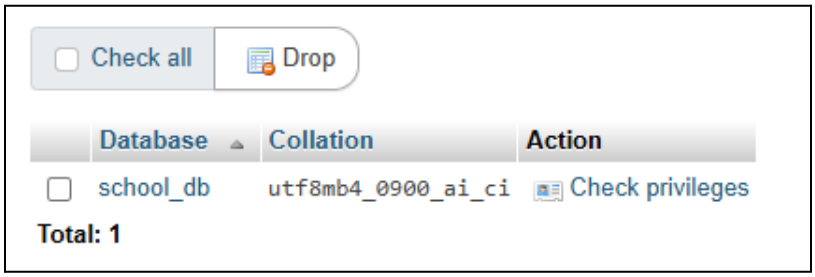
Module 4 – Introduction to DBMS

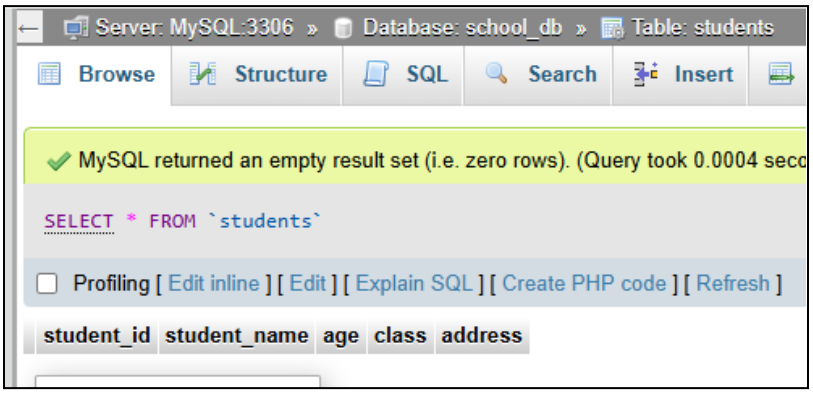
1. **Introduction to SQL**

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.

CREATE DATABASE school_db;

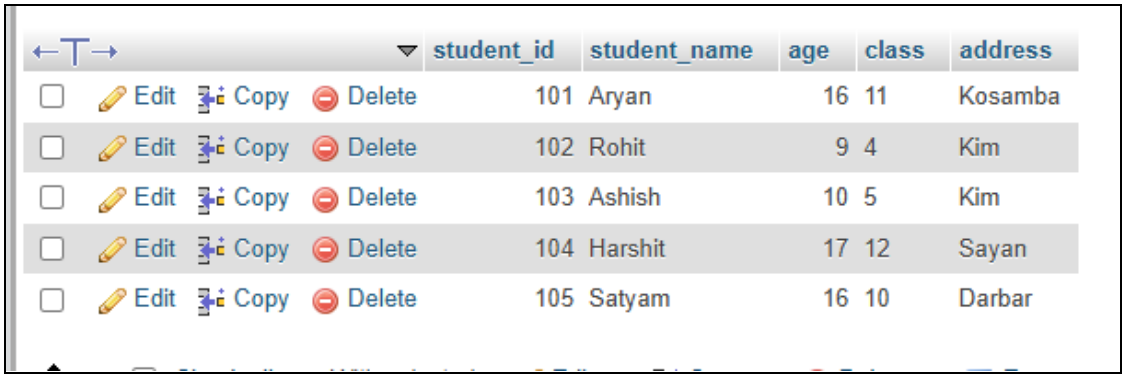



```
CREATE TABLE students (
   student_id int(50) PRIMARY KEY,
   student_name varchar(50) NOT NULL,
   age int(50) NOT NULL,
   class varchar(50) NOT NULL,
   address VARCHAR(100) NOT NULL
);
```



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

```
INSERT INTO students (student_id, student_name, age, class, address) VALUES (101, "Aryan", 16, 11 , "Kosamba" );
INSERT INTO students (student_id, student_name, age, class, address) VALUES (102, "Rohit", 9, 4 , "Kim" );
INSERT INTO students (student_id, student_name, age, class, address) VALUES (103, "Ashish", 10, 5 , "Kim" );
INSERT INTO students (student_id, student_name, age, class, address) VALUES (104, "Harshit", 17, 12 , "Sayan" );
INSERT INTO students (student_id, student_name, age, class, address) VALUES (105, "Satyam", 16, 10 , "Darbar" );
```

