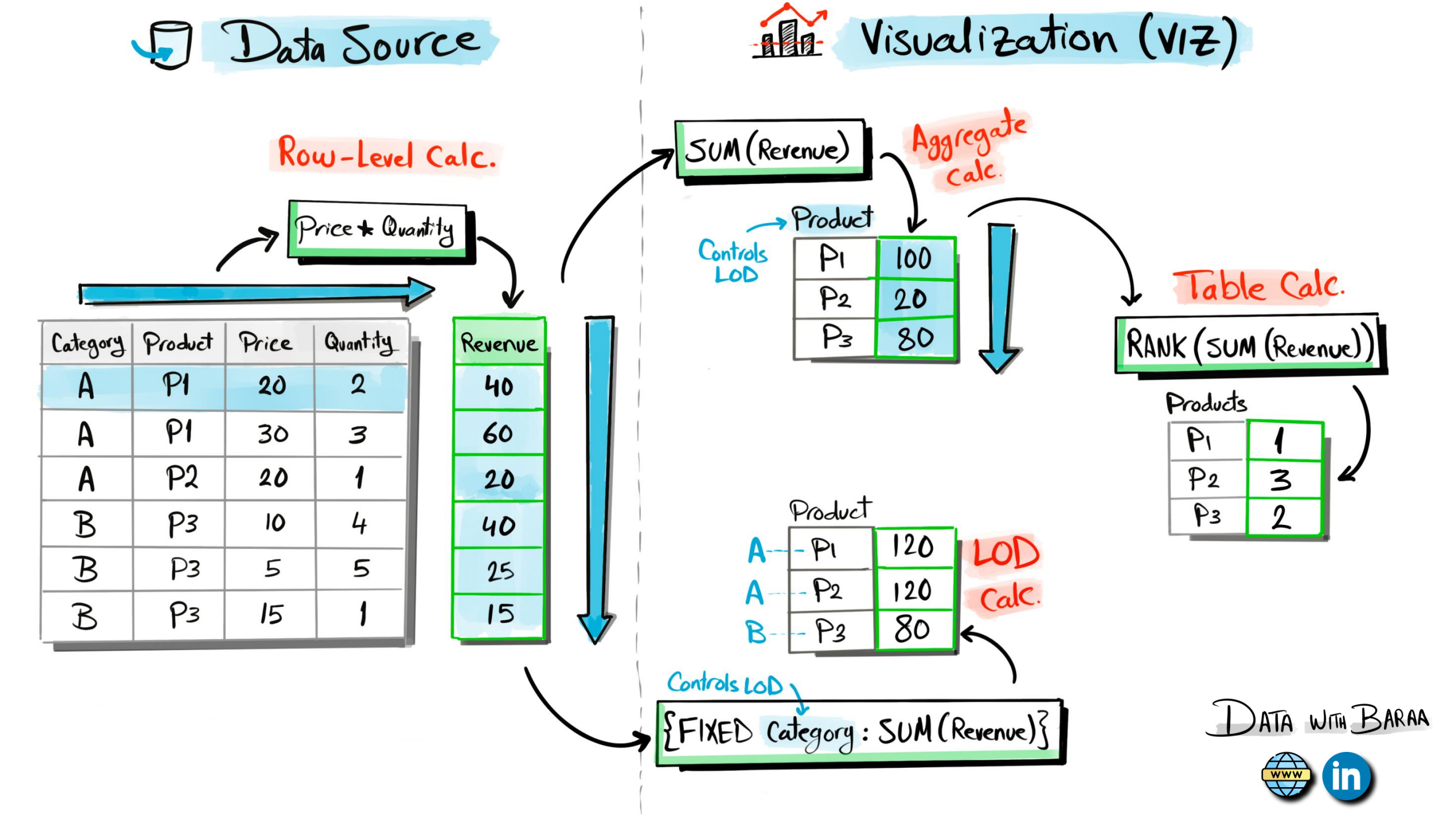
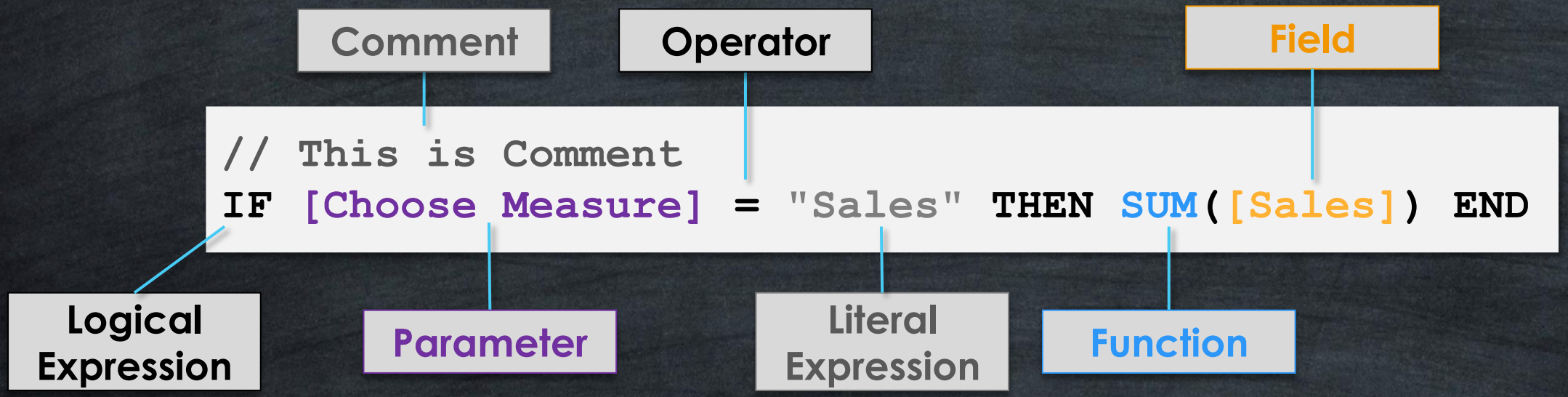

Data Source
Visualization (VIZ)
Row-Level Calc.
Price * Quantity
Category | Product | Price | Quantity
A | P1 | 20 | 2
A | P1 | 30 | 3
A | P2 | 20 | 1
B | P3 | 10 | 4
B | P3 | 5 | 5
B | P3 | 15 | 1
Revenue
40
60
20
40
25
15
SUM (Revenue)
Aggregate Calc.
Controls LOD
Product
P1 | 100
P2 | 20
P3 | 80
Table Calc.
RANK (SUM (Revenue))
Products
P1 | 1
P2 | 3
P3 | 2
Product
A P1 | 120
A P2 | 120
B P3 | 80
LOD Calc.
Controls LOD
{FIXED Category : SUM (Revenue)}
DATA WITH BARAA
WWW
in

# Basic Components of Calculations





# Order of Computation



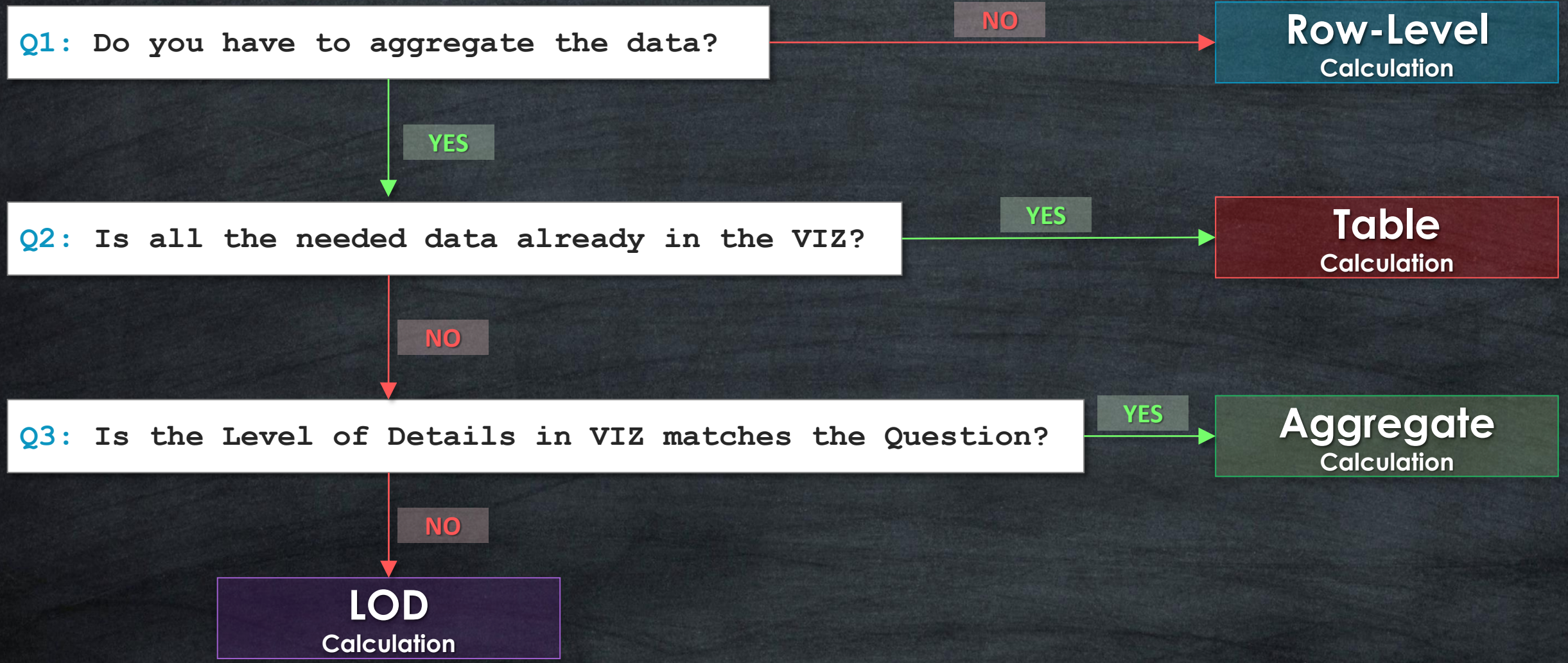
```
RANK(SUM([Quantity]*[Price]))
```

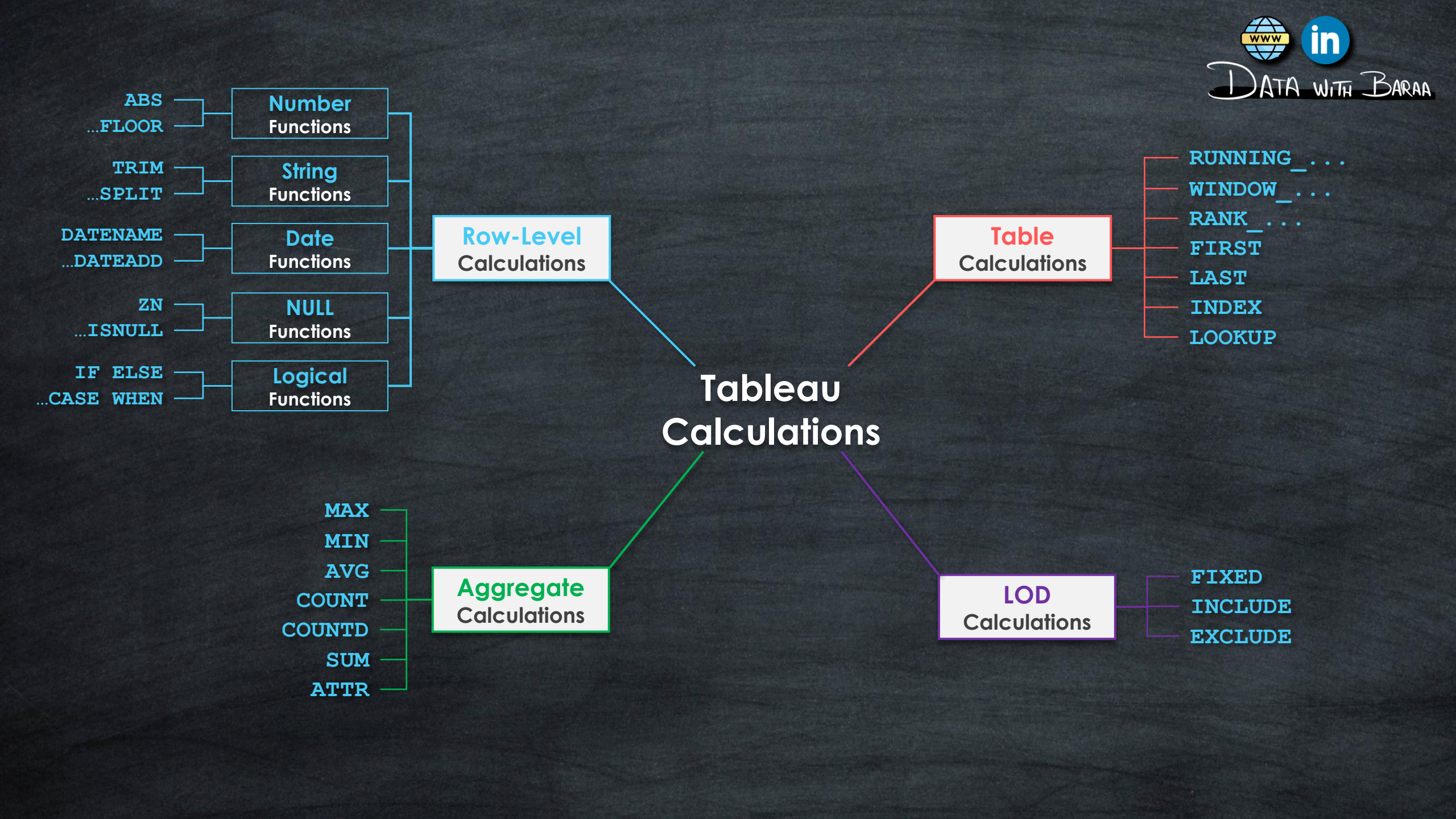
1. **Row-Level** Calculation
2. **Aggregate** Calculation
3. **Table** Calculation

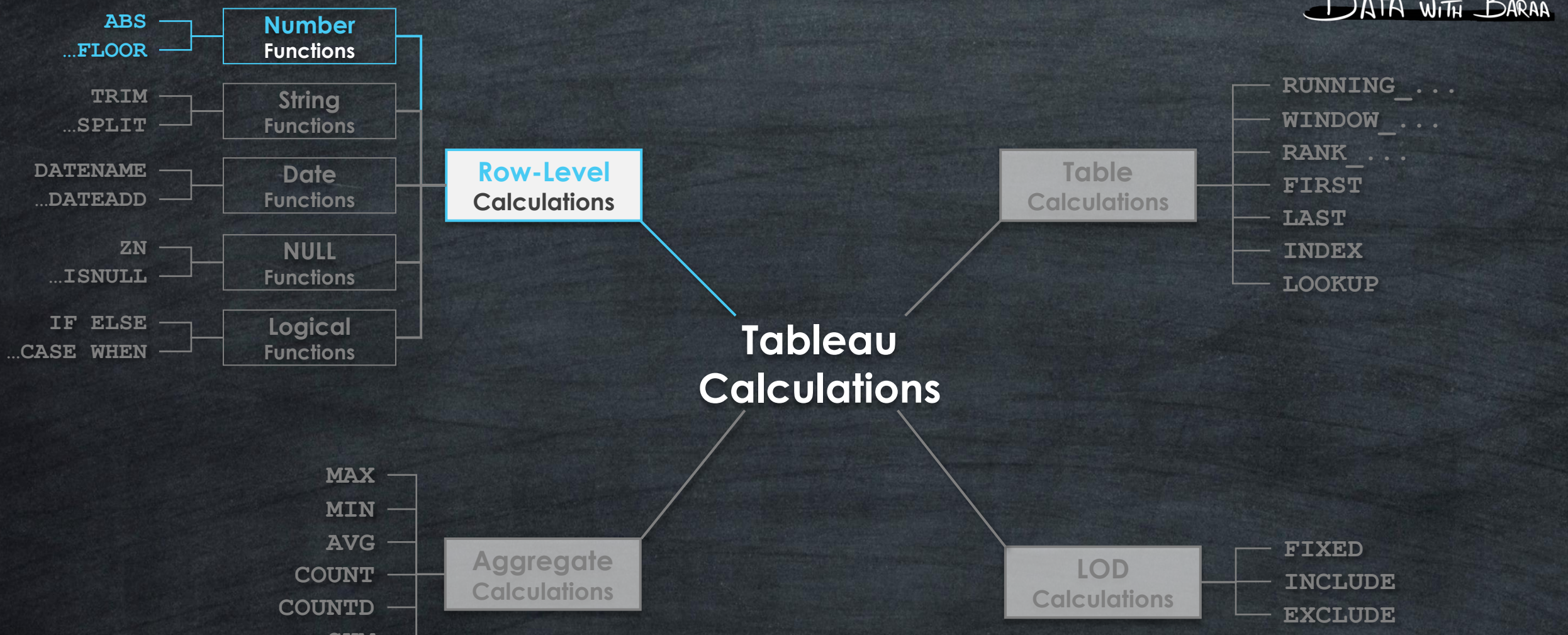
| Row-Level Calculation | Aggregate Calculation | LOD Calculation | Table Calculation |
| --- | --- | --- | --- |
| `[Quantity]*[Price]` | `SUM([Revenue])` | `{FIXED [Category]:SUM([Revenue])}` | `RANK(SUM([Revenue]))` |
| Do Not Aggregate Data | Aggregate Data | Aggregate Data | Aggregate Data |
| Row 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 Calcuations | Simple Calcuations | Complex Calcuations | Complex Calcuations |

Calculation Intro

# Choose the Right Calculation Type



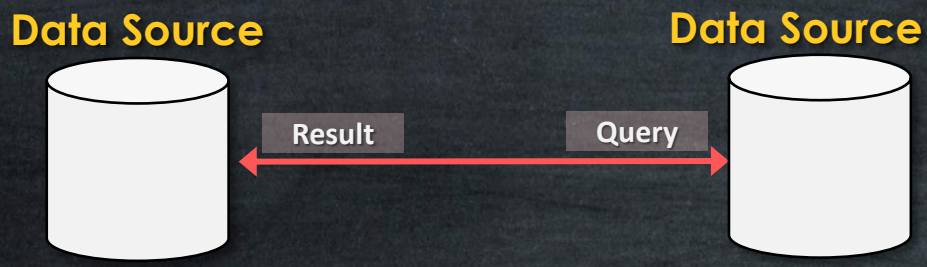

DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# Row-Level
## Calculations



- Perform calculations at the row level individually
- Level of Details is the Data Source Rows
- Data will not be aggregated
- The calculations are performed on the data within the data source
- Calculation results will be stored in Data source and will not be calculated on the FLY



# Number Functions
Use Cases



## Main Purpose to is mainuplate Numerical Values

## Simplify the numbers

- Functions to round the numbers to simplier form – `CEILING`, `FLOOR`, `ROUND`

NUMBER FUNCTIONS

# CEILING, FLOOR, ROUND

DATA WITH BARAA

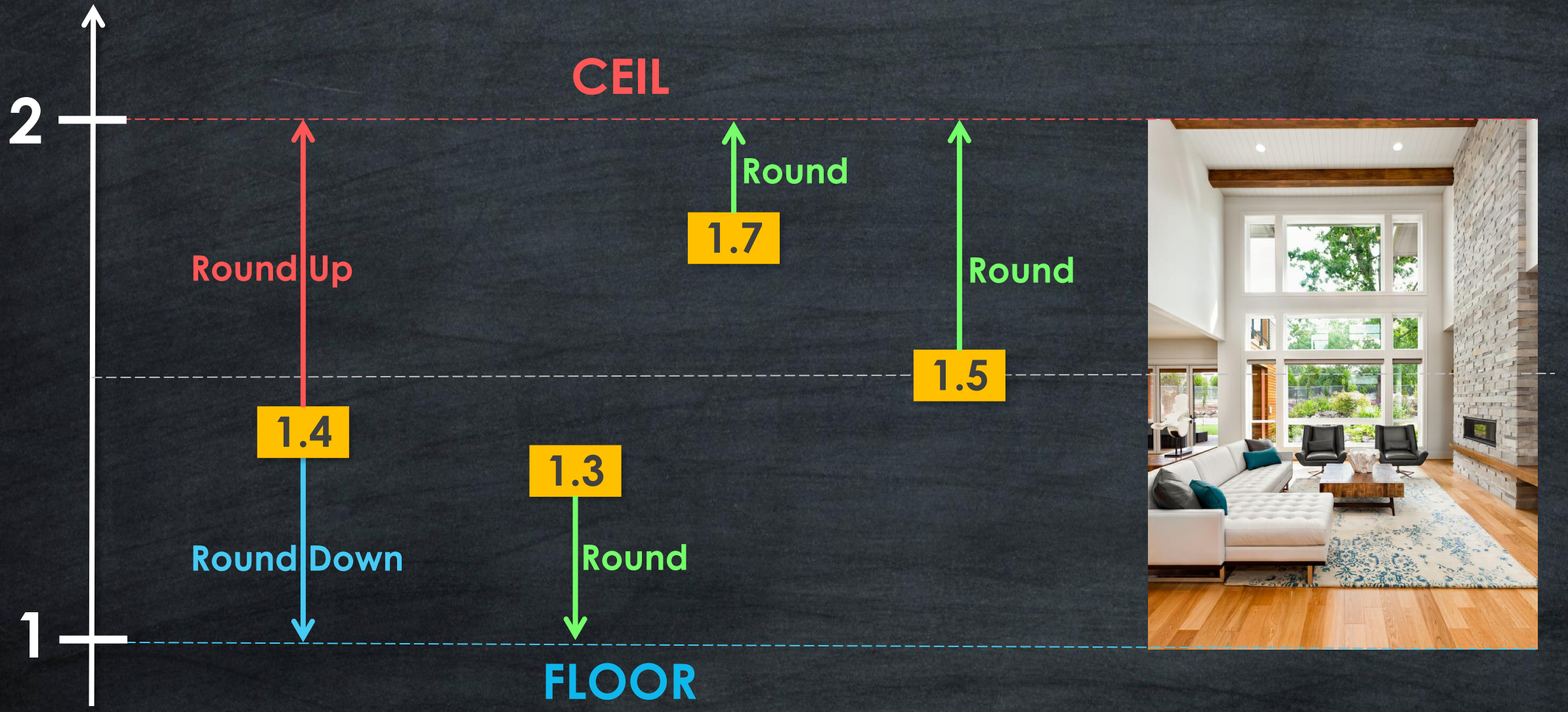
| Sub C.. | Sales - Decimals | Sales |
|---|---|---|
| Copiers | 139,098.212 | 139,098 |
| Machines | 276,783.934 | 276,784 |
| Tables | 275,341.609 | 275,342 |
| Bookcases | 177,883.848 | 177,884 |
| Chairs | 400,898.015 | 400,898 |
| Supplies | 122,461.634 | 122,462 |
| Phones | 417,312.062 | 417,312 |
| Storage | 235,530.304 | 235,530 |
| Accessories | 185,790.794 | 185,791 |
| Appliances | 120,157.372 | 120,157 |
| Binders | 224,848.616 | 224,849 |
| Furnishings | 104,211.868 | 104,212 |
| Envelopes | 20,999.068 | 20,999 |
| Paper | 93,144.558 | 93,145 |
| Art | 31,210.714 | 31,211 |
| Labels | 14,442.388 | 14,442 |
| Fasteners | 4,354.200 | 4,354 |

The Purpose is to **Round**, **simplify** the numbers, and **hide details** in Visulizations.

