

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:

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'ProductOrders' database selected. The right pane shows the 'SQL Query 1.sql' window with the following SQL code:

```
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
```

The bottom pane shows the 'Results' window with the following data:

	CustName	OrderDate	ShippedDate	OrderID
1	Anders Rishansen	2021-01-19 00:00:00.000	2021-02-02 00:00:00.000	651
2	Anders Rishansen	2021-01-23 00:00:00.000	2021-02-02 00:00:00.000	658
3	Ania Irvin	2020-02-14 00:00:00.000	2020-02-22 00:00:00.000	231
4	Dakota Baylee	2020-10-08 00:00:00.000	2020-10-14 00:00:00.000	491
5	Dakota Baylee	2020-12-21 00:00:00.000	2020-12-27 00:00:00.000	601
6	Deborah Damien	2019-07-05 00:00:00.000	2019-07-11 00:00:00.000	29
7	Derek Chaddick	2020-07-08 00:00:00.000	2020-07-16 00:00:00.000	381
8	Erick Kaleigh	2019-09-29 00:00:00.000	2019-10-02 00:00:00.000	97
9	Erick Kaleigh	2020-12-25 00:00:00.000	2021-01-04 00:00:00.000	607
10	Erick Kaleigh	2021-01-08 00:00:00.000	2021-01-18 00:00:00.000	630

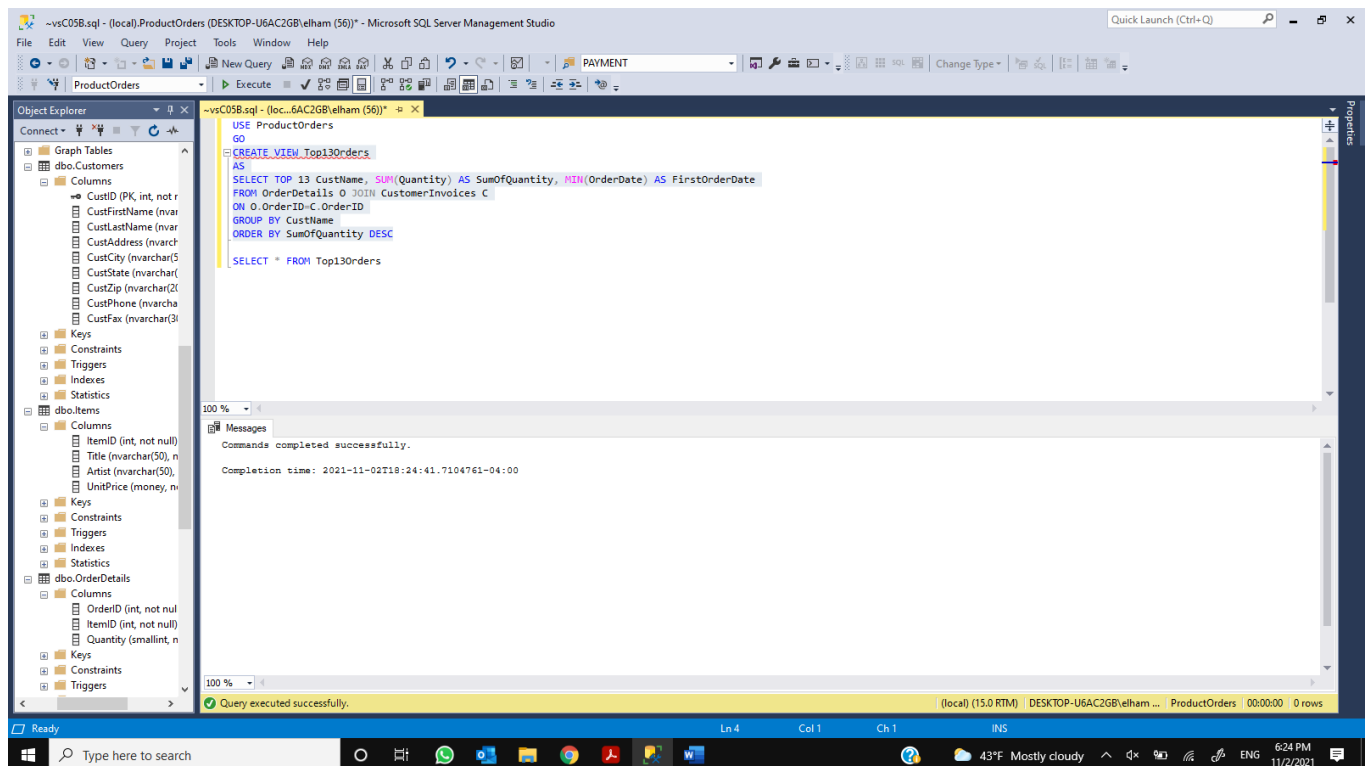
Remark: This result will let us know the full name of the customers who their name starts with either A or D or E, when the order was received and when the order was shipped. So, we can let them know about their status of orders.

