

Muhamad Fadhli Akbar

Cortana

## ASSIGMENT 3 ABOUT SQL

### SQL 3.1

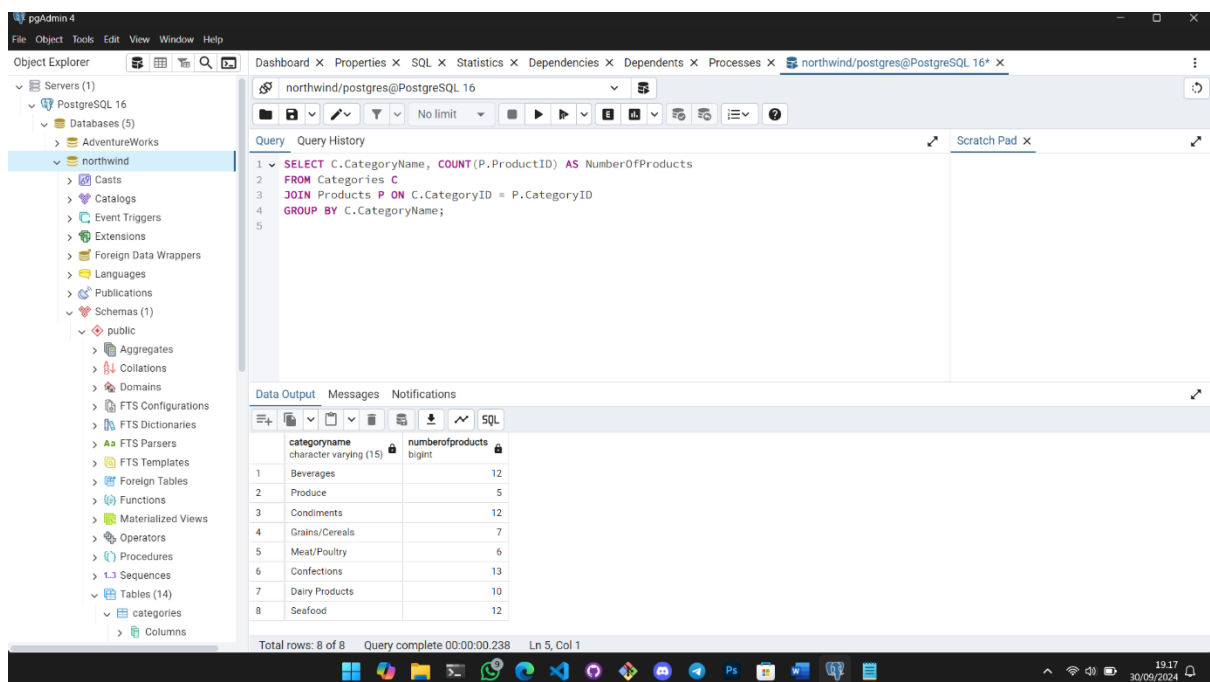
#### 1. What is the number of products for each category (do it with joins)

```
SELECT C.CategoryName, COUNT(P.ProductID) AS NumberOfProducts
```

```
FROM Categories C
```

```
JOIN Products P ON C.CategoryID = P.CategoryID
```

```
GROUP BY C.CategoryName;
```



The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure, including the 'northwind' database and its 'categories' table. The central pane shows the following SQL query:

```
1 SELECT C.CategoryName, COUNT(P.ProductID) AS NumberOfProducts
2 FROM Categories C
3 JOIN Products P ON C.CategoryID = P.CategoryID
4 GROUP BY C.CategoryName;
5
```

The bottom pane displays the query results in a table format:

categoryname	numberofproducts
Beverages	12
Produce	5
Condiments	12
Grains/Cereals	7
Meat/Poultry	6
Confections	13
Dairy Products	10
Seafood	12

Total rows: 8 of 8. Query complete 00:00:00.238. Ln 5, Col 1.

#### 2. Total value of each product sold for year of 1997

```
SELECT p.ProductID, SUM(od.UnitPrice * od.Quantity) AS total_value
```

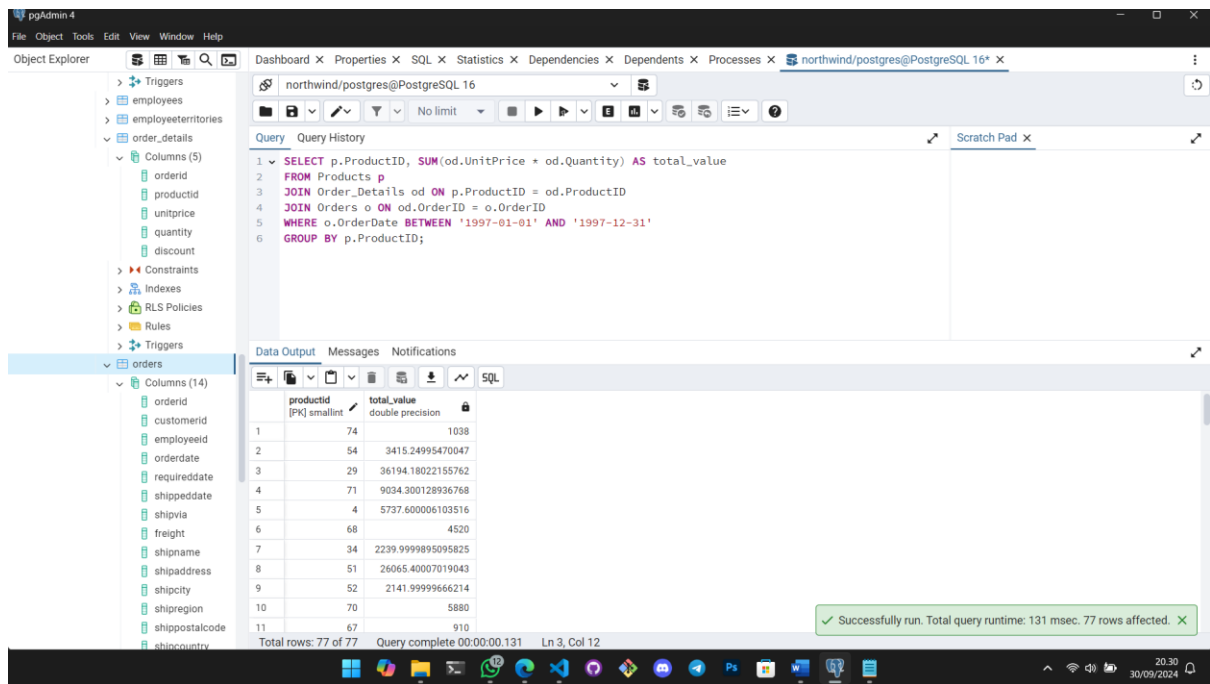
```
FROM Products p
```

```
JOIN Order_Details od ON p.ProductID = od.ProductID
```

```
JOIN Orders o ON od.OrderID = o.OrderID
```

```
WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'
```

```
GROUP BY p.ProductID;
```



