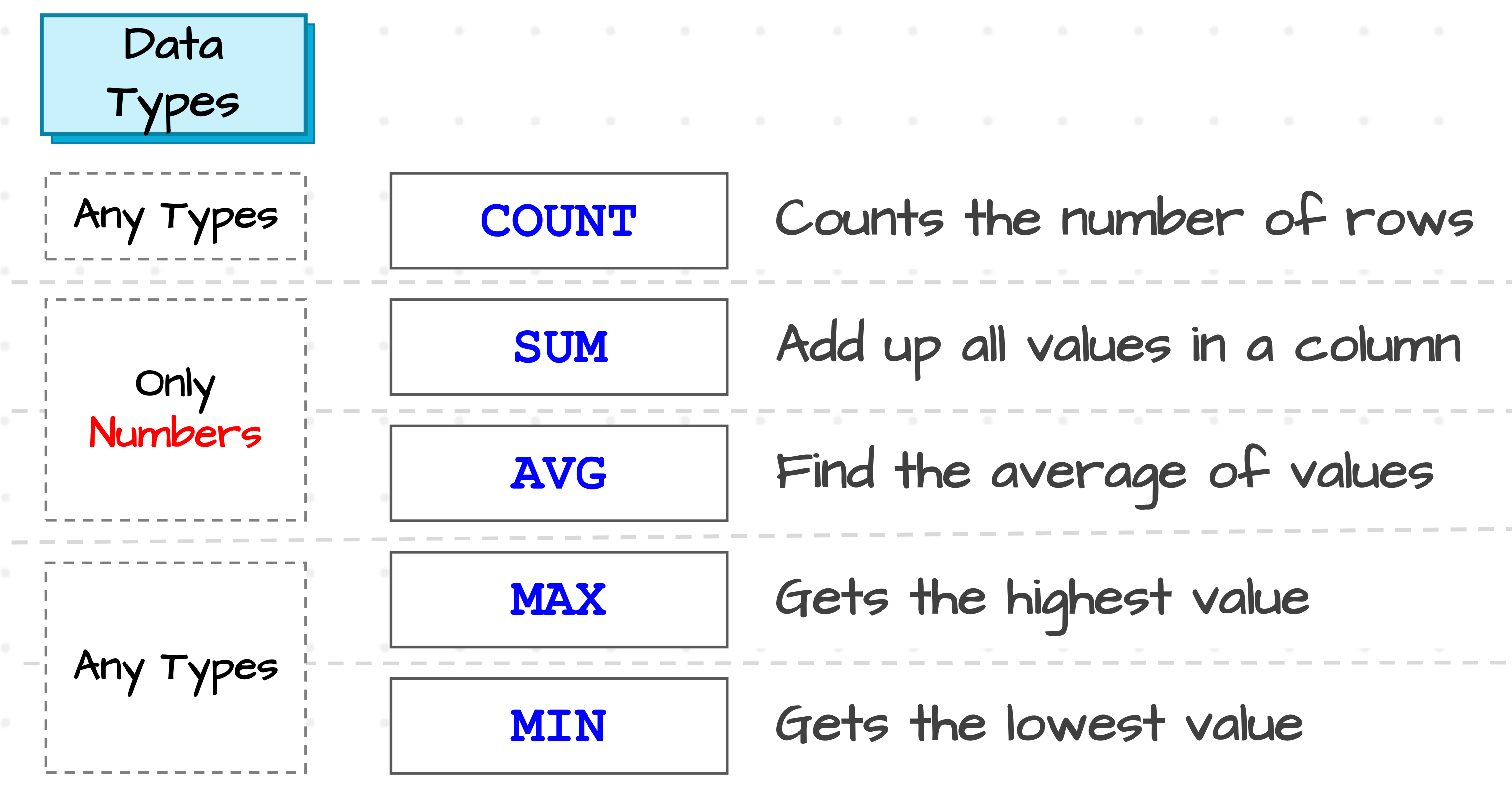
AGGREGATE

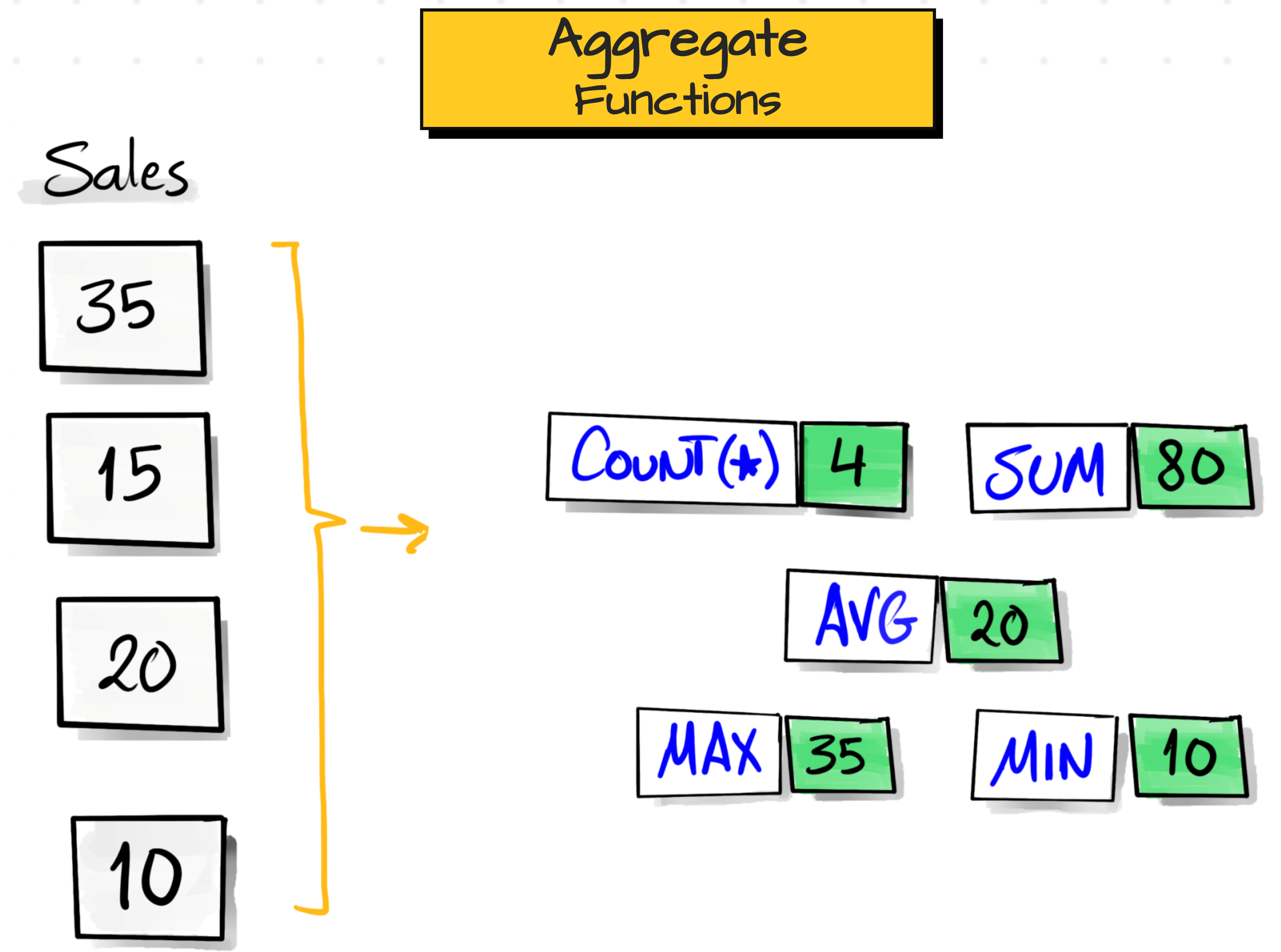
FUNCTIONS

Baraa Khatib Salkini

SQL Course | Aggregate Functions

Aggregate Functions




Aggregate Functions
Sales
35
15
20
10
COUNT(*)
4
SUM
80
AVG
20
MAX
35
MIN
10

WINDOW FUNCTIONS

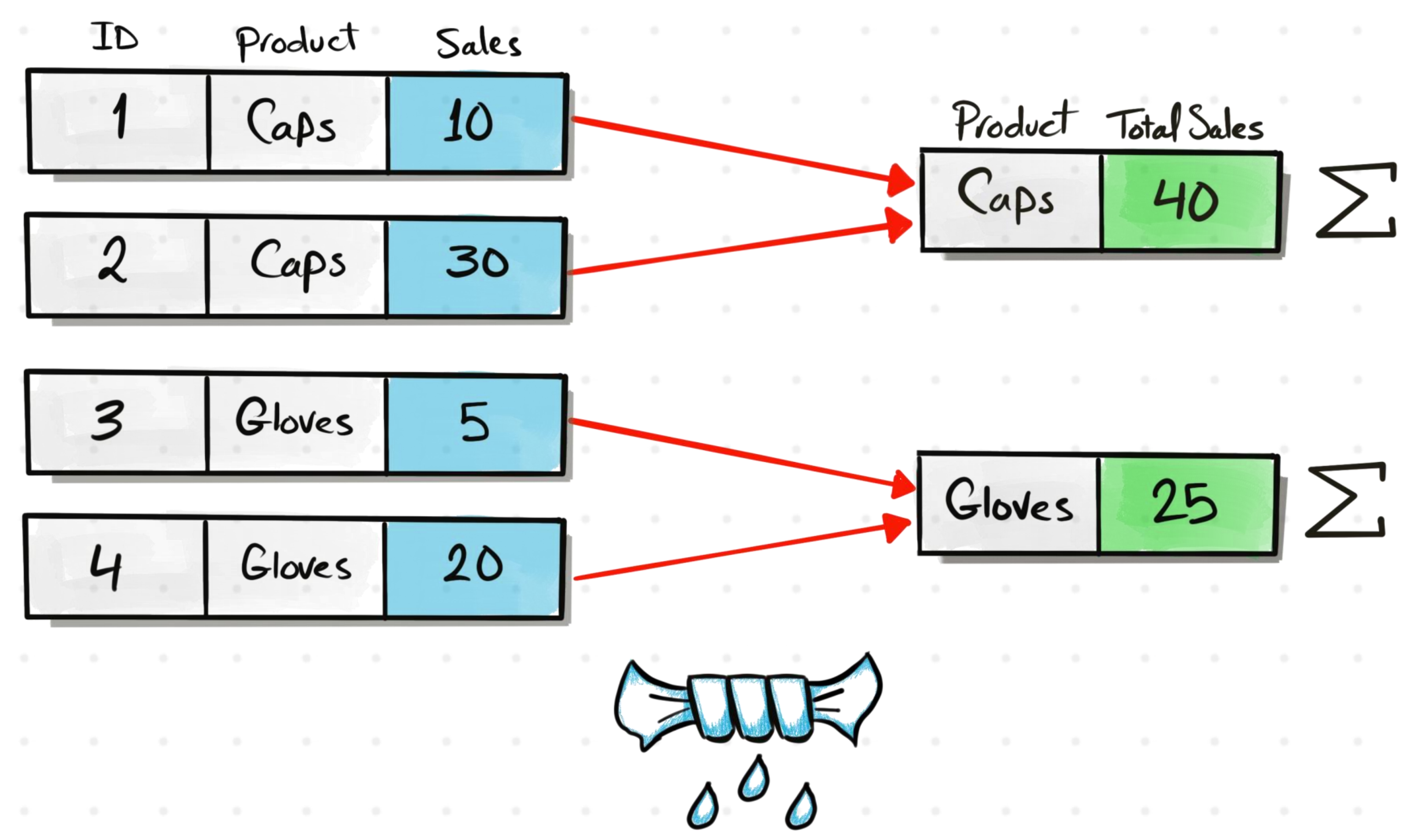
BASICS

Baraa Khatib Salkini
SQL Course | Window 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



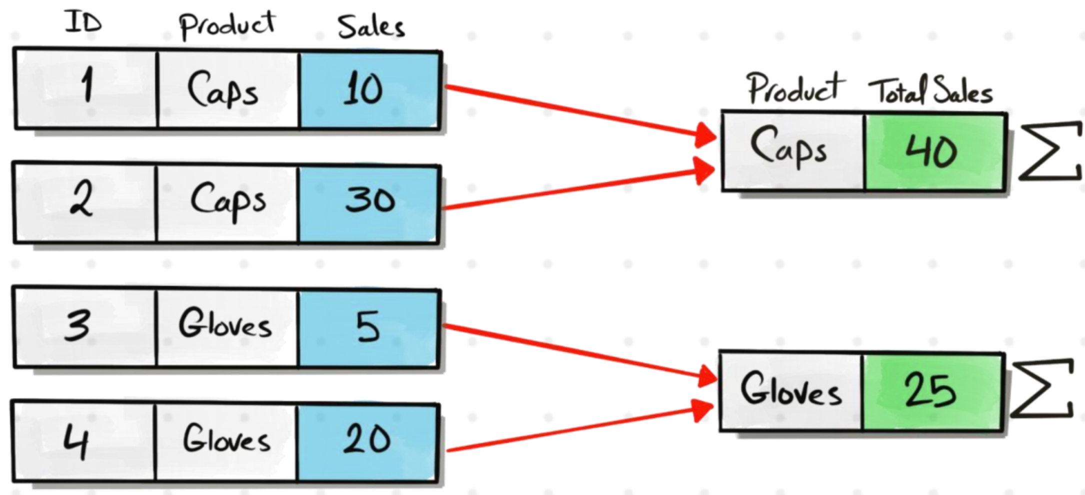
Compute aggregates but keep details of individual rows at the same time

ID	Product	Sales
1	Caps	10
2	Caps	30
3	Gloves	5
4	Gloves	20

ID	Product	Total Sales	
1	Caps	40	Σ
2	Caps	40	Σ
3	Gloves	25	Σ
4	Gloves	25	Σ

Row-Level Calculation

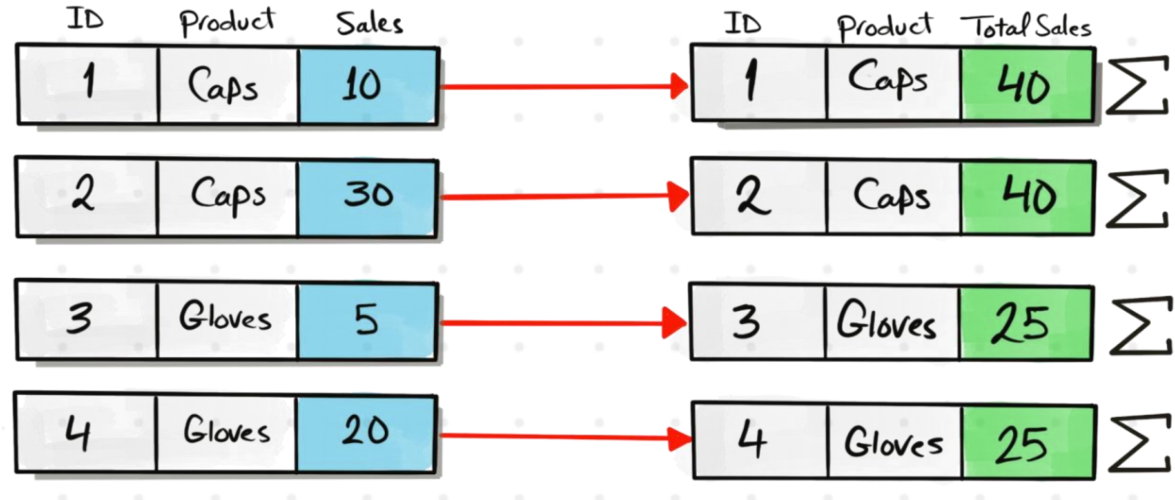
GROUP BY



It collapse groups of rows into a single row

(Group-Level-Calculations)

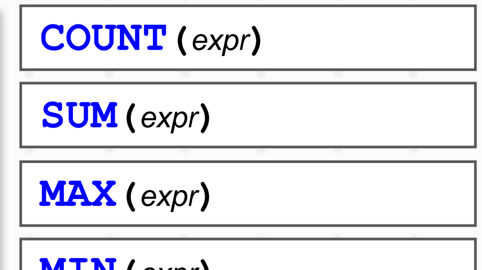
Window Functions



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*)

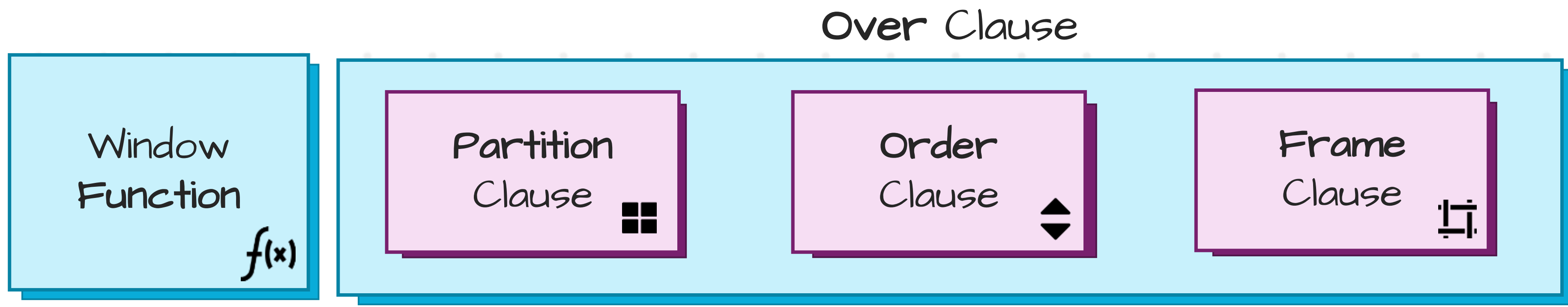
GROUP BY

Simple Data Analysis
(Aggregations)

WINDOW

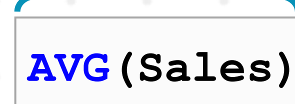
Advanced Data Analysis
(Aggregations + Details)

