UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Experiment No. 4

Student Name: Roshan Kumar Singh
UID: 25MCA20067
Branch: MCA(GEN)
Section/Group: 25MCA-1_A
Semester: 2nd
Date of Performance: 03/02/26
Subject Name: Technical Training-1
Subject Code: 25CAP-652

Aim:

To understand and implement iterative control structures in PostgreSQL conceptually, including FOR loops, WHILE loops, and basic LOOP constructs, for repeated execution of database logic.

Objective:

- To understand why iteration is required in database programming.
- To learn the purpose and behaviour of FOR, WHILE, and LOOP constructs.
- To understand how repeated data processing is handled in databases.
- To relate loop concepts to real-world batch processing scenarios.
- To strengthen conceptual knowledge of procedural SQL used in enterprise systems.

Tools Used:

PostgreSQL

Procedure:
Step 1: FOR Loop - Simple Iteration

- The loop runs a fixed number of times
- Each iteration represents one execution cycle
- Useful for understanding basic loop behaviour

Step 2: FOR Loop with Query (Row-by-Row Processing)

- The loop processes database records one at a time
- Each iteration handles a single row
- Simulates cursor-based processing

Step 3: WHILE Loop - Conditional Iteration

- The loop runs until a condition becomes false
- Execution depends entirely on the condition
- The condition is checked before every iteration

Step 4: LOOP with EXIT WHEN

- The loop does not stop automatically
- An explicit exit condition controls termination
- Gives flexibility in complex logic

Step 5: Salary Increment Using FOR Loop

- Employee records are processed one by one
- Salary values are updated iteratively
- Represents real-world payroll processing

Step 6: Combining LOOP with IF Condition

- Loop processes each record
- Conditional logic classifies data during iteration
- Demonstrates decision-making inside loops

Code:

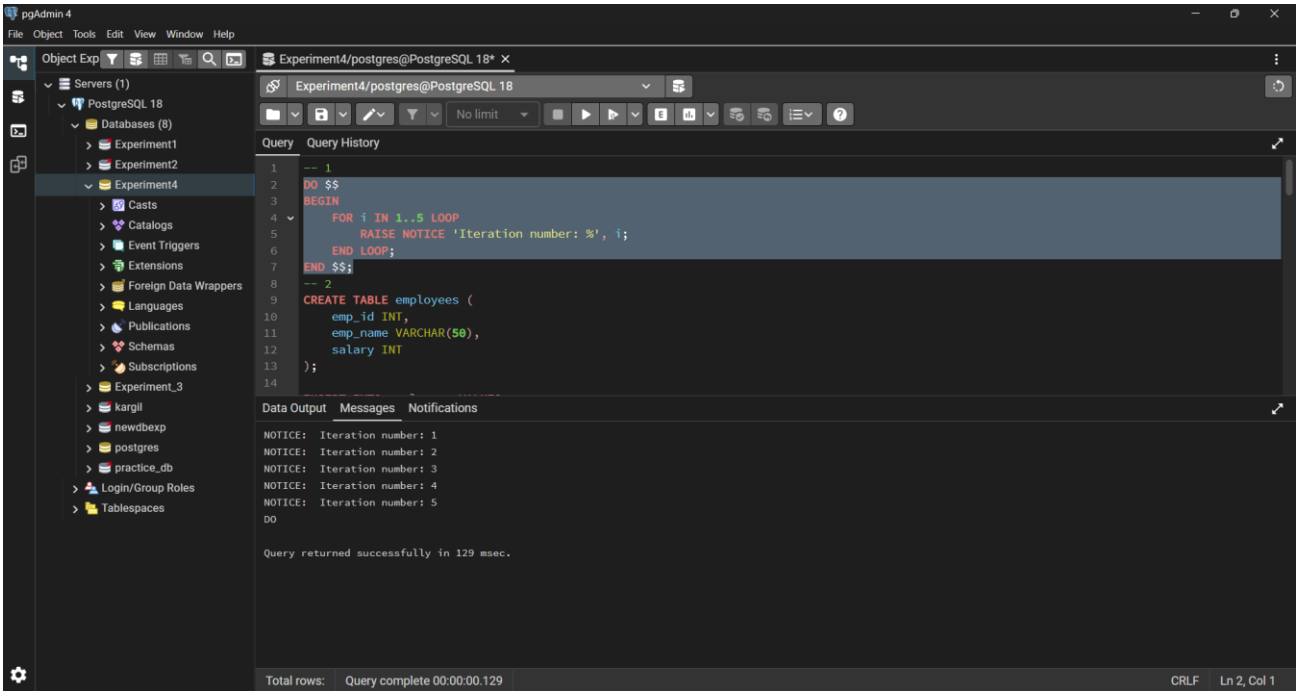
```
--1
DO $$
BEGIN
   FOR i IN 1..5 LOOP
      RAISE NOTICE 'Iteration number: %', i;
   END LOOP;
END $$;
```

```
--2
CREATE TABLE employees (
    emp_id INT,
    emp_name VARCHAR(50),
    salary INT
);

INSERT INTO employees VALUES
(1, 'Amit', 30000),
(2, 'Neha', 45000),
(3, 'Rahul', 28000);

DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT * FROM employees LOOP
        RAISE NOTICE 'ID: %, Name: %, Salary: %',
        rec.emp_id, rec.emp_name, rec.salary;
    END LOOP;
END $$;

--3
DO $$
DECLARE
    counter INT := 1;
BEGIN
    WHILE counter <= 5 LOOP
        RAISE NOTICE 'Counter value: %', counter;
        counter := counter + 1;
    END LOOP;
END $$;

--4
DO $$
DECLARE
    num INT := 1;
BEGIN
    LOOP
        RAISE NOTICE 'Number: %', num;
        num := num + 1;
```

