DAY 9 assignment

1. Create AFTER UPDATE trigger to track product price changes

---Create a 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

);

----1. define trigger function

CREATE OR REPLACE FUNCTION log\_new\_product()

RETURNs TRIGGER as $$

BEGIN

----2. Insert into aufit log table

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

);

---3. return new row

RETURN NEW; ----required for AFTER TRIGGERS

END;

$$ LANGUAGE plpgsql;

---Create a row level trigger for below event: AFTER UPDATE OF unit\_price ON products

CREATE TRIGGER after\_product\_uodate

AFTER UPDATE OF unit\_price ON products

FOR EACH ROW

EXECUTE FUNCTION log\_new\_product();

--· Test the trigger by updating the product price by 10% to any one product\_id.

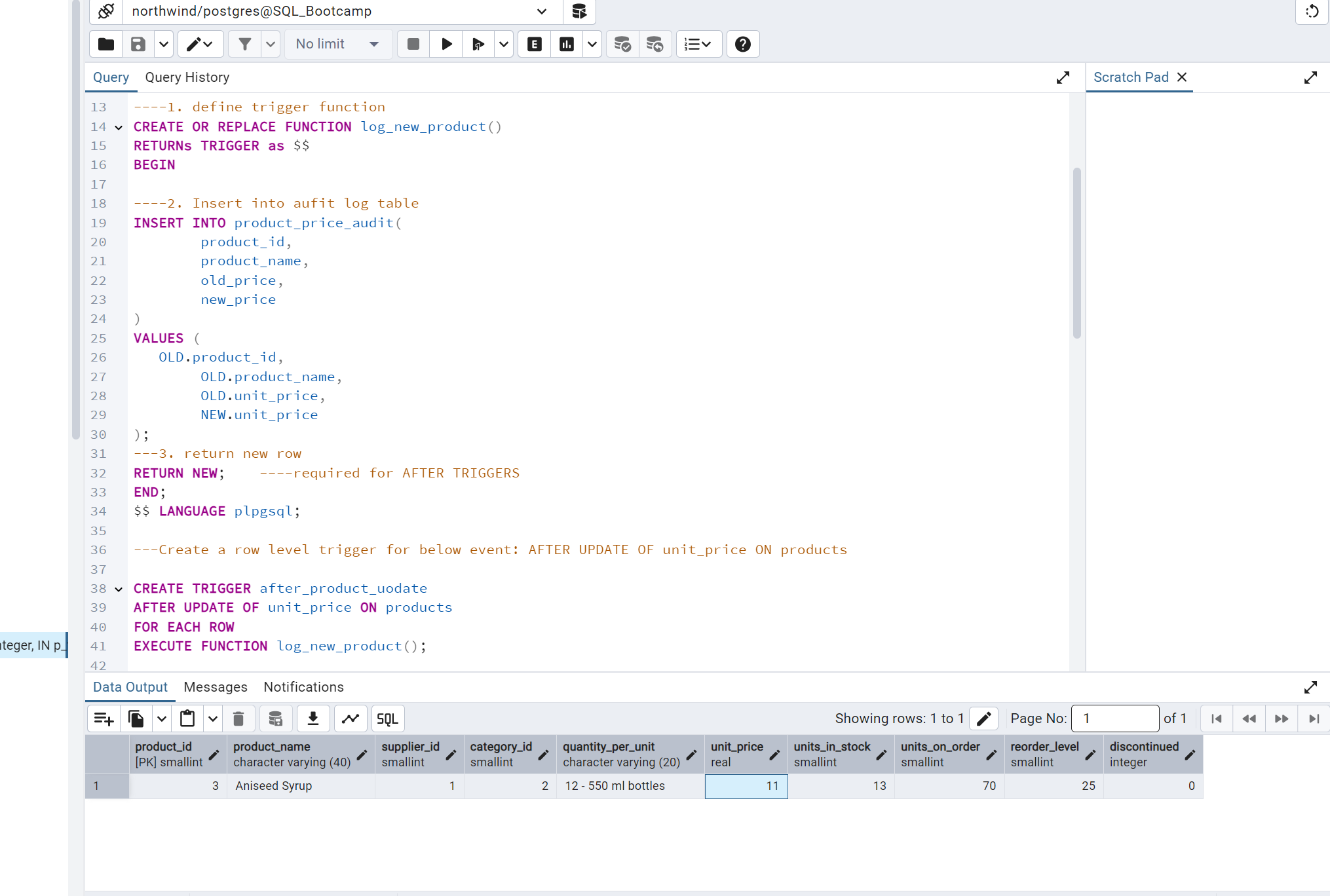
UPDATE products

SET unit\_price = unit\_price\* 1.1

WHERE product\_id = 3;

SELECT \* FROM products

WHERE product\_id = 3;



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

CREATE or REPLACE PROCEDURE assign\_tasks(

IN p\_employee\_id INT,

IN p\_task\_name VARCHAR(50),

INOUT p\_task\_count INT DEFAULT 0

)

LANGUAGE plpgsql

AS $$

BEGIN

---· Inside Logic: Create table employee\_tasks:

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);

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

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

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

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

SELECT

count(\*)

INTO p\_task\_count INT

FROM employee\_tasks;

----· Raise NOTICE message:

RAISE NOTICE 'ask "%" assigned to employee %. Total tasks: %',p\_task\_name, p\_employee\_id, p\_task\_count;

END;

$$;

---After creating stored procedure test by calling it:

CALL assign\_tasks( 2, 'Review Reports');

----You should see the entry in employee\_tasks table.

SELECT \* FROM employee\_tasks;

