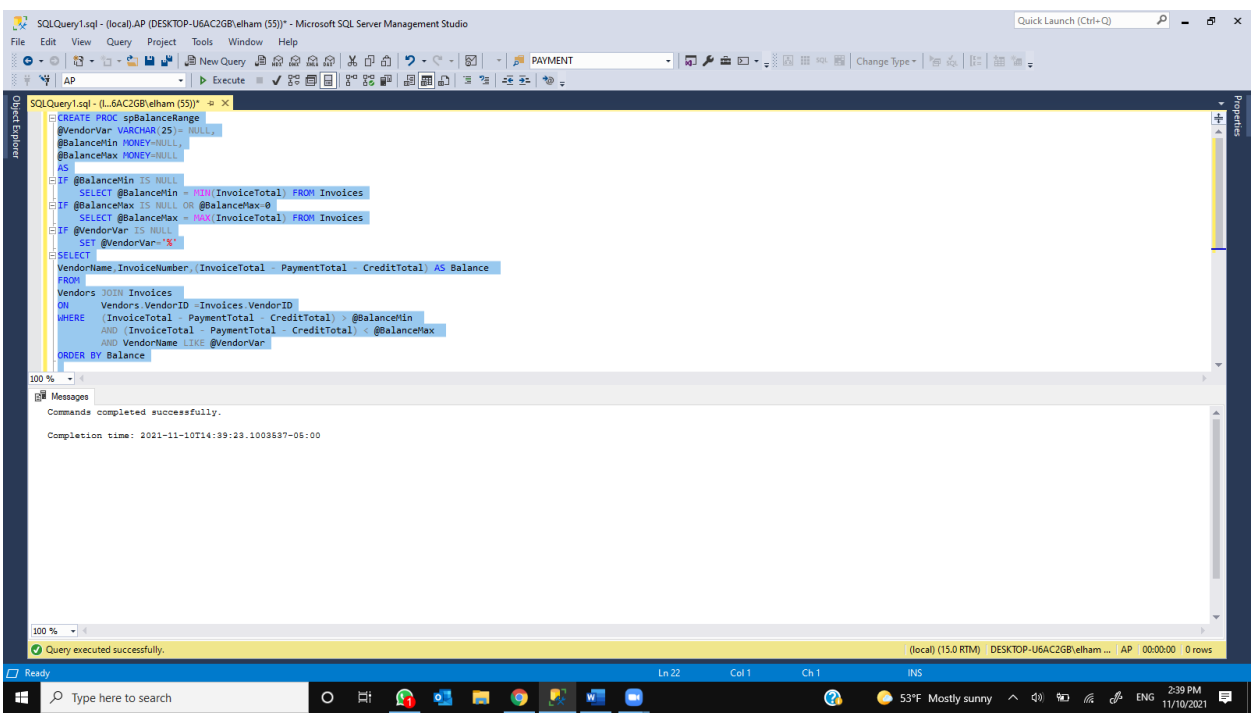
LAB 9

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

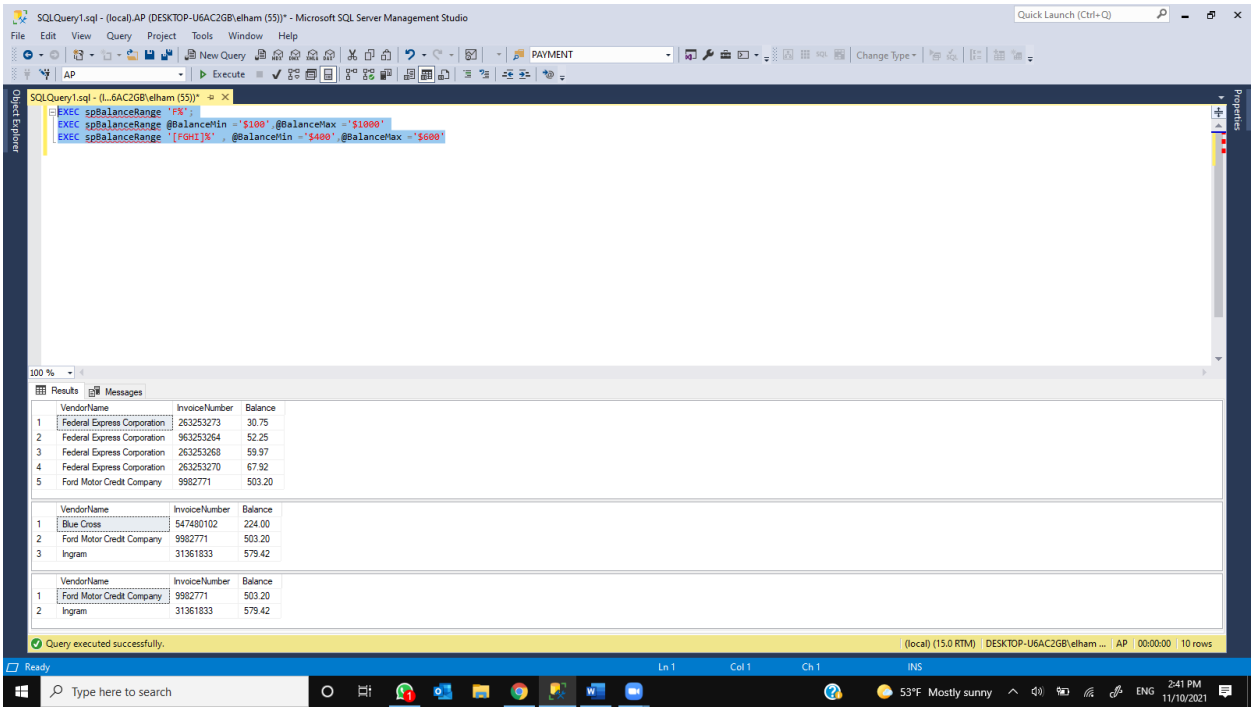
```
CREATE PROC spBalanceRange
@VendorVar VARCHAR(25)= NULL,
@BalanceMin MONEY=NULL,
@BalanceMax MONEY=NULL
AS
IF @BalanceMin IS NULL
       SELECT @BalanceMin = MIN(InvoiceTotal) FROM Invoices
IF @BalanceMax IS NULL OR @BalanceMax=0
       SELECT @BalanceMax = MAX(InvoiceTotal) FROM Invoices
IF @VendorVar IS NULL
       SET @VendorVar='%'
SELECT
VendorName,InvoiceNumber,(InvoiceTotal - PaymentTotal - CreditTotal) AS Balance
FROM
Vendors JOIN Invoices
ON            Vendors.VendorID =Invoices.VendorID
WHERE  (InvoiceTotal - PaymentTotal - CreditTotal) > @BalanceMin
              AND (InvoiceTotal - PaymentTotal - CreditTotal) < @BalanceMax
