---########### Day 3 ###########---

------USE Northwind from Kaggle:-------

---\*\*\*\*\*\*\*\*\*\*\* 1) Update the categoryName From “Beverages” to "Drinks" in the categories table.\*\*\*\*\*\*\*\*\*\*\*---

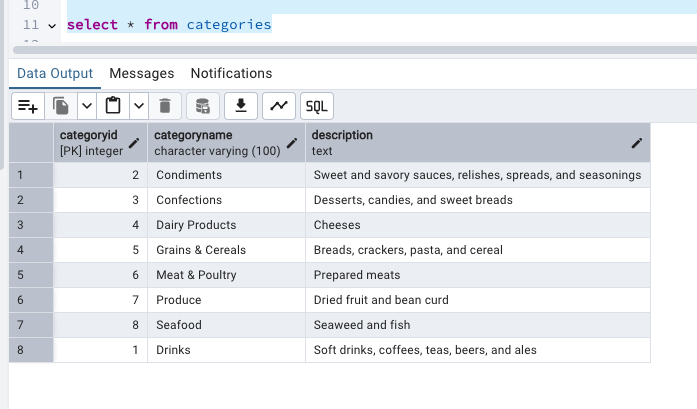
UPDATE categories

SET categoryName = 'Drinks'

WHERE categoryName = 'Beverages';



select \* from categories

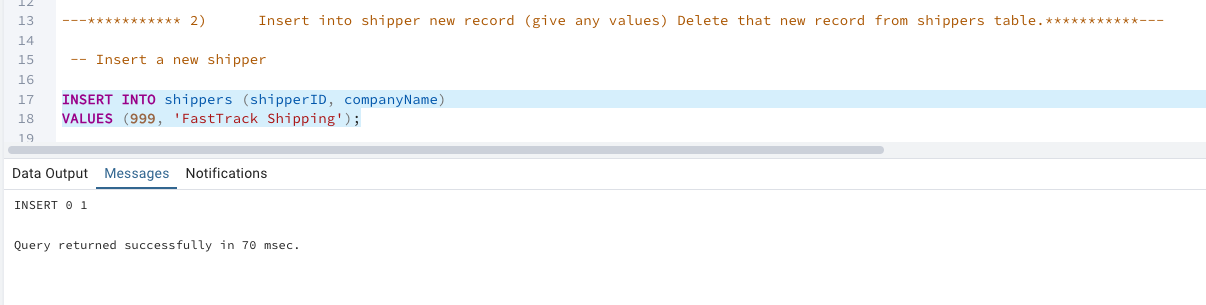


---\*\*\*\*\*\*\*\*\*\*\* 2) Insert into shipper new record (give any values) Delete that new record from shippers table.\*\*\*\*\*\*\*\*\*\*\*---

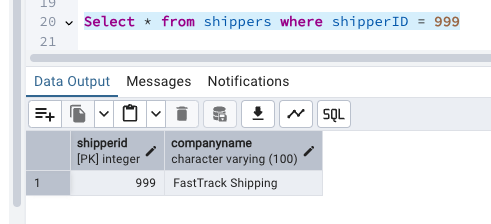
-- Part 1 Insert a new shipper

INSERT INTO shippers (shipperID, companyName)

VALUES (999, 'FastTrack Shipping');



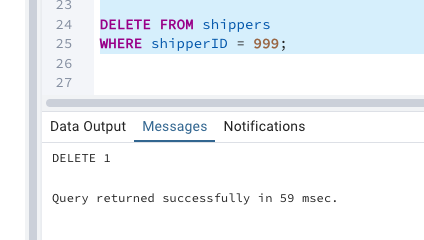
Select \* from shippers where shipperID = 999



-- Part 2 Delete that shipper

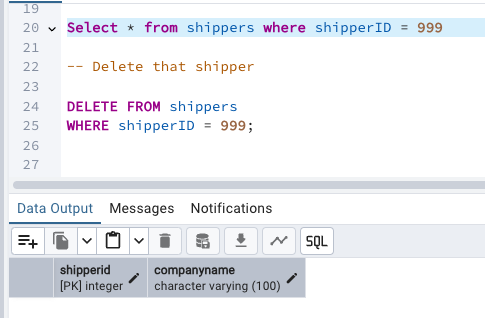
DELETE FROM shippers

WHERE shipperID = 999;



After delete

Select \* from shippers where shipperID = 999



---\*\*\*\*\*\*\*\*\*\*\*

/\* 3) Update categoryID=1 to categoryID=1001. Make sure related products update their categoryID too. Display the both category and products table to show the cascade.

Delete the categoryID= “3” from categories. Verify that the corresponding records are deleted automatically from products.

(HINT: Alter the foreign key on products(categoryID) to add ON UPDATE CASCADE, ON DELETE CASCADE)\*/--\*\*\*\*\*\*\*\*\*\*\*---

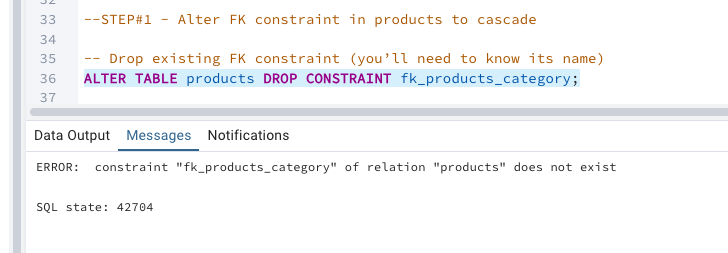
**---Update categoryID=1 to categoryID=1001. Make sure related products update their**

**--categoryID too. Display the both category and products table to show the cascade.**

--STEP#1 - Alter FK constraint in products to cascade

-- Dropping existing FK constraint

ALTER TABLE products DROP CONSTRAINT fk\_products\_category;



---Finding constraints

SELECT conname

FROM pg\_constraint

WHERE conrelid = 'products'::regclass

AND contype = 'f';

