Experiment No.

Title: To Implement Triggers.

Batch:A2 **Roll No.:** 16010421063 **Experiment No.:8**

Title: To implement Triggers for given database.

Resources needed: PostgreSQL 9.3

Theory
Pre Lab/ Prior Concepts:
Trigger:
A trigger is a statement that the system executes automatically as a side effect of a modification to the database. Triggers are used to ensure some types of integrity.

To design a trigger mechanism you should:
1. Specify when a trigger is to be executed. This is broken up into an event that causes a trigger to be checked and a condition that must be satisfied for trigger execution to proceed
2. Specify the actions to be taken when the trigger executes.

Generalized Model:
Triggers are based on the Event- condition- Action (ECA) Model. A rule in the ECA model has three components.
1. The event(s) that triggers the rule.
2. The condition that determines whether the rule action should be executed. If no action is specified, the action will be executed once the event occurs.
3. The action to be taken. It could be a sequence of SQL statements, a DB transaction or an external program that will be executed automatically.

When to use Trigger:
In many cases, it is convenient to specify the type of action to be taken when certain event occurs and when certain conditions are satisfied.
It may be useful to specify a condition that, if violated, causes some user to be informed of the violation.
For example:-
A manager may want to be informed, if an employee’s travel expenses exceed a certain limit by receiving a message whenever this occurs.
The condition is thus used to monitor the database.

CREATE TRIGGER statement is used to implement such action in SQL. Consider the triggers for following cases:-
1. Trigger for insertion:-
This trigger executes whenever condition is satisfied, during the insertion statement. If no condition is satisfied, then it executed for every insertion statement, for the relation specified.
Example:-
In DB, we have created a trigger for insertion, on the relation Employee. If salaries <
1500 then print ‘Unsuccessful’ else print ‘Successful’.
2. Trigger for Updation:-
This trigger executes for updating of a column name(s) of a particular relation. A condition may or may not be specified.
Example:-

In our DB, we have created a trigger for updating, on the relation Employee. If salary is updated to unacceptable amount, then print unsuccessful and rollback else print successful.
3. Trigger for Deletion:-
This trigger executes during the deletion statement. If a particular condition is satisfied, the system may allow or not allow the deletion of certain tuples,
Example:-
If SSN from Employee =101, then print 'Unsuccessful' and rollback, else print 'Successful'.

In Postgresql we need to create a function to execute all action statements of trigger and call this function in create trigger statement for certain event.
Example.
In employeee table dnum is a foreign key. So the trigger writen here will increment total_emp of department table by one and total_sal of department by salary of newly inserted employee's salary.

```
CREATE or replace  FUNCTION inse_function() RETURNS trigger As $emp_update$


begin
     UPDATE dept set total_emp=total_emp + 1,
         total_sal=total_sal+ new.salary where dept.dno=new.dno;
    RETURN new;

END
$emp_update$ LANGUAGE plpgsql;



CREATE TRIGGER emp_update AFTER INSERT
       ON emp_table FOR each ROW
       EXECUTE PROCEDURE inse_function();
```

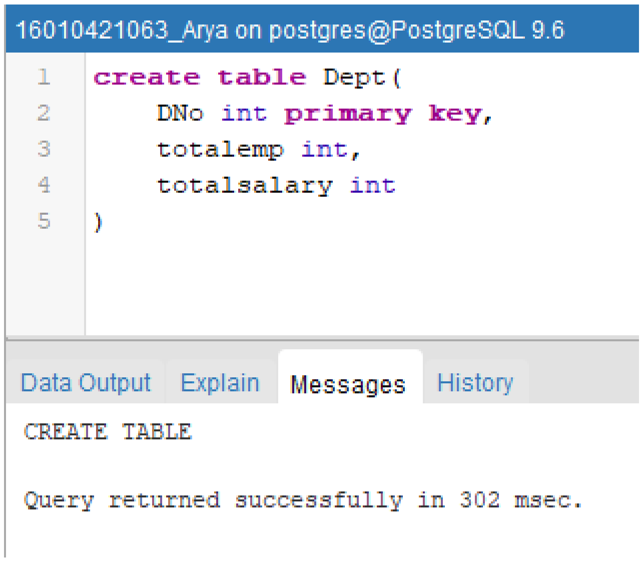
Procedure / Approach /Algorithm / Activity Diagram

1. Create new database for your application
2. Apply required integrity constraints on tables in your database

To design a trigger.

