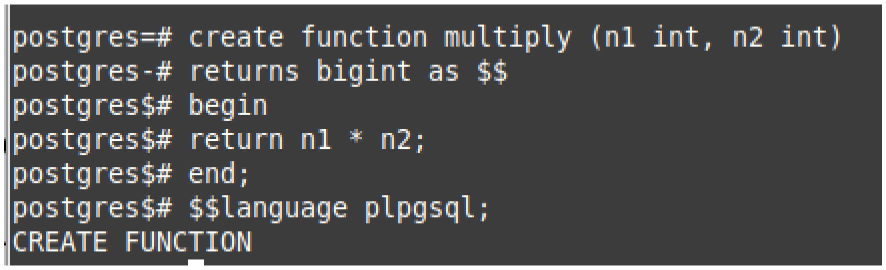
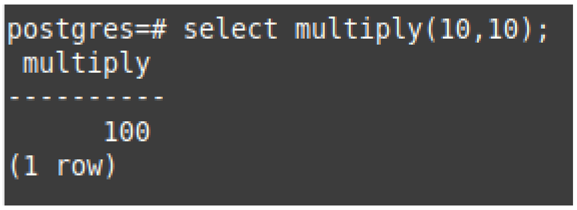
1. Create a multiply function that accepts two numbers and returns their product.

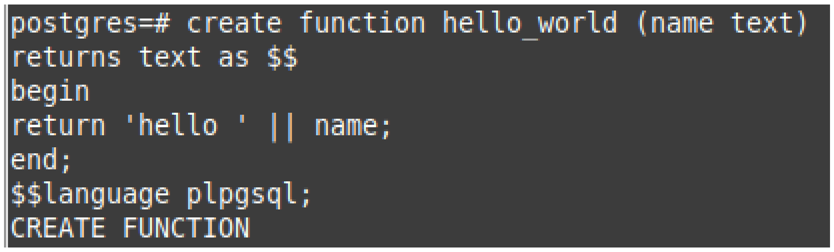


```
postgres=# create function multiply (n1 int, n2 int)
postgres-# returns bigint as $$
postgres$# begin
postgres$# return n1 * n2;
postgres$# end;
postgres$# $$language plpgsql;
CREATE FUNCTION
```

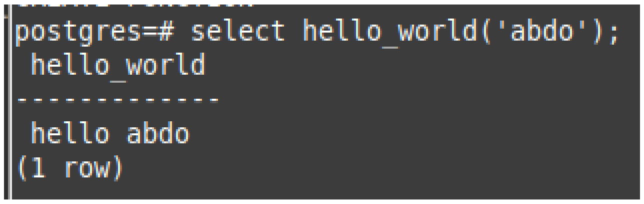


```
postgres=# select multiply(10,10);
 multiply
----------
      100
(1 row)
```

2. Create a hello_world function that takes a name as input and returns a personalized welcome message for that name.

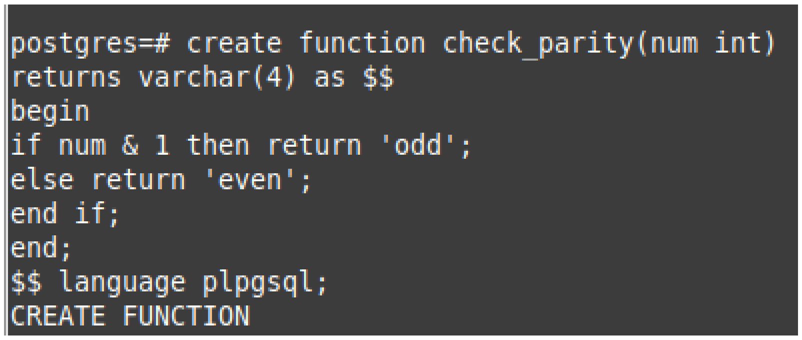


```
postgres=# create function hello_world (name text)
returns text as $$
begin
return 'hello ' || name;
end;
$$language plpgsql;
CREATE FUNCTION
```