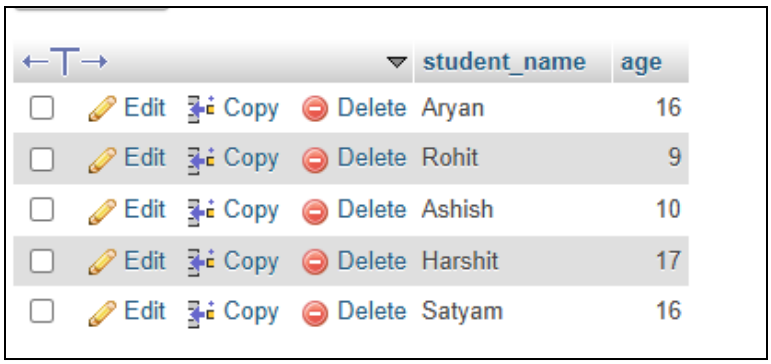


	student_id	student_name	age	class	address
Edit Copy Delete	101	Aryan	16	11	Kosamba
Edit Copy Delete	102	Rohit	9	4	Kim
Edit Copy Delete	103	Ashish	10	5	Kim
Edit Copy Delete	104	Harshit	17	12	Sayan
Edit Copy Delete	105	Satyam	16	10	Darbar

2. SQL Syntax

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

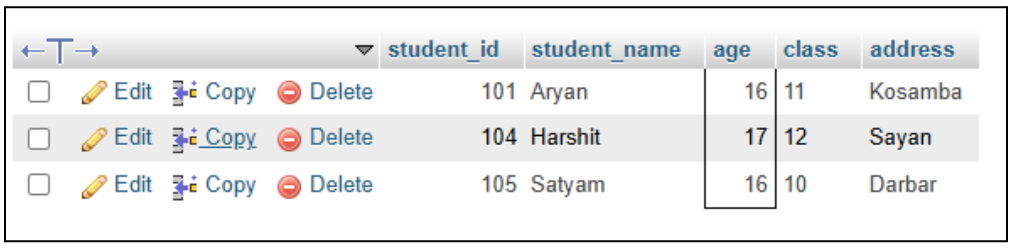
SELECT student_name, age FROM students;



	student_name	age
Edit Copy Delete	Aryan	16
Edit Copy Delete	Rohit	9
Edit Copy Delete	Ashish	10
Edit Copy Delete	Harshit	17
Edit Copy Delete	Satyam	16

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

SELECT * FROM students WHERE age>10;

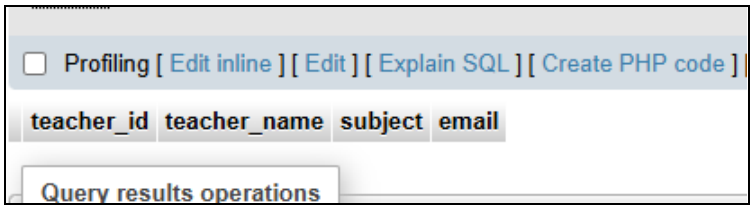


	student_id	student_name	age	class	address
Edit Copy Delete	101	Aryan	16	11	Kosamba
Edit Copy Delete	104	Harshit	17	12	Sayan
Edit Copy Delete	105	Satyam	16	10	Darbar

3. SQL Constraints

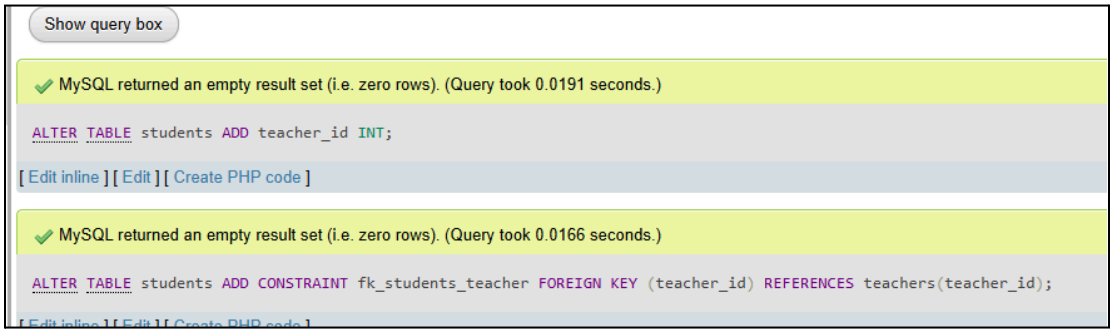
Lab 1: Create a table teachers with the following columns: teacher_id (Primary Key), teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

```
CREATE TABLE teachers(
   teacher_id int(50) PRIMARY KEY,
   teacher_name varchar(50) NOT NULL,
   subject varchar(50) NOT NULL,
   email varchar(50) UNIQUE
);
```



