LAB 2

**1.** Write a SELECT statement that returns three columns from the Employees table:

LastName, FirstName, and DeptNo. Use Examples database.

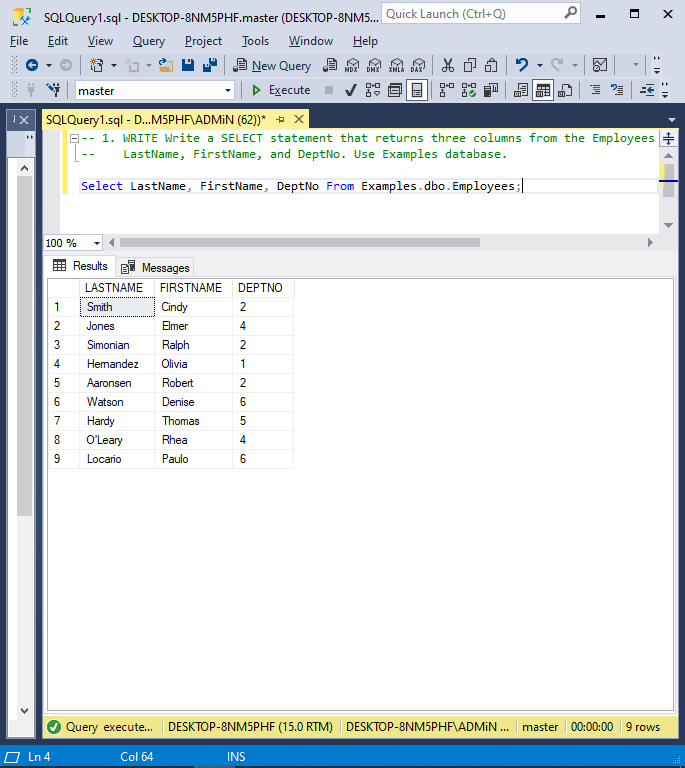
**Comment:**

The SELECT statement is used in the following query to display all the rows from three columns namely LastName, FirstName and DeptNo for Employees table contained in the Examples database, and when executing the source code, *9 rows* are returned as an output.

**Query:**

Select LastName, FirstName, DeptNo From Examples.dbo.Employees;

**Output:**



**2.** Write a SELECT statement that returns two columns from the Employees table, named ‘Name’, and ‘DeptNumber’: Name Column alias for the concatenated format of LastName and FirstName columns (Format: LastName followed by comma and space followed by FirstName) DeptNumber Column alias for the DeptNo column And filter for Employees with DeptNo value as 4. *Use Examples database.*

