Assignment

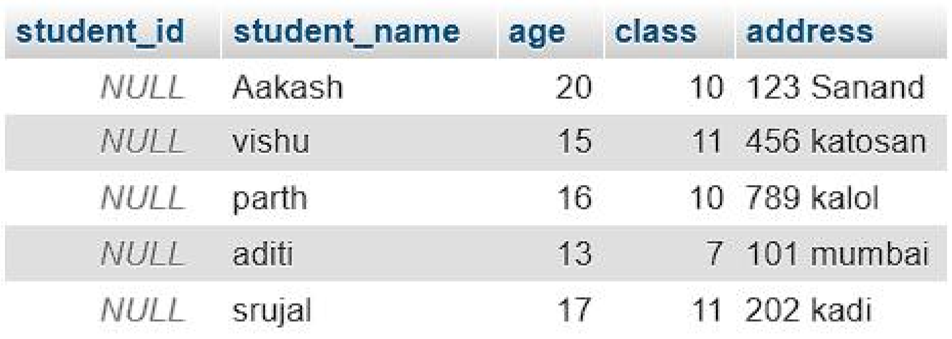
Introduction to SQL

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
 );
```



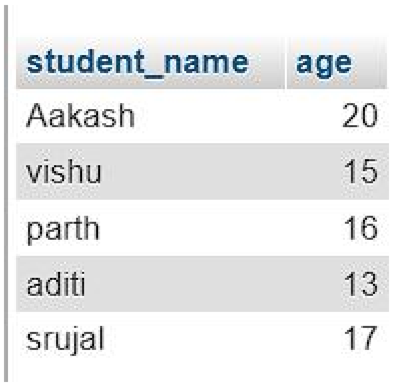
student_id	student_name	age	class	address
NULL	Aakash	20	10	123 Sanand
NULL	vishu	15	11	456 katosan
NULL	parth	16	10	789 kalol
NULL	aditi	13	7	101 mumbai
NULL	srujal	17	11	202 kadi

2. 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;

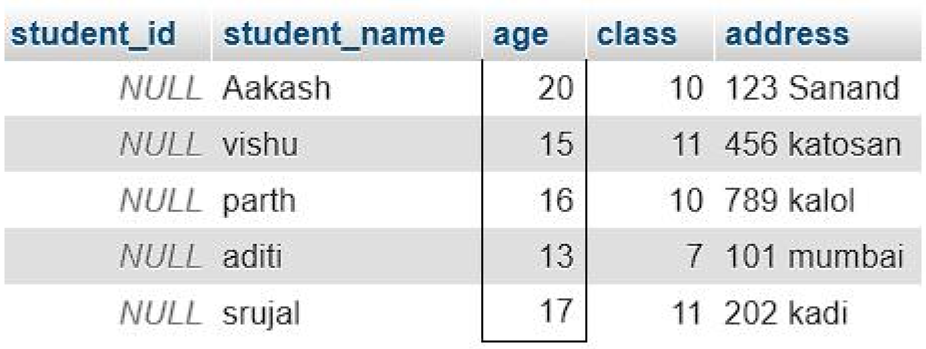


student_name	age
Aakash	20
vishu	15
parth	16
aditi	13
srujal	17

->

-> SELECT * FROM students

WHERE age > 10;



student_id	student_name	age	class	address
NULL	Aakash	20	10	123 Sanand
NULL	vishu	15	11	456 katosan
NULL	parth	16	10	789 kalol
NULL	aditi	13	7	101 mumbai
NULL	srujal	17	11	202 kadi

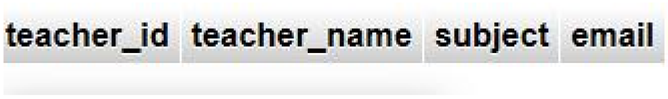
3. SQL Constraints

• 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.**

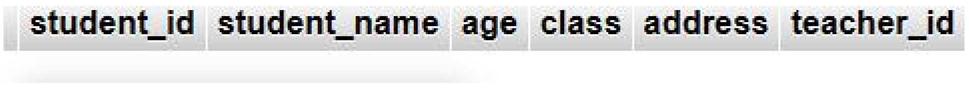
```
CREATE TABLE teachers (
   teacher_id INT AUTO_INCREMENT PRIMARY KEY,
   teacher_name VARCHAR(100) NOT NULL,
   subject VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE
);
```



teacher_id	teacher_name	subject	email

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)
);
```