              AND VendorName LIKE @VendorVar
ORDER BY Balance
```



Remark: This query creates a stored procedure that shows VendorName, InvoiceNumber, and Balance for each invoice with a balance due .

2. Code three calls to the procedure created in question 1: a. passed by position with @VendorVar = ‘F%’ and no balance range b. passed by name with @VendorVar omitted and a balance range from $100 to $1000 c. passed by position with a balance due from $400 to $600, filtering for vendors whose names begin with F or G or H or I.

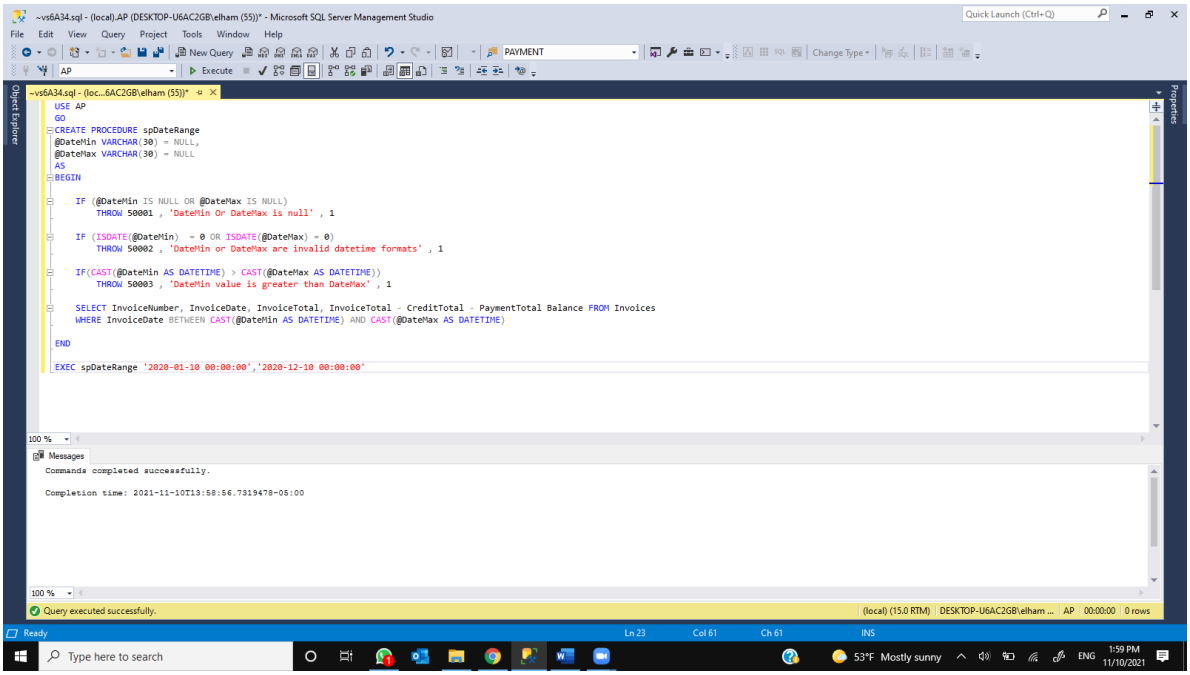
```
EXEC spBalanceRange 'F%';
EXEC spBalanceRange @BalanceMin ='$100',@BalanceMax ='$1000'
EXEC spBalanceRange '[FGHI]%' , @BalanceMin ='$400',@BalanceMax ='$600'
```



