Suman Paudel (33)

Assignment V

# Lab 5:

**Prepare Lab Sheet of MYSQL Statements for following.**

1. Write a stored procedure named “GetEmployee()” to get name, birthdate, address of employees.
2. Execute the procedure in Q.1 and show the result.

**Solution:**

**\*\*\*Note\*\*\***  
The procedures in the PostgreSQL doesn’t return the result set after doing operations; in order to return the result set, functions are used in Postgres. So, I have used functions in this assignment in order to achieve the desired output as per questions. It’s not like that it can’t be achieved using procedures but it’s tedious and uses for loops which will hamper the query time in production systems. I have provided the sample output as well using for Q1 just to show case how can we get result set using procedures.

Use this link for references

<https://stackoverflow.com/questions/58507979/how-to-get-result-set-from-postgresql-stored-procedure>

**SQL Script using Procedures**:

**CREATE** **OR** **REPLACE** **PROCEDURE** GetEmployee()

**LANGUAGE** plpgsql

**AS** **$$**

**DECLARE**

emp\_record RECORD;

**BEGIN**

**FOR** emp\_record **IN** (**SELECT** e.ename, e.bdate, e.address **FROM** Employee e)

**LOOP**

**RAISE** **NOTICE** **'Employee Name: %, Employee Bdate: %, Employee Address: %'**,

emp\_record.ename, emp\_record.bdate,

emp\_record.address;

**END** **LOOP**;

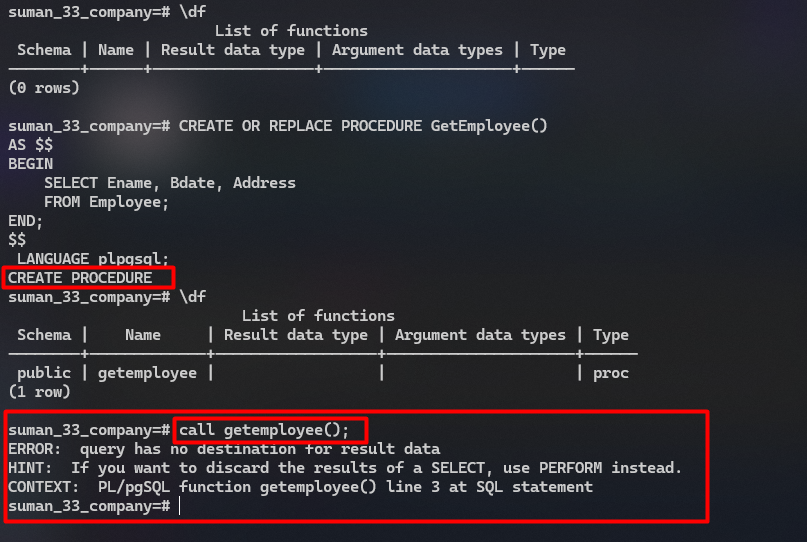
**END**;

