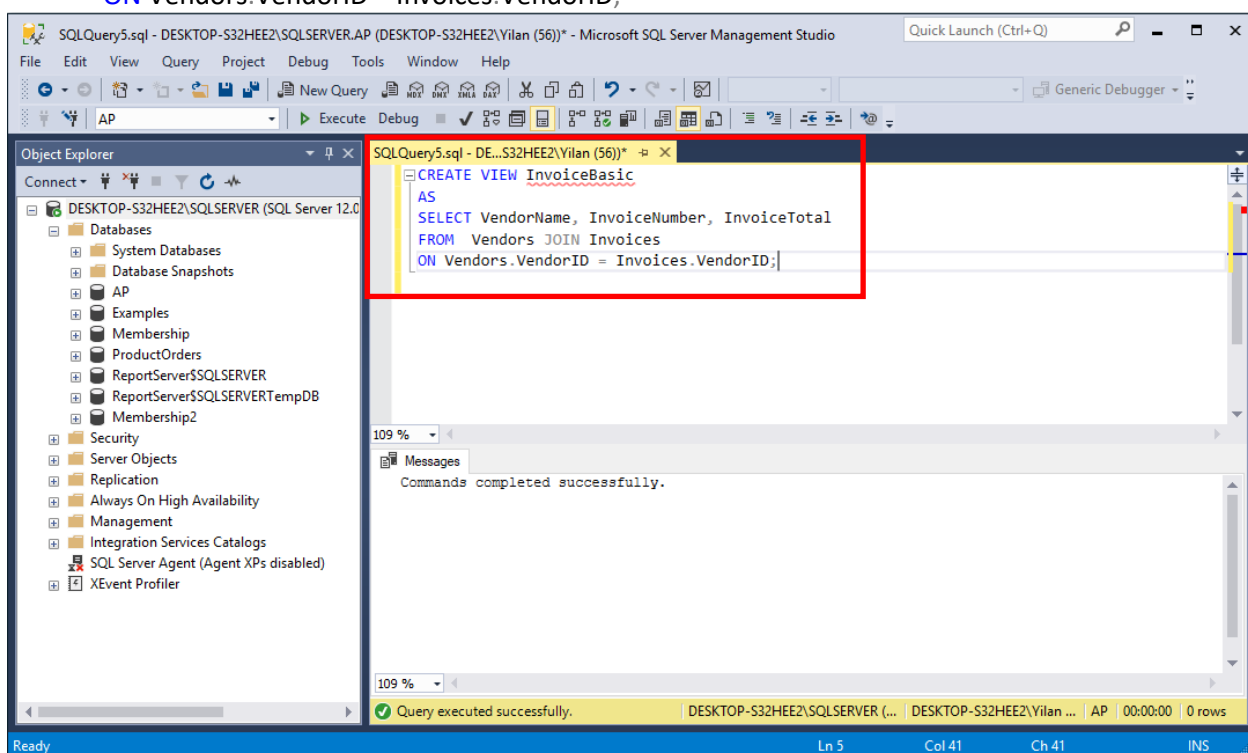




Lab 8: Views, Scripts Solution

1. Write a view named InvoiceBasic that returns three columns: VendorName, InvoiceNumber, and InvoiceTotal. Then, write a SELECT statement that returns all the columns in the view, sorted by InvoiceTotal from smallest to largest, where the first letter of the vendor name is X, Y or Z.

```
CREATE VIEW InvoiceBasic
AS
SELECT VendorName, InvoiceNumber, InvoiceTotal
FROM Vendors JOIN Invoices
ON Vendors.VendorID = Invoices.VendorID;
```



```
SELECT *  
FROM InvoiceBasic  
WHERE VendorName LIKE '[X-Z]%'  
ORDER BY InvoiceTotal ASC;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor displays the following SQL code:

```
CREATE VIEW InvoiceBasic  
AS  
SELECT VendorName, InvoiceNumber, InvoiceTotal  
FROM Vendors JOIN Invoices  
ON Vendors.VendorID = Invoices.VendorID;  
  
