CSE 581
Introduction to Database Management System

Project 1

Leo Wang

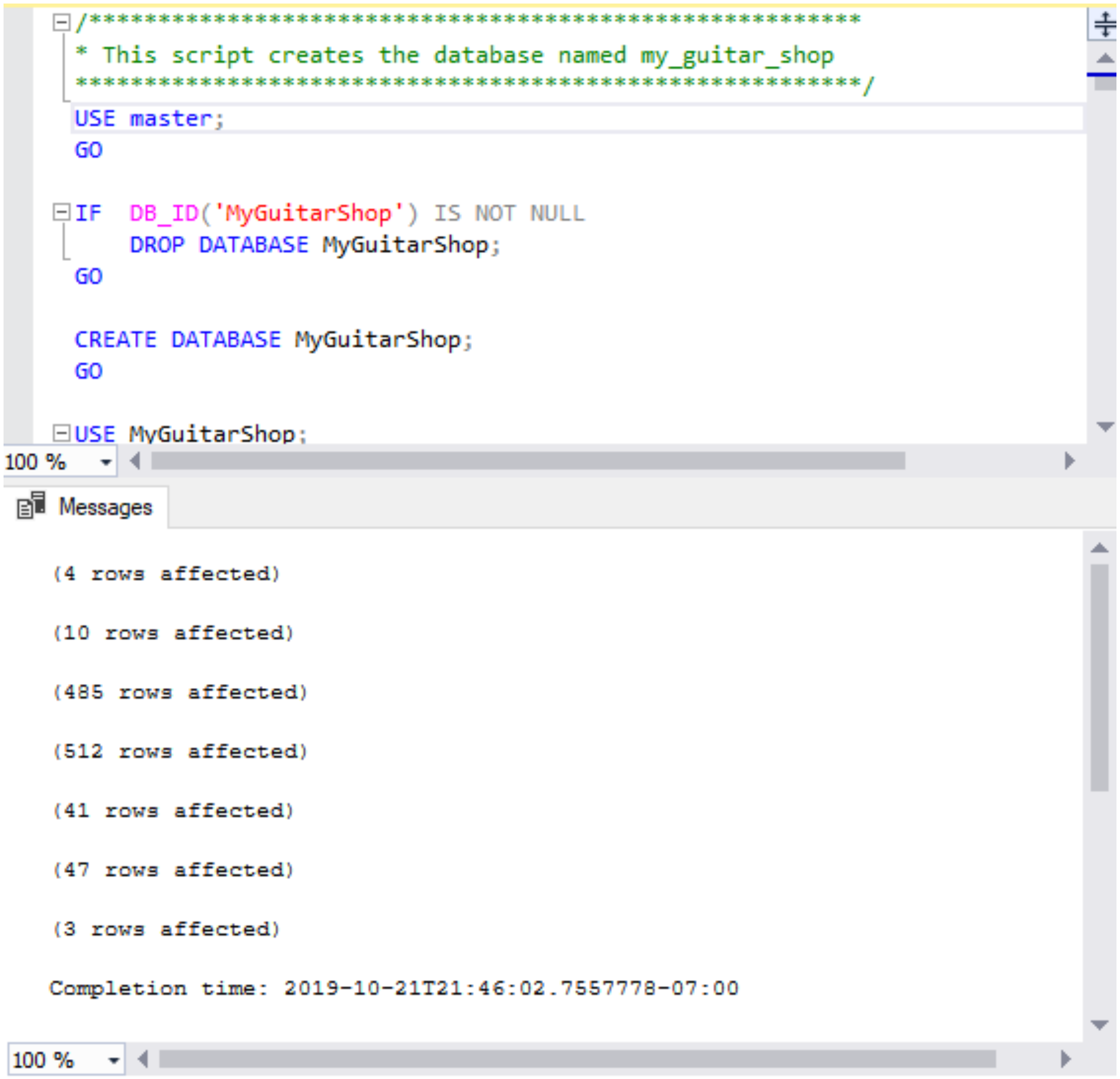
Dr. Ehat Ercanli

2019 October 30

I. My Guitar Shop Database

A. Database Setup

1. (1) The CreateMyGuitarShop.sql is downloaded and executed successfully. The complete screenshot of the execution result is shown below.



```
/*****************************************************
* 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;
```

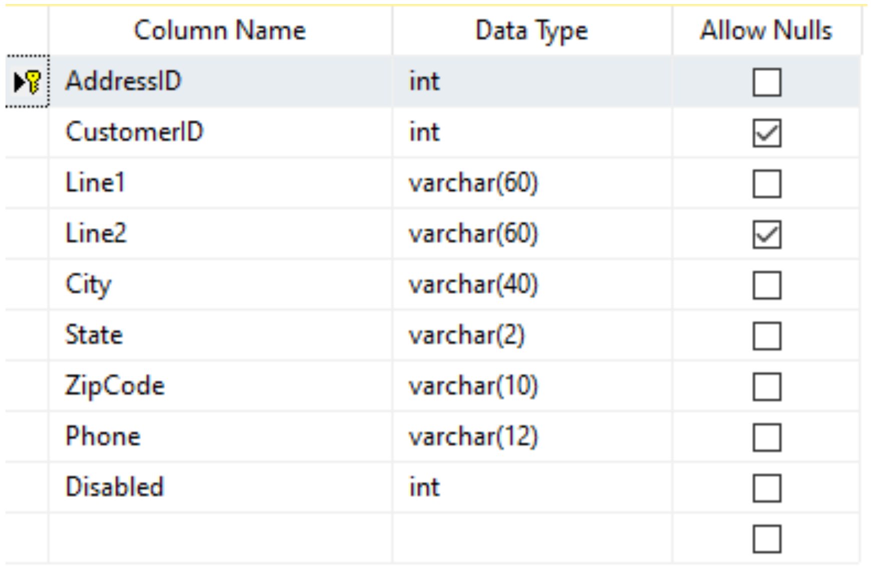
100 %

Messages

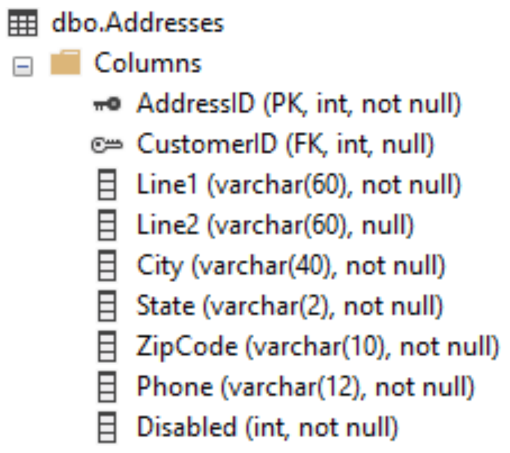
```
(4 rows affected)

(10 rows affected)

(485 rows affected)

(512 rows affected)

(41 rows affected)

(47 rows affected)

(3 rows affected)

Completion time: 2019-10-21T21:46:02.7557778-07:00
```

100 %

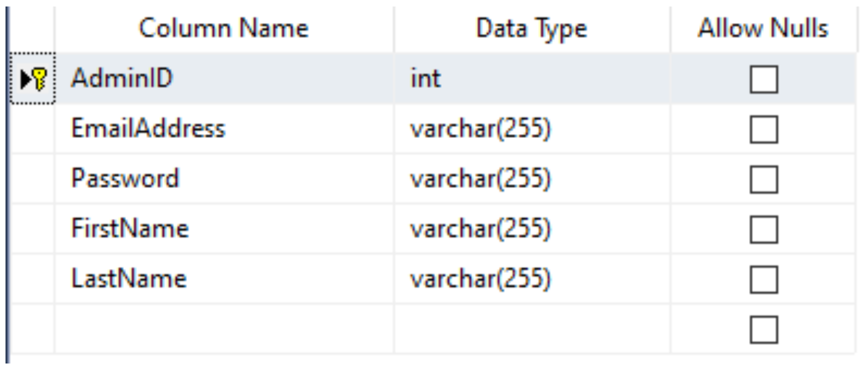
1 (2) Navigate through the database objects and view the column definitions of each table.



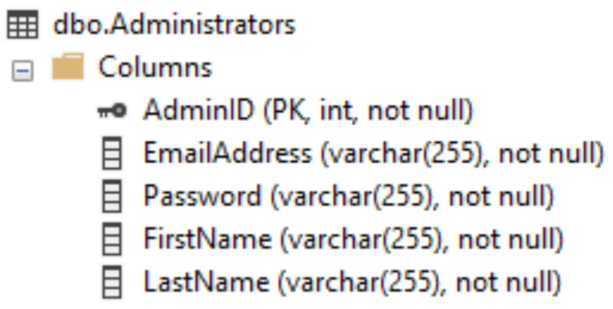
Column Name	Data Type	Allow Nulls
AddressID	int	☐
CustomerID	int	☑
Line1	varchar(60)	☐
Line2	varchar(60)	☑
City	varchar(40)	☐
State	varchar(2)	☐
ZipCode	varchar(10)	☐
Phone	varchar(12)	☐
Disabled	int	☐
		☐



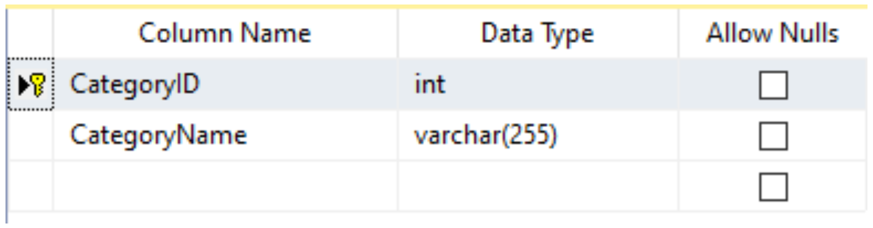
dbo.Addresses
- Columns
  - AddressID (PK, int, not null)
  - CustomerID (FK, int, null)
  - Line1 (varchar(60), not null)
  - Line2 (varchar(60), null)
  - City (varchar(40), not null)
  - State (varchar(2), not null)
  - ZipCode (varchar(10), not null)
  - Phone (varchar(12), not null)
  - Disabled (int, not null)



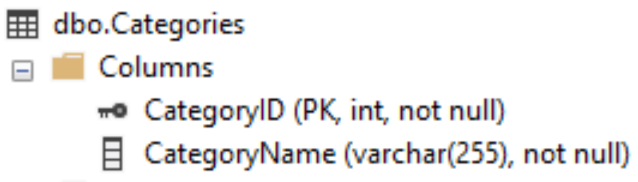
Column Name	Data Type	Allow Nulls
AdminID	int	☐
EmailAddress	varchar(255)	☐
Password	varchar(255)	☐
FirstName	varchar(255)	☐
LastName	varchar(255)	☐
		☐



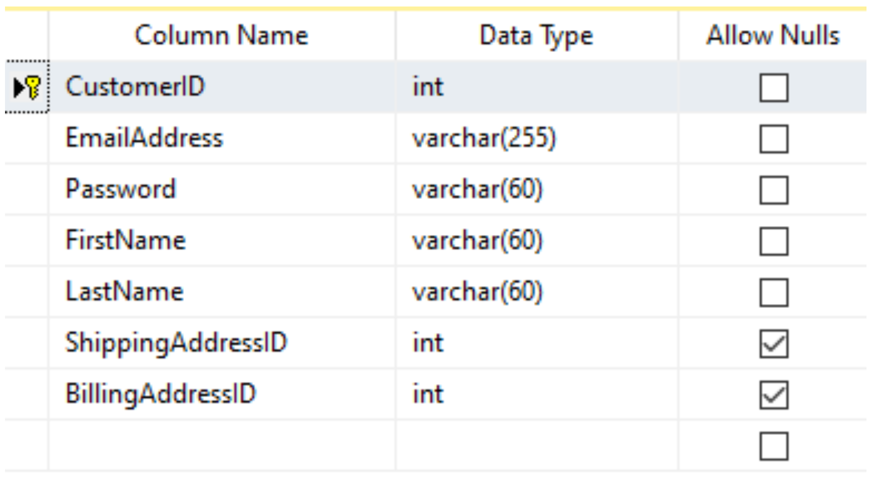
dbo.Administrators
- Columns
  - AdminID (PK, int, not null)
  - EmailAddress (varchar(255), not null)
  - Password (varchar(255), not null)
  - FirstName (varchar(255), not null)
  - LastName (varchar(255), not null)



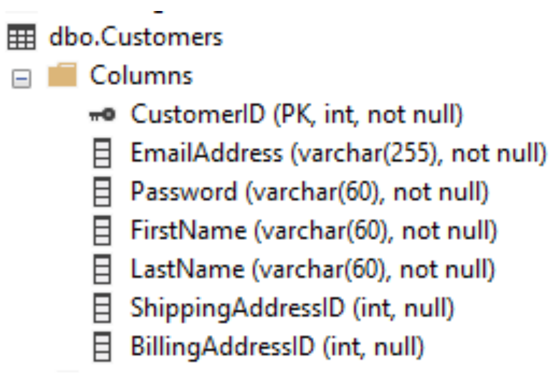
Column Name	Data Type	Allow Nulls
CategoryID	int	☐
CategoryName	varchar(255)	☐
		☐



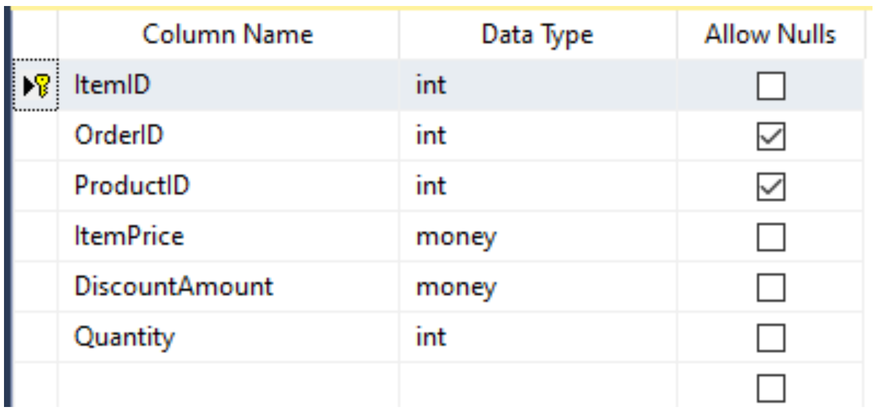
dbo.Categories
- Columns
  - CategoryID (PK, int, not null)
  - CategoryName (varchar(255), not null)



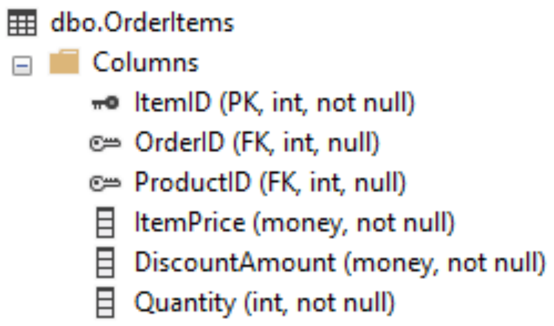
Column Name	Data Type	Allow Nulls
CustomerID	int	☐
EmailAddress	varchar(255)	☐
Password	varchar(60)	☐
FirstName	varchar(60)	☐
LastName	varchar(60)	☐
ShippingAddressID	int	☑
BillingAddressID	int	☑
		☐



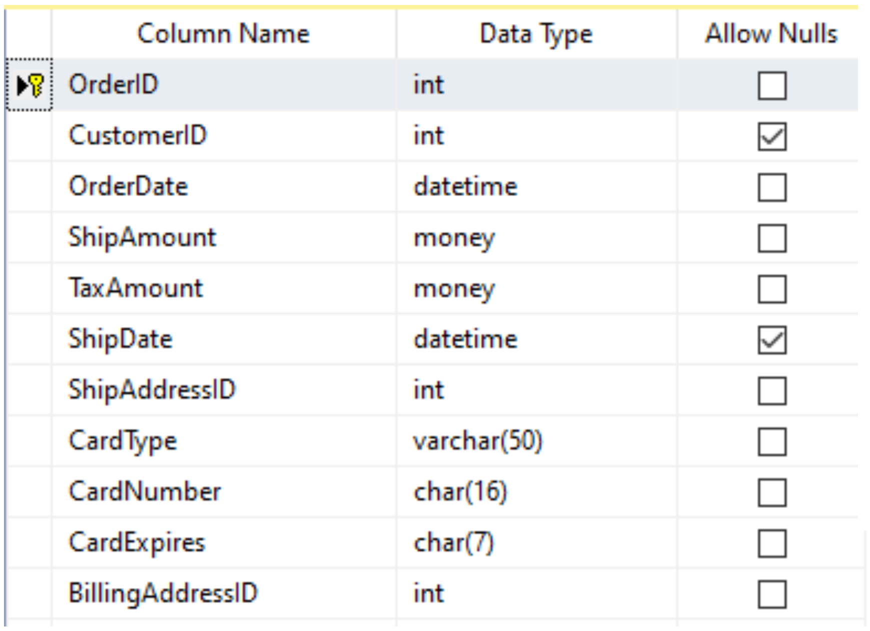
dbo.Customers
- Columns
  - CustomerID (PK, int, not null)
  - EmailAddress (varchar(255), not null)
  - Password (varchar(60), not null)
  - FirstName (varchar(60), not null)
  - LastName (varchar(60), not null)
  - ShippingAddressID (int, null)
  - BillingAddressID (int, null)



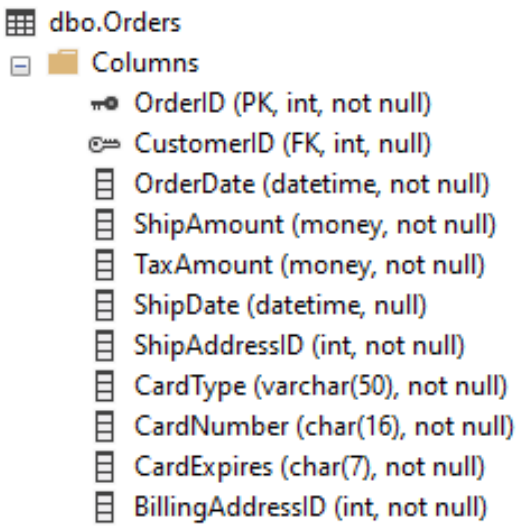
Column Name	Data Type	Allow Nulls
ItemID	int	☐
OrderID	int	☑
ProductID	int	☑
ItemPrice	money	☐
DiscountAmount	money	☐
Quantity	int	☐
		☐



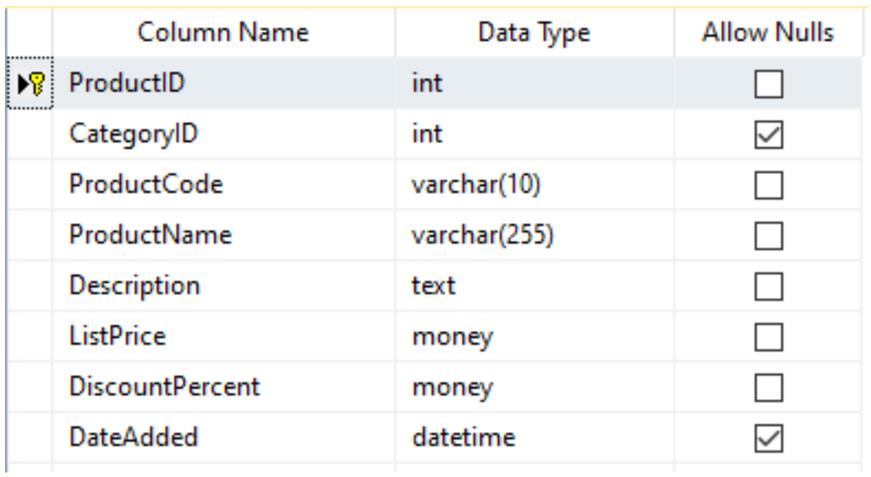
dbo.OrderItems
- Columns
  - ItemID (PK, int, not null)
  - OrderID (FK, int, null)
  - ProductID (FK, int, null)
  - ItemPrice (money, not null)
  - DiscountAmount (money, not null)
  - Quantity (int, not null)



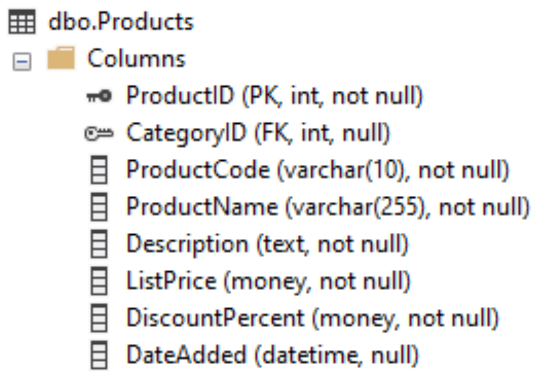
Column Name	Data Type	Allow Nulls
OrderID	int	☐
CustomerID	int	☑
OrderDate	datetime	☐
ShipAmount	money	☐
TaxAmount	money	☐
ShipDate	datetime	☑
ShipAddressID	int	☐
CardType	varchar(50)	☐
CardNumber	char(16)	☐
CardExpires	char(7)	☐
BillingAddressID	int	☐



dbo.Orders
- Columns
  - OrderID (PK, int, not null)
  - CustomerID (FK, int, null)
  - OrderDate (datetime, not null)
  - ShipAmount (money, not null)
  - TaxAmount (money, not null)
  - ShipDate (datetime, null)
  - ShipAddressID (int, not null)
  - CardType (varchar(50), not null)
  - CardNumber (char(16), not null)
  - CardExpires (char(7), not null)
  - BillingAddressID (int, not null)

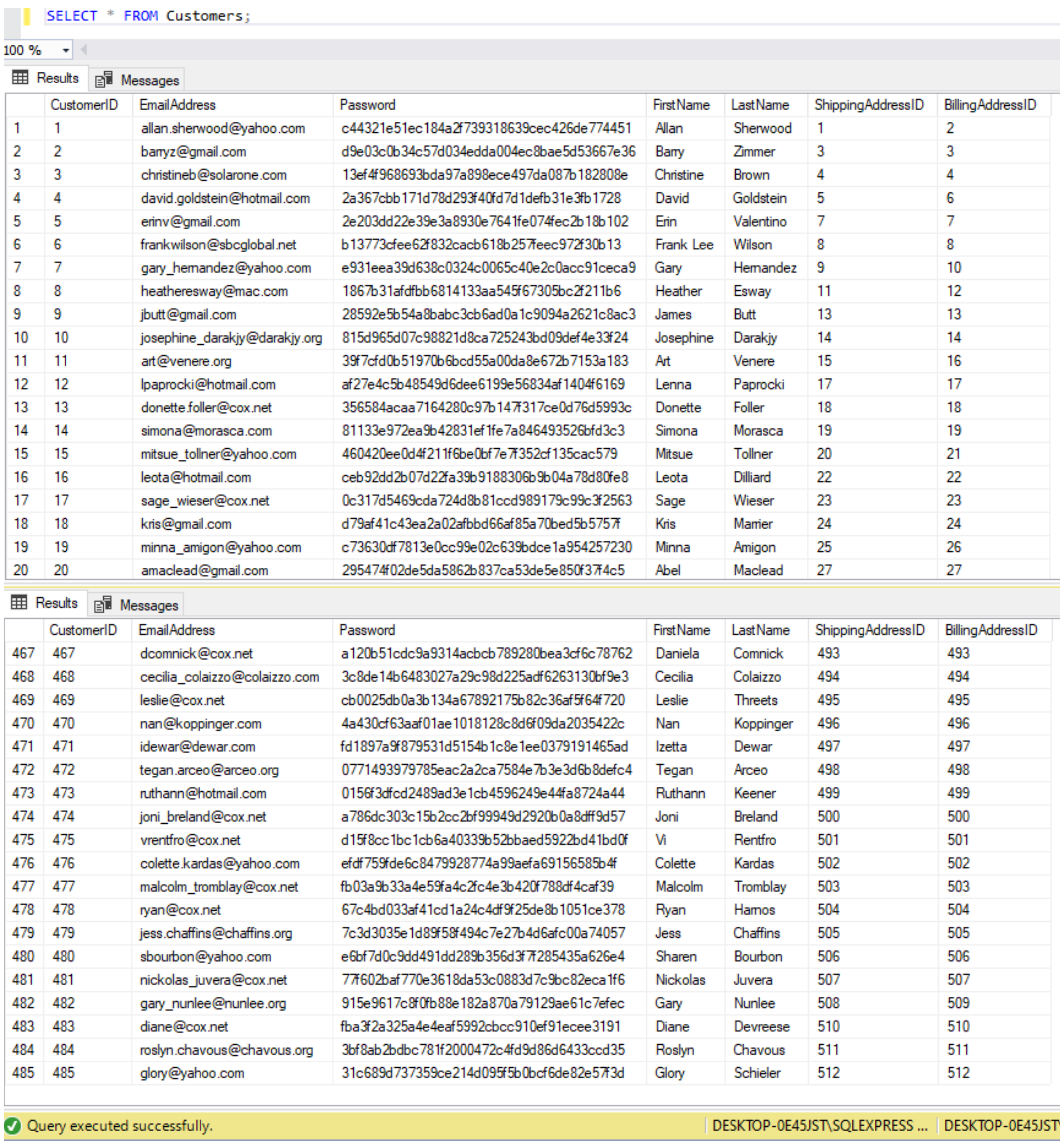


Column Name	Data Type	Allow Nulls
ProductID	int	☐
CategoryID	int	☑
ProductCode	varchar(10)	☐
ProductName	varchar(255)	☐
Description	text	☐
ListPrice	money	☐
DiscountPercent	money	☐
DateAdded	datetime	☑



dbo.Products
- Columns
  - ProductID (PK, int, not null)
  - CategoryID (FK, int, null)
  - ProductCode (varchar(10), not null)
  - ProductName (varchar(255), not null)
  - Description (text, not null)
  - ListPrice (money, not null)
  - DiscountPercent (money, not null)
  - DateAdded (datetime, null)

The following are the details of Customers table using SELECT statement.



```sql
SELECT * FROM Customers;
```

100 %

Results | Messages

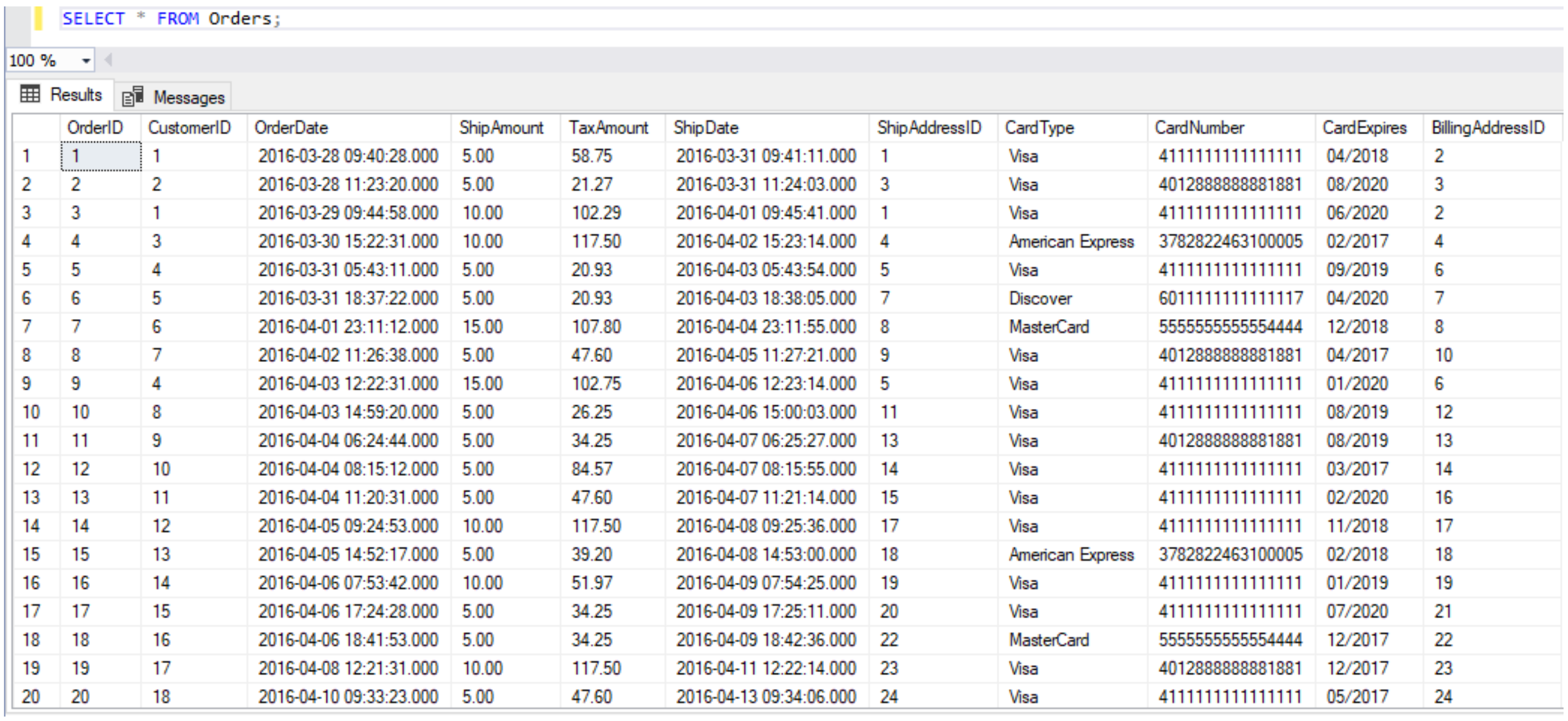
	CustomerID	EmailAddress	Password	FirstName	LastName	ShippingAddressID	BillingAddressID
1	1	allan.sherwood@yahoo.com	c44321e51ec184a2f739318639cec426de774451	Allan	Sherwood	1	2
2	2	barryz@gmail.com	d9e03c0b34c57d034edda004ec8bae5d53667e36	Barry	Zimmer	3	3
3	3	christineb@solarone.com	13ef4f968693bda97a898ece497da087b182808e	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
12	12	lpaprocki@hotmail.com	af27e4c5b48549d6dee6199e56834af1404f6169	Lenna	Paprocki	17	17
13	13	donette.foller@cox.net	356584acaa7164280c97b147f317ce0d76d5993c	Donette	Foller	18	18
14	14	simona@morasca.com	81133e972ea9b42831ef1fe7a846493526bfd3c3	Simona	Morasca	19	19
15	15	mitsue_tollner@yahoo.com	460420ee0d4f211f6be0bf7e7f352cf135cac579	Mitsue	Tollner	20	21
16	16	leota@hotmail.com	ceb92dd2b07d22fa39b9188306b9b04a78d80fe8	Leota	Dilliard	22	22
17	17	sage_wieser@cox.net	0c317d5469cda724d8b81ccd989179c99c3f2563	Sage	Wieser	23	23
18	18	kris@gmail.com	d79af41c43ea2a02afbbd66af85a70bed5b5757f	Kris	Marrier	24	24
19	19	minna_amigon@yahoo.com	c73630df7813e0cc99e02c639bdce1a954257230	Minna	Amigon	25	26
20	20	amaclead@gmail.com	295474f02de5da5862b837ca53de5e850f37f4c5	Abel	Maclead	27	27