NUMBER FUNCTIONS

# CEILING, FLOOR, ROUND

DATA WITH BARAA



# CEILING

## Round up numbers

Syntax

```
CEILING(number)
```

Examples

| | | |
|---|---|---|
| `CEILING(1.2)` | → | 2 |
| `CEILING(1.8)` | → | 2 |
| `CEILING(1.5)` | → | 2 |

# FLOOR

## Round Down numbers

Syntax

```
FLOOR(number)
```

Examples

| | | |
|---|---|---|
| `FLOOR(1.2)` | → | 1 |
| `FLOOR(1.8)` | → | 1 |
| `FLOOR(1.5)` | → | 1 |

# ROUND

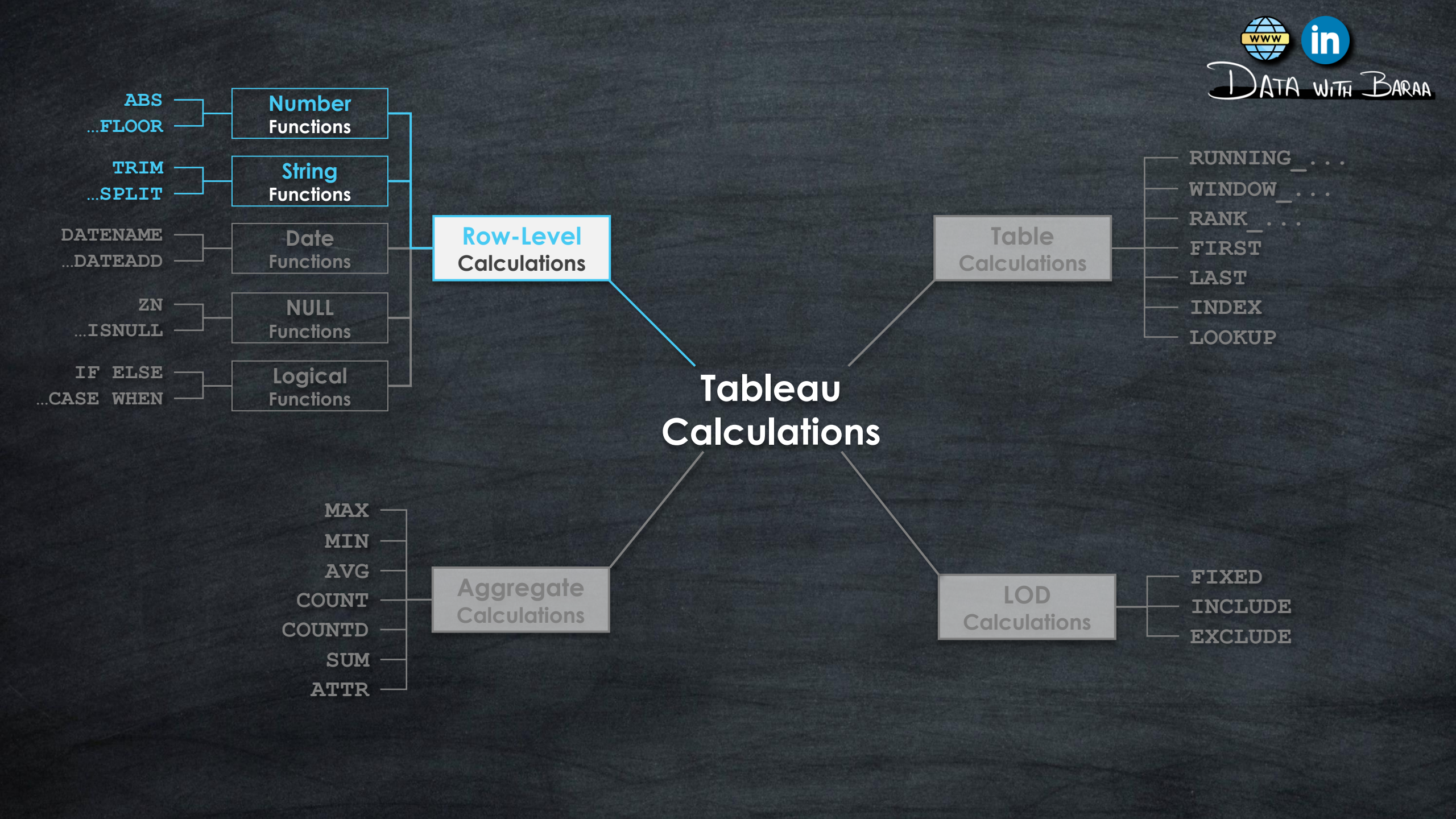
## Round Numbers to nearest Integer

Syntax

```
ROUND(number,[decimal])
```

Examples

| | | |
|---|---|---|
| `ROUND(1.2)` | → | 1 |
| `ROUND(1.8)` | → | 2 |
| `ROUND(1.5)` | → | 2 |


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# String Functions
Use Cases

DATA WITH BARAA

## Main Purpose to is Mainuplate Text Values

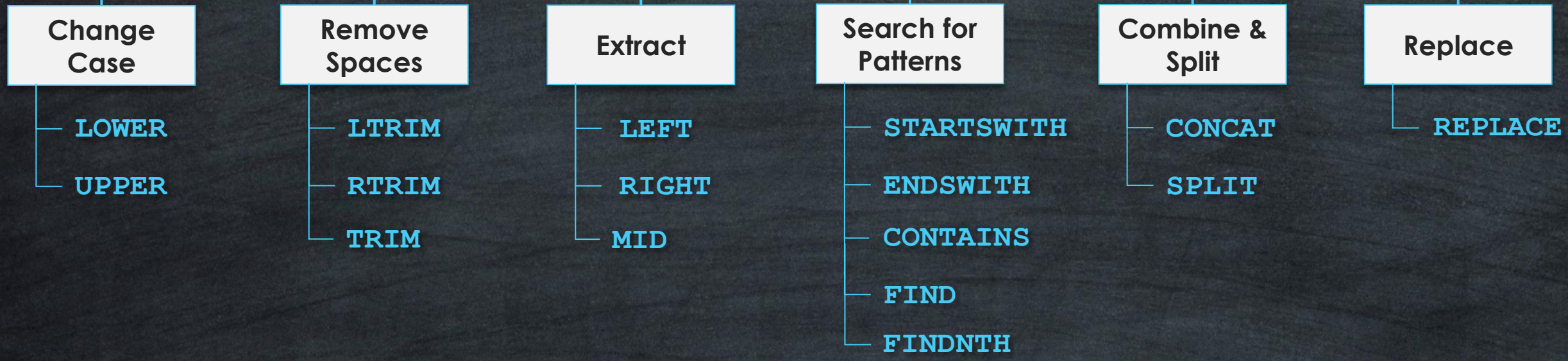
### Data Cleaning

- Removing unwanted Characters - `REPLACE`
- Trimming Leading or trailing Spaces – `LTRIM,RTRIM,TRIM`

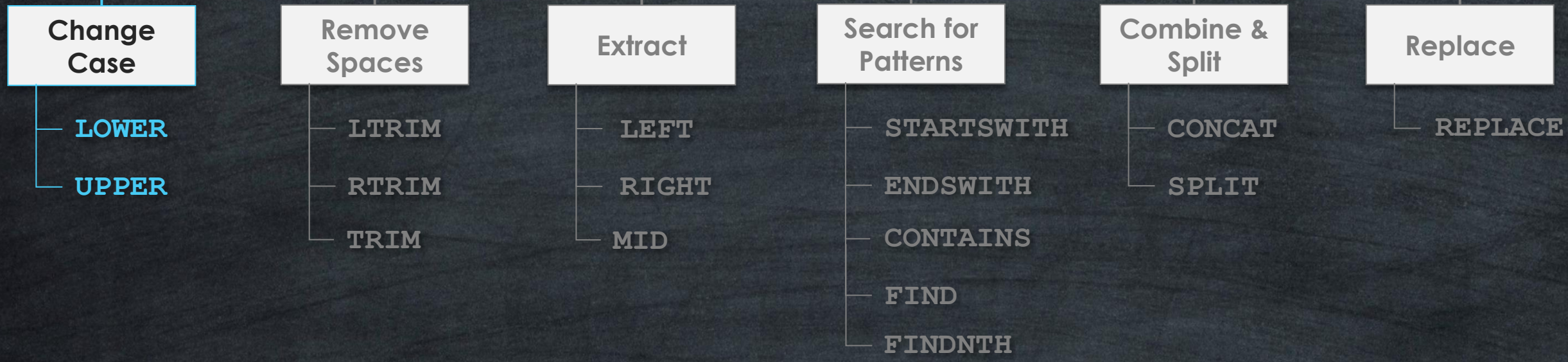
### Data Transformation

- Data Extraction – `LEFT,RIGHT,MID`
- Spliting Data- `SPLIT`

# String Functions
## Use Cases

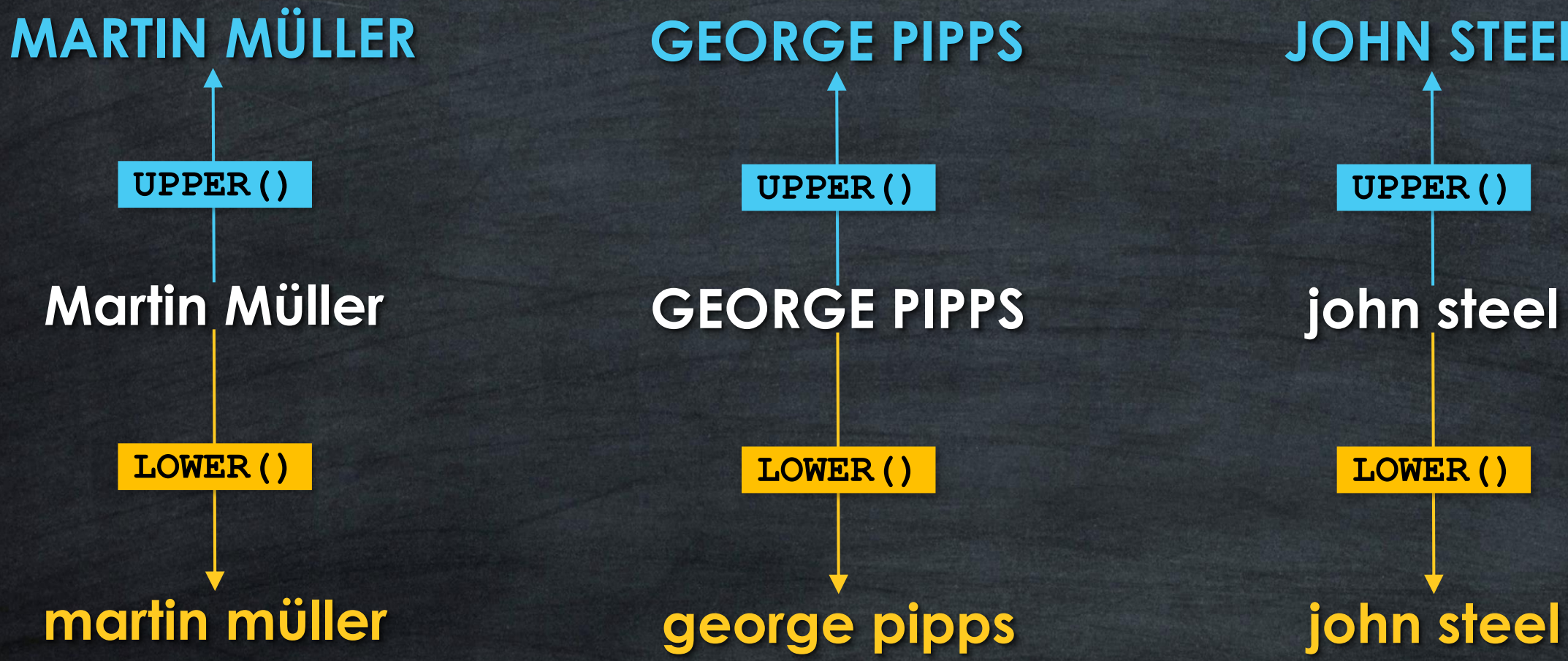


# String Functions
## Use Cases



STRING FUNCTIONS

# LOWER & UPPER



# UPPER

## Convert characters to uppercase

Syntax

```
UPPER(string)  ➡  string
```

Example

```
UPPER("Maria")  ➡  "MARIA"
```

# LOWER

## Convert characters to lowercase

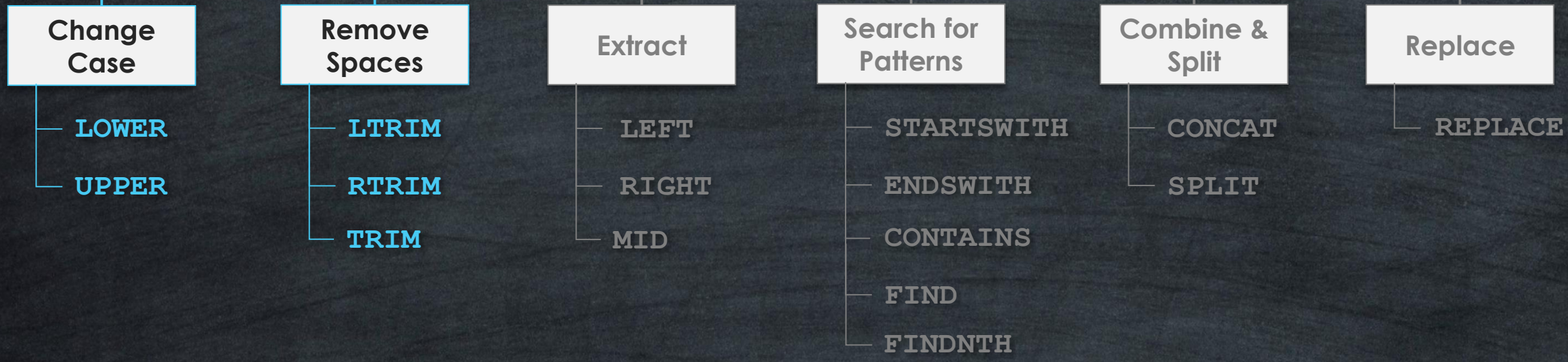
Syntax

```
LOWER(string)  ➡  string
```

Example

```
LOWER("Maria")  ➡  "maria"
```

# String Functions
## Use Cases



STRING FUNCTIONS

# LTRIM, RTRIM, TRIM



LTRIM()



RTRIM()


TRIM()

LTRIM + RTRIM

# LTRIM

Remove any leading spaces

Syntax

```
LTRIM(string)
```

Example

```
LTRIM(" Maria ") → "Maria "
```

# RTRIM

Remove any trailing spaces

Syntax

```
RTRIM(string)
```

Example

```
RTRIM(" Maria ") → " Maria"
```

# TRIM

Remove both leading & trailing spaces

Syntax

```
TRIM(string)
```

Example

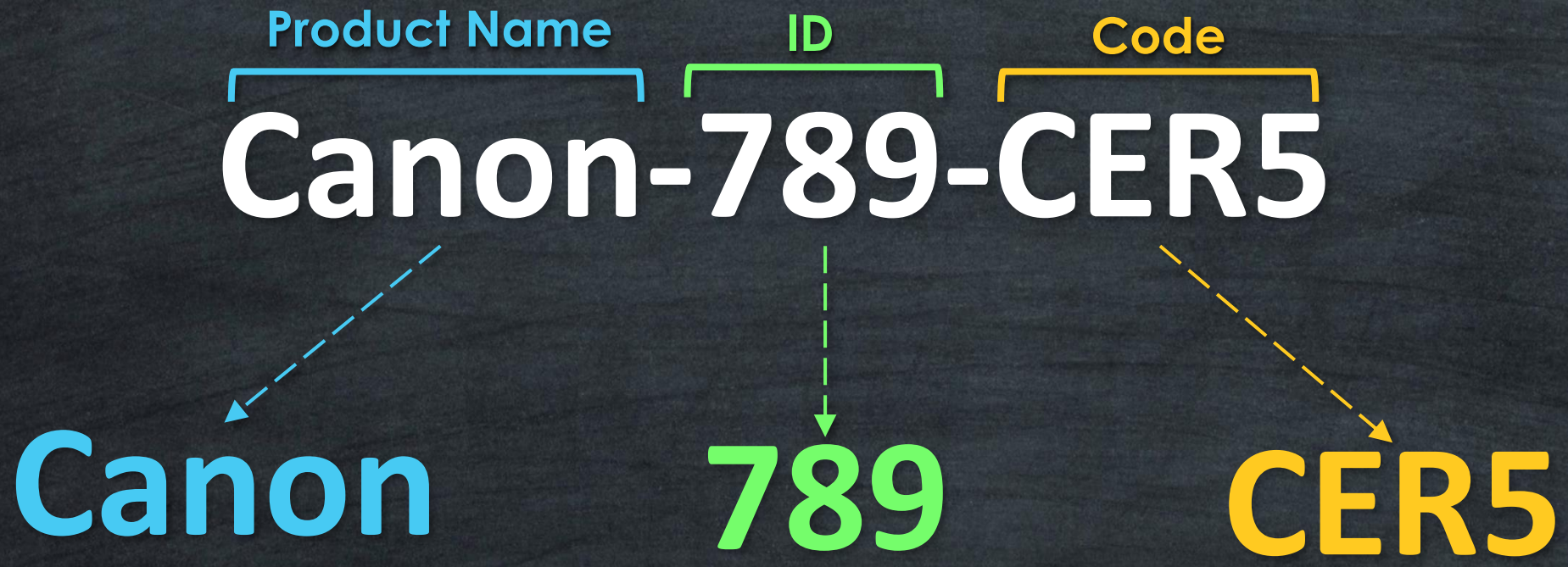
```
TRIM(" Maria ") → "Maria"
```

# String Functions
## Use Cases

STRING FUNCTIONS

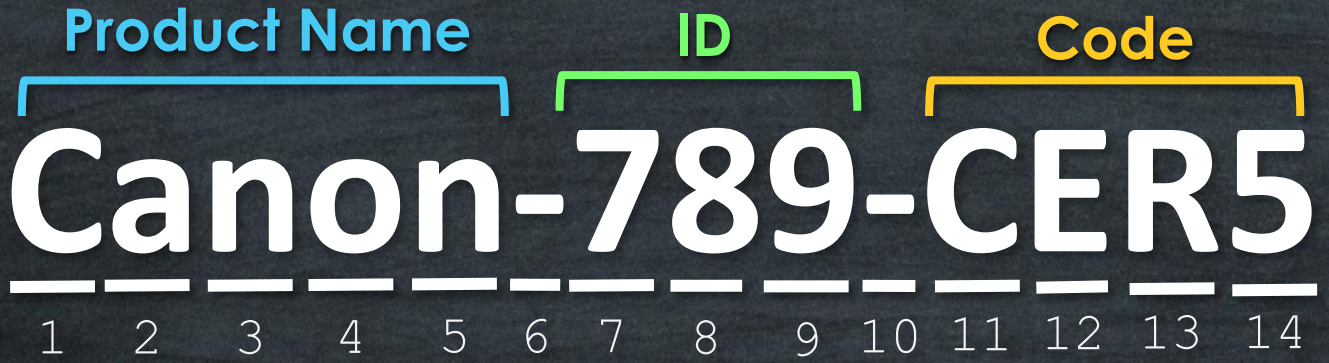
# LEFT, RIGHT, MID





# LEFT





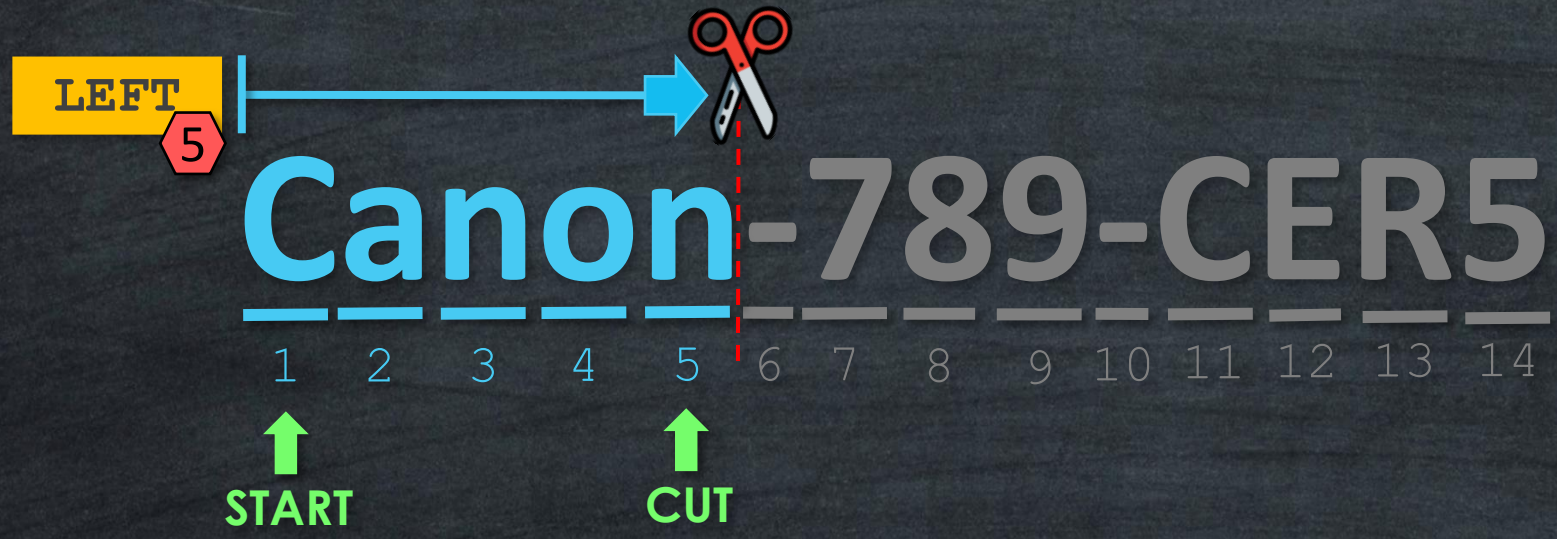

Results

Syntax `LEFT(string,num_chars)` → `string`

Examples `LEFT("Canon-789-CER5",5)`

# LEFT

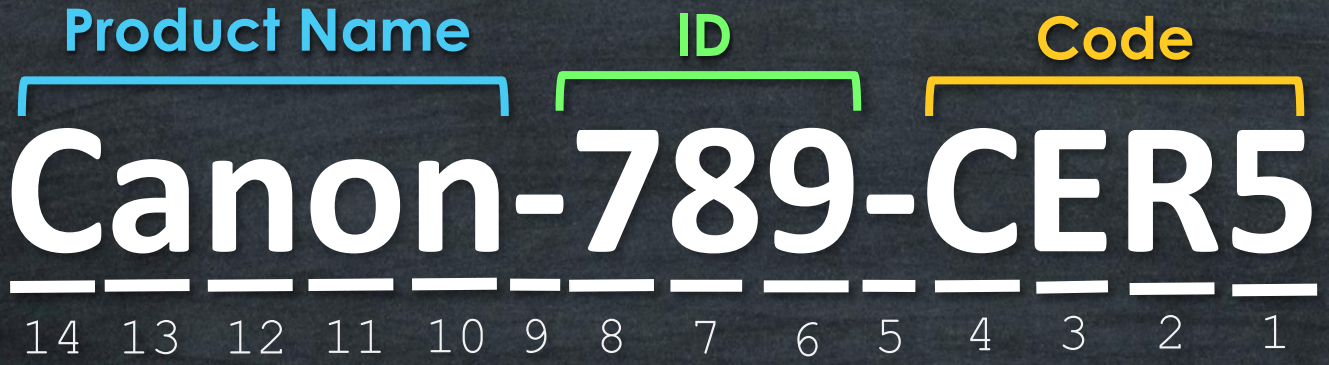





| | | Results |
|---|---|---|
| Syntax | `LEFT(string,num_chars)` | `string` |
| Examples | `LEFT("Canon-789-CER5",5)` | `"Canon"` |

# RIGHT





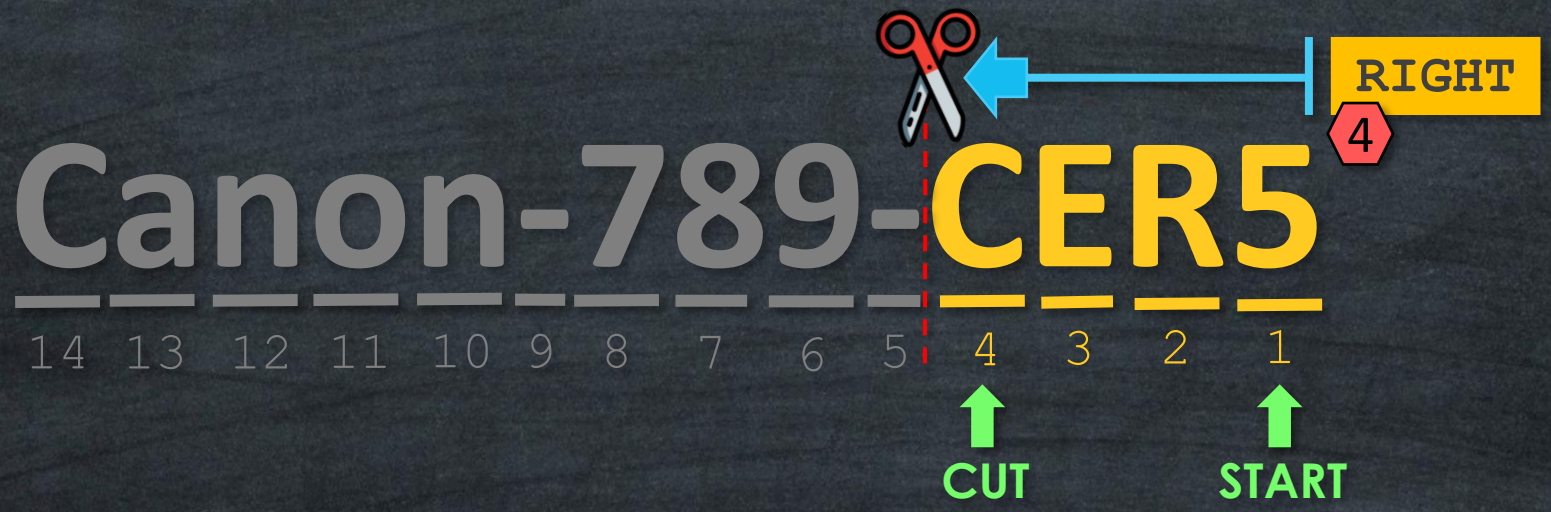

Results

Syntax `RIGHT(string,num_chars)` → `string`

Examples `RIGHT("Canon-789-CER5",4)`

STRING FUNCTIONS

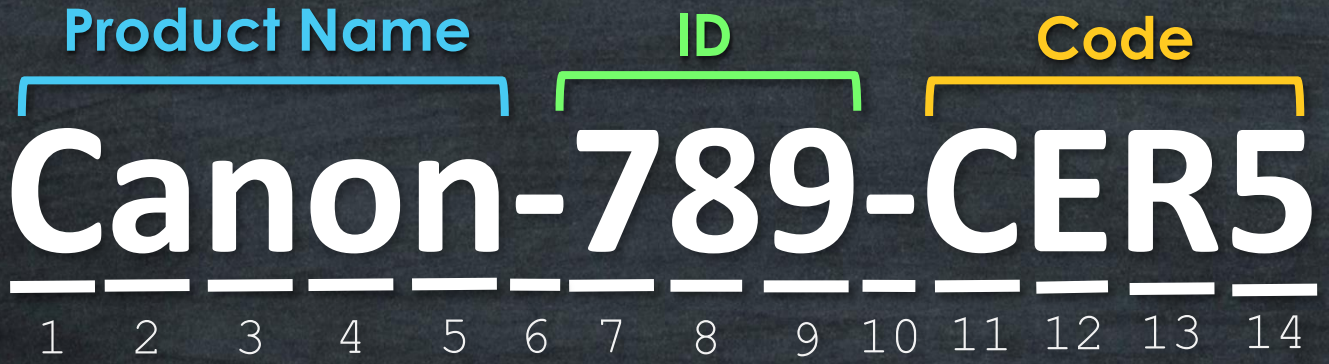
# RIGHT




| | | Results |
|---|---|---|
| Syntax | `RIGHT(string,num_chars)` | `string` |
| Examples | `RIGHT("Canon-789-CER5",4)` | `"CER5"` |

