

# 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))

The screenshot shows the phpMyAdmin interface. On the left, the database tree shows 'school\_db' selected. In the main area, a message says 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)'. Below it is a query editor with the SQL command: 'SELECT \* FROM `students`'. The results pane is empty. At the bottom, there are options to bookmark the query.

**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`

The screenshot shows the phpMyAdmin interface after executing the SELECT statement. The results pane displays five records from the 'students' table:

	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	ahemddabad
<input type="checkbox"/>	5	jay	9	5th	gandhinagar

## 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 the phpMyAdmin interface for a database named 'school\_db' with a table named 'students'. The query executed was 'SELECT student\_name,age FROM students;'. The results show five rows of data:

	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

**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 phpMyAdmin interface for a database named 'school\_db' with a table named 'students'. The query executed was 'SELECT \* FROM `students` WHERE age>10;'. The results show four rows of data:

	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

### 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)

The screenshot shows the MySQL Workbench interface with the 'Table structure' tab selected for the 'teachers' table. The table has four columns: teacher\_id, teacher\_name, subject, and email. The 'teacher\_id' column is defined as int(11) with AUTO\_INCREMENT, primary key, and no nulls. The 'teacher\_name' and 'subject' columns are defined as varchar(30) with no nulls. The 'email' column is defined as varchar(30) with unique constraint. Below the table definition, there are various management buttons like Check all, With selected:, Browse, Change, Drop, Primary, Unique, Index, Spatial, Fulltext, Add to central columns, and Remove from central columns.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	teacher_id	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
2	teacher_name	varchar(30)	utf8mb4_general_ci		No	None			Change  Drop  More
3	subject	varchar(30)	utf8mb4_general_ci		No	None			Change  Drop  More
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);

The screenshot shows the MySQL Workbench interface with the 'Table structure' tab selected for the 'students' table. The table has six columns: student\_id, student\_name, age, class, address, and teacher\_id. The 'student\_id' column is defined as int(11) with AUTO\_INCREMENT, primary key, and no nulls. The 'student\_name' column is defined as varchar(20) with no nulls. The 'age' and 'class' columns are defined as int(11) with no nulls. The 'address' column is defined as varchar(20) with no nulls. The 'teacher\_id' column is defined as int(11) with no nulls. Below the table definition, there are various management buttons like Check all, With selected:, Browse, Change, Drop, Primary, Unique, Index, Spatial, Fulltext, Add to central columns, and Remove from central columns.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	student_id	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
2	student_name	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
3	age	int(11)			Yes	NULL			Change  Drop  More
4	class	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
5	address	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
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 the MySQL Workbench interface with the following details:

- Server: 127.0.0.1
- Database: university\_db
- Table: courses
- Table structure view is selected.
- Columns listed:

  - course\_id (int(11), Primary Key, AUTO\_INCREMENT)
  - course\_name (varchar(30))
  - course\_credits (varchar(30))

- Action buttons for each column: Change, Drop, More.
- Bottom buttons: Check all, With selected:, Browse, Change, Drop, Primary, Unique, Index, Spatial, Fulltext, Add to central columns, 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 MySQL Workbench interface with the following details:

- Server: 127.0.0.1
- Database: university\_db
- Table: courses
- Table structure view is selected.
- Columns listed:

  - course\_id (int(11), Primary Key, AUTO\_INCREMENT)
  - course\_name (varchar(30))
  - course\_duration (varchar(20))

- Action buttons for each column: Change, Drop, More.
- Bottom buttons: Check all, With selected:, Browse, Change, Drop, Primary, Unique, Index, Spatial, Fulltext, Add to central columns, Remove from central columns.

## 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

The screenshot shows the phpMyAdmin interface. On the left, the database tree is visible with 'school\_db' selected. The main area shows a green success message: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0012 seconds.)'. Below it is the SQL query: 'DROP TABLE students;'. There are buttons for 'Edit inline', 'Edit', and 'Create PHP code'.