**$$**;

**call** GetEmployee()

**Error Output using procedure:**

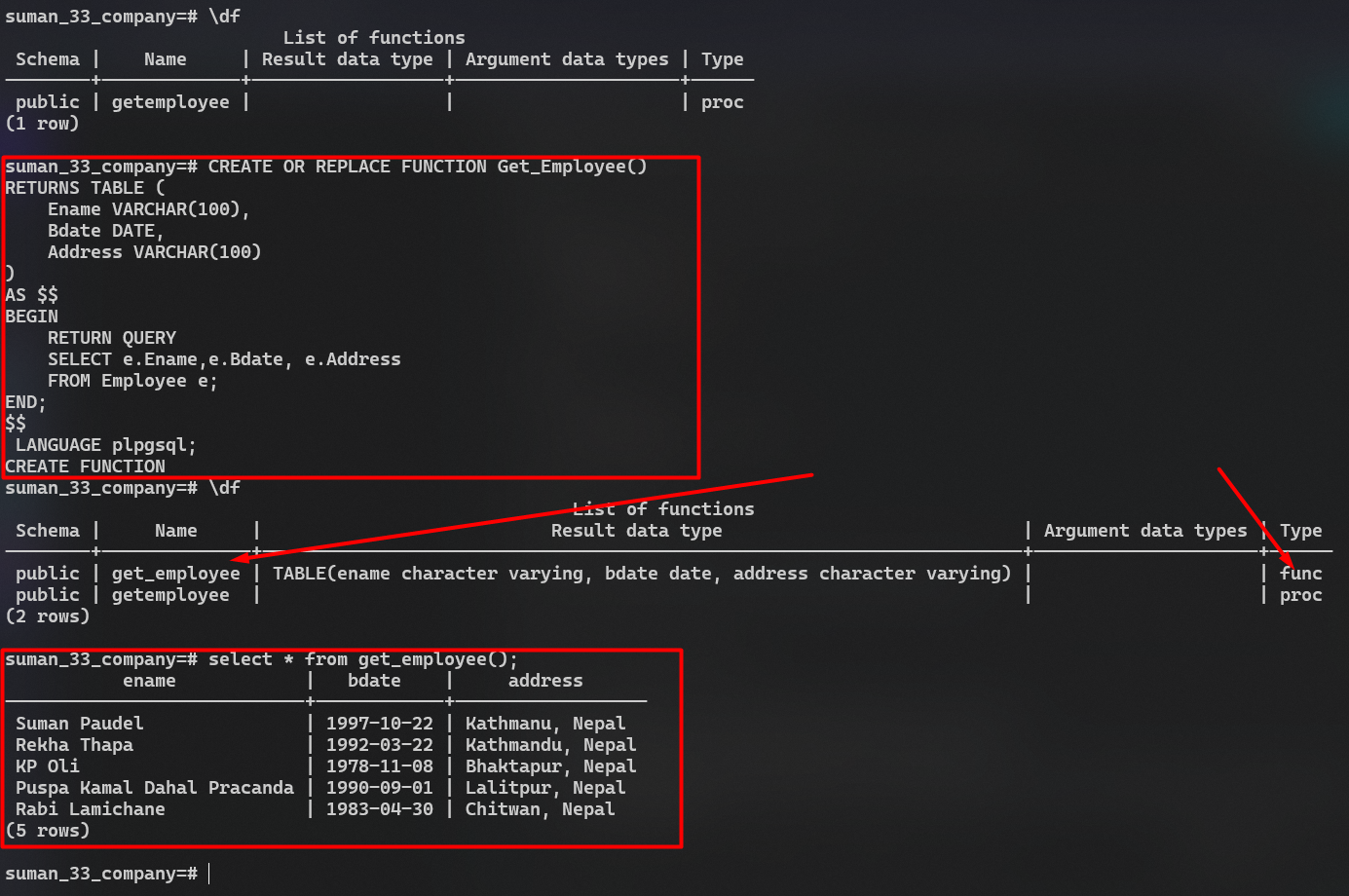
Since PostgreSQL doesn’t have result set to display it will throw error. Though the output can be achieved by working around using loops.

****

**Output of procedure using Loops:**

****

**Final Output using Functions:**

****

1. Write a stored procedure to get PF category name, Amount and start date where the amount is greater than provided input value. Your procedure should contain an IN parameter named amt to take input value of amount. Call the procedure with inputs 1000 and 3000 respectively.

**SQL Script:**

**CREATE** **OR** **REPLACE** **FUNCTION** getPfCategoryName(**IN** amt **NUMERIC**)

**RETURNS** **TABLE** (

pfcategoryname **VARCHAR**(100),

amount **NUMERIC**,

start\_date **DATE**

)

**AS** **$$**

**BEGIN**

**RETURN** QUERY

**select** p.pfcategoryname, p.amount, p.start\_date

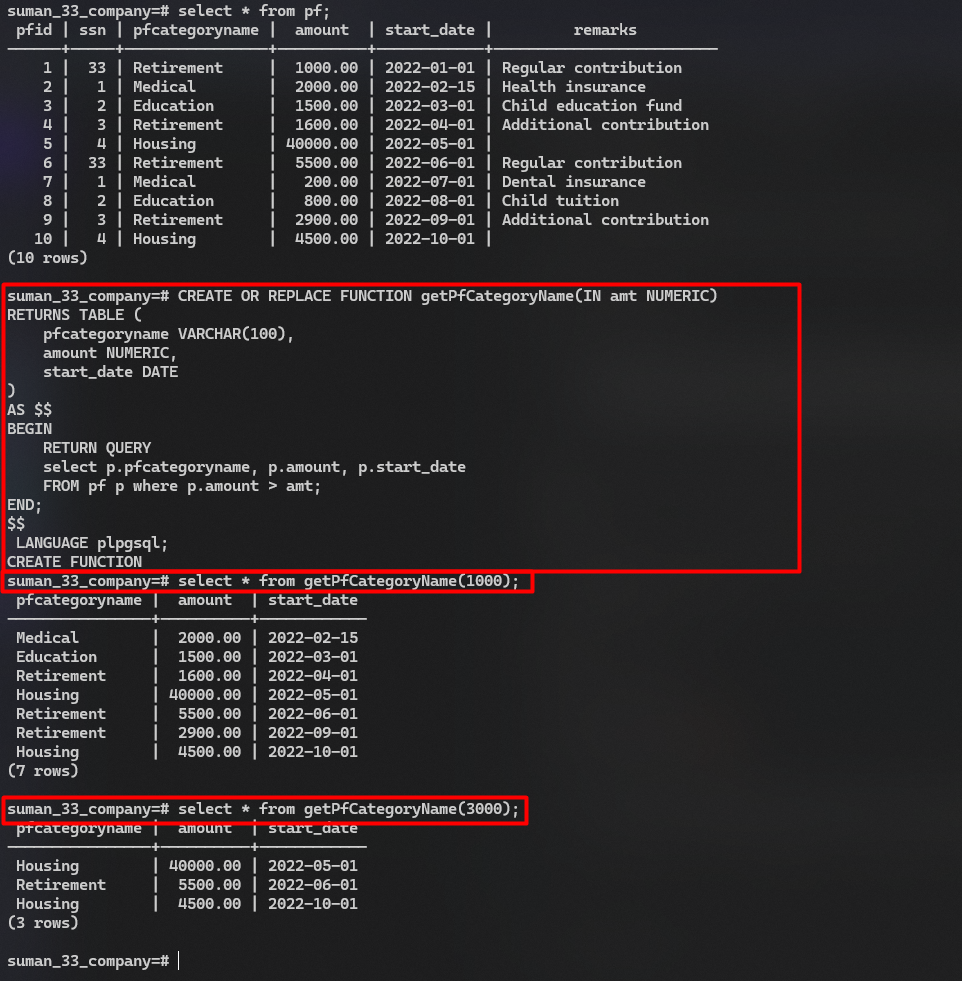
**FROM** pf p **where** p.amount > amt;