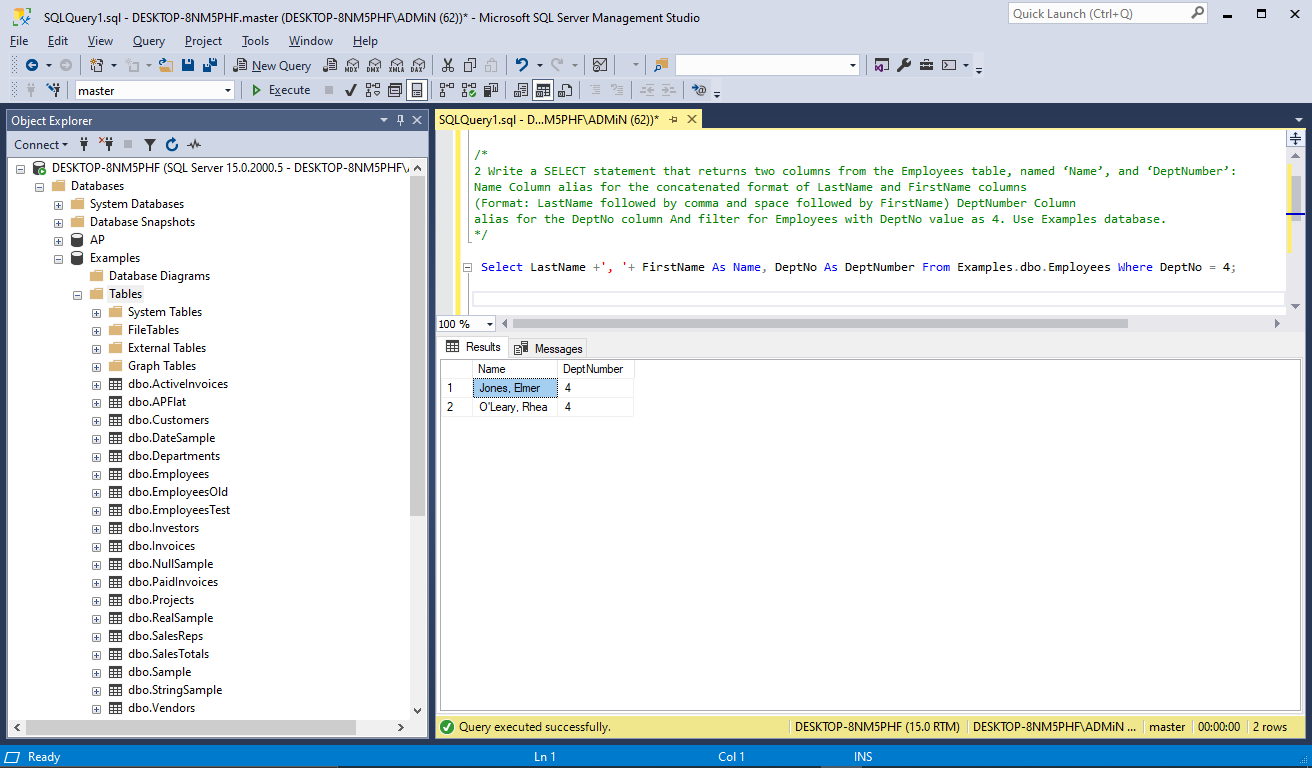
**Comment:**

* The following query utilizes the SELECT statement to retrieve the ‘Name’ and ‘DeptNumber’ columns of just those rows from the Employees table of the Examples database whose DeptNo is equal to 4.
* Furthermore, Name is an alias for the concatenated format of LastName and FirstName columns, and DeptNumber is an alias for the DeptNo column.
* ‘AS’ keyword is used to provide an alias and Concatenation was carried out using a concatination operator, which is represented by using ‘+’.
* *2 rows* are returned after executing the source code.

**Query:**

Select LastName + ', ' + FirstName As Name, DeptNo As DeptNumber From Examples.dbo.Employees Where DeptNo = 4;

**Output:**



**3**. Write a SELECT statement that returns one column from the Customers table named “Full Name”. Create this column from the CustomerFirst and CustomerLast columns. Format it as follows: CustomerLast, comma, space, CustomerFirst. Sort the result set by CustomerLast from “A-Z”. Use Examples database.

