Uber Competitor Database

Haoyu Gong

List

Introduction

In the first view, this database will have three main parts, customers, drivers, and trips. One customer could take many drivers' car. And one driver could offer many customers a ride. Therefore, the drivers table and customers tables will have a many-to-many relationship. A linking table will be very helpful. The trip part is like an invoice, connecting the customers and the drivers. It is a record among them.

Therefore, the primary focus of this project will be on the relationship between each other. The following operations will be on those basic implementations on trips. The order of this report is followed by my operating action. And all the indexes will be remarked.

The Uber Competitor DB is created by using the Microsoft SQL Server Management Studio. And this database can track and record all the information which could be given.

Design – Necessary Information

To make sure the Uber Competitor Database with effective functionality, we need to figure out the relationships among the given information.

Business problem:

Your job is to create a DB for an uber competitor. The following is the information that the DB needs to store.

Customers

- name
- home address
- stored credit cards
- user name
- email
- phone numbers (home, cell, business)
- active/inactive
- date of last trip taken
- number of trips taken this year
- ratings
  - date when review is left
  - driver that left it
  - score
  - text
  - for which trip was the review for

Trips/Reservations

- date when it was booked
- pick-up time
- drop-off time
- completed (yes/no)
- address where to pick up
- address where to drop off
- number of people
- number of bags
- customer notes
- driver notes
- customer
- driver
- cost paid
- tip
- credit card information of the card that paid the bill

Drivers

- name
- status (possible options – inactive, off work, working – available, working – with a fare)
- date of birth
- when did the driver start with our company
- driver license information
  - state
  - date of issue
  - date of expiry
  - license number
- insurance information
  - company
  - policy number
  - date of issue
  - date if expiry
- SSN
- home address
- bank account information (bank name, routing & account numbers, type of account (checking, savings))
- records of payments that the driver was paid (dates & the amount as well as a record of all pickups)
- customer ratings
  - text
  - score
  - date when left
  - customer that left it
  - for which trip was this review for
- driver's car information
  - make
  - model
  - year
  - color
  - car class (regular, luxury, SUV)
  - number of passengers it can fit
  - number of bags it can fit

We can clearly see all the basic attributes from above. Before I finish the final table decision, I made a primary mind map for the whole database, based on the need. The following excel will be attached.

	A	B	C
1		Customers	
2	CustomersInfo (table)	PaymentInfo (table)	RatingSummary (table)
3	name (will be divided into F and L)	CustomerID	CustomerID
4	home address (will be divided)	stored credit cards	TripID
5	phone numbers (home, cell, business)	active/inactive	ReviewDate
6	email		DriverID
7	user name		Score
8	CustomerID (new, PK)		Text
9			date of last trip taken
10			number of trips taken this year

	A	B	C
1		Trips/Reservations	
2	TripInfo (table)	CustomerDetail (table)	DriverDetail (table)
3	TripID	TripID	TripID
4	BookedDate	CustomerID	DriverID
5	PickUpDate	CostPaid	DriverNotes
6	DropOffDate	Tip	
7	PickUpAddress	CreditCard	
8	DropOffAddress	CustomerNotes	
9	NumberOfPeople		
10	NumberOfBages		
11	Completed		

	A	B	C	D	E	F	G
1				Drivers			
2	DriversInfo (table)	LicenseInfo (table)	InsuranceInfo (table)	BankAccount (table)	PaymentRecord (table)	CustomerRating (table)	CarInfo (table)
3	DriverID	DiverID	DriverID	DriverID	TripID	TripID	DriverID
4	DriverName	State	Company	BankName	DriverID	CustomerID	Make
5	Status	DateOfIssue	PolicyNumber	R&Anumber	PickUpDate	Score	Model
6	DOB	DateOfExpiry	DateOfIssue	Type	Cost	Text	Year
7	JoinDate	LicenseNumber	DateOfExpiry		Tip	LeftDate	Color
8	SSN				PaymentTotal	DriverID	CarClass
9	DriverAddress (will be divided)						PassengersNumber
10							BagsNumber

After having a primary thought and transforming into 3NF, we can separate it into the final version.

