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

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

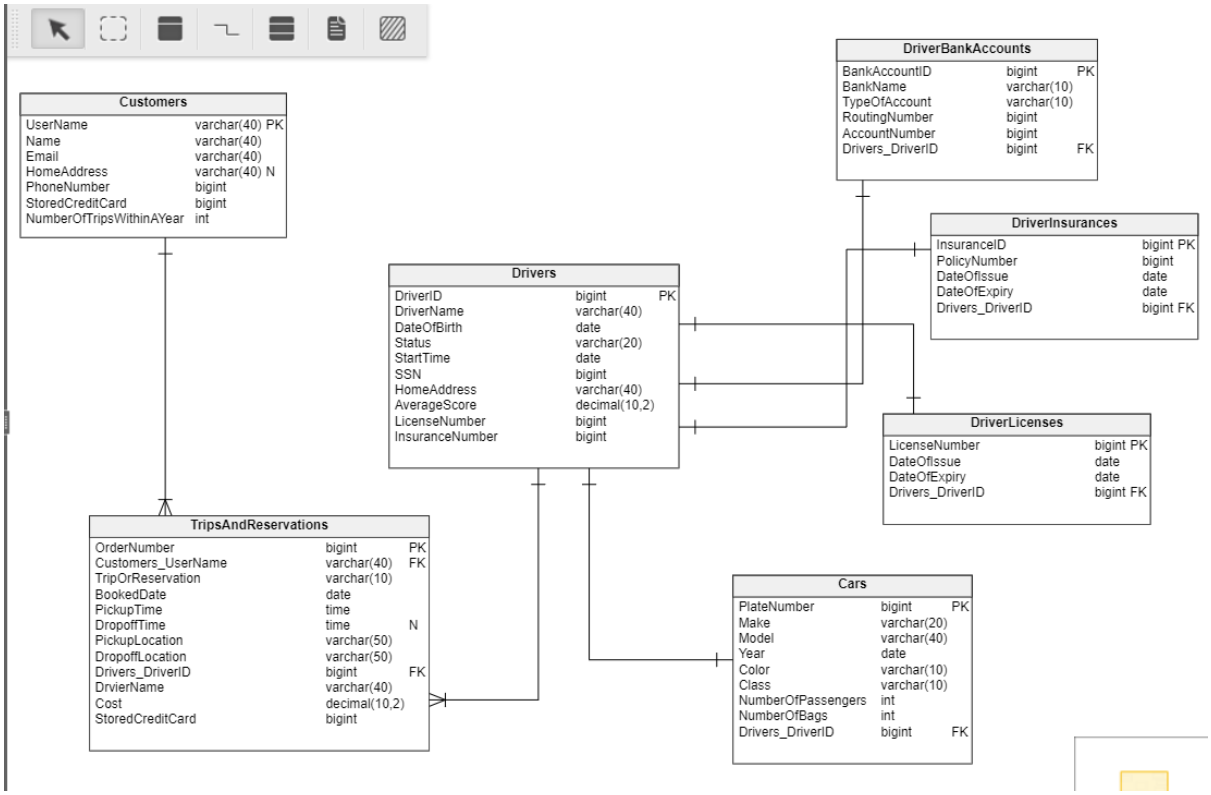
A. Tables：

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**'.

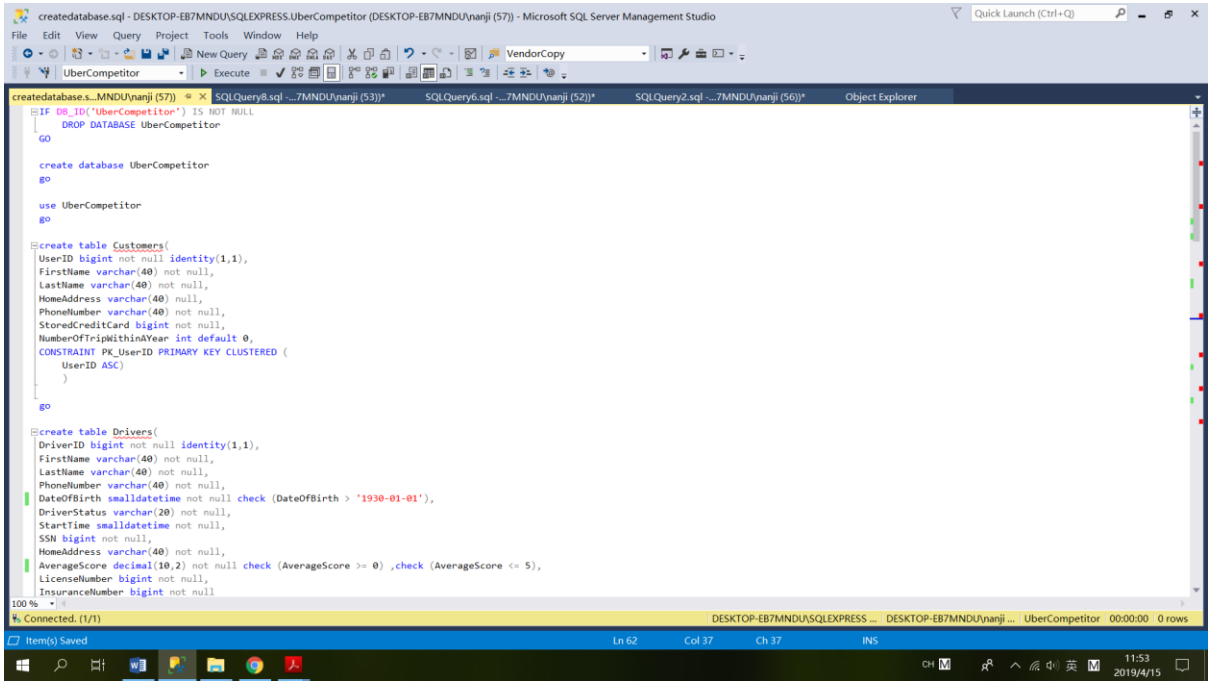
The column '**Name**' in '**Customers**' and '**DriverName'** in '**Drivers**' will be splatted up into '**FirstName**' and '**LastName**'.)

B. Relations:

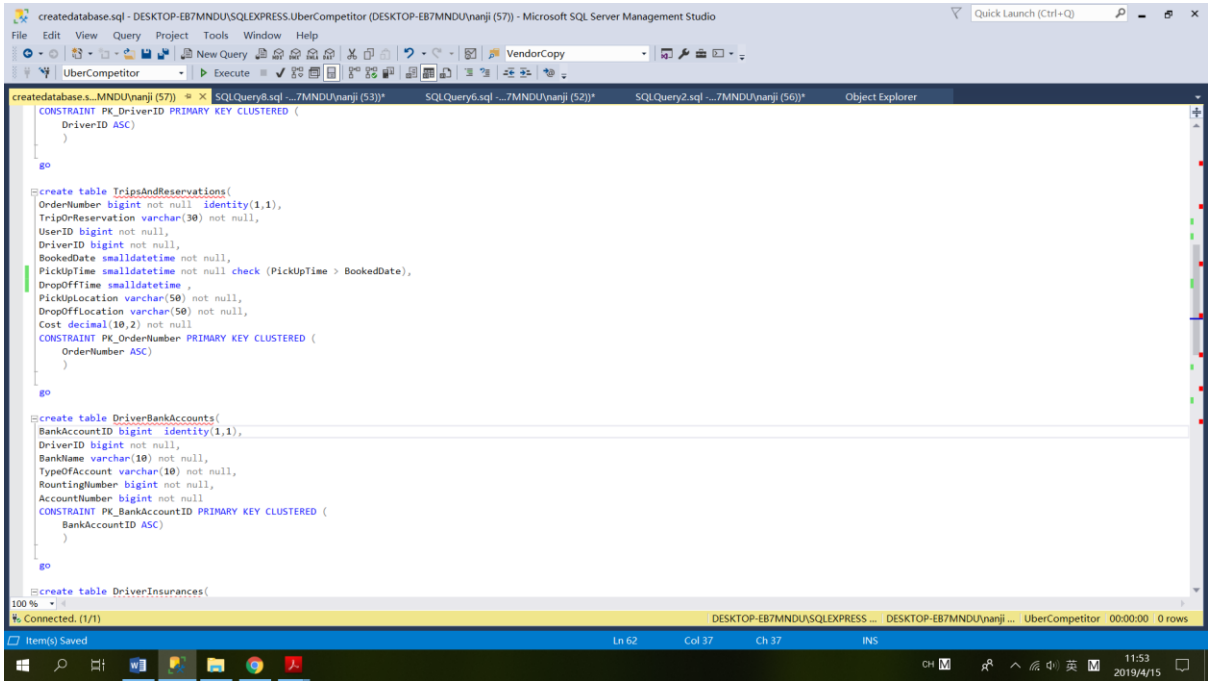
We assume that each driver can only have one bank account, insurance license and car registered. Therefore, the relation between '**Drivers**' and '**DriverBankAccounts**', '**DriverInsurance**', '**DriverLicenses**' and '**Cars**' are 1 to 1. The relation between '**Customers**' and **'Drivers'** is 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

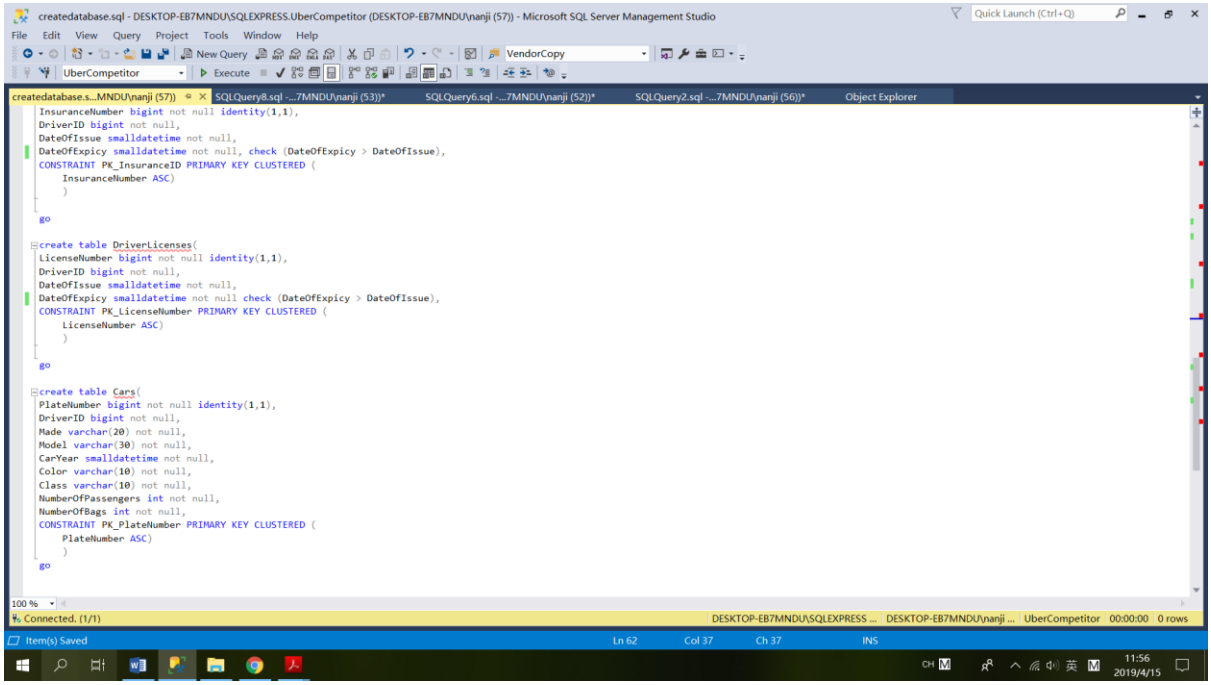
A. Code



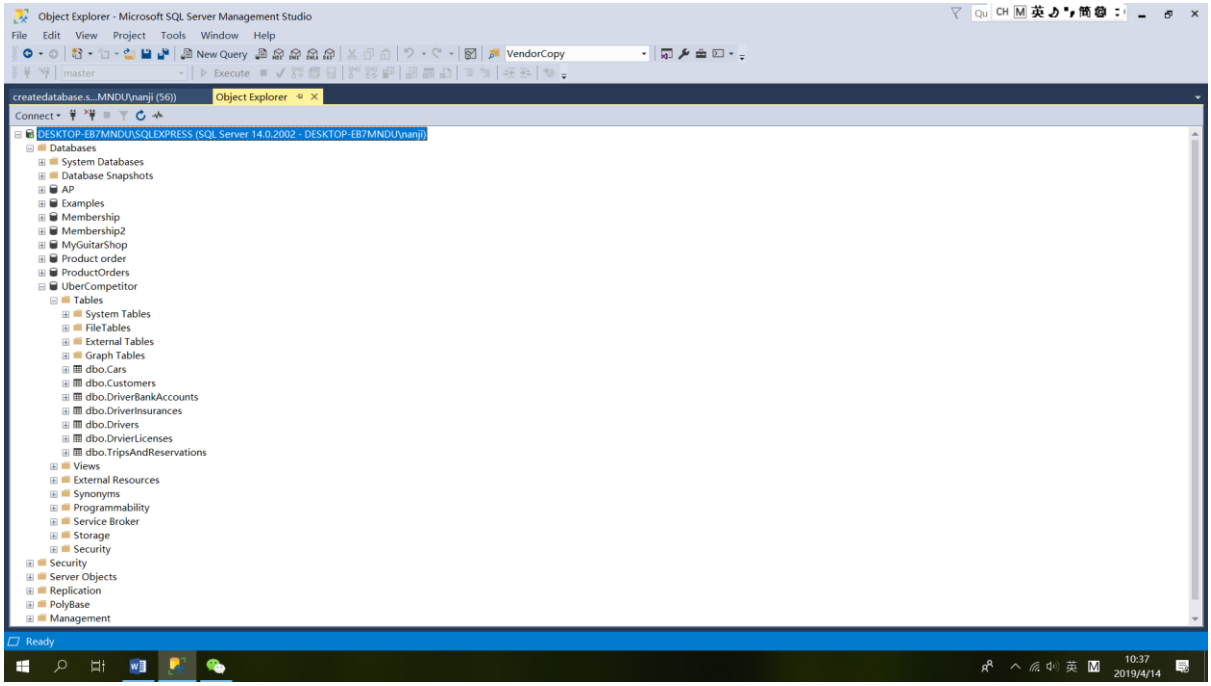
```
IF DB_ID('UberCompetitor') IS NOT NULL
    DROP DATABASE UberCompetitor
GO

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,
NumberOfTripWithinAYear 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
```



```
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,
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,
RountingNumber bigint not null,
AccountNumber bigint not null
CONSTRAINT PK_BankAccountID PRIMARY KEY CLUSTERED (
    BankAccountID ASC)
    )
go

create table DriverInsurances(
```

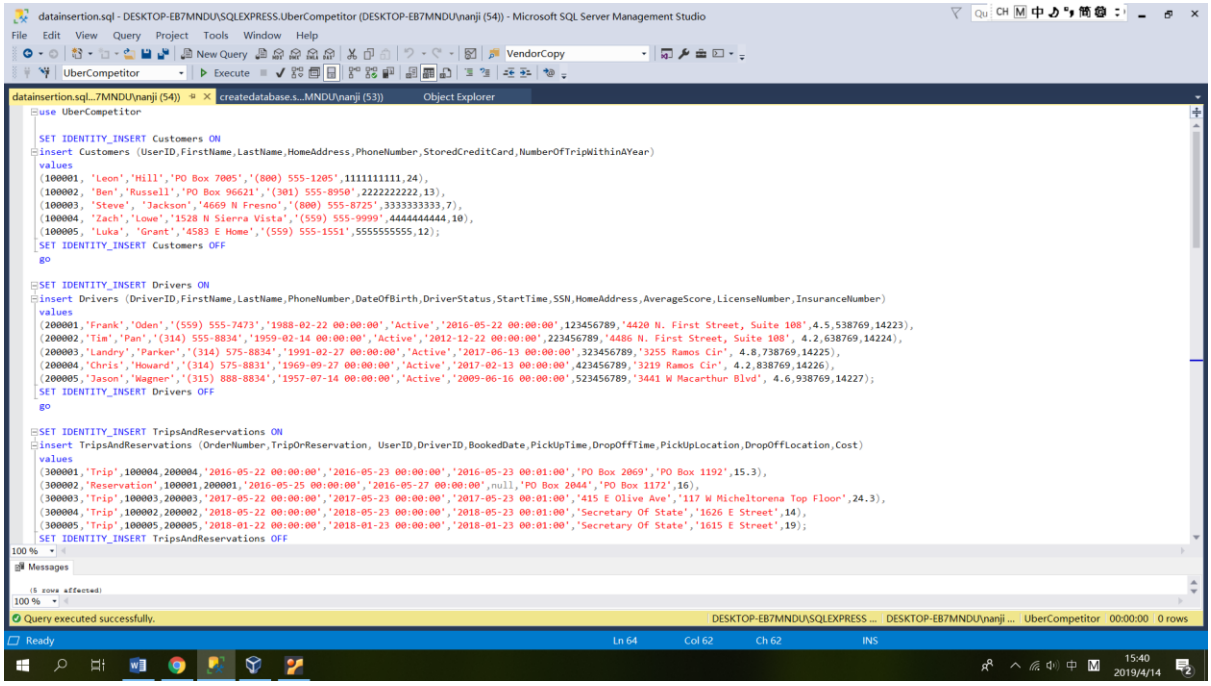

createdatabase.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (57)) - Microsoft SQL Server Management Studio
InsuranceNumber bigint not null identity(1,1),
DriverID bigint not null,
DateOfIssue smalldatetime not null,
DateOfExpicy smalldatetime not null, check (DateOfExpicy > 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,
Color 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

