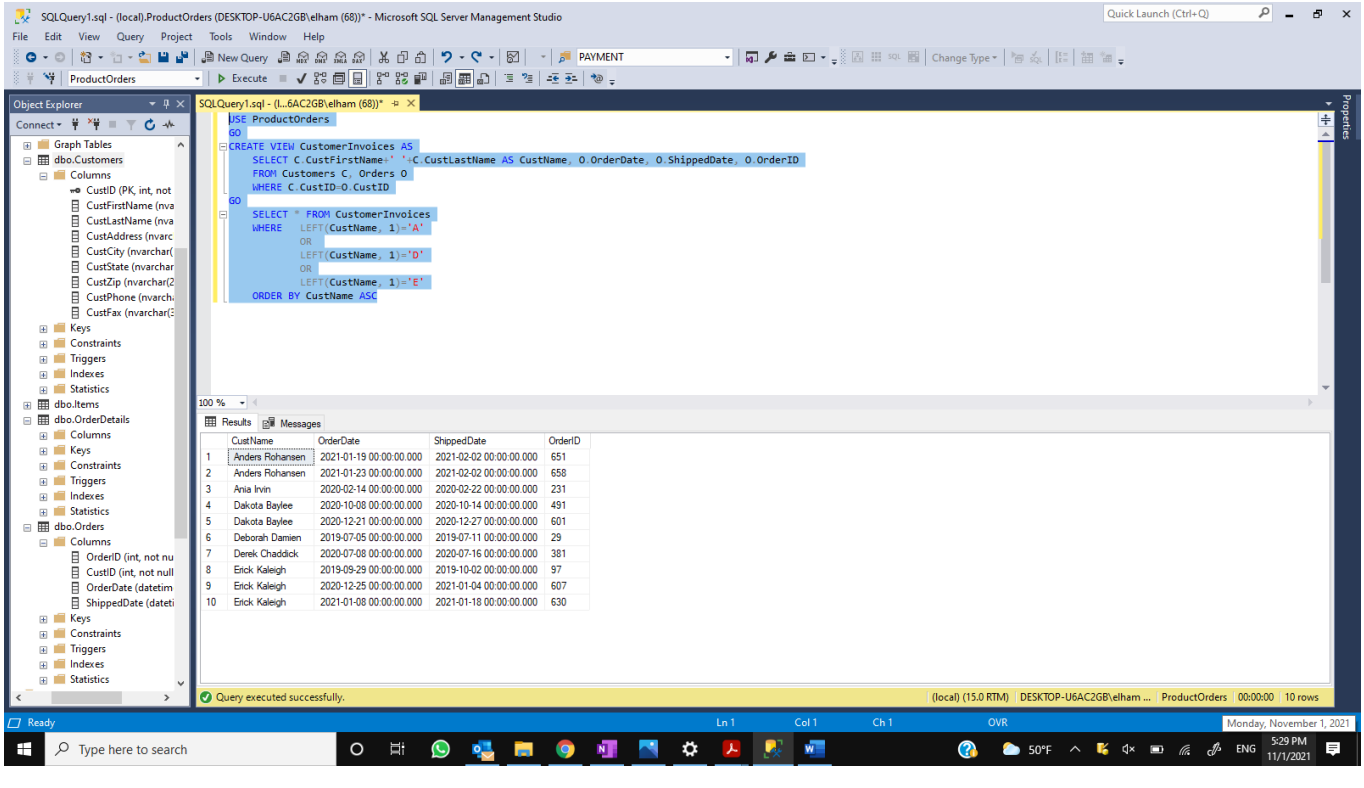
Q1. Write a view named CustomerInvoices that returns four columns: CustName (CustFirstName+ ' '+CustLastName), OrderDate and ShippedDate, OrderID. Then, write a SELECT statement that returns all the columns in the view, sorted by CustName in ascending order, where the first letter of the customer's name is A, D or E. Use ProductOrders Database.

A1.

```
USE ProductOrders
GO
CREATE VIEW CustomerInvoices AS
	SELECT C.CustFirstName+' '+C.CustLastName AS CustName, O.OrderDate, O.ShippedDate,
O.OrderID
	FROM Customers C, Orders O
	WHERE C.CustID=O.CustID
GO
	SELECT * FROM CustomerInvoices
	WHERE  LEFT(CustName, 1)='A'
			OR
			LEFT(CustName, 1)='D'
			OR
			LEFT(CustName, 1)='E'
	ORDER BY CustName ASC
```

Screen Shot:



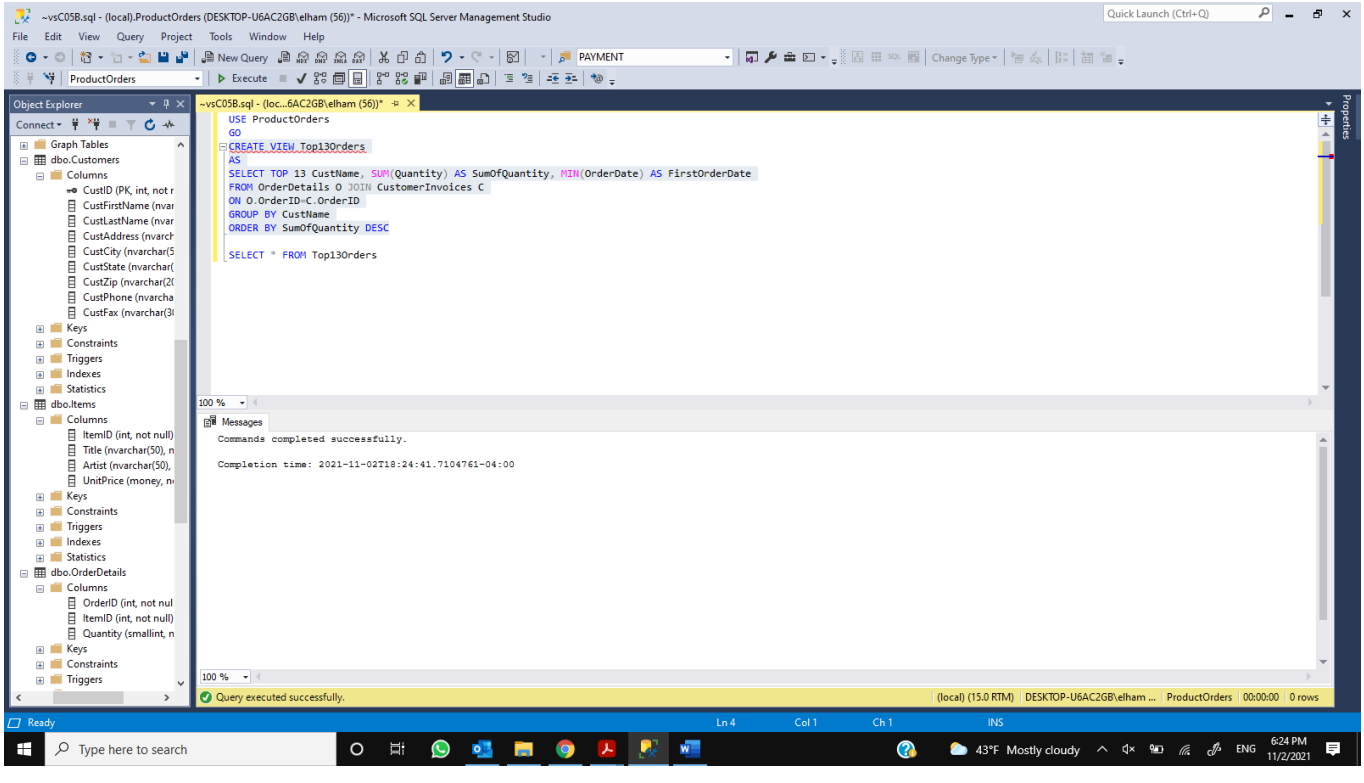
Remark: This result will let us know the full name of the customers who their name starts with either A or D or E, when the order was received and when the order was shipped. So, we can let them know about their status of orders.

Q2. Create a view names Top13Orders that returns three columns: CustName, FirstOrderDate (the least recent invoice date), and SumOfQuantity (the sum of quantities column in OrderDetails. Return only top 13 Customers with the largest quantity bought by the customer. Then write a SELECT statement to show results of the view. Use the view that created in question 1 and OrderDetails table of ProductOrders Database.

A2.

```
CREATE VIEW Top13Orders
AS
SELECT TOP 13 CustName, SUM(Quantity) AS SumOfQuantity, MIN(OrderDate) AS FirstOrderDate
FROM OrderDetails O JOIN CustomerInvoices C
ON O.OrderID=C.OrderID
GROUP BY CustName
ORDER BY SumOfQuantity DESC
```

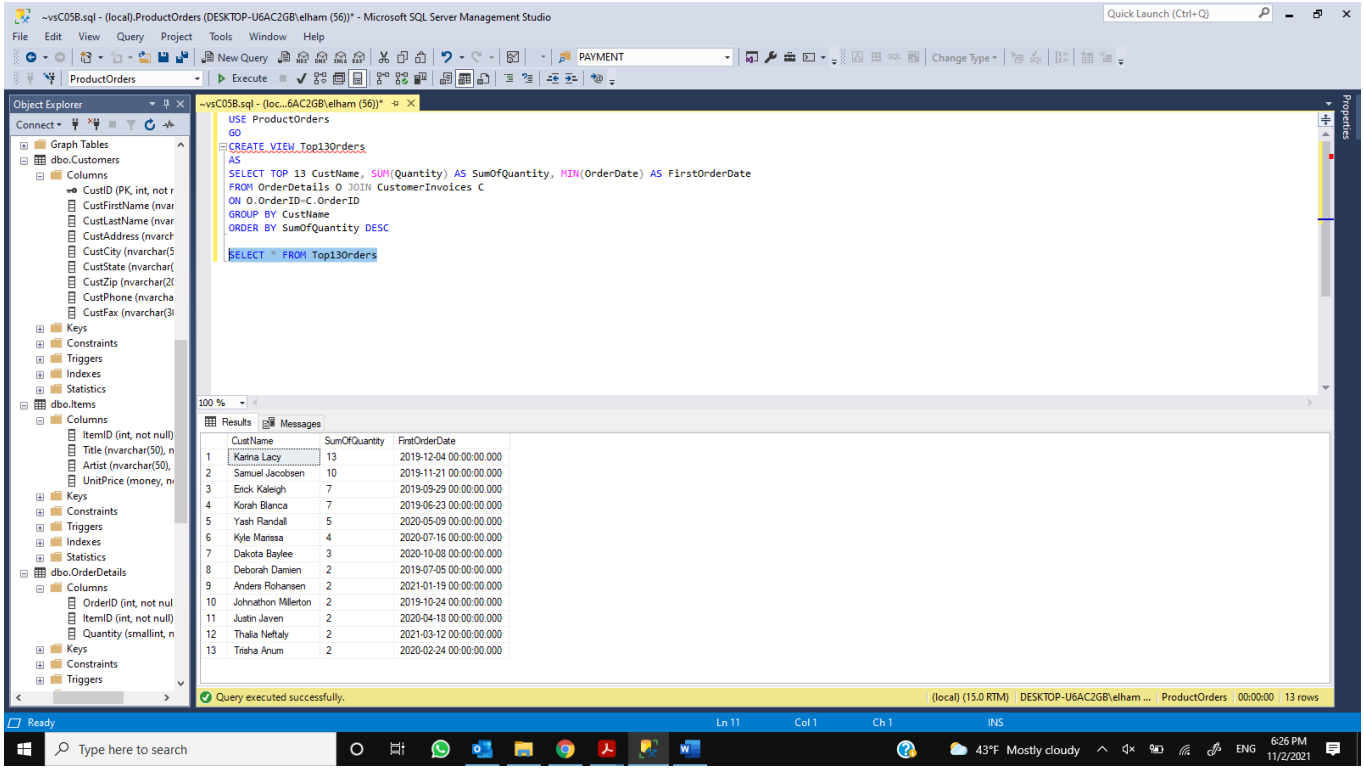
Screen Shot:



A2.

```
SELECT * FROM Top13Orders
```

Screen Shot:



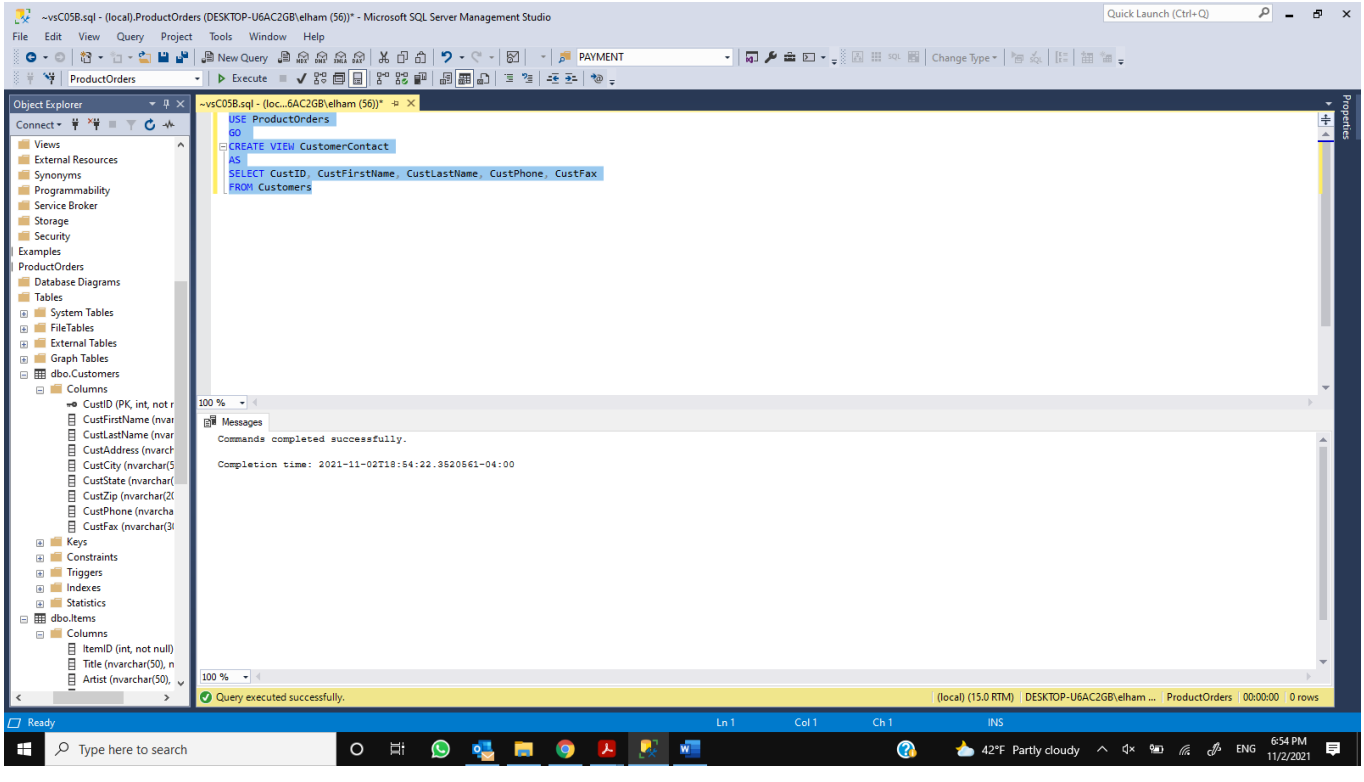
Remark: Here we can see our valuable customers who have ordered the most. We can also see how long they have been our customers so we may use this data to send a discount code and appreciate their presence.

Q3. Create an updatable view named CustomerContact that returns the CustID, CustFirstName, CustLastName, CustPhone, CustFax. Then write a UPDATE query to update CustFax with 9999999999 where CustFax is NULL. Use ProductOrders Database.

A3.

```
USE ProductOrders
GO
CREATE VIEW CustomerContact
AS
SELECT CustID, CustFirstName, CustLastName, CustPhone, CustFax
FROM Customers
```

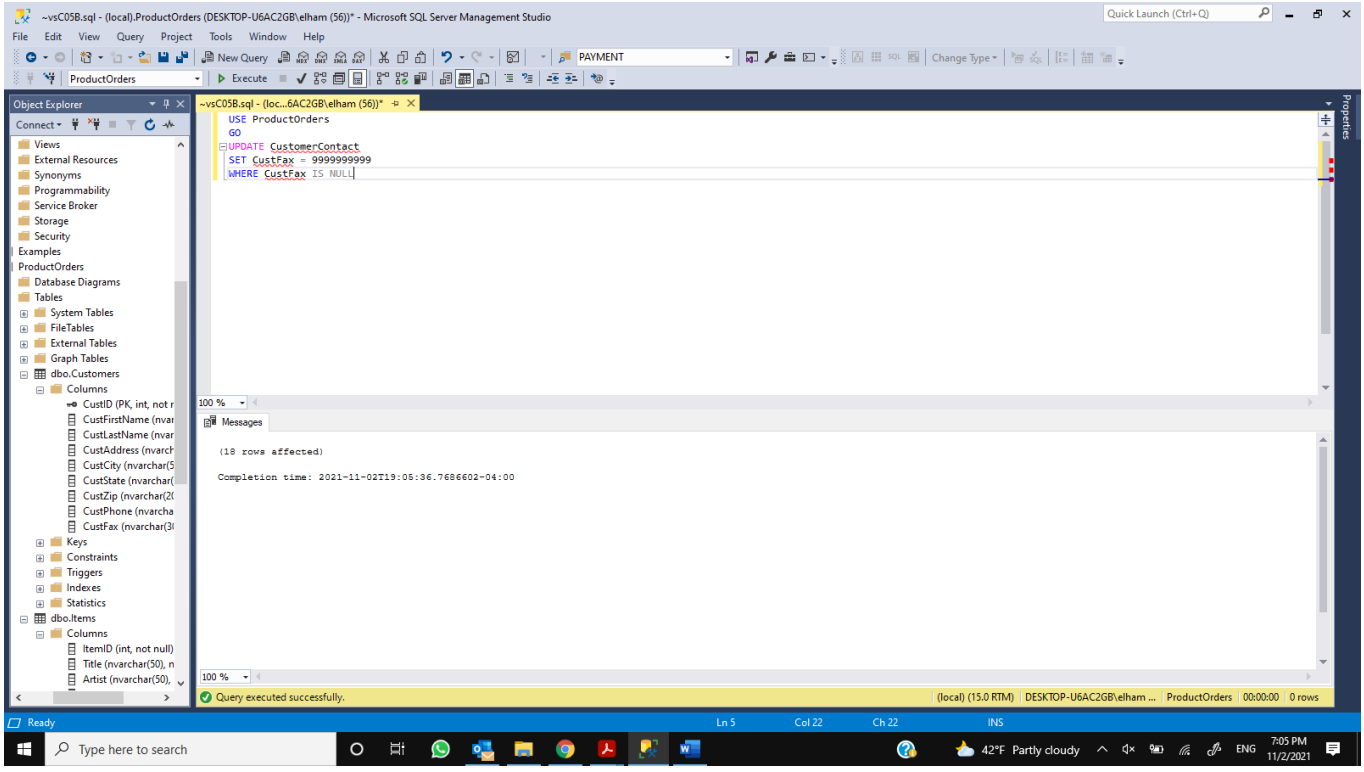
Screen Shot:



A3.

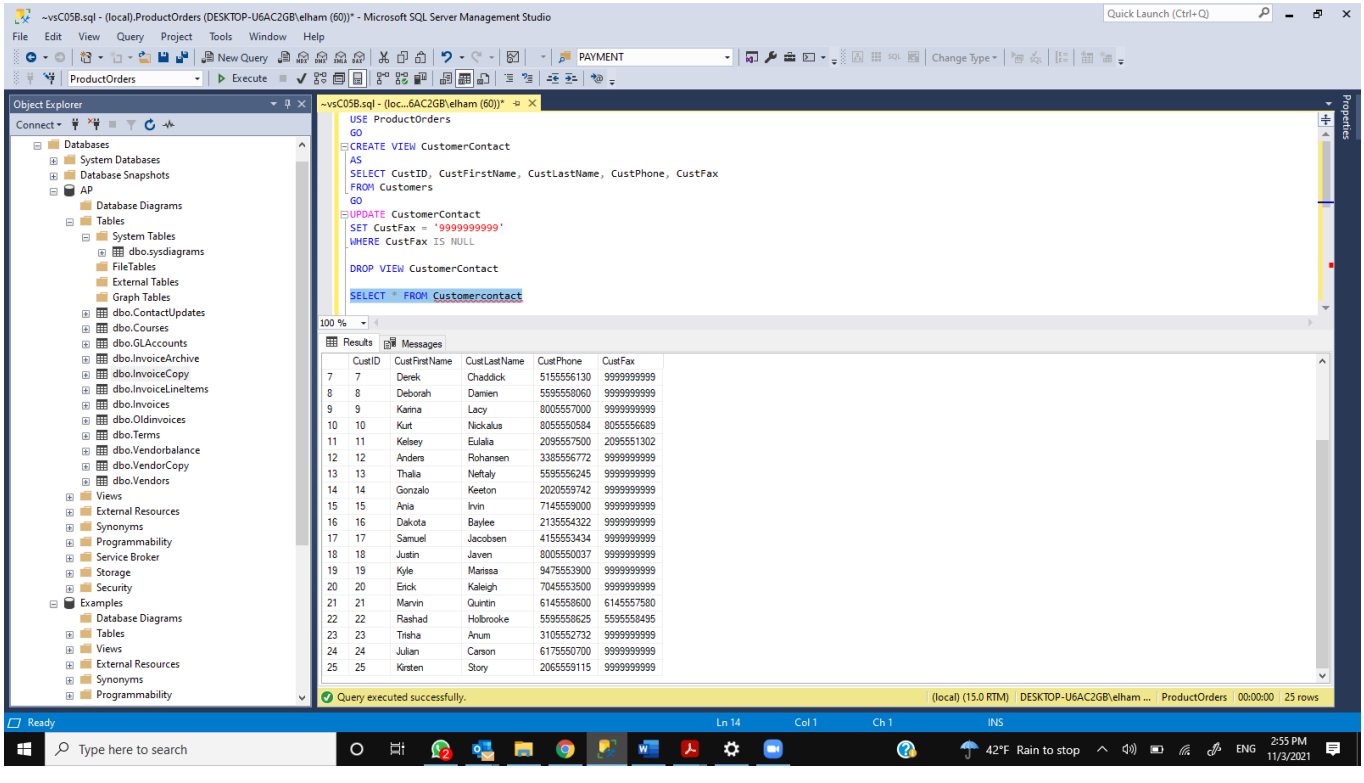
```
USE ProductOrders
GO
UPDATE CustomerContact
SET CustFax = 9999999999
WHERE CustFax IS NULL
```

Screen Shot:



Remark: we now have a view that shows us the customer information which can be used for marketing purposes. When our employees see 9999999999 they understand that the customer have not provided their fax for our database.

