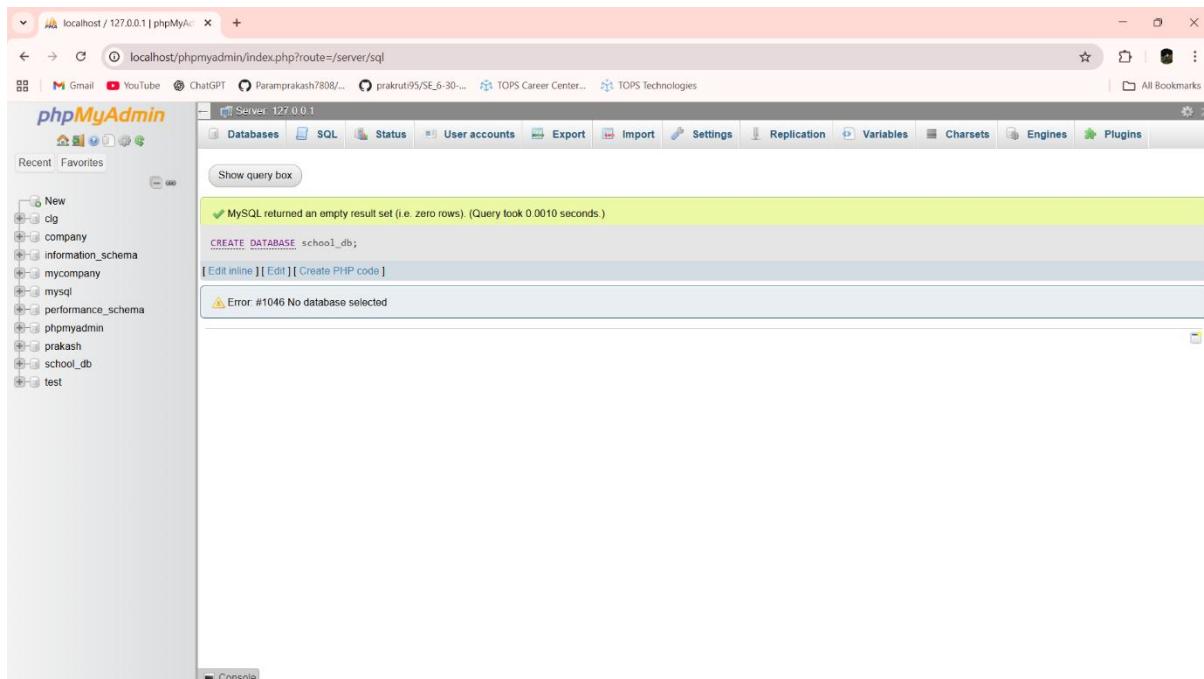


**Name:** Paramprakash Makwana

## **Module 4 – Introduction to DBMS(Lab)**

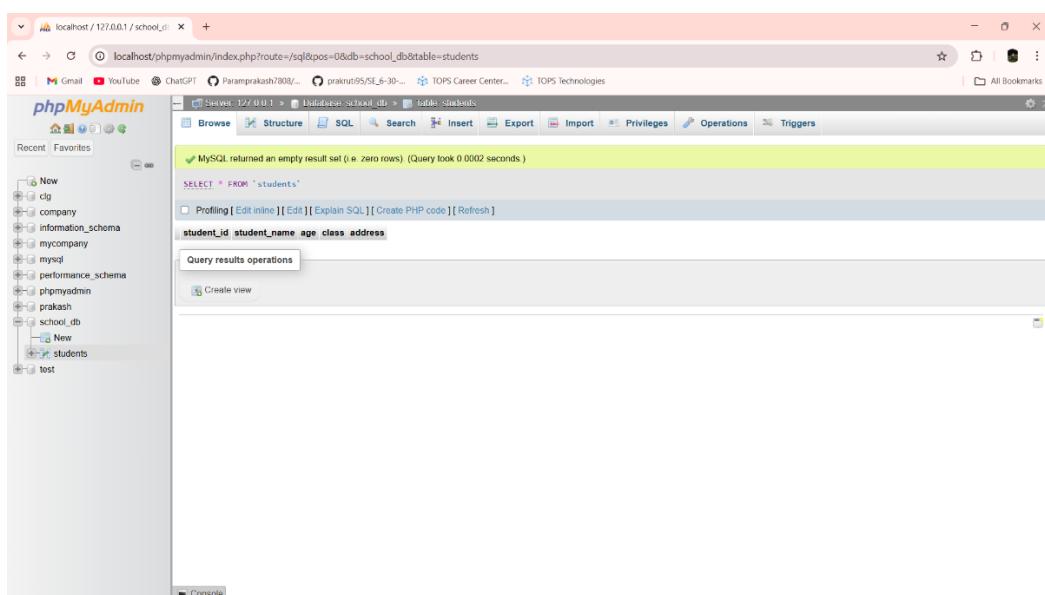
**Lab 1.1:** Create a new database named school\_db and a table called students with the following columns: student\_id, student\_name, age, class, and address.

**Ans:-** CREATE DATABASE school\_db



The screenshot shows the phpMyAdmin interface on a Windows operating system. The title bar says "localhost / 127.0.0.1 | phpMyAdmin". The left sidebar lists databases: New, clg, company, information\_schema, mycompany, mysql, performance\_schema, phpmyadmin, prakash, school\_db, and test. The main area shows a query result: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0010 seconds)". Below it is a code editor with the SQL command: "CREATE DATABASE school\_db;". A warning message "Error #1046 No database selected" is displayed at the bottom.

**CREATE TABLE** students (student\_id int PRIMARY KEY  
AUTO\_INCREMENT,student\_name varchar(30),age int,class varchar(30),address  
varchar(50))



The screenshot shows the phpMyAdmin interface on a Windows operating system. The title bar says "localhost / 127.0.0.1 / school\_db | phpMyAdmin". The left sidebar shows the database "school\_db" expanded, revealing "New" and "students". The main area shows a query result: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds)". Below it is a code editor with the SQL command: "SELECT \* FROM `students`;". The interface shows the "Structure" tab is active, and the table "students" is listed with columns: student\_id, student\_name, age, class, address.

Lab 1.2: Insert five records into the students table and retrieve all records using the SELECT statement.

Ans:- `INSERT INTO students (student_id,student_name,age,class,address) VALUES (101,'Prakash',21,'8TD2','Morbi Road Bedi Village Rajkot')`

`INSERT INTO students (student_id,student_name,age,class,address) VALUES (102,'Sahil',22,'1TD2','Morbi Road Bedi Village Rajkot')`

`INSERT INTO students (student_id,student_name,age,class,address) VALUES (103,'Ajay',21,'2TD2','Morbi Road Bedi Village Rajkot')`

`INSERT INTO students (student_id,student_name,age,class,address) VALUES (104,'Nik',22,'3TD2','Morbi Road Bedi Village Rajkot')`

`INSERT INTO students (student_id,student_name,age,class,address) VALUES (105,'Mayur',20,'4TD2','Morbi Road Bedi Village Rajkot')`

`SELECT * FROM students`

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows the database structure with databases like school\_db, prakash, and test, and tables like students.
- Top navigation:** Shows the URL `localhost/phpmyadmin/index.php?route=/table/sql&db=school_db&table=students`.
- Toolbar:** Includes buttons for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, and Triggers.
- Query results:** A green bar indicates "Showing rows 0 - 4 (5 total, Query took 0 0003 seconds.)". The query `SELECT * FROM students;` is shown. Below it are buttons for Profiling, Edit inline, Explain SQL, Create PHP code, and Refresh.
- Data grid:** Displays the following data:

	student_id	student_name	age	class	address
<input type="checkbox"/>	101	Prakash	21	8TD2	Morbi Road Bedi Village Rajkot
<input type="checkbox"/>	102	Sahil	22	1TD2	Morbi Road Bedi Village Rajkot
<input type="checkbox"/>	103	Ajay	21	2TD2	Morbi Road Bedi Village Rajkot
<input type="checkbox"/>	104	Nik	22	3TD2	Morbi Road Bedi Village Rajkot
<input type="checkbox"/>	105	Mayur	20	4TD2	Morbi Road Bedi Village Rajkot

- Bottom buttons:** Show all, Number of rows: 25, Filter rows, Search this table, Sort by key, None.
- Query results operations:** Buttons for Print, Copy to clipboard, Export, Display chart, Create view, and Console.

