**STUDENT MANAGEMENT SYSTEM**

Project submitted to

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

AMRUTA LAXMI AMMU|AP22110010194

**A picture containing text

Description automatically generated**

Under the Guidance of  
**DR.KAVITHA RANI KAREENA**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 503**

**April, 2024**

**Certificate**

Date: 22-04-2024

This is to certify that the work present in this Project titled “**STUDENT MANAGEMENT SYSTEM**” has been carried out by AMMU AMRUTA LAXMI. The work is genuine, original, and suitable for submission to SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Dr. Kavitha Rani Kareena

Designation,

Affiliation.

**Co-supervisor**

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

# Table of Contents

1. Abstract

2. Project Description

3. ER Diagram

4. Description of ER Diagram

5. Conversion of ER Diagram into Tables

6. Description of Tables

7. Creation of Data in the tables

8. SQL Queries

9. Normalization of Tables Upto 3NF

10. Creation of Views using the Tables

# ABSTRACT :

# The student management system enhances academic efficiency by streamlining academic operations, proficiently managing student records, course offerings, instructor assignments, departmental organization and classroom management. It enhances communication between stakeholders, facilitates accurate course scheduling, and supports curriculum planning. This system's advantages lie in its ability to promote organizational clarity, enhance communication, and optimize resource allocation within educational institutions.

**PROJECT DESCRIPTION:**

The college management system depicted in the Entity-Relationship (ER) diagram is designed to efficiently coordinate various aspects of academic administration, including student information, course offerings, instructor assignments, departmental organization, and classroom management.

The "Students" table serves as a central repository for student data, capturing essential details such as **StudentID, FirstName, LastName, DateOfBirth, Gender, Email, DepartmentID, City,** and **Pincode**. This table facilitates efficient management of student records and enables seamless communication with students.

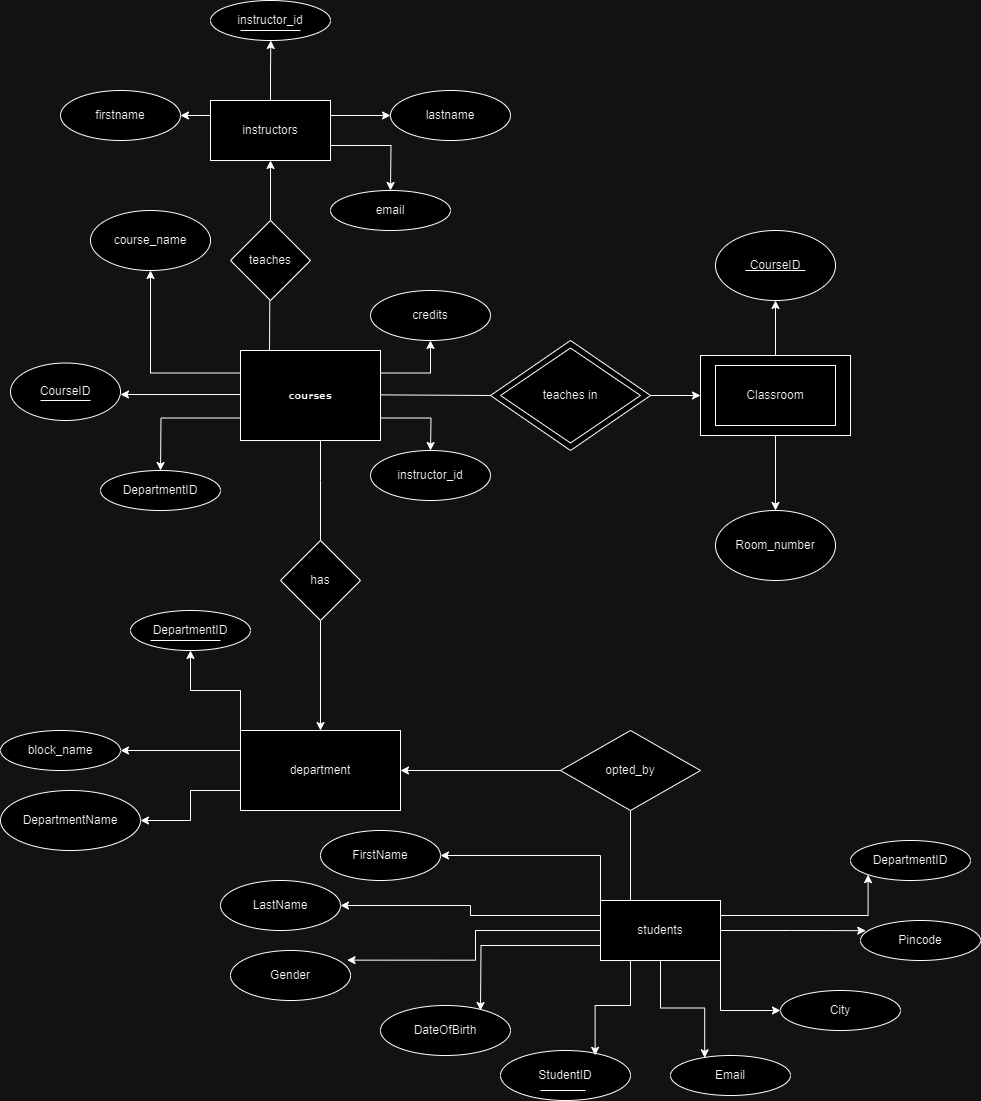
Courses are managed through the "Courses" table, which stores **CourseID, CourseName, DepartmentID, Credits,** and **InstructorID**. The CourseID serves as a unique identifier for each course, while DepartmentID links courses to specific academic departments, promoting organizational clarity and facilitating course categorization.

