Problem Statements 1 to 22 are compulsory. Also 23,26,28,29,30 are also compulsory.

23. Write a PL/SQL stored procedure that inserts a new employee into the Employees table. The procedure should accept the EmployeeID, Name, and Salary as input parameters. After executing the procedure, retrieve the details of the newly inserted employee.

Solution:

✅ **Step 1: Assume a table structure**

```
CREATE TABLE Employees (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR(100),
   Salary DECIMAL(10,2)
);
```

✅ **Step 2: Create the stored procedure**

```
DELIMITER //
CREATE PROCEDURE InsertAndGetEmployee(
   IN p_EmployeeID INT,
   IN p_Name VARCHAR(100),
   IN p_Salary DECIMAL(10,2)
)
BEGIN
   -- Insert the new employee
   INSERT INTO Employees (EmployeeID, Name, Salary)
   VALUES (p_EmployeeID, p_Name, p_Salary);

   -- Return the inserted employee details
   SELECT * FROM Employees WHERE EmployeeID = p_EmployeeID;
END //
DELIMITER ;
```

✅ **Step 3: Call the procedure**

```
CALL InsertAndGetEmployee(101, 'Alice Johnson', 55000.00);
```

---

28. Write a row-level trigger that logs any update to the Salary field in the Employees table. Whenever an employee's salary is updated, the trigger should insert a record into a SalaryHistory table to keep track of the previous salary and the new salary.

Solution:

✅ **Step 1: Assume the existing Employees table**

```
CREATE TABLE Employees (
  EmployeeID INT PRIMARY KEY,
  Name VARCHAR(100),
  Salary DECIMAL(10,2)
);
```

✅ **Step 2: Create the SalaryHistory table**

```
CREATE TABLE SalaryHistory (
  HistoryID INT AUTO_INCREMENT PRIMARY KEY,
  EmployeeID INT,
  OldSalary DECIMAL(10,2),
  NewSalary DECIMAL(10,2),
  ChangeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

✅ **Step 3: Create the trigger**

This trigger activates **after any update** to the Salary field **only if the salary actually changes**.

```
DELIMITER //
CREATE TRIGGER trg_SalaryUpdate
AFTER UPDATE ON Employees
FOR EACH ROW
BEGIN
  IF OLD.Salary != NEW.Salary THEN
    INSERT INTO SalaryHistory (EmployeeID, OldSalary, NewSalary)
    VALUES (OLD.EmployeeID, OLD.Salary, NEW.Salary);
  END IF;
END //
DELIMITER ;
```

✅ **Usage Example**

```
-- Update salary
```

```
UPDATE Employees
SET Salary = 60000.00
WHERE EmployeeID = 101;

-- Check log
SELECT * FROM SalaryHistory WHERE EmployeeID = 101;
```

---

26. Write a statement-level trigger that logs the total number of employees in the Employees table to a LogTable whenever a new employee is added.

Solution:

✅ **Step 1: Create Employees table**

```
CREATE TABLE Employees (
  EmployeeID INT PRIMARY KEY,
  Name VARCHAR(100),
  Salary DECIMAL(10,2)
);
```

✅ **Step 2: Create LogTable**

```
CREATE TABLE LogTable (
  LogID INT AUTO_INCREMENT PRIMARY KEY,
  LogTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  TotalEmployees INT
);
```

✅ **Step 3: Create the trigger (row-level, simulating statement-level)**

```
DELIMITER //
CREATE TRIGGER trg_LogEmployeeCount
AFTER INSERT ON Employees
FOR EACH ROW
BEGIN
  INSERT INTO LogTable (TotalEmployees)
  SELECT COUNT(*) FROM Employees;
END //
DELIMITER ;
```

---

29. Write a PL/SQL block that uses an implicit cursor to fetch a single employee's details (e.g., EmployeeID, Name, and Salary) from the Employees table, where the EmployeeID is passed as a parameter. The block should display the details using DBMS_OUTPUT.

Solution:

```
DELIMITER //
CREATE PROCEDURE GetEmployeeDetails(IN p_EmployeeID INT)
BEGIN
  DECLARE v_Name VARCHAR(100);
  DECLARE v_Salary DECIMAL(10,2);

  SELECT Name, Salary
  INTO v_Name, v_Salary
  FROM Employees
  WHERE EmployeeID = p_EmployeeID;

  SELECT
    p_EmployeeID AS EmployeeID,
    v_Name AS Name,
    v_Salary AS Salary;
END //
DELIMITER ;
```

Call the procedure with:

```
CALL GetEmployeeDetails(101);
```

---

30. Write a PL/SQL block that uses an explicit cursor to fetch all employees' details (e.g., EmployeeID, Name, and Salary) from the Employees table and displays them using DBMS_OUTPUT.

Solution:

```
DELIMITER //
CREATE PROCEDURE ShowEmployees()
BEGIN
```

```
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_EmployeeID INT;
    DECLARE v_Name VARCHAR(100);
    DECLARE v_Salary DECIMAL(10,2);

    DECLARE emp_cursor CURSOR FOR
        SELECT EmployeeID, Name, Salary FROM Employees;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN emp_cursor;

    read_loop: LOOP
        FETCH emp_cursor INTO v_EmployeeID, v_Name, v_Salary;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- In MySQL, SELECT outputs to client
        SELECT CONCAT('ID: ', v_EmployeeID, ', Name: ', v_Name, ', Salary: ', v_Salary) AS
EmployeeDetails;
    END LOOP;

    CLOSE emp_cursor;
END //
DELIMITER ;
```

You can run it with:

```
CALL ShowEmployees();
```