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

--Try updating the view with the 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;

CREATE OR REPLACE VIEW VW\_UPDATABLE\_PRODUCTS AS

SELECT PRODUCT\_ID,

PRODUCT\_NAME,

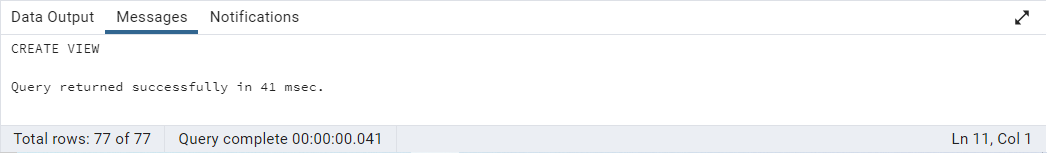
UNIT\_PRICE,

UNITS\_IN\_STOCK,

DISCONTINUED

FROM PRODUCTS

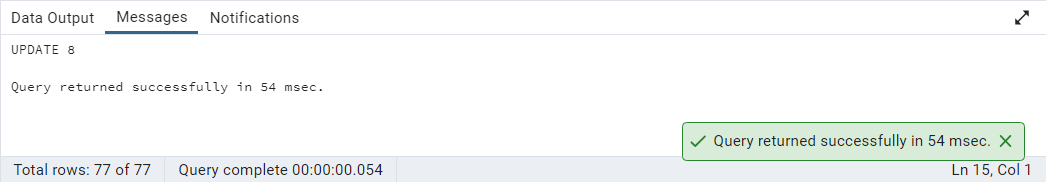
WHERE DISCONTINUED = 0 WITH CHECK OPTION



UPDATE VW\_UPDATABLE\_PRODUCTS

SET UNIT\_PRICE = UNIT\_PRICE \* 1.1

WHERE UNITS\_IN\_STOCK < 10



SELECT PRODUCT\_ID,

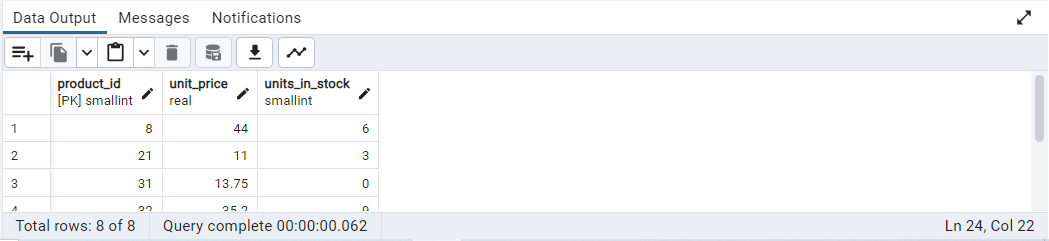
UNIT\_PRICE,

UNITS\_IN\_STOCK

FROM PRODUCTS

WHERE UNITS\_IN\_STOCK < 10

AND DISCONTINUED = 0



--2.Transaction:

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

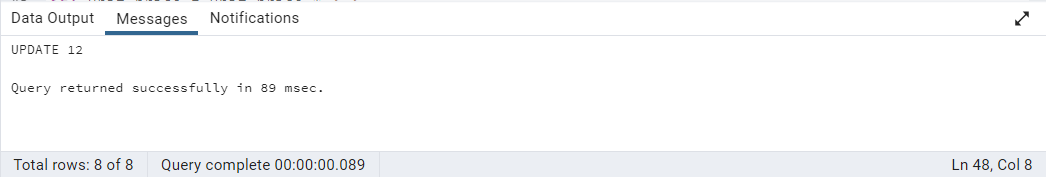
--Try COMMIT and ROLLBACK and observe what happens.

