Case Study: Student Management System

Background:

A small educational institution wants to maintain student data in a MySQL database. The system should store students' personal information, their enrolled courses, and their performance.

You are tasked to create and manage this database using basic **CRUD operations**.

Database Name: `student_management`

Tables to Create:

1. `students`

```
CREATE TABLE students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    dob DATE,
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15),
    address TEXT
);
```

2. `courses`

```
CREATE TABLE courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(100),
    course_description TEXT,
    credits INT
);
```

3. `enrollments`

```
CREATE TABLE enrollments (
    enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    grade VARCHAR(2),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

□ CRUD Operations to Practice:

1. CREATE (INSERT)

- Add 5 sample students
- Add 3 sample courses
- Enroll students in different courses

```
-- Add a student
INSERT INTO students (first_name, last_name, dob, email, phone, address)
VALUES ('John', 'Doe', '2002-05-14', 'john.doe@example.com', '9876543210',
'123 Main St');

-- Add a course
INSERT INTO courses (course_name, course_description, credits)
VALUES ('Web Development', 'Learn HTML, CSS, JavaScript', 4);

-- Enroll a student
INSERT INTO enrollments (student_id, course_id, enrollment_date, grade)
VALUES (1, 1, CURDATE(), NULL);
```

---

2. READ (SELECT)

- List all students
- Get details of a specific student
- Show all courses with their enrolled students
- Show a student’s grades and enrolled courses

```
-- List all students
SELECT * FROM students;

-- Get student details by email
SELECT * FROM students WHERE email = 'john.doe@example.com';

-- Courses with enrolled students
SELECT s.first_name, s.last_name, c.course_name, e.enrollment_date
FROM enrollments e
JOIN students s ON e.student_id = s.student_id
JOIN courses c ON e.course_id = c.course_id;

-- Grades for a student
SELECT s.first_name, c.course_name, e.grade
FROM enrollments e
JOIN students s ON e.student_id = s.student_id
JOIN courses c ON e.course_id = c.course_id
WHERE s.student_id = 1;
```

---

3. UPDATE

- Update a student’s phone number
- Change a student’s grade
- Edit a course description

```
-- Update phone number
UPDATE students SET phone = '9999999999' WHERE student_id = 1;

-- Update grade
UPDATE enrollments SET grade = 'A' WHERE enrollment_id = 1;

-- Update course description
UPDATE courses SET course_description = 'Complete Web Development course
including React and Node.js'
WHERE course_id = 1;
```

---

4. DELETE

- Delete a student and their enrollments
- Remove a course
- Delete specific enrollment

```
-- Delete enrollment
DELETE FROM enrollments WHERE enrollment_id = 1;

-- Delete student (make sure to delete enrollments first due to FK
constraint)
DELETE FROM enrollments WHERE student_id = 1;
DELETE FROM students WHERE student_id = 1;

-- Delete a course
DELETE FROM courses WHERE course_id = 1;
```

Practice Tasks

1. Add 3 new students with different details.
2. Create 2 new courses.
3. Enroll each student in at least one course.
4. Update one student’s name and email.
5. Change grades for 2 enrollments.
6. Write a query to list all students enrolled in “Web Development”.
7. Delete one course and ensure it doesn’t break other relationships.