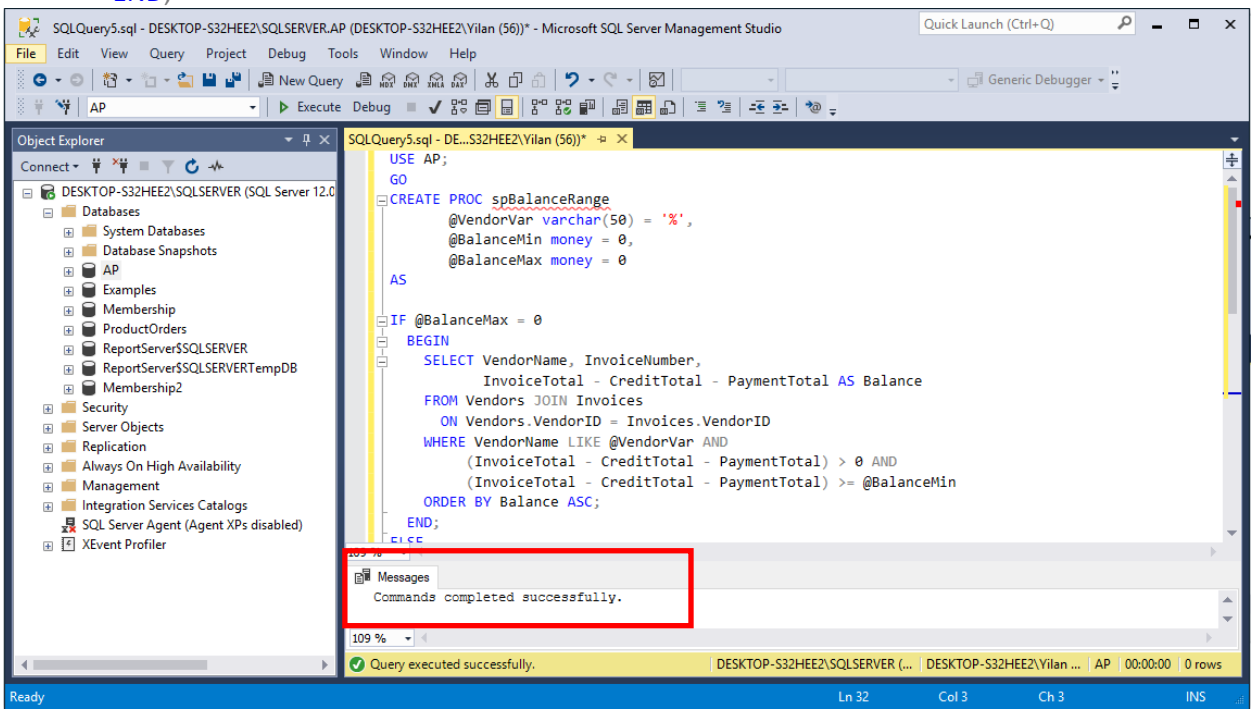
Lab 9: Stored Procedures, Functions Solution

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 smallest 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.

```
USE AP;
GO
CREATE PROC spBalanceRange
    @VendorVar varchar(50) = '%',
    @BalanceMin money = 0,
    @BalanceMax money = 0
AS

IF @BalanceMax = 0
 BEGIN
  SELECT VendorName, InvoiceNumber,
       InvoiceTotal - CreditTotal - PaymentTotal AS Balance
  FROM Vendors JOIN Invoices
   ON Vendors.VendorID = Invoices.VendorID
  WHERE VendorName LIKE @VendorVar AND
     (InvoiceTotal - CreditTotal - PaymentTotal) > 0 AND
     (InvoiceTotal - CreditTotal - PaymentTotal) >= @BalanceMin
  ORDER BY Balance ASC;
 END;
ELSE
 BEGIN
  SELECT VendorName, InvoiceNumber,
       InvoiceTotal - CreditTotal - PaymentTotal AS Balance
  FROM Vendors JOIN Invoices
   ON Vendors.VendorID = Invoices.VendorID
  WHERE VendorName LIKE @VendorVar AND
     (InvoiceTotal - CreditTotal - PaymentTotal) > 0 AND
     (InvoiceTotal - CreditTotal - PaymentTotal)
      BETWEEN @BalanceMin AND @BalanceMax
```

