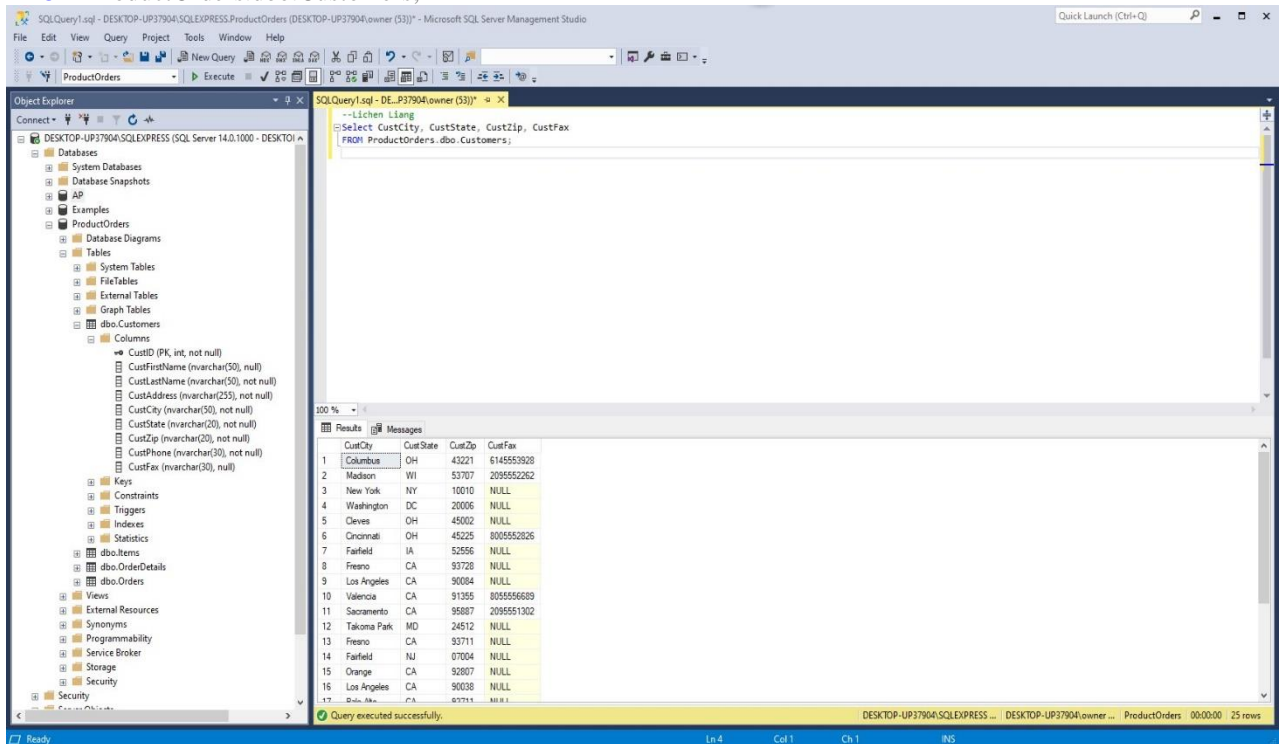
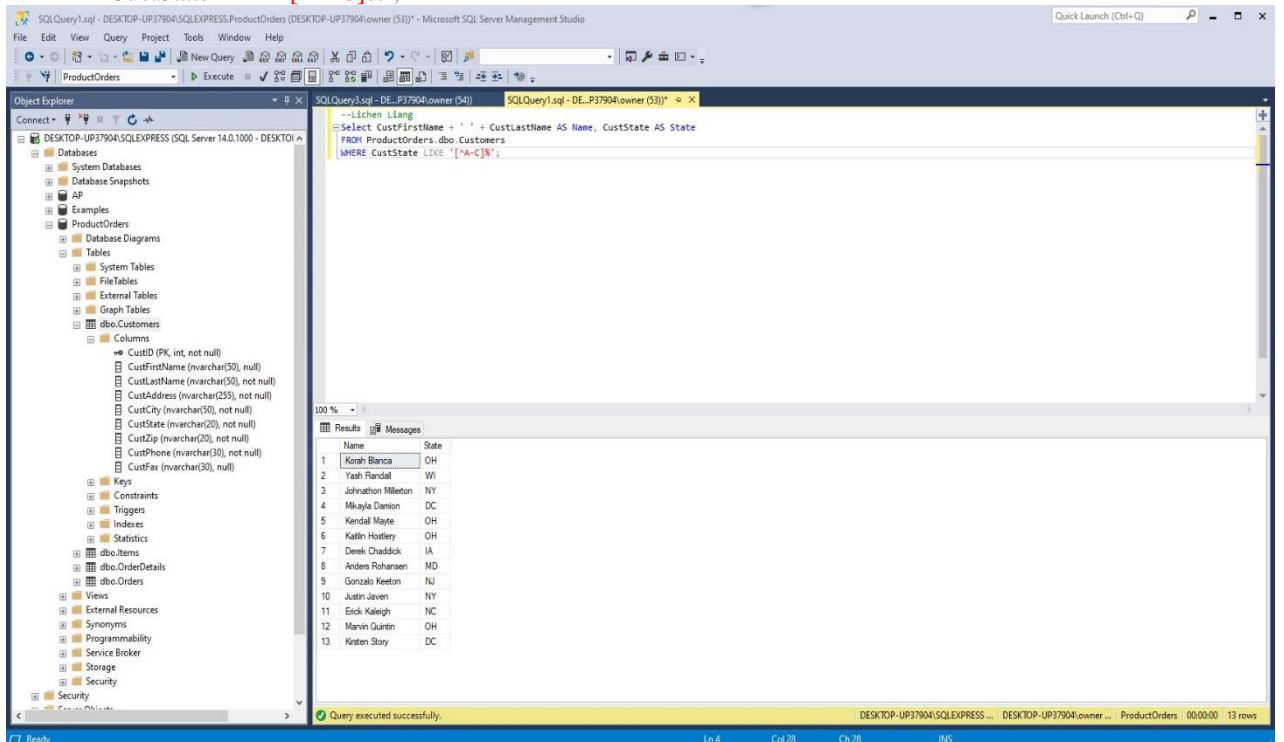


