>" "</code> as you like.</p>

Function	Notes
SWITCH(Expression, value1, expression1 [, value2, expression2] ... [, else, expression])	<p>Expression is evaluated and the result is matched with the given values. If a match is found, the corresponding expression is evaluated.</p> <p>If the result is not matched with any of the given values, and else is given, the corresponding expression is evaluated.</p> <p>All expressions must be of the same data type.</p>
TRUE()	Returns the logical value TRUE .

DAX Math and Trig Functions

[DAX Math and Trig Functions](#) are similar to Excel mathematical and trigonometric functions.

Function	Notes
ABS(number)	Removes the negative sign if it exists.
ACOS(number)	Returns the arccosine, or inverse cosine, of a number. The arccosine is the angle whose cosine is number. The returned angle is given in radians in the range 0 (zero) to pi.
ACOSH(number)	Returns the inverse hyperbolic cosine of a number. The number must be greater than or equal to 1. The inverse hyperbolic cosine is the value whose hyperbolic cosine is <i>number</i> , so ACOSH(COSH(number)) equals <i>number</i> .
ACOT(number)	Returns the arccotangent, or inverse cotangent of a number. The arccotangent is the angle whose cotangent is number.
ACOTH(number)	Returns the inverse hyperbolic cotangent of a number. The absolute value of Number must be greater than 1.
ASIN(number)	Returns the arcsine, or inverse sine, of a number. The arcsine is the angle whose sine is number. The returned angle is given in radians in the range -pi/2 to pi/2.
ASINH(number)	Returns the inverse hyperbolic sine of a number. The inverse hyperbolic sine is the value whose hyperbolic sine is number, so ASINH(SINH(number)) equals number.
ATAN(number)	Returns the arctangent, or inverse tangent, of a number. The arctangent is the angle whose tangent is number. The returned angle is given in radians in the range -pi/2 to pi/2.
ATANH(number)	Returns the inverse hyperbolic tangent of a number. Number must be between -1 and 1 (excluding -1 and 1). The inverse hyperbolic tangent is the value whose hyperbolic tangent is number, so ATANH(TANH(number)) equals number.
CEILING(number, significance)	Rounds a number up, to the nearest integer or to the nearest multiple of significance.
COMBIN(number, number_chosen)	<p>Returns the number of combinations for a given number of items.</p> <p>Numeric arguments are truncated to integers.</p> <p>If either argument is nonnumeric, COMBIN returns the #VALUE! error value.</p> <p>If $\text{number} < 0$, $\text{number_chosen} < 0$, or $\text{number} < \text{number_chosen}$, COMBIN returns the #NUM! error value.</p>

Function	Notes
COMBINA (number, number_chosen)	<p>Returns the number of combinations (with repetitions) for a given number of items.</p> <ul style="list-style-type: none"> number must be greater than or equal to 0, and greater than or equal to number_chosen. number_chosen must be greater than or equal to 0. <p>If the value of either argument is outside of it's constraints, COMBINA returns the #NUM! error value.</p> <p>If either argument is a non-numeric value, COMBINA returns the #VALUE! error value.</p> <p>Non-integer values are truncated.</p>
COS (number)	Returns the cosine of the given angle.
COSH (number)	Returns the hyperbolic cosine of a number.
COT (number)	Returns the cotangent of an angle specified in radians.
COTH (number)	Returns the hyperbolic cotangent of a number.
CURRENCY (value)	The value of the expression evaluated and returned as a currency type value.
DEGREES (angle)	Converts radians into degrees.
DIVIDE (numerator, denominator [, alternate-result])	<p>Safe divide function that gracefully handles a divide by zero error.</p> <p>Performs division and returns alternate result or BLANK() on division by 0.</p>
EVEN (number)	Returns number rounded up to the nearest even integer.
EXP (number)	Returns e raised to the power of a given number. The constant e equals 2.71828182845904, the base of the natural logarithm.
FACT (number)	<p>Returns the factorial of a number, equal to</p> <p>$1*2*3*...number..$</p> <p>number should be a non-negative integer.</p>
FLOOR (number, significance)	Rounds a number down, toward zero, to the nearest multiple of significance.

Function	Notes
GCD (number1 [number2] ...)	Returns the greatest common divisor of two or more integers. The greatest common divisor is the largest integer that divides both number1 and number2 without a remainder. If only one number is given, the function returns the number itself.
INT (number)	Rounds a number down to the nearest integer.
ISO.CEILING (number [significance])	Rounds number up, to the nearest multiple of significance. Rounds number up, to the nearest integer if significance is omitted.
LCM (number1 [number2] ...)	Returns the least common multiple of integers. The least common multiple is the smallest positive integer that is a multiple of all integer arguments number1, number2, and so on.
LN (number)	Returns the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).
LOG (number , [base])	Returns the logarithm of a number to the base you specify. If base is omitted, it is taken as 10. You might receive an error if the value is too large to be displayed.
LOG10 (number)	Returns the base-10 logarithm of a number.
MOD (number , divisor)	Returns the remainder after a number is divided by a divisor. The result always has the same sign as the divisor. If the divisor is 0 (zero), MOD returns an error. You cannot divide by 0.
MROUND (number , multiple)	Returns a number rounded to the desired multiple.
ODD (number)	Returns number rounded up to the nearest odd integer.

Function	Notes
PERMUT(number, number_chosen)	Returns the number of permutations for a given number of objects that can be selected from number objects. A permutation is any set or subset of objects or events where internal order is significant.
PI()	Returns the value of Pi, 3.14159265358979, accurate to 15 digits.
POWER(number, power)	Returns the result of a number raised to a power.
QUOTIENT(numerator, denominator)	Performs division and returns only the integer portion of the division result. Use this function when you want to discard the remainder of division.
RADIANS(angle)	Converts degrees to radians.
RAND()	Returns a random number greater than or equal to 0 and less than 1, evenly distributed. The number that is returned changes each time the cell containing this function is recalculated.
RANDBETWEEN(bottom, top)	Returns a random number in the range between two numbers you specify.
ROUND(number, num_digits)	Rounds a number to the specified number of digits.
ROUNDDOWN(number, num_digits)	Rounds a number down, toward zero.

Function	Notes
ROUNDUP (number, num_digits)	Rounds a number up, away from 0 (zero).
SIGN (number)	Determines the sign of a number, the result of a calculation, or a value in a column. The function returns 1 if the number is positive, 0 (zero) if the number is zero, or -1 if the number is negative.
SIN (number)	Returns the sine of the given angle.
SINH (number)	Returns the hyperbolic sine of a number.
SQRT (number)	Returns the square root of a number. If the number is negative, the SQRT function returns an error.
SQRTPI (number)	Returns the square root of (number * pi).
TAN (number)	Returns the tangent of the given angle.
TANH (number)	Returns the hyperbolic tangent of a number.
TRUNC (number, num_digits)	Truncates a number to an integer by removing the decimal, or fractional, part of the number. num_digits specifies the precision of the truncation; if omitted, 0 (zero).

DAX Other Functions

[These functions](#) perform unique actions that cannot be defined by any of the categories.