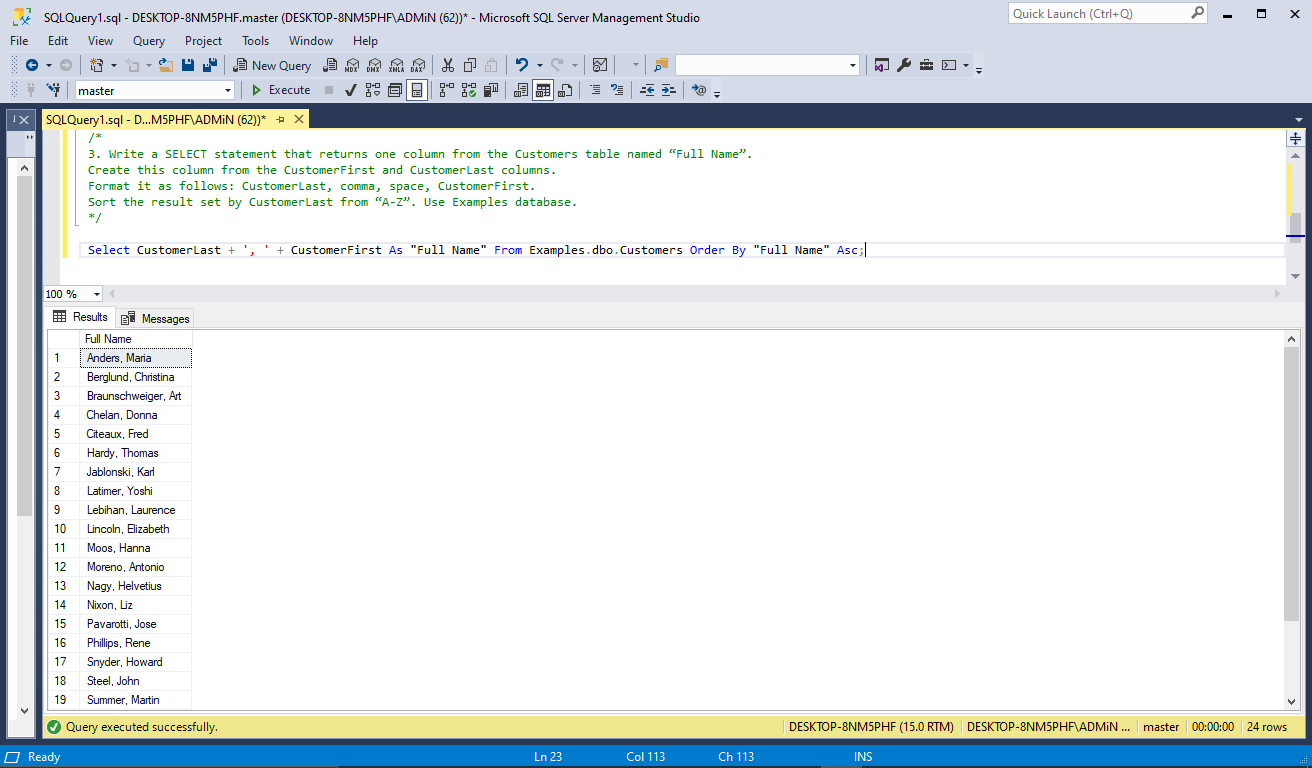
**Comment:**

* The SELECT statement is used in the following query to retrieve the ‘Full Name’ column in ascending order from the Customers Table of the Examples Database.
* The alias ‘Full Name’ is a concatenated format of CustomerFirst and CustomerLast columns separated by a comma and a space by using the Concatenation Operator, which is represented by using ‘+’.
* ‘Order By’ Keyword is used to specify the table’s order and ‘ASC’ keyword is used meet the Ascending condition.
* However, since the order is in ascending format by default, it is not necessary to include the ASC keyword.
* *24 Rows* are returned by running the source code.

**Query:**

Select CustomerLast + ', ' + CustomerFirst As "Full Name" From Examples.dbo.Customers Order By "Full Name" Asc;

**Output:**



**4.** Write a SELECT statement that determines whether the PaymentDate column of the Invoices table has any valid values.To be valid, PaymentDate must be a non-null value if there is no balance due, and a null value if there is balance due. Code a compound condition in the WHERE clause that tests for these conditions. (Balance: InvoiceTotal minus the sum of PaymentTotal and CreditTotal). Use AP database.