1. Identify events in the database.
2. Specify conditions under which the trigger is to be executed.
3. Specify actions to be taken when trigger is executed.

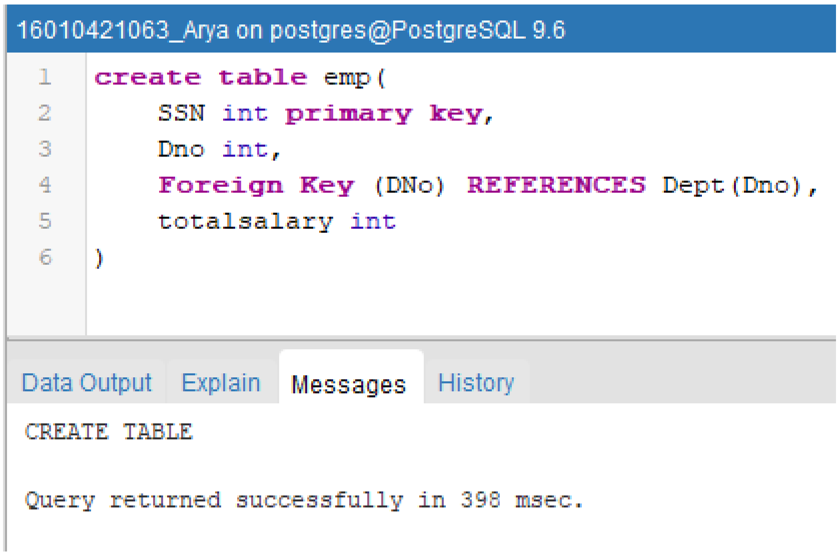
Results: (Program printout with output / Document printout as per the format)


16010421063_Arya on postgres@PostgreSQL 9.6

```sql
create table Dept(
     DNo int primary key,
     totalemp int,
     totalsalary int
)
```

Data Output | Explain | Messages | History

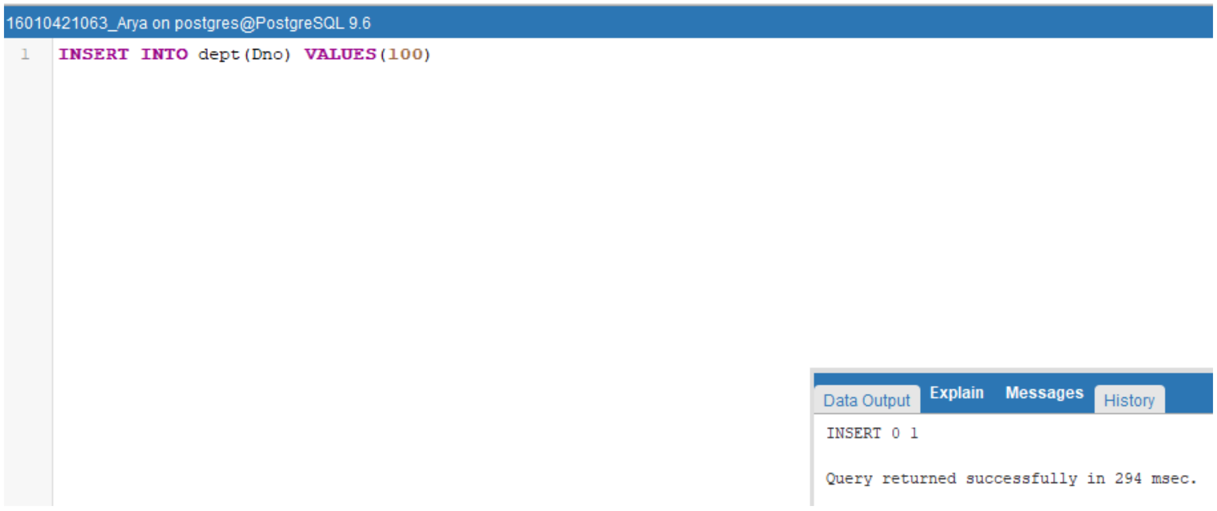
```
CREATE TABLE

Query returned successfully in 302 msec.
```


