■ Window Functions in SQL – Full Interview Answer

■ 1. Definition

A window function performs a calculation across a set of rows related to the current row, without collapsing the rows into a single result.

They are different from aggregate functions (like SUM(), AVG()) because they do not group rows into one. Instead, they return a value for every row while still considering other rows.

■ Key Points to Mention in Interview:

- Window functions use the OVER() clause.
- They don't reduce the number of rows, unlike GROUP BY.
- Useful for analytics: rankings, moving averages, running totals.
- Aggregate functions → Group multiple rows → return one row per group.
- Window functions → Work on a group but still return row-wise results.

■ 2. Syntax

```
SELECT column_names,
    function_name (expression) OVER (
        PARTITION BY column_name
        ORDER BY column_name
        ROWS/RANGE specification
    ) AS alias_name
FROM table_name;
```

■ Explanation of each part:

- function_name() → The window function (e.g., RANK, SUM, AVG, ROW_NUMBER).
- OVER() → Defines the "window" of rows to look at.
- PARTITION BY → Divides rows into groups (like GROUP BY, but rows are not collapsed).
- ORDER BY → Defines the order of rows in each partition (important for ranking, running totals).
- ROWS/RANGE → Optional, specifies how many rows to include (e.g., last 3 rows).

■ 3. Types of Window Functions

(a) Ranking Functions

These functions assign ranks or row numbers based on ordering.

- **ROW_NUMBER()** → Assigns a unique sequence number to each row.
- **RANK()** → Assigns rank, but leaves gaps if there are ties.
- **DENSE_RANK()** → Assigns rank without gaps (no skipping of numbers).

Example:

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

■ Output: Employees ranked by salary (highest salary = rank 1).

(b) Aggregate Window Functions

These functions perform aggregate operations but keep row details.

Examples: SUM(), AVG(), COUNT(), MIN(), MAX().

Example:

```
SELECT employee_id, department_id, salary,
    AVG(salary) OVER (PARTITION BY department_id) AS dept_avg_salary
FROM employees;
```

■ Output: Each employee row + department average salary.

(c) Value Functions

These functions help compare the current row with other rows.
- **LAG()** → Accesses the previous row's value.
- **LEAD()** → Accesses the next row's value.

Example:

```
SELECT order_id, order_date,
    LAG(order_date) OVER (ORDER BY order_date) AS previous_order,
    LEAD(order_date) OVER (ORDER BY order_date) AS next_order
FROM orders;
```

■ Output: Shows each order with its previous and next order dates.

■ 4. Why are Window Functions Important? (Advantages)

- They simplify complex queries (no need for subqueries or self-joins).
- Essential in analytics, dashboards, BI tools.
- Used for: Ranking (leaderboards, top-N queries).
- Running totals (sales trends, cumulative data).
- Moving averages (financial analysis, stock data).
- Lag/Lead comparisons (time-series analysis).
- Supported in MySQL (8.0+), SQL Server, Oracle, PostgreSQL.

■ Final Interview Answer

"Window functions in SQL let us perform calculations across a set of related rows while still returning row-level detail. Unlike aggregate functions, they don't collapse rows. They are defined using the OVER() clause and support operations like ranking, aggregates, comparisons (LAG/LEAD), running totals, and moving averages.

For example:
- RANK() OVER(ORDER BY salary DESC) → ranks employees by salary.
- AVG(salary) OVER(PARTITION BY department_id) → shows each employee's salary with department's average.
- SUM(amount) OVER(ORDER BY order_date) → calculates running totals.