Module 4 – Introduction to DBMS

Introduction to SQL

LAB EXERCISES:

Lab 1: Create a new database named school\_db and a table called students with the following columns: student\_id, student\_name, age, class, and address.

Ans. CREATE DATABASE school\_db;

USE school\_db;

CREATE TABLE students (

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

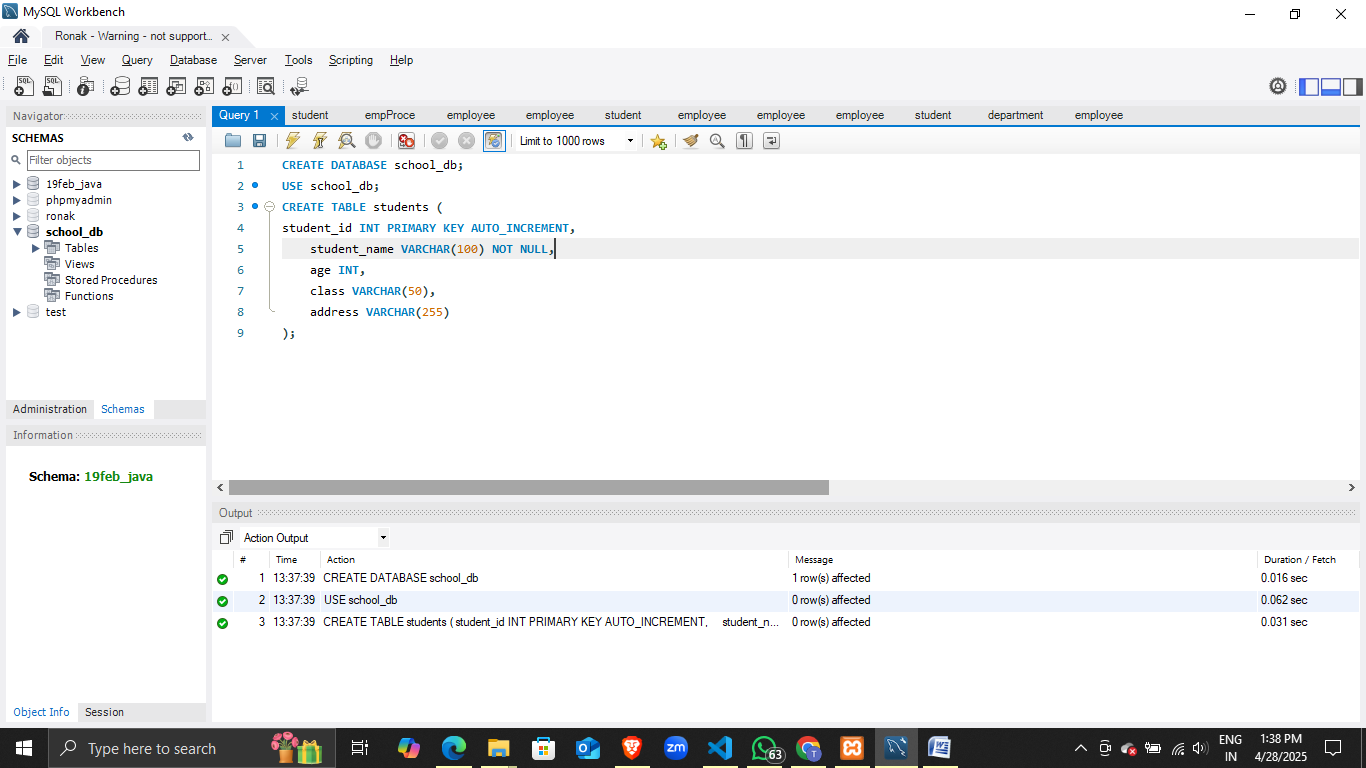
student\_name VARCHAR(100) NOT NULL,

age INT,

class VARCHAR(50),

address VARCHAR(255)

);



Lab 2: Insert five records into the students table and retrieve all records using the SELECT statement.

Ans. INSERT INTO students (student\_name, age, class, address)

VALUES

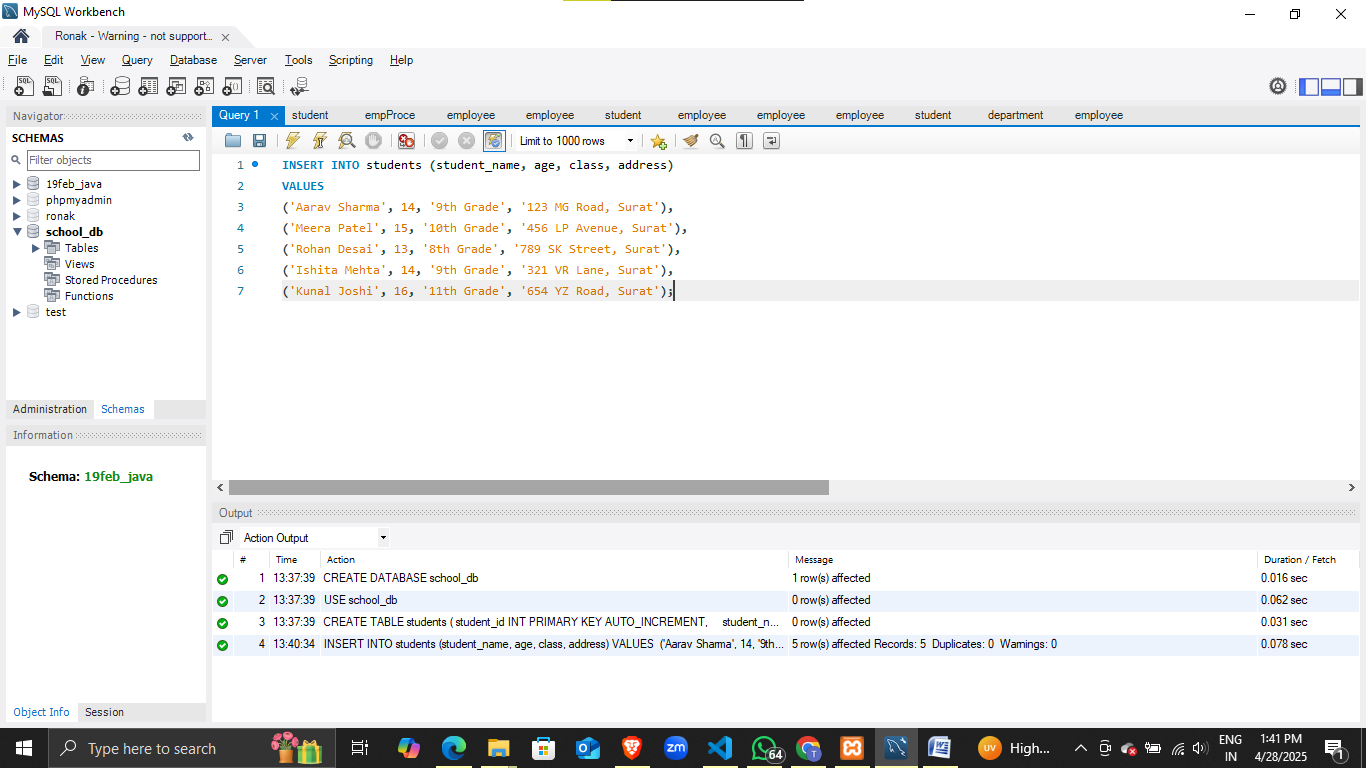
('Aarav Sharma', 14, '9th Grade', '123 MG Road, Surat'),

('Meera Patel', 15, '10th Grade', '456 LP Avenue, Surat'),

('Rohan Desai', 13, '8th Grade', '789 SK Street, Surat'),

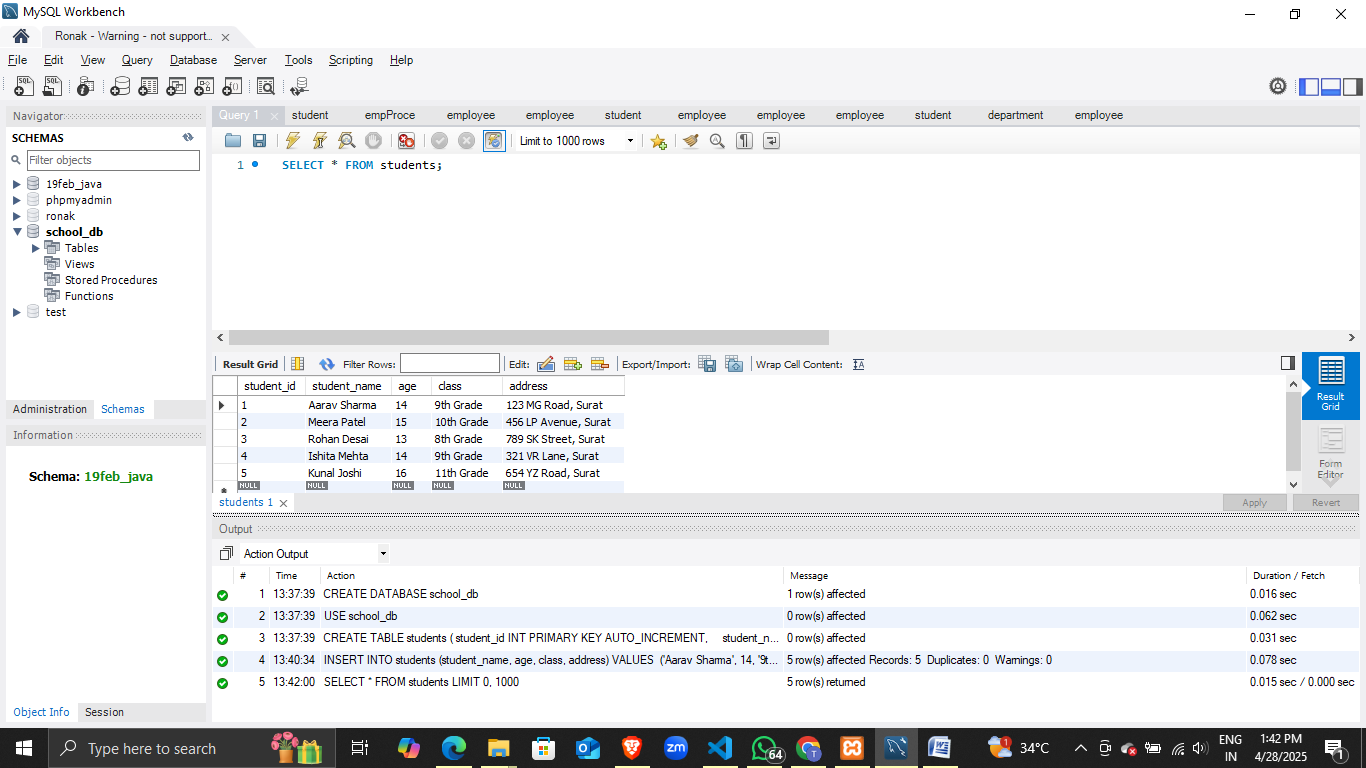
('Ishita Mehta', 14, '9th Grade', '321 VR Lane, Surat'),

('Kunal Joshi', 16, '11th Grade', '654 YZ Road, Surat');