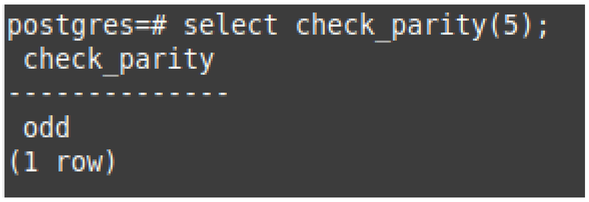


```
postgres=# select hello_world('abdo');
 hello_world
-------------
 hello abdo
(1 row)
```

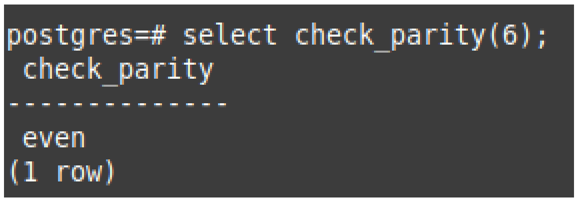
3. Create a function that accepts a number and determines whether it is odd or even.



```
postgres=# create function check_parity(num int)
returns varchar(4) as $$
begin
if num & 1 then return 'odd';
else return 'even';
end if;
end;
$$ language plpgsql;
CREATE FUNCTION
```

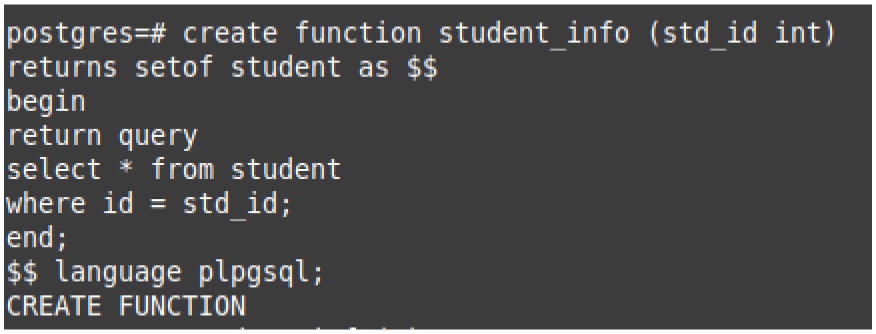


```
postgres=# select check_parity(5);
 check_parity
--------------
 odd
(1 row)
```

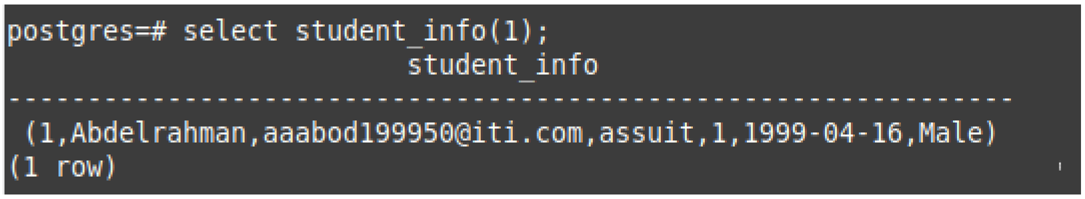


```
postgres=# select check_parity(6);
 check_parity
--------------
 even
(1 row)
```