### 3. Customers that have bought more than \$5000 of products

```
SELECT c.CustomerID, c.CompanyName, SUM(od.UnitPrice * od.Quantity) AS total_spent
```

```
FROM Customers c
```

```
JOIN Orders o ON c.CustomerID = o.CustomerID
```

```
JOIN Order_Details od ON o.OrderID = od.OrderID
```

```
GROUP BY c.CustomerID, c.CompanyName
```

```
HAVING SUM(od.UnitPrice * od.Quantity) > 5000
```

```
ORDER BY total_spent DESC;
```

pgAdmin 4

Object Explorer

- Triggers
- employees
- employee\_territories
- order\_details
  - Columns (5)
    - orderid
    - productid
    - unitprice
    - quantity
    - discount
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
- orders
  - Columns (14)
    - orderid
    - customerid
    - employeeid
    - orderdate
    - requireddate
    - shippeddate
    - shipvia
    - freight
    - shipname
    - shipaddress
    - shipcity
    - shipregion
    - shippostalcode
    - shipcountry

Query

```

1 SELECT c.CustomerID, c.CompanyName, SUM(od.UnitPrice * od.Quantity) AS total_spent
2 FROM Customers c
3 JOIN Orders o ON c.CustomerID = o.CustomerID
4 JOIN Order_Details od ON o.OrderID = od.OrderID
5 GROUP BY c.CustomerID, c.CompanyName
6 HAVING SUM(od.UnitPrice * od.Quantity) > 5000
7 ORDER BY total_spent DESC;

```

Data Output

	customerid [PK] character	companyname character varying (40)	total_spent double precision
1	QUICK	QUICK-Stop	117483.390147686
2	SAVEA	Save-a-lot Markets	115673.38964271545
3	ERNSH	Ernst Handel	113236.67978191376
4	HUNGO	Hungry Owl All-Night Grocers	57317.39016246796
5	RATTC	Rattlesnake Canyon Grocery	52245.90034675598
6	HANAR	Hanari Carnes	34101.149973869324
7	FOLKO	Folk och fä HB	32555.55001926422
8	MEREP	Mère Pailarde	32203.900234222412
9	KOENE	Königlich Essen	31745.749893188477
10	QUEEN	Queen Cozinha	30226.10017967224
11	WHITC	White Clover Markets	29073.449927330017

Successfully run. Total query runtime: 89 msec. 56 rows affected.

4. Customers that have bought more than \$5000 of products with order date in first six month of the year of 1997 (see the effect of WHERE clause)

SELECT c.CustomerID, c.CompanyName, SUM(od.UnitPrice \* od.Quantity) AS total\_spent

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN Order\_Details od ON o.OrderID = od.OrderID

WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-06-30'

GROUP BY c.CustomerID, c.CompanyName

HAVING SUM(od.UnitPrice \* od.Quantity) > 5000

ORDER BY total\_spent DESC;

pgAdmin 4

Object Explorer

- Triggers
- employees
- employee\_territories
- order\_details
  - Columns (5)
    - orderid
    - productid
    - unitprice
    - quantity
    - discount
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
- orders
  - Columns (14)
    - orderid
    - customerid
    - employeeid
    - orderdate
    - requireddate
    - shippeddate
    - shipvia
    - freight
    - shipname
    - shipaddress
    - shipcity
    - shipregion
    - shippostalcode
    - shipcountry

Query

```

1 SELECT c.CustomerID, c.CompanyName, SUM(od.UnitPrice * od.Quantity) AS total_spent
2 FROM Customers c
3 JOIN Orders o ON c.CustomerID = o.CustomerID
4 JOIN Order_Details od ON o.OrderID = od.OrderID
5 WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-06-30'
6 GROUP BY c.CustomerID, c.CompanyName
7 HAVING SUM(od.UnitPrice * od.Quantity) > 5000
8 ORDER BY total_spent DESC;

```

Data Output

	customerid [PK] character	companyname character varying (40)	total_spent double precision
1	QUICK	QUICK-Stop	32723.900236606598
2	ERNSH	Ernst Handel	20578.599815368652
3	RATTC	Rattlesnake Canyon Grocery	16640.70014190674
4	SAVEA	Save-a-lot Markets	16305.039953231812
5	MEREP	Mère Pailarde	16025.100114822388
6	SIMOB	Simons bistro	12118.400121688843
7	HUNGO	Hungry Owl All-Night Grocers	9520.499900817871
8	WARTH	Wartian Herkku	9473.200092315674
9	LEHMS	Lehmanns Marktstand	7793.299932479858
10	BLONP	Blondessdls père et fils	7261.800029754639
11	SUPRD	Suprêmes délices	6375.100052833557

Successfully run. Total query runtime: 80 msec. 21 rows affected.

