Module 4 – Introduction to DBMS

Introduction to SQL

Theory Questions:

1. What is SQL, and why is it essential in database management?

Ans. SQL is a (structured query language).

SQL is a standard language for storing, manipulating and retrieving data in databases.

SQL allows you to access and manipulate the databases.

why is it essential in database management:

1. Data Organization and Storage.
2. Efficient Data Retrieval.
3. Data Integrity.
4. Security and Access Control.
5. Data Analysis and Reporting.

2. Explain the difference between DBMS and RDBMS.

Ans.

RDBMS	DBMS
The data stored is in table format.	The data stored is in the file format.
Data in the form of a table are linked together.	No connection between data.
Support distributed database.	No support for distributed databases.
Data is stored in a large amount.	The data stored is a small quantity.
RDBMS supports multiple users.	DBMS supports a single user.
The software and hardware requirements are higher.	The software and hardware requirements are low.
Example: Oracle, SQL Server.	Example: XML, Microsoft Access.

3. Describe the role of SQL in managing relational databases.

Ans.

the key roles SQL plays in relational database management:

SQL is crucial in managing relational databases by providing a standardized way to interact with data.

It is used for:

1. Data Definition (DDL - Data Definition Language):

   Creating, modifying, and deleting database structures such as tables, indexes, and schemas.

   Commands: CREATE, ALTER, DROP.

2. Data Manipulation (DML - Data Manipulation Language):

   Inserting, updating, and deleting data stored in tables.

   Commands: INSERT, UPDATE, DELETE.

3. Data Retrieval (DQL - Data Query Language):

   Fetching data from the database using filtering, sorting, and aggregation queries.

   Command: SELECT.

4. Data Control (DCL - Data Control Language):

   Managing user permissions and access control.

   Commands: GRANT, REVOKE.

5. Transaction Control (TCL - Transaction Control Language):

   Ensuring data integrity and consistency during transactions.

   Commands: COMMIT, ROLLBACK, SAVEPOINT.

4. What are the key features of SQL?

Ans. The key features of SQL are:

1. Data Querying

   Retrieve data using SELECT.

2. Data Manipulation

   Insert, update, and delete data (INSERT, UPDATE, DELETE).

3. Data Definition

   Create and modify database structures (CREATE, ALTER, DROP).

4. Data Control

   Manage user access (GRANT, REVOKE).

5. Transaction Management

   Ensure data consistency (COMMIT, ROLLBACK).

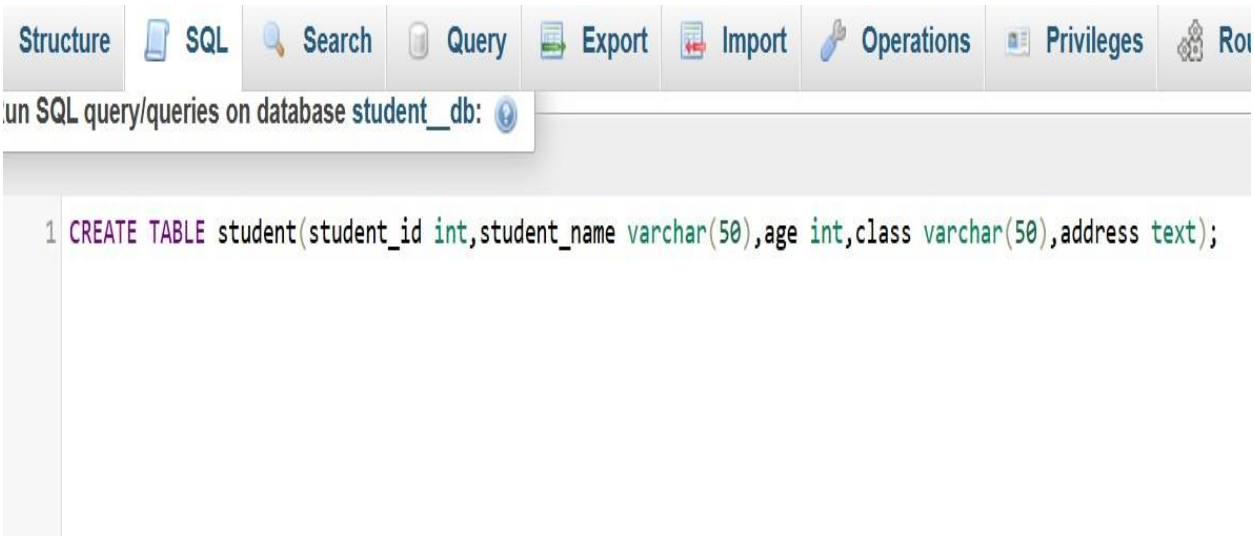
6. Integrity Constraints

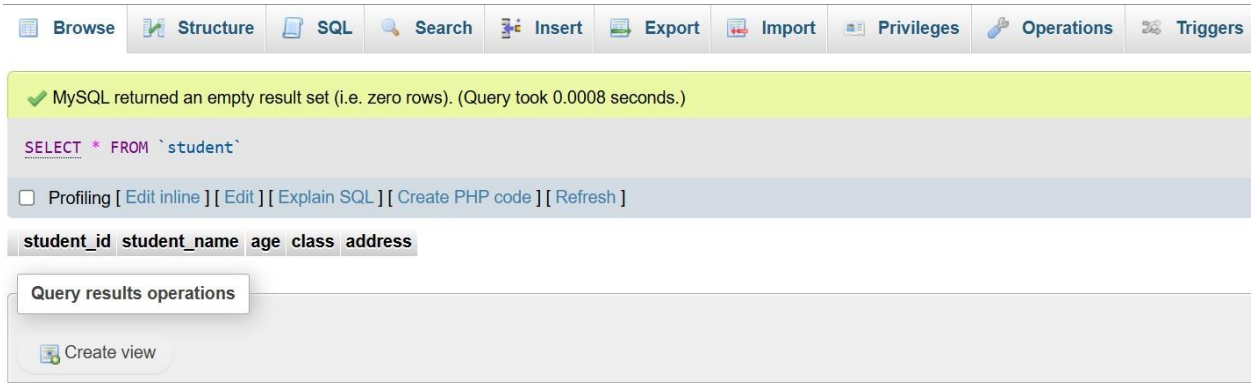
   Maintain data accuracy (PRIMARY KEY, FOREIGN KEY).

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.

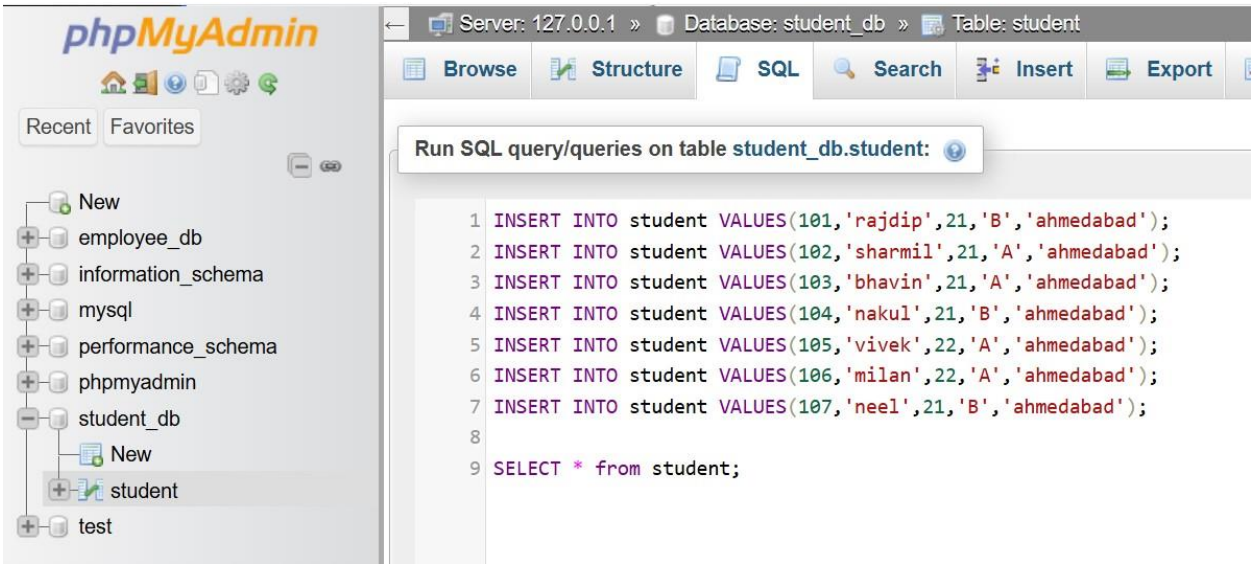
Ans.

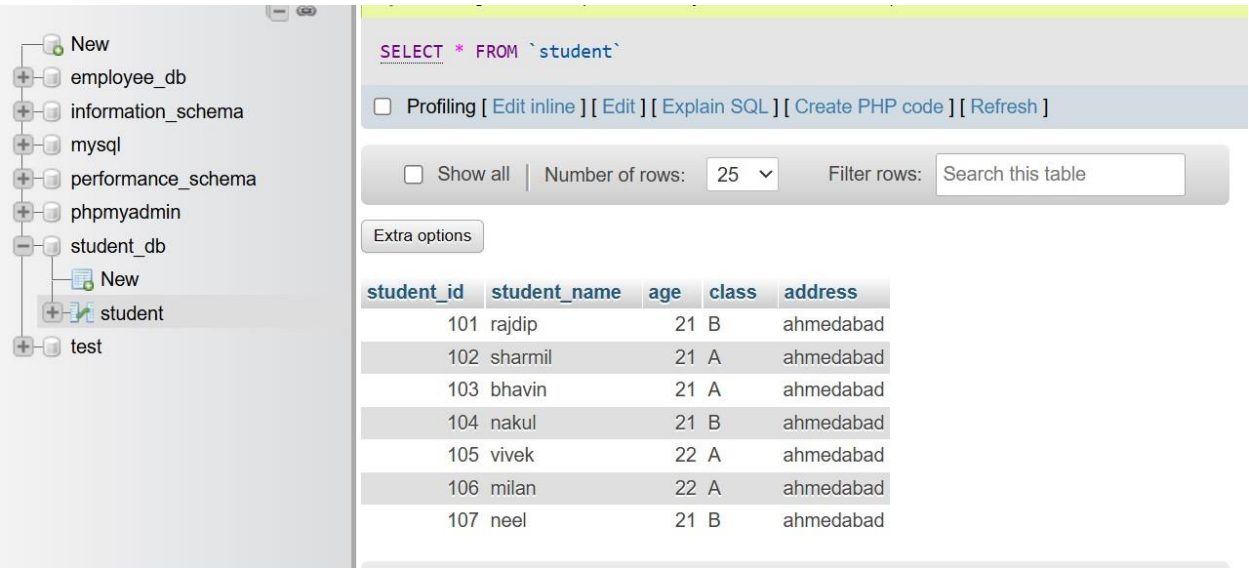




Lab 2: Insert five records into the student's table and retrieve all records using the SELECT statement.

Ans.





2. SQL Syntax

Theory Questions:

1. What are the basic components of SQL syntax?

Ans. The basic components of SQL syntax include:

1. **Statements**

   Commands used to interact with the database (example: SELECT, INSERT, UPDATE, DELETE).

2. **Clauses**

   Components of statements that define conditions (WHERE, HAVING).

3. **Expressions**

   Combinations of values, columns, and operators that return a result

   (price * quantity).

4. **Operators**

   Symbols used for comparisons and calculations (=, >, <, AND, OR).

5. **Keywords**

Reserved words that have special meaning in SQL (FROM, GROUP BY, ORDER BY).

6. **Functions**

   Built-in operations for calculations, aggregations, and formatting (COUNT(), SUM(), AVG()).

7. **Identifiers**

   Names of database objects like tables, columns, and indexes.

8. **Literals**

   Fixed values like numbers, text, and dates ('John', 100, '2024-01-01').

9. **Comments**

   Used for documentation within SQL code (-- Single-line comment or /* Multi-line comment */).

2. Write the general structure of an SQL SELECT statement.

Ans.

The SQL SELECT Statement:-

```
SELECT * FROM Employee;

SELECT Emp_No, Emp_Name from Employee;
```

The SQL SELECT DISTINCT Statement:

Inside a table, a column often contains many duplicate values; and

sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT * FROM Employee;

SELECT DISTINCT Emp_No, Emp_Name FROM Employee;
```

3. Explain the role of clauses in SQL statements.

Ans.

role of clauses in SQL statements:

The SQL WHERE Clause:

1. SELECT * FROM Employee WHERE Emp_No=101;

2. SELECT * FROM Employee WHERE Emp_Name='c.mam';

The SQL ORDER BY clause:

SELECT * FROM Employee ORDER BY Emp_Name;

SELECT * FROM Employee ORDER BY Emp_Name DESC;

GROUP BY Clause:

Groups those rows that have the same values into summary rows.

It collects data from multiple records and groups the results by one or more columns.

```
SELECT department, COUNT(name)
FROM employees
GROUP BY department;
```

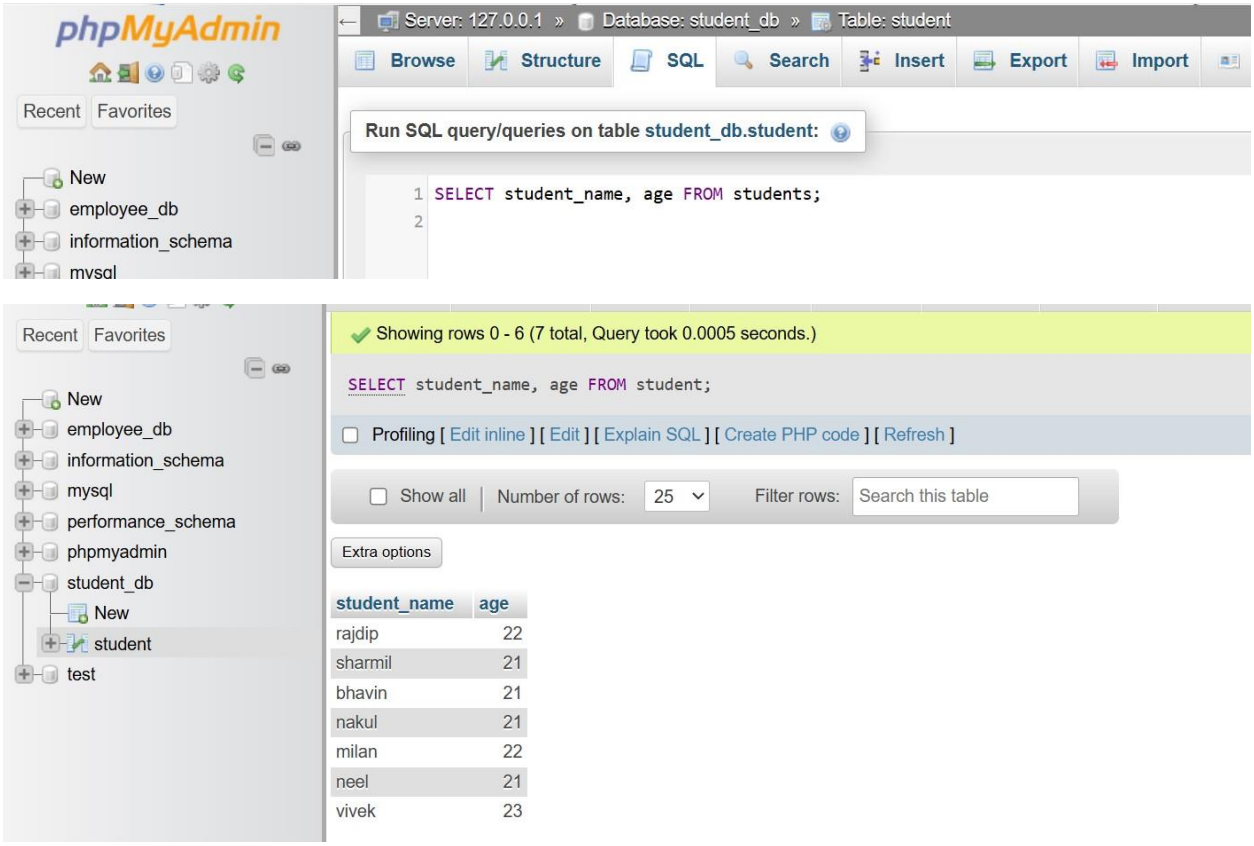
HAVING Clause:

```
SELECT COUNT(name), CITY
FROM employees
GROUP BY department
HAVING max(marks)> 5;
```

LAB EXERCISES:

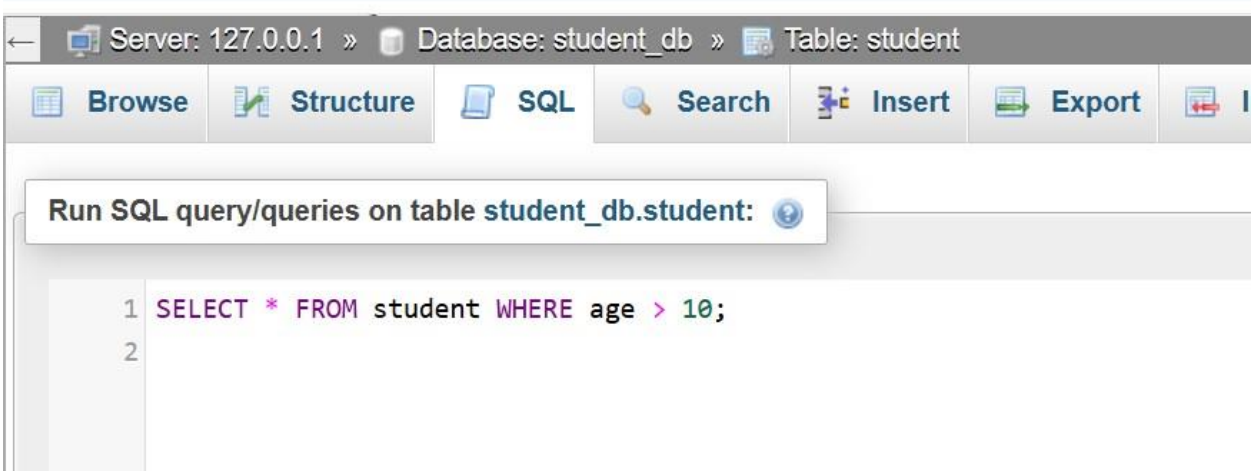
Lab 1: Write SQL queries to retrieve specific columns (student_name and age) from the student's table.

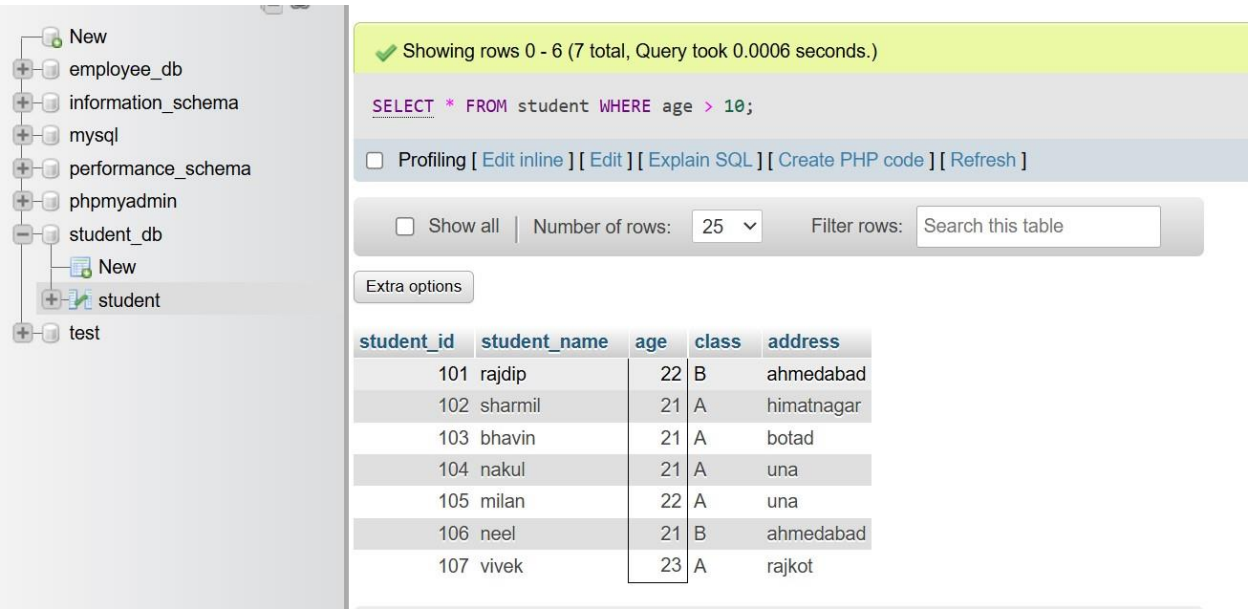
Ans.



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

Ans.





3. SQL Constraints

Theory Questions:

1. What are the constraints in SQL? List and explain the different types of constraints.

Ans. SQL constraints are used to specify rules for the data in a table.

Constraints can be column level or table level.

Column-level constraints apply to a column and table-level constraints apply to the whole table.

The following constraints are commonly used in SQL:

NOT NULL

Ensures that a column cannot have a NULL value

UNIQUE

Ensures that all values in a column are different

PRIMARY KEY

A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.

FOREIGN KEY

Prevents actions that would destroy links between tables

CHECK

Ensures that the values in a column satisfy a specific condition

DEFAULT

Sets a default value for a column if no value is specified

CREATE INDEX

Used to create and retrieve data from the database very quickly

2. How do PRIMARY KEY and FOREIGN KEY constraints differ?

Ans.

Feature	PRIMARY KEY	FOREIGN KEY
Purpose	Uniquely identifies each record in a table.	Establishes a relationship between two tables.
Uniqueness	Must be unique for each row.	Can have duplicate values in the referencing table.
NULL Values	Cannot be NULL.	Can contain NULL values.
Number Allowed	Only one per table.	Multiple foreign keys can exist in a table.
Reference	Defined in the same table.	References a PRIMARY KEY in another table.
Example	id in the employee's table uniquely identifies employees.	employee_id in the orders table links to the id in employees.

3. What is the role of NOT NULL and UNIQUE constraints?

Ans.

SQL NOT NULL Constraint:

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

