CSE581: Introduction to DBMS

Project 2

Uber Database

(A demo database using to store uber company data with basic relationship)

Tianjia He

454553451

Date: 2019/5/1

Guidelines

1.Uber Database Design

1.1 abstract

This document is a report document of project 2 Uber Database Design. This Uber database Design include 11 tables and all of It follow the 3rd normal form. First chapter is the introduction of the design of this database. I will talk about the detailed information about how to design each table. The second chapter is about the sample views I designed for this database to show some business reports. The Third chapter introduced the triggers and some constraints to illustrate the business logic to incorporate into the design. The fourth chapter related to the established security and some stored procedures and function. The fifth chapter is my short conclusion of this project.

1.2 Introduction to Uber Database

This database is used to store and manage some of Uber's necessary data, such as passenger information, driver information, order information, and so on. Reasonable information management can greatly improve the company's business efficiency and performance. This database design is based on the third paradigm, which greatly reduces data redundancy and improves structural rationality.

The following is the specific information for each table.

Customers:
CustomerID
FirstName
LastName
Address1
Address2
City
State
Email
Phone
LastTripDate
TripsNum
Status

Trips:
TripID
DateBooked
PickUpTime
DropOffTime
Completed
PickUpAddress
DropOffAddress
DriverID
Tips
CreditCardID

CreditCard:
CreditCardID

CustomerID
CreditCardInfo

CustomerRatings:
CRatingID
RatingDate
Score
Text
TripID

Drivers:
DriverID
SSN
DriverLicenseID
InsuranceID
AccountID
FirstName
LastName
Address1
Address2
City
State
Phone
StartTime

BankAccounts:
AccountID
BankName
AccountNum
Type

PaymentsRecords:
PaymentID
Date
Amount
DriverID

DriverRatings:
DRatingID
RatingDate
Score
Text
TripID

Cars:
CarID
DriverID
Model
Make
Year
Color
Class

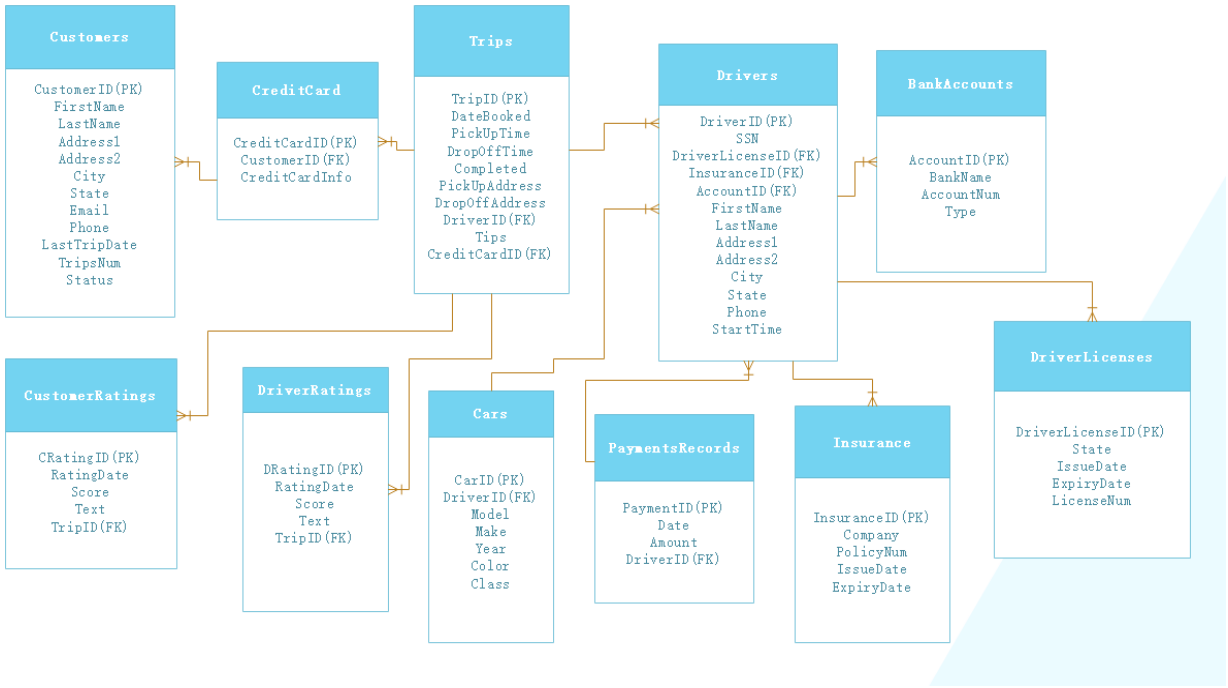
DriverLicenses:
DriverLicenseID
State
IssueDate
ExpiryDate
LicenseNum

Insurance:
InsuranceID
Company
PolicyNum
IssueDate
ExpiryDate

Customers	Trips	CreditCard	CustomerRatings	Drivers	BankAccounts
CustomerID	TripID	CreditCardID	CRatingID	DriverID	AccountID
FirstName	DateBooked	CustomerID	RatingDate	SSN	BankName
LastName	PickUpTime	CreditCardInfo	Score	DriverLicenseID	AccountNum
Address1	DropOffTime		Text	InsuranceID	Type
Address2	Completed		TripID	AccountID	
City	PickUpAddress			FirstName	
State	DropOffAddress			LastName	
Email	DriverID			Address1	
Phone	Tips			Address2	
LastTripDate	CreditCardID			City	
TripsNum				State	
Status				Phone	
				StartTime	

PaymentsRecords	DriverRatings	Cars	DriverLicenses	Insurance
PaymentID	DRatingID	CarID	DriverLicenseID	InsuranceID
Date	RatingDate	DriverID	State	Company
Amount	Score	Model	IssueDate	PolicyNum
DriverID	Text	Make	ExpiryDate	IssueDate
	TripID	Year	LicenseNum	ExpiryDate
		Color		
		Class		

Here is the E/R diagram of the database. You can see every table name, PK and FK



1.3 Created the Uber Database

I created an Uber database using the Microsoft SQL Server Management Studio.
Here is my source SQL code. **You can see how I defined each table and its columns, the data nullability and datatype.**

SOURCE CODE:

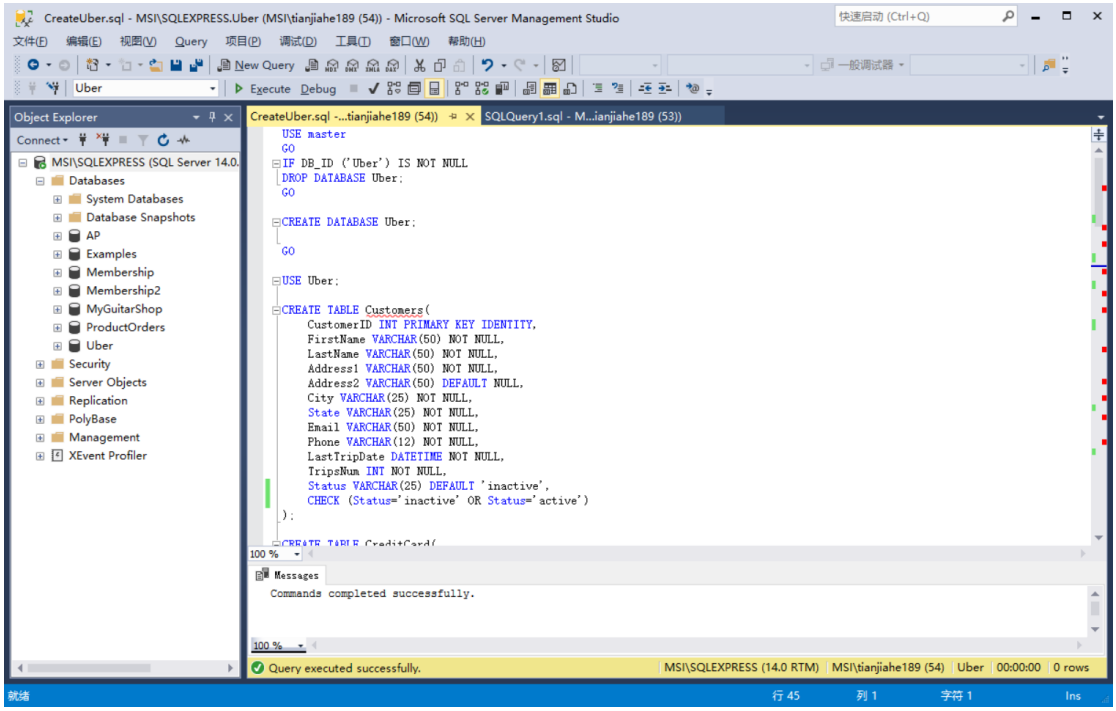
```
////////////////////////////////////////////////////////////////////////////////////////////////////
USE master
GO
```

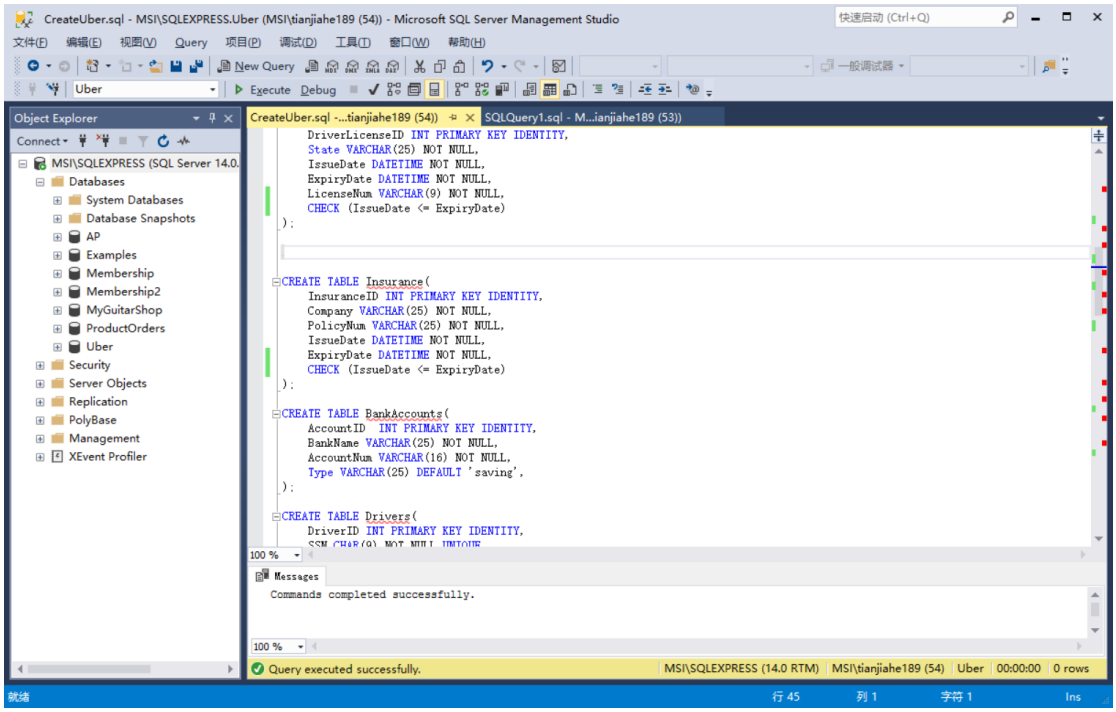
```
IF DB_ID ('Uber') IS NOT NULL
DROP DATABASE Uber;
GO

CREATE DATABASE Uber;

GO

USE Uber;

CREATE TABLE Customers(
    CustomerID INT PRIMARY KEY IDENTITY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Address1 VARCHAR(50) NOT NULL,
    Address2 VARCHAR(50) DEFAULT NULL,
    City VARCHAR(25) NOT NULL,
    State VARCHAR(25) NOT NULL,
    Email VARCHAR(50) NOT NULL,
    Phone VARCHAR(12) NOT NULL,
    LastTripDate DATETIME NOT NULL,
    TripsNum INT NOT NULL,
    Status VARCHAR(25) DEFAULT 'inactive',
    CHECK (Status='inactive' OR Status='active')
);

CREATE TABLE CreditCard(
    CreditCardID INT PRIMARY KEY IDENTITY,
    CustomerID INT REFERENCES Customers(CustomerID),
    CreditCardInfo VARCHAR(50) NOT NULL,
);


CREATE TABLE DriverLicenses(
    DriverLicenseID INT PRIMARY KEY IDENTITY,
    State VARCHAR(25) NOT NULL,
    IssueDate DATETIME NOT NULL,
    ExpiryDate DATETIME NOT NULL,
    LicenseNum VARCHAR(9) NOT NULL,
    CHECK (IssueDate <= ExpiryDate)
);
```

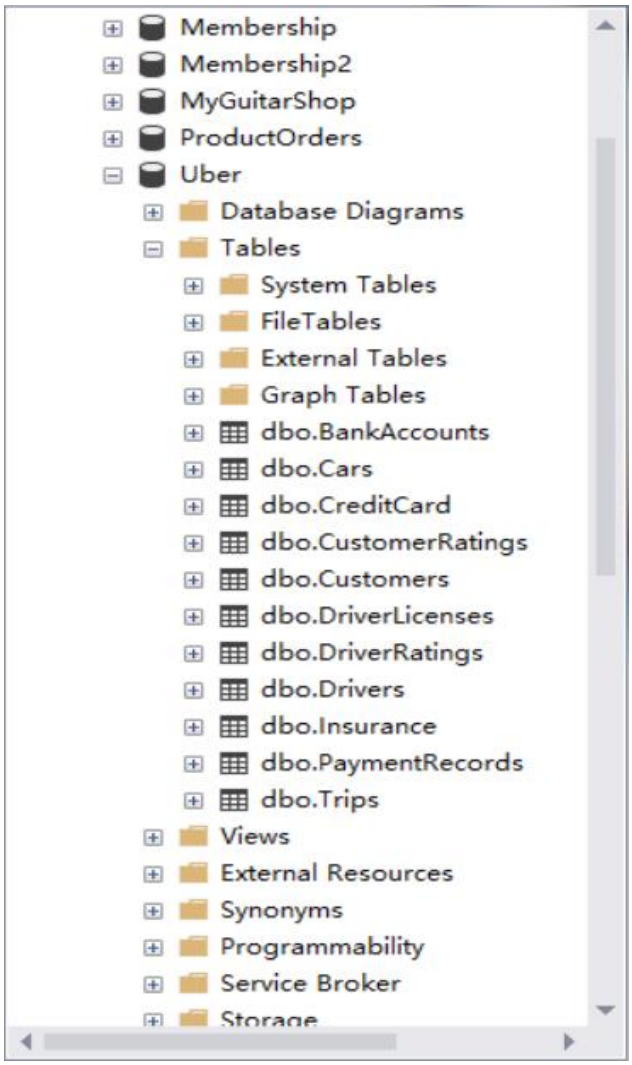
```
CREATE TABLE Insurance(
     InsuranceID INT PRIMARY KEY IDENTITY,
     Company VARCHAR(25) NOT NULL,
     PolicyNum VARCHAR(25) NOT NULL,
     IssueDate DATETIME NOT NULL,
     ExpiryDate DATETIME NOT NULL,
     CHECK (IssueDate <= ExpiryDate)
);

CREATE TABLE BankAccounts(
     AccountID  INT PRIMARY KEY IDENTITY,
     BankName VARCHAR(25) NOT NULL,
     AccountNum VARCHAR(16) NOT NULL,
     Type VARCHAR(25) DEFAULT 'saving',
);

CREATE TABLE Drivers(
     DriverID INT PRIMARY KEY IDENTITY,
     SSN CHAR(9) NOT NULL UNIQUE,
     DriverLicenseID INT REFERENCES DriverLicenses(DriverLicenseID),
     InsuranceID INT REFERENCES Insurance(InsuranceID),
     AccountID INT REFERENCES BankAccounts(AccountID),
     FirstName VARCHAR(50) NOT NULL,
     LastName VARCHAR(50) NOT NULL,
     Address1 VARCHAR(50) NOT NULL,
     Address2 VARCHAR(50) DEFAULT NULL,
     City VARCHAR(25) NOT NULL,
     State VARCHAR(25) NOT NULL,
     Phone VARCHAR(12) NOT NULL,
     StartTime DATETIME NOT NULL,
);

CREATE TABLE Trips(
     TripID INT PRIMARY KEY IDENTITY,
     DateBooked DATETIME NOT NULL,
     PickUpTime TIME NOT NULL,
     DropOffTime TIME NOT NULL,
     Completed VARCHAR(25) DEFAULT 'yes',
     PickUpAddress VARCHAR(50) NOT NULL,
     DropOffAddress VARCHAR(50) NOT NULL,
     DriverID INT REFERENCES Drivers(DriverID),
     Tips MONEY NOT NULL,
     CreditCardID INT REFERENCES CreditCard(CreditCardID),
);
```

```
CREATE TABLE PaymentRecords(
    PaymentID INT PRIMARY KEY IDENTITY,
    Date DATETIME NOT NULL,
    Amount MONEY NOT NULL,
    DriverID INT REFERENCES Drivers(DriverID),
);

CREATE TABLE DriverRatings(
    DRatingID INT PRIMARY KEY IDENTITY,
    RatingDate DATETIME NOT NULL,
    Score INT DEFAULT NULL,
    Text VARCHAR(50) DEFAULT NULL,
    TripID INT REFERENCES Trips(TripID),
);

CREATE TABLE Cars(
    CardID INT PRIMARY KEY IDENTITY,
    DriverID INT REFERENCES Drivers(DriverID),
    Model VARCHAR(50) NOT NULL,
    Make VARCHAR(50) NOT NULL,
    Year VARCHAR(10) NOT NULL,
    Color VARCHAR(25) NOT NULL,
    Class VARCHAR(25) DEFAULT 'regular',
);

CREATE TABLE CustomerRatings(
    CRatingID INT PRIMARY KEY IDENTITY,
    RatingDate DATETIME NOT NULL,
    Score INT DEFAULT NULL,
    Text VARCHAR(50) DEFAULT NULL,
    TripID INT REFERENCES Trips(TripID),
);
////////////////////////////////////////////////////////////////////////////////
```

