DATA WITH BARAA

AGGREGATE

FUNCTIONS

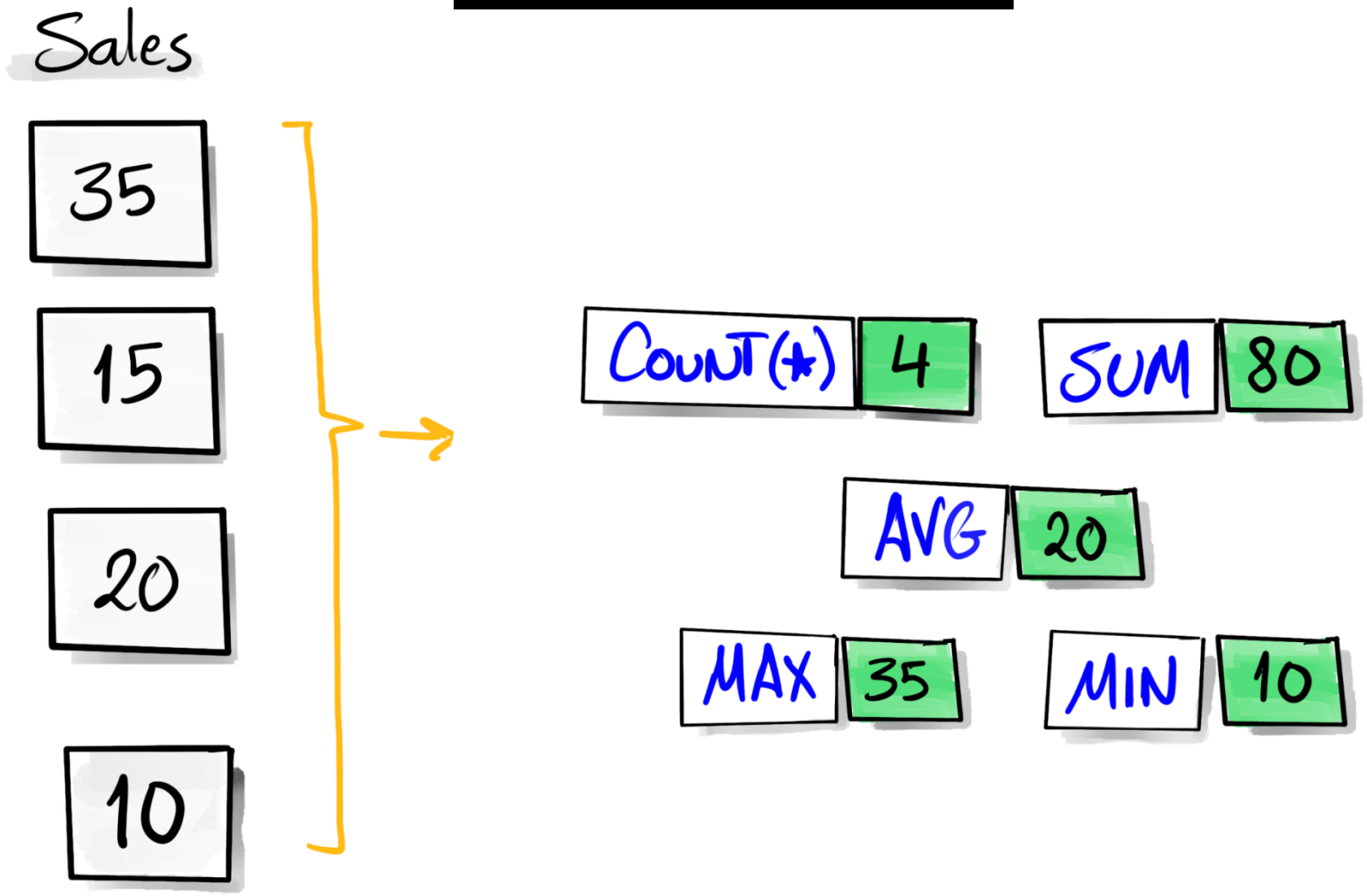
Baraa Khatib Salkini
YouTube | **DATA WITH BARAA**
SQL Course | Aggregate Functions

Aggregate Functions

DATA WITH BARAA

Data Types		
Any Types	COUNT	Counts the number of rows
Only Numbers	SUM	Add up all values in a column
	AVG	Find the average of values
Any Types	MAX	Gets the highest value
	MIN	Gets the lowest value

Aggregate Functions



DATA WITH BARAA

WINDOW FUNCTIONS

BASICS

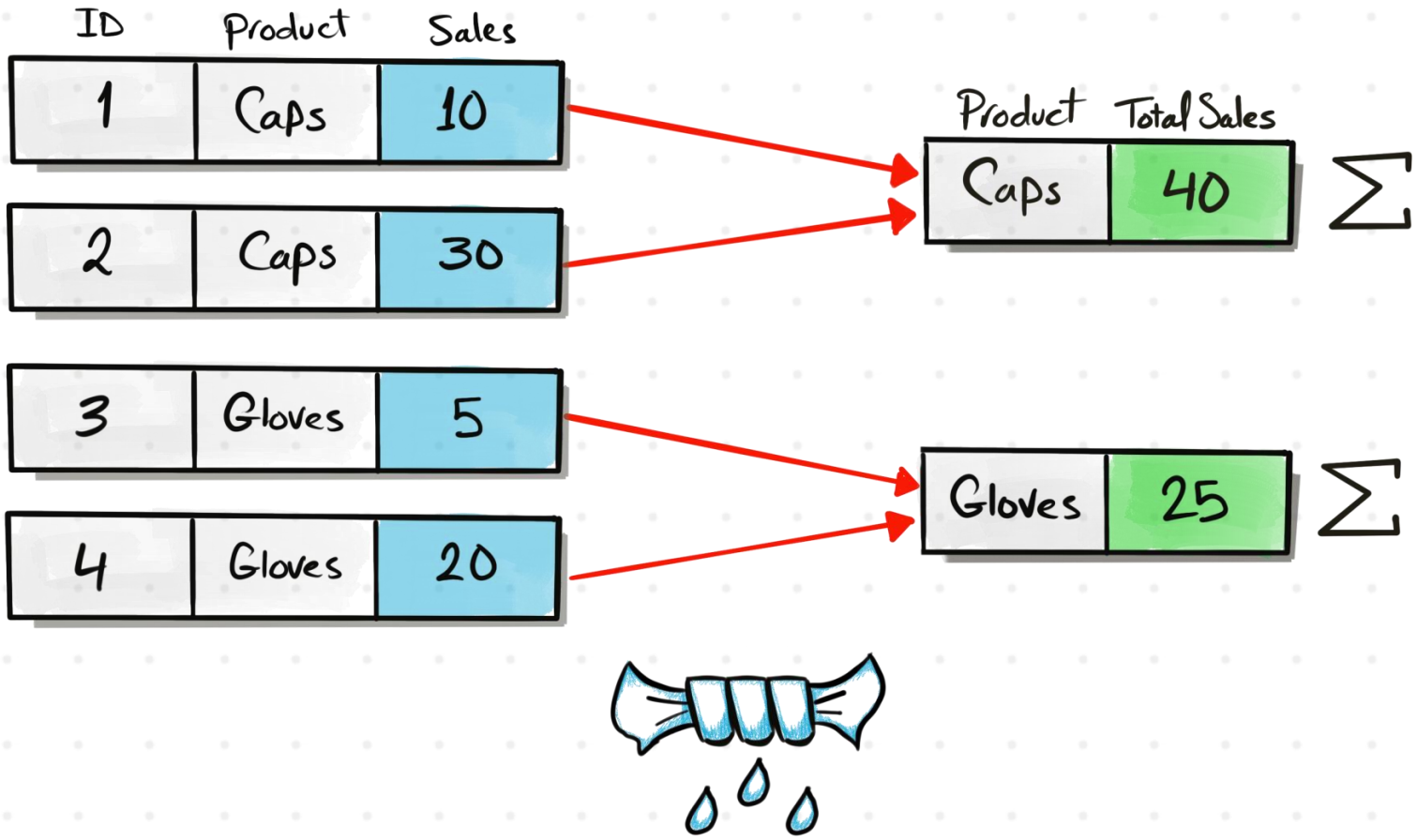
Baraa Khatib Salkini
YouTube | **DATA WITH BARAA**
SQL Course | Window Functions

WINODW FUNCTIONS

Perform calculations (e.g. aggregation)
on a specific subset of data,
without losing the level of details of rows.

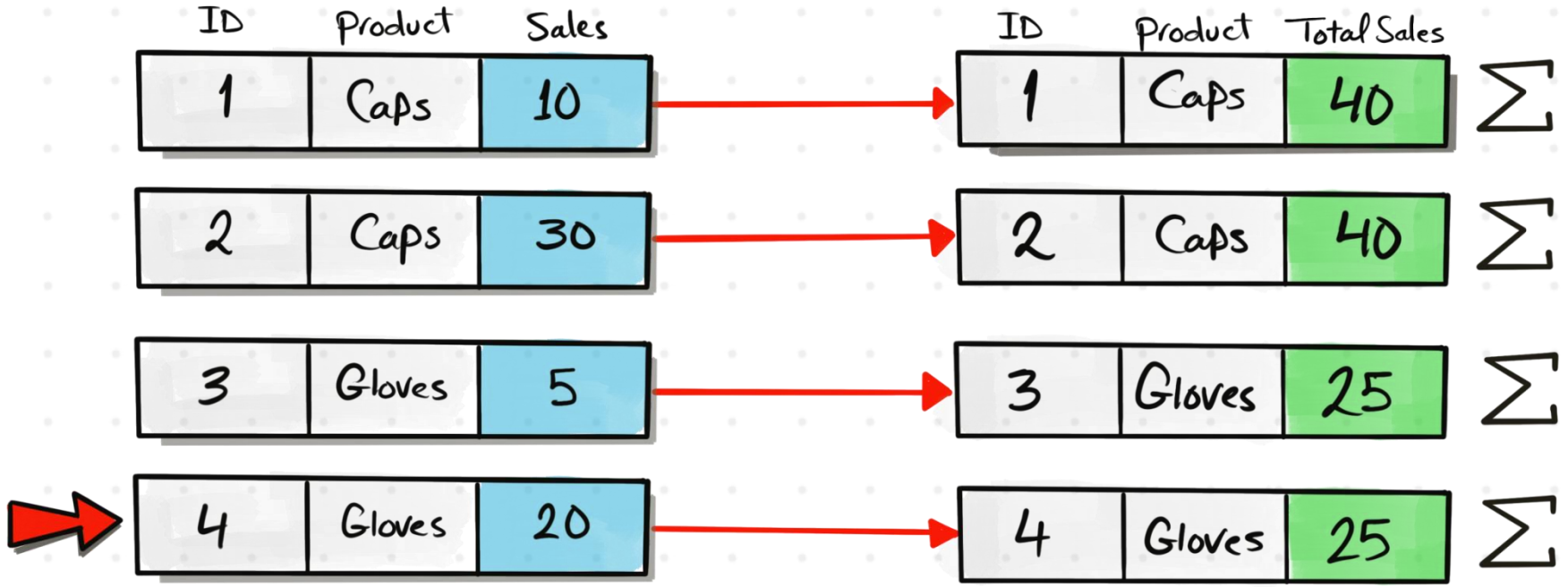
GROUP BY

Aggregates and groups rows based on column/s into summary rows



WINDOW Functions

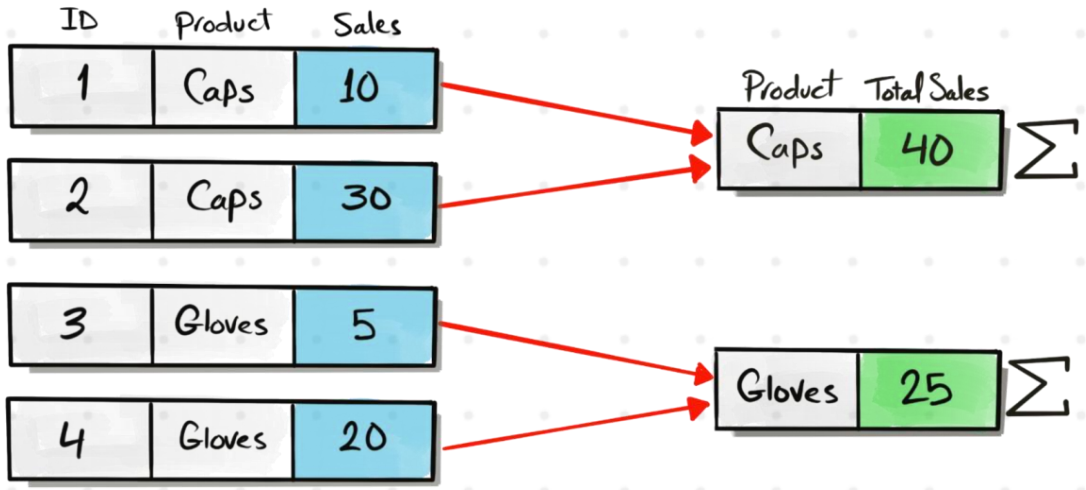
Compute aggregates but keep details of individual rows at the same time



Row-Level Calculation

DATA WITH BARAA

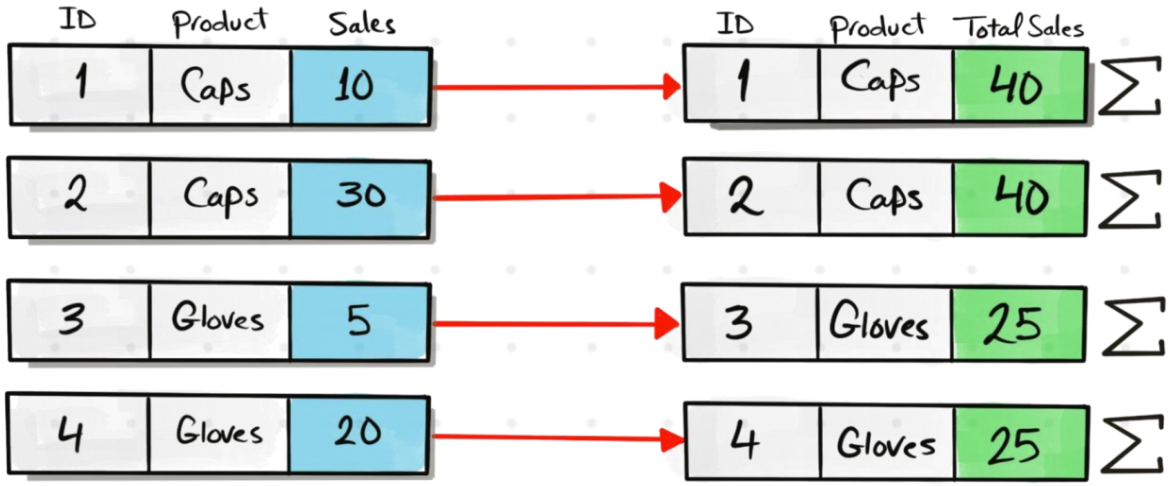
GROUP BY



It collapse groups of rows into a single row

(Group-Level-Calculations)

Window Functions



It doesn't collapse rows into a single row

(Row-Level-Calculations)

GROUP BY Functions

Aggregate Functions

- `COUNT(expr)`
- `SUM(expr)`
- `MAX(expr)`
- `MIN(expr)`
- `MIN(expr)`

WINDOW Functions

Aggregate Functions

- `COUNT(expr)`
- `SUM(expr)`
- `MAX(expr)`
- `MIN(expr)`
- `MIN(expr)`

Rank Functions

- `ROW_NUMBER()`
- `RANK()`
- `DENSE_RANK()`
- `CUME_DIST()`
- `PERCENT_RANK()`
- `NTILE(n)`

Value (Analytics) Functions

- `LEAD(expr,offset,default)`
- `LAG(expr,offset,default)`
- `FIRST_VALUE(expr)`
- ~~`FIRST`~~`_VALUE(expr)`
  LAST

DATA WITH BARAA

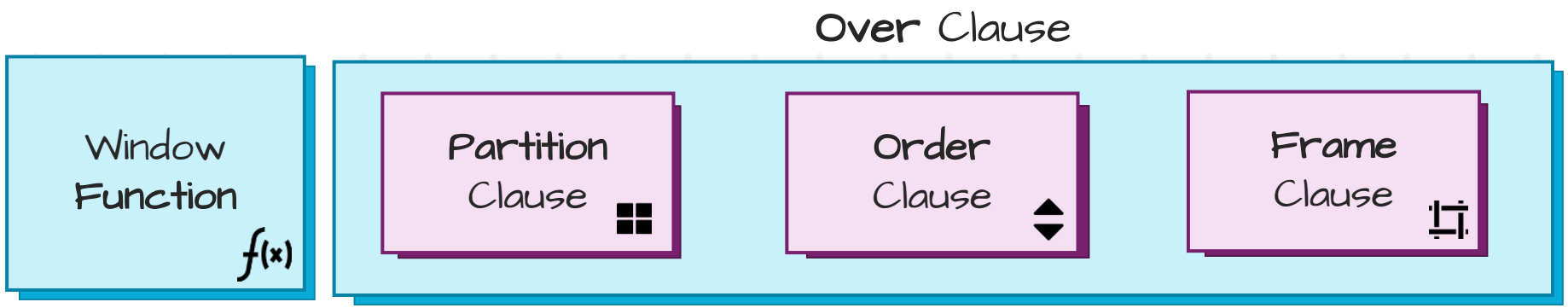
GROUP BY

Simple Data Analysis
(Aggregations)

WINDOW

Advanced Data Analysis
(Aggregations + Details)

Window Syntax



Window
Syntax

```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

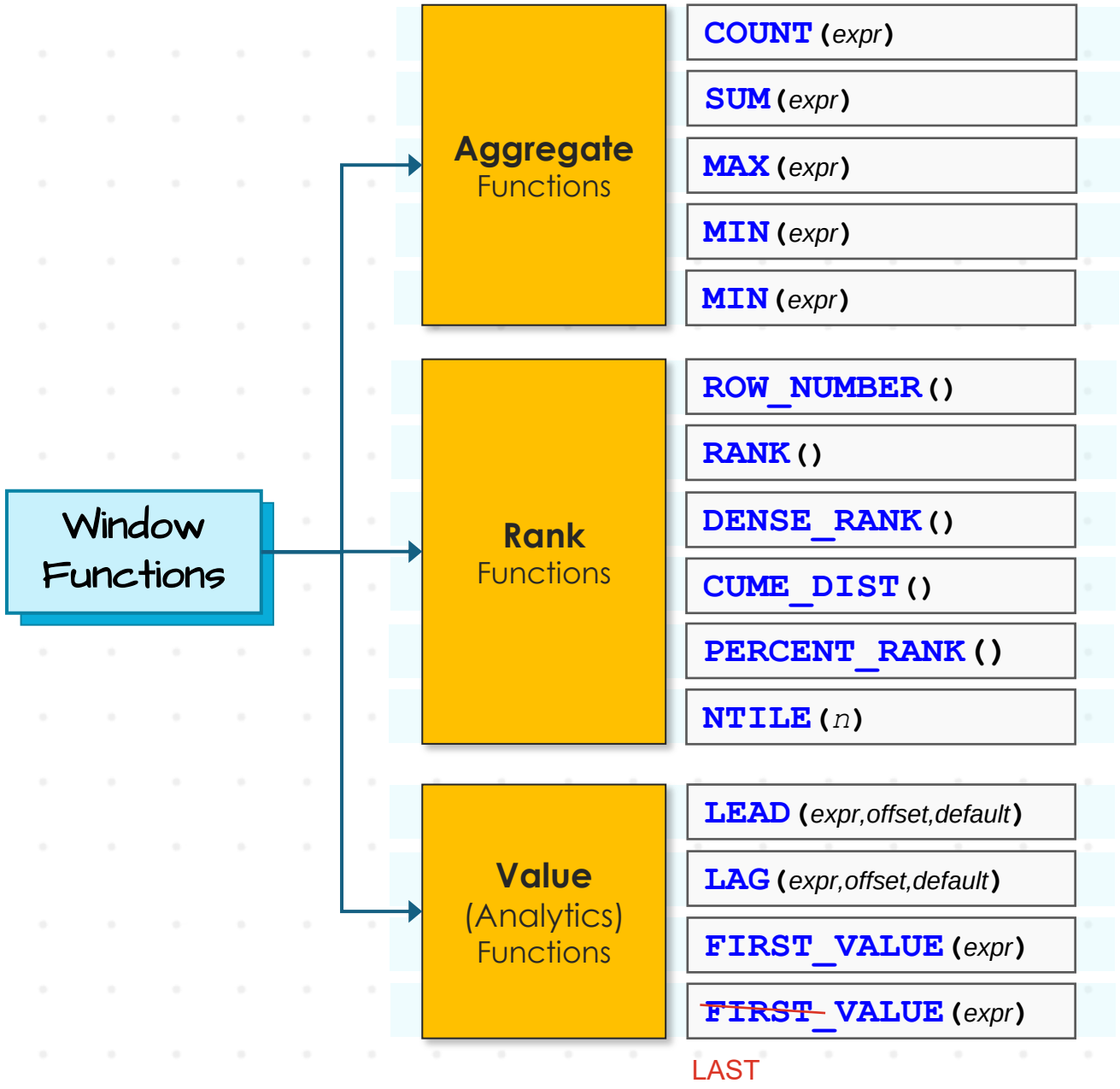
Window Syntax

Calculation used on the Window

f(x) Window Function

```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```


WINODW FUNCTIONS
Perform calculations within a window


Window Functions
Aggregate Functions
COUNT(expr)
SUM(expr)
MAX(expr)
MIN(expr)
MIN(expr)
Rank Functions
ROW_NUMBER()
RANK()
DENSE_RANK()
CUME_DIST()
PERCENT_RANK()
NTILE(n)
Value (Analytics) Functions
LEAD(expr,offset,default)
LAG(expr,offset,default)
FIRST_VALUE(expr)
~~FIRST~~_VALUE(expr)
LAST

Window Syntax

Calculation used on the Window

f(x) Window Function

```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

Function Expression

FUNCTION EXPRESSION

Arguments you pass to a function

Window Expressions

Empty

```
RANK() OVER (ORDER BY OrderDate)
```

Column

```
AVG(Sales) OVER (ORDER BY OrderDate)
```

Number

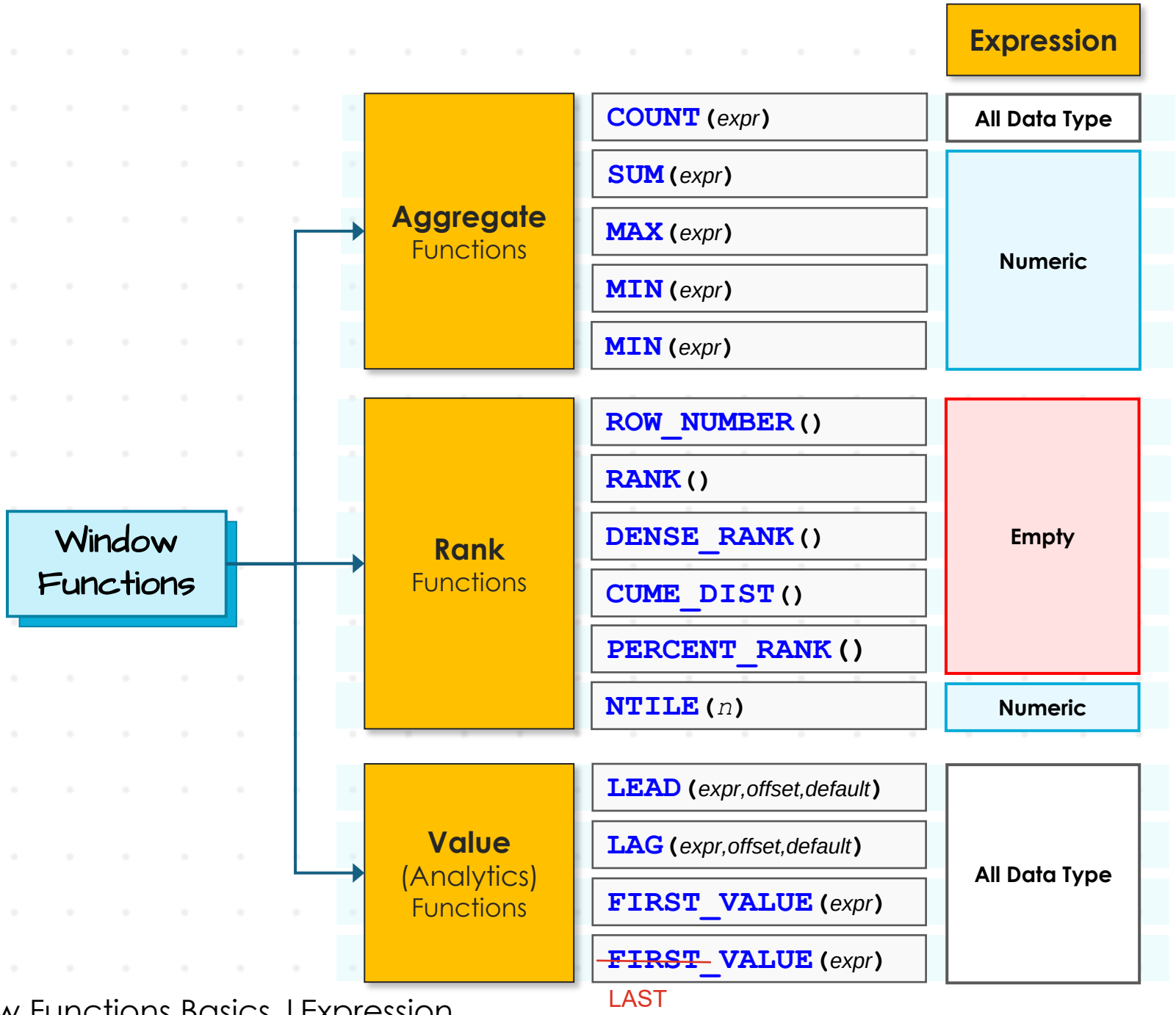
```
NTEIL(2) OVER (ORDER BY OrderDate)
```

Multiple Arguments

