DAY 9 ASSIGNMENT

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

   Step 1: Create product_price_audit table with below columns.

```
CREATE TABLE IF NOT EXISTS 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
);
```

```
/*1. Create AFTER UPDATE trigger to track product price changes.*/

--> Step 1: Create product_price_audit table with below columns

CREATE TABLE IF NOT EXISTS 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
);
```

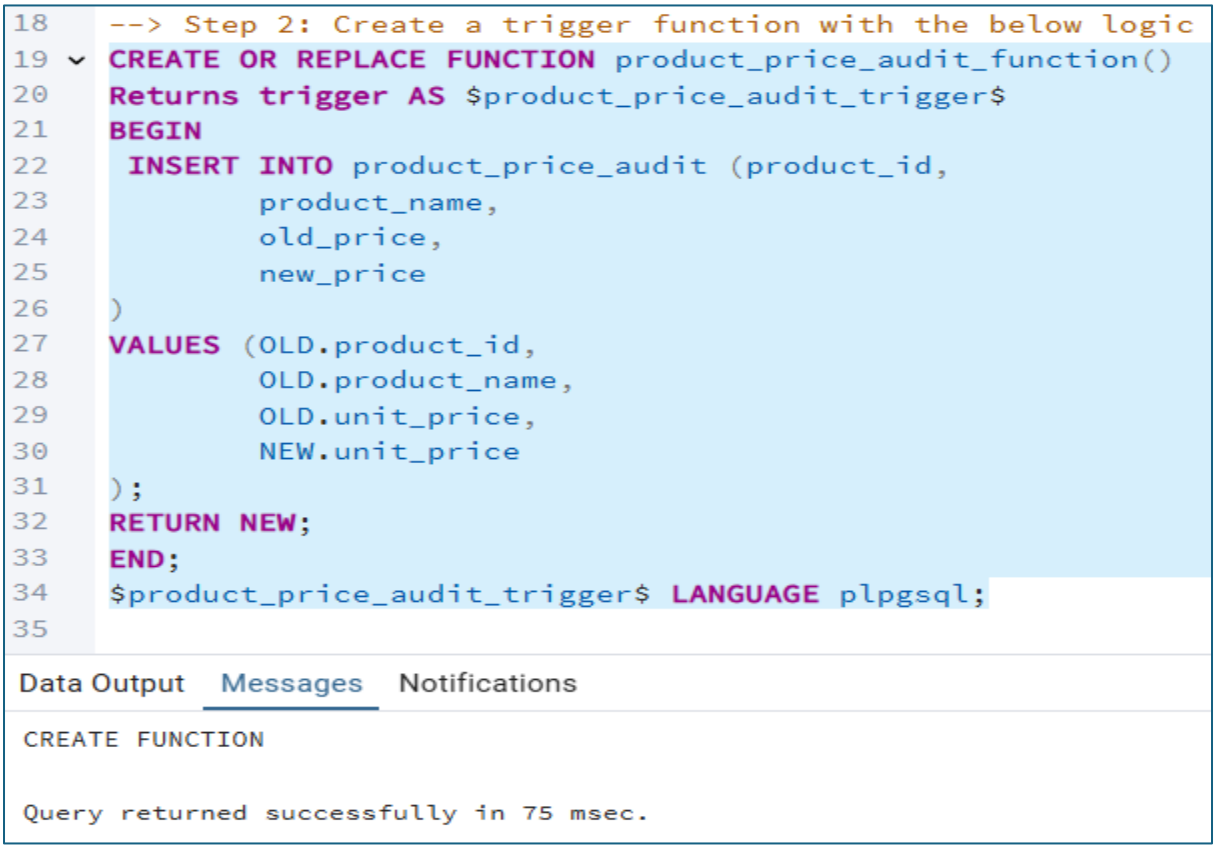
Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 66 msec.
```

Step 2: Create a trigger function with the below logic.