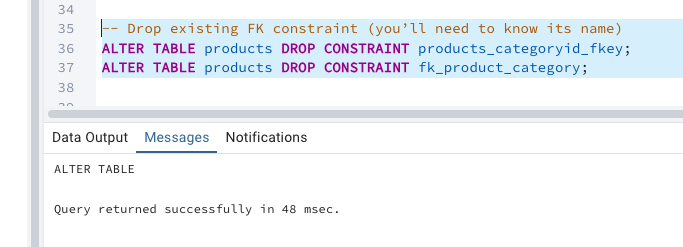
"products\_categoryid\_fkey"

"fk\_product\_category"

-- Drop existing FK constraint (you’ll need to know its name)

ALTER TABLE products DROP CONSTRAINT products\_categoryid\_fkey;

ALTER TABLE products DROP CONSTRAINT fk\_product\_category;



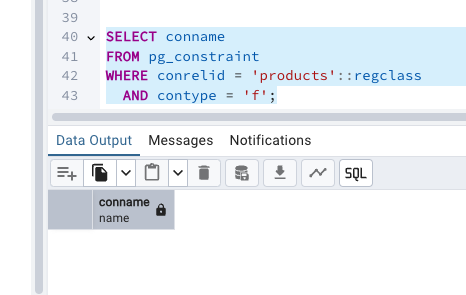
---Checking constraints removed?

SELECT conname

FROM pg\_constraint

WHERE conrelid = 'products'::regclass

AND contype = 'f';



-- Recreating FK with ON UPDATE CASCADE, ON DELETE CASCADE

ALTER TABLE products

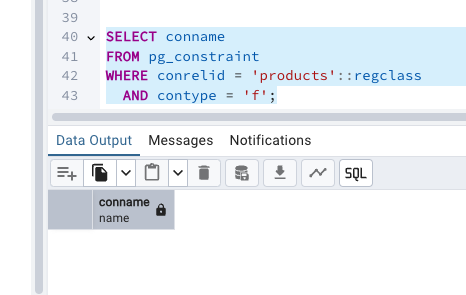
ADD CONSTRAINT fk\_products\_category

FOREIGN KEY (categoryID)

REFERENCES categories(categoryID)

ON UPDATE CASCADE

ON DELETE CASCADE;



-- Recreating FK with ON UPDATE CASCADE, ON DELETE CASCADE

ALTER TABLE products

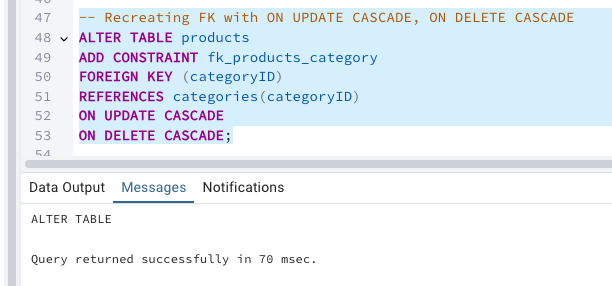
ADD CONSTRAINT fk\_products\_category

FOREIGN KEY (categoryID)

REFERENCES categories(categoryID)

ON UPDATE CASCADE

ON DELETE CASCADE;



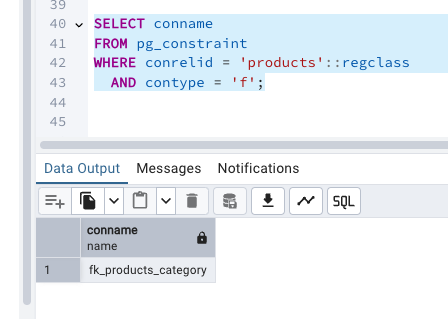
-- -Checking products table on the fk alteration

SELECT conname

FROM pg\_constraint

WHERE conrelid = 'products'::regclass

AND contype = 'f';

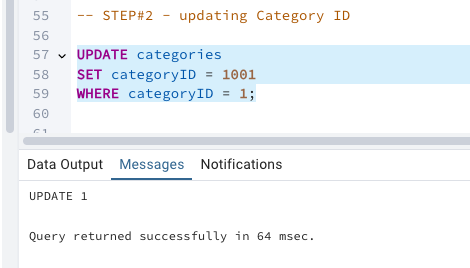


-- STEP#2 - updating Category ID

UPDATE categories

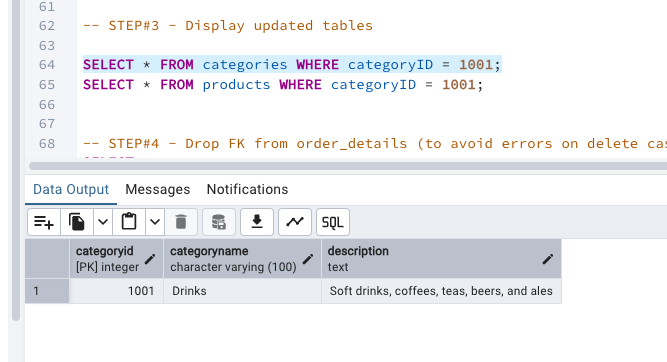
SET categoryID = 1001

WHERE categoryID = 1;

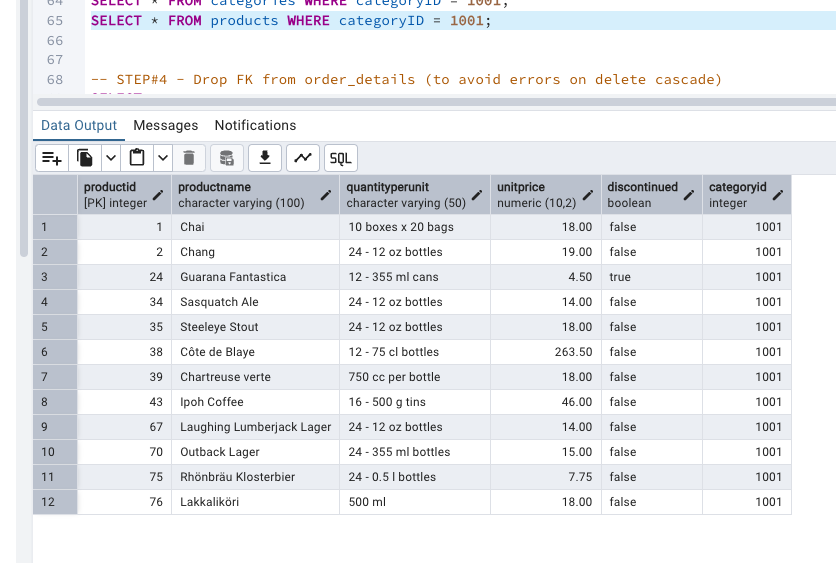


-- STEP#3 - Display updated tables

SELECT \* FROM categories WHERE categoryID = 1001;