1. **Select** CustCity, CustState, CustZip, CustFax
FROM ProductOrders.dbo.Customers;



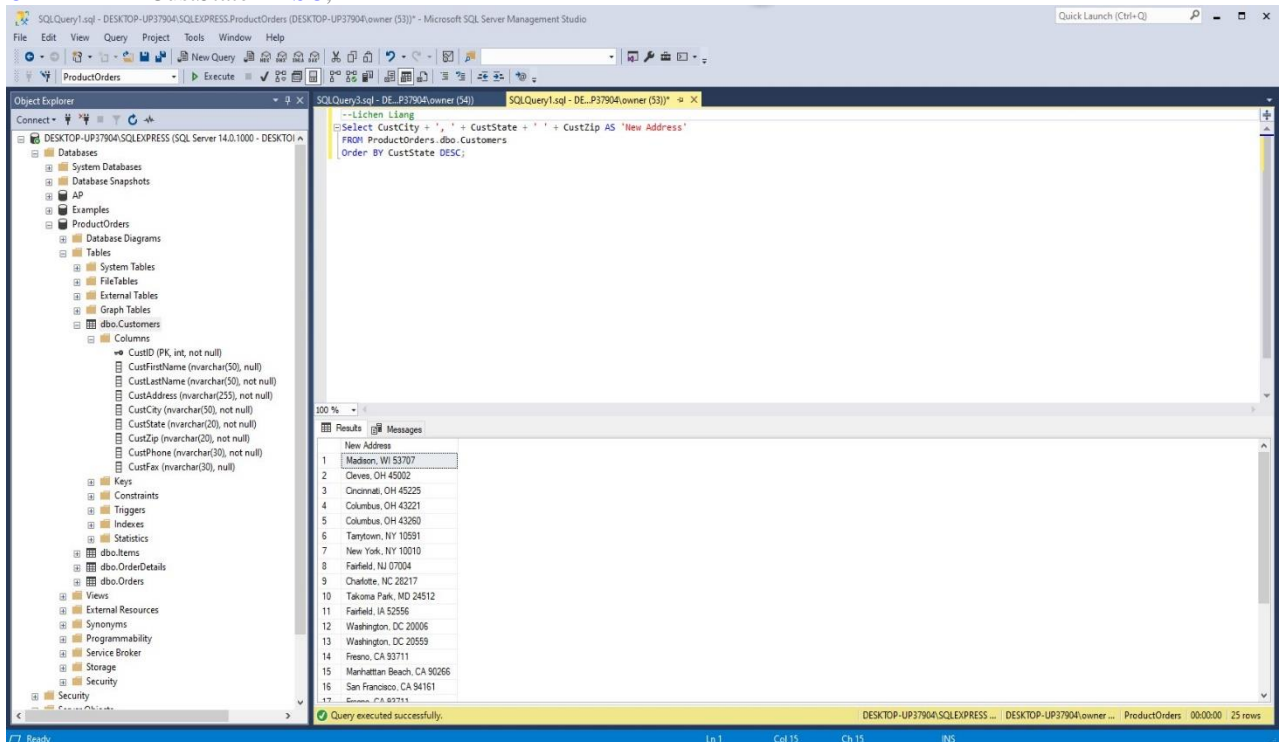
Select four columns from the table.