****

To View

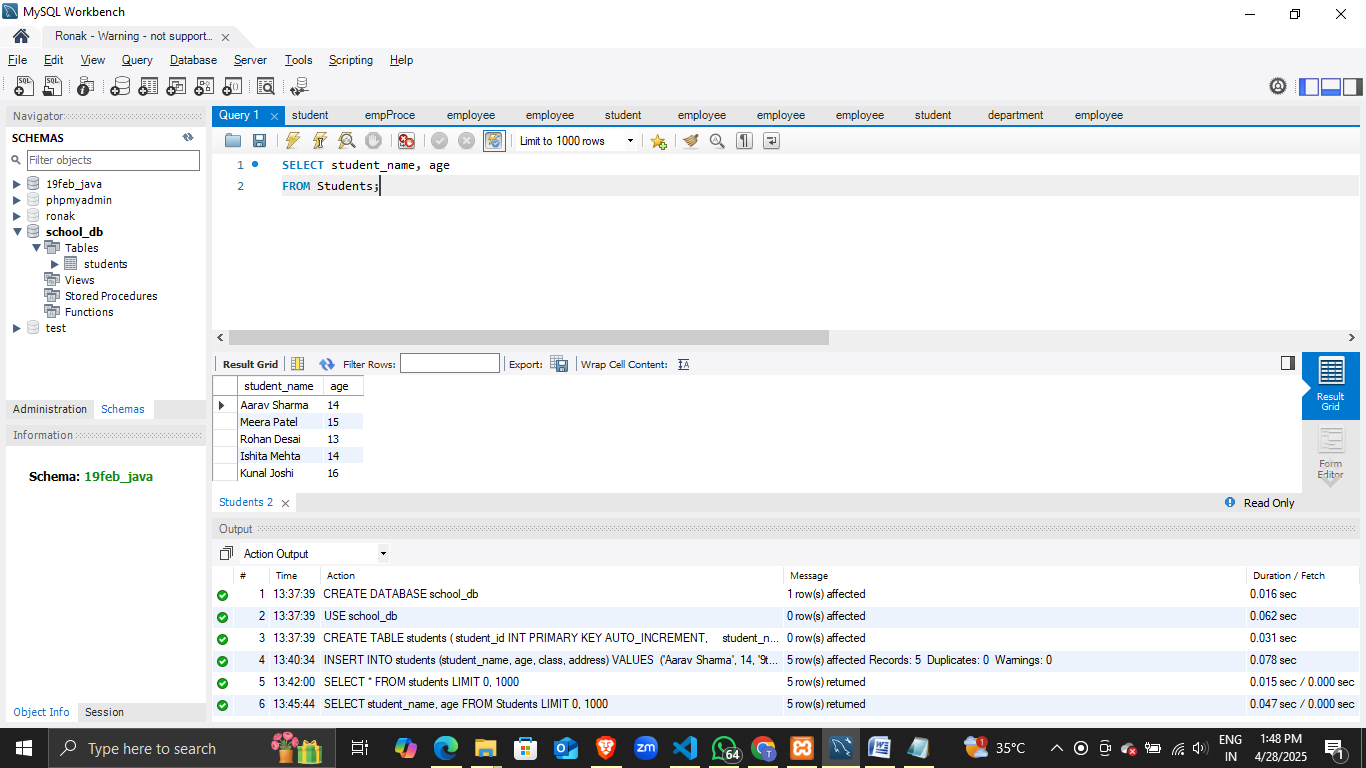
SELECT \* FROM students;



Lab 1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.

Ans. SELECT student\_name, age

FROM students;

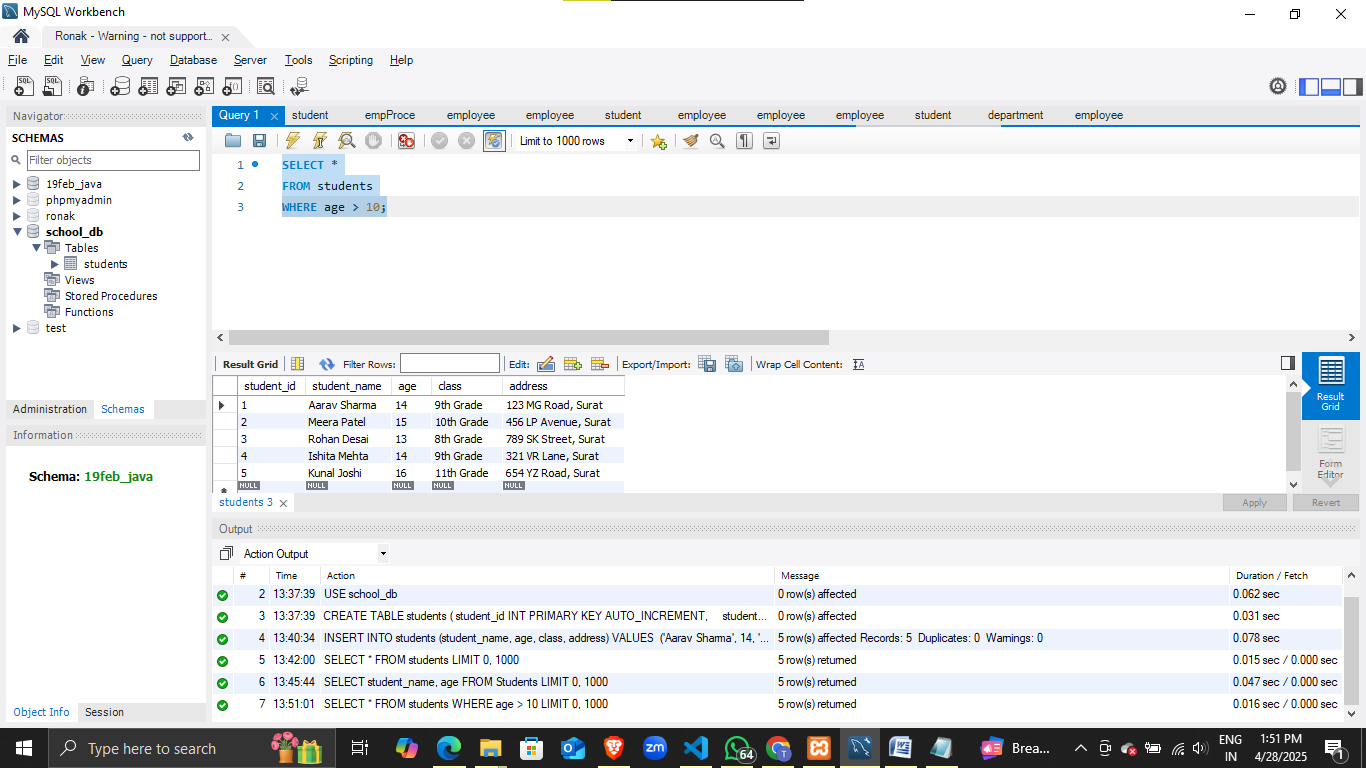


Lab 2: Write SQL queries to retrieve all students whose age is greater than 10?

Ans. SELECT \*

FROM students

WHERE age > 10;



Lab 1: Create a table teachers with the following columns: teacher\_id (Primary Key), teacher\_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

Ans. CREATE TABLE teachers (

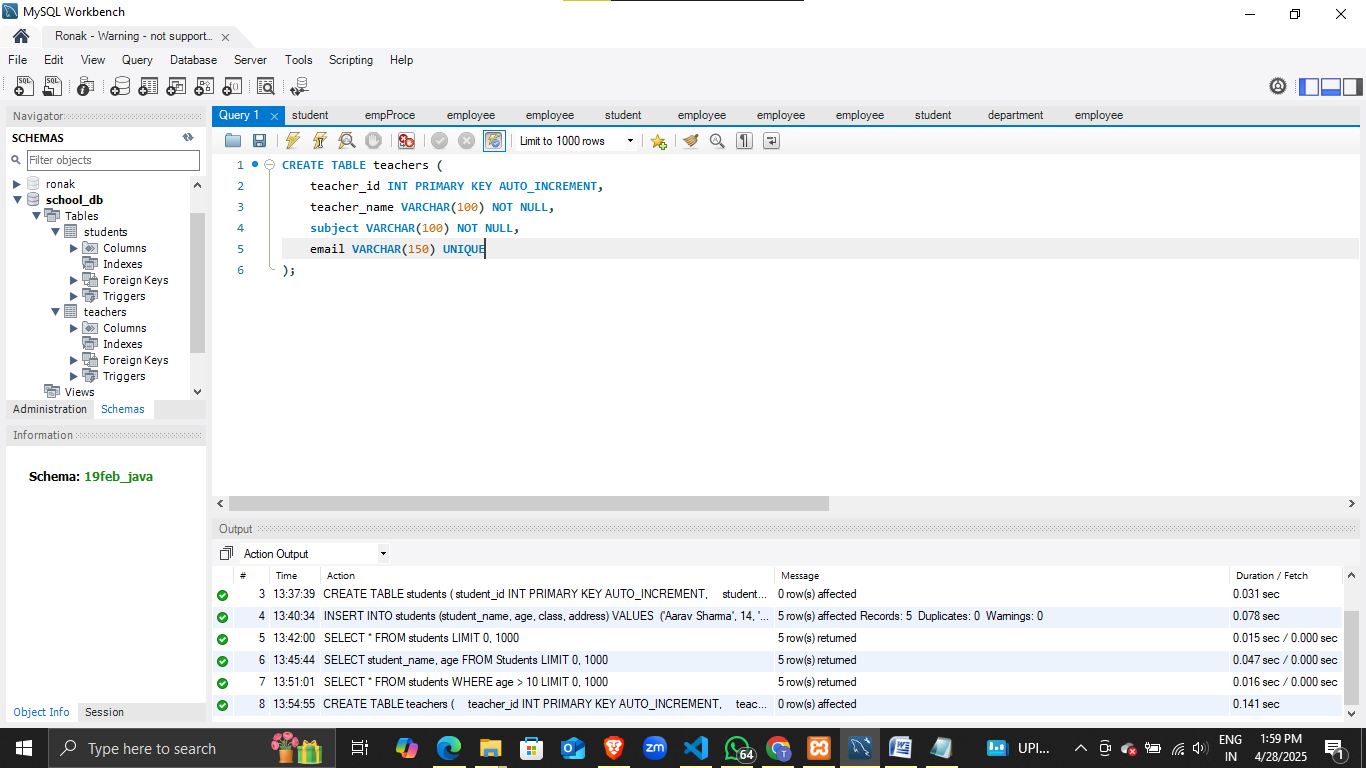
teacher\_id INT PRIMARY KEY AUTO\_INCREMENT,

teacher\_name VARCHAR(100) NOT NULL,

subject VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE

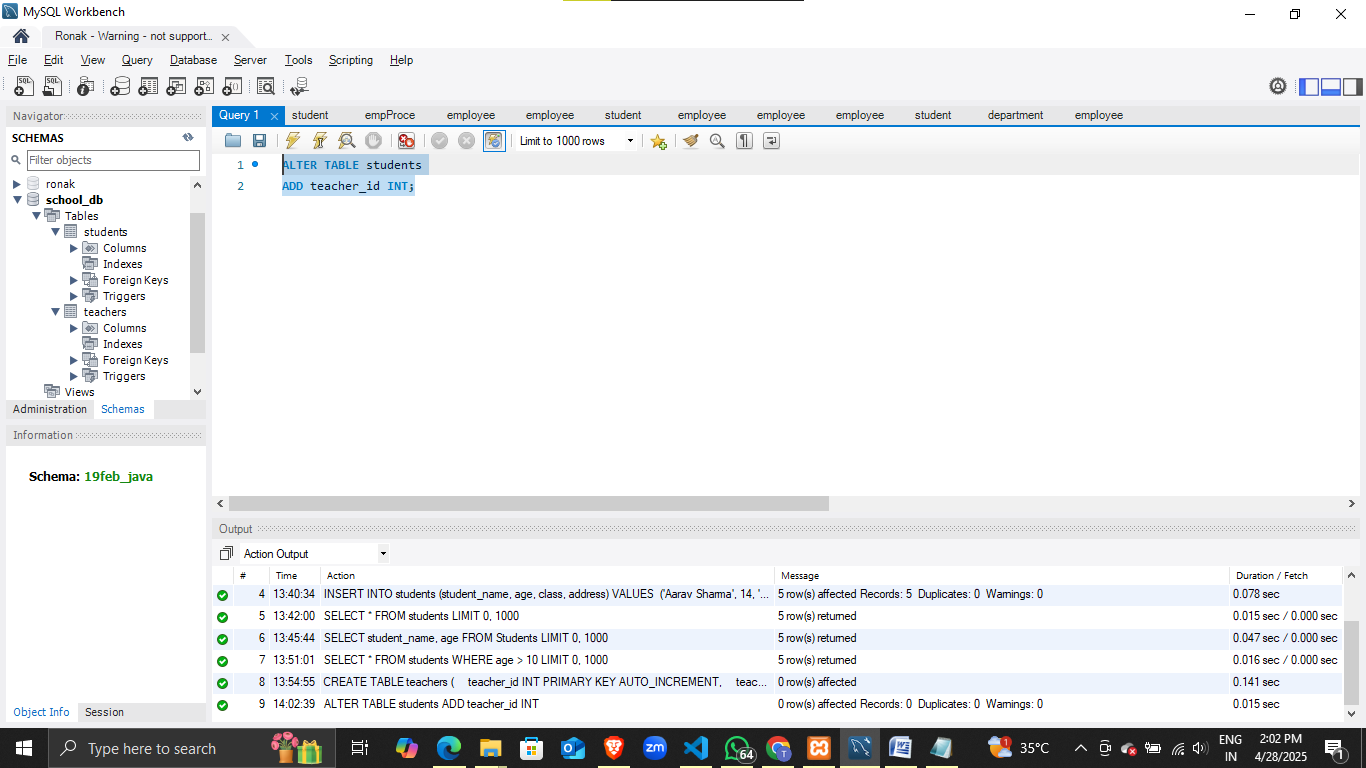
);



Lab 2: Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.