Our view has 25 rows

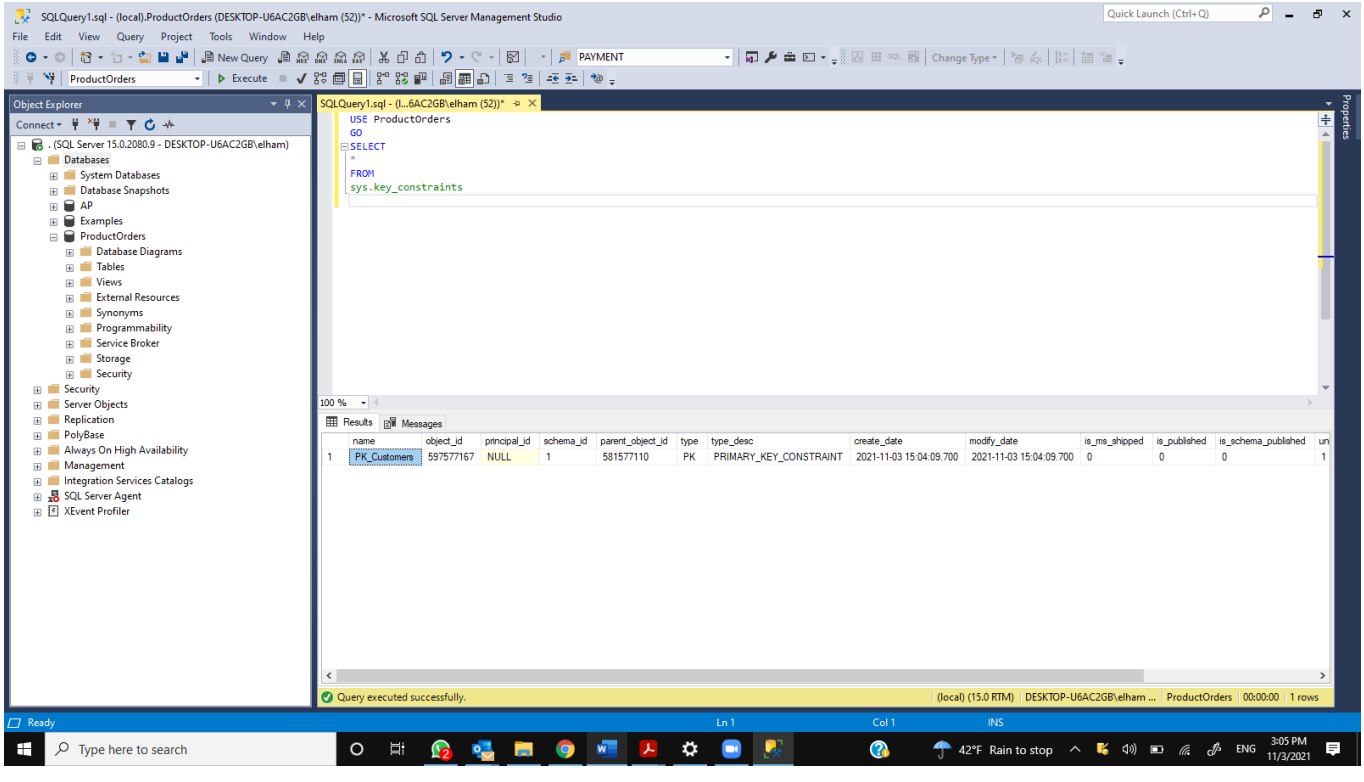


Q4. Write a SELECT statement that selects all the columns for the catalog view that returns information about primary keys in the ProductOrders database. How many primary key(s) is/are defined in the ProductOrders database and what is/are they?

A4.

```
USE ProductOrders
GO
SELECT
*
FROM
sys.key_constraints
```

Screen Shot:



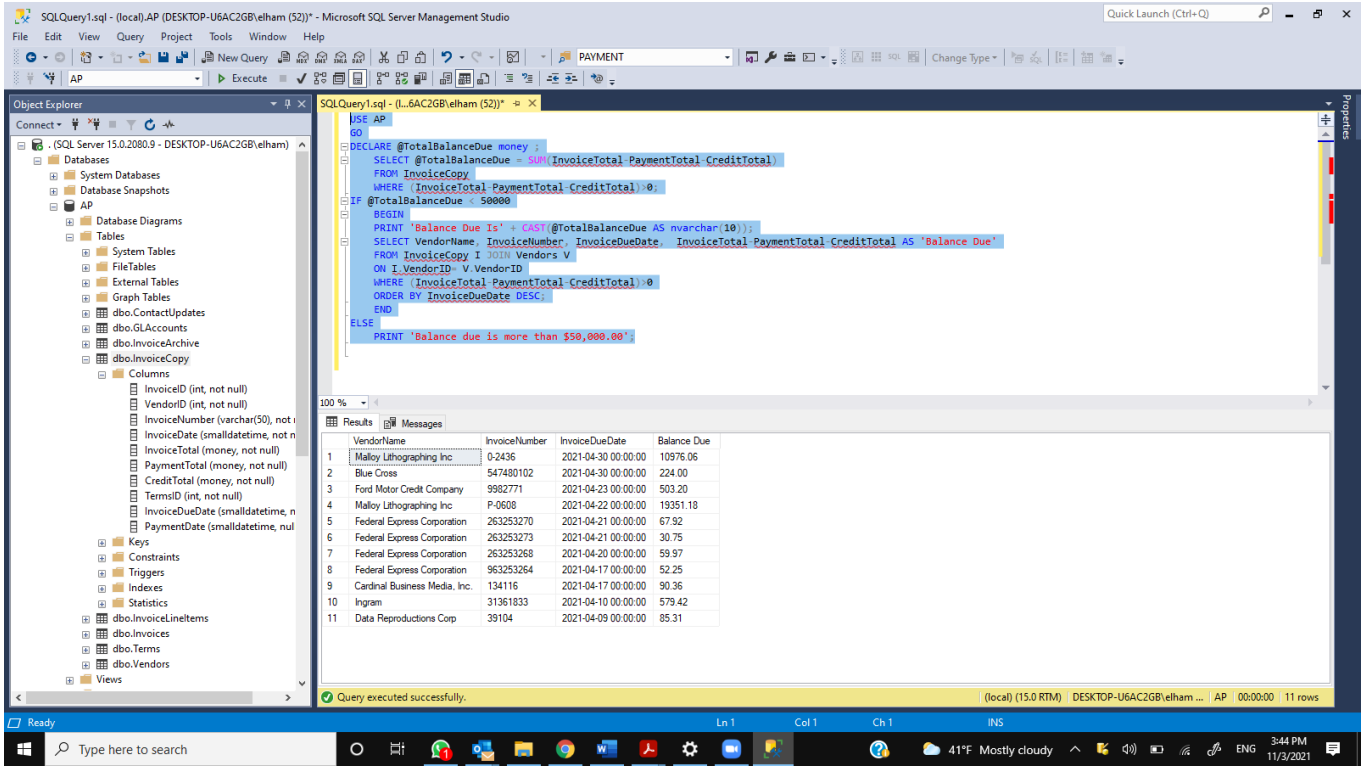
Remark: There is One Primary key defined as can be seen in the result. We have PK_Customers as primary key (PK).

