**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;

**Ans**:

CREATE VIEW vw\_updatable\_products AS

SELECT

product\_id,

product\_name,

unit\_price,

units\_in\_stock,

discontinued

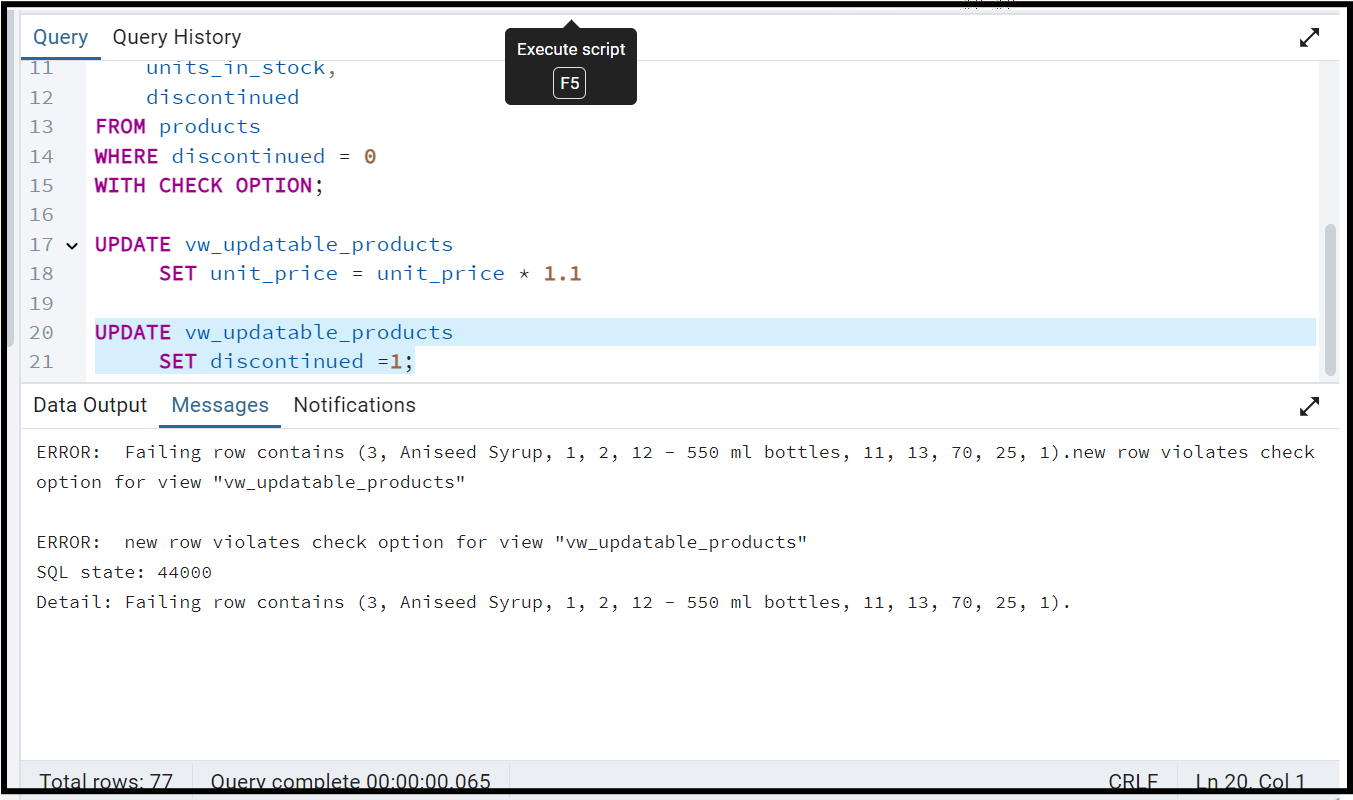
FROM products

WHERE discontinued = 0

WITH CHECK OPTION;