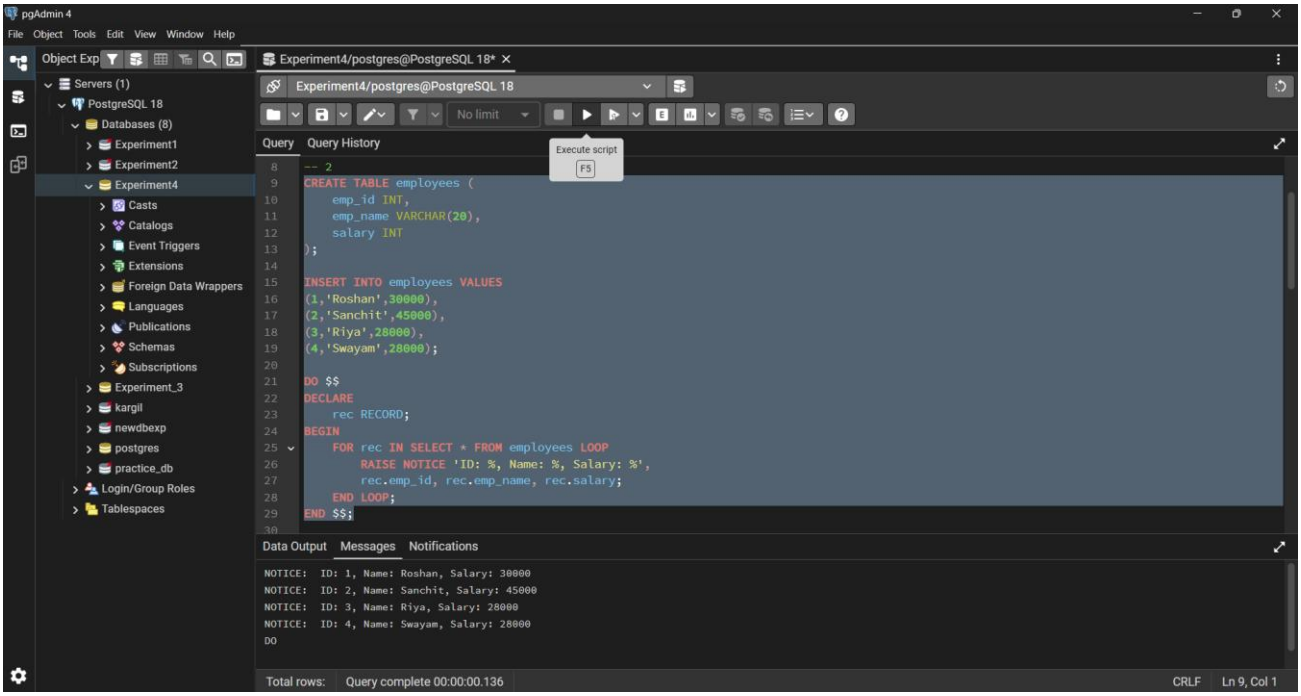
```
        EXIT WHEN num > 5;
    END LOOP;
END $$;

--5
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT * FROM employees LOOP
        UPDATE employees
        SET salary = salary + 5000
        WHERE emp_id = rec.emp_id;
    END LOOP;
END $$;

SELECT * FROM employees;

-- 6
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT * FROM employees LOOP
        IF rec.salary >= 40000 THEN
            RAISE NOTICE '% is High Salary Employee', rec.emp_name;
        ELSE
            RAISE NOTICE '% is Low Salary Employee', rec.emp_name;
        END IF;
    END LOOP;
END $$;
```

