Experiment No. 4

Student Name: Sanchit Katoch
UID: 25MCA20059
Branch: MCA-General
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:

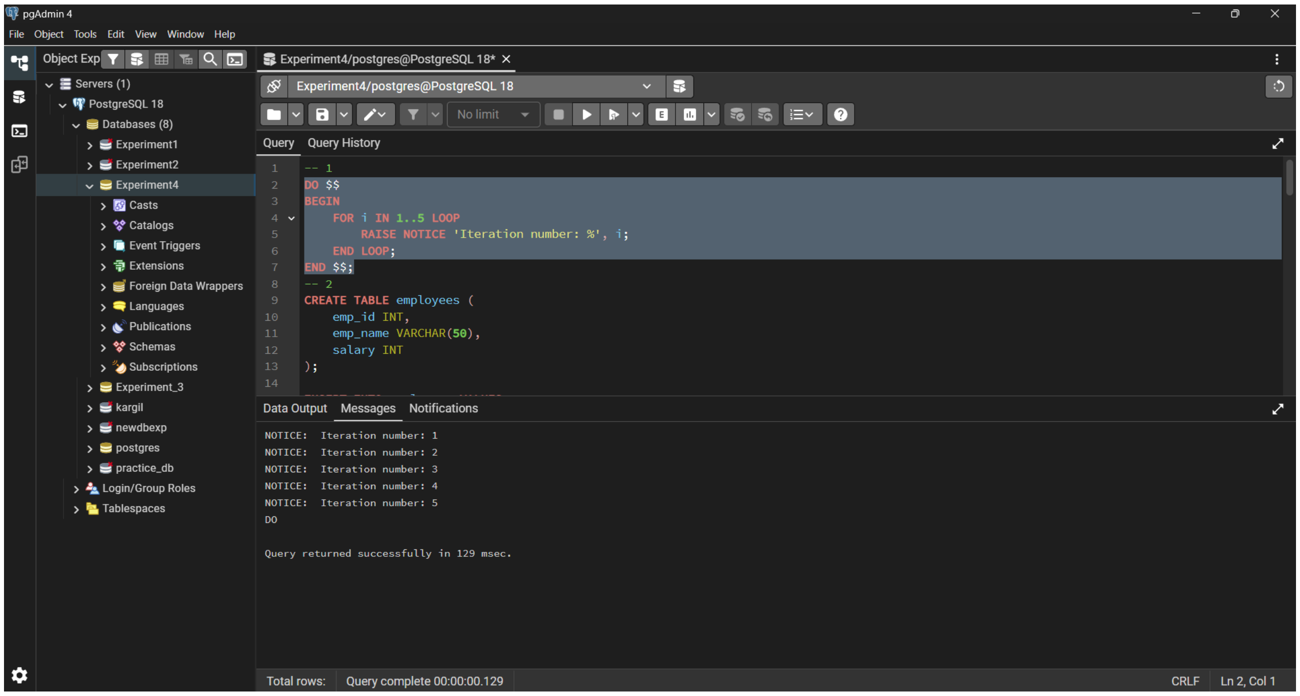
```
--1
DO $$
BEGIN
   FOR i IN 1..5 LOOP
       RAISE NOTICE 'Iteration number: %', i;
```

```
    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:

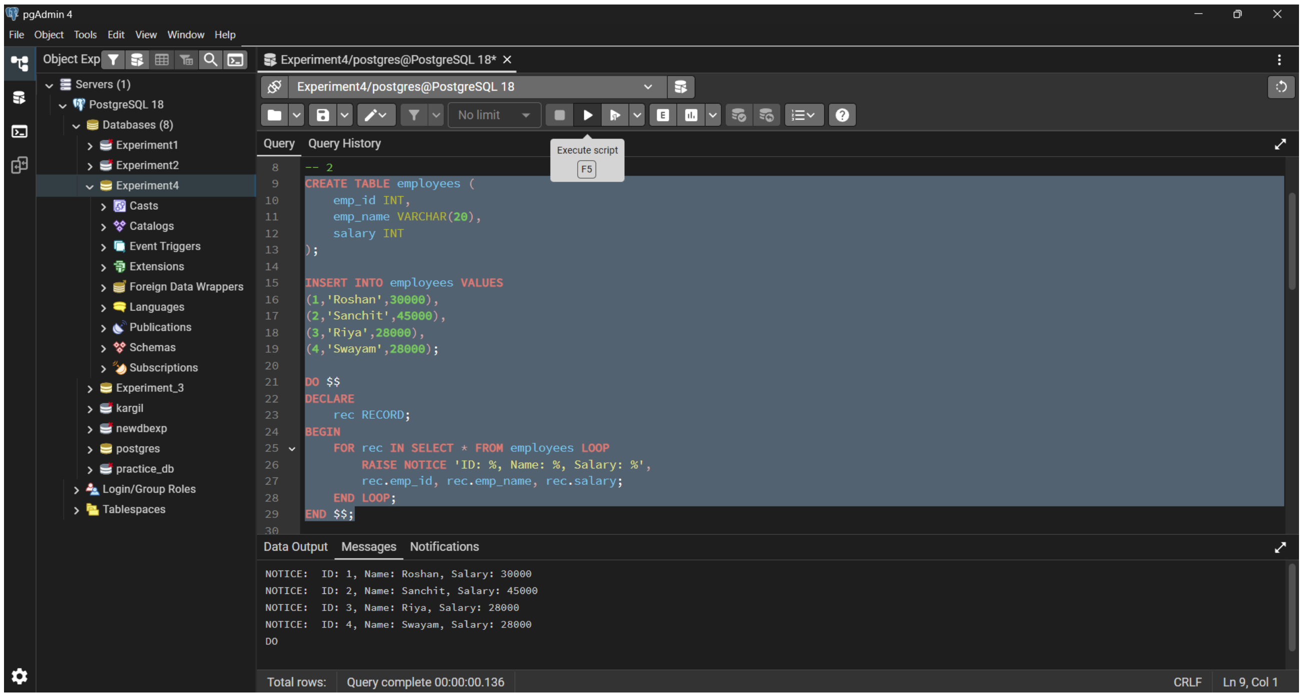
Step1: FOR Loop - Simple Iteration


pgAdmin 4
File Object Tools Edit View Window Help
Experiment4/postgres@PostgreSQL 18*
Query Query History
Data Output Messages Notifications
Total rows: Query complete 00:00:00.129
CRLF Ln 2, Col 1