2. Transaction:

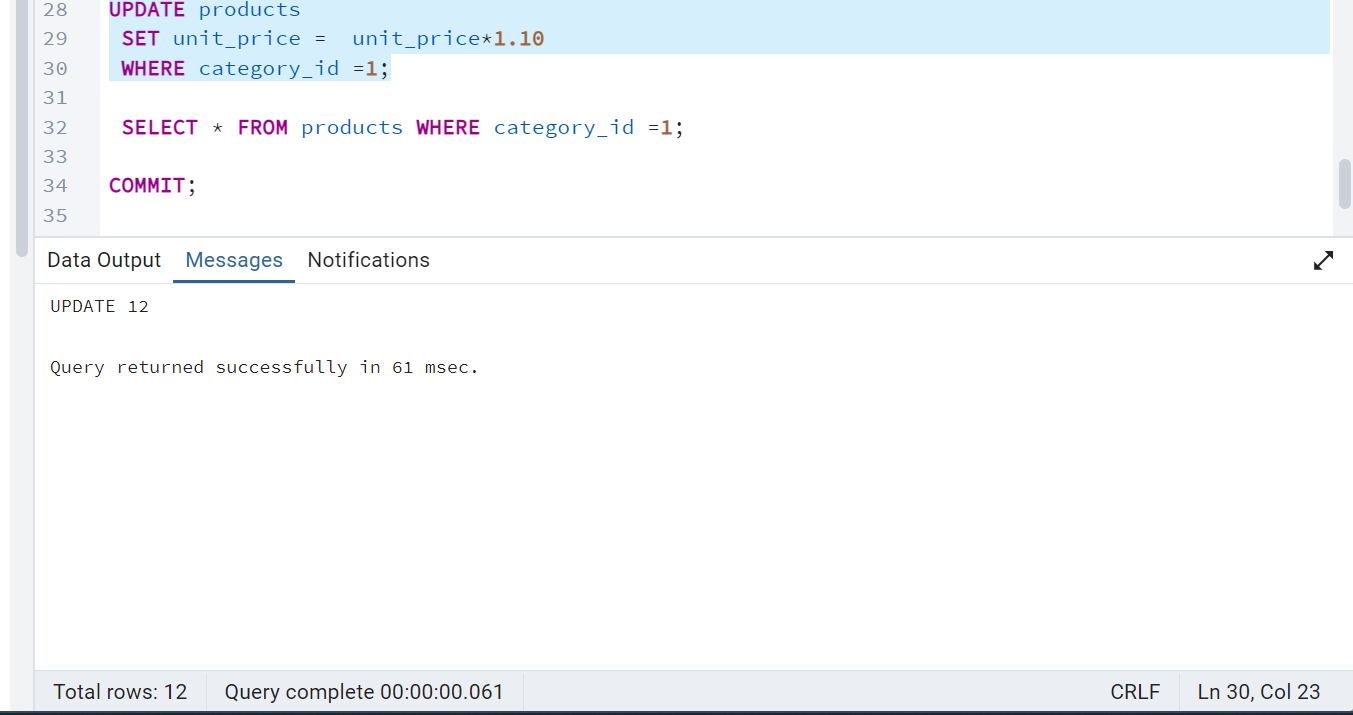
Update the product price for products by 10% in category id=1

Try COMMIT and ROLLBACK and observe what happens.

