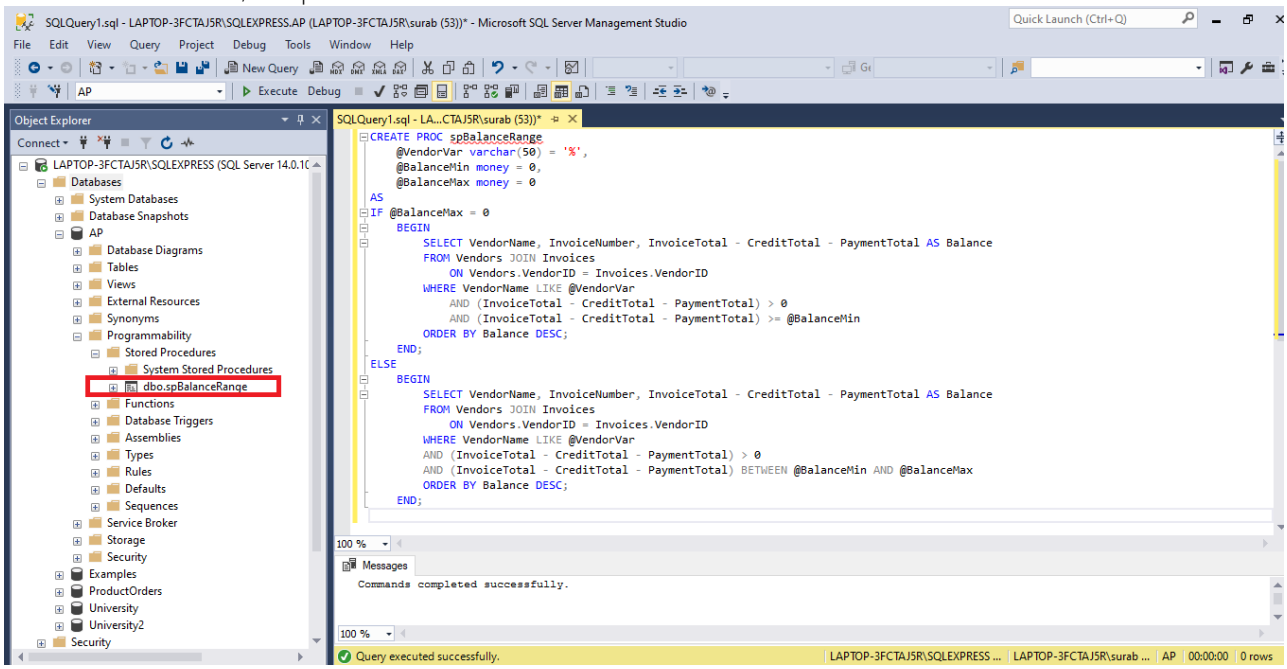
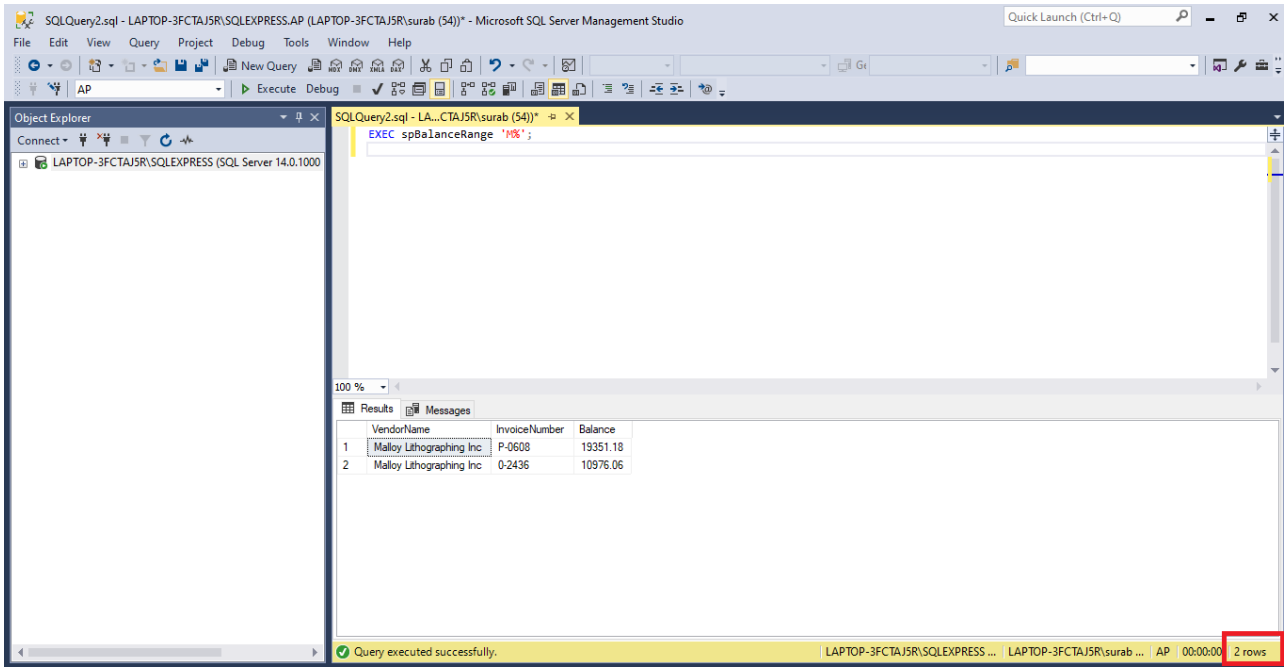
1. Create a stored procedure named spBalanceRange that accepts three optional parameters. The procedure should return a result set consisting of VendorName, InvoiceNumber, and Balance for each invoice with a balance due, sorted with largest balance due first. The parameter @VendorVar is a mask that's used with a LIKE operator to filter by vendor name. @BalanceMin and @BalanceMax are parameters used to specify the requested range of balances due. If called with no parameters or with a maximum value of 0, the procedure should return all invoices with a balance due.

