College of Engineering, Trivandrum

Department of Computer Science and Engineering

CS333 APPLICATION SOFTWARE DEVELOPMENT LAB

LABORATORY REPORT 11

Trigger and Exception Handling

Student Name	Student ID
1. Justine Biju(S5)	170445(Roll No:37)

Submission Date : 19/08/2019

1 Introduction

1. Trigger
   A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.
   The syntax is as follows:

   ```
   create trigger [trigger_name]
   [before | after]
   {insert | update | delete}
   on [table_name]
   [for each row]
   [trigger_body]
   ```

2. Exception
   An exception is an error which disrupts the normal flow of program instructions. PL/SQL provides us the exception block which raises the exception thus helping the programmer to find out the fault and resolve it.
   The syntax is as follows:

   ```
   DECLARE
   declarations section;

   BEGIN
   executable command(s);

   EXCEPTION
   WHEN exception1 THEN
   statement1;
   WHEN exception2 THEN
   statement2;
   [WHEN others THEN]
   /* default exception handling code */

   END;
   ```

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

2 Questions

1. Create a trigger whenever a new record is inserted in the customer_details table.

```
--function

CREATE OR REPLACE FUNCTION show_ins() RETURNS TRIGGER AS $$
BEGIN
    RAISE NOTICE 'A row is inserted';
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

--trigger

CREATE TRIGGER t1
AFTER INSERT
ON customer_details
FOR EACH ROW
EXECUTE PROCEDURE show_ins();
```

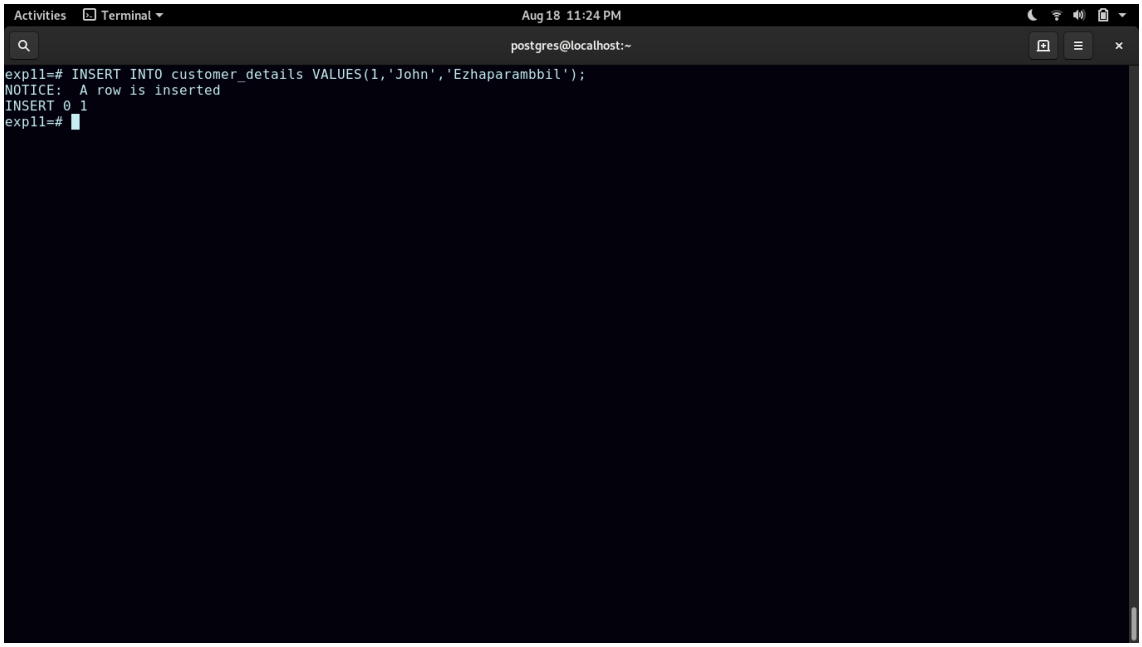

Figure 1: Question 1

2. Create a trigger to display a message when a user enters a value ¿ 20000 in the salary.

```
--function

CREATE OR REPLACE FUNCTION sal_check() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.salary > 20000 THEN
        RAISE NOTICE 'Employee has salary greater than 20000/-';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

--trigger

CREATE TRIGGER t2
AFTER INSERT
ON emp_details
FOR EACH ROW
EXECUTE PROCEDURE sal_check();
```

