MATH and TRIG FUNCTIONS OVERVIEW

When do we go for math and trigonometry functions?

ABS

Returns the absolute value of a number.

Syntax

ABS(<NUMBER>)

Example
The following example returns the absolute value of the difference between the list price and the dealer price, which you might use in a new calculated column, **DealerMarkup**.

=ABS([DealerPrice]-[ListPrice])

ACOS

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.

Syntax

ACOS(number)

Example

FORMULA	DESCRIPTION	RESULT
=ACOS(-0.5)	Arccosine of -0.5 in radians, 2*pi/3.	2.094395102
=ACOS(-0.5)*180/PI()	Arccosine of -0.5 in degrees.	120

ACOSH

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 *number*, so ACOSH(COSH(number)) equals number.

Syntax

ACOSH(number)

Example

FORMULA	DESCRIPTION	RESULT
=ACOSH(1)	Inverse hyperbolic cosine of 1.	0
=ACOSH(10)	Inverse hyperbolic cosine of 10.	2.993228

ASIN

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.

Syntax

ASIN(number)

Example

FORMULA	DESCRIPTION	RESULT
=ASIN(-0.5)	Arcsine of -0.5 in radians, -pi/6	-0.523598776
=ASIN(-0.5)*180/PI()	Arcsine of -0.5 in degrees	-30
=DEGREES(ASIN(-0.5))	Arcsine of -0.5 in degrees	-30

ASINH

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*.

Syntax

ASINH(number)

ATAN

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.

Syntax

ATAN(number)

Example

FORMULA	DESCRIPTION	RESULT
=ATAN(1)	Arctangent of 1 in radians, pi/4	0.785398163
=ATAN(1)*180/PI()	Arctangent of 1 in degrees	45

ATANH

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*.

Syntax

ATANH(number)

Example

FORMULA	DESCRIPTION	RESULT
=ATANH(0.76159416)	Inverse hyperbolic tangent of 0.76159416	1.00000001
=ATANH(-0.1)		-0.100335348

CEILING

Rounds a number up, to the nearest integer or to the nearest multiple of significance.

Syntax

CEILING(<number>, <significance>)

Example

The following formula returns 4.45. This might be useful if you want to avoid using smaller units in your pricing. If an existing product is priced at $4.42, you can use CEILING to round prices up to the nearest unit of five cents.

=CEILING(4.42,0.05)

Example

The following formula returns similar results as the previous example, but uses numeric values stored in the column, **ProductPrice**.

=CEILING([ProductPrice],0.05)

COMBIN

Returns the number of combinations for a given number of items. Use COMBIN to determine the total possible number of groups for a given number of items.

Syntax

COMBIN(number, number_chosen)

Q: Difference between COMBIN and COMBINA?

Example

FORMULA	DESCRIPTION	RESULT
=COMBIN(8,2)	Possible two-person teams that can be formed from 8 candidates.	28

COMBINA

Returns the number of combinations (with repetitions) for a given number of items.

Syntax

COMBINA(number, number_chosen)

Example

FORMULA	DESCRIPTION	RESULT
=COMBINA(4,3)	Returns the number of combinations (with repetitions) for 4 and 3.	20
=COMBINA(10,3)	Returns the number of combinations (with repetitions) for 10 and 3.	220

COS

Returns the cosine of the given angle.

Syntax

COS(<NUMBER>)

Example

FORMULA	DESCRIPTION	RESULT
=COS(1.047)	Cosine of 1.047 radians	0.5001711

=COS(60*PI())/180)	Cosine of 60 degrees	0.5
=COS(RADIANS(60))	Cosine of 60 degrees	0.5

COSH

Returns the hyperbolic cosine of a number
Syntax
COSH(<NUMBER>)
Example

FORMULA	DESCRIPTION	RESULT
=COSH(4)	Hyperbolic cosine of 4	27.308233
=COSH(EXP(1))	Hyperbolic cosine of the base of the natural logarithm.	7.6101251

CURRENCY

Evaluates the argument and returns the result as currency data type.
Syntax
CURRENCY(<value>)
Example
Convert number 1234.56 to currency data type.
=CURRENCY(1234.56)
Returns the value $1234.5600.