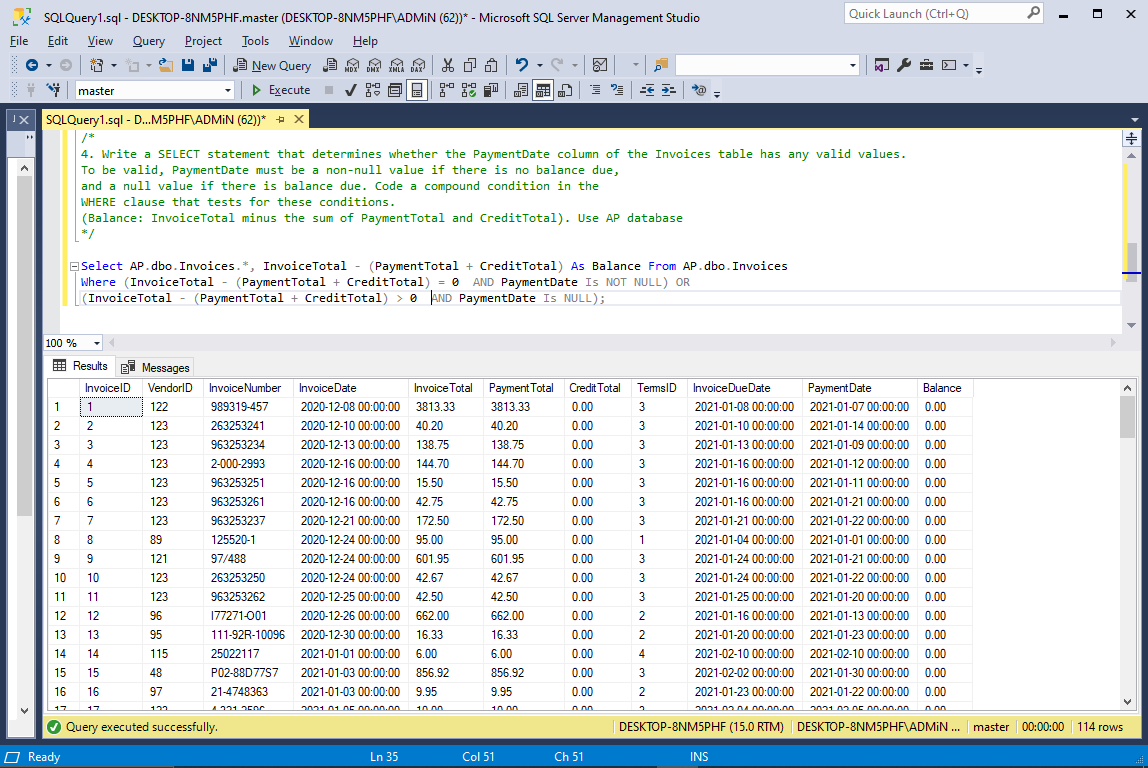
**Comment:**

* The PaymentDate column of the InvoiceTable from AP database is checked by using WHERE clause to see whether any valid values exist in it i.e., whether or not the Balance due is null depending upon the result of arithmetic and logical operations performed on other three associated columns.
* *114 Rows* are retrieved after running the Source Code.

**Query:**

Select AP.dbo.Invoices.\*, InvoiceTotal - (PaymentTotal + CreditTotal) As Balance From AP.dbo.Invoices Where (InvoiceTotal - (PaymentTotal + CreditTotal) = 0 AND PaymentDate Is NOT NULL) OR (InvoiceTotal - (PaymentTotal + CreditTotal) > 0 AND PaymentDate Is NULL);

**Output:**



**5.** Write a SELECT statement that returns five columns: CustLastName, CustCity, CustState, OrderDate and ShippedDate from the Customers table and Orders table. The result set should have one row for each customer, with the city, order date and shipped date for that customer’s ID. Filter for Customers whose CustState is ‘MA’ and ShippedDate is null. Sort the result set by CustLastName from A to Z. Use ProductOrders database.