```
LEAD(Sales,2,10) OVER (ORDER BY OrderDate)
```

Conditional Logic

```
SUM(CASE WHEN Sales > 100 THEN 1 ELSE 0 END) OVER (ORDER BY OrderDate)
```


Expression
Window Functions
Aggregate Functions
COUNT (expr)
SUM (expr)
MAX (expr)
MIN (expr)
MIN (expr)
All Data Type
Numeric
Rank Functions
ROW_NUMBER ()
RANK ()
DENSE_RANK ()
CUME_DIST ()
PERCENT_RANK ()
NTILE (n)
Empty
Numeric
Value (Analytics) Functions
LEAD (expr,offset,default)
LAG (expr,offset,default)
FIRST_VALUE (expr)
FIRST_VALUE (expr)
LAST
All Data Type

Window Syntax

Calculation used on the Window

f(x) Window Function

Define the Window

Over Clause

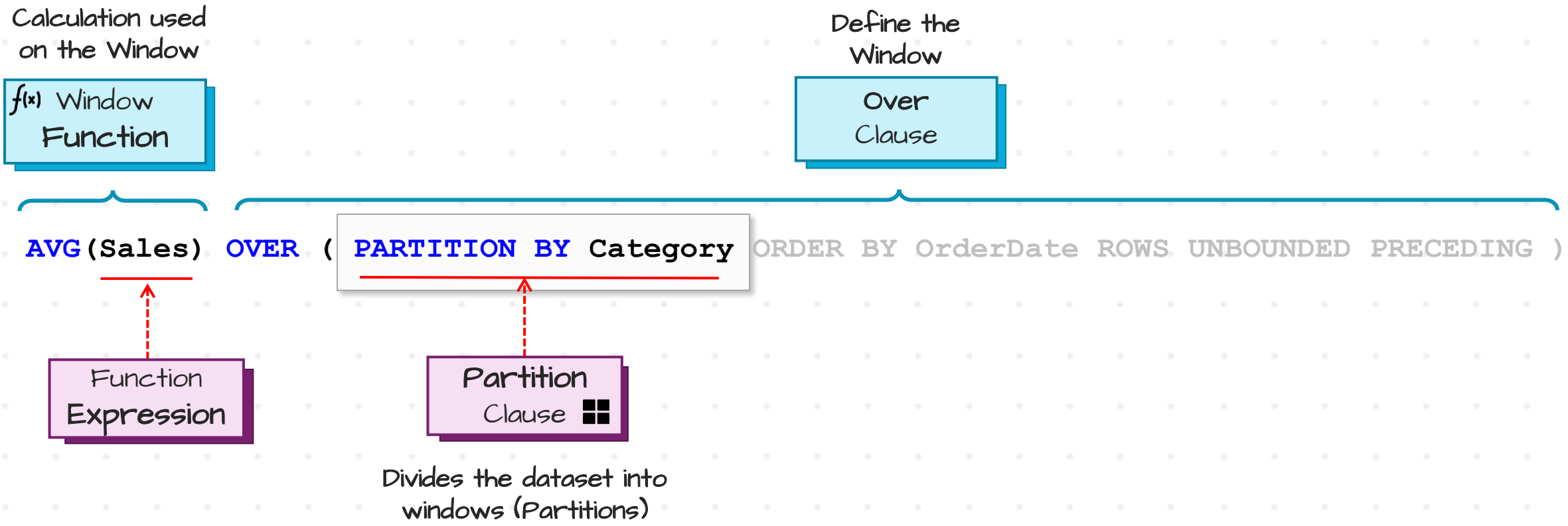
```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

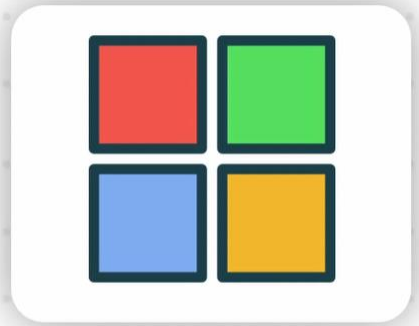
Function Expression

OVER CLAUSE

Tells SQL that the function used is a window function

Window Syntax





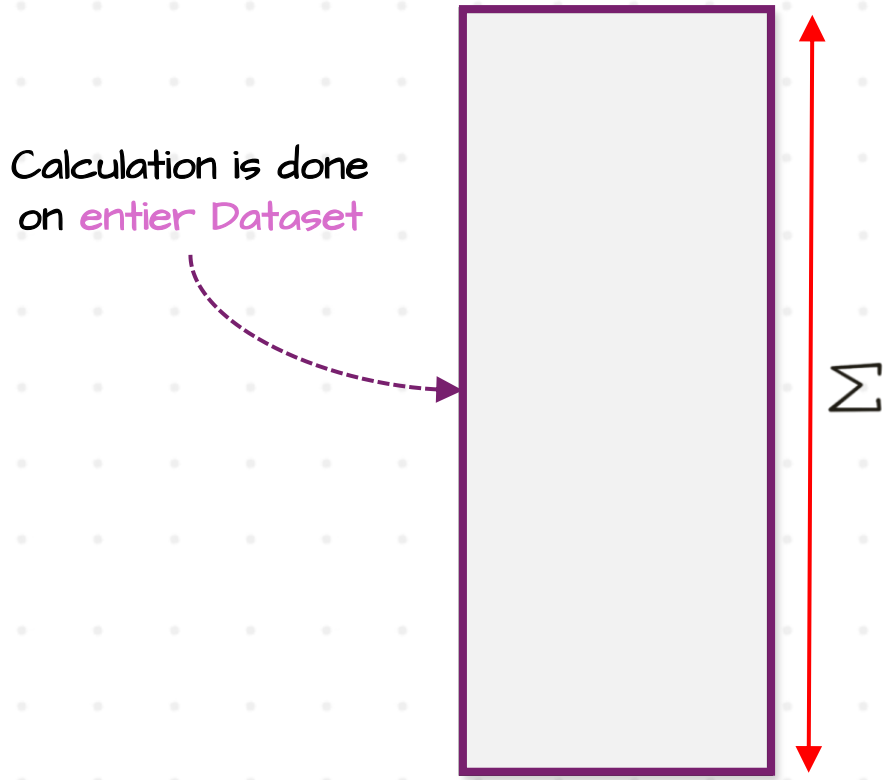
PARTITION BY

Divides the result set into partitions (Windows)

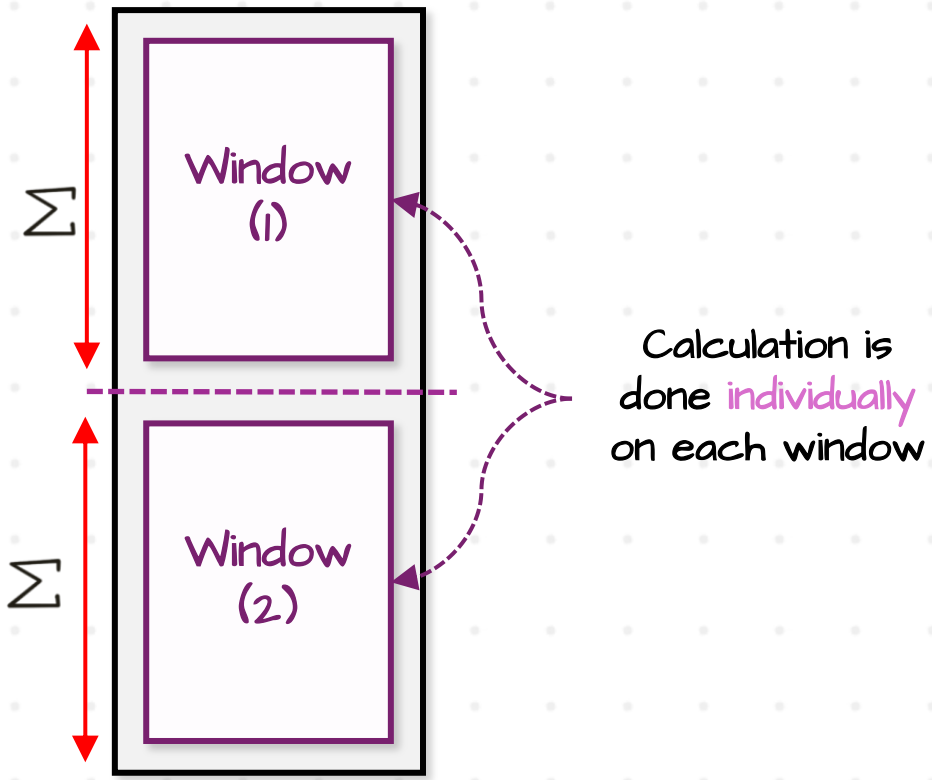
Partition By

PARTITION BY divides the rows into groups, based on the column/s

```
SUM(Sales) OVER()
```




```
SUM(Sales) OVER(PARTITION BY Product)
```



Partition By

`PARTITION BY` divides the rows into groups, based on the column/s

Without Partition By

Total sales across all rows (Entire Result Set)

```
SUM(Sales) OVER ()
```

Partition By Single Column

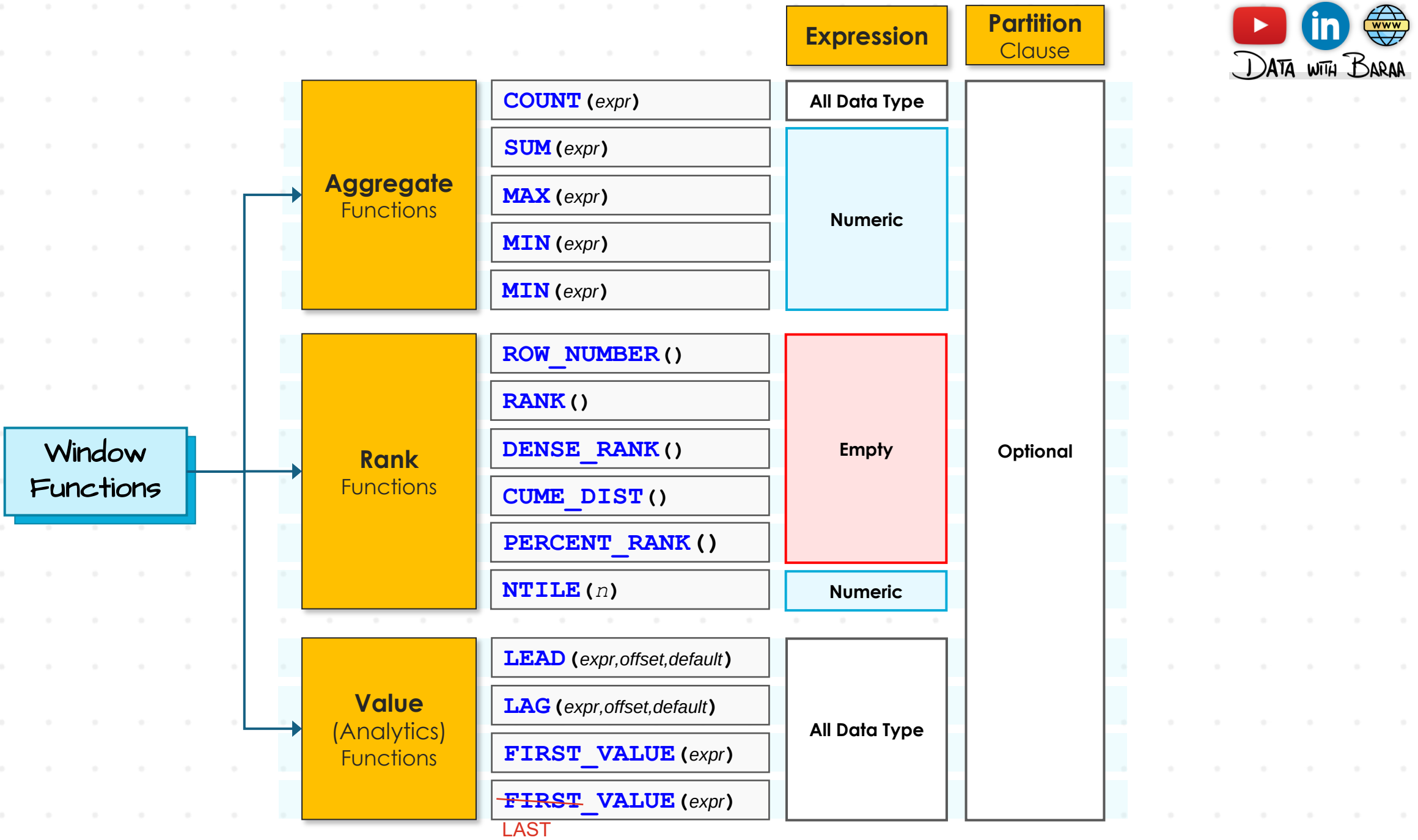
Total sales for each Product

```
SUM(Sales) OVER (PARTITION BY Product)
```

Partition By Combined-Columns

Total sales for each combination of Product and Order Status

```
SUM(Sales) OVER (PARTITION BY Product, OrderStatus)
```


Expression
Partition Clause
Window Functions
Aggregate Functions
COUNT (expr)
SUM (expr)
MAX (expr)
MIN (expr)
MIN (expr)
All Data Type
Numeric
Rank Functions
ROW_NUMBER ()
RANK ()
DENSE_RANK ()
CUME_DIST ()
PERCENT_RANK ()
NTILE (n)
Empty
Numeric
Value (Analytics) Functions
LEAD (expr,offset,default)
LAG (expr,offset,default)
FIRST_VALUE (expr)
FIRST_VALUE (expr)
LAST
All Data Type
Optional

Window Syntax

Calculation used on the Window

Window Function

Define the Window

Over Clause

```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

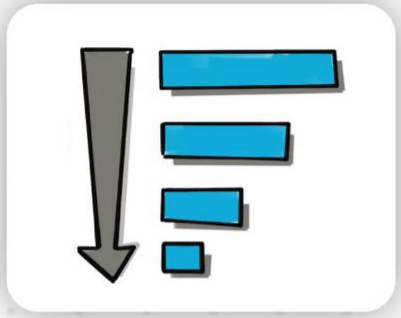
Function Expression

Partition Clause

Divides the dataset into windows (Partitions)

Order Clause

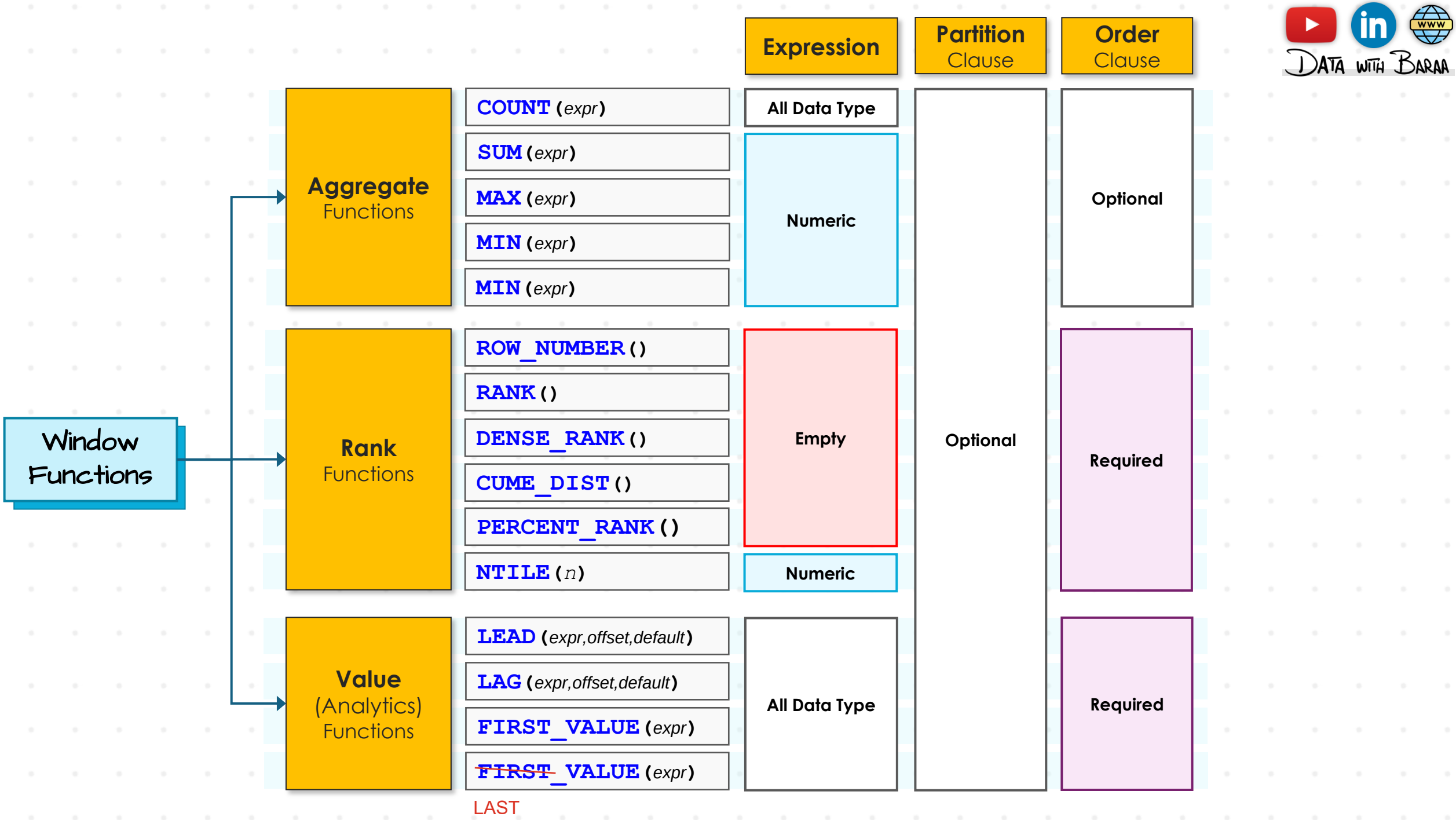
Sort the data in a window



ORDER BY

Sort the data within a window

(Ascending | Descending)


Expression
Partition Clause
Order Clause
Window Functions
Aggregate Functions
COUNT (expr)
SUM (expr)
MAX (expr)
MIN (expr)
MIN (expr)
All Data Type
Numeric
Optional
Rank Functions
ROW_NUMBER ()
RANK ()
DENSE_RANK ()
CUME_DIST ()
PERCENT_RANK ()
NTILE (n)
Empty
Numeric
Optional
Required
Value (Analytics) Functions
LEAD (expr,offset,default)
LAG (expr,offset,default)
FIRST_VALUE (expr)
FIRST_VALUE (expr)
LAST
All Data Type
Required
DATA WITH BARAA

Window Syntax

Calculation used on the Window

Window Function

Define the Window

Over Clause

```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

Function Expression

Partition Clause

Divides the dataset into windows (Partitions)

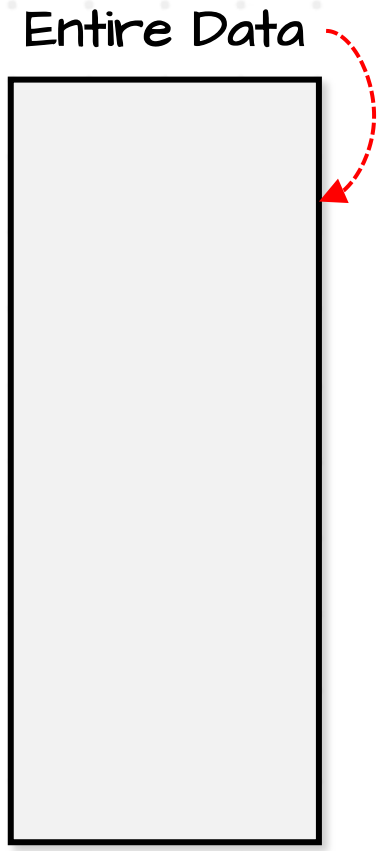
Order Clause

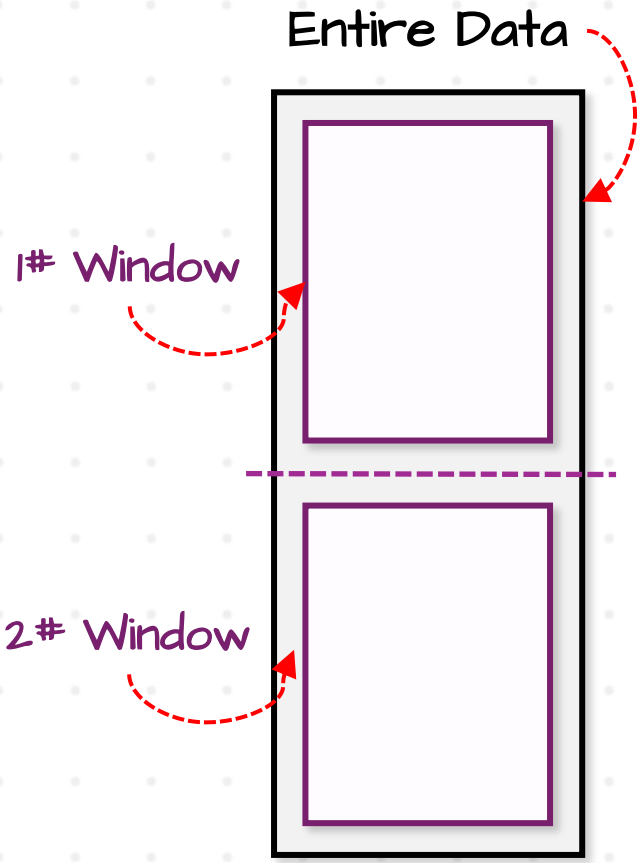
Sort the data in a window

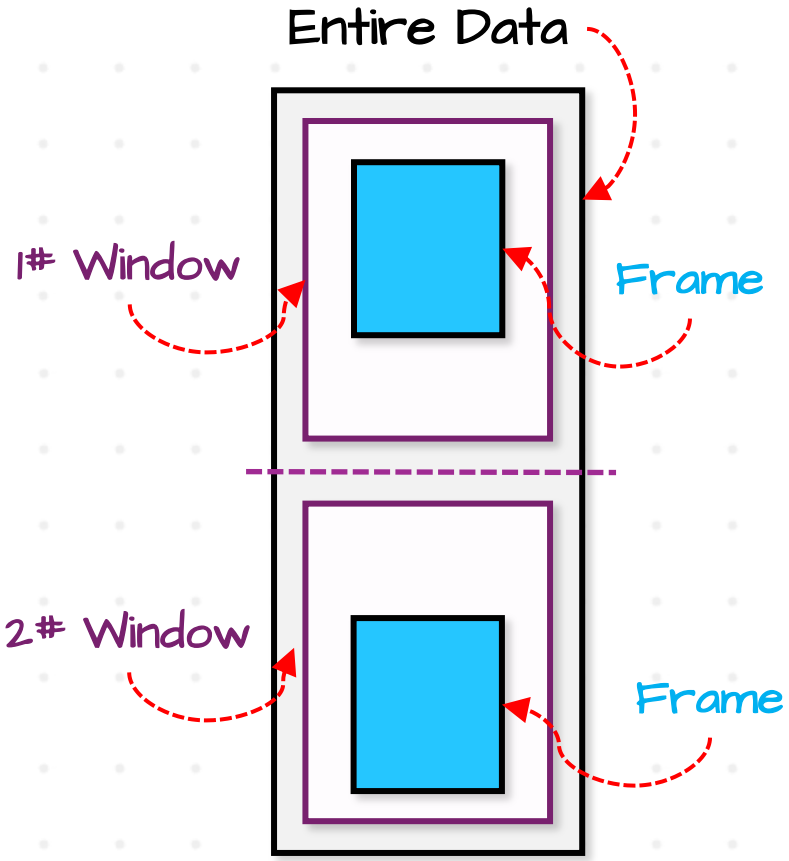
Frame Clause

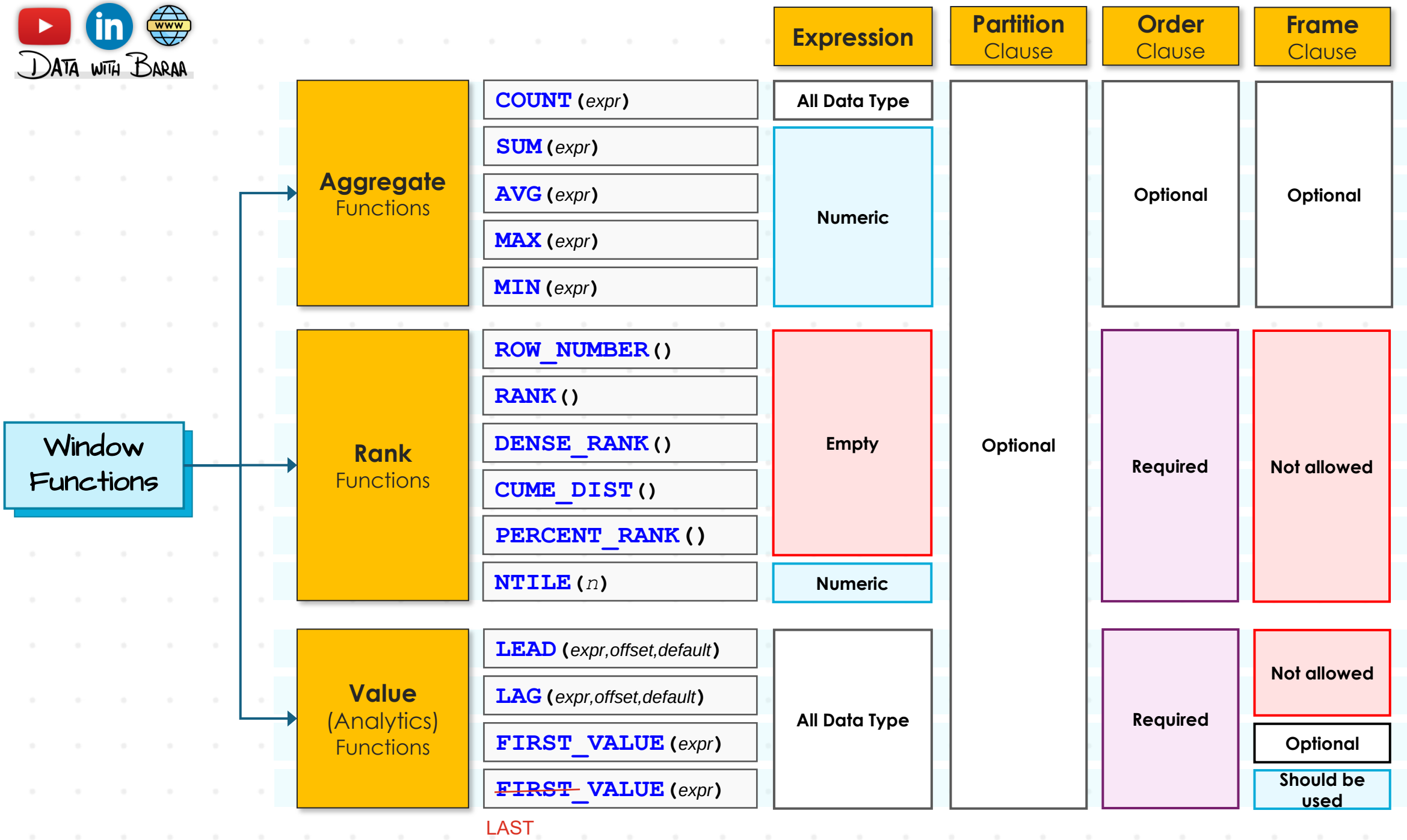
Define a subset of rows in a window

Frame Clause



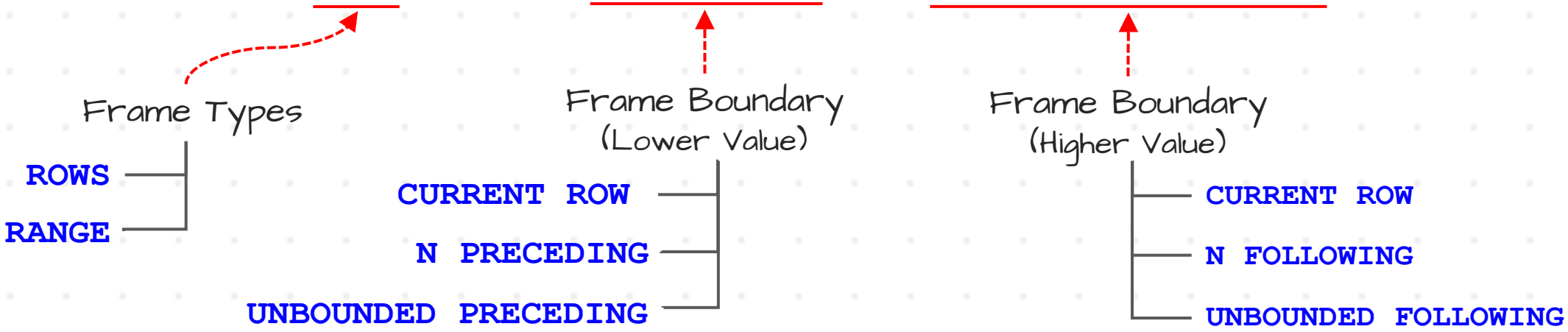





DATA WITH BARAA
Expression
Partition Clause
Order Clause
Frame Clause
Window Functions
Aggregate Functions
COUNT (expr)
SUM (expr)
AVG (expr)
MAX (expr)
MIN (expr)
All Data Type
Numeric
Optional
Optional
Rank Functions
ROW_NUMBER ()
RANK ()
DENSE_RANK ()
CUME_DIST ()
PERCENT_RANK ()
NTILE (n)
Empty
Numeric
Optional
Required
Not allowed
Value (Analytics) Functions
LEAD (expr,offset,default)
LAG (expr,offset,default)
FIRST_VALUE (expr)
FIRST_VALUE (expr)
LAST
All Data Type
Required
Not allowed
Optional
Should be used

