# STUDENT DATA BASE MANAGEMENT SYSTEM

## Introduction

The Student Database Management System is designed to efficiently manage student information, grades, courses, and enrollment data. This system is built using RDBMS software.

## System Setup

1. Install RDBMS software.

   Example: My SQL Workbench

2. Create a new database to hold the student information.

## Database Schema

The database schema consists of the following tables:

- **Students:**

```
create table students (
student_id int primary key,
name varchar(50),
date_of_birth date,
address varchar(100),
contact_number varchar(15)
);
```

- **Courses:**

```
create table courses (
course_id int primary key,
name varchar(50),
description varchar(200),
```

credits int

);

- **Grades:**

create table grades (

student_id int,

course_id int,

grade float,

primary key (student_id, course_id),

foreign key (student_id) references students(student_id),

foreign key (course_id) references courses(course_id)

);

- **Enrollments:**

create table enrollments (

student_id int,

course_id int,

enrollment_date date,

primary key (student_id, course_id),

foreign key (student_id) references students(student_id),

foreign key (course_id) references courses(course_id)

);

## Requirements:

Inserting sample data into tables:

### Student Table:

insert into students (student_id, name, date_of_birth, address, contact_number)

values (1, 'john doe', '1995-07-15', '123 main street', '555-1234');

**Note:** Repeat the same for N number of students

### Courses Table:

insert into courses (course_id, name, description, credits)

values (101, 'mathematics', 'introduction to calculus', 3);

**Note:** Repeat the same for N number of courses

### Grade Table:

insert into grades (student_id, course_id, grade)

values (1, 101, 85.5);

### Enrollments Table:

insert into enrollments (student_id, course_id, enrollment_date)

values (1, 101, '2023-01-01');

Implementing CRUD Operations:

### Retrieve student information:

SELECT * FROM students WHERE student_id = 1;

### Retrieve course information:

SELECT * FROM courses WHERE course_id = 101;

**Update a student's grade for a course:**

```
UPDATE grades SET grade = 4.0 WHERE student_id = 1 AND course_id = 101;
```

**Retrieve enrolled courses for a student:**

```
SELECT courses.* FROM courses
JOIN enrollments ON courses.course_id = enrollments.course_id
WHERE enrollments.student_id = 1;
```

**Delete a student record :**

```
DELETE FROM students WHERE student_id = 1;
```

**Calculate average grade for a student:**

```
SELECT AVG(grade) FROM grades WHERE student_id = 1;
```

**Display top-performing students:**

```
SELECT students.*, AVG(grades.grade) AS average_grade
FROM students JOIN grades ON students.student_id = grades.student_id
GROUP BY students.student_id
ORDER BY average_grade DESC LIMIT 5;
```

**Conclusion:**

The Student Database Management System provides an efficient solution for managing student information, grades, courses, and enrollments. By leveraging a relational database management system (RDBMS) and SQL queries, this system allows users to easily add, retrieve, update, and delete student records, course details, and grades.

The database schema, consisting of tables such as "students," "courses," "grades," and "enrollments," establishes relationships between entities and ensures data integrity through primary and foreign key constraints. Additional functionalities, such as reporting

and searching, enhance the system's capabilities, allowing users to generate reports based on student data, calculate average grades, and filter records based on specific criteria.

The documentation provides comprehensive guidance on setting up and using the system, including installation instructions, SQL query examples, and testing recommendations.

Overall, the Student Database Management System offers a user-friendly and effective way to manage student data, providing educational institutions with a powerful tool for organizing, analyzing, and maintaining student records. By following the guidelines outlined in this documentation, users can successfully utilize and maintain the system to meet their specific needs.