Window
Syntax


Over Clause
Window
Function
Partition
Clause
Order
Clause
Frame
Clause

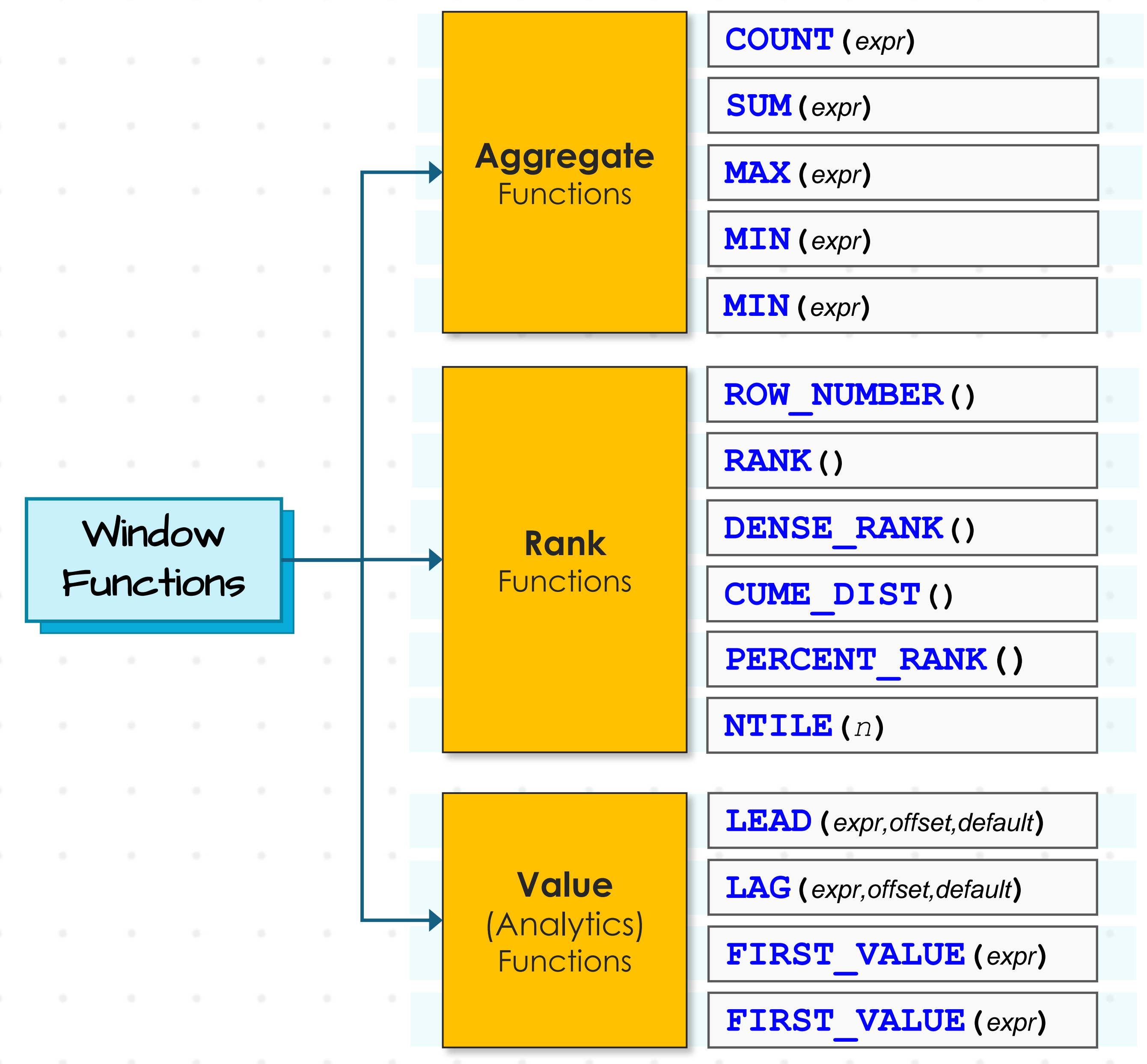
```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

Calculation used on the Window




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

f(x)
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)

Calculation used on the Window

f(x) Window Function

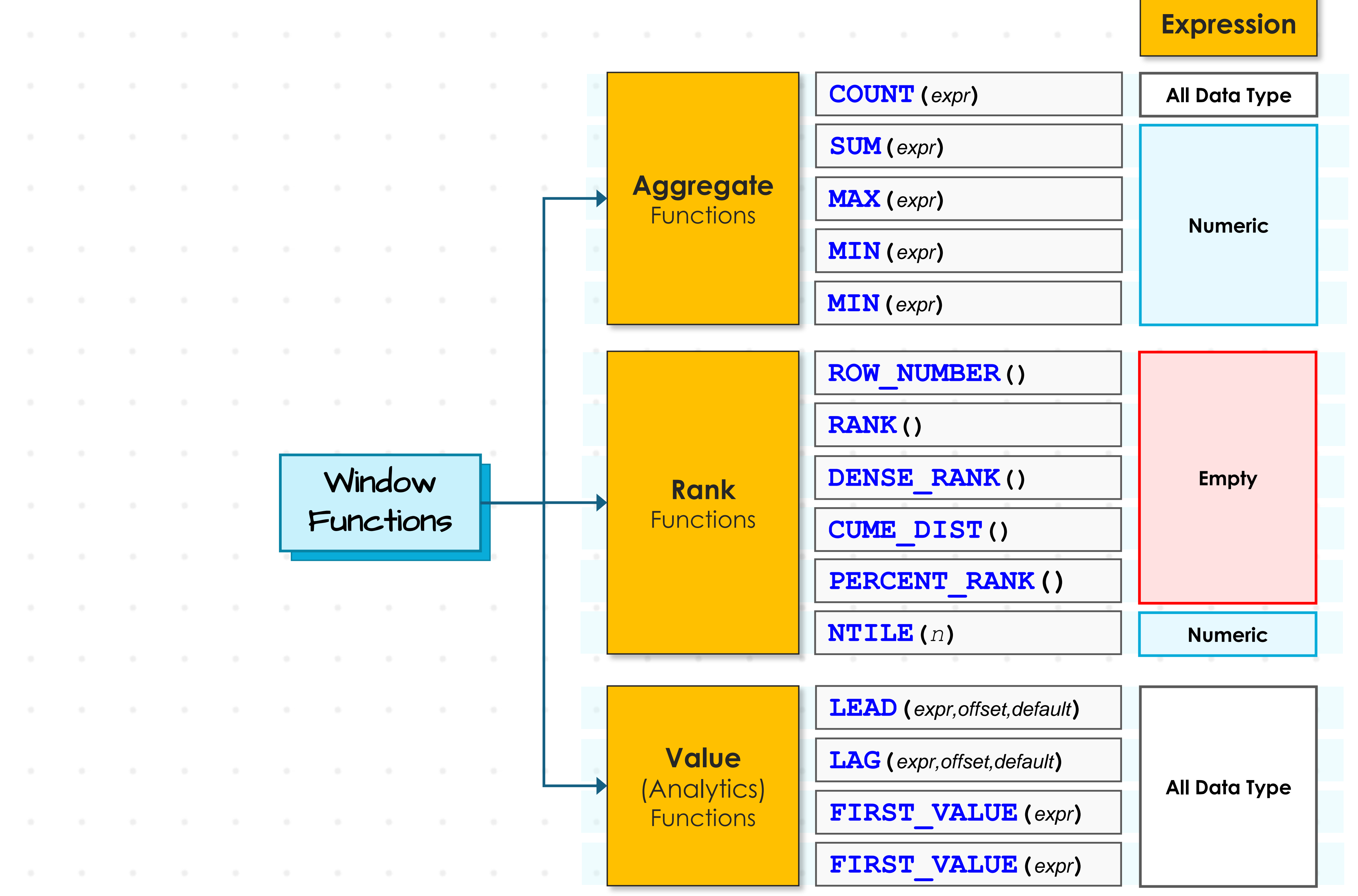
```
AVG(Sales) OVER ( PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING )
```

Function Expression

FUNCTION EXPRESSION

Arguments you pass to a function

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)
All Data Type

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

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

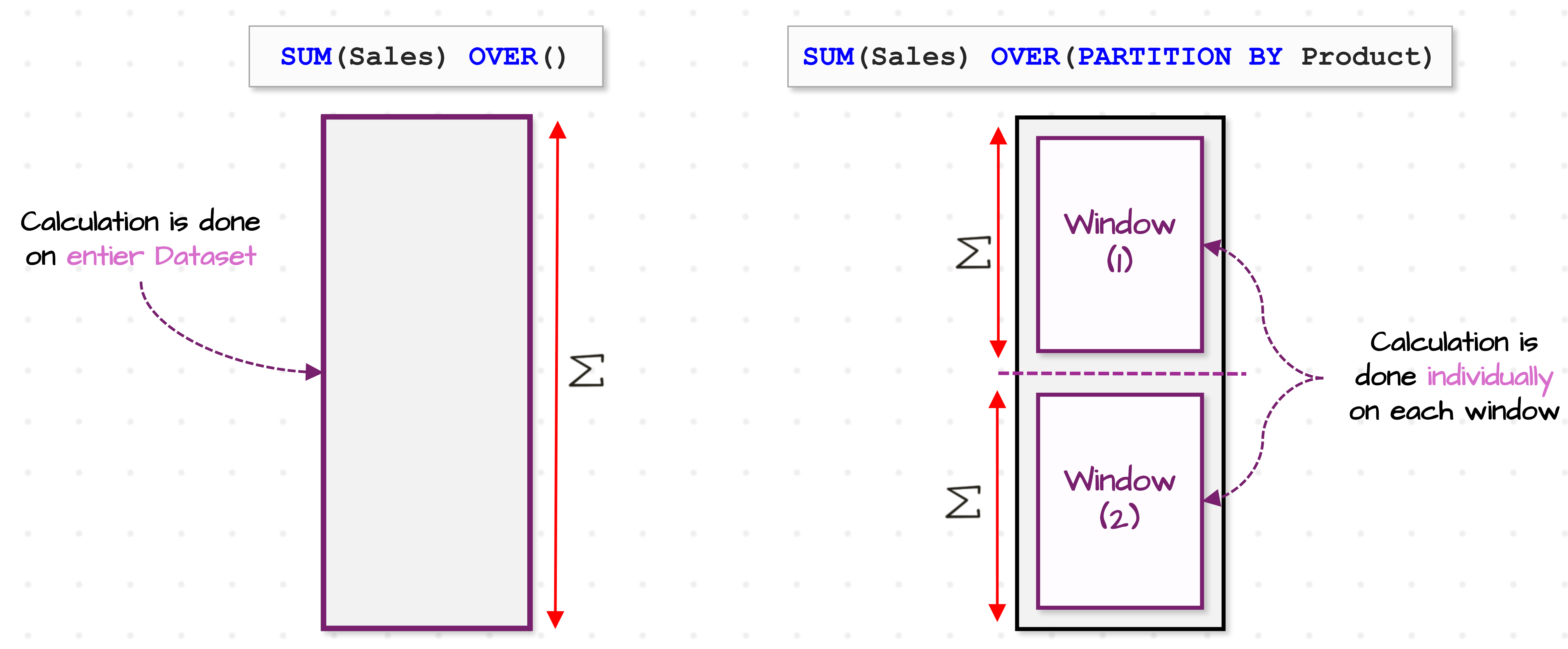
Partition Clause

Divides the dataset into windows (Partitions)

PARTITION BY
Divides the result set into partitions (Windows)

Partition By

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



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

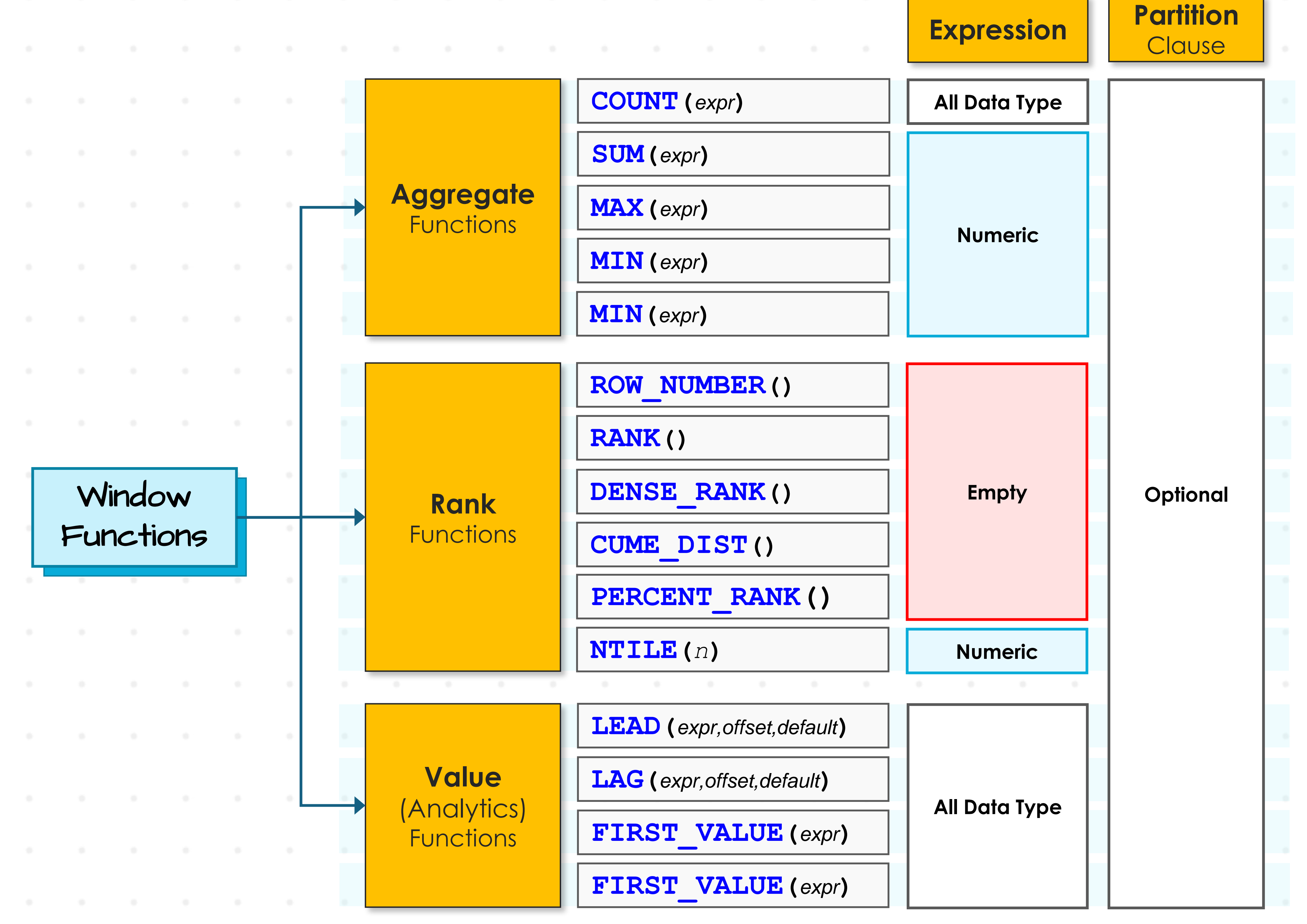
Total sales for each Product

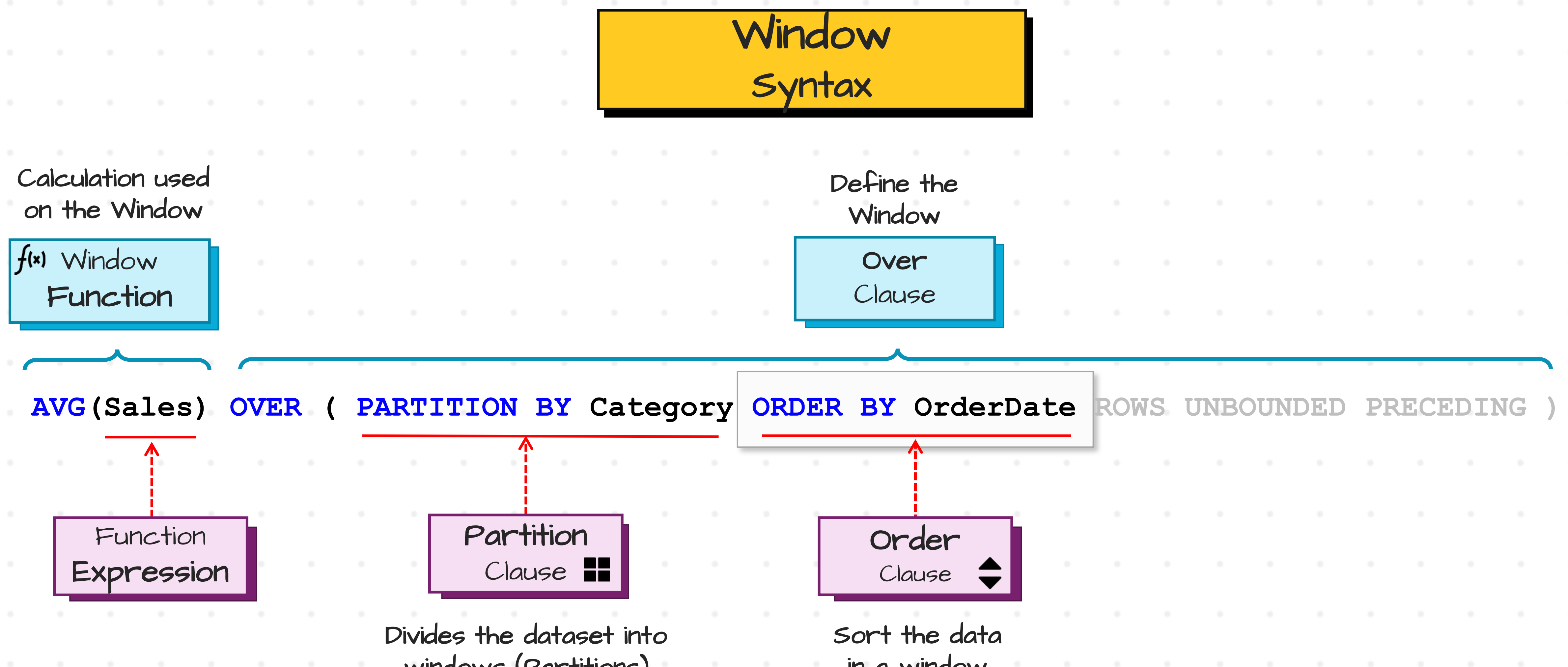
```
SUM(Sales) OVER (PARTITION BY Product)
```

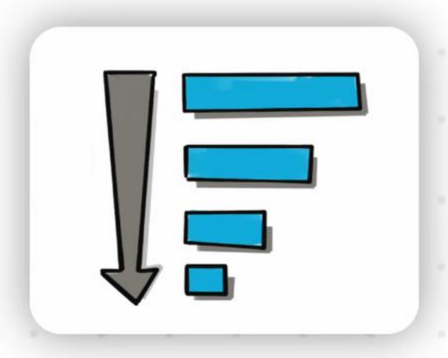
Partition By Combined-Columns

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)
All Data Type
Optional

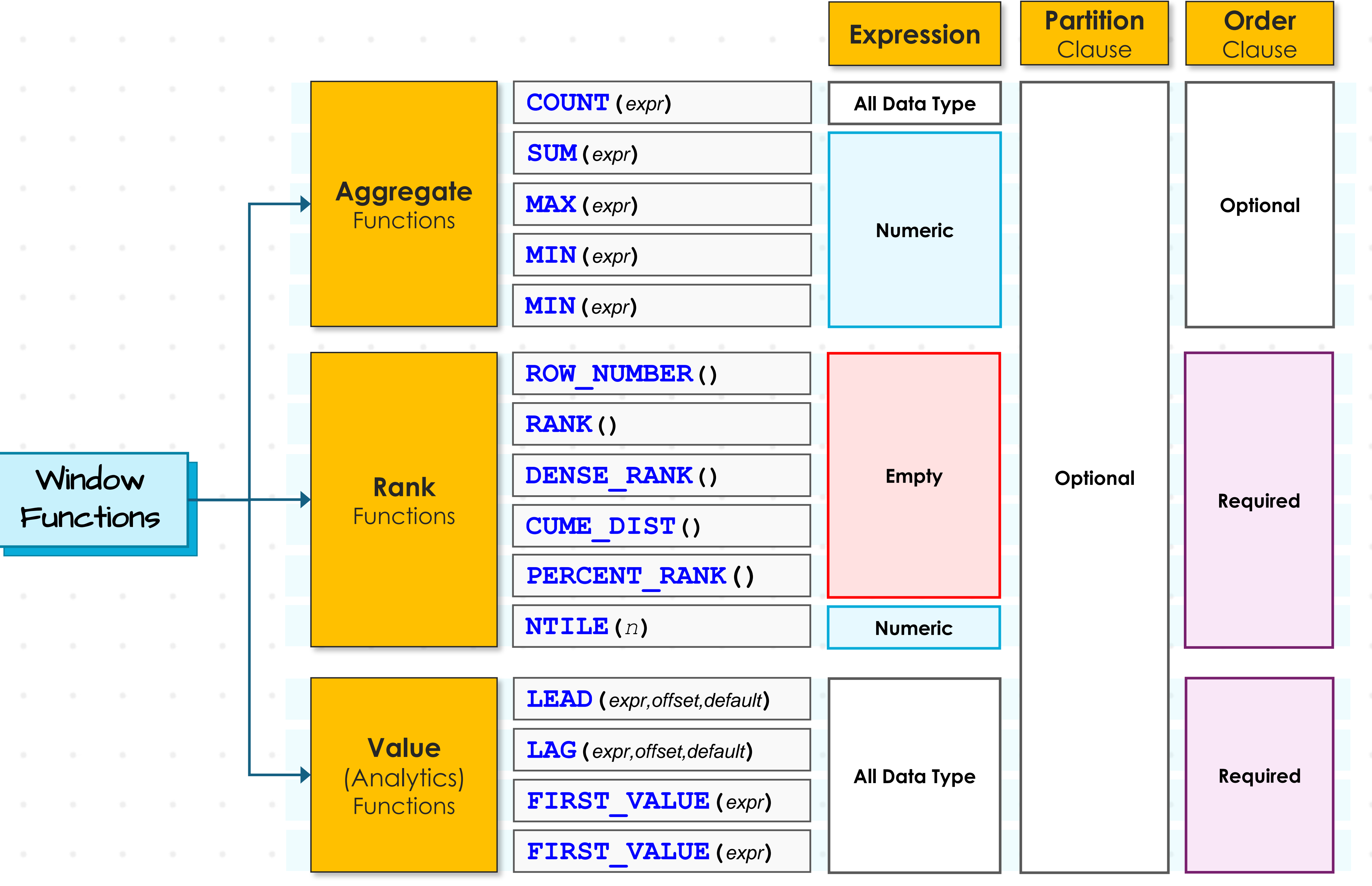

Window
Syntax
Calculation used
on the Window
f(x) Window
Function
Define the
Window
Over
Clause
AVG(Sales) OVER (PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING)
Function
Expression
Partition
Clause
Order
Clause
Divides the dataset into
windows (Partitions)
Sort the data
in a window



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)
All Data Type
Required

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

Partition Clause

Divides the dataset into windows (Partitions)

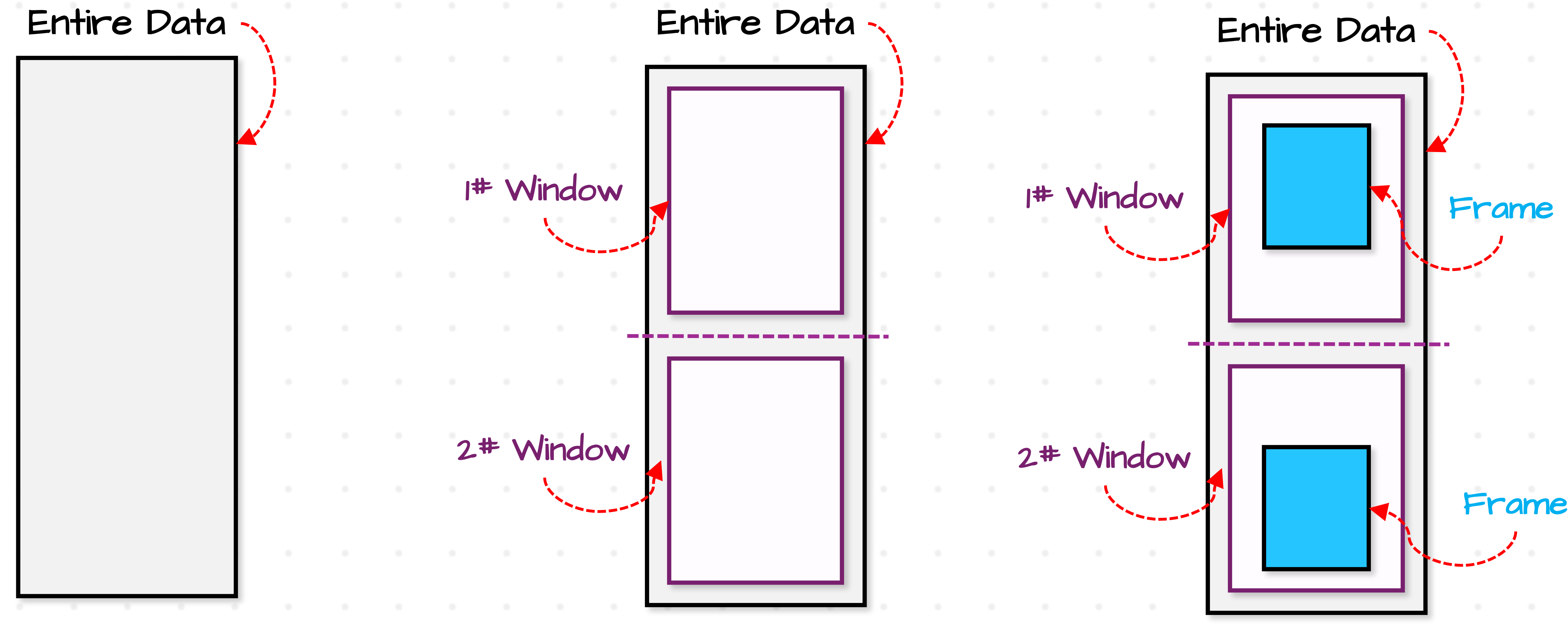
Order Clause

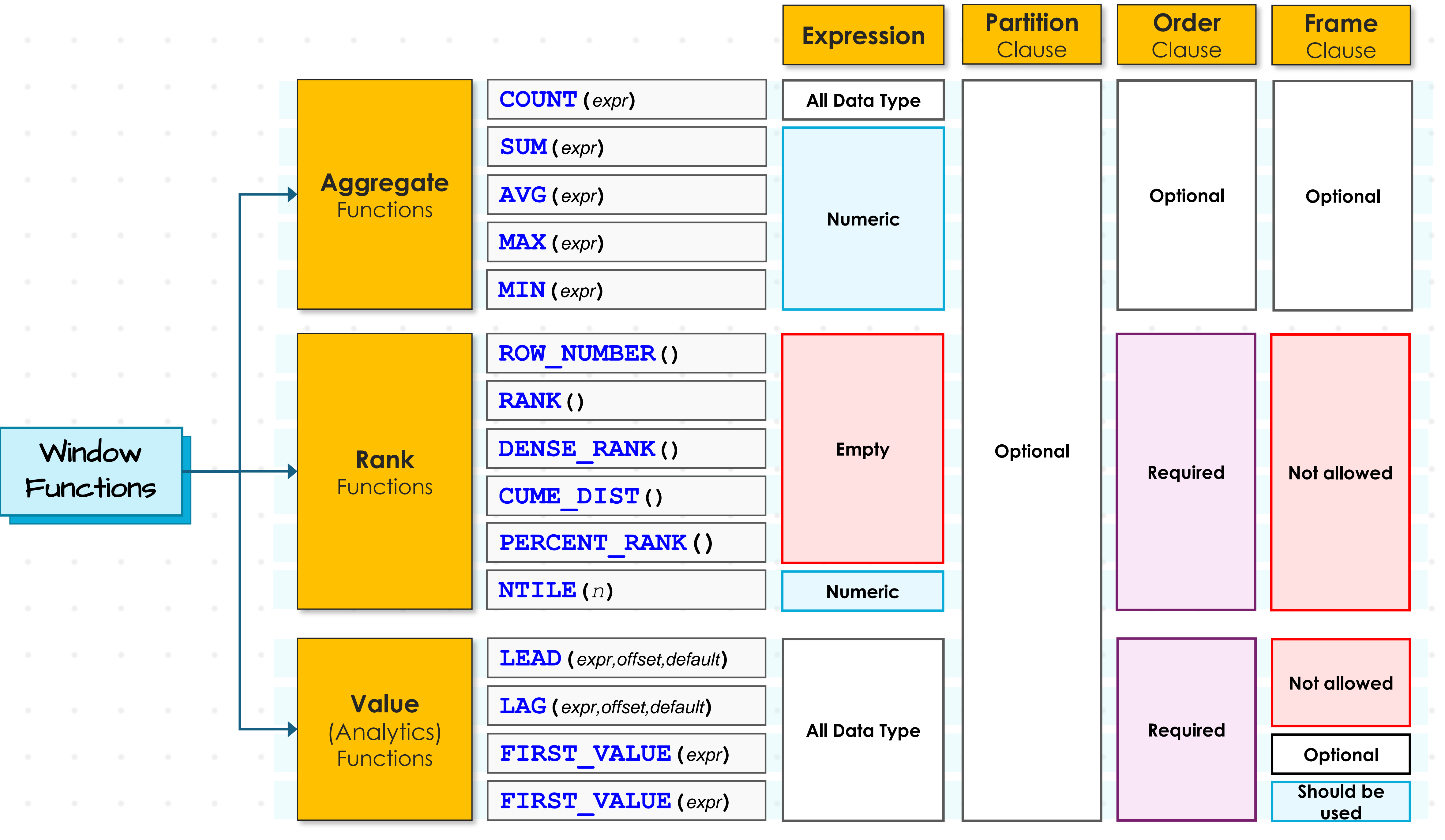
Sort the data in a window

Frame Clause

Define a subset of rows in a window

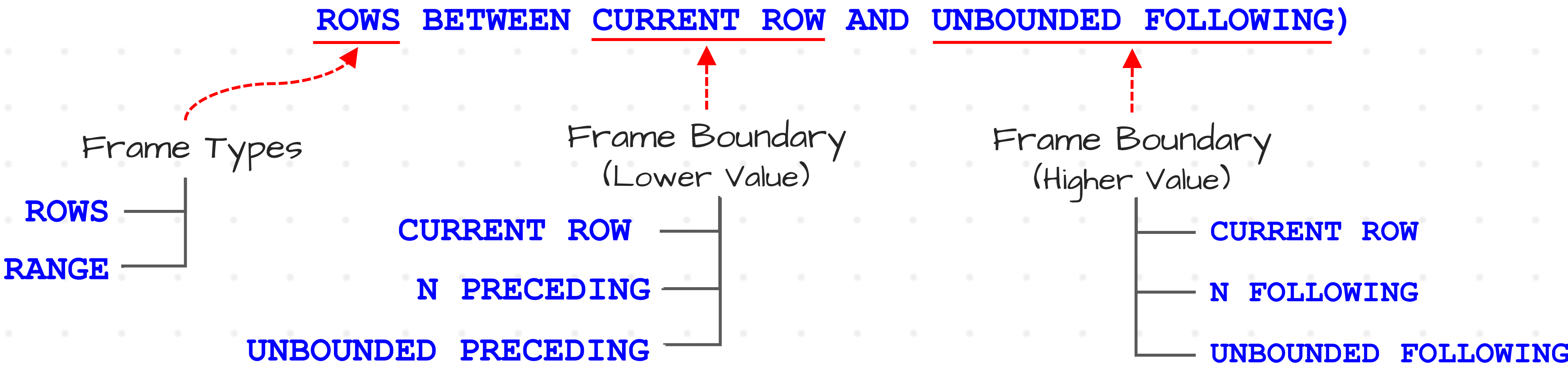
Frame Clause





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)
All Data Type
Required
Not allowed
Optional
Should be used

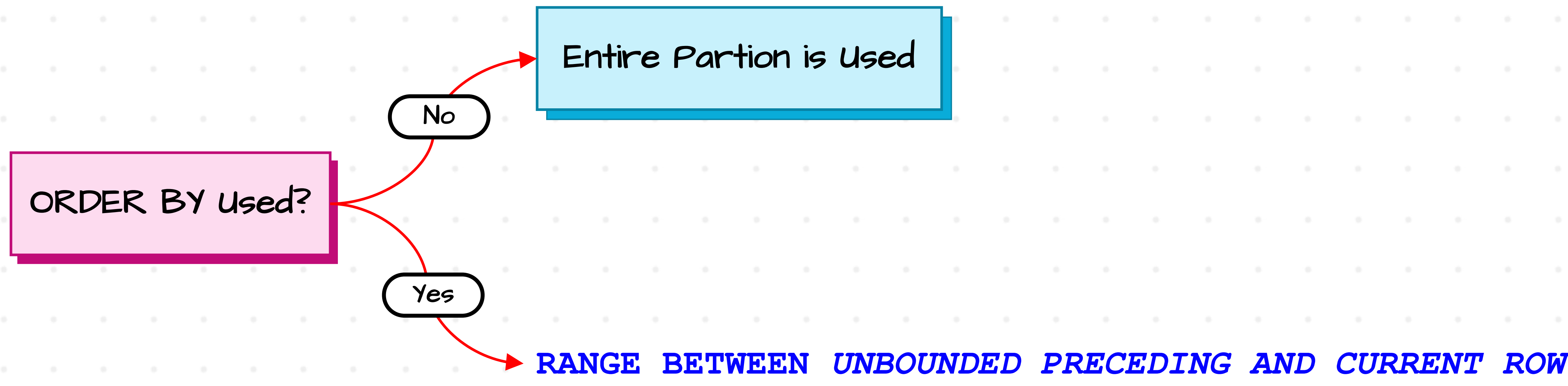
Frame



Rules

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

Frame



COMPACT FRAME

For only PRECEDING, the CURRENT ROW can be skipped

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

#1 RULE

Window functions can be used ONLY in SELECT and ORDER BY Clauses

#2 RULE

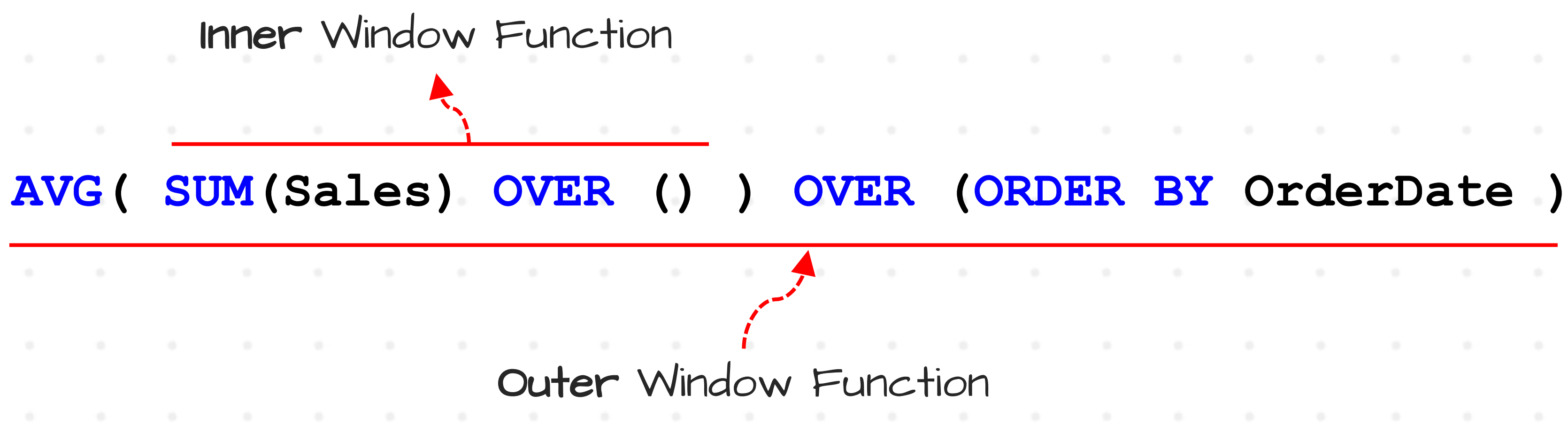
Nesting Window Functions is not allowed !

#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



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 Group By

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

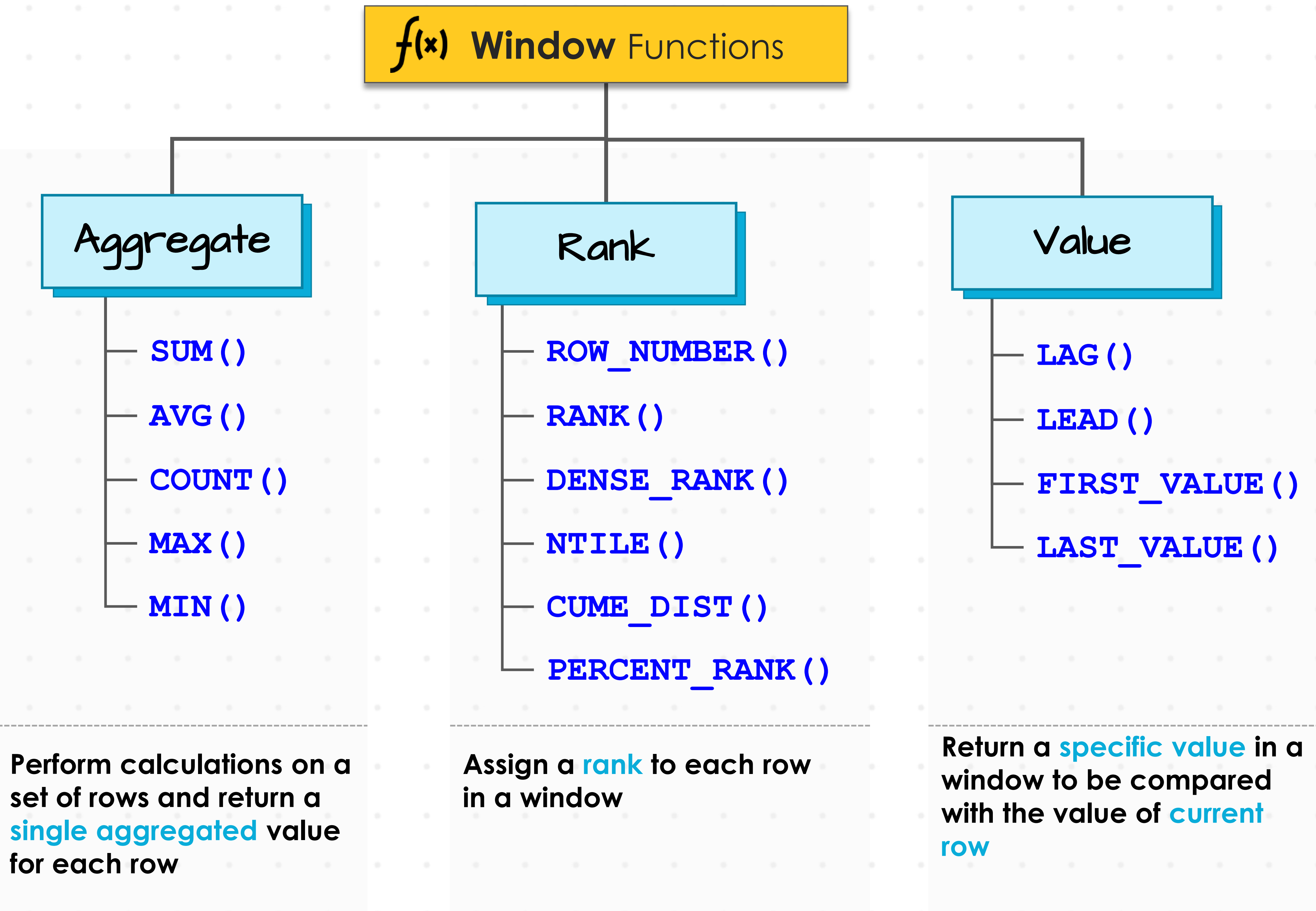
Components

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()
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

WINDOW AGGREGATE

FUNCTIONS

Baraa Khatib Salkini

SQL Course | Window Aggregate Functions