## 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')

The screenshot shows the phpMyAdmin interface for the 'courses' table in the 'university\_db' database. The table has columns: course\_id, course\_name, and course\_duration. Two rows are present: one for 'java' with a duration of '6 to 8 months', and another for 'Flutter' with a duration of '5 to 7 months'. The 'Structure' tab is selected at the top.

course_id	course_name	course_duration
1	java	6 to 8 months
2	Flutter	5 to 7 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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Table:** courses
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import
- Message Bar:** Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)
- Query Bar:** SELECT \* FROM `courses`
- Table Options:** Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Table Row Selection:** Show all | Number of rows: 25 | Filter rows: Search this table | Sort by course\_id
- Extra Options:** Extra options button
- Data Table:** course\_id | course\_name | course\_duration

<input type="checkbox"/>	1	java	7 to 9 months
<input type="checkbox"/>	2	Flutter	5 to 7 months

- Action Buttons:** Up, 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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Table:** courses
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import
- Message Bar:** Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)
- Query Bar:** SELECT \* FROM `courses`
- Table Options:** Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Table Row Selection:** Show all | Number of rows: 25 | Filter rows: Search this table
- Extra Options:** Extra options button
- Data Table:** course\_id | course\_name | course\_duration

<input type="checkbox"/>	2	Flutter	5 to 7 months
--------------------------	---	---------	---------------

- Action Buttons:** Up, 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 the phpMyAdmin interface for the 'courses' table. The top navigation bar includes 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: courses'. Below the navigation are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', and 'Privileges'. A 'Show query box' button is visible. The main area displays a green success message: 'Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)'. The SQL query is shown as 'SELECT \* FROM `courses`;'. Below the query are buttons for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. A search bar and filter options are present. The data grid shows the following rows:

	course_id	course_name	course_duration
<input type="checkbox"/>	1	Java	7 to 9 months
<input type="checkbox"/>	2	flutter	6 to 8 months
<input type="checkbox"/>	3	python	10 to 12 months
<input type="checkbox"/>	4	software engineering	2 to 4 months
<input type="checkbox"/>	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 phpMyAdmin interface for the 'courses' table. The top navigation bar includes 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: courses'. Below the navigation are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', and 'Privileges'. A green success message at the top states: 'Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.) [course\_duration: 8 TO 9 MONTHS... - 2 TO 4 MONTHS...]' followed by the SQL query 'SELECT \* FROM `courses` ORDER BY `course\_duration` DESC'. Below the query are buttons for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. A search bar and filter options are present. The data grid shows the following rows, sorted by course\_duration in descending order:

	course_id	course_name	course_duration
<input type="checkbox"/>	3	python	8 to 9 months
<input type="checkbox"/>	1	Java	7 to 9 months
<input type="checkbox"/>	2	flutter	6 to 8 months
<input type="checkbox"/>	5	php	4 to 6 months
<input type="checkbox"/>	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 the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Table:** courses
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges.
- Show query box:** Shows the executed SQL query: `SELECT * FROM `courses` ORDER BY course_duration DESC LIMIT 2;`
- Profiling:** Options to edit inline, edit, explain SQL, create PHP code, and refresh.
- Extra options:** Includes check all, with selected, edit, copy, delete, and export buttons.
- Data Table:** Displays the results of the query:

course_id	course_name	course_duration
3	python	8 to 9 months
1	Java	7 to 9 months

## 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
Any	%	No	USAGE		No	Edit privileges Export Lock
pma	localhost	No	USAGE		No	Edit privileges Export Lock
root	127.0.0.1	No	ALL PRIVILEGES		Yes	Edit privileges Export Lock
root	:1	No	ALL PRIVILEGES		Yes	Edit privileges Export Lock
root	localhost	No	ALL PRIVILEGES		Yes	Edit privileges Export Lock
user1	localhost	Yes	USAGE		No	Edit privileges Export Lock
user2	localhost	Yes	USAGE		No	Edit privileges Export Lock

