**Perform following SQL queries on the database created in assignment 1.**

**• Implementation of relational operators in SQL**

**• Boolean operators and pattern matching**

**• Arithmetic operations and built in functions**

**• Group functions**

**• Processing Date and Time functions**

**• Complex queries and set operators**

**Ans :**

**-- Step 1: Create the database**

CREATE DATABASE IndianCollegeDB;

**-- Step 2: Use the database**

USE IndianCollegeDB;

**-- Step 3: Create the Instructor table with Primary Key**

CREATE TABLE Instructor (

InstructorID INT PRIMARY KEY,

Name VARCHAR(100),

Department VARCHAR(100)

);

**-- Step 4: Create the Student table with Primary Key**

CREATE TABLE Student (

StudentID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100)

);

**-- Step 5: Create the Course table with Foreign Key reference to Instructor table**

CREATE TABLE Course (

CourseID INT PRIMARY KEY,

Title VARCHAR(100),

InstructorID INT,

FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID)

);

**-- Step 6: Create the Enrollment table (junction table) with Foreign Keys**

CREATE TABLE Enrollment (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

EnrollmentDate DATE,

FOREIGN KEY (StudentID) REFERENCES Student(StudentID),

FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

);

**-- Step 7: Insert sample data into Instructor table (using Indian names)**

INSERT INTO Instructor (InstructorID, Name, Department) VALUES

(1, 'Dr. Rajesh Kumar', 'Computer Science'),

(2, 'Prof. Meena Agarwal', 'Mathematics'),

(3, 'Dr. Arvind Sharma', 'Physics');

**-- Step 8: Insert sample data into Student table (using Indian names)**

INSERT INTO Student (StudentID, Name, Email) VALUES

(1, 'Amit Patel', 'amit.patel@example.com'),

(2, 'Priya Sharma', 'priya.sharma@example.com'),

(3, 'Ravi Kumar', 'ravi.kumar@example.com'),

(4, 'Neha Singh', 'neha.singh@example.com'),

(5, 'Vikram Joshi', 'vikram.joshi@example.com');

**-- Step 9: Insert sample data into Course table**

INSERT INTO Course (CourseID, Title, InstructorID) VALUES

(1, 'Introduction to Programming', 1),

(2, 'Data Structures', 1),

(3, 'Calculus I', 2),

(4, 'Quantum Mechanics', 3),

(5, 'Linear Algebra', 2);

**-- Step 10: Insert sample data into Enrollment table**

INSERT INTO Enrollment (EnrollmentID, StudentID, CourseID, EnrollmentDate) VALUES

(1, 1, 1, '2025-04-01'),

(2, 2, 2, '2025-04-02'),

(3, 3, 3, '2025-04-03'),

(4, 4, 4, '2025-04-04'),

(5, 5, 5, '2025-04-05');

**-- -- Step 11: Relational Operators**

**-- Find all students whose names start with 'A' and whose StudentID is greater than 2**

SELECT \* FROM Student

WHERE Name LIKE 'A%' AND StudentID > 2;

**-- Step 12: Boolean Operators and Pattern Matching**

**-- Find students who either belong to the "Computer Science" department or have an email address containing "example.com"**

SELECT \* FROM Student

WHERE Email LIKE '%example.com%' OR StudentID IN (SELECT StudentID FROM Enrollment WHERE CourseID = 1);

**-- Step 13: Arithmetic Operations and Built-in Functions**

**-- Calculate the total number of students enrolled in each course and display their names in uppercase.**

SELECT C.Title AS CourseTitle, COUNT(E.StudentID) AS TotalStudents,

UPPER(S.Name) AS StudentName

FROM Course C

JOIN Enrollment E ON C.CourseID = E.CourseID

JOIN Student S ON E.StudentID = S.StudentID

GROUP BY C.CourseID, C.Title, S.Name;

**-- Step 14: Group Functions**

**-- Find the average enrollment count per course.**

SELECT C.Title AS CourseTitle, AVG(TotalStudents) AS AvgEnrollment

FROM Course C

JOIN (SELECT CourseID, COUNT(StudentID) AS TotalStudents FROM Enrollment GROUP BY CourseID) AS E

ON C.CourseID = E.CourseID

GROUP BY C.Title;

**-- Step 15: Processing Date and Time Functions**

**-- Find all students who enrolled after '2025-04-03'**.

SELECT S.Name, E.EnrollmentDate

FROM Student S

JOIN Enrollment E ON S.StudentID = E.StudentID

WHERE E.EnrollmentDate > '2025-04-03';

**-- Calculate the number of days between the enrollment date and today's date for each student.**

SELECT S.Name, DATEDIFF(CURDATE(), E.EnrollmentDate) AS DaysSinceEnrollment

FROM Student S

JOIN Enrollment E ON S.StudentID = E.StudentID;

**-- Step 16: Complex Queries and Set Operators**

-- Find the names of students who are enrolled in both 'Introduction to Programming' and 'Data Structures' (using INTERSECT).

SELECT S.Name

FROM Student S

JOIN Enrollment E ON S.StudentID = E.StudentID

JOIN Course C ON E.CourseID = C.CourseID

WHERE C.Title = 'Introduction to Programming'

INTERSECT

SELECT S.Name

FROM Student S

JOIN Enrollment E ON S.StudentID = E.StudentID

JOIN Course C ON E.CourseID = C.CourseID

WHERE C.Title = 'Data Structures';

-- Find all students who are enrolled in either 'Introduction to Programming' or 'Data Structures' (using UNION).

SELECT S.Name

FROM Student S

JOIN Enrollment E ON S.StudentID = E.StudentID

JOIN Course C ON E.CourseID = C.CourseID

WHERE C.Title = 'Introduction to Programming'

UNION

SELECT S.Name

FROM Student S

JOIN Enrollment E ON S.StudentID = E.StudentID

JOIN Course C ON E.CourseID = C.CourseID

WHERE C.Title = 'Data Structures';

**Explanation of the Code**

1. **Database Creation & Table Definitions**:
   * We create a new database IndianCollegeDB.
   * We define the tables Instructor, Student, Course, and Enrollment, with their respective primary keys and foreign key relationships.
2. **Data Insertion**:
   * Sample data is inserted into the tables. The Instructor table contains information about instructors, the Student table contains student details, the Course table contains course details, and the Enrollment table tracks which students are enrolled in which courses.
3. **SQL Operations**:
   * **Relational Operators**: We use relational operators (LIKE, >, etc.) to filter students whose names start with 'A' and whose StudentID is greater than 2.
   * **Boolean Operators and Pattern Matching**: We combine conditions with OR and use LIKE to find students whose email contains "example.com".
   * **Arithmetic Operations and Built-in Functions**: We calculate the total number of students per course and display their names in uppercase.
   * **Group Functions**: We calculate the average number of students enrolled in each course.
   * **Date and Time Functions**: We find students who enrolled after a specific date and calculate the days since enrollment using DATEDIFF().
   * **Complex Queries and Set Operators**: We use INTERSECT to find students enrolled in both 'Introduction to Programming' and 'Data Structures', and UNION to find students enrolled in either course.