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

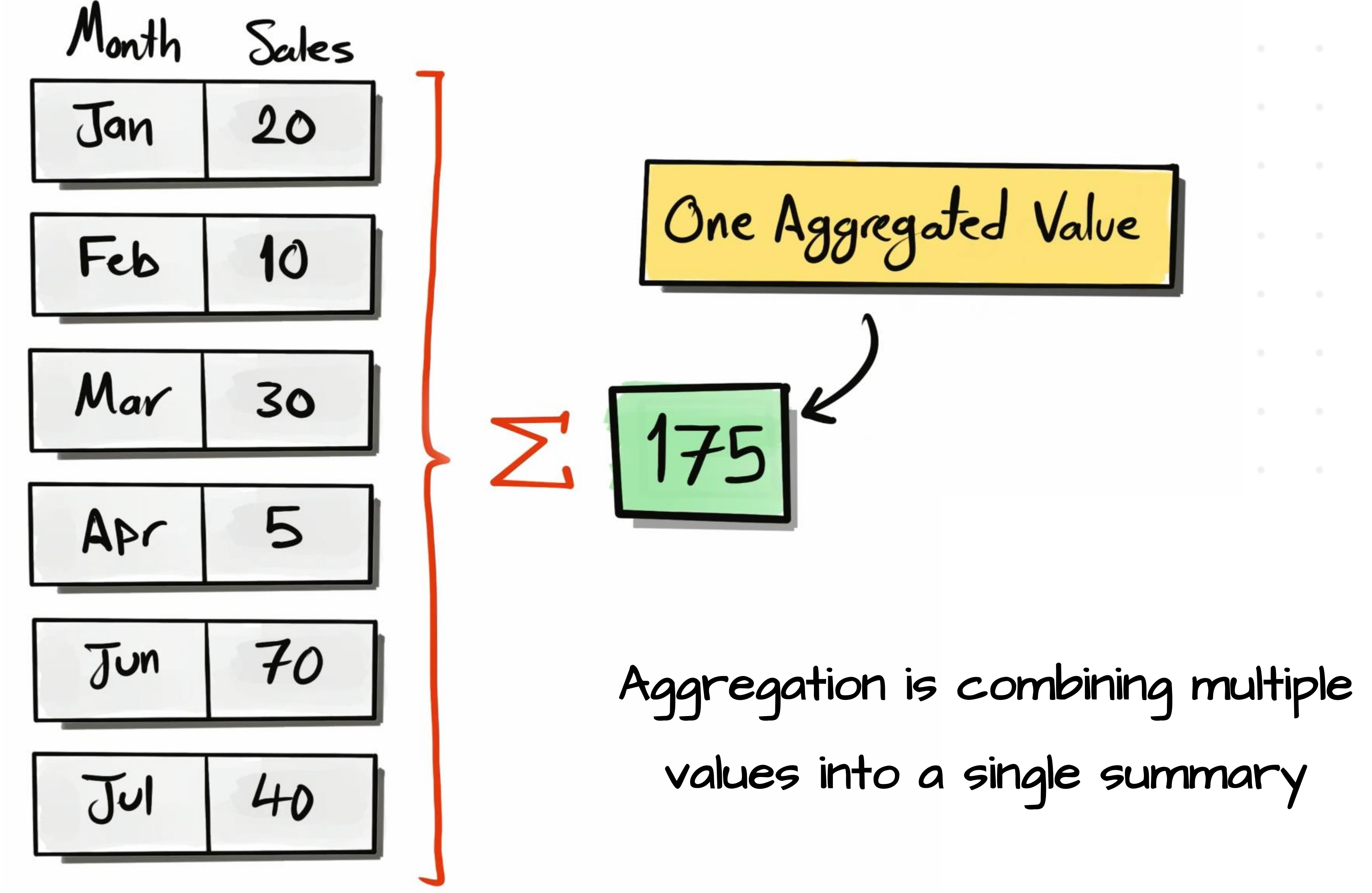
- ROW_NUMBER()
- RANK()
- DENSE_RANK()
- NTILE()
- CUME_DIST()
- PERCENT_RANK()

Assign a rank to each row in a window

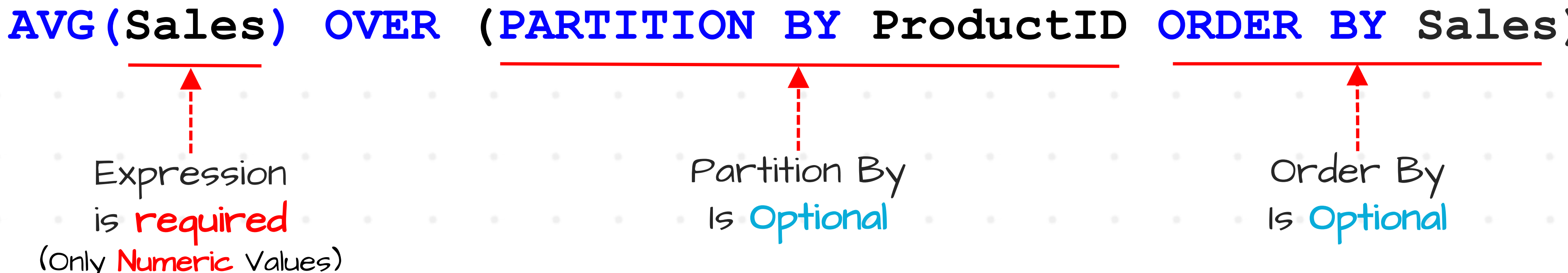
Value

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

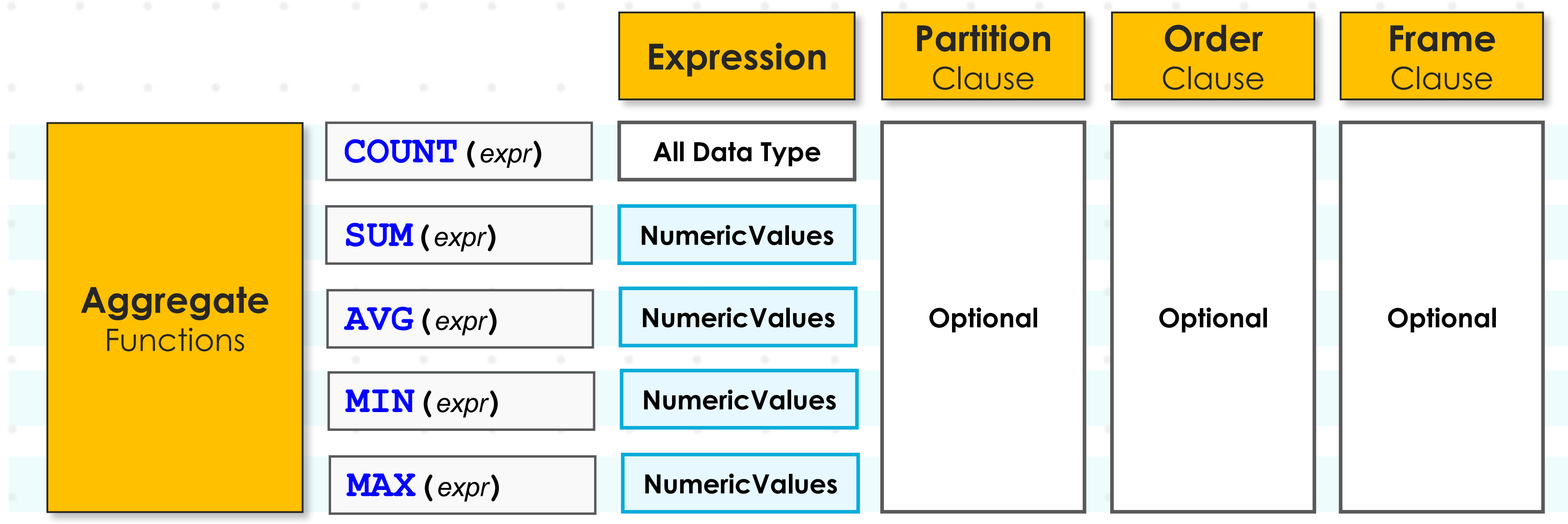
Return a specific value in a window to be compared with the value of current row


Month
Sales
Jan
20
Feb
10
Mar
30
Apr
5
Jun
70
Jul
40
Σ
175
One Aggregated Value
Aggregation is combining multiple values into a single summary

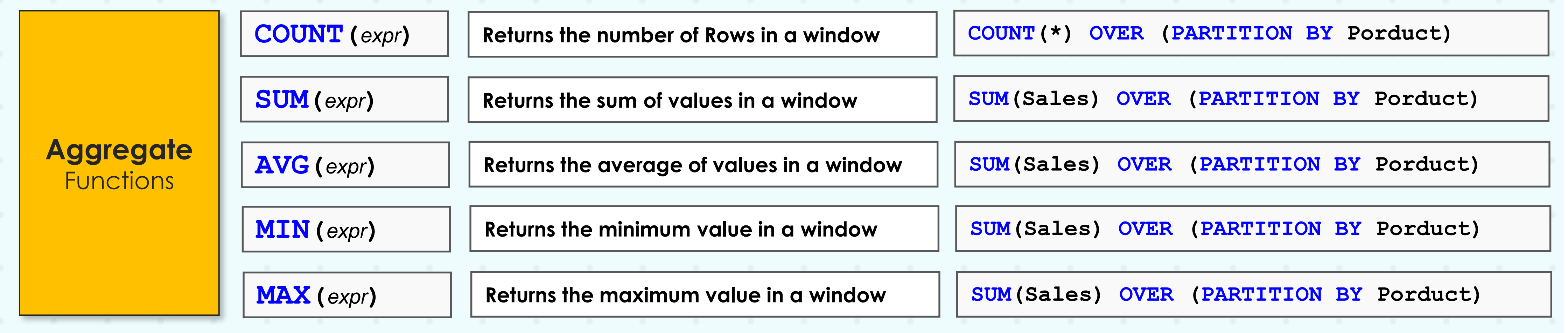
Aggregate
Functions


AVG(Sales) OVER (PARTITION BY ProductID ORDER BY Sales)
Expression
is required
(Only Numeric Values)
Partition By
Is Optional
Order By
Is Optional

Aggregate
Functions


Expression
Partition Clause
Order Clause
Frame Clause
Aggregate Functions
COUNT (expr)
All Data Type
SUM (expr)
Numeric Values
AVG (expr)
Numeric Values
MIN (expr)
Numeric Values
MAX (expr)
Numeric Values
Optional
Optional
Optional

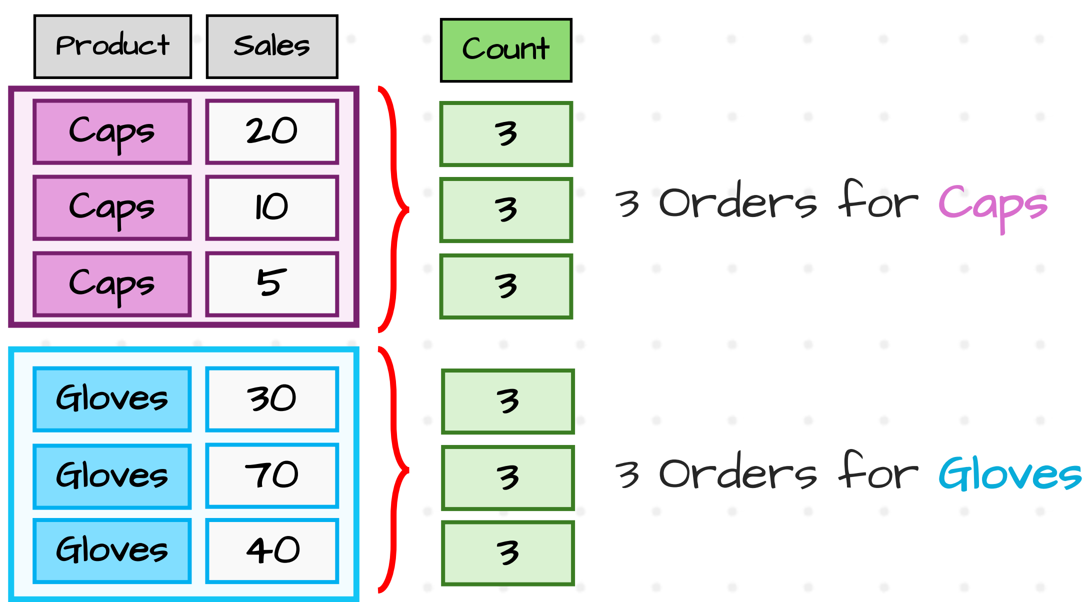
Aggregate
Functions


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)

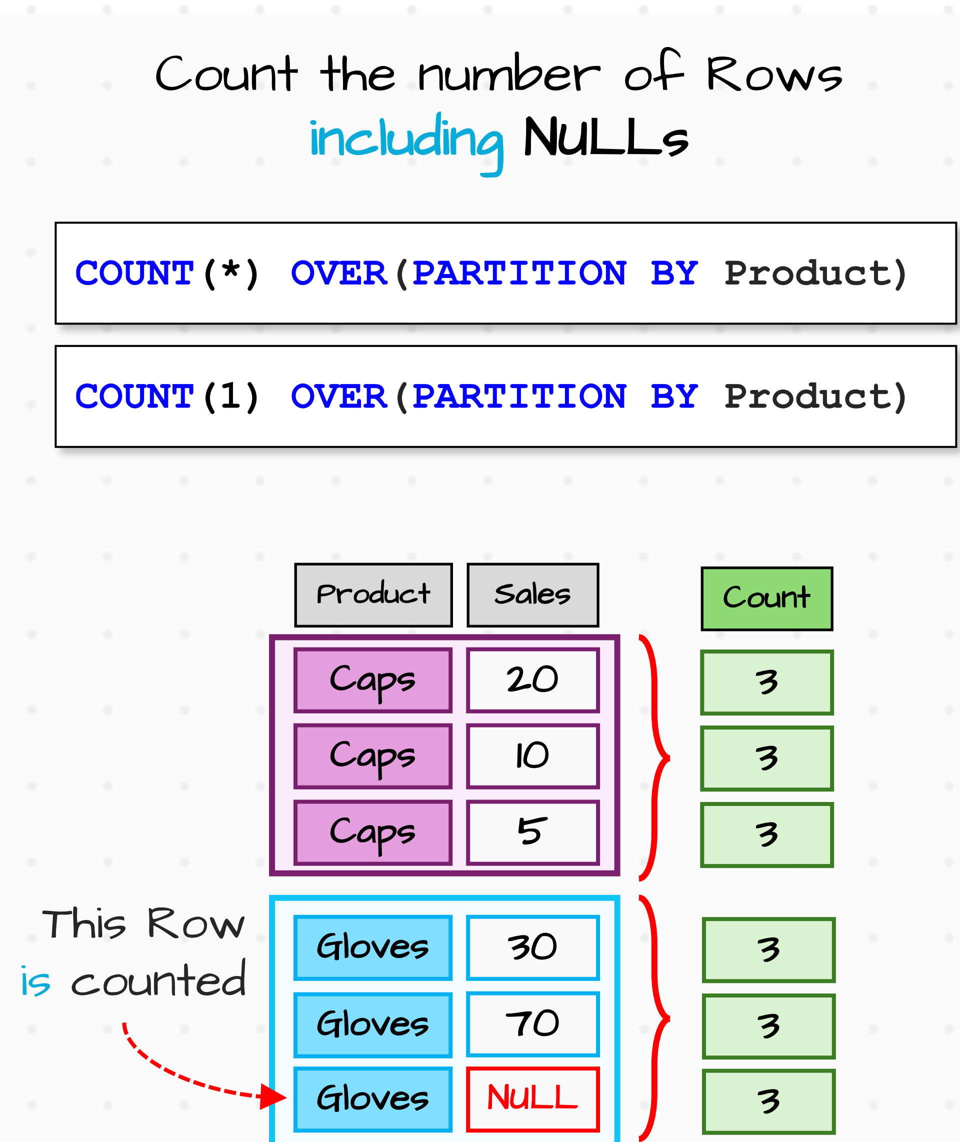
COUNT Function

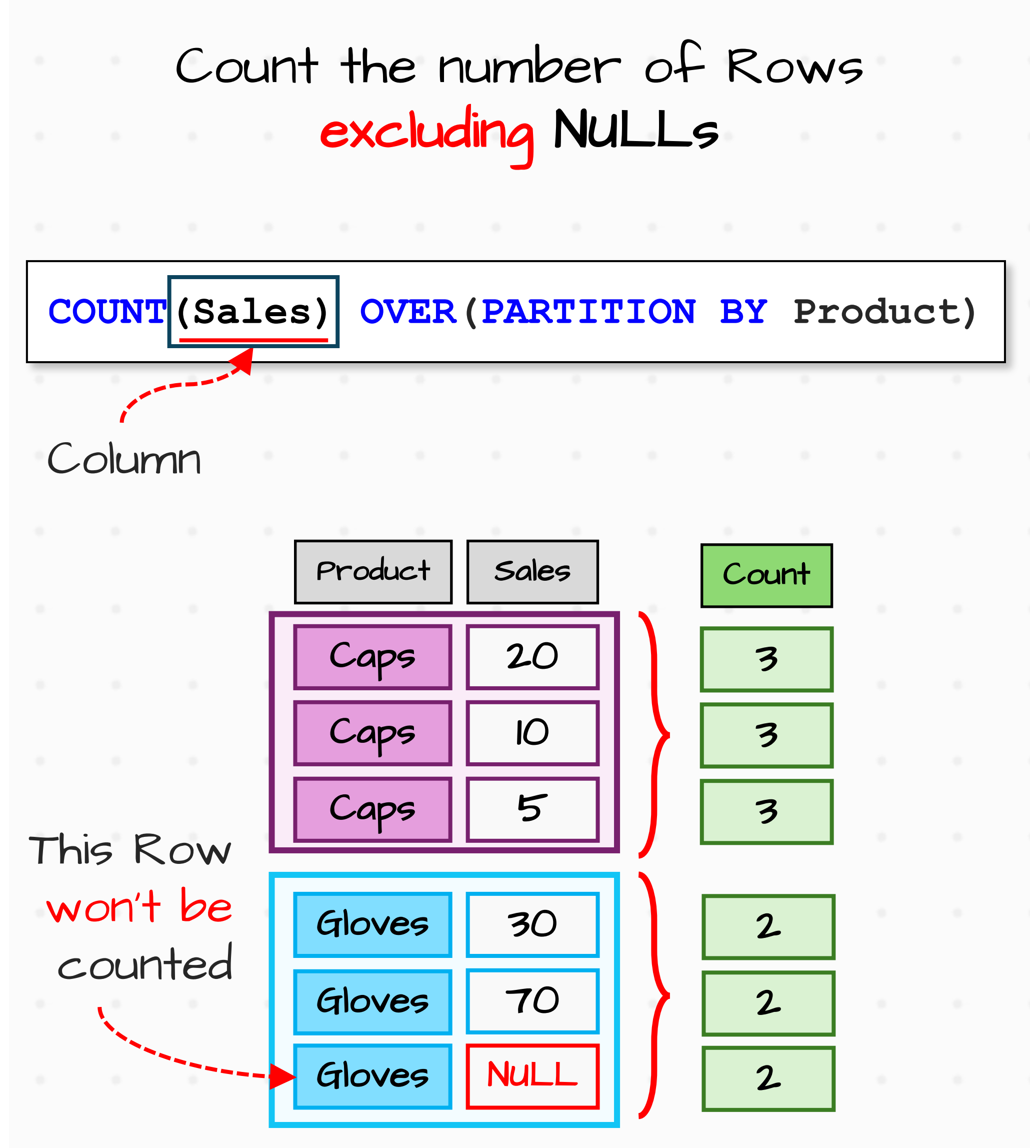
Returns the number of Rows in a window

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

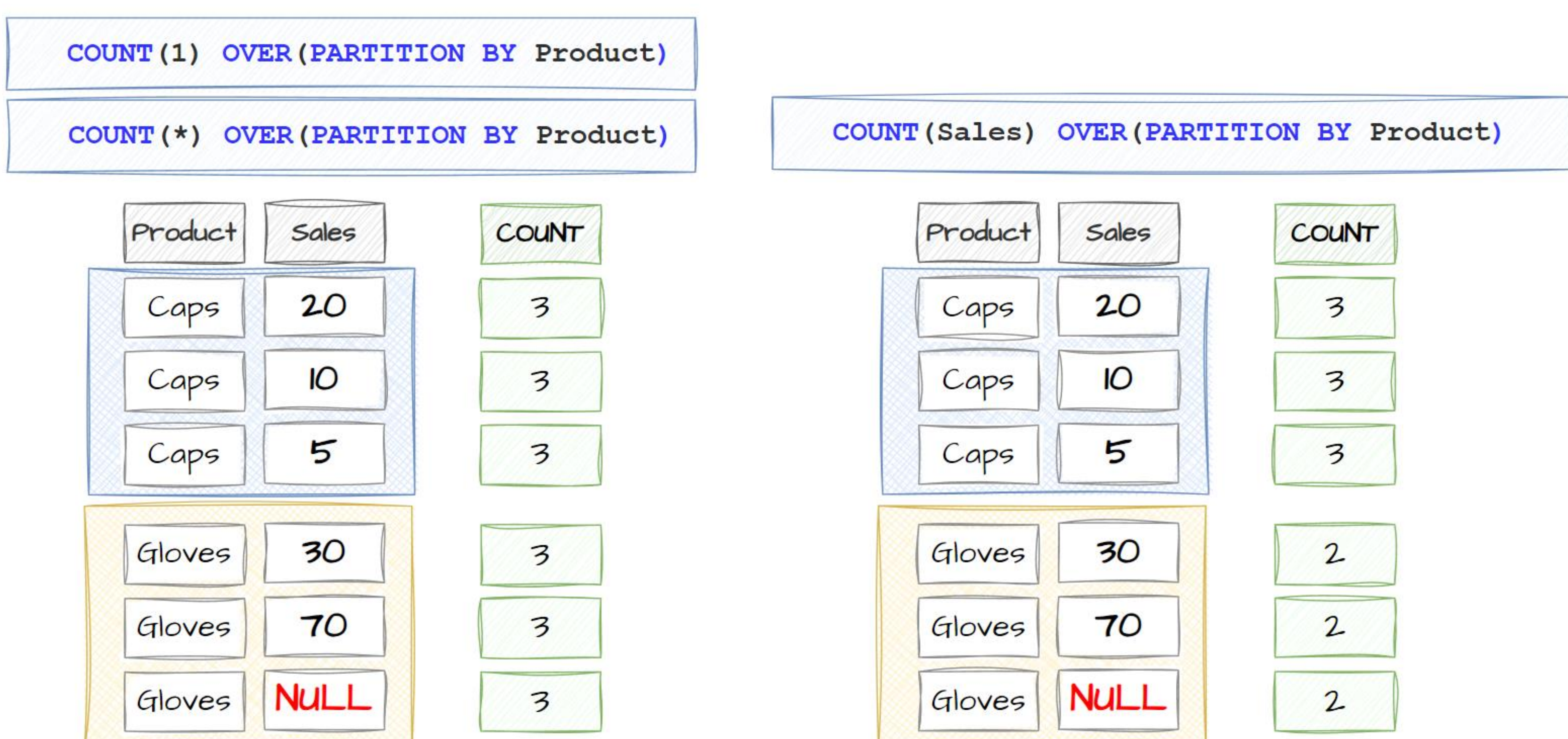


COUNT Function



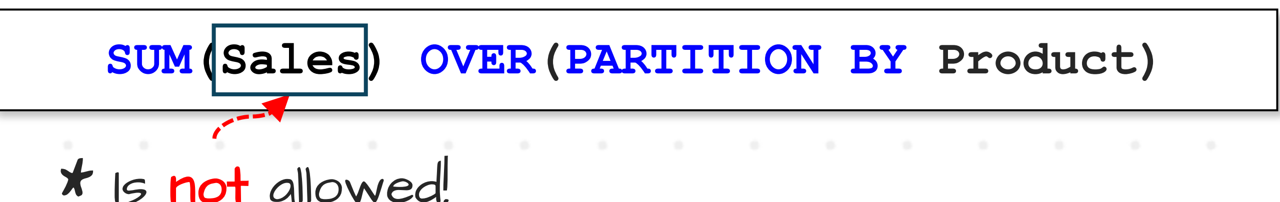


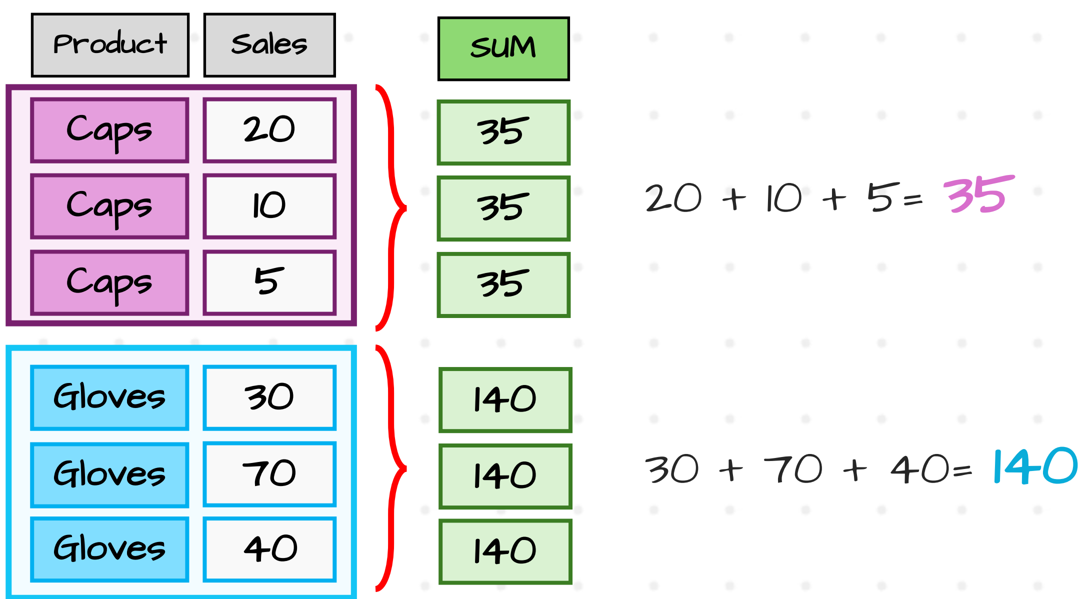
COUNT Function



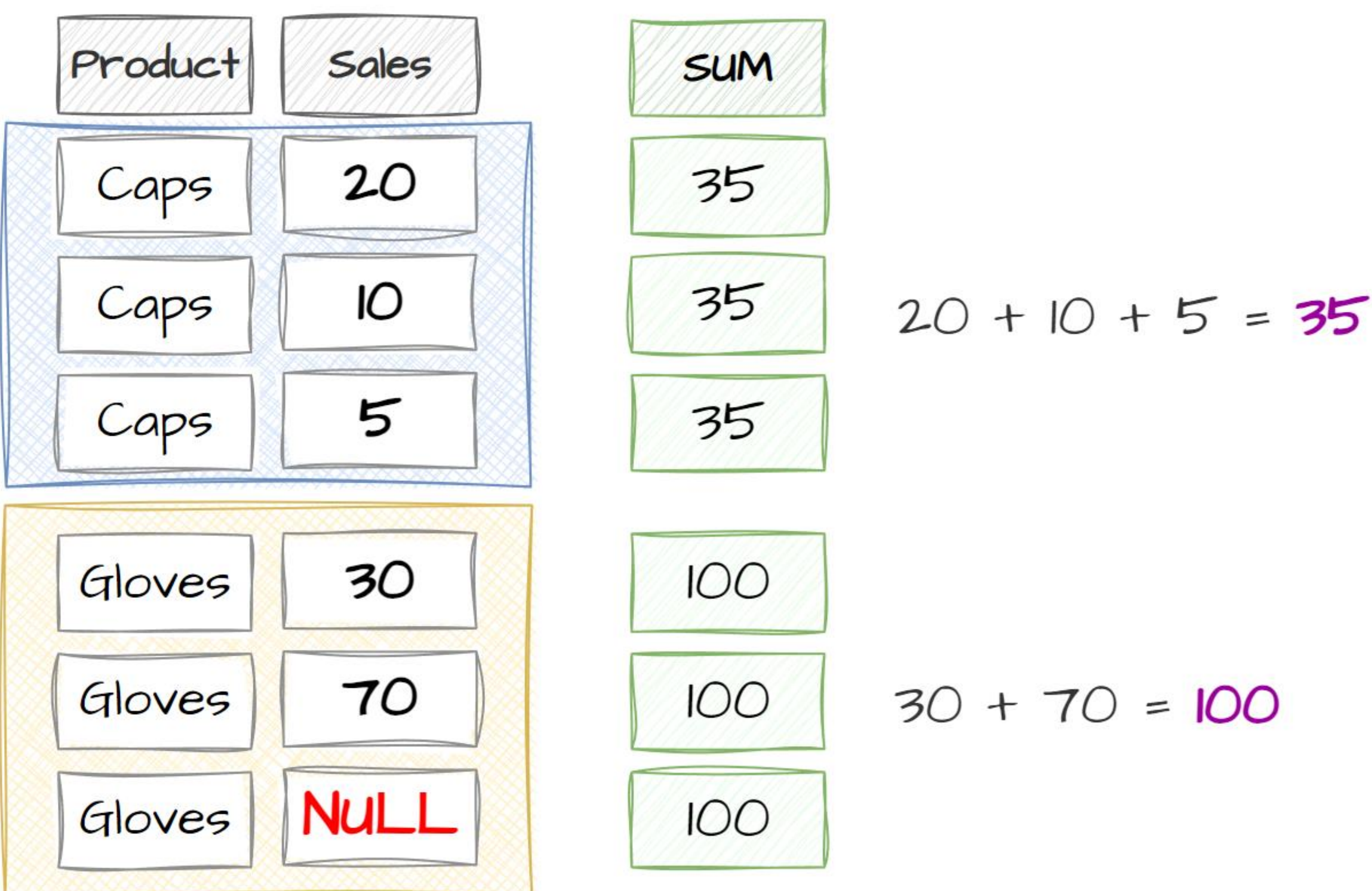
SUM Function

Returns the sum of values in a window



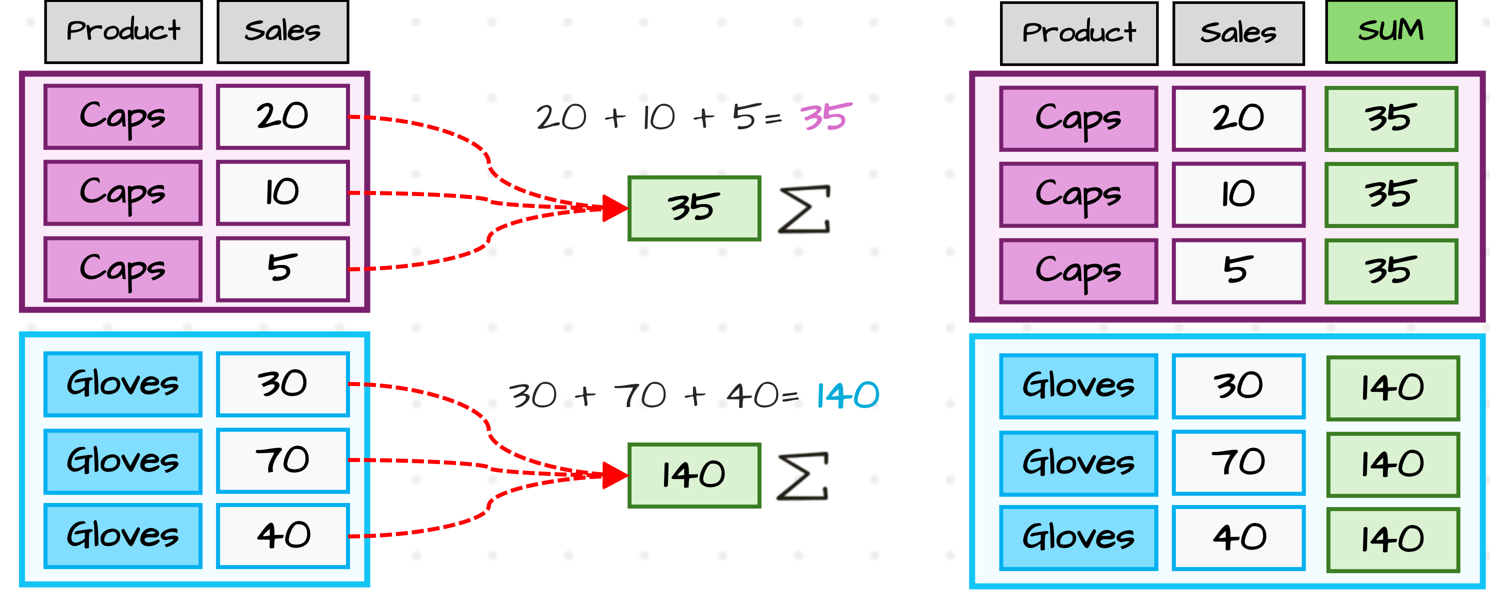


SUM Function



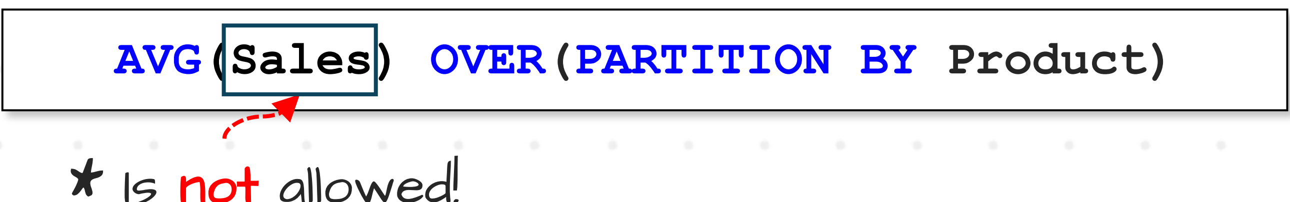
COUNT Function

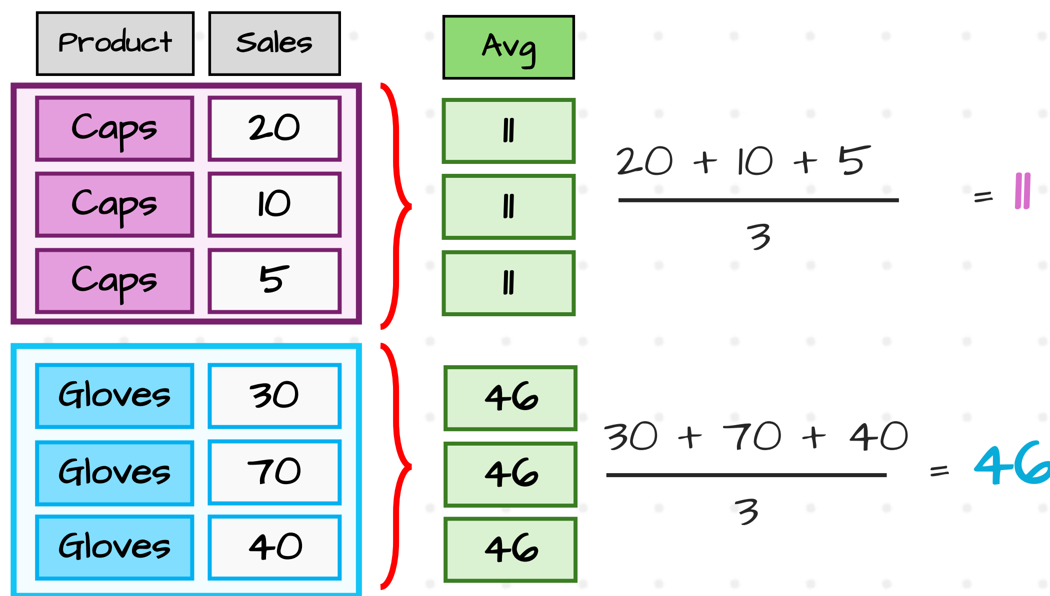
Perform calculations on a set of rows and return a single aggregated value for each row



AVG Function

Returns the average of values in a window

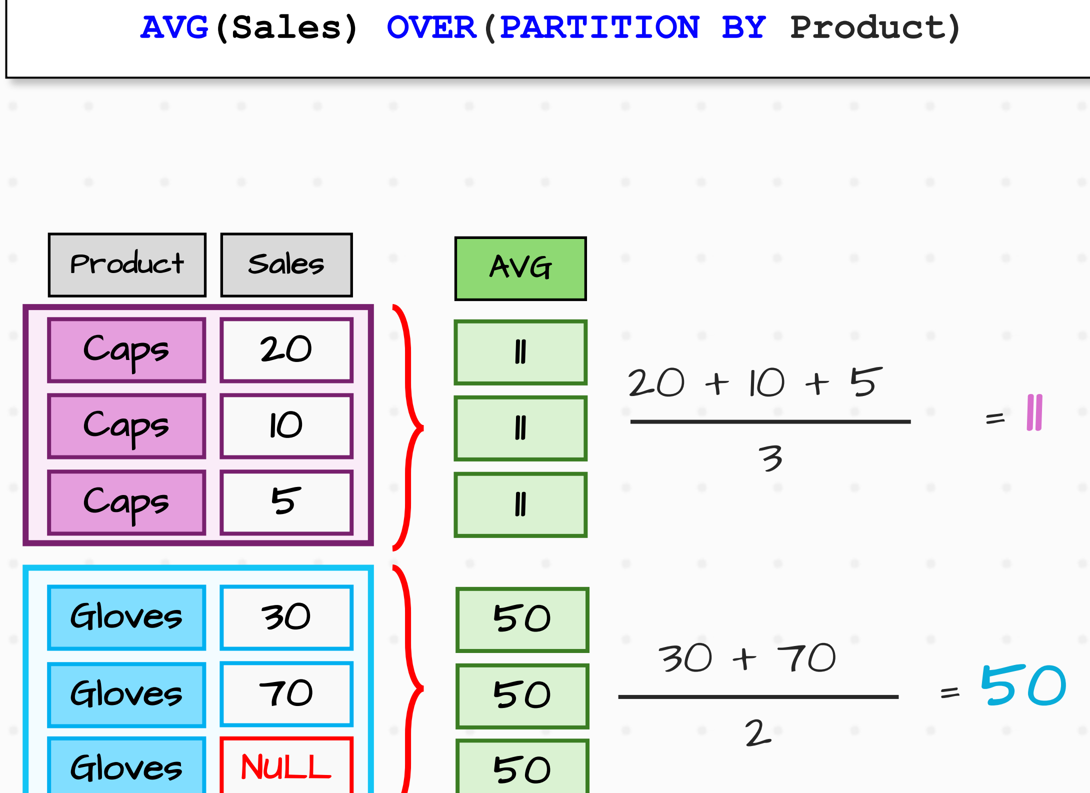




AVG Function

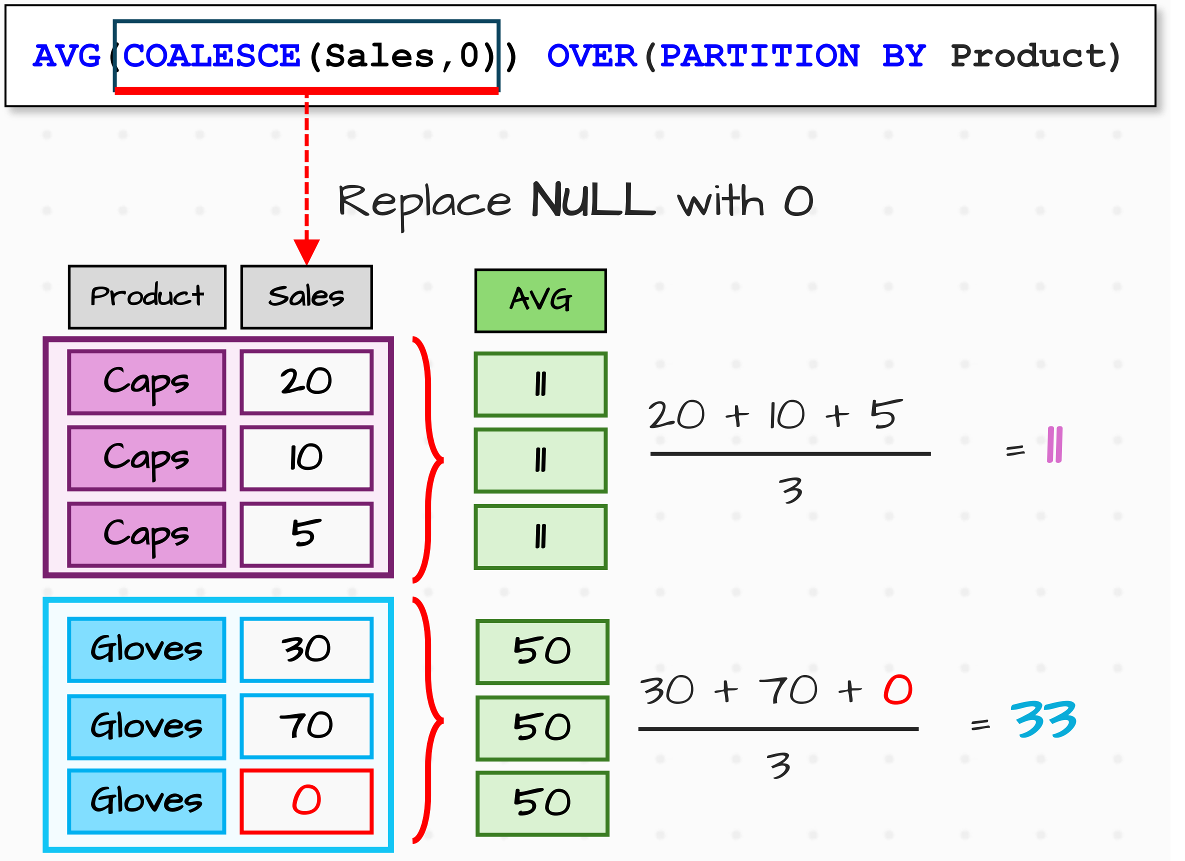
Default Average Function **exclude** NULLs

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



Deal with Nulls using **COALESCE** to **include** NULLs

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



AVG Function

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

Product	Sales	AVG
Caps	20	11
Caps	10	11
Caps	5	11
Gloves	30	33
Gloves	70	33
Gloves	0	33

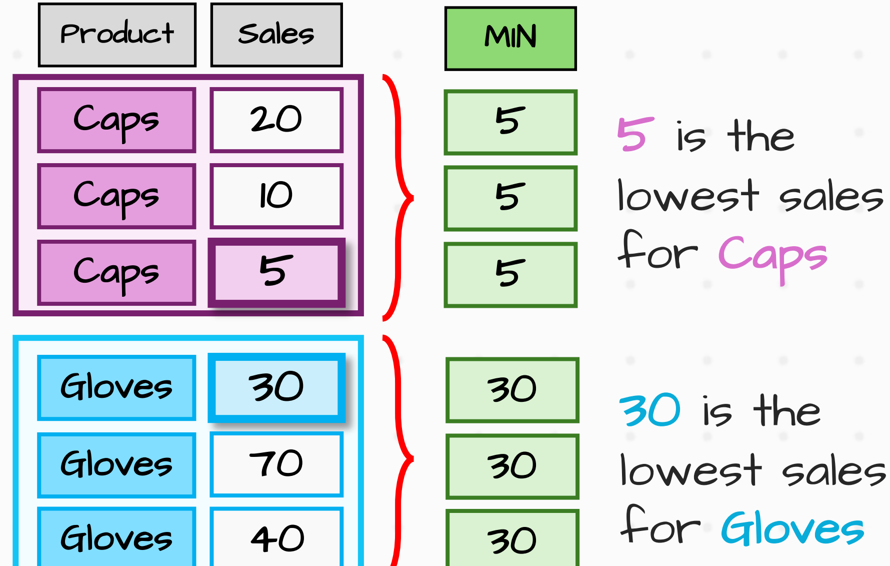
$(20 + 10 + 5) / 3 = 11$

$(30 + 70 + 0) / 3 = 33$

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

Returns the minimum value in a window

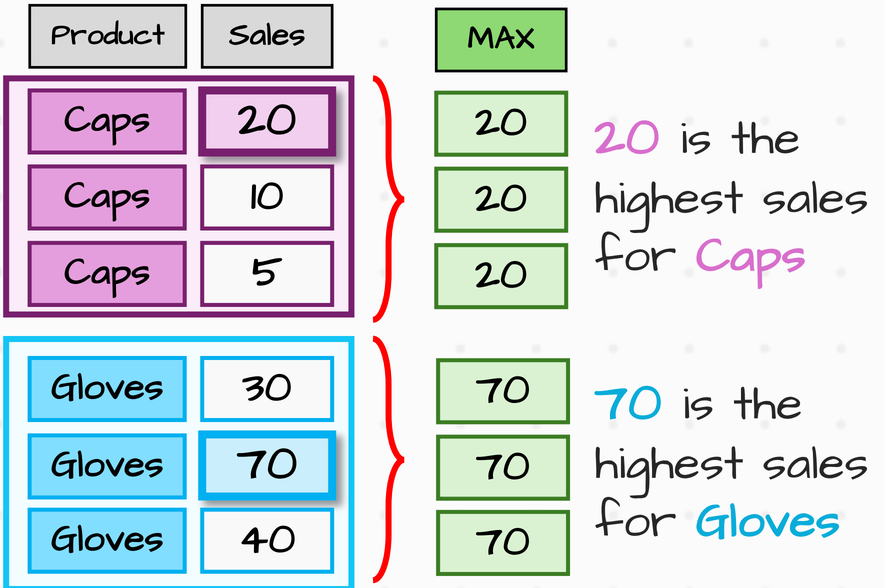
```
MIN(Sales) OVER(PARTITION BY Product)
```



MAX

Returns the maximum value in a window

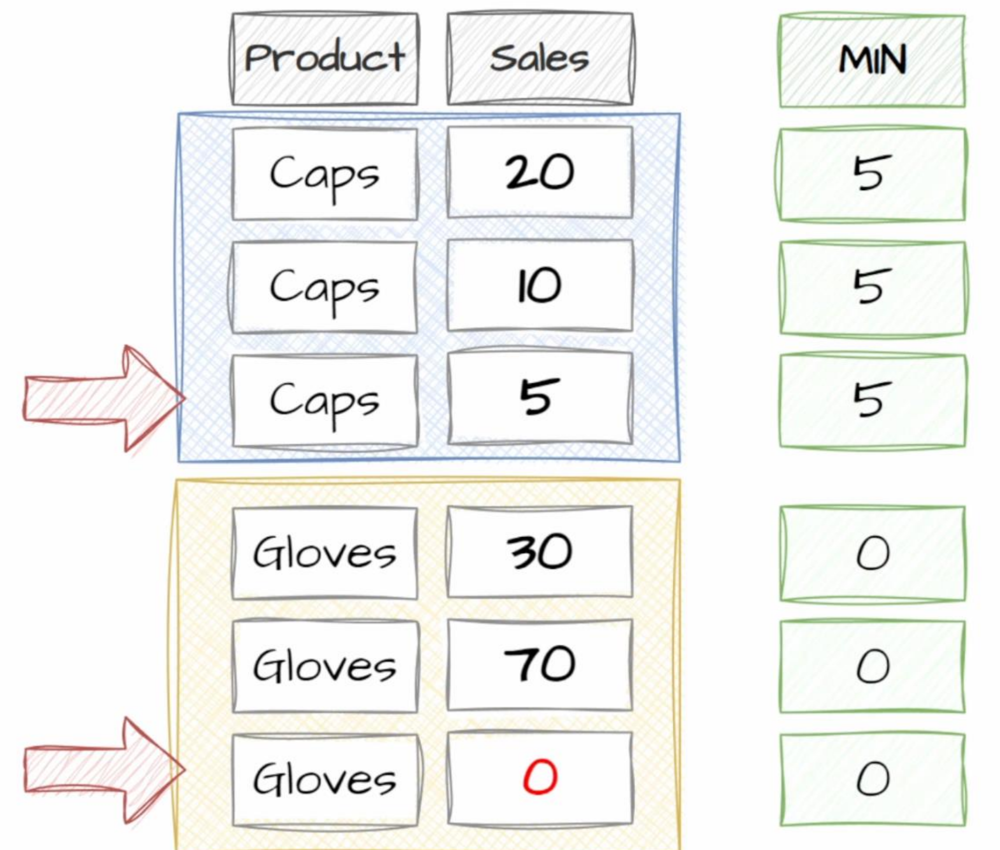
```
MAX(Sales) OVER(PARTITION BY Product)
```



MAX & MIN Function

Find the highest sales for each product

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



Find the lowest sales for each product

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

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
```