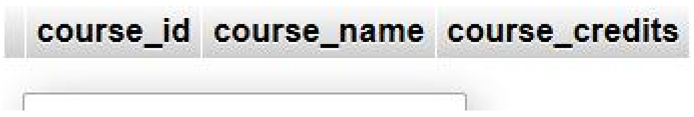
student_id	student_name	age	class	address	teacher_id

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
);
```



course_id	course_name	course_credits

Command :

```
CREATE DATABASE university_db;
```

5. ALTER Command

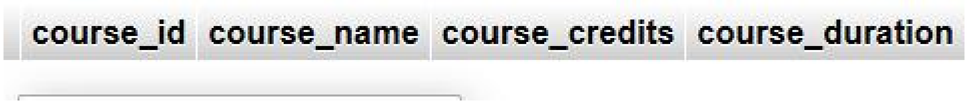
• **Lab 1: Modify the courses table by adding a column course_duration using the ALTER command.**

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

➢ **Ans.**

Command:

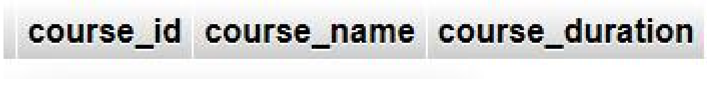
```
ALTER TABLE courses ADD course_duration INT NOT NULL;
```



course_id	course_name	course_credits	course_duration

Command :

```
ALTER TABLE courses
DROP COLUMN course_credits;
```



course_id	course_name	course_duration

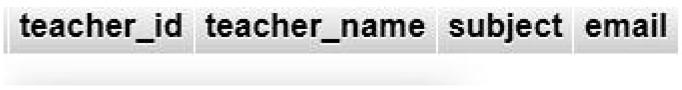
6. DROP Command

• **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;



teacher_id	teacher_name	subject	email

Command :

DROP TABLE IF EXISTS students;

7. Data Manipulation Language (DML)

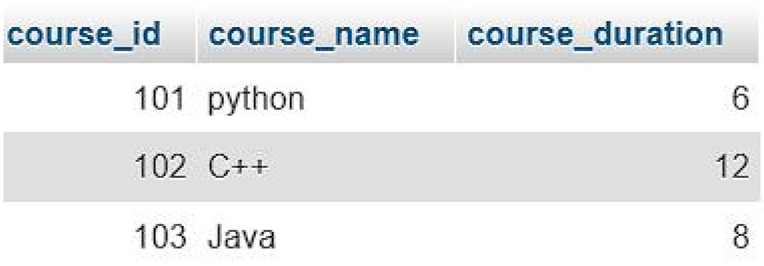
• Lab 1: Insert three records into the courses table using the INSERT command.

• 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:

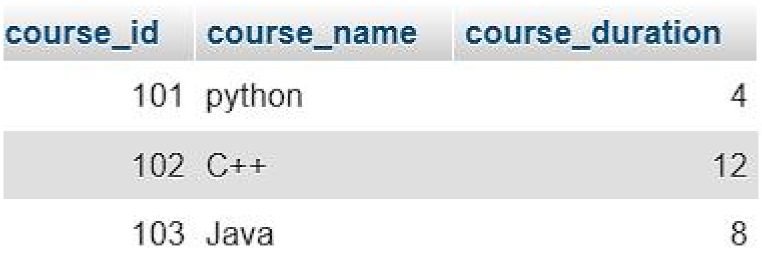
```
INSERT INTO courses VALUES(
    101,'python','6 month'),
    (102,'C++','12 month'),
    (103,'Java','8 month');
```



course_id	course_name	course_duration
101	python	6
102	C++	12
103	Java	8

Command :

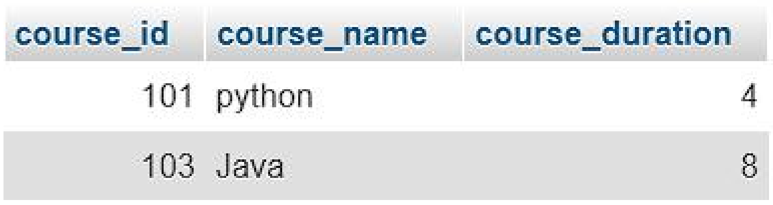
```
UPDATE courses SET course_duration=4 WHERE course_name='python';
```



course_id	course_name	course_duration
101	python	4
102	C++	12
103	Java	8

Command:

```
DELETE FROM courses WHERE course_id=102;
```



course_id	course_name	course_duration
101	python	4
103	Java	8

8. Data Query Language (DQL)

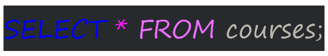
• Lab 1: Retrieve all courses from the courses table using the SELECT statement.

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

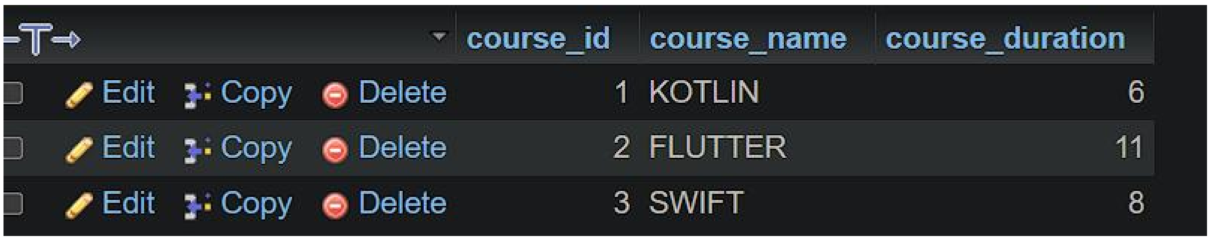
• Lab 3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

➢ **Ans.**

Command:



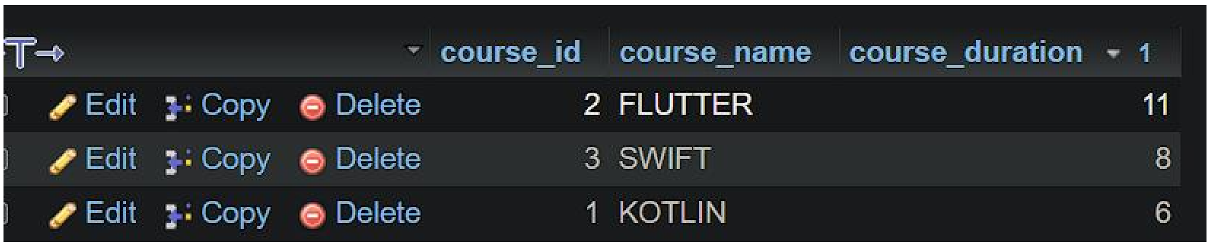
```
SELECT * FROM courses;
```



			course_id	course_name	course_duration
Edit	Copy	Delete	1	KOTLIN	6
Edit	Copy	Delete	2	FLUTTER	11
Edit	Copy	Delete	3	SWIFT	8

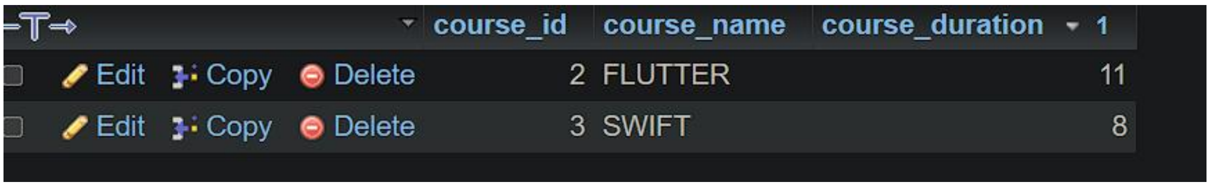
Command:

```
SELECT * FROM courses

ORDER BY course_duration DESC;
```



course_id	course_name	course_duration
2	FLUTTER	11
3	SWIFT	8
1	KOTLIN	6

```
Command:

SELECT * FROM courses

ORDER BY course_duration DESC

LIMIT 2;
```



course_id	course_name	course_duration
2	FLUTTER	11
3	SWIFT	8

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

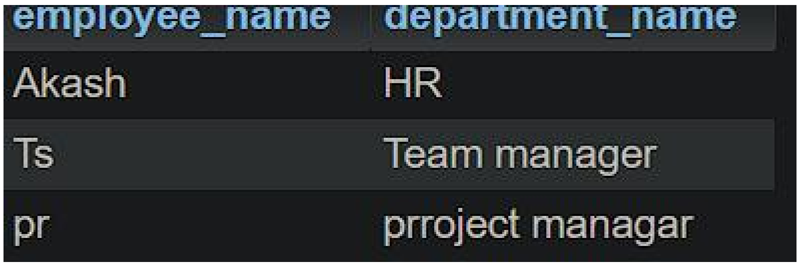
INNER JOIN departments ON employees.department_id = departments.department_id;

Command:

SELECT departments.department_name, employees.employee_name

FROM departments

LEFT JOIN employees ON departments.department_id = employees.department_id;



employee_name	department_name
Akash	HR
Ts	Team manager
pr	prroject managar

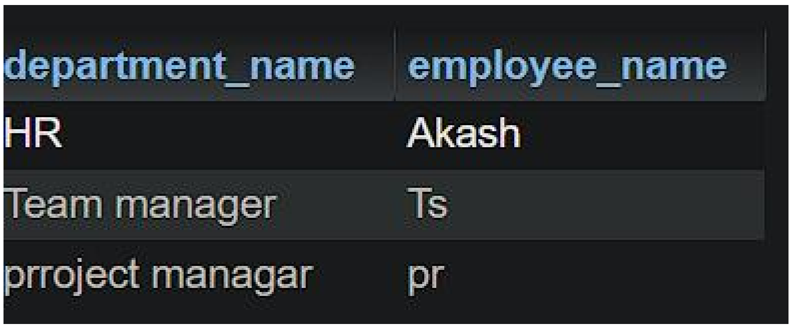
Inner join

Commanmd:

SELECT departments.department_name, employees.employee_name

FROM departments

LEFT JOIN employees ON departments.department_id = employees.department_id;



department_name	employee_name
HR	Akash
Team manager	Ts
prroject managar	pr

12. SQL Group By

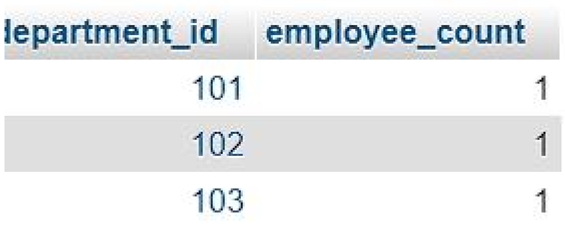
• Lab 1: Group employees by department and count the number of employees in each department using GROUP BY.

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

➢ **Ans :**

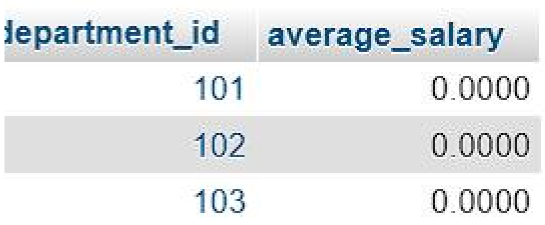
Command:

```
SELECT department_id, COUNT(employee_id) AS employee_count
FROM employees
GROUP BY department_id;
```



lepartment_id	employee_count
101	1
102	1
103	1

Command:



lepartment_id	average_salary
101	0.0000
102	0.0000
103	0.0000

13. SQL Stored Procedure

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

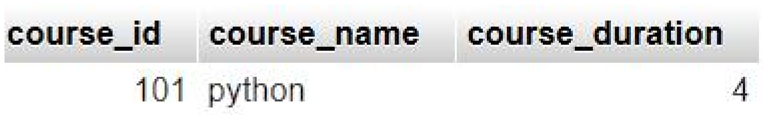
• **Lab 2: Write a stored procedure that accepts course_id as input and returns the course details.**

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 GetCourseDetails(101);
```



course_id	course_name	course_duration
101	python	4

14. SQL View

LAB EXERCISES:

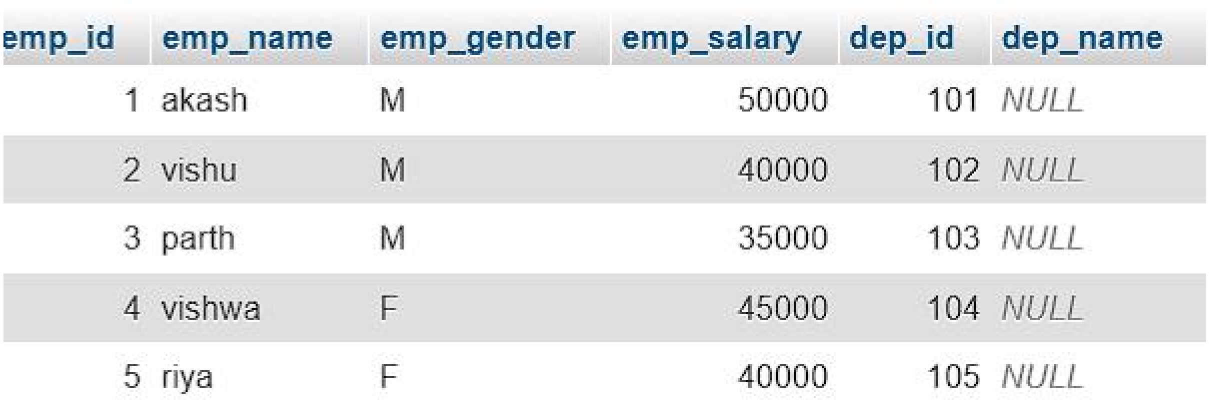
- **Lab 1: Create a view to show all employees along with their department names.**
- **Lab 2: Modify the view to exclude employees whose salaries are below $50,000.**

