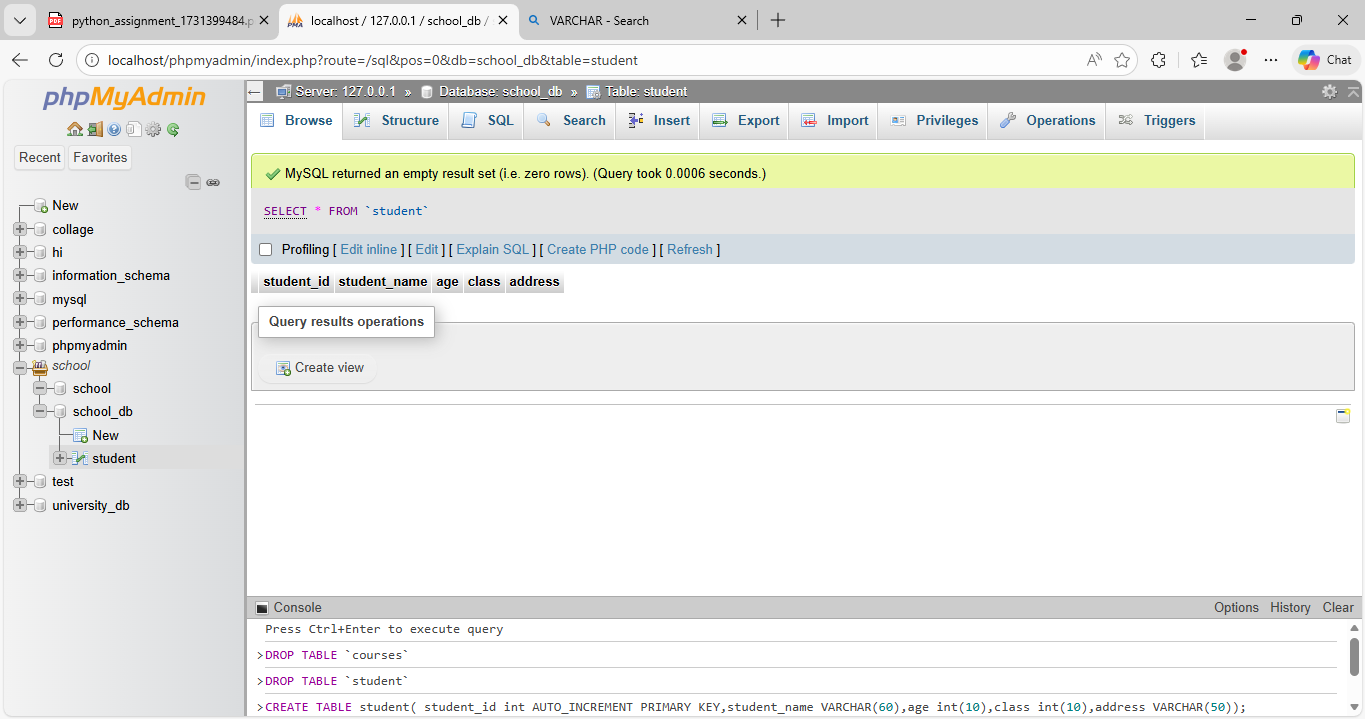
**Module - 4 Introduction to DBMS**

1. **Introduction to SQL**

**Leb 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.**

CREATE TABLE student( student\_id int AUTO\_INCREMENT PRIMARY KEY,student\_name VARCHAR(60),age int(10),class int(10),address VARCHAR(50));



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

INSERT INTO student (student\_name,age,class,address)VALUES

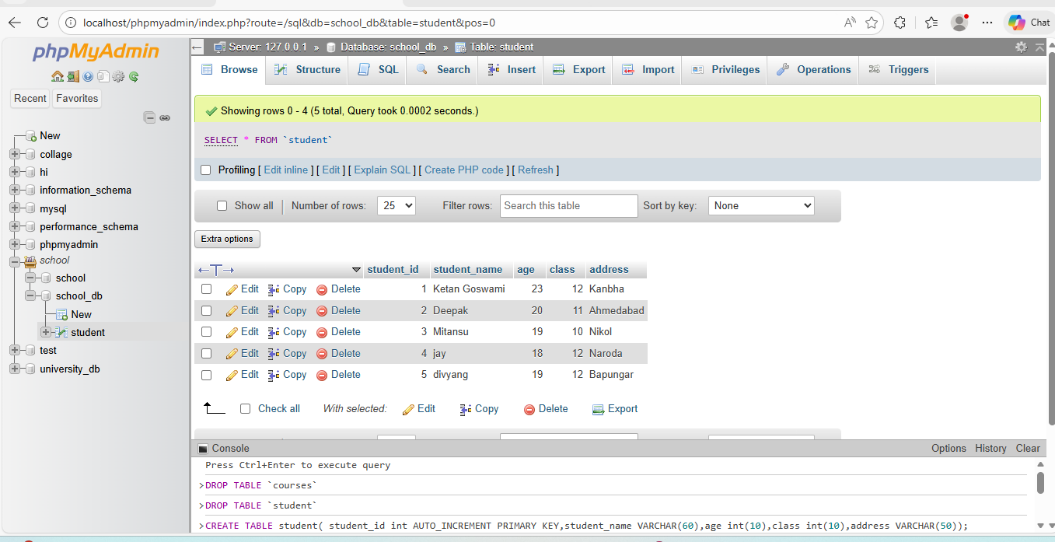
("Ketan Goswami",23,12,"Kanbha"),

("Deepak",20,11,"Ahmedabad"),

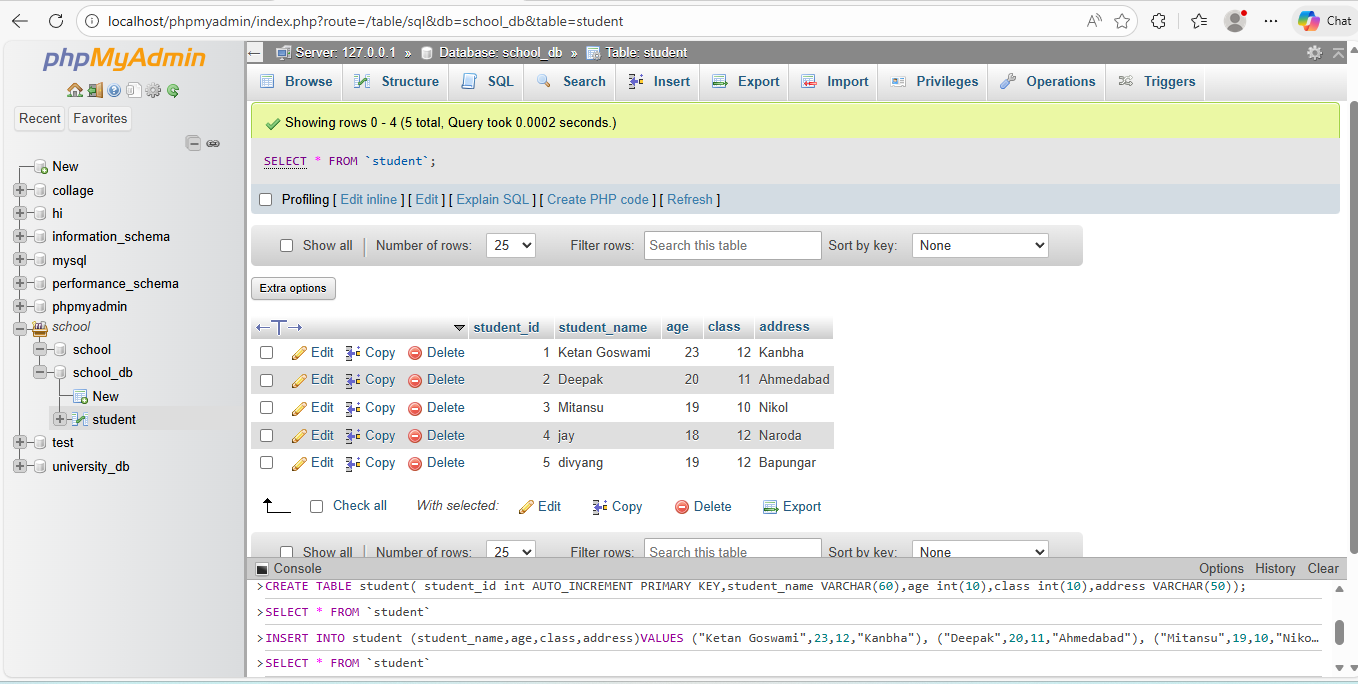
("Mitansu",19,10,"Nikol"),

("jay",18,12,"Naroda"),

("divyang",19,12," Bapungar");