Lab 2.1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.

Ans:- SELECT student\_name,age FROM students

The screenshot shows the phpMyAdmin interface for a database named 'school\_db'. The 'students' table is selected. The results of the query 'SELECT student\_name, age FROM students;' are displayed in a table with columns 'student\_name' and 'age'. The data shows five rows: Prakash (21), Sahil (22), Ajay (21), Nik (22), and Mayur (20). Below the table are buttons for 'Edit', 'Copy', 'Delete', and 'Export'. The status bar at the top indicates 'Showing rows 0 - 4 (total, Query took 0.0003 seconds.)'.

student_name	age
Prakash	21
Sahil	22
Ajay	21
Nik	22
Mayur	20

Lab 2.2: Write SQL queries to retrieve all students whose age is greater than 10.

Ans:- SELECT \* FROM students WHERE age > 10

The screenshot shows the phpMyAdmin interface for a database named 'school\_db'. The 'students' table is selected. The results of the query 'SELECT \* FROM students WHERE age > 10;' are displayed in a table with columns 'student\_id', 'student\_name', 'age', 'class', and 'address'. The data shows five rows: 101 Prakash (21, 8TD2, Morbi Road Bedi Village Rajkot), 102 Sahil (22, 1TD2, Morbi Road Bedi Village Rajkot), 103 Ajay (21, 2TD2, Morbi Road Bedi Village Rajkot), 104 Nik (22, 3TD2, Morbi Road Bedi Village Rajkot), and 105 Mayur (20, 4TD2, Morbi Road Bedi Village Rajkot). Below the table are buttons for 'Edit', 'Copy', 'Delete', and 'Export'. The status bar at the top indicates 'Showing rows 0 - 4 (total, Query took 0.0003 seconds.)'.

student_id	student_name	age	class	address
101	Prakash	21	8TD2	Morbi Road Bedi Village Rajkot
102	Sahil	22	1TD2	Morbi Road Bedi Village Rajkot
103	Ajay	21	2TD2	Morbi Road Bedi Village Rajkot
104	Nik	22	3TD2	Morbi Road Bedi Village Rajkot
105	Mayur	20	4TD2	Morbi Road Bedi Village Rajkot

Lab 3.1: 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(30),subject varchar(30),email varchar(30))

The screenshot shows the phpMyAdmin interface for a database named 'school\_db'. In the left sidebar, under the 'Tables' section, there is a new table named 'teachers'. The main area displays the SQL query for creating the table:

```
CREATE TABLE `teachers` (
  `teacher_id` int NOT NULL AUTO_INCREMENT,
  `teacher_name` varchar(30) NOT NULL,
  `subject` varchar(30) NOT NULL,
  `email` varchar(30) UNIQUE,
  PRIMARY KEY (`teacher_id`)
);
```

The status bar at the top indicates: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

Lab 3.2: Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.

Ans:- ALTER TABLE students ADD CONSTRAINT fk\_teacher FOREIGN KEY (teacher\_id)  
REFERENCES teachers(teacher\_id)

The screenshot shows the phpMyAdmin interface for the same 'school\_db' database. In the left sidebar, under the 'Tables' section, there is a table named 'students'. The main area displays the SQL query for adding a foreign key constraint:

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

The status bar at the top indicates: Showing rows 0 - 4 (5 total), Query took 0.0002 seconds.

The 'students' table has the following data:

student_id	student_name	age	class	address	teacher_id
101	Prakash	21	8TD2	Morbi Road Bedi Village Rajkot	NULL
102	Sahil	22	1TD2	Morbi Road Bedi Village Rajkot	NULL
103	Ajay	21	2TD2	Morbi Road Bedi Village Rajkot	NULL
104	Nik	22	3TD2	Morbi Road Bedi Village Rajkot	NULL
105	Mayur	20	4TD2	Morbi Road Bedi Village Rajkot	NULL

Lab 4.1: 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(30), course_credits int)
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'school\_db'. On the left, the database structure is visible with tables like 'New', 'clg', 'company', etc. In the center, the 'courses' table is being created. The SQL query entered is:

```
SELECT * FROM `courses`
```

The results pane shows a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)". Below the query, there are tabs for 'Structure', 'Insert', 'Export', 'Privileges', 'Operations', and 'Triggers'. A 'Query results operations' section is also present.

Lab 4.2: Use the CREATE command to create a database university\_db.

Ans:-

```
CREATE DATABASE university_db
```

The screenshot shows the phpMyAdmin interface for a MySQL server. On the left, the database structure is visible with databases like 'New', 'clg', 'company', etc. In the center, a query is being run to create a new database:

```
CREATE DATABASE university_db;
```

The results pane shows a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0008 seconds.)". Below the query, there are tabs for 'Databases', 'SQL', 'Status', 'User accounts', 'Export', 'Import', 'Settings', 'Replication', 'Variables', 'Charsets', 'Engines', and 'Plugins'. A message at the bottom states: "Error: #1046 No database selected".

**Lab 5.1:** Modify the courses table by adding a column course\_duration using the ALTER command.

**Ans:-** ALTER TABLE courses ADD course\_duration varchar(30)

The screenshot shows the phpMyAdmin interface for a database named 'school\_db'. On the left, the database structure is visible with tables like 'courses', 'students', and 'teachers'. The main panel displays the 'courses' table structure. The 'Structure' tab is selected, showing columns: 'course\_id', 'course\_name', 'course\_credits', and 'course\_duration'. A note at the top states: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)'. Below the table structure, there is a SQL query editor with the following code:

```
SELECT * FROM `courses`
```

**Lab 5.2:** Drop the course\_credits column from the courses table.

**Ans:-** ALTER TABLE courses DROP COLUMN course\_credits

This screenshot is identical to the one above, showing the 'courses' table structure in phpMyAdmin. The 'Structure' tab is selected, and the columns listed are 'course\_id', 'course\_name', and 'course\_duration'. The note at the top still reads: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)'. This indicates that the 'course\_credits' column has been successfully dropped.

**Lab 6.1: Drop the teachers table from the school\_db database.**

**Ans:-** `DROP TABLE teachers`

The screenshot shows the phpMyAdmin interface. The left sidebar lists databases: New, clg, company, information\_schema, Views, mycompany, mysql, performance\_schema, phpmysql, prakash, school\_db, New, courses, students, test, and university\_db. The main area is titled 'Database: school\_db' and shows the 'SQL' tab selected. A green message bar at the top says 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0149 seconds.)'. Below it, the SQL query 'DROP TABLE teachers;' is entered. At the bottom of the main area, there are buttons for 'Edit inline', 'Edit', and 'Create PHP code'.

