# Student Course Registration System

**Overview**

Design and implement a relational database for managing student registrations, course offerings, instructor assignments, and grades in a fictional university. This project-based course will walk learners through schema design, table creation, data manipulation, querying, and integrity enforcement.

Install any of the following tools:

- PostgreSQL, Microsoft SQL Server, MySQL or SQLite

| Week | Step | Focus | Key Deliverables |
|---|---|---|---|
| 1 | Step 1 | Problem Analysis | Entity list |
| 1 | Step 2 | ERD Design | ERD diagram with relationships |
| 1 | Step 3 | Schema Implementation | SQL CREATE TABLE scripts |
| 1 | Step 4 | Constraints | Schema enforcement for integrity |
| 2 | Step 5 | Data Population | SQL INSERT scripts |
| 2 | Step 6 | SQL Querying | Practical SQL use case queries |
| 2 | Step 7 | Views (Optional) | Logical layers for users/admins |

**Week 1: Data Modelling, ERD, and Schema Implementation**

**Goal: Understand the problem domain, design the data model using ERD, and implement the schema in SQL and enforce business rules using constraints.**

**Step 1: Analyze the Problem Domain**

- Objective: Understand what a Student Course Registration System must support.
- Tasks:
    - Identify key entities: Students, Courses, Instructors, Course Offerings, Enrollments, Prerequisites.
    - Determine what information needs to be stored for each entity.
    - Think about how these entities interact.

a) Students Table

- StudentID (Primary Key)
- FirstName
- LastName
- Email
- Phone
- Major
- Year (Year 1, Year 2, Year 3, Year 4)

b) Courses Table

- CourseID (Primary Key)
- CourseCode (e.g., CS101)
- Title
- Credits
- Department

c) CourseOfferings Table

- OfferingID (Primary Key)
- CourseID (Foreign Key referencing Courses table)
- Semester
- InstructorID (Foreign Key referencing Instructors table)
- Schedule (e.g., "Mon/Wed 10:00-11:30")

d) Instructors Table

- InstructorID (Primary Key)
- FirstName
- LastName
- Email
- Department

e) Enrollments Table

- EnrollmentID (Primary Key)
- StudentID (Foreign Key referencing Students table)
- OfferingID (Foreign Key referencing CourseOfferings table)
- EnrollmentDate
- Grade (can be NULL initially)

f) Prerequisites Table

- CourseID (Foreign Key referencing Courses table)
- PrerequisiteID (Foreign Key referencing Courses table)
- *(Composite Primary Key: CourseID + PrerequisiteID)*

**Step 2: Design the ERD (Entity-Relationship Diagram)**

- Objective: Create a visual representation of the database structure.
- Tasks:
    - Define entities and their attributes.
    - Define relationships between entities using:
        - One-to-Many (e.g., one Student can have many Enrollments)
        - Many-to-Many (handled via junction tables like Enrollments or Prerequisites)
    - Apply cardinality (1:1, 1:N, M:N) to each relationship.
- Deliverable: ER diagram using tools like dbdiagram.io, draw.io, Lucidchart, or hand-drawn.

Relationships in ERD:

| **Relationship** | **ERD Representation** |
|---|---|
| Students → Enrollments (1:N) | Each student can enroll in multiple course offerings |
| Enrollments → CourseOfferings (N:1) | Each enrollment is tied to one offering and one student i.e. Many Enrollments connect to one Offering |
| CourseOfferings → Instructors (N:1) | Each Offering is taught by one Instructor |
| Courses → CourseOfferings (1:N) | Each course can be offered multiple times |

| | |
|---|---|
| Courses → Prerequisites (M:N via self-join) | Each course may have multiple prerequisites. Handled with Prerequisites table as self-referencing |

**Step 3: Define the Schema in SQL**

- Objective: Convert ERD into SQL CREATE TABLE statements.
- Tasks:
    - Create all tables with appropriate primary keys.
    - Define data types.
    - Apply foreign keys as per ERD relationships.
    - Add basic integrity constraints (NOT NULL, UNIQUE where necessary).

**Step 4: Apply Constraints for Data Integrity**

A) Primary Keys

- Every table must have a unique, non-null identifier.

B) Foreign Keys

- Maintain referential integrity:
    - StudentID → Students
    - InstructorID → Instructors
    - CourseID → Courses
    - OfferingID → CourseOfferings
    - PrerequisiteID → Courses (self-reference)

C) Check Constraints

- Credits > 0 in Courses
- Grade IN ('A', 'B', 'C', 'D', 'F', NULL)
- Year IN (Year 1, Year 2, Year 3, Year 4)

D) Unique Constraints

- Unique student and instructor emails
- Unique CourseCode values
- Prevent duplicate enrollments

**Week 2: Querying, Data Integrity, and Business Logic**

**Goal: Data population, write effective SQL queries and views (optionally).**

**Step 5: Populate Tables with Sample Data**

- Objective: Insert meaningful test data to simulate a real environment.
- Tasks:
    - Use INSERT INTO statements to add:
        - 50–100 students
        - 50–100 courses
        - 10–20 instructors
        - 10–15 course offerings
        - 20–25 enrollments
        - 3–5 prerequisite rules

**Step 6: Write SQL Queries for Common Tasks**

A) Student Services Queries

- Retrieve all courses a student is enrolled in for a given semester.
- Display a student's transcript (course titles, semesters, grades, GPA).
- Find students who haven't completed prerequisites for a course.

B) Administrative Queries

- List all courses offered in a department in a given semester.
- Generate instructor-wise course loads per semester.
- Find under-enrolled offerings (less than 5 students).

C) Update and Maintain Data

- Update a student’s major.
- Assign grades to students post-semester.

- Drop a student from a course.

**Step 7: Create Views (Optional)**

- StudentTranscriptView: Combines student, course, and grade info with GPA.
- InstructorScheduleView: Lists courses and schedule for each instructor.

**Final Deliverables**

- ERD Diagram
    - Documentation with explanation of relationships, constraints, and logic
- SQL scripts for table creation and sample data
- SQL queries for common operations