SELECT *  
FROM InvoiceBasic  
WHERE VendorName LIKE '[X-Z]%'  
ORDER BY InvoiceTotal ASC;
```

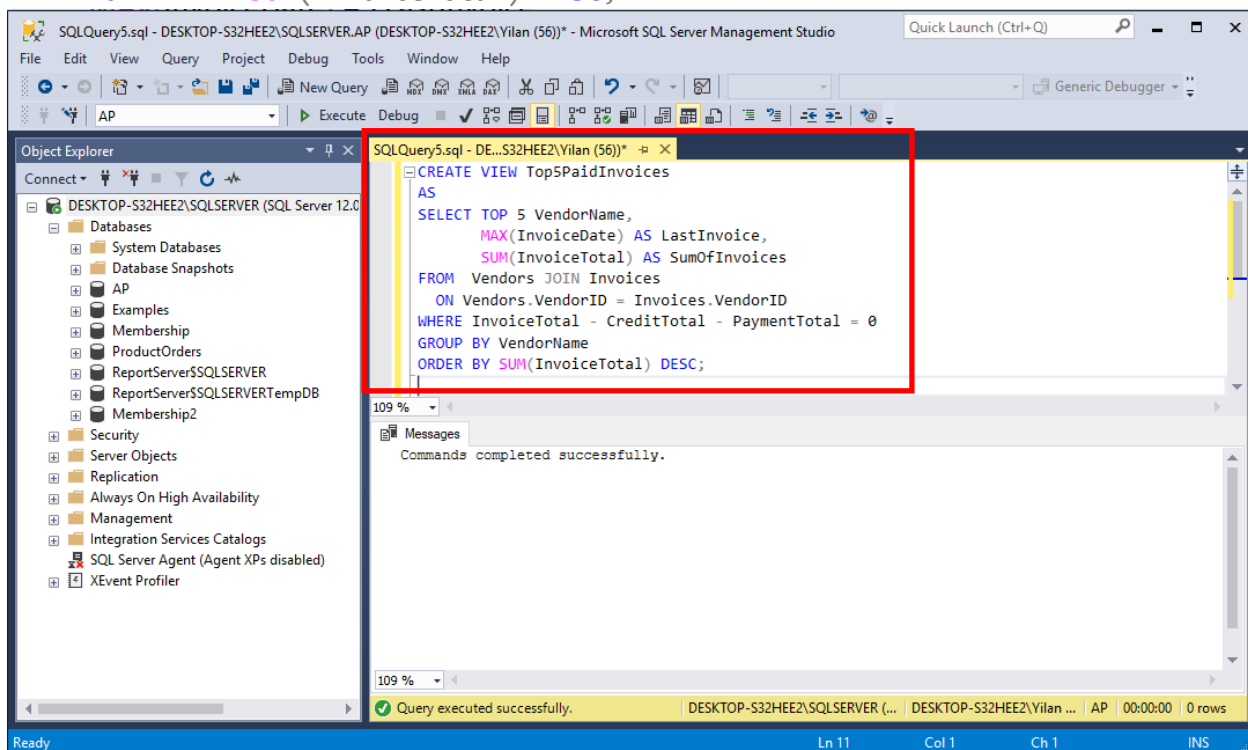
The query is highlighted with a red box. The results grid shows the following data:

	VendorName	InvoiceNumber	InvoiceTotal
1	Zylka Design	97/553B	313.55
2	Zylka Design	97/465	565.15
3	Zylka Design	97/488	601.95
4	Zylka Design	97/503	639.77
5	Zylka Design	97/553	904.14
6	Zylka Design	97/486	953.10
7	Zylka Design	97/222	1000.46
8	Zylka Design	97/522	1962.13
9	Yeamed, Inc	10843	4901.26

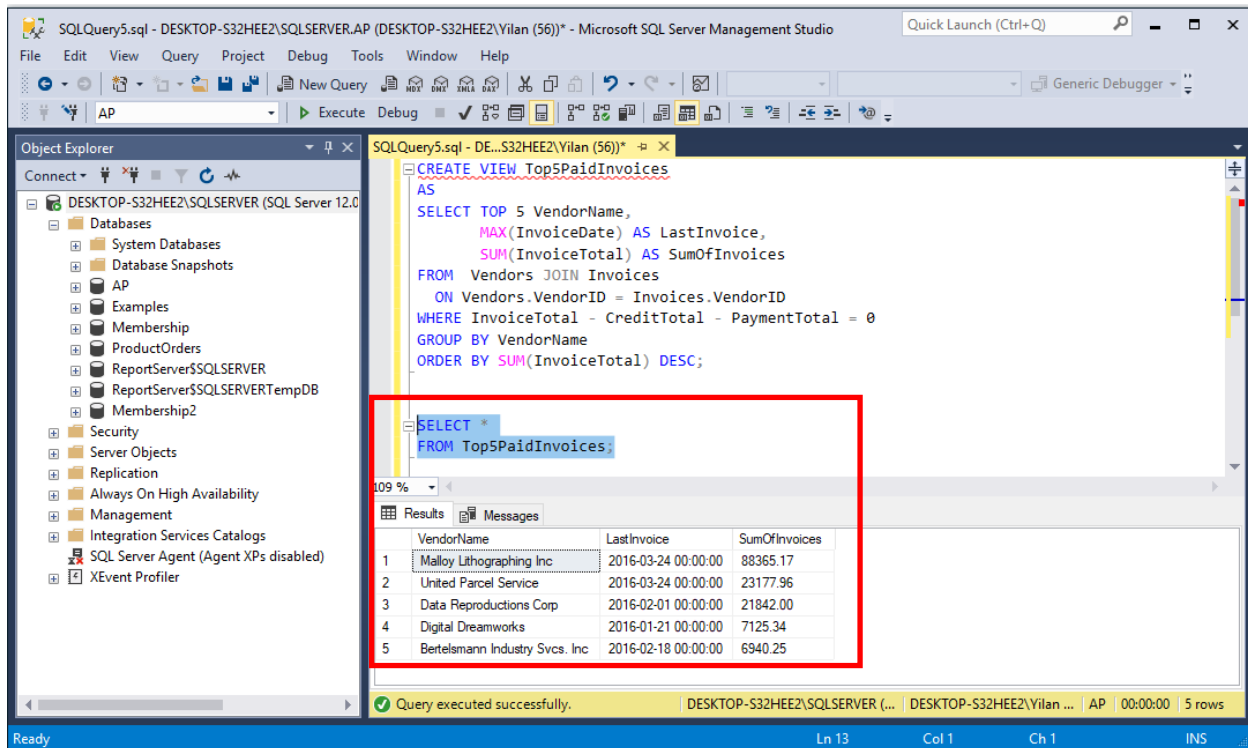
The status bar at the bottom right indicates '9 rows'.

2. Create a view named Top5PaidInvoices that returns three columns for each vendor: VendorName, LastInvoiceDate (the most recent invoice date), and SumOfInvoices (the sum of the InvoiceTotal column). Return only the 5 vendors with the largest SumOfInvoices and include only paid invoices (i.e. InvoiceTotal – CreditTotal – PaymentTotal = 0). Then write a SELECT statement to show results of the view.