DEGREES

Converts radians into degrees.
Syntax
DEGREES(angle)
Example

FORMULA	DESCRIPTION	RESULT
=DEGREES(PI())	Degrees of pi radians	180

DIVIDE

Performs division and returns alternate result or BLANK() on division by 0.
Syntax
DIVIDE(<numerator>, <denominator> [,<alternateresult>])
Example
The following example returns 2.5.
=DIVIDE(5,2)

Result difference (5/0) in Power BI Query and DAX?

Example
The following example returns BLANK.
=DIVIDE(5,0)

Example

The following example returns 1.

=DIVIDE(5,0,1)

EVEN

Returns number rounded up to the nearest even integer. You can use this function for processing items that come in twos. For example, a packing crate accepts rows of one or two items. The crate is full when the number of items, rounded up to the nearest two, matches the crate's capacity.

Syntax

EVEN(number)

Q: Differences between EVEN and ISEVEN?

Example

FORMULA	DESCRIPTION	RESULT
=EVEN(1.5)	Rounds 1.5 to the nearest even integer	2
=EVEN(3)	Rounds 3 to the nearest even integer	4
=EVEN(2)	Rounds 2 to the nearest even integer	2
=EVEN(-1)	Rounds -1 to the nearest even integer	-2

EXP

Returns e raised to the power of a given number. The constant e equals 2.71828182845904, the base of the natural logarithm.

Syntax

EXP(<number>)

Example

The following formula calculates e raised to the power of the number contained in the column, [Power] .

=EXP([Power])

FACT

Returns the factorial of a number, equal to the series 1*2*3*...* , ending in the given number.

Syntax

FACT(<number>)

Example

The following formula returns the factorial for the series of integers in the column, [Values] .

=FACT([Values])

FLOOR

Rounds a number down, toward zero, to the nearest multiple of significance.

Syntax

FLOOR(<number>, <significance>)

Example

The following formula takes the values in the [Total Product Cost] column from the table, InternetSales.and rounds down to the nearest multiple of .1.

=FLOOR(InternetSales[Total Product Cost],.5)

GCD

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.

Syntax

GCD(number1, [number2], ...)

Example

FORMULA	DESCRIPTION	RESULT
=GCD(5, 2)	Greatest common divisor of 5 and 2.	1
=GCD(24, 36)	Greatest common divisor of 24 and 36.	12

INT

Rounds a number down to the nearest integer.

Syntax

INT(<number>)

Example

The following expression rounds the value to 1. If you use the ROUND function, the result would be 2.

=INT(1.5)

ISO.CEILING

Rounds a number up, to the nearest integer or to the nearest multiple of significance.

Syntax

ISO.CEILING(<number>[, <significance>])

Differneces between Ceiling, Floor and ISO.Ceiling and ISO.Floor?

Example: Positive Numbers

Description
The following formula returns 4.45. This might be useful if you want to avoid using smaller units in your pricing. If an existing product is priced at $4.42, you can use ISO.CEILING to round prices up to the nearest unit of five cents.

Code

=ISO.CEILING(4.42,0.05)

Example: Negative Numbers

The following formula returns the ISO ceiling value of -4.40.

Code

=ISO.CEILING(-4.42,0.05)

LCM

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. Use LCM to add fractions with different denominators.

Syntax

LCM(number1, [number2], ...)

Example

FORMULA	DESCRIPTION	RESULT
=LCM(5, 2)	Least common multiple of 5 and 2.	10
=LCM(24, 36)	Least common multiple of 24 and 36.	72

LN

Returns the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).

Syntax

LN(<number>)

Example

The following example returns the natural logarithm of the number in the column, [Values] .

=LN([Values])

LOG

Returns the logarithm of a number to the base you specify.

Syntax

LOG(<number>,<base>)

Example

The following formulas return the same result, 2.

=LOG(100,10)

=LOG(100)

LOG10

Returns the base-10 logarithm of a number

Syntax

LOG10(<NUMBER>)

Example

The following formulas return the same result, 2:

=LOG(100,10)
=LOG(100)
=LOG10(100)

MOD

Returns the remainder after a number is divided by a divisor. The result always has the same sign as the divisor.

Syntax

MOD(<number>, <divisor>)

Example

The following formula returns 1, the remainder of 3 divided by 2.