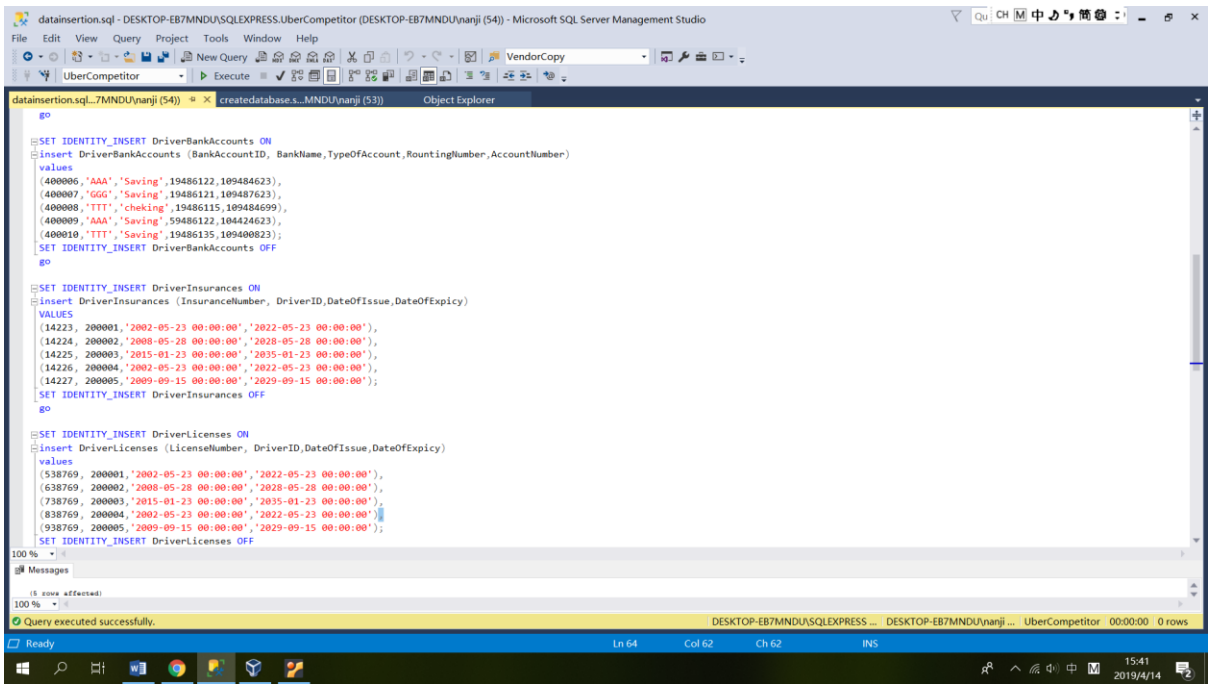
B. Results

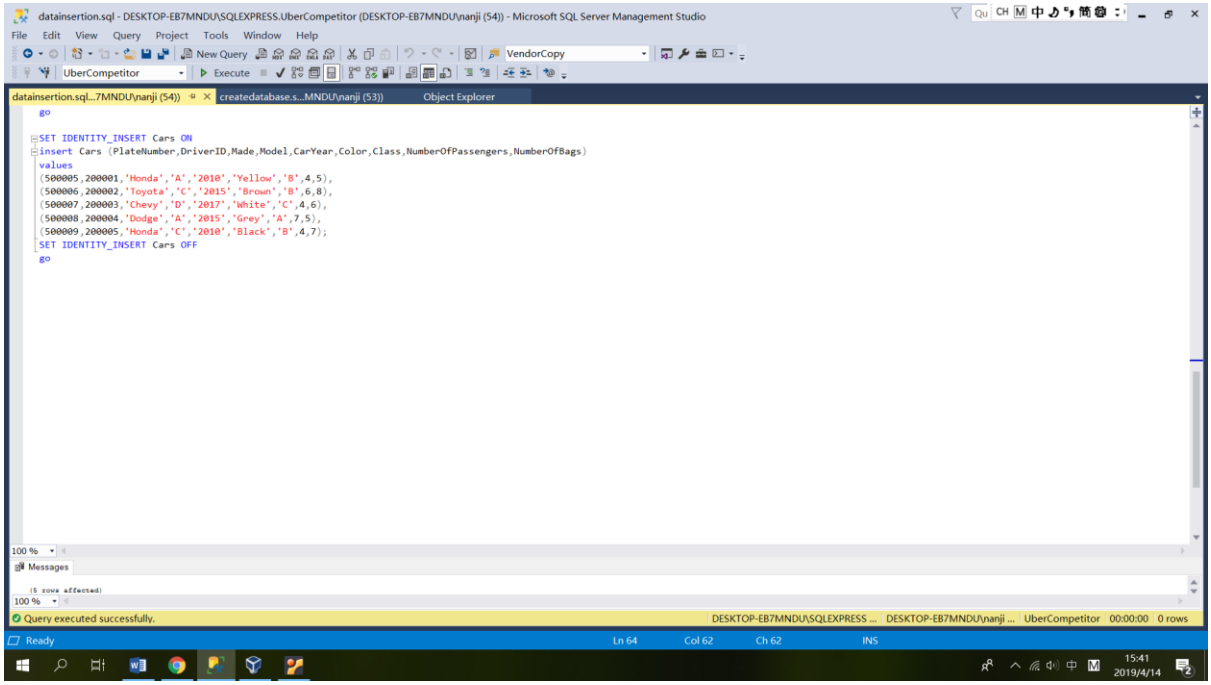

Object Explorer - Microsoft SQL Server Management Studio
DESKTOP-EB7MNDU\SQLEXPRESS (SQL Server 14.0.2002 - DESKTOP-EB7MNDU\nanji)
Databases
System Databases
Database Snapshots
AP
Examples
Membership
Membership2
MyGuitarShop
Product order
ProductOrders
UberCompetitor
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Cars
dbo.Customers
dbo.DriverBankAccounts
dbo.DriverInsurances
dbo.Drivers
dbo.DrvierLicenses
dbo.TripsAndReservations
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
PolyBase
Management

Part III: Data Insertion

A. Code

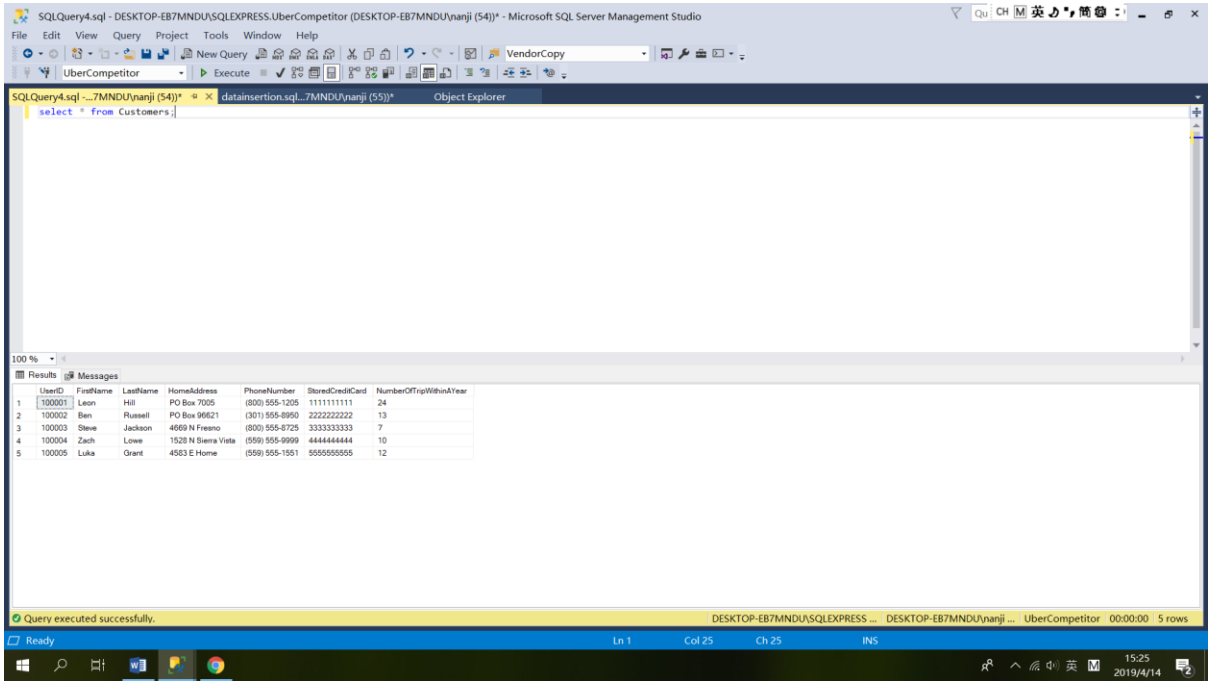




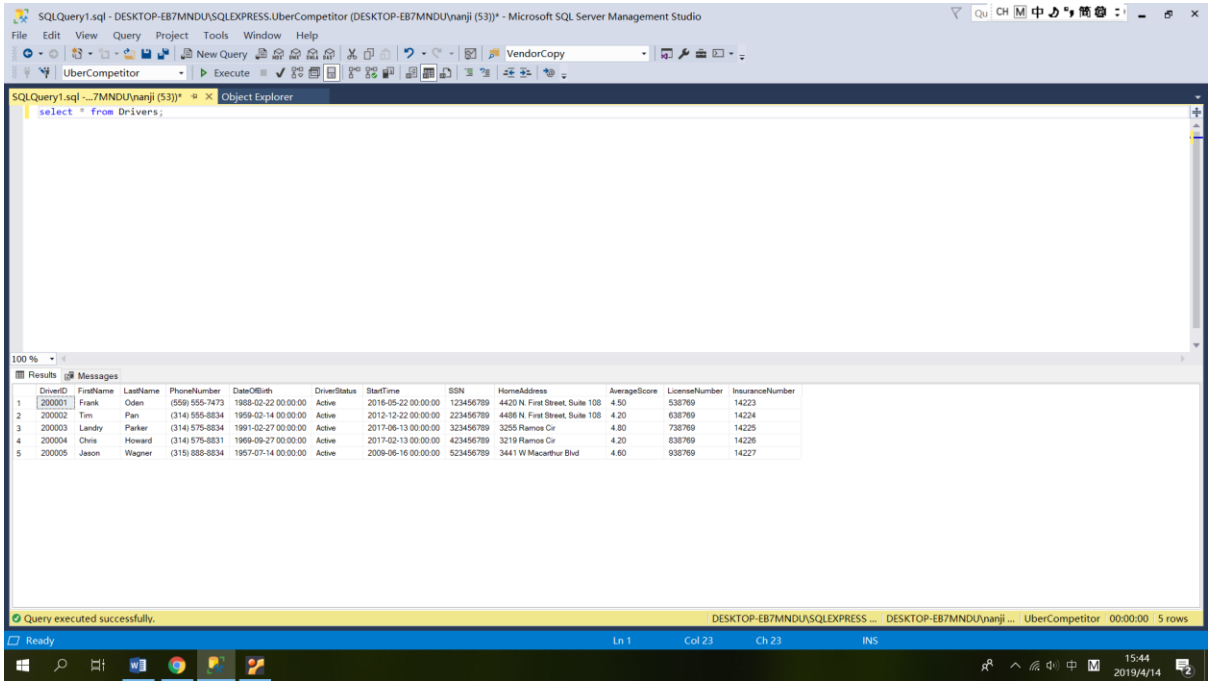


B. Results

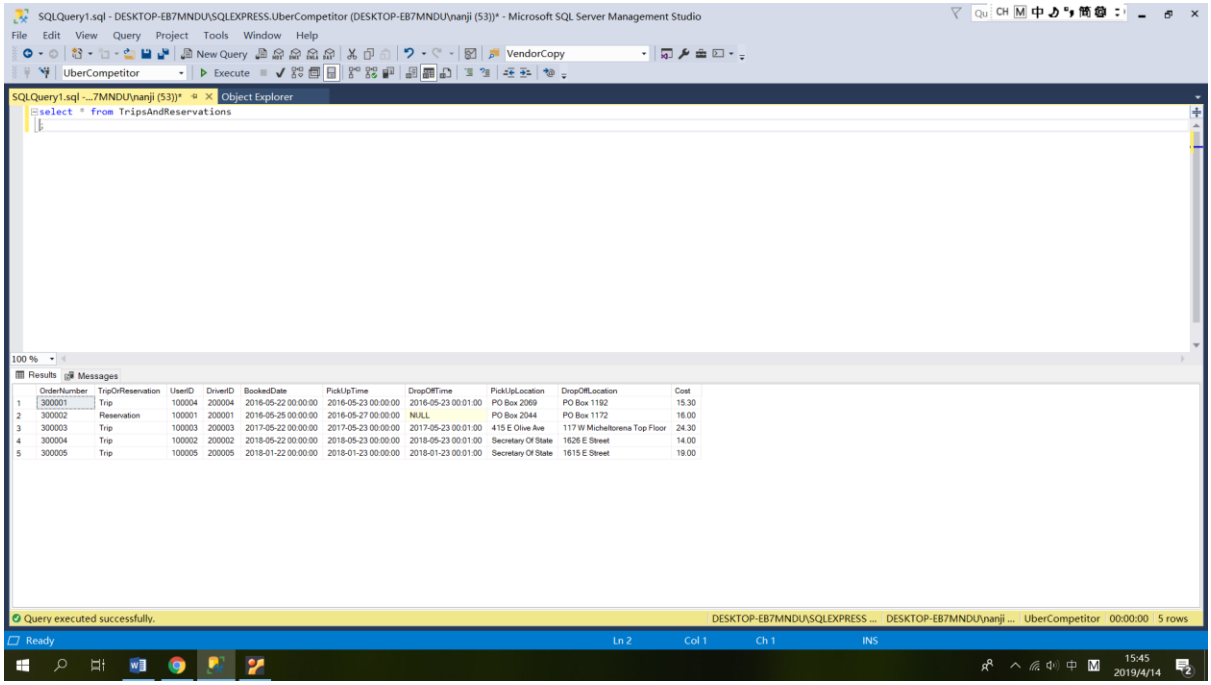
Customers：



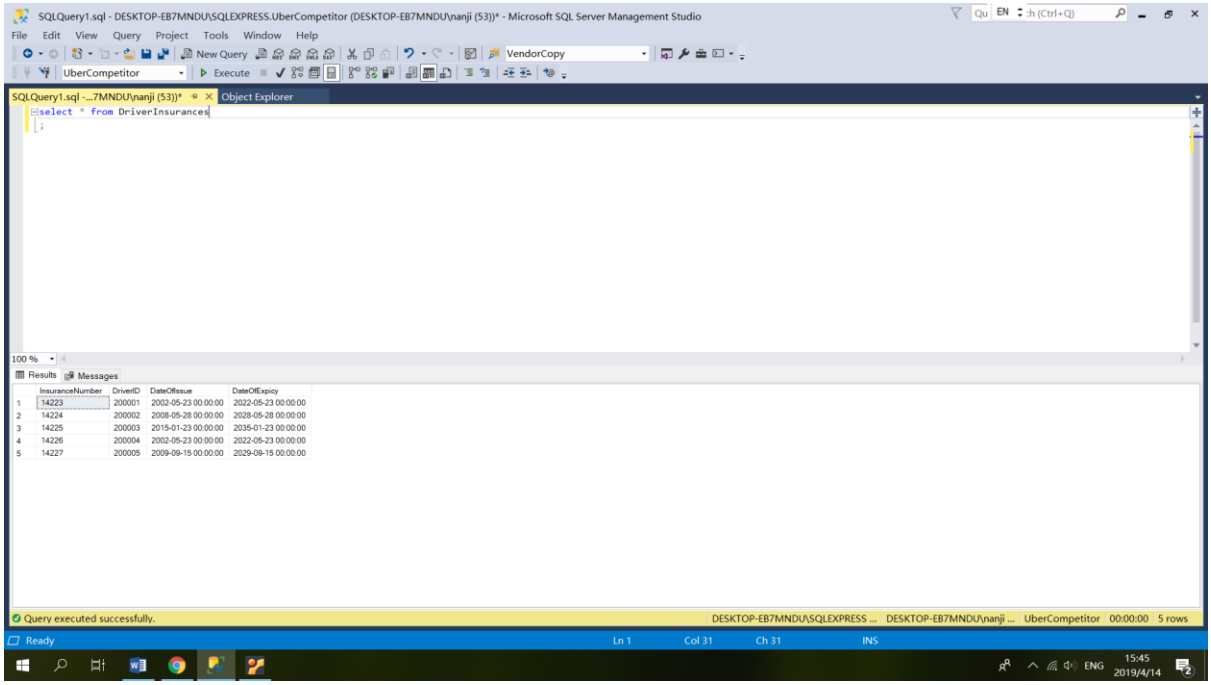
Drivers:



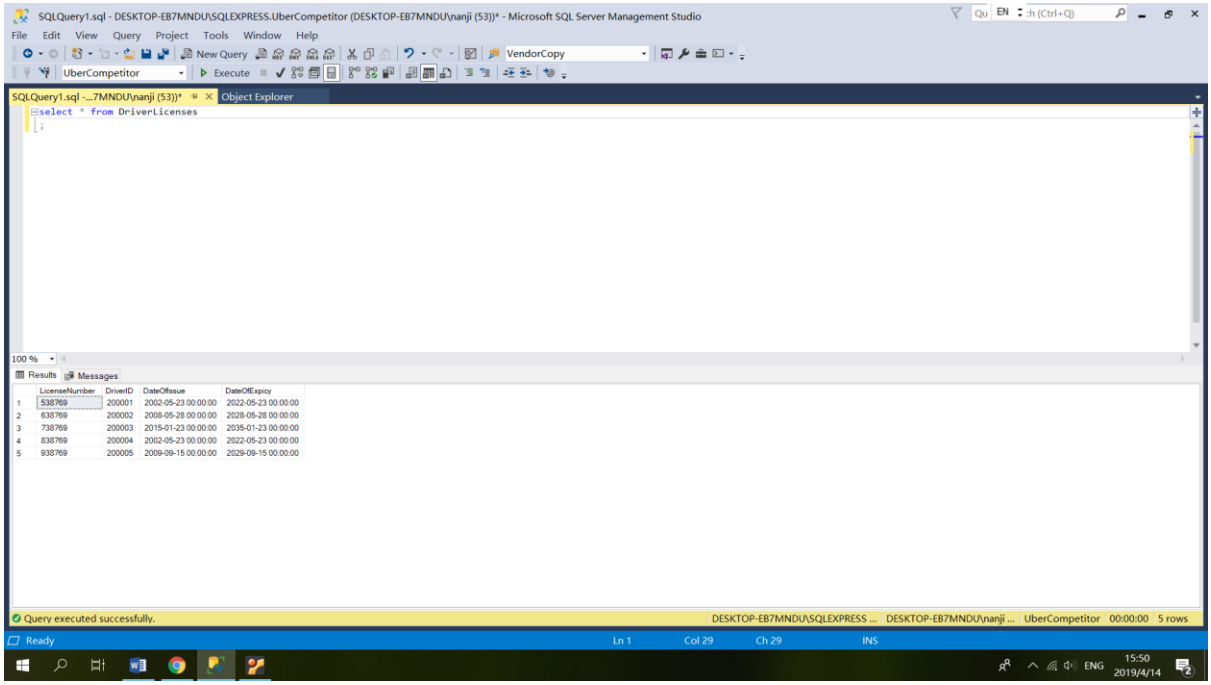
TripsAndReservations:



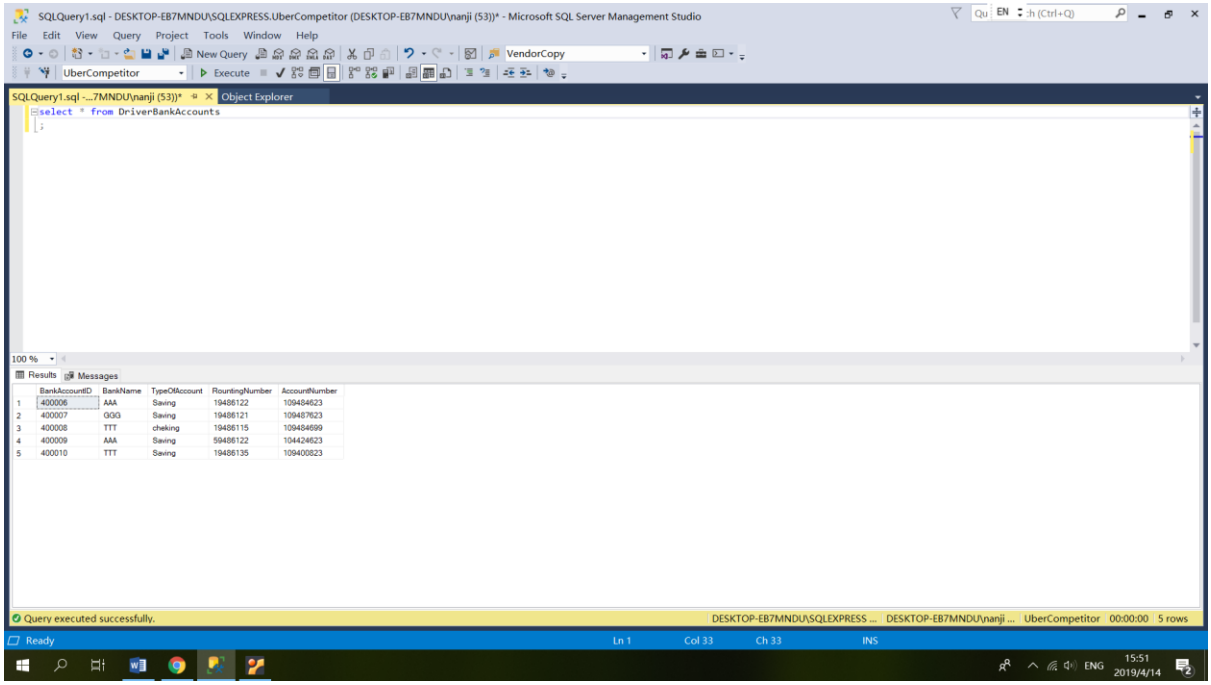
DriverInsurances:



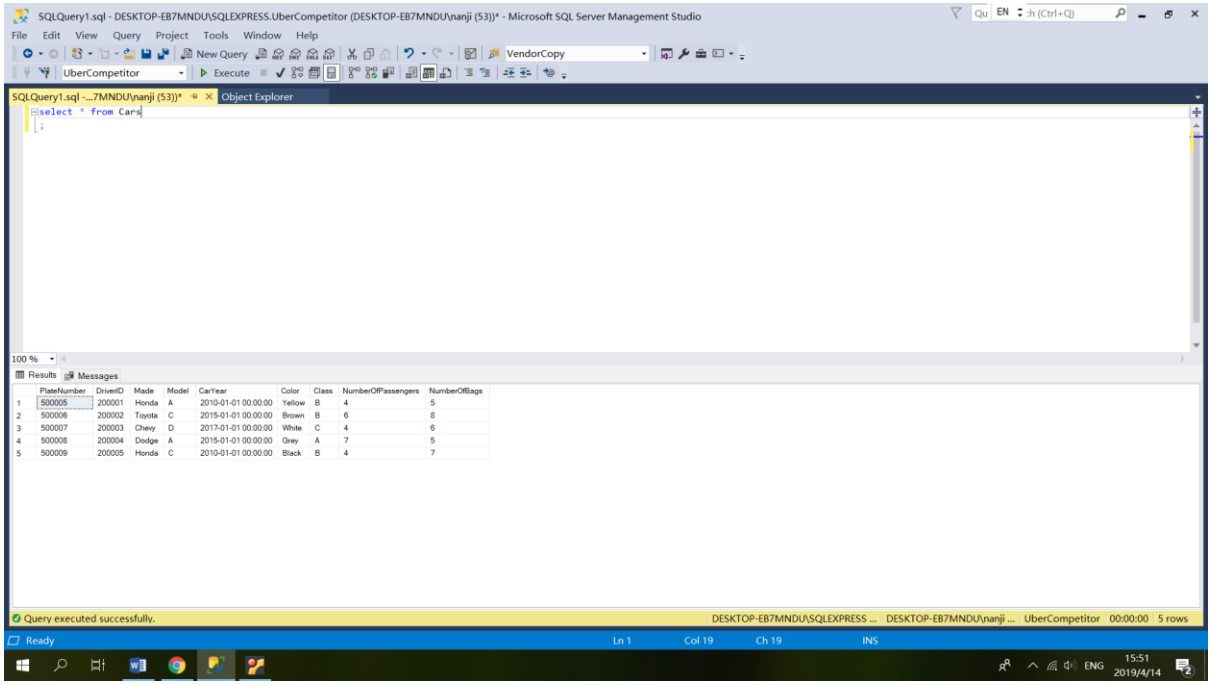
DriverLicenses:



DriverBankAccounts:


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
VendorCopy
UberCompetitor
Execute
SQLQuery1.sql -...7MNDU\nanji (53))*
Object Explorer
select * from DriverBankAccounts
100 %
Results
Messages
BankAccountID BankName TypeOfAccount RountingNumber AccountNumber
1 400006 AAA Saving 19486122 109484623
2 400007 GGG Saving 19486121 109487623
3 400008 TTT cheking 19486115 109484699
4 400009 AAA Saving 59486122 104424623
5 400010 TTT Saving 19486135 109400823
Query executed successfully.
DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 5 rows
Ready
Ln 1 Col 33 Ch 33 INS
ENG
15:51
2019/4/14

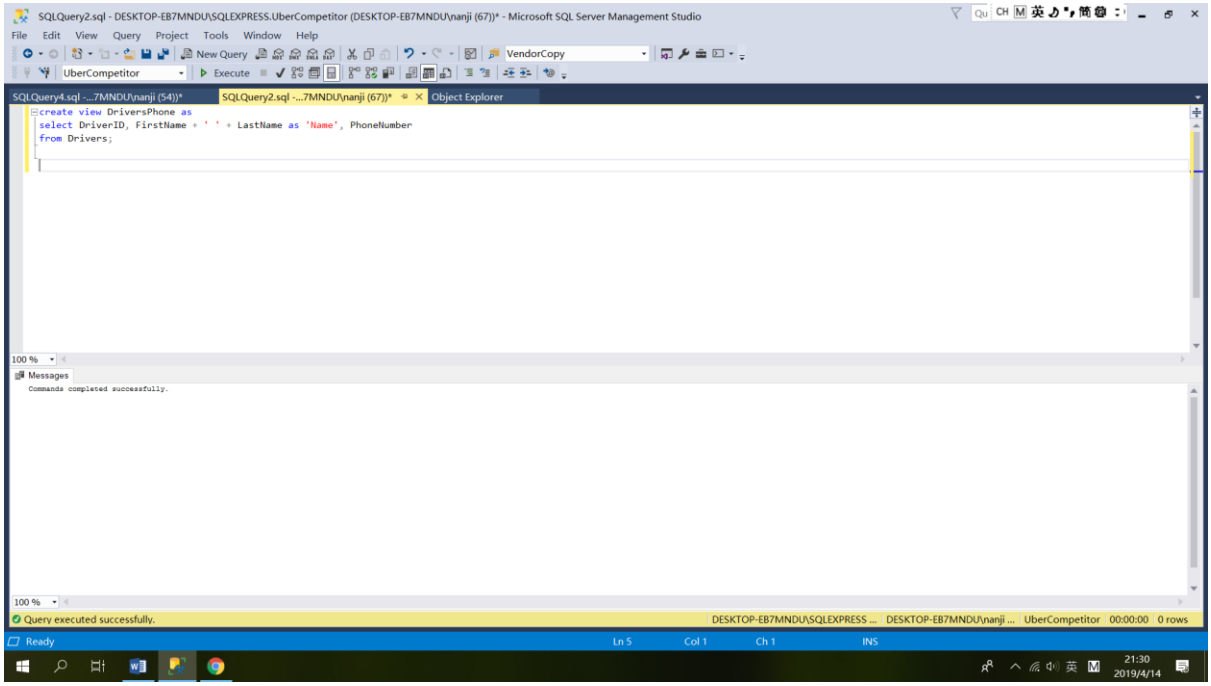
Cars:

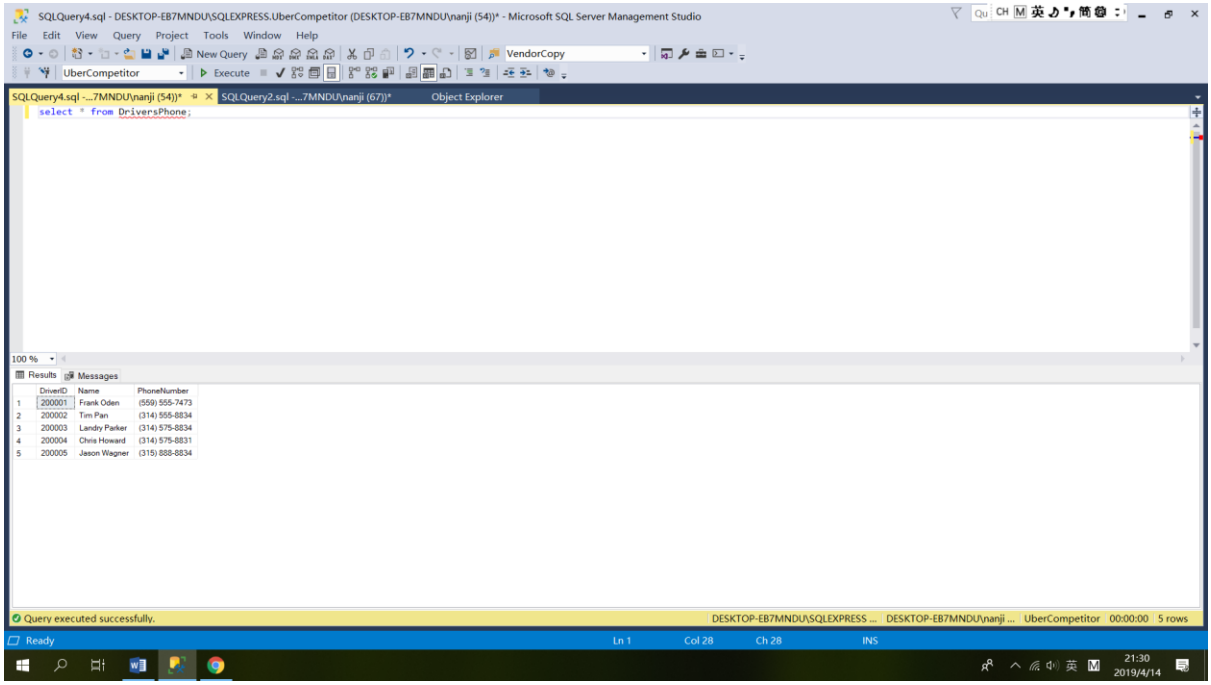

SQLQuery1.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (53))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
VendorCopy
UberCompetitor
Execute
SQLQuery1.sql -...7MNDU\nanji (53))*
Object Explorer
select * from Cars
100 %
Results
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.
DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 5 rows
Ready
Ln 1 Col 19 Ch 19 INS
ENG
15:51
2019/4/14

Part IV: Testing

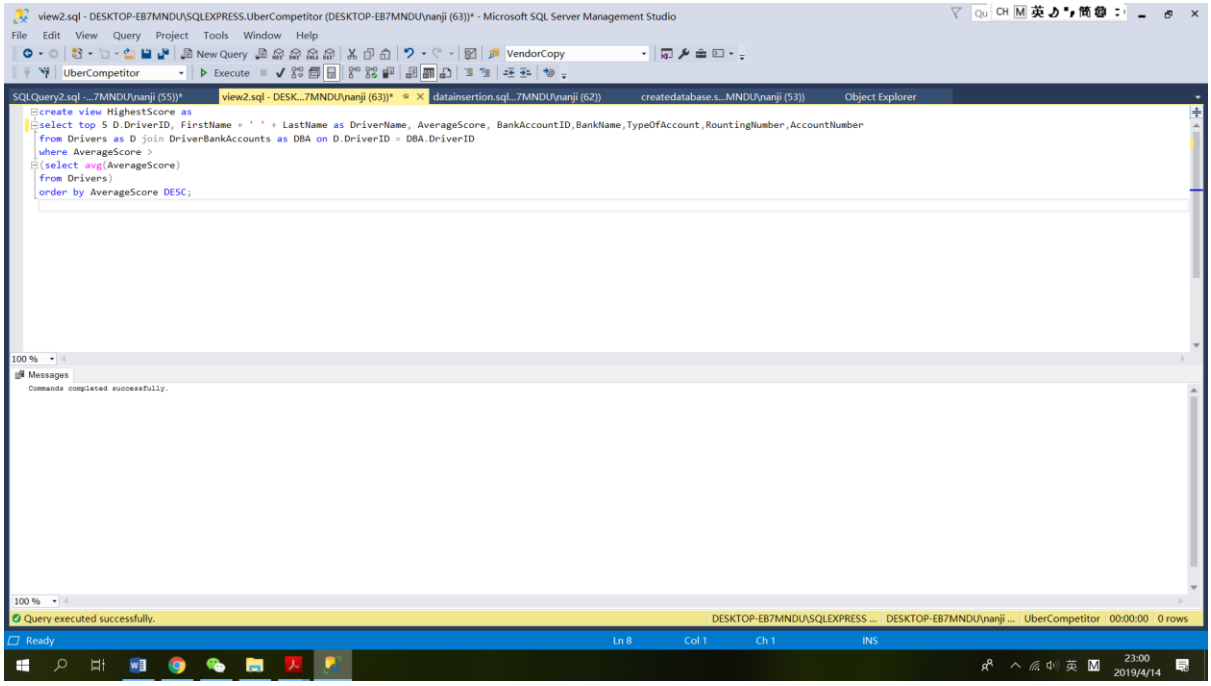
A. Views

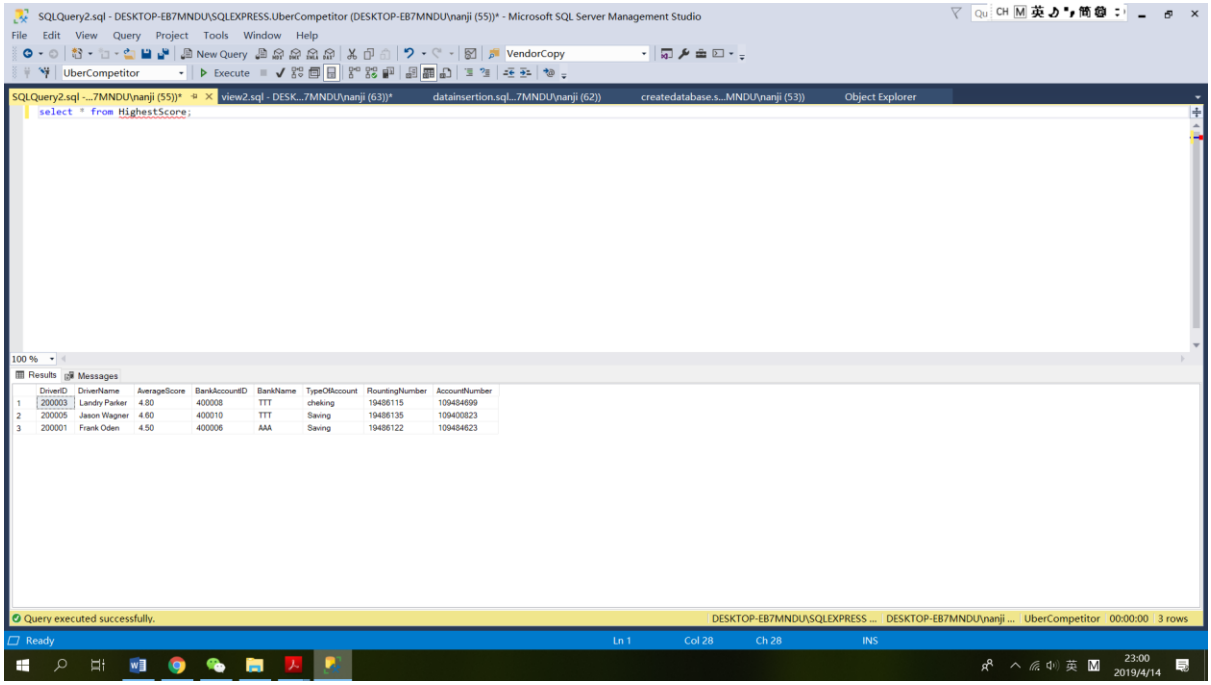
(1) View drivers' phone numbers.



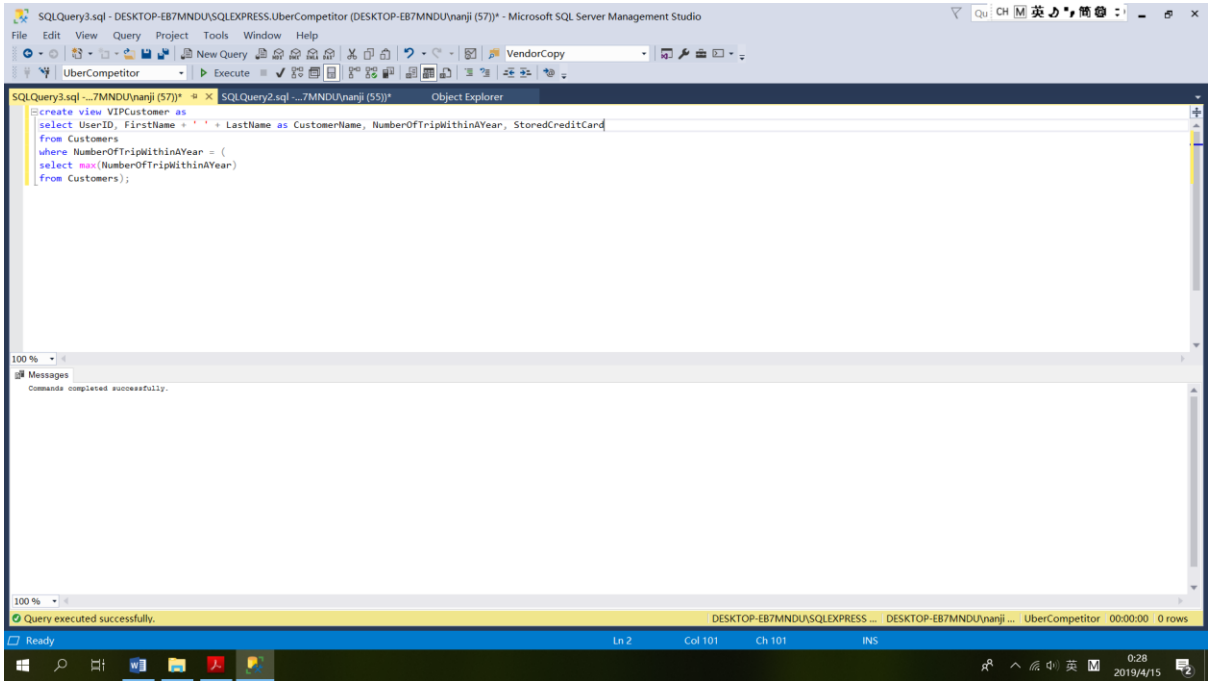


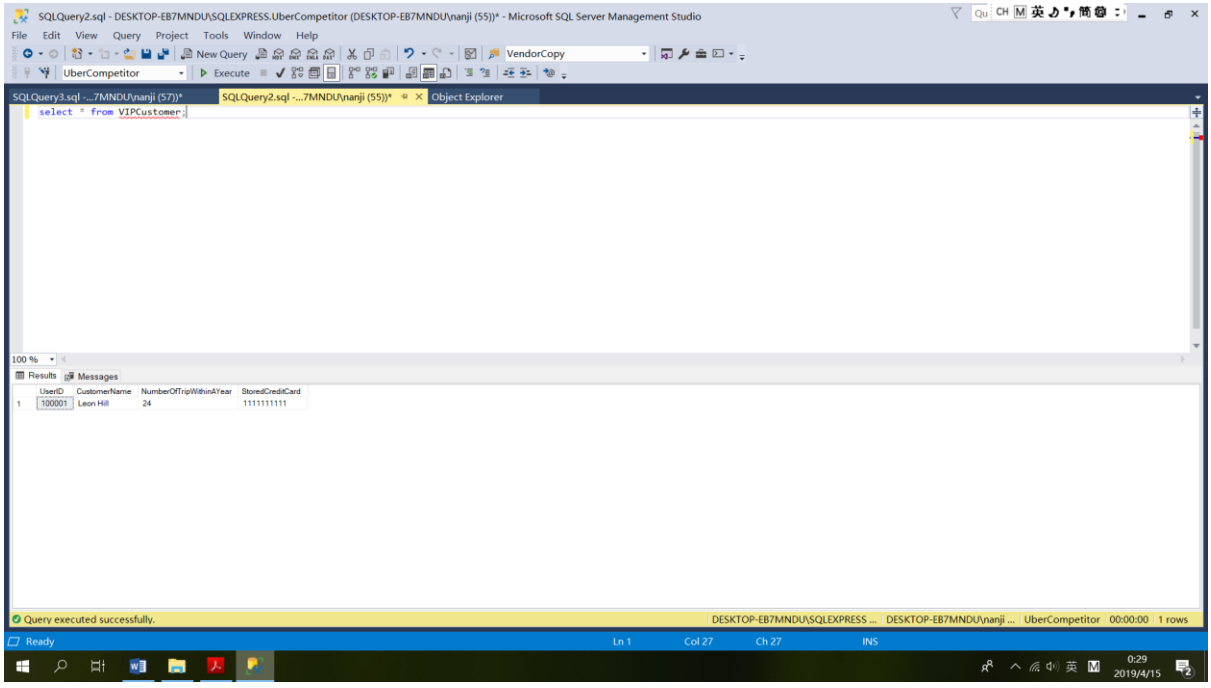
(2) view all the drivers whose score is larger than the average. Return their bank accounts also.



