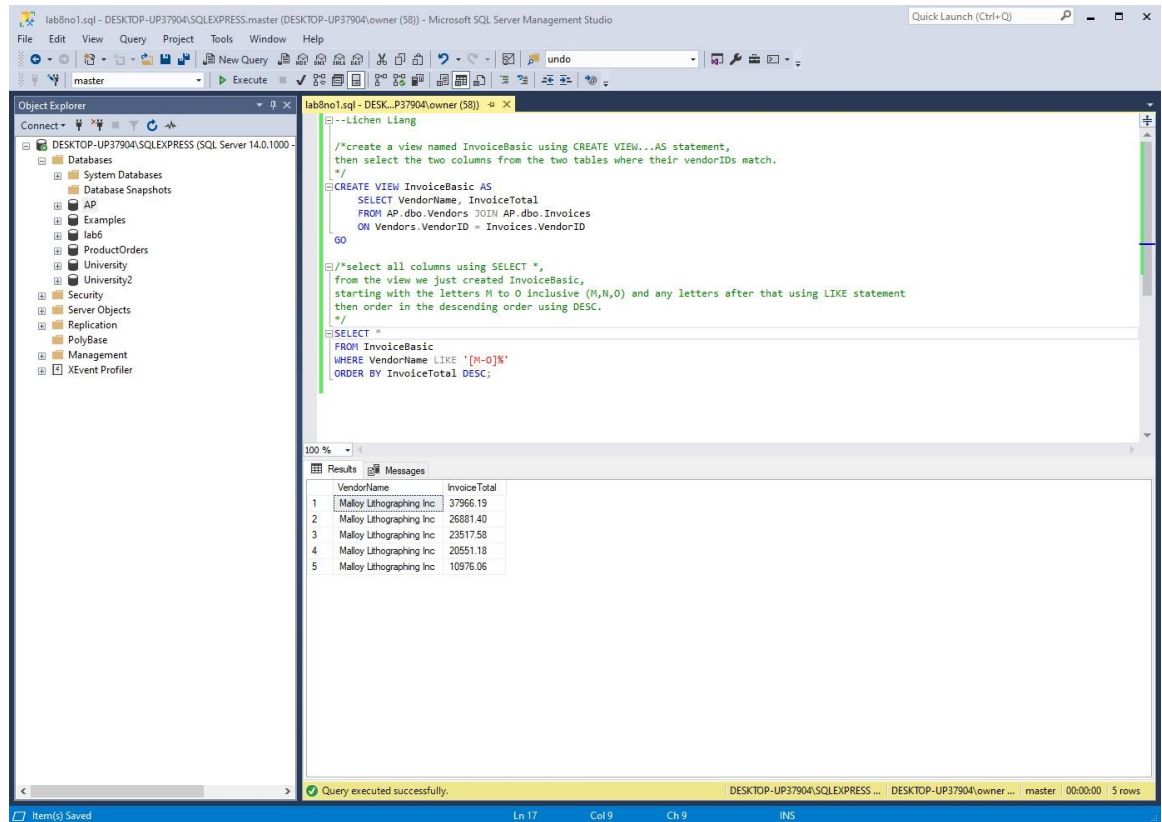


```
1. CREATE VIEW InvoiceBasic AS
    SELECT VendorName, InvoiceTotal
    FROM AP.dbo.Vendors JOIN AP.dbo.Invoices
    ON Vendors.VendorID = Invoices.VendorID

GO

SELECT *
FROM InvoiceBasic
WHERE VendorName LIKE '[M-O]%'
ORDER BY InvoiceTotal DESC;
```



The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'AP' database selected. The right pane shows the 'Query Editor' with the following SQL script:

```
--Lichen Liang
/*create a view named InvoiceBasic using CREATE VIEW...AS statement,
then select the two columns from the two tables where their vendorIDs match.
*/
CREATE VIEW InvoiceBasic AS
    SELECT VendorName, InvoiceTotal
    FROM AP.dbo.Vendors JOIN AP.dbo.Invoices
    ON Vendors.VendorID = Invoices.VendorID
GO

/*select all columns using SELECT *,
from the view we just created InvoiceBasic,
starting with the letters M to O inclusive (M,N,O) and any letters after that using LIKE statement
then order in the descending order using DESC.
*/
SELECT *
FROM InvoiceBasic
WHERE VendorName LIKE '[M-O]%'
ORDER BY InvoiceTotal DESC;
```

