

# College of Engineering, Trivandrum

Department of Computer Science and Engineering



## CS333 APPLICATION SOFTWARE DEVELOPMENT LAB

---

### LABORATORY REPORT 12

Procedures, Functions and Packages

---

**Student Name**

1. Justine Biju(S5)

**Student ID**

170445(Roll No:37)

Submission Date : 19/08/2019

# 1 Introduction

## 1. Function

A function is a subprogram that computes a value. Functions and procedures are structured alike, except that functions have a RETURN clause.

The syntax is as follows:

```
FUNCTION name [(parameter[, parameter, ...])] RETURN  
datatype IS [local declarations]  
BEGIN  
executable  
statements  
[EXCEPTION  
exception  
handlers] END  
[name];
```

## 2. Procedure

A procedure is a subprogram that performs a specific action.

The syntax is as follows:

```
PROCEDURE name [(parameter[,  
parameter, ...])] IS [local declarations]  
BEGIN  
executable  
statements  
[EXCEPTION  
exception  
handlers] END  
[name];
```

## 3. Procedure

A package is a schema object that groups logically related PL/SQL types, items and subprograms.

The syntax is as follows:

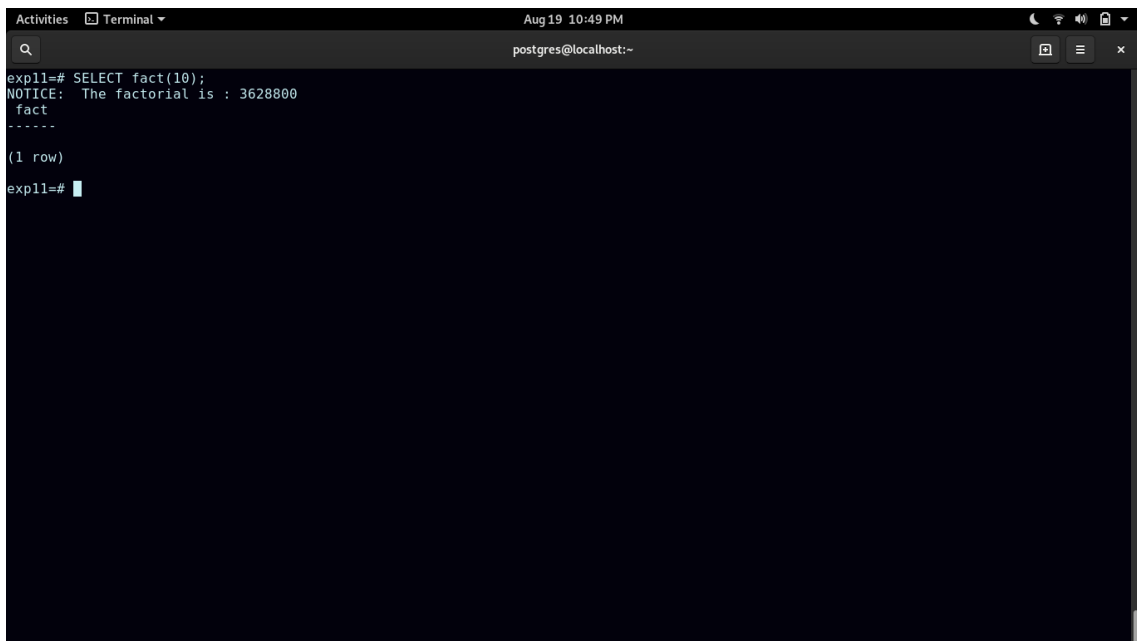
```
CREATE [OR REPLACE] PACKAGE  
package_name [AUTHID {CURRENT_USER |  
DEFINER}] {IS | AS} [type_definition  
[type_definition] ...]  
[cursor_spec [cursor_spec] ...]  
[item_declaration  
[item_declaration] ...]  
[{subprogram_spec | call_spec} [{subprogram_spec |  
call_spec}]...] END [package_name];
```

All these are supported by PostgreSQL and can be implemented in a similar fashion.