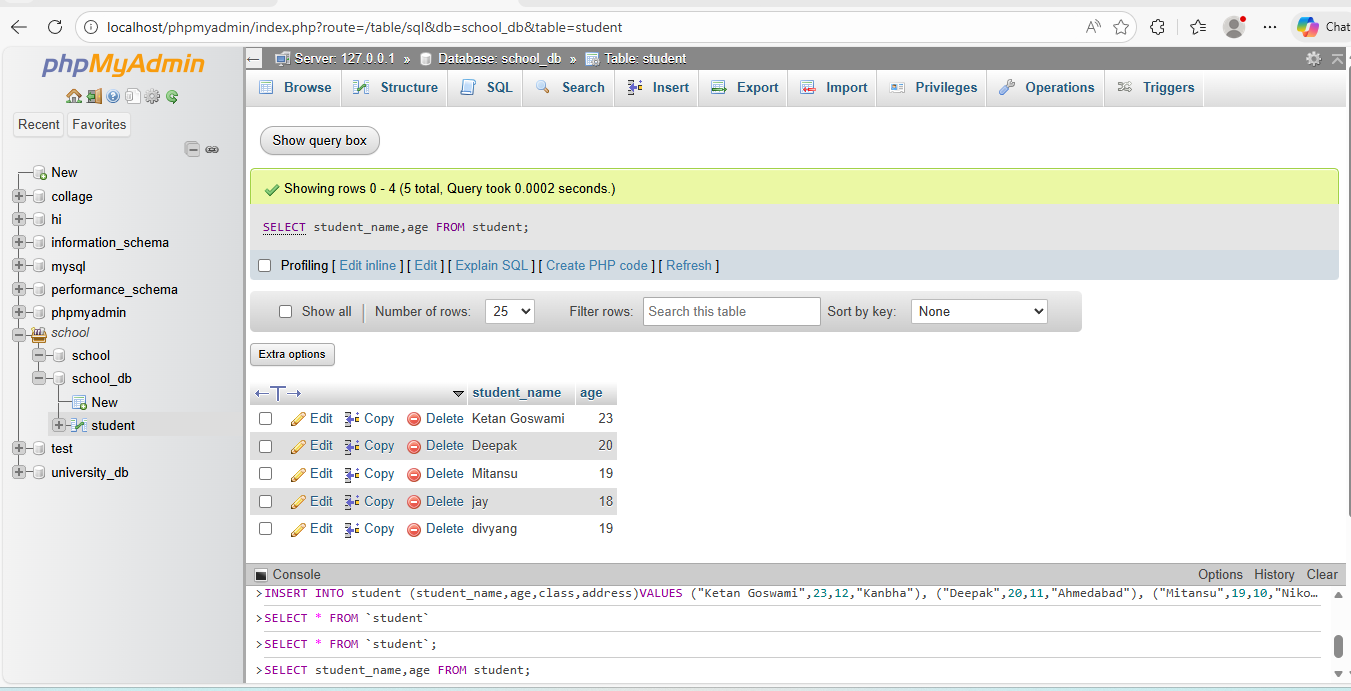
SELECT \* FROM student;



1. **SQL Syntax**

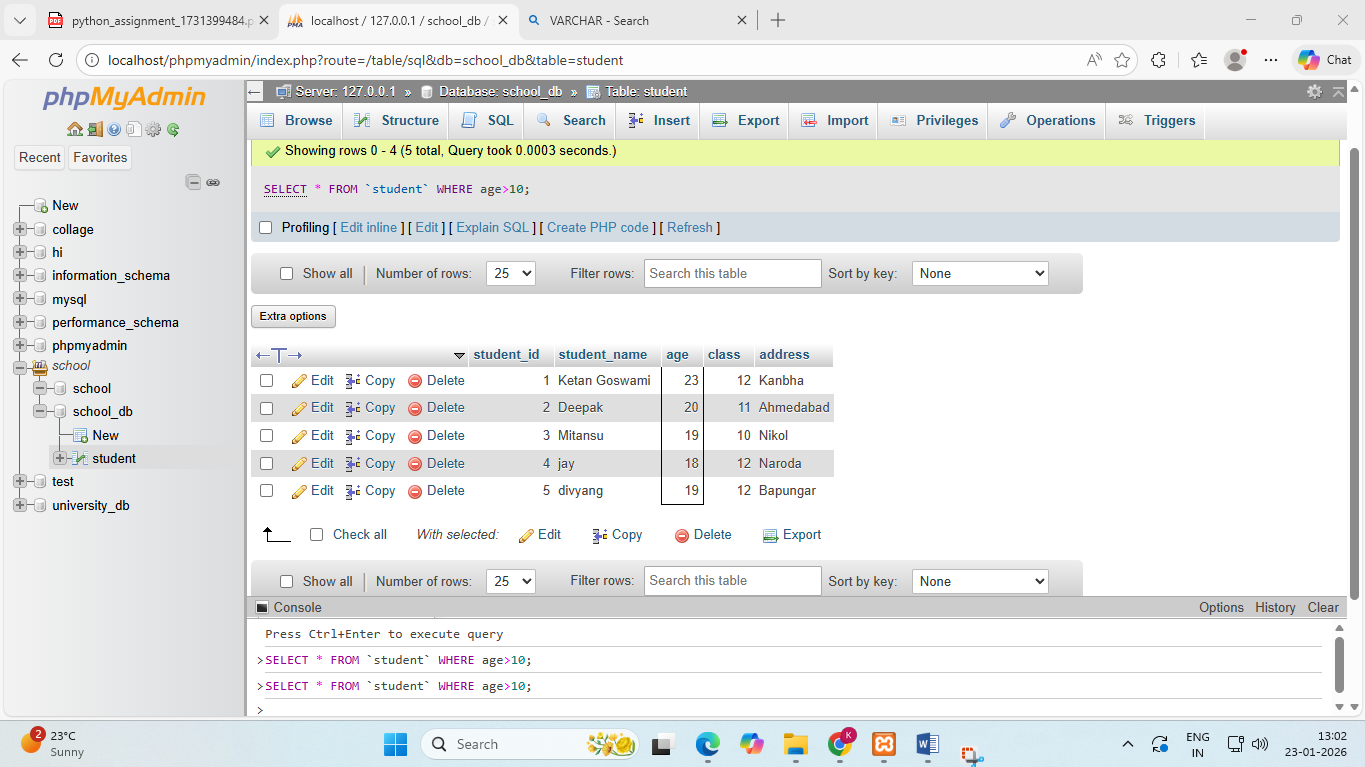
**Lab 1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.**

SELECT student\_name,age FROM student;



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

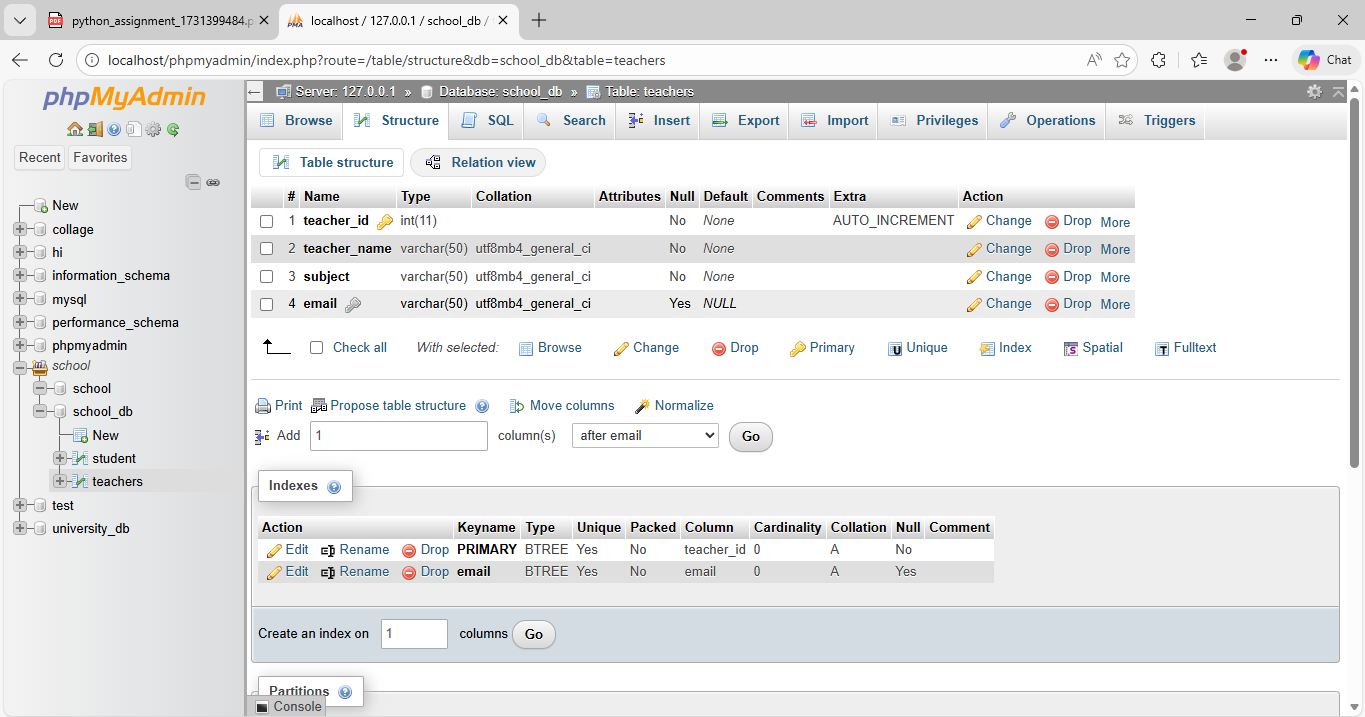
SELECT \* FROM student WHERE age>10;