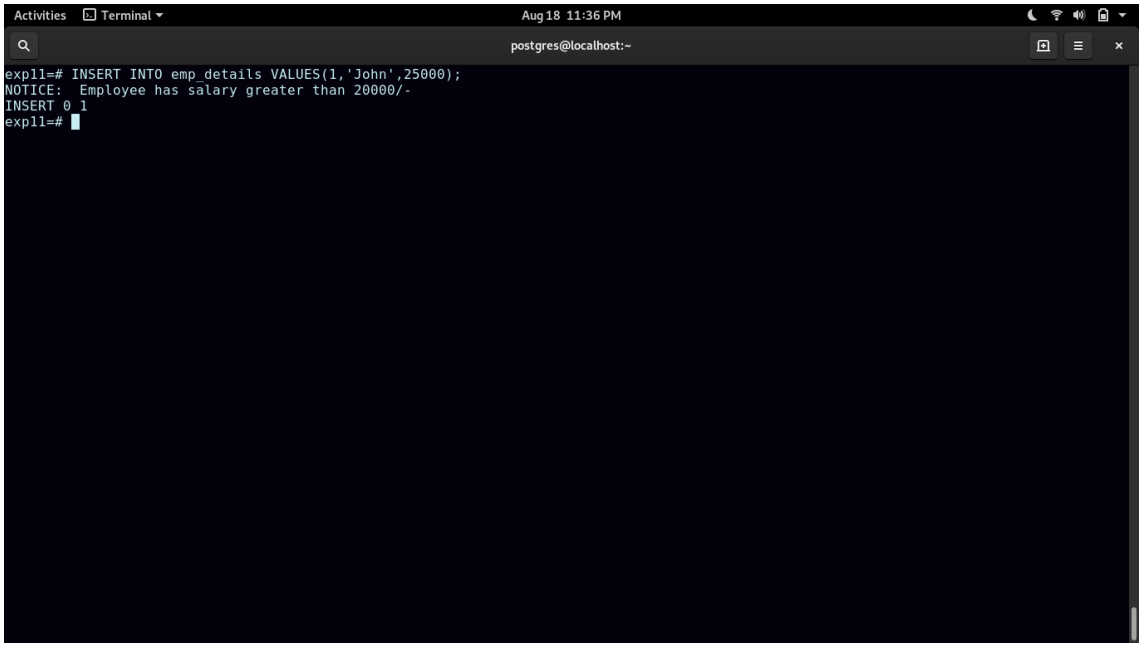

Figure 2: Question 2

3. Create a trigger w.r.tcustomer_detailstable.Increment the value of count_row (in cust_count table) whenever a new tuple is inserted and decrement the value of count_row when a tuple is deleted. Initial value of the count_row is set to 0.

```
--function

CREATE OR REPLACE FUNCTION inc_count() RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE cust_count
        SET count_row = count_row+1;
    ELSE
        UPDATE cust_count
        SET count_row = count_row-1;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

--trigger

CREATE TRIGGER t3
AFTER INSERT OR DELETE
ON customer_details
FOR EACH ROW
EXECUTE PROCEDURE inc_count();
```

