

Title:

Designing a Database for an Uber Competitor Company

Name / ID:

Tong Cui / 305853364

Abstract:

My goal is to design a database for an Uber Competitor company. The database is used to manage all the data the company needs in its everyday work, including the detailed information about customers, drivers and trips/reservations.

Introduction:

I did the following jobs for the project:

- 1.Decide what tables to be included in the database and what columns to be included in each table.
- 2.Write the codes for implementing the database and inserting data into it.
- 3.Use views/procedures/functions to test its reliability.
- 4.Use login/role/permissions to guarantee its security.

Description:

I use Vetabelo to draw the E/R diagram and use MS SQL Server to do the rest of work.

Each aspect of SQL knowledge covered during this semester is included in the project.

Catalog:

Part I Database Design	3
Tables	3
Relations	4
Part II Database Implementation	5
Codes	5
Results	6
Part III Data Insertion	7
Codes	7
Results	8
Part IV Testing	12
Views	12
Triggers	15
Constraints	19
Stored Procedures	21
Functions	25
Part V Security	28
Roles	28
Permissions	29
Login, User	30
Role member	31
Part VI Conclusion/Project Analysis/Remark	33

Part I: Database Design

To avoid redundancy, I decide to use the following seven tables to store the data.

A. Tables:

The table '**Customers**' is used to store all the necessary information of the passengers.

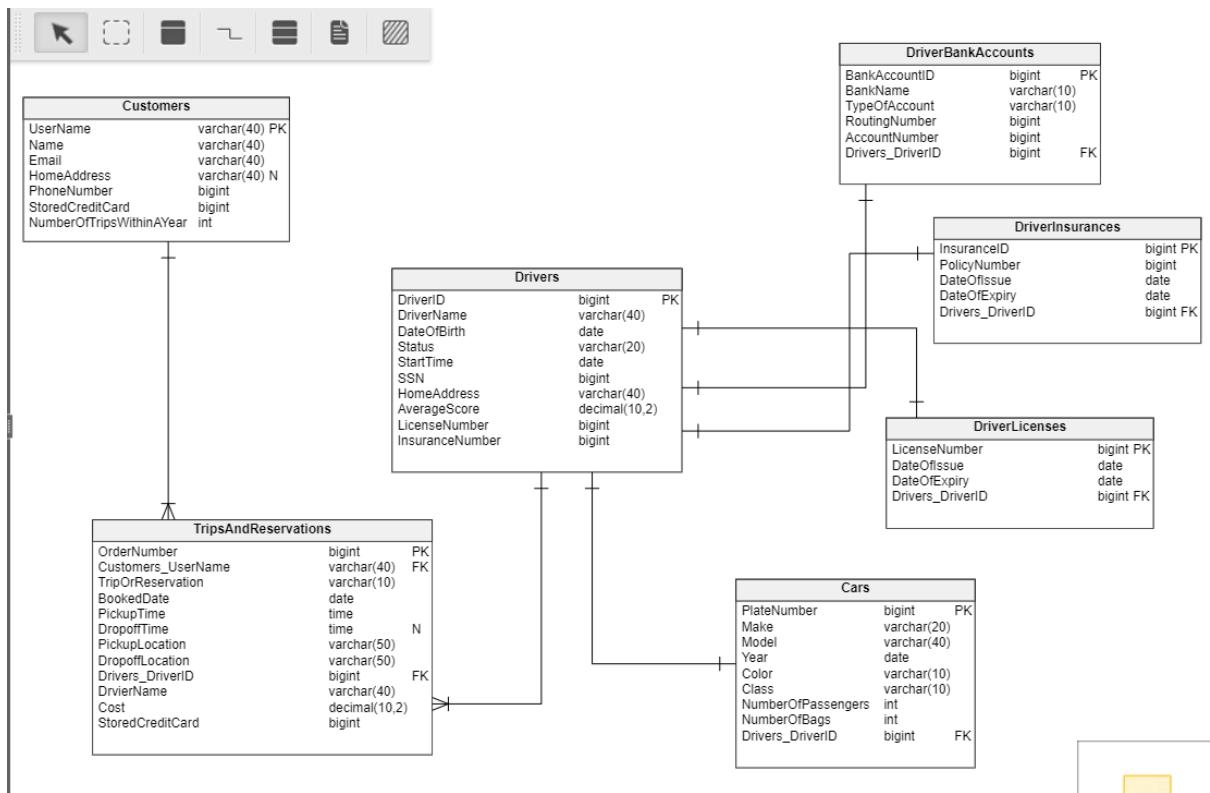
The column '**UserID**' is the primary key. Each customer has a unique UserID used for logging into the app. The password will not be contained in the table for safety's sake.

The home address can be null because it does not really matter when people call for a ride and some people may not want to reveal it to others.

The table '**Drivers**' contains information about the drivers. The column '**DriverID**' is the primary key. It is assigned to drivers by the company.

The table '**DriverBankAccount**', '**DriverInsurance**', '**DriverLicenses**' and '**Cars**' are all related to drivers. They provide drivers' additional information. Primary and foreign keys are displayed in the image.

The table '**TripsAndReservations**' is used to store the information about every trip/reservation of the company. The column '**OrderNumber**' is the primary key. If a customer has ordered a ride but has not met the driver, the value of the column '**TripOrReservation**' is 'Reservation'. If he arrives at his destination, then 'Trip'. The '**DropoffTime**' is nullable because we won't know the ending time of the trip until it is over.



(When the database is implemented, some slight modifications are made to make the database more rational. For instance:

The column '**UserName varchar (40)**' will be replaced by '**UserID bigint**'.

The column '**Name**' in '**Customers**' and '**DriverName**' in '**Drivers**' will be splatted up into '**FirstName**' and '**LastName**'.)

B. Relations:

We assume that each driver can only have one bank account, insurance license and car registered. Therefore, the relation between '**Drivers**' and '**DriverBankAccounts**', '**DriverInsurance**', '**DriverLicenses**' and '**Cars**' are 1 to 1. The relation between '**Customers**' and '**Drivers**' is many to many, so I use '**TripsAndReservations**' as a linking table. The relation between '**Customers**' and '**TripsAndReservations**' is one to many. So as it between '**Drivers**' and '**TripsAndReservations**'.

Part II: Database Implementation

A. Code

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the creation script for the 'UberCompetitor' database and its initial tables:

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

CREATE DATABASE UberCompetitor
GO

USE UberCompetitor
GO

CREATE TABLE Customers(
    UserID bigint not null identity(1,1),
    FirstName varchar(40) not null,
    LastName varchar(40) not null,
    HomeAddress varchar(40) null,
    PhoneNumber varchar(40) not null,
    StoredCreditCard bigint not null,
    NumberOfTripWithinYear int default 0,
    CONSTRAINT PK_UserID PRIMARY KEY CLUSTERED (
        UserID ASC
    )
)
GO

CREATE TABLE Drivers(
    DriverID bigint not null identity(1,1),
    FirstName varchar(40) not null,
    LastName varchar(40) not null,
    PhoneNumber varchar(40) not null,
    DateOfBirth smalldatetime not null check (DateOfBirth > '1930-01-01'),
    DriverStatus varchar(20) not null,
    StartTime smalldatetime not null,
    SSN bigint not null,
    HomeAddress varchar(40) not null,
    AverageScore decimal(10,2) not null check (AverageScore >= 0) ,check (AverageScore <= 5),
    LicenseNumber bigint not null,
    InsuranceNumber bigint not null
)
```

The status bar at the bottom indicates the connection is established (Connected. (1/1)), the script has saved changes (Item(s) Saved), and the current date and time are 2019/4/15.

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the continuation of the creation script for the 'UberCompetitor' database:

```
CONSTRAINT PK_DriverID PRIMARY KEY CLUSTERED (
    DriverID ASC
)
)
GO

CREATE TABLE TripsAndReservations(
    OrderNumber bigint not null identity(1,1),
    TripIDReservation varchar(30) not null,
    UserID bigint not null,
    DriverID bigint not null,
    BookedDate smalldatetime not null,
    PickUpTime smalldatetime not null check (PickUpTime > BookedDate),
    DropOffTime smalldatetime ,
    PickUpLocation varchar(50) not null,
    DropOffLocation varchar(50) not null,
    Cost decimal(10,2) not null
    CONSTRAINT PK_OrderNumber PRIMARY KEY CLUSTERED (
        OrderNumber ASC
    )
)
GO

CREATE TABLE DriverBankAccounts(
    BankAccountID bigint identity(1,1),
    DriverID bigint not null,
    BankName varchar(10) not null,
    TypeOfAccount varchar(10) not null,
    RoutingNumber bigint not null,
    AccountNumber bigint not null
    CONSTRAINT PK_BankAccountID PRIMARY KEY CLUSTERED (
        BankAccountID ASC
    )
)
GO

CREATE TABLE DriverInsurances(
```

The status bar at the bottom indicates the connection is established (Connected. (1/1)), the script has saved changes (Item(s) Saved), and the current date and time is 11:53 on 2019/4/15.

```

USE createdatabase.s_...MNDU\yuanji (57)
GO
CREATE TABLE Insurance (
    InsuranceNumber BIGINT NOT NULL IDENTITY(1,1),
    DriverID BIGINT NOT NULL,
    DateOfIssue SMALLDATETIME NOT NULL,
    DateOfExpiry SMALLDATETIME NOT NULL, CHECK (DateOfExpiry > DateOfIssue),
    CONSTRAINT PK_InsuranceID PRIMARY KEY CLUSTERED (
        InsuranceNumber ASC
    )
)
GO

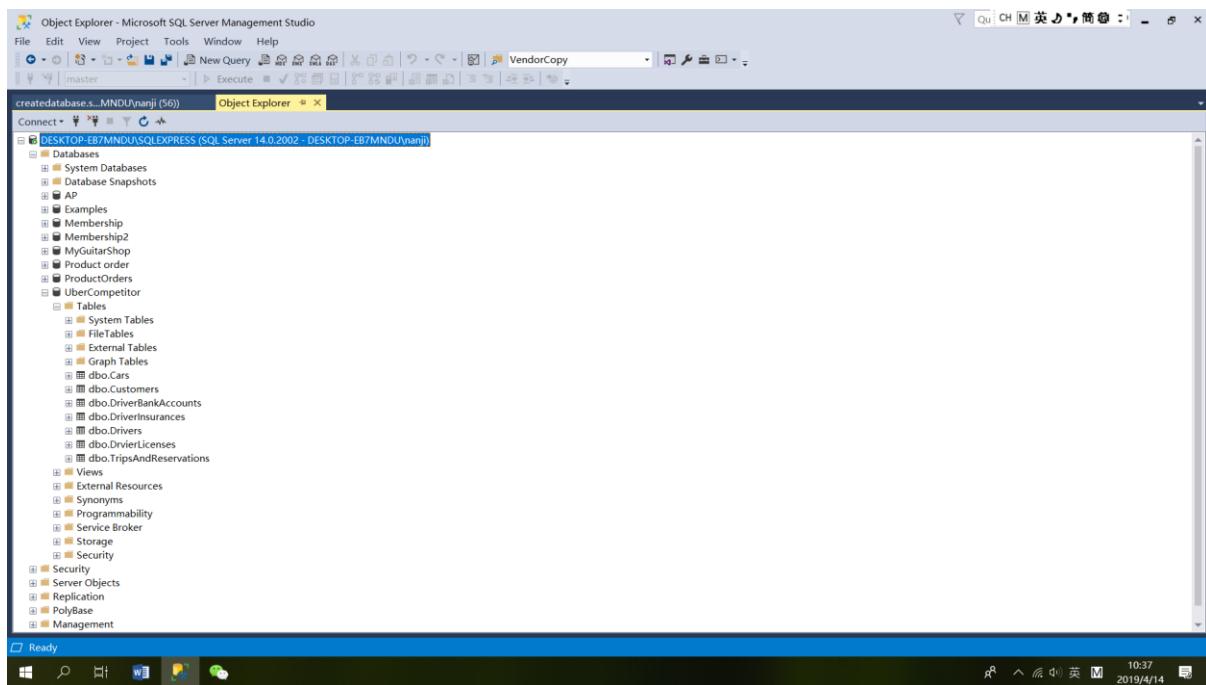
CREATE TABLE DriverLicenses (
    LicenseNumber BIGINT NOT NULL IDENTITY(1,1),
    DriverID BIGINT NOT NULL,
    DateOfIssue SMALLDATETIME NOT NULL,
    DateOfExpiry SMALLDATETIME NOT NULL CHECK (DateOfExpiry > DateOfIssue),
    CONSTRAINT PK_LicenseNumber PRIMARY KEY CLUSTERED (
        LicenseNumber ASC
    )
)
GO