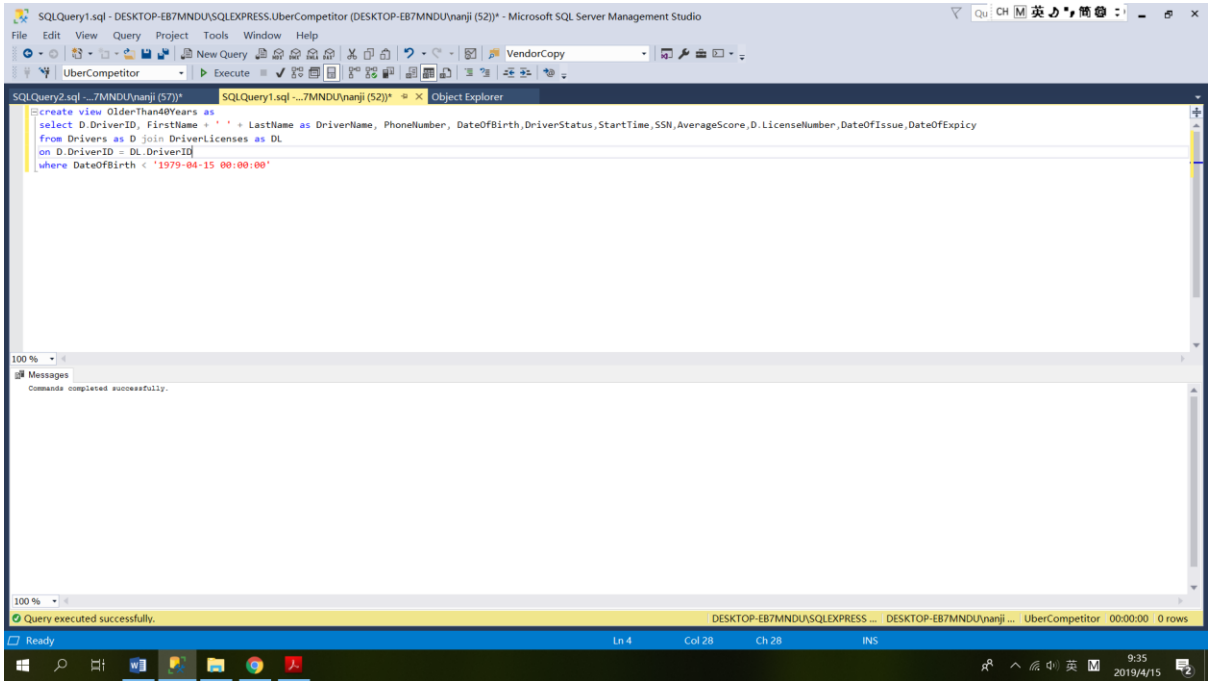


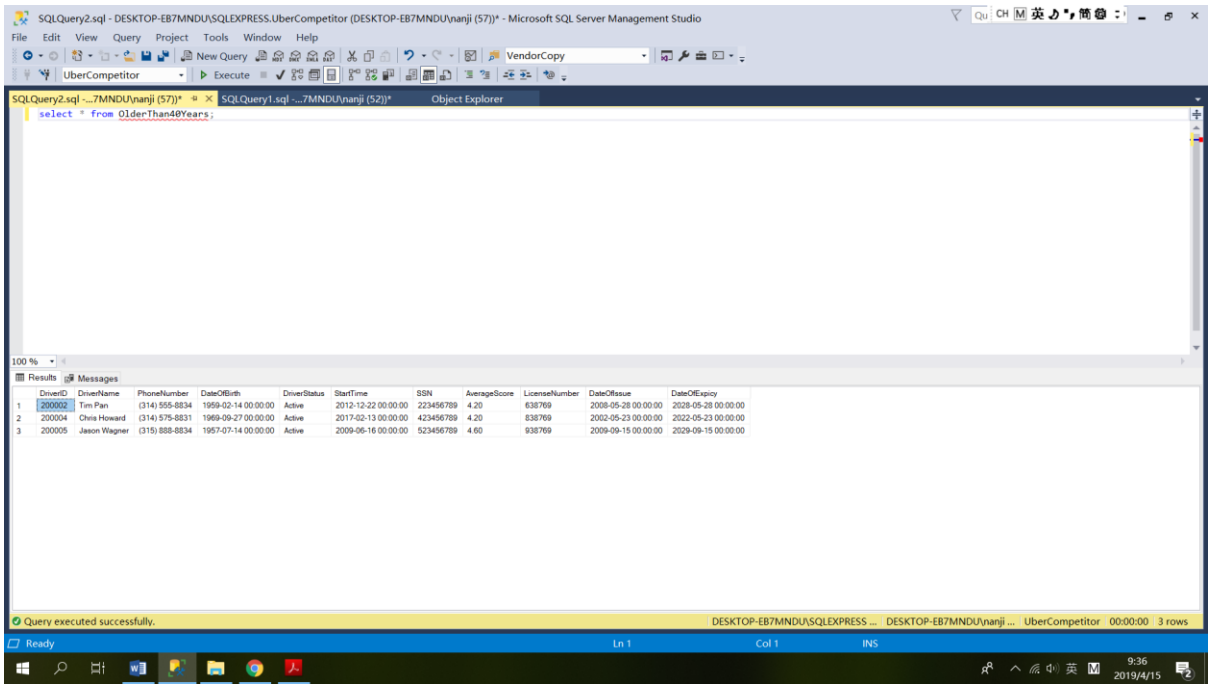
(3) view the customer who has taken the most trips within a year.





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

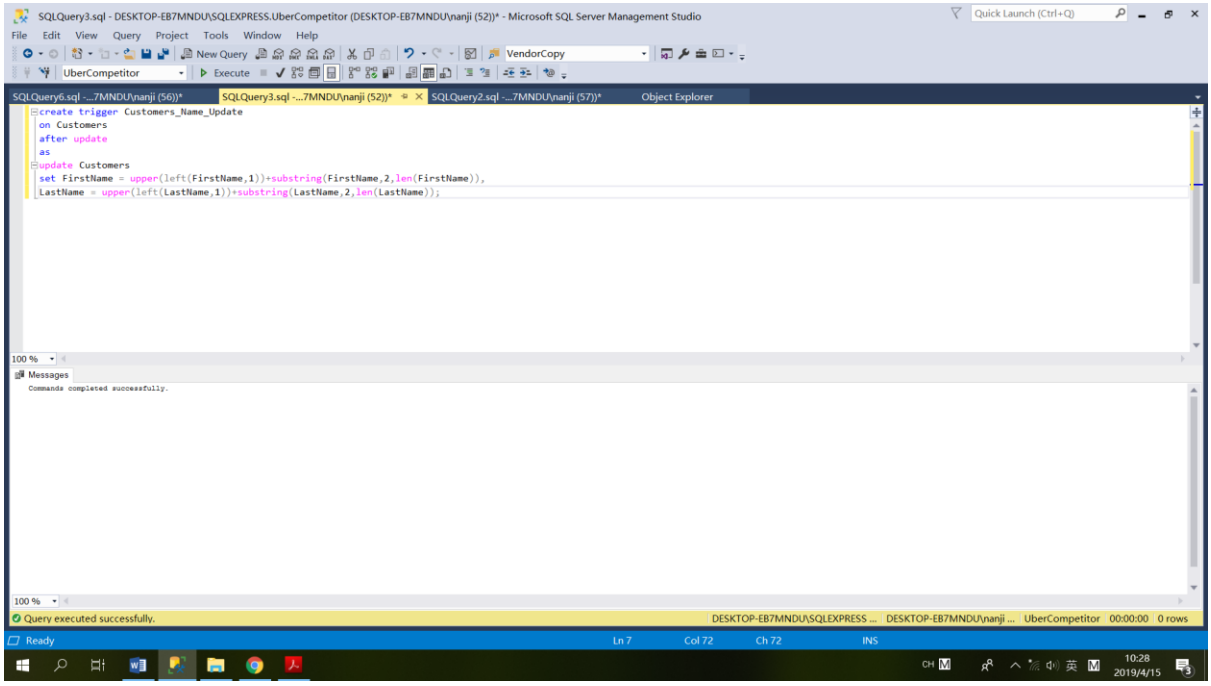


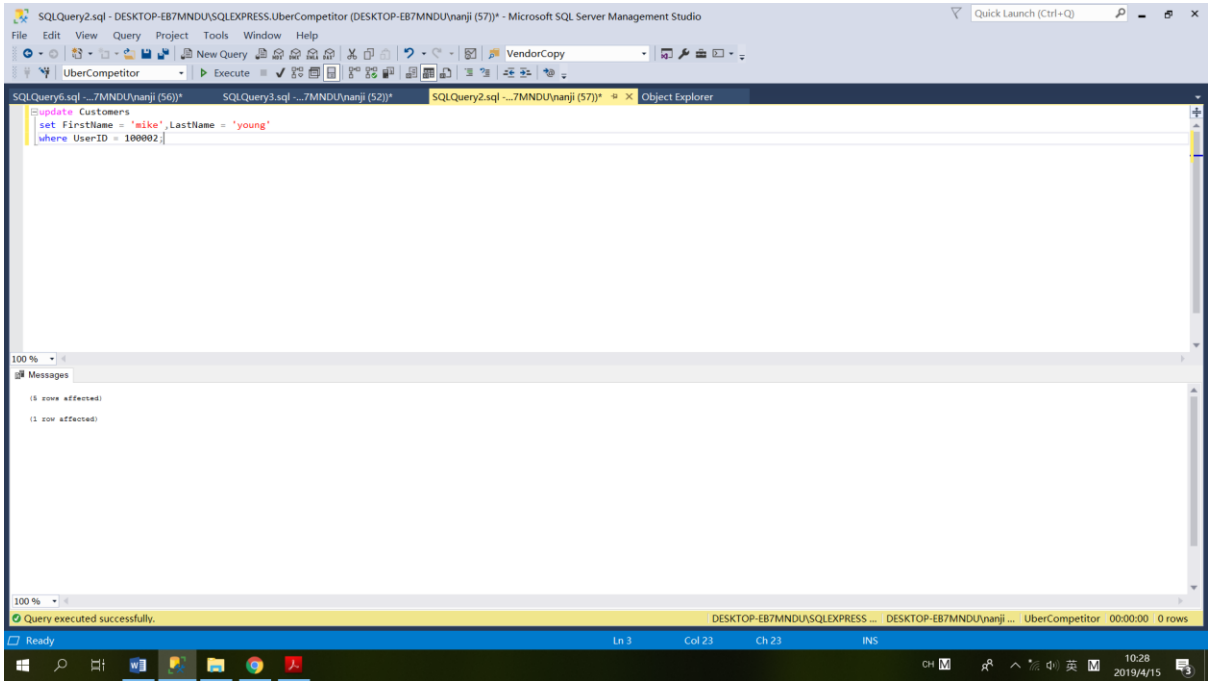


B. Triggers

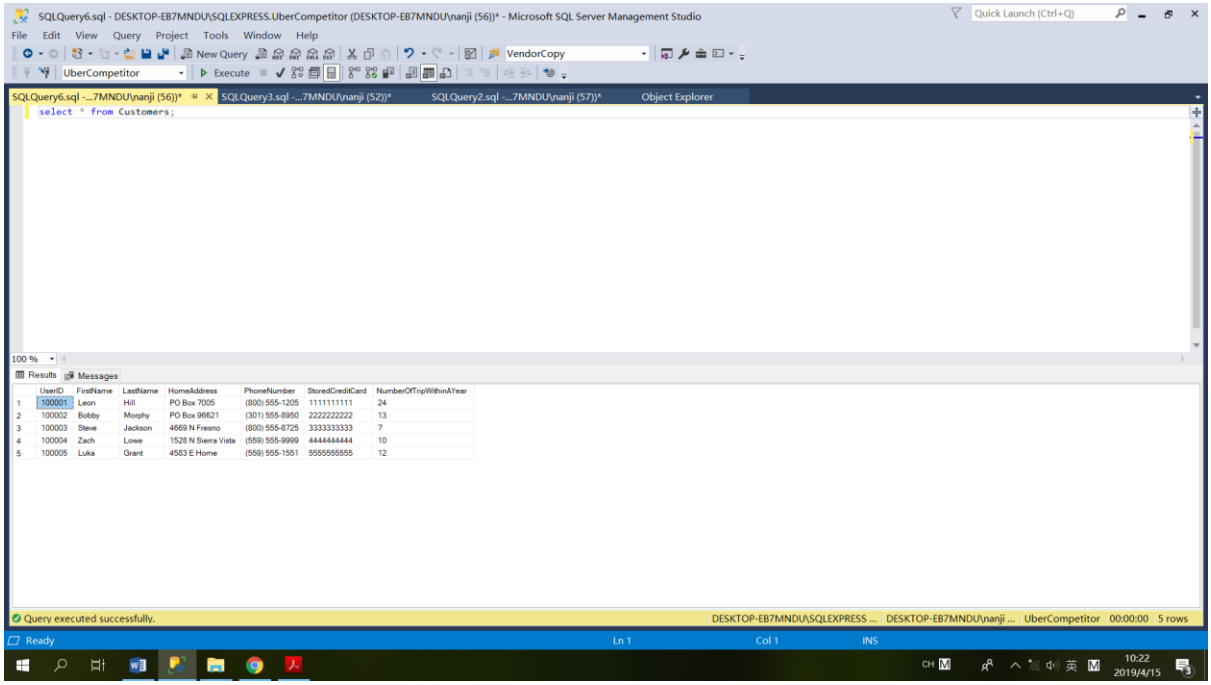
(1) Someone wants to change the customer's name from 'Bobby Murphy' to 'Mike Young' whose UseID is 100002. But by mistake the characters that he typed in

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

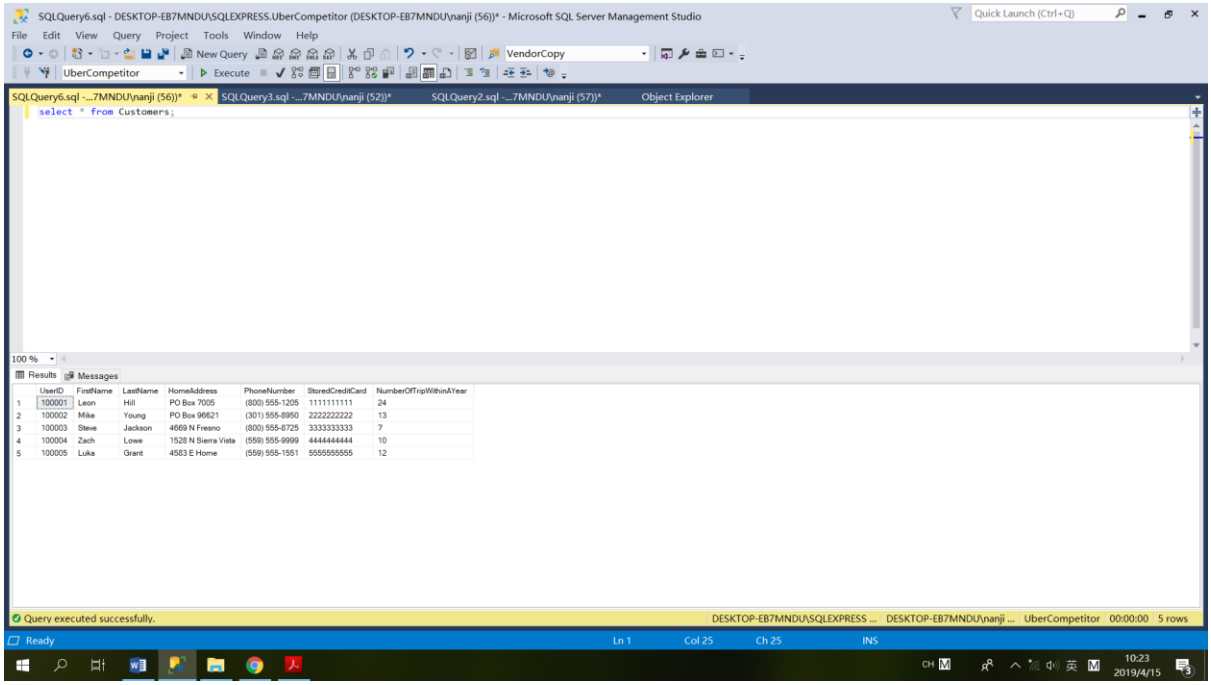




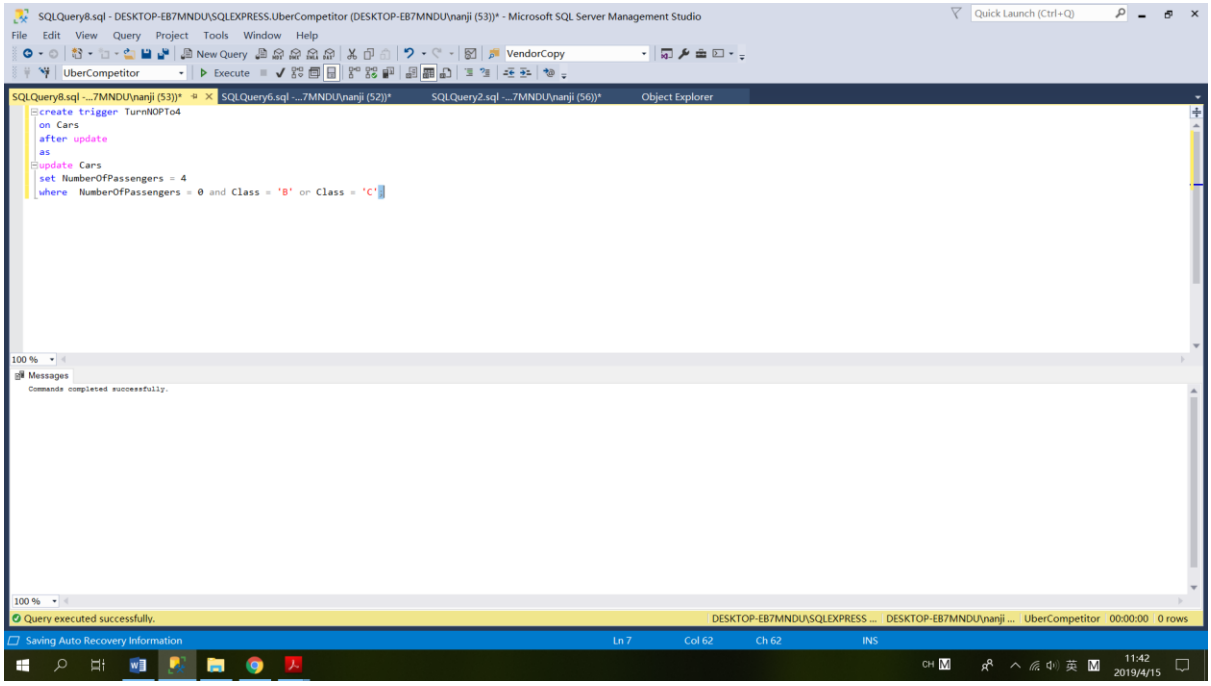
Before:

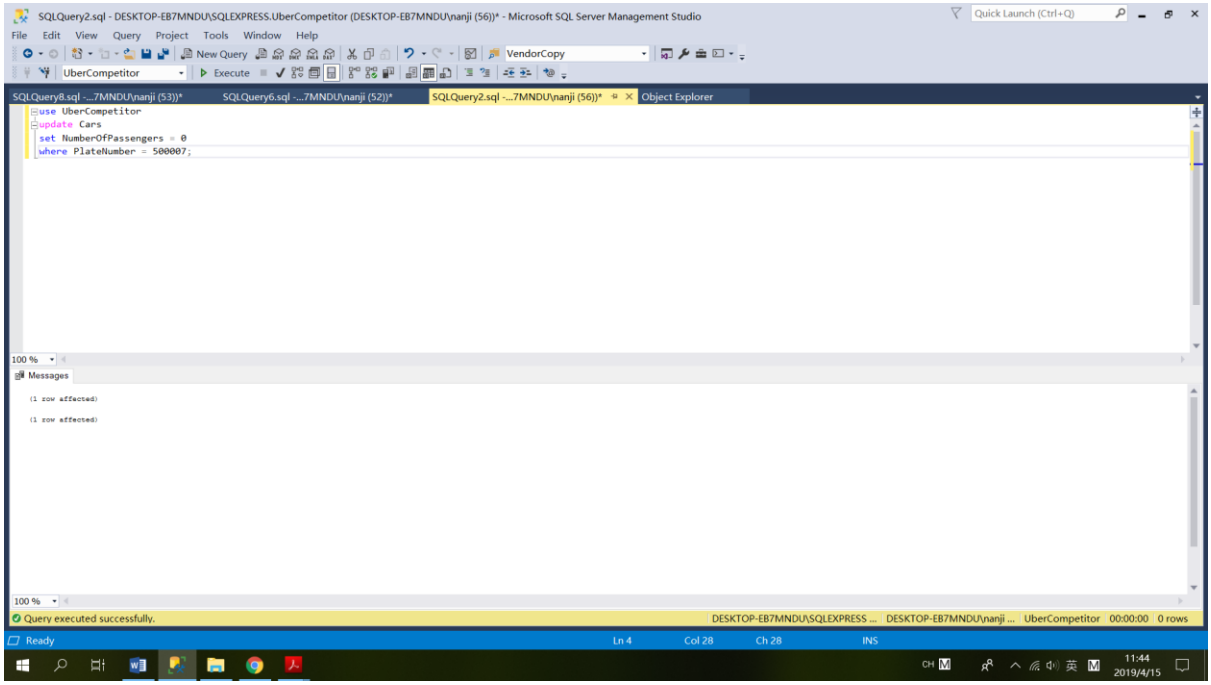

SQLQuery6.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (56))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
VendorCopy
UberCompetitor
Execute
SQLQuery6.sql -...7MNDU\nanji (56))*
SQLQuery3.sql -...7MNDU\nanji (52))*
SQLQuery2.sql -...7MNDU\nanji (57))*
Object Explorer
select * from Customers;
100 %
Results
Messages
UserID FirstName LastName HomeAddress PhoneNumber StoredCreditCard NumberOfTripWithinAYear
1 100001 Leon Hill PO Box 7005 (800) 555-1205 1111111111 24
2 100002 Bobby Morphy PO Box 96621 (301) 555-8950 2222222222 13
3 100003 Steve Jackson 4669 N Fresno (800) 555-8725 3333333333 7
4 100004 Zach Lowe 1528 N Sierra Vista (559) 555-9999 4444444444 10
5 100005 Luke Grant 4583 E Home (559) 555-1551 5555555555 12
Query executed successfully.
DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 5 rows
Ready
Ln 1 Col 1 INS
10:22
2019/4/15

After:

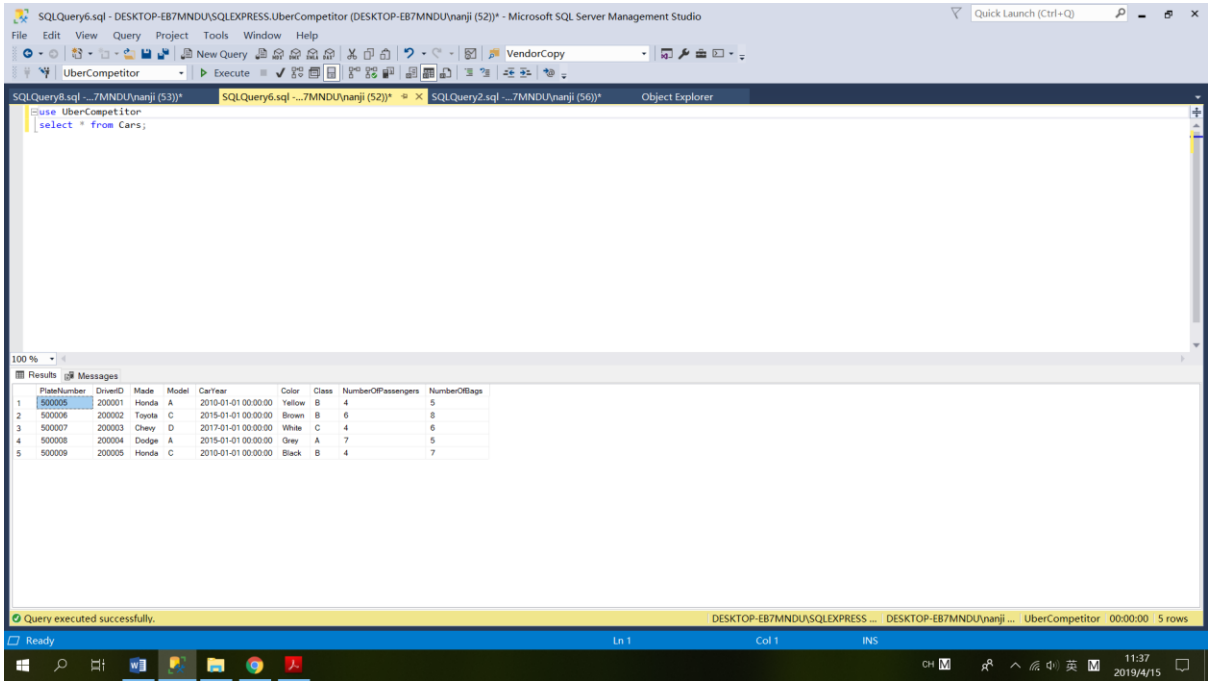

SQLQuery6.sql - DESKTOP-EB7MNDU\SQLEXPRESS.UberCompetitor (DESKTOP-EB7MNDU\nanji (56))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
VendorCopy
UberCompetitor
Execute
SQLQuery6.sql -...7MNDU\nanji (56))*
SQLQuery3.sql -...7MNDU\nanji (52))*
SQLQuery2.sql -...7MNDU\nanji (57))*
Object Explorer
select * from Customers;
100 %
Results
Messages
UserID FirstName LastName HomeAddress PhoneNumber StoredCreditCard NumberOfTripWithinAYear
1 100001 Leon Hill PO Box 7005 (800) 555-1205 1111111111 24
2 100002 Mike Young PO Box 96621 (301) 555-8950 2222222222 13
3 100003 Steve Jackson 4669 N Fresno (800) 555-8725 3333333333 7
4 100004 Zach Lowe 1528 N Sierra Vista (559) 555-9999 4444444444 10
5 100005 Luke Grant 4583 E Home (559) 555-1551 5555555555 12
Query executed successfully.
DESKTOP-EB7MNDU\SQLEXPRESS ... DESKTOP-EB7MNDU\nanji ... UberCompetitor 00:00:00 5 rows
Ready
Ln 1 Col 25 Ch 25 INS
10:23
2019/4/15

(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.

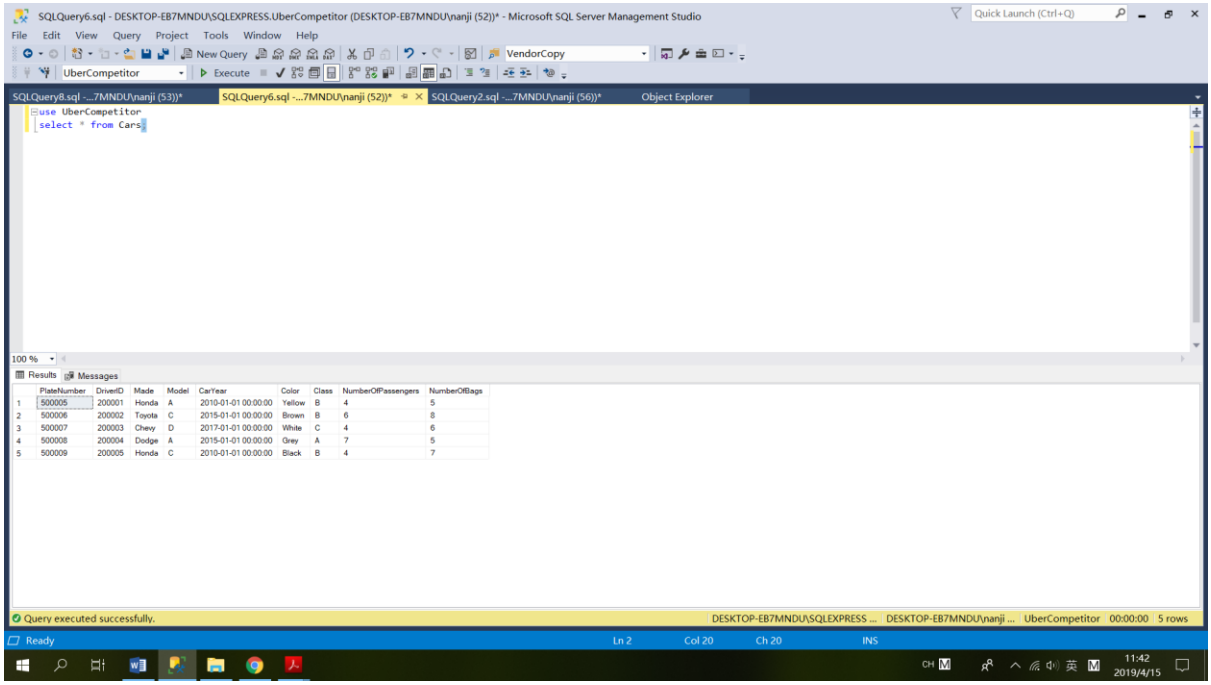




Before:



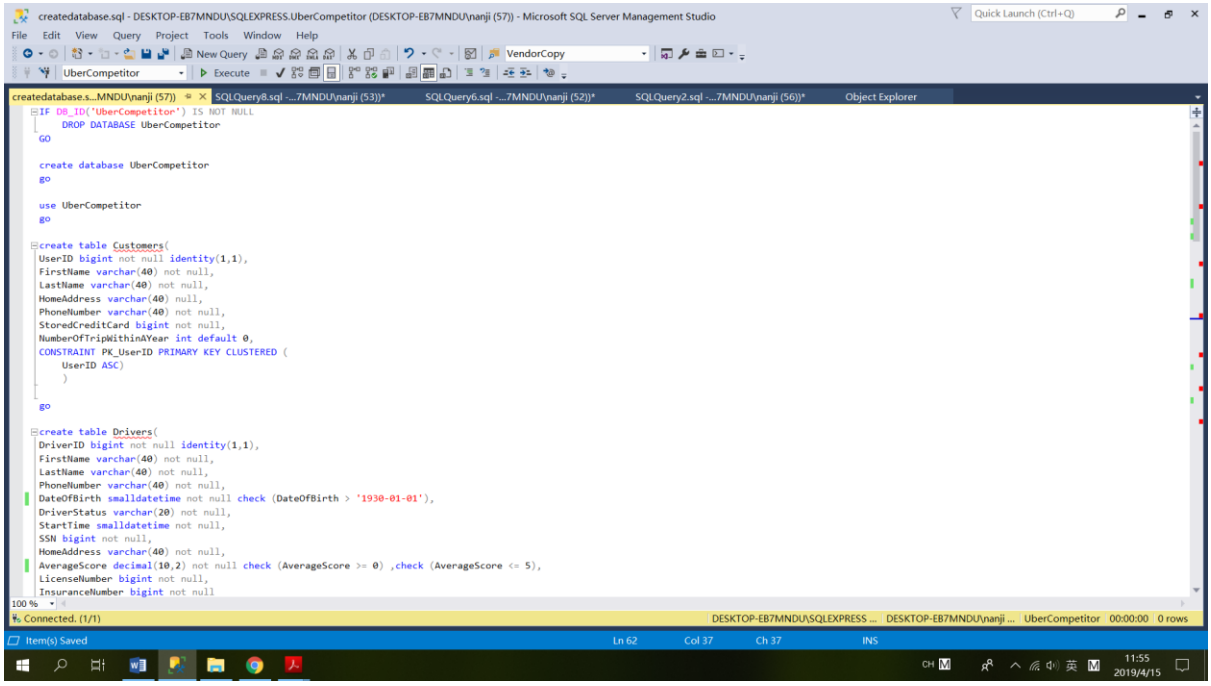
After:

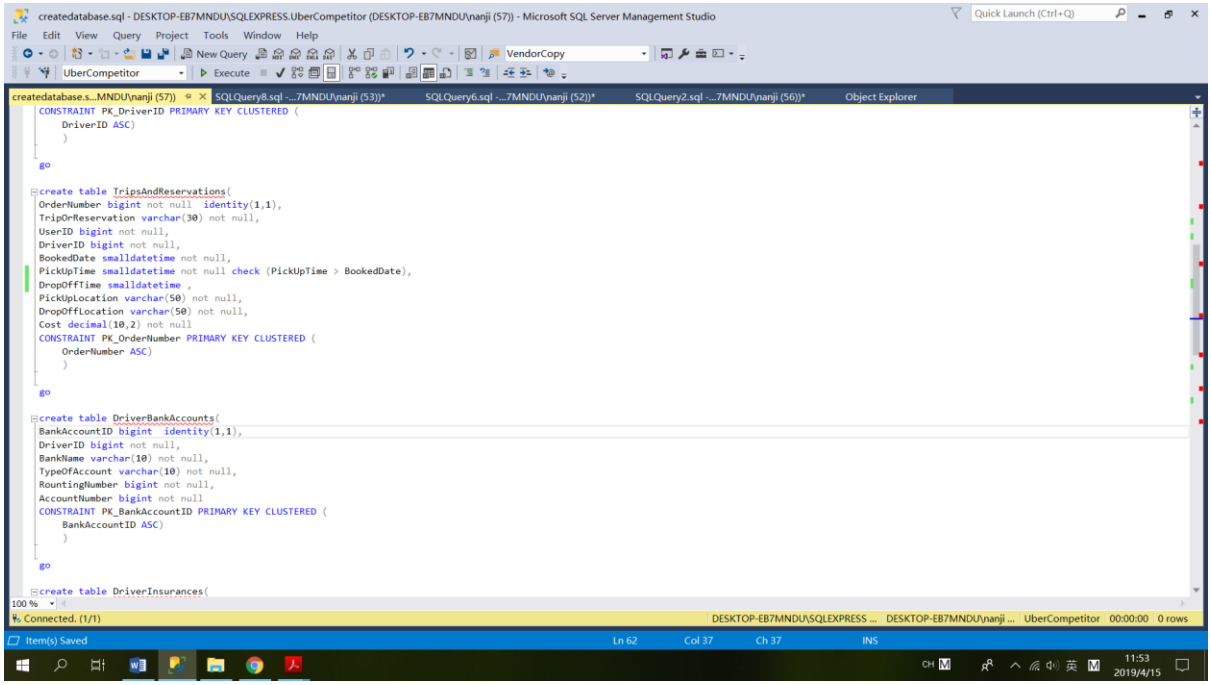


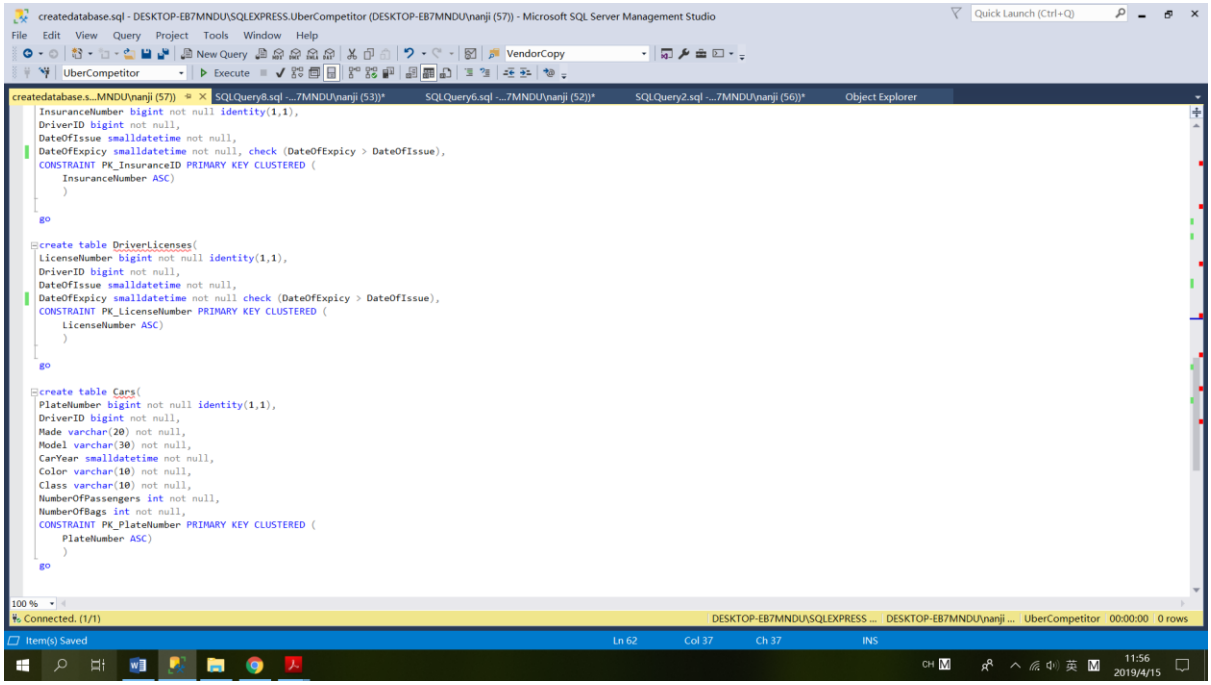
C. Constraints

Constraints are implemented when the database is implemented.

