Project I

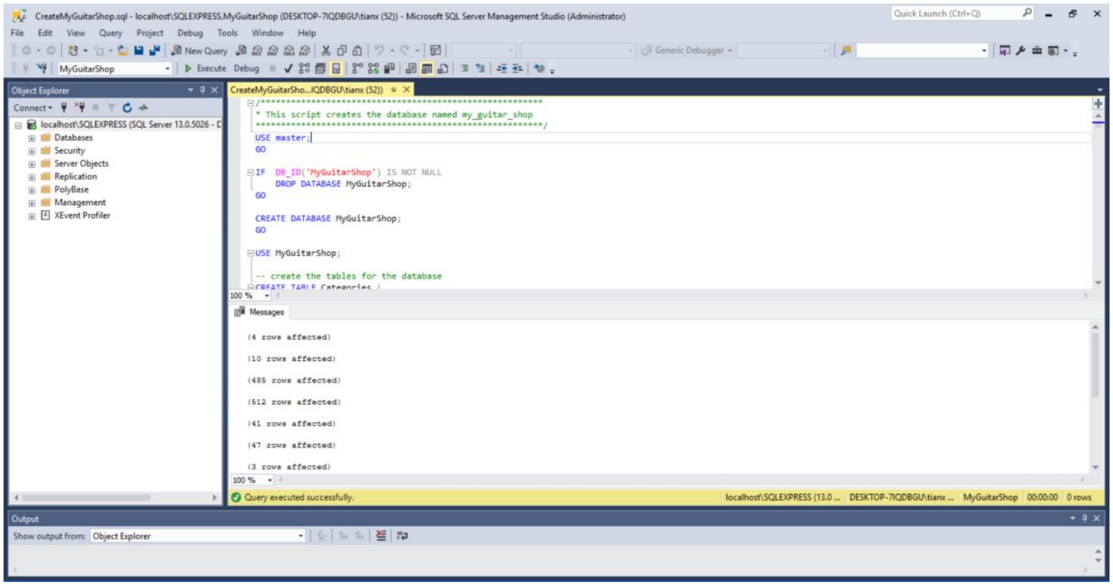
TIANXIANG YI

I.My Guitar Shop Database

A. Database Setup 2’

1.

(1) Download CreateMyGuitarShop.sql from Project 1 directory on Blackboard and open it in SQL server management studio. Execute the entire script and show the message in the Message tab, indicating the script is executed successfully. A complete screenshot of execution result is required.


CreateMyGuitarShop.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (52)) - Microsoft SQL Server Management Studio (Administrator)
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
MyGuitarShop
Execute Debug
Object Explorer
Connect
localhost\SQLEXPRESS (SQL Server 13.0.5026 - D
Databases
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
CreateMyGuitarSho...IQDBGU\tianx (52))
100 %
Messages
Output
Show output from: Object Explorer

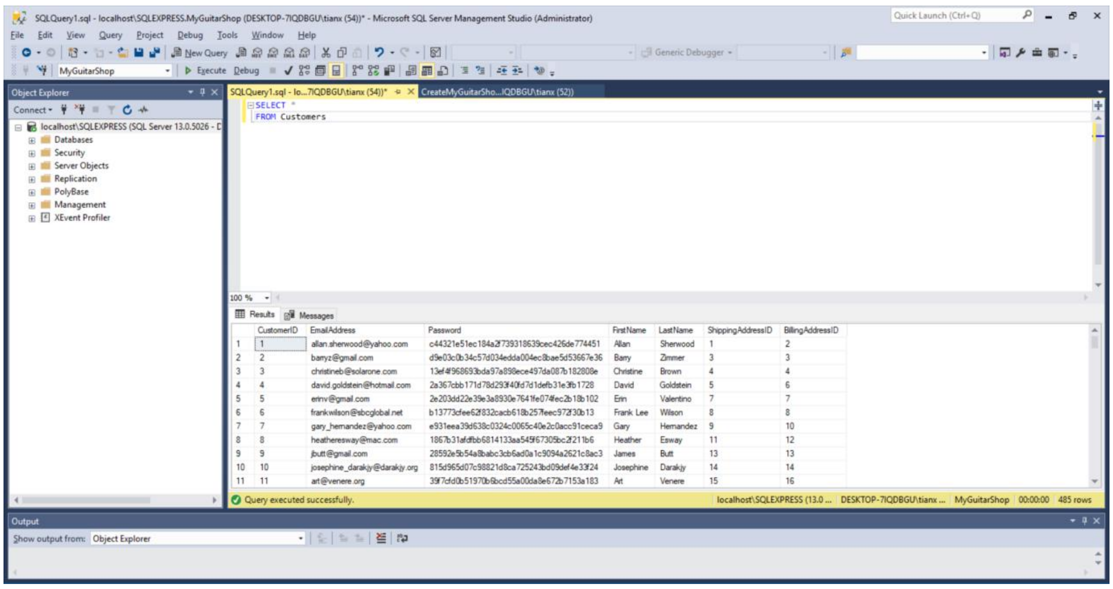
```
/*************************************************************
* This script creates the database named my_guitar_shop
*************************************************************/
USE master;
GO

IF  DB_ID('MyGuitarShop') IS NOT NULL
    DROP DATABASE MyGuitarShop;
GO

CREATE DATABASE MyGuitarShop;
GO

USE MyGuitarShop;

-- create the tables for the database
CREATE TABLE Categories (
```

```
(4 rows affected)

(10 rows affected)

(485 rows affected)

(512 rows affected)

(41 rows affected)

(47 rows affected)

(3 rows affected)
```

Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 0 rows

(2) Navigate through the database objects and view the column definitions for each table. Open a new Query Editor window. Show details in Customers table and Orders table using SELECT statement. Full screenshots of execution results are required.

Customers table



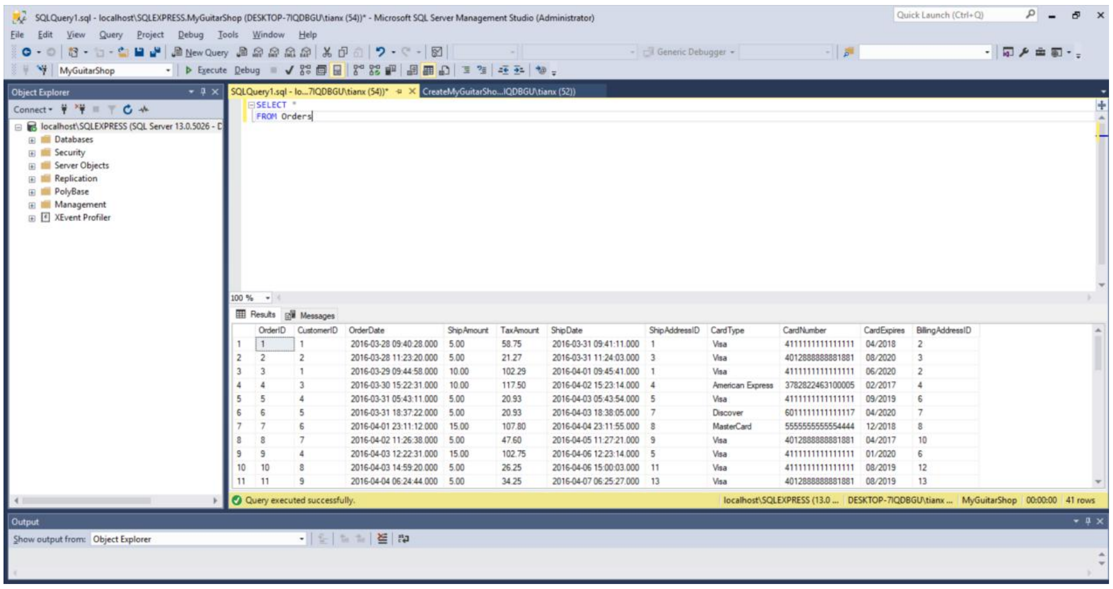
SQLQuery1.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (54))* - Microsoft SQL Server Management Studio (Administrator)

```
SELECT *
FROM Customers
```

	CustomerID	EmailAddress	Password	FirstName	LastName	ShippingAddressID	BillingAddressID
1	1	allan.sherwood@yahoo.com	c44321e51ec184a2f739318639cec426de774451	Allan	Sherwood	1	2
2	2	barryz@gmail.com	d9e03c0b34c57d034edda004ec8bae5d53667e36	Barry	Zimmer	3	3
3	3	christineb@solarone.com	13ef4f968693bda97a898ece497da087b182808e	Christine	Brown	4	4
4	4	david.goldstein@hotmail.com	2a367cbb171d78d293f40fd7d1defb31e3fb1728	David	Goldstein	5	6
5	5	erinv@gmail.com	2e203dd22e39e3a8930e7641fe074fec2b18b102	Erin	Valentino	7	7
6	6	frankwilson@sbcglobal.net	b13773cfee62f832cacb618b257feec972f30b13	Frank Lee	Wilson	8	8
7	7	gary_hernandez@yahoo.com	e931eea39d638c0324c0065c40e2c0acc91ceca9	Gary	Hernandez	9	10
8	8	heatheresway@mac.com	1867b31afdfbb6814133aa545f67305bc2f211b6	Heather	Esway	11	12
9	9	jbutt@gmail.com	28592e5b54a8babc3cb6ad0a1c9094a2621c8ac3	James	Butt	13	13
10	10	josephine_darakjy@darakjy.org	815d965d07c98821d8ca725243bd09def4e33f24	Josephine	Darakjy	14	14
11	11	art@venere.org	39f7cfd0b51970b6bcd55a00da8e672b7153a183	Art	Venere	15	16

Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 485 rows

Orders table



SQLQuery1.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (54))* - Microsoft SQL Server Management Studio (Administrator)

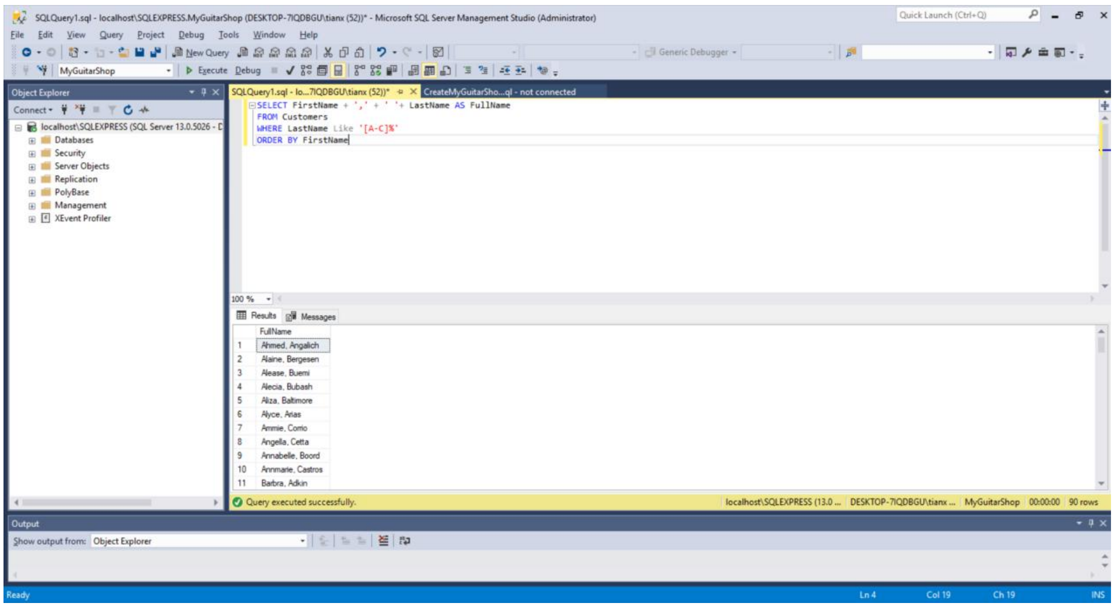
```
SELECT *
FROM Orders
```

	OrderID	CustomerID	OrderDate	ShipAmount	TaxAmount	ShipDate	ShipAddressID	CardType	CardNumber	CardExpires	BillingAddressID
1	1	1	2016-03-28 09:40:28.000	5.00	58.75	2016-03-31 09:41:11.000	1	Visa	4111111111111111	04/2018	2
2	2	2	2016-03-28 11:23:20.000	5.00	21.27	2016-03-31 11:24:03.000	3	Visa	4012888888881881	08/2020	3
3	3	1	2016-03-29 09:44:58.000	10.00	102.29	2016-04-01 09:45:41.000	1	Visa	4111111111111111	06/2020	2
4	4	3	2016-03-30 15:22:31.000	10.00	117.50	2016-04-02 15:23:14.000	4	American Express	378282246310005	02/2017	4
5	5	4	2016-03-31 05:43:11.000	5.00	20.93	2016-04-03 05:43:54.000	5	Visa	4111111111111111	09/2019	6
6	6	5	2016-03-31 18:37:22.000	5.00	20.93	2016-04-03 18:38:05.000	7	Discover	6011111111111117	04/2020	7
7	7	6	2016-04-01 23:11:12.000	15.00	107.80	2016-04-04 23:11:55.000	8	MasterCard	5555555555554444	12/2018	8
8	8	7	2016-04-02 11:26:38.000	5.00	47.60	2016-04-05 11:27:21.000	9	Visa	4012888888881881	04/2017	10
9	9	4	2016-04-03 12:22:31.000	15.00	102.75	2016-04-06 12:23:14.000	5	Visa	4111111111111111	01/2020	6
10	10	8	2016-04-03 14:59:20.000	5.00	26.25	2016-04-06 15:00:03.000	11	Visa	4111111111111111	08/2019	12
11	11	9	2016-04-04 06:24:44.000	5.00	34.25	2016-04-07 06:25:27.000	13	Visa	4012888888881881	08/2019	13

Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 41 rows

B. An Introduction to SQL 6'

1. [3] Write a SELECT statement that returns one column from the Customers table named FullName that joins the FirstName and LastName columns. Format this column with the first name, a comma, a space, and the last name like this: John, Doe Add an ORDER BY clause to this statement that sorts the result set by first name in ascending sequence. Return only the contacts whose last name begins with a letter from A to C.