Function	Notes
CONVERT (Expression, Datatype)	<p>Converts an expression of one data type to another.</p> <p>Datatype can be any of the following.</p> <ul style="list-style-type: none"> • INTEGER (Whole Number) • DOUBLE (Decimal Number) • STRING (Text) • BOOLEAN (True/False) • CURRENCY (Fixed Decimal Number) • DATETIME (Date, Time, etc.)
DETAILROWS (Measure)	<p>A table with the data returned by the Detail Rows Expression.</p> <p>If no Detail Rows Expression is defined, the data for the table containing the measure is returned.</p>
ERROR (text)	Raises an error with an error message.
EVALUATEANDLOG (Value, [Label], [MaxRows])	<p>Returns the value of the first argument and logs it in a DAX Evaluation Log profiler event.</p> <p>This function is fully functional in Power BI Desktop only.</p> <p>It acts as a simple passthrough function in other environments.</p>

Function	Notes
VAR VarName = Expression	<p>Stores the result of an expression as a named variable, which can then be passed as an argument to other measure expressions. Once resultant values have been calculated for a variable expression, those values do not change, even if the variable is referenced in another expression.</p> <p>VarName is the name of the variable (identifier).</p> <ul style="list-style-type: none"> Supported character set: a-z, A-Z, 0-9. 0-9 are not valid as first character. __ (double underscore) is allowed as a prefix. No other special characters are supported. Delimiters are not supported. For example, 'VarName' or [VarName] will result in an error. Reserved keywords not allowed. Names of existing tables are not allowed. Empty spaces are not allowed. <p>Expression is a DAX expression which returns a scalar or table value.</p> <ul style="list-style-type: none"> Expression can contain another VAR declaration. <p>When referencing a variable:</p> <ul style="list-style-type: none"> Measures cannot refer to variables defined outside the measure expression, but can refer to functional scope variables defined within the expression. Variables can refer to measures. Variables can refer to previously defined variables. <p>Columns in table variables cannot be referenced via TableName[ColumnName] syntax.</p>

DAX Other Special Functions (X-Functions / Iterators)

These functions perform specific actions that complement the other DAX functions.

DAX Iterator Functions, called **iterators** for short, take a column or a table as the argument and aggregate the values just as aggregation functions—but using a different approach. These functions aggregate the values in a row context.

These are “**X-functions**” (i.e., any function that has an X on the end of the name). The iterators given below also include statistical iterator functions.

DAX also has **two financial functions** that got added in Excel 2016.

Function	Notes
AVERAGEX (table , expression)	Calculates the average (arithmetic mean) of a set of expressions evaluated over a table.
CONCATENATEX (table , expression [, delimiter])	Concatenates the result of an expression evaluated for each row in a table.
COUNTAX (table , expression)	The COUNTAX function counts nonblank results when evaluating the result of an expression over a table. That is, it works just like the COUNTA function, but is used to iterate through the rows in a table and count rows where the specified expressions result in a nonblank result.
COUNTX (table , expression)	Counts the number of rows that contain a number or an expression that evaluates to a number, when evaluating an expression over a table.
GEOMEANX (table , expression)	Returns the geometric mean of an expression evaluated for each row in a table.
MAXX (table , expression)	Returns the largest numeric value that results from evaluating an expression for each row of a table. <ul style="list-style-type: none"> • The first argument can be a table name, or an expression that evaluates to a table. • The second argument indicates the expression to be evaluated for each row of the table.

Function	Notes
MEDIANX (table , expression)	Returns the median of an expression evaluated for each row in a table.
MINX (table , expression)	Returns the smallest numeric value that results from evaluating an expression for each row of a table. <ul style="list-style-type: none"> • The first argument can be a table name, or an expression that evaluates to a table. • The second argument indicates the expression to be evaluated for each row of the table.
PERCENTILEX.EXC (table , expression , k)	Returns the percentile number of an expression evaluated for each row in a table.
PERCENTILEX.INC (table , expression , k)	Returns the percentile number of an expression evaluated for each row in a table.
PRODUCTX (table , expression)	Returns the product of an expression evaluated for each row in a table.
RANKX (table , expression , [value] , [order] , [ties])	Returns the ranking of a number in a list of numbers for each row in the table argument.

Function	Notes
STDEVX.P (table , expression)	Returns the standard deviation of the entire population. expression is any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).
STDEVX.S (table , expression)	Returns the standard deviation of a sample population.
VARX.S (table , expression)	Returns the variance of a sample population.

DAX Other Special Functions (Argument Functions)

In DAX, there are also some special functions that have a very specific purpose of usability in other DAX functions only. i.e. they can be used only as arguments to certain DAX functions. These functions cannot be used as standalone functions.

Function	Notes
CURRENTGROUP()	<p>CURRENTGROUP can only be used in an expression that defines a column within the GROUPBY function.</p> <p>In-effect, CURRENTGROUP returns a set of rows from the “table” argument of GROUPBY that belong to the current row of the GROUPBY result.</p> <p>The CURRENTGROUP function takes no arguments and is only supported as the first argument to one of the following aggregation functions: AverageX, CountAX, CountX, GeoMeanX, MaxX, MinX, ProductX, StDevX.S, StDevX.P, SumX, VarX.S, VarX.P.</p>
IGNORE(expression)	<p>IGNORE function does not return a value.</p> <p>IGNORE can be used as an expression argument to SUMMARIZECOLUMNS function.</p>
NONVISUAL(expression)	<p>Returns a table of values and used with SUMMARIZECOLUMNS.</p> <p>Marks a value filter in SUMMARIZECOLUMNS function as not affecting measure values, but only applying to group-by columns.</p>
ORDERBY(orderBy_columnName [, order] [, orderBy_columnName [, order]] [, ...])	<p>Defines the columns that determine the sort order within each of a window function’s partitions.</p> <p>This function does not return a value.</p> <p>This function can only be used within a window function expression.</p>
PARTITIONBY(partitionBy_columnName [, partitionBy_columnName [, ...]])	<p>Defines the columns that are used to partition a Window function’s relation parameter.</p> <p>This function does not return a value.</p> <p>This function can only be used within a window function expression.</p>
ROLLUP(groupBy_columnName [, groupBy_columnName [, ...]])	<p>Modifies the behavior of the SUMMARIZE function by adding rollup rows to the result on columns defined by the groupBy_columnName parameter.</p> <p>This function can only be used within a SUMMARIZE expression.</p>

Function	Notes
ROLLUPADDISSUBTOTAL (groupBy_columnName, isSubtotal_columnName, [groupBy_columnName, isSubtotal_columnName], ...)	<p>ROLLUPADDISSUBTOTAL function does not return a value. It only specifies the set of columns to be subtotalled.</p> <p>ROLLUPADDISSUBTOTAL is used with SUMMARIZECOLUMNS function.</p>
ROLLUPGROUP (groupBy_columnName, groupBy_columnName)	<p>ROLLUPGROUP can only be used as a groupBy_columnName argument to the ROLLUPADDISSUBTOTAL and / or the SUMMARIZE functions.</p> <p>ROLLUPGROUP is used inside the ROLLUPISSUBTOTAL function to reflect ROLLUPGROUPs present in the supplied table argument.</p>
ROLLUPISSUBTOTAL (groupingColumn, isSubtotal_columnName, [groupingColumn, isSubtotal_columnName], ...)	<p>ROLLUPISSUBTOTAL is used to define the supplied table argument to the ADDMISSINGITEMS function.</p> <p>The ADDMISSINGITEMS function requires that, if ROLLUPISSUBTOTAL was used to define the supplied table argument, ISSUBTOTAL columns corresponding to each group by column, or ROLLUPGROUP, are present in the supplied table argument. Also, the names of the ISSUBTOTAL columns must be supplied in the ROLLUPISSUBTOTAL function inside ADDMISSINGITEMS and they must match names of Boolean columns in the supplied table argument. This enables the ADDMISSINGITEMS function to identify BLANK values stemming from the fact that a row is a subtotal row from other BLANK values.</p>

DAX Parent and Child Functions

[DAX Parent and Child functions](#) help to manage data that is presented as a parent/child hierarchy in the data model.

