Database Systems

Lab Manual (Lab 6)

****

Session: Fall 2024

LAB INSTRUCTOR: AYESHA MAJID ALI

Lab Manual: Aggregate Functions, Joins, GROUP BY, and HAVING

### Objective:

The objective of this lab is to understand and apply the usage of **aggregate functions**, **joins**, **GROUP BY**, and **HAVING** clauses in SQL to analyze and manipulate data in relational databases. These features are essential for performing data aggregation, filtering grouped data, and combining data from multiple tables.

### Prerequisites:

* Basic knowledge of SQL.
* Access to Oracle 10g Database.

### Lab Environment Setup:

1. **Oracle Installation**: Ensure that Oracle 10g is installed and running.

#### **Topics Covered:**

* **Aggregate Functions** (COUNT, SUM, AVG, MIN, MAX)
* **Joins** (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, CROSS JOIN)
* **GROUP BY** Clause
* **HAVING** Clause

### **1. Introduction to Aggregate Functions**

Aggregate functions are used in SQL to perform a calculation on a set of values and return a single value. These functions are often used in conjunction with the **GROUP BY** clause to perform calculations on grouped data.

#### **1.1 Common Aggregate Functions:**

1. **COUNT()** – Returns the number of rows that match a specified condition.
2. **SUM()** – Returns the sum of a numeric column.
3. **AVG()** – Returns the average value of a numeric column.
4. **MIN()** – Returns the minimum value in a set.
5. **MAX()** – Returns the maximum value in a set.

#### **Syntax:**

SELECT aggregate\_function(column\_name)

FROM table\_name

WHERE condition;

#### **Example 1: Using Aggregate Functions**

Suppose we have a table Employees with the following structure

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

Salary DECIMAL(10, 2),

DepartmentID INT

);

Sample Data:

INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, DepartmentID)

VALUES (1, 'Alice', 'Johnson', 50000, 1);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, DepartmentID)

VALUES (2, 'Bob', 'Smith', 60000, 2);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, DepartmentID)

VALUES (3, 'Charlie', 'Brown', 55000, 1);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, DepartmentID)

VALUES (4, 'David', 'White', 70000, 2);

TASK

Query 1: Count the number of employees:

SELECT COUNT(\*) AS TotalEmployees

FROM Employees;

Query 2: Get the total salary paid to employees:

SELECT SUM(Salary) AS TotalSalary

FROM Employees;

Query 3: Get the average salary of employees:

SELECT AVG(Salary) AS AverageSalary

FROM Employees;

Query 4: Find the highest and lowest salaries:

SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary

FROM Employees;

### **2. Introduction to Joins**

Joins are used to combine data from two or more tables based on a related column. The most common types of joins are **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, **FULL OUTER JOIN**, and **CROSS JOIN**.

#### **2.4 FULL OUTER JOIN**

A **FULL OUTER JOIN** returns all records when there is a match in either the left or the right table. If there is no match, the result will contain NULL for missing values from either side.

**Example : Using FULL OUTER JOIN**

SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName

FROM Employees

FULL OUTER JOIN Departments

ON Employees.DepartmentID = Departments.DepartmentID;

#### **CROSS JOIN**

A **CROSS JOIN** returns the Cartesian product of two tables. Each row from the first table is combined with every row from the second table.

**Example : Using CROSS JOIN**

SELECT Employees.FirstName, Departments.DepartmentName

FROM Employees

CROSS JOIN Departments;

### **3. GROUP BY Clause**

The GROUP BY clause is used to group rows that have the same values into summary rows. Often, the GROUP BY clause is used with aggregate functions to group data by one or more columns.

#### **Syntax:**

SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

GROUP BY column\_name;

**Example : Using GROUP BY**

SELECT DepartmentID, AVG(Salary) AS AverageSalary

FROM Employees

GROUP BY DepartmentID;

### **4. HAVING Clause**

The HAVING clause is used to filter records that are grouped by the GROUP BY clause. It is similar to the WHERE clause, but WHERE is used for filtering individual rows, while HAVING filters groups.