```
CREATE TABLE Persons
(
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int
);
```

SQL UNIQUE Constraint:

The UNIQUE constraint ensures that all values in a column are different.

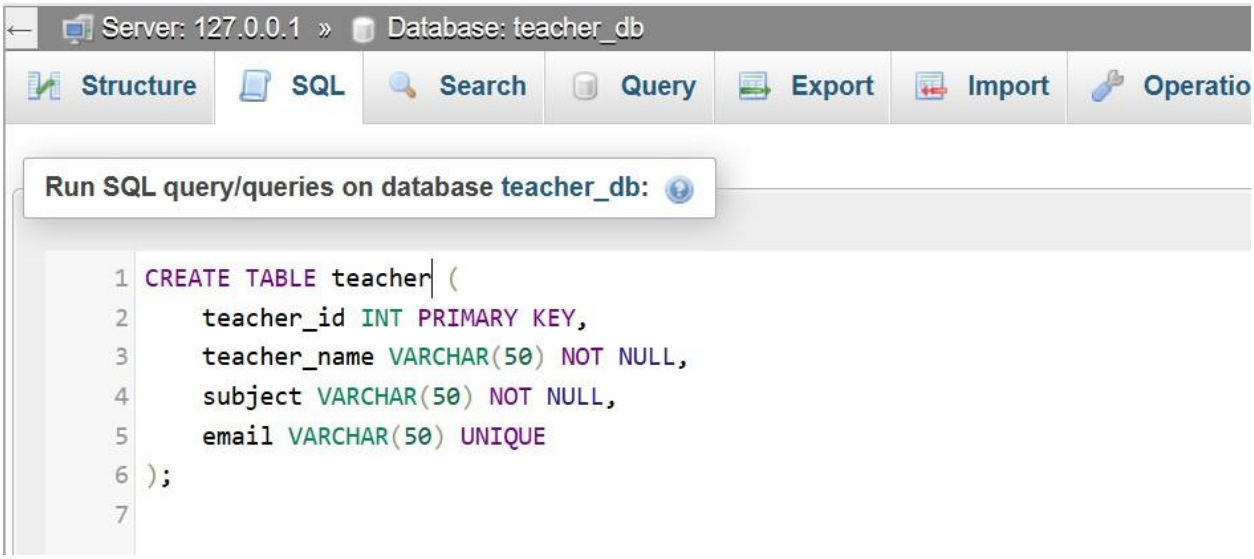
Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

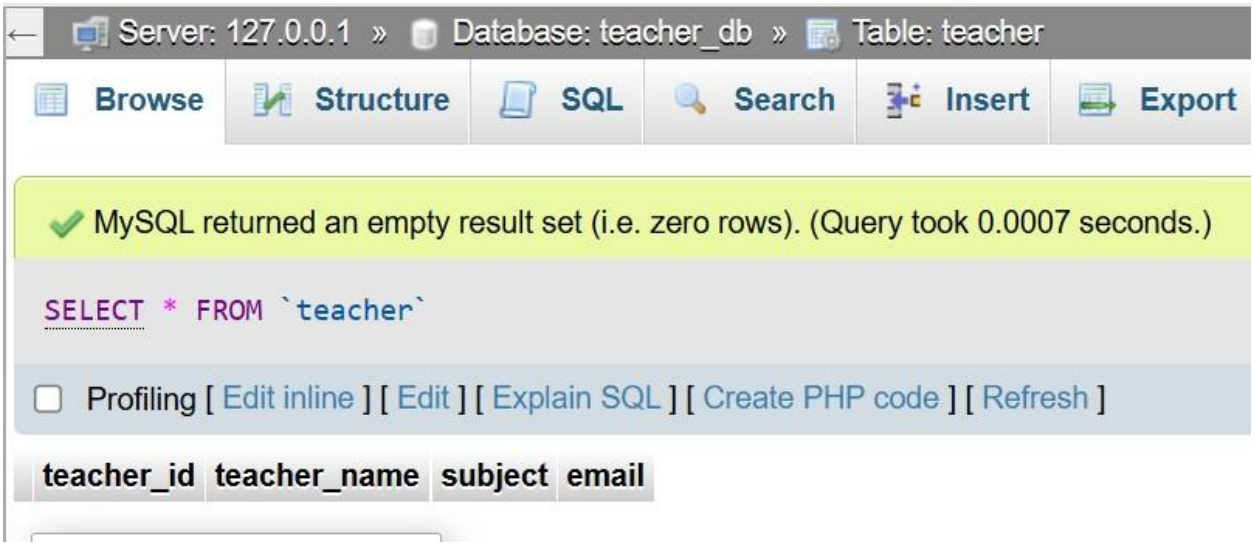
```
CREATE TABLE Persons
(
ID int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int
);
```

LAB EXERCISES:

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

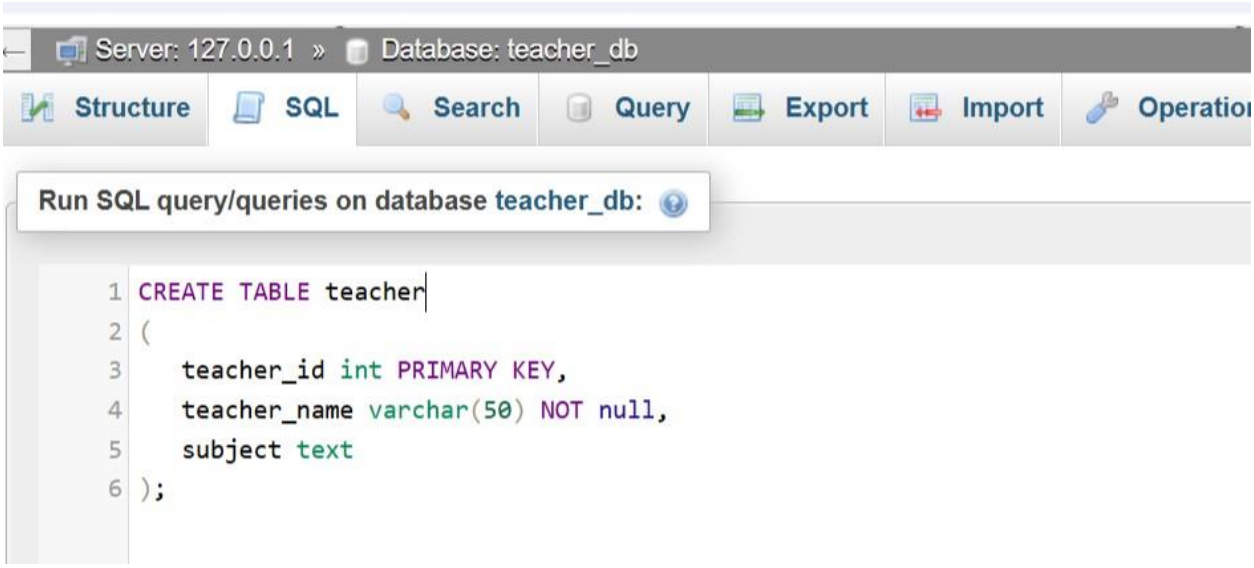
Ans.

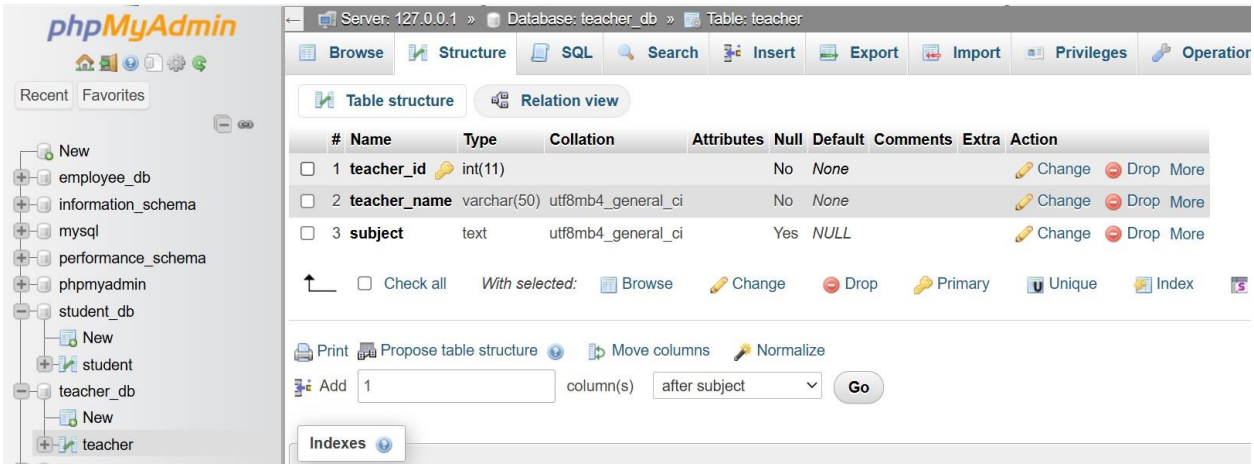


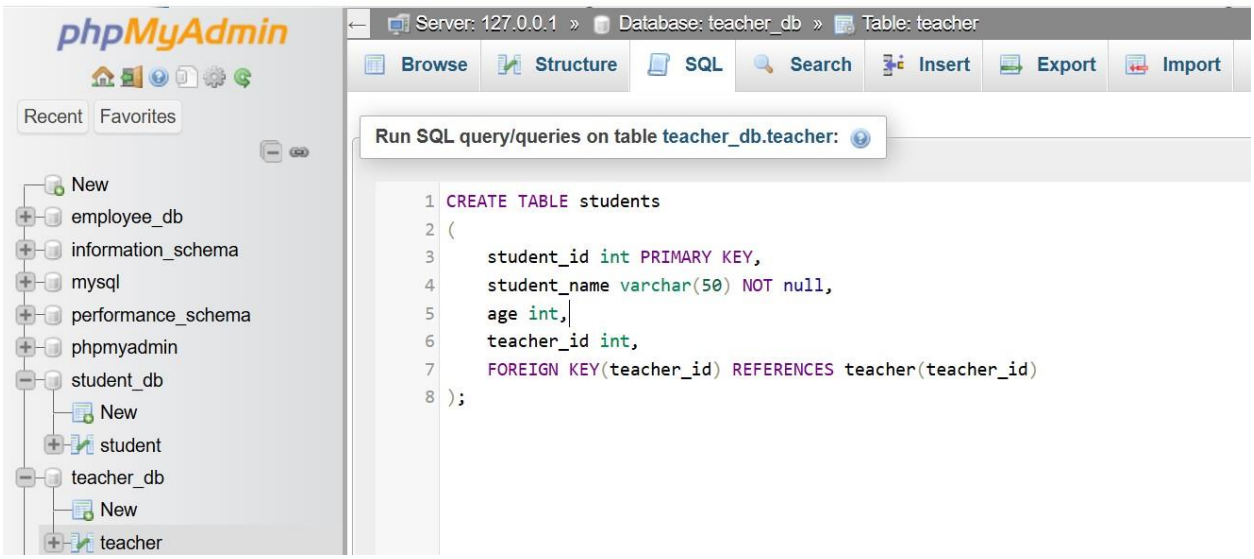


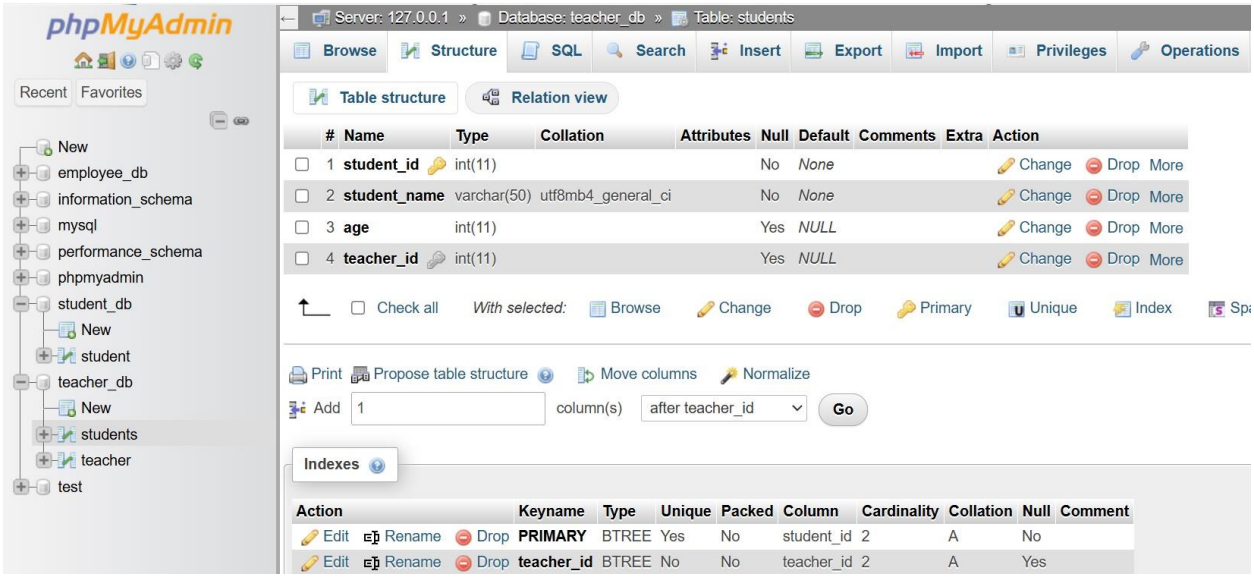
Lab 2: Implement a FOREIGN KEY constraint to relate the teacher_id from the teacher's table with the student's table.

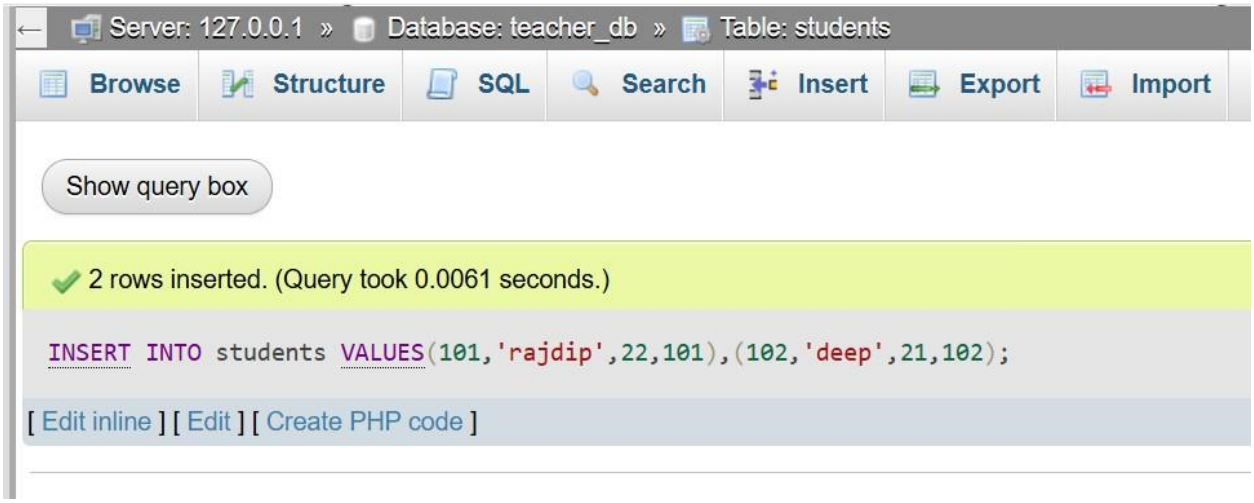
Ans.

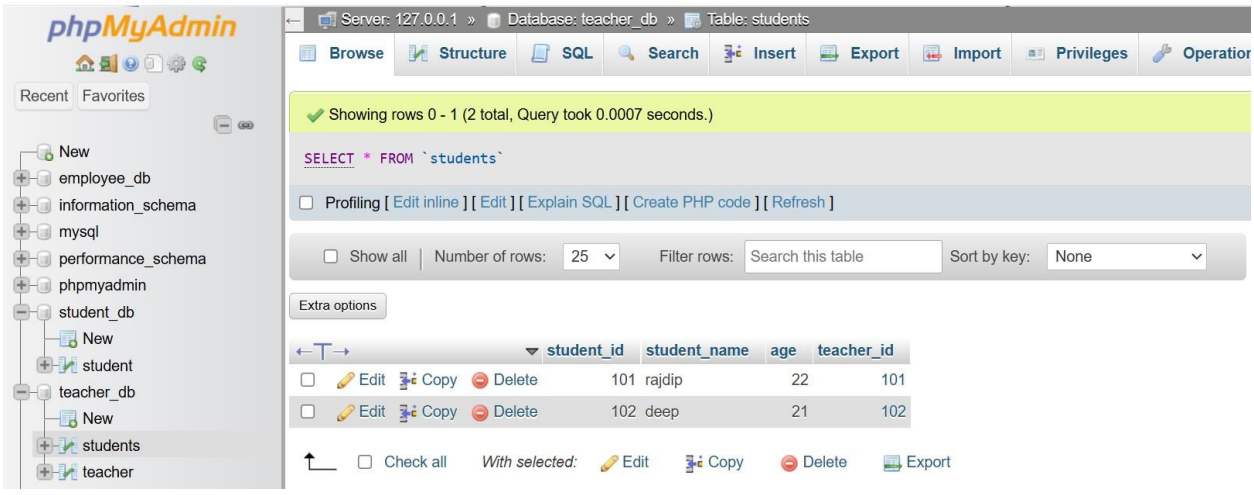

Server: 127.0.0.1 » Database: teacher_db
Structure
SQL
Search
Query
Export
Import
Operatio
Run SQL query/queries on database teacher_db:
1 CREATE TABLE teacher
2 (
3 teacher_id int PRIMARY KEY,
4 teacher_name varchar(50) NOT null,
5 subject text
6);

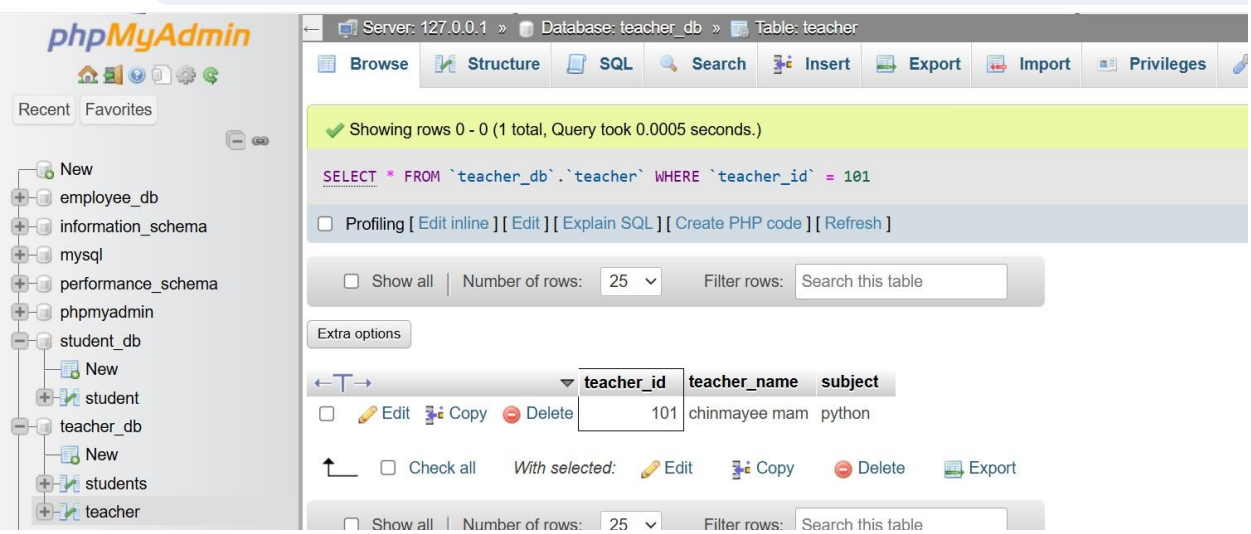

phpMyAdmin
Recent
Favorites
New
employee_db
information_schema
mysql
performance_schema
phpmyadmin
student_db
New
student
teacher_db
New
teacher
Server: 127.0.0.1 » Database: teacher_db » Table: teacher
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operation
Table structure
Relation view
Name Type Collation Attributes Null Default Comments Extra Action
1 teacher_id int(11) No None Change Drop More
2 teacher_name varchar(50) utf8mb4_general_ci No None Change Drop More
3 subject text utf8mb4_general_ci Yes NULL Change Drop More
Check all
With selected:
Browse
Change
Drop
Primary
Unique
Index
Print
Propose table structure
Move columns
Normalize
Add 1 column(s) after subject Go
Indexes


phpMyAdmin
Recent
Favorites
New
employee_db
information_schema
mysql
performance_schema
phpmyadmin
student_db
New
student
teacher_db
New
teacher
Server: 127.0.0.1 » Database: teacher_db » Table: teacher
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table teacher_db.teacher:
1 CREATE TABLE students
2 (
3 student_id int PRIMARY KEY,
4 student_name varchar(50) NOT null,
5 age int,
6 teacher_id int,
7 FOREIGN KEY(teacher_id) REFERENCES teacher(teacher_id)
8);


phpMyAdmin
Recent Favorites
New
employee_db
information_schema
mysql
performance_schema
phpmyadmin
student_db
New
student
teacher_db
New
students
teacher
test
Server: 127.0.0.1 » Database: teacher_db » Table: students
Browse Structure SQL Search Insert Export Import Privileges Operations
Table structure Relation view
Name Type Collation Attributes Null Default Comments Extra Action
1 student_id int(11) No None Change Drop More
2 student_name varchar(50) utf8mb4_general_ci No None Change Drop More
3 age int(11) Yes NULL Change Drop More
4 teacher_id int(11) Yes NULL Change Drop More
Check all With selected: Browse Change Drop Primary Unique Index
Print Propose table structure Move columns Normalize
Add 1 column(s) after teacher_id Go
Indexes
Action Keyname Type Unique Packed Column Cardinality Collation Null Comment
Edit Rename Drop PRIMARY BTREE Yes No student_id 2 A No
Edit Rename Drop teacher_id BTREE No No teacher_id 2 A Yes


Server: 127.0.0.1 » Database: teacher_db » Table: students
Browse Structure SQL Search Insert Export Import
Show query box
2 rows inserted. (Query took 0.0061 seconds.)
INSERT INTO students VALUES(101,'rajdip',22,101),(102,'deep',21,102);
[Edit inline] [Edit] [Create PHP code]


phpMyAdmin
Recent Favorites
New
employee_db
information_schema
mysql
performance_schema
phpmyadmin
student_db
New
student
teacher_db
New
students
teacher
Server: 127.0.0.1 » Database: teacher_db » Table: students
Browse Structure SQL Search Insert Export Import Privileges Operation
Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)
SELECT * FROM `students`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
student_id student_name age teacher_id
Edit Copy Delete 101 rajdip 22 101
Edit Copy Delete 102 deep 21 102
Check all With selected: Edit Copy Delete Export



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

Theory Questions:

1. Define the SQL Data Definition Language (DDL).

Ans.

1. Data Definition (DDL - Data Definition Language):

   Creating, modifying, and deleting database structures such as tables, indexes, and schemas.

   Commands: CREATE, TRUNCATE, DROP.

SQL CREATE TABLE Statement:

CREATE TABLE Employee (Emp_No int, Emp_Name varchar(50), Salary int);

The SQL DROP TABLE Statement:

DROP TABLE TestEmployee;

SQL TRUNCATE TABLE:

TRUNCATE TABLE TestEmployee;

To delete the data inside a table, but not the table itself.

2. Explain the CREATE command and its syntax.

Ans.

SQL CREATE Command:

The CREATE command in SQL is used to create new database objects such as databases, tables, indexes, views, and schemas.

It is part of the Data Definition Language (DDL) and helps define the structure of these objects.

The CREATE command defines the structure of database objects.

1. Create a Database:

   CREATE DATABASE company_db;

Syntax of CREATE TABLE:

```
CREATE TABLE table_name (
column1 datatype constraints,
column2 datatype constraints,
...
);
```

3. What is the purpose of specifying data types and constraints during table creation?

Ans.

1. Purpose of Data Types:

Data types define the kind of data a column can store, preventing invalid entries and optimizing storage.

Data types ensure correct and efficient storage of values.

Example:

```
CREATE TABLE employees (
```

```
id INT,
name VARCHAR(50),
salary INT
);
```

2. Purpose of Constraints:

Constraints enforce rules on the data stored in a table, ensuring its validity and integrity.

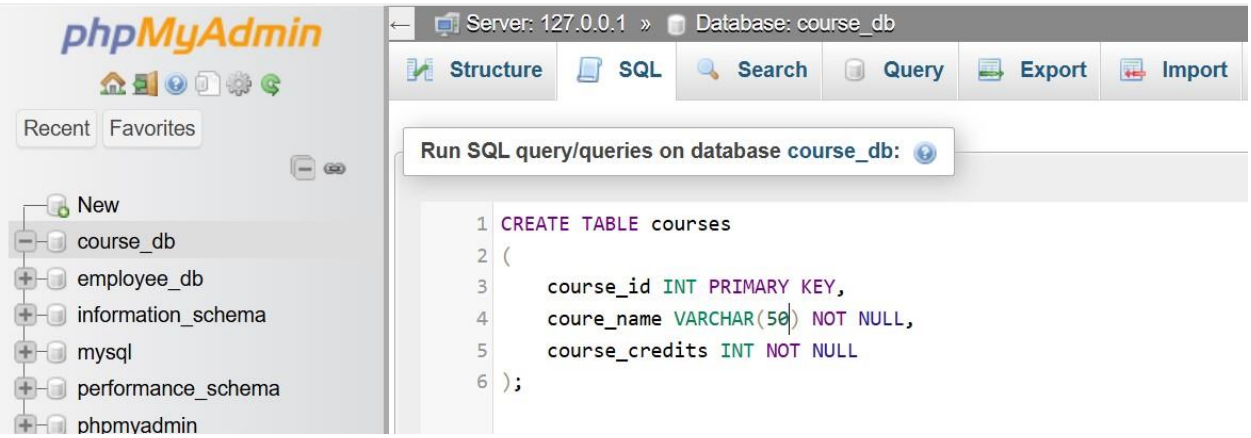
Example:

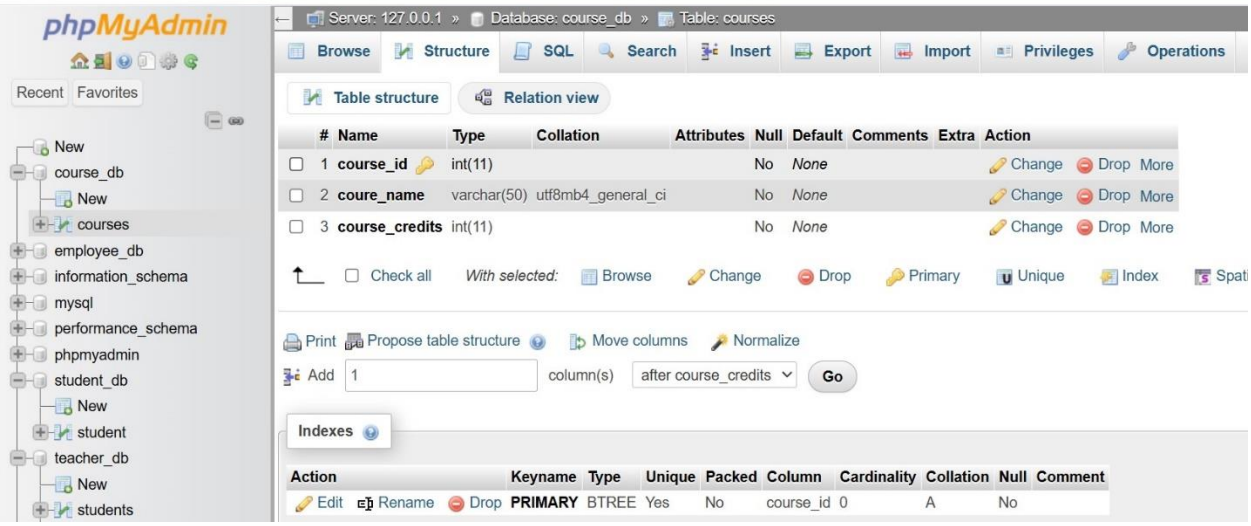
```
CREATE TABLE employees (
id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
age INT CHECK (age >= 18),
email VARCHAR(100) UNIQUE,
department_id INT,
FOREIGN KEY (department_id) REFERENCES departments(id)
);
```

LAB EXERCISES:

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

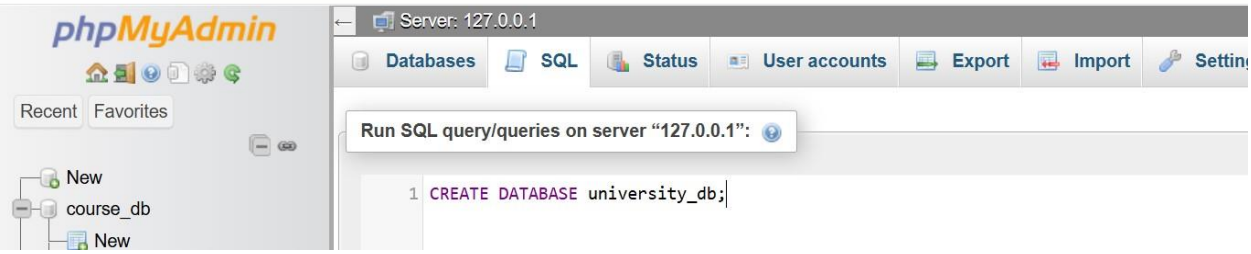
Ans.

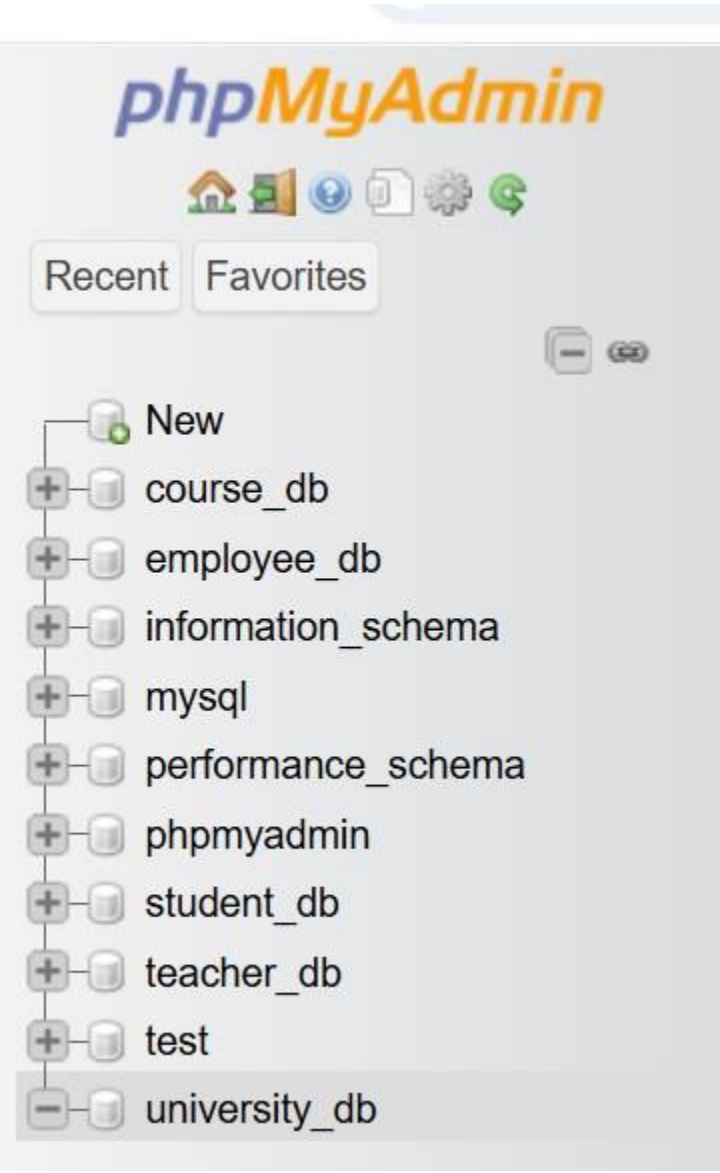




Lab 2: Use the CREATE command to create a database university_db.

Ans.





5. ALTER Command

Theory Questions:

1. What is the use of the ALTER command in SQL?

Ans.

It is part of the Data Definition Language (DDL).

allows adding, modifying, or deleting table columns and constraints.

ALTER allows modifying table structure without losing data.

It can add, modify, rename, or delete columns and constraints.

Helps adapt the database to changing requirements efficiently.

1. Add a New Column to a Table:

   ALTER TABLE employees ADD COLUMN salary int;

2. Drop (Delete) a Column from a Table:
   ALTER TABLE employees DROP COLUMN age;
3. Rename a Table:
   ALTER TABLE employees RENAME TO staff;

2. How can you add, modify, and drop columns from a table using ALTER?

Ans.

The ALTER TABLE command allows you to add, modify, and drop columns in an existing table without losing data.

1. Add a Column to a Table:

You can add a new column to an existing table using the ADD clause.

Example:

    ALTER TABLE employees ADD COLUMN salary float;

2. Modify an Existing Column:

You can change the data type, size, or constraints of an existing column using the MODIFY or ALTER COLUMN clause.

Example:

    ALTER TABLE employees MODIFY COLUMN salary float;

3. Drop (Delete) a Column from a Table:

To remove a column from a table, use the DROP COLUMN clause.

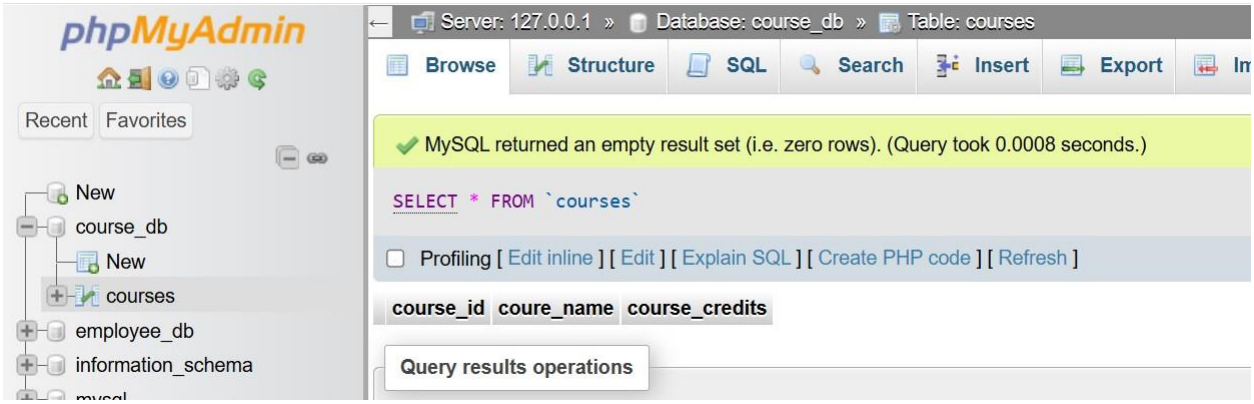
Example:

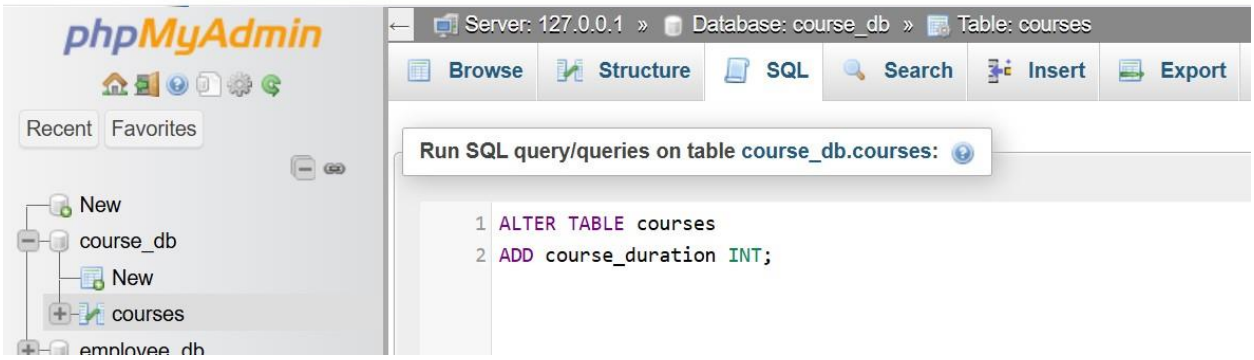
    ALTER TABLE employees DROP COLUMN age;

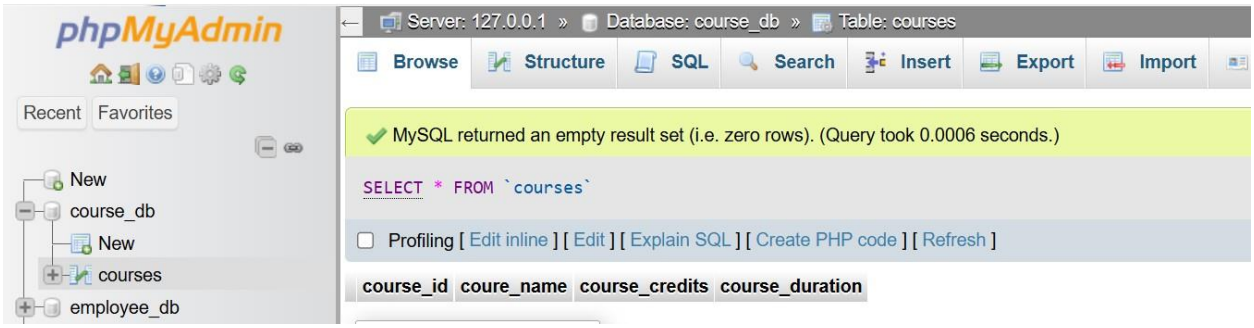
LAB EXERCISES:

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

Ans.

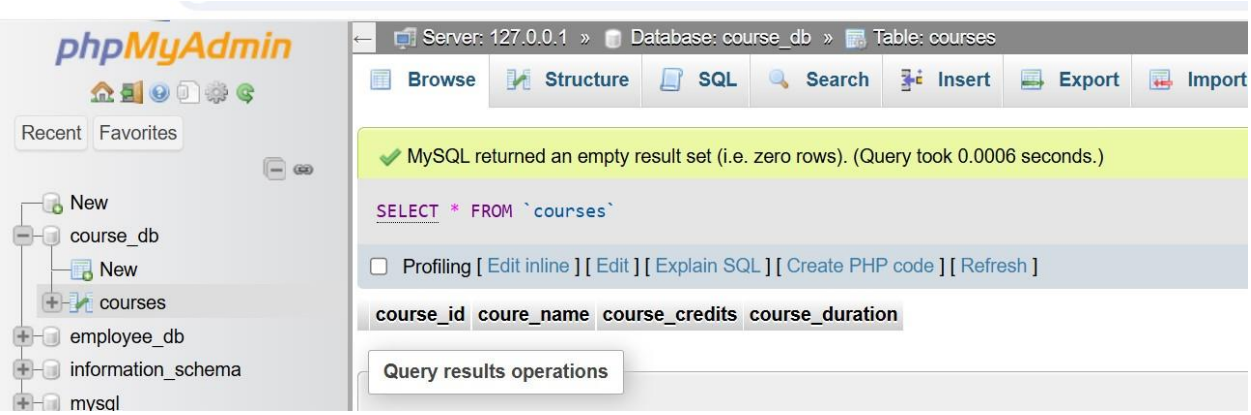


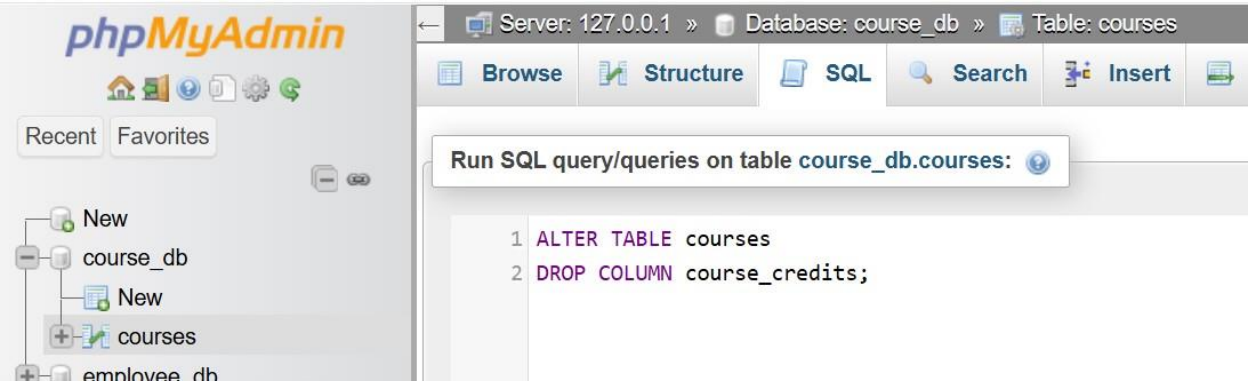


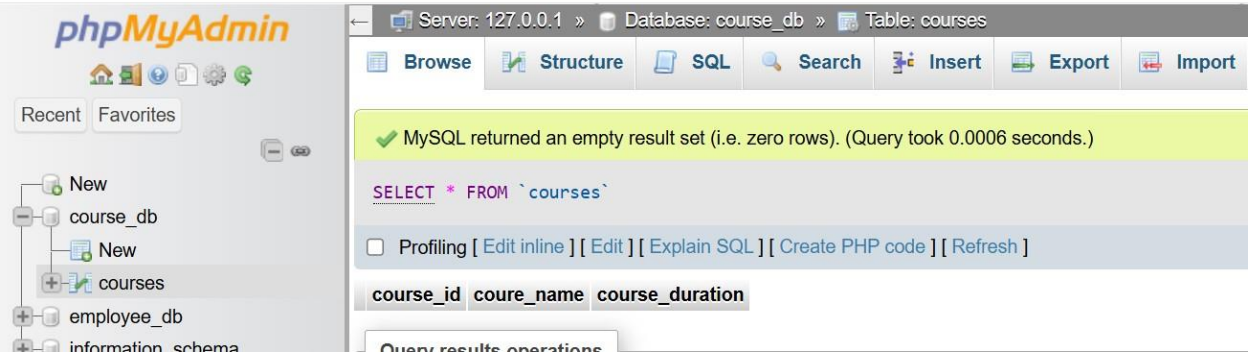


Lab 2: Drop the course_credits column from the courses table.

Ans.







6. DROP Command

Theory Questions:

1. What is the function of the DROP command in SQL?

Ans.

The DROP command permanently deletes database objects.

It cannot be undone, so use it carefully.

It can remove tables, databases, columns, indexes, and views.

1. Drop a Table (Delete a Table):

Example:

```
DROP TABLE employees;
```

2. Drop a Database (Delete an Entire Database):

Example:

```
DROP DATABASE company_db;
```

3. Drop a Column from a Table:

Example:

```
ALTER TABLE employees DROP COLUMN age;
```

2. What are the implications of dropping a table from a database?

Ans.

Dropping a table is permanent – use it cautiously.

Foreign key constraints and relationships may break.

Back up your data before executing the DROP TABLE.

1. Data Loss:

   All records in the table are permanently deleted.

   The data cannot be recovered unless a backup exists.

2. Loss of Table Structure:

   The table definition (like columns, and data types) is removed.

   You must recreate the table manually if needed later.

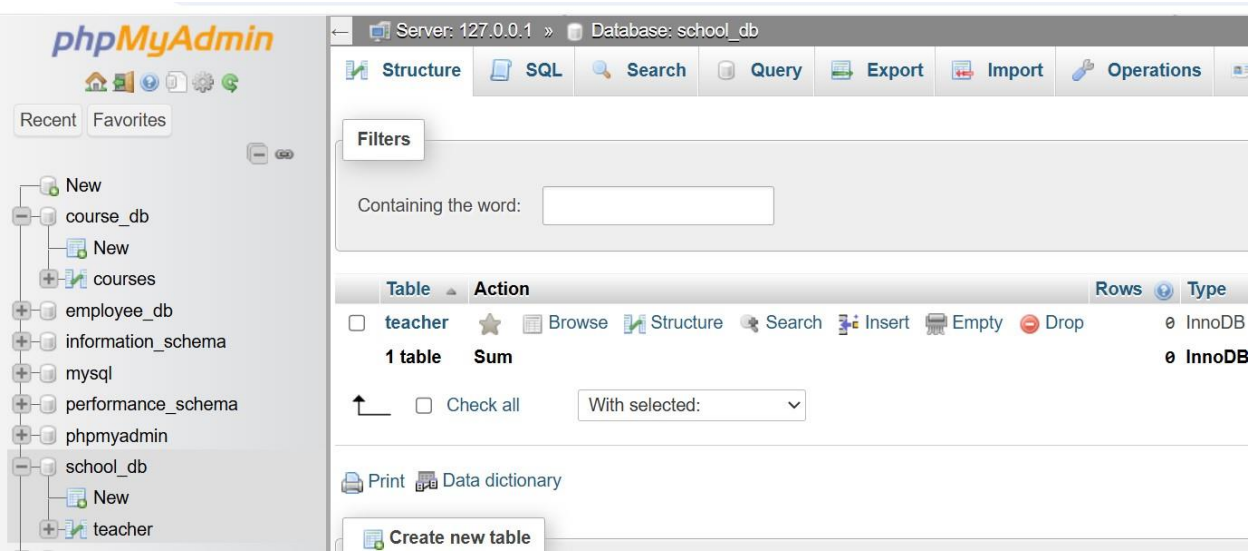
Example: Dropping a Table

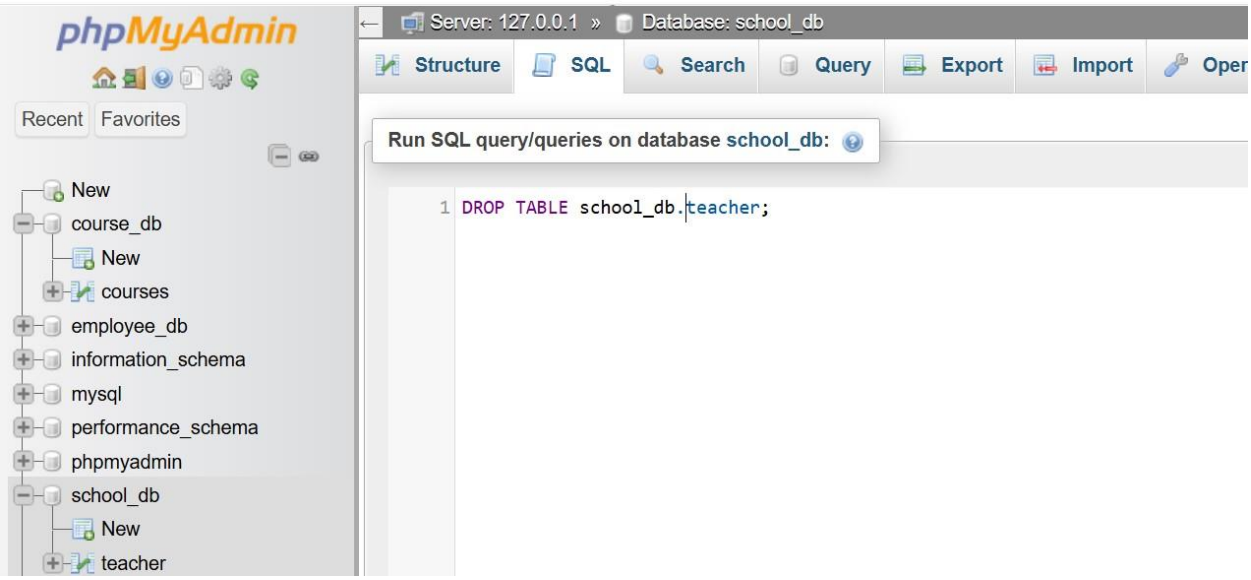
```
DROP TABLE employees;
```

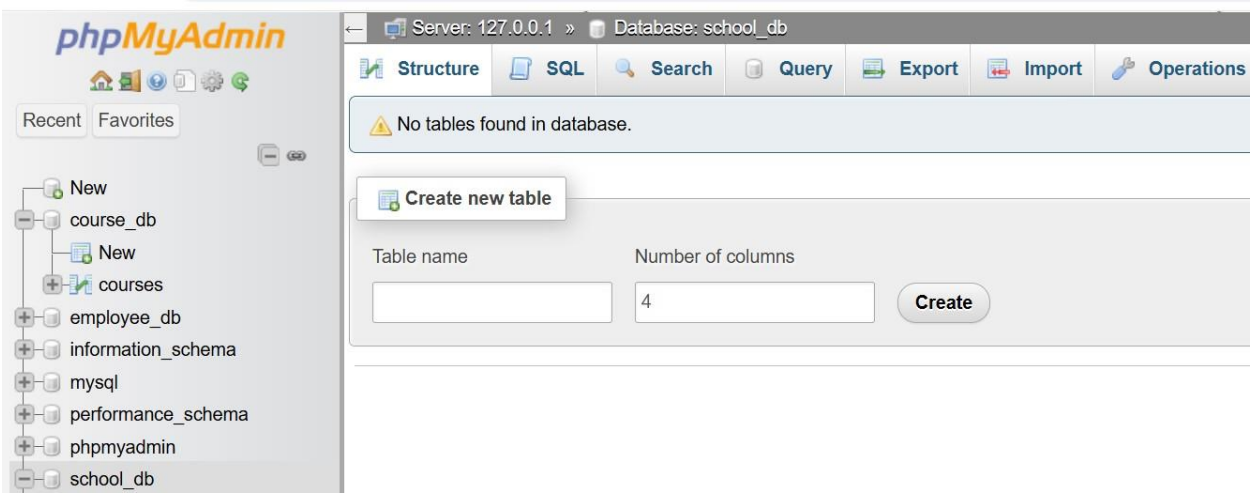
LAB EXERCISES:

Lab 1: Drop the teacher's table from the school_db database.

Ans.

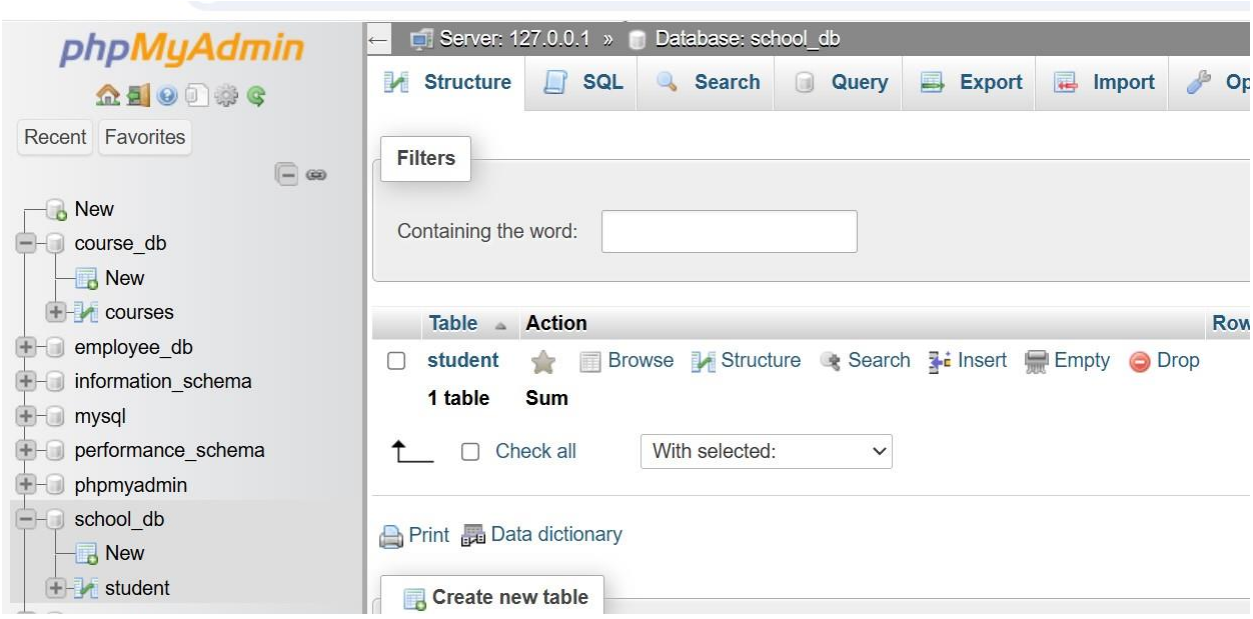


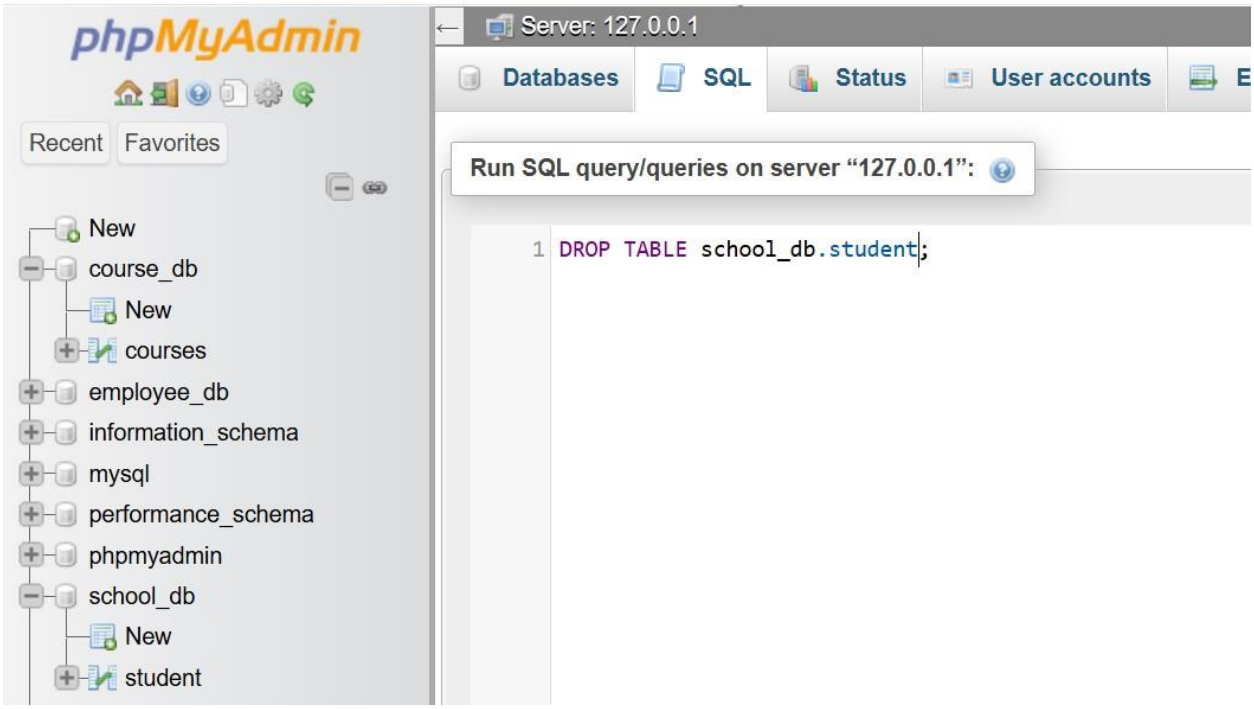


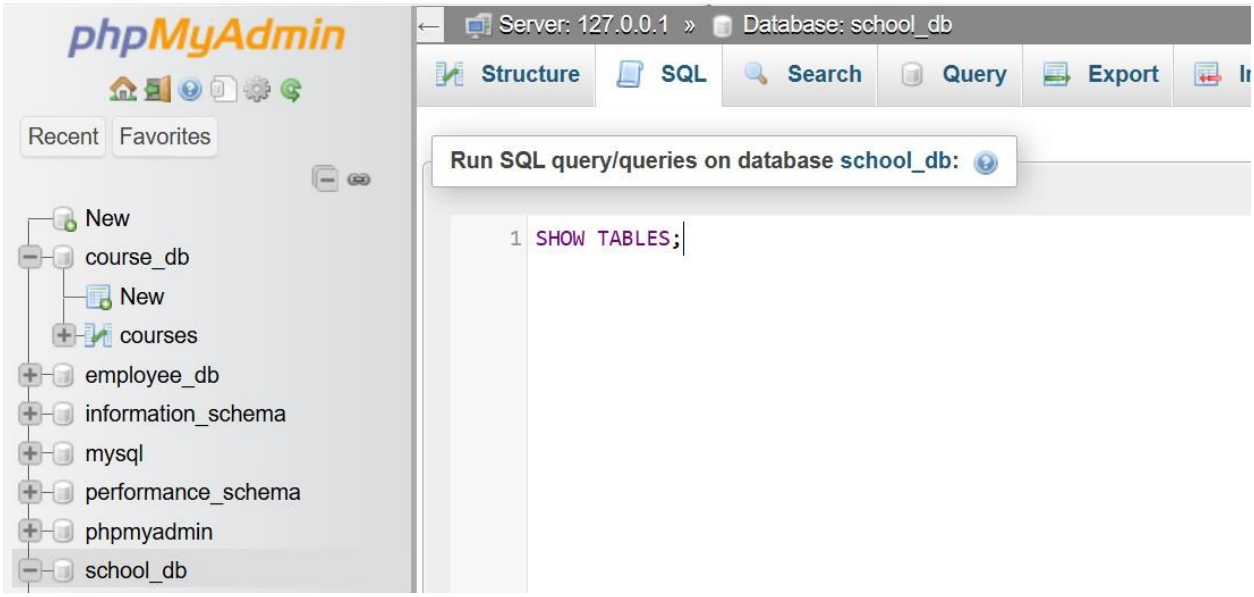


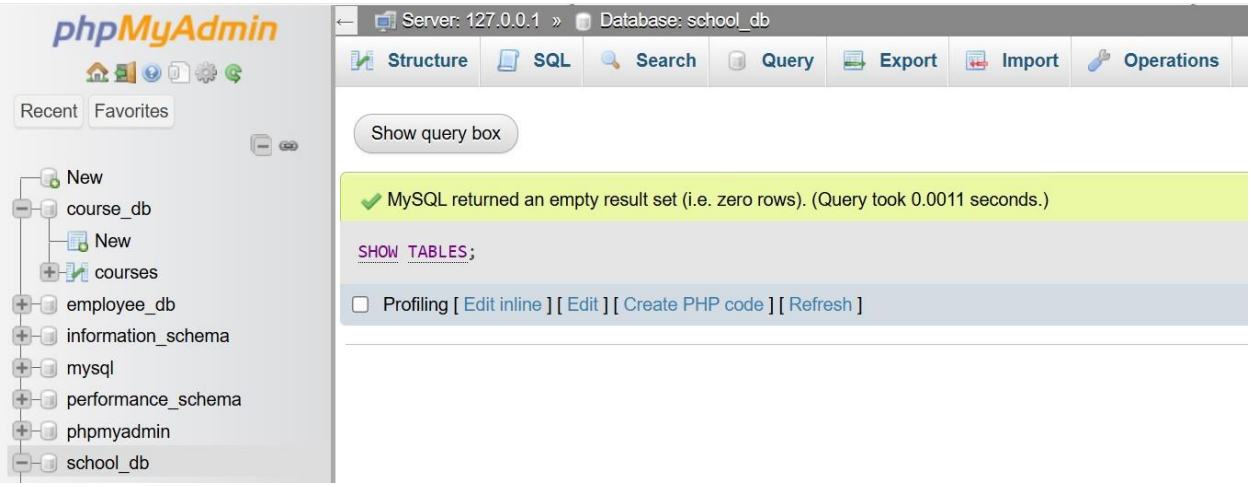
Lab 2: Drop the student's table from the school_db database and verify that the table has been removed.

Ans.




phpMyAdmin
Recent
Favorites
New
course_db
New
courses
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
New
student
Server: 127.0.0.1
Databases
SQL
Status
User accounts
Run SQL query/queries on server "127.0.0.1":
1 DROP TABLE school_db.student;


phpMyAdmin
Recent
Favorites
New
course_db
New
courses
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
Server: 127.0.0.1 » Database: school_db
Structure
SQL
Search
Query
Export
Run SQL query/queries on database school_db:
1 SHOW TABLES;



7. Data Manipulation Language (DML)

Theory Questions:

1. Define the INSERT, UPDATE, and DELETE commands in SQL.

Ans.

These commands are part of Data Manipulation Language (DML).

SQL is used to manage and modify data within database tables.

1. INSERT Command – Add New Records:

The INSERT command is used to add new rows to a table.

Example:

```
INSERT INTO employees (id, name, age, department)
VALUES (101, 'meet', 30, 'HR');
```

2. UPDATE Command – Modify Existing Records:

The UPDATE command modifies existing data in a table.

Example:

```
UPDATE employees
SET age = 32
```

WHERE id = 101;

3. DELETE Command – Remove Records:

The DELETE command is used to remove records from a table.

Example:

DELETE FROM employees WHERE id = 103;

2. What is the importance of the WHERE clause in UPDATE and DELETE operations?

Ans.

All rows in the table will be modified or deleted, leading to data loss or unintended changes.

1. Importance in UPDATE Statements:

The WHERE clause ensures that only specific records are updated.

Example:

UPDATE employees

SET age = 35

WHERE id = 101;

Without WHERE, every row is updated!

Example:

UPDATE employees

SET age = 35;

2. Importance in DELETE Statements:

The WHERE clause prevents accidental deletion of all records in the table.

Example:

DELETE FROM employees

WHERE id = 103;

Without WHERE, every record is deleted!

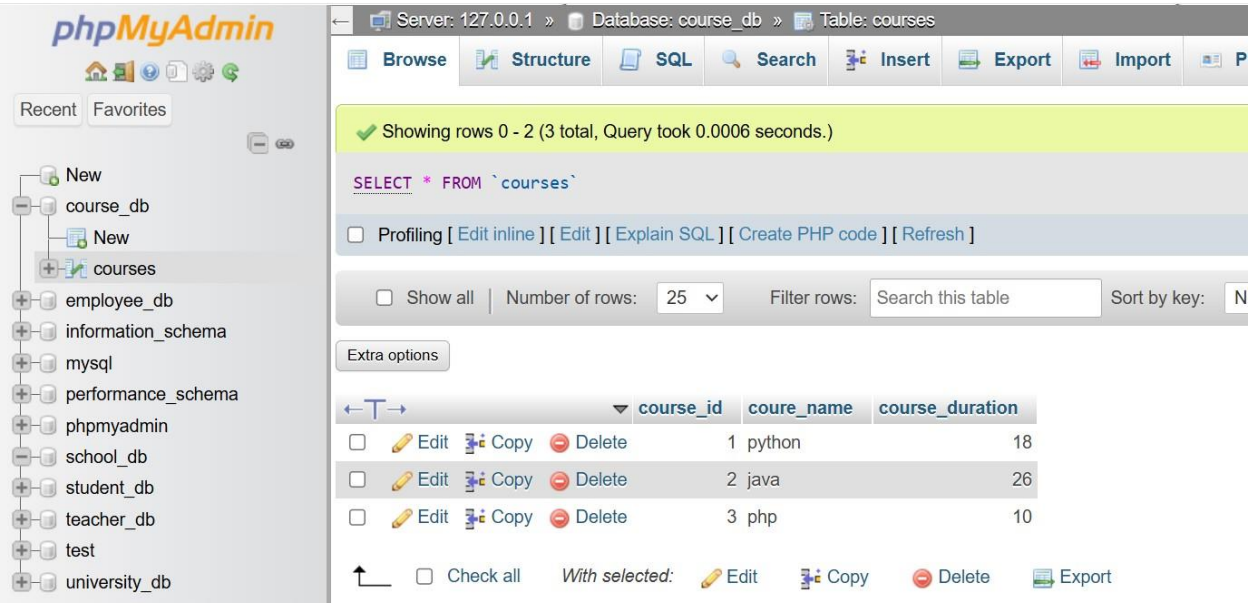
Example:

```
DELETE FROM employees;
```

LAB EXERCISES:

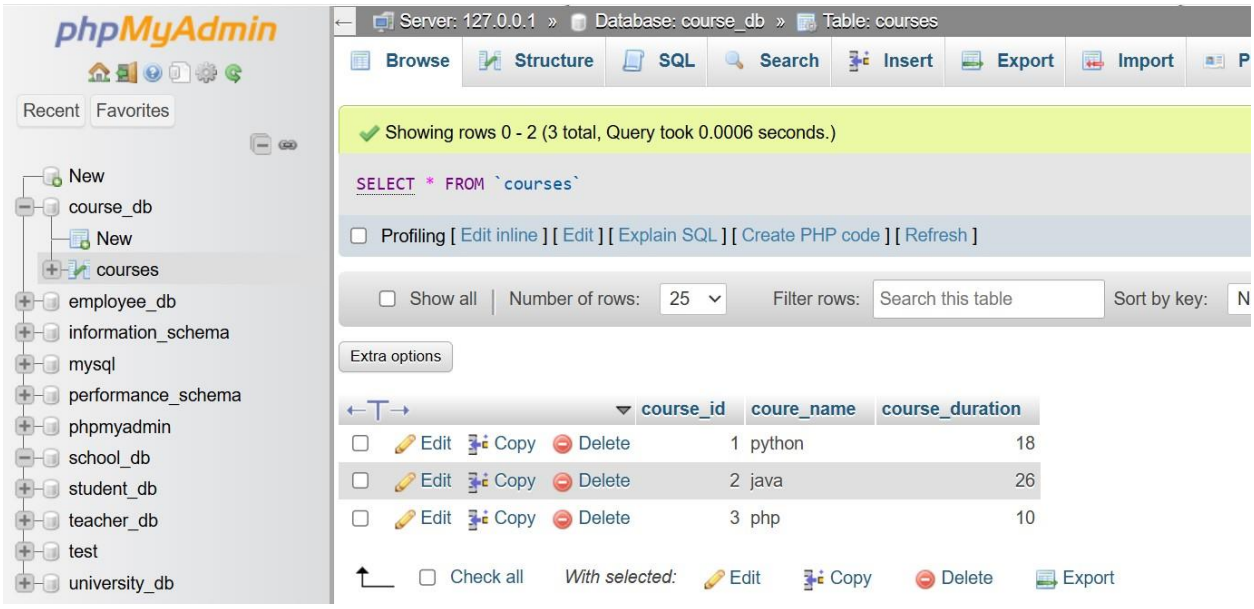
Lab 1: Insert three records into the courses table using the INSERT command.

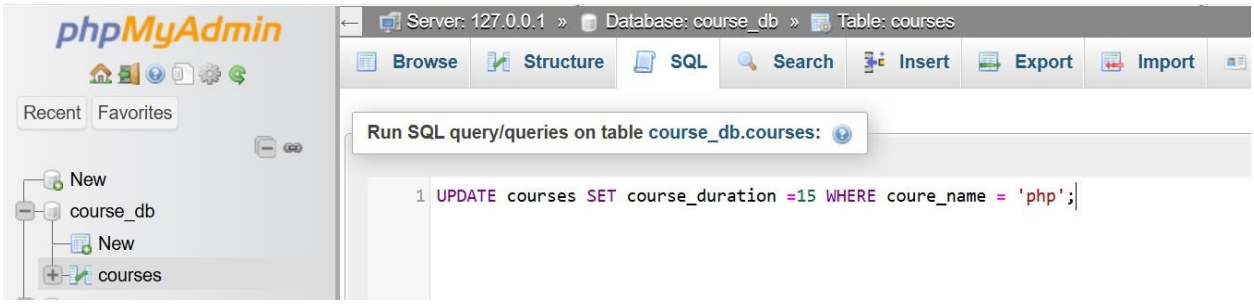
Ans.

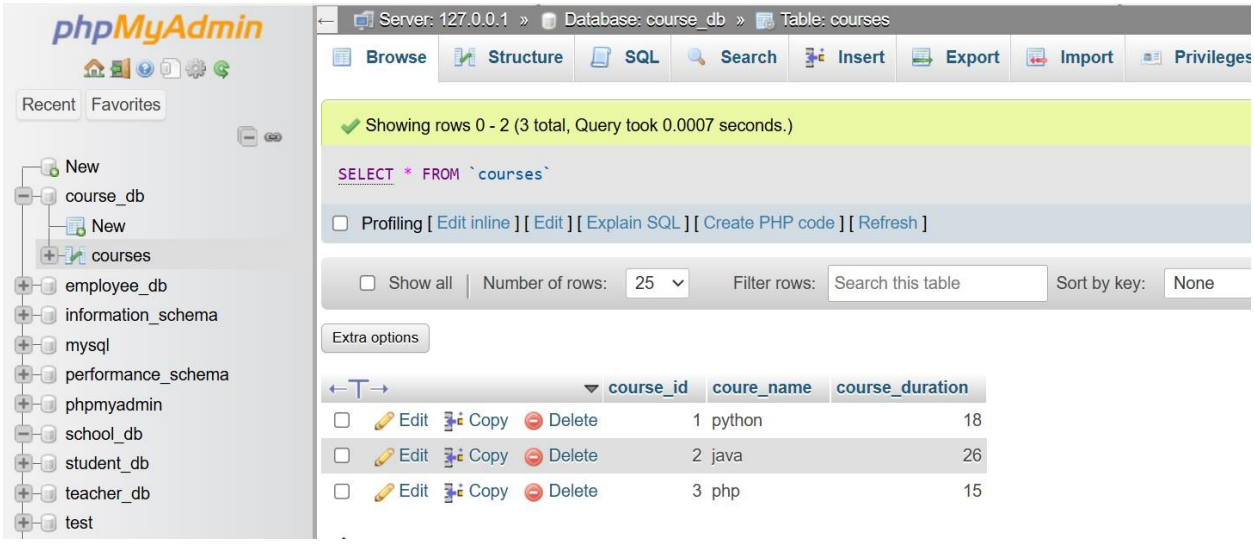


Lab 2: Update the course duration of a specific course using the UPDATE command.

Ans.

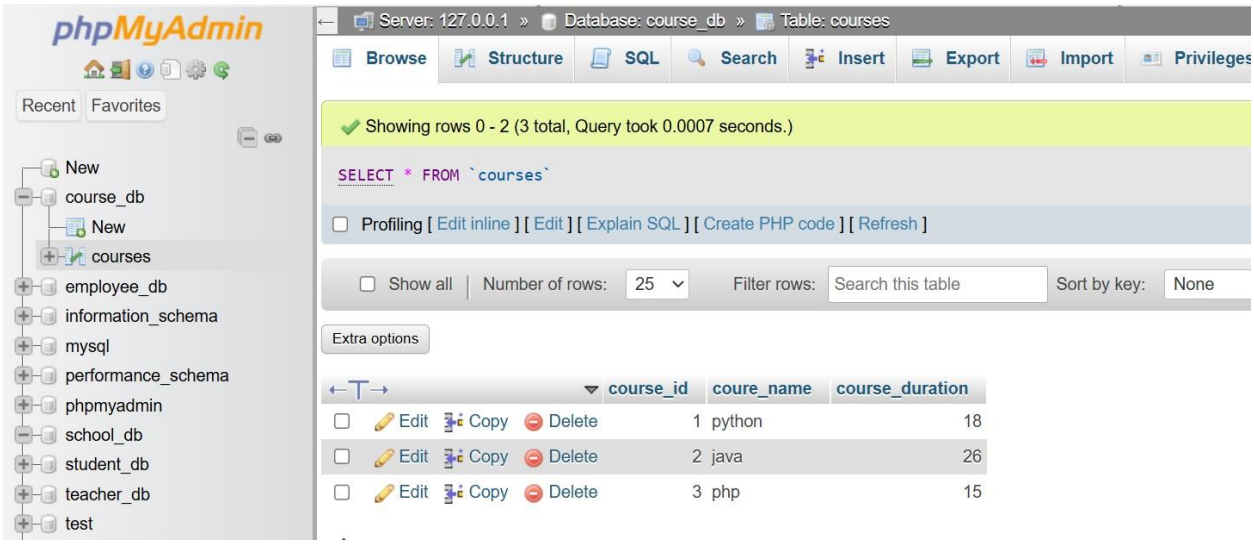


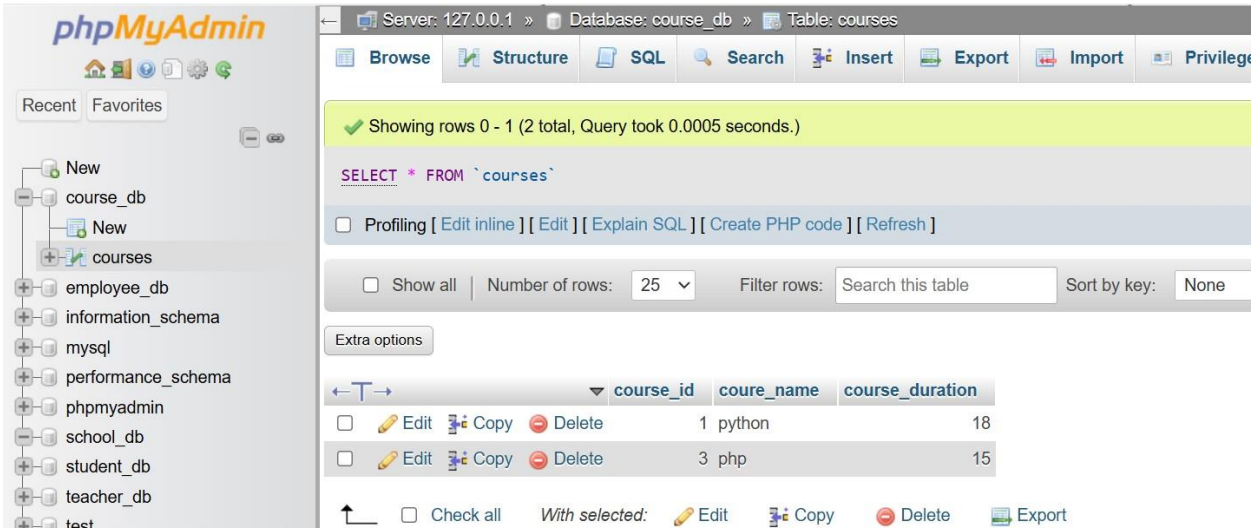




Lab 3: Delete a course with a specific course_id from the courses table using the DELETE command.

Ans.





8. Data Query Language (DQL)

Theory Questions:

1. What is the SELECT statement, and how is it used to query data?

Ans.

The SELECT statement is one of the most important SQL commands.

It is used to query data from a table and retrieve specific records based on conditions.

1. Basic Syntax of SELECT:

```
SELECT column1, column2, ... FROM table_name;
```

Example:

```
SELECT name, age FROM employees;
```

2. Using WHERE to Filter Data:

The WHERE clause helps retrieve specific records based on a condition.

Example:

```
SELECT name, department FROM employees
WHERE department = 'IT';
```

3. Sorting Results Using ORDER BY:

The ORDER BY clause sorts query results in ascending (ASC) or descending (DESC) order.

Example:

```
SELECT name, age FROM employees
ORDER BY age DESC;
```

4. Using GROUP BY for Aggregations:

The GROUP BY clause groups results based on a column and HAVING filters aggregated results.

Example:

```
SELECT department, COUNT(*) AS total_employees
FROM employees
GROUP BY department;
```

2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.

Ans.

1. WHERE Clause – Filtering Data:

The WHERE clause is used to filter records by specifying conditions. It helps retrieve only the relevant rows from a table.

Syntax:

```
SELECT column1, column2
FROM table_name
WHERE condition;
```

Example:

```
SELECT name, department
FROM employees
WHERE department = 'IT';
```

2. ORDER BY Clause – Sorting Data:

The ORDER BY clause is used to sort query results in ascending (ASC) or descending (DESC) order.

Syntax:

```
SELECT column1, column2
FROM table_name
ORDER BY column_name [ASC | DESC];
```

Example:

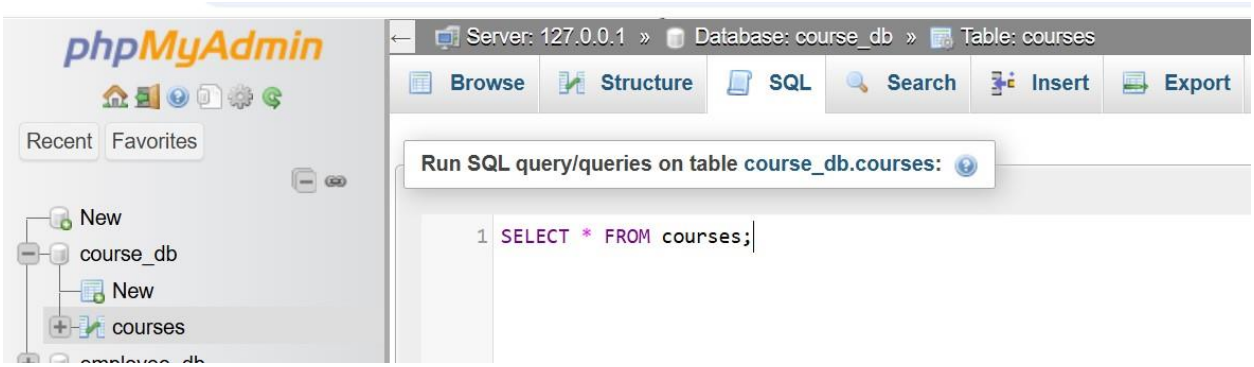
```
SELECT name, age FROM employees
```

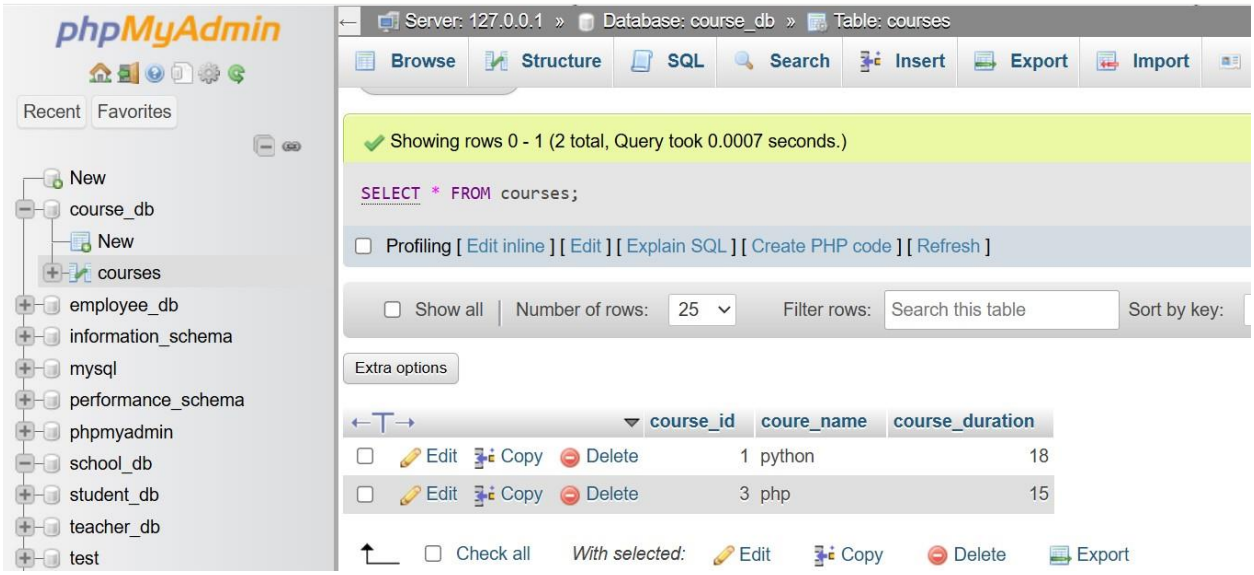
ORDER BY age ASC;

LAB EXERCISES:

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

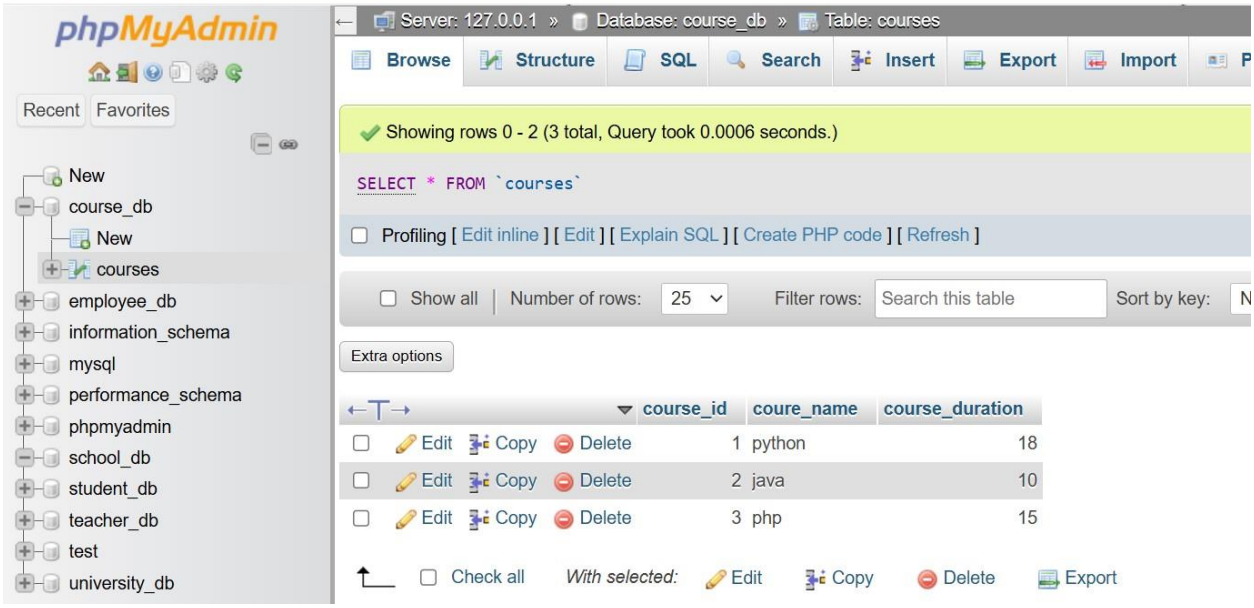
Ans.

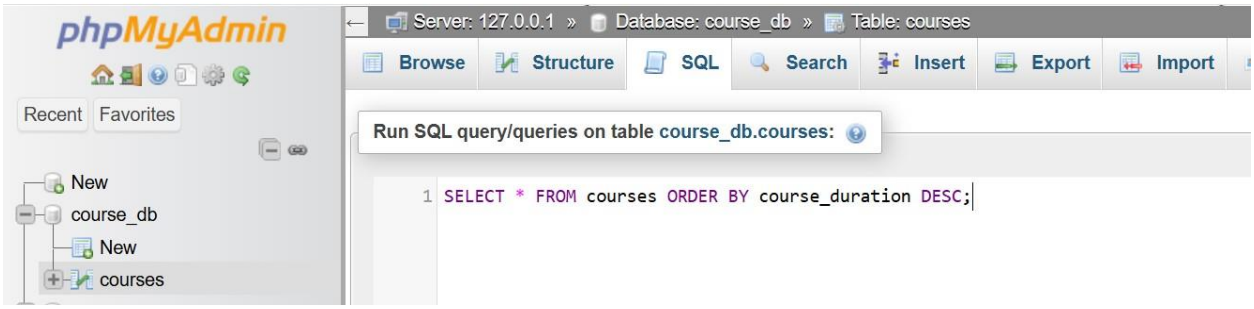


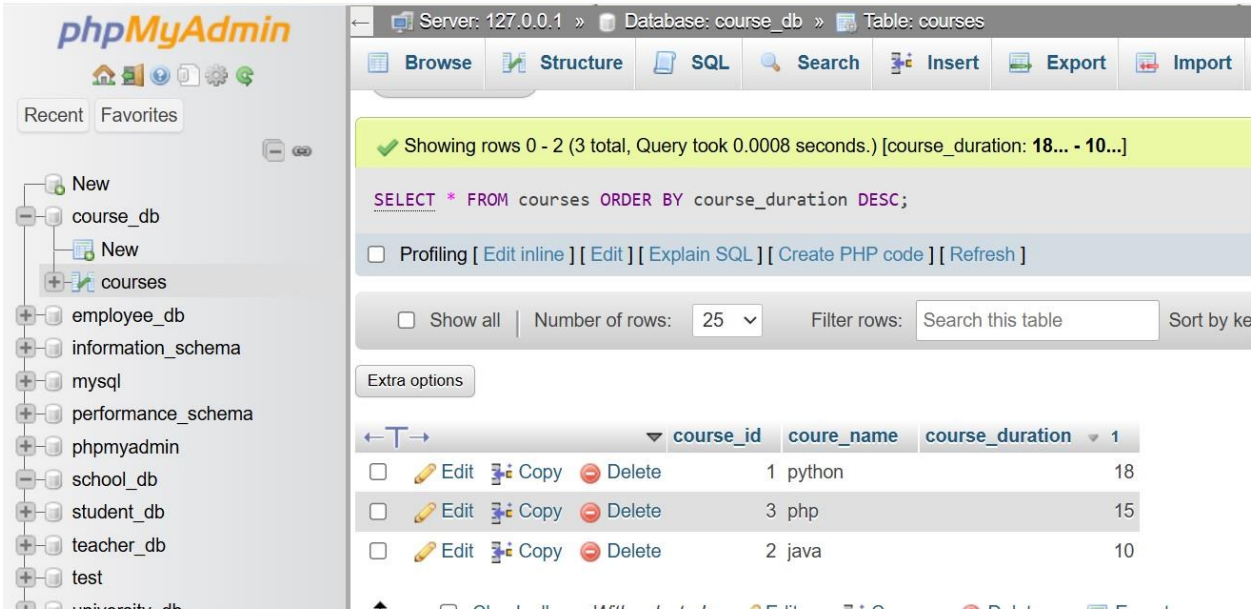


Lab 2: Sort the courses based on course_duration in descending order using ORDER BY.

Ans.

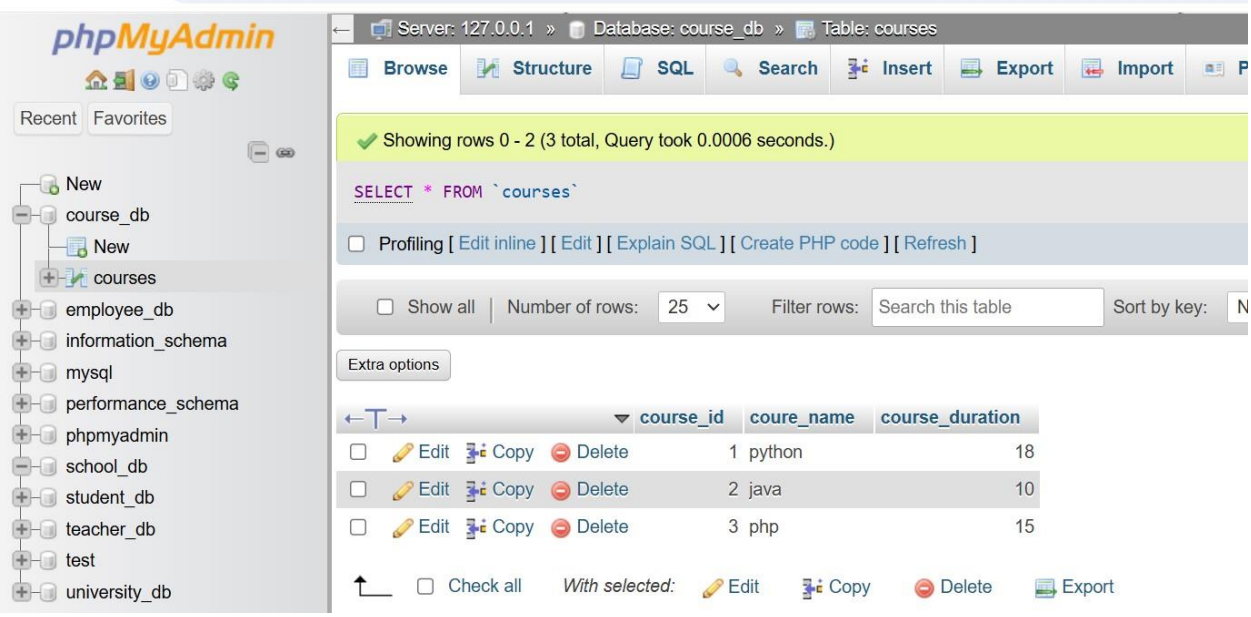

phpMyAdmin
Recent
Favorites
New
course_db
New
courses
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
student_db
teacher_db
test
university_db
Server: 127.0.0.1 » Database: course_db » Table: courses
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)
SELECT * FROM `courses`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key:
Extra options
course_id
coure_name
course_duration
Edit Copy Delete 1 python 18
Edit Copy Delete 2 java 10
Edit Copy Delete 3 php 15
Check all
With selected:
Edit
Copy
Delete
Export

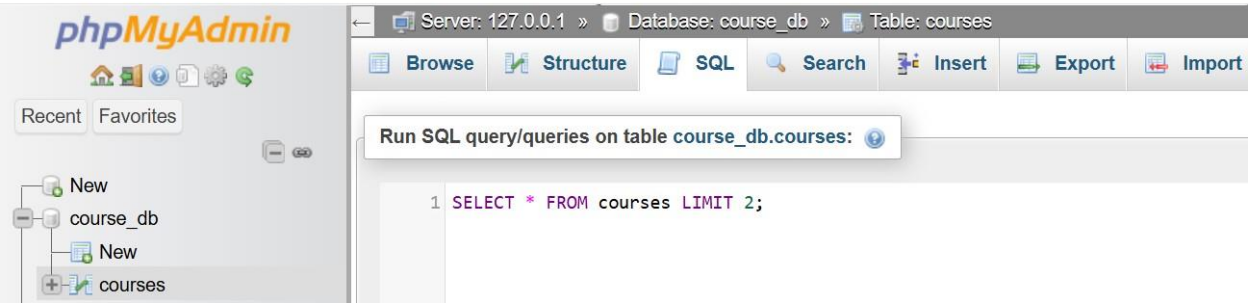

phpMyAdmin
Recent
Favorites
New
course_db
New
courses
Server: 127.0.0.1 » Database: course_db » Table: courses
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table course_db.courses:
1 SELECT * FROM courses ORDER BY course_duration DESC;

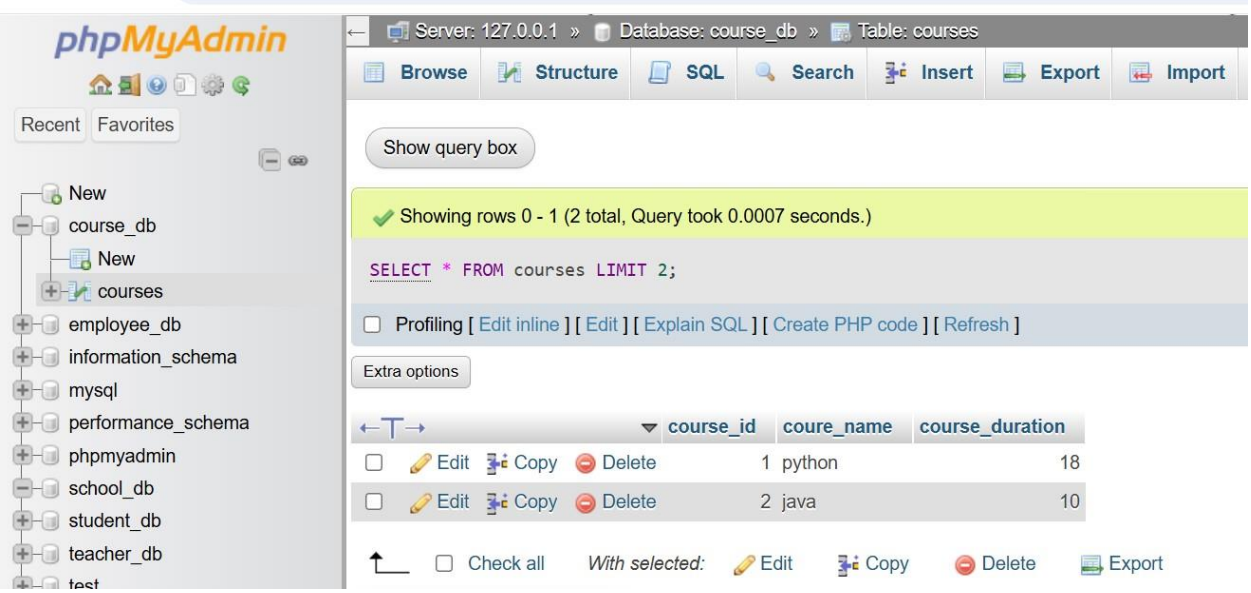

phpMyAdmin
Recent
Favorites
New
course_db
New
courses
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
student_db
teacher_db
test
Server: 127.0.0.1 » Database: course_db » Table: courses
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.) [course_duration: 18... - 10...]
SELECT * FROM courses ORDER BY course_duration DESC;
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Extra options
course_id
coure_name
course_duration 1
Edit Copy Delete 1 python 18
Edit Copy Delete 3 php 15
Edit Copy Delete 2 java 10

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

Ans.







9. Data Control Language (DCL)

Theory Questions:

1. What is the purpose of GRANT and REVOKE in SQL?

Ans.

GRANT is used to give users access to database objects.

REVOKE is used to remove permissions from users.

Database administrators use these commands to control security and protect data integrity.

Command	Purpose	Example
GRANT	Assigns privileges to a user.	GRANT SELECT ON employees to Raj;
REVOKE	Removes privileges from a user.	REVOKE SELECT ON employees FROM Raj;
Effect	Increases user access.	Restricts user access.

2. How do you manage privileges using these commands?

Ans.

Use roles instead of assigning privileges to each user manually.
Grant only necessary permissions to protect data.
Regularly audit privileges to maintain security.

Command	Purpose	Example
GRANT	Assign privileges to a user or role.	GRANT SELECT ON employees to Raj;
REVOKE	Remove privileges from a user or role.	REVOKE UPDATE ON employees FROM Raj;
WITH GRANT OPTION	Allows a user to grant the same privileges to others.	GRANT SELECT ON employees to Raj WITH GRANT OPTION;
Roles	Group privileges for easier management.	GRANT SELECT TO database_db;

LAB EXERCISES:

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.courses TO 'user1'@'localhost';
```

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

Ans.

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

GRANT INSERT ON your_database.courses TO 'user2'@'localhost';
```

10. Transaction Control Language (TCL)

Theory Questions:

1. What is the purpose of the COMMIT and ROLLBACK commands in SQL?

Ans. Transactions improve database consistency and integrity.

1. COMMIT – Saving Changes Permanently:

The COMMIT command saves all the changes made during a transaction permanently in the database.

COMMIT saves changes permanently.

Syntax:

```
COMMIT;
```

Example:

```
BEGIN TRANSACTION;

INSERT INTO employees (id, name, department) VALUES (101, 'raj', 'HR');

COMMIT;
```

2. ROLLBACK – Undoing Changes:

The ROLLBACK command undoes all changes made during the current transaction if they haven't been committed.

ROLLBACK undoes changes made in a transaction.

Syntax:

```
ROLLBACK;
```

Example:

```
BEGIN TRANSACTION;

INSERT INTO employees (id, name, department) VALUES (102, 'Bob',
'Finance');

ROLLBACK;
```

2. Explain how transactions are managed in SQL databases.

Ans.

A transaction in SQL is a sequence of operations performed as a single unit of work.

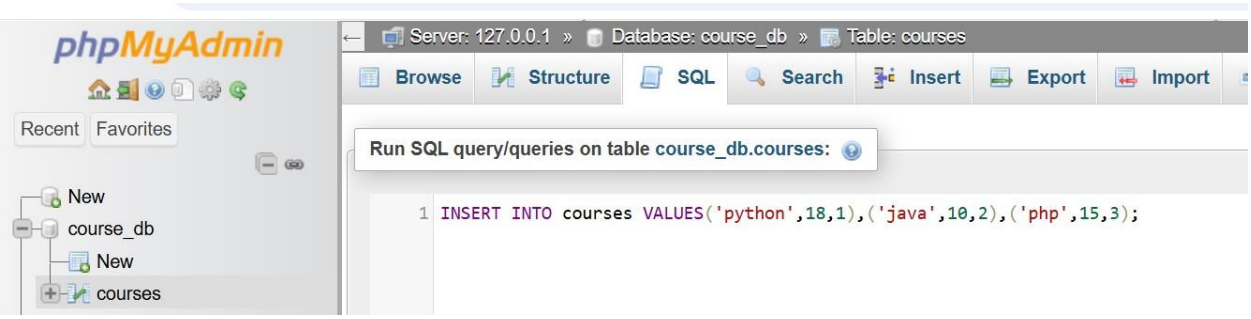
transactions Table:

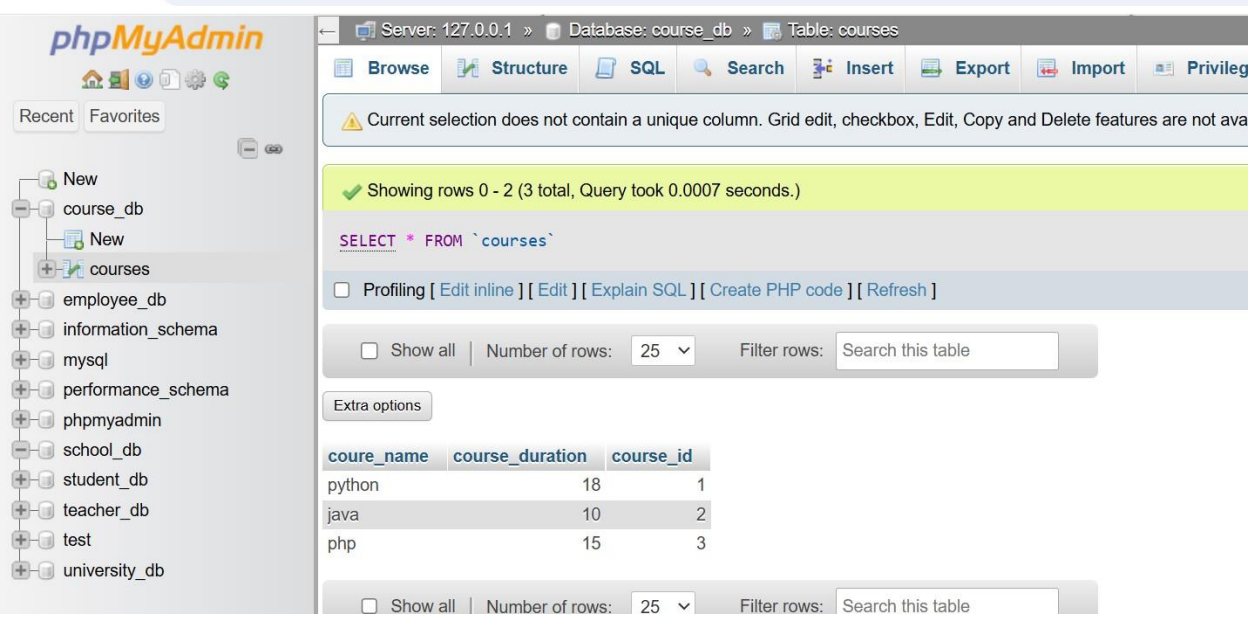
command	purpose
BEGIN TRANSACTION	Starts a new transaction
COMMIT	Saves all changes permanently
ROLLBACK	Undoes all changes in the transaction
SAVEPOINT	Creates a checkpoint to roll back part of a transaction
ROLLBACK TO SAVEPOINT	Rolls back to a specific checkpoint
SET TRANSACTION ISOLATION LEVEL	Controls isolation level for transactions

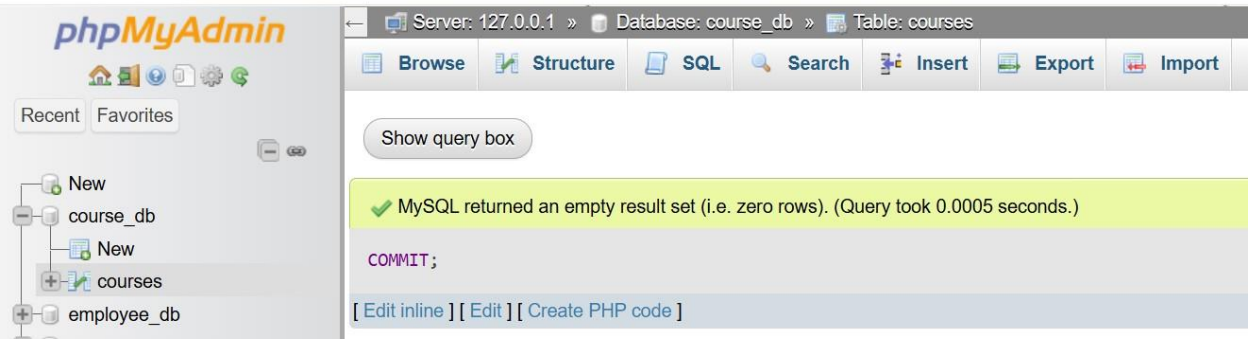
LAB EXERCISES:

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

Ans.

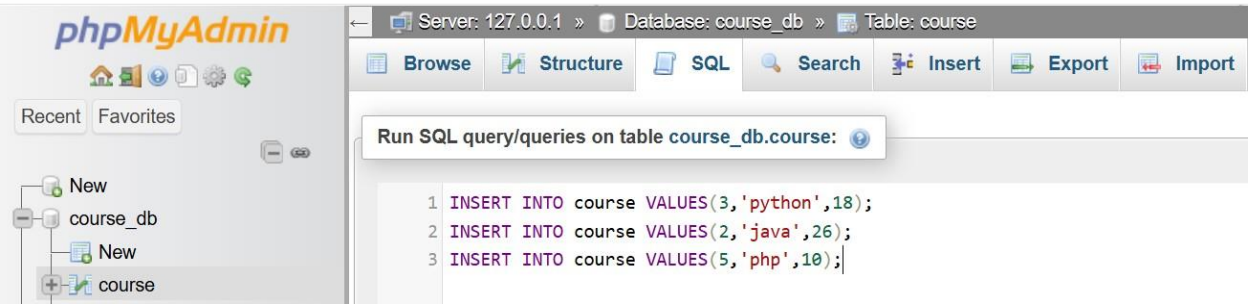


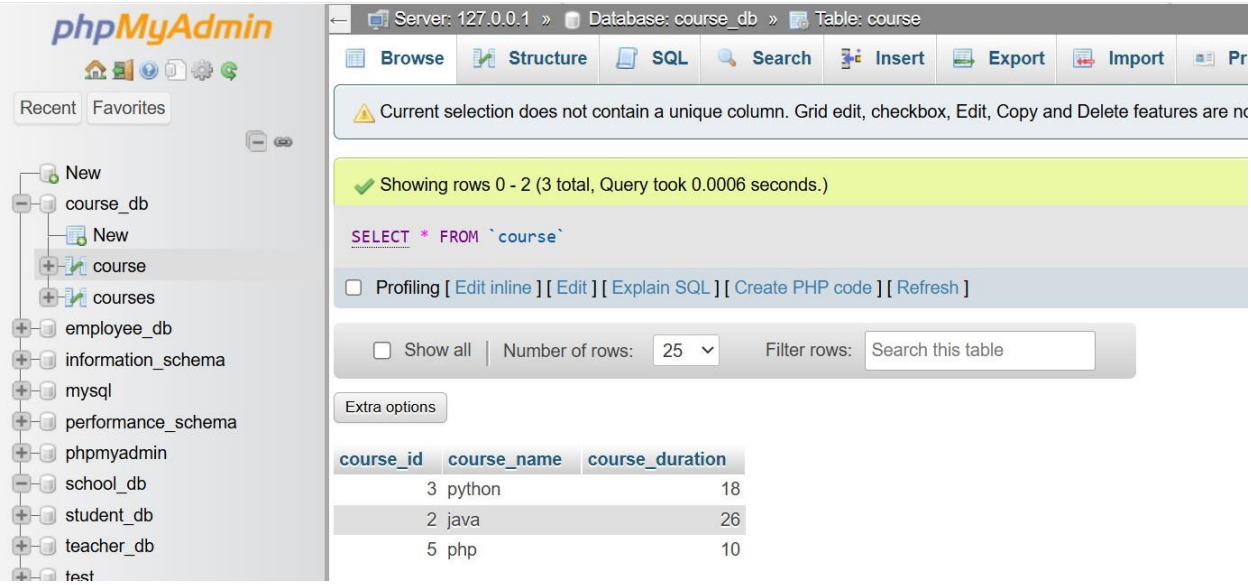


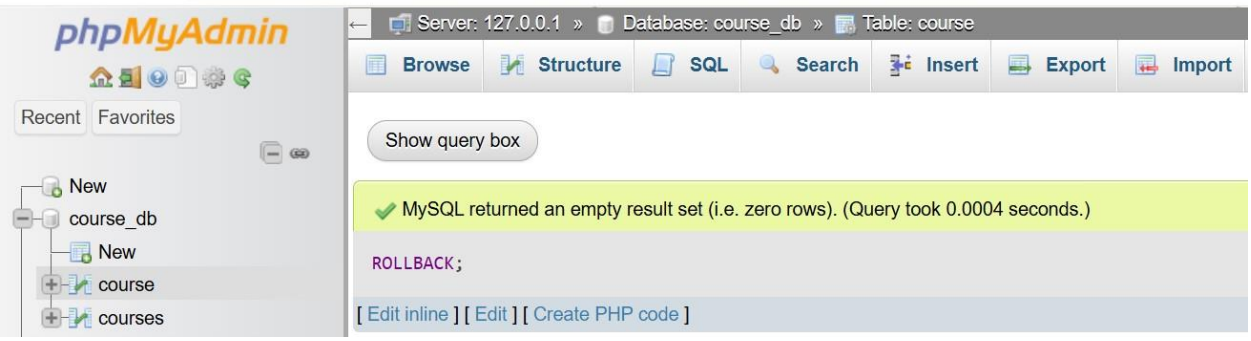


Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

Ans.







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

Ans.

START TRANSACTION; SAVEPOINT before_changes;

INSERT INTO courses VALUES(3, 'PHP', 6), (4, '.NET', 7);