## SQL 3.2

### 1. Distinct countries of all our customers and suppliers in alphabetical order

```
SELECT DISTINCT Country
```

```
FROM (
```

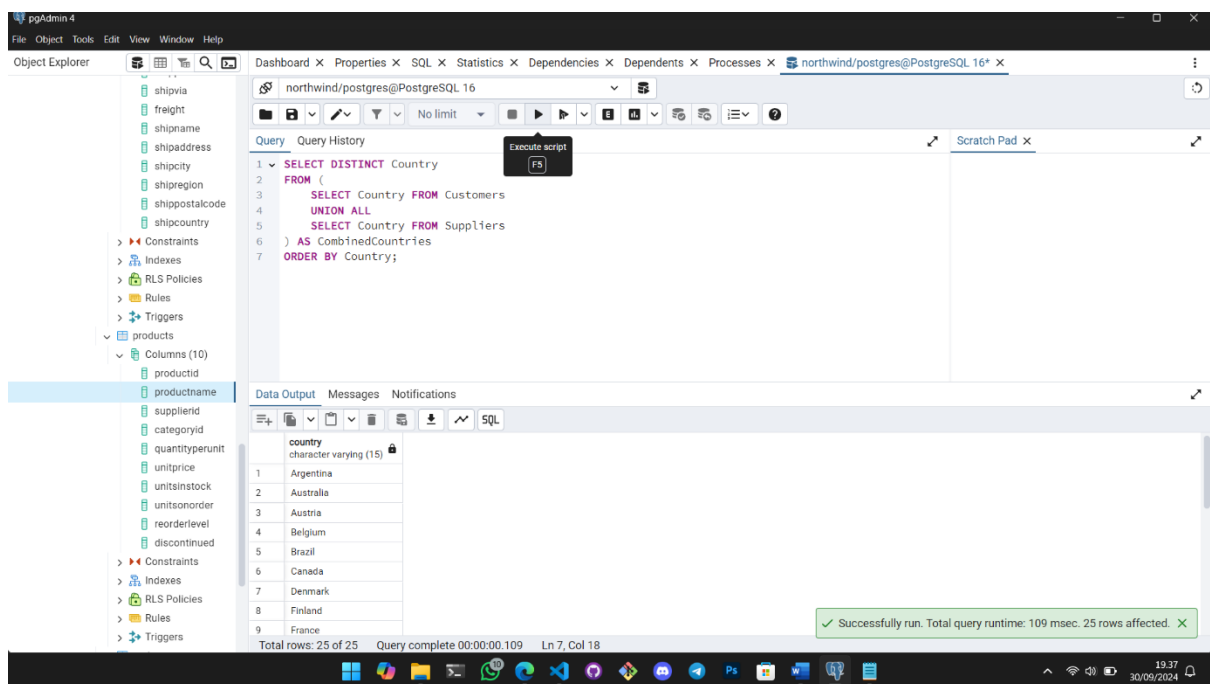
```
    SELECT Country FROM Customers
```

```
    UNION ALL
```

```
    SELECT Country FROM Suppliers
```

```
) AS CombinedCountries
```

```
ORDER BY Country;
```



### 2. All list of countries of our suppliers and customers, with a record for each one

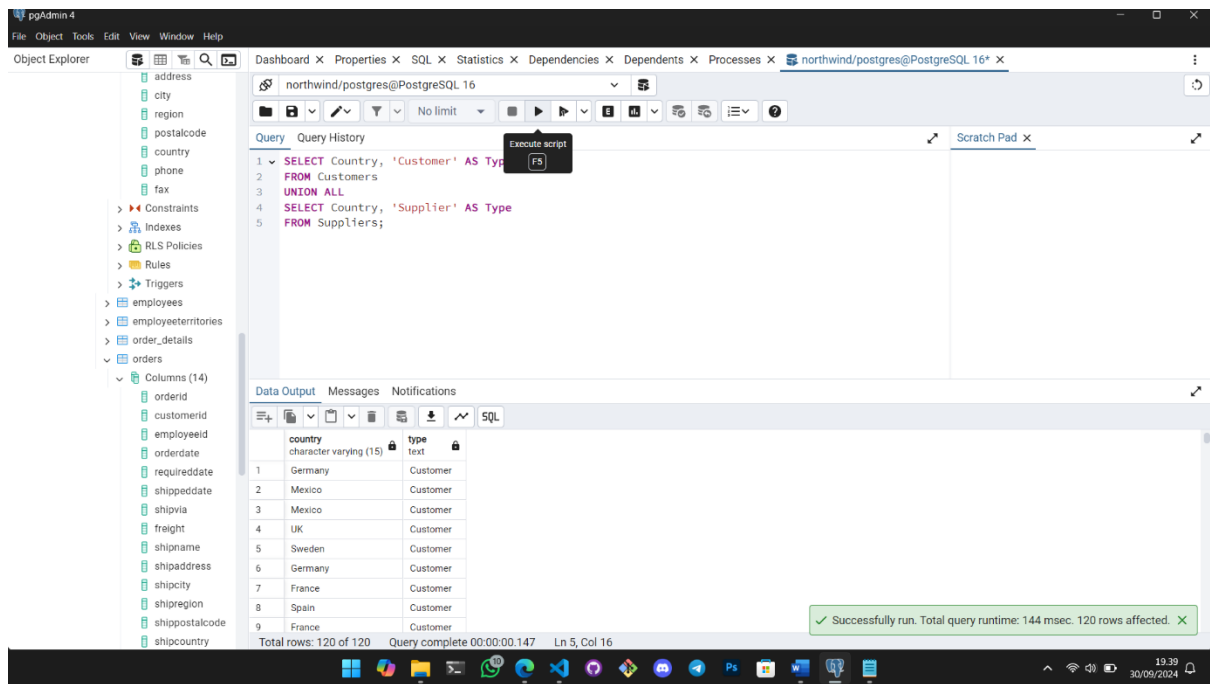
```
SELECT Country, 'Customer' AS Type
```

```
FROM Customers
```

```
UNION ALL
```

```
SELECT Country, 'Supplier' AS Type
```

```
FROM Suppliers;
```