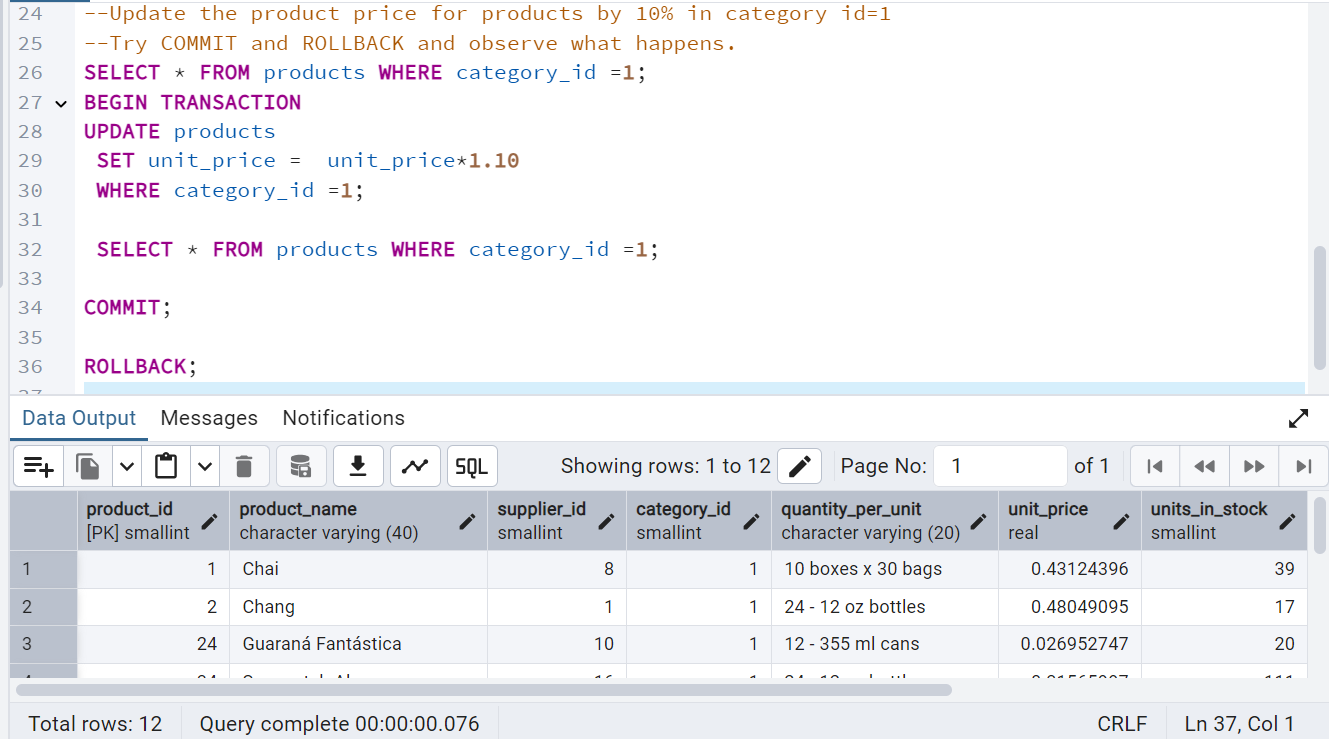
UPDATE products

SET unit\_price = unit\_price\*1.10

WHERE category\_id =1;



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

Ans:

CREATE VIEW vw\_employee\_territories\_region AS

SELECT e.employee\_id,

e.first\_name ||' '||e.last\_name AS Employee\_full\_name,

e.title,

et.territory\_id,

t.territory\_description,

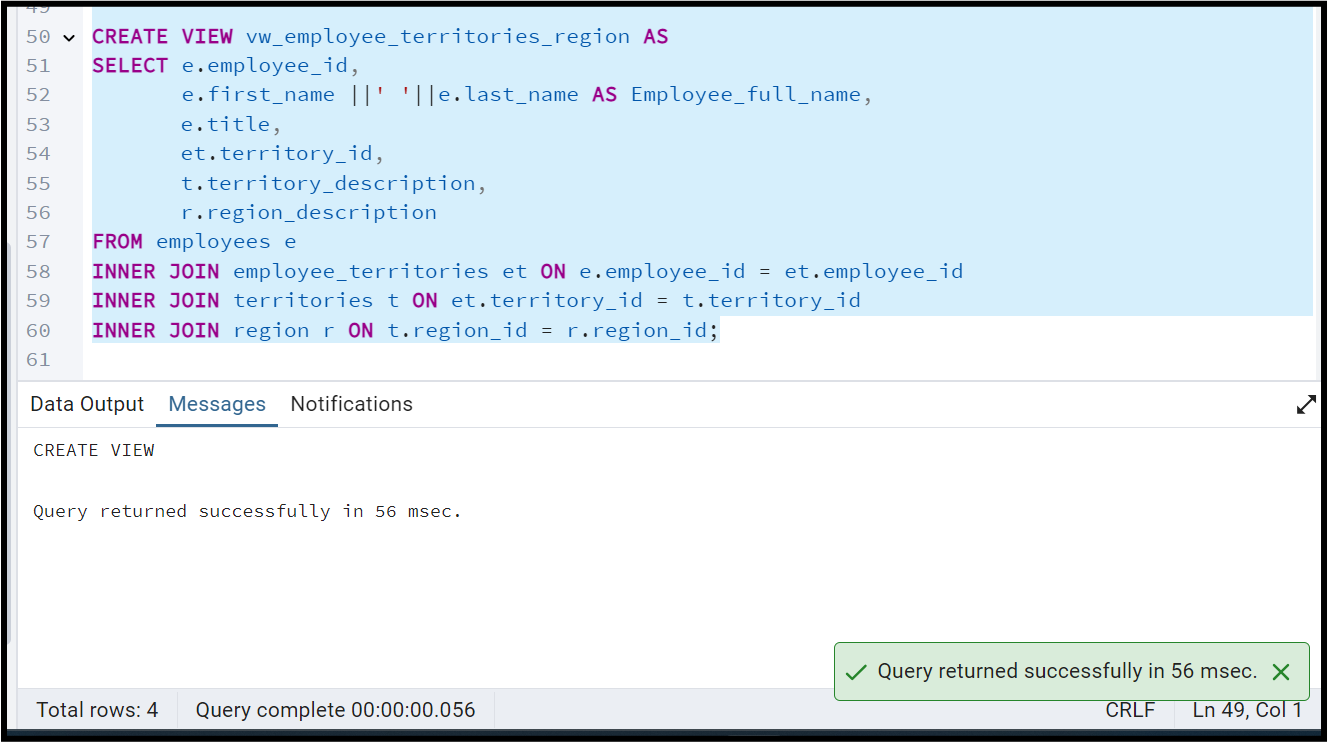
r.region\_description

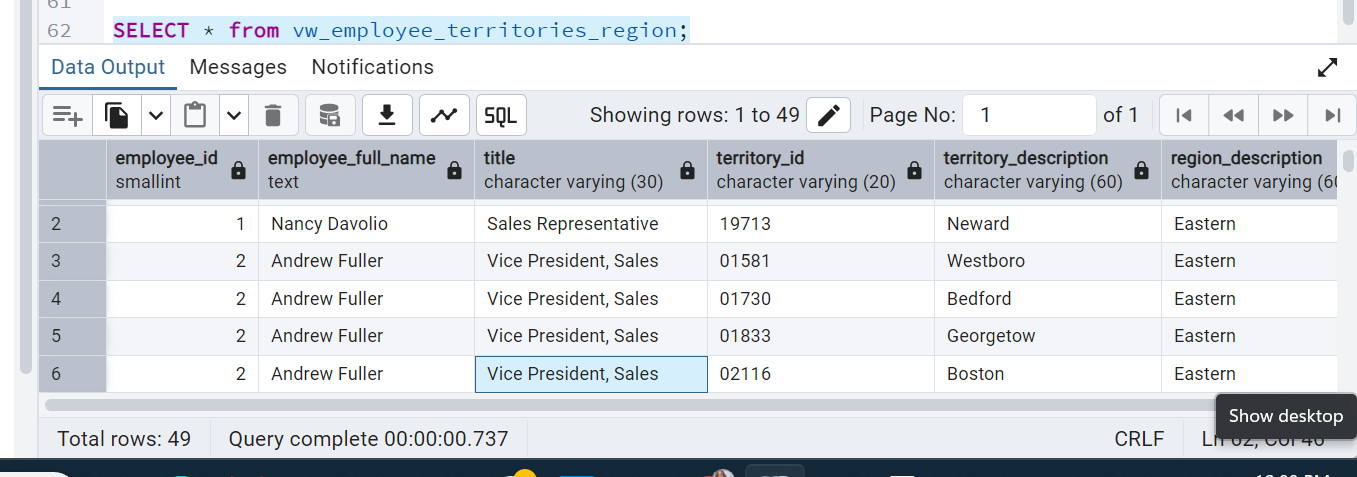
FROM employees e

INNER JOIN employee\_territories et ON e.employee\_id = et.employee\_id

INNER JOIN territories t ON et.territory\_id = t.territory\_id

INNER JOIN region r ON t.region\_id = r.region\_id;





4. Create a recursive CTE based on Employee Hierarchy

**Ans**:

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 \* FROM cte\_employee\_hierarchy

ORDER BY level;