Output:

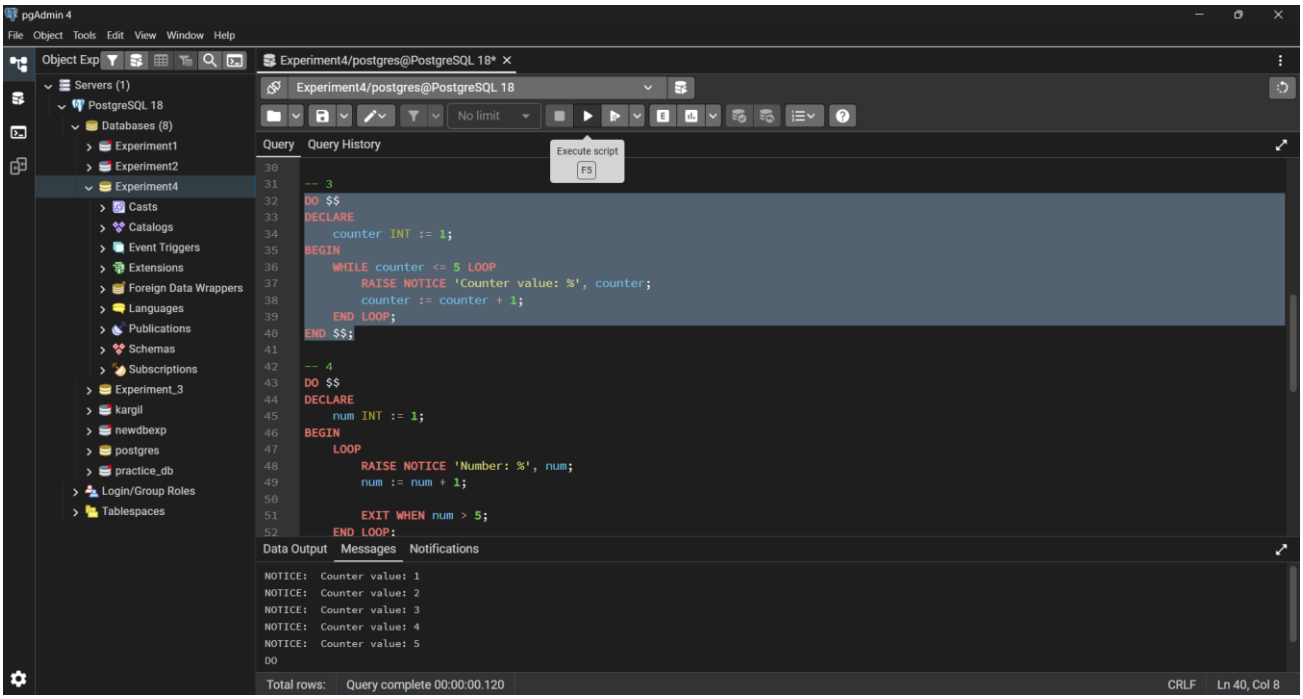
Step1: FOR Loop - Simple Iteration



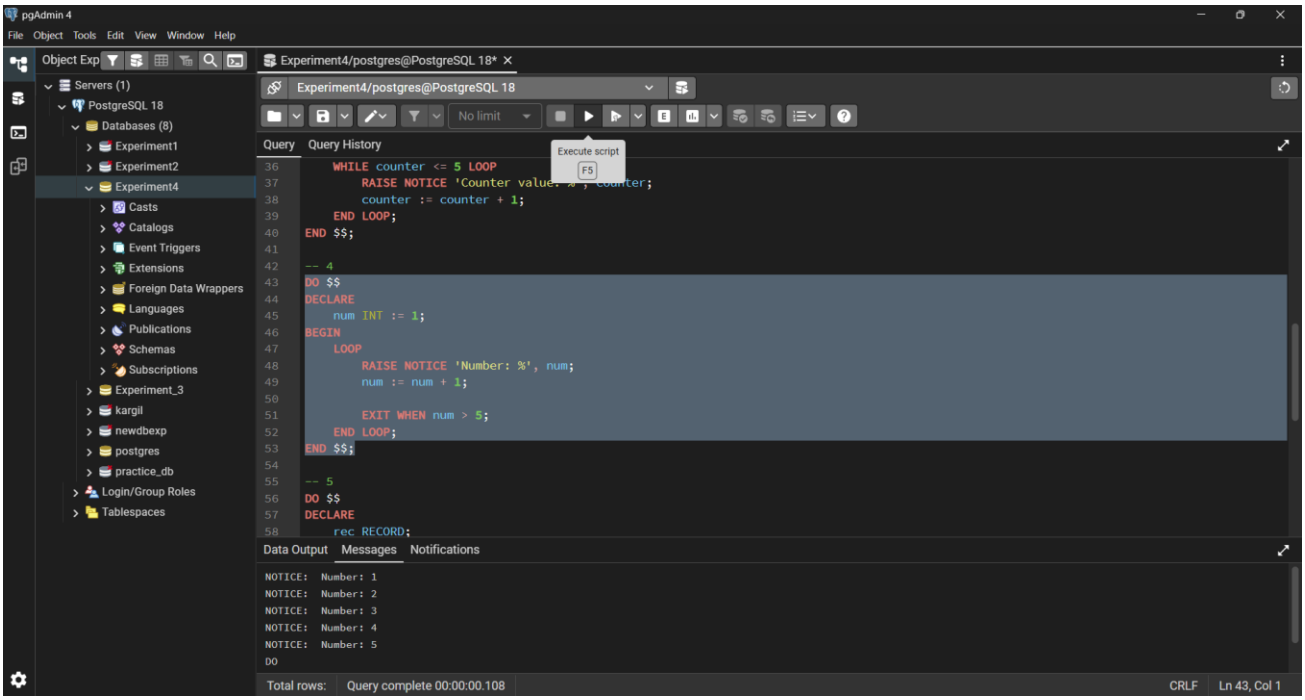