Ans. ALTER TABLE students

ADD teacher\_id INT;

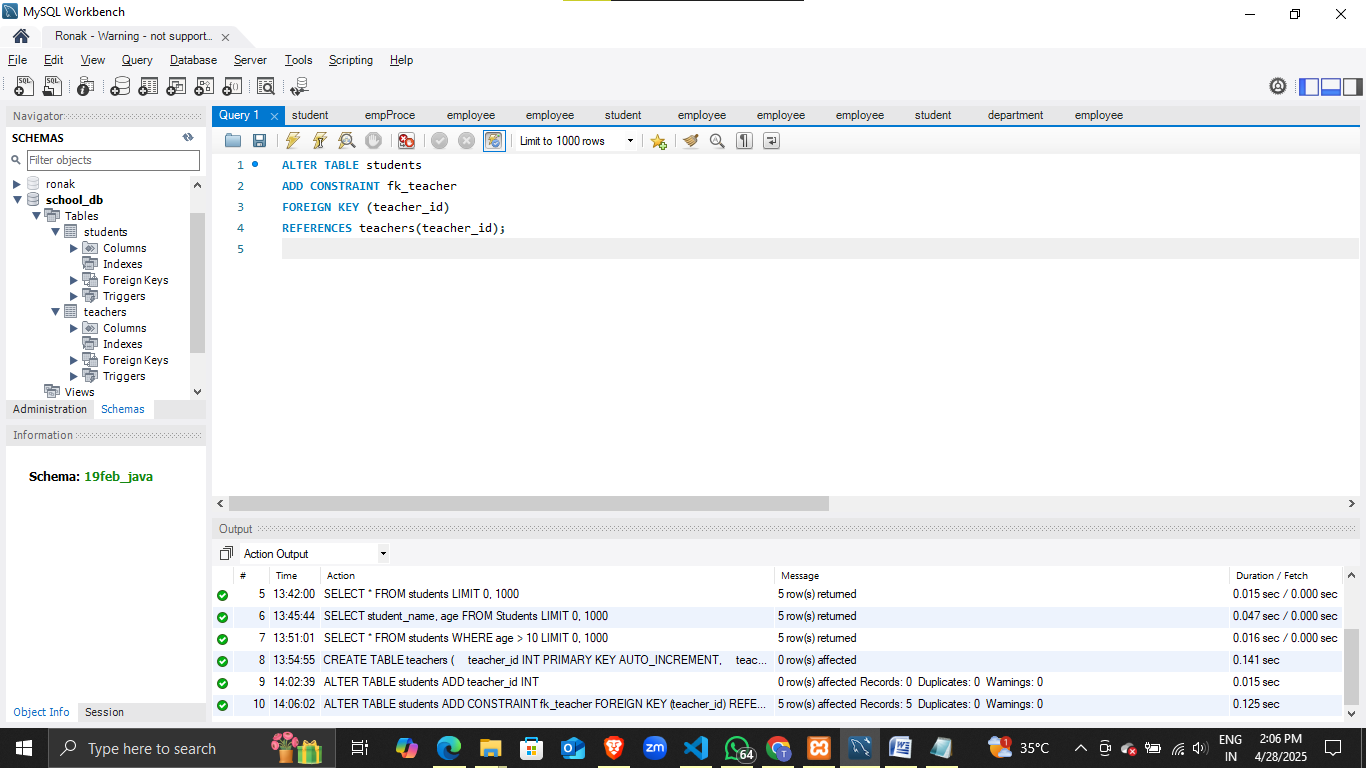


ALTER TABLE students

ADD CONSTRAINT fk\_teacher

FOREIGN KEY (teacher\_id)

REFERENCES teachers(teacher\_id);



Lab 1: Create a table courses with columns: course\_id, course\_name, and course\_credits. Set the course\_id as the primary key.

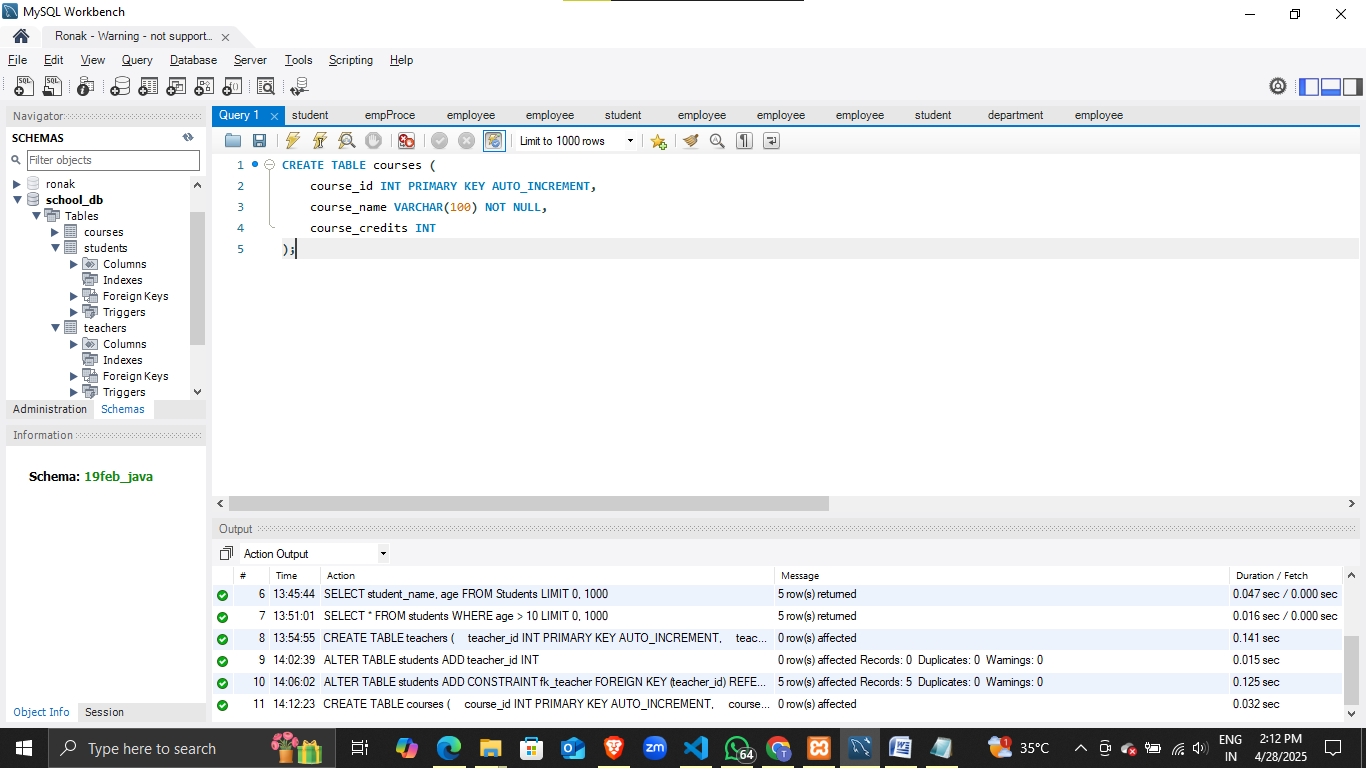
Ans. CREATE TABLE courses (

course\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_name VARCHAR(100) NOT NULL,

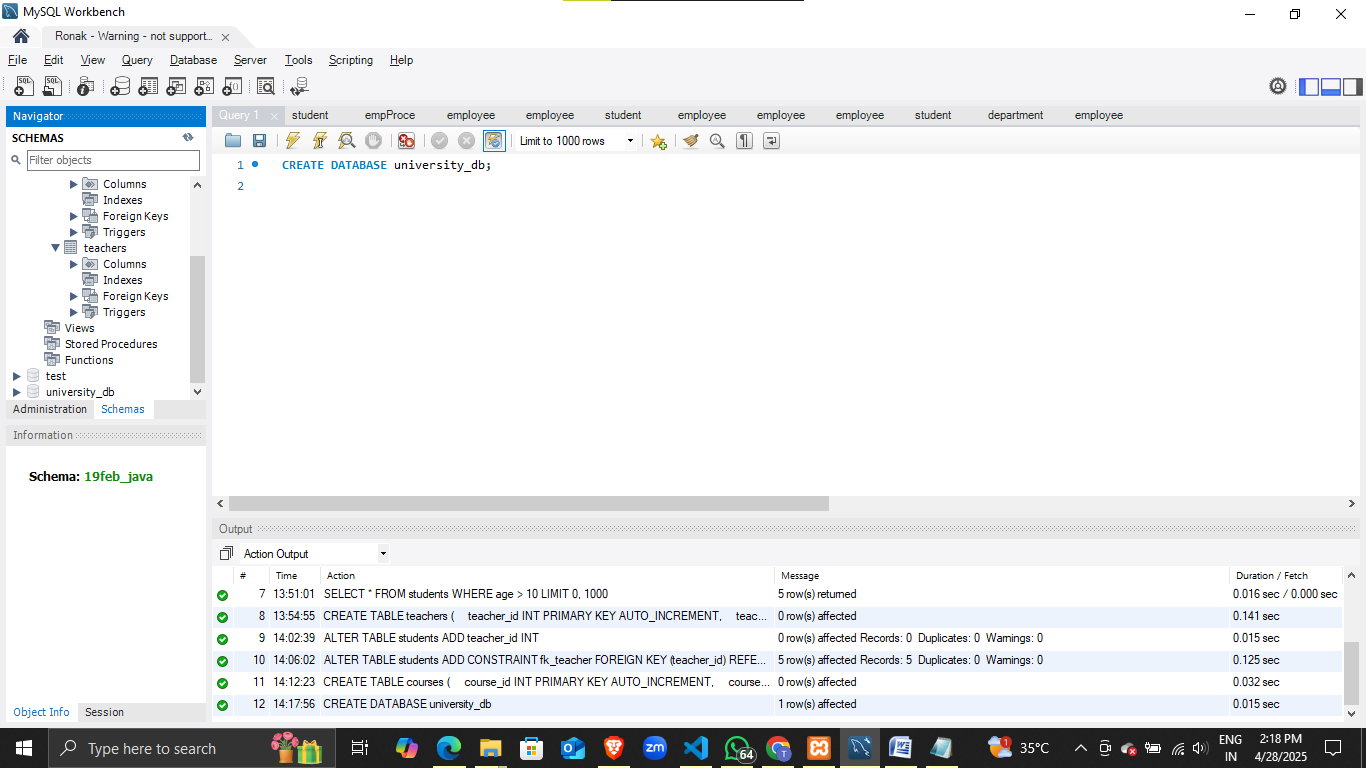
course\_credits INT

);



Lab 2: Use the CREATE command to create a database university\_db.

