**Assignment**

**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.**

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

**Command :**

**- > CREATE DATABASE school\_db;**

**- > CREATE TABLE students(**

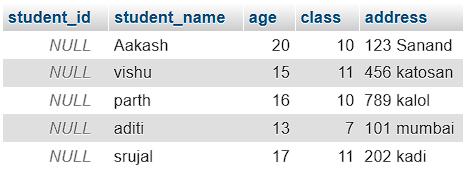
**student\_id int,**

**student\_name varchar(30),**

**age int , class int ,**

**address text**

**);**

****

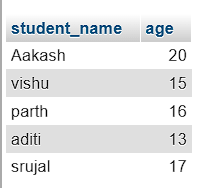
1. **SQL Syntax**

**• Lab 1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.**

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

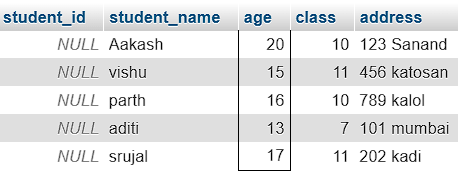
**Command:**

**-> SELECT student\_name, age FROM students;**

-> 

**-> SELECT \* FROM students**

**WHERE age > 10;**

****

**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).**

**• Lab 2: Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.**

* **Ans.**

**CREATE TABLE teachers (**

**teacher\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**teacher\_name VARCHAR(100) NOT NULL,**

**subject VARCHAR(100) NOT NULL,**

**email VARCHAR(100) UNIQUE**

**);**

**Picture2**

**Command :**

**CREATE TABLE students(**

**student\_id int PRIMARY KEY,**

**student\_name varchar(30),**

**age int , class int ,**

**address text,**

**teacher\_id int,**

**FOREIGN KEY(teacher\_id) REFERENCES students(teacher\_id)**

**);**



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

**• Lab 1: Create a table courses with columns: course\_id, course\_name, and course\_credits. Set the course\_id as the primary key.**

**• Lab 2: Use the CREATE command to create a database university\_db.**

**Command :**

CREATE TABLE courses (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL,

course\_credits INT NOT NULL

);



**Command :**

**CREATE DATABASE university\_db;**

**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.**

* **Ans.**

**Command:**

ALTER TABLE courses ADD course\_duration INT NOT NULL;

****

**Command :**

**ALTER TABLE courses**

**DROP COLUMN course\_credits;**

****

**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.**

**Command :**

**DROP TABLE IF EXISTS teachers;**

Screenshot 2025-01-22 172024

**Command :**

**DROP TABLE IF EXISTS students;**

**7. Data Manipulation Language (DML)**

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

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

**• Lab 3: Delete a course with a specific course\_id from the courses table using the DELETE command.**

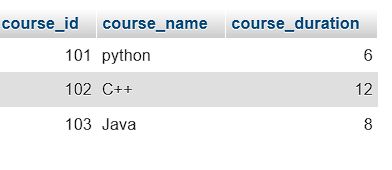
**Command:**

INSERT INTO courses VALUES(

101,'python','6 month'),

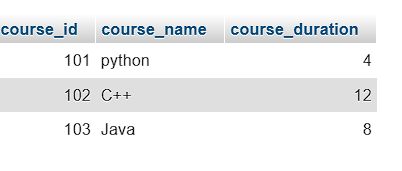
(102,'C++','12 month'),

(103,'Java','8 month');



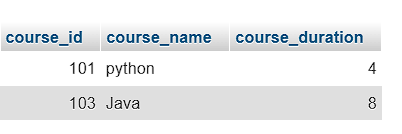
**Command :**

UPDATE courses SET course\_duration=4 WHERE course\_name=’python’;



**Command:**

DELETE FROM courses WHERE course\_id=102;



**8. Data Query Language (DQL)**

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

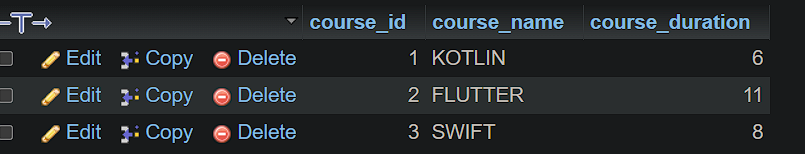
**• Lab 2: Sort the courses based on course\_duration in descending order using ORDER BY.**

