

Name : Dhaval Jethva

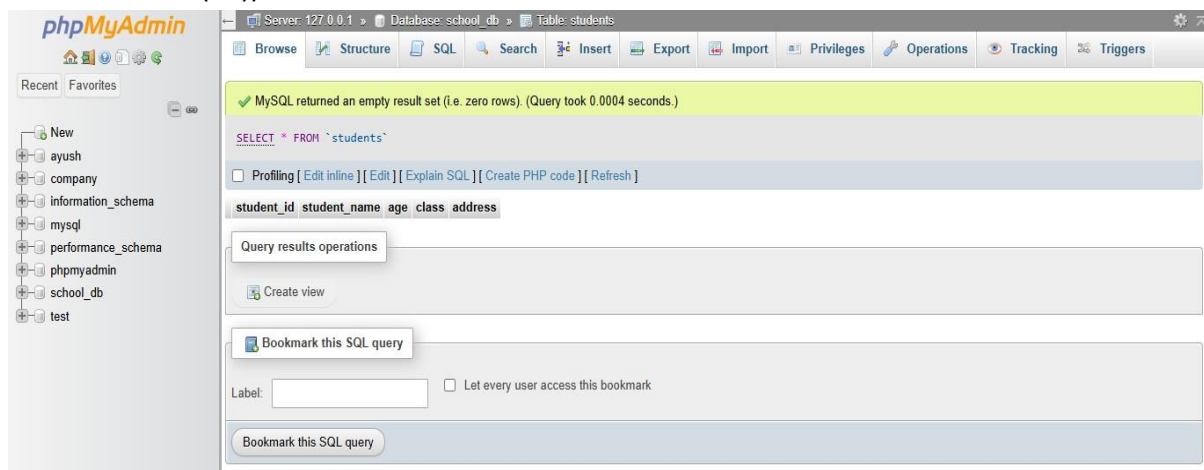
Module 4 – Introduction to

DBMS

1). Introduction to SQL

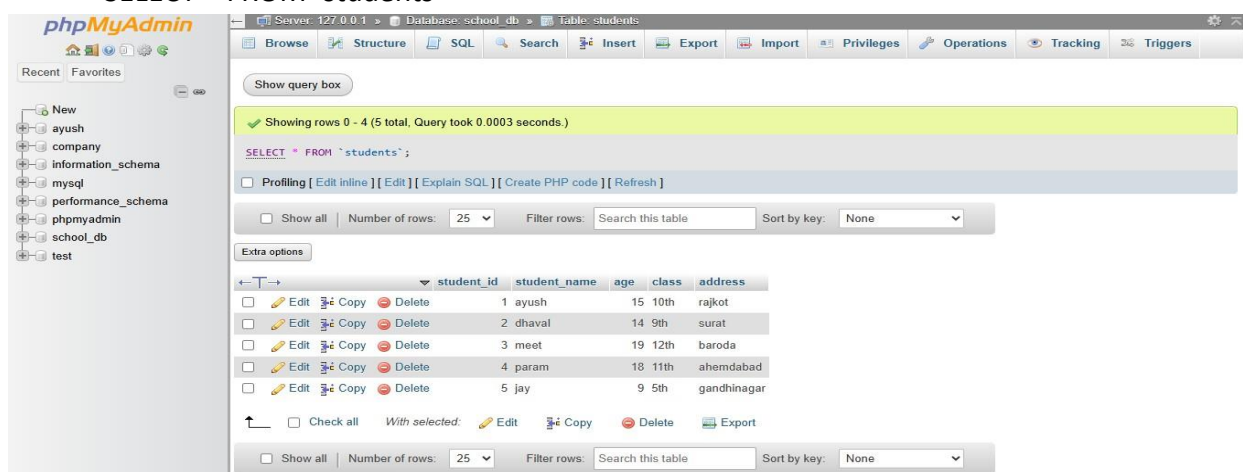
Lab 1 : Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.

- CREATE DATABASE school_db
- CREATE TABLE students(student_id int PRIMARY KEY
AUTO_INCREMENT,student_name varchar(20),age int,class varchar(20),address
varchar(20))



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

- INSERT INTO students(student_name,age,class,address)
VALUES('ayush',15,'10th','rajkot');
- SELECT * FROM `students`



2. SQL Syntax

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

- SELECT student_name,age FROM students

The screenshot shows a database management tool interface. At the top, it displays 'Server: 127.0.0.1', 'Database: school_db', and 'Table: students'. Below this is a toolbar with icons for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, and Operations. A 'Show query box' button is visible. The main area shows a green status bar indicating 'Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)'. Below this, the SQL query 'SELECT student_name,age FROM students;' is entered. A 'Profiling' section with links for 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh' is present. Below the query, there are controls for 'Show all', 'Number of rows' (set to 25), 'Filter rows' (Search this table), and 'Sort by key' (None). An 'Extra options' section is also visible. The results table has two columns: 'student_name' and 'age'. The data rows are:

	student_name	age
<input type="checkbox"/>	ayush	15
<input type="checkbox"/>	dhaval	14
<input type="checkbox"/>	meet	19
<input type="checkbox"/>	param	18
<input type="checkbox"/>	jay	9

Below the table, there are 'Check all', 'With selected' (Edit, Copy, Delete), and 'Export' options. At the bottom, there are more controls for 'Show all', 'Number of rows' (25), 'Filter rows' (Search this table), and 'Sort by key' (None). A 'Query results operations' section is at the very bottom.

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

- SELECT * FROM `students` WHERE age>10

The screenshot shows the same database management tool interface. The SQL query is now 'SELECT * FROM `students` WHERE age>10;'. The status bar indicates 'Showing rows 0 - 3 (4 total, Query took 0.0005 seconds.)'. The results table has five columns: 'student_id', 'student_name', 'age', 'class', and 'address'. The data rows are:

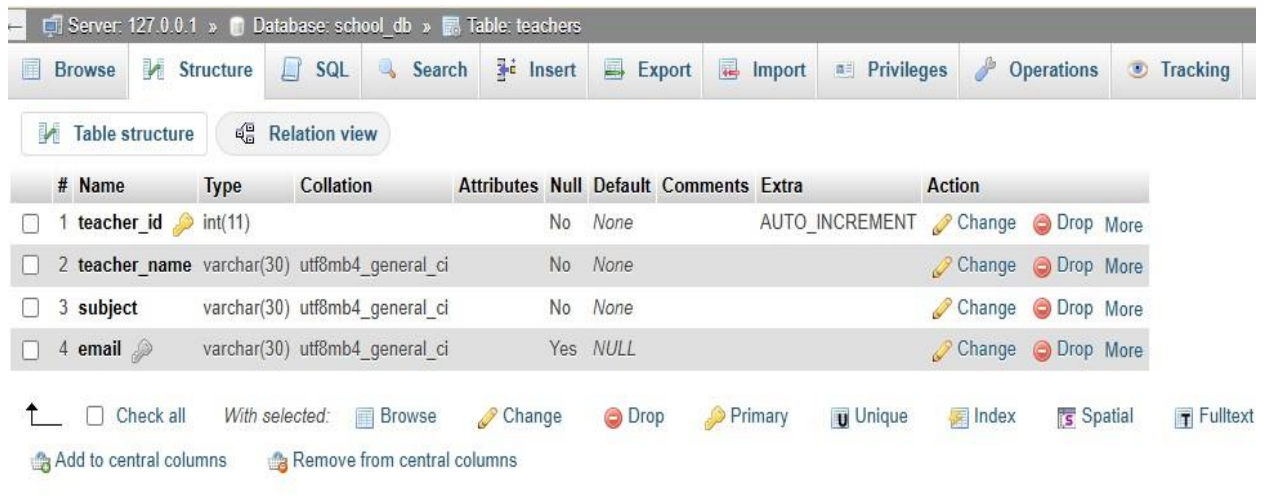
	student_id	student_name	age	class	address
<input type="checkbox"/>	1	ayush	15	10th	rajkot
<input type="checkbox"/>	2	dhaval	14	9th	surat
<input type="checkbox"/>	3	meet	19	12th	baroda
<input type="checkbox"/>	4	param	18	11th	ahemdabad

The interface includes the same toolbar, query box, profiling section, and controls as the previous screenshot.

3. SQL Constraints

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

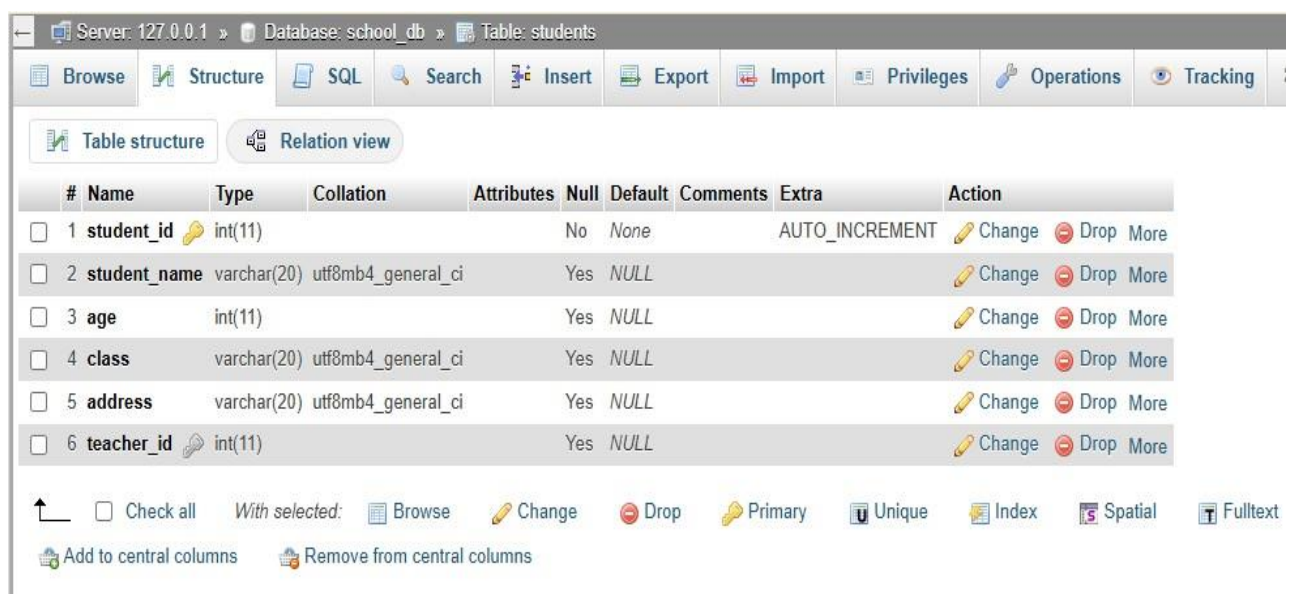
- CREATE TABLE teachers (teacher_id int PRIMARY KEY AUTO_INCREMENT, teacher_name varchar(30) NOT NULL, subject varchar(30) NOT NULL, email varchar(30) UNIQUE)



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 teacher_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 teacher_name	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 subject	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 email	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More

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

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



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 student_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 student_name	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	3 age	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/>	4 class	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	5 address	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	6 teacher_id	int(11)			Yes	NULL			Change Drop More

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

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

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

- CREATE DATABASE university_db
- CREATE TABLE courses(course_id int PRIMARY KEY AUTO_INCREMENT, course_name varchar(30), course_credits varchar(30))



The screenshot shows a database management interface with the following table structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	course_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	course_name	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	course_credits	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Below the table structure, there are options to 'Check all', 'With selected', and various actions like 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. There are also buttons to 'Add to central columns' and 'Remove from central columns'.