## 2 Questions

1. Create a function factorial to find the factorial of a number. Use this function in a PL/SQL Program

```
CREATE OR REPLACE FUNCTION fact(n integer) RETURNS VOID AS $$  
DECLARE  
    c INTEGER := 1;  
    prod INTEGER := 1;  
BEGIN  
    LOOP  
        EXIT WHEN c = n+1;  
        prod := prod * c;  
        c := c + 1;  
    END LOOP;  
    RAISE NOTICE 'The factorial is : %',prod;  
END;  
$$ LANGUAGE plpgsql;
```



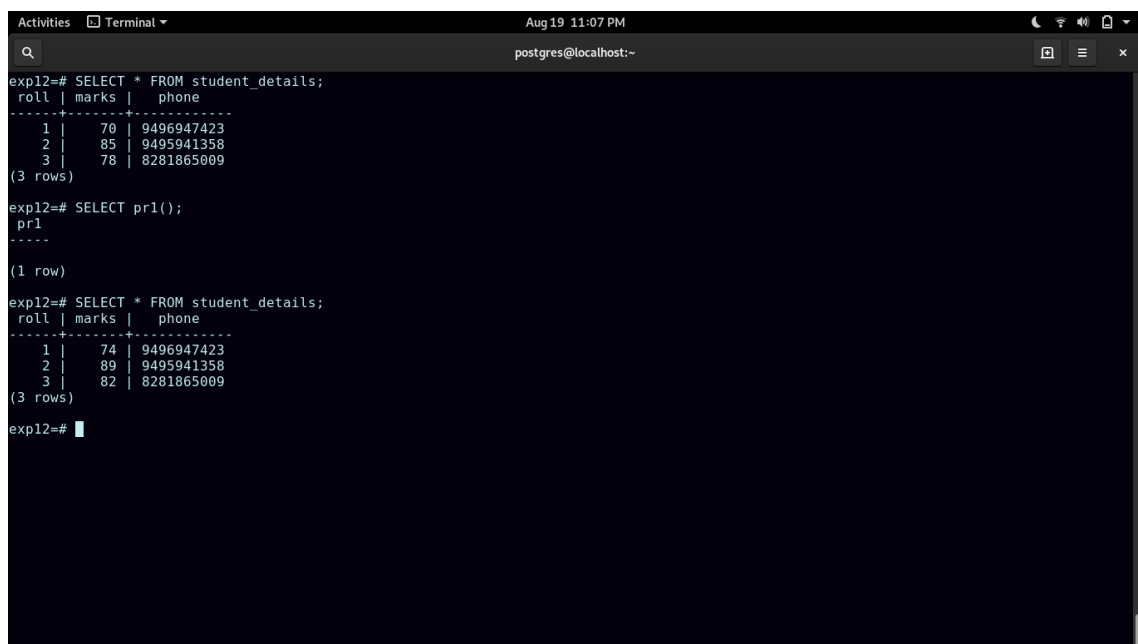
The screenshot shows a terminal window titled "Terminal" with a search bar and window controls. The terminal output shows a SQL prompt "exp11=#" followed by the command "SELECT fact(10);". The response is a "NOTICE: The factorial is : 3628800" followed by a table header "fact" and a separator line "-----". The table contains one row with the value "3628800". The prompt "exp11=#" is shown again at the bottom.

```
exp11=# SELECT fact(10);  
NOTICE: The factorial is : 3628800  
fact  
-----  
(1 row)  
exp11=#
```

Figure 1: Question 1

2. Create a table `student_details`(roll int,marksint, phone int). Create a procedure `pr1` to update all rows in the database. Boost the marks of all students by 5%.

```
CREATE OR REPLACE FUNCTION pr1() AS $$
DECLARE
    c1 CURSOR FOR SELECT * FROM student_details;
    rec RECORD;
BEGIN
    OPEN c1;
    LOOP
        FETCH FROM c1 INTO rec;
        EXIT WHEN NOT FOUND;
        UPDATE student_details
        SET marks = marks * 1.05
        WHERE CURRENT OF c1;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```



The screenshot shows a terminal window with the following content:

```
exp12=# SELECT * FROM student_details;
roll | marks | phone
-----+-----+-----
  1  |   70  | 9496947423
  2  |   85  | 9495941358
  3  |   78  | 8281865009
(3 rows)

exp12=# SELECT pr1();
pr1
-----
(1 row)