SELECT \* FROM products WHERE categoryID = 1001;



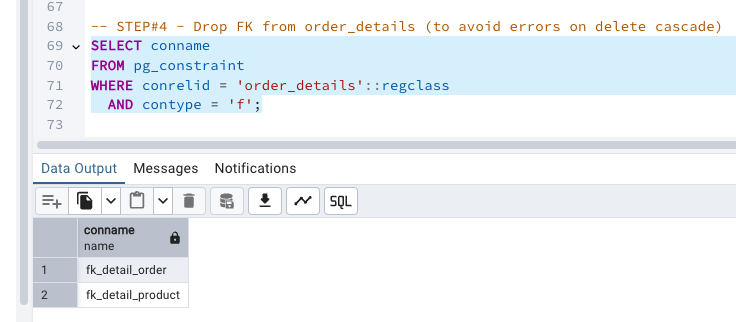
-- STEP#4 - Drop FK from order\_details (to avoid errors on delete cascade)

SELECT conname

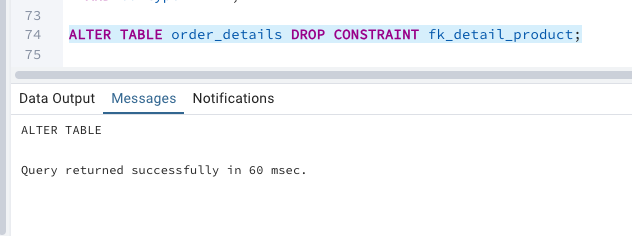
FROM pg\_constraint

WHERE conrelid = 'order\_details'::regclass

AND contype = 'f';



ALTER TABLE order\_details DROP CONSTRAINT fk\_detail\_product;



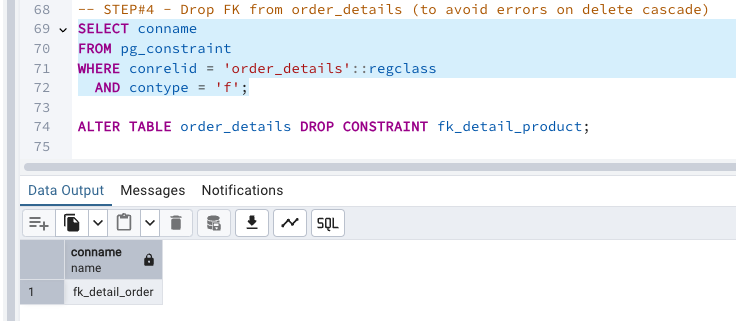
---checking constraint alteration

SELECT conname

FROM pg\_constraint

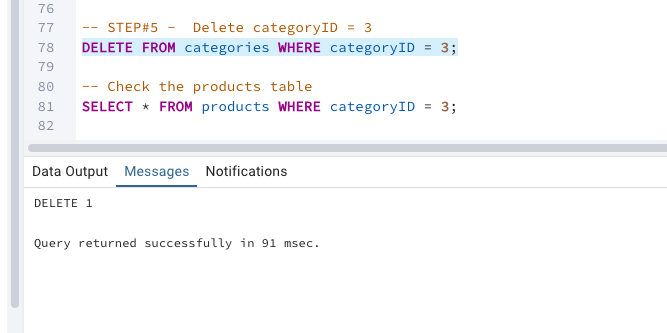
WHERE conrelid = 'order\_details'::regclass

AND contype = 'f';



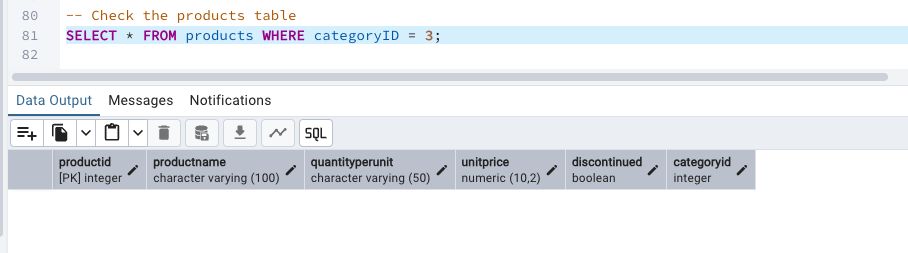
-- STEP#5 - Delete categoryID = 3

DELETE FROM categories WHERE categoryID = 3;



-- Check the products table

SELECT \* FROM products WHERE categoryID = 3;



---\*\*\*\*\*\*\*\*\*\*\* 4) Delete the customer = “VINET” from customers. Corresponding customers in orders table should be set to null

--(HINT: Alter the foreign key on orders(customerID) to use ON DELETE SET NULL)

--Altering FK

SELECT conname

FROM pg\_constraint

WHERE conrelid = 'orders'::regclass

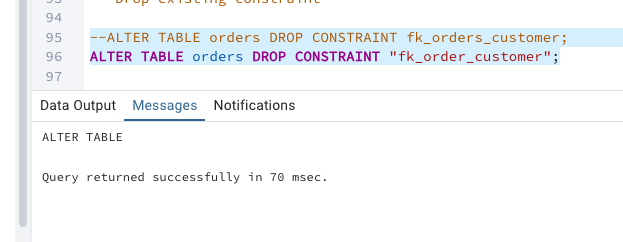
AND contype = 'f';