5. ALTER Command

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

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

- ALTER TABLE courses ADD course_duration varchar(20)
- ALTER TABLE courses DROP course_credits



The screenshot shows the updated table structure of 'courses' in 'university_db' after the ALTER commands:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	course_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	course_name	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	course_duration	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More

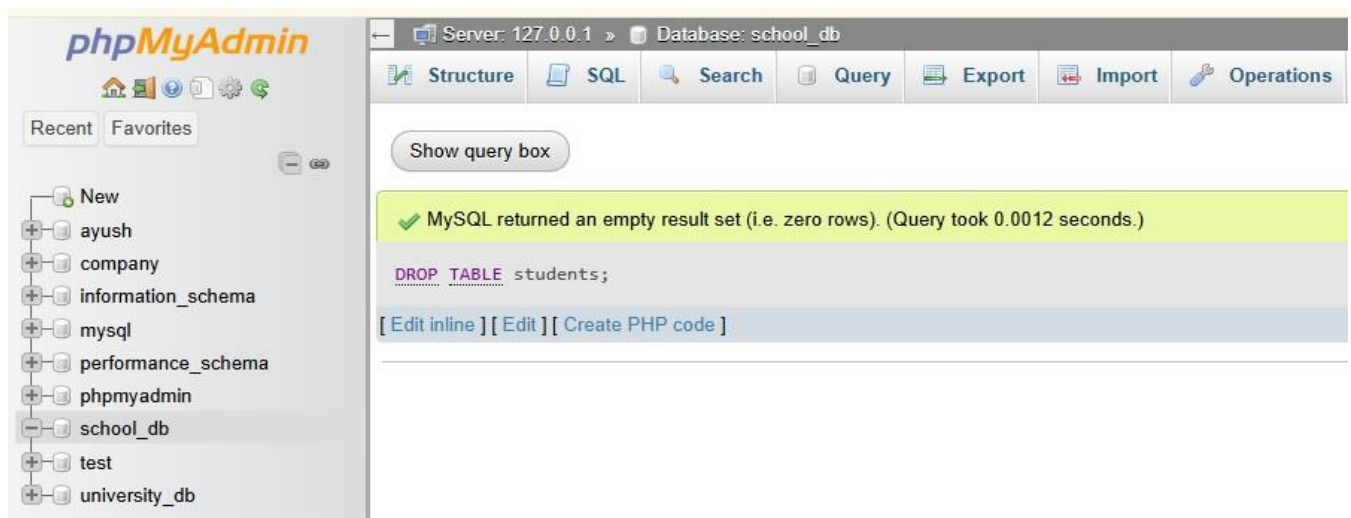
The 'course_credits' column has been removed, and the 'course_duration' column has been added. The interface also shows the same set of action buttons as the previous screenshot.

6. DROP Command

Lab 1 : Drop the teachers table from the school_db database.

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

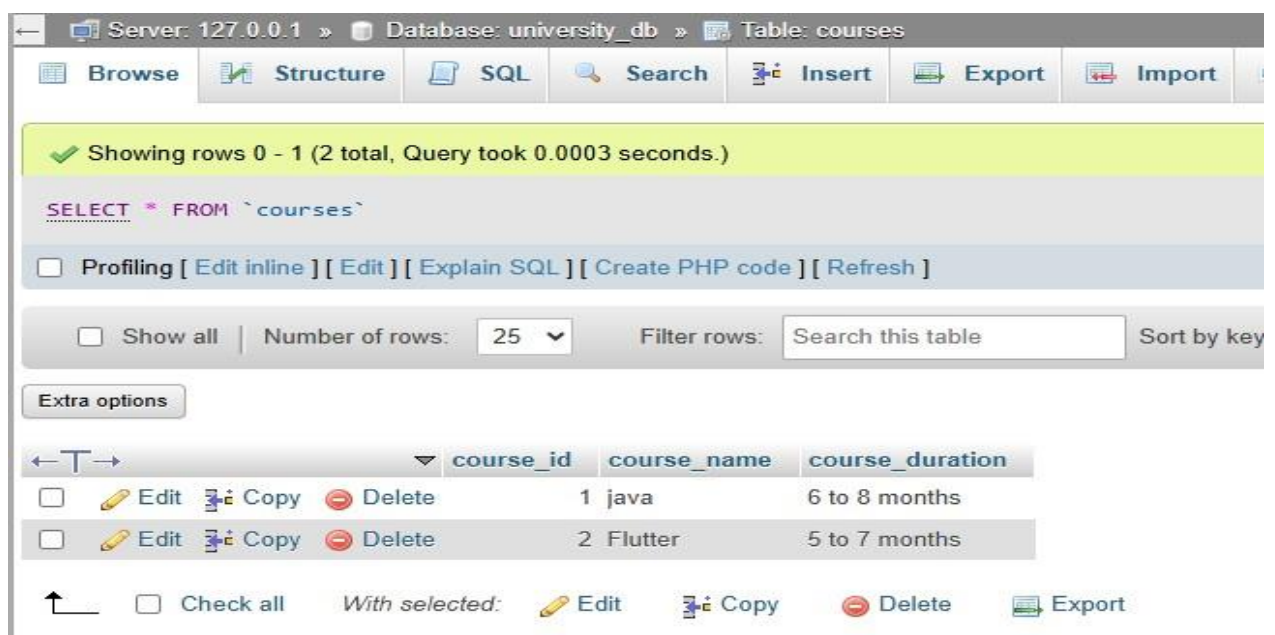
- DROP TABLE teachers
- DROP TABLE students



7. Data Manipulation Language (DML)

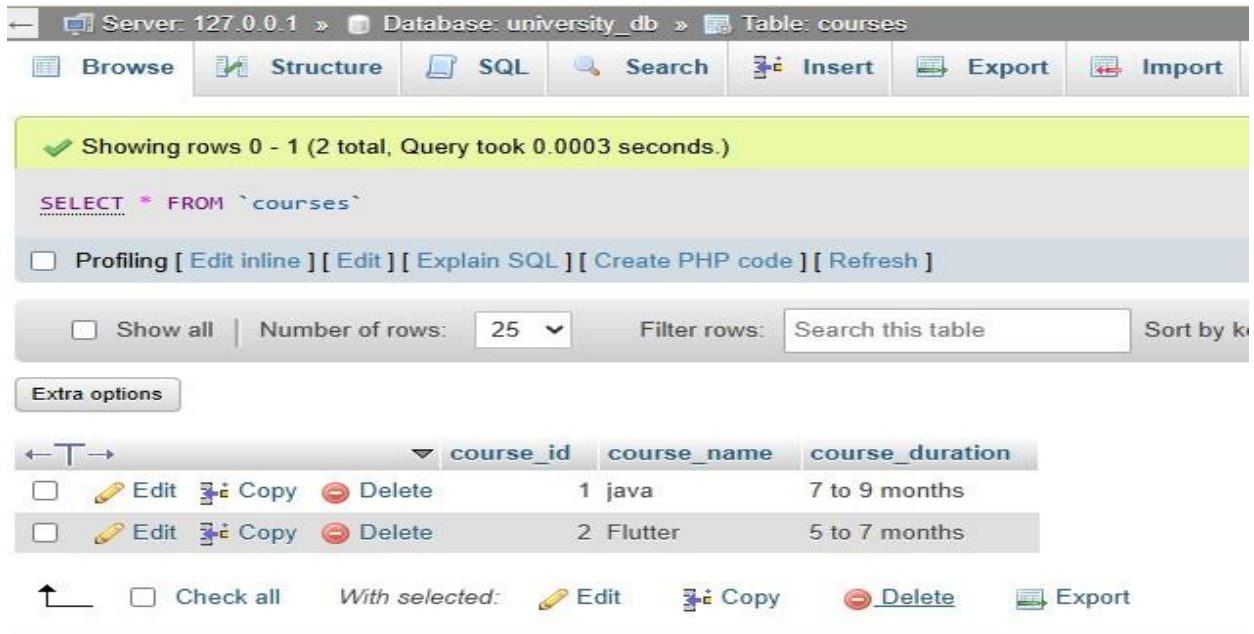
Lab 1 : Insert three records into the courses table using the INSERT command.