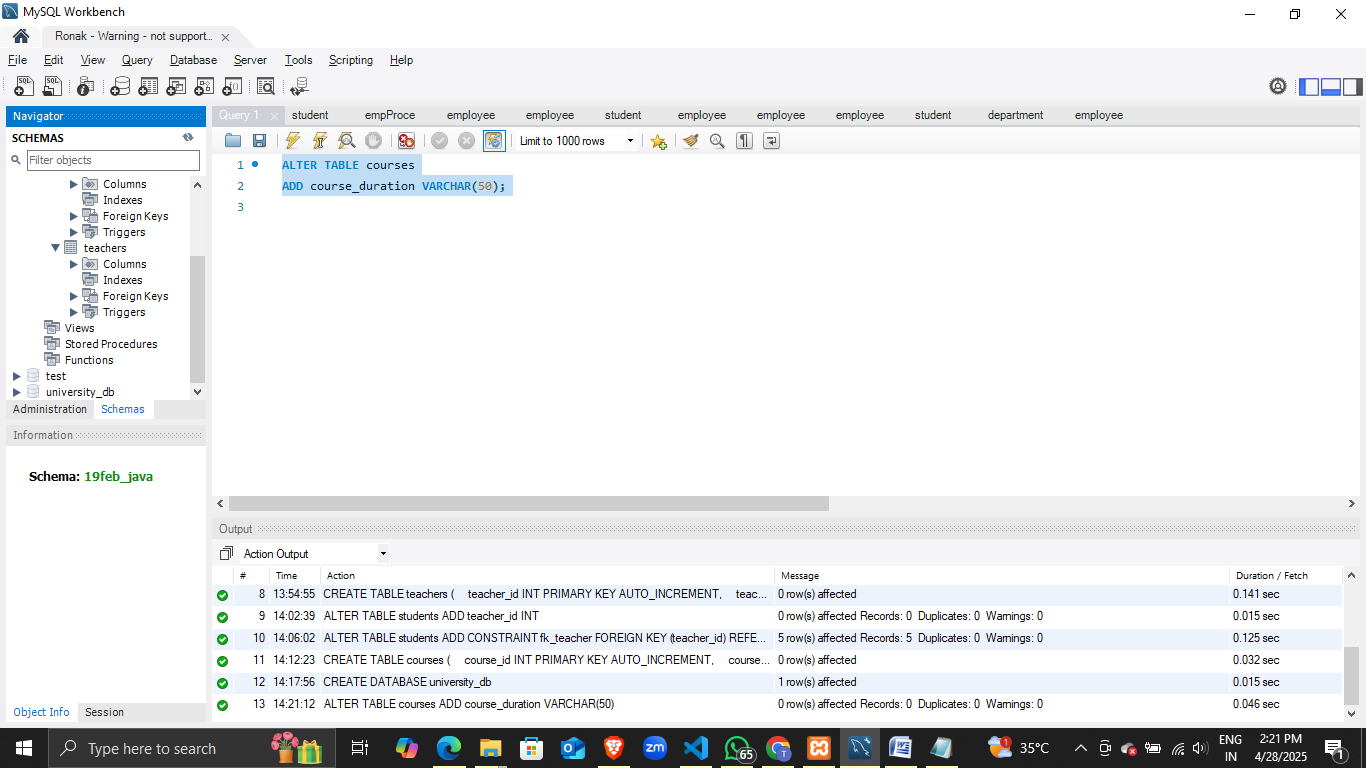
Ans. CREATE DATABASE university\_db;



Lab 1: Modify the courses table by adding a column course\_duration using the ALTER command.

Ans. ALTER TABLE courses

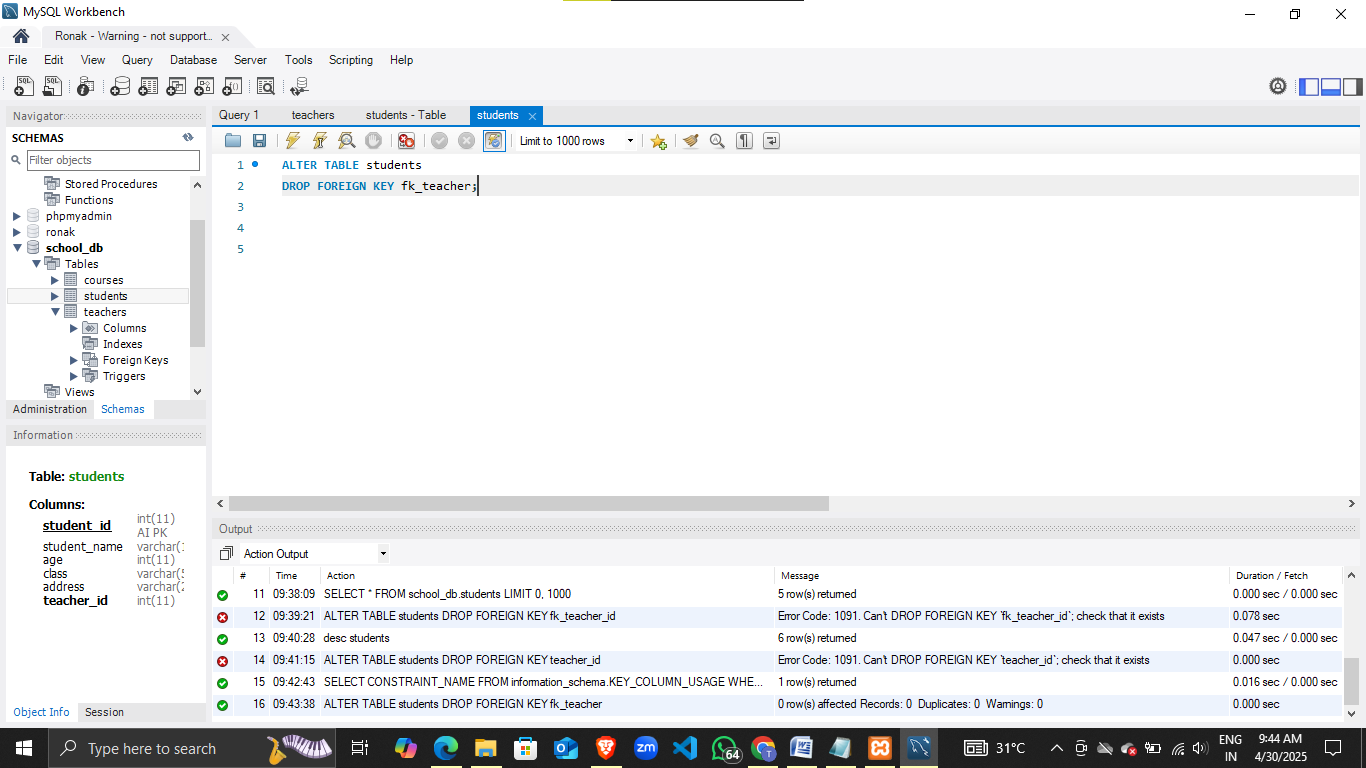
ADD course\_duration VARCHAR(50);



Lab 2: Drop the course\_credits column from the courses table.

Ans. ALTER TABLE courses

DROP COLUMN course\_credits;

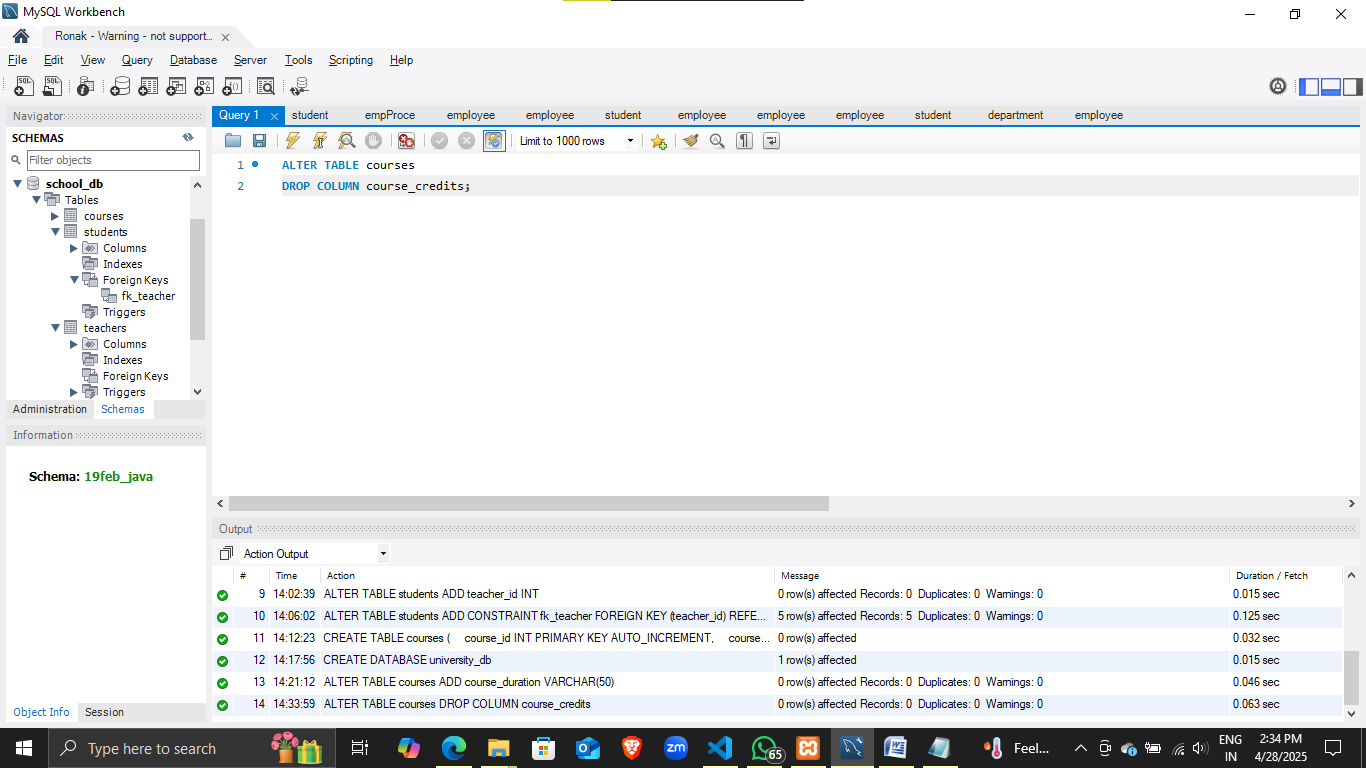


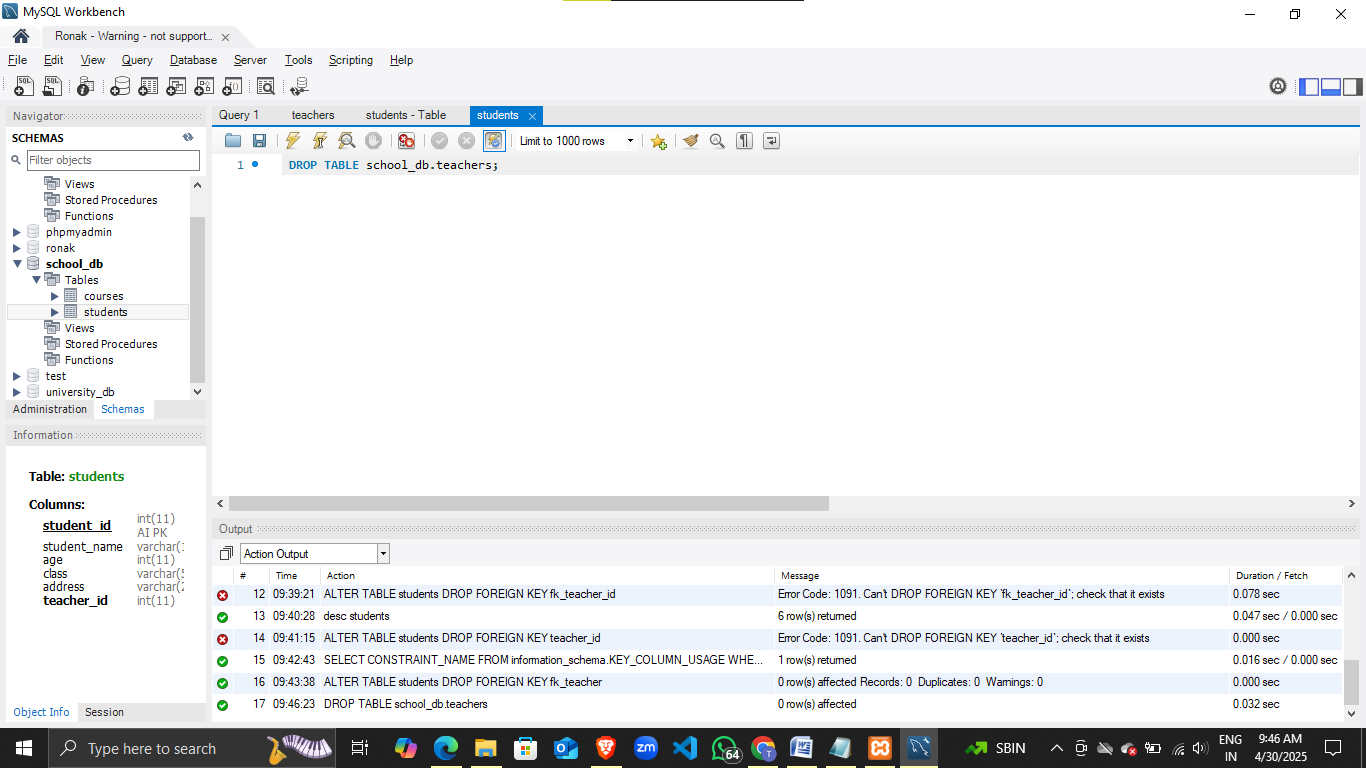
Lab 1: Drop the teachers table from the school\_db database.