Customers	CustomerAddress	CustomerPaymentInfo	DriverRating	CustomerSummary
CustomerID	AddressID	PaymentID	RatingID	SummaryID
CustomerFName	CustomerID	CustomerID	DriverID	CustomerID
CustomerLName	AddressLine1	CreditCard	TripID	Active
UserName	AddressLine2		ReviewDate	LastTripDate
	City		Score	TripsNumberThisYear
	State		Text	
	ZipCode			
	HomePhoneNumber			
	CellPhoneNumber			
	BusinessPhoneNumber			
	Email			

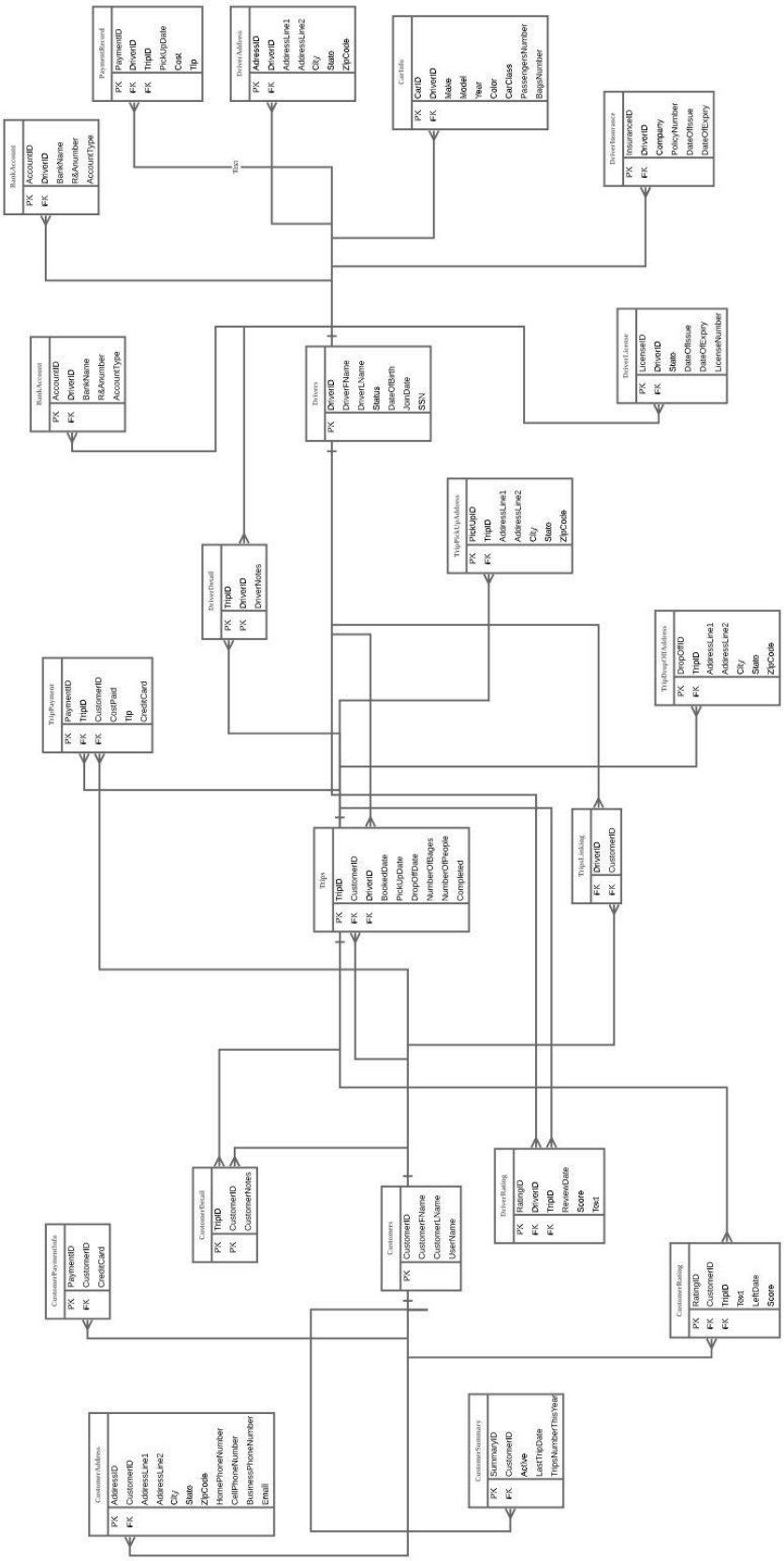
Trips	TripPayment	DriverDetail	CustomerDetail	TripsLinking	TripPickUpAddress	TripDropOffAddress
TripID	PaymentID	TripID	TripID	DriverID	PickUpID	DropOffID
CustomerID	TripID	DriverID	CustomerID	CustomerID	TripID	TripID
DriverID	CustomerID	DriverNotes	CustomerNotes		AddressLine1	AddressLine1
BookedDate	CostPaid				AddressLine2	AddressLine2
PickUpDate	Tip				City	City
DropOffDate	CreditCard				State	State
NumberOfBages					ZipCode	ZipCode
Completed						
NumberOfPeople						