Remark: Here we can find the vendors with a balance between 400 and 600 and who name start with certain letters

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 latest invoice first.

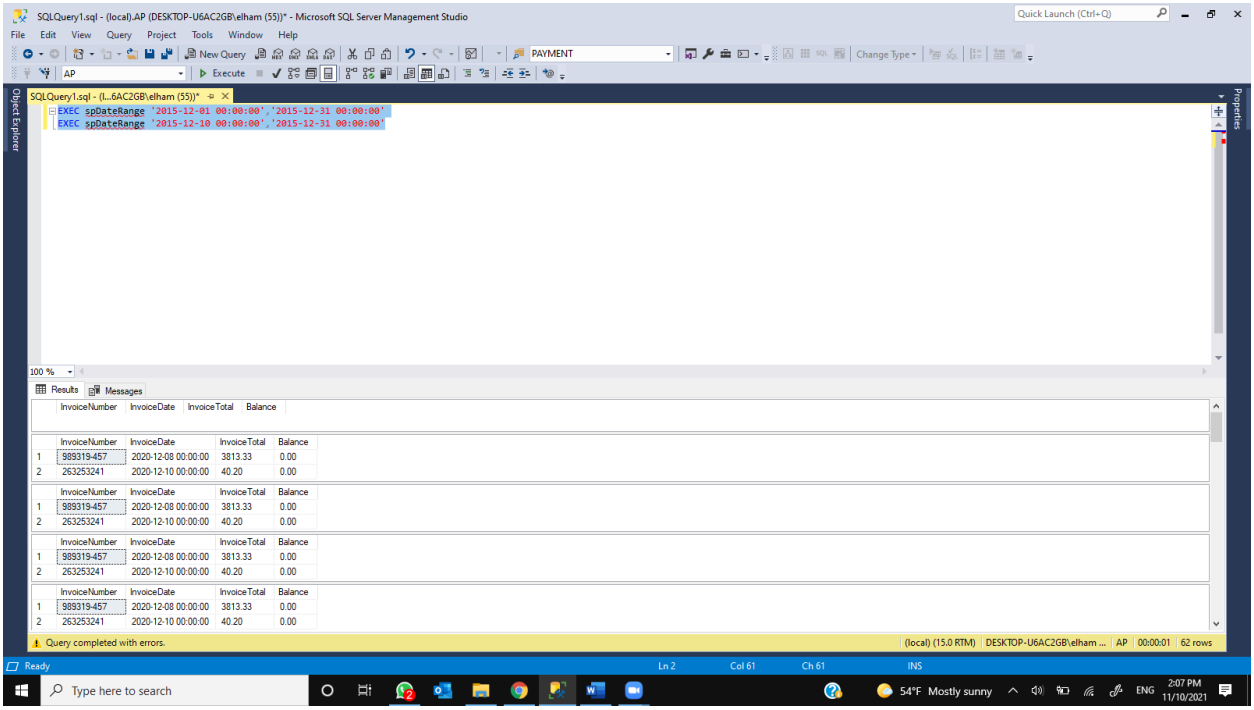
```
USE AP
GO
CREATE PROCEDURE spDateRange
@DateMin VARCHAR(30) = NULL,
@DateMax VARCHAR(30) = NULL
AS
BEGIN

	IF (@DateMin IS NULL OR @DateMax IS NULL)
		THROW 50001 , 'DateMin Or DateMax is null' , 1

	IF (ISDATE(@DateMin)  = 0 OR ISDATE(@DateMax) = 0)
		THROW 50002 , 'DateMin or DateMax are invalid datetime formats' , 1

	IF(CAST(@DateMin AS DATETIME) > CAST(@DateMax AS DATETIME))
		THROW 50003 , 'DateMin value is greater than DateMax' , 1

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

END

EXEC spDateRange '2020-01-10 00:00:00','2020-12-10 00:00:00'
```