- INSERT INTO courses (course_name,course_duration) VALUES ('java','6 to 8 months')



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

- UPDATE courses SET course_duration = '7 to 9 months' WHERE course_id = 1



Server: 127.0.0.1 » Database: university_db » Table: courses

Browse Structure SQL Search Insert Export Import

✓ Showing rows 0 - 1 (2 total, Query took 0.0003 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 k

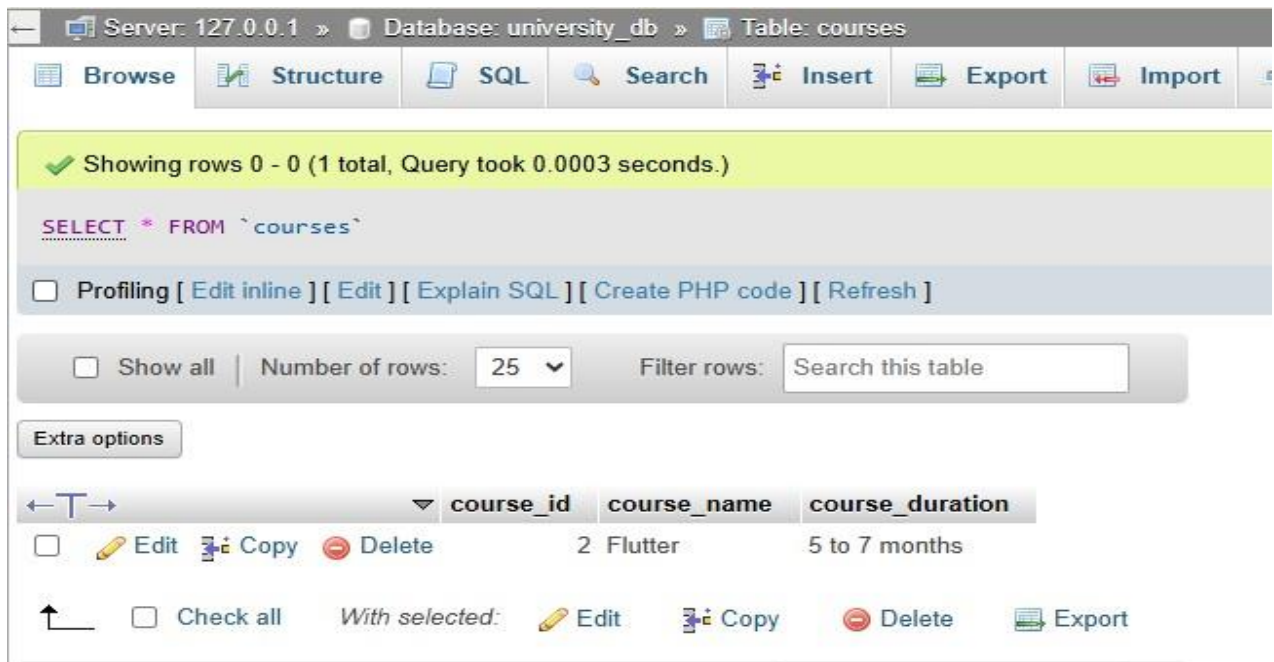
Extra options

	course_id	course_name	course_duration
<input type="checkbox"/> Edit Copy Delete	1	java	7 to 9 months
<input type="checkbox"/> Edit Copy Delete	2	Flutter	5 to 7 months

↑ ☐ Check all With selected: Edit Copy Delete Export

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

- DELETE FROM courses WHERE course_id = 1



Server: 127.0.0.1 » Database: university_db » Table: courses

Browse Structure SQL Search Insert Export Import

✓ Showing rows 0 - 0 (1 total, Query took 0.0003 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

Extra options

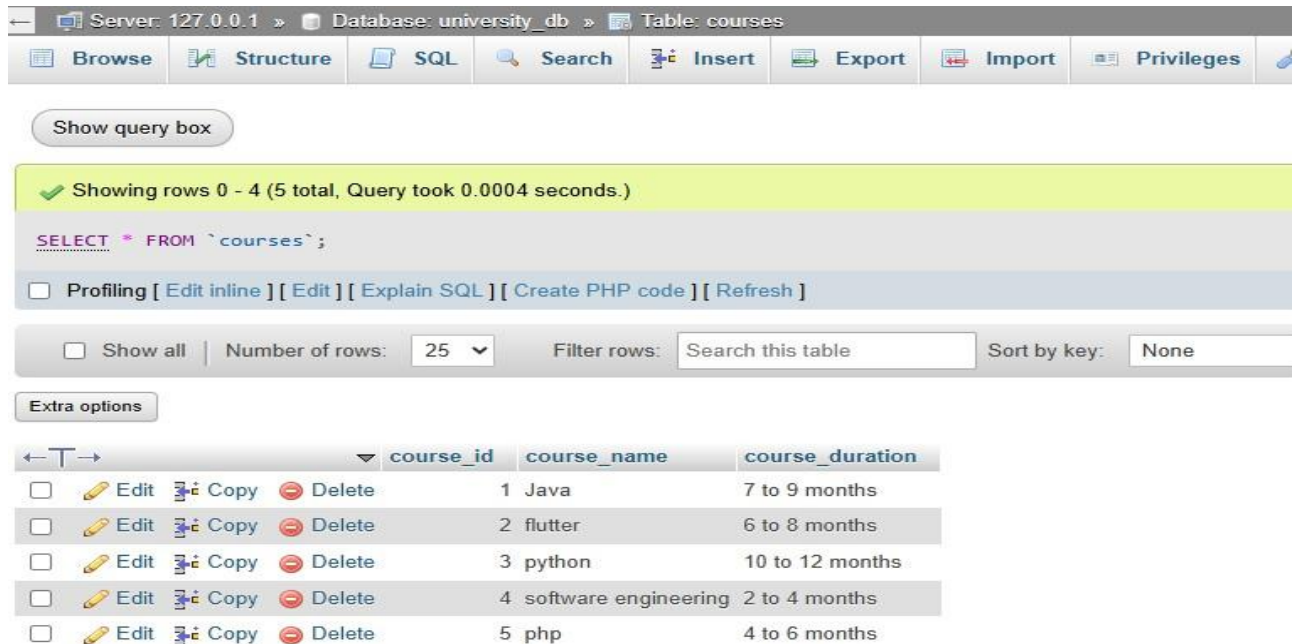
	course_id	course_name	course_duration
<input type="checkbox"/> Edit Copy Delete	2	Flutter	5 to 7 months

↑ ☐ Check all With selected: Edit Copy Delete Export

8. Data Query Language (DQL)

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

- `SELECT * FROM `courses``



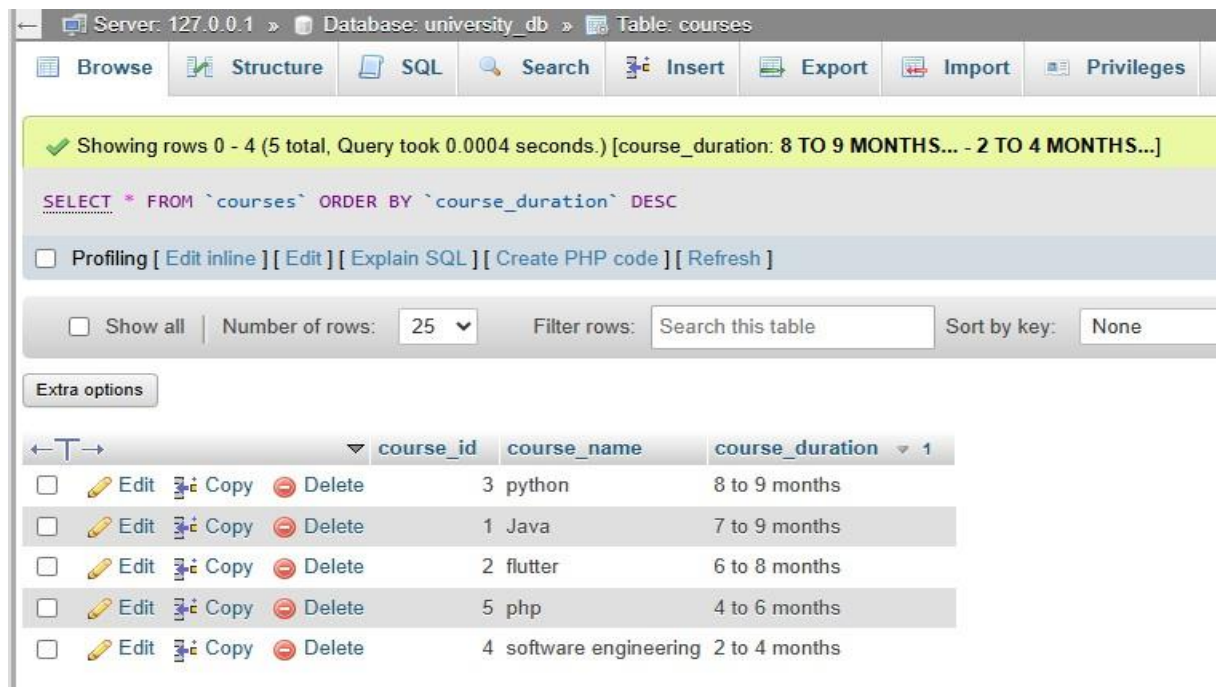
The screenshot shows a database management interface with the following components:

- Header: Server: 127.0.0.1 » Database: university_db » Table: courses
- Navigation bar: Browse, Structure, SQL, Search, Insert, Export, Import, Privileges
- Buttons: Show query box
- Status bar: Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)
- Query editor: `SELECT * FROM `courses`;`
- Options: Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Controls: Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None
- Extra options button
- Table view:

	course_id	course_name	course_duration
<input type="checkbox"/> Edit Copy Delete	1	Java	7 to 9 months
<input type="checkbox"/> Edit Copy Delete	2	flutter	6 to 8 months
<input type="checkbox"/> Edit Copy Delete	3	python	10 to 12 months
<input type="checkbox"/> Edit Copy Delete	4	software engineering	2 to 4 months
<input type="checkbox"/> Edit Copy Delete	5	php	4 to 6 months

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

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



The screenshot shows the same database management interface as before, but with the following changes:

- Status bar: Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.) [course_duration: 8 TO 9 MONTHS... - 2 TO 4 MONTHS...]
- Query editor: `SELECT * FROM `courses` ORDER BY `course_duration` DESC`
- Options: Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Controls: Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None
- Extra options button
- Table view:

	course_id	course_name	course_duration
<input type="checkbox"/> Edit Copy Delete	3	python	8 to 9 months
<input type="checkbox"/> Edit Copy Delete	1	Java	7 to 9 months
<input type="checkbox"/> Edit Copy Delete	2	flutter	6 to 8 months
<input type="checkbox"/> Edit Copy Delete	5	php	4 to 6 months
<input type="checkbox"/> Edit Copy Delete	4	software engineering	2 to 4 months

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

- `SELECT * FROM `courses` ORDER BY course_duration DESC LIMIT 2;`

The screenshot shows a database management interface with the following components:

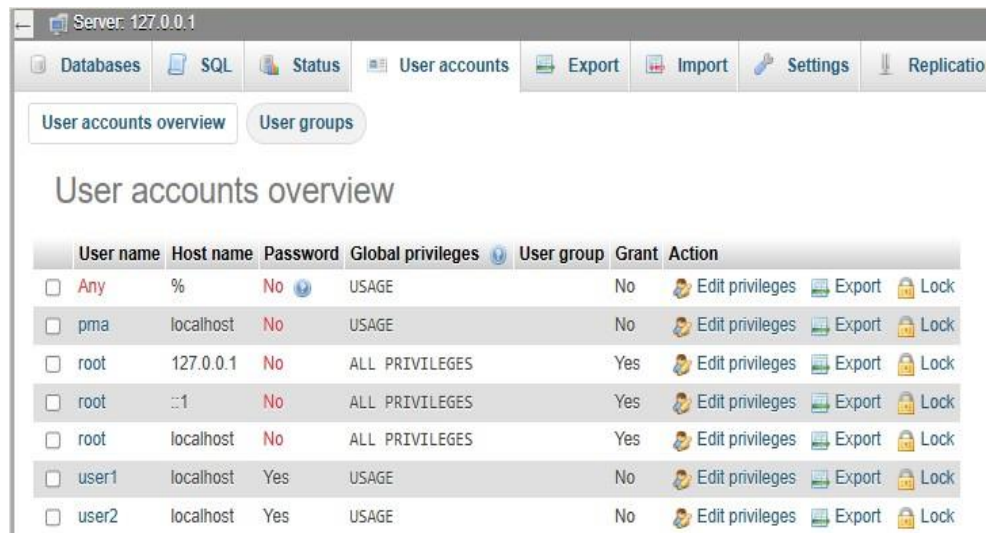
- Header:** Server: 127.0.0.1 » Database: university_db » Table: courses
- Navigation Bar:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges
- Query Execution:** A green status bar indicates "Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.) [course_duration: 8 TO 9 MONTHS... - 7 TO 9 MONTHS...]". Below it, the SQL query is displayed: `SELECT * FROM `courses` ORDER BY course_duration DESC LIMIT 2;`
- Options:** A checkbox for "Profiling" and links for "Edit inline", "Edit", "Explain SQL", "Create PHP code", and "Refresh".
- Table View:** A table with columns `course_id`, `course_name`, and `course_duration`. The results are sorted by `course_duration` in descending order.
- Table Data:**

	course_id	course_name	course_duration
<input type="checkbox"/>	3	python	8 to 9 months
<input type="checkbox"/>	1	Java	7 to 9 months
- Actions:** Below the table, there are checkboxes for "Check all" and "With selected:", followed by buttons for "Edit", "Copy", "Delete", and "Export".

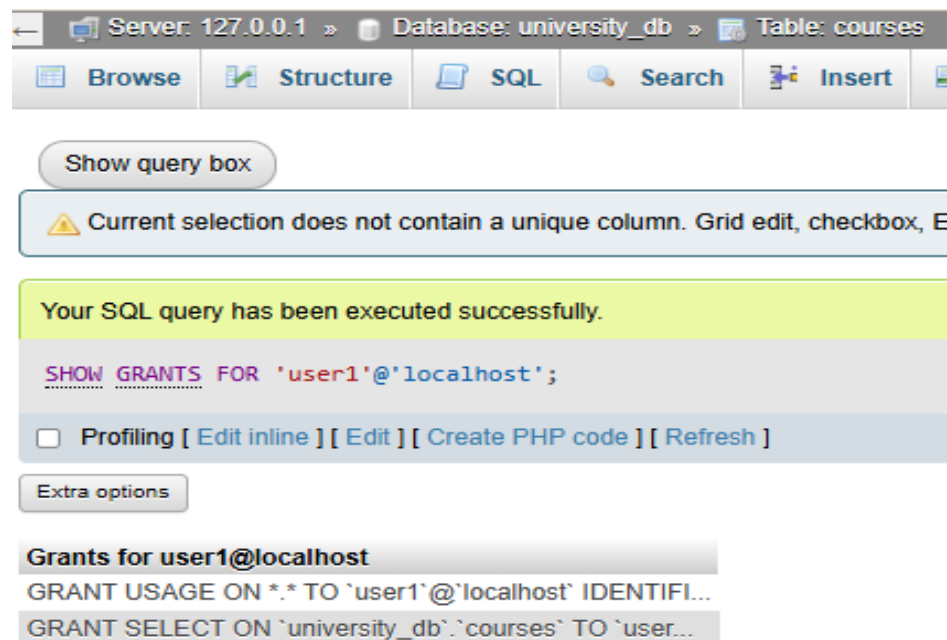
9. Data Control Language (DCL)

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

- CREATE USER 'user1'@'localhost' IDENTIFIED BY 'user1pass';
- CREATE USER 'user2'@'localhost' IDENTIFIED BY 'user2pass';
- GRANT SELECT ON university_db.courses TO 'user1'@'localhost';



	User name	Host name	Password	Global privileges	User group	Grant	Action
<input type="checkbox"/>	Any	%	No	USAGE		No	Edit privileges Export Lock
<input type="checkbox"/>	pma	localhost	No	USAGE		No	Edit privileges Export Lock
<input type="checkbox"/>	root	127.0.0.1	No	ALL PRIVILEGES	Yes	Yes	Edit privileges Export Lock
<input type="checkbox"/>	root	::1	No	ALL PRIVILEGES	Yes	Yes	Edit privileges Export Lock
<input type="checkbox"/>	root	localhost	No	ALL PRIVILEGES	Yes	Yes	Edit privileges Export Lock
<input type="checkbox"/>	user1	localhost	Yes	USAGE		No	Edit privileges Export Lock
<input type="checkbox"/>	user2	localhost	Yes	USAGE		No	Edit privileges Export Lock



Server: 127.0.0.1 » Database: university_db » Table: courses

Browse Structure SQL Search Insert

Show query box

⚠ Current selection does not contain a unique column. Grid edit, checkbox, E

Your SQL query has been executed successfully.

```
SHOW GRANTS FOR 'user1'@'localhost';
```

☐ Profiling [Edit inline] [Edit] [Create PHP code] [Refresh]

Extra options

Grants for user1@localhost

```
GRANT USAGE ON *.* TO 'user1'@'localhost' IDENTIFI...
GRANT SELECT ON `university_db`.`courses` TO `user...
```

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

- REVOKE INSERT ON university_db.courses FROM 'user1'@'localhost';
- GRANT INSERT ON university_db.courses TO 'user2'@'localhost';

Server: 127.0.0.1 » Database: university_db » Table: courses

Browse Structure SQL Search Insert Export

Show query box

⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and

Your SQL query has been executed successfully.

```
SHOW GRANTS FOR 'user1'@'localhost';
```

☐ Profiling [Edit inline] [Edit] [Create PHP code] [Refresh]

Extra options

Grants for user1@localhost

```
GRANT USAGE ON *.* TO 'user1'@'localhost' IDENTIFI...
GRANT SELECT ON `university_db`.`courses` TO `user...
```

Server: 127.0.0.1 » Database: university_db » Table: courses

Browse Structure SQL Search Insert Export Import

Show query box

⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete feature

Your SQL query has been executed successfully.

```
SHOW GRANTS FOR 'user2'@'localhost';
```

☐ Profiling [Edit inline] [Edit] [Create PHP code] [Refresh]

Extra options

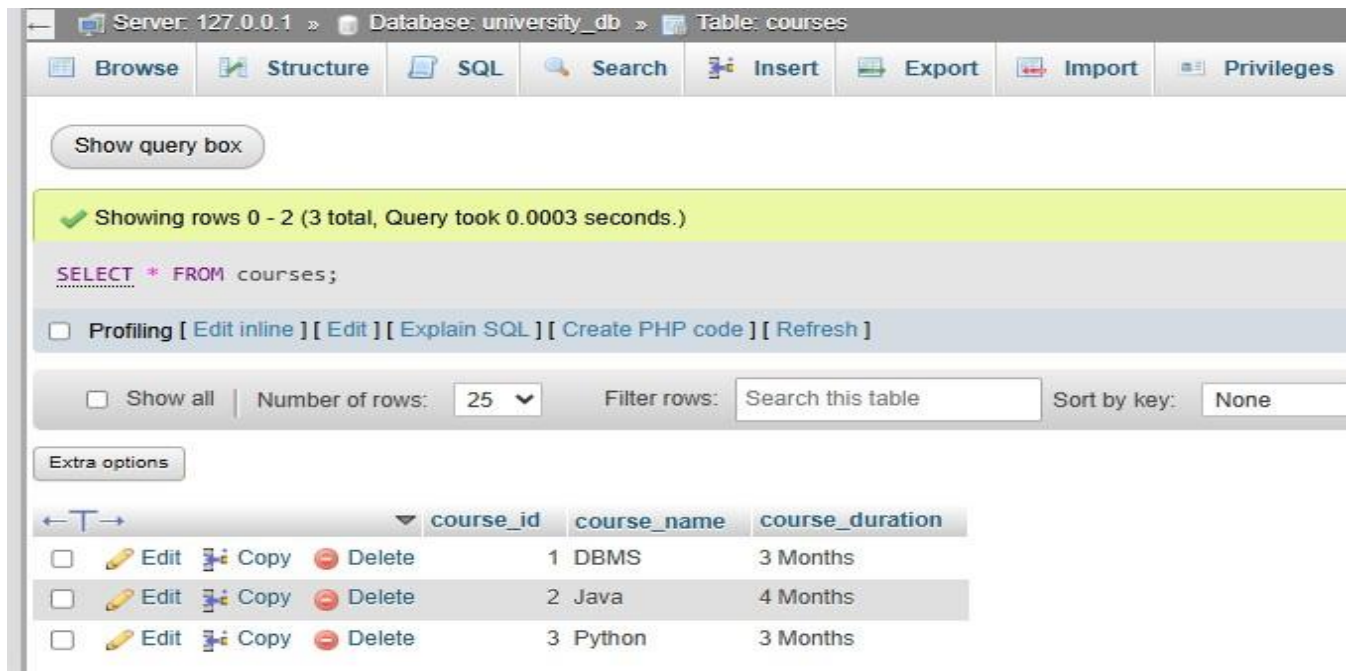
Grants for user2@localhost

```
GRANT USAGE ON *.* TO 'user2'@'localhost' IDENTIFI...
GRANT INSERT ON `university_db`.`courses` TO `user...
```