16010421063_Arya on postgres@PostgreSQL 9.6

```sql
create table emp(
     SSN int primary key,
     Dno int,
     Foreign Key (DNo) REFERENCES Dept(Dno),
     totalsalary int
)
```

Data Output | Explain | Messages | History

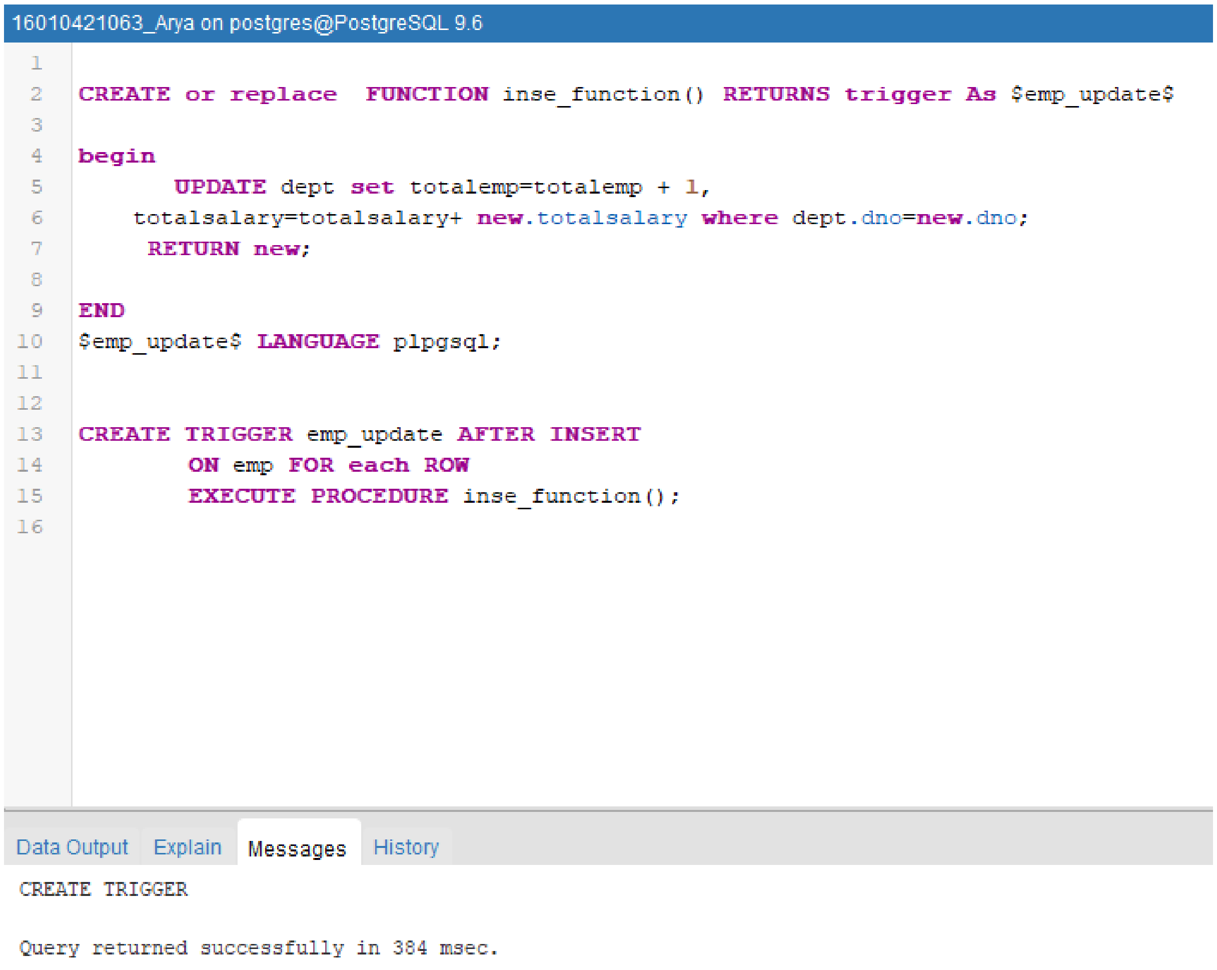
```
CREATE TABLE

Query returned successfully in 398 msec.
```