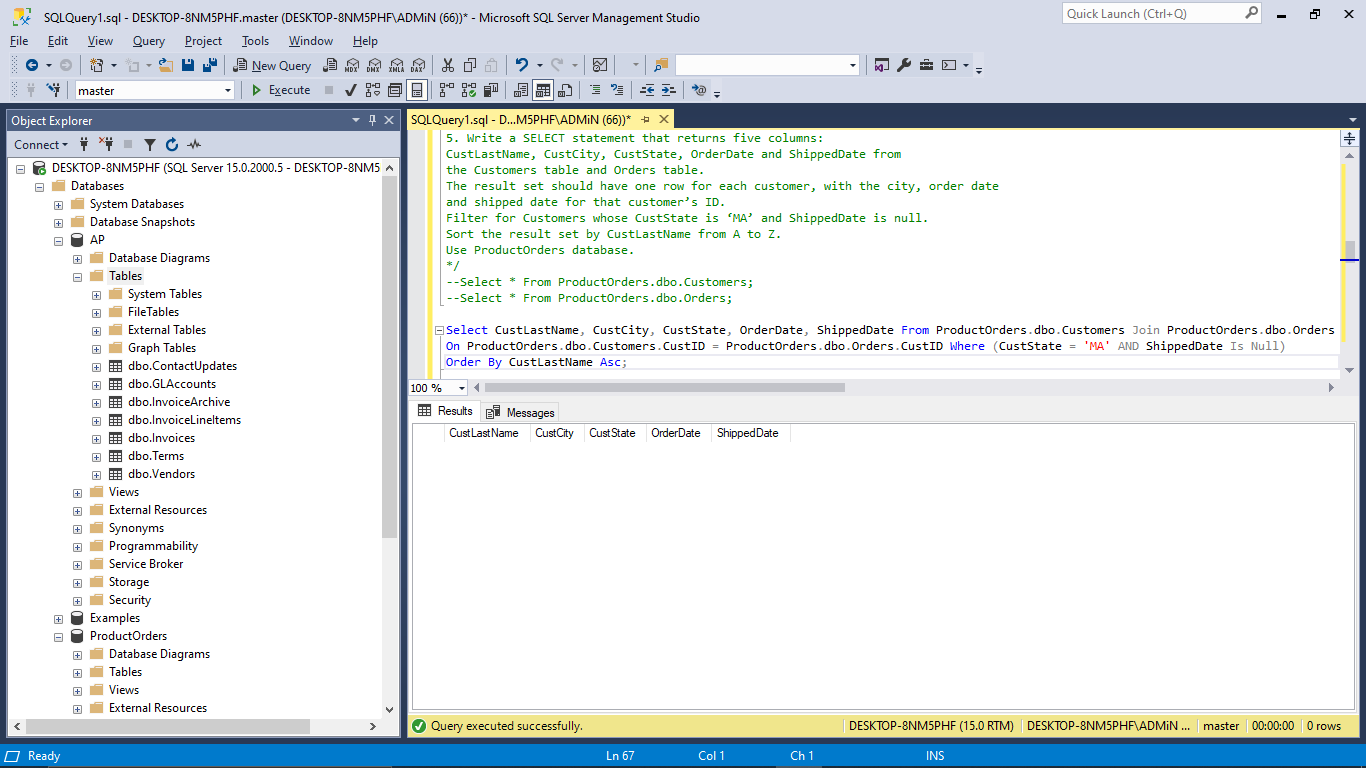
**Comment:**

* The Customers table and the Orders Table from ProductOrders database are combined togather using inner join i.e. JOIN clause because this query requires to display data, namely CustLastName, CustCity, CustState, OrderDate and ShippedDate from two different tables, of which OrderDate and ShippedDate are present only in Orders Table.
* Because both the tables have a column called ‘OrderID’, which works as bridge between them, linking both the tables together. (works as a Primary Key in Orders table and as a Foreign Key in Customers table)
* After joining both tables, the requirement with a ‘NULL’ shippedDate and a CustState of ‘MA’ is checked.
* However, there is no such customer who satisfies both the condition so there are *no rows* displayed in the result set.

**Query:**

Select CustLastName, CustCity, CustState, OrderDate, ShippedDate From ProductOrders.dbo.Customers Join ProductOrders.dbo.Orders On ProductOrders.dbo.Customers.CustID = ProductOrders.dbo.Orders.CustID Where (CustState = 'MA' AND ShippedDate Is Null) Order By CustLastName;

**Output:**



**6.** Write a SELECT statement that returns two columns: VendorName and FullName

(A concatenation of VendorContactLName and VendorContactFName, with a space in between). The result set should have one row for each vendor whose contact has the same first name (i.e. VendorContactFName) as another vendor's contact. Sort the final result set by FullName column from Z to A. Use AP database.