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 Top130Orders`

Screen Shot:

The screenshot displays the Microsoft SQL Server Management Studio interface. The left pane shows the Object Explorer with the database structure of 'ProductOrders'. The central query editor contains the following SQL script:

```
USE ProductOrders
GO
CREATE VIEW Top130Orders
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
SELECT * FROM Top130Orders
```

The bottom pane shows the 'Results' tab with the following data:

	CustName	SumOfQuantity	FirstOrderDate
1	Karina Lacy	13	2019-12-04 00:00:00.000
2	Samuel Jacobsen	10	2019-11-21 00:00:00.000
3	Erick Kaleigh	7	2019-09-29 00:00:00.000
4	Korah Blanca	7	2019-06-23 00:00:00.000
5	Yash Randall	5	2020-05-09 00:00:00.000
6	Kyle Marissa	4	2020-07-16 00:00:00.000
7	Dakota Baylee	3	2020-10-08 00:00:00.000
8	Deborah Damien	2	2019-07-05 00:00:00.000
9	Anders Rohansen	2	2021-01-19 00:00:00.000
10	Johnathon Miletton	2	2019-10-24 00:00:00.000
11	Justin Javen	2	2020-04-18 00:00:00.000
12	Thalia Neftaly	2	2021-03-12 00:00:00.000
13	Trisha Anum	2	2020-02-24 00:00:00.000

The status bar at the bottom indicates 'Query executed successfully.' and '13 rows'.