```
UPDATE courses SET course_duration = 10 WHERE course_id = 5;

 ROLLBACK TO SAVEPOINT before_changes;
```

11. SQL Joins

Theory Questions:

1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

Ans.

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

The join keyword merges two or more tables and creates a temporary image of the merged table.

(INNER) JOIN:

Returns records that have matching values in both tables.

LEFT (OUTER) JOIN:

Returns all records from the left table, and the matched records from the right table.

RIGHT (OUTER) JOIN:

Returns all records from the right table, and the matched records from the left table.

FULL (OUTER) JOIN:

Returns all records when there is a match in either the left or right table.

JOIN Differences:

JOIN type	Includes Unmatched Rows?	Matching Rows
INNER JOIN	No	Yes
LEFT JOIN	Left table only	Yes

RIGHT JOIN	Right table only	Yes
FULL OUTER JOIN	Both tables	yes

2. How are joins used to combine data from multiple tables?

Ans.

JOIN is used to combine data from multiple tables based on a common column. This helps retrieve meaningful, structured data by linking related tables.

1. Why Use JOIN?

In relational databases, data is often stored in separate tables to avoid redundancy.

Example:

We have two tables:

orders (Order details)

customers (Customer details)

orders (Order details):

Order_id	Customer_id	Totoal_amount
1	101	500
2	102	400
3	103	300

Customers (Customer details):

Customer_id	Name	City
101	Rajdip	Ahmedabad
102	Raj	Rajkot
104	deep	gandhinagar

(INNER) JOIN:

Returns records that have matching values in both tables.

LEFT (OUTER) JOIN:

Returns all records from the left table, and the matched records from the right table.

RIGHT (OUTER) JOIN:

Returns all records from the right table, and the matched records from the left table.

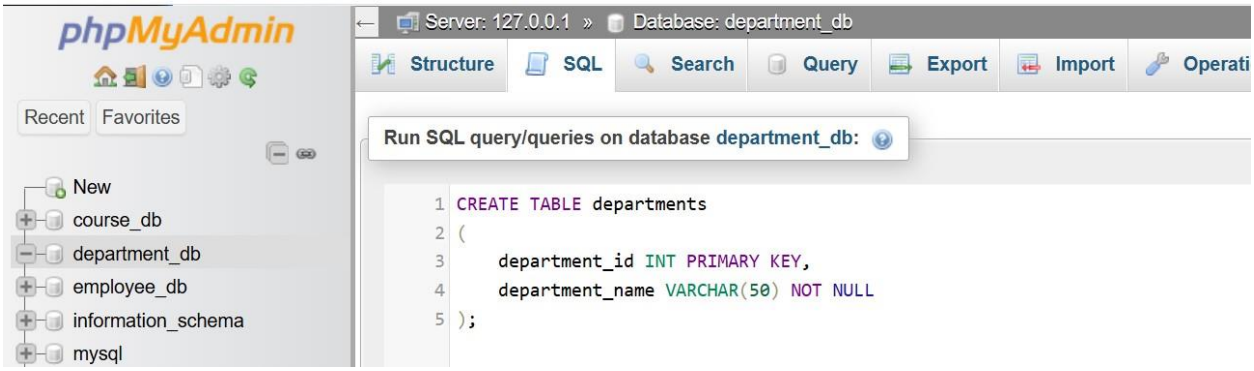
FULL (OUTER) JOIN:

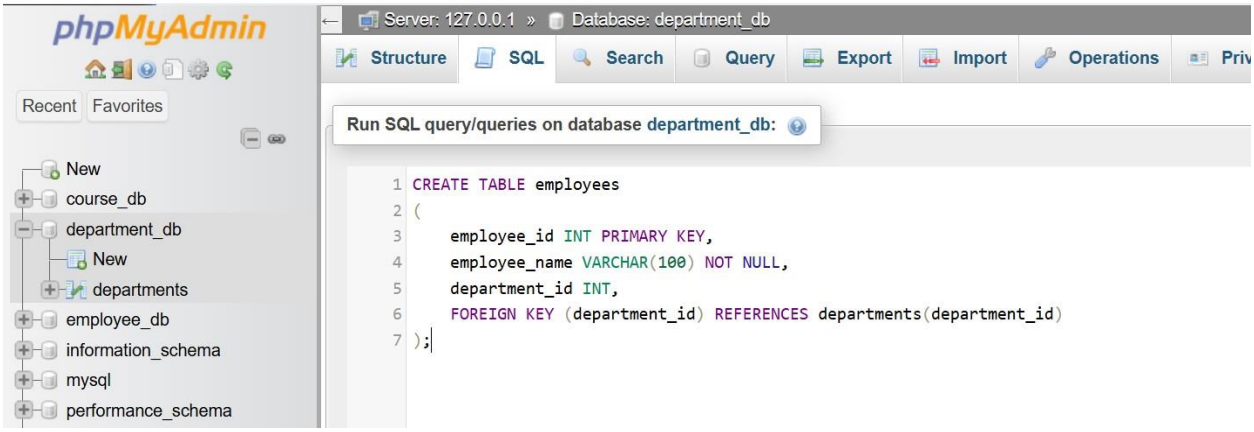
Returns all records when there is a match in either the left or right table.

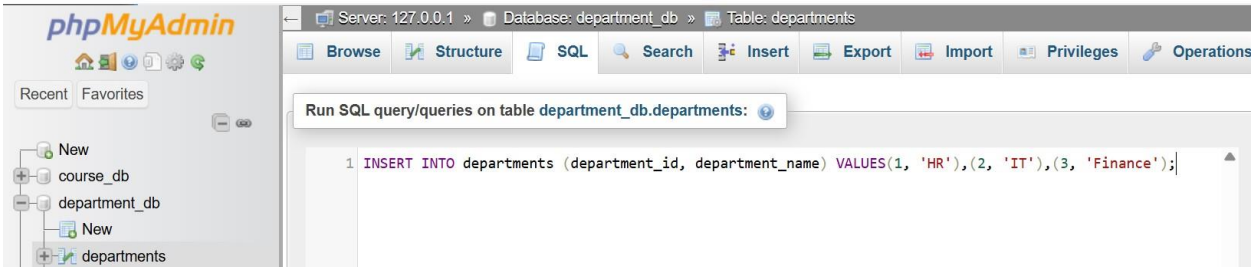
LAB EXERCISES:

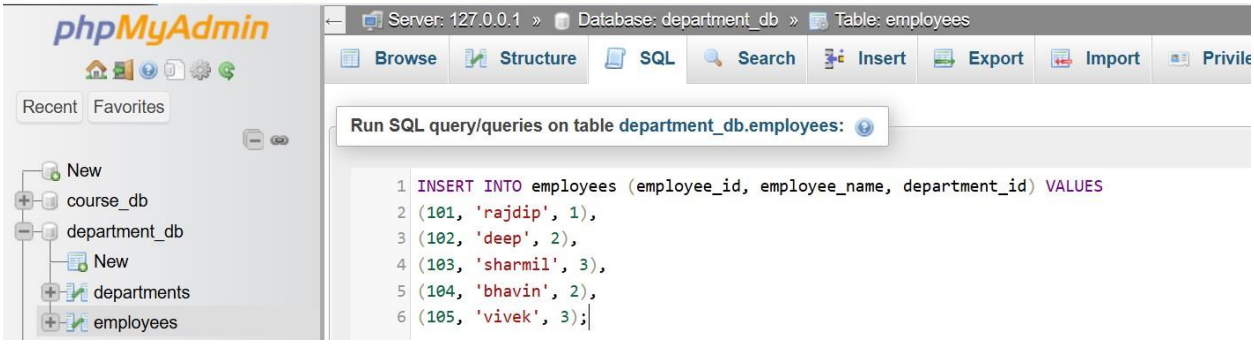
Lab 1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

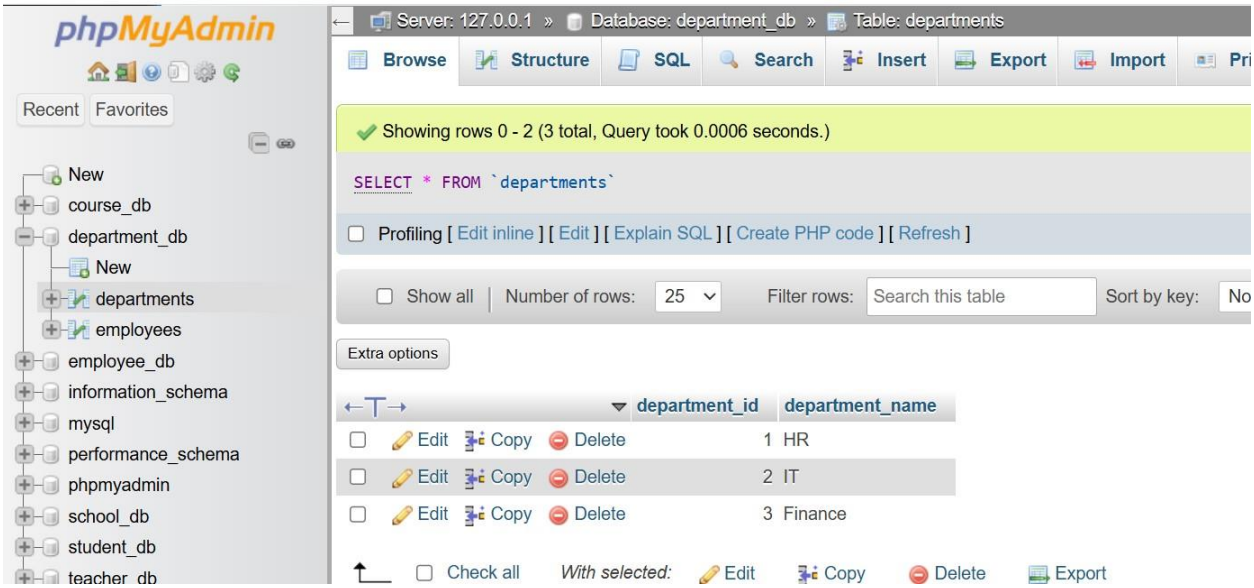
Ans.

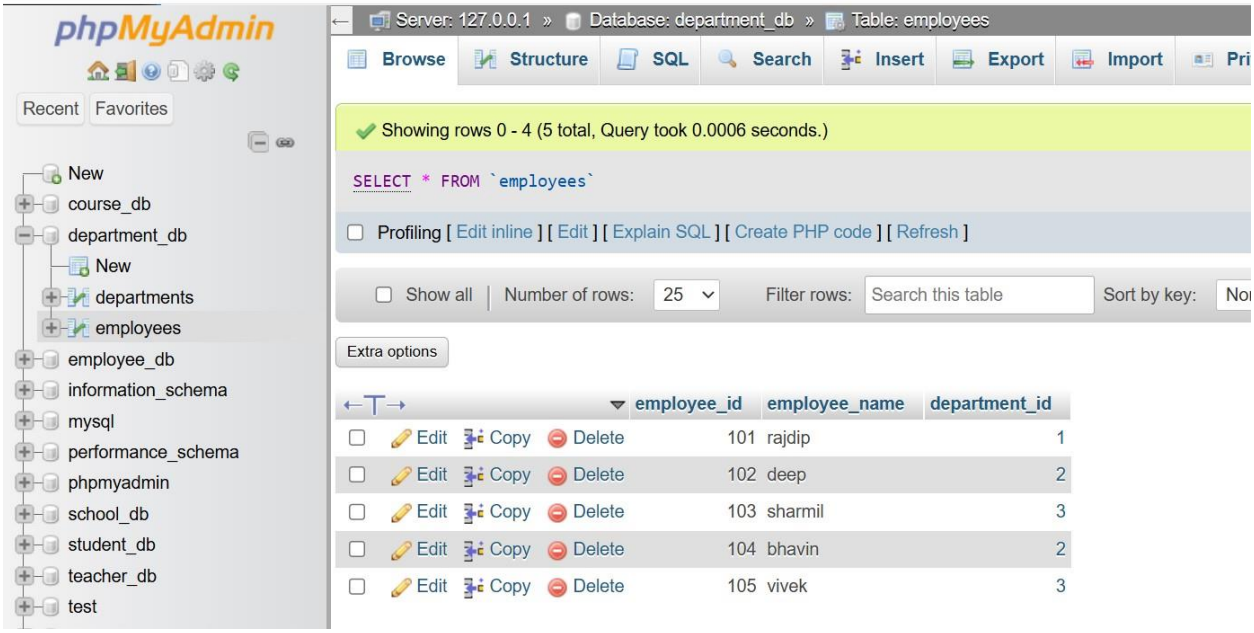


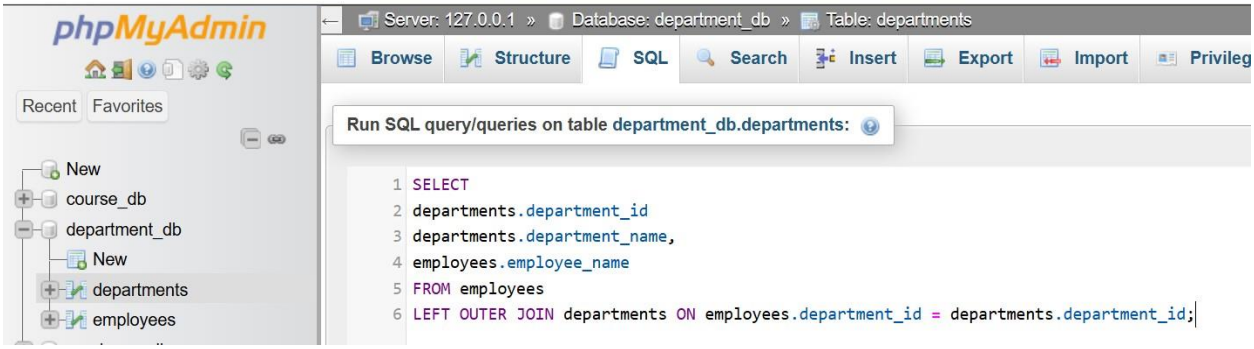


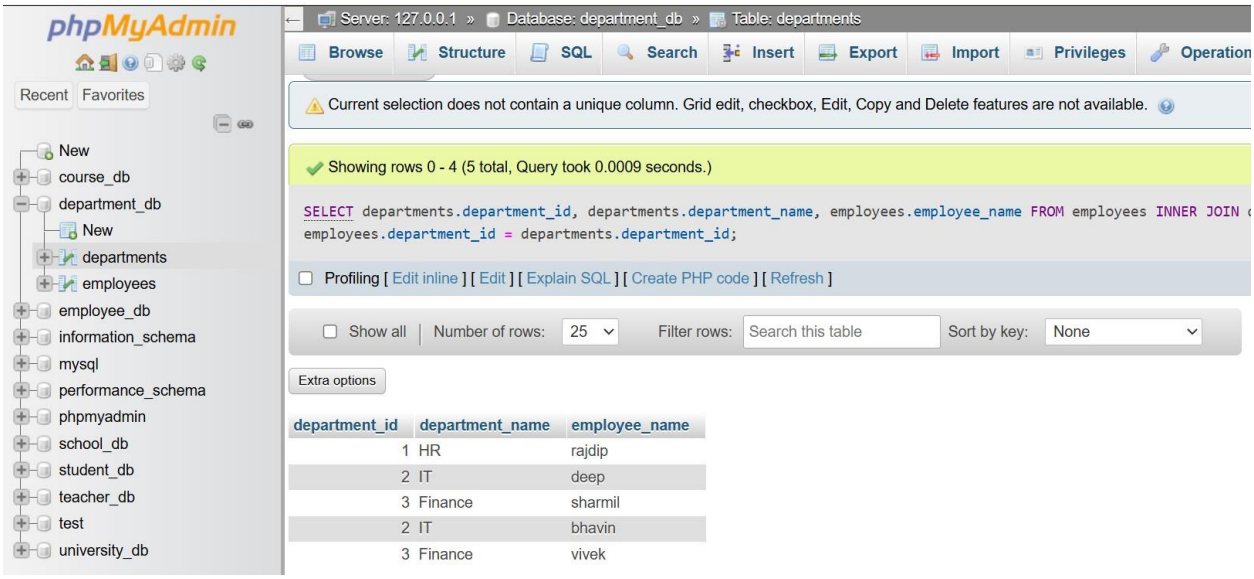

phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
Server: 127.0.0.1 » Database: department_db » Table: departments
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Run SQL query/queries on table department_db.departments:
1 INSERT INTO departments (department_id, department_name) VALUES(1, 'HR'),(2, 'IT'),(3, 'Finance');


phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
employees
Server: 127.0.0.1 » Database: department_db » Table: employees
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table department_db.employees:
1 INSERT INTO employees (employee_id, employee_name, department_id) VALUES
2 (101, 'rajdip', 1),
3 (102, 'deep', 2),
4 (103, 'sharmil', 3),
5 (104, 'bhavin', 2),
6 (105, 'vivek', 3);


phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
employees
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
student_db
teacher_db
Server: 127.0.0.1 » Database: department_db » Table: departments
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)
SELECT * FROM `departments`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key:
Extra options
department_id
department_name
Edit Copy Delete 1 HR
Edit Copy Delete 2 IT
Edit Copy Delete 3 Finance
Check all
With selected:
Edit
Copy
Delete
Export

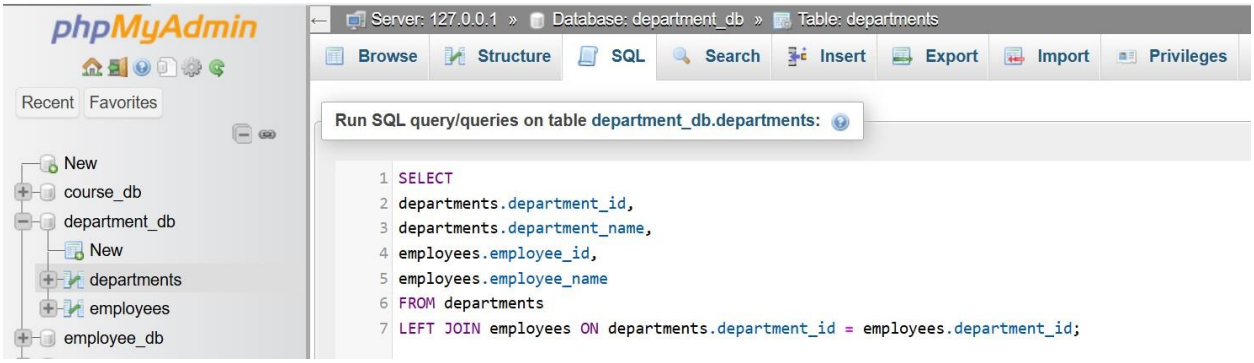

phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
employees
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
student_db
teacher_db
test
Server: 127.0.0.1 » Database: department_db » Table: employees
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 4 (5 total, Query took 0.0006 seconds.)
SELECT * FROM `employees`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Extra options
employee_id
employee_name
department_id
Edit Copy Delete 101 rajdip 1
Edit Copy Delete 102 deep 2
Edit Copy Delete 103 sharmil 3
Edit Copy Delete 104 bhavin 2
Edit Copy Delete 105 vivek 3

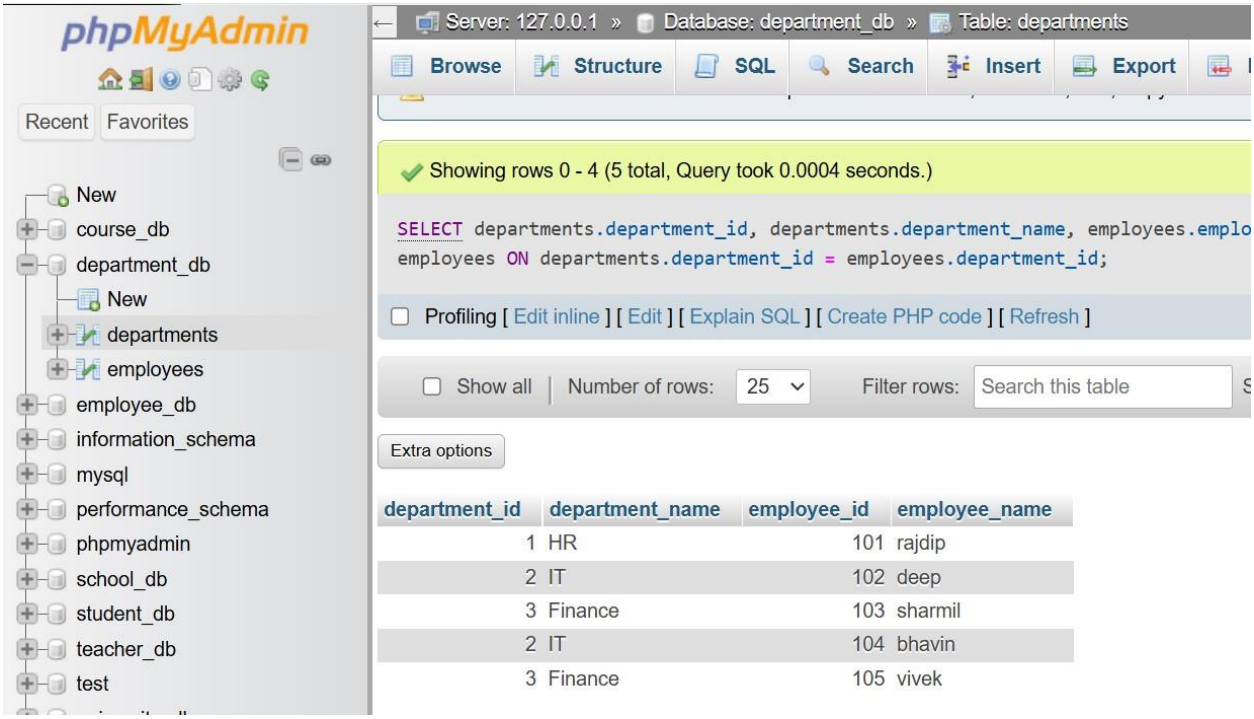

phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
employees
Server: 127.0.0.1 » Database: department_db » Table: departments
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table department_db.departments:
SELECT
departments.department_id
departments.department_name,
employees.employee_name
FROM employees
LEFT OUTER JOIN departments ON employees.department_id = departments.department_id;


phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
employees
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
student_db
teacher_db
test
university_db
Server: 127.0.0.1 » Database: department_db » Table: departments
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
Showing rows 0 - 4 (5 total, Query took 0.0009 seconds.)
SELECT departments.department_id, departments.department_name, employees.employee_name FROM employees INNER JOIN
employees.department_id = departments.department_id;
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
Extra options
department_id
department_name
employee_name
1 HR rajdip
2 IT deep
3 Finance sharmil
2 IT bhavin
3 Finance vivek

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

Ans.





12. SQL Group By

Theory Questions:

1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?

Ans.

GROUP BY Clause:

Groups those rows that have the same values into summary rows.

It collects data from multiple records and groups the results by one or more columns.

```
SELECT department, COUNT(name)
FROM employees
GROUP BY department;
```

It is often used with aggregate functions to perform calculations on each group.

Basic syntax:

```
SELECT column_name, AGGREGATE_FUNCTION(column_name)
FROM table_name
GROUP BY column_name;
```

Aggregate Function	Description
COUNT(*)	Counts the number of rows in each group
SUM(column)	Calculates the total sum of values in each group
AVG(column)	Finds the average value in each group
MAX(column)	Finds the highest value in each group
MIN(column)	Finds the lowest value in each group

2. Explain the difference between GROUP BY and ORDER BY.

Ans.

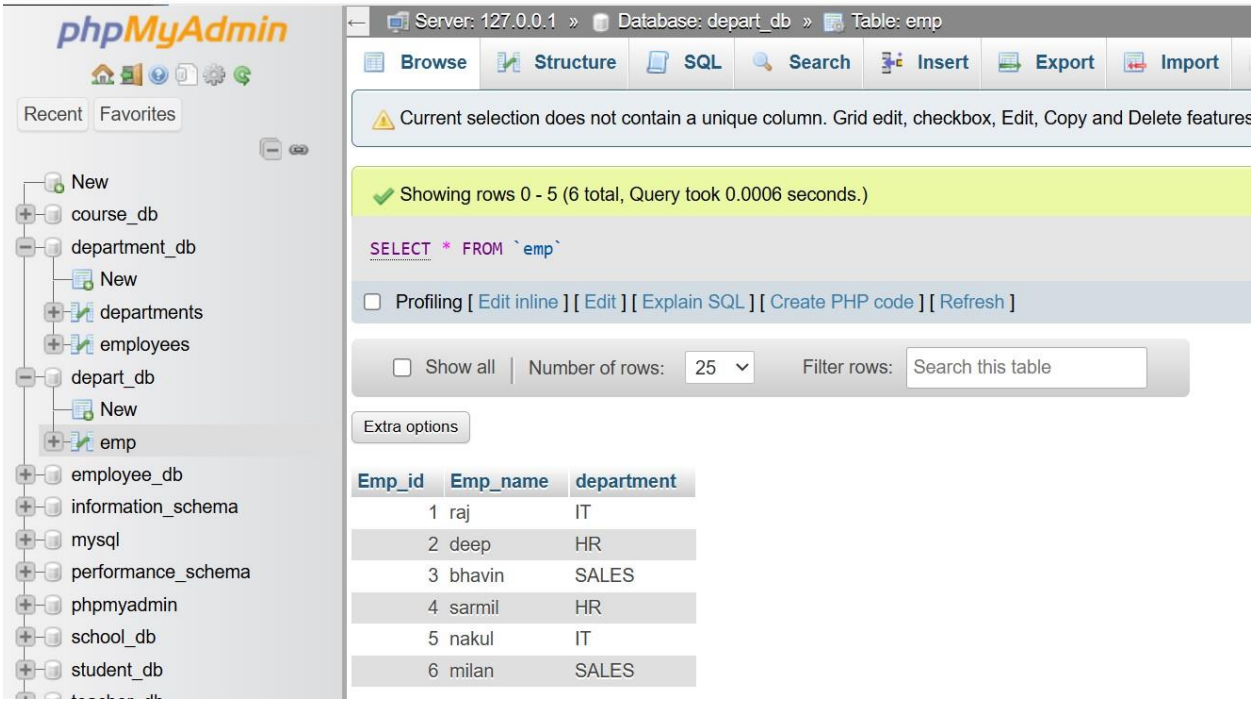
feature	GROUP BY	ORDER BY
Purpose	Groups rows based on shared values in a column	Sorts rows in ascending (ASC) or descending (DESC) order

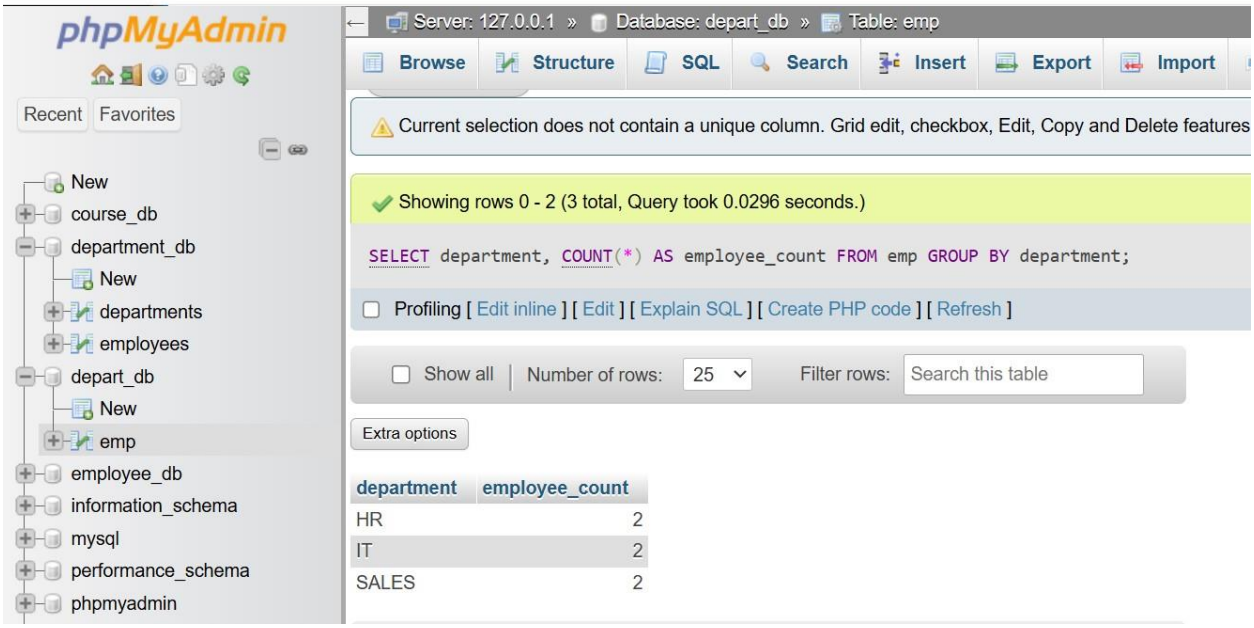
Usage	Used with aggregate functions (SUM(), COUNT(), AVG(), etc.)	Used to sort final query results
Result	Returns one row per group	Returns all rows sorted
Aggregate Function	Yes, when selecting non-grouped columns	No, can sort without aggregation
Filters Applied	Can use HAVING to filter groups	Can use WHERE to filter rows before sorting

LAB EXERCISES:

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

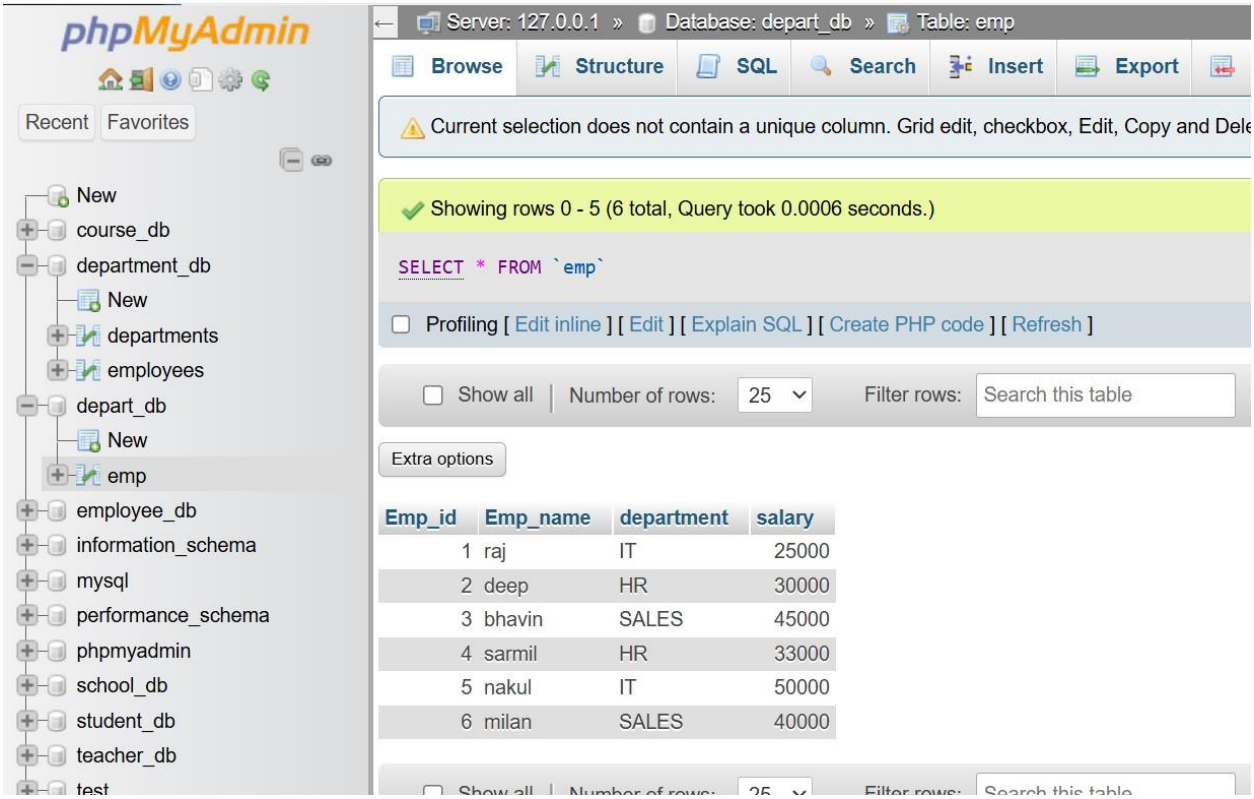
Ans.

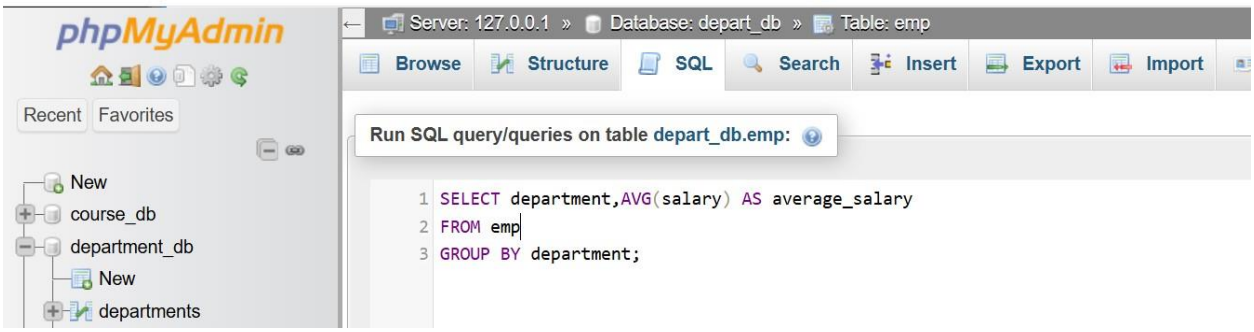


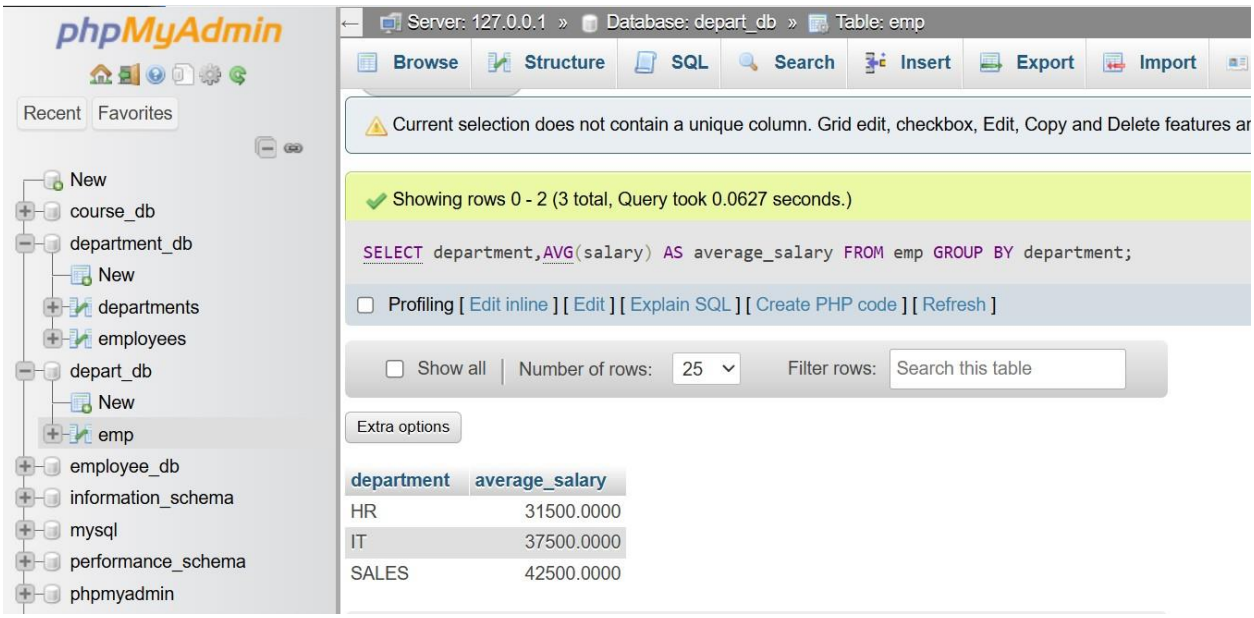


Lab 2: Use the AVG aggregate function to find the average salary of employees in each department.

Ans.


phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
employees
depart_db
New
emp
employee_db
information_schema
mysql
performance_schema
phpmyadmin
school_db
student_db
teacher_db
test
Server: 127.0.0.1 » Database: depart_db » Table: emp
Browse
Structure
SQL
Search
Insert
Export
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Dele
Showing rows 0 - 5 (6 total, Query took 0.0006 seconds.)
SELECT * FROM `emp`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows:
25
Filter rows:
Search this table
Extra options
Emp_id
Emp_name
department
salary
1 raj IT 25000
2 deep HR 30000
3 bhavin SALES 45000
4 sarmil HR 33000
5 nakul IT 50000
6 milan SALES 40000


phpMyAdmin
Recent
Favorites
New
course_db
department_db
New
departments
Server: 127.0.0.1 » Database: depart_db » Table: emp
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table depart_db.emp:
1 SELECT department,AVG(salary) AS average_salary
2 FROM emp
3 GROUP BY department;



13. SQL Stored Procedure

Theory Questions:

1. What is a stored procedure in SQL, and how does it differ from a standard SQL query?

Ans.

What is a Stored Procedure?

It can contain multiple SQL statements, including SELECT, INSERT, UPDATE, DELETE, and even conditional logic.

Stored procedures are faster, reusable, and more secure than standard SQL queries.

Syntax:

```
CREATE PROCEDURE procedure_name
AS
BEGIN
    -- SQL statements
```

```
END;
```

Standard SQL Query (One-Time Execution):

Standard SQL queries are best for quick one-time operations.

Example:

```
SELECT employee_id, name, salary
FROM employees
WHERE department = 'HR';
```

Runs once and retrieves HR employees.

Must be rewritten to query a different department.

2. Explain the advantages of using stored procedures.

Ans.

It can contain multiple SQL statements, including SELECT, INSERT, UPDATE, DELETE, and even conditional logic.

Stored procedures are faster, reusable, and more secure than standard SQL queries.

Syntax:

```
CREATE PROCEDURE procedure_name
AS
BEGIN
    -- SQL statements
END;
```

Advantages:

Performance: Faster execution due to pre-compilation.

Reusability: Avoids rewriting queries, making code cleaner.

Security: Limits table access, preventing unauthorized data exposure.

Reduced Traffic: Sends only procedure calls, reducing network load.

Transaction Control: Ensures data consistency with COMMIT and ROLLBACK.

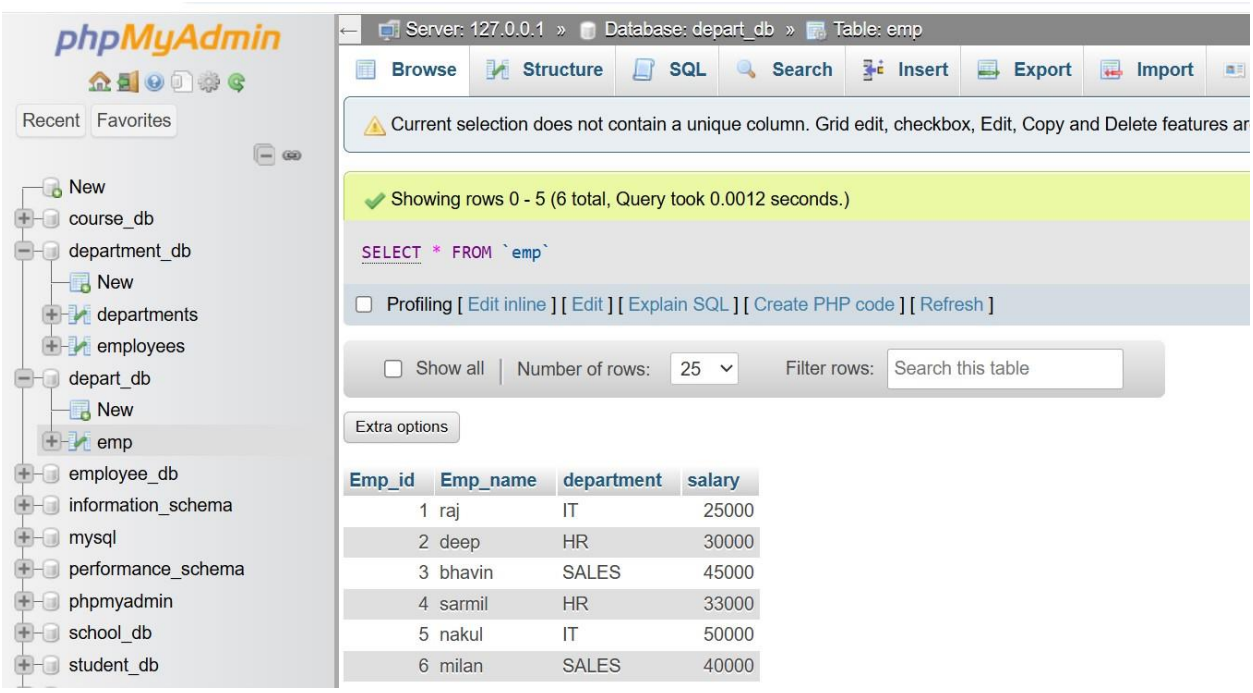
Modular Code: Organizes logic inside the database.

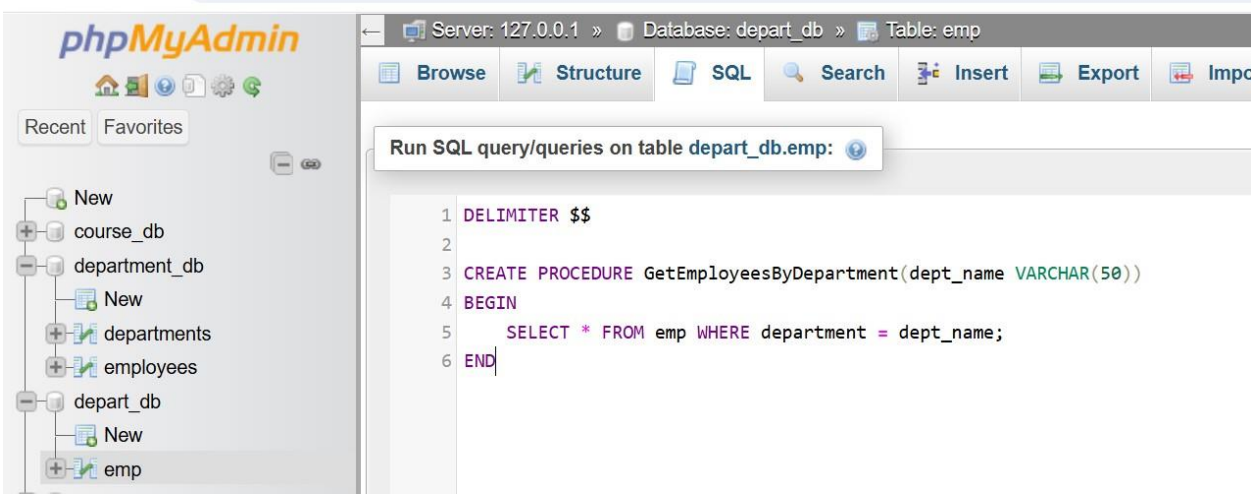
Scalability: Handles large workloads efficiently.

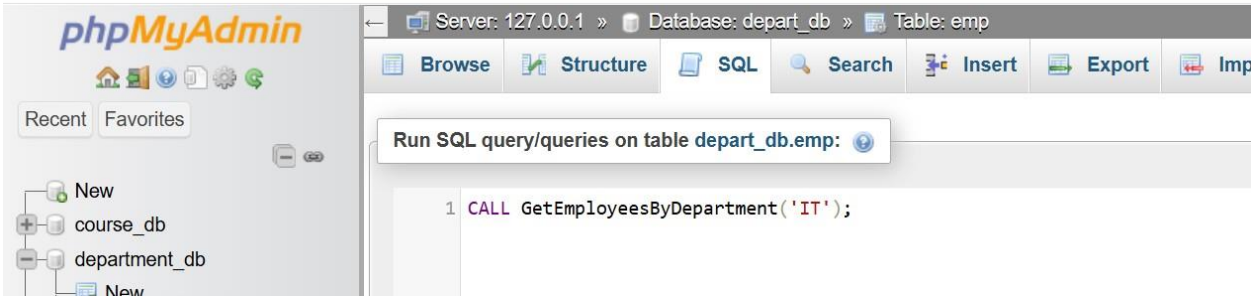
LAB EXERCISES:

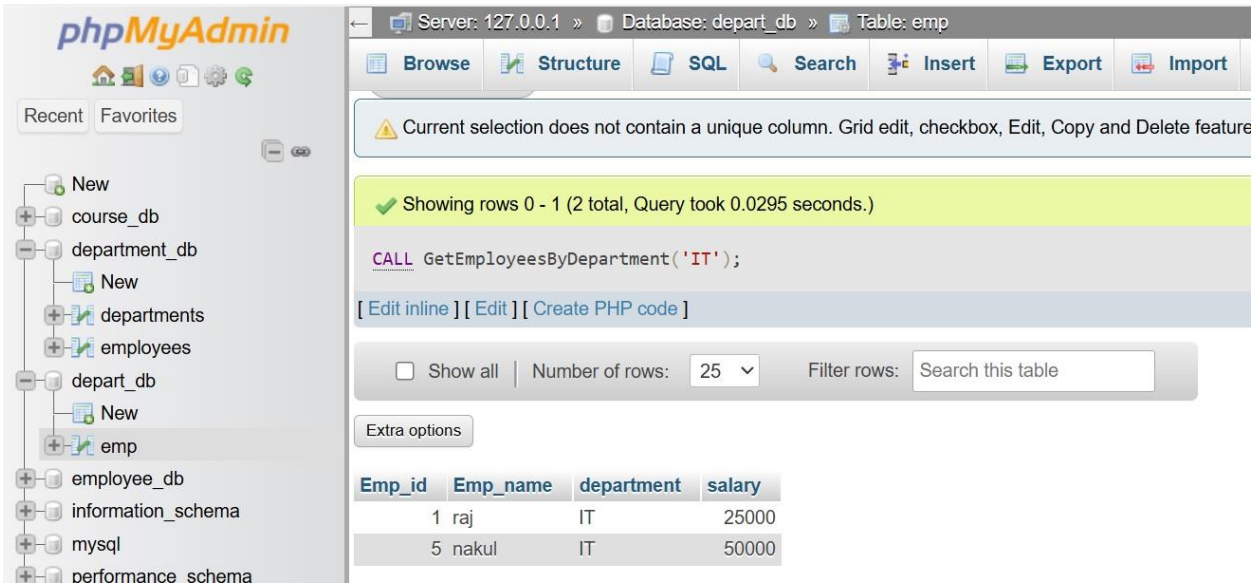
Lab 1: Write a stored procedure to retrieve all employees from the employee's table based on department.

Ans.



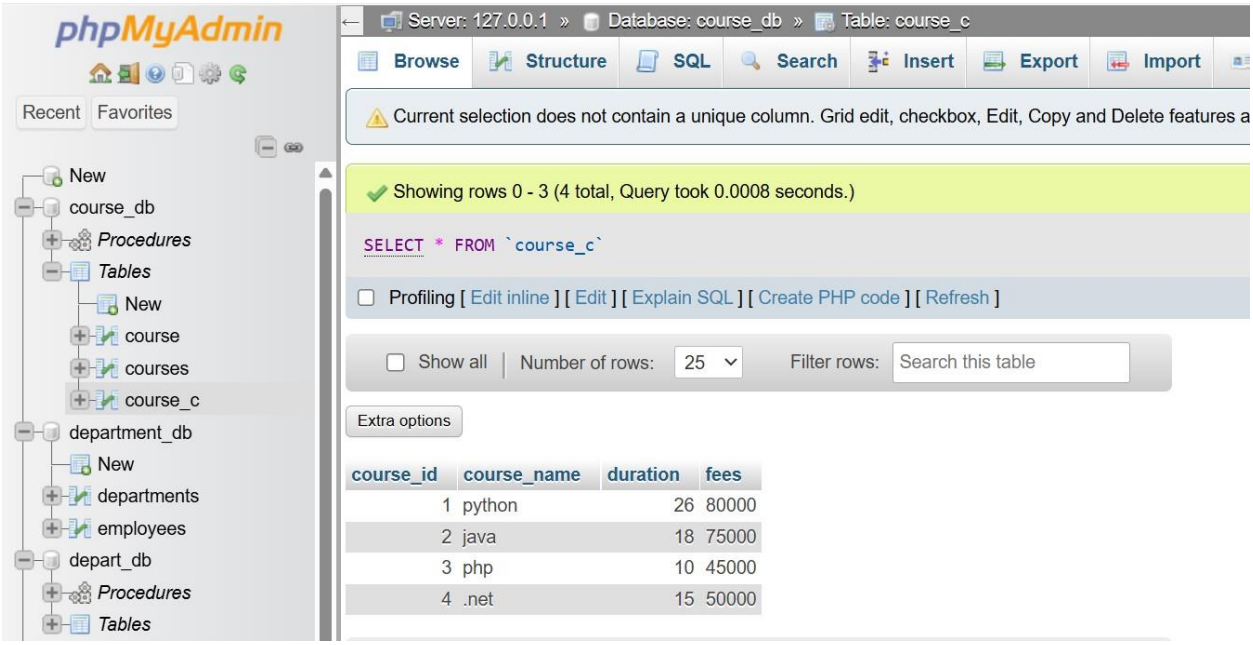


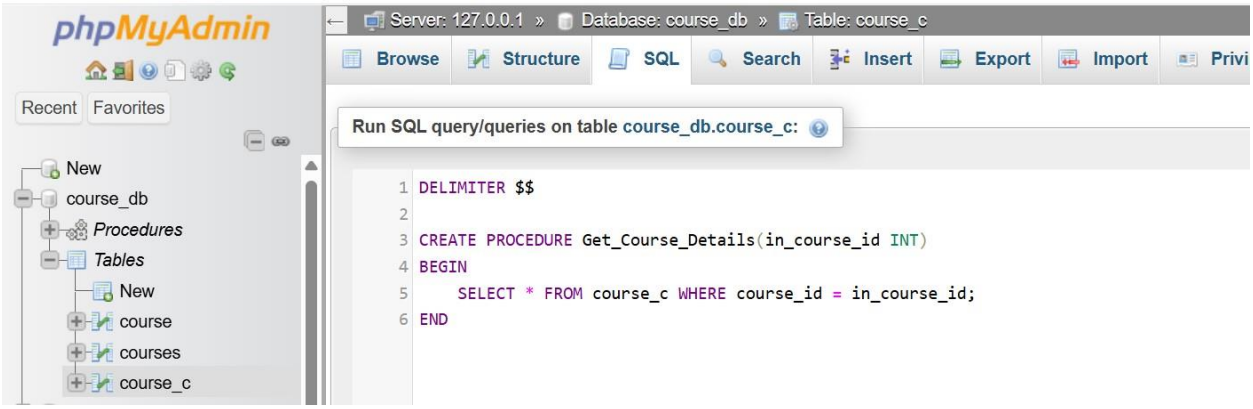


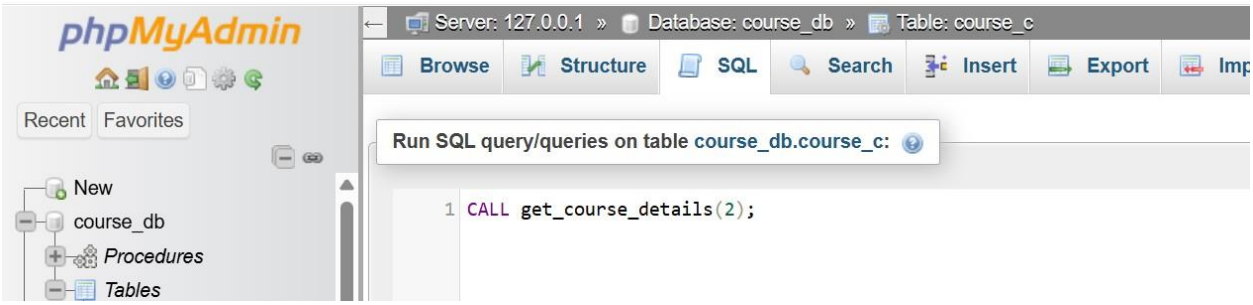


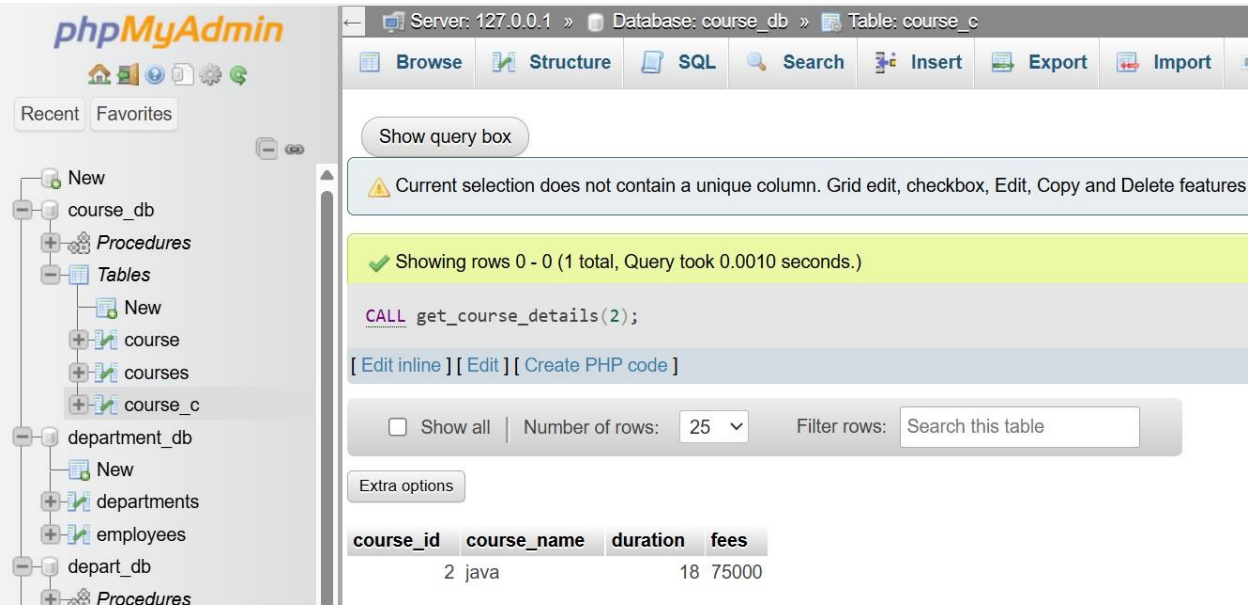
Lab 2: Write a stored procedure that accepts course_id as input and returns the course details.

Ans.


phpMyAdmin
Recent
Favorites
New
course_db
Procedures
Tables
New
course
courses
course_c
department_db
New
departments
employees
depart_db
Procedures
Tables
Server: 127.0.0.1 » Database: course_db » Table: course_c
Browse
Structure
SQL
Search
Insert
Export
Import
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features a
Showing rows 0 - 3 (4 total, Query took 0.0008 seconds.)
SELECT * FROM `course_c`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Extra options
course_id course_name duration fees
1 python 26 80000
2 java 18 75000
3 php 10 45000
4 .net 15 50000


phpMyAdmin
Recent
Favorites
New
course_db
Procedures
Tables
New
course
courses
course_c
Server: 127.0.0.1 » Database: course_db » Table: course_c
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table course_db.course_c:
DELIMITER $$
CREATE PROCEDURE Get_Course_Details(in_course_id INT)
BEGIN
SELECT * FROM course_c WHERE course_id = in_course_id;
END


phpMyAdmin
Recent
Favorites
New
course_db
Procedures
Tables
Server: 127.0.0.1 » Database: course_db » Table: course_c
Browse
Structure
SQL
Search
Insert
Export
Run SQL query/queries on table course_db.course_c:
CALL get_course_details(2);



14. SQL View

Theory Questions:

1. What is a view in SQL, and how is it different from a table?

Ans.

What is a View?

A view is a virtual table in SQL that is created using a SELECT query.

It does not store data physically but dynamically retrieves it from one or more tables when accessed.

feature	View	Table
Definition	A stored SQL query that acts like a virtual table	A physical storage structure that holds actual data
Storage	Does not store data; only the query is stored	Stores data physically in the database
Data Updates	Can be updated	Can always be updated

Performance	Can be slower as it executes the query each time	Faster as data is pre-stored
Use Case	Used for simplifying complex queries, security, and abstraction	Used for storing and managing data

2. Explain the advantages of using views in SQL databases.

Ans.

Views in SQL offer several benefits, making them a powerful tool for data security, simplification, and reusability.

Advantages of Using Views in SQL Databases:

Security & Access Control.

Simplifies Complex Queries.

Improves Maintainability.

Saves Storage Space.

Enhances Read Performance.

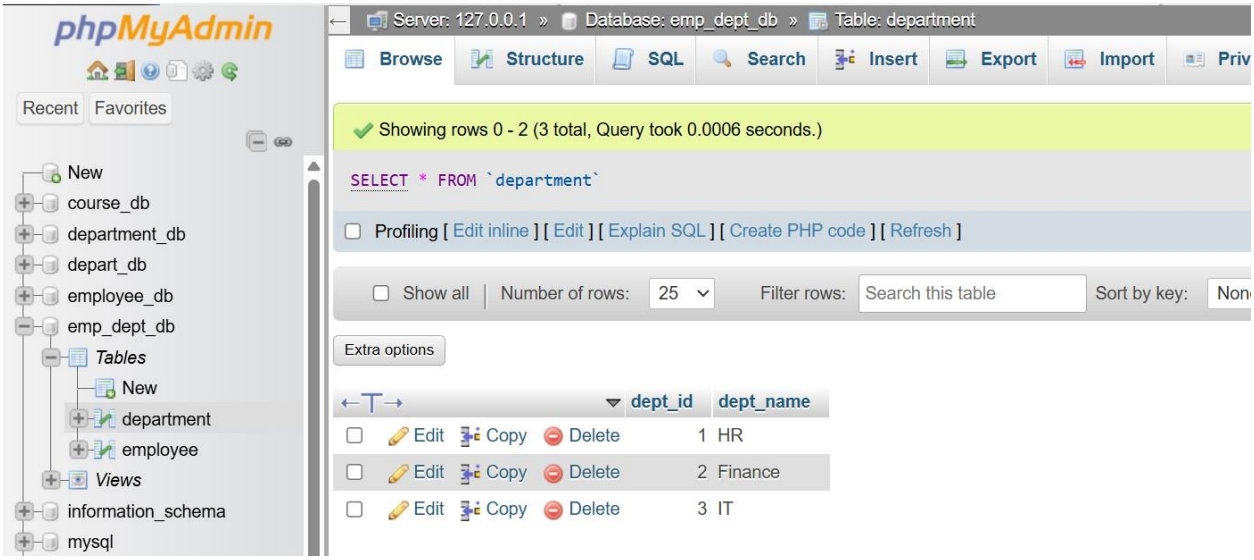
Provides Logical Independence.

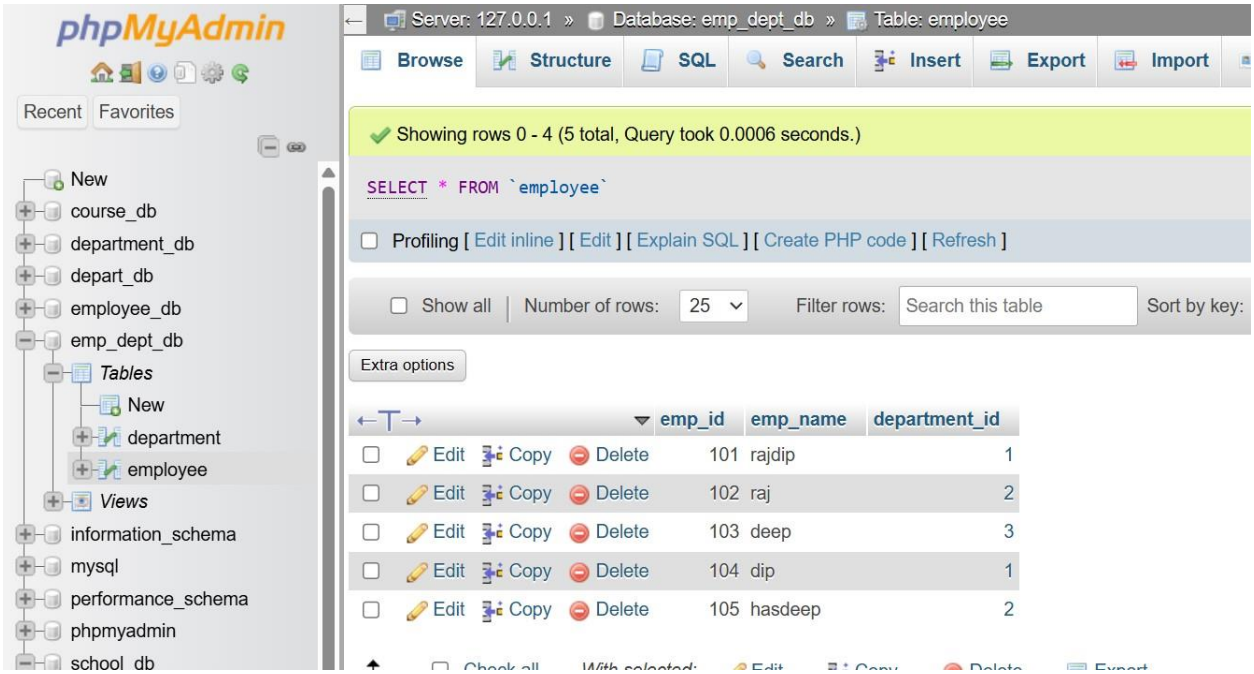
Supports Aggregation & Reporting.

LAB EXERCISES:

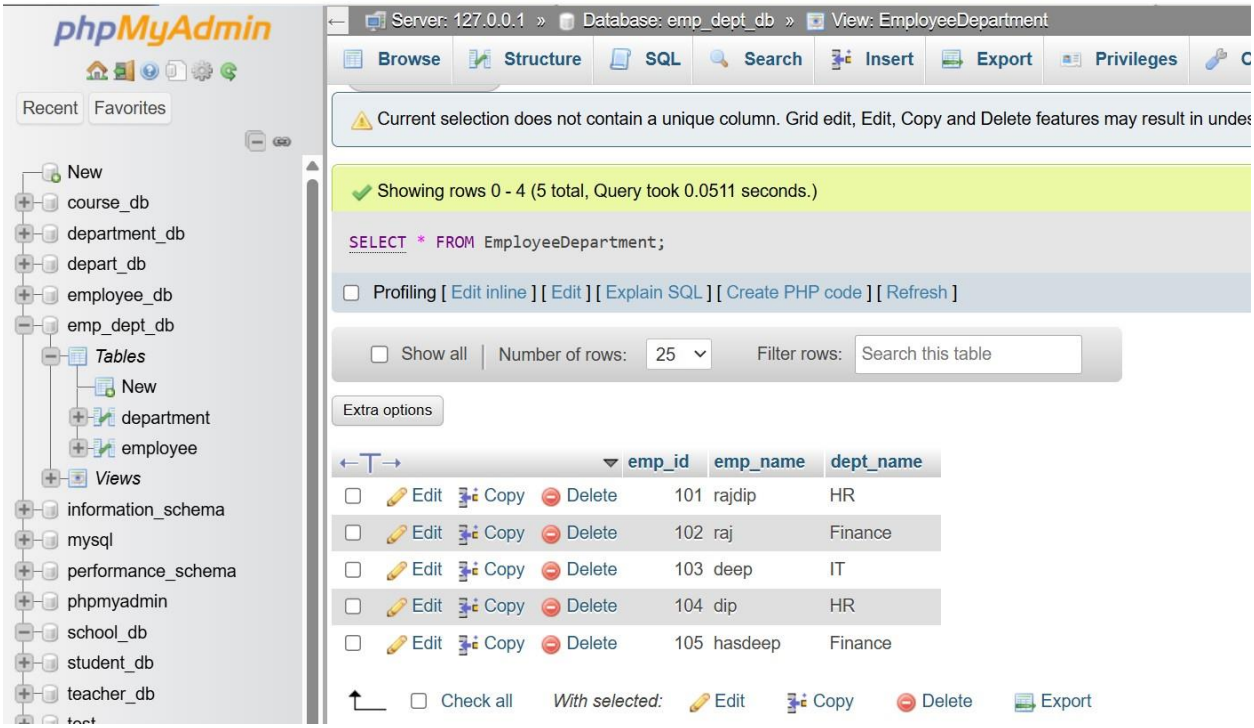
Lab 1: Create a view to show all employees along with their department names.

Ans.


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee_db
emp_dept_db
Tables
New
department
employee
Views
information_schema
mysql
Server: 127.0.0.1 » Database: emp_dept_db » Table: department
Browse
Structure
SQL
Search
Insert
Export
Import
Priv
Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)
SELECT * FROM `department`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key:
Extra options
dept_id
dept_name
Edit Copy Delete 1 HR
Edit Copy Delete 2 Finance
Edit Copy Delete 3 IT

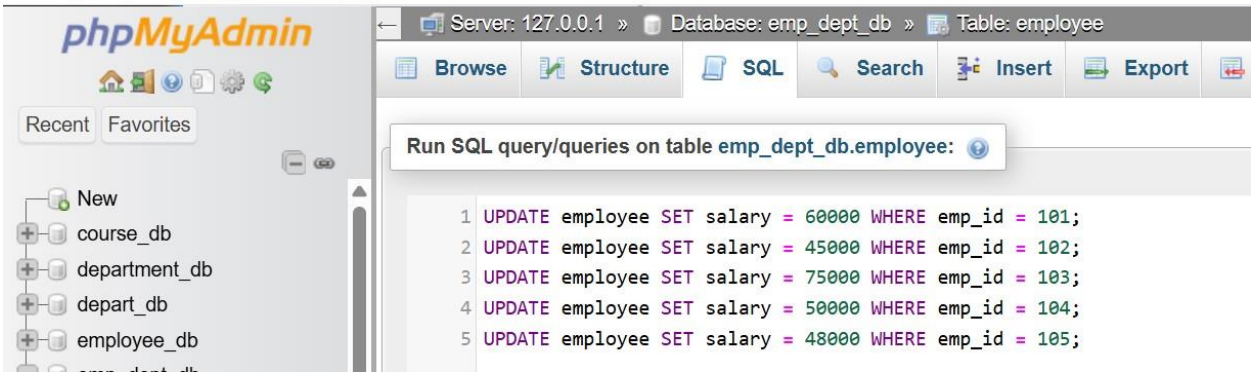

phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee_db
emp_dept_db
Tables
New
department
employee
Views
information_schema
mysql
performance_schema
phpmyadmin
school_db
Server: 127.0.0.1 » Database: emp_dept_db » Table: employee
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 4 (5 total, Query took 0.0006 seconds.)
SELECT * FROM `employee`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key:
Extra options
emp_id
emp_name
department_id
Edit Copy Delete 101 rajdip 1
Edit Copy Delete 102 raj 2
Edit Copy Delete 103 deep 3
Edit Copy Delete 104 dip 1
Edit Copy Delete 105 hasdeep 2

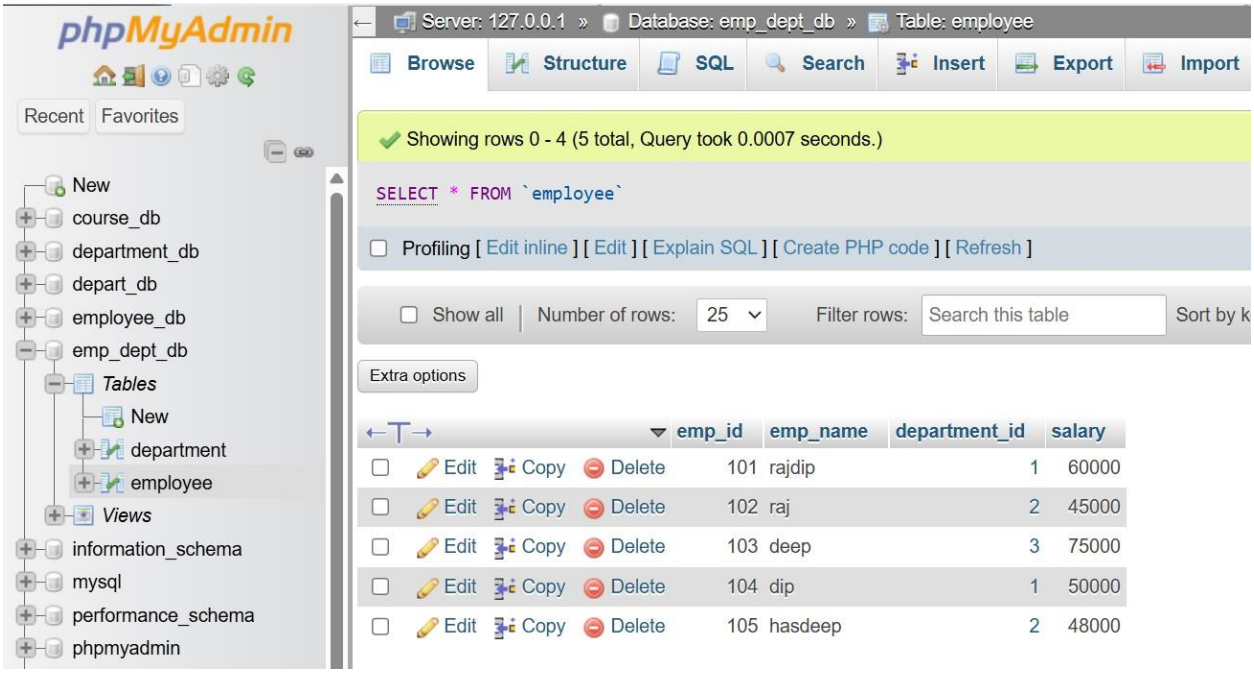

phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee_db
emp_dept_db
New
department
employee
Server: 127.0.0.1 » Database: emp_dept_db » Table: department
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Run SQL query/queries on table emp_dept_db.department:
CREATE VIEW EmployeeDepartment
AS SELECT employee.emp_id,
employee.emp_name,
department.dept_name
FROM employee JOIN department ON employee.department_id = department.dept_id;



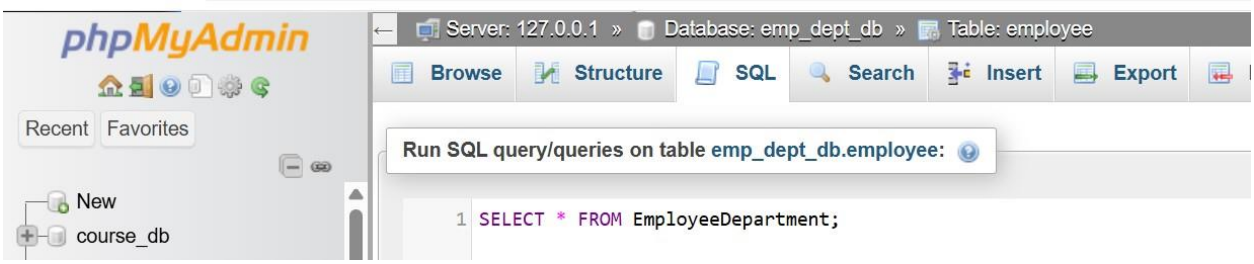
Lab 2: Modify the view to exclude employees whose salaries are below $50,000.

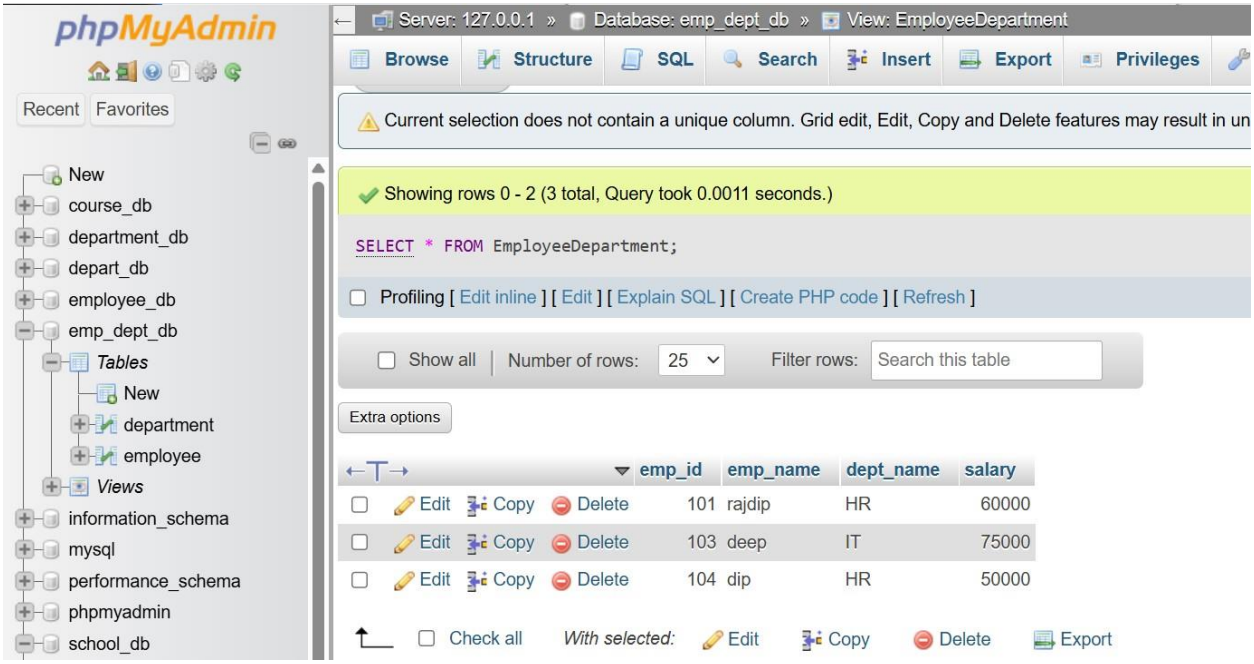
Ans.


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee_db
Server: 127.0.0.1 » Database: emp_dept_db » Table: employee
Browse
Structure
SQL
Search
Insert
Export
Run SQL query/queries on table emp_dept_db.employee:
1 UPDATE employee SET salary = 60000 WHERE emp_id = 101;
2 UPDATE employee SET salary = 45000 WHERE emp_id = 102;
3 UPDATE employee SET salary = 75000 WHERE emp_id = 103;
4 UPDATE employee SET salary = 50000 WHERE emp_id = 104;
5 UPDATE employee SET salary = 48000 WHERE emp_id = 105;


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee_db
emp_dept_db
Tables
New
department
employee
Views
information_schema
mysql
performance_schema
phpmyadmin
Server: 127.0.0.1 » Database: emp_dept_db » Table: employee
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 4 (5 total, Query took 0.0007 seconds.)
SELECT * FROM `employee`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by k
Extra options
emp_id emp_name department_id salary
Edit Copy Delete 101 rajdip 1 60000
Edit Copy Delete 102 raj 2 45000
Edit Copy Delete 103 deep 3 75000
Edit Copy Delete 104 dip 1 50000
Edit Copy Delete 105 hasdeep 2 48000


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee_db
emp_dept_db
Tables
New
department
employee
Views
Server: 127.0.0.1 » Database: emp_dept_db » Table: employee
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table emp_dept_db.employee:
1 CREATE OR REPLACE VIEW EmployeeDepartment
2 AS SELECT employee.emp_id,
3 employee.emp_name,
4 department.dept_name,
5 employee.salary
6 FROM employee JOIN department ON employee.department_id = department.dept_id
7 WHERE employee.salary >= 50000;





15. SQL Triggers

Theory Questions:

1. What is a trigger in SQL? Describe its types and when they are used.

Ans.

What is a Trigger in SQL?

A trigger is a special type of stored procedure in SQL that automatically executes in response to specific events on a table (such as INSERT, UPDATE, or DELETE).