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

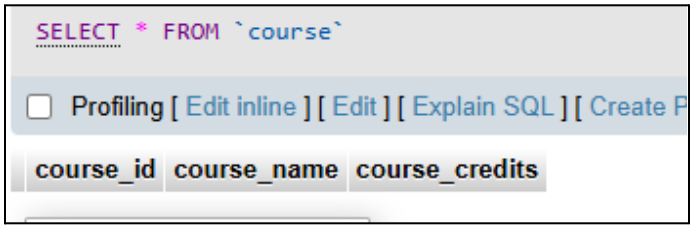
```
ALTER TABLE students
ADD teacher_id INT;

ALTER TABLE students
ADD CONSTRAINT fk_students_teacher
FOREIGN KEY (teacher_id)
REFERENCES teachers(teacher_id);
```



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

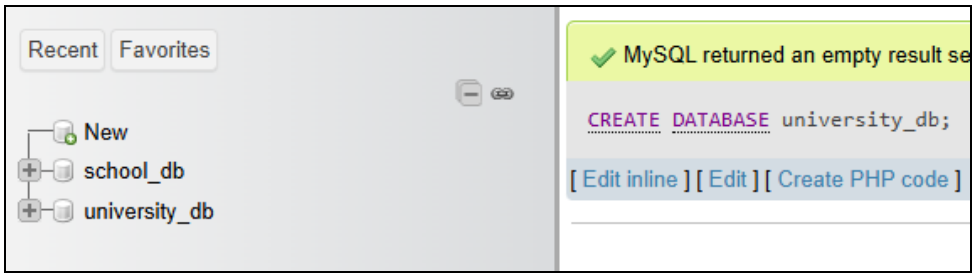
Lab 1: Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

```
CREATE TABLE course(
   course_id int(50) PRIMARY KEY,
   course_name VARCHAR(50) NOT NULL,
   course_credits VARCHAR(50) NOT NULL
);
```



Lab 2: Use the CREATE command to create a database university_db

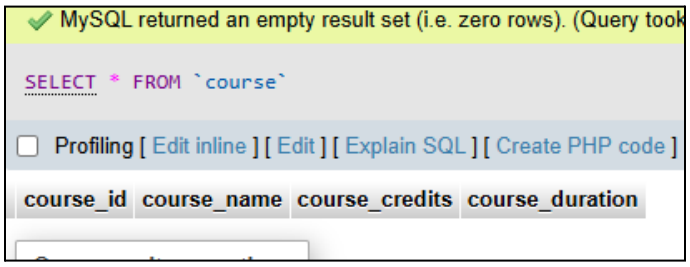
```
CREATE DATABASE university_db;
```



5. **ALTER Command**

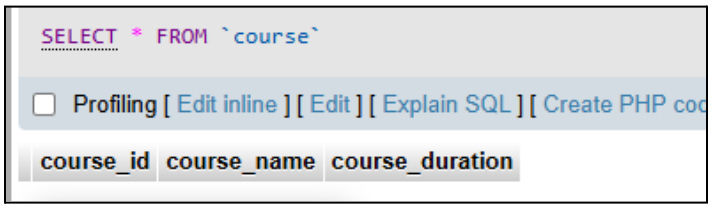
Lab 1: Modify the courses table by adding a column course_duration using the ALTER command.

ALTER TABLE course
ADD course_duration int(50) NOT NULL;



Lab 2: Drop the course_credits column from the courses table.

ALTER TABLE course
DROP COLUMN course_credits;



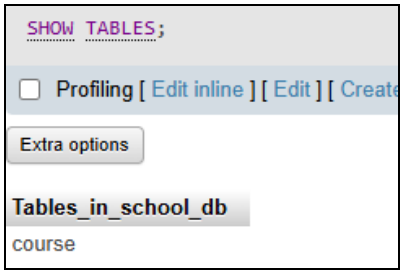
6. **DROP command**

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.