#### **Syntax:**

SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

GROUP BY column\_name

HAVING condition;

Example : Using HAVING

SELECT DepartmentID, AVG(Salary) AS AverageSalary

FROM Employees

GROUP BY DepartmentID

HAVING AVG(Salary) > 60000;

This query will return only those departments where the average salary is greater than 60,000.

**TASKS:**

### **Software House Schema**:

* **Employee** (Employee\_ID, Employee\_Name, Role, Department\_ID, Salary)
* **Department** (Department\_ID, Department\_Name)
* **Project** (Project\_ID, Project\_Name, Start\_Date, End\_Date, Client\_ID)
* **Client** (Client\_ID, Client\_Name, Client\_Contact)
* **Task** (Task\_ID, Task\_Name, Project\_ID, Assigned\_Employee\_ID, Start\_Date, End\_Date, Status)
* **Time\_Log** (Time\_Log\_ID, Task\_ID, Employee\_ID, Hours\_Worked, Log\_Date)
* **Deliverable** (Deliverable\_ID, Project\_ID, Description, Delivery\_Date)

CREATE ABOVE TABLES:

-- Create Department Table

CREATE TABLE Department (

Department\_ID NUMBER PRIMARY KEY,

Department\_Name VARCHAR2(255) NOT NULL

);

-- Create Employee Table

CREATE TABLE Employee (

Employee\_ID NUMBER PRIMARY KEY,

Employee\_Name VARCHAR2(255) NOT NULL,

Role VARCHAR2(255) NOT NULL,

Department\_ID NUMBER,

Salary NUMBER(10, 2),

CONSTRAINT fk\_department FOREIGN KEY (Department\_ID) REFERENCES Department(Department\_ID)

);

-- Create Client Table

CREATE TABLE Client (

Client\_ID NUMBER PRIMARY KEY,

Client\_Name VARCHAR2(255) NOT NULL,

Client\_Contact VARCHAR2(255)

);

-- Create Project Table

CREATE TABLE Project (

Project\_ID NUMBER PRIMARY KEY,

Project\_Name VARCHAR2(255) NOT NULL,

Start\_Date DATE,

End\_Date DATE,

Client\_ID NUMBER,

CONSTRAINT fk\_client FOREIGN KEY (Client\_ID) REFERENCES Client(Client\_ID)

);

-- Create Task Table

CREATE TABLE Task (

Task\_ID NUMBER PRIMARY KEY,

Task\_Name VARCHAR2(255) NOT NULL,

Project\_ID NUMBER,

Assigned\_Employee\_ID NUMBER,

Start\_Date DATE,

End\_Date DATE,

Status VARCHAR2(50),

CONSTRAINT fk\_project FOREIGN KEY (Project\_ID) REFERENCES Project(Project\_ID),

CONSTRAINT fk\_employee FOREIGN KEY (Assigned\_Employee\_ID) REFERENCES Employee(Employee\_ID)

);

-- Create Time\_Log Table

CREATE TABLE Time\_Log (

Time\_Log\_ID NUMBER PRIMARY KEY,

Task\_ID NUMBER,

Employee\_ID NUMBER,

Hours\_Worked NUMBER(5, 2),

Log\_Date DATE,

CONSTRAINT fk\_task FOREIGN KEY (Task\_ID) REFERENCES Task(Task\_ID),

CONSTRAINT fk\_employee\_log FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID)

);

-- Create Deliverable Table

CREATE TABLE Deliverable (

Deliverable\_ID NUMBER PRIMARY KEY,

Project\_ID NUMBER,

Description VARCHAR2(255),

Delivery\_Date DATE,

CONSTRAINT fk\_project\_deliverable FOREIGN KEY (Project\_ID) REFERENCES Project(Project\_ID)