```
-- 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
);
```

```
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 129 msec.
```

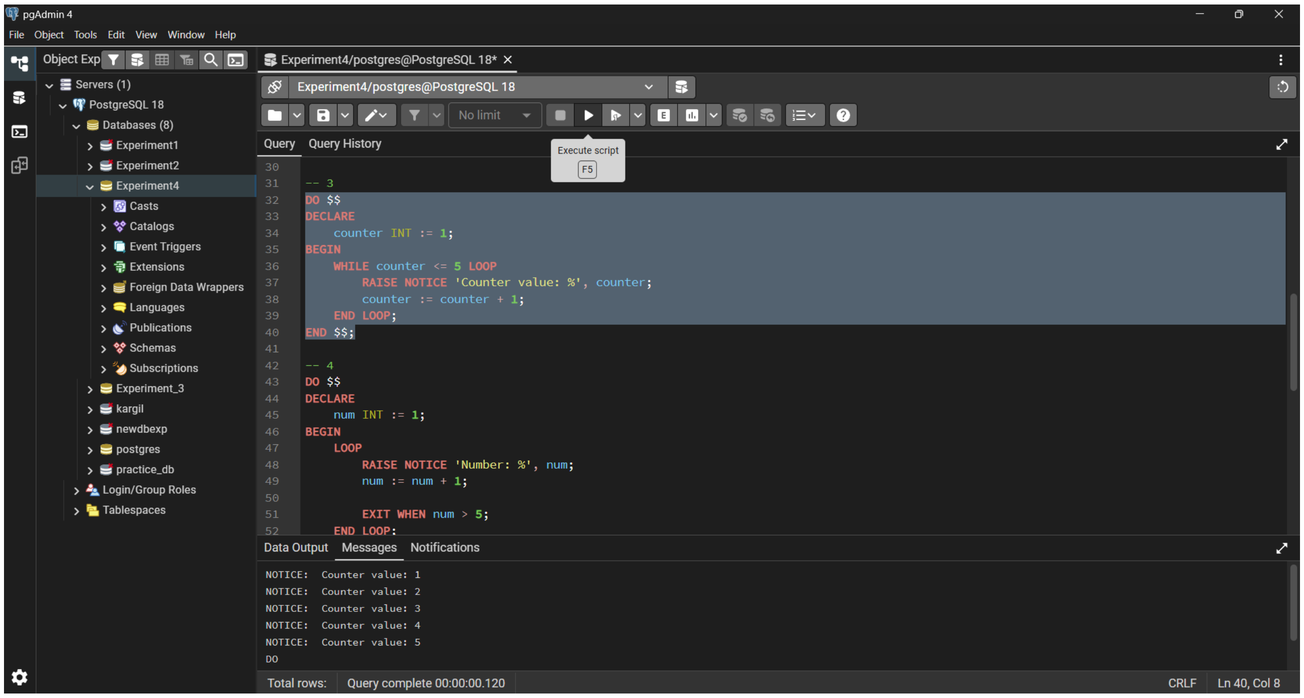
Step2: FOR Loop with Query (Row-by-Row Processing)


pgAdmin 4
File Object Tools Edit View Window Help
Experiment4/postgres@PostgreSQL 18*
Query Query History
Execute script F5
Data Output Messages Notifications
Total rows: Query complete 00:00:00.136
CRLF Ln 9, Col 1

