

# **DBMS LAB WEEK 9-10**

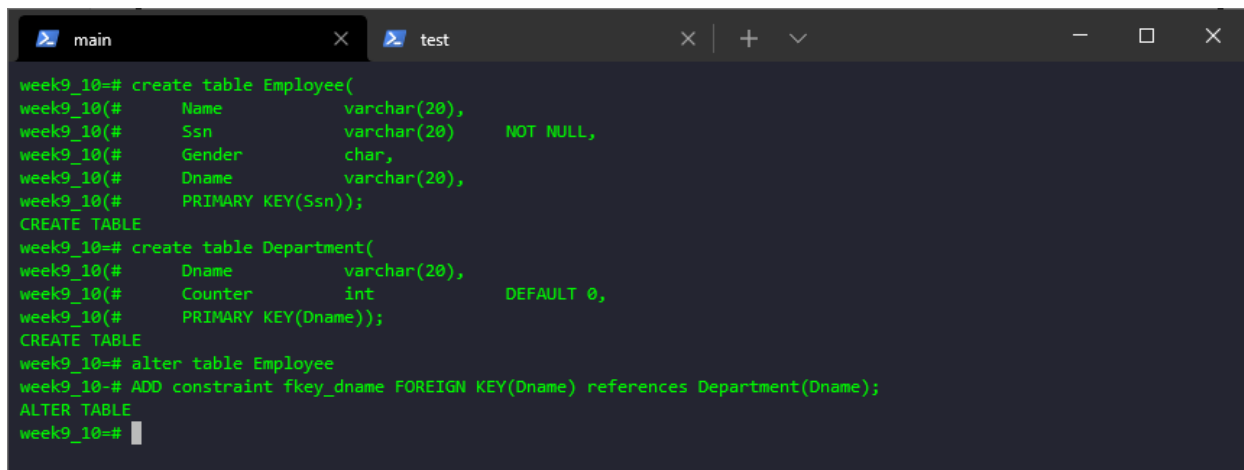
**ATUL ANURAG  
PES2UG19CS075  
SECTION B**

## **SQL – Creating Triggers and Functions**

### **Problem Statement:**

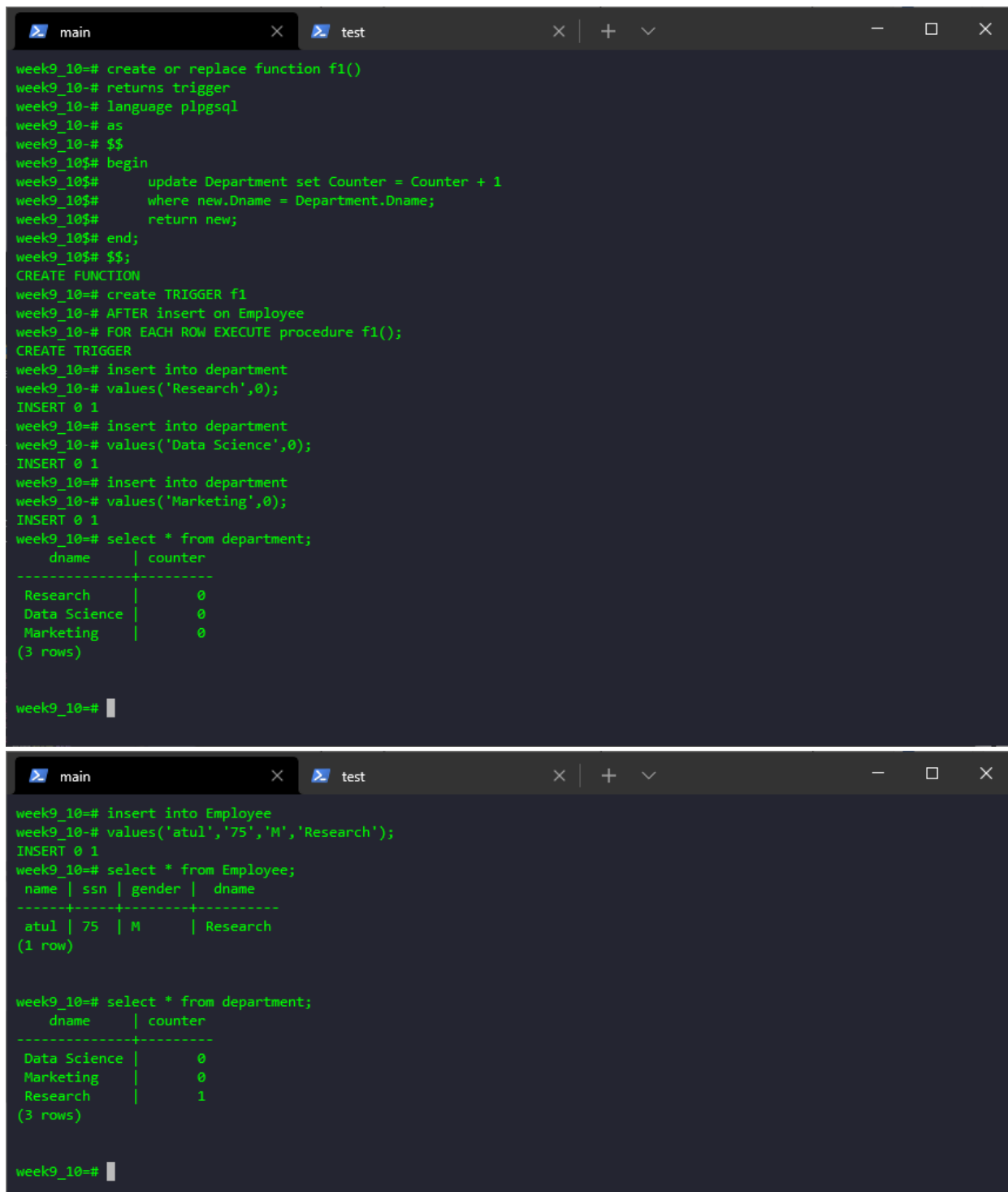
Write the SQL Triggers and functions for the following using Postgres sql.

