## **Day 8 Assignment**

1. Create view vw\_updatable\_products

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;

Step1: Create a view table

From products;

```
10 v /* 1.Create view vw_updatable_products.
11
     Try updating the view with the below query and see
12
     Update query:
13 UPDATE updatable_products
14
     SET unit_price = unit_price * 1.1
15
     WHERE units_in_stock < 10;
16
     */
17
18 ∨ Create View vw_updatable_products As
     Select product_id,
19
20
             product_name,
21
             unit_price,
22
             units_in_stock
23
     From products;
Data Output Messages Notifications
CREATE VIEW
Query returned successfully in 52 msec.
```

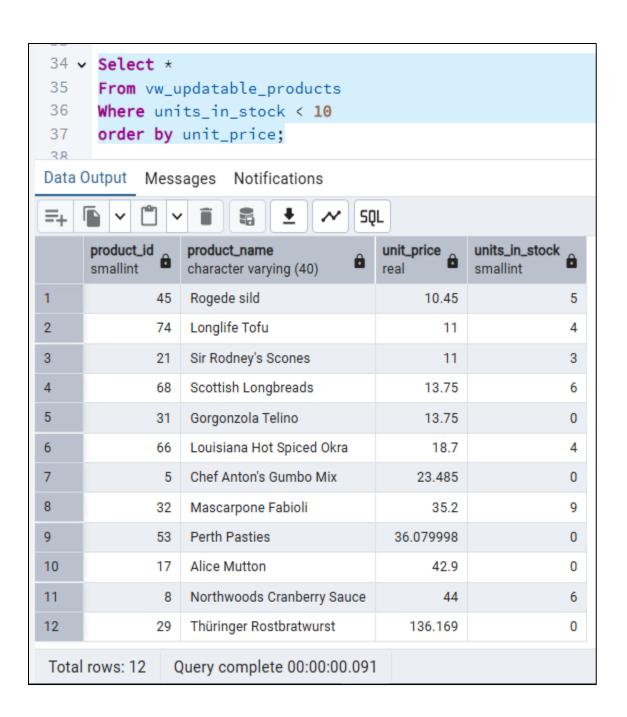
Step 2: Update the view table created

Update vw\_updatable\_products
Set unit\_price = unit\_price \* 1.1
Where units\_in\_stock < 10;</pre>

```
18 v Create View vw_updatable_products As
      Select product_id,
19
20
              product_name,
21
              unit_price,
              units_in_stock
22
23 From products;
24
25 v Update vw_updatable_products
26
      Set unit_price = unit_price * 1.1
27
      Where units_in_stock < 10;
28
Data Output Messages Notifications
UPDATE 12
Query returned successfully in 55 msec.
```

Step 3: Check in the view table for update

Select \*
From vw\_updatable\_products
Where units\_in\_stock < 10
order by unit\_price;