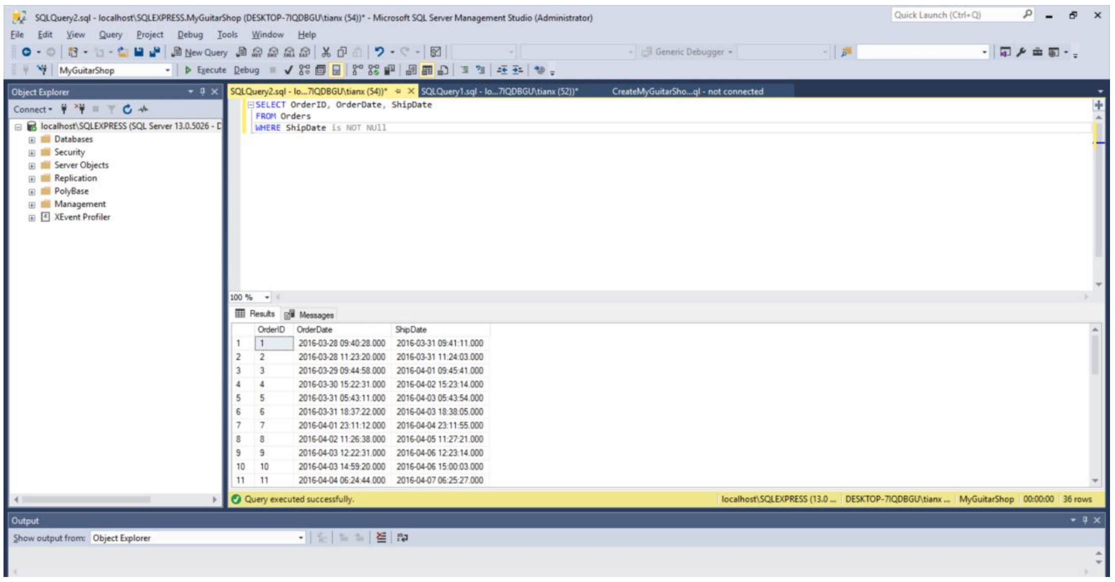


```
SELECT FirstName + ',' + ' '+ LastName AS FullName
FROM Customers
WHERE LastName Like '[A-C]%'
ORDER BY FirstName
```

```
     FullName
1    Ahmed, Angalich
2    Alaine, Bergesen
3    Alease, Buemi
4    Alecia, Bubash
5    Aliza, Baltimore
6    Alyce, Arias
7    Ammie, Corrio
8    Angella, Cetta
9    Annabelle, Boord
10   Annmarie, Castros
11   Barbra, Adkin
```

Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 90 rows

2. [3] Write a SELECT statement that returns these columns from the Orders table: OrderID The OrderID column OrderDate The OrderDate column ShipDate The ShipDate column Return only the rows where the ShipDate column does not contain a null value.



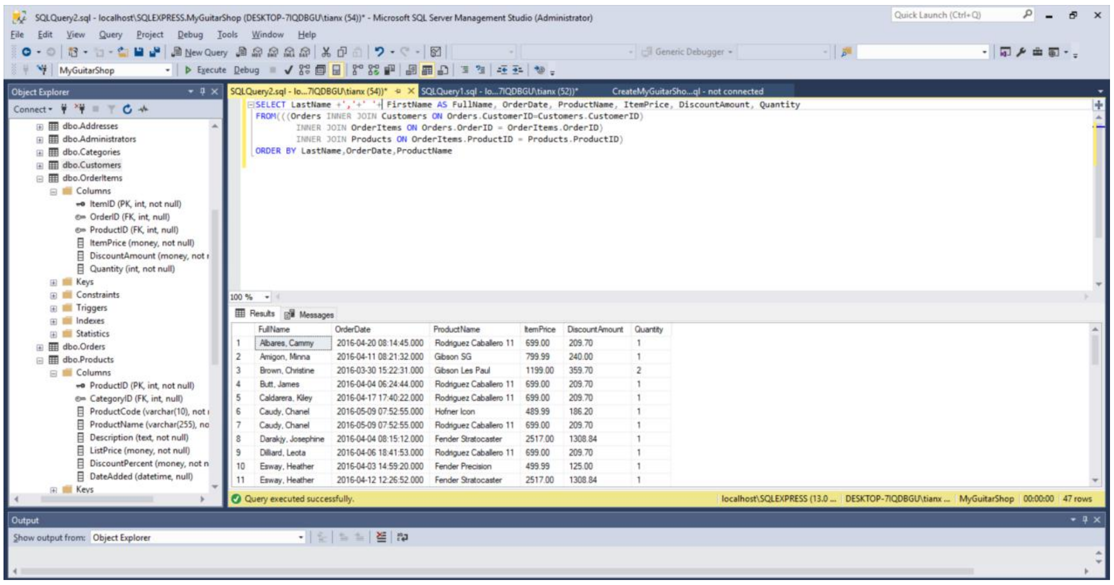
```
SELECT OrderID, OrderDate, ShipDate
FROM Orders
WHERE ShipDate is NOT NUll
```

```
     OrderID  OrderDate                 ShipDate
1    1        2016-03-28 09:40:28.000   2016-03-31 09:41:11.000
2    2        2016-03-28 11:23:20.000   2016-03-31 11:24:03.000
3    3        2016-03-29 09:44:58.000   2016-04-01 09:45:41.000
4    4        2016-03-30 15:22:31.000   2016-04-02 15:23:14.000
5    5        2016-03-31 05:43:11.000   2016-04-03 05:43:54.000
6    6        2016-03-31 18:37:22.000   2016-04-03 18:38:05.000
7    7        2016-04-01 23:11:12.000   2016-04-04 23:11:55.000
8    8        2016-04-02 11:26:38.000   2016-04-05 11:27:21.000
9    9        2016-04-03 12:22:31.000   2016-04-06 12:23:14.000
10   10       2016-04-03 14:59:20.000   2016-04-06 15:00:03.000
11   11       2016-04-04 06:24:44.000   2016-04-07 06:25:27.000
```

Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 36 rows

C. The essential SQL skills 44'

1. [4] Write a SELECT statement that joins the Customers, Orders, OrderItems, and Products tables. This statement should return these columns: LastName, FirstName, OrderDate, ProductName, ItemPrice, DiscountAmount, and Quantity. Use aliases for the tables. Sort the final result set by LastName, OrderDate, and ProductName.


SQLQuery2.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (54))* - Microsoft SQL Server Management Studio (Administrator)

```sql
SELECT LastName +','+' '+ FirstName AS FullName, OrderDate, ProductName, ItemPrice, DiscountAmount, Quantity
FROM(((Orders INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID)
        INNER JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID)
        INNER JOIN Products ON OrderItems.ProductID = Products.ProductID)
ORDER BY LastName,OrderDate,ProductName
```

	FullName	OrderDate	ProductName	ItemPrice	DiscountAmount	Quantity
1	Albares, Cammy	2016-04-20 08:14:45.000	Rodriguez Caballero 11	699.00	209.70	1
2	Amigon, Minna	2016-04-11 08:21:32.000	Gibson SG	799.99	240.00	1
3	Brown, Christine	2016-03-30 15:22:31.000	Gibson Les Paul	1199.00	359.70	2
4	Butt, James	2016-04-04 06:24:44.000	Rodriguez Caballero 11	699.00	209.70	1
5	Caldarera, Kiley	2016-04-17 17:40:22.000	Rodriguez Caballero 11	699.00	209.70	1
6	Caudy, Chanel	2016-05-09 07:52:55.000	Hofner Icon	489.99	186.20	1
7	Caudy, Chanel	2016-05-09 07:52:55.000	Rodriguez Caballero 11	699.00	209.70	1
8	Darakjy, Josephine	2016-04-04 08:15:12.000	Fender Stratocaster	2517.00	1308.84	1
9	Dilliard, Leota	2016-04-06 18:41:53.000	Rodriguez Caballero 11	699.00	209.70	1
10	Esway, Heather	2016-04-03 14:59:20.000	Fender Precision	499.99	125.00	1
11	Esway, Heather	2016-04-12 12:26:52.000	Fender Stratocaster	2517.00	1308.84	1

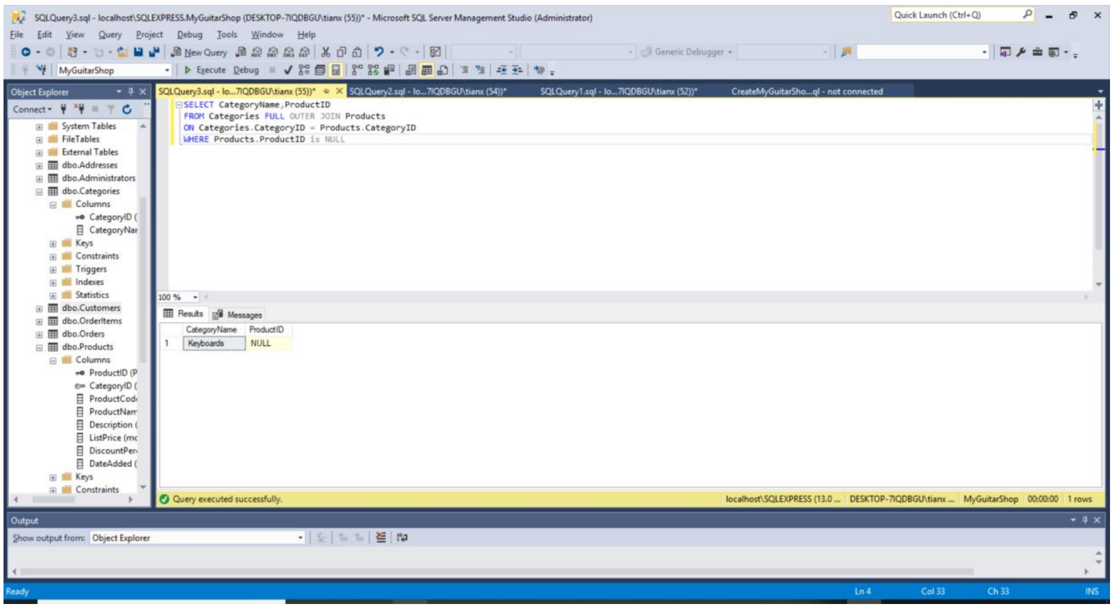
Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 47 rows

2. [4] Write a SELECT statement that returns these two columns:

CategoryName	The CategoryName column from the Categories table
ProductID	The ProductID column from the Products table

Return one row for each category that has never been used. (Hint: Use an **outer join** and only return rows where the ProductID column contains a null value.)