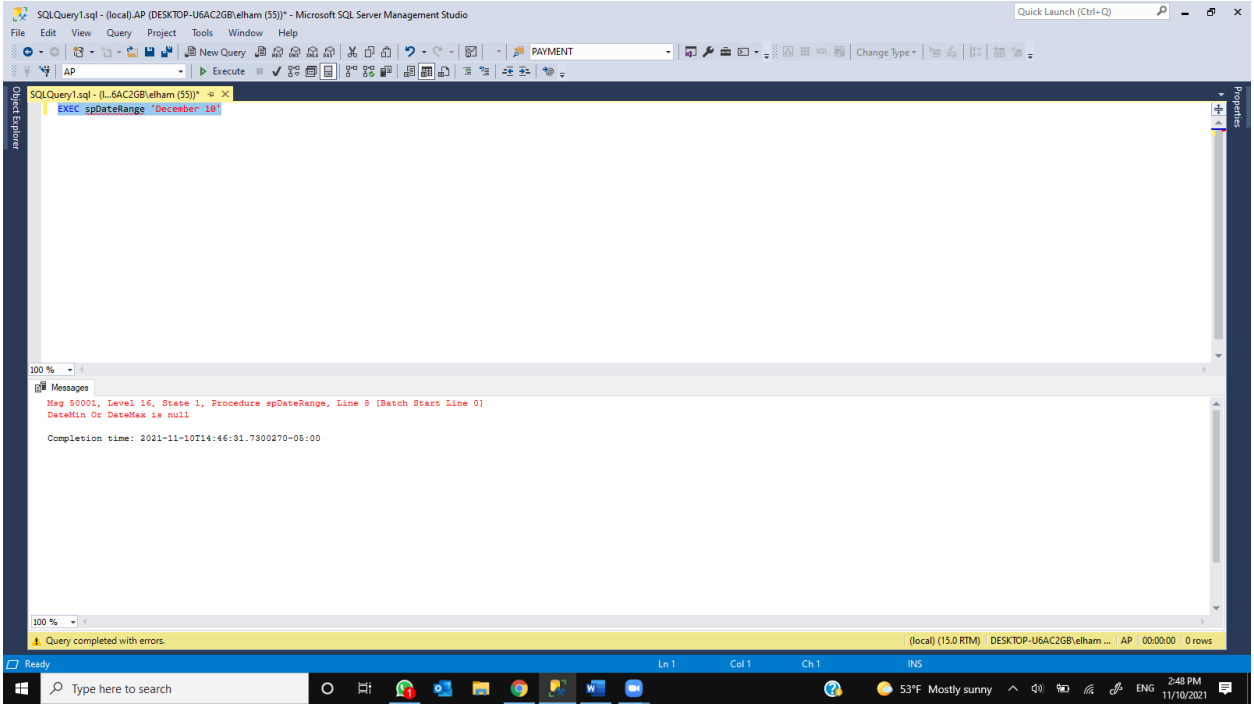
Remark: Here we are Creating a stored procedure that returns the the InvoiceNumber, InvoiceDate, InvoiceTotal, and Balance for each invoice for which the invoice date are within the Mindate and MaxDate. This will also raise an error that specifies the problem if called with no parameters or null values and If called with non-null values, validate the parameters.