);

INSERTING RECORD:

-- Insert sample data into Department table

INSERT INTO Department (Department\_ID, Department\_Name) VALUES (1, 'Development');

INSERT INTO Department (Department\_ID, Department\_Name) VALUES (2, 'Testing');

INSERT INTO Department (Department\_ID, Department\_Name) VALUES (3, 'Project Management');

INSERT INTO Department (Department\_ID, Department\_Name) VALUES (4, 'HR');

-- Insert sample data into Employee table

INSERT INTO Employee (Employee\_ID, Employee\_Name, Role, Department\_ID, Salary) VALUES (1, 'John Doe', 'Developer', 1, 70000);

INSERT INTO Employee (Employee\_ID, Employee\_Name, Role, Department\_ID, Salary) VALUES (2, 'Jane Smith', 'Tester', 2, 60000);

INSERT INTO Employee (Employee\_ID, Employee\_Name, Role, Department\_ID, Salary) VALUES (3, 'Alice Johnson', 'Project Manager', 3, 80000);

INSERT INTO Employee (Employee\_ID, Employee\_Name, Role, Department\_ID, Salary) VALUES (4, 'Bob Brown', 'HR Manager', 4, 75000);

INSERT INTO Employee (Employee\_ID, Employee\_Name, Role, Department\_ID, Salary) VALUES (5, 'Charlie White', 'Developer', 1, 72000);

-- Insert sample data into Client table

INSERT INTO Client (Client\_ID, Client\_Name, Client\_Contact) VALUES (1, 'ABC Corp', 'abc@corp.com');

INSERT INTO Client (Client\_ID, Client\_Name, Client\_Contact) VALUES (2, 'XYZ Inc', 'xyz@inc.com');

INSERT INTO Client (Client\_ID, Client\_Name, Client\_Contact) VALUES (3, 'TechSolutions', 'tech@solutions.com');

-- Insert sample data into Project table

INSERT INTO Project (Project\_ID, Project\_Name, Start\_Date, End\_Date, Client\_ID) VALUES (1, 'Website Redesign', TO\_DATE('2021-06-01', 'YYYY-MM-DD'), TO\_DATE('2021-12-31', 'YYYY-MM-DD'), 1);

INSERT INTO Project (Project\_ID, Project\_Name, Start\_Date, End\_Date, Client\_ID) VALUES (2, 'Mobile App Development', TO\_DATE('2022-01-15', 'YYYY-MM-DD'), TO\_DATE('2022-08-15', 'YYYY-MM-DD'), 2);

INSERT INTO Project (Project\_ID, Project\_Name, Start\_Date, End\_Date, Client\_ID) VALUES (3, 'Software Maintenance', TO\_DATE('2021-07-01', 'YYYY-MM-DD'), TO\_DATE('2022-06-30', 'YYYY-MM-DD'), 3);

-- Insert sample data into Task table

INSERT INTO Task (Task\_ID, Task\_Name, Project\_ID, Assigned\_Employee\_ID, Start\_Date, End\_Date, Status) VALUES (1, 'Frontend Development', 1, 1, TO\_DATE('2021-06-01', 'YYYY-MM-DD'), TO\_DATE('2021-09-30', 'YYYY-MM-DD'), 'Completed');

INSERT INTO Task (Task\_ID, Task\_Name, Project\_ID, Assigned\_Employee\_ID, Start\_Date, End\_Date, Status) VALUES (2, 'Backend Development', 1, 5, TO\_DATE('2021-06-01', 'YYYY-MM-DD'), TO\_DATE('2021-09-30', 'YYYY-MM-DD'), 'In Progress');

INSERT INTO Task (Task\_ID, Task\_Name, Project\_ID, Assigned\_Employee\_ID, Start\_Date, End\_Date, Status) VALUES (3, 'Testing', 2, 2, TO\_DATE('2022-01-16', 'YYYY-MM-DD'), TO\_DATE('2022-07-30', 'YYYY-MM-DD'), 'Completed');

