• POWER BI DAX FORMULA

I. Aggregation Functions

These functions calculate a scalar value (single value) from a column or table.

- **AVERAGE()**: Returns the average of values in a column.
- **AVERAGEA()**: Returns the average of values in a column, including text and logical values.
- **AVERAGEX(<table_name>,)**: Returns the average of an expression evaluated for each row in a table.
- **COUNT()**: Counts numeric values in a column.
- **COUNTA()**: Counts non-blank values in a column.
- **COUNTBLANK()**: Counts blank cells in a column.
- **COUNTROWS(<table_name>)**: Counts the number of rows in a table.
- **COUNTX(<table_name>,)**: Counts rows where an expression evaluates to a non-blank value.
- **DISTINCTCOUNT()**: Counts the number of distinct values in a column.
- **DISTINCTCOUNTNOBLANK()**: Counts the number of distinct non-blank values in a column.
- **MAX()**: Returns the largest value in a column.
- **MAXA()**: Returns the largest value in a column, including text and logical values.
- **MAXX(<table_name>,)**: Returns the largest value resulting from evaluating an expression for each row of a table.
- **MEDIAN()**: Returns the median value of a column.
- **MEDIANX(<table_name>,)**: Returns the median value of an expression evaluated for each row in a table.
- **MIN()**: Returns the smallest value in a column.
- **MINA()**: Returns the smallest value in a column, including text and logical values.
- **MINX(<table_name>,)**: Returns the smallest value resulting from evaluating an expression for each row of a table.
- **STDEV.S()**: Returns the sample standard deviation of a column.
- **STDEV.P()**: Returns the population standard deviation of a column.
- **STDEVX.S(<table_name>,)**: Returns the sample standard deviation of an expression evaluated for each row in a table.
- **STDEVX.P(<table_name>,)**: Returns the population standard deviation of an expression evaluated for each row in a table.

- **SUM()**: Returns the sum of values in a column.
- **SUMX(<table_name>,)**: Returns the sum of an expression evaluated for each row in a table.
- **VAR.S()**: Returns the sample variance of a column.
- **VAR.P()**: Returns the population variance of a column.
- **VARX.S(<table_name>,)**: Returns the sample variance of an expression evaluated for each row in a table.
- **VARX.P(<table_name>,)**: Returns the population variance of an expression evaluated for each row in a table.
- **GEOMEAN()**: Returns the geometric mean of the values in a column.
- **GEOMEANX(<table_name>,)**: Returns the geometric mean of an expression evaluated for each row in a table.

II. Date and Time Functions

These functions manipulate date and time values.

- **CALENDAR(<start_date>, <end_date>)**: Returns a table of dates between start and end dates.
- **CALENDARAUTO([<fiscal_year_end_month>])**: Returns a table of dates for the model, automatically detecting the range.
- **DATE(, ,)**: Returns a date value.
- **DATEVALUE(<date_text>)**: Converts text to a date.
- **DAY()**: Returns the day of the month from a date.
- **HOUR()**: Returns the hour from a datetime value.
- **MINUTE()**: Returns the minute from a datetime value.
- **MONTH()**: Returns the month from a date.
- **NOW()**: Returns the current date and time.
- **SECOND()**: Returns the second from a datetime value.
- **TIME(, ,)**: Returns a time value.
- **TIMEVALUE(<time_text>)**: Converts text to a time value.
- **TODAY()**: Returns the current date.
- **WEEKDAY(, [<return_type>])**: Returns the day of the week.
- **WEEKNUM(,[<return_type>])**:Returns the week number in the year.
- **YEAR()**: Returns the year from a date.
- **DATEDIFF(<start_date>, <end_date>,)**: Returns the difference between two dates.
- **DATEADD(, ,)**: Returns a table of dates shifted by a specified interval.
- **EDATE(<start_date>,)**: Returns the date that is the specified number of months before or after the start date.