10. Transaction Control Language (TCL)

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

- INSERT INTO courses (course_name, course_duration) VALUES ('DBMS', '3 Months');
INSERT INTO courses (course_name, course_duration) VALUES ('Java', '4 Months');
INSERT INTO courses (course_name, course_duration) VALUES ('Python', '3 Months');
COMMIT;

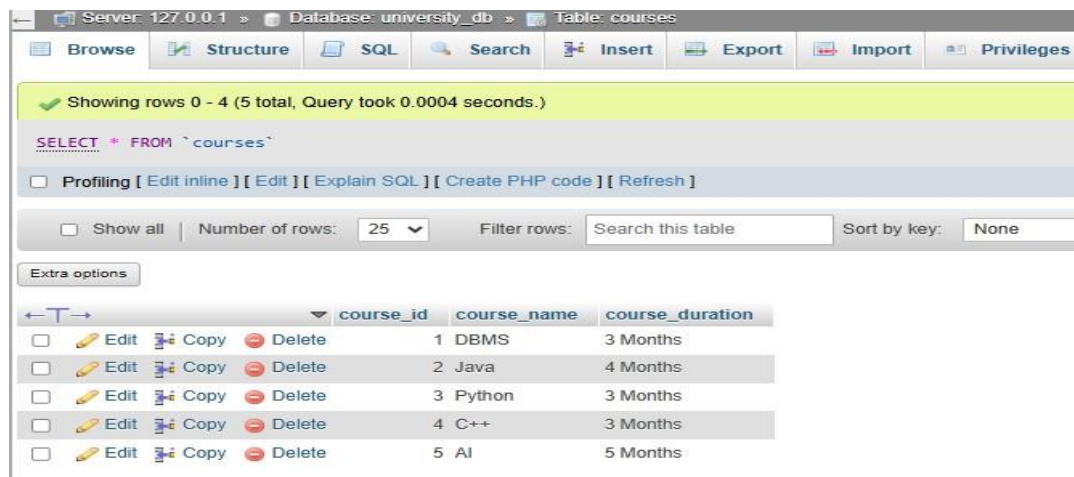


The screenshot shows a database management interface for a server at 127.0.0.1, database 'university_db', and table 'courses'. The interface includes tabs for Browse, Structure, SQL, Search, Insert, Export, Import, and Privileges. A 'Show query box' button is present. A green status bar indicates 'Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)'. The SQL query 'SELECT * FROM courses;' is entered. Below the query, there are links for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. A control bar shows 'Show all', 'Number of rows: 25', 'Filter rows: Search this table', and 'Sort by key: None'. An 'Extra options' button is also visible. The table data is displayed as follows:

	course_id	course_name	course_duration
<input type="checkbox"/> Edit Copy Delete	1	DBMS	3 Months
<input type="checkbox"/> Edit Copy Delete	2	Java	4 Months
<input type="checkbox"/> Edit Copy Delete	3	Python	3 Months

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

- SET autocommit = 0;
INSERT INTO courses (course_id, course_name, course_duration) VALUES (6, 'ML', '4 Months');
INSERT INTO courses (course_id, course_name, course_duration) VALUES (7, 'Data Science', '6 Months');
ROLLBACK;
SET autocommit = 1;

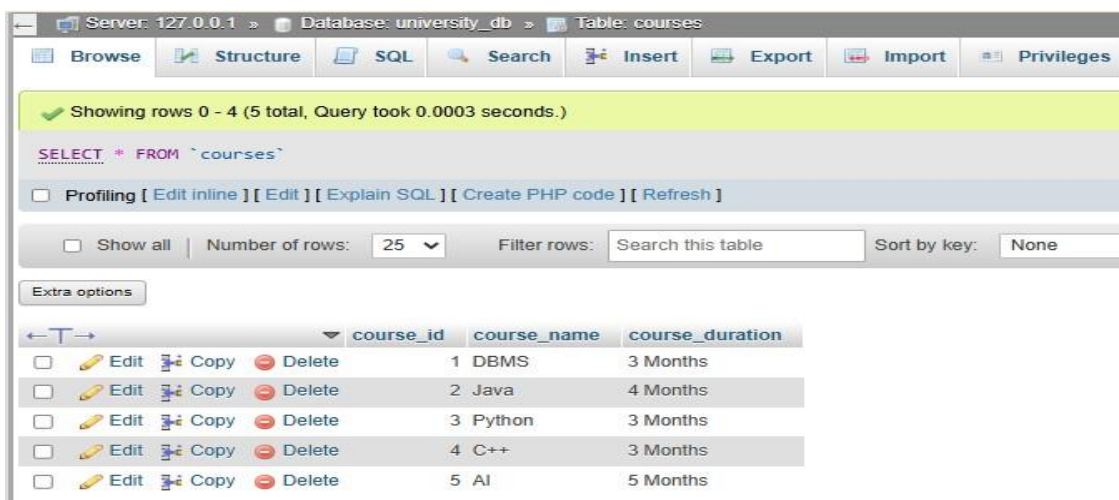


The screenshot shows a database management interface with a toolbar at the top containing buttons for Browse, Structure, SQL, Search, Insert, Export, Import, and Privileges. Below the toolbar, a status bar indicates 'Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)'. The SQL query editor contains the command 'SELECT * FROM `courses`'. Below the query editor, there are options for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. A section for filtering rows shows 'Number of rows: 25' and 'Filter rows: Search this table'. Below this, an 'Extra options' button is visible. The main table displays the following data:

	course_id	course_name	course_duration
<input type="checkbox"/>	1	DBMS	3 Months
<input type="checkbox"/>	2	Java	4 Months
<input type="checkbox"/>	3	Python	3 Months
<input type="checkbox"/>	4	C++	3 Months
<input type="checkbox"/>	5	AI	5 Months

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

- START TRANSACTION;
SAVEPOINT before_update;
UPDATE courses SET course_duration = '7 Months' WHERE course_id=5;
ROLLBACK TO before_update;
COMMIT;



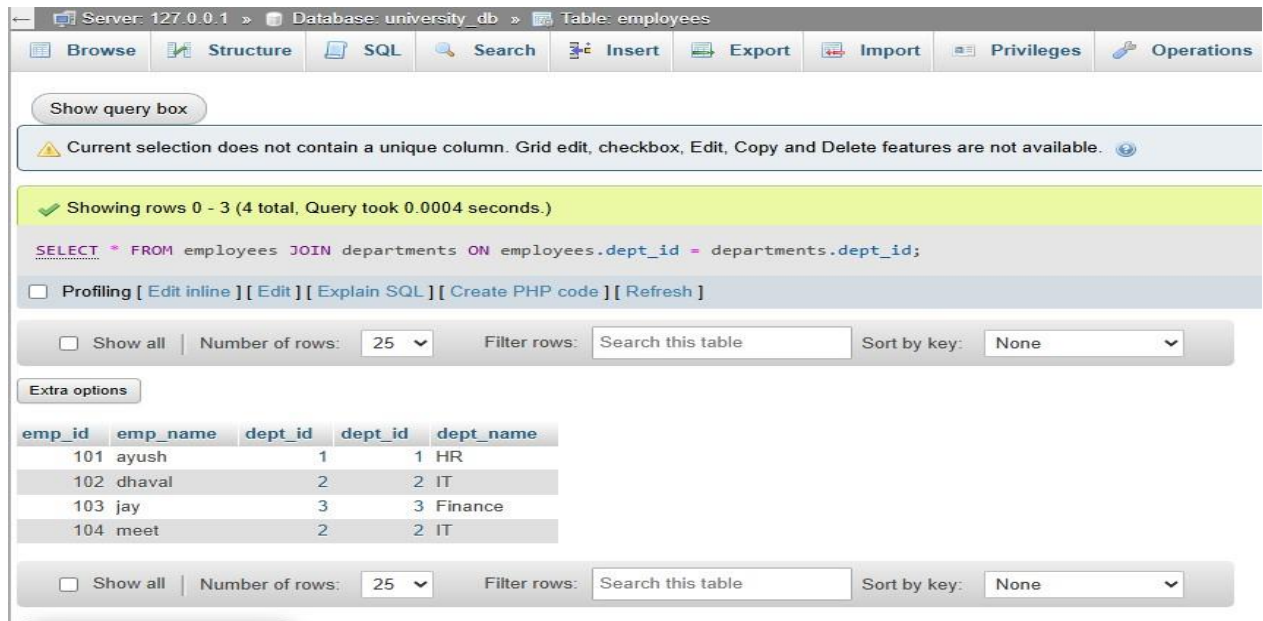
The screenshot shows the same database management interface as before. The status bar indicates 'Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)'. The SQL query editor contains the command 'SELECT * FROM `courses`'. The table displays the following data:

	course_id	course_name	course_duration
<input type="checkbox"/>	1	DBMS	3 Months
<input type="checkbox"/>	2	Java	4 Months
<input type="checkbox"/>	3	Python	3 Months
<input type="checkbox"/>	4	C++	3 Months
<input type="checkbox"/>	5	AI	5 Months