4. Code (1) A call to the stored procedure created in question 3 that returns invoices with an InvoiceDate between December 1 and December 31, 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.

```
EXEC spDateRange '2015-12-01 00:00:00','2015-12-31 00:00:00'
EXEC spDateRange '2015-12-10 00:00:00','2015-12-31 00:00:00'
```



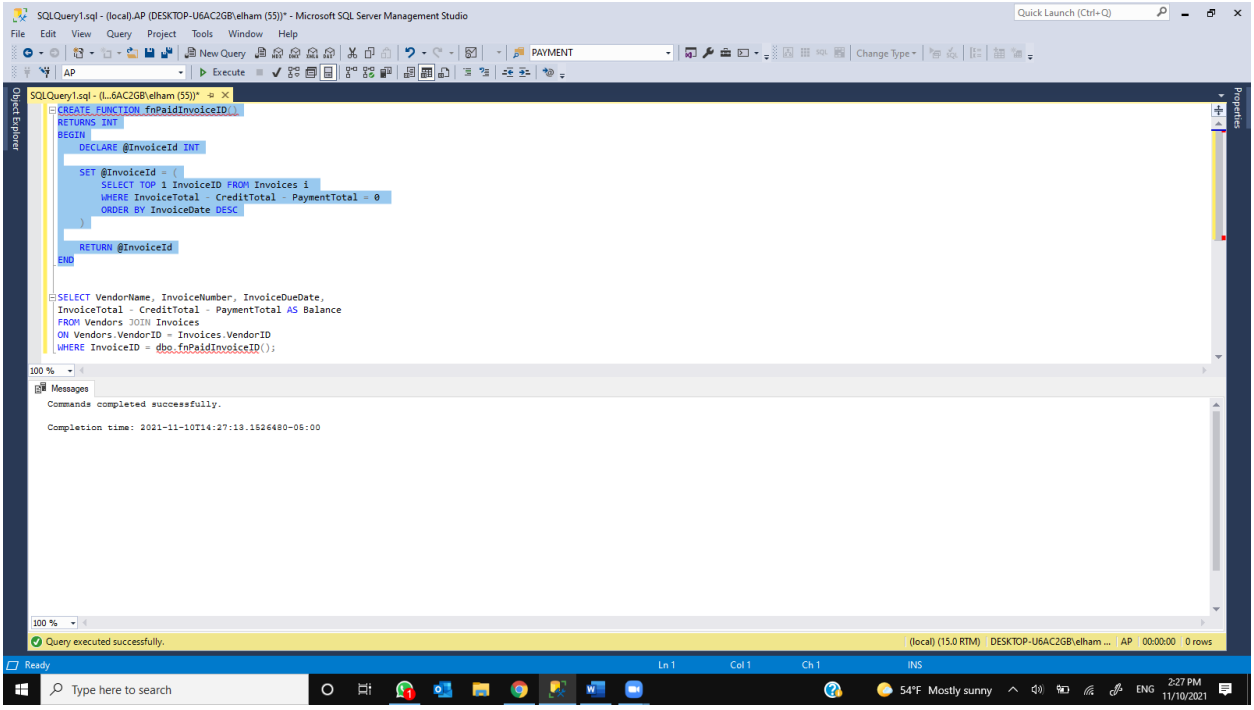
```
EXEC spDateRange 'December 10'
```



Remark: the stored procedure built in question 3 that returns bills with an InvoiceDate of December 1 through December 31, 2015, The stored procedure is called again, and invoices with a @DateMin of December 10 are returned.