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

* **Ans.**

**Command:**

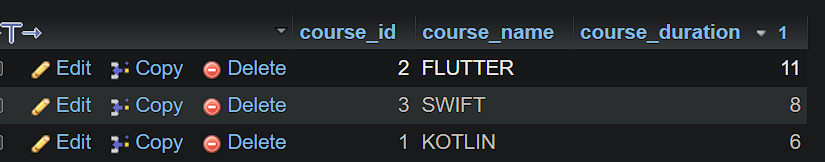
[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html" \t "http://localhost/phpmyadmin/index.php?route=/table/mysql_doc) \* FROM courses;

****

**Command:**

SELECT \* FROM courses

ORDER BY course\_duration DESC;

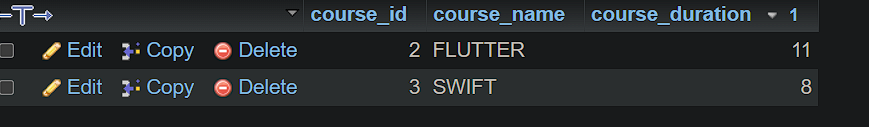


Command:

SELECT \* FROM courses

ORDER BY course\_duration DESC

LIMIT 2;



**11. SQL Joins**

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

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

* **Ans.**

**Command:**

**CREATE TABLE departments (**

**department\_id INT PRIMARY KEY,**

**department\_name VARCHAR(100)**

**);**

**CREATE TABLE employees (**

**employee\_id INT PRIMARY KEY,**

**employee\_name VARCHAR(100),**

**department\_id INT,**

**FOREIGN KEY (department\_id) REFERENCES departments(department\_id)**

**);**

**Command:**

**SELECT employees.employee\_name, departments.department\_name**

**FROM employees**

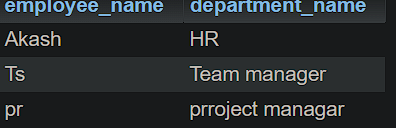
**INNER JOIN departments ON employees.department\_id = departments.department\_id;**

**Command:**

**SELECT departments.department\_name, employees.employee\_name**

**FROM departments**

**LEFT JOIN employees ON departments.department\_id = employees.department\_id;**

****

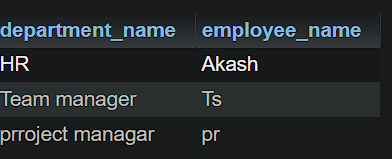
**Inner join**

**Commanmd:**

**SELECT departments.department\_name, employees.employee\_name**

**FROM departments**

**LEFT JOIN employees ON departments.department\_id = employees.department\_id;**



**12. SQL Group By**

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

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

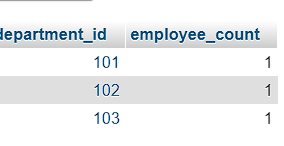
* **Ans :**

**Command:**

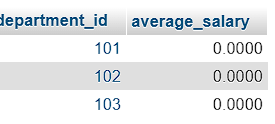
**SELECT department\_id, COUNT(employee\_id) AS employee\_count**

**FROM employees**

**GROUP BY department\_id;**

****

**Command:**

****

**13. SQL Stored Procedure**

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

**• Lab 2: Write a stored procedure that accepts course\_id as input and returns the course details.**

**Command:**

DELIMITER $$

CREATE PROCEDURE GetEmployeesByDepartment(IN dep\_id INT)

BEGIN

SELECT

emp\_id,

emp\_name,

emp\_gander,

emp\_salary,

dep\_id

FROM

employees

WHERE

dep\_id = dept\_id;

END $$

DELIMITER ;

**Command:**

CREATE PROCEDURE GetCourseDetails(IN input\_course\_id INT)

