DAX Cheatsheet

by Venkata Reddy Konasani

Function Name	Syntax	Explanation
Maths & Statistical Functions		
SUM	`SUM(Sales)`	This function would add up all the values in the Sales column.
SUMX	`SUMX(Orders, Orders[Quantity]*Orders[Price])`	This function would evaluate the expression (Quantity * Price) for each row in the Orders table, and then return the sum of those values.
AVERAGE	`AVERAGE(Age)`	This function would return the average age of all the values in the Age column.
AVERAGEX	`AVERAGEX(Employees, Employees[Salary]/Employees[Experience])`	This function would evaluate the expression (Salary/Experience) for each row in the Employees table, and then return the average of those values.
MEDIAN	`MEDIAN(TestScores)`	This function would return the median of all the values in the TestScores column.
MEDIANX	`MEDIANX(Students, Students[MathScore] + Students[ScienceScore])`	This function would evaluate the expression (MathScore + ScienceScore) for each row in the Students table, and then return the median of those values.
GEOMEAN	`GEOMEAN(Sales)`	This function would calculate the geometric mean of all the values in the Sales column.
GEOMEANX	`GEOMEANX(Products, Products[Price]*Products[Quantity])`	This function would evaluate the expression (Price*Quantity) for each row in the Products table, and then return the geometric mean of those values.
COUNT	`COUNT(Customers)`	This function would return the number of cells in the Customers column that contain non-blank values.
COUNTX	`COUNTX(Orders, Orders[Quantity]>0)`	This function would evaluate the expression (Quantity>0) for each row in the Orders table, and then return the number of rows where that expression evaluates to true (i.e. the number of non-blank rows in the Quantity column).
DIVIDE	`DIVIDE(TotalSales, TotalCost, BLANK())`	This function would perform the division TotalSales/TotalCost, and return BLANK() if the denominator is 0.
DIVIDE	`DIVIDE(TotalSales, TotalCost, 0)`	This function would perform the division TotalSales/TotalCost, and return 0 if the denominator is 0.
MIN	`MIN(Age)`	This function would return the minimum value in the Age column.

MAX	`MAX(Price)`	This function would return the maximum value in the Price column.
COUNTROWS	`COUNTROWS(Orders)`	This function would return the number of rows in the Orders table.
DISTINCTCOUNT	`DISTINCTCOUNT(Customers[Name])`	This function would return the number of distinct names in the Name column of the Customers table.
RANKX	`RANKX(Orders, Orders[Total],,ASC,DENSE)`	This function would return the ranking of the Total column of the Orders table, with the smallest value having rank 1, and if there are ties, return the same rank to all the tied elements.
RANKX	`RANKX(Orders, Orders[Total],,DESC,DENSE)`	This function would return the ranking of the Total column of the Orders table, with the largest value having rank 1, and if there are ties, return the same rank to all the tied elements.

Filter Functions

FILTER	`FILTER(Orders, Orders[Total] > 100)`	This function would return a table that is a subset of the Orders table and includes only the rows where the value in the Total column is greater than 100.
CALCULATE	`CALCULATE(SUM(Sales), FILTER(Customers, Customers[City] = "Seattle"))`	This function would evaluate the SUM of the Sales column in a context where the filter is applied to only include the rows where the City column in the Customers table is "Seattle".
HASONEVALUE	`HASONEVALUE(City)`	This function would return TRUE if the context for the City column has been filtered down to one distinct value only. Otherwise, it would return FALSE.
ALLNOBLANKROW	`ALLNOBLANKROW(Orders)`	This function would return a table that is a subset of the Orders table, including only the rows that are not blank.
ALL	`ALL(Customers)`	This function would return all the rows in the Customers table, ignoring any filters that might have been applied.
ALLEXCEPT	`ALLEXCEPT(Customers, Customers[City])`	This function would return all the rows in the Customers table except for those rows that are affected by the specified City column filter.
REMOVEFILTERS	`REMOVEFILTERS(Orders)`	This function would clear all filters from the Orders table.

Logical Functions

IF	`IF(Sales[Qty]>10, "High", "Low")`	This function checks if the value in the Qty column of the Sales table is greater than 10. If it is, it returns "High", otherwise it returns "Low".