16010421063_Arya on postgres@PostgreSQL 9.6

```sql
INSERT INTO dept(Dno) VALUES(100)
```

Data Output | Explain | Messages | History

```
INSERT 0 1

Query returned successfully in 294 msec.
```

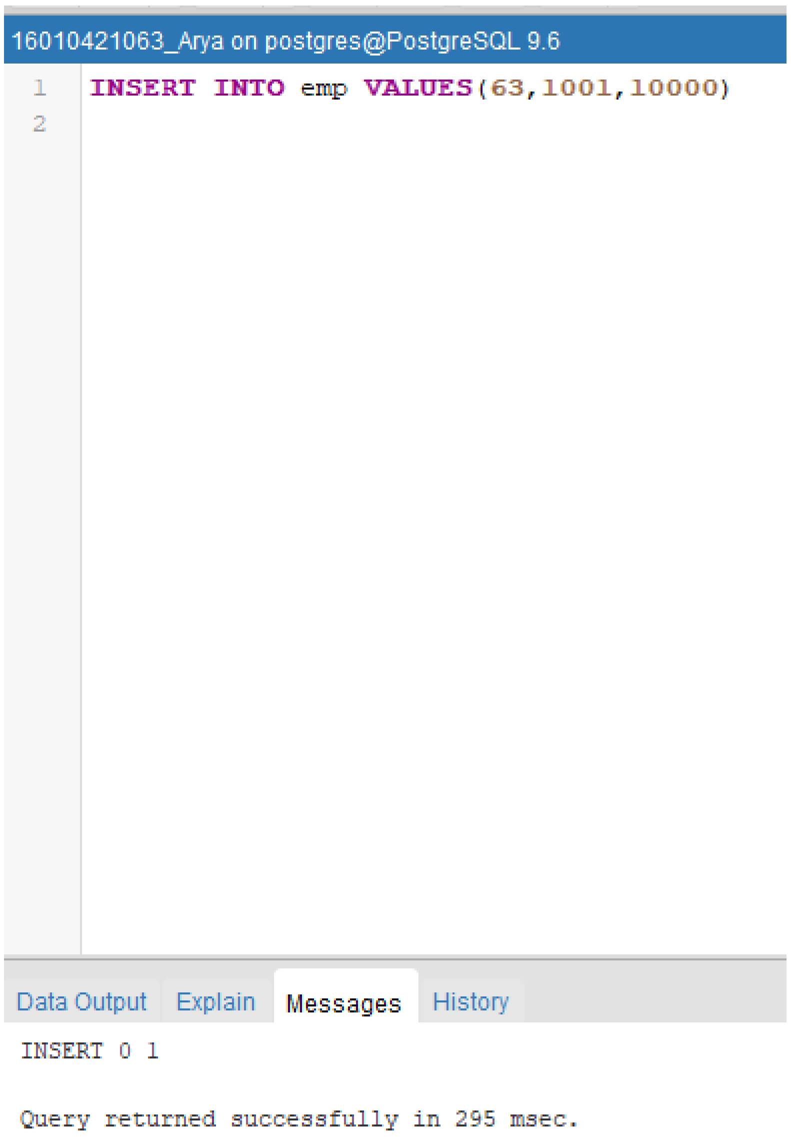


16010421063_Arya on postgres@PostgreSQL 9.6

```

CREATE or replace  FUNCTION inse_function() RETURNS trigger As $emp_update$

begin
       UPDATE dept set totalemp=totalemp + 1,
    totalsalary=totalsalary+ new.totalsalary where dept.dno=new.dno;
     RETURN new;

END
$emp_update$ LANGUAGE plpgsql;


CREATE TRIGGER emp_update AFTER INSERT
        ON emp FOR each ROW
        EXECUTE PROCEDURE inse_function();