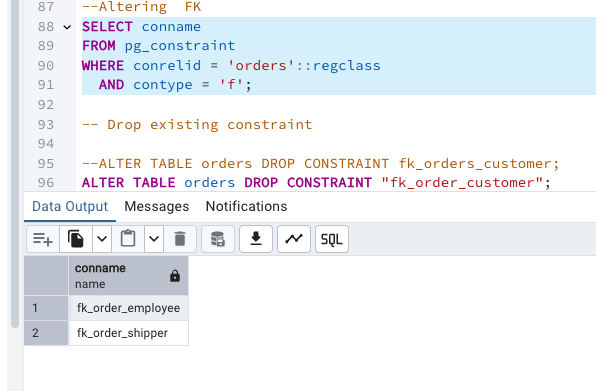
-- Drop existing constraint

--ALTER TABLE orders DROP CONSTRAINT fk\_orders\_customer; -- this does not work

ALTER TABLE orders DROP CONSTRAINT "fk\_order\_customer";



--checking fk alteration



-- Add new constraint with SET NULL on delete

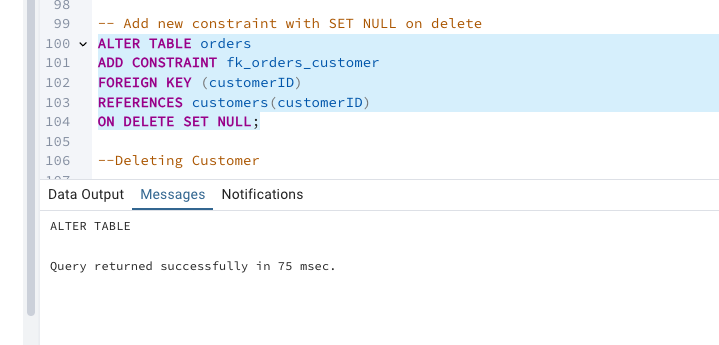
ALTER TABLE orders

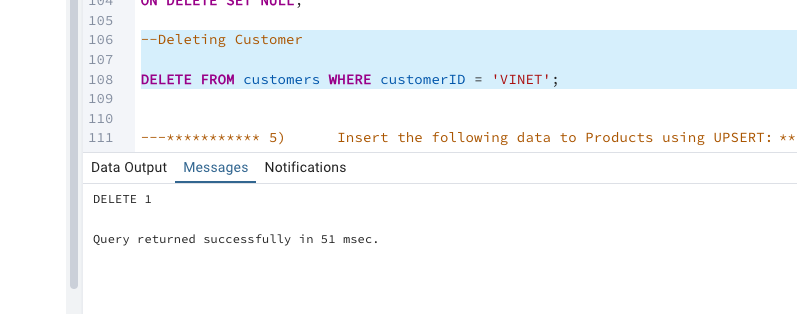
ADD CONSTRAINT fk\_orders\_customer

FOREIGN KEY (customerID)

REFERENCES customers(customerID)

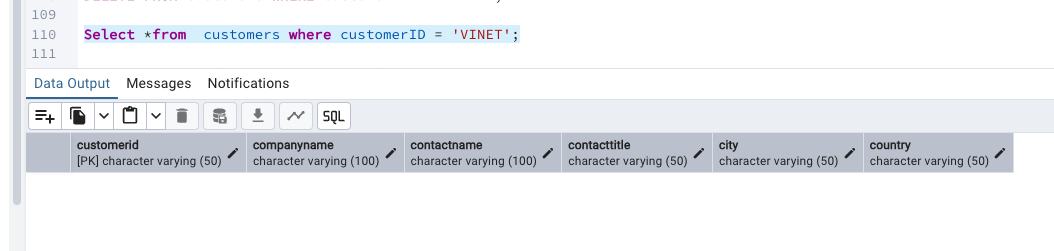
ON DELETE SET NULL;





--cross verifying delete

Select \*from customers where customerID = 'VINET';



---\*\*\*\*\*\*\*\*\*\*\* 5) Insert the following data to Products using UPSERT: \*\*\*\*\*\*\*\*\*\*\*---

/\*product\_id = 100, product\_name = Wheat bread, quantityperunit=1,unitprice = 13, discontinued = 0, categoryID=3

product\_id = 101, product\_name = White bread, quantityperunit=5 boxes,unitprice = 13, discontinued = 0, categoryID=3

product\_id = 100, product\_name = Wheat bread, quantityperunit=10 boxes,unitprice = 13, discontinued = 0, categoryID=3

(this should update the quantityperunit for product\_id = 100)\*/

--as per Chaitra's Slack suggestion / UPSERT with new product\_new table

--Step1 Create new table

CREATE TABLE product\_new (

productID INT PRIMARY KEY,

productName VARCHAR(255),

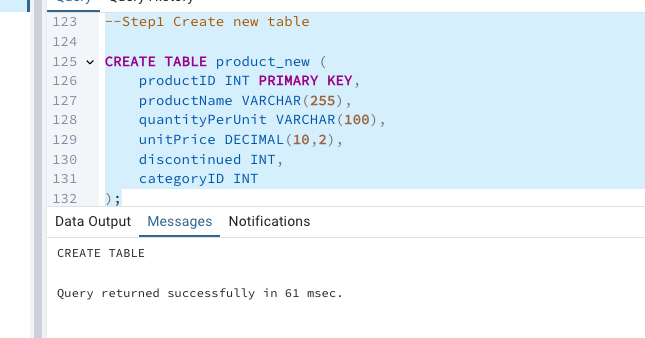
quantityPerUnit VARCHAR(100),

unitPrice DECIMAL(10,2),

discontinued INT,

categoryID INT

);



--Step2 Importing CSV in product\_new table via import option

--Import product.csv

---Step 3 UPSERT queries

-- First insert Wheat and White bread

INSERT INTO product\_new (productID, productName, quantityPerUnit, unitPrice, discontinued, categoryID)

VALUES

(100, 'Wheat bread', '1', 13, 0, 3),

(101, 'White bread', '5 boxes', 13, 0, 3)