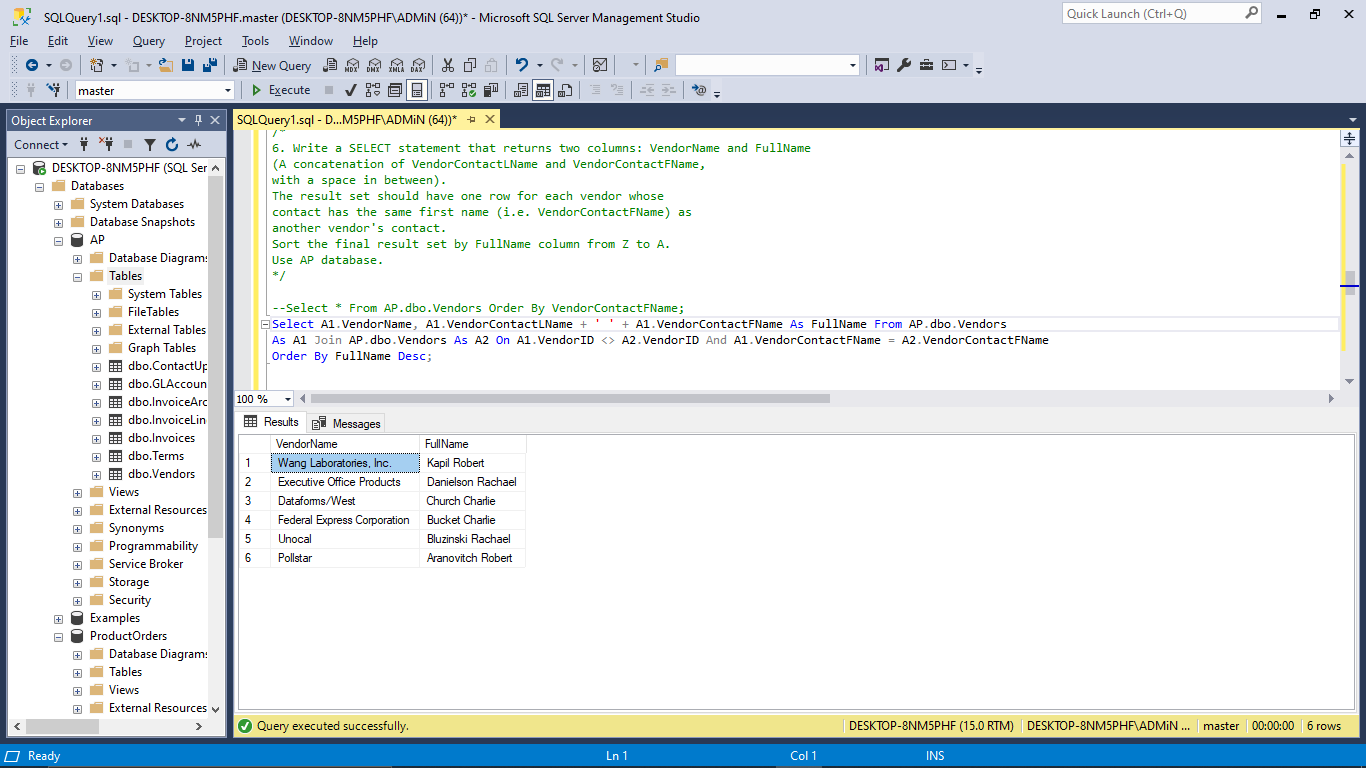
**Comment:**

* Two Columns are displayed using the SELECT statement: VendorName and “FullName”, where “FullName” is an alias created by concatenating VendorContactLName and VendorContactFName with a space between them.
* The vendors table is linked to itself using SELF JOIN, which is given the aliases A1 and A2, to display the vendors whose contact’s firstname is similar to another contact’s first name and JOIN clause is used to join the vendor table to itself.
* To achieve the desired output, the VendorID of A1 should not be equal the VendorID of A2.Instead, the firstname of the contact in one table should match the firstname of the contact in other table.
* These two conditions were combined using the logical operator “AND” and were applied using the ON clause.
* As a result, when the source code was executed, only *6 rows* were returned that met both the requirements.

**Query:**

Select A1.VendorName, A1.VendorContactLName + ' ' + A1.VendorContactFName As FullName From AP.dbo.Vendors As A1 Join AP.dbo.Vendors As A2 On A1.VendorID <> A2.VendorID And A1.VendorContactFName = A2.VendorContactFName Order By FullName Desc;

**Output:**



**7.** Use the UNION operator to generate a result set consisting of two columns from the Customers table: CustomerFirst and CustState. If the customer is in Illinois, the CustState value should be “IL”; otherwise, the CustState value should be “Not in IL”. Sort the final result set by CustomerFirst from Z-A. Use Examples database.

