UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

EXPERIMENT 4

Student Name: Jatin Tasoria **UID: 25MCC20054**

Branch: MCA-Cloud Computing **Section/Group: MCC 101-A**

Semester: 2nd **Date of Performance: 03/02/26**

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

1. 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.

2. Objective:

After completing this practical, the learner will be able to:

- 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.

3. Practical / Experiment Steps

a. Start the system.
b. Open pgAdmin.
c. Create and select the database in which you want to perform the experiment.
d. Establish connection to the database using Alt+Shift+Q.
e. Run the following queries given in Experiment Steps (5).

4. Procedure of the Practical

- 1. Start the system and log in
- 2. Start the PostgreSQL service
- 3. Open PostgreSQL client (psql or pgAdmin)
- 4. Create a new database
- 5. Create the required table
- 6. Insert sample data into the table
- 7. Execute SELECT queries for LOOP, WHILE, FOR
- 8. Ensure successful execution of queries
- 9. Save the work for documentation

5. SQL Queries Used in the Experiment

a. For Loop (simple iteration):

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

b. For Loop with Query (Row-by-Row Processing):

```
DO $$
DECLARE
   rec RECORD;
BEGIN
   FOR rec IN
      SELECT id, marks FROM  students
   LOOP
      RAISE NOTICE 'ID: %, Marks: %', rec.id, rec.marks;
   END LOOP;
END $$;
```

c. While Loop (conditional iteration):

```
DO $$
```

```
DECLARE
        i int:=1;
BEGIN
        WHILE i<=5 LOOP
                RAISE NOTICE 'WHILE LOOP turn %',i;
                i:=i+1;
        END LOOP;
END $$
```

d. Loop with 'EXIT WHEN':

```
DO $$
DECLARE
        i int :=1;
BEGIN
        LOOP
                RAISE NOTICE 'i=%',i;
                i:=i+1;
                EXIT WHEN i>5;
        END LOOP;
END $$;
```

e. Salary increment using For Loop:

```
DO $$
DECLARE
   rec RECORD;
BEGIN
   FOR rec IN
      SELECT id, salary FROM salaries
   LOOP
      UPDATE salaries
      SET salary = rec.salary * 1.10
      WHERE id = rec.id;
   END LOOP;
END $$;
```

f. Combining Loop with IF condition:

```
DO $$
DECLARE
   rec RECORD;
BEGIN
   FOR rec IN
      SELECT id, marks FROM students
   LOOP
      IF rec.marks > 50 THEN
         RAISE NOTICE 'ID: %, Marks: %', rec.id, rec.marks;
      END IF;
   END LOOP;
END $$;
```

6. Input / Output Analysis (I/O Analysis)

- For Loop (simple iteration):

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

- For Loop with Query (Row-by-Row Processing):

```
NOTICE:  ID: 1, Marks: 45
NOTICE:  ID: 2, Marks: 72
NOTICE:  ID: 3, Marks: 88
NOTICE:  ID: 4, Marks: 39
NOTICE:  ID: 5, Marks: 61
DO
```

- While Loop (conditional iteration):

```
NOTICE:  WHILE LOOP turn 1
NOTICE:  WHILE LOOP turn 2
NOTICE:  WHILE LOOP turn 3
NOTICE:  WHILE LOOP turn 4
NOTICE:  WHILE LOOP turn 5
DO
```

- Loop with 'EXIT WHEN':

```
NOTICE:  i=1
NOTICE:  i=2
NOTICE:  i=3
NOTICE:  i=4
NOTICE:  i=5
DO
```

- Salary increment using For Loop:

```
DO

Query returned successfully in 74 msec.
```

- Combining Loop with IF condition:

```
NOTICE:  ID: 2, Marks: 72
NOTICE:  ID: 3, Marks: 88
NOTICE:  ID: 5, Marks: 61
DO
```

7. Learning Outcome

a. Understanding of how iterative control structures work in PostgreSQL at a conceptual level.

b. Usage of loops in database systems, such as workflow engines, complex decision cycles, validation loops, etc.

c. Foundational knowledge required for writing procedural logic in enterprise-grade applications.