Drivers	DriverAddress	DriverLicense	DriverInsurance	BankAccount	PaymentRecord	CustomerRating	CarInfo
DriverID	AdressID	LicenseID	InsuranceID	AccountID	PaymentID	RatingID	CarID
DriverFName	DriverID	DriverID	DriverID	DriverID	DriverID	CustomerID	DriverID
DriverLName	AddressLine1	State	Company	BankName	TripID	TripID	Make
Status	AddressLine2	DateOfIssue	PolicyNumber	R&Anumber	PickUpDate	Text	Model
DateOfBirth	City	DateOfExpiry	DateOfIssue	AccountType	Cost	LeftDate	Year
JoinDate	State	LicenseNumber	DateOfExpiry		Tip	Score	Color
SSN	ZipCode						CarClass
							PassengersNumber
							BagsNumber

As we can see above, this version marked table name, primary keys (red), foreign keys (blue). All of them have been transformed into 3NF. The relationship between drivers and customers is many-to-many. Therefore, there is a linking table. Besides, I put the email information and phone number into address table, rather than customers. It is because I think it belongs to the information about how to contact this guy, and the customers table is just about the pure, basic customer information.

Design – E/R diagram

The following image describes the complete database diagram, by ER diagram.



Design - Constraints

After figuring out all the table details and the relationships between them, we can create a query to implement this database. During coding, I check every column to find whether I need to add some constraints or not, and somewhere to allow default value.

```sql
CREATE TABLE Drivers (
        DriverID INT PRIMARY KEY IDENTITY(21,1),
        DriverFName VARCHAR(50) NOT NULL,
        DriverLName VARCHAR(50) NOT NULL,
        Status VARCHAR(25) NOT NULL,
        DateOfBirth DATETIME NOT NULL,
        JoinDate DATETIME NOT NULL,
        SSN CHAR(9) NOT NULL UNIQUE,
        CHECK (DateOfBirth < JoinDate)
);
```

