Day9\_PavithraMani\_SDET177

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

-- Create the product\_price\_audit Table

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

);

--Create the Trigger Function

CREATE

OR REPLACE FUNCTION TRACK\_PRICE\_CHANGES () 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;

-- Create the Trigger on products Table

CREATE TRIGGER TRG\_AFTER\_PRICE\_UPDATE

AFTER

UPDATE OF UNIT\_PRICE ON PRODUCTS FOR EACH ROW WHEN (OLD.UNIT\_PRICE IS DISTINCT FROM NEW.UNIT\_PRICE)

EXECUTE FUNCTION TRACK\_PRICE\_CHANGES ();

--Update To verify the Trigger

UPDATE PRODUCTS

SET

UNIT\_PRICE = UNIT\_PRICE \* 1.1

WHERE

PRODUCT\_ID = 1;

--check the audit log to verify

SELECT

\*

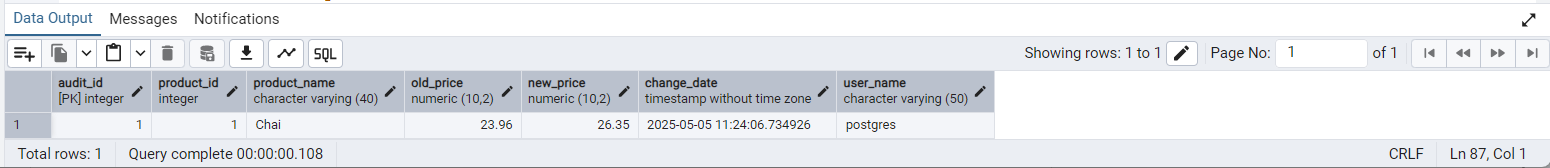
FROM

PRODUCT\_PRICE\_AUDIT

ORDER BY

CHANGE\_DATE DESC;

### OUTPUT



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

-- Create employee\_tasks 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

);

--Create the 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

-- Insert new task for the employee

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

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

-- Count total tasks assigned to the employee

SELECT COUNT(\*) INTO p\_task\_count

FROM employee\_tasks

WHERE employee\_id = p\_employee\_id;

-- Show notice message

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

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

END;

$$;

--Define a variable here and

-- To Call the Procedure to Test

DO $$

DECLARE

task\_total INT := 0;

BEGIN

CALL assign\_task(1, 'Review Reports', task\_total);

RAISE NOTICE 'Returned task count: %', task\_total;

END;

$$;

### OUTPUT

