Online Course Registration System

---Group 6  
(Roll Numbers: 26–30)

* Group Members:
  + - 26 **– Provat**
    - 27 **– K. Jatin**
    - 28 **– Tharun**
    - 29 **– Alok Yadav**
    - 30 **– Vivek**

# Table of Contents

1. Introduction
2. Database Design
3. Entity-Relationship Diagram
4. SQL Scripts Overview
5. Sample Queries
6. Tools used
7. Conclusion

## **Introduction**

### **Motivation for the Project**

Educational institutions often face difficulties in managing course registrations, tracking student enrollments, and assigning instructors efficiently. Manual systems are time-consuming, prone to errors, and difficult to scale. This project aims to build a structured and reliable system that simplifies academic data management.

### **Brief Overview of Current Manual/Online Systems**

Manual course registration typically involves paperwork, spreadsheets, or basic form submissions, which can lead to issues like data duplication, mismanagement, and lack of real-time updates. Some institutions use partial online systems, but they may lack integration between students, courses, and instructors.

### **Objective of the System**

The objective of this system is to design a relational database that manages students, instructors, and course enrollments efficiently. It allows for seamless tracking of course assignments and student registrations, supports data updates, and enables insightful queries and reports for academic planning and decision-making.

### **Problem Statement**

Design and implement a relational database system to manage student enrollments, course offerings, and instructor assignments in a scalable and efficient manner.

# Database Design

The system consists of four main entities:  
- Students: Stores student information.  
- Courses: Stores course details and assigned instructors.  
- Instructors: Stores instructor information.  
- Enrollments: Tracks which student is enrolled in which course.  
  
Relationships:  
- One-to-many between Instructors and Courses.  
- Many-to-many between Students and Courses (via Enrollments).

# SQL Scripts Overview/Procedure

- create\_tables.sql: Defines the schema and constraints.  
- insert\_data.sql: Adds sample data.  
- queries.sql: Includes SELECT, UPDATE, DELETE, and advanced queries.  
- stored\_procedures.sql: Contains reusable procedures.  
- triggers.sql: Automates tasks like logging and constraint enforcement.

# Entity-Relationship Diagram

# 

# Sample Queries

1. List all students enrolled in a specific course

This query retrieves the names and IDs of students who are enrolled in a given course.

SQL:

SELECT s.StudentID, s.Name

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

WHERE e.CourseID = 'CSE101';

2. Retrieve course details along with the instructor’s name

This query provides course information along with the name of the instructor assigned to each course.

SQL:

SELECT c.CourseID, c.CourseName, c.Credits, i.Name AS InstructorName

FROM Courses c

JOIN Instructors i ON c.InstructorID = i.InstructorID;

3. Find courses with the highest number of enrollments

This query finds the course that has the most student enrollments.

SQL:

SELECT c.CourseID, c.CourseName, COUNT(e.EnrollmentID) AS TotalEnrollments

FROM Courses c

JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseID, c.CourseName

ORDER BY TotalEnrollments DESC

LIMIT 1;

4. Update an instructor’s details

This query updates the name and department of a specific instructor.

SQL:

UPDATE Instructors

SET Name = 'Dr. Rahul Sharma', Department = 'Computer Science'

WHERE InstructorID = 'I102';

5. Generate a report of all enrollments in the last semester

Assuming the last semester is from July 1, 2024 to December 31, 2024.

SQL:

SELECT e.EnrollmentID, s.Name AS StudentName, c.CourseName, e.EnrollmentDate

FROM Enrollments e

JOIN Students s ON e.StudentID = s.StudentID

JOIN Courses c ON e.CourseID = c.CourseID

WHERE e.EnrollmentDate BETWEEN '2024-07-01' AND '2024-12-31';

6. Delete a student enrollment record

This query deletes a specific enrollment record based on the enrollment ID.

SQL:

DELETE FROM Enrollments

WHERE EnrollmentID = 'E105';

7. Get students who have enrolled in more than 3 courses

This query lists students who have registered for more than three courses.

SQL:

SELECT s.StudentID, s.Name, COUNT(e.CourseID) AS CoursesEnrolled

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

GROUP BY s.StudentID, s.Name

HAVING COUNT(e.CourseID) > 3;

8. Identify courses with low enrollment

This query identifies courses with fewer than 5 students enrolled.

SQL:

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

FROM Courses c

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

GROUP BY c.CourseID, c.CourseName

HAVING COUNT(e.StudentID) < 5;

9. Retrieve instructors who teach multiple courses

This query finds instructors assigned to more than one course.

SQL:

SELECT i.InstructorID, i.Name, COUNT(c.CourseID) AS TotalCourses

FROM Instructors i

JOIN Courses c ON i.InstructorID = c.InstructorID

GROUP BY i.InstructorID, i.Name

HAVING COUNT(c.CourseID) > 1;

10. Find students who have taken the most credits

This query returns the student who has earned the highest total credits across all enrolled courses.

SQL:

SELECT s.StudentID, s.Name, SUM(c.Credits) AS TotalCredits

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID

GROUP BY s.StudentID, s.Name

ORDER BY TotalCredits DESC

LIMIT 1;

Tools & Technologies Used  
Mention database used

* + Database: MySQL 8.0
  + Query Editor: MySQL Workbench
  + Language: SQL
  + ER Diagram Tool: MySQL Workbench (Reverse Engineering)

## **Conclusion**

The Online Course Registration System successfully demonstrates the practical application of relational database concepts in solving real-world academic management problems. By designing and implementing structured tables for students, courses, instructors, and enrollments, the system ensures data integrity, consistency, and ease of access.

With the help of SQL queries, the system efficiently performs key operations such as enrollment tracking, course-instructor mapping, report generation, and data updates. The project not only improves understanding of database normalization and relationships but also highlights the importance of query optimization and data handling.

This mini project lays a solid foundation for further enhancements like a user-friendly interface, login system, and role-based access, which can transform it into a fully functional academic portal. Overall, the project fulfills its objectives and provides a scalable solution for course registration management in educational institutions.