- **EOMONTH(<start_date>,)**: Returns the last day of the month.

III. Time Intelligence Functions

These functions facilitate calculations across time periods.

- **CLOSINGBALANCEMONTH(, , [])**: Returns the closing balance for a month.
- **CLOSINGBALANCEQUARTER(, , [])**: Returns the closing balance for a quarter.
- **CLOSINGBALANCEYEAR(, , [])**: Returns the closing balance for a year.
- **OPENINGBALANCEMONTH(, , [])**: Returns the opening balance for a month.
- **OPENINGBALANCEQUARTER(, , [])**: Returns the opening balance for a quarter.
- **OPENINGBALANCEYEAR(, , [])**: Returns the opening balance for a year.
- **ENDOFMONTH()**: Returns the last date of the month.
- **ENDOFQUARTER()**: Returns the last date of the quarter.
- **ENDOFYEAR(, [<year_end_date>])**: Returns the last date of the year.
- **FIRSTDATE()**: Returns the first date in a column.
- **FIRSTNONBLANK(,)**: Returns the first non-blank value in a column.
- **LASTDATE()**: Returns the last date in a column.
- **LASTNONBLANK(,)**: Returns the last non-blank value in a column.
- **NEXTDAY()**: Returns the next day after the date.
- **NEXTMONTH()**: Returns the first day of the next month.
- **NEXTQUARTER()**: Returns the first day of the next quarter.
- **NEXTYEAR()**: Returns the first day of the next year.
- **PREVIOUSDAY()**: Returns the previous day before the date.
- **PREVIOUSMONTH()**: Returns the first day of the previous month.
- **PREVIOUSQUARTER()**: Returns the first day of the previous quarter.
- **PREVIOUSYEAR()**: Returns the first day of the previous year.
- **SAMEPERIODLASTYEAR(, [<date_filter>])**: Returns a table of dates from the same period in the previous year.
- **DATESBETWEEN(, <start_date>, <end_date>)**: Returns a table of dates between two dates.
- **DATESINPERIOD(, <start_date>, <number_of_intervals>,)**: Returns a table of dates within a specified period.
- **DATEADD(, ,)**: Returns a table of dates shifted by a specified number of intervals.
- **DATESMTD()**: Returns a table containing the dates for the month to date.
- **DATESQTD()**: Returns a table containing the dates for the quarter to date.
- **DATESYTD(, [<year_end_date>])**: Returns a table containing the dates for the year to date.
- **PARALLELPERIOD(, <number_of_intervals>,)**: Returns a table of dates shifted

by a specified interval.

IV. Filter Functions

These functions manipulate the filter context of calculations.

- **ALL([<table_name> | <column_name>])**: Removes filters from a table or column.
- **ALLCROSSFILTERED(<table_name>)**: Removes all filters that come from the same table.
- **ALLEXCEPT(<table_name>, , , ...)**: Removes all filters from a table except for those on specified columns.
- **ALLNOBLANKROW()**:Removes all filters from the table that are applied as a result of blank row elimination.
- **ALLSELECTED([<table_name> | <column_name>])**: Returns all values in a table or column, ignoring filters from the same table but keeping filters from other tables.
- **CALCULATE(, , , ...)**: Evaluates an expression in a modified filter context.
- **CALCULATETABLE(<table_expression>, , , ...)**: Evaluates a table expression in a modified filter context.
- **FILTER(<table_name>, <filter_expression>)**: Returns a table that is a subset of another table, based on a filter.
- **FILTERS()**: Returns the filters currently applied to a column.
- **HASONEFILTER()**: Returns TRUE if a column has one filter applied.
- **HASONEVALUE()**: Returns TRUE if a column has one distinct value in the current filter context.
- **ISFILTERED()**: Returns TRUE if a column is filtered directly.
- **ISCROSSFILTERED()**: Returns TRUE if a column is filtered by a cross-filter.
- **KEEPFILTERS()**: Modifies how filters are applied during the evaluation of a CALCULATE function.
- **REMOVEFILTERS([<table_name> | <column_name>])**: Clears filters from a table or column.