```
-- 2
CREATE TABLE employees (
    emp_id INT,
    emp_name VARCHAR(20),
    salary INT
);

INSERT INTO employees VALUES
(1,'Roshan',30000),
(2,'Sanchit',45000),
(3,'Riya',28000),
(4,'Swayam',28000);

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

```
NOTICE:  ID: 1, Name: Roshan, Salary: 30000
NOTICE:  ID: 2, Name: Sanchit, Salary: 45000
NOTICE:  ID: 3, Name: Riya, Salary: 28000
NOTICE:  ID: 4, Name: Swayam, Salary: 28000
DO
```

Step3: WHILE Loop - Conditional Iteration

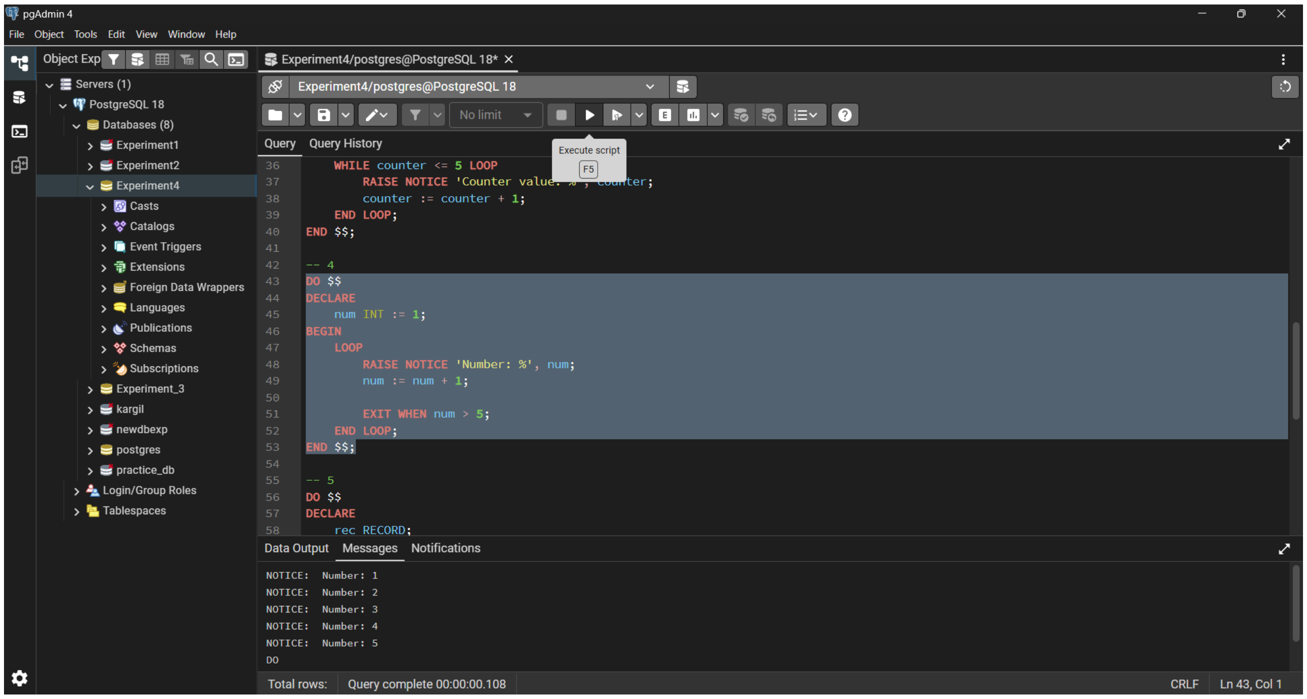


```
-- 3
DO $$
DECLARE
    counter INT := 1;
BEGIN
    WHILE counter <= 5 LOOP
        RAISE NOTICE 'Counter value: %', counter;
        counter := counter + 1;
    END LOOP;
END $$;

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

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

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

Total rows: Query complete 00:00:00.120 CRLF Ln 40, Col 8

Step4: LOOP with EXIT WHEN

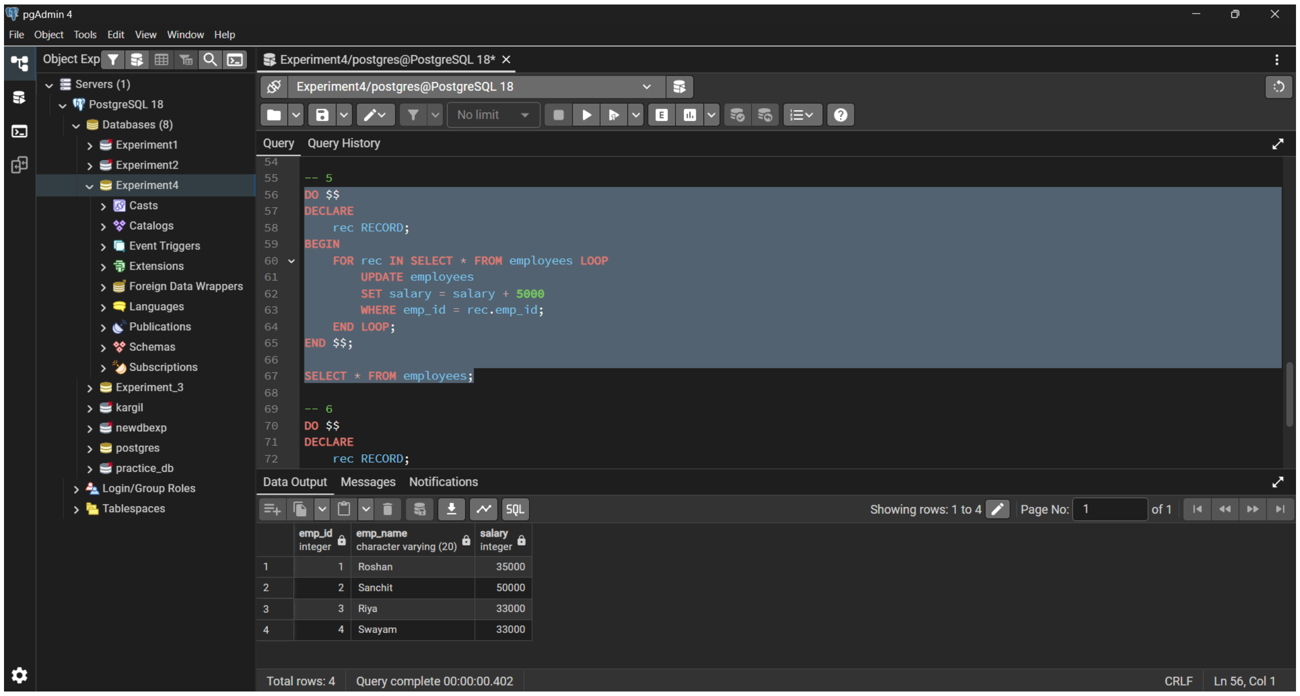