REMARK: INNER JOIN,LEFT JOIN,RIGHT JOIN, FULL OUTER JOIN


SQLQuery3.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (55))* - Microsoft SQL Server Management Studio (Administrator)

```sql
SELECT CategoryName,ProductID
FROM Categories FULL OUTER JOIN Products
ON Categories.CategoryID = Products.CategoryID
WHERE Products.ProductID is NULL
```

	CategoryName	ProductID
1	Keyboards	NULL

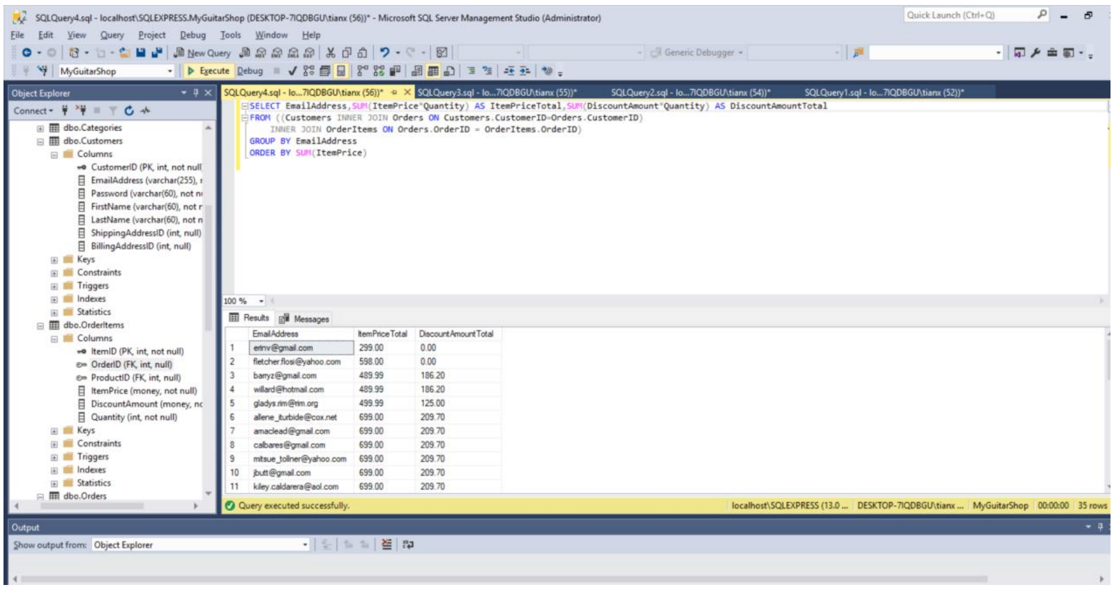
Query executed successfully. localhost\SQLEXPRESS (13.0 ... DESKTOP-7IQDBGU\tianx ... MyGuitarShop 00:00:00 1 rows

3. [4] Write a SELECT statement that returns one row for each customer that has orders with these columns:

a) The EmailAddress column from the Customers table

b) The sum of the item price in the OrderItems table multiplied by the quantity in the OrderItems table

c) The sum of the discount amount column in the OrderItems table multiplied by the quantity in the OrderItems table. Sort the result set in ascending sequence by the item price total for each customer.



```sql
SELECT EmailAddress,SUM(ItemPrice*Quantity) AS ItemPriceTotal,SUM(DiscountAmount*Quantity) AS DiscountAmountTotal
FROM ((Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID)
    INNER JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID)
GROUP BY EmailAddress
ORDER BY SUM(ItemPrice)
```

	EmailAddress	ItemPriceTotal	DiscountAmountTotal
1	erinv@gmail.com	299.00	0.00
2	fletcher.flosi@yahoo.com	598.00	0.00
3	barryz@gmail.com	489.99	186.20
4	willard@hotmail.com	489.99	186.20
5	gladys.rim@rim.org	499.99	125.00
6	allene_iturbide@cox.net	699.00	209.70
7	amaclead@gmail.com	699.00	209.70
8	calbares@gmail.com	699.00	209.70
9	mitsue_tollner@yahoo.com	699.00	209.70
10	jbutt@gmail.com	699.00	209.70
11	kiley.caldarera@aol.com	699.00	209.70

```
Query executed successfully.    localhost\SQLEXPRESS (13.0 ...  DESKTOP-7IQDBGU\tianx ...  MyGuitarShop  00:00:00  35 rows
```

4. [4] Write a SELECT statement that returns one row for each customer that has orders with these columns:

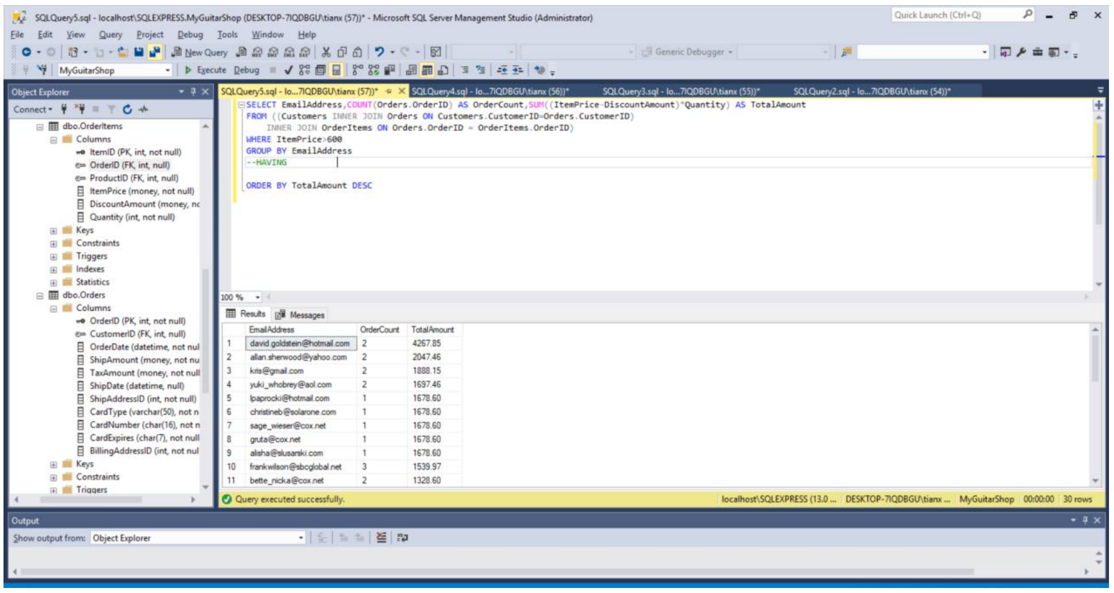
a) The EmailAdress column from the Customers table

b) A count of the number of orders

c) The total amount for each order (Hint: First, subtract the discount amount from the price. Then, multiply by the quantity)

Return only those rows where items have a more than 600 ItemPrice value.

Sort the result set in descending sequence by the sum of the line item amounts.

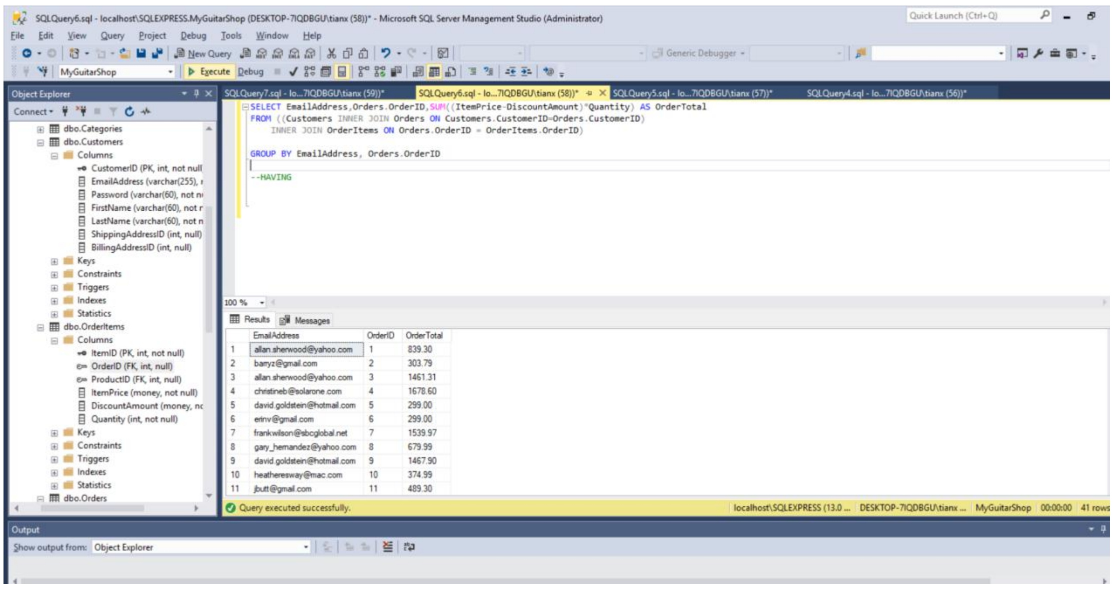


```sql
SELECT EmailAddress,COUNT(Orders.OrderID) AS OrderCount,SUM((ItemPrice-DiscountAmount)*Quantity) AS TotalAmount
FROM ((Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID)
    INNER JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID)
WHERE ItemPrice>600
GROUP BY EmailAddress
--HAVING

ORDER BY TotalAmount DESC
```

	EmailAddress	OrderCount	TotalAmount
1	david.goldstein@hotmail.com	2	4267.85
2	allan.sherwood@yahoo.com	2	2047.46
3	kris@gmail.com	2	1888.15
4	yuki_whobrey@aol.com	2	1697.46
5	lpaprocki@hotmail.com	1	1678.60
6	christineb@solarone.com	1	1678.60
7	sage_wieser@cox.net	1	1678.60
8	gruta@cox.net	1	1678.60
9	alisha@slusarski.com	1	1678.60
10	frankwilson@sbcglobal.net	3	1539.97
11	bette_nicka@cox.net	2	1328.60

```
Query executed successfully.    localhost\SQLEXPRESS (13.0 ...  DESKTOP-7IQDBGU\tianx ...  MyGuitarShop  00:00:00  30 rows
```

5. [4] (1) Write a SELECT statement that returns three columns: EmailAddress, OrderID, and the order total for each customer. To do this, you can group the result set by the EmailAddress and OrderID columns. In addition, you must calculate the order total from the columns in the OrderItems table.*REMARK:DIFFERENCE BETWEEN CUBE AND ROLL UP*

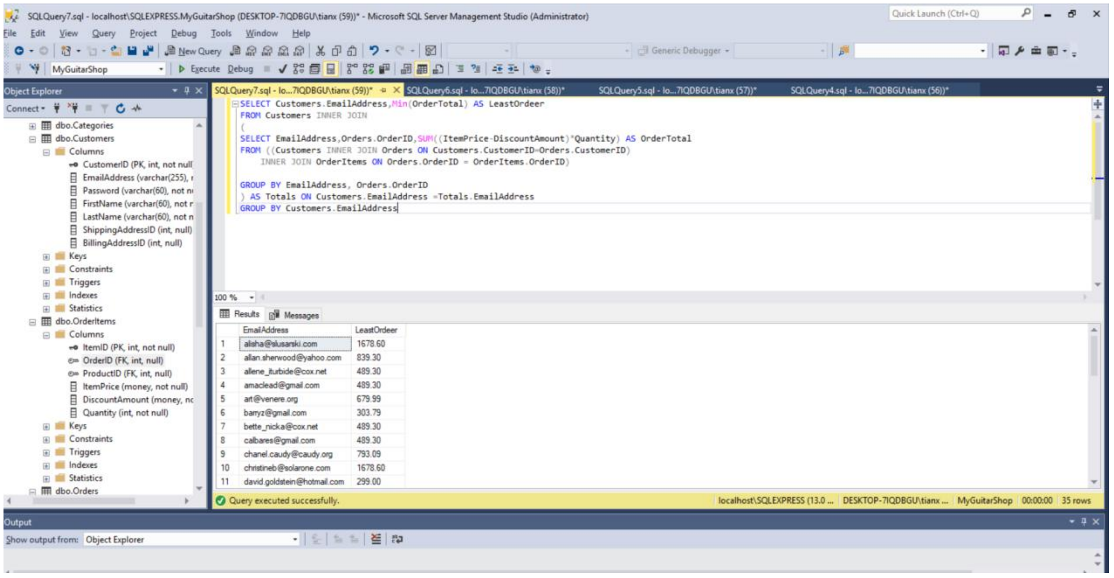


```
SELECT EmailAddress,Orders.OrderID,SUM((ItemPrice-DiscountAmount)*Quantity) AS OrderTotal
FROM ((Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID)
    INNER JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID)

GROUP BY EmailAddress, Orders.OrderID

--HAVING
```

	EmailAddress	OrderID	OrderTotal
1	allan.sherwood@yahoo.com	1	839.30
2	barryz@gmail.com	2	303.79
3	allan.sherwood@yahoo.com	3	1461.31
4	christineb@solarone.com	4	1678.60
5	david.goldstein@hotmail.com	5	299.00
6	erinv@gmail.com	6	299.00
7	frankwilson@sbcglobal.net	7	1539.97
8	gary_hernandez@yahoo.com	8	679.99
9	david.goldstein@hotmail.com	9	1467.90
10	heatheresway@mac.com	10	374.99
11	jbutt@gmail.com	11	489.30

```
Query executed successfully.        localhost\SQLEXPRESS (13.0 ...   DESKTOP-7IQDBGU\tianx ...   MyGuitarShop   00:00:00   41 rows
```

(2) Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer's email address and the least order for that customer. To do this, you can group the result set by the EmailAddress column.