Frame

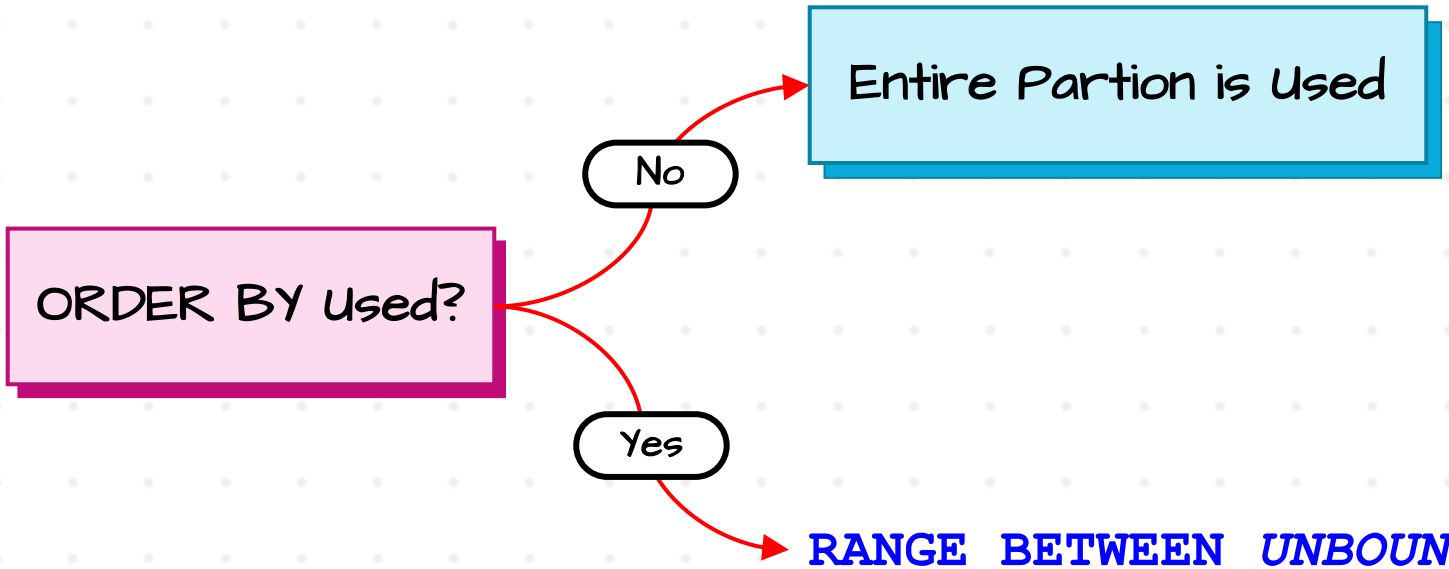
```
AVG(Sales) OVER (PARTITION BY Category ORDER BY OrderDate
    ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING)
```



Rules

- Frame Clause can only be used together with order by clause.
- Lower Value must be BEFORE the higher Value.

Frame


ORDER BY Used?
No
Entire Partion is Used
Yes
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
(Default FRAME with ORDER BY)

Frame

DATA WITH BARAA

COMPACT FRAME

For only PRECEDING, the CURRENT ROW can be skipped

2 PRECEDING AND

NORMAL FORM `ROWS BETWEEN CURRENT ROW ~~AND 2 FOLLOWING~~`

SHORT FORM `ROWS 2 ~~FOLLOWING~~` PRECEDING

Window Rules

DATA WITH BARAA

#1 RULE

Window functions can be used ONLY

in SELECT and ORDER BY Clauses

#2 RULE

Nesting Window Functions is not allowed !

#3 RULE

SQL execute WINDOW Functions after WHERE Clause

#4 RULE

Window Function can be used together with GROUP BY

in the same query, ONLY if the same columns are used

Window Rules

Inner Window Function

```
AVG( SUM(Sales) OVER () ) OVER (ORDER BY OrderDate )
```

Outer Window Function

Not allowed to nest window functions !

Windowed functions cannot be used in the context of another windowed function or aggregate.

SQL WINDOW FUNCTIONS

Performs Calculations on Subset of data without losing details

Window v/s GroupBy

- Window is more Powerfull & Dynamic than GroupBy.
- Data Analysis → Advanced → Window
  Data Analysis → Simple → Group By
- Use GroupBy + window in Same Query, only if Same Column used.

Components

Window Functions + Window Definition OVER

- Divide Data — PARTITION BY
- Sort Data — ORDER BY
- Define Subset — FRAME

Rules

- Nesting is not allowed!
- Window Can be used only in SELECT and ORDER BY
- SQL executes window AFTER filtering data using WHERE

f(x) Window Functions

Aggregate

- SUM()
- AVG()
- COUNT()
- MAX()
- MIN()

Perform calculations on a set of rows and return a single aggregated value for each row

Rank

- ROW_NUMBER()
- RANK()
- DENSE_RANK()
- NTILE()
- CUME_DIST()
- PERCENT_RANK()

Assign a rank to each row in a window

Value

- LAG()
- LEAD()
- FIRST_VALUE()
- LAST_VALUE()

Return a specific value in a window to be compared with the value of current row

DATA WITH BARAA

WINDOW AGGREGATE FUNCTIONS

Baraa Khatib Salkini
YouTube | **DATA WITH BARAA**
SQL Course | Window Aggregate Functions

f(x) Window Functions

Aggregate

- SUM()
- AVG()
- COUNT()
- MAX()
- MIN()

Perform calculations on a set of rows and return a single aggregated value for each row

Rank

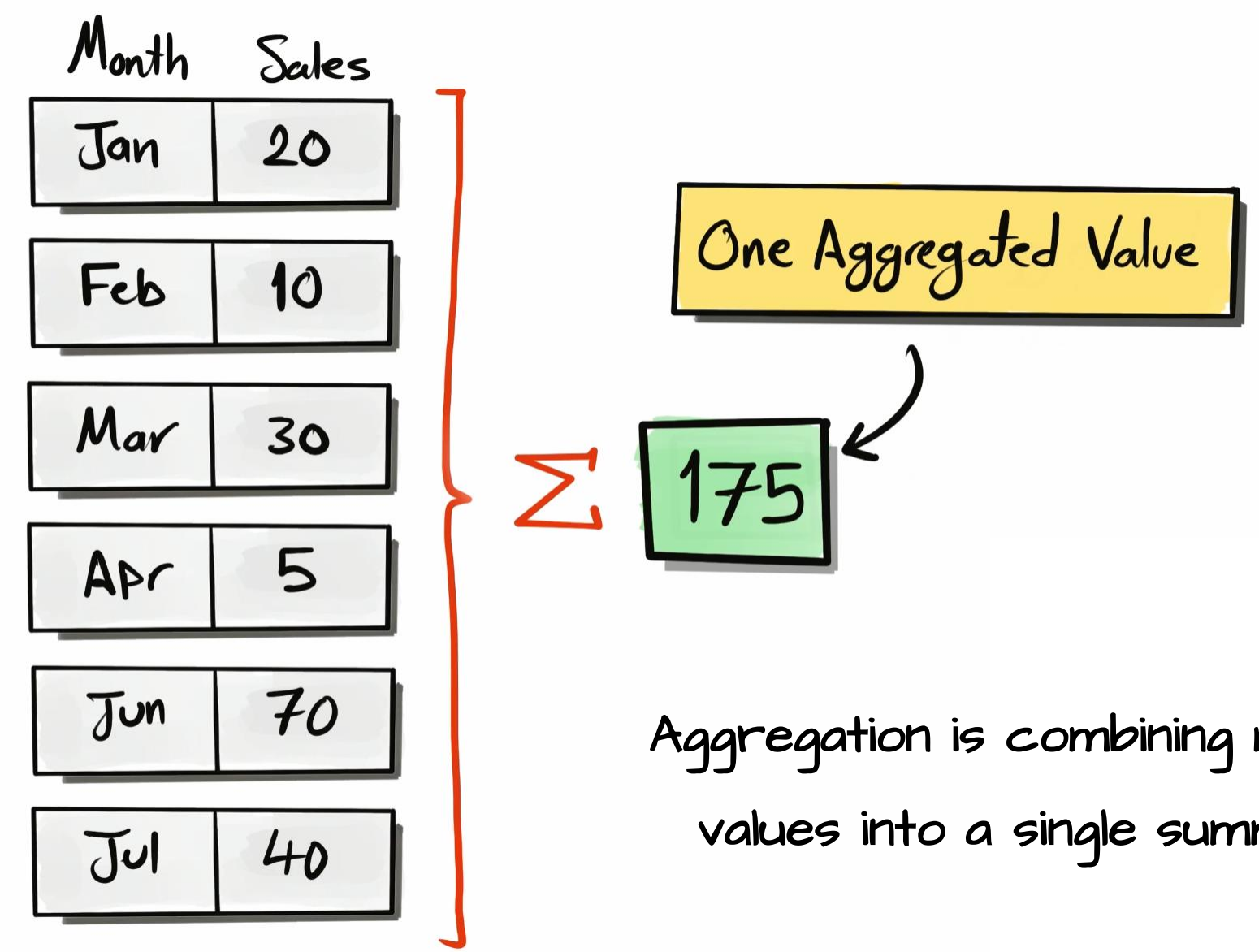
- ROW_NUMBER()
- RANK()
- DENSE_RANK()
- NTILE()
- CUME_DIST()
- PERCENT_RANK()

Assign a rank to each row in a window

Value

- LAG()
- LEAD()
- FIRST_VALUE()
- LAST_VALUE()

Return a specific value in a window to be compared with the value of current row



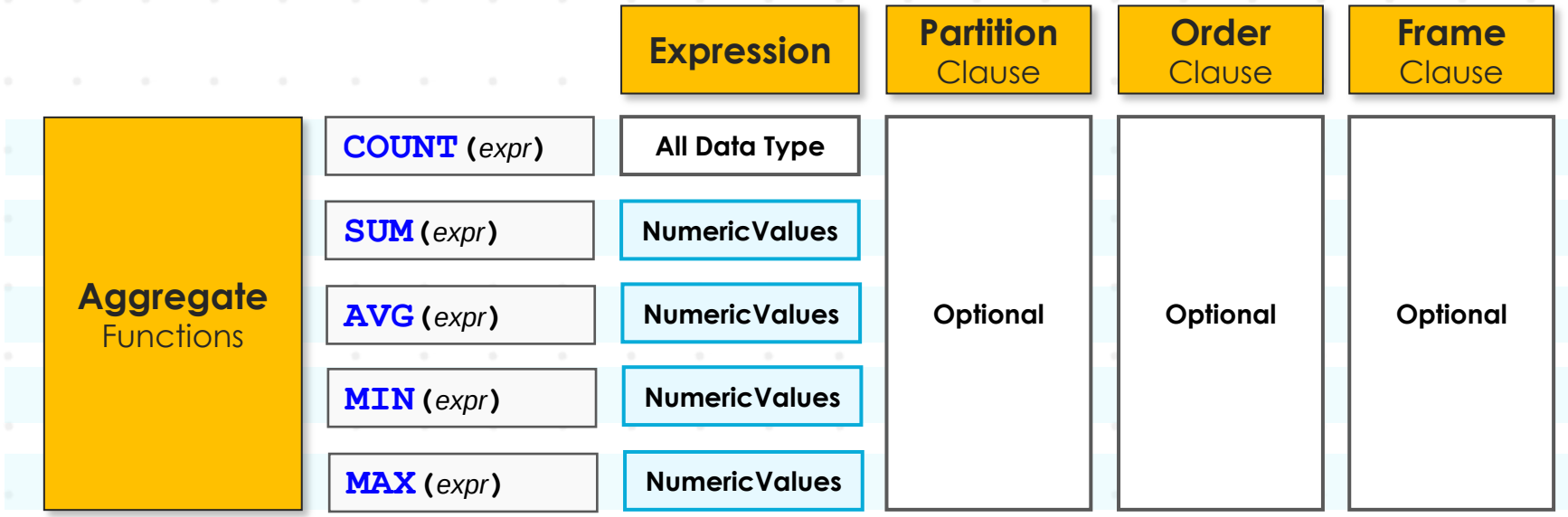
Aggregation is combining multiple values into a single summary

Aggregate Functions


```
AVG(Sales) OVER (PARTITION BY ProductID ORDER BY Sales)
```

- **Sales** — Expression is required (Only Numeric Values)
- **PARTITION BY ProductID** — Partition By Is Optional
- **ORDER BY Sales** — Order By Is Optional

Aggregate Functions



Aggregate Functions

DATA WITH BARAA

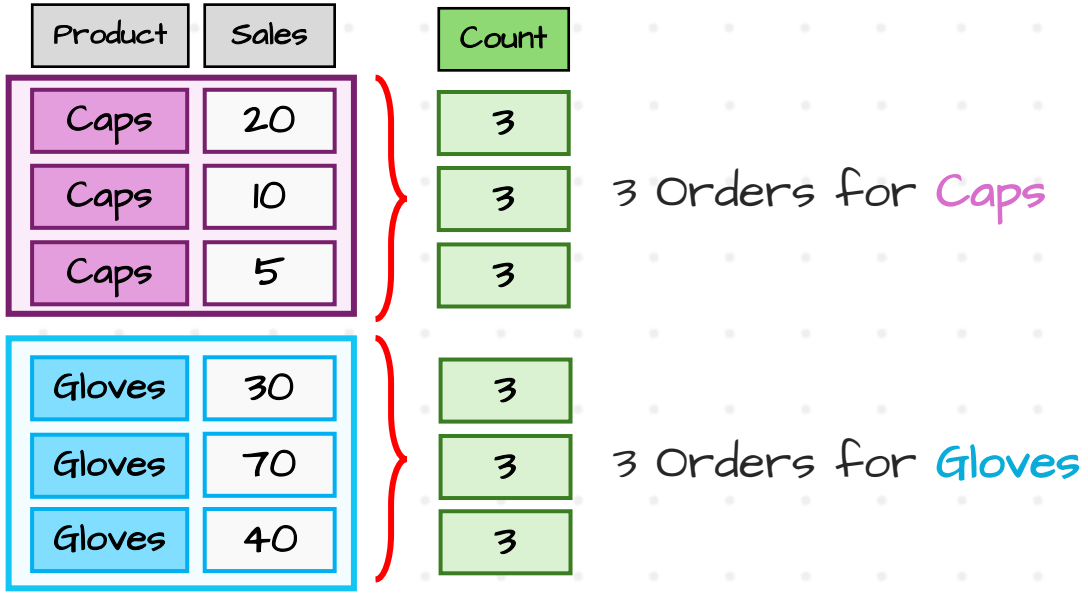
Aggregate Functions			
	COUNT (*expr*)	Returns the number of Rows in a window	`COUNT(*) OVER (PARTITION BY Porduct)`
	SUM (*expr*)	Returns the sum of values in a window	`SUM(Sales) OVER (PARTITION BY Porduct)`
	AVG (*expr*)	Returns the average of values in a window	`SUM(Sales) OVER (PARTITION BY Porduct)`
	MIN (*expr*)	Returns the minimum value in a window	`SUM(Sales) OVER (PARTITION BY Porduct)`
	MAX (*expr*)	Returns the maximum value in a window	`SUM(Sales) OVER (PARTITION BY Porduct)`

COUNT Function

Data with Baraa

Returns the number of Rows in a window

```
COUNT(*) OVER(PARTITION BY Product)
```



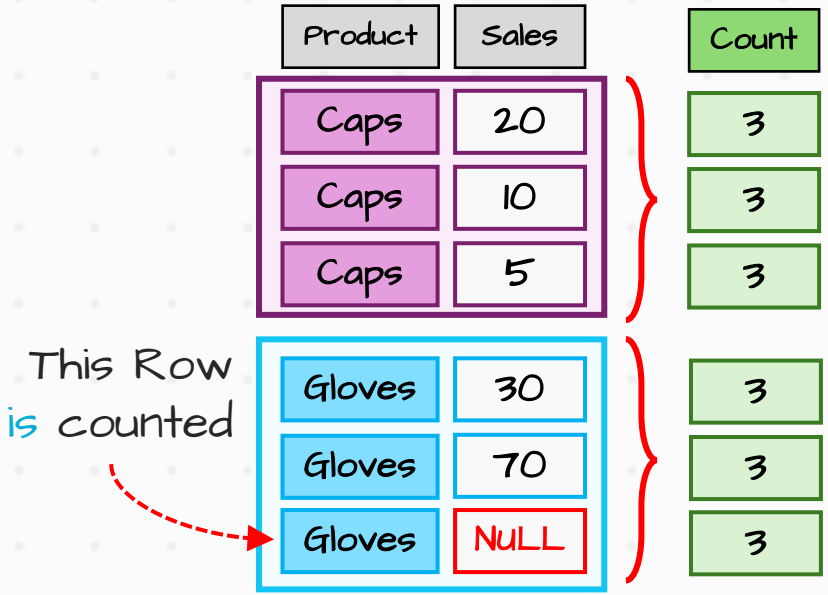
COUNT Function

DATA WITH BARAA

Count the number of Rows including NULLs

```
COUNT(*) OVER(PARTITION BY Product)
```

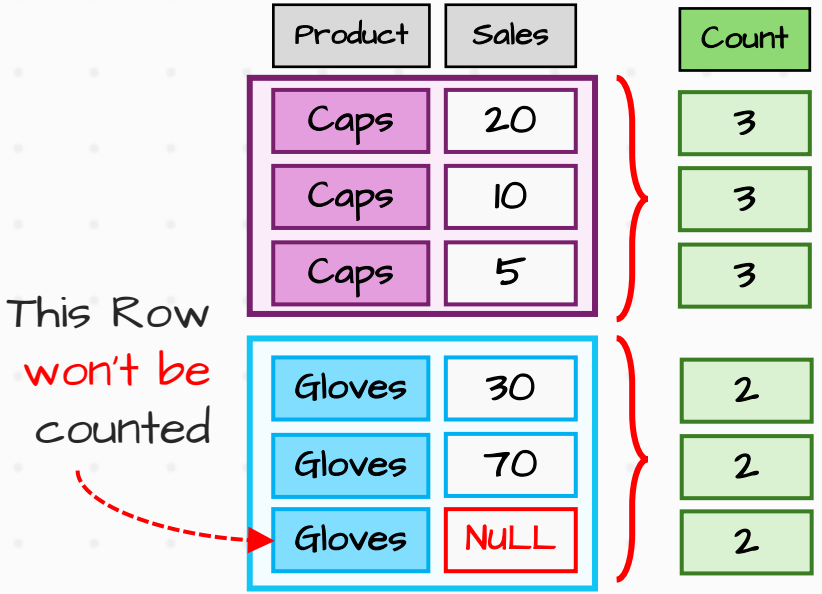
```
COUNT(1) OVER(PARTITION BY Product)
```



Count the number of Rows excluding NULLs

```
COUNT(Sales) OVER(PARTITION BY Product)
```

Column



COUNT Function

```
COUNT(1) OVER(PARTITION BY Product)
COUNT(*) OVER(PARTITION BY Product)
```

Product	Sales	COUNT
Caps	20	3
Caps	10	3
Caps	5	3
Gloves	30	3
Gloves	70	3
Gloves	NULL	3

```
COUNT(Sales) OVER(PARTITION BY Product)
```

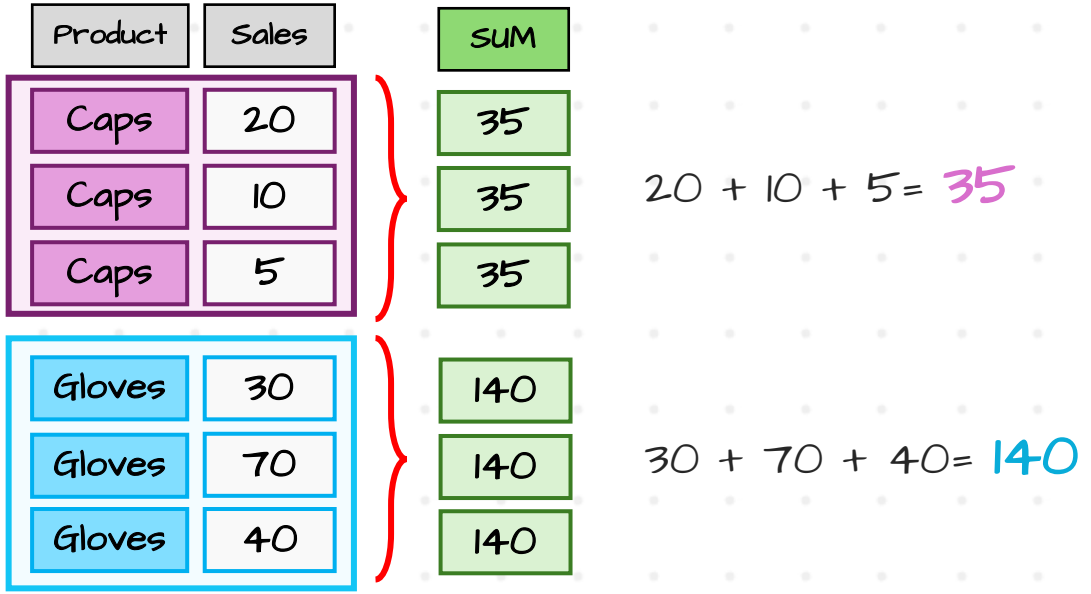
Product	Sales	COUNT
Caps	20	3
Caps	10	3
Caps	5	3
Gloves	30	2
Gloves	70	2
Gloves	NULL	2

SUM Function

Returns the sum of values in a window

```
SUM(Sales) OVER(PARTITION BY Product)
```

* Is not allowed!



