--Day 9

--1.Create AFTER UPDATE trigger to track product price changes

--·Create product\_price\_audit table with below columns:

Create Table product\_price\_audit

(audit\_id SERIAL PRIMARY KEY,

product\_id INT,

product\_name VARCHAR(40),

old\_price DECIMAL(10,2),

new\_price DECIMAL(10,2),

change\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_name VARCHAR(50) DEFAULT CURRENT\_USER

)

-----------Creating function----------

Create Or Replace Function product\_price\_update()

Returns Trigger

As $$

Begin

Insert Into product\_price\_audit (

product\_id,

product\_name,

old\_price,

new\_price

)

Values (

OLD.product\_id,

OLD.product\_name,

OLD.unit\_price,

NEW.unit\_price

);

Return New;

End;

$$ Language plpgsql;

---------Creating trigger----

Create Trigger after\_product\_update

After Update Of unit\_price On products

For Each Row

Execute Function product\_price\_update();

-----testing the trigger-updating products price---

Update products

Set unit\_price = unit\_price \* 1.10

WHERE product\_id = 9;

----view the product\_price\_audit table---

Select \* From product\_price\_audit

Order By change\_date Desc;

Select \* from public.products

Where product\_id = 9;

A screenshot of a computer

AI-generated content may be incorrect.

--2.Create stored procedure using IN and INOUT parameters to assign tasks to employees

--Parameters:

-----------Creating table--------------

Create Table If Not Exists employee\_tasks (

task\_id SERIAL Primary Key,

employee\_id INT,

task\_name VARCHAR(50),

assigned\_date DATE Default current\_date

)

-----------Creating store procedure------------

-----------Insert employee\_id, task\_name into employee\_tasks------

Create Or Replace Procedure employee\_tasks(

In p\_employee\_id INT,

In p\_task\_name VARCHAR(50),

Inout p\_task\_count INT Default 0

)

Language plpgsql

As $$

Begin

---------Insert task assignment-------------------

INSERT INTO employee\_tasks (employee\_id, task\_name)

VALUES (p\_employee\_id, p\_task\_name);

----------p\_task\_name, p\_employee\_id, p\_task\_count-------------

---------Count total tasks for employee and put the total count into p\_task\_count------

Select Count(\*) Into p\_task\_count

FROM employee\_tasks

WHERE employee\_id = p\_employee\_id;

---------RAISE NOTICE 'Task "%" assigned to employee %. Total tasks: %',----

Raise Notice 'Task "%" assigned to employee %. Total tasks: %',

p\_task\_name, p\_employee\_id, p\_task\_count;

End;

$$;

----------Calling the procedure------------

Call employee\_tasks(1, 'Review Reports', 0);

-------Verify the task was inserted---------

Select \* From employee\_tasks

Where employee\_id = 1;

A screenshot of a computer

AI-generated content may be incorrect.