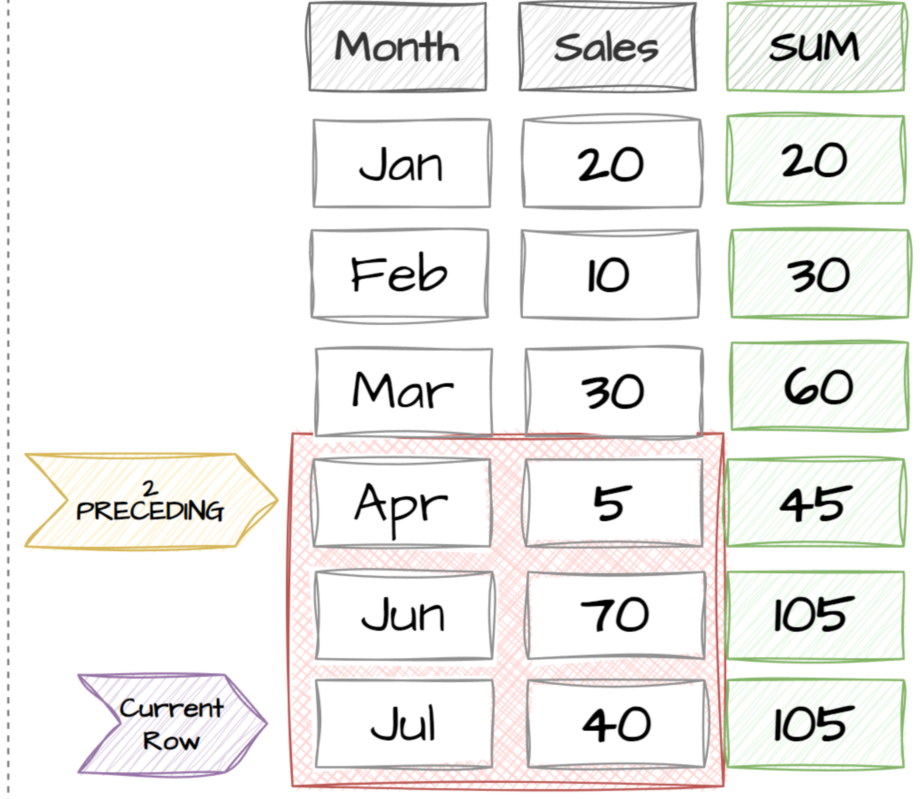
UNBOUNDED PRECEDING

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

Current Row

Rolling Total

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

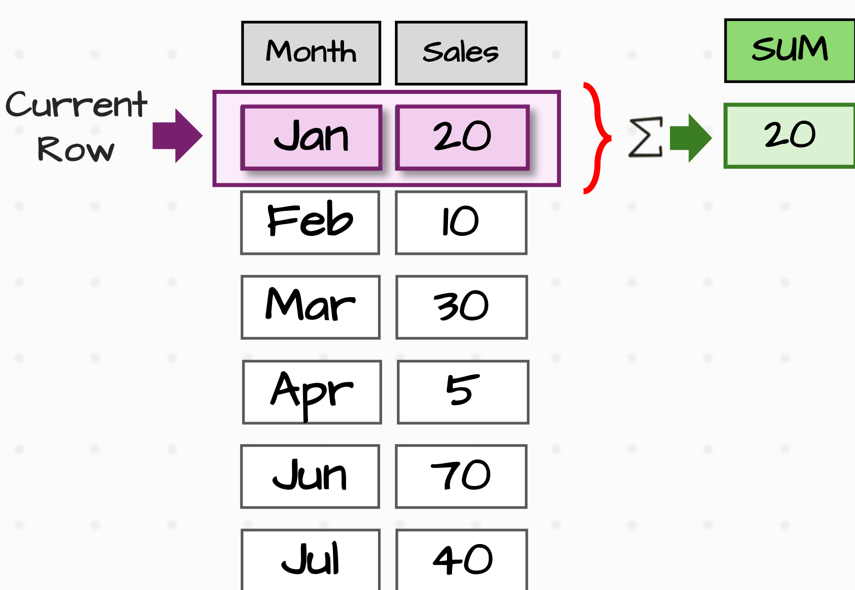


Running Total

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

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

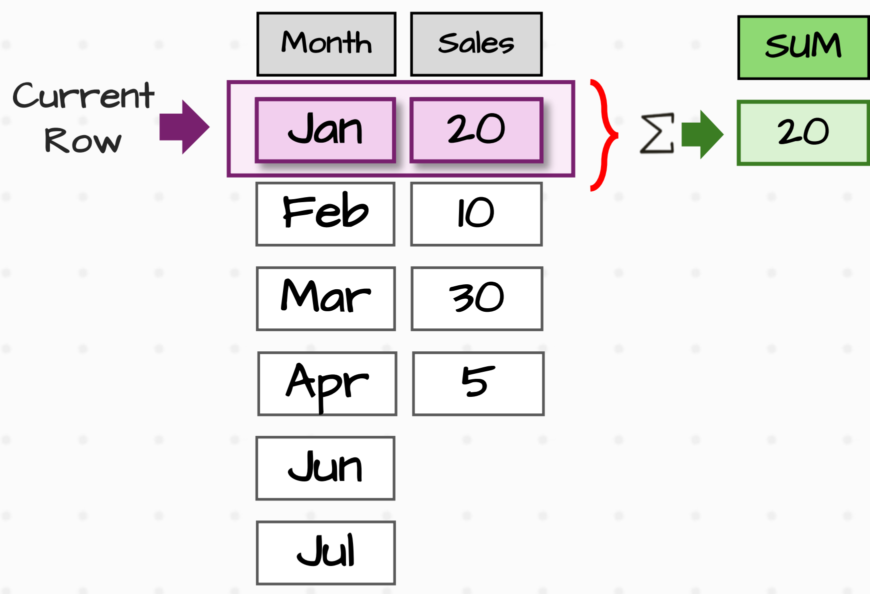
Default Frame
`RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`



Rolling Total

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

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



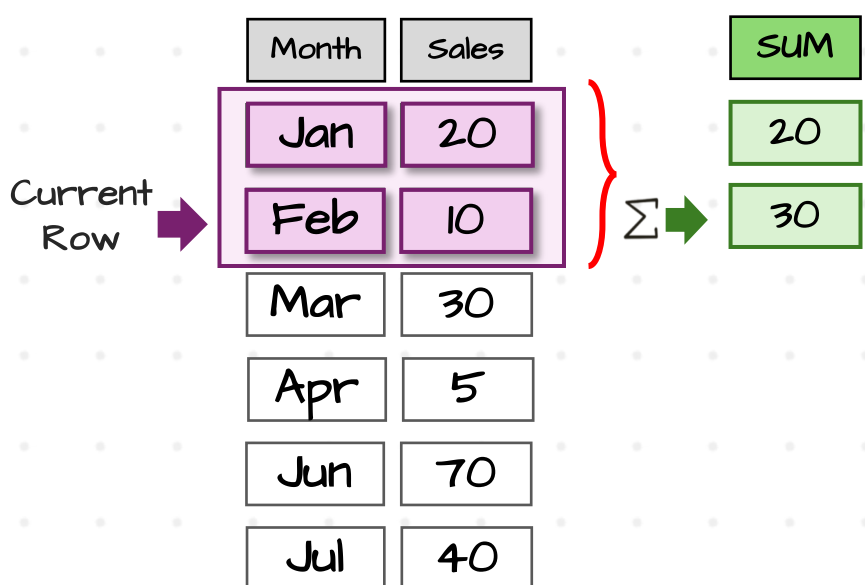
Running Total

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

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

Default Frame

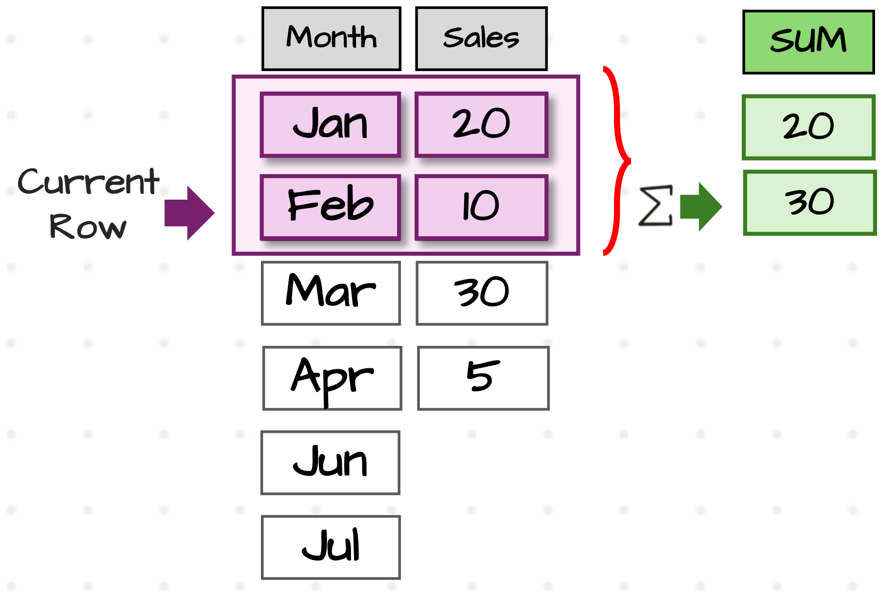
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

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

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

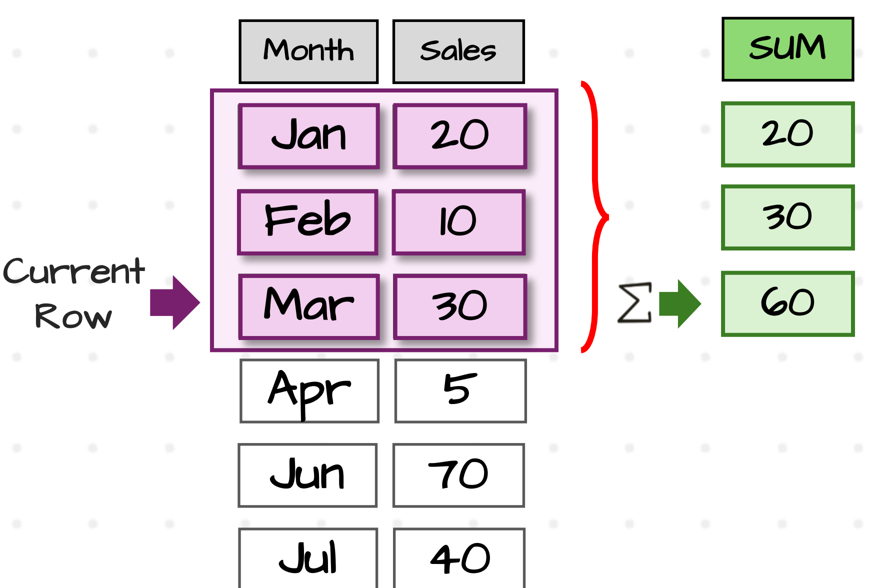


Running Total

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

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

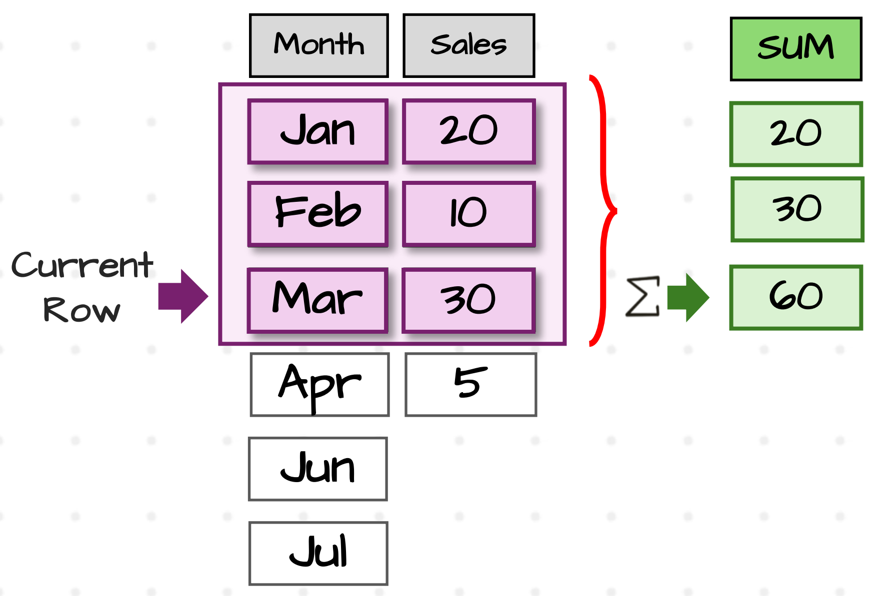
Default Frame
RANGE BETWEEN *UNBOUNDED PRECEDING AND CURRENT ROW*