SUM Function

```
SUM(Sales) OVER(PARTITION BY Product)
```

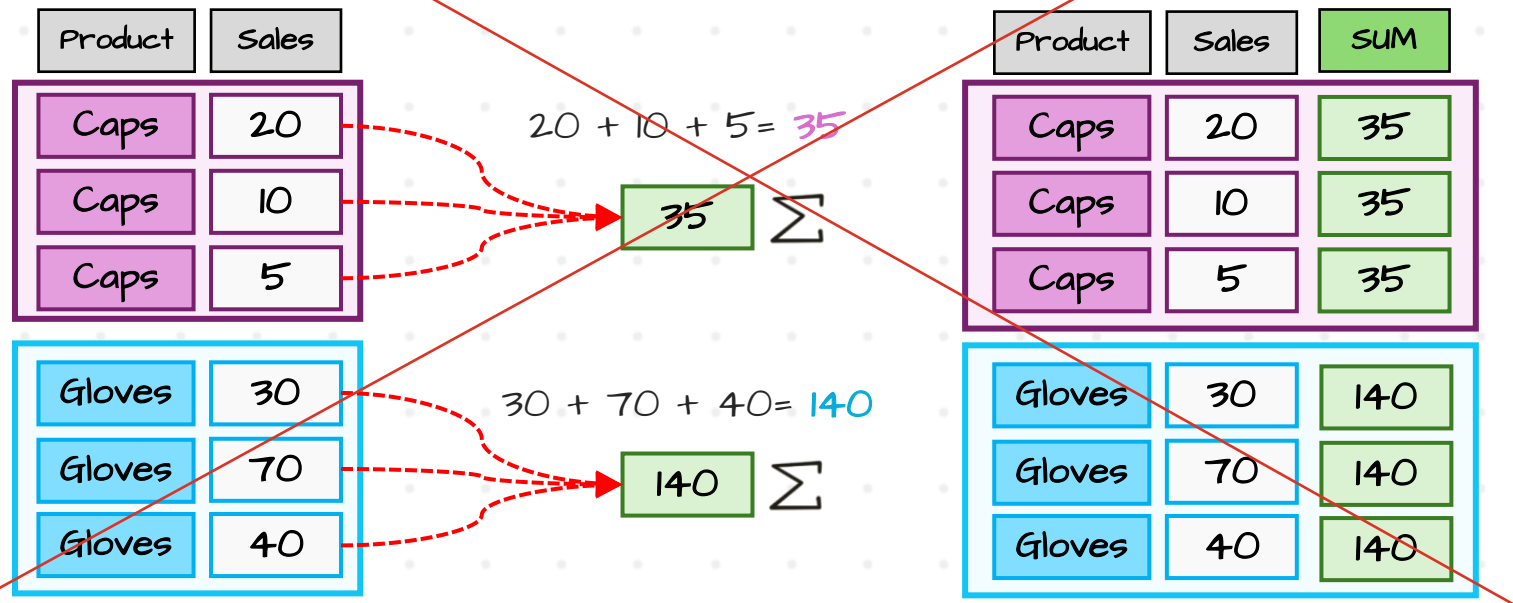
Product	Sales	SUM
Caps	20	35
Caps	10	35
Caps	5	35
Gloves	30	100
Gloves	70	100
Gloves	NULL	100

20 + 10 + 5 = 35

30 + 70 = 100

COUNT Function

Perform calculations on a set of rows and return a single aggregated value for each row

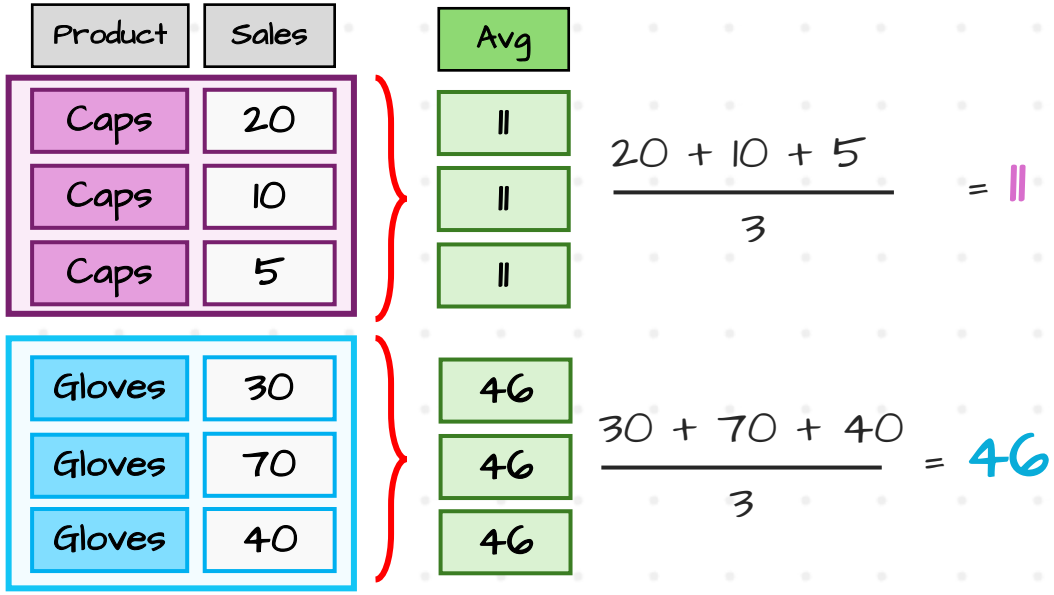


AVG Function

Returns the average of values in a window

```
AVG(Sales) OVER(PARTITION BY Product)
```

* Is not allowed!



AVG Function

DATA WITH BARAA

Default Average Function exclude NULLs

```
AVG(Sales) OVER(PARTITION BY Product)
```

Product	Sales	AVG
Caps	20	11
Caps	10	11
Caps	5	11
Gloves	30	50
Gloves	70	50
Gloves	NULL	50

$$\frac{20 + 10 + 5}{3} = 11$$

$$\frac{30 + 70}{2} = 50$$

Deal with Nulls using COALESCE to include NULLs

```
AVG(COALESCE(Sales,0)) OVER(PARTITION BY Product)
```

Replace NULL with 0

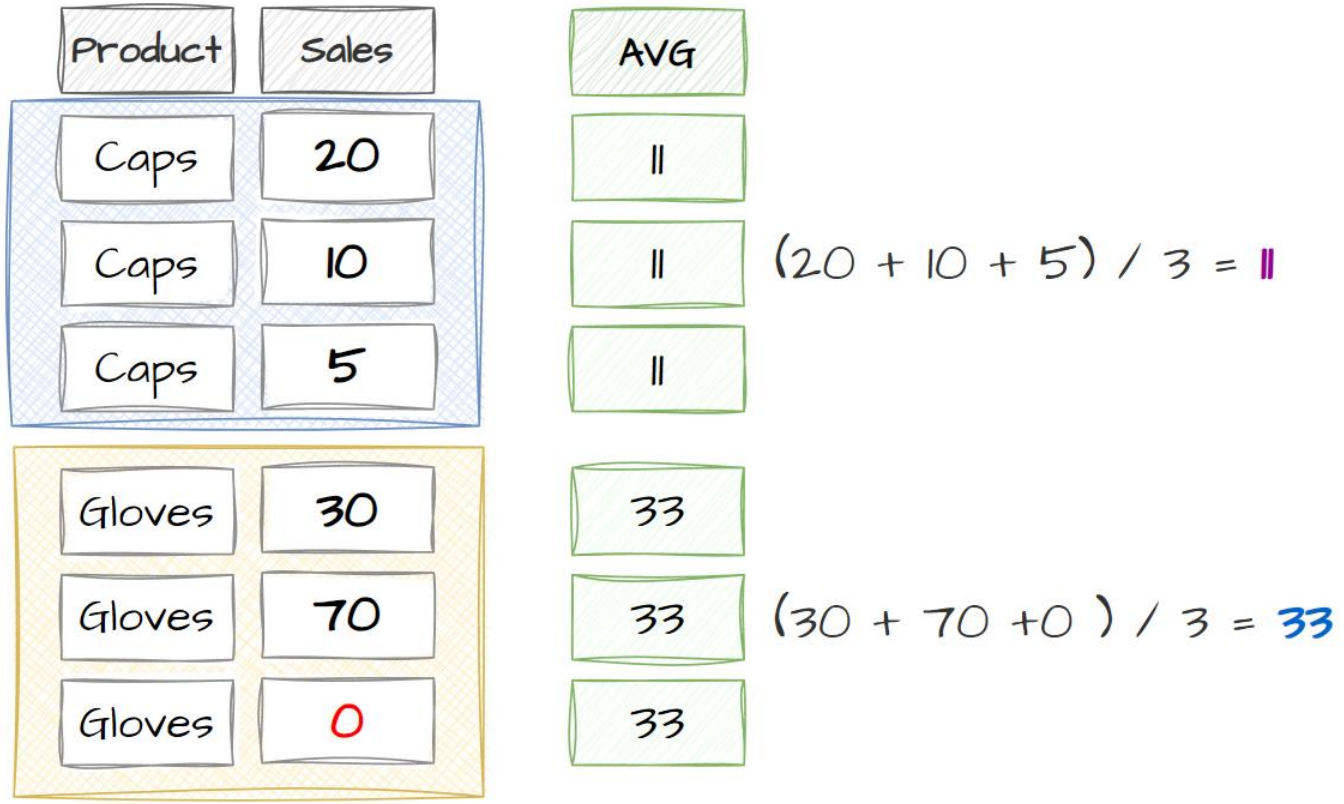
Product	Sales	AVG
Caps	20	11
Caps	10	11
Caps	5	11
Gloves	30	50
Gloves	70	50
Gloves	0	50

$$\frac{20 + 10 + 5}{3} = 11$$

$$\frac{30 + 70 + 0}{3} = 33$$

AVG Function

```
AVG(COALESCE(Sales,0)) OVER(PARTITION BY Product)
```

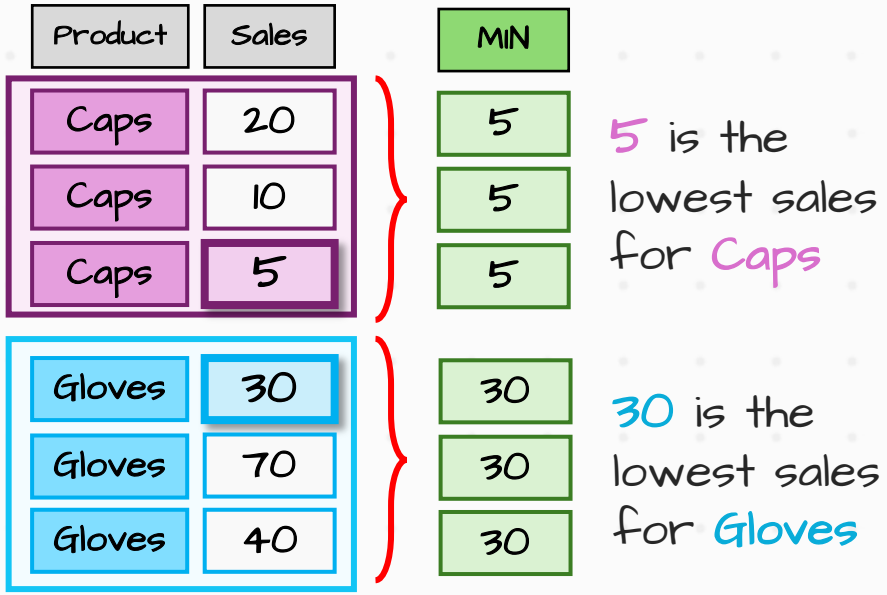



```
AVG(Sales) OVER(PARTITION BY Product)
```

MIN

Returns the minimum value in a window

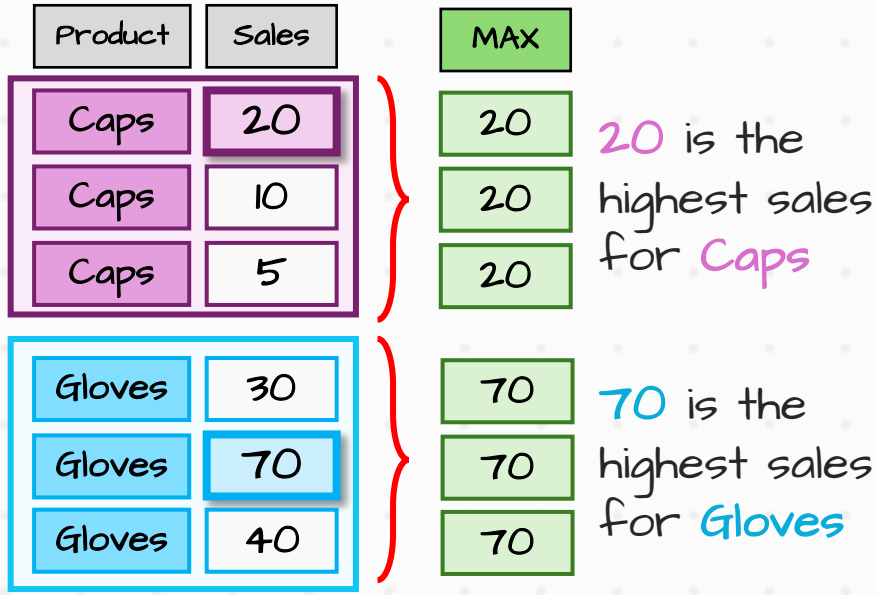
```
MIN(Sales) OVER(PARTITION BY Product)
```



MAX

Returns the maximum value in a window

```
MAX(Sales) OVER(PARTITION BY Product)
```



MAX & MIN Function

DATA WITH BARAA

Find the highest sales for each product

```
MIN(Sales) OVER(PARTITION BY Product)
```

Product	Sales	MIN
Caps	20	5
Caps	10	5
Caps	5	5
Gloves	30	0
Gloves	70	0
Gloves	0	0

Find the lowest sales for each product

```
MAX(Sales) OVER(PARTITION BY Product)
```

Product	Sales	MAX
Caps	20	20
Caps	10	20
Caps	5	20
Gloves	30	70
Gloves	70	70
Gloves	0	70

RUNNING TOTAL

Aggregate all values from the beginning up to the current point without dropping off older data.

ROLLING TOTAL

Aggregate all values within a fixed time window (e.g. 30 days).
As new data is added, the oldest data point will be dropped.

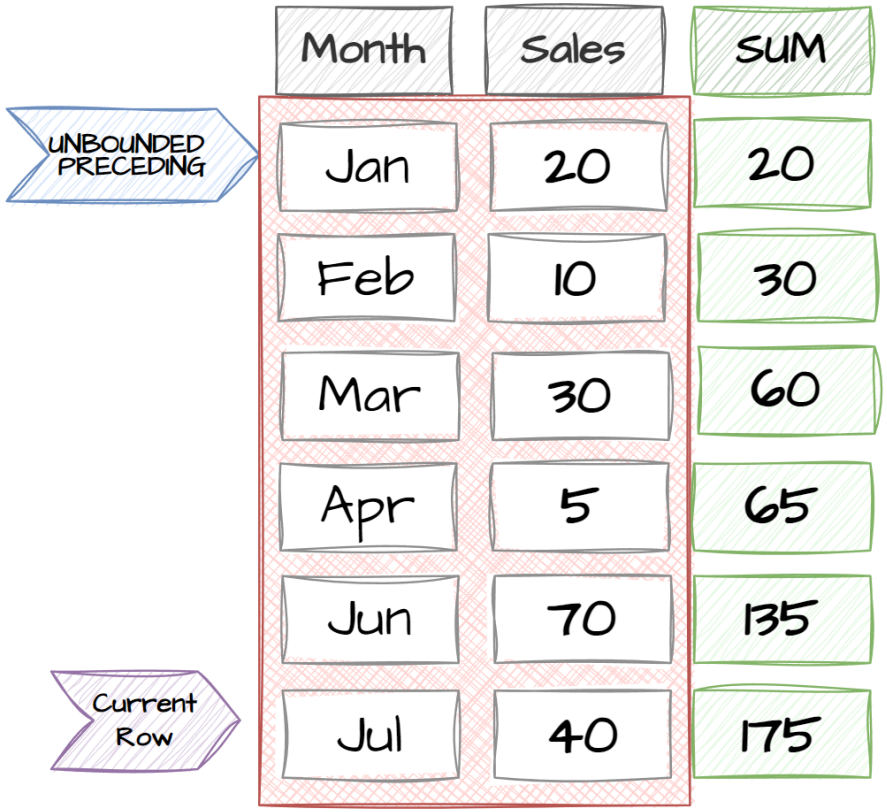
Rolling/Shifting Window

Running Total

```
SUM(Sales) OVER( ORDER BY Month)
```

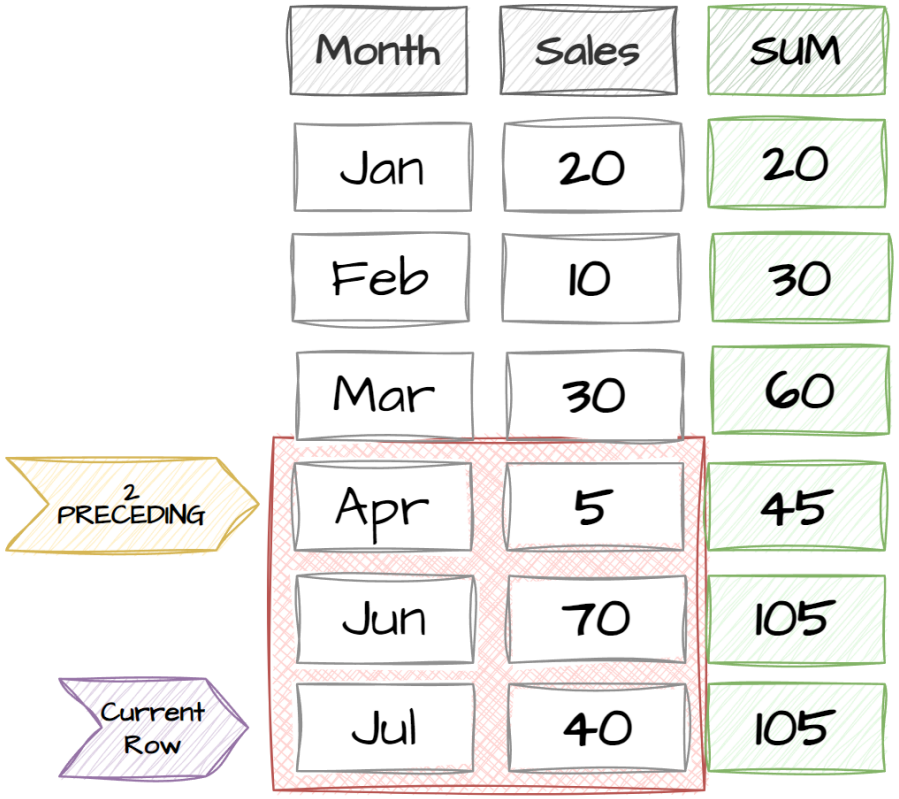
Default

```
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

```
SUM(Sales) OVER( ORDER BY MONTH
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
```



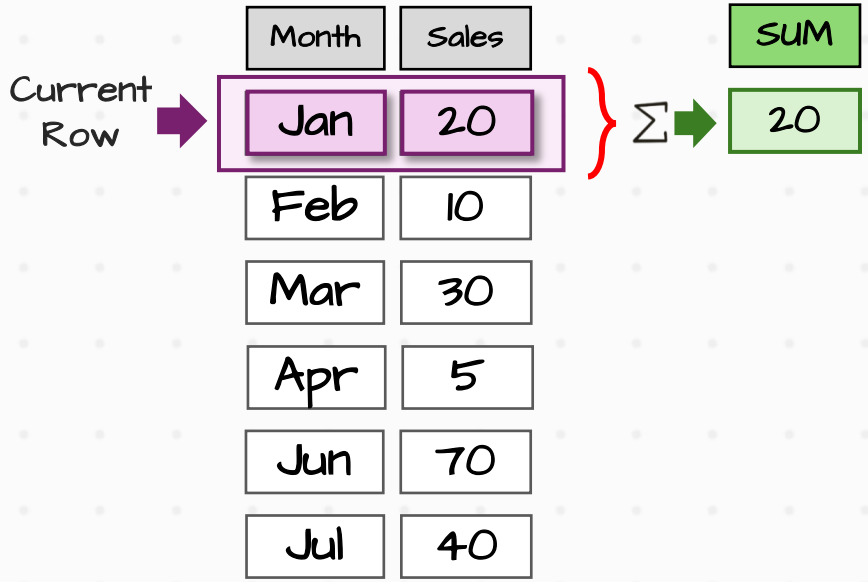
Running Total

Summarize all values from the first row up to the current row

```
SUM(Sales) OVER(
ORDER BY Month)
```

Default Frame

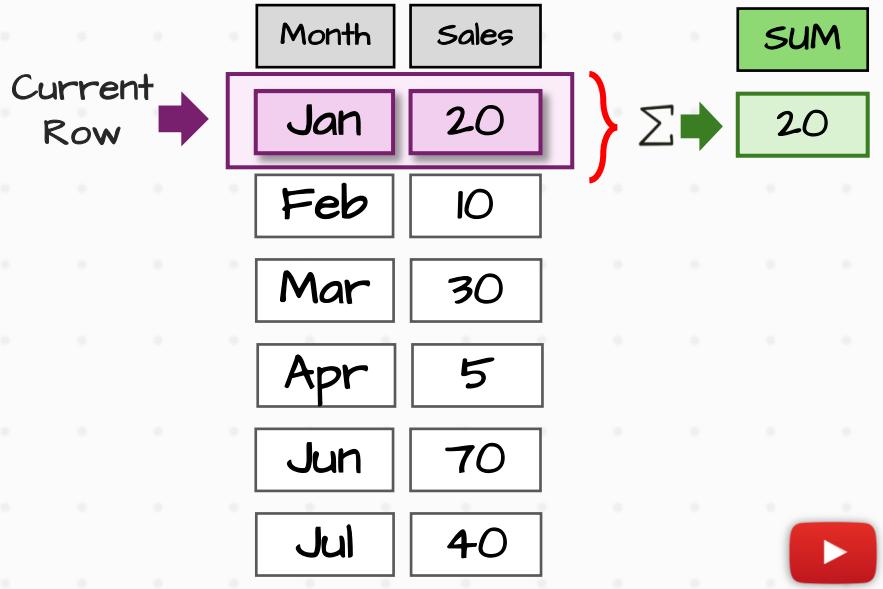
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

Summarize a fixed number of consecutive rows calculated within a moving window

```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```



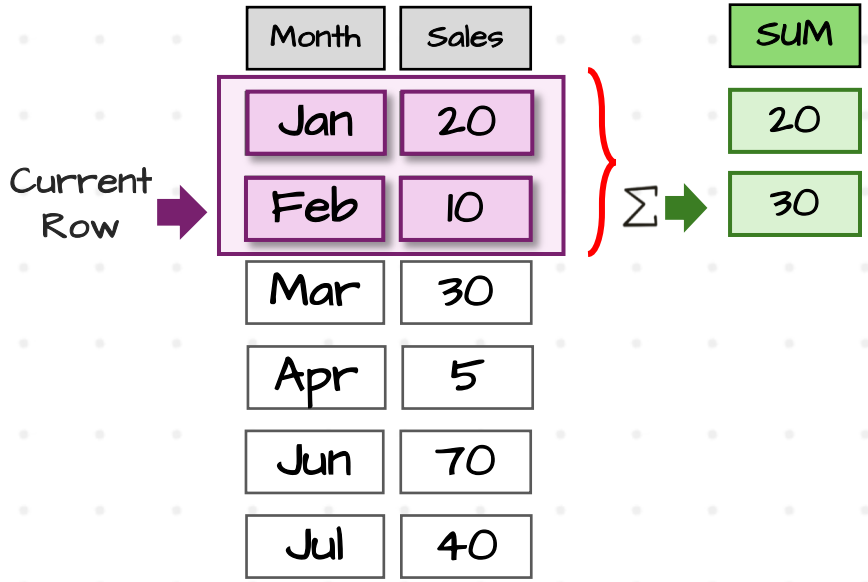
Running Total

Summarize all values from the first row up to the current row

```
SUM(Sales) OVER(
ORDER BY Month)
```

Default Frame

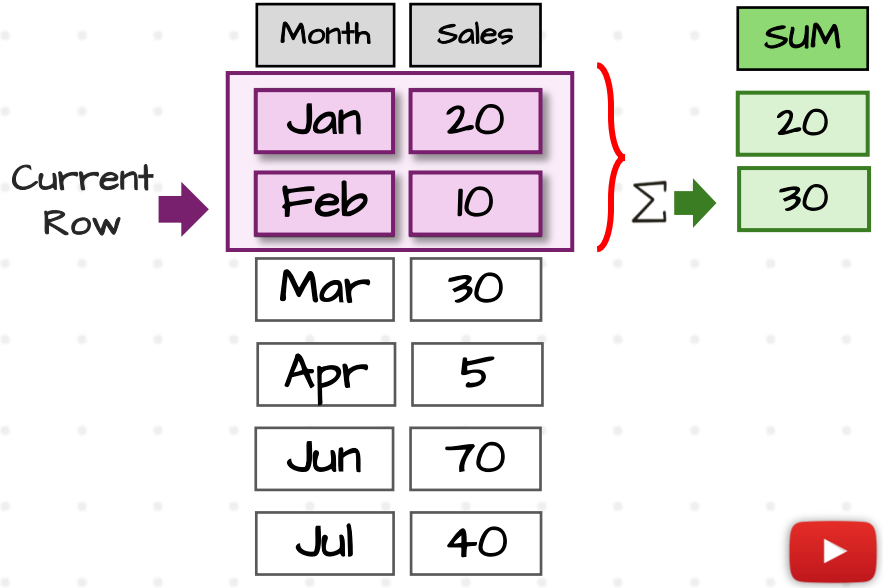
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

Summarize a fixed number of consecutive rows calculated within a moving window

```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```



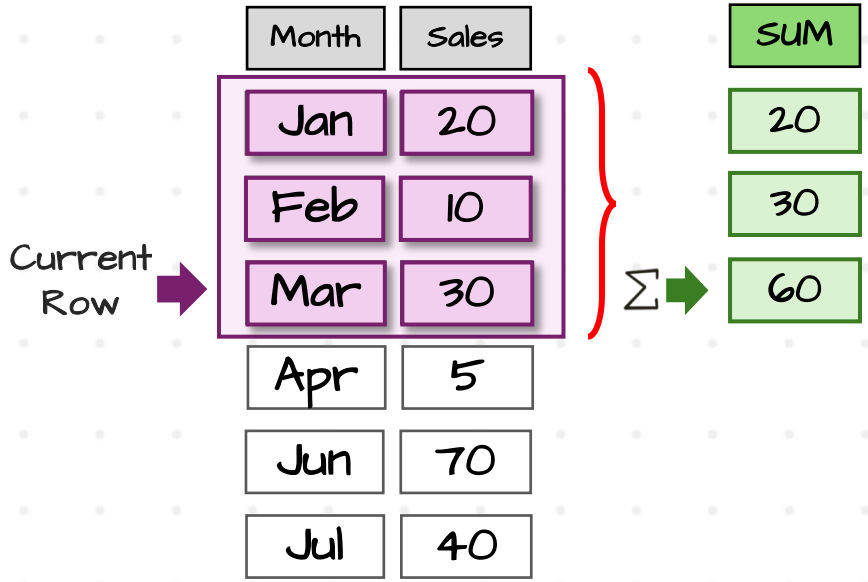
Running Total

Summarize all values from the first row up to the current row