For more information read. [Understanding Functions for Parent-Child Hierarchies in DAX.](#)

Function	Notes
PATH (ID_columnName, parent_columnName)	Returns a delimited text string with the identifiers of all the parents of the current identifier, starting with the oldest and continuing until current.
PATHCONTAINS (path, item)	Returns TRUE if the specified <i>item</i> exists within the specified <i>path</i> .
PATHITEM (path, position, [type])	Returns the item at the specified <i>position</i> from a string resulting from evaluation of a PATH function. Positions are counted from left to right.
PATHITEMREVERSE (path, position, [type])	Returns the item at the specified <i>position</i> from a string resulting from evaluation of a PATH function. Positions are counted backwards from right to left.
PATHLENGTH (path)	Returns the number of parents to the specified item in a given PATH result, including self.

DAX Query Functions

These DAX functions are helpful in writing queries in DAX. These functions return tables that can be used in other DAX functions as input. The resulting tables are virtual and are not materialised.

DAX Studio is a great way to learn about table functions because you can “see” the tables materialised on the screen. Read about DAX Studio as a tool to query your data model here. <http://exceleratorbi.com.au/getting-started-dax-studio/>.

Function	Notes
ADDCOLUMNS (table , name, expression [name, expression] ...)	Adds calculated columns to the given table or table expression.
CROSSJOIN (table , table [table] ...)	Returns a table that contains the Cartesian product of all rows from all tables in the arguments. The columns in the new table are all the columns in all the argument tables.
DATATABLE (ColumnName1 , DataType1 , ColumnName2 , DataType2 , ..., {{ Value1 , Value2 ...}}, { Value1 , Value2 ...}, ... })	Returns a table declaring an inline set of values. Each of the columns is given a name and the data type of the column is provided. Then, the set of values is given – {{Row 1 values}, {Row 2 values}, ... }
EXCEPT (table_expression1 , table_expression2)	Returns a table that contains the rows of one table minus all the rows of another table.

Function	Notes
GENERATE (table1 , table2)	Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.
GENERATEALL (table1 , table2)	Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.
GROUPBY (table , [<groupBy_columnName1>], [<name, expression >] ...)	<p>Returns a table with the selected columns for the groupBy_columnName arguments and the grouped by columns designated by the name arguments.</p> <p>The GROUPBY function is similar to the SUMMARIZE function. However, GROUPBY does not do an implicit CALCULATE and the group isn't placed into the filter context.</p>
GENERATESERIES (StartValue , EndValue [, IncrementValue])	Generates a table of values using the parameters provided
INTERSECT (table, table)	Returns the row intersection of two tables, retaining duplicates.
NATURALINNERJOIN (leftJoinTable , rightJoinTable)	<p>Performs an inner join of a table with another table.</p> <p>The tables are joined on common columns (by name) in the two tables. If the two tables have no common column names, an error is returned.</p>
NATURALLEFTOUTERJOIN (leftJoinTable , rightJoinTable)	<p>Performs an inner join of a table with another table.</p> <p>The tables are joined on common columns (by name) in the two tables. If the two tables have no common column names, an error is returned.</p>

Function	Notes
ROW (name, expression [, name, expression] ...)	Returns a table with a single row containing values that result from the expressions given to each column.
SELECTCOLUMNS (table, name, scalar_expression [, name, scalar_expression] ...)	Adds calculated columns to the given table or table expression. SELECTCOLUMNS starts with an empty table and that is the only difference between ADDCOLUMNS and SELECTCOLUMNS .
SUMMARIZE (table, groupBy_columnName [, groupBy_columnName] ..., name, expression, [name, expression] ...)	Returns a table of values for use in a query or inside a formula that uses a table. If there are relationships between tables, always specify the table on the many side of the relationship as the table parameter. This function is semantically similar to “group by” in SQL.
SUMMARIZECOLUMNS (groupBy_columnName [, groupBy_columnName] ..., [filterTable] ... [, name, expression] ...)	Returns a summary table over a set of groups. A table which includes combinations of values from the supplied columns, based on the grouping specified. Only rows for which at least one of the supplied expressions return a non-blank value are included in the table returned. If all expressions evaluate to BLANK/NULL for a row, that row is not included in the table returned. A column cannot be specified more than once in the groupBy_columnName.

Function	Notes
Table Constructor <pre>{scalar_expression1, scalar_expression2, ... }</pre> <pre>{{(scalar_expression11, scalar_expression12, ..., scalar_expression1N), (scalar_expression21), scalar_expression22, ..., scalar_expression2N), ... }</pre>	<p>Returns a table of one or more columns.</p> <ul style="list-style-type: none"> When there is only one column, the name of the column is Value. When there are N columns where $N > 1$, the names of the columns from left to right are Value1, Value2, ..., ValueN. <p>Note:</p> <ul style="list-style-type: none"> The number of scalar expressions must be the same for all rows. When the data types of the values for a column are different in different rows, all values are converted to a common data type.
UNION (table_expression1, table_expression2)	<p>Creates a union (join) table from a pair of tables.</p> <p>Returns a table that contains all the rows from each of the two table expressions.</p> <p>The two tables must have the same number of columns.</p> <p>Columns are combined by position in their respective tables.</p> <p>The column names in the return table will match the column names in table_expression1.</p> <p>Duplicate rows are retained.</p> <p>The returned table has lineage where possible. When data types differ, the resulting data type is determined based on the rules for data type coercion.</p> <p>The returned table will not contain columns from related tables.</p>

DAX Statistical Functions

Following are the [DAX Statistical Functions](#):

Function	Notes
BETA.DIST(x, alpha, beta, cumulative [, A] [, B])	<p>Returns the beta distribution. The beta distribution is commonly used to study variation in the percentage of something across samples, such as the fraction of the day people spend watching television.</p>
BETA.INV(probability, alpha, beta [, A] [, B])	<p>Returns the inverse of the beta cumulative probability density function (BETA.DIST).</p> <p>If probability = BETA.DIST(x, ...TRUE), then BETA.INV(probability, ...) = x.</p>
CHISQ.DIST(x, deg_freedom, cumulative)	<p>Returns the chi-squared distribution.</p> <p>The chi-squared distribution is commonly used to study variation in the percentage of something across samples.</p> <p>If $x < 0$, an error is returned.</p> <p>If x or $deg_freedom$ is nonnumeric, an error is returned.</p> <p>If $deg_freedom$ is not an integer, it is rounded. < 1 or $> 10^{10}$, an error is returned.</p> <p>This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.</p>

Function	Notes
CHISQ.DIST.RT (x , deg_freedom)	<p>Returns the right-tailed probability of the chi-squared distribution.</p> <p>The chi-squared distribution is associated with a chi-squared test. Use the chi-squared test to compare observed and expected values.</p> <p>If $x < 0$, an error is returned.</p> <p>If x or $deg_freedom$ is nonnumeric, an error is returned.</p> <p>If $deg_freedom$ is not an integer, it is rounded. < 1 or $> 10^{10}$, an error is returned.</p> <p>This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.</p>
CHISQ.INV (probability , deg_freedom)	Returns the inverse of the left-tailed probability of the chi-squared distribution.
CHISQ.INV.RT (probability , deg_freedom)	<p>Returns the inverse of the right-tailed probability of the chi-squared distribution.</p> <p>If $probability = CHISQ.DIST.RT(x, \dots)$, then $CHISQ.INV.RT(probability, \dots) = x$.</p> <p>Use this function to compare observed results with expected ones in order to decide whether your original hypothesis is valid.</p>
CONFIDENCE.NORM (alpha , standard_dev , size)	<p>The confidence interval is a range of values.</p> <p>Your sample mean, x, is at the center of this range and the range is $x \pm CONFIDENCE.NORM$.</p>
CONFIDENCE.T (alpha , standard_dev , size)	Returns the confidence interval for a population mean, using a Student's t distribution.