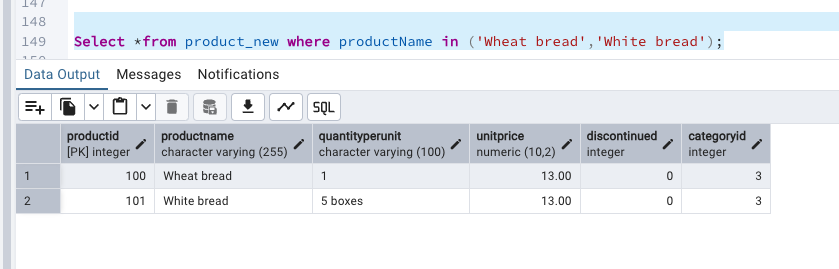
ON CONFLICT (productID) DO UPDATE

SET quantityPerUnit = EXCLUDED.quantityPerUnit;



--checking insert operation

Select \*from product\_new where productName in ('Wheat bread','White bread');



INSERT INTO product\_new (productID, productName, quantityPerUnit, unitPrice, discontinued, categoryID)

VALUES

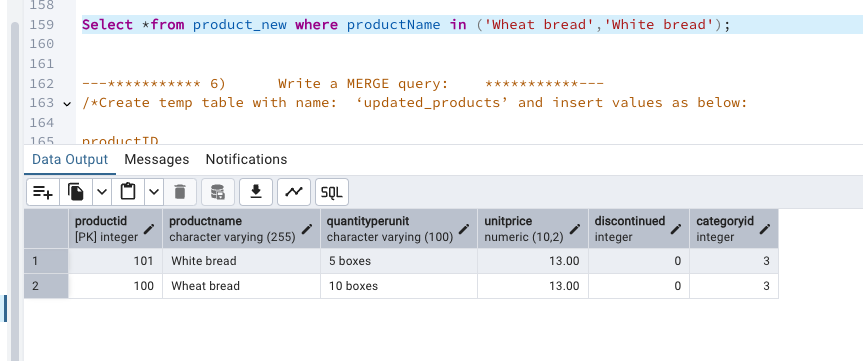
(100, 'Wheat bread', '10 boxes', 13, 0, 3)

ON CONFLICT (productID) DO UPDATE

SET quantityPerUnit = EXCLUDED.quantityPerUnit;



Select \*from product\_new where productName in ('Wheat bread','White bread');



---\*\*\*\*\*\*\*\*\*\*\* 6) Write a MERGE query: \*\*\*\*\*\*\*\*\*\*\*---

/\*Create temp table with name: ‘updated\_products’ and insert values as below:

productID

productName

quantityPerUnit

unitPrice

discontinued

categoryID

100

Wheat bread

10

20

1

3

101

White bread

5 boxes

19.99

0

3

102

Midnight Mango Fizz

24 - 12 oz bottles

19

0

1

103

Savory Fire Sauce

12 - 550 ml bottles

10

0

2\*/

-- Create temp table and insert values

CREATE TEMP TABLE updated\_products (

productID INT PRIMARY KEY,

productName VARCHAR(255),

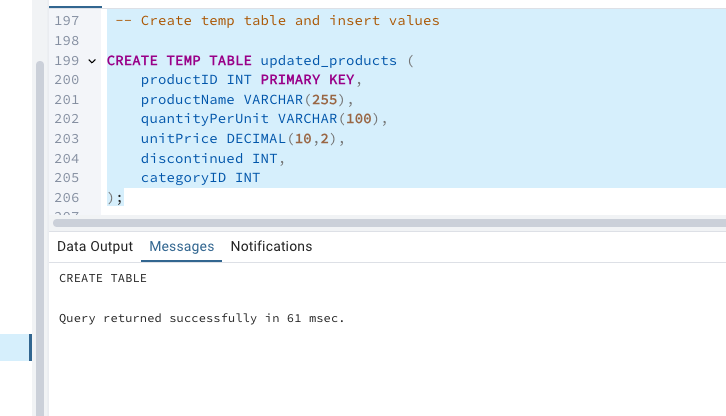
quantityPerUnit VARCHAR(100),

unitPrice DECIMAL(10,2),

discontinued INT,

categoryID INT

);



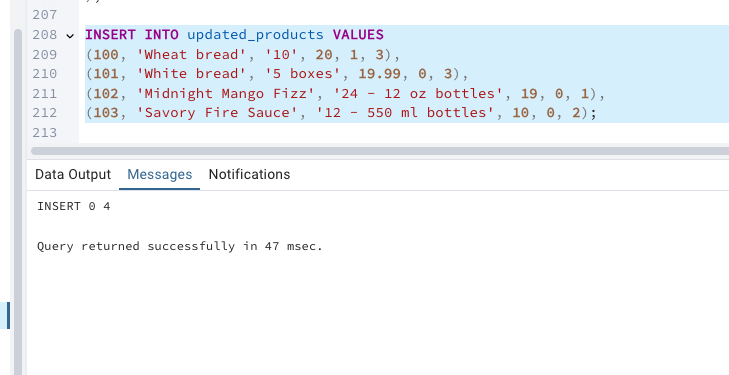
INSERT INTO updated\_products VALUES

(100, 'Wheat bread', '10', 20, 1, 3),

(101, 'White bread', '5 boxes', 19.99, 0, 3),

(102, 'Midnight Mango Fizz', '24 - 12 oz bottles', 19, 0, 1),

(103, 'Savory Fire Sauce', '12 - 550 ml bottles', 10, 0, 2);



---\*\*\*\*\*\*\*\*\*\*\* Update the price and discontinued status for from below table ‘updated\_products’ only if there are matching products and updated\_products .discontinued =0 \*\*\*\*\*\*\*\*\*\*\*---