```
CREATE VIEW Top5PaidInvoices
AS
SELECT TOP 5 VendorName,
             MAX(InvoiceDate) AS LastInvoice,
             SUM(InvoiceTotal) AS SumOfInvoices
FROM Vendors JOIN Invoices
ON Vendors.VendorID = Invoices.VendorID
WHERE InvoiceTotal - CreditTotal - PaymentTotal = 0
GROUP BY VendorName
ORDER BY SUM(InvoiceTotal) DESC;
```



```
SELECT *  
FROM Top5PaidInvoices;
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor displays the following SQL code:

```
CREATE VIEW Top5PaidInvoices  
AS  
SELECT TOP 5 VendorName,  
             MAX(InvoiceDate) AS LastInvoice,  
             SUM(InvoiceTotal) AS SumOfInvoices  
FROM Vendors JOIN Invoices  
ON Vendors.VendorID = Invoices.VendorID  
WHERE InvoiceTotal - CreditTotal - PaymentTotal = 0  
GROUP BY VendorName  
ORDER BY SUM(InvoiceTotal) DESC;
```

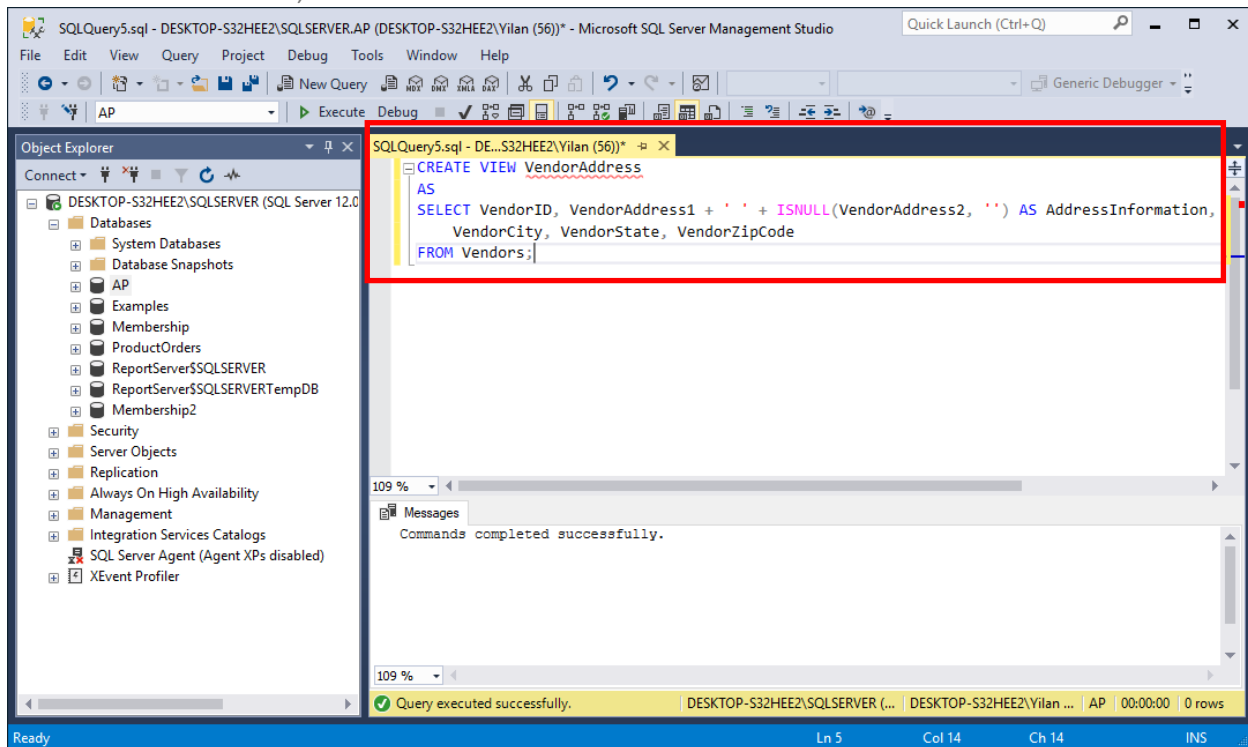
Below the query editor, the 'Results' tab shows the output of the query. The results are displayed in a table with 5 rows and 3 columns: VendorName, LastInvoice, and SumOfInvoices.

	VendorName	LastInvoice	SumOfInvoices
1	Malloy Lithographing Inc	2016-03-24 00:00:00	88365.17
2	United Parcel Service	2016-03-24 00:00:00	23177.96
3	Data Reproductions Corp	2016-02-01 00:00:00	21842.00
4	Digital Dreamworks	2016-01-21 00:00:00	7125.34
5	Bertelsmann Industry Svcs. Inc	2016-02-18 00:00:00	6940.25

The status bar at the bottom indicates 'Query executed successfully.' and '5 rows'.