```
SUM(Sales) OVER(
ORDER BY Month)
```

Default Frame

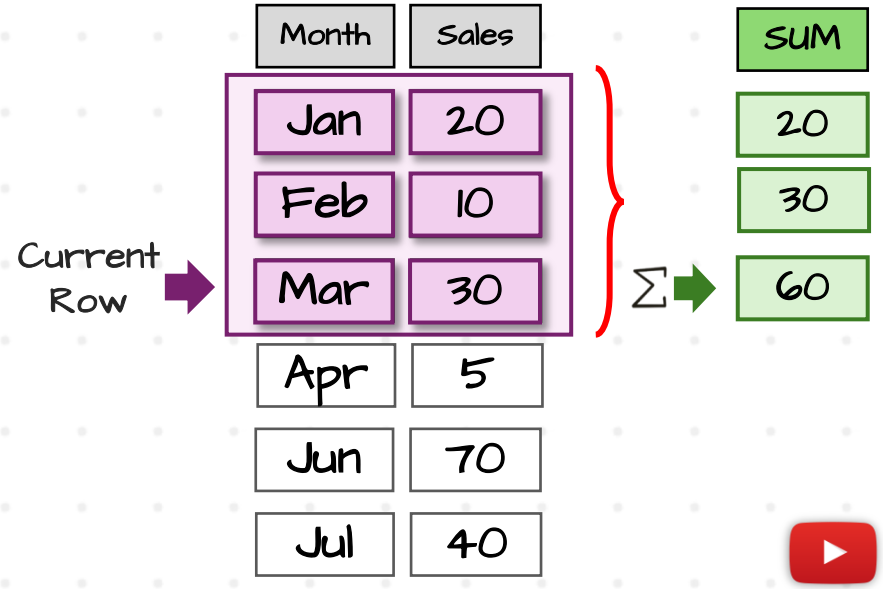
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

Summarize a fixed number of consecutive rows calculated within a moving window

```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```



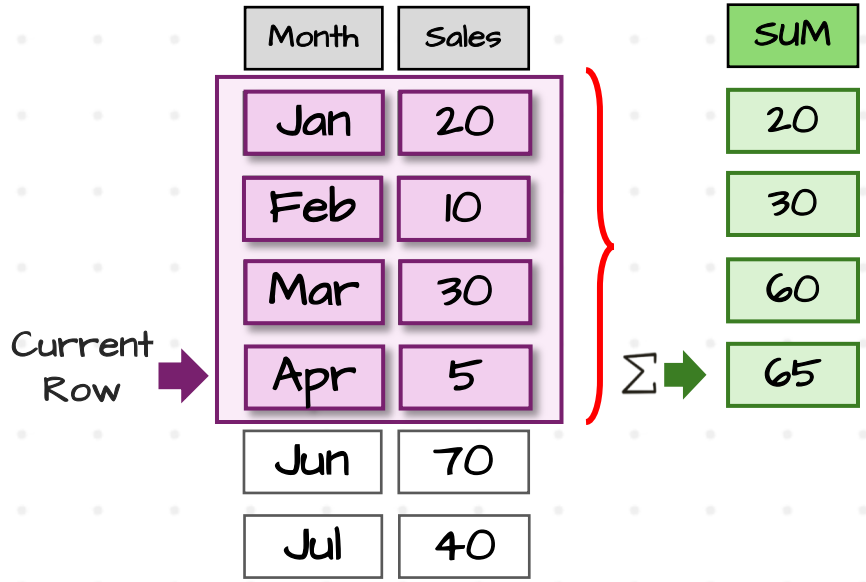
Running Total

Summarize all values from the first row up to the current row

```
SUM(Sales) OVER(
ORDER BY Month)
```

Default Frame

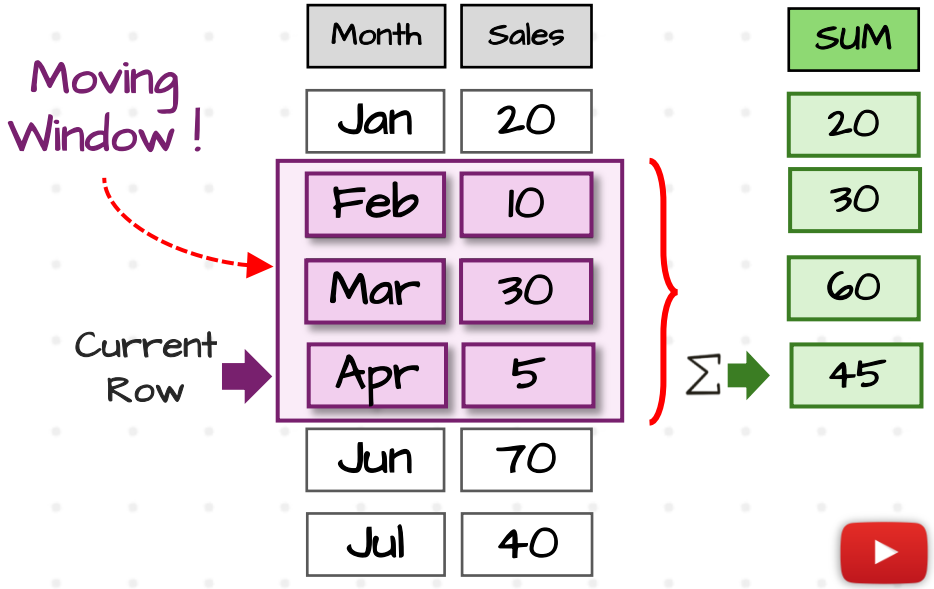
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

Summarize a fixed number of consecutive rows calculated within a moving window

```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```



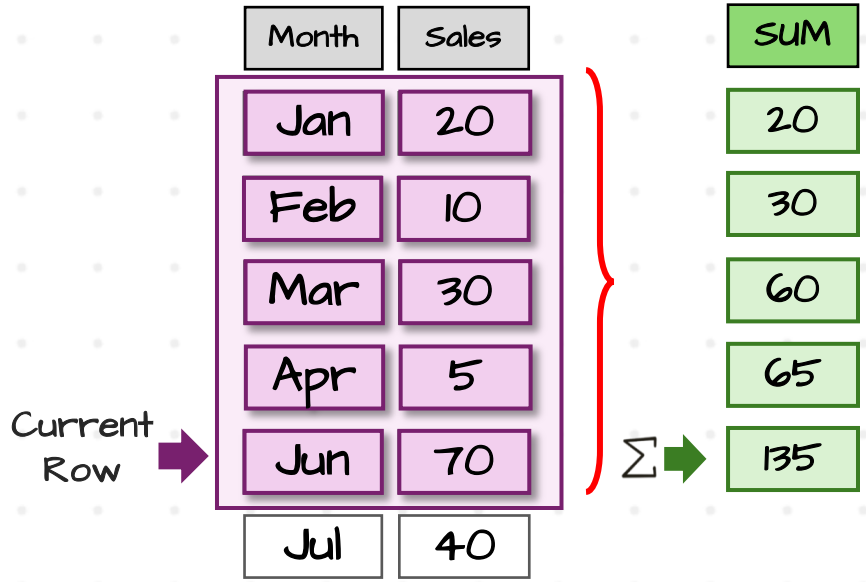
Running Total

Summarize all values from the first row up to the current row

```
SUM(Sales) OVER(
ORDER BY Month)
```

Default Frame

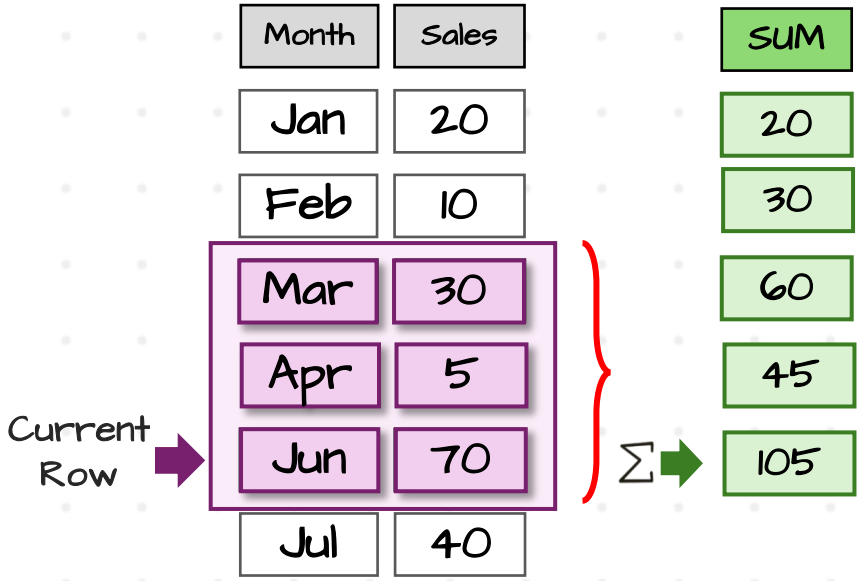
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

Summarize a fixed number of consecutive rows calculated within a moving window

```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```



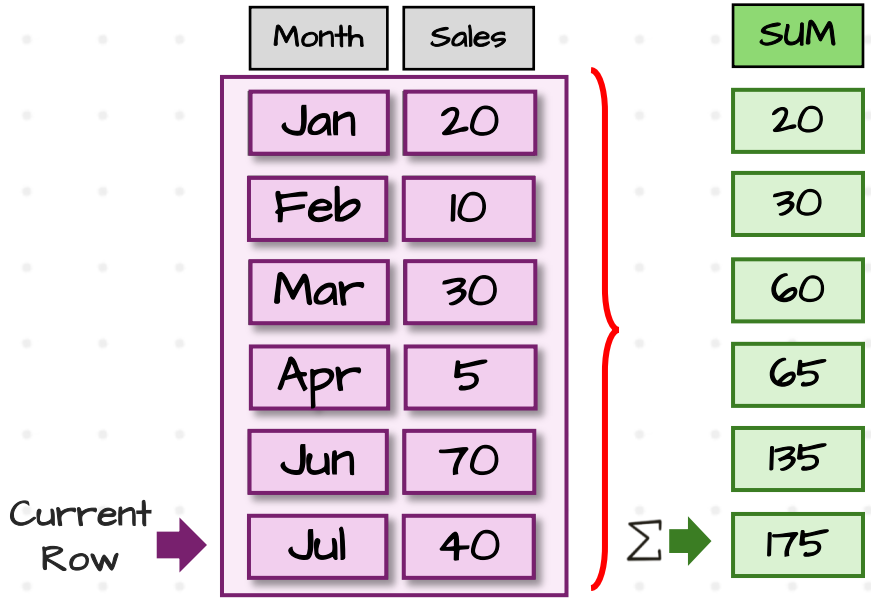
Running Total

Summarize all values from the first row up to the current row

```
SUM(Sales) OVER(
ORDER BY Month)
```

Default Frame

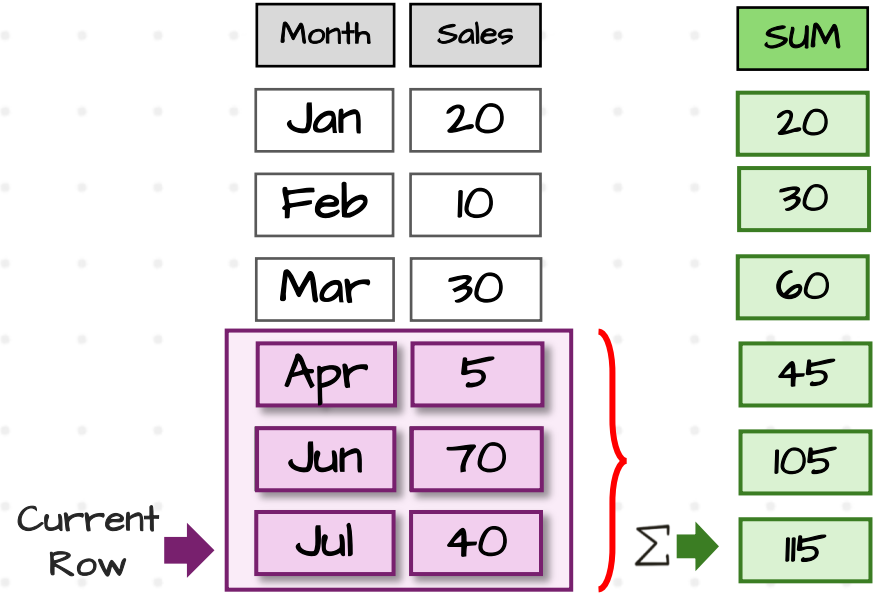
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

Summarize a fixed number of consecutive rows calculated within a moving window

```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```



Overall Total

```
SUM(Sales) OVER()
```

Overview of entire data

Month	Sales	SUM
Caps	20	175
Caps	10	175
Caps	30	175
Gloves	5	175
Gloves	70	175
Gloves	40	175

Total Per Groups

```
SUM(Sales) OVER(
PARTITION BY Product)
```

Compare Categories

Month	Sales	SUM
Caps	20	60
Caps	10	60
Caps	30	60
Gloves	5	105
Gloves	70	105
Gloves	40	105

Running Total

```
SUM(Sales) OVER(
ORDER BY MONTH)
```

progress over time

Month	Sales	SUM
Jan	20	20
Feb	10	30
Mar	30	60
Apr	5	65
Jun	70	135
Jul	40	175

Rolling Total

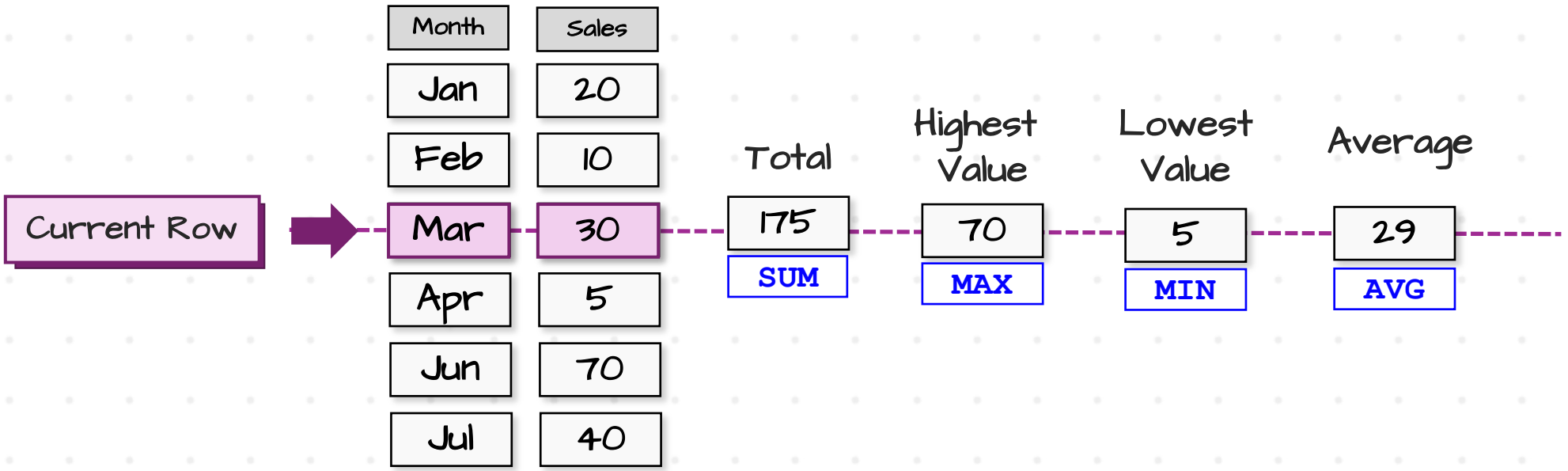
```
SUM(Sales) OVER(
ORDER BY MONTH
ROWS 2 PRECEDING)
```

progress over time in specific fixed window

Month	Sales	SUM
Jan	20	20
Feb	10	30
Mar	30	60
Apr	5	45
Jun	70	105
Jul	40	105

Comparision Use Cases

Compare the current value and aggregated value of window functions



WINDOW AGGREGATE FUNCTIONS

Aggregate set of values and return a single aggregated value

Rules

- Expressions
  - Numbers (All Functions)
  - Any Data Type – COUNT()
- All Clouses are Optional

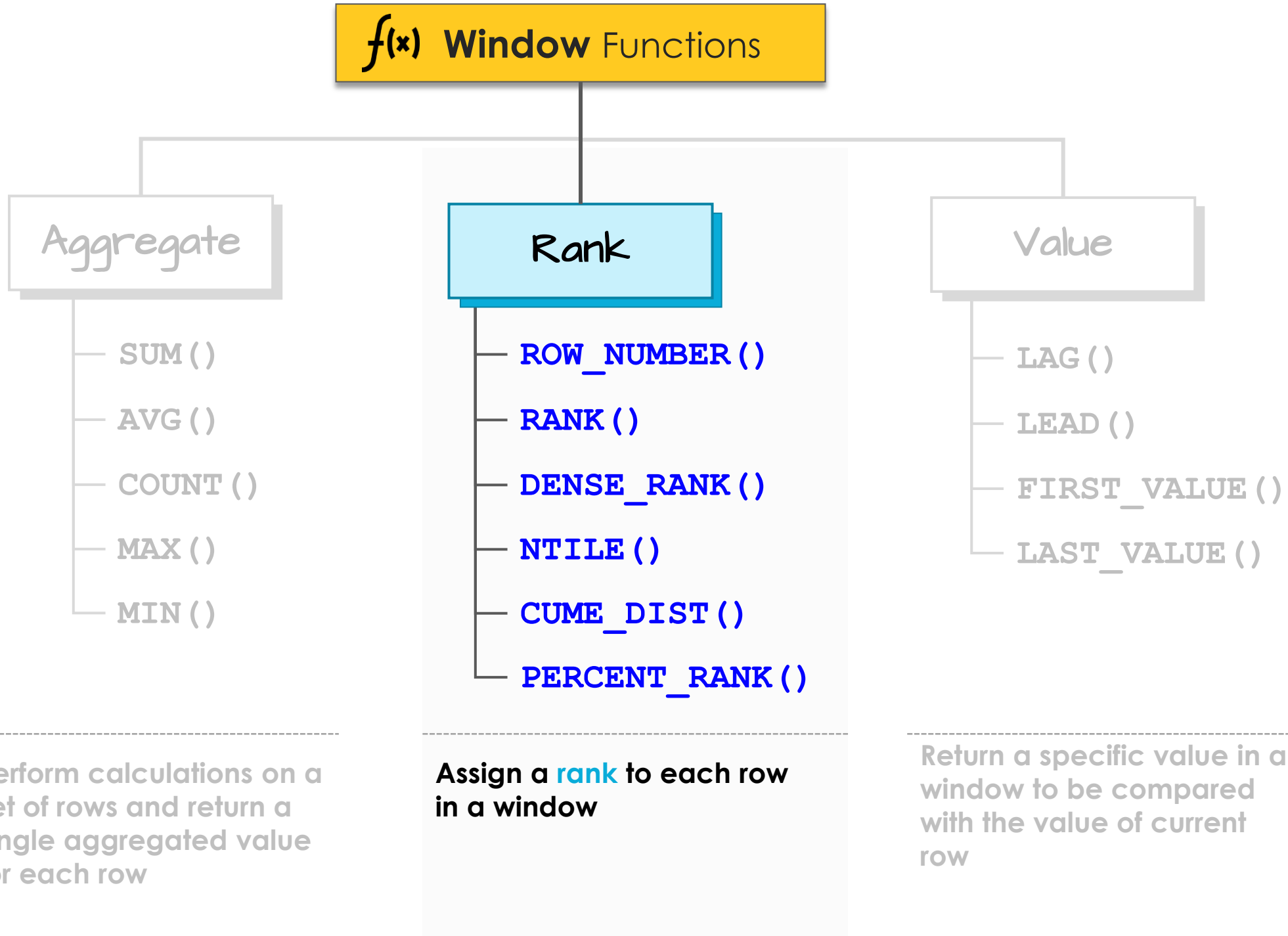
Use Cases

- Overall Analysis
- Total Per Groups Analysis
- Part-to-Whole Analysis
- Comparison Analysis
  - Average
  - Extreme: Highest/Lowest
- Identify Duplicates
- Outlier Detection
- Running Total
- Rolling Total
- Moving Average

DATA WITH BARAA

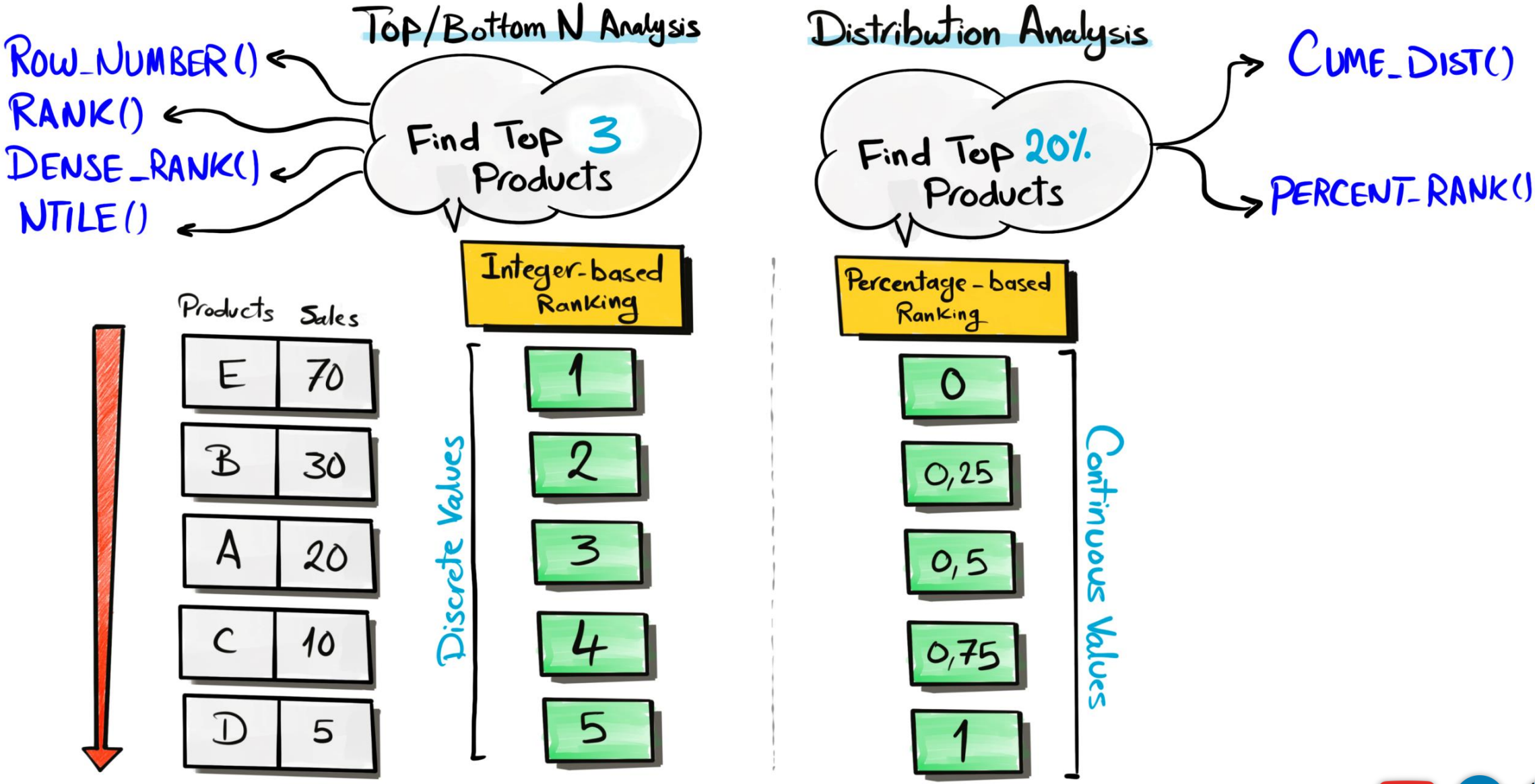
WINDOW RANKING FUNCTIONS

Baraa Khatib Salkini
YouTube | **DATA WITH BARAA**
SQL Course | Window Rank Functions


f(x) Window Functions
Aggregate
SUM()
AVG()
COUNT()
MAX()
MIN()
Rank
ROW_NUMBER()
RANK()
DENSE_RANK()
NTILE()
CUME_DIST()
PERCENT_RANK()
Value
LAG()
LEAD()
FIRST_VALUE()
LAST_VALUE()
Perform calculations on a set of rows and return a single aggregated value for each row
Assign a rank to each row in a window
Return a specific value in a window to be compared with the value of current row

Month	Sales	Rank
Jun	70	1
Jul	40	2
Mar	30	3
Jan	20	4
Feb	10	5
Apr	5	6

Assign a Rank (Number) for each Row



Ranking Function

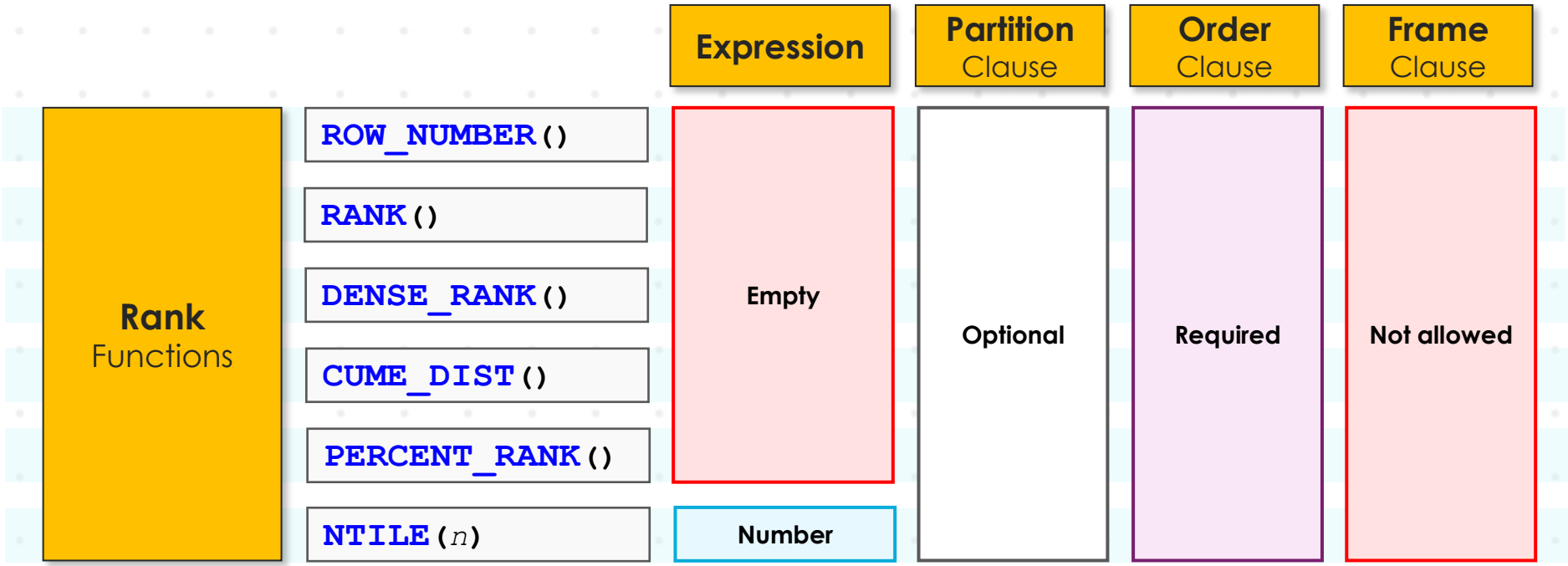