**Lab 6.2: Drop the students table from the school\_db database and verify that the table has been removed.**

**Ans:-** `DROP TABLE students`

The screenshot shows the phpMyAdmin interface. The left sidebar lists the same databases as the previous screenshot. The main area is titled 'Database: school\_db' and shows the 'SQL' tab selected. A green message bar at the top says 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0157 seconds.)'. Below it, the SQL query 'DROP TABLE students;' is entered. At the bottom of the main area, there are buttons for 'Edit inline', 'Edit', and 'Create PHP code'.

Lab 7.1: Insert three records into the courses table using the INSERT command.

Ans:- `INSERT INTO courses (course_id, course_name, course_duration) VALUES (101, 'Python', '6 Months')`

`INSERT INTO courses (course_id, course_name, course_duration) VALUES (102, 'Flutter', '6 Months')`

`INSERT INTO courses (course_id, course_name, course_duration) VALUES (103, 'Java', '6 Months')`

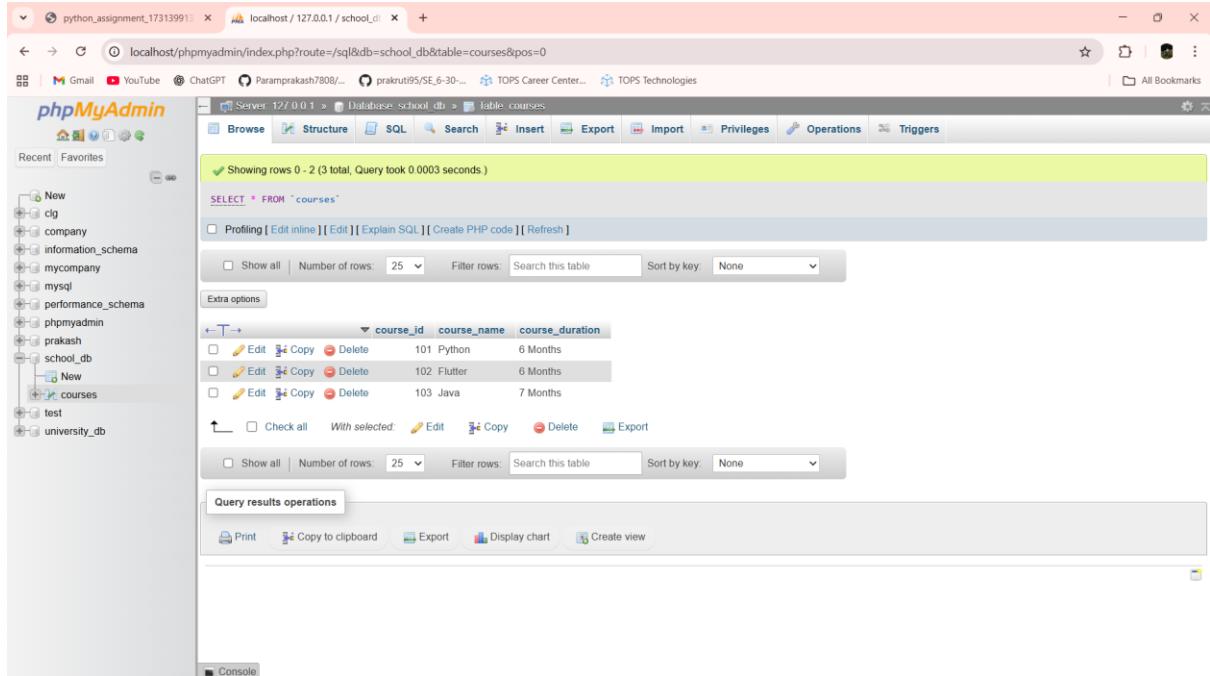
The screenshot shows the phpMyAdmin interface for a MySQL database named 'school\_db'. The left sidebar lists databases and tables, including 'courses'. The main area displays the 'courses' table with three rows:

	course_id	course_name	course_duration
<input type="checkbox"/>	101	Python	6 Months
<input type="checkbox"/>	102	Flutter	6 Months
<input type="checkbox"/>	103	Java	6 Months

Below the table, there are buttons for 'Edit', 'Copy', 'Delete', and 'Export'. The status bar at the bottom indicates 'Showing rows 0 - 2 (3 total, Query took 0.0002 seconds)'.

Lab 7.2: Update the course duration of a specific course using the UPDATE command.

Ans:- UPDATE courses SET course\_duration = '7 Months' WHERE course\_id = 103

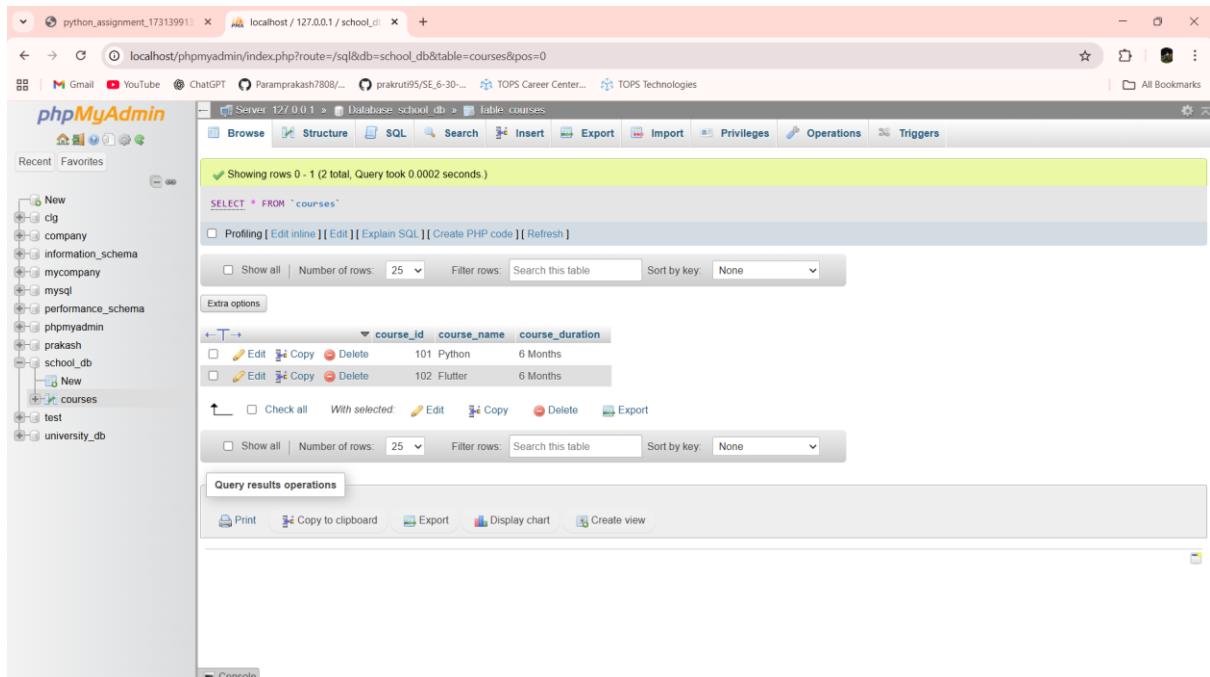


The screenshot shows the phpMyAdmin interface for the 'courses' table. The table has columns: course\_id, course\_name, and course\_duration. There are three rows:

course_id	course_name	course_duration
101	Python	6 Months
102	Flutter	6 Months
103	Java	7 Months

Lab 7.3: Delete a course with a specific course\_id from the courses table using the DELETE command.

Ans:- DELETE FROM courses WHERE course\_id = 103



The screenshot shows the phpMyAdmin interface for the 'courses' table after the deletion of course\_id 103. Now there are only two rows:

course_id	course_name	course_duration
101	Python	6 Months
102	Flutter	6 Months

Lab 8.1: Retrieve all courses from the courses table using the SELECT statement.