11. SQL Joins

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

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



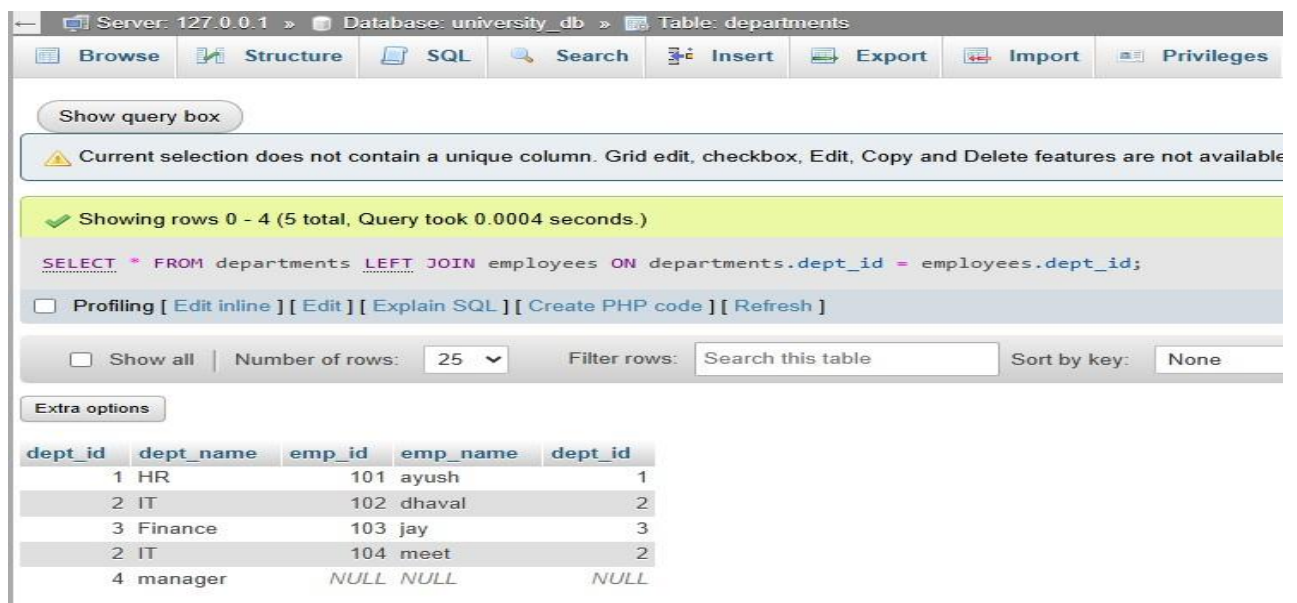
The screenshot shows a SQL IDE interface with the following components:

- Server:** 127.0.0.1 » **Database:** university_db » **Table:** employees
- Navigation Bar:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations
- Show query box:** ☐ (checked)
- Warning:** Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
- Status:** Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)
- Query:** `SELECT * FROM employees JOIN departments ON employees.dept_id = departments.dept_id;`
- Options:** ☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Display:** ☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Extra options:** ☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Table:**

emp_id	emp_name	dept_id	dept_id	dept_name
101	ayush	1	1	HR
102	dhaval	2	2	IT
103	jay	3	3	Finance
104	meet	2	2	IT

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

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



The screenshot shows a SQL IDE interface with the following components:

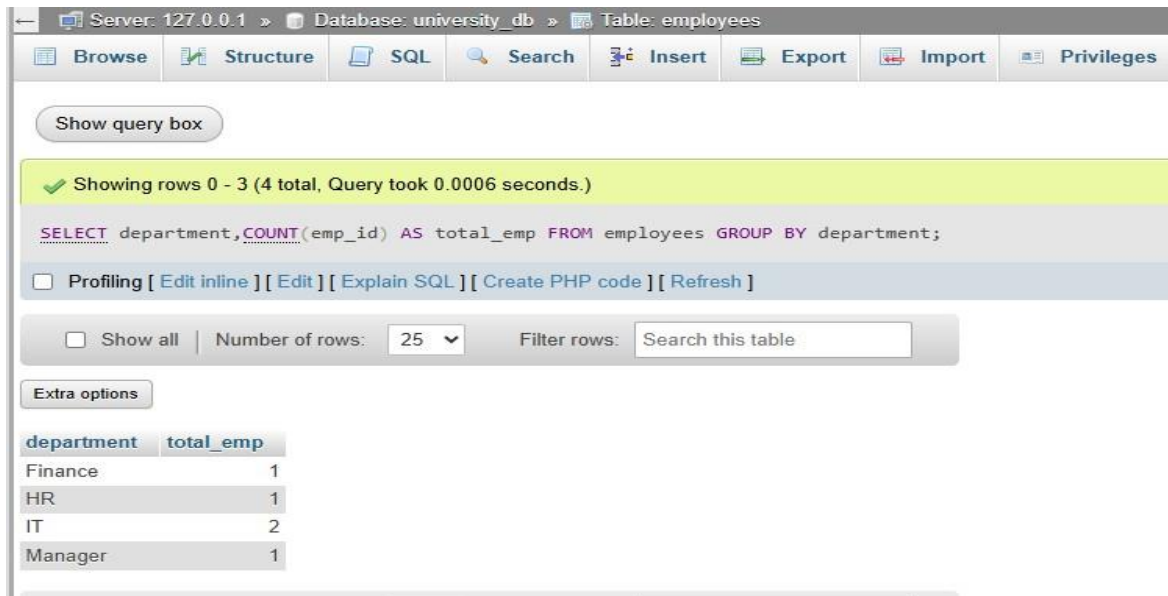
- Server:** 127.0.0.1 » **Database:** university_db » **Table:** departments
- Navigation Bar:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges
- Show query box:** ☐ (checked)
- Warning:** Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available
- Status:** Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)
- Query:** `SELECT * FROM departments LEFT JOIN employees ON departments.dept_id = employees.dept_id;`
- Options:** ☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Display:** ☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Extra options:** ☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Table:**

dept_id	dept_name	emp_id	emp_name	dept_id
1	HR	101	ayush	1
2	IT	102	dhaval	2
3	Finance	103	jay	3
2	IT	104	meet	2
4	manager	NULL	NULL	NULL

12. SQL Group By

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

- SELECT department,COUNT(emp_id) AS total_emp FROM employees GROUP BY department

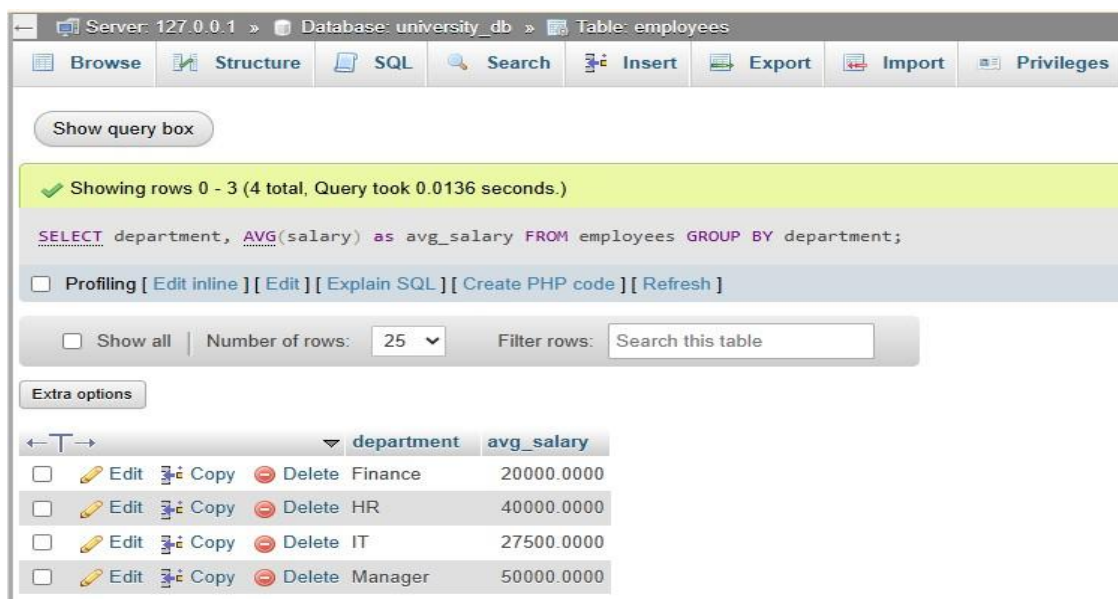


The screenshot shows a database management tool interface. At the top, the breadcrumb navigation indicates 'Server: 127.0.0.1 » Database: university_db » Table: employees'. Below this is a toolbar with buttons for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', and 'Privileges'. A 'Show query box' button is located below the toolbar. The main area displays a green status bar indicating 'Showing rows 0 - 3 (4 total, Query took 0.0006 seconds.)'. Below this, the SQL query is shown: `SELECT department,COUNT(emp_id) AS total_emp FROM employees GROUP BY department;`. There are links for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. Below the query, there are controls for 'Show all', 'Number of rows' (set to 25), and a 'Filter rows' search box. An 'Extra options' button is also present. The results are displayed in a table with two columns: 'department' and 'total_emp'.

department	total_emp
Finance	1
HR	1
IT	2
Manager	1

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

- SELECT department, AVG(salary) as avg_salary FROM employees GROUP BY department



The screenshot shows the same database management tool interface as before. The SQL query is now: `SELECT department, AVG(salary) as avg_salary FROM employees GROUP BY department;`. The results are displayed in a table with two columns: 'department' and 'avg_salary'.

department	avg_salary
Finance	20000.0000
HR	40000.0000
IT	27500.0000
Manager	50000.0000

13. SQL Stored Procedure

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

- DELIMITER //

```
CREATE PROCEDURE get_emp_proc (IN dept_name varchar(20))
BEGIN
SELECT * FROM employees WHERE department = dept_name;
END //

DELIMITER ;
```

The screenshot shows a database management interface with a top toolbar containing icons for Structure, SQL, Search, Query, Export, Import, and Operations. A green status bar at the top indicates: "Your SQL query has been executed successfully. 2 rows affected by the last statement inside the procedure." Below this, the SQL editor contains the code: `SET @p0='manager'; CALL `get_emp_proc` (@p0);`. A pop-up window titled "Execution results of routine `get_emp_proc`" displays a table with the following data:

emp_id	emp_name	department	salary
4	meet	Manager	50000
8	suresh	manager	55000

Below the execution results, there is a "Routines" section with a "Check all" checkbox, "Export" and "Drop" buttons. A table lists the routine "get_emp_proc" as a "PROCEDURE". Below this table are buttons for "Edit", "Execute", "Export", and "Drop".

