Assignment - 5

DBMS

Name: Kamithkar Vinod
Course: PG DAC AUGUST 2025
PRN: 250850320040
Form No: 250500480
Date: 28-10-2025

Problem 1: LOOP

Task: Increase salary of first 3 employees by 10

Code:

```
drop procedure if exists loop_increase_salary;

delimiter //
create procedure loop_increase_salary()
begin
    declare i int default 1;
    declare total int;

    select count(*) into total from employees;

    loop_label: loop
        if i > 3 then
            leave loop_label;
        end if;

        update employees set salary = salary * 1.10 where id = i;
        set i = i + 1;
    end loop loop_label;
end //
delimiter ;

call loop_increase_salary();

select * from employees;
```

Output: —

```
mysql> select * from employees;
+----+---------+------------+----------+
| id | name    | department | salary   |
+----+---------+------------+----------+
|  1 | Anita   | HR         | 27500.00 |
|  2 | Bhavesh | IT         | 35200.00 |
|  3 | Chitra  | Finance    | 30800.00 |
|  4 | Deepak  | IT         | 40000.00 |
|  5 | Esha    | HR         | 35000.00 |
|  6 | Farhan  | Finance    | 30000.00 |
+----+---------+------------+----------+
```

Problem 2: LOOP

Task: Display all employee names using LOOP.

Code: —

```
DROP PROCEDURE IF EXISTS loop_display_names;
DELIMITER //
CREATE PROCEDURE loop_display_names()
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE total INT;
   DECLARE emp_name VARCHAR(50);
   SELECT COUNT(*) INTO total FROM employees;

   display_loop: LOOP
       IF i > total THEN
           LEAVE display_loop;
       END IF;
       SELECT name INTO emp_name FROM employees WHERE id = i;
       SELECT emp_name AS Employee_Name;
       SET i = i + 1;
   END LOOP display_loop;
END //
DELIMITER ;
CALL loop_display_names();
```

Output: —

```
mysql> CALL loop_display_names();
+---------------+
| Employee_Name |
+---------------+
| Anita         |
+---------------+
1 row in set (0.01 sec)

+---------------+
| Employee_Name |
+---------------+
| Bhavesh       |
+---------------+
1 row in set (0.02 sec)

+---------------+
| Employee_Name |
+---------------+
| Chitra        |
+---------------+
```

Problem 3: LOOP

Task: Calculate total salary of all employees using LOOP.

Code:

```
DROP PROCEDURE IF EXISTS loop_total_salary;
DELIMITER //
CREATE PROCEDURE loop_total_salary()
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE total INT;
   DECLARE total_salary DECIMAL(10,2) DEFAULT 0;
   DECLARE sal DECIMAL(10,2);
   SELECT COUNT(*) INTO total FROM employees;

   salary_loop: LOOP
       IF i > total THEN
           LEAVE salary_loop;
       END IF;
       SELECT salary INTO sal FROM employees WHERE id = i;
       SET total_salary = total_salary + sal;
       SET i = i + 1;
   END LOOP salary_loop;
   SELECT total_salary AS Total_Salary;
END //
DELIMITER ;
CALL loop_total_salary();
```

Output: —

```
mysql> CALL loop_total_salary();
+--------------+
| Total_Salary |
+--------------+
|    198500.00 |
+--------------+
```

Problem 4: LOOP

Task: Insert 3 new temporary employees into the table using LOOP.

Code: —

```
DROP PROCEDURE IF EXISTS loop_insert_temps;
DELIMITER //
CREATE PROCEDURE loop_insert_temps()
BEGIN
   DECLARE i INT DEFAULT 1;
   temp_loop: LOOP
       IF i > 3 THEN
           LEAVE temp_loop;
       END IF;
        INSERT INTO employees(name, department, salary)
       VALUES (CONCAT('TempEmp', i), 'Temp', 20000);
       SET i = i + 1;
   END LOOP temp_loop;
END //
DELIMITER ;
CALL loop_insert_temps();
SELECT * FROM employees;
```