Rolling Total

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

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



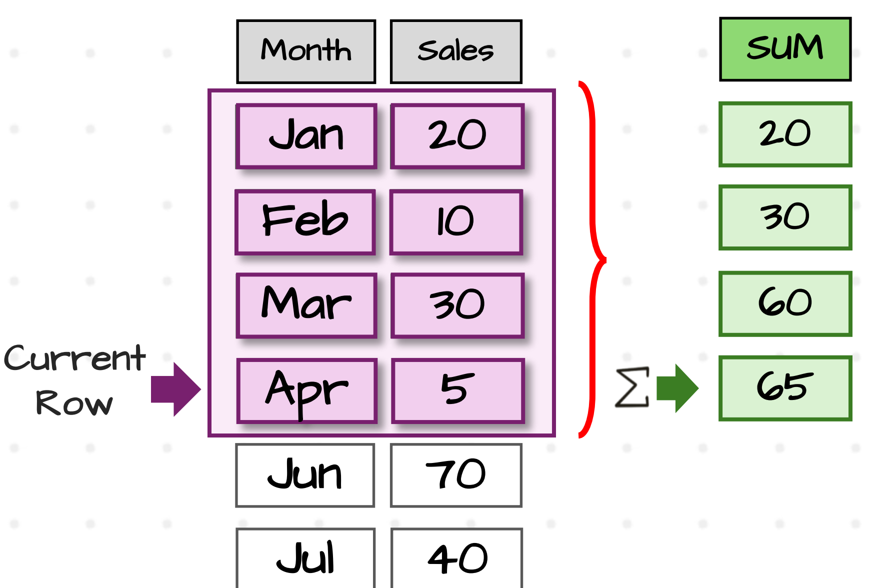
Running Total

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

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

Default Frame

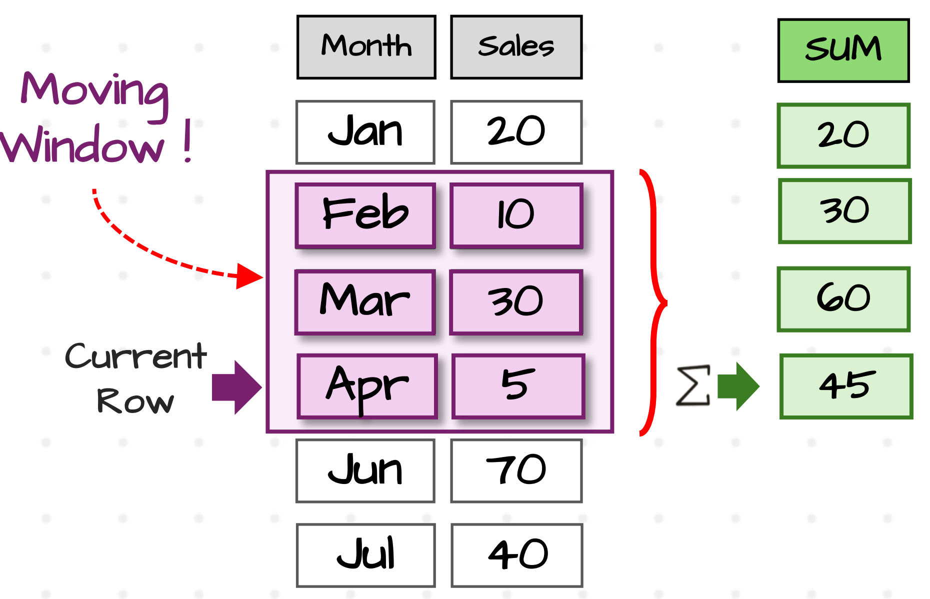
```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```



Rolling Total

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

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



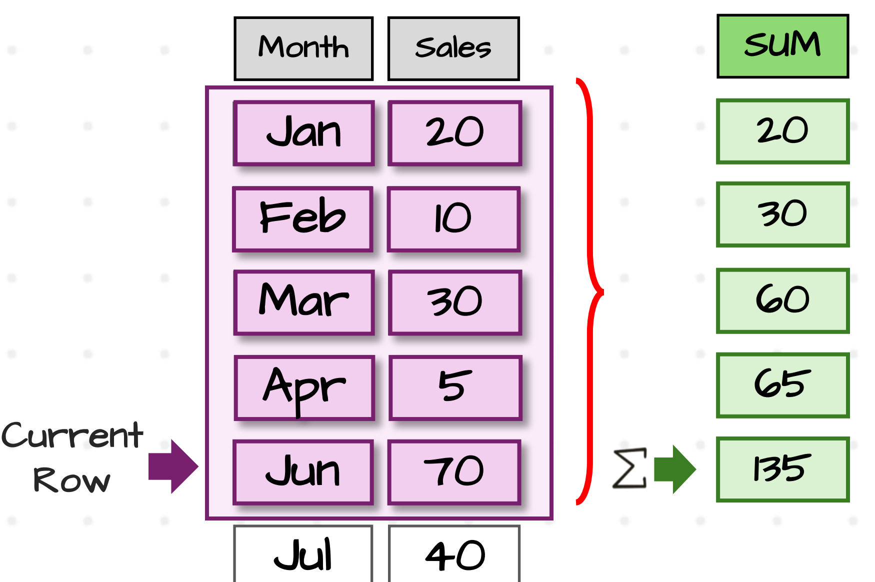
Running Total

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

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

Default Frame

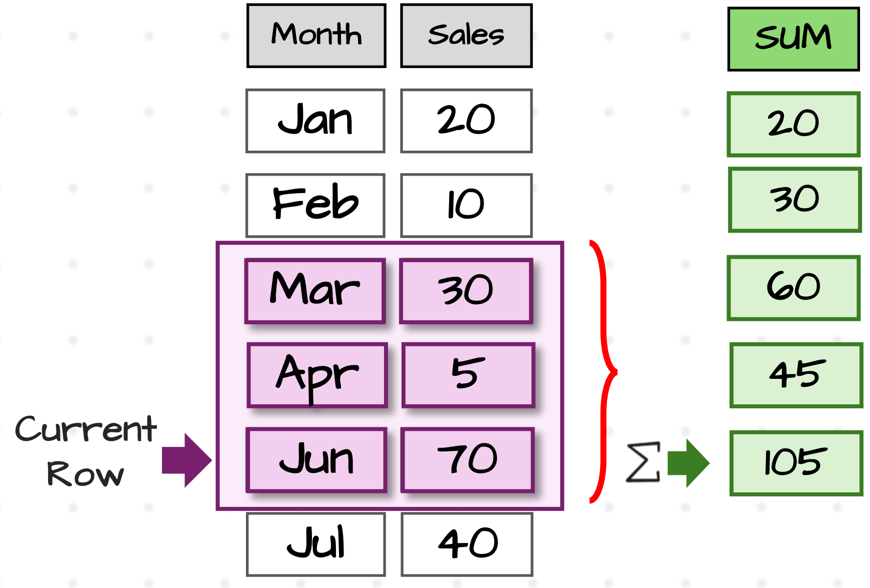
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW



Rolling Total

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

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

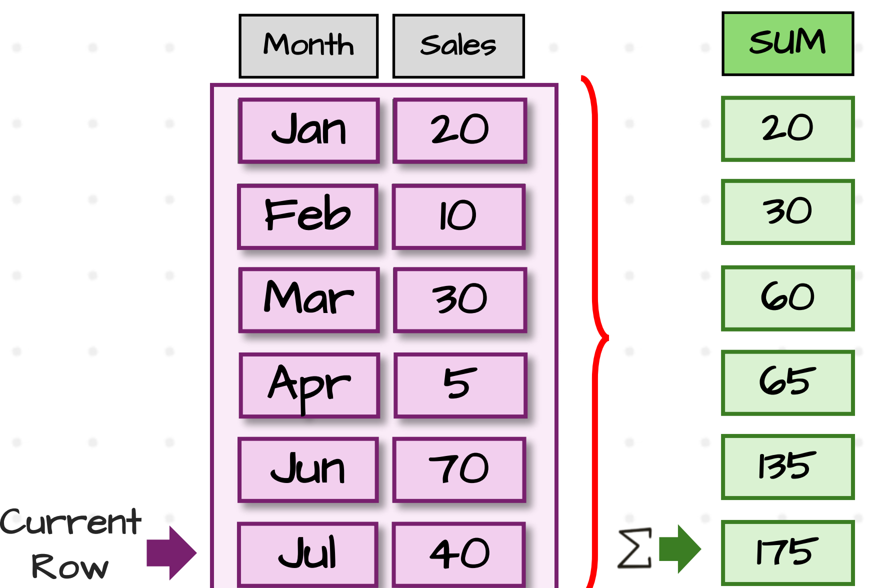


Running Total

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

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

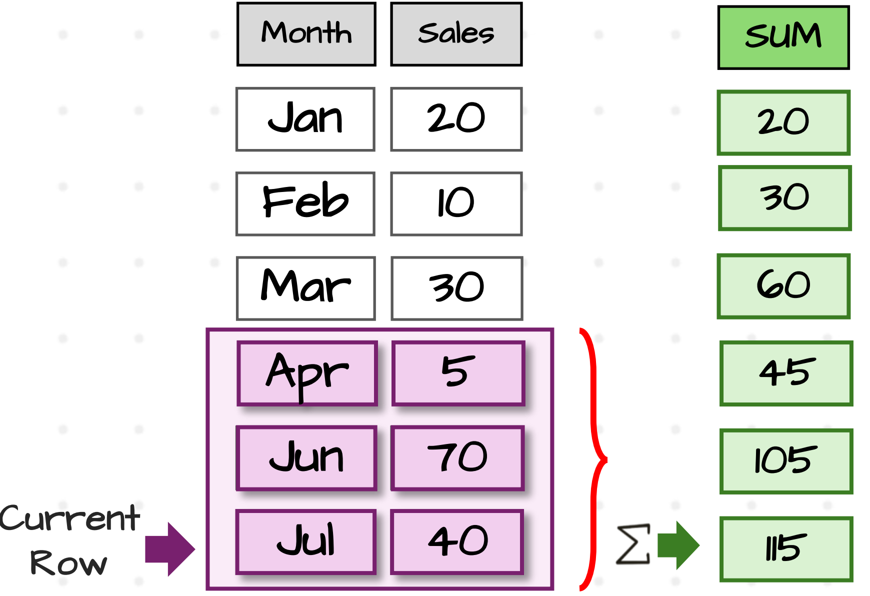
Default Frame
`RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`

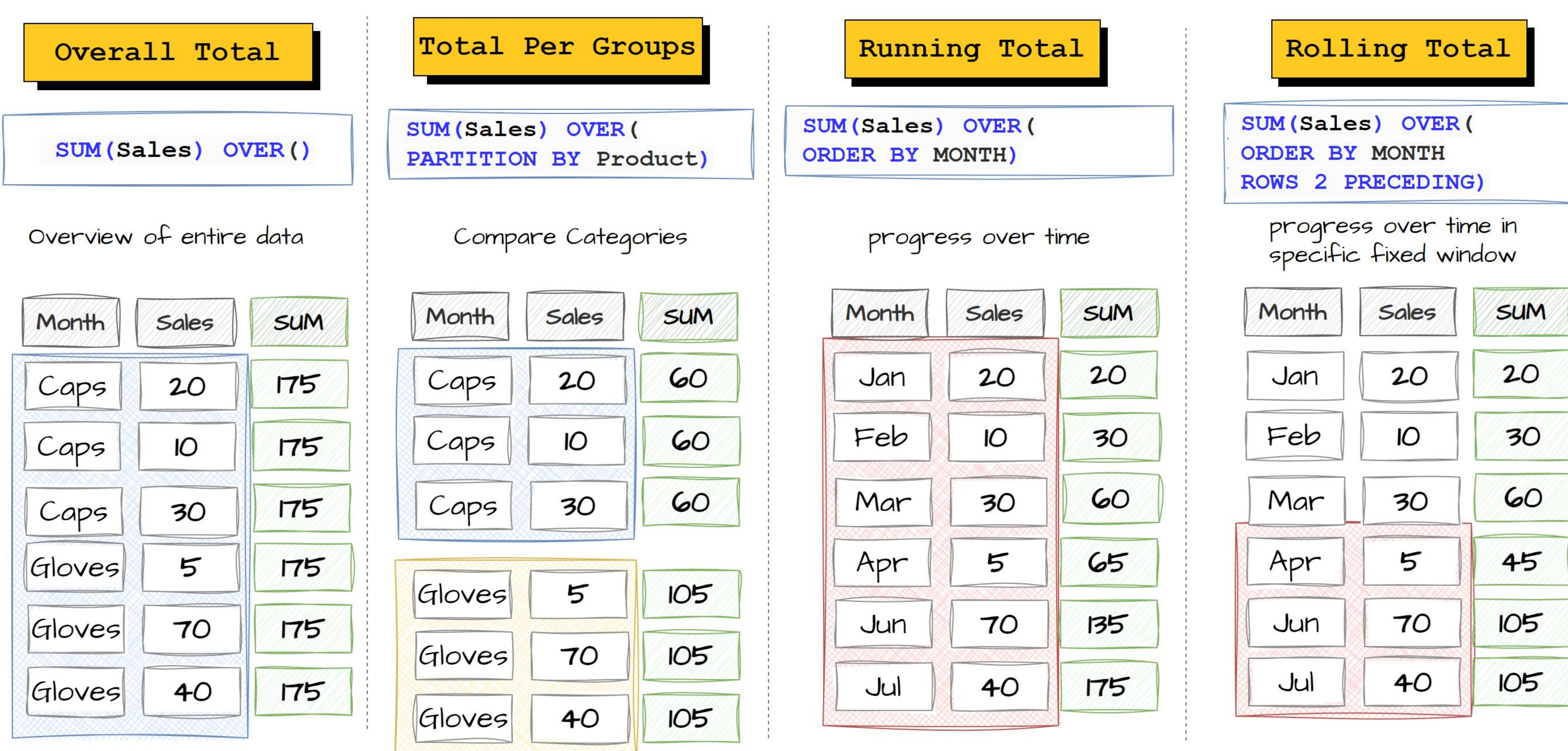


Rolling Total

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

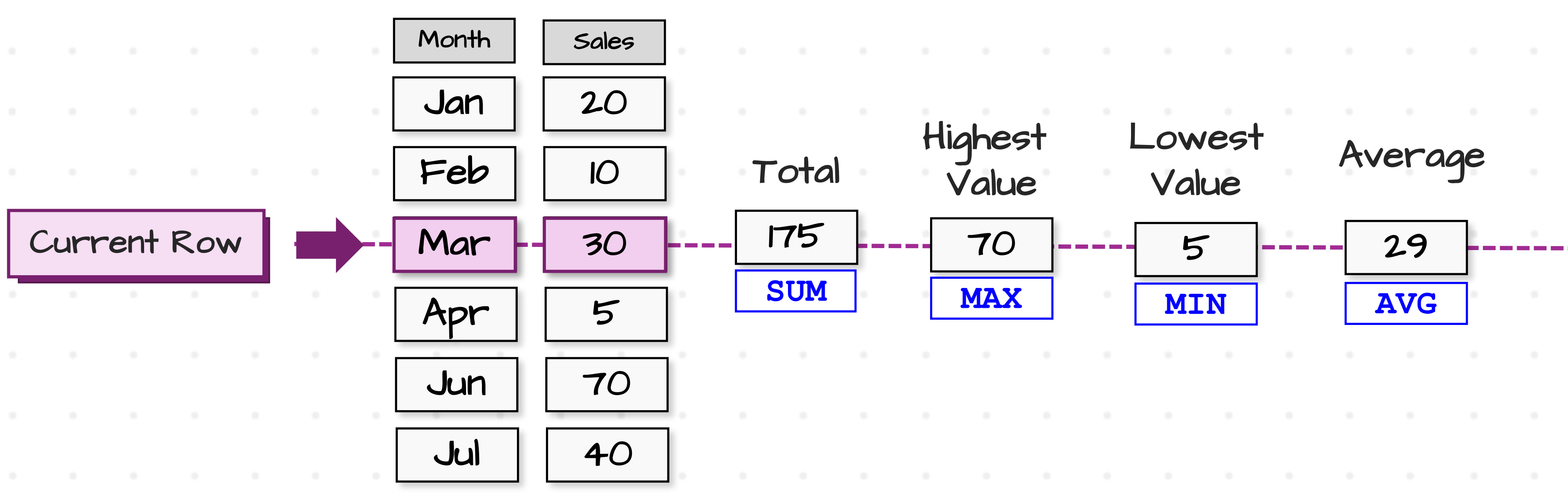
```
SUM(Sales) OVER(
ORDER BY Month ROWS 2 PRECEDING)
```




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

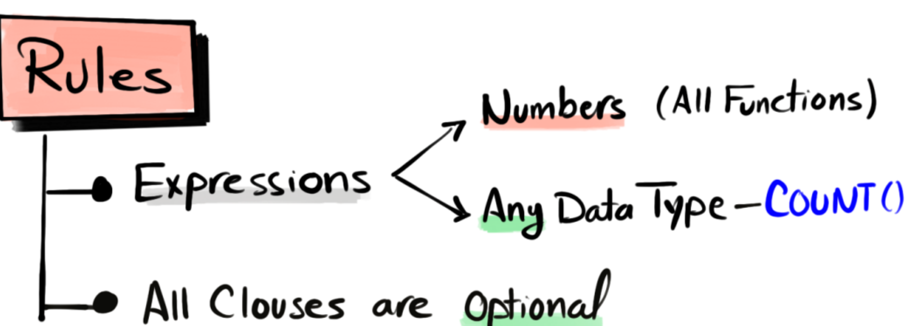
Comparision Use Cases

Compare the current value and aggregated value of window functions



WINDOW AGGREGATE FUNCTIONS

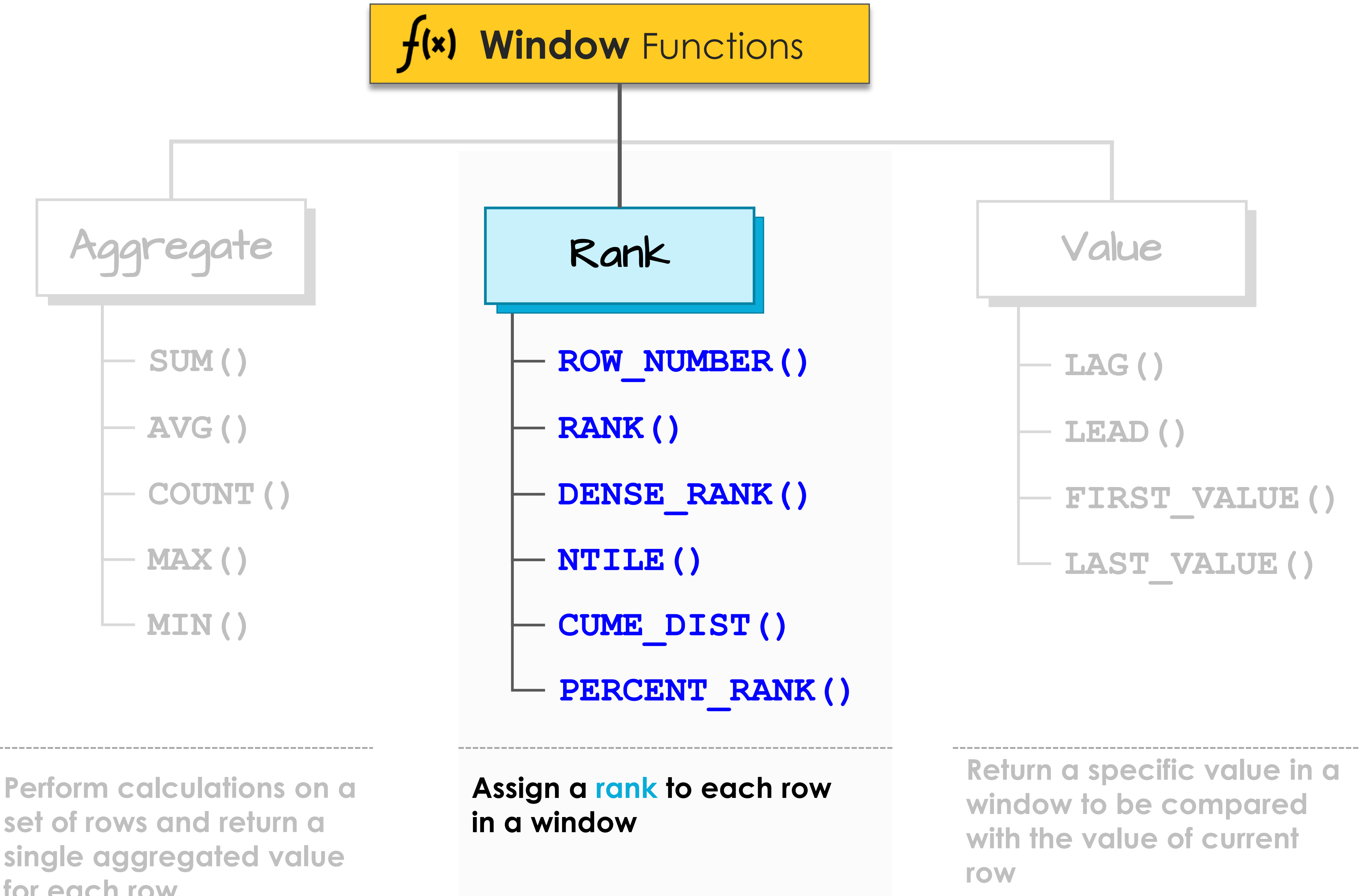
Aggregate set of Values and return a single aggregated Value

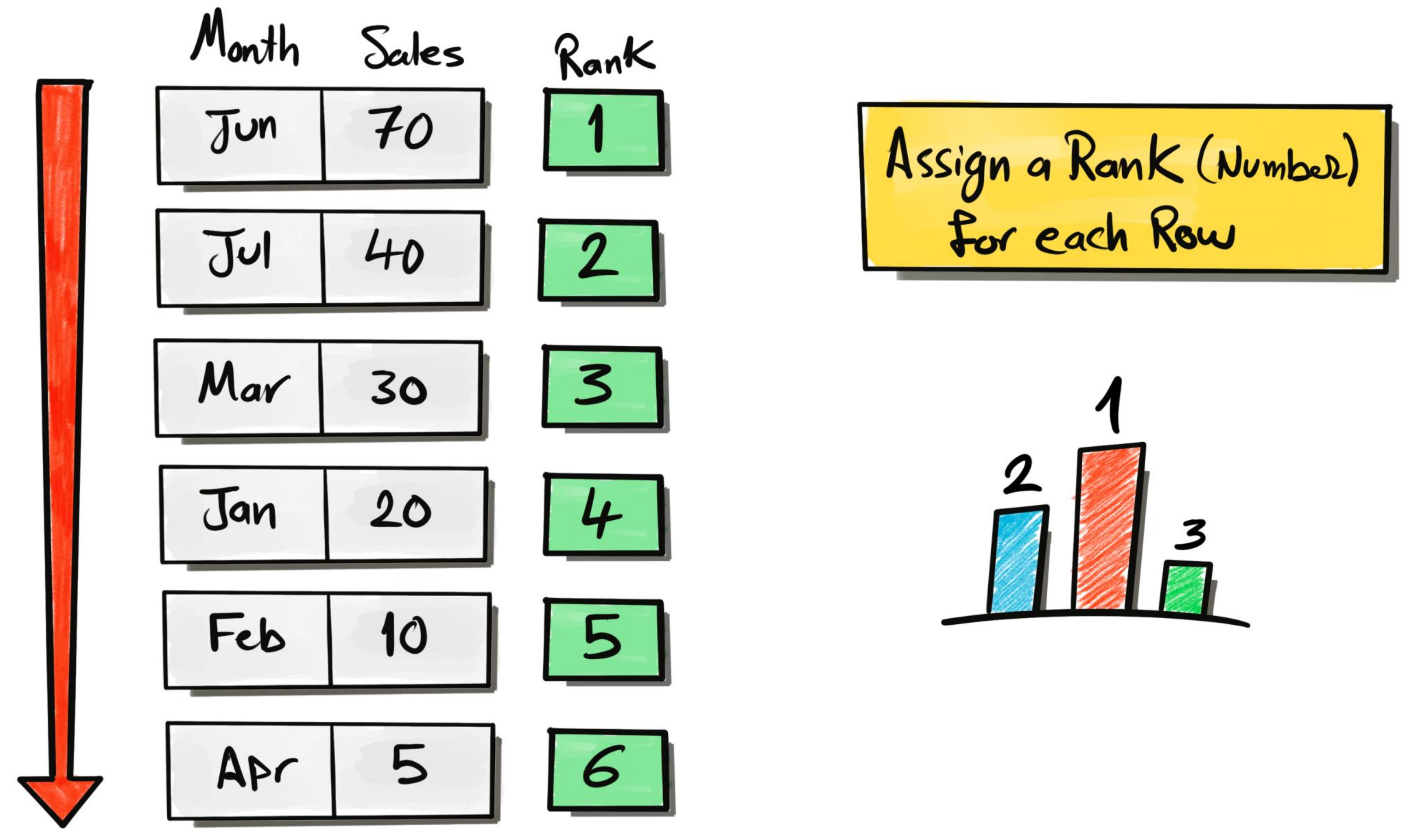


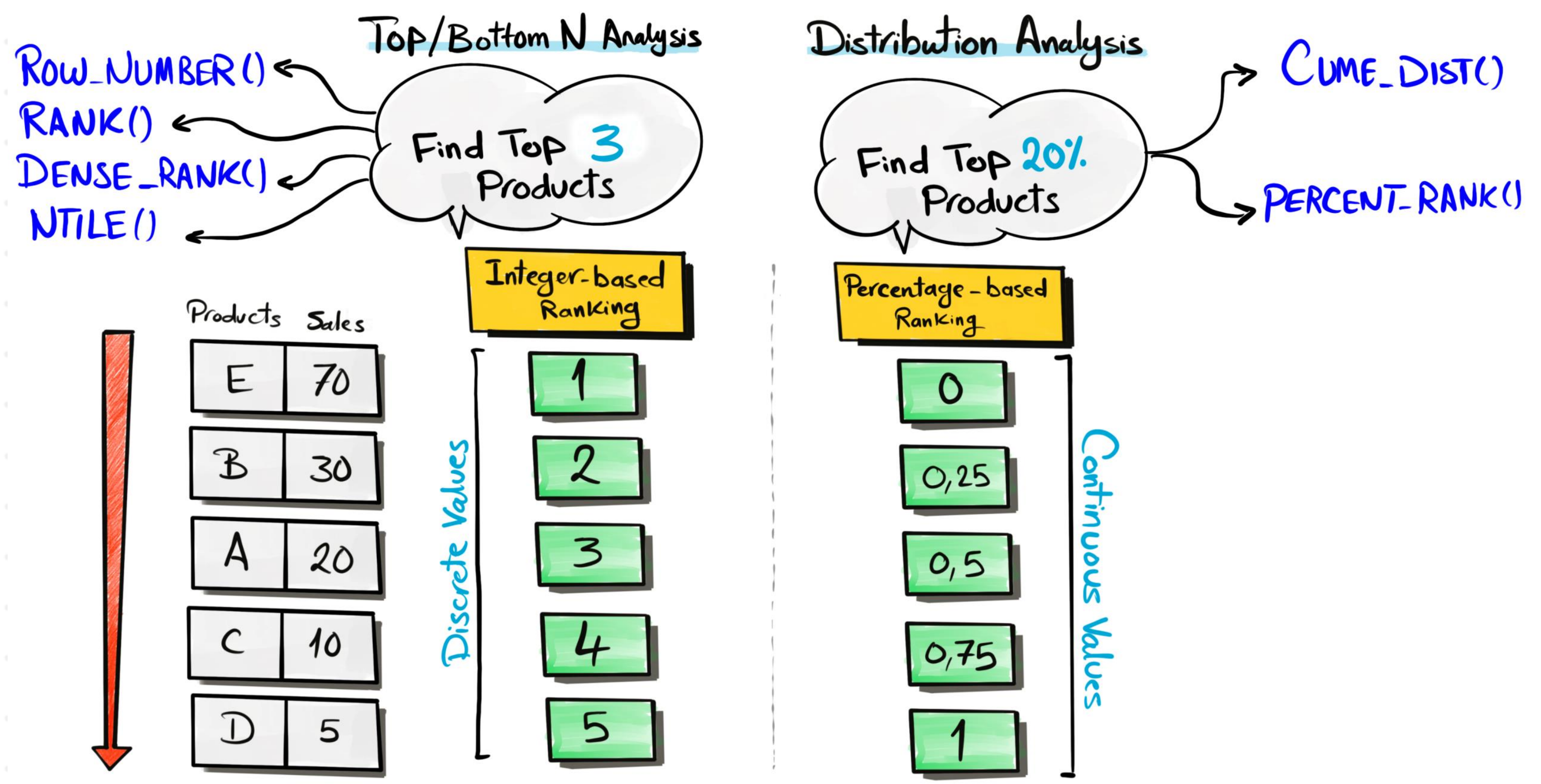
WINDOW RANKING

FUNCTIONS

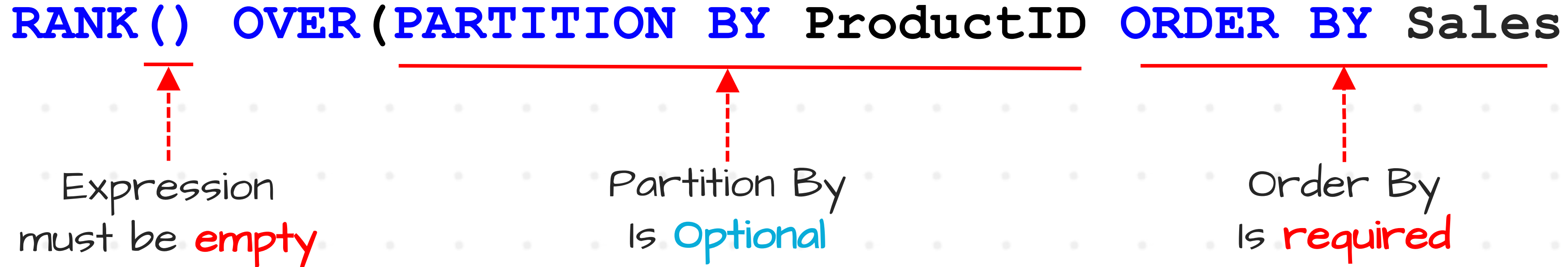
Baraa Khatib Salkini


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

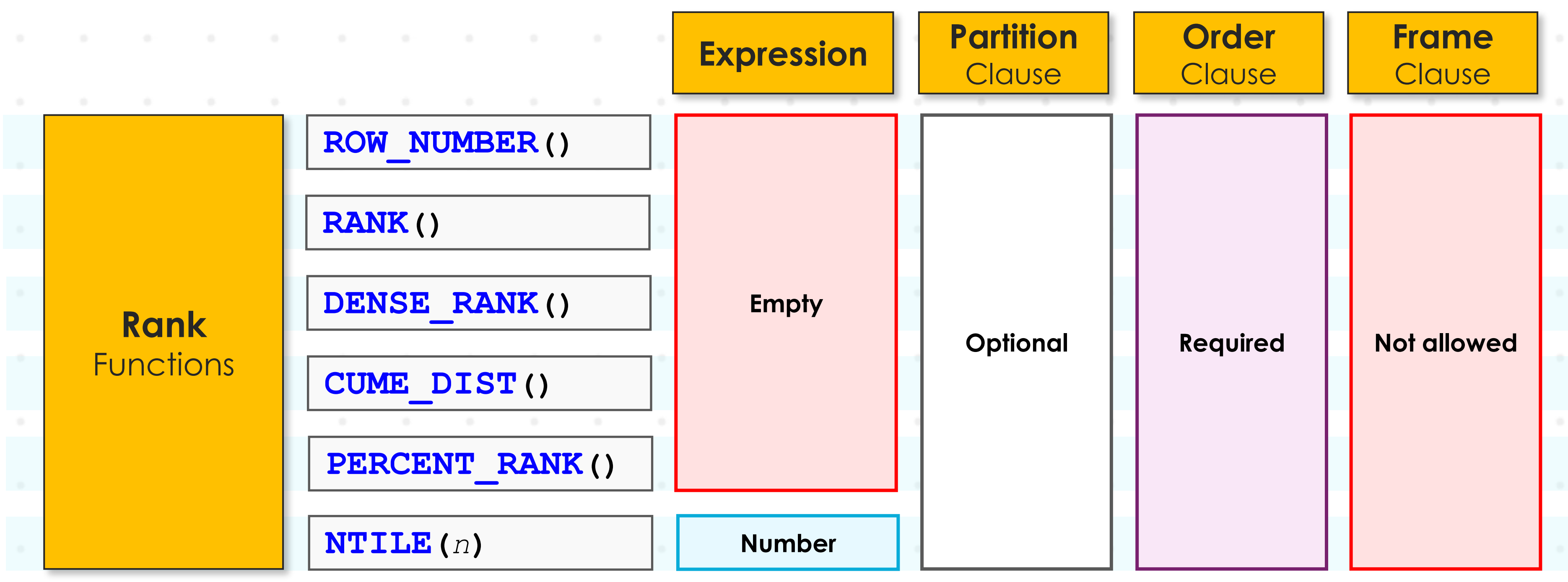

Month
Sales
Rank
Jun
70
1
Jul
40
2
Mar
30
3
Jan
20
4
Feb
10
5
Apr
5
6
Assign a Rank (Number) for each Row
2
1
3


Top/Bottom N Analysis
Distribution Analysis
ROW_NUMBER()
RANK()
DENSE_RANK()
NTILE()
Find Top 3 Products
Find Top 20% Products
CUME_DIST()
PERCENT_RANK()
Integer-based Ranking
Percentage-based Ranking
Products
Sales
E
70
B
30
A
20
C
10
D
5
Discrete Values
1
2
3
4
5
0
0,25
0,5
0,75
1
Continuous Values

Ranking Function



Ranking Function



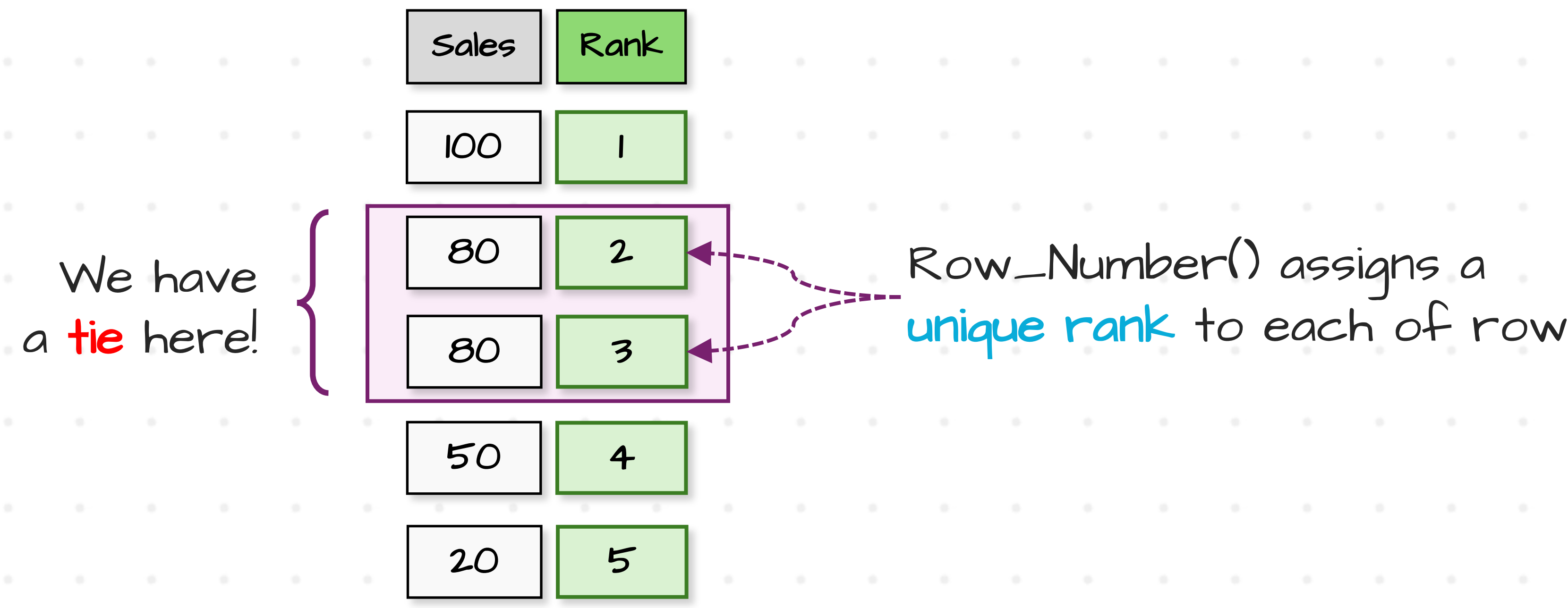
Ranking Function

Rank Functions	Function	Description	Example
	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

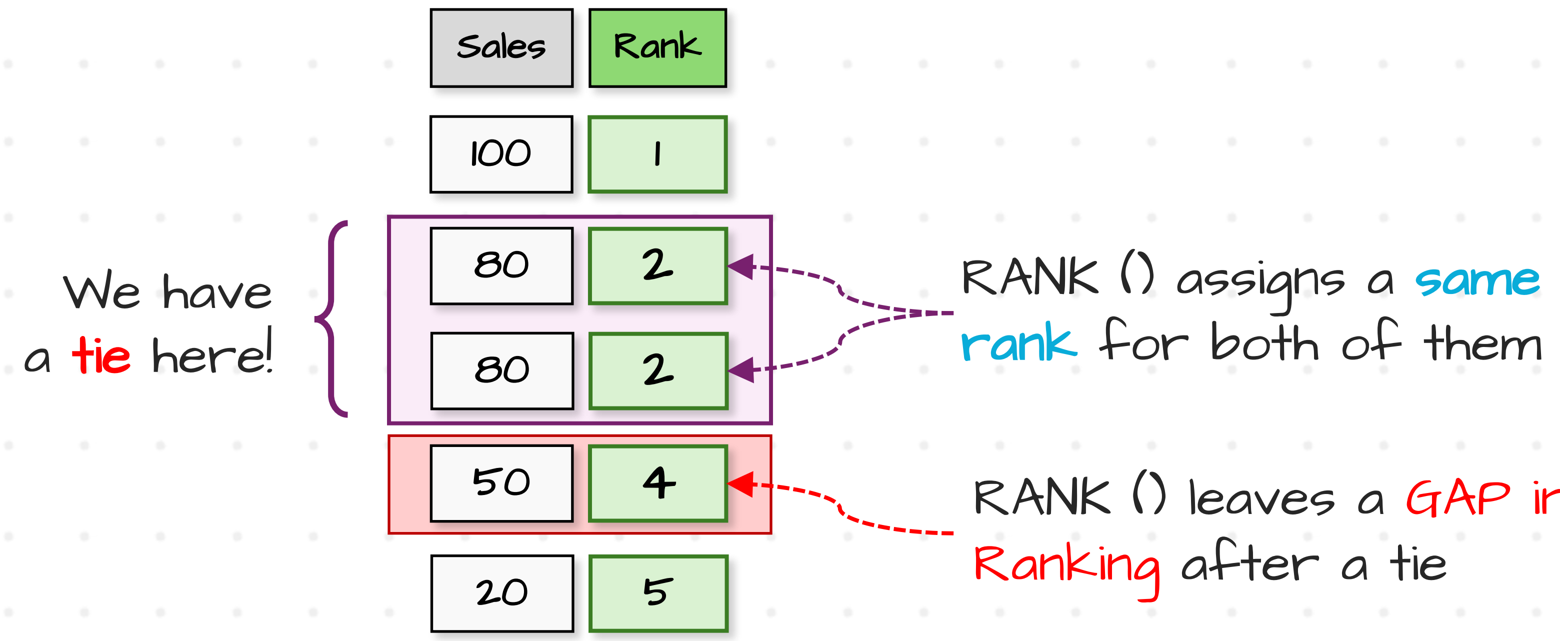
```
ROW_NUMBER() OVER (ORDER BY Sales DESC)
```



RANK

Assign a rank to each row with in a window

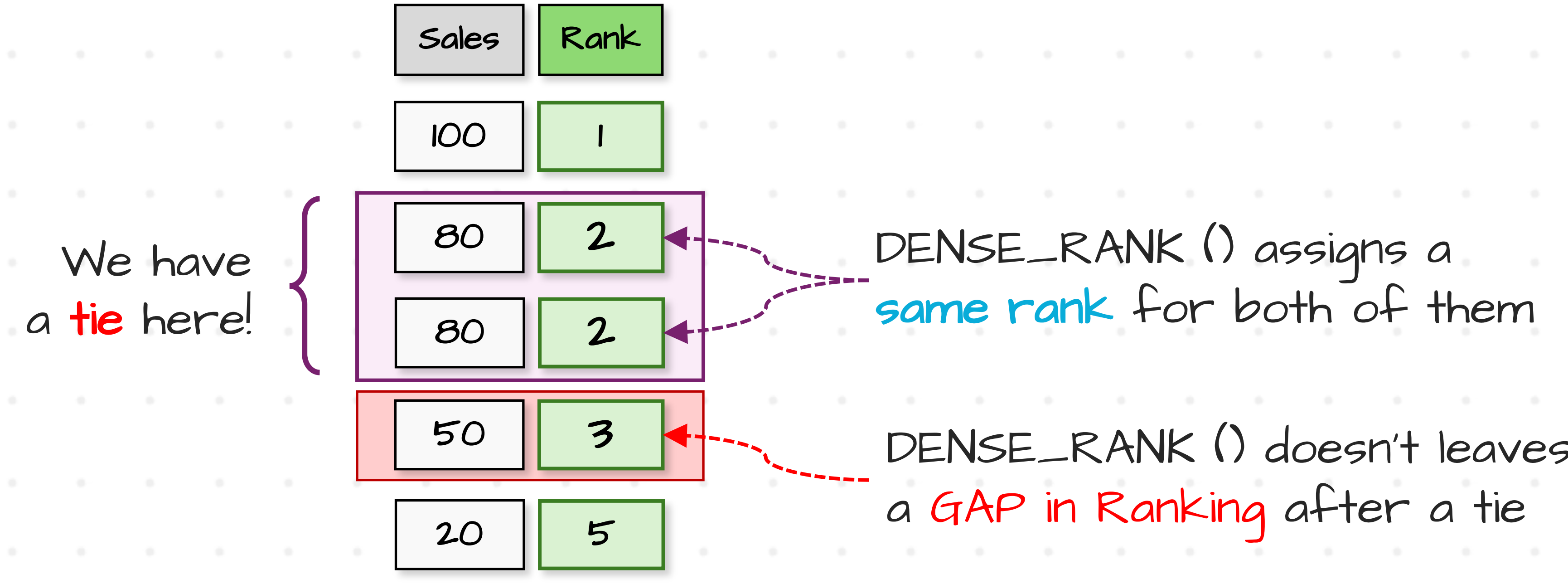
```
RANK() OVER(ORDER BY Sales DESC)
```



DENSE_RANK

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)
```