Ans:- `SELECT * FROM courses`

The screenshot shows the phpMyAdmin interface for a database named 'school\_db'. The left sidebar lists various databases and tables, including 'courses'. The main area displays the results of the query `SELECT * FROM courses;`. The results table has columns: course\_id, course\_name, and course\_duration. It contains two rows: one for Python (6 Months) and one for Flutter (6 Months).

course_id	course_name	course_duration
101	Python	6 Months
102	Flutter	6 Months

Lab 8.2: Sort the courses based on course\_duration in descending order using ORDER BY.

Ans:- `SELECT * FROM courses ORDER BY course_duration DESC`

The screenshot shows the phpMyAdmin interface for a database named 'school\_db'. The left sidebar lists various databases and tables, including 'courses'. The main area displays the results of the query `SELECT * FROM courses ORDER BY course_duration DESC;`. The results table has columns: course\_id, course\_name, and course\_duration. It contains two rows: one for Flutter (7 Months) and one for Python (6 Months).

course_id	course_name	course_duration
102	Flutter	7 Months
101	Python	6 Months

Lab 8.3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

Ans:- `SELECT * FROM courses LIMIT 2`

The screenshot shows the phpMyAdmin interface for a MySQL database named 'school\_db'. The left sidebar lists various databases and tables. The main area displays the results of a SQL query: `SELECT * FROM courses LIMIT 2;`. The results table shows two rows of course data:

	course_id	course_name	course_duration
<input type="checkbox"/>	101	Python	6 Months
<input type="checkbox"/>	102	Flutter	7 Months

Below the table are standard MySQL operations buttons: Print, Copy to clipboard, Export, Display chart, and Create view.

Lab 9.1: Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

Ans:- CREATE USER 'user101'@'localhost' IDENTIFIED BY 'password1';

CREATE USER 'user102'@'localhost' IDENTIFIED BY 'password2';

GRANT SELECT ON university\_db.courses TO 'user101'@'localhost';

The screenshot shows the phpMyAdmin interface for a MySQL database named 'university'. The left sidebar lists various databases and their structures. The main panel displays two separate SQL queries for granting privileges:

**Grant for user101@localhost:**

```
SHOW GRANTS FOR 'user101'@'localhost';
GRANT USAGE ON *.* TO 'user101'@'localhost' IDENTIFIED BY 'password1';
GRANT SELECT ON university_db.courses TO 'user101'@'localhost';
```

**Grant for user102@localhost:**

```
SHOW GRANTS FOR 'user102'@'localhost';
GRANT USAGE ON *.* TO 'user102'@'localhost' IDENTIFIED BY 'password2';
```

Both grants are successful, as indicated by the green status bars above each query.

Lab 9.2: Revoke the INSERT permission from user1 and give it to user2.

Ans:- REVOKE INSERT ON university\_db.courses FROM 'user101'@'localhost';

GRANT INSERT ON university\_db.courses TO 'user102'@'localhost';

The screenshot shows the phpMyAdmin interface for a MySQL database named 'university\_db'. The left sidebar lists various databases and their structures. The main query window displays two sets of SQL grants:

**Grants for user103@localhost**

```
SHOW GRANTS FOR 'user103'@'localhost';
GRANT USAGE ON *.* TO 'user103'@'localhost' IDENTIFIED BY '';
GRANT SELECT ON 'university_db'.'courses' TO 'user103'@'localhost';
```

**Grants for user104@localhost**

```
SHOW GRANTS FOR 'user104'@'localhost';
GRANT USAGE ON *.* TO 'user104'@'localhost' IDENTIFIED BY '';
GRANT INSERT ON 'university_db'.'courses' TO 'user104'@'localhost';
```

Below each grant set, there is a 'Query results operations' panel with options like Print, Copy to clipboard, and Create view. A note at the top of the second panel states: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." The browser address bar shows the URL as localhost/phpmyadmin/index.php?route=/table/sql&db=university\_db&table=courses.

Lab 10.1: Insert a few rows into the courses table and use COMMIT to save the changes.

Ans:- START TRANSACTION;

INSERT INTO courses VALUES

(1, 'Web Development', 6, 12000),

(2, 'Data Science', 8, 15000);

COMMIT;

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The 'courses' table is selected. The table structure is as follows:

	course_id	course_name	course_duration	course_fee
	1	Web Development	6	12000.00
	2	Data Science	8	15000.00

Below the table, there are buttons for Edit, Copy, Delete, and Export. The 'Query results operations' section includes Print, Copy to clipboard, Export, Display chart, and Create view options.

Lab 10.2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

Ans:- START TRANSACTION;

INSERT INTO courses VALUES

(3, 'Python', 4, 8000),

(4, 'Java', 5, 10000);

ROLLBACK;

course_id	course_name	course_duration	course_fee
1	Web Development	6	12000.00
2	Data Science	8	15000.00

Lab 10.3: 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\_fee = 13000

WHERE course\_id = 1;

ROLLBACK TO before\_update;

COMMIT;

The screenshot shows the phpMyAdmin interface for a MySQL database named 'university\_db'. The left sidebar lists various databases and their structures. The main panel displays the 'courses' table with two rows of data. The table has columns: course\_id, course\_name, course\_duration, and course\_fee. The first row has values: 1, Web Development, 6, 12000.00. The second row has values: 2, Data Science, 8, 15000.00. Below the table, there are buttons for Edit, Copy, Delete, and Export. At the bottom of the page, there is a 'Query results operations' section with options like Print, Copy to clipboard, Export, Display chart, and Create view.

course_id	course_name	course_duration	course_fee
1	Web Development	6	12000.00
2	Data Science	8	15000.00

Lab 11.1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

Ans:- CREATE TABLE departments (dept\_id INT PRIMARY KEY,dept\_name VARCHAR(30))

The screenshot shows the phpMyAdmin interface for a database named 'university\_db'. On the left, the database structure is visible, including the 'departments' table under the 'university\_db' schema. The main panel displays a query results operations window with a green status bar indicating 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)'. Below this, a SQL query is shown: 'SELECT \* FROM `departments`'. The results section is empty.

INSERT INTO departments VALUES(1, 'HR'),(2, 'IT'),(3, 'Finance')

The screenshot shows the phpMyAdmin interface for the same database. The 'departments' table now contains three rows of data: (1, 'HR'), (2, 'IT'), and (3, 'Finance'). The data is displayed in a grid with columns 'dept\_id' and 'dept\_name'. Each row has edit, copy, and delete options. The main panel shows a green status bar indicating 'Showing rows 0 - 2 (total, Query took 0.0002 seconds.)'. The SQL query 'SELECT \* FROM `departments`' is still present in the query results operations window.

CREATE TABLE employees (emp\_id INT PRIMARY KEY, emp\_name VARCHAR(50), dept\_id INT, FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id))

The screenshot shows the phpMyAdmin interface for a database named 'university\_db'. On the left, the database structure is visible with tables like 'New', 'clg', 'company', etc., and the newly created 'employees' table. The main panel shows the SQL query: 'CREATE TABLE employees (emp\_id INT PRIMARY KEY, emp\_name VARCHAR(50), dept\_id INT, FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id))'. A message at the top indicates 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)'. Below the message, the table structure is defined with columns: emp\_id, emp\_name, and dept\_id.

