DAY 8 ASSIGNMENT

-- 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;

create view vw\_updatable\_products AS

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

from products

where unit\_price >0

with check option;

drop view vw\_updatable\_products

select \* from products

select \* from vw\_updatable\_products

Update vw\_updatable\_products

SET unit\_price =unit\_price\*1.1

WHERE 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.

SELECT \* FROM PRODUCTS where category\_id =3

A screenshot of a computer

AI-generated content may be incorrect.

BEGIN

UPDATE PRODUCTS

SET UNIT\_PRICE = UNIT\_PRICE\*0.1

WHERE CATEGORY\_ID =3

A screenshot of a computer

AI-generated content may be incorrect.

UPDATE PRODUCTS

SET UNIT\_PRICE = UNIT\_PRICE\*0.1

WHERE CATEGORY\_ID =2

COMMIT;

rollback;

-- 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

select \* from employees

select \* from employee\_territories

select \* from territories

select \* from region

create view vw\_employee\_territories AS

select e.employee\_id, first\_name || ' ' || last\_name as employee\_full\_name , title, t.territory\_id,territory\_description,region\_description from employees e

inner join employee\_territories et on

e.employee\_id = et.employee\_id

inner join territories t on

t.territory\_id = et.territory\_id

inner join region r on

r.region\_id = t.region\_id

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 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 employee\_name

from cte\_employeehierarchy

order by

level,employee\_id; A screenshot of a computer

AI-generated content may be incorrect.