Experiment-3

Student Name: Harshit Kumawat
UID: 24BAI70025
Branch: CSE - AIML
Section/Group: 24AIT_KRG G1
Semester: 4
Date of Performance: 28/1/26
Subject Name: DBMS
Subject Code: 24CSH-298

Aim

To understand the basic structure of a PL/SQL program by creating and executing a simple PL/SQL block that includes declaration and execution sections, and to display output using built-in procedures.

Software Requirements

- Database Management System:
  - PostgreSQL
  - Oracle
- Database Administration Tool:
  - pgAdmin

Objectives

- To create a simple PL/SQL program demonstrating Declaration Section and Execution Section.

Problem Statement

Design and implement a simple PL/SQL program that demonstrates the basic structure of a PL/SQL block. The program should include a declaration section to define variables and an execution section to perform operations using those variables and display the results using appropriate output statements.
Declaration Section (DECLARE)
Variables are declared and initialized:

- emp_id → Employee ID
- emp_name → Employee Name

- emp_salary → Employee Salary

Execution Section (BEGIN ... END)
DBMS_OUTPUT.PUT_LINE is used to display output.

Practical/Experiment Steps

- Block Structuring: Designed a foundational PL/SQL block consisting of a Declaration section for memory allocation and an Execution section for logic processing.
- Variable Initialization: Defined and assigned static values to organizational variables, including Employee ID, Name, and Salary, to simulate a single-record data environment.
- Computational Logic: Integrated an arithmetic operation within the block to calculate the House Rent Allowance (HRA) at 40% of the base salary.
- Conditional Processing: Implemented a selection structure using an IF-ELSE statement to evaluate tax liability based on the defined salary threshold.
- Output Orchestration: Utilized the DBMS_OUTPUT.PUT_LINE procedure to format and display the processed information in the console.

Procedure

1. Setup and Preparation

- **Establish Connection:** Open your Database Administration Tool (pgAdmin) and establish a connection to your PostgreSQL database environment.
- **Identify Goal:** Confirm the goal is to create a foundational PL/SQL block demonstrating declaration, execution, and output.

2. Block Implementation: Declaration Section

- **Begin Block Structuring:** Initiate the procedural block with the `DO $$` and `END $$;` structure, and the core PL/SQL block with the `DECLARE` and `BEGIN` keywords.
- **Reserve Memory:** Within the `DECLARE` section, define and reserve memory for the required variables:
    - `emp_id` (INT)
    - `emp_name` (VARCHAR)

    - emp_salary (NUMERIC)
- **Variable Initialization:** Assign static initial values to simulate a single-record data environment, for example:
    - emp_id := 101
    - emp_name := 'THOMAS'
    - emp_salary := 25000

3. Execution Logic and Output

- **Start Execution:** Begin the execution part of the code after the BEGIN keyword.
- **Display Primary Details:** Utilize the built-in output procedure (RAISE NOTICE in PostgreSQL's DO block, which is analogous to DBMS_OUTPUT.PUT_LINE) to display the primary employee details: ID, Name, and Salary.
- **Perform Calculation:** Integrate an arithmetic operation to calculate a derived value, such as the House Rent Allowance (HRA) at 40% of the base salary: (0.40 * emp_salary). Display this calculated value.
- **Implement Conditional Logic:** Apply a selection structure (an IF-ELSE statement) to evaluate a tax liability condition:
    - Check if emp_salary is greater than a defined threshold (e.g., 60,000).
    - Define the output path for the TRUE condition: RAISE NOTICE 'YOU NEED TO PAY TAX'.
    - Define the output path for the FALSE condition: RAISE NOTICE 'YOU DO NOT HAVE TO PAY TAX'.

4. Execution and Verification

- **Finalize and Execute:** Conclude the block with the END keyword and execute the entire script to trigger the PL/SQL engine.
- **Verify Output:** Check the console output against manual calculations to ensure:
    - All variable values were correctly displayed.
    - The HRA calculation is accurate.

- The conditional logic resulted in the correct tax status message for the assigned salary.

Input/Output Analysis

SQL Input Queries

```
DO $$
DECLARE
   emp_id     INT := 101;
   emp_name   VARCHAR(20) := 'THOMAS';
   emp_salary NUMERIC := 25000;
BEGIN
   RAISE NOTICE 'EMPLOYEE ID: %', emp_id;
   RAISE NOTICE 'EMPLOYEE NAME: %', emp_name;
   RAISE NOTICE 'SALARY: RS. %', emp_salary;
   RAISE NOTICE 'HOUSE RENT ALLOWANCE: RS. %', (0.40 * emp_salary);

   IF emp_salary > 60000 THEN
      RAISE NOTICE 'YOU NEED TO PAY TAX';
   ELSE
      RAISE NOTICE 'YOU DO NOT HAVE TO PAY TAX';
   END IF;
END $$;
```

Output

```
DO $$
DECLARE
    emp_id     INT := 101;
    emp_name   VARCHAR(20) := 'THOMAS';
    emp_salary NUMERIC := 25000;
BEGIN
    RAISE NOTICE 'EMPLOYEE ID: %', emp_id;
    RAISE NOTICE 'EMPLOYEE NAME: %', emp_name;
    RAISE NOTICE 'SALARY: RS. %', emp_salary;
    RAISE NOTICE 'HOUSE RENT ALLOWANCE: RS. %', (0.40 * emp_salary);

    IF emp_salary > 60000 THEN
        RAISE NOTICE 'YOU NEED TO PAY TAX';
    ELSE
        RAISE NOTICE 'YOU DO NOT HAVE TO PAY TAX';
    END IF;
END $$;
```

Data Output | Messages | Notifications

```
NOTICE:  EMPLOYEE ID: 101
NOTICE:  EMPLOYEE NAME: THOMAS
NOTICE:  SALARY: RS. 25000
NOTICE:  HOUSE RENT ALLOWANCE: RS. 10000.00
NOTICE:  YOU DO NOT HAVE TO PAY TAX
DO

Query returned successfully in 1 secs 0 msec.
```

Input/Output Analysis Table

Section	Element	Value / Description	Result / Expected Output
Input (Declaration Section)	`emp_id` (INT)	`101`	N/A
	`emp_name` (VARCHAR)	`'THOMAS'`	N/A
	`emp_salary`	`25000`	N/A

	(NUMERIC)		
Process (Execution Section)	HRA Calculation	`0.40 * emp_salary`	`0.40 * 25000…`
	Conditional Check	`emp_salary > 60000`	`25000 > 6000…`
Output (Execution Section)	Employee ID	`RAISE NOTICE 'EMPLOYEE ID: 101'`	EMPLOYEE ID…
	Employee Name	`RAISE NOTICE 'EMPLOYEE NAME: THOMAS'`	EMPLOYEE N…
	Salary	`RAISE NOTICE 'SALARY: RS. 25000'`	SALARY: RS. 2…
	House Rent Allowance	`RAISE NOTICE 'HOUSE RENT ALLOWANCE: RS. 10000'`	HOUSE RENT …
	Tax Status Message	Executes the `ELSE` block	YOU DO NOT …

Learning Outcomes

- Understanding the fundamental organisation of PL/SQL declaration and execution sections.
- Learnt to declare, initialise, and use different data types within a procedural block.
- Implementing conditional branching and basic arithmetic operations to process data dynamically.
- Utilising built-in procedures to format and display calculated results to the user.