```
RANK() OVER(PARTITION BY ProductID ORDER BY Sales)
```

Expression must be empty

Partition By Is Optional

Order By Is required

Ranking Function



Ranking Function

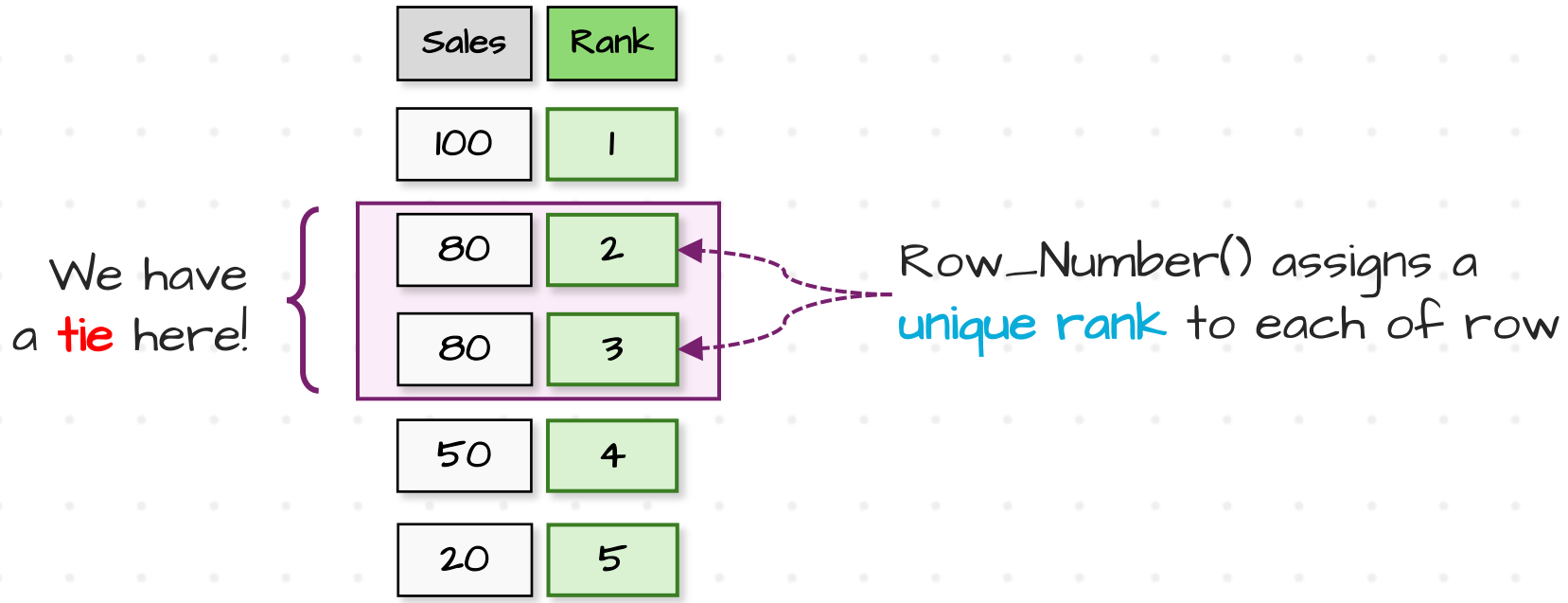
DATA WITH BARAA

Rank Functions			
	`ROW_NUMBER()`	Assign a unique number to each in a window	`ROW_NUMBER() OVER (ORDER BY Sales)`
	`RANK()`	Assign a rank to each row in a window, with gaps	`RANK() OVER (ORDER BY Sales)`
	`DENSE_RANK()`	Assign a rank to each row in a window, without gaps	`DENSE_RANK() OVER (ORDER BY Sales)`
	`CUME_DIST()`	calculates the cumulative distribution of a value within a set of values	`CUME_DIST() OVER (ORDER BY Sales)`
	`PERCENT_RANK()`	Returns the percentile ranking number of a row.	`PERCENT_RANK() OVER (ORDER BY Sales)`
	`NTILE(n)`	Divides the rows into a specified number of approximately equal groups	`NTILE(2) OVER (ORDER BY Sales)`

ROW_NUMBER

Assign a unique sequential integer to each row with in a window

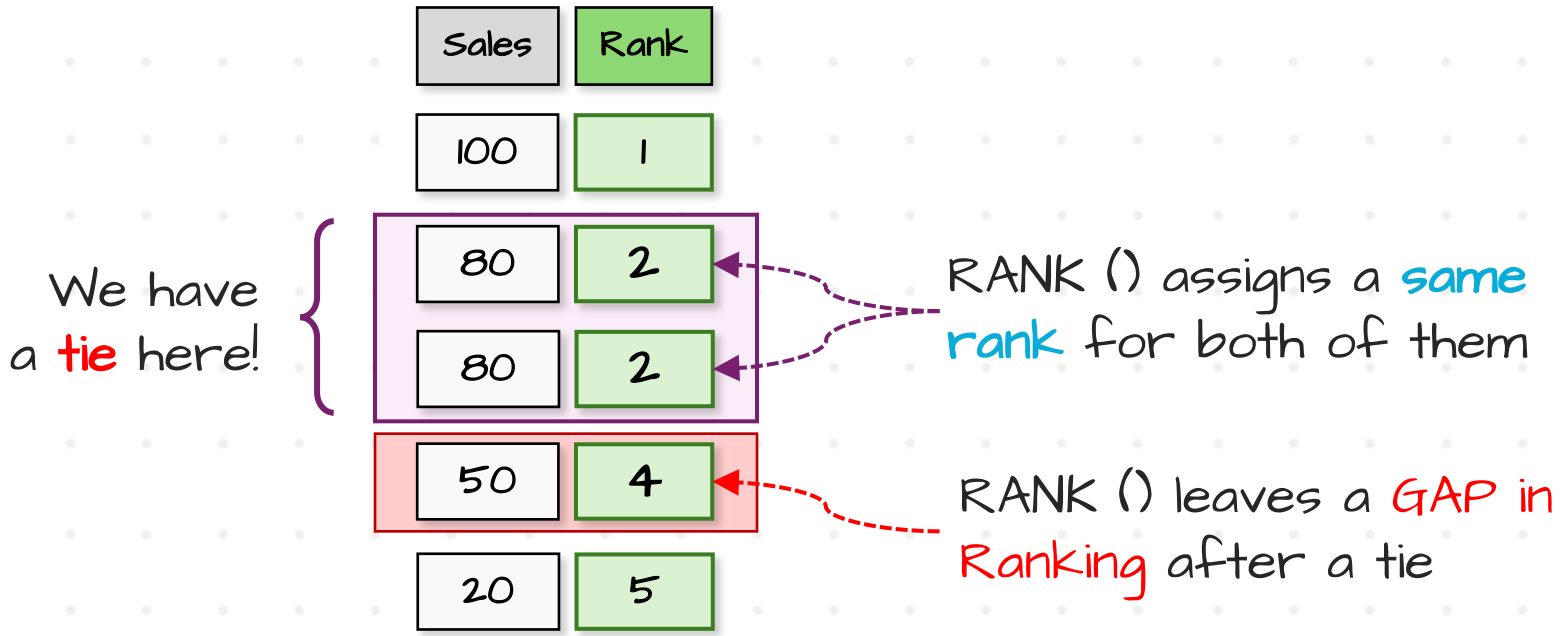
```
ROW_NUMBER() OVER (ORDER BY Sales DESC)
```



RANK

Assign a rank to each row with in a window

```
RANK() OVER(ORDER BY Sales DESC)
```

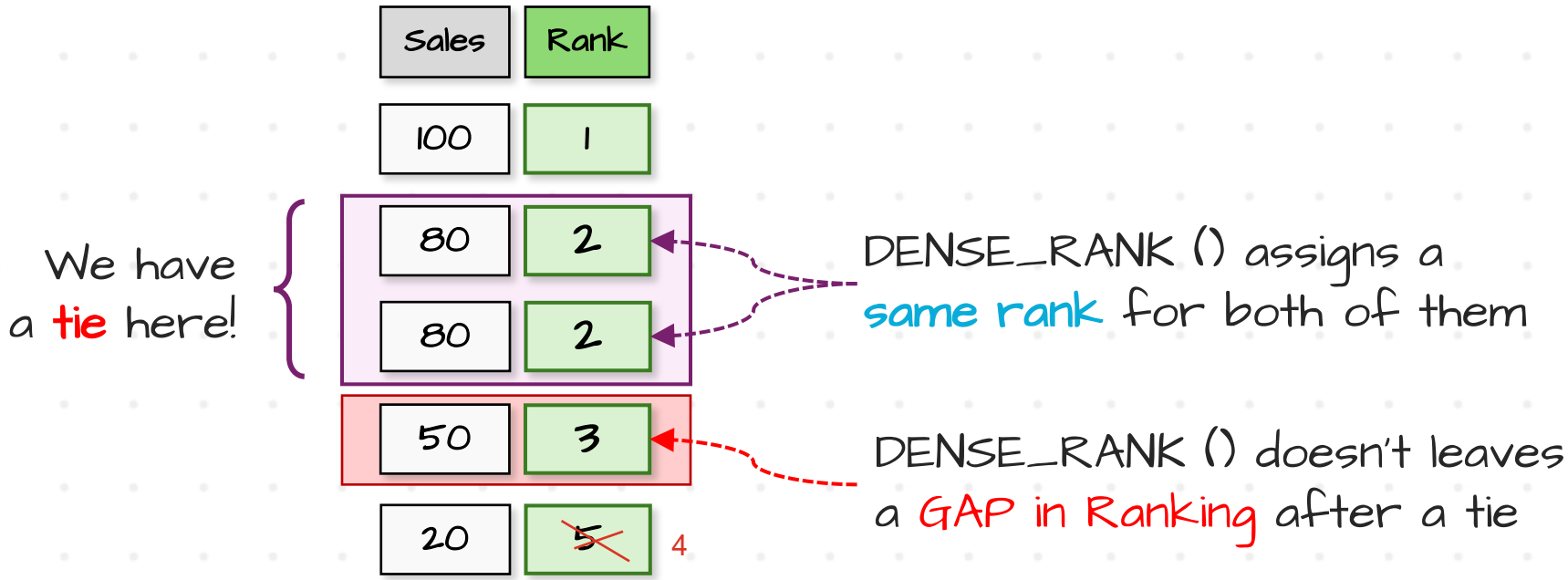


DENSE_RANK

Data with Baraa

Assign a rank to each row with in a window, but does not leave gaps in the ranking

```
DENSE_RANK() OVER(ORDER BY Sales DESC)
```



DATA WITH BARAA

ROW_NUMBER()

```
ROW_NUMBER() OVER(ORDER BY Sales DESC)
```

Sales	Rank
100	1
80	2
80	3
50	4
20	5

- Unique Rank
- Does NOT handle Ties
- No Gaps in Ranks

RANK()

```
RANK() OVER(ORDER BY Sales DESC)
```

Sales	Rank
100	1
80	2
80	2
50	4
20	5

- Shared Rank
- Handles Ties
- Gaps in Ranks

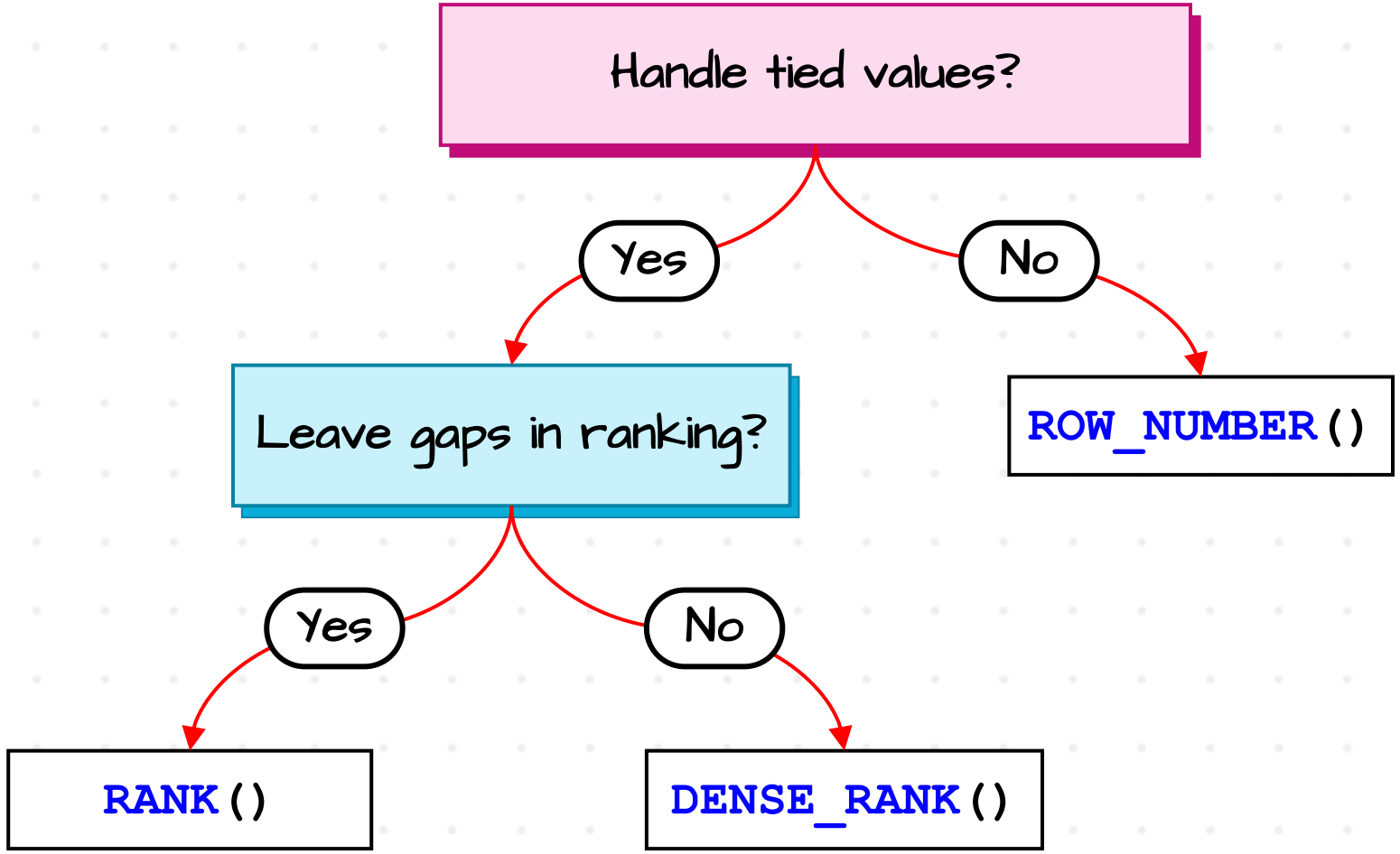
DENSE_RANK()

```
DENSE_RANK() OVER(ORDER BY Sales DESC)
```

Sales	Rank
100	1
80	2
80	2
50	3
20	4

- Shared Rank
- Handles Ties
- No Gaps in Ranks

Which One To Use?



NTILE()

Divides the rows into a specified number of approximately equal groups (Buckets)

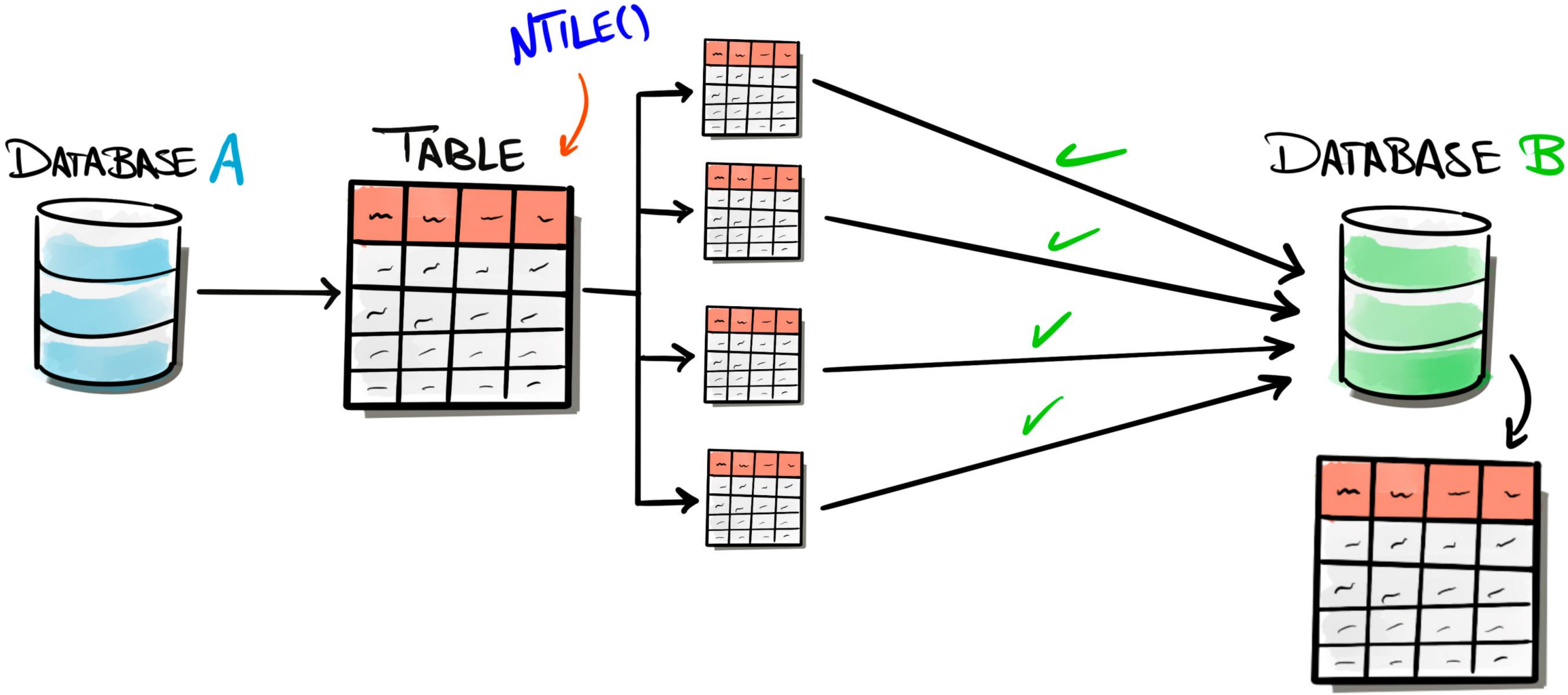
DATA ANALYST

Data Segmentation

DATA ENGINEER

Equalizing load processing

NTILE Use Case



NTILE use Case

```
NTILE(2) OVER (ORDER BY Sales DESC)
```

Sales	NTILE
100	1
80	1
80	1
50	2
30	2

$$\text{Bucket Size} = \frac{\text{Number of Rows}}{\text{Number of Buckets}}$$

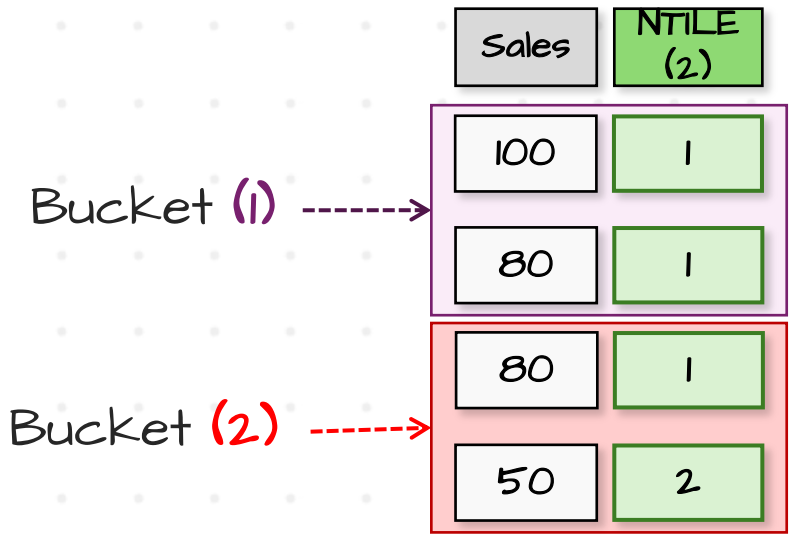
$$2 = \frac{5}{2}$$

NTILE

Divides the rows into a specified number of approximately equal groups (buckets)

```
NTILE(2) OVER (ORDER BY Sales DESC)
```

Number of Buckets



$$\text{Bucket Size} = \frac{\text{Number of Rows}}{\text{Number of Buckets}}$$

(Nr of Rows in each Bucket)

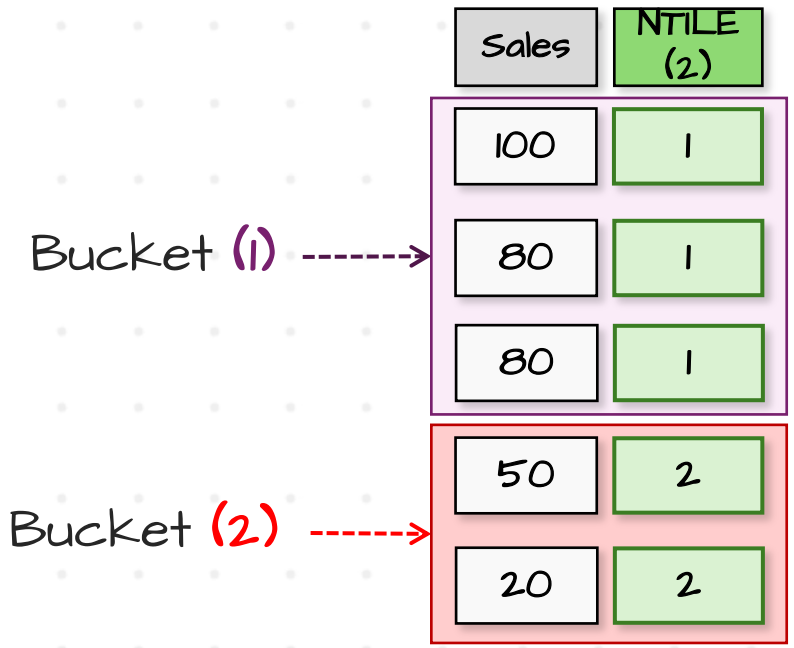
$$\text{Bucket Size} = \frac{4}{2} = 2$$

NTILE

Divides the rows into a specified number of approximately equal groups (buckets)

```
NTILE(2) OVER (ORDER BY Sales DESC)
```

Number of Buckets



$$\text{Bucket Size} = \frac{\text{Number of Rows}}{\text{Number of Buckets}}$$

(Nr of Rows in each Bucket)

$$\text{Bucket Size} = \frac{5}{2} = 2.5$$

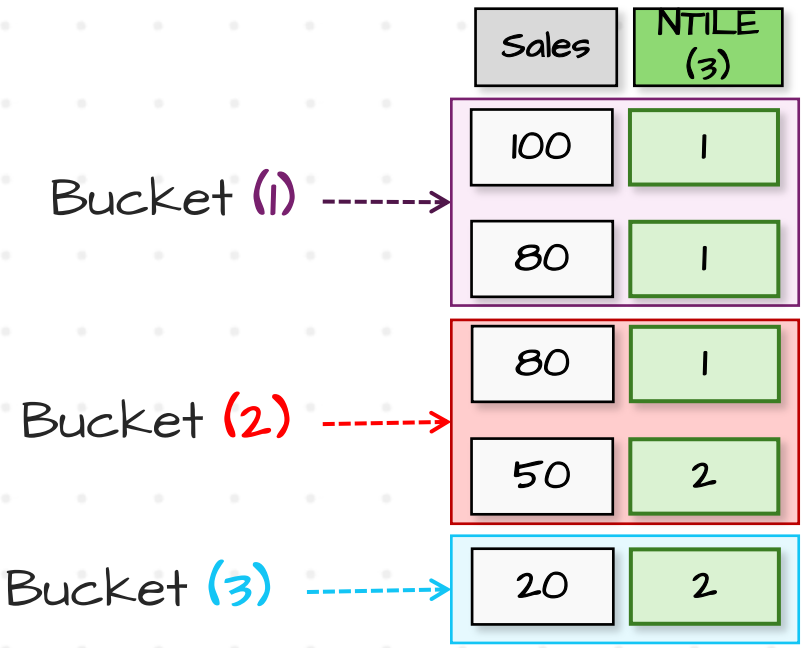
Larger groups come first then smaller groups

NTILE

Divides the rows into a specified number of approximately equal groups (buckets)

```
NTILE(3) OVER (ORDER BY Sales DESC)
```

Number of Buckets



$$\text{Bucket Size} = \frac{\text{Number of Rows}}{\text{Number of Buckets}}$$

(Nr of Rows in each Bucket)

$$\text{Bucket Size} = \frac{5}{3} = 1.7$$

Larger groups come first then smaller groups

CUME_DIST

```
CUME_DIST() OVER (ORDER BY Sales DESC)
```

Sales	DIST
100	0,2
80	0,6
80	0,6
50	0,8
30	1

$$\text{CUME\_DIST} = \frac{\text{Position Nr}}{\text{Number of Rows}}$$

$$\text{CUME\_DIST} = \frac{5}{5}$$

CUME_DIST()

Cumulative Distribution calculates the distribution of data points within a window

$$\frac{\text{Position Nr}}{\text{Number of Rows}}$$

Inclusive
(The current row is included)

PERCENT_RANK()

Calculates the relative position of each row

$$\frac{\text{Position Nr} - 1}{\text{Number of Rows} - 1}$$

Exclusive
(The current row is excluded)

```
CUME_DIST() OVER (ORDER BY Sales DESC)
```

```
PERCENT_RANK() OVER (ORDER BY Sales DESC)
```

Sales	DIST	Per
100	0,2	0
80	0,6	0,25
80	0,6	0,25
50	0,8	0,75
30	1	1

$$\text{CUME\_DIST} = \frac{\text{Position Nr}}{\text{Number of Rows}}$$

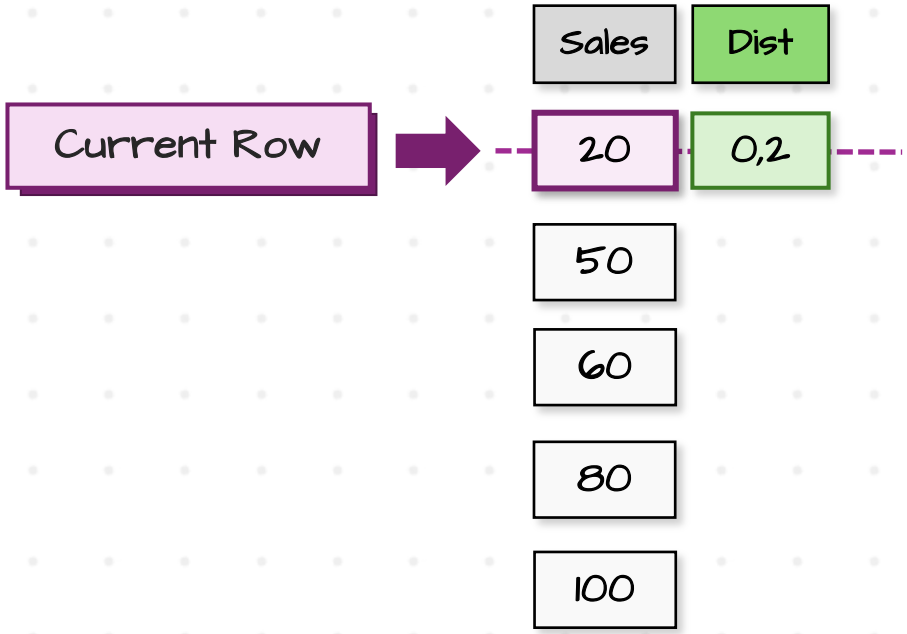
$$\text{Percent\_Rank} = \frac{\text{Position Nr} - 1}{\text{Number of Rows} - 1}$$

CUME_DIST

Data with Baraa

Calculates the relative position of a specified value in a group of values.

