--####################################################################--

-- 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;\*/

-- Creating View

CREATE VIEW vw\_updatable\_products AS

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

FROM products WHERE discontinued=0

--Updating View

UPDATE vw\_updatable\_products

SET unit\_price = unit\_price \* 1.1

WHERE units\_in\_stock < 10;

--Cross verifying chnages applied

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

FROM products WHERE discontinued=0

AND units\_in\_stock < 10;

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.\*/

-- Start transaction

BEGIN;

A screenshot of a computer

AI-generated content may be incorrect.

-- Updating

UPDATE products

SET unit\_price = unit\_price \* 1.10

WHERE category\_id = 1;

A screenshot of a computer

AI-generated content may be incorrect.

-- cross checking

SELECT product\_id, product\_name, unit\_price

FROM products

WHERE category\_id = 1;

A screenshot of a computer

AI-generated content may be incorrect.

-- Commit Changes

COMMIT;

A screenshot of a computer

AI-generated content may be incorrect.

-- Revert the changes

ROLLBACK;

A screenshot of a computer

AI-generated content may be incorrect.

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

CREATE VIEW vw\_employee\_territories 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;

A screenshot of a computer screen

AI-generated content may be incorrect.

-- CHECK

SELECT \* FROM vw\_employee\_territories;

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 (

SELECT

employee\_id,

first\_name || ' ' || last\_name AS full\_name,

reports\_to AS manager\_id,

1 AS level

FROM employees WHERE reports\_to IS NULL -- Top-level managers

UNION ALL

SELECT

e.employee\_id,

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

e.reports\_to,

eh.level + 1

FROM employees e

INNER JOIN employee\_hierarchy eh ON e.reports\_to = eh.employee\_id

)

SELECT

employee\_id,

full\_name,

manager\_id,

level

FROM employee\_hierarchy

ORDER BY level, manager\_id;

A screenshot of a computer

AI-generated content may be incorrect.