Results | Messages

	CustomerID	EmailAddress	Password	FirstName	LastName	ShippingAddressID	BillingAddressID
467	467	dcomnick@cox.net	a120b51cdc9a9314acbcb789280bea3cf6c78762	Daniela	Comnick	493	493
468	468	cecilia_colaizzo@colaizzo.com	3c8de14b6483027a29c98d225adf6263130bf9e3	Cecilia	Colaizzo	494	494
469	469	leslie@cox.net	cb0025db0a3b134a67892175b82c36af5f64f720	Leslie	Threets	495	495
470	470	nan@koppinger.com	4a430cf63aaf01ae1018128c8d6f09da2035422c	Nan	Koppinger	496	496
471	471	idewar@dewar.com	fd1897a9f879531d5154b1c8e1ee0379191465ad	Izetta	Dewar	497	497
472	472	tegan.arceo@arceo.org	0771493979785eac2a2ca7584e7b3e3d6b8defc4	Tegan	Arceo	498	498
473	473	ruthann@hotmail.com	0156f3dfcd2489ad3e1cb4596249e44fa8724a44	Ruthann	Keener	499	499
474	474	joni_breland@cox.net	a786dc303c15b2cc2bf99949d2920b0a8dff9d57	Joni	Breland	500	500
475	475	vrentfro@cox.net	d15f8cc1bc1cb6a40339b52bbaed5922bd41bd0f	Vi	Rentfro	501	501
476	476	colette.kardas@yahoo.com	efdf759fde6c8479928774a99aefa69156585b4f	Colette	Kardas	502	502
477	477	malcolm_tromblay@cox.net	fb03a9b33a4e59fa4c2fc4e3b420f788df4caf39	Malcolm	Tromblay	503	503
478	478	ryan@cox.net	67c4bd033af41cd1a24c4df9f25de8b1051ce378	Ryan	Harnos	504	504
479	479	jess.chaffins@chaffins.org	7c3d3035e1d89f58f494c7e27b4d6afc00a74057	Jess	Chaffins	505	505
480	480	sbourbon@yahoo.com	e6bf7d0c9dd491dd289b356d3f7f285435a626e4	Sharen	Bourbon	506	506
481	481	nickolas_juvera@cox.net	77f602baf770e3618da53c0883d7c9bc82eca1f6	Nickolas	Juvera	507	507
482	482	gary_nunlee@nunlee.org	915e9617c8f0fb88e182a870a79129ae61c7efec	Gary	Nunlee	508	509
483	483	diane@cox.net	fba3f2a325a4e4eaf5992cbcc910ef91ecee3191	Diane	Devreese	510	510
484	484	roslyn.chavous@chavous.org	3bf8ab2bdbc781f2000472c4fd9d86d6433ccd35	Roslyn	Chavous	511	511
485	485	glory@yahoo.com	31c689d737359ce214d095f5b0bcf6de82e57f3d	Glory	Schieler	512	512

Query executed successfully. | DESKTOP-0E45JST\SQLEXPRESS ... | DESKTOP-0E45JST

The following are the details of Orders table using SELECT statement:

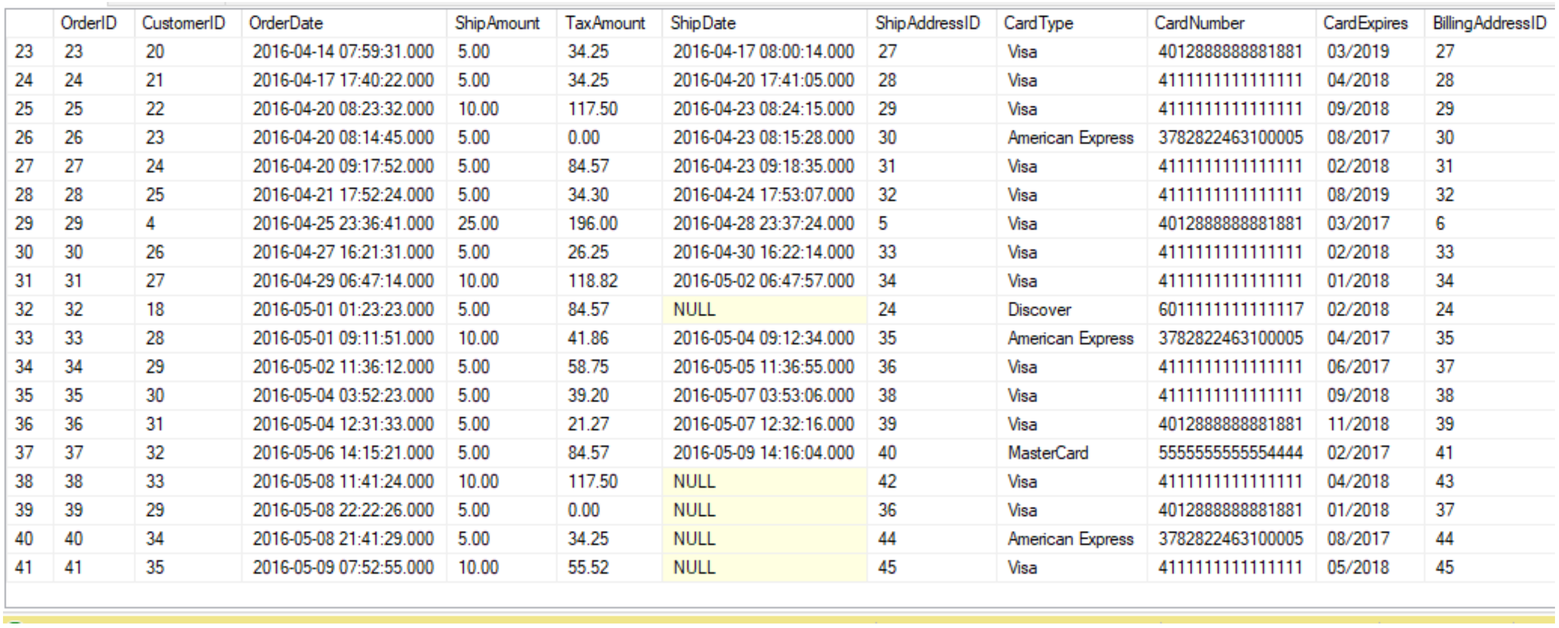


```
SELECT * FROM Orders;
```

100 %

Results | Messages

	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	3782822463100005	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
12	12	10	2016-04-04 08:15:12.000	5.00	84.57	2016-04-07 08:15:55.000	14	Visa	4111111111111111	03/2017	14
13	13	11	2016-04-04 11:20:31.000	5.00	47.60	2016-04-07 11:21:14.000	15	Visa	4111111111111111	02/2020	16
14	14	12	2016-04-05 09:24:53.000	10.00	117.50	2016-04-08 09:25:36.000	17	Visa	4111111111111111	11/2018	17
15	15	13	2016-04-05 14:52:17.000	5.00	39.20	2016-04-08 14:53:00.000	18	American Express	3782822463100005	02/2018	18
16	16	14	2016-04-06 07:53:42.000	10.00	51.97	2016-04-09 07:54:25.000	19	Visa	4111111111111111	01/2019	19
17	17	15	2016-04-06 17:24:28.000	5.00	34.25	2016-04-09 17:25:11.000	20	Visa	4111111111111111	07/2020	21
18	18	16	2016-04-06 18:41:53.000	5.00	34.25	2016-04-09 18:42:36.000	22	MasterCard	5555555555554444	12/2017	22
19	19	17	2016-04-08 12:21:31.000	10.00	117.50	2016-04-11 12:22:14.000	23	Visa	4012888888881881	12/2017	23
20	20	18	2016-04-10 09:33:23.000	5.00	47.60	2016-04-13 09:34:06.000	24	Visa	4111111111111111	05/2017	24



	OrderID	CustomerID	OrderDate	ShipAmount	TaxAmount	ShipDate	ShipAddressID	CardType	CardNumber	CardExpires	BillingAddressID
23	23	20	2016-04-14 07:59:31.000	5.00	34.25	2016-04-17 08:00:14.000	27	Visa	4012888888881881	03/2019	27
24	24	21	2016-04-17 17:40:22.000	5.00	34.25	2016-04-20 17:41:05.000	28	Visa	4111111111111111	04/2018	28
25	25	22	2016-04-20 08:23:32.000	10.00	117.50	2016-04-23 08:24:15.000	29	Visa	4111111111111111	09/2018	29
26	26	23	2016-04-20 08:14:45.000	5.00	0.00	2016-04-23 08:15:28.000	30	American Express	3782822463100005	08/2017	30
27	27	24	2016-04-20 09:17:52.000	5.00	84.57	2016-04-23 09:18:35.000	31	Visa	4111111111111111	02/2018	31
28	28	25	2016-04-21 17:52:24.000	5.00	34.30	2016-04-24 17:53:07.000	32	Visa	4111111111111111	08/2019	32
29	29	4	2016-04-25 23:36:41.000	25.00	196.00	2016-04-28 23:37:24.000	5	Visa	4012888888881881	03/2017	6
30	30	26	2016-04-27 16:21:31.000	5.00	26.25	2016-04-30 16:22:14.000	33	Visa	4111111111111111	02/2018	33
31	31	27	2016-04-29 06:47:14.000	10.00	118.82	2016-05-02 06:47:57.000	34	Visa	4111111111111111	01/2018	34
32	32	18	2016-05-01 01:23:23.000	5.00	84.57	NULL	24	Discover	6011111111111117	02/2018	24
33	33	28	2016-05-01 09:11:51.000	10.00	41.86	2016-05-04 09:12:34.000	35	American Express	3782822463100005	04/2017	35
34	34	29	2016-05-02 11:36:12.000	5.00	58.75	2016-05-05 11:36:55.000	36	Visa	4111111111111111	06/2017	37
35	35	30	2016-05-04 03:52:23.000	5.00	39.20	2016-05-07 03:53:06.000	38	Visa	4111111111111111	09/2018	38
36	36	31	2016-05-04 12:31:33.000	5.00	21.27	2016-05-07 12:32:16.000	39	Visa	4012888888881881	11/2018	39
37	37	32	2016-05-06 14:15:21.000	5.00	84.57	2016-05-09 14:16:04.000	40	MasterCard	5555555555554444	02/2017	41
38	38	33	2016-05-08 11:41:24.000	10.00	117.50	NULL	42	Visa	4111111111111111	04/2018	43
39	39	29	2016-05-08 22:22:26.000	5.00	0.00	NULL	36	Visa	4012888888881881	01/2018	37
40	40	34	2016-05-08 21:41:29.000	5.00	34.25	NULL	44	American Express	3782822463100005	08/2017	44
41	41	35	2016-05-09 07:52:55.000	10.00	55.52	NULL	45	Visa	4111111111111111	05/2018	45

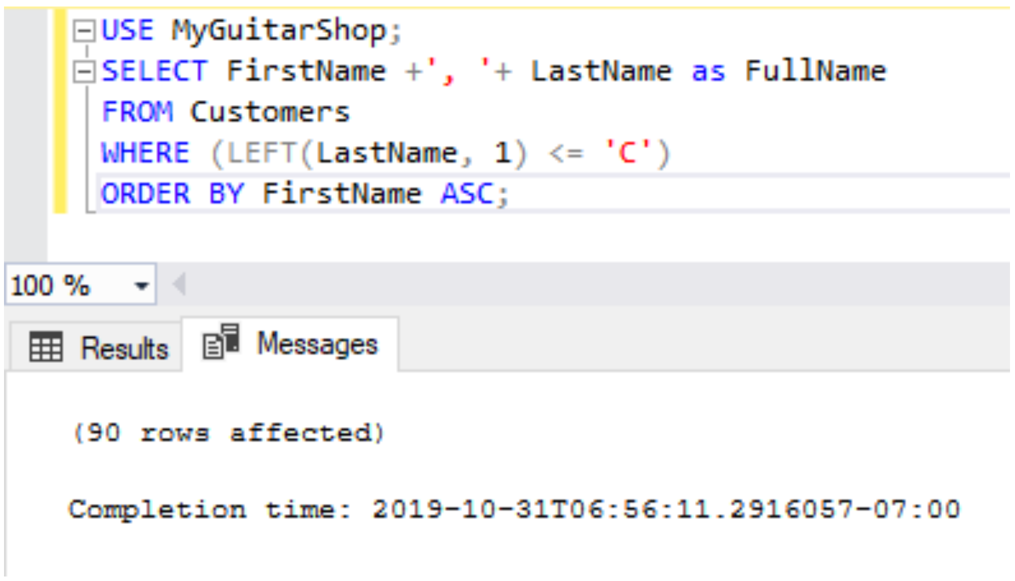
B. An Introduction to SQL

1.

The SQL command is as following.
It returns one column from the Customers table named FullName that joins the FirstName and LastName columns.
The format is first-name, last-name like John, Doe
Result set is sorted by first name in ascending sequence.
Return only the contacts whose last name begins with a letter from A to C.

```
USE MyGuitarShop;
SELECT FirstName +', '+ LastName as FullName
FROM Customers
WHERE (LEFT(LastName, 1) <= 'C')
ORDER BY FirstName ASC;
```

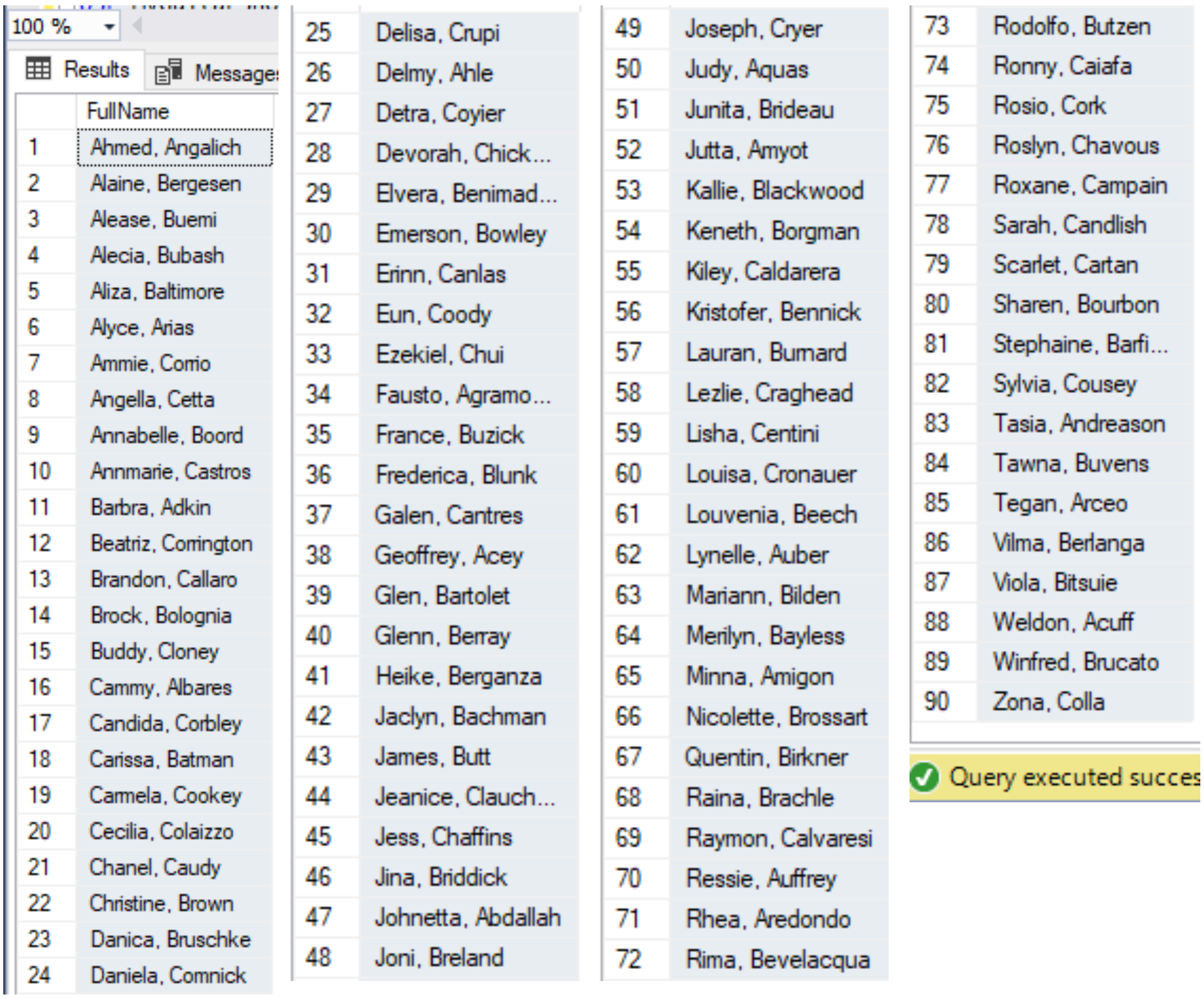
The query executed successfully:



```
USE MyGuitarShop;
SELECT FirstName +', '+ LastName as FullName
FROM Customers
WHERE (LEFT(LastName, 1) <= 'C')
ORDER BY FirstName ASC;
```

```
(90 rows affected)

Completion time: 2019-10-31T06:56:11.2916057-07:00
```

The query produced result set of 90 rows as below:



	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
12	Beatriz, Corrington
13	Brandon, Callaro
14	Brock, Bolognia
15	Buddy, Cloney
16	Cammy, Albares
17	Candida, Corbley
18	Carissa, Batman
19	Carmela, Cookey
20	Cecilia, Colaizzo
21	Chanel, Caudy
22	Christine, Brown
23	Danica, Bruschke
24	Daniela, Comnick
25	Delisa, Crupi
26	Delmy, Ahle
27	Detra, Coyier
28	Devorah, Chick...
29	Elvera, Benimad...
30	Emerson, Bowley
31	Erinn, Canlas
32	Eun, Coody
33	Ezekiel, Chui
34	Fausto, Agramo...
35	France, Buzick
36	Frederica, Blunk
37	Galen, Cantres
38	Geoffrey, Acey
39	Glen, Bartolet
40	Glenn, Berray
41	Heike, Berganza
42	Jaclyn, Bachman
43	James, Butt
44	Jeanice, Clauch...
45	Jess, Chaffins
46	Jina, Briddick
47	Johnetta, Abdallah
48	Joni, Breland
49	Joseph, Cryer
50	Judy, Aquas
51	Junita, Brideau
52	Jutta, Amyot
53	Kallie, Blackwood
54	Keneth, Borgman
55	Kiley, Caldarera
56	Kristofer, Bennick
57	Lauran, Burnard
58	Lezlie, Craghead
59	Lisha, Centini
60	Louisa, Cronauer
61	Louvenia, Beech
62	Lynelle, Auber
63	Mariann, Bilden
64	Merilyn, Bayless
65	Minna, Amigon
66	Nicolette, Brossart
67	Quentin, Birkner
68	Raina, Brachle
69	Raymon, Calvaresi
70	Ressie, Auffrey
71	Rhea, Aredondo
72	Rima, Bevelacqua
73	Rodolfo, Butzen
74	Ronny, Caiafa
75	Rosio, Cork
76	Roslyn, Chavous
77	Roxane, Campain
78	Sarah, Candlish
79	Scarlet, Cartan
80	Sharen, Bourbon
81	Stephaine, Barfi...
82	Sylvia, Cousey
83	Tasia, Andreason
84	Tawna, Buvens
85	Tegan, Arceo
86	Vilma, Berlanga
87	Viola, Bitsuie
88	Weldon, Acuff
89	Winfred, Brucato
90	Zona, Colla

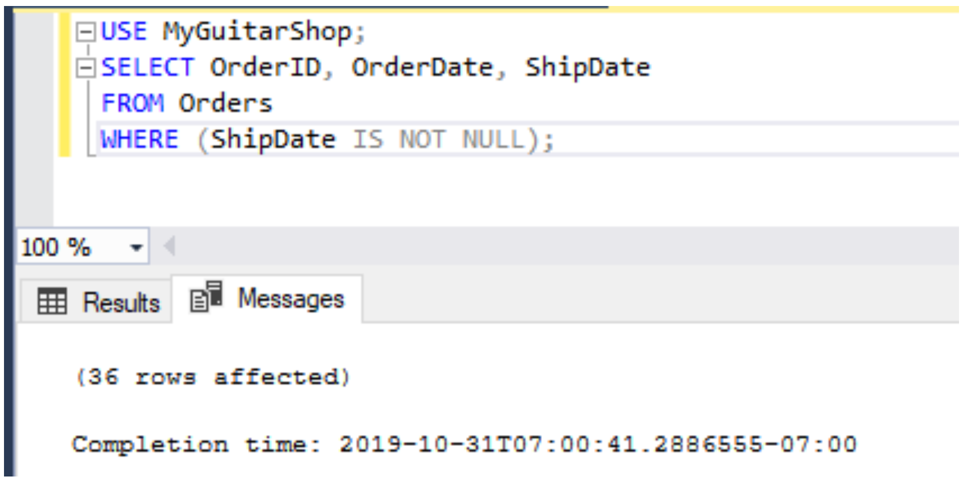
Query executed succes

2.

The SQL command is as following. It return only the rows where the ShipDate column does not contain a null value.

```
USE MyGuitarShop;
SELECT OrderID, OrderDate, ShipDate
FROM Orders
WHERE (ShipDate IS NOT NULL);
```

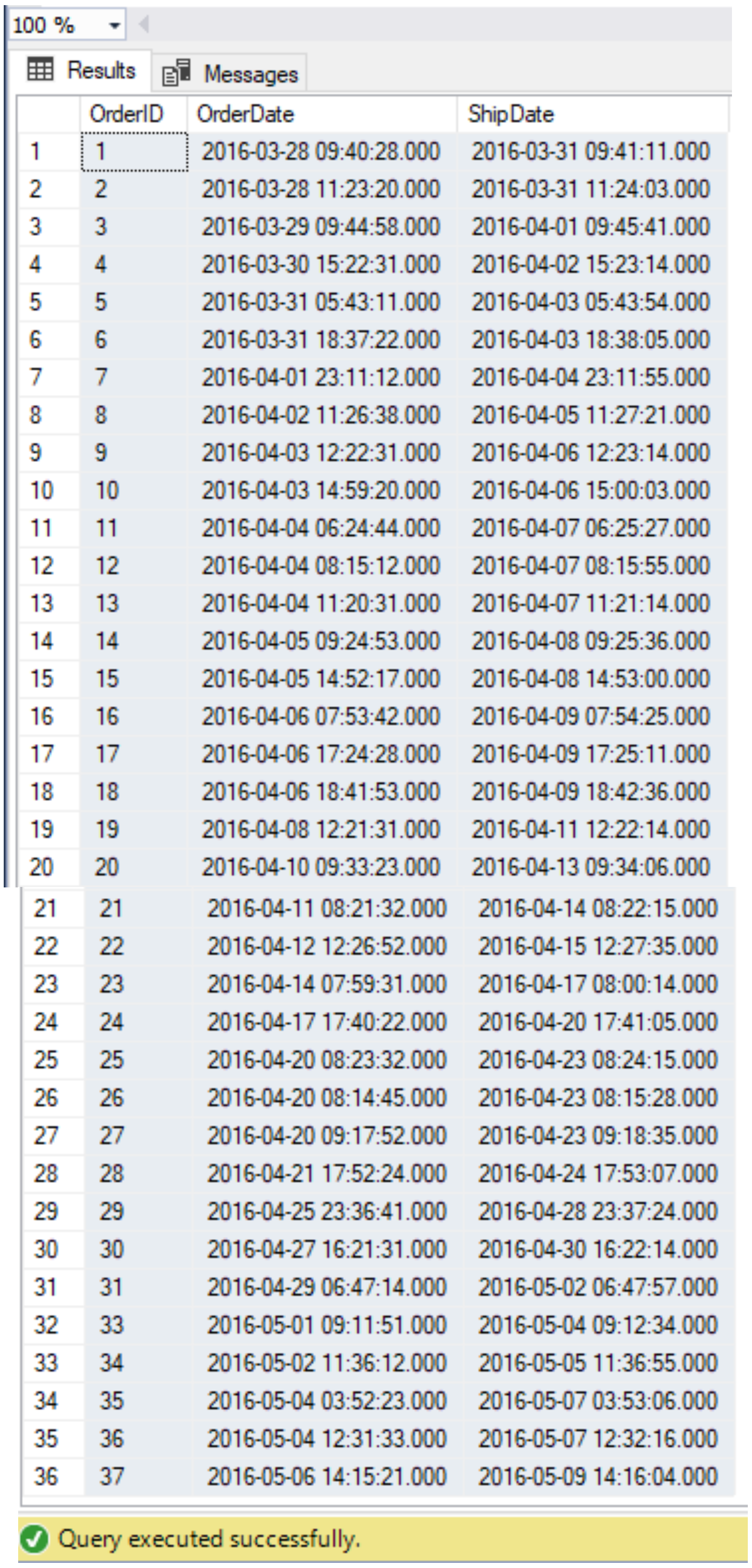
The query executed successfully:



```
USE MyGuitarShop;
SELECT OrderID, OrderDate, ShipDate
FROM Orders
WHERE (ShipDate IS NOT NULL);
```

```
(36 rows affected)

Completion time: 2019-10-31T07:00:41.2886555-07:00
```

The query produced result set of 36 rows as below:



100 %

Results | Messages

	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
12	12	2016-04-04 08:15:12.000	2016-04-07 08:15:55.000
13	13	2016-04-04 11:20:31.000	2016-04-07 11:21:14.000
14	14	2016-04-05 09:24:53.000	2016-04-08 09:25:36.000
15	15	2016-04-05 14:52:17.000	2016-04-08 14:53:00.000
16	16	2016-04-06 07:53:42.000	2016-04-09 07:54:25.000
17	17	2016-04-06 17:24:28.000	2016-04-09 17:25:11.000
18	18	2016-04-06 18:41:53.000	2016-04-09 18:42:36.000
19	19	2016-04-08 12:21:31.000	2016-04-11 12:22:14.000
20	20	2016-04-10 09:33:23.000	2016-04-13 09:34:06.000
21	21	2016-04-11 08:21:32.000	2016-04-14 08:22:15.000
22	22	2016-04-12 12:26:52.000	2016-04-15 12:27:35.000
23	23	2016-04-14 07:59:31.000	2016-04-17 08:00:14.000
24	24	2016-04-17 17:40:22.000	2016-04-20 17:41:05.000
25	25	2016-04-20 08:23:32.000	2016-04-23 08:24:15.000
26	26	2016-04-20 08:14:45.000	2016-04-23 08:15:28.000
27	27	2016-04-20 09:17:52.000	2016-04-23 09:18:35.000
28	28	2016-04-21 17:52:24.000	2016-04-24 17:53:07.000
29	29	2016-04-25 23:36:41.000	2016-04-28 23:37:24.000
30	30	2016-04-27 16:21:31.000	2016-04-30 16:22:14.000
31	31	2016-04-29 06:47:14.000	2016-05-02 06:47:57.000
32	33	2016-05-01 09:11:51.000	2016-05-04 09:12:34.000
33	34	2016-05-02 11:36:12.000	2016-05-05 11:36:55.000
34	35	2016-05-04 03:52:23.000	2016-05-07 03:53:06.000
35	36	2016-05-04 12:31:33.000	2016-05-07 12:32:16.000
36	37	2016-05-06 14:15:21.000	2016-05-09 14:16:04.000

Query executed successfully.