Q5. 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".

A5.

]

Screen Shot:



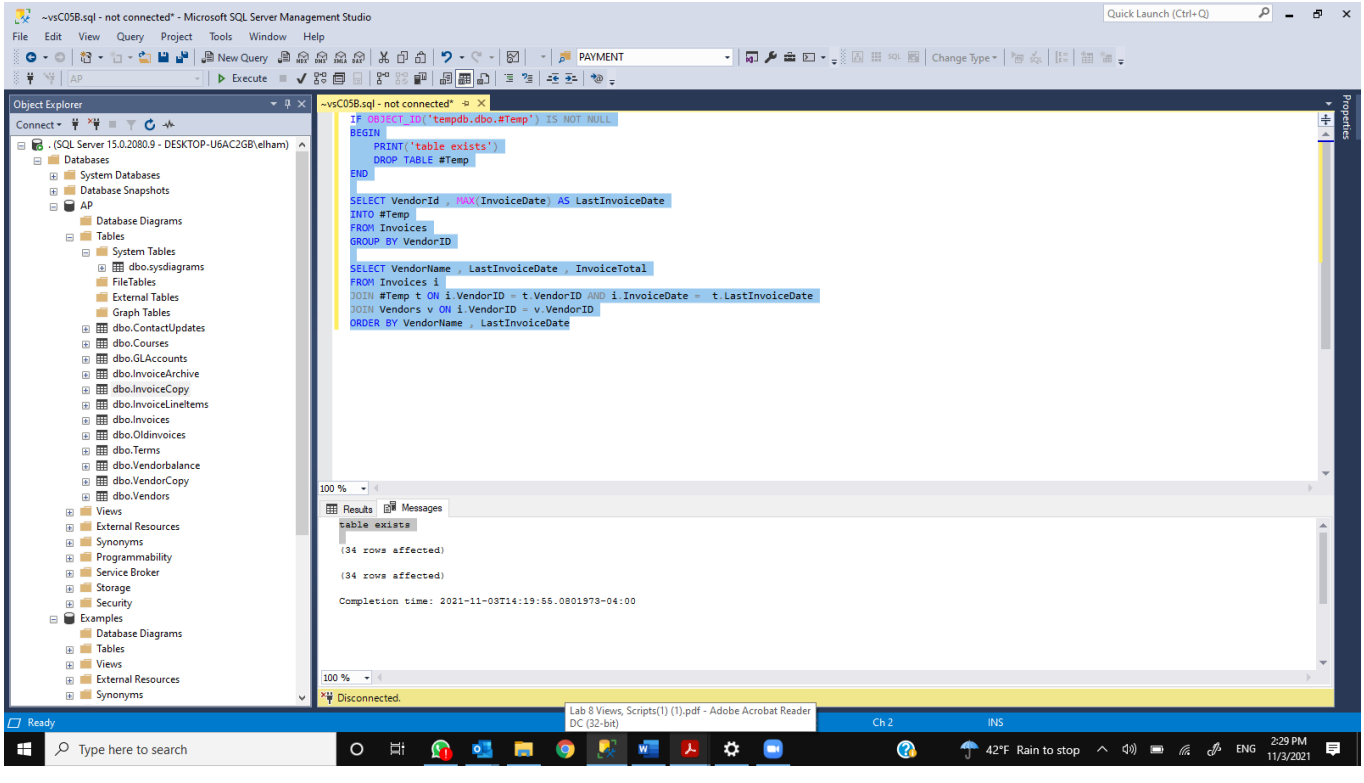
Remark: Here we can see that the value is declared as money and we will be able to find the vendors who had invoices less that $50000.

Q6. 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.

```
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;
```

A6.