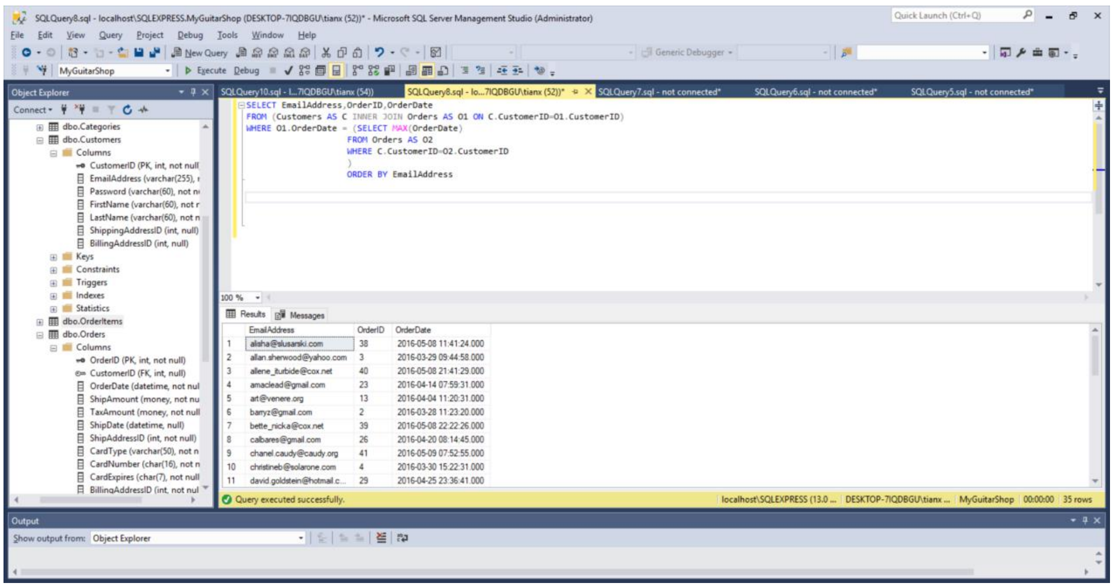


```
SELECT Customers.EmailAddress,Min(OrderTotal) AS LeastOrdeer
FROM Customers INNER JOIN
(
SELECT EmailAddress,Orders.OrderID,SUM((ItemPrice-DiscountAmount)*Quantity) AS OrderTotal
FROM ((Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID)
    INNER JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID)

GROUP BY EmailAddress, Orders.OrderID
) AS Totals ON Customers.EmailAddress =Totals.EmailAddress
GROUP BY Customers.EmailAddress
```

	EmailAddress	LeastOrdeer
1	alisha@slusarski.com	1678.60
2	allan.sherwood@yahoo.com	839.30
3	allene_iturbide@cox.net	489.30
4	amaclead@gmail.com	489.30
5	art@venere.org	679.99
6	barryz@gmail.com	303.79
7	bette_nicka@cox.net	489.30
8	calbares@gmail.com	489.30
9	chanel.caudy@caudy.org	793.09
10	christineb@solarone.com	1678.60
11	david.goldstein@hotmail.com	299.00

```
Query executed successfully.        localhost\SQLEXPRESS (13.0 ...   DESKTOP-7IQDBGU\tianx ...   MyGuitarShop   00:00:00   35 rows
```

6. [4] Use a correlated subquery to return one row per customer, representing the customer's newest order (the one with the latest date). Each row should include these three columns: EmailAddress, OrderID, and OrderDate.

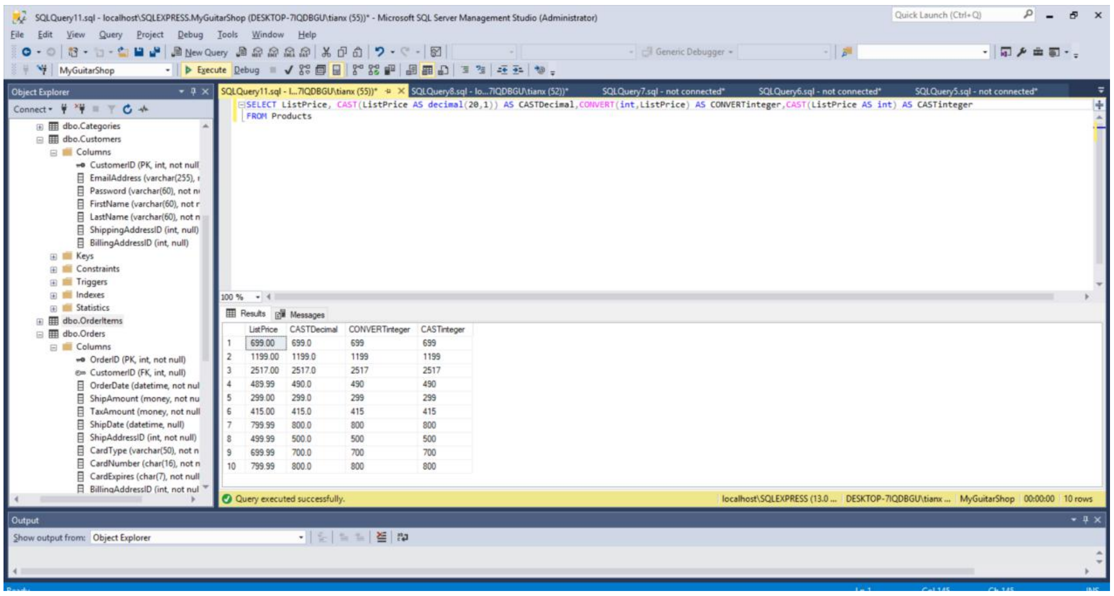


```sql
SELECT EmailAddress,OrderID,OrderDate
FROM (Customers AS C INNER JOIN Orders AS O1 ON C.CustomerID=O1.CustomerID)
WHERE O1.OrderDate = (SELECT MAX(OrderDate)
                      FROM Orders AS O2
                      WHERE C.CustomerID=O2.CustomerID
                      )
                      ORDER BY EmailAddress
```

	EmailAddress	OrderID	OrderDate
1	alisha@slusarski.com	38	2016-05-08 11:41:24.000
2	allan.sherwood@yahoo.com	3	2016-03-29 09:44:58.000
3	allene_iturbide@cox.net	40	2016-05-08 21:41:29.000
4	amaclead@gmail.com	23	2016-04-14 07:59:31.000
5	art@venere.org	13	2016-04-04 11:20:31.000
6	barryz@gmail.com	2	2016-03-28 11:23:20.000
7	bette_nicka@cox.net	39	2016-05-08 22:22:26.000
8	calbares@gmail.com	26	2016-04-20 08:14:45.000
9	chanel.caudy@caudy.org	41	2016-05-09 07:52:55.000
10	christineb@solarone.com	4	2016-03-30 15:22:31.000
11	david.goldstein@hotmail.c...	29	2016-04-25 23:36:41.000

Query executed successfully.

7. [4] Write a SELECT statement that returns these columns from the Products table:
a) The ListPrice column
  b) A column that uses the CAST function to return the ListPrice column with 1 digit to the right of the decimal point
  c) A column that uses the CONVERT function to return the ListPrice column as an integer
  d) A column that uses the CAST function to return the ListPrice column as an integer.
REMARK:DIFFERENCE BETWEEN CAST AND CONVERT

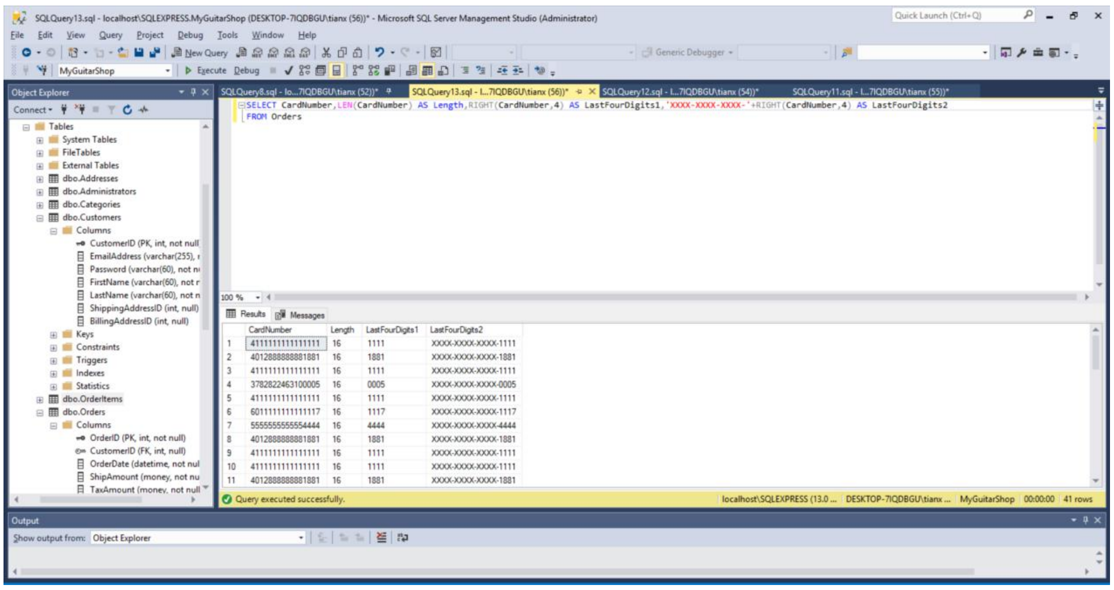


```sql
SELECT ListPrice, CAST(ListPrice AS decimal(20,1)) AS CASTDecimal,CONVERT(int,ListPrice) AS CONVERTinteger,CAST(ListPrice AS int) AS CASTinteger
FROM Products
```

	ListPrice	CASTDecimal	CONVERTinteger	CASTinteger
1	699.00	699.0	699	699
2	1199.00	1199.0	1199	1199
3	2517.00	2517.0	2517	2517
4	489.99	490.0	490	490
5	299.00	299.0	299	299
6	415.00	415.0	415	415
7	799.99	800.0	800	800
8	499.99	500.0	500	500
9	699.99	700.0	700	700
10	799.99	800.0	800	800

Query executed successfully.

8. [4] Write a SELECT statement that returns these columns from the Orders table:
  a) The CardNumber column
b) The length of the CardNumber column
  c) The last four digits of the CardNumber column
  d) A column that displays the last four digits of the CardNumber column in this format: XXXXXXXX-XXXX-1234. In other words, use Xs for the first 12 digits of the card number and actual numbers for the last four digits of the number.



```sql
SELECT CardNumber,LEN(CardNumber) AS Length,RIGHT(CardNumber,4) AS LastFourDigits1,'XXXX-XXXX-XXXX-'+RIGHT(CardNumber,4) AS LastFourDigits2
FROM Orders
```

	CardNumber	Length	LastFourDigits1	LastFourDigits2
1	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
2	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
3	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
4	378282246310005	16	0005	XXXX-XXXX-XXXX-0005
5	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
6	6011111111111117	16	1117	XXXX-XXXX-XXXX-1117
7	5555555555554444	16	4444	XXXX-XXXX-XXXX-4444
8	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
9	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
10	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
11	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881

Query executed successfully.

9. [4] Write a SELECT statement that returns these columns from the Orders table:

a) The OrderID column

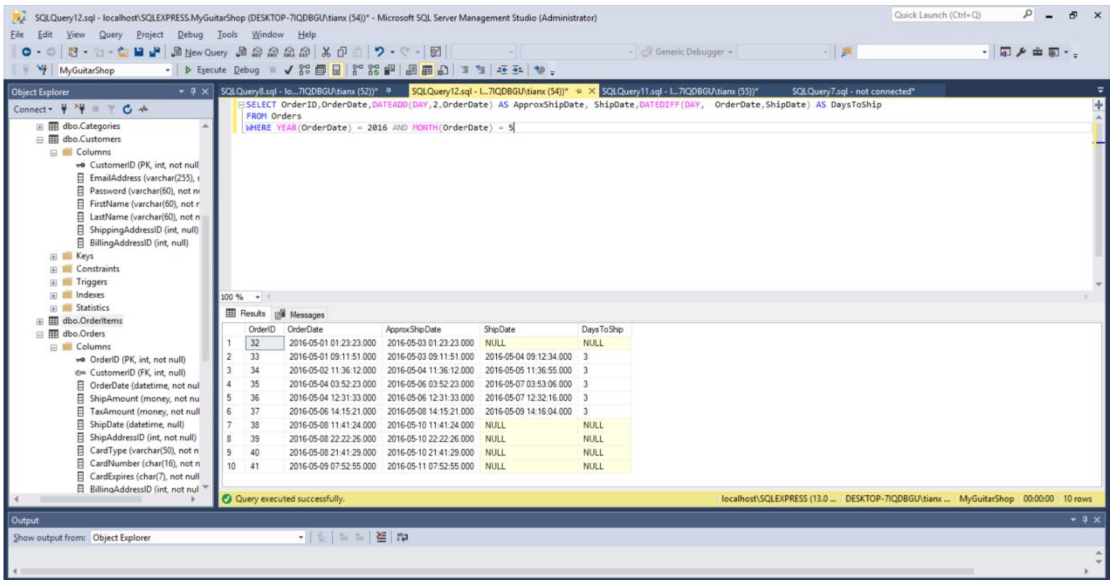
b) The OrderDate column

c) A column named ApproxShipDate that's calculated by adding 2 days to the OrderDate column

d) The ShipDate column

e) A column named DaysToShip that shows the number of days between the order date and the ship date

When you have this working, add a WHERE clause that retrieves just the orders for May 2016.



```sql
SELECT OrderID,OrderDate,DATEADD(DAY,2,OrderDate) AS ApproxShipDate, ShipDate,DATEDIFF(DAY,  OrderDate,ShipDate) AS DaysToShip
FROM Orders
WHERE YEAR(OrderDate) = 2016 AND MONTH(OrderDate) = 5
```

	OrderID	OrderDate	ApproxShipDate	ShipDate	DaysToShip
1	32	2016-05-01 01:23:23.000	2016-05-03 01:23:23.000	NULL	NULL
2	33	2016-05-01 09:11:51.000	2016-05-03 09:11:51.000	2016-05-04 09:12:34.000	3
3	34	2016-05-02 11:36:12.000	2016-05-04 11:36:12.000	2016-05-05 11:36:55.000	3
4	35	2016-05-04 03:52:23.000	2016-05-06 03:52:23.000	2016-05-07 03:53:06.000	3
5	36	2016-05-04 12:31:33.000	2016-05-06 12:31:33.000	2016-05-07 12:32:16.000	3
6	37	2016-05-06 14:15:21.000	2016-05-08 14:15:21.000	2016-05-09 14:16:04.000	3
7	38	2016-05-08 11:41:24.000	2016-05-10 11:41:24.000	NULL	NULL
8	39	2016-05-08 22:22:26.000	2016-05-10 22:22:26.000	NULL	NULL
9	40	2016-05-08 21:41:29.000	2016-05-10 21:41:29.000	NULL	NULL
10	41	2016-05-09 07:52:55.000	2016-05-11 07:52:55.000	NULL	NULL

Query executed successfully.