3. Create an updatable views named VendorAddress that returns the VendorID, Address (i.e. VendorAddress1 + ' ' + VendorAddress2), and the city, state, and zip code columns for each vendor. Then write a SELECT query to examine the result set where VendorID=6. Write a SELECT statement to verify the result.

```
CREATE VIEW VendorAddress
AS
SELECT VendorID, VendorAddress1 + ' ' + ISNULL(VendorAddress2, '') AS AddressInformation, VendorCity, VendorState, VendorZipCode
FROM Vendors;
```



```
SELECT *  
FROM VendorAddress;
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'DESKTOP-S32HEE2\SQLSERVER (SQL Server 12.0)'. The central pane shows a query window with the following SQL code:

```
CREATE VIEW VendorAddress  
AS  
SELECT VendorID, VendorAddress1 + ' ' + ISNULL(VendorAddress2, '') AS AddressInformation,  
       VendorCity, VendorState, VendorZipCode  
FROM Vendors;  
  
SELECT *  
FROM VendorAddress;
```

The Results pane at the bottom displays a table with 6 columns: VendorID, AddressInformation, VendorCity, VendorState, and VendorZipCode. The table contains 6 rows of data. A status bar at the bottom right indicates '122 rows'.

VendorID	AddressInformation	VendorCity	VendorState	VendorZipCode
1	Attn: Supt. Window Services PO Box 7005	Madison	WI	53707
2	PO Box 96621	Washington	DC	20090
3	Library Of Congress	Washington	DC	20559
4	1990 Westwood Blvd Ste 260	Los Angeles	CA	90025
5	3000 Cindel Drive	Washington	NJ	07882
6	3255 Ramos Cir	Sacramento	CA	95827

```
SELECT *  
FROM VendorAddress  
WHERE VendorID = 6;
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'DESKTOP-S32HEE2\SQLSERVER (SQL Server 12.0)'. The central pane shows a query window with the following SQL code:

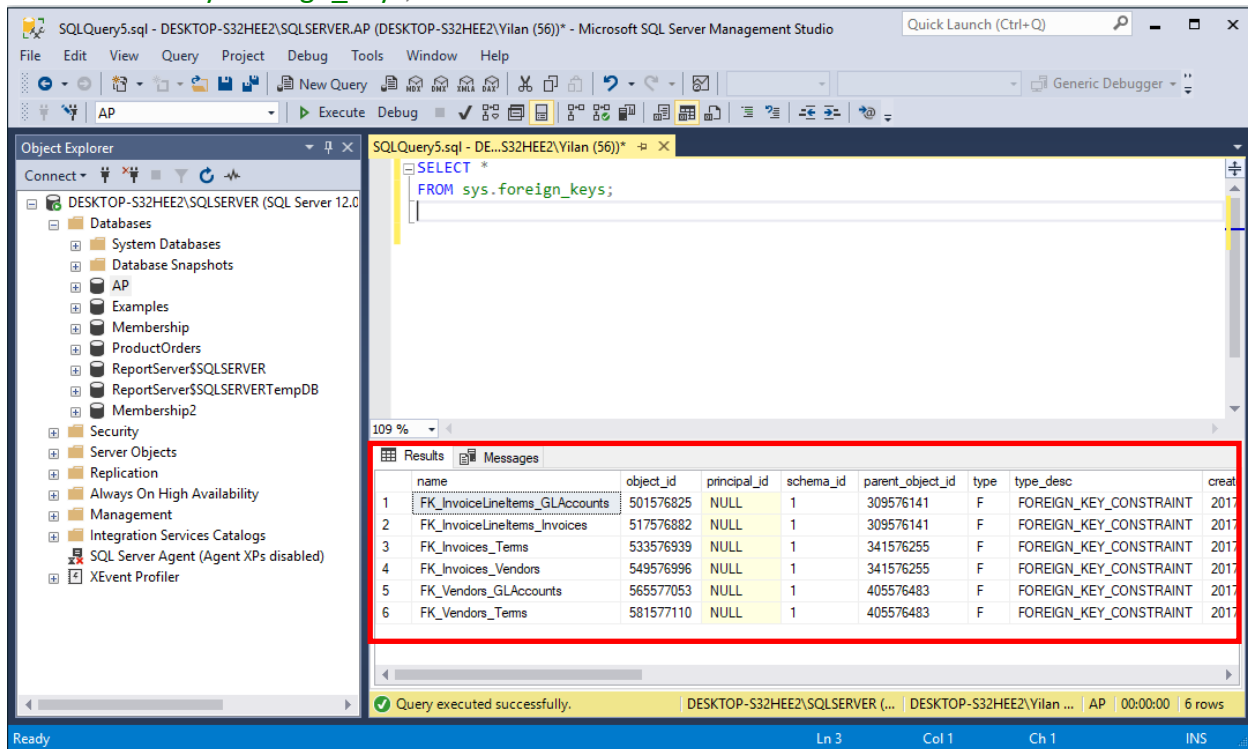
```
CREATE VIEW VendorAddress  
AS  
SELECT VendorID, VendorAddress1 + ' ' + ISNULL(VendorAddress2, '') AS AddressInformation,  
       VendorCity, VendorState, VendorZipCode  
FROM Vendors;  
  
SELECT *  
FROM VendorAddress;  
  
-- VendorID = 6  
SELECT *  
FROM VendorAddress  
WHERE VendorID = 6;
```

The Results pane at the bottom displays a table with 5 columns: VendorID, AddressInformation, VendorCity, VendorState, and VendorZipCode. The table contains 1 row of data. A status bar at the bottom right indicates '1 rows'.

