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.

```
suman_33_company=# \df
                          List of functions
 Schema | Name | Result data type | Argument data types | Type
--------+------+-------------------+----------------------+------
(0 rows)

suman_33_company=# CREATE OR REPLACE PROCEDURE GetEmployee()
AS $$
BEGIN
    SELECT Ename, Bdate, Address
    FROM Employee;
END;
$$
 LANGUAGE plpgsql;
CREATE PROCEDURE
suman_33_company=# \df
                              List of functions
 Schema |    Name      | Result data type | Argument data types | Type
--------+--------------+-------------------+----------------------+------
 public | getemployee |                   |                      | proc
(1 row)

suman_33_company=# call getemployee();
ERROR:  query has no destination for result data
HINT:  If you want to discard the results of a SELECT, use PERFORM instead.
CONTEXT:  PL/pgSQL function getemployee() line 3 at SQL statement
suman_33_company=#
```

Output of procedure using Loops:

```
suman_33_company=# \df
                          List of functions
 Schema | Name | Result data type | Argument data types | Type
--------+------+-------------------+----------------------+------
(0 rows)

suman_33_company=# CREATE OR REPLACE PROCEDURE GetEmployee()
LANGUAGE plpgsql
AS $$
DECLARE
    emp_record RECORD;
BEGIN
    FOR emp_record IN (SELECT e.ename, e.bdate, e.address FROM Employee e)
    LOOP
        RAISE NOTICE 'Employee Name: %, Employee Bdate: %, Employee Address: %',
                     emp_record.ename, emp_record.bdate,
                     emp_record.address;
    END LOOP;
END;
$$;
CREATE PROCEDURE
suman_33_company=# \df
                              List of functions
 Schema |    Name      | Result data type | Argument data types | Type
--------+--------------+-------------------+----------------------+------
 public | getemployee |                   |                      | proc
(1 row)

suman_33_company=# call getemployee ();
NOTICE:  Employee Name: Suman Paudel, Employee Bdate: 1997-10-22, Employee Address: Kathmanu, Nepal
NOTICE:  Employee Name: Rekha Thapa, Employee Bdate: 1992-03-22, Employee Address: Kathmandu, Nepal
NOTICE:  Employee Name: KP Oli, Employee Bdate: 1978-11-08, Employee Address: Bhaktapur, Nepal
NOTICE:  Employee Name: Puspa Kamal Dahal Pracanda, Employee Bdate: 1990-09-01, Employee Address: Lalitpur, Nepal
NOTICE:  Employee Name: Rabi Lamichane, Employee Bdate: 1983-04-30, Employee Address: Chitwan, Nepal
CALL
suman_33_company=#
```

Final Output using Functions:

```
suman_33_company=# \df
                                List of functions
 Schema |     Name      | Result data type | Argument data types | Type
--------+---------------+------------------+---------------------+------
 public | getemployee   |                  |                     | proc
(1 row)

suman_33_company=# CREATE OR REPLACE FUNCTION Get_Employee()
RETURNS TABLE (
    Ename VARCHAR(100),
    Bdate DATE,
    Address VARCHAR(100)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT e.Ename,e.Bdate, e.Address
    FROM Employee e;
END;
$$
 LANGUAGE plpgsql;
CREATE FUNCTION
suman_33_company=# \df
                                                    List of functions
 Schema |     Name      |                          Result data type                          | Argument data types | Type
--------+---------------+--------------------------------------------------------------------+---------------------+------
 public | get_employee  | TABLE(ename character varying, bdate date, address character varying) |                     | func
 public | getemployee   |                                                                    |                     | proc
(2 rows)

suman_33_company=# select * from get_employee();
           ename            |   bdate    |     address
----------------------------+------------+-----------------
 Suman Paudel               | 1997-10-22 | Kathmanu, Nepal
 Rekha Thapa                | 1992-03-22 | Kathmandu, Nepal
 KP Oli                     | 1978-11-08 | Bhaktapur, Nepal
 Puspa Kamal Dahal Pracanda | 1990-09-01 | Lalitpur, Nepal
 Rabi Lamichane             | 1983-04-30 | Chitwan, Nepal
(5 rows)

suman_33_company=# |
```