Server: 127.0.0.1 » Database: university\_db » Table: courses

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';

The screenshot shows the phpMyAdmin interface for the 'courses' table in the 'university\_db' database. The top navigation bar shows 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: courses'. Below the navigation bar is a toolbar with 'Browse', 'Structure', 'SQL', 'Search', 'Insert', and 'Export' buttons. A 'Show query box' button is also present. A warning message in a red box states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled.' A green box below displays the message: 'Your SQL query has been executed successfully.' The SQL query shown is: 'SHOW GRANTS FOR 'user1'@'localhost'';'. Under the grants section, it shows: 'Grants for user1@localhost' followed by two grant entries: 'GRANT USAGE ON \*.\* TO 'user1'@'localhost' IDENTIFI...' and 'GRANT SELECT ON `university\_db`.`courses` TO 'user...''.

The screenshot shows the phpMyAdmin interface for the 'courses' table in the 'university\_db' database. The top navigation bar shows 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: courses'. Below the navigation bar is a toolbar with 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'Import' buttons. A 'Show query box' button is also present. A warning message in a red box states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled.' A green box below displays the message: 'Your SQL query has been executed successfully.' The SQL query shown is: 'SHOW GRANTS FOR 'user2'@'localhost'';'. Under the grants section, it shows: 'Grants for user2@localhost' followed by two grant entries: 'GRANT USAGE ON \*.\* TO 'user2'@'localhost' IDENTIFI...' and '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 the phpMyAdmin interface for a database named 'university\_db' with the table 'courses'. The table has columns: course\_id, course\_name, and course\_duration. Three rows have been inserted:

course_id	course_name	course_duration
1	DBMS	3 Months
2	Java	4 Months
3	Python	3 Months

The SQL query executed was: `SELECT * FROM courses;`

**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 the MySQL Workbench interface with the 'courses' table selected. The table has columns: course\_id, course\_name, and course\_duration. The data is as follows:

|                          | 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 MySQL Workbench interface with the 'courses' table selected. The table has columns: course\_id, course\_name, and course\_duration. The data is as follows:

	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 the phpMyAdmin interface for the 'employees' table in the 'university\_db'. The query executed was:

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

The results show four rows of data:

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 the phpMyAdmin interface for the 'departments' table in the 'university\_db'. The query executed was:

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

The results show five rows of data, including departments with no employees:

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 the phpMyAdmin interface for the 'employees' table. The top navigation bar includes 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: employees'. Below the navigation are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', and 'Privileges'. A 'Show query box' button is visible. The main area displays a green status message: 'Showing rows 0 - 3 (4 total, Query took 0.0006 seconds.)'. The SQL query is: 'SELECT department,COUNT(emp\_id) AS total\_emp FROM employees GROUP BY department;'. Below the query are buttons for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. A row selection tool allows 'Show all' or 'Number of rows: 25'. A 'Filter rows' search bar is present. The data table has columns 'department' and 'total\_emp'. The data is:

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 phpMyAdmin interface for the 'employees' table. The top navigation bar includes 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: employees'. Below the navigation are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', and 'Privileges'. A 'Show query box' button is visible. The main area displays a green status message: 'Showing rows 0 - 3 (4 total, Query took 0.0136 seconds.)'. The SQL query is: 'SELECT department, AVG(salary) as avg\_salary FROM employees GROUP BY department;'. Below the query are buttons for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. A row selection tool allows 'Show all' or 'Number of rows: 25'. A 'Filter rows' search bar is present. The data table has columns 'department' and 'avg\_salary'. The data is:

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 the MySQL Workbench interface. The top bar indicates the connection is to 'Server: 127.0.0.1' and the database is 'university\_db'. The tabs at the top include Structure, SQL, Search, Query, Export, Import, and Operations. A message box in the center says 'Your SQL query has been executed successfully.' and '2 rows affected by the last statement inside the procedure.' Below this, the SQL query 'SET @p0='manager'; CALL `get\_emp\_proc`(@p0);' is shown. A results pane titled 'Execution results of routine `get\_emp\_proc`' displays a table with four columns: emp\_id, emp\_name, department, and salary. The data is as follows:

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

## Routines

The screenshot shows the 'Routines' section of MySQL Workbench. At the top, there are buttons for 'Check all', 'Export', and 'Drop'. Below is a table with columns: Name, Type, and Returns. One row is visible: 'get\_emp\_proc' of type 'PROCEDURE'. To the right of the table are buttons for 'Edit', 'Execute', 'Export', and 'Drop'.

Name	Type	Returns
get_emp_proc	PROCEDURE	

**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 the MySQL Workbench interface. The top bar displays 'Server: 127.0.0.1 » Database: university\_db'. Below the bar are tabs for Structure, SQL, Search, Query, Export, Import, and Operations. A green message box indicates: 'Your SQL query has been executed successfully. 1 row affected by the last statement inside the procedure.' The SQL pane contains the code: 'SET @p0='2'; CALL `course\_detail\_proc`(@p0);'. The results pane shows the output of the procedure:

course_id	course_name	course_duration
2	flutter	6 to 8 months

## Routines

The screenshot shows the 'Routines' section of MySQL Workbench. At the top are buttons for 'Check all', 'Export', and 'Drop'. Below is a table listing two procedures:

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

## 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 the MySQL Workbench interface with the following details:

- Server: 127.0.0.1
- Database: university\_db
- View: emp\_with\_dept\_view
- Toolbar buttons: Browse, Structure, SQL, Search, Insert, Export, Privileges.
- Message bar: "Current selection does not contain a unique column. Grid edit, Edit, Copy and Delete features may result in un..."
- Status bar: "Showing rows 0 - 7 (8 total, Query took 0.0004 seconds.)"
- SQL pane: "SELECT \* FROM `emp\_with\_dept\_view`"
- Buttons: Profiling, Edit inline, Edit, Explain SQL, Create PHP code, Refresh.
- Table pane:

	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
- Buttons below table: Show all, Number of rows: 25, Filter rows: Search this table, Extra options, 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 the MySQL Workbench interface with the following details:

- Server: 127.0.0.1
- Database: university\_db
- View: emp\_salary\_view
- Toolbar buttons: Browse, Structure, SQL, Search, Insert, Export, Privileges.
- Message bar: "Current selection does not contain a unique column. Grid edit, Edit, Copy and Delete features may result in un..."
- Status bar: "Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)"
- SQL pane: "SELECT \* FROM `emp\_salary\_view`"
- Buttons: Profiling, Edit inline, Edit, Explain SQL, Create PHP code, Refresh.
- Table pane:

	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
- Buttons below table: Show all, Number of rows: 25, Filter rows: Search this table, Extra options, 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 the phpMyAdmin interface for the 'employee\_log' table. The table has four columns: log\_id, emp\_id, action, and log\_date. Two rows are displayed, both showing 'New employee added' as the action and a timestamp around December 20, 2025.

log_id	emp_id	action	log_date
1	9	New employee added	2025-12-20 13:56:51
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

Browse Structure SQL Search Insert Export Import P

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: Nc

Extra options

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

Server: 127.0.0.1 » Database: university\_db » Table: employees

Browse Structure SQL Search Insert Export Import P

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: Nc

Extra options

	emp_id	emp_name	department	salary	last_modified
<input type="checkbox"/>	1	ayush	IT	35000	2025-12-20 14:30:12
<input type="checkbox"/>	2	dhaval	HR	45000	2025-12-20 14:30:12
<input type="checkbox"/>	3	jay	Finance	25000	2025-12-20 14:30:12
<input type="checkbox"/>	4	meet	Manager	55000	2025-12-20 14:30:12
<input type="checkbox"/>	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 the MySQL Workbench interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations
- Message Bar:** Shows a warning: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available."
- Status Bar:** Shows "Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)"
- Query Editor:** Contains the SQL code: `CALL total_employees();`
- Buttons:** [Edit inline], [Edit], [Create PHP code]
- Table View:** A single row named "total\_employees" with the value "5".
- Filter:** Number of rows: 25, Filter rows: Search this table

