CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

Worksheet No. 4

Student Name: ANINDITA DHAR **UID:** 25MCA20259
Branch: MCA (GENERAL) **Section/Group:** MCA- 1-A
Semester: II **Date of Performance:** 01/02/2026

Subject Name: Technical Training **Subject Code:** 25CAP-652

1.Aim/Overview of the practical:

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.

2. Objectives:

- 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

3. Input/Apparatus Used:

- PostgreSQL
- pgAdmin

4. Practical / Experiment Steps

Pre-requisite Understanding

• Students should first understand that iterative control structures are executed inside PL/pgSQL blocks, not in normal SQL queries

• Students should create a table that stores multiple records so that loop execution over rows can be demonstrated

• The table should contain:

- A unique identifier
- A descriptive attribute (such as name or category)
- A numeric value to be processed repeatedly

4.Procedure/Algorithm/Code:

i.Start the system and log in to the computer.
ii. Open PostgreSQL software.

iii. **Create a table using the DDL command.**

```
CREATE TABLE  Employees (
emp_id SERIAL PRIMARY KEY,
emp_name VARCHAR(100),
salary NUMERIC(10,2)
);
```

iv. Insert records into the table named customer_orders.

```
INSERT INTO Employees VALUES('Rahul', 40000),
INSERT INTO Employees VALUES('Ankit', 52000),
INSERT INTO Employees VALUES('Priya', 60000),
INSERT INTO Employees VALUES('Neha', 35000),
INSERT INTO Employees VALUES('Aman', 70000);
```

vi. Display all records.

```
SELECT * FROM Employees;
```

	emp_id [PK] integer	emp_name character varying (50)	salary numeric (10,2)
1	6	Rahul	40000.00
2	7	Ankit	52000.00
3	8	Priya	60000.00
4	9	Neha	35000.00
5	10	Aman	70000.00

vii. (Step 1) FOR Loop – Simple Iteration

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

```
NOTICE:  Iteration number: 1
NOTICE:  Iteration number: 2
NOTICE:  Iteration number: 3
NOTICE:  Iteration number: 4
NOTICE:  Iteration number: 5
DO

Query returned successfully in 62 msec.
```

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

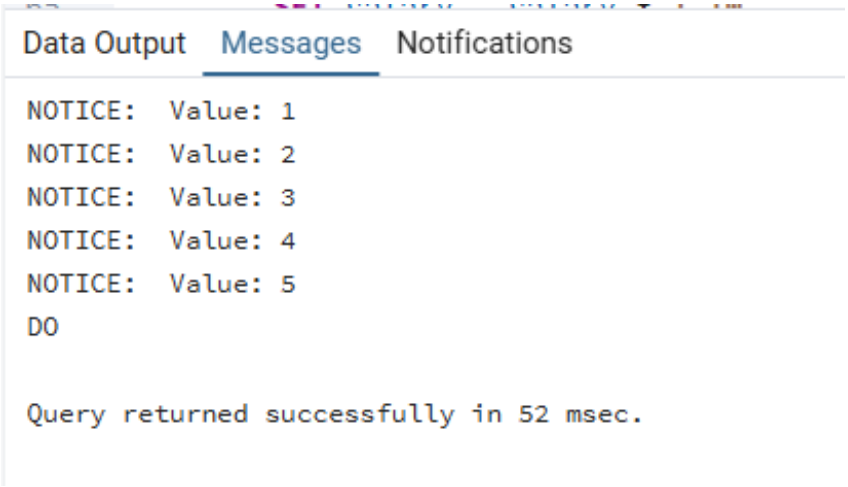
```
DO $$
DECLARE
   emp RECORD;
BEGIN
   FOR emp IN SELECT emp_id, emp_name FROM Employees LOOP
      RAISE NOTICE 'Employee ID: %, Name: %', emp.emp_id, emp.emp_name;
   END LOOP;
END $$;
```

```
NOTICE:  Employee ID: 6, Name: Rahul
NOTICE:  Employee ID: 7, Name: Ankit
NOTICE:  Employee ID: 8, Name: Priya
NOTICE:  Employee ID: 9, Name: Neha
NOTICE:  Employee ID: 10, Name: Aman
DO

Query returned successfully in 52 msec.
```

ix.(Step 3) WHILE Loop – Conditional Iteration

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



x. (Step 4) LOOP with EXIT WHEN

```
DO $$
DECLARE
  x INT := 1;
BEGIN
    LOOP
        RAISE NOTICE 'Value: %', x;
        x := x + 1;

        EXIT WHEN x > 5;
    END LOOP;
END $$.
```

Data Output | Messages | Notifications

```
NOTICE:  Counter: 1
NOTICE:  Counter: 2
NOTICE:  Counter: 3
NOTICE:  Counter: 4
NOTICE:  Counter: 5
DO

Query returned successfully in 39 msec.
```

xi.(Step 5) Salary Increment Using FOR Loop

```
DO $$
DECLARE
   emp RECORD;
BEGIN
   FOR emp IN SELECT emp_id, salary FROM Employees LOOP
      UPDATE employee
      SET salary = salary * 1.10
      WHERE emp_id = emp.emp_id;
   END LOOP;
END $$;
select*from Employees;
```

	emp_id [PK] integer	emp_name character varying (50)	salary numeric (10,2)
1	6	Rahul	44000.00
2	7	Ankit	57200.00
3	8	Priya	66000.00
4	9	Neha	38500.00
5	10	Aman	77000.00

Step 6: Combining LOOP with IF Condition

```
DO $$
DECLARE
   emp RECORD;
BEGIN
   FOR emp IN SELECT emp_name, salary FROM Employees  LOOP
      IF emp. salary > 50000 THEN
         RAISE NOTICE '% is a High Earner', emp.emp_name;
      ELSE
         RAISE NOTICE '% is a Regular Employee', emp.emp_name;
      END IF;
   END LOOP;
END $$;
```

Data Output | Messages | Notifications

```
NOTICE:  Rahul is a Regular Employee
NOTICE:  Ankit is a High Earner
NOTICE:  Priya is a High Earner
NOTICE:  Neha is a Regular Employee
NOTICE:  Aman is a High Earner
DO

Query returned successfully in 82 msec.
```

6. I/O Analysis (Input / Output)

Input:

- Employee or sample records inserted into a database table for iterative processing
- PL/pgSQL DO block containing procedural logic
- FOR loop constructs for fixed-range and query-based iteration
- WHILE loop with condition-based execution
- LOOP construct with explicit EXIT WHEN condition
- IF–ELSE conditions used inside loops for decision making
- UPDATE statements executed repeatedly within loop structures

Output:

- Repeated execution of SQL statements based on loop conditions
- Row-by-row processing of table records using FOR loops
- Conditional messages are displayed during each iteration
- Successful salary updates or value modifications through iterative logic
- Proper termination of loops based on defined conditions
- Correct execution of procedural SQL demonstrating iteration control

7. Learning Outcomes

- Understood the need for iterative control structures in database programming
- Learned the usage of FOR, WHILE, and LOOP constructs in PostgreSQL
- Gained knowledge of executing repeated logic using PL/pgSQL
- Understood row-by-row processing and conditional execution in the database
- Developed foundational skills for writing procedural SQL in real-world applications