```
CUME_DIST() OVER (ORDER BY Sales)
```



$$\text{CUME\_DIST}(x) = \frac{\text{Number of Rows less than or equal to X}}{\text{Total Number of Rows}}$$

$$\text{CUME\_DIST}(20) = \frac{1}{5} = 0{,}2$$

CUME_DIST

Calculates the relative position of a specified value in a group of values.

```
CUME_DIST() OVER (ORDER BY Sales)
```

Sales	Dist
20	0,2
50	0,4
60	0,6
80	
100	

Current Row

$$\text{CUME\_DIST}(x) = \frac{\text{Number of Rows less than or equal to X}}{\text{Total Number of Rows}}$$

$$\text{CUME\_DIST}(60) = \frac{3}{5} = 0{,}6$$

CUME_DIST

Data with Baraa

Calculates the relative position of a specified value in a group of values.

```
CUME_DIST() OVER (ORDER BY Sales)
```

Sales	Dist
20	0,2
50	0,4
60	0,6
80	0,8
100	1

Current Row

$$CUME\_DIST(x) = \frac{\text{Number of Rows less than or equal to X}}{\text{Total Number of Rows}}$$

$$CUME\_DIST(100) = \frac{5}{5} = 1$$

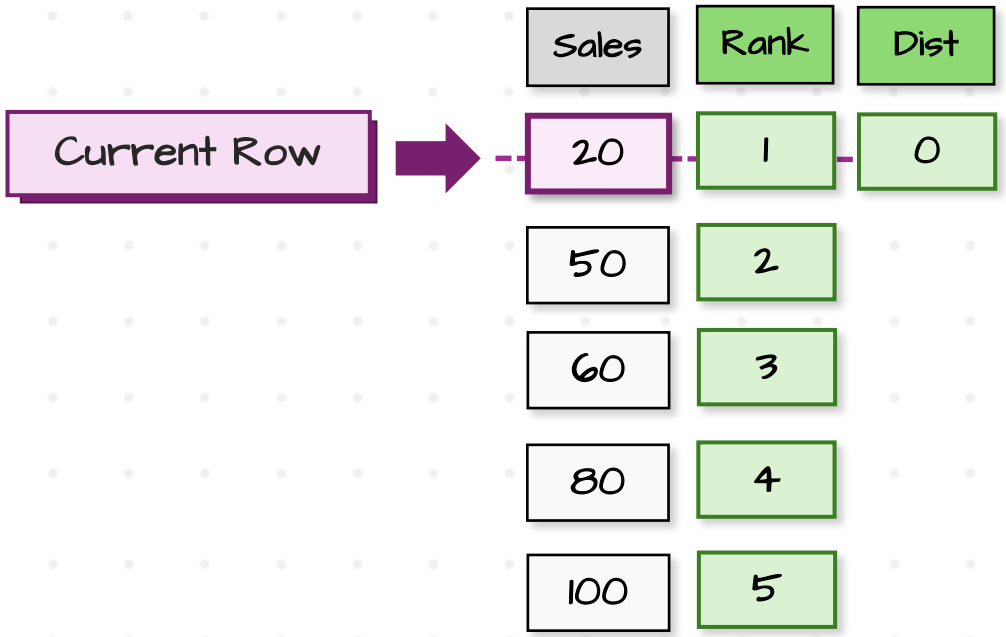
It returns values greater than 0 and less and equal to 1

PERCENT_RANK

DATA WITH BARAA

Returns the percentile ranking number of a row

```
PERCENT_RANK() OVER (ORDER BY Sales)
```



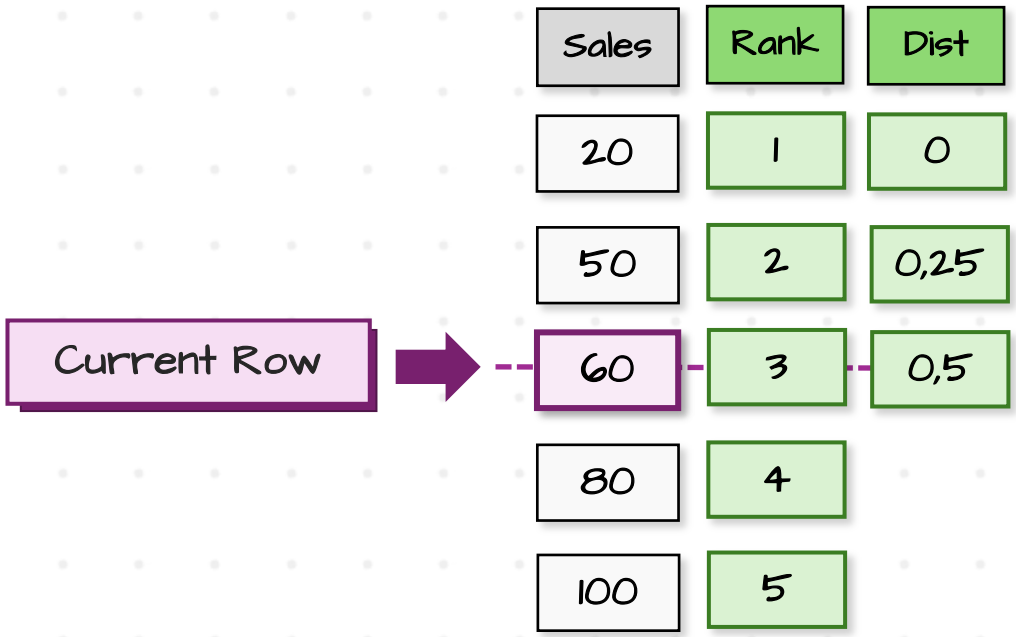
$$PERCENT\_RANK(x) = \frac{\text{Rank of X-1}}{\text{Total Number of Rows- 1}}$$

$$PERCENT\_RANK(20) = \frac{0}{4} = 0$$

PERCENT_RANK

Returns the percentile ranking number of a row

```
PERCENT_RANK() OVER (ORDER BY Sales)
```



$$PERCENT\_RANK(x) = \frac{\text{Rank of X-1}}{\text{Total Number of Rows- 1}}$$

$$PERCENT\_RANK(60) = \frac{2}{4} = 0.5$$

PERCENT_RANK

Data with Baraa

Returns the percentile ranking number of a row

```
PERCENT_RANK() OVER (ORDER BY Sales)
```

Sales	Rank	Dist
20	1	0
50	2	0,25
60	3	0,5
80	4	0,75
100	5	1

Current Row

$$PERCENT\_RANK(x) = \frac{\text{Rank of X-1}}{\text{Total Number of Rows- 1}}$$

$$PERCENT\_RANK(100) = \frac{4}{4} = 1$$

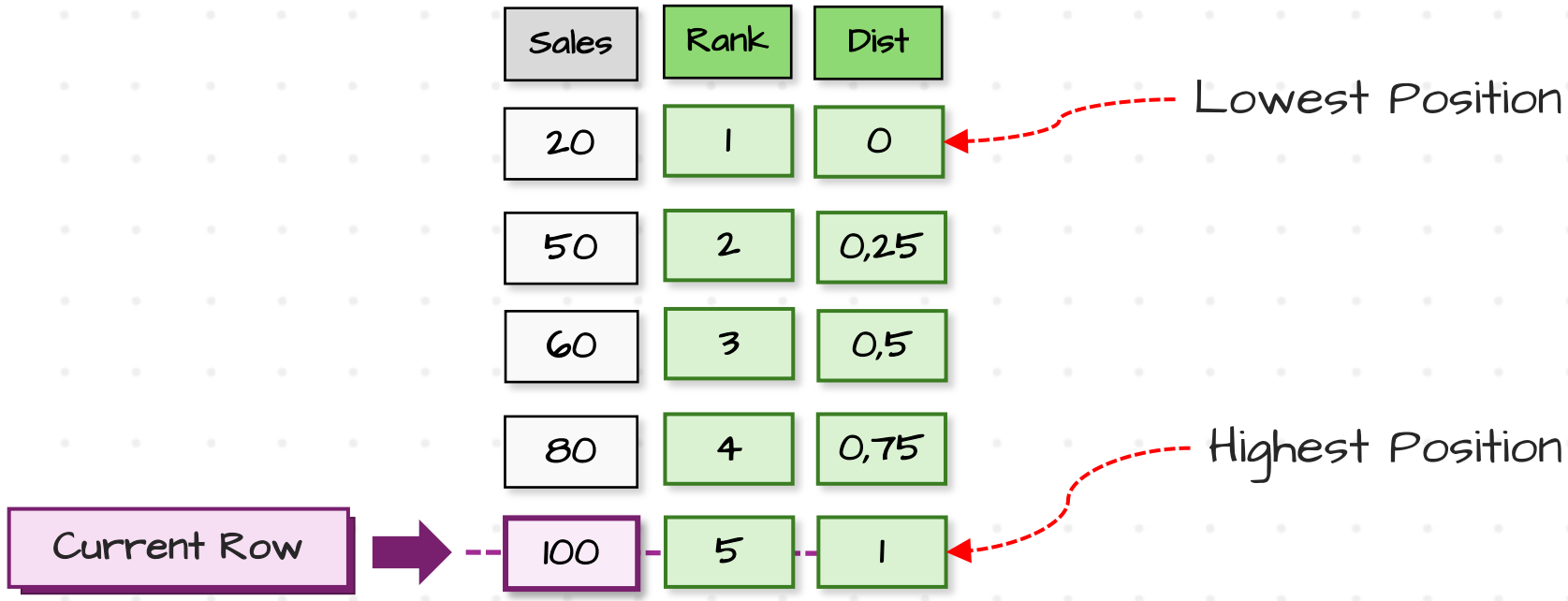
It returns values between 0 and 1

PERCENT_RANK

Data with Baraa

Returns the percentile ranking number of a row

```
PERCENT_RANK() OVER (ORDER BY Sales)
```



WINDOW RANK FUNCTIONS

DATA WITH BARAA

Assign a **RANK** for each row within a window

Types

- Integer-based-Ranking
  - ROW_NUMBER()
  - RANK()
  - DENSE_RANK()
  - NTILE()
- Percentage-based-Ranking
  - CUME_DIST()
  - PERCENT_RANK()

Rules

- Expression → Empty
- ORDER BY → Required
- FRAME → Not Allowed

Use Cases

- Top N Analysis
- Bottom N Analysis
- Identify & Remove Duplicates
- Assign Unique IDs & Pagination
- Data Segmentation
- Data Distribution Analysis
- Equalizing Load Processing

DATA WITH BARAA

WINDOW VALUE FUNCTIONS

Baraa Khatib Salkini
YouTube | **DATA WITH BARAA**
SQL Course | Window Value Functions

f(x) Window Functions

Aggregate

- SUM()
- AVG()
- COUNT()
- MAX()
- MIN()

Perform calculations on a set of rows and return a single aggregated value for each row

Rank

- ROW_NUMBER()
- RANK()
- DENSE_RANK()
- NTILE()
- CUME_DIST()
- PERCENT_RANK()

Assign a rank to each row in a window

Value

- **LAG()**
- **LEAD()**
- **FIRST_VALUE()**
- **LAST_VALUE()**

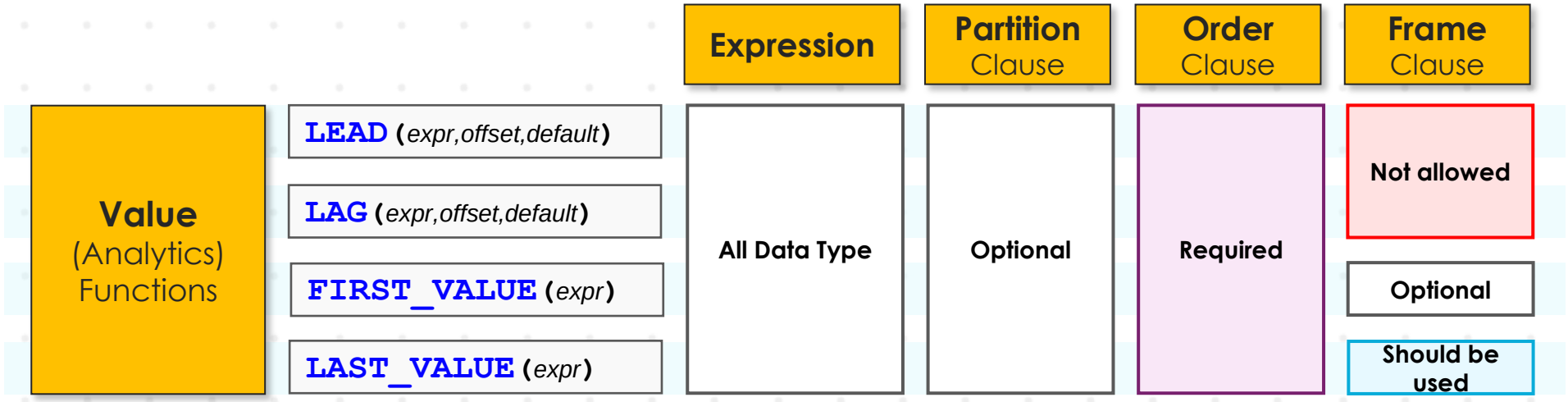
Return a specific value in a window to be compared with the value of current row

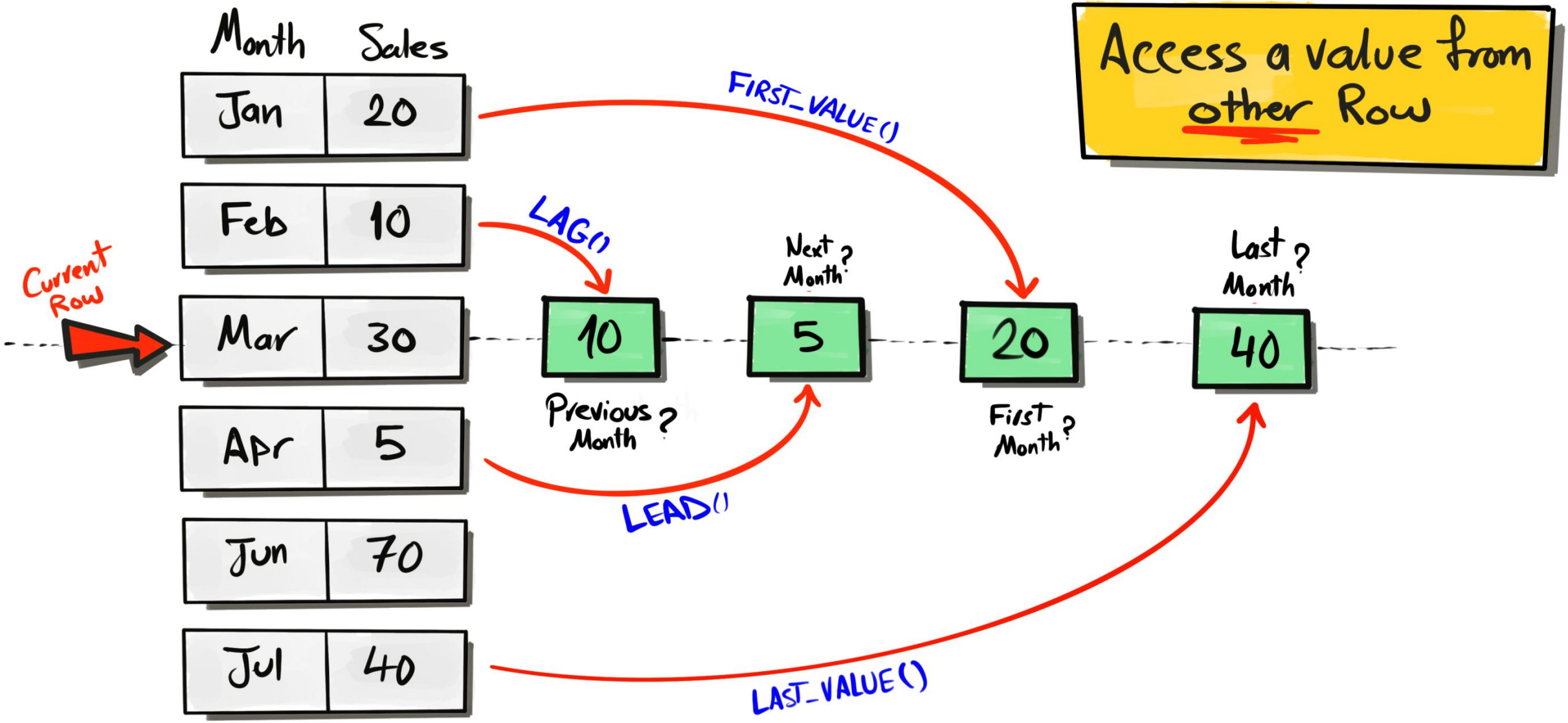
Value Functions

DATA WITH BARAA

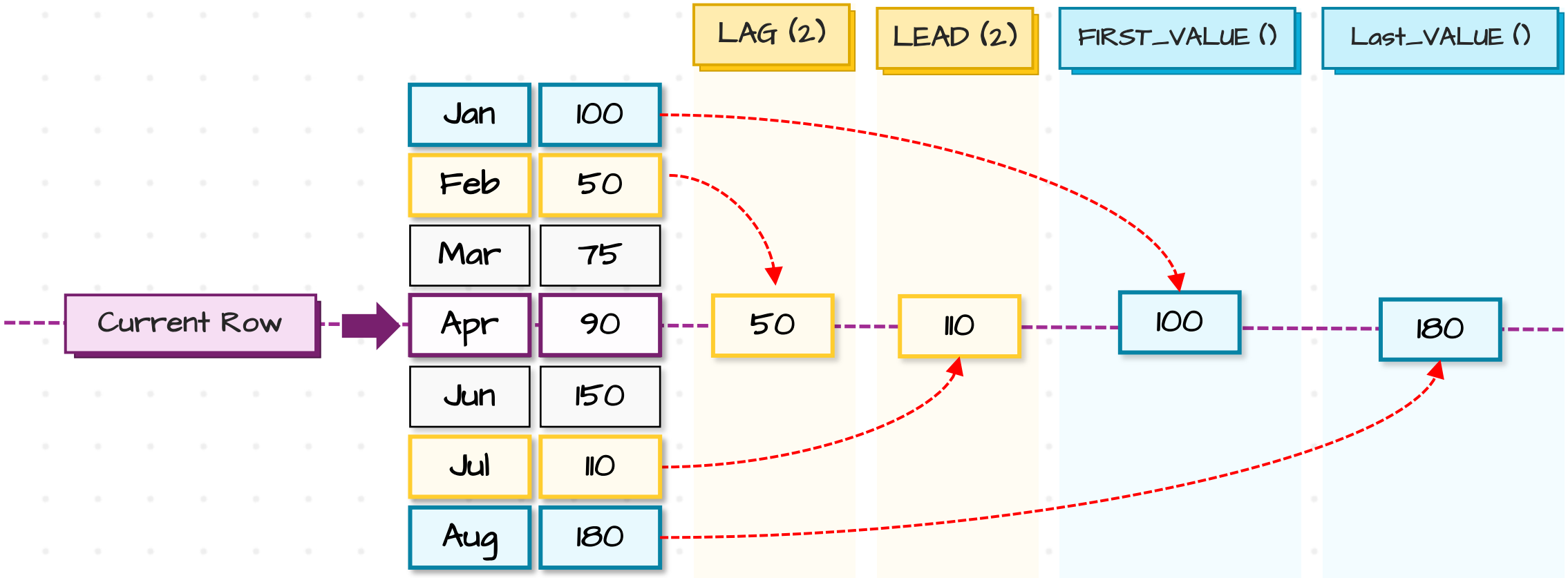
Value (Analytics) Functions	**LEAD** (*expr,offset,default*)	Returns the value from a previews row	`LEAD(Sales,2,0) OVER (ORDER BY OrderDate)`
	LAG (*expr,offset,default*)	Returns the value from a subsequent row	`LAG(Sales,2,0) OVER (ORDER BY OrderDate)`
	FIRST_VALUE (*expr*)	Returns the first value in a window	`FIRST_VALUE(Sales) OVER (ORDER BY OrderDate)`
	LAST_VALUE (*expr*)	Returns the last value in a window	`FIRST_VALUE(Sales) OVER (ORDER BY OrderDate)`

Value Functions



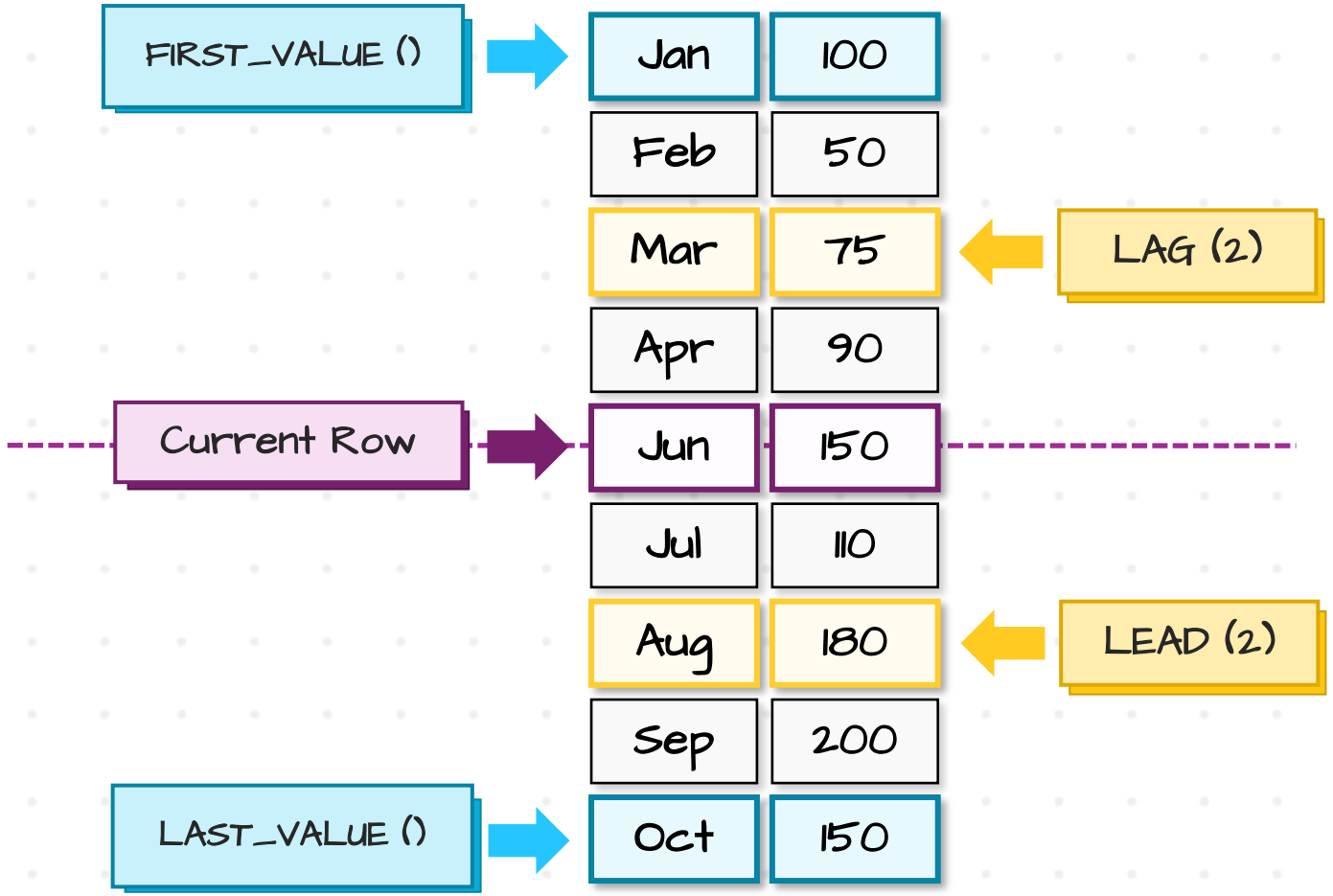


Value Functions

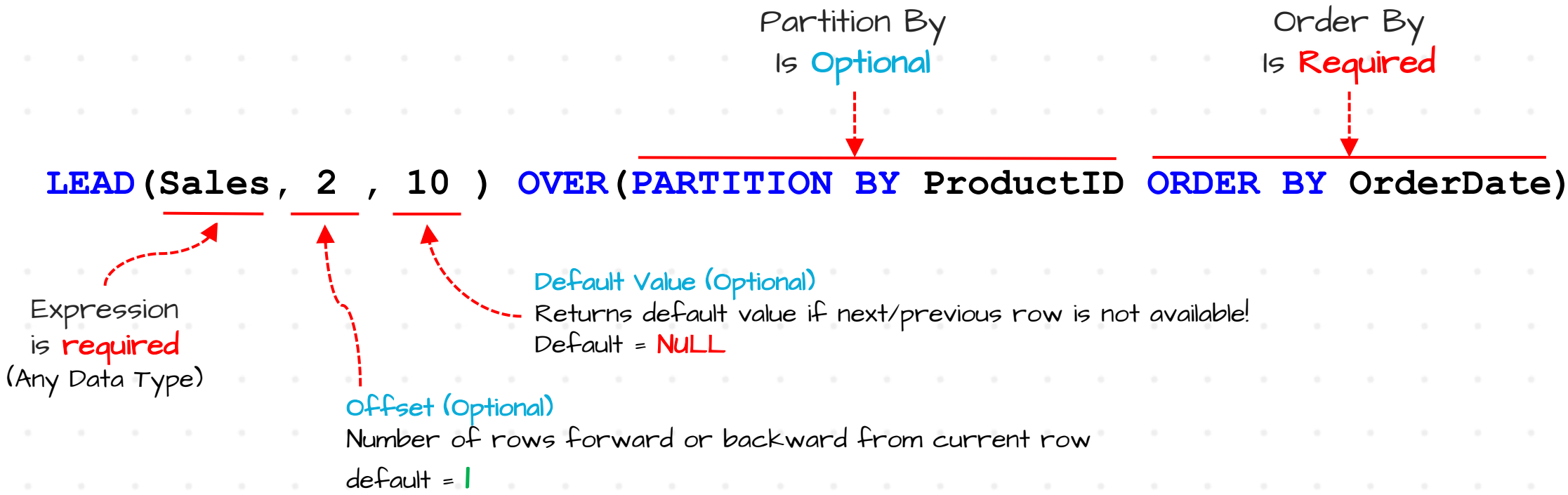


Value Functions

Return a specific value in a window to be compared with the value of current row

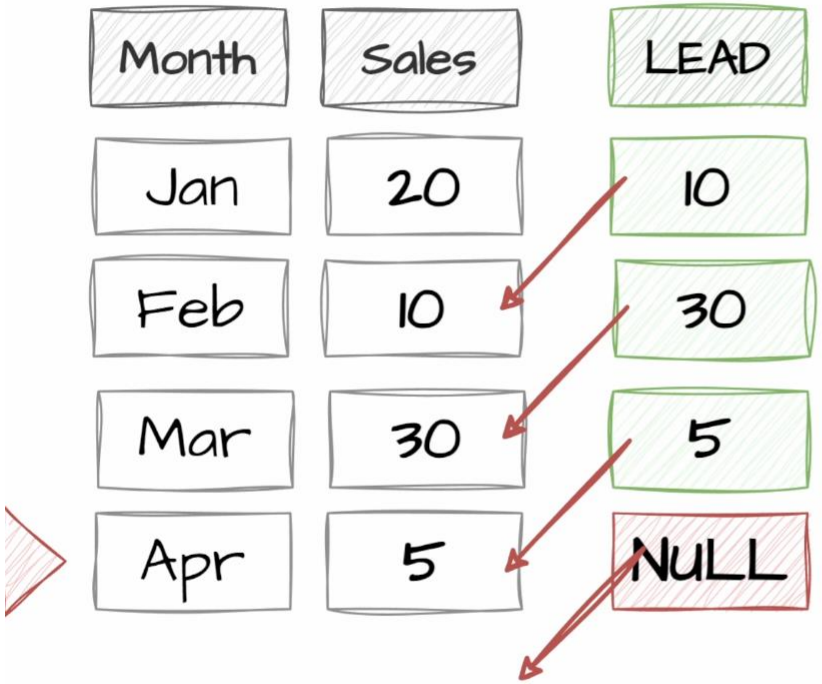


LEAD & LAG


Partition By
Is Optional
Order By
Is Required
LEAD(Sales, 2 , 10) OVER(PARTITION BY ProductID ORDER BY OrderDate)
Expression
is required
(Any Data Type)
Offset (Optional)
Number of rows forward or backward from current row
default = 1
Default Value (Optional)
Returns default value if next/previous row is not available!
Default = NULL

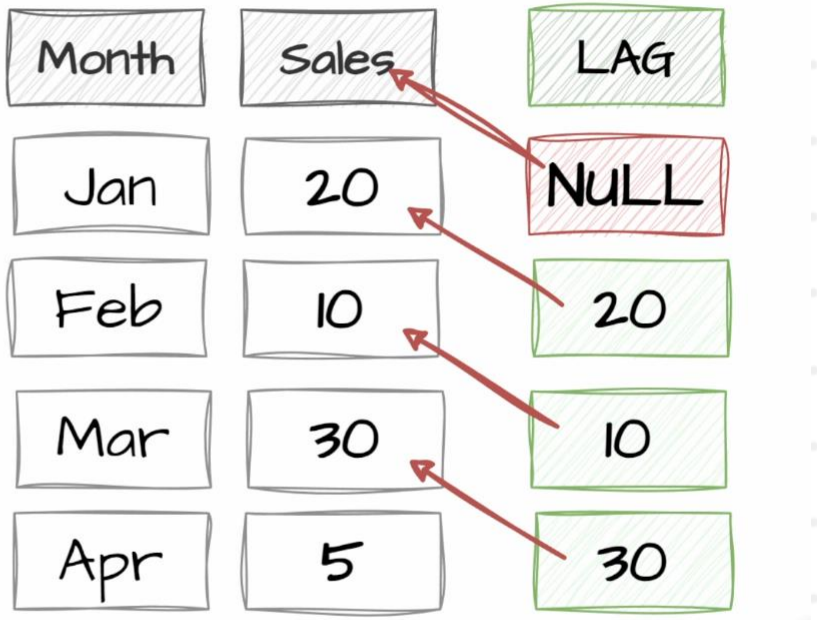
LEAD & LAG