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

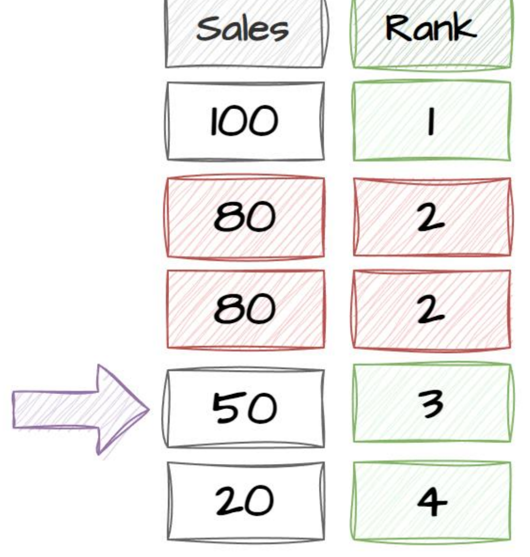
Shared Rank

Handles Ties

Gaps in Ranks

DENSE_RANK()

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

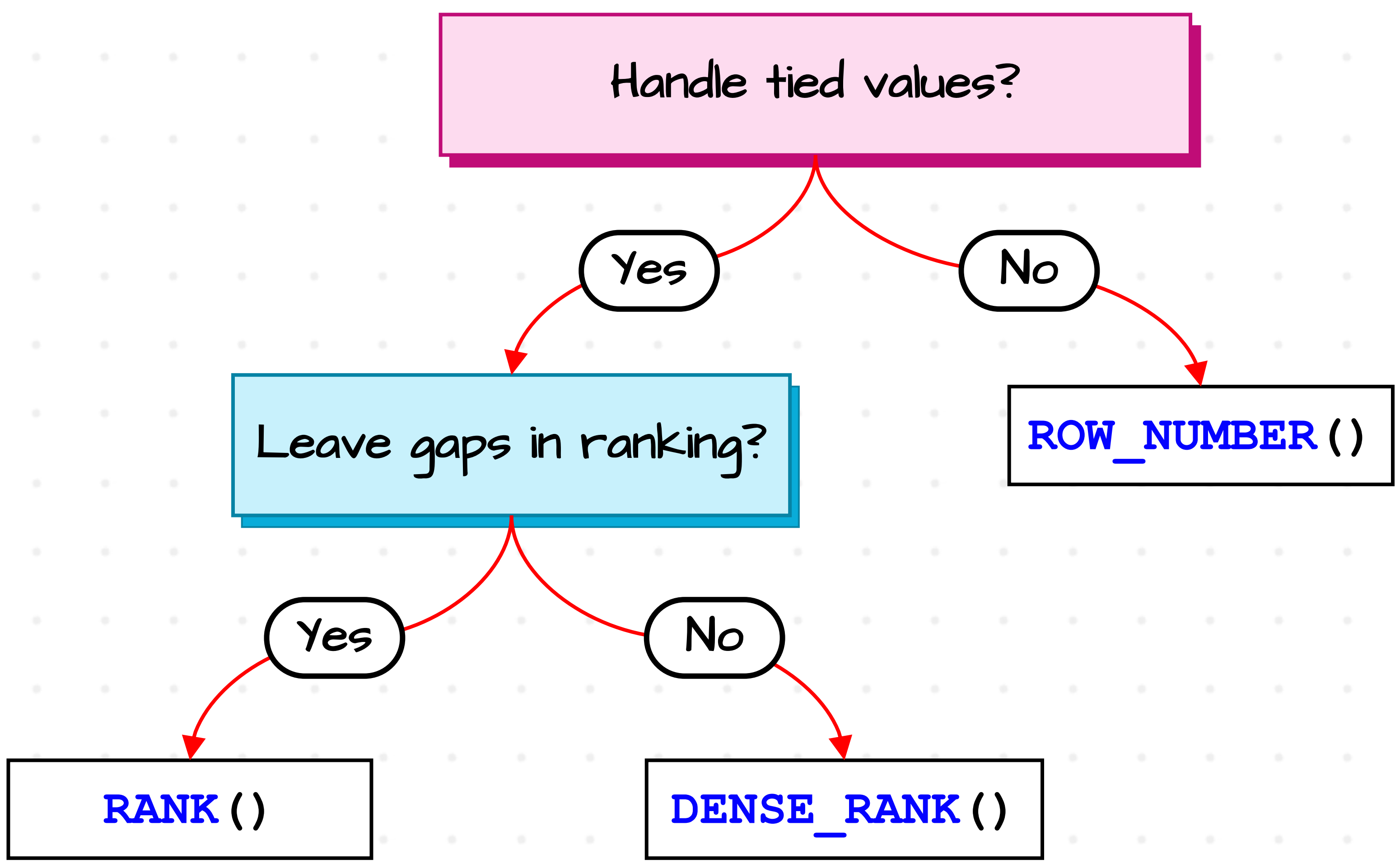


Shared Rank

Handles Ties

No Gaps in Ranks

Which One To Use?

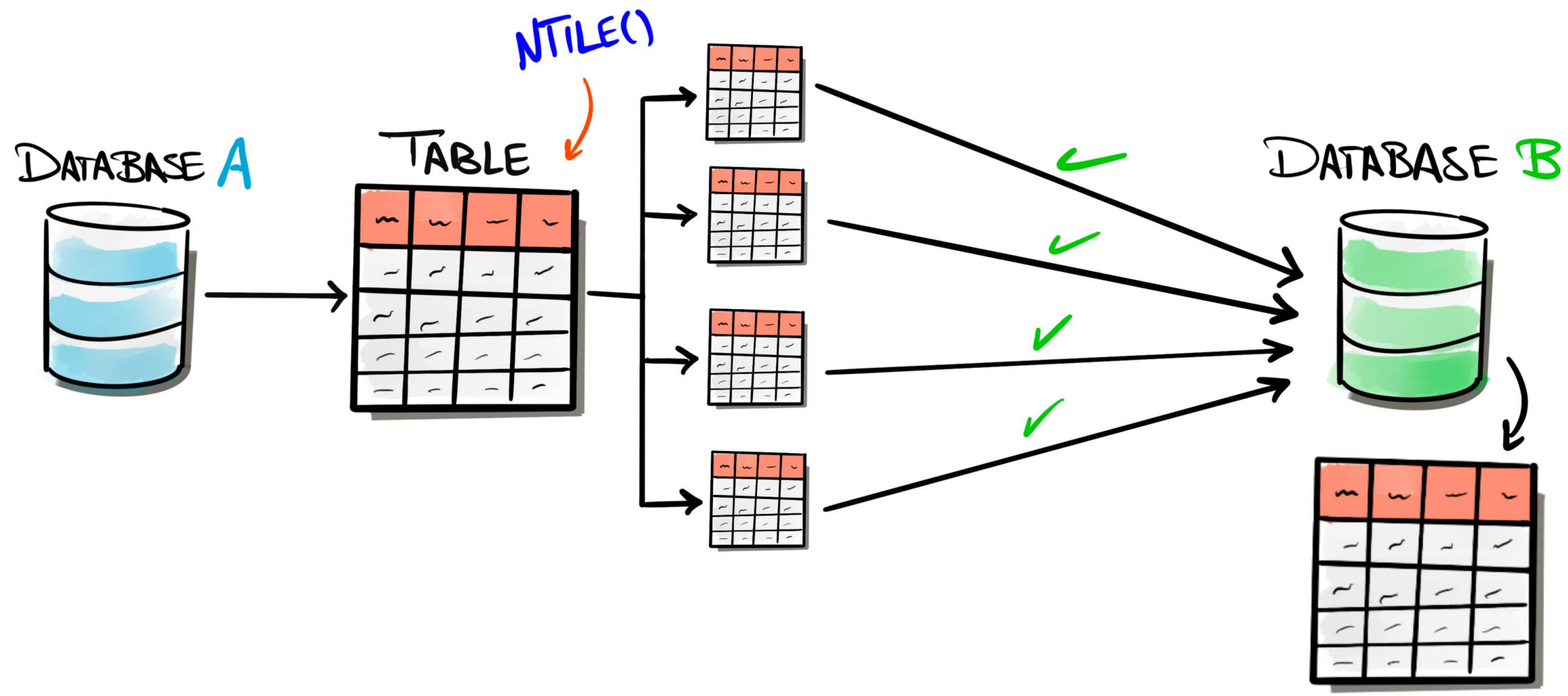


NTILE()

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

DATA ANALYST	DATA ENGINEER
Data Segmentation	Equalizing load processing

NTILE Use Case



NTILE Use Case

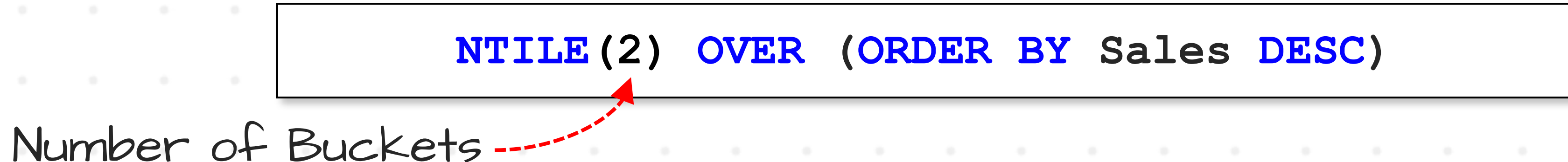
```
NTILE(2) OVER (ORDER BY Sales DESC)
```

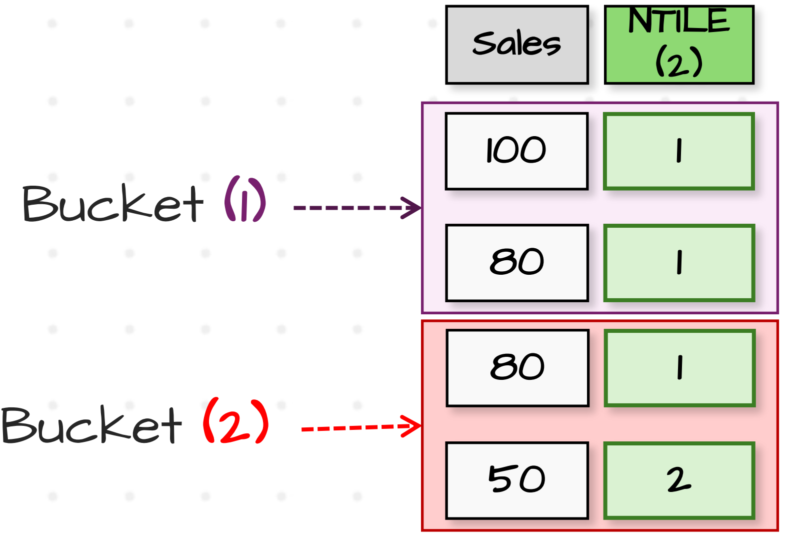
$$\text{Bucket Size} = \frac{\text{Number of Rows}}{\text{Number of Buckets}}$$

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

NTILE

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





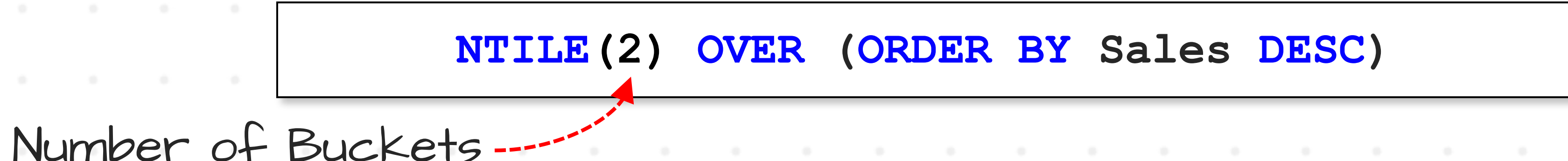
$$\text{Bucket Size} = \frac{\text{Number of Rows}}{\text{Number of Buckets}}$$

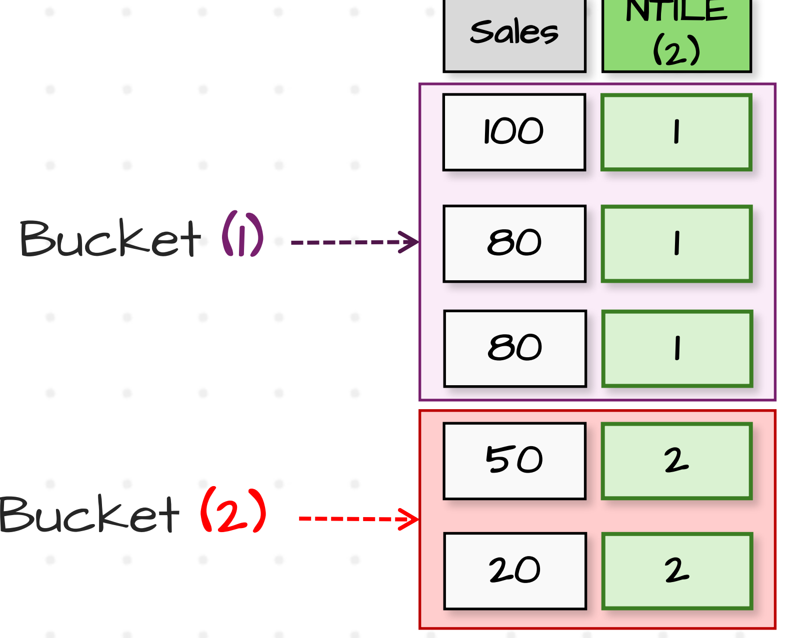
(Nr of Rows in each Bucket)

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

NTILE

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





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

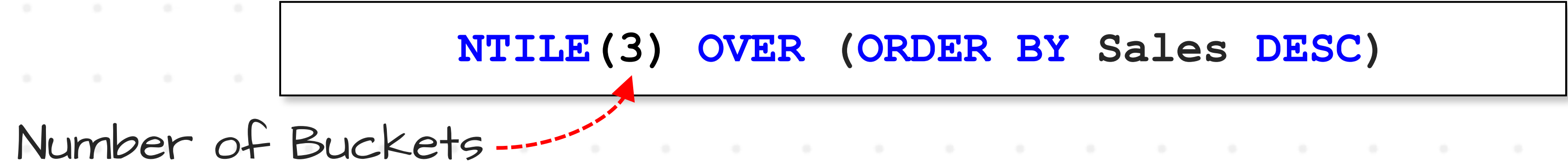
(Nr of Rows in each Bucket)

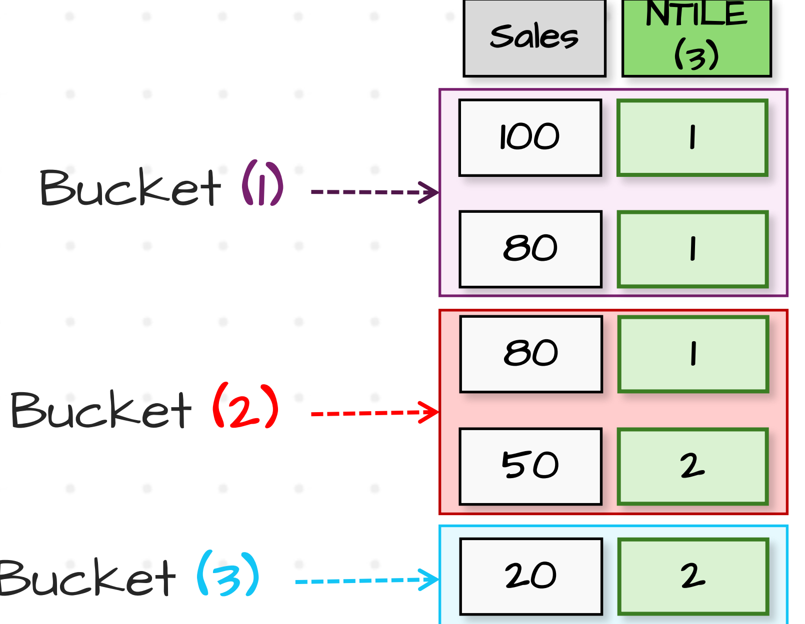
$$\text{Bucket Size} = \frac{5}{2} = 2.5$$

Larger groups come first then smaller groups

NTILE

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





Bucket Size (Nr of Rows in each Bucket) = Number of Rows / Number of Buckets

Bucket Size = 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

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

$$CUME_DIST = \frac{5}{5}$$

CUME_DIST()

Cumulative Distribution calculates the distribution of data points within a window

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

PERCENT_RANK()