Output: —

```
mysql> SELECT * FROM employees;
+----+----------+------------+----------+
| id | name     | department | salary   |
+----+----------+------------+----------+
|  1 | Anita    | HR         | 27500.00 |
|  2 | Bhavesh  | IT         | 35200.00 |
|  3 | Chitra   | Finance    | 30800.00 |
|  4 | Deepak   | IT         | 40000.00 |
|  5 | Esha     | HR         | 35000.00 |
|  6 | Farhan   | Finance    | 30000.00 |
|  7 | TempEmp1 | Temp       | 20000.00 |
|  8 | TempEmp2 | Temp       | 20000.00 |
|  9 | TempEmp3 | Temp       | 20000.00 |
+----+----------+------------+----------+
```

Problem 5: LOOP

Task: Display employee names department-wise using LOOP. Loop through all departments and show employee names under each.

Code: —

```
DROP PROCEDURE IF EXISTS loop_names_by_department;
DELIMITER //
CREATE PROCEDURE loop_names_by_department()
BEGIN
  DECLARE i INT DEFAULT 1;
  DECLARE total INT;
  DECLARE dept VARCHAR(30);
  DECLARE emp_name VARCHAR(50);
  SELECT COUNT(*) INTO total FROM employees;

  loop_label: LOOP
      IF i > total THEN
          LEAVE loop_label;
      END IF;
      SELECT department, name INTO dept, emp_name FROM employees
         WHERE id = i;
      SELECT CONCAT('Department: ', dept, ' - ', emp_name) AS
         Employee_Info;
      SET i = i + 1;
  END LOOP loop_label;
END //
DELIMITER ;
CALL loop_names_by_department();
```

Output: —

```
mysql> CALL loop_names_by_department();
+------------------------+
| Employee_Info          |
+------------------------+
| Department: HR - Anita |
+------------------------+
1 row in set (0.01 sec)

+--------------------------+
| Employee_Info            |
+--------------------------+
| Department: IT - Bhavesh |
+--------------------------+
1 row in set (0.01 sec)

+-----------------------------+
| Employee_Info               |
+-----------------------------+
| Department: Finance - Chitra |
+-----------------------------+
```

Problem 6: LOOP

Task: Grant a 2,000 performance bonus to each IT department employee using LOOP. Traverse through each employee and update if the department is 'IT'.

Code:

```sql
DROP PROCEDURE IF EXISTS loop_bonus_it;
DELIMITER //
CREATE PROCEDURE loop_bonus_it()
BEGIN
  DECLARE i INT DEFAULT 1;
  DECLARE total INT;
  DECLARE dept VARCHAR(30);
  SELECT COUNT(*) INTO total FROM employees;

  bonus_loop: LOOP
      IF i > total THEN
          LEAVE bonus_loop;
      END IF;
      SELECT department INTO dept FROM employees WHERE id = i;
      IF dept = 'IT' THEN
          UPDATE employees SET salary = salary + 2000 WHERE id =
             i;
      END IF;
      SET i = i + 1;
  END LOOP bonus_loop;
END //
```

```
DELIMITER ;
CALL loop_bonus_it();
SELECT * FROM employees;
```

Output: —

```
mysql> SELECT * FROM employees;
+----+----------+------------+----------+
| id | name     | department | salary   |
+----+----------+------------+----------+
|  1 | Anita    | HR         | 27500.00 |
|  2 | Bhavesh  | IT         | 37200.00 |
|  3 | Chitra   | Finance    | 30800.00 |
|  4 | Deepak   | IT         | 42000.00 |
|  5 | Esha     | HR         | 35000.00 |
|  6 | Farhan   | Finance    | 30000.00 |
|  7 | TempEmp1 | Temp       | 20000.00 |
|  8 | TempEmp2 | Temp       | 20000.00 |
|  9 | TempEmp3 | Temp       | 20000.00 |
+----+----------+------------+----------+
```

Problem 7: LOOP

Task: Find and display the highest salary among all employees using LOOP.