INSERT INTO employees VALUES(101, 'Sahil', 1),(102, 'Ajay', 2),(103, 'Mayur', 2)

The screenshot shows the phpMyAdmin interface for the same database. The 'employees' table now contains three rows of data: (101, 'Sahil', 1), (102, 'Ajay', 2), and (103, 'Mayur', 2). The table structure is identical to the one in the previous screenshot. The data is listed in a grid format with columns: emp\_id, emp\_name, and dept\_id. Each row has options for Edit, Copy, Delete, and Export.

```
SELECT employees.emp_name, departments.dept_name FROM employees INNER JOIN departments ON employees.dept_id = departments.dept_id
```

The screenshot shows the phpMyAdmin interface for a database named 'university\_db'. The left sidebar lists various databases and tables. The main area displays the results of a query: 'SELECT employees.emp\_name, departments.dept\_name FROM employees INNER JOIN departments ON employees.dept\_id = departments.dept\_id;'. The results show three rows:

emp_name	dept_name
Sahil	HR
Ajay	IT
Mayur	IT

Lab 11.2: Use a LEFT JOIN to show all departments, even those without employees.

Ans:- SELECT departments.dept\_name, employees.emp\_name FROM departments LEFT JOIN employees ON departments.dept\_id = employees.dept\_id

The screenshot shows the phpMyAdmin interface for a database named 'university\_db'. The left sidebar lists various databases and tables. The main area displays the results of a query: 'SELECT departments.dept\_name, employees.emp\_name FROM departments LEFT JOIN employees ON departments.dept\_id = employees.dept\_id;'. The results show four rows, including one for the 'Finance' department which has no employee assigned:

dept_name	emp_name
HR	Sahil
IT	Ajay
IT	Mayur
Finance	NULL

Lab 12.1: Group employees by department and count the number of employees in each department using GROUP BY.

Ans:- `SELECT dept_id, COUNT(*) AS total_employees FROM employees GROUP BY dept_id`

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The 'employees' table is selected. A query has been run: `SELECT dept_id, COUNT(*) AS total_employees FROM employees GROUP BY dept_id;`. The results are displayed in a table:

dept_id	total_employees
1	1
2	2

Lab 12.2: Use the AVG aggregate function to find the average salary of employees in each department.

Ans:- `SELECT dept_id, AVG(salary) AS average_salary FROM employees GROUP BY dept_id`

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The 'employees' table is selected. A query has been run: `SELECT dept_id, AVG(salary) AS average_salary FROM employees GROUP BY dept_id;`. The results are displayed in a table:

dept_id	average_salary
1	15000.0000
2	15000.0000

Lab 13.1: Write a stored procedure to retrieve all employees from the employees table based on department.

Ans:- DELIMITER \$\$

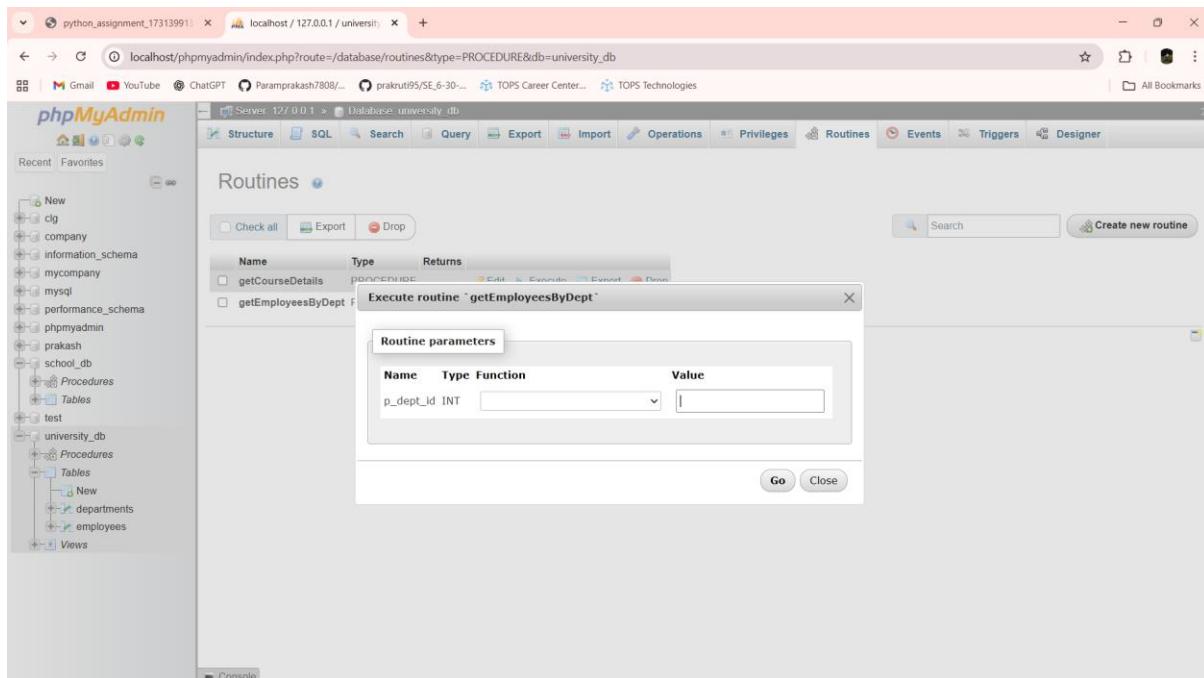
```
CREATE PROCEDURE getEmployeesByDept(IN p_dept_id INT)
```

```
BEGIN
```

```
    SELECT * FROM employees WHERE dept_id = p_dept_id;
```

```
END $$
```

```
DELIMITER ;
```



```
CALL getEmployeesByDept(2);
```

The screenshot shows the phpMyAdmin interface on a web browser. The left sidebar lists databases and tables. The main area displays the results of a SQL query:

```
Showing rows 0 - 1 (total: 2) (Query took 0.0155 seconds.)  
CALL getEmployeesByDept(2);
```

The results table has columns: emp\_id, emp\_name, dept\_id, salary. The data is:

emp_id	emp_name	dept_id	salary
102	Ajay	2	18000
103	Mayur	2	12000

Below the table are buttons for Print, Copy to clipboard, and Create view.

Lab 13.2: Write a stored procedure that accepts course\_id as input and returns the course details.

Ans:- DELIMITER \$\$

```
CREATE PROCEDURE getCourseDetails(IN p_course_id INT)
```

```
BEGIN
```