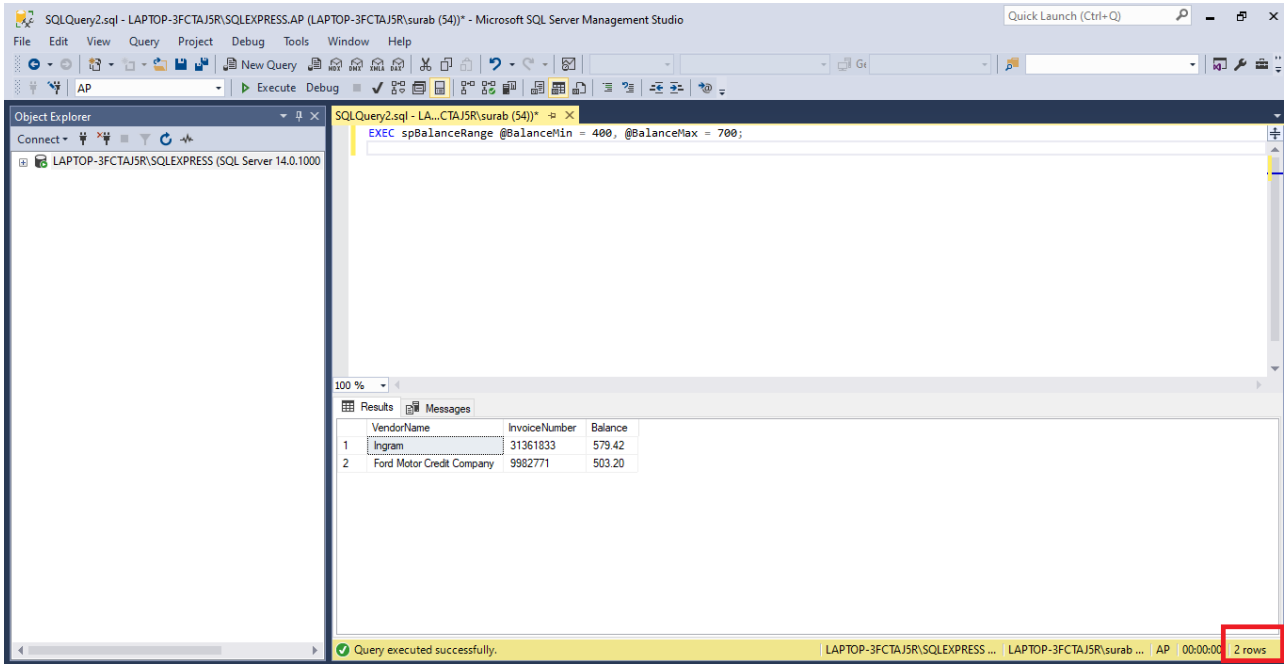



2. Code three calls to the procedure created in question 1:

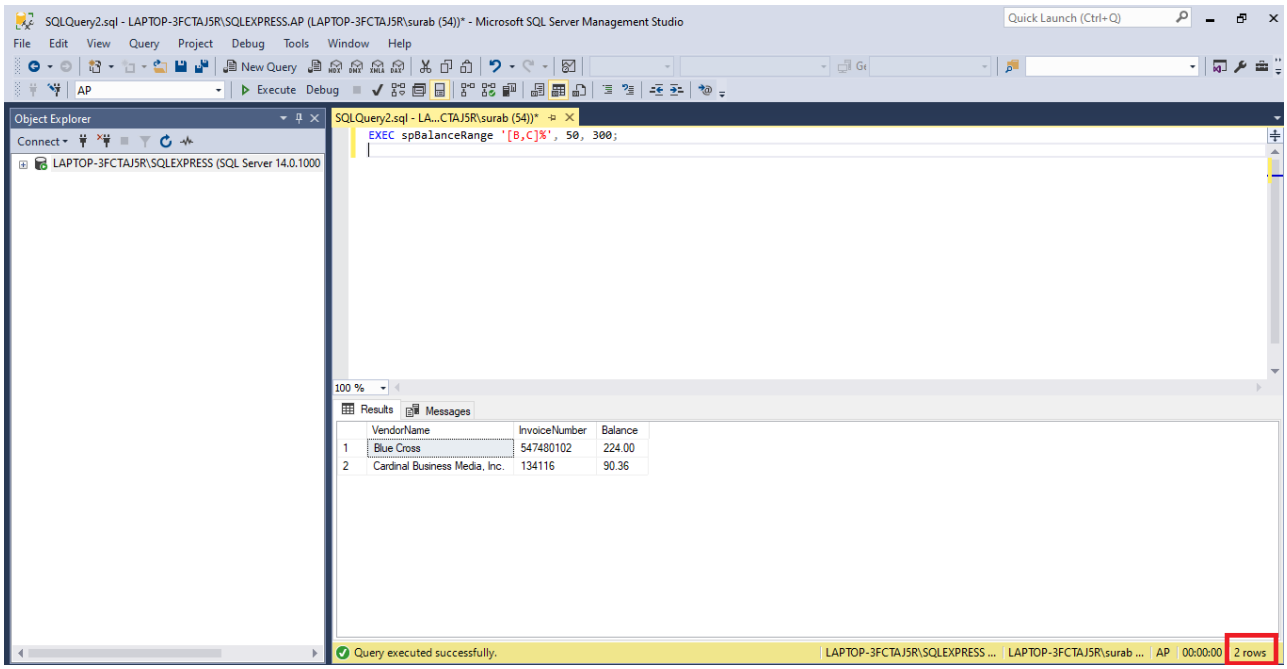
(a) passed by position with @VendorVar = 'M%' and no balance range