2. **Select** CustFirstName + ' ' + CustLastName **AS** Name, CustState **AS** State
FROM ProductOrders.dbo.Customers
WHERE CustState **LIKE** '[^A-C]%';



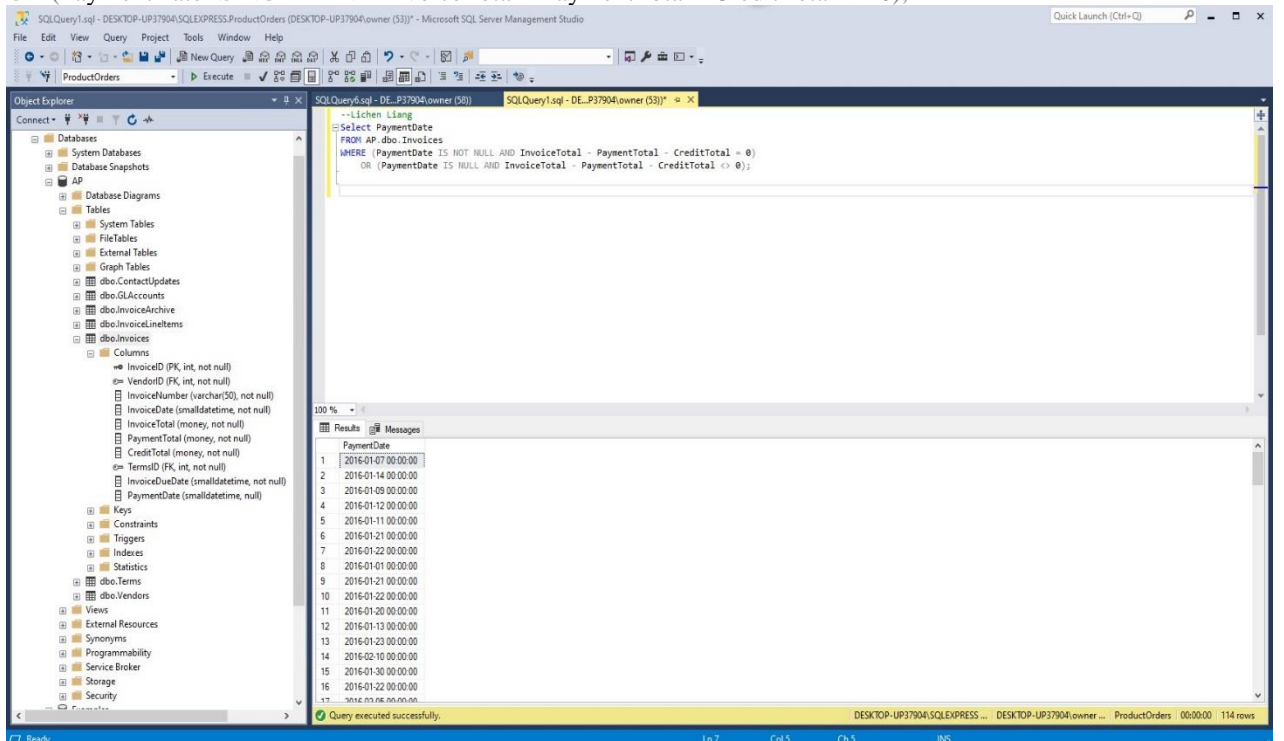
Concatenating two columns and giving them new column alias with specific requirement on the State column

- Select CustCity + ',' + CustState + ' ' + CustZip AS 'New Address'
FROM ProductOrders.dbo.Customers
ORDER BY CustState DESC;



Concatenating three columns, aliasing, and ordering.