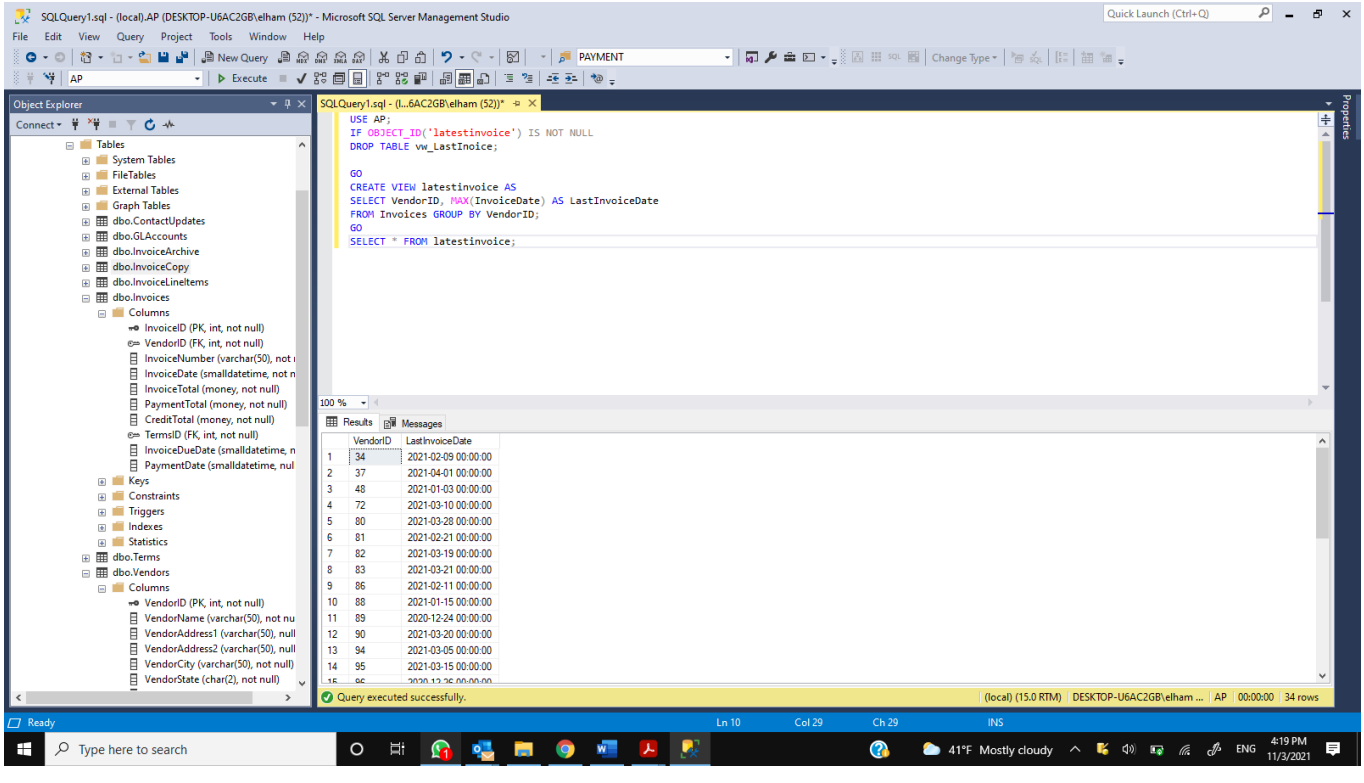
```
IF OBJECT_ID('tempdb.dbo.#Temp') IS NOT NULL
BEGIN
	PRINT('table exists')
	DROP TABLE #Temp
END

SELECT VendorId , MAX(InvoiceDate) AS LastInvoiceDate
INTO #Temp
FROM Invoices
GROUP BY VendorID

SELECT VendorName , LastInvoiceDate , InvoiceTotal
FROM Invoices i
JOIN #Temp t ON i.VendorID = t.VendorID AND i.InvoiceDate =  t.LastInvoiceDate
JOIN Vendors v ON i.VendorID = v.VendorID
ORDER BY VendorName , LastInvoiceDate
```



Remark: This Script uses a derived table to get the invoice date and invoice total amount of the earliest invoices for each vendor

Q7. Write a script that generates the date and invoice total of the latest 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.

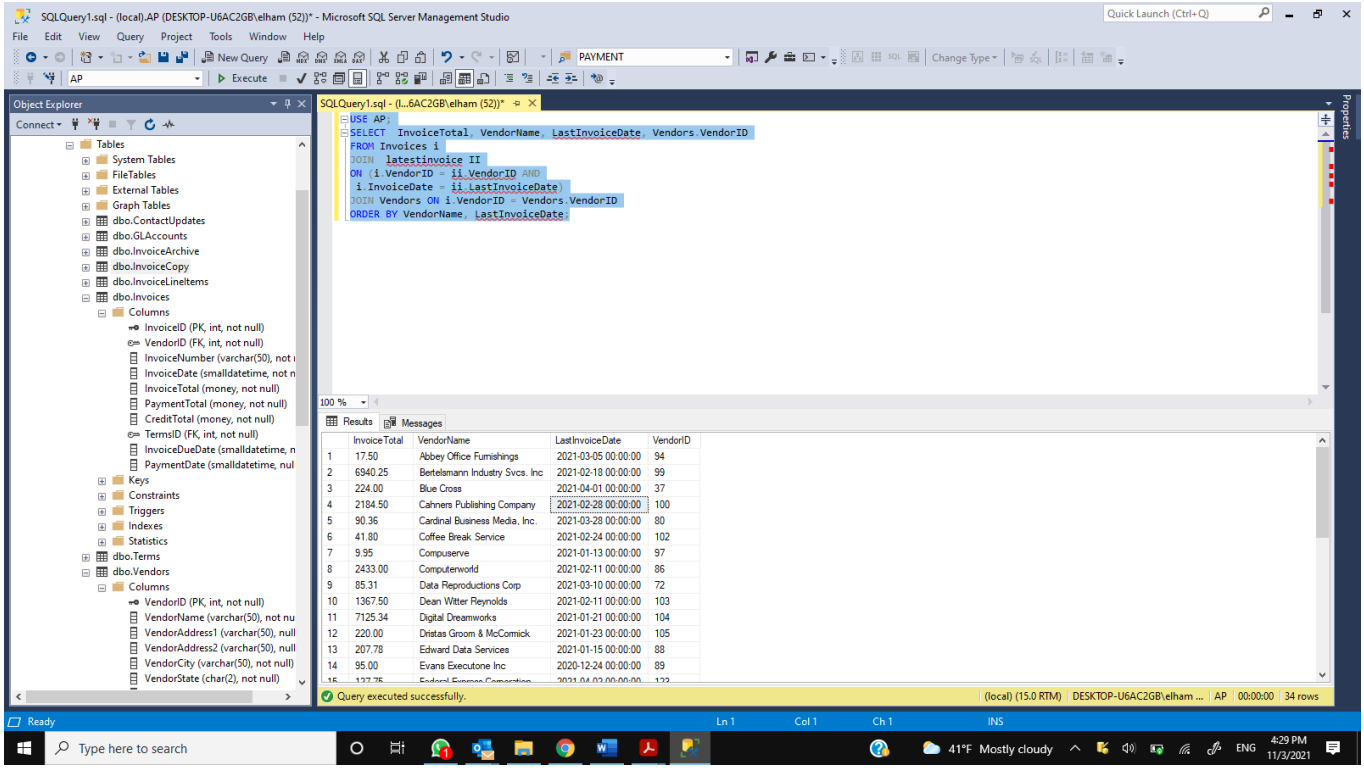
```
USE AP;
IF OBJECT_ID('latestinvoice') IS NOT NULL
DROP VIEW latestinvoice;

GO
CREATE VIEW latestinvoice AS
SELECT VendorID, MAX(InvoiceDate) AS LastInvoiceDate
FROM Invoices GROUP BY VendorID;
GO
SELECT * FROM latestinvoice;
```