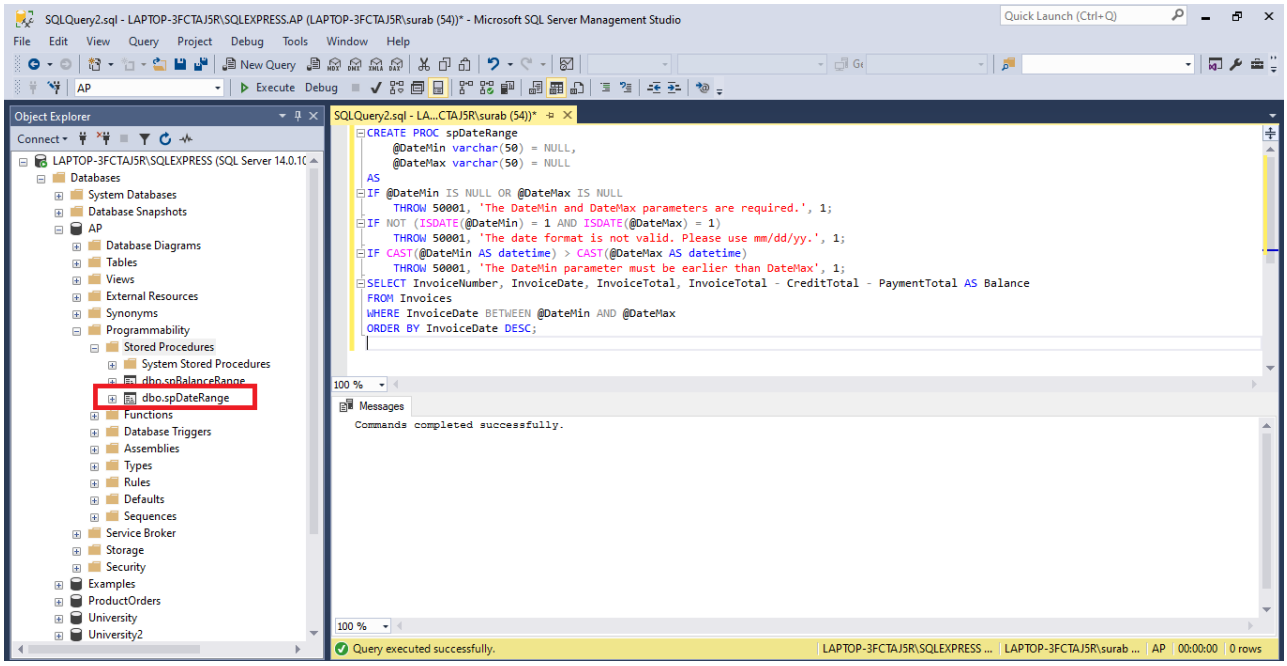
(b) passed by name with @VendorVar omitted and a balance range from $400 to $700




(c) passed by position with a balance due from $50 to $300, filtering for vendors whose names begin with B or C

