Ramya Narasimha Prabhu
PES1UG19CS380
5F2

SQL – Creating Triggers and Functions

WEEK 9-10

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.

Ans.

SQL for funcs and triggers:

--function to increment emp count on a new hire

```
create or replace function new_hire_f()
returns trigger as $example_table$
BEGIN
update dept
set count_emp=count_emp+1
where new.dno=dept.dnumber;
return new;
end;
$example_table$ language plpgsql;
```

--Trigger for new hire

```
create trigger new_hire
after insert
on emp
for each row
```

Ramya Narasimha Prabhu
PES1UG19CS380
5F2

execute procedure new_hire_f();

-- function to decrement count_emp

create or replace function new_fire_f()

returns trigger as $example_table$

BEGIN

update dept

set count_emp=count_emp-1

where old.dno=dept.dnumber;

return old;

end;

$example_table$ language plpgsql;'

--Trigger to deceremnt count once employee entry is deleted

create trigger new_fire

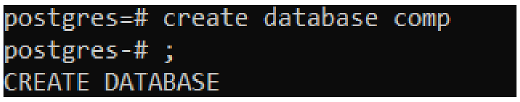
before delete

on emp

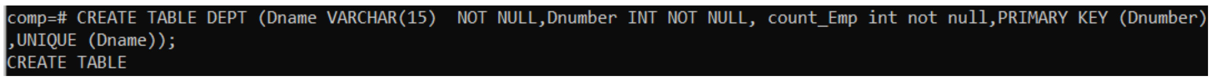
for each row

execute procedure new_fire_f();

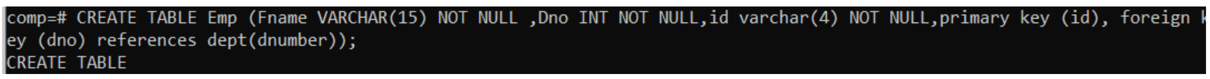
Creating db, creating tables:



```
postgres=# create database comp
postgres-# ;
CREATE DATABASE
```



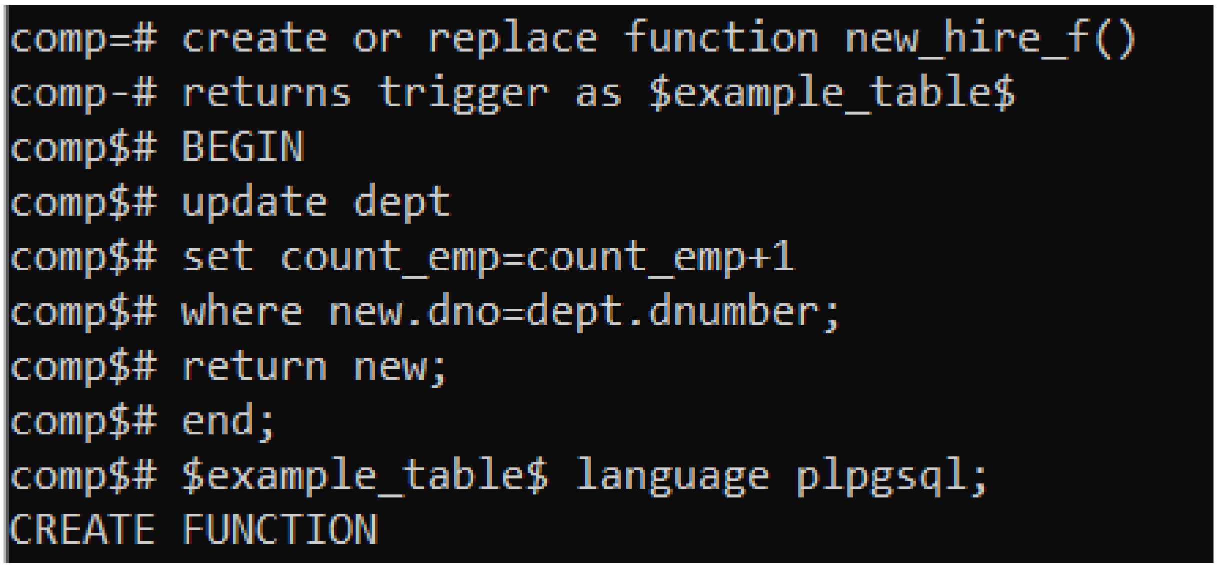
```
comp=# CREATE TABLE DEPT (Dname VARCHAR(15)  NOT NULL,Dnumber INT NOT NULL, count_Emp int not null,PRIMARY KEY (Dnumber)
,UNIQUE (Dname));
CREATE TABLE
```



```
comp=# CREATE TABLE Emp (Fname VARCHAR(15) NOT NULL ,Dno INT NOT NULL,id varchar(4) NOT NULL,primary key (id), foreign k
ey (dno) references dept(dnumber));
CREATE TABLE
```

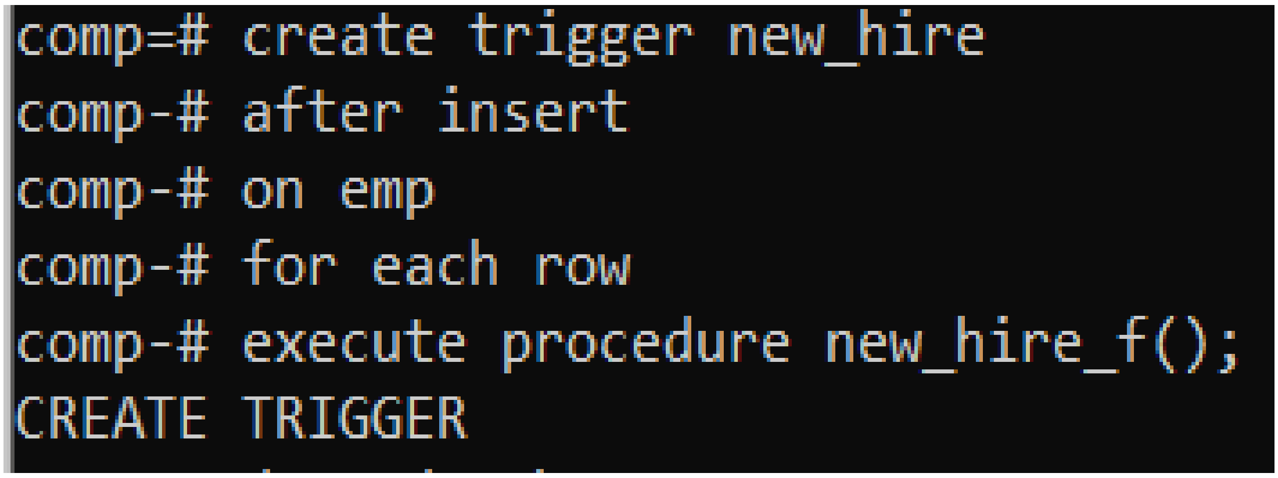
Creating function:

Ramya Narasimha Prabhu
PES1UG19CS380
5F2



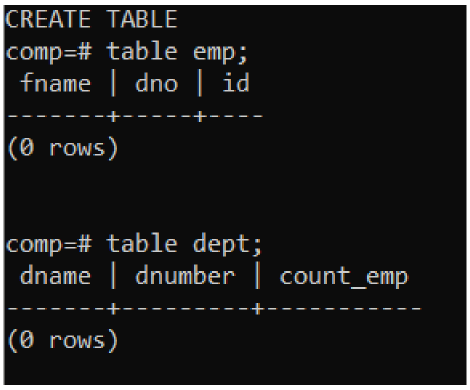
```
comp=# create or replace function new_hire_f()
comp-# returns trigger as $example_table$
comp$# BEGIN
comp$# update dept
comp$# set count_emp=count_emp+1
comp$# where new.dno=dept.dnumber;
comp$# return new;
comp$# end;
comp$# $example_table$ language plpgsql;
CREATE FUNCTION
```

Creating the trigger:



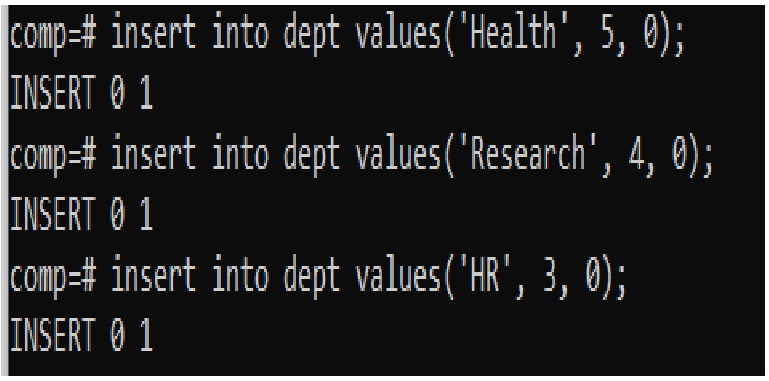
```
comp=# create trigger new_hire
comp-# after insert
comp-# on emp
comp-# for each row
comp-# execute procedure new_hire_f();
CREATE TRIGGER
```

Inserting records into dept table:



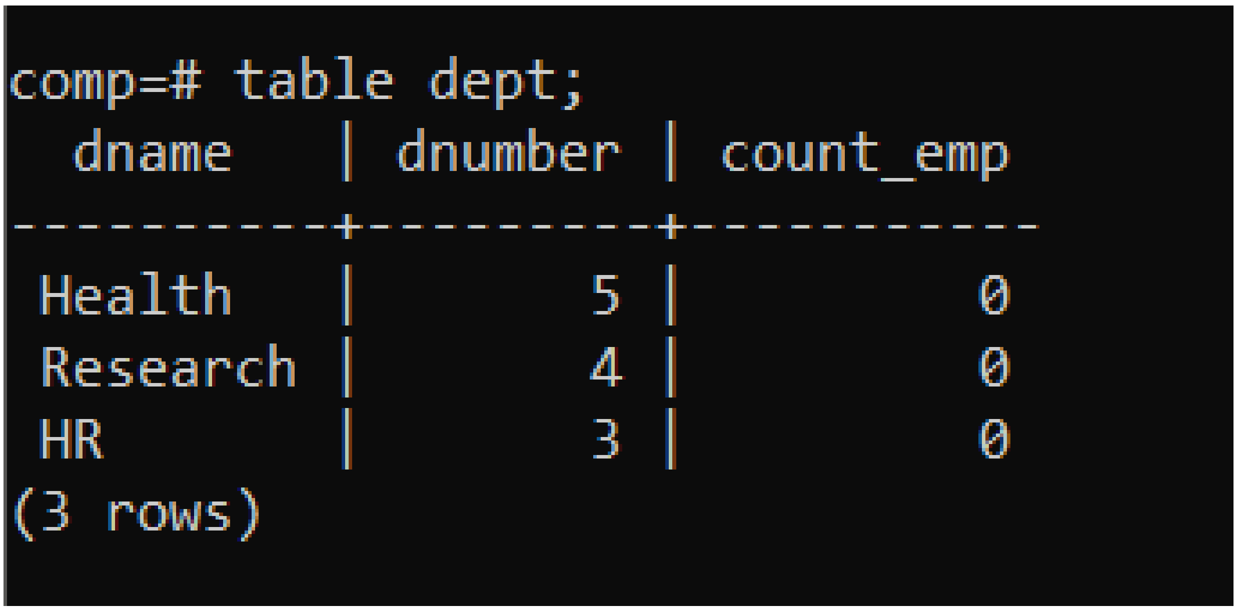
```
CREATE TABLE
comp=# table emp;
 fname | dno | id
-------+-----+----
(0 rows)


comp=# table dept;
 dname | dnumber | count_emp
-------+---------+-----------
(0 rows)
```



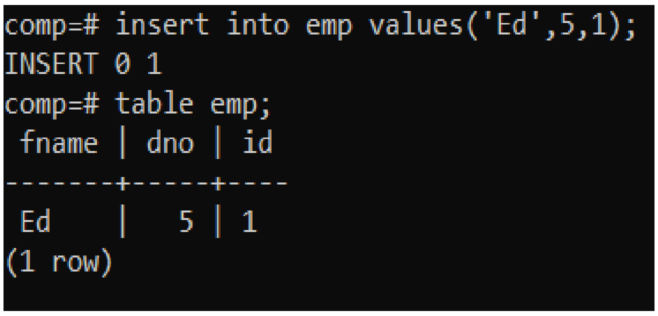
```
comp=# insert into dept values('Health', 5, 0);
INSERT 0 1
comp=# insert into dept values('Research', 4, 0);
INSERT 0 1
comp=# insert into dept values('HR', 3, 0);
INSERT 0 1
```