4. Create a function that takes a Student ID as input and retrieves all information related to that student.

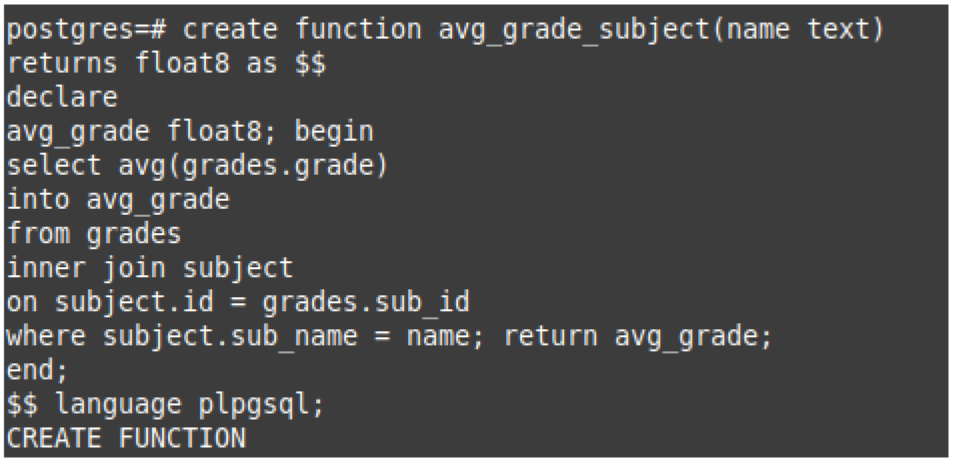


```
postgres=# create function student_info (std_id int)
returns setof student as $$
begin
return query
select * from student
where id = std_id;
end;
$$ language plpgsql;
CREATE FUNCTION
```

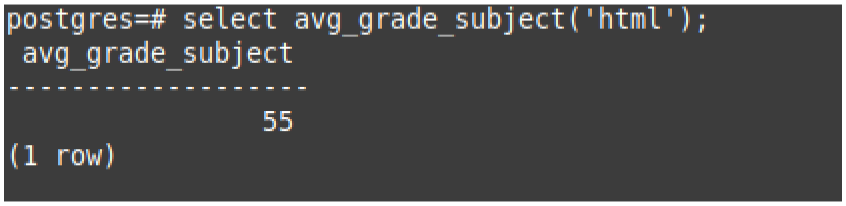


```
postgres=# select student_info(1);
                         student_info
---------------------------------------------------------------
 (1,Abdelrahman,aaabod199950@iti.com,assuit,1,1999-04-16,Male)
(1 row)
```

5. Implement a function that takes the name of a subject and calculates the average grades for that subject.

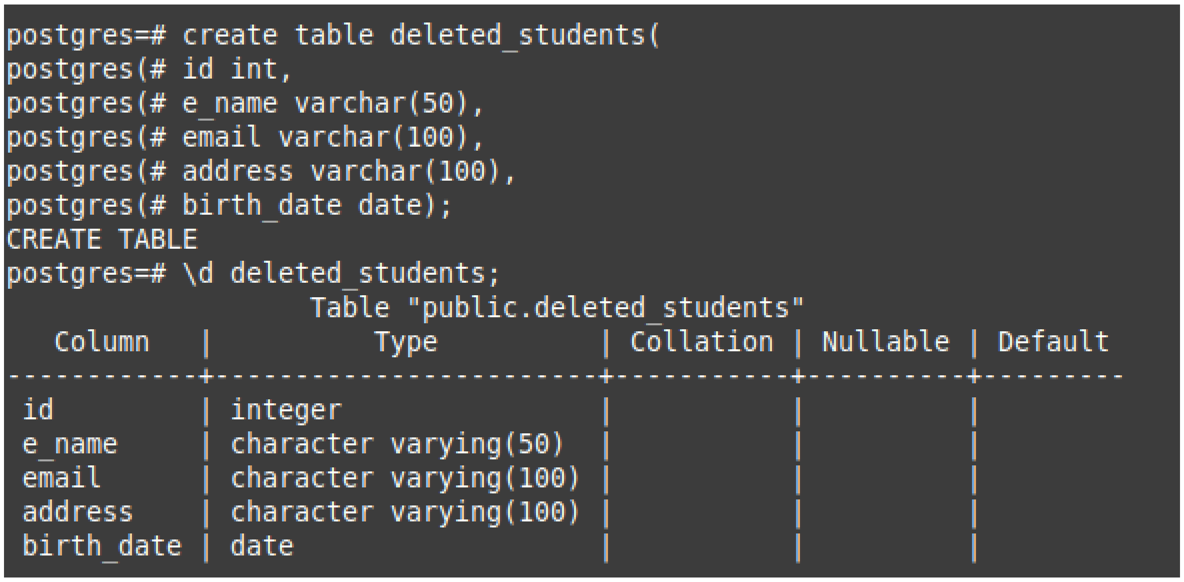


```
postgres=# create function avg_grade_subject(name text)
returns float8 as $$
declare
avg_grade float8; begin
select avg(grades.grade)
into avg_grade
from grades
inner join subject
on subject.id = grades.sub_id
where subject.sub_name = name; return avg_grade;
end;
$$ language plpgsql;
CREATE FUNCTION
```