C. The essential SQL skills

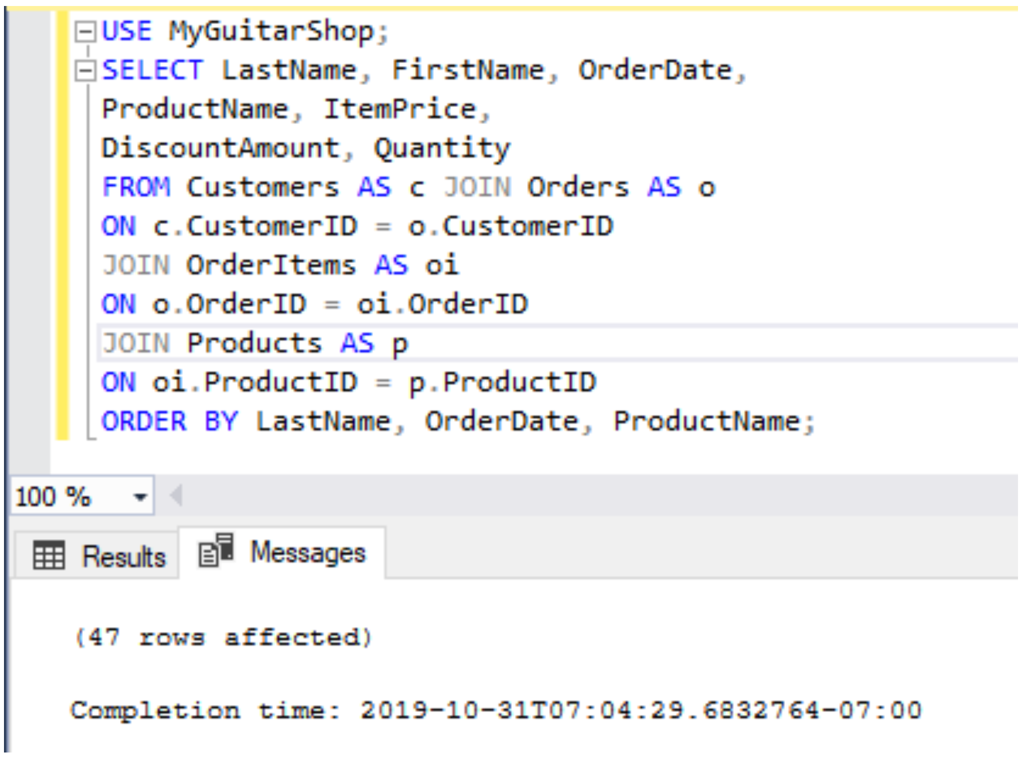
1.

The SQL command is as following.

```
USE MyGuitarShop;
SELECT LastName, FirstName, OrderDate, ProductName, ItemPrice,
        DiscountAmount, Quantity
FROM Customers AS c
JOIN Orders AS o
  ON c.CustomerID = o.CustomerID
JOIN OrderItems AS oi
  ON o.OrderID = oi.OrderID
JOIN Products AS p
  ON oi.ProductID = p.ProductID
ORDER BY LastName, OrderDate, ProductName;
```

This query joins the Customers, Orders, OrderItems, and Products tables. and use aliases for the tables.

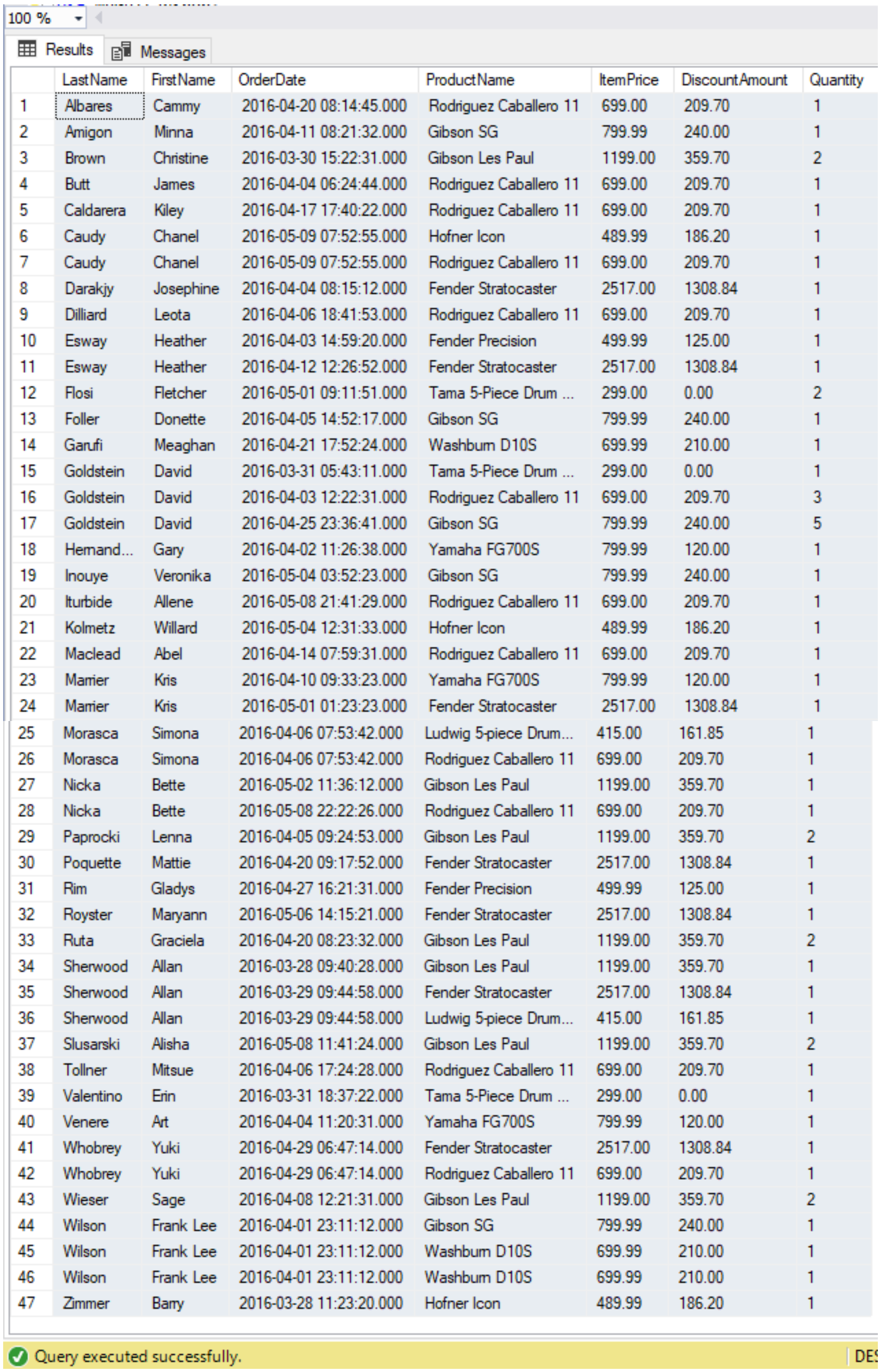
The query executed successfully:



```
USE MyGuitarShop;
SELECT LastName, FirstName, OrderDate,
ProductName, ItemPrice,
DiscountAmount, Quantity
FROM Customers AS c JOIN Orders AS o
ON c.CustomerID = o.CustomerID
JOIN OrderItems AS oi
ON o.OrderID = oi.OrderID
JOIN Products AS p
ON oi.ProductID = p.ProductID
ORDER BY LastName, OrderDate, ProductName;
```

```
(47 rows affected)

Completion time: 2019-10-31T07:04:29.6832764-07:00
```

The query produced result set of 47 rows as below:



	LastName	FirstName	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
12	Flosi	Fletcher	2016-05-01 09:11:51.000	Tama 5-Piece Drum ...	299.00	0.00	2
13	Foller	Donette	2016-04-05 14:52:17.000	Gibson SG	799.99	240.00	1
14	Garufi	Meaghan	2016-04-21 17:52:24.000	Washburn D10S	699.99	210.00	1
15	Goldstein	David	2016-03-31 05:43:11.000	Tama 5-Piece Drum ...	299.00	0.00	1
16	Goldstein	David	2016-04-03 12:22:31.000	Rodriguez Caballero 11	699.00	209.70	3
17	Goldstein	David	2016-04-25 23:36:41.000	Gibson SG	799.99	240.00	5
18	Hernand...	Gary	2016-04-02 11:26:38.000	Yamaha FG700S	799.99	120.00	1
19	Inouye	Veronika	2016-05-04 03:52:23.000	Gibson SG	799.99	240.00	1
20	Iturbide	Allene	2016-05-08 21:41:29.000	Rodriguez Caballero 11	699.00	209.70	1
21	Kolmetz	Willard	2016-05-04 12:31:33.000	Hofner Icon	489.99	186.20	1
22	Maclead	Abel	2016-04-14 07:59:31.000	Rodriguez Caballero 11	699.00	209.70	1
23	Marrier	Kris	2016-04-10 09:33:23.000	Yamaha FG700S	799.99	120.00	1
24	Marrier	Kris	2016-05-01 01:23:23.000	Fender Stratocaster	2517.00	1308.84	1
25	Morasca	Simona	2016-04-06 07:53:42.000	Ludwig 5-piece Drum...	415.00	161.85	1
26	Morasca	Simona	2016-04-06 07:53:42.000	Rodriguez Caballero 11	699.00	209.70	1
27	Nicka	Bette	2016-05-02 11:36:12.000	Gibson Les Paul	1199.00	359.70	1
28	Nicka	Bette	2016-05-08 22:22:26.000	Rodriguez Caballero 11	699.00	209.70	1
29	Paprocki	Lenna	2016-04-05 09:24:53.000	Gibson Les Paul	1199.00	359.70	2
30	Poquette	Mattie	2016-04-20 09:17:52.000	Fender Stratocaster	2517.00	1308.84	1
31	Rim	Gladys	2016-04-27 16:21:31.000	Fender Precision	499.99	125.00	1
32	Royster	Maryann	2016-05-06 14:15:21.000	Fender Stratocaster	2517.00	1308.84	1
33	Ruta	Graciela	2016-04-20 08:23:32.000	Gibson Les Paul	1199.00	359.70	2
34	Sherwood	Allan	2016-03-28 09:40:28.000	Gibson Les Paul	1199.00	359.70	1
35	Sherwood	Allan	2016-03-29 09:44:58.000	Fender Stratocaster	2517.00	1308.84	1
36	Sherwood	Allan	2016-03-29 09:44:58.000	Ludwig 5-piece Drum...	415.00	161.85	1
37	Slusarski	Alisha	2016-05-08 11:41:24.000	Gibson Les Paul	1199.00	359.70	2
38	Tollner	Mitsue	2016-04-06 17:24:28.000	Rodriguez Caballero 11	699.00	209.70	1
39	Valentino	Erin	2016-03-31 18:37:22.000	Tama 5-Piece Drum ...	299.00	0.00	1
40	Venere	Art	2016-04-04 11:20:31.000	Yamaha FG700S	799.99	120.00	1
41	Whobrey	Yuki	2016-04-29 06:47:14.000	Fender Stratocaster	2517.00	1308.84	1
42	Whobrey	Yuki	2016-04-29 06:47:14.000	Rodriguez Caballero 11	699.00	209.70	1
43	Wieser	Sage	2016-04-08 12:21:31.000	Gibson Les Paul	1199.00	359.70	2
44	Wilson	Frank Lee	2016-04-01 23:11:12.000	Gibson SG	799.99	240.00	1
45	Wilson	Frank Lee	2016-04-01 23:11:12.000	Washburn D10S	699.99	210.00	1
46	Wilson	Frank Lee	2016-04-01 23:11:12.000	Washburn D10S	699.99	210.00	1
47	Zimmer	Barry	2016-03-28 11:23:20.000	Hofner Icon	489.99	186.20	1

Query executed successfully.

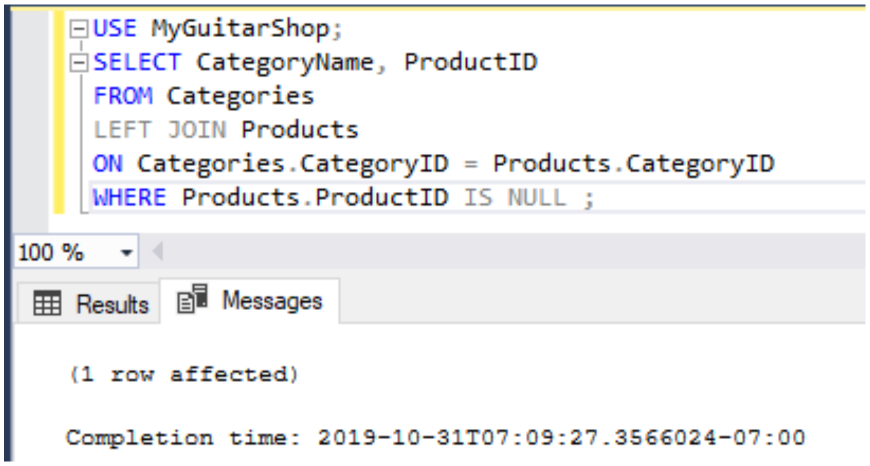
2.

The SQL command is as following.

```
USE MyGuitarShop;
SELECT CategoryName, ProductID
FROM Categories
LEFT JOIN Products
        ON Categories.CategoryID = Products.CategoryID
WHERE.Products.ProductID IS NULL ;
```

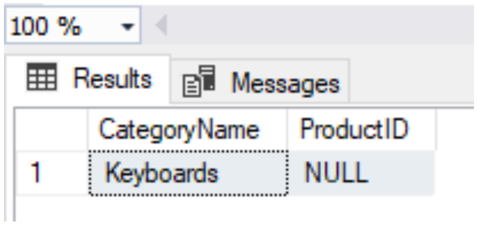
It returns these two columns: CategoryName and ProductID. Return one row for each category that has never been used. An outer join is used and only return rows where the ProductID column contains a null value.

The query executed successfully and produced result set of 1 row as below:



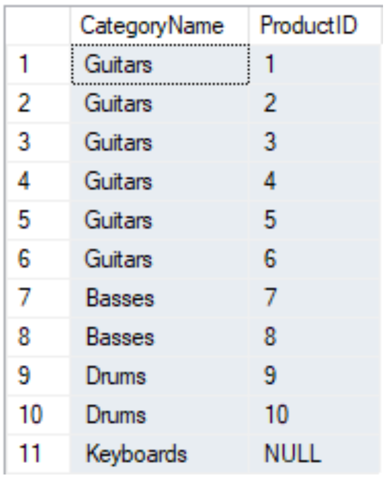
```
USE MyGuitarShop;
SELECT CategoryName, ProductID
FROM Categories
LEFT JOIN Products
ON Categories.CategoryID = Products.CategoryID
WHERE Products.ProductID IS NULL ;
```

```
(1 row affected)

Completion time: 2019-10-31T07:09:27.3566024-07:00
```



	CategoryName	ProductID
1	Keyboards	NULL

As shown below, Keyboards is the only unused category.



	CategoryName	ProductID
1	Guitars	1
2	Guitars	2
3	Guitars	3
4	Guitars	4
5	Guitars	5
6	Guitars	6
7	Basses	7
8	Basses	8
9	Drums	9
10	Drums	10
11	Keyboards	NULL

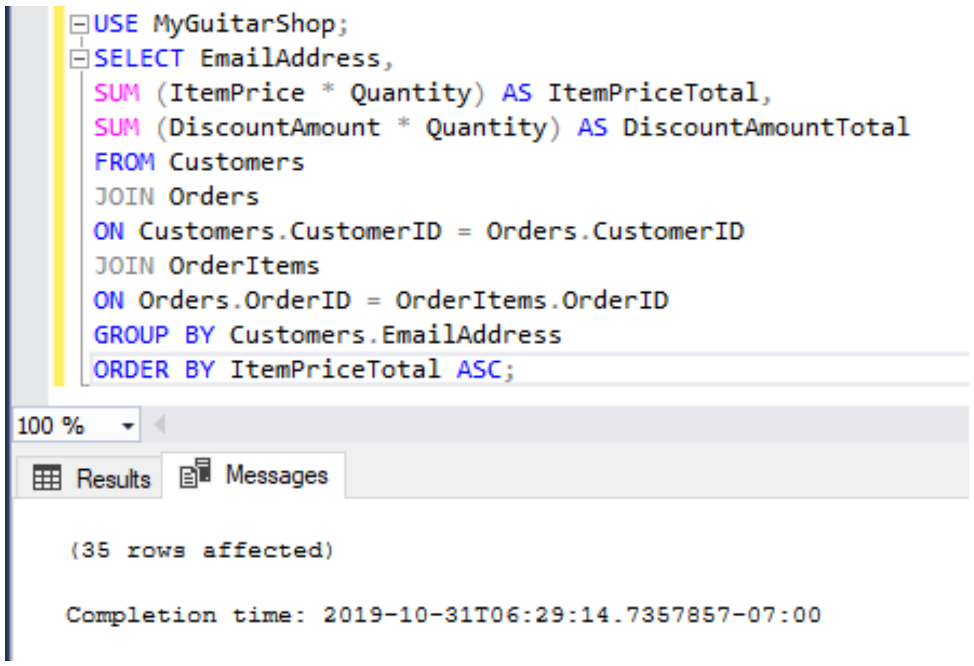
3.

The SQL command is as following.

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

It returns one row for each customer that has orders with EmailAddress, sum of the item price multiplied by the quantity , and the sum of the discount amount multiplied by the quantity.

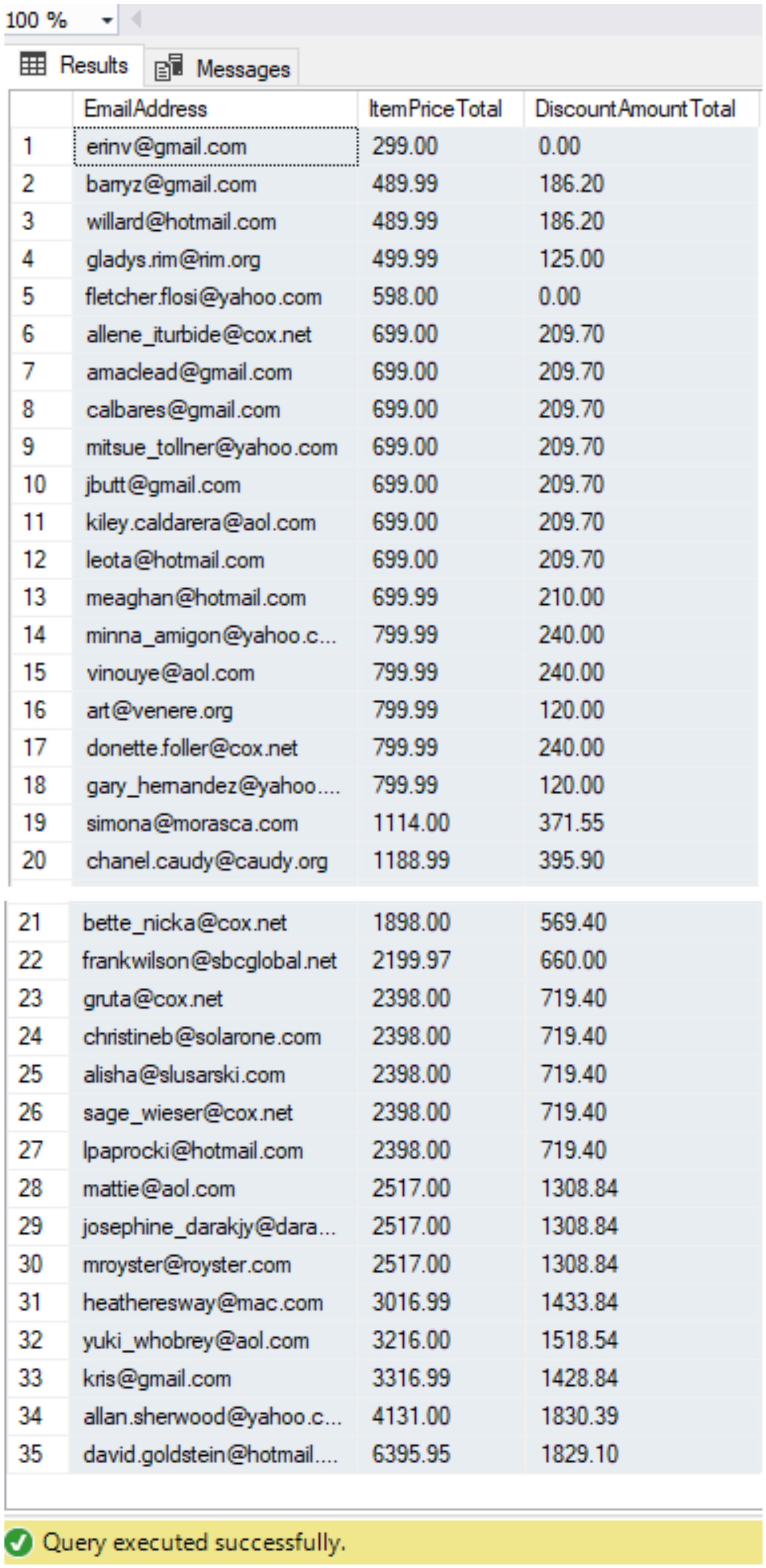
The query executed successfully:



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

```
(35 rows affected)

Completion time: 2019-10-31T06:29:14.7357857-07:00
```

The query produced result set of 35 rows as below:



	EmailAddress	ItemPriceTotal	DiscountAmountTotal
1	erinv@gmail.com	299.00	0.00
2	barryz@gmail.com	489.99	186.20
3	willard@hotmail.com	489.99	186.20
4	gladys.rim@rim.org	499.99	125.00
5	fletcher.flosi@yahoo.com	598.00	0.00
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
12	leota@hotmail.com	699.00	209.70
13	meaghan@hotmail.com	699.99	210.00
14	minna_amigon@yahoo.c...	799.99	240.00
15	vinouye@aol.com	799.99	240.00
16	art@venere.org	799.99	120.00
17	donette.foller@cox.net	799.99	240.00
18	gary_hernandez@yahoo....	799.99	120.00
19	simona@morasca.com	1114.00	371.55
20	chanel.caudy@caudy.org	1188.99	395.90
21	bette_nicka@cox.net	1898.00	569.40
22	frankwilson@sbcglobal.net	2199.97	660.00
23	gruta@cox.net	2398.00	719.40
24	christineb@solarone.com	2398.00	719.40
25	alisha@slusarski.com	2398.00	719.40
26	sage_wieser@cox.net	2398.00	719.40
27	lpaprocki@hotmail.com	2398.00	719.40
28	mattie@aol.com	2517.00	1308.84
29	josephine_darakjy@dara...	2517.00	1308.84
30	mroyster@royster.com	2517.00	1308.84
31	heatheresway@mac.com	3016.99	1433.84
32	yuki_whobrey@aol.com	3216.00	1518.54
33	kris@gmail.com	3316.99	1428.84
34	allan.sherwood@yahoo.c...	4131.00	1830.39
35	david.goldstein@hotmail....	6395.95	1829.10

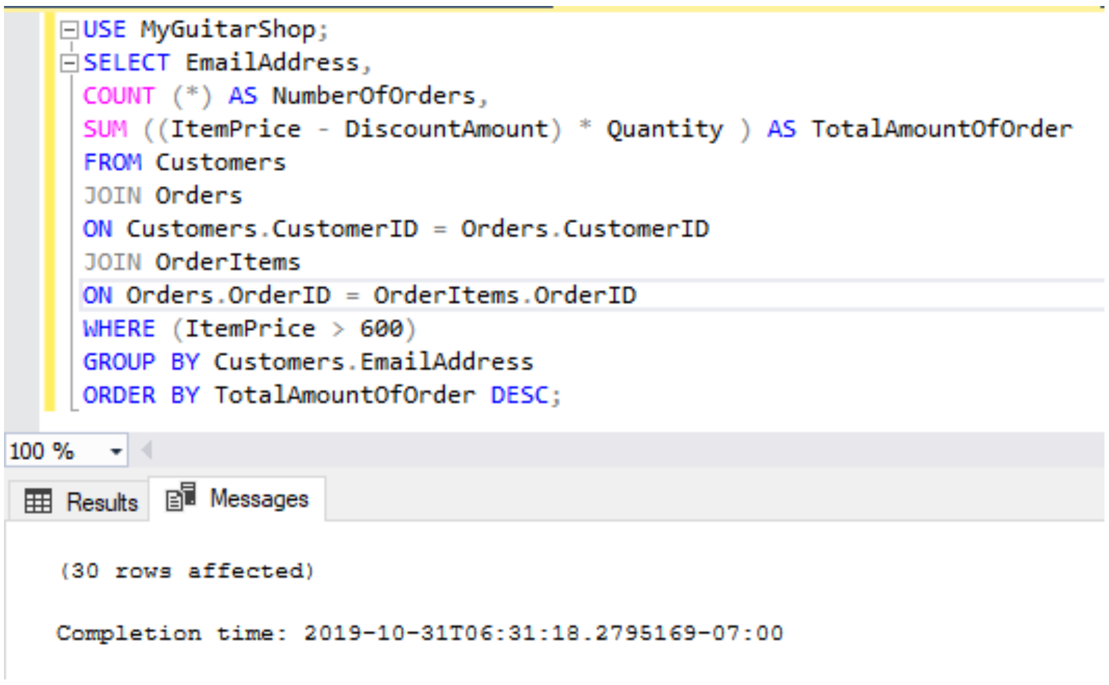
Query executed successfully.

4.