1. **SQL Constraints**

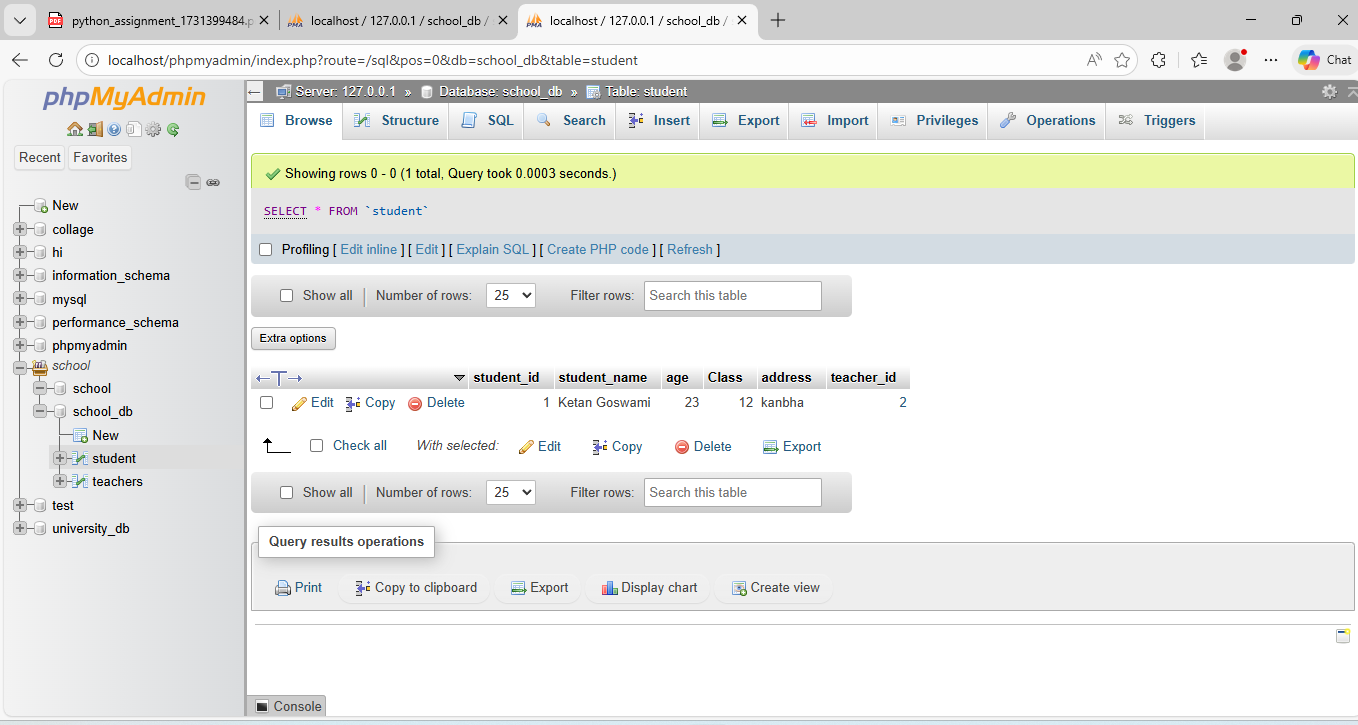
**CREATE TABLE teachers(teacher\_id INT PRIMARY KEY AUTO\_INCREMENT,teacher\_name VARCHAR(50)NOT NULL, subject VARCHAR(50) NOT NULL,email VARCHAR(50) UNIQUE KEY)**

CREATE TABLE teachers(teacher\_id INT PRIMARY KEY AUTO\_INCREMENT,teacher\_name VARCHAR(50)NOT NULL, subject VARCHAR(50) NOT NULL,email VARCHAR(50) UNIQUE KEY);



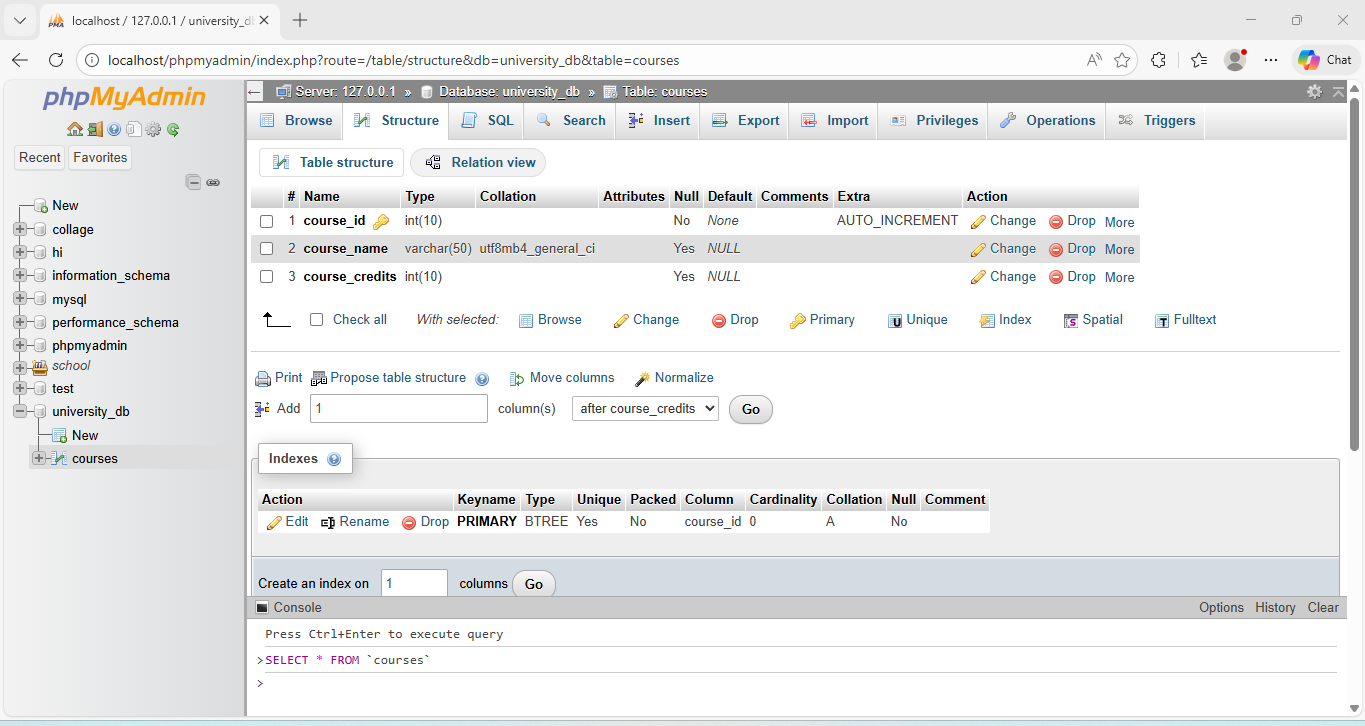
**Lab 2: Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.**

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) student(student\_id INT(5) PRIMARY KEY AUTO\_INCREMENT,student\_name VARCHAR(50),age INT(5),Class int(10),address VARCHAR(50),teacher\_id INT(5),FOREIGN KEY(teacher\_id) REFERENCES teachers(teachers\_id));