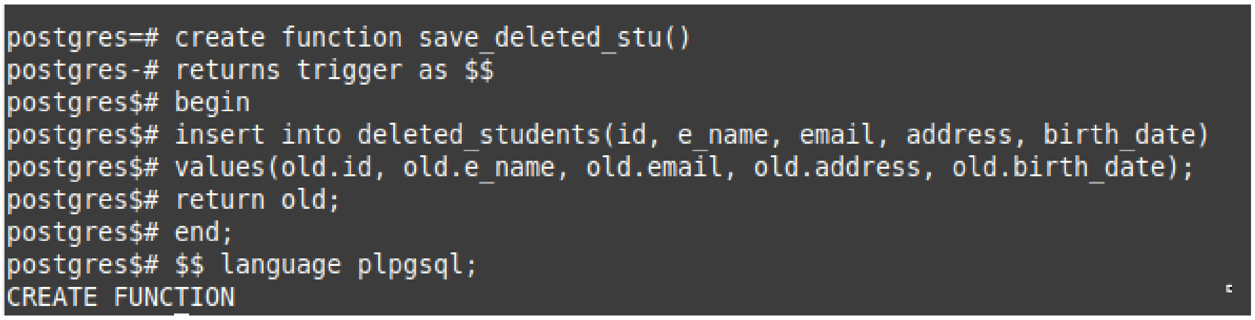


```
postgres=# select avg_grade_subject('html');
 avg_grade_subject
-------------------
                55
(1 row)
```

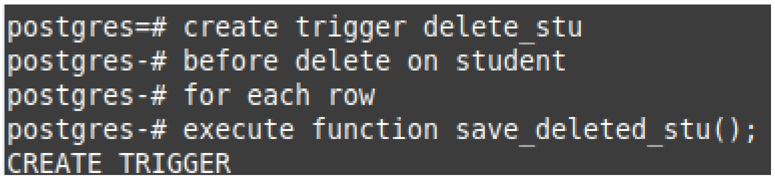
6. Create a trigger to automatically save deleted student records from the Student table to the Deleted_Students table.