Code: —

```
DROP PROCEDURE IF EXISTS loop_highest_salary;
DELIMITER //
CREATE PROCEDURE loop_highest_salary()
BEGIN
  DECLARE i INT DEFAULT 1;
  DECLARE total INT;
  DECLARE sal DECIMAL(10,2);
  DECLARE max_sal DECIMAL(10,2) DEFAULT 0;
  SELECT COUNT(*) INTO total FROM employees;

  find_loop: LOOP
      IF i > total THEN
          LEAVE find_loop;
        END IF;
                  SELECT salary INTO sal FROM employees WHERE id
                     = i;
                  IF sal > max_sal THEN
                          SET max_sal = sal;
                  END IF;
                  SET i = i + 1;
          END LOOP find_loop;
          SELECT max_sal AS Highest_Salary;
END //
```

```
DELIMITER ;
CALL loop_highest_salary();
```

Output: —

```
mysql> CALL loop_highest_salary();
+----------------+
| Highest_Salary |
+----------------+
|       42000.00 |
+----------------+
```

Problem 8: LOOP

Task: Copy data of first 2 employees into a new table employee$_{b}ackupusingLOOP$.

Code: —

```
DROP PROCEDURE IF EXISTS loop_copy_to_backup;
DELIMITER //
CREATE PROCEDURE loop_copy_to_backup()
BEGIN
  DECLARE i INT DEFAULT 1;
  CREATE TABLE IF NOT EXISTS employee_backup LIKE employees;

  copy_loop: LOOP
      IF i > 2 THEN
          LEAVE copy_loop;
      END IF;
      INSERT INTO employee_backup(name, department, salary)
      SELECT name, department, salary FROM employees WHERE id = i
         ;
      SET i = i + 1;
  END LOOP copy_loop;
END //
DELIMITER ;
CALL loop_copy_to_backup();
SELECT * FROM employee_backup;
```

Output: —

```
mysql> SELECT * FROM employee_backup;
+----+---------+------------+----------+
| id | name    | department | salary   |
+----+---------+------------+----------+
|  1 | Anita   | HR         | 27500.00 |
|  2 | Bhavesh | IT         | 37200.00 |
+----+---------+------------+----------+
```

Problem 9: WHILE

Task: Display employees having salary greater than 30,000.

Code: —

```
DROP PROCEDURE IF EXISTS while_high_salary;
DELIMITER //
CREATE PROCEDURE while_high_salary()
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE total INT;
   DECLARE sal DECIMAL(10,2);
   DECLARE emp_name VARCHAR(50);
   SELECT COUNT(*) INTO total FROM employees;

   WHILE i <= total DO
       SELECT salary, name INTO sal, emp_name FROM employees
          WHERE id = i LIMIT 1;
       IF sal > 30000 THEN
           SELECT emp_name AS Name, sal AS Salary;
       END IF;
       SET i = i + 1;
   END WHILE;
END //
DELIMITER ;
CALL while_high_salary();
```

Output: —

```
mysql> CALL while_high_salary();
+---------+----------+
| Name    | Salary   |
+---------+----------+
| Bhavesh | 37200.00 |
+---------+----------+
1 row in set (0.01 sec)

+--------+----------+
| Name   | Salary   |
+--------+----------+
| Chitra | 30800.00 |
+--------+----------+
1 row in set (0.01 sec)
```

Problem 10: WHILE

Task: Increase one employee's salary incrementally by 5000 until it reaches 50,000.

Code: —

```
DROP PROCEDURE IF EXISTS while_increase_until;
DELIMITER //
CREATE PROCEDURE while_increase_until()
BEGIN
        DECLARE sal DECIMAL(10,2);
        DECLARE emp_name VARCHAR(50);
        SELECT salary, name INTO sal, emp_name FROM employees
           WHERE id = 1;
        WHILE sal < 50000 DO
           SET sal = sal + 5000;
           SELECT CONCAT(emp_name, ' salary increased to ', sal)
              AS Progress;
        END WHILE;
END //
DELIMITER ;
CALL while_increase_until();
```

Output: —

```
mysql> CALL while_increase_until();
+-----------------------------------+
| Progress                          |
+-----------------------------------+
| Anita salary increased to 32500.00 |
+-----------------------------------+
1 row in set (0.00 sec)

+-----------------------------------+
| Progress                          |
+-----------------------------------+
| Anita salary increased to 37500.00 |
+-----------------------------------+
```

Problem 11: WHILE

Task: Count total IT department employees.