1. **Main SQL Commands and Sub-commands (DDL)**

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



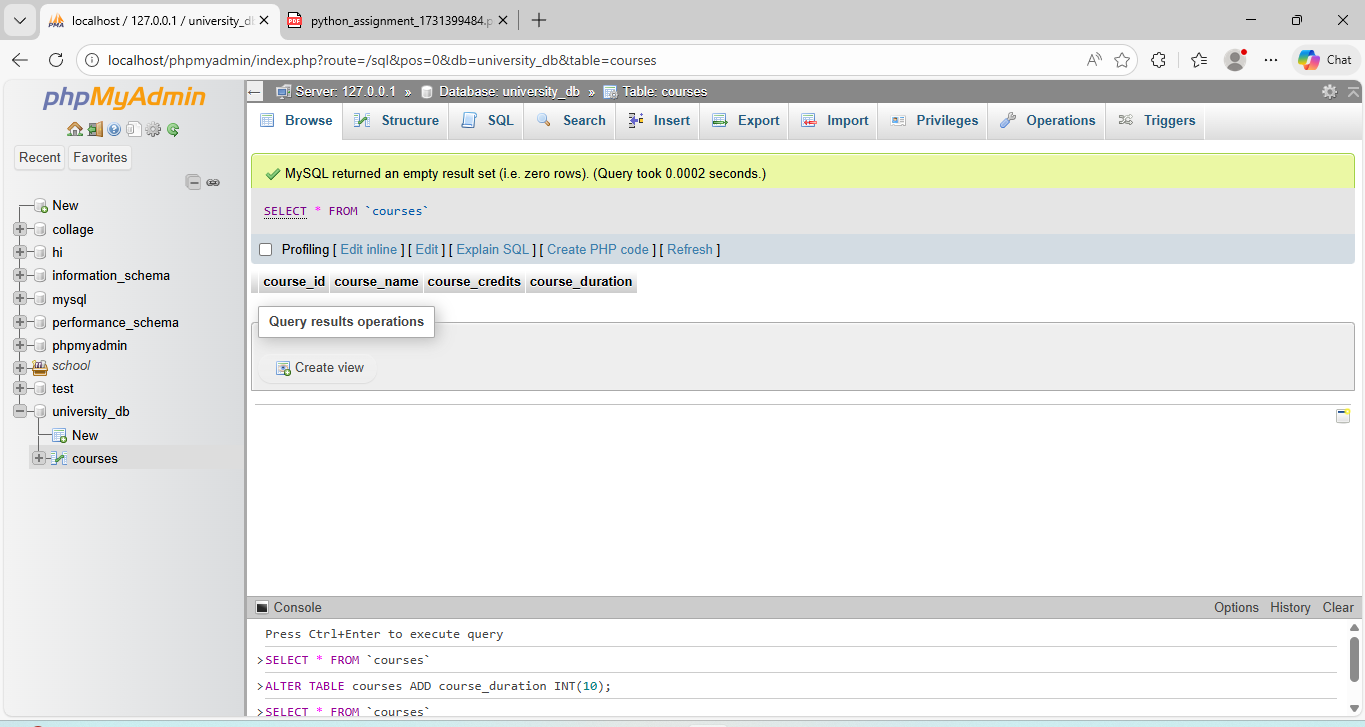
**Lab 2: Use the CREATE command to create a database university\_db.**

CREAT DATABASE university\_db;

1. **Main SQL Commands and Sub-commands (DDL)**

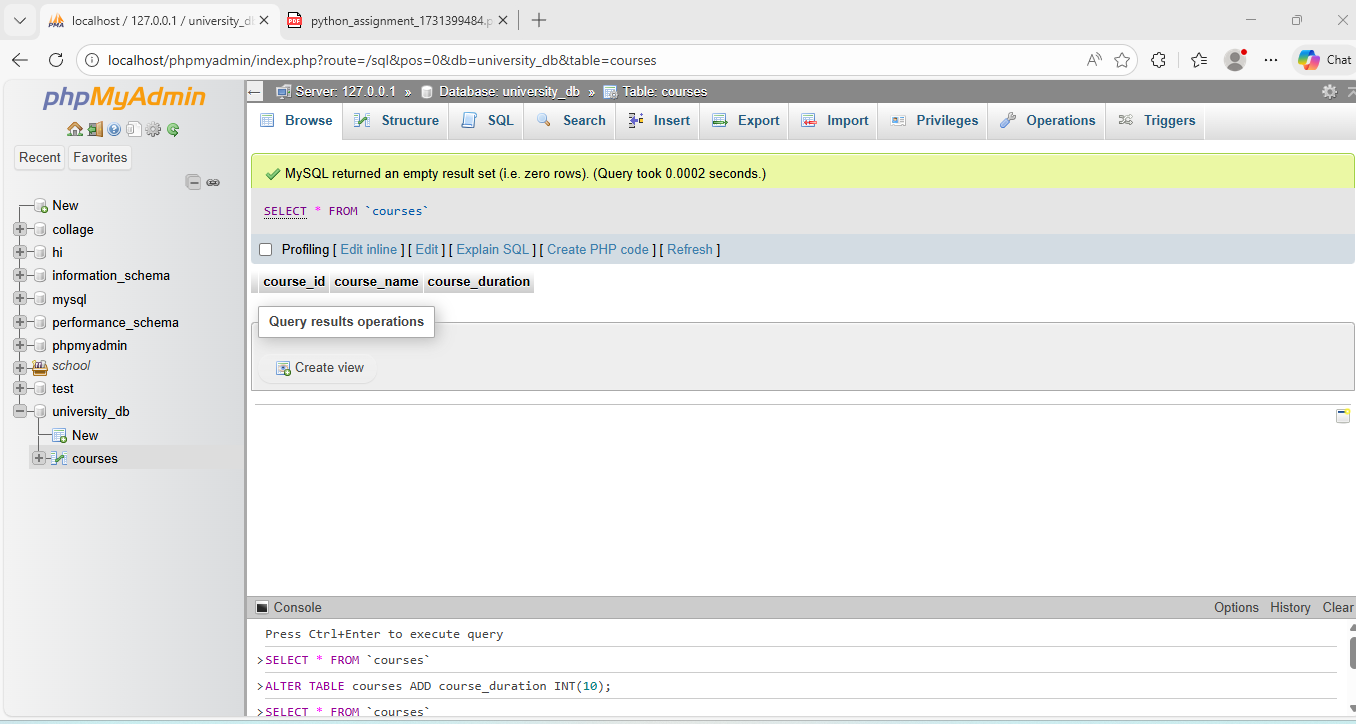
**Lab 1: Modify the courses table by adding a column course\_duration using the ALTER command.**

**ALTER TABLE courses ADD course\_duration INT(10);**

****

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

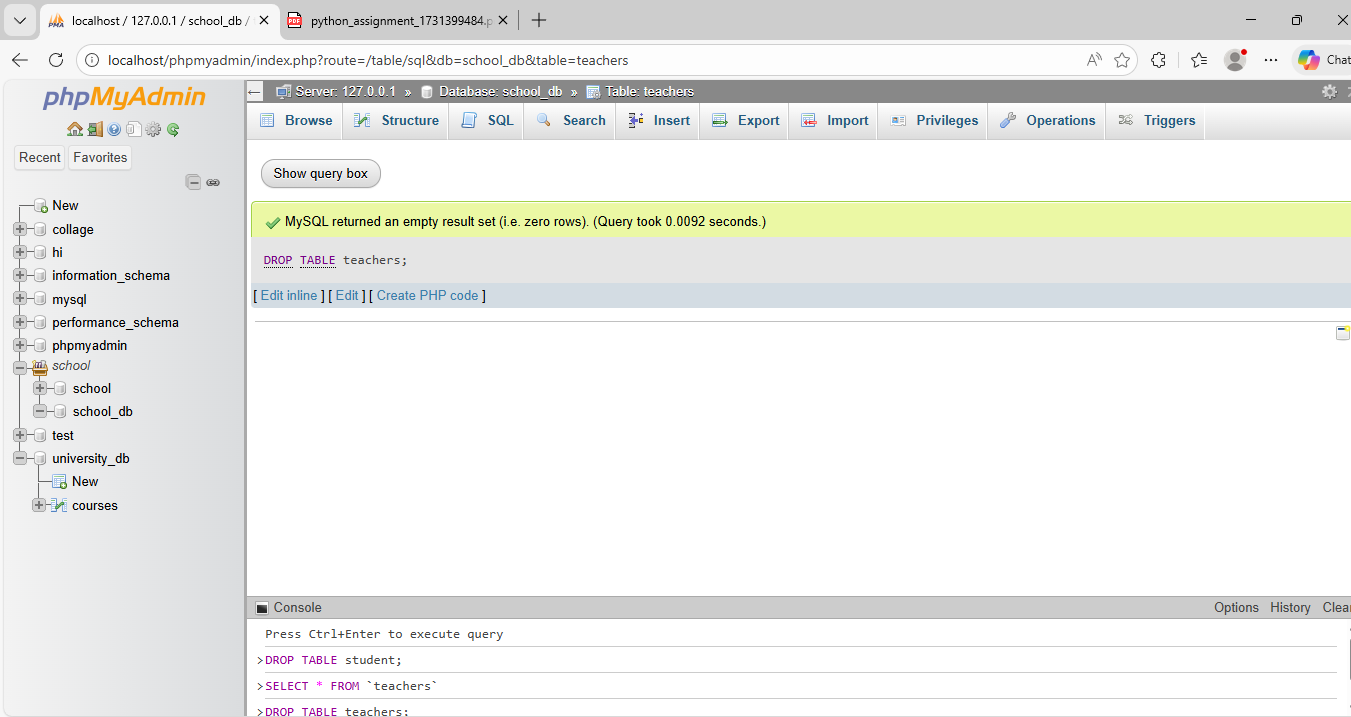
**ALTER TABLE courses DROP course\_credits;**