I use several checks and primary key constraints.

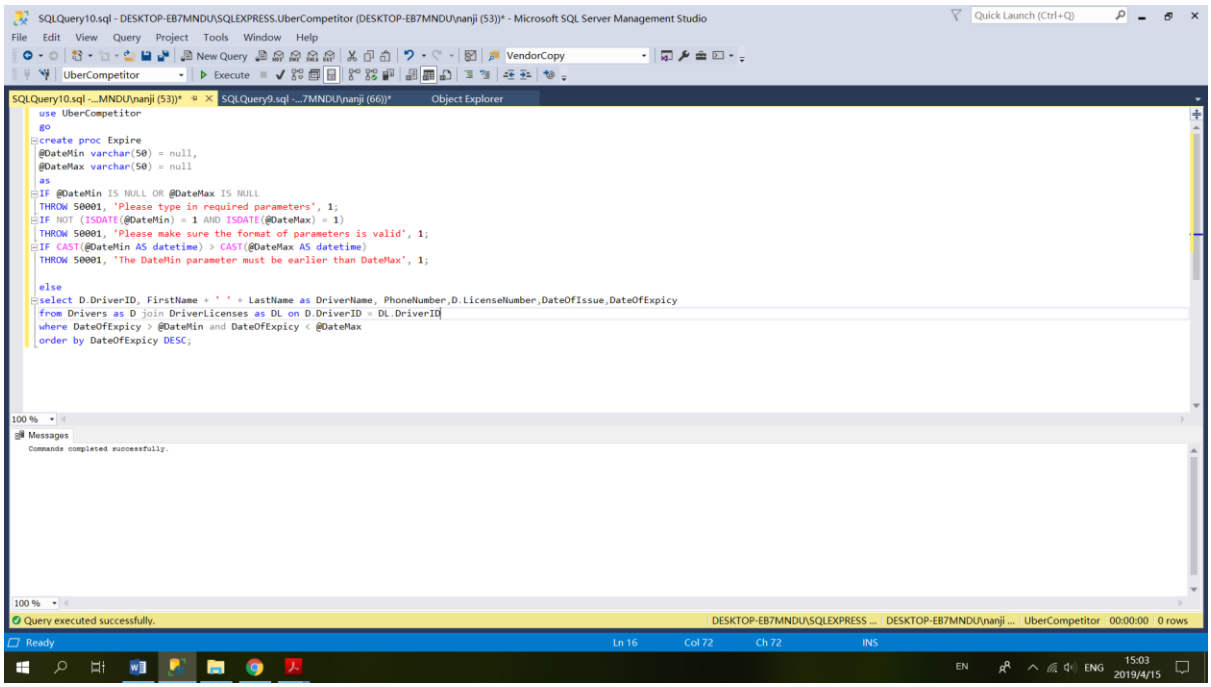




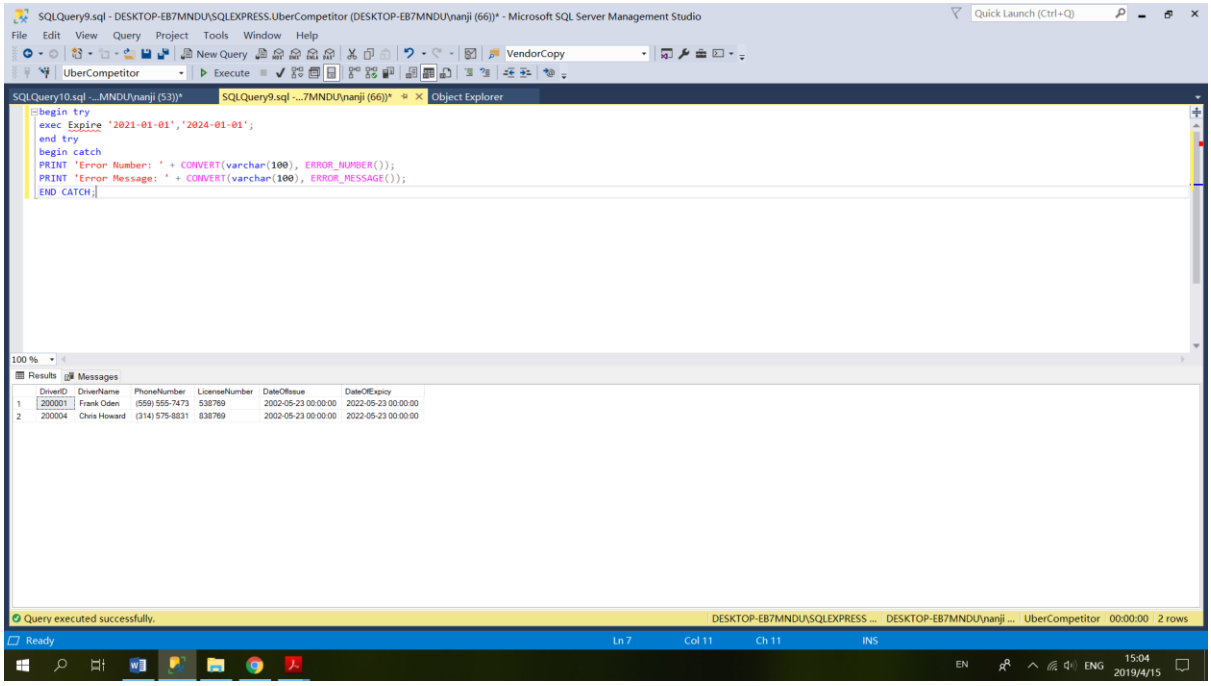


D. Stored Procedures

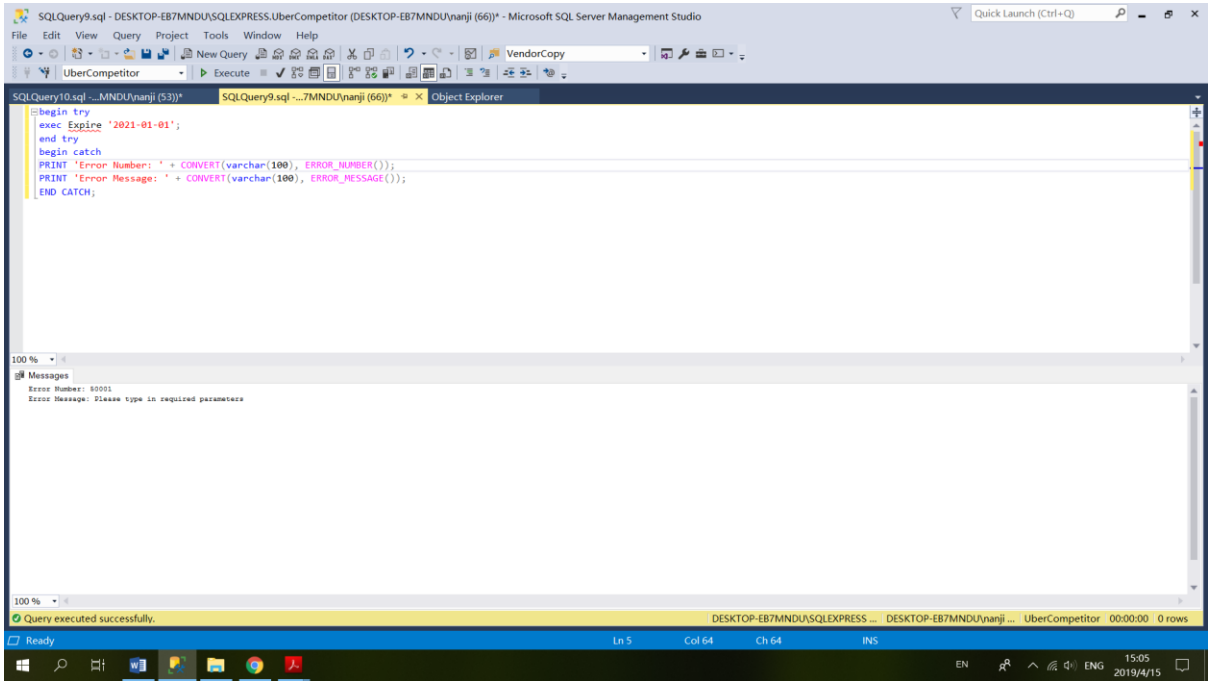
(1) This procedure is used to view some required information about drivers whose license is going to expire between two specific point-in-times.

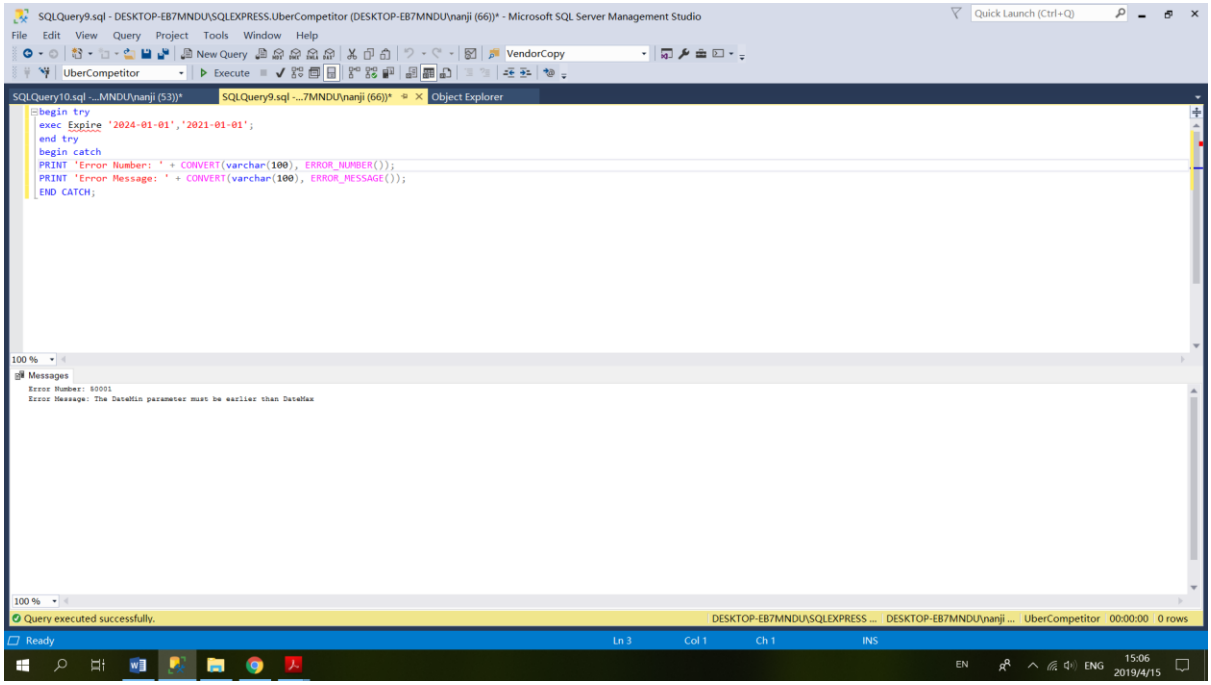


Test and result:

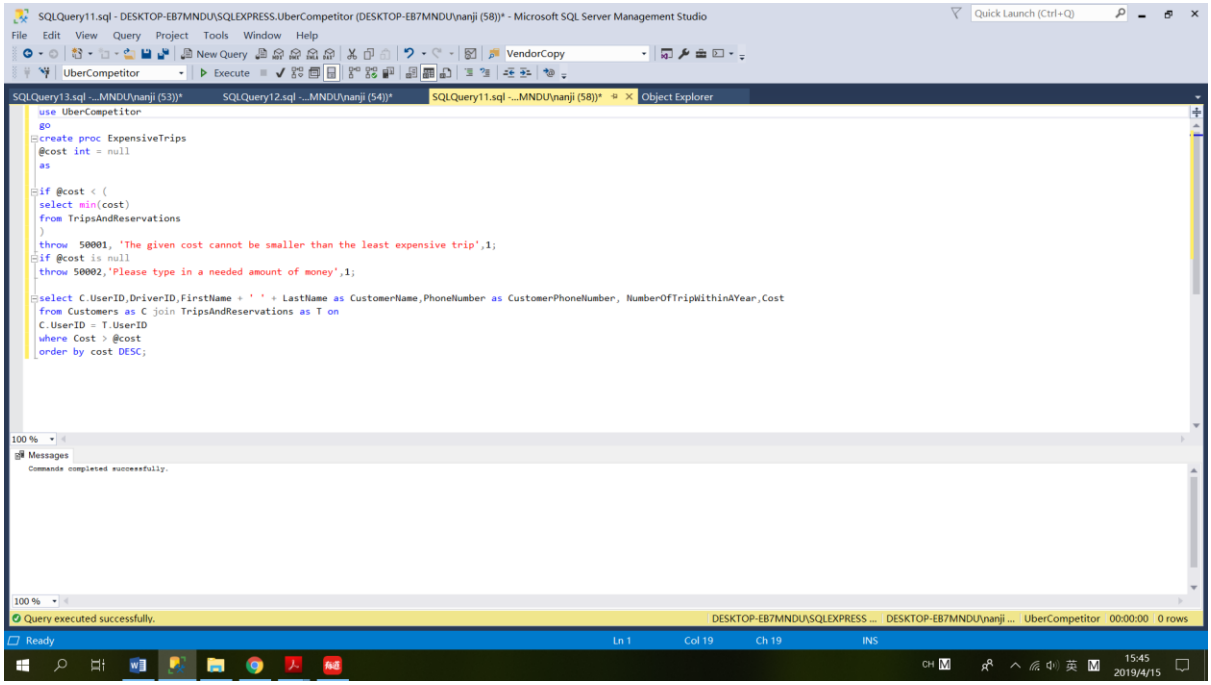


Error handling:

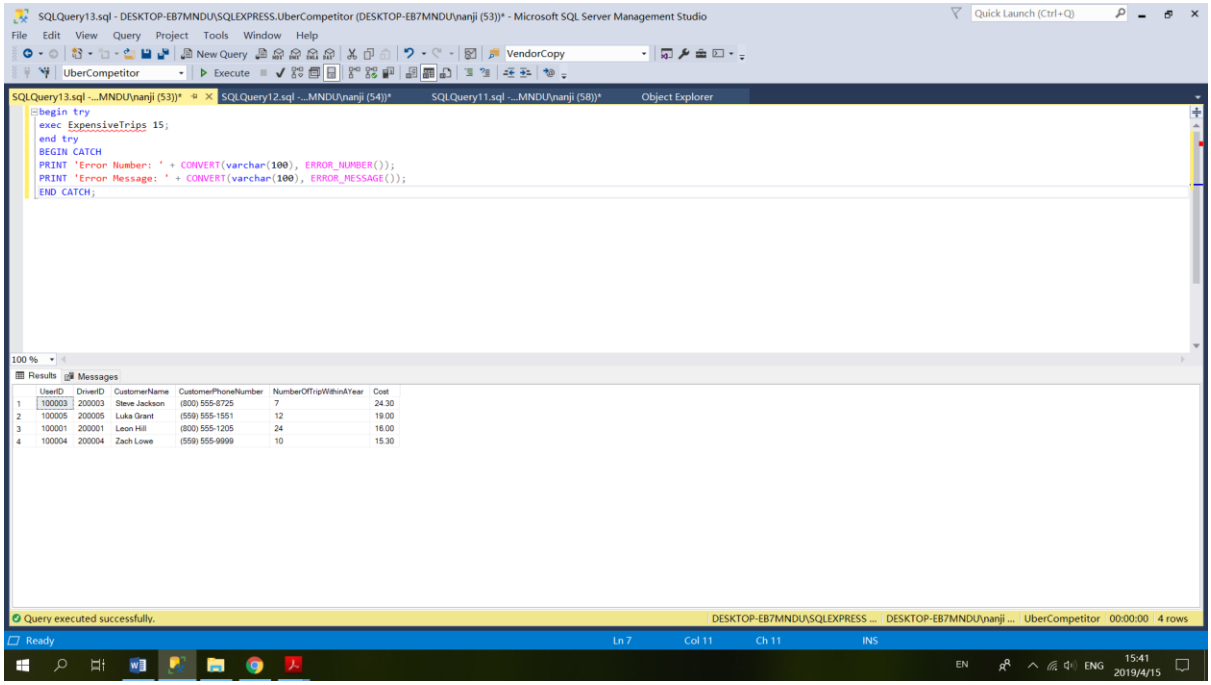




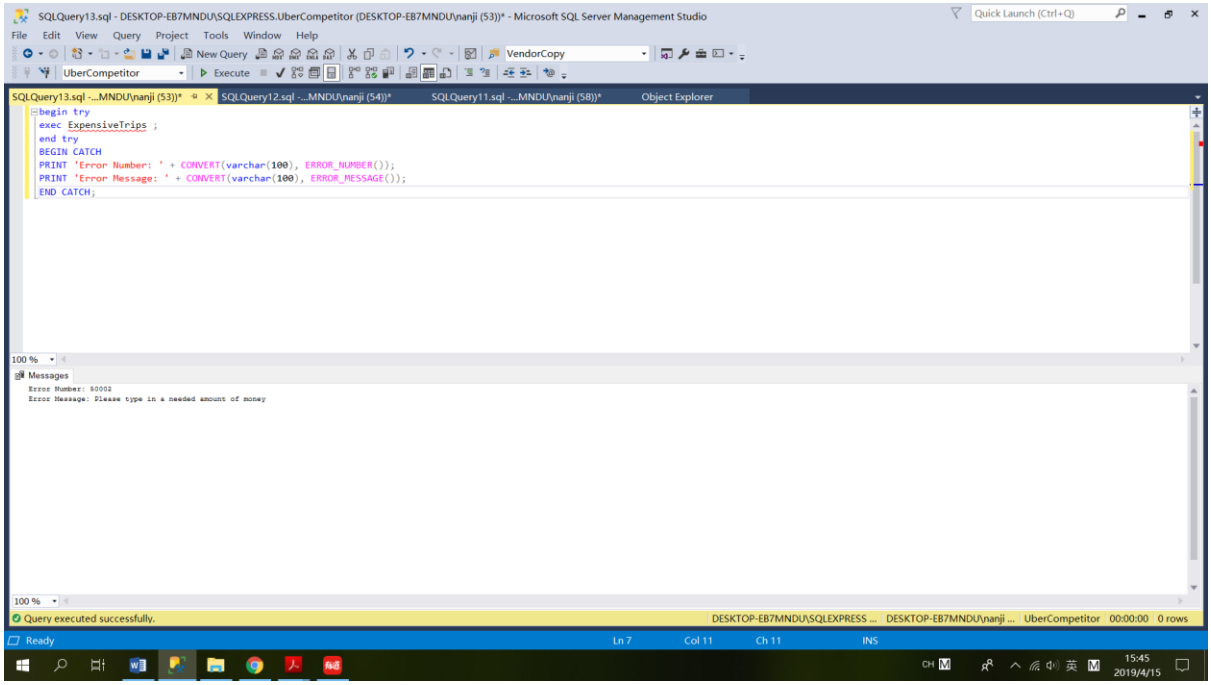
(2) This procedure is used to return information about customers whose cost on their trip is larger than a certain value.