```

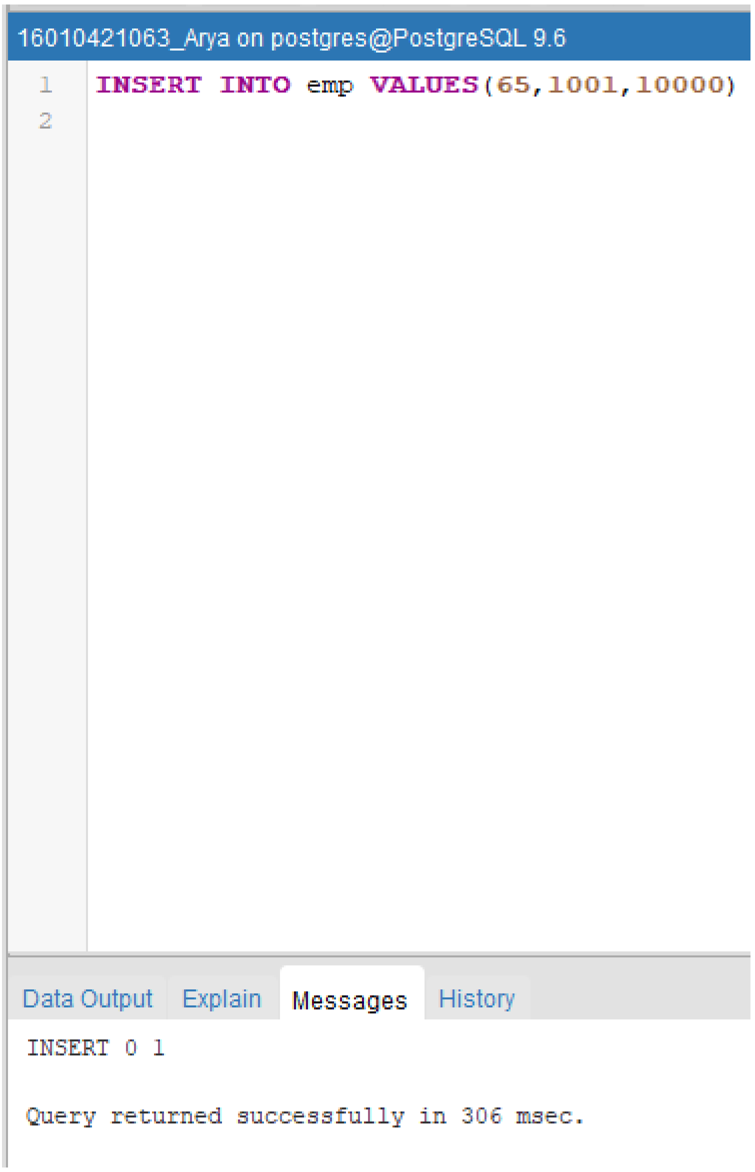
Data Output | Explain | Messages | History

```
CREATE TRIGGER

Query returned successfully in 384 msec.
```