Step 4: Check in the products table also for update

Select product\_id,

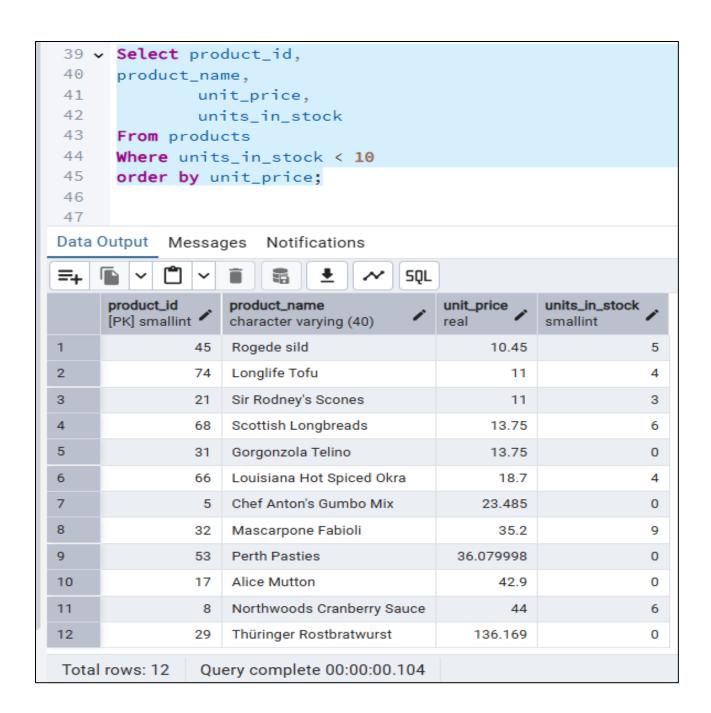
product\_name,

unit\_price,

units\_in\_stock

From products

Where units\_in\_stock < 10 order by unit\_price;



The products also got updated

2. Transaction: Update the product price for products by 10% in category id=1 Try COMMIT and ROLLBACK and observe what happens.

Step1: SELECT \* FROM products WHERE category\_id = 1 order by unit\_price;

30 31 32 33 34	Try COMMIT	product price for p and ROLLBACK and ob * from products; ROM products WHERE o	serve what	happens.*/						
	product_id [PK] smallint	product_name character varying (40)	supplier_id smallint	category_id smallint	quantity_per_unit character varying (20)	unit_price real	units_in_stock smallint	units_on_order smallint	reorder_level smallint	discontinued integer
1	24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.5	20	0	0	1
2	75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7.75	125	0	25	0
3	67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14	52	0	10	0
4	34	Sasquatch Ale	16	1	24 - 12 oz bottles	14	111	0	15	0
5	70	Outback Lager	7	1	24 - 355 ml bottles	15	15	10	30	0
6	76	Lakkalikööri	23	1	500 ml	18	57	0	20	0
7	35	Steeleye Stout	16	1	24 - 12 oz bottles	18	20	0	15	0
8	39	Chartreuse verte	18	1	750 cc per bottle	18	69	0	5	0
9	1	Chai	8	1	10 boxes x 30 bags	18	39	0	10	1
10	2	Chang	1	1	24 - 12 oz bottles	19	17	40	25	1
11	43	Ipoh Coffee	20	1	16 - 500 g tins	46	17	10	25	0
12	38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5	17	0	15	0

## Step 2: Begin;

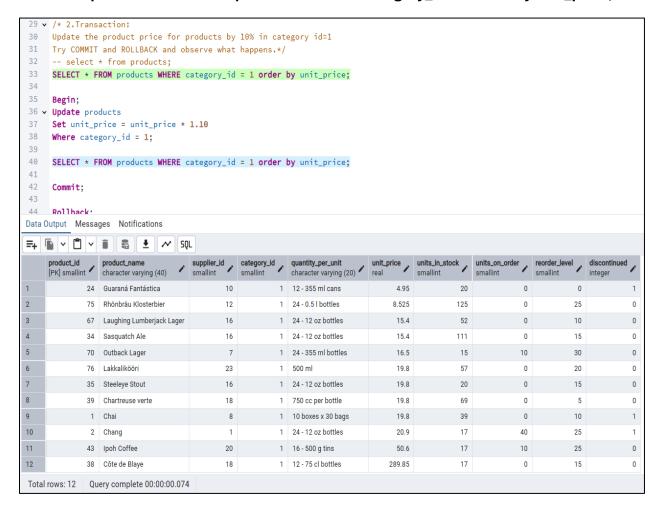
**Update products** 

Set unit\_price = unit\_price \* 1.10

Where category\_id = 1;

```
29 v /* 2.Transaction:
30 Update the product price for products by 10% in category id=1
31 Try COMMIT and ROLLBACK and observe what happens.*/
32 -- select * from products;
33
     SELECT * FROM products WHERE category_id = 1 order by unit_price;
34
35 Begin;
36 v Update products
37
     Set unit_price = unit_price * 1.10
     Where category_id = 1;
38
39
Data Output Messages Notifications
UPDATE 12
Query returned successfully in 81 msec.
```

Step 3: SELECT \* FROM products WHERE category\_id = 1 order by unit\_price;



We can observe the updated prices, but the change is not permanent yet.