The screenshots:


CreateUber.sql - MSI\SQLEXPRESS.Uber (MSI\tianjiahe189 (54)) - Microsoft SQL Server Management Studio
New Query
Execute
Debug
Uber
Object Explorer
Connect
MSI\SQLEXPRESS (SQL Server 14.0.
Databases
System Databases
Database Snapshots
AP
Examples
Membership
Membership2
MyGuitarShop
ProductOrders
Uber
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
CreateUber.sql -...tianjiahe189 (54))
SQLQuery1.sql - M...ianjiahe189 (53))
USE master
GO
IF DB_ID ('Uber') IS NOT NULL
DROP DATABASE Uber;
GO
CREATE DATABASE Uber;
GO
USE Uber;
CREATE TABLE Customers(
CustomerID INT PRIMARY KEY IDENTITY,
FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,
Address1 VARCHAR(50) NOT NULL,
Address2 VARCHAR(50) DEFAULT NULL,
City VARCHAR(25) NOT NULL,
State VARCHAR(25) NOT NULL,
Email VARCHAR(50) NOT NULL,
Phone VARCHAR(12) NOT NULL,
LastTripDate DATETIME NOT NULL,
TripsNum INT NOT NULL,
Status VARCHAR(25) DEFAULT 'inactive',
CHECK (Status='inactive' OR Status='active')
);
100 %
Messages
Commands completed successfully.
Query executed successfully.
MSI\SQLEXPRESS (14.0 RTM)
MSI\tianjiahe189 (54)
Uber
00:00:00
0 rows
就绪
行 45
列 1
字符 1
Ins


CreateUber.sql - MSI\SQLEXPRESS.Uber (MSI\tianjiahe189 (54)) - Microsoft SQL Server Management Studio
New Query
Execute
Debug
Uber
Object Explorer
Connect
MSI\SQLEXPRESS (SQL Server 14.0.
Databases
System Databases
Database Snapshots
AP
Examples
Membership
Membership2
MyGuitarShop
ProductOrders
Uber
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
CreateUber.sql -...tianjiahe189 (54))
SQLQuery1.sql - M...ianjiahe189 (53))
DriverLicenseID INT PRIMARY KEY IDENTITY,
State VARCHAR(25) NOT NULL,
IssueDate DATETIME NOT NULL,
ExpiryDate DATETIME NOT NULL,
LicenseNum VARCHAR(9) NOT NULL,
CHECK (IssueDate <= ExpiryDate)
);
CREATE TABLE Insurance(
InsuranceID INT PRIMARY KEY IDENTITY,
Company VARCHAR(25) NOT NULL,
PolicyNum VARCHAR(25) NOT NULL,
IssueDate DATETIME NOT NULL,
ExpiryDate DATETIME NOT NULL,
CHECK (IssueDate <= ExpiryDate)
);
CREATE TABLE BankAccounts(
AccountID INT PRIMARY KEY IDENTITY,
BankName VARCHAR(25) NOT NULL,
AccountNum VARCHAR(16) NOT NULL,
Type VARCHAR(25) DEFAULT 'saving',
);
CREATE TABLE Drivers(
DriverID INT PRIMARY KEY IDENTITY,
100 %
Messages
Commands completed successfully.
Query executed successfully.
MSI\SQLEXPRESS (14.0 RTM)
MSI\tianjiahe189 (54)
Uber
00:00:00
0 rows
就绪
行 45
列 1
字符 1
Ins



1.4 Insert some data

In this section I insert 5 record for each table.
Here is the source code:

```
////////////////////////////////////////////////////////////////////////////////////////
USE Uber;

INSERT INTO Customers([FirstName],[LastName],[Address1],[Address2],[City],
[State],[Email],[Phone],[LastTripDate],[TripsNum],[Status])

VALUES('Paul','George','325 Geneess St','nobhill
304','Syracuse','NY','PG123@gmail.com','3152487748','12/20/2018',1,'inactive'),
('Jimmy','Butler','325 Geneess St','nobhill
305','Syracuse','NY','JB123@gmail.com','3152487541','12/21/2018',1,'inactive'),
('Lebron','James','325 Geneess St','nobhill
```

```
306','Syracuse','NY','LJ123@gmail.com','3152485444','12/22/2018',1,'inactive'),
('Chris','Paul','700 Codland Ed','Theory
234','Newark','NJ','CJ123@gmail.com','3152487854','12/23/2018',1,'inactive'),
('James','Harden','700 Codland Ed','Theory
235','Newark','NJ','JH123@gmail.com','3152489845','12/24/2018',1,'inactive')

GO

INSERT INTO CreditCard([CustomerID],[CreditCardInfo])

VALUES(1,'7894-5845-5587-7413'),
(2,'7894-5845-5487-7467'),
(3,'7894-5845-5434-7456'),
(4,'7894-5845-5323-7421'),
(5,'7894-5845-5233-7423')

GO

INSERT INTO DriverLicenses([State],[IssueDate],[ExpiryDate],[LicenseNum])

VALUES('NY','1/1/2015','1/1/2020','895647125'),
('NY','1/1/2015','1/1/2021','895647156'),
('NY','1/1/2015','1/1/2020','895647134'),
('NJ','1/1/2016','1/1/2022','895647123'),
('NJ','1/1/2016','1/1/2021','895647112')

GO

INSERT INTO Insurance([Company],[PolicyNum],[IssueDate],[ExpiryDate])

VALUES('Ehome','54785965145','8/30/2016','8/30/2020'),
('Ehome','54785965144','8/30/2016','8/30/2020'),
('Ehome','54785965143','8/30/2016','8/30/2020'),
('Iheath','54785965458','8/30/2016','8/30/2020'),
('Iheath','54785965459','8/30/2016','8/30/2020')

GO

INSERT INTO BankAccounts([BankName],[AccountNum],[Type])

VALUES('Chase','4565874596547458','saving'),
('Chase','4565874596542342','saving'),
('Chase','4565874596549876','saving'),
('Chase','4565874596548452','college'),
```

('Chase','4565874596546324','college')

GO

INSERT INTO
Drivers([SSN],[DriverLicenseID],[InsuranceID],[AccountID],[FirstName],[LastName],[Address1],[Address2],
[City],[State],[Phone],[StartTime])

VALUES('123456789',1,1,1,'Kevin','Durant','451 Brookest Ed','Westlife
212','Syracuse','NY','3154587414','5/20/2018'),
('123456790',2,2,2,'Stephen','Curry','451 Brookest Ed','Westlife
212','Syracuse','NY','3154583423','5/20/2018'),
('123456791',3,3,3,'Russell','Westbrook','451 Brookest Ed','Westlife
212','Syracuse','NY','3154586547','5/20/2018'),
('123456792',4,4,4,'Damian','Lillard','855 Linked St','Cbc
112','Newark','NJ','3152188456','5/20/2018'),
('123456793',5,5,5,'Blake','Griffin','855 Linked St','Cbc
113','Newark','NJ','3152387413','5/20/2018')

GO

INSERT INTO
Trips([DateBooked],[PickUpTime],[DropOffTime],[Completed],[PickUpAddress],[DropOffAddress],[DriverID],[Tips],
[CreditCardID])

VALUES('12/20/2018','13:45:30','14:45:00','yes','374 Marland St','SyracuseU',1,'5',1),
('12/21/2018','14:12:30','15:44:00','yes','374 Marland St','SyracuseU',2,'5',2),
('12/22/2018','15:15:30','16:34:00','yes','374 Marland St','SyracuseU',3,'5',3),
('12/23/2018','16:35:30','17:24:00','yes','454 Marrypark St','SyracuseU',4,'5',4),
('12/24/2018','17:25:30','18:14:00','yes','454 Marrypark St','SyracuseU',5,'5',5)

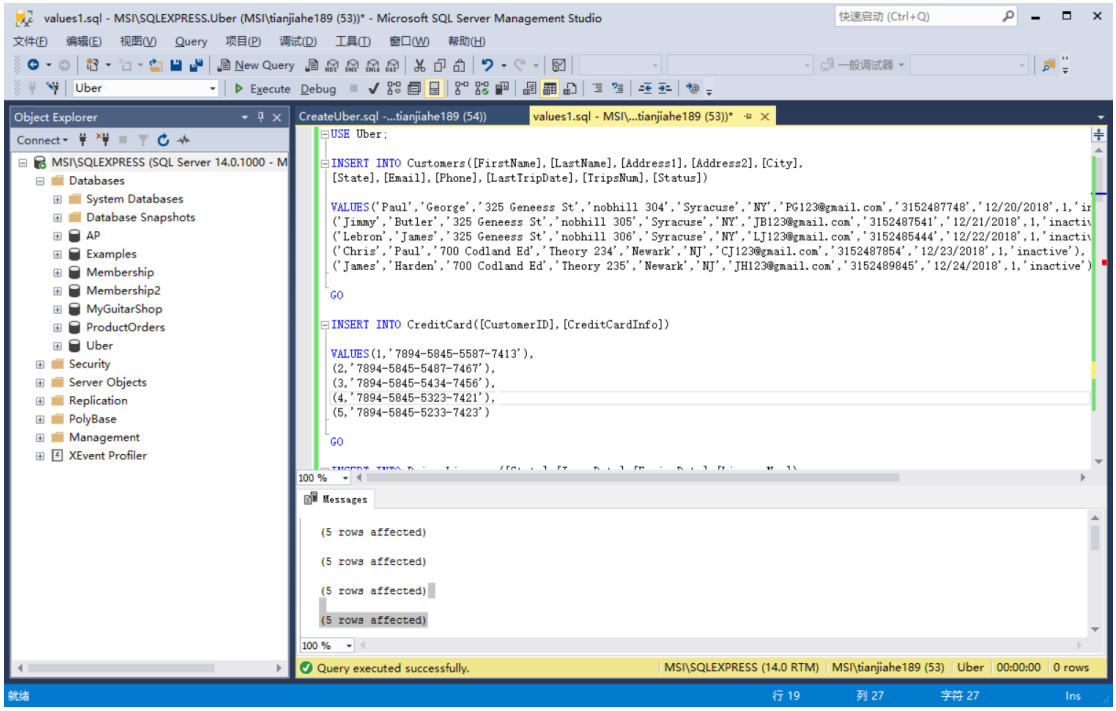
GO

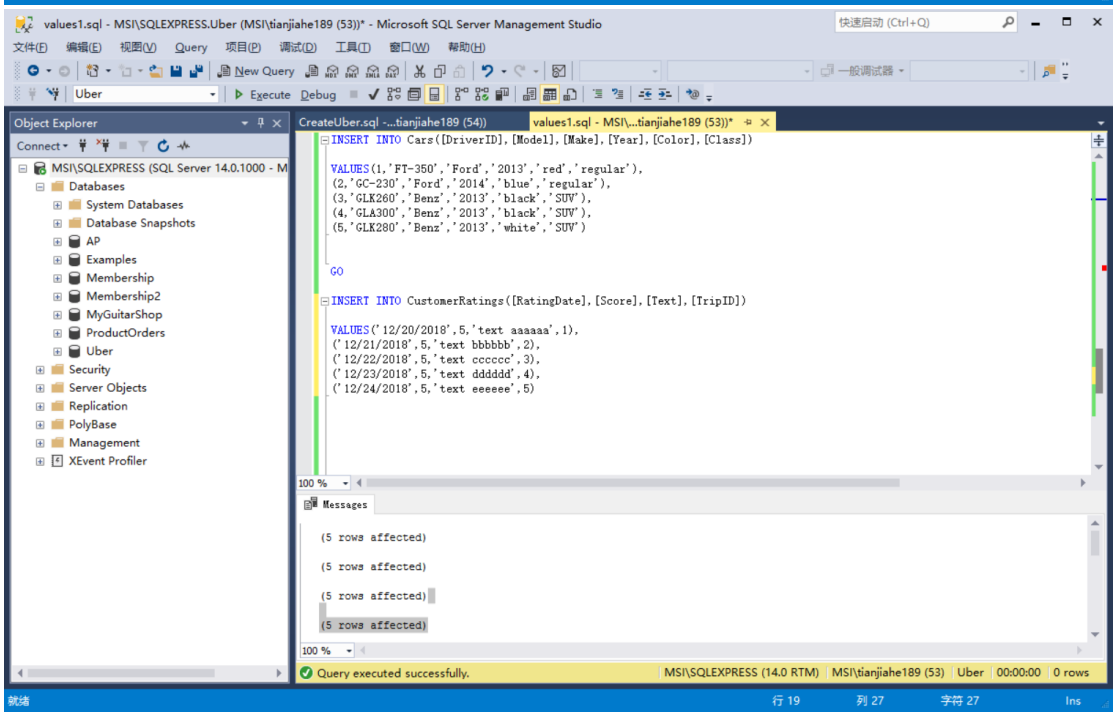
INSERT INTO PaymentRecords([Date],[Amount],[DriverID])

VALUES('12/20/2018','35',1),
('12/21/2018','32',2),
('12/22/2018','23',3),
('12/23/2018','27',4),
('12/24/2018','15',5)

```
GO

INSERT INTO DriverRatings([RatingDate],[Score],[Text],[TripID])

VALUES('12/20/2018',5,'text aaaaaa',1),
('12/21/2018',5,'text bbbbbb',2),
('12/22/2018',5,'text cccccc',3),
('12/23/2018',5,'text dddddd',4),
('12/24/2018',5,'text eeeeee',5)

GO

INSERT INTO Cars([DriverID],[Model],[Make],[Year],[Color],[Class])

VALUES(1,'FT-350','Ford','2013','red','regular'),
(2,'GC-230','Ford','2014','blue','regular'),
(3,'GLK260','Benz','2013','black','SUV'),
(4,'GLA300','Benz','2013','black','SUV'),
(5,'GLK280','Benz','2013','white','SUV')


GO

INSERT INTO CustomerRatings([RatingDate],[Score],[Text],[TripID])

VALUES('12/20/2018',5,'text aaaaaa',1),
('12/21/2018',5,'text bbbbbb',2),
('12/22/2018',5,'text cccccc',3),
('12/23/2018',5,'text dddddd',4),
('12/24/2018',5,'text eeeeee',5)
////////////////////////////////////////////////////////////////////////////////
```