Calculates the relative position of each row

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

```
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

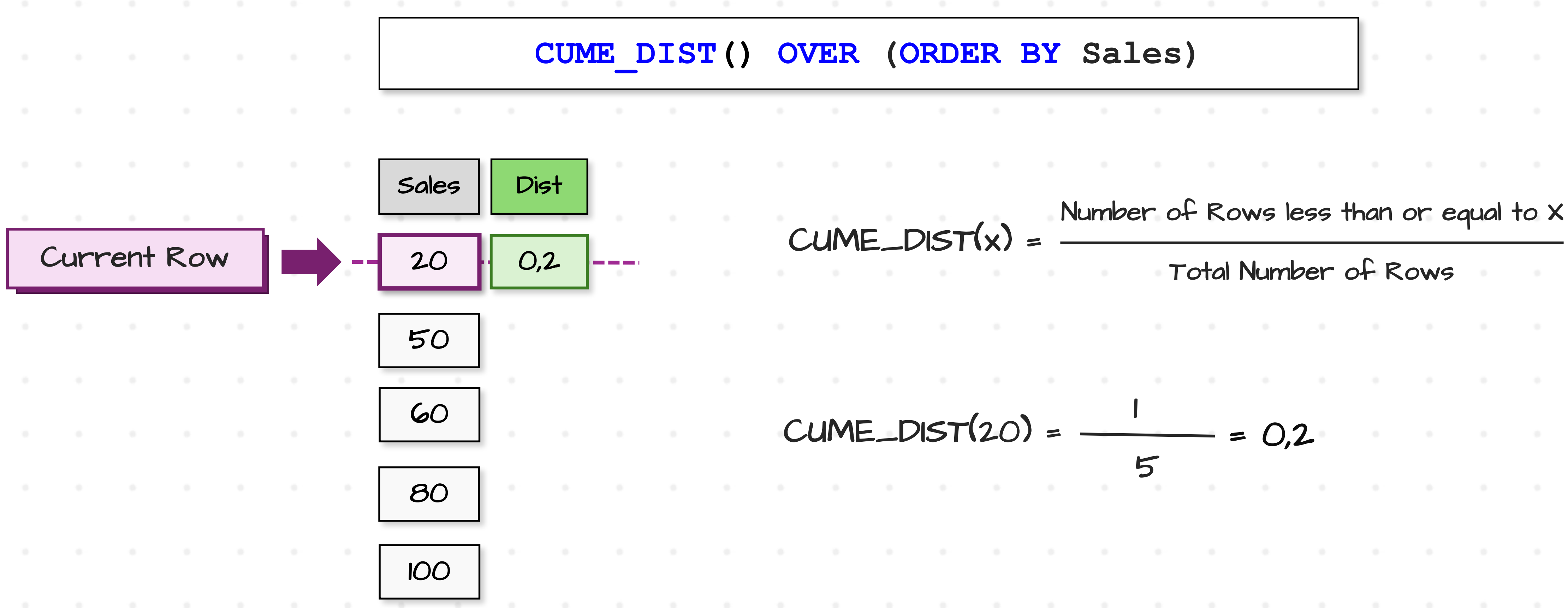
$$\text{CUME_DIST} = \frac{\text{Position Nr}}{\text{Number of Rows}}$$

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

CUME_DIST

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

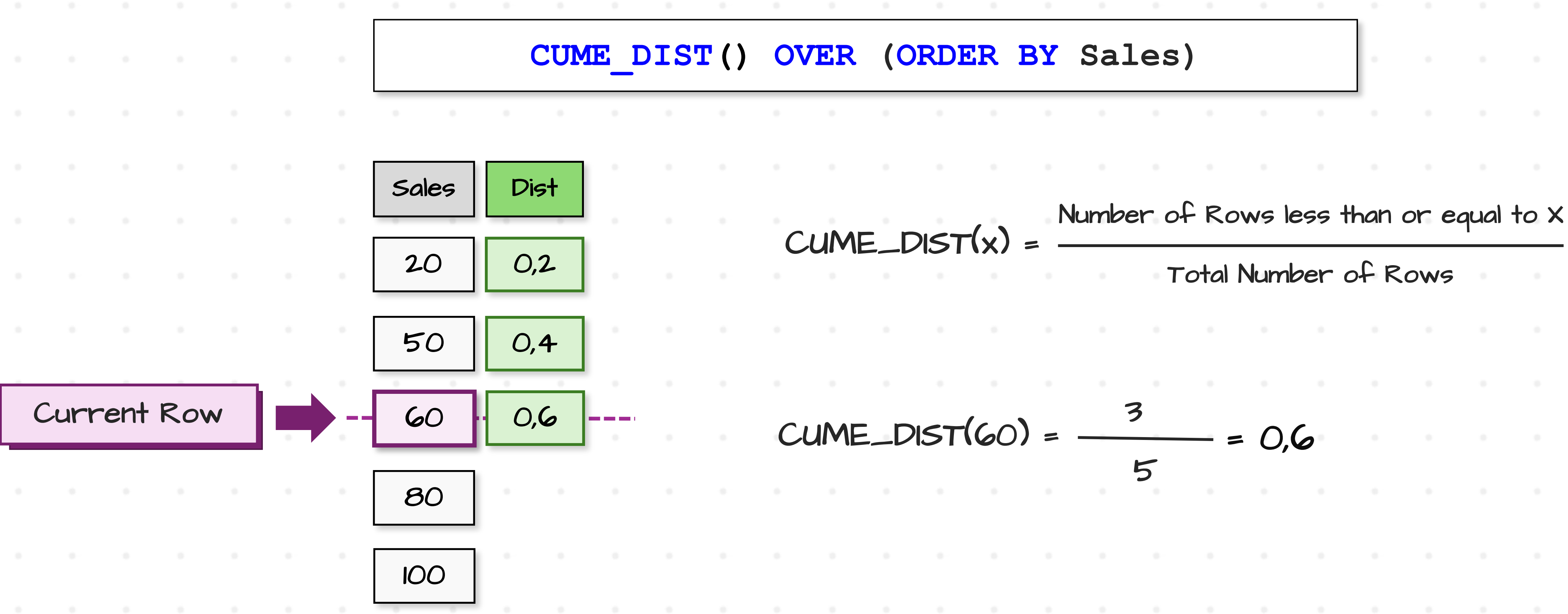
```
CUME_DIST() OVER (ORDER BY Sales)
```



CUME_DIST

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

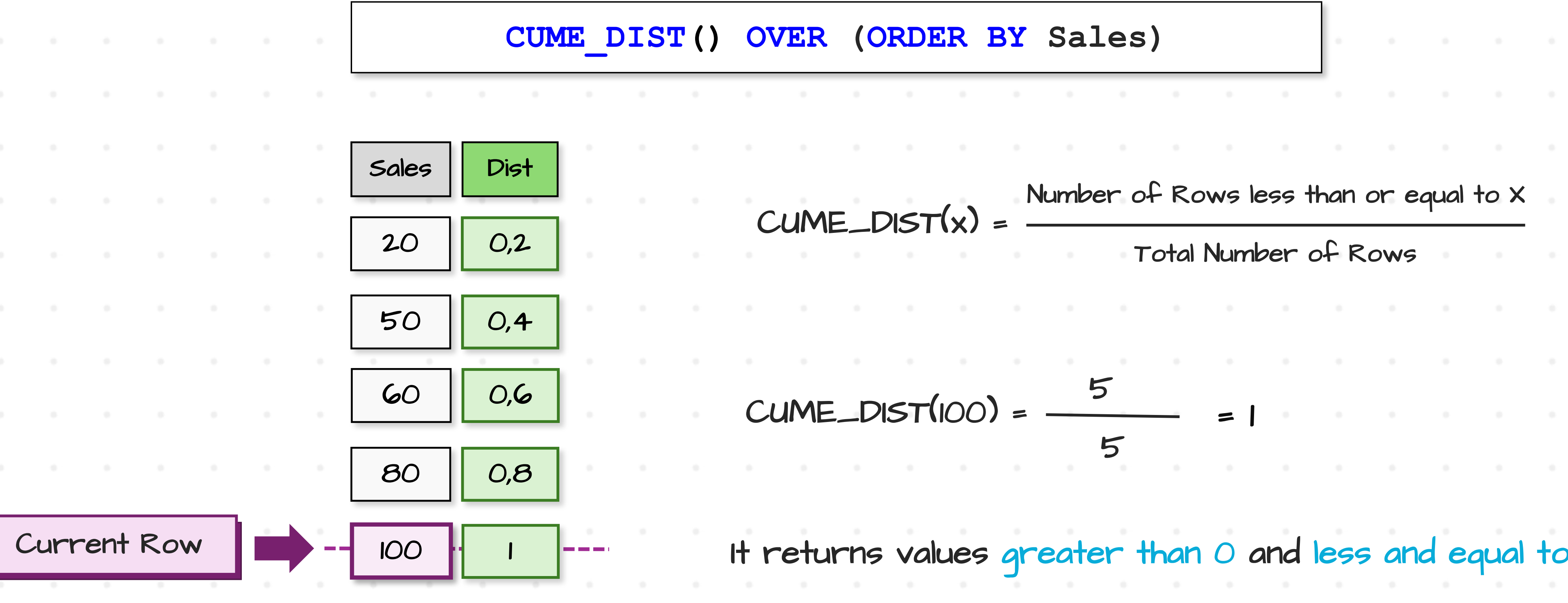
```
CUME_DIST() OVER (ORDER BY Sales)
```



CUME_DIST

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

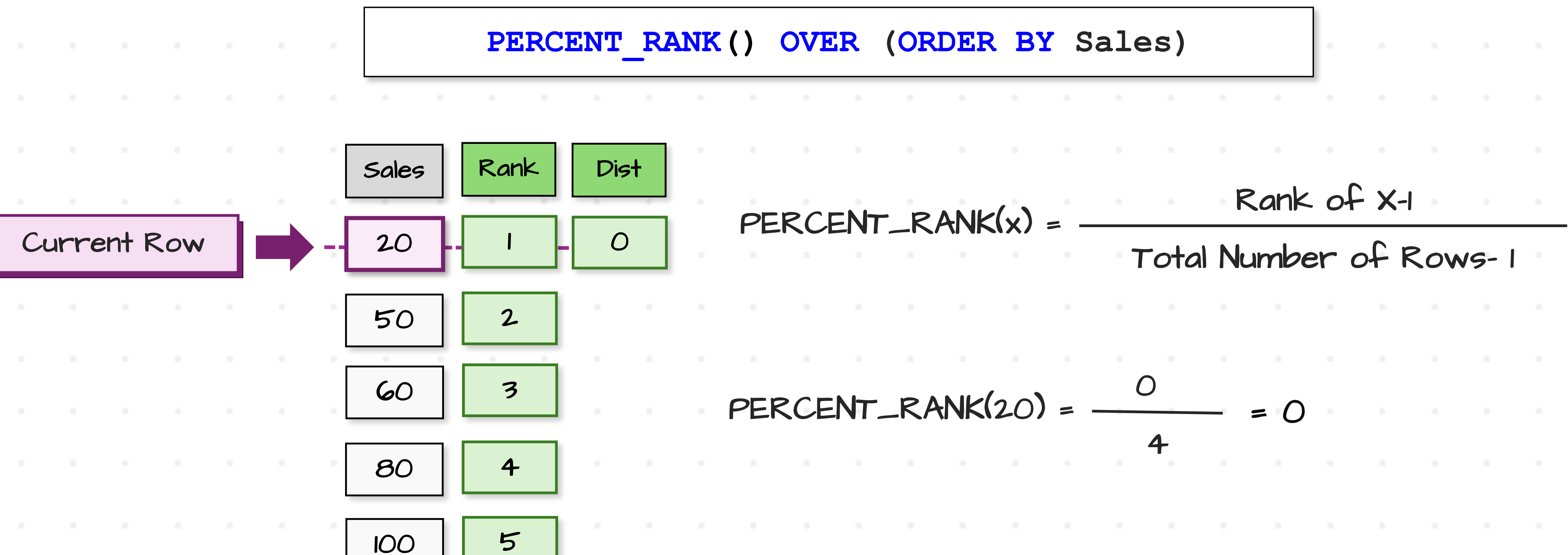
```
CUME_DIST() OVER (ORDER BY Sales)
```



PERCENT_RANK

Returns the percentile ranking number of a row

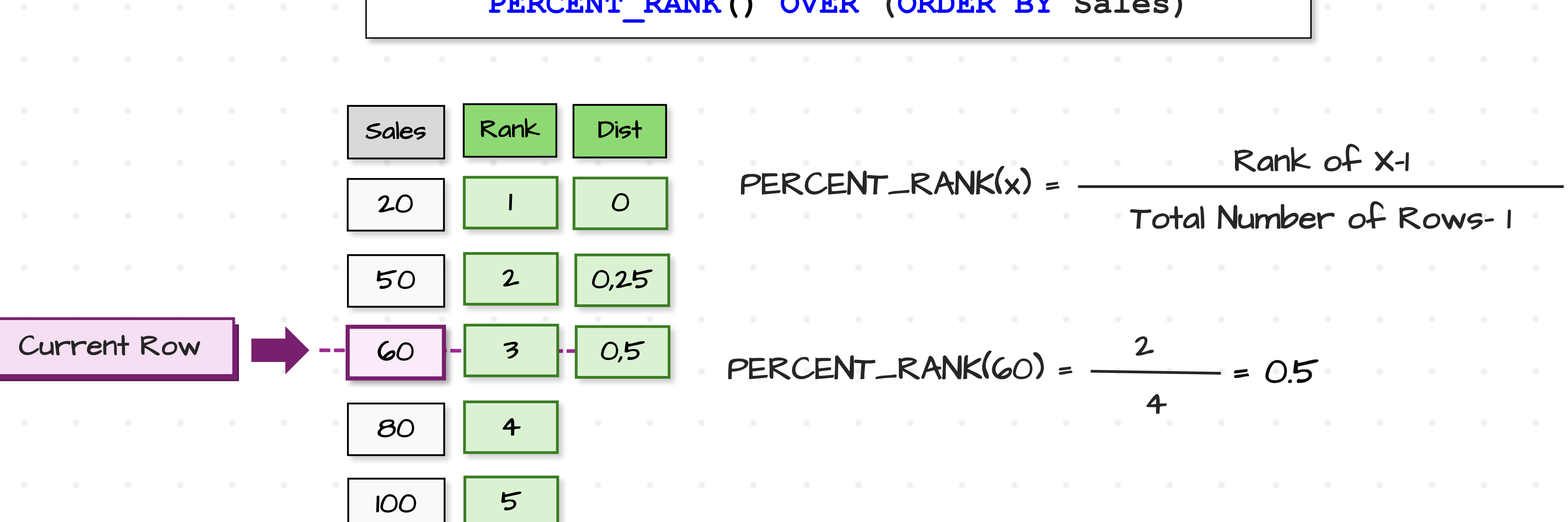
```
PERCENT_RANK() OVER (ORDER BY Sales)
```



PERCENT_RANK

Returns the percentile ranking number of a row

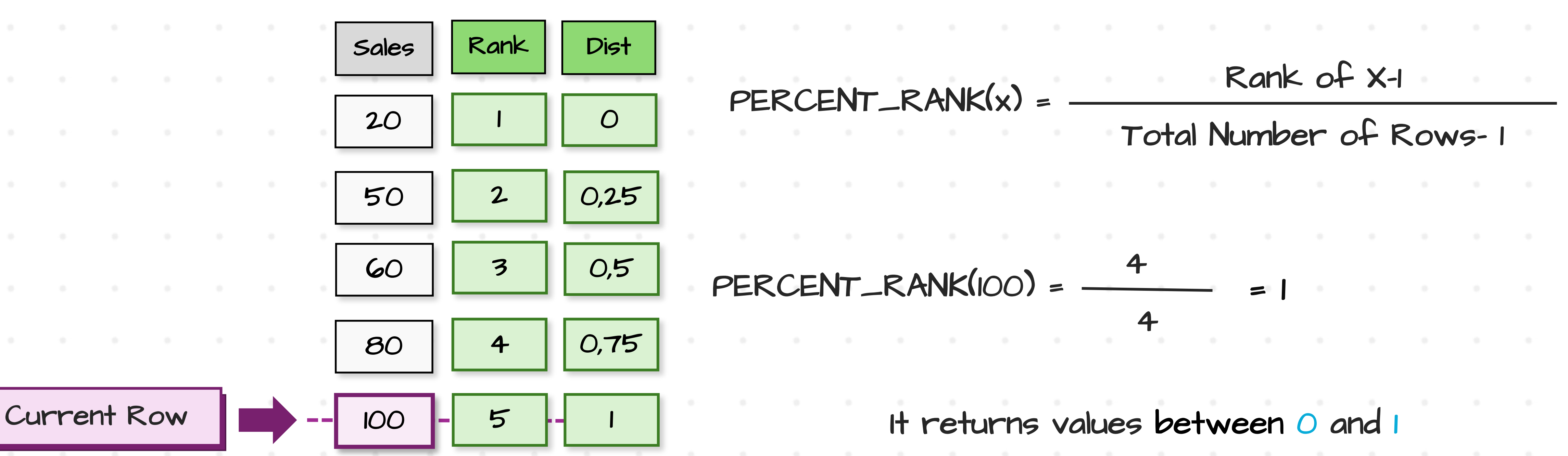
```
PERCENT_RANK() OVER (ORDER BY Sales)
```



PERCENT_RANK

Returns the percentile ranking number of a row

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



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

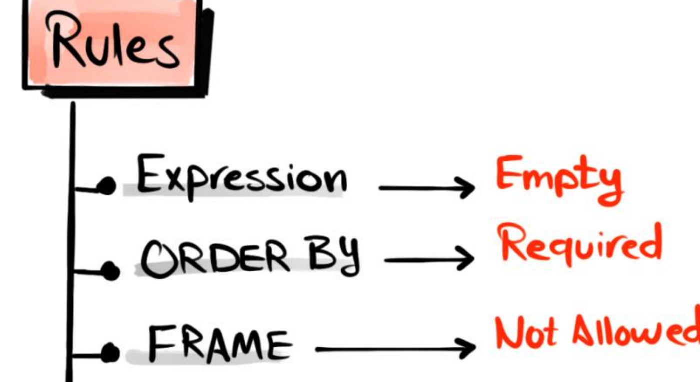
Lowest Position

Highest Position

Current Row

WINDOW RANK FUNCTIONS

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



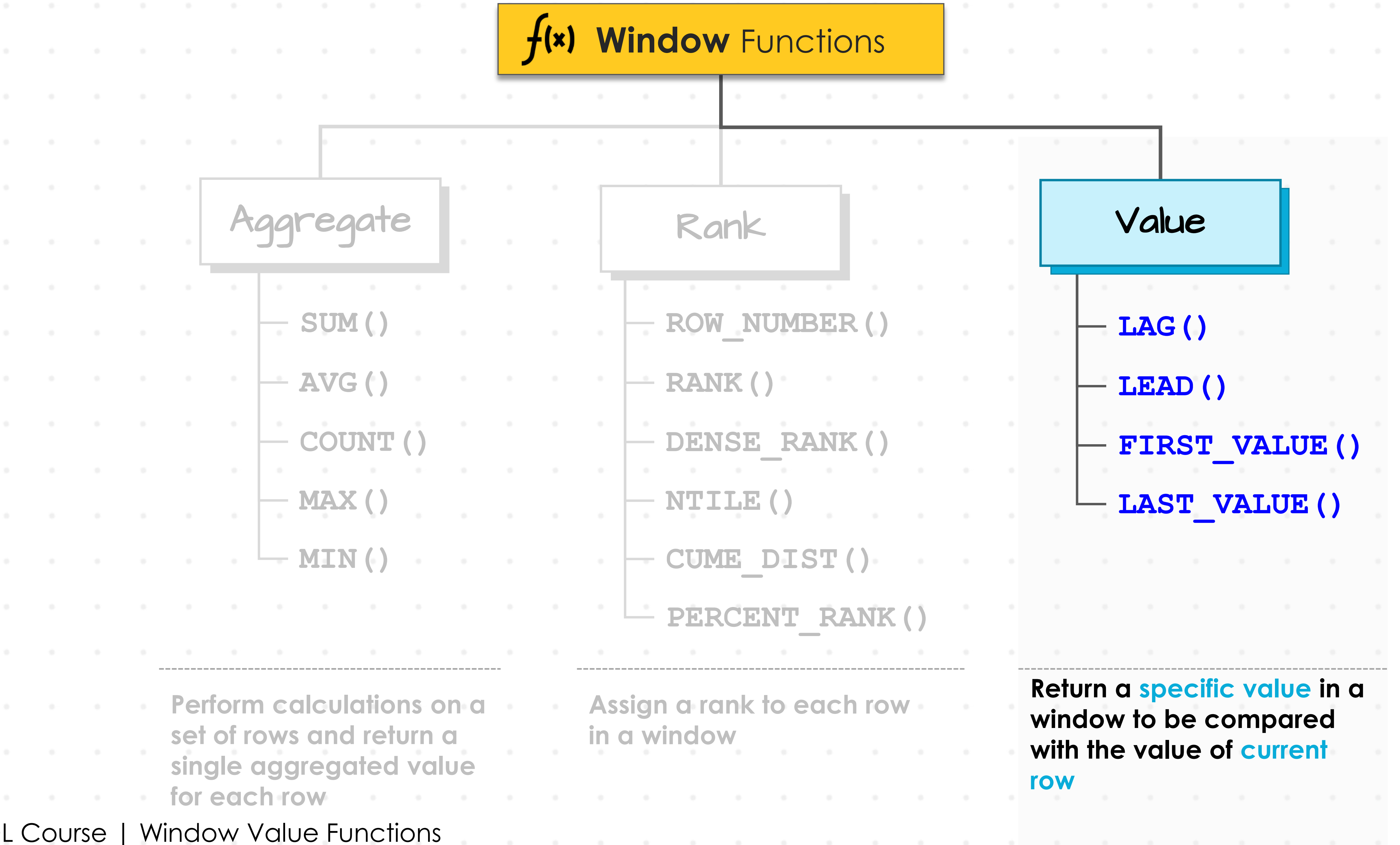
Use Cases

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

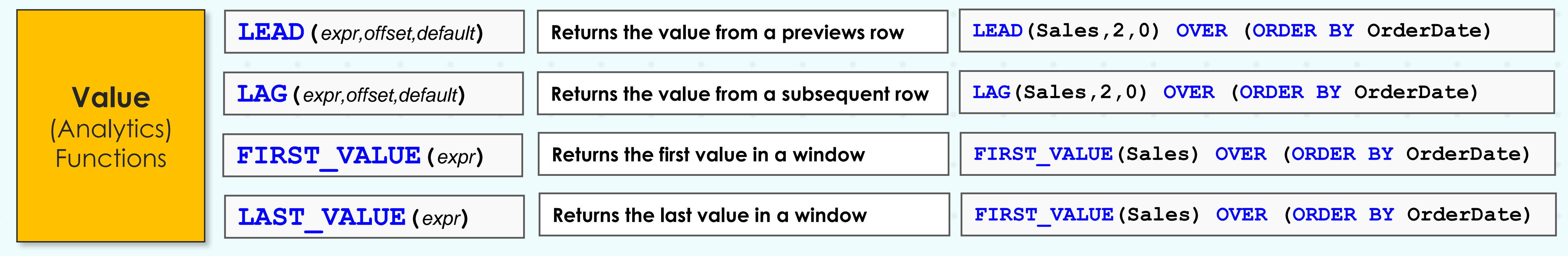
WINDOW VALUE

FUNCTIONS

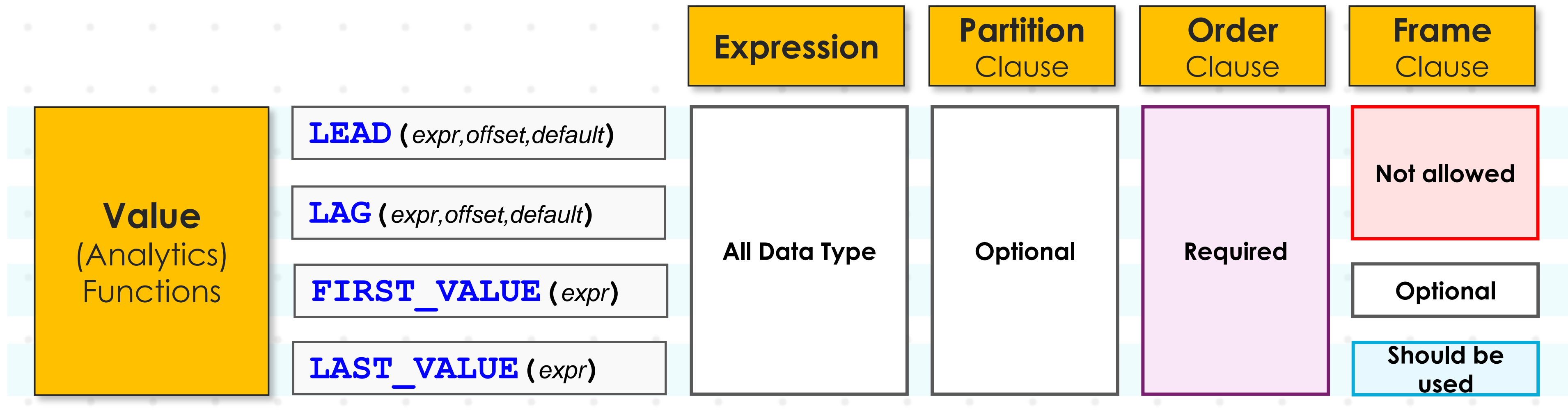
Baraa Khatib Salkini
SQL Course | Window Value Functions

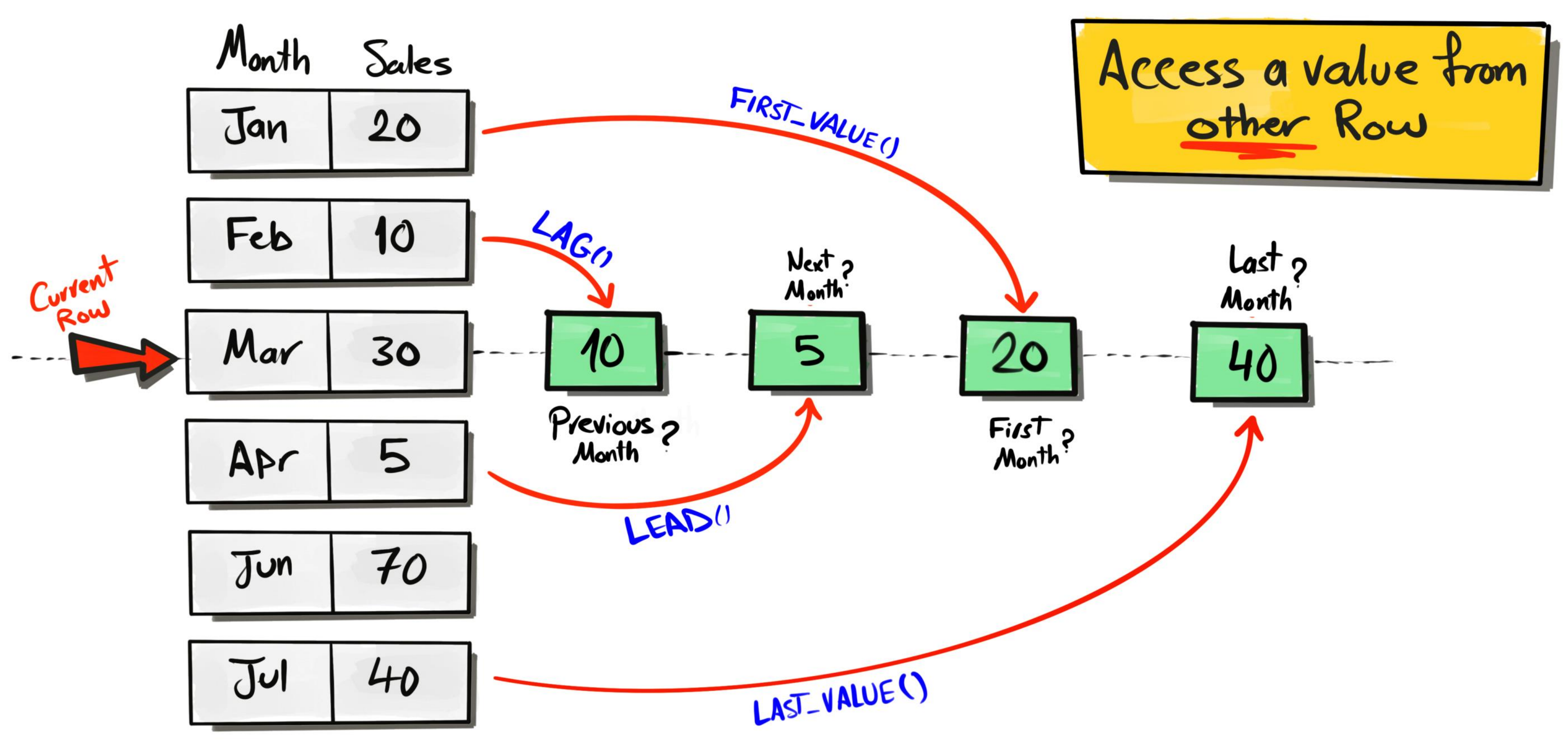

f(x) Window Functions
Aggregate
SUM()
AVG()
COUNT()
MAX()
MIN()
Perform calculations on a set of rows and return a single aggregated value for each row
Rank
ROW_NUMBER()
RANK()
DENSE_RANK()
NTILE()
CUME_DIST()
PERCENT_RANK()
Assign a rank to each row in a window
Value
LAG()
LEAD()
FIRST_VALUE()
LAST_VALUE()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

Value Functions