Ans. ALTER TABLE students

DROP FOREIGN KEY fk\_teacher;

DROP TABLE school\_db.teachers;



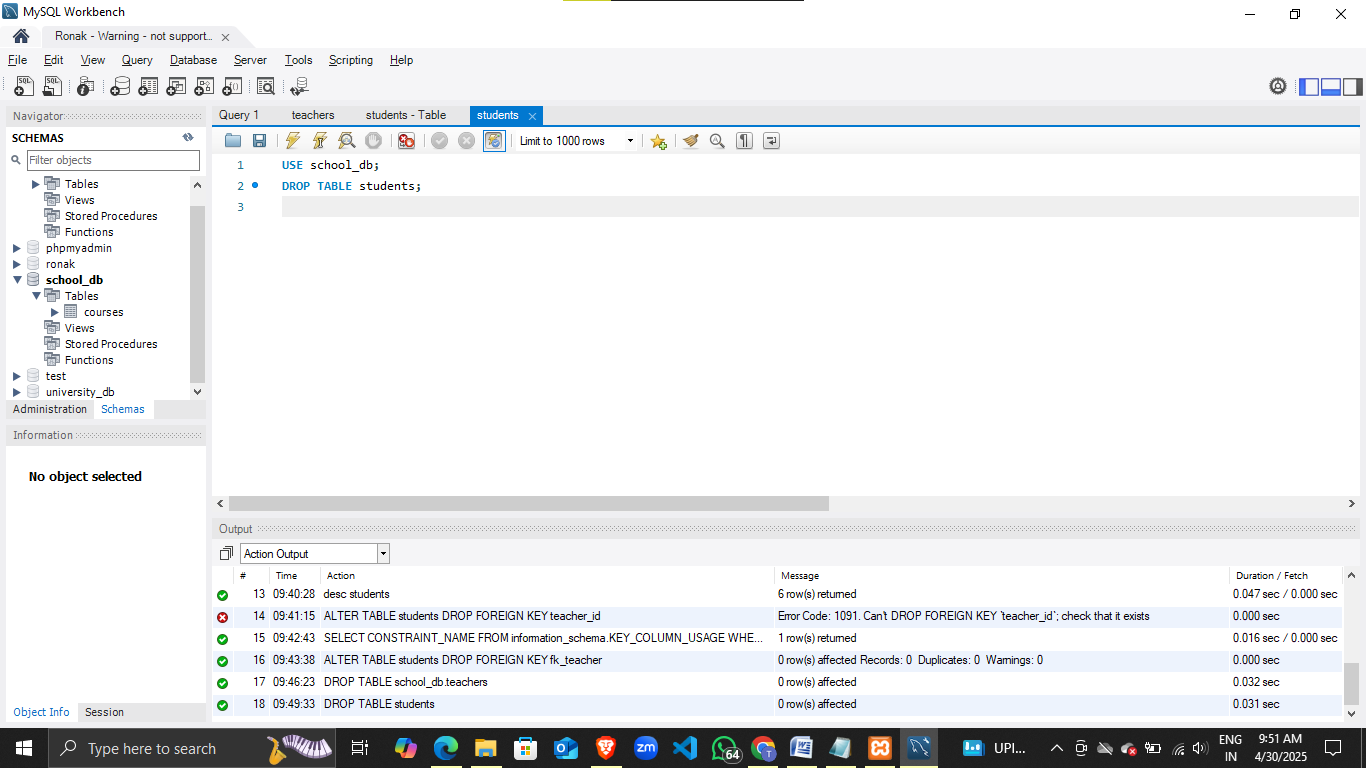


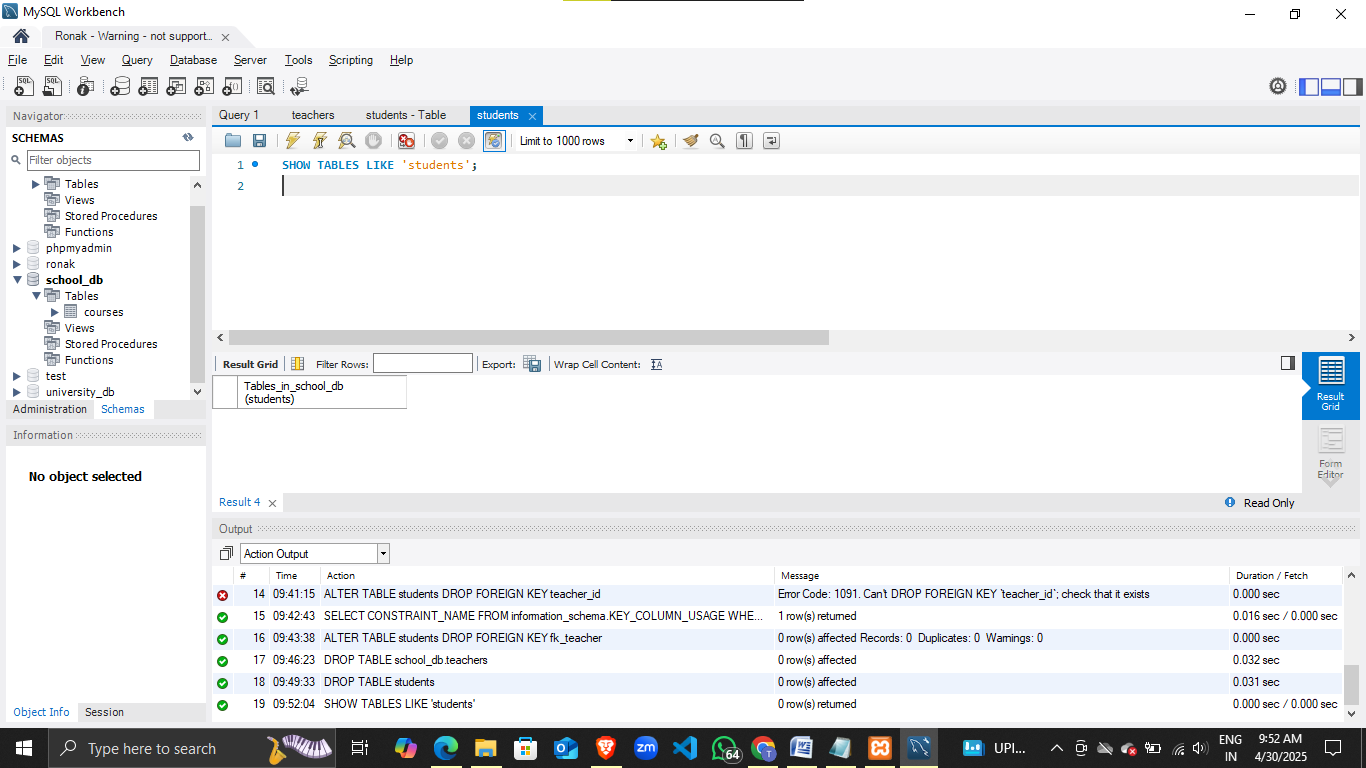
Lab 2: Drop the students table from the school\_db database and verify that the table has been removed.

Ans. USE school\_db;

DROP TABLE students;

SHOW TABLES LIKE 'students';





Lab 1: Insert three records into the courses table using the INSERT command.•

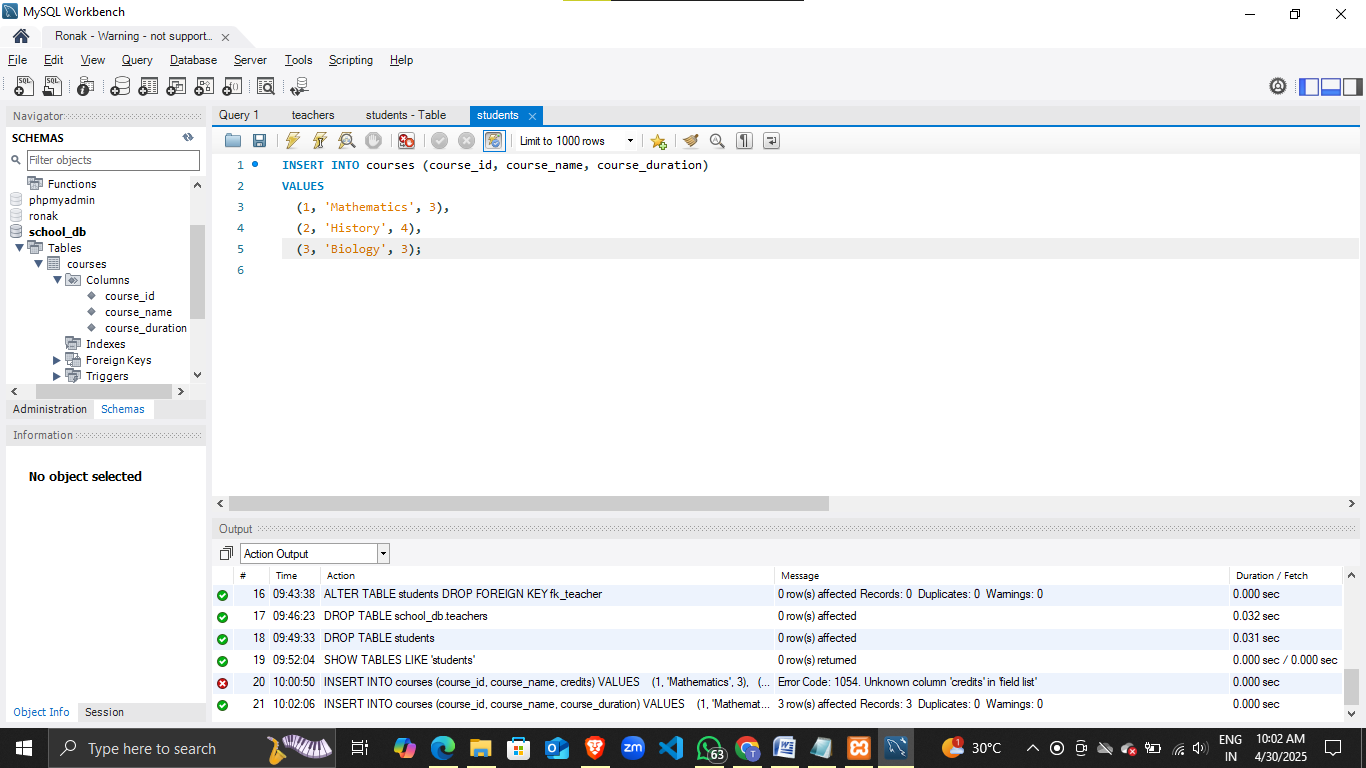
Ans. INSERT INTO courses (course\_id, course\_name, course\_duration)

VALUES

(1, 'Mathematics', 3),

(2, 'History', 4),

(3, 'Biology', 3);

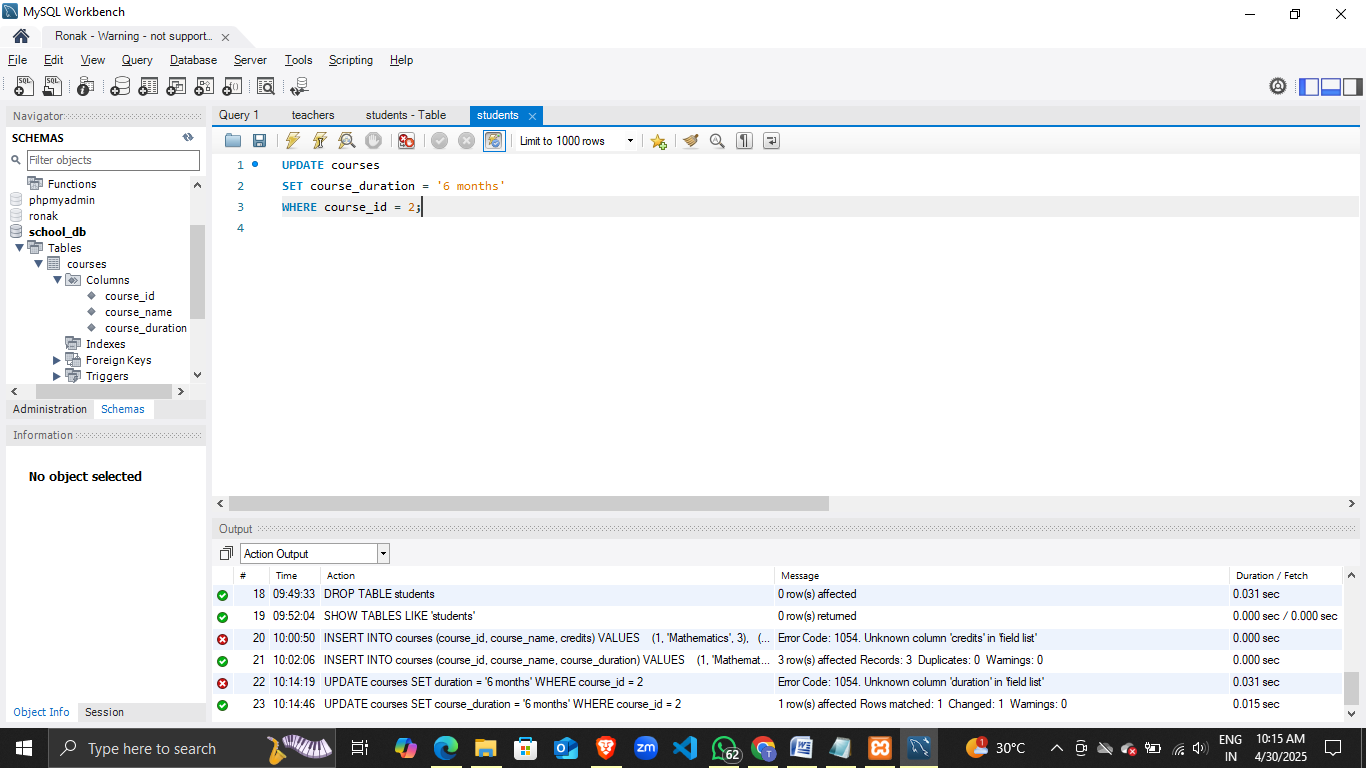


Lab 2: Update the course duration of a specific course using the UPDATE command.