BEGIN;

UPDATE PRODUCTS

SET UNIT\_PRICE = UNIT\_PRICE \* 1.1

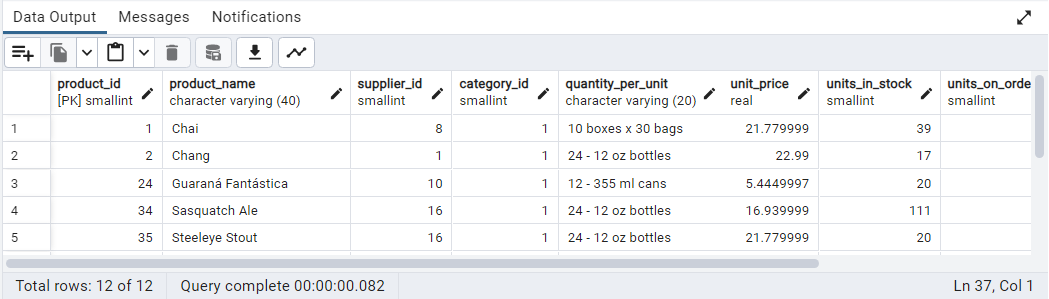
WHERE CATEGORY\_ID = 1



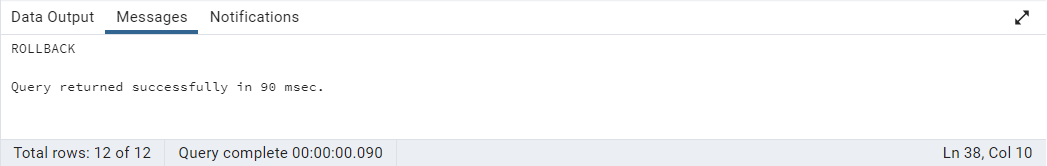
SELECT \*

FROM PRODUCTS

WHERE CATEGORY\_ID = 1



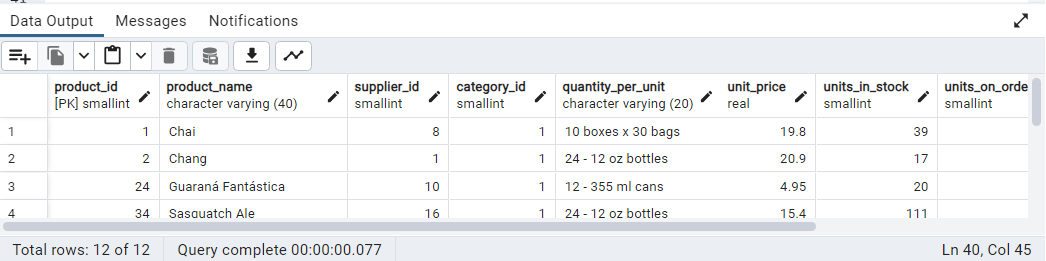
ROLLBACK;



SELECT \*

FROM PRODUCTS

WHERE CATEGORY\_ID = 1

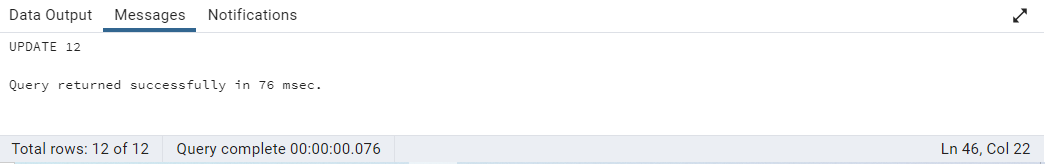


BEGIN;