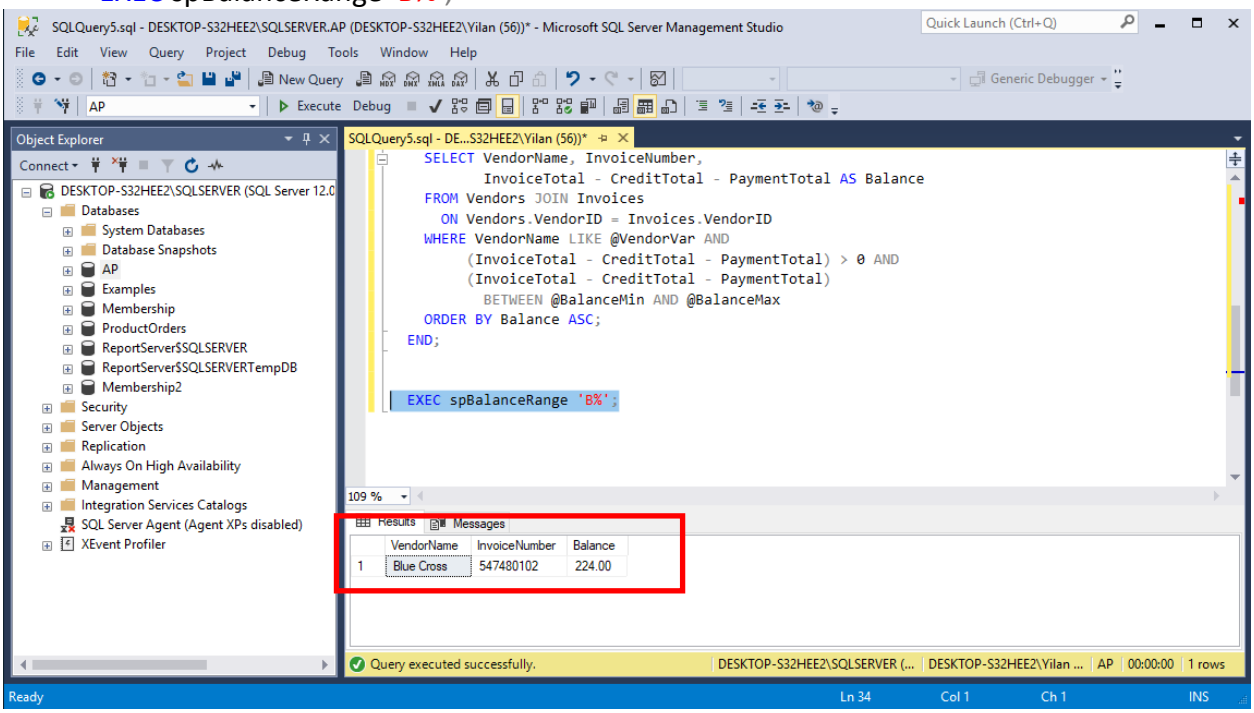
```
    ORDER BY Balance ASC;
END;
```



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

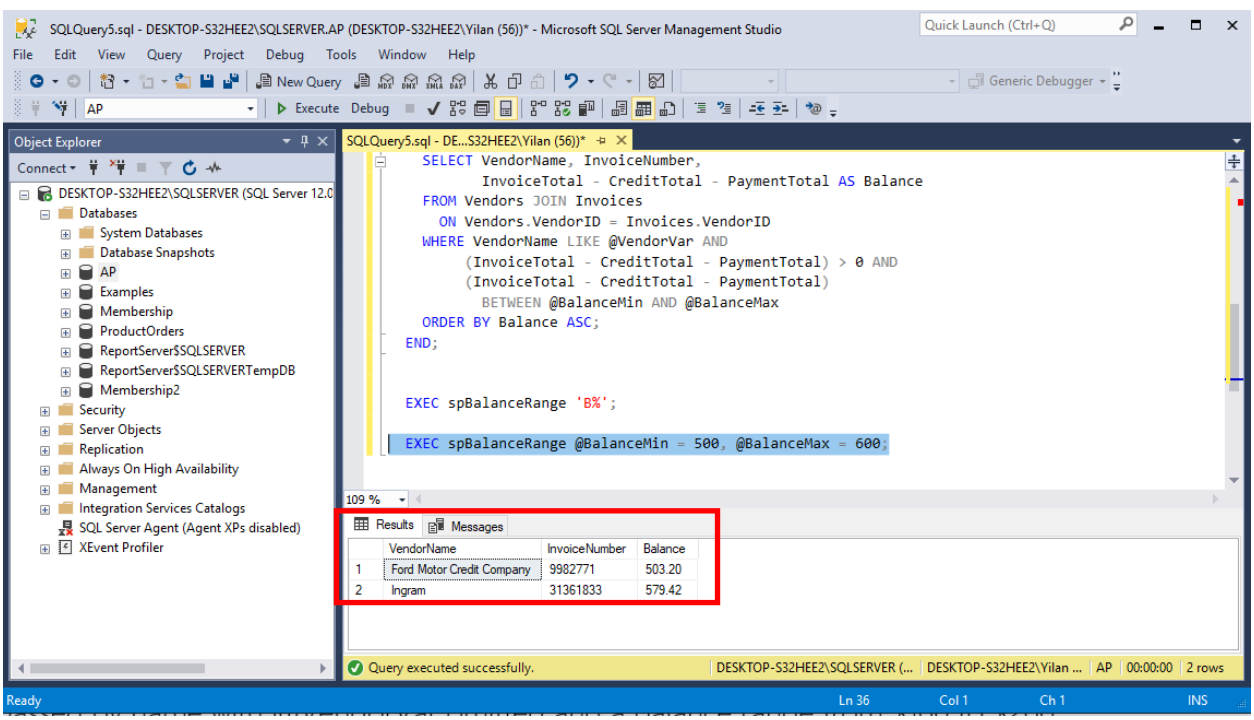
(a) passed by position with @VendorVar = 'B%' and no balance range

```
EXEC spBalanceRange 'B%';
```



