**3. SQL Queries - all types of Join, Sub-Query and View:**

**Write at least 10 SQL queries for suitable database application using SQL DML statements.**

**Step 1: Create the Tables**

1. Create Department Table

CREATE TABLE Department (

dept\_no INT PRIMARY KEY,

dept\_name VARCHAR(30) UNIQUE NOT NULL,

bldg\_name VARCHAR(30)

);

2. Create Instructor Table

CREATE TABLE Instructor (

ins\_id INT PRIMARY KEY,

ins\_name VARCHAR(30) NOT NULL,

dept\_no INT,

salary DECIMAL(10, 2),

FOREIGN KEY (dept\_no) REFERENCES Department(dept\_no)

);

3. Create Course Table

CREATE TABLE Course (

course\_id INT PRIMARY KEY,

title VARCHAR(30) NOT NULL,

dept\_no INT,

credits INT,

FOREIGN KEY (dept\_no) REFERENCES Department(dept\_no)

);

4. Create Enrollment Table

CREATE TABLE Enrollment (

enrollment\_id INT PRIMARY KEY,

stud\_id INT,

course\_id INT,

grade CHAR(1),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id)

);

**Step 2: Insert Sample Data**

Insert Data into Department

INSERT INTO Department (dept\_no, dept\_name, bldg\_name) VALUES

(1, 'Computer Science', 'B1'),

(2, 'Electronics', 'B2'),

(3, 'Information Technology', 'B3');

Insert Data into Instructor

INSERT INTO Instructor (ins\_id, ins\_name, dept\_no, salary) VALUES

(101, 'Alice', 1, 80000),

(102, 'Bob', 2, 75000),

(103, 'Charlie', 1, 90000),

(104, 'David', 3, 70000);

Insert Data into Course

INSERT INTO Course (course\_id, title, dept\_no, credits) VALUES

(201, 'Data Structures', 1, 3),

(202, 'Circuit Theory', 2, 4),

(203, 'Web Development', 1, 3),

(204, 'Database Systems', 3, 4);

Insert Data into Enrollment

INSERT INTO Enrollment (enrollment\_id, stud\_id, course\_id, grade) VALUES

(1, 1001, 201, 'A'),

(2, 1002, 202, 'B'),

(3, 1001, 203, 'A'),

(4, 1003, 204, 'C');

**Step 3: SQL Queries Demonstrating Joins, Sub-Queries, and Views**

**1. Inner Join**

Retrieve a list of instructors along with their department names.

SELECT i.ins\_name, d.dept\_name

FROM Instructor i

INNER JOIN Department d ON i.dept\_no = d.dept\_no;

**2. Left Join**

Get all departments and their instructors, including departments without instructors.

SELECT d.dept\_name, i.ins\_name

FROM Department d

LEFT JOIN Instructor i ON d.dept\_no = i.dept\_no;

**3. Right Join**

List all instructors and their departments, including instructors who may not belong to any department.

SELECT i.ins\_name, d.dept\_name

FROM Instructor i

RIGHT JOIN Department d ON i.dept\_no = d.dept\_no;

**4.inner join**

SELECT c.title AS Course\_Title, i.ins\_name AS Instructor\_Name

FROM Course c

INNER JOIN Instructor i ON c.dept\_no = i.dept\_no

WHERE c.credits >= 3;

**5. Cross Join**

Generate a list of all possible combinations of instructors and courses.

SELECT i.ins\_name, c.title

FROM Instructor i

CROSS JOIN Course c;

**6. Sub-Query**

Find the names of instructors who earn more than the average salary of all instructors.

SELECT ins\_name

FROM Instructor

WHERE salary > (SELECT AVG(salary) FROM Instructor);

**7. Sub-Query with EXISTS**

List all departments that have at least one instructor.

SELECT dept\_name

FROM Department d

WHERE EXISTS (SELECT 1 FROM Instructor i WHERE i.dept\_no = d.dept\_no);

**8. View Creation**

Create a view to display a list of instructors and their corresponding department names.

CREATE VIEW Instructor\_View AS

SELECT i.ins\_name, d.dept\_name

FROM Instructor i

JOIN Department d ON i.dept\_no = d.dept\_no;

**9. Using the View**

Retrieve all data from the Instructor\_View.

SELECT \* FROM Instructor\_View;

**10. Nested Sub-Query**

Get the titles of courses that are taught by instructors in the "Computer Science" department.

SELECT title

FROM Course

WHERE dept\_no = (SELECT dept\_no FROM Department WHERE dept\_name = 'Computer Science');

**Theory: SQL Joins and Conditional Queries**

In relational database management systems, data is organized into tables that can be related to one another. To retrieve meaningful information from multiple tables, SQL provides several types of joins, which allow us to combine rows from two or more tables based on related columns.

**Types of Joins**

Inner Join:

An inner join retrieves only the rows that have matching values in both tables. It is commonly used when we want to combine related data from two tables.

Example: In our query, we used an inner join between the Course table and the Instructor table based on the dept\_no column. This allows us to fetch courses and their respective instructors who belong to the same department.

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 is no match, NULL values are returned for columns from the right table.

Right Join (or Right Outer Join):

A right join is the opposite of a left join. It returns all rows from the right table and the matched rows from the left table.

Full Outer Join:

A full outer join combines the results of both left and right joins, returning all records when there is a match in either left or right table.