AND	`AND(Sales[Qty]>10, Sales[Qty]<20)`	This function checks if the value in the Qty column of the Sales table is greater than 10 and less than 20. If both conditions are true, it returns TRUE, otherwise it returns FALSE.
OR	`OR(Sales[Qty]>10, Sales[Qty]<20)`	This function checks if the value in the Qty column of the Sales table is greater than 10 or less than 20. If either condition is true, it returns TRUE, otherwise it returns FALSE.
NOT	`NOT(Sales[Qty]>10)`	This function checks if the value in the Qty column of the Sales table is greater than 10. If it is, it returns FALSE, otherwise it returns TRUE.
SWITCH	`SWITCH(Sales[Qty], 10, "Low", 20, "Medium", 30, "High", "Out of range")`	This function checks the value in the Qty column of the Sales table. If it is 10, it returns "Low", if it is 20, it returns "Medium", if it is 30, it returns "High", and if it is none of these values, it returns "Out of range".
IFERROR	`IFERROR(DIVIDE(Sales[Qty],0),"Error")`	This function checks if the division of Qty column of Sales table by 0 is an error. If it is, it returns "Error", otherwise it returns the result of the division.

Date & Time Functions

CALENDAR	`CALENDAR(DATE(2022,1,1),DATE(2022,12,31))`	Returns a table with a single column named "Date" that contains a contiguous set of dates between January 1, 2022 and December 31, 2022
DATE	`DATE(2022,1,1)`	Returns January 1, 2022 in datetime format
DATEDIFF	`DATEDIFF(DATE(2022,1,1),DATE(2022,2,1),"month")`	Returns the number of months between January 1, 2022 and February 1, 2022
DATEVALUE	`DATEVALUE("2022-01-01")`	Converts the text "2022-01-01" to the date January 1, 2022 in datetime format
DAY	`DAY(DATE(2022,1,1))`	Returns the day of the month for January 1, 2022, which is 1
WEEKNUM	`WEEKNUM(DATE(2022,1,1))`	Returns the week number for January 1, 2022 which is 1
MONTH	`MONTH(DATE(2022,1,1))`	Returns the month for January 1, 2022, which is 1
QUARTER	`QUARTER(DATE(2022,1,1))`	Returns the quarter for January 1, 2022, which is 1

Time Intelligence Functions

DATEADD	`DATEADD(Dates[Date], 1, "month")`	This function moves the date in the "Date" column of the "Dates" table by 1 month and returns the new date.
DATESBETWEEN	`DATESBETWEEN(Dates[Date], "2022-01-01", "2022-12-31")`	This function returns all the dates in the "Date" column of the "Dates" table that are between "2022-01-01" and "2022-12-31".

TOTALYTD	`TOTALYTD(SUM(Sales[Revenue]), Dates[Date], "2022-12-31")`	This function returns the year-to-date total revenue for the current context, up until the date "2022-12-31", for the "Revenue" column in the "Sales" table.
SAMEPERIODLASTYEAR	`SAMEPERIODLASTYEAR(Dates[Date])`	This function returns a table with a single column "Date" that contains the same dates as the current context, but shifted back one year.
STARTOFMONTH	`STARTOFMONTH(Dates[Date])`	This function returns the first day of the month for each date in the "Date" column of the "Dates" table.
ENDOFMONTH	`ENDOFMONTH(Dates[Date])`	This function returns the last day of the month for each date in the "Date" column of the "Dates" table.
STARTOFQUARTER	`STARTOFQUARTER(Dates[Date])`	This function returns the first day of the quarter for each date in the "Date" column of the "Dates" table.
ENDOFQUARTER	`ENDOFQUARTER(Dates[Date])`	This function returns the last day of the quarter for each date in the "Date" column of the "Dates" table.
STARTOFYEAR	`STARTOFYEAR(Dates[Date])`	This function returns the first day of the year for each date in the "Date" column of the "Dates" table.
ENDOFYEAR	`ENDOFYEAR(Dates[Date])`	This function returns the last day of the year for each date in the "Date" column of the "Dates" table.
PARALLELPERIOD	`PARALLELPERIOD(Dates, -3, MONTH)`	Returns a table that contains a column of dates shifted back by 3 months from the current date in the current filter context.
LASTDATE	`LASTDATE(Dates)`	Returns the last date of the specified column in the current filter context.
LASTNONBLANK	`LASTNONBLANK(Sales, Dates)`	Returns the last non-blank value of the Sales expression for the current date.
TOTALMTD	`TOTALMTD(Sales, Dates)`	Returns the month-to-date total of the Sales expression in the current filter context.
TOTALQTD	`TOTALQTD(Sales, Dates)`	Returns the quarter-to-date total of the Sales expression in the current filter context.
TOTALWTD	`TOTALWTD(Sales, Dates)`	Returns the week-to-date total of the Sales expression in the current filter context.
YEARFRAC	`YEARFRAC(Dates[StartDate], Dates[EndDate], 365)`	Returns the year fraction representing the number of years between two dates based on a day count basis of 365 days.