**END**;

**$$**

**LANGUAGE** plpgsql;

**Output:**



1. Write a stored procedure to get number of PF records where the amount of PF is equal to the provided input value. Your procedure should contain an IN parameter named amt to take input value of amount and should contain OUT parameter named total to return the total number of PF records satisfying the condition.
2. Call the procedure in Q4. with input of 3000 and print the @total.

**SQL Script:**

**CREATE** **OR** **REPLACE** **FUNCTION** get\_pf\_records(input\_amount **numeric**, **OUT** total **integer**)

**AS** **$$**

**BEGIN**

**SELECT** **COUNT**(\*) **INTO** total

**FROM** pf

**WHERE** amount = input\_amount;

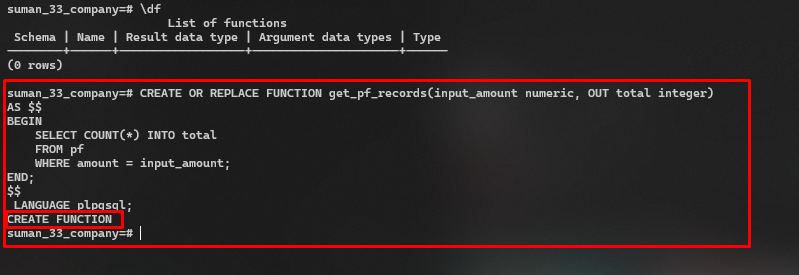
**END**;

**$$**

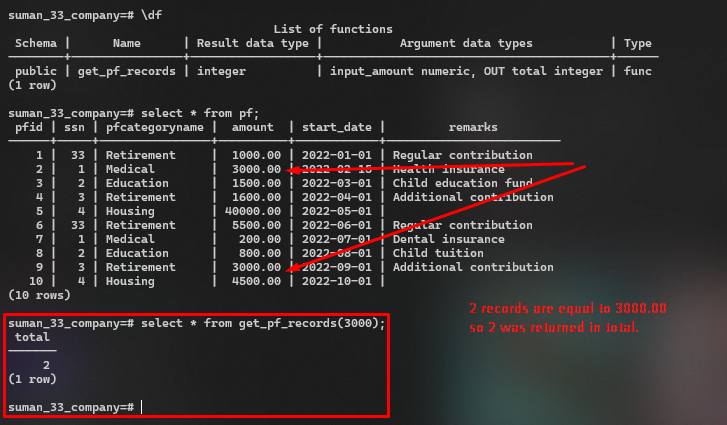
**LANGUAGE** plpgsql;