SELECT

productID,

productName,

quantityPerUnit,

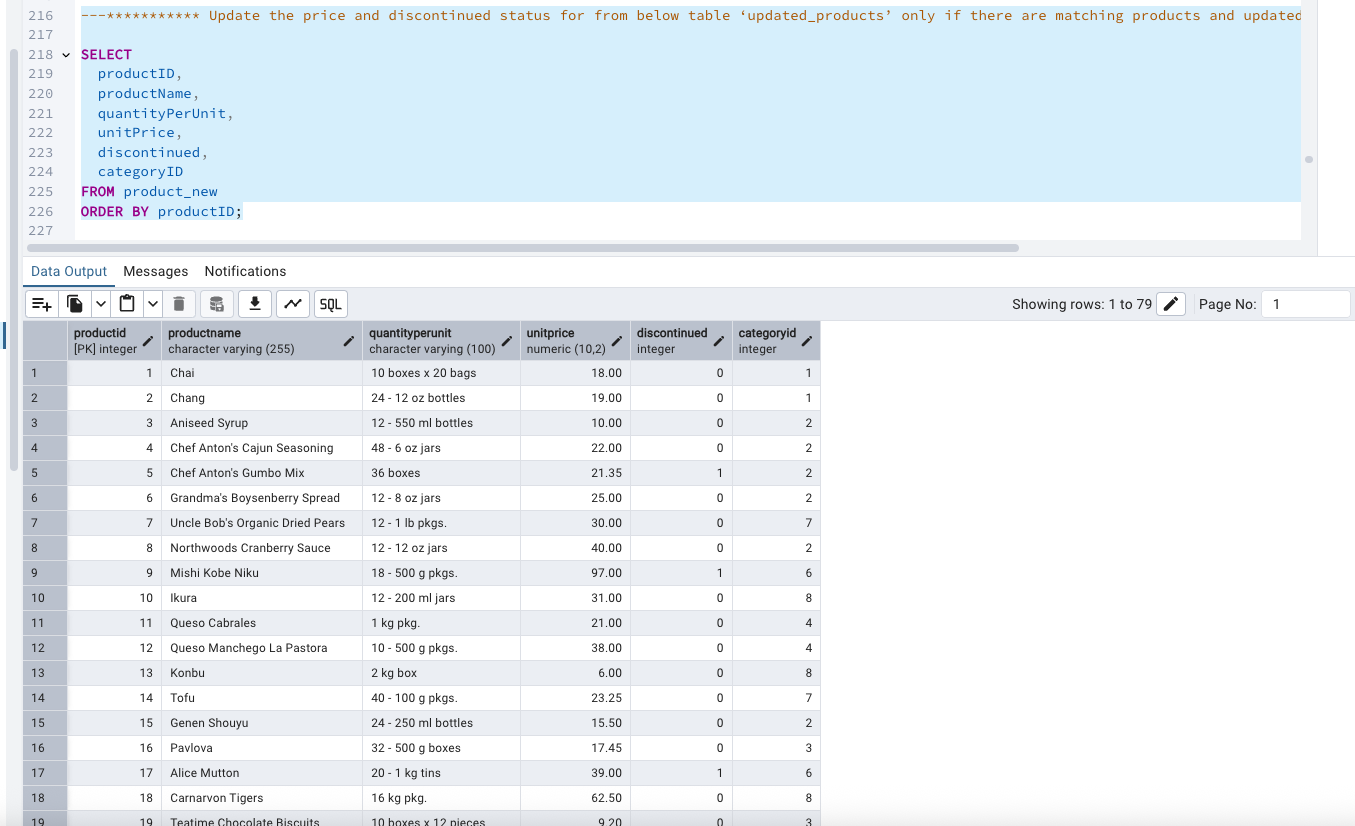
unitPrice,

discontinued,

categoryID

FROM product\_new

ORDER BY productID;



SELECT

productID,

productName,

quantityPerUnit,

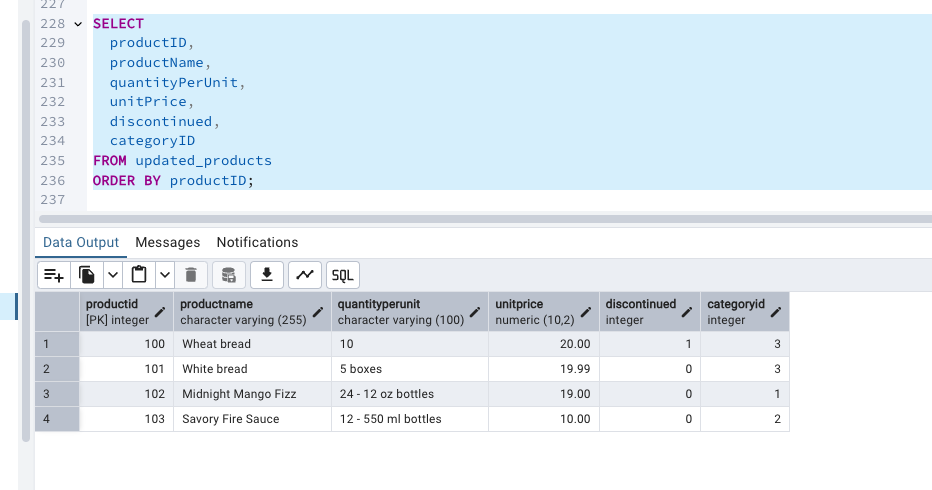
unitPrice,

discontinued,

categoryID

FROM updated\_products

ORDER BY productID;



-- Update matching and discontinued = 0

UPDATE product\_new p

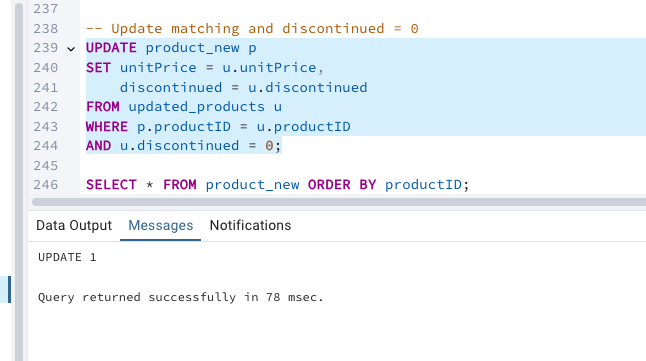
SET unitPrice = u.unitPrice,

discontinued = u.discontinued

FROM updated\_products u

WHERE p.productID = u.productID

AND u.discontinued = 0;