Step2: FOR Loop with Query (Row-by-Row Processing)



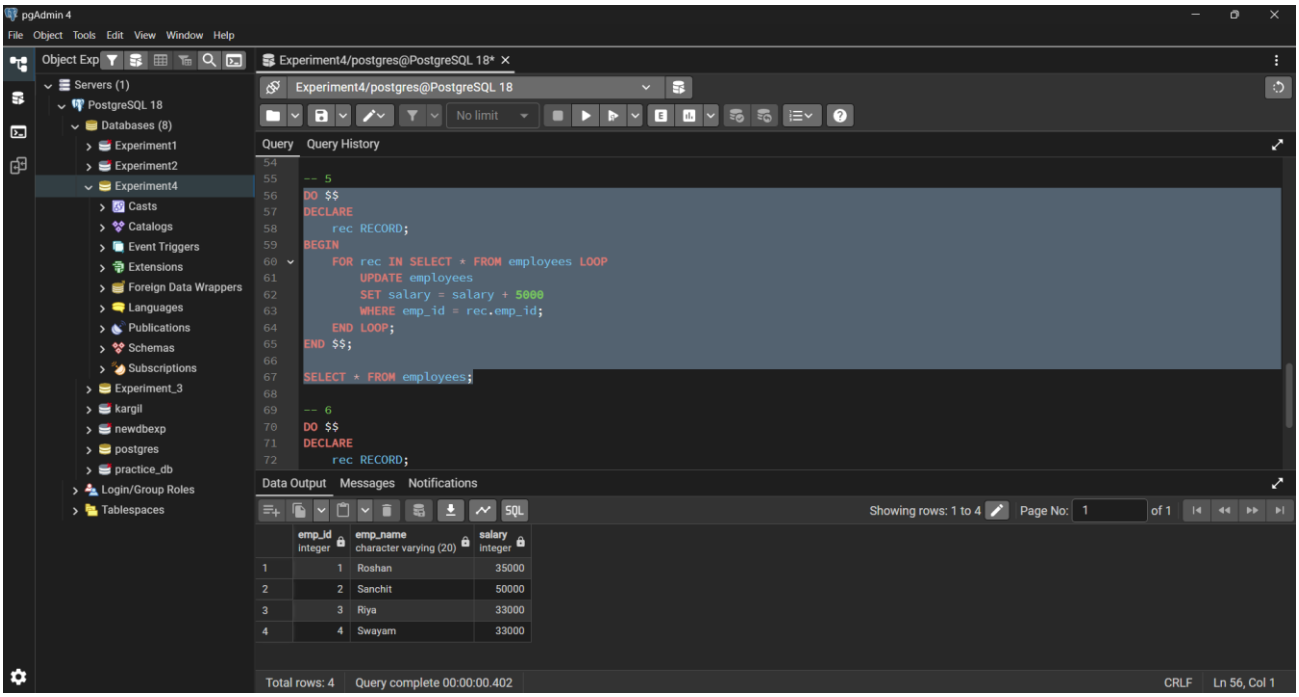
Step3: WHILE Loop - Conditional Iteration