Relationship Functions

CROSSFILTER	`CROSSFILTER(Product[ProductName], Sales[ProductName], "Both")`	This function specifies that both the Product and Sales tables should be filtered by the ProductName column.
RELATED	`RELATED(Product[Price])`	This function returns the value of the Price column from the related Product table in the current context.

USERELATIONSHIP	`USERELATIONSHIP(Orders[CustomerID], Customers[CustomerID], "=")`	Forces the use of a specific relationship between two columns.
VALUES	`VALUES(Customers)`	Returns a one-column table that contains the distinct values from the specified column of the table.
FILTER	`FILTER(Orders, Orders[Total] > 100)`	Returns a table that includes only the rows from the original table that match the specified filter conditions.
ALL	`ALL(Customers)`	Returns a table that has the same columns as the input table, but with all rows removed that have a BLANK or NULL value in any column.
COUNTROWS	`COUNTROWS(Orders)`	Returns the number of rows in a table.
COUNTAX	`COUNTAX(Orders, Orders[Product])`	Returns the number of non-BLANK values in a column.
MINX	`MINX(Orders, Orders[Total])`	Returns the minimum value of the expression, after evaluating the expression for each row in the table.
MAXX	`MAXX(Orders, Orders[Total])`	Returns the maximum value of the expression, after evaluating the expression for each row in the table.
SUMX	`SUMX(Orders, Orders[Total])`	Returns the sum of the expression evaluated for each row in the table.
RANKX	`RANKX(Orders, Orders[Total], 100, "asc")`	Returns the rank of a value within a column, based on the specified order.
GROUPBY	`GROUPBY(Orders, Orders[CustomerID], "SUMX", Orders[Total])`	Returns a summary table over a set of groups.

Table Manipulation Functions

SUMMARIZE	`SUMMARIZE(Sales, Sales[Region], "Total Sales", SUM(Sales[SalesAmount]))`	Returns a summary table that groups the sales data by region and calculates the total sales for each region.
DISTINCT	`DISTINCT(FILTER(Product, Product[Color] = "Red"))`	Returns a table that contains only the distinct products that have a color of red.
ADDCOLUMNS	`ADDCOLUMNS(Product, "DiscountedPrice", Product[Price] * 0.8)`	Adds a new column "DiscountedPrice" to the Product table with a calculated value of 80% of the original price.
SELECTCOLUMNS	`SELECTCOLUMNS(Product, "Name", Product[Name], "Price", Product[Price])`	Returns a table that contains only the "Name" and "Price" columns from the Product table.
GROUPBY	`GROUPBY(FILTER(Sales, Sales[Region] = "West"), Sales[Category], "Total Sales", SUM(Sales[SalesAmount]))`	Returns a summary table that groups the sales data by category, filtered to only include data from the "West" region, and calculates the total sales for each category.
INTERSECT	`INTERSECT(FILTER(Customers, Customers[Country] = "USA"), FILTER(Orders, Orders[OrderDate] > "01/01/2020"))`	Returns the rows of the Customers table where the country is "USA" and also exist in the Orders table with an OrderDate greater than "01/01/2020".
NATURALINNERJOIN	`NATURALINNERJOIN(Customers, Orders)`	Joins the Customers and Orders table where the CustomerID column in both tables match.

NATURALLEFTOUTERJOIN	`NATURALLEFTOUTERJOIN(Customers, Orders)`	Joins the Customers and Orders table where the CustomerID column in both tables match, with all rows from the Customers table included and any matching rows from the Orders table. If there is no match, the columns from the Orders table will be filled with BLANK.
UNION	`UNION(FILTER(Customers, Customers[Country] = "USA"), FILTER(Customers, Customers[Country] = "Canada"))`	Returns a new table with all the rows from Customers table where the country is "USA" and also all the rows from the Customers table where the country is "Canada".