# MID

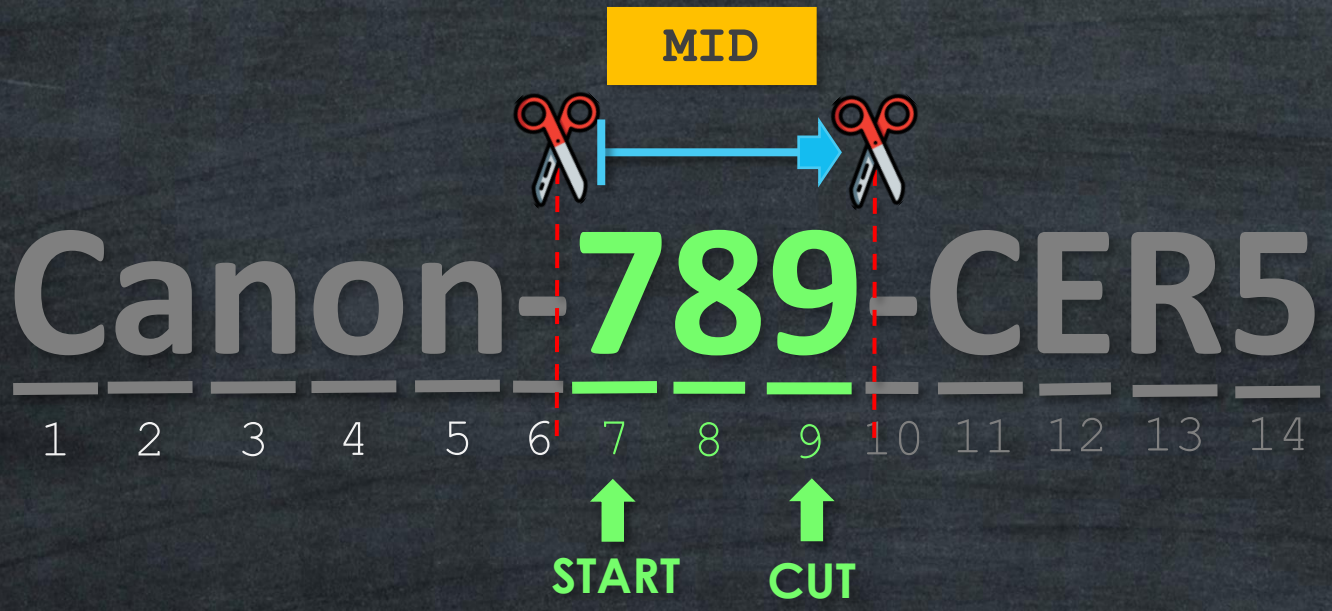





| | | Results |
|---|---|---|
| Syntax | `MID(string,start,[length])` | `string` |
| Examples | `MID("Canon-789-CER5",7,3)` | |

# MID






| | | Results |
|---|---|---|
| Syntax | `MID(string,start,[length])` | `string` |
| Examples | `MID("Canon-789-CER5",7,3)` | `"789"` |

STRING FUNCTIONS

# LEFT, RIGHT, MID

## USE CASES

DATA WITH BARAA

### #1 Use Case | Extracting File Extensions

`Document.txt` → RIGHT → `txt`

### #2 Use Case | Extracting Area Code

`(123) 456-7890` → MID → `123`

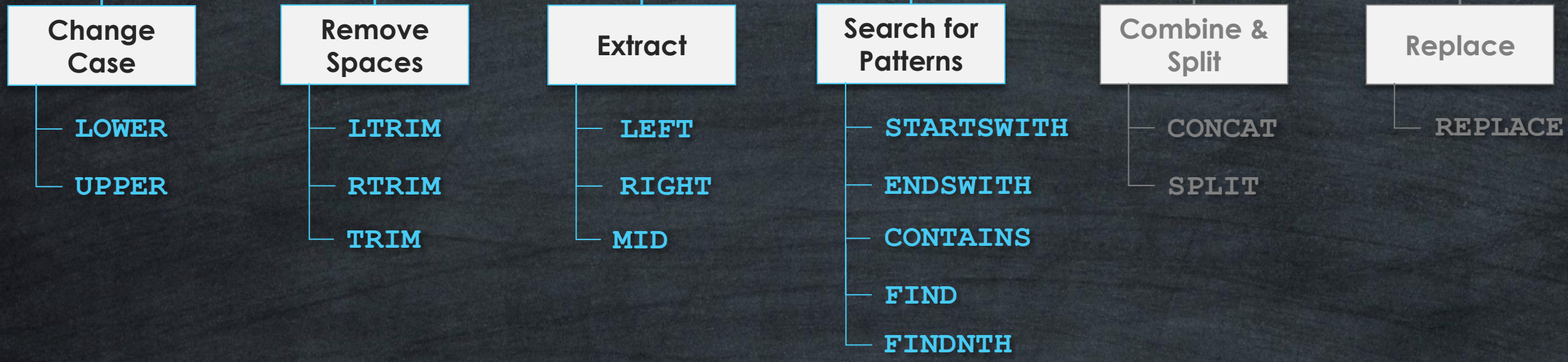
### #3 Use Case | Extracting TLD from URL

`https://Datawithbaraa.com` → RIGHT → `com`

### #4 Use Case | Extracting protocoll from URL

`https://Datawithbaraa.com` → LEFT → `https`

# String Functions
## Use Cases



# Search Functions

## 1# GROUP

Return whether the Substring exists or not

Functions

- STARTSWITH
- ENDSWTIH
- CONTAINS

Result TRUE FALSE

```
CONTAINS("Canon-789-CER5","-")
```

→ TRUE

## 2# GROUP

Return the Position of Substring

Functions

- FIND
- FINDNTH

Result Number

```
FIND("Canon-789-CER5","-")
```

→ 6

STRING FUNCTIONS

# STARTSWITH




Results

Syntax

```
STARTWITH(string,substring)
```

→ TRUE | FALSE

Examples

```
STARTWITH("MonitorLG-4k","Monitor")
```

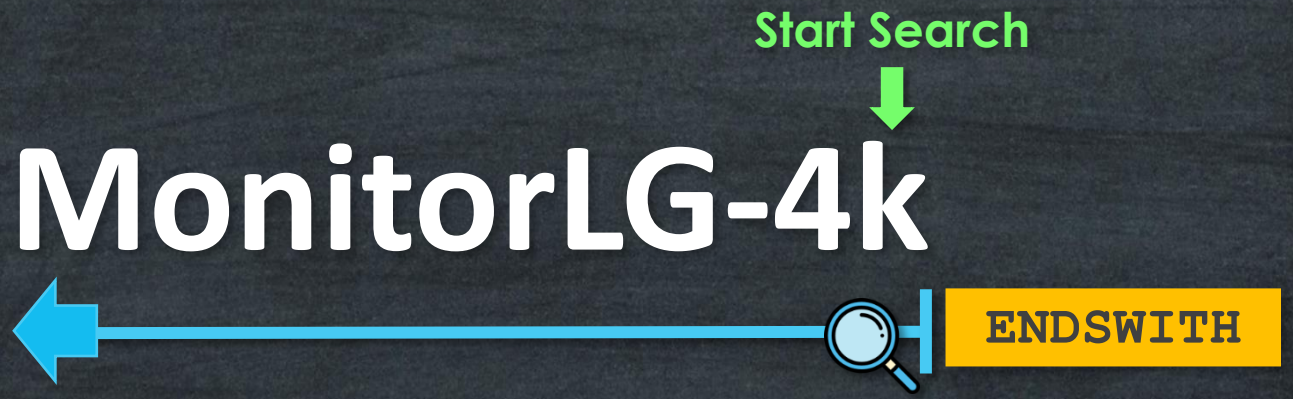
→ TRUE

```
STARTWITH("MonitorLG-4k","LG")
```

→ FALSE

# ENDSWITH






Syntax

```
ENDSWITH(string,substring)
```

Results: TRUE | FALSE

Examples

```
ENDSWITH("MonitorLG-4k","4k")
```

→ TRUE

```
ENDSWITH("MonitorLG-4k","LG")
```

→ FALSE

# CONTAINS



Search Everywhere

## MonitorLG-4k


| | | Results |
|---|---|---|
| Syntax | `CONTAINS(string,substring)` | TRUE \| FALSE |
| Examples | `CONTAINS("MonitorLG-4k","Monitor")` | TRUE |
| | `CONTAINS("MonitorLG-4k","LG")` | TRUE |
| | `CONTAINS("MonitorLG-4k","4G")` | FALSE |

# Search Functions

## 1# GROUP

Return whether the Substring exists or not

Functions

- STARTSWITH
- ENDSWTIH
- CONTAINS

Result TRUE FALSE

```
CONTAINS("Canon-789-CER5","-")
```

→ TRUE

## 2# GROUP

Return the Position of Substring

Functions

- FIND
- FINDNTH

Result Number

```
FIND("Canon-789-CER5","-")
```

→ 6

STRING FUNCTIONS

# FIND

Returns the position of **First** occurrence

Example

```
FIND("Canon-789-CER5","-")
```

→ 6

# FINDNTH

Returns the position of **Nth** occurrence

Example

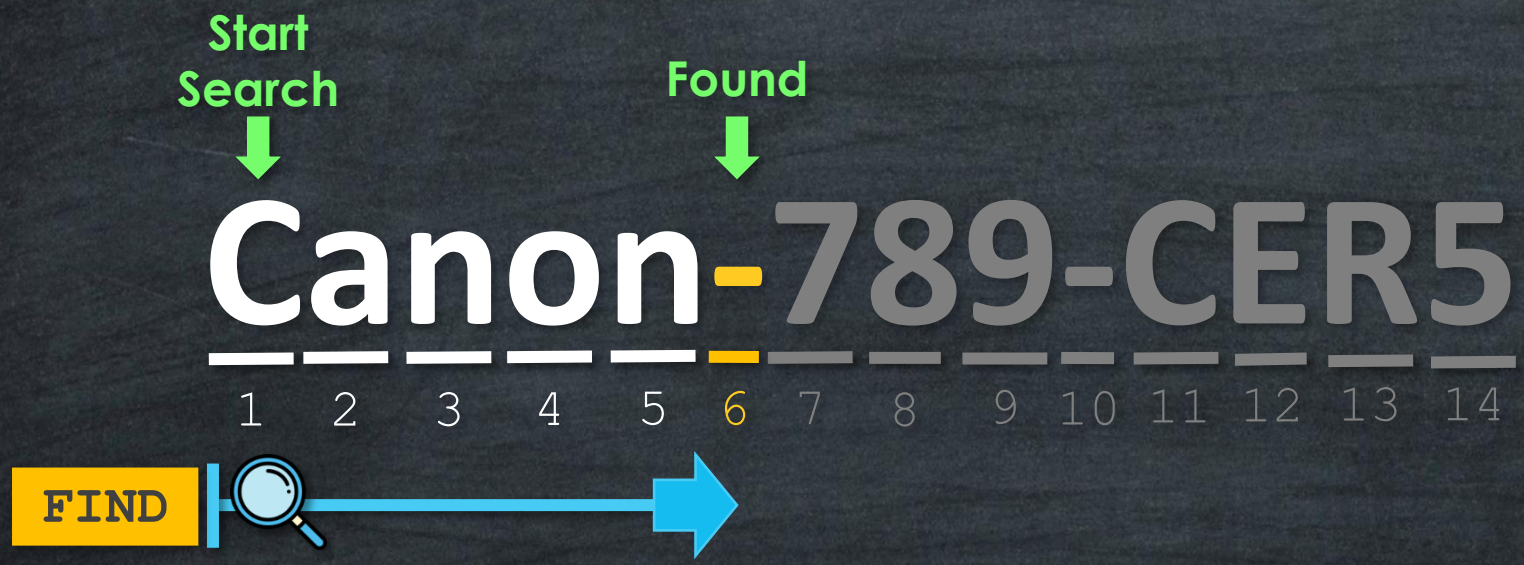
```
FINDNTH("Canon-789-CER5","-",2)
```

→ 10

STRING FUNCTIONS

# FIND

Data with Baraa




Results

**Syntax** `FIND(string,substring,[start])` → Number

**Examples** `FIND("Canon-789-CER5","-")` → 6

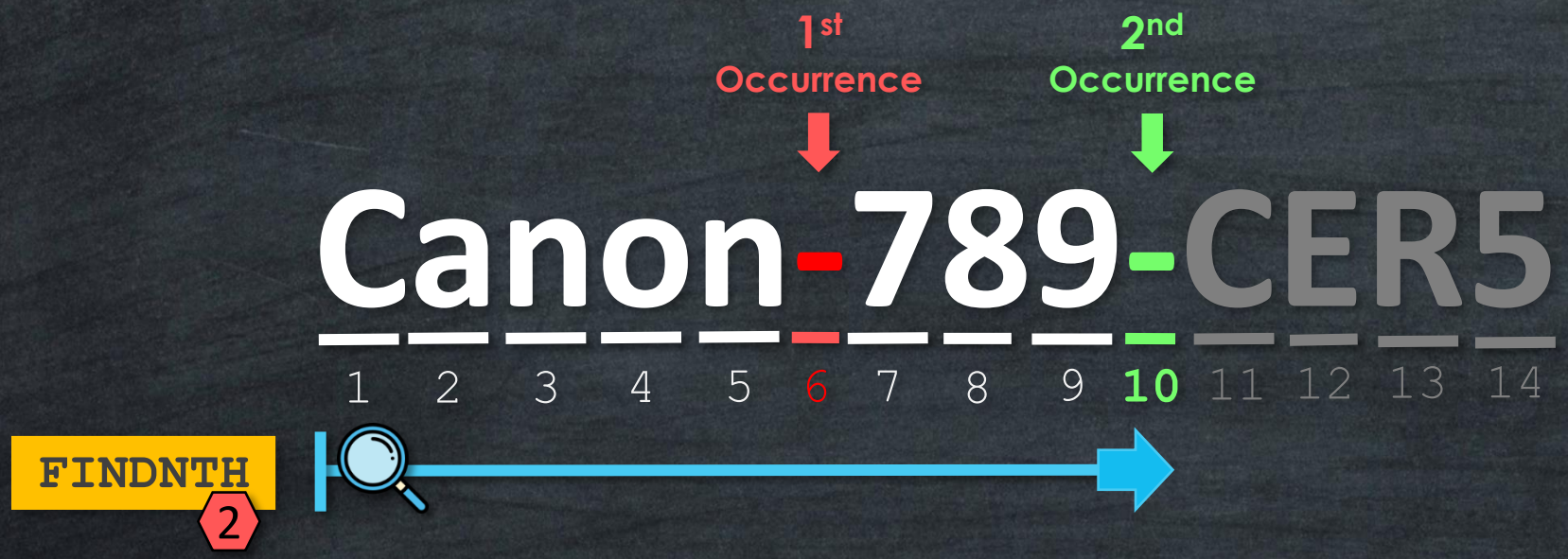
STRING FUNCTIONS

# FIND


| | | Results |
|---|---|---|
| Syntax | `FIND(string,substring,[start])` | Number |
| Examples | `FIND("Canon-789-CER5","-")` | 6 |
| | `FIND("Canon-789-CER5","-",7)` | 10 |

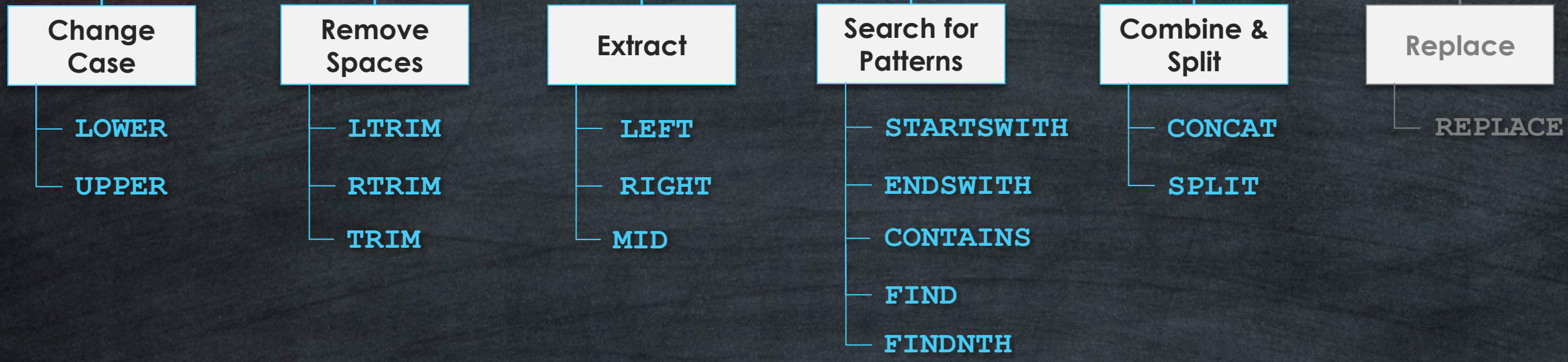
# FINDNTH






| | | Results |
|---|---|---|
| Syntax | `FINDNTH(string,substring,occurrence)` | Number |
| Examples | `FINDNTH("Canon-789-CER5","-",2)` | 10 |

# String Functions
## Use Cases



STRING FUNCTIONS

# CONCAT(+)

DATA WITH BARAA

First Name: Michael + Last Name: Scott → Full Name: MichaelScott

First Name: Michael + Space: ␣ + Last Name: Scott → Full Name: Michael Scott

Examples

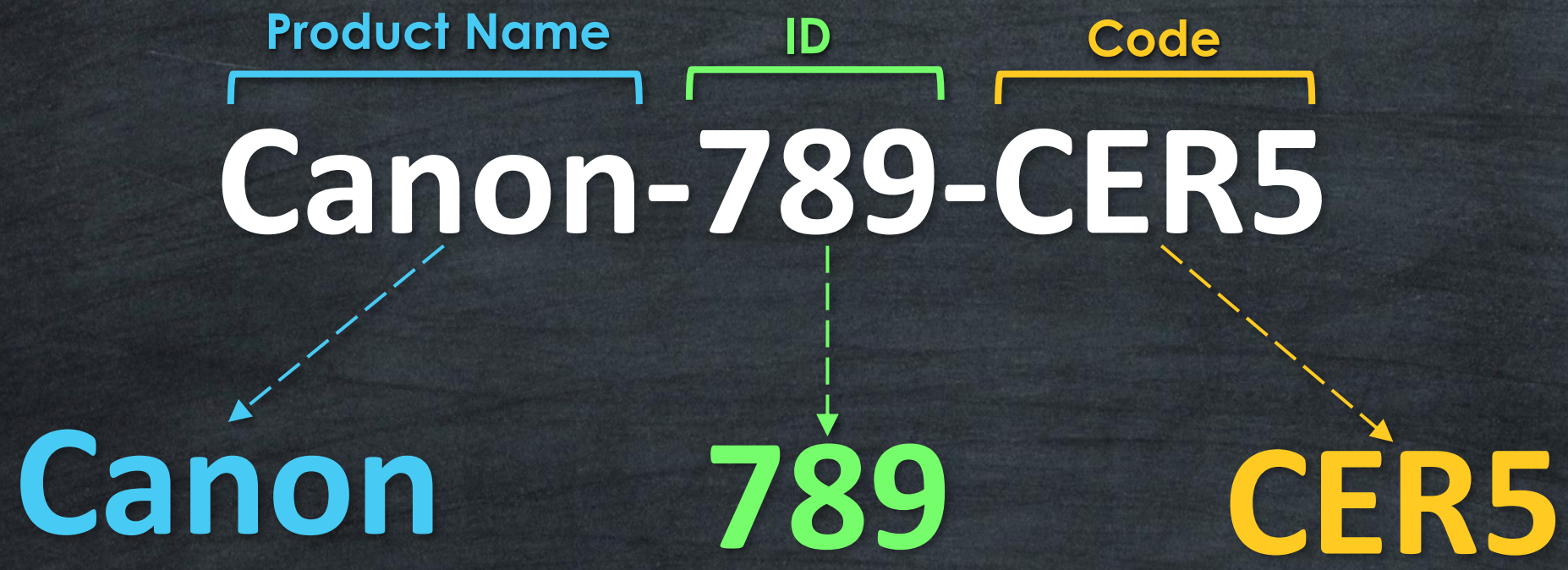
```
"Michael" + " " + "scott"
```

→

```
"Michael scott"
```

STRING FUNCTIONS

# SPLIT



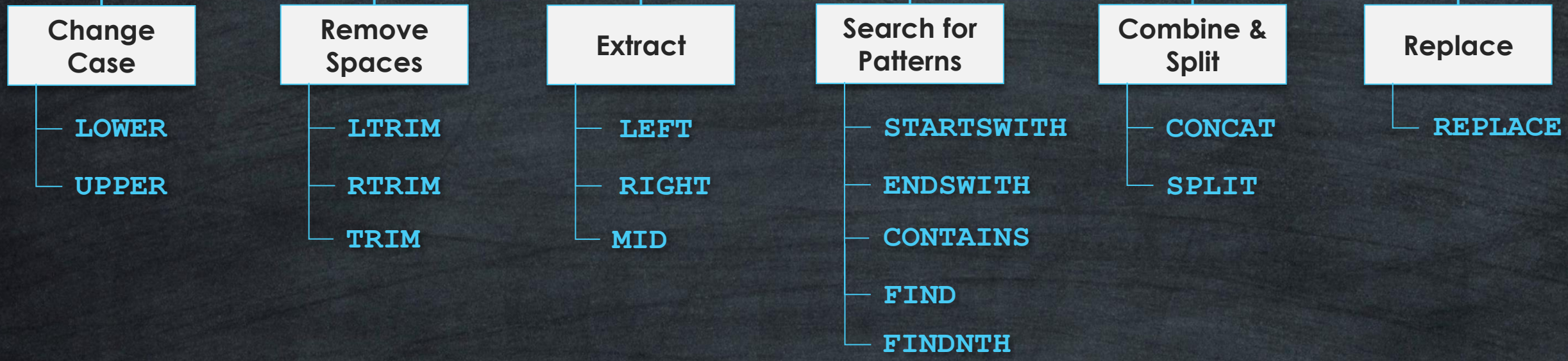
STRING FUNCTIONS

# SPLIT

DATA WITH BARAA

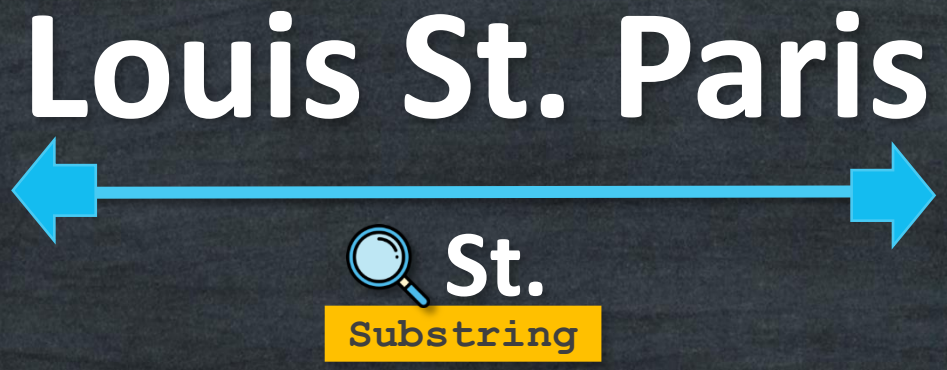

| | | Results |
|---|---|---|
| Syntax | `SPLIT(string,delimiter,token number)` | String |
| Example | `SPLIT("Canon-789-CER5","-",1)` | "Canon" |
| | `SPLIT("Canon-789-CER5","-",2)` | "789" |
| | `SPLIT("Canon-789-CER5","-",3)` | "CER5" |