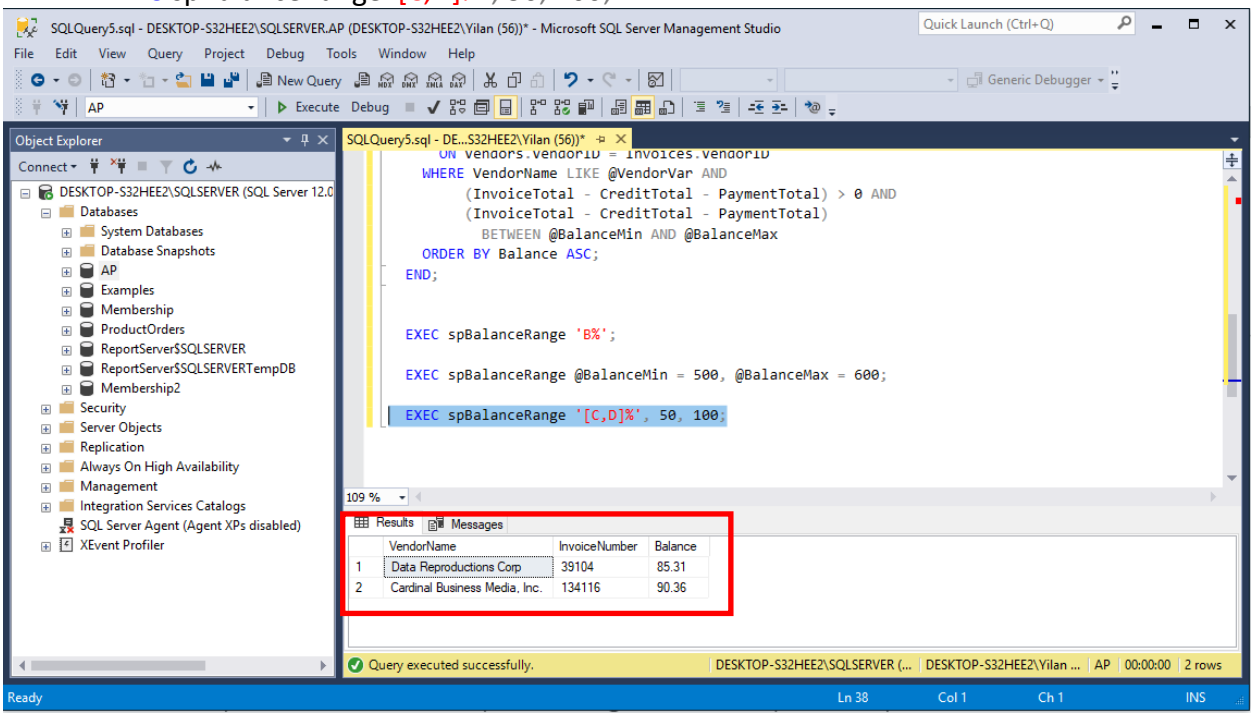
(b) passed by name with @VendorVar omitted and a balance range from $500 to $600

```
EXEC spBalanceRange @BalanceMin = 500, @BalanceMax = 600;
```



(c) passed by position with a balance due from $50 to $100, filtering for vendors whose names begin with C or D