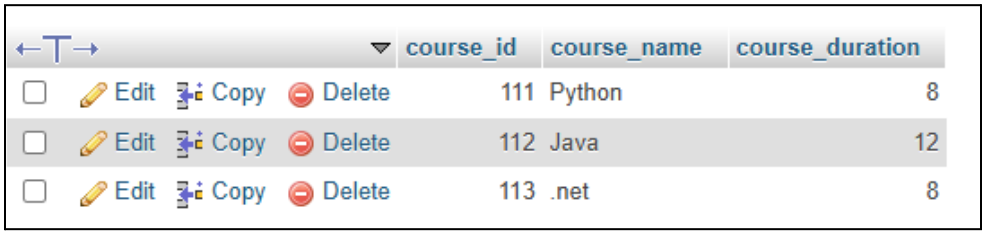
DROP TABLE students;



7. Data Manipulation Language (DML)

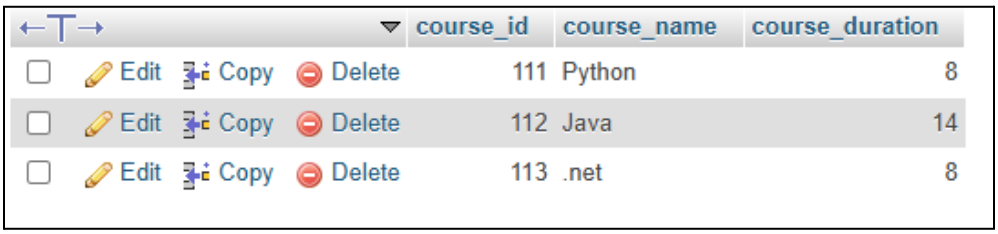
Lab 1: Insert three records into the courses table using the INSERT command.

INSERT INTO course (course_id, course_name, course_duration) VALUES (111, "Python", 8);
INSERT INTO course (course_id, course_name, course_duration) VALUES (112, "Java", 12);
INSERT INTO course (course_id, course_name, course_duration) VALUES (113, ".net", 8);



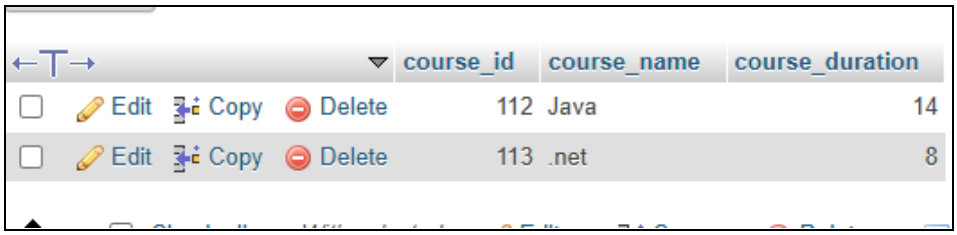
Lab 2: Update the course duration of a specific course using the UPDATE command.

UPDATE course
set course_duration = 14
WHERE course_name = "java";



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

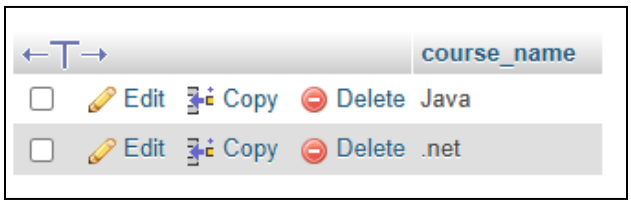
DELETE FROM course WHERE course_id = 111;



8. Data Query Language (DQL)

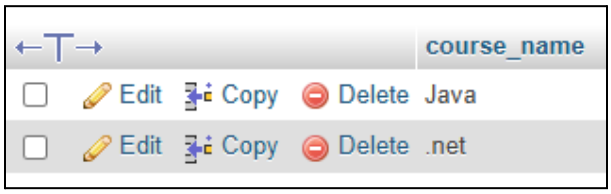
Lab 1: Retrieve all courses from the courses table using the SELECT statement.

SELECT course_name from course;



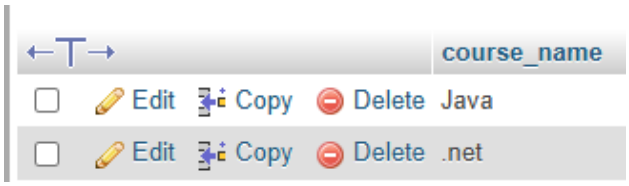
Lab 2: Sort the courses based on course_duration in descending order using ORDER BY.