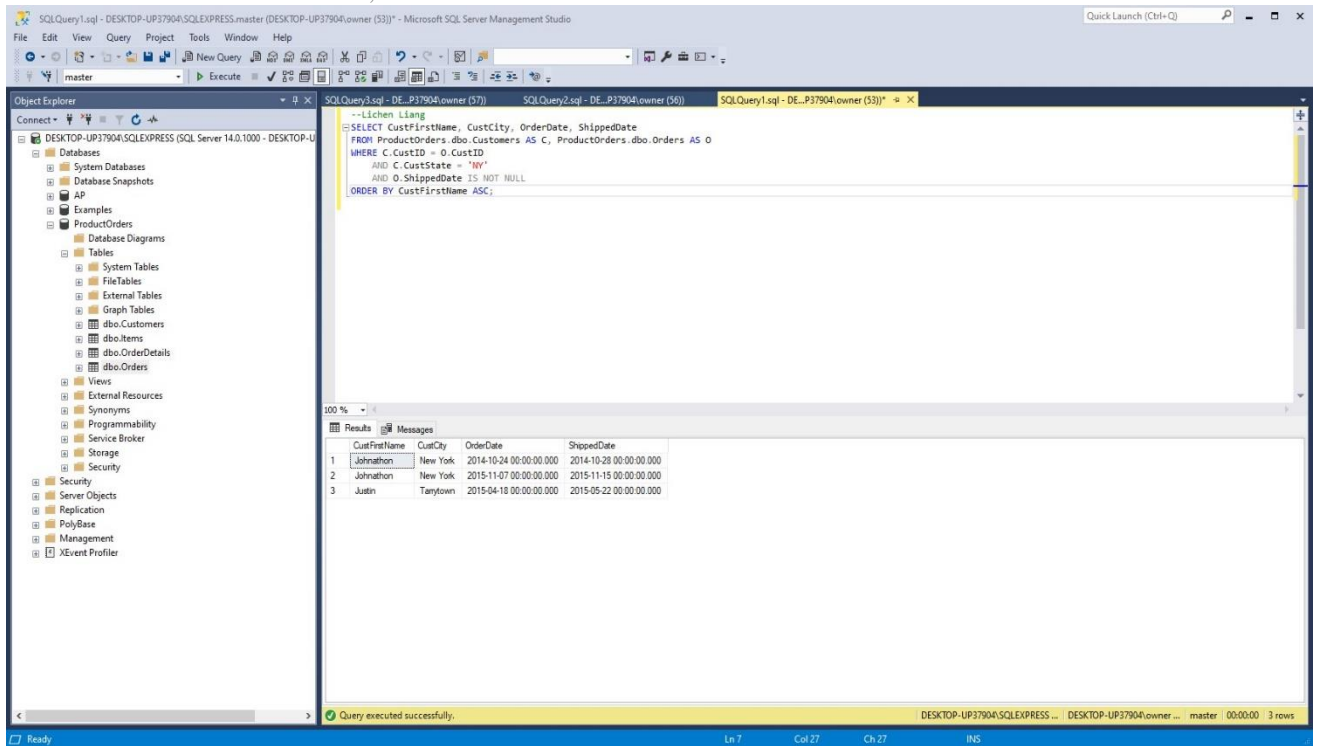
- Select PaymentDate
FROM AP.dbo.Invoices
WHERE (PaymentDate IS NOT NULL AND InvoiceTotal - PaymentTotal - CreditTotal = 0)
OR (PaymentDate IS NULL AND InvoiceTotal - PaymentTotal - CreditTotal <> 0);