Ramya Narasimha Prabhu
PES1UG19CS380
5F2

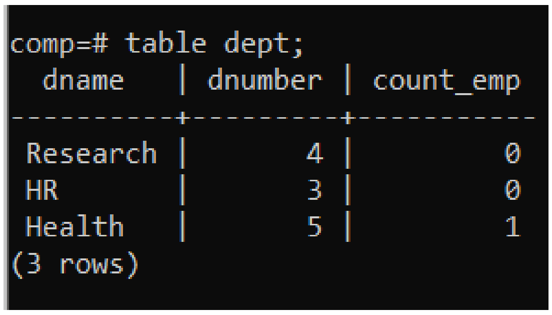


```
comp=# table dept;
  dname    | dnumber | count_emp
-----------+---------+-----------
 Health    |       5 |         0
 Research  |       4 |         0
 HR        |       3 |         0
(3 rows)
```

Inserting Values into emp table [increment]:



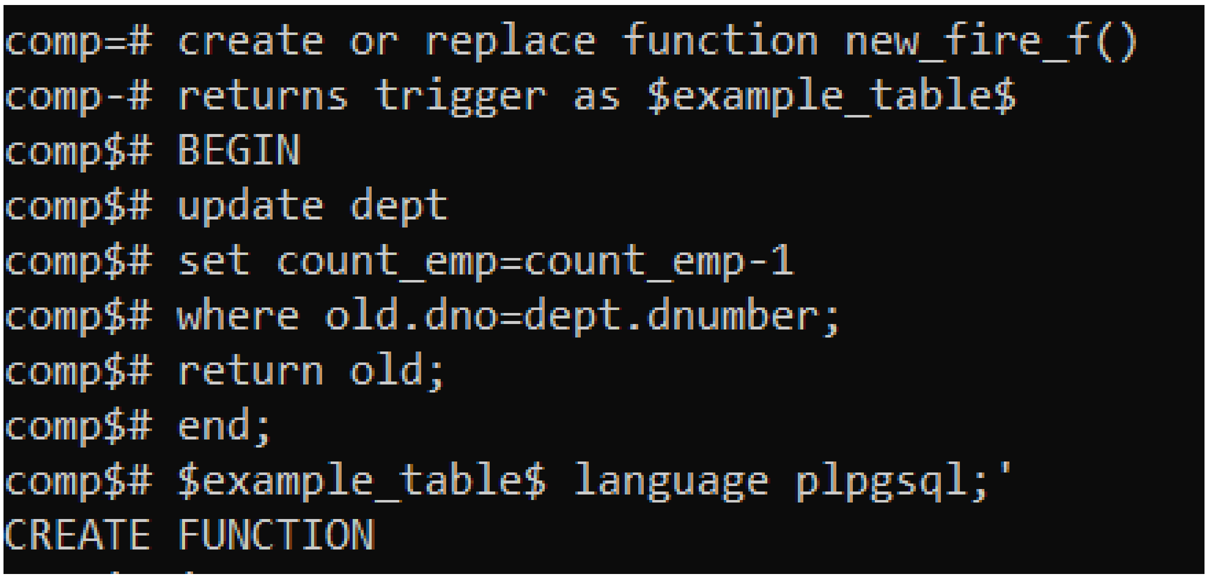
```
comp=# insert into emp values('Ed',5,1);
INSERT 0 1
comp=# table emp;
 fname | dno | id
-------+-----+----
 Ed    |   5 | 1
(1 row)
```



```
comp=# table dept;
  dname    | dnumber | count_emp
-----------+---------+-----------
 Research  |       4 |         0
 HR        |       3 |         0
 Health    |       5 |         1
(3 rows)
```

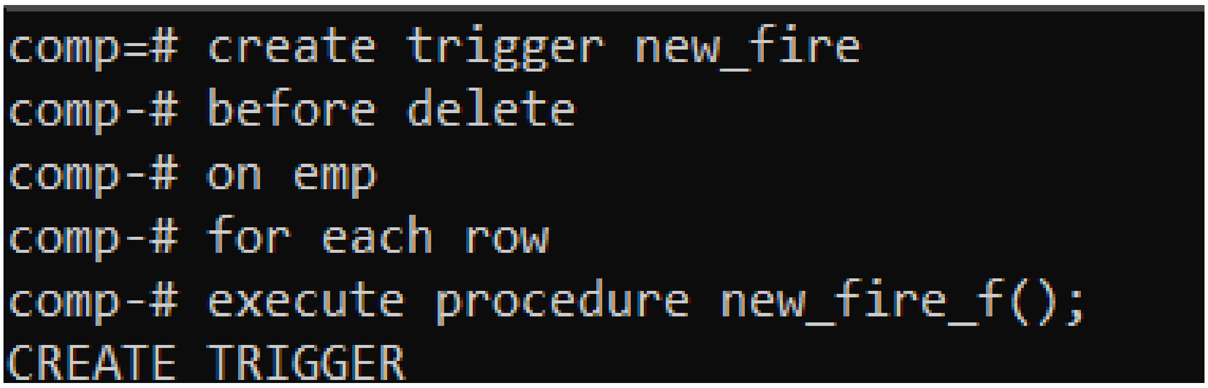
Creating a function for delete:

Ramya Narasimha Prabhu
PES1UG19CS380
5F2



```
comp=# create or replace function new_fire_f()
comp-# returns trigger as $example_table$
comp$# BEGIN
comp$# update dept
comp$# set count_emp=count_emp-1
comp$# where old.dno=dept.dnumber;
comp$# return old;
comp$# end;
comp$# $example_table$ language plpgsql;'
CREATE FUNCTION
```