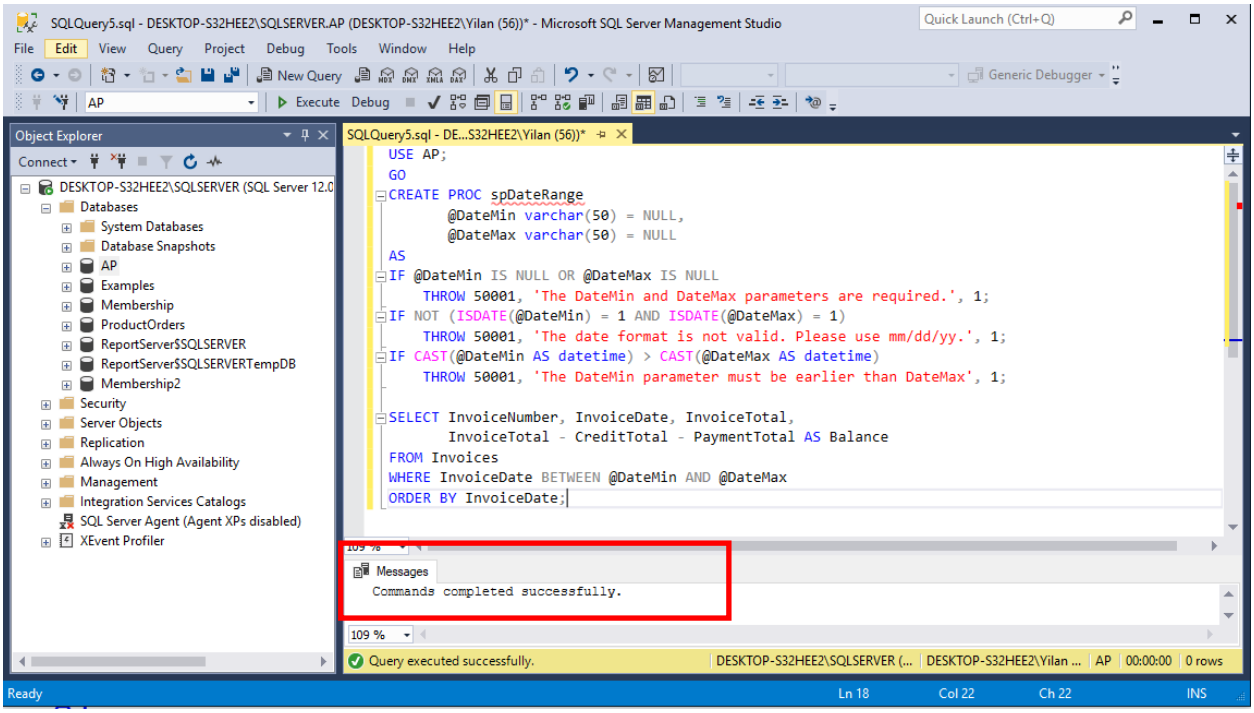
```
EXEC spBalanceRange '[C,D]%', 50, 100;
```



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 first.

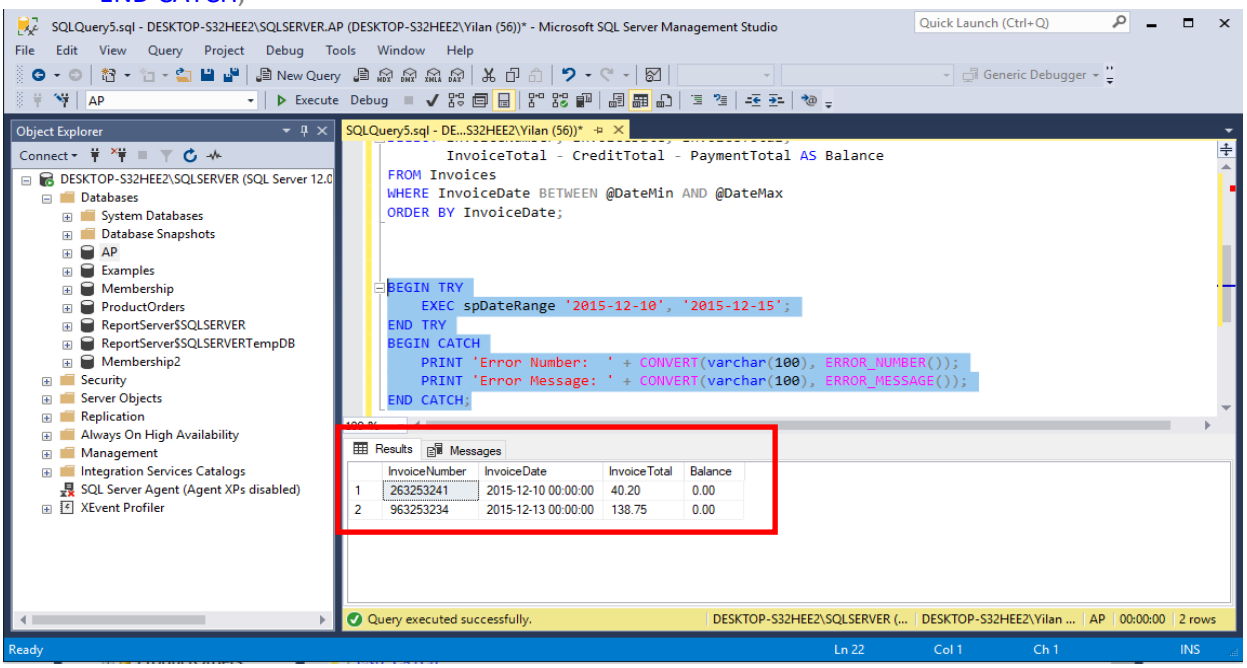
```
USE AP;
GO
CREATE PROC spDateRange
    @DateMin varchar(50) = NULL,
    @DateMax varchar(50) = NULL
AS
IF @DateMin IS NULL OR @DateMax IS NULL
        THROW 50001, 'The DateMin and DateMax parameters are required.', 1;
IF NOT (ISDATE(@DateMin) = 1 AND ISDATE(@DateMax) = 1)
        THROW 50001, 'The date format is not valid. Please use mm/dd/yy.', 1;
IF CAST(@DateMin AS datetime) > CAST(@DateMax AS datetime)
        THROW 50001, 'The DateMin parameter must be earlier than DateMax', 1;

SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
    InvoiceTotal - CreditTotal - PaymentTotal AS Balance
FROM Invoices
WHERE InvoiceDate BETWEEN @DateMin AND @DateMax
ORDER BY InvoiceDate;
```