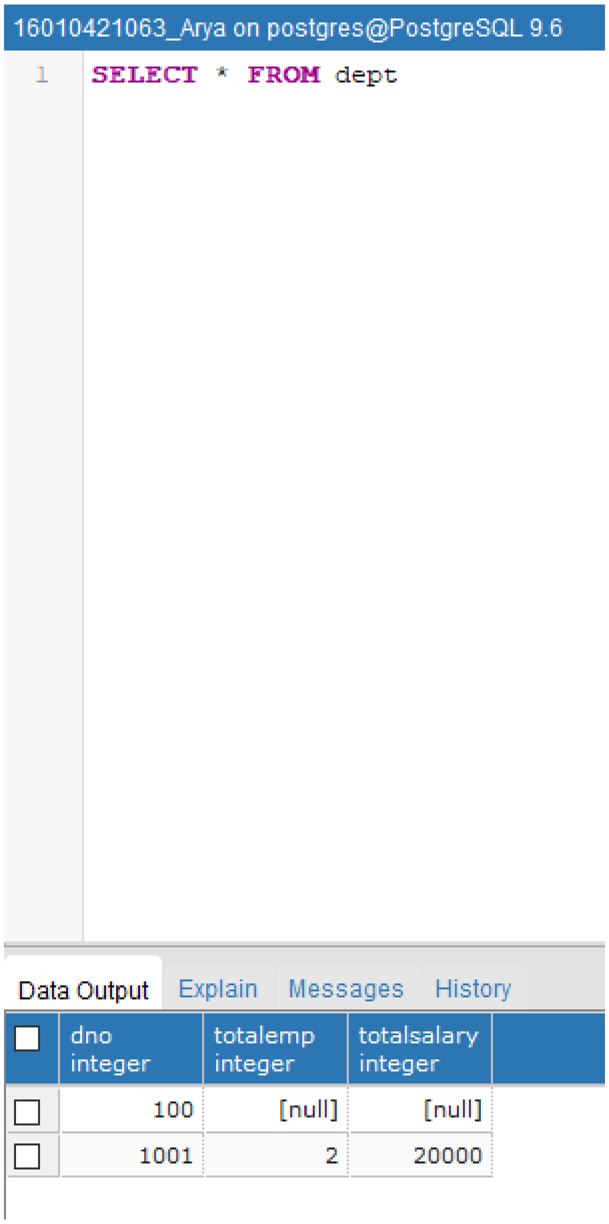
```sql
INSERT INTO emp VALUES(63,1001,10000)
```

```
INSERT 0 1

Query returned successfully in 295 msec.
```



```sql
INSERT INTO emp VALUES(65,1001,10000)
```

```
INSERT 0 1

Query returned successfully in 306 msec.
```



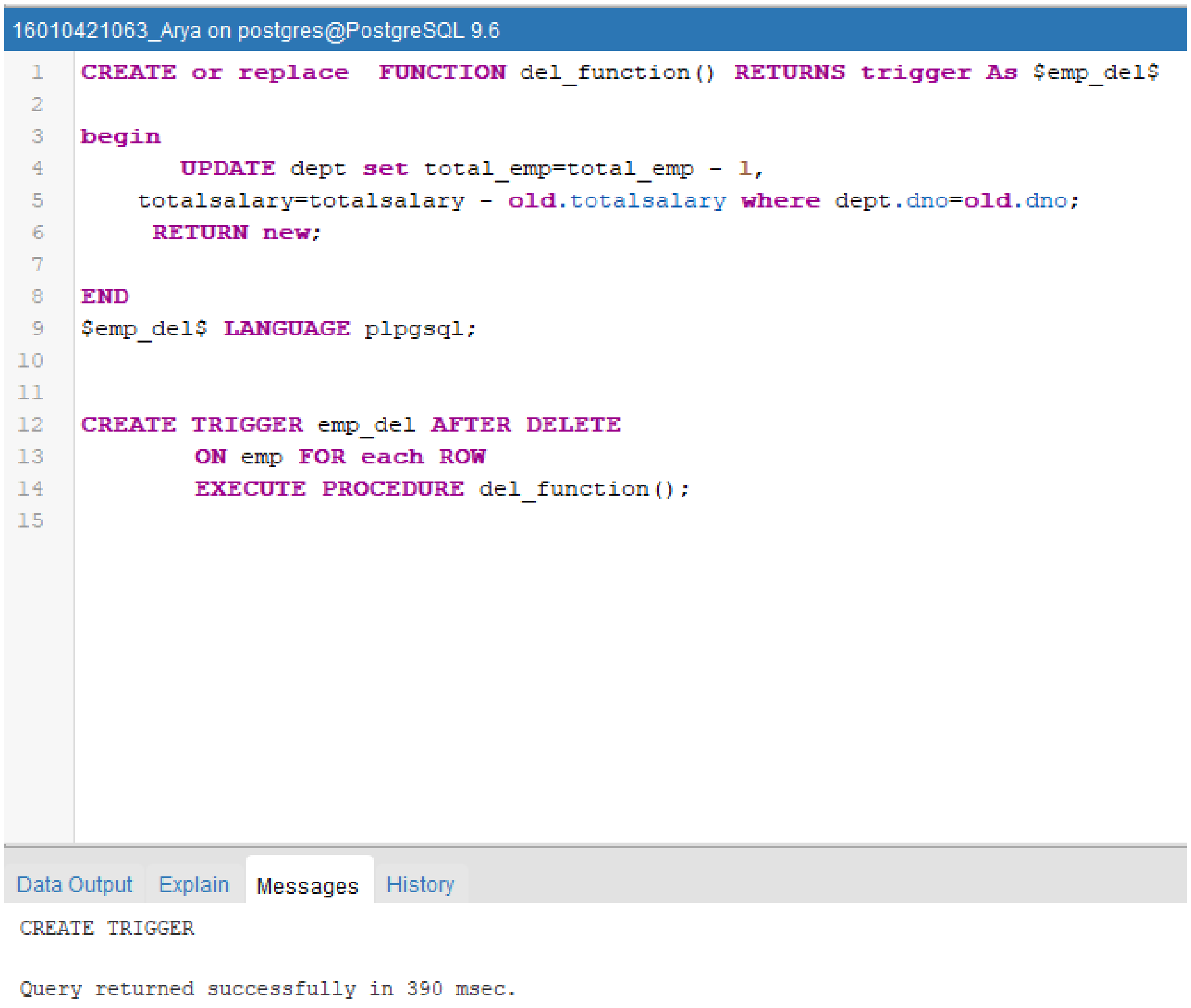
16010421063_Arya on postgres@PostgreSQL 9.6

```
SELECT * FROM dept
```