The SQL command is as following. It returns one row for each customer and only those rows where items have a more than 600 ItemPrice value.

```
USE MyGuitarShop;
SELECT EmailAddress,
          COUNT (*) AS NumberOfOrders,
          SUM ((ItemPrice - DiscountAmount) * Quantity )
               AS TotalAmountOfOrder
FROM Customers
JOIN Orders
   ON Customers.CustomerID = Orders.CustomerID
JOIN OrderItems
   ON Orders.OrderID = OrderItems.OrderID
WHERE (ItemPrice > 600)
GROUP BY Customers.EmailAddress
ORDER BY TotalAmountOfOrder DESC;
```

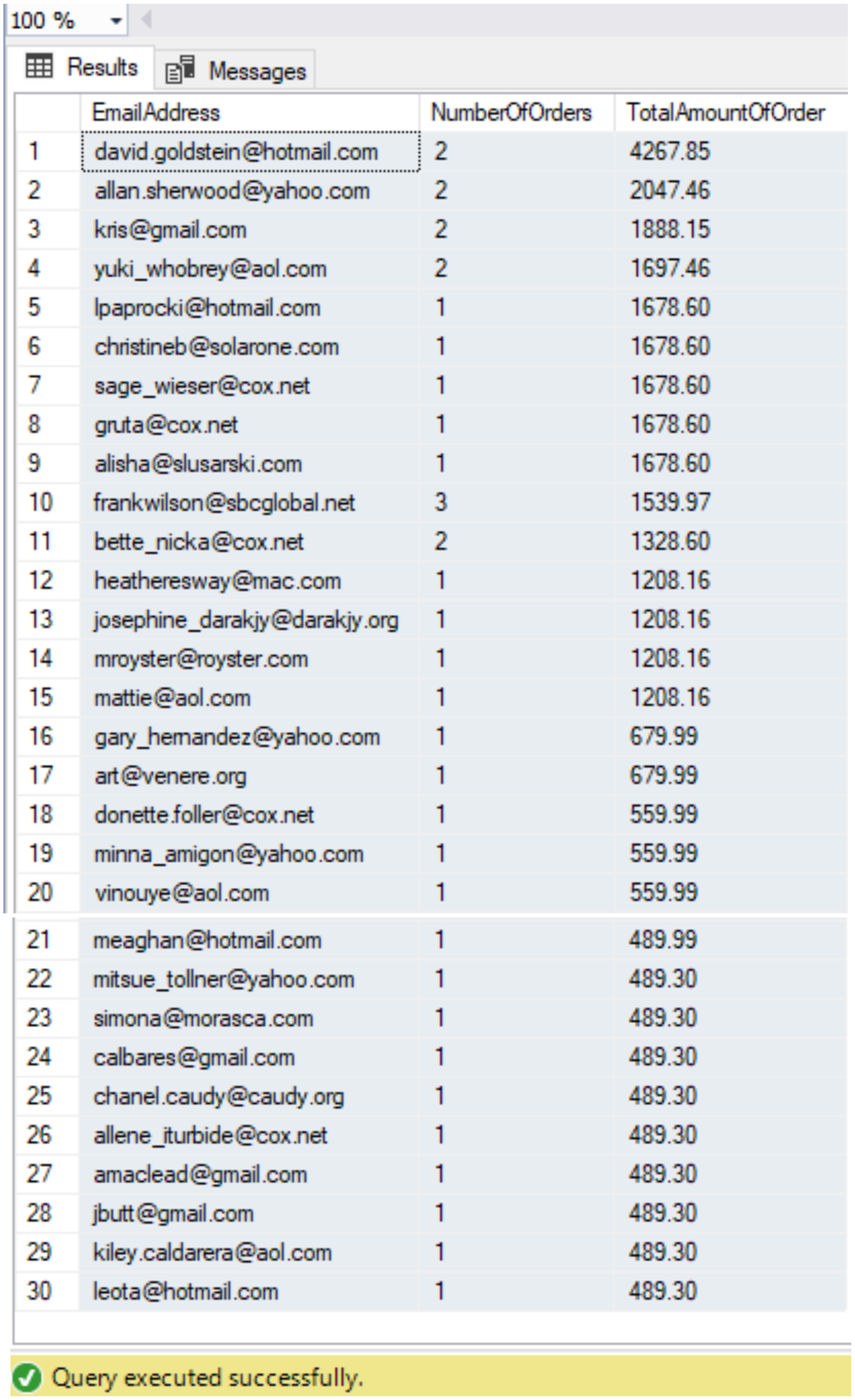
The query executed successfully:



```
USE MyGuitarShop;
SELECT EmailAddress,
COUNT (*) AS NumberOfOrders,
SUM ((ItemPrice - DiscountAmount) * Quantity ) AS TotalAmountOfOrder
FROM Customers
JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
JOIN OrderItems
ON Orders.OrderID = OrderItems.OrderID
WHERE (ItemPrice > 600)
GROUP BY Customers.EmailAddress
ORDER BY TotalAmountOfOrder DESC;
```

```
(30 rows affected)

Completion time: 2019-10-31T06:31:18.2795169-07:00
```

The query produced result set of 30 rows as below:



	EmailAddress	NumberOfOrders	TotalAmountOfOrder
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
12	heatheresway@mac.com	1	1208.16
13	josephine_darakjy@darakjy.org	1	1208.16
14	mroyster@royster.com	1	1208.16
15	mattie@aol.com	1	1208.16
16	gary_hernandez@yahoo.com	1	679.99
17	art@venere.org	1	679.99
18	donette.foller@cox.net	1	559.99
19	minna_amigon@yahoo.com	1	559.99
20	vinouye@aol.com	1	559.99
21	meaghan@hotmail.com	1	489.99
22	mitsue_tollner@yahoo.com	1	489.30
23	simona@morasca.com	1	489.30
24	calbares@gmail.com	1	489.30
25	chanel.caudy@caudy.org	1	489.30
26	allene_iturbide@cox.net	1	489.30
27	amaclead@gmail.com	1	489.30
28	jbutt@gmail.com	1	489.30
29	kiley.caldarera@aol.com	1	489.30
30	leota@hotmail.com	1	489.30

Query executed successfully.

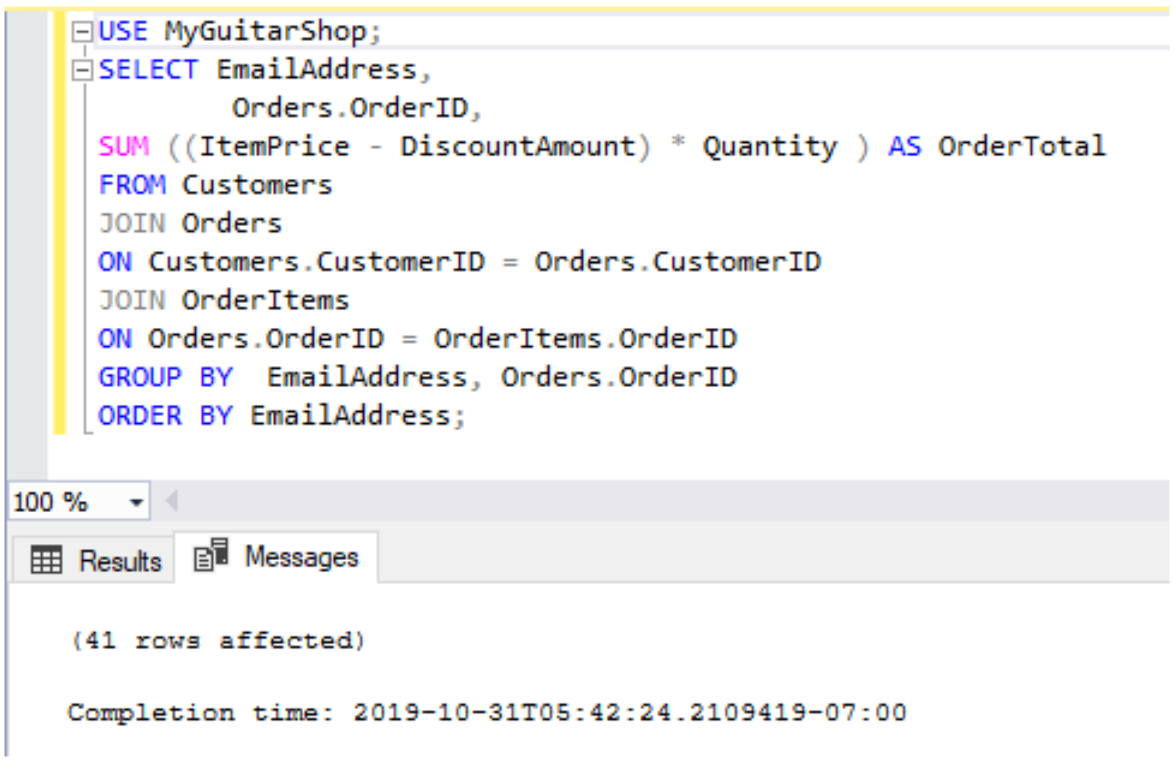
5. (1)

The SQL command is as following.

```
USE MyGuitarShop;
SELECT EmailAddress,
            Orders.OrderID,
            SUM ((ItemPrice - DiscountAmount) * Quantity ) AS OrderTotal
FROM Customers
JOIN Orders
   ON Customers.CustomerID = Orders.CustomerID
JOIN OrderItems
   ON Orders.OrderID = OrderItems.OrderID
GROUP BY  EmailAddress, Orders.OrderID
ORDER BY EmailAddress;
```

The query inner join three tables, group by EmailAddress and OrderID. Aggregate function SUM is used to produce the OrderTotal.

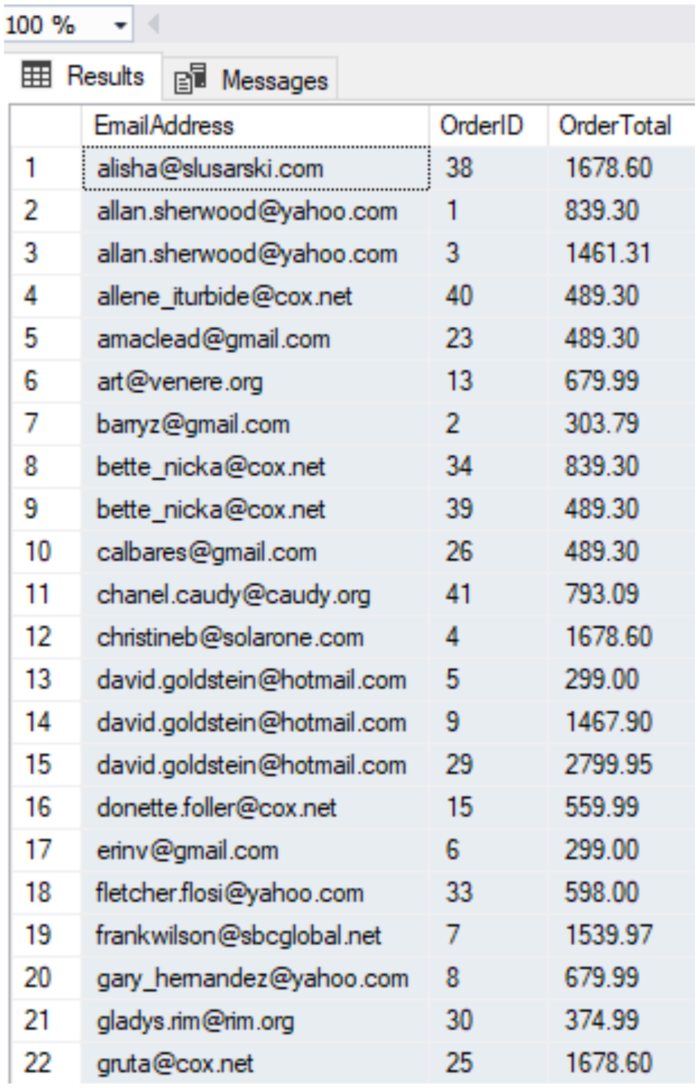
The query executed successfully:

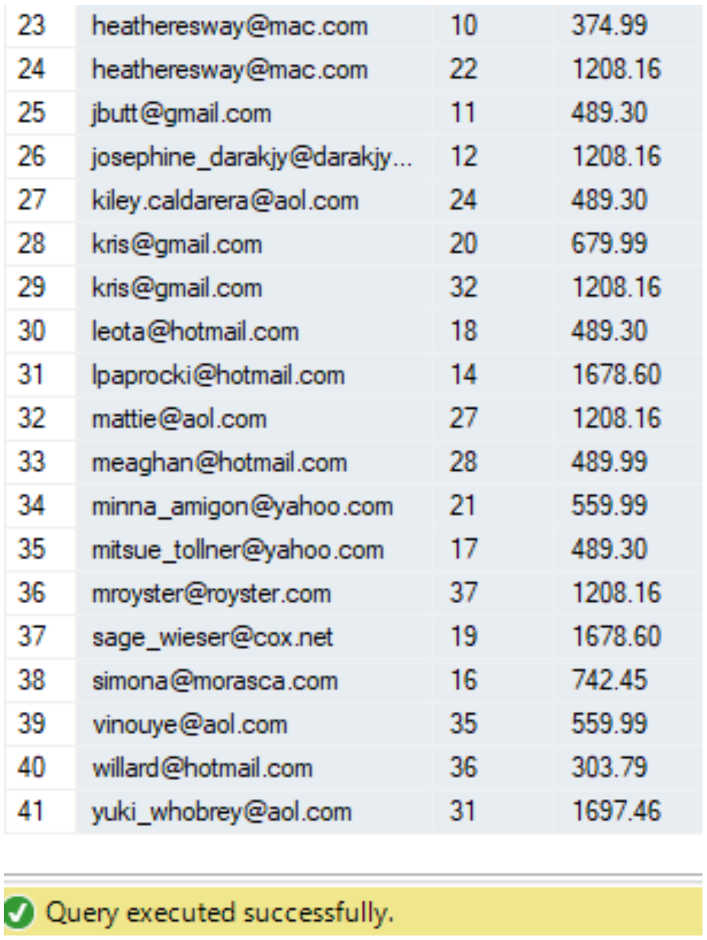


```
USE MyGuitarShop;
SELECT EmailAddress,
        Orders.OrderID,
SUM ((ItemPrice - DiscountAmount) * Quantity ) AS OrderTotal
FROM Customers
JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
JOIN OrderItems
ON Orders.OrderID = OrderItems.OrderID
GROUP BY  EmailAddress, Orders.OrderID
ORDER BY EmailAddress;
```

```
(41 rows affected)

Completion time: 2019-10-31T05:42:24.2109419-07:00
```

The query produced result set of 41 rows as below:





	EmailAddress	OrderID	OrderTotal
1	alisha@slusarski.com	38	1678.60
2	allan.sherwood@yahoo.com	1	839.30
3	allan.sherwood@yahoo.com	3	1461.31
4	allene_iturbide@cox.net	40	489.30
5	amaclead@gmail.com	23	489.30
6	art@venere.org	13	679.99
7	barryz@gmail.com	2	303.79
8	bette_nicka@cox.net	34	839.30
9	bette_nicka@cox.net	39	489.30
10	calbares@gmail.com	26	489.30
11	chanel.caudy@caudy.org	41	793.09
12	christineb@solarone.com	4	1678.60
13	david.goldstein@hotmail.com	5	299.00
14	david.goldstein@hotmail.com	9	1467.90
15	david.goldstein@hotmail.com	29	2799.95
16	donette.foller@cox.net	15	559.99
17	erinv@gmail.com	6	299.00
18	fletcher.flosi@yahoo.com	33	598.00
19	frankwilson@sbcglobal.net	7	1539.97
20	gary_hernandez@yahoo.com	8	679.99
21	gladys.rim@rim.org	30	374.99
22	gruta@cox.net	25	1678.60
23	heatheresway@mac.com	10	374.99
24	heatheresway@mac.com	22	1208.16
25	jbutt@gmail.com	11	489.30
26	josephine_darakjy@darakjy...	12	1208.16
27	kiley.caldarera@aol.com	24	489.30
28	kris@gmail.com	20	679.99
29	kris@gmail.com	32	1208.16
30	leota@hotmail.com	18	489.30
31	lpaprocki@hotmail.com	14	1678.60
32	mattie@aol.com	27	1208.16
33	meaghan@hotmail.com	28	489.99
34	minna_amigon@yahoo.com	21	559.99
35	mitsue_tollner@yahoo.com	17	489.30
36	mroyster@royster.com	37	1208.16
37	sage_wieser@cox.net	19	1678.60
38	simona@morasca.com	16	742.45
39	vinouye@aol.com	35	559.99
40	willard@hotmail.com	36	303.79
41	yuki_whobrey@aol.com	31	1697.46

✓ Query executed successfully.

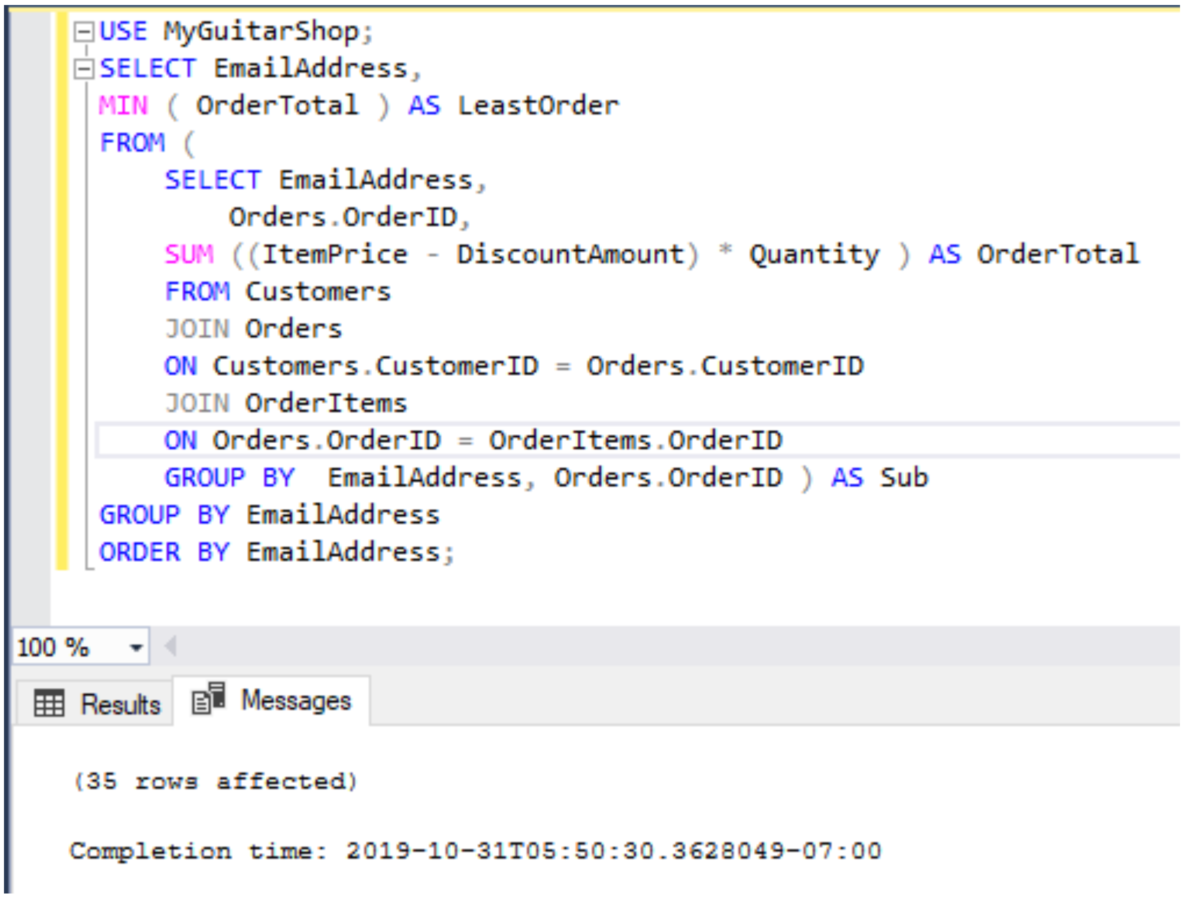
5. (2)

The SQL command is as following.

```
USE MyGuitarShop;
SELECT EmailAddress,
        MIN ( OrderTotal ) AS LeastOrder
FROM (
        SELECT EmailAddress,
                Orders.OrderID,
                SUM ((ItemPrice - DiscountAmount) * Quantity ) AS OrderTotal
        FROM Customers
        JOIN Orders
           ON Customers.CustomerID = Orders.CustomerID
        JOIN OrderItems
           ON Orders.OrderID = OrderItems.OrderID
        GROUP BY  EmailAddress, Orders.OrderID ) AS Sub
GROUP BY EmailAddress
ORDER BY EmailAddress;
```

The query use the SELECT query of part (a) and group by EmailAddress. Aggregate function MIN is used to produce the LeastOrder.

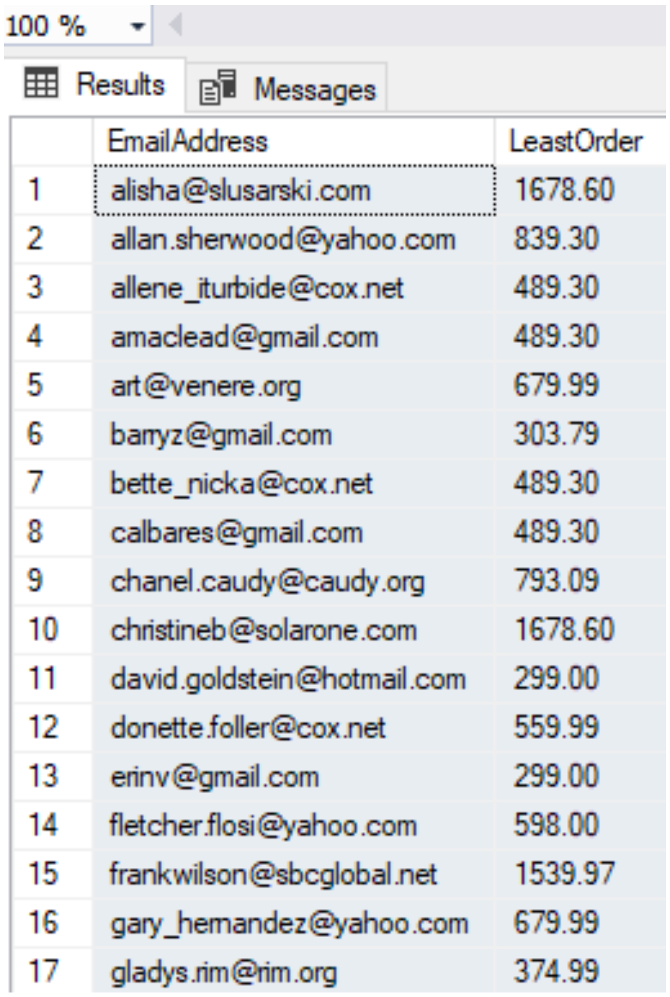
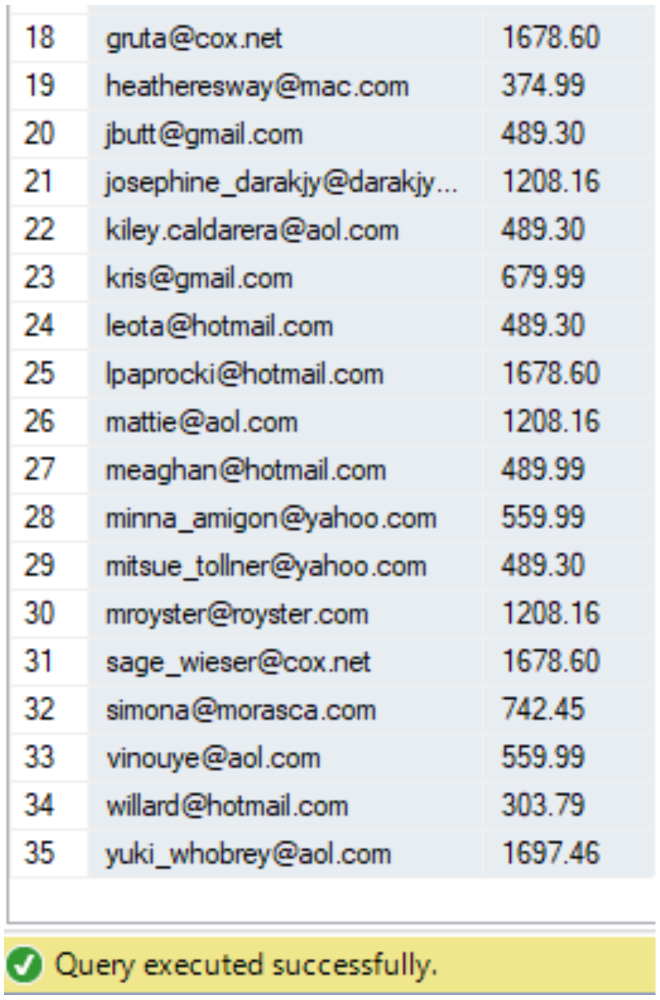
The query executed successfully as below:



```
USE MyGuitarShop;
SELECT EmailAddress,
MIN ( OrderTotal ) AS LeastOrder
FROM (
    SELECT EmailAddress,
        Orders.OrderID,
    SUM ((ItemPrice - DiscountAmount) * Quantity ) AS OrderTotal
    FROM Customers
    JOIN Orders
    ON Customers.CustomerID = Orders.CustomerID
    JOIN OrderItems
    ON Orders.OrderID = OrderItems.OrderID
    GROUP BY  EmailAddress, Orders.OrderID ) AS Sub
GROUP BY EmailAddress
ORDER BY EmailAddress;
```

```
(35 rows affected)

Completion time: 2019-10-31T05:50:30.3628049-07:00
```

The query produced result set of 35 rows as below:

	EmailAddress	LeastOrder
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
12	donette.foller@cox.net	559.99
13	erinv@gmail.com	299.00
14	fletcher.flosi@yahoo.com	598.00
15	frankwilson@sbcglobal.net	1539.97
16	gary_hernandez@yahoo.com	679.99
17	gladys.rim@rim.org	374.99
18	gruta@cox.net	1678.60
19	heatheresway@mac.com	374.99
20	jbutt@gmail.com	489.30
21	josephine_darakjy@darakjy...	1208.16
22	kiley.caldarera@aol.com	489.30
23	kris@gmail.com	679.99
24	leota@hotmail.com	489.30
25	lpaprocki@hotmail.com	1678.60
26	mattie@aol.com	1208.16
27	meaghan@hotmail.com	489.99
28	minna_amigon@yahoo.com	559.99
29	mitsue_tollner@yahoo.com	489.30
30	mroyster@royster.com	1208.16
31	sage_wieser@cox.net	1678.60
32	simona@morasca.com	742.45
33	vinouye@aol.com	559.99
34	willard@hotmail.com	303.79
35	yuki_whobrey@aol.com	1697.46

Query executed successfully.

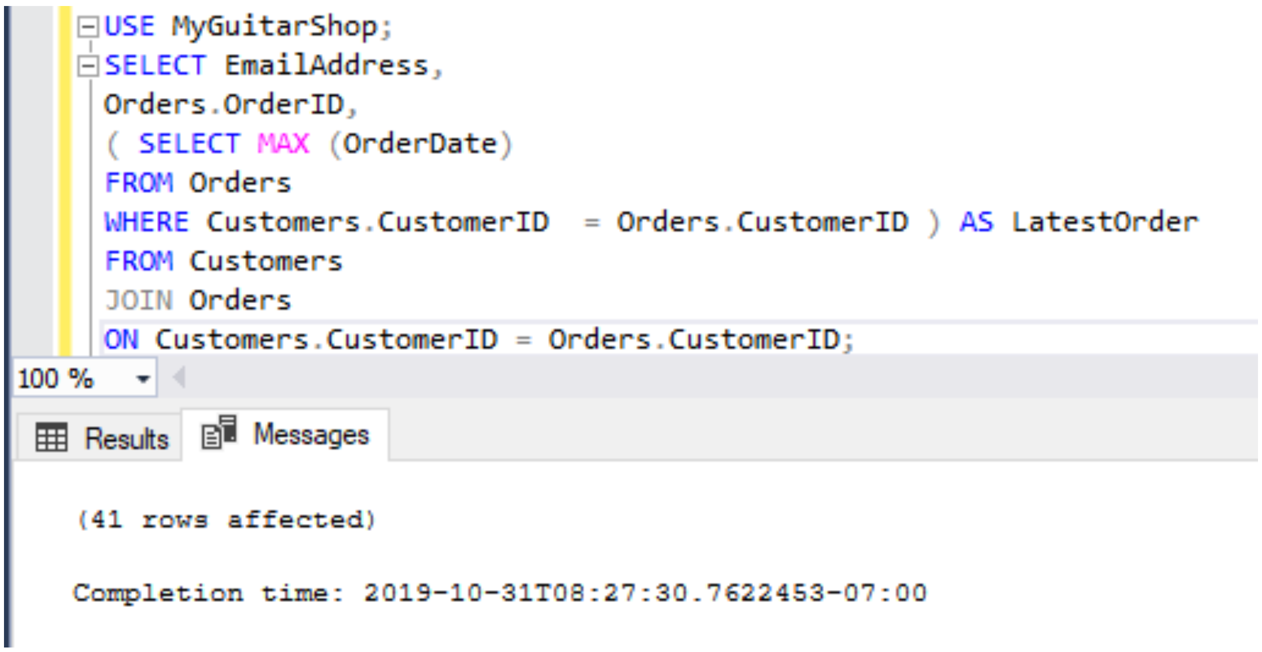
6.

The query command is as following.

```
USE MyGuitarShop;
SELECT EmailAddress,
       Orders.OrderID,
       ( SELECT MAX (OrderDate)
         FROM Orders
         WHERE Customers.CustomerID = Orders.CustomerID )
         AS LatestOrder
FROM Customers
JOIN Orders
   ON Customers.CustomerID = Orders.CustomerID;
```

This correlated subquery return one row per customer, representing the customer's newest order (the one with the latest date).