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

VendorName	InvoiceTotal
Malloy Lithographing Inc.	37966.19
Malloy Lithographing Inc.	26881.40
Malloy Lithographing Inc.	23517.58
Malloy Lithographing Inc.	20551.18
Malloy Lithographing Inc.	10976.06

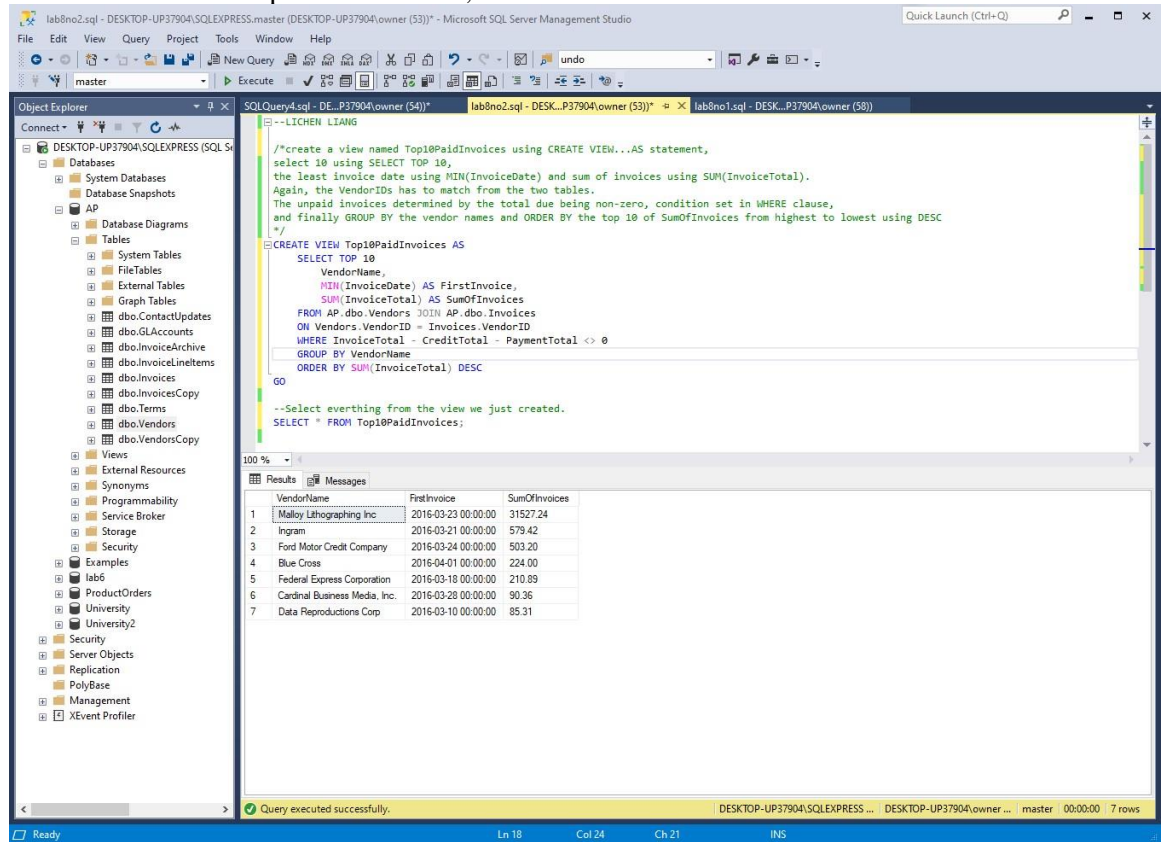
The status bar at the bottom indicates 'Query executed successfully.' and '5 rows'.

```

2. CREATE VIEW Top10PaidInvoices AS
    SELECT TOP 10
        VendorName,
        MIN(InvoiceDate) AS FirstInvoice,
        SUM(InvoiceTotal) AS SumOfInvoices
    FROM AP.dbo.Vendors JOIN AP.dbo.Invoices
    ON Vendors.VendorID = Invoices.VendorID
    WHERE InvoiceTotal - CreditTotal - PaymentTotal <> 0
    GROUP BY VendorName
    ORDER BY SUM(InvoiceTotal) DESC
GO

