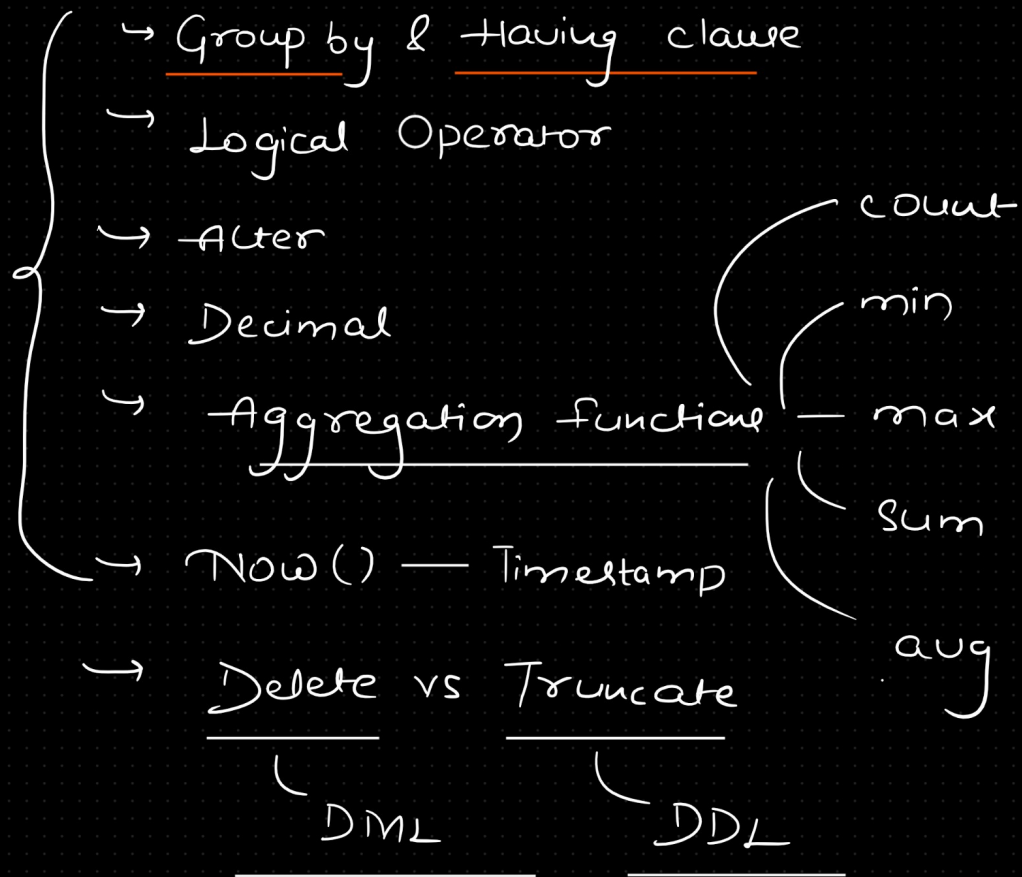


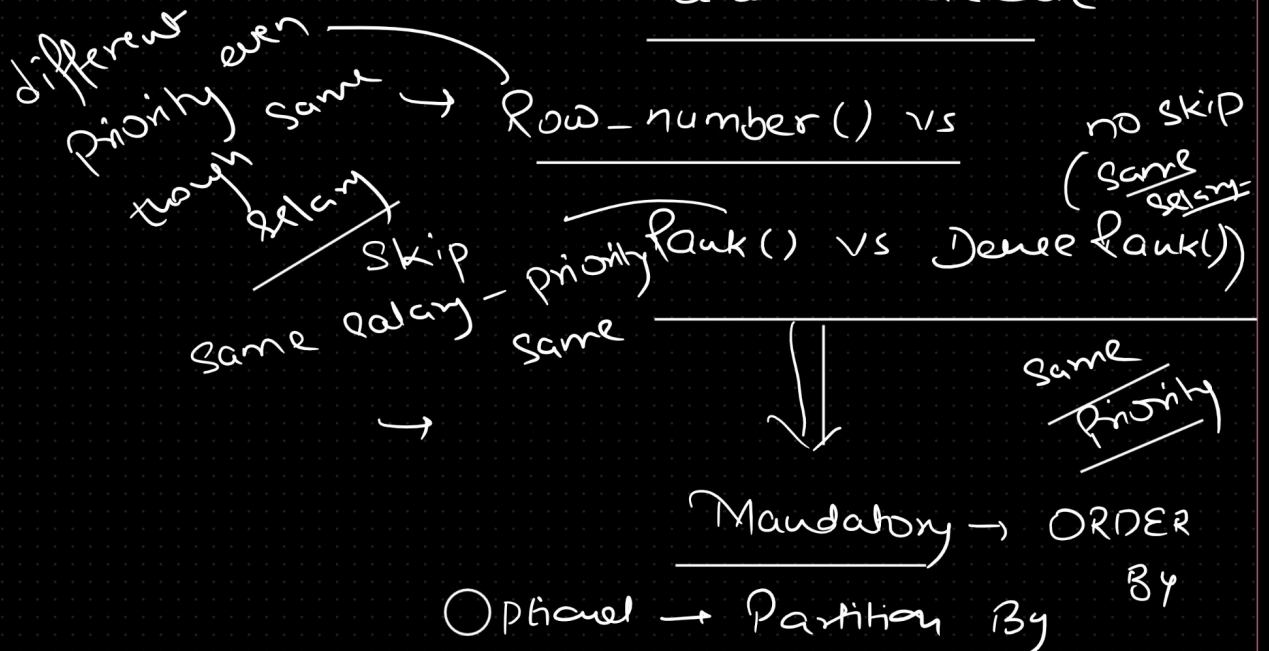
## Window functions



## Day-4

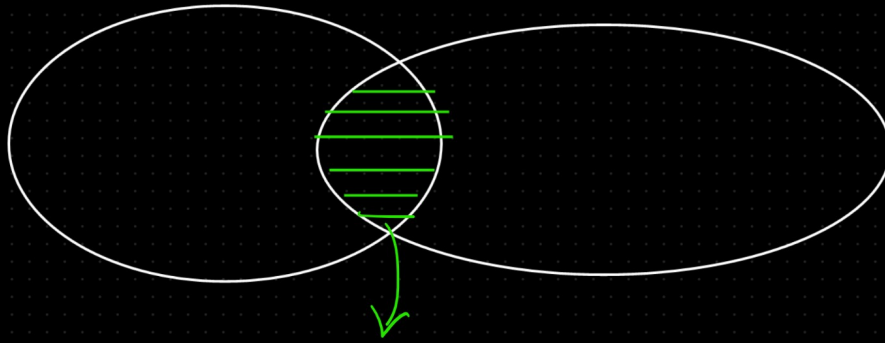
→ Inner Join

→ OVER & PARTITION BY clause  
and its use case



## Inner Join (JOIN)

→ two table



Output

```
SELECT FirstName, LastName, Employee.Location, Total, Avg_Salary
FROM Employee
JOIN
(SELECT Location, COUNT(Location) as Total, AVG(Salary) as Avg_Salary
FROM Employee
GROUP BY Location) AS temp
ON Employee.Location = temp.Location
```

Employee

EID	FirstName	LastName	Age	Salary	Location
1	Priya	Bhatt	27	200000	Bengaluru
2	Akhil	George	26	100000	Gurugram
3	Kaneesha	Brownlee	27	300000	London
5	Naga	Sai	30	150000	Bengaluru
6	Prabhutva	Kakkar	35	70000	Noida
7	Yashswi	Gandhak	29	90000	Delhi

temp

AS

Location	Total	Avg_Salary
Bengaluru	2	175000.0000
Gurugram	1	100000.0000
London	1	300000.0000
Noida	1	70000.0000
Delhi	1	90000.0000

JOIN (INNER JOIN)

→ Computationally expensive

FirstName	LastName	Location	Total	Avg salary