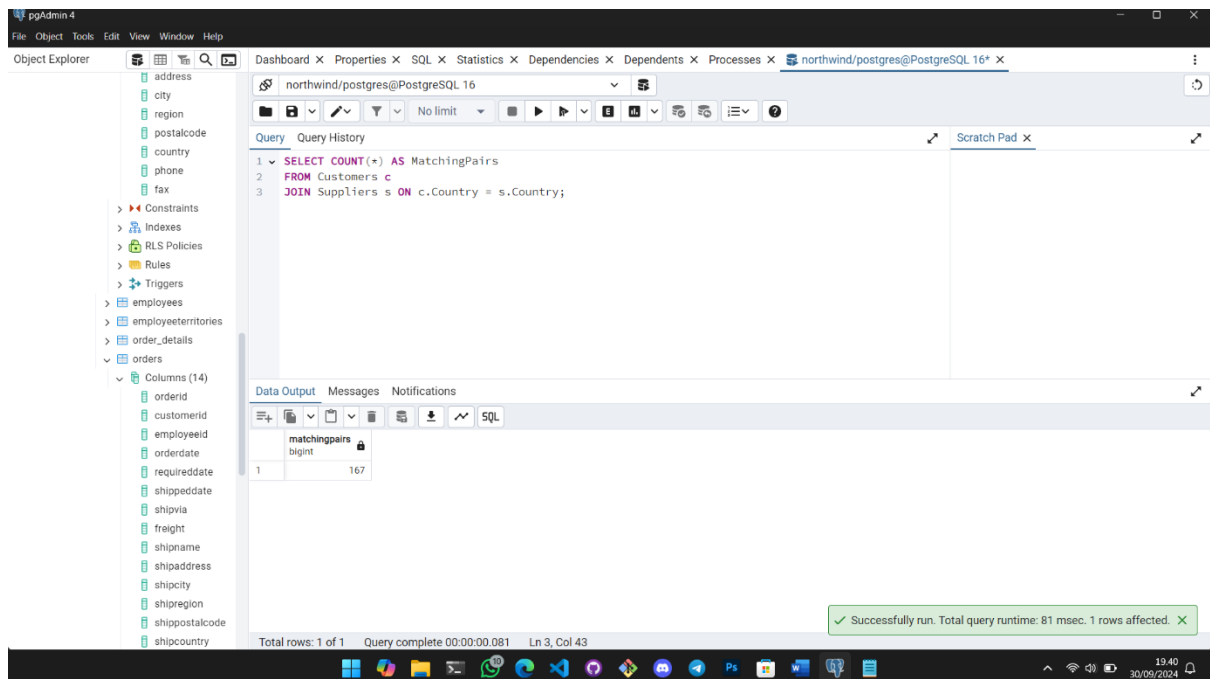


### 3. Find the number of customer and supplier pairs that are in the same country

SELECT COUNT(\*) AS MatchingPairs

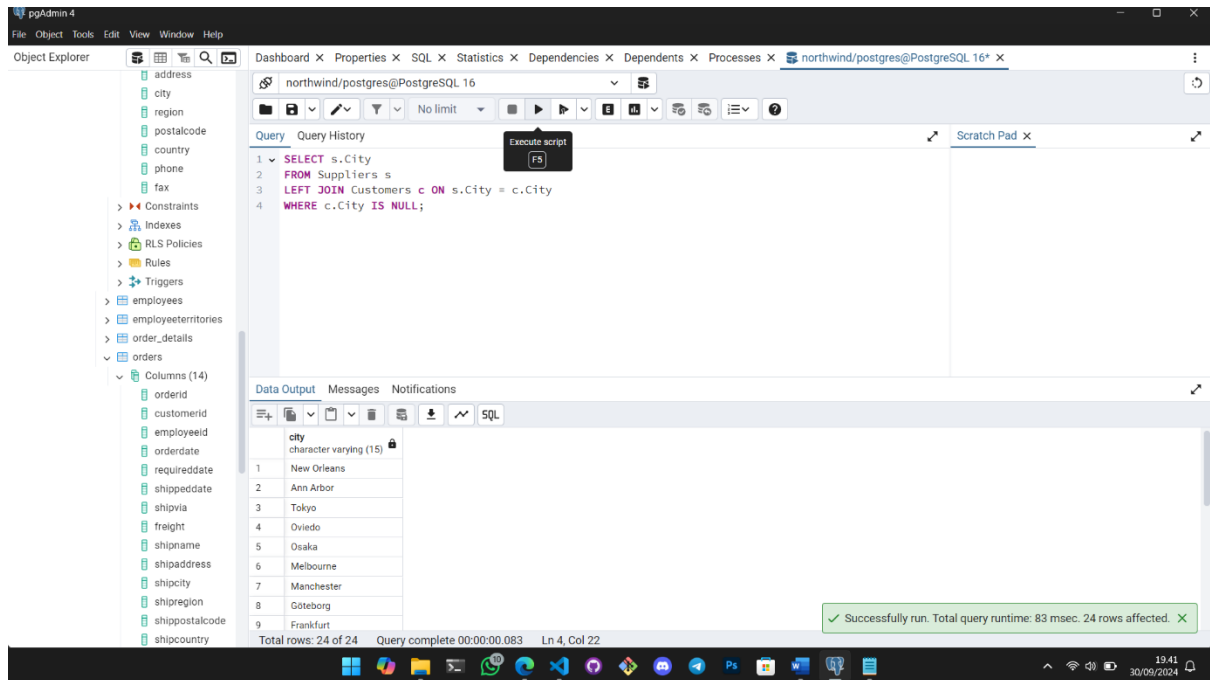
FROM Customers c

JOIN Suppliers s ON c.Country = s.Country;