Ans. UPDATE courses

SET course\_duration = '6 months'

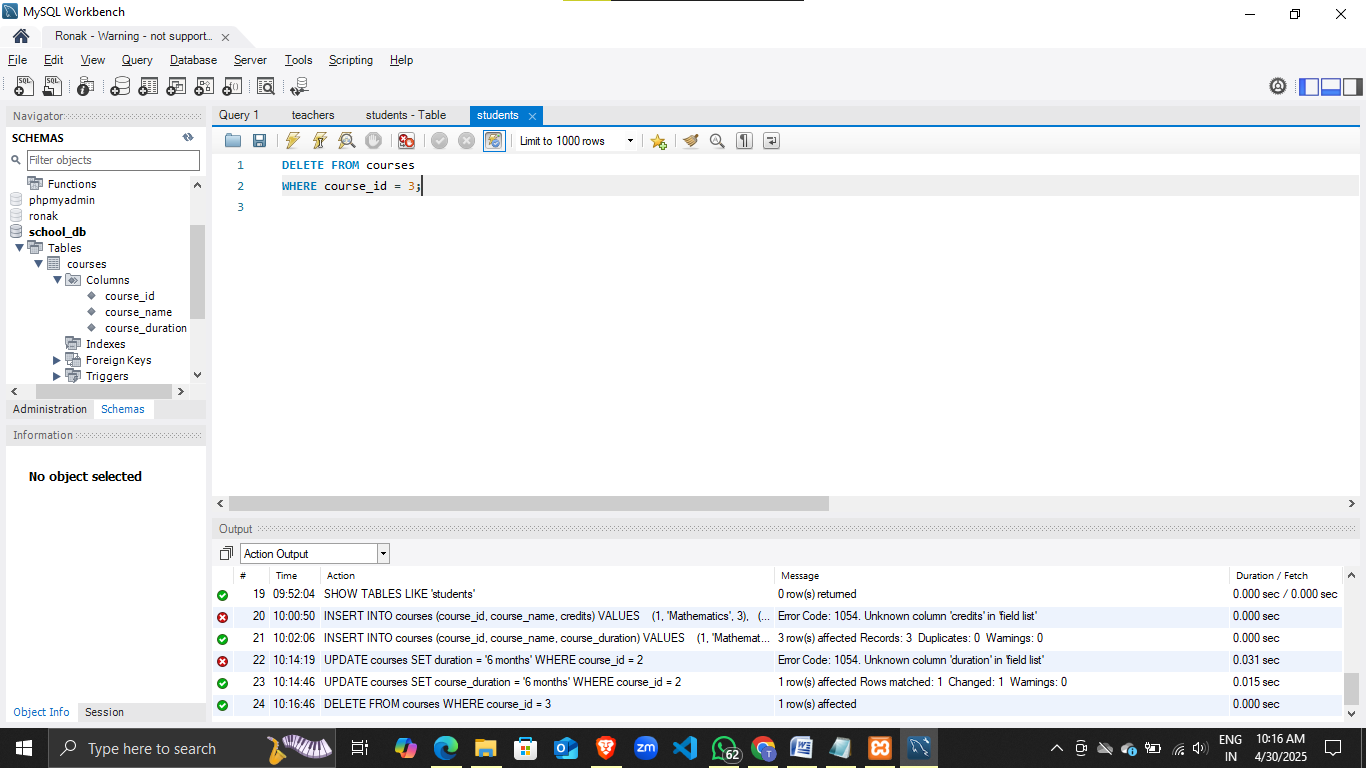
WHERE course\_id = 2;



Lab 3: Delete a course with a specific course\_id from the courses table using the DELETE command.

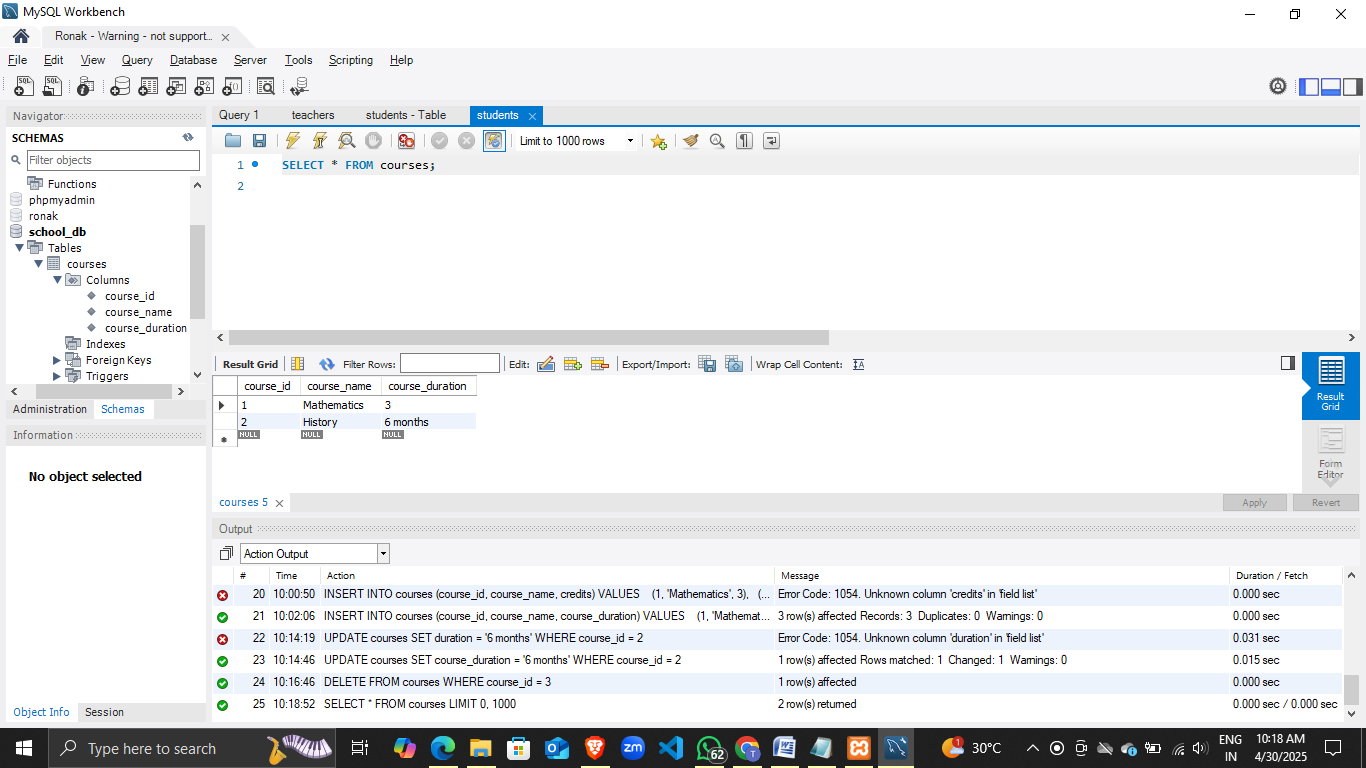
Ans. DELETE FROM courses

WHERE course\_id = 3;



Lab 1: Retrieve all courses from the courses table using the SELECT statement.

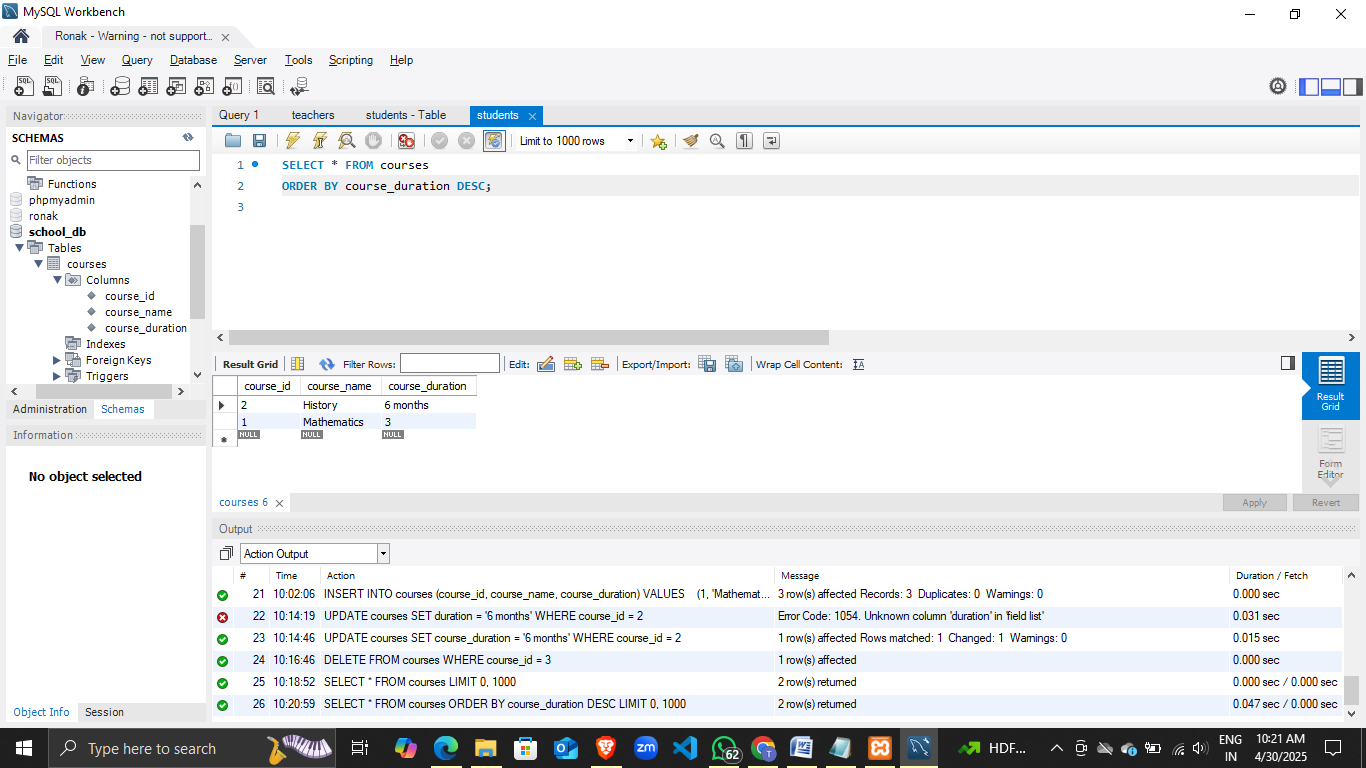
Ans. SELECT \* FROM courses;



Lab 2: Sort the courses based on course\_duration in descending order using ORDER BY.