VendorID	AddressInformation	VendorCity	VendorState	VendorZipCode
6	3255 Ramos Cir	Sacramento	CA	95827

4. Write a SELECT statement that selects all the columns for the catalog view that returns information about foreign keys. How many foreign keys are defined in the AP database and what are they?

```
SELECT *  
FROM sys.foreign_keys;
```



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left shows the database structure, including the AP database. The SQL Query window on the right contains the query: `SELECT * FROM sys.foreign_keys;`. The Results pane at the bottom displays the output of the query, which is a table of foreign keys. The table has columns: name, object_id, principal_id, schema_id, parent_object_id, type, type_desc, and create_date. The results show 6 rows of data, all of which are foreign key constraints in the AP database.

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date
1	FK_InvoiceLineItems_GLAaccounts	501576825	NULL	1	309576141	F	FOREIGN_KEY_CONSTRAINT	2017
2	FK_InvoiceLineItems_Invoices	517576882	NULL	1	309576141	F	FOREIGN_KEY_CONSTRAINT	2017
3	FK_Invoices_Terms	533576939	NULL	1	341576255	F	FOREIGN_KEY_CONSTRAINT	2017
4	FK_Invoices_Vendors	549576996	NULL	1	341576255	F	FOREIGN_KEY_CONSTRAINT	2017
5	FK_Vendors_GLAaccounts	565577053	NULL	1	405576483	F	FOREIGN_KEY_CONSTRAINT	2017
6	FK_Vendors_Terms	581577110	NULL	1	405576483	F	FOREIGN_KEY_CONSTRAINT	2017

Query executed successfully. DESKTOP-S32HEE2\SQLSERVER (... DESKTOP-S32HEE2\Vilan ... AP 00:00:00 6 rows

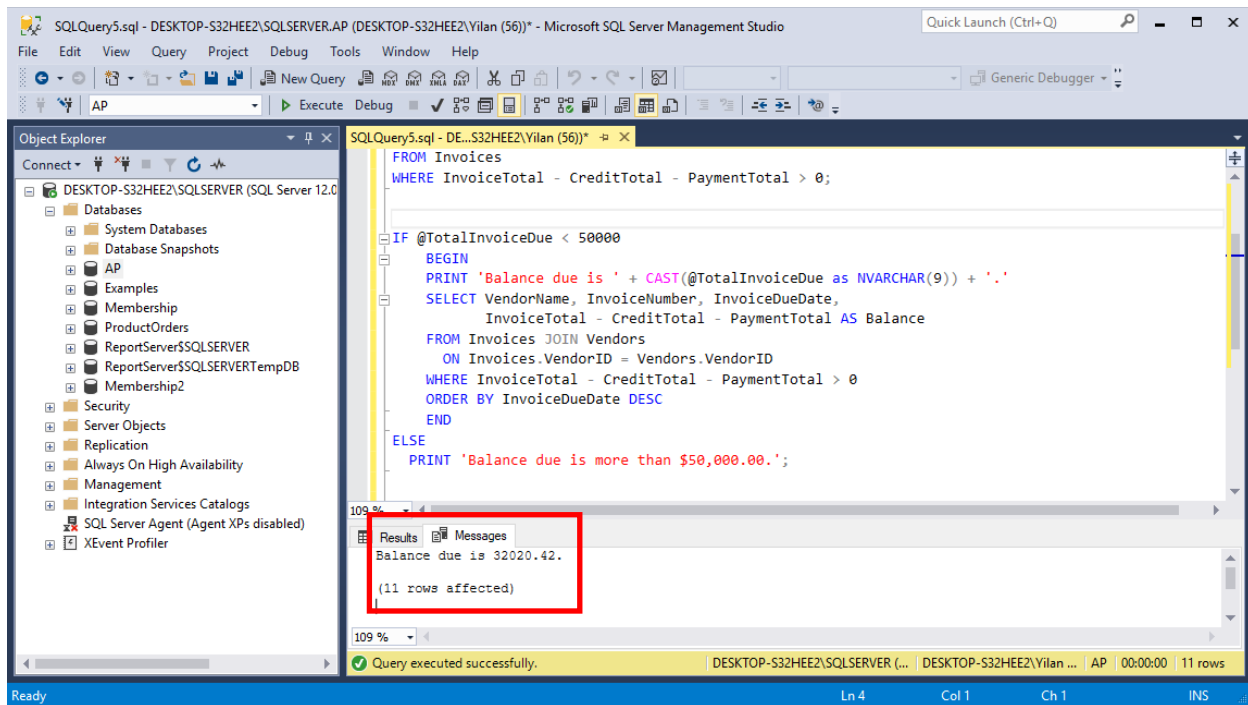
5. Write a script that declares and sets a variable named @TotalBalanceDue, which is equal to the total outstanding balance due. What is the datatype of the variable @TotalBalanceDue? If that balance due is less than \$50,000.00, the script should return a result set consisting of VendorName, InvoiceNumber, InvoiceDueDate, and Balance for each invoice with a balance due, sorted with the newest due date first. Then also return the value of @TotalBalanceDue in the format of "Balance due is ...". If the total outstanding balance due is more than \$50,000.00, the script should return the message "Balance due is more than \$50,000.00".

```
USE AP;
-- declare name and datatype of the avriable
DECLARE @TotalInvoiceDue money;
-- set the variable
SELECT @TotalInvoiceDue =
    SUM(InvoiceTotal - CreditTotal - PaymentTotal)
FROM Invoices
WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0;

IF @TotalInvoiceDue < 50000
BEGIN
    PRINT 'Balance due is ' + CAST(@TotalInvoiceDue as NVARCHAR(9)) + '.';
    SELECT VendorName, InvoiceNumber, InvoiceDueDate,
        InvoiceTotal - CreditTotal - PaymentTotal AS Balance
    FROM Invoices JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
    WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
    ORDER BY InvoiceDueDate DESC
END
ELSE
    PRINT 'Balance due is more than $50,000.00.';
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The query editor shows the T-SQL script being executed. The Results pane shows the output of the query, which is a table with 11 rows of invoice data. The status bar at the bottom indicates that the query executed successfully and returned 11 rows.

VendorName	InvoiceNumber	InvoiceDueDate	Balance
Malloy Lithographing Inc	0-2436	2016-04-30 00:00:00	10976.06
Blue Cross	547480102	2016-04-30 00:00:00	224.00
Ford Motor Credit Company	9982771	2016-04-23 00:00:00	503.20
Malloy Lithographing Inc	P-0608	2016-04-22 00:00:00	19351.18
Federal Express Corporation	263253270	2016-04-21 00:00:00	67.92
Federal Express Corporation	263253273	2016-04-21 00:00:00	30.75
Federal Express Corporation	263253268	2016-04-20 00:00:00	59.97
Federal Express Corporation	963253264	2016-04-17 00:00:00	52.25
Cardinal Business Media, I...	134116	2016-04-17 00:00:00	90.36
Ingram	31361833	2016-04-10 00:00:00	579.42
Data Reproductions Corp	39104	2016-04-09 00:00:00	85.31



6. Explain the execution result generated by the following script. Then Write a script that generates the same result set but uses a temporary table in place of the derived table. Make sure your script tests for the existence of any objects it creates.

(1) The following script uses a derived table to return the date and invoice total of the latest invoice issued by each vendor.