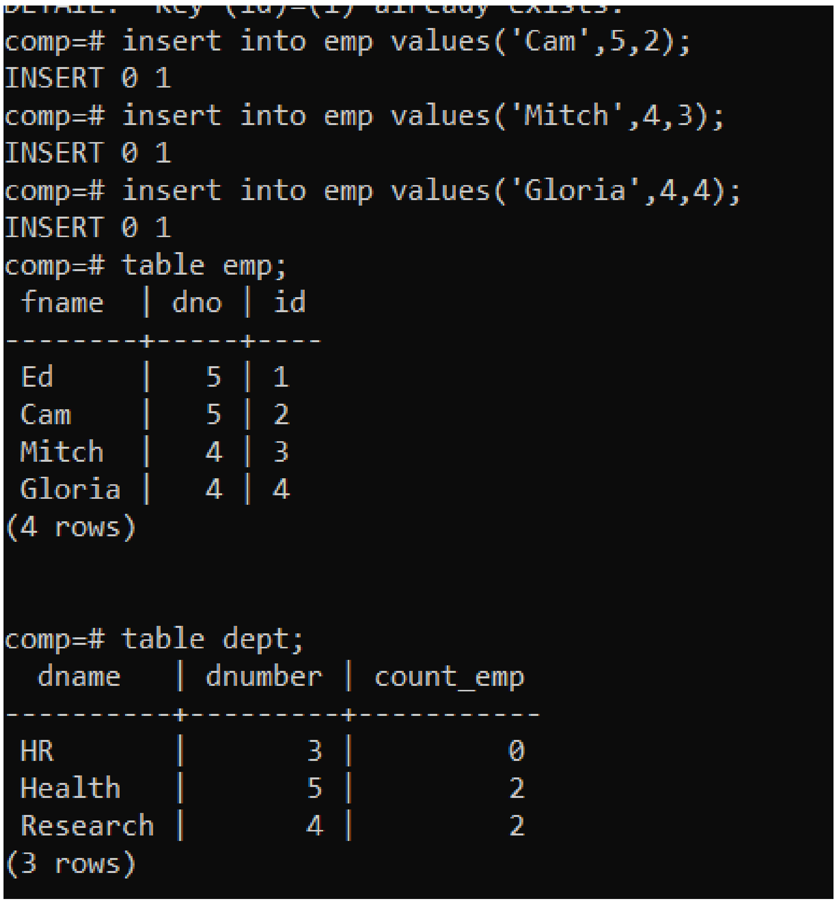
Creating trigger:



```
comp=# create trigger new_fire
comp-# before delete
comp-# on emp
comp-# for each row
comp-# execute procedure new_fire_f();
CREATE TRIGGER
```

Inserting more records:

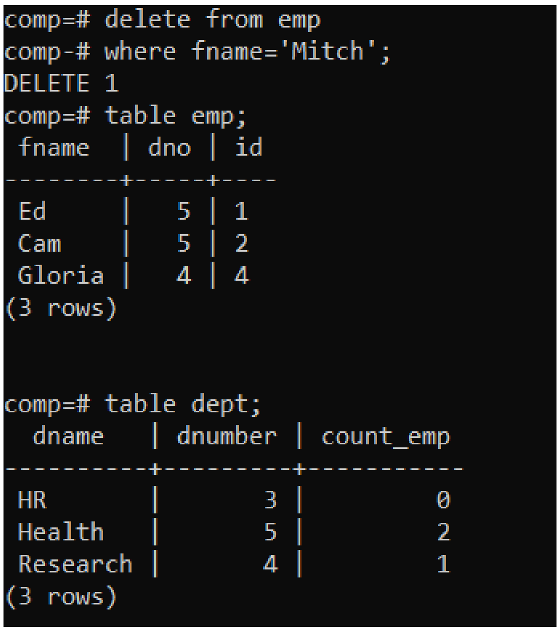
Ramya Narasimha Prabhu
PES1UG19CS380
5F2



```
DETAIL:  Key (id)=(1) already exists.
comp=# insert into emp values('Cam',5,2);
INSERT 0 1
comp=# insert into emp values('Mitch',4,3);
INSERT 0 1
comp=# insert into emp values('Gloria',4,4);
INSERT 0 1
comp=# table emp;
 fname  | dno | id
--------+-----+----
 Ed     |   5 | 1
 Cam    |   5 | 2
 Mitch  |   4 | 3
 Gloria |   4 | 4
(4 rows)


comp=# table dept;
  dname   | dnumber | count_emp
----------+---------+-----------
 HR       |       3 |         0
 Health   |       5 |         2
 Research |       4 |         2
(3 rows)
```

Deleting a record to check if the trigger works:



```
comp=# delete from emp
comp-# where fname='Mitch';
DELETE 1
comp=# table emp;
 fname  | dno | id
--------+-----+----
 Ed     |   5 | 1
 Cam    |   5 | 2
 Gloria |   4 | 4
(3 rows)


comp=# table dept;
  dname   | dnumber | count_emp
----------+---------+-----------
 HR       |       3 |         0
 Health   |       5 |         2
 Research |       4 |         1
(3 rows)
```

It works!

Ramya Narasimha Prabhu
PES1UG19CS380
5F2

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.

Ans:

```
--Creating function for insert

create or replace function add_item_f()

returns trigger as $example_table$

BEGIN

update order_summary

set tot_price=tot_price+new.qty*new.unit_price,

no_item=no_item+new.qty;

return new;

end;

$example_table$ language plpgsql;


--trigger for insert

create trigger new_item_added

after insert

on order_item

for each row

execute procedure add_item_f();


--function on delete

create or replace function remove_item_f()

returns trigger as $example_table$

BEGIN

update order_summary

set tot_price=tot_price-old.qty*old.unit_price,

no_item=no_item-old.qty;

return old;
```

```
end;
$example_table$ language plpgsql;
```

--trigger on delete

```
create trigger item_removed
before delete
on order_item
for each row
execute procedure remove_item_f();
```

