Experiment No. 8

TITLE

Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

PROBLEM STATEMENT

Create and execute a PL/SQL stored procedure to update employee salaries by a percentage and a function to calculate the average salary of a given department. Demonstrate their usage on the employee database.

OBJECTIVE

- Understand the concept of PL/SQL stored procedures and functions.
- Write and execute a stored procedure to perform bulk salary updates.
- Write and execute a function to compute average salary per department.
- Learn how to call procedures and functions in PL/SQL.
- Analyze the effect of procedures and functions on the database.

PLATFORM REQUIRED

- Oracle Database (11g, 12c, 19c or later)
- Oracle SQL Developer or SQL*Plus
- Windows / Linux / macOS

THEORY

- PL/SQL is Oracle's procedural extension to SQL.
- **Stored Procedure**: A named PL/SQL block that performs tasks (e.g., update salaries), can take parameters, but does not return a value.
- **Function**: Similar to procedure, but **must return a value** and can be used in SQL queries.
- Both centralize business logic in the database, improve performance, security, and maintainability.

Database Setup (Employee Table)

```
CREATE TABLE employees (
   emp_id     NUMBER PRIMARY KEY,
   name       VARCHAR2(50),
   salary     NUMBER(10,2),
   dept_id    NUMBER
);


-- Insert sample records
INSERT INTO employees VALUES (101, 'Amit', 50000, 10);
```

```
INSERT INTO employees VALUES (102, 'Priya', 60000, 20);
INSERT INTO employees VALUES (103, 'Raj', 55000, 10);
INSERT INTO employees VALUES (104, 'Sneha', 70000, 30);
COMMIT;
```

Stored Procedure – Increase Salary by Percentage

```
CREATE OR REPLACE PROCEDURE IncreaseSalary (
   p_dept_id   IN NUMBER,
   p_percent   IN NUMBER
) AS
BEGIN
   UPDATE employees
   SET salary = salary + (salary * p_percent / 100)
   WHERE dept_id = p_dept_id;

   DBMS_OUTPUT.PUT_LINE('Salary updated for department ' || p_dept_id);
END;
/
```

Execution of Procedure

```
BEGIN
   IncreaseSalary(10, 10);  -- Increase salaries of dept 10 by 10%
END;
/
```

Function – Get Average Salary of Department

```
CREATE OR REPLACE FUNCTION GetAvgSalary (
   p_dept_id IN NUMBER
) RETURN NUMBER IS
   v_avg_salary NUMBER;
BEGIN
   SELECT AVG(salary)
   INTO v_avg_salary
```

```
    FROM employees
    WHERE dept_id = p_dept_id;
    RETURN v_avg_salary;
END;
/
```

Execution of Function

```
DECLARE
    avg_sal NUMBER;
BEGIN
    avg_sal := GetAvgSalary(10);
    DBMS_OUTPUT.PUT_LINE('Average salary of department 10 is: ' || avg_sal);
END;
/
```

Or directly in SQL:

```
SELECT GetAvgSalary(10) AS Avg_Salary FROM dual;
```

STEP-BY-STEP ALGORITHM

1. Connect to Oracle Database.
2. Create a stored procedure IncreaseSalary(department_id, percentage) to update salaries.
3. Create a function GetAvgSalary(department_id) that returns the average salary.
4. Execute the procedure to update salaries for a specific department.
5. Call the function to retrieve average salary.
6. Display results.

QUESTIONS

1. What is the difference between a stored procedure and a function in PL/SQL?
2. How do you pass parameters to a PL/SQL procedure?
3. Write a stored procedure to increase salary of employees by a fixed amount.
4. Explain how a function can be used within an SQL query.
5. How does exception handling work in PL/SQL?

CONCLUSION

This lab demonstrated creation and execution of PL/SQL stored procedures and functions. Procedures automate tasks like salary increments, while functions return computed results such as average salary. Centralizing business logic improves performance, maintainability, and security in database systems.