SQL Aggregate Functions

SQL aggregate functions perform calculations on a set of values and return a single summarized result. These functions are often used with the **GROUP BY** clause to group rows that share the same values in specified columns.

---

1. COUNT()

- **Description:** Counts the number of rows in a result set or the number of non-NULL values in a column.
- **Syntax:**

```sql
SELECT COUNT(column_name)
FROM table_name;
```

- **Example:**

```sql
SELECT COUNT(employee_id) AS total_employees
FROM employees;
```

Result: Total number of employees.

---

2. SUM()

- **Description:** Calculates the total sum of a numeric column.
- **Syntax:**

```sql
SELECT SUM(column_name)
FROM table_name;
```

- **Example:**

```sql
SELECT SUM(salary) AS total_salary
FROM employees;
```

Result: Total salary of all employees.

---

3. AVG()

- **Description:** Calculates the average value of a numeric column.
- **Syntax:**

```sql
SELECT AVG(column_name)
FROM table_name;
```

- **Example:**

```sql
SELECT AVG(salary) AS average_salary
FROM employees;
```

Result: Average salary of all employees.

---

4. MIN()

- **Description:** Finds the smallest value in a column.
- **Syntax:**

```sql
SELECT MIN(column_name)
FROM table_name;
```

- **Example:**

```sql
SELECT MIN(salary) AS lowest_salary
FROM employees;
```

Result: Lowest salary in the employees table.

---

5. MAX()

- **Description:** Finds the largest value in a column.
- **Syntax:**

```sql
SELECT MAX(column_name)
FROM table_name;
```

- **Example:**

```sql
SELECT MAX(salary) AS highest_salary
FROM employees;
```

Result: Highest salary in the employees table.

---

6. GROUP_CONCAT() (MySQL-specific)

- **Description:** Concatenates values from a column into a single string, separated by a specified delimiter.
- **Syntax:**

```sql
SELECT GROUP_CONCAT(column_name SEPARATOR 'delimiter')
FROM table_name;
```

- **Example:**

```sql
SELECT GROUP_CONCAT(name SEPARATOR ', ') AS employee_names
FROM employees;
```

Result: A string of all employee names separated by commas.

Using Aggregate Functions with GROUP BY

- **GROUP BY** groups rows that have the same values in specified columns and applies aggregate functions to each group.

Example 1: Count Employees in Each Department

```sql
SELECT department_id, COUNT(employee_id) AS total_employees
FROM employees
GROUP BY department_id;
```

Result:

	department_id	total_employees
1	1	10
2	2	8

Neu

Example 2: Average Salary by Job Title

```sql
SELECT job_title, AVG(salary) AS average_salary
FROM employees
GROUP BY job_title;
```

Using Aggregate Functions with HAVING

- The **HAVING** clause filters groups after aggregation (similar to WHERE but for grouped data).

Example: Departments with Total Salary Above 100,000

```sql
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id
HAVING SUM(salary) > 100000;
```

Combining Aggregate Functions

- You can use multiple aggregate functions in a single query.

Example: Department Statistics

```sql
SELECT department_id,
       COUNT(employee_id) AS total_employees,
```

```
        AVG(salary) AS average_salary,
        MIN(salary) AS lowest_salary,
        MAX(salary) AS highest_salary
FROM employees
GROUP BY department_id;

```

Key Points

1. Aggregate functions ignore `NULL` values, except for `COUNT(*)`.
2. Combine with `GROUP BY` for meaningful grouping and summarization.
3. Use `HAVING` for conditions on aggregated values (not `WHERE`).

Would you like practical examples with a sample dataset or further explanations on any function?