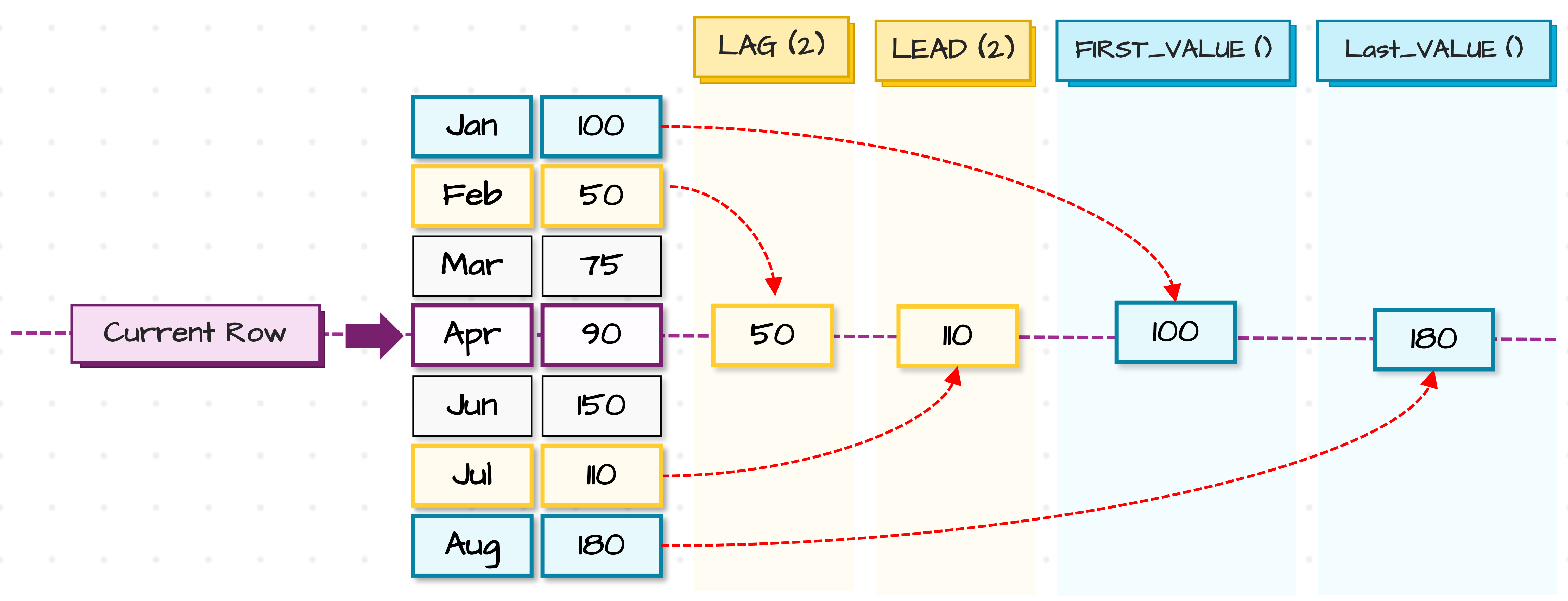
Value Functions





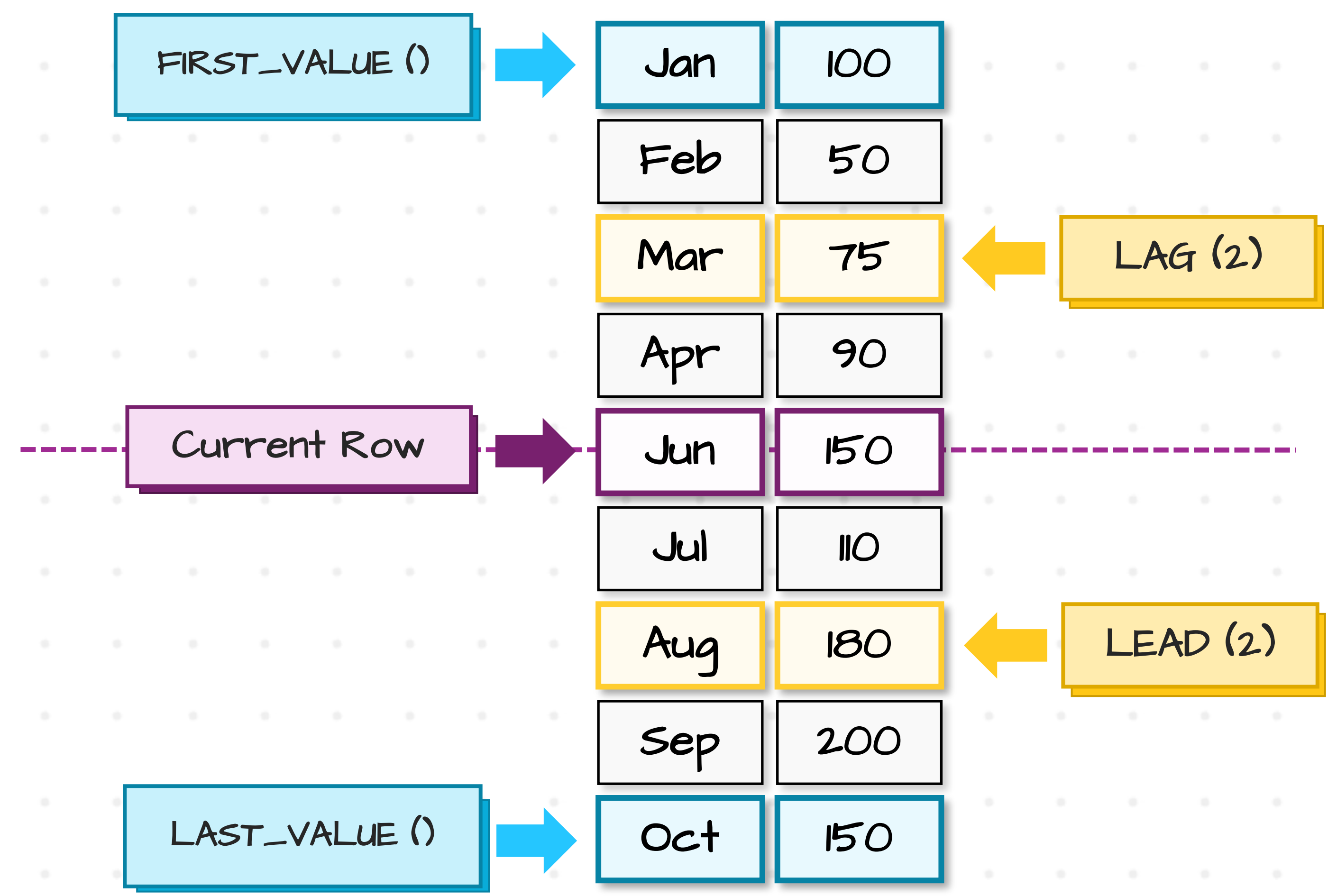
Access a value from other Row
Month
Sales
Jan
20
Feb
10
Mar
30
Apr
5
Jun
70
Jul
40
Current Row
LAG()
FIRST_VALUE()
LEAD()
LAST_VALUE()
10
Previous Month?
5
Next Month?
20
First Month?
40
Last Month?

Value Functions

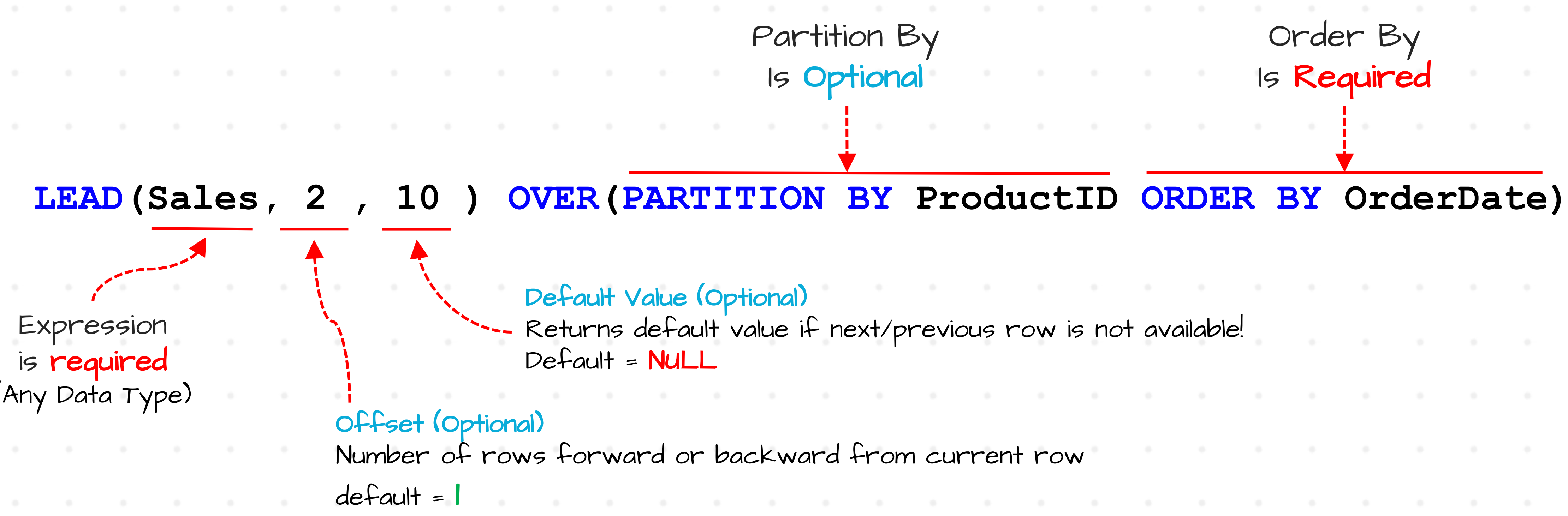


Value Functions

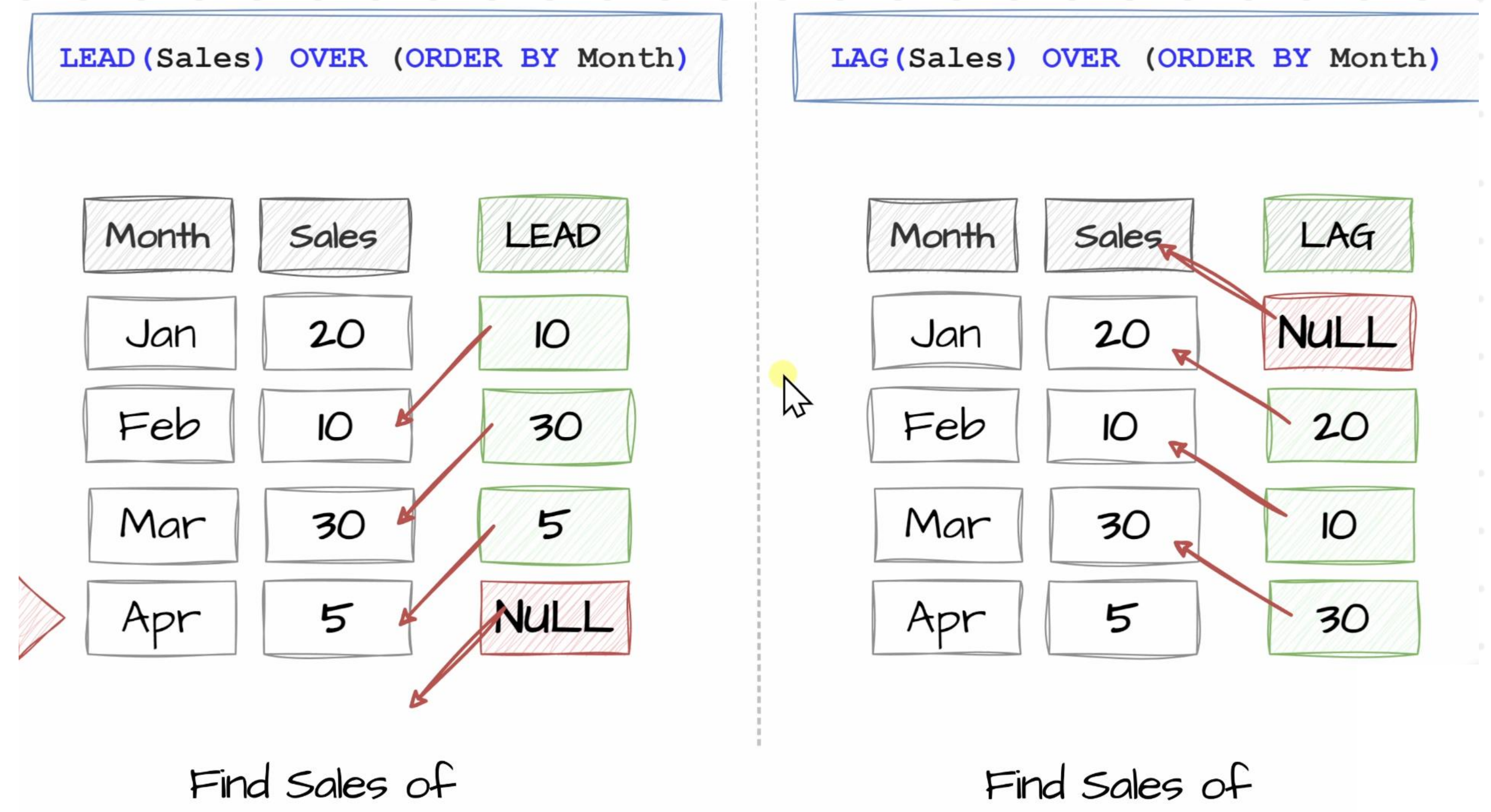
Return a specific value in a window to be compared with the value of current row



LEAD & LAG

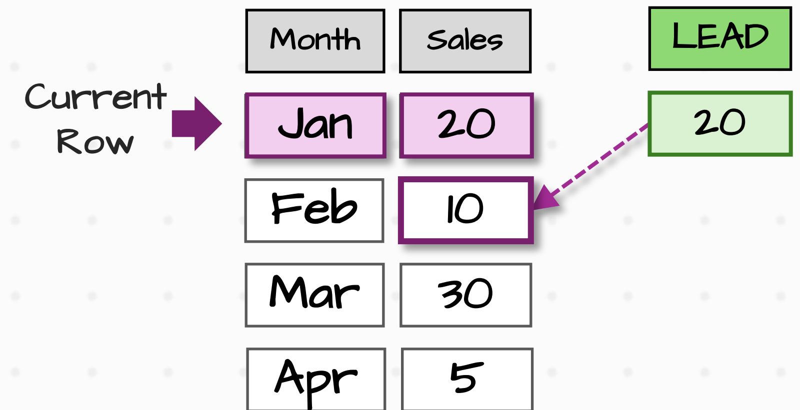


LEAD & LAG



Access Next Row

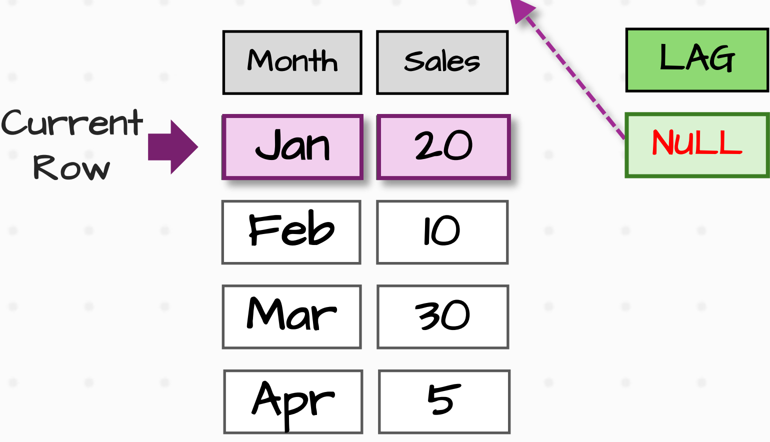
```
LEAD(Sales) OVER( ORDER BY Month)
```



Access Previous Row

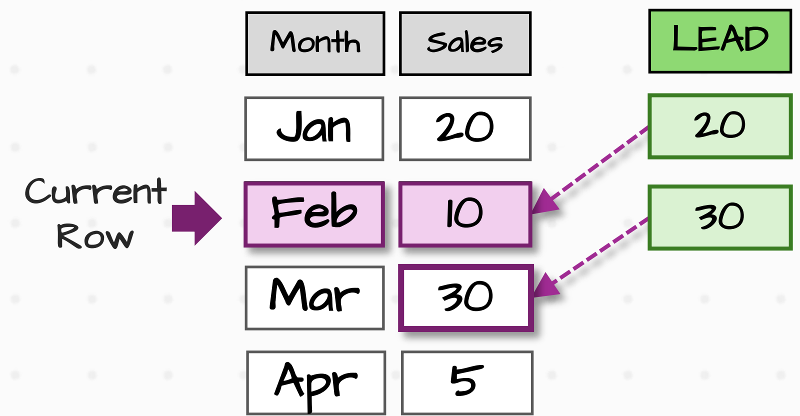
```
LEAD(Sales) OVER( ORDER BY Month)
```

First Row has No Previous Row !



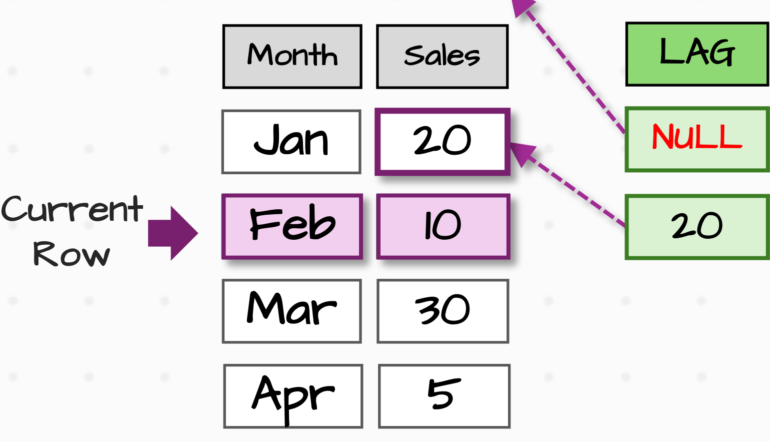
Access Next Row

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



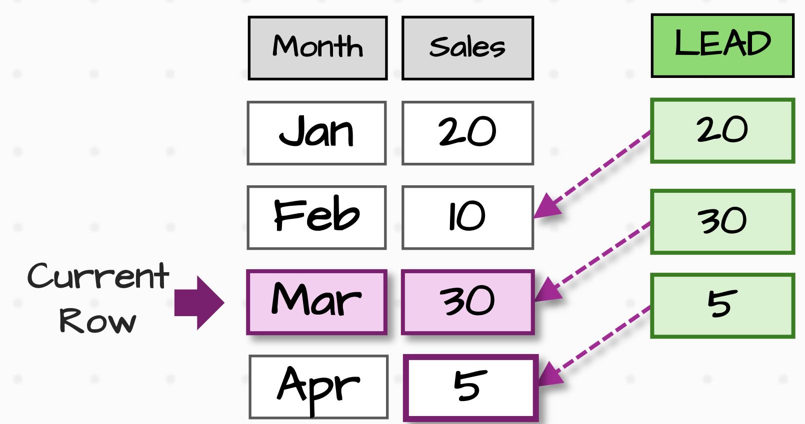
Access Previous Row

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



Access Next Row

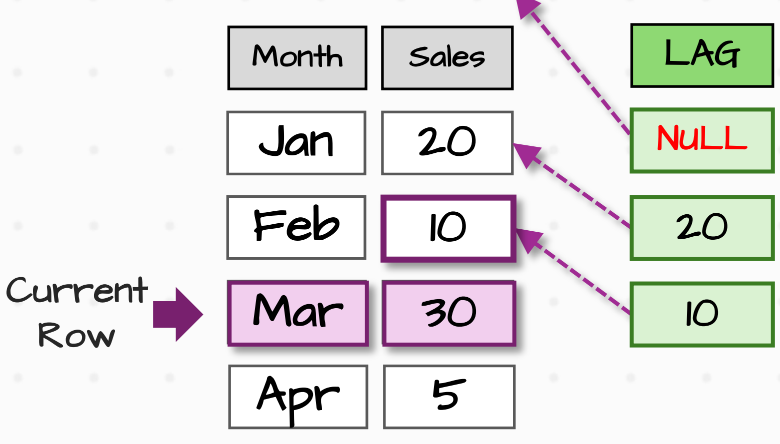
```
LEAD(Sales) OVER( ORDER BY Month)
```



Access Previous Row

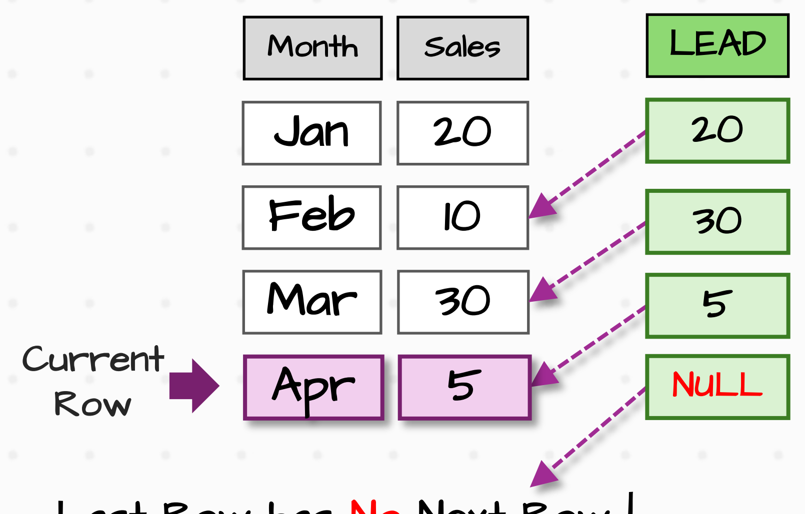
```
LEAD(Sales) OVER( ORDER BY Month)
```

First Row has No Previous Row !



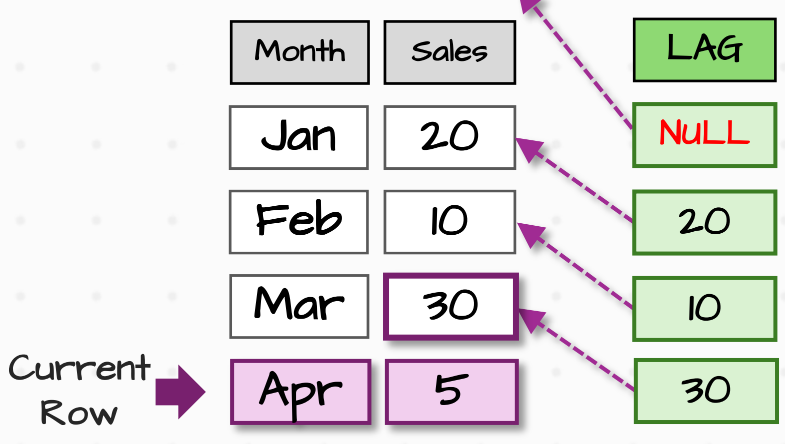
Access Next Row

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



Access Previous Row

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



LEAD & LAG

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

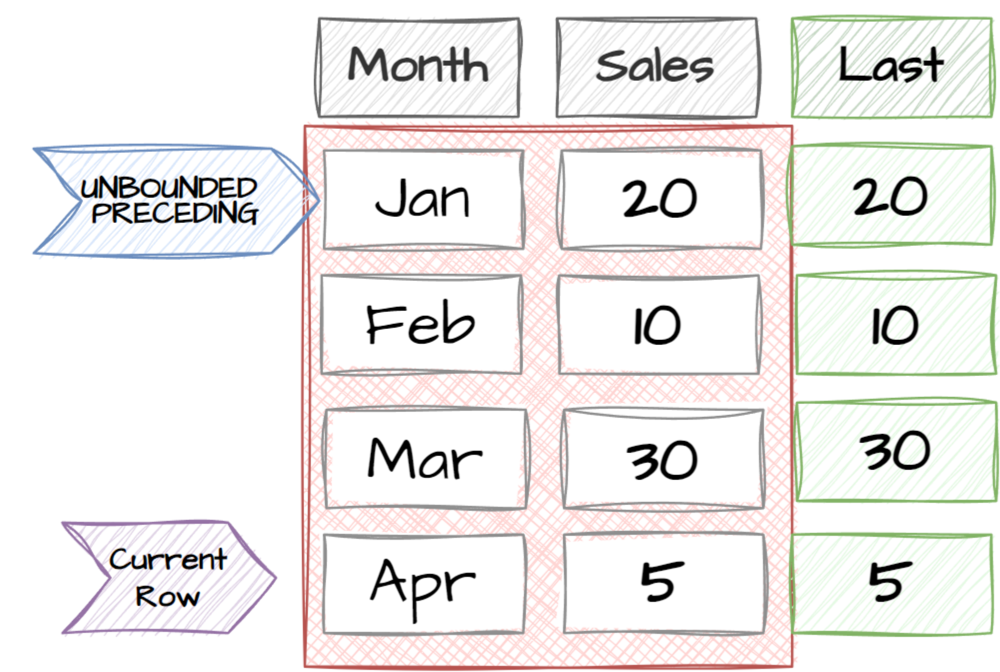
```sql
FIRST_VALUE(Sales) OVER (ORDER BY Month)
```

Month	Sales	First
Jan	20	20
Feb	10	20
Mar	30	20
Apr	5	20

UNBOUNDED PRECEDING → Jan

Current Row → Apr

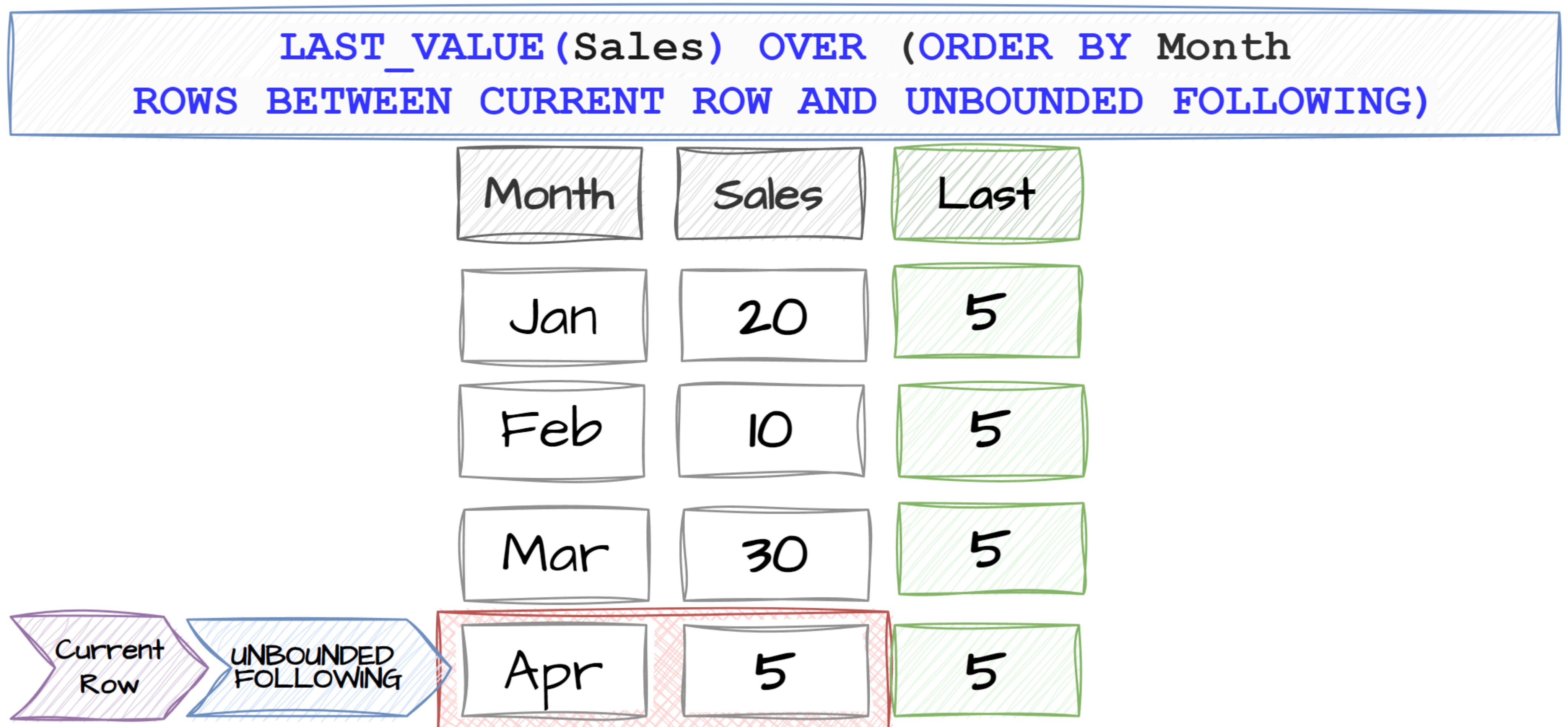
```sql
LAST_VALUE(Sales) OVER (ORDER BY Month)
```



Default

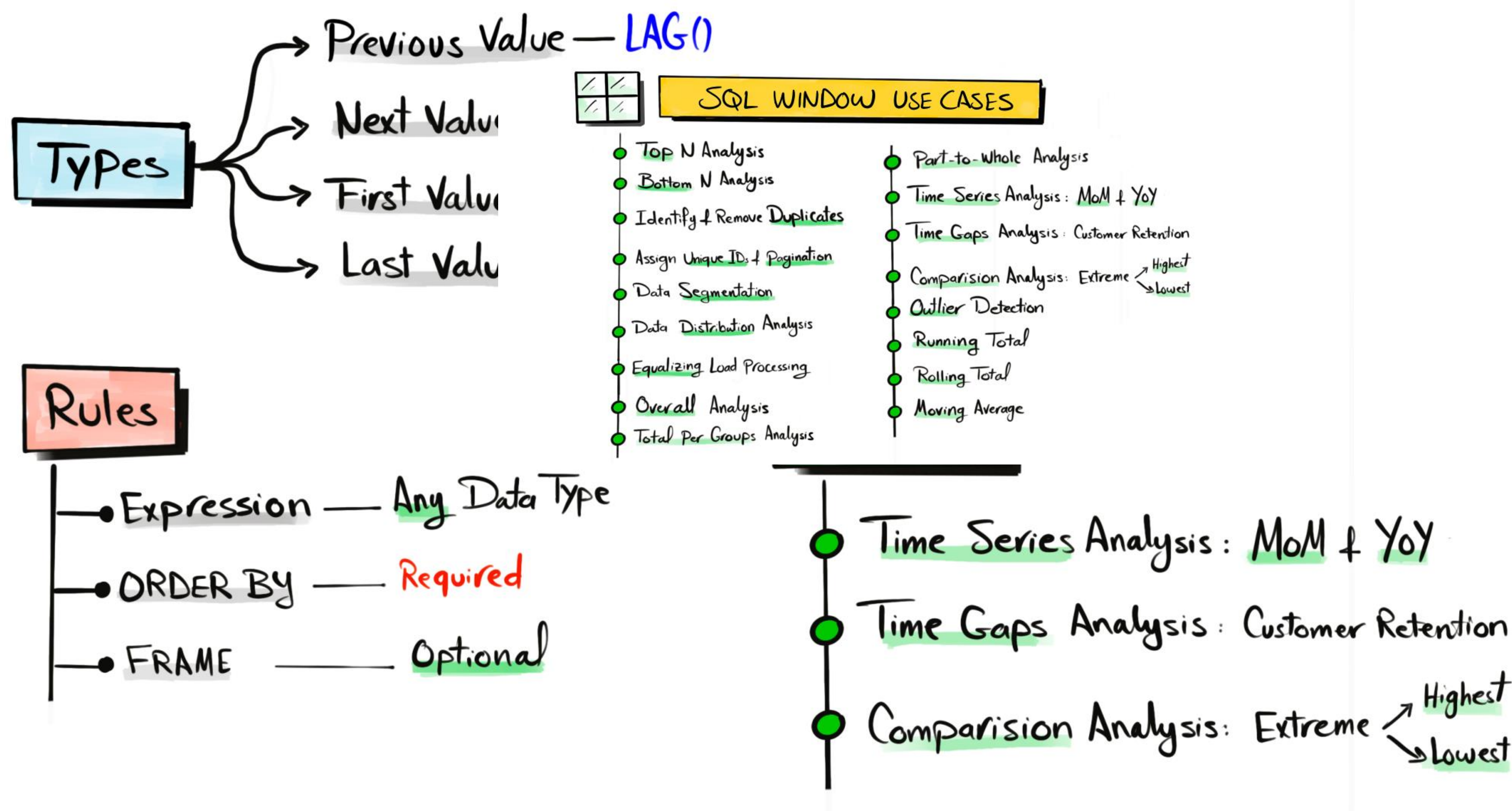
```sql
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

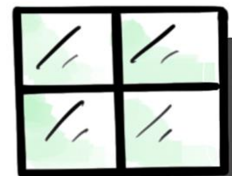
FIRST & LAST



WINDOW VALUE (ANALYTICAL) FUNCTIONS

Allow Access specific Value from another Row





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