# String Functions
## Use Cases



# REPLACE






Syntax `REPLACE(string,substring,replacement)`

Example `REPLACE("Louis St. Paris","St.","Street")`

Results: `String`

# REPLACE




**Syntax**

```
REPLACE(string,substring,replacement)
```

**Results**

```
String
```

**Example**

```
REPLACE("Louis St. Paris","St.","Street")
```

# REPLACE

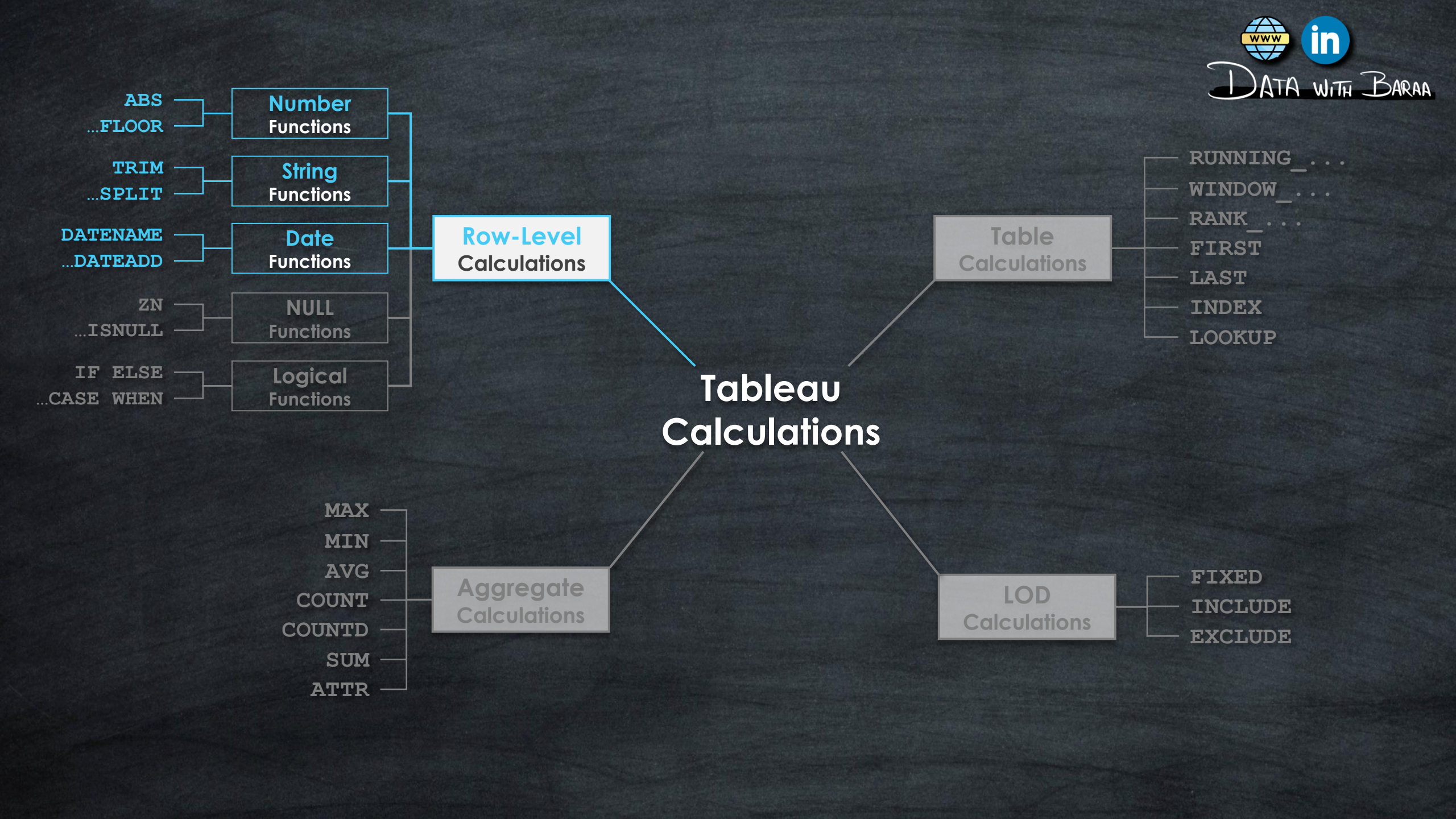



Results

Syntax `REPLACE(string,substring,replacement)` → `String`

Example `REPLACE("Louis St. Paris","St.","Street")` → `"Louis Street Paris"`

`REPLACE("Ann Paris","St.","Street")` → `"Ann Paris"`


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# Date Functions
Use Cases

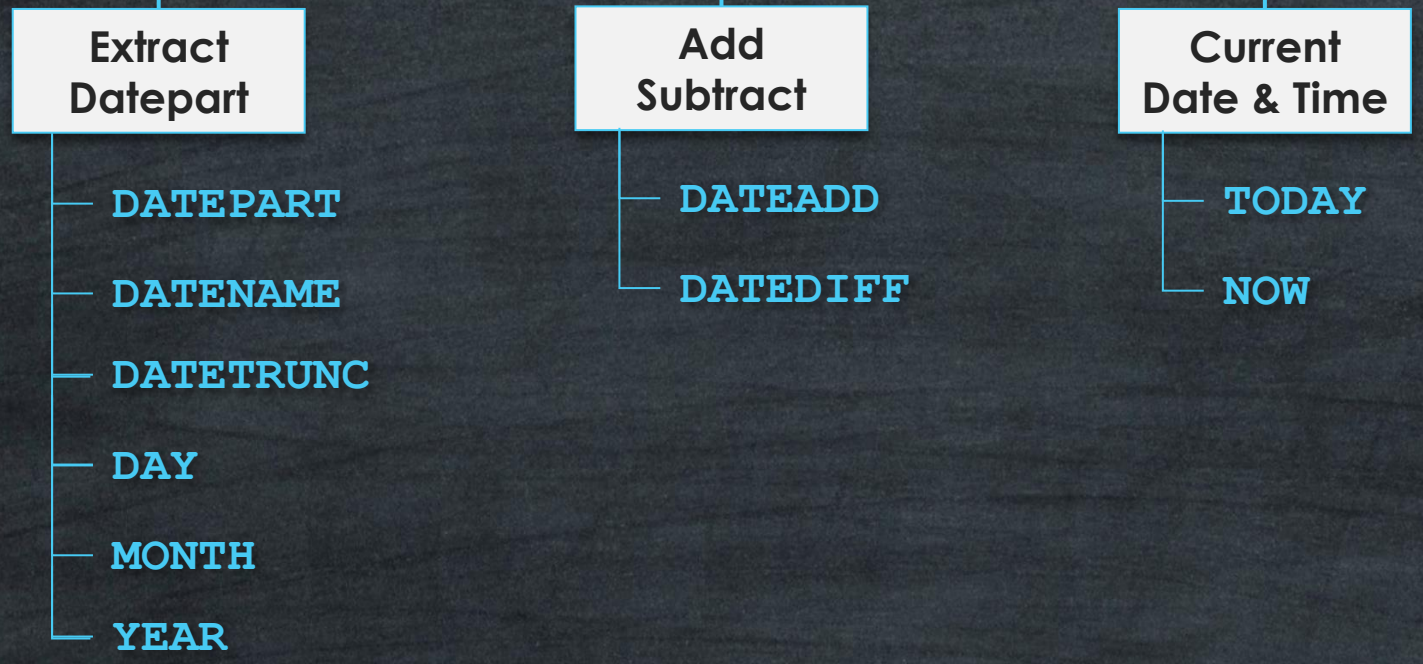
DATA WITH BARAA

## Main Purpose to is Mainuplate Date Values

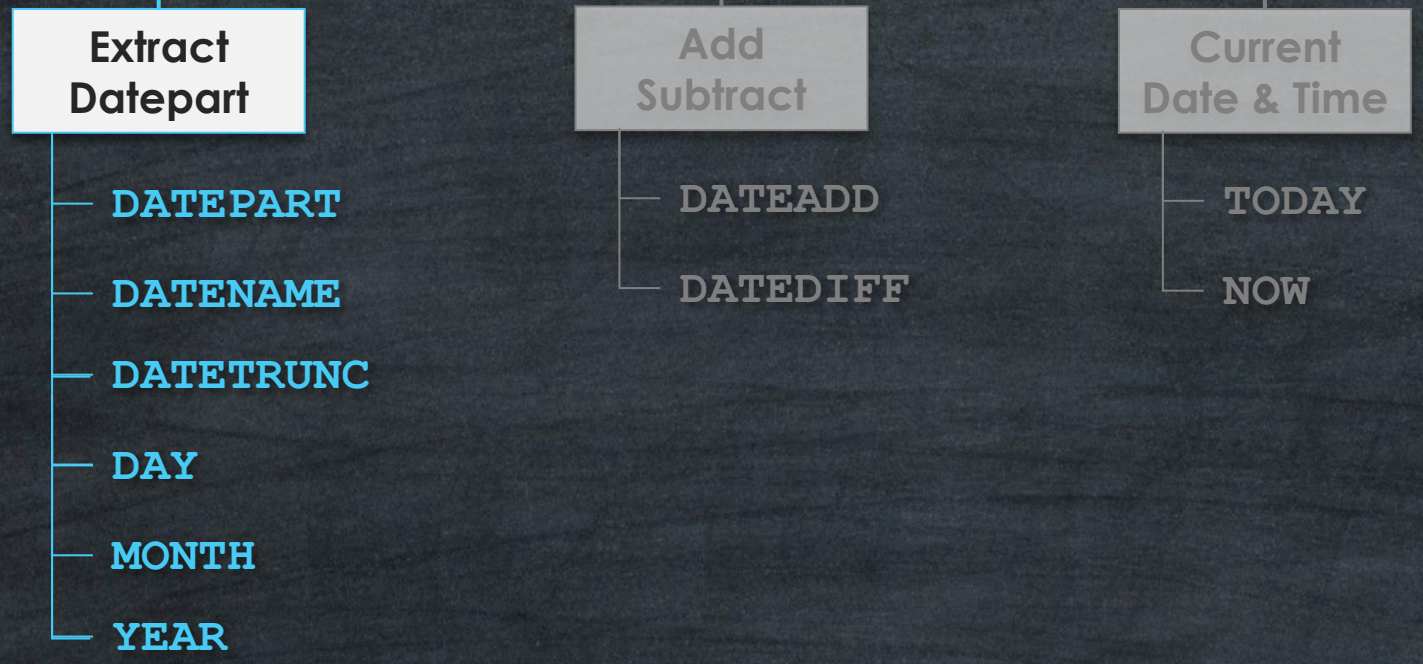
### Data Cleaning
- Removing unwanted Characters - `REPLACE`
- Trimming Leading or trailing Spaces – `LTRIM,RTRIM,TRIM`

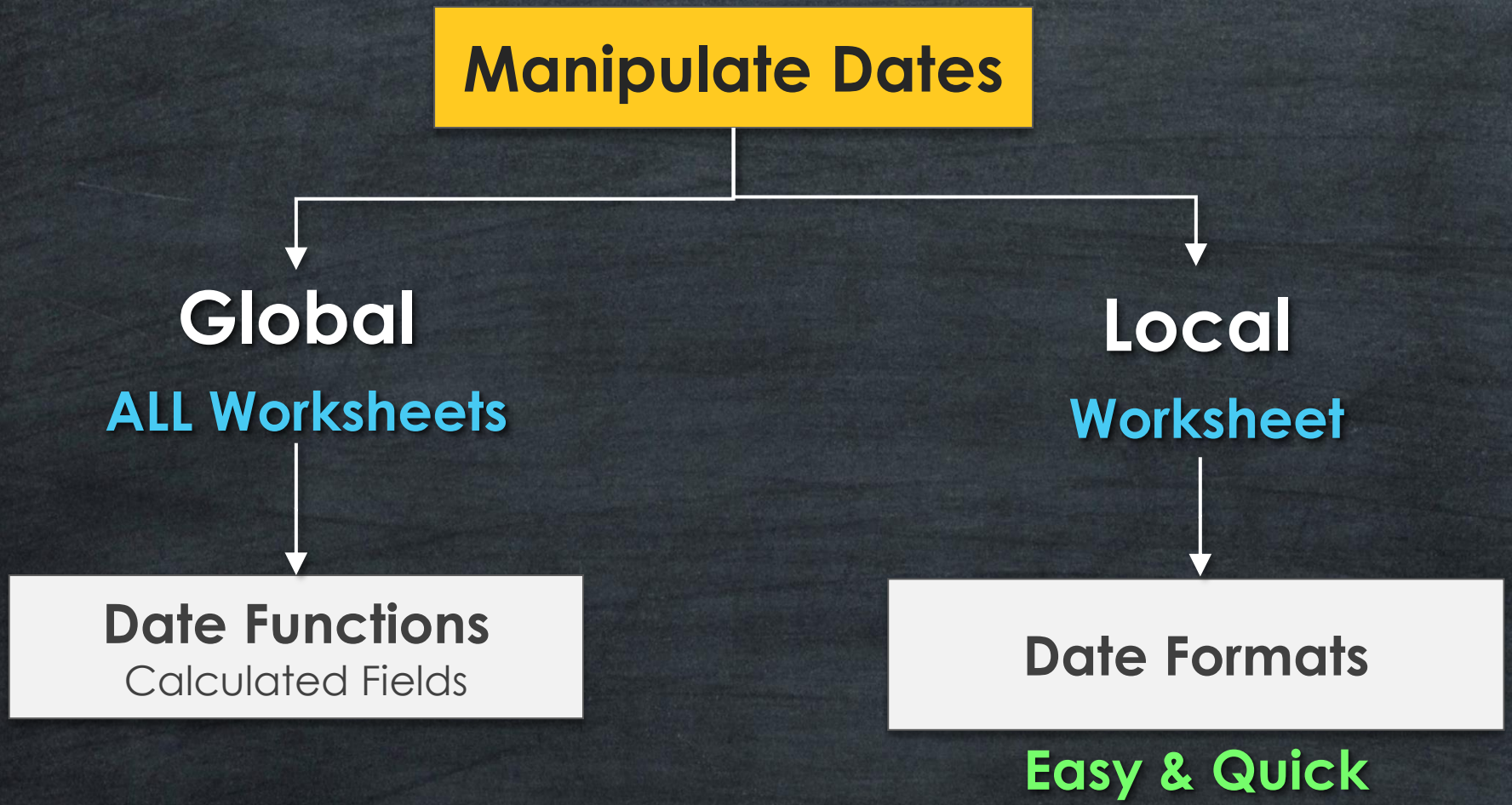
### Data Transformation
- Data Extraction – `LEFT,RIGHT,MID`
- Spliting Data- `SPLIT`

# Date Functions
## Use Cases



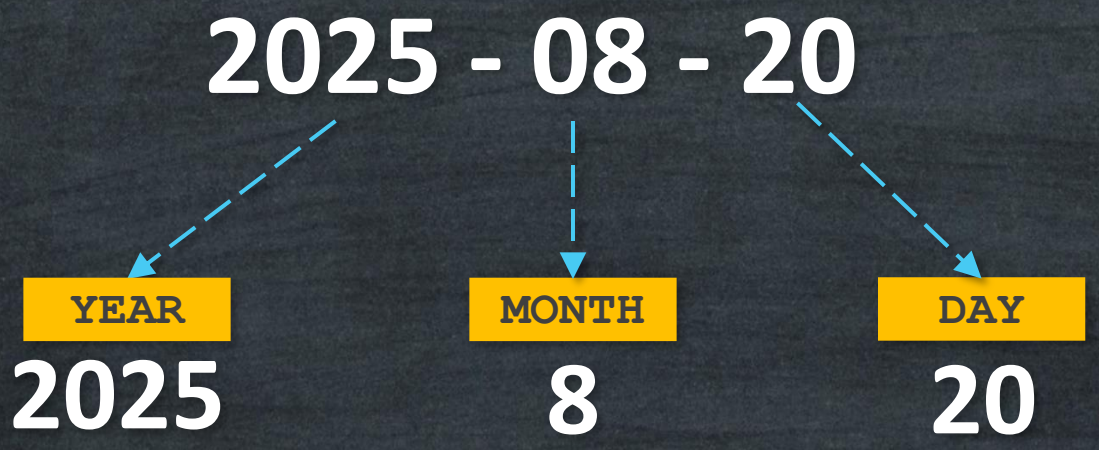
# Date Functions
## Use Cases





# DATEPART






**Syntax**

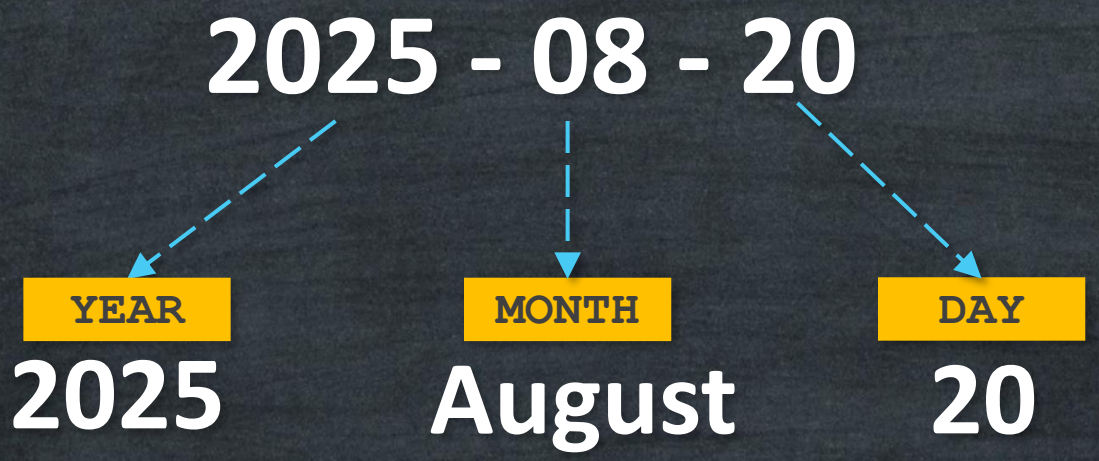
```
DATEPART(date_part,date)  ➡  Number
```

**Examples**

```
DATEPART('day',#2025-08-20#)  ➡  20
DATEPART('month',#2025-08-20#)  ➡  8
DATEPART('year',#2025-08-20#)  ➡  2025
```

# DATENAME






**Syntax**

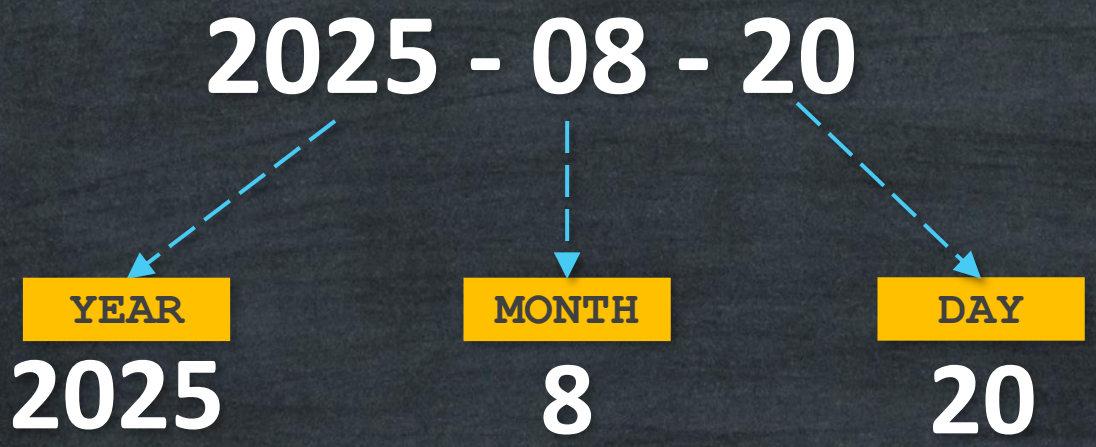
```
DATENAME(date_part,date)  ➡  String
```

**Examples**

```
DATENAME('year',#2025-08-20#)   ➡  '2025'
DATENAME('month',#2025-08-20#)  ➡  'August'
DATENAME('day',#2025-08-20#)    ➡  '20'
```

# DAY, MONTH, YEAR






**Syntax**

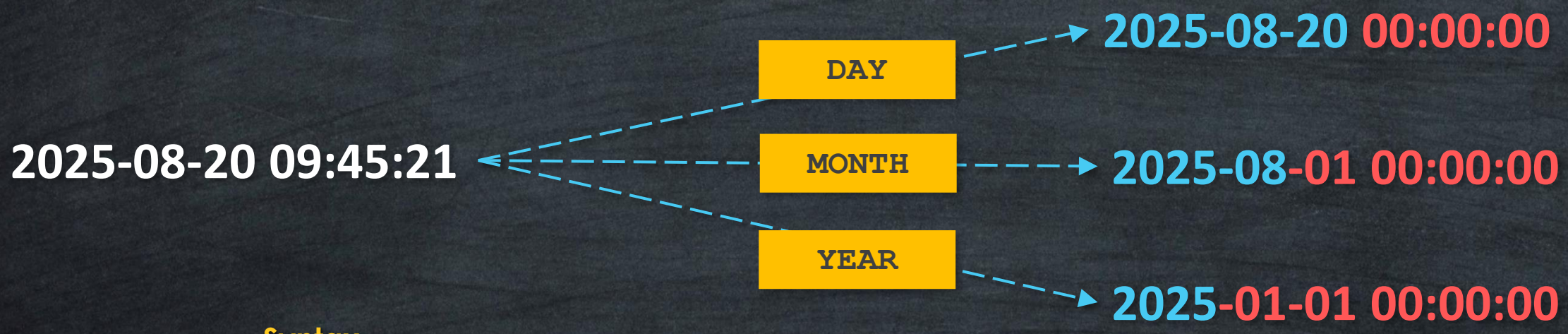
```
DAY(date)   MONTH(date)   YEAR(date)   → Number
```

| Examples | Results |
|---|---|
| DAY(#2025-08-20#) | 20 |
| MONTH(#2025-08-20#) | 8 |
| YEAR(#2025-08-20#) | 2025 |

DATE FUNCTIONS

# DATETRUNC

DATA WITH BARAA




Syntax

```
DATETRUNC(date_part,date)  →  DATE & TIME
```

Examples

```
DATETRUNC('day',#2025-08-20 09:45:21#)    →  2025-08-20 00:00:00
DATETRUNC('month',#2025-08-20 09:45:21#)  →  2025-08-01 00:00:00
DATETRUNC('year',#2025-08-20 09:45:21#)   →  2025-01-01 00:00:00
```

DATE FUNCTIONS

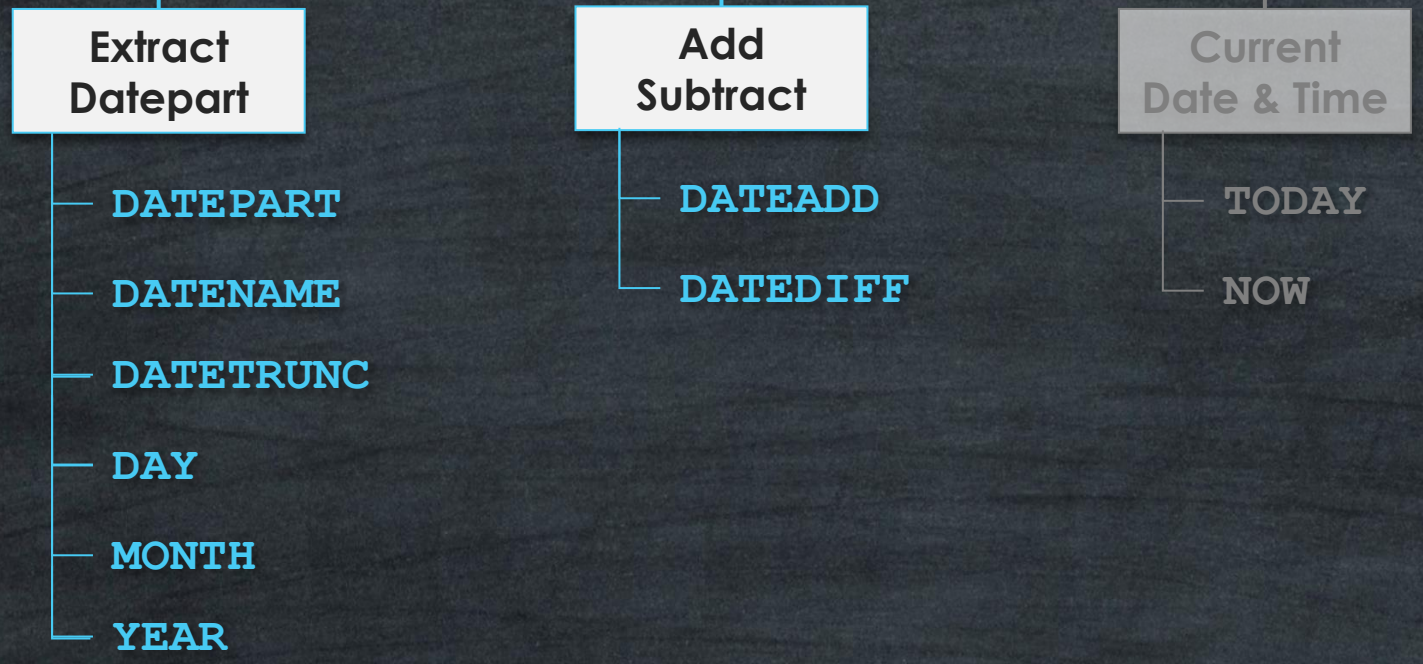
# All Functions

DATA WITH BARAA

2025-08-20 09:45:21

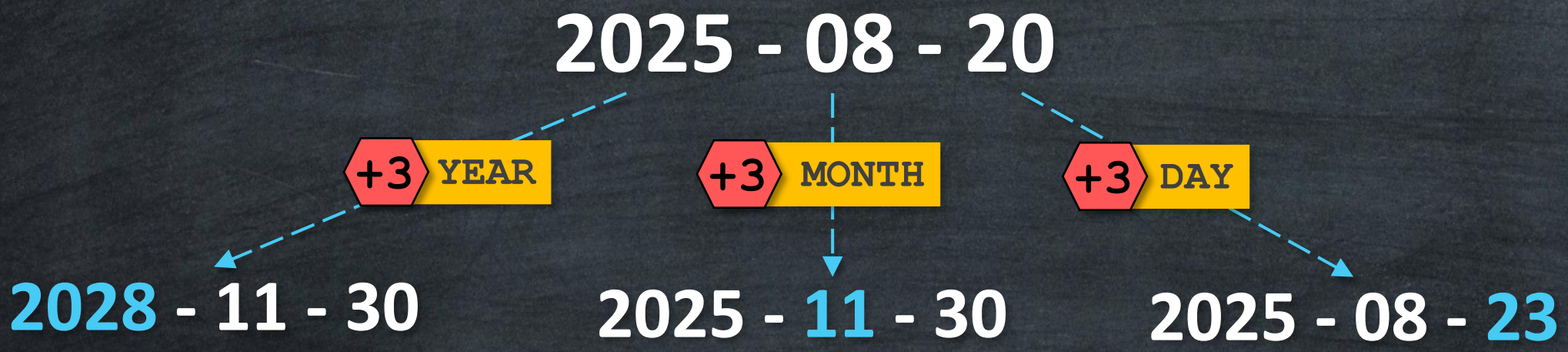
| | Number | String | Date & Time |
|---|---|---|---|
| date_part | DATEPART | DATENMAME | DATETRUNC |
| year | 2025 | 2025 | 2025-01-01 00:00:00 |
| quarter | 3 | 3 | 2025-07-01 00:00:00 |
| month | 8 | August | 2025-08-01 00:00:00 |
| day | 20 | 20 | 2025-08-20 00:00:00 |
| weekday | 4 | Wednesday | 2025-08-20 00:00:00 |
| hour | 9 | 9 | 2025-08-20 09:00:00 |
| minute | 45 | 45 | 2025-08-20 09:45:00 |
| second | 21 | 21 | 2025-08-20 09:45:21 |