The "Instructors" table maintains instructor information, including **InstructorID, FirstName, LastName, Email**. This table supports effective management of instructor profiles, contact information, and teaching assignments, enhancing communication and coordination within the academic staff.

Departments are managed through the "Departments" table, which contains **DepartmentID, DepartmentName,** and **BlockName** fields. DepartmentID uniquely identifies each department, while DepartmentName provides a descriptive label for easy identification and categorization of departments within the institution. block\_name denotes the name of the block where classes for the department are conducted.

The "Classroom" table tracks information about classrooms, including **CourseID** and **Room\_Number**. CourseID serves as a unique identifier for each course, linking it to the corresponding classroom, while Room\_Number specifies the room number where the course is conducted.

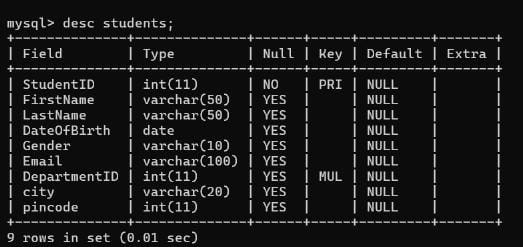
**ER DIAGRAM:**

****  
**DESCRIPTION OF ER DIAGRAM:**

* The "Students" table has attributes like **StudentID (Primary Key), FirstName, LastName, DateOfBirth, Gender, Email, City** and **Pincode.**
* The "Courses" table includes **CourseID (Primary Key), CourseName, DepartmentID (Foreign Key referencing Departments), Credits,** and **InstructorID (Foreign Key referencing Instructors)**.
* The "Instructors" table has **InstructorID (Primary Key), FirstName, LastName, Email,** and **Phone.**
* The "Departments" table contains **DepartmentID(Primary Key), DepartmentName** and **block\_name.**
* The "Classroom" table contains **Course\_ID** and **Room\_Number.**

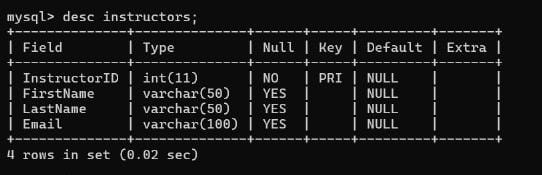
This diagram visually represents the relationships between students, courses, instructors, departments, and the teaching assignments.

**CONVERSION OF ER DIAGRAM INTO TABLES:**



A black and white screen with white text

Description automatically generated



A black and white screen with white text

Description automatically generated



**DESCRIPTION OF TABLES:**

**Students Table**:

- **StudentID**: Unique identifier for each student.

- **FirstName**: First name of the student.

- **LastName**: Last name of the student.

- **DateOfBirth**: Date of birth of the student.

- **Gender**: Gender of the student.

- **Email**: Email address of the student.

- **DepartmentID**: Identifier linking the student to a department.

- **City**: Name of the city which the student belongs to.

- **Pincode**: Pincode of the city which the student belongs to.