Function	Notes
EXPON.DIST (x, lambda, cumulative)	Returns the exponential distribution. Use EXPON.DIST to model the time between events.
GEOMEAN (column)	Returns the geometric mean of the numbers in a column.
MEDIAN (column)	Returns the median of numbers in a column.
NORM.DIST (X, Mean, Standard_dev, Cumulative)	Returns the normal distribution for the specified mean and standard deviation.
NORM.INV (Probability, Mean, Standard_dev)	Returns the inverse of the normal cumulative distribution for the specified mean and standard deviation.
NORM.S.DIST (Z, Cumulative)	Returns the standard normal distribution (has a mean of zero and a standard deviation of one).
NORM.S.INV (Probability)	Returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.
PERCENTILE.EXC (column, k)	Returns the k-th percentile of values in a range, where k is in the range 0..1, exclusive.
PERCENTILE.INC (column, k)	Returns the k-th percentile of values in a range, where k is in the range 0..1, inclusive.
POISSON.DIST (x, mean, cumulative)	Returns the Poisson distribution. A common application of the Poisson distribution is predicting the number of events over a specific time.
RANK.EQ (value, columnName, [order])	Returns the ranking of a number in a list of numbers.

Function	Notes
SAMPLE (n_value, table, orderBy_expression, [order] [, orderBy_expression, [order], ...])	Returns a sample of N rows from the specified table.
STDEV.P (ColumnName)	Returns the standard deviation of the entire population.
STDEV.S (ColumnName)	Returns the standard deviation of a sample population.
T.DIST (X,Deg_freedom,Cumulative)	Returns the Student's left-tailed t-distribution.
T.DIST.2T (X,Deg_freedom)	Returns the two-tailed Student's t-distribution.
T.DIST.RT (X,Deg_freedom)	Returns the right-tailed Student's t-distribution.
T.INV (Probability,Deg_freedom)	Returns the left-tailed inverse of the Student's t-distribution.
T.INV.2T (Probability,Deg_freedom)	Returns the two-tailed inverse of the Student's t-distribution.
VAR.P (ColumnName)	Returns the variance of the entire population.
VAR.S (ColumnName)	Returns the variance of a sample population.

DAX Text Functions

[DAX Text Functions](#) are based on the Excel string functions, but have been modified to work with tables and columns.

Function	Notes
BLANK()	Returns a blank. Blanks are not equivalent to nulls. DAX uses blanks for both database nulls and for blank cells in Excel.
CODE(text)	Returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer.
COMBINEVALUES(delimiter, expression, expression [,expression] ...)	Returns the concatenated string where the values of the DAX expressions are joined by the delimiter (which is a constant).
CONCATENATE(text1, text2)	Joins two text strings into one text string. If you need to add more than two arguments, use the AND (&) operator.
CONTAINSSTRING(within_text, find_text)	Returns TRUE or FALSE indicating whether one string contains another string. <ul style="list-style-type: none"> CONTAINSSTRING is not case-sensitive. You can use ? and * wildcard characters. Use ~ to escape wildcard characters.
CONTAINSSTRINGEXACT(within_text, find_text)	Returns TRUE or FALSE indicating whether one string contains another string. <ul style="list-style-type: none"> CONTAINSSTRING is case-sensitive.
EXACT(text1, text2)	Compares two text strings and returns TRUE if they are exactly the same, otherwise returns FALSE . EXACT is case-sensitive.

Function	Notes
FIND (find_text , within_text [, start_num] [, NotFoundValue])	Returns the starting position of one text string within another text string. FIND is case-sensitive.
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.
FORMAT (value , format_string [, < locale_name >])	Converts a value to text according to the specified format. locale_name is optional and is the name of the locale to be used by the function. Possible values are strings accepted by the Windows API function LocaleNameToLCID() .
LEFT (text [, num_chars])	Returns the specified number of characters from the start of a text string.
LEN (text)	Returns the number of characters in a text string.
LOWER (text)	Converts all letters in a text string to lowercase.
MID (text , start_num , num_chars)	Returns a string of characters from the middle of a text string, given a starting position and length.

Function	Notes
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.
REPT (text, num_times)	Repeats text a given number of times. Use REPT to fill a cell with a number of instances of a text string.
RIGHT (text, [num_chars])	RIGHT returns the last character or characters in a text string, based on the number of characters you specify.
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 function is case insensitive and accent sensitive.
SUBSTITUTE (text, old_text, new_text, instance_num)	Replaces existing text with new text in a text string.
TOCSV (Table, [MaxRows], [Delimiter], [IncludeHeaders])	Returns a table as a string in CSV format. MaxRows is the maximum number of rows to convert. If omitted, it is 10 rows. Delimiter is a column delimiter. If omitted, it is comma ",". IncludeHeaders specifies a header with column name as the first row. If omitted, it is True.

Function	Notes
TOJSON (Table, [MaxRows])	Returns a table as a string using JSON format. MaxRows is the maximum number of rows to convert. If omitted, it is 10 rows.
TRIM (text)	Removes all spaces from text except for single spaces between words.
UNICHAR (number)	Returns the Unicode character referenced by the numeric value.
UNICODE (text)	Returns the Unicode number of the first character of the text.
UPPER (text)	Converts a text string to all uppercase letters
VALUE (text)	<p>You need not use the VALUE function in a formula because Power Pivot implicitly converts text to numbers.</p> <p>The argument text can be a constant or in one of the formats - number, date or time format. Otherwise, an error is returned.</p> <p>You can use a column reference as argument to VALUE function. E.g., if you have a column that contains mixed number types, VALUE can be used to convert all values to a single numeric data type. However, if you use the VALUE function with a column that contains mixed numbers and text, the entire column is flagged with an error, because not all values in all rows can be converted to numbers.</p>

DAX Time Intelligence Functions

[DAX Time Intelligence Functions](#) enable you to manipulate data using time periods, including days, months, quarters and years, and then build and compare calculations over those periods.

There are 3 classes of DAX Time Intelligence functions based on their return value.

- Those that return scalar values (e.g. TOTALYTD). They are stand-alone functions and so can be used as a stand-alone measure/function.
- Those that return a table containing a single column and single row with a date value (e.g. FIRSTNONBLANK). They can be used either as stand-alone functions to define a measure (as a scalar value) or as an argument to any function that requires a table in its arguments. These types of functions therefore behave as both a scalar value and a table, depending on the use case.
- Those that return a table of dates (e.g. DATESYTD) and are designed to be used inside a CALCULATE function.

DAX Time Intelligence Functions that return Scalar Values

Function	Notes
CLOSINGBALANCEMONTH (expression , dates , [filter])	Returns a scalar value that represents the expression evaluated at the last date of the month in the current context. E.g. Use to create a measure that calculates the 'Month End Inventory Value' of the product inventory.
CLOSINGBALANCEQUARTER (expression , dates , [filter])	Returns a scalar value that represents the expression evaluated at the last date of the month in the current context. E.g. Use to create a measure that calculates the 'Quarter End Inventory Value' of the product inventory.
CLOSINGBALANCEYEAR (expression , dates , [filter] , [year_end_date])	Returns a scalar value that represents the expression at the last date of the year in the current context. E.g. Use to create a measure that calculates the ' Year End Inventory Value ' of the product inventory.
OPENINGBALANCEMONTH (expression , dates [, filter])	Returns a scalar value that represents the expression evaluated at the first date of the month in the current context. E.g. You can create a measure that calculates the ' Month Start Inventory Value ' of the product inventory.

