Simple | Smart | Speed
CHOOLS

Calculation Components


```
// This is Comment
IF [Choose Measure] = "Sales" THEN SUM([Sales]) END
```

Comment · Operator · Field
Logical Expression · Parameter · Literal Expression · Function
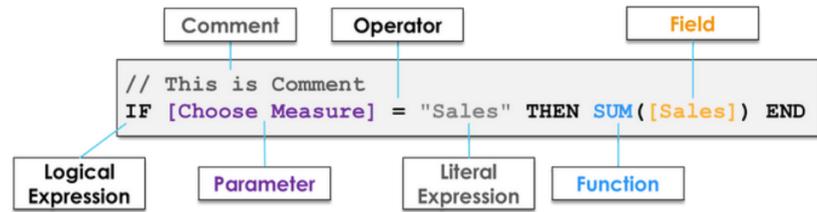
Calculation 4 Types

| Type	Description
Row-Level-Calculations	Perform calculations at the row level individually. Data will not be aggregated and out of calculation will be stored in data source
Aggregate Calculations	Aggregate the rows at the dimension level used in the VIZ
LOD Calculations	Aggregate the rows at the dimension level used in the calculation to control the level of details
Table Calculation	Performed after the execute of aggregate calculation. The calculations are performed on the data displayed in the visualization

| Row-Level Calculation	Aggregate Calculation	LOD Calculation	Table Calculation
[Quantity]*[Price]	SUM([Revenue])	{FIXED[Category]:SUM([Revenue])}	RANK(SUM([Revenue]))
Do Not Aggregate Data	Aggregate Data	Aggregate Data	Aggregate Data
Row Level	VIZ Level Of Details	Specific Level Of Details	VIZ Level Of Details
Calculated using Data in Data Source	Calculated using Data in Data Source	Calculated using Data in Data Source	Calculated using Data in VIZ
Pre-Calculated	Calculated in the Fly	Calculated in the Fly	Calculated in the Fly
Simple Calculations	Simple Calculations	Complex Calcuations	Complex Calcuations

Basic Components of Calculations

```
RANK(SUM([Quantity]*[Price]))
```
① Row-Level Calculation
② Aggregate Calculation
③ Table Calculation
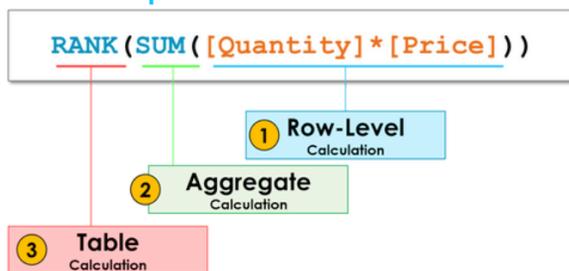
Number Functions

CEILING(1.2) = 2	Round up numbers
FLOOR(1.2) = 1	Round down numbers
ROUND(1.2) = 1	Round numbers to nearest integer

String Functions

- Change Case
  - LOWER
  - UPPER
- Remove Spaces
  - LTRIM
  - RTRIM
  - TRIM
- Extract
  - LEFT
  - RIGHT
  - MID
- Search for Patterns
  - STARTSWITH
  - ENDSWITH
  - CONTAINS
  - FIND
  - FINDNTH
- Combine & Split
  - CONCAT
  - SPLIT
- Replace
  - REPLACE
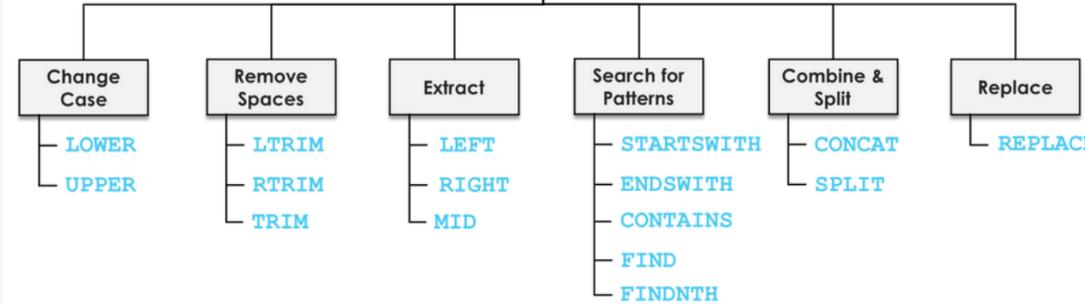
LOWER("Paris") = "paris"	Converts all characters to lowercase
UPPER("Paris") = "PARIS"	Converts all characters to uppercase
LTRIM(" Paris ") = "PARIS "	Removes any leading spaces
RTRIM(" Paris ") = " PARIS"	Removes any trailing spaces
TRIM(" Paris ") = "PARIS"	Removes both leading & trailing spaces
LEFT("Paris",2) = "Pa"	Extracts the left-most number of characters in string
RIGHT("Paris",2) = "is"	Extracts the right-most number of characters in string
MID("Paris",2,2) = "ar"	Extracts specified number of characters in string, starting at specified position
CONTAINS("Paris","ar") = true	Returns true if the given string contains a specified substring
STARTSWITH("Paris","Pa") = true	Returns true if string starts with substring.
ENDSWITH("Paris","ar") = true	Returns true if the given string ends with the specified substring
FIND("Baraa","a") = 2	Returns the position of substring in string
FINDNTH("Baraa","a",2) = 4	Returns the position of the nth occurrence of substring within the specified string
SPLIT("a-b","-",2) = b	Returns a substring from a string based on specified delimiter and
REPLACE("a-b","-","+") = b	Replaces occurrences of a specified substring with another substring within a string

