**Day 3-4: Mastering Aggregations**

**COUNT()**

The COUNT() function is used to **tally the number of rows** in your table or in the result set of your query.

It's your basic tool for answering "How many?" questions. The most common way to use it is with an asterisk (\*) to count all rows, even if some columns in those rows are empty. If you use a column name, it only counts rows where that specific column has a value (it skips any rows that are missing data in that column).

**Syntax**

**SELECT COUNT(\*)**

**FROM** table\_name;

**Basic Example Query**

To find the total number of employees listed in the 'Employees' table:

**SELECT COUNT(\*)**

**FROM** Employees;

**SUM()**

The SUM() function is used to **add up all the numbers** found in a specific column and return the grand total.

This function only works on columns that contain numeric data. It is essential for getting totals, like adding up all the sales amounts to find your total revenue. Importantly, it completely **ignores** any missing (NULL) values in the column when calculating the total.

**Syntax**

SELECT SUM(column\_name)

FROM table\_name;

**Basic Example Query**

To calculate the total revenue from the 'OrderDetails' table using the 'TotalAmount' column:

SELECT SUM(TotalAmount)

FROM OrderDetails;

**AVG()**

The AVG() function calculates the **arithmetic mean (average)** of all the numbers in a specific column.

It takes the SUM() of the values and divides it by the COUNT() of those values, giving you a single number that represents the typical value. Like SUM(), it is only used on number columns and it automatically **ignores** any missing (NULL) values in its calculation.

**Syntax**

SELECT AVG(column\_name)

FROM table\_name;

**Basic Example Query**

To find the average score from the 'Students' table using the 'TestScore' column:

SELECT AVG(TestScore)

FROM Students;

**MIN, MAX**

The MIN() and MAX() functions are two sides of the same coin: they quickly find the **smallest** and **largest** values in a column, respectively.

* **MIN()** returns the single lowest number, the earliest date, or the first word alphabetically in a text column.
* **MAX()** returns the single highest number, the latest date, or the last word alphabetically in a text column.

Both functions skip over any missing (NULL) values in their calculation.

**Syntax (Applies to both)**

SELECT MIN(column\_name)

FROM table\_name;

SELECT MAX(column\_name)

FROM table\_name;

**Basic Example Query**

To find both the cheapest and most expensive product from your 'Products' table:

SELECT MIN(Price), MAX(Price)

FROM Products;

**GROUP BY clause**

The GROUP BY clause is like asking SQL to **sort your data into categories** so you can run a calculation on each one separately.

Instead of getting a single total for the whole table, you can use GROUP BY with an aggregate function (like SUM() or COUNT()) to get a separate total for every unique value in a specific column.

For example, you can group by the 'Department' column to get the average salary for *each* department, not just the company-wide average.

**Syntax**

SELECT column\_to\_group\_by, aggregate\_function(column\_to\_calculate)

FROM table\_name

GROUP BY column\_to\_group\_by;

**Basic Example Query**

To count how many employees work in each 'Department' listed in your 'Employees' table:

SELECT Department, COUNT(\*)

FROM Employees

GROUP BY Department;

**Filtering Groups (HAVING vs. WHERE)**

Both WHERE and HAVING are used to filter (or limit) the data you see, but they work at different times in your query:

1. **WHERE** is for filtering **individual rows** *before* grouping. You use it to look at the original data and throw out rows you don't want to include in a summary. You **cannot** use aggregate functions (like SUM() or COUNT()) in the WHERE clause.
2. **HAVING** is for filtering **groups** *after* aggregation. It is always used after GROUP BY and lets you set a condition on the calculated summary value.

**Basic Example Query (Using both)**

Let's say you want to find departments where the average salary is high, but you only want to consider full-time employees:

**SELECT** Department, **AVG**(Salary)

**FROM** Employees

**WHERE** EmploymentType = 'Full-Time'

-- 1. **WHERE** filters individual employees first

**GROUP** BY Department

**HAVING** **AVG**(Salary) > 60000;

-- 2. **HAVING** filters the results of the average calculation