Function	Notes
OPENINGBALANCEQUARTER (expression , dates [filter])	Returns a scalar value that represents the expression evaluated at the first date of the quarter in the current context. E.g. You can create a measure that calculates the ' Quarter Start Inventory Value ' of the product inventory.
OPENINGBALANCEYEAR (expression , dates [filter])	Returns a scalar value that represents the expression evaluated at the first date of the year in the current context. E.g. You can create a measure that calculates the ' Year Start Inventory Value ' of the product inventory.
TOTALMTD (expression , dates [filter])	Returns a scalar value that represents the expression evaluated for the dates in the current month-to-date, given the dates in dates .
TOTALQTD (expression , dates [filter])	Returns a scalar value that represents the expression evaluated for the dates in the current quarter-to-date, given the dates in dates .
TOTALYTD (expression , dates [filter] [year_end_date])	Returns a scalar value that represents the expression evaluated for the dates in the current year-to-date, given the dates in dates .

DAX Time Intelligence Functions that return both a Table and a Scalar

These functions return a table containing a single column and single row with a date value (can be used as a scalar value or a table input to another function).

Function	Notes
ENDOFMONTH(dates)	<p>Returns a table containing a single column and single row with the last date of the month in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p>
ENDOFQUARTER(dates)	<p>Returns a table containing a single column and single row with the last date of the quarter in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p>
ENDOFYEAR(dates [, year_end_date])	<p>Returns a table containing a single column and single row with the last date of the year in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p>
FIRSTDATE(dates)	<p>Returns a table containing a single column and single row with the first date in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p> <p>When the current context is a single date, the date returned by the FIRSTDATE and LASTDATE functions will be equal.</p>
FIRSTNONBLANK(column, expression)	<p>Returns a table containing a single column and single row with the first value in the column filtered by the current context, where the expression is not blank.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p> <p>E.g. You could get the first value for which there were sales of a product.</p>
FIRSTNONBLANKVALUE(column, expression)	<p>Evaluates an expression filtered by the sorted values of a column and returns the first value of the expression that is not blank.</p> <p>This function is different from FIRSTNONBLANK in that the column is added to the filter context for the evaluation of expression.</p>

Function	Notes
)	
LASTDATE(dates)	<p>Returns a table containing a single column and single row with the last date in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p> <p>When the current context is a single date, the date returned by the FIRSTDATE and LASTDATE functions will be equal.</p>
LASTNONBLANK(column, expression)	<p>Returns a table containing a single column and single row with the last value in the column, filtered by the current context, where the expression is not blank.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p> <p>E.g. You could get the last value for which there were sales of a product.</p>
LASTNONBLANKVALUE(column, expression)	<p>Evaluates an expression filtered by the sorted values of a column and returns the last value of the expression that is not blank.</p> <p>This function is different from LASTNONBLANK in that the column is added to the filter context for the evaluation of expression.</p>
STARTOFMONTH(dates)	<p>Returns a table containing a single column and single row with the first date of the month in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p>
STARTOFQUARTER(dates)	<p>Returns a table containing a single column and single row with the first date of the quarter in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p>
STARTOFYEAR(Dates, [year end date])	<p>Returns a table containing a single column and single row with the first date of the year in the current context for the specified column of dates.</p> <p>Therefore, this function can be used as an argument to any function that requires a table in its arguments. Also, the returned value can be used whenever a date value is required.</p>

DAX Time Intelligence Functions that return a Table of Dates

Function	Notes
DATEADD (dates , number_of_intervals , interval)	<p>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.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>If the number specified for number_of_intervals is positive, the dates in dates are moved forward in time; if the number is negative, the dates in dates are shifted back in time.</p> <p>The interval parameter is an enumeration, not a set of strings. Therefore, values should not be enclosed in quotation marks. Also, the values: year, quarter, month, day should be spelled in full when using them.</p> <p>The result table includes only dates that exist in the dates column.</p> <p>If the dates in the current context do not form a contiguous interval, the function returns an error.</p>
DATESBETWEEN (dates , start_date , end_date)	<p>Returns a table that contains a column of dates that begins with the start_date and continues until the end_date.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>The DATESBETWEEN function is provided for working with custom date ranges.</p> <p>If you are working with common date intervals such as months, quarters, and years, the more appropriate function is DATESINPERIOD.</p>
DATESINPERIOD (dates , start_date , number_of_intervals , interval)	<p>Returns a table that contains a column of dates that begins with the start_date and continues for the specified number_of_intervals.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>If the number specified for number_of_intervals is positive, the dates are moved forward in time; if the number is negative, the dates are shifted back in time.</p> <p>The interval parameter is an enumeration, not a set of strings. Therefore values should not be enclosed in quotation marks. Also, the values: year, quarter, month, day should be spelled in full when using them.</p> <p>The result table includes only dates that appear in the values of the underlying table column.</p> <p>This function can be used as an input to CALCULATE to create a rolling total such as Moving Annual Total (MAT), rolling 90 days, rolling 90 day average etc.</p>
DATESMTD (dates)	<p>Returns a table that contains a column of the dates for the month to date, in the current context.</p>

Function	Notes
	Designed to be used inside a CALCULATE as a table filter.
DATESQTD(dates)	Returns a table that contains a column of the dates for the quarter to date, in the current context. Designed to be used inside a CALCULATE as a table filter.
DATESYTD(dates [, year_end_date])	Returns a table that contains a column of the dates for the year to date, in the current context. Designed to be used inside a CALCULATE as a table filter. year_end_date (optional) is a literal string with a date that defines the year-end date. The default is December 31.
NEXTDAY(dates)	Returns a table that contains a column of all dates from the next day, based on the first date specified in the dates column in the current context. Designed to be used inside a CALCULATE as a table filter. E.g. You can create a measure that calculates the ' next day sales ' of the product sales.
NEXTMONTH(dates)	Returns a table that contains a column of all dates from the next month, based on the first date in the dates column in the current context. Designed to be used inside a CALCULATE as a table filter. E.g. You can create a measure that calculates the ' next month sales ' of the product sales.
NEXTQUARTER(dates)	Returns a table that contains a column of all dates in the next quarter, based on the first date specified in the dates column, in the current context. Designed to be used inside a CALCULATE as a table filter. E.g. You can create a measure that calculates the ' next quarter sales ' of the product sales.
NEXTYEAR(dates [, year_end_date])	Returns a table that contains a column of all dates in the next year, based on the first date in the dates column, in the current context. Designed to be used inside a CALCULATE as a table filter. E.g. You can create a measure that calculates the ' next year sales ' of the product sales.