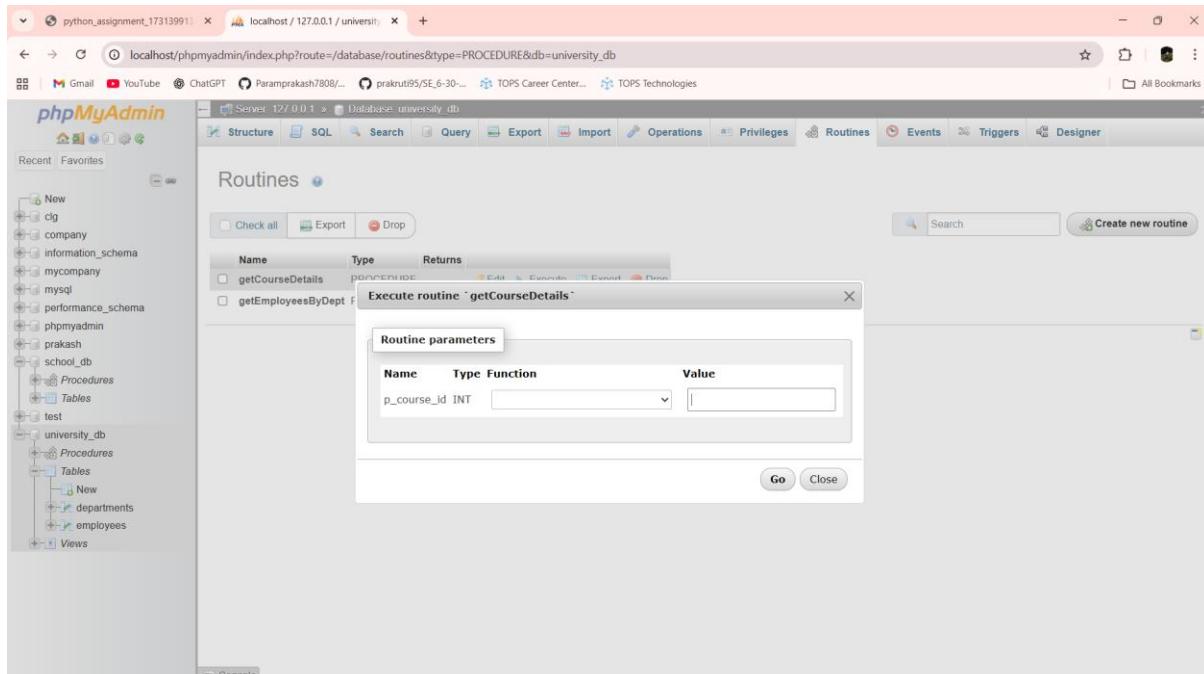
```
    SELECT *
```

```
    FROM courses
```

```
    WHERE course_id = p_course_id;
```

```
END $$
```

```
DELIMITER ;
```



CALL getCourseDetails(101)

The screenshot shows the phpMyAdmin interface for a MySQL database named 'school\_db'. The left sidebar lists various databases and their tables. The main area displays the results of a query:

```
Showing rows 0 - 0 (1 total, Query took 0 0003 seconds.)  
CALL getCourseDetails(101);
```

The results table shows one row:

course_id	course_name	course_duration
101	Python	6 Months

Below the table are 'Query results operations' buttons: Print, Copy to clipboard, and Create view.

Lab 14.1: Create a view to show all employees along with their department names.

Ans:- CREATE VIEW emp\_dept\_view AS

SELECT

```
e.emp_id,  
e.emp_name,  
d.dept_name,  
e.salary  
FROM employees e  
INNER JOIN departments d  
ON e.dept_id = d.dept_id;
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'university\_db'. The left sidebar displays the database structure with tables like 'employees', 'departments', and 'emp\_dept\_view'. The main area is titled 'View emp\_dept\_view' and shows the results of the query: 'SELECT \* FROM `emp\_dept\_view`'. The results table contains three rows of data:

	emp_id	emp_name	dept_name	salary
<input type="checkbox"/>	101	Sahil	HR	20000
<input type="checkbox"/>	102	Ajay	IT	18000
<input type="checkbox"/>	103	Mayur	IT	12000

Lab 14.2: Modify the view to exclude employees whose salaries are below \$50,000.

Ans:- CREATE OR REPLACE VIEW emp\_dept\_view AS

SELECT

```
e.emp_id,  
e.emp_name,  
d.dept_name,  
e.salary  
FROM employees e  
INNER JOIN departments d  
ON e.dept_id = d.dept_id  
WHERE e.salary >= 50000;
```

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The left sidebar lists databases like 'New', 'clg', 'company', etc., and tables such as 'employees' and 'emp\_dept\_view'. The main panel displays the 'emp\_dept\_view' table with the following data:

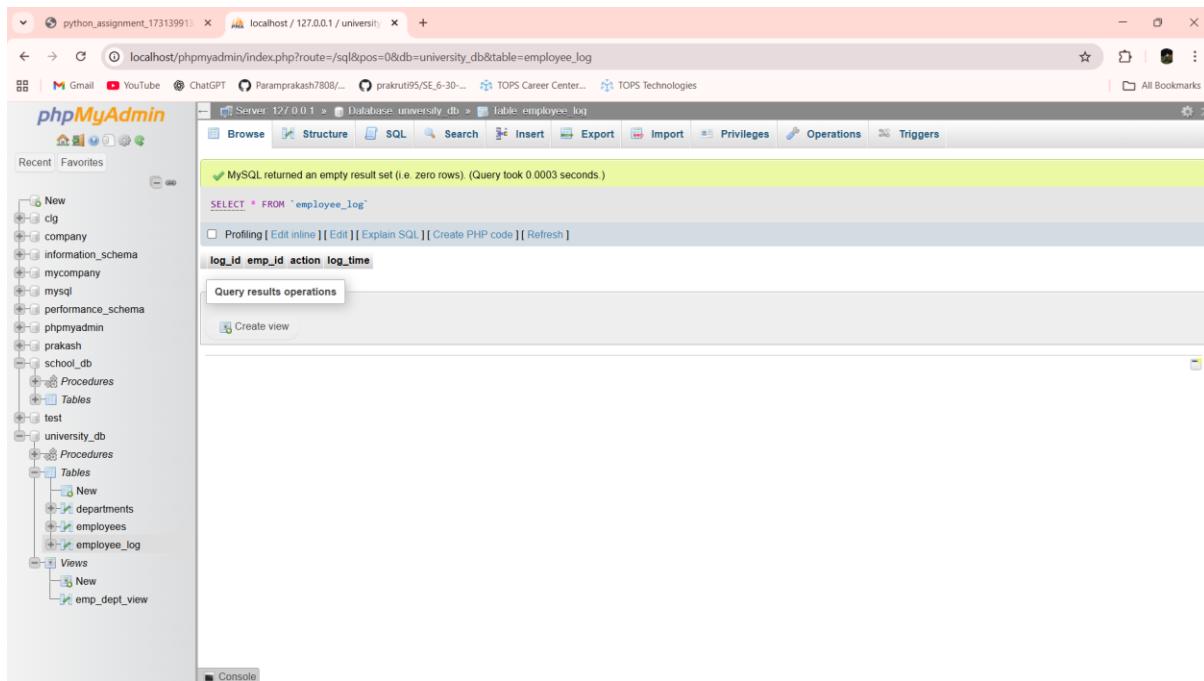
emp_id	emp_name	dept_name	salary
102	Ajay	IT	65000

Below the table, there are buttons for Edit, Copy, Delete, and Export. The status bar at the bottom indicates "Showing rows 0 - 0 (total, Query took 0 0003 seconds)".

Lab 15.1: Create a trigger to automatically log changes to the employees table when a new employee is added.

Ans:-

```
CREATE TABLE employee_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    emp_id INT,
    action VARCHAR(50),
    log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```



**DELIMITER \$\$**

**CREATE TRIGGER after\_employee\_insert**

**AFTER INSERT ON employees**

**FOR EACH ROW**

**BEGIN**

**INSERT INTO employee\_log (emp\_id, action)**

**VALUES (NEW.emp\_id, 'INSERT');**

**END \$\$**

**DELIMITER ;**

The screenshot shows the phpMyAdmin interface for the 'employee\_log' table in the 'university\_db'. The table has four columns: log\_id, emp\_id, action, and log\_time. The first row has log\_id 100, emp\_id 101, action 'NULL', and log\_time '2025-12-19 16:18:40'. The second row has log\_id 1001, emp\_id 101, action 'NULL', and log\_time '2025-12-19 16:18:27'. There are buttons for Edit, Copy, Delete, and Export.

log_id	emp_id	action	log_time
100	101	NULL	2025-12-19 16:18:40
1001	101	NULL	2025-12-19 16:18:27

Lab 15.2: Create a trigger to update the last\_modified timestamp whenever an employee record is updated.