Ans. SELECT \* FROM courses

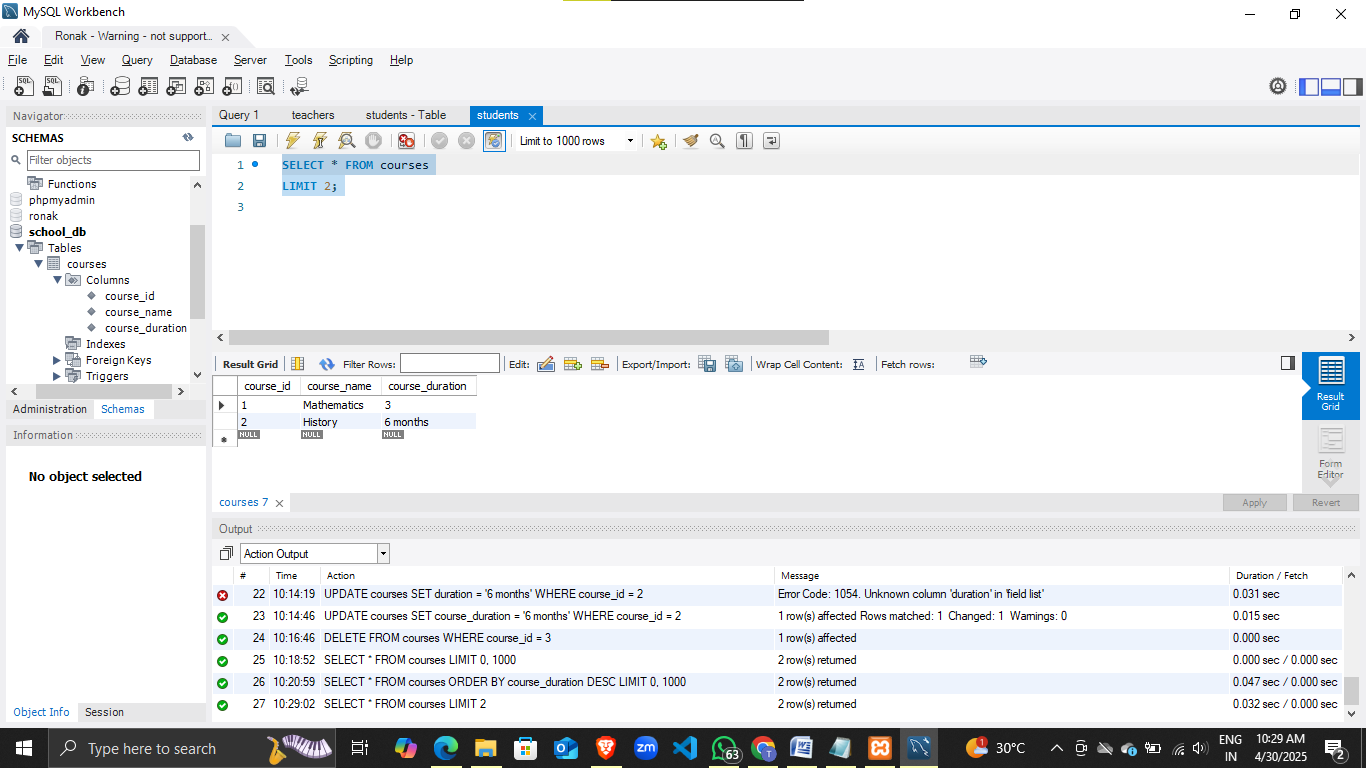
ORDER BY course\_duration DESC;



Lab 3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

Ans. SELECT \* FROM courses

LIMIT 2;

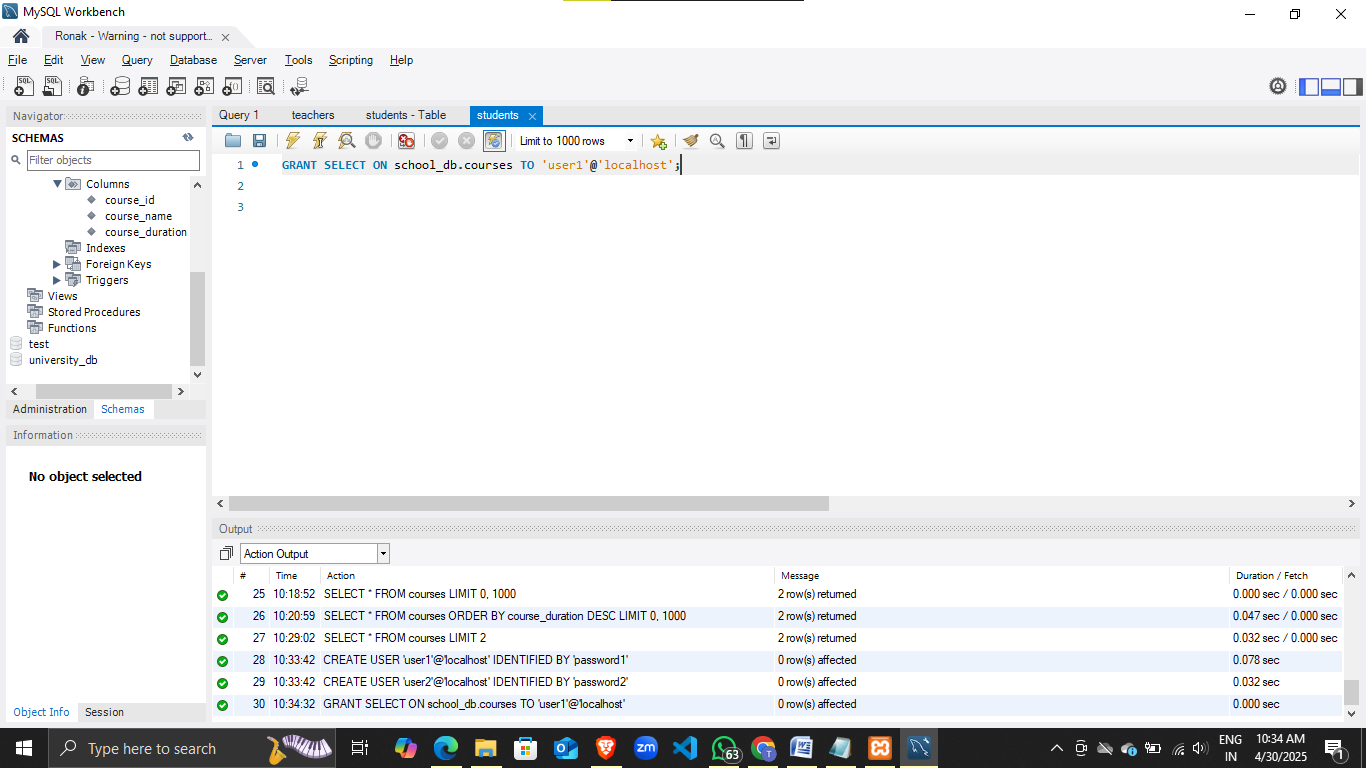


Lab 1: Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

Ans. CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';

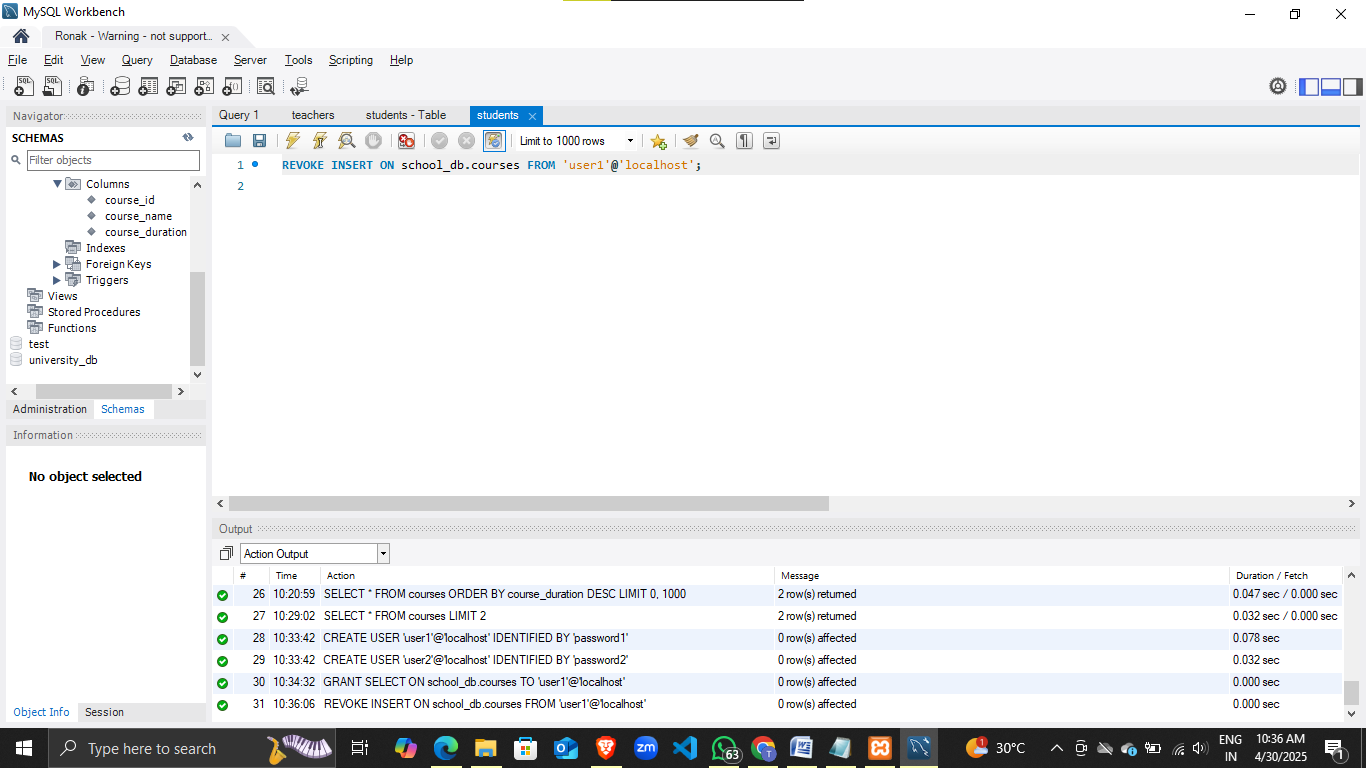
CREATE USER 'user2'@'localhost' IDENTIFIED BY 'password2';

GRANT SELECT ON school\_db.courses TO 'user1'@'localhost';

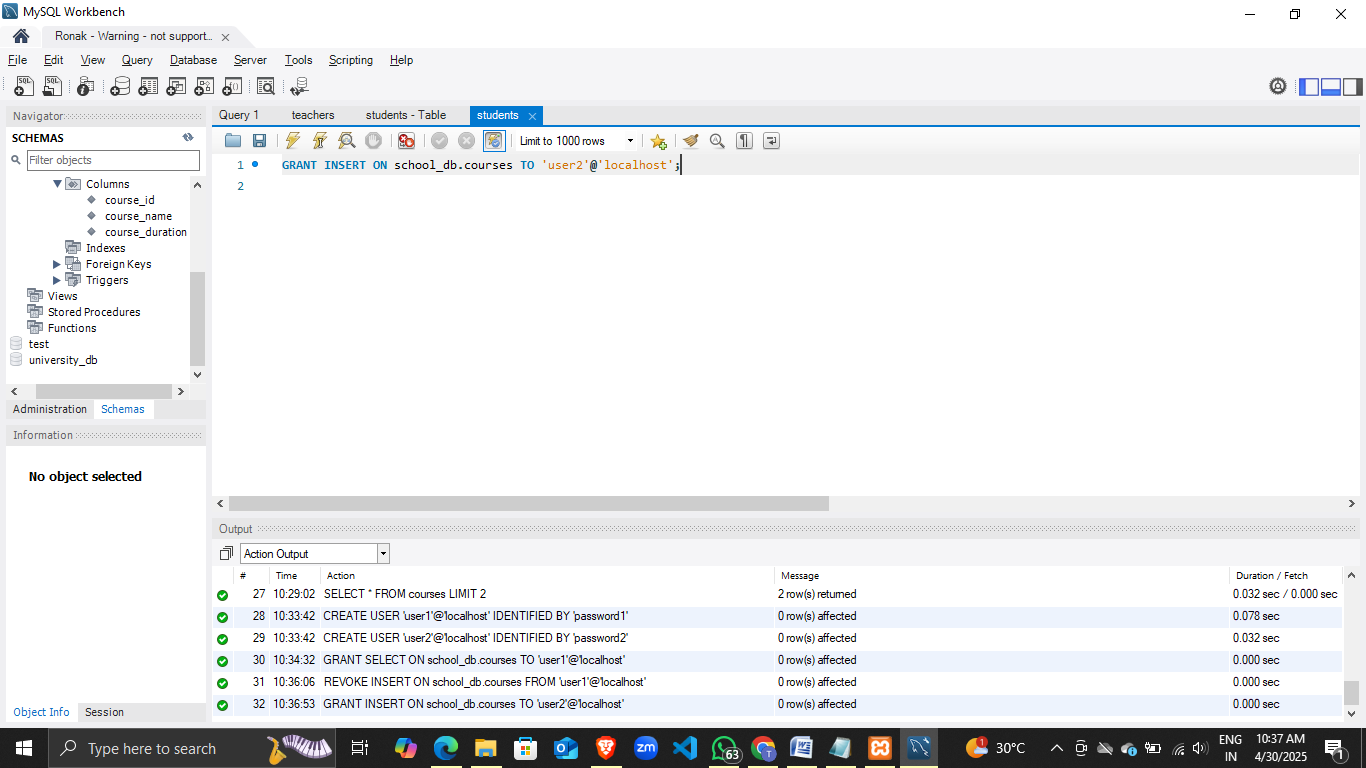


Lab 2: Revoke the INSERT permission from user1 and give it to user2.