```

```
SELECT * FROM Top10PaidInvoices;
```



The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'AP' database selected. The right pane shows the 'SQL Query Editor' with the following T-SQL script:

```

/*create a view named Top10PaidInvoices using CREATE VIEW...AS statement,
select 10 using SELECT TOP 10,
the least invoice date using MIN(InvoiceDate) and sum of invoices using SUM(InvoiceTotal).
Again, the VendorIDs has to match from the two tables.
The unpaid invoices determined by the total due being non-zero, condition set in WHERE clause,
and finally GROUP BY the vendor names and ORDER BY the top 10 of SumOfInvoices from highest to lowest using DESC
*/
CREATE VIEW Top10PaidInvoices AS
    SELECT TOP 10
        VendorName,
        MIN(InvoiceDate) AS FirstInvoice,
        SUM(InvoiceTotal) AS SumOfInvoices
    FROM AP.dbo.Vendors JOIN AP.dbo.Invoices
    ON Vendors.VendorID = Invoices.VendorID
    WHERE InvoiceTotal - CreditTotal - PaymentTotal <> 0
    GROUP BY VendorName
    ORDER BY SUM(InvoiceTotal) DESC
GO

--Select everthing from the view we just created.
SELECT * FROM Top10PaidInvoices;

```

The 'Results' pane at the bottom shows the output of the query, displaying 7 rows of data:

VendorName	FirstInvoice	SumOfInvoices
Malloy Lithographing Inc	2016-03-23 00:00:00	31527.24
Ingram	2016-03-21 00:00:00	579.42
Ford Motor Credit Company	2016-03-24 00:00:00	503.20
Blue Cross	2016-04-01 00:00:00	224.00
Federal Express Corporation	2016-03-18 00:00:00	210.89
Cardinal Business Media, Inc.	2016-03-28 00:00:00	90.36
Data Reproductions Corp	2016-03-10 00:00:00	85.31

The status bar at the bottom indicates 'Query executed successfully.' and '7 rows'.

There are only 7 such vendors.

```

3. CREATE VIEW VendorAddress AS
    SELECT VendorID,
    (ISNULL(VendorAddress1,'')+ ' ' + ISNULL(VendorAddress2,'')+ ',' + VendorCity +
    ' ' + VendorState + ', ' + VendorZipCode) AS Address
    FROM AP.dbo.Vendors

```

GO

```

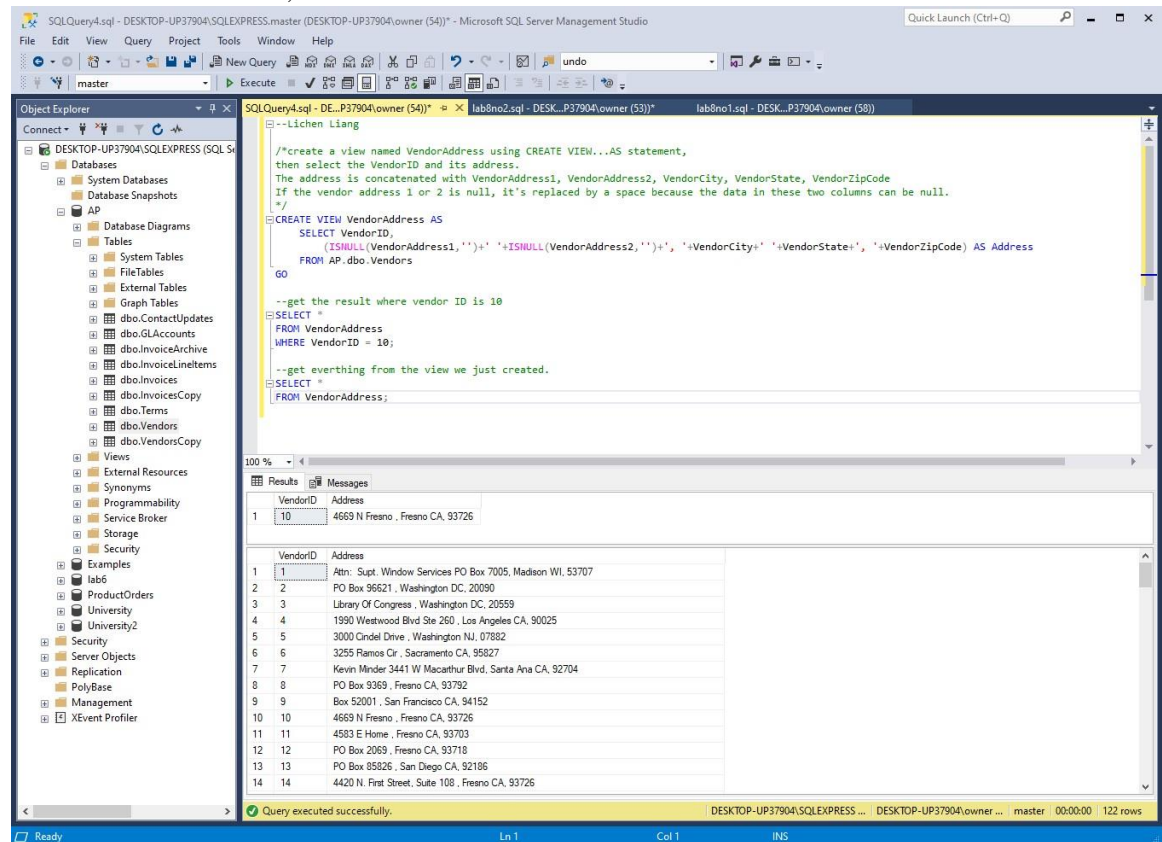
SELECT *
FROM VendorAddress
WHERE VendorID = 10;

```

```

SELECT *
FROM VendorAddress;

```



In this case, (before you announced the changes to the question) there is a vendor who has both address 1 and 2 are NULL. If you right click Vendors table and select Design, you can see that both Address 1 and 2 allow NULL values, so whenever address 2 is NULL replace with space, I also did the same with address 1. Therefore, I did not rewrite the code. Hope you can understand my point here.

4. USE Examples;

SELECT * FROM sys.foreign_keys;

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'Examples' database selected. The right pane shows the 'Query Editor' with the following SQL query:

```
--USE Examples;  
--select the foreign keys in the EXAMPLES database, using the SQL catalog views: sys.foreign_keys  
SELECT * FROM sys.foreign_keys;
```

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

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published	is_collated
1	FK_SalesTotals_SalesReps	1445580188	NULL	1	1221579390	F	FOREIGN_KEY_CONSTRAINT	2020-01-19 20:11:52.880	2020-01-19 20:11:52.880	0	0	0

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

One foreign key defined called RepID that relates SalesReps table(PK) -> SalesTotals table(FK).

5. USE AP;

```
DECLARE @TotalBalanceDue money;

SET @TotalBalanceDue =
    (SELECT SUM(InvoiceTotal - CreditTotal - PaymentTotal)
     FROM Invoices
     WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0);

IF @TotalBalanceDue > 40000
    PRINT 'Balance due is more than $40,000.00'
ELSE
    BEGIN
        PRINT 'Balance due is $'+ CAST(@TotalBalanceDue AS varchar(10))
        SELECT      VendorName,
                     InvoiceNumber,
                     InvoiceDueDate,
                     (InvoiceTotal - CreditTotal - PaymentTotal) AS Balance
        FROM Vendors JOIN Invoices
            ON Vendors.VendorID = Invoices.VendorID
        WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0
        ORDER BY InvoiceDueDate ASC
    END;
```

lab8no3.sql - DESKTOP-UP37904\SQLEXPRESS.AP (DESKTOP-UP37904\owner (56)) - Microsoft SQL Server Management Studio

Object Explorer: DESKTOP-UP37904\SQLEXPRESS (SQL Server 14.0.10) > Databases > AP > Tables > dbo.Invoices

```
1 --Lichen Liang
2
3 USE AP;
4
5 --set datatype of TotalBalanceDue to money, using DECLARE
6 DECLARE @TotalBalanceDue money;
7
8 --set the TotalBalanceDue to be the sum of balance due, using SET.
9 --then use subquery to find the sum of balance due where balance due is greater than 0
10 SET @TotalBalanceDue =
11 (SELECT SUM(InvoiceTotal - CreditTotal - PaymentTotal)
12 FROM Invoices
13 WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0);
14
15 /*if total balance due is more than 40000, print the message using PRINT.
16 Otherwise(total balance due below 40000), use BEGIN clause to start
17 print the message concatenated with total balance due, which is casted into varchar datatype.
18 Also return the four columns VendorName, InvoiceNumber, InvoiceDueDate, and Balance,
19 where balance should be greater than 0 and the vendor IDs should match from the two tables.
20 Order by the oldest due date first(in ASC).
21 and END clause to stop
22 */
23 IF @TotalBalanceDue > 40000
24 PRINT 'Balance due is more than $40,000.00'
25 ELSE
26 BEGIN
27 PRINT 'Balance due is $'+ CAST(@TotalBalanceDue AS varchar(10))
28 SELECT
29 VendorName,
30 InvoiceNumber,
31 InvoiceDueDate,
32 (InvoiceTotal - CreditTotal - PaymentTotal) AS Balance
33 FROM Vendors JOIN Invoices
34 ON Vendors.VendorID = Invoices.VendorID
35 WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0
36 ORDER BY InvoiceDueDate ASC
37 END;
```

Results: 100% | Messages

VendorName	InvoiceNumber	InvoiceDueDate	Balance
Data Reproductions Corp	39104	2016-04-09 00:00:00	85.31
Ingram	31361833	2016-04-10 00:00:00	579.42
Federal Express Corporation	963253264	2016-04-17 00:00:00	52.25
Cardinal Business Media, Inc.	134116	2016-04-17 00:00:00	90.36
Federal Express Corporation	263253268	2016-04-20 00:00:00	59.97
Federal Express Corporation	263253270	2016-04-21 00:00:00	67.92
Federal Express Corporation	263253273	2016-04-21 00:00:00	30.75
Malloy Lithographing Inc	P-0508	2016-04-22 00:00:00	19351.18
Ford Motor Credit Company	9982771	2016-04-23 00:00:00	503.20
Malloy Lithographing Inc	Q-2436	2016-04-30 00:00:00	10978.06

Query executed successfully. DESKTOP-UP37904\SQLEXPRESS ... DESKTOP-UP37904\owner ... AP 00:00:00 11 rows

lab8no3.sql - DESKTOP-UP37904\SQLEXPRESS.AP (DESKTOP-UP37904\owner (56)) - Microsoft SQL Server Management Studio

Object Explorer: DESKTOP-UP37904\SQLEXPRESS (SQL Server 14.0.10) > Databases > AP > Tables > dbo.Invoices

```
1 --Lichen Liang
2
3 USE AP;
4
5 --set datatype of TotalBalanceDue to money, using DECLARE
6 DECLARE @TotalBalanceDue money;
7
8 --set the TotalBalanceDue to be the sum of balance due, using SET.
9 --then use subquery to find the sum of balance due where balance due is greater than 0
10 SET @TotalBalanceDue =
11 (SELECT SUM(InvoiceTotal - CreditTotal - PaymentTotal)
12 FROM Invoices
13 WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0);
14
15 /*if total balance due is more than 40000, print the message using PRINT.
16 Otherwise(total balance due below 40000), use BEGIN clause to start
17 print the message concatenated with total balance due, which is casted into varchar datatype.
18 Also return the four columns VendorName, InvoiceNumber, InvoiceDueDate, and Balance,
19 where balance should be greater than 0 and the vendor IDs should match from the two tables.
20 Order by the oldest due date first(in ASC).
21 and END clause to stop
22 */
23 IF @TotalBalanceDue > 40000
24 PRINT 'Balance due is more than $40,000.00'
25 ELSE
26 BEGIN
27 PRINT 'Balance due is $'+ CAST(@TotalBalanceDue AS varchar(10))
28 SELECT
29 VendorName,
30 InvoiceNumber,
31 InvoiceDueDate,
32 (InvoiceTotal - CreditTotal - PaymentTotal) AS Balance
33 FROM Vendors JOIN Invoices
34 ON Vendors.VendorID = Invoices.VendorID
35 WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0
36 ORDER BY InvoiceDueDate ASC
37 END;
```

Results: 100% | Messages

Balance due is \$32020.42

(11 rows affected)

Completion time: 2020-03-16T03:29:33.2399570-04:00

Query executed successfully. DESKTOP-UP37904\SQLEXPRESS ... DESKTOP-UP37904\owner ... AP 00:00:00 11 rows

Datatype for @TotalBalanceDue is money

6. USE AP;

```
IF OBJECT_ID('tempdb..#FirstInvoice') IS NOT NULL
DROP TABLE #FirstInvoice;
```

```
SELECT VendorID, MIN(InvoiceDate) AS FirstInvoiceDate
INTO #FirstInvoice
FROM Invoices
GROUP BY VendorID;
```

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

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor displays a T-SQL script that creates a temporary table #FirstInvoice and then joins it with the Invoices and Vendors tables. The script is as follows:

```
1 --Lichen Liang
2
3 /*In AP database,
4 return three columns VendorName, FirstInvoiceDate, and InvoiceTotal from three tables:
5 Invoices, Vendors, and a temporary table from the subquery FirstInvoice.
6 The FirstInvoice table contains VendorID and its oldest invoice date called FirstInvoiceDate.
7 VendorID and InvoiceDate from Invoices table should match VendorID and FirstInvoiceDate from FirstInvoice table.
8 VendorID between Invoices and Vendors should also match.
9 Finally, order by the vendor name and their oldest invoice date.
10 */
11
12 USE AP;
13
14 --checks if the temporary table exists, if so, delete it so it can be recreated again
15 IF OBJECT_ID('tempdb..#FirstInvoice') IS NOT NULL
16 DROP TABLE #FirstInvoice;
17
18 /*use SELECT...INTO to create the temporary table FirstInvoice. # denotes a temp table.
19 Then select the three columns we want to return,
20 Join the three tables, including the temporary table, and the rest are the same.
21 */
22 SELECT VendorID, MIN(InvoiceDate) AS FirstInvoiceDate
23 INTO #FirstInvoice
24 FROM Invoices
25 GROUP BY VendorID;
26
27 SELECT VendorName, FirstInvoiceDate, InvoiceTotal
28 FROM Invoices JOIN #FirstInvoice
29 ON ((Invoices.VendorID = #FirstInvoice.VendorID) AND (Invoices.InvoiceDate = #FirstInvoice.FirstInvoiceDate))
30 JOIN Vendors
31 ON Invoices.VendorID = Vendors.VendorID
32 ORDER BY VendorName, FirstInvoiceDate;
```

The Results pane shows the output of the query, displaying a list of vendors with their first invoice date and total invoice amount. The data is as follows:

VendorName	FirstInvoiceDate	InvoiceTotal
Alley Office Furnishings	2016-03-09 00:00:00	17.50
Bertelmann Industry Svcs. Inc.	2016-02-18 00:00:00	6940.25
Blue Cross	2016-02-03 00:00:00	224.00
Cahners Publishing Company	2016-02-28 00:00:00	2184.50
Cardinal Business Media, Inc.	2016-02-22 00:00:00	175.00
Coffee Break Service	2016-02-24 00:00:00	41.80
Compuserve	2016-01-03 00:00:00	9.95
Computersworld	2016-02-11 00:00:00	2433.00
Data Reproductions Corp	2016-02-01 00:00:00	21842.00
Dean Vitter Reynolds	2016-02-11 00:00:00	1367.50
Digital Dreamworks	2016-01-21 00:00:00	7125.34
Distas Groom & McComick	2016-01-23 00:00:00	220.00
Edward Data Services	2016-01-15 00:00:00	207.78

The status bar at the bottom indicates that the query was executed successfully, returning 34 rows.

Explanation see comment in screenshot.

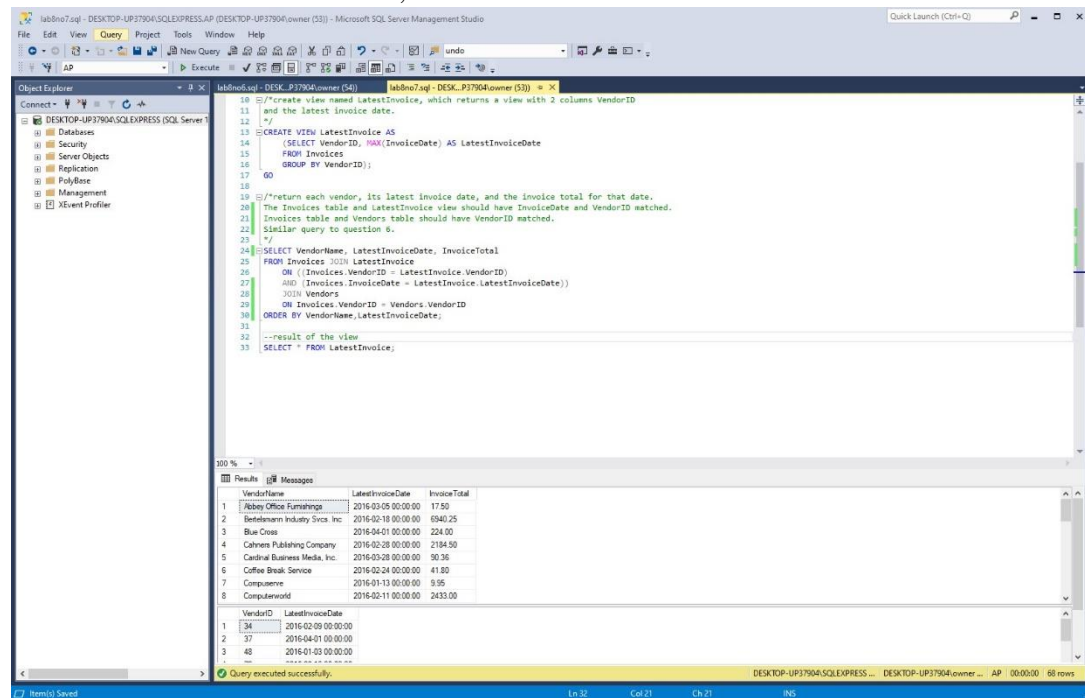
7. USE AP;

```
IF OBJECT_ID('LatestInvoice') IS NOT NULL
    DROP VIEW LatestInvoice;
GO
```

```
CREATE VIEW LatestInvoice AS
    (SELECT VendorID, MAX(InvoiceDate) AS LatestInvoiceDate
     FROM Invoices
     GROUP BY VendorID);
GO
```

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

```
SELECT * FROM LatestInvoice;
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

```
10 --create view named LatestInvoice, which returns a view with 2 columns VendorID
11 --and the latest invoice date.
12 --
13 CREATE VIEW LatestInvoice AS
14     (SELECT VendorID, MAX(InvoiceDate) AS LatestInvoiceDate
15      FROM Invoices
16      GROUP BY VendorID);
17 GO
18
19 --return each vendor, its latest invoice date, and the invoice total for that date.
20 --The Invoices table and LatestInvoice view should have InvoiceDate and VendorID matched.
21 --Invoices table and Vendors table should have VendorID matched.
22 --Similar query to question 6.
23 --
24 SELECT VendorName, LatestInvoiceDate, InvoiceTotal
25 FROM Invoices JOIN LatestInvoice
26     ON ((Invoices.VendorID = LatestInvoice.VendorID)
27     AND (Invoices.InvoiceDate = LatestInvoice.LatestInvoiceDate))
28 JOIN Vendors
29     ON Invoices.VendorID = Vendors.VendorID
30 ORDER BY VendorName, LatestInvoiceDate;
31
32 --result of the view
33 SELECT * FROM LatestInvoice;
```

The results window displays the output of the query, showing a list of vendors and their latest invoice details:

VendorName	LatestInvoiceDate	InvoiceTotal
1 Abbey Office Furnishings	2016-03-05 00:00:00	17.50
2 Bethlehem Industry Svcs. Inc.	2016-02-18 00:00:00	6940.25
3 Blue Cross	2016-04-01 00:00:00	224.00
4 Calhoun Publishing Company	2016-02-28 00:00:00	2184.50
5 Central Business Media, Inc.	2016-03-28 00:00:00	50.36
6 Coffee Break Service	2016-01-24 00:00:00	41.80
7 Compuserve	2016-01-13 00:00:00	9.95
8 Computerworld	2016-02-11 00:00:00	2433.00

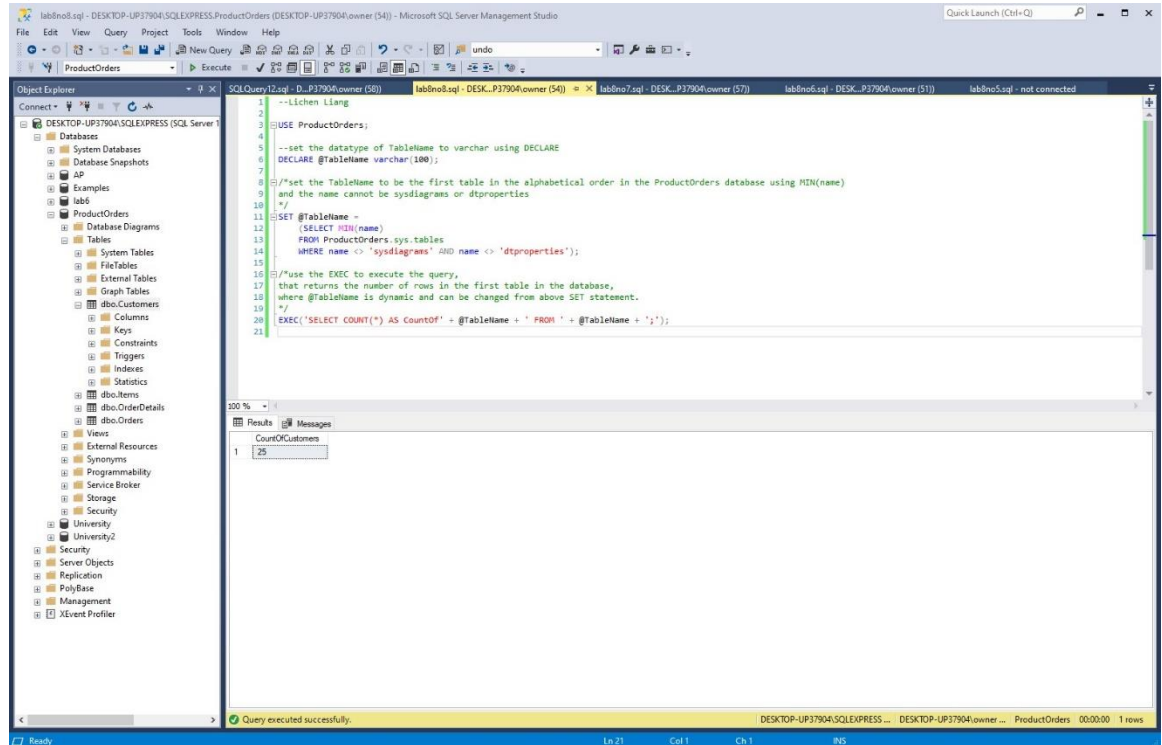
The status bar at the bottom indicates that the query was executed successfully, returning 58 rows.

8. USE ProductOrders;

```
DECLARE @TableName varchar(100);
```

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

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



Remarks

In this lab we practiced how to use views and write scripts. Also practiced using DDL, catalog views, statements for controlling the flow of execution, etc. I think this is a very good practice for the lecture. A more complicated challenge would be increasing the requirement in the question, such as more tables, columns, conditions, and so on.