=MOD(3,2)

Example

The following formula returns -1, the remainder of 3 divided by 2. Note that the sign is always the same as the sign of the divisor.

=MOD(-3,-2)

MROUND

Q: Difference between ROUND and MROUND?

Returns a number rounded to the desired multiple

Syntax

MROUND(<NUMBER>,<MULTIPLE>)

Return value

A decimal number.

Description

The following expression rounds 1.3 to the nearest multiple of .2. The expected result is 1.4.

Code

=MROUND(1.3,0.2)

Example: Negative Numbers

Description

The following expression rounds -10 to the nearest multiple of -3. The expected result is -9.

Code

=MROUND(-10,-3)

Example: Error

Description

The following expression returns an error, because the numbers have different signs.

Code

=MROUND(5,-2)

ODD

Returns number rounded uo to the nearest odd integer.

Syntax

ODD(NUMBER)

Example

FORMULA	DESCRIPTION	RESULT
=ODD(1.5)	Rounds 1.5 up to the nearest odd integer	**3**
=ODD(3)	Rounds 3 up to the nearest odd integer	**3**
=ODD(-1)	Rounds -1 up to the nearest odd integer	**-3**

PI

Returns the value of Pi, 3.14159265358979, accurate to 15 digits.

Syntax

PI()

Example

The following formula calculates the area of a circle given the radius in the column, [Radius] .

=PI()*([Radius]*2)

POWER

Returns the result of number raised to power.

Syntax

POWER(<NUMBER,<POWER>)

Return value

A decimal number.

Example

The following example returns 25.

=POWER(5,2)

PRODUCT

Returns the product of the numbers in a column.

To return the product of an expression evaluated for each row in a table, use PRODUCTX function (DAX).

Syntax

PRODUCT(<column>)

Example

The following computes the product of the AdjustedRates column in an Annuity table:

=PRODUCT(Annuity[AdjustedRates])

PRODUCTX

Returns the product of an expression evaluated for each row in a table.

To return the product of the numbers in a column, use PRODUCT function (DAX).

Syntax

PRODUCTX(<table>, <expression>)

Example

The following computes the future value of an investment:

= [PresentValue] * PRODUCTX(AnnuityPeriods, 1+[FixedInterestRate])

QUOTIENT

Performs division and returns only the integer portion of the division result. Use this function when you want to discard the remainder of division.

Syntax

QUOTIENT(<numerator>, <denominator>)

Example

The following formulas return the same result, 2.

=QUOTIENT(5,2)
=QUOTIENT(10/2,2)

RADIANS

Converts degrees to radians.
Syntax
RADIANS(angle)
Example

FORMULA	DESCRIPTION	RESULT
=RADIANS(270)	270 degrees as radians (4.712389 or 3π/2 radians)	4.712389

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.
Syntax
RAND()
Example

Differences between RAND and RANDBETWEEN?

To generate a random real number between two other numbers, you can use a formula like the following:

= RAND()*(int1-int2)+int1

RANDBETWEEN

Returns a random number in the range between two numbers you specify.
Syntax

RANDBETWEEN(<bottom>,<top>)
Example

The following formula returns a random number between 1 and 10.

=RANDBETWEEN(1,10)

ROUND

Rounds a number to the specified number of digits.

Syntax
ROUND(<NUMBER>,<NUM_DIGITS>)
Related functions

To always round up (away from zero), use the ROUNDUP function.
To always round down (toward zero), use the ROUNDDOWN function.
To round a number to a specific multiple (for example, to round to the nearest multiple of 0.5), use the MROUND function.

You can use the functions TRUNC and INT to obtain the integer portion of the number.
The following formula rounds 2.15 up, to one decimal place. The expected result is 2.2.
=ROUND(2.15,1)

Example Example
The following formula rounds 21.5 to one decimal place to the left of the decimal point. The expected result is 20.

=ROUND(21.5,-1)

ROUNDDOWN

Rounds a number down, toward zero.

Syntax

ROUNDDOWN(<number>, <num_digits>)

Related functions

ROUNDDOWN behaves like ROUND, except that it always rounds a number down. The INT function also rounds down, but with INT the result is always an integer, whereas with ROUNDDOWN you can control the precision of the result.

Example

The following example rounds 3.14159 down to three decimal places. The expected result is 3.141.