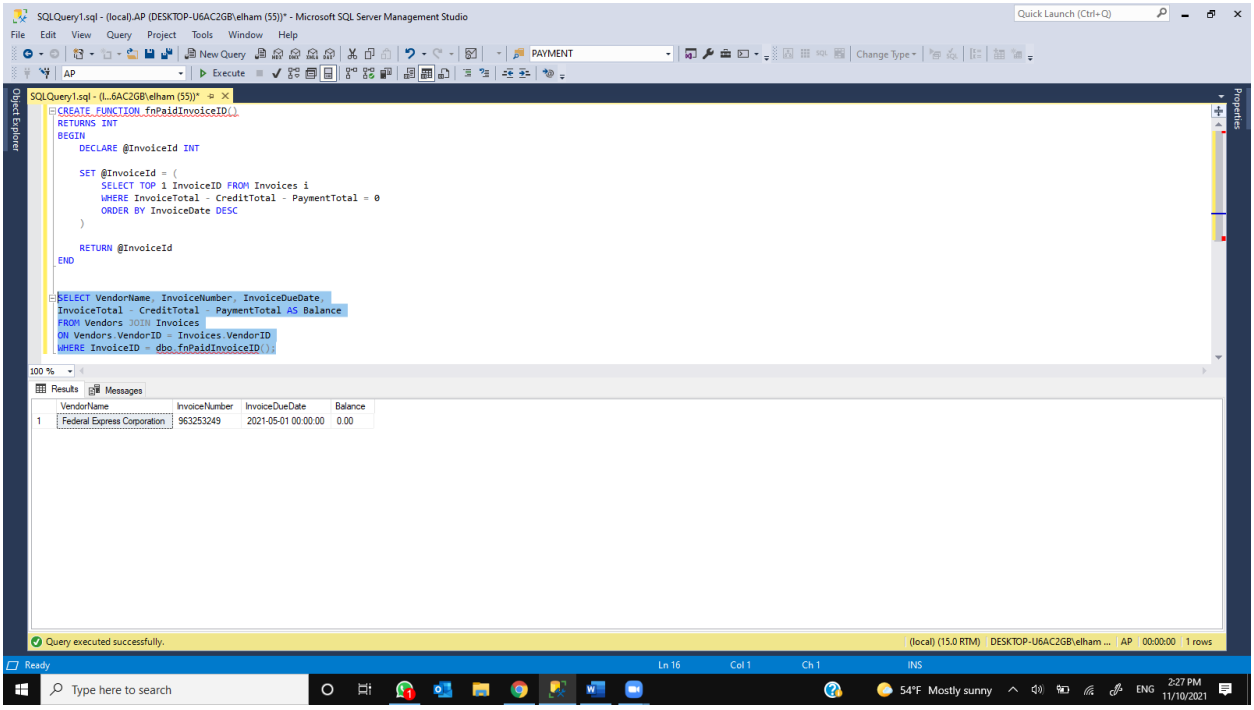
5. Create a scalar-valued function named fnPaidInvoiceID that returns the InvoiceID of the latest invoice with an paid balance. Test the function in the following SELECT statement: SELECT VendorName, InvoiceNumber, InvoiceDueDate, InvoiceTotal - CreditTotal - PaymentTotal AS Balance FROM Vendors JOIN Invoices ON Vendors.VendorID = Invoices.VendorID WHERE InvoiceID = dbo.fnPaidInvoiceID();

```
CREATE FUNCTION fnPaidInvoiceID()
RETURNS INT
BEGIN
	DECLARE @InvoiceId INT

	SET @InvoiceId = (
		SELECT TOP 1 InvoiceID FROM Invoices i
		WHERE InvoiceTotal - CreditTotal - PaymentTotal = 0
		ORDER BY InvoiceDate DESC
	)

	RETURN @InvoiceId
END
```



Remark: Here we create a scalar-valued function called fnPaidInvoiceID, which returns the InvoiceID of the most recent invoice with a paid balance.