CREATE TABLE Cars (
    PlateNumber BIGINT NOT NULL IDENTITY(1,1),
    DriverID BIGINT NOT NULL,
    Made VARCHAR(20) NOT NULL,
    Model VARCHAR(30) NOT NULL,
    Color SMALLDATETIME NOT NULL,
    Class VARCHAR(10) NOT NULL,
    NumberOfPassengers INT NOT NULL,
    NumberOfBags INT NOT NULL,
    CONSTRAINT PK_PlateNumber PRIMARY KEY CLUSTERED (
        PlateNumber ASC
    )
)
GO

```

100 % 1/1
Connected. (1/1)
DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\yuanji ... UberCompetitor 00:00:00 0 rows
Item(s) Saved Ln 62 Col 37 Ch 37 INS CH M 11:56 2019/4/15

B. Results



Part III: Data Insertion

A. Code

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'datainsertion.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (54)) - Microsoft SQL Server Management Studio' is open. The code in the window is as follows:

```
USE UberCompetitor

SET IDENTITY_INSERT Customers ON
INSERT Customers (UserID, FirstName, LastName, HomeAddress, PhoneNumber, StoredCreditCard, NumberOfTripsWithinAYear)
VALUES
    ('100001', 'Leon', 'Hull', 'PO Box 7005', '(800) 555-1205', '1111111111,24'),
    ('100002', 'Bent', 'Russell', 'PO Box 66631', '(301) 555-8950', '202323232323,13'),
    ('100003', 'Steve', 'Jackson', '4669 N Fresno', '(800) 555-8725', '3333333333,7'),
    ('100004', 'Zach', 'Love', '1528 N Sierra Vista', '(559) 555-9999', '4444444444,10'),
    ('100005', 'Luka', 'Grant', '4583 E Home', '(559) 555-1551', '5555555555,12'),
    SET IDENTITY_INSERT Customers OFF
go

SET IDENTITY_INSERT Drivers ON
INSERT Drivers (DriverID, FirstName, LastName, PhoneNumber, DateOfBirth, DriverStatus, StartTime, SSN, HomeAddress, AverageScore, LicenseNumber, InsuranceNumber)
VALUES
    ('200001', 'Frank', 'Oden', '(559) 555-7473', '1988-02-22 00:00:00', 'Active', '2016-05-22 00:00:00', '123456789', '4420 N. First Street, Suite 108', '4.5,538769,14223),
    ('200002', 'Tim', 'Pan', '(314) 555-8834', '1959-02-14 00:00:00', 'Active', '2012-12-22 00:00:00', '223456789', '4486 N. First Street, Suite 108', '4.2,538769,14224),
    ('200003', 'Landry', 'Parker', '(314) 575-8834', '1991-02-27 00:00:00', 'Active', '2017-06-13 00:00:00', '323456789', '3255 Ramos Cir', '4.8,738769,14225),
    ('200004', 'Chris', 'Howard', '(314) 575-8831', '1969-09-27 00:00:00', 'Active', '2017-02-13 00:00:00', '423456789', '3219 Ramos Cir', '4.2,838769,14226),
    ('200005', 'Jason', 'Wagner', '(315) 888-8834', '1957-07-14 00:00:00', 'Active', '2009-06-16 00:00:00', '523456789', '3441 W MacArthur Blvd', '4.6,938769,14227),
    SET IDENTITY_INSERT Drivers OFF
go

SET IDENTITY_INSERT TripsAndReservations ON
INSERT TripsAndReservations (OrderNumber, TripOrReservation, UserID, DriverID, BookedDate, PickUpTime, DropOffTime, PickUpLocation, DropOffLocation, Cost)
VALUES
    ('300001', 'Trip', '100004', '2016-05-22 00:00:00', '2016-05-23 00:00:00', '2016-05-23 00:01:00', 'PO Box 2069', 'PO Box 1192', '15.3),
    ('300002', 'Reservation', '100001', '200001', '2016-05-25 00:00:00', '2016-05-27 00:00:00', 'null', 'PO Box 2044', 'PO Box 1172', '16),
    ('300003', 'Trip', '100003', '200003', '2017-05-22 00:00:00', '2017-05-23 00:00:00', '415 F Olive Ave', '117 W Michelobrena Top Floor', '24.3),
    ('300004', 'Trip', '100002', '200002', '2018-05-22 00:00:00', '2018-05-23 00:00:00', 'Secretary Of State', '1626 E Street', '14),
    ('300005', 'Trip', '100005', '200005', '2018-01-22 00:00:00', '2018-01-23 00:00:00', 'Secretary Of State', '1615 E Street', '19),
    SET IDENTITY_INSERT TripsAndReservations OFF
go

100 % ▾
Messages
(0 rows affected)
100 % ▾
Query executed successfully.
```

The status bar at the bottom shows 'Ready', '15:40', '2019/4/14', and '0 rows'.

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'datainsertion.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (54)) - Microsoft SQL Server Management Studio' is open. The code in the window is as follows:

```
go

SET IDENTITY_INSERT DriverBankAccounts ON
INSERT DriverBankAccounts (BankAccountID, BankName, TypeOfAccount, RoutingNumber, AccountNumber)
VALUES
    ('400006', 'AAA', 'Savings', '19486122,109484623),
    ('400007', 'GGG', 'Savings', '19486121,109487623),
    ('400008', 'TTT', 'Checking', '19486115,109484699),
    ('400009', 'AAA', 'Savings', '59486122,104424623),
    ('400010', 'TTT', 'Savings', '19486135,109400823),
    SET IDENTITY_INSERT DriverBankAccounts OFF
go

SET IDENTITY_INSERT DriverInsurances ON
INSERT DriverInsurances (InsuranceNumber, DriverID, DateOfIssue, DateOfExpiry)
VALUES
    ('14223', '200001', '2002-05-23 00:00:00', '2028-05-23 00:00:00'),
    ('14224', '200002', '2008-05-28 00:00:00', '2028-05-28 00:00:00'),
    ('14225', '200003', '2015-01-23 00:00:00', '2035-01-23 00:00:00'),
    ('14226', '200004', '2002-05-23 00:00:00', '2022-05-23 00:00:00'),
    ('14227', '200005', '2009-09-15 00:00:00', '2029-09-15 00:00:00');
    SET IDENTITY_INSERT DriverInsurances OFF
go

SET IDENTITY_INSERT DriverLicenses ON
INSERT DriverLicenses (LicenseNumber, DriverID, DateOfIssue, DateOfExpiry)
VALUES
    ('538769', '200001', '2002-05-23 00:00:00', '2022-05-23 00:00:00'),
    ('638769', '200002', '2008-05-28 00:00:00', '2028-05-28 00:00:00'),
    ('738769', '200003', '2015-01-23 00:00:00', '2035-01-23 00:00:00'),
    ('838769', '200004', '2003-05-23 00:00:00', '2023-05-23 00:00:00'),
    ('938769', '200005', '2009-09-15 00:00:00', '2029-09-15 00:00:00');
    SET IDENTITY_INSERT DriverLicenses OFF
go

100 % ▾
Messages
(0 rows affected)
100 % ▾
Query executed successfully.
```

The status bar at the bottom shows 'Ready', '15:41', '2019/4/14', and '0 rows'.

```

datainsertion.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (54)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
... New Query VendorCopy ...
UberCompetitor Execute Object Explorer
datainsertion.sql...7MNDU\nanji (54) createdatabase.s...MNDU\nanji (53)
go
SET IDENTITY_INSERT Cars ON
Insert Cars (PlateNumber,DriverID,Made,Model,Color,Class,NumberOfPassengers,NumberOfBags)
values
(500001, 200001, 'Honda', 'A', '2018', 'Yellow', 'B', 4, 5),
(500002, 200002, 'Toyota', 'C', '2015', 'Brown', 'B', 6, 8),
(500003, 200003, 'Chevrolet', 'D', '2017', 'White', 'C', 4, 6),
(500004, 200004, 'Ford', 'A', '2015', 'Grey', 'A', 7, 5),
(500005, 200005, 'Honda', 'C', '2010', 'Black', 'B', 4, 7);
SET IDENTITY_INSERT Cars OFF
go

```

Messages

(0 rows affected)

Query executed successfully.

Ready

Ln 64 Col 62 Ch 62 INS

15:41 2019/4/14

B. Results

Customers:

```

SQLQuery4.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (54)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
... New Query VendorCopy ...
UberCompetitor Execute Object Explorer
SQLQuery4.sql...7MNDU\nanji (54)* datainsertion.sql...7MNDU\nanji (55)*
select * from Customers;

```

Results

UserID	FirstName	LastName	HomeAddress	PhoneNumber	StoredCreditCard	NumberOfTripsWithinAYear
1	Leon	Hill	PO Box 7005	(800) 555-1205	1111111111	24
2	Ben	Russell	PO Box 96521	(30) 555-1990	2333333322	13
3	Steve	Jackson	4669 N Fresno	(800) 555-8725	3333333333	7
4	Zach	Lowe	1528 N Sierra Vista	(559) 555-9999	4444444444	10
5	Luka	Grant	4063 E Home	(559) 555-1351	5555555555	12

Messages

Query executed successfully.

Ready

Ln 1 Col 25 Ch 25 INS

15:25 2019/4/14

Drivers:

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

```
select * from Drivers;
```

The results pane displays a table with the following data:

DriverID	FirstName	LastName	PhoneNumber	DateOfBirth	DriverStatus	StartTime	SSN	HomeAddress	AverageScore	LicenseNumber	InsuranceNumber
1	Frank	Olsen	(509) 555-7473	1988-05-22 00:00:00	Active	2010-05-22 00:00:00	523456789	12345 N First Street, Suite 108	4.50	538769	14226
2	John	Parker	(314) 555-8834	1999-01-14 00:00:00	Active	2013-12-22 00:00:00	232456789	4488 N First Street, Suite 108	4.00	632769	14224
3	200002	Landy	(314) 579-8834	1991-05-27 00:00:00	Active	2017-08-13 00:00:00	323456789	3255 N Ramos Cir	4.80	728769	14225
4	200003	Landy	(314) 579-8834	1991-05-27 00:00:00	Active	2017-08-13 00:00:00	423456789	3219 Ramos Cir	4.20	838769	14226
5	200004	Chris	(314) 579-8831	1969-09-27 00:00:00	Active	2017-02-13 00:00:00	523456789	3441 W MacArthur Blvd	4.60	938769	14227
5	200005	Jason	(314) 888-8834	1957-07-14 00:00:00	Active	2008-06-16 00:00:00	523456789	3441 W MacArthur Blvd	4.60	938769	14227

The status bar at the bottom indicates "Query executed successfully." and "15:44 2019/4/14".

TripsAndReservations:

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

```
select * from TripsAndReservations;
```

The results pane displays a table with the following data:

OrderNumber	TripOrReservation	UserID	DriverID	BookedDate	PickUpTime	DropOffTime	PickUpLocation	DropOffLocation	Cost
1	Trip	100004	200004	2016-05-22 00:00:00	2016-05-23 00:00:00	2016-05-23 00:01:00	PO Box 2069	PO Box 1192	15.30
2	Reservation	100001	200001	2016-05-25 00:00:00	2016-05-27 00:00:00	NULL	PO Box 2044	PO Box 1172	18.00
3	Trip	100003	200003	2017-05-22 00:00:00	2017-05-23 00:00:00	2017-05-23 00:01:00	415 E Olive Ave	117 W Michillinda Top Floor	24.30
4	Trip	100002	200002	2018-05-22 00:00:00	2018-05-23 00:00:00	2018-05-23 00:01:00	Secretary Of State	1628 E Street	14.00
5	Trip	100005	200005	2018-01-22 00:00:00	2018-01-23 00:00:00	2018-01-23 00:01:00	Secretary Of State	1615 E Street	19.00

The status bar at the bottom indicates "Query executed successfully." and "15:45 2019/4/14".

DriverInsurances:

SQLQuery1.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (53)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
New Query Object Explorer VendorCopy
SQLQuery1.sql - 7MNDU\nanji (53)* Execute
SELECT * FROM DriverInsurances
;
```

Results Messages

InsuranceNumber	DriverID	DateIssue	DateOfExpiry
1	14224	200001	2020-05-21 00:00:00
2	14224	200002	2008-05-21 00:00:00
3	14225	200003	2015-01-23 00:00:00
4	14226	200004	2002-05-23 00:00:00
5	14227	200005	2009-09-15 00:00:00

Query executed successfully.

