Student Name: KAPISH GUPTA UID: 24BAI70577
Branch: AIT-CSE (AIML) Section/Group: 24AIT_KRG2
Semester: 4
Subject Name: Database Management System Subject Code: 24CSH-298

## Experiment

## Aim

To design and implement PL/SQL programs utilizing conditional control statements such as **IF–ELSE, ELSIF, ELSIF ladder, and CASE constructs** in order to control the flow of execution based on logical conditions and to analyze decision-making capabilities in PL/SQL blocks.

## Software Requirements

**Database Management System:**
PostgreSQL

**Database Administration Tool:**
pgAdmin

## Objective

To implement control structures in PL/SQL such as:

- IF–ELSE
- IF–ELSIF–ELSE
- ELSIF Ladder
- CASE Statements
  in PL/SQL blocks to control program flow based on conditions.

## Practical / Experiment Steps

 Start the system and log in.

- Open **pgAdmin**.
- Connect to the PostgreSQL server.
- Open the **Query Tool**.
- Write PL/pgSQL blocks using DO $$.
- Declare variables in the DECLARE section.
- Apply conditional statements:
  - IF–ELSE
  - IF–ELSIF–ELSE
  - ELSIF Ladder
  - CASE
- Use RAISE NOTICE to display output.
- Execute each program.
- Verify output in the **Messages/Output panel**.
- Take screenshots of code and output.
- Save the work.

**Procedure of the Experiment**

1. Launch pgAdmin and connect to PostgreSQL.
2. Open the Query Tool.
3. Create a PL/pgSQL block using DO $$.
4. Declare required variables in the DECLARE section.
5. Write conditional logic using:
   - IF–ELSE
   - IF–ELSIF–ELSE
   - ELSIF ladder
   - CASE statement
6. Use RAISE NOTICE for output display.
7. Execute the code block.

8. Observe the output in the message console.
9. Validate logical conditions.
10. Record results and screenshots.

**CODE :**

**1) IF–ELSE Statement**

(Check whether number is positive or non-positive)

```
DO $$
DECLARE
   num INTEGER := -5;
BEGIN
   IF num > 0 THEN
      RAISE NOTICE 'Number is Positive';
   ELSE
      RAISE NOTICE 'Number is Non-Positive';
   END IF;
END;
$$ LANGUAGE plpgsql;
```

---

**2) IF–ELSIF–ELSE Statement**

**(Grade of student based on marks)**

```
DO $$
DECLARE
   marks INTEGER := 85;
BEGIN
   IF marks >= 90 THEN
      RAISE NOTICE 'Grade: A+';
   ELSIF marks >= 80 THEN
```

```
        RAISE NOTICE 'Grade: A';
    ELSIF marks >= 70 THEN
        RAISE NOTICE 'Grade: B';
    ELSIF marks >= 60 THEN
        RAISE NOTICE 'Grade: C';
    ELSE
        RAISE NOTICE 'Grade: Fail';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

---

**3) ELSIF Ladder**

**(Performance status of student)**

```
DO $$
DECLARE
    marks INTEGER := 72;
BEGIN
    IF marks >= 90 THEN
        RAISE NOTICE 'Performance: Excellent';
    ELSIF marks >= 75 THEN
        RAISE NOTICE 'Performance: Very Good';
    ELSIF marks >= 60 THEN
        RAISE NOTICE 'Performance: Good';
    ELSIF marks >= 50 THEN
        RAISE NOTICE 'Performance: Average';
    ELSE
        RAISE NOTICE 'Performance: Poor';
    END IF;
```

```
END;
$$ LANGUAGE plpgsql;
```

---

**4) CASE Statement**

**(Display day name using day number)**

```
DO $$
DECLARE
   day_no INTEGER := 3;
   day_name VARCHAR(20);
BEGIN
   day_name := CASE day_no
      WHEN 1 THEN 'Monday'
      WHEN 2 THEN 'Tuesday'
      WHEN 3 THEN 'Wednesday'
      WHEN 4 THEN 'Thursday'
      WHEN 5 THEN 'Friday'
      WHEN 6 THEN 'Saturday'
      WHEN 7 THEN 'Sunday'
      ELSE 'Invalid Day'
   END;