1. Create an employee table which contains employee details and the department he works for. Create another table department consisting of dname and number of employees. Write triggers to increment or decrement the number of employees in a department table when the record in the employee table is inserted or deleted respectively.



```
main × test × + -
week9_10=# create table Employee(
week9_10(#      Name          varchar(20),
week9_10(#      Ssn           varchar(20)    NOT NULL,
week9_10(#      Gender        char,
week9_10(#      Dname          varchar(20),
week9_10(#      PRIMARY KEY(Ssn));
CREATE TABLE
week9_10=# create table Department(
week9_10(#      Dname          varchar(20),
week9_10(#      Counter        int          DEFAULT 0,
week9_10(#      PRIMARY KEY(Dname));
CREATE TABLE
week9_10=# alter table Employee
week9_10=# ADD constraint fkey_dname FOREIGN KEY(Dname) references Department(Dname);
ALTER TABLE
week9_10=#
```

## Trigger function for insertion:



```
main test
week9_10=# create or replace function f1()
week9_10=# returns trigger
week9_10=# language plpgsql
week9_10=# as
week9_10=# $$
week9_10$# begin
week9_10$#     update Department set Counter = Counter + 1
week9_10$#     where new.Dname = Department.Dname;
week9_10$#     return new;
week9_10$# end;
week9_10$# $$;
CREATE FUNCTION
week9_10=# create TRIGGER f1
week9_10=# AFTER insert on Employee
week9_10=# FOR EACH ROW EXECUTE procedure f1();
CREATE TRIGGER
week9_10=# insert into department
week9_10=# values('Research',0);
INSERT 0 1
week9_10=# insert into department
week9_10=# values('Data Science',0);
INSERT 0 1
week9_10=# insert into department
week9_10=# values('Marketing',0);
INSERT 0 1
week9_10=# select * from department;
  dname | counter
-----+-----
 Research |      0
Data Science |      0
 Marketing |      0
(3 rows)

week9_10=#
```

```
main test
week9_10=# insert into Employee
week9_10=# values('atul','75','M','Research');
INSERT 0 1
week9_10=# select * from Employee;
 name | ssn | gender | dname
-----+-----+-----+-----
  atul | 75  | M      | Research
(1 row)

week9_10=# select * from department;
  dname | counter
-----+-----
Data Science |      0
 Marketing   |      0
 Research    |      1
(3 rows)

week9_10=#
```

## Trigger Function for Deletion:

```
main test
week9_10=# create or replace function f2()
week9_10=# returns trigger
week9_10=# language plpgsql
week9_10=# as
week9_10=# $$
week9_10$# begin
week9_10$#     update Department set Counter = Counter - 1
week9_10$#     where old.Dname = Department.Dname;
week9_10$#     return old;
week9_10$# end;
week9_10$# $$;
CREATE FUNCTION
week9_10=# create TRIGGER f2
week9_10=# BEFORE DELETE on Employee
week9_10=# FOR EACH ROW EXECUTE procedure f2();
ERROR:  trigger "f2" for relation "employee" already exists
week9_10=# delete from employee where Ssn='75';
DELETE 1
week9_10=# select * from Department;
  dname | counter
-----+-----
Data Science |      0
Marketing   |      0
Research    |      0
(3 rows)

week9_10=#
```

2. Create an order\_item table which contains details like name, quantity and unit price of every item purchased. Create an order summary table that contains number of items and total price. Create triggers to update entry in order summary whenever an item is inserted or deleted in the order item table.

```
main test
week9_10=# create table order_item(
week9_10(#      name          varchar(20)          NOT NULL,
week9_10(#      quantity      int,
week9_10(#      price         int,
week9_10(#      id            varchar(10)         NOT NULL,
week9_10(#      PRIMARY KEY(id));
CREATE TABLE
week9_10=# create table summary_table(
week9_10(#      number_items  int,
week9_10(#      total_price   int);
CREATE TABLE
week9_10=#

main test
week9_10=# language plpgsql
week9_10=# as
week9_10=# $$
week9_10$# begin
week9_10$#      update summary_table set number_items = number_items + 1, total_price = total_price + new.quantity * new.price;
week9_10$#      return new;
week9_10$# end;
week9_10$# $$;
CREATE FUNCTION
week9_10=# create trigger f3
week9_10=# after insert on order_item
week9_10=# for each row execute procedure f3();
CREATE TRIGGER
week9_10=# insert into summary_table values(0,0);
INSERT 0 1
week9_10=# insert into order_item values('apple', 5, 100);
ERROR:  null value in column "id" of relation "order_item" violates not-null constraint
DETAIL:  Failing row contains (apple, 5, 100, null).
week9_10=# insert into order_item values('apple', 5, 100, 1);
INSERT 0 1
week9_10=# insert into order_item values('mango', 10, 150, 2);
INSERT 0 1
week9_10=# select * from summary_table;
 number_items | total_price
-----
            2 |         2000
(1 row)

week9_10=#
```