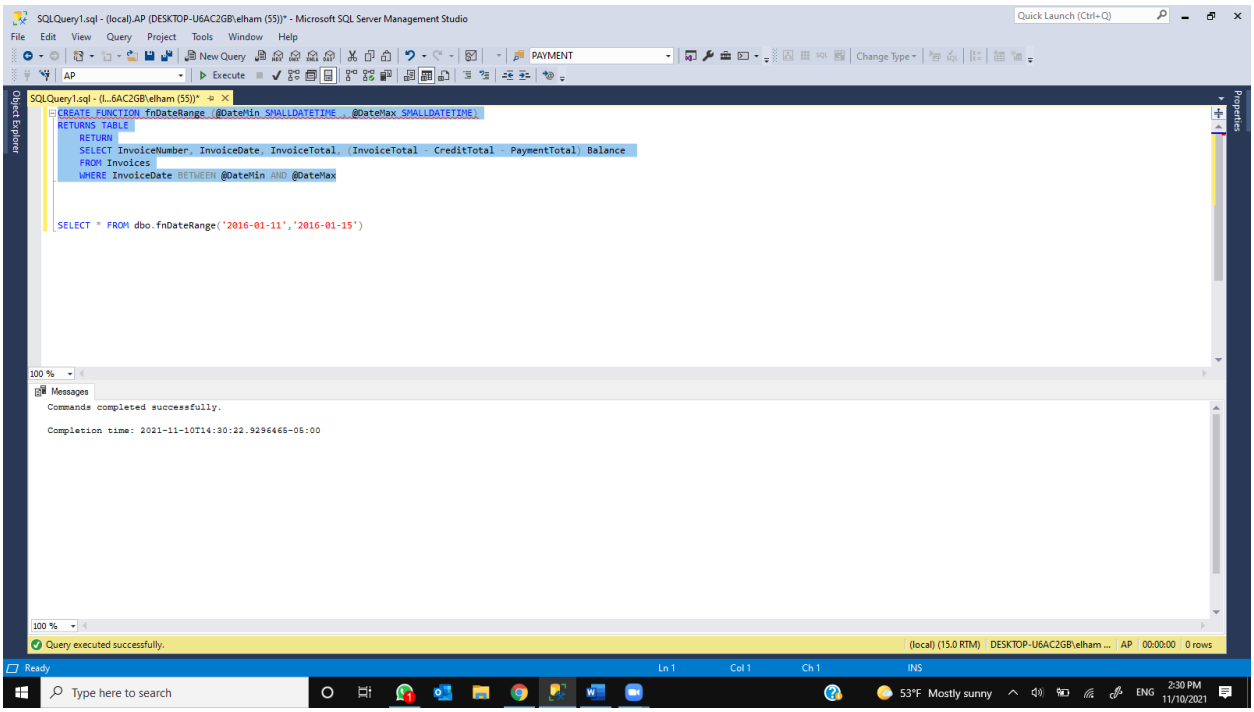
```
SELECT VendorName, InvoiceNumber, InvoiceDueDate,
InvoiceTotal - CreditTotal - PaymentTotal AS Balance
FROM Vendors JOIN Invoices
ON Vendors.VendorID = Invoices.VendorID
WHERE InvoiceID = dbo.fnPaidInvoiceID();
```



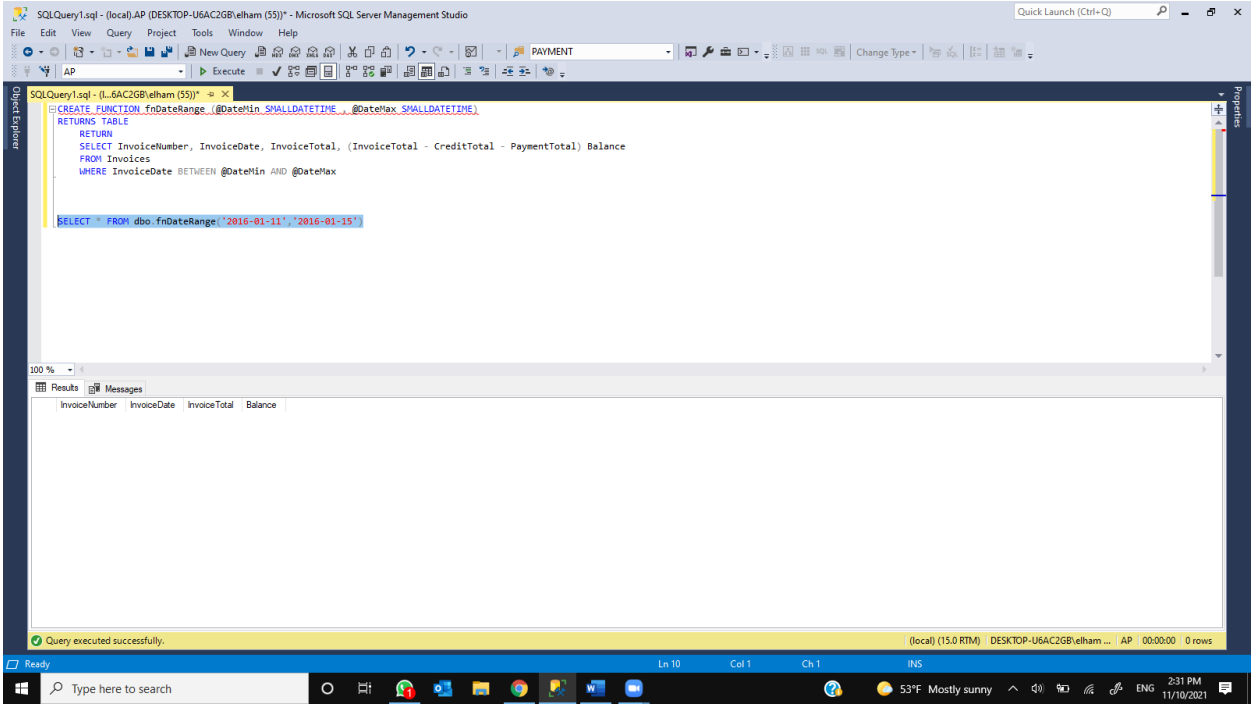
Remark: In the SELECT query above, we'll see how well the function works.