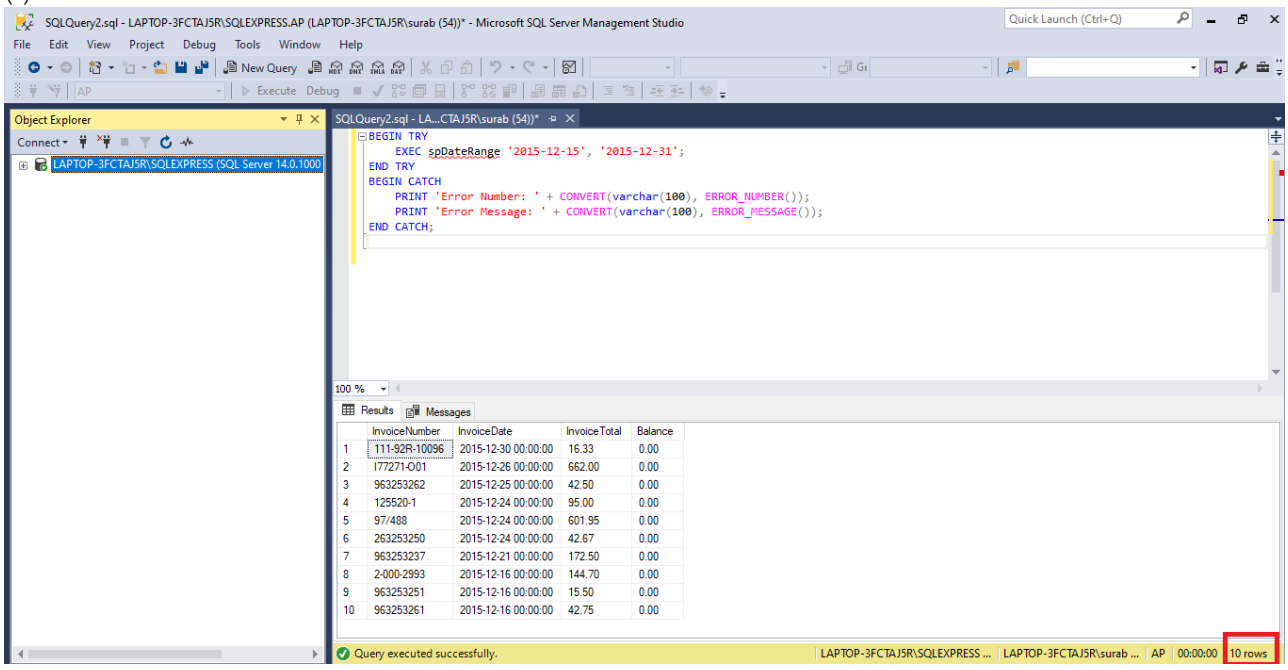



3. Create a stored procedure named spDateRange that accepts two parameters, @DateMin and @DateMax, with data type varchar and default value null. If called with no parameters or with null values, raise an error that describes the problem. If called with non-null values, validate the parameters. Test that the literal strings are valid dates and test that @DateMin is earlier than @DateMax. If the parameters are valid, return a result set that includes the InvoiceNumber, InvoiceDate, InvoiceTotal, and