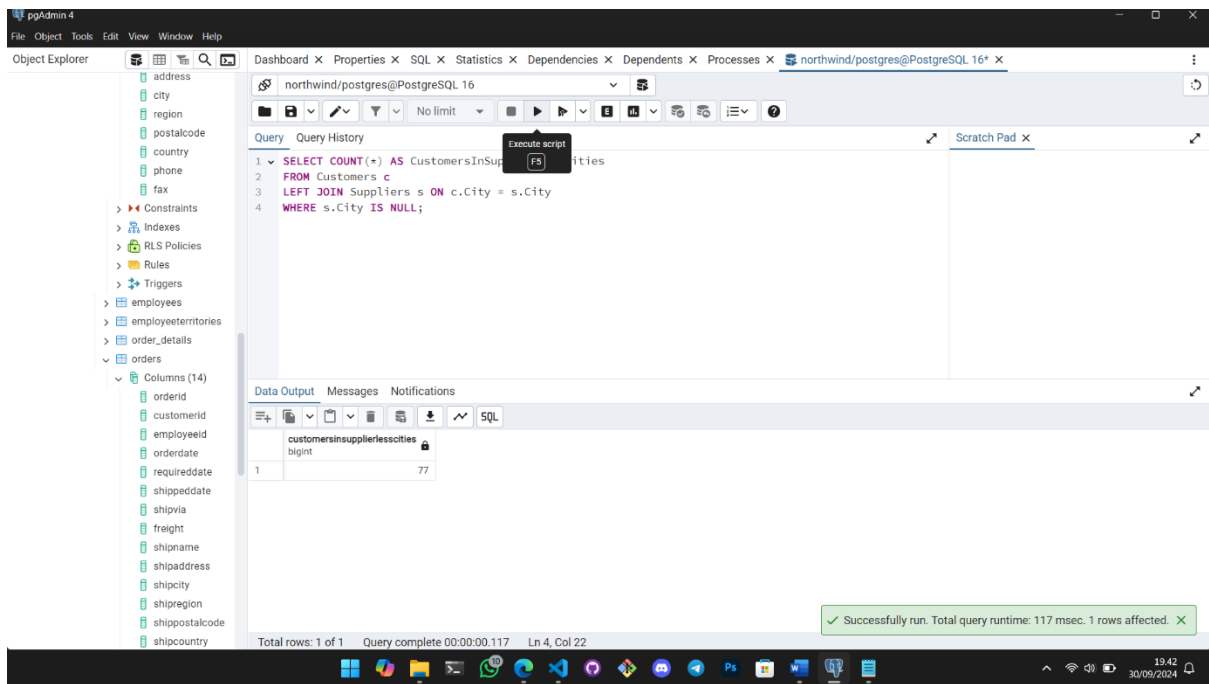
#### 4. Cities that have a supplier with no customer

```
SELECT s.City  
  
FROM Suppliers s  
  
LEFT JOIN Customers c ON s.City = c.City  
  
WHERE c.City IS NULL;
```



#### 5. How many customers do we have in cities without suppliers

```
SELECT COUNT(*) AS CustomersInSupplierlessCities  
  
FROM Customers c  
  
LEFT JOIN Suppliers s ON c.City = s.City  
  
WHERE s.City IS NULL;
```



## SQL 3.3

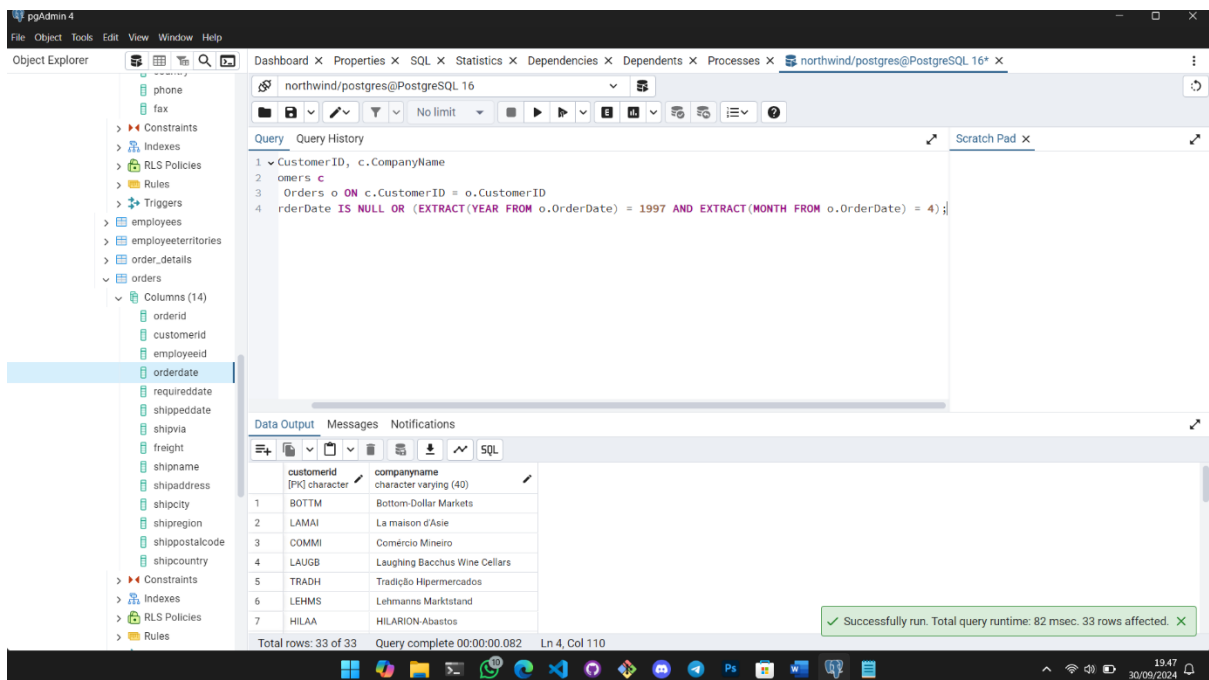
### 1. How do you find customers who didn't have an order in April, 1997

SELECT c.CustomerID, c.CompanyName

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE o.OrderDate IS NULL OR (EXTRACT(YEAR FROM o.OrderDate) = 1997 AND EXTRACT(MONTH FROM o.OrderDate) = 4);



