**Student Management System – SQL Project**

**Project Overview**

The Student Management System SQL project demonstrates how SQL can be used to design and manage a relational database for storing and retrieving academic data.  
It covers table creation, data insertion, and execution of practical queries to manage students, courses, grades, and instructors efficiently.  
The system supports administrative needs like generating academic performance reports, identifying top students, and managing course enrollment records.

**Objectives**

* Design relational database tables with primary and foreign keys.
* Perform CRUD operations (Create, Read, Update, Delete) efficiently.
* Implement joins, grouping, and aggregate functions for meaningful insights.
* Demonstrate practical SQL skills in an academic context.

**Database Design**

**Entities:**

* **Students**: Stores student details.
* **Courses**: Stores course information.
* **Instructors**: Stores details of instructors.
* **Enrollments**: Links students and courses with grades.

**SQL Implementation**

CREATE DATABASE StudentDB;

USE StudentDB;

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(50),

Email VARCHAR(50),

EnrollmentDate DATE

);

CREATE TABLE Instructors (

InstructorID INT PRIMARY KEY,

Name VARCHAR(50),

Department VARCHAR(30)

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100),

InstructorID INT,

Credits INT,

FOREIGN KEY (InstructorID) REFERENCES Instructors(InstructorID)

);

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

Grade DECIMAL(4,2),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

);

INSERT INTO Students VALUES

(1, 'Manaswini’, manu@example.com', '2025-01-15'),

(2, 'Hemesh’, hemesh@example.com', '2025-01-18'),

(3, ‘Sunny’, sunny@example.com', '2025-02-05');

INSERT INTO Instructors VALUES

(101, 'Dr. Suresh Rao', 'Computer Science'),

(102, 'Prof. Kavita', 'Mathematics');

INSERT INTO Courses VALUES

(201, 'Database Systems', 101, 4),

(202, 'Calculus II', 102, 3),

(203, 'Data Structures', 101, 4);

INSERT INTO Enrollments VALUES

(1001, 1, 201, 89.5),

(1002, 1, 202, 76.0),

(1003, 2, 203, 92.0),

(1004, 3, 201, 84.0);

SELECT s.Name AS Student, c.CourseName, e.Grade

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID;

SELECT s.Name, c.CourseName, e.Grade

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID

WHERE e.Grade >= 90;

SELECT c.CourseName, COUNT(e.StudentID) AS TotalStudents

FROM Courses c

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseName;

SELECT Name FROM Students

WHERE StudentID IN (

SELECT StudentID FROM Enrollments

WHERE CourseID = (SELECT CourseID FROM Courses WHERE CourseName = 'Database Systems')

);

CREATE VIEW HighScorers AS

SELECT s.Name, AVG(e.Grade) AS AverageGrade

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

GROUP BY s.Name

HAVING AVG(e.Grade) > 85;

**Expected Output Examples**

* **List all students with their enrolled courses** →  
  Shows each student’s name, the courses they are enrolled in, and their grades.
* **Find top-performing students** →  
  Displays names, courses, and grades of students who scored 90 or above.
* **Count students per course** →  
  Returns each course name along with the number of enrolled students.
* **Nested query: Students in 'Database Systems'** →  
  Displays names of all students enrolled in the “Database Systems” course.
* **HighScorers view** →  
  Returns student names with their average grades where the average is greater than 85.

**Skills Demonstrated:**

* Relational database design and schema creation.
* Use of primary/foreign keys to maintain integrity.
* CRUD operations for academic data management.
* Joins, grouping, and aggregate functions for reports.
* Creation of views for reusable queries.