Ready

Ln 1 Col 31 Ch 31 INS 15:45 2019/4/14

DriverLicenses:

SQLQuery1.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (53)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
New Query Object Explorer VendorCopy
SQLQuery1.sql - 7MNDU\nanji (53)* Execute
SELECT * FROM DriverLicenses
;
```

Results Messages

LicenseNumber	DriverID	DateIssue	DateOfExpiry
1	538769	200001	2002-05-23 00:00:00
2	638769	200002	2008-05-28 00:00:00
3	738769	200003	2015-01-23 00:00:00
4	838769	200004	2002-05-23 00:00:00
5	938769	200005	2009-09-15 00:00:00

Query executed successfully.

Ready

Ln 1 Col 29 Ch 29 INS 15:50 2019/4/14

DriverBankAccounts:

SQLQuery1.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\yanji (53)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
New Query New Results Object Explorer
SQLQuery1.sql - 7MNDU\yanji (53)* Execute VendorCopy
SELECT * FROM DriverBankAccounts
;
```

100 %

Results Messages

BankAccountID	BankName	TypeOfAccount	RoutingNumber	AccountNumber
1	400001	Saving	19458102	109400823
2	400007	Saving	19458121	109400823
3	400008	Checking	19458115	109400823
4	400009	AAA	5948122	109400823
5	400010	Saving	19458135	109400823

Query executed successfully.

Ready

Ln 1 Col 33 Ch 33 INS 15:51 2019/4/14

Cars:

SQLQuery1.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\yanji (53)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
New Query New Results Object Explorer
SQLQuery1.sql - 7MNDU\yanji (53)* Execute VendorCopy
SELECT * FROM Cars
;
```

100 %

Results Messages

PlateNumber	DriverID	Made	Model	CarYear	Color	Class	NumberOfPassengers	NumberOfBags	
1	500005	200001	Honda	A	2010-01-01 00:00:00	Yellow	B	4	5
2	500006	200002	Toyota	C	2015-01-01 00:00:00	Brown	B	6	8
3	500007	200003	Chevy	D	2017-01-01 00:00:00	White	C	4	6
4	500008	200004	Dodge	A	2015-01-01 00:00:00	Grey	A	7	5
5	500009	200005	Honda	C	2010-01-01 00:00:00	Black	B	4	7

Query executed successfully.

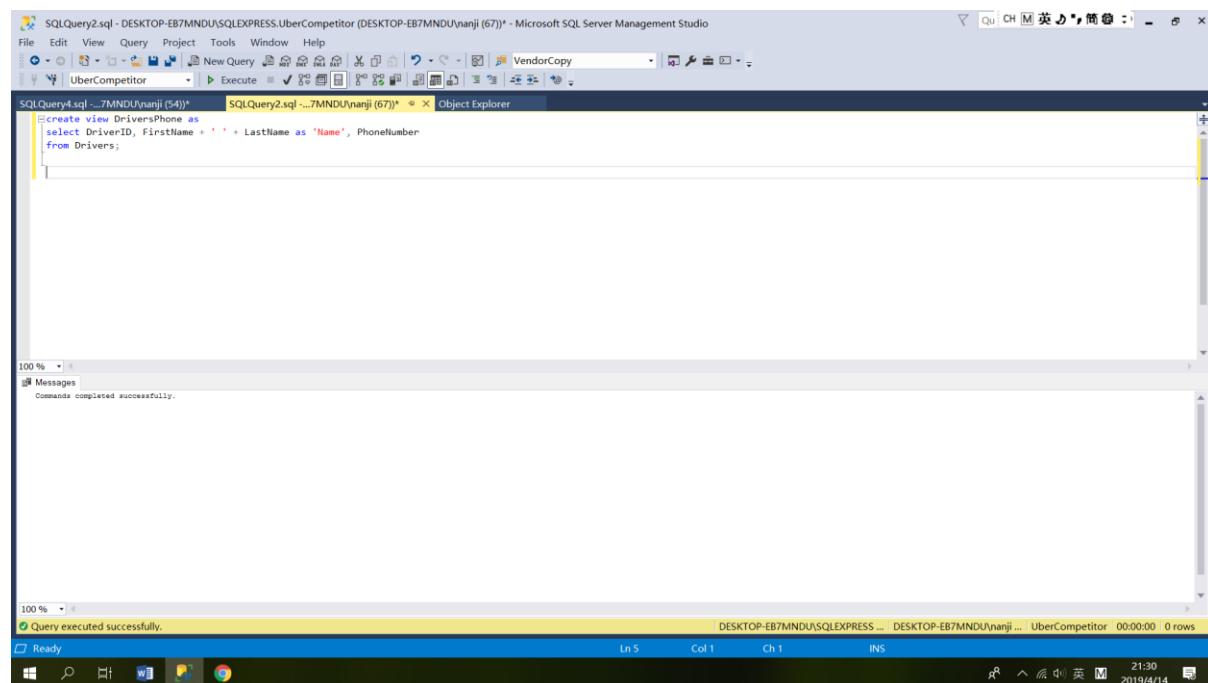
Ready

Ln 1 Col 19 Ch 19 INS 15:51 2019/4/14

Part IV: Testing

A. Views

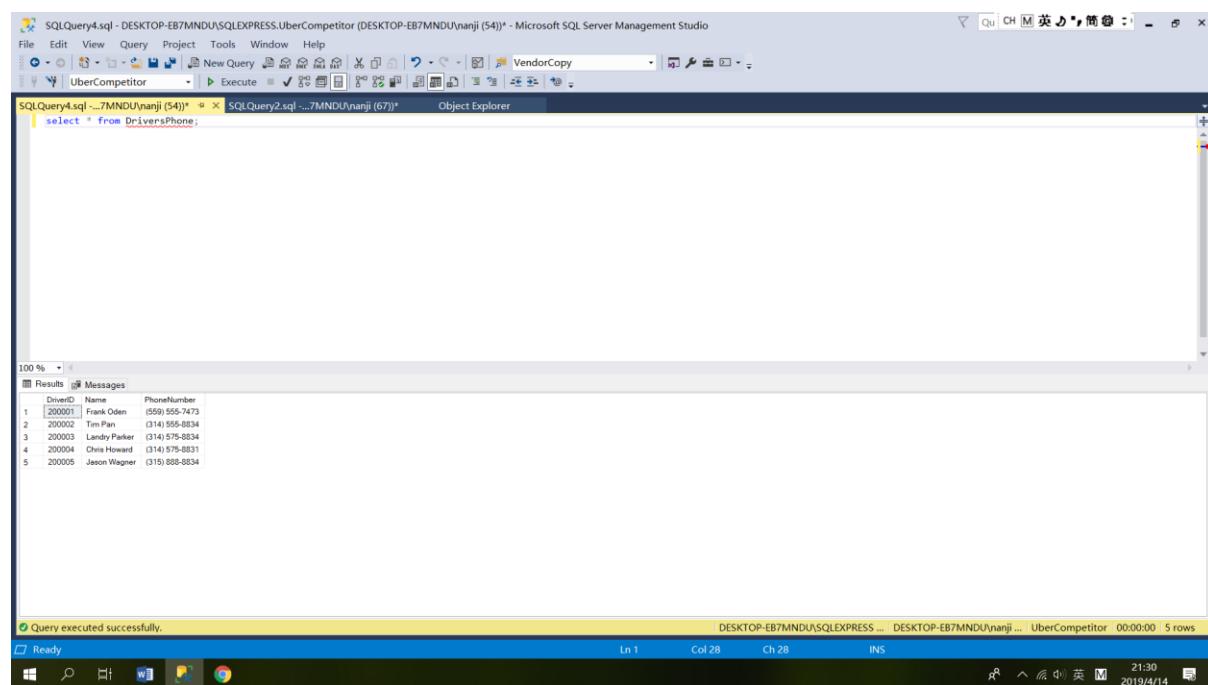
(1) View drivers' phone numbers.



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery4.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (67))' is open. It contains the following SQL code:

```
CREATE VIEW DriversPhone AS
SELECT DriverID, FirstName + ' ' + LastName AS 'Name', PhoneNumber
FROM Drivers;
```

The status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery4.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (54))' is open. It contains the following SQL code:

```
SELECT * FROM DriversPhone;
```

The results pane displays a table with the following data:

DriverID	Name	PhoneNumber
1	Frank Oden	(559) 555-7473
2	Tim Park	(314) 575-8834
3	Chris Parker	(314) 575-8834
4	Chris Howard	(314) 575-8831
5	Jason Wagner	(319) 690-8834

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

(2) view all the drivers whose score is larger than the average. Return their bank accounts also.

```

view2.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (63)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
... New Query VendorCopy
... UberCompetitor Execute ✓
SQLQuery2.sql -7MNDU\nanji (55)* view2.sql - DESK...-7MNDU\nanji (63)* × datainsertion.sql -7MNDU\nanji (62) createdatabase.s...MNDU\nanji (53) Object Explorer
CREATE VIEW HighestScore AS
SELECT TOP 5 D.DriverID, FirstName + ' ' + LastName AS DriverName, AverageScore, BankAccountID, BankName, TypeOfAccount, RoutingNumber, AccountNumber
FROM Drivers AS D JOIN DriverBankAccounts AS DBA ON D.DriverID = DBA.DriverID
WHERE AverageScore >
(SELECT AVG(AverageScore)
FROM Drivers)
ORDER BY AverageScore DESC;

```

Messages
Command completed successfully.

Query executed successfully.

Ready

```

SQLQuery2.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (55)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
... New Query VendorCopy
... UberCompetitor Execute ✓
SQLQuery2.sql -7MNDU\nanji (55)* × view2.sql - DESK...-7MNDU\nanji (63)* × datainsertion.sql -7MNDU\nanji (62) createdatabase.s...MNDU\nanji (53) Object Explorer
SELECT * FROM HighestScore;

```

Results

DriverID	DriverName	AverageScore	BankAccountID	BankName	TypeOfAccount	RoutingNumber	AccountNumber
1	Landry Parker	4.80	400008	TTT	cheking	19496115	109484699
2	Jason Wagner	4.80	400010	TTT	Saving	19496135	109400623
3	Frank Oden	4.50	400006	AAA	Saving	19496122	109484623

Messages
Query executed successfully.

Ready

(3) view the customer who has taken the most trips within a year.

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery3.sql' is open, displaying the following T-SQL code:

```
CREATE VIEW VIPCustomer AS
SELECT UserID, FirstName + ' ' + LastName AS CustomerName, NumberOfTripWithinAYear, StoredCreditCard
FROM Customers
WHERE NumberOfTripWithinAYear = (
    SELECT MAX(NumberOfTripWithinAYear)
    FROM Customers)
```

The status bar at the bottom indicates 'Commands completed successfully.'

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery2.sql' is open, displaying the following T-SQL code:

```
SELECT * FROM VIPCustomer
```

The results pane shows a single row of data:

	UserID	CustomerName	NumberOfTripWithinAYear	StoredCreditCard
1	100001	Leon Hill	24	1111111111

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

(4) view the drivers older than 40 years. Return their license information also.