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

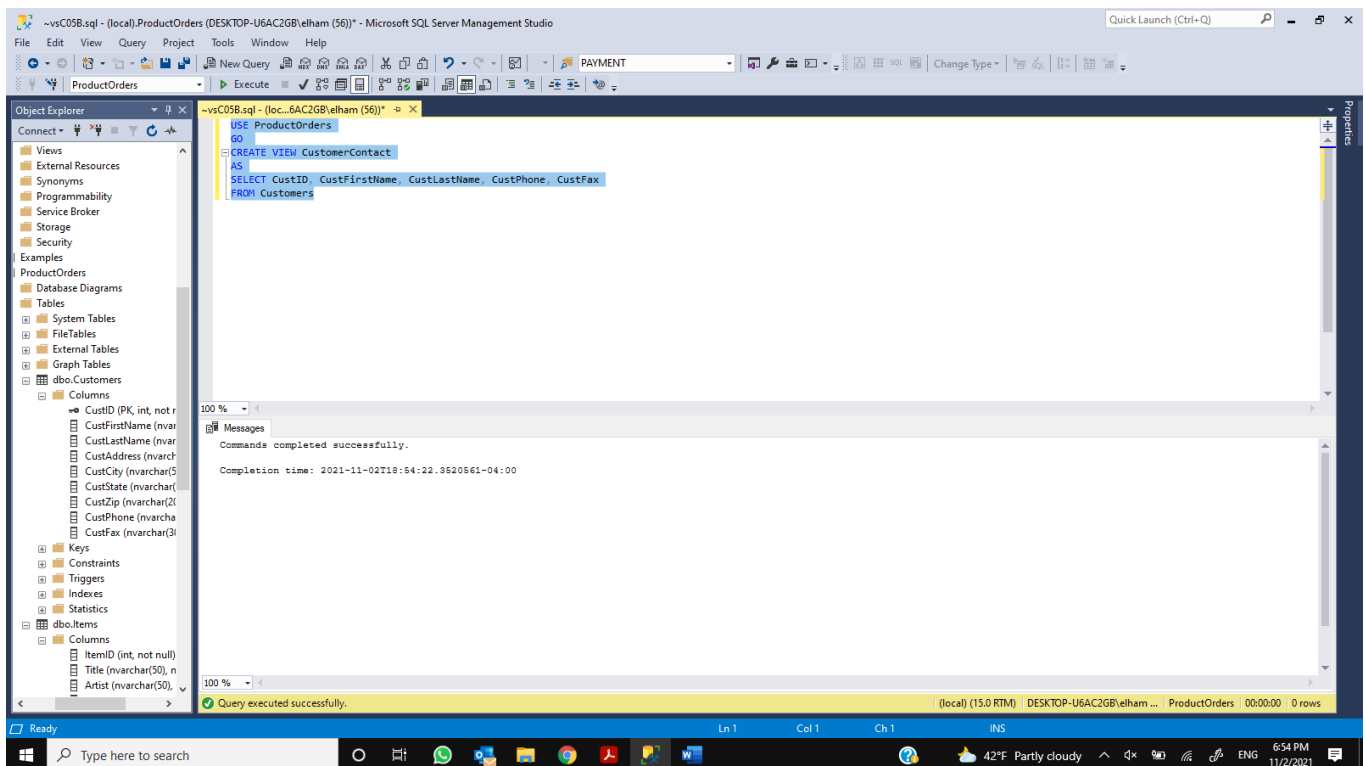
CREATE VIEW CustomerContact

AS

SELECT CustID, CustFirstName, CustLastName, CustPhone, CustFax

FROM Customers

Screen Shot:



A3.