Data Output | Explain | Messages | History

```
dno      totalemp  totalsalary
integer  integer   integer
    100    [null]       [null]
   1001         2        20000
```

```
CREATE or replace  FUNCTION del_function() RETURNS trigger As $emp_del$

begin
      UPDATE dept set total_emp=total_emp - 1,
          totalsalary=totalsalary - old.totalsalary where dept.dno=old.dno;
    RETURN new;

END
$emp_del$ LANGUAGE plpgsql;


CREATE TRIGGER emp_del AFTER DELETE
       ON emp FOR each ROW
       EXECUTE PROCEDURE del_function();
```

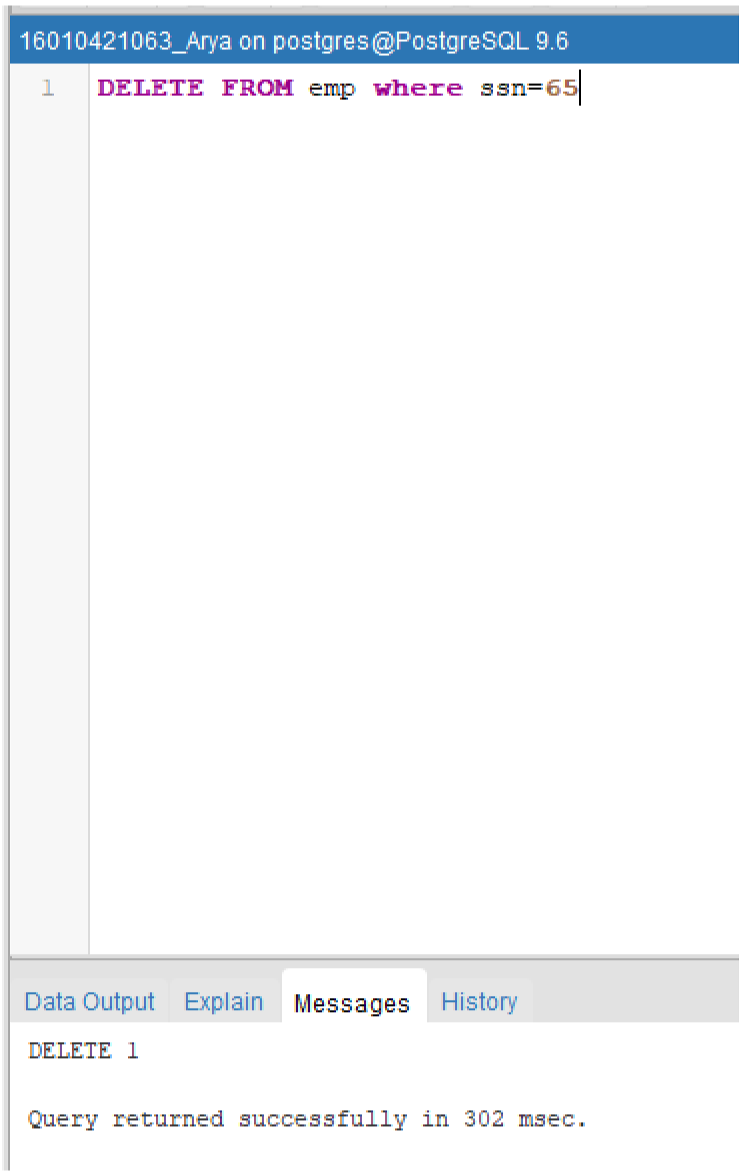


16010421063_Arya on postgres@PostgreSQL 9.6

```
CREATE or replace  FUNCTION del_function() RETURNS trigger As $emp_del$

begin
       UPDATE dept set total_emp=total_emp - 1,
    totalsalary=totalsalary - old.totalsalary where dept.dno=old.dno;
     RETURN new;

END