```
    WHILE counter <= 5 LOOP
        RAISE NOTICE 'Counter value: %', counter;
        counter := counter + 1;
    END LOOP;
END $$;

-- 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;
```

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

Total rows: Query complete 00:00:00.108 CRLF Ln 43, Col 1

Step5: Salary Increment Using FOR Loop

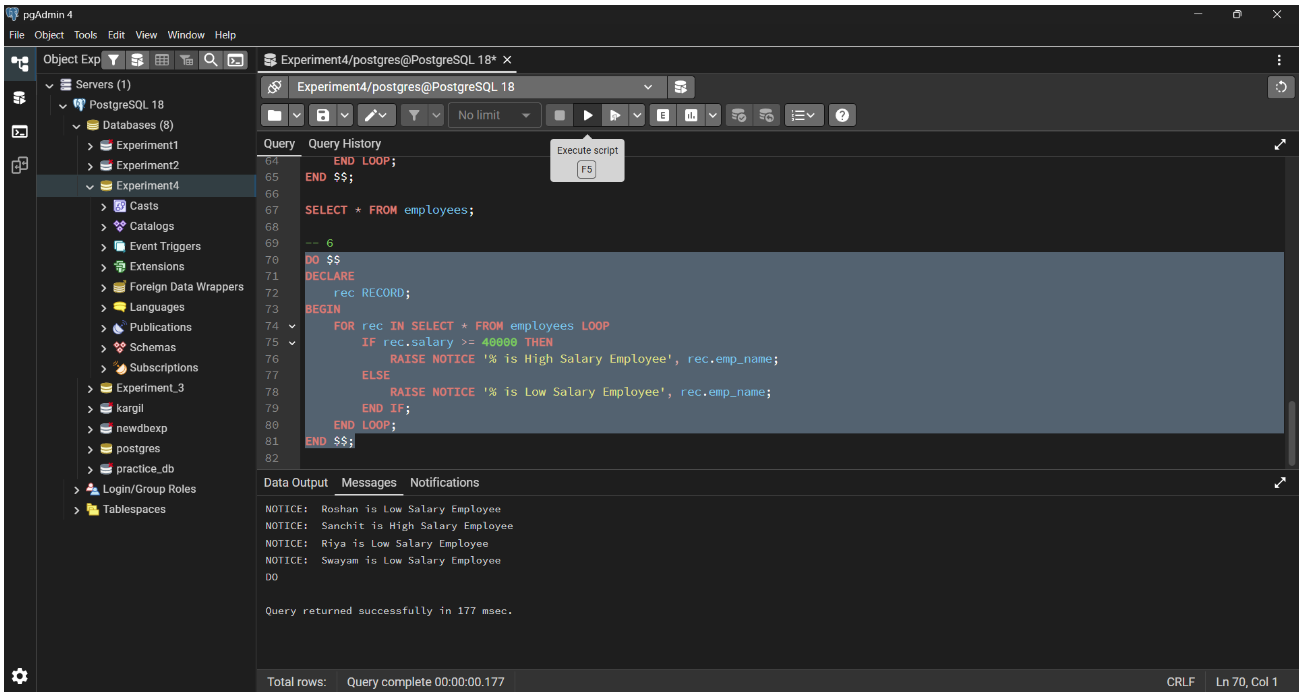


```
-- 5
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT * FROM employees LOOP
        UPDATE employees
        SET salary = salary + 5000
        WHERE emp_id = rec.emp_id;
    END LOOP;
END $$;

SELECT * FROM employees;

-- 6
DO $$
DECLARE
    rec RECORD;
```

	emp_id integer	emp_name character varying (20)	salary integer
1	1	Roshan	35000
2	2	Sanchit	50000
3	3	Riya	33000
4	4	Swayam	33000

Total rows: 4   Query complete 00:00:00.402

Step6: Combining LOOP with IF Condition



```
    END LOOP;
END $$;

SELECT * FROM employees;

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

```
NOTICE:  Roshan is Low Salary Employee
NOTICE:  Sanchit is High Salary Employee
NOTICE:  Riya is Low Salary Employee
NOTICE:  Swayam is Low Salary Employee
DO

Query returned successfully in 177 msec.
```

Total rows: Query complete 00:00:00.177

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.