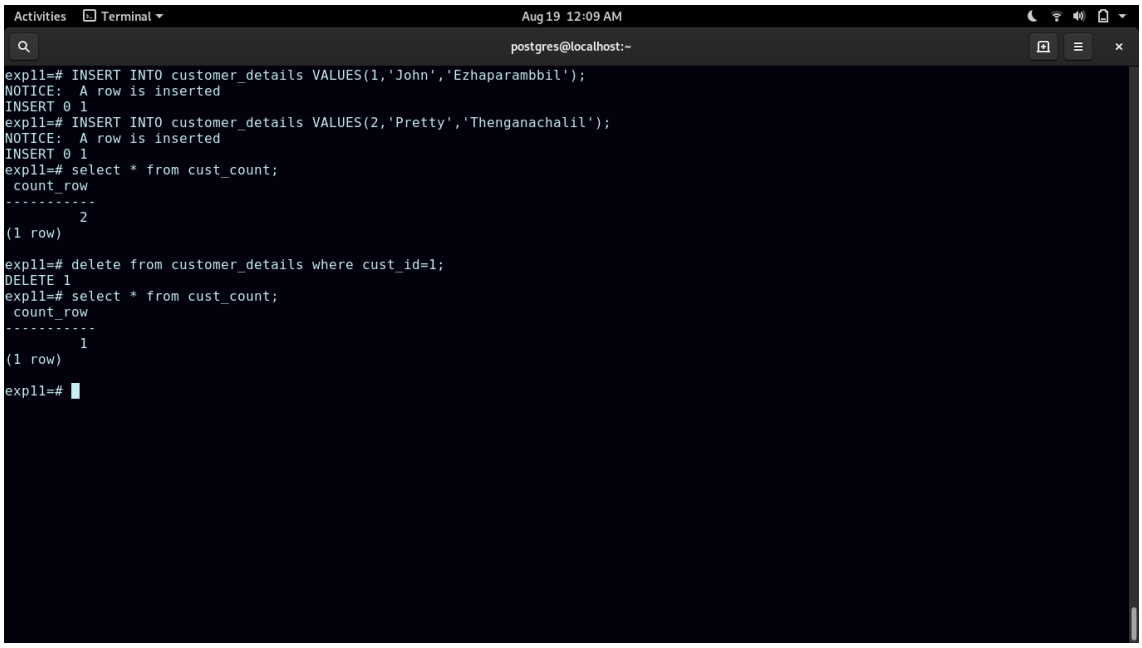

Figure 3: Question 3

4. Create a trigger to insert the deleted rows from emp_details to another table and updated rows to another table. (Create the tables deleted and updated)

```
--function

CREATE OR REPLACE FUNCTION del_upd() RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'DELETE' THEN
        INSERT INTO del VALUES(old.empid, old.empname, old.salary);
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO upd VALUES(new.empid, new.empname, new.salary);
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

--trigger

CREATE TRIGGER t4
AFTER UPDATE OR DELETE
ON emp_details
FOR EACH ROW
EXECUTE PROCEDURE del_upd();
```

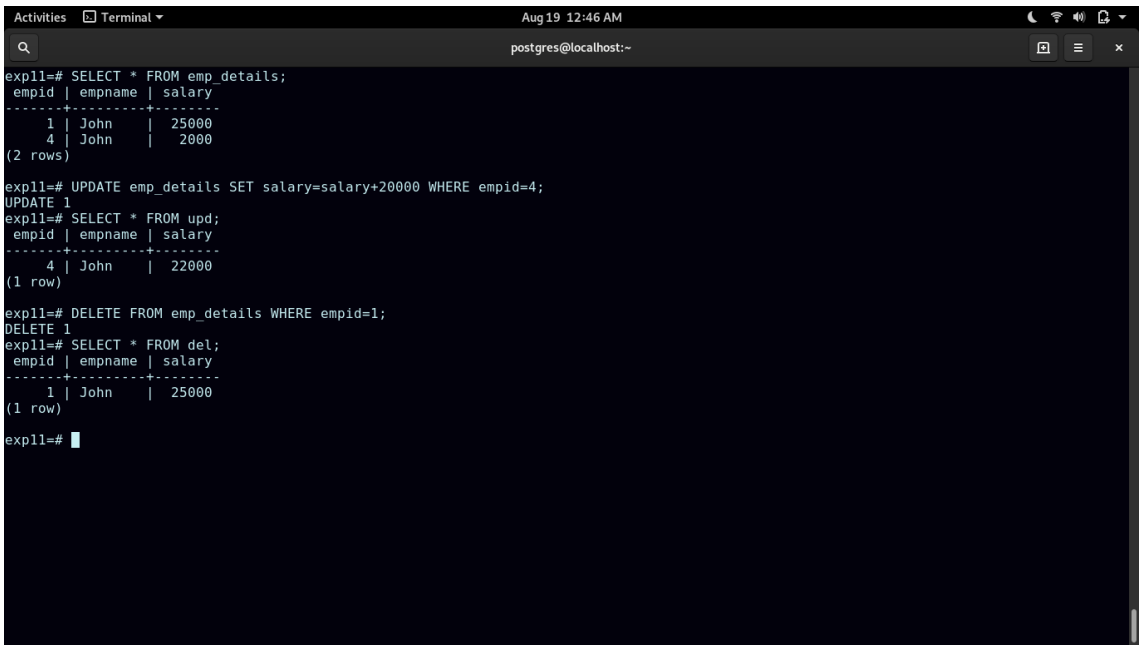

Figure 4: Question 4

5. Write a PL/SQL to show divide by zero exception.

```
CREATE OR REPLACE FUNCTION div(a integer, b integer) RETURNS VOID A
BEGIN
    IF b = 0 THEN
        RAISE EXCEPTION  USING errcode = 22012;
    END IF;
    RAISE NOTICE 'The quotient is = %',a/b;
    EXCEPTION
        WHEN SQLSTATE '22012' THEN
            RAISE NOTICE 'Divide by zero error';
END;
$$ LANGUAGE plpgsql;
```