Looking for valid entries given specific condition

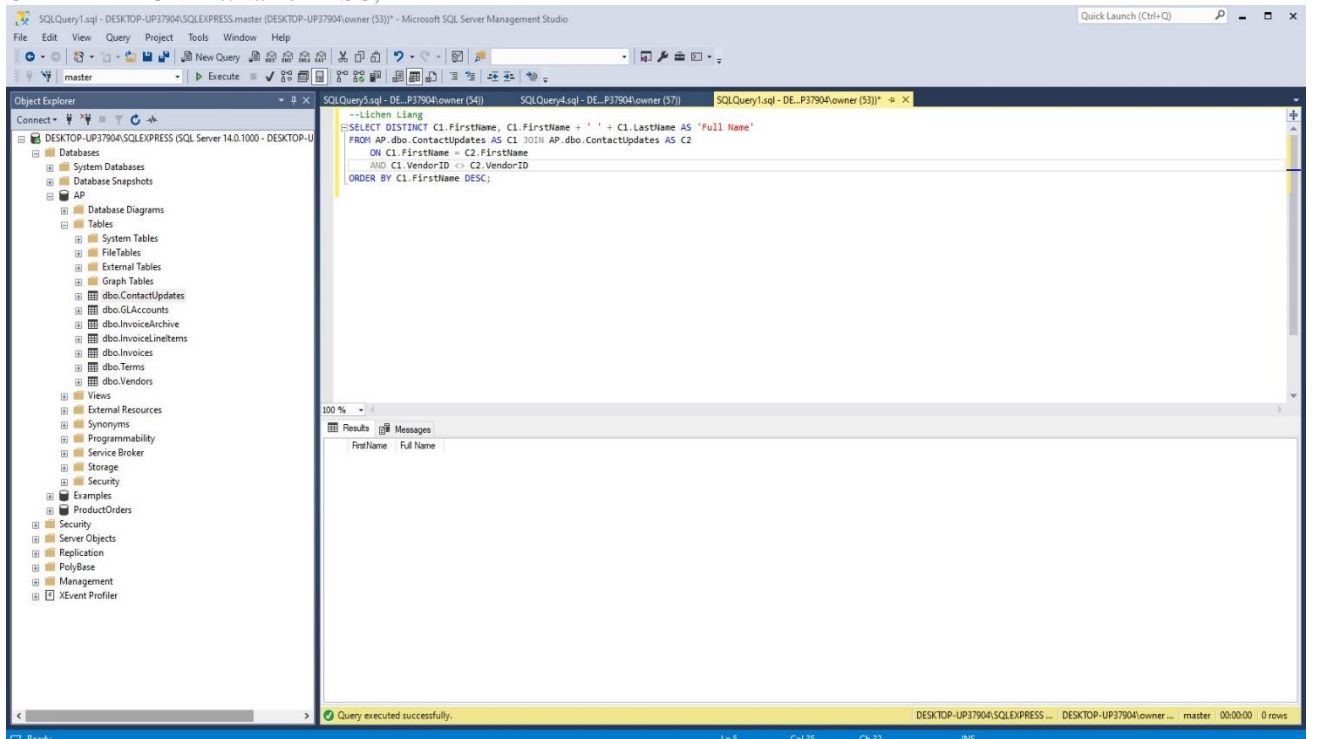
The output shows the valid entries.

- SELECT** CustFirstName, CustCity, OrderDate, ShippedDate
FROM ProductOrders.dbo.Customers **AS** C, ProductOrders.dbo.Orders **AS** O
WHERE C.CustID = O.CustID
 AND C.CustState = 'NY'
 AND O.ShippedDate IS NOT NULL
ORDER BY CustFirstName **ASC**;



Matching the columns from two different tables, given conditions, order.

- SELECT DISTINCT** C1.FirstName, C1.FirstName + ' ' + C1.LastName **AS** 'Full Name'
FROM AP.dbo.ContactUpdates **AS** C1 **JOIN** AP.dbo.ContactUpdates **AS** C2
 ON C1.FirstName = C2.FirstName
 AND C1.VendorID <> C2.VendorID
ORDER BY C1.FirstName **DESC**;



Looking for people with same first name and different last name from the same table.

There are no people who have the same first name.

7. `SELECT CustLastName, 'Columbus' AS CustCity
From ProductOrders.dbo.Customers
WHERE CustCity = 'Columbus'
UNION
SELECT CustLastName, 'Not in Co' AS CustCity
From ProductOrders.dbo.Customers
WHERE CustCity <> 'Columbus'
ORDER BY CustLastName ASC;`

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'ProductOrders' database selected. The right pane shows a SQL query window with the following code:

```
--Lichen Liang  
SELECT CustLastName, 'Columbus' AS CustCity  
From ProductOrders.dbo.Customers  
WHERE CustCity = 'Columbus'  
  
UNION  
  
SELECT CustLastName, 'Not in Co' AS CustCity  
From ProductOrders.dbo.Customers  
WHERE CustCity <> 'Columbus'  
  
ORDER BY CustLastName ASC;
```