Balance for each invoice for which the InvoiceDate is within the date range, sorted with earliest invoice last.

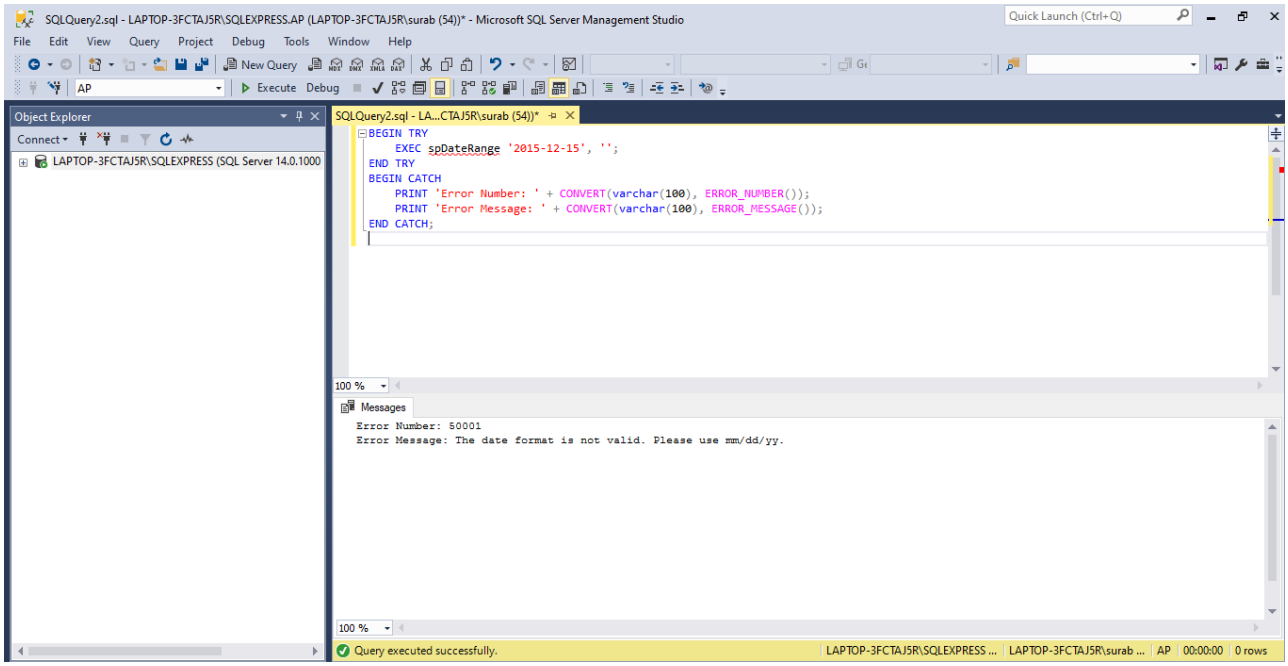



4. Code (1) a call to the stored procedure created in question 3 that returns invoices with an InvoiceDate between December 15 and December 31, 2015, (2) a call to the stored procedure again that returns invoices with an @DateMin is December 15. These calls should also catch any errors that are raised by the procedure and print the error number and description.