Code: —

```
DROP PROCEDURE IF EXISTS while_count_it;
DELIMITER //
CREATE PROCEDURE while_count_it()
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE total INT;
   DECLARE dept VARCHAR(30);
   DECLARE count_it INT DEFAULT 0;
   SELECT COUNT(*) INTO total FROM employees;
```

```
    WHILE i <= total DO
        SELECT department INTO dept FROM employees WHERE id = i;
        IF dept = 'IT' THEN
            SET count_it = count_it + 1;
        END IF;
        SET i = i + 1;
    END WHILE;
    SELECT count_it AS Total_IT_Employees;
END //
DELIMITER ;
CALL while_count_it();
```

Output: —

```
mysql> CALL while_count_it();
+--------------------+
| Total_IT_Employees |
+--------------------+
|                  2 |
+--------------------+
```

Problem 12: WHILE

Task: Display employees whose name length is less than 6 characters using WHILE.

Code: —

```
DROP PROCEDURE IF EXISTS while_short_names;
DELIMITER //
CREATE PROCEDURE while_short_names()
BEGIN
  DECLARE i INT DEFAULT 1;
  DECLARE total INT;
  DECLARE emp_name VARCHAR(50);
  SELECT COUNT(*) INTO total FROM employees;

  WHILE i <= total DO
      SELECT name INTO emp_name FROM employees WHERE id = i;
      IF CHAR_LENGTH(emp_name) < 6 THEN
          SELECT emp_name AS Short_Name;
      END IF;
      SET i = i + 1;
  END WHILE;
END //
DELIMITER ;
CALL while_short_names();
```

Output: —

```
mysql> CALL while_short_names();
+------------+
| Short_Name |
+------------+
| Anita      |
+------------+
1 row in set (0.01 sec)

+------------+
| Short_Name |
+------------+
| Esha       |
+------------+
```

Problem 13: WHILE

Task: Deduct 1,000 from one employee's salary until it reaches 25,000 using WHILE.

Code: —

```
DROP PROCEDURE IF EXISTS while_deduct_salary;
DELIMITER //
CREATE PROCEDURE while_deduct_salary()
BEGIN
  DECLARE sal DECIMAL(10,2);
  DECLARE emp_name VARCHAR(50);
  SELECT salary, name INTO sal, emp_name FROM employees WHERE id
     = 3;

  WHILE sal > 25000 DO
      SET sal = sal - 1000;
      SELECT CONCAT(emp_name, ' salary reduced to ', sal) AS
         Progress;
  END WHILE;
END //
DELIMITER ;
CALL while_deduct_salary();
```

Output: —

```
mysql> CALL while_deduct_salary();
+-----------------------------------+
| Progress                          |
+-----------------------------------+
| Chitra salary reduced to 29800.00 |
+-----------------------------------+
1 row in set (0.00 sec)

+-----------------------------------+
| Progress                          |
+-----------------------------------+
| Chitra salary reduced to 28800.00 |
+-----------------------------------+
```

Problem 14: WHILE

Task: Calculate average salary of all employees using WHILE.

Code: —

```
DROP PROCEDURE IF EXISTS while_average_salary;
DELIMITER //
CREATE PROCEDURE while_average_salary()
BEGIN
  DECLARE i INT DEFAULT 1;
  DECLARE total INT;
  DECLARE total_sal DECIMAL(10,2) DEFAULT 0;
  DECLARE sal DECIMAL(10,2);
  SELECT COUNT(*) INTO total FROM employees;

  WHILE i <= total DO
      SELECT salary INTO sal FROM employees WHERE id = i;
      SET total_sal = total_sal + sal;
      SET i = i + 1;
  END WHILE;
  SELECT (total_sal / total) AS Average_Salary;
END //
DELIMITER ;
CALL while_average_salary();
```

Output: —

```
mysql> CALL while_average_salary();
+----------------+
| Average_Salary |
+----------------+
|   29166.666667 |
+----------------+
```

Problem 15: REPEAT

Task: Add 5,000 to one employee's salary until it exceeds 40,000.