--creating function on update

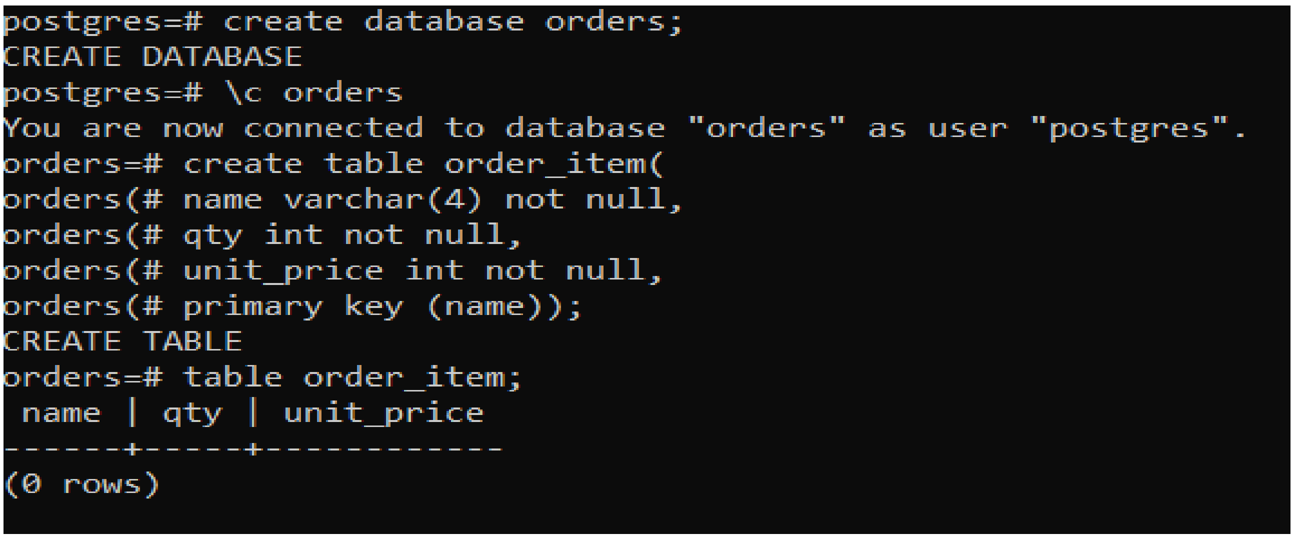
```
create or replace function update_qty_f()
returns trigger as $example_table$
BEGIN
update order_summary
set tot_price=tot_price+((new.qty-old.qty)*old.unit_price),
no_item=no_item-old.qty+new.qty;
return new;
end;
$example_table$ language plpgsql;
```

-- Creating Trigger on update

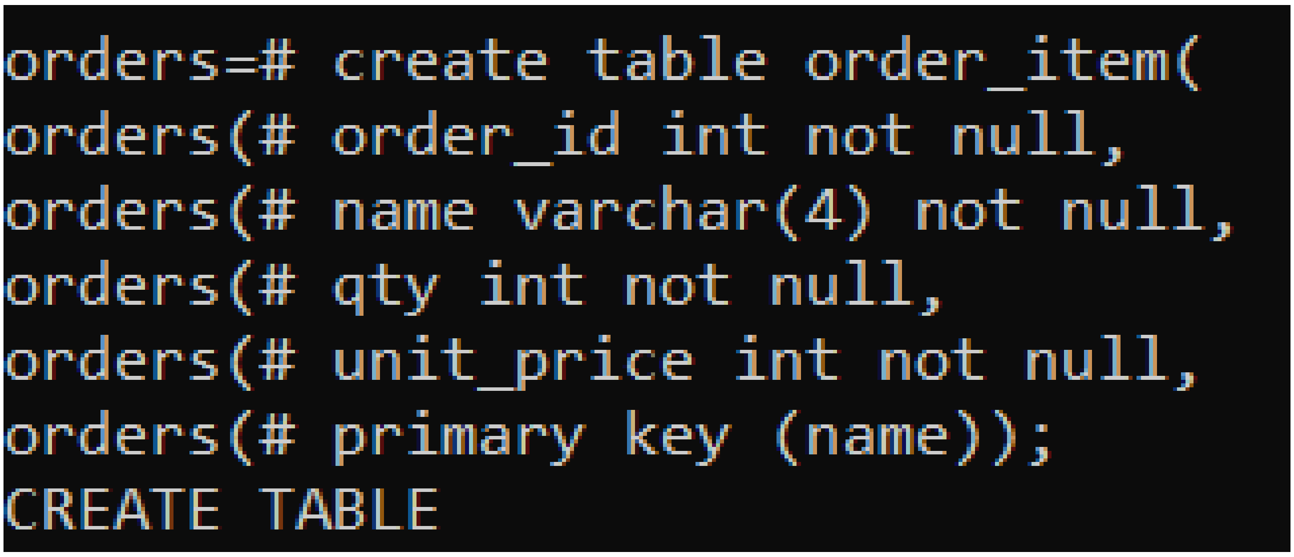
```
create trigger qty_changed
after update
on order_item
for each row
execute procedure remove_item_f();
```

creating db and tables:

Ramya Narasimha Prabhu
PES1UG19CS380
5F2

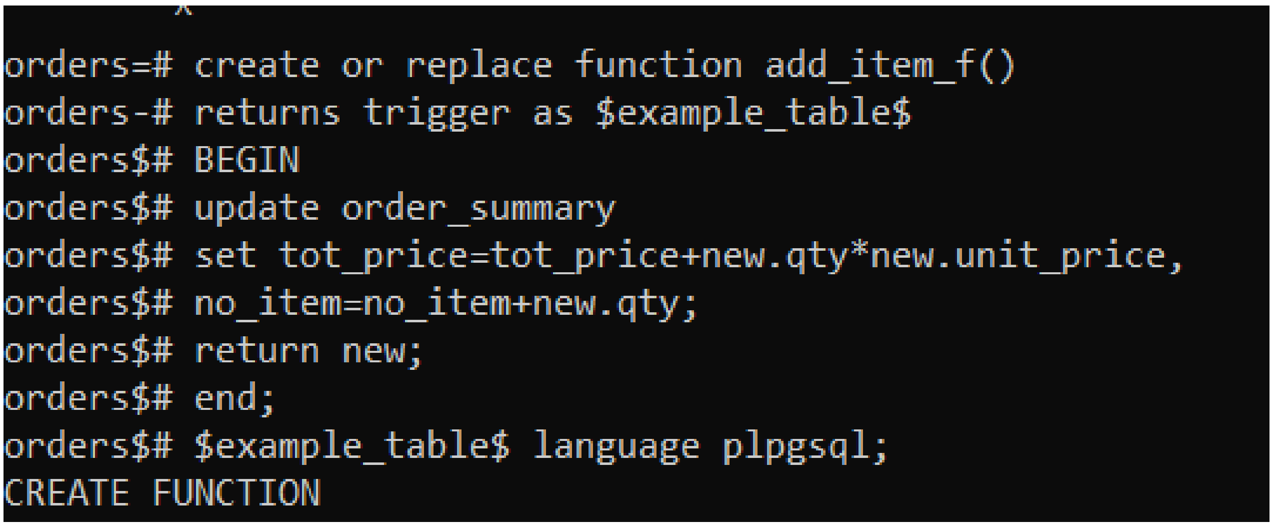


```
postgres=# create database orders;
CREATE DATABASE
postgres=# \c orders
You are now connected to database "orders" as user "postgres".
orders=# create table order_item(
orders(# name varchar(4) not null,
orders(# qty int not null,
orders(# unit_price int not null,
orders(# primary key (name));
CREATE TABLE
orders=# table order_item;
 name | qty | unit_price
------+-----+------------
(0 rows)
```



```
orders=# create table order_item(
orders(# order_id int not null,
orders(# name varchar(4) not null,
orders(# qty int not null,
orders(# unit_price int not null,
orders(# primary key (name));
CREATE TABLE
```