SELECT course_name FROM course ORDER BY course_duration DESC;



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

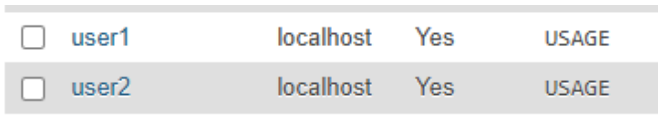
SELECT course_name FROM course LIMIT 2;



9. Data Control Language (DCL)

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

CREATE USER 'user1'@'localhost' IDENTIFIED BY 'user1';
CREATE USER 'user2'@'localhost' IDENTIFIED BY 'user2';



GRANT SELECT ON school_db.course to 'user1'@'localhost';
FLUSH PRIVILEGES;

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

```
REVOKE INSERT ON school_db.courses FROM 'user1'@'localhost';

GRANT INSERT ON school_db.courses TO 'user2'@'localhost';
FLUSH PRIVILEGES;
```

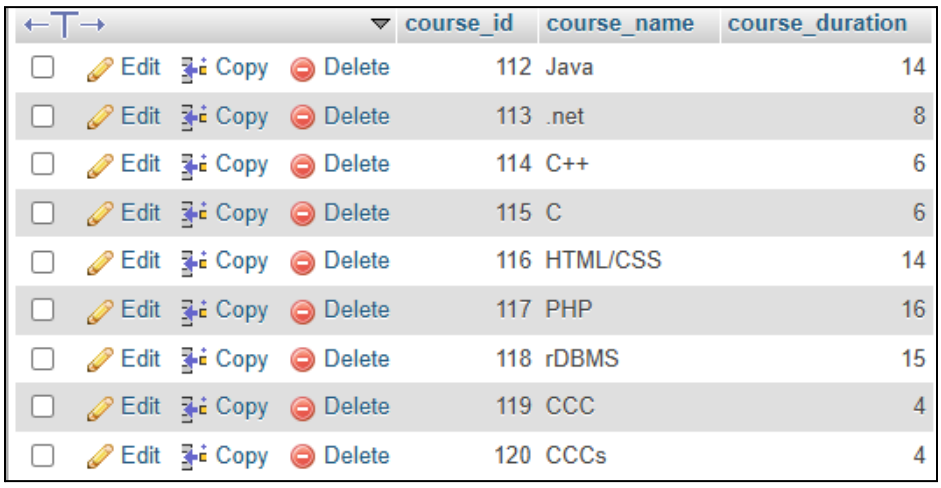
10. Transaction Control Language (TCL)

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

```
BEGIN;

INSERT INTO course (course_id, course_name, course_duration) VALUES (114, "C++", 6);
INSERT INTO course (course_id, course_name, course_duration) VALUES (115, "C", 6);
INSERT INTO course (course_id, course_name, course_duration) VALUES (116, "HTML/CSS", 14);

COMMIT;
```

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

```
BEGIN;

INSERT INTO course (course_id, course_name, course_duration) VALUES (121, "Cs", 6);

ROLLBACK;
```



	course_id	course_name	course_duration
Edit Copy Delete	112	Java	14
Edit Copy Delete	113	.net	8
Edit Copy Delete	114	C++	6
Edit Copy Delete	115	C	6
Edit Copy Delete	116	HTML/CSS	14
Edit Copy Delete	117	PHP	16
Edit Copy Delete	118	rDBMS	15
Edit Copy Delete	119	CCC	4
Edit Copy Delete	120	CCCs	4

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

```
START TRANSACTION;
INSERT INTO course (course_id, course_name, course_duration) VALUES (122, "Css", 6);
SAVEPOINT sp1;
```

11. 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 (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50),
    location VARCHAR(50)
);

CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    salary DECIMAL(10,2),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

```
SELECT * FROM departments INNER JOIN employees on departments.dept_id = employees.dept_id;
```

dept_id	dept_name	location	emp_id	emp_name	salary	dept_id
2	IT	Bangalore	101	Ashish	45000.00	2
1	HR	Mumbai	102	Rohit	38000.00	1
3	Finance	Delhi	103	Neha	52000.00	3
4	Sales	Pune	104	Priya	41000.00	4
2	IT	Bangalore	105	Amit	60000.00	2
7	Support	Kolkata	106	Kiran	36000.00	7
5	Marketing	Hyderabad	107	Sneha	48000.00	5

