**What Are Window Functions?**

**Window functions** perform **calculations across a set of table rows** that are **related to the current row**, **without collapsing the result** into a single output (unlike GROUP BY).

🔹 **Introduced in MySQL 8.0+**  
🔹 Syntax: Uses the OVER() clause.

Common window functions in mysql

| **Function** | **Description** |
| --- | --- |
| ROW\_NUMBER() | Assigns a unique row number per partition. |
| RANK() | Gives rank with gaps for ties. |
| DENSE\_RANK() | Rank without gaps for ties. |
| NTILE(n) | Divides rows into n buckets. |
| LAG() / LEAD() | Access previous/next row value. |
| FIRST\_VALUE() | First value in the window. |
| LAST\_VALUE() | Last value in the window. |
| SUM(), AVG(), COUNT() | Aggregates **without collapsing rows**. |

Implementation

Sales table

| **emp\_id** | **Department t** | **amount** | **sale\_date** |
| --- | --- | --- | --- |
| 1 | A | 100 | 2024-01-01 |
| 2 | A | 200 | 2024-01-02 |
| 3 | B | 300 | 2024-01-01 |
| 4 | A | 150 | 2024-01-03 |

**1. ROW\_NUMBER()**

SELECT emp\_id, department, amount,

ROW\_NUMBER() OVER (PARTITION BY department ORDER BY amount DESC) AS row\_num

FROM sales;

\*Gives a unique row number for each department, ordered by amount.

2. RANK() vs DENSE\_RANK()

SELECT emp\_id, department, amount,

RANK() OVER (PARTITION BY department ORDER BY amount DESC) AS rank,

DENSE\_RANK() OVER (PARTITION BY department ORDER BY amount DESC) AS dense\_rank

FROM sales;

RANK() skips numbers if there's a tie.

DENSE\_RANK() doesn’t skip.

3. SUM() as Window Function

SELECT emp\_id, department, amount,

SUM(amount) OVER (PARTITION BY department) AS dept\_total

FROM sales;

\*Adds a **department-wise total** to each row without grouping.

4. LAG() and LEAD()

SELECT emp\_id, amount,

LAG(amount) OVER (ORDER BY sale\_date) AS prev\_amount,

LEAD(amount) OVER (ORDER BY sale\_date) AS next\_amount

FROM sales;

\*Fetches the **previous and next row values** relative to the current row.

5. FIRST\_VALUE() and LAST\_VALUE()

SELECT emp\_id, department, amount,

FIRST\_VALUE(amount) OVER (PARTITION BY department ORDER BY sale\_date) AS first\_sale,

LAST\_VALUE(amount) OVER (PARTITION BY department ORDER BY sale\_date ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS last\_sale

FROM sales;

 FIRST\_VALUE() – gets first sale per department

 LAST\_VALUE() – be careful with window frame (otherwise returns current row)

6. NTILE()

SELECT emp\_id, amount,

NTILE(2) OVER (ORDER BY amount) AS bucket

FROM sales;

\*Divides rows into **equal-sized groups (tiles)**.

The **CASE statement** in MySQL is like an **if-else ladder** used to add **conditional logic** inside queries such as SELECT, UPDATE, ORDER BY, or even in WHERE clauses.

Employees table

| **emp\_id** | **name** | **department** | **salary** |
| --- | --- | --- | --- |
| 1 | Alice | HR | 40000 |
| 2 | Bob | IT | 60000 |
| 3 | Carol | Finance | 55000 |
| 4 | David | IT | 72000 |

CASE IN

SELECT name, salary,

CASE

WHEN salary >= 70000 THEN 'High'

WHEN salary BETWEEN 50000 AND 69999 THEN 'Medium'

ELSE 'Low'

END AS salary\_level

FROM employees;

Simple CASE

SELECT name, department,

CASE department

WHEN 'IT' THEN 'Technology'

WHEN 'HR' THEN 'People'

ELSE 'Others'

END AS dept\_category

FROM employees;

CASE IN ORDER BY

SELECT \* FROM employees

ORDER BY

CASE department

WHEN 'IT' THEN 1

WHEN 'HR' THEN 2

ELSE 3

END;

CASE IN UPDATE

UPDATE employees

SET salary =

CASE

WHEN department = 'IT' THEN salary + 5000

WHEN department = 'HR' THEN salary + 2000

ELSE salary

END;

A **CTE** is a **temporary named result set** (like a virtual table) defined within a SQL query. It improves query **readability**, **reusability**, and supports **recursion**.

**Introduced in MySQL 8.0**

Employees table

| **emp\_id** | **name** | **department** | **salary** |
| --- | --- | --- | --- |
| 1 | Alice | HR | 40000 |
| 2 | Bob | IT | 60000 |
| 3 | Carol | Finance | 55000 |
| 4 | David | IT | 72000 |
| 5 | Eva | HR | 50000 |

WITH high\_earners AS (

SELECT name, salary

FROM employees

WHERE salary > 50000

)

SELECT \* FROM high\_earners;

CTE with JOIN

WITH it\_employees AS (

SELECT emp\_id, name

FROM employees

WHERE department = 'IT'

),

salaries AS (

SELECT emp\_id, salary

FROM employees

)

SELECT it.name, s.salary

FROM it\_employees it

JOIN salaries s ON it.emp\_id = s.emp\_id;