**Courses Table**:

- **CourseID**: Unique identifier for each course.

- **CourseName**: Name of the course.

- **DepartmentID**: Identifier linking the course to a department.

- **Credits**: Number of credits associated with the course.

- **InstructorID**: Identifier linking the course to an instructor.

**Instructors Table**:

- **InstructorID**: Unique identifier for each instructor.

- **FirstName**: First name of the instructor.

- **LastName**: Last name of the instructor.

- **Email**: Email address of the instructor.

**Departments Table:**

- **DepartmentID**: Unique identifier for each department.

- **DepartmentName**: Name of the department.

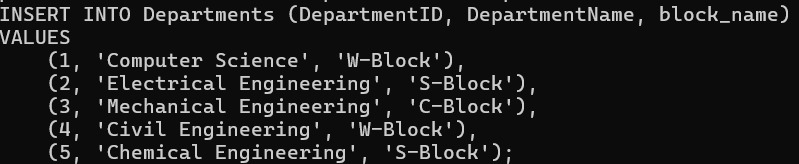
- **block\_name**: Name of the block where classes for the department are conducted.

**Classroom Table:**

**-** **CourseID**: Unique identifier for each course.

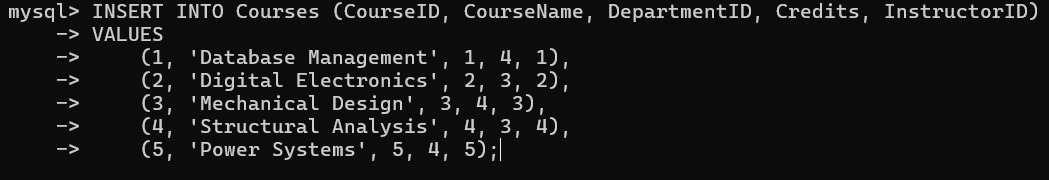
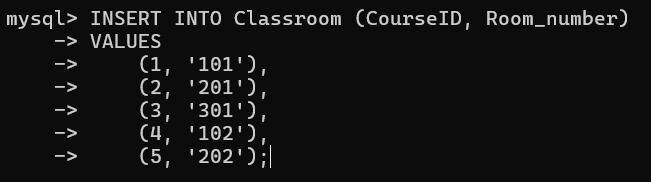
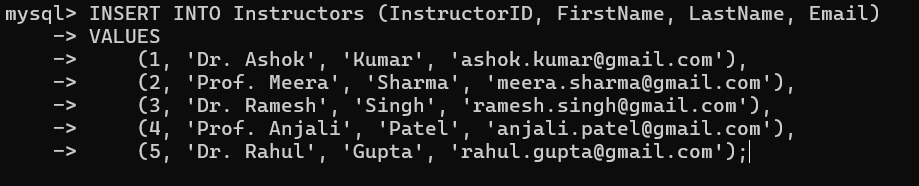
**- Room\_Number**: Room number of the classroom.

**CREATION OF DATA IN THE TABLES:**



A screen shot of a computer

Description automatically generated



# SQL QUERIES:

Query 1: Subquery to Retrieve Student Details with Department Names  
  
SELECT s.FirstName, s.LastName, d.DepartmentName

FROM Students s

JOIN Departments d ON s.DepartmentID = d.DepartmentID;

A computer screen shot of a computer program

Description automatically generated

Query 2: Aggregate Function to Find Average Credits per Department:  
  
SELECT d.DepartmentName, AVG(c.Credits) AS AverageCredits

FROM Courses c

JOIN Departments d ON c.DepartmentID = d.DepartmentID

GROUP BY d.DepartmentName;

A computer screen with white text

Description automatically generated

Query 3: Aggregate Function to Count Courses in Each Department:  
  
SELECT d.DepartmentName, COUNT(c.CourseID) AS CourseCount

FROM Departments d

LEFT JOIN Courses c ON d.DepartmentID = c.DepartmentID

GROUP BY d.DepartmentName;

A black screen with white text

Description automatically generated

Query 4: Subquery to Find Courses Taught by a Specific Instructor:  
  
SELECT CourseName

FROM Courses

WHERE InstructorID = (

SELECT InstructorID FROM Instructors WHERE FirstName = 'Dr. Ashok' AND LastName = 'Kumar'

);

A black screen with white text

Description automatically generated

Query 5: Join with Subquery to Retrieve Classroom Details with Course Names:  
  
SELECT cl.Room\_number, cr.CourseName

FROM Classroom cl

JOIN (

SELECT CourseID, CourseName

FROM Courses

) cr ON cl.CourseID = cr.CourseID;

A computer screen with white text

Description automatically generated

**NORMALIZATION OF TABLES UPTO 3NF:**

**NORMALIZATION:**

Normalization is the process of organizing data in a database efficiently by reducing redundancy and dependency. It involves breaking down large tables into smaller, more manageable ones and establishing relationships between them to minimize data duplication and ensure data integrity.

