**DBMS Practical No: 3**

**# Write the SQL queries using of concepts like all types of Join, Sub- Query and View**

* **Some Important Things:**

**a. INNER JOIN:**

**An INNER JOIN returns only the rows that have matching values in both tables.**

**b. LEFT JOIN (or LEFT OUTER JOIN):**

**A LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there's no match, NULL values are returned for right table columns.**

**c. RIGHT JOIN (or RIGHT OUTER JOIN):**

**A RIGHT JOIN is similar to a LEFT JOIN, but it returns all rows from the right table and the matched rows from the left table.**

**d. FULL JOIN (or FULL OUTER JOIN):**

**A FULL JOIN returns all rows when there is a match in either the left or right table. It includes rows that don't have matches in both tables.**

**2. Sub-Queries:**

**Sub-queries, or nested queries, are queries within another query. They are used to retrieve data based on the results of another query.**

**a. Scalar Sub-Query:**

**A scalar sub-query returns a single value. For example, to find the highest salary in a table:**

**b. Correlated Sub-Query:**

**A correlated sub-query refers to a column in the outer query, allowing you to filter results based on the outer query. For example, to find employees with salaries greater than the average salary in their department:**

**3. Views:**

**A view is a virtual table based on the result of a SELECT query. It can simplify complex queries and provide a more user-friendly interface to the data.**

**Execution Starts 🡪**

**1. Employees Table:**

sql

**CREATE TABLE employees ( employee\_id INT AUTO\_INCREMENT PRIMARY KEY, first\_name VARCHAR(50), last\_name VARCHAR(50), department\_id INT, salary DECIMAL(10, 2) );**

**2. Departments Table:**

sql

**CREATE TABLE departments ( department\_id INT AUTO\_INCREMENT PRIMARY KEY, department\_name VARCHAR(50) );**

Let's perform the practical exercises step by step:

**Step 1: Populate the Tables with Data**

Insert some sample data into the tables:

sql

**-- Insert employees**

**INSERT INTO employees (first\_name, last\_name, department\_id, salary)**

**VALUES ('John', 'Doe', 1, 50000.00), ('Jane', 'Smith', 2, 60000.00), ('Bob', 'Johnson', 1, 55000.00);**

**-- Insert departments**

**INSERT INTO departments (department\_name) VALUES ('HR'), ('Engineering');**

**Step 2: SQL Joins**

**a. INNER JOIN:**

**Retrieve the employee names and their department names using an INNER JOIN:**

sql

**SELECT e.first\_name, e.last\_name, d.department\_name FROM employees e INNER JOIN departments d ON e.department\_id = d.department\_id;**

**b. LEFT JOIN:**

**List all employees and their respective departments. If an employee doesn't belong to a department, display "No Department."**

sql

**SELECT e.first\_name, e.last\_name, IFNULL(d.department\_name, 'No Department') AS department\_name FROM employees e LEFT JOIN departments d ON e.department\_id = d.department\_id;**

**Step 3: Sub-Queries**

**a. Scalar Sub-Query:**

**Find the highest salary among all employees using a scalar sub-query:**

sql

**SELECT MAX(salary) FROM employees;**

**b. Correlated Sub-Query:**

**Retrieve employees with salaries greater than the average salary in their respective departments:**

sql

**SELECT e.first\_name, e.last\_name, e.salary, d.department\_name FROM employees e JOIN departments d ON e.department\_id = d.department\_id WHERE e.salary > (SELECT AVG(salary) FROM employees e2 WHERE e2.department\_id = d.department\_id);**

**Step 4: Views**

**Creating a View:**

**Create a view to show the high-paid employees (those with a salary greater than 55000):**

sql

**CREATE VIEW high\_paid\_employees AS SELECT first\_name, last\_name, salary FROM employees WHERE salary > 55000;**

Now, you can query the "high\_paid\_employees" view as if it were a regular table:

sql

**SELECT \* FROM high\_paid\_employees;**