The screenshots:


values1.sql - MSI\SQLEXPRESS.Uber (MSI\tianjiahe189 (53))* - Microsoft SQL Server Management Studio
Object Explorer
CreateUber.sql -...tianjiahe189 (54))
values1.sql - MSI\...tianjiahe189 (53))*
USE Uber;
INSERT INTO Customers([FirstName],[LastName],[Address1],[Address2],[City],
[State],[Email],[Phone],[LastTripDate],[TripsNum],[Status])
VALUES('Paul','George','325 Geneess St','nobhill 304','Syracuse','NY','PG123@gmail.com','3152487748','12/20/2018',1,'ir
('Jimmy','Butler','325 Geneess St','nobhill 305','Syracuse','NY','JB123@gmail.com','3152487541','12/21/2018',1,'inactiv
('Lebron','James','325 Geneess St','nobhill 306','Syracuse','NY','LJ123@gmail.com','3152485444','12/22/2018',1,'inactiv
('Chris','Paul','700 Codland Ed','Theory 234','Newark','NJ','CJ123@gmail.com','3152487854','12/23/2018',1,'inactive'),
('James','Harden','700 Codland Ed','Theory 235','Newark','NJ','JH123@gmail.com','3152489845','12/24/2018',1,'inactive')
GO
INSERT INTO CreditCard([CustomerID],[CreditCardInfo])
VALUES(1,'7894-5845-5587-7413'),
(2,'7894-5845-5487-7467'),
(3,'7894-5845-5434-7456'),
(4,'7894-5845-5323-7421'),
(5,'7894-5845-5233-7423')
GO
Messages
(5 rows affected)
(5 rows affected)
(5 rows affected)
(5 rows affected)
Query executed successfully.
MSI\SQLEXPRESS (14.0 RTM)
MSI\tianjiahe189 (53)
Uber
00:00:00
0 rows
就绪
行 19
列 27
字符 27
Ins

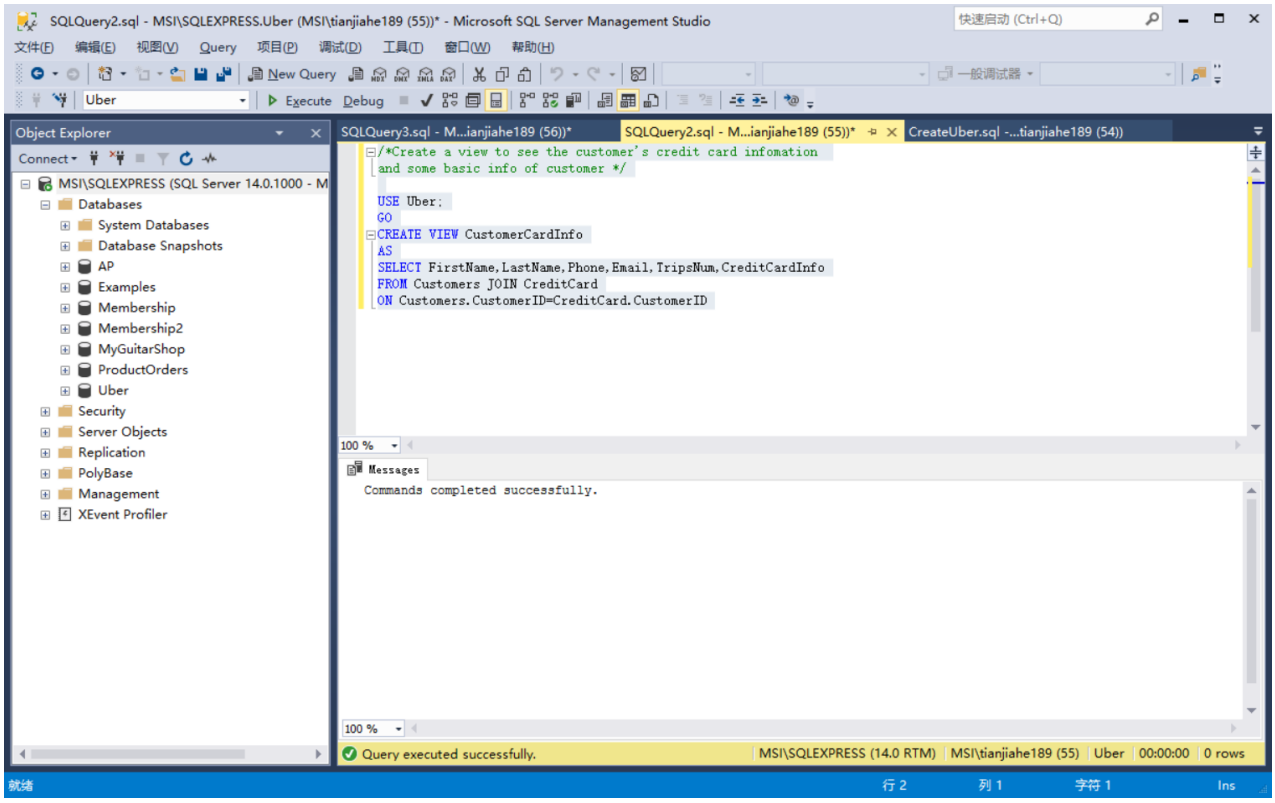

values1.sql - MSI\SQLEXPRESS.Uber (MSI\tianjiahe189 (53))* - Microsoft SQL Server Management Studio
Object Explorer
CreateUber.sql -...tianjiahe189 (54))
values1.sql - MSI\...tianjiahe189 (53))*
INSERT INTO Cars([DriverID],[Model],[Make],[Year],[Color],[Class])
VALUES(1,'FT-350','Ford','2013','red','regular'),
(2,'GC-230','Ford','2014','blue','regular'),
(3,'GLK260','Benz','2013','black','SUV'),
(4,'GLA300','Benz','2013','black','SUV'),
(5,'GLK280','Benz','2013','white','SUV')
GO
INSERT INTO CustomerRatings([RatingDate],[Score],[Text],[TripID])
VALUES('12/20/2018',5,'text aaaaaa',1),
('12/21/2018',5,'text bbbbbb',2),
('12/22/2018',5,'text cccccc',3),
('12/23/2018',5,'text dddddd',4),
('12/24/2018',5,'text eeeeee',5)
Messages
(5 rows affected)
(5 rows affected)
(5 rows affected)
(5 rows affected)
Query executed successfully.
MSI\SQLEXPRESS (14.0 RTM)
MSI\tianjiahe189 (53)
Uber
00:00:00
0 rows
就绪
行 19
列 27
字符 27
Ins

2. The Views Design

In this section, I will show the implementation of each views, their testcase and screenshots.

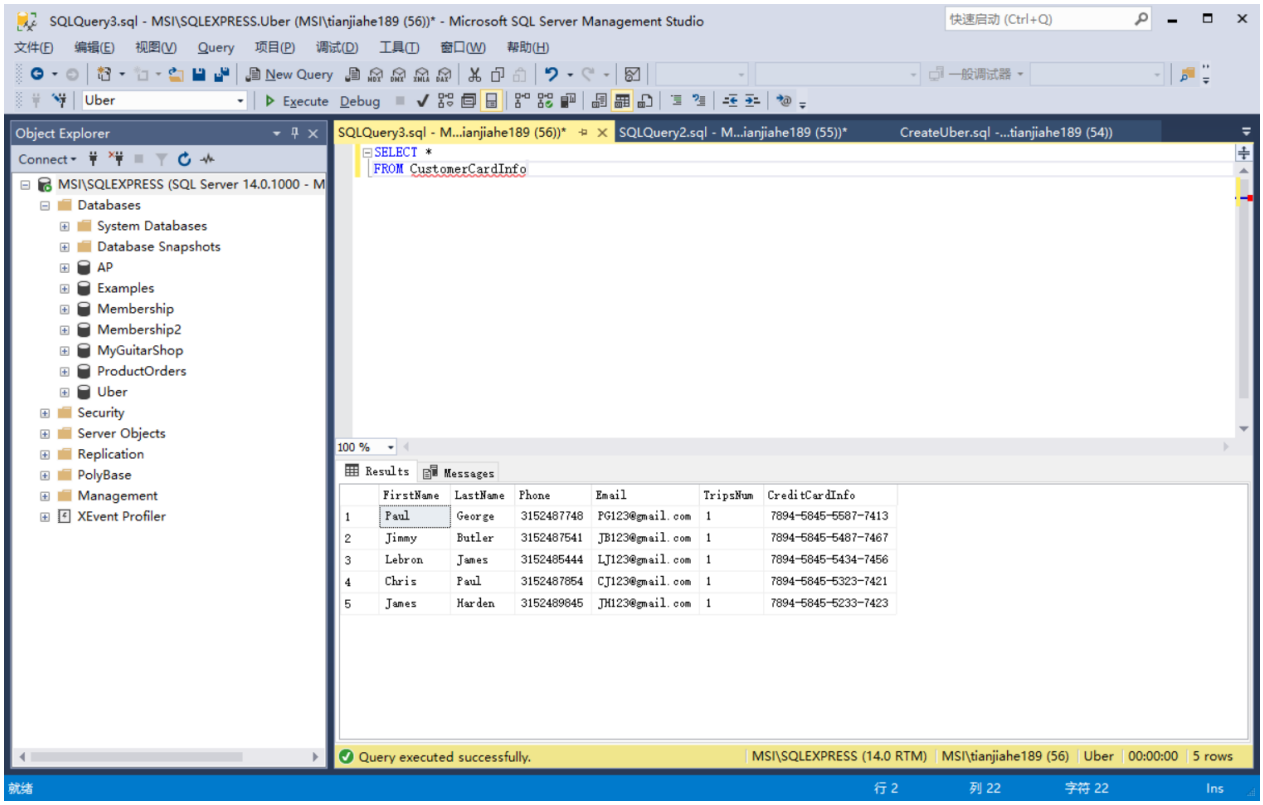
2.1 View 1 and test

```
/*Create a view to see the customer's credit card information
and some basic info of customer */

USE Uber;
GO
CREATE VIEW CustomerCardInfo
AS
SELECT FirstName,LastName,Phone,Email,TripsNum,CreditCardInfo
FROM Customers JOIN CreditCard
ON Customers.CustomerID=CreditCard.CustomerID
```



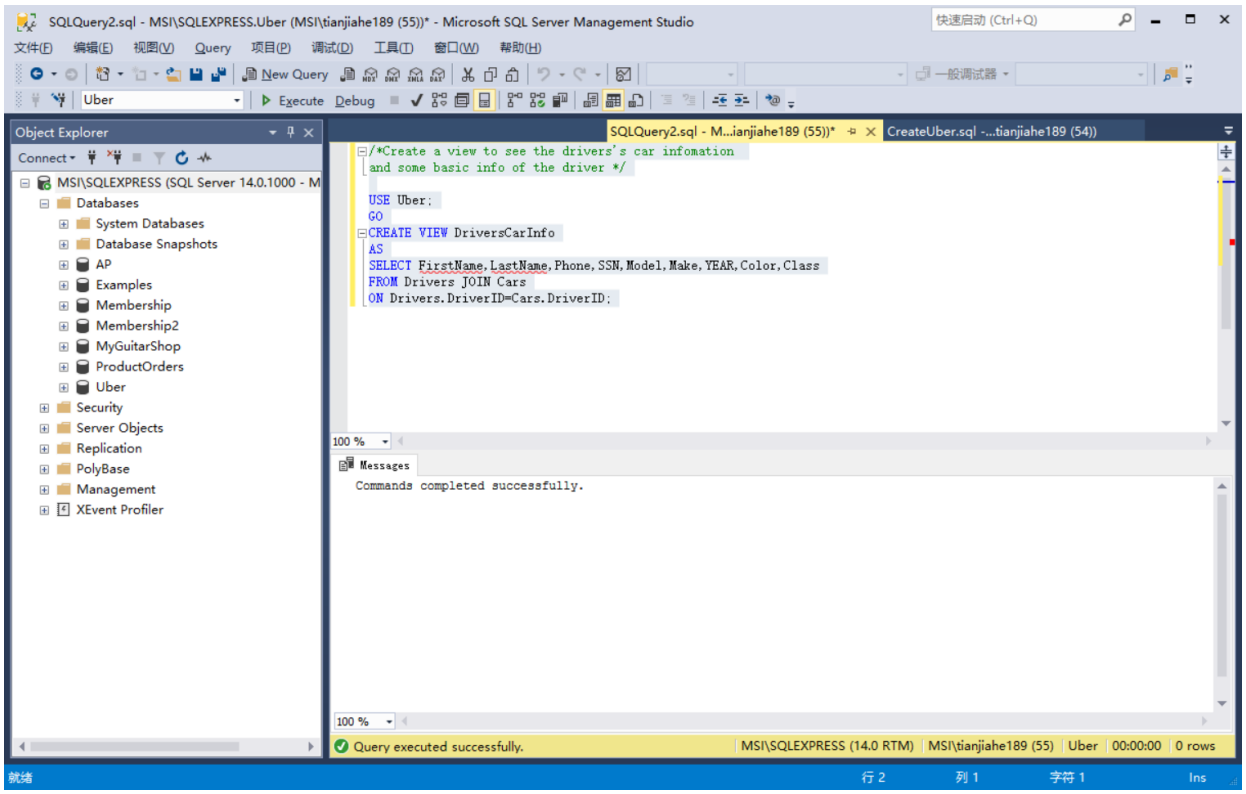
Testcase and screenshots:

```
SELECT *
FROM CustomerCardInfo;
```



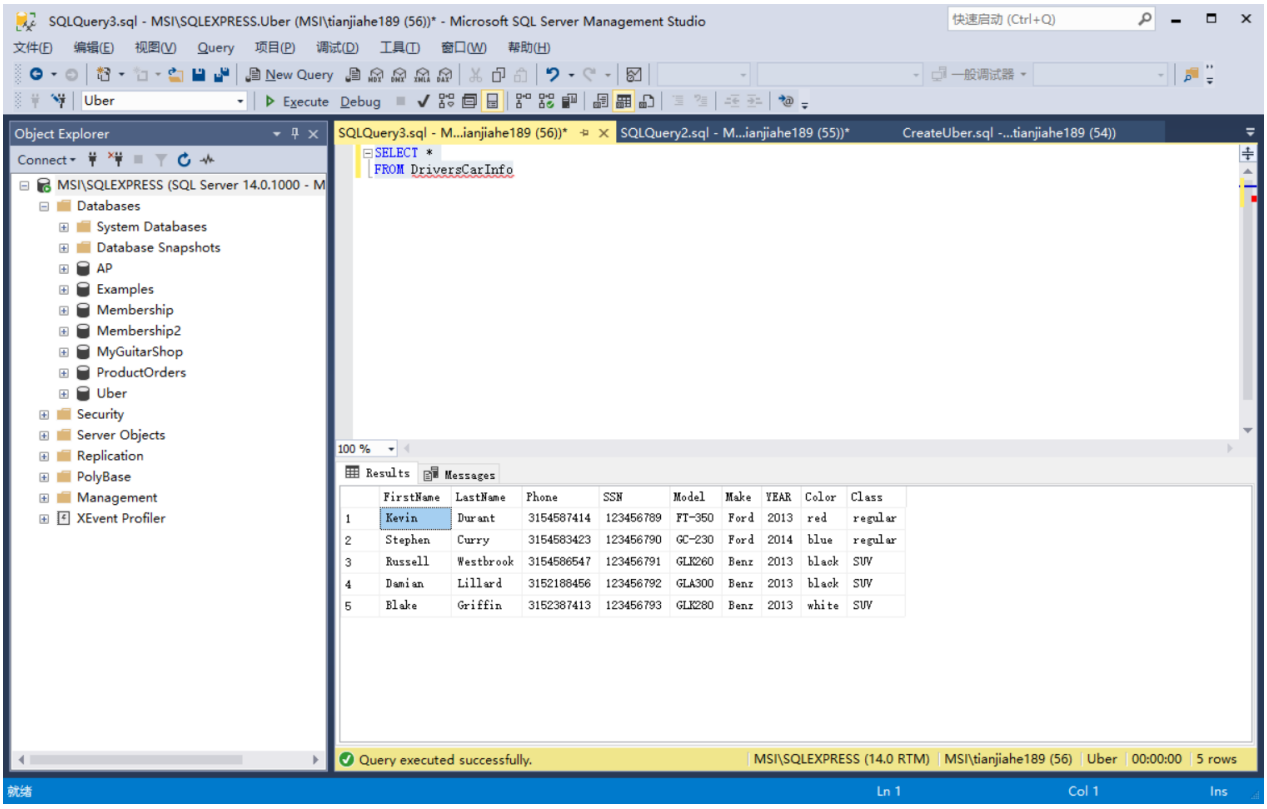
2.2 View 2 and test

```
/*Create a view to see the car's information
and some basic info of the driver */

USE Uber;
GO
CREATE VIEW DriversCarInfo
AS
SELECT FirstName, LastName, Phone, SSN, Model, Make, YEAR, Color, Class
FROM Drivers JOIN Cars
ON Drivers.DriverID=Cars.DriverID;
```



