**Day 8**

1. Create view vw\_updatable\_products (use same query whatever I used in the training)

Try updating view with below query and see if the product table also gets updated.

Update query:

UPDATE updatable\_products SET unit\_price = unit\_price \* 1.1 WHERE units\_in\_stock < 10;

**Query:**

CREATE OR REPLACE VIEW VW\_UPDATABLE\_PRODUCTS AS

SELECT

PRODUCT\_ID,

PRODUCT\_NAME,

UNIT\_PRICE,

UNITS\_IN\_STOCK,

DISCONTINUED

FROM

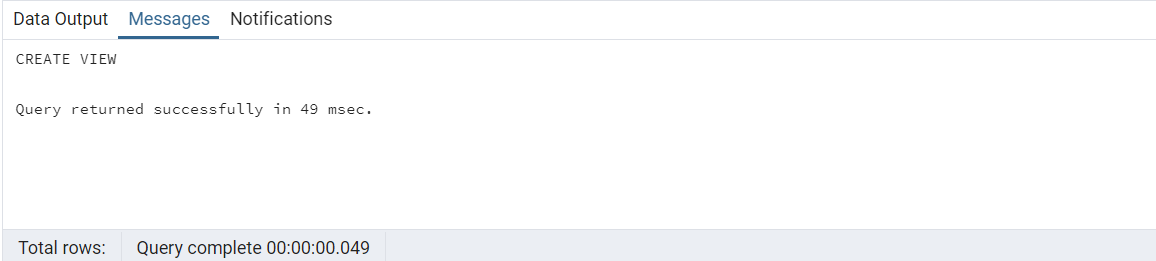
PRODUCTS

WHERE

DISCONTINUED = 0

WITH

CHECK OPTION;



**Output:**

-- query to check before udpateing view

SELECT

PRODUCT\_ID,

PRODUCT\_NAME,

UNIT\_PRICE,

UNITS\_IN\_STOCK,

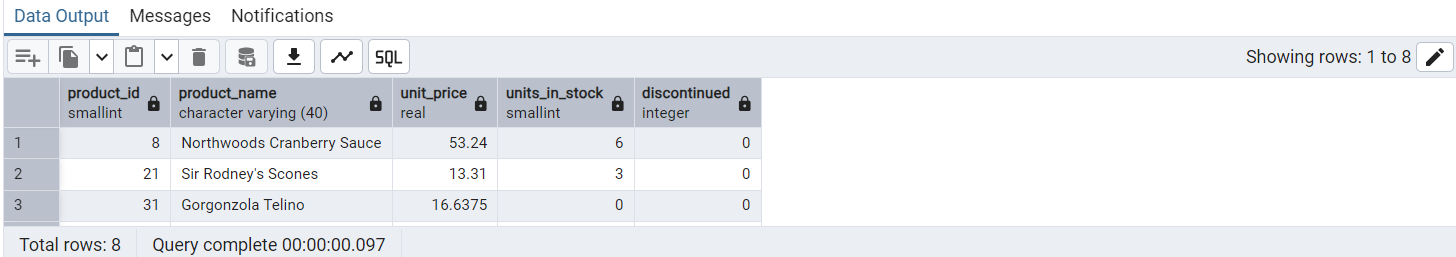
DISCONTINUED

FROM

VW\_UPDATABLE\_PRODUCTS

WHERE

UNITS\_IN\_STOCK < 10;



UPDATE VW\_UPDATABLE\_PRODUCTS

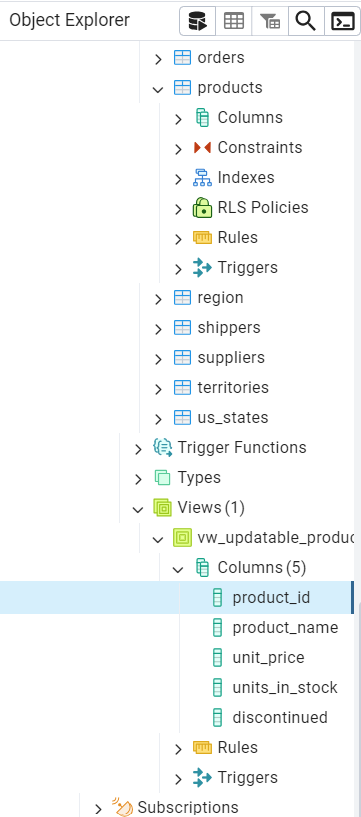
SET

UNIT\_PRICE = UNIT\_PRICE \* 1.1

WHERE

UNITS\_IN\_STOCK < 10;



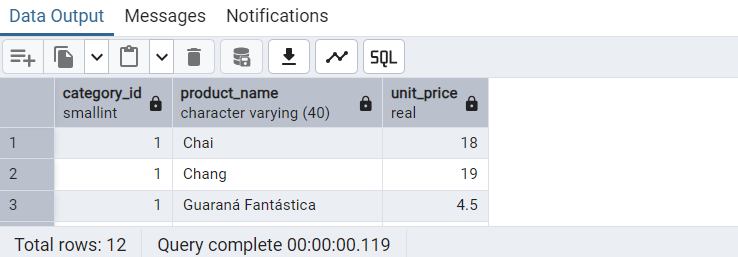


2. Transaction:

Update the product price for products by 10% in category id=1

Try COMMIT and ROLLBACK and observe what happens.