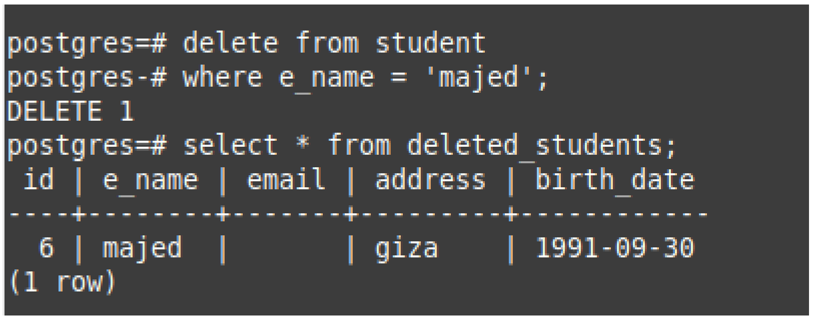
```
postgres=# create table deleted_students(
postgres(# id int,
postgres(# e_name varchar(50),
postgres(# email varchar(100),
postgres(# address varchar(100),
postgres(# birth_date date);
CREATE TABLE
postgres=# \d deleted_students;
                   Table "public.deleted_students"
   Column   |          Type          | Collation | Nullable | Default
------------+------------------------+-----------+----------+---------
 id         | integer                |           |          |
 e_name     | character varying(50)  |           |          |
 email      | character varying(100) |           |          |
 address    | character varying(100) |           |          |
 birth_date | date                   |           |          |
```



```
postgres=# create function save_deleted_stu()
postgres-# returns trigger as $$
postgres$# begin
postgres$# insert into deleted_students(id, e_name, email, address, birth_date)
postgres$# values(old.id, old.e_name, old.email, old.address, old.birth_date);
postgres$# return old;
postgres$# end;
postgres$# $$ language plpgsql;
CREATE FUNCTION
```

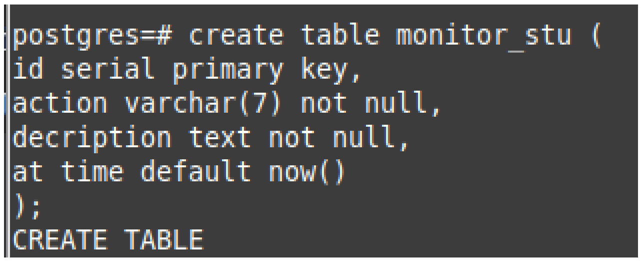


```
postgres=# create trigger delete_stu
postgres-# before delete on student
postgres-# for each row
postgres-# execute function save_deleted_stu();
CREATE TRIGGER
```

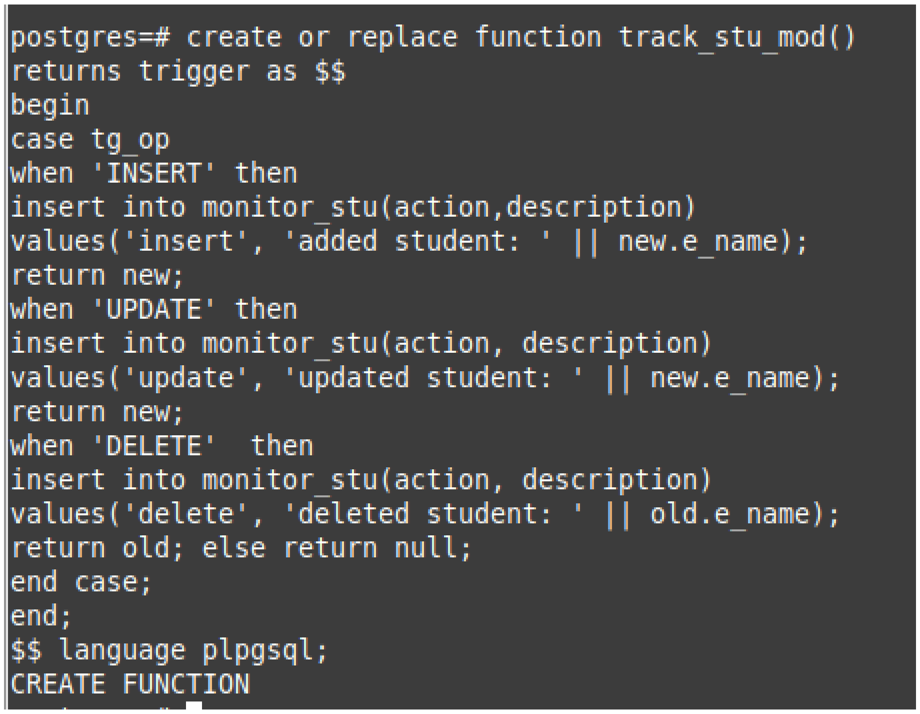


```
postgres=# delete from student
postgres-# where e_name = 'majed';
DELETE 1
postgres=# select * from deleted_students;
 id | e_name | email | address | birth_date
----+--------+-------+---------+------------
  6 | majed  |       | giza    | 1991-09-30
(1 row)
```