(1)



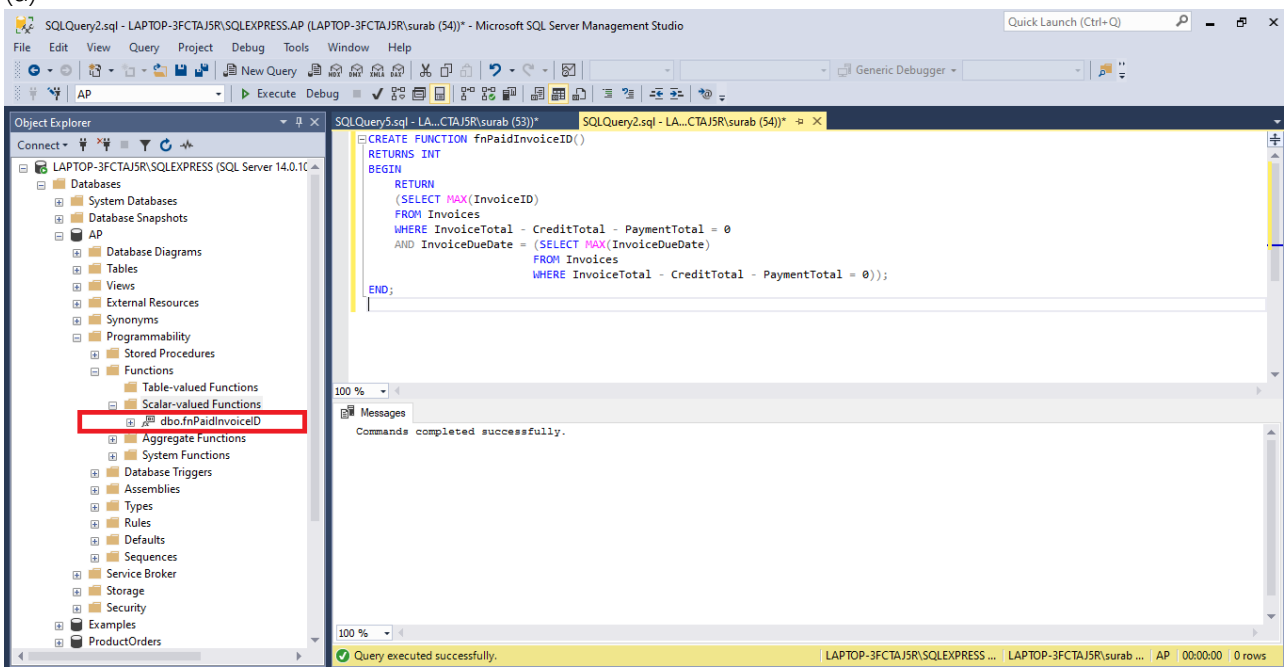

(2)


SQLQuery2.sql - LAPTOP-3FCTAJ5R\SQLEXPRESS.AP (LAPTOP-3FCTAJ5R\surab (54))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
Object Explorer
Connect
LAPTOP-3FCTAJ5R\SQLEXPRESS (SQL Server 14.0.1000
SQLQuery2.sql - LA...CTAJ5R\surab (54))*
BEGIN TRY
EXEC spDateRange '2015-12-15', '';
END TRY
BEGIN CATCH
PRINT 'Error Number: ' + CONVERT(varchar(100), ERROR_NUMBER());
PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;
100 %
Messages
Error Number: 50001
Error Message: The date format is not valid. Please use mm/dd/yy.
100 %
Query executed successfully.
LAPTOP-3FCTAJ5R\SQLEXPRESS ...
LAPTOP-3FCTAJ5R\surab ...
AP
00:00:00
0 rows