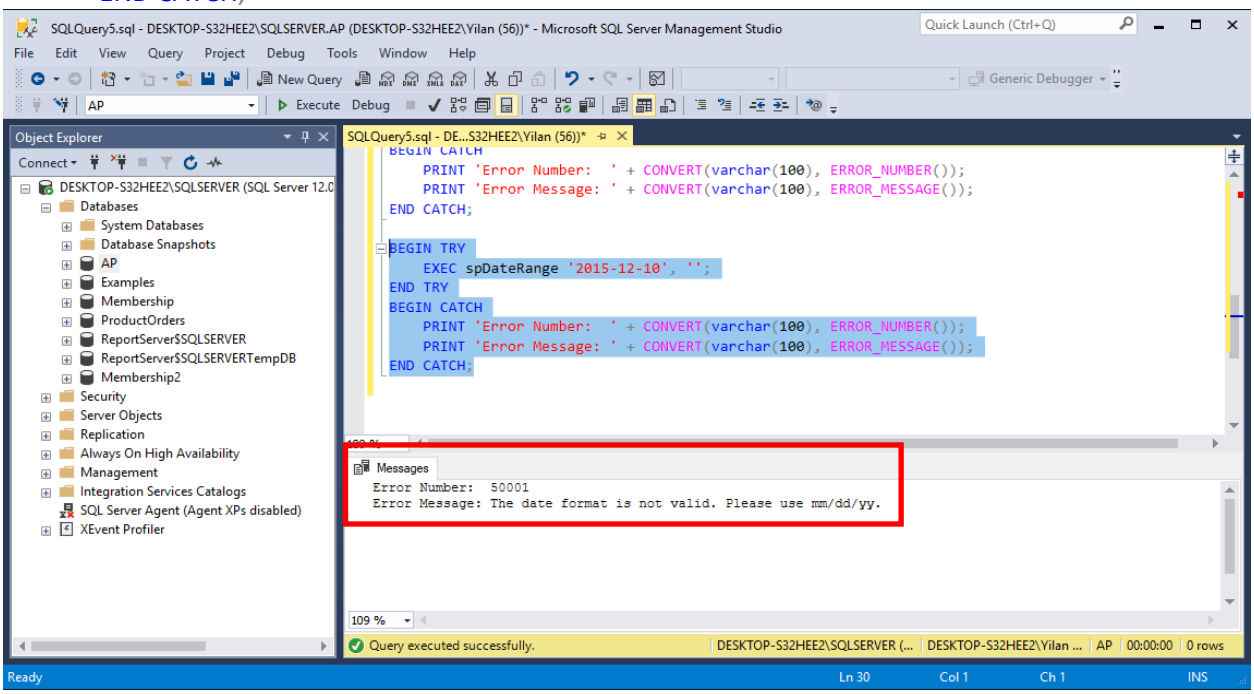

SQLQuery5.sql - DESKTOP-S32HEE2\SQLSERVER.AP (DESKTOP-S32HEE2\Yilan (56))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
Object Explorer
Connect
DESKTOP-S32HEE2\SQLSERVER (SQL Server 12.0
Databases
System Databases
Database Snapshots
AP
Examples
Membership
ProductOrders
ReportServer$SQLSERVER
ReportServer$SQLSERVERTempDB
Membership2
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
SQLQuery5.sql - DE...S32HEE2\Yilan (56))*
USE AP;
GO
CREATE PROC spDateRange
@DateMin varchar(50) = NULL,
@DateMax varchar(50) = NULL
AS
IF @DateMin IS NULL OR @DateMax IS NULL
THROW 50001, 'The DateMin and DateMax parameters are required.', 1;
IF NOT (ISDATE(@DateMin) = 1 AND ISDATE(@DateMax) = 1)
THROW 50001, 'The date format is not valid. Please use mm/dd/yy.', 1;
IF CAST(@DateMin AS datetime) > CAST(@DateMax AS datetime)
THROW 50001, 'The DateMin parameter must be earlier than DateMax', 1;
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
InvoiceTotal - CreditTotal - PaymentTotal AS Balance
FROM Invoices
WHERE InvoiceDate BETWEEN @DateMin AND @DateMax
ORDER BY InvoiceDate;
Messages
Commands completed successfully.
109 %
Query executed successfully.
DESKTOP-S32HEE2\SQLSERVER (...
DESKTOP-S32HEE2\Yilan ...
AP
00:00:00
0 rows
Ready
Ln 18
Col 22
Ch 22
INS