```

SQLQuery2.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query New Object Explorer VendorCopy
SQLQuery2.sql -7MNDU\nanji (57)* SQLQuery1.sql -7MNDU\nanji (52)* Object Explorer
CREATE VIEW OlderThan48Years AS
SELECT D.DriverID, FirstName + ' ' + LastName AS DriverName, PhoneNumber, DateOfBirth, DriverStatus, StartTime, SSN, AverageScore, LicenseNumber, DateOfIssue, DateOfExpiry
FROM Drivers AS D JOIN DriverLicenses AS DL
ON D.DriverID = DL.DriverID
WHERE DateOfBirth < '1979-04-15 00:00:00'

```

Messages

Commands completed successfully.

Query executed successfully.

Ready

Ln 4 Col 28 Ch 28 INS

9:35 2019/4/15

```

SQLQuery2.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (57)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query New Object Explorer VendorCopy
SQLQuery2.sql -7MNDU\nanji (57)* SQLQuery1.sql -7MNDU\nanji (52)* Object Explorer
SELECT * FROM OlderThan48Years;

```

Results

DriverID	DriverName	PhoneNumber	DateOfBirth	DriverStatus	StartTime	SSN	AverageScore	LicenseNumber	DateOfIssue	DateOfExpiry
1	200002 Tim Pan	(314) 555-8834	1959-02-14 00:00:00	Active	2012-12-22 00:00:00	223456789	4.20	838769	2008-05-28 00:00:00	2028-05-28 00:00:00
2	200004 Chris Howard	(314) 575-8831	1969-09-27 00:00:00	Active	2017-02-13 00:00:00	423456789	4.20	838769	2002-05-23 00:00:00	2022-05-23 00:00:00
3	200005 Jason Wagner	(315) 588-8834	1957-07-14 00:00:00	Active	2009-06-16 00:00:00	523456789	4.60	938769	2009-09-15 00:00:00	2029-09-15 00:00:00

Query executed successfully.

Ready

Ln 1 Col 1 INS

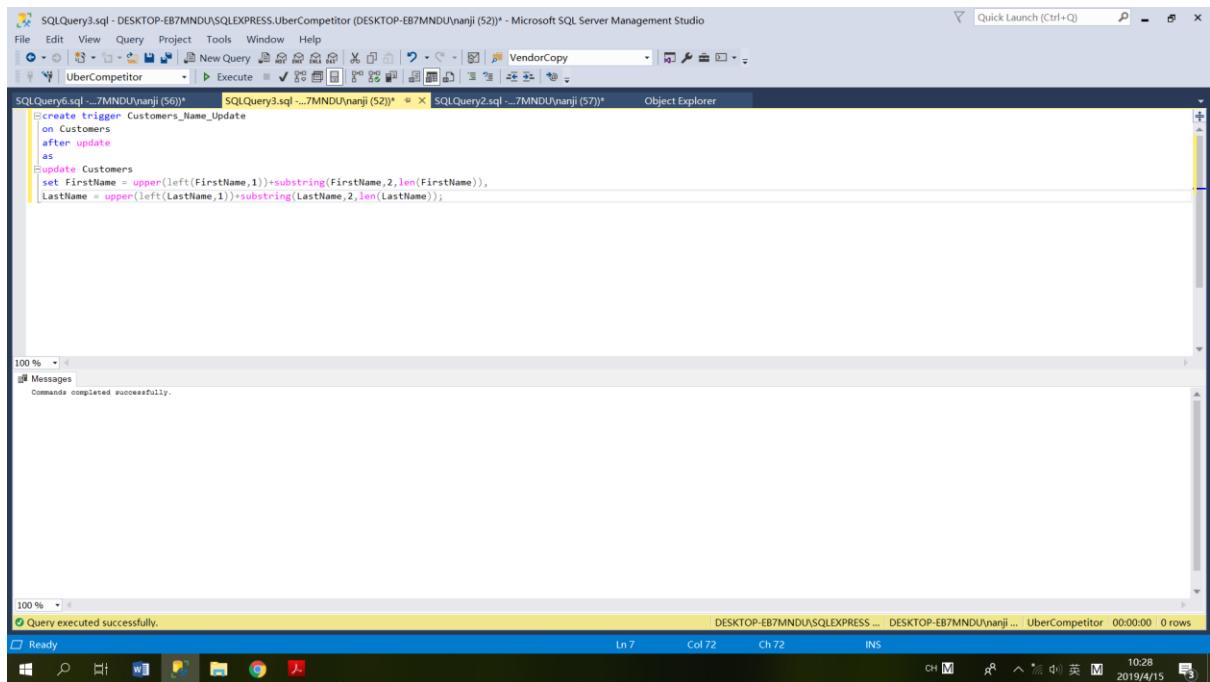
9:36 2019/4/15

B. Triggers

(1) Someone wants to change the customer's name from 'Bobby Murphy' to 'Mike

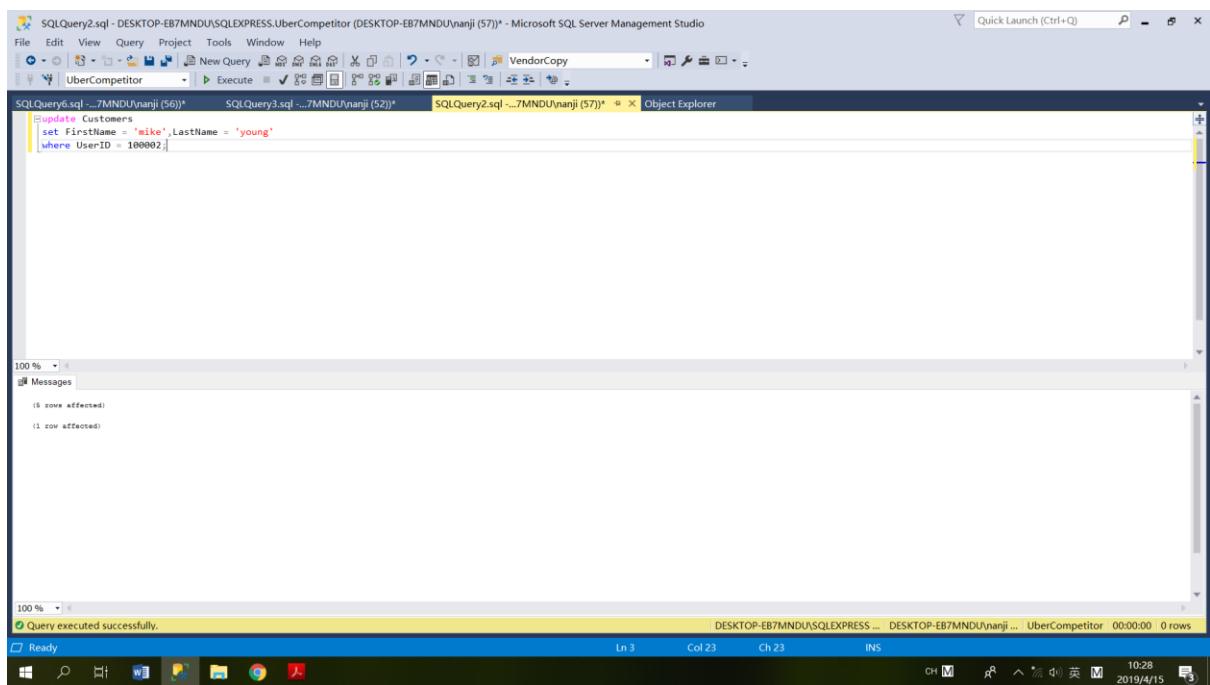
Young' whose UserID is 100002. But by mistake the characters that he typed in

are all in lower case. Use a trigger to automatically turn the first character of the first name and last name into upper case.



Screenshot of Microsoft SQL Server Management Studio showing the creation of a trigger named 'Customers_Name_Update' on the 'Customers' table. The trigger uses an 'after update' event and updates the 'Customers' table by setting the 'FirstName' and 'LastName' columns to uppercase. The 'Messages' pane shows the command completed successfully.

```
CREATE TRIGGER Customers_Name_Update
ON Customers
AFTER UPDATE
AS
UPDATE Customers
SET FirstName = UPPER(left(FirstName,1))+SUBSTRING(FirstName,2,len(FirstName)),
    LastName = UPPER(left(LastName,1))+SUBSTRING(LastName,2,len(LastName));
```



Screenshot of Microsoft SQL Server Management Studio showing an update query executed successfully. The query updates the 'Customers' table for the user with ID 100002, setting both 'FirstName' and 'LastName' to 'mike'. The 'Messages' pane shows 1 row affected.

```
UPDATE Customers
SET FirstName = 'mike', LastName = 'young'
WHERE UserID = 100002;
```

Before:

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery6.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\yanji (56)) - Microsoft SQL Server Management Studio". The query window contains the following SQL code:

```
select * from Customers;
```

The results grid displays the following data:

UserID	FirstName	LastName	HomeAddress	PhoneNumber	StoredCreditCard	NumberOfTripWithinAYear
1	Leon	Hill	PO Box 7005	(800) 555-1205	1111111111	24
2	Mike	Young	PO Box 66621	(301) 555-8990	2222222222	13
3	Steve	Jackson	4669 N Fresno	(800) 555-8725	3333333333	7
4	Zach	Lowe	1528 N Sierra Vista	(599) 555-9999	4444444444	10
5	Luka	Grant	4583 E Home	(599) 555-1551	5555555555	12

A message bar at the bottom left says "Query executed successfully." The status bar at the bottom right shows "DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\yanji ... UberCompetitor 00:00:00 | 5 rows".

After:

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery6.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\yanji (56)) - Microsoft SQL Server Management Studio". The query window contains the same SQL code as before:

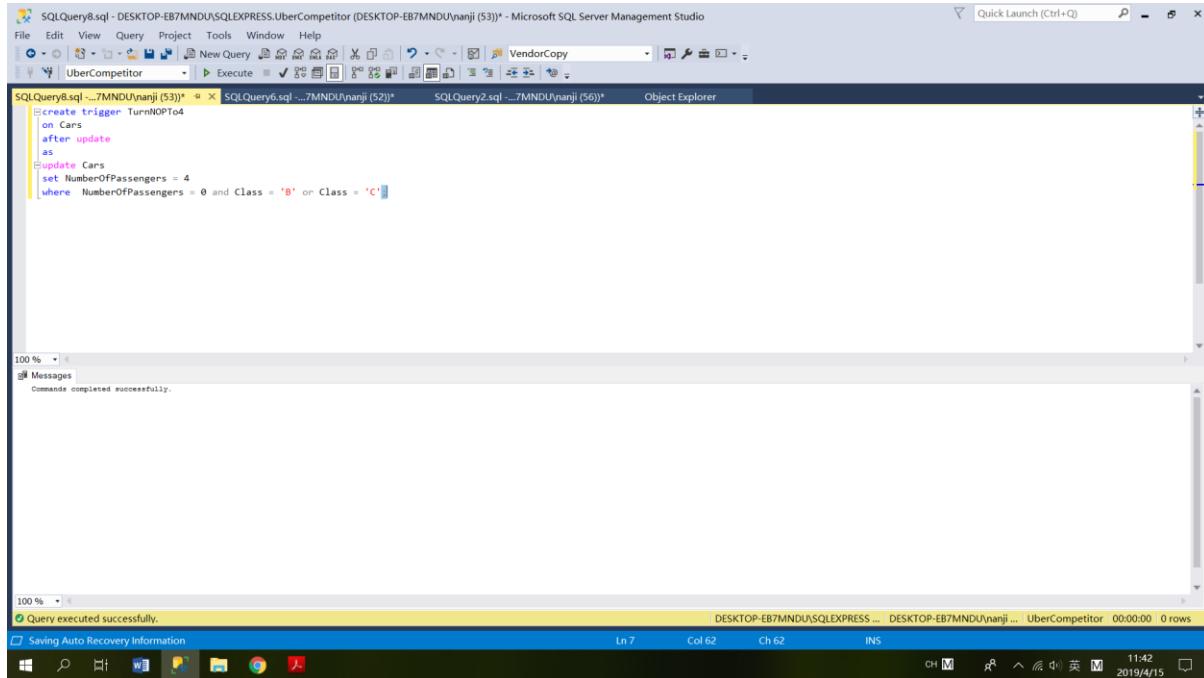
```
select * from Customers;
```

The results grid displays the same data as the previous screenshot, indicating no changes have been made to the table.

A message bar at the bottom left says "Query executed successfully." The status bar at the bottom right shows "DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\yanji ... UberCompetitor 00:00:00 | 5 rows".

(2) When the number of passengers of a car whose class is 'B' or 'C' is updated to 0

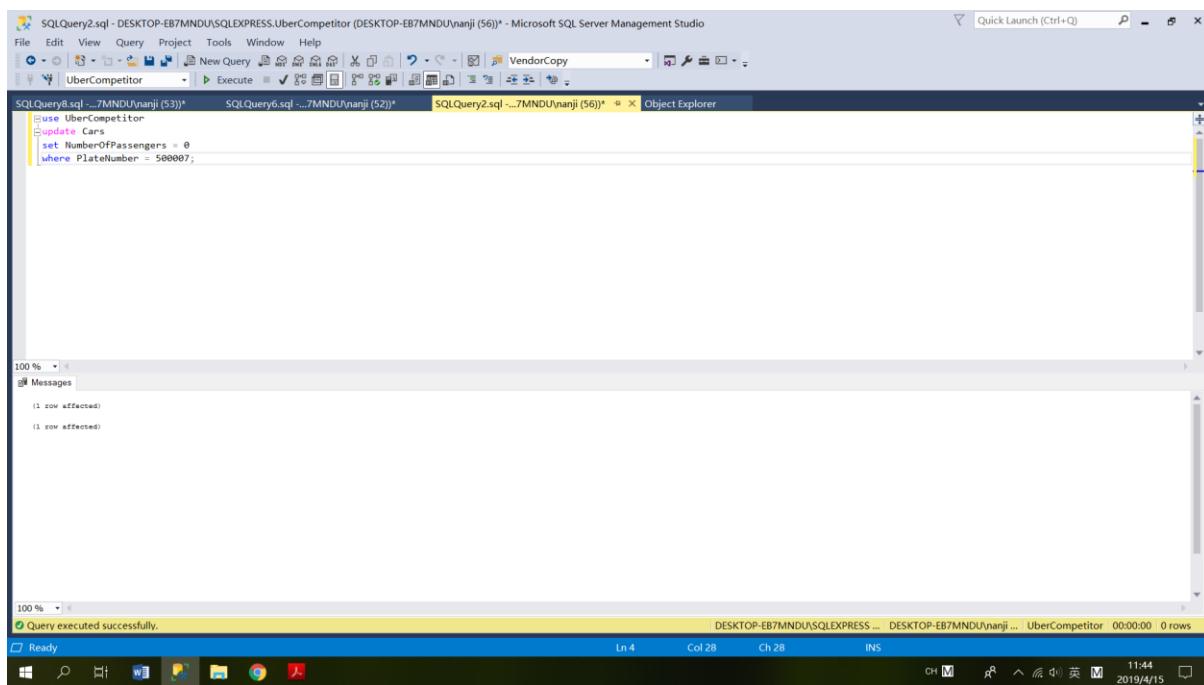
by mistake, use a trigger to automatically change it back to 4.



The screenshot shows the Microsoft SQL Server Management Studio interface. A new query window titled 'SQLQuery8.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (53))' is open. The code in the query window is:

```
CREATE TRIGGER TurnNPTo4
ON Cars
AFTER UPDATE
AS
UPDATE Cars
SET NumberOfPassengers = 4
WHERE NumberOfPassengers = 0 AND Class = 'B' OR Class = 'C';
```

The status bar at the bottom right indicates the session is 'Saving Auto Recovery Information' and the current time is '11:42 2019/4/15'.



The screenshot shows the Microsoft SQL Server Management Studio interface. A new query window titled 'SQLQuery8.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (56))' is open. The code in the query window is:

```
USE UberCompetitor
UPDATE Cars
SET NumberOfPassengers = 0
WHERE PlateNumber = 500007;
```

The status bar at the bottom right indicates the session is 'Ready' and the current time is '11:44 2019/4/15'.

Before:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'UberCompetitor' is selected. In the center pane, a query window displays the following SQL code:

```
use UberCompetitor
select * from Cars;
```

The results pane shows a table with the following data:

PlateNumber	DriverID	Made	Model	CarYear	Color	Class	NumberOfPassengers	NumberOfBags
1	200001	Honda	A	2010-01-01 00:00:00	Yellow	B	4	5
2	500006	200002	Toyota	2015-01-01 00:00:00	Brown	B	6	8
3	500007	200003	Chevy	2017-01-01 00:00:00	White	C	4	6
4	500008	200004	Dodge	2015-01-01 00:00:00	Grey	A	7	5
5	500009	200005	Honda	C	2010-01-01 00:00:00	Black	B	4

Below the results, a message bar indicates: "Query executed successfully." The status bar at the bottom right shows the date and time: "2019/4/15 11:37".

After:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'UberCompetitor' is selected. In the center pane, a query window displays the following SQL code:

```
use UberCompetitor
select * from Cars;
```

The results pane shows a table with the same data as the previous screenshot:

PlateNumber	DriverID	Made	Model	CarYear	Color	Class	NumberOfPassengers	NumberOfBags
1	200001	Honda	A	2010-01-01 00:00:00	Yellow	B	4	5
2	500006	200002	Toyota	2015-01-01 00:00:00	Brown	B	6	8
3	500007	200003	Chevy	2017-01-01 00:00:00	White	C	4	6
4	500008	200004	Dodge	2015-01-01 00:00:00	Grey	A	7	5
5	500009	200005	Honda	C	2010-01-01 00:00:00	Black	B	4

Below the results, a message bar indicates: "Query executed successfully." The status bar at the bottom right shows the date and time: "2019/4/15 11:42".

C. Constraints

Constraints are implemented when the database is implemented.

I use several checks and primary key constraints.

The screenshot shows the Microsoft SQL Server Management Studio interface. A new database named 'UberCompetitor' has been created. The 'Customers' table is being defined with columns: UserID (bigint, identity(1,1), not null), FirstName (varchar(40), not null), LastName (varchar(40), not null), HomeAddress (varchar(40), null), PhoneNumber (varchar(40), not null), StoredCreditCard (bigint, not null), and NumberOfTripsWithinAYear (int, default 0). A primary key constraint 'CONSTRAINT PK_UserID PRIMARY KEY CLUSTERED' is defined on the UserID column. The 'Drivers' table is also being defined with similar columns, including a check constraint 'DateOfBirth > '1930-01-01''. The 'Customer' table is being defined with columns: DriverID (bigint, identity(1,1), not null), FirstName (varchar(40), not null), LastName (varchar(40), not null), PhoneNumber (varchar(40), not null), DateOfBirth (smalldatetime, not null, check (DateOfBirth > '1930-01-01')), DriverStatus (varchar(20), not null), StartTime (smalldatetime, not null), SSN (bigint, not null), HomeAddress (varchar(40), not null), AverageScore (decimal(10,2), not null, check (AverageScore >= 0) ,check (AverageScore <= 5)), LicenseNumber (bigint, not null), and InsuranceNumber (bigint, not null). A primary key constraint 'CONSTRAINT PK_DriverID PRIMARY KEY CLUSTERED' is defined on the DriverID column. The 'TripsAndReservations' table is being defined with columns: OrderNumber (bigint, identity(1,1), not null), TripOrReservation (varchar(30), not null), UserID (bigint, not null), DriverID (bigint, not null), BookedDate (smalldatetime, not null, check (BookedDate > GetDate)), PickUpTime (smalldatetime, not null, check (PickUpTime > BookedDate)), DropOffTime (smalldatetime, not null), PickUpLocation (varchar(50), not null), DropOffLocation (varchar(50), not null), and Cost (decimal(10,2), not null). A primary key constraint 'CONSTRAINT PK_OrderNumber PRIMARY KEY CLUSTERED' is defined on the OrderNumber column. The 'DriverBankAccounts' table is being defined with columns: BankAccountID (bigint, identity(1,1), not null), DriverID (bigint, not null), BankName (varchar(10), not null), TypeOfAccount (varchar(10), not null), RoutingNumber (bigint, not null), AccountNumber (bigint, not null). A primary key constraint 'CONSTRAINT PK_BankAccountID PRIMARY KEY CLUSTERED' is defined on the BankAccountID column. The 'DriverInsurances' table is being defined with columns: (empty table definition).

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

CREATE DATABASE UberCompetitor
GO

USE UberCompetitor
GO

CREATE TABLE Customers(
    UserID bigint NOT NULL IDENTITY(1,1),
    FirstName varchar(40) NOT NULL,
    LastName varchar(40) NOT NULL,
    HomeAddress varchar(40) NULL,
    PhoneNumber varchar(40) NOT NULL,
    StoredCreditCard bigint NOT NULL,
    NumberOfTripsWithinAYear int DEFAULT 0,
    CONSTRAINT PK_UserID PRIMARY KEY CLUSTERED (
        UserID ASC
    )
)
GO

CREATE TABLE Drivers(
    DriverID bigint NOT NULL IDENTITY(1,1),
    FirstName varchar(40) NOT NULL,
    LastName varchar(40) NOT NULL,
    PhoneNumber varchar(40) NOT NULL,
    DateOfBirth smalldatetime NOT NULL CHECK (DateOfBirth > '1930-01-01'),
    DriverStatus varchar(20) NOT NULL,
    StartTime smalldatetime NOT NULL,
    SSN bigint NOT NULL,
    HomeAddress varchar(40) NOT NULL,
    AverageScore decimal(10,2) NOT NULL CHECK (AverageScore >= 0) ,CHECK (AverageScore <= 5),
    LicenseNumber bigint NOT NULL,
    InsuranceNumber bigint NOT NULL
)
GO

CONSTRAINT PK_DriverID PRIMARY KEY CLUSTERED (
    DriverID ASC
)
GO

CREATE TABLE TripsAndReservations(
    OrderNumber bigint NOT NULL IDENTITY(1,1),
    TripOrReservation varchar(30) NOT NULL,
    UserID bigint NOT NULL,
    DriverID bigint NOT NULL,
    BookedDate smalldatetime NOT NULL CHECK (BookedDate > GetDate),
    PickUpTime smalldatetime NOT NULL CHECK (PickUpTime > BookedDate),
    DropOffTime smalldatetime NOT NULL,
    PickUpLocation varchar(50) NOT NULL,
    DropOffLocation varchar(50) NOT NULL,
    Cost decimal(10,2) NOT NULL
)
GO

CONSTRAINT PK_OrderNumber PRIMARY KEY CLUSTERED (
    OrderNumber ASC
)
GO

CREATE TABLE DriverBankAccounts(
    BankAccountID bigint IDENTITY(1,1),
    DriverID bigint NOT NULL,
    BankName varchar(10) NOT NULL,
    TypeOfAccount varchar(10) NOT NULL,
    RoutingNumber bigint NOT NULL,
    AccountNumber bigint NOT NULL
)
GO

CONSTRAINT PK_BankAccountID PRIMARY KEY CLUSTERED (
    BankAccountID ASC
)
GO
```

The screenshot shows the continuation of the table definitions from the previous screen. The 'DriverBankAccounts' table is being defined with columns: BankAccountID (bigint, identity(1,1), not null), DriverID (bigint, not null), BankName (varchar(10), not null), TypeOfAccount (varchar(10), not null), RoutingNumber (bigint, not null), and AccountNumber (bigint, not null). A primary key constraint 'CONSTRAINT PK_BankAccountID PRIMARY KEY CLUSTERED' is defined on the BankAccountID column. The 'DriverInsurances' table is being defined with columns: (empty table definition).

```
CONSTRAINT PK_BankAccountID PRIMARY KEY CLUSTERED (
    BankAccountID ASC
)
GO

CREATE TABLE DriverInsurances(
)
```

```

USE [MNDU\nanji]
GO
CREATE TABLE Insurance(
    InsuranceNumber BIGINT NOT NULL IDENTITY(1,1),
    DriverID BIGINT NOT NULL,
    DateOfIssue SMALLDATETIME NOT NULL,
    DateOfExpicy SMALLDATETIME NOT NULL, CHECK (DateOfExpicy > DateOfIssue),
    CONSTRAINT PK_InsuranceID PRIMARY KEY CLUSTERED (
        InsuranceNumber ASC
    )
)
GO

CREATE TABLE DriverLicenses(
    LicenseNumber BIGINT NOT NULL IDENTITY(1,1),
    DriverID BIGINT NOT NULL,
    DateOfIssue SMALLDATETIME NOT NULL,
    DateOfExpicy SMALLDATETIME NOT NULL, CHECK (DateOfExpicy > DateOfIssue),
    CONSTRAINT PK_LicenseNumber PRIMARY KEY CLUSTERED (
        LicenseNumber ASC
    )
)
GO

CREATE TABLE Cars(
    PlateNumber BIGINT NOT NULL IDENTITY(1,1),
    DriverID BIGINT NOT NULL,
    Made VARCHAR(20) NOT NULL,
    Model VARCHAR(30) NOT NULL,
    CarYear SMALLDATETIME NOT NULL,
    Colour VARCHAR(10) NOT NULL,
    Class VARCHAR(10) NOT NULL,
    NumberOfPassengers INT NOT NULL,
    NumberOfBags INT NOT NULL,
    CONSTRAINT PK_PlateNumber PRIMARY KEY CLUSTERED (
        PlateNumber ASC
    )
)
GO

```

Connected: (1/1)

DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\anji ... UberCompetitor 00:00:00 0 rows

Item(s) Saved.