```
LEAD(Sales) OVER (ORDER BY Month)
```



Find Sales of
the next month

```
LAG(Sales) OVER (ORDER BY Month)
```

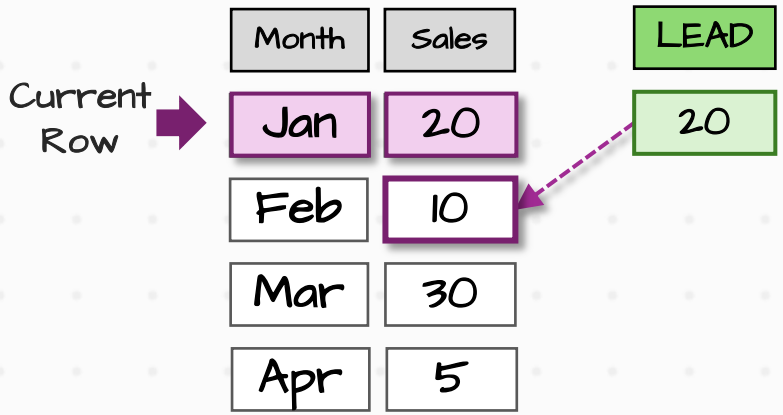


Find Sales of
the previous month

LEAD

Access Next Row

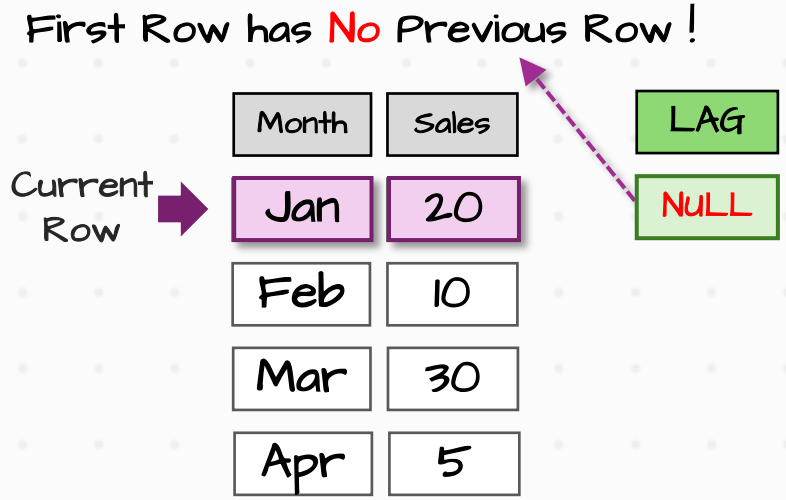
```
LEAD(Sales) OVER( ORDER BY Month)
```



LAG

Access Previous Row

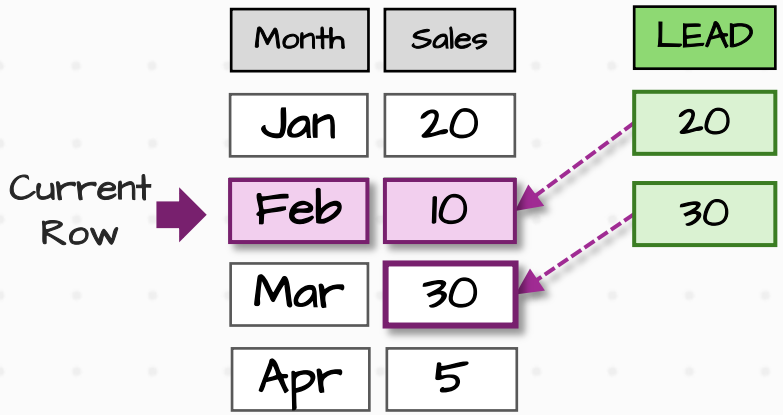
```
LEAD(Sales) OVER( ORDER BY Month)
```



LEAD

Access Next Row

```
LEAD(Sales) OVER( ORDER BY Month)
```

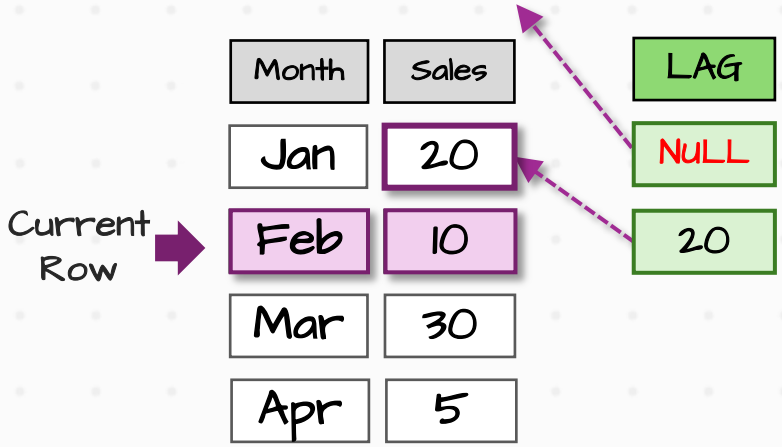


LAG

Access Previous Row

```
LEAD(Sales) OVER( ORDER BY Month)
```

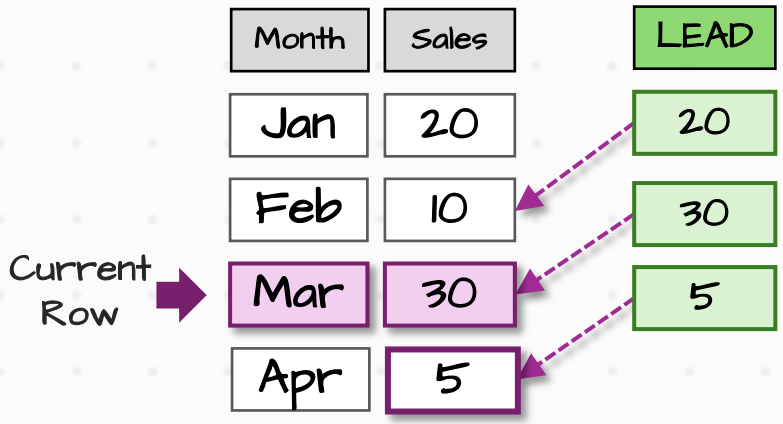
First Row has No Previous Row !



LEAD

Access Next Row

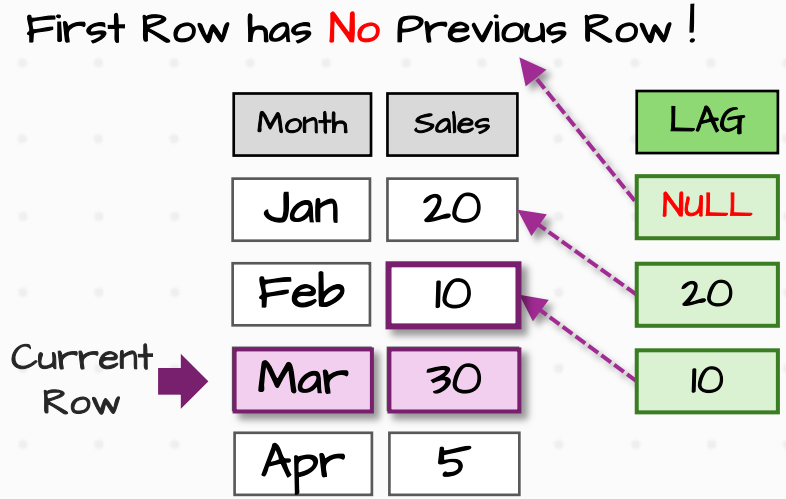
```
LEAD(Sales) OVER( ORDER BY Month)
```



LAG

Access Previous Row

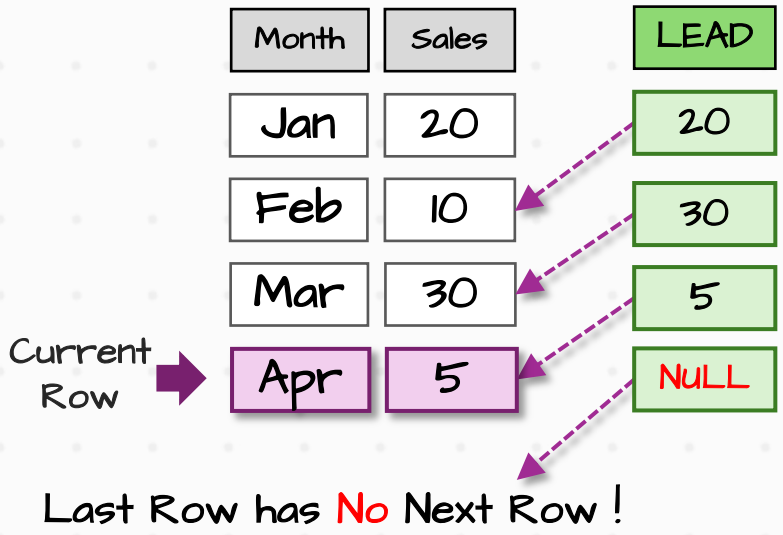
```
LEAD(Sales) OVER( ORDER BY Month)
```



LEAD

Access Next Row

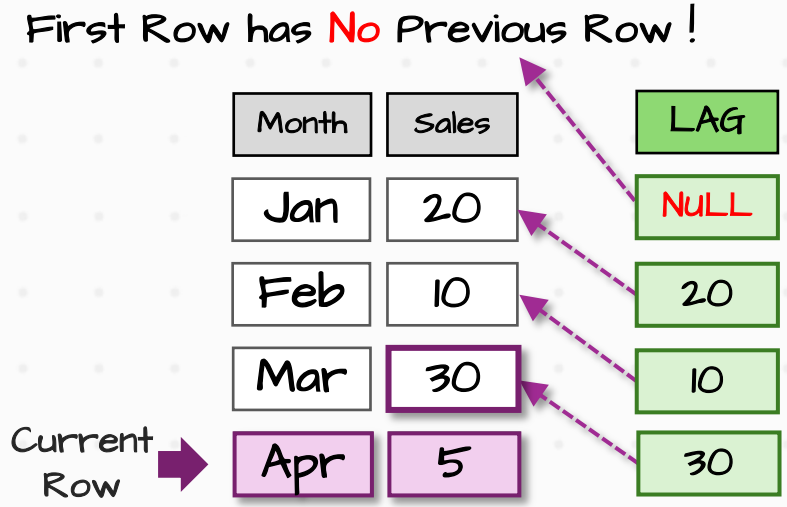
```
LEAD(Sales) OVER( ORDER BY Month)
```



LAG

Access Previous Row

```
LEAD(Sales) OVER( ORDER BY Month)
```



TIME SERIES ANALYSIS

Year-over-Year (YoY)

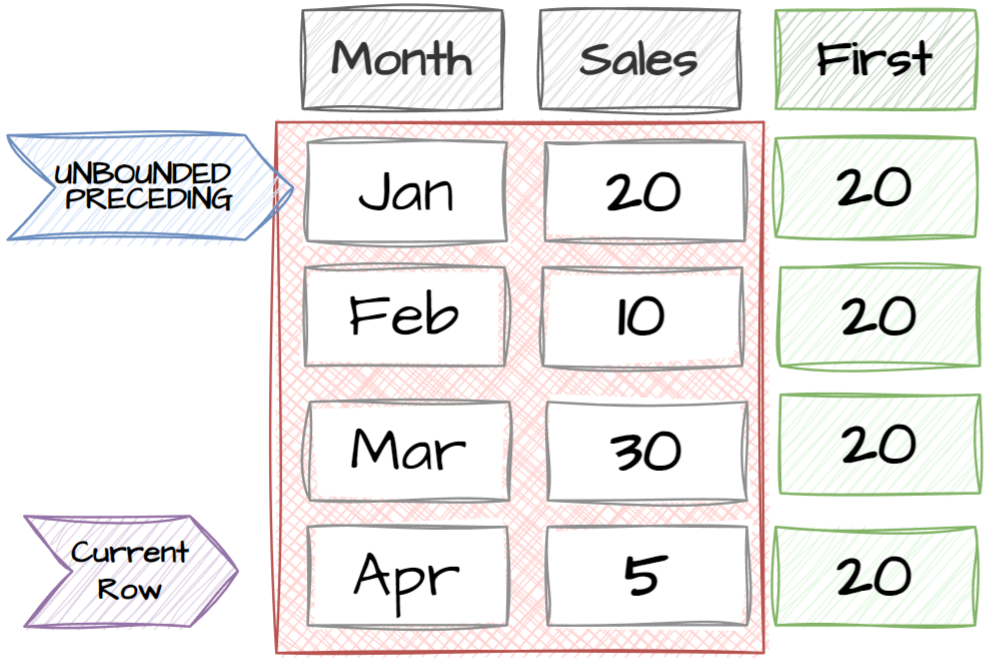
Analyze the overall growth or decline of the business's performance over time

Month-over-Month (MoM)

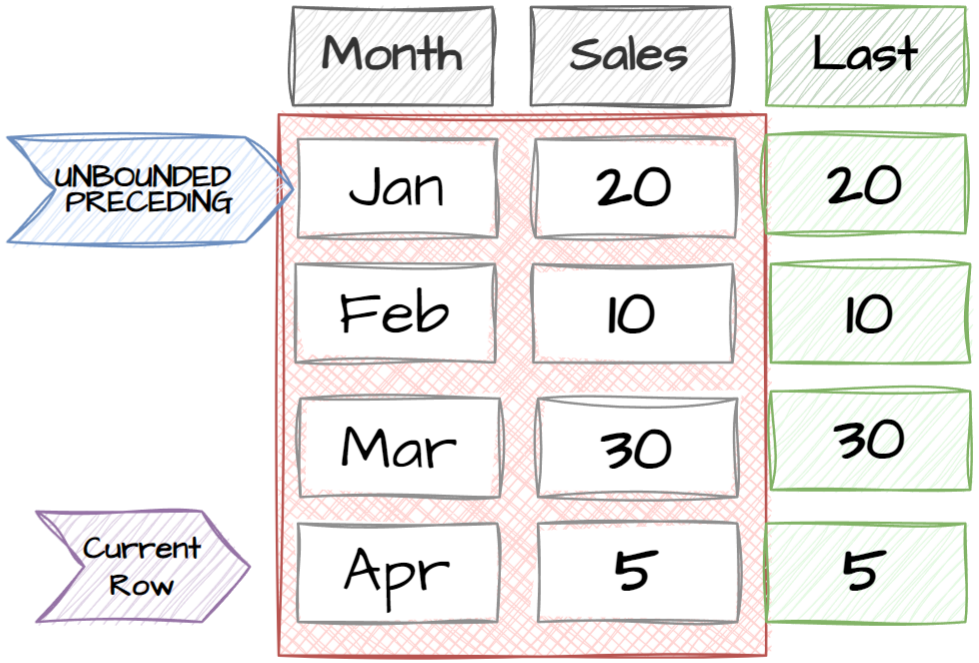
Analyze short-term trends and discover patterns in seasonality

FIRST & LAST


```
FIRST_VALUE(Sales) OVER (ORDER BY Month)
```




```
LAST_VALUE(Sales) OVER (ORDER BY Month)
```

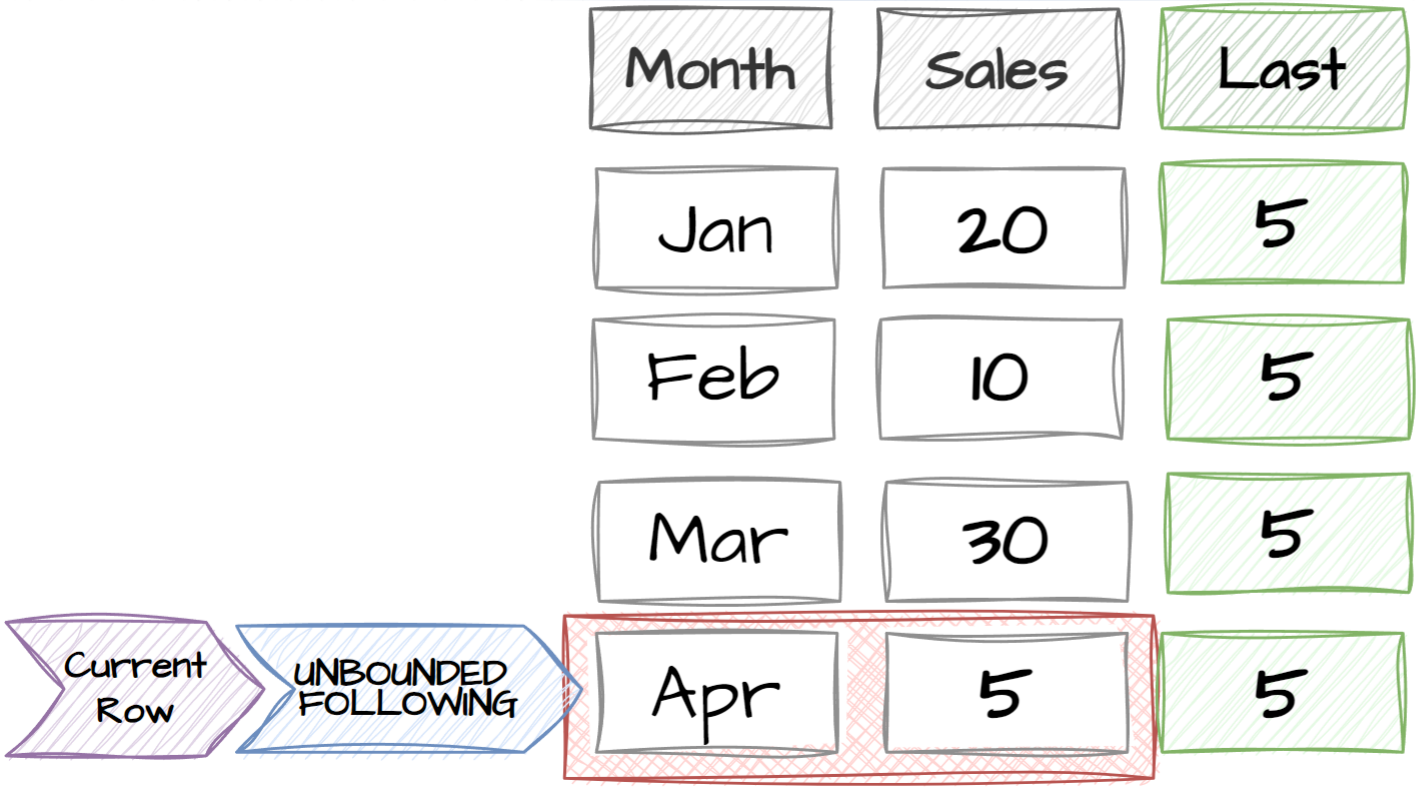


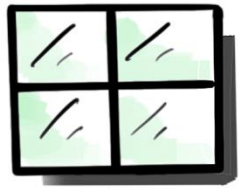
Default

```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

FIRST & LAST


```
LAST_VALUE(Sales) OVER (ORDER BY Month
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING)
```





SQL WINDOW USE CASES

- Top N Analysis
- Bottom N Analysis
- Identify & Remove Duplicates
- Assign Unique IDs & Pagination
- Data Segmentation
- Data Distribution Analysis
- Equalizing Load Processing
- Overall Analysis
- Total Per Groups Analysis
- Part-to-Whole Analysis
- Time Series Analysis : MoM & YoY
- Time Gaps Analysis : Customer Retention
- Comparision Analysis: Extreme → Highest, Lowest
- Outlier Detection
- Running Total
- Rolling Total
- Moving Average