4. Code (1) a call to the stored procedure created in question 3 that returns invoices with an InvoiceDate between December 10 and December 15, 2015, (2)a call to the stored procedure again that returns invoices with an @DateMin is December 10. These calls should also catch any errors that are raised by the procedure and print the error number and description.

```
BEGIN TRY
        EXEC spDateRange '2015-12-10', '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;
```

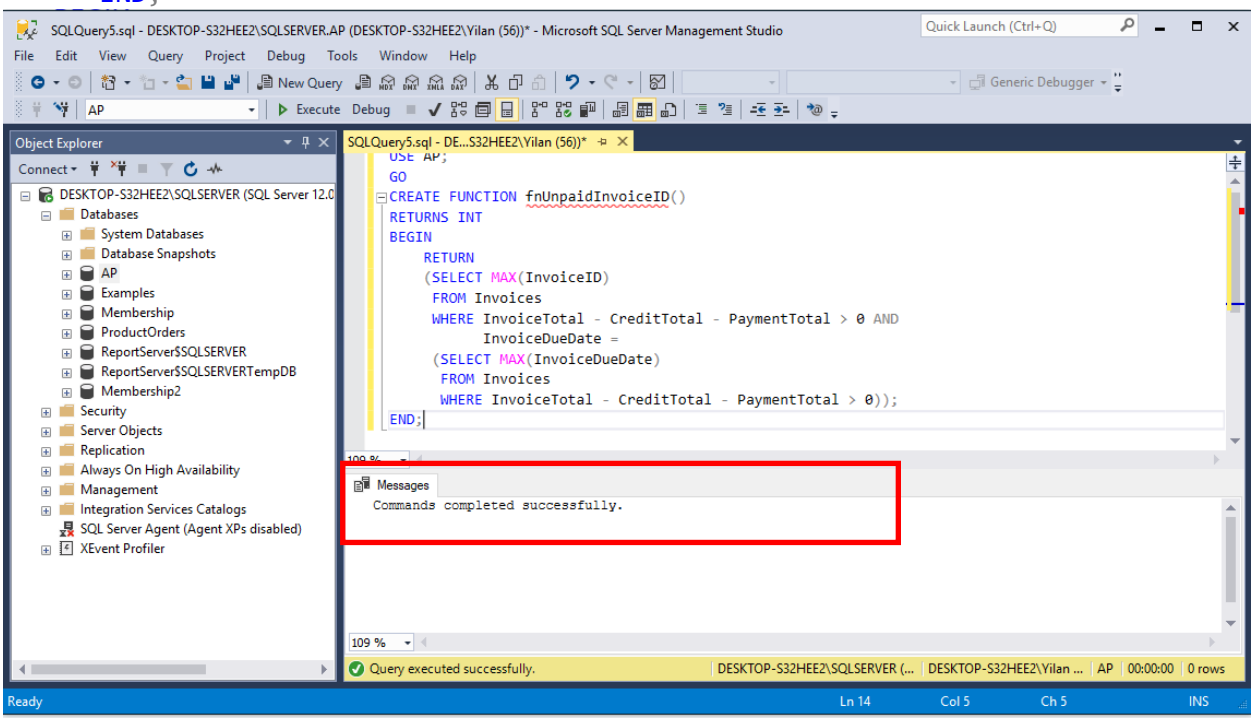


```
BEGIN TRY
        EXEC spDateRange '2015-12-10', '';
END TRY
BEGIN CATCH
        PRINT 'Error Number:  ' + CONVERT(varchar(100), ERROR_NUMBER());
        PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;
```