```
USE AP;
SELECT VendorName, LastInvoiceDate, InvoiceTotal
FROM Invoices JOIN
    (SELECT VendorID, MAX(InvoiceDate) AS LastInvoiceDate
     FROM Invoices
     GROUP BY VendorID) AS LastInvoice
ON (Invoices.VendorID = LastInvoice.VendorID AND
    Invoices.InvoiceDate = LastInvoice.LastInvoiceDate)
JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
ORDER BY VendorName, LastInvoiceDate;
```

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The 'Object Explorer' on the left shows the database structure. The 'Query Editor' in the center shows the SQL script. The 'Results' pane at the bottom displays the output of the query, which is a table with 3 columns: VendorName, LastInvoiceDate, and InvoiceTotal. The status bar at the bottom indicates that the query was executed successfully and returned 34 rows.

VendorName	LastInvoiceDate	InvoiceTotal
Abbey Office Furnishings	2016-03-05 00:00:00	17.50
Bertelsmann Industry Svcs. Inc	2016-02-18 00:00:00	6940.25
Blue Cross	2016-04-01 00:00:00	224.00
Cahners Publishing Company	2016-02-28 00:00:00	2184.50
Cardinal Business Media, Inc.	2016-03-28 00:00:00	90.36
Coffee Break Service	2016-02-24 00:00:00	41.80
Compuserve	2016-01-13 00:00:00	9.95
Computerworld	2016-02-11 00:00:00	2433.00

```

USE AP;
IF OBJECT_ID('tempdb..#LastInvoice') IS NOT NULL
    DROP TABLE #LastInvoice;

SELECT VendorID, MAX(InvoiceDate) AS LastInvoiceDate
INTO #LastInvoice
FROM Invoices
GROUP BY VendorID;

SELECT VendorName, LastInvoiceDate, InvoiceTotal
FROM Invoices JOIN #LastInvoice
    ON (Invoices.VendorID = #LastInvoice.VendorID AND
        Invoices.InvoiceDate = #LastInvoice.LastInvoiceDate)
JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
ORDER BY VendorName, LastInvoiceDate;

```

SQLQuery5.sql - DESKTOP-S32HEE2\SQLSERVER.AP (DESKTOP-S32HEE2\Yilan (56)) - Microsoft SQL Server Management Studio

Object Explorer: DESKTOP-S32HEE2\SQLSERVER (SQL Server 12.0)

Query Window: SQLQuery5.sql - DE...S32HEE2\Yilan (56))

```

USE AP;
IF OBJECT_ID('tempdb..#LastInvoice') IS NOT NULL
    DROP TABLE #LastInvoice;

SELECT VendorID, MAX(InvoiceDate) AS LastInvoiceDate
INTO #LastInvoice
FROM Invoices
GROUP BY VendorID;

