Aggregate Functions in SQL

Objective: - Aggregate functions in SQL perform calculations on multiple rows, such as count, sum, average, minimum, and maximum, to generate summarized results for data analysis.

Aggregate Functions

- **Definition:** Aggregate functions perform calculations on a set of rows and return a single value. They are commonly used with GROUP BY to summarize data.
- **Key Characteristics:**
  - Work on groups of rows.
  - Often used in SELECT statements with group-based operations.

Syntax:

SELECT aggregate_function(column_name) AS any_reference_name FROM table_name [WHERE condition] [GROUP BY column_name] [HAVING Conditions];

Common Aggregate Functions and Examples:

Function	Description	Example
COUNT()	Counts the number of rows or non-NULL values.	SELECT COUNT(emp_id) FROM Employees;
SUM()	Calculates the total of numeric values.	SELECT SUM(salary) FROM Employees;
AVG()	Calculates the average of numeric values.	SELECT AVG(salary) FROM Employees;
MAX()	Returns the maximum value in a column.	SELECT MAX(salary) FROM Employees;
MIN()	Returns the minimum value in a column.	SELECT MIN(salary) FROM Employees;

Some Example:

I. Count Employees:

```
SELECT COUNT(emp_id) AS total_employees
FROM Employees;
```

II. Find Maximum Salary:

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

III. Group by Department and Sum Salaries:

```
SELECT dept_id, SUM(salary) AS total_salary
FROM Employees GROUP BY dept_id;
```

IV. Find all departments that are "high-cost," meaning the total sum of their salaries exceeds $200,000.

```
SELECT dept_id, SUM(salary) AS total_department_cost
FROM Employees
GROUP BY dept_id HAVING SUM(salary) > 200000;
```

V. We want to see the average salary for these employees per department, but only for departments where that average is greater than $60,000.

```
SELECT dept_id,
   AVG(salary) AS avg_ft_salary
   FROM Employees
   WHERE status = 'Full-time'      -- Step 1: Filter out Part-time/Contractors first
   GROUP BY dept_id              -- Step 2: Group the remaining Full-time staff
   HAVING AVG(salary) > 60000;     -- Step 3: Filter the groups based on the average
```

Term Work

Create the following table "**Customers**"

Column Name (Constraint)	Datatype	Size
CustomerID (PRIMARY KEY)	NUMBER	10
CustomerName (NOT NULL)	VARCHAR2	20
City (DEFAULT 'Delhi')	VARCHAR2	10
Email (UNIQUE)	VARCHAR2	20
Age (CHECK > 18)	NUMBER	5

Insert the following data in the **Customers** table

CustomerID	CustomerName	City	Email	Age
101	Rahul Kumar	Delhi	rahul@gmail.com	22
102	Neha Sharma	Mumbai	neha@gmail.com amit@gmail.com	25
103	Amit Verma	Pune	amit@gmail.com	30
104	Priya Singh	Delhi	priya@gmail.com	28
105	Rohan Mehta	Jaipur	rohan@gmail.com	35

Create the following table "**Orders**" with constraint **FOREIGN KEY** using department table.

Column Name (Constraint)	Datatype	Size
OrderID (PRIMARY KEY)	NUMBER	10
CustomerID	NUMBER	10
OrderAmount CHECK(OrderAmount > 0)	NUMBER	10,2
OrderDate (CURRENT_DATE)	DATE	
OrderStatus (DEFAULT 'Pending')	VARCHAR2	10

Insert the following data in the **Orders** table

OrderID	CustomerID	OrderAmount	OrderDate	OrderStatus
501	101	2500.00	2026-02-01	Pending
502	102	4500.00	2026-02-02	Delivered
503	103	3200.00	2026-02-03	Pending
504	104	5000.00	2026-02-04	Shipped
505	105	1800.00	2026-02-05	Cancelled

Based on above two tables answer the following Questionaries: -

1. Display the maximum and minimum order amount placed.
2. Count how many orders are currently in 'Pending' status.
3. Count the number of orders placed on each OrderDate.
4. Display cities where average customer age is greater than 25.
5. Display OrderStatus where the number of orders is more than 1, sorted by total amount in descending order.
6. Display OrderStatus with total amount and total orders where amount is greater than 2000 and status is not Cancelled.
7. Display OrderStatus where total order amount exceeds ₹4000 and number of orders is at least 1.
8. Display cities having more than one customer and average age greater than 25.
9. Display OrderDate where total sales exceed ₹3000 and minimum order amount is greater than ₹2000.
10. Display OrderStatus where maximum order amount is greater than ₹3000 and minimum order amount is greater than ₹1500.
11. Display cities where total number of customers is more than 1 and minimum age is greater than 20.
12. Display OrderStatus where total amount is between ₹3000 and ₹6000 and total orders are less than 3.
13. Display OrderStatus where average order amount is greater than ₹2500 and total amount is above ₹3000.
14. Display cities where average age is between 25 and 35 and total customers are at least 1.
15. Display OrderStatus where total sales are greater than ₹2000, excluding Cancelled orders, sorted by total sales.