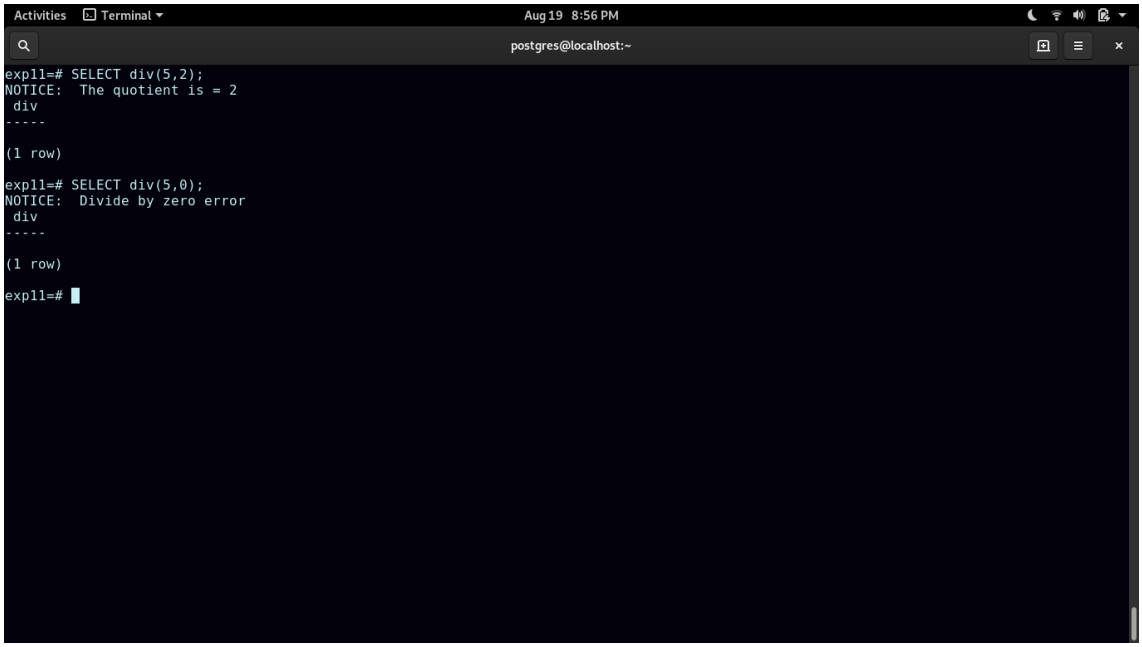

Figure 5: Question 5

6. Write a PL/SQL to show no data found exception.

```
CREATE OR REPLACE FUNCTION ret(a integer) RETURNS VOID AS $$
DECLARE
name VARCHAR(30);
BEGIN
    SELECT empname INTO STRICT name FROM emp_details WHERE empid =
    RAISE NOTICE 'Name is = %',name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE NOTICE 'No data error';
END;
$$ LANGUAGE plpgsql;
```