3. 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:

```
suman_33_company=# select * from pf;
 pfid | ssn | pfcategoryname |  amount  | start_date |         remarks
------+-----+----------------+----------+------------+-----------------------
    1 |  33 | Retirement     |  1000.00 | 2022-01-01 | Regular contribution
    2 |   1 | Medical        |  2000.00 | 2022-02-15 | Health insurance
    3 |   2 | Education      |  1500.00 | 2022-03-01 | Child education fund
    4 |   3 | Retirement     |  1600.00 | 2022-04-01 | Additional contribution
    5 |   4 | Housing        | 40000.00 | 2022-05-01 |
    6 |  33 | Retirement     |  5500.00 | 2022-06-01 | Regular contribution
    7 |   1 | Medical        |   200.00 | 2022-07-01 | Dental insurance
    8 |   2 | Education      |   800.00 | 2022-08-01 | Child tuition
    9 |   3 | Retirement     |  2900.00 | 2022-09-01 | Additional contribution
   10 |   4 | Housing        |  4500.00 | 2022-10-01 |
(10 rows)

suman_33_company=# 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;
CREATE FUNCTION
suman_33_company=# select * from getPfCategoryName(1000);
 pfcategoryname |  amount  | start_date
----------------+----------+------------
 Medical        |  2000.00 | 2022-02-15
 Education      |  1500.00 | 2022-03-01
 Retirement     |  1600.00 | 2022-04-01
 Housing        | 40000.00 | 2022-05-01
 Retirement     |  5500.00 | 2022-06-01
 Retirement     |  2900.00 | 2022-09-01
 Housing        |  4500.00 | 2022-10-01
(7 rows)

suman_33_company=# select * from getPfCategoryName(3000);
 pfcategoryname |  amount  | start_date
----------------+----------+------------
 Housing        | 40000.00 | 2022-05-01
 Retirement     |  5500.00 | 2022-06-01
 Housing        |  4500.00 | 2022-10-01
(3 rows)

suman_33_company=#
```

4. 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.
5. 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:

```
suman_33_company=# \df
                        List of functions
 Schema | Name | Result data type | Argument data types | Type
--------+------+------------------+---------------------+------
(0 rows)

suman_33_company=# 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;
CREATE FUNCTION
suman_33_company=# |
```

Output Q5:

```
suman_33_company=# \df
                                    List of functions
 Schema |      Name      | Result data type |           Argument data types           | Type
--------+----------------+------------------+-----------------------------------------+------
 public | get_pf_records | integer          | input_amount numeric, OUT total integer | func
(1 row)

suman_33_company=# select * from pf;
 pfid | ssn | pfcategoryname |  amount  | start_date |          remarks
------+-----+----------------+----------+------------+-------------------------
    1 |  33 | Retirement     |  1000.00 | 2022-01-01 | Regular contribution
    2 |   1 | Medical        |  3000.00 | 2022-02-15 | Health insurance
    3 |   2 | Education      |  1500.00 | 2022-03-01 | Child education fund
    4 |   3 | Retirement     |  1600.00 | 2022-04-01 | Additional contribution
    5 |   4 | Housing        | 40000.00 | 2022-05-01 |
    6 |  33 | Retirement     |  5500.00 | 2022-06-01 | Regular contribution
    7 |   1 | Medical        |   200.00 | 2022-07-01 | Dental insurance
    8 |   2 | Education      |   800.00 | 2022-08-01 | Child tuition
    9 |   3 | Retirement     |  3000.00 | 2022-09-01 | Additional contribution
   10 |   4 | Housing        |  4500.00 | 2022-10-01 |
(10 rows)

suman_33_company=# select * from get_pf_records(3000);
 total
-------
     2
(1 row)

suman_33_company=# |
```

2 records are equal to 3000.00
so 2 was returned in total.

6. 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:

```
suman_33_company=# 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 FUNCTION
suman_33_company=# CREATE TRIGGER employee_before_insert_trigger
BEFORE INSERT ON employee
FOR EACH ROW
EXECUTE PROCEDURE employee_before_insert();
CREATE TRIGGER
suman_33_company=# 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);
NOTICE:  Inserting a new employee with details: SSN: 10, Ename: Suman Paudel Lab 5, Ono: 1
INSERT 0 1
suman_33_company=# 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);
NOTICE:  Inserting a new employee with details: SSN: 11, Ename: John Cena, Ono: 3
INSERT 0 1
suman_33_company=#
```

some event for trigger like in this case raising notice

insertion 1

insertion 2

7. 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:

```
suman_33_company=# CREATE OR REPLACE FUNCTION pf_after_delete()
RETURNS TRIGGER AS $$
BEGIN
    RAISE NOTICE 'It is deleted';
    RETURN OLD;
END;
$$
 LANGUAGE plpgsql;
CREATE FUNCTION
suman_33_company=# CREATE TRIGGER pf_after_delete_trigger
AFTER DELETE ON pf
FOR EACH ROW
EXECUTE PROCEDURE pf_after_delete();
CREATE TRIGGER
suman_33_company=# DELETE FROM pf WHERE PFID = 10;
NOTICE:  It is deleted
DELETE 1
suman_33_company=#
```

event after delete