Function	Notes
PARALLELPERIOD (dates , number_of_intervals , interval)	<p>Returns a table that contains a column of dates that represents a period parallel to the dates in the specified dates column, in the current context, with the dates shifted a number of intervals either forward in time or back in time.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>If the number specified for number_of_intervals is positive, the dates in dates are moved forward in time; if the number is negative, the dates in dates are shifted back in time.</p> <p>The interval parameter is an enumeration, not a set of strings. Therefore, values should not be enclosed in quotation marks. Also, the values: year, quarter, month should be spelled in full when using them.</p> <p>The result table includes only dates that appear in the values of the underlying table column.</p> <p>The PARALLELPERIOD function is similar to the DATEADD function except that PARALLELPERIOD always returns full periods at the given granularity level instead of the partial periods that DATEADD returns.</p> <p>For example, if you have a selection of dates that starts at June 10 and finishes at June 21 of the same year, and you want to shift that selection forward by one month then the PARALLELPERIOD function will return all dates from the next month (July 1 to July 31), however, if DATEADD is used instead, then the result will include only dates from July 10 to July 21.</p>
PREVIOUSDAY (dates)	<p>Returns a table that contains a column of all dates representing the day that is previous to the first date in the dates column, in the current context.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>E.g. You can create a measure 'Previous Day Sales'.</p>
PREVIOUSMONTH (dates)	<p>Returns a table that contains a column of all dates from the previous month, based on the first date in the dates column, in the current context.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>E.g. You can create a measure 'Previous Month Sales'.</p>

Function	Notes
PREVIOUSQUARTER(dates)	<p>Returns a table that contains a column of all dates from the previous quarter, based on the first date in the dates column, in the current context.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>E.g. You can create a measure 'Previous Quarter Sales'.</p>
PREVIOUSYEAR(dates [, year_end_date])	<p>Returns a table that contains a column of all dates from the previous year, given the last date in the dates column, in the current context.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>E.g. You can create a measure 'Previous Year Sales'.</p>
SAMEPERIODLASTYEAR(dates)	<p>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.</p> <p>Designed to be used inside a CALCULATE as a table filter.</p> <p>The dates returned are the same as the dates returned by this equivalent formula:</p> <p>DATEADD(dates, -1, year)</p>

DAX Functions List

S. No.	Function	XBI DAX Functions Category
1.	ABS	DAX Math and Trig Functions
2.	ACCRINT	DAX Financial Functions
3.	ACCRINTM	DAX Financial Functions
4.	ACOS	DAX Math and Trig Functions
5.	ACOSH	DAX Math and Trig Functions
6.	ACOT	DAX Math and Trig Functions
7.	ACOTH	DAX Math and Trig Functions
8.	ADDCOLUMNS	DAX Query Functions
9.	ADDMISSINGITEMS	DAX Filter Functions
10.	ALL	DAX Filter Functions
11.	ALLCROSSFILTERED	DAX Filter Functions
12.	ALLEXCEPT	DAX Filter Functions
13.	ALLNOBLANKROW	DAX Filter Functions
14.	ALLSELECTED	DAX Filter Functions
15.	AMORDEGRC	DAX Financial Functions
16.	AMORLINC	DAX Financial Functions
17.	AND	DAX Logical Functions
18.	APPROXIMATEDISTINCTCOUNT	DAX Aggregation Functions (Aggregators)
19.	ASIN	DAX Math and Trig Functions
20.	ASINH	DAX Math and Trig Functions
21.	ATAN	DAX Math and Trig Functions
22.	ATANH	DAX Math and Trig Functions
23.	AVERAGE	DAX Aggregation Functions (Aggregators)
24.	AVERAGEA	DAX Aggregation Functions (Aggregators)
25.	AVERAGEX	DAX Other Special Functions (X-Functions / Iterators)
26.	BETA.DIST	DAX Statistical Functions
27.	BETA.INV	DAX Statistical Functions
28.	BITAND	DAX Logical Functions
29.	BITLSHIFT	DAX Logical Functions
30.	BITOR	DAX Logical Functions
31.	BITRSHIFT	DAX Logical Functions

S. No.	Function	XBI DAX Functions Category
32.	BITXOR	DAX Logical Functions
33.	BLANK	DAX Text Functions
34.	CALCULATE	DAX Filter Functions
35.	CALCULATETABLE	DAX Filter Functions
36.	CALENDAR	DAX Date and Time Functions
37.	CALENDARAUTO	DAX Date and Time Functions
38.	CEILING	DAX Math and Trig Functions
39.	CHISQ.DIST	DAX Statistical Functions
40.	CHISQ.DIST.RT	DAX Statistical Functions
41.	CHISQ.INV	DAX Statistical Functions
42.	CHISQ.INV.RT	DAX Statistical Functions
43.	CLOSINGBALANCEMONTH	DAX Time Intelligence Functions that return Scalar Values
44.	CLOSINGBALANCEQUARTER	DAX Time Intelligence Functions that return Scalar Values
45.	CLOSINGBALANCEYEAR	DAX Time Intelligence Functions that return Scalar Values
46.	COALESCE	DAX Logical Functions
47.	CODE	DAX Text Functions
48.	COMBIN	DAX Math and Trig Functions
49.	COMBINA	DAX Math and Trig Functions
50.	COMBINEVALUES	DAX Text Functions
51.	CONCATENATE	DAX Text Functions
52.	CONCATENATEX	DAX Other Special Functions (X-Functions / Iterators)
53.	CONFIDENCE.NORM	DAX Statistical Functions
54.	CONFIDENCE.T	DAX Statistical Functions
55.	CONTAINS	DAX Information Functions
56.	CONTAINSROW	DAX Information Functions
57.	CONTAINSSTRING	DAX Text Functions
58.	CONTAINSSTRINGEXACT	DAX Text Functions
59.	CONVERT	DAX Other Functions
60.	COS	DAX Math and Trig Functions
61.	COSH	DAX Math and Trig Functions
62.	COT	DAX Math and Trig Functions
63.	COTH	DAX Math and Trig Functions

S. No.	Function	XBI DAX Functions Category
64.	COUNT	DAX Aggregation Functions (Aggregators)
65.	COUNTA	DAX Aggregation Functions (Aggregators)
66.	COUNTAX	DAX Other Special Functions (X-Functions / Iterators)
67.	COUNTBLANK	DAX Aggregation Functions (Aggregators)
68.	COUNTROWS	DAX Aggregation Functions (Aggregators)
69.	COUNTX	DAX Other Special Functions (X-Functions / Iterators)
70.	COUPDAYBS	DAX Financial Functions
71.	COUPDAYS	DAX Financial Functions
72.	COUPDAYSNC	DAX Financial Functions
73.	COUPNCD	DAX Financial Functions
74.	COUPNUM	DAX Financial Functions
75.	COUPPCD	DAX Financial Functions
76.	CROSSFILTER	DAX Filter Functions
77.	CROSSJOIN	DAX Query Functions
78.	CUMIPMT	DAX Financial Functions
79.	CUMPRINC	DAX Financial Functions
80.	CURRENCY	DAX Math and Trig Functions
81.	CURRENTGROUP	DAX Other Special Functions (Argument Functions)
82.	CUSTOMDATA	DAX Information Functions
83.	DATATABLE	DAX Query Functions
84.	DATE	DAX Date and Time Functions
85.	DATEADD	DAX Time Intelligence Functions that return a Table of Dates
86.	DATEDIFF	DAX Date and Time Functions
87.	DATESBETWEEN	DAX Time Intelligence Functions that return a Table of Dates
88.	DATESINPERIOD	DAX Time Intelligence Functions that return a Table of Dates
89.	DATESMTD	DAX Time Intelligence Functions that return a Table of Dates
90.	DATESQTD	DAX Time Intelligence Functions that return a Table of Dates
91.	DATESYTD	DAX Time Intelligence Functions that return a Table of Dates
92.	DATEVALUE	DAX Date and Time Functions
93.	DAY	DAX Date and Time Functions
94.	DB	DAX Financial Functions
95.	DDB	DAX Financial Functions