$emp_del$ LANGUAGE plpgsql;


CREATE TRIGGER emp_del AFTER DELETE
        ON emp FOR each ROW
        EXECUTE PROCEDURE del_function();

```

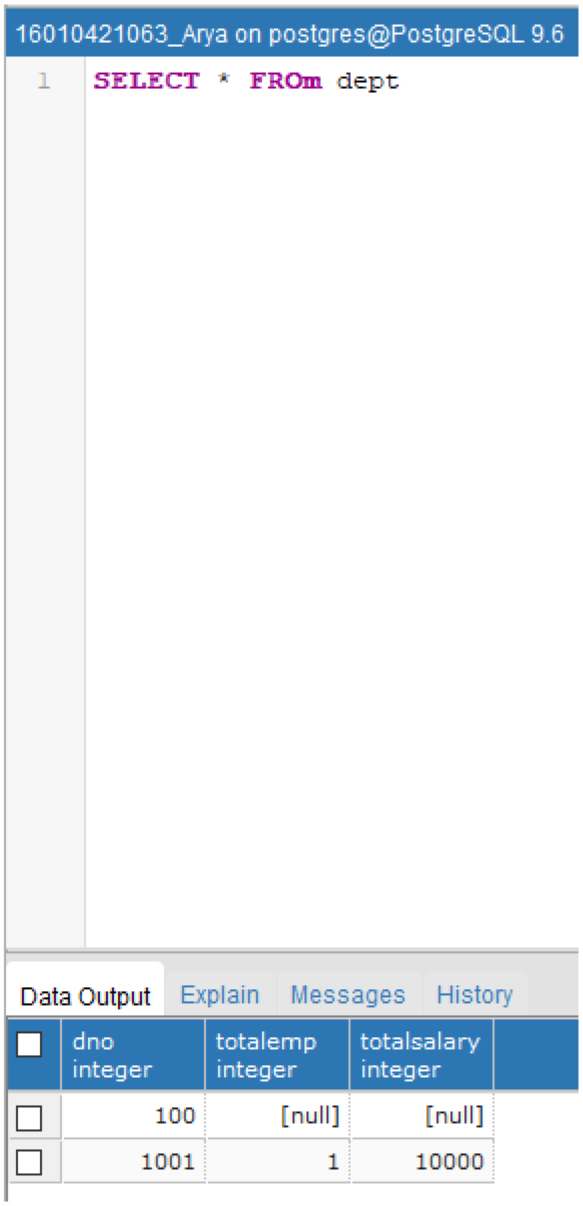
Data Output Explain Messages History

```
CREATE TRIGGER

Query returned successfully in 390 msec.
```



```
DELETE FROM emp where ssn=65
```

```
DELETE 1

Query returned successfully in 302 msec.
```



```
SELECT * FROm dept
```

dno integer	totalemp integer	totalsalary integer
100	[null]	[null]
1001	1	10000

Questions:

1 Explain any one real-time application of trigger

The most common use of trigger is in College timetable management system. As soon as we add a lecture in the timetable of a class it should activate a trigger and add that particular lecture to the student table as well as to the faculty table which would allow them to see how many hours they have to be present.