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 11 and January 15, 2016.

```
CREATE FUNCTION fnDateRange (@DateMin SMALLDATETIME , @DateMax SMALLDATETIME)
RETURNS TABLE
       RETURN
       SELECT InvoiceNumber, InvoiceDate, InvoiceTotal, (InvoiceTotal - CreditTotal -
PaymentTotal) Balance
       FROM Invoices
       WHERE InvoiceDate BETWEEN @DateMin AND @DateMax
```



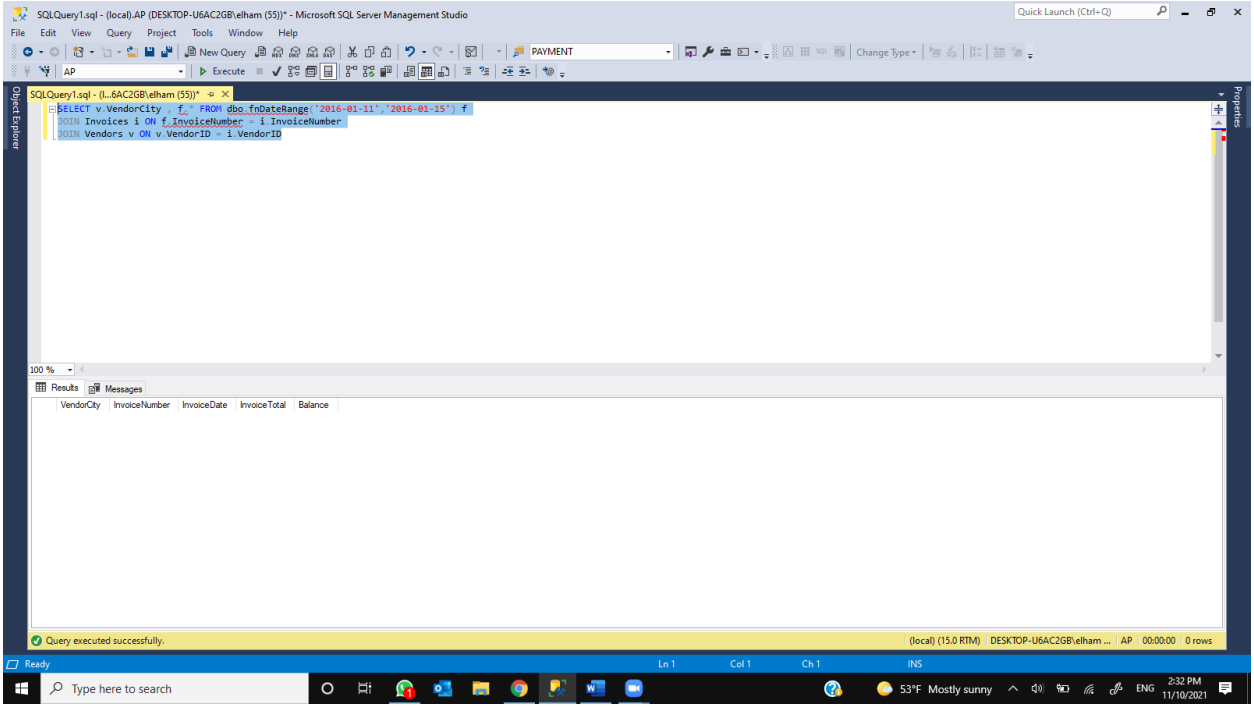
```sql
SELECT * FROM dbo.fnDateRange('2016-01-11','2016-01-15')
```



Remark: This query create a table with fields such as InvoceNumber. Just like our stored procedure, we have InvoiceDate, InvoiceTotal, and Balance. This function also returns data from a defined date range.

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.

```
SELECT v.VendorCity , f.* FROM dbo.fnDateRange('2016-01-11','2016-01-15') f
JOIN Invoices i ON f.InvoiceNumber = i.InvoiceNumber
JOIN Vendors v ON v.VendorID = i.VendorID
```



Remark: We used a select statement to show a vendor city column. Because there is no direct connection between the Vendors and the fnDateRange columns, we must use the invoice table as a connector. The Invoices table's field InvoiceNumber was joined to the FnDaterange table.