# Date Functions
## Use Cases



DATE FUNCTIONS

# DATEADD




**Syntax**

```
DATEADD(date_part,interval,date)
```

→ Date

**Examples**

```
DATEADD("Year", 3, #2025-08-20#)
```

→ 2028-08-20

```
DATEADD("Year", -3, #2025-08-20#)
```

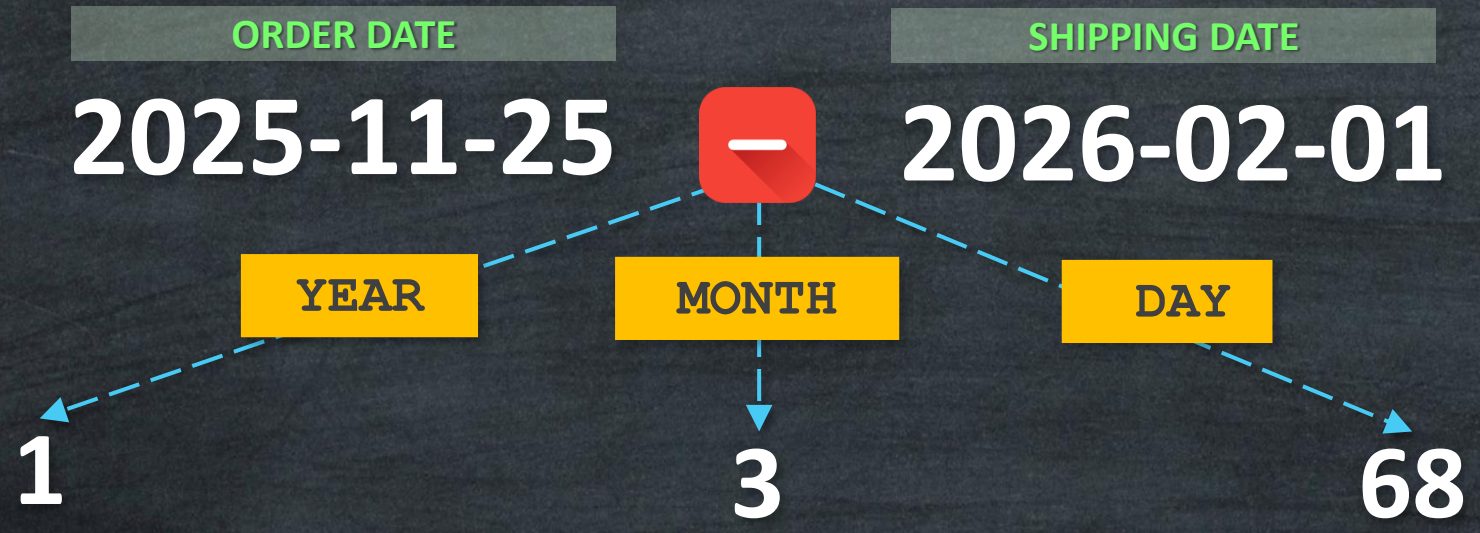
→ 2022-08-20

```
DATEADD("day", -3, #2025-08-20#)
```

→ 2025-08-17

DATE FUNCTIONS

# DATEDIFF




**Syntax**

```
DATEDIFF(date_part,start_Date,end_date)
```

→ Number

**Examples**

```
DATEDIFF('year',#2025-11-25#, #2026-02-01#)
```

→ 1

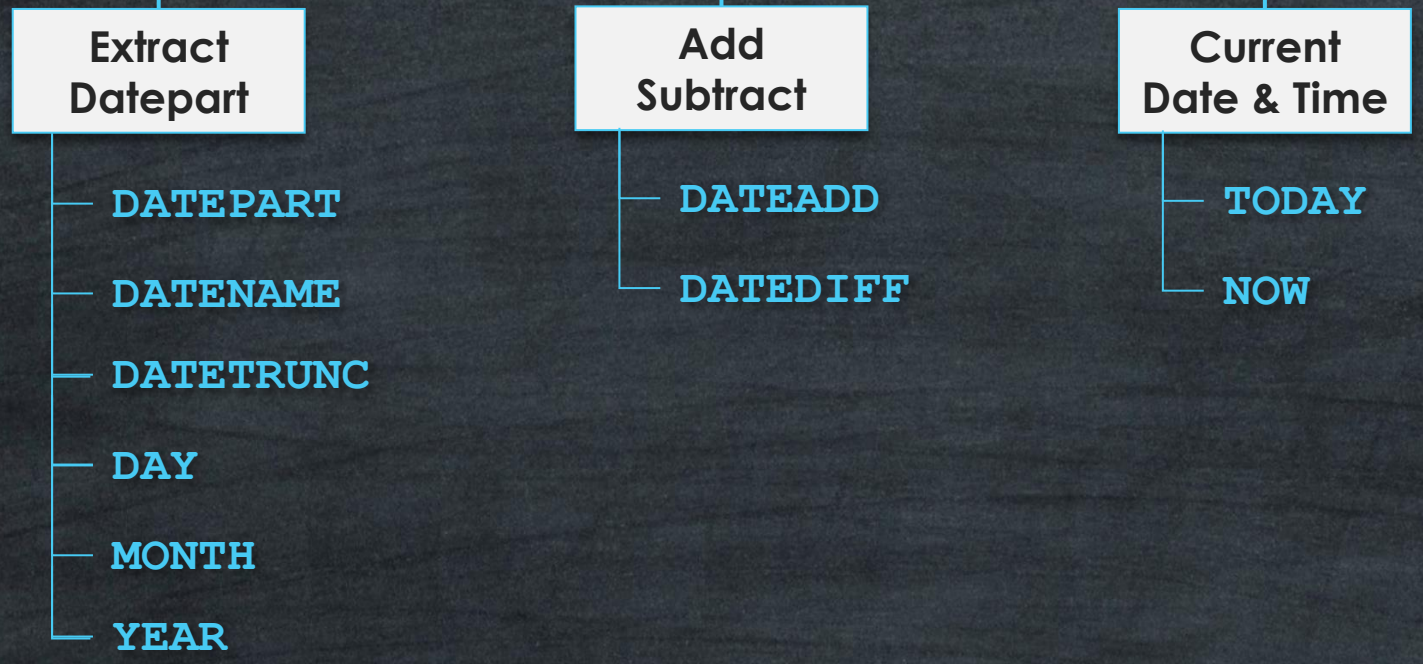
```
DATEDIFF('month',#2025-11-25#, #2026-02-01#)
```

→ 3

```
DATEDIFF('day',#2025-11-25#, #2026-02-01#)
```

→ 68

# Date Functions
## Use Cases



DATE FUNCTIONS

# TODAY, NOW

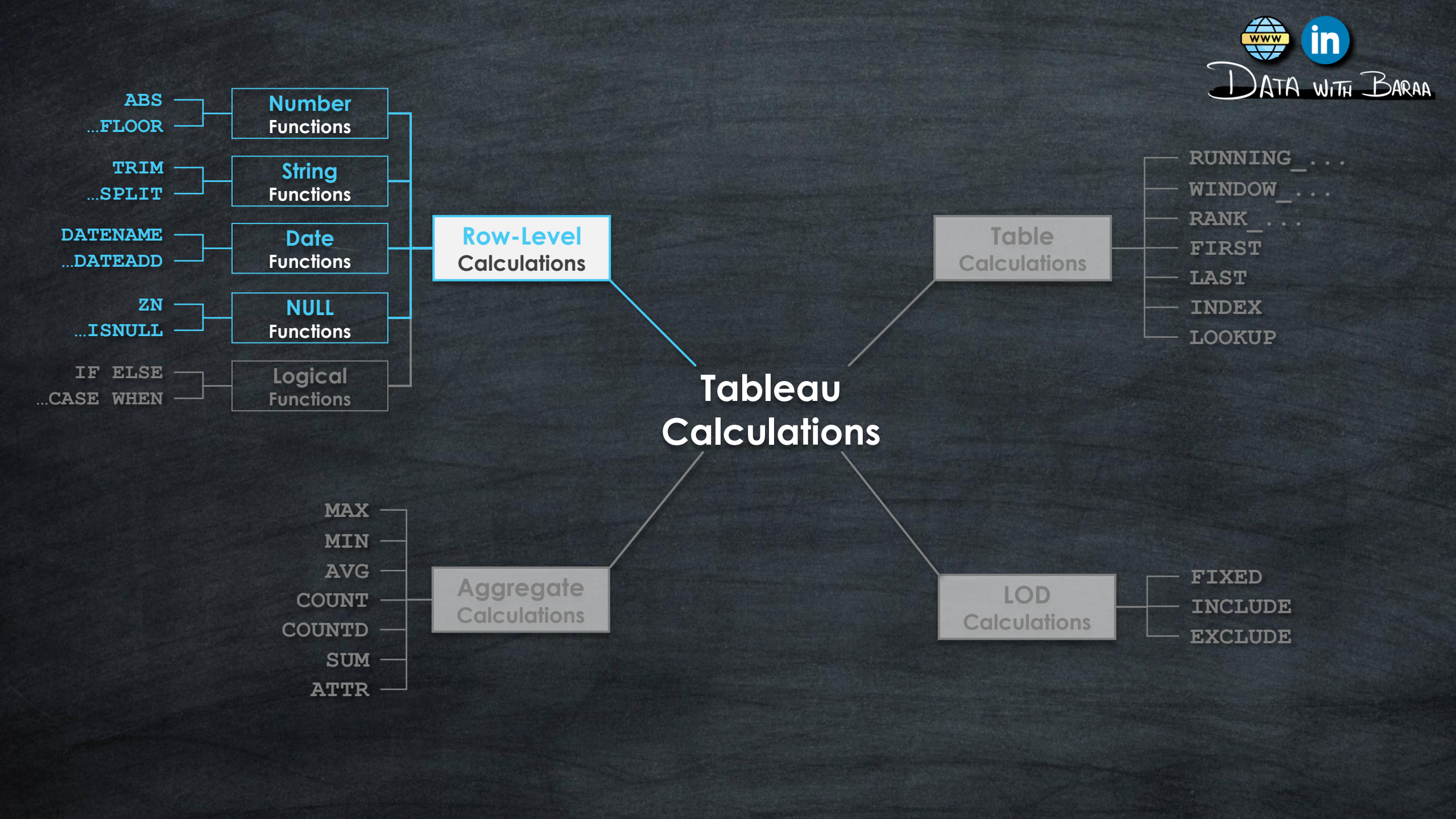
DATA WITH BARAA

DATE

```
TODAY()  →  2023-05-30
```

DATE & TIME

```
NOW()    →  2023-05-30 18:10:40
```


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# NULL Functions
Use Cases

Data with Baraa

## Main Purpose is to Handle Missing Values (NULLs)

### Calculation Accuracy

- Null Values can affect calculations and aggregations.

### Data Quality and Completness

- Identifiy data gaps, data entry, and data collection issues.

NULL FUNCTIONS

# ZN, IFNULL, ISNULL

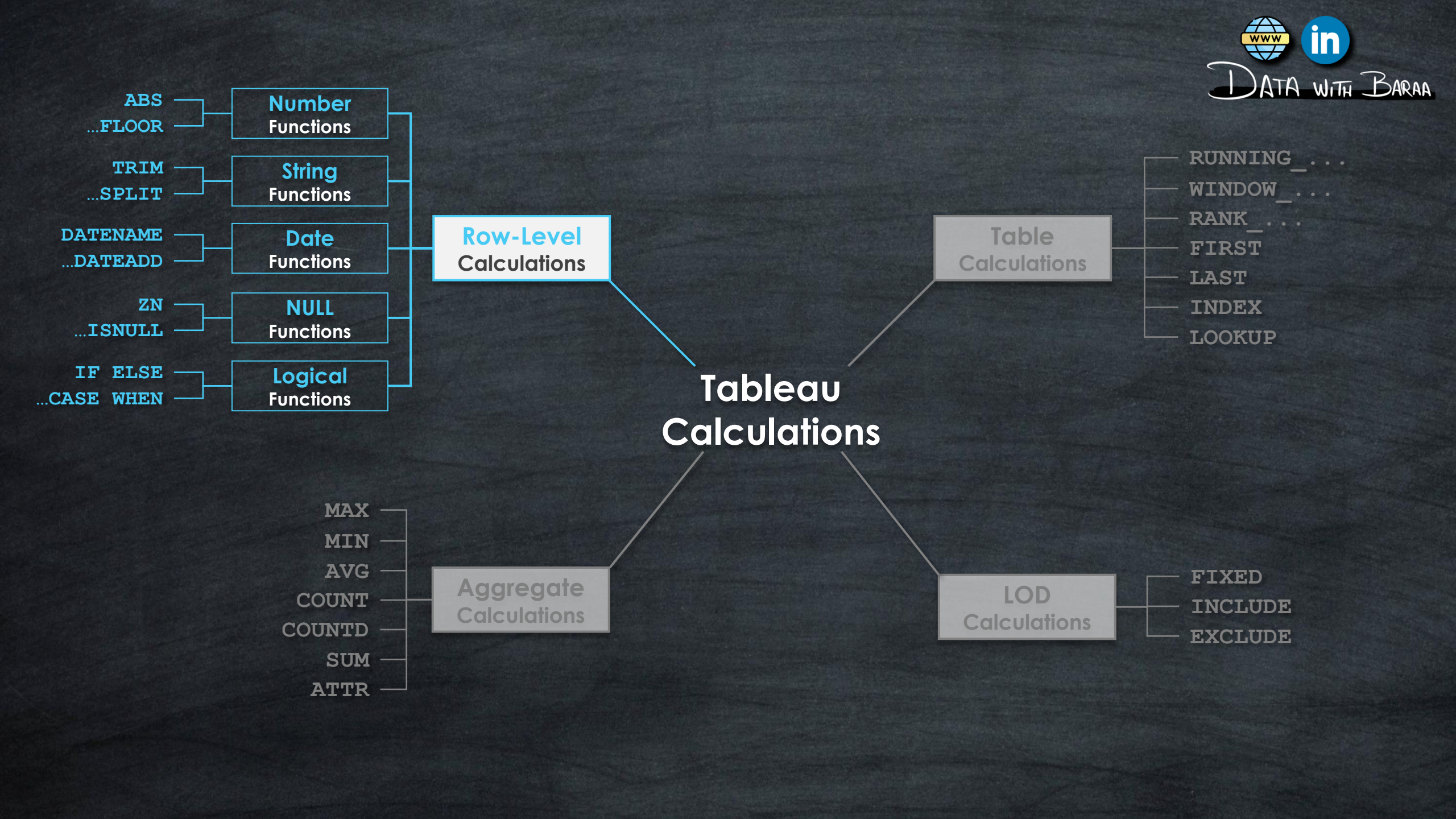
DATA WITH BARAA

**ZN** – Replace **NULL** values with **Zero**

**IFNULL** – Replace **NULL** with **Specific Value**

**ISNULL** – Return **TRUE** if value is **NULL**, and **FALSE** otherwise

| | | | Number | Any type, depends on Input | | TRUE | FALSE |
|---|---|---|---|---|---|---|---|
| Customer | Sales | Country | ZN([Sales]) | IFNULL([Sales], 0) | IFNULL([Country], "N/A") | ISNULL([Country]) | |
| John | 1800 | *NULL* | 1800 | 1800 | N/A | TRUE | |
| Maria | *NULL* | USA | 0 | 0 | USA | FALSE | |
| Martin | 350 | *NULL* | 350 | 350 | N/A | TRUE | |
| Georg | 250 | France | 250 | 250 | France | FALSE | |


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# Logical Functions
Use Cases

Data with Baraa

Main Purpose is to make logical decisions based on conditions

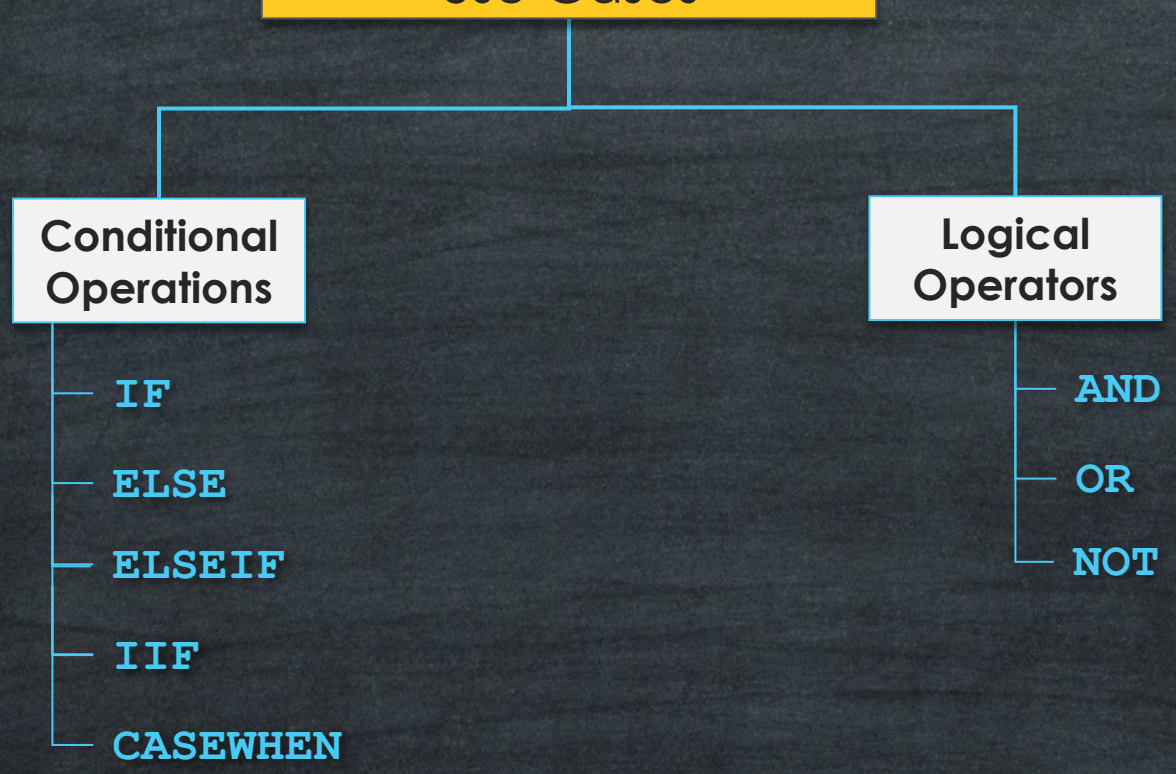
## Calculation Accuracy

- Null Values can affect calculations and aggregations.

## Data Quality and Completness

- Identifiy data gaps, data entry, and data collection issues.

# Logical Functions
## Use Cases



# Logical Functions
## Use Cases

LOGICAL FUNCTIONS

# IF

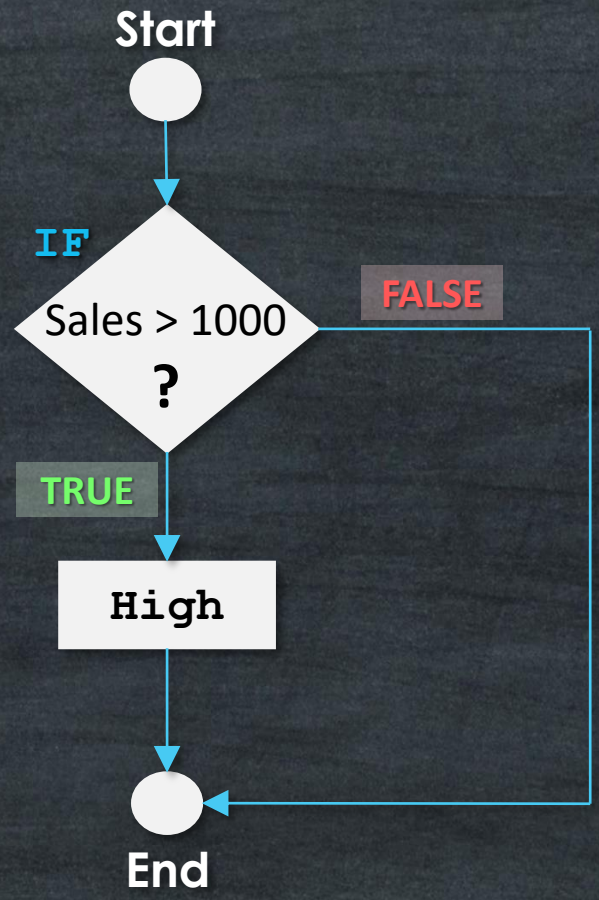
DATA WITH BARAA

```
[SALES] = 1200
```

```
IF [SALES] > 1000
   THEN "HIGH"
```

TRUE

```
END
```



LOGICAL FUNCTIONS

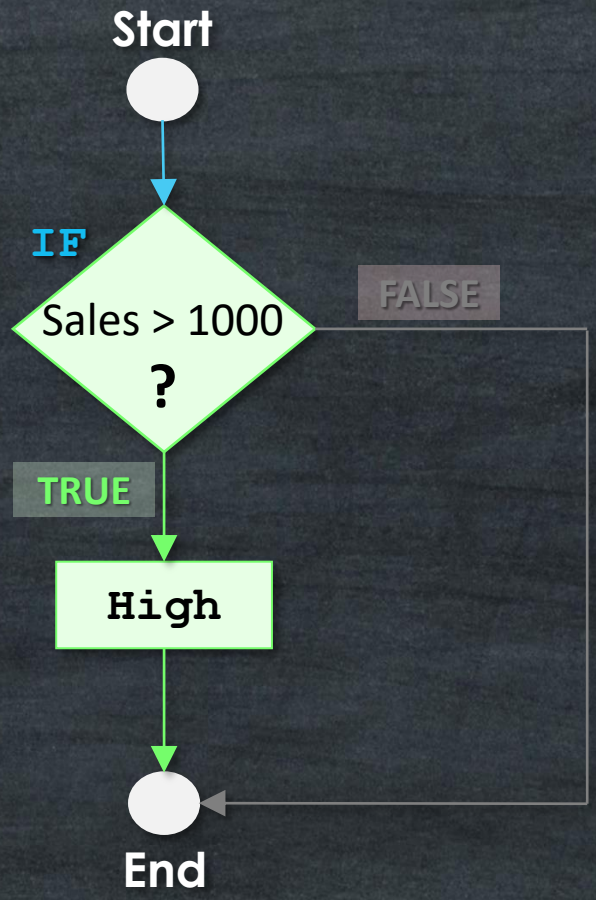
# IF

[SALES] = 1200 → High

```
IF [SALES] > 1000
   THEN "HIGH"
```

TRUE

```
END
```



LOGICAL FUNCTIONS

# IF



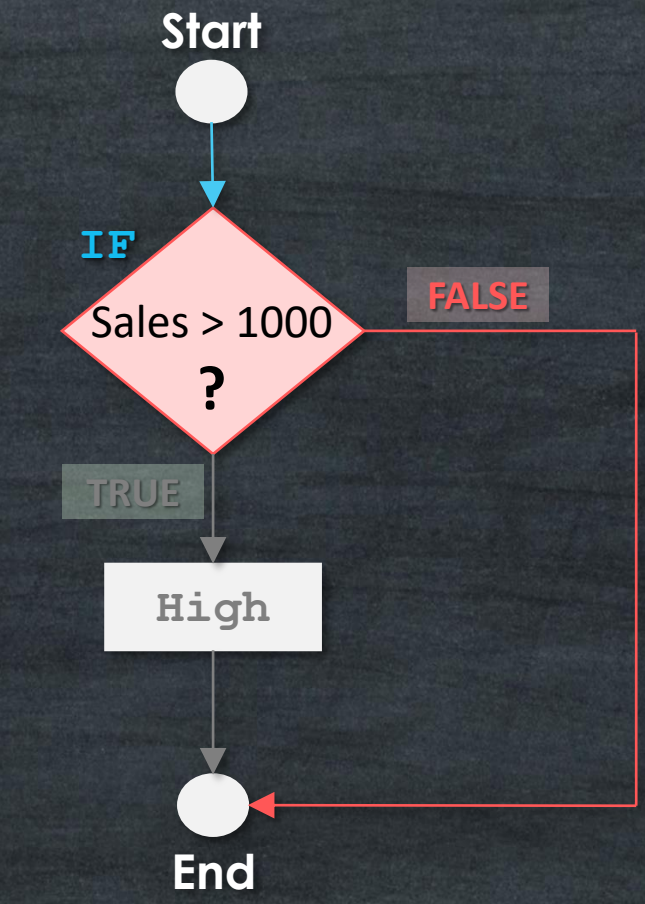
```
[SALES] = 700
```

➡ *NULL*

```
IF [SALES] > 1000
   THEN "HIGH"
```

FALSE

```
END
```



LOGICAL FUNCTIONS

# IF, ELSE

DATA WITH BARAA

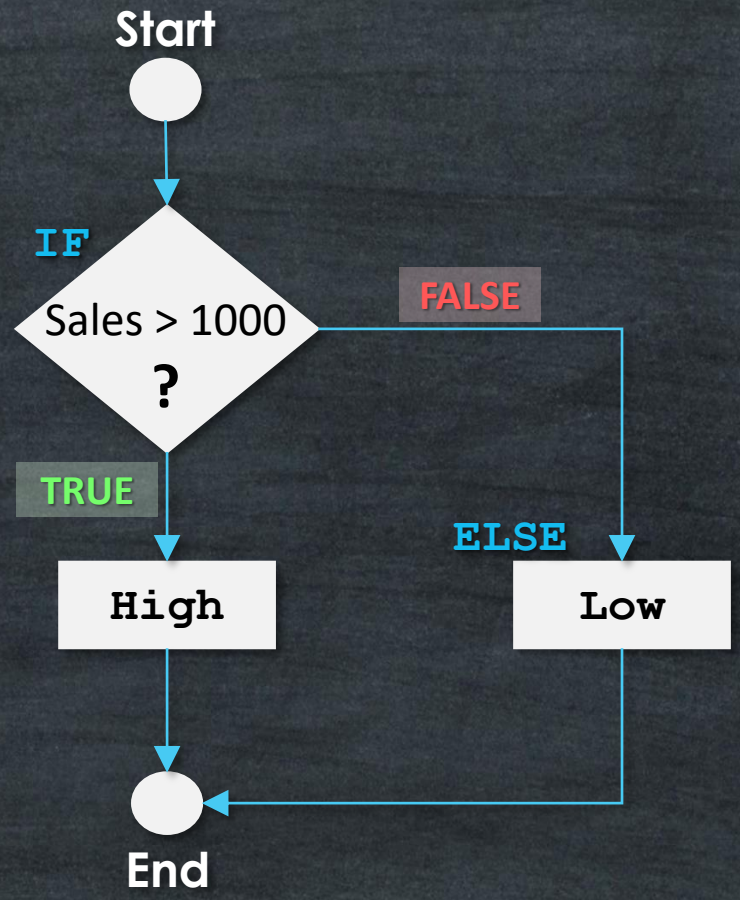
```
[SALES] = 1200
```

```
IF [SALES] > 1000
   THEN "HIGH"

ELSE "LOW"

END
```