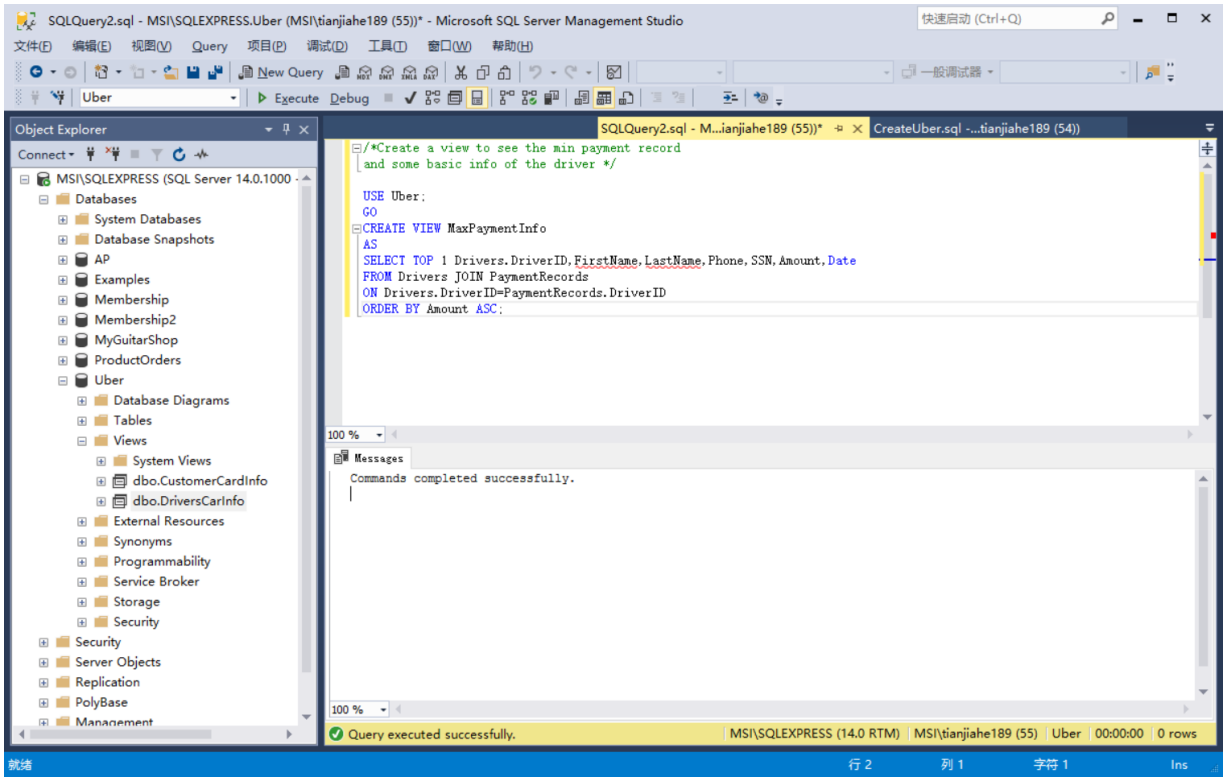
Testcase and screenshots:

```
SELECT *
FROM DriversCarInfo
```



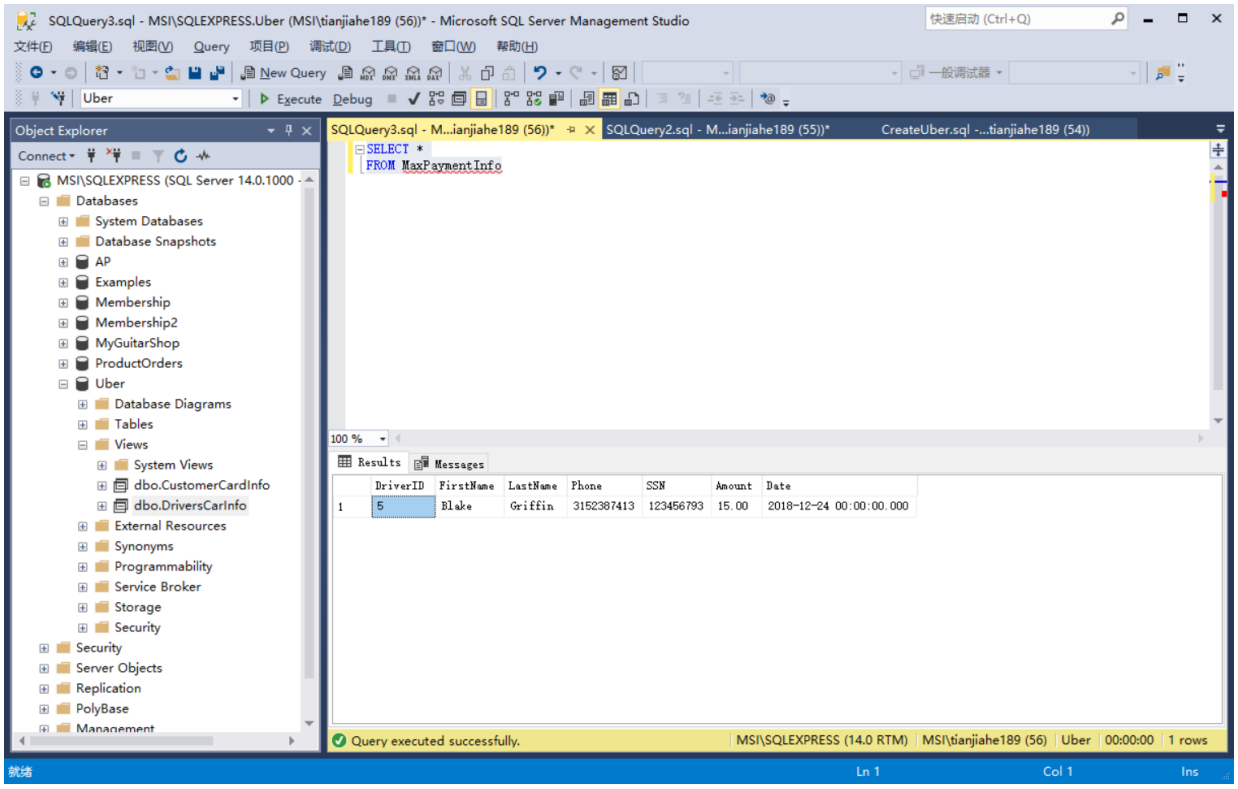
2.3 View 3 and test

```
/*Create a view to see the min payment record
and some basic info of the driver */

USE Uber;
GO
CREATE VIEW MaxPaymentInfo
AS
SELECT TOP 1 Drivers.DriverID,FirstName,LastName,Phone,SSN,Amount,Date
FROM Drivers JOIN PaymentRecords
ON Drivers.DriverID=PaymentRecords.DriverID
ORDER BY Amount ASC;
```



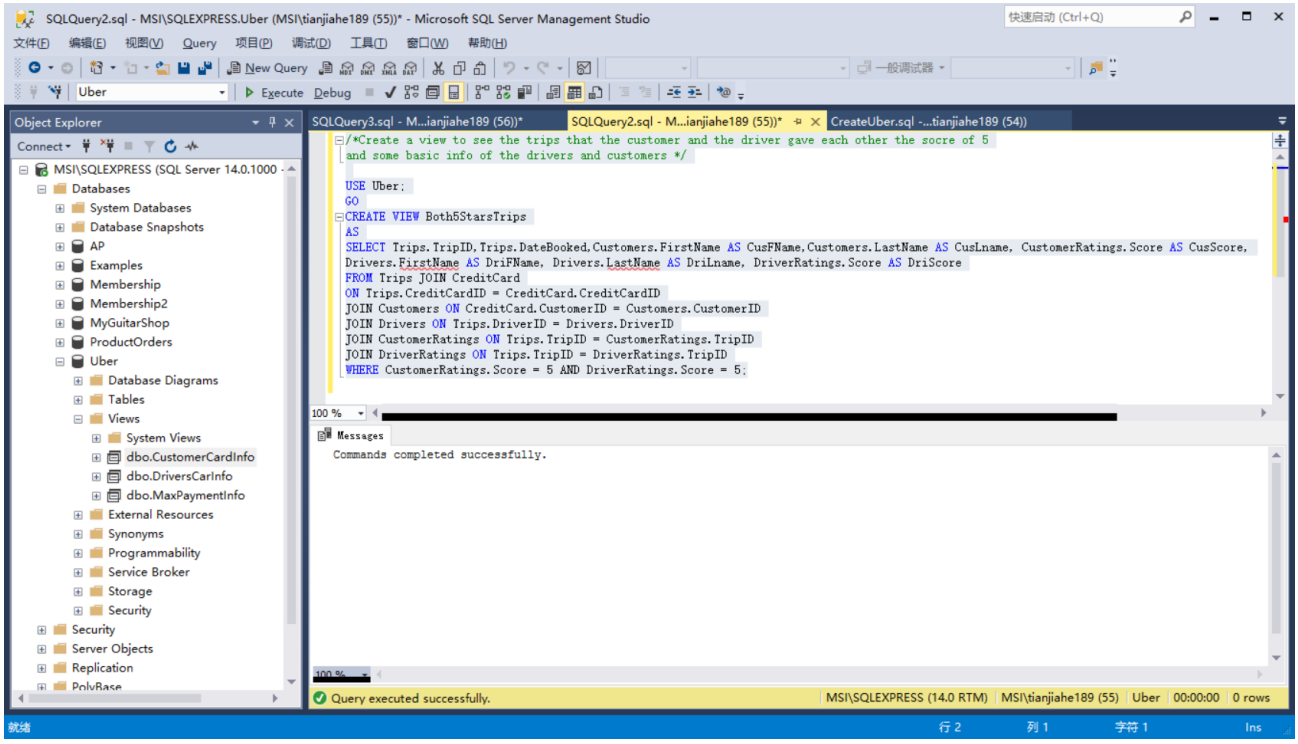
Testcase and screenshots:

```
SELECT *
FROM MaxPaymentInfo
```



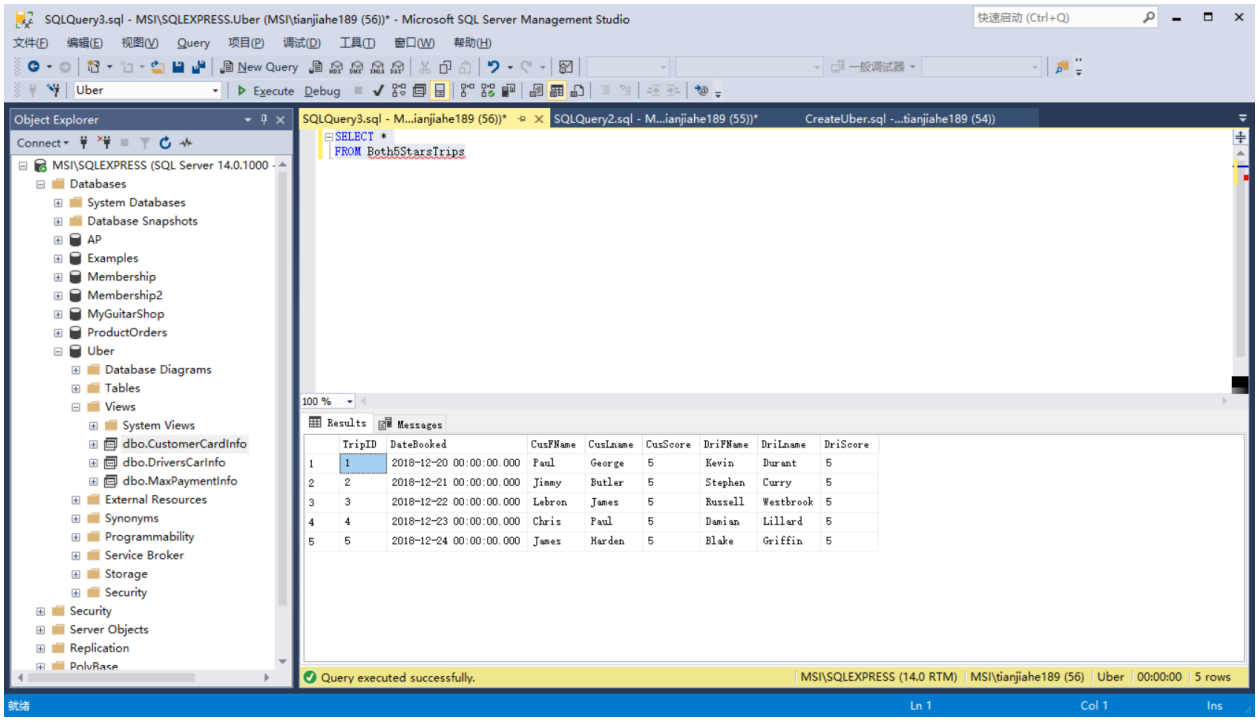
2.4 View 4 and test

```
/*Create a view to see the trips that the customer and the driver gave each other the score of 5 and some basic info of the drivers and customers */

USE Uber;
GO
CREATE VIEW Both5StarsTrips
AS
SELECT Trips.TripID,Trips.DateBooked,Customers.FirstName AS CusFName,Customers.LastName AS CusLname, CustomerRatings.Score AS CusScore,
Drivers.FirstName AS DriFName, Drivers.LastName AS DriLname, DriverRatings.Score AS DriScore
FROM Trips JOIN CreditCard
ON Trips.CreditCardID = CreditCard.CreditCardID
JOIN Customers ON CreditCard.CustomerID = Customers.CustomerID
JOIN Drivers ON Trips.DriverID = Drivers.DriverID
JOIN CustomerRatings ON Trips.TripID = CustomerRatings.TripID
JOIN DriverRatings ON Trips.TripID = DriverRatings.TripID
WHERE CustomerRatings.Score = 5 AND DriverRatings.Score = 5;
```



Testcase and screenshots:

```
SELECT *
FROM Both5StarsTrips
```



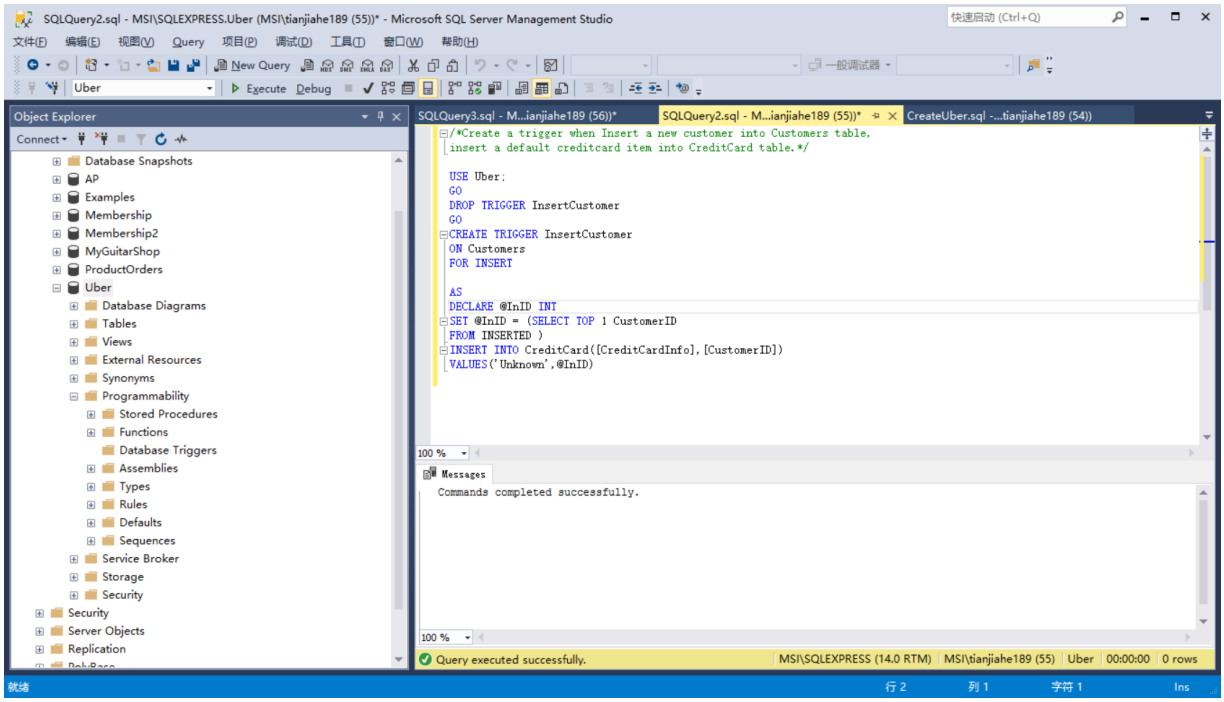
//

3. Triggers and Constraints

In this section I will show the implementation of the triggers and constraints. Each of them has the testcase and screenshots.