S. No.	Function	XBI DAX Functions Category
96.	DEGREES	DAX Math and Trig Functions
97.	DETAILROWS	DAX Other Functions
98.	DISC	DAX Financial Functions
99.	DISTINCT	DAX Filter Functions
100.	DISTINCTCOUNT	DAX Aggregation Functions (Aggregators)
101.	DISTINCTCOUNTNOBLANK	DAX Aggregation Functions (Aggregators)
102.	DIVIDE	DAX Math and Trig Functions
103.	DOLLARDE	DAX Financial Functions
104.	DOLLARFR	DAX Financial Functions
105.	DURATION	DAX Financial Functions
106.	EARLIER	DAX Filter Functions
107.	EARLIEST	DAX Filter Functions
108.	EDATE	DAX Date and Time Functions
109.	EFFECT	DAX Financial Functions
110.	ENDOFMONTH	DAX Time Intelligence Functions that return both a Table & a Scalar
111.	ENDOFQUARTER	DAX Time Intelligence Functions that return both a Table & a Scalar
112.	ENDOFYEAR	DAX Time Intelligence Functions that return both a Table & a Scalar
113.	EOMONTH	DAX Date and Time Functions
114.	ERROR	DAX Other Functions
115.	EVALUATEANDLOG	DAX Other Functions
116.	EVEN	DAX Math and Trig Functions
117.	EXACT	DAX Text Functions
118.	EXCEPT	DAX Query Functions
119.	EXP	DAX Math and Trig Functions
120.	EXPON.DIST	DAX Statistical Functions
121.	FACT	DAX Math and Trig Functions
122.	FALSE	DAX Logical Functions
123.	FILTER	DAX Filter Functions
124.	FILTERS	DAX Filter Functions
125.	FIND	DAX Text Functions
126.	FIRSTDATE	DAX Time Intelligence Functions that return both a Table & a Scalar
127.	FIRSTNONBLANK	DAX Time Intelligence Functions that return both a Table & a Scalar

S. No.	Function	XBI DAX Functions Category
128.	FIRSTNONBLANKVALUE	DAX Time Intelligence Functions that return both a Table & a Scalar
129.	FIXED	DAX Text Functions
130.	FLOOR	DAX Math and Trig Functions
131.	FORMAT	DAX Text Functions
132.	FV	DAX Financial Functions
133.	GCD	DAX Math and Trig Functions
134.	GENERATE	DAX Query Functions
135.	GENERATEALL	DAX Query Functions
136.	GENERATESERIES	DAX Query Functions
137.	GEOMEAN	DAX Statistical Functions
138.	GEOMEANX	DAX Other Special Functions (X-Functions / Iterators)
139.	GROUPBY	DAX Query Functions
140.	HASONEFILTER	DAX Filter Functions
141.	HASONEVALUE	DAX Filter Functions
142.	HOUR	DAX Date and Time Functions
143.	IF	DAX Logical Functions
144.	IF.EAGER	DAX Logical Functions
145.	IFERROR	DAX Logical Functions
146.	IGNORE	DAX Other Special Functions (Argument Functions)
147.	IN Operator	DAX Information Functions
148.	INDEX	DAX Filter Functions
149.	INT	DAX Math and Trig Functions
150.	INTERSECT	DAX Query Functions
151.	INTRATE	DAX Financial Functions
152.	IPMT	DAX Financial Functions
153.	ISAFTER	DAX Information Functions
154.	ISBLANK	DAX Information Functions
155.	ISCROSSFILTERED	DAX Filter Functions
156.	ISEMPTY	DAX Information Functions
157.	ISERROR	DAX Information Functions
158.	ISEVEN	DAX Information Functions
159.	ISFILTERED	DAX Filter Functions

S. No.	Function	XBI DAX Functions Category
160.	ISINSCOPE	DAX Information Functions
161.	ISLOGICAL	DAX Information Functions
162.	ISNONTEXT	DAX Information Functions
163.	ISNUMBER	DAX Information Functions
164.	ISO.CEILING	DAX Math and Trig Functions
165.	ISODD	DAX Information Functions
166.	ISONORAFTER	DAX Information Functions
167.	ISPMT	DAX Financial Functions
168.	ISSELECTEDMEASURE	DAX Information Functions
169.	ISSUBTOTAL	DAX Information Functions
170.	ISTEXT	DAX Information Functions
171.	KEEPFILTERS	DAX Filter Functions
172.	LASTDATE	DAX Time Intelligence Functions that return both a Table & a Scalar
173.	LASTNONBLANK	DAX Time Intelligence Functions that return both a Table & a Scalar
174.	LASTNONBLANKVALUE	DAX Time Intelligence Functions that return both a Table & a Scalar
175.	LCM	DAX Math and Trig Functions
176.	LEFT	DAX Text Functions
177.	LEN	DAX Text Functions
178.	LN	DAX Math and Trig Functions
179.	LOG	DAX Math and Trig Functions
180.	LOG10	DAX Math and Trig Functions
181.	LOOKUPVALUE	DAX Information Functions
182.	LOWER	DAX Text Functions
183.	MAX	DAX Aggregation Functions (Aggregators)
184.	MAXA	DAX Aggregation Functions (Aggregators)
185.	MAXX	DAX Other Special Functions (X-Functions / Iterators)
186.	MDURATION	DAX Financial Functions
187.	MEDIAN	DAX Statistical Functions
188.	MEDIANX	DAX Other Special Functions (X-Functions / Iterators)
189.	MID	DAX Text Functions
190.	MIN	DAX Aggregation Functions (Aggregators)
191.	MINA	DAX Aggregation Functions (Aggregators)

S. No.	Function	XBI DAX Functions Category
192.	MINUTE	DAX Date and Time Functions
193.	MINX	DAX Other Special Functions (X-Functions / Iterators)
194.	MOD	DAX Math and Trig Functions
195.	MONTH	DAX Date and Time Functions
196.	MROUND	DAX Math and Trig Functions
197.	NATURALINNERJOIN	DAX Query Functions
198.	NATURALLEFTOUTERJOIN	DAX Query Functions
199.	NETWORKDAYS	DAX Date and Time Functions
200.	NEXTDAY	DAX Time Intelligence Functions that return a Table of Dates
201.	NEXTMONTH	DAX Time Intelligence Functions that return a Table of Dates
202.	NEXTQUARTER	DAX Time Intelligence Functions that return a Table of Dates
203.	NEXTYEAR	DAX Time Intelligence Functions that return a Table of Dates
204.	NOMINAL	DAX Financial Functions
205.	NONVISUAL	DAX Other Special Functions (Argument Functions)
206.	NORM.DIST	DAX Statistical Functions
207.	NORM.INV	DAX Statistical Functions
208.	NORM.S.DIST	DAX Statistical Functions
209.	NORM.S.INV	DAX Statistical Functions
210.	NOT	DAX Logical Functions
211.	NOW	DAX Date and Time Functions
212.	NPER	DAX Financial Functions
213.	ODD	DAX Math and Trig Functions
214.	ODDFPRICE	DAX Financial Functions
215.	ODDFYIELD	DAX Financial Functions
216.	ODDLPRICE	DAX Financial Functions
217.	ODDLYIELD	DAX Financial Functions
218.	OFFSET	DAX Filter Functions
219.	OPENINGBALANCEMONTH	DAX Time Intelligence Functions that return Scalar Values
220.	OPENINGBALANCEQUARTER	DAX Time Intelligence Functions that return Scalar Values
221.	OPENINGBALANCEYEAR	DAX Time Intelligence Functions that return Scalar Values
222.	OR	DAX Logical Functions
223.	ORDERBY	DAX Other Special Functions (Argument Functions)