Priya	Bhatt	Bengaluru	2	175000
Akhil	George	Gurugram	1	100000
Kaneesha	Brownlee	London	1	300000
Naga	Sai	Bengl.	2	175000

Prabhutva	Kakkar	Noida	1	70000
Yashswi	Gandhak	Delhi	1	90000

## PARTITION BY

→ Group By

(Allows non-aggregation  
column to display)

Display non-  
aggregated  
columns  
→ Grouping

SELECT FirstName, LastName, Location,  
COUNT(Location) OVER (PARTITION BY Location) AS Total,  
AVG(Salary) OVER (PARTITION BY Location) AS AVG\_Salary  
FROM Employee

	EID	FirstName	LastName	Age	Salary	Location
1	1	Priya	Bhatt	27	200000	Bengaluru
2	2	Akhil	George	26	100000	Gurugram
3	3	Kaneesha	Brownlee	27	300000	London
5	5	Naga	Sai	30	150000	Bengaluru
6	6	Prabhutva	Kakkar	35	70000	Noida
7	7	Yashswi	Gandhak	29	90000	Delhi

Location	Total
Beng.	2
Gurugram	1
London	1
Noida	1
Delhi	1

→ Location Avg Salary

Ben ——— 175000

Gur ——— 100000

London ——— 300000

Noida ——— 70000

Delhi ——— 90000

✓ ROW-NUMBER()

└ map the value with

numbers

Mandatory

(Sorted data)

└ ORDER BY