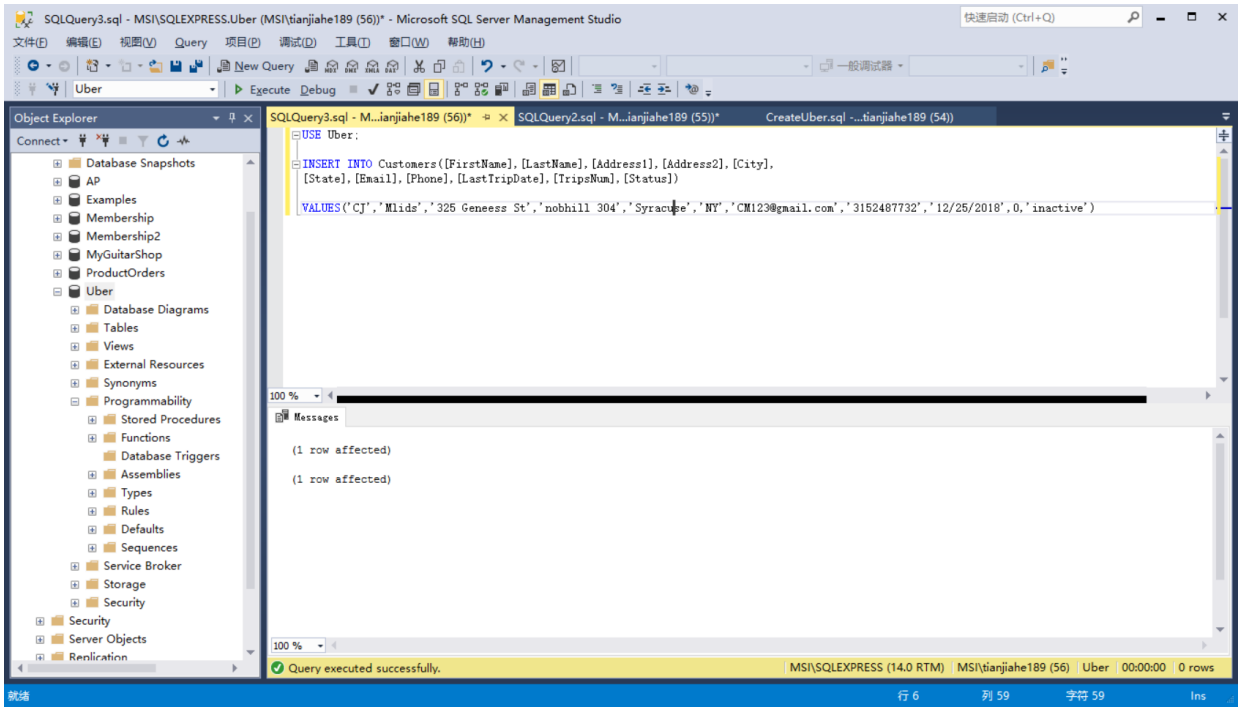
3.1 Trigger 1 and test.

```
/*Create a trigger when Insert a new customer into Customers table,
insert a default creditcard item into CreditCard table.*/

USE Uber;
GO
DROP TRIGGER InsertCustomer
GO
CREATE TRIGGER InsertCustomer
ON Customers
FOR INSERT

AS
DECLARE @InID INT
SET @InID = (SELECT TOP 1 CustomerID
FROM INSERTED )
INSERT INTO CreditCard([CreditCardInfo],[CustomerID])
VALUES('Unknown',@InID)
```

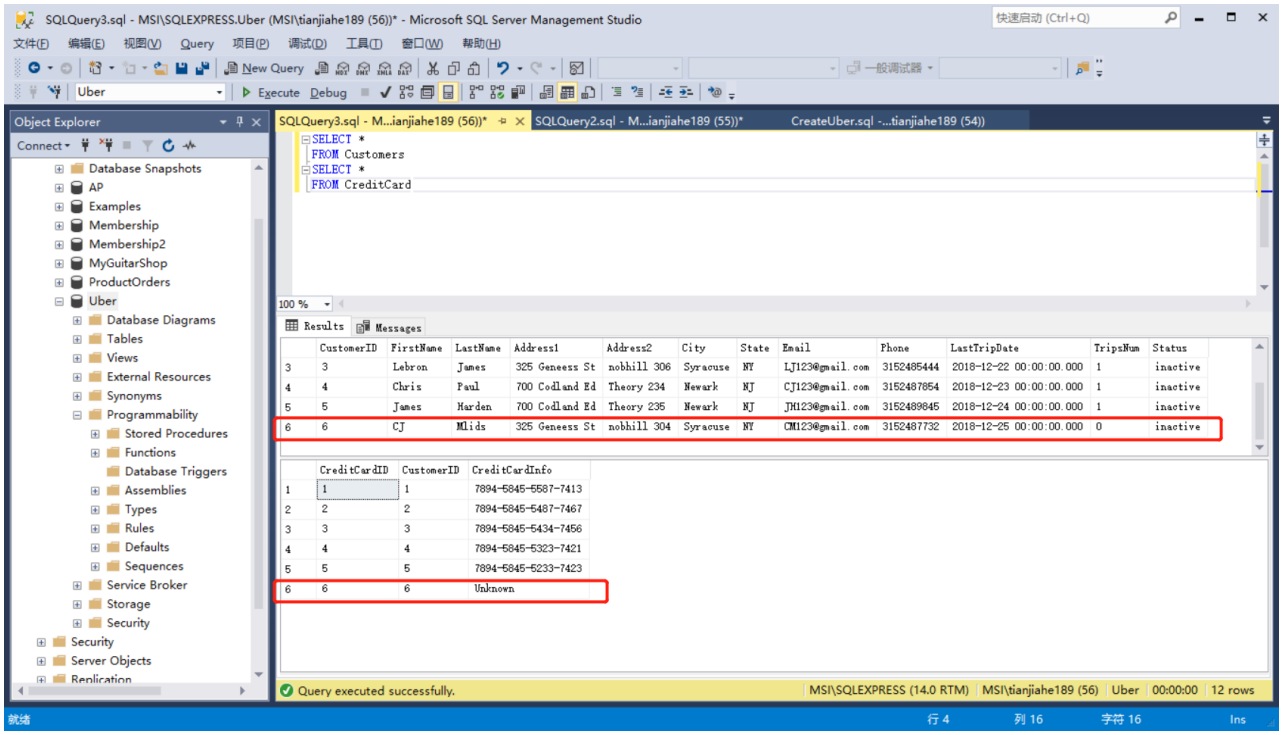


Testcase and screenshots:

```
USE Uber;

INSERT INTO Customers([FirstName],[LastName],[Address1],[Address2],[City],
[State],[Email],[Phone],[LastTripDate],[TripsNum],[Status])

VALUES('CJ','Mlids','325 Geneess St','nobhill
304','Syracuse','NY','CM123@gmail.com','3152487732','12/25/2018',0,'inactive')
```

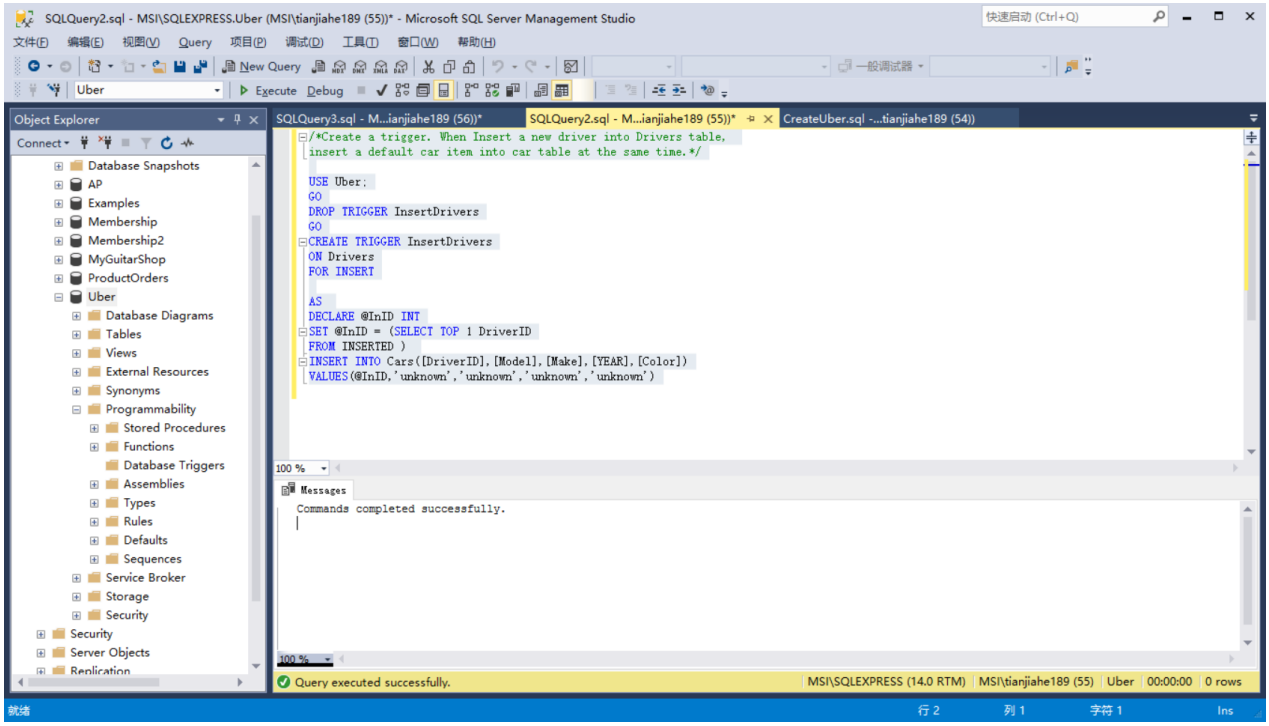



```
SELECT *
FROM Customers
SELECT *
FROM CreditCard
```



3.2 Trigger 2 and test.

```
/*Create a trigger. When Insert a new driver into Drivers table,
insert a default car item into car table at the same time.*/

USE Uber;
GO
DROP TRIGGER InsertDrivers
GO
CREATE TRIGGER InsertDrivers
ON Drivers
FOR INSERT

AS
DECLARE @InID INT
SET @InID = (SELECT TOP 1 DriverID
FROM INSERTED )
INSERT INTO Cars([DriverID],[Model],[Make],[YEAR],[Color])
VALUES(@InID,'unknown','unknown','unknown','unknown')
```

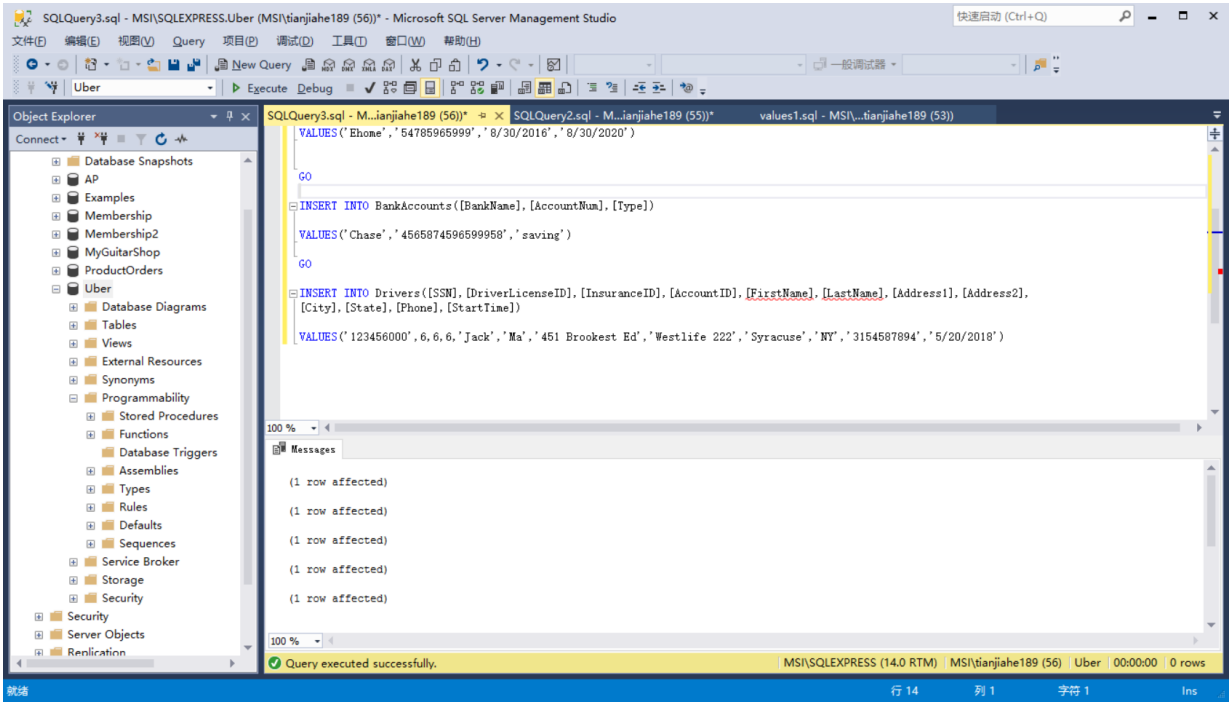


Testcase and screenshots:

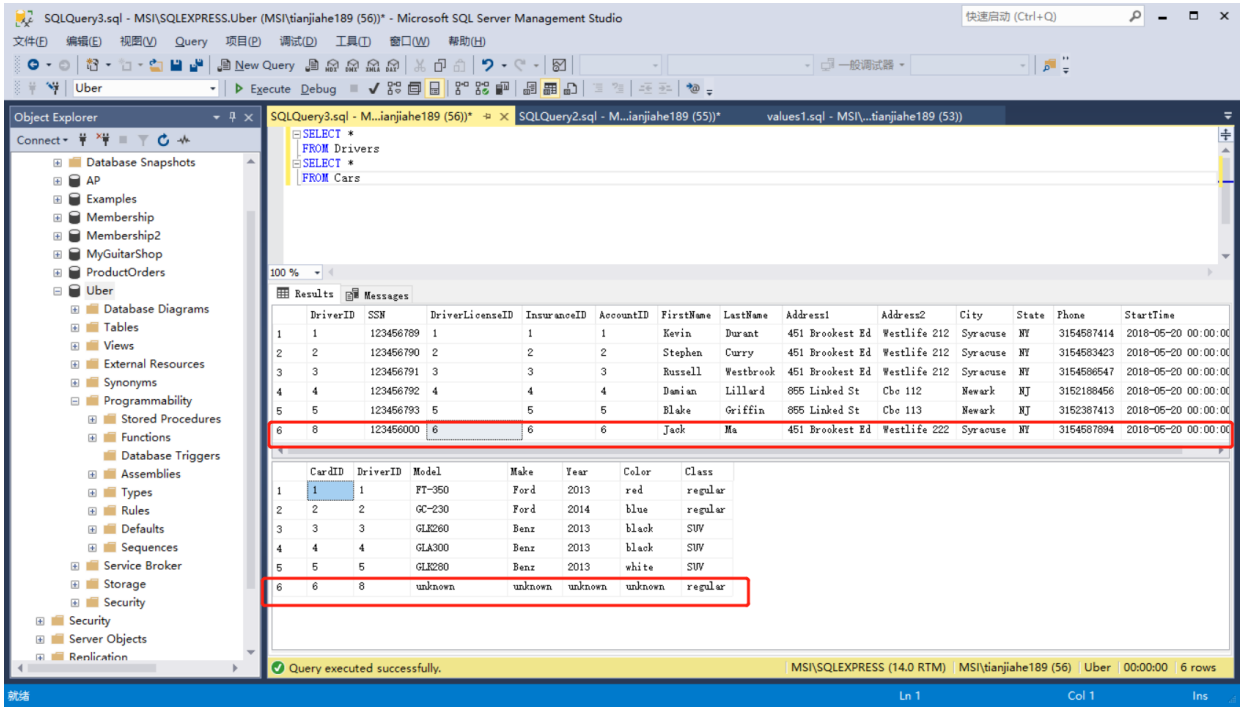
Here we should insert DriverLicenses, Insurance, BankAccounts first to insert a driver.

```
INSERT INTO DriverLicenses([State],[IssueDate],[ExpiryDate],[LicenseNum])
VALUES('NY','1/1/2015','1/1/2020','895647999')

GO
INSERT INTO Insurance([Company],[PolicyNum],[IssueDate],[ExpiryDate])
VALUES('Ehome','54785965999','8/30/2016','8/30/2020')

GO
INSERT INTO BankAccounts([BankName],[AccountNum],[Type])
VALUES('Chase','4565874596599958','saving')
GO
```

```
INSERT INTO
Drivers([SSN],[DriverLicenseID],[InsuranceID],[AccountID],[FirstName],[LastName],[Addre
ss1],[Address2],
[City],[State],[Phone],[StartTime])

VALUES('123456000',6,6,6,'Jack','Ma','451 Brookest Ed','Westlife
222','Syracuse','NY','3154587894','5/20/2018')
```




```
SELECT *
FROM Drivers
SELECT *
FROM Cars
```



3.3 constraints

```
CREATE TABLE Customers(
    CustomerID INT PRIMARY KEY IDENTITY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Address1 VARCHAR(50) NOT NULL,
    Address2 VARCHAR(50) DEFAULT NULL,
    City VARCHAR(25) NOT NULL,
    State VARCHAR(25) NOT NULL,
    Email VARCHAR(50) NOT NULL,
    Phone VARCHAR(12) NOT NULL,
    LastTripDate DATETIME NOT NULL,
    TripsNum INT NOT NULL,
    Status VARCHAR(25) DEFAULT 'inactive',
    CHECK (Status='inactive' OR Status='active')
);
```

Here I checked the customer's status, it can only be 'inactive' or 'active'.

```
////////////////////////////////////////////////////////////////////////////////
CREATE TABLE DriverLicenses(
    DriverLicenseID INT PRIMARY KEY IDENTITY,
```

```
    State VARCHAR(25) NOT NULL,
    IssueDate DATETIME NOT NULL,
    ExpiryDate DATETIME NOT NULL,
    LicenseNum VARCHAR(9) NOT NULL,
    CHECK (IssueDate <= ExpiryDate)
);



CREATE TABLE Insurance(
    InsuranceID INT PRIMARY KEY IDENTITY,
    Company VARCHAR(25) NOT NULL,
    PolicyNum VARCHAR(25) NOT NULL,
    IssueDate DATETIME NOT NULL,
    ExpiryDate DATETIME NOT NULL,
    CHECK (IssueDate <= ExpiryDate)
);
```

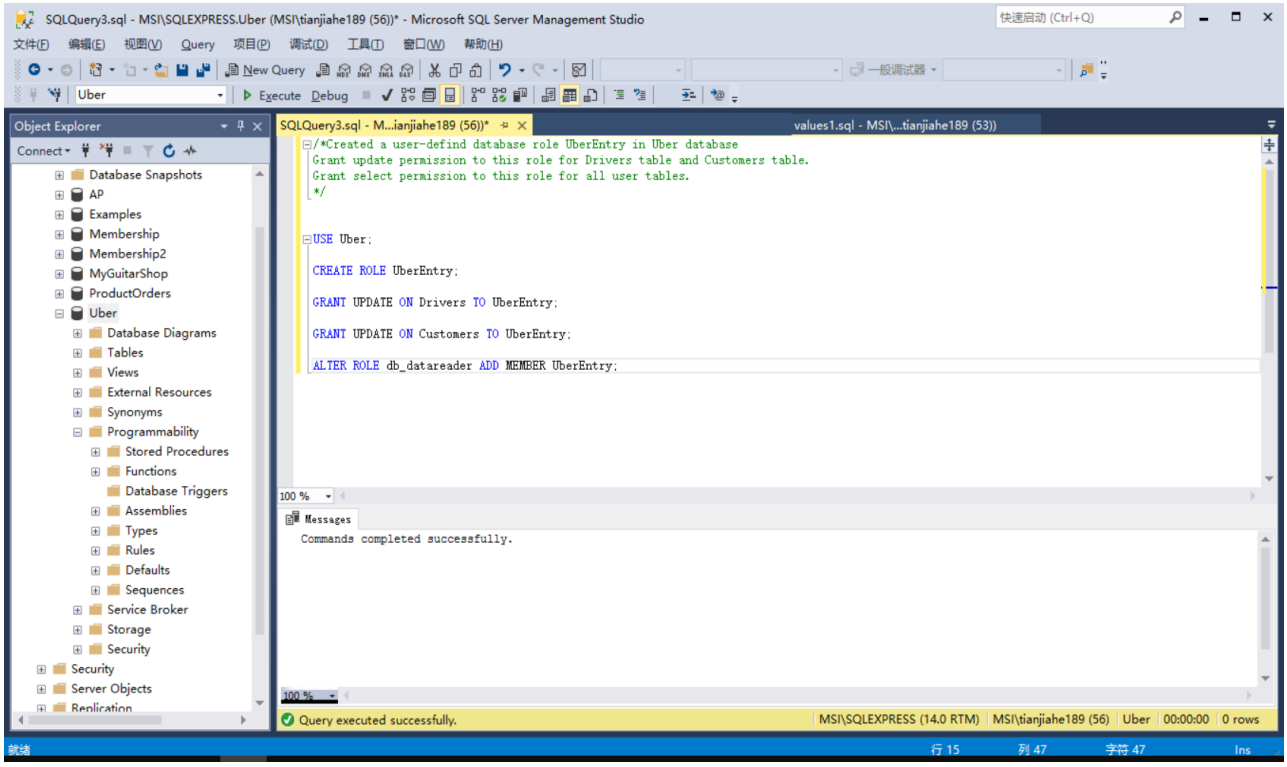
Here I checked the Issue date and expiry date of insurance and Driver Licenses. You can't insert a row with the IssueDate later than ExpiryDate.

4. Security levels

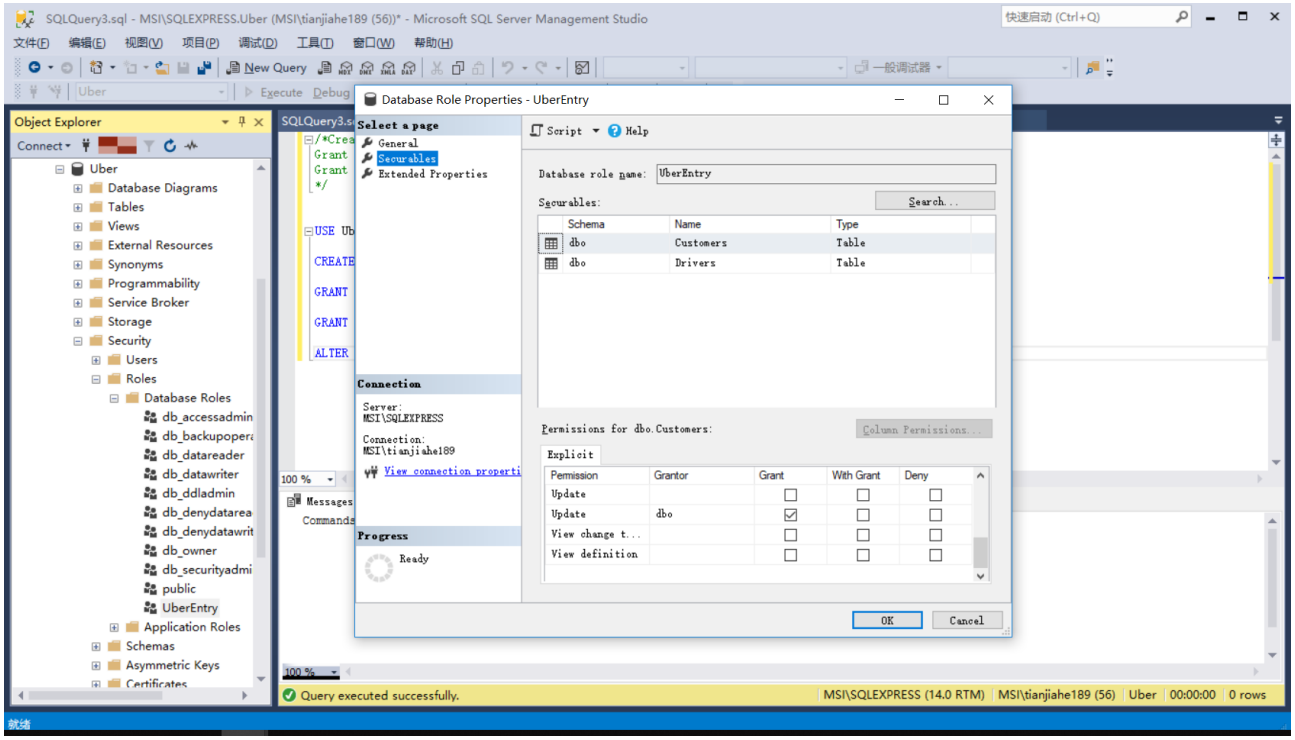
4.1 Create security role

```
/*Created a user-defind database role UberEntry in Uber database
Grant update permission to this role for Drivers table and Customers table.
Grant select permission to this role for all user tables.
*/


USE Uber;

CREATE ROLE UberEntry;
```

```
GRANT UPDATE ON Drivers TO UberEntry;

GRANT UPDATE ON Customers TO UberEntry;

ALTER ROLE db_datareader ADD MEMBER UberEntry;
```



Testcase and screenshots:

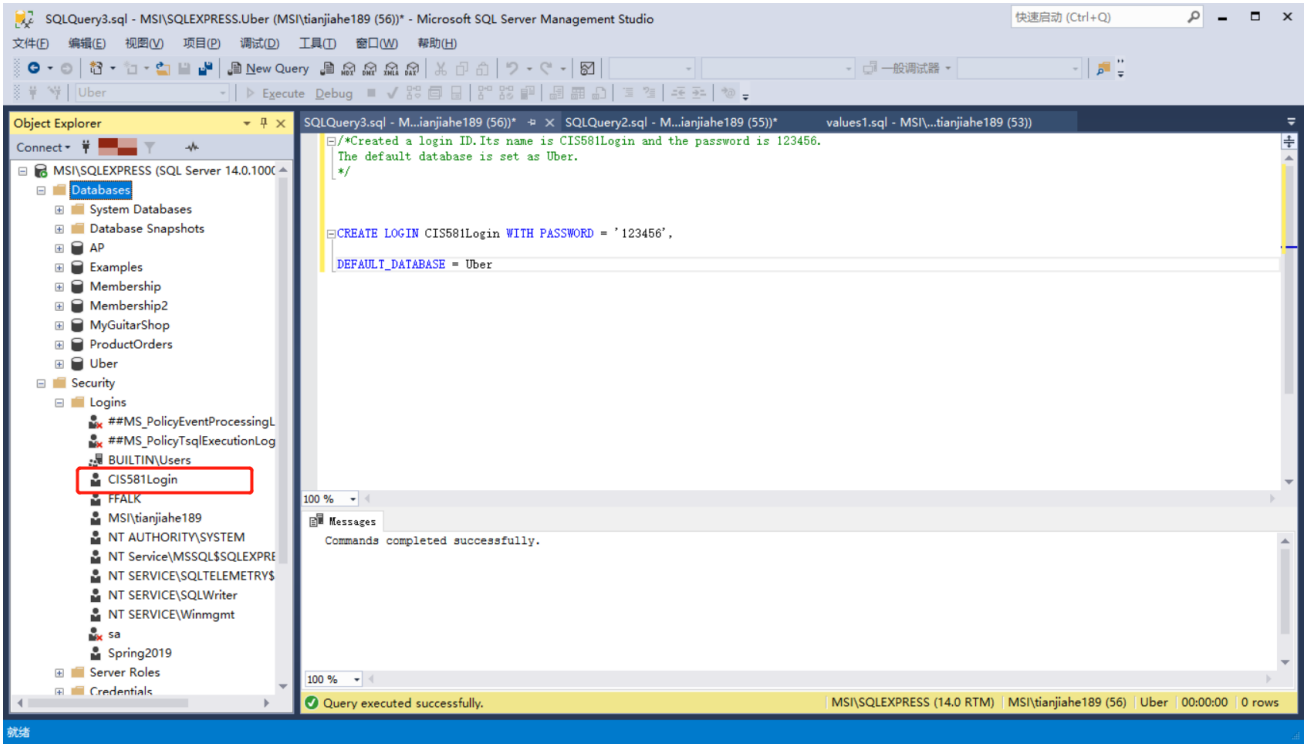


//

4.2 Create login with password

```
/*Created a login ID.Its name is CIS581Login and the password is 123456.
The default database is set as Uber.
*/

CREATE LOGIN CIS581Login WITH PASSWORD = '123456',

DEFAULT_DATABASE = Uber
```

Testcase and screenshots:



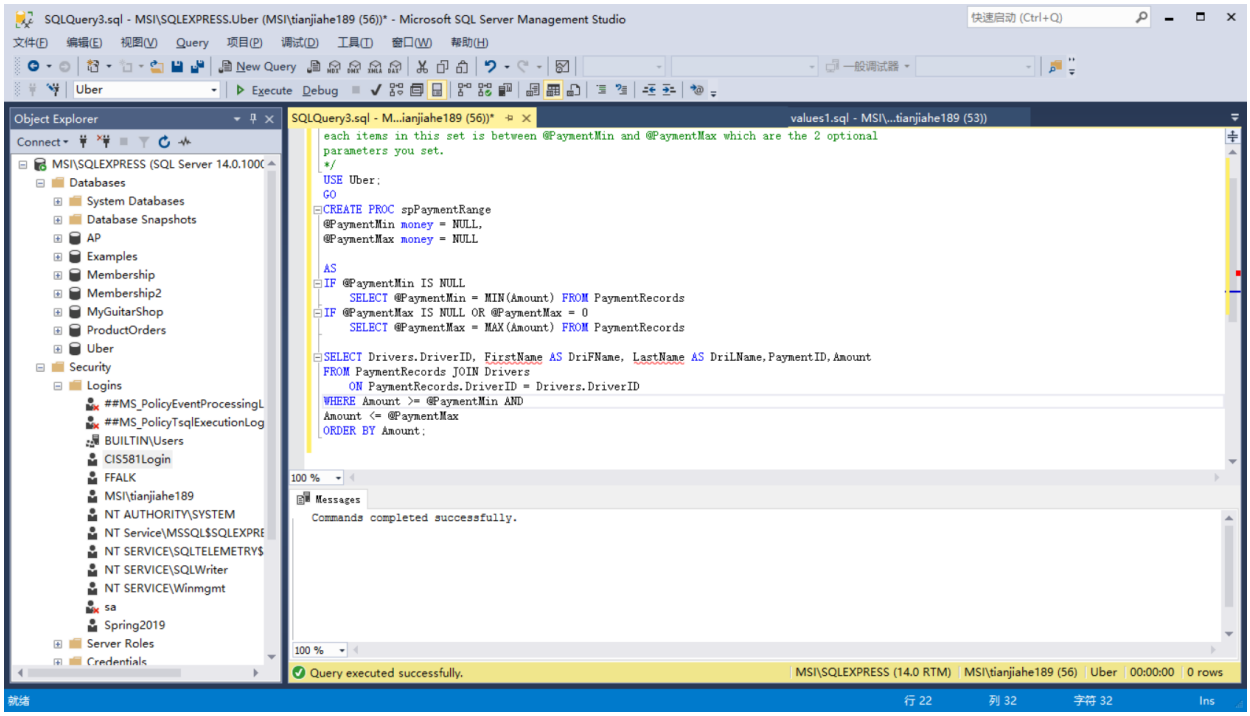
//

5.Stored procedures and functions

5.1 Stored procedure 1 and test

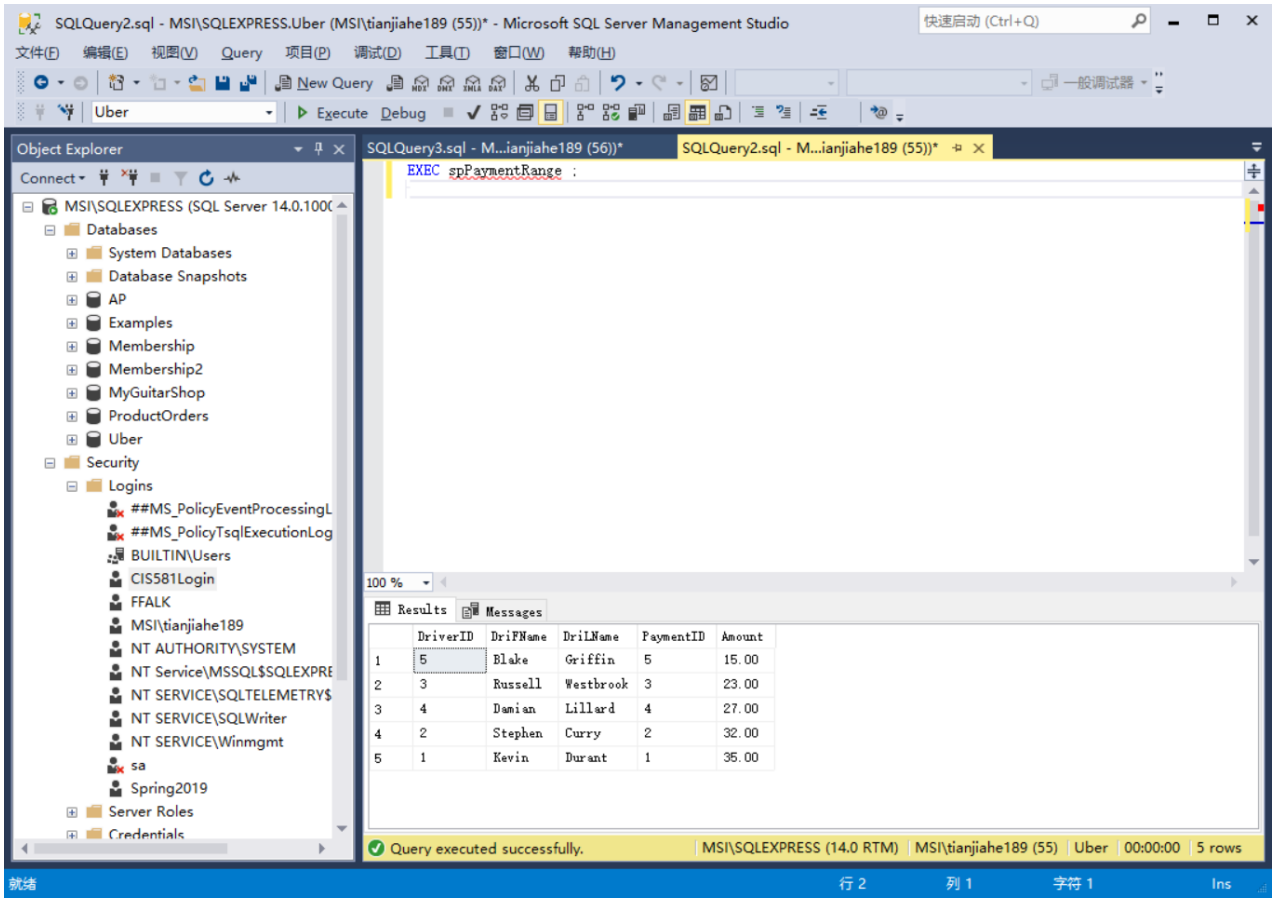
```
/*Created a user-defined stored procedure named spPaymentRange. It can accept
2 optional parameters. This procedure will return a result set consisting of
DriverID, FirstName AS DriFName, LastName AS DriLName,PaymentID,Amount. And the amount
of each items in this set is between @PaymentMin and @PaymentMax which are the 2
optional parameters you set.
*/
USE Uber;
GO
CREATE PROC spPaymentRange
@PaymentMin money = NULL,
@PaymentMax money = NULL

AS
IF @PaymentMin IS NULL
    SELECT @PaymentMin = MIN(Amount) FROM PaymentRecords
IF @PaymentMax IS NULL OR @PaymentMax = 0
```

```
    SELECT @PaymentMax = MAX(Amount) FROM PaymentRecords

SELECT Drivers.DriverID, FirstName AS DriFName, LastName AS DriLName,PaymentID,Amount
FROM PaymentRecords JOIN Drivers
    ON PaymentRecords.DriverID = Drivers.DriverID
WHERE Amount >= @PaymentMin AND
Amount <= @PaymentMax
ORDER BY Amount;
```



Testcase and screenshots:

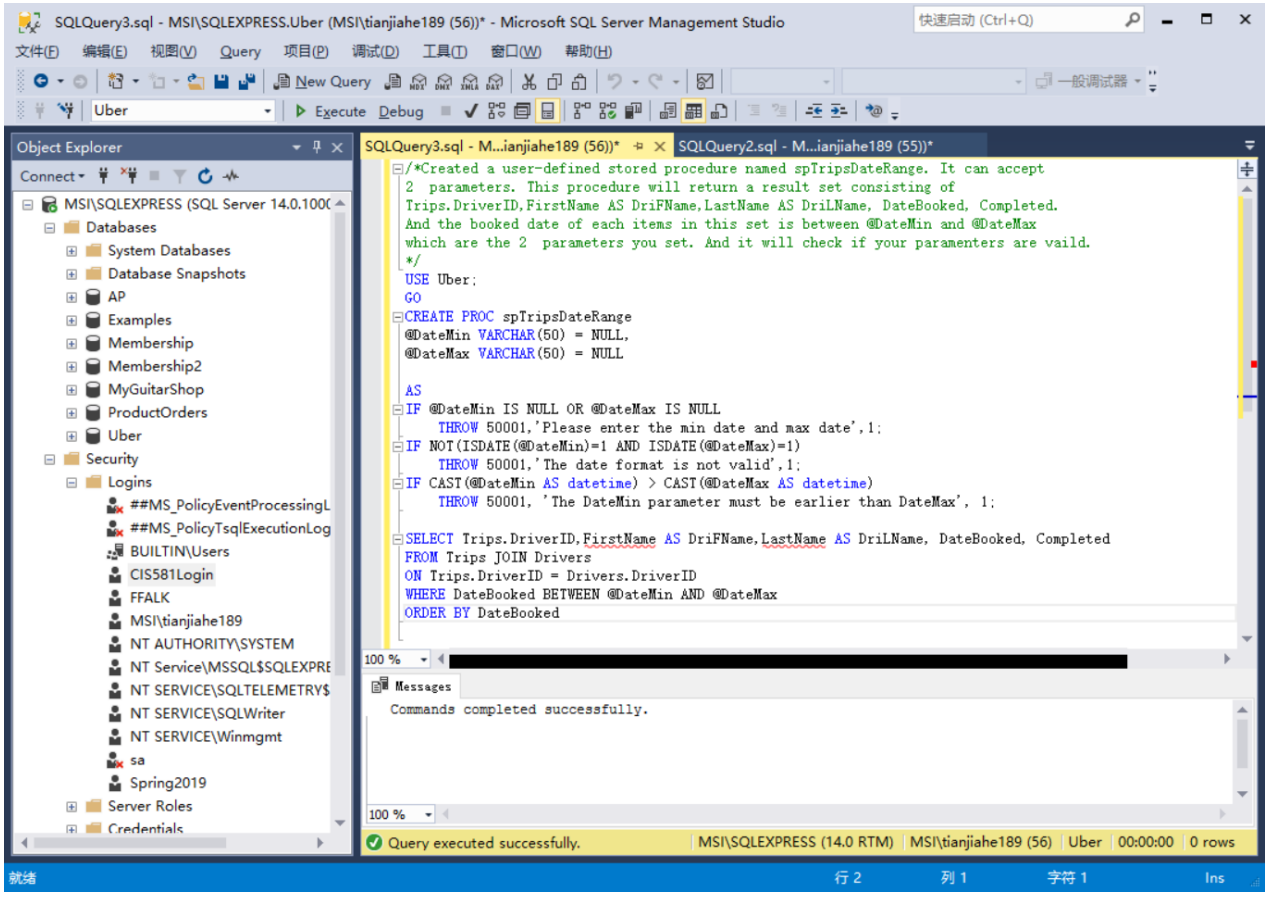
```
EXEC spPaymentRange ;
```



5.2 Stored procedure 2 and test

```
/*Created a user-defined stored procedure named spTripsDateRange. It can accept
2  parameters. This procedure will return a result set consisting of
Trips.DriverID,FirstName AS DriFName,LastName AS DriLName, DateBooked, Completed.
And the booked date of each items in this set is between @DateMin and @DateMax
which are the 2  parameters you set. And it will check if your paramenters are vaild.
*/
USE Uber;
GO
CREATE PROC spTripsDateRange
@DateMin VARCHAR(50) = NULL,
@DateMax VARCHAR(50) = NULL

AS
```

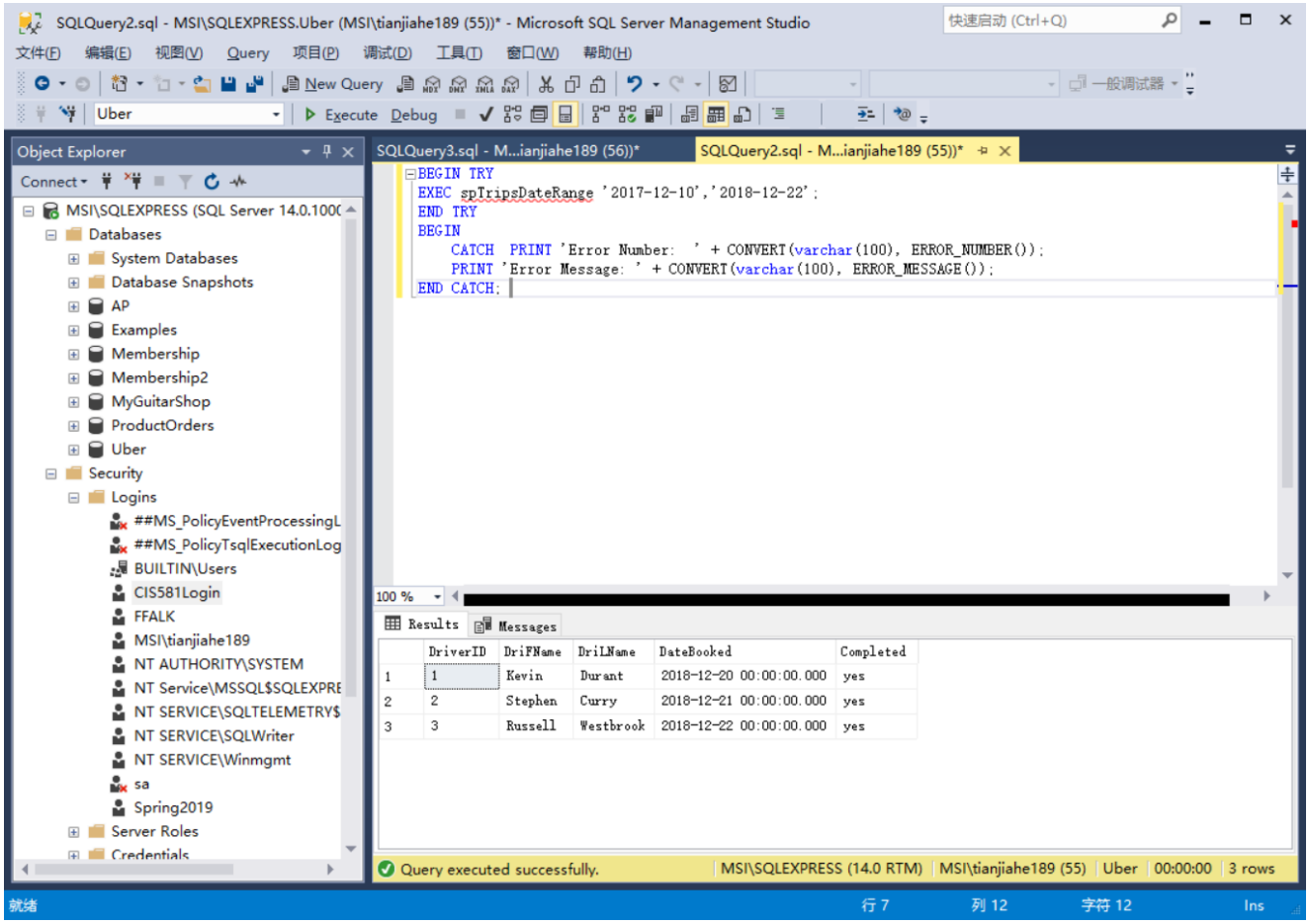
```
IF @DateMin IS NULL OR @DateMax IS NULL
    THROW 50001,'Please enter the min date and max date',1;
IF NOT(ISDATE(@DateMin)=1 AND ISDATE(@DateMax)=1)
    THROW 50001,'The date format is not valid',1;
IF CAST(@DateMin AS datetime) > CAST(@DateMax AS datetime)
    THROW 50001, 'The DateMin parameter must be earlier than DateMax', 1;

SELECT Trips.DriverID,FirstName AS DriFName,LastName AS DriLName, DateBooked, Completed
FROM Trips JOIN Drivers
ON Trips.DriverID = Drivers.DriverID
WHERE DateBooked BETWEEN @DateMin AND @DateMax
ORDER BY DateBooked
```



Testcase and screenshots:

```
BEGIN TRY
EXEC spTripsDateRange '2017-12-10','2018-12-22';
END TRY
```