S. No.	Function	XBI DAX Functions Category
224.	PARALLELPERIOD	DAX Time Intelligence Functions that return a Table of Dates
225.	PARTITIONBY	DAX Other Special Functions (Argument Functions)
226.	PATH	DAX Parent and Child Functions
227.	PATHCONTAINS	DAX Parent and Child Functions
228.	PATHITEM	DAX Parent and Child Functions
229.	PATHITEMREVERSE	DAX Parent and Child Functions
230.	PATHLENGTH	DAX Parent and Child Functions
231.	PDURATION	DAX Financial Functions
232.	PERCENTILE.EXC	DAX Statistical Functions
233.	PERCENTILE.INC	DAX Statistical Functions
234.	PERCENTILEX.EXC	DAX Other Special Functions (X-Functions / Iterators)
235.	PERCENTILEX.INC	DAX Other Special Functions (X-Functions / Iterators)
236.	PERMUT	DAX Math and Trig Functions
237.	PI	DAX Math and Trig Functions
238.	PMT	DAX Financial Functions
239.	POISSON.DIST	DAX Statistical Functions
240.	POWER	DAX Math and Trig Functions
241.	PPMT	DAX Financial Functions
242.	PREVIOUSDAY	DAX Time Intelligence Functions that return a Table of Dates
243.	PREVIOUSMONTH	DAX Time Intelligence Functions that return a Table of Dates
244.	PREVIOUSQUARTER	DAX Time Intelligence Functions that return a Table of Dates
245.	PREVIOUSYEAR	DAX Time Intelligence Functions that return a Table of Dates
246.	PRICE	DAX Financial Functions
247.	PRICEDISC	DAX Financial Functions
248.	PRICEMAT	DAX Financial Functions
249.	PRODUCT	DAX Aggregation Functions (Aggregators)
250.	PRODUCTX	DAX Other Special Functions (X-Functions / Iterators)
251.	PV	DAX Financial Functions
252.	QUARTER	DAX Date and Time Functions
253.	QUOTIENT	DAX Math and Trig Functions
254.	RADIANS	DAX Math and Trig Functions
255.	RAND	DAX Math and Trig Functions

S. No.	Function	XBI DAX Functions Category
256.	RANDBETWEEN	DAX Math and Trig Functions
257.	RANK.EQ	DAX Statistical Functions
258.	RANKX	DAX Other Special Functions (X-Functions / Iterators)
259.	RATE	DAX Financial Functions
260.	RECEIVED	DAX Financial Functions
261.	RELATED	DAX Filter Functions
262.	RELATEDTABLE	DAX Filter Functions
263.	REMOVEFILTERS	DAX Filter Functions
264.	REPLACE	DAX Text Functions
265.	REPT	DAX Text Functions
266.	RIGHT	DAX Text Functions
267.	ROLLUP	DAX Other Special Functions (Argument Functions)
268.	ROLLUPADDISUBTOTAL	DAX Other Special Functions (Argument Functions)
269.	ROLLUPGROUP	DAX Other Special Functions (Argument Functions)
270.	ROLLUPISSUBTOTAL	DAX Other Special Functions (Argument Functions)
271.	ROUND	DAX Math and Trig Functions
272.	ROUNDDOWN	DAX Math and Trig Functions
273.	ROUNDUP	DAX Math and Trig Functions
274.	ROW	DAX Query Functions
275.	RRI	DAX Financial Functions
276.	SAMEPERIODLASTYEAR	DAX Time Intelligence Functions that return a Table of Dates
277.	SAMPLE	DAX Statistical Functions
278.	SEARCH	DAX Text Functions
279.	SECOND	DAX Date and Time Functions
280.	SELECTCOLUMNS	DAX Query Functions
281.	SELECTEDMEASURE	DAX Information Functions
282.	SELECTEDMEASUREFORMATSTRING	DAX Information Functions
283.	SELECTEDMEASURENAME	DAX Information Functions
284.	SELECTEDVALUE	DAX Filter Functions
285.	SIGN	DAX Math and Trig Functions
286.	SIN	DAX Math and Trig Functions

S. No.	Function	XBI DAX Functions Category
287.	SINH	DAX Math and Trig Functions
288.	SLN	DAX Financial Functions
289.	SQRT	DAX Math and Trig Functions
290.	SQRTPI	DAX Math and Trig Functions
291.	STARTOFMONTH	DAX Time Intelligence Functions that return both a Table & a Scalar
292.	STARTOFQUARTER	DAX Time Intelligence Functions that return both a Table & a Scalar
293.	STARTOFYEAR	DAX Time Intelligence Functions that return both a Table & a Scalar
294.	STDEV.P	DAX Statistical Functions
295.	STDEV.S	DAX Statistical Functions
296.	STDEVX.P	DAX Other Special Functions (X-Functions / Iterators)
297.	STDEVX.S	DAX Other Special Functions (X-Functions / Iterators)
298.	SUBSTITUTE	DAX Text Functions
299.	SUBSTITUTEWITHINDEX	DAX Filter Functions
300.	SUM	DAX Aggregation Functions (Aggregators)
301.	SUMMARIZE	DAX Query Functions
302.	SUMMARIZECOLUMNS	DAX Query Functions
303.	SUMX	DAX Other Special Functions (X-Functions / Iterators)
304.	SWITCH	DAX Logical Functions
305.	SYD	DAX Financial Functions
306.	T.DIST	DAX Statistical Functions
307.	T.DIST.2T	DAX Statistical Functions
308.	T.DIST.RT	DAX Statistical Functions
309.	T.INV	DAX Statistical Functions
310.	T.INV.2T	DAX Statistical Functions
311.	TAN	DAX Math and Trig Functions
312.	TANH	DAX Math and Trig Functions
313.	TBILLEQ	DAX Financial Functions
314.	TBILLPRICE	DAX Financial Functions
315.	TBILLYIELD	DAX Financial Functions
316.	TIME	DAX Date and Time Functions
317.	TIMEVALUE	DAX Date and Time Functions
318.	TOCSV	DAX Text Functions

S. No.	Function	XBI DAX Functions Category
319.	TODAY	DAX Date and Time Functions
320.	TOJSON	DAX Text Functions
321.	TOPN	DAX Aggregation Functions (Aggregators)
322.	TOTALMTD	DAX Time Intelligence Functions that return Scalar Values
323.	TOTALQTD	DAX Time Intelligence Functions that return Scalar Values
324.	TOTALYTD	DAX Time Intelligence Functions that return Scalar Values
325.	TREATAS	DAX Filter Functions
326.	TRIM	DAX Text Functions
327.	TRUE	DAX Logical Functions
328.	TRUNC	DAX Math and Trig Functions
329.	UNICHAR	DAX Text Functions
330.	UNICODE	DAX Text Functions
331.	UNION	DAX Query Functions
332.	UPPER	DAX Text Functions
333.	USERCULTURE	DAX Information Functions
334.	USERRELATIONSHIP	DAX Filter Functions
335.	USERNAME	DAX Information Functions
336.	USEROBJECTID	DAX Information Functions
337.	USERPRINCIPALNAME	DAX Information Functions
338.	UTCNOW	DAX Date and Time Functions
339.	UTCTODAY	DAX Date and Time Functions
340.	VALUE	DAX Text Functions
341.	VALUES	DAX Filter Functions
342.	VAR.P	DAX Statistical Functions
343.	VAR.S	DAX Statistical Functions
344.	VARX.P	DAX Other Special Functions (X-Functions / Iterators)
345.	VARX.S	DAX Other Special Functions (X-Functions / Iterators)
346.	VDB	DAX Financial Functions
347.	WEEKDAY	DAX Date and Time Functions
348.	WEEKNUM	DAX Date and Time Functions
349.	WINDOW	DAX Filter Functions
350.	XIRR	DAX Financial Functions

S. No.	Function	XBI DAX Functions Category
351.	XNPV	DAX Financial Functions
352.	YEAR	DAX Date and Time Functions
353.	YEARFRAC	DAX Date and Time Functions
354.	YIELD	DAX Financial Functions
355.	YIELDDISC	DAX Financial Functions
356.	YIELDMAT	DAX Financial Functions