University course management system

1)Students table

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Email VARCHAR(100),
   BirthDate DATE,
   EnrollmentDate DATE
);

INSERT INTO Students VALUES
(1, 'John', 'Doe', 'john.doe@email.com', '2000-01-15',
'2022-08-01'),
(2, 'Jane', 'Smith', 'jane.smith@email.com', '1999-05-25',
'2021-08-01’);
```

2)Courses

```
CREATE TABLE Courses (
   CourseID INT PRIMARY KEY,
   CourseName VARCHAR(100),
   DepartmentID INT,
   Credits INT,
   FOREIGN KEY (DepartmentID) REFERENCES
Departments(DepartmentID)
);

INSERT INTO Courses VALUES
(101, 'Introduction to SQL', 1, 3),
(102, 'Data Structures', 2, 4);
```

3)Insturctors

```
CREATE TABLE Instructors (
   InstructorID INT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Email VARCHAR(100),
   DepartmentID INT,
   Salary DECIMAL(10,2),
   FOREIGN KEY (DepartmentID) REFERENCES
Departments(DepartmentID)
);

INSERT INTO Instructors VALUES
(1, 'Alice', 'Johnson', 'alice.johnson@univ.com', 1, 75000),
(2, 'Bob', 'Lee', 'bob.lee@univ.com', 2, 65000);
```

4)Enrollments

```
CREATE TABLE Enrollments (
   EnrollmentID INT PRIMARY KEY,
   StudentID INT,
   CourseID INT,
   EnrollmentDate DATE,
   FOREIGN KEY (StudentID) REFERENCES
Students(StudentID),
   FOREIGN KEY (CourseID) REFERENCES
Courses(CourseID)
);

INSERT INTO Enrollments VALUES
(1, 1, 101, '2022-08-01'),
(2, 2, 102, '2021-08-01’);
```

5)Departments

```
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(100)
);

INSERT INTO Departments VALUES
(1, 'Computer Science'),
(2, 'Mathematics’);
```

Queries to perform

1)

```
-- INSERT
INSERT INTO Students VALUES (3, 'Mark', 'Taylor',
'mark@email.com', '2001-04-10', '2023-08-01');

-- READ
SELECT * FROM Students;

-- UPDATE
UPDATE Students
SET Email = 'mark.taylor@email.com'
WHERE StudentID = 3;

-- DELETE
DELETE FROM Students WHERE StudentID = 3;
```

2)

```
SELECT *
FROM Students
WHERE EnrollmentDate > '2022-12-31';
```

3)

```
SELECT c.*
FROM Courses c
JOIN Departments d ON c.DepartmentID =
d.DepartmentID
WHERE d.DepartmentName = 'Mathematics'
LIMIT 5;
```

4)

```
SELECT CourseID, COUNT(StudentID) AS TotalStudents
FROM Enrollments
GROUP BY CourseID
HAVING COUNT(StudentID) > 5;
```

5)

```
SELECT s.StudentID, s.FirstName, s.LastName
FROM Students s
WHERE s.StudentID IN (
    SELECT e1.StudentID
    FROM Enrollments e1
    JOIN Enrollments e2 ON e1.StudentID = e2.StudentID
    WHERE e1.CourseID = 101
      AND e2.CourseID = 102
);
```

6)

```
SELECT DISTINCT s.StudentID, s.FirstName, s.LastName
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
WHERE e.CourseID IN (101, 102);
```

7)

```sql
SELECT AVG(Credits) AS AverageCredits
FROM Courses;
```

8)

```sql
SELECT MAX(Salary) AS MaxSalary
FROM Instructors i
JOIN Departments d ON i.DepartmentID = d.DepartmentID
WHERE d.DepartmentName = 'Computer Science';
```

9)

```sql
SELECT d.DepartmentName, COUNT(e.StudentID) AS
TotalStudents
FROM Departments d
JOIN Courses c ON d.DepartmentID = c.DepartmentID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY d.DepartmentName;
```

10)

```sql
SELECT s.FirstName, s.LastName, c.CourseName
FROM Students s
INNER JOIN Enrollments e ON s.StudentID = e.StudentID
INNER JOIN Courses c ON e.CourseID = c.CourseID;
```

11)

```sql
SELECT s.FirstName, s.LastName, c.CourseName
FROM Students s
LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
LEFT JOIN Courses c ON e.CourseID = c.CourseID;
```

12)

```sql
SELECT DISTINCT s.StudentID, s.FirstName, s.LastName
FROM Students s
```

```
JOIN Enrollments e ON s.StudentID = e.StudentID
WHERE e.CourseID IN (
   SELECT CourseID
   FROM Enrollments
   GROUP BY CourseID
   HAVING COUNT(StudentID) > 10
);
```

13)

```
SELECT StudentID, YEAR(EnrollmentDate) AS
EnrollmentYear
FROM Students;
```

14)

```
SELECT CONCAT(FirstName, ' ', LastName) AS
InstructorName
FROM Instructors;
```

15)

```
SELECT CourseID,
      COUNT(StudentID) AS StudentsPerCourse,
      SUM(COUNT(StudentID)) OVER (ORDER BY
CourseID) AS RunningTotal
FROM Enrollments
GROUP BY CourseID;
```

16)

```
SELECT StudentID, FirstName, LastName,
CASE
   WHEN EnrollmentDate <= DATE_SUB(CURDATE(),
INTERVAL 4 YEAR)
   THEN 'Senior'
```

```
    ELSE 'Junior'
END AS Status
FROM Students;
```