SELECT

productID,

productName,

quantityPerUnit,

unitPrice,

discontinued,

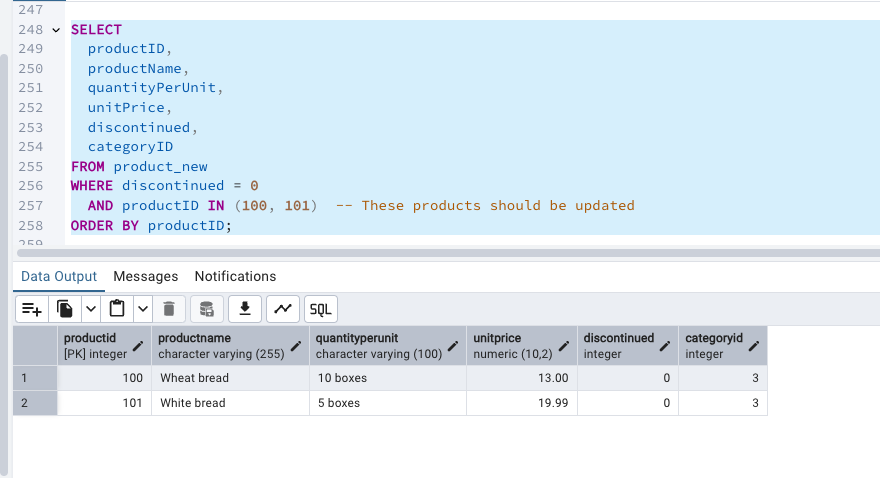
categoryID

FROM product\_new

WHERE discontinued = 0

AND productID IN (100, 101) -- These products should be updated

ORDER BY productID;



---\*\*\*\*\*\*\*\*\*\*\* If there are matching products and updated\_products .discontinued =1 then delete \*\*\*\*\*\*\*\*\*\*\*---

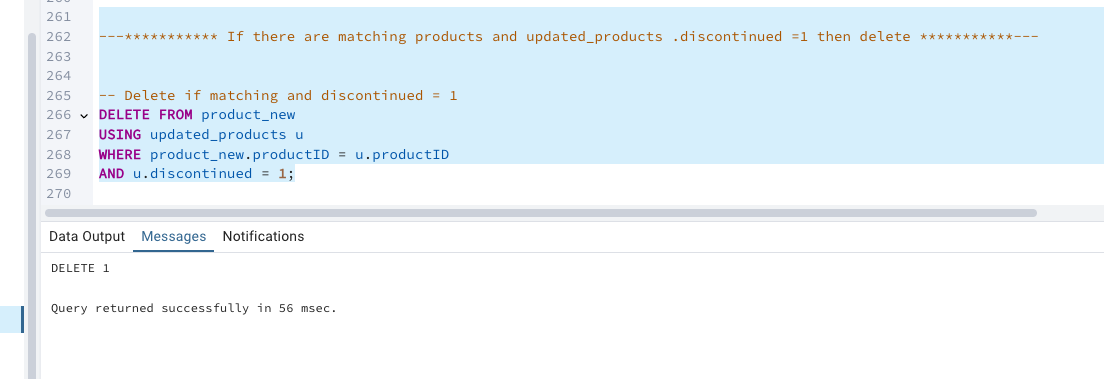
-- Delete if matching and discontinued = 1

DELETE FROM product\_new

USING updated\_products u

WHERE product\_new.productID = u.productID

AND u.discontinued = 1;



SELECT

productID,

productName,

quantityPerUnit,

unitPrice,

discontinued,

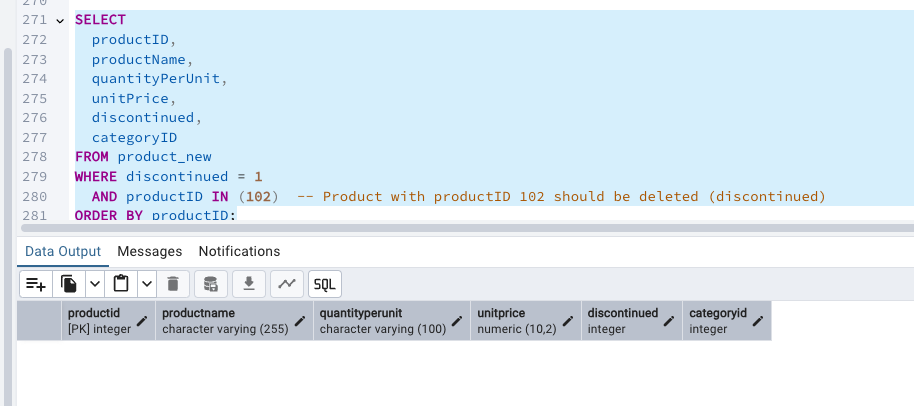
categoryID

FROM product\_new

WHERE discontinued = 1

AND productID IN (102) -- Product with productID 102 should be deleted (discontinued)

ORDER BY productID;



---\*\*\*\*\*\*\*\*\*\*\* Insert any new products from updated\_products that don’t exist in products only if updated\_products .discontinued =0.\*\*\*\*\*\*\*\*\*\*\*---

-- Insert new if not exists and discontinued = 0

INSERT INTO product\_new (productID, productName, quantityPerUnit, unitPrice, discontinued, categoryID)

SELECT u.productID, u.productName, u.quantityPerUnit, u.unitPrice, u.discontinued, u.categoryID