Code: —

```
DROP PROCEDURE IF EXISTS repeat_add_salary;
DELIMITER //
CREATE PROCEDURE repeat_add_salary()
BEGIN
   DECLARE sal DECIMAL(10,2);
   DECLARE emp_name VARCHAR(50);
   SELECT salary, name INTO sal, emp_name FROM employees WHERE id
       = 2;
   REPEAT
       SET sal = sal + 5000;
       SELECT CONCAT('New salary of ', emp_name, ' is ', sal) AS
          Info;
   UNTIL sal > 40000
   END REPEAT;
END //
DELIMITER ;
CALL repeat_add_salary();
```

Output: —

```
mysql> CALL repeat_add_salary();
+-----------------------------------+
| Info                              |
+-----------------------------------+
| New salary of Bhavesh is 42200.00 |
+-----------------------------------+
```

Problem 16: REPEAT

Task: Insert employees until 10 total records exist.

Code: —

```
DROP PROCEDURE IF EXISTS repeat_insert_until10;
DELIMITER //
CREATE PROCEDURE repeat_insert_until10()
BEGIN
   DECLARE total INT;
   SELECT COUNT(*) INTO total FROM employees;
   REPEAT
       INSERT INTO employees(name, department, salary)
       VALUES (CONCAT('ExtraEmp', total + 1), 'Support', 25000);
       SELECT COUNT(*) INTO total FROM employees;
   UNTIL total >= 10
   END REPEAT;
```

```
END //
DELIMITER ;
CALL repeat_insert_until10();
SELECT * FROM employees;
```

Output: —

```
mysql> SELECT * FROM employees;
+----+------------+------------+----------+
| id | name       | department | salary   |
+----+------------+------------+----------+
|  1 | Anita      | HR         | 27500.00 |
|  2 | Bhavesh    | IT         | 37200.00 |
|  3 | Chitra     | Finance    | 30800.00 |
|  4 | Deepak     | IT         | 42000.00 |
|  5 | Esha       | HR         | 35000.00 |
|  6 | Farhan     | Finance    | 30000.00 |
|  7 | TempEmp1   | Temp       | 20000.00 |
|  8 | TempEmp2   | Temp       | 20000.00 |
|  9 | TempEmp3   | Temp       | 20000.00 |
| 10 | ExtraEmp10 | Support    | 25000.00 |
+----+------------+------------+----------+
```

Problem 17: REPEAT

Task: Keep adding bonuses until total crosses 1,00,000.

Code: —

```
DROP PROCEDURE IF EXISTS repeat_bonus_pool;
DELIMITER //
CREATE PROCEDURE repeat_bonus_pool()
BEGIN
   DECLARE total_bonus DECIMAL(10,2) DEFAULT 0;
   DECLARE bonus DECIMAL(10,2) DEFAULT 15000;
   REPEAT
       SET total_bonus = total_bonus + bonus;
       SELECT total_bonus AS Current_Bonus;
   UNTIL total_bonus > 100000
   END REPEAT;
END //
DELIMITER ;
CALL repeat_bonus_pool();
```

Output: —

```
mysql> CALL repeat_bonus_pool();
+---------------+
| Current_Bonus |
+---------------+
|      15000.00 |
+---------------+
1 row in set (0.00 sec)

+---------------+
| Current_Bonus |
+---------------+
|      30000.00 |
+---------------+
1 row in set (0.01 sec)
```

Problem 18: REPEAT

Task: Keep inserting random temporary employees until 15 total employees exist using REPEAT.