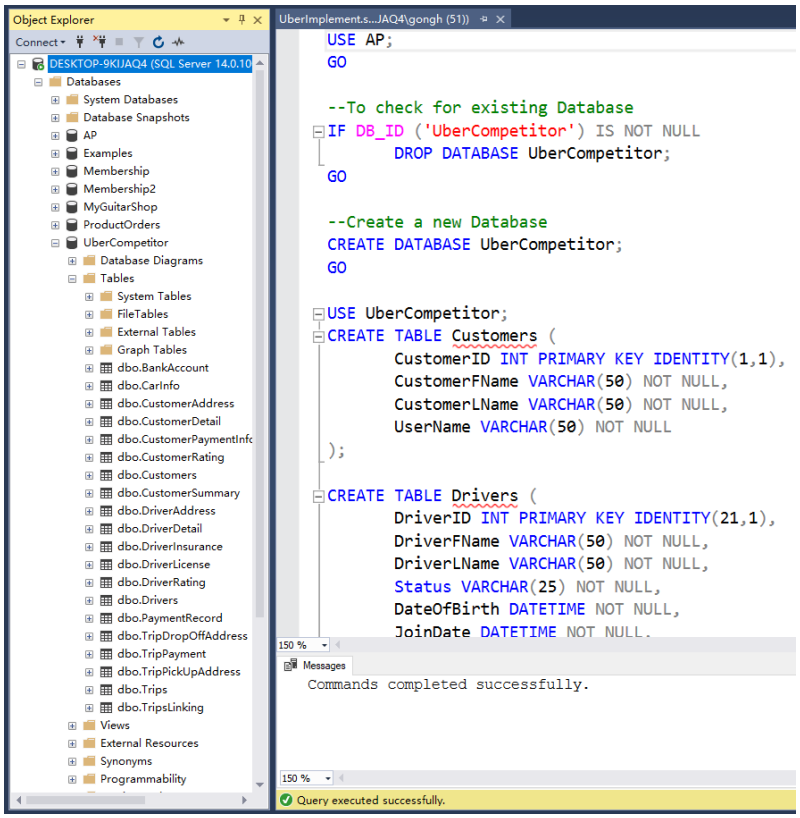
```sql
CREATE TABLE CustomerSummary (
        SummaryID INT PRIMARY KEY IDENTITY(91,1),
        CustomerID INT REFERENCES Customers (CustomerID),
        Active CHAR(1) NOT NULL,
        LastTripDate DATETIME DEFAULT 'NO TRIP YET',
        TripsNumberThisYear DECIMAL DEFAULT 0
);
```

Besides, I used the Primary Key attribute to generate composite primary key:

```sql
CREATE TABLE DriverDetail (
        TripID INT NOT NULL REFERENCES Trips(TripID),
        DriverID INT NOT NULL REFERENCES Drivers(DriverID),
        DriverNotes TEXT DEFAULT 'NO NOTES YET',
        PRIMARY KEY (TripID, DriverID)
);
```