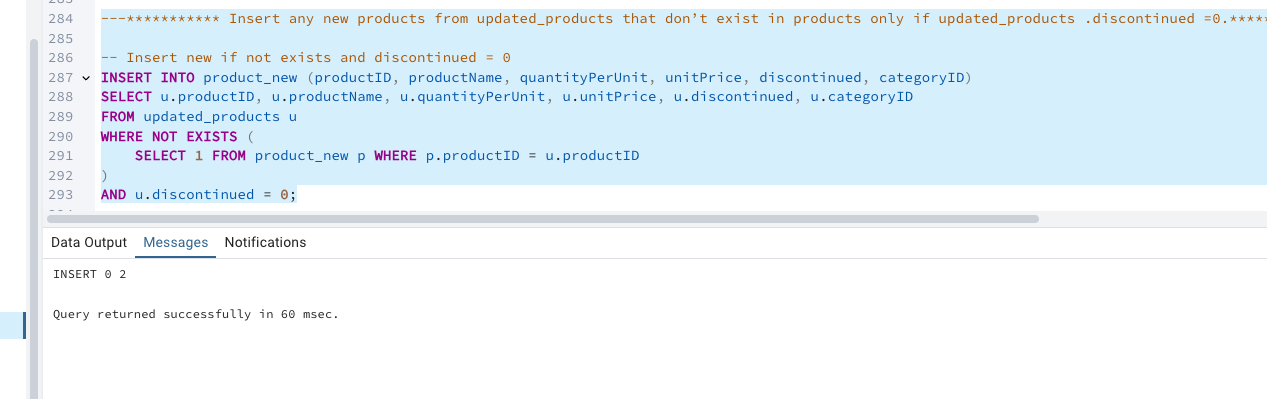
FROM updated\_products u

WHERE NOT EXISTS (

SELECT 1 FROM product\_new p WHERE p.productID = u.productID

)

AND u.discontinued = 0;



SELECT

productID,

productName,

quantityPerUnit,

unitPrice,

discontinued,

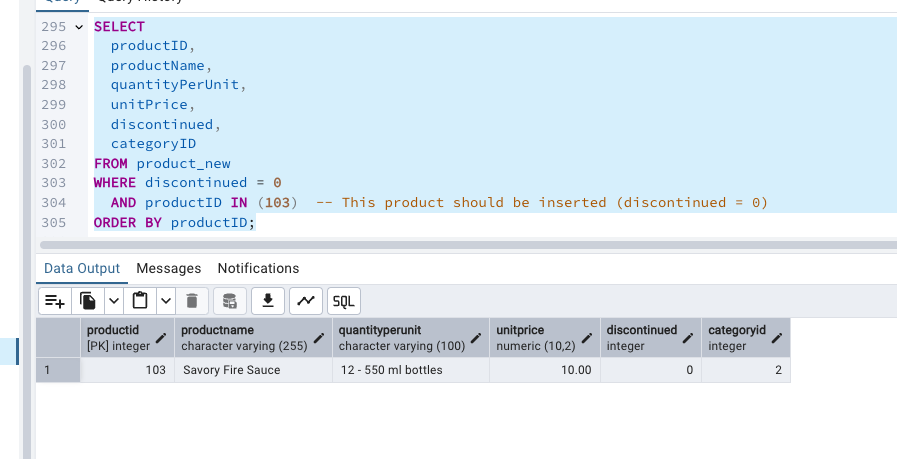
categoryID

FROM product\_new

WHERE discontinued = 0

AND productID IN (103) -- This product should be inserted (discontinued = 0)

ORDER BY productID;



SELECT

productID,

productName,

quantityPerUnit,

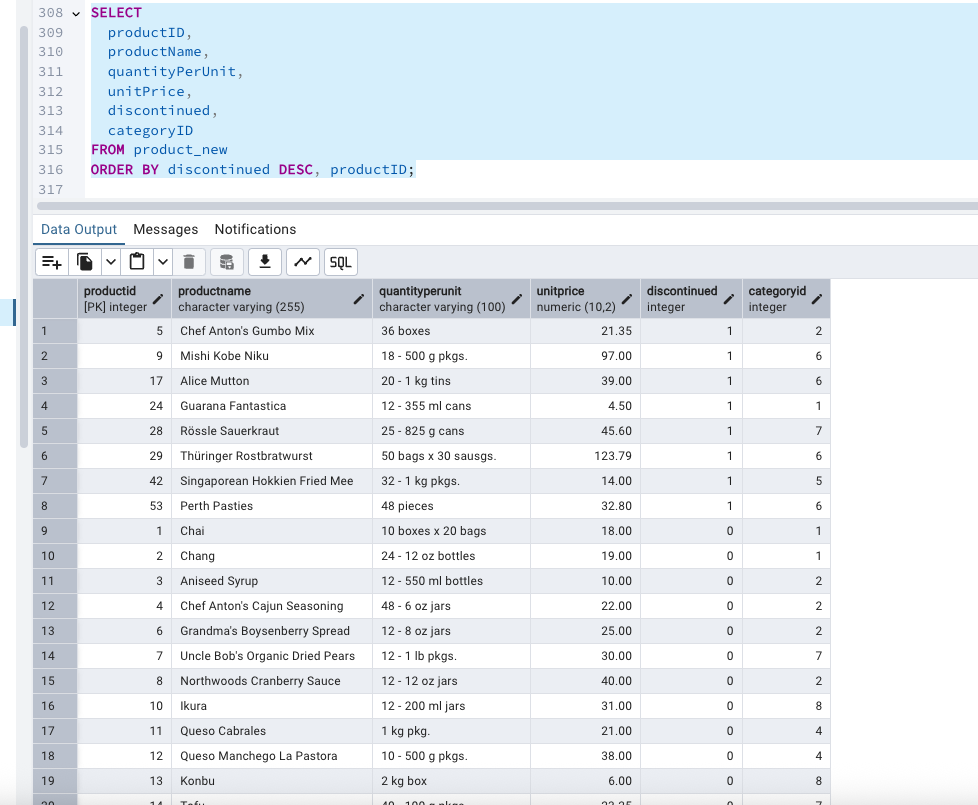
unitPrice,

discontinued,

categoryID

FROM product\_new

ORDER BY discontinued DESC, productID;



---\*\*\*\*\*\*\*\*\*\*\* USE NEW Northwind DB:\*\*\*\*\*\*\*\*\*\*\*---

---\*\*\*\*\*\*\*\*\*\*\* 7) List all orders with employee full names. (Inner join)\*\*\*\*\*\*\*\*\*\*\*---

SELECT

o.order\_id,

o.order\_date,

e.first\_name || ' ' || e.last\_name AS employee\_full\_name

FROM

orders o

INNER JOIN

employees e ON o.employee\_id = e.employee\_id;