****

1. **DROP Command**

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

**DROP TABLE teachers;**

****

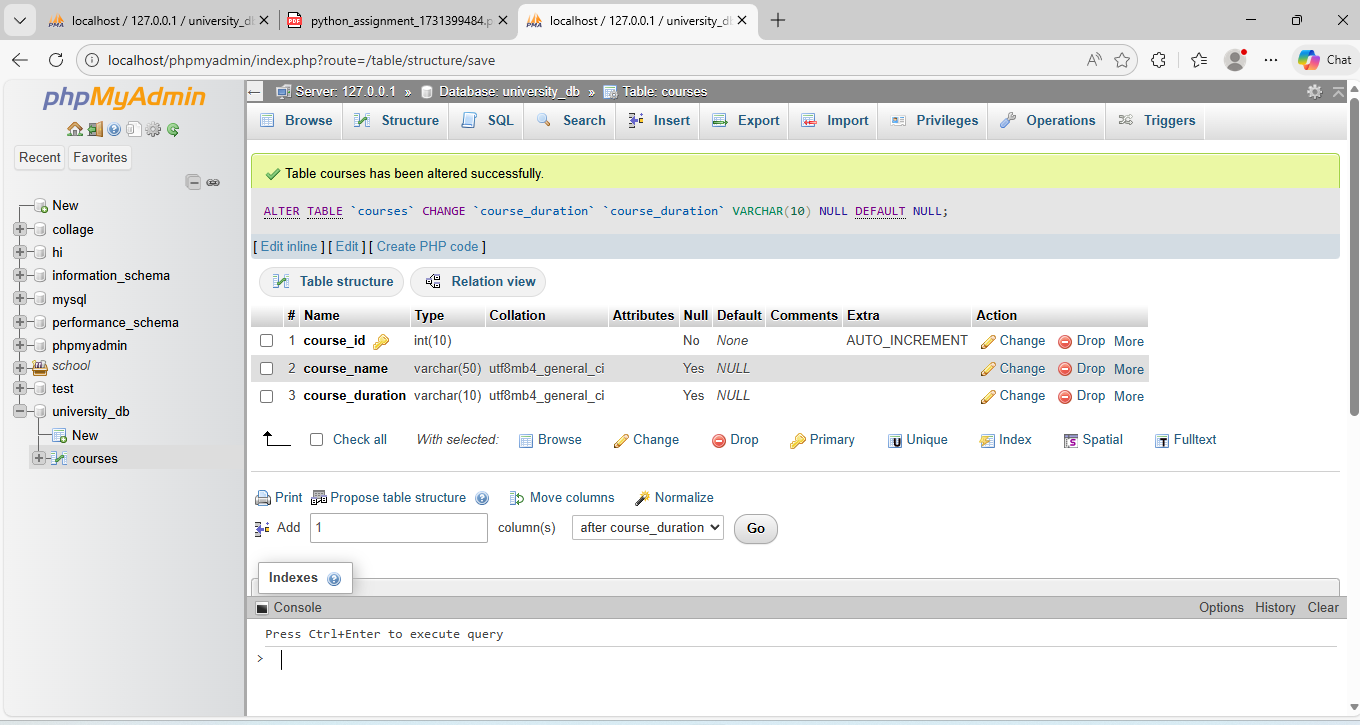
** Lab 2: Drop the students table from the school\_db database and verify that the table has been removed.**

**DROP TABLE studens;**

1. **Data Manipulation Language (DML)**

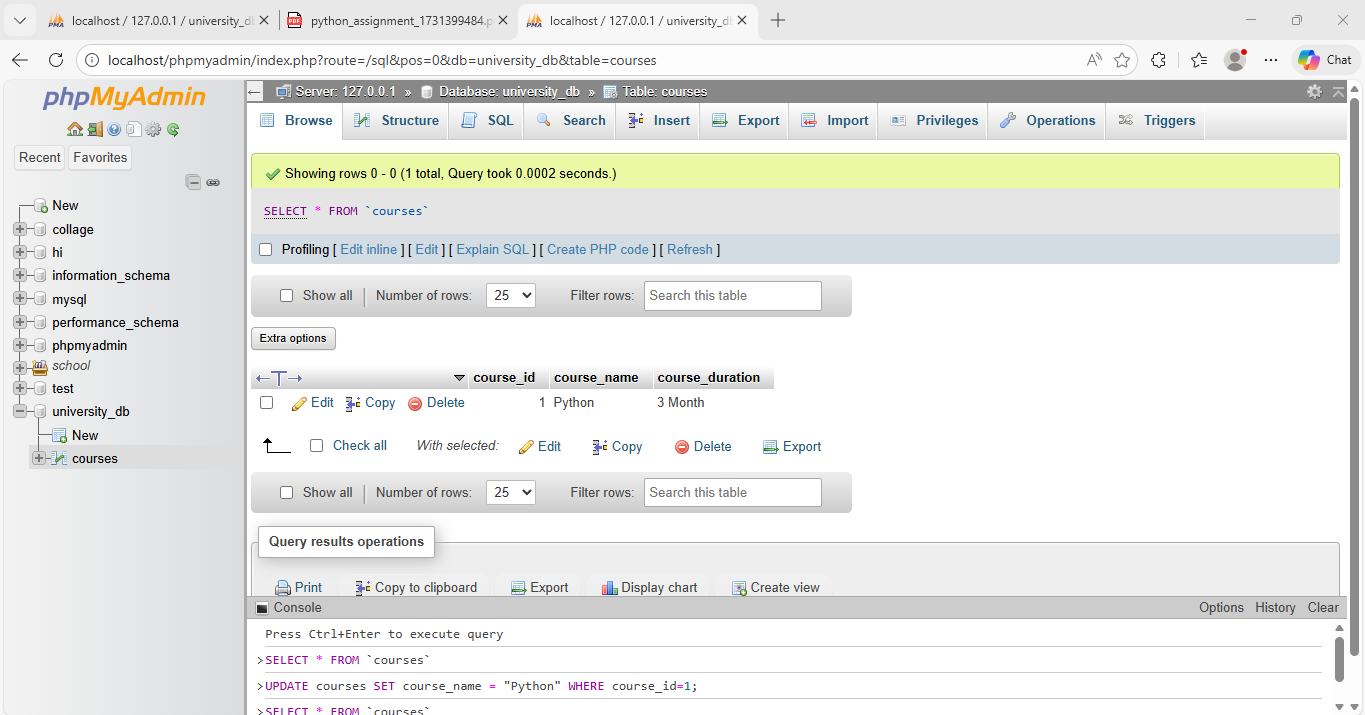
**Lab 1: Insert three records into the courses table using the INSERT command.**

