❓ What Are Window Functions in SQL?

Window functions are the special functions in SQL that we use to perform operations across a set of related rows **without collapsing the rows into a single result.**

Basically, when we want to perform operations on rows that are related to each other, we use the **GROUP BY** clause, which combines the rows into a single row per group and returns one result.
But if we want to perform the operation corresponding to a group of records and want the result **separately for each row**, we use **window functions.**

Here, we use the **OVER** clause to perform the query.

🔹 Syntax:

sql Copy code

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

Here,

- **function_name** means the window function applied on the window.
- **OVER clause** is used to define how and on which operation the function is performed.
- **PARTITION BY** defines the field on which rows are grouped (like GROUP BY, but separately).
- **ORDER BY** defines the order of rows in each partition.

Using this, we can get the correct result as per the required calculation.

🔹 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:

```sql
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:

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

(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:

```sql
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.