BEGIN

SELECT

course\_id,

course\_name,

course\_duration

FROM

courses

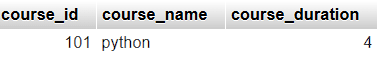
WHERE

course\_id = input\_course\_id;

END $$

DELIMITER ;

[CALL](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/call.html" \t "mysql_doc) GetCourseDetails(101);



**14. SQL View**

LAB EXERCISES:

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

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

**Command:**

CREATE VIEW EmployeeDepartmentView AS

SELECT

e.emp\_id,

e.emp\_name,

e.emp\_gander,

e.emp\_salary,

d.dep\_id,

d.dep\_name

FROM

employees e

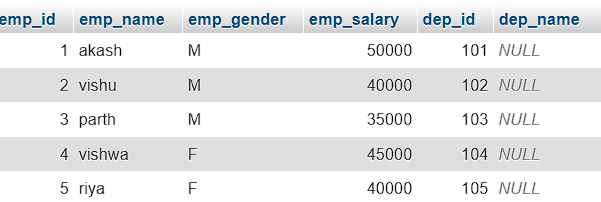
JOIN

departments d

ON

e.dep\_id = d.dep\_id;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html" \t "mysql_doc) \* FROM EmployeeDepartmentView;



**Command:**

CREATE OR REPLACE VIEW EmployeeDepartmentView AS

SELECT

e.emp\_id,

e.emp\_name,

e.emp\_gander,

e.emp\_salary,

d.dep\_id,

d.dep\_name

FROM

employees e

JOIN

departments d

ON

e.dep\_id = d.dep\_id

WHERE

e.emp\_salary >= 34000;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html" \t "mysql_doc) \* FROM EmployeeDepartmentView;

****

**15. SQL Triggers**

LAB EXERCISES:

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

**• Lab 2: Create a trigger to update the last\_modified timestamp whenever an employee record is updated.**

**Command:**

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html" \t "mysql_doc) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html" \t "mysql_doc) employee\_changes\_log ( log\_id INT AUTO\_INCREMENT PRIMARY KEY, employee\_id INT [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not" \t "mysql_doc) NULL, change\_type VARCHAR(50) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not" \t "mysql_doc) NULL, change\_date TIMESTAMP [DEFAULT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_default" \t "mysql_doc) [CURRENT\_TIMESTAMP](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp" \t "mysql_doc), details TEXT );

DELIMITER $$

CREATE TRIGGER after\_employee\_insert

AFTER INSERT ON employees

FOR EACH ROW

BEGIN

INSERT INTO employee\_changes\_log (

employee\_id,

change\_type,

details

)

VALUES (

NEW.emp\_id,

'INSERT',

CONCAT('New employee added: ', NEW.emp\_name, ' ',

', Department ID: ', NEW.dep\_id,

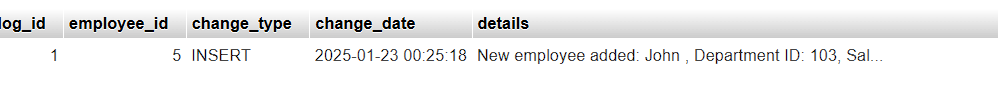
', Salary: $', NEW.emp\_salary)

);

END $$

DELIMITER ;

INSERT INTO employees (emp\_id, emp\_name, dep\_id, emp\_salary) VALUES (4, 'John', 101, 60000);

****

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html" \t "mysql_doc) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html" \t "mysql_doc) employees ADD COLUMN last\_modified TIMESTAMP [DEFAULT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_default" \t "mysql_doc) [CURRENT\_TIMESTAMP](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp" \t "mysql_doc) ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html" \t "mysql_doc) [CURRENT\_TIMESTAMP](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp" \t "mysql_doc);

DELIMITER $$

CREATE TRIGGER before\_employee\_update

BEFORE UPDATE ON employees

FOR EACH ROW

BEGIN

SET NEW.last\_modified = CURRENT\_TIMESTAMP;

END $$

DELIMITER ;

UPDATE employees

SET emp\_salary = 65000

WHERE emp\_id = 5;