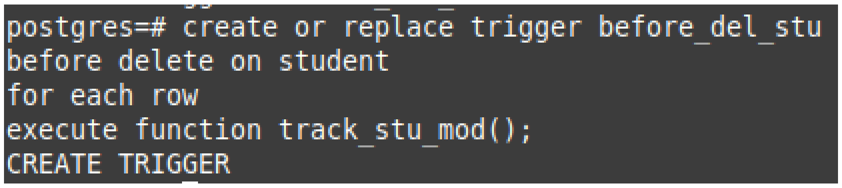
7. Create a trigger to monitor changes made to the student table, including additions, updates, and deletions. This trigger will record the time of each action and provide a description of the action in another table.



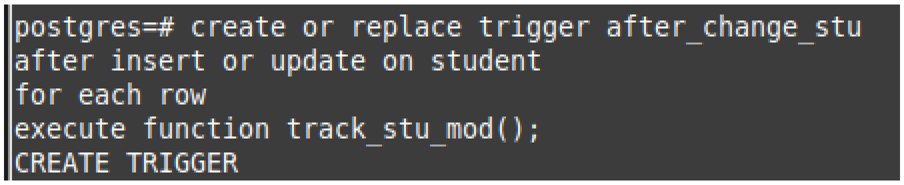
```
postgres=# create table monitor_stu (
id serial primary key,
action varchar(7) not null,
decription text not null,
at time default now()
);
CREATE TABLE
```



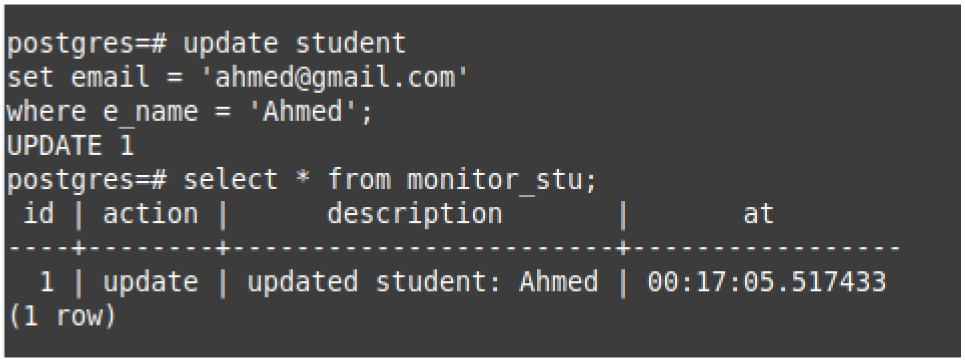
```
postgres=# create or replace function track_stu_mod()
returns trigger as $$
begin
case tg_op
when 'INSERT' then
insert into monitor_stu(action,description)
values('insert', 'added student: ' || new.e_name);
return new;
when 'UPDATE' then
insert into monitor_stu(action, description)
values('update', 'updated student: ' || new.e_name);
return new;
when 'DELETE'  then
insert into monitor_stu(action, description)
values('delete', 'deleted student: ' || old.e_name);
return old; else return null;
end case;
end;
$$ language plpgsql;
CREATE FUNCTION
```