Implementation - SQL Code

My query was executed successfully. Here is the image.




```
USE AP;
GO

--To check for existing Database
IF DB_ID ('UberCompetitor') IS NOT NULL
      DROP DATABASE UberCompetitor;
GO

--Create a new Database
CREATE DATABASE UberCompetitor;
GO

USE UberCompetitor;
CREATE TABLE Customers (
      CustomerID INT PRIMARY KEY IDENTITY(1,1),
       CustomerFName VARCHAR(50) NOT NULL,
       CustomerLName VARCHAR(50) NOT NULL,
       UserName VARCHAR(50) NOT NULL
);

CREATE TABLE Drivers (
```

```sql
	DriverID INT PRIMARY KEY IDENTITY(21,1),
	DriverFName VARCHAR(50) NOT NULL,
	DriverLName VARCHAR(50) NOT NULL,
	Status VARCHAR(25) NOT NULL,
	DateOfBirth DATETIME NOT NULL,
	JoinDate DATETIME NOT NULL,
	SSN CHAR(9) NOT NULL UNIQUE,
	CHECK (DateOfBirth < JoinDate)
);

CREATE TABLE Trips (
	TripID INT PRIMARY KEY IDENTITY(31,1),
	CustomerID INT REFERENCES Customers (CustomerID),
	DriverID INT REFERENCES Drivers(DriverID),
	BookedDate DATETIME NOT NULL,
	PickUpDate DATETIME NOT NULL,
	DropOffDate DATETIME NOT NULL,
	NumberOfPeople DECIMAL DEFAULT 1,
	NumberOfBages DECIMAL DEFAULT 1,
	Completed CHAR(1) NOT NULL,
	CHECK (BookedDate <= PickUpDate),
	CHECK (PickUpDate <= DropOffDate)
);

CREATE TABLE TripPickUpAddress (
	PickUpID INT PRIMARY KEY IDENTITY(41,1),
	TripID INT REFERENCES Trips(TripID),
	AddressLine1 VARCHAR(50) NOT NULL,
	AddressLine2 VARCHAR(50) DEFAULT NULL,
	City VARCHAR(25) NOT NULL,
	State VARCHAR(25) NOT NULL,
	ZipCode CHAR(5) NOT NULL
);

CREATE TABLE TripDropOffAddress (
	DropOffID INT PRIMARY KEY IDENTITY(51,1),
	TripID INT REFERENCES Trips(TripID),
	AddressLine1 VARCHAR(50) NOT NULL,
	AddressLine2 VARCHAR(50) DEFAULT NULL,
	City VARCHAR(25) NOT NULL,
	State VARCHAR(25) NOT NULL,
	ZipCode CHAR(5) NOT NULL
);
```