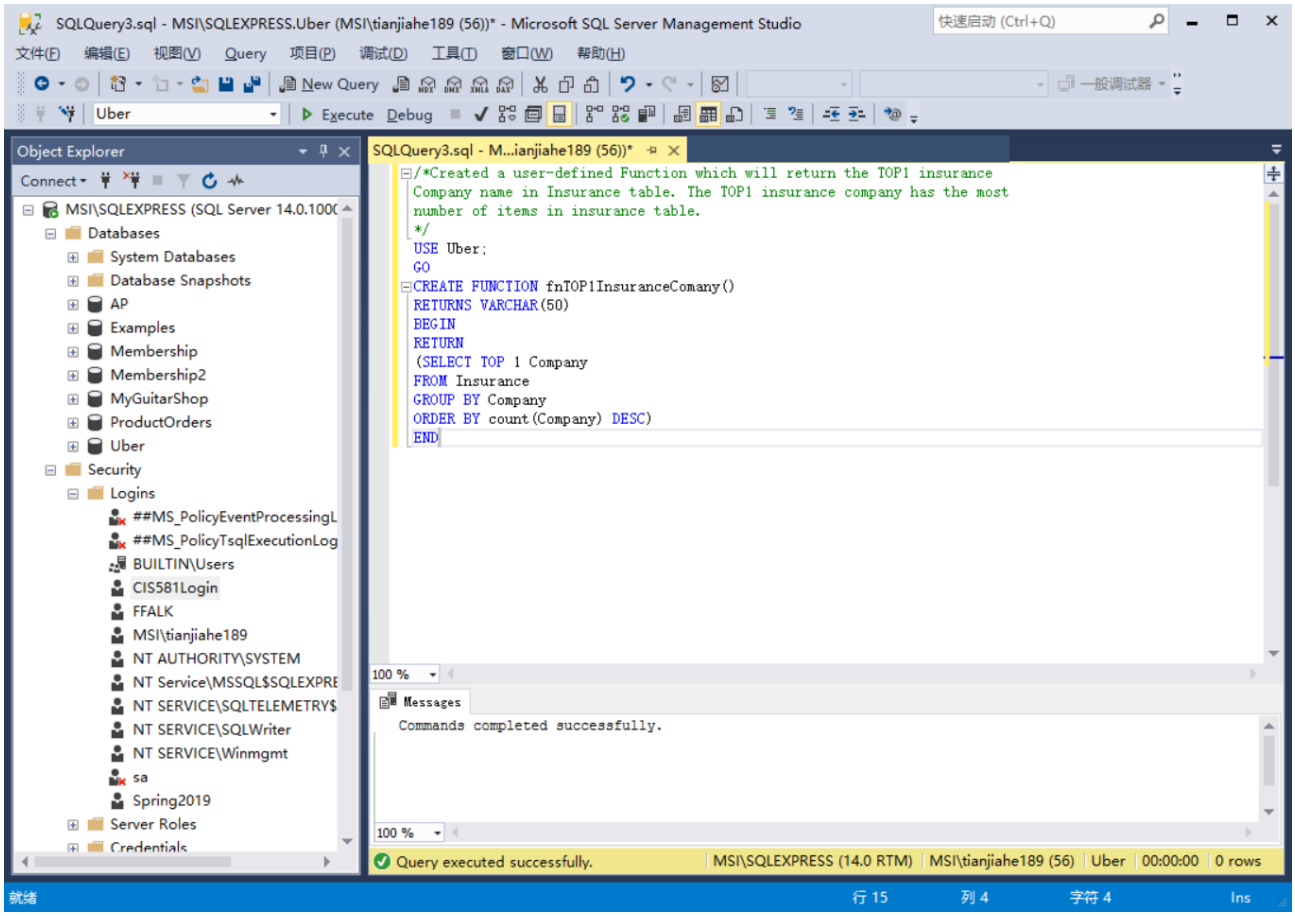
```
BEGIN
    CATCH  PRINT 'Error Number:  ' + CONVERT(varchar(100), ERROR_NUMBER());
    PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;
```



5.3 Function 1 and test

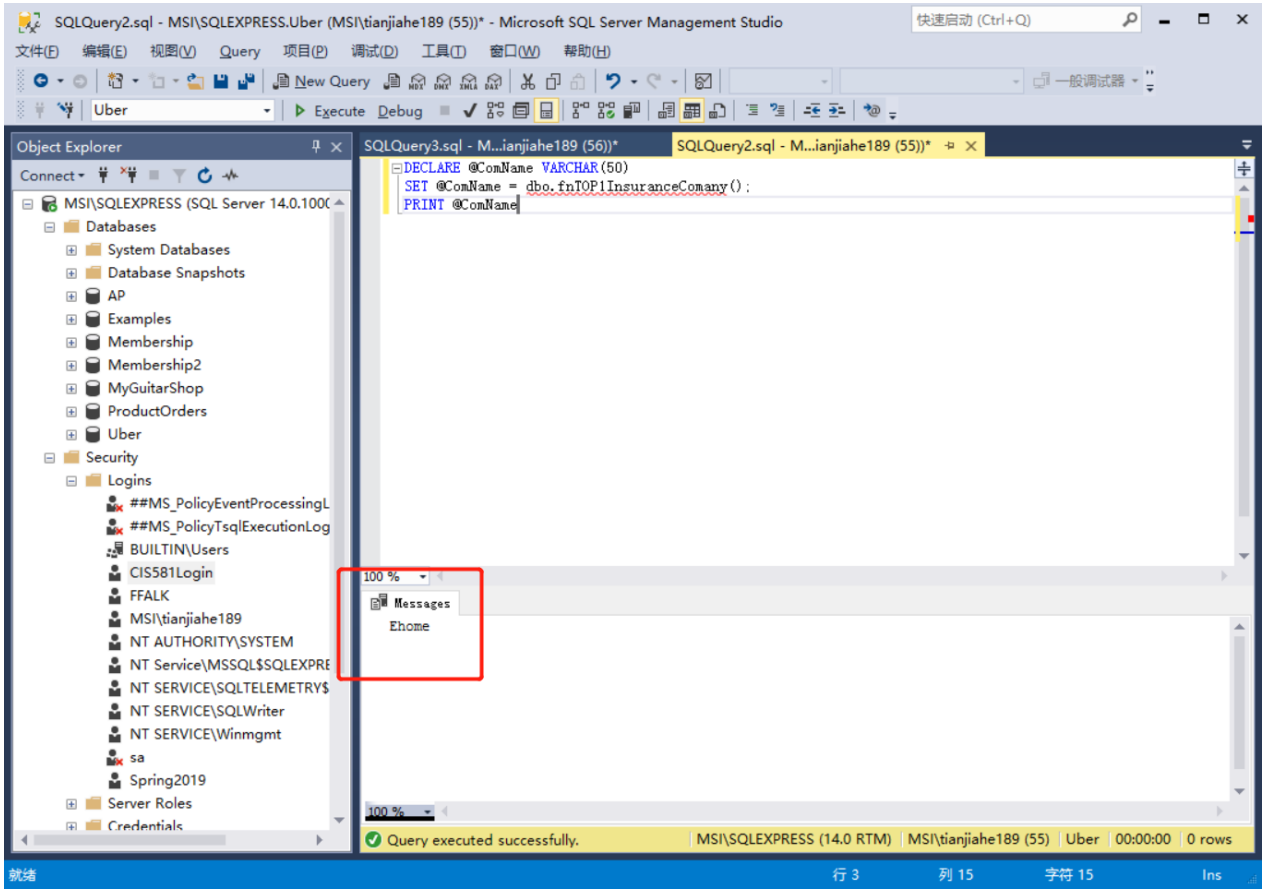
```
/*Created a user-defined Function which will return the TOP1 insurance
Company name in Insurance table. The TOP1 insurance company has the most
number of items in insurance table.
*/
USE Uber;
GO
CREATE FUNCTION fnTOP1InsuranceComany()
RETURNS VARCHAR(50)
BEGIN
RETURN
(SELECT TOP 1 Company
```

```
FROM Insurance
GROUP BY Company
ORDER BY count(Company) DESC)
END
```



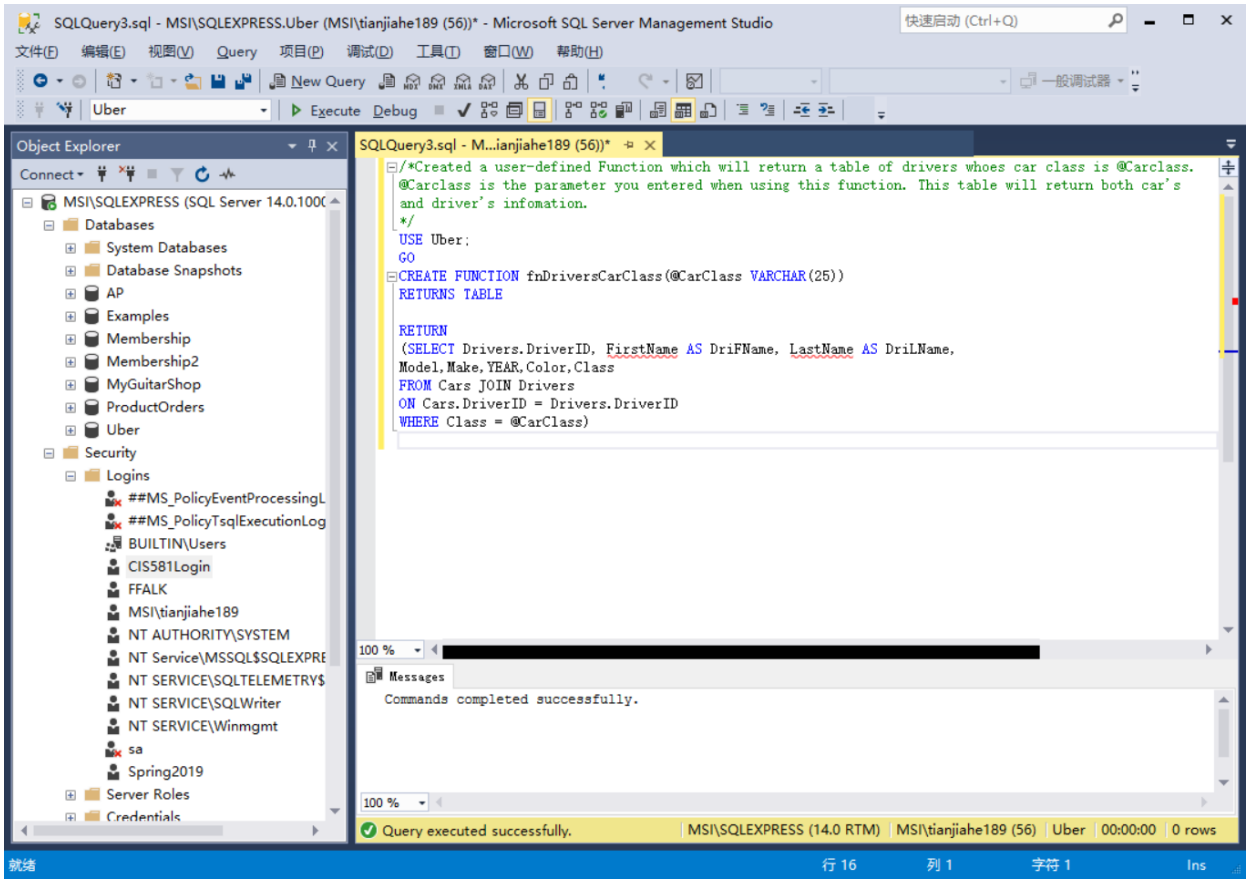
Testcase and screenshots:

```
DECLARE @ComName VARCHAR(50)
SET @ComName = dbo.fnTOP1InsuranceComany();
PRINT @ComName
```



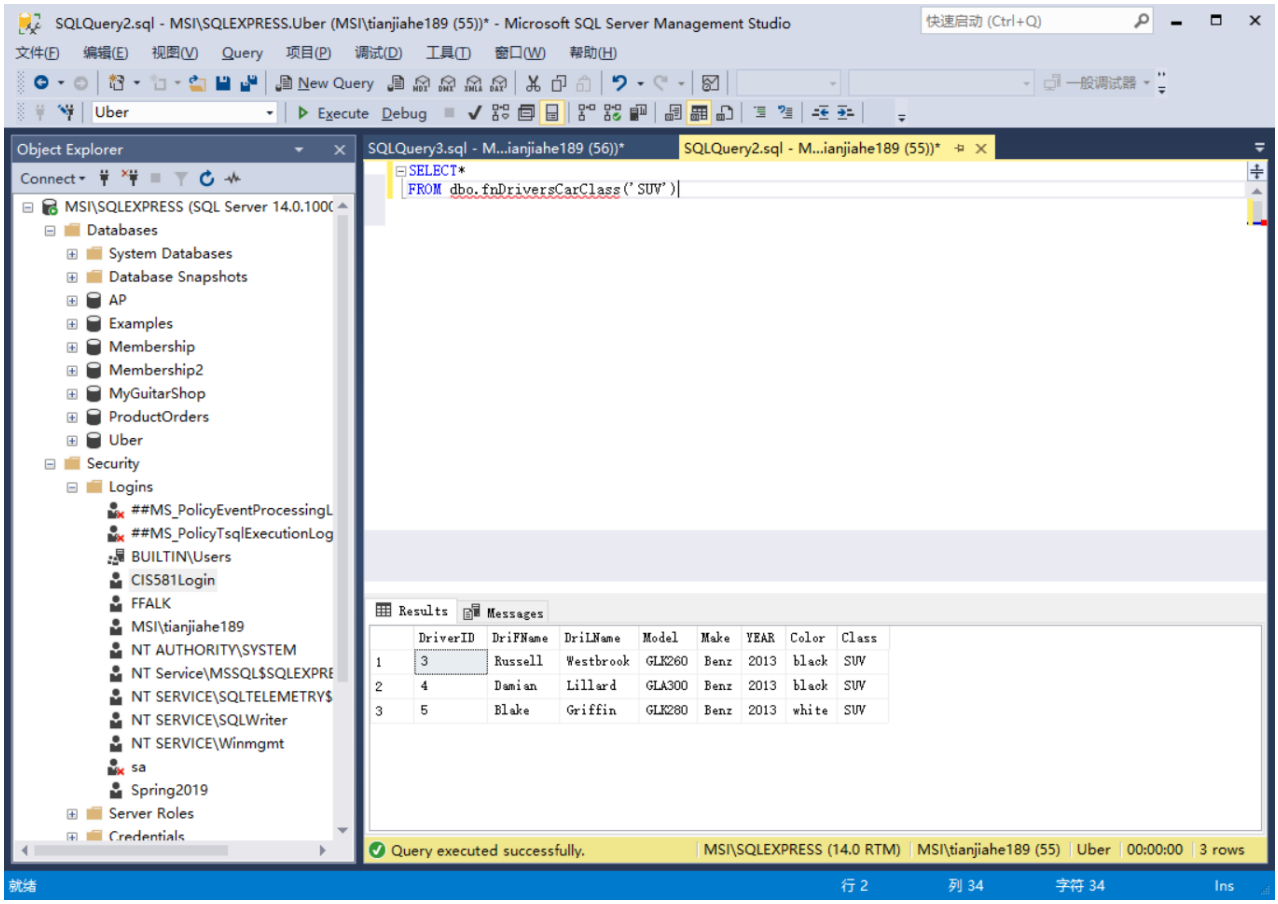
5.4 Function 2 and test

```
/*Created a user-defined Function which will return a table of drivers whoes car class
is @Carclass.
@Carclass is the parameter you entered when using this function. This table will return
both car's
and driver's infomation.
*/
USE Uber;
GO
CREATE FUNCTION fnDriversCarClass(@CarClass VARCHAR(25))
RETURNS TABLE

RETURN
(SELECT Drivers.DriverID, FirstName AS DriFName, LastName AS DriLName,
Model,Make,YEAR,Color,Class
FROM Cars JOIN Drivers
ON Cars.DriverID = Drivers.DriverID
WHERE Class = @CarClass)
```



Testcase and screenshots:

```
SELECT*
FROM dbo.fnDriversCarClass('SUV')
```



6. Conclusion

There are a lot of things to be aware of when designing and using a database. Views can help users make queries more easily, and using Functions and Stored procedures can improve database productivity. A reasonable security level setting protects the database. The overall design of the database guarantees its operational efficiency and data security issues.