**select** \* **from** get\_pf\_records(3000);

**Output Q4:**



**Output Q5:**



1. Write before insert trigger before inserting a record into the Employee table. Show some action on the event.

**SQL Script:**

**CREATE** **OR** **REPLACE** **FUNCTION** employee\_before\_insert()

**RETURNS** **TRIGGER** **AS** **$$**

**BEGIN**

**RAISE** **NOTICE**

**'Inserting a new employee with details: SSN: %, Ename: %, Ono: %'**,

**NEW**.SSN, **NEW**.Ename, **NEW**.Ono;

**RETURN** **NEW**;

**END**;

**$$**

**LANGUAGE** plpgsql;

**CREATE** **TRIGGER** employee\_before\_insert\_trigger

**BEFORE** **INSERT** **ON** employee

**FOR** **EACH** **ROW**

**EXECUTE** **PROCEDURE** employee\_before\_insert();

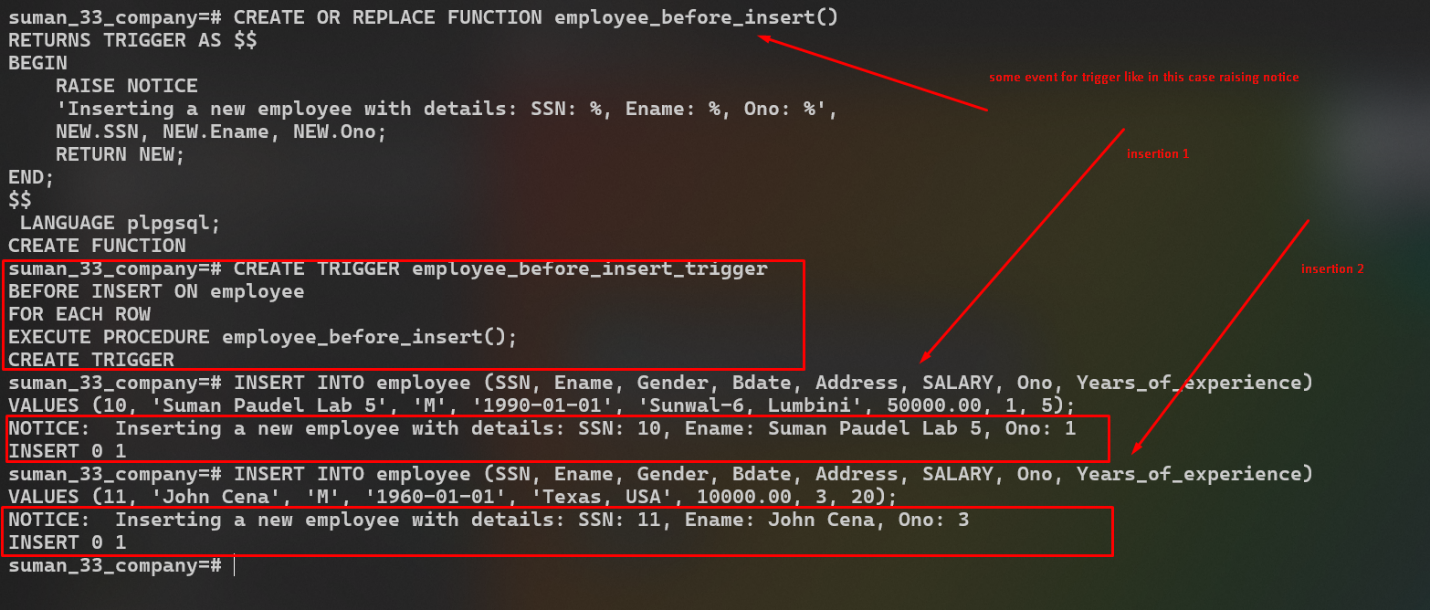
**INSERT** **INTO** ***employee*** (***SSN***, ***Ename***, ***Gender***, ***Bdate***, ***Address***, ***SALARY***, ***Ono***, ***Years\_of\_experience***)

**VALUES** (10, **'Suman Paudel Lab 5'**, **'M'**, **'1990-01-01'**, **'Sunwal-6, Lumbini'**, 50000.00, 1, 5);

**INSERT** **INTO** ***employee*** (***SSN***, ***Ename***, ***Gender***, ***Bdate***, ***Address***, ***SALARY***, ***Ono***, ***Years\_of\_experience***)

**VALUES** (11, **'John Cena'**, **'M'**, **'1960-01-01'**, **'Texas, USA'**, 10000.00, 3, 20);

**Output:**

****

1. Write after delete trigger on PF table during delete operation. Print “It is deleted”.

**SQL Script:**

**CREATE** **OR** **REPLACE** **FUNCTION** pf\_after\_delete()

**RETURNS** **TRIGGER** **AS** **$$**

**BEGIN**

**RAISE** **NOTICE** **'It is deleted'**;

**RETURN** **OLD**;

**END**;

**$$**

**LANGUAGE** plpgsql;

**CREATE** **TRIGGER** pf\_after\_delete\_trigger

**AFTER** **DELETE** **ON** pf

**FOR** **EACH** **ROW**

**EXECUTE** **PROCEDURE** pf\_after\_delete();

**DELETE** **FROM** ***pf*** **WHERE** ***PFID*** = 10;

**Output:**

