**Lab Exercises on TABLE Datatype in MySQL**

**Exercise 1: Creating a Simple Table**

Create an in-memory table variable to store employee details with columns: id (INT), name (VARCHAR), and salary (DECIMAL).

**Solution:**

DECLARE emp\_table TABLE (

id INT PRIMARY KEY,

name VARCHAR(50),

salary DECIMAL(10,2)

);

**Exercise 2: Inserting Data into TABLE Type**

Insert 3 employees (1, 'John', 50000), (2, 'Alice', 60000), (3, 'Bob', 70000) into the table variable.

**Solution:**

INSERT INTO emp\_table VALUES (1, 'John', 50000), (2, 'Alice', 60000), (3, 'Bob', 70000);

**Exercise 3: Selecting Data from TABLE Type**

Retrieve all records from the table variable.

**Solution:**

SELECT \* FROM emp\_table;

**Exercise 4: Using TABLE Type in a Stored Procedure**

Create a stored procedure that declares a TABLE type, inserts values, and returns them.

**Solution:**

DELIMITER //

CREATE PROCEDURE GetEmployees()

BEGIN

DECLARE temp\_table TABLE (id INT, name VARCHAR(50), salary DECIMAL(10,2));

INSERT INTO temp\_table VALUES (1, 'John', 50000), (2, 'Alice', 60000);

SELECT \* FROM temp\_table;

END //

DELIMITER ;

**Exercise 5: Using TABLE Type with Cursors**

Declare a cursor for iterating over a table variable.

**Solution:**

DECLARE cur CURSOR FOR SELECT \* FROM emp\_table;

OPEN cur;

FETCH cur INTO @id, @name, @salary;

CLOSE cur;

**Exercise 6: Using TABLE Type in a Function**

Write a function that returns a table variable.

**Solution:**

DELIMITER //

CREATE FUNCTION GetEmployees() RETURNS TABLE (id INT, name VARCHAR(50), salary DECIMAL(10,2))

BEGIN

RETURN TABLE (SELECT \* FROM emp\_table);

END //

DELIMITER ;

**Exercise 7: Using TABLE Type in a JOIN**

Join a table variable with another table.

**Solution:**

SELECT e.id, e.name, e.salary, d.department

FROM emp\_table e

JOIN departments d ON e.id = d.emp\_id;

**Exercise 8: Updating Records in TABLE Type**

Update salary of Alice to 65000.

**Solution:**

UPDATE emp\_table SET salary = 65000 WHERE name = 'Alice';

**Exercise 9: Deleting a Record from TABLE Type**

Delete Bob from the table variable.

**Solution:**

DELETE FROM emp\_table WHERE name = 'Bob';

**Exercise 10: Counting Records in a TABLE Type**

Count the number of employees in the table variable.

**Solution:**

SELECT COUNT(\*) FROM emp\_table;

**Exercise 11: Using TABLE Type with IF Condition**

Check if there are any employees with a salary greater than 60000.

**Solution:**

IF EXISTS (SELECT \* FROM emp\_table WHERE salary > 60000)

THEN

SELECT 'Yes' AS Result;

ELSE

SELECT 'No' AS Result;

END IF;

**Exercise 12: Using a TABLE Type in a LOOP**

Iterate over a table variable using a loop.

**Solution:**

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR SELECT \* FROM emp\_table;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO @id, @name, @salary;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT @id, @name, @salary;

END LOOP;

CLOSE cur;

**Exercise 13: Adding a Column to TABLE Type**

Add an email column to the table variable.

**Solution:**

ALTER TABLE emp\_table ADD COLUMN email VARCHAR(100);

**Exercise 14: Sorting Data in TABLE Type**

Sort employees by salary in descending order.

**Solution:**

SELECT \* FROM emp\_table ORDER BY salary DESC;

**Exercise 15: Using TABLE Type in a CASE Statement**

Apply a condition to set employee categories based on salary.

**Solution:**

SELECT id, name, salary,

CASE

WHEN salary > 60000 THEN 'High'

WHEN salary BETWEEN 40000 AND 60000 THEN 'Medium'

ELSE 'Low'

END AS salary\_category

FROM emp\_table;

**Exercise 16: Creating a TEMPORARY Table with TABLE Type**

Use a temporary table to store employee data.

**Solution:**

CREATE TEMPORARY TABLE temp\_emp\_table AS SELECT \* FROM emp\_table;

SELECT \* FROM temp\_emp\_table;

**Exercise 17: Using a TABLE Type with Aggregation**

Find the average salary of employees.

**Solution:**

SELECT AVG(salary) FROM emp\_table;

**Exercise 18: Filtering Data in TABLE Type**

Retrieve employees with a salary greater than 55000.

**Solution:**

SELECT \* FROM emp\_table WHERE salary > 55000;

**Exercise 19: Using TABLE Type in a UNION**

Combine data from a table variable with another table.

**Solution:**

SELECT id, name, salary FROM emp\_table

UNION

SELECT id, name, salary FROM employees;

**Exercise 20: Using TABLE Type with a Subquery**

Find employees with salaries above the average salary.

**Solution:**

SELECT \* FROM emp\_table WHERE salary > (SELECT AVG(salary) FROM emp\_table);