**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;**

SELECT \* FROM products WHERE units\_in\_stock<10

CREATE VIEW vw\_updatable\_products AS

SELECT

product\_id,

product\_name,

unit\_price,

units\_in\_stock,

discontinued

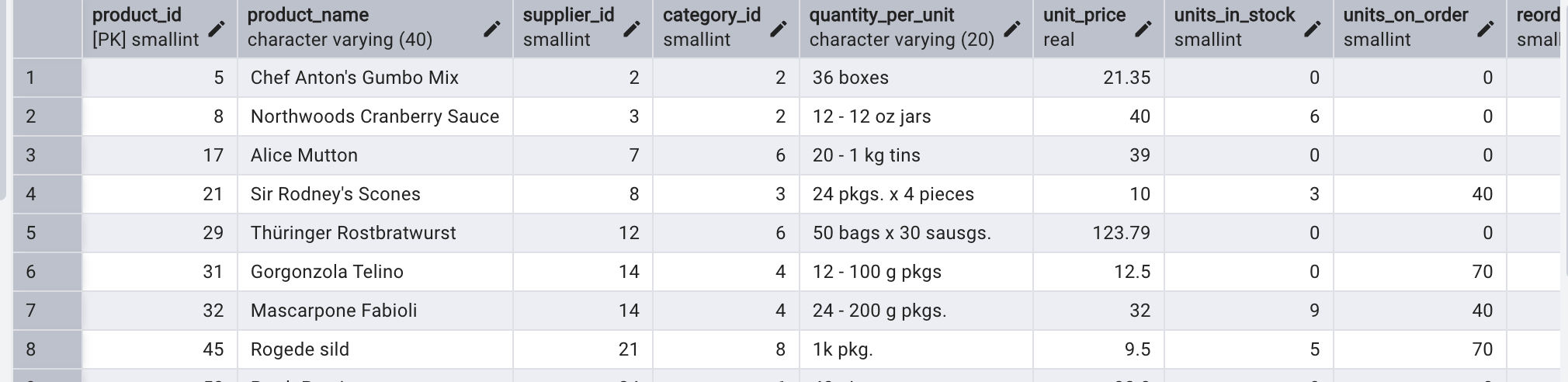
FROM products

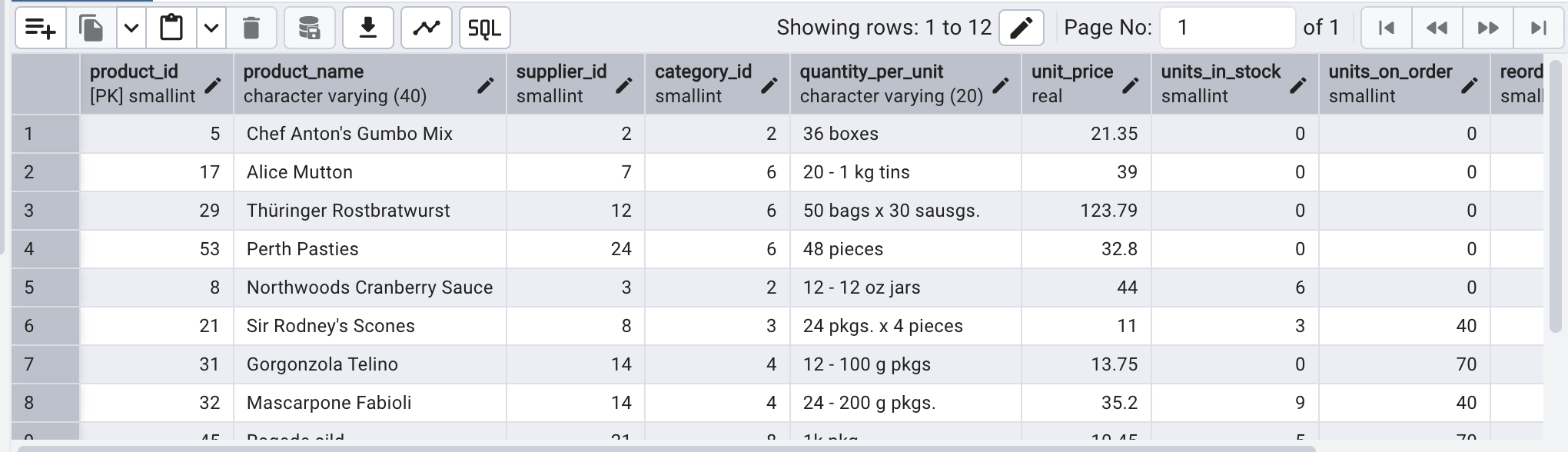
WHERE discontinued=0

WITH CHECK OPTION;