**INSERT INTO courses(course\_name,course\_duration)VALUE("Java","3 Month")**

****

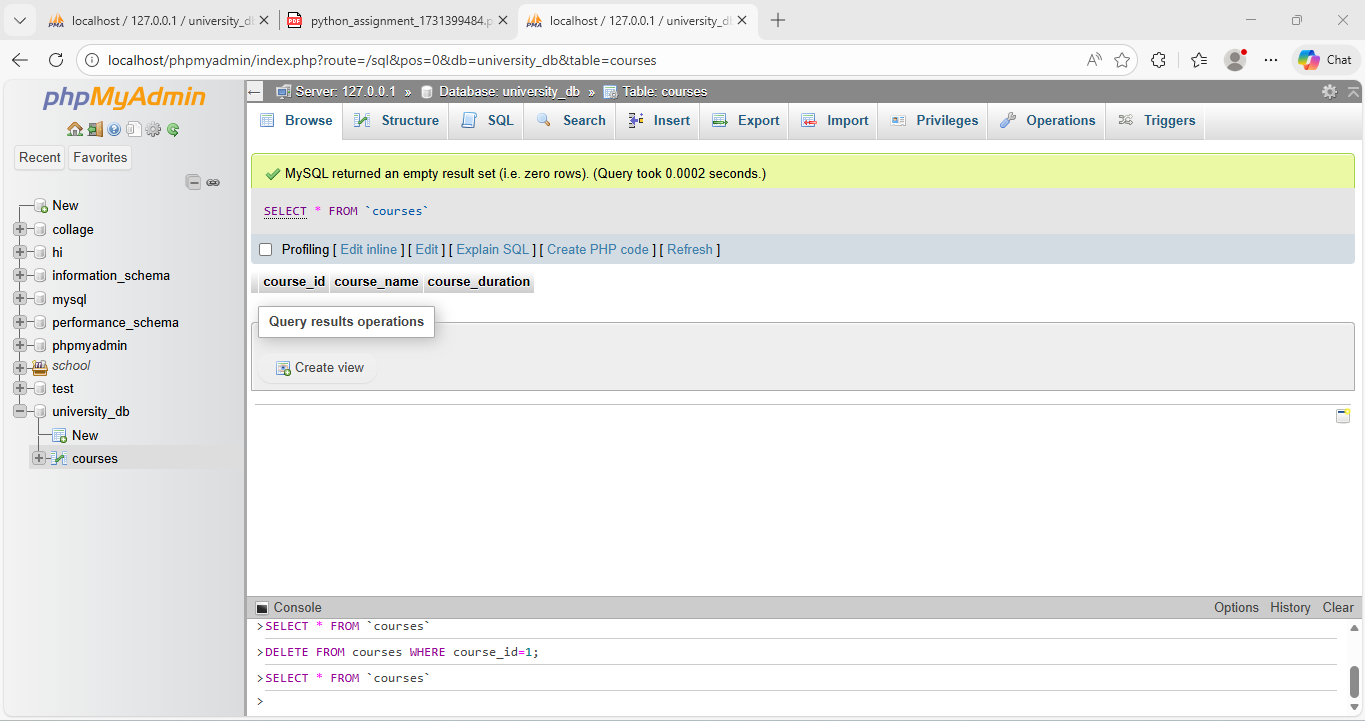
**Lab 2: Update the course duration of a specific course using the UPDATE command.**

**UPDATE courses SET course\_name = "Python" WHERE course\_id=1;**

****

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

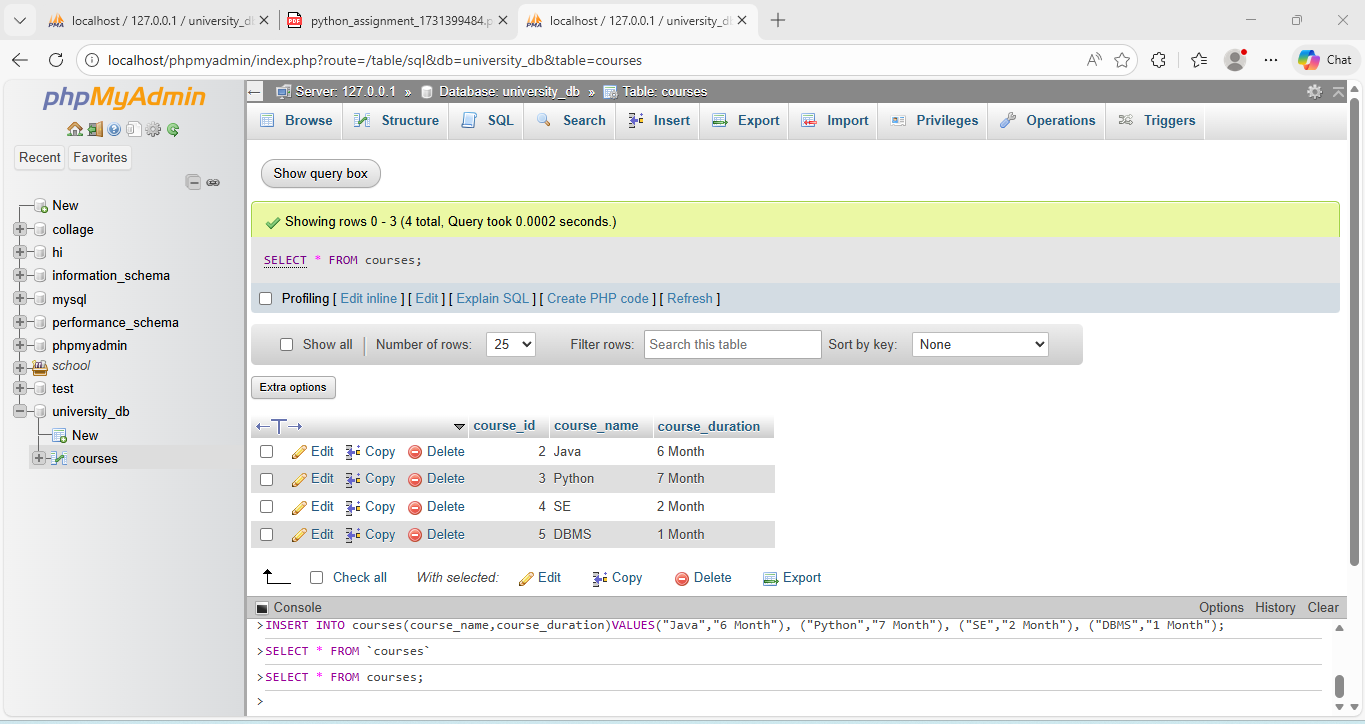
**DELETE FROM courses WHERE course\_id=1;**

****

1. **Data Query Language (DQL)**

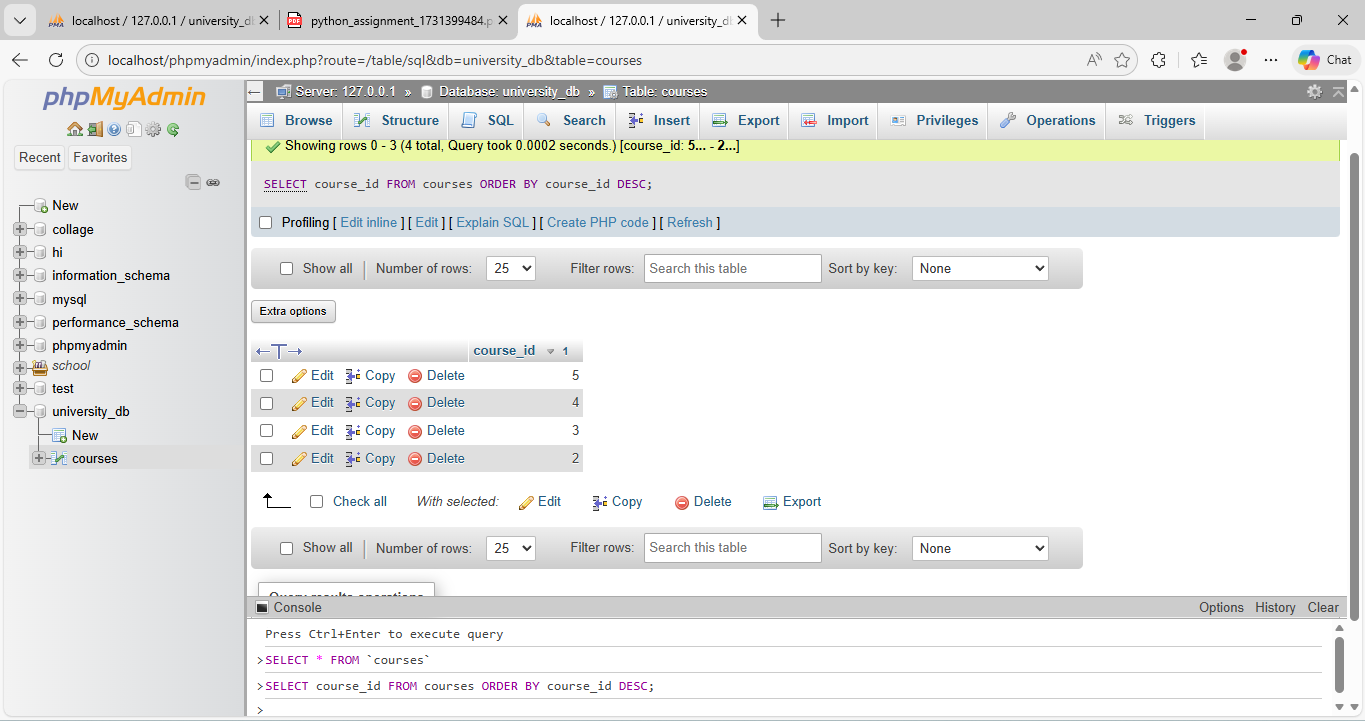
**Lab 1: Retrieve all courses from the courses table using the SELECT statement.**