Code: —

```
DROP PROCEDURE IF EXISTS repeat_insert_until15;
DELIMITER //
CREATE PROCEDURE repeat_insert_until15()
BEGIN
  DECLARE total INT;
  SELECT COUNT(*) INTO total FROM employees;
  REPEAT
      INSERT INTO employees(name, department, salary)
      VALUES (CONCAT('TempEmp', total + 1), 'Support', 22000);
      SELECT COUNT(*) INTO total FROM employees;
  UNTIL total >= 15
  END REPEAT;
END //
DELIMITER ;
CALL repeat_insert_until15();
SELECT * FROM employees;
```

Output: —

```
mysql> SELECT * FROM employees;
+----+------------+------------+----------+
| id | name       | department | salary   |
+----+------------+------------+----------+
|  1 | Anita      | HR         | 27500.00 |
|  2 | Bhavesh    | IT         | 37200.00 |
|  3 | Chitra     | Finance    | 30800.00 |
|  4 | Deepak     | IT         | 42000.00 |
|  5 | Esha       | HR         | 35000.00 |
|  6 | Farhan     | Finance    | 30000.00 |
|  7 | TempEmp1   | Temp       | 20000.00 |
|  8 | TempEmp2   | Temp       | 20000.00 |
|  9 | TempEmp3   | Temp       | 20000.00 |
| 10 | ExtraEmp10 | Support    | 25000.00 |
| 11 | TempEmp11  | Support    | 22000.00 |
| 12 | TempEmp12  | Support    | 22000.00 |
| 13 | TempEmp13  | Support    | 22000.00 |
| 14 | TempEmp14  | Support    | 22000.00 |
| 15 | TempEmp15  | Support    | 22000.00 |
+----+------------+------------+----------+
```

Problem 19: REPEAT

Task: Keep adding 500 increments to a specific employee's salary until it reaches 45,000 using REPEAT.

Code: —

```
DROP PROCEDURE IF EXISTS repeat_raise_salary;
DELIMITER //
CREATE PROCEDURE repeat_raise_salary()
BEGIN
  DECLARE sal DECIMAL(10,2);
  DECLARE emp_name VARCHAR(50);
  SELECT salary, name INTO sal, emp_name FROM employees WHERE id
     = 4;
  REPEAT
      SET sal = sal + 500;
      SELECT CONCAT('Salary of ', emp_name, ' is now ', sal) AS
         UpdateInfo;
  UNTIL sal >= 45000
  END REPEAT;
END //
DELIMITER ;
CALL repeat_raise_salary();
```

Output: —

```
+-----------------------------------+
| UpdateInfo                        |
+-----------------------------------+
| Salary of Deepak is now 44500.00 |
+-----------------------------------+
1 row in set (0.02 sec)

+-----------------------------------+
| UpdateInfo                        |
+-----------------------------------+
| Salary of Deepak is now 45000.00 |
+-----------------------------------+
```

Problem 20: REPEAT

Task: Display employee salaries one by one using REPEAT until all records are printed.

Code: —

```
DROP PROCEDURE IF EXISTS repeat_display_salaries;
DELIMITER //
CREATE PROCEDURE repeat_display_salaries()
BEGIN
  DECLARE i INT DEFAULT 1;
  DECLARE total INT;
  DECLARE emp_name VARCHAR(50);
  DECLARE sal DECIMAL(10,2);
  SELECT COUNT(*) INTO total FROM employees;

  REPEAT
      SELECT name, salary INTO emp_name, sal FROM employees WHERE
          id = i;
      SELECT CONCAT(emp_name, ' earns    ', sal) AS
         Employee_Salary;
      SET i = i + 1;
  UNTIL i > total
  END REPEAT;
END //
DELIMITER ;
CALL repeat_display_salaries();
```

Output: —

```
+-------------------------+
| Employee_Salary         |
+-------------------------+
| Chitra earns ?30800.00 |
+-------------------------+
1 row in set (0.01 sec)

+-------------------------+
| Employee_Salary         |
+-------------------------+
| Deepak earns ?42000.00 |
+-------------------------+
```