

**Name-Harshit Selarka**

**Prn-23070521131**

**Section-B**

**Batch-B2**

## **SQL Aggregate Functions in SQL\*Plus (Oracle) & MySQL**

Aggregate functions perform calculations on multiple rows and return a **single** value. These functions are useful in **summarizing** data, such as totals, averages, counts, etc.

Below are detailed examples covering all aggregate functions in **SQL\*Plus (Oracle)** and **MySQL**.

### **1. Aggregate Functions in SQL\*Plus (Oracle)**

#### **1.1 SUM() – Total of a Column**

```
SELECT SUM(salary) AS total_salary FROM employees; -- Total salary of  
all employees  
SELECT department_id, SUM(salary) FROM employees GROUP BY  
department_id; -- Total salary per department
```

Customers [-]

- customer\_id [int]
- first\_name [varchar(100)]
- last\_name [varchar(100)]
- age [int]
- country [varchar(100)]

Employees [-]

- employee\_id [integer]
- department\_id [integer]
- salary [real]

Orders [-]

- order\_id [integer]
- item [varchar(100)]
- amount [integer]
- customer\_id [integer]

Shippings [-]

- shipping\_id [integer]
- status [integer]
- customer [integer]

Input

```

-- 1. Drop the table if it already exists (Prevents duplicate table creation error)
DROP TABLE IF EXISTS employees;

-- 2. Create the 'employees' table
CREATE TABLE employees (
    employee_id INTEGER PRIMARY KEY,
    department_id INTEGER,
    salary REAL
);

-- 3. Insert sample data
INSERT INTO employees (employee_id, department_id, salary) VALUES
(1, 10, 5000),
(2, 10, 6000),
(3, 20, 7000),
(4, 20, 8000),
(5, 30, 5500),
(6, 30, 6500);

```

Run SQL

Output

total_salary	
45500	

department_id	SUM(salary)
10	11000
20	15000
30	19500

avg_salary	
4500	

## 1.2 AVG() – Average of a Column

```

SELECT AVG(salary) AS avg_salary FROM employees; -- Average salary
SELECT department_id, AVG(salary) FROM employees GROUP BY
department_id; -- Average salary per department

```

## 1.3 COUNT() – Counting Rows

```

SELECT COUNT(*) FROM employees; -- Total number of employees
SELECT COUNT(employee_id) FROM employees WHERE department_id = 10; -- Count employees in department 10

```

```
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id;
-- Count per department
```

Customers [-]

- customer\_id [int]
- first\_name [varchar(100)]
- last\_name [varchar(100)]
- age [int]
- country [varchar(100)]

Employees [-]

- employee\_id [integer]
- department\_id [integer]
- salary [real]

Orders [-]

- order\_id [integer]
- item [varchar(100)]
- amount [integer]
- customer\_id [integer]

Shippings [-]

- shipping\_id [integer]
- status [integer]
- customer [integer]

Input

```

-- 2. Create the 'employees' table
CREATE TABLE employees (
    employee_id INTEGER PRIMARY KEY,
    department_id INTEGER,
    salary REAL
);

-- 3. Insert sample data
INSERT INTO employees (employee_id, department_id, salary) VALUES
(1, 10, 5000),
(2, 10, 6000),
(3, 20, 7000),
(4, 20, 8000),
(5, 30, 5500),
(6, 30, 6500),
(7, 30, 7500);

-- 4. SUM() - Total salary

```

Output

45000

department_id	SUM(salary)
10	11000
20	15000
30	19500

avg\_salary

6500

department_id	AVG(salary)
--	----

## 1.4 MAX() – Maximum Value

```
SELECT MAX(salary) FROM employees; -- Highest salary in the company
SELECT department_id, MAX(salary) FROM employees GROUP BY
department_id; -- Highest salary per department
```

## 1.5 MIN() – Minimum Value

```
SELECT MIN(salary) FROM employees; -- Lowest salary in the company
```

```
SELECT department_id, MIN(salary) FROM employees GROUP BY
department_id; -- Lowest salary per department
```

The screenshot shows a SQL IDE interface. On the left is a schema tree with tables: Customers, Employees, Orders, and Shippings. The main area is titled 'Input' and contains SQL code for various aggregate functions. Below the code is an 'Output' section showing the results of the queries.