=ROUNDDOWN(3.14159,3)

Example

The following example rounds the value of 31415.92654 down to 2 decimal places to the left of the decimal. The expected result is 31400.

=ROUNDDOWN(31415.92654, -2)

ROUNDUP

Rounds a number up, away from 0 (zero).

Syntax

ROUNDUP(<number>, <num_digits>)

Example

The following formula rounds Pi to four decimal places. The expected result is 3.1416.

=ROUNDUP(PI(),4)

Example: Decimals as Second Argument

Description

The following formula rounds 1.3 to the nearest multiple of 0.2. The expected result is 2.

Code

=ROUNDUP(1.3,0.2)

Example: Negative Number as Second Argument

Description

The following formula rounds the value in the column, **FreightCost**, with the expected results shown in the following table:

Code

=ROUNDUP([Values],-1)

Comments

When **num_digits** is less than zero, the number of places to the left of the decimal sign is increased by the value you specify.

FREIGHTCOST	EXPECTED RESULT
13.25	20
2.45	10
25.56	30
1.34	10
345.01	350

SIGN

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.

Syntax

SIGN(<number>)

Parameters

TERM	DEFINITION
number	Any real number, a column that contains numbers, or an expression that evaluates to a number.

Return value

A whole number. The possible Return values are 1, 0, and -1.

RETURN VALUE	DESCRIPTION
1	The number is positive
0	The number is zero
-1	The number is negative

Example

The following formula returns the sign of the result of the expression that calculates sale price minus cost.

=SIGN(([Sale Price] - [Cost]))

SQRT

Return the square root of a number

Syntax

SQRT(<number>)

Example

The following formula returns 5.

=SQRT(25)

SUM

Differences between SUM and SUMX?

Adds all the numbers ina column.

Syntax

SUM(<COLUMN>)

Example

The following example adds all the numbers that are contained in the column, Amt, from the table, Sales.

=SUM(Sales[Amt])

SUMX

Returns the sum of an expression evaluated for each row in a table.

Syntax

SUMX(<table>, <expression>)

Example

The following example first filters the table, InternetSales, on the expression, ShippingTerritoryID = 5 , and then returns the sum of all values in the column, Freight. In other words, the expression returns the sum of freight charges for only the specified sales area.

=SUMX(FILTER(InternetSales, InternetSales[SalesTerritoryID]=5),[Freight])

If you do not need to filter the column, use the SUM function. The SUM function is similar to the Excel function of the same name, except that it takes a column as a reference.

TRUNC

Truncates a number to an integer by removing the decimal, or fractional, part of the number.

Syntax

TRUNC(<number>,<num_digits>)

Example

The following formula returns 3, the integer part of pi.

=TRUNC(PI())

Example

The following formula returns -8, the integer part of -8.9.

=TRUNC(-8.9)

OTHER FUNCTIONS OVERVIEW

When do we go for Othe Functions in DAX?

These functions perform unique actions that cannot be defined by any of the categories.

DATATABLE	GENERATESERIES	ISEMPTY	SUMMARIZECOLUMNS
ERROR	GROUPBY	NATURALINNERJOIN	TREATAS
EXCEPT	INTERSECT	NATURALLEFTOUTERJOIN	UNION
			VAR

DATATABLE

How many ways we can construct a table?

Provides a mechanism for declaring an inline set of data values.

Syntax

DATATABLE (ColumnName1, DataType1, ColumnName2, DataType2..., {{Value1, Value2...}, {ValueN, ValueN+1...}...})

Example


```
=DataTable("Name", STRING,
           "Region", STRING
           ,{
                    {" User1","East"},
                    {" User2","East"},
                    {" User3","West"},
                    {" User4","West"},
                    {" User4","East"}
            }
      )
```

ERROR

Difference between Error and IFError?

Raises an error with an error message.

Syntax

Example 1

The following DAX query:

```
DEFINE
MEASURE DimProduct[Measure] =
  IF(
   SELECTEDVALUE(DimProduct[Color]) = "Red",
   ERROR("red color encountered"),
   SELECTEDVALUE(DimProduct[Color])
  )
EVALUATE SUMMARIZECOLUMNS(DimProduct[Color], "Measure", [Measure])
```

```
ORDER BY [Color]
```

Fails and raises and error message containing “red color encountered”.