**SELECT \* FROM courses;**

****

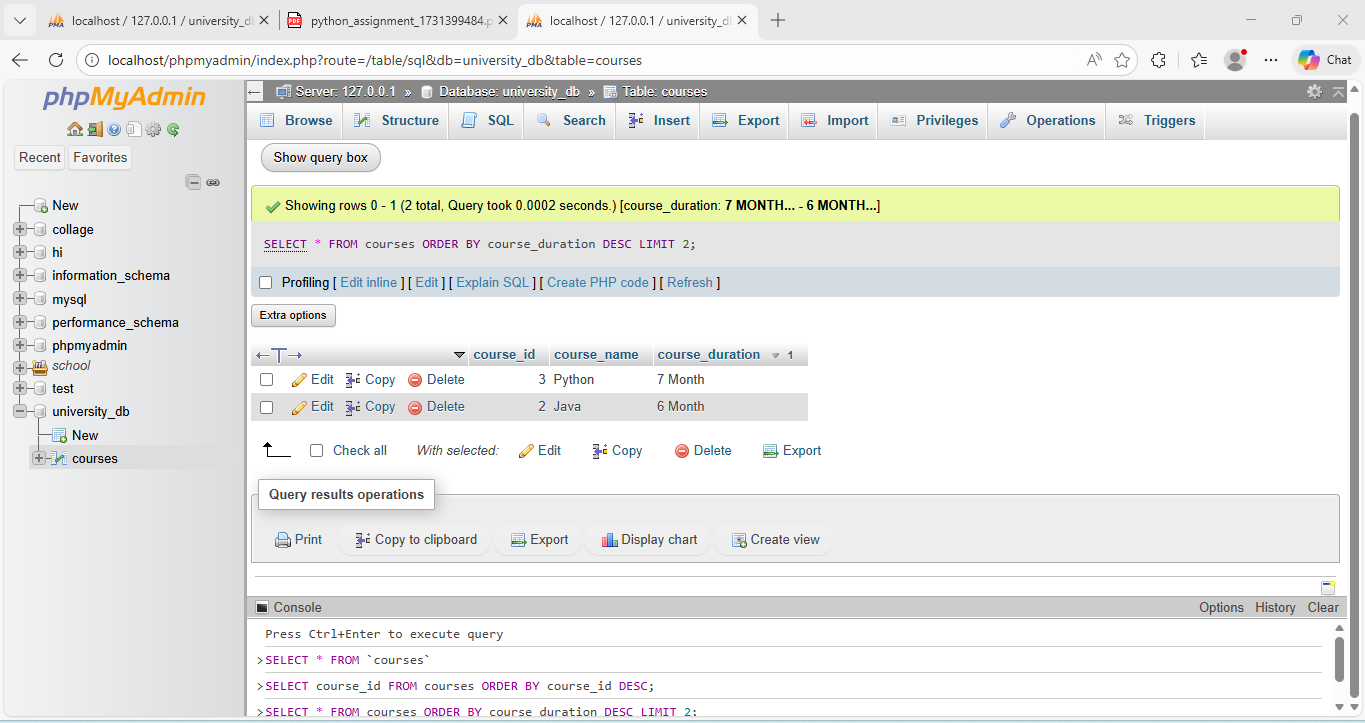
**Lab 2: Sort the courses based on course\_duration in descending order using ORDERBY**

**SELECT course\_id FROM courses ORDER BY course\_id DESC;**

****

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

**SELECT \* FROM courses ORDER BY course\_duration DESC LIMIT 2;**

****

1. **SQL Joins**

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

**CREATE TABLE departments(departments\_id INT(10) PRIMARY KEY AUTO\_INCREMENT,departments\_name VARCHAR(50));**

**INSERT INTO departments(departments\_name)VALUES("IT"),("HR"),**

**("Account");**

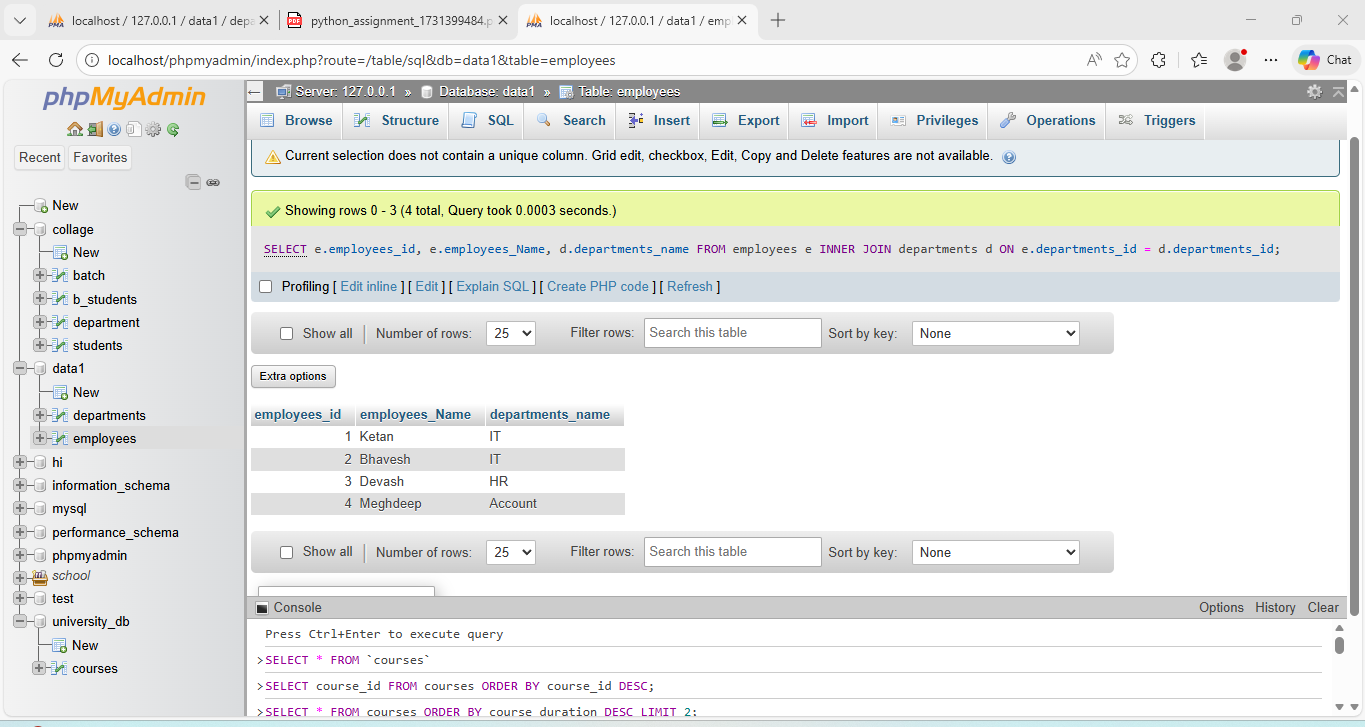
**CREATE TABLE employees (employees\_id INT(10) PRIMARY KEY AUTO\_INCREMENT,employees\_Name VARCHAR(50),departments\_id INT(10),**

**FOREIGN KEY(departments\_id) REFERENCES departments(departments\_id));**

**INSERT INTO employees (employees\_Name,departments\_id)**

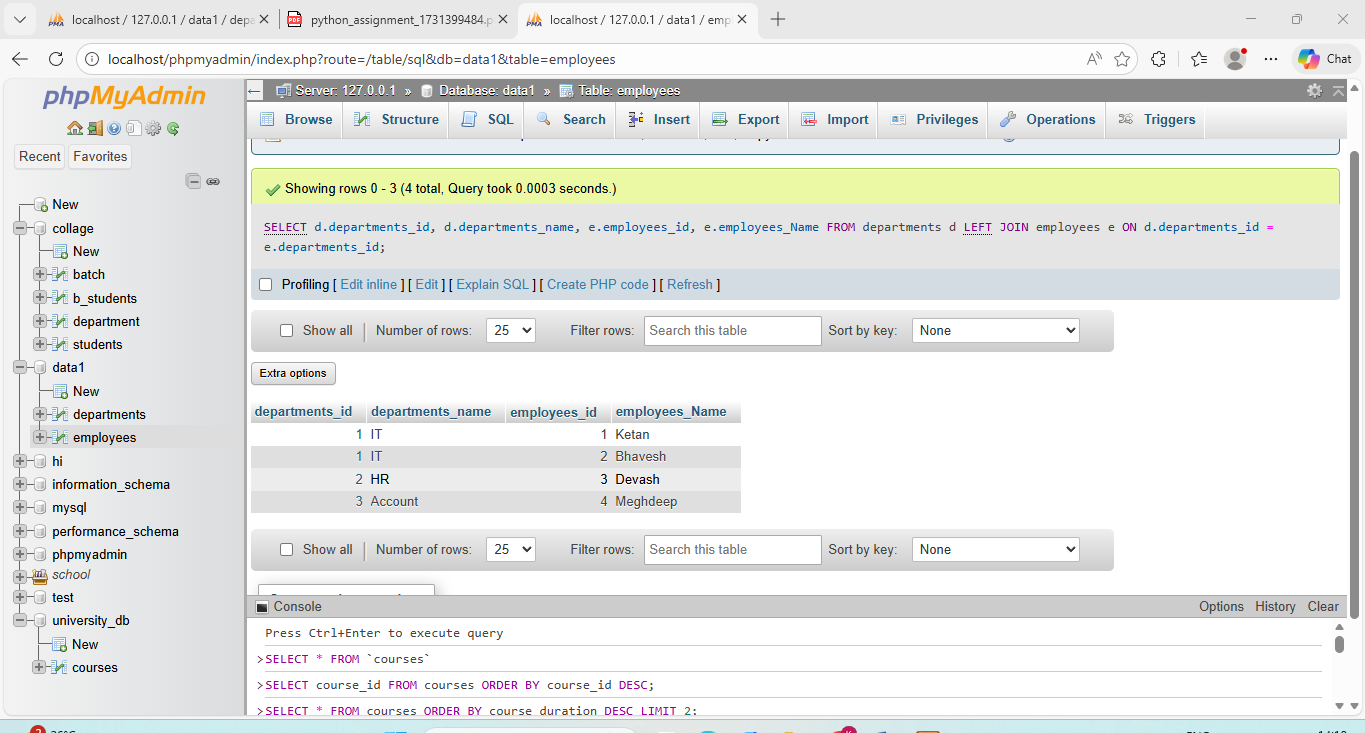
**VALUES ("Ketan",1),("Bhavesh",1),("Devash",2),("Meghdeep",3);**