1. BEFORE Trigger (Pre-Trigger):

Executes before the event (INSERT, UPDATE, DELETE) happens.
Used to validate data before modifying the table.

2. AFTER Trigger (Post-Trigger):

Executes after the event (INSERT, UPDATE, DELETE) occurs.
Used to log changes, update related tables, or maintain audit records.

3. INSTEAD OF Trigger:

Used on views where INSERT, UPDATE, or DELETE operations are not directly possible.
Allows updates on virtual tables (views) by redirecting operations to base tables.

2. Explain the difference between INSERT, UPDATE, and DELETE triggers.

Ans.

1. INSERT Trigger:

Fires when a new record is inserted into a table.
Used to validate data, log new entries, or modify inserted values before they are saved.

INSERT triggers handle new data entries.

2. UPDATE Trigger:

Fires when an existing record is updated in a table.
Used to track changes, enforce constraints, or update related tables.

UPDATE triggers track modifications to existing records.

3. DELETE Trigger:

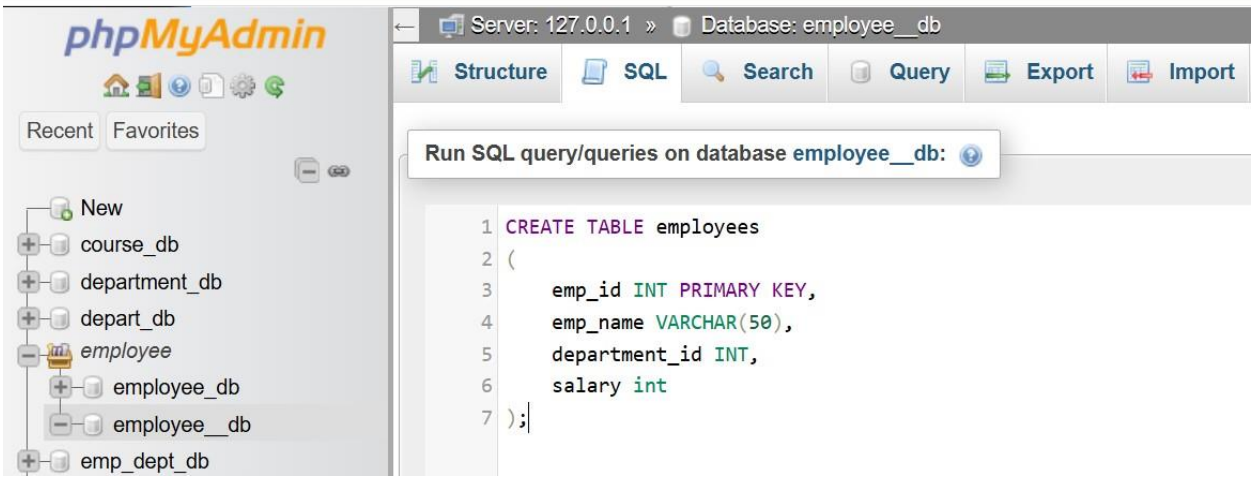
Fires when a record is deleted from a table.
Used to prevent accidental deletions, maintain audit logs, or enforce business rules.

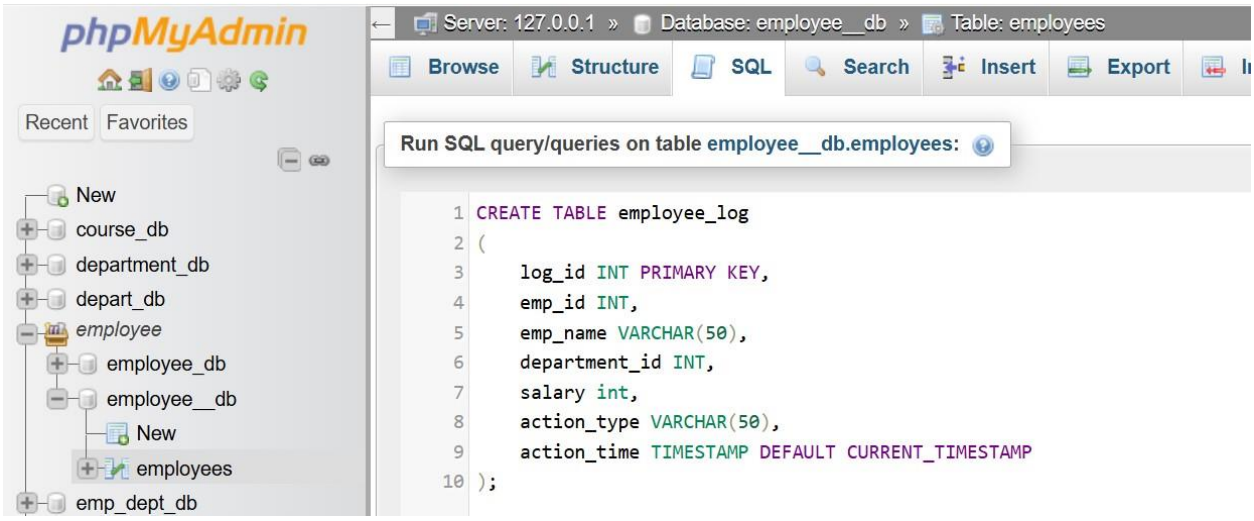
DELETE triggers manage data removal, preventing accidental loss.

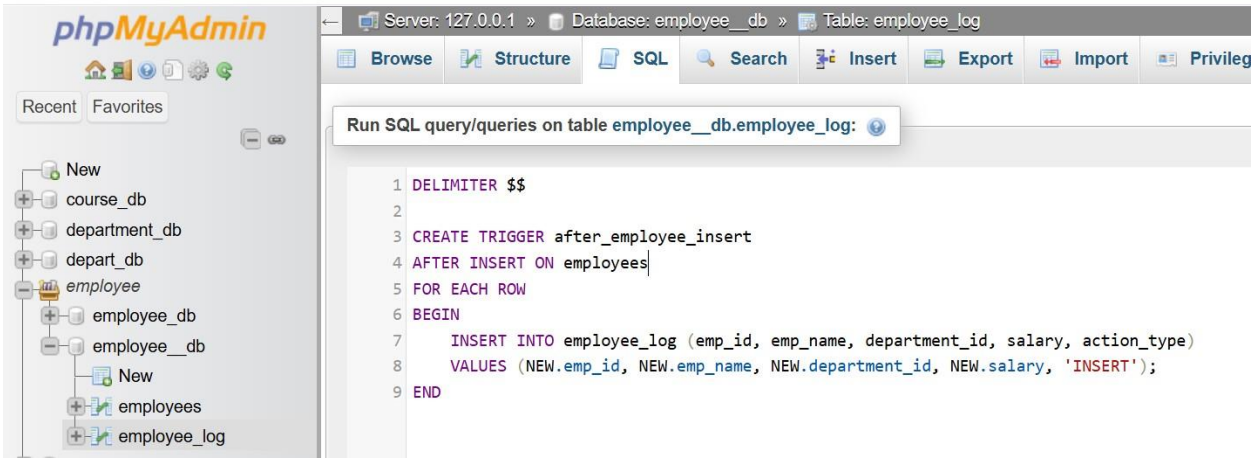
LAB EXERCISES:

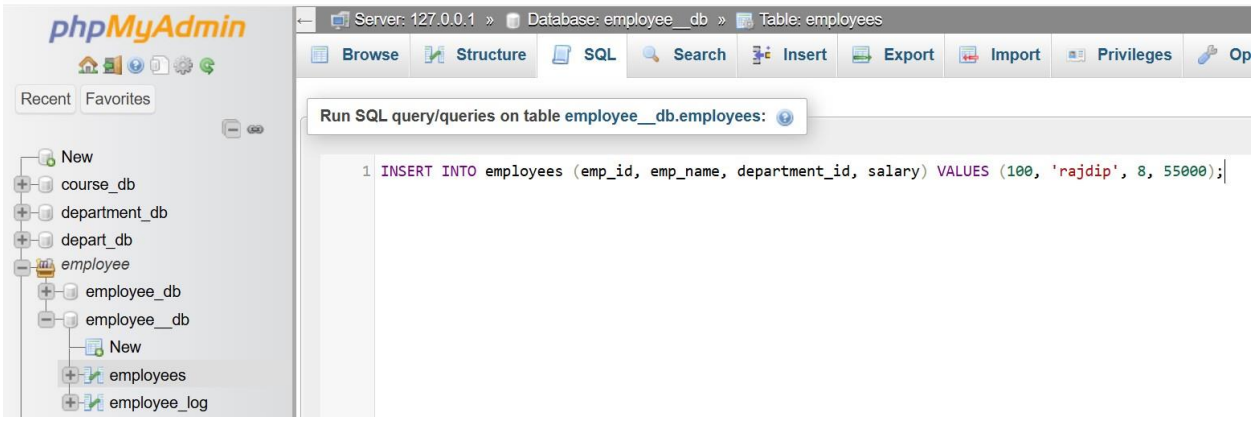
Lab 1: Create a trigger to automatically log changes to the employee's table when a new employee is added.

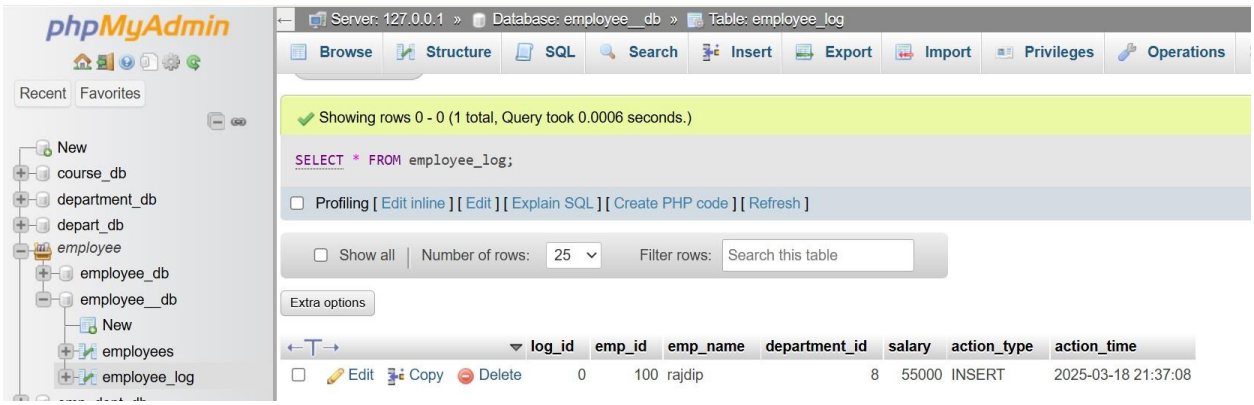
Ans.


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee
employee_db
employee__db
emp_dept_db
Server: 127.0.0.1 » Database: employee__db
Structure
SQL
Search
Query
Export
Import
Run SQL query/queries on database employee__db:
CREATE TABLE employees
(
emp_id INT PRIMARY KEY,
emp_name VARCHAR(50),
department_id INT,
salary int
);


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee
employee_db
employee__db
New
employees
emp_dept_db
Server: 127.0.0.1 » Database: employee__db » Table: employees
Browse
Structure
SQL
Search
Insert
Export
Run SQL query/queries on table employee__db.employees:
CREATE TABLE employee_log
(
log_id INT PRIMARY KEY,
emp_id INT,
emp_name VARCHAR(50),
department_id INT,
salary int,
action_type VARCHAR(50),
action_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

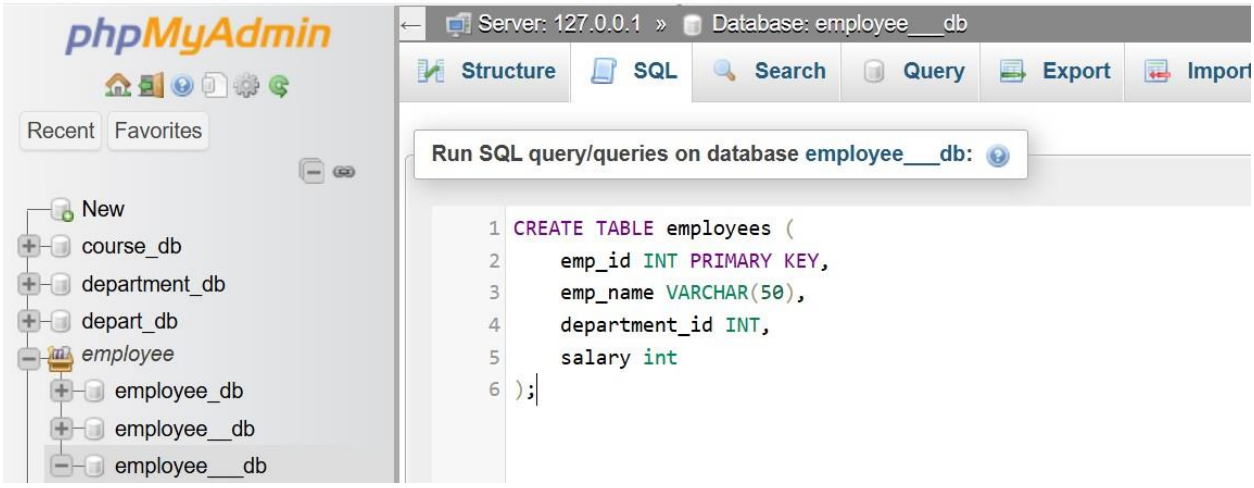

phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee
employee_db
employee__db
New
employees
employee_log
Server: 127.0.0.1 » Database: employee__db » Table: employee_log
Browse
Structure
SQL
Search
Insert
Export
Import
Run SQL query/queries on table employee__db.employee_log:
DELIMITER $$
CREATE TRIGGER after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
INSERT INTO employee_log (emp_id, emp_name, department_id, salary, action_type)
VALUES (NEW.emp_id, NEW.emp_name, NEW.department_id, NEW.salary, 'INSERT');
END

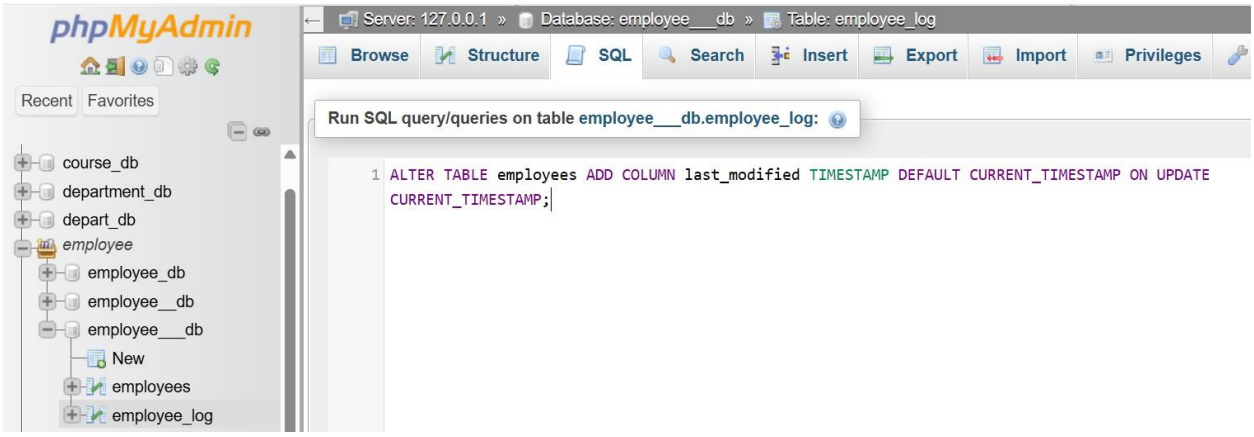


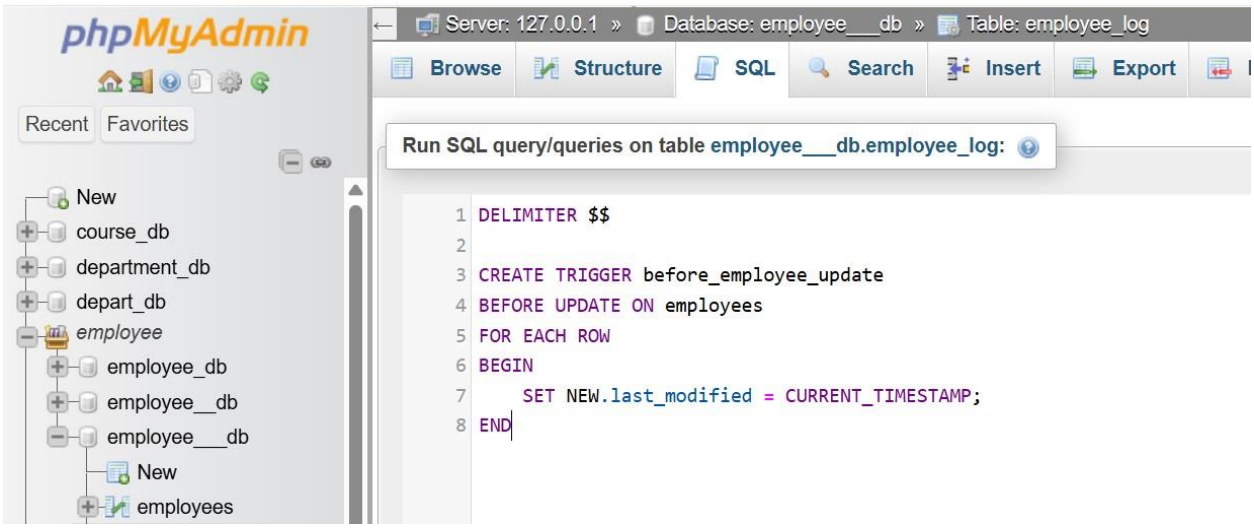


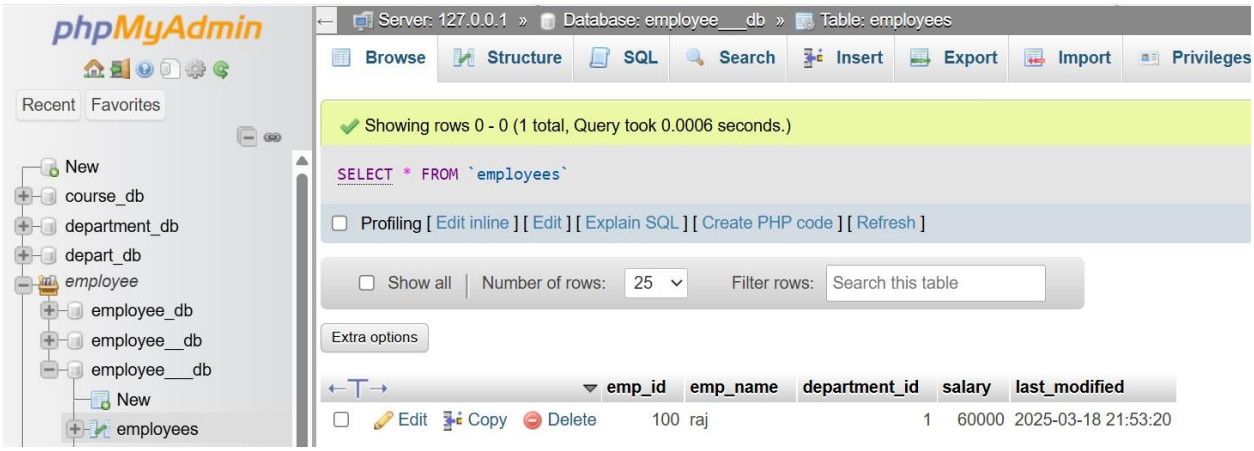
Lab 2: Create a trigger to update the last_modified timestamp whenever an employee record is updated.

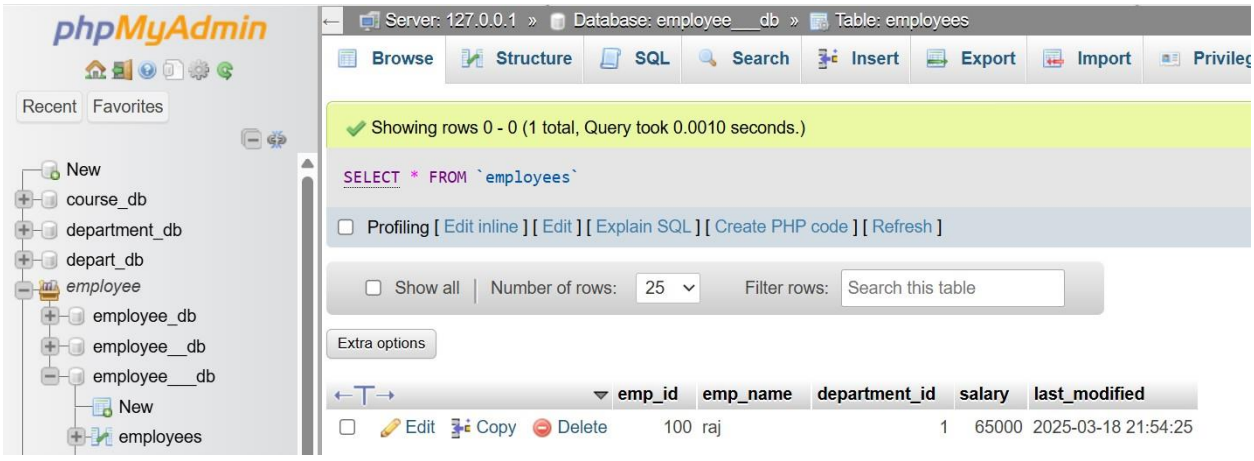
Ans.


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee
employee_db
employee__db
employee___db
Server: 127.0.0.1 » Database: employee___db
Structure
SQL
Search
Query
Export
Import
Run SQL query/queries on database employee___db:
CREATE TABLE employees (
emp_id INT PRIMARY KEY,
emp_name VARCHAR(50),
department_id INT,
salary int
);


phpMyAdmin
Recent
Favorites
course_db
department_db
depart_db
employee
employee_db
employee__db
employee___db
New
employees
employee_log
Server: 127.0.0.1 » Database: employee___db » Table: employee_log
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Run SQL query/queries on table employee___db.employee_log:
ALTER TABLE employees ADD COLUMN last_modified TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;


phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee
employee_db
employee__db
employee___db
New
employees
Server: 127.0.0.1 » Database: employee___db » Table: employee_log
Browse
Structure
SQL
Search
Insert
Export
Run SQL query/queries on table employee___db.employee_log:
DELIMITER $$
CREATE TRIGGER before_employee_update
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
SET NEW.last_modified = CURRENT_TIMESTAMP;
END





16. Introduction to PL/SQL

Theory Questions:

1. What is PL/SQL, and how does it extend SQL's capabilities?

Ans.

What is PL/SQL:

PL/SQL (Procedural Language/Structured Query Language).

How PL/SQL Extends SQL's Capabilities:

feature	SQL	PL/SQL
Query Execution	Executes single SQL statements (e.g., SELECT, INSERT)	Executes multiple SQL statements within a block
Procedural Logic	Not supported	Supports loops, conditions (IF), and error handling
Stored Procedures & Functions	Not available in pure SQL	Allows creation of stored procedures & functions
Exception Handling	No built-in handling for errors	Supports structured error handling
Triggers & Cursors	Limited	Supports triggers & cursors for advanced data manipulation
Performance	Executes queries one at a time	Reduces network traffic by processing multiple statements together

2. List and explain the benefits of using PL/SQL.

Ans.

PL/SQL (Procedural Language/Structured Query Language).

PL/SQL Benefits:

Performance Optimization.

Code Reusability & Modularity.

Procedural Logic.

Exception Handling.

Security & Access Control.

Triggers for Automation.

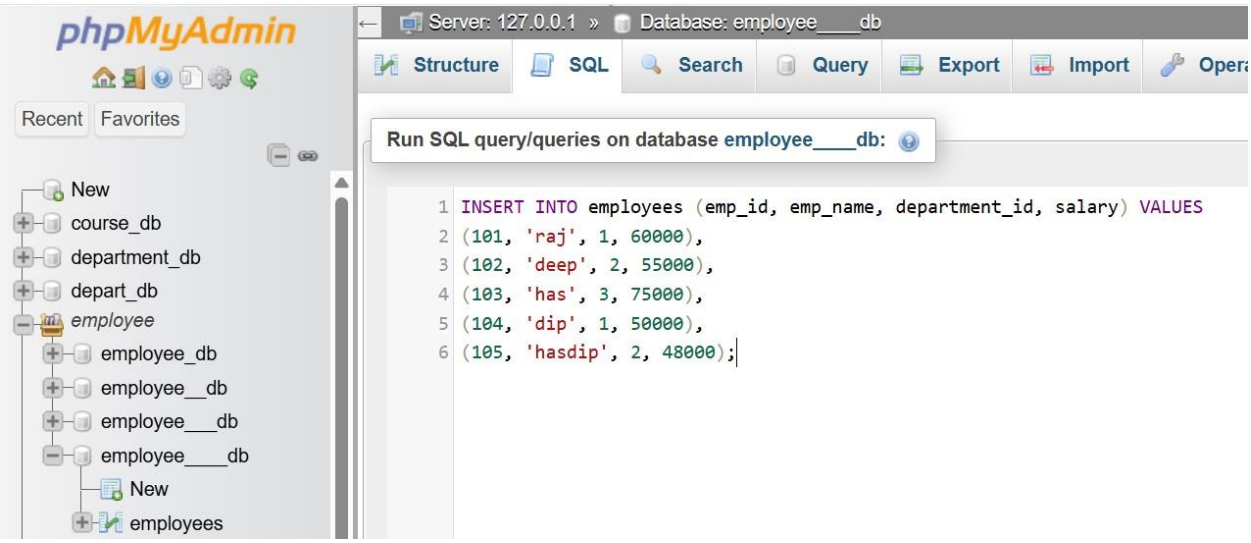
Efficient Data Handling.

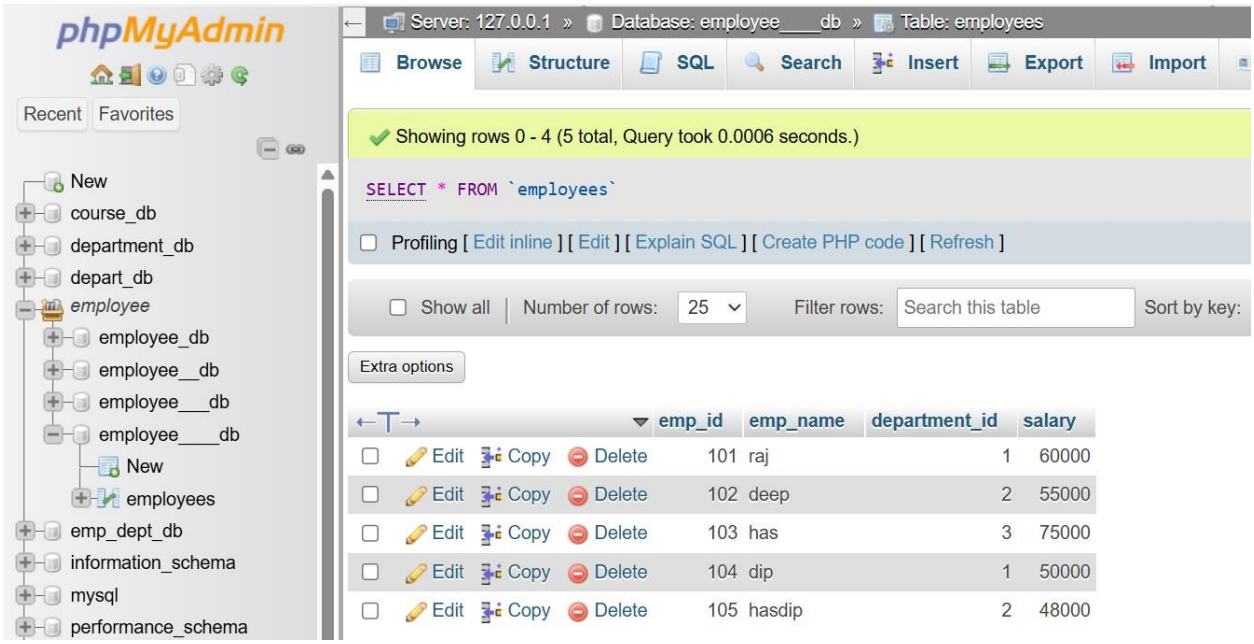
Reduces Application Load.

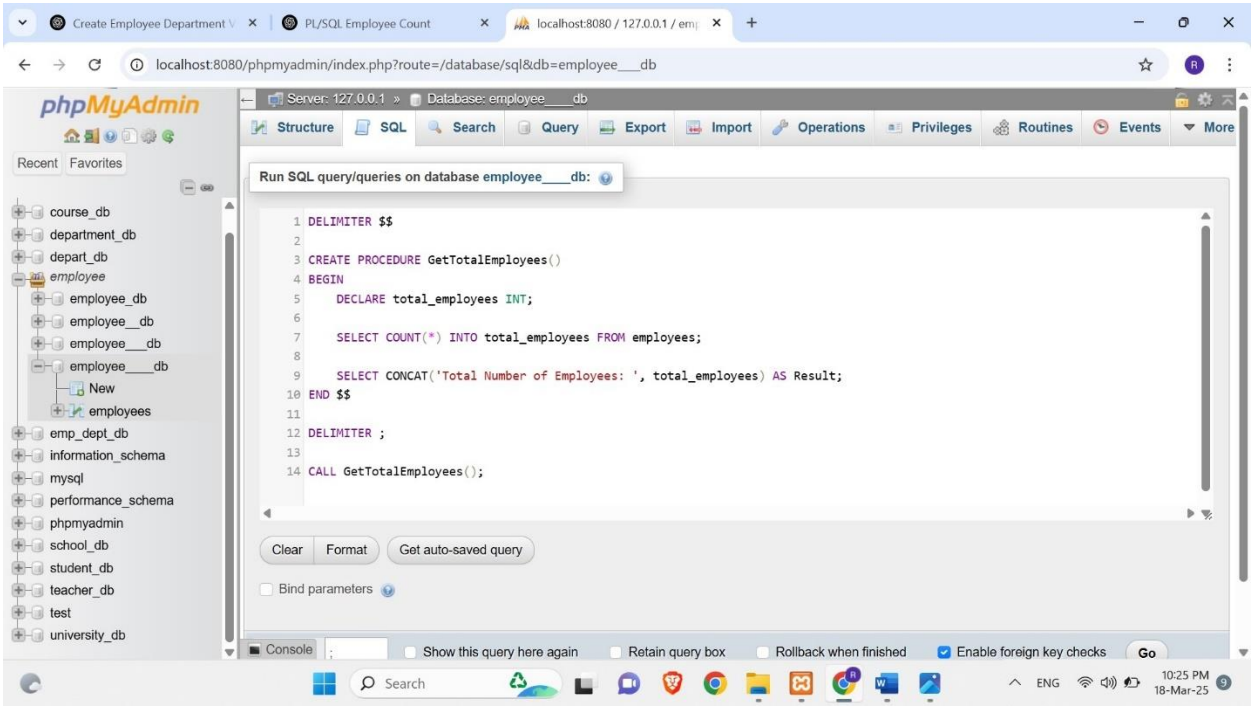
LAB EXERCISES:

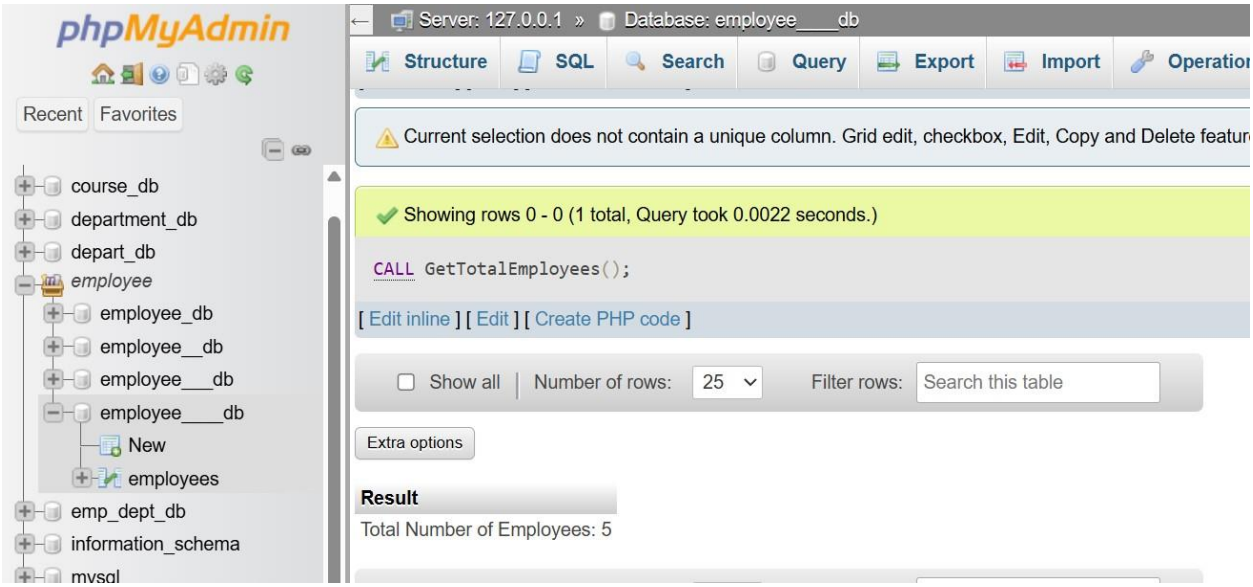
Lab 1: Write a PL/SQL block to print the total number of employees from the employee's table.

Ans.



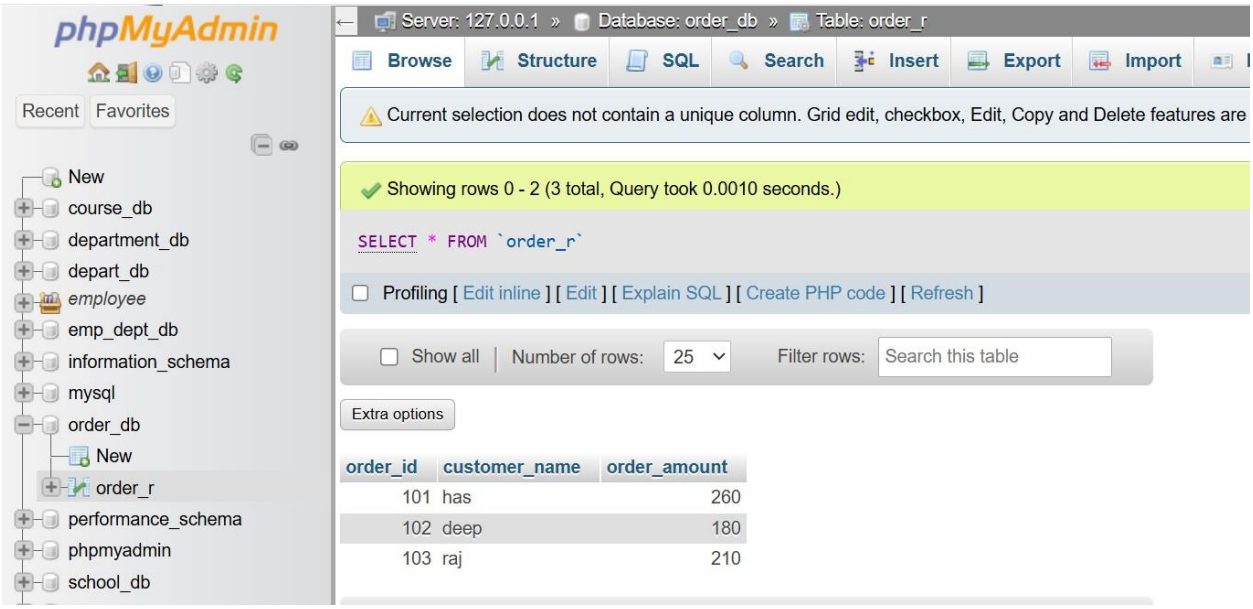

phpMyAdmin
Recent
Favorites
New
course_db
department_db
depart_db
employee
employee_db
employee__db
employee___db
employee____db
New
employees
emp_dept_db
information_schema
mysql
performance_schema
Server: 127.0.0.1 » Database: employee____db » Table: employees
Browse
Structure
SQL
Search
Insert
Export
Import
Showing rows 0 - 4 (5 total, Query took 0.0006 seconds.)
SELECT * FROM `employees`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key:
Extra options
emp_id emp_name department_id salary
Edit Copy Delete 101 raj 1 60000
Edit Copy Delete 102 deep 2 55000
Edit Copy Delete 103 has 3 75000
Edit Copy Delete 104 dip 1 50000
Edit Copy Delete 105 hasdip 2 48000

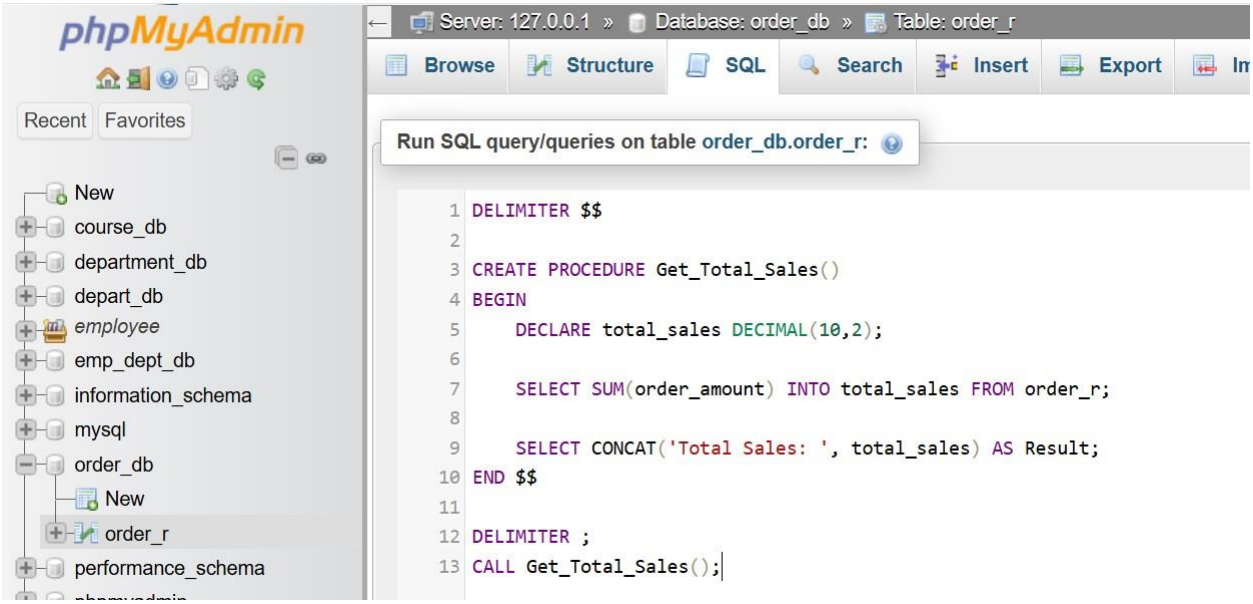

Create Employee Department
PL/SQL Employee Count
localhost:8080 / 127.0.0.1 / emp
localhost:8080/phpmyadmin/index.php?route=/database/sql&db=employee____db
phpMyAdmin
Server: 127.0.0.1 » Database: employee____db
Structure
SQL
Search
Query
Export
Import
Operations
Privileges
Routines
Events
More
Run SQL query/queries on database employee____db:
DELIMITER $$
CREATE PROCEDURE GetTotalEmployees()
BEGIN
DECLARE total_employees INT;
SELECT COUNT(*) INTO total_employees FROM employees;
SELECT CONCAT('Total Number of Employees: ', total_employees) AS Result;
END $$
DELIMITER ;
CALL GetTotalEmployees();
Clear
Format
Get auto-saved query
Bind parameters
Console
Show this query here again
Retain query box
Rollback when finished
Enable foreign key checks
Go
Search
ENG
10:25 PM
18-Mar-25

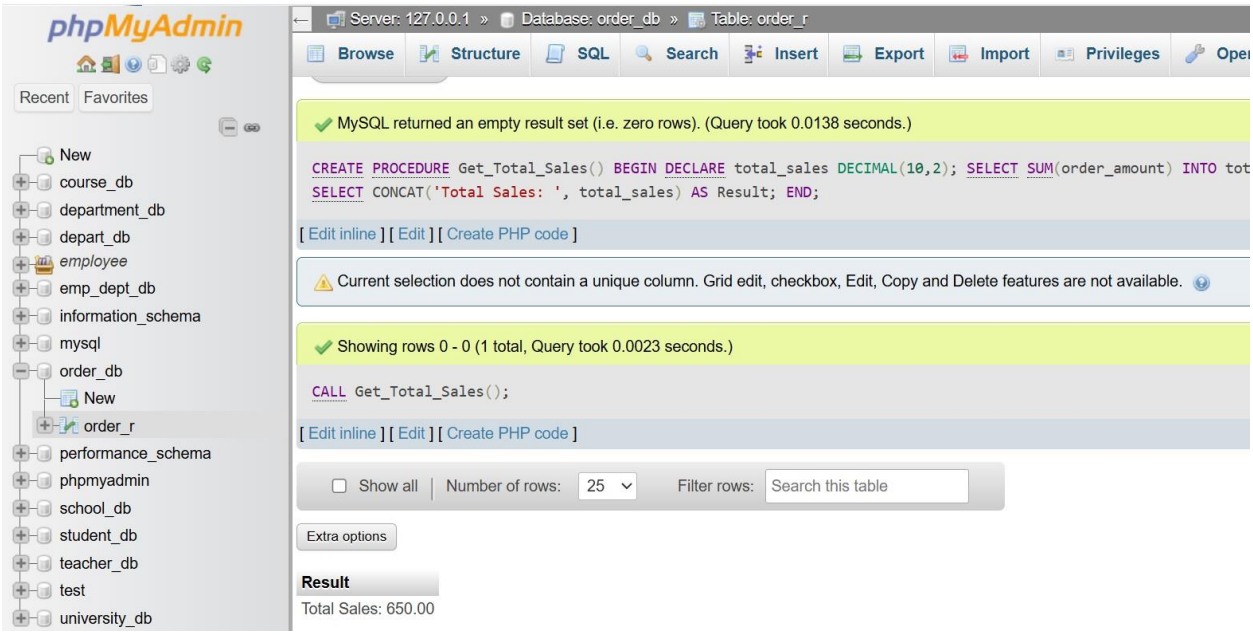


Lab 2: Create a PL/SQL block that calculates the total sales from an orders table.

Ans.







17. PL/SQL Control Structures

Theory Questions:

1. What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

Ans.

Control structures in PL/SQL control the flow of execution within a program.

There are three types of control structures:

1. Conditional Statements – (IF-THEN, IF-THEN-ELSE, CASE)
2. Looping Statements – (LOOP, WHILE LOOP, FOR LOOP)
3. Sequential Control – (GOTO, EXIT).

1. IF-THEN Control Structure:

The IF-THEN structure executes a block of code only if a condition is TRUE.

Syntax:

```
IF condition THEN
-- Code to execute if the condition is TRUE
END IF;
```

2. LOOP Control Structure (Repeated Execution):

Loops repeat a block of code multiple times until a condition is met.

Types of Loops in PL/SQL:

1. Simple LOOP – Runs indefinitely until explicitly exited.
2. WHILE LOOP – Runs as long as a condition is TRUE.
3. FOR LOOP – Runs a fixed number of times.

Simple loop syntax:

```
LOOP
 -- Code to execute repeatedly
EXIT WHEN condition;
END LOOP;
```

2. How do control structures in PL/SQL help in writing complex queries?

Ans.

Why Use Control Structures in Complex Queries:

feature	benefit
Conditional Logic (IF-THEN, CASE)	Customizes execution based on conditions
Looping (LOOP, FOR, WHILE)	Automates bulk data processing
Cursors	Processes multiple rows efficiently
Triggers	Automates data integrity & auditing
Exception Handling (EXCEPTION)	Prevents query failures & logs errors

LAB EXERCISES:

Lab 1: Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

Ans.

```
DECLARE

v_employee_id NUMBER := 101;

v_department_id NUMBER;

BEGIN

 SELECT department_id INTO v_department_id

FROM employees

WHERE employee_id = v_employee_id;

IF v_department_id = 10 THEN

DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in the HR department.');

ELSIF v_department_id = 20 THEN DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in the Sales department.');

ELSE DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in another department.');
```

```
END IF;

END;
```

Lab 2: Use a FOR LOOP to iterate through employee records and display their names.

Ans.

```
DECLARE

CURSOR emp_cursor IS

SELECT employee_id, first_name, last_name FROM employees;

BEGIN FOR emp_rec IN emp_cursor LOOP

 DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_rec.employee_id || ', Name: '
|| emp_rec.first_name || ' ' || emp_rec.last_name);

END LOOP;

END;
```

18. SQL Cursors

Theory Questions:

1. What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

Ans.

What is a Cursor in PL/SQL?

A cursor in PL/SQL is a pointer to a result set that allows you to retrieve and process multiple rows from a query one at a time.

Key Differences: Implicit vs. Explicit Cursors:

Feature	Implicit cursors	Explicit cursors
Declaration	No need (automatically created)	Must be explicitly declared

Control	Managed by PL/SQL	User controls opening, fetching, and closing
Usage	Used for single-row queries	Used for multi-row queries
Performance	More efficient for single-row operations	Better for handling multiple rows
Example	SELECT INTO, INSERT, UPDATE, DELETE.	SELECT returning multiple rows

2. When would you use an explicit cursor over an implicit one?

Ans.

You should use an explicit cursor instead of an implicit one when you need fine control over multi-row queries.

While implicit cursors are automatically managed, explicit cursors allow row-by-row processing, making them ideal for complex data operations.

Explicit cursors allow you to OPEN, FETCH, and CLOSE the result set manually.

Explicit cursors can lock rows to prevent changes by other transactions.

explicit cursor allows row-by-row processing.

When to Use Explicit vs. Implicit Cursors:

feature	Explicit cursors	Implicit cursors
Single-row queries	Yes	No
Single-row queries	No	Yes
Loop through results	No	Yes
Row-by-row processing	No	Yes
Locking rows (FOR UPDATE)	No	Yes

LAB EXERCISES:

Lab 1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

Ans.

```
DECLARE

CURSOR emp_cursor IS

SELECT employee_id, first_name, last_name, department_id, salary

FROM employees;

emp_rec emp_cursor%ROWTYPE;

BEGIN OPEN emp_cursor;


LOOP FETCH emp_cursor INTO emp_rec;

EXIT WHEN emp_cursor%NOTFOUND;


DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_rec.employee_id || ', Name: '
|| emp_rec.first_name || ' ' || emp_rec.last_name || ', Department ID: ' ||
emp_rec.department_id || ', Salary: ' || emp_rec.salary);

END LOOP;

CLOSE emp_cursor;

END;
```

Lab 2: Create a cursor to retrieve all courses and display them one by one.

Ans.

```
DECLARE

CURSOR course_cursor IS

SELECT course_id, course_name, instructor FROM courses;
```

```
 course_rec course_cursor%ROWTYPE;
BEGIN
OPEN course_cursor;
 LOOP
 FETCH course_cursor INTO course_rec;
EXIT WHEN course_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Course ID: ' || course_rec.course_id || ', Course Name: ' || course_rec.course_name || ', Instructor: ' || course_rec.instructor);
END LOOP;
 CLOSE course_cursor;
 END;
```

19. Rollback and Commit Savepoint

Theory Questions:

1. Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?

Ans.

Concept of SAVEPOINT in Transaction Management:

A SAVEPOINT in SQL is a marker inside a transaction that allows partial rollbacks.

It helps in managing complex transactions by letting you roll back only to a certain point instead of undoing the entire transaction.

How SAVEPOINT Works:

A transaction begins with INSERT, UPDATE, DELETE, or other SQL operations.

You create a SAVEPOINT at a specific stage in the transaction.

If an error occurs, you can roll back to SAVEPOINT instead of rolling back the entire transaction.

If everything is successful, you COMMIT the changes permanently.

How SAVEPOINT, COMMIT, and ROLLBACK Interact:

command	function
SAVEPOINT savepoint_name	Creates a rollback point in the transaction.
ROLLBACK TO savepoint_name	Rolls back only to the savepoint (keeps changes before it).
COMMIT	Saves all changes permanently, removing all savepoints.
ROLLBACK	Undoes the entire transaction (ignores savepoints).

2. When is it useful to use savepoints in a database transaction?

Ans.

When to Use SAVEPOINTs:

Handling Errors in Multi-Step Transactions:

If a transaction involves multiple operations, some parts may fail.

SAVEPOINTs allow you to roll back only the failed part instead of the whole transaction.

Preventing Total Data Loss in Batch Processing:

When updating or deleting a large number of records, SAVEPOINTs help avoid losing all changes if an error occurs in one part of the batch.

Nested Transactions:

When a transaction has multiple dependent steps, SAVEPOINTs allow rolling back only certain steps while continuing with the rest.

Key Benefits of Using SAVEPOINT:

Multi-step transactions: Allows partial rollbacks instead of restarting the entire transaction.

Batch processing: Prevents total data loss by rolling back only failed batches.

Error handling: Helps recover from failures in one part of the transaction.

Nested transactions: Maintains consistency while handling sub-transactions.

Performance optimization: Saves time by avoiding re-executing the entire transaction.

LAB EXERCISES:

Lab 1: Perform a transaction where you create a savepoint, insert records, and then roll back to the savepoint.

Ans.

```
INSERT INTO employees (employee_id, first_name, department_id, salary)

VALUES (5001, 'raj', 10, 50000);


SAVEPOINT before_insertion;


INSERT INTO employees (employee_id, first_name, department_id, salary)

VALUES (5002, 'deep', 20, 60000);


INSERT INTO employees (employee_id, first_name, department_id, salary)
VALUES (5003, 'Alice', 30, 55000);


ROLLBACK TO before_insertion;

COMMIT;
```

Lab 2: Commit part of a transaction after using a savepoint and then roll back the remaining changes.

Ans.

```
INSERT INTO employees (employee_id, first_name, last_name, department_id,
salary)

VALUES (6001, 'raj', 'chavda', 10, 55000);


 SAVEPOINT before_more_inserts;

INSERT INTO employees (employee_id, first_name, last_name, department_id,
salary)

VALUES (6002, 'deep', 'shah', 20, 60000);


INSERT INTO employees (employee_id, first_name, last_name, department_id,
salary)

VALUES (6003, 'dip', 'Vyas', 30, 65000);


COMMIT;

ROLLBACK TO before_more_inserts;
```