Example 2

The following DAX query:

```
DEFINE
MEASURE DimProduct[Measure] =
   IF(
    SELECTEDVALUE(DimProduct[Color]) = "Magenta",
    ERROR("magenta color encountered"),
    SELECTEDVALUE(DimProduct[Color])
   )
EVALUATE SUMMARIZECOLUMNS(DimProduct[Color], "Measure", [Measure])
ORDER BY [Color]
```

Returns the following table:

DIMPRODUCT[COLOR]	[MEASURE]
Black	Black
Blue	Blue
Grey	Grey
Multi	Multi
NA	NA
Red	Red
Silver	Silver
Silver\Black	Silver\Black
White	White
Yellow	Yellow

Because Magenta is not one of the product colors, the ERROR function is not executed.

Scenario:

EXCEPT

How is it helpful in real-time?

Returns the rows of one table which do not appear in another table.

Syntax

EXCEPT(<table_expression1>, <table_expression2>)

Example

```
Table_Exclusive = except(DimCourses_new,DimCourse)
```

GENERATESERIES

What is the equivalent SQL instruction?

Returns a single column table containing the values of an arithmetic series, that is, a sequence of values in which each differs from the preceding by a constant quantity. The name of the column returned is Value.

Syntax

GENERATESERIES(<startValue>, <endValue>[, <incrementValue>])

Example 1

The following DAX query:

EVALUATE GENERATESERIES(1, 5)

Returns the following table with a single column:

Example 2

The following DAX query:

EVALUATE GENERATESERIES(1.2, 2.4, 0.4)

Returns the following table with a single column:

VALUE	VALUE
1.2	**1.2**
1.6	**1.6**
2	**2**
2.4	**2.4**

Example 3

The following DAX query:

EVALUATE GENERATESERIES(CURRENCY(10), CURRENCY(12.4), CURRENCY(0.5))

Returns the following table with a single column:

VALUE
10
10.5
11
11.5
12

Difference between GROUP BY and SUMMARIZE?

GROUP BY

The GROUPBY function is similar to the SUMMARIZE function. However, GROUPBY does not do an implicit CALCULATE for any extension columns that it adds. GROUPBY permits a new function, CURRENTGROUP(), to be used inside aggregation functions in the extension columns that it adds. GROUPBY attempts to reuse the data that has been grouped making it highly performant.

Syntax

GROUPBY (<table>, [<groupBy_columnName1>], [<name>, <expression>]…)

Example

Assume a data model has four tables: Sales, Customer, Product, Geography where Sales is on the “many” side of a relationship to each of the other three tables.

```
GROUPBY (
Sales,

Geography[Country],

Product[Category],

“Total Sales”, SUMX( CURRENTGROUP(), Sales[Price] * Sales[Qty])
```

)

This will start with the Sales table, extended with all the columns from all the related tables.Then it will build a result with three columns.

- The first column is each of the countries for which there is a sale.
- The second column is each product category for which there is a sale in that country.
- The third column is the sum of sales amount (as calculated from price*qty) for the selected country and product category.

Suppose we've built the previous result. We can use GROUPBY again, to find the maximum category sales figure within each country as shown here.

```
DEFINE
VAR SalesByCountryAndCategory =
GROUPBY (
Sales,
Geography[Country],
Product[Category],
"Total Sales", SUMX( CURRENTGROUP(), Sales[Price] * Sales[Qty])
)
Evaluate GROUPBY (
SalesByCountryAndCategory,
Geography[Country],
 "Max Sales", MAXX( CURRENTGROUP(), [Total Sales])
)
```

Scenario: Institute and Mode wise total discount fee

```
Inst_Mode_Total =
GROUPBY(FactPayments,FactPayments[InstituteID],FactPayments[ModeID],"Total Fee",SUMx(currentgroup(),FactPayments[Discount_Fee]))
```

INTERSECT

Returns the row intersection of two tables, retaining duplicates.

Syntax : INTERSECT(<table_expression1>, <table_expression2>)

```
Table_Common = Intersect(DimCourses_new,DimCourse)
```


ISEMPTY

Checks if a table is empty.