INSERT INTO Task (Task\_ID, Task\_Name, Project\_ID, Assigned\_Employee\_ID, Start\_Date, End\_Date, Status) VALUES (4, 'Mobile Development', 2, 1, TO\_DATE('2022-01-16', 'YYYY-MM-DD'), TO\_DATE('2022-08-01', 'YYYY-MM-DD'), 'Completed');

INSERT INTO Task (Task\_ID, Task\_Name, Project\_ID, Assigned\_Employee\_ID, Start\_Date, End\_Date, Status) VALUES (5, 'Maintenance and Updates', 3, 3, TO\_DATE('2021-07-01', 'YYYY-MM-DD'), TO\_DATE('2022-06-30', 'YYYY-MM-DD'), 'In Progress');

-- Insert sample data into Time\_Log table

INSERT INTO Time\_Log (Time\_Log\_ID, Task\_ID, Employee\_ID, Hours\_Worked, Log\_Date) VALUES (1, 1, 1, 40, TO\_DATE('2021-06-01', 'YYYY-MM-DD'));

INSERT INTO Time\_Log (Time\_Log\_ID, Task\_ID, Employee\_ID, Hours\_Worked, Log\_Date) VALUES (2, 2, 5, 45, TO\_DATE('2021-06-15', 'YYYY-MM-DD'));

INSERT INTO Time\_Log (Time\_Log\_ID, Task\_ID, Employee\_ID, Hours\_Worked, Log\_Date) VALUES (3, 3, 2, 38, TO\_DATE('2022-01-20', 'YYYY-MM-DD'));

INSERT INTO Time\_Log (Time\_Log\_ID, Task\_ID, Employee\_ID, Hours\_Worked, Log\_Date) VALUES (4, 4, 1, 50, TO\_DATE('2022-02-01', 'YYYY-MM-DD'));

INSERT INTO Time\_Log (Time\_Log\_ID, Task\_ID, Employee\_ID, Hours\_Worked, Log\_Date) VALUES (5, 5, 3, 35, TO\_DATE('2021-07-15', 'YYYY-MM-DD'));

-- Insert sample data into Deliverable table

INSERT INTO Deliverable (Deliverable\_ID, Project\_ID, Description, Delivery\_Date) VALUES (1, 1, 'Completed Website Design', TO\_DATE('2021-09-30', 'YYYY-MM-DD'));

INSERT INTO Deliverable (Deliverable\_ID, Project\_ID, Description, Delivery\_Date) VALUES (2, 2, 'Mobile App Prototype', TO\_DATE('2022-07-15', 'YYYY-MM-DD'));

INSERT INTO Deliverable (Deliverable\_ID, Project\_ID, Description, Delivery\_Date) VALUES (3, 3, 'Software Update Version 1.1', TO\_DATE('2022-05-15', 'YYYY-MM-DD'));

Q1: Write an SQL query to find the total number of hours worked by each employee across all tasks.

Q2: Retrieve all projects that were completed after 2020, including their project name and client name.

Q3: List all employees who have worked on a project that was completed before 2021. Show the employee name, project name, and task status.

Q4: Find the total number of tasks completed in the "Development" department, and show the department name and total tasks completed.

Q5: Retrieve all tasks with their assigned employees, but only for tasks that have been assigned to employees with the role 'Developer'.

Q6: Find all clients who have projects with deliverables due in the next 30 days. Show the client name, project name, and deliverable description.

Q7: List all employees who have worked more than 40 hours in a week (considering time logs within the same week).

Q8: Retrieve the department-wise total salary paid to employees, and show the department name along with the total salary amount.

Q9: Find all employees who have not worked on any tasks from the project "Website Redesign".

Q10: List all tasks with their corresponding project names, along with the number of hours logged for each task.