### 2. Find all suppliers that have had an order with 1 item

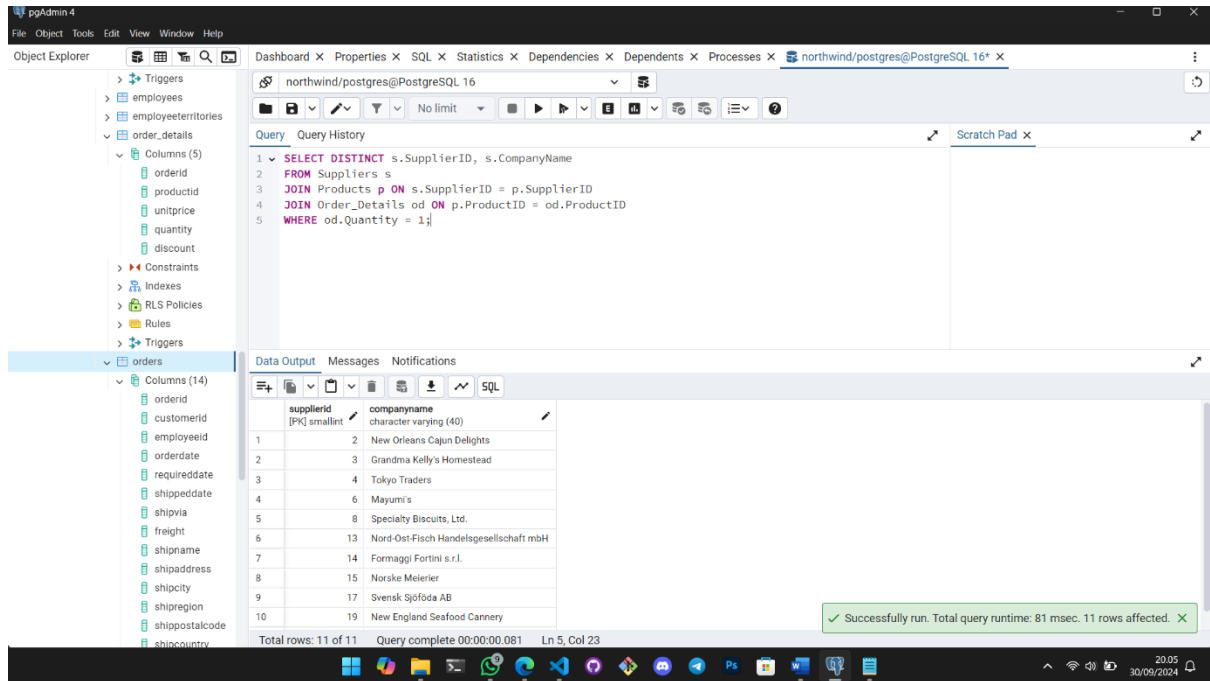
SELECT DISTINCT s.SupplierID, s.CompanyName

FROM Suppliers s

JOIN Products p ON s.SupplierID = p.SupplierID

JOIN Order\_Details od ON p.ProductID = od.ProductID

WHERE od.Quantity = 1;



3. Find all distinct customers that ordered more in one item than the average order amount per item of all customers

WITH AvgOrderItemQuantity AS (

SELECT AVG(od.Quantity) AS AvgQuantity

FROM Order\_Details od

)

SELECT DISTINCT c.CustomerID, c.CompanyName

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN Order\_Details od ON o.OrderID = od.OrderID

WHERE od.Quantity > (SELECT AvgQuantity FROM AvgOrderItemQuantity);



The screenshot shows the pgAdmin 4 interface with a query window open. The query is as follows:

```

1 WITH AvgOrderItemQuantity AS (
2   SELECT AVG(od.Quantity) AS AvgQuantity
3   FROM Order_Details od
4 )
5 SELECT DISTINCT c.CustomerID, c.CompanyName
6 FROM Customers c
7 JOIN Orders o ON c.CustomerID = o.CustomerID
8 JOIN Order_Details od ON o.OrderID = od.OrderID
9 WHERE od.Quantity > (SELECT AvgQuantity FROM AvgOrderItemQuantity);

```

The Data Output pane shows the results of the query:

customerid	companyname
1	FOLKO
2	HILAA
3	WARTH
4	SUPRD
5	QUICK
6	WHITC
7	BOTTM
8	OTTIK
9	TRADH
10	SEVES

Success message: Successfully run. Total query runtime: 88 msec. 66 rows affected.

#### 4. Find all suppliers that are in the same city as a customer

SELECT DISTINCT s.SupplierID, s.CompanyName

FROM Suppliers s

JOIN Customers c ON s.City = c.City;

The screenshot shows the pgAdmin 4 interface with a query window open. The query is as follows:

```

1 SELECT DISTINCT s.SupplierID, s.CompanyName
2 FROM Suppliers s
3 JOIN Customers c ON s.City = c.City;