   RAISE NOTICE 'Day is: %', day_name;
END;
$$ LANGUAGE plpgsql;
```

## Learning Outcomes:

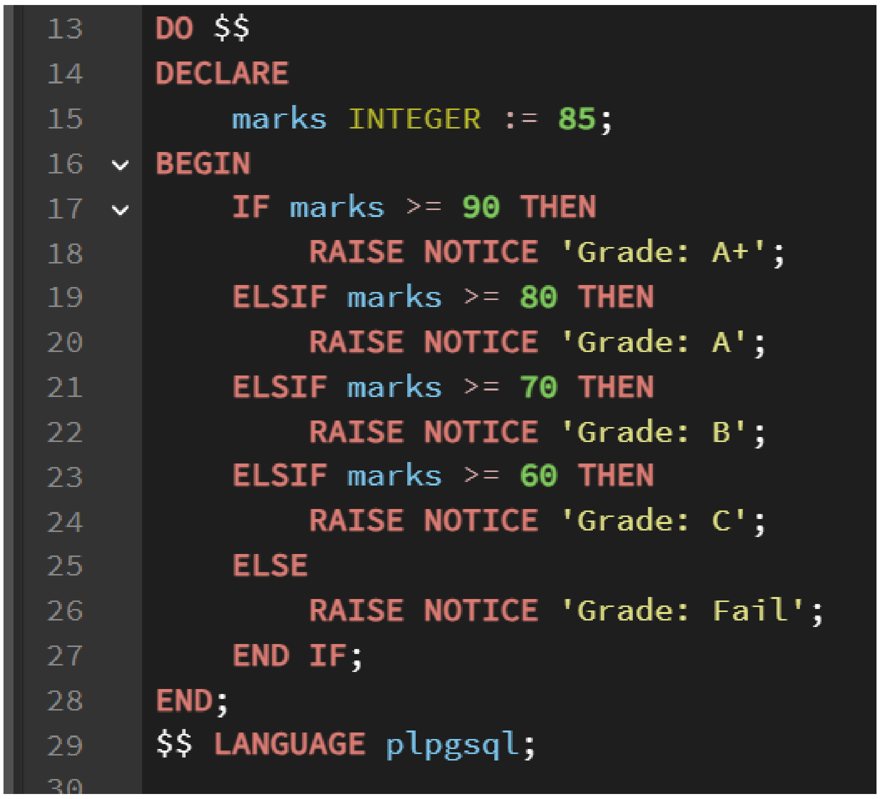
- Understood the structure of a PL/SQL block.
- Learned about declaration and execution sections.

• Learned how to declare and initialize variables.

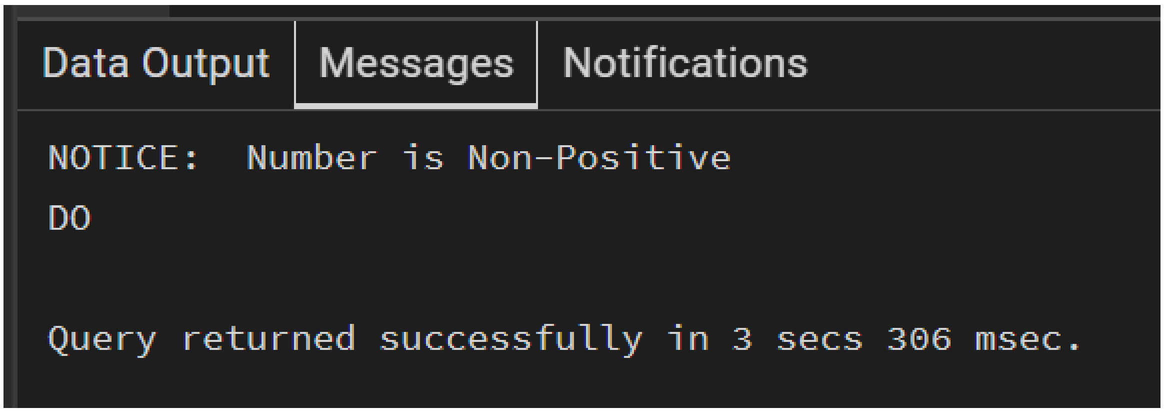
• Understood usage of DBMS_OUTPUT.PUT_LINE.

• Gained basic hands-on experience in PL/SQL programming.

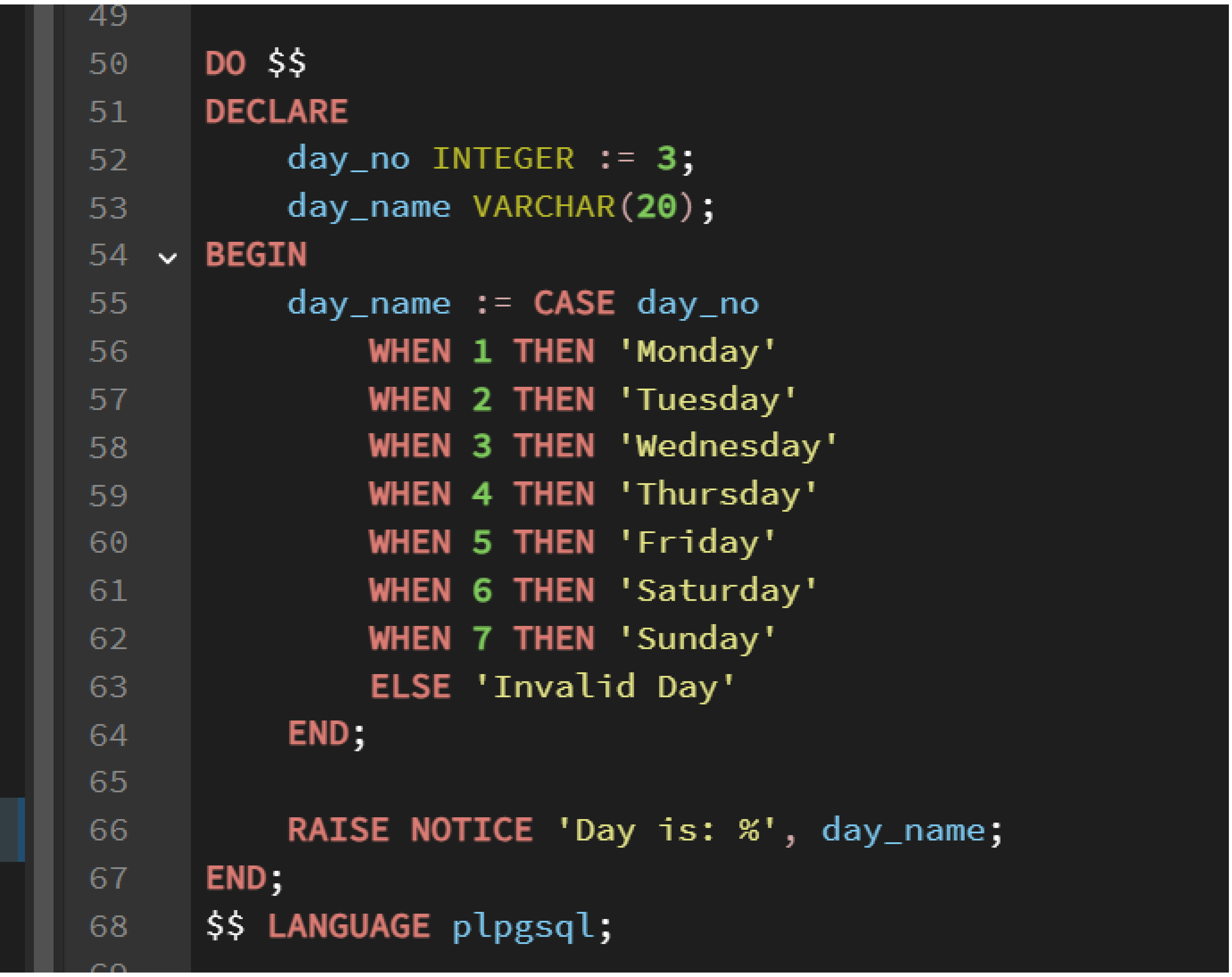
SCREENSHOTS