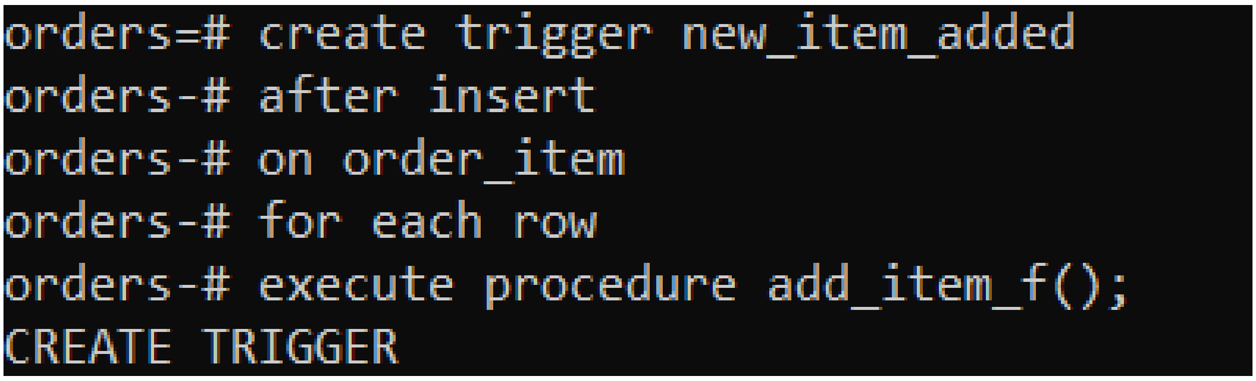
Creating function for insert:



```
orders=# create or replace function add_item_f()
orders-# returns trigger as $example_table$
orders$# BEGIN
orders$# update order_summary
orders$# set tot_price=tot_price+new.qty*new.unit_price,
orders$# no_item=no_item+new.qty;
orders$# return new;
orders$# end;
orders$# $example_table$ language plpgsql;
CREATE FUNCTION
```

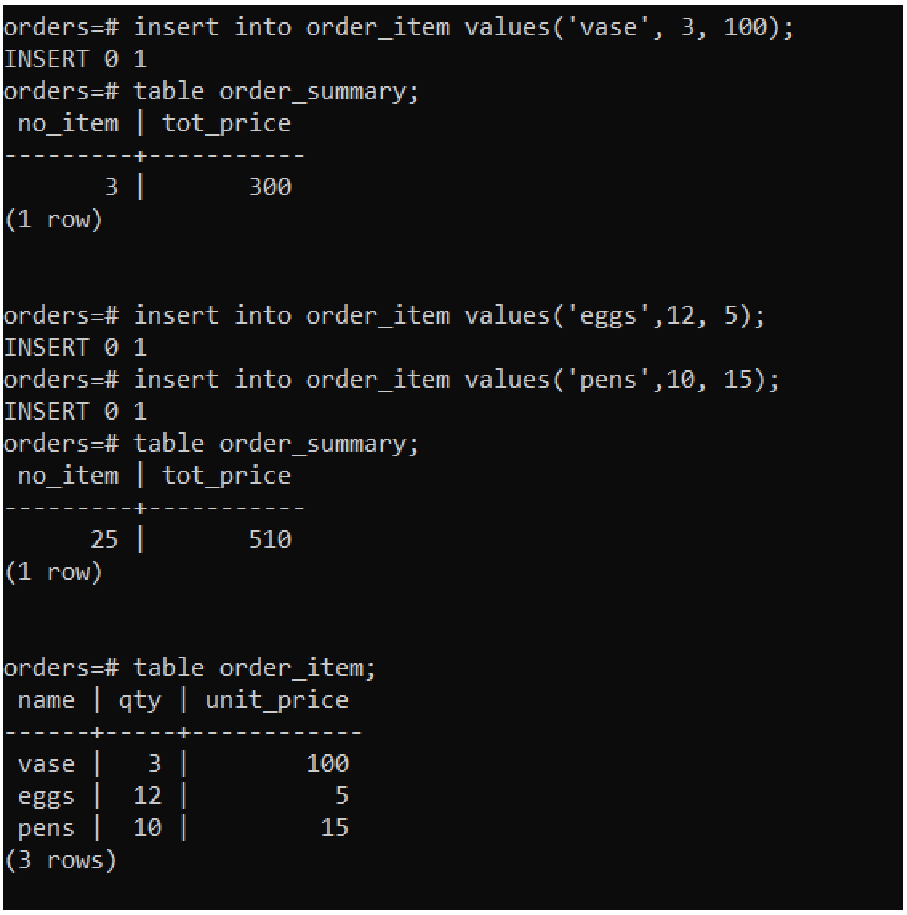
Ramya Narasimha Prabhu
PES1UG19CS380
5F2

Create trigger for insert:



```
orders=# create trigger new_item_added
orders-# after insert
orders-# on order_item
orders-# for each row
orders-# execute procedure add_item_f();
CREATE TRIGGER
```

Insert trigger in action:



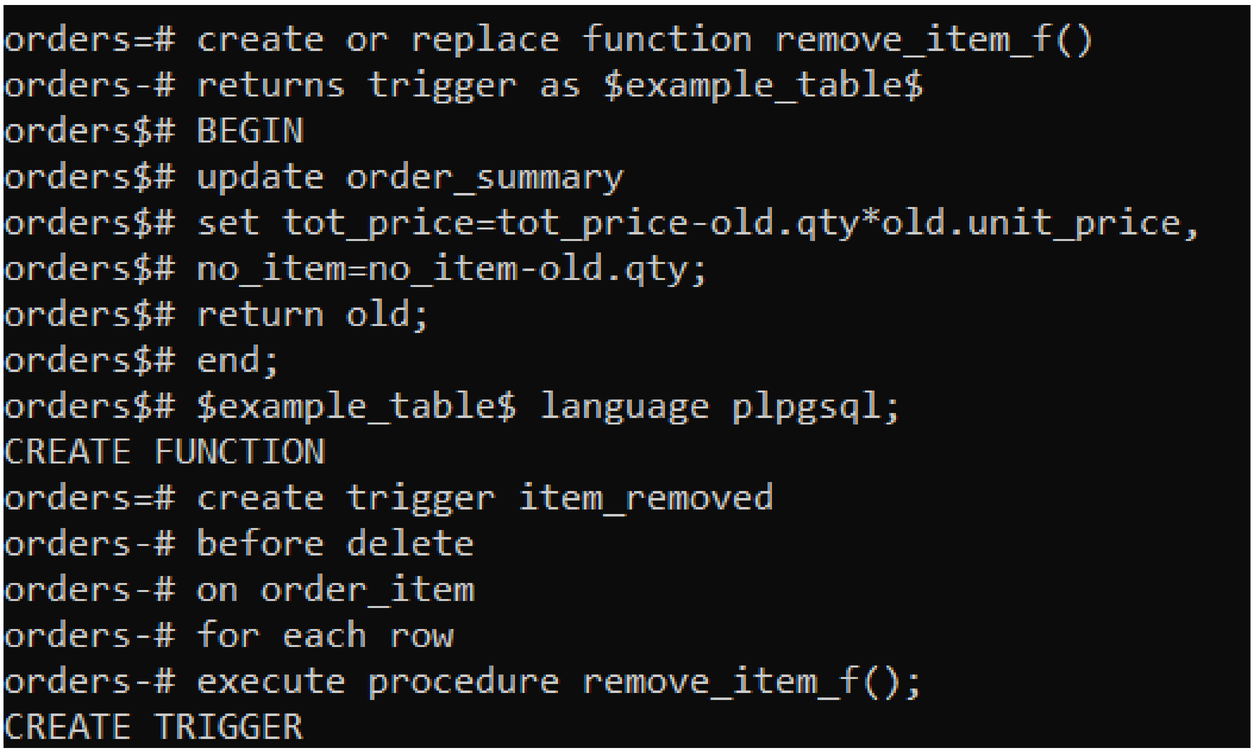
```
orders=# insert into order_item values('vase', 3, 100);
INSERT 0 1
orders=# table order_summary;
 no_item | tot_price
---------+-----------
       3 |       300
(1 row)


orders=# insert into order_item values('eggs',12, 5);
INSERT 0 1
orders=# insert into order_item values('pens',10, 15);
INSERT 0 1
orders=# table order_summary;
 no_item | tot_price
---------+-----------
      25 |       510
(1 row)


orders=# table order_item;
 name | qty | unit_price
------+-----+------------
 vase |   3 |        100
 eggs |  12 |          5
 pens |  10 |         15
(3 rows)
```

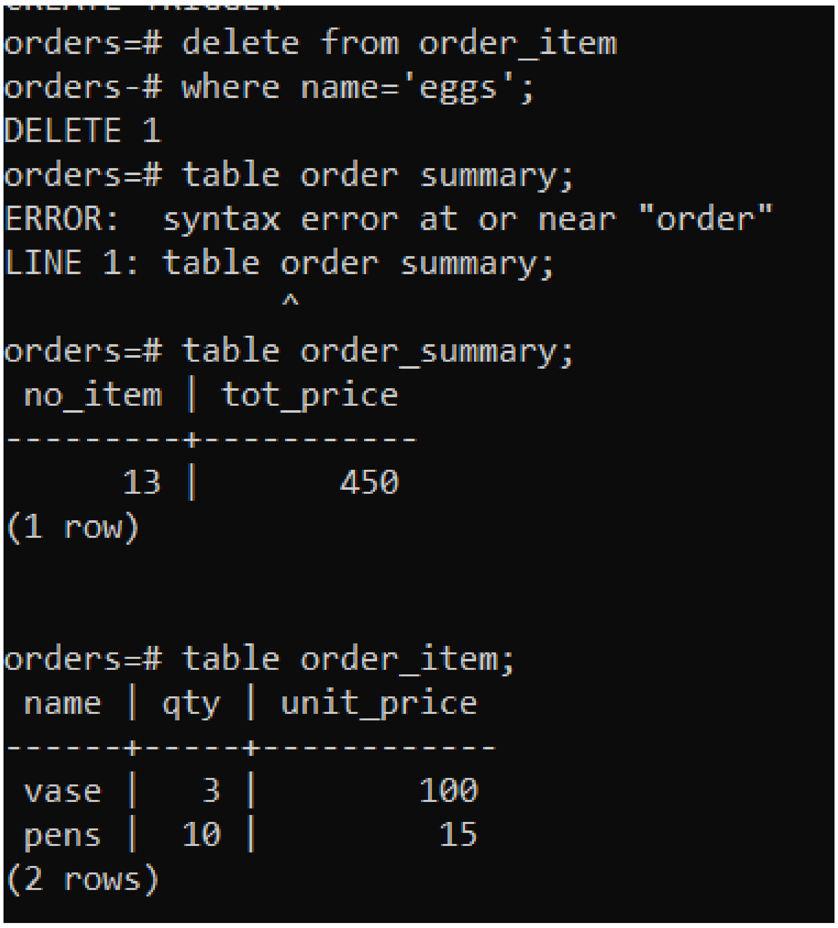
Creating function and trigger for delete:

Ramya Narasimha Prabhu
PES1UG19CS380
5F2



```
orders=# create or replace function remove_item_f()
orders-# returns trigger as $example_table$
orders$# BEGIN
orders$# update order_summary
orders$# set tot_price=tot_price-old.qty*old.unit_price,
orders$# no_item=no_item-old.qty;
orders$# return old;
orders$# end;
orders$# $example_table$ language plpgsql;
CREATE FUNCTION
orders=# create trigger item_removed
orders-# before delete
orders-# on order_item
orders-# for each row
orders-# execute procedure remove_item_f();
CREATE TRIGGER
```