USE ProductOrders

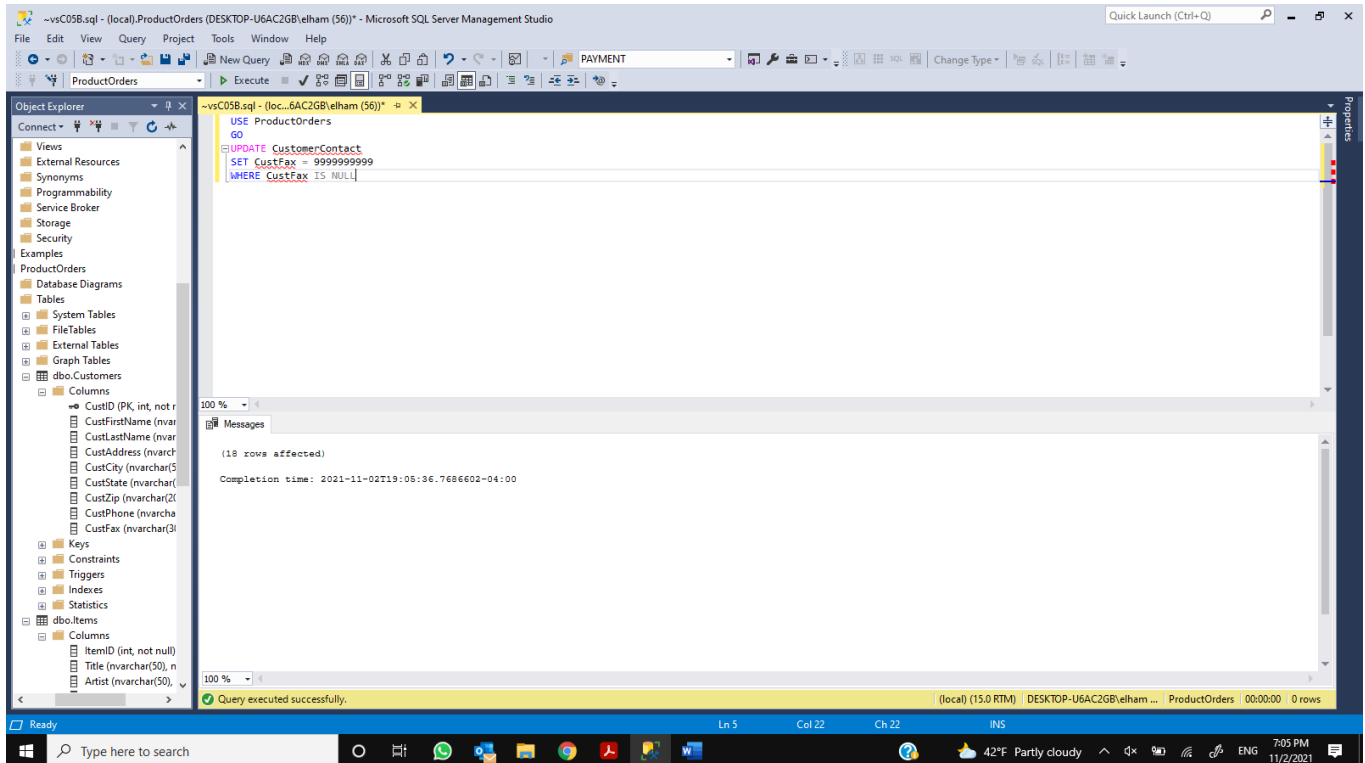
GO

UPDATE CustomerContact

SET CustFax = 9999999999

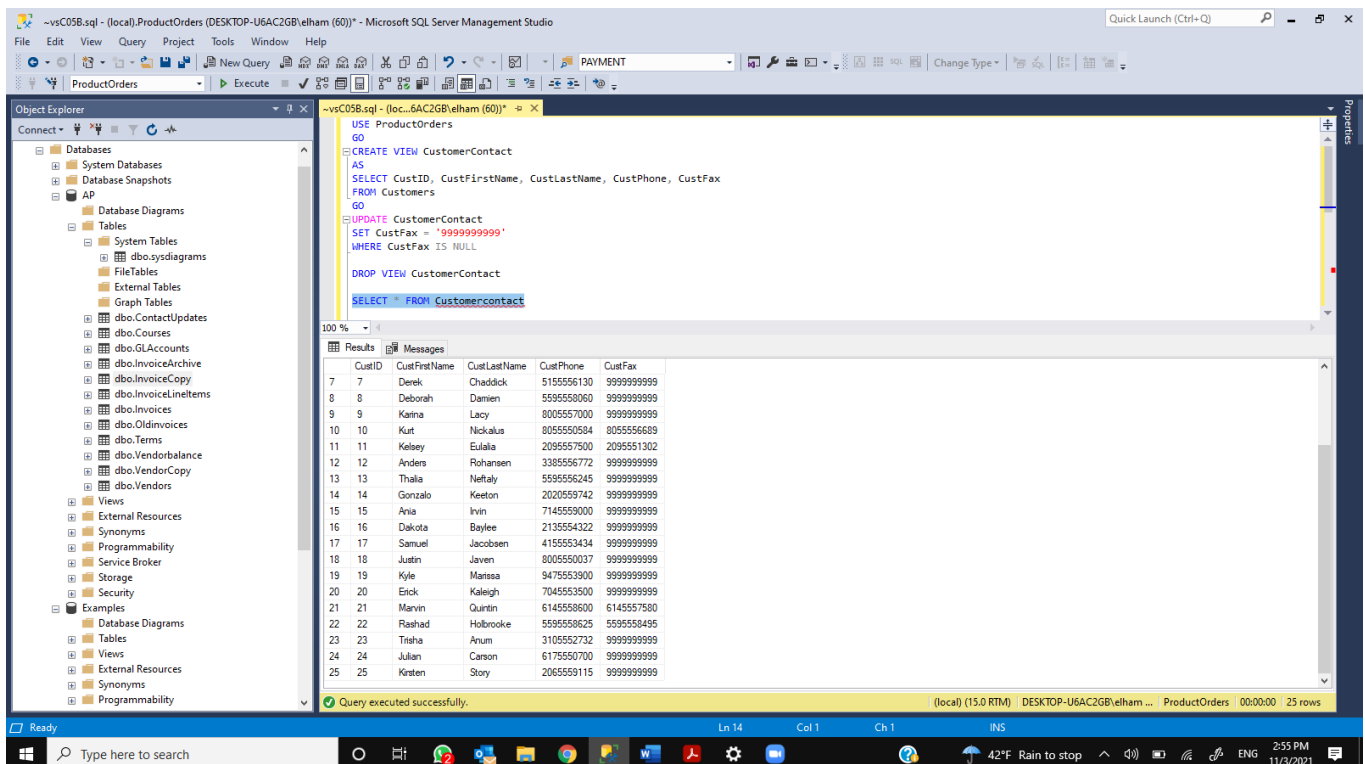
WHERE CustFax IS NULL

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:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the 'ProductOrders' database. The central query window shows the following SQL code:

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

The Results pane at the bottom displays the output of the query, showing a single row for the primary key constraint 'PK_Customers'.

name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published	is_schema_published	un
PK_Customers	597577167	NULL	1	581577110	PK	PRIMARY_KEY_CONSTRAINT	2021-11-03 15:04:09.700	2021-11-03 15:04:09.700	0	0	0	1

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

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:

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the database structure of 'AP' (AdventureWorks2008). The central pane shows a T-SQL script in the 'SQLQuery1.sql' window. The script declares a variable @TotalBalanceDue of type money, calculates its value from the InvoiceCopy table, and then returns a list of invoices with their vendor names, invoice numbers, due dates, and balance due, sorted by due date descending. If the total balance due is less than \$50,000.00, it returns the list; otherwise, it prints a message. The bottom pane shows the 'Results' tab with 11 rows of data.

```

USE AP
GO
DECLARE @TotalBalanceDue money;
SELECT @TotalBalanceDue = SUM(InvoiceTotal - PaymentTotal - CreditTotal)
FROM InvoiceCopy
WHERE (InvoiceTotal - PaymentTotal - CreditTotal) > 0;
IF @TotalBalanceDue < 50000
BEGIN
    PRINT 'Balance Due Is' + CAST(@TotalBalanceDue AS nvarchar(10));
    SELECT VendorName, InvoiceNumber, InvoiceDueDate, InvoiceTotal - PaymentTotal - CreditTotal AS 'Balance Due'
    FROM InvoiceCopy I JOIN Vendors V
    ON I.VendorID = V.VendorID
    WHERE (InvoiceTotal - PaymentTotal - CreditTotal) > 0
    ORDER BY InvoiceDueDate DESC;
END
ELSE
    PRINT 'Balance due is more than $50,000.00';

```

VendorName	InvoiceNumber	InvoiceDueDate	Balance Due
Malloy Lithographing Inc	0-2436	2021-04-30 00:00:00	10976.06
Blue Cross	547480102	2021-04-30 00:00:00	224.00
Ford Motor Credit Company	9982771	2021-04-23 00:00:00	503.20
Malloy Lithographing Inc	P-0608	2021-04-22 00:00:00	19351.18
Federal Express Corporation	263253270	2021-04-21 00:00:00	67.92
Federal Express Corporation	263253273	2021-04-21 00:00:00	30.75
Federal Express Corporation	263253268	2021-04-20 00:00:00	59.97
Federal Express Corporation	963253264	2021-04-17 00:00:00	52.25
Cardinal Business Media, Inc.	134116	2021-04-17 00:00:00	90.36
Ingram	31361833	2021-04-10 00:00:00	579.42
Data Reproductions Corp	39104	2021-04-09 00:00:00	85.31

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 than \$50,000.

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;

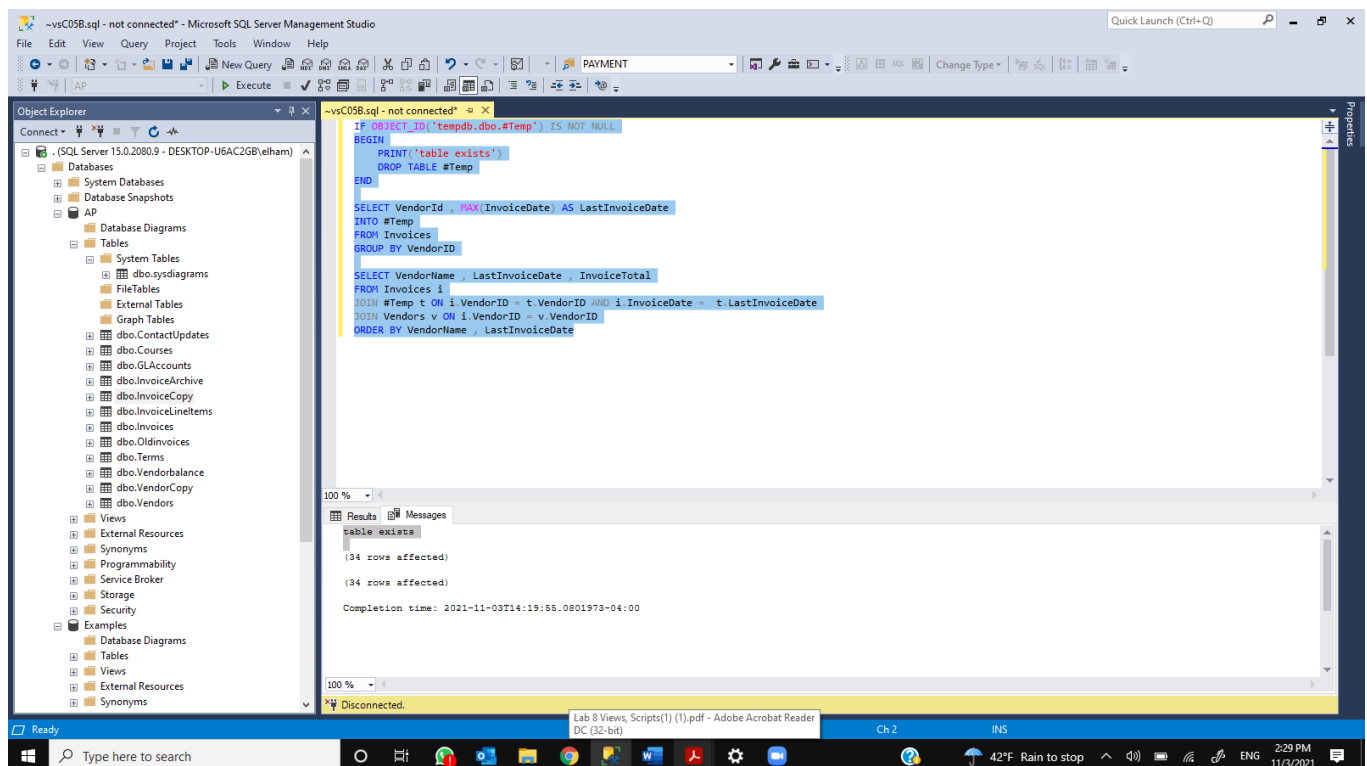
```

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;
```

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'Invoices' table expanded, showing its columns: InvoiceID (PK, int, not null), VendorID (FK, int, not null), InvoiceNumber (varchar(50), not null), InvoiceDate (smalldatetime, not null), InvoiceTotal (money, not null), PaymentTotal (money, not null), CreditTotal (money, not null), TermsID (FK, int, not null), InvoiceDueDate (smalldatetime, not null), and PaymentDate (smalldatetime, not null). The right pane shows the 'SQL Query 1.sql' script, which includes the following code:

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

GO

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

The bottom pane shows the 'Results' tab with the following data:

VendorID	LastInvoiceDate
34	2021-02-09 00:00:00
37	2021-04-01 00:00:00
48	2021-01-03 00:00:00
72	2021-03-10 00:00:00
80	2021-03-28 00:00:00
81	2021-02-21 00:00:00
82	2021-03-19 00:00:00
83	2021-03-21 00:00:00
86	2021-02-11 00:00:00
88	2021-01-15 00:00:00
89	2020-12-24 00:00:00
90	2021-03-20 00:00:00
94	2021-03-05 00:00:00
95	2021-03-15 00:00:00
96	2021-01-02 00:00:00

The status bar at the bottom indicates 'Query executed successfully.' and '34 rows'.

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;
```

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

```
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;
```

The Object Explorer on the left shows the database structure, including tables like `dbo.Invoices` and `dbo.Vendors`. The Results pane at the bottom shows the output of the query, which is a table with 4 columns: `InvoiceTotal`, `VendorName`, `LastInvoiceDate`, and `VendorID`. The results are sorted by `VendorName` and `LastInvoiceDate`.

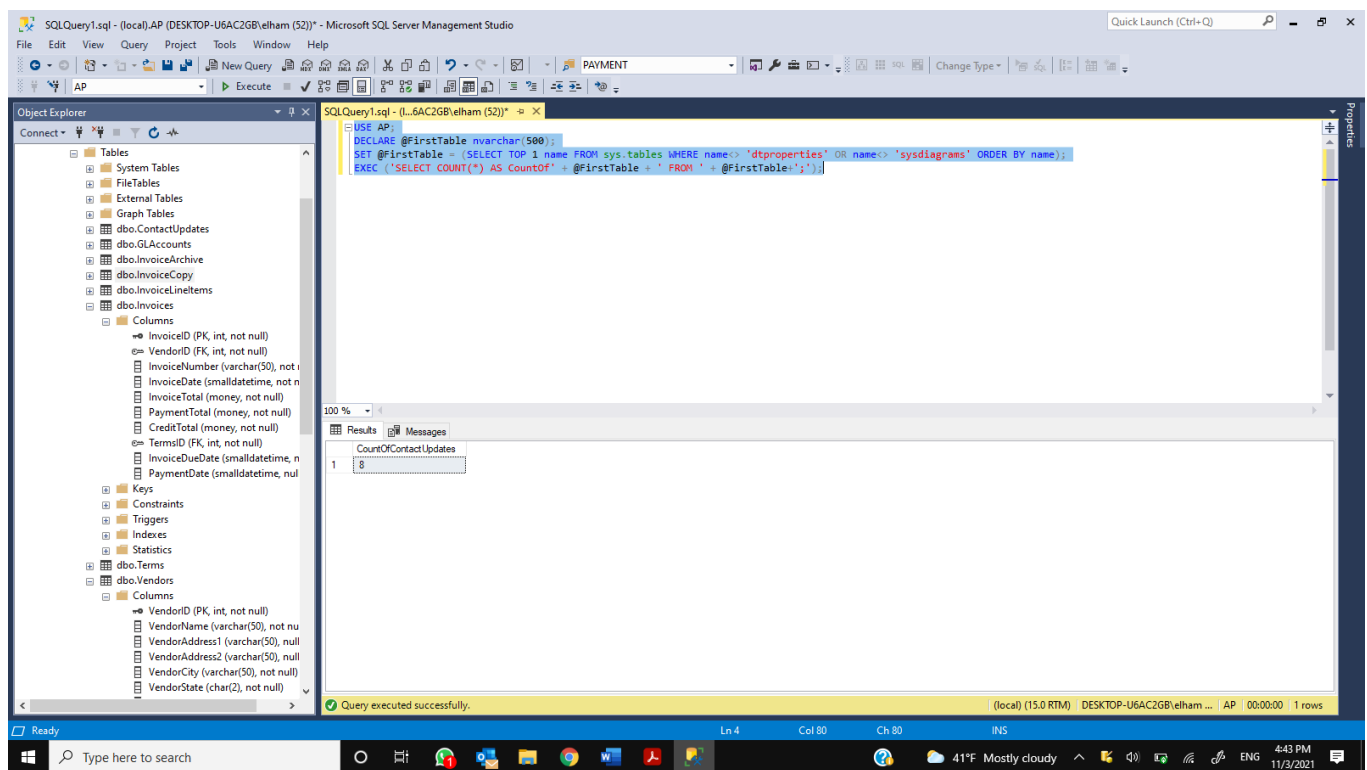
InvoiceTotal	VendorName	LastInvoiceDate	VendorID
17.50	Abbey Office Furnishings	2021-03-05 00:00:00	94
6940.25	Betelmann Industry Svcs. Inc	2021-02-18 00:00:00	99
224.00	Blue Cross	2021-04-01 00:00:00	37
2184.50	Cahners Publishing Company	2021-02-28 00:00:00	100
90.36	Cardinal Business Media, Inc.	2021-03-28 00:00:00	80
41.80	Coffee Break Service	2021-02-24 00:00:00	102
9.95	Compuserve	2021-01-13 00:00:00	97
2433.00	Computerworld	2021-02-11 00:00:00	86
85.31	Data Reproductions Corp	2021-03-10 00:00:00	72
1367.50	Dean Witter Reynolds	2021-02-11 00:00:00	103
7125.34	Digital Dreamworks	2021-01-21 00:00:00	104
220.00	Dintaa Groom & McCormick	2021-01-23 00:00:00	105
207.78	Edward Data Services	2021-01-15 00:00:00	88
95.00	Evans Executive Inc	2020-12-24 00:00:00	89
177.78	Edwards Business Publications	2021-04-01 00:00:00	109

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)