**Customers [-]**

- customer\_id [int]
- first\_name [varchar(100)]
- last\_name [varchar(100)]
- age [int]
- country [varchar(100)]

**Employees [-]**

- employee\_id [integer]
- department\_id [integer]
- salary [real]

**Orders [-]**

- order\_id [integer]
- item [varchar(100)]
- amount [integer]
- customer\_id [integer]

**Shippings [-]**

- shipping\_id [integer]
- status [integer]
- customer [integer]

**Input**

```
(7, 30, 7500);

-- 4. SUM() - Total salary
SELECT SUM(salary) AS total_salary FROM employees;
SELECT department_id, SUM(salary) FROM employees GROUP BY department_id;

-- 5. AVG() - Average salary
SELECT AVG(salary) AS avg_salary FROM employees;
SELECT department_id, AVG(salary) FROM employees GROUP BY department_id;

-- 6. COUNT() - Counting employees
SELECT COUNT(*) FROM employees;
SELECT COUNT(employee_id) FROM employees WHERE department_id = 10;
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id;

-- 7. MAX() - Highest salary
SELECT MAX(salary) FROM employees;
SELECT department_id, MAX(salary) FROM employees GROUP BY department_id;
```

**Output**

department_id	AVG(salary)
10	5500
20	7500
30	6500

COUNT(*)
7

COUNT(employee_id)
2

## 1.6 MEDIAN() – Median Value (Oracle-Only)

```
SELECT MEDIAN(salary) FROM employees; -- Median salary of all
employees
```

## 1.7 STDDEV() – Standard Deviation

```
SELECT STDDEV(salary) FROM employees; -- Standard deviation of salaries
```

```
SELECT department_id, STDDEV(salary) FROM employees GROUP BY department_id; -- Std deviation per department
```

## 1.8 VARIANCE() – Variance of a Column

```
SELECT VARIANCE(salary) FROM employees; -- Variance of salaries
```

The screenshot shows a SQL IDE interface. On the left is a database schema with four tables: Customers, Employees, Orders, and Shippings. The main area is titled 'Input' and contains a SQL query. The query includes comments for each step and the corresponding SQL statements. The output section shows the results of the query, including the count of employees and the standard deviation of salaries grouped by department.

**Customers [-]**

- customer\_id [int]
- first\_name [varchar(100)]
- last\_name [varchar(100)]
- age [int]
- country [varchar(100)]

**Employees [-]**

- employee\_id [integer]
- department\_id [integer]
- salary [real]

**Orders [-]**

- order\_id [integer]
- item [varchar(100)]
- amount [integer]
- customer\_id [integer]

**Shippings [-]**

- shipping\_id [integer]
- status [integer]
- customer [integer]

**Input**

```
-- 6. COUNT() - Counting employees
SELECT COUNT(*) FROM employees;
SELECT COUNT(employee_id) FROM employees WHERE department_id = 10;
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id;

-- 7. MAX() - Highest salary
SELECT MAX(salary) FROM employees;
SELECT department_id, MAX(salary) FROM employees GROUP BY department_id;

-- 8. MIN() - Lowest salary
SELECT MIN(salary) FROM employees;
SELECT department_id, MIN(salary) FROM employees GROUP BY department_id;

-- 9. STDDEV() - Standard Deviation (SQLite does not support STDDEV, using alternative)
SELECT department_id,
       (AVG(salary * salary) - AVG(salary) * AVG(salary)) AS stddev_salary
FROM employees GROUP BY department_id;
```

**Output**

COUNT(employee_id)	
2	

department_id	COUNT(*)
10	2
20	2
30	3

MAX(salary)	
8000	

## 1.9 GROUP BY with Aggregate Functions

```
SELECT department_id, COUNT(*), AVG(salary), MAX(salary), MIN(salary)
FROM employees
GROUP BY department_id; -- Aggregate calculations per department
```

## 1.10 HAVING Clause (Filtering Groups)

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5; -- Only departments with more than 5 employees
```

Customers [-]

- customer\_id [int]
- first\_name [varchar(100)]
- last\_name [varchar(100)]
- age [int]
- country [varchar(100)]

Employees [-]

- employee\_id [integer]
- department\_id [integer]
- salary [real]

Orders [-]

- order\_id [integer]
- item [varchar(100)]
- amount [integer]
- customer\_id [integer]

Shippings [-]

- shipping\_id [integer]
- status [integer]
- customer [integer]

Input

```

SELECT department_id, MIN(salary) FROM employees GROUP BY department_id;