For question 10-11: To test whether a table has been modified correctly as you do these questions, please write and run an appropriate SELECT statement and take full screenshots of the verification.

10. [4] Write an INSERT statement that adds this row to the Customers table:

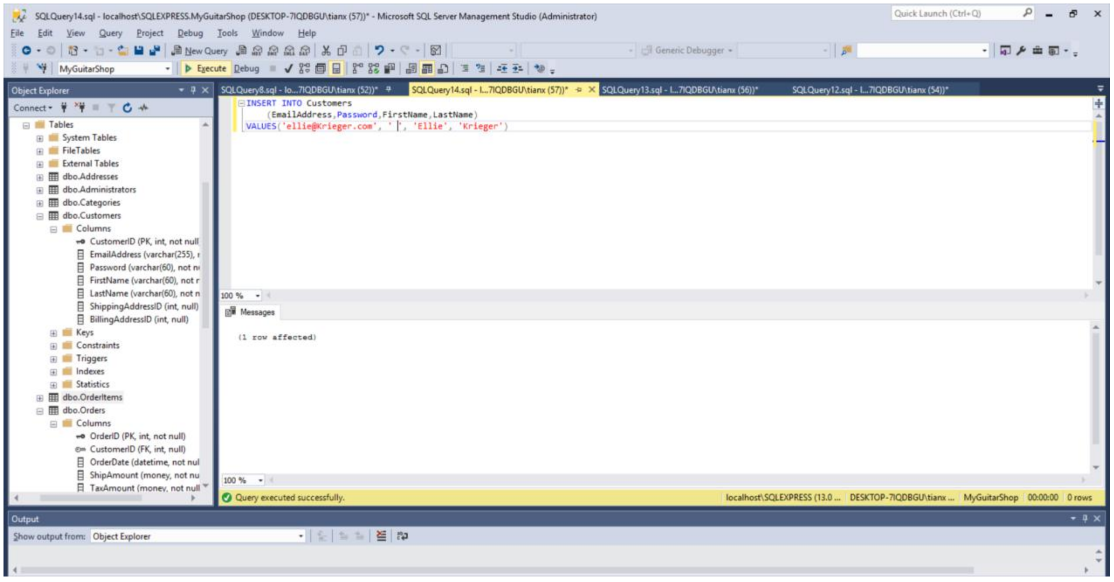
EmailAddress: ellie@Krieger.com

Password: (empty string)

FirstName: Ellie

LastName: Krieger

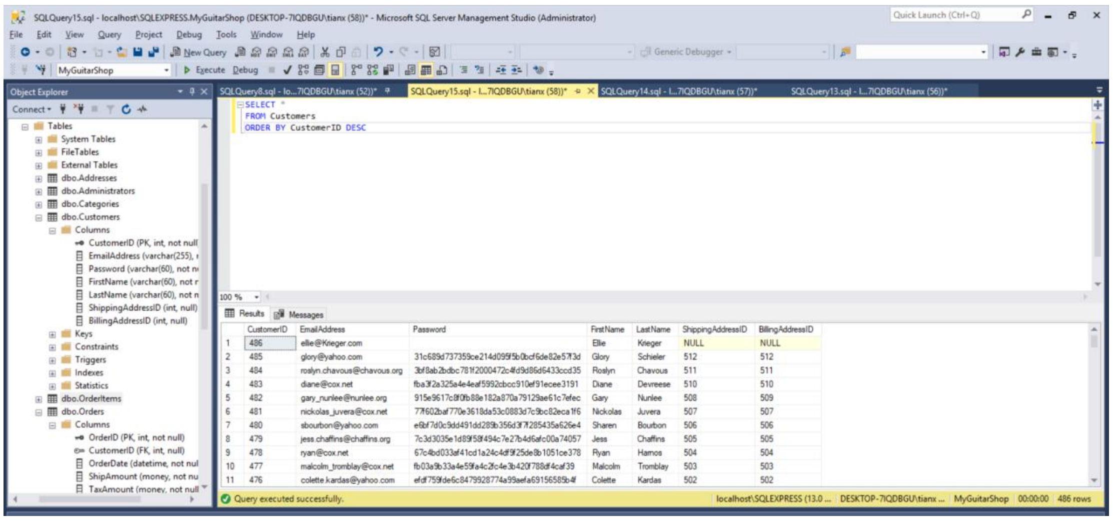
Use a column list for this statement



```
INSERT INTO Customers
    (EmailAddress,Password,FirstName,LastName)
VALUES('ellie@Krieger.com', ' ', 'Ellie', 'Krieger')
```

```
(1 row affected)
```

Query executed successfully.

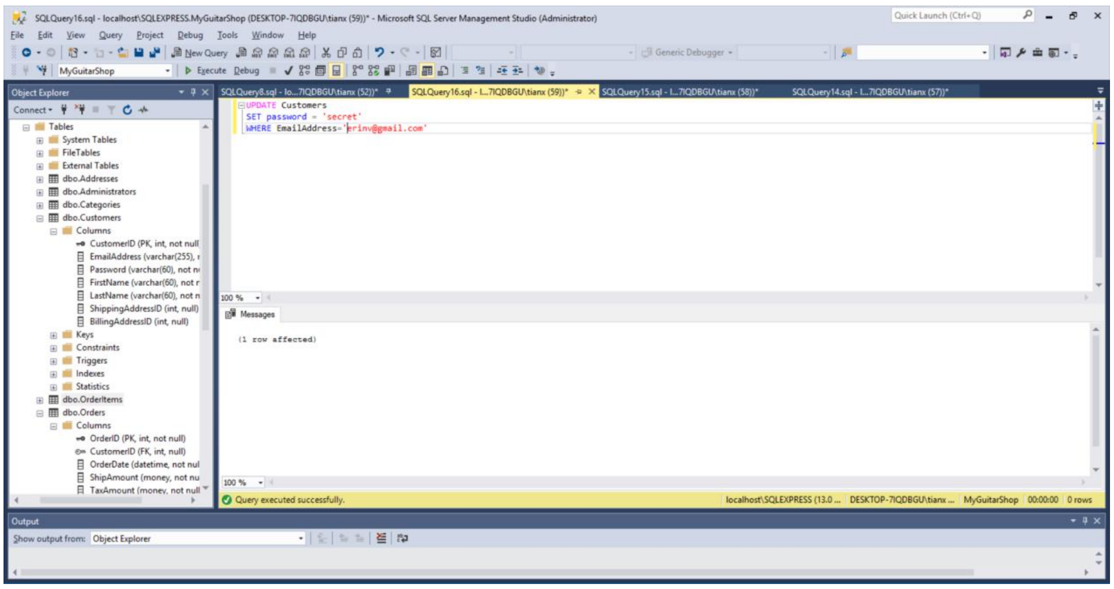


```
SELECT *
FROM Customers
ORDER BY CustomerID DESC
```

	CustomerID	EmailAddress	Password	FirstName	LastName	ShippingAddressID	BillingAddressID
1	486	ellie@Krieger.com		Ellie	Krieger	NULL	NULL
2	485	glory@yahoo.com	31c689d737359ce214d095f5b0bcf6de82e57f3d	Glory	Schieler	512	512
3	484	roslyn.chavous@chavous.org	3bf8ab2bdbc781f2000472c4fd9d86d6433ccd35	Roslyn	Chavous	511	511
4	483	diane@cox.net	fba3f2a325a4e4eaf5992cbcc910ef91ecee3191	Diane	Devreese	510	510
5	482	gary_nunlee@nunlee.org	915e9617c8f0fb88e182a870a79129ae61c7efec	Gary	Nunlee	508	509
6	481	nickolas_juvera@cox.net	77f602baf770e3618da53c0883d7c9bc82eca1f6	Nickolas	Juvera	507	507
7	480	sbourbon@yahoo.com	e6bf7d0c9dd491dd289b356d3f7f285435a626e4	Sharen	Bourbon	506	506
8	479	jess.chaffins@chaffins.org	7c3d3035e1d89f58f494c7e27b4d6afc00a74057	Jess	Chaffins	505	505
9	478	ryan@cox.net	67c4bd033af41cd1a24c4df9f25de8b1051ce378	Ryan	Hamos	504	504
10	477	malcolm_tromblay@cox.net	fb03a9b33a4e59fa4c2fc4e3b420f788df4caf39	Malcolm	Tromblay	503	503
11	476	colette.kardas@yahoo.com	efdf759fde6c8479928774a99aefa69156585b4f	Colette	Kardas	502	502

Query executed successfully. 486 rows

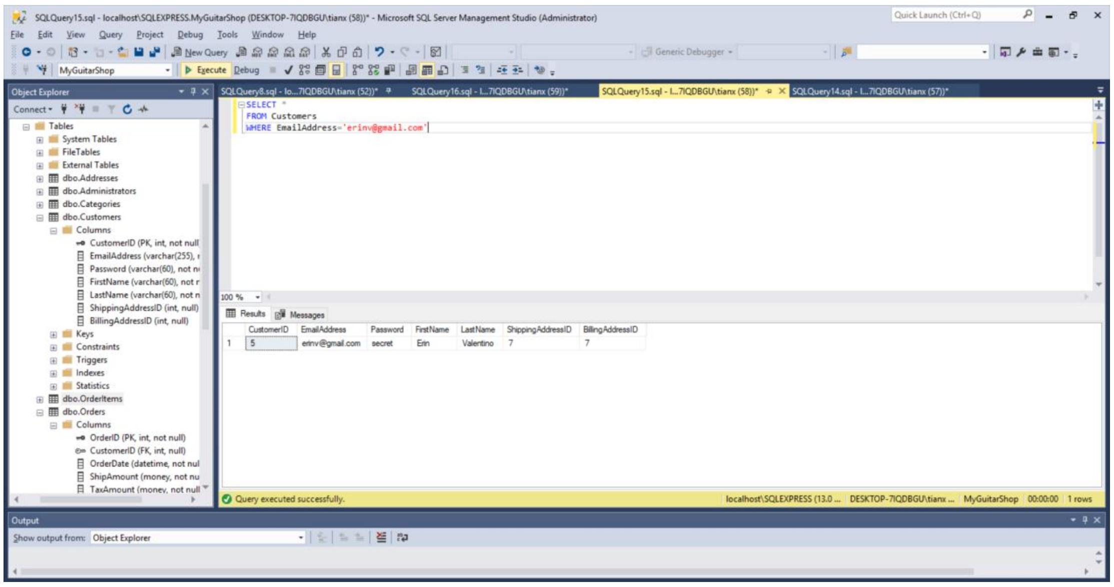
11. [4] Write an UPDATE statement that modifies the Customers table. Change the password column to “secret” for the customer with an email address of erinv@gmail.com.



```
UPDATE Customers
SET password = 'secret'
WHERE EmailAddress='erinv@gmail.com'
```

```
(1 row affected)
```

Query executed successfully.



```
SELECT *
FROM Customers
WHERE EmailAddress='erinv@gmail.com'
```

	CustomerID	EmailAddress	Password	FirstName	LastName	ShippingAddressID	BillingAddressID
1	5	erinv@gmail.com	secret	Erin	Valentino	7	7

Query executed successfully.