Ans. REVOKE INSERT ON school\_db.courses FROM 'user1'@'localhost';



GRANT INSERT ON school\_db.courses TO 'user2'@'localhost';



Lab 1: Insert a few rows into the courses table and use COMMIT to save the changes.

Ans. START TRANSACTION;

INSERT INTO courses (course\_id, course\_name, course\_duration)

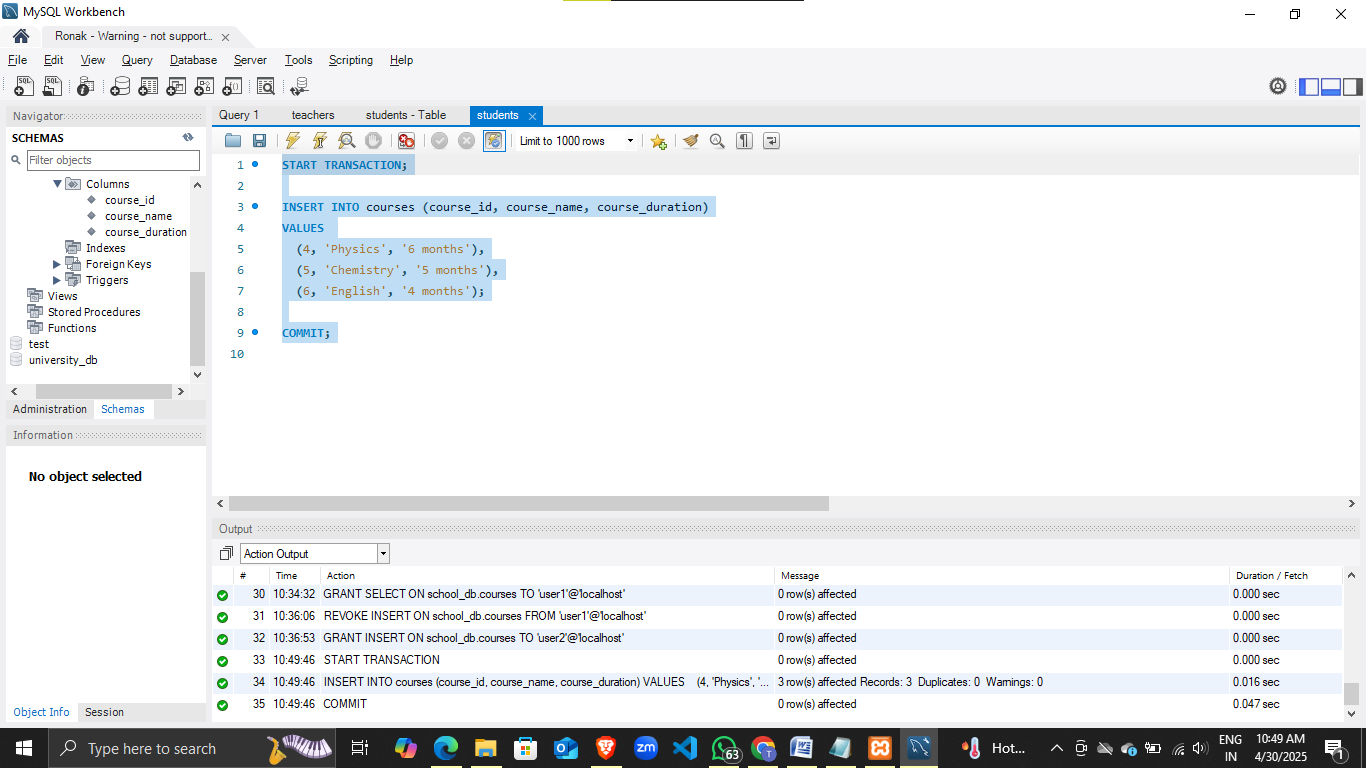
VALUES

(4, 'Physics', '6 months'),

(5, 'Chemistry', '5 months'),

(6, 'English', '4 months');

COMMIT;



Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

Ans. START TRANSACTION;

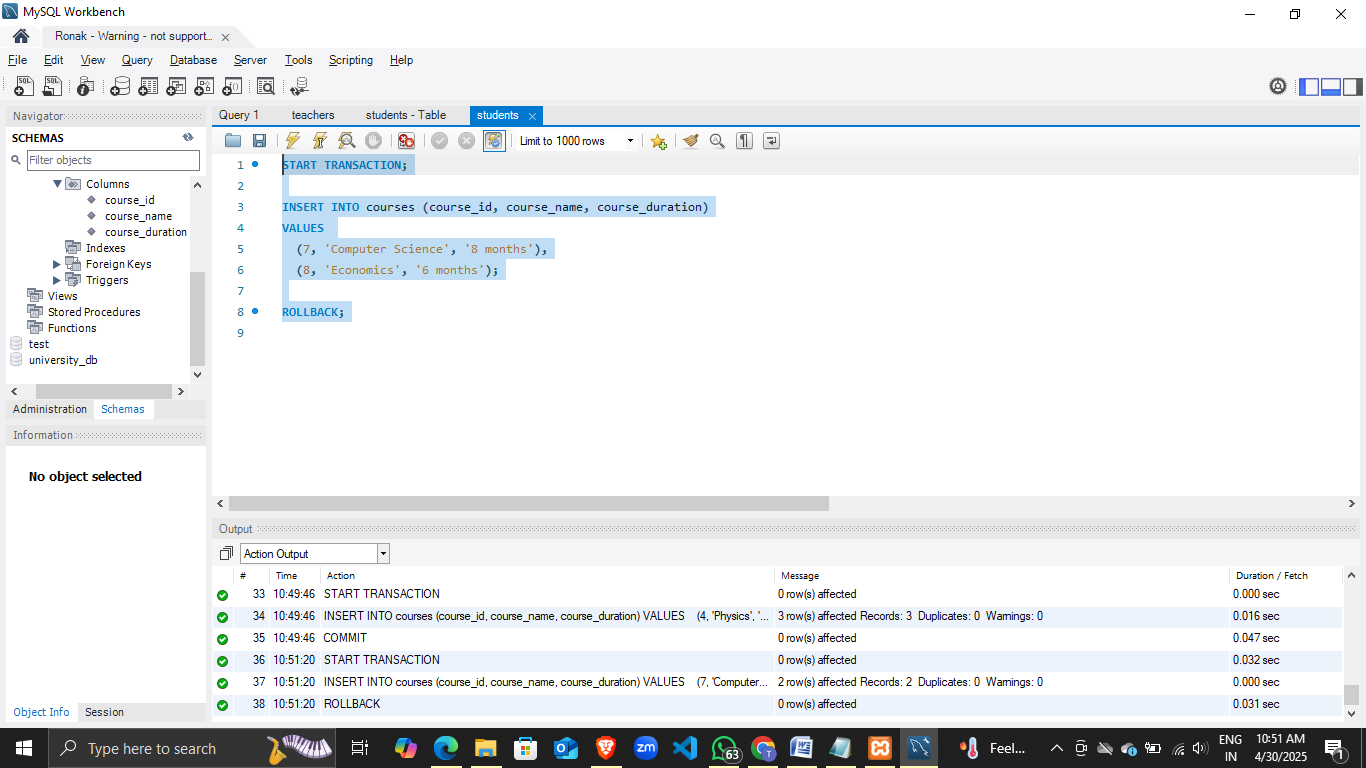
INSERT INTO courses (course\_id, course\_name, course\_duration)

VALUES

(7, 'Computer Science', '8 months'),

(8, 'Economics', '6 months');

ROLLBACK;



Lab 3: Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

Ans. START TRANSACTION;

SAVEPOINT before\_update;

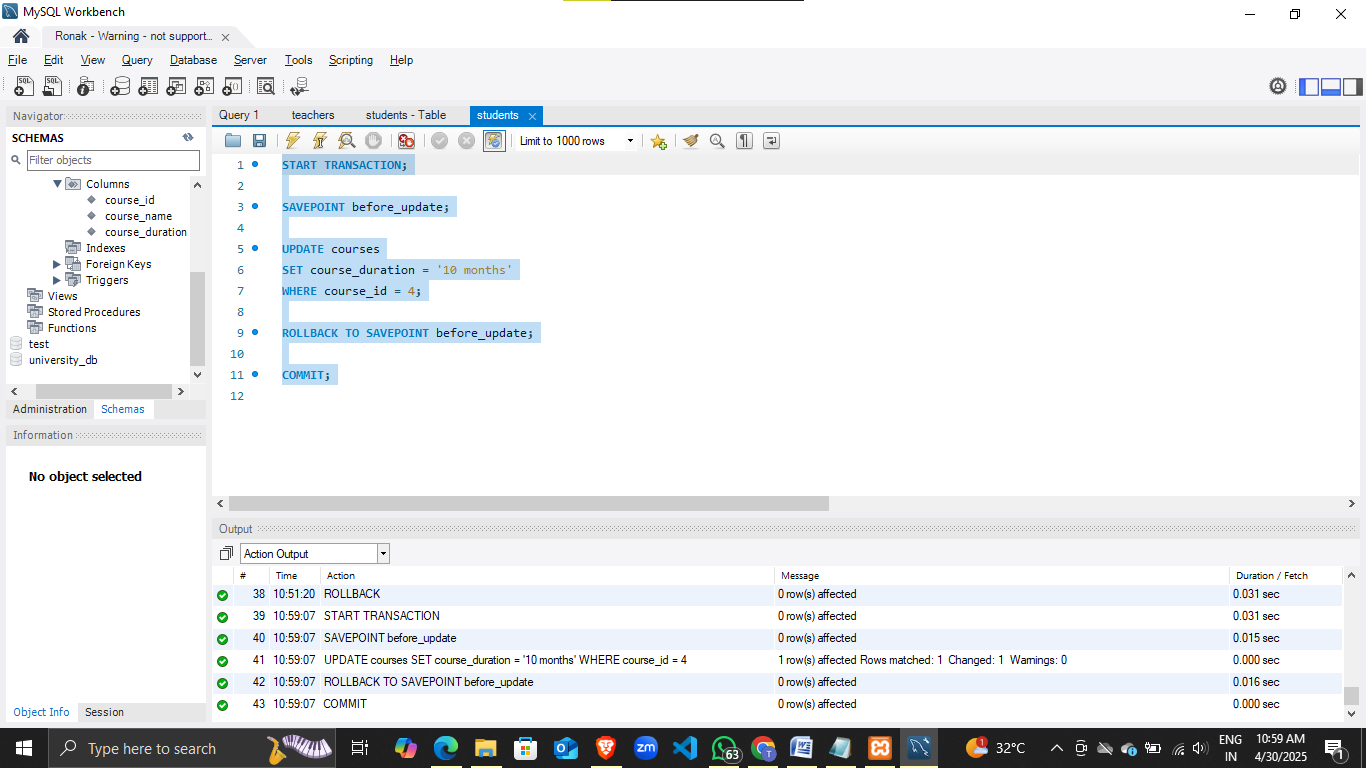
UPDATE courses

SET course\_duration = '10 months'

WHERE course\_id = 4;

ROLLBACK TO SAVEPOINT before\_update;

COMMIT;



Lab 1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

Ans. CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(100)

);