Ans:- DELIMITER \$\$

```
CREATE TRIGGER before_employee_update
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
    SET NEW.last_modified = CURRENT_TIMESTAMP;
END $$

DELIMITER ;
```

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The 'employees' table is selected. The table structure is as follows:

emp_id	emp_name	dept_id	salary	last_modified
101	Sahl	1	20000	2025-12-19 16:19:36
102	Ajay	2	65000	2025-12-19 16:19:36
103	Mayur	2	12000	2025-12-19 16:19:36
110	Prakash	3	100000	2025-12-19 16:23:40

Lab 16.1: Write a PL/SQL block to print the total number of employees from the employees table.

Ans:-

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The 'employees' table is selected. The table structure includes columns: emp\_id, emp\_name, dept\_id, salary, and last\_modified. The data grid displays four rows of employee information:

emp_id	emp_name	dept_id	salary	last_modified
101	Sahil	1	20000	2025-12-19 16:19:36
102	Ajay	2	65000	2025-12-19 16:19:36
103	Mayur	2	12000	2025-12-19 16:19:36
110	Prakash	3	100000	2025-12-19 16:23:40

DELIMITER \$\$

```
CREATE PROCEDURE total_employees_count()
```

```
BEGIN
```

```
    DECLARE total_employees INT;
```

```
    SELECT COUNT(*) INTO total_employees
```

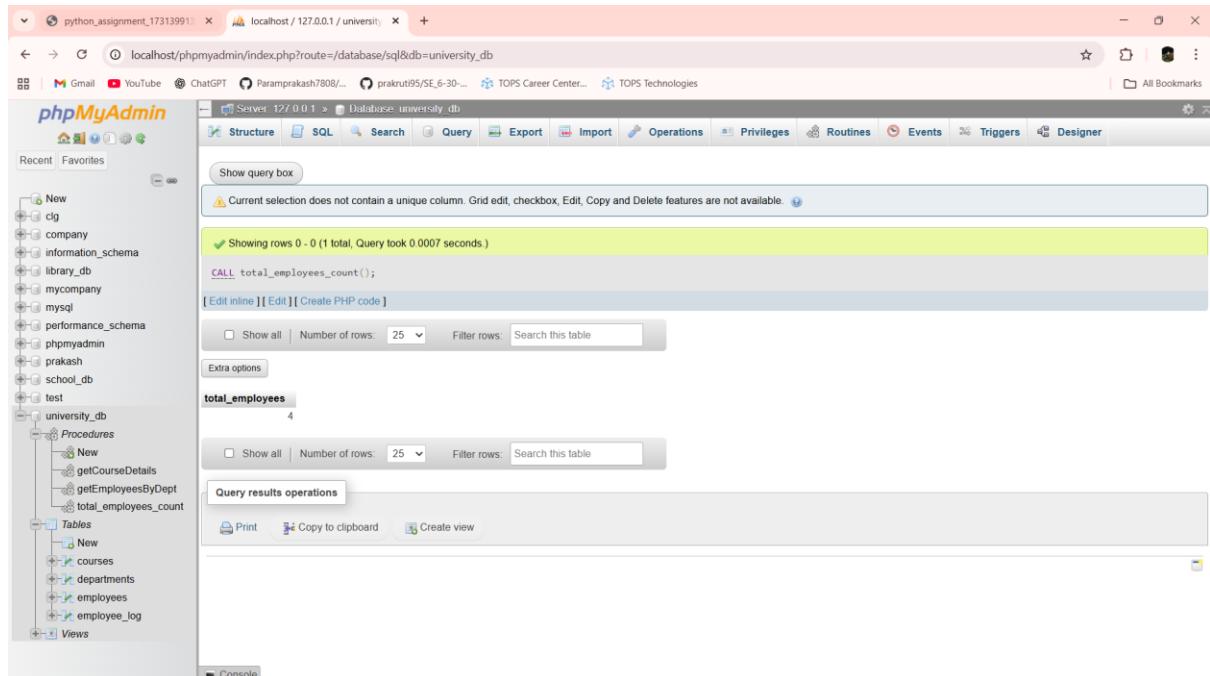
```
    FROM employees;
```

```
    SELECT total_employees AS total_employees;
```

```
END $$
```

DELIMITER ;

```
CALL total_employees_count();
```



## Lab 16.2: Create a PL/SQL block that calculates the total sales from an orders table.

Ans:-

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The left sidebar lists various databases and their tables. The 'orders' table under the 'university\_db' is selected. The main area displays the table structure and data. The table has columns: order\_id, customer\_name, order\_amount, and order\_date. There are three rows of data:

	order_id	customer_name	order_amount	order_date
<input type="checkbox"/>	1	Ajay	5000.00	2024-01-10
<input type="checkbox"/>	2	Sahil	7500.00	2024-01-12
<input type="checkbox"/>	3	Mayur	6200.00	2024-01-15

Below the table, there are buttons for Edit, Copy, Delete, and Export. The 'Query results operations' section includes Print, Copy to clipboard, Export, Display chart, and Create view.

```
DELIMITER $$
```

```
CREATE PROCEDURE total_sales_amount()
BEGIN
    DECLARE total_sales DECIMAL(10,2);
    SELECT SUM(order_amount) INTO total_sales
    FROM orders;
    SELECT total_sales AS total_sales;
END $$
```

```
DELIMITER ;
```

```
CALL total_sales_amount();
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'university\_db'. The left sidebar displays the database structure with tables like 'New', 'clg', 'company', etc., and a 'Procedures' section containing 'total\_sales\_amount'. The main query results pane shows the output of the executed stored procedure:

```
Showing rows 0 - 0 (1 total, Query took 0.0003 seconds)

CALL total_sales_amount();
+-----+
| total_sales |
+-----+
| 18700.00 |
+-----+
```

Below the results, there are 'Query results operations' buttons for Print, Copy to clipboard, and Create view.

Lab 17.1: Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

Ans:- DELIMITER \$\$

```
CREATE PROCEDURE check_department(IN emp_id INT)
```

```
BEGIN
```

```
    DECLARE dept VARCHAR(50);
```

```
    SELECT department INTO dept
```

```
    FROM employees
```

```
    WHERE employee_id = emp_id;
```

```
    IF dept = 'HR' THEN
```

```
        SELECT 'Employee belongs to HR department' AS result;
```

```
    ELSE
```

```
        SELECT 'Employee belongs to another department' AS result;
```

```
    END IF;
```

```
END $$
```

```
DELIMITER ;
```

```
CALL check_department(1);
```

The screenshot shows the phpMyAdmin interface for a database named 'university'. The left sidebar displays the database structure with 'Procedures' expanded, showing 'check\_department' as one of the procedures. The main query results pane shows the output of the executed stored procedure:

```
CALL check_department(1);
+-----+
| result |
+-----+
| Employee belongs to HR department |
+-----+
```

The results are displayed in a table with a single column labeled 'result' containing the text 'Employee belongs to HR department'.

Lab 17.2: Use a FOR LOOP to iterate through employee records and display their names.

Ans:-

The screenshot shows the phpMyAdmin interface for the 'employees' table in the 'university\_db'. The table has three columns: 'employee\_id', 'employee\_name', and 'department'. The data is as follows:

employee_id	employee_name	department
1	Ajay	HR
2	Mayur	Finance
3	Sahil	IT
4	Nik	HR
5	Prakash	Marketing

DELIMITER \$\$

```
CREATE PROCEDURE show_employee_names()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE emp_name VARCHAR(100);
    DECLARE emp_cursor CURSOR FOR
        SELECT employee_name FROM employees;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN emp_cursor;
    read_loop: LOOP
        FETCH emp_cursor INTO emp_name;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;
```