D. Advanced SQL skills(views/stored procedures/functions/scripts) 24'

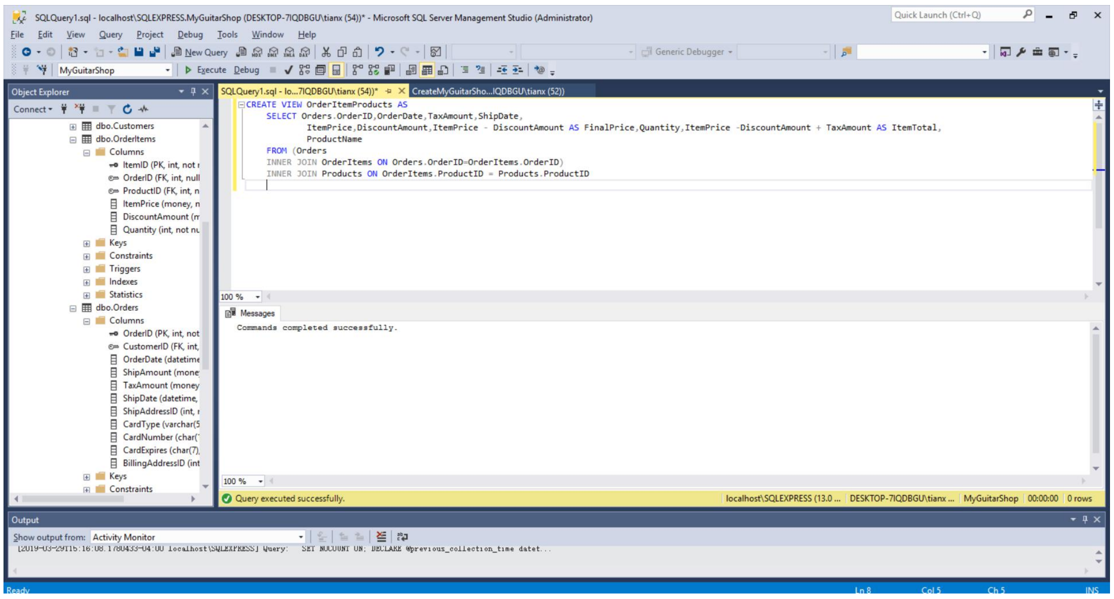
Open the script named CreateMyGuitarShop.sql and run this script. That should restore the data that's in the database. Then complete questions in Section D. Screenshot of execution is required for each question. Please also use SELECT statement to verify results of your codes (Screenshots of verification are required).

1. [4] Create a view named OrderItemProducts that returns columns from the Orders, OrderItems, and Products tables.

a) This view should return these columns from the Orders table: OrderID, OrderDate, TaxAmount, and ShipDate.

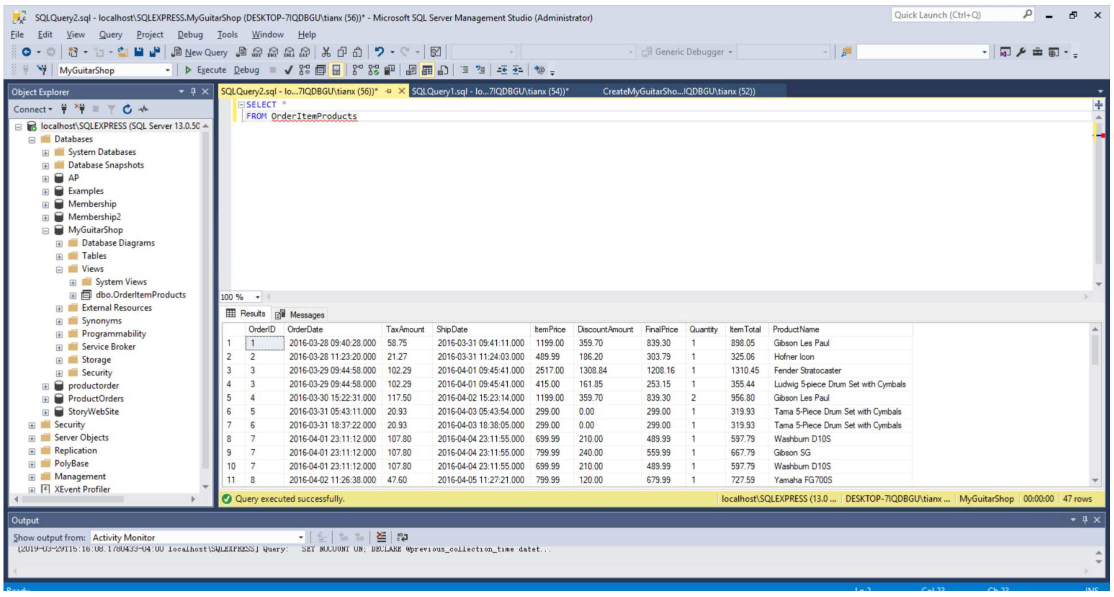
b) This view should return these columns from the OrderItems table: ItemPrice, DiscountAmount, FinalPrice (the discount amount subtracted from the item price), Quantity, and ItemTotal (the calculated total for the item).

c) This view should return the ProductName column from the Products table.



```sql
CREATE VIEW OrderItemProducts AS
    SELECT Orders.OrderID,OrderDate,TaxAmount,ShipDate,
           ItemPrice,DiscountAmount,ItemPrice - DiscountAmount AS FinalPrice,Quantity,ItemPrice -DiscountAmount + TaxAmount AS ItemTotal,
           ProductName
    FROM (Orders
    INNER JOIN OrderItems ON Orders.OrderID=OrderItems.OrderID)
    INNER JOIN Products ON OrderItems.ProductID = Products.ProductID
```

```
Commands completed successfully.
```

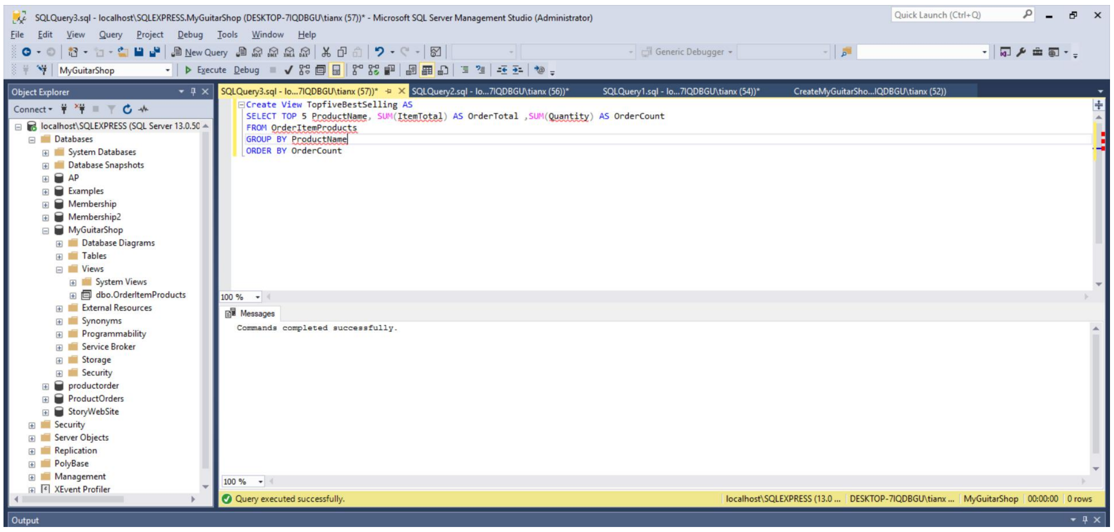


```sql
SELECT *
FROM OrderItemProducts
```

	OrderID	OrderDate	TaxAmount	ShipDate	ItemPrice	DiscountAmount	FinalPrice	Quantity	ItemTotal	ProductName
1	1	2016-03-28 09:40:28.000	58.75	2016-03-31 09:41:11.000	1199.00	359.70	839.30	1	898.05	Gibson Les Paul
2	2	2016-03-28 11:23:20.000	21.27	2016-03-31 11:24:03.000	489.99	186.20	303.79	1	325.06	Hofner Icon
3	3	2016-03-29 09:44:58.000	102.29	2016-04-01 09:45:41.000	2517.00	1308.84	1208.16	1	1310.45	Fender Stratocaster
4	3	2016-03-29 09:44:58.000	102.29	2016-04-01 09:45:41.000	415.00	161.85	253.15	1	355.44	Ludwig 5-piece Drum Set with Cymbals
5	4	2016-03-30 15:22:31.000	117.50	2016-04-02 15:23:14.000	1199.00	359.70	839.30	2	956.80	Gibson Les Paul
6	5	2016-03-31 05:43:11.000	20.93	2016-04-03 05:43:54.000	299.00	0.00	299.00	1	319.93	Tama 5-Piece Drum Set with Cymbals
7	6	2016-03-31 18:37:22.000	20.93	2016-04-03 18:38:05.000	299.00	0.00	299.00	1	319.93	Tama 5-Piece Drum Set with Cymbals
8	7	2016-04-01 23:11:12.000	107.80	2016-04-04 23:11:55.000	699.99	210.00	489.99	1	597.79	Washburn D10S
9	7	2016-04-01 23:11:12.000	107.80	2016-04-04 23:11:55.000	799.99	240.00	559.99	1	667.79	Gibson SG
10	7	2016-04-01 23:11:12.000	107.80	2016-04-04 23:11:55.000	699.99	210.00	489.99	1	597.79	Washburn D10S
11	8	2016-04-02 11:26:38.000	47.60	2016-04-05 11:27:21.000	799.99	120.00	679.99	1	727.59	Yamaha FG700S

```
Query executed successfully.    localhost\SQLEXPRESS (13.0 ...   DESKTOP-7IQDBGU\tianx ...   MyGuitarShop   00:00:00   47 rows
```

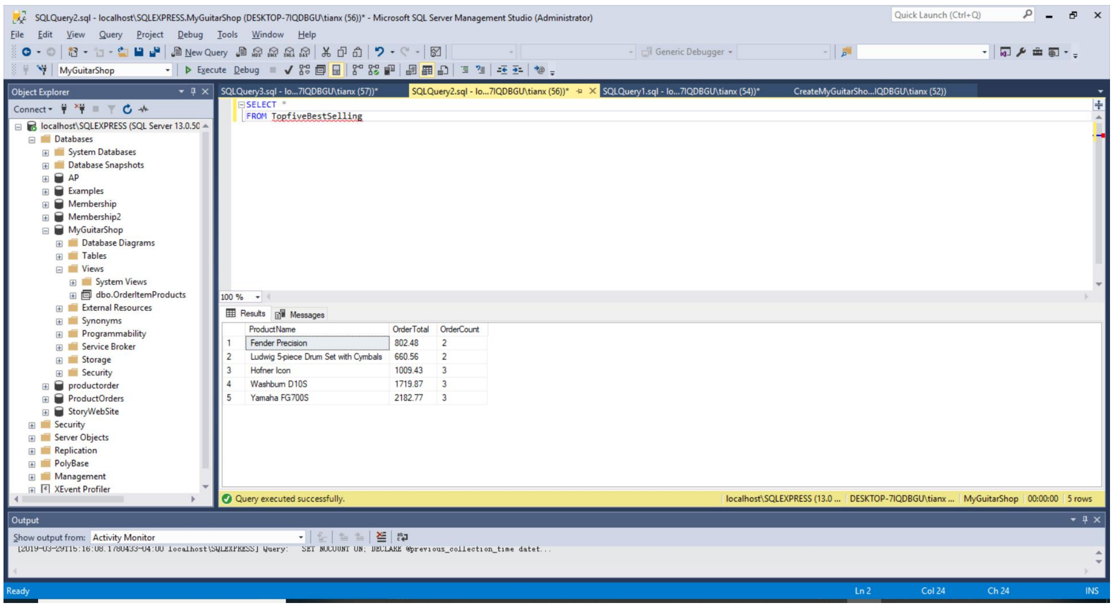
2. [5] Create a view named Top5BestSelling that uses the view you created in Section D Question This view should return some summary information about five best selling products. Each row should include these columns: ProductName, **OrderTotal (the total sales for the product)** and **OrderCount (the number of times the product has been ordered).**



```
Create View TopfiveBestSelling AS
SELECT TOP 5 ProductName, SUM(ItemTotal) AS OrderTotal ,SUM(Quantity) AS OrderCount
FROM OrderItemProducts
GROUP BY ProductName
ORDER BY OrderCount
```

```
Commands completed successfully.
```

Query executed successfully.

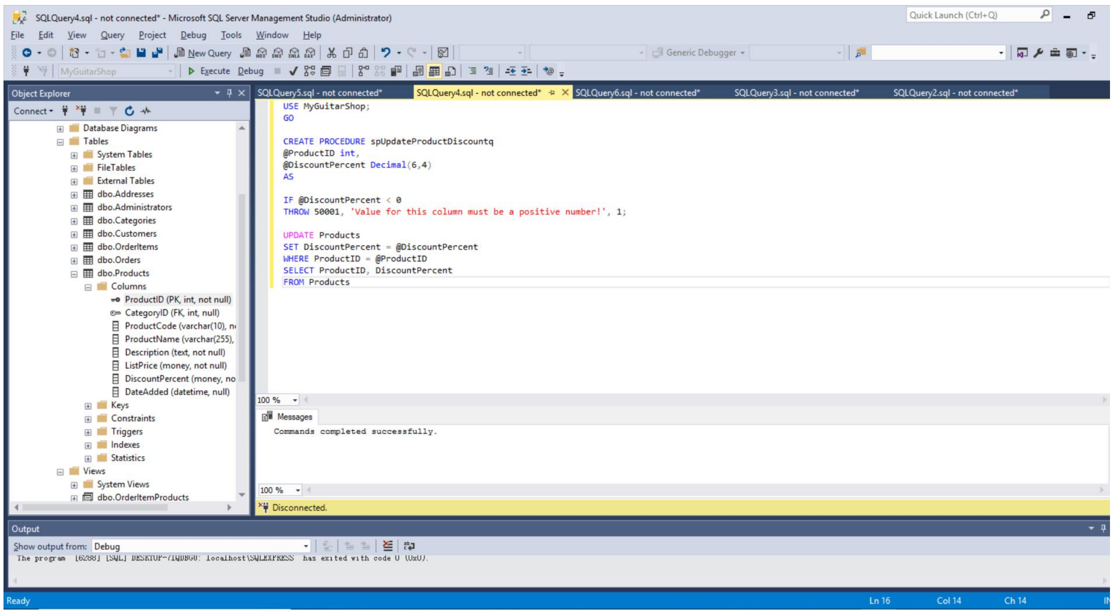


```
SELECT *
FROM TopfiveBestSelling
```

	ProductName	OrderTotal	OrderCount
1	Fender Precision	802.48	2
2	Ludwig 5-piece Drum Set with Cymbals	660.56	2
3	Hofner Icon	1009.43	3
4	Washburn D10S	1719.87	3
5	Yamaha FG700S	2182.77	3

Query executed successfully.

