Chapter: Crunching Numbers with SQL

Topic: Aggregation Functions in SQL

Section 1: Learn

1.1 Introduction to Aggregation Functions

In SQL, **aggregation functions** help you summarize large amounts of data. These functions are used when you want to **group** or **analyze** rows by computing a **single value** from multiple rows.

Aggregation is commonly used in:

- Reporting totals and averages
- Identifying maximum or minimum values
- Counting records
- Summarizing data by categories (e.g., sales by region, average marks by class)

1.2 SUM(): Total of Values

Purpose:

Calculates the **total sum** of values in a numeric column.

Example:

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

> Adds up all salary values.

Use Cases:

- Total sales, revenue, expenses
- Total quantity of products sold

1.3 AVG(): Average of Values

Purpose:

Returns the **average (mean)** of a numeric column.

Example:

```
SELECT AVG(marks) AS average_marks
FROM students;
```

> Computes the average score from all rows.

Use Cases:

- Class average
- Average customer spend

1.4 COUNT(): Counting Records

Purpose:

Returns the **number of rows**.

Examples:

```
SELECT COUNT(*) FROM orders;              -- Counts all rows
SELECT COUNT(email) FROM customers;       -- Counts only non-NULL emails
```

Use Cases:

- Number of employees
- Number of valid (non-null) entries in a column

1.5 MAX(): Maximum Value

Purpose:

Returns the **highest** value in a column.

Example:

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

Use Cases:

- Highest sale of the month
- Oldest person or latest date

1.6 MIN(): Minimum Value

Purpose:

Returns the **lowest** value in a column.

Example:

```
SELECT MIN(birth_date) AS oldest_birth
FROM students;
```

Use Cases:

- Earliest submission
- Minimum order quantity

1.7 Combining Aggregates with GROUP BY

Aggregate functions are often used with GROUP BY to summarize data across **categories**.

Example:

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department;
```

> Shows average salary per department.

Section 2: Practise

Exercise 1: Total Revenue

```
SELECT SUM(amount) AS total_revenue
FROM transactions;
```

Exercise 2: Count Number of Female Students

```
SELECT COUNT(*) AS total_females
FROM students
```

```
WHERE gender = 'Female';
```

Exercise 3: Find Max and Min Product Prices

```
SELECT MAX(price) AS highest_price,
       MIN(price) AS lowest_price
FROM products;
```

Exercise 4: Average Marks Per Class

```
SELECT class, AVG(marks) AS average_marks
FROM results
GROUP BY class;
```

Exercise 5: Number of Orders Per Customer

```
SELECT customer_id, COUNT(*) AS order_count
FROM orders
GROUP BY customer_id;
```

Exercise 6: Total Stock Per Category

```
SELECT category, SUM(stock_quantity) AS total_stock
FROM inventory
GROUP BY category;
```

Section 3: FAQ – Know More

Q1. What's the difference between COUNT*() and COUNT(column_name)?*

- COUNT(*) **counts all rows, including those with** NULL **values.**
- COUNT(column_name) **counts only rows where the column is not NULL.**

Q2. Can I use multiple aggregate functions in one query?

Yes.

```
SELECT COUNT(*), AVG(salary), MAX(salary)
FROM employees;
```

Q3. Can I filter aggregated results?

Yes. Use HAVING **(not** WHERE**) with** GROUP BY**.**

```
SELECT department, AVG(salary) AS avg_sal
FROM employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

Q4. Can I use aggregation functions without GROUP BY?

Yes. If no grouping is specified, the function runs over the entire table.

```
SELECT SUM(amount) FROM transactions;
```

Q5. What happens if a column has NULL values?

Most aggregate functions ignore NULLs, except COUNT(*)**.**

End of Notes for Chapter: Crunching Numbers with SQL