Syntax

ISEMPTY(<table_expression>)

Example

For the below table named 'Info':

COUNTRY	STATE	COUNTY	TOTAL
IND	JK	20	800
IND	MH	25	1000
IND	WB	10	900
USA	CA	5	500
USA	WA	10	900

```
EVALUATE
ROW("Any countries with count > 25?", NOT(ISEMPTY(FILTER(Info, [Count]>25))))
```

Return value: **FALSE**

NATURALINNERJOIN

Performs an inner join of a table with another table. 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.

Syntax

NATURALINNERJOIN(<leftJoinTable>, <rightJoinTable>)

For example,
Products[ProductID], WebSales[ProductdID], StoreSales[ProductdID] with many-to-one relationships between WebSales and StoreSales and the Products table based on the ProductID column, WebSales and StoreSales tables are joined on [ProductID].

Strict comparison semantics are used during join. There is no type coercion; for example, 1 does not equal 1.0.

NATURALLEFTOUTERJOIN

Performs an inner join of a table with another table. 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.

Syntax

NATURALLEFTOUTERJOIN(<leftJoinTable>, <rightJoinTable>)

For example,

Products[ProductID], WebSales[ProductdID], StoreSales[ProductdID] with many-to-one relationships between WebSales and StoreSales and the Products table based on the ProductID column, WebSales and StoreSales tables are joined on [ProductID].

Difference between Group By and SummarizeColumns?

SUMMARIZECOLUMNS

Strict comparison semantics are used during join. There is no type coercion; for example, 1 does not equal 1.0.
Returns a summary table over a set of groups.
Syntax

SUMMARIZECOLUMNS(<groupBy_columnName> [, < groupBy_columnName >]…, [<filterTable>]…[, <name>, <expression>]…)

Remarks

SUMMARIZECOLUMNS does not guarantee any sort order for the results.
A column cannot be specified more than once in the groupBy_columnName parameter. For example, the following formulas are invalid.

SUMMARIZECOLUMNS(Sales[StoreId], Sales[StoreId])

Filter context

Consider the following query:

SUMMARIZECOLUMNS ('Sales Territory'[Category], FILTER('Customer', 'Customer' [First Name] = “Alicia”))

In this query, without a measure the groupBy columns do not contain any columns from the Filter expression (i.e.from Customer table). The filter is not applied to the groupBy columns. The Sales Territory and the Customer table may be indirectly related through the Reseller sales fact table. Since they're not directly related, the filter expression is a no-op and the groupBy columns are not impacted.

However, with this query:

SUMMARIZECOLUMNS ('Sales Territory'[Category], 'Customer' [Education], FILTER('Customer', 'Customer'[First Name] = “Alicia”))

The groupBy columns contain a column which is impacted by the filter and that filter is applied to the groupBy results.

Scenario: InstituteID and ModeID wise total discount fee

Inst_Mode_Total = GROUPBY(FactPayments,FactPayments[InstituteID],FactPayments[ModeID],"Total Fee",SUMx(**currentgroup(),**FactPayments[Discount_Fee]))

Inst_Mode_Total1 = SUMMARIZECOLUMNS(FactPayments[InstituteID],FactPayments[ModeID],FactPayments,"Total",sum(Fact Payments[Discount_Fee]))

IGNORE

The IGNORE() syntax can be used to modify the behavior of the SUMMARIZECOLUMNS function by omitting specific expressions from the BLANK/NULL evaluation. Rows for which all expressions not using IGNORE return BLANK/NULL will be excluded independent of whether the expressions which do use IGNORE evaluate to BLANK/NULL or not.

Syntax

IGNORE(<expression>)

With SUMMARIZECOLUMNS

SUMMARIZECOLUMNS(<groupBy_columnName>[, < groupBy_columnName >]…, [<filterTable>]…[, <name>, IGNORE(…)]…)

Example

SUMMARIZECOLUMNS(Sales[CustomerId], "Total Qty", IGNORE(SUM(Sales[Qty])), "BlankIfTotalQtyIsNot3", IF(SUM(Sales[Qty])=3, 3))

This rolls up the Sales[CustomerId] column, creating a subtotal for all customers in the given grouping. Without IGNORE, the result is:

CUSTOMERID	TOTALQTY	BLANKIFTOTALQTYISNOT3
A	5	
B	3	3

CUSTOMERID	TOTALQTY	BLANKIFTOTALQTYISNOT3
C	3	3

With IGNORE

CUSTOMERID	TOTALQTY	BLANKIFTOTALQTYISNOT3
B	3	3
C	3	3

All expression ignored

SUMMARIZECOLUMNS(Sales[CustomerId], "Blank", IGNORE(Blank()), "BlankIfTotalQtyIsNot5", IGNORE(IF(SUM(Sales[Qty])=5, 5)))

Even though both expressions return blank for some rows, they're included since there are no non-ignored expressions which return blank.