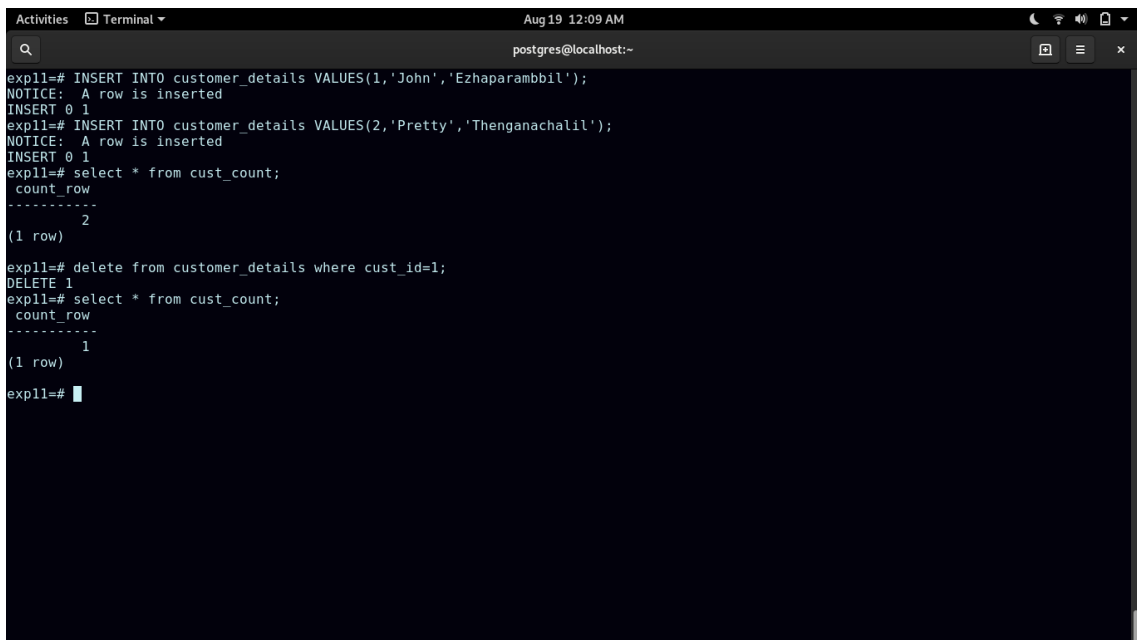
exp12=# SELECT * FROM student_details;
roll | marks | phone
-----+-----+-----
  1  |   74  | 9496947423
  2  |   89  | 9495941358
  3  |   82  | 8281865009
(3 rows)

exp12=#
```

Figure 2: Question 2

3. Create table student (id, name, m1, m2, m3, total, grade). Create a function f1 to calculate grade. Create a procedure p1 to update the total and grade.

```
CREATE OR REPLACE PROCEDURE porf() AS $$
DECLARE
    sum INTEGER;
    c1 CURSOR FOR SELECT * FROM student;
    rec RECORD;
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO rec;
        EXIT WHEN NOT FOUND;
        sum = (rec.m1 + rec.m2 + rec.m3)/3;
        IF sum > 40 THEN
            UPDATE student
            SET total = m1+m2+m3, grade = 'P'
            WHERE CURRENT OF c1;
        ELSE
            UPDATE student
            SET total = m1+m2+m3, grade = 'F'
            WHERE CURRENT OF c1;
        END IF;
    END LOOP;
    CLOSE c1;
END;
$$ LANGUAGE plpgsql;
```



The screenshot shows a terminal window titled "Terminal" with the date and time "Aug 19 12:09 AM". The prompt is "postgres@localhost:~". The user enters several SQL commands: two INSERT statements into "customer\_details", a SELECT statement from "cust\_count", and a DELETE statement from "customer\_details". The output shows successful insertions, the count of rows (2 and then 1), and the deletion of a row.