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

- DELIMITER //

```
CREATE PROCEDURE course_detail_proc (IN c_id int)
BEGIN
SELECT * FROM courses WHERE course_id = c_id;
END //
```

DELIMITER ;

The screenshot shows a database management tool interface. At the top, the server is identified as '127.0.0.1' and the database as 'university_db'. A toolbar contains icons for Structure, SQL, Search, Query, Export, Import, and Operations. A green message box states: 'Your SQL query has been executed successfully. 1 row affected by the last statement inside the procedure.' Below this, the SQL command is displayed: `SET @p0='2'; CALL `course_detail_proc` (@p0);`. A pop-up window titled 'Execution results of routine `course_detail_proc`' shows a table with the following data:

course_id	course_name	course_duration
2	flutter	6 to 8 months

Below the results, the 'Routines' section is visible. It includes a toolbar with 'Check all', 'Export', and 'Drop' buttons. A table lists the routines:

Name	Type	Returns
<input type="checkbox"/> course_detail_proc	PROCEDURE	Edit Execute Export Drop
<input type="checkbox"/> get_emp_proc	PROCEDURE	Edit Execute Export Drop

14. SQL View

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

- CREATE VIEW emp_with_dept_view AS SELECT emp_name,department FROM employees

The screenshot shows a SQL Studio window with the following details:

- Server: 127.0.0.1, Database: university_db, View: emp_with_dept_view
- Warning: Current selection does not contain a unique column. Grid edit, Edit, Copy and Delete features may result in un
- Status: Showing rows 0 - 7 (8 total, Query took 0.0004 seconds.)
- SQL Query: `SELECT * FROM `emp_with_dept_view``
- Buttons: Profiling, Edit inline, Edit, Explain SQL, Create PHP code, Refresh
- Controls: Show all, Number of rows: 25, Filter rows: Search this table
- Extra options: emp_name, department
- Data Table:

	emp_name	department
<input type="checkbox"/>	ayush	IT
<input type="checkbox"/>	dhaval	HR
<input type="checkbox"/>	jay	Finance
<input type="checkbox"/>	meet	Manager
<input type="checkbox"/>	param	IT
<input type="checkbox"/>	mahesh	HR
<input type="checkbox"/>	ramesh	chemical
<input type="checkbox"/>	suresh	manager

Bottom controls: Check all, With selected: Edit, Copy, Delete, Export

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

- CREATE VIEW emp_salary_view AS SELECT emp_id,emp_name,department,salary FROM employees WHERE salary<=50000

The screenshot shows a SQL Studio window with the following details:

- Server: 127.0.0.1, Database: university_db, View: emp_salary_view
- Warning: Current selection does not contain a unique column. Grid edit, Edit, Copy and Delete features may result in
- Status: Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)
- SQL Query: `SELECT * FROM `emp_salary_view``
- Buttons: Profiling, Edit inline, Edit, Explain SQL, Create PHP code, Refresh
- Controls: Show all, Number of rows: 25, Filter rows: Search this table
- Extra options: emp_id, emp_name, department, salary
- Data Table:

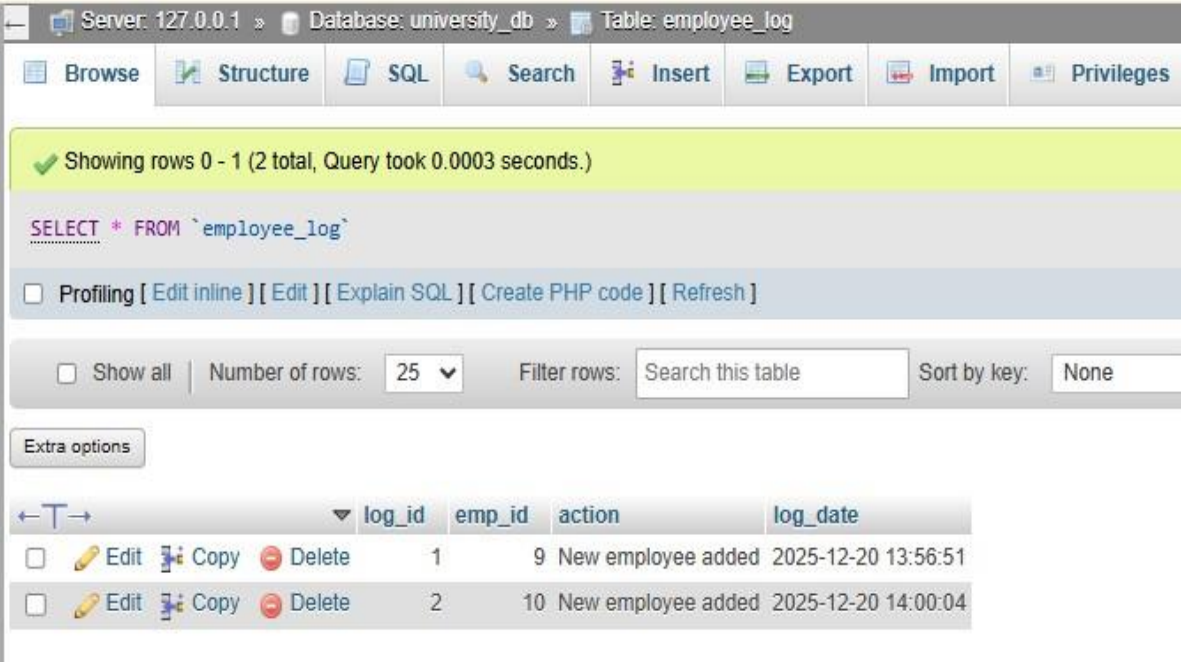
	emp_id	emp_name	department	salary
<input type="checkbox"/>	1	ayush	IT	30000
<input type="checkbox"/>	2	dhaval	HR	40000
<input type="checkbox"/>	3	jay	Finance	20000
<input type="checkbox"/>	5	param	IT	25000

Bottom controls: Check all, With selected: Edit, Copy, Delete, Export

15. SQL Triggers

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

- DELIMITER \$\$
CREATE TRIGGER after_emp_insert AFTER INSERT ON employees FOR EACH ROW
BEGIN
INSERT INTO employee_log (emp_id, action) VALUES (NEW.emp_id, 'New employee
added');
END \$\$
DELIMITER ;



The screenshot shows a database management interface with the following components:

- Header:** Server: 127.0.0.1 » Database: university_db » Table: employee_log
- Navigation Bar:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges
- Status Bar:** Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)
- SQL Editor:** `SELECT * FROM `employee_log``
- Actions:** Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Filters:** Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None
- Extra options:** (button)
- Table View:**

	log_id	emp_id	action	log_date
<input type="checkbox"/> Edit Copy Delete	1	9	New employee added	2025-12-20 13:56:51
<input type="checkbox"/> Edit Copy Delete	2	10	New employee added	2025-12-20 14:00:04

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

- CREATE TRIGGER before_emp_update BEFORE UPDATE ON employees FOR EACH ROW
BEGIN
 SET NEW.last_modified = NOW();
END;

Server: 127.0.0.1 » Database: university_db » Table: employees

Showing rows 0 - 4 (5 total, Query took 0.0003 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: No

Extra options

		emp_id	emp_name	department	salary	last_modified
<input type="checkbox"/>	Edit	1	ayush	IT	30000	2025-12-20 14:17:09
<input type="checkbox"/>	Edit	2	dhaval	HR	40000	2025-12-20 14:17:09
<input type="checkbox"/>	Edit	3	jay	Finance	20000	2025-12-20 14:17:09
<input type="checkbox"/>	Edit	4	meet	Manager	52000	2025-12-20 14:17:09
<input type="checkbox"/>	Edit	5	param	IT	25000	2025-12-20 14:17:09

Server: 127.0.0.1 » Database: university_db » Table: employees

Showing rows 0 - 4 (5 total, Query took 0.0004 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: No

Extra options

		emp_id	emp_name	department	salary	last_modified
<input type="checkbox"/>	Edit	1	ayush	IT	35000	2025-12-20 14:30:12
<input type="checkbox"/>	Edit	2	dhaval	HR	45000	2025-12-20 14:30:12
<input type="checkbox"/>	Edit	3	jay	Finance	25000	2025-12-20 14:30:12
<input type="checkbox"/>	Edit	4	meet	Manager	55000	2025-12-20 14:30:12
<input type="checkbox"/>	Edit	5	tilak	IT	30000	2025-12-20 14:30:12

16. Introduction to PL/SQL

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

```
- DELIMITER $$  
CREATE PROCEDURE total_employees()  
BEGIN  
    SELECT COUNT(*) AS total_employees  
    FROM employees;  
END $$  
DELIMITER ;  
CALL total_employees();
```

The screenshot shows a database management tool interface with the following components:

- Header:** Server: 127.0.0.1 » Database: university_db
- Navigation Bar:** Structure, SQL, Search, Query, Export, Import, Operations
- Buttons:** Show query box
- Message Bar:** ⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
- Execution Status:** ✓ Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)
- SQL Editor:** CALL total_employees();
[Edit inline] [Edit] [Create PHP code]
- Table Controls:** ☐ Show all | Number of rows: 25 ▼ | Filter rows: Search this table
- Extra options:** Extra options
- Table View:**

total_employees
5

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

```
- 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 a database management tool interface. At the top, the breadcrumb navigation indicates the path: Server: 127.0.0.1 > Database: university_db > Table: orders. Below this is a toolbar with buttons for Browse, Structure, SQL, Search, Insert, Export, and Import. A 'Show query box' button is located below the toolbar. A warning message states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled.' Below the warning, a green status bar indicates: 'Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)'. The SQL editor contains the PL/SQL block code: `CALL total_sales_amount();`. Below the editor are links for '[Edit inline]', '[Edit]', and '[Create PHP code]'. A control bar shows 'Show all' (unchecked), 'Number of rows: 25' (dropdown), and 'Filter rows: Search this table' (input field). An 'Extra options' button is at the bottom left. The results table has one row with the column 'total_sales' and the value '15000.00'.

total_sales
15000.00

17. PL/SQL Control Structures

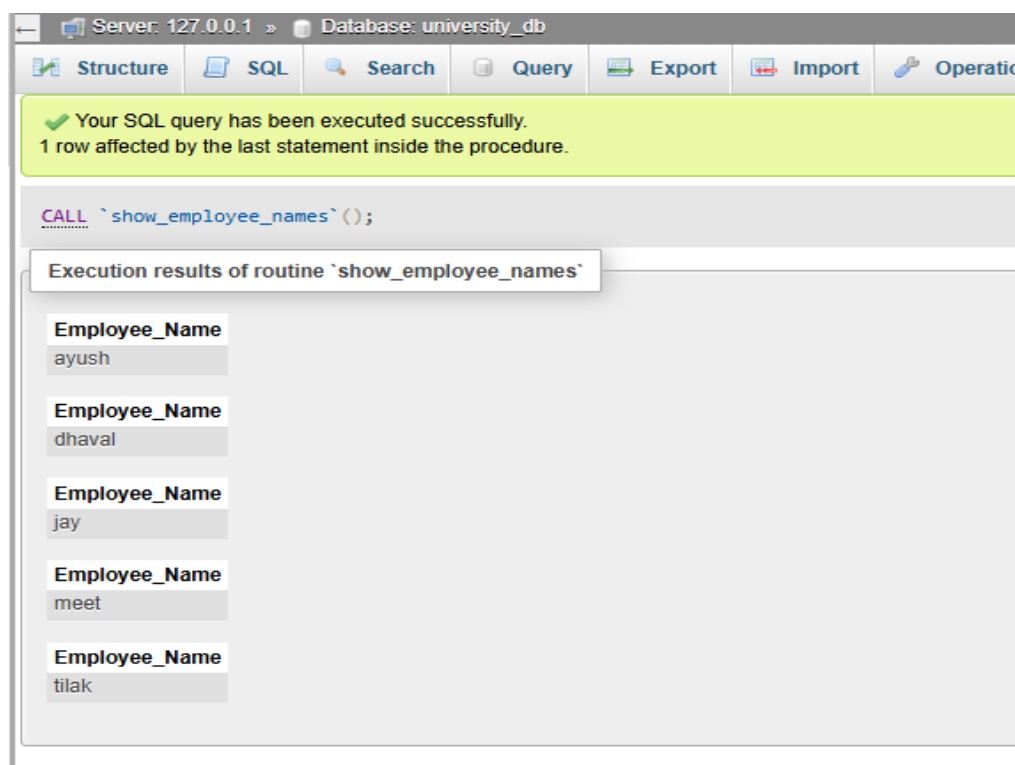
Lab 1 : Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