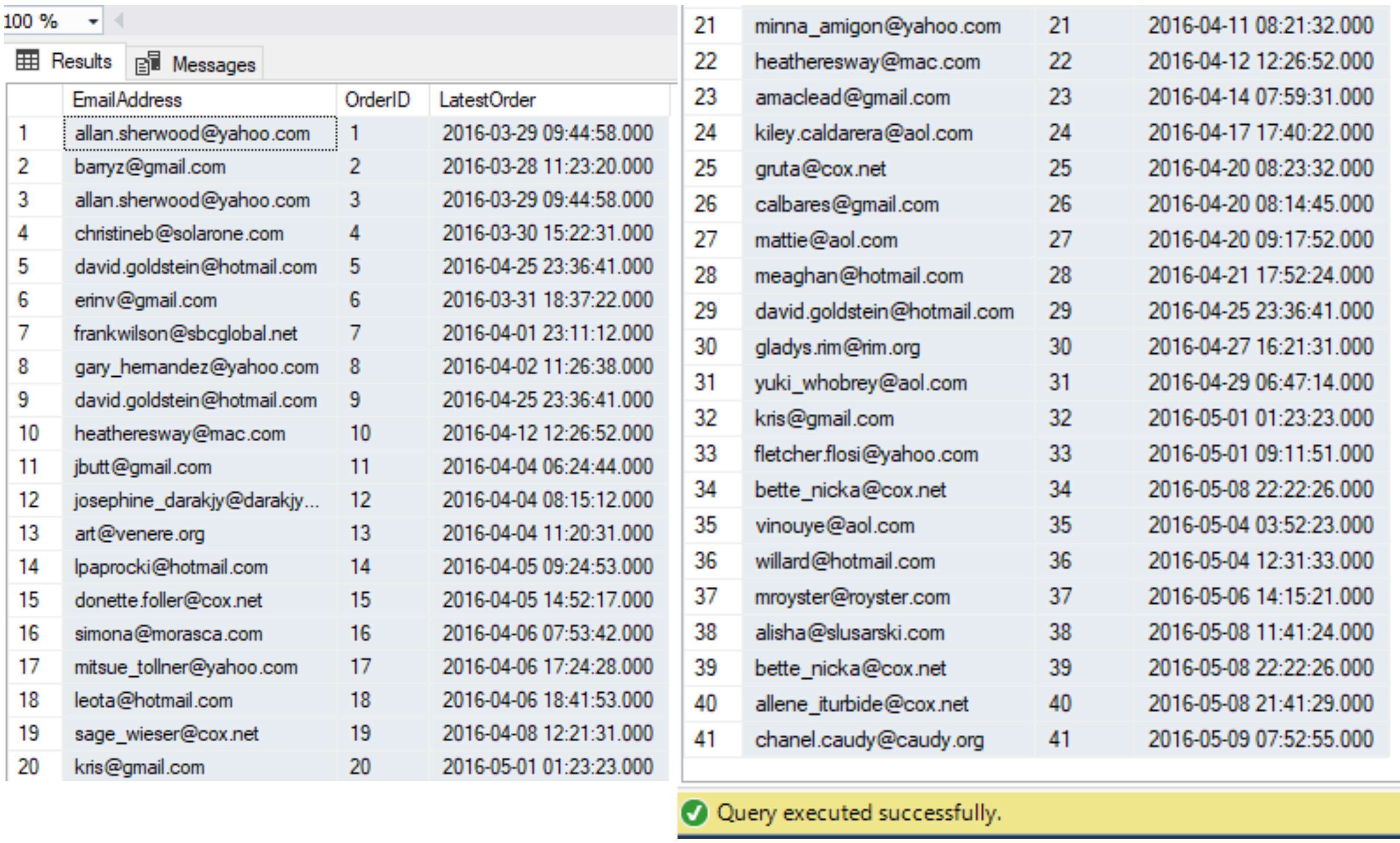
The query executed successfully:



```
USE MyGuitarShop;
SELECT EmailAddress,
Orders.OrderID,
( SELECT MAX (OrderDate)
FROM Orders
WHERE Customers.CustomerID  = Orders.CustomerID ) AS LatestOrder
FROM Customers
JOIN Orders
ON Customers.CustomerID = Orders.CustomerID;
```

```
(41 rows affected)

Completion time: 2019-10-31T08:27:30.7622453-07:00
```

The query produced result set of 41 rows as below:



	EmailAddress	OrderID	LatestOrder
1	allan.sherwood@yahoo.com	1	2016-03-29 09:44:58.000
2	barryz@gmail.com	2	2016-03-28 11:23:20.000
3	allan.sherwood@yahoo.com	3	2016-03-29 09:44:58.000
4	christineb@solarone.com	4	2016-03-30 15:22:31.000
5	david.goldstein@hotmail.com	5	2016-04-25 23:36:41.000
6	erinv@gmail.com	6	2016-03-31 18:37:22.000
7	frankwilson@sbcglobal.net	7	2016-04-01 23:11:12.000
8	gary_hernandez@yahoo.com	8	2016-04-02 11:26:38.000
9	david.goldstein@hotmail.com	9	2016-04-25 23:36:41.000
10	heatheresway@mac.com	10	2016-04-12 12:26:52.000
11	jbutt@gmail.com	11	2016-04-04 06:24:44.000
12	josephine_darakjy@darakjy...	12	2016-04-04 08:15:12.000
13	art@venere.org	13	2016-04-04 11:20:31.000
14	lpaprocki@hotmail.com	14	2016-04-05 09:24:53.000
15	donette.foller@cox.net	15	2016-04-05 14:52:17.000
16	simona@morasca.com	16	2016-04-06 07:53:42.000
17	mitsue_tollner@yahoo.com	17	2016-04-06 17:24:28.000
18	leota@hotmail.com	18	2016-04-06 18:41:53.000
19	sage_wieser@cox.net	19	2016-04-08 12:21:31.000
20	kris@gmail.com	20	2016-05-01 01:23:23.000
21	minna_amigon@yahoo.com	21	2016-04-11 08:21:32.000
22	heatheresway@mac.com	22	2016-04-12 12:26:52.000
23	amaclead@gmail.com	23	2016-04-14 07:59:31.000
24	kiley.caldarera@aol.com	24	2016-04-17 17:40:22.000
25	gruta@cox.net	25	2016-04-20 08:23:32.000
26	calbares@gmail.com	26	2016-04-20 08:14:45.000
27	mattie@aol.com	27	2016-04-20 09:17:52.000
28	meaghan@hotmail.com	28	2016-04-21 17:52:24.000
29	david.goldstein@hotmail.com	29	2016-04-25 23:36:41.000
30	gladys.rim@rim.org	30	2016-04-27 16:21:31.000
31	yuki_whobrey@aol.com	31	2016-04-29 06:47:14.000
32	kris@gmail.com	32	2016-05-01 01:23:23.000
33	fletcher.flosi@yahoo.com	33	2016-05-01 09:11:51.000
34	bette_nicka@cox.net	34	2016-05-08 22:22:26.000
35	vinouye@aol.com	35	2016-05-04 03:52:23.000
36	willard@hotmail.com	36	2016-05-04 12:31:33.000
37	mroyster@royster.com	37	2016-05-06 14:15:21.000
38	alisha@slusarski.com	38	2016-05-08 11:41:24.000
39	bette_nicka@cox.net	39	2016-05-08 22:22:26.000
40	allene_iturbide@cox.net	40	2016-05-08 21:41:29.000
41	chanel.caudy@caudy.org	41	2016-05-09 07:52:55.000

Query executed successfully.

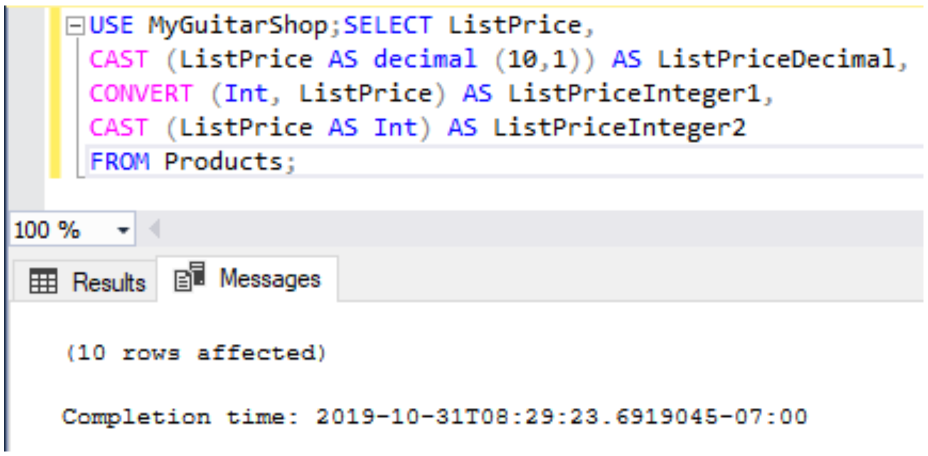
7.

The query command is as following.

```
USE MyGuitarShop;
SELECT ListPrice,
        CAST (ListPrice AS decimal (10,1)) AS ListPriceDecimal,
        CONVERT (Int, ListPrice) AS ListPriceInteger1,
        CAST (ListPrice AS Int) AS ListPriceInteger2
FROM Products;
```

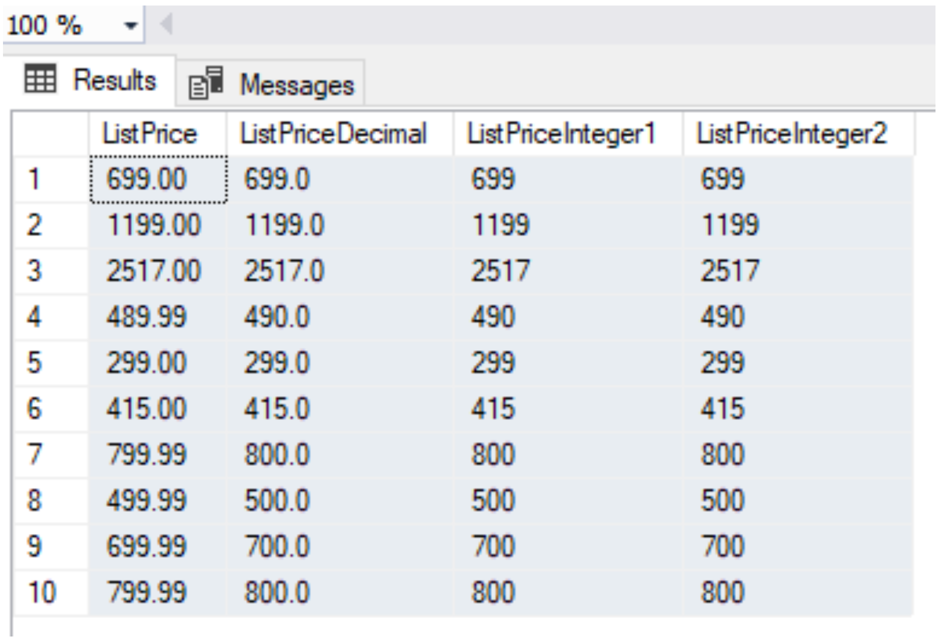
This query returns the ListPrice and adjusted decimal and integer formats using CAST function and the CONVERT function.

The query executed successfully and produced result set of 10 rows as below:



```
USE MyGuitarShop;SELECT ListPrice,
CAST (ListPrice AS decimal (10,1)) AS ListPriceDecimal,
CONVERT (Int, ListPrice) AS ListPriceInteger1,
CAST (ListPrice AS Int) AS ListPriceInteger2
FROM Products;
```

```
(10 rows affected)

Completion time: 2019-10-31T08:29:23.6919045-07:00
```



	ListPrice	ListPriceDecimal	ListPriceInteger1	ListPriceInteger2
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

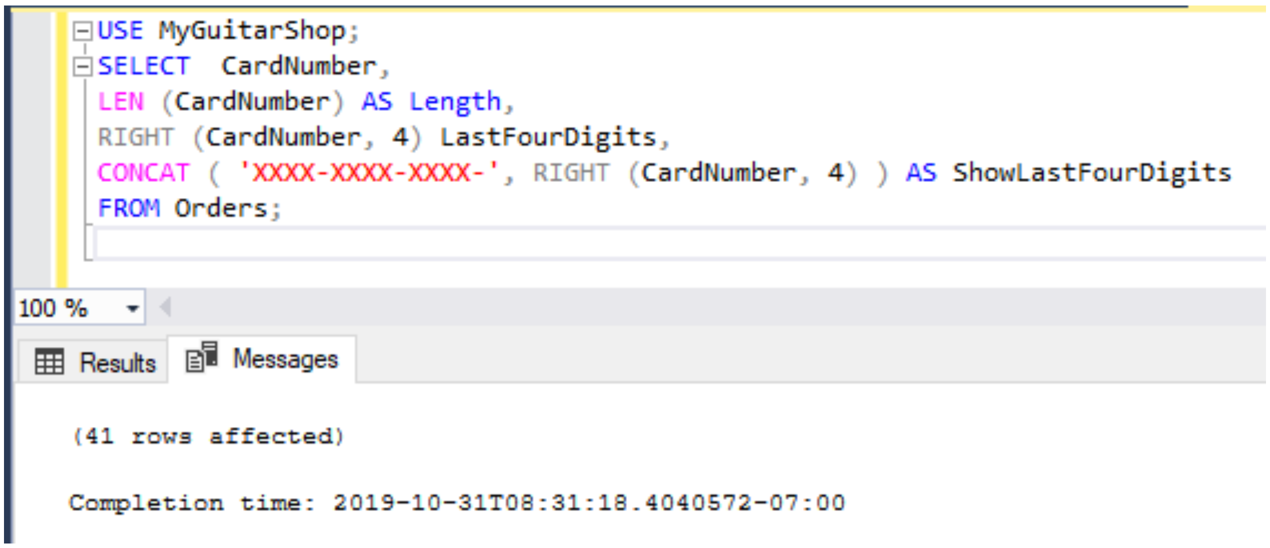
8.

The query command is as following.

```
USE MyGuitarShop;
SELECT   CardNumber,
         LEN (CardNumber) AS Length,
         RIGHT (CardNumber, 4) LastFourDigits,
         CONCAT ( 'XXXX-XXXX-XXXX-', RIGHT (CardNumber, 4) )
             AS ShowLastFourDigits
FROM Orders;
```

This query returns the CardNumber , the length of the CardNumber, the last four digits of the CardNumber, and masked CardNumber column in this format: XXXX- XXXX-XXXX-1234.

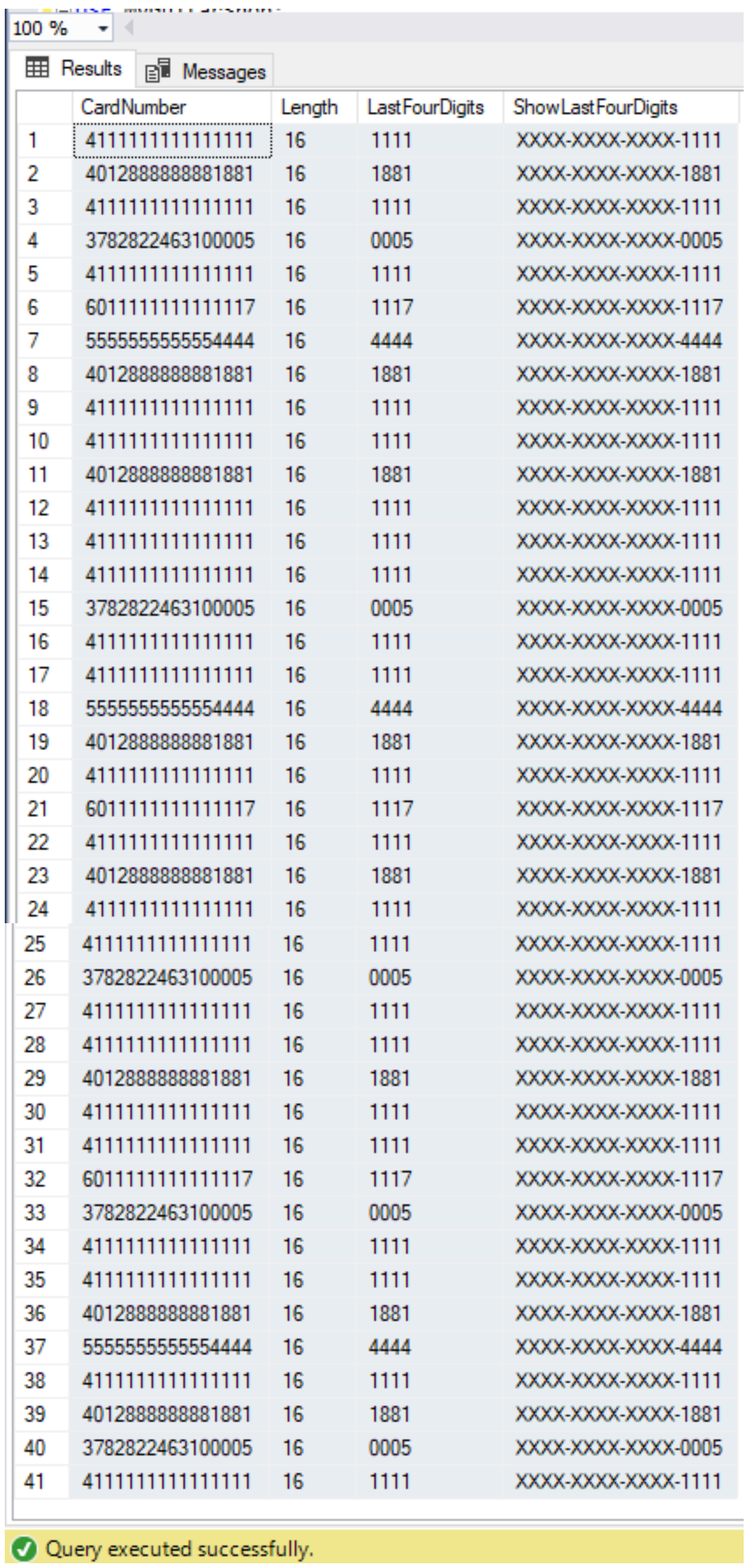
The query executed successfully:



```
USE MyGuitarShop;
SELECT   CardNumber,
LEN (CardNumber) AS Length,
RIGHT (CardNumber, 4) LastFourDigits,
CONCAT ( 'XXXX-XXXX-XXXX-', RIGHT (CardNumber, 4) ) AS ShowLastFourDigits
FROM Orders;
```

```
(41 rows affected)

Completion time: 2019-10-31T08:31:18.4040572-07:00
```

The query produced result set of 41 rows as below:



	CardNumber	Length	LastFourDigits	ShowLastFourDigits
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	3782822463100005	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
12	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
13	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
14	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
15	3782822463100005	16	0005	XXXX-XXXX-XXXX-0005
16	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
17	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
18	5555555555554444	16	4444	XXXX-XXXX-XXXX-4444
19	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
20	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
21	6011111111111117	16	1117	XXXX-XXXX-XXXX-1117
22	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
23	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
24	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
25	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
26	3782822463100005	16	0005	XXXX-XXXX-XXXX-0005
27	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
28	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
29	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
30	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
31	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
32	6011111111111117	16	1117	XXXX-XXXX-XXXX-1117
33	3782822463100005	16	0005	XXXX-XXXX-XXXX-0005
34	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
35	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
36	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
37	5555555555554444	16	4444	XXXX-XXXX-XXXX-4444
38	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
39	4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
40	3782822463100005	16	0005	XXXX-XXXX-XXXX-0005
41	4111111111111111	16	1111	XXXX-XXXX-XXXX-1111

Query executed successfully.

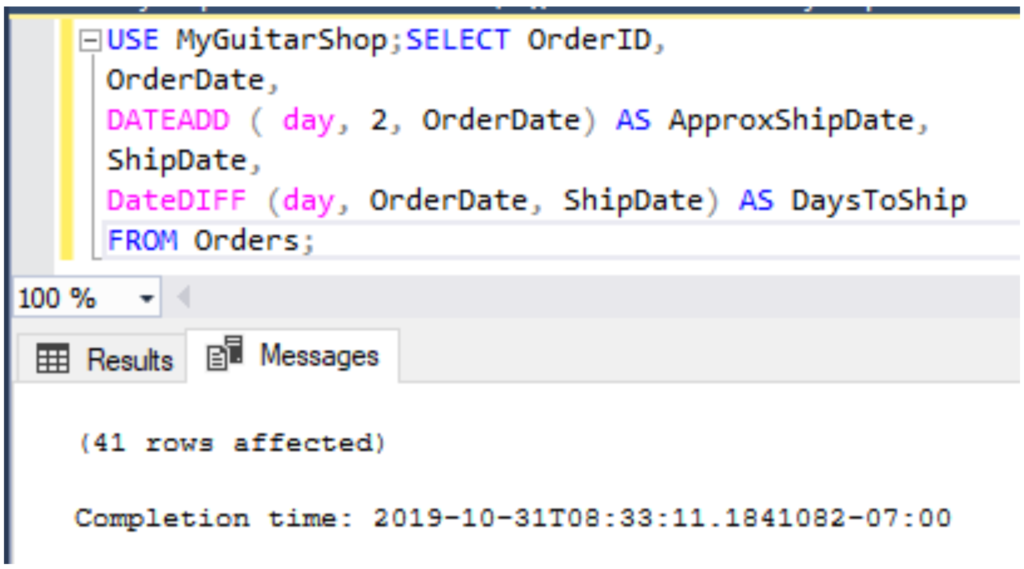
9.

The query command is as following.

```
USE MyGuitarShop;
SELECT      OrderID,
            OrderDate,
            DATEADD ( day, 2, OrderDate) AS ApproxShipDate,
            ShipDate,
            DateDIFF (day, OrderDate, ShipDate) AS DaysToShip
FROM Orders;
```

It retrieves just the orders for May 2016 and returns OrderID, OrderDate ApproxShipDate (2 days after the OrderDate), ShipDate, and DaysToShip (difference of the order date and the ship date).

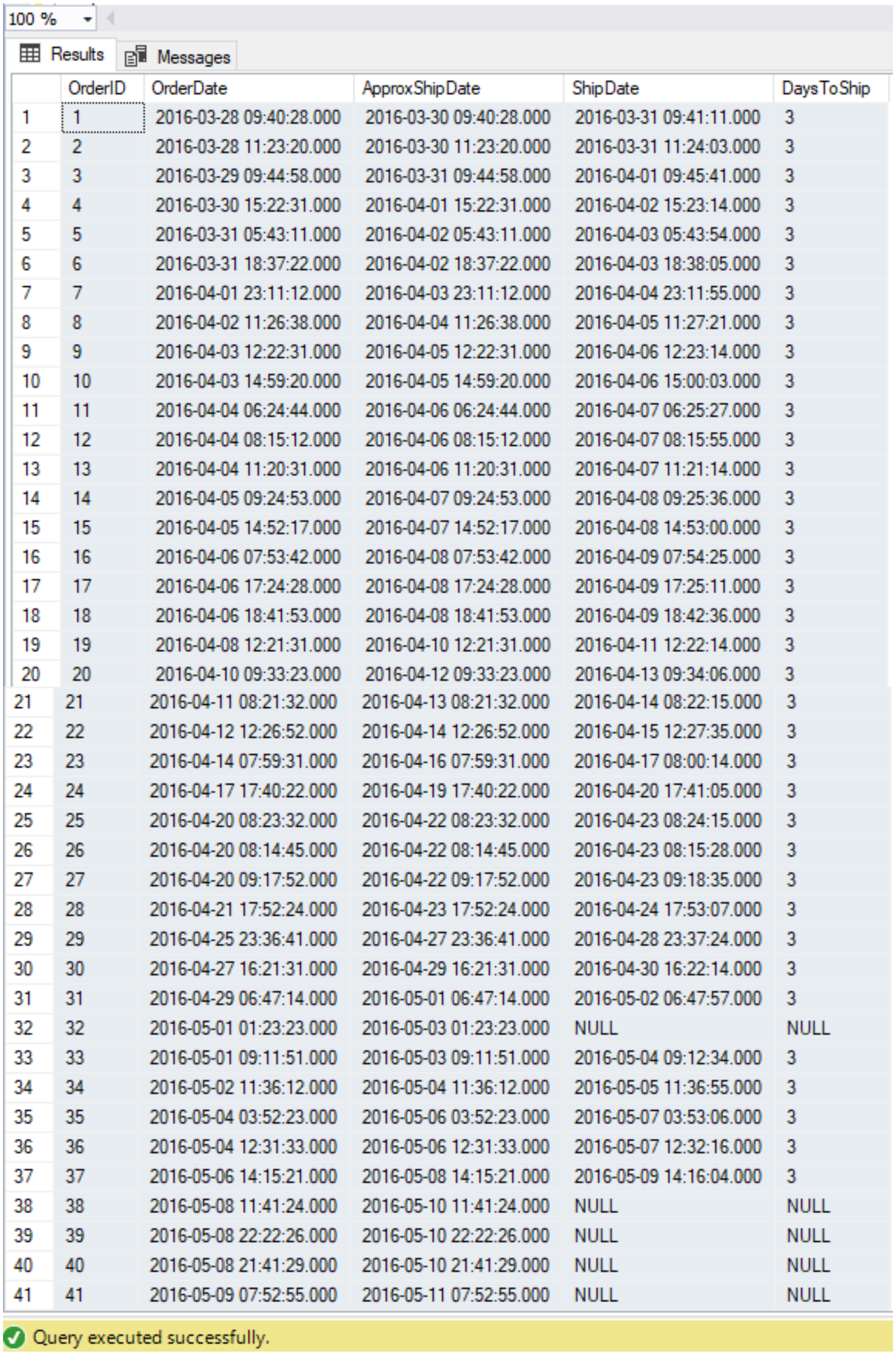
The query executed successfully:



```
USE MyGuitarShop;SELECT OrderID,
OrderDate,
DATEADD ( day, 2, OrderDate) AS ApproxShipDate,
ShipDate,
DateDIFF (day, OrderDate, ShipDate) AS DaysToShip
FROM Orders;
```

```
(41 rows affected)

Completion time: 2019-10-31T08:33:11.1841082-07:00
```

The query produced result set of 41 rows as below:



	OrderID	OrderDate	ApproxShipDate	ShipDate	DaysToShip
1	1	2016-03-28 09:40:28.000	2016-03-30 09:40:28.000	2016-03-31 09:41:11.000	3
2	2	2016-03-28 11:23:20.000	2016-03-30 11:23:20.000	2016-03-31 11:24:03.000	3
3	3	2016-03-29 09:44:58.000	2016-03-31 09:44:58.000	2016-04-01 09:45:41.000	3
4	4	2016-03-30 15:22:31.000	2016-04-01 15:22:31.000	2016-04-02 15:23:14.000	3
5	5	2016-03-31 05:43:11.000	2016-04-02 05:43:11.000	2016-04-03 05:43:54.000	3
6	6	2016-03-31 18:37:22.000	2016-04-02 18:37:22.000	2016-04-03 18:38:05.000	3
7	7	2016-04-01 23:11:12.000	2016-04-03 23:11:12.000	2016-04-04 23:11:55.000	3
8	8	2016-04-02 11:26:38.000	2016-04-04 11:26:38.000	2016-04-05 11:27:21.000	3
9	9	2016-04-03 12:22:31.000	2016-04-05 12:22:31.000	2016-04-06 12:23:14.000	3
10	10	2016-04-03 14:59:20.000	2016-04-05 14:59:20.000	2016-04-06 15:00:03.000	3
11	11	2016-04-04 06:24:44.000	2016-04-06 06:24:44.000	2016-04-07 06:25:27.000	3
12	12	2016-04-04 08:15:12.000	2016-04-06 08:15:12.000	2016-04-07 08:15:55.000	3
13	13	2016-04-04 11:20:31.000	2016-04-06 11:20:31.000	2016-04-07 11:21:14.000	3
14	14	2016-04-05 09:24:53.000	2016-04-07 09:24:53.000	2016-04-08 09:25:36.000	3
15	15	2016-04-05 14:52:17.000	2016-04-07 14:52:17.000	2016-04-08 14:53:00.000	3
16	16	2016-04-06 07:53:42.000	2016-04-08 07:53:42.000	2016-04-09 07:54:25.000	3
17	17	2016-04-06 17:24:28.000	2016-04-08 17:24:28.000	2016-04-09 17:25:11.000	3
18	18	2016-04-06 18:41:53.000	2016-04-08 18:41:53.000	2016-04-09 18:42:36.000	3
19	19	2016-04-08 12:21:31.000	2016-04-10 12:21:31.000	2016-04-11 12:22:14.000	3
20	20	2016-04-10 09:33:23.000	2016-04-12 09:33:23.000	2016-04-13 09:34:06.000	3
21	21	2016-04-11 08:21:32.000	2016-04-13 08:21:32.000	2016-04-14 08:22:15.000	3
22	22	2016-04-12 12:26:52.000	2016-04-14 12:26:52.000	2016-04-15 12:27:35.000	3
23	23	2016-04-14 07:59:31.000	2016-04-16 07:59:31.000	2016-04-17 08:00:14.000	3
24	24	2016-04-17 17:40:22.000	2016-04-19 17:40:22.000	2016-04-20 17:41:05.000	3
25	25	2016-04-20 08:23:32.000	2016-04-22 08:23:32.000	2016-04-23 08:24:15.000	3
26	26	2016-04-20 08:14:45.000	2016-04-22 08:14:45.000	2016-04-23 08:15:28.000	3
27	27	2016-04-20 09:17:52.000	2016-04-22 09:17:52.000	2016-04-23 09:18:35.000	3
28	28	2016-04-21 17:52:24.000	2016-04-23 17:52:24.000	2016-04-24 17:53:07.000	3
29	29	2016-04-25 23:36:41.000	2016-04-27 23:36:41.000	2016-04-28 23:37:24.000	3
30	30	2016-04-27 16:21:31.000	2016-04-29 16:21:31.000	2016-04-30 16:22:14.000	3
31	31	2016-04-29 06:47:14.000	2016-05-01 06:47:14.000	2016-05-02 06:47:57.000	3
32	32	2016-05-01 01:23:23.000	2016-05-03 01:23:23.000	NULL	NULL
33	33	2016-05-01 09:11:51.000	2016-05-03 09:11:51.000	2016-05-04 09:12:34.000	3
34	34	2016-05-02 11:36:12.000	2016-05-04 11:36:12.000	2016-05-05 11:36:55.000	3
35	35	2016-05-04 03:52:23.000	2016-05-06 03:52:23.000	2016-05-07 03:53:06.000	3
36	36	2016-05-04 12:31:33.000	2016-05-06 12:31:33.000	2016-05-07 12:32:16.000	3
37	37	2016-05-06 14:15:21.000	2016-05-08 14:15:21.000	2016-05-09 14:16:04.000	3
38	38	2016-05-08 11:41:24.000	2016-05-10 11:41:24.000	NULL	NULL
39	39	2016-05-08 22:22:26.000	2016-05-10 22:22:26.000	NULL	NULL
40	40	2016-05-08 21:41:29.000	2016-05-10 21:41:29.000	NULL	NULL
41	41	2016-05-09 07:52:55.000	2016-05-11 07:52:55.000	NULL	NULL

Query executed successfully.

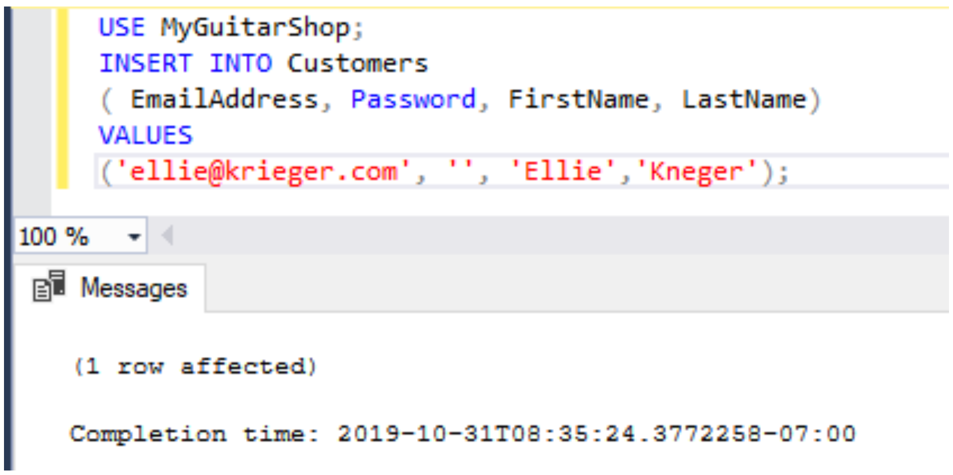
10.

The query command is as following.

```
USE MyGuitarShop;
INSERT INTO Customers
        ( EmailAddress, Password, FirstName, LastName)
VALUES
        ('ellie@krieger.com', '', 'Ellie','Kneger');
```

This action query adds a row to the Customers table using column list for this INSERT statement.

The query executed successfully:



```
USE MyGuitarShop;
INSERT INTO Customers
( EmailAddress, Password, FirstName, LastName)
VALUES
('ellie@krieger.com', '', 'Ellie','Kneger');
```

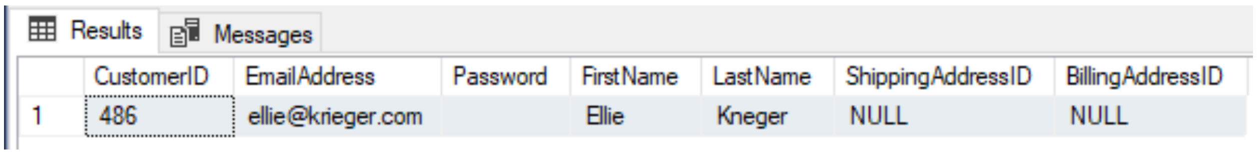
100 %

Messages

```
(1 row affected)

Completion time: 2019-10-31T08:35:24.3772258-07:00
```

The query updated 1 row in result set and can be queried as following.

```
USE MyGuitarShop;
SELECT * FROM Customers
WHERE EmailAddress = 'ellie@krieger.com';
```



Results Messages

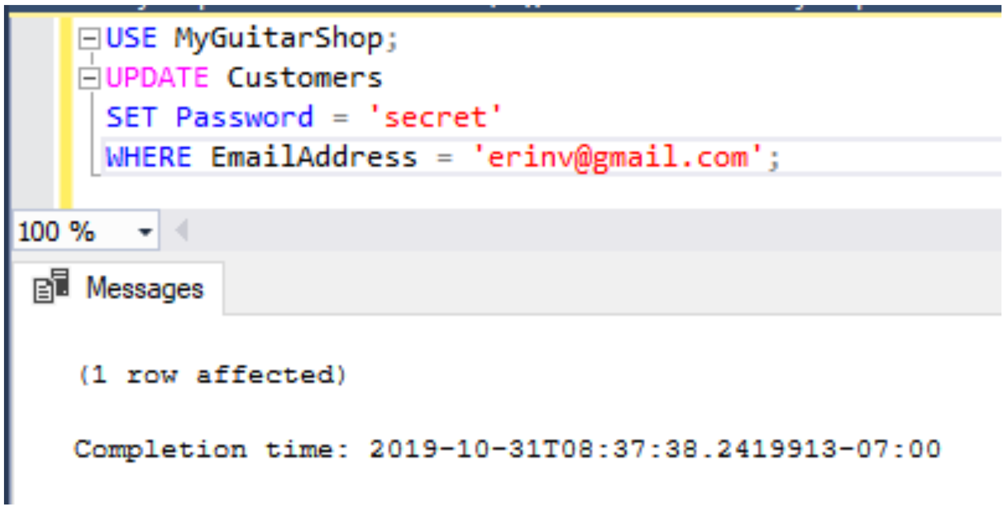
	CustomerID	EmailAddress	Password	FirstName	LastName	ShippingAddressID	BillingAddressID
1	486	ellie@krieger.com		Ellie	Kneger	NULL	NULL

11.

The query command is as following. This action query updates password for customer with a specific email address with UPDATE clause.

```
USE MyGuitarShop;
UPDATE Customers
SET Password = 'secret'
WHERE EmailAddress = 'erinv@gmail.com';
```

The query executed successfully:

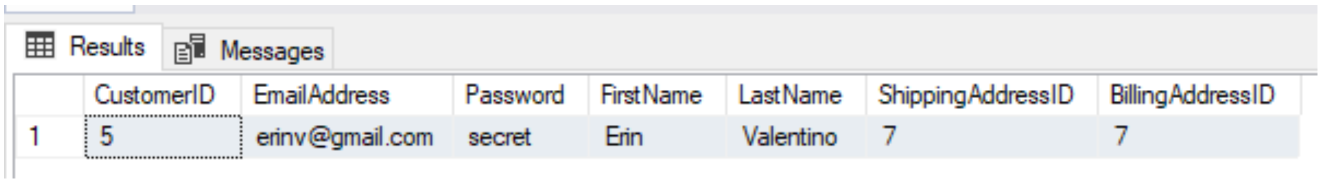


```
USE MyGuitarShop;
UPDATE Customers
SET Password = 'secret'
WHERE EmailAddress = 'erinv@gmail.com';
```

```
(1 row affected)

Completion time: 2019-10-31T08:37:38.2419913-07:00
```

The query updated 1 row in result set and can be queried as following.

```
USE MyGuitarShop;
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

D. Advanced SQL skills

1.

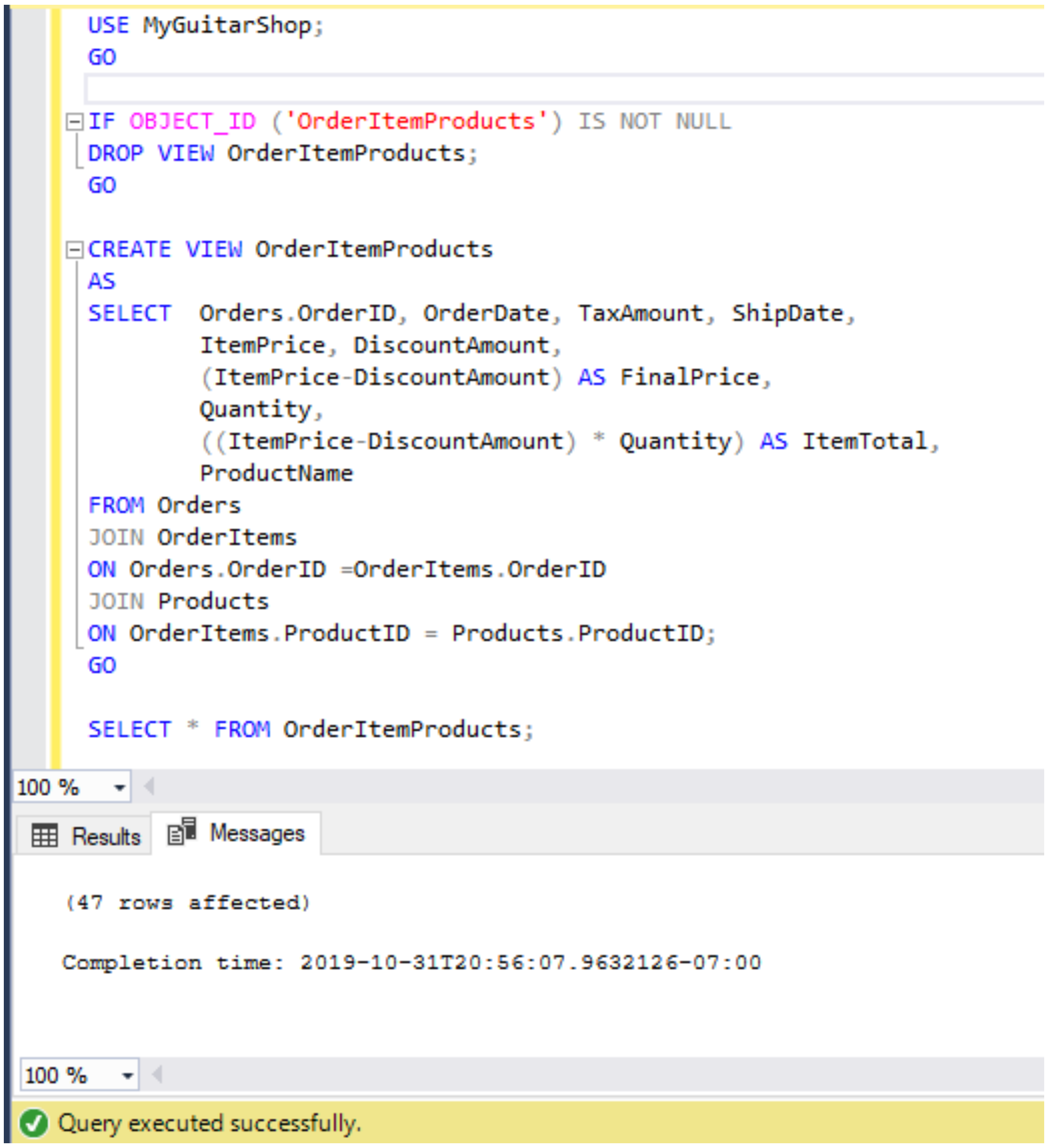
The script creates a view OrderItemProducts that returns columns from 3 tables :

Orders:	OrderID, OrderDate, TaxAmount, ShipDate
OrderItems:	ItemPrice, DiscountAmount, FinalPrice (the discount amount subtracted from the item price), Quantity, ItemTotal (the calculated total for the item)
Products:	ProductName

The query command source code is as following.

```
USE MyGuitarShop;
GO

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

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

SELECT * FROM OrderItemProducts;
```

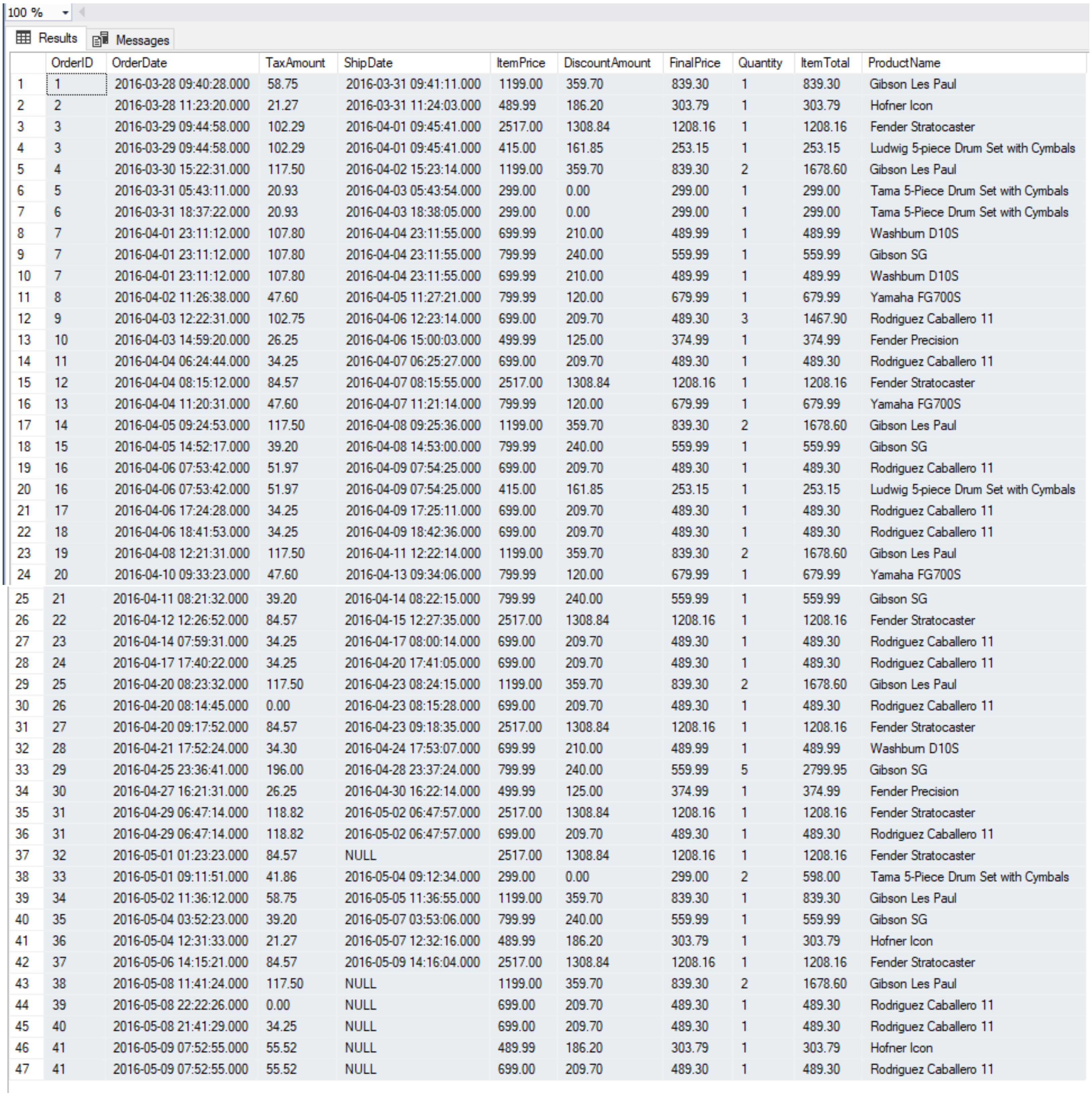
The query executed successfully:



```
USE MyGuitarShop;
GO

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

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

SELECT * FROM OrderItemProducts;
```

```
(47 rows affected)

Completion time: 2019-10-31T20:56:07.9632126-07:00
```

The query produced result set of 47 rows as below:

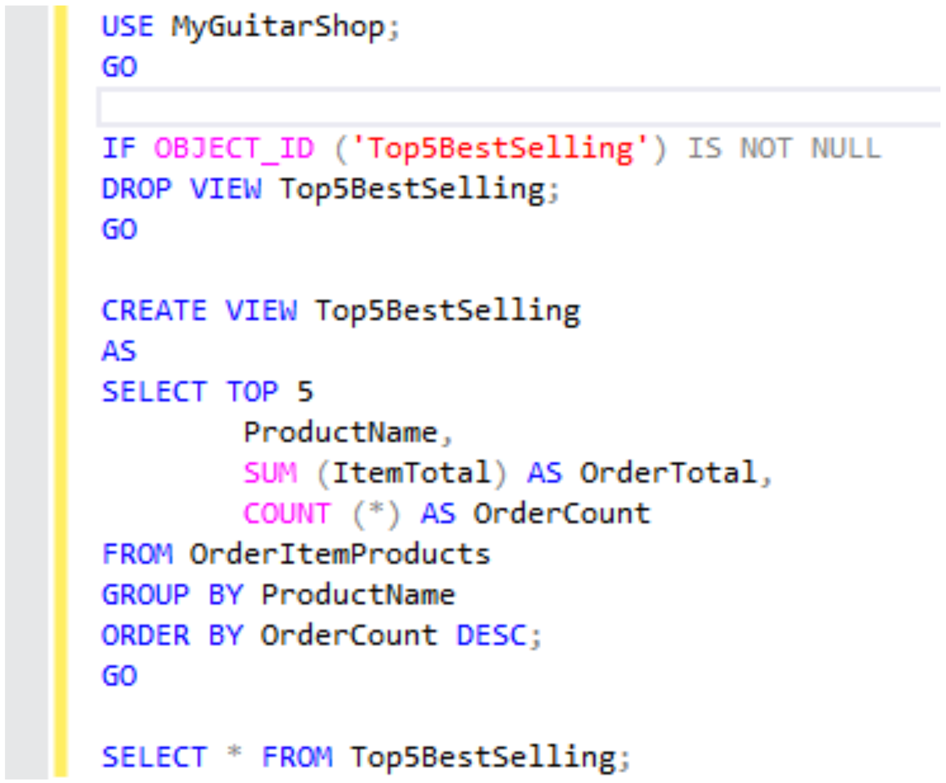


	OrderID	OrderDate	TaxAmount	ShipDate	ItemPrice	DiscountAmount	FinalPrice	Quantity	ItemTotal	ProductName
1	1	2016-03-28 09:40:28.000	58.75	2016-03-31 09:41:11.000	1199.00	359.70	839.30	1	839.30	Gibson Les Paul
2	2	2016-03-28 11:23:20.000	21.27	2016-03-31 11:24:03.000	489.99	186.20	303.79	1	303.79	Hofner 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	1208.16	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	253.15	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	1678.60	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	299.00	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	299.00	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	489.99	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	559.99	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	489.99	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	679.99	Yamaha FG700S
12	9	2016-04-03 12:22:31.000	102.75	2016-04-06 12:23:14.000	699.00	209.70	489.30	3	1467.90	Rodriguez Caballero 11
13	10	2016-04-03 14:59:20.000	26.25	2016-04-06 15:00:03.000	499.99	125.00	374.99	1	374.99	Fender Precision
14	11	2016-04-04 06:24:44.000	34.25	2016-04-07 06:25:27.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
15	12	2016-04-04 08:15:12.000	84.57	2016-04-07 08:15:55.000	2517.00	1308.84	1208.16	1	1208.16	Fender Stratocaster
16	13	2016-04-04 11:20:31.000	47.60	2016-04-07 11:21:14.000	799.99	120.00	679.99	1	679.99	Yamaha FG700S
17	14	2016-04-05 09:24:53.000	117.50	2016-04-08 09:25:36.000	1199.00	359.70	839.30	2	1678.60	Gibson Les Paul
18	15	2016-04-05 14:52:17.000	39.20	2016-04-08 14:53:00.000	799.99	240.00	559.99	1	559.99	Gibson SG
19	16	2016-04-06 07:53:42.000	51.97	2016-04-09 07:54:25.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
20	16	2016-04-06 07:53:42.000	51.97	2016-04-09 07:54:25.000	415.00	161.85	253.15	1	253.15	Ludwig 5-piece Drum Set with Cymbals
21	17	2016-04-06 17:24:28.000	34.25	2016-04-09 17:25:11.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
22	18	2016-04-06 18:41:53.000	34.25	2016-04-09 18:42:36.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
23	19	2016-04-08 12:21:31.000	117.50	2016-04-11 12:22:14.000	1199.00	359.70	839.30	2	1678.60	Gibson Les Paul
24	20	2016-04-10 09:33:23.000	47.60	2016-04-13 09:34:06.000	799.99	120.00	679.99	1	679.99	Yamaha FG700S
25	21	2016-04-11 08:21:32.000	39.20	2016-04-14 08:22:15.000	799.99	240.00	559.99	1	559.99	Gibson SG
26	22	2016-04-12 12:26:52.000	84.57	2016-04-15 12:27:35.000	2517.00	1308.84	1208.16	1	1208.16	Fender Stratocaster
27	23	2016-04-14 07:59:31.000	34.25	2016-04-17 08:00:14.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
28	24	2016-04-17 17:40:22.000	34.25	2016-04-20 17:41:05.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
29	25	2016-04-20 08:23:32.000	117.50	2016-04-23 08:24:15.000	1199.00	359.70	839.30	2	1678.60	Gibson Les Paul
30	26	2016-04-20 08:14:45.000	0.00	2016-04-23 08:15:28.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
31	27	2016-04-20 09:17:52.000	84.57	2016-04-23 09:18:35.000	2517.00	1308.84	1208.16	1	1208.16	Fender Stratocaster
32	28	2016-04-21 17:52:24.000	34.30	2016-04-24 17:53:07.000	699.99	210.00	489.99	1	489.99	Washburn D10S
33	29	2016-04-25 23:36:41.000	196.00	2016-04-28 23:37:24.000	799.99	240.00	559.99	5	2799.95	Gibson SG
34	30	2016-04-27 16:21:31.000	26.25	2016-04-30 16:22:14.000	499.99	125.00	374.99	1	374.99	Fender Precision
35	31	2016-04-29 06:47:14.000	118.82	2016-05-02 06:47:57.000	2517.00	1308.84	1208.16	1	1208.16	Fender Stratocaster
36	31	2016-04-29 06:47:14.000	118.82	2016-05-02 06:47:57.000	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
37	32	2016-05-01 01:23:23.000	84.57	NULL	2517.00	1308.84	1208.16	1	1208.16	Fender Stratocaster
38	33	2016-05-01 09:11:51.000	41.86	2016-05-04 09:12:34.000	299.00	0.00	299.00	2	598.00	Tama 5-Piece Drum Set with Cymbals
39	34	2016-05-02 11:36:12.000	58.75	2016-05-05 11:36:55.000	1199.00	359.70	839.30	1	839.30	Gibson Les Paul
40	35	2016-05-04 03:52:23.000	39.20	2016-05-07 03:53:06.000	799.99	240.00	559.99	1	559.99	Gibson SG
41	36	2016-05-04 12:31:33.000	21.27	2016-05-07 12:32:16.000	489.99	186.20	303.79	1	303.79	Hofner Icon
42	37	2016-05-06 14:15:21.000	84.57	2016-05-09 14:16:04.000	2517.00	1308.84	1208.16	1	1208.16	Fender Stratocaster
43	38	2016-05-08 11:41:24.000	117.50	NULL	1199.00	359.70	839.30	2	1678.60	Gibson Les Paul
44	39	2016-05-08 22:22:26.000	0.00	NULL	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
45	40	2016-05-08 21:41:29.000	34.25	NULL	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11
46	41	2016-05-09 07:52:55.000	55.52	NULL	489.99	186.20	303.79	1	303.79	Hofner Icon
47	41	2016-05-09 07:52:55.000	55.52	NULL	699.00	209.70	489.30	1	489.30	Rodriguez Caballero 11

2.

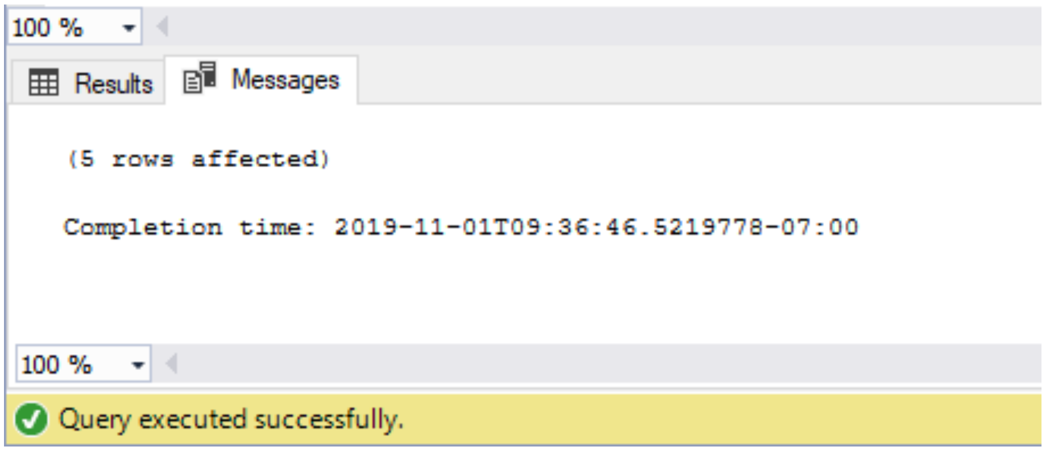
The query command is as following. The script creates a view named Top5BestSelling that uses the view OrderItemProducts created in question 1. This view return some summary information about five best selling products including : ProductName, OrderTotal (the total sales for the product) and OrderCount (the number of times the product has been ordered).

```
USE MyGuitarShop;
GO

IF OBJECT_ID ('Top5BestSelling') IS NOT NULL
        DROP VIEW Top5BestSelling;
GO

CREATE VIEW Top5BestSelling
AS
SELECT TOP 5
        ProductName,
        SUM (ItemTotal) AS OrderTotal,
        COUNT (*) AS OrderCount
FROM OrderItemProducts
GROUP BY ProductName
ORDER BY OrderCount DESC;
GO

SELECT * FROM Top5BestSelling;
```



```
USE MyGuitarShop;
GO

IF OBJECT_ID ('Top5BestSelling') IS NOT NULL
DROP VIEW Top5BestSelling;
GO

CREATE VIEW Top5BestSelling
AS
SELECT TOP 5
        ProductName,
        SUM (ItemTotal) AS OrderTotal,
        COUNT (*) AS OrderCount
FROM OrderItemProducts
GROUP BY ProductName
ORDER BY OrderCount DESC;
GO

SELECT * FROM Top5BestSelling;
```

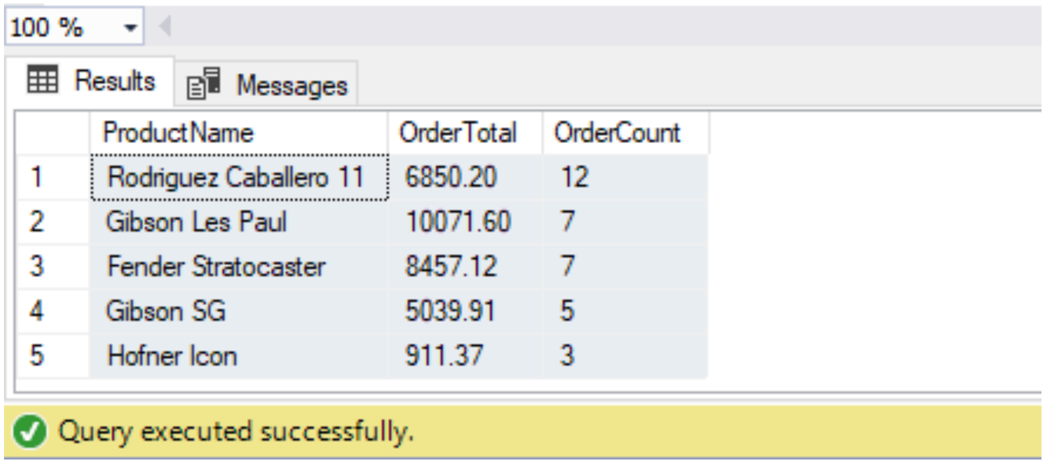
The query executed successfully:



```
(5 rows affected)

Completion time: 2019-11-01T09:36:46.5219778-07:00
```

Query executed successfully.

The query produced result set of 5 rows as below:



	ProductName	OrderTotal	OrderCount
1	Rodriguez Caballero 11	6850.20	12
2	Gibson Les Paul	10071.60	7
3	Fender Stratocaster	8457.12	7
4	Gibson SG	5039.91	5
5	Hofner Icon	911.37	3

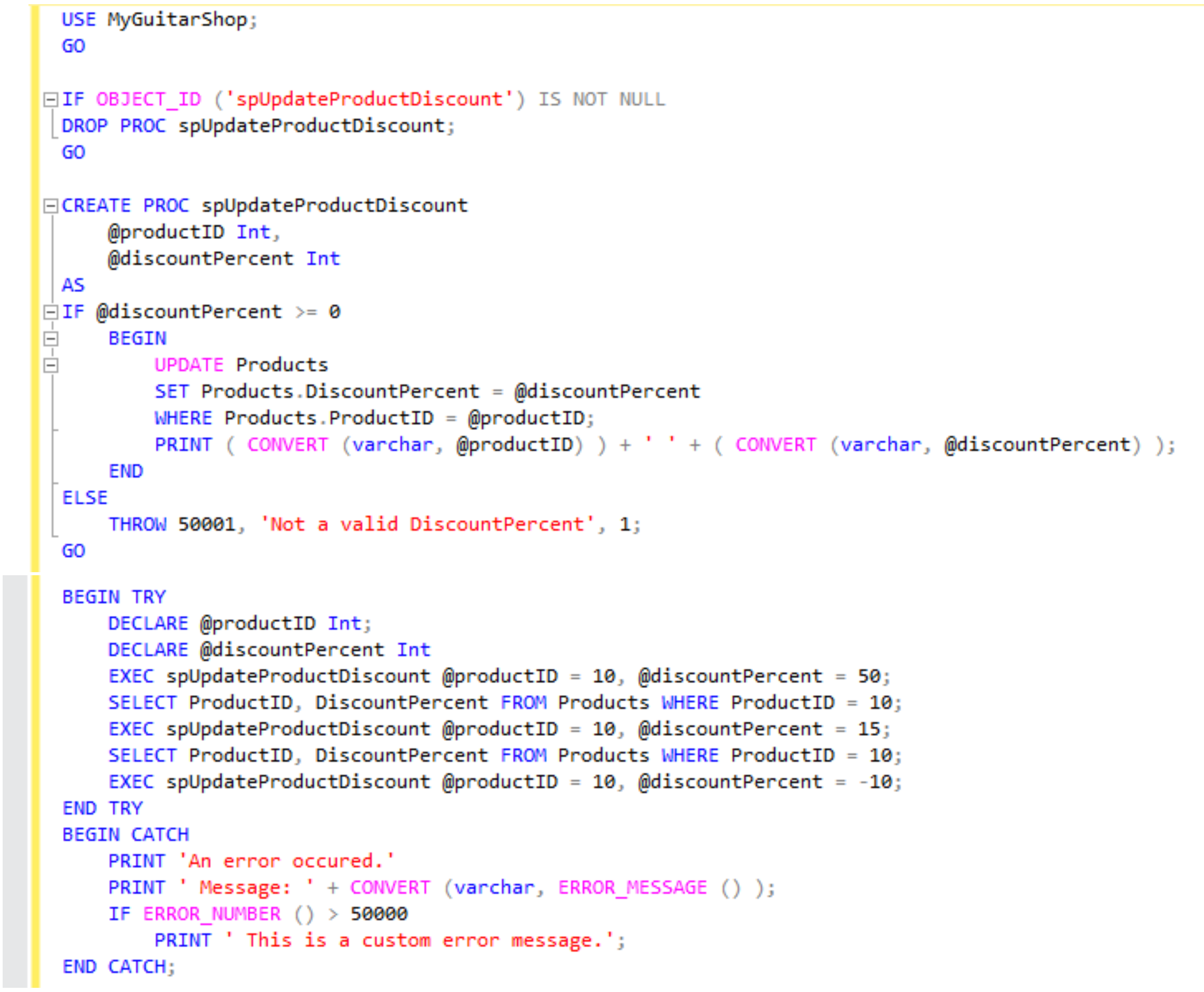
Query executed successfully.

3.

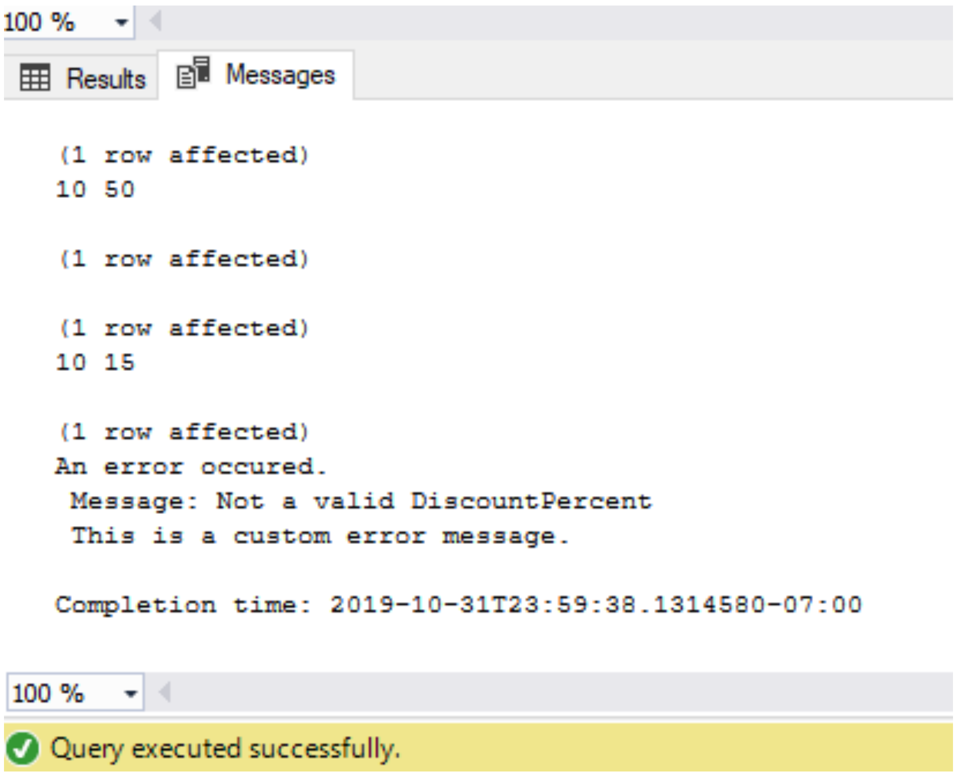
The script is as following. This script creates and calls a stored procedure spUpdateProductDiscount. This stored procedure with two parameters (product ID and DiscountPercent) updates the DiscountPercent column in the Products table. Error handling implemented for negative value in DiscountPercent.

```
USE MyGuitarShop;
GO

IF OBJECT_ID ('spUpdateProductDiscount') IS NOT NULL
DROP PROC spUpdateProductDiscount;
GO

CREATE PROC spUpdateProductDiscount
        @productID Int,
        @discountPercent Int
AS
IF @discountPercent >= 0
        BEGIN
                UPDATE Products
                SET Products.DiscountPercent = @discountPercent
        WHERE Products.ProductID = @productID;
                PRINT ( CONVERT (varchar, @productID) ) + ' ' + ( CONVERT (varchar,
@discountPercent) );
        END
ELSE
        THROW 50001, 'Not a valid DiscountPercent', 1;
GO

BEGIN TRY
        DECLARE @productID Int;
        DECLARE @discountPercent Int
        EXEC spUpdateProductDiscount @productID = 10, @discountPercent = 50;
        SELECT ProductID, DiscountPercent FROM Products WHERE ProductID = 10;
        EXEC spUpdateProductDiscount @productID = 10, @discountPercent = 15;
        SELECT ProductID, DiscountPercent FROM Products WHERE ProductID = 10;
        EXEC spUpdateProductDiscount @productID = 10, @discountPercent = -10;
END TRY
BEGIN CATCH
        PRINT 'An error occured.'
        PRINT ' Message: ' + CONVERT (varchar, ERROR_MESSAGE () );
        IF ERROR_NUMBER () > 50000
                PRINT ' This is a custom error message.';
END CATCH;
```

The script executed successfully:

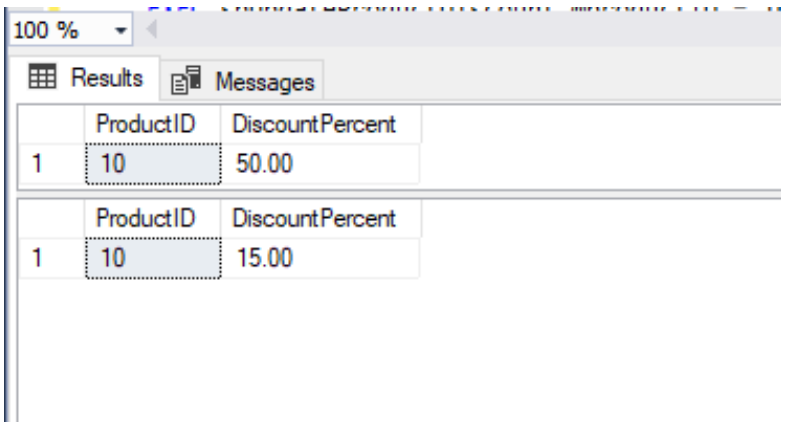


```
USE MyGuitarShop;
GO

IF OBJECT_ID ('spUpdateProductDiscount') IS NOT NULL
DROP PROC spUpdateProductDiscount;
GO

CREATE PROC spUpdateProductDiscount
    @productID Int,
    @discountPercent Int
AS
IF @discountPercent >= 0
    BEGIN
        UPDATE Products
        SET Products.DiscountPercent = @discountPercent
        WHERE Products.ProductID = @productID;
        PRINT ( CONVERT (varchar, @productID) ) + ' ' + ( CONVERT (varchar, @discountPercent) );
    END
ELSE
    THROW 50001, 'Not a valid DiscountPercent', 1;
GO

BEGIN TRY
    DECLARE @productID Int;
    DECLARE @discountPercent Int
    EXEC spUpdateProductDiscount @productID = 10, @discountPercent = 50;
    SELECT ProductID, DiscountPercent FROM Products WHERE ProductID = 10;
    EXEC spUpdateProductDiscount @productID = 10, @discountPercent = 15;
    SELECT ProductID, DiscountPercent FROM Products WHERE ProductID = 10;
    EXEC spUpdateProductDiscount @productID = 10, @discountPercent = -10;
END TRY
BEGIN CATCH
    PRINT 'An error occured.'
    PRINT ' Message: ' + CONVERT (varchar, ERROR_MESSAGE () );
    IF ERROR_NUMBER () > 50000
        PRINT ' This is a custom error message.';
END CATCH;
```



```
(1 row affected)
10 50

(1 row affected)

(1 row affected)
10 15

(1 row affected)
An error occured.
 Message: Not a valid DiscountPercent
 This is a custom error message.

Completion time: 2019-10-31T23:59:38.1314580-07:00
```

The script updated result set as following. The DiscountPercent column is changed from original 15% to 50%, then change back to 15%. The third trial with -50% cause error since discount percentage cannot be negative.



	ProductID	DiscountPercent
1	10	50.00

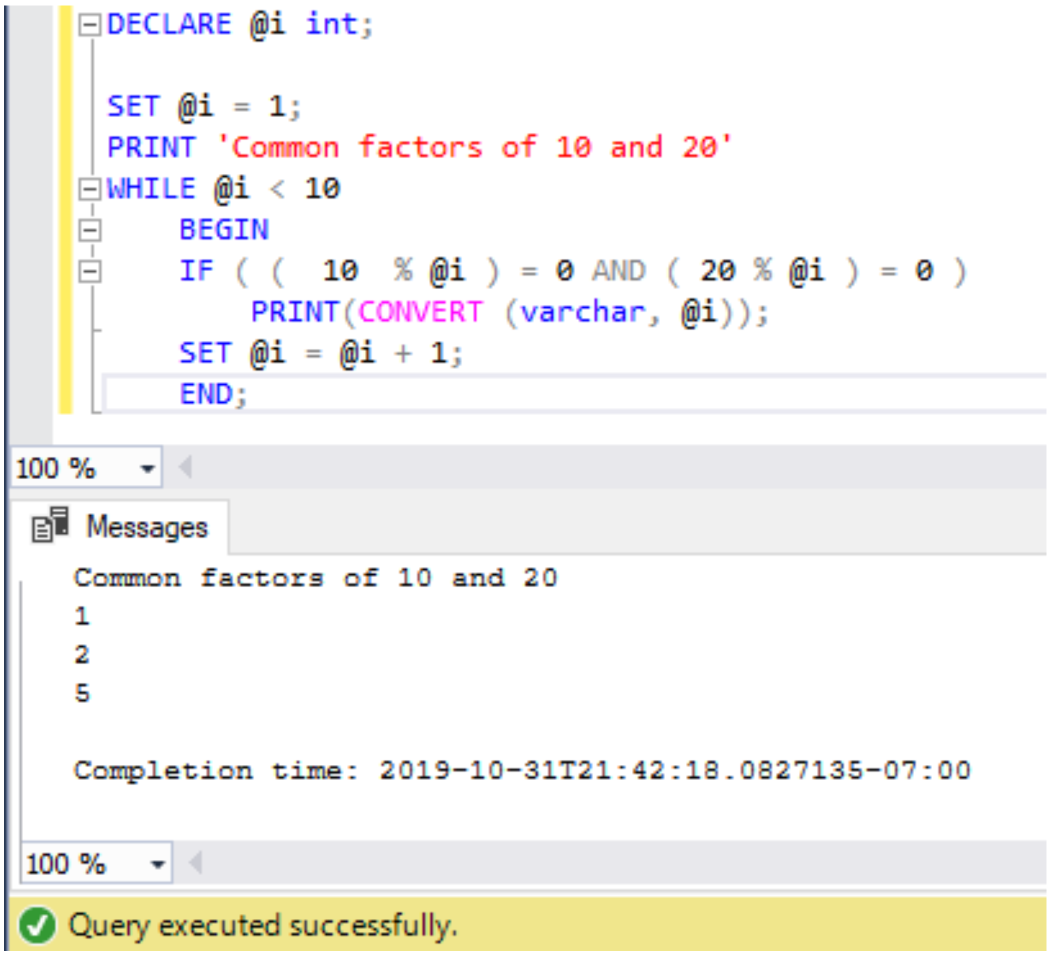
	ProductID	DiscountPercent
1	10	15.00

4.

The script is as following. This script calculates the common factors between 10 and 20.

```
DECLARE @i int;

SET @i = 1;
PRINT 'Common factors of 10 and 20'
WHILE @i < 10
        BEGIN
        IF ( ( 10 % @i ) = 0 AND ( 20 % @i ) = 0 )
                PRINT(CONVERT (varchar, @i));
        SET @i = @i + 1;
        END;
```

The script executed and generated the common factors of 10 and 20 successfully as below.



```
DECLARE @i int;

SET @i = 1;
PRINT 'Common factors of 10 and 20'
WHILE @i < 10
    BEGIN
    IF ( (  10  % @i ) = 0 AND ( 20 % @i ) = 0 )
        PRINT(CONVERT (varchar, @i));
    SET @i = @i + 1;
    END;
```

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

Completion time: 2019-10-31T21:42:18.0827135-07:00
```

Query executed successfully.

5.

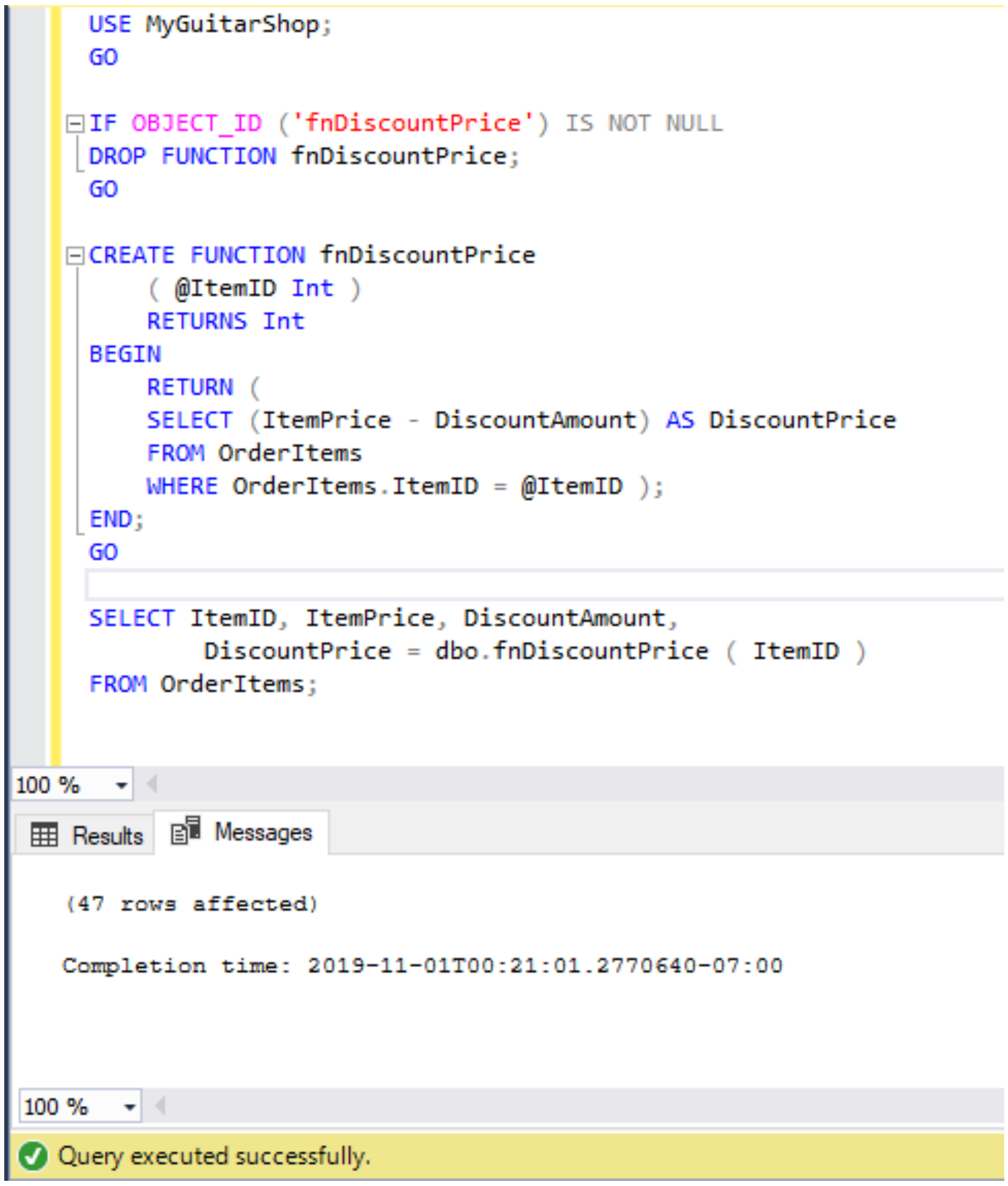
(1)

The script is as following.

```
USE MyGuitarShop;
GO

IF OBJECT_ID ('fnDiscountPrice') IS NOT NULL
DROP FUNCTION fnDiscountPrice;
GO

CREATE FUNCTION fnDiscountPrice
        ( @ItemID Int )
        RETURNS Int
BEGIN
        RETURN (
        SELECT (ItemPrice - DiscountAmount) AS DiscountPrice
        FROM OrderItems
        WHERE OrderItems.ItemID = @ItemID );
END;
GO

SELECT ItemID,
           ItemPrice,
           DiscountAmount,
           DiscountPrice = dbo.fnDiscountPrice ( ItemID )
FROM OrderItems;
```

First script creates and calls function fnDiscountPrice that calculates the discount price of an item in the OrderItems table (discount amount subtracted from item price). This function takes parameter item ID, and return the value of the discount price for the item.

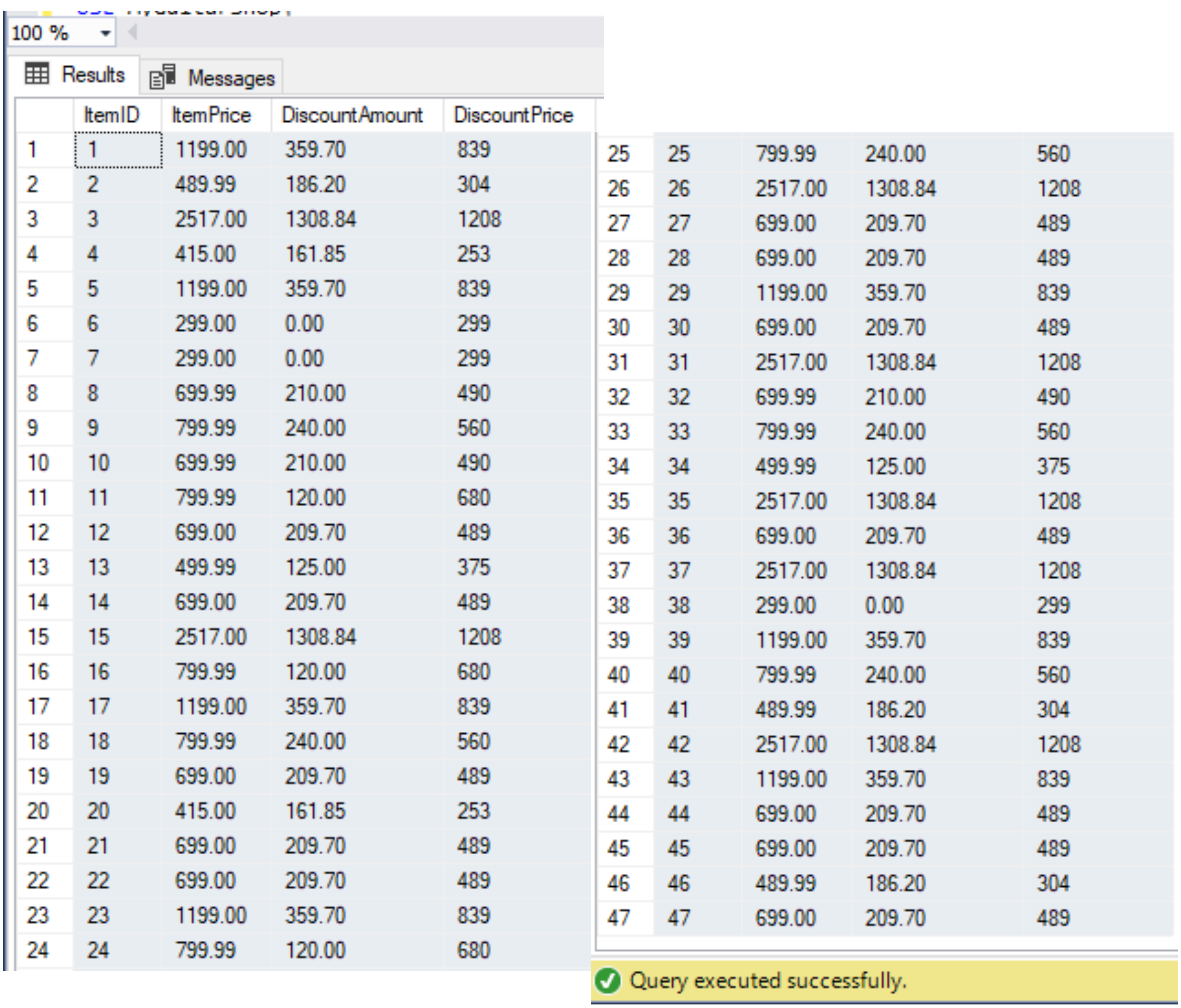
The script executed successfully:



```
USE MyGuitarShop;
GO

IF OBJECT_ID ('fnDiscountPrice') IS NOT NULL
DROP FUNCTION fnDiscountPrice;
GO

CREATE FUNCTION fnDiscountPrice
    ( @ItemID Int )
    RETURNS Int
BEGIN
    RETURN (
    SELECT (ItemPrice - DiscountAmount) AS DiscountPrice
    FROM OrderItems
    WHERE OrderItems.ItemID = @ItemID );
END;
GO

SELECT ItemID, ItemPrice, DiscountAmount,
        DiscountPrice = dbo.fnDiscountPrice ( ItemID )
FROM OrderItems;
```

```
(47 rows affected)

Completion time: 2019-11-01T00:21:01.2770640-07:00
```

The script produced result set of 47 rows as below:



	ItemID	ItemPrice	DiscountAmount	DiscountPrice
1	1	1199.00	359.70	839
2	2	489.99	186.20	304
3	3	2517.00	1308.84	1208
4	4	415.00	161.85	253
5	5	1199.00	359.70	839
6	6	299.00	0.00	299
7	7	299.00	0.00	299
8	8	699.99	210.00	490
9	9	799.99	240.00	560
10	10	699.99	210.00	490
11	11	799.99	120.00	680
12	12	699.00	209.70	489
13	13	499.99	125.00	375
14	14	699.00	209.70	489
15	15	2517.00	1308.84	1208
16	16	799.99	120.00	680
17	17	1199.00	359.70	839
18	18	799.99	240.00	560
19	19	699.00	209.70	489
20	20	415.00	161.85	253
21	21	699.00	209.70	489
22	22	699.00	209.70	489
23	23	1199.00	359.70	839
24	24	799.99	120.00	680
25	25	799.99	240.00	560
26	26	2517.00	1308.84	1208
27	27	699.00	209.70	489
28	28	699.00	209.70	489
29	29	1199.00	359.70	839
30	30	699.00	209.70	489
31	31	2517.00	1308.84	1208
32	32	699.99	210.00	490
33	33	799.99	240.00	560
34	34	499.99	125.00	375
35	35	2517.00	1308.84	1208
36	36	699.00	209.70	489
37	37	2517.00	1308.84	1208
38	38	299.00	0.00	299
39	39	1199.00	359.70	839
40	40	799.99	240.00	560
41	41	489.99	186.20	304
42	42	2517.00	1308.84	1208
43	43	1199.00	359.70	839
44	44	699.00	209.70	489
45	45	699.00	209.70	489
46	46	489.99	186.20	304
47	47	699.00	209.70	489

Query executed successfully.

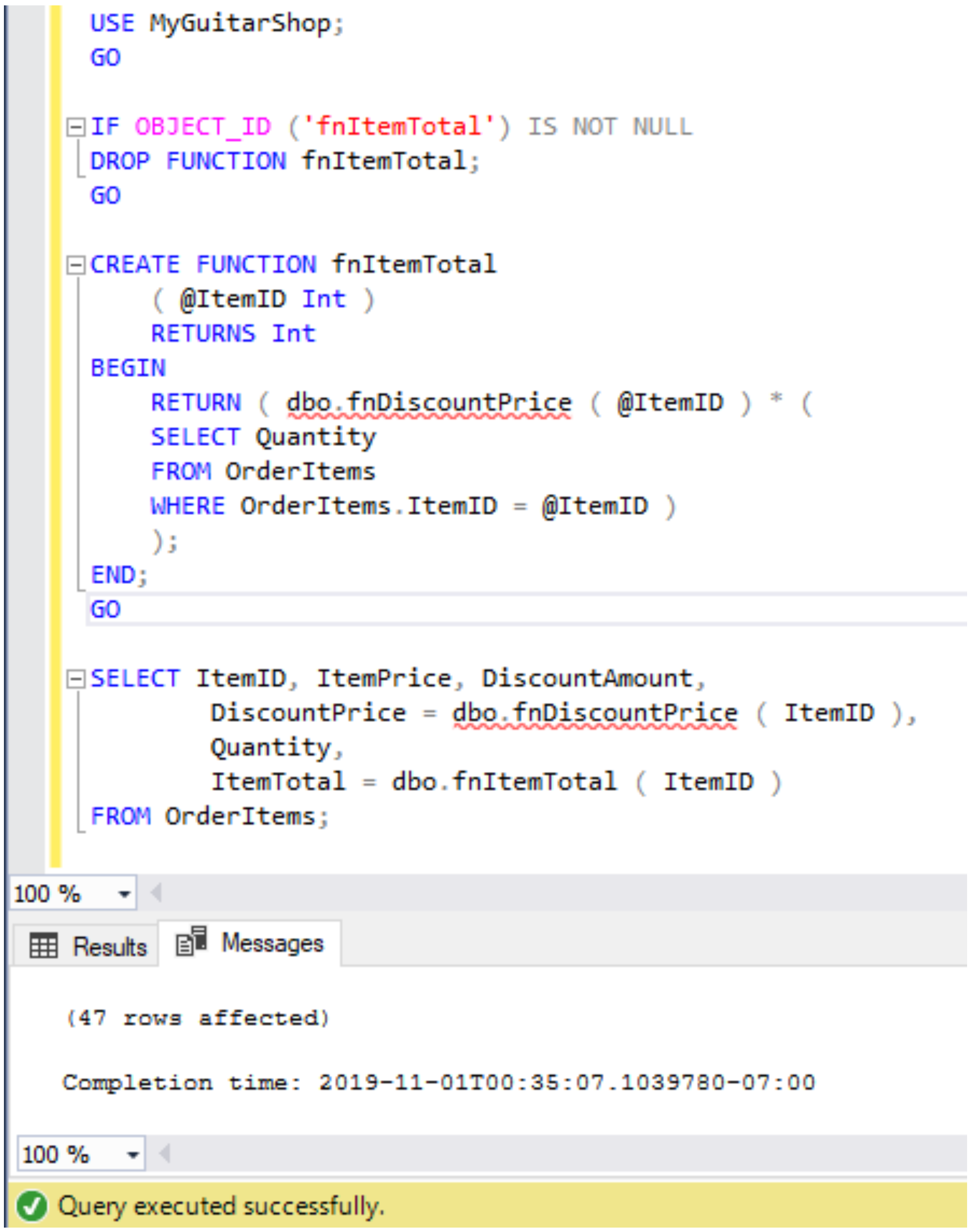
(2)

The script is as following.

Second script creates and calls function fnItemTotal that calculates the total amount of an item in the OrderItems table (discount price multiplied by quantity). This function takes parameter item ID, uses fnDiscountPrice, and returns the total value for the item.

```
USE MyGuitarShop;
GO

IF OBJECT_ID ('fnItemTotal') IS NOT NULL
DROP FUNCTION fnItemTotal;
GO

CREATE FUNCTION fnItemTotal
        ( @ItemID Int )
        RETURNS Int
BEGIN
        RETURN ( dbo.fnDiscountPrice ( @ItemID ) *
                ( SELECT Quantity
                  FROM OrderItems
                  WHERE OrderItems.ItemID = @ItemID ) );
END;
GO

SELECT   ItemID,
         ItemPrice,
         DiscountAmount,
         DiscountPrice = dbo.fnDiscountPrice ( ItemID ),
         Quantity,
         ItemTotal = dbo.fnItemTotal ( ItemID )
FROM OrderItems;
```

The script executed successfully:

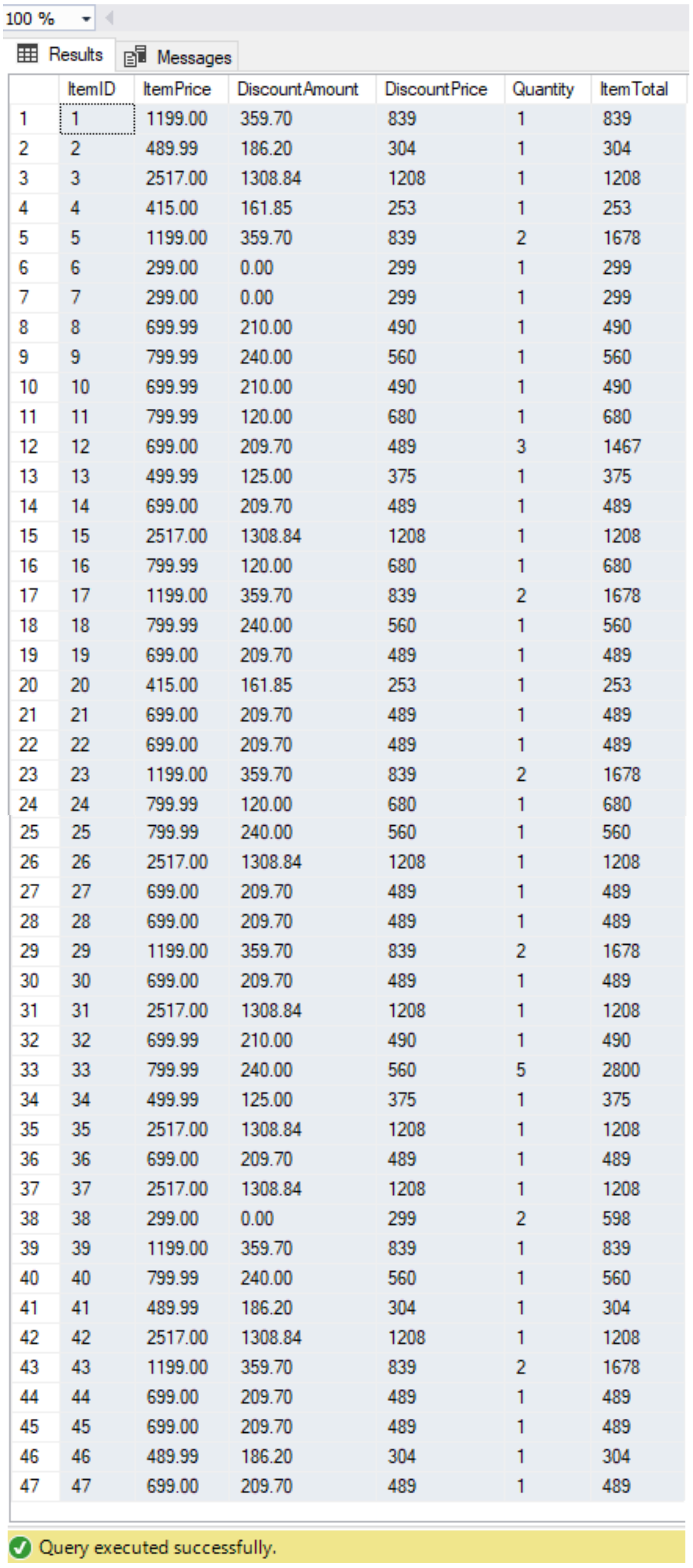


```
USE MyGuitarShop;
GO

IF OBJECT_ID ('fnItemTotal') IS NOT NULL
DROP FUNCTION fnItemTotal;
GO

CREATE FUNCTION fnItemTotal
    ( @ItemID Int )
    RETURNS Int
BEGIN
    RETURN ( dbo.fnDiscountPrice ( @ItemID ) * (
    SELECT Quantity
    FROM OrderItems
    WHERE OrderItems.ItemID = @ItemID )
    );
END;
GO

SELECT ItemID, ItemPrice, DiscountAmount,
        DiscountPrice = dbo.fnDiscountPrice ( ItemID ),
        Quantity,
        ItemTotal = dbo.fnItemTotal ( ItemID )
FROM OrderItems;
```

```
(47 rows affected)

Completion time: 2019-11-01T00:35:07.1039780-07:00
```

Query executed successfully.

The script produced result set of 47 rows as below:



	ItemID	ItemPrice	DiscountAmount	DiscountPrice	Quantity	ItemTotal
1	1	1199.00	359.70	839	1	839
2	2	489.99	186.20	304	1	304
3	3	2517.00	1308.84	1208	1	1208
4	4	415.00	161.85	253	1	253
5	5	1199.00	359.70	839	2	1678
6	6	299.00	0.00	299	1	299
7	7	299.00	0.00	299	1	299
8	8	699.99	210.00	490	1	490
9	9	799.99	240.00	560	1	560
10	10	699.99	210.00	490	1	490
11	11	799.99	120.00	680	1	680
12	12	699.00	209.70	489	3	1467
13	13	499.99	125.00	375	1	375
14	14	699.00	209.70	489	1	489
15	15	2517.00	1308.84	1208	1	1208
16	16	799.99	120.00	680	1	680
17	17	1199.00	359.70	839	2	1678
18	18	799.99	240.00	560	1	560
19	19	699.00	209.70	489	1	489
20	20	415.00	161.85	253	1	253
21	21	699.00	209.70	489	1	489
22	22	699.00	209.70	489	1	489
23	23	1199.00	359.70	839	2	1678
24	24	799.99	120.00	680	1	680
25	25	799.99	240.00	560	1	560
26	26	2517.00	1308.84	1208	1	1208
27	27	699.00	209.70	489	1	489
28	28	699.00	209.70	489	1	489
29	29	1199.00	359.70	839	2	1678
30	30	699.00	209.70	489	1	489
31	31	2517.00	1308.84	1208	1	1208
32	32	699.99	210.00	490	1	490
33	33	799.99	240.00	560	5	2800
34	34	499.99	125.00	375	1	375
35	35	2517.00	1308.84	1208	1	1208
36	36	699.00	209.70	489	1	489
37	37	2517.00	1308.84	1208	1	1208
38	38	299.00	0.00	299	2	598
39	39	1199.00	359.70	839	1	839
40	40	799.99	240.00	560	1	560
41	41	489.99	186.20	304	1	304
42	42	2517.00	1308.84	1208	1	1208
43	43	1199.00	359.70	839	2	1678
44	44	699.00	209.70	489	1	489
45	45	699.00	209.70	489	1	489
46	46	489.99	186.20	304	1	304
47	47	699.00	209.70	489	1	489

Query executed successfully.

II. Database Design

II. Database Design

1.

A 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. A large complex library database can be very complex. For example there could be multiple copies of the same books. This implementation assumes:

(a) As common practice, different ISBN issued for regular paper book, electronic book, and audio book of the same book.

(b) Library collects different kind (paper, ebook, audio) of the same book and there could be multiple copies of the same book items such as popular books, ebook files, or audio files.

(c) Stored information :

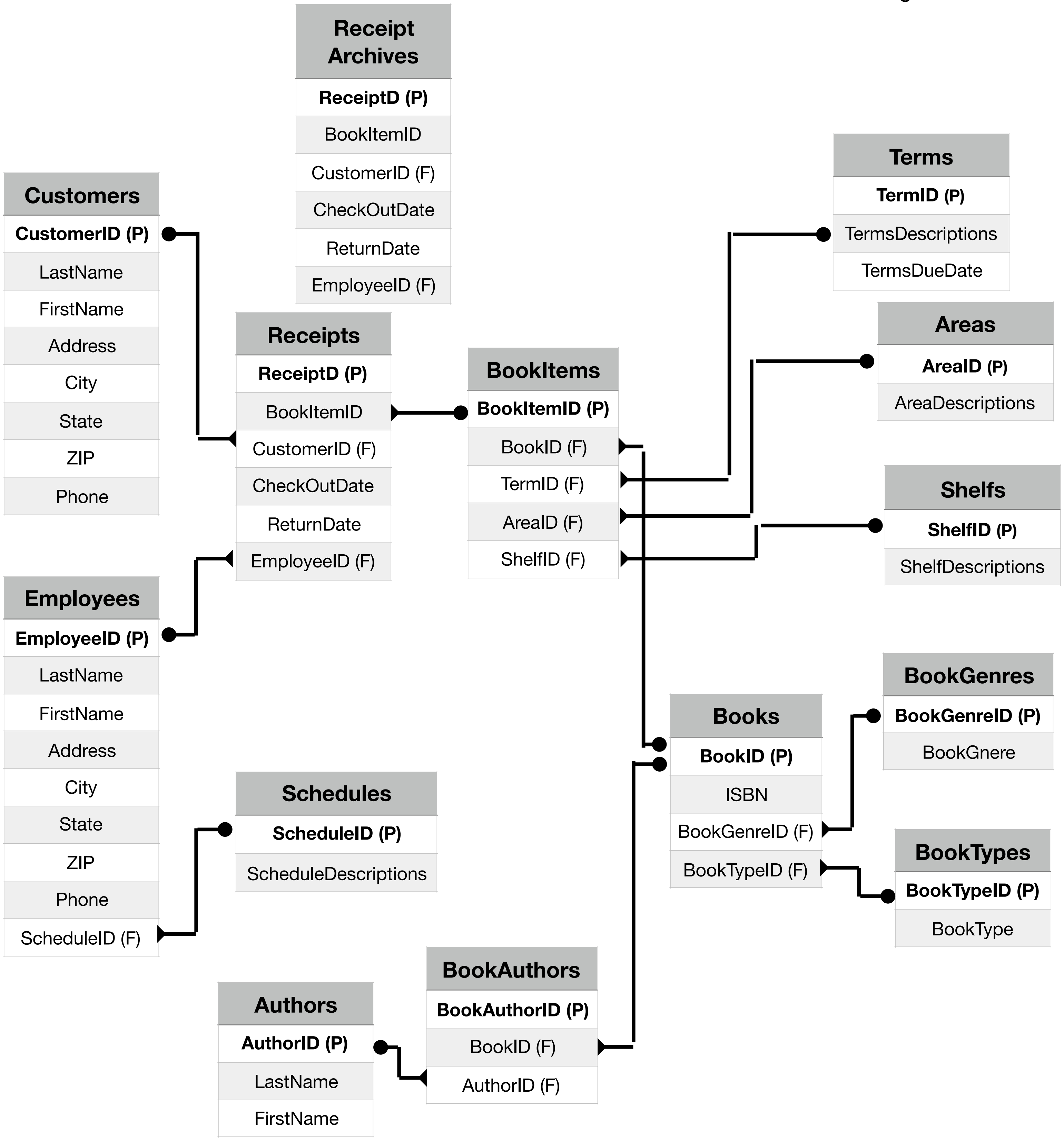
Books information :
Book type (regular books/electronic/audio);
Book authors;
Book genres;
book location (area/shelves).

Book borrow transaction information :
books that the customers borrowed;
when they are due;
employee that checked them out.

Customer information.

Employee information and employee working schedules.

The following design model is designed to meet the requirements and assumptions. Details of the tables, columns, and relationships are explained in question 2.



Receipt Archives
ReceiptD (P)
BookItemID
CustomerID (F)
CheckOutDate
ReturnDate
EmployeeID (F)
Customers
CustomerID (P)
LastName
FirstName
Address
City
State
ZIP
Phone
Receipts
ReceiptD (P)
BookItemID
CustomerID (F)
CheckOutDate
ReturnDate
EmployeeID (F)
BookItems
BookItemID (P)
BookID (F)
TermID (F)
AreaID (F)
ShelfID (F)
Terms
TermID (P)
TermsDescriptions
TermsDueDate
Areas
AreaID (P)
AreaDescriptions
Shelfs
ShelfID (P)
ShelfDescriptions
Employees
EmployeeID (P)
LastName
FirstName
Address
City
State
ZIP
Phone
ScheduleID (F)
Schedules
ScheduleID (P)
ScheduleDescriptions
Books
BookID (P)
ISBN
BookGenreID (F)
BookTypeID (F)
BookGenres
BookGenreID (P)
BookGnere
BookTypes
BookTypeID (P)
BookType
BookAuthors
BookAuthorID (P)
BookID (F)
AuthorID (F)
Authors
AuthorID (P)
LastName
FirstName

2.

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

Customers	
CustomerID (P)	**Primary key, int, NOT NULL**
LastName	varchar(25), NOT NULL
FirstName	varchar(25), NOT NULL
Address	varchar(50), NOT NULL
City	varchar(25), NOT NULL
State	varchar(2), NOT NULL
ZIP	varchar(10), NOT NULL
Phone	varchar(50), NOT NULL

Employees	
EmployeeID (P)	**Primary key, int, NOT NULL**
LastName	varchar(25), NOT NUL
FirstName	varchar(25), NOT NUL
Address	varchar(50), NOT NULL
City	varchar(25), NOT NULL
State	varchar(2), NOT NULL
ZIP	varchar(10), NOT NULL
Phone	varchar(50), NOT NULL
ScheduleID (F)	Foreign key, int, NOT NULL

Schedules	
ScheduleID (P)	**Primary key, int, NOT NULL**
ScheduleDescriptions	varchar(100), NOT NULL

Receipts	& ReceiptArchives
ReceiptID (P)	**Primary key, NOT NULL**
BookItemID	Foreign Key, NOT NULL
CustomerID (F)	Foreign Key, NOT NULL
CheckOutDate	smalldatetime, NOT NULL
ReturnDate	smalldatetime, NULL
EmployeeID (F)	Foreign key, int, NOT NULL

BookItems	
BookItemID (P)	**Primary Key, int, NOT NULL**
BookID (F)	Foreign key, int, NOT NULL
TermID (F)	Foreign key, int, NOT NULL
AreaID (F)	Foreign key, int, NOT NULL
ShelfID (F)	Foreign key, int, NOT NULL

Books	
BookID (P)	**Primary Key, int, NOT NULL**
ISBN	varchar(25), NOT NULL
BookGenreID (F)	Foreign key, int, NOT NULL
BookTypeID (F)	Foreign key, int, NOT NULL

BookAuthors	
BookAuthorID (P)	**Primary Key, int, NOT NULL**
BookID (F)	Foreign key, int, NOT NULL
AuthorID (F)	Foreign key, int, NOT NULL

Authors	
AuthorID (P)	**Primary Key, int, NOT NULL**
LastName	varchar(25), NOT NULL
FirstName	varchar(25), NOT NULL

Terms	
TermID (P)	**Primary Key, int, NOT NULL**
TermsDescriptions	varchar(25), NOT NULL
TermsDueDate	smallint, NOT NULL

Areas	
AreaID (P)	**Primary Key, int, NOT NULL**
AreaDescriptions	varchar(25), NOT NULL

Shelfs	
ShelfID (P)	**Primary Key, int, NOT NULL**
ShelfDescriptions	varchar(25), NOT NULL

BookGenres	
BookGenreID (P)	**Primary Key, int, NOT NULL**
BookGnere	varchar(25), NOT NULL

BookTypes	
BookTypeID (P)	**Primary Key, int, NOT NULL**
BookType	varchar(25), NOT NULL

The relationships between tables are listed below:

Customers table to Receipts & ReceiptArchives table :	One-to-Many
Schedules table to Employees table :	One-to-Many
Employees table to Receipts & ReceiptArchives table :	One-to-Many
BookItems table to Receipts & ReceiptArchives table :	One-to-Many
At most one active receipt row in Receipts for one book-item row in BookItems. Returned book records to be archived to ReceiptsArchives with batch operation.	
BookGenres table to Books table :	One-to-Many
BookTypes table to Books table :	One-to-Many
Authors table to BookAuthors table :	One-to-Many
Books table to BookAuthors table :	One-to-Many
Authors table to Books table :	Many-to-Many
Authors can have multiple books and books can have multiple authors. BookAuthors table serve as link table of this Many-to-Many relationship	
Terms table to BookItems table :	One-to-Many
Areas table to BookItems table :	One-to-Many
Shelfs table to BookItems table :	One-to-Many
Books table to BookItems table :	One-to-Many

3.

Normalize your design into 3rd Normal Form.

(1) First (1NF) : The value stored in each cell must be scalar value
All cells must have scalar value only. Many descriptive cells are using pre-defined ID in integer to use the pre-defined descriptions such as

Terms ID	for	TermsDescriptions
AreaID	for	AreaDescriptions
ShelfID	for	ShelfDescriptions
BookGenreID	for	BookGenres
BookTypsID	for	BookTypes
ScheduleID	for	ScheduleDescriptions

Therefore this database is design into 1NF

(2) Second (2NF): Every non-key column must depends on the entire primary key
Third (3NF) : Every non-key column must depend ONLY on the primary key

All non-key columns are checked and confirmed that they depend on entire primary key and depends ONLY on primary key. Therefore this database is design into 3NF.

Customers	
CustomerID (P)	
LastName	V
FirstName	V
Address	V
City	V
State	V
ZIP	V
Phone	V

Employees	
EmployeeID (P)	
LastName	V
FirstName	V
Address	V
City	V
State	V
ZIP	V
Phone	V
ScheduleID (F)	

Receipts & ReceiptArchives	
BookItemID (P)	
CustomerID (F)	
CheckOutDate	V
ReturnDate	V
EmployeeID (F)	

BookItems	
BookItemID (P)	
BookID (F)	
TermID (F)	
AreaID (F)	
ShelfID (F)	

BookAuthors	
BookAuthorID (P)	
BookID (F)	
AuthorID (F)	

Books	
BookID (P)	
ISBN	V
BookGenreID (F)	
BookTypeID (F)	

Schedules	
ScheduleID (P)	
ScheduleDescriptions	V

Areas	
AreaID (P)	
AreaDescriptions	V

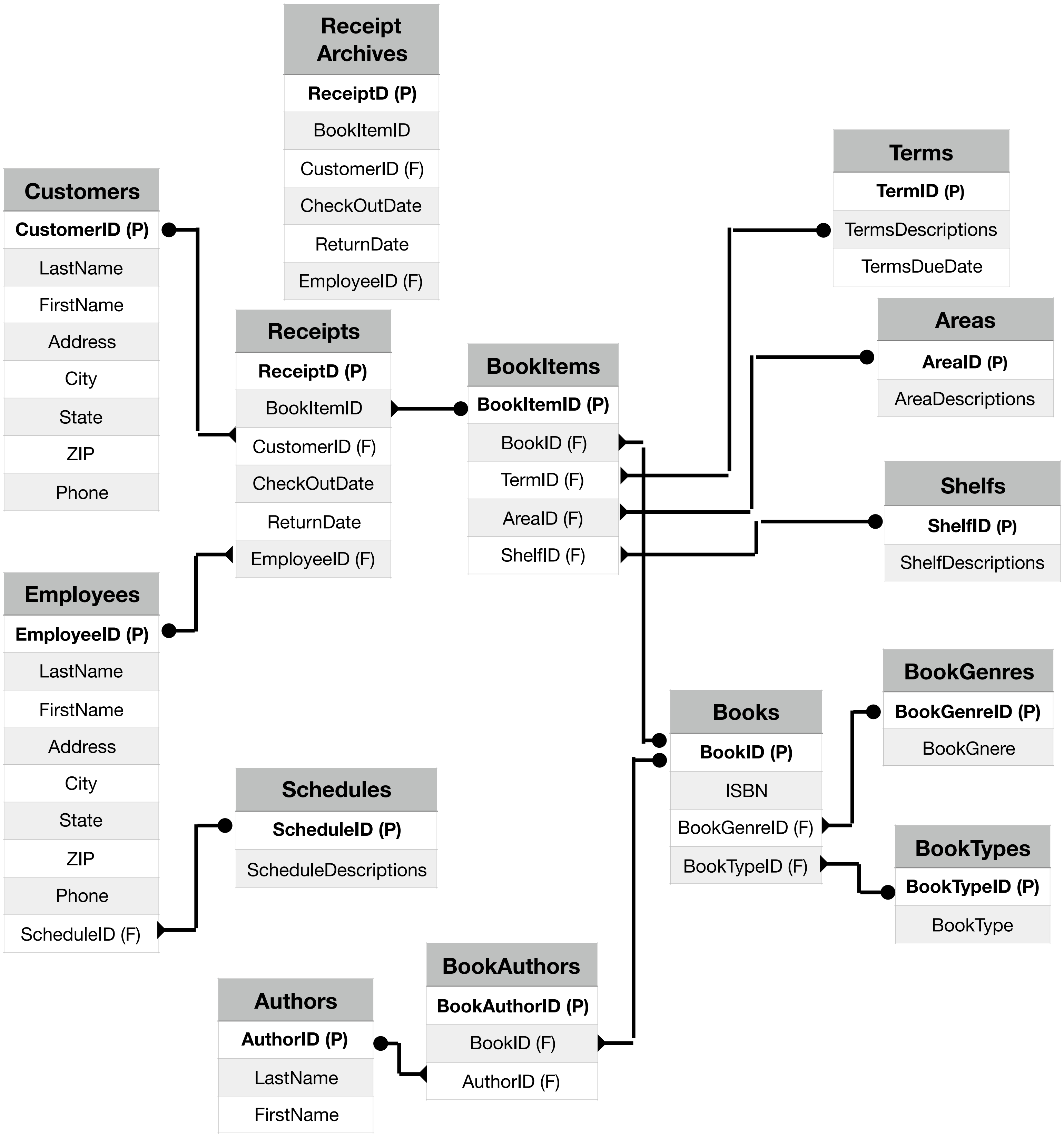
Authors	
AuthorID (P)	
LastName	V
FirstName	V

Terms	
TermID (P)	
TermsDescriptions	V
TermsDueDate	V

Shelfs	
ShelfID (P)	
ShelfDescriptions	V

BookGenres	
BookGenreID (P)	
BookGnere	V

BookTypes	
BookTypeID (P)	
BookType	V


Receipt Archives
ReceiptD (P)
BookItemID
CustomerID (F)
CheckOutDate
ReturnDate
EmployeeID (F)
Customers
CustomerID (P)
LastName
FirstName
Address
City
State
ZIP
Phone
Receipts
ReceiptD (P)
BookItemID
CustomerID (F)
CheckOutDate
ReturnDate
EmployeeID (F)
BookItems
BookItemID (P)
BookID (F)
TermID (F)
AreaID (F)
ShelfID (F)
Terms
TermID (P)
TermsDescriptions
TermsDueDate
Areas
AreaID (P)
AreaDescriptions
Shelfs
ShelfID (P)
ShelfDescriptions
Employees
EmployeeID (P)
LastName
FirstName
Address
City
State
ZIP
Phone
ScheduleID (F)
Schedules
ScheduleID (P)
ScheduleDescriptions
Books
BookID (P)
ISBN
BookGenreID (F)
BookTypeID (F)
BookGenres
BookGenreID (P)
BookGnere
BookTypes
BookTypeID (P)
BookType
BookAuthors
BookAuthorID (P)
BookID (F)
AuthorID (F)
Authors
AuthorID (P)
LastName
FirstName

4.

Explain your design, including relationship between tables.

A large complex library database can be very complex. I tried to collect information how library system works and the following is the features I implemented:

1. Unique ISBN number for each kind (paper, ebook, audio) of the same book. This is common practice according to information on Google.

2. Library collects different kind (paper, ebook, audio) of the same book and there could be multiple copies of the same book items such as popular books, ebook files, or audio files.

3. Each books item borrowed has their own receipt ticket in the database.

The database design can be described by giving some examples as following.

Customers table

One row per customer. Only customer's specific information stored in this table

CustomerID (P)	**LastName**	**FirstName**	**Address**	**City**	**State**	**ZIP**	**Phone**
100	Joe	Bidon	1 Main St.	Reno	NV	12345	123-456-7890
101	Donald	Trump	2 Main St	Tampa	FL	87654	999-888-7777

Employees table

One row per customer. Only customer's specific information stored in this table

Employee ID (P)	**LastNa me**	**FirstNam e**	**Address**	**City**	**State**	**ZIP**	**Phone**	**ScueduleID**
10	Joe	Li	1 State St.	Reno	NV	12345	123-456-7891	1
11	Harrison	Ford	2 State St	Tampa	FL	87654	999-888-6666	3

Schedules table

One row per schedule type Schedule types pre-defined in this table.
ScheduleID (scalar integer) used in ScheduleID column of Employees table

ScheduleID (P)	**ScheduleDescriptions**
1	Weekday ShiftA MWF 8AM to 5PM
2	Weekday ShiftB T,Th,Sat 8AM to 5PM
3	Weekday ShiftC M,T,W,Th,F 3PM to 7PM
4	Weekend ShiftD Sunday 8AM to 5PM

Receipts & ReceiptArchives table

One row per book item borrowed. Only one active receipt per book item. Returned book item receipt row is inactive and to be (moved) archived to ReceiptArchices table in batch.

ReceiptID (P)	BookItemID	CustomerID	CheckOutDate	ReturnDate	EmployeeID
10000	5003331	100	10/30/2019	NULL	10
10001	5018882	101	10/30/2019	11/2/2019	11
10002	4902261	101	11/1/2019	NULL	11
10003	5018882	100	11/3/2019	NULL	11
10004	2000101	101	11/2/2019	NULL	10

Terms table

One row per term type Terms for each book items are pre-defined in this table. Some popular items could have shorter return due date.

TermID (P)	TermsDescriptions	TermsDueDate
1	30 days return	30
2	15 days return	15
3	7 days return	7
4	popular items, 2 days return	2

Aresa table

One row per book storage area. Area information for each book items are pre-defined in this table.

AreaID (P)	AreaDescriptions
100	1F left wing
200	2F right wing
300	3F special collection
400	4F electronic and audio collection

Shelfs table

One row per book storage shelf. shelf information for each book items are pre-defined in this table.

ShelfID (P)	AreaDescriptions
1	shelf 1
2	shelf 2
3	shelf 3
4	air-conditioned shelf 4

BookItems table

One row per book item which can be individually checked-out. Multiple copies of the same book can exist for popular titles or availability reason.

BookItemID (P)	BookID	TermID	AreaID	ShelfID
5003331	30021	2	1	3
5018882	53487	3	3	2
4902261	99765	4	4	4
5018882	44820	1	2	1
2000101	11000	1	2	

Books table

One row per book title with unique ISBN. Paper, electronic, audio books of the same title have different ISBN, So, separate rows are defined for the title.

BookID (P)	ISBN	BookGenreID	BookTypeID
30021	0-321-29535-8	2	1
53487	978-1-890774-96-7	3	1
99765	978-1-593-27283-8	4	3
44820	220-334-87990-12	14	2
11000	100-20-4875927-23	1	1

BookGebres Table

One row per book genre types.   Genres are pre-defined in this table. Scalar data BookGenreID is used to indicate the book categories.

BookGenreID (P)	**BookGenre**
1	Science - Computer
2	Fiction - Adventure
3	Literature - English
4	Art - Painting

BookTypes table

One row per book types.   Book types are pre-defined in this table. Scalar data BookTypeD is used to indicate the book categories.

BookTypeID (P)	**BookType**
1	Paper book
2	electronic book
3	Audio book
4	Video Tape

BookAuthors table

Link table for Books and Authors tables for many-to-many relationship. A book can have multiple authories and an author can have multiple books collected in the library.

BookAuthorID	**BookID (P)**	**AuthorID**
44000	30021	22398
44983	53487	39864
44987	99765	49087
44988	44820	77651
29376	11000	10334

Authors table

One row per author. Author informations are stored in this table

A book can have multiple authories and an author can have multiple books collected in the library. The many-to-many relationship is maintained by link table BookAuthors table,

AuthorID	LastName	FirstName
22398	Syverson	Bryan
39864	Murach	Joel
49087	Kleinberg	Jon
77651	Tardos	Eva
10334	Clinton	Bill

Remarks on the project

This is a very length project consists of two parts. The first part is the review of the foundations of SQL language. It includes the query from single, multiple table, summary query, sub-query, usage of functions, action query, views, script, stored function. The selected one or two questions for each chapter refreshed the memory of the SQL coding.

The second part is the design of the ‘library’ database. The planning, partition, defining the database tables, relationships, columns, and data-types, attributes exercised the basic steps of a database design process.