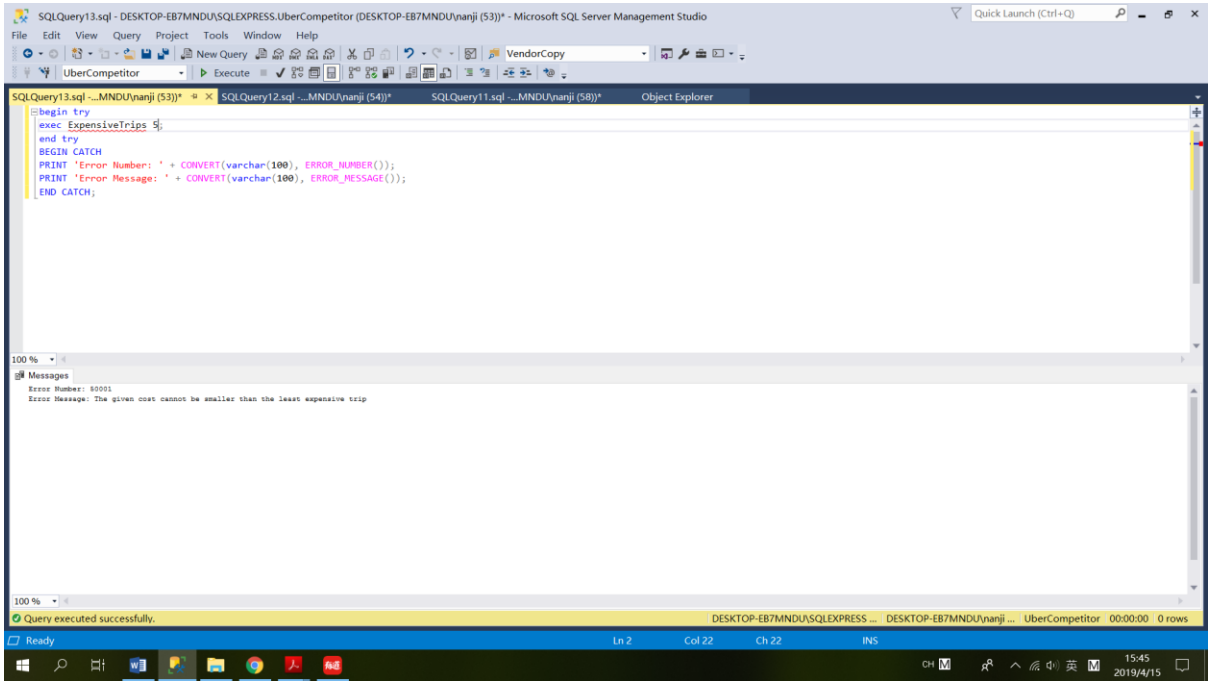


Test and result:



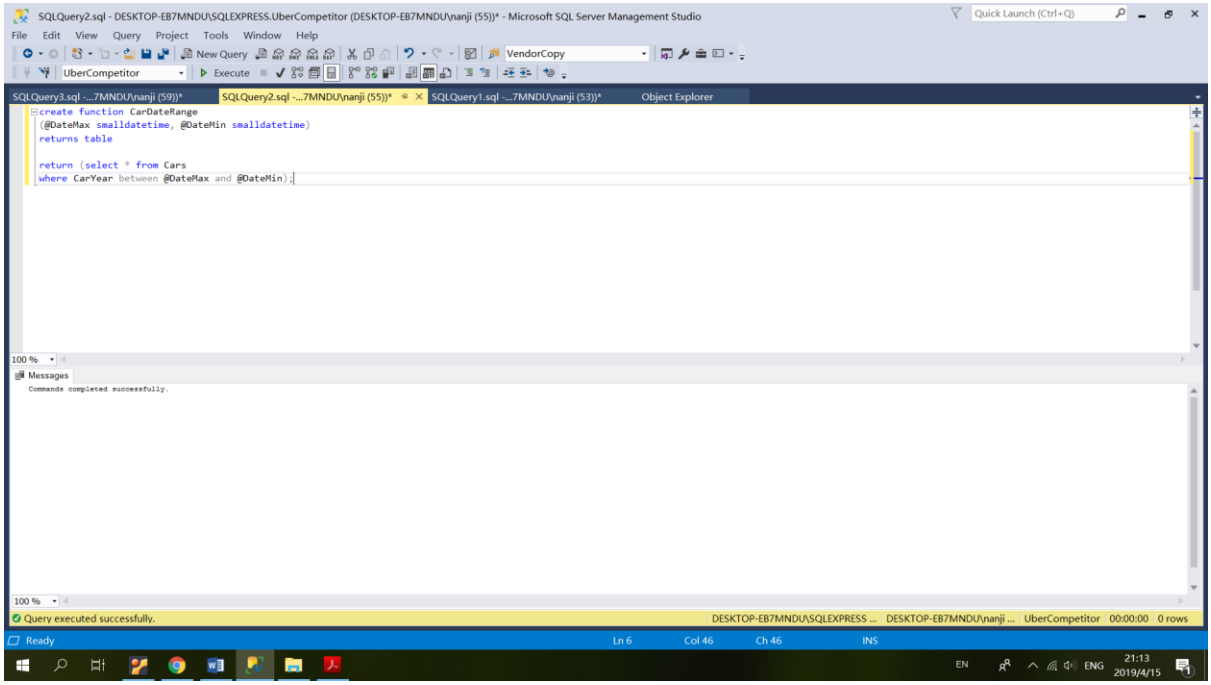
Error handling:



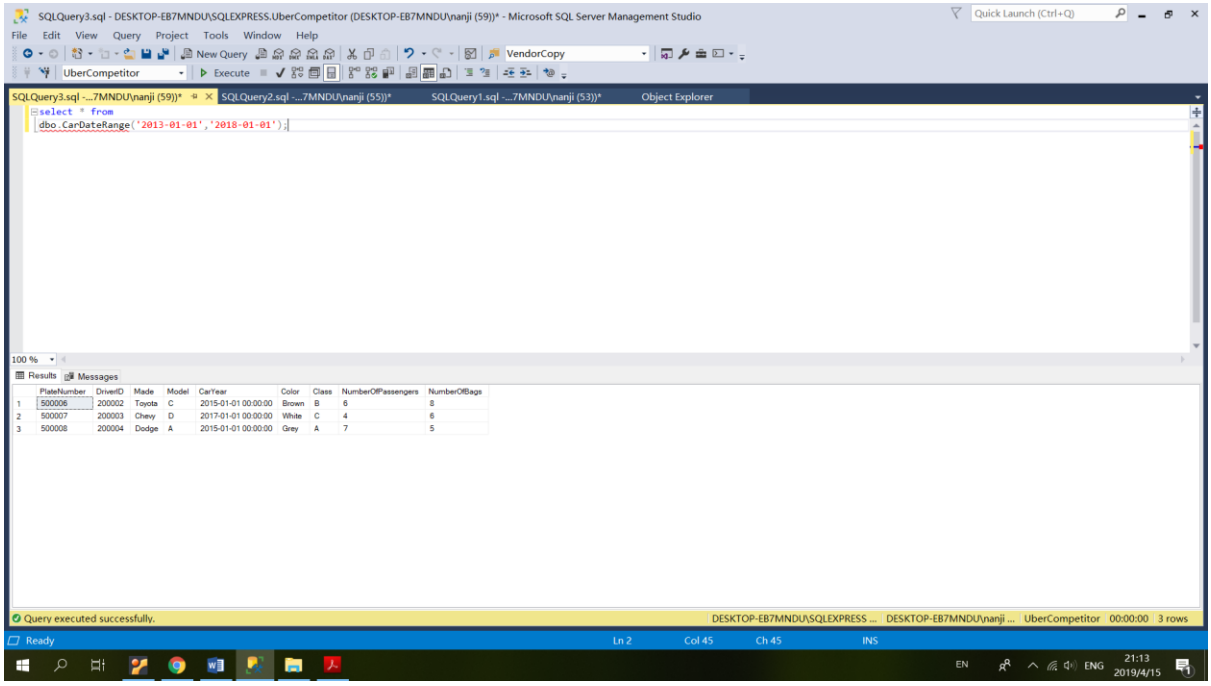


E. Functions

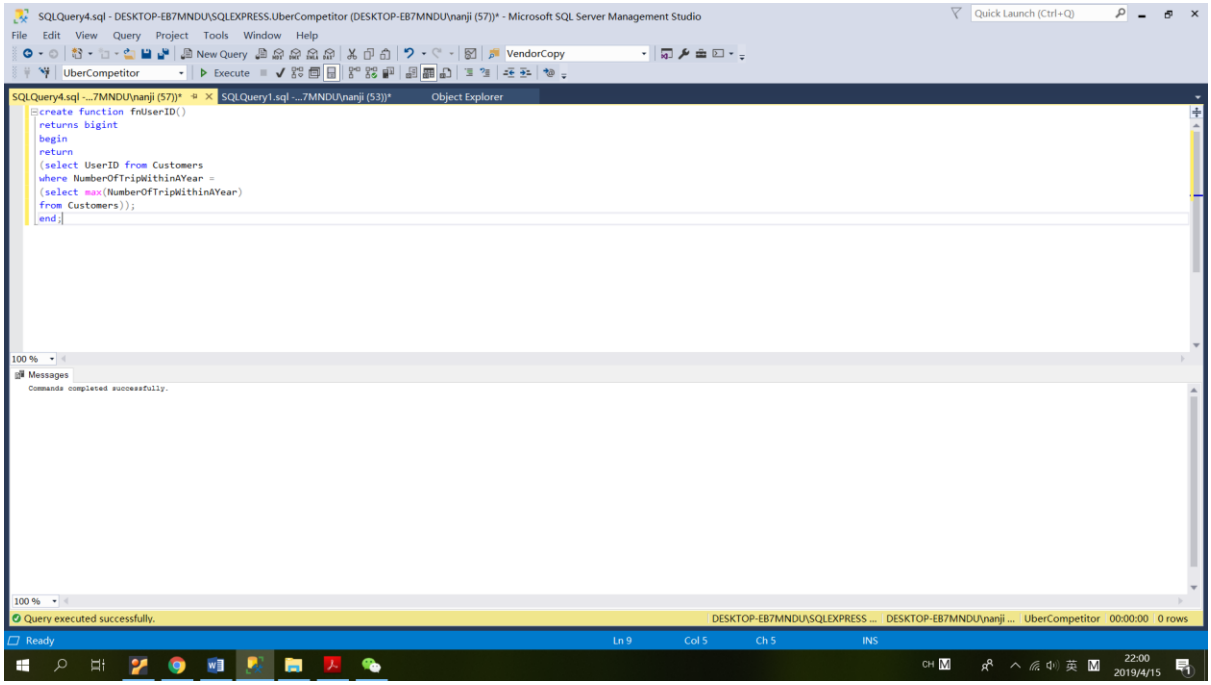
(1) Use a function called 'CarDateRange' to return the information about the cars produced within a period.



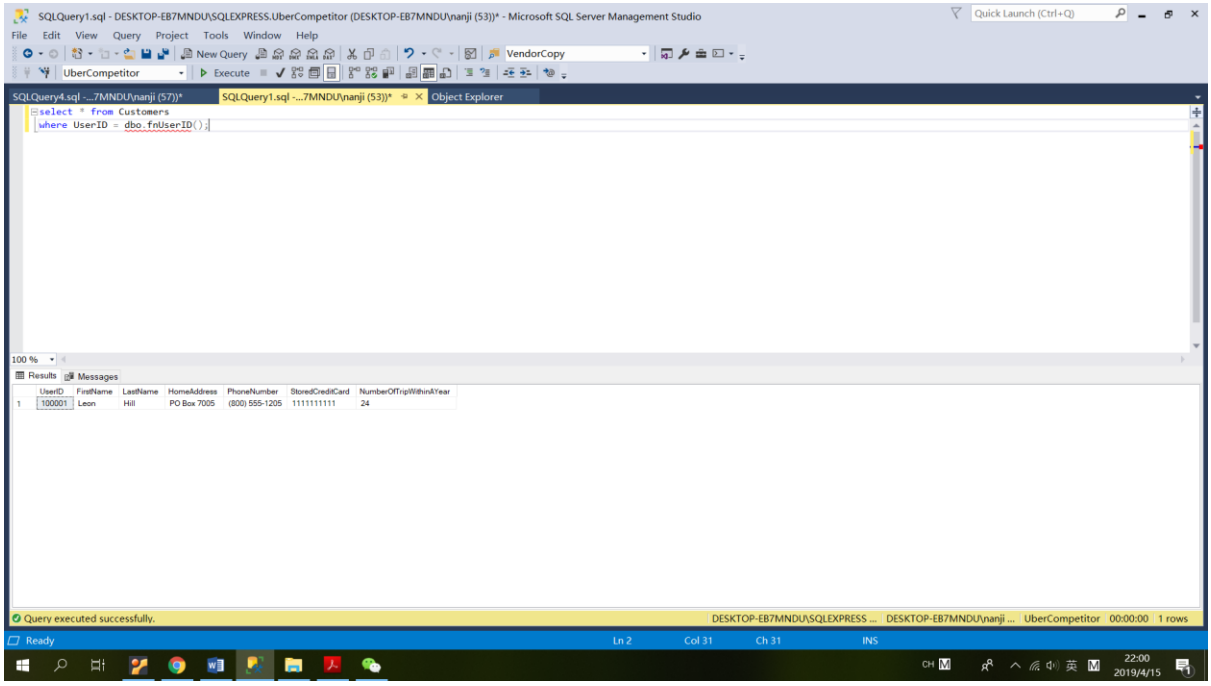
Test and result:



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

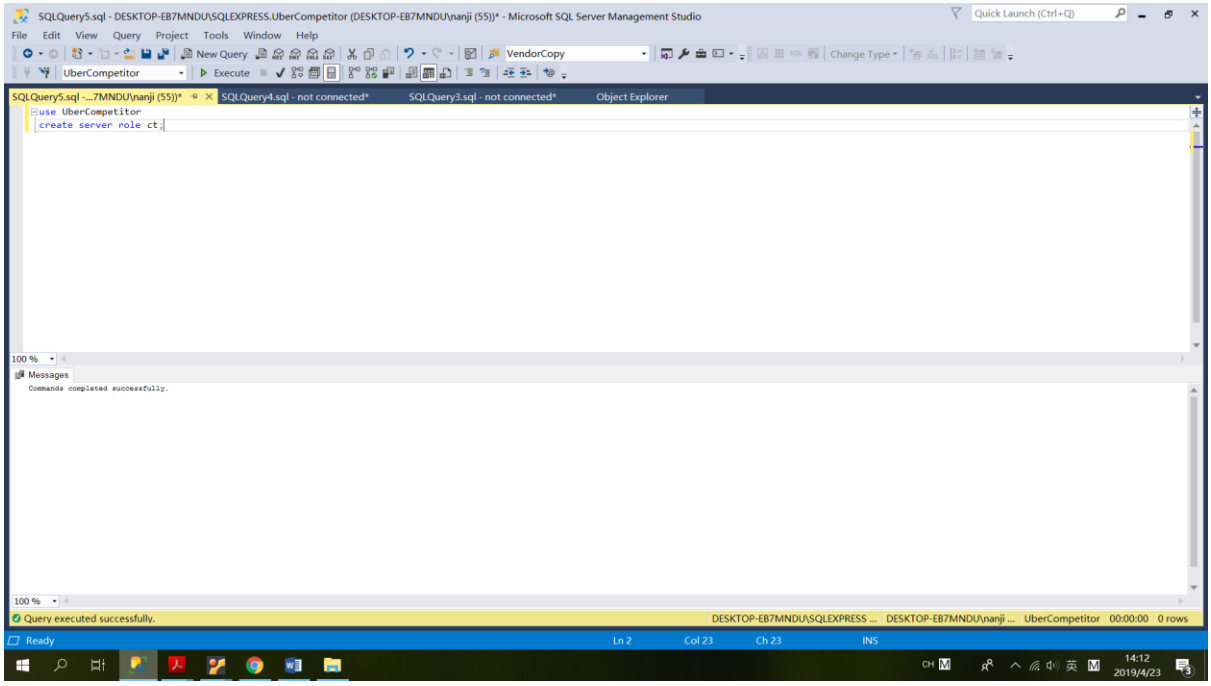


Test and results:



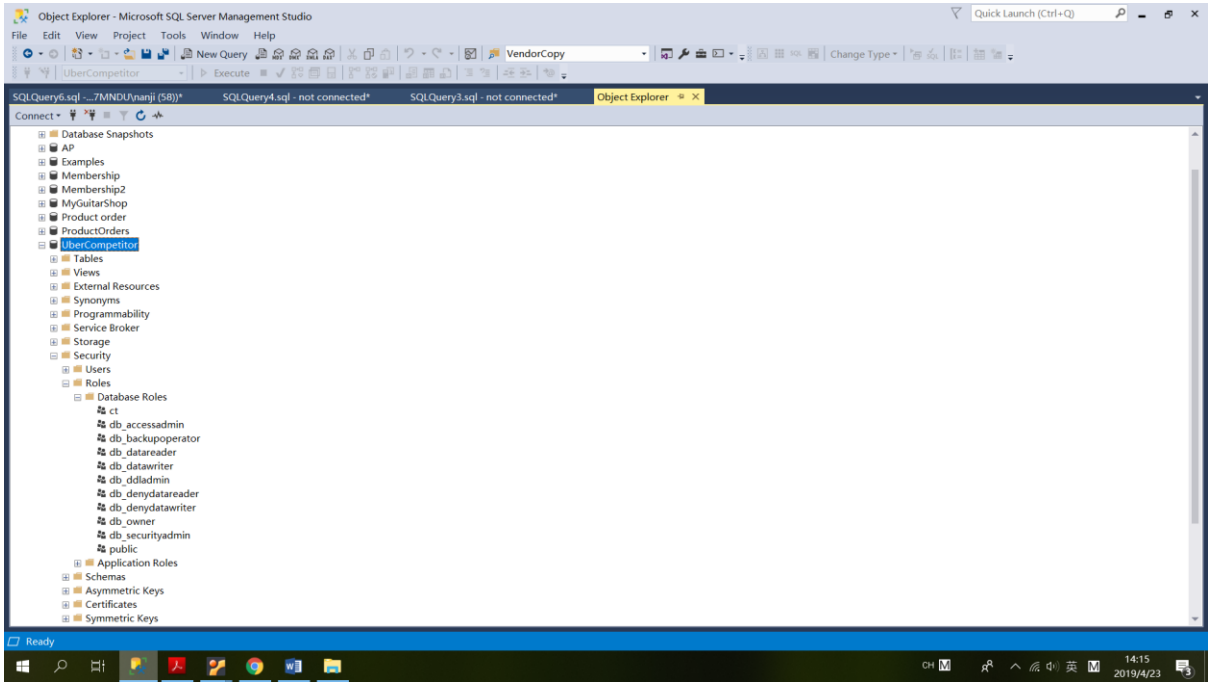
Part V: Security

A. Role

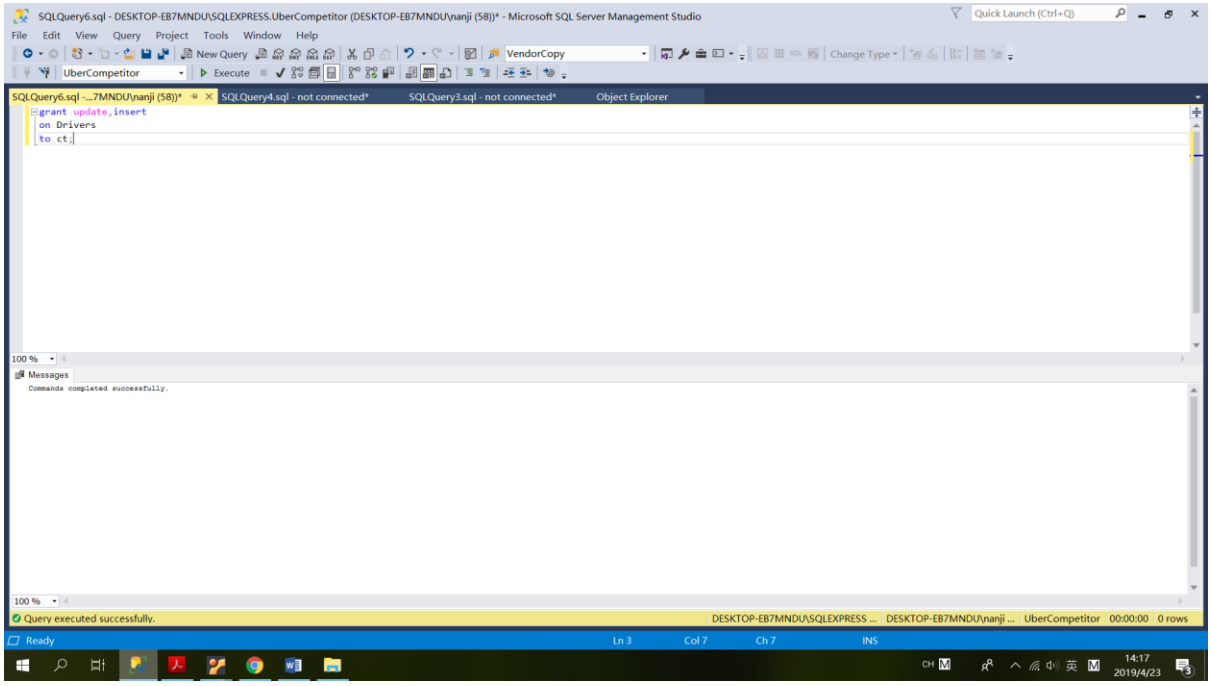


Create a server role called ‘ct’ in UberCompetitor databse.