```
- DELIMITER $$
CREATE PROCEDURE check_department(IN p_emp_id INT)
BEGIN
  DECLARE v_department VARCHAR(50);
  SELECT department
  INTO v_department
  FROM employees
  WHERE emp_id = p_emp_id;
  IF v_department = 'IT' THEN
    SELECT 'Employee belongs to IT Department' AS Result;
  ELSE
    SELECT 'Employee does not belong to IT Department' AS Result;
  END IF;
END $$
DELIMITER ;
CALL check_department(2);
```

The screenshot shows a database management tool interface with a top toolbar containing buttons for Structure, SQL, Search, Query, Export, Import, and Operations. Below the toolbar is a 'Show query box' button. A warning message states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled.' A green status bar indicates: 'Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)'. The SQL editor contains the text: `CALL check_department(1);`. Below the editor are links for '[Edit inline]', '[Edit]', and '[Create PHP code]'. A control bar includes a 'Show all' checkbox, a 'Number of rows' dropdown set to 25, and a 'Filter rows' search box with the placeholder text 'Search this table'. An 'Extra options' button is located below the control bar. The 'Result' section at the bottom displays the output: 'Employee belongs to IT Department'.

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

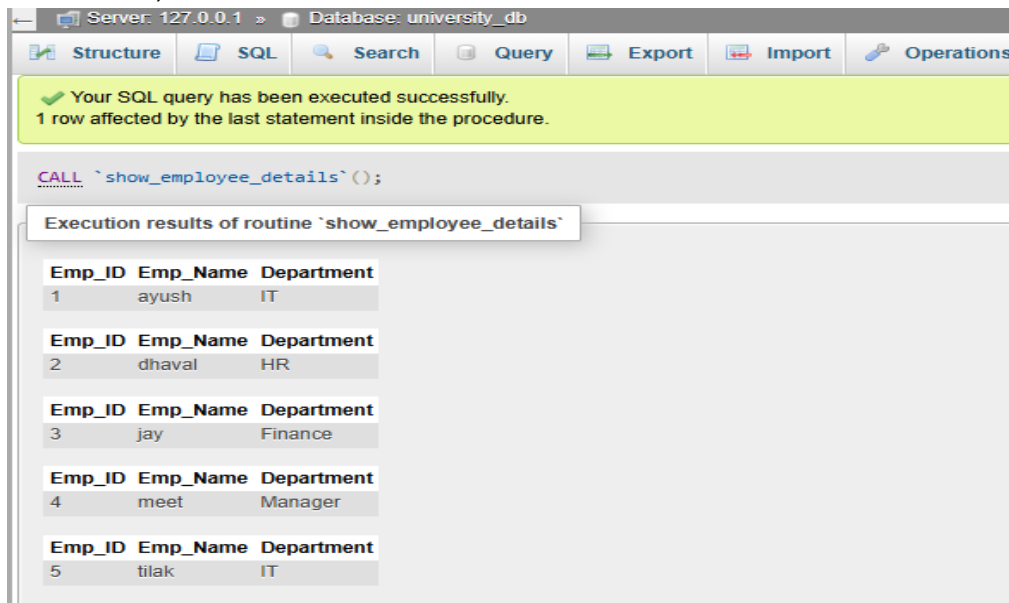
```
- DELIMITER $$  
CREATE PROCEDURE show_employee_names()  
BEGIN  
    DECLARE done INT DEFAULT 0;  
    DECLARE v_name VARCHAR(100);  
    DECLARE emp_cursor CURSOR FOR  
    SELECT emp_name FROM employees;  
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;  
    OPEN emp_cursor;  
    read_loop: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



18. SQL Cursors

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

```
- DELIMITER $$
CREATE PROCEDURE show_employee_details()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_id INT;
    DECLARE v_name VARCHAR(100);
    DECLARE v_dept VARCHAR(50);
    DECLARE emp_cursor CURSOR FOR
        SELECT emp_id, emp_name, department FROM employees;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN emp_cursor;
    emp_loop: LOOP
        FETCH emp_cursor INTO v_id, v_name, v_dept;
        IF done = 1 THEN
            LEAVE emp_loop;
        END IF;
        SELECT v_id AS Emp_ID,
            v_name AS Emp_Name,
            v_dept AS Department;
    END LOOP;
    CLOSE emp_cursor;
END$$
DELIMITER ;
```



Server: 127.0.0.1 > Database: university_db

Structure SQL Search Query Export Import Operations

✓ 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`

Emp_ID	Emp_Name	Department
1	ayush	IT

Emp_ID	Emp_Name	Department
2	dhaval	HR

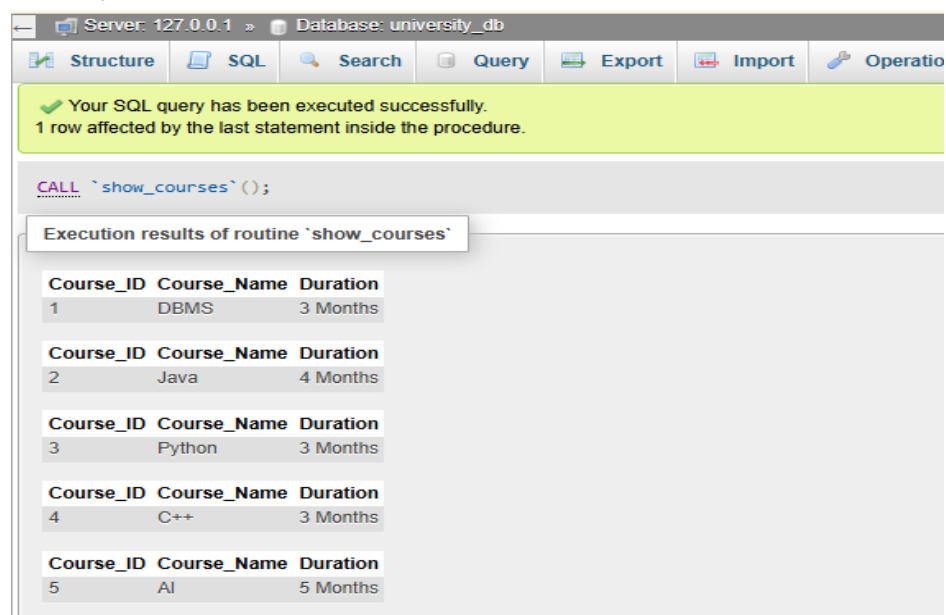
Emp_ID	Emp_Name	Department
3	jay	Finance

Emp_ID	Emp_Name	Department
4	meet	Manager

Emp_ID	Emp_Name	Department
5	tilak	IT

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

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



19. Rollback and Commit Savepoint

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

- START TRANSACTION;
INSERT INTO courses (course_name, course_duration)
VALUES ('Cloud', '4 Months');
SAVEPOINT sp1;
INSERT INTO courses (course_name, course_duration)
VALUES ('DevOps', '5 Months');
ROLLBACK TO sp1;
COMMIT;

The screenshot shows a database management tool interface. At the top, the breadcrumb navigation indicates 'Server: 127.0.0.1 > Database: university_db > Table: courses'. Below this is a toolbar with buttons for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'Import'. A 'Show query box' button is located below the toolbar. The main area displays a green status bar indicating 'Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)'. Below this, the SQL query 'SELECT * FROM `courses`;' is shown. A row of controls includes a 'Profiling' checkbox and links for 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. Below the query area, there are controls for 'Show all', 'Number of rows' (set to 25), and a 'Filter rows' search box. An 'Extra options' button is also present. At the bottom, a table view shows the 'courses' table with columns 'course_id', 'course_name', and 'course_duration'. The first row contains the values '1', 'Cloud', and '4 Months'. Action buttons for 'Edit', 'Copy', and 'Delete' are visible next to the first row.

course_id	course_name	course_duration
1	Cloud	4 Months

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

- START TRANSACTION;
INSERT INTO courses (course_name, course_duration)
VALUES ('Java Advanced', '5 Months');
SAVEPOINT sp2;
INSERT INTO courses (course_name, course_duration)
VALUES ('Spring Boot', '4 Months');
COMMIT;
START TRANSACTION;
INSERT INTO courses (course_name, course_duration)
VALUES ('Microservices', '6 Months');
ROLLBACK;

Server: 127.0.0.1 » Database: university_db » Table: courses

Browse Structure SQL Search Insert Export Import

Show query box

✓ Showing rows 0 - 2 (3 total, Query took 0.0004 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	course_name	course_duration
<input type="checkbox"/>	Edit	Copy	Delete	1 Cloud	4 Months
<input type="checkbox"/>	Edit	Copy	Delete	3 Java Advanced	5 Months
<input type="checkbox"/>	Edit	Copy	Delete	4 Spring Boot	4 Months