**TYPES :**

1. First Normal Form (1NF):-Ensure each cell in a table holds single, atomic value.

2. Second Normal Form (2NF):-Remove partial dependencies by ensuring non-key attributes are fully dependent on the entire primary key.

3. Third Normal Form (3NF):Eliminate transitive dependencies by ensuring non-key attributes are not dependent on other attributes.

**NORMALIZATION OF TABLES:**

1. **Students Table:**

* **Primary Key**: StudentID is the primary key.
* **StudentID (PK)**: No transitive dependencies are present because each attribute (FirstName, LastName, DateOfBirth, Gender, Email, DepartmentID, city, pincode) is directly dependent on the primary key (StudentID).
* Hence it is in 3NF.

**2. Courses Table:**

* **Primary Key**: CourseID is the primary key.
* **CourseID (PK)**: CourseName is directly associated with the CourseID and not indirectly through the DepartmentID. So, there's no actual transitive dependency.
* Hence it is in 3NF.

**3. Instructors Table:**

* **Primary Key**: instructorID is the primary key.
* **InstructorID (PK)**: No transitive dependencies are present because each attribute (FirstName, LastName, Email) is directly dependent on the primary key (InstructorID).
* Hence it is in 3NF.

**4. Departments Table:**

* **Primary Key**: DepartmentID is the primary key
* **DepartmentID (PK)**: No transitive dependencies are present because each attribute (DepartmentName, block\_name) is directly dependent on the primary key (DepartmentID).
* Hence it is in 3NF.

**5. Classroom Table:**

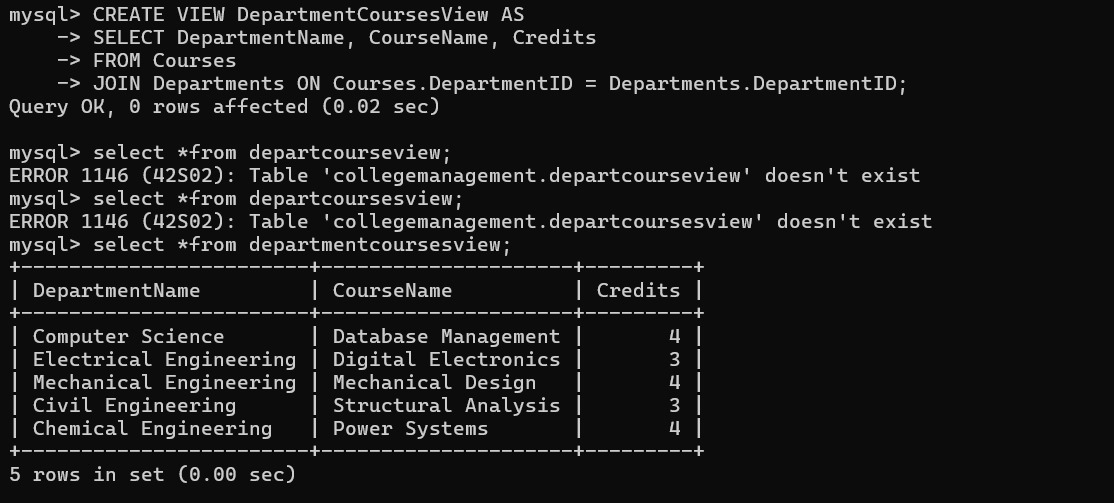
* **Primary Key**: CourseID is the primary key.
* **CourseID (PK)**: No transitive dependencies are present because the Room\_number attribute is directly dependent on the CourseID.
* Hence it is in 3NF.

**CREATION OF VIEWS USING THE TABLES:**

A computer screen with white text

Description automatically generated





A black screen with white text

Description automatically generated