Text Functions

EXACT(<text_1>, <text_2>)	`EXACT("Apple", "apple")`	Returns false as the text inputs are not identical (case sensitive)
FIND(<text_tofind>, <in_text>)	`FIND("e", "Example")`	Returns 2 as "e" is the second character in "Example"
FORMAT(<value>, <format>)	`FORMAT(12345.678, "#,##0.00")`	Returns 12,345.68 as the value is formatted as a number with two decimal places
LEFT(<text>, <num_chars>)	`LEFT("Example", 3)`	Returns "Exa" as it returns the first 3 characters of the input text
RIGHT(<text>, <num_chars>)	`RIGHT("Example", 4)`	Returns "mple" as it returns the last 4 characters of the input text
LEN(<text>)	`LEN("Example")`	Returns 7 as it returns the number of characters in the input text
LOWER	`LOWER("HeLLo WoRLD")`	Converts all letters in a string to lowercase.
UPPER	`UPPER("HeLLo WoRLD")`	Converts all letters in a string to uppercase.
TRIM	`TRIM(" Hello World ")`	Removes all spaces from a text string.
CONCATENATE	`CONCATENATE("Hello", " World")`	Joins two strings together into one string.
SUBSTITUTE	`SUBSTITUTE("Hello World", "World", "Universe", 1)`	Replaces the first occurrence of "World" with "Universe" in the string "Hello World".
REPLACE	`REPLACE("Hello World", 7, 5, "Universe")`	Replaces the 5 characters starting at position 7 in the string "Hello World" with "Universe" resulting in "Hello Universe"

Information Functions

COLUMNSTATISTICS()	`COLUMNSTATISTICS()`	Returns statistics regarding every column in every table. This function has no arguments.
NAMEOF(<value>)	`NAMEOF(SUM(Sales))`	Returns the column or measure name of a value, in this case "Sales".
ISBLANK(<value>)	`ISBLANK(#REF!)`	Returns whether the value in cell A1 is blank.
ISERROR(<value>)	`ISERROR(#REF!/#REF!)`	Returns whether the result of the expression (A1/B1) is an error.

ISLOGICAL(<value>)	`ISLOGICAL(TRUE())`	Checks whether the value is logical and returns true if it is.
ISNUMBER(<value>)	`ISNUMBER(#REF!)`	Checks whether the value in cell A1 is a number and returns true if it is.
USERPRINCIPALNAME ()	`USERPRINCIPALNAME()`	Returns the user principal name or email address. This function has no arguments.

DAX Statements

VAR(<name> = <expression>)	`VAR SalesTotal = SUM(Sales)`	Stores the result of the SUM(Sales) expression as a named variable called SalesTotal. To return the variable, use RETURN after the variable is defined.
COLUMN(<table>[<column>] = <expression>)	`COLUMN(Sales[Total Sales] = SUM(Sales))`	Stores the result of the SUM(Sales) expression as a new column called Total Sales in the Sales table.
ORDER BY(<table>[<column>])	`ORDERBY(Sales[Sales], DESC)`	Sorts the Sales column in descending order in the Sales table.

Calculate & Filter Functions

CALCULATE	`CALCULATE(SUM(Sales),FILTER(ALL(Product),Product[Category] = "Bikes"))`	This example uses the CALCULATE function to sum the total sales for all products in the "Bikes" category. The FILTER function is used to filter the products table to only include the "Bikes" category. The ALL function is used to remove any existing filters on the Product[Category] column.
FILTER	`FILTER(ALL(Product),Product[Color] = "Red" && Product[Size] = "Large")`	This example uses the FILTER function to select all products that are "Red" and "Large" size. The ALL function is used to remove any existing filters on the Product[Color] and Product[Size] columns. The logical operator "&&" is used to combine the two filter conditions.
CALCULATE	`CALCULATE(SUM(Sales),FILTER(ALL(Product),Product[Color]="Red"))`	This calculates the total sales of all red products by summing the Sales column while filtering only red products using the Product[Color] column.
FILTER	`FILTER(ALL(Product),Product[Color]="Red" && Product[Price]>10)`	This filters all products where the color is red and the price is greater than 10, returning only the rows that meet this criteria.
CALCULATE	`CALCULATE(COUNT(Customer),FILTER(ALL(Region),Region[Country]="USA"))`	This calculates the total number of customers in the USA by counting the Customer column while filtering only customers in the USA using the Region[Country] column.
FILTER	`FILTER(ALL(Orders),Orders[OrderDate]>=DATE(2020,1,1) && Orders[OrderDate]<DATE(2020,12,31))`	This filters all orders with an order date greater than or equal to January 1st, 2020 and less than December 31st, 2020, returning only the rows that meet this criteria.

CALCULATE	`CALCULATE(AVERAGE(Inventory[Stock]),FILTER(ALL(Category),Category[Name]="Electronics"))`	This calculates the average stock level of all electronics products by averaging the Stock column of the Inventory table while filtering only the rows where the Category[Name] column is equal to "Electronics".
FILTER	`FILTER(ALL(Employee),Employee[Salary]>50000 && Employee[Department]="IT")`	This filters all employees with a salary greater than 50000 and who are in the IT department, returning only the rows that meet this criteria.