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

```
SELECT * FROM departments LEFT JOIN employees ON departments.dept_id = employees.dept_id;
```

dept_id	dept_name	location	emp_id	emp_name	salary	dept_id
1	HR	Mumbai	102	Rohit	38000.00	1
2	IT	Bangalore	101	Ashish	45000.00	2
2	IT	Bangalore	105	Amit	60000.00	2
3	Finance	Delhi	103	Neha	52000.00	3
4	Sales	Pune	104	Priya	41000.00	4
5	Marketing	Hyderabad	107	Sneha	48000.00	5
6	Operations	Chennai	*NULL*	*NULL*	*NULL*	*NULL*
7	Support	Kolkata	106	Kiran	36000.00	7

12. SQL Group By

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

SELECT dept_name , COUNT(emp_id) as Total_Employees FROM departments INNER JOIN employees on departments.dept_id = employees.dept_id GROUP BY departments.dept_name;

dept_name	Total_Employees
IT	2
HR	1
Finance	1
Sales	1
Support	1
Marketing	1

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

SELECT dept_name , AVG(salary) as Avg_salary FROM departments INNER JOIN employees on departments.dept_id = employees.dept_id GROUP BY departments.dept_name;

dept_name	Avg_salary
IT	52500.000000
HR	38000.000000
Finance	52000.000000
Sales	41000.000000
Support	36000.000000
Marketing	48000.000000

13. SQL Stored Procedure

Lab 1: Write a stored procedure to retrieve all employees from the employees table based on department.

```
DELIMITER $$
CREATE PROCEDURE GetEmployeesByDepartment(
   IN deptName VARCHAR(50)
)
BEGIN
   SELECT
      e.emp_id,
      e.emp_name,
      e.salary,
      d.dept_name,
      d.location
   FROM employees e
   INNER JOIN departments d
```

```
    ON e.dept_id = d.dept_id
  WHERE d.dept_name = deptName;
END $$

DELIMITER ;
```

```
CALL GetEmplByDept("IT");
```

emp_id	emp_name	salary	dept_name	location
101	Ashish	45000.00	IT	Bangalore
105	Amit	60000.00	IT	Bangalore

Lab 2: Write a stored procedure that accepts course_id as input and returns the course details.

```
DELIMITER $$

CREATE PROCEDURE get_course_details (
  IN p_course_id INT
)
BEGIN
  SELECT
    course_id,
    course_name,
    course_duration
  FROM courses
  WHERE course_id = p_course_id;
END $$

DELIMITER ;
```

```
CALL get_course_details(112);
```

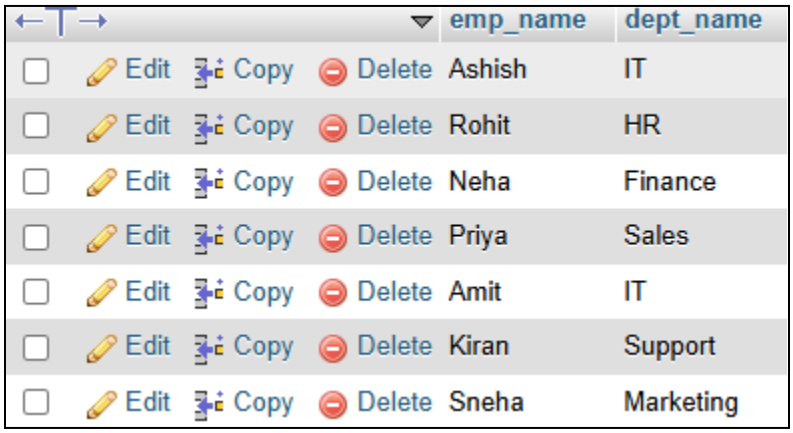
course_id	course_name	course_duration
112	Java	14

14. SQL View

Lab 1: Create a view to show all employees along with their department names.

CREATE VIEW Emp_with_dept AS SELECT emp_name, dept_name FROM departments INNER JOIN employees ON departments.dept_id = employees.dept_id;

SELECT * FROM emp_with_dept;

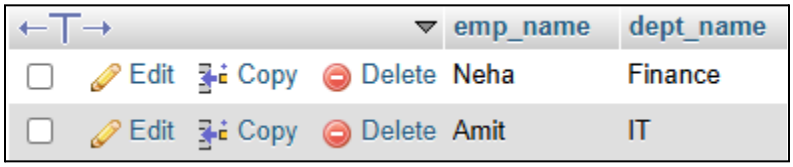


	emp_name	dept_name
Edit Copy Delete	Ashish	IT
Edit Copy Delete	Rohit	HR
Edit Copy Delete	Neha	Finance
Edit Copy Delete	Priya	Sales
Edit Copy Delete	Amit	IT
Edit Copy Delete	Kiran	Support
Edit Copy Delete	Sneha	Marketing

Lab 2: Modify the view to exclude employees whose salaries are below $50,000.