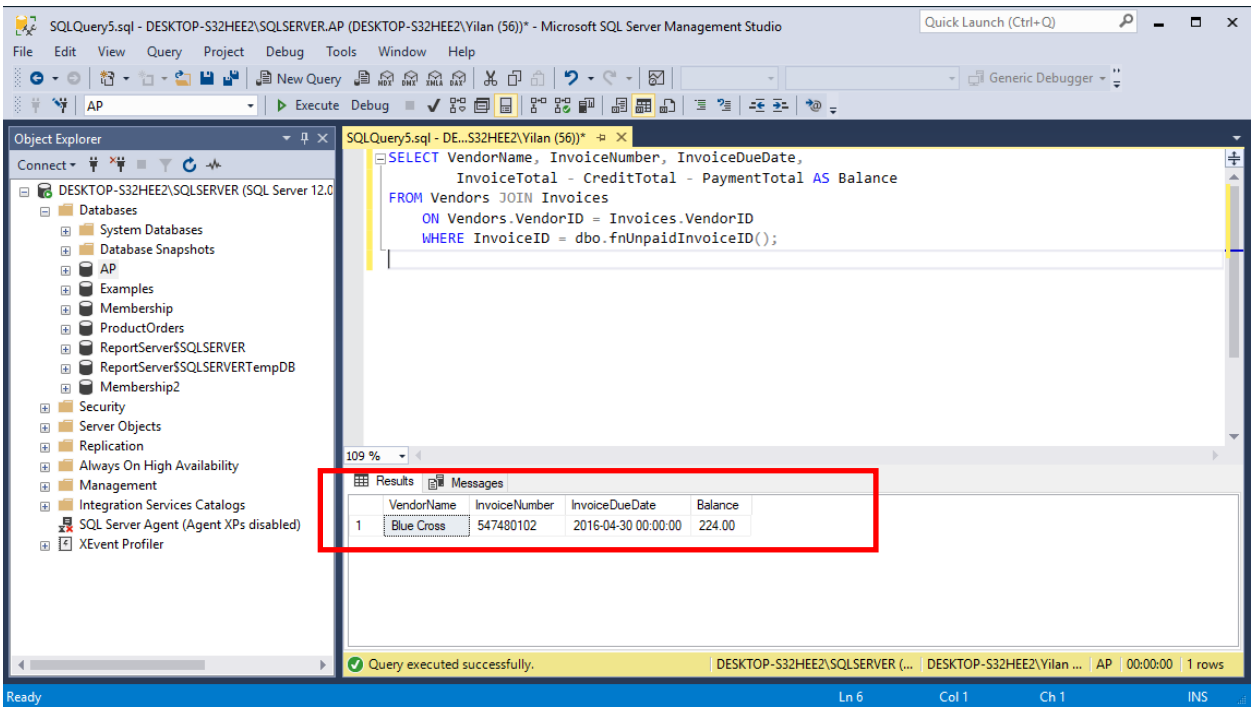


5. Create a scalar-valued function named fnUnpaidInvoiceID that returns the InvoiceID of the latest invoice with an unpaid balance.

```
USE AP;
GO
CREATE FUNCTION fnUnpaidInvoiceID()
RETURNS INT
BEGIN
    RETURN
    (SELECT MAX(InvoiceID)
     FROM Invoices
     WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0 AND
           InvoiceDueDate =
     (SELECT MAX(InvoiceDueDate)
      FROM Invoices
      WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0));
END;
```



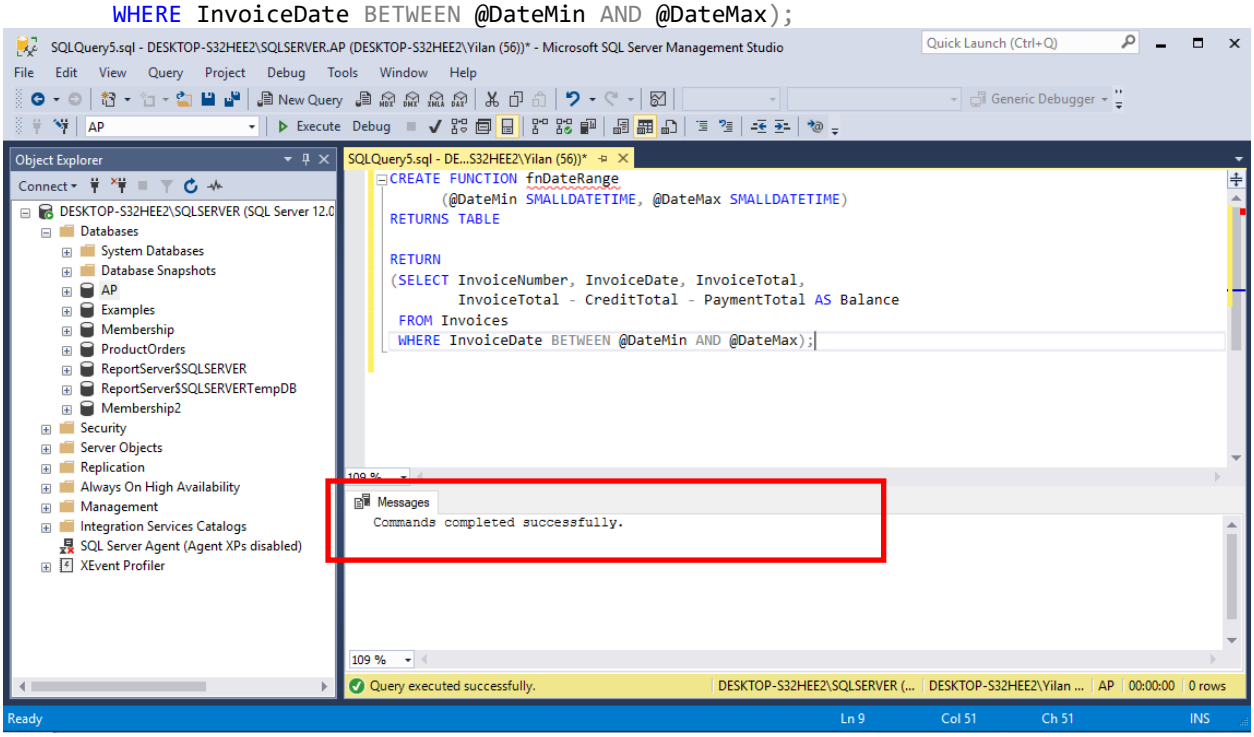
Test results:



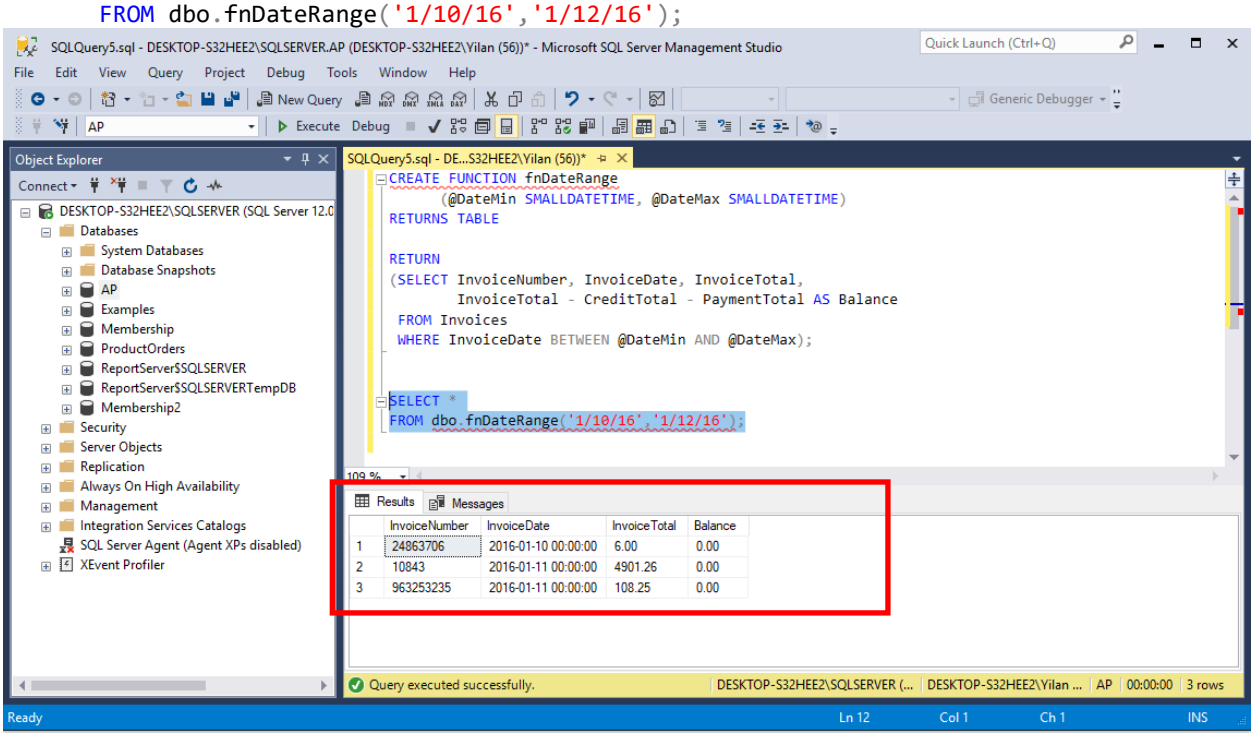
6. Create a table-valued function named fnDateRange, similar to the stored procedure of question 3. The function requires two parameters of data type smalldatetime. Don't validate the parameters. Return a result set that includes the InvoiceNumber, InvoiceDate, InvoiceTotal, and Balance for each invoice 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 10 and January 12, 2016.

```
CREATE FUNCTION fnDateRange
      (@DateMin SMALLDATETIME, @DateMax SMALLDATETIME)
RETURNS TABLE

RETURN
(SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
        InvoiceTotal - CreditTotal - PaymentTotal AS Balance
 FROM Invoices
 WHERE InvoiceDate BETWEEN @DateMin AND @DateMax);
```



```
SELECT *
FROM dbo.fnDateRange('1/10/16','1/12/16');
```



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

```
SELECT VendorName, FunctionTable.*
FROM Vendors JOIN Invoices
  ON Vendors.VendorID = Invoices.VendorID
JOIN dbo.fnDateRange('1/10/16','1/12/16') AS FunctionTable
  ON Invoices.InvoiceNumber = FunctionTable.InvoiceNumber;
```