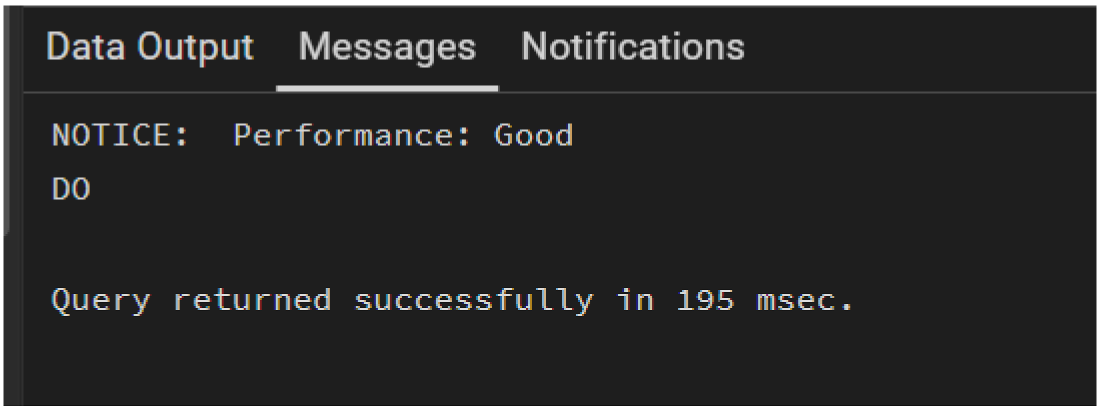
```
DO $$
DECLARE
    marks INTEGER := 85;
BEGIN
    IF marks >= 90 THEN
        RAISE NOTICE 'Grade: A+';
    ELSIF marks >= 80 THEN
        RAISE NOTICE 'Grade: A';
    ELSIF marks >= 70 THEN
        RAISE NOTICE 'Grade: B';
    ELSIF marks >= 60 THEN
        RAISE NOTICE 'Grade: C';
    ELSE
        RAISE NOTICE 'Grade: Fail';
    END IF;
END;
$$ LANGUAGE plpgsql;
```



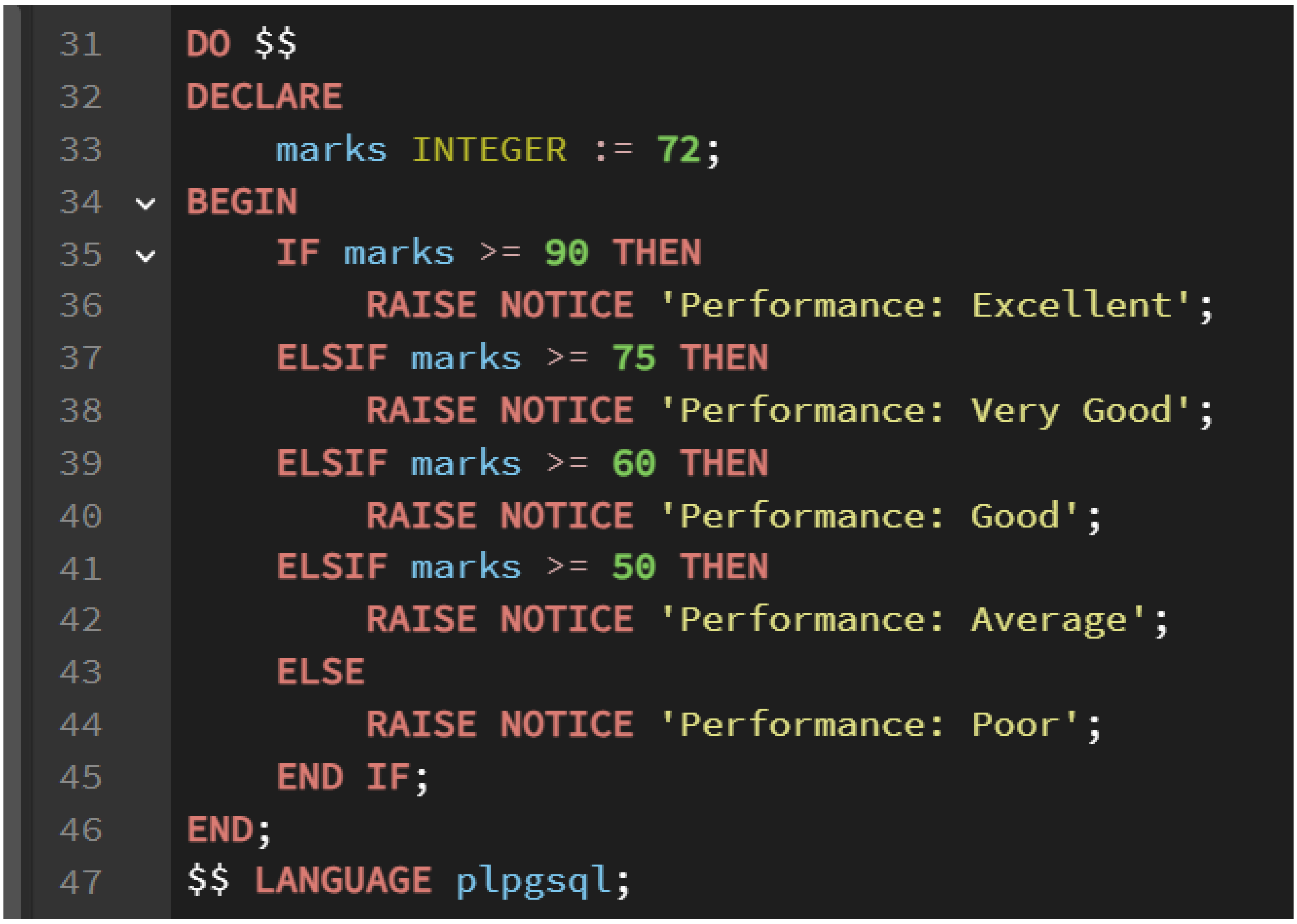
```
NOTICE:  Number is Non-Positive
DO