TRUE



# IF, ELSE



```
[SALES] = 1200
```

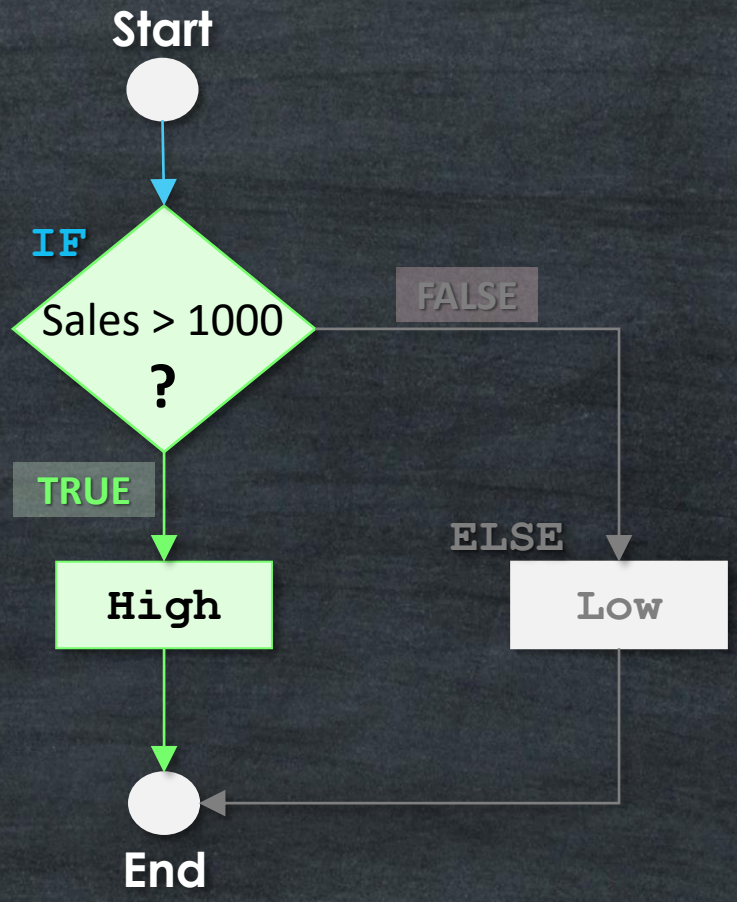
➡ High

```
IF [SALES] > 1000
  THEN "HIGH"

ELSE "LOW"

END
```

TRUE



LOGICAL FUNCTIONS

# IF, ELSE

DATA WITH BARAA

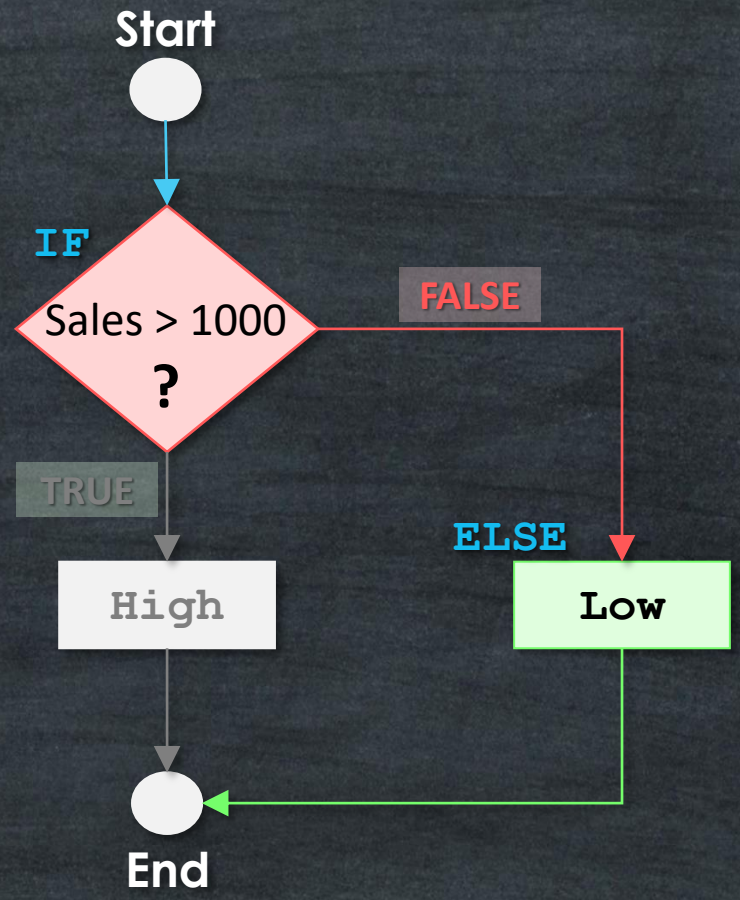
```
[SALES] = 700
```

LOW

```
IF [SALES] > 1000
   THEN "HIGH"
ELSE "LOW"

END
```

FALSE



LOGICAL FUNCTIONS

# IF, ELSE, ELSEIF

DATA WITH BARAA

[SALES] = 1200

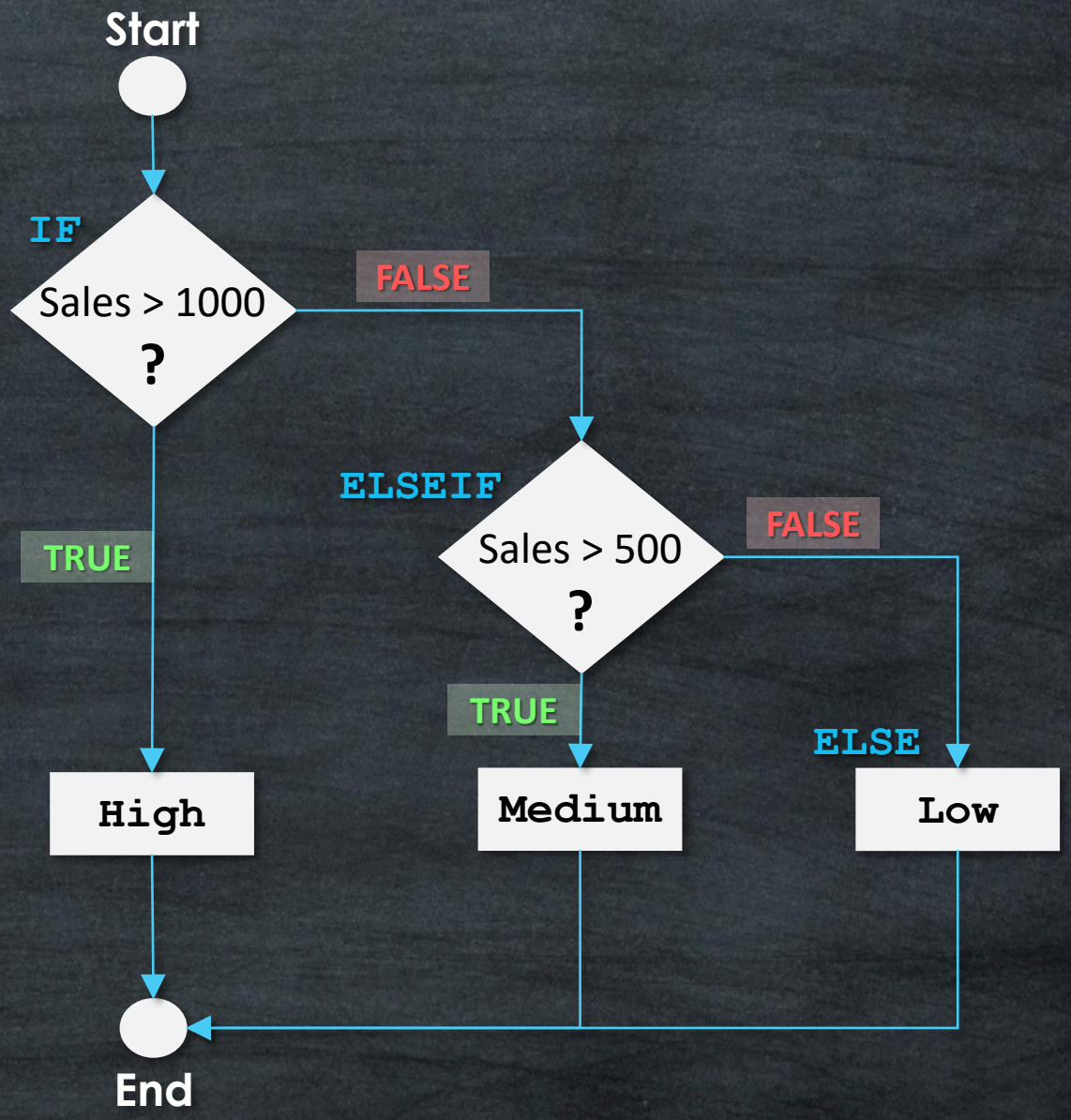
```
IF [SALES] > 1000
   THEN "HIGH"

ELSEIF [SALES] > 500
   THEN "MEDIUM"

ELSE "LOW"

END
```

TRUE



LOGICAL FUNCTIONS

# IF, ELSE, ELSEIF

DATA WITH BARAA

[SALES] = 1200 ➡ HIGH

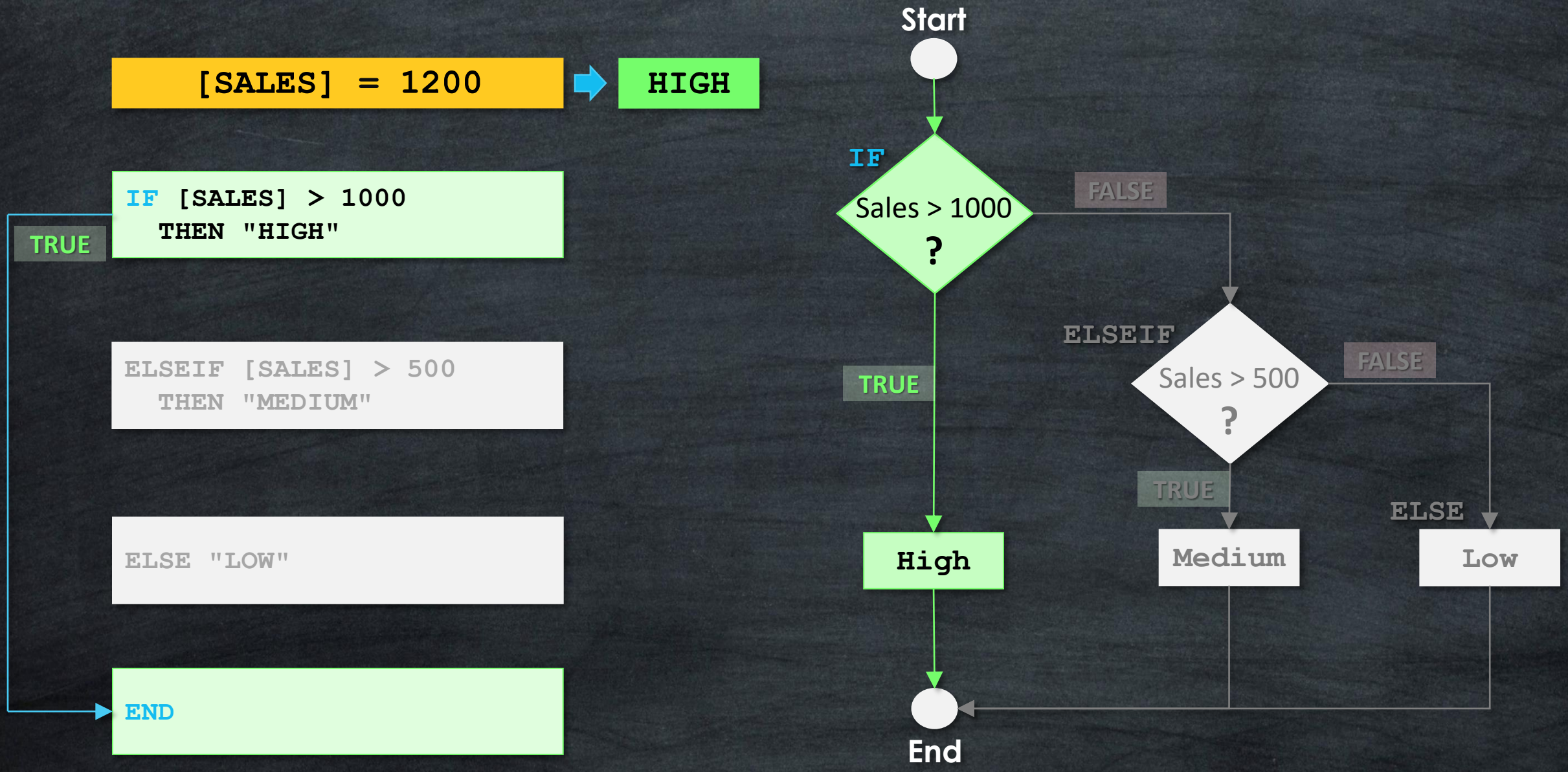
```
IF [SALES] > 1000
   THEN "HIGH"

ELSEIF [SALES] > 500
   THEN "MEDIUM"

ELSE "LOW"

END
```

TRUE



LOGICAL FUNCTIONS

# IF, ELSE, ELSEIF

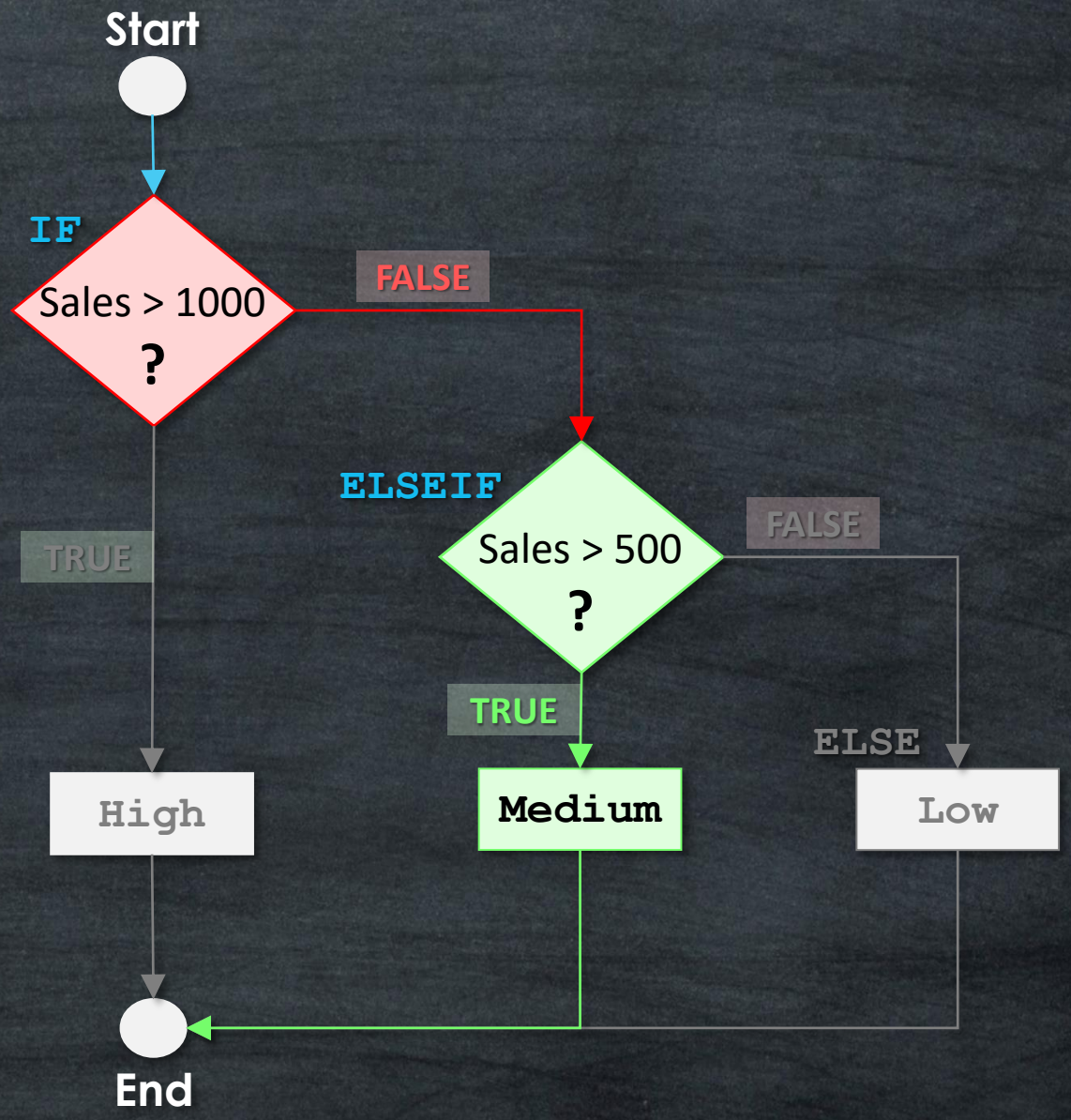
DATA WITH BARAA

[SALES] = 700 → MEDIUM

```
IF [SALES] > 1000
   THEN "HIGH"
ELSEIF [SALES] > 500
   THEN "MEDIUM"
ELSE "LOW"
END
```

FALSE

TRUE



LOGICAL FUNCTIONS

# IF, ELSE, ELSEIF

DATA WITH BARAA

[SALES] = 350 ➡ LOW

```
IF [SALES] > 1000
   THEN "HIGH"
```

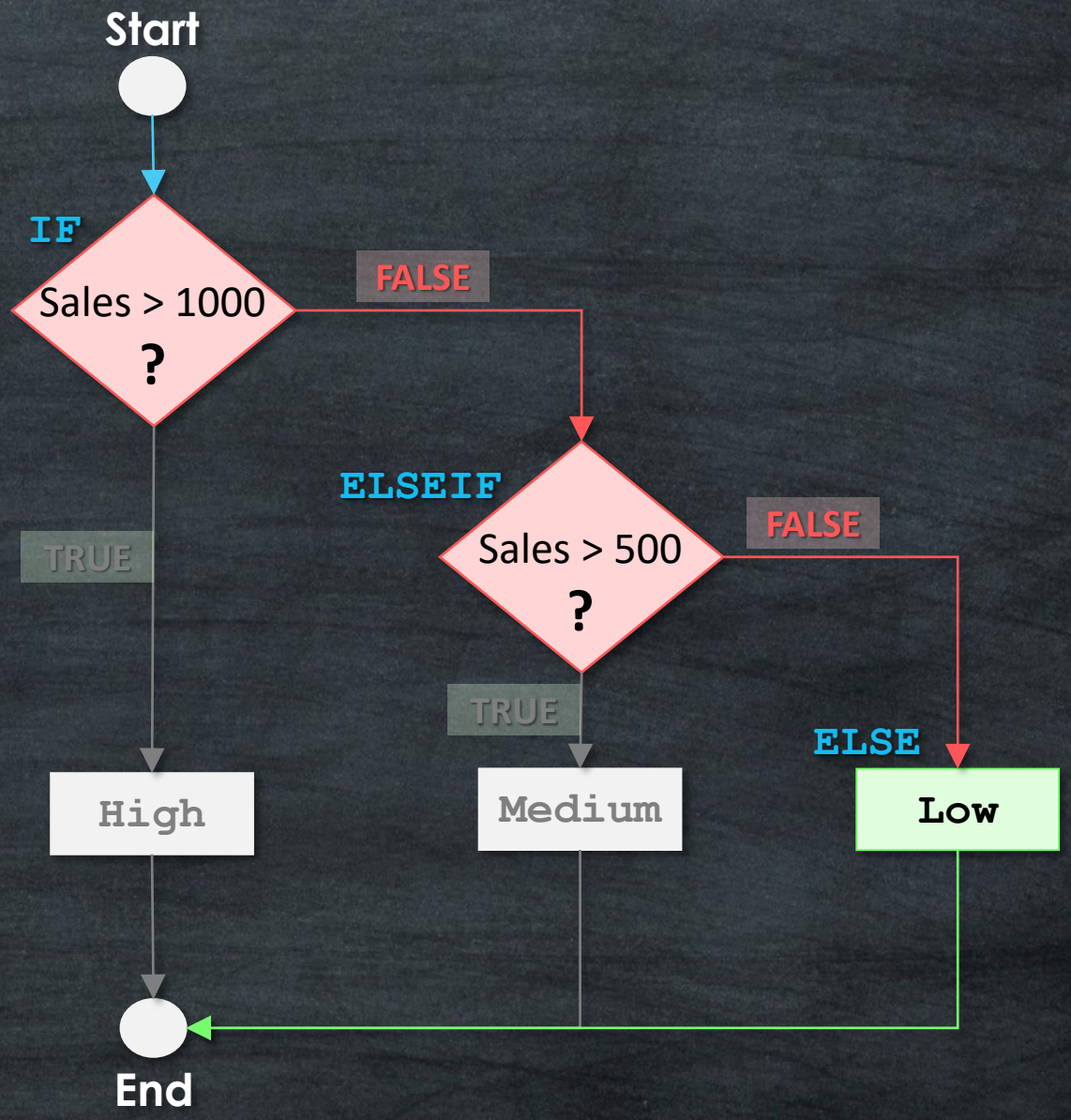
FALSE

```
ELSEIF [SALES] > 500
   THEN "MEDIUM"
```

FALSE

```
ELSE "LOW"
```

```
END
```



# CASE WHEN



```
[Country] = "Germany"
```

→ DE

```
CASE [Country]
WHEN "Germany" THEN "DE"
WHEN "France" THEN "FR"
WHEN "USA" THEN "US"
ELSE "n/a"
END
```

