UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

Experiment- 4

Branch: MCA (AI&ML)	**Semester: 2**
Student Name: Ajay Rakwal	**UID:** 25MCI10329
Subject Name: Technical Training	**Subject Code**: 25CAP-652
Section/Group: MAM-1(A)	**Date of Performance:** 03-02-2026

Aim of the program:

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.

Software Requirements

- **Operating System:** Windows / Linux
- **Database Management System:** MySQL / Oracle / PostgreSQL
- **SQL Interface:** MySQL Workbench /Web Based / pgAdmin

Objective

- To understand why iteration is required in database programming
- To learn the purpose and behavior 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

Procedure of the Practical

I. Open PostgreSQL using pgAdmin or a browser-based PostgreSQL interface and connect to the database.
II. Study the need for iterative control structures in database programming through the given theory and real-world use cases.
III. Write and execute a simple FOR loop to understand fixed-iteration execution.
IV. Implement a FOR loop with query-based iteration to process records one row at a time.
V. Implement a WHILE loop to observe condition-controlled repeated execution.
VI. Use a LOOP construct with an EXIT WHEN condition to understand flexible and custom termination logic.
VII. Apply a FOR loop to simulate real-world batch processing such as salary increment for employee records.
VIII. Combine LOOP and IF conditions to perform decision-making during iteration.
IX. Observe the execution flow and output messages to understand how iterative logic works in PostgreSQL.

Practical / Experiment Steps

```
-- employee table (used in examples)
CREATE TABLE employee (
    emp_id INT,
    salary INT
);
INSERT INTO employee VALUES
(1, 30000),
(2, 35000),
(3, 40000);
```

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

-- Example 2: FOR Loop with Query
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT emp_id FROM employee LOOP
        RAISE NOTICE 'Processing Employee ID: %', rec.emp_id;
    END LOOP;
END $$;

-- Example 3: WHILE Loop
DO $$
DECLARE
    counter INT := 1;
BEGIN
    WHILE counter <= 5 LOOP
```

```
        RAISE NOTICE 'Counter value: %', counter;
        counter := counter + 1;
    END LOOP;
END $$;

-- Example 4: LOOP with EXIT WHEN
DO $$
DECLARE
    num INT := 1;
BEGIN
    LOOP
        RAISE NOTICE 'Number: %', num;
        num := num + 1;
        EXIT WHEN num > 5;
    END LOOP;
END $$;

-- Example 5: Salary Increment Using FOR Loop
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT emp_id FROM employee LOOP
        UPDATE employee
```

```
        SET salary = salary + 1000
        WHERE emp_id = rec.emp_id;
    END LOOP;
END $$;
SELECT * FROM employee