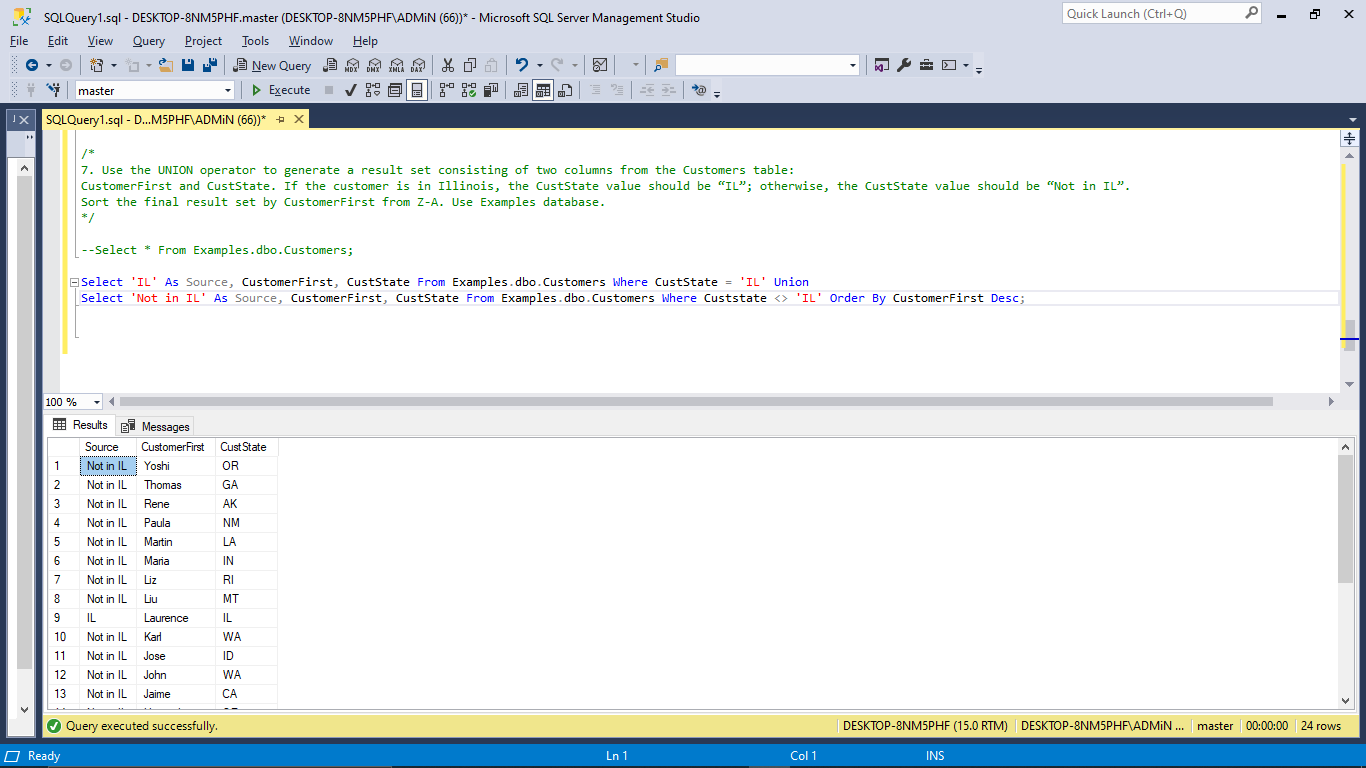
**Comment:**

* In this query, as the requirement is to show ‘IL’ if the customer is in Illinois else ‘Not in IL’, it is not feasible to display it in the same select query because then it will produce two different columns; one with just ‘IL’ ones and other with the latter one.
* Hence, to achieve the desired output, two select queries were generated separately and combined togather with the UNION operator to show not just the texts under the same column called SOURCE but to also display the rows with distinct values.
* As a result, this query generates three columns namely Source, CustomerFirst and CustState from Customers table of Examples database.
* After running this code, *24 rows* are returned as an output.

**Query:**

Select 'IL' As Source, CustomerFirst, CustState From Examples.dbo.Customers Where CustState = 'IL' Union Select 'Not in IL' As Source, CustomerFirst, CustState From Examples.dbo.Customers Where Custstate <> 'IL' Order By CustomerFirst Desc;

**Output:**



**8.** Write a SELECT statement that returns two columns from the GLAccounts table: AccountNo and AccountDescription. The result set should have one row for each account number that has never been used (i.e. AccountNo in InvoiceLineItems table has null value). Sort the final result set by AccountNo in descending order. Use AP database. (HINT: Join GLAccounts table and InvoiceLineItems table.)