Query returned successfully in 3 secs 306 msec.
```

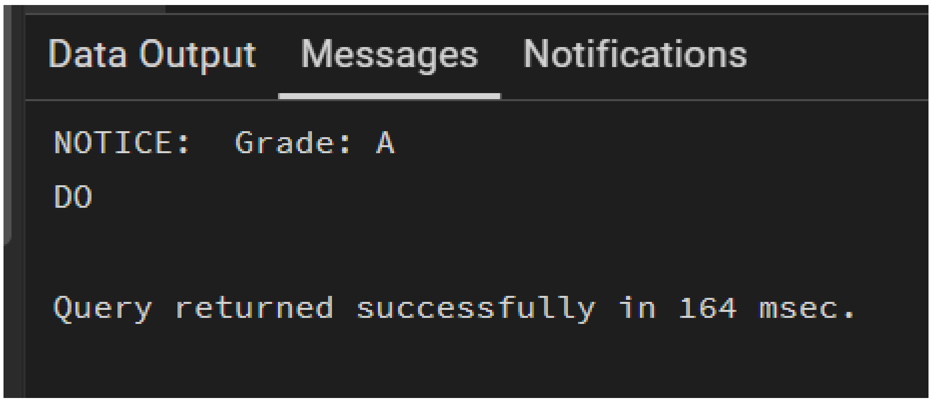