Delete trigger in action:

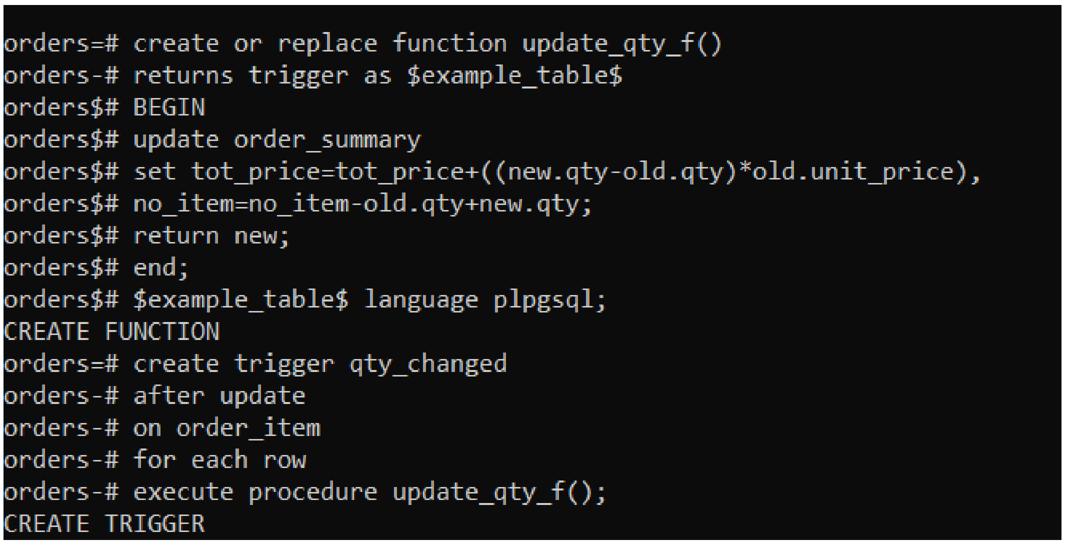


```
orders=# delete from order_item
orders-# where name='eggs';
DELETE 1
orders=# table order summary;
ERROR:  syntax error at or near "order"
LINE 1: table order summary;
              ^
orders=# table order_summary;
 no_item | tot_price
---------+-----------
      13 |       450
(1 row)


orders=# table order_item;
 name | qty | unit_price
------+-----+------------
 vase |   3 |        100
 pens |  10 |         15
(2 rows)
```

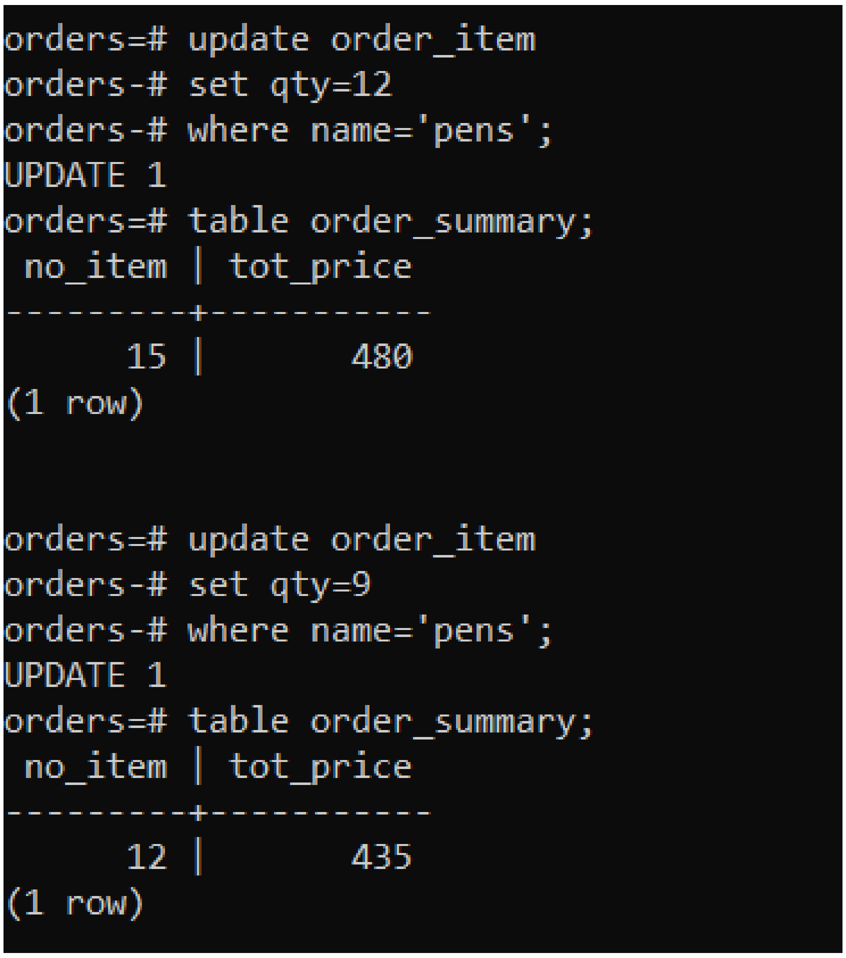
Ramya Narasimha Prabhu
PES1UG19CS380
5F2

Creating a function and trigger for update:



```
orders=# create or replace function update_qty_f()
orders-# returns trigger as $example_table$
orders$# BEGIN
orders$# update order_summary
orders$# set tot_price=tot_price+((new.qty-old.qty)*old.unit_price),
orders$# no_item=no_item-old.qty+new.qty;
orders$# return new;
orders$# end;
orders$# $example_table$ language plpgsql;
CREATE FUNCTION
orders=# create trigger qty_changed
orders-# after update
orders-# on order_item
orders-# for each row
orders-# execute procedure update_qty_f();
CREATE TRIGGER
```

Update trigger in action:



```
orders=# update order_item
orders-# set qty=12
orders-# where name='pens';
UPDATE 1
orders=# table order_summary;
 no_item | tot_price
---------+-----------
      15 |       480
(1 row)


orders=# update order_item
orders-# set qty=9
orders-# where name='pens';
UPDATE 1
orders=# table order_summary;
 no_item | tot_price
---------+-----------
      12 |       435
(1 row)
```

Ramya Narasimha Prabhu
PES1UG19CS380
5F2