En 62 Col 37 Ch 37 INS

CH M R A ^ ⌂ 英 M 11:56 2019/4/15

D. Stored Procedures

(1) This procedure is used to view some required information about drivers whose license is going to expire between two specific point-in-times.

```

USE [UberCompetitor]
GO
CREATE PROCEDURE Expire
    @DateMin VARCHAR(50) = NULL,
    @DateMax VARCHAR(50) = NULL
AS
BEGIN
    IF (@DateMin IS NULL OR @DateMax IS NULL)
        THROW 50001, 'Please type in required parameters', 1;
    IF NOT ISDATE(@DateMin) = 1 AND ISDATE(@DateMax) = 1
        THROW 50001, 'Please make sure the format of parameters is valid', 1;
    IF CAST(@DateMin AS DATETIME) > CAST(@DateMax AS DATETIME)
        THROW 50001, 'The DateMin parameter must be earlier than DateMax', 1;

    ELSE
        SELECT D.DriverID, FirstName + ' ' + LastName AS DriverName, PhoneNumber, D.LicenseNumber, DateOfIssue, DateOfExpicy
        FROM Drivers AS D JOIN DriverLicenses AS DL ON D.DriverID = DL.DriverID
        WHERE DateOfExpicy > @DateMin AND DateOfExpicy < @DateMax
        ORDER BY DateOfExpicy DESC;
END

```

Messages

Commands completed successfully.

Query executed successfully.

DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\anji ... UberCompetitor 00:00:00 0 rows

Ready

Ln 16 Col 72 Ch 72 INS

EN R A ^ ⌂ ENG 15:03 2019/4/15

Test and result:

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery9.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (66))' is open. The code in the query window is:

```

BEGIN TRY
    EXEC Expire '2021-01-01', '2024-01-01';
END TRY
BEGIN CATCH
    PRINT 'Error Number: ' + CONVERT(varchar(100), ERROR_NUMBER());
    PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;

```

The results pane shows a table with two rows of data:

DriverID	DriverName	PhoneNumber	LicenseNumber	DateOfIssue	DateOfExpiry
1	Frank Oden	(559) 555-7473	538769	2002-05-23 00:00:00	2022-05-23 00:00:00
2	Chris Howard	(314) 575-6831	038769	2002-05-23 00:00:00	2022-05-23 00:00:00

The status bar at the bottom indicates 'Query executed successfully.' and '15:04 2019/4/15'. The taskbar shows various application icons.

Error handling:

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery10.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (66))' is open. The code in the query window is identical to the one in the previous screenshot:

```

BEGIN TRY
    EXEC Expire '2021-01-01', '2024-01-01';
END TRY
BEGIN CATCH
    PRINT 'Error Number: ' + CONVERT(varchar(100), ERROR_NUMBER());
    PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;

```

The results pane shows an error message:

```

Error Number: 50001
Error Message: Please type in required parameters

```

The status bar at the bottom indicates 'Query executed successfully.' and '15:05 2019/4/15'. The taskbar shows various application icons.

SQLQuery9.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (66)) - Microsoft SQL Server Management Studio

```

SQLQuery10.sql - ...MNDU\nanji (53)* SQLQuery9.sql - ...MNDU\nanji (66)* Object Explorer

BEGIN TRY
    EXEC Expire '2024-01-01', '2021-01-01';
END TRY
BEGIN CATCH
    PRINT 'Error Number: ' + CONVERT(varchar(100), ERROR_NUMBER());
    PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;


```

100 %

Messages

Error Number: 50001
Error Message: The Datetime parameter must be earlier than DatedMax

100 %

Query executed successfully.

Ready

Ln 3 Col 1 Ch 1 INS EN 15:06 2019/4/15

(2) This procedure is used to return information about customers whose cost on

their trip is larger than a certain value.

SQLQuery11.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (58)) - Microsoft SQL Server Management Studio

```

SQLQuery12.sql - ...MNDU\nanji (54)* SQLQuery11.sql - ...MNDU\nanji (58)* Object Explorer

use UberCompetitor
go
CREATE PROCEDURE [ExpensiveTrips]
    @cost int = null
    as
        If @cost < (
            select min(cost)
            from TripsAndReservations
        )
        throw 50001, 'The given cost cannot be smaller than the least expensive trip',1;
        If @cost is null
        throw 50002,'Please type in a needed amount of money',1;
        Select C.UserID, DriverID, FirstName + ' ' + LastName as CustomerName, PhoneNumber as CustomerPhoneNumber, NumberOfTripsWithinAYear, Cost
        From Customers as C join TripsAndReservations as T on
        C.UserID = T.UserID
        Where Cost > @cost
        Order by cost DESC;


```

100 %

Messages

Command completed successfully.

100 %

Query executed successfully.

Ready

Ln 1 Col 19 Ch 19 INS CH M R ENG 15:45 2019/4/15

Test and result:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a code editor window containing the following T-SQL script:

```


BEGIN TRY
    exec ExpensiveTrips 15;
END TRY
BEGIN CATCH
    PRINT 'Error Number: ' + CONVERT(varchar(100), ERROR_NUMBER());
    PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;


```

Below the code editor, the results pane displays a table titled "Messages" with the following data:

User ID	Driver ID	Customer Name	Customer Phone Number	Number of Trip Within Year	Cost	
1	1000003	200003	Steve Jackson	(800) 555-2701	7	24.30
2	1000003	200005	Luke Grant	(509) 555-1551	12	18.00
3	100001	200001	Leon Hill	(800) 555-1205	24	16.00
4	100004	200004	Zach Lowe	(509) 555-9999	10	15.30

At the bottom of the results pane, it says "Query executed successfully." and shows the status bar with "DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 | 4 rows".

Error handling:

The screenshot shows the Microsoft SQL Server Management Studio interface. The code editor window contains the same T-SQL script as before:

```


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


```

The results pane shows the following error message:

```


Error Number: 80002
Error Message: Please type a needed amount of money


```

At the bottom of the results pane, it says "Query executed successfully." and shows the status bar with "DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 | 0 rows".

SQLQuery13.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (53)) - Microsoft SQL Server Management Studio

```

SQLQuery13.sql -...MNDU\nanji (53)* SQLQuery12.sql -...MNDU\nanji (54)* SQLQuery11.sql -...MNDU\nanji (58)* Object Explorer

BEGIN TRY
    exec ExpensiveTrips $;
END TRY
BEGIN CATCH
    PRINT 'Error Number: ' + CONVERT(varchar(100), ERROR_NUMBER());
    PRINT 'Error Message: ' + CONVERT(varchar(100), ERROR_MESSAGE());
END CATCH;


```

100 %

Messages

Error Number: 50001
Error Message: The given cost cannot be smaller than the least expensive trip

Query executed successfully.

Ready

Ln 2 Col 22 Ch 22 INS CH M 15:45 2019/4/15

E. Functions

(1) Use a function called 'CarDateRange' to return the information about the cars produced within a period.

SQLQuery2.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (55)) - Microsoft SQL Server Management Studio

```

SQLQuery3.sql -...7MNDU\nanji (59)* SQLQuery2.sql -...7MNDU\nanji (55)* SQLQuery1.sql -...7MNDU\nanji (53)* Object Explorer

CREATE FUNCTION CarDateRange
    (@DateMax smalldatetime, @DateMin smalldatetime)
    RETURNS TABLE
    RETURN (SELECT * FROM Cars
    WHERE CarYear BETWEEN @DateMax AND @DateMin);


```

100 %

Messages

Commande completed successfully.

Query executed successfully.

Ready

Ln 6 Col 46 Ch 46 INS EN 21:13 2019/4/15

Test and result:

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery3.sql' is open, displaying the following SQL code:

```
select * from
dbo.CarDateRange('2013-01-01','2018-01-01')
```

The results pane shows a table with the following data:

	PlateNumber	DriverID	Made	Model	Car's	Color	Class	NumberOfPassengers	NumberOfBags
1	500006	200002	Toyota	C	2015-01-01 00:00:00	Brown	B	6	8
2	500007	200003	Chevy	D	2017-01-01 00:00:00	White	C	4	6
3	500008	200004	Dodge	A	2015-01-01 00:00:00	Grey	A	7	5

The status bar at the bottom indicates 'Query executed successfully.' and shows the system information: DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 | 3 rows.

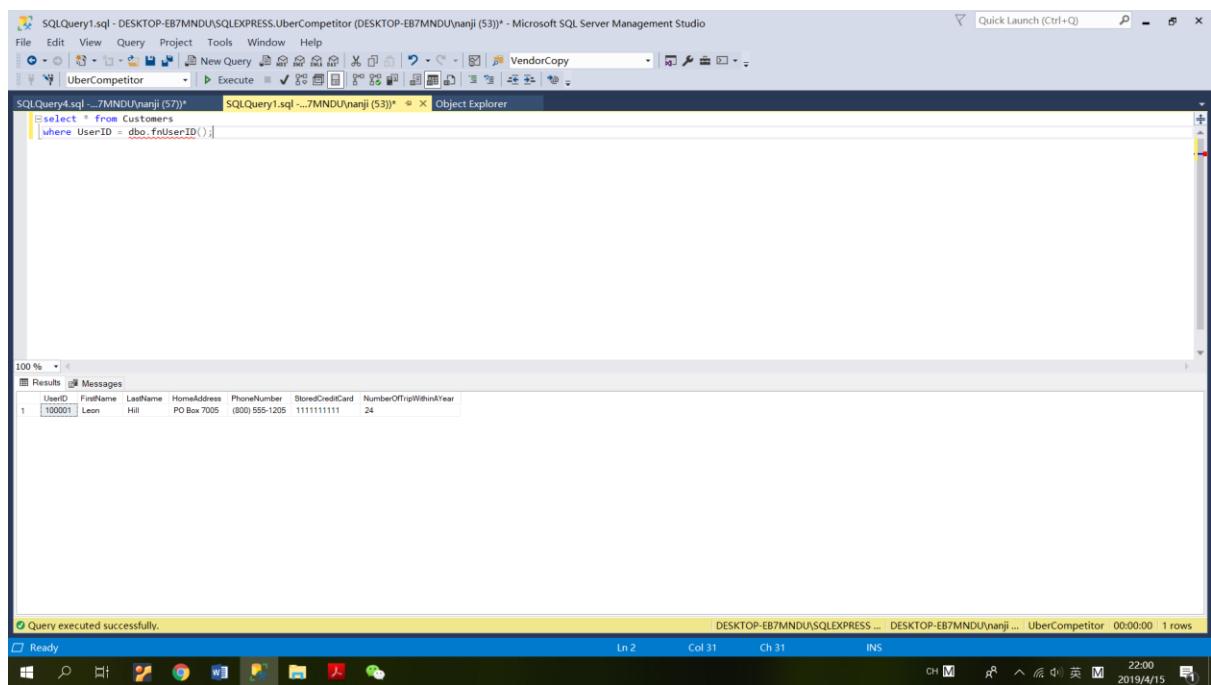
(2) Use a function called 'fnUserID' to return the ID of the customer who has taken the most trips within a year.

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery4.sql' is open, displaying the following SQL code:

```
CREATE FUNCTION fnUserID()
RETURNS BIGINT
BEGIN
RETURN
(SELECT UserID FROM Customers
WHERE NumberoftripwithinAYear =
(SELECT MAX(NumberoftripwithinAYear)
FROM Customers));
END;
```

The status bar at the bottom indicates 'Command completed successfully.'