```
CREATE OR REPLACE FUNCTION product_price_audit_function()
Returns trigger AS $product_price_audit_trigger$
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;
$product_price_audit_trigger$ LANGUAGE plpgsql;
```



Step 3: Create a row level trigger for the event below.

CREATE TRIGGER product_price_audit_trigger
AFTER UPDATE OF unit_price ON products
FOR EACH ROW
EXECUTE FUNCTION product_price_audit_function();

```
37  --> Step 3: Create a row level trigger for the event below.
38  CREATE TRIGGER product_price_audit_trigger
39  AFTER UPDATE OF unit_price ON products
40  FOR EACH ROW
41  EXECUTE FUNCTION product_price_audit_function();
42
```

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 59 msec.
```

Step 4: Test the trigger by updating the product price by 10% to any one product_id.

Check the unit_price current value for product_id = 1

Select * from products WHERE product_id = 1 ;

```
44  select * from products WHERE product_id = 1 ;
45
```

Data Output | Messages | Notifications

	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	1	Chai	8	1	10 boxes x 30 bags	19.8	39	0	10	1

Check the audit table-----EMPTY table

select * from product_price_audit;

```
47  -- check the audit table
48  select * from product_price_audit;
49
```

Data Output | Messages | Notifications

audit_id [PK] integer	product_id integer	product_name character varying (40)	old_price numeric (10,2)	new_price numeric (10,2)	change_date timestamp without time zone	user_name character varying (50)

Now update the unit_price for product_id =1

UPDATE products
SET unit_price = unit_price * 1.10
WHERE product_id = 1 ;

```
50  --Update the products table unit_price for product id =1
51  UPDATE products
52  SET unit_price = unit_price * 1.10
53  WHERE product_id = 1 ;
54
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 78 msec.
```

Now check the products table for update

select * from products WHERE product_id = 1 ;

```
56  -- Now check the products table for update
57  select * from products WHERE product_id = 1 ;
58
```

Data Output | Messages | Notifications

	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	1	Chai	8	1	10 boxes x 30 bags	21.779999	39	0	10	1

Now check the audit table also for updates

select * from product_price_audit;

```
60  -- Now check the audit table also for updates
61  select * from product_price_audit;
62
```

Data Output | Messages | Notifications

	audit_id [PK] integer	product_id integer	product_name character varying (40)	old_price numeric (10,2)	new_price numeric (10,2)	change_date timestamp without time zone	user_name character varying (50)
1	1	1	Chai	19.80	21.78	2025-05-03 07:59:25.320412	postgres

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

Step 1: Create table employee_tasks

```
69
70  --> Step 1: Create table employee_tasks
71  CREATE TABLE IF NOT EXISTS employee_tasks (
72          task_id SERIAL PRIMARY KEY,
73          employee_id INT,
74          task_name VARCHAR(50),
75          assigned_date DATE DEFAULT CURRENT_DATE
76      );
77
78
79
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 65 msec.
```

Step 2: Create a Stored Procedure

```
--> Step 2: Create a Stored Procedure

CREATE OR REPLACE PROCEDURE assign_task (
IN p_employee_id INT,
IN p_task_name VARCHAR(50),
INOUT p_task_count INT DEFAULT 0
)
LANGUAGE plpgsql
AS $$
BEGIN
-- Step 1: Insert a new task for the employee
    INSERT INTO employee_tasks (employee_id, task_name)
    VALUES (p_employee_id, p_task_name);

    -- Step 2: Count total tasks for the employee and assign to INOUT parameter
    SELECT COUNT(*) INTO p_task_count
    FROM employee_tasks
    WHERE employee_id = p_employee_id;

    -- Step 3: Raise NOTICE message
    RAISE NOTICE 'Task "%" assigned to employee %. Total tasks: %',
        p_task_name, p_employee_id, p_task_count;
END;
$$;
```

Data Output | Messages | Notifications

```
CREATE PROCEDURE

Query returned successfully in 94 msec.
```

Step 3: Call the Stored Procedure

CALL assign_task(1, 'Review Reports');

```
--> Step 3: Call the Stored Procedure
CALL assign_task(1, 'Review Reports');
```

Data Output | Messages | Notifications

	p_task_count integer
1	1

You should see the entry in employee_tasks table.

```
110  --> You should see the entry in employee_tasks table.
111  SELECT * FROM employee_tasks;
112
113
114
```

Data Output | Messages | Notifications

SQL

	task_id [PK] integer	employee_id integer	task_name character varying (50)	assigned_date date
1	1	1	Review Reports	2025-05-03