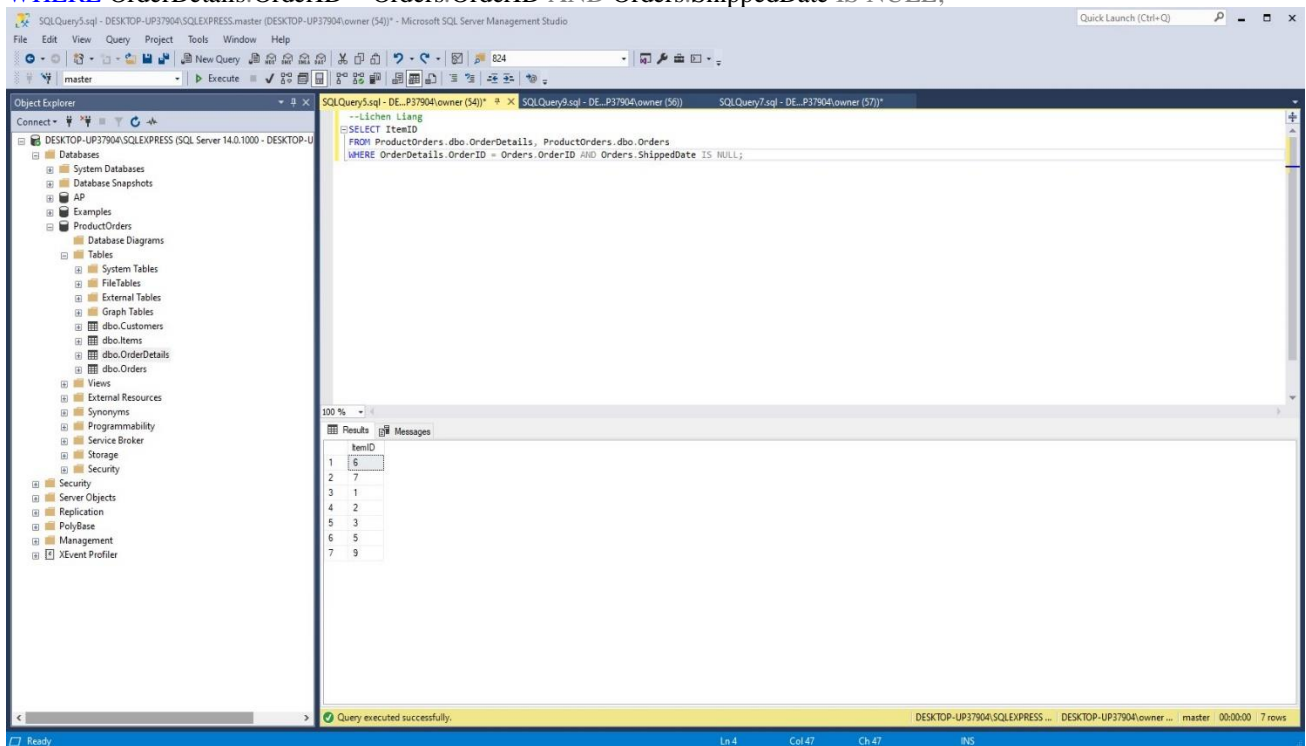
Below the query window, the 'Results' pane displays the output of the query as a table with two columns: 'CustLastName' and 'CustCity'. The table contains 16 rows of data, showing customers and their corresponding city labels.

CustLastName	CustCity
Anum	Not in Co
Baylee	Not in Co
Blanca	Columbus
Carson	Not in Co
Chaddock	Not in Co
Damien	Not in Co
Danion	Not in Co
Eulala	Not in Co
Holbrooke	Not in Co
Hostery	Not in Co
Ivin	Not in Co
Jacobsen	Not in Co
Javen	Not in Co
Kaleigh	Not in Co
Keston	Not in Co
Lacy	Not in Co
Mishra	Not in Co

The status bar at the bottom indicates that the query was executed successfully, returning 25 rows.

Listing people who live and don't live in Columbus.

8. **SELECT** ItemID
FROM ProductOrders.dbo.OrderDetails, ProductOrders.dbo.Orders
WHERE OrderDetails.OrderID = Orders.OrderID **AND** Orders.ShippedDate **IS NULL**;



Query data that has underlying conditions of which those are not displayed.

Remarks

In this lab we practiced query using different databases, tables, columns, etc. In the query, we learned how to use joins, unions, where, and order by, for setting the conditions for a query that meets the lab requirement. I think this lab is very efficient practice for getting used to SQL language. The challenge would be more specific requirements.