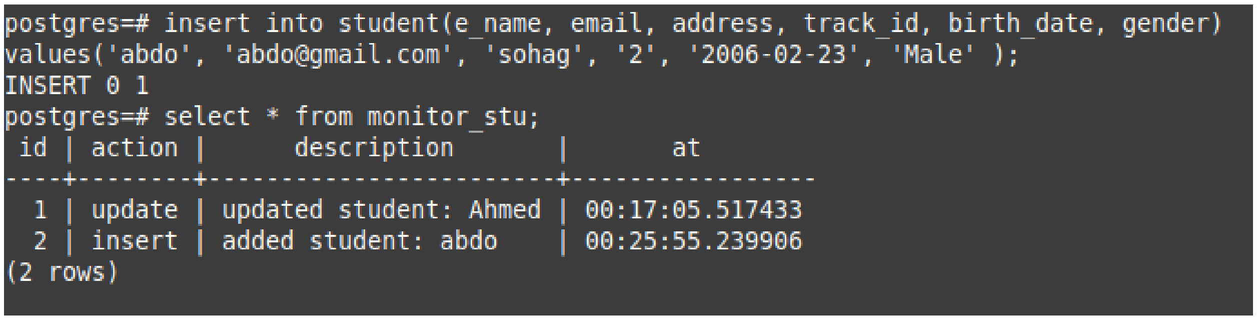
```
postgres=# create or replace trigger before_del_stu
before delete on student
for each row
execute function track_stu_mod();
CREATE TRIGGER
```



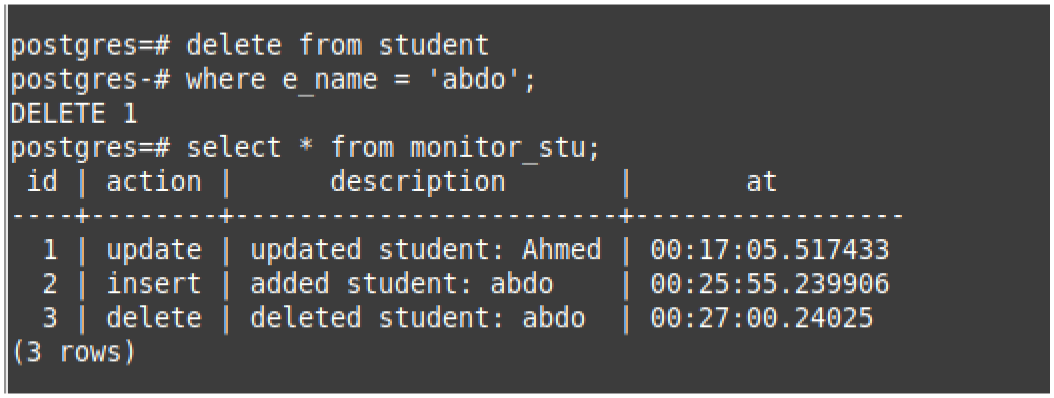
```
postgres=# create or replace trigger after_change_stu
after insert or update on student
for each row
execute function track_stu_mod();
CREATE TRIGGER
```



```
postgres=# update student
set email = 'ahmed@gmail.com'
where e_name = 'Ahmed';
UPDATE 1
postgres=# select * from monitor_stu;
 id | action |      description       |       at
----+--------+------------------------+-----------------
  1 | update | updated student: Ahmed | 00:17:05.517433
(1 row)
```



```
postgres=# insert into student(e_name, email, address, track_id, birth_date, gender)
values('abdo', 'abdo@gmail.com', 'sohag', '2', '2006-02-23', 'Male' );
INSERT 0 1
postgres=# select * from monitor_stu;
 id | action |      description       |       at
----+--------+------------------------+-----------------
  1 | update | updated student: Ahmed | 00:17:05.517433
  2 | insert | added student: abdo    | 00:25:55.239906
(2 rows)
```



```
postgres=# delete from student
postgres-# where e_name = 'abdo';
DELETE 1
postgres=# select * from monitor_stu;
 id | action |      description       |       at
----+--------+------------------------+-----------------
  1 | update | updated student: Ahmed | 00:17:05.517433
  2 | insert | added student: abdo    | 00:25:55.239906
  3 | delete | deleted student: abdo  | 00:27:00.24025
(3 rows)
```