Lab Report

Course Code: CSE-312

Course Title: Database Management System Lab

Submitted To:

Mr. Mahmudul Islam Rakib

Lecturer

Department of CSE

Daffodil International University

Submitted By

Raisul Islam Nahid

02422200051013347

63_D1

Department of CSE

Daffodil International University

Submission Date: 10 December 2024

Scetion A

1. Database Creation and Schema Setup

```
CREATE DATABASE UniversityDB;


CREATE TABLE Faculty (
   FacultyID INT PRIMARY KEY,
   Name VARCHAR(100),
   Age INT,
   Designation VARCHAR(50),
   Salary DECIMAL(10,2)
);

CREATE TABLE Department (
   DepartmentID INT PRIMARY KEY,
   DeptName VARCHAR(100),
   HeadID INT,
   FOREIGN KEY (HeadID) REFERENCES Faculty(FacultyID)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);



CREATE TABLE Student (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(100),
   Age INT,
   Gender VARCHAR(10),
   DepartmentID INT,
```

```
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);



CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    Credits INT,
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);



CREATE TABLE Enrollment (
    StudentID INT,
    CourseID INT,
    Grade VARCHAR(2),
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
    ON DELETE CASCADE
```

```
    ON UPDATE CASCADE   );
```

2. Data insertion

```
INSERT INTO Faculty VALUES
(1, 'Dr. John Smith', 45, 'Associate Professor', 75000),
(2, 'Dr. Emily Wong', 50, 'Professor', 90000),
(3, 'Prof. Michael Lee', 38, 'Assistant Professor', 60000),
(4, 'Dr. Sarah Johnson', 55, 'Professor', 95000),
(5, 'Prof. David Kim', 42, 'Associate Professor', 78000);
```

```
INSERT INTO Department VALUES
(1, 'Computer Science', 2),
(2, 'Mathematics', 4),
(3, 'Physics', 1),
(4, 'Biology', 3),
(5, 'Chemistry', 5);
```

```
INSERT INTO Student VALUES
(1, 'Alex Chen', 20, 'Male', 1),
(2, 'Emma Rodriguez', 22, 'Female', 1),
(3, 'Ryan Patel', 21, 'Male', 2),
(4, 'Sophia Kim', 19, 'Female', 3),
(5, 'Daniel Wu', 23, 'Male', 4);
```

```
INSERT INTO Course VALUES
(1, 'DBMS Lab', 4, 1),
(2, 'Linear Algebra', 3, 2),
(3, 'Quantum Physics', 4, 3),
(4, 'Molecular Biology', 3, 4),
(5, 'Organic Chemistry', 4, 5);

INSERT INTO Enrollment VALUES
(1, 1, 'A'),
(2, 1, 'B'),
(3, 2, 'A'),
(4, 3, 'B'),
(5, 4, 'A');
```

3. Basic Retrieval Queries

```
SELECT s.Name, d.DeptName
FROM Student s
JOIN Department d ON s.DepartmentID = d.DepartmentID;

SELECT c.CourseName
FROM Course c
JOIN Department d ON c.DepartmentID = d.DepartmentID
WHERE d.DeptName = 'Computer Science';
```

Scetion B

4. Schema Modification

```
ALTER TABLE Student
ADD COLUMN PhoneNumber VARCHAR(15);

ALTER TABLE Student
CHANGE COLUMN Gender Sex VARCHAR(10);
```

5. Data Updates and Deletions

```
UPDATE Enrollment e
JOIN Course c ON e.CourseID = c.CourseID
SET e.Grade = 'B'
WHERE c.CourseName = 'DBMS Lab';
DELETE FROM Course WHERE CourseName = 'NLP';
```

Section C

6. Stored Procedure

```
DELIMITER //
CREATE PROCEDURE Getavg(IN dept_id INT)
BEGIN
    SELECT AVG(Age) AS AverageAge
    FROM Student
    WHERE DepartmentID = dept_id;
END//
DELIMITER ;
```

7. View

```
CREATE VIEW TopStudents AS
SELECT DISTINCT Student.Name, Course.CourseName
FROM Student
JOIN Enrollment ON Student.StudentID = Enrollment.StudentID
JOIN Course ON Enrollment.CourseID = Course.CourseID
WHERE Enrollment.Grade = 'A';
```

8. Trigger

```
Delimiter//
CREATE TRIGGER UpdateSalaryOnPromotionToProf
BEFORE UPDATE ON Faculty
FOR EACH ROW
BEGIN
   IF OLD.Designation != 'Professor' AND NEW.Designation = 'Professor' THEN
      SET NEW.Salary = NEW.Salary * 1.1;
   END IF;
END // Delimiter ;
```

9. Join

```
SELECT s.Name, d.DeptName, c.CourseName
FROM Student s
JOIN Department d ON s.DepartmentID = d.DepartmentID
JOIN Enrollment e ON s.StudentID = e.StudentID
JOIN Course c ON e.CourseID = c.CourseID;
```

10.Nested Query

```
SELECT s.Name
FROM Student s
```

```
JOIN Enrollment e ON s.StudentID = e.StudentID
JOIN Course c ON e.CourseID = c.CourseID
WHERE s.Age = (
  SELECT MIN(Age)
  FROM Student s2
  JOIN Enrollment e2 ON s2.StudentID = e2.StudentID
  JOIN Course c2 ON e2.CourseID = c2.CourseID
  WHERE c2.CourseName = 'DBMS'
);
```

11.GROUP BY and HAVING

```
SELECT d.DeptName, COUNT(s.StudentID) AS StudentCount
FROM Department d
JOIN Student s ON d.DepartmentID = s.DepartmentID
GROUP BY d.DeptName
HAVING StudentCount > 5;
```

12.Correlated Subquery

```
SELECT f.Name
FROM Faculty f
WHERE EXISTS (
  SELECT 1
  FROM Department d
  JOIN Course c ON d.DepartmentID = c.DepartmentID
  WHERE d.HeadID = f.FacultyID
  GROUP BY d.DepartmentID
  HAVING COUNT(c.CourseID) > 3
);
```