CREATE OR REPLACE VIEW emp_with_dept AS SELECT emp_name, dept_name FROM departments INNER JOIN employees ON departments.dept_id = employees.dept_id WHERE salary>50000;

SELECT * FROM emp_with_dept;



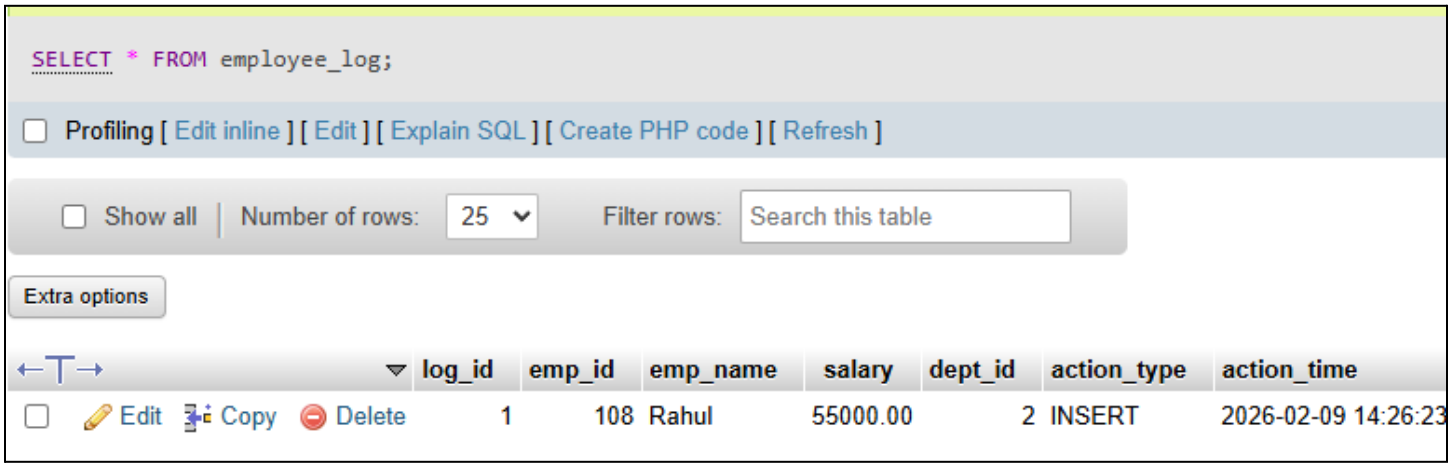
	emp_name	dept_name
Edit Copy Delete	Neha	Finance
Edit Copy Delete	Amit	IT

15. SQL Triggers

Lab 1: Create a trigger to automatically log changes to the employees table when a new employee is added.

```
DELIMITER $$

CREATE TRIGGER trg_after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
   INSERT INTO employee_log (
      emp_id,
      emp_name,
      salary,
      dept_id,
```

```
        action_type,
        action_time
    )
    VALUES (
        NEW.emp_id,
        NEW.emp_name,
        NEW.salary,
        NEW.dept_id,
        'INSERT',
        NOW()
    );
END $$

DELIMITER ;

INSERT INTO employees (emp_id, emp_name, salary, dept_id)
VALUES (108, 'Rahul', 55000, 2);

SELECT * FROM employee_log;
```



Lab 2: Create a trigger to update the last_modified timestamp whenever an employee record is updated.

```
ALTER TABLE employees
ADD last_modified DATETIME;


DELIMITER $$

CREATE TRIGGER trg_before_employee_update
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
```

```
    SET NEW.last_modified = NOW();
END $$

DELIMITER ;


UPDATE employees
SET salary = salary + 5000
WHERE emp_id = 101;



SELECT emp_id, emp_name, salary, last_modified
FROM employees
WHERE emp_id = 101;
```

16. Introduction to PL/SQL

Lab 1: Write a PL/SQL block to print the total number of employees from the employees table.

Lab 2: Create a PL/SQL block that calculates the total sales from an orders table.