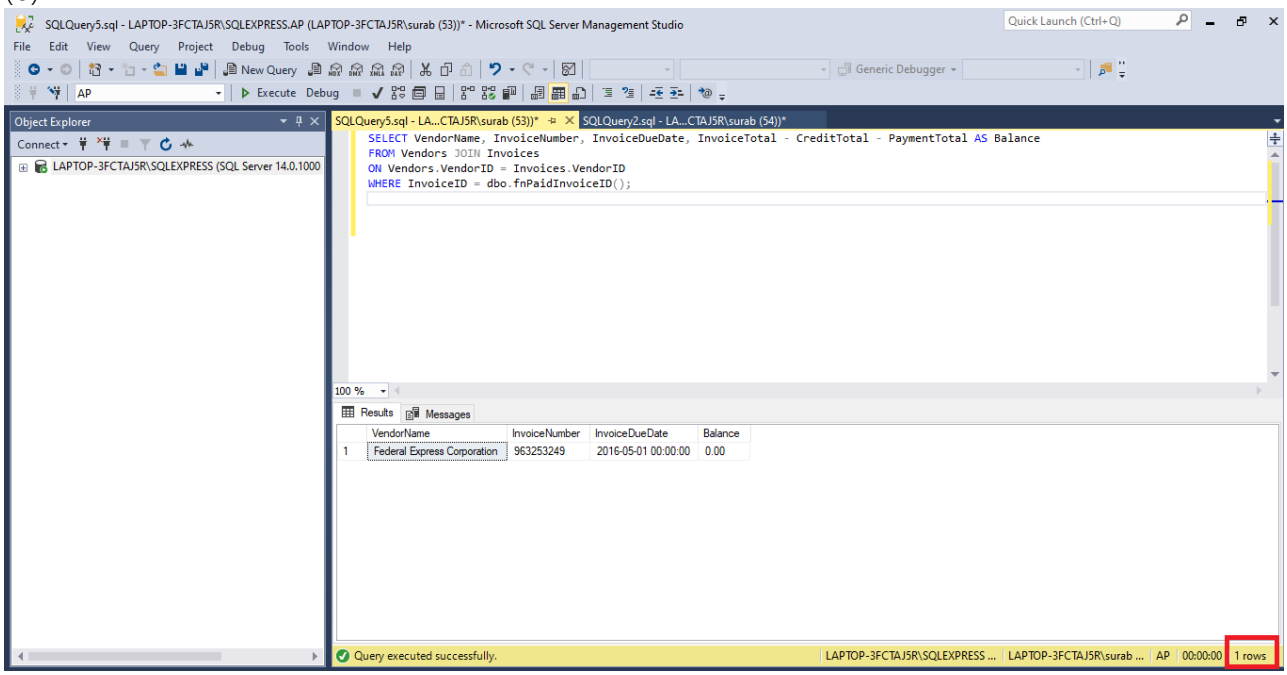
5. Create a scalar-valued function named fnPaidInvoiceID that returns the InvoiceID of the latest invoice with paid balance. Test the function with the following SELECT statement.

```
SELECT VendorName, InvoiceNumber, InvoiceDueDate,
                    InvoiceTotal - CreditTotal - PaymentTotal AS Balance
FROM Vendors JOIN Invoices
        ON Vendors.VendorID = Invoices.VendorID
        WHERE InvoiceID = dbo.fnPaidInvoiceID();
```

(a)