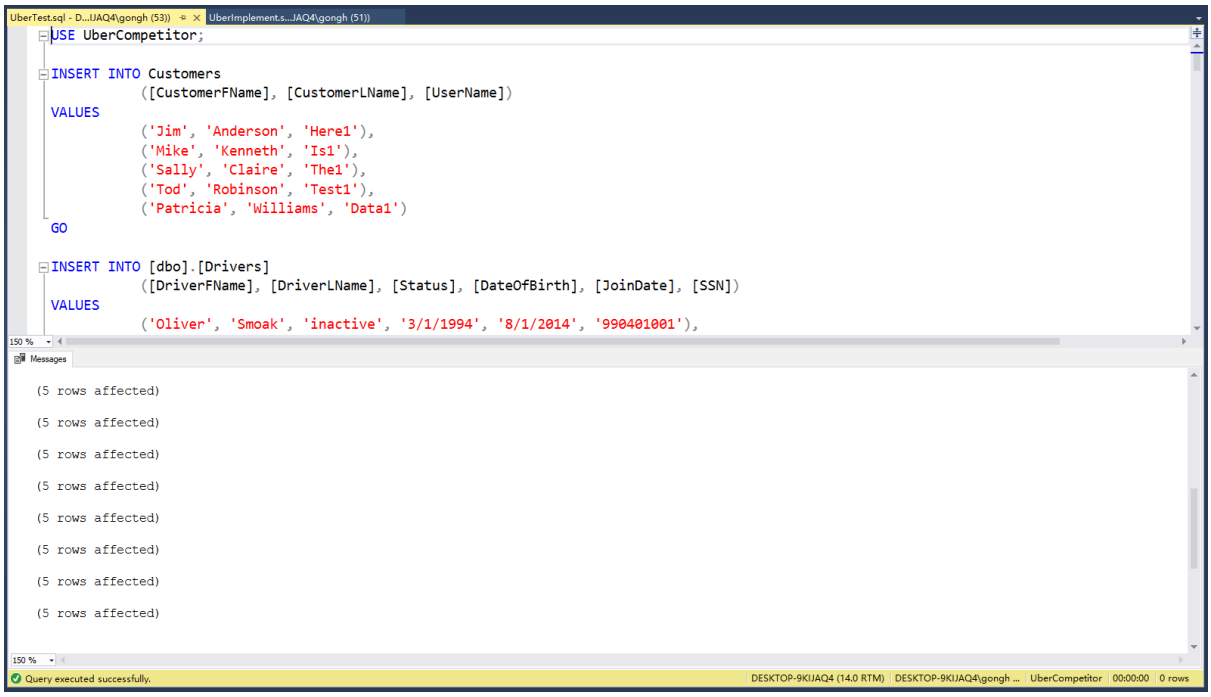
```sql
CREATE TABLE CustomerAddress (
	AddressID CHAR(6) NOT NULL PRIMARY KEY,
	CustomerID INT REFERENCES Customers (CustomerID),
	AddressLine1 VARCHAR(50) NOT NULL,
	AddressLine2 VARCHAR(50) DEFAULT NULL,
	City VARCHAR(25) NOT NULL,
	State VARCHAR(25) NOT NULL,
	ZipCode CHAR(5) NOT NULL,
	CellPhoneNumber VARCHAR(12) NOT NULL,
	HomePhoneNumber VARCHAR(12) DEFAULT NULL,
	BusinessPhoneNumber VARCHAR(12) DEFAULT NULL,
	Email VARCHAR(50) NOT NULL
);

CREATE TABLE CustomerPaymentInfo (
	PaymentID INT PRIMARY KEY IDENTITY(71,1),
	CustomerID INT REFERENCES Customers (CustomerID),
	CreditCard CHAR(16) NOT NULL
);

CREATE TABLE DriverRating (
	RatingID INT PRIMARY KEY IDENTITY(81,1),
	DriverID INT REFERENCES Drivers(DriverID),
	TripID INT REFERENCES Trips(TripID),
	ReviewDate DATETIME NOT NULL,
	Score CHAR(1) NOT NULL,
	RatingText TEXT DEFAULT 'NO COMMENTS YET'
);

CREATE TABLE CustomerSummary (
	SummaryID INT PRIMARY KEY IDENTITY(91,1),
	CustomerID INT REFERENCES Customers (CustomerID),
	Active CHAR(1) NOT NULL,
	LastTripDate DATETIME DEFAULT 'NO TRIP YET',
	TripsNumberThisYear DECIMAL DEFAULT 0
);

CREATE TABLE TripPayment (
	PaymentID INT PRIMARY KEY IDENTITY(101,1),
	TripID INT REFERENCES Trips(TripID),
	CustomerID INT REFERENCES Customers (CustomerID),
	CostPaid MONEY NOT NULL,
	Tip MONEY DEFAULT 0.00,
	CreditCard CHAR(16) NOT NULL
```

```sql
);

CREATE TABLE DriverDetail (
	TripID INT NOT NULL REFERENCES Trips(TripID),
	DriverID INT NOT NULL REFERENCES Drivers(DriverID),
	DriverNotes TEXT DEFAULT 'NO NOTES YET',
	PRIMARY KEY (TripID, DriverID)
);

CREATE TABLE CustomerDetail (
	TripID INT NOT NULL REFERENCES Trips(TripID),
	CustomerID INT NOT NULL REFERENCES Customers(CustomerID),
	CustomerNotes TEXT DEFAULT 'NO NOTES YET',
	PRIMARY KEY (TripID, CustomerID)
);

CREATE TABLE TripsLinking (
	CustomerID INT REFERENCES Customers (CustomerID),
	DriverID INT REFERENCES Drivers(DriverID),
);

CREATE TABLE DriverAddress (
	AddressID INT PRIMARY KEY IDENTITY(111,1),
	DriverID INT REFERENCES Drivers (DriverID),
	AddressLine1 VARCHAR(50) NOT NULL,
	AddressLine2 VARCHAR(50) DEFAULT NULL,
	City VARCHAR(25) NOT NULL,
	State VARCHAR(25) NOT NULL,
	ZipCode CHAR(5) NOT NULL
);

CREATE TABLE DriverLicense (
	LicenseID INT PRIMARY KEY IDENTITY(121,1),
	DriverID INT REFERENCES Drivers (DriverID),
	State VARCHAR(25) NOT NULL,
	DateOfIssue DATETIME NOT NULL,
	DateOfExpiry DATETIME NOT NULL,
	LicenseNumber CHAR(20) NOT NULL,
	CHECK (DateOfIssue < DateOfExpiry)
);

CREATE TABLE DriverInsurance (
	InsuranceID INT PRIMARY KEY IDENTITY(131,1),
	DriverID INT REFERENCES Drivers (DriverID),
```

```sql
        Company VARCHAR(25) NOT NULL,
        PolicyNumber CHAR(20) NOT NULL,
        DateOfIssue DATETIME NOT NULL,
        DateOfExpiry DATETIME NOT NULL,
        CHECK (DateOfIssue < DateOfExpiry)
);

CREATE TABLE BankAccount (
       AccountID INT PRIMARY KEY IDENTITY(141,1),
        DriverID INT REFERENCES Drivers (DriverID),
        BankName VARCHAR(25) NOT NULL,
        RAnumber CHAR(20) NOT NULL,
        AccountType VARCHAR(20) NOT NULL
);

CREATE TABLE PaymentRecord (
       PaymentID INT PRIMARY KEY IDENTITY(151,1),
        DriverID INT REFERENCES Drivers (DriverID),
        TripID INT REFERENCES Trips(TripID),
        PickUpDate DATETIME NOT NULL,
        Cost MONEY NOT NULL,
        Tip MONEY DEFAULT 0.00,
);

CREATE TABLE CustomerRating (
       RatingID INT PRIMARY KEY IDENTITY(161,1),
        TripID INT REFERENCES Trips(TripID),
        CustomerID INT REFERENCES Customers (CustomerID),
        LeftDate DATETIME NOT NULL,
        Score CHAR(1) NOT NULL,
        RatingText TEXT DEFAULT 'NO COMMENTS YET'
);

CREATE TABLE CarInfo (
       CarID INT PRIMARY KEY IDENTITY(171,1),
        DriverID INT REFERENCES Drivers (DriverID),
        Make VARCHAR(25) NOT NULL,
        Model VARCHAR(25) NOT NULL,
        Year CHAR(25) NOT NULL,
        Color VARCHAR(25) NOT NULL,
        CarClass VARCHAR(25) NOT NULL,
        PassengersNumber DECIMAL NOT NULL,
        BagsNumber DECIMAL NOT NULL
);
```