Date Functions

DATEPART('month',#2025-08-20#) = 8	Extracts a specific part of date as an integer
DATENAME('month',#2025-08-20#) = "August"	Extracts a specific part of date as a string
MONTH(#2025-08-20#) = 8	Extracts the month of a given date as an integer
YEAR(#2025-08-20#) = 2025	Extracts the year of a given date as an integer
DAY(#2025-08-20#) = 25	Extracts the day of a given date as an integer
DATETRUNC('month',#2025-08-20#) = 2025-08-01	Truncates a date or time to a specified level of precision
DATEADD('month',3,#2025-08-20#) = 2025-11-20	Adds an increment to specified date and returns
DATEDIFF('month',#2025-11-25#,#2026-02-01#) = 3	Returns the difference between two dates
TODAY() = 2024-08-20	Returns the current date
NOW() = 2024-08-20 1:08:21 PM	Returns the current date and time

NULL Functions

ZN(NULL) 0	Converts NULL to Zero
IFNULL(NULL,1) 1	Converts NULL to the specified value
ISNULL(NULL) = true	Return true if value is NULL, and false otherwise

Logical Calculations

Logical Conditions

`IF [Sales] >1200 THEN "High" END`	Classifies Sales as "High" if greater than 1200, and NULL otherwise
`IF [Sales] >1200 THEN "High" ELSE "LOW" END`	Classifies Sales as "High" if greater than 1200, and "Low" otherwise
`IF [Sales] >1200 THEN "High" ELSEIF [Sales] >500 THEN "Medium" ELSE "LOW" END`	Classifies Sales as "High" if greater than 1200, "Medium" if between 500 and 1200, and "Low" otherwise
`IIF ([Sales] >1200,"High","Low")`	Classifies Sales as "High" if greater than 1200, and "Low" otherwise
`CASE [Country] WHEN "Germany" THEN"DE" WHEN "USA" THEN "US" ELSE "n/a" END`	Assigns country codes "DE" for Germany, "US" for USA, and "n/a" for other countries

Logical Operators

`IF [Sales] > 1200 OR [Country] = "Germany" THEN "High" END`	Classifies Sales as "High" if greater than 1200 or if the country is Germany, and NULL otherwise
`IF [Sales] > 1200 OR [Country] = "Germany" THEN "High" END`	Classifies Sales as "High" if greater than 1200 and if the country is Germany, and NULL otherwise

Aggregate Calculations

SUM([Sales])	Returns the total sum of all values
AVG([Sales])	Returns the average of all values
MAX([Sales])	Returns the maximum values
MIN([Sales])	Returns the minimum value
COUNT([ID])	Counts the number of values
COUNTD([ID])	Counts the number of unique values
ATTR([Customer])	If all values are same, then it returns single value, otherweise Asterisk *

LOD Calculations

{ FIXED[Category] : SUM([Sales]) }	Sums the sales using only category, ignoring other dimensions in the view
{ EXCLUDE[Category] : SUM([Sales]) }	Sums the sales using view dimensions and excluding category if present in the view
{ INCLUDE[Customer] : SUM([Sales]) }	Sums the sales using not only view dimensions but also includes the dimenion customer

Table Calculations

FIRST()	Returns the number of rows from current row to first row in partition
LAST()	Returns the number of rows from current row to last row in partition
INDEX()	Returns the index of the current row in the partition
RANK(SUM([Sales]),	Ranks the total sales in descending order, assigning a rank to each row
RUNNING_SUM(SUM([Sales]))	Calculates the running sum of the total sales, providing a cumulative sum as moving