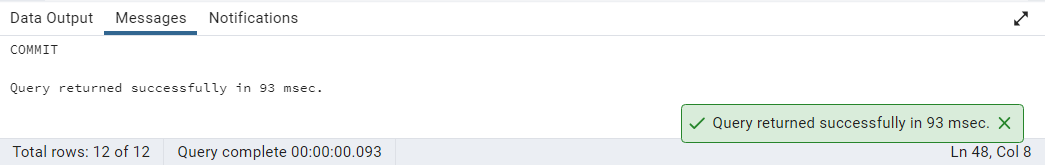
UPDATE PRODUCTS

SET UNIT\_PRICE = UNIT\_PRICE \* 1.1

WHERE CATEGORY\_ID = 1



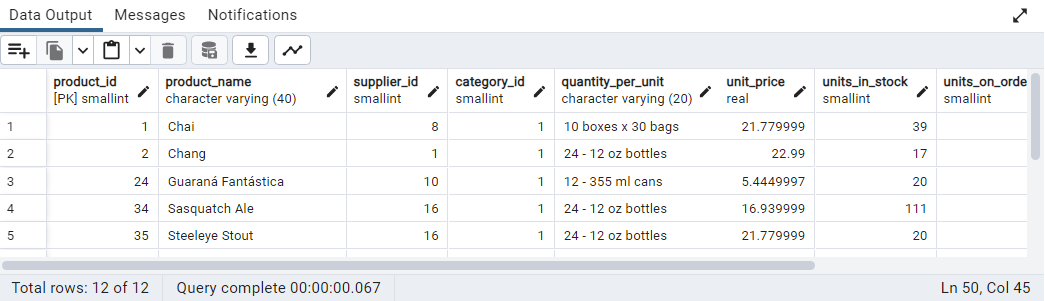
COMMIT;



SELECT \*

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

CREATE VIEW VW\_EMPLOYEE\_DETAILS AS

SELECT E.EMPLOYEE\_ID,

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

E.TITLE,

ET.TERRITORY\_ID,

T.TERRITORY\_DESCRIPTION,

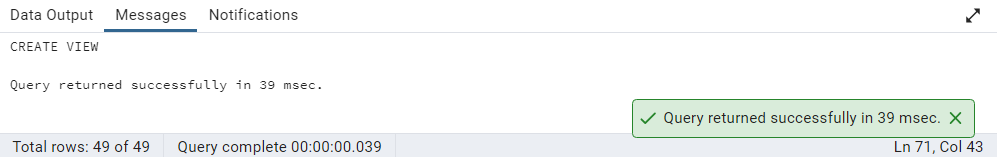
R.REGION\_DESCRIPTION

FROM EMPLOYEES E

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

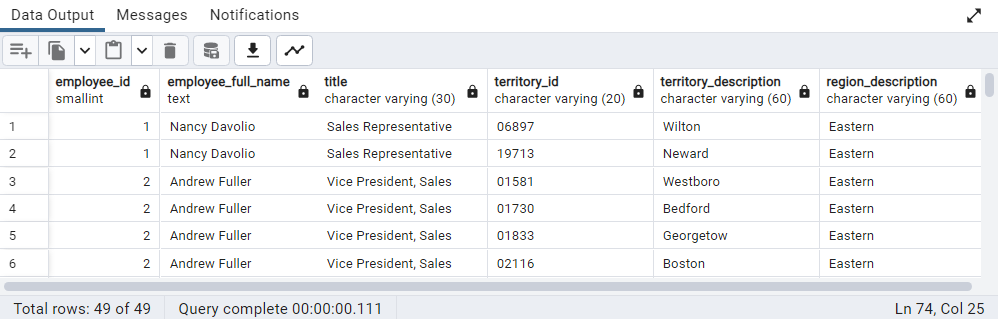
JOIN TERRITORIES T ON ET.TERRITORY\_ID = T.TERRITORY\_ID

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



SELECT \*

FROM VW\_EMPLOYEE\_DETAILS



--4.Create a recursive CTE based on Employee Hierarchy

WITH RECURSIVE CTE\_EMPLOYEE\_HIERARCHY AS

(SELECT EMPLOYEE\_ID,

FIRST\_NAME,

LAST\_NAME,

REPORTS\_TO,

0 AS LEVEL

FROM EMPLOYEES E

WHERE REPORTS\_TO IS NULL

UNION ALL SELECT E.EMPLOYEE\_ID,

E.FIRST\_NAME,

E.LAST\_NAME,

E.REPORTS\_TO,

EH.LEVEL + 1

FROM EMPLOYEES E

JOIN CTE\_EMPLOYEE\_HIERARCHY EH ON EH.EMPLOYEE\_ID = E.REPORTS\_TO)

SELECT LEVEL,

EMPLOYEE\_ID,

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

FROM CTE\_EMPLOYEE\_HIERARCHY

ORDER BY LEVEL