Remark : In above query the latestinvoice VIEW is selecting the VendorId and latest InvoiceDate of each vendor from invoices table

A7.

```
USE AP;
SELECT  InvoiceTotal, VendorName, LastInvoiceDate, Vendors.VendorID
FROM Invoices i
JOIN  latestinvoice II
ON (i.VendorID = ii.VendorID AND
 i.InvoiceDate = ii.LastInvoiceDate)
JOIN Vendors ON i.VendorID = Vendors.VendorID
ORDER BY VendorName, LastInvoiceDate;
```



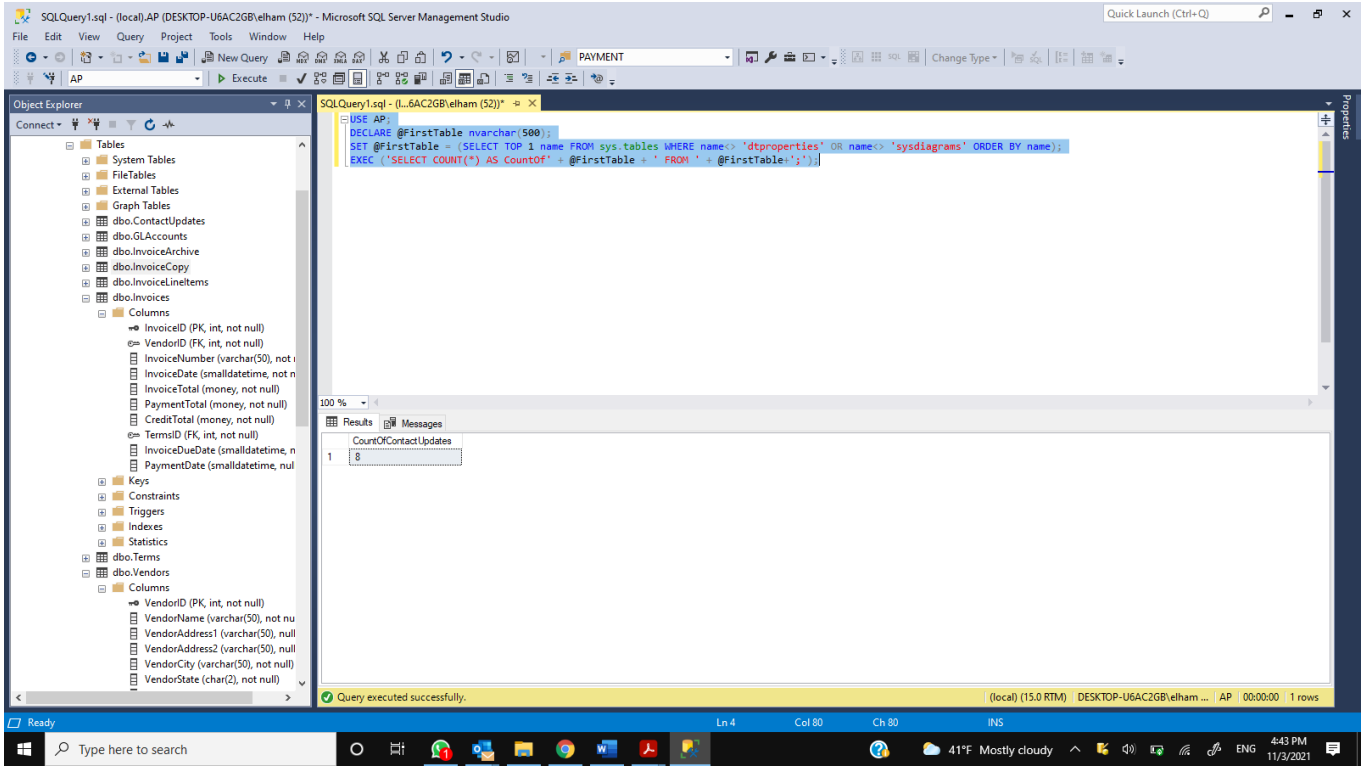
	InvoiceTotal	VendorName	LastInvoiceDate	VendorID
1	17.50	Abbey Office Furnishings	2021-03-05 00:00:00	94
2	6940.25	Bertelsmann Industry Svcs. Inc	2021-02-18 00:00:00	99
3	224.00	Blue Cross	2021-04-01 00:00:00	37
4	2184.50	Cahners Publishing Company	2021-02-28 00:00:00	100
5	90.36	Cardinal Business Media, Inc.	2021-03-28 00:00:00	80
6	41.80	Coffee Break Service	2021-02-24 00:00:00	102
7	9.95	Compuserve	2021-01-13 00:00:00	97
8	2433.00	Computerworld	2021-02-11 00:00:00	86
9	85.31	Data Reproductions Corp	2021-03-10 00:00:00	72
10	1367.50	Dean Witter Reynolds	2021-02-11 00:00:00	103
11	7125.34	Digital Dreamworks	2021-01-21 00:00:00	104
12	220.00	Dristas Groom & McCormick	2021-01-23 00:00:00	105
13	207.78	Edward Data Services	2021-01-15 00:00:00	88
14	95.00	Evans Executone Inc	2020-12-24 00:00:00	89

Remark: Here we can see the invoice total and last invoice date and vendorID and we can realize what is the

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. Show results for AP database.

A8.

```
USE AP;
DECLARE @FirstTable nvarchar(500);
SET @FirstTable = (SELECT TOP 1 name FROM sys.tables WHERE name<> 'dtproperties' OR name<>
'sysdiagrams' ORDER BY name);
EXEC ('SELECT COUNT(*) AS CountOf' + @FirstTable + ' FROM ' + @FirstTable+';');
```



Remark: The sys.tables table of a database represent it's tables so we select the name of it order by it's name where it isn't equal to dtproperties and sysdiagrams the by using EXEC command we run a dynamic query which select the count of rows from the table found in the first query (@FirstTable)