**Q1) .Create AFTER UPDATE trigger to track product price changes.**

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

**Query:**

--·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),

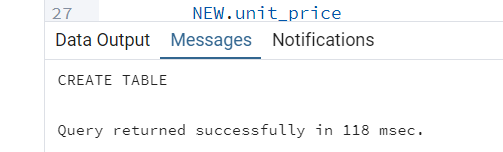
new\_price DECIMAL(10,2),

change\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_name VARCHAR(50) DEFAULT CURRENT\_USER

);

**Output:**



**CREATE OR REPLACE FUNCTION log\_price\_change()**

CREATE OR REPLACE FUNCTION fn\_product\_price\_audit()

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 trigger function:**

CREATE TRIGGER trg\_product\_price\_update

AFTER UPDATE OF unit\_price ON products

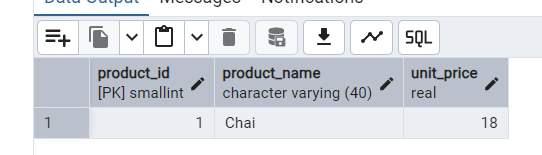
FOR EACH ROW

WHEN (OLD.unit\_price IS DISTINCT FROM NEW.unit\_price)

EXECUTE FUNCTION fn\_product\_price\_audit();

**Output:**

SELECT product\_id, product\_name, unit\_price FROM products WHERE product\_id = 1;



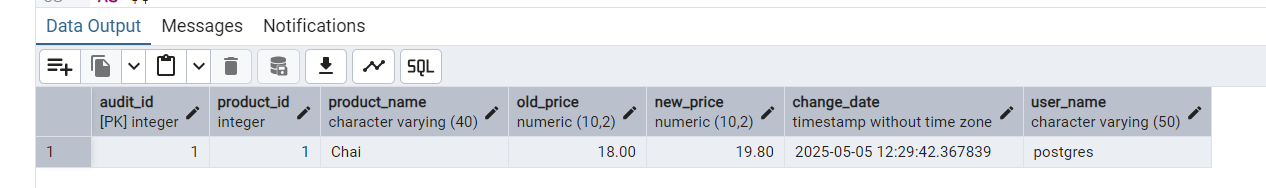
**Update price by 10% and verify the audit log:**

**Query:**

UPDATE products SET unit\_price = unit\_price \* 1.10 WHERE product\_id = 1;

SELECT \* FROM product\_price\_audit WHERE product\_id = 1;

**Output:**

****

**Q2)** **Create stored procedure using IN and INOUT parameters to assign tasks to employees**

**Create Table:**

**Query:**

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 Table or Replace Procedure:**

**Query:**

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

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

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

FROM employee\_tasks

WHERE employee\_id = p\_employee\_id;

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

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

END;

$$;

**-- First call**

CALL assign\_task(1, 'Review Reports', 0);

DO $$

DECLARE

task\_count INT := 0;

BEGIN

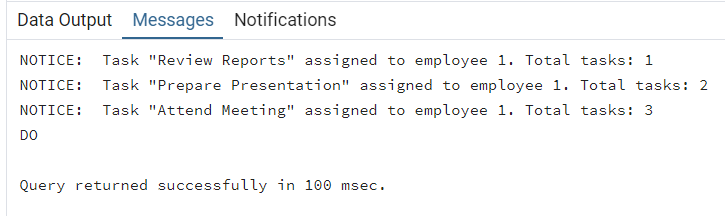
CALL assign\_task(1, 'Prepare Presentation', task\_count);

CALL assign\_task(1, 'Attend Meeting', task\_count);

END;

$$;

**Output:**



**SELECT \* FROM employee\_tasks ORDER BY task\_id;**