```sql
DO $$
DECLARE
    day_no INTEGER := 3;
    day_name VARCHAR(20);
BEGIN
    day_name := CASE day_no
        WHEN 1 THEN 'Monday'
        WHEN 2 THEN 'Tuesday'
        WHEN 3 THEN 'Wednesday'
        WHEN 4 THEN 'Thursday'
        WHEN 5 THEN 'Friday'
        WHEN 6 THEN 'Saturday'
        WHEN 7 THEN 'Sunday'
        ELSE 'Invalid Day'
    END;

    RAISE NOTICE 'Day is: %', day_name;
END;
$$ LANGUAGE plpgsql;
```



```
NOTICE:  Performance: Good
DO


Query returned successfully in 195 msec.
```


```
DO $$
DECLARE
    marks INTEGER := 72;
BEGIN
    IF marks >= 90 THEN
        RAISE NOTICE 'Performance: Excellent';
    ELSIF marks >= 75 THEN
        RAISE NOTICE 'Performance: Very Good';
    ELSIF marks >= 60 THEN
        RAISE NOTICE 'Performance: Good';
    ELSIF marks >= 50 THEN
        RAISE NOTICE 'Performance: Average';
    ELSE
        RAISE NOTICE 'Performance: Poor';
    END IF;
END;
$$ LANGUAGE plpgsql;
```



```
NOTICE:  Grade: A
DO

Query returned successfully in 164 msec.
```