```

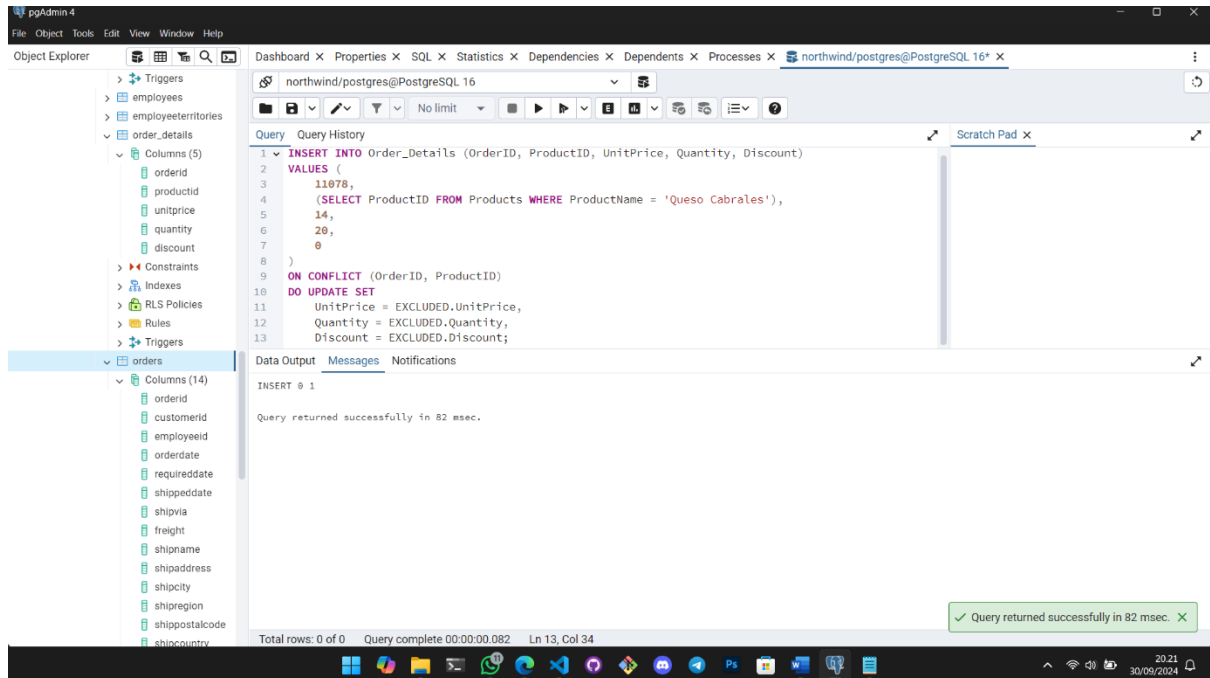
The Data Output pane shows the results of the query:

supplierid	companyname
11	Heli Süßwaren GmbH & Co. KG
25	Ma Maison
18	Aux joyeux ecclésiastiques
10	Refrescos Americanas LTDA
1	Exotic Liquids

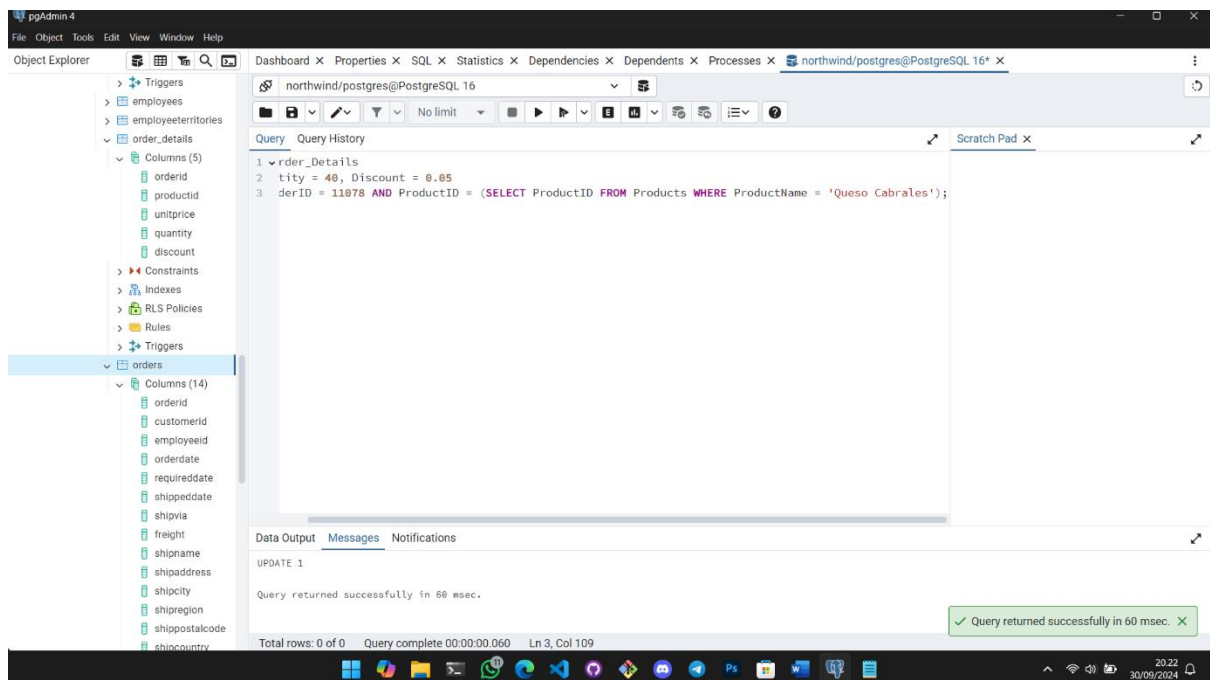
Success message: Successfully run. Total query runtime: 91 msec. 5 rows affected.