TRUE

# CASE WHEN



```
[Country] = "France"
```

→ FR

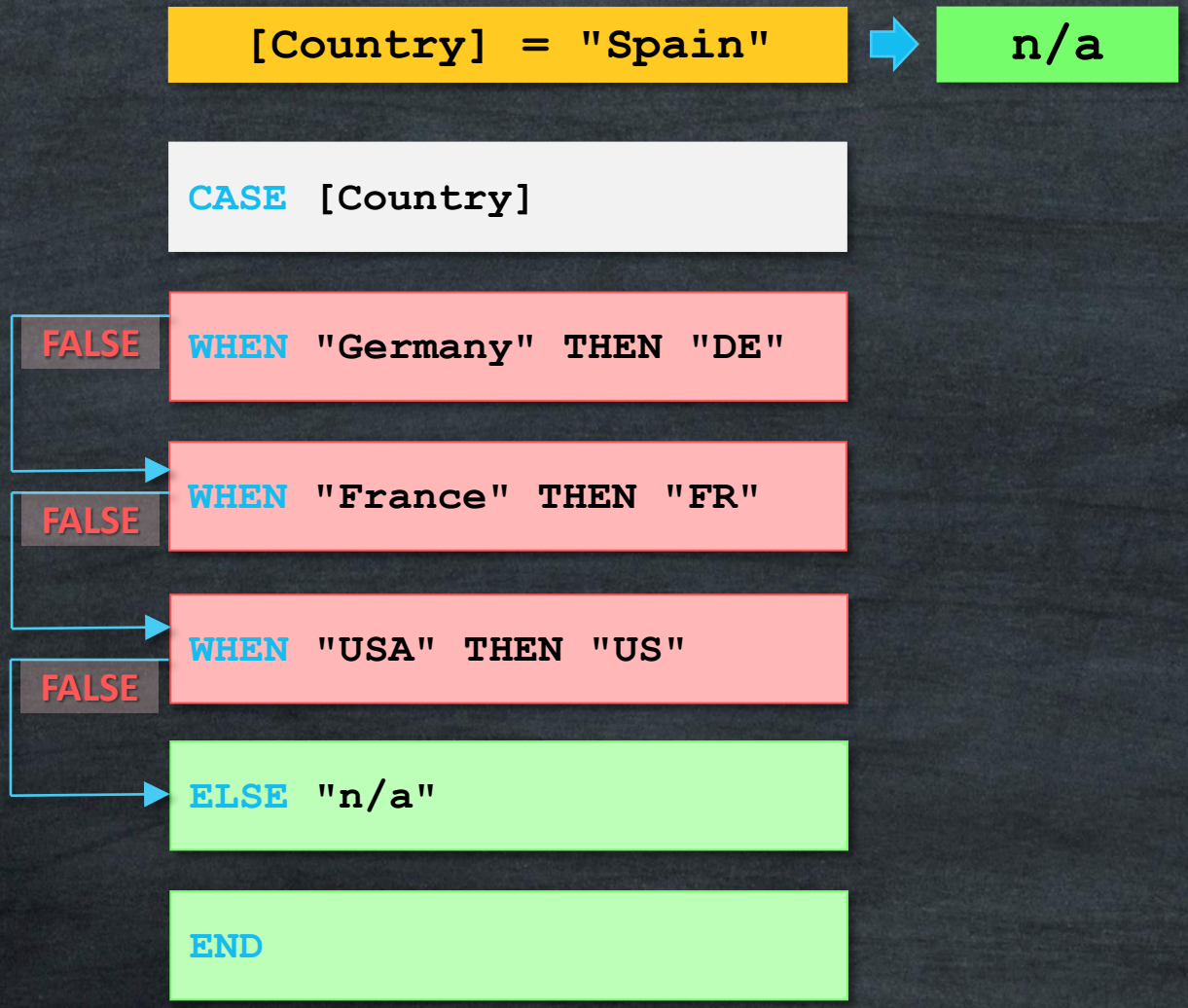
```
CASE [Country]
WHEN "Germany" THEN "DE"
WHEN "France" THEN "FR"
WHEN "USA" THEN "US"
ELSE "n/a"
END
```

FALSE

TRUE

# CASE WHEN





## IF, ELSEIF

```
IF [SALES] > 1000
    THEN "HIGH"
ELSEIF [SALES] > 500
    THEN "MEDIUM"
ELSE "LOW"
END
```

Supports Multiple Conditions

Evalulte Multiple Fields

Supports Any Data Type

No Limitations

## IIF

```
IIF([Sales] > 1000, "HIGH", "LOW")
```

Supports Only One Conditions

Evalulte Multiple Fields

Supports Any Data Type

Easy to Write

## CASE WHEN

```
CASE [Country]
WHEN "Germany" THEN "DE"
WHEN "France" THEN "FR"
ELSE "U/A"
END
```

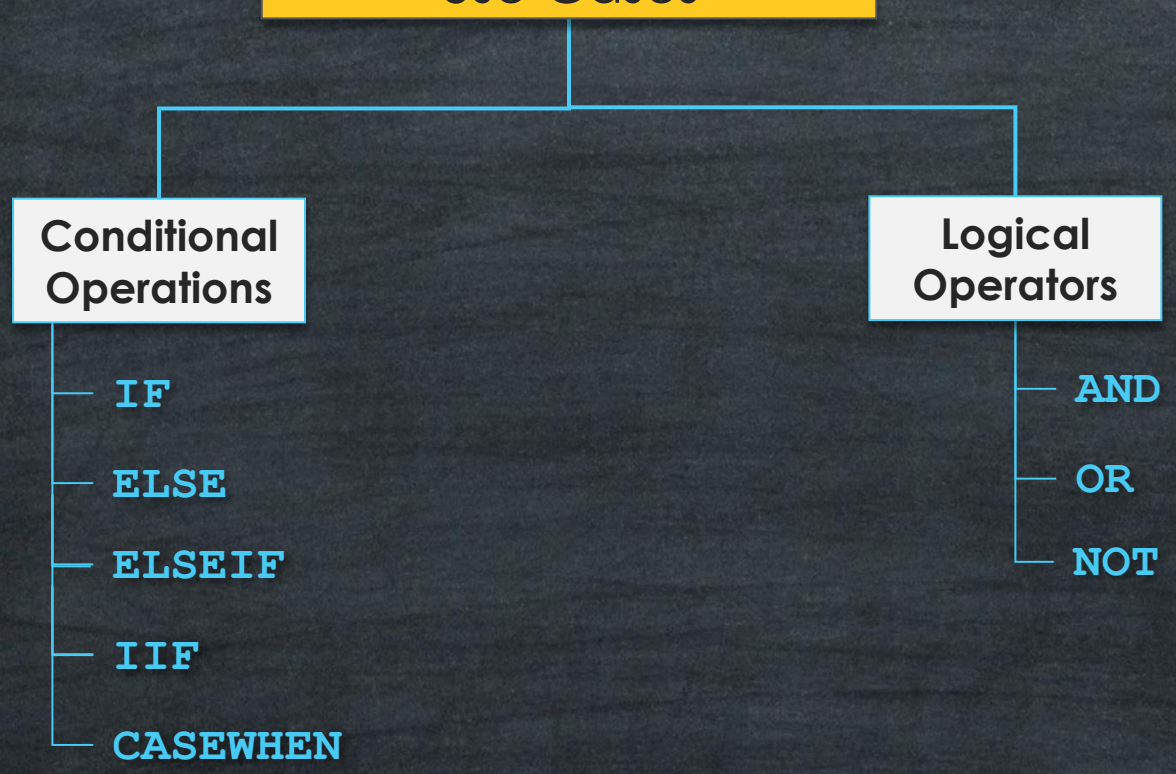
Supports Multiple Conditions

Evalulte Only One Dimension

Supports Only Strings

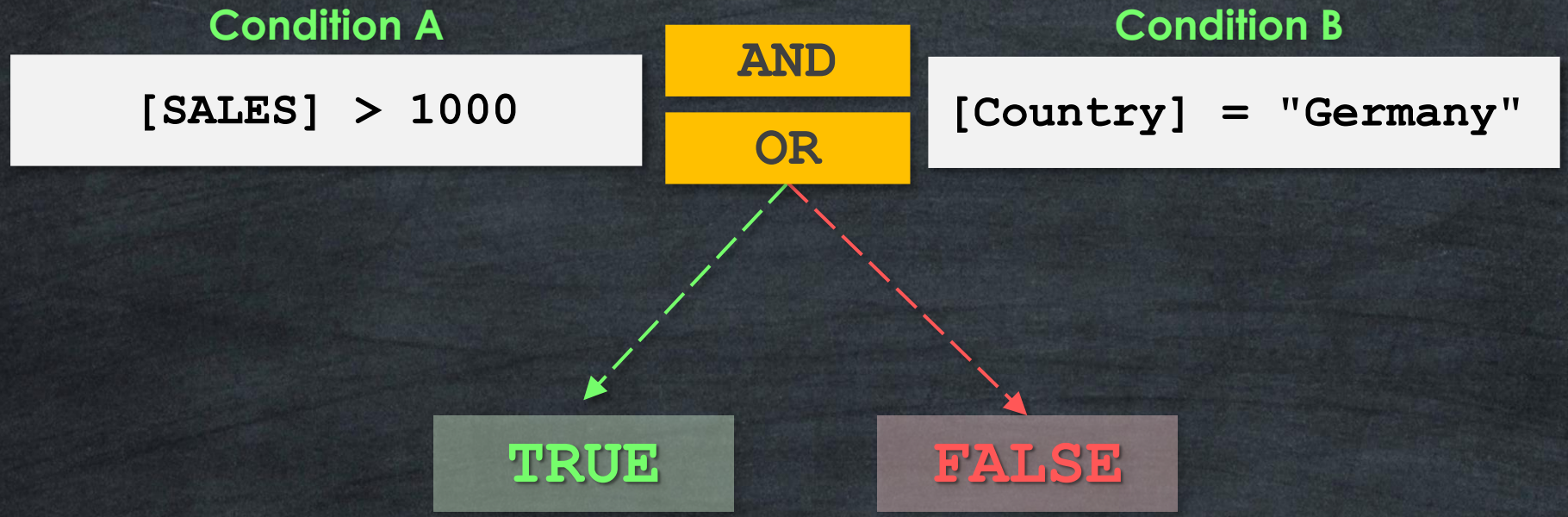
Easy to Write & Read

# Logical Functions
## Use Cases



# AND, OR






AND/OR logical operators are used to combine multiple conditions

# AND, OR



## AND Syntax

```
IF [SALES] > 1000 AND [Country] = "Germany"
   THEN "HIGH"
END
```

## OR Syntax

```
IF [SALES] > 1000 OR [Country] = "Germany"
   THEN "HIGH"
END
```

LOGICAL FUNCTIONS

# AND, OR

DATA WITH BARAA

**AND** - Returns TRUE if both conditions are TRUE, and FALSE otherwise

**OR** - Returns TRUE if at least one condition is TRUE, and FALSE otherwise

| Customer | Sales | Country | Condition A [Sales] > 1000 | Condition B [Country] = "Germany" | A AND B | A OR B |
|---|---|---|---|---|---|---|
| John | 1800 | Germany | TRUE | TRUE | TRUE | TRUE |
| Maria | 1250 | USA | TRUE | FALSE | FALSE | TRUE |
| Martin | 350 | Germany | FALSE | TRUE | FALSE | TRUE |
| Georg | 400 | France | FALSE | FALSE | FALSE | FALSE |

# NOT



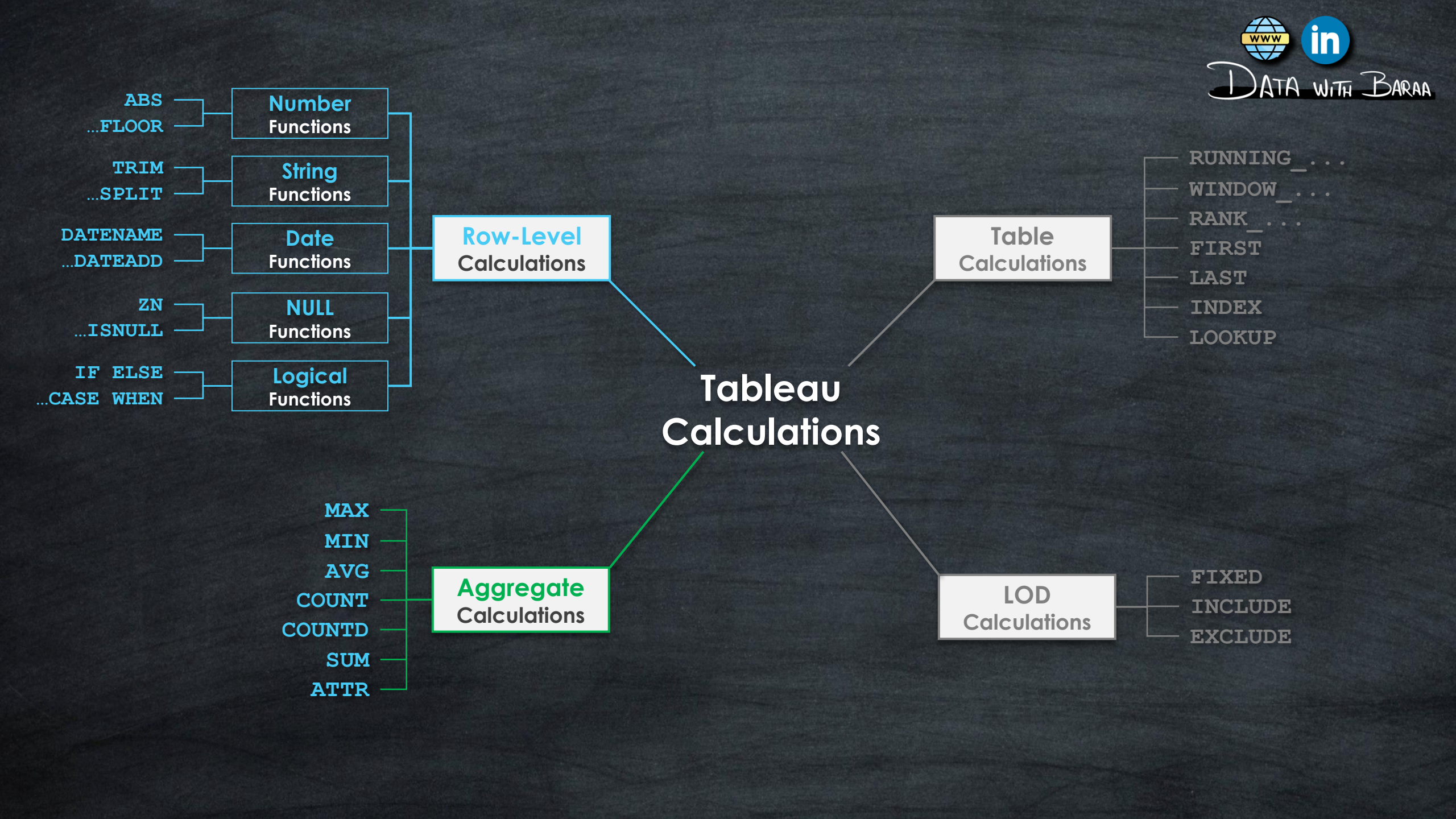
**NOT – Reverse Logical Operator:**

- Return TRUE if the condition is FALSE
- Return FALSE if the condition is TRUE

| Customer | Sales | Country | Condition A [Sales] > 1000 | NOT A |
|---|---|---|---|---|
| John | 1800 | Germany | TRUE | FALSE |
| Maria | 1250 | USA | TRUE | FALSE |
| Martin | 350 | Germany | FALSE | TRUE |
| Georg | 400 | France | FALSE | TRUE |

**NOT Syntax**

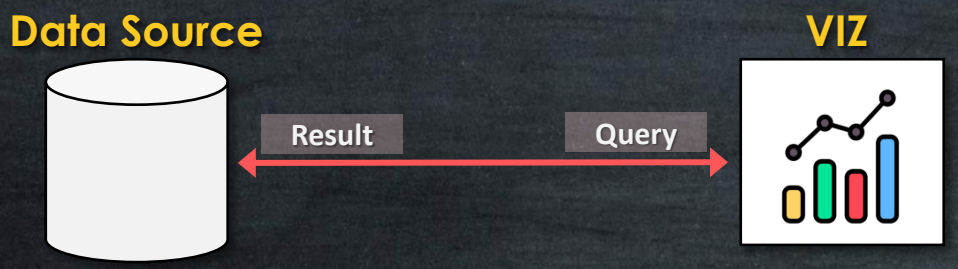
```
IF NOT [Sales] > 1000
    THEN "Low"
END
```


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

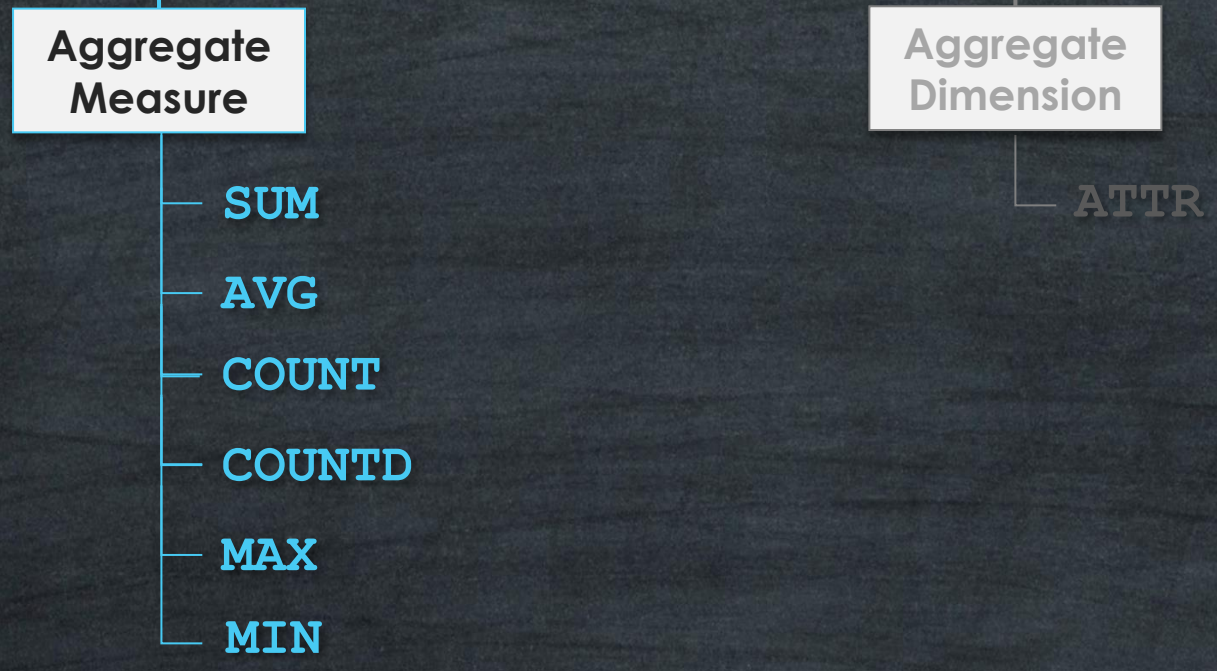
# Aggregate
## Calculations



- Aggregate the rows at the dimension level used in the VIZ
- Level of Details is the Visualization | VIZ LOD
- The calculations are performed on the data within the data source
- Results will be calculated on the FLY



# Aggregate Functions
## Use Cases



# Aggregate Calculations

DATA WITH BARAA

## Aggregate the data at the visualization level of details (VIZ LOD)

| Order ID | Product | Customer | Sales |
|---|---|---|---|
| 1 | P-1 | John | 5 |
| 2 | P-1 | John | 15 |
| 3 | P-1 | John | 20 |
| 4 | P-2 | John | 15 |
| 5 | P-2 | Maria | 30 |
| 6 | P-2 | Maria | 5 |
| 7 | P-3 | John | 25 |
| 8 | P-3 | Maria | 20 |
| 9 | P-4 | Maria | 10 |
| 10 | P-4 | Maria | 5 |

Data Source

```
SUM([Sales])
```

| Product | Sales |
|---|---|
| P-1 | 40 |
| P-2 | 50 |
| P-3 | 45 |
| P-4 | 15 |

View

# Aggregate Calculations

DATA WITH BARAA

## Aggregate the data at the visualization level of details (VIZ LOD)

| | | Syntax |
|---|---|---|
| SUM | Returns the total sum of all values | SUM([Sales]) |
| AVG | Returns the average of all values | AVG([Sales]) |
| COUNT | Counts the number of values | COUNT([Sales]) |
| COUNTD | Counts the number of unique values | COUNTD([Sales]) |
| MAX | Returns the maximum value | MAX([Sales]) |
| MIN | Returns the minimum value | MIN([Sales]) |

# Aggregate Functions
Use Cases

# Aggregate Calculations

DATA WITH BARAA

## Aggregate Dimension's values?

| Order ID | Product | Customer | Sales |
|---|---|---|---|
| 1 | P-1 | John | 5 |
| 2 | P-1 | John | 15 |
| 3 | P-1 | John | 20 |
| 4 | P-2 | John | 15 |
| 5 | P-2 | Maria | 30 |
| 6 | P-2 | Maria | 5 |
| 7 | P-3 | John | 25 |
| 8 | P-3 | Maria | 20 |
| 9 | P-4 | Maria | 10 |
| 10 | P-4 | Maria | 5 |

Data Source

```
SUM([Sales])
```

| Product | Sales |
|---|---|
| P-1 | 40 |
| P-2 | 50 |
| P-3 | 45 |
| P-4 | 15 |

View

# Attribute – ATTR()

## Attribute Function ATTR () aggregates the values of Dimensions

- If all values are same, then it returns single value
- If there are multiple values, then it returns Asterisk *

| Order ID | Product | Customer | Sales |
|---|---|---|---|
| 1 | P-1 | John | 5 |
| 2 | P-1 | John | 15 |
| 3 | P-1 | John | 20 |
| 4 | P-2 | John | 15 |
| 5 | P-2 | Maria | 30 |
| 6 | P-2 | Maria | 5 |
| 7 | P-3 | John | 25 |
| 8 | P-3 | Maria | 20 |
| 9 | P-4 | Maria | 10 |
| 10 | P-4 | Maria | 5 |

Data Source

```
ATTR([Customer])
```

| Product | Customer |
|---|---|
| P-1 | John |
| P-2 | * |
| P-3 | * |
| P-4 | Maria |

View

# Attribute – ATTR()

SUM() aggregate Measures

ATTR() aggregate Dimensions

| Dimensions | | | Measure |
|---|---|---|---|
| Order ID | Product | Customer | Sales |
| 1 | P-1 | John | 5 |
| 2 | P-1 | John | 15 |
| 3 | P-1 | John | 20 |
| 4 | P-2 | John | 15 |
| 5 | P-2 | Maria | 30 |
| 6 | P-2 | Maria | 5 |
| 7 | P-3 | John | 25 |
| 8 | P-3 | Maria | 20 |
| 9 | P-4 | Maria | 10 |
| 10 | P-4 | Maria | 5 |

Data Source

```
ATTR([Customer])
```

Aggregate Dimensions

| Product | Customer |
|---|---|
| P-1 | John |
| P-2 | * |
| P-3 | * |
| P-4 | Maria |

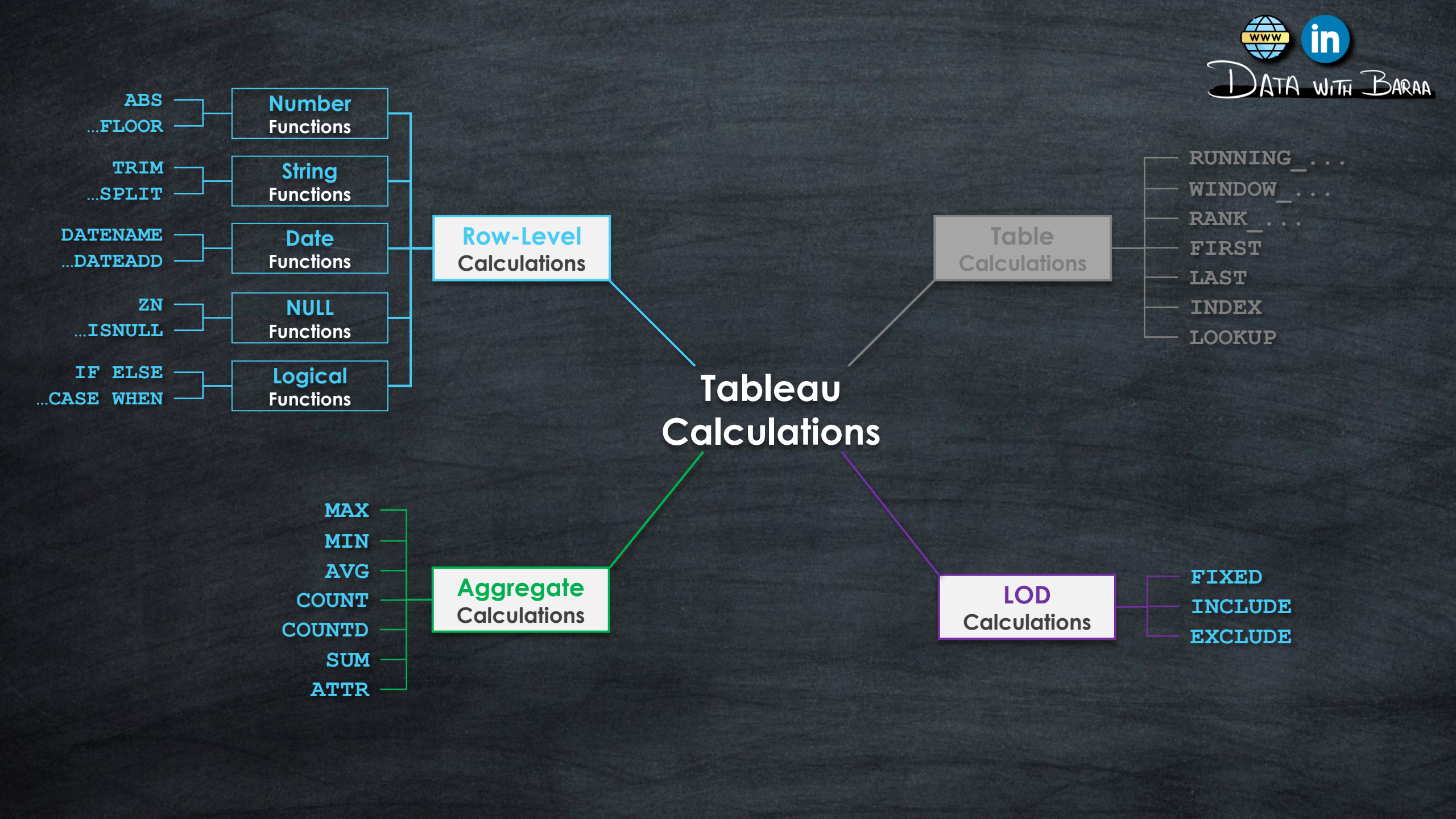
View

```
SUM([Sales])
```

Aggregate Measures

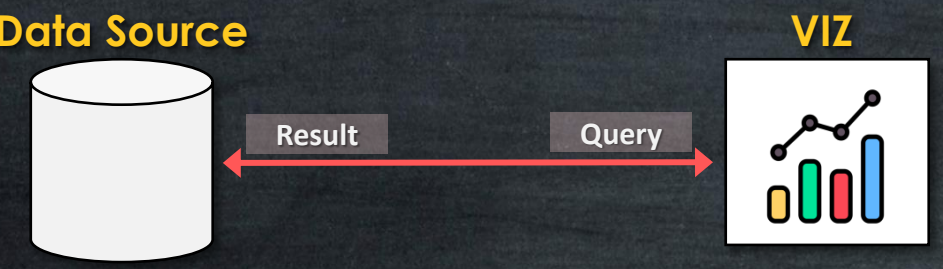
| Product | Sales |
|---|---|
| P-1 | 40 |
| P-2 | 50 |
| P-3 | 45 |
| P-4 | 15 |