-- Example 6: Combining LOOP with IF Condition
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT emp_id, salary FROM employee LOOP
        IF rec.salary >= 40000 THEN
            RAISE NOTICE 'Employee % : High Salary', rec.emp_id;
        ELSE
            RAISE NOTICE 'Employee % : Normal Salary', rec.emp_id;
        END IF;
    END LOOP;
END $$;
```

I/O Analysis (Input / Output)

Input

- Table creation queries
- Sample data insertion commands

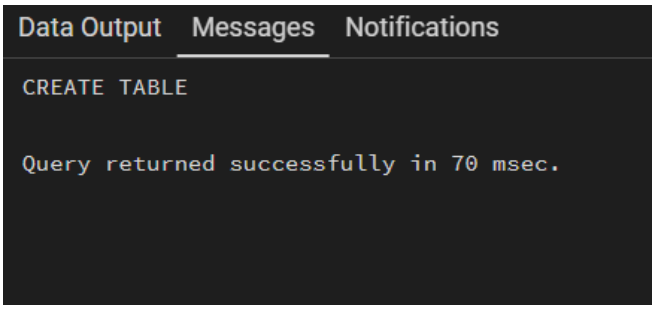
- Loop-based PL/pgSQL blocks (FOR, WHILE, and LOOP)
- IF–ELSE conditional logic inside loops
- UPDATE statements executed within loop constructs
- SELECT queries to display processed results

Output

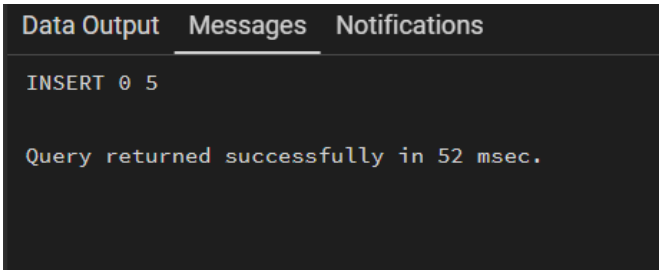
- Tables created successfully
- Records inserted and processed correctly
- Iterative execution performed as per loop logic
- Conditional classification applied correctly using IF–ELSE
- Data updated accurately during iteration
- Correct execution flow and messages displayed for each loop operation

OUTPUT:

Table Created:

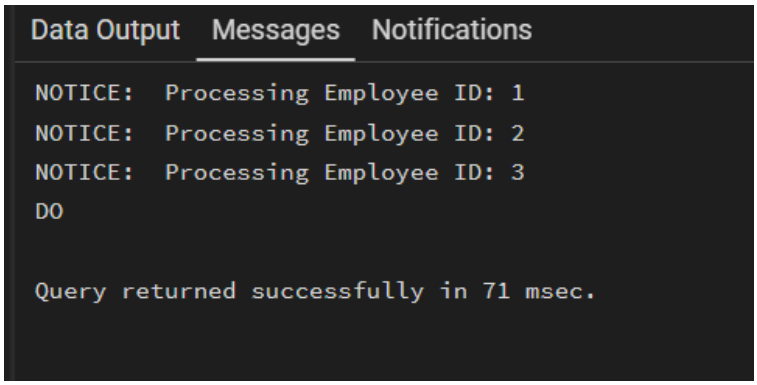


Insert sample records:

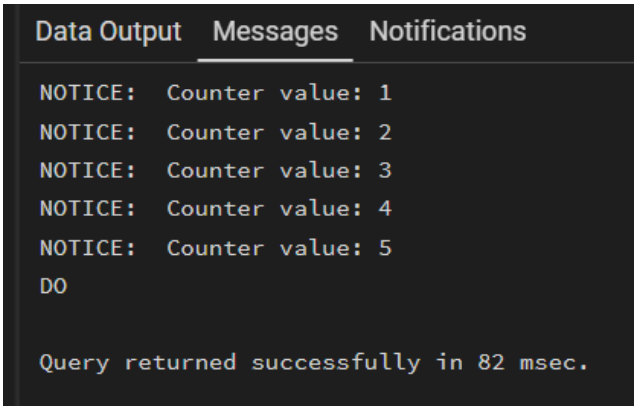


-- Example 1: FOR Loop

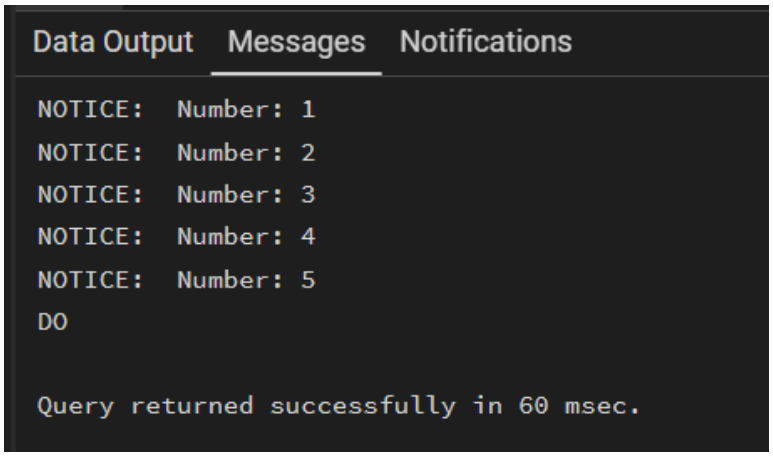
-- Example 2: FOR Loop with Query



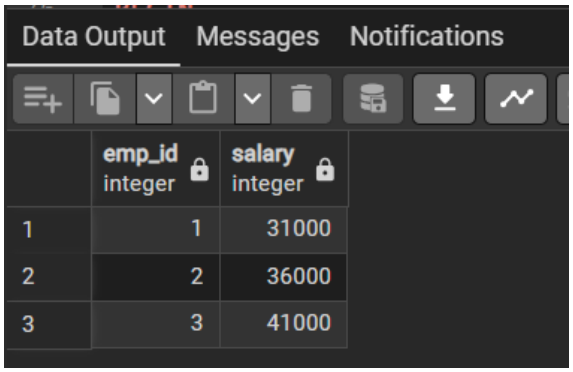
-- Example 3: WHILE Loop



-- Example 4: LOOP with EXIT WHEN

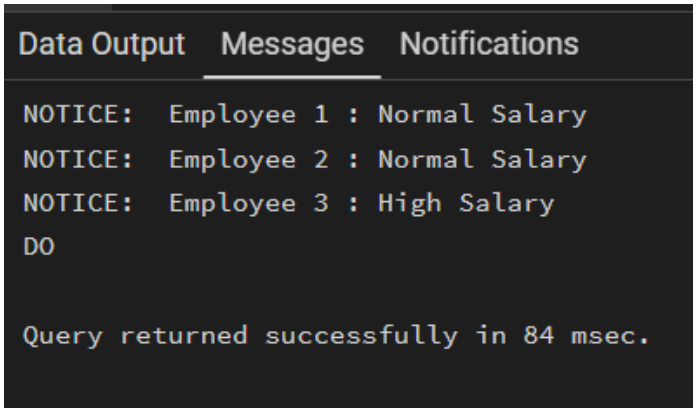


-- Example 5: Salary Increment Using FOR Loop



	emp_id integer	salary integer
1	1	31000
2	2	36000
3	3	41000

-- Example 6: Combining LOOP with IF Condition



Learning Outcome:

- Understand the need for iteration in database programming and procedural SQL.
- Learn the working principles and use cases of FOR, WHILE, and LOOP constructs in PostgreSQL.
- Gain conceptual clarity on how repeated data processing is handled using PL/pgSQL.
- Relate loop-based database logic to real-world applications such as payroll, reporting, and batch jobs.
- Develop foundational knowledge required to write procedural and enterprise-level database programs.