CUSTOMERID	TOTALQTY	BLANKIFTOTALQTYISNOT3
A		5
B		
C		

NONVISUAL

Marks a value filter in SUMMARIZECOLUMNS function as not affecting measure values, but only applying to group-by columns.

Syntax

NONVISUAL(<expression>)

Return value

A table of values.

Example

```
DEFINE

MEASURE FactInternetSales[Sales] = SUM(FactInternetSales[Sales Amount])
EVALUATE
SUMMARIZECOLUMNS
(
 DimDate[CalendarYear],
 NONVISUAL(TREATAS({2007, 2008}, DimDate[CalendarYear])),
 "Sales", [Sales],
 "Visual Total Sales", CALCULATE([Sales], ALLSELECTED(DimDate[CalendarYear]))
)

ORDER BY [CalendarYear]
```


Result

Returns the result where [Visual Total Sales] is the total across all years:

DIMDATE[CALENDARYEAR]	[SALES]	[VISUAL TOTAL SALES]
2007	9,791,060.30	29,358,677.22
2008	9,770,899.74	29,358,677.22

In contrast, the same query without the NONVISUAL function:

```
DEFINE
MEASURE FactInternetSales[Sales] = SUM(FactInternetSales[Sales Amount])
EVALUATE
SUMMARIZECOLUMNS
(
 DimDate[CalendarYear],
 TREATAS({2007, 2008}, DimDate[CalendarYear]),
 "Sales", [Sales],
 "Visual Total Sales", CALCULATE([Sales], ALLSELECTED(DimDate[CalendarYear]))
)
ORDER BY [CalendarYear]
```

Result

Returns the result where [Visual Total Sales] is the total across the two selected years:

DIMDATE[CALENDARYEAR]	[SALES]	[VISUAL TOTAL SALES]
2007	9,791,060.30	19,561,960.04
2008	9,770,899.74	19,561,960.04

ROLLUPADDISSUBTOTAL

The addition of the ROLLUPADDISSUBTOTAL() syntax modifies the behavior of the SUMMARIZECOUMNS function by adding roll-up/subtotal rows to the result based on the columns.

Syntax

ROLLUPADDISSUBTOTAL ([<filter> …,] <groupBy_columnName>,
<isSubtotal_columnName>[, <filter> …][, <groupBy_columnName >,
<isSubtotal_columnName>[, <filter> …]…])

Example

Single subtotal

```
DEFINE
VAR vCategoryFilter =
  TREATAS({"Accessories", "Clothing"}, Product[Category])
VAR vSubcategoryFilter =
  TREATAS({"Bike Racks", "Mountain Bikes"}, Product[Subcategory])
EVALUATE
  SUMMARIZECOLUMNS
  (
    ROLLUPADDISSUBTOTAL
    (
      Product[Category], "IsCategorySubtotal", vCategoryFilter,
      Product[Subcategory], "IsSubcategorySubtotal",
      vSubcategoryFilter
    ),
    "Total Qty", SUM(Sales[Qty])
  )
  ORDER BY
  [IsCategorySubtotal] DESC, [Category],
  [IsSubcategorySubtotal] DESC, [Subcategory]
```


ROLLUPGROUP

Like with the SUMMARIZE function, ROLLUPGROUP can be used together with ROLLUPADDISSUBTOTAL to specify which summary groups/granularities (subtotals) to include (reducing the number of subtotal rows returned).