View


Data with Baraa
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# LOD
## Calculations

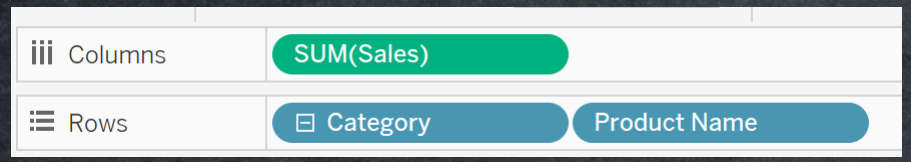
- Aggregate the rows at the dimension level used in the calculation
- Level of Details is the LOD Expression
- The calculations are performed on the data within the data source
- Results will be calculated on the FLY



# Level Of Details (LOD)

DATA WITH BARAA

## View




## Syntax

```
{ FIXED [Category], [Product Name] : SUM([Sales]) }
```

Scoping FIXED | INCLUDE | EXCLUDE

List of Dimensions

Aggregation

## Examples

```
{ EXCLUDE [Category] : SUM([Sales]) }
```

```
{ FIXED : SUM([Sales]) }
```

```
{ INCLUDE [Customer ID] : AVG([Sales]) }
```

# Level Of Details (LOD)

DATA WITH BARAA

## Syntax

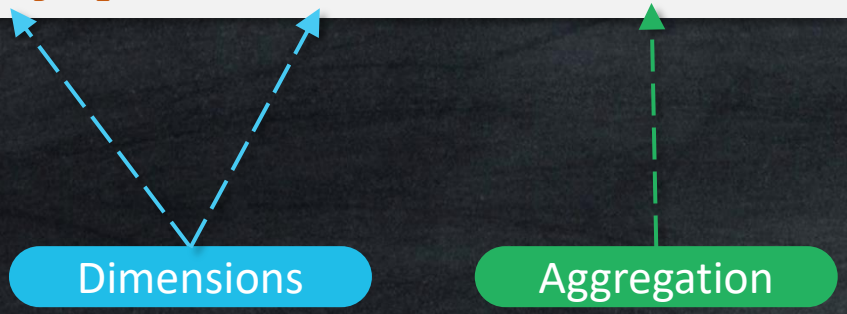
```
{ FIXED|INCLUDE|EXCLUDE <List of Dimensions> : <Aggregation> }
```

## LOD Expression

```
{ FIXED : SUM([Sales])}
```

```
{ FIXED [Category] : SUM([Sales])}
```

```
{ FIXED [Category],[Product Name] : SUM([Sales])}
```




## View

SUM(Sales)

Category | SUM(Sales)

Category | Product Name | SUM(Sales)

Level 1

Level 2

Level 3

Level 4

Level 5

No Dimensions

Category

Country

Product Name

Order ID

Low

High

Level of Details (Granularity)

Total Sales = 6.506

| Category | |
|---|---|
| Accessories | 2.377 |
| Monitor | 4.129 |

| Country | |
|---|---|
| France | 172 |
| Germany | 2.774 |
| USA | 3.560 |

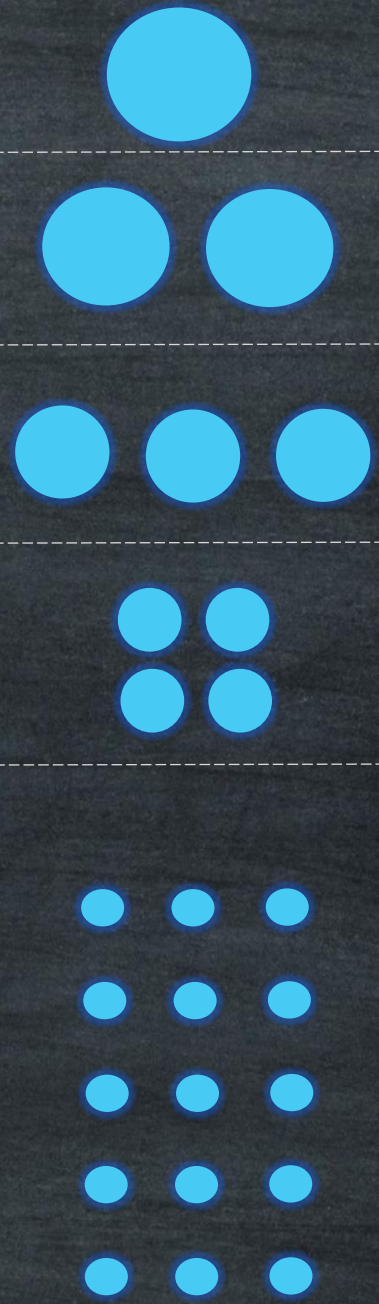
| Product Name | |
|---|---|
| Cherry USB Mouse | 672 |
| LG FULL HD Monitor | 3.182 |
| Logitech Keyboard K120 | 1.705 |
| Samsung FHD Monitor | 948 |

| Order ID | |
|---|---|
| 1 | 216 |
| 2 | 1.691 |
| 3 | 40 |
| 4 | 501 |
| 5 | 164 |
| 6 | 430 |
| 7 | 91 |
| 8 | 375 |
| 9 | 525 |
| 10 | 56 |
| 11 | 221 |
| 12 | 1.491 |
| 13 | 41 |
| 14 | 451 |
| 15 | 214 |

High

Low

Level of Aggregations



Level 1

Level 2

Level 3

Level 4

Level 5

| Total Sales = 6.506 |
|---|

| Category | |
|---|---|
| Accessories | 2.377 |
| Monitor | 4.129 |

| Country | |
|---|---|
| France | 172 |
| Germany | 2.774 |
| USA | 3.560 |

| Product Name | |
|---|---|
| Cherry USB Mouse | 672 |
| LG FULL HD Monitor | 3.182 |
| Logitech Keyboard K120 | 1.705 |
| Samsung FHD Monitor | 948 |

| Order ID | |
|---|---|
| 1 | 216 |
| 2 | 1.691 |
| 3 | 40 |
| 4 | 501 |
| 5 | 164 |
| 6 | 430 |
| 7 | 91 |
| 8 | 375 |
| 9 | 525 |
| 10 | 56 |
| 11 | 221 |
| 12 | 1.491 |
| 13 | 41 |
| 14 | 451 |
| 15 | 214 |

Lowest level of Details
Highest Level of Aggregation

Highest level of Details
Lowest Level of Aggregation

Exclude/Fixed LOD

Current View LOD

Include/Fixed LOD

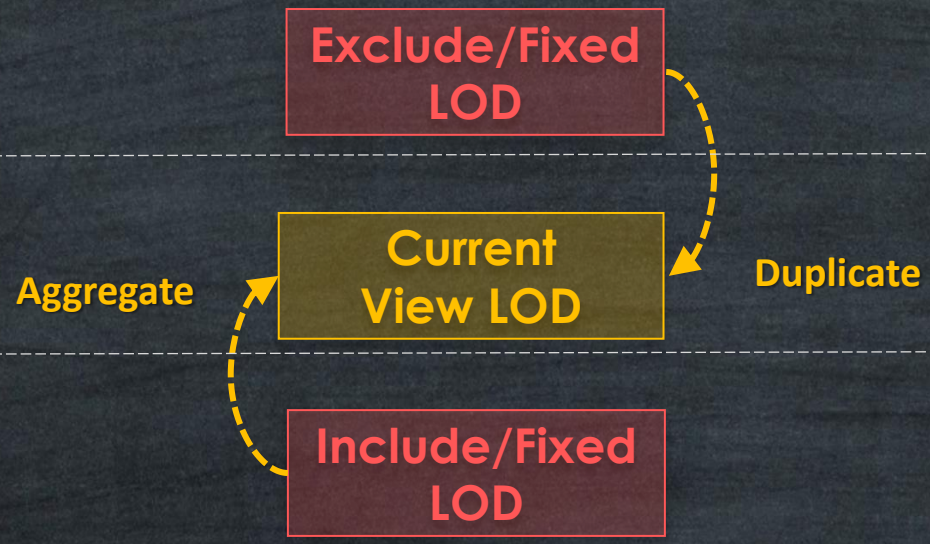
Aggregate

Duplicate

DATA WITH BARAA

Lowest level of Details

IF Level of details in LOD-Expression Lower than LOD in VIZ ➡ Duplicate Results




IF Level of details in LOD-Expression Higher than LOD in VIZ ➡ Aggregate Results

Highest level of Details

# LOD (VIZ)

View 1

| Category | Country | Sales |
|---|---|---|
| | | |

View 2

| Product | Category | Country | Sales |
|---|---|---|---|
| | | | |

# LOD (Expression)

```
FIXED [Category] : SUM([Sales])
```

| Category | Sales |
|---|---|
| | |

| Category | Sales |
|---|---|
| | |

# LOD (VIZ)

# LOD (Expression)

EXCLUDE [Category] : SUM ([Sales])

View 1

| Category | Country | Sales |
|---|---|---|
| | | |

-1 DIM

| Category | Country | Sales |
|---|---|---|
| | | |

View 2

| Product | Category | Country | Sales |
|---|---|---|---|
| | | | |

-1 DIM

| Product | Category | Country | Sales |
|---|---|---|---|
| | | | |

# LOD (VIZ)

# LOD (Expression)

INCLUDE [Customer] : SUM([Sales])

View 1

| Category | Country | Sales |
|---|---|---|
| | | |

+1 DIM

| Customer |
|---|
| |

| Category | Country | Sales |
|---|---|---|
| | | |

View 2

| Product | Category | Country | Sales |
|---|---|---|---|
| | | | |

+1 DIM

| Customer |
|---|
| |

| Product | Category | Country | Sales |
|---|---|---|---|
| | | | |




DATA WITH BARAA

# LOD (VIZ)

View 1

| Category | Country | Sales |
|---|---|---|
| | | |

# LOD (Expression)

**FIXED**

| Category | Sales |
|---|---|
| | |

**EXCLUDE**

-1 DIM

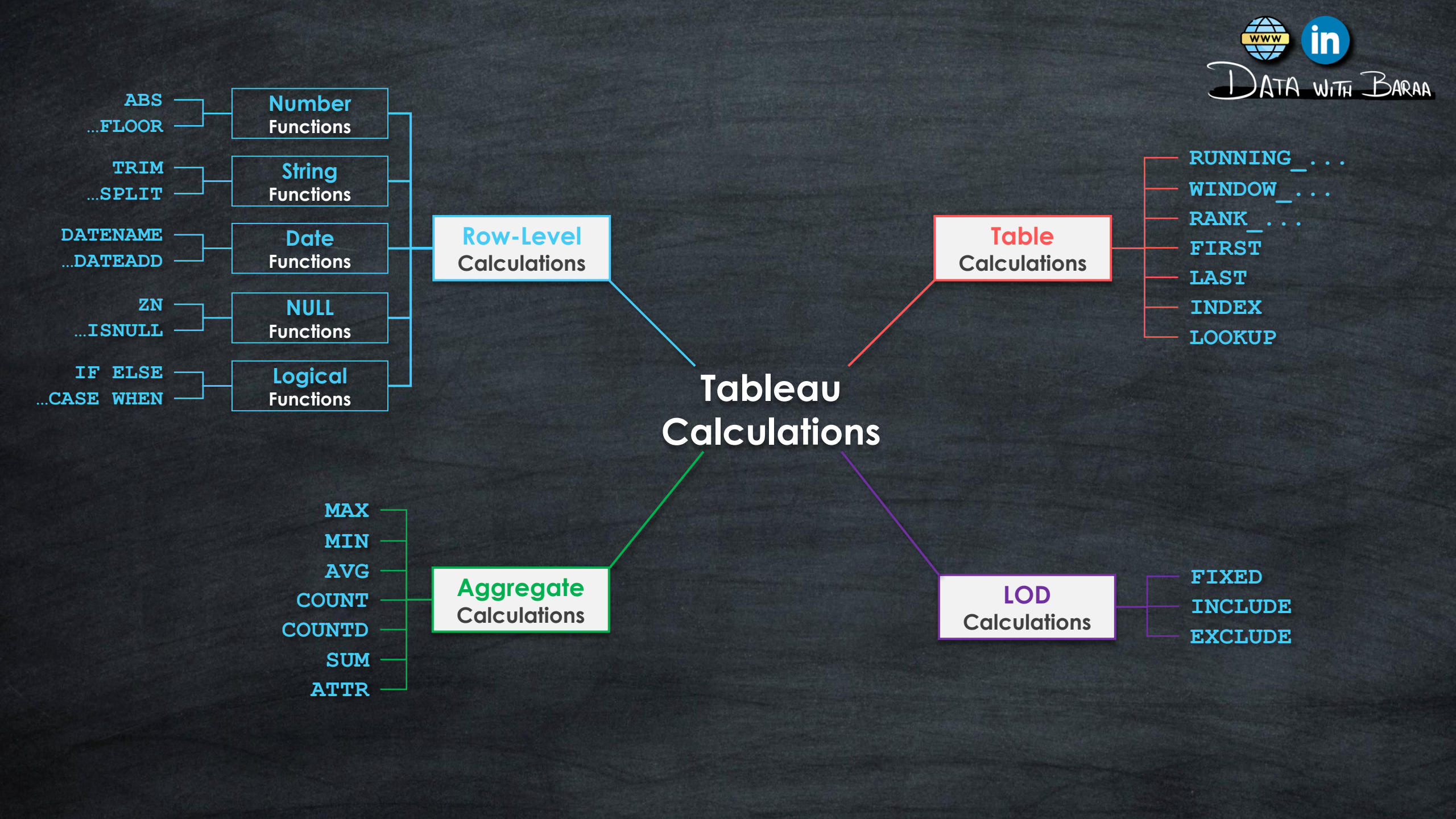
| Category | Country | Sales |
|---|---|---|
| | | |

**INCLUDE**

+1 DIM

| Customer |
|---|
| |

| Category | Country | Sales |
|---|---|---|
| | | |


DATA WITH BARAA
Tableau Calculations
Row-Level Calculations
Number Functions
ABS
...FLOOR
String Functions
TRIM
...SPLIT
Date Functions
DATENAME
...DATEADD
NULL Functions
ZN
...ISNULL
Logical Functions
IF ELSE
...CASE WHEN
Table Calculations
RUNNING_...
WINDOW_...
RANK_...
FIRST
LAST
INDEX
LOOKUP
Aggregate Calculations
MAX
MIN
AVG
COUNT
COUNTD
SUM
ATTR
LOD Calculations
FIXED
INCLUDE
EXCLUDE

# Table
## Calculations

- Table Calculations are calculated after the aggregation
- Aggregate the Aggregation!
- Level of Details is the Visualization | VIZ LOD
- The calculations are performed on the data displayed in the Visualization
- Results will be calculated on the FLY

Scope

## Table

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

## Pane

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

## Cell

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

Direction

## Down

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

## Across

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

## Down then Across

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

## Across then Down

| Quarter .. | Month .. | Order Date 2020 | 2021 | 2022 |
|---|---|---|---|---|
| Q1 | Jan | 2.118 | 2.040 | 5.520 |
| | Feb | 3.962 | 6.713 | 9.729 |
| | Mar | 6.628 | 20.854 | 22.765 |
| Q2 | Apr | 8.867 | 23.219 | 28.864 |
| | May | 13.036 | 30.600 | 30.941 |
| | Jun | 16.190 | 32.553 | 33.996 |
| Q3 | Jul | 18.960 | 35.067 | 50.233 |
| | Aug | 22.642 | 37.932 | 57.142 |
| | Sept | 24.394 | 40.337 | 58.414 |
| Q4 | Oct | 26.961 | 43.248 | 60.893 |
| | Nov | 30.307 | 45.931 | 69.838 |
| | Dec | 31.167 | 52.359 | 74.770 |

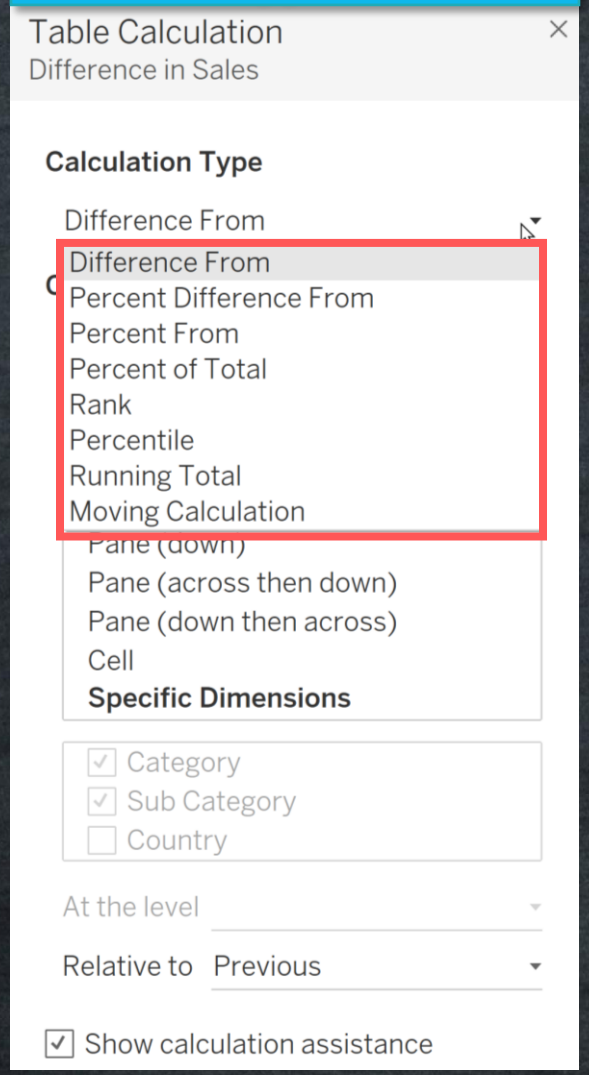
# Methods to Create
## Table Calculations

DATA WITH BARAA

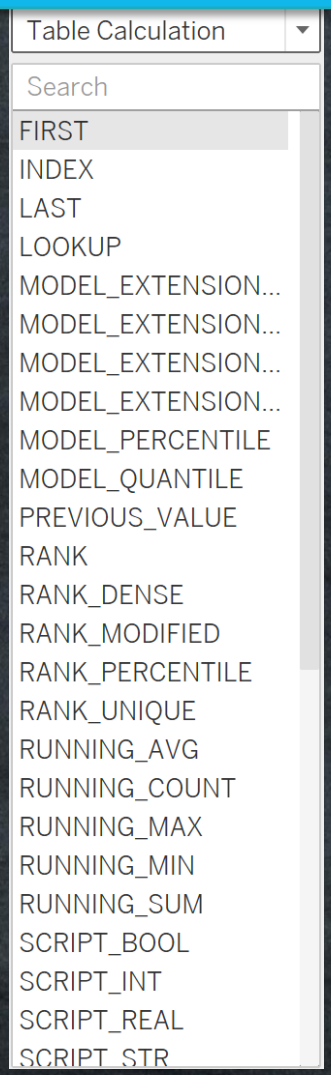
### Quick Table Calculations

- Running Total
- Difference
- Percent Difference
- Percent of Total
- Rank
- Percentile
- Moving Average
- YTD Total
- Compound Growth Rate
- Year Over Year Growth
- YTD Growth

### Table Calculation Types




### Table Calculation Functions



# Running Total

Add each value to the sum of all previous values

# Running Total

DATA WITH BARAA

Current Running Total = Sales Value

Current Row

| Month .. | Running S.. | Sales |
|---|---|---|
| Jan | | 2.067 |
| Feb | | |
| Mar | | |
| Apr | | |
| May | | |
| Jun | | |
| Jul | | |
| Aug | | |
| Sept | | |
| Oct | | |
| Nov | | |
| Dec | | |

Sales Value

# Running Total

Previous Running Value

| Month .. | Running S.. | Sales |
|---|---|---|
| Jan | 2.067 | 2.067 |
| Feb | | 523 |
| Mar | | |
| Apr | | |
| May | | |
| Jun | | |
| Jul | | |
| Aug | | |
| Sept | | |
| Oct | | |
| Nov | | |
| Dec | | |

Current Row

Sales Value

Current Running Total = Previous Running Total + Sales Value

2590 = 2067 + 523

# Running Total

DATA WITH BARAA

Current Running Total = Previous Running Total + Sales Value

Previous Running Value

| Month .. | Running S.. | Sales |
|---|---|---|
| Jan | 2.067 | 2.067 |
| Feb | 2.590 | 523 |
| Mar | | 6.422 |
| Apr | | |
| May | | |
| Jun | | |
| Jul | | |
| Aug | | |
| Sept | | |
| Oct | | |
| Nov | | |
| Dec | | |

Current Row

Sales Value

9013 = 2590 + 6422

# Running Total

DATA WITH BARAA

**Current Running Total = Previous Running Total + Sales Value**

| Month .. | Running S.. | Sales |
|---|---|---|
| Jan | 2.067 | 2.067 |
| Feb | 2.590 | 523 |
| Mar | 9.013 | 6.422 |
| Apr | 11.624 | 2.611 |
| May | 11.923 | 299 |
| Jun | 12.165 | 242 |
| Jul | 12.935 | 769 |
| Aug | 16.930 | 3.996 |
| Sept | | 107 |
| Oct | 17.703 | 666 |
| Nov | 23.247 | 5.544 |
| Dec | | 2.798 |

**Previous Running Value**

**Current Row**

**Sales Value**

**26045= 23247+ 2798**

# Difference

| Quarter .. | Month .. | 2020 |
|---|---|---|
| Q1 | Jan | 2.118 |
| | Feb | 1.844 |
| | Mar | 2.666 |
| Q2 | Apr | 2.240 |
| Current Row | May | 4.169 |
| | Jun | 3.154 |
| Q3 | Jul | 2.770 |
| | Aug | 3.682 |
| | Sept | 1.752 |
| Q4 | Oct | 2.567 |
| | Nov | 3.346 |
| | Dec | 860 |

Previous

Current

```
Difference = Current - Previous
1929 = 4169 - 2240
```

# Difference



| Quarter .. | Month .. | 2020 |
|---|---|---|
| Q1 | Jan | 2.118 |
| | Feb | 1.844 |
| | Mar | 2.666 |
| Q2 | Apr | 2.240 |
| Current Row | May | 4.169 |
| | Jun | 3.154 |
| Q3 | Jul | 2.770 |
| | Aug | 3.682 |
| | Sept | 1.752 |
| Q4 | Oct | 2.567 |
| | Nov | 3.346 |
| | Dec | 860 |

Current

Next

```
Difference = Current - Next
1015 = 4169 - 3154
```

# Difference

| Quarter .. | Month .. | 2020 |
|---|---|---|
| Q1 | Jan | 2.118 |
| | Feb | 1.844 |
| | Mar | 2.666 |
| Q2 | Apr | 2.240 |
| Current Row | May | 4.169 |
| | Jun | 3.154 |
| Q3 | Jul | 2.770 |
| | Aug | 3.682 |
| | Sept | 1.752 |
| Q4 | Oct | 2.567 |
| | Nov | 3.346 |
| | Dec | 860 |

First

Current

```
Difference = Current - First
2051 = 4169 - 2118
```

# Difference



| Quarter .. | Month .. | 2020 |
|---|---|---|
| Q1 | Jan | 2.118 |
| | Feb | 1.844 |
| | Mar | 2.666 |
| Q2 | Apr | 2.240 |
| Current Row | May | 4.169 |
| | Jun | 3.154 |
| Q3 | Jul | 2.770 |
| | Aug | 3.682 |
| | Sept | 1.752 |
| Q4 | Oct | 2.567 |
| | Nov | 3.346 |
| | Dec | 860 |

Current

Last

```
Difference = Current - Last
3309 = 4169 - 860
```

# Difference


| Quarter .. | Month .. | 2020 |
|---|---|---|
| Q1 | Jan | 2.118 |
| | Feb | 1.844 |
| | Mar | 2.666 |
| Q2 | Apr | 2.240 |
| Current Row | May | 4.169 |
| | Jun | 3.154 |
| Q3 | Jul | 2.770 |
| | Aug | 3.682 |
| | Sept | 1.752 |
| Q4 | Oct | 2.567 |
| | Nov | 3.346 |
| | Dec | 860 |

First

Previous

Current

Next

Last