V. Logical Functions

These functions perform logical tests and return TRUE or FALSE.

- **AND(<logical_test1>, <logical_test2>)**: Returns TRUE if both arguments are TRUE.
- **FALSE()**: Returns the logical value FALSE.
- **IF(<logical_test>, <value_if_true>, <value_if_false>)**: Returns one value if a condition is TRUE, another if it's FALSE.
- **IFERROR(, <value_if_error>)**: Returns a specified value if an expression

evaluates to an error.
- **NOT(<logical_test>)**: Reverses the logical value of its argument.
- **OR(<logical_test1>, <logical_test2>)**: Returns TRUE if either argument is TRUE.
- **SWITCH(, , , [, , ...],)**: Evaluates an expression against a list of values and returns one of several possible result expressions.
- **TRUE()**: Returns the logical value TRUE.
- **ISBLANK()**: Returns TRUE if a value is blank.
- **ISERROR()**: Returns TRUE if a value is an error.
- **ISLOGICAL()**: Returns TRUE if a value is a logical value.
- **ISNUMBER()**: Returns TRUE if a value is a number.
- **ISTEXT()**: Returns TRUE if a value is text.

VI. Information Functions

These functions provide information about the data model and context.

- **COLUMNSTATISTICS()**: Returns statistics about columns in a table.
- **CONTAINS(<table_name>, <column_name>,)**: Returns TRUE if a value is found in a column of a table.
- **CONTAINSROW(<table_name>, <column_name>, , ...)**: Returns TRUE if a specified row exists in a table.
- **CUSTOMDATA()**: Returns the value of the Custom Data property.
- **LOOKUPVALUE(<result_column>, <search_column>, <search_value>, ...)**: Returns a value from a column based on a search in another column.
- **PATH(<id_column_name>, <parent_column_name>)**: Returns a delimited text string with the identifiers of all the parents of the current row.
- **PATHCONTAINS(<path_column_name>,)**: Returns TRUE if the specified item exists within a path.
- **PATHITEM(<path_column_name>, , [<data_type>])**: Returns the nth item from a path.
- **PATHLENGTH(<path_column_name>)**: Returns the number of items in a path.
- **SELECTEDVALUE(, [<alternate_result>])**: Returns the value when the context for the column has been filtered to one value. Otherwise returns the alternate result.
- **USERNAME()**: Returns the Windows user name of the current user.
- **USERPRINCIPALNAME()**: Returns the user principal name (UPN) of the current user.

VII. Mathematical Functions

These functions perform mathematical calculations.

- **ABS()**: Returns the absolute value of a number.
- **ACOS()**: Returns the arccosine of a number.
- **ACOSH()**: Returns the inverse hyperbolic cosine of a number.
- **ASIN()**: Returns the arcsine of a number.
- **ASINH()**: Returns the inverse hyperbolic sine of a number.
- **ATAN()**: Returns the arctangent of a number.
- **ATAN2(,)**: Returns the arctangent of y/x.
- **ATANH()**: Returns the inverse hyperbolic tangent of a number.
- **COS()**: Returns the cosine of an angle.
- **COSH()**: Returns the hyperbolic cosine of a number.
- **COT()**: Returns the cotangent of an angle.
- **COTH()**: Returns the hyperbolic cotangent of a number.
- **DEGREES()**: Converts radians to degrees.
- **DIVIDE(, , [<alternate_result>])**: Performs division, but handles cases where the denominator is zero.
- **EXP()**: Returns e raised to the power of a number.
- **FACT()**: Returns the factorial of a number.
- **FLOOR(,)**: Rounds a number down to the nearest multiple of significance.
- **INT()**: Rounds a number down to the nearest integer