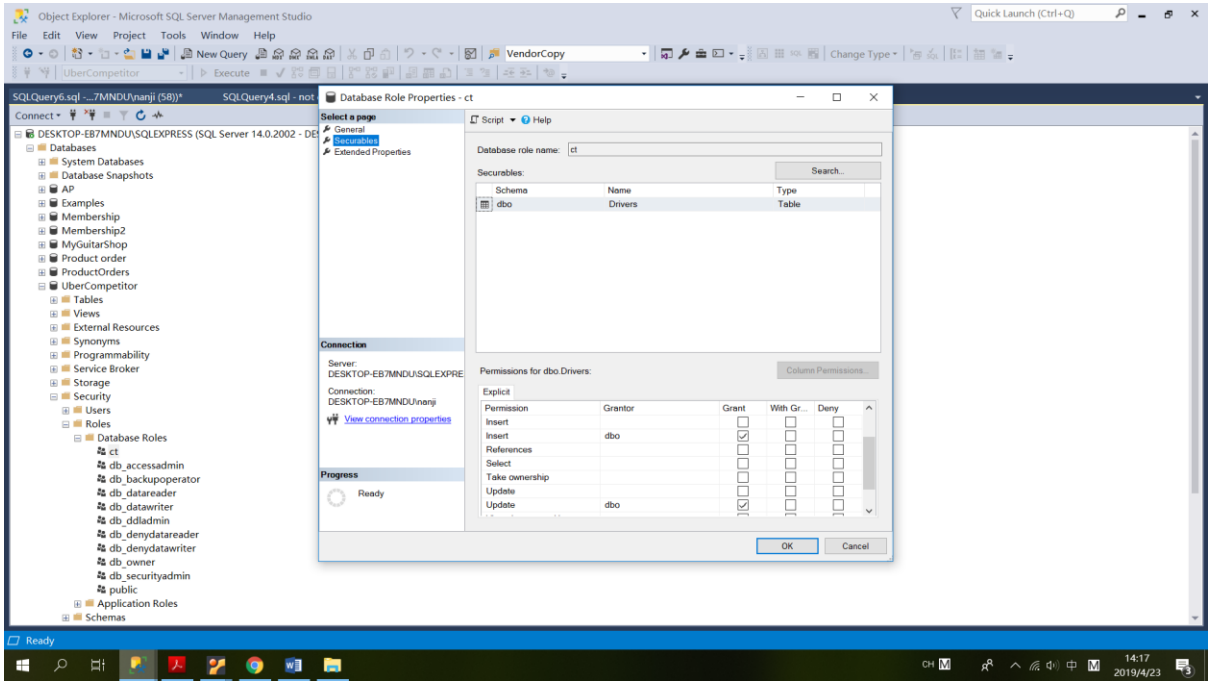
Result:



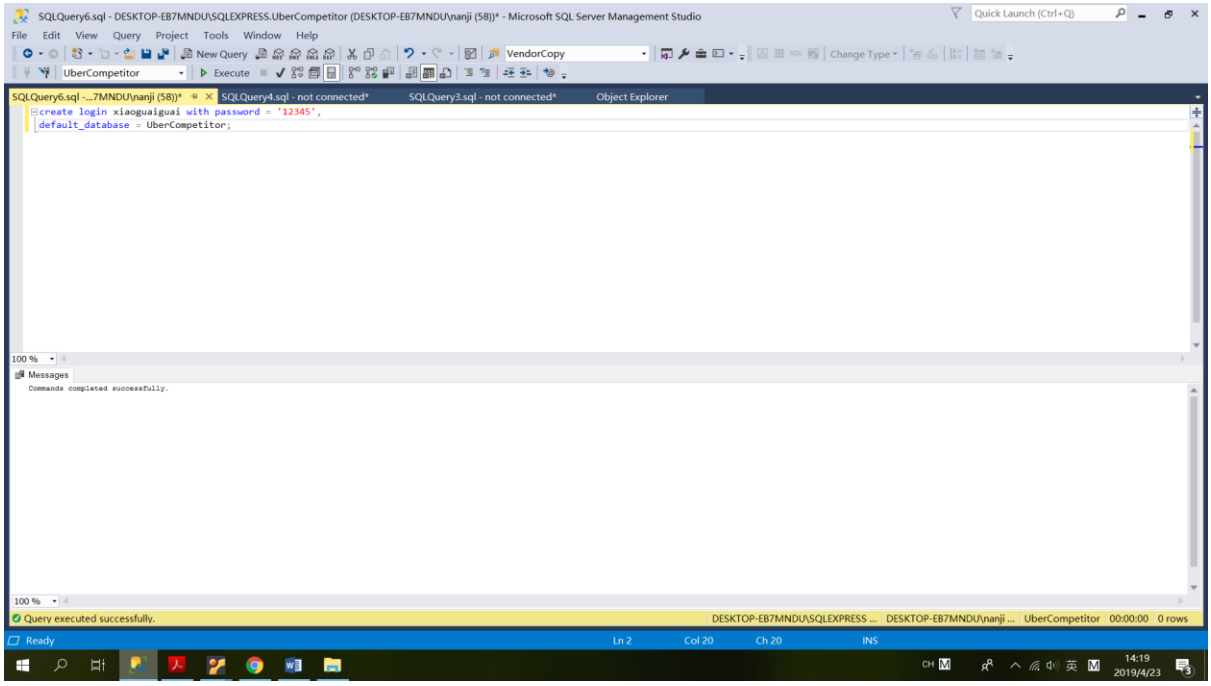
B. Permissions:



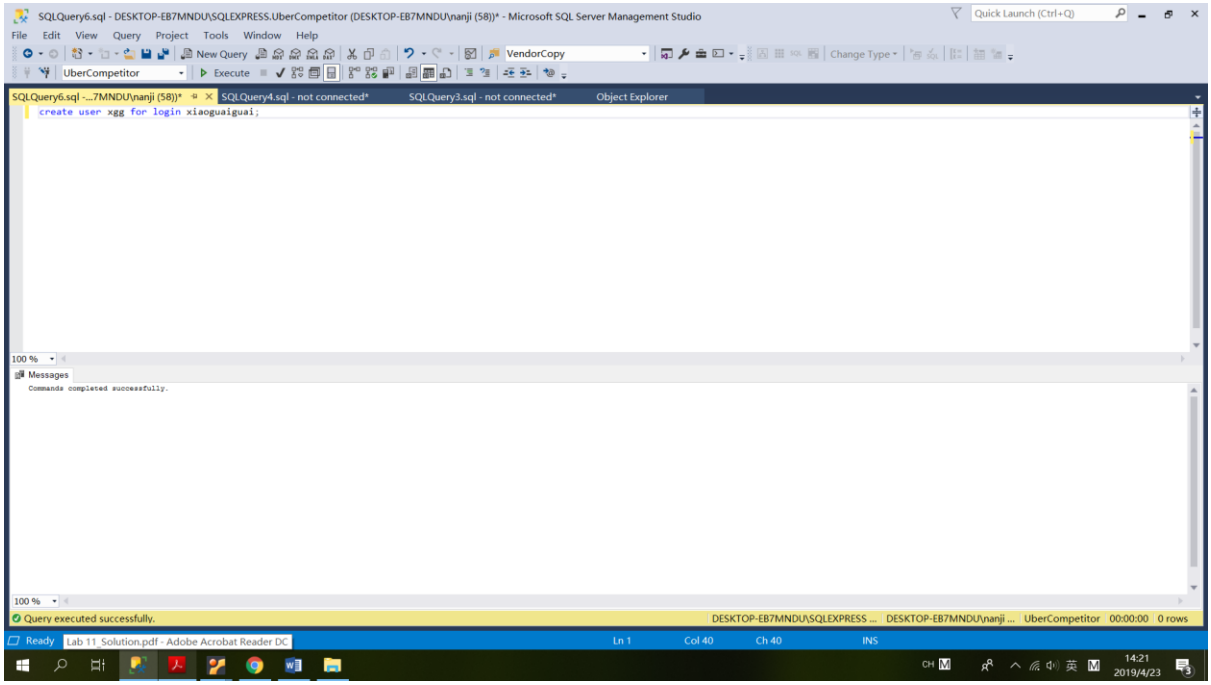
Result:



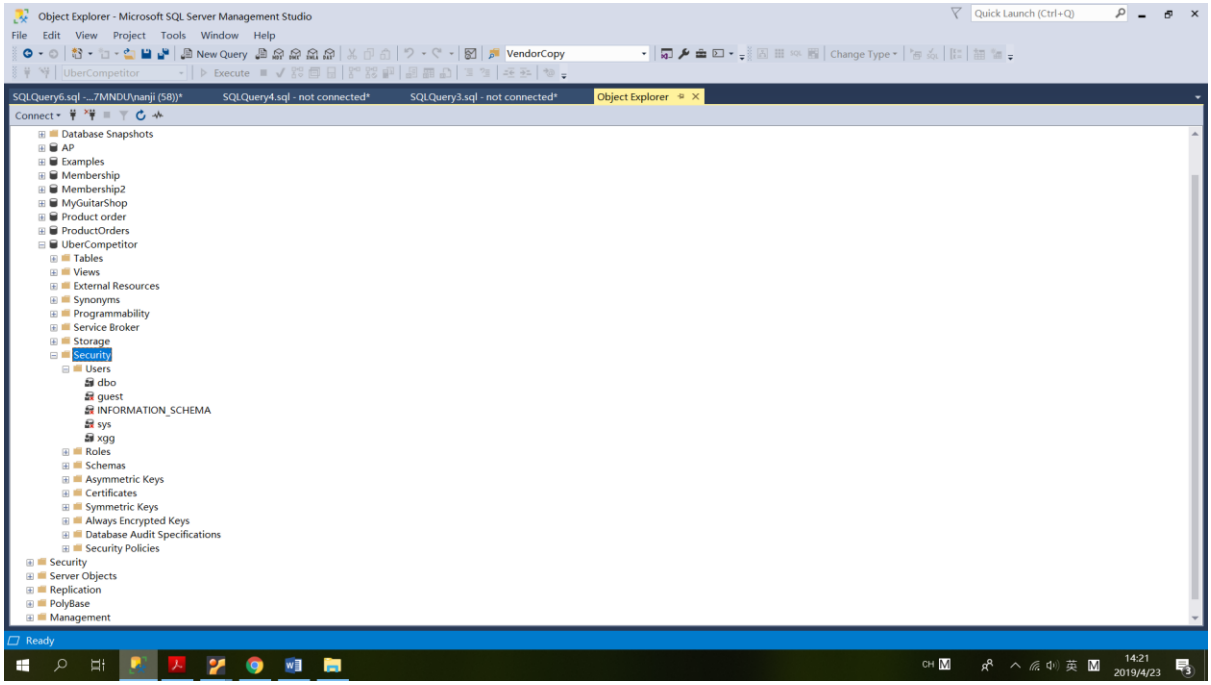
C. Login



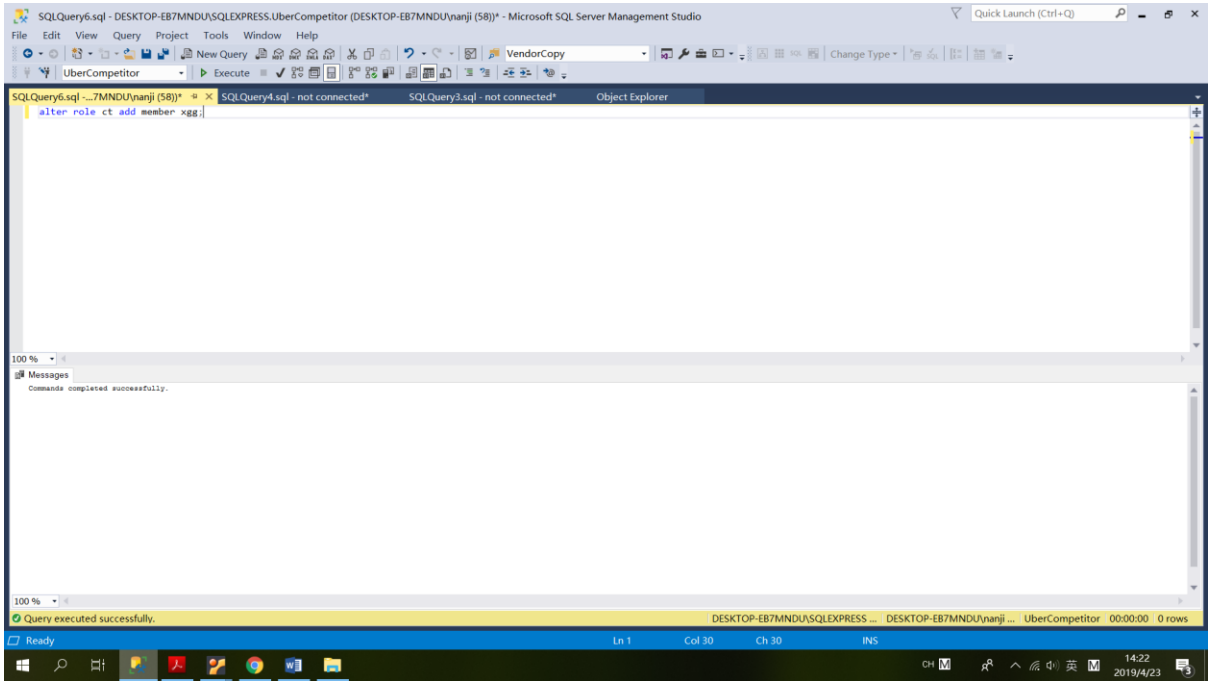
D. User



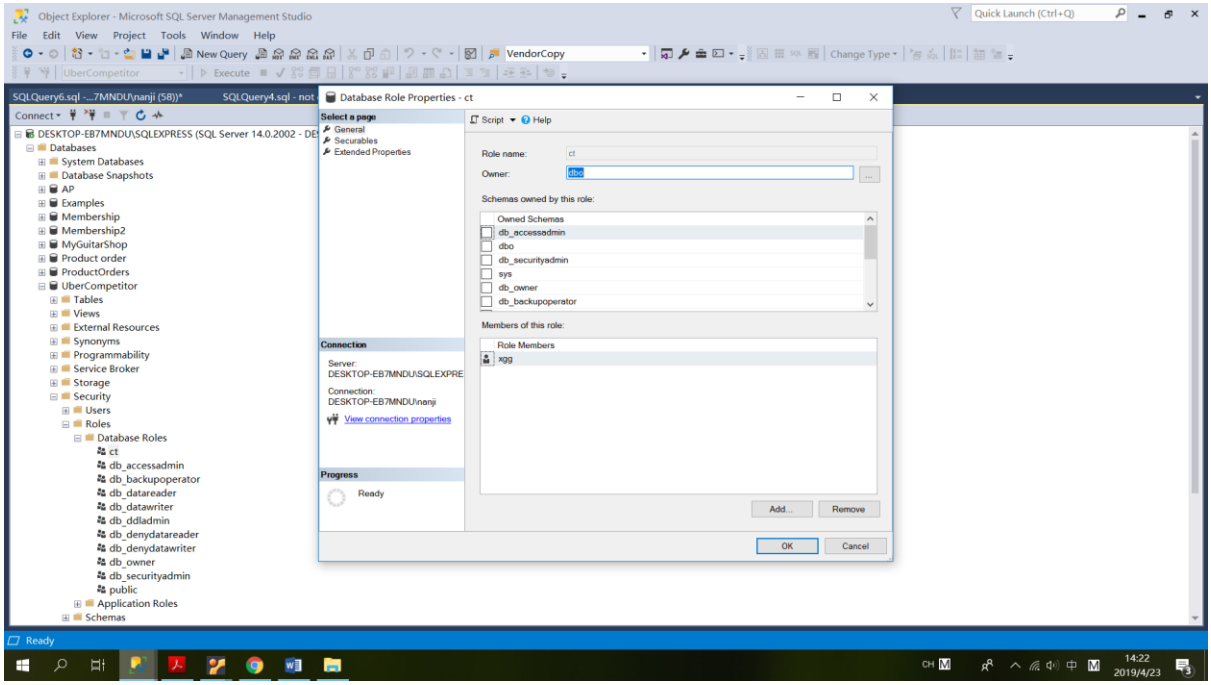
Result:



E. Role Member



Result:


Object Explorer - Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
New Query
VendorCopy
Change Type
Execute
Database Role Properties - ct
Select a page
General
Securables
Extended Properties
Script
Help
Role name:
ct
Owner:
dbo
Schemas owned by this role:
Owned Schemas
db_accessadmin
dbo
db_securityadmin
sys
db_owner
db_backupoperator
Members of this role:
Role Members
xgg
Connection
Server:
DESKTOP-EB7MNDU\SQLEXPRE
Connection:
DESKTOP-EB7MNDU\nanji
View connection properties
Progress
Ready
Add...
Remove
OK
Cancel
Connect
DESKTOP-EB7MNDU\SQLEXPRESS (SQL Server 14.0.2002 - DE
Databases
System Databases
Database Snapshots
AP
Examples
Membership
Membership2
MyGuitarShop
Product order
ProductOrders
UberCompetitor
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Users
Roles
Database Roles
ct
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public
Application Roles
Schemas
Ready
14:22
2019/4/23

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 leant 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.