**Assignment 8**

**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;

-- To create view

CREATE VIEW vw\_updatable\_products AS

SELECT product\_id, product\_name, unit\_price, units\_in\_stock

FROM products

WHERE discontinued = 0;

SELECT \* FROM products;

-- To update the view

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

SELECT \* FROM products;

A screenshot of a computer

AI-generated content may be incorrect.

After Updating the view

A screenshot of a computer

AI-generated content may be incorrect.

 -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2.     Transaction:

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

Try COMMIT and ROLLBACK and observe what happens.

-- To check the current price

SELECT product\_id,

product\_name,

unit\_price

FROM products

WHERE category\_id = 1;

-- To Commit transaction and apply the 10% increase

BEGIN TRANSACTION;

-- To apply the 10% price increase

UPDATE products

SET unit\_price = unit\_price \* 1.10

WHERE category\_id = 1;

-- To verify uncommitted changes in your session

SELECT product\_id,

product\_name,

unit\_price

FROM products

WHERE category\_id = 1;

-- To commit the transaction to make changes permanent

COMMIT;

-- To Rollback the transaction

BEGIN TRANSACTION;

-- To apply the 10% price increase

UPDATE products

SET unit\_price = unit\_price \* 1.10

WHERE category\_id = 1;

-- To verify uncommitted changes in your session

SELECT product\_id,

product\_name,

unit\_price

FROM products

WHERE category\_id = 1;

-- To rollback the transaction to undo the update

ROLLBACK;

-- To confirm that prices are back to their original values

SELECT product\_id,

product\_name,

unit\_price

FROM products

WHERE category\_id = 1;

A screenshot of a computer

AI-generated content may be incorrect.

--After increasing price by 10 %

A screenshot of a computer

AI-generated content may be incorrect.

--Rollback the transaction

A screenshot of a computer

AI-generated content may be incorrect.

 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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\_territory\_details AS

SELECT

e.employee\_id,

e.first\_name || ' ' || e.last\_name AS employee\_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;

SELECT \*

FROM vw\_employee\_territory\_details;

A screenshot of a computer

AI-generated content may be incorrect.

 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.     Create a recursive CTE based on Employee Hierarchy

WITH RECURSIVE employee\_hierarchy AS (

-- Anchor: top‐level employees (no manager)

SELECT

e.employee\_id,

e.first\_name || ' ' || e.last\_name AS employee\_name,

e.title,

e.reports\_to AS manager\_id,

NULL::text AS manager\_name,

1 AS level

FROM employees e

WHERE e.reports\_to IS NULL

UNION ALL

-- Recursive step: find direct reports of everyone in the CTE so far

SELECT

e.employee\_id,

e.first\_name || ' ' || e.last\_name,

e.title,

e.reports\_to,

eh.employee\_name,

eh.level + 1

FROM employees e

JOIN employee\_hierarchy eh

ON e.reports\_to = eh.employee\_id

)

SELECT

employee\_id,

employee\_name,

title,

manager\_id,

manager\_name,

level

FROM employee\_hierarchy

ORDER BY level, manager\_name NULLS FIRST, employee\_name;

A screenshot of a computer

AI-generated content may be incorrect.

 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------