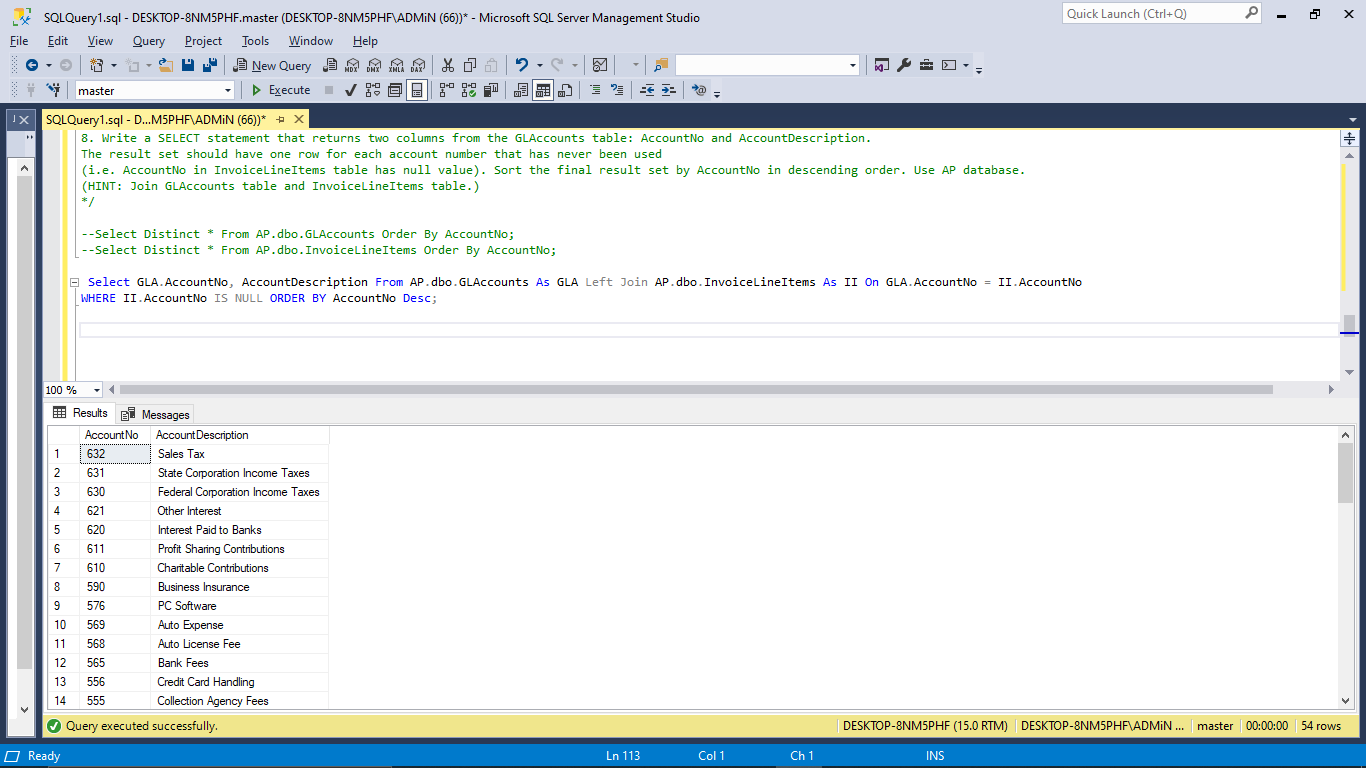
**Comments:**

* SELECT statement is used that returns two columns from the GLAccounts table: AccountNo and AccountDescription.
* In order to achieve the requirement i.e. to display those account numbers who are present in the GLAccount table but not used in Invoice table, a join was required between the two tables.
* However, just the account number was needed to be retrieved to compare those who do not exist in the invoice table so, Left join is used to join the two tables instead of inner join, and the requirement that both tables have the same account number and that the account number is NULL is then added.
* Because, when the left join is applied, the account numbers which are not utilized in the invoice table will display NULL in their account number row which will help in retrieving the unutilized numbers easily.
* Hence, as a result *54 rows* are returned as an output.

**Query:** Select GLA.AccountNo, AccountDescription From AP.dbo.GLAccounts As GLA Left Join AP.dbo.InvoiceLineItems As II On GLA.AccountNo = II.AccountNo

WHERE II.AccountNo IS NULL ORDER BY AccountNo Desc;

**Output:**



## Remarks of the Lab:

The overall lab consisted the utilization of several data retrieval clauses and statements like JOIN, UNION, WHERE, FROM, SELECT as well as the Arithmetic Operators and logical operators, and, so on.