## SQL 3.4

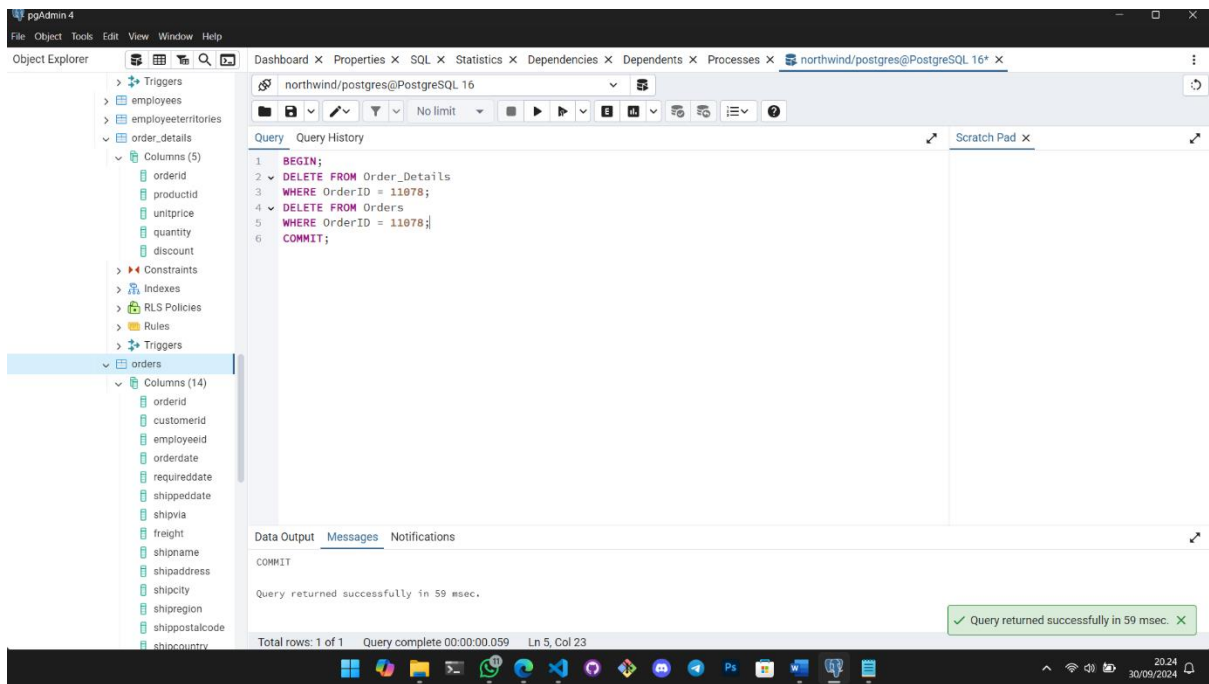
1. Insert an order detail for order we just created. Make it a quantity of 20 of Queso Cabrales (you will have to look up id) with a price of \$14



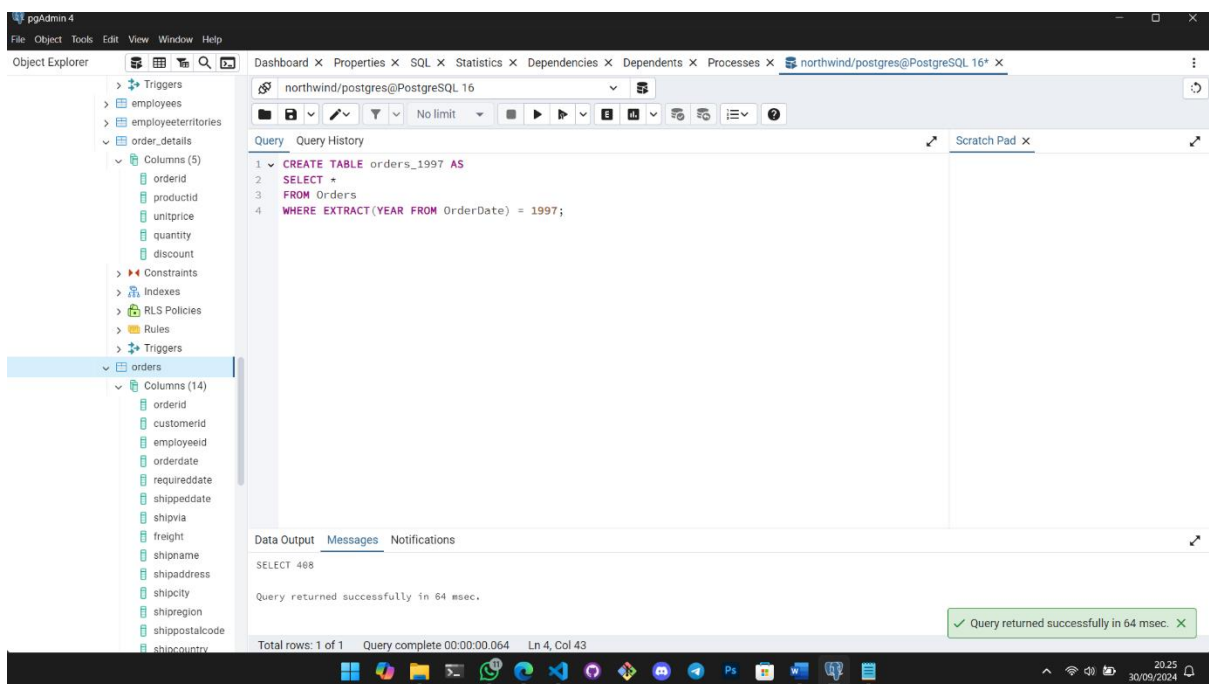
2. They also want 40 Queso Cabrales instead of 20 and we are giving a discount of 0.05. (trick is the WHERE clause to make sure we update the right order\_details since there is no order detail id field)



3. Delete the order for the customer using the order id 11078



#### 4. Backup orders in the year 1997 to a new table orders\_1997



#### 5. Add orders from December 2016 to table orders\_1997

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Triggers
- employees
- employeeterritories
- order\_details
  - Columns (5)
    - orderid
    - productid
    - unitprice
    - quantity
    - discount
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers
- orders
  - Columns (14)
    - orderid
    - customerid
    - employeeid
    - orderdate
    - requireddate
    - shippeddate
    - shipvia
    - freight
    - shipname
    - shipaddress
    - shipcity
    - shipregion
    - shippostalcode
    - shipcountry

Dashboard Properties SQL Statistics Dependencies Dependents Processes northwind/postgres@PostgreSQL 16\*

northwind/postgres@PostgreSQL 16

Query Query History

```
1 INSERT INTO orders_1997
2 SELECT *
3 FROM Orders
4 WHERE EXTRACT(YEAR FROM OrderDate) = 2016 AND EXTRACT(MONTH FROM OrderDate) = 12;
```

Scratch Pad

Data Output Messages Notifications

INSERT 0 0

Query returned successfully in 99 msec.

Query returned successfully in 99 msec.

Total rows: 1 of 1 Query complete 00:00:00.099 Ln 4, Col 82

20:25 30/09/2024