**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 the MySQL Workbench interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Table:** orders
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import
- Show query box:** A button labeled "Show query box".
- Warning message:** "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete feature are disabled." (with a warning icon).
- Success message:** "Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)" (with a green checkmark icon).
- SQL Query:** `CALL total_sales_amount();`
- Action buttons:** [Edit inline] [Edit] [Create PHP code]
- Table Row:** **total\_sales** 15000.00
- Filter:** Show all, Number of rows: 25, Filter rows: Search this table
- Extra options:** A button labeled "Extra options".

## 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 the MySQL Workbench interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations
- Show query box:** A button to open a query editor.
- Message Bar:** A warning message: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled." followed by a success message: "Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)".
- Query Editor:** Contains the PL/SQL code: `CALL check_department(1);`
- Action Buttons:** [Edit inline] [Edit] [Create PHP code]
- Table Filter:** Includes "Show all", "Number of rows: 25", and a search bar "Filter rows: Search this table".
- Extra Options:** A button labeled "Extra options".
- Result:** A section titled "Result" displaying 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 ;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Execution Results:**
  - Your SQL query has been executed successfully.
  - 1 row affected by the last statement inside the procedure.
- SQL Query:** CALL `show\_employee\_names`();
- Execution Results of routine `show\_employee\_names`:**

Employee_Name
ayush
dhaval
jay
meet
tilak

## 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 ;

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Execution Results:** The query `CALL `show_employee_details`();` was executed successfully, affecting 1 row.
- Table Output:** The results are displayed in a table format with columns: Emp\_ID, Emp\_Name, and Department. The data is as follows:

Emp_ID	Emp_Name	Department
1	ayush	IT
2	dhaval	HR
3	jay	Finance
4	meet	Manager
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 ;

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations
- Status Bar:** Your SQL query has been executed successfully. 1 row affected by the last statement inside the procedure.
- Query Editor:** CALL `show\_courses`();
- Execution Results:** A table titled "Execution results of routine 'show\_courses'" showing the following data:

Course_ID	Course_Name	Duration
1	DBMS	3 Months
2	Java	4 Months
3	Python	3 Months
4	C++	3 Months
5	AI	5 Months

## 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 the phpMyAdmin interface for a MySQL database named 'university\_db'. The current table is 'courses'. The top navigation bar includes 'Server: 127.0.0.1', 'Database: university\_db', and 'Table: courses'. Below the navigation are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'Import'. A 'Show query box' button is visible. The main area displays a green success message: 'Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)'. The SQL query shown is 'SELECT \* FROM `courses`;'. Below the query, there are options for 'Profiling' and other actions. The bottom part of the interface shows the table structure with columns: course\_id, course\_name, and course\_duration. One row is displayed: course\_id 1, course\_name Cloud, and course\_duration 4 Months. There are buttons for 'Edit', 'Copy', and 'Delete'.

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;

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** university\_db
- Table:** courses
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import
- Show query box:** A button to show the query input area.
- Query Result:** Shows the result of the query: "Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)". The query is: "SELECT \* FROM `courses`;".
- Actions:** Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Table Options:** Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: (dropdown menu)
- Extra options:** A button to show additional options.
- Table Data:** A grid showing the 'courses' table with 3 rows:

	course_id	course_name	course_duration
<input type="checkbox"/>	1	Cloud	4 Months
<input type="checkbox"/>	3	Java Advanced	5 Months
<input type="checkbox"/>	4	Spring Boot	4 Months

Each row has Edit, Copy, and Delete buttons.