University Database Management System
1. Student Management: Store student details such as StudentID, Name, Age, Gender, Department, and Email.
2. Course Management: Maintain course details including CourseID, CourseName, Credits, and
Department.
3. Enrollment System: Allow students to enroll in multiple courses, tracking StudentID,
CourseID, EnrollmentDate, and Grade.
4. Professor Management: Store professor details like ProfessorID, Name, Department, and
Email
Write queries for the following questions:
1. List all courses a specific student is enrolled in (e.g., Pooja)
2. Identify students who failed more than 2 courses (assuming grade < 2.0 is fail)
3. Count the number of students in each department
4. Find courses with zero enrollments
5. Find the most popular course (course with the highest number of enrollments)

```
-- Student Management
CREATE TABLE Students (
    StudentID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Age INT,
    Gender VARCHAR(10),
    Department VARCHAR(50),
    Email VARCHAR(100)
);

-- Course Management
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY AUTO_INCREMENT,
    CourseName VARCHAR(100),
    Credits INT,
    Department VARCHAR(50)
);

-- Enrollment System
CREATE TABLE Enrollments (
    StudentID INT,
    CourseID INT,
    EnrollmentDate DATE,
    Grade FLOAT,
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

```sql
-- Professor Management
CREATE TABLE Professors (
    ProfessorID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Email VARCHAR(100)
);

-- Students
INSERT INTO Students (Name, Age, Gender, Department, Email) VALUES
('Pooja', 21, 'Female', 'Computer Science', 'pooja@example.com'),
('Ravi', 22, 'Male', 'Electronics', 'ravi@example.com'),
('Meena', 20, 'Female', 'Mechanical', 'meena@example.com'),
('Amit', 23, 'Male', 'Computer Science', 'amit@example.com');

-- Courses
INSERT INTO Courses (CourseName, Credits, Department) VALUES
('Data Structures', 4, 'Computer Science'),
('Digital Circuits', 3, 'Electronics'),
('Thermodynamics', 4, 'Mechanical'),
('Algorithms', 4, 'Computer Science'),
('Machine Learning', 3, 'Computer Science');

-- Enrollments
INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate, Grade) VALUES
(1, 1, '2025-01-10', 3.5),
(1, 4, '2025-01-12', 2.7),
(2, 2, '2025-01-15', 1.8),
(2, 1, '2025-01-16', 1.5),
(2, 4, '2025-01-17', 1.0),
(2, 5, '2025-01-18', 2.8),
(3, 3, '2025-01-11', 3.0);

INSERT INTO Professors (Name, Department, Email) VALUES
('Dr. Sharma', 'Computer Science', 'sharma.cs@university.edu'),
('Dr. Reddy', 'Electronics', 'reddy.ec@university.edu'),
('Dr. Verma', 'Mechanical', 'verma.me@university.edu'),
('Dr. Khan', 'Computer Science', 'khan.cs@university.edu');

-- 1. List all courses a specific student is enrolled in (e.g., Pooja)
SELECT c.CourseName
FROM Enrollments e
JOIN Students s ON e.StudentID = s.StudentID
JOIN Courses c ON e.CourseID = c.CourseID
WHERE s.Name = 'Pooja';

--  2. Identify students who failed more than 2 courses (grade < 2.0)
SELECT s.Name, COUNT(*) AS FailedCourses
FROM Enrollments e
```

```
JOIN Students s ON e.StudentID = s.StudentID
WHERE e.Grade < 2.0
GROUP BY s.StudentID
HAVING COUNT(*) > 2;

-- 3. Count the number of students in each department
SELECT Department, COUNT(*) AS StudentCount
FROM Students
GROUP BY Department;

-- 4. Find courses with zero enrollments
SELECT CourseName
FROM Courses
WHERE CourseID NOT IN (
    SELECT DISTINCT CourseID FROM Enrollments
);

-- 5. Find the most popular course (highest enrollments)
SELECT c.CourseName, COUNT(e.StudentID) AS EnrollCount
FROM Enrollments e
JOIN Courses c ON e.CourseID = c.CourseID
GROUP BY e.CourseID
ORDER BY EnrollCount DESC
LIMIT 1;
```