**Query:**

****

begin;

update products set unit\_price = unit\_price \* 1.10 where category\_id =1;

do $$

begin

if exists (select 1 from products where category\_id=1 and unit\_price >50)

then raise notice 'Some unit prices are >50';

else raise notice 'Updated products unitprice';

end if;

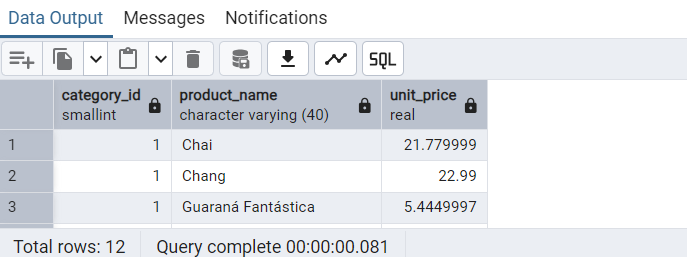
end$$;

commit;

rollback;

**Output:**

select category\_id,product\_name,unit\_price from products where category\_id=1;



3. Create a regular view which will have below details (Need to do joins):

Employee\_id,

Employee\_full\_name,

Title,

Territory\_id,

territory\_description,

region\_description

**Query:**

CREATE OR REPLACE VIEW VW\_EMPLOYEE\_TERRITORY\_REGION AS

SELECT

E.EMPLOYEE\_ID,

FIRST\_NAME || ' ' || LAST\_NAME AS EMPLOYEE\_FULL\_NAME,

TITLE,

T.TERRITORY\_ID,

TERRITORY\_DESCRIPTION,

R.REGION\_DESCRIPTION

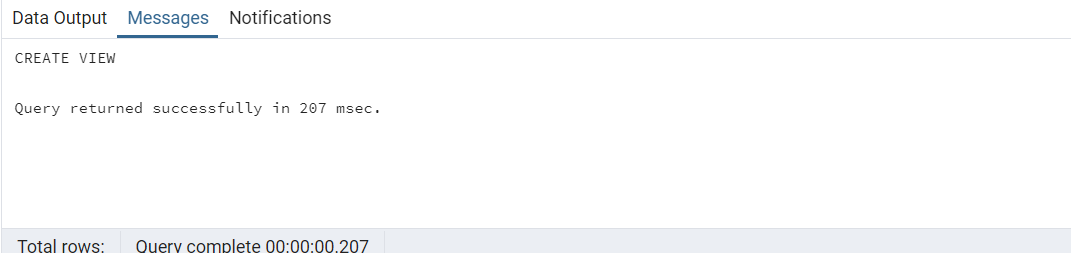
FROM

EMPLOYEES E

JOIN EMPLOYEE\_TERRITORIES ET ON E.EMPLOYEE\_ID = ET.EMPLOYEE\_ID

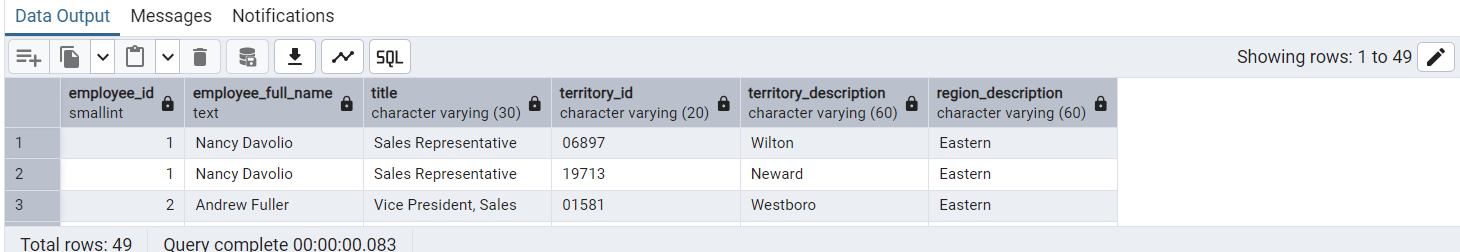
JOIN TERRITORIES T ON T.TERRITORY\_ID = ET.TERRITORY\_ID

JOIN REGION R ON R.REGION\_ID = T.REGION\_ID;



**Output:**

SELECT \* FROM VW\_EMPLOYEE\_TERRITORY\_REGION;



4. Create a recursive CTE based on Employee Hierarchy

**Query:**

WITH RECURSIVE RCTE\_EMPLOYEE AS (

SELECT

EMPLOYEE\_ID,

FIRST\_NAME,

LAST\_NAME,

TITLE,

REPORTS\_TO

FROM

EMPLOYEES

WHERE

REPORTS\_TO IS NULL

UNION

SELECT

E.EMPLOYEE\_ID,

E.FIRST\_NAME,

E.LAST\_NAME,

E.TITLE,

E.REPORTS\_TO

FROM

EMPLOYEES E

JOIN RCTE\_EMPLOYEE RE ON E.REPORTS\_TO = RE.EMPLOYEE\_ID

)

SELECT

EMPLOYEE\_ID,

FIRST\_NAME,

LAST\_NAME,

TITLE,

REPORTS\_TO

FROM RCTE\_EMPLOYEE;

**Output:**