Testing – Populate with test data

This part followed the related instruction, each table 5 rows. All of them have been executed successfully.




```
USE UberCompetitor;

INSERT INTO Customers
          ([CustomerFName], [CustomerLName], [UserName])
VALUES
          ('Jim', 'Anderson', 'Here1'),
          ('Mike', 'Kenneth', 'Is1'),
          ('Sally', 'Claire', 'The1'),
          ('Tod', 'Robinson', 'Test1'),
          ('Patricia', 'Williams', 'Data1')
GO

INSERT INTO [dbo].[Drivers]
          ([DriverFName], [DriverLName], [Status], [DateOfBirth], [JoinDate], [SSN])
VALUES
          ('Oliver', 'Smoak', 'inactive', '3/1/1994', '8/1/2014', '990401001'),
          ('Lenna', 'Rodcliff', 'off work', '10/19/1980', '8/1/2014', '990203124'),
          ('James', 'GoldStein', 'working available', '5/8/1991', '10/3/2016',
'990158970'),
          ( 'Barry', 'Zimmer', 'working available', '12/20/1989', '11/4/2015',
'990101330'),
          ('David', 'GoldStein', 'working with a fare', '6/16/1986', '5/20/2016',
'990020982')
```

```sql
GO

INSERT INTO [dbo].[Trips]
        ([CustomerID], [DriverID], [BookedDate], [PickUpDate], [DropOffDate],
[NumberOfPeople], [NumberOfBages], [Completