SQL 07- Window functions 2.

Recap:

over

partition by

Ranking,

Aggregate - Sum, Count

Ntile

Agenda:

Aggregate -- Over clause.

Lag, lead.

nth value

First value

last value.

```
Select *,
Sum(sales) over (order by date)
```

Emp.	Date	Sales
A	1	10
A	2	5
B	3	15
B	4	10
B	5.	20.

Emp.	Date	Sales	Sum(Sales)
A	1	10	10
A	2	5	15
B	3	15	30
B	4	10	40
B	5.	20	60

Window frames:

In agg. functions when order by is used then the window frame come into picture.

Emp.	Date	Sales	Sum(Sales)
A	1	10	10
A	2	5	15
B	3	15	30
B	4	10	40
B	5.	20	60

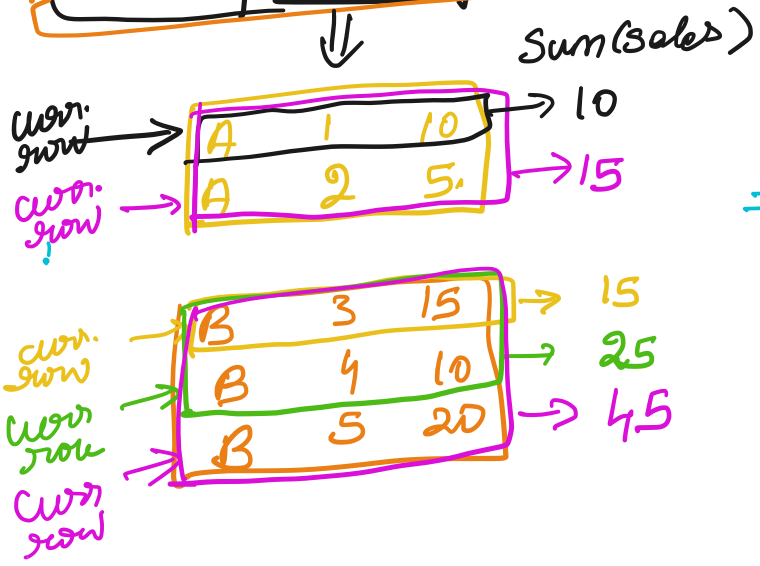
This is called

frame.

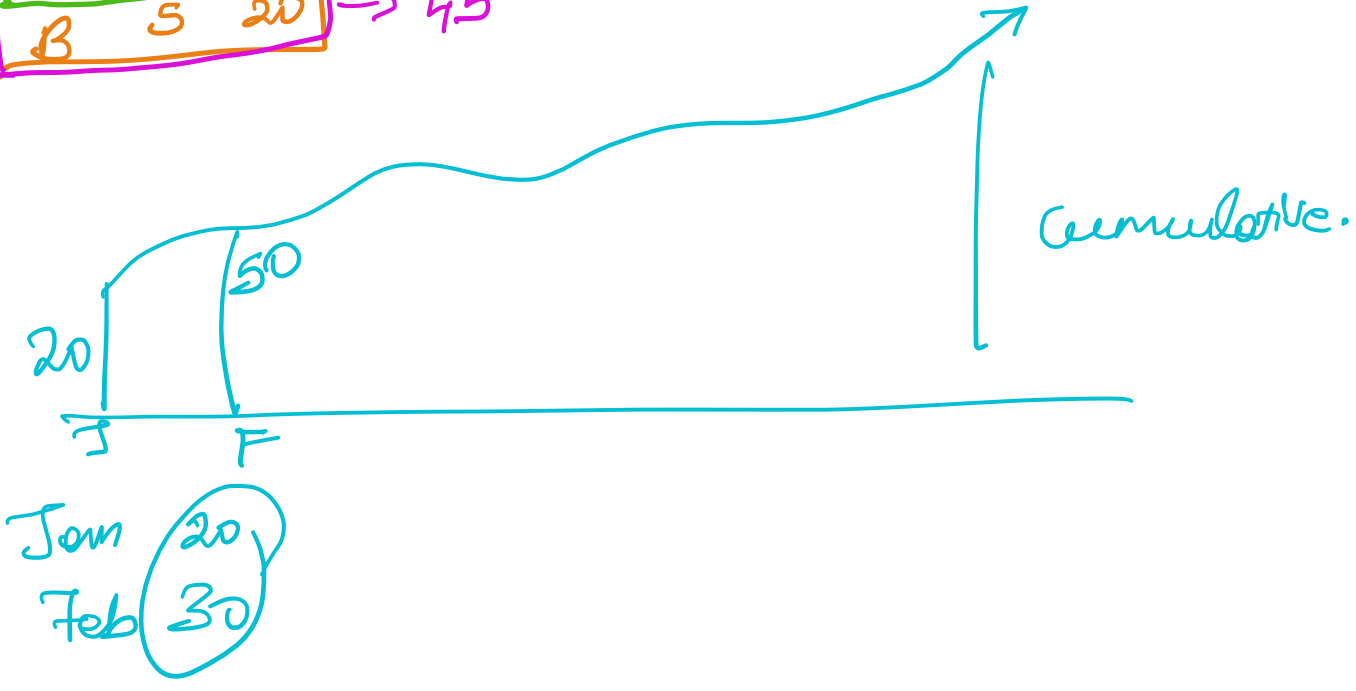
When the execution happen at the curr. row how many rows it has access to above it or below it is called window frames.

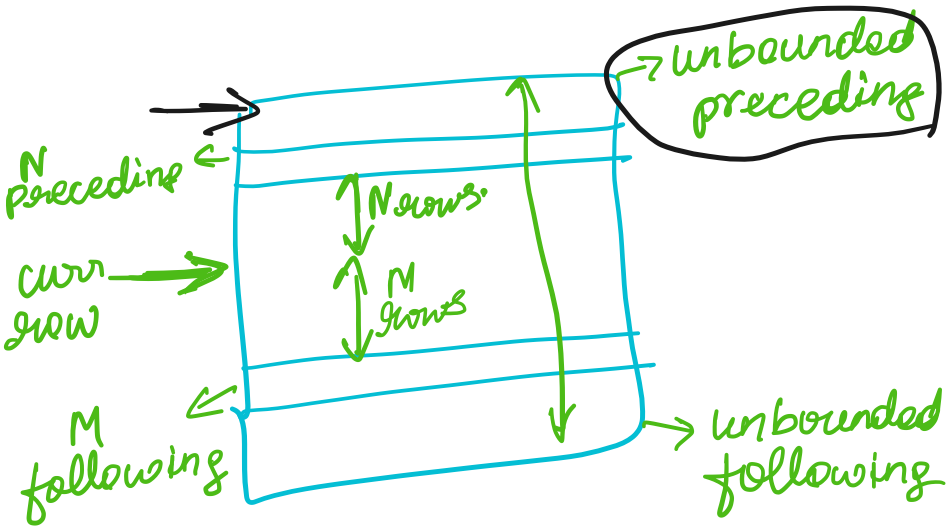
Emp.	Date	Sales
A	1	10
A	2	5
B	3	15
B	4	10
B	5.	20.

Sum(sales) over (partition by Emp order by date)



A	1	10	10
A	2	5	15
B	3	15	15
B	4	10	25
B	5	20	45





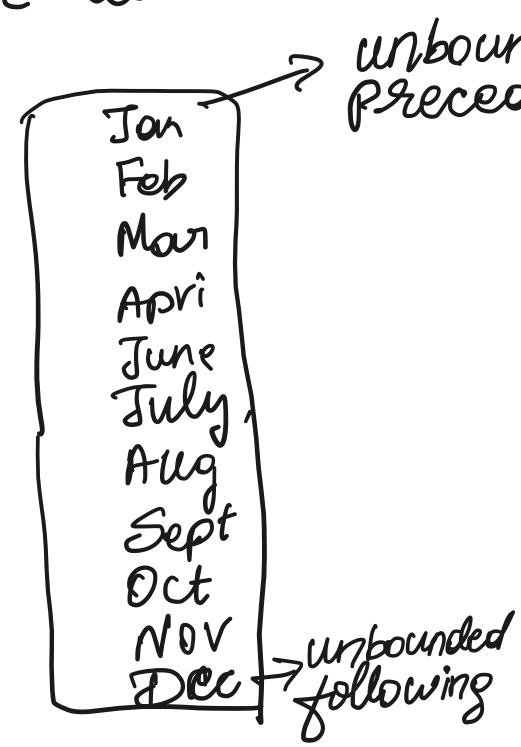
sum(sales) over (order by date) → cummulative sum.

is same as.

sum(sales) over (order by date range between unbounded preceding and current row)

The 1st row is called the unbounded preceding

The last row is called the unbounded following

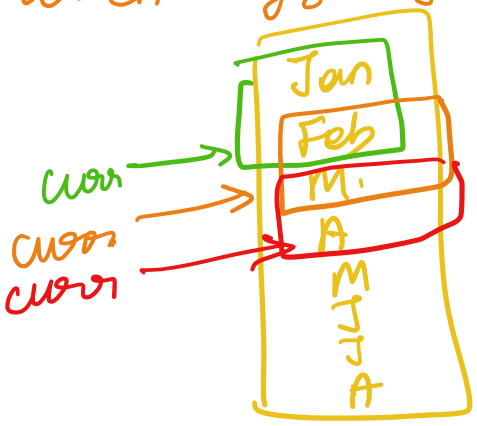


order by

default : Range between unbounded preceding and current row.

Range: Logically looks at the order by column and aggregates it.

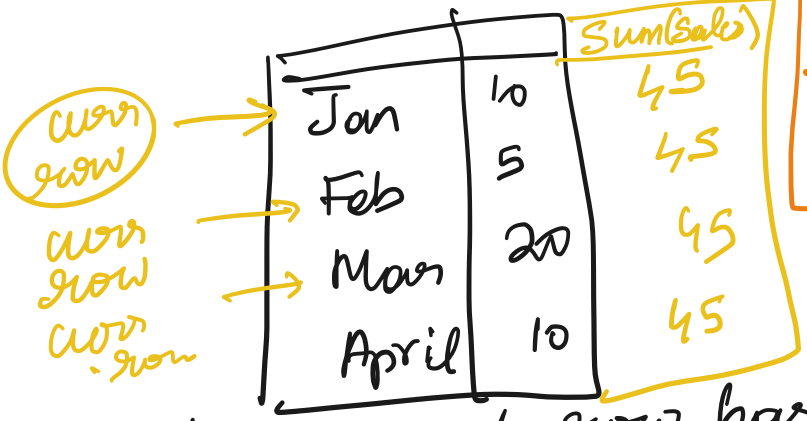
Rows: Considers each row independently and does not logically look at the order by column when aggregating.



rows 1 preceding and current row

⇓

has access to one row above and the current row.



sum(sales) over(order by date range between unbounded preceding and unbounded following

⇓

the current row has access to all rows above it and all rows below it.

Where is it used? Find the contribution of each month to the overall sales of the year

sum(sales) over(order by date range between unbounded preceding and unbounded following

is same as

sum(sales) over()

⇓

The entire table is considered 1 partition and the entire table is considered 1 frame

sum(sales) over(partition by vendor_id order by date range between unbounded preceding and unbounded following

is same as

sum(sales) over(partition by vendor_id)

⇓

Each partition is considered 1 frame.

Vendor	Date	Sales
V1	Jan 1	5
V1	Jan 2	10
V2	Jan 3	25
V2	Jan 4	15
V2	Jan 5	40

⇓

Vendor	Date	Sales	Sum (Sales)
V1	Jan 1	5	95
V1	Jan 2	10	95
V2	Jan 3	25	95
V2	Jan 4	15	95
V2	Jan 5	40	95.

Sum (sales) over (order by date)

Vendor	Date	Sales	Sum (Sales)
V1	Jan 1	5	5
V1	Jan 2	10	15
V2	Jan 3	25	40
V2	Jan 4	15	55
V2	Jan 5	40	95

Vendor	Date	Sales
V1	Jan 1	5
V1	Jan 2	10
V2	Jan 3	25
V2	Jan 4	15
V2	Jan 5	40

⇓

Vendor	Date	Sales	Sum Sales)
V1	Jan 1	5	15
V2	Jan 2	10	15
V3	Jan 3	25	80
V4	Jan 4	15	80
V5	Jan 5	40	80

Sum (Sales) over (partition by vendor_id order by date)

Vendor	Date	Sales	Sum Sales)
V1	Jan 1	5	5
V2	Jan 2	10	15
V3	Jan 3	25	25
V4	Jan 4	15	40
V5	Jan 5	40	80

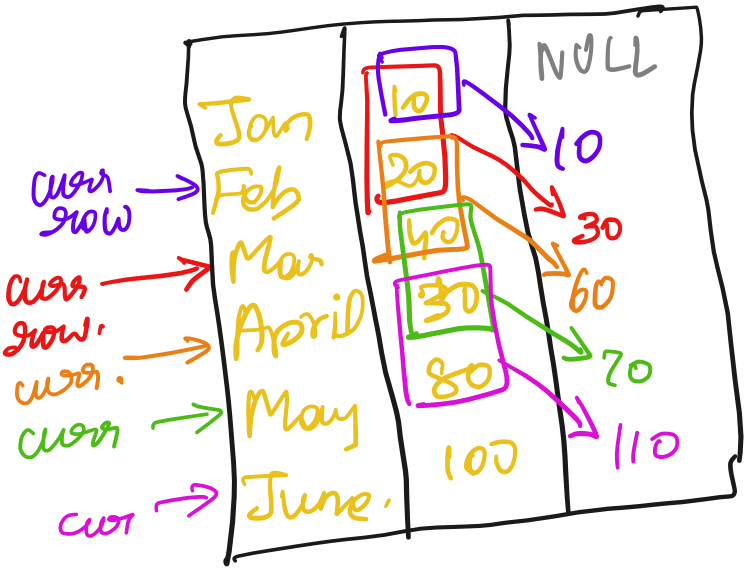
2022.

	Sales	Sum (Sales)
curr row → Jan	10	15.
curr row → Feb	5	35
curr row → Mar	20	35
curr → April	10	10
curr row → May	40	50

sum (sales) over (order by date rows between 1 preceding and 1 following)

→ in stock markets to find the moving

average.



Sum (sales)
over (order by date
rows between
2 preceding and
1 preceding

Lag To get the previous value. N steps before

lead : to get next values, N steps ahead

Vend	Date	Both numbers	Prev.
V1	Jan1	B1	NULL.
V1	Jan2	B4	B1
V2	Mar1	B18	NULL
V2	Mar2	B12	B18
V2.	Mar3	B3	B2

lag ( )

If the current days is better the previous day

Date	Sales
	5

Date	Sales
	5

1	10
1	15
2	10
2	100
3	20
3	30
3	10.

1	10
1	15
2	10
2	100
3	20
3	30
3	10.

date, sum(sales)
group by (date)

Date	Sum(sales.)	lag.
1	20	20
2	110	110
3	60	

Doubt session.

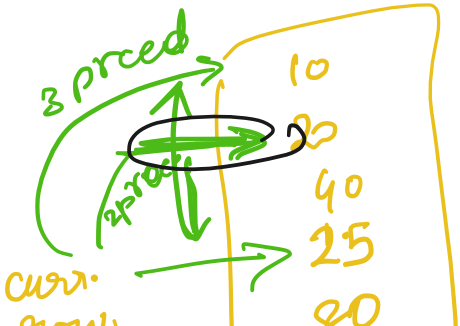
lag(sales) over(partition by V.id order by M.D)

V.id	M.D
A	2
A	1
B	2
B	1
B	3.
C	

M.D.

A	1	
A	2	1
B	1	
B	2	1
B	3.	2.

3 preceding and 2 preced.



2 preding

row

80

100.

and (3p)recedens.

3 - 2 = 1 - row

2 - 3 = -1