Experiment No 24: Create Procedures and stored procedures for modularity

I. **Practical Significance:** Procedures in PL/SQL is a named block of code that can be stored in a database. Procedures accept input parameters, perform a specific task. They provide a way to modularize code, enhance reusability, and promote efficient development practices. This practical allows students to implement PL/SQL procedures and call them to perform tasks.

II. **INDUSTRY / EMPLOYER EXPECTED OUTCOME:**

To implement PL/SQL procedure for modularity and to optimize performance, enhance security

III. **COURSE LEVEL LEARNING OUTCOMES (COS):**

CO4 - Implement PL/SQL codes for given application.

IV. **LABORATORY LEARNING OUTCOME:**

Create Procedures and stored procedures for modularity.

V. **Relevant Affective Domain related outcome(s)**

a. Follow precautionary measures.

b. Follow installation steps.

c. Follow ethical practices.

VI. **Relevant Theoretical Background**

PL/SQL procedures, also known as stored procedures, are subprograms that can be called to perform a specific action in a database. They can contain a series of SQL statements and take parameters. PL/SQL procedures can be stored in a database and called by name from an application.

Syntax for creating procedure:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [ (parameter [, parameter]) ]
IS
  [declaration_section]
BEGIN
  executable_section
[EXCEPTION
  exception_section]
```

END [procedure_name];

Example:

```
CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
  dbms_output.put_line('Hello World!');
END;
```

Calling a procedure:

```
BEGIN
  greetings;
END;
```

In the above example, procedure is called by its name *greetings*

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and RDBMS applications such as Oracle Express Edition,MySql,SQLite,Oracle Apex etc.	All

VIII. Procedure

1.Define the PL/SQL block structure for procedure

2.Implement the logic for the given problem

3. Call the stored procedure

IX. Result(s)

In this practical we studied to create procedure and stored procedures for modularity

X. Practical related questions (Provide space for answers)

Note: Below are a few sample questions for reference. Teacher must design more such questions to ensure the achievement of identified CO.

2] →

Step 1: Create the procedure
Use the CREATE OR REPLACE PROCEDURE statement to define your procedure.

Step 2: Invoke the procedure
Within an anonymous PL/SQL block, use the procedure name followed by the parameters (s) in the parentheses to call the procedure.

Ex:

```
CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
dbms_output.put_line ('Hello world!');
END;
```

Calling a procedure:

```
BEGIN
  greetings;
END;
```

3] →

Parameters in the procedure are:
1] IN parameter
2] OUT parameter
3] IN OUT parameter.

① IN Parameter

Used to pass Values to the procedure. The procedure Can read but not modify these Values.

Ex:

```
CREATE OR REPLACE PROCEDURE greet_user (p_name
IN VARCHAR2) AS
BEGIN
DBMS_OUTPUT.PUT_LINE ('Hello, ' || p_name || '!');
END greet_user;
/
```

② OUT Parameter

Used to return Values From the procedure to the Calling program Can modify these Values & the Calling program Can read them after the procedure Complete.

Ex:

```
CREATE OR REPLACE PROCEDURE get_age (p_age OUT NUMBER)
AS BEGIN
p_age := 25;
END get_age;
/
```

③ IN OUT Parameter

Used to pass initial Values to the procedure and return updated Values to the Calling program. The procedure Can both read & modify these Values.

* Exercise

1] →

```
CREATE OR REPLACE PROCEDURE emp_count (p_dept_no
IN NUMBER) AS V_emp_count NUMBER;
BEGIN
SELECT COUNT (*) INTO V_emp_count FROM employees
WHERE department_id = p_dept_no;
DBMS_OUTPUT.PUT_LINE (' NUMBER OF
employess in department ' || p_dept_no ||
':' || V_emp_count);
END Emp_count;
/
```

Call the procedure

```
BEGIN
emp_count (10);
END;
/
```

4] →

```
CREATE OR REPLACE PROCEDURE greet_user
(p_name IN VARCHAR2) AS
BEGIN
DBMS_OUTPUT.PUT_LINE ('Hello, ' || p_name || '!');
END greet user;
/
```

Call the Procedure.

```
BEGIN
greet_user('Alice');
END;
/
```

3] ⟶

```
CREATE OR REPLACE PROCEDURE insert_emp_records
AS BEGIN
INSERT INTO emp (emp_id, emp_name, dept_no, Salary)
VALUES (101, 'John Doe', 10, 50000);

INSERT INTO emp (emp_id, emp_name, dept_no, Salary)
VALUES (102, 'Jane Smith', 20, 60000);

INSERT INTO emp( emp_id, emp_name, dept_no, Salary)
VALUES (103, 'Alice Johson', 30, 55000);
COMMIT;
END insert_emp_records;
/
```

Call the Procedure

```
BEGIN
  insert_emp_records;
END;
/
```

1. Write syntax for creating PL/SQL Procedure.
2. Write steps to call a procedure in PL/SQL
3. List types of parameters in Procedure and explain them.

(Space for answer)

1) →

Syntax for creating PL/SQL procedure

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [(parameter) [, parameter])]
IS
   [declaration_section]
BEGIN
   executable_section
[EXCEPTION
      exception_section]
END [procedure_name];
```

XI. **Exercise**

1. Write a procedure emp_count () to count number of employees in department, use dept_no as input parameter
2. Create a stored procedure to accept name and greet user with name.