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 * FROM EmployeeDepartmentView;
```



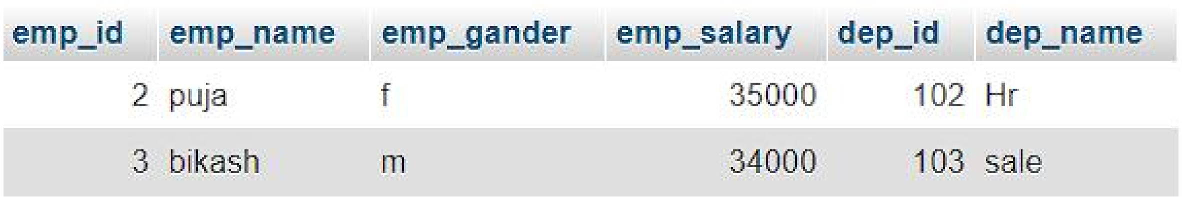
emp_id	emp_name	emp_gender	emp_salary	dep_id	dep_name
1	akash	M	50000	101	*NULL*
2	vishu	M	40000	102	*NULL*
3	parth	M	35000	103	*NULL*
4	vishwa	F	45000	104	*NULL*
5	riya	F	40000	105	*NULL*

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 * FROM EmployeeDepartmentView;
```



emp_id	emp_name	emp_gander	emp_salary	dep_id	dep_name
2	puja	f	35000	102	Hr
3	bikash	m	34000	103	sale

15. SQL Triggers

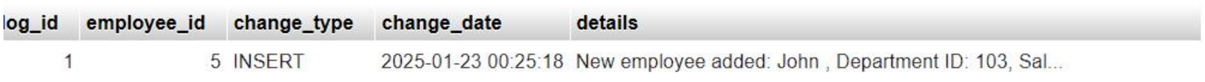
LAB EXERCISES:

• 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.

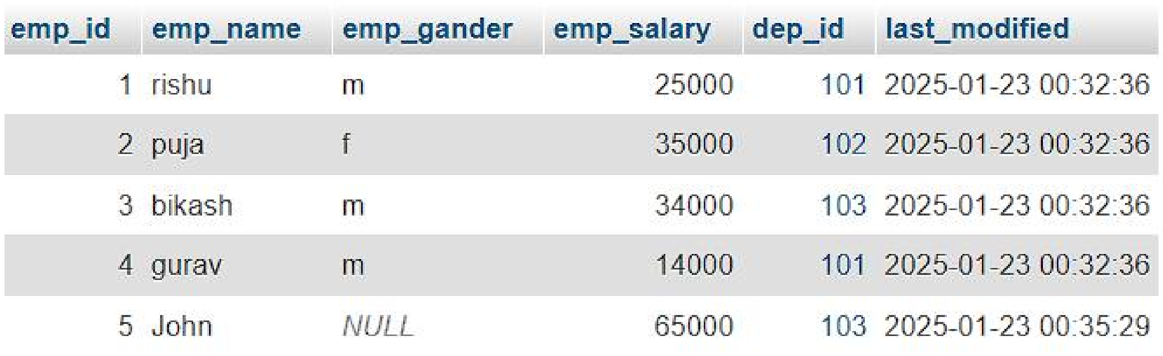
Command:

```
CREATE TABLE employee_changes_log ( log_id INT AUTO_INCREMENT PRIMARY KEY,
employee_id INT NOT NULL, change_type VARCHAR(50) NOT NULL, change_date TIM
ESTAMP DEFAULT CURRENT_TIMESTAMP, 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);
```



log_id	employee_id	change_type	change_date	details
1	5	INSERT	2025-01-23 00:25:18	New employee added: John , Department ID: 103, Sal...

```
ALTER TABLE employees ADD COLUMN last_modified TIMESTAMP DEFAULT CURRENT_TI
MESTAMP ON UPDATE CURRENT_TIMESTAMP;

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;
```



emp_id	emp_name	emp_gander	emp_salary	dep_id	last_modified
1	rishu	m	25000	101	2025-01-23 00:32:36
2	puja	f	35000	102	2025-01-23 00:32:36
3	bikash	m	34000	103	2025-01-23 00:32:36
4	gurav	m	14000	101	2025-01-23 00:32:36
5	John	*NULL*	65000	103	2025-01-23 00:35:29