Syntax

ROLLUPGROUP(<groupBy_columnName>, <groupBy_columnName>)

With ROLLUPADDISSUBTOTAL

ROLLUPADDISSUBTOTAL(ROLLUPGROUP(…), isSubtotal_columnName[, <groupBy_columnName>…])

Example

Multiple subtotals

SUMMARIZECOLUMNS(ROLLUPADDISSUBTOTAL(Sales[CustomerId], "IsCustomerSubtotal"),
ROLLUPADDISSUBTOTAL(ROLLUPGROUP(Regions[City], Regions[State]),
“IsCityStateSubtotal”),"Total Qty", SUM(Sales[Qty]))

Still grouped by City and State, but rolled together when reporting a subtotal.

EVALUATE

When do we use evaluate in Real-time?

Returns a table of one or more columns.

Syntax

{ <scalarExpr1>, <scalarExpr2>, … }

{ (<scalarExpr1>, <scalarExpr2>, …), (<scalarExpr1>, <scalarExpr2>, …), … }

Example 1

The following DAX queries:

EVALUATE { 1, 2, 3 }

and

EVALUATE { (1), (2), (3) }

Return the following table of a single column:

Example 2

The following DAX query:

```
EVALUATE
 {
  (1.5, DATE(2017, 1, 1), CURRENCY(199.99), "A"),
  (2.5, DATE(2017, 1, 2), CURRENCY(249.99), "B"),
  (3.5, DATE(2017, 1, 3), CURRENCY(299.99), "C")
 }
```


[VALUE1]	[VALUE2]	[VALUE3]	[VALUE4]
1.5	1/1/2017	199.99	A
Row2	1/2/2017	249.99	B
Row3	1/3/2017	299.99	C

Example 3

The following DAX query

EVALUATE { 1, DATE(2017, 1, 1), TRUE, "A" }

Returns the following table of a single column of String data type:

[VALUE]
1
1/1/2017
TRUE
A

When do we use this in real-time?

TREATAS

Applies the result of a table expression as filters to columns from an unrelated table.

Syntax

TREATAS(table_expression, <column>[, <column>[, <column>[,…]]]})

Examples

In the following example, the model contains two unrelated product tables. If a user applies a filter to DimProduct1[ProductCategory] selecting Bikes, Seats, Tires, the same filter, Bikes, Seats, Tires is applied to DimProduct2[ProductCategory].

```
CALCULATE(
SUM(Sales[Amount]),
TREATAS(VALUES(DimProduct1[ProductCategory]), DimProduct2[ProductCategory])

)
```

Scenario: Finding the total from DimCourse and DimCourses_new based on the selection from DimCourse query

```
Multitblsum = calculate(sum(FactPayments[Discount_Fee]),
treatas(values(DimCourse[CourseID]),DimCourses_new[CourseID]))
```

UNION

Creates a union (join) table from a pair of tables.

Syntax

UNION(<table_expression1>, <table_expression2> [,<table_expression>]…)

Example

The following expression creates a union by combining the USAInventory table and the INDInventory table into a single table:

UNION(UsaInventory, IndInventory)

```
Table_Merge = union(DimCourses_new,DimCourse)
```

VAR

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.

Syntax

VAR <name> = <expression>

The usage of Sum in Real-time?

SUM

To calculate a percentage of year-over-year growth without using a variable, you could create three separate measures. This first measure calculates Sum of Sales Amount:

Sum of SalesAmount = SUM(SalesTable[SalesAmount])

A second measure calculates the sales amount for the previous year:

SalesAmount PreviousYear=CALCULATE([Sum of SalesAmount], SAMEPERIODLASTYEAR(Calendar[Date]))

You can then create a third measure that combines the other two measures to calculate a growth percentage. Notice the Sum of SalesAmount measure is used in two places; first to determine if there is a sale, then again to calculate a percentage.

Sum of SalesAmount YoY%:=IF([Sum of SalesAmount] ,

DIVIDE(([Sum of SalesAmount] – [SalesAmount PreviousYear]), [Sum of PreviousYear]))

By using a variable, you can create a single measure that calculates the same result:

YoY% = var Sales = SUM(SalesTable[SalesAmount])

var SalesLastYear=CALCULATE(Sales, SAMEPERIODLASTYEAR('Calendar'[Date]))

return if(Sales, DIVIDE(Sales – SalesLastYear, Sales))

By using a variable, you can get the same outcome, but in a more readable way. In addition, the result of the expression is stored in the variable upon declaration. It doesn't have to be recalculated each time it is used, as it would without using a variable. This can improve the measure's performance.