**SELECT e.employees\_id, e.employees\_Name, d.departments\_name FROM employees e INNER JOIN departments d ON e.departments\_id = d.departments\_id;**

****

** Lab 2: Use a LEFT JOIN to show all departments, even those without employees.**

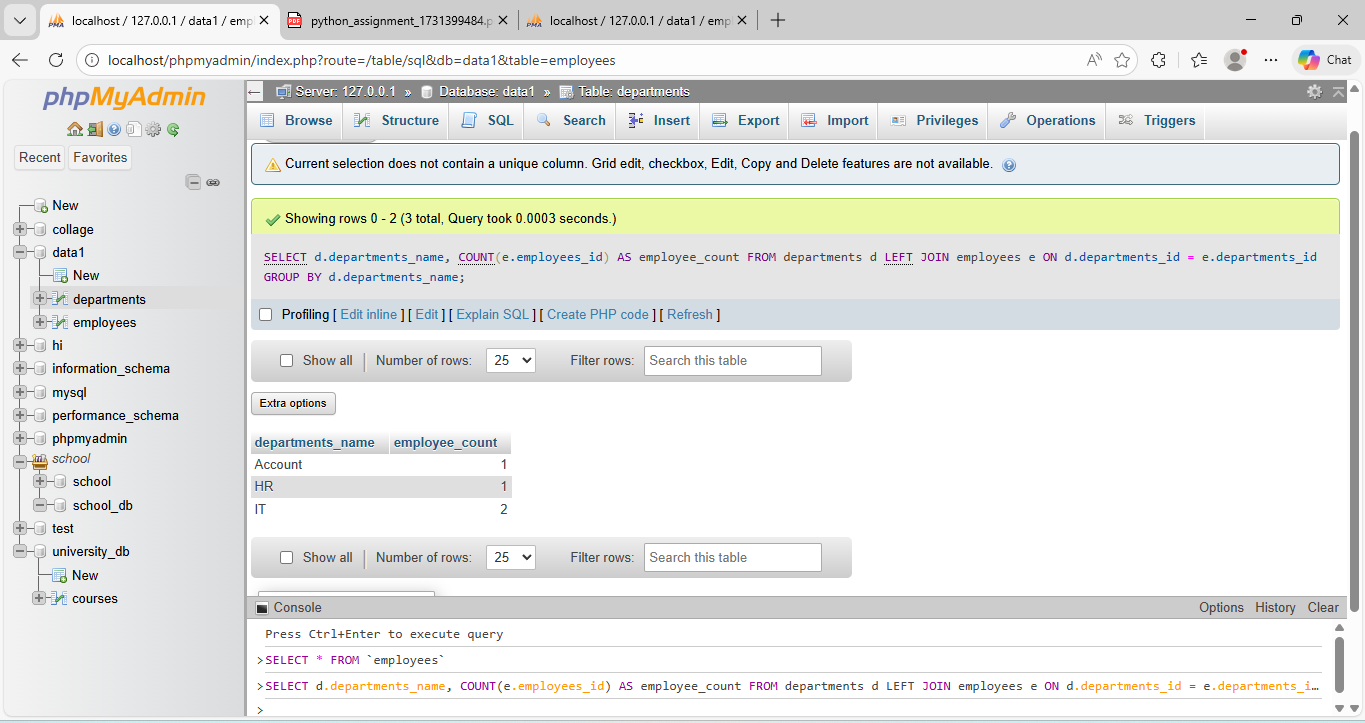
**SELECT d.departments\_id, d.departments\_name, e.employees\_id, e.employees\_Name FROM departments d LEFT JOIN employees e ON d.departments\_id = e.departments\_id;**

****

1. **SQL Group By**

**Lab 1: Group employees by department and count the number of employees in each department using GROUP BY.**

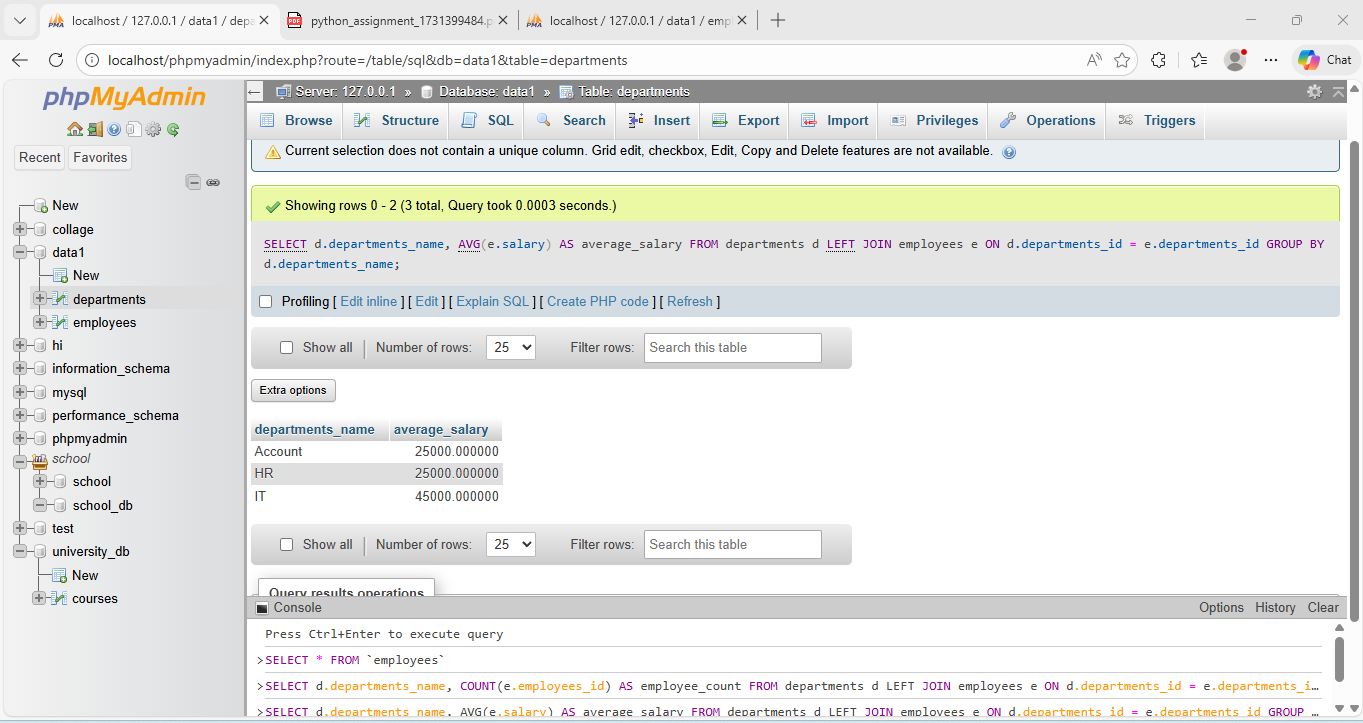
**SELECT d.departments\_name, COUNT(e.employees\_id) AS employee\_count FROM departments d LEFT JOIN employees e ON d.departments\_id = e.departments\_id GROUP BY d.departments\_name;**

****

**Lab 2: Use the AVG aggregate function to find the average salary of employees in each**

**department.**

**SELECT d.departments\_name, AVG(e.salary) AS average\_salary FROM departments d LEFT JOIN employees e ON d.departments\_id = e.departments\_id GROUP BY d.departments\_name;**

****