-- 9. STDDEV() - Standard Deviation (SQLite does not support STDDEV, using alternative)
SELECT department_id,
    (AVG(salary * salary) - AVG(salary) * AVG(salary)) AS stddev_salary
FROM employees GROUP BY department_id;

-- 10. VARIANCE() - Variance (Not available in SQLite, using alternative)
SELECT department_id,
    (AVG(salary * salary) - AVG(salary) * AVG(salary)) AS variance_salary
FROM employees GROUP BY department_id;

-- 11. GROUP BY with Aggregate Functions
SELECT department_id, COUNT(*) AS count, AVG(salary), MAX(salary), MIN(salary)
FROM employees
GROUP BY department_id;

-- 12. HAVING Clause (Filtering departments with more than 2 employees)
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 2;

```

Run SQL

Output

department_id	variance_salary
10	250000
20	250000
30	666666.66666666642

department_id	count	AVG(salary)	MAX(salary)	MIN(salary)
10	2	5500	6000	5000
20	2	7500	8000	7000
30	3	6500	7500	5500

department_id	employee_count
30	3

## 2. Aggregate Functions in MySQL

### 2.1 SUM() – Total of a Column

SELECT SUM(salary) AS total\_salary FROM employees; -- Total salary of all employees

SELECT department\_id, SUM(salary) FROM employees GROUP BY department\_id; -- Total salary per department

### 2.2 AVG() – Average of a Column

```
SELECT AVG(salary) AS avg_salary FROM employees; -- Average salary
SELECT department_id, AVG(salary) FROM employees GROUP BY
department_id; -- Average salary per department
```

### **2.3 COUNT() – Counting Rows**

```
SELECT COUNT(*) FROM employees; -- Total number of employees
SELECT COUNT(employee_id) FROM employees WHERE department_id = 10; -- Count
employees in department 10
SELECT department_id, COUNT(*) FROM employees GROUP BY department_id;
-- Count per department
```

### **2.4 MAX() – Maximum Value**

```
SELECT MAX(salary) FROM employees; -- Highest salary in the company
SELECT department_id, MAX(salary) FROM employees GROUP BY
department_id; -- Highest salary per department
```

### **2.5 MIN() – Minimum Value**

```
SELECT MIN(salary) FROM employees; -- Lowest salary in the company
SELECT department_id, MIN(salary) FROM employees GROUP BY
department_id; -- Lowest salary per department
```

### **2.6 STDDEV() – Standard Deviation**

```
SELECT STDDEV(salary) FROM employees; -- Standard deviation of
salaries
SELECT department_id, STDDEV(salary) FROM employees GROUP BY
department_id; -- Std deviation per department
```



## 2.7 VARIANCE() – Variance of a Column

```
SELECT VARIANCE(salary) FROM employees; -- Variance of salaries
```

## 2.8 GROUP BY with Aggregate Functions

```
SELECT department_id, COUNT(*), AVG(salary), MAX(salary), MIN(salary)
FROM employees
GROUP BY department_id; -- Aggregate calculations per department
```

## 2.9 HAVING Clause (Filtering Groups)

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5; -- Only departments with more than 5 employees
```

## 3. Key Differences Between SQL\*Plus (Oracle) and MySQL Aggregate Functions

Feature	Oracle (SQL*Plus)	MySQL
Basic Aggregate Functions	<code>SUM()</code> , <code>AVG()</code> , <code>COUNT()</code> , <code>MAX()</code> , <code>MIN()</code>	<code>SUM()</code> , <code>AVG()</code> , <code>COUNT()</code> , <code>MAX()</code> , <code>MIN()</code>
Median Calculation	<code>MEDIAN()</code>	Not available (requires workaround)
Standard Deviation	<code>STDDEV()</code>	<code>STDDEV()</code>
Variance Calculation	<code>VARIANCE()</code>	<code>VARIANCE()</code>

Handling NULL values	Ignores NULL values in aggregate functions	Ignores NULL values in aggregate functions
HAVING Clause	Used after <b>GROUP BY</b>	Used after <b>GROUP BY</b>
GROUP BY	Used to group and aggregate	Used to group and aggregate

## 4. Special Notes

- **Oracle** provides the **MEDIAN()** function, but **MySQL does not**. In MySQL, median must be calculated using a workaround.
- **Both** ignore NULL values when computing aggregates unless explicitly handled. • **HAVING** is used **after GROUP BY** to filter aggregated results in both Oracle and MySQL.