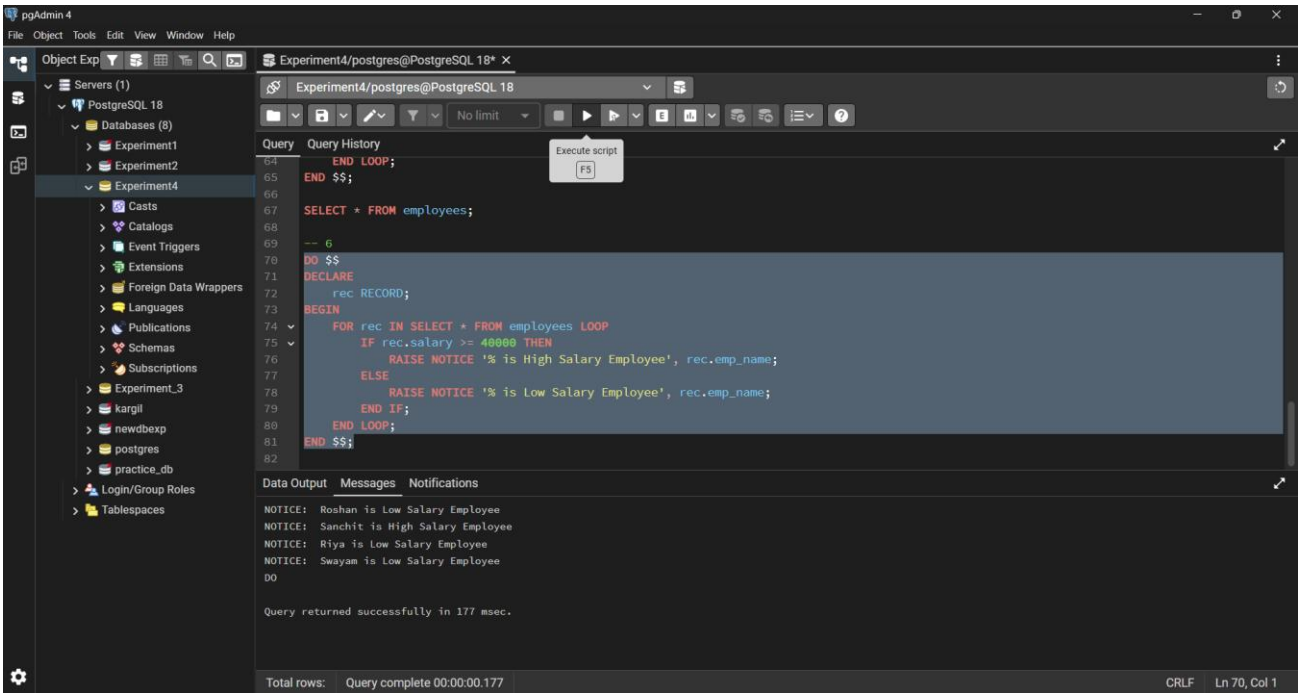
Step4: LOOP with EXIT WHEN



Step5: Salary Increment Using FOR Loop



Step6: Combining LOOP with IF Condition



Learning Outcomes:

- Understood the importance of iteration in database programming for repeated execution of logic.
- Learnt the working and use of FOR, WHILE, and LOOP constructs in PostgreSQL.
- Gained practical knowledge of row-by-row data processing using loops in PL/SQL.
- Understood how iterative updates and conditional logic can be applied inside loops.
- Developed confidence in writing procedural SQL programs for real-world batch processing scenarios.