■ Ranking Functions in SQL – Full Explanation

■ Ranking Functions Recap

- ROW_NUMBER() → assigns a unique number to each row.
- RANK() → assigns rank with gaps when values tie.
- DENSE_RANK() → assigns rank without gaps when values tie.

■ But how are these ranks decided?

■ On What Basis Is Rank Given?

Ranking is always based on the ORDER BY inside the OVER() clause.

Example:

```
SELECT employee_id, salary, RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM employees;
```

Explanation:

- ORDER BY salary DESC → sorts employees by highest salary first.
- The rank is then assigned in that sorted order.
- If two employees have the same salary:
- - RANK() → gives them the same rank but skips the next number (gap).
- - DENSE_RANK() → gives them the same rank without skipping.
- - ROW_NUMBER() → still forces a unique number for each row.

■ Is Sorting Required?

■ Yes. Without ORDER BY, the rank would be meaningless, because there's no basis to assign it. Think of ranking like a sports leaderboard → you must sort by score/time before assigning positions.

■ Example with Tied Values

```
SELECT employee_id, salary, ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num,
RANK() OVER (ORDER BY salary DESC) AS rank_num, DENSE_RANK() OVER (ORDER BY
salary DESC) AS dense_rank_num FROM employees;
```

employee_id	salary	ROW_NUMBER	RANK	DENSE_RANK
E1	90000	1	1	1
E2	90000	2	1	1
E3	85000	3	3	2
E4	80000	4	4	3

■ So in interview answer:

"Ranking functions depend on the ORDER BY clause inside the window function. You must specify the column(s) to sort on, because rank is assigned in that order. Without ORDER BY, ranking has no meaning."