**Day 9 Assignment**

**--Qn1. Create AFTER UPDATE trigger to track product price changes**

**QUERY DETAILS**

--Create product\_price\_audit table with below columns:

create table product\_price\_audit (

audit\_id serial primary key,

product\_id int,

product\_name varchar(40),

old\_price decimal(10,2),

new\_price decimal(10,2),

change\_date timestamp default current\_timestamp,

user\_name varchar(50) default current\_user

);

-- Create a trigger function with the below logic:

create or replace function track\_price\_changes()

returns trigger as $$

begin

insert into product\_price\_audit (

product\_id,

product\_name,

old\_price,

new\_price

)

values (

old.product\_id,

old.product\_name,

old.unit\_price,

new.unit\_price

);

return new;

end;

$$ language plpgsql;

-- Create the trigger on products

create trigger after\_price\_update

after update of unit\_price on products

for each row

when (old.unit\_price is distinct from new.unit\_price)

execute function track\_price\_changes();

-- Test the trigger by updating the product price by 10% to any one product\_id.

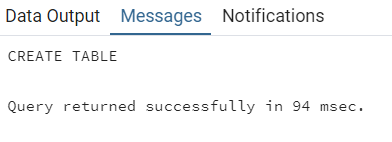
update products

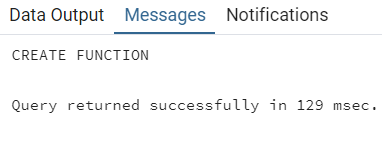
set unit\_price = unit\_price \* 1.10

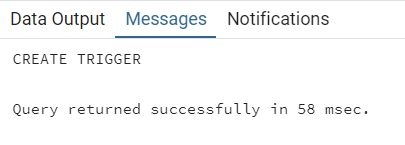
where product\_id = 1;

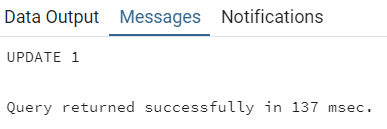
select \* from product\_price\_audit;

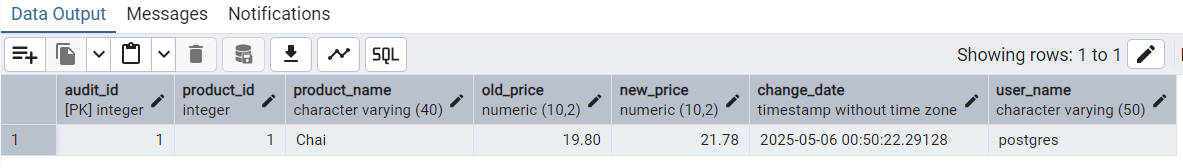
**OUTPUT**

****









**--Qn2.Create stored procedure using IN and INOUT parameters to assign tasks to employees**

**QUERY DETAILS**

-- Create the employee\_tasks table using the Parameters

create table if not exists employee\_tasks (

task\_id serial primary key,

employee\_id int,

task\_name varchar(50),

assigned\_date date default current\_date

);

-- Create the stored procedure

create or replace procedure assign\_task(

in p\_employee\_id int,

in p\_task\_name varchar(50),

inout p\_task\_count int default 0

)

language plpgsql

as $$

begin

insert into employee\_tasks (employee\_id, task\_name)

values (p\_employee\_id, p\_task\_name);

select count(\*) into p\_task\_count

from employee\_tasks

where employee\_id = p\_employee\_id;

raise notice 'Task "%" assigned to employee %. Total tasks: %',

p\_task\_name, p\_employee\_id, p\_task\_count;

end;

$$;

-- After creating stored procedure test by calling it:

call assign\_task(1, 'review reports');

-- You should see the entry in employee\_tasks table.

select \* from employee\_tasks;

**OUTPUT**