3. [5] Write a **script** that creates and calls a **stored procedure** named spUpdateProductDiscount that updates the DiscountPercent column in the Products table. This procedure should have one parameter for the product ID and another for the discount percent. If the value for the DiscountPercent column is a negative number, the stored procedure should raise an error that indicates that the value for this column must be a positive number. Code at least two EXEC statements that test this procedure.

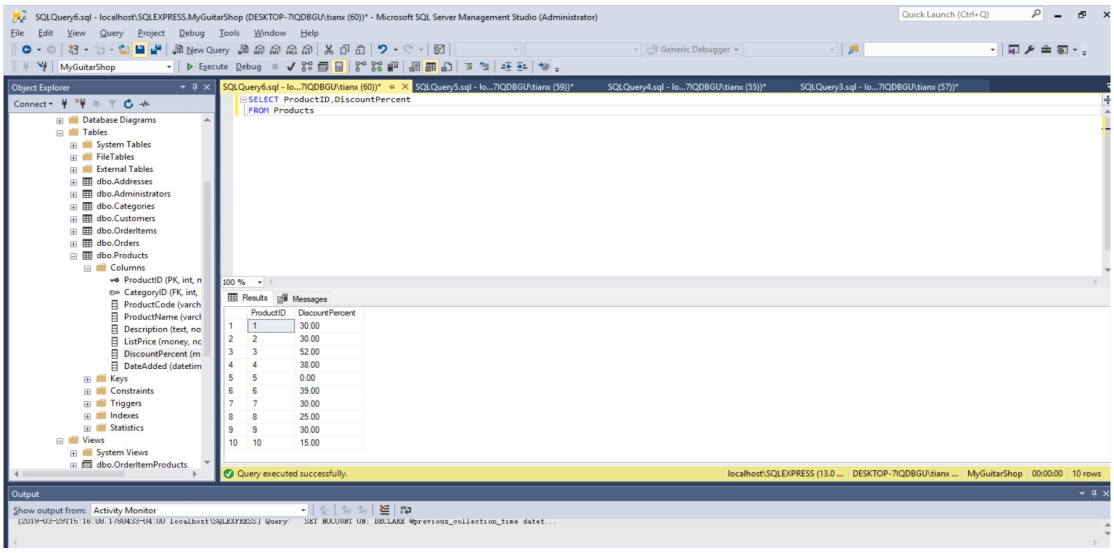


```
USE MyGuitarShop;
GO

CREATE PROCEDURE spUpdateProductDiscountq
@ProductID int,
@DiscountPercent Decimal(6,4)
AS

IF @DiscountPercent < 0
THROW 50001, 'Value for this column must be a positive number!', 1;

UPDATE Products
SET DiscountPercent = @DiscountPercent
WHERE ProductID = @ProductID
SELECT ProductID, DiscountPercent
FROM Products
```

```
Commands completed successfully.
```

```
The program '[6288] [SQL] DESKTOP-7IQDBGU: localhost\SQLEXPRESS' has exited with code 0 (0x0).
```

Execute before

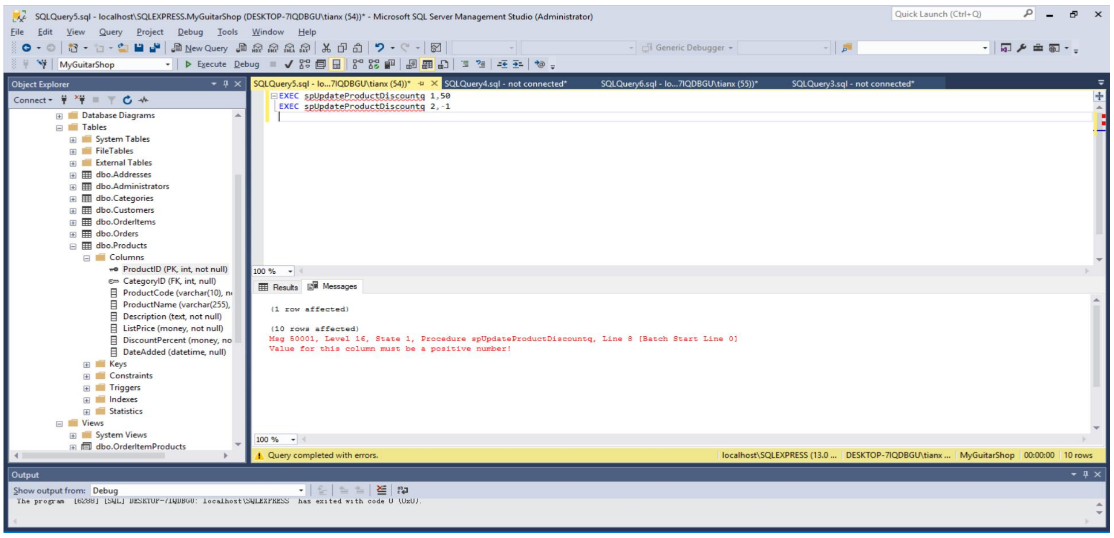


```
SELECT ProductID,DiscountPercent
FROM Products
```

	ProductID	DiscountPercent
1	1	30.00
2	2	30.00
3	3	52.00
4	4	38.00
5	5	0.00
6	6	39.00
7	7	30.00
8	8	25.00
9	9	30.00
10	10	15.00

Query executed successfully.

Execute after



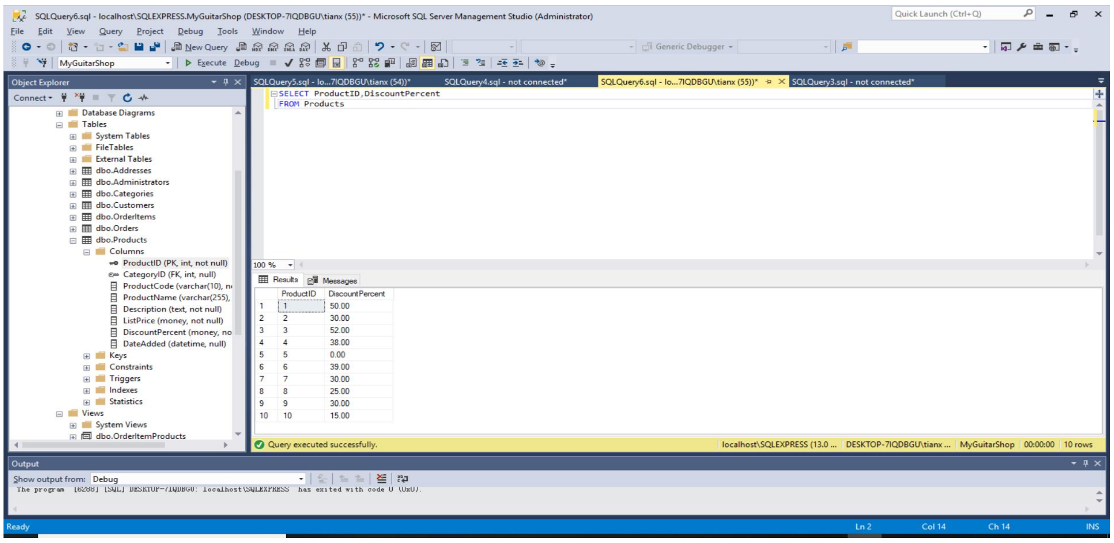
SQLQuery5.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (54))* - Microsoft SQL Server Management Studio (Administrator)

```
EXEC spUpdateProductDiscountq 1,50
EXEC spUpdateProductDiscountq 2,-1
```

```
(1 row affected)

(10 rows affected)
Msg 50001, Level 16, State 1, Procedure spUpdateProductDiscountq, Line 8 [Batch Start Line 0]
Value for this column must be a positive number!
```

Query completed with errors.

```
The program [6288] [SQL] DESKTOP-7IQDBGU: localhost\SQLEXPRESS' has exited with code 0 (0x0).
```



SQLQuery6.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (55))* - Microsoft SQL Server Management Studio (Administrator)

```
SELECT ProductID,DiscountPercent
FROM Products
```

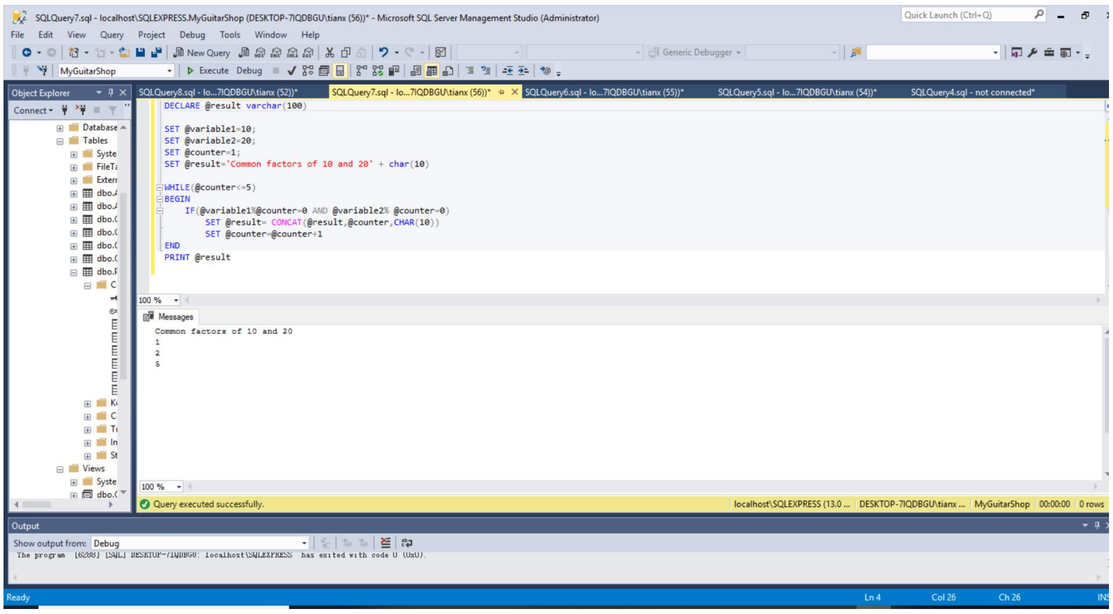
	ProductID	DiscountPercent
1	1	50.00
2	2	30.00
3	3	52.00
4	4	38.00
5	5	0.00
6	6	39.00
7	7	30.00
8	8	25.00
9	9	30.00
10	10	15.00

Query executed successfully.

```
The program [6288] [SQL] DESKTOP-7IQDBGU: localhost\SQLEXPRESS' has exited with code 0 (0x0).
```

4. [5] Write a **script** that calculates the **common factors** between 10 and 20. To find a common factor, you can use the modulo operator (%) to check whether a number can be evenly divided into both numbers. Then, this script should print lines that display the common factors like this: Common factors of 10 and 20

```
1
2
5
```

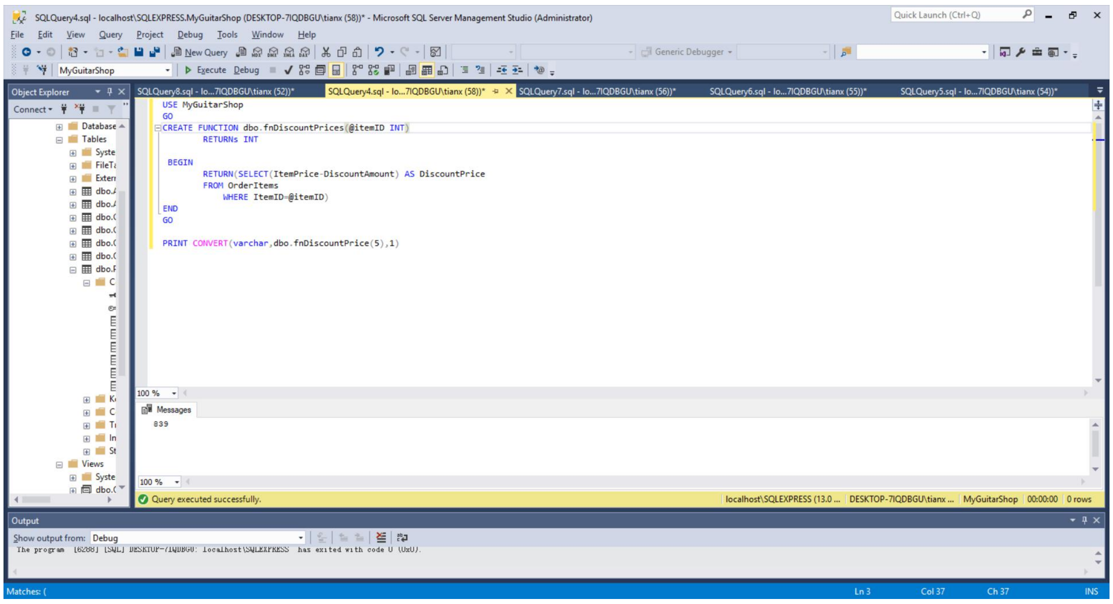


```
DECLARE @result varchar(100)

SET @variable1=10;
SET @variable2=20;
SET @counter=1;
SET @result='Common factors of 10 and 20' + char(10)

WHILE(@counter<=5)
BEGIN
    IF(@variable1%@counter=0 AND @variable2% @counter=0)
        SET @result= CONCAT(@result,@counter,CHAR(10))
        SET @counter=@counter+1
END
PRINT @result
```

```
Common factors of 10 and 20
1
2
5
```

Query executed successfully.

