In MySQL, **aggregate functions** and the **GROUP BY clause** are commonly used to summarize data. Here's a detailed explanation with examples:

**Aggregate Functions**

These functions perform calculations on multiple rows of data and return a single value.

| **Function** | **Description** | **Example Usage** |
| --- | --- | --- |
| COUNT() | Counts the number of rows. | COUNT(\*) |
| SUM() | Calculates the total sum of a column. | SUM(column\_name) |
| AVG() | Calculates the average of a column. | AVG(column\_name) |
| MAX() | Returns the maximum value in a column. | MAX(column\_name) |
| MIN() | Returns the minimum value in a column. | MIN(column\_name) |

**GROUP BY Clause**

The **GROUP BY clause** is used with aggregate functions to group rows that have the same values in specified columns into summary rows.

**Syntax:**

sql

Copy code

SELECT column1, aggregate\_function(column2)

FROM table\_name

GROUP BY column1;

**Examples**

**1. Basic Example with COUNT**

**Query:**

sql

Copy code

SELECT department\_id, COUNT(\*) AS employee\_count

FROM employees

GROUP BY department\_id;

**Explanation:**

* Groups employees by department\_id.
* Counts the number of employees in each department.

**Result:**

| **department\_id** | **employee\_count** |
| --- | --- |
| 1 | 5 |
| 2 | 7 |

**2. Using SUM**

**Query:**

sql

Copy code

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id;

**Explanation:**

* Groups employees by department\_id.
* Calculates the total salary for each department.

**Result:**

| **department\_id** | **total\_salary** |
| --- | --- |
| 1 | 25000 |
| 2 | 37000 |

**3. Using AVG**

**Query:**

sql

Copy code

SELECT department\_id, AVG(salary) AS average\_salary

FROM employees

GROUP BY department\_id;

**Explanation:**

* Groups employees by department\_id.
* Calculates the average salary for each department.

**Result:**

| **department\_id** | **average\_salary** |
| --- | --- |
| 1 | 5000 |
| 2 | 5285.71 |

**4. Using MAX and MIN**

**Query:**

sql

Copy code

SELECT department\_id, MAX(salary) AS highest\_salary, MIN(salary) AS lowest\_salary

FROM employees

GROUP BY department\_id;

**Explanation:**

* Groups employees by department\_id.
* Finds the highest and lowest salary in each department.

**Result:**

| **department\_id** | **highest\_salary** | **lowest\_salary** |
| --- | --- | --- |
| 1 | 7000 | 3000 |
| 2 | 8000 | 4000 |

**5. Using HAVING with GROUP BY**

The **HAVING clause** is used to filter grouped data, similar to WHERE for individual rows.

**Query:**

sql

Copy code

SELECT department\_id, COUNT(\*) AS employee\_count

FROM employees

GROUP BY department\_id

HAVING employee\_count > 5;

**Explanation:**

* Groups employees by department\_id.
* Displays only departments with more than 5 employees.

**Result:**

| **department\_id** | **employee\_count** |
| --- | --- |
| 2 | 7 |

**When to Use WHERE vs. HAVING**

* **WHERE** filters rows **before grouping**.
* **HAVING** filters groups **after grouping**.

**Example:**

sql

Copy code

SELECT department\_id, AVG(salary) AS average\_salary

FROM employees

WHERE salary > 4000

GROUP BY department\_id

HAVING average\_salary > 5000;

1. WHERE salary > 4000 filters rows first.
2. GROUP BY department\_id groups the remaining rows.
3. HAVING average\_salary > 5000 filters groups based on the calculated average.

**Aggregate Functions in MySQL**

1. **COUNT()**
   * Returns the number of rows that match a condition.

sql

Copy code

SELECT COUNT(\*)

FROM employees;

1. **SUM()**
   * Calculates the total sum of a numeric column.

sql

Copy code

SELECT SUM(salary)

FROM employees;

1. **AVG()**
   * Calculates the average of a numeric column.

sql

Copy code

SELECT AVG(salary)

FROM employees;

1. **MAX()**
   * Returns the maximum value in a column.

sql

Copy code

SELECT MAX(salary)

FROM employees;

1. **MIN()**
   * Returns the minimum value in a column.

sql

Copy code

SELECT MIN(salary)

FROM employees;

**GROUP BY Clause**

The **GROUP BY** clause is used to arrange identical data into groups.

**Syntax:**

sql

Copy code

SELECT column1, aggregate\_function(column2)

FROM table\_name

GROUP BY column1;

**Examples**

**1. Count employees in each department**

sql

Copy code

SELECT department\_id, COUNT(\*) AS employee\_count

FROM employees

GROUP BY department\_id;

**Result:**

| **department\_id** | **employee\_count** |
| --- | --- |
| 1 | 5 |
| 2 | 3 |

**2. Total salary by department**

sql

Copy code

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id;

**Result:**

| **department\_id** | **total\_salary** |
| --- | --- |
| 1 | 50000 |
| 2 | 75000 |

**3. Average salary by job title**

sql

Copy code

SELECT job\_title, AVG(salary) AS average\_salary

FROM employees

GROUP BY job\_title;

**Result:**

| **job\_title** | **average\_salary** |
| --- | --- |
| Manager | 80000 |
| Developer | 60000 |

**Using GROUP BY with HAVING**

The **HAVING** clause filters groups after aggregation (similar to WHERE, which filters rows before aggregation).

**Example:**

Find departments with total salaries above 50,000:

sql

Copy code

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id

HAVING total\_salary > 50000;

**Result:**

| **department\_id** | **total\_salary** |
| --- | --- |
| 2 | 75000 |

**Key Notes:**

* **GROUP BY vs. ORDER BY:**
  + GROUP BY groups rows, while ORDER BY sorts them.
  + You can use both together:

sql

Copy code

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id

ORDER BY total\_salary DESC;

* Columns in the **SELECT** statement must either:
  + Appear in the GROUP BY clause.
  + Be used in an aggregate function.