SELECT VendorName, LastInvoiceDate, InvoiceTotal
FROM Invoices JOIN #LastInvoice
    ON (Invoices.VendorID = #LastInvoice.VendorID AND
        Invoices.InvoiceDate = #LastInvoice.LastInvoiceDate)
JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
ORDER BY VendorName, LastInvoiceDate;

```

Results:

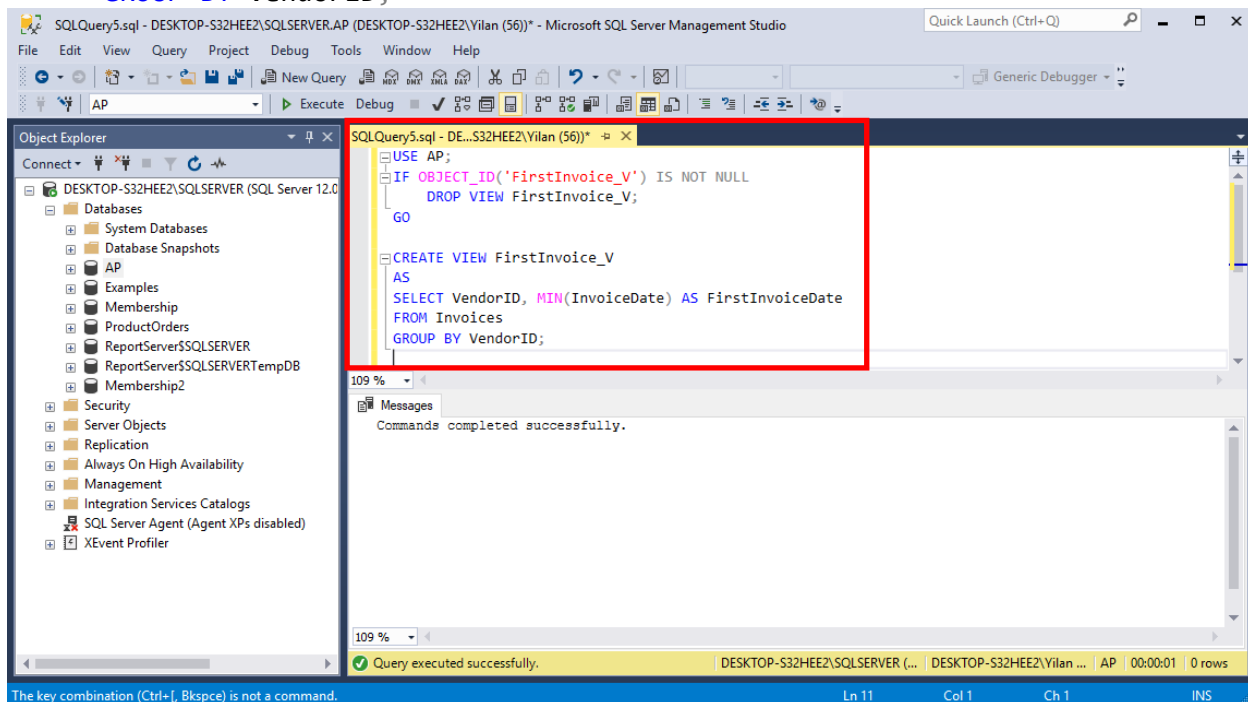
	VendorName	LastInvoiceDate	InvoiceTotal
1	Abbey Office Furnishings	2016-03-05 00:00:00	17.50
2	Bertelsmann Industry Svcs. Inc	2016-02-18 00:00:00	6940.25
3	Blue Cross	2016-04-01 00:00:00	224.00
4	Cahners Publishing Company	2016-02-28 00:00:00	2184.50
5	Cardinal Business Media, Inc.	2016-03-28 00:00:00	90.36

Query executed successfully. DESKTOP-S32HEE2\SQLSERVER (... DESKTOP-S32HEE2\Yilan ... AP 00:00:00 34 rows

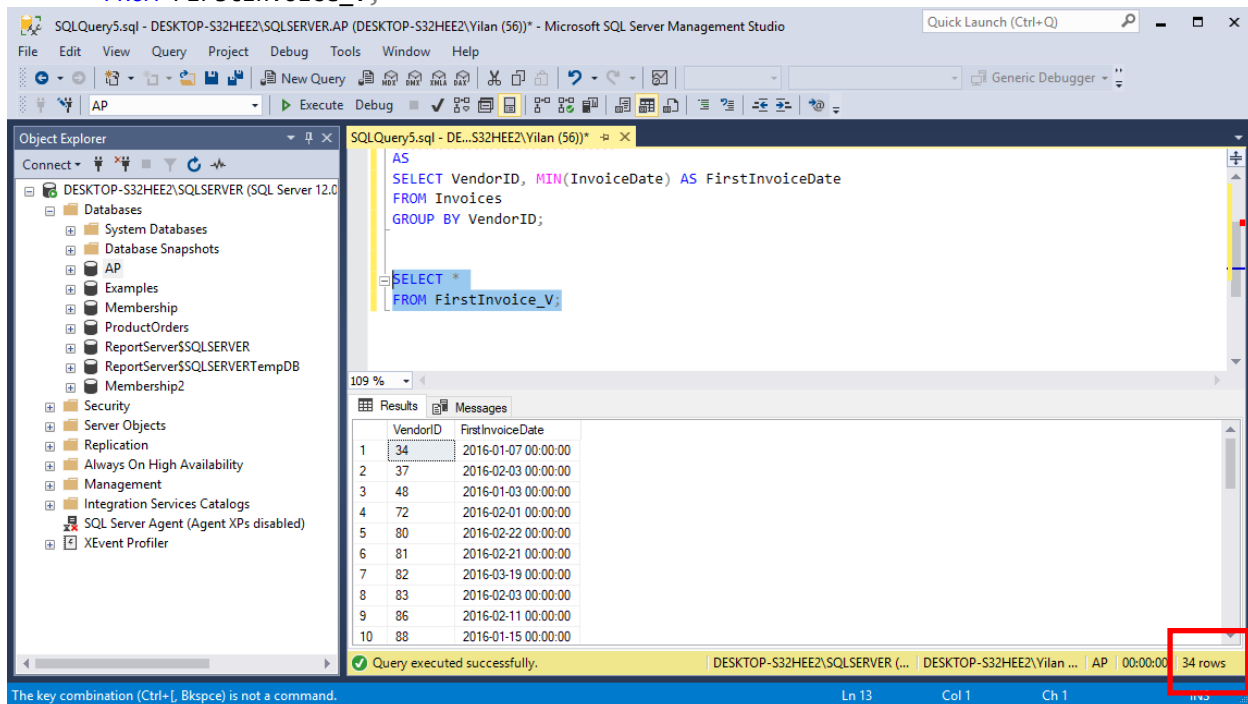
7. Write a script that generates the date and invoice total of the earliest invoice issued by each vendor, using a view instead of a derived table. Also write the script that creates the view, then use SELECT statement to show result of the view. Make sure that your script tests for the existence of the view. The view doesn't need to be redefined each time the script is executed.