5. [5] (1) Write a **script** that **creates and calls a function** named fnDiscountPrice that calculates the discount price of an item in the OrderItems table (discount amount subtracted from item price). To do that, this function should accept one parameter for the item ID, and it should return the value of the discount price for that item.

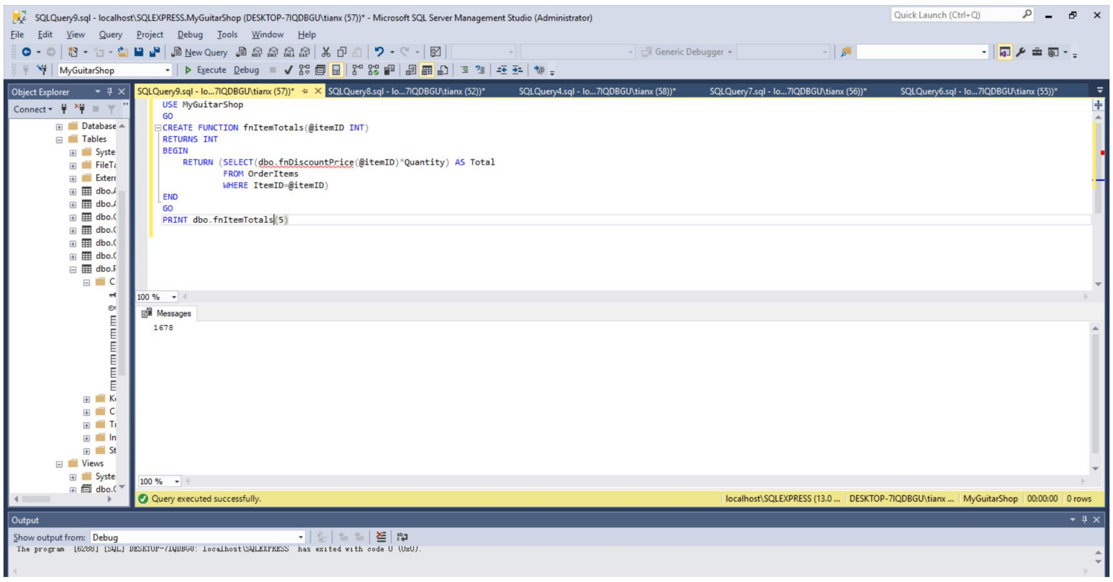


```
USE MyGuitarShop
GO
CREATE FUNCTION dbo.fnDiscountPrices(@itemID INT)
        RETURNs INT

 BEGIN
        RETURN(SELECT(ItemPrice-DiscountAmount) AS DiscountPrice
        FROM OrderItems
            WHERE ItemID=@itemID)
END
GO

PRINT CONVERT(varchar,dbo.fnDiscountPrice(5),1)
```

```
839
```

Query executed successfully.

(2) Write a **script** that **creates and calls** a function named **fnItemTotal** that calculates the **total amount of an item** in the OrderItems table (Discount price multiplied by quantity). To do that, this function should accept one parameter for the item ID, it should use the DiscountPrice function that you created in (1), and it should return the value of the **total** for that item.



SQLQuery9.sql - localhost\SQLEXPRESS.MyGuitarShop (DESKTOP-7IQDBGU\tianx (57))* - Microsoft SQL Server Management Studio (Administrator)

```
USE MyGuitarShop
GO
CREATE FUNCTION fnItemTotals(@itemID INT)
RETURNS INT
BEGIN
    RETURN (SELECT(dbo.fnDiscountPrice(@itemID)*Quantity) AS Total
            FROM OrderItems
            WHERE ItemID=@itemID)
END
GO
PRINT dbo.fnItemTotals(5)
```

Messages

```
1678
```

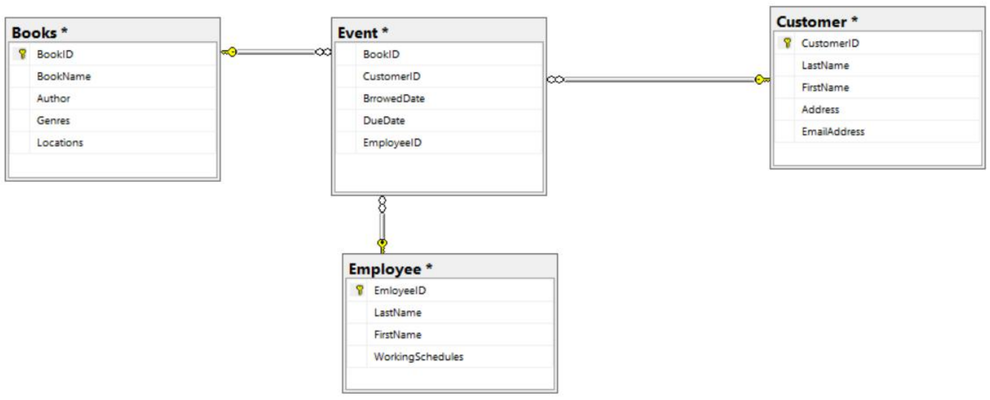
Query executed successfully. | localhost\SQLEXPRESS (13.0 ... | DESKTOP-7IQDBGU\tianx ... | MyGuitarShop | 00:00:00 | 0 rows

```
The program [6288] [SQL] DESKTOP-7IQDBGU: localhost\SQLEXPRESS has exited with code 0 (0x0).
```

II.Database Design 20’

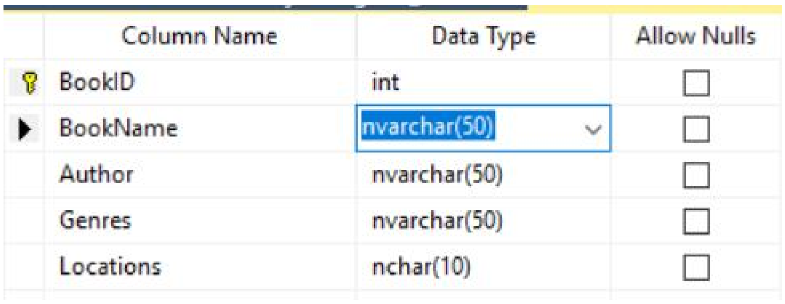
Create a sample database design for a **library** and draw one **database model** for it. The library wants to keep track of books (regular books/electronic/audio); book authors; book genres; book location (area/shelves); customer information; books that the customers borrowed, when they are due, employee that checked them out; employee information; and employee working schedules.

1. [3] Design the database that makes sense for the problem and select fields that make the most sense. A complete screenshot of your final design model is required.



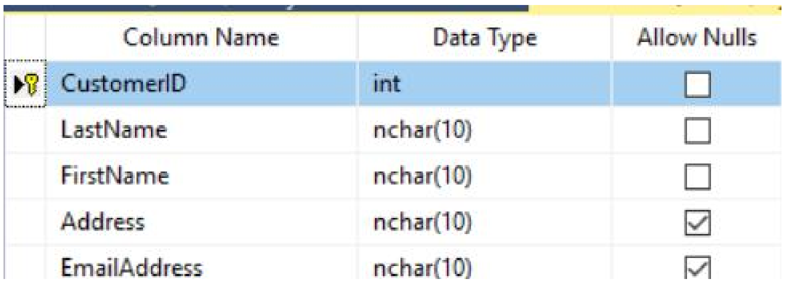
2. [10] Determine the tables, columns, primary keys, nullabilities and show relationships between tables (one -one/one-many/many-many).

Books Primary Key:BookID



	Column Name	Data Type	Allow Nulls
🔑	BookID	int	☐
▶	BookName	nvarchar(50)	☐
	Author	nvarchar(50)	☐
	Genres	nvarchar(50)	☐
	Locations	nchar(10)	☐

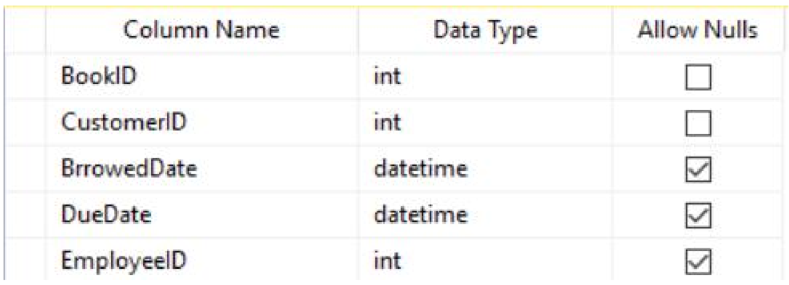
Customer Primary Key:CustomerID



	Column Name	Data Type	Allow Nulls
▶🔑	CustomerID	int	☐
	LastName	nchar(10)	☐
	FirstName	nchar(10)	☐
	Address	nchar(10)	☑
	EmailAddress	nchar(10)	☑

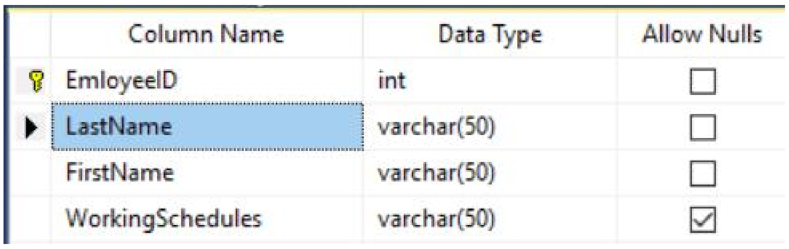
Event Primary Key: Composite key BookID and Customer ID

Foreign Key: BookID, CustomerID,EmployeeID



	Column Name	Data Type	Allow Nulls
	BookID	int	☐
	CustomerID	int	☐
	BrrowedDate	datetime	☑
	DueDate	datetime	☑
	EmployeeID	int	☑

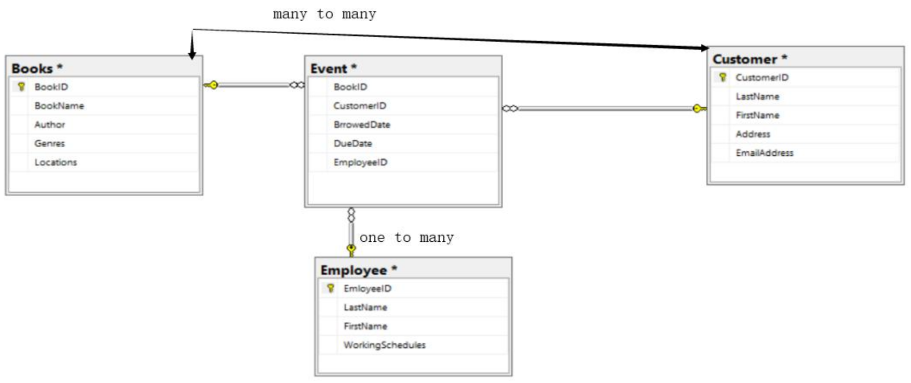
Employee Primary Key:EmployeeID



	Column Name	Data Type	Allow Nulls
🔑	EmloyeeID	int	☐
▶	LastName	varchar(50)	☐
	FirstName	varchar(50)	☐
	WorkingSchedules	varchar(50)	☑

3. [5] Normalize your design into 3rd Normal Form. Please use MS Visio or any similar database design tool.

All the attributes in 4 table are determined only by the candidate key of that relation and not by any non-prime attributes.



4. [2] Explain your design, including relationship between tables.

Relationship:

1) Books and Customers is many to many relationship, one customer could brrow different book, one book could be borrowed from different customer.

2) Event and Employee is one to many relationship, one Employee could check many event, but one event could only checked by one employee

Entity: Book, Customer, Event,Employee

Entity Property: **Book** has ID,bookname,author, genre to describle its basic information
Customer has CustomerID and basic information about them.
Event record each borrowing book from library, it has borrowed bookID,borrow borrowing customer and Due Date, also the employee who check book out

In your typed report (pdf or docx only), please include the following items:

1. Complete SQL source codes with your comments.
2. Full screenshots of the requested actions or each question including the total number of rows in result set and your name, which are listed above with your comments on them.
3. Your remarks on the project [4 pts.]. The remarks should be specific thoughts and references you learned that are related with knowledge points