Test and results:



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery4.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESktop-EB7MNDU\nanji (S))' is open. The query is:

```
select * from Customers  
where UserID = dbo.fnUserID();
```

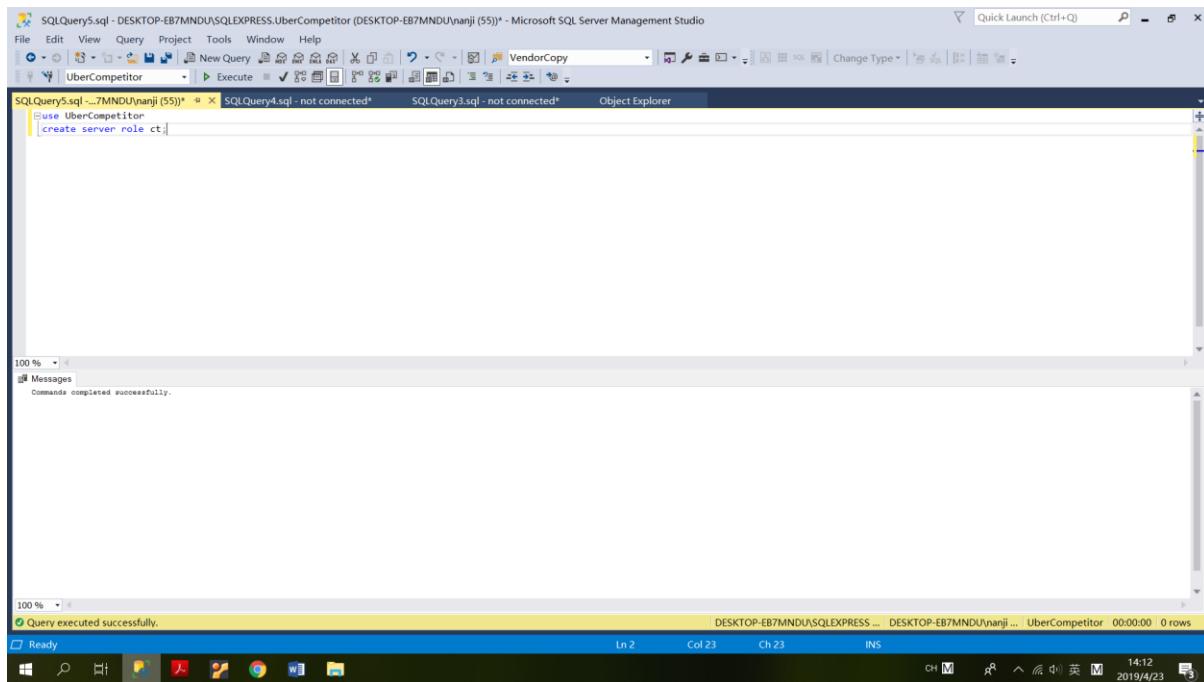
The results pane shows one row of data:

UserID	FirstName	LastName	HomeAddress	PhoneNumber	StoredCreditCard	NumberOfTripWithinAYear
100001	Leon	Hill	PO Box 7005	(800) 555-1205	1111111111	24

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

Part V: Security

A. Role



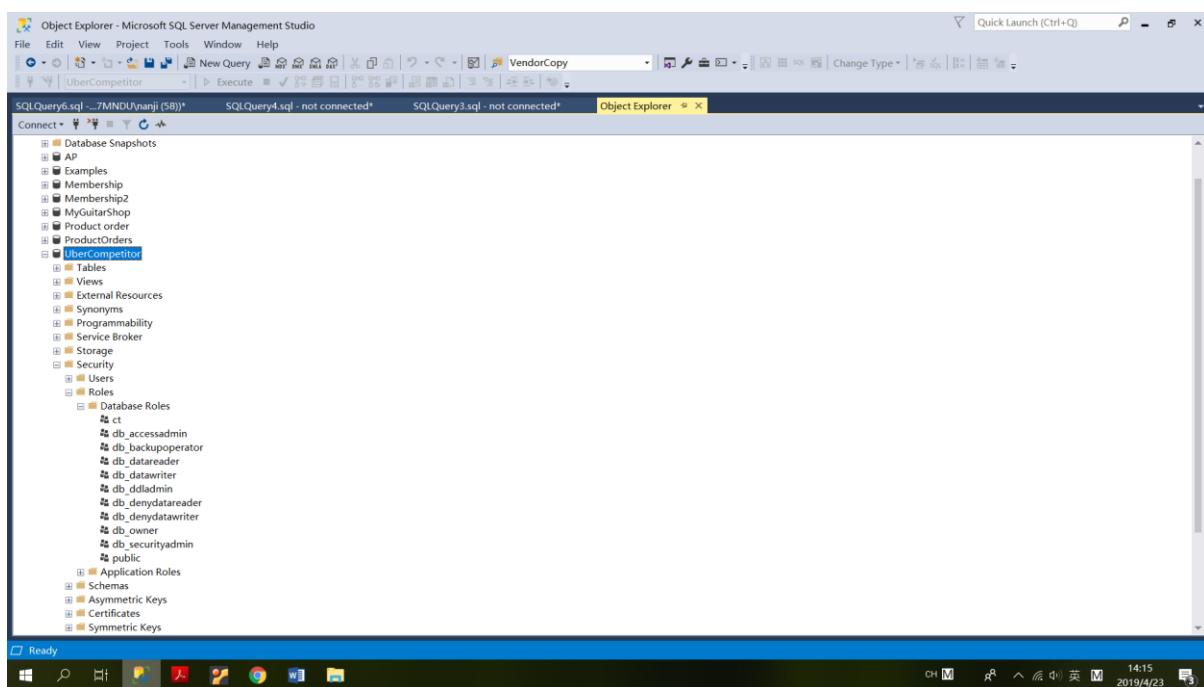
The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'UberCompetitor' is selected. In the center pane, a query window titled 'SQLQuery5.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (55))' contains the T-SQL command:

```
[use UberCompetitor
CREATE SERVER ROLE ct]
```

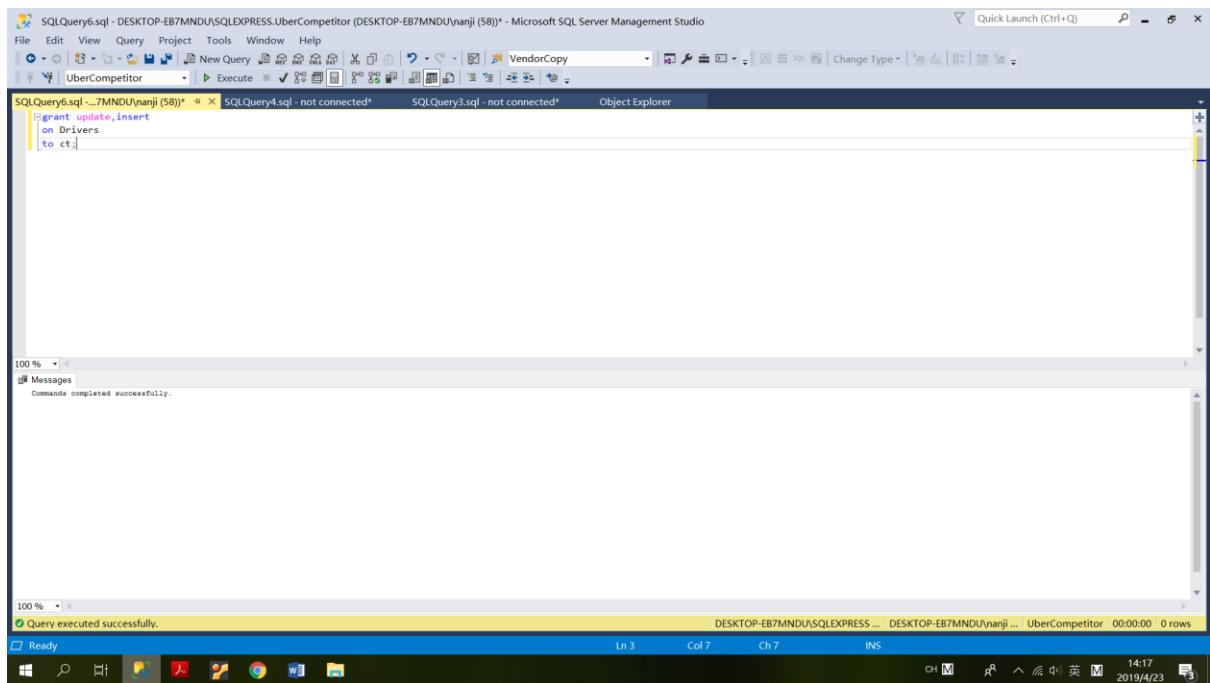
The status bar at the bottom right indicates the command was completed successfully.

Create a server role called 'ct' in UberCompetitor database.

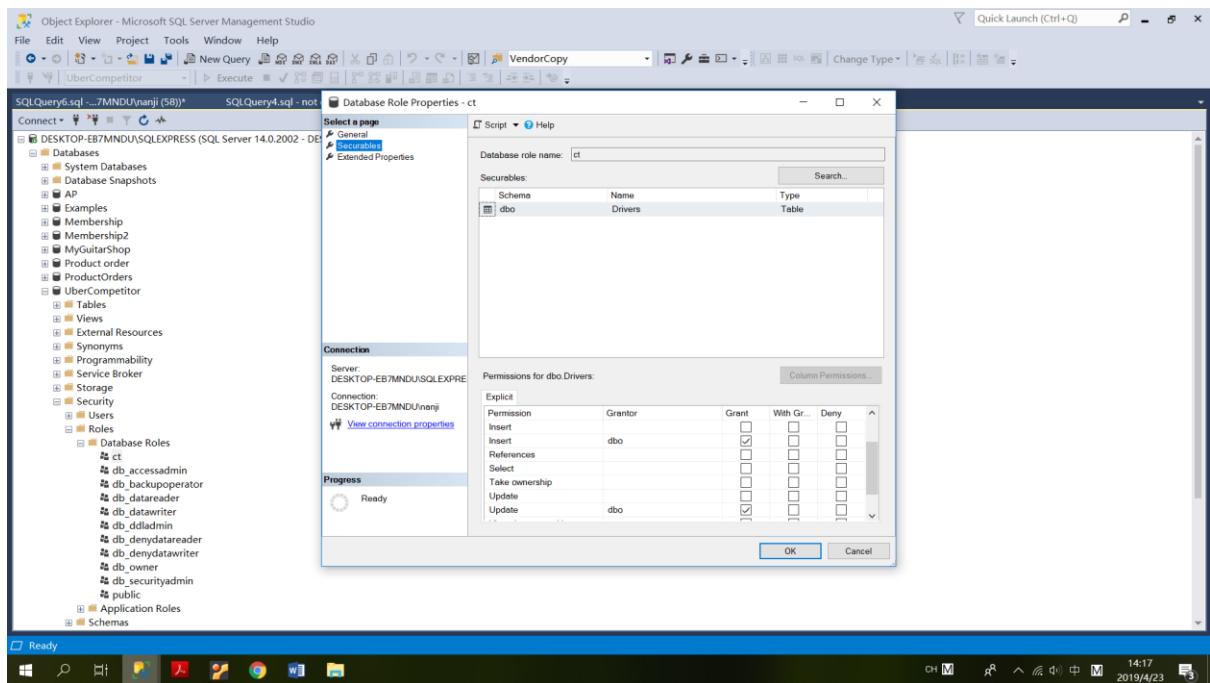
Result:



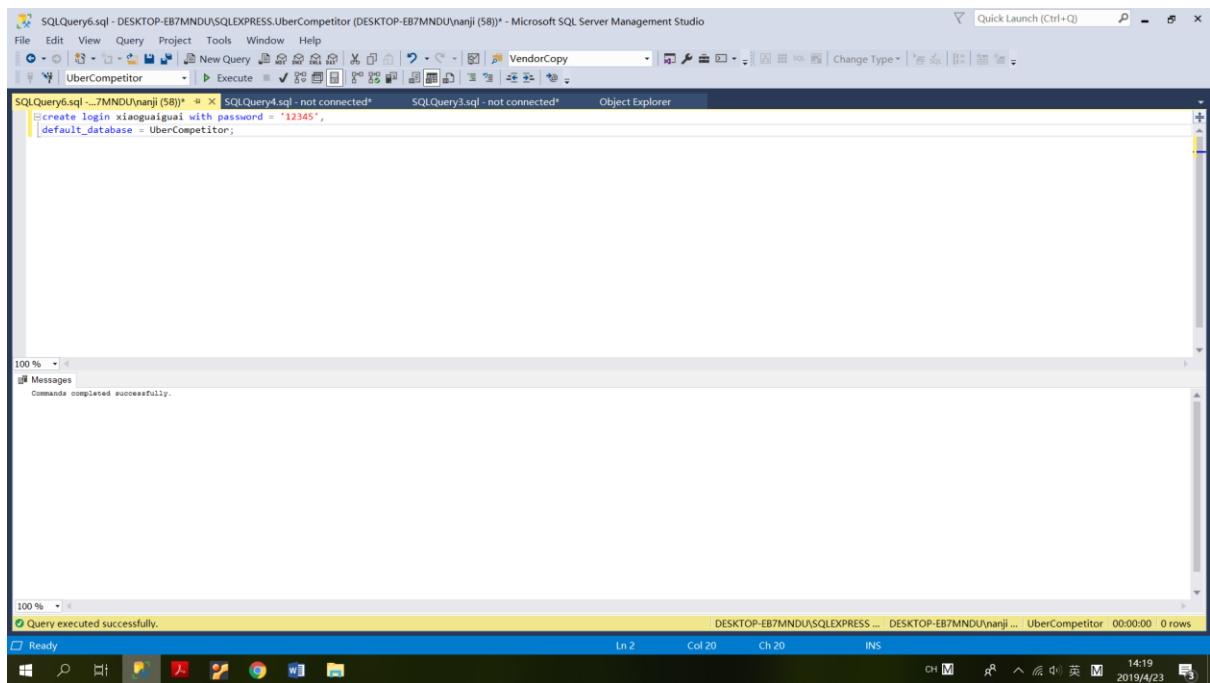
B. Permissions:



Result:



C. Login

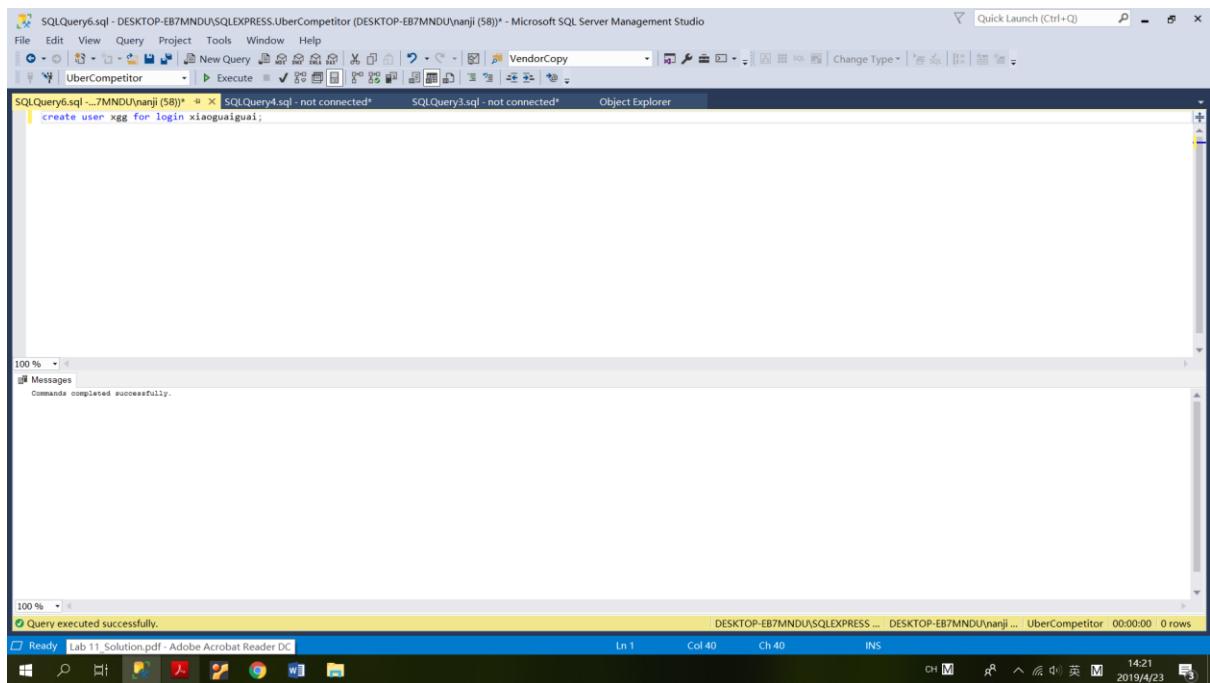


The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'UberCompetitor' is selected. In the center pane, a query window displays the following T-SQL code:

```
create login xiaoguaigui with password = '12345';
default_database = UberCompetitor;
```

The status bar at the bottom right indicates the session is connected to 'DESKTOP-EB7MNDU\SQLEXPRESS' and the current date and time are '2019/4/23 14:19'. The message bar at the bottom shows 'Query executed successfully.'

D. User

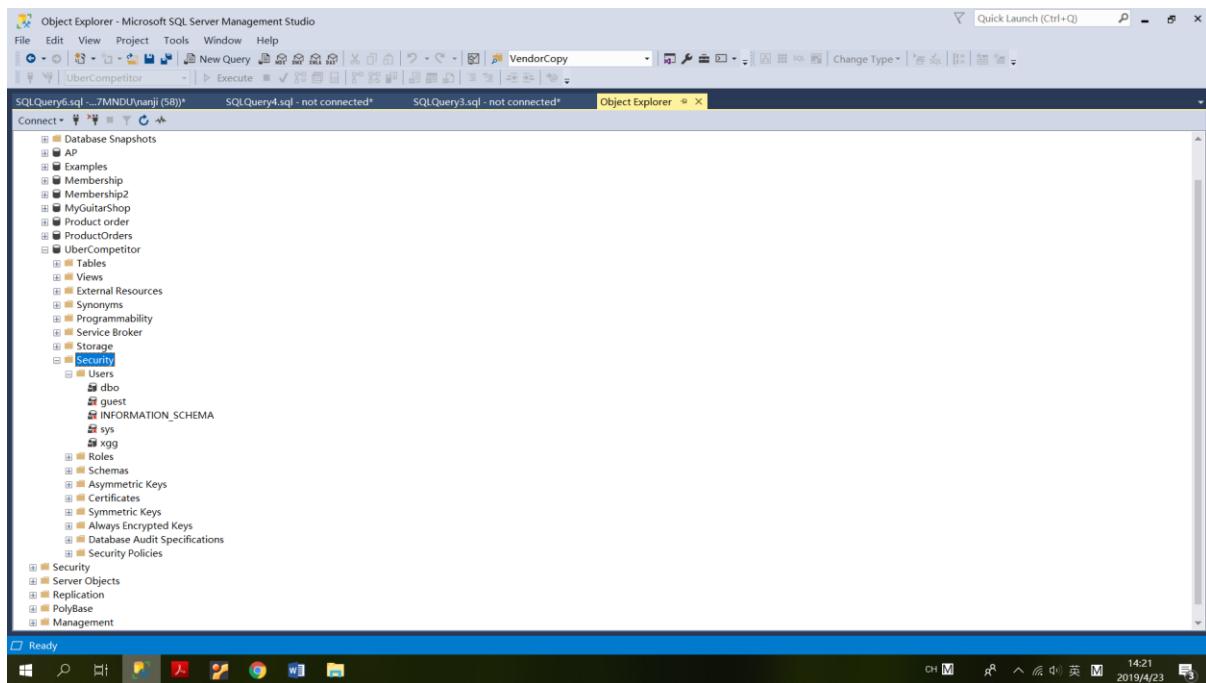


The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'UberCompetitor' is selected. In the center pane, a query window displays the following T-SQL code:

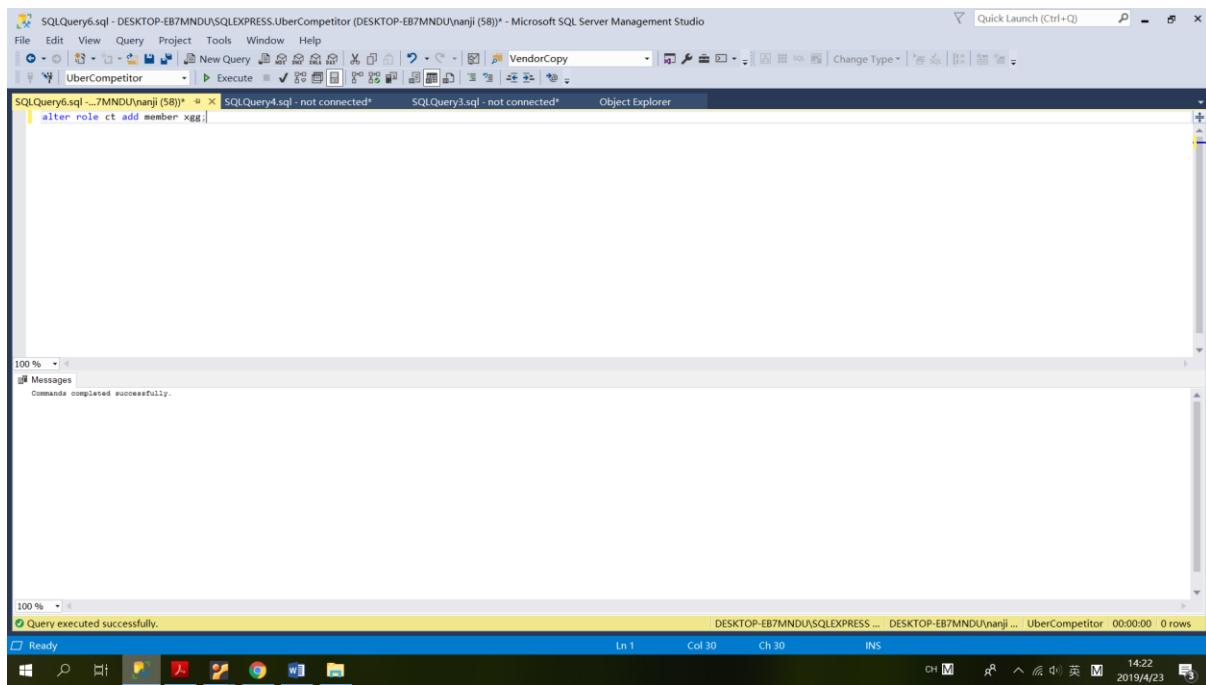
```
create user xgg for login xiaoguaigui;
```

The status bar at the bottom right indicates the session is connected to 'DESKTOP-EB7MNDU\SQLEXPRESS' and the current date and time are '2019/4/23 14:21'. The message bar at the bottom shows 'Query executed successfully.'

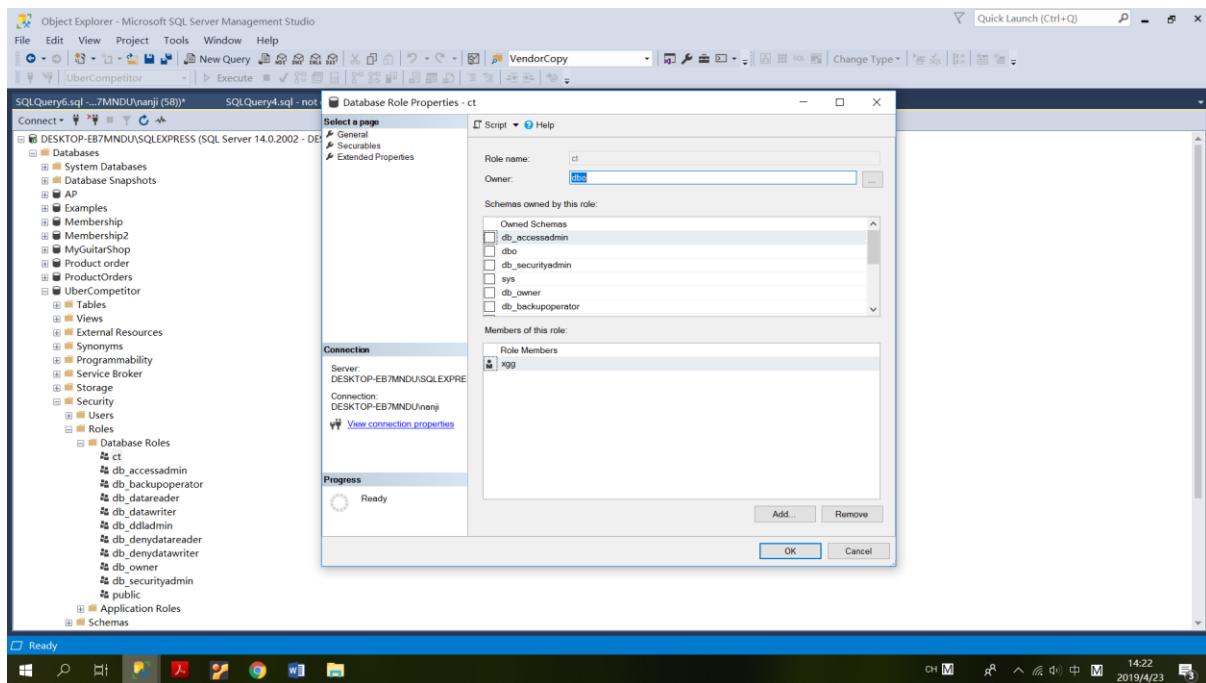
Result:



E. Role Member



Result:



Part VI: Conclusion/Project Analysis/Remark:

Conclusion:

Through this project, I successfully establish a reliable and useful database for a Uber-like company. Through the testing, it can be verified that my database works well.

Project analysis:

I do not think it is a hard work for me because our labs and slides cover almost everything needed for the project. Views, stored procedures, functions, triggers I have practiced using all these things in the labs. Some problems occurred when I was trying to implement the database, but finally I found the solution from the textbook.

Remark:

I do think this is a very good project because it covers almost everything the professor has talked about this semester. I can tell that the professor worked hard on designing such a project. I was not only working on this project but also reviewing those SQL knowledges we have learnt during this semester at the same time. After doing this project, I can say that I have had a better understanding of SQL syntax. Moreover, I have been more proficient in using database management system. Since my current direction is about data science, this project along with this course helps me a lot.