**Task 4: Aggregate Functions and Grouping**

**Objective:**

Use aggregate functions and grouping to summarize data Tools :DB Browser for SQLite / MySQL Workbench

**Deliverables:**

SQL queries using SUM, COUNT, AVG, GROUP BY

**Hints/Mini Guide:**

1.Apply aggregate functions on numeric columns

2.Use GROUP BY to categorize

3.Filter groups using HAVING

**Aggregate Functions Overview**

Aggregate functions operate on **sets of rows** and return a **single value**.

| **Function** | **Purpose** |
| --- | --- |
| SUM() | Total of numeric values |
| COUNT() | Number of rows |
| AVG() | Average value |
| MIN() | Smallest value |
| MAX() | Largest value |

**Aggregate functions operate on sets of rows and return a single value.**

**Aggregate functions work only with Group By Clause.**

**These Aggregate functions work on only numeric values only, doesn’t work on character values.**

**1. COUNT() — Count of Rows**

**Syntax**

SELECT COUNT(\*) FROM table\_name;

SELECT COUNT(column\_name) FROM table\_name WHERE condition;

**Rules & Notes**

* COUNT(\*) counts all rows including NULLs.
* COUNT(column) **ignores NULLs** in that column.
* Can be used with GROUP BY.

**2. SUM() — Total of a Numeric Column**

**Syntax**

SELECT SUM(column\_name) FROM table\_name;

**Rules & Notes**

* Works only on numeric data.
* NULL values are ignored.
* Use with GROUP BY for categorized totals.

**3. AVG() — Average of a Numeric Column**

**Syntax**

SELECT AVG(column\_name) FROM table\_name;

**Rules & Notes**

* Ignores NULL values.
* Always returns a decimal.

**4. MAX() — Highest Value in Column**

**Syntax**

SELECT MAX(column\_name) FROM table\_name;

**Rules & Notes**

* Can be applied on numbers, dates, and strings (lexicographically).
* NULLs are ignored.

**5. MIN() — Lowest Value in Column**

**Syntax**

SELECT MIN(column\_name) FROM table\_name;

**6. GROUP BY — Grouping Records**

**Syntax**

SELECT column, AGG\_FUNC(column)

FROM table\_name

GROUP BY column;

**Rules & Notes**

* Every column in SELECT must be either:
  + In GROUP BY, or
  + An aggregate function.
* Cannot use WHERE on aggregate results — use HAVING.

**7. HAVING — Filtering Groups (Post-Aggregation)**

**Syntax**

SELECT column, AGG\_FUNC(column)

FROM table

GROUP BY column

HAVING AGG\_FUNC(column) condition;

**Rules & Notes**

* HAVING filters **aggregated results**.
* WHERE filters **individual rows before grouping**.

**1. What is GROUP BY?**

**Definition:**  
GROUP BY is used to **combine rows that have the same values** in specified columns into summary rows, often with aggregate functions.

**Rules (Oracle 11g):**

* Every column in SELECT must either be in GROUP BY or be an aggregate function.
* GROUP BY happens **after** WHERE filtering but **before** HAVING.
* Sorting is not guaranteed — use ORDER BY.

**Example:**

-- Count books by category

SELECT CategoryID, COUNT(\*) AS TotalBooks

FROM Books

GROUP BY CategoryID;

**2. Difference between WHERE and HAVING**

| **Feature** | **WHERE** | **HAVING** |
| --- | --- | --- |
| Purpose | Filters rows before grouping | Filters groups after aggregation |
| Works on | Columns | Aggregate results |
| Aggregate Functions | Not allowed directly | Allowed |
| Execution Order | Before GROUP BY | After GROUP BY |

**Example:**

-- WHERE: Filter books before grouping

SELECT CategoryID, COUNT(\*)

FROM Books

WHERE Price > 500

GROUP BY CategoryID;

-- HAVING: Filter after aggregation

SELECT CategoryID, COUNT(\*)

FROM Books

GROUP BY CategoryID

HAVING COUNT(\*) > 5;

**3. How does COUNT(\*) differ from COUNT(column)?**

* COUNT(\*) → Counts **all rows**, including those with NULLs.
* COUNT(column) → Counts only rows where **column is NOT NULL**.

**Example:**

-- Counts all rows

SELECT COUNT(\*) FROM Members;

-- Counts only rows with email

SELECT COUNT(Email) FROM Members;

**4. Can you group by multiple columns?**

**Yes.** This creates **sub-groups** for each combination of column values.

**Example:**

-- Count books per category and year

SELECT CategoryID, YearPublished, COUNT(\*) AS TotalBooks

FROM Books

GROUP BY CategoryID, YearPublished;

**5. What is ROUND() used for?**

**Definition:**  
ROUND() rounds a number to a specified number of decimal places.

**Syntax:**

ROUND(number, decimal\_places)

**Example:**

-- Average book price rounded to 2 decimal places

SELECT ROUND(AVG(Price), 2) AS AvgPrice

FROM Books;

**6. How do you find the highest salary by department?**

**Example:**

SELECT DepartmentID, MAX(Salary) AS HighestSalary

FROM Employees

GROUP BY DepartmentID;

**7. What is the default behavior of GROUP BY?**

* Groups rows **by unique values** of the specified column(s).
* Order of results is **not guaranteed** — add ORDER BY if needed.
* Null values are treated as a single group.

**8. Explain AVG and SUM.**

**AVG(column)** → Returns the average value of a numeric column (ignores NULLs).  
**SUM(column)** → Returns the total sum of a numeric column (ignores NULLs).

**Example:**

SELECT AVG(Price) AS AvgPrice, SUM(Price) AS TotalPrice

FROM Books;

**9. How to count distinct values?**

**Example:**

-- Count distinct categories

SELECT COUNT(DISTINCT CategoryID) AS UniqueCategories

FROM Books;

**10. What is an aggregate function?**

**Definition:**  
An aggregate function performs a **calculation on a set of values** and returns a single result.

**Common Aggregate Functions:**

* COUNT() → Number of rows
* SUM() → Total sum
* AVG() → Average
* MAX() → Highest value
* MIN() → Lowest value

**Example:**

SELECT MAX(Price), MIN(Price), AVG(Price)

FROM Books;