```
exp11=# INSERT INTO customer_details VALUES(1,'John','Ezhaparambbil');
NOTICE: A row is inserted
INSERT 0 1
exp11=# INSERT INTO customer_details VALUES(2,'Pretty','Thenganachalil');
NOTICE: A row is inserted
INSERT 0 1
exp11=# select * from cust_count;
 count_row
-----
        2
(1 row)

exp11=# delete from customer_details where cust_id=1;
DELETE 1
exp11=# select * from cust_count;
 count_row
-----
        1
(1 row)

exp11=#
```

Figure 3: Question 3

4. 4.Create a package pk1 consisting of the following functions and procedures  
proc1, proc2, fn11, fn22.

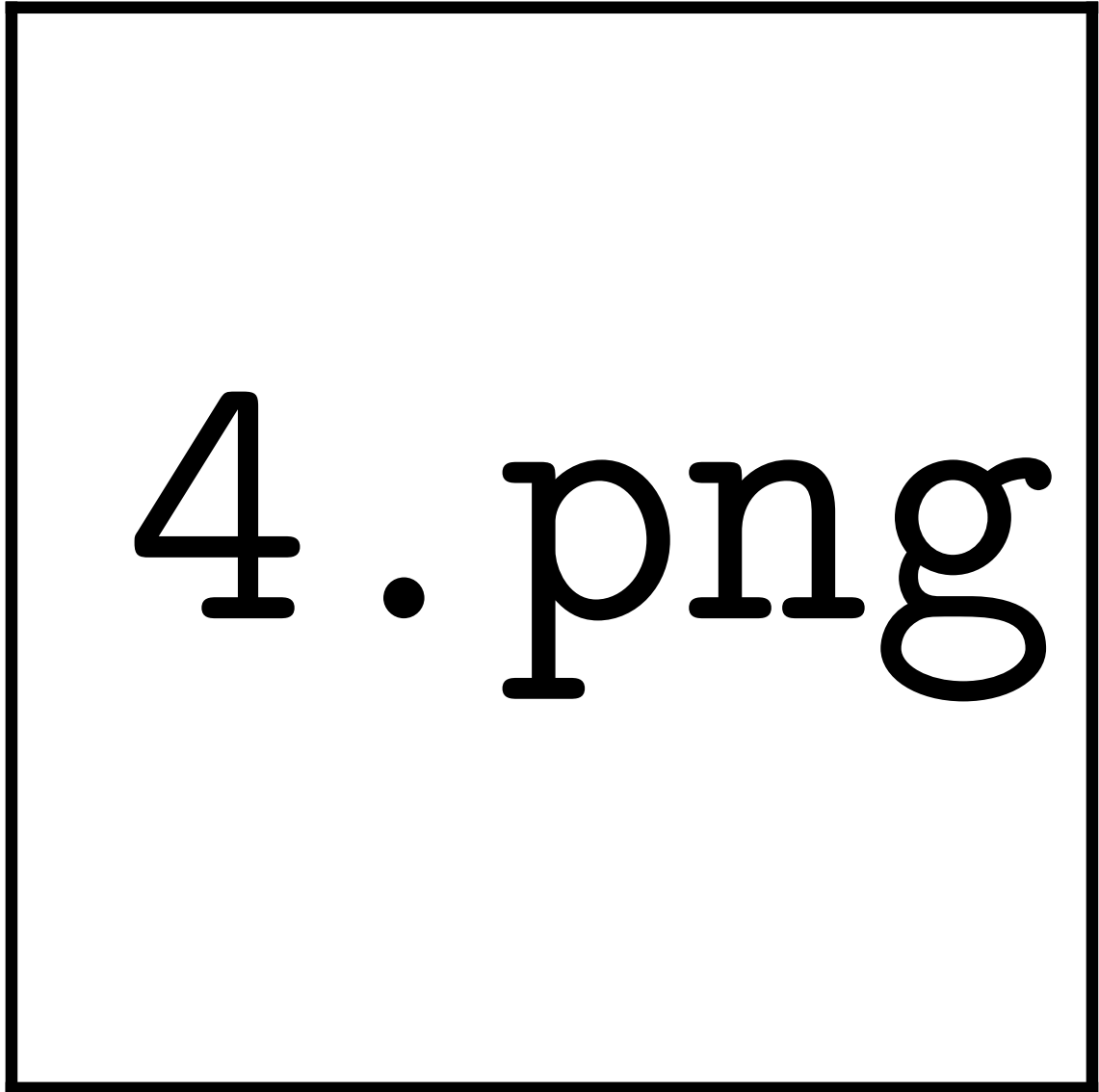


Figure 4: Question 4

### 3 Result

- Successfully implemented Functions, Procedures and Packages in PostgreSQL.