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. Find students who have not enrolled in any course
2. Find students who are enrolled in more than 3 courses
3. Find the average grade of students per course
4. Retrieve the highest grade in each course
5. Get the department with the highest number of students

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

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

-- Enrollments
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)
);
```

```
-- Professors
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'),
('Neha', 22, 'Female', 'Computer Science', 'neha@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),
(1, 5, '2025-01-14', 3.2),
(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),
(4, 1, '2025-01-10', 2.9),
(4, 4, '2025-01-12', 3.7),
(4, 5, '2025-01-14', 3.1),
(4, 2, '2025-01-15', 2.5);

INSERT INTO Professors (Name, Department, Email) VALUES
('Dr. Sharma', 'Computer Science', 'sharma@university.edu'),
('Dr. Iyer', 'Electronics', 'iyer@university.edu'),
('Dr. Rao', 'Mechanical', 'rao@university.edu'),
('Dr. Verma', 'Computer Science', 'verma@university.edu'),
('Dr. Joshi', 'Mathematics', 'joshi@university.edu');

-- 1. Find students who have not enrolled in any course
SELECT s.StudentID, s.Name
FROM Students s
```

```
WHERE s.StudentID NOT IN (
    SELECT DISTINCT StudentID FROM Enrollments
);

-- 2. Find students who are enrolled in more than 3 courses
SELECT s.Name, COUNT(e.CourseID) AS CourseCount
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.StudentID
HAVING CourseCount > 3;

-- 3. Find the average grade of students per course
SELECT c.CourseName, ROUND(AVG(e.Grade), 2) AS AvgGrade
FROM Enrollments e
JOIN Courses c ON e.CourseID = c.CourseID
GROUP BY e.CourseID;

-- 4. Retrieve the highest grade in each course
SELECT c.CourseName, MAX(e.Grade) AS HighestGrade
FROM Enrollments e
JOIN Courses c ON e.CourseID = c.CourseID
GROUP BY e.CourseID;

-- 5. Get the department with the highest number of students
SELECT Department, COUNT(*) AS StudentCount
FROM Students
GROUP BY Department
ORDER BY StudentCount DESC
LIMIT 1;
```