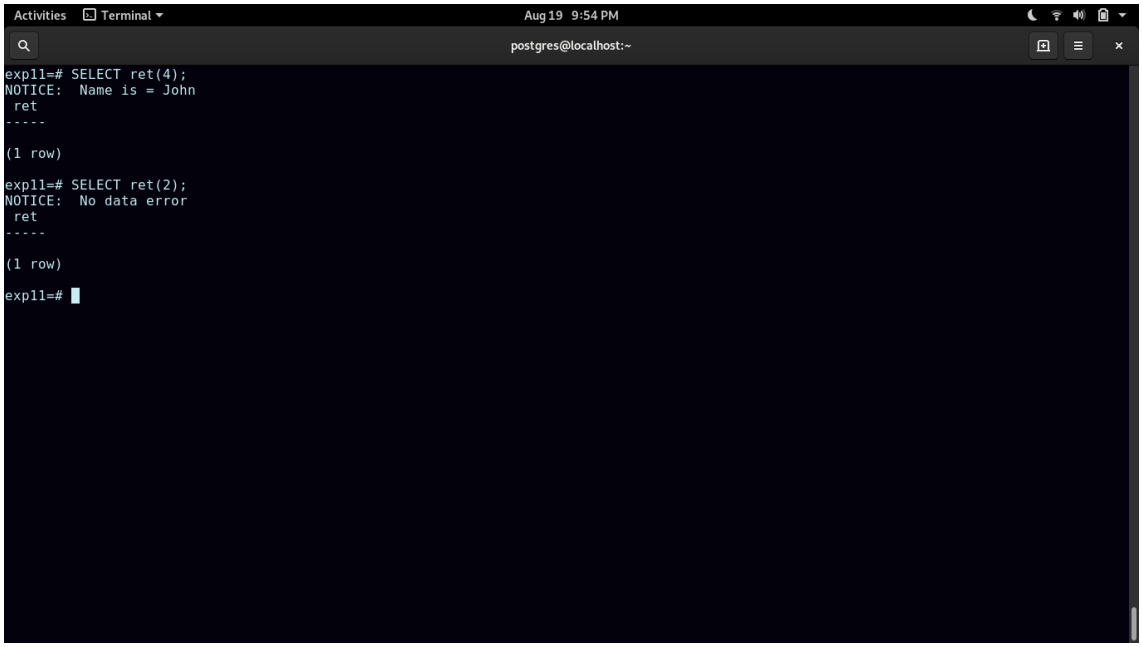

Figure 6: Question 6

7. Create a table with ebill(cname,prevreading,currreading). If prevreading = currreading then raise an exception 'Data Entry Error'.

```
CREATE OR REPLACE FUNCTION add_ebill(name VARCHAR(20), p integer, c
BEGIN
    IF p = c THEN
        RAISE EXCEPTION USING errcode = '50001';
    END IF;
    INSERT INTO ebill VALUES (name,p,c);
    RAISE NOTICE 'Statement processed';
    EXCEPTION
        WHEN SQLSTATE '50001' THEN
            RAISE NOTICE 'Data Entry Error';
END;
$$ LANGUAGE plpgsql;
```

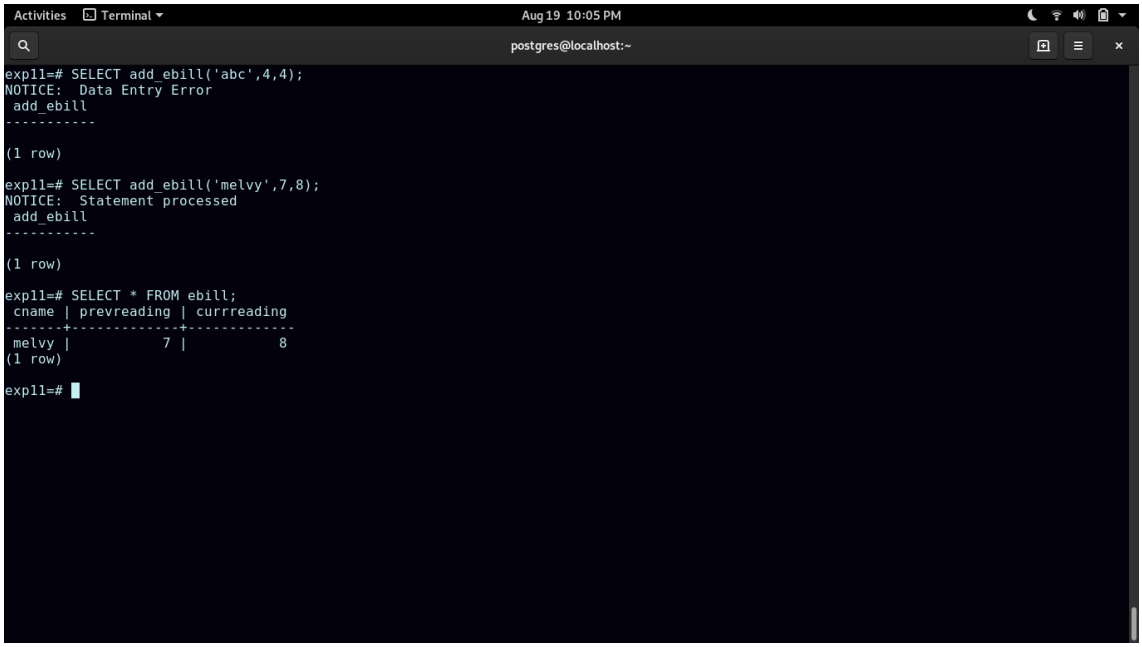

Figure 7: Question 7

3 Result

- Successfully implemented Triggers and Exceptions in PostgreSQL.