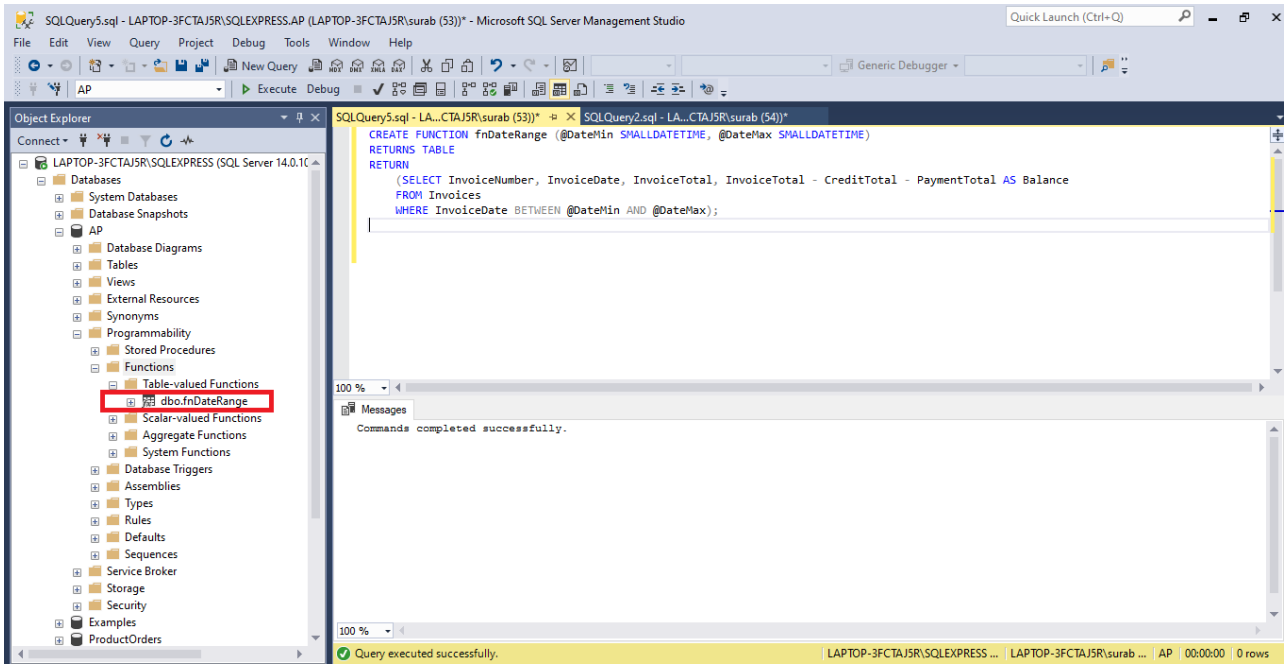



(b)

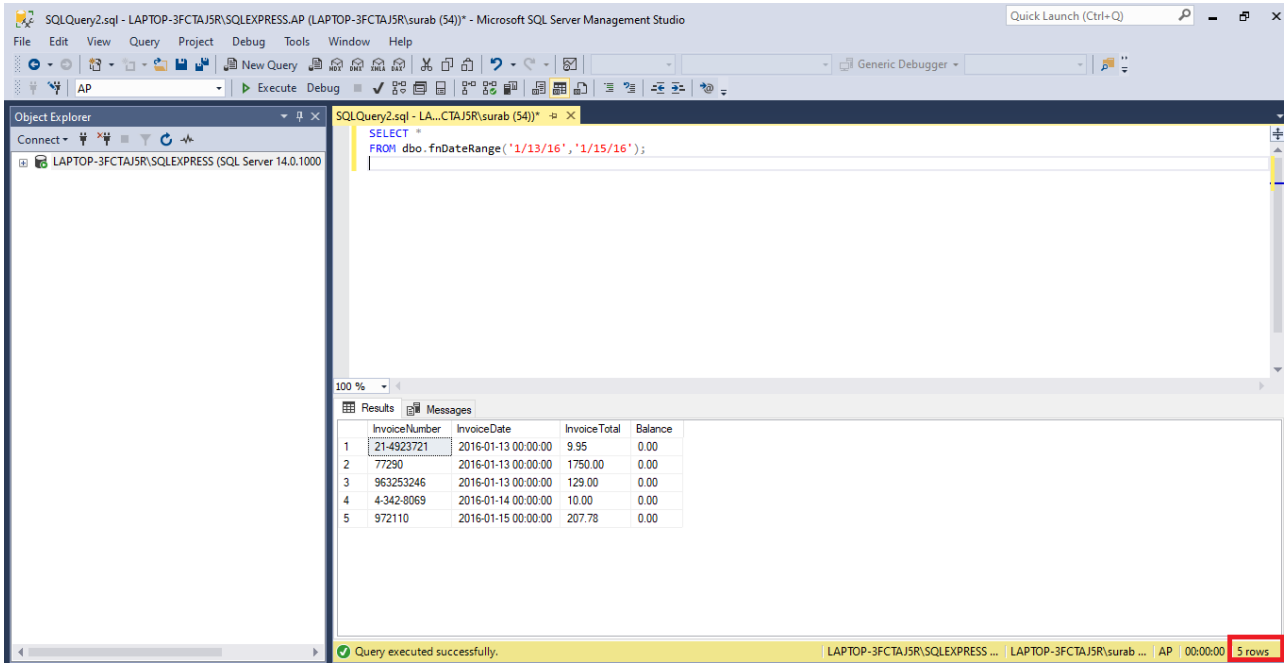


6. Create a table-valued function named fnDateRange, similar to the stored procedure of question 3. The function requires two parameters of data type smalldatetime. Don't validate the parameters. Return a result set that includes the InvoiceNumber, InvoiceDate, InvoiceTotal, and Balance for each invoice for which the InvoiceDate is within the date range. Invoke the function from within a SELECT statement to return those invoices with InvoiceDate between January 13 and January 15, 2016.

(a)




(b)



7. Use the function you created in question 6 in a SELECT statement that returns five columns: VendorCity and the four columns returned by the function.