**Insert Statements:**

* Insert into Student Table:

INSERT INTO Student (StudentID, Name, Email, Address) VALUES (1, 'Jane Smith', 'jane@example.com', '456 Pine St');

* Insert into Course Table:

INSERT INTO Course (CourseID, Title, Description) VALUES (201, 'Physics', 'Fundamentals of Physics');

**Update Statements:**

* Update Student's Email:

UPDATE Student SET Email = 'updated\_email@example.com' WHERE StudentID = 1;

* Update Course Description:

UPDATE Course SET Description = 'Advanced Physics Concepts' WHERE CourseID = 201;

**Delete Statement:**

* Delete Enrollment Record:

DELETE FROM Enrollment WHERE EnrollmentID = 3;

**Select Statement:**

* Select Subset of Rows and Columns from Student Table:

SELECT StudentID, Name, Email FROM Student WHERE StudentID > 1;

**Select Statements with Joins:**

* Select Data from Joining Student and Enrollment Tables:

SELECT s.Name, c.Title FROM Student s INNER JOIN Enrollment e ON s.StudentID = e.StudentID INNER JOIN Course c ON e.CourseID = c.CourseID;

* Select Data from Joining Course and Instructor Tables:

SELECT c.Title, i.Name AS Instructor FROM Course c INNER JOIN Instructor i ON c.CourseID = i.CourseID;

**Summary Functions:**

* Summary Function - Count of Students:

SELECT COUNT(\*) AS TotalStudents FROM Student;

* Summary Function - Average Enrollment per Course:

SELECT AVG(COUNT(e.EnrollmentID)) AS AvgEnrollment FROM Course c LEFT JOIN Enrollment e ON c.CourseID = e.CourseID GROUP BY c.CourseID;

* Summary Function - Maximum Enrollment Date per Course:

SELECT CourseID, MAX(EnrollmentDate) AS MaxEnrollmentDate FROM Enrollment GROUP BY CourseID;

**Multi-Table Query:**

* Multi-Table Query - Information from Multiple Tables:

SELECT s.Name AS StudentName, c.Title AS CourseTitle, i.Name AS InstructorName FROM Student s INNER JOIN Enrollment e ON s.StudentID = e.StudentID INNER JOIN Course c ON e.CourseID = c.CourseID INNER JOIN Instructor

MAIN:

1: Add a new ENUM column with a constraint:

-- Add a new ENUM column 'status' with choices and a NOT NULL constraint ALTER TABLE students ADD COLUMN status ENUM('Active', 'Inactive', 'Suspended') NOT NULL DEFAULT 'Active';

SQL:

-- Add a new ENUM column 'status' with choices and a NOT NULL constraint to the 'students' table

ALTER TABLE students

ADD COLUMN status ENUM('Active', 'Inactive', 'Suspended') NOT NULL DEFAULT 'Active';

-- For Instructor Table

ALTER TABLE instructor

ADD COLUMN employment\_status ENUM('Employed', 'Unemployed', 'Retired') NOT NULL DEFAULT 'Employed';

2: Insert a new record:

a) Attempt to insert without providing ENUM value (to show error):

-- Attempt to insert a record without providing ENUM value (will throw an error) INSERT INTO students (column1, column2, ..., status) VALUES (value1, value2, ..., ''); -- Providing an empty string or NULL to status column

SQL:

INSERT INTO students (StudentID, Name, Email, Address)

VALUES (1, 'John Doe', 'john@example.com', '123 Main St');

-- Attempting to insert a record without providing ENUM value (will throw an error)

INSERT INTO instructor (InstructorID, Name, Email)

VALUES (1, 'John Doe', 'john@example.com');

b) Modify insert with a valid ENUM value:

-- Modify insert with a valid ENUM value INSERT INTO students (column1, column2, ..., status) VALUES (value1, value2, ..., 'Active'); -- Replace 'Active' with the desired ENUM value

SQL:

-- Inserting into Students table with a valid ENUM value

INSERT INTO students (column1, column2, enrollment\_status)

VALUES ('John', 'Doe', 'Active'); -- Replace 'John', 'Doe' with actual values and 'Active' with a valid ENUM value

-- Inserting into Instructor table with a valid ENUM value

INSERT INTO instructor (column1, column2, employment\_status)

VALUES ('Jane', 'Smith', 'Employed'); -- Replace 'Jane', 'Smith' with actual values and 'Employed' with a valid ENUM value

3: Explain changes and modify data for the inserted record:

Inserted Record in Students Table:

Assuming a record was inserted into the students table as follows:

SQL: INSERT INTO students (column1, column2, enrollment\_status) VALUES ('John', 'Doe', 'Active');

Here, 'John Doe' was inserted with an enrollment\_status of 'Active'.

Inserted Record in Instructor Table:

Assuming a record was inserted into the instructor table as follows:

SQL:

INSERT INTO instructor (column1, column2, employment\_status) VALUES ('Jane', 'Smith', 'Employed');

Here, 'Jane Smith' was inserted with an employment\_status of 'Employed'.

Changes to the Inserted Records:

Now, let's say you want to modify the data for these inserted records:

Changes for Students Table:

-- Explanation: Changing 'Active' status to 'Suspended' for the student record UPDATE students SET enrollment\_status = 'Suspended' WHERE column1 = 'John' AND column2 = 'Doe'; -- Replace with appropriate conditions to identify the record

Changes for Instructor Table:

-- Explanation: Changing 'Employed' status to 'Retired' for the instructor record UPDATE instructor SET employment\_status = 'Retired' WHERE column1 = 'Jane' AND column2 = 'Smith'; -- Replace with appropriate conditions to identify the record

Modifications and Explanation:

Adjust the UPDATE queries by specifying the conditions to identify the specific record you want to modify. In these examples, the queries are structured to change the enrollment\_status for 'John Doe' in the students table from 'Active' to 'Suspended', and the employment\_status for 'Jane Smith' in the instructor table from 'Employed' to 'Retired'.

Execute these UPDATE SQL queries in your MariaDB environment, replacing the column names and conditions with the appropriate values to reflect the changes you intend to make to the inserted records. Adjust the conditions to uniquely identify the records you want to modify.

4: Delete the new record using a WHERE clause:

-- Deleting the specific new record from the Students table using a WHERE clause

DELETE FROM students

WHERE column1 = 'John' AND column2 = 'Doe' AND enrollment\_status = 'Suspended';

-- Deleting the specific new record from the Instructor table using a WHERE clause

DELETE FROM instructor

WHERE column1 = 'Jane' AND column2 = 'Smith' AND employment\_status = 'Retired';

5: Demonstrate AUTO\_INCREMENT constraint:

-- Show the current state of the students table

SELECT \* FROM students;

-- Inserting a new record into the students table with AUTO\_INCREMENT

INSERT INTO students (name, age, enrollment\_status)

VALUES ('Alice', 20, 'Active');

-- Insert values for other columns except student\_id

**Demonstration Summary:**

The AUTO\_INCREMENT constraint allows the database system to automatically generate a unique value for a column (typically used for primary keys) when a new record is inserted without specifying a value for that column.

The column with the AUTO\_INCREMENT constraint will automatically increment by 1 for each new record insertion, ensuring a unique and sequentially increasing identifier for each row.

6: Demonstrate DEFAULT constraint:

-- Example of adding a new record to Instructor table

INSERT INTO instructor (name, department, email)

VALUES ('John Doe', 'Mathematics', 'john@example.com');

-- View the contents of the Instructor table

SELECT \* FROM instructor;