```

SELECT emp_name AS employee_name;

END LOOP;

CLOSE emp_cursor;

END $$

DELIMITER ;

```

CALL show\_employee\_names();

The screenshot shows the phpMyAdmin interface for the 'university' database. In the left sidebar, under 'Procedures', there is a list of stored procedures including 'check\_department', 'getCourseDetails', 'getEmployeesByDept', 'show\_employee\_name', 'total\_employees\_count', and 'total\_sales\_amount'. The 'show\_employee\_name' procedure is selected. In the main pane, the SQL query 'CALL `show\_employee\_names`();' has been executed successfully, and the results are displayed in a table:

employee_name
Ajay
Mayur
Sahil
Nik
Prakash

Lab 18.1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

Ans:- DELIMITER \$\$

```
CREATE PROCEDURE show_employee_details1()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_id INT;
    DECLARE v_name VARCHAR(100);
    DECLARE v_dept VARCHAR(50);
    DECLARE emp_cursor CURSOR FOR
        SELECT employee_id, employee_name, department FROM employees;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN emp_cursor;
    read_loop: LOOP
        FETCH emp_cursor INTO v_id, v_name, v_dept;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;
        SELECT v_id AS employee_id, v_name AS employee_name,
               v_dept AS department;
    END LOOP;
    CLOSE emp_cursor;
END $$
```

DELIMITER ;

```
CALL show_employee_details();
```

python\_assignment\_17313991 x localhost / 127.0.0.1 / university\_db +

localhost/phpmyadmin/index.php?route=/database/routines&type=PROCEDURE&db=university\_db

Gmail YouTube ChatGPT Paramprakash7809... prakruti95/SE\_6-30... TOPS Career Center... TOPS Technologies All Bookmarks

phpMyAdmin

Recent Favorites

library\_db mycompany mysql performance\_schema phpmyadmin prakash school\_db test university\_db Procedures New check\_department getCourseDetails getEmployeesByDept show\_employee\_detail show\_employee\_name total\_employees\_count Tables New courses departments employees employee\_log orders Views

Your SQL query has been executed successfully.  
1 row affected by the last statement inside the procedure.

CALL `show\_employee\_details`();

Execution results of routine 'show\_employee\_details'

employee_id	employee_name	department
1	Ajay	HR
2	Mayur	Finance
3	Sahil	IT
4	Nik	HR
5	Prakash	Marketing

Routines

Check all Export Drop

Name	Type	Returns
k_department	PROCEDURE	Edit Execute Export Drop

Search Create new routine

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. In the 'Procedures' section, the 'show\_employee\_details' procedure is selected. The 'Execution results of routine' tab displays the output of the procedure, which lists five employees with their IDs, names, and departments. Below the results, the 'Routines' section shows the definition of the 'k\_department' procedure. The 'Edit' and 'Execute' buttons are visible for this procedure.

Lab 18.2: Create a cursor to retrieve all courses and display them one by one.

Ans:- DELIMITER \$\$

```
CREATE PROCEDURE show_courses()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_id INT;
    DECLARE v_name VARCHAR(100);
    DECLARE v_duration VARCHAR(50);
    DECLARE course_cursor CURSOR FOR
        SELECT course_id, course_name, course_duration FROM courses;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN course_cursor;
    read_loop: LOOP
        FETCH course_cursor INTO v_id, v_name, v_duration;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;
        SELECT v_id AS course_id, v_name AS course_name, v_duration AS
course_duration;
    END LOOP;
    CLOSE course_cursor;
END $$

DELIMITER ;
CALL show_courses();
```

python\_assignment\_17313991 x localhost / 127.0.0.1 / university\_db +

localhost/phpmyadmin/index.php?route=/database/routines&type=PROCEDURE&db=university\_db

Gmail YouTube ChatGPT Paramprakash7809... prakruti95/SE\_6-30... TOPS Career Center... TOPS Technologies All Bookmarks

phpMyAdmin

Recent Favorites

mycompany mysql performance\_schema phpmyadmin prakash school\_db test university\_db

Procedures

- New
- check\_department
- getCourseDetails
- getEmployeesByDept
- show\_courses
- show\_employee\_details
- show\_employee\_name
- total\_employees\_count
- total\_sales\_amount

Tables

- New
- courses
- departments
- employees
- employee\_log
- orders

Views

Structure SQL Search Export Import Operations Privileges Routines Events Triggers Designer

Your SQL query has been executed successfully.  
1 row affected by the last statement inside the procedure.

CALL `show\_courses`();

Execution results of routine 'show\_courses'

course_id	course_name	course_duration
1	Web Development	6
2	Data Science	8

Routines

Check all Export Drop

Name	Type	Returns
check_department	PROCEDURE	Edit Execute Export Drop
getCourseDetails	PROCEDURE	Edit Execute Export Drop
getEmployeesByDept	PROCEDURE	Edit Execute Export Drop
show_courses	PROCEDURE	Edit Execute Export Drop
show_employee_details	PROCEDURE	Edit Execute Export Drop
show_employee_details1	PROCEDURE	Edit Execute Export Drop
show_employee_names	PROCEDURE	Edit Execute Export Drop
Console: employees count	PROCEDURE	Edit Execute Export Drop

Search Create new routine

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. In the left sidebar, under 'Procedures', the 'show\_courses' procedure is selected. The main panel displays the execution results of this procedure, which returns two rows of course information: 'course\_id' 1 with 'course\_name' 'Web Development' and 'course\_duration' 6, and 'course\_id' 2 with 'course\_name' 'Data Science' and 'course\_duration' 8. Below this, the 'Routines' section lists several other stored procedures, each with edit, execute, export, and drop options. The 'show\_courses' procedure is also listed here.

Lab 19.1: Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

Ans:- START TRANSACTION;

```
INSERT INTO employees (employee_id, employee_name, department)
```

```
VALUES (1, 'Alice', 'HR');
```

```
SAVEPOINT sp1;
```

```
INSERT INTO employees (employee_id, employee_name, department)
```

```
VALUES (2, 'Bob', 'IT');
```

```
ROLLBACK TO SAVEPOINT sp1;
```

```
COMMIT;
```

The screenshot shows the phpMyAdmin interface for the 'employees' table in the 'university\_db' database. The table has three columns: 'employee\_id', 'employee\_name', and 'department'. One record is present: employee\_id 1, employee\_name Alice, and department HR. The interface includes a sidebar with database and schema navigation, and a top bar with browser tabs and bookmarks.

employee_id	employee_name	department
1	Alice	HR

Lab 19.2: Commit part of a transaction after using a savepoint and then rollback the remaining changes.

Ans:- START TRANSACTION;

```
INSERT INTO employees (employee_id, employee_name, department)
```

```
VALUES (3, 'Charlie', 'Finance');
```

```
SAVEPOINT sp2;
```

```
INSERT INTO employees (employee_id, employee_name, department)
```

```
VALUES (4, 'David', 'Marketing');
```

```
COMMIT;
```

```
START TRANSACTION;
```

```
INSERT INTO employees (employee_id, employee_name, department)
```

```
VALUES (5, 'Eve', 'Sales');
```

```
ROLLBACK;
```

The screenshot shows the phpMyAdmin interface for the 'university\_db' database. The 'employees' table is selected, displaying three rows:

employee_id	employee_name	department
1	Alice	HR
3	Charlie	Finance
4	David	Marketing

The left sidebar shows the database structure with various tables and procedures defined under the 'university\_db' schema.