UPDATE INTO vw\_updatable\_products

SET unit\_price = unit\_price \*1.1

WHERE units\_in\_stock<10

Note: After update, unit\_price gets updated for 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 \* 1.10

WHERE category\_id =1

DO$$

BEGIN

IF EXISTS(

SELECT 1

FROM products

WHERE category\_id = 1 AND unit\_price>50

) THEN

RAISE EXCEPTION 'some prices exceed $50';

ELSE

RAISE NOTICE 'price update successful';

END IF;

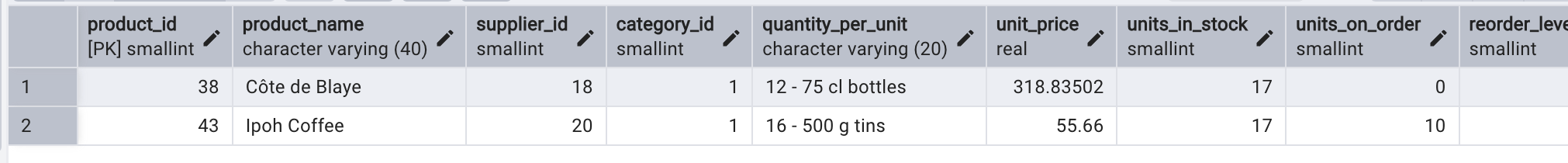
END $$;

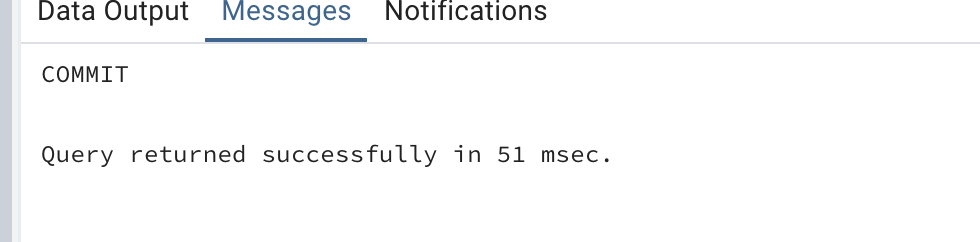
COMMIT;

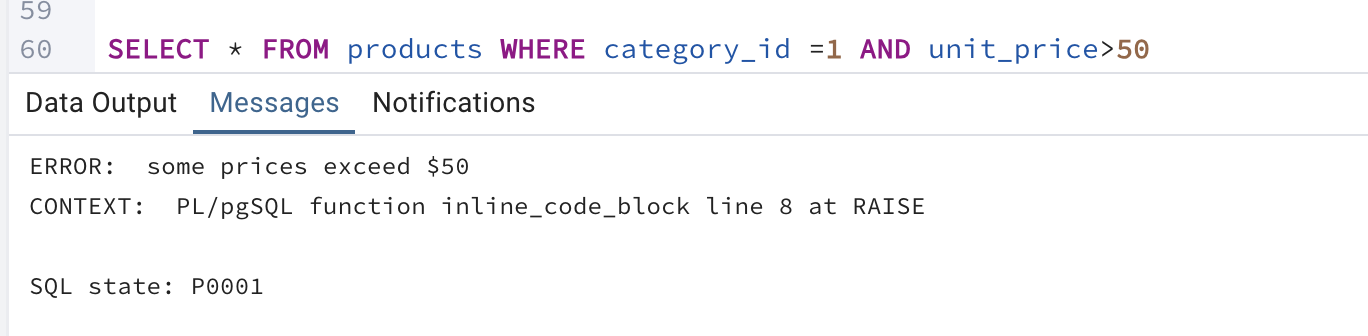
ROLLBACK;

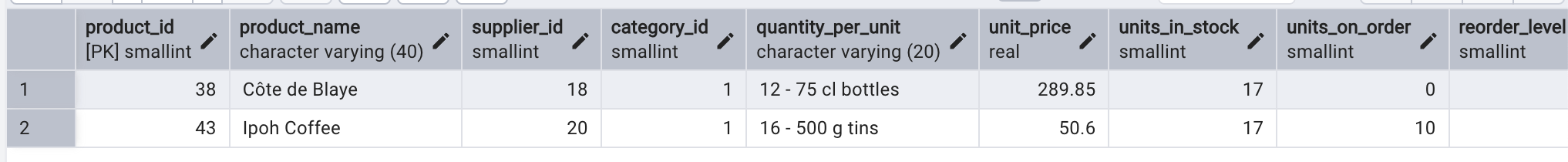
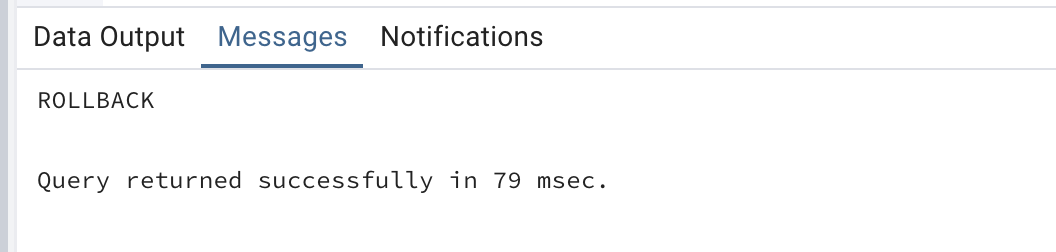
SELECT \* FROM products WHERE category\_id =1 AND

unit\_price>50









**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 full\_name,e.title,

t.territory\_id, t.territory\_description,r.region\_description

FROM employees e

JOIN employee\_territories et

ON e.employee\_id = et.employee\_id

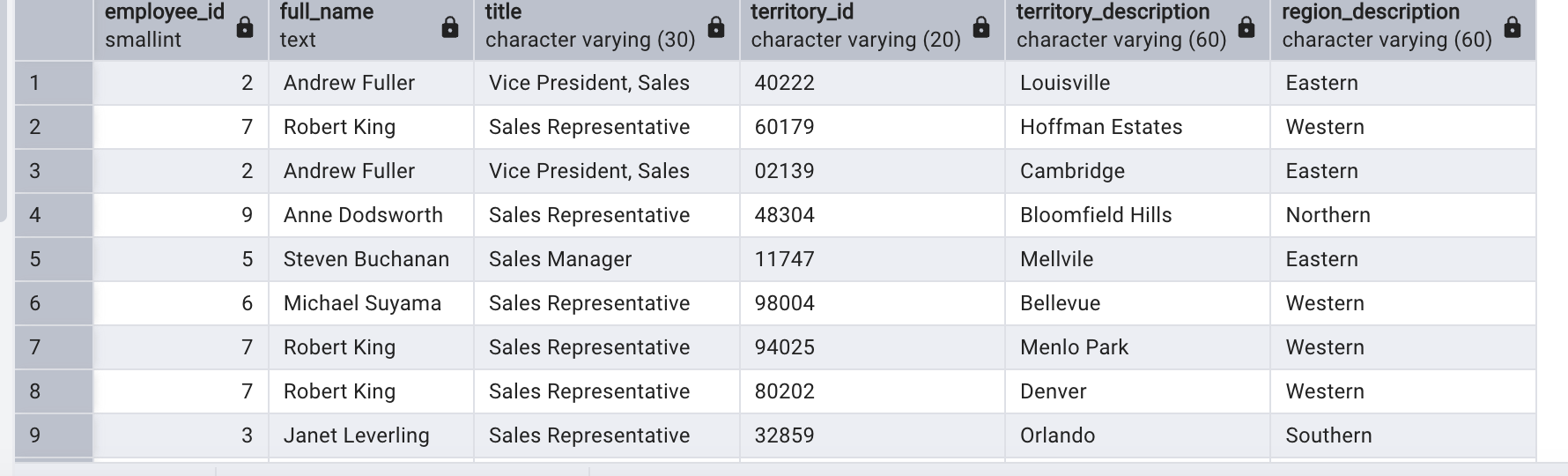
JOIN territories t

ON et.territory\_id = t.territory\_id

JOIN region r

ON t.region\_id = r.region\_id

GROUP BY e.employee\_id,t.territory\_id, t.territory\_description,r.region\_description

SELECT \* FROM vw\_employee\_details

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

WITH RECURSIVE cte\_employeehierarchy 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\_employeehierarchy eh

ON

eh.employee\_id = e.reports\_to

)

SELECT level,employee\_id,first\_name||' '||last\_name AS full\_name

FROM

cte\_employeehierarchy

ORDER BY level,employee\_id;