```
USE AP;
IF OBJECT_ID('FirstInvoice_V') IS NOT NULL
    DROP VIEW FirstInvoice_V;
GO

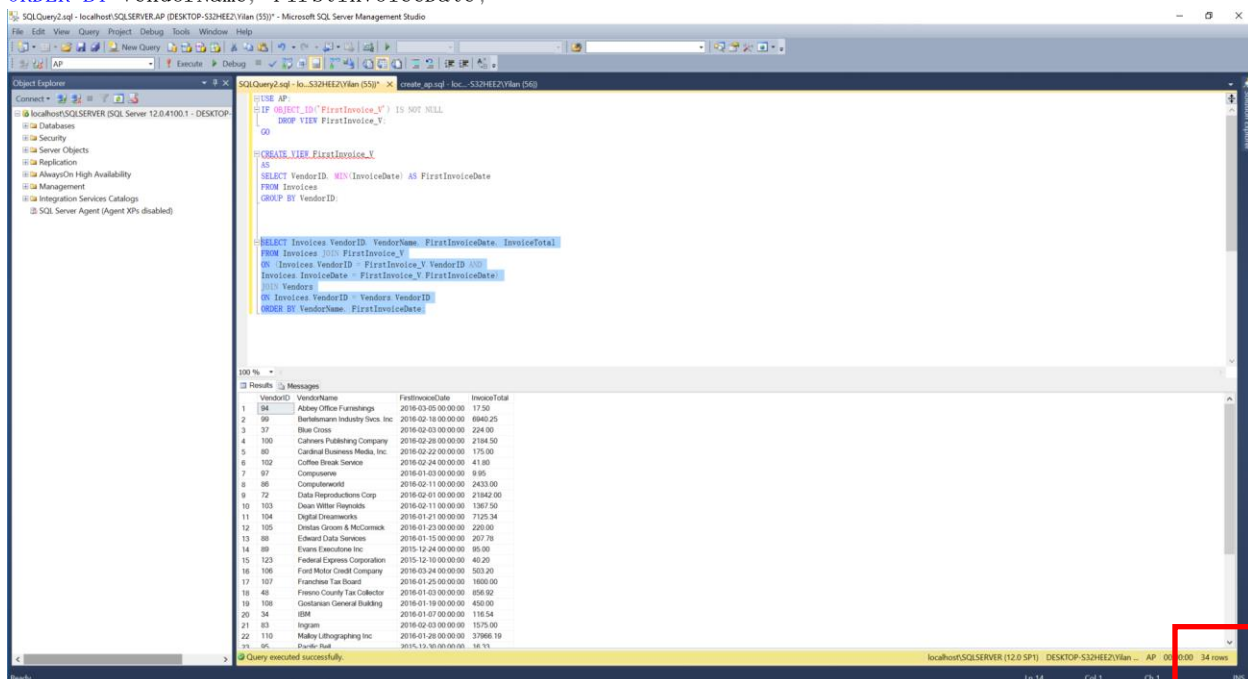
CREATE VIEW FirstInvoice_V
AS
SELECT VendorID, MIN(InvoiceDate) AS FirstInvoiceDate
FROM Invoices
GROUP BY VendorID;
```



```
SELECT *
FROM FirstInvoice_V;
```



```
SELECT Invoices.VendorID, VendorName, FirstInvoiceDate, InvoiceTotal
FROM Invoices JOIN FirstInvoice_V
ON (Invoices.VendorID = FirstInvoice_V.VendorID AND
Invoices.InvoiceDate = FirstInvoice_V.FirstInvoiceDate)
JOIN Vendors
ON Invoices.VendorID = Vendors.VendorID
ORDER BY VendorName, FirstInvoiceDate;
```



8. Write a script that uses dynamic SQL to return a single column that represents the number of rows in the first table in the current database. The script should automatically choose the table that appears first alphabetically, and it should exclude tables named `dtproperties` and `sysdiagrams`. Name the column `CountOfTable`, where `Table` is the chosen table name.

```
DECLARE @TableName varchar(128);

SELECT @TableName = MIN(name)
FROM sys.tables
WHERE name <> 'dtproperties' AND name <> 'sysdiagrams';

EXEC ('SELECT COUNT(*) AS CountOf' + @TableName +
      ' FROM ' + @TableName + ';');
```