## Step 4: Rollback;

```
Rollback:
45
46
47
48 🗸 /* 3.Create a regular view which wil
      Employee_id,
50
      Employee_full_name,
51
      Title,
      Territory_id,
      territory_description,
Data Output
            Messages
                      Notifications
ROLLBACK
Query returned successfully in 78 msec.
```

## Check again: SELECT \* FROM products WHERE category\_id = 1;



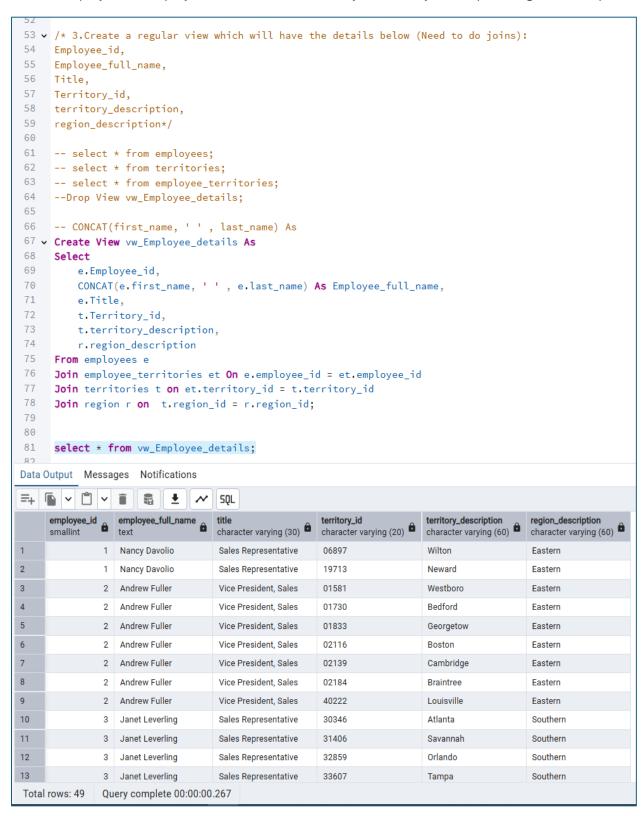
We can see the original prices, since the transaction was rolled back.

Step 5: Commit; Commit makes the updated prices permanent in the database.



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

Employee\_id,Employee\_full\_name,Title,Territory\_id,territory\_description,region\_description



4. Create a recursive CTE based on Employee Hierarchy

```
/* 4.Create a recursive CTE based on Employee Hierarchy*/
35
     --Select employee_id, first_name, reports_to From employees;
37 • With Recursive cte_employee_hierarchy
38
39
     Select
          employee_id,
40
41
          first_name,
42
         last_name,
43
          reports_to,
44
          0 As Level
45
     From employees e
46
     Where reports_to Is Null
47
48
     Union All
49
50
     Select
51
          e.employee_id,
52
          e.first_name,
53
          e.last_name,
54
          e.reports_to,
55
          eh.level+1
56
     From employees e
57
     Join cte_employee_hierarchy eh
58
     On eh.employee_id = e.reports_to
59
60
61
     Select Level, employee_id, concat(first_name, ' ' , last_name) as employee_name
62
     From cte_employee_hierarchy
63
     Order By level, employee_id;
Data Output Messages Notifications
=+
                                     SQL
              employee_id
                           employee_name
      level
     integer 🎰
1
                        2 Andrew Fuller
            0
2
                           Nancy Davolio
            1
3
                           Janet Leverling
4
            1
                           Margaret Peacock
5
                        5 Steven Buchanan
            1
6
            1
                        8 Laura Callahan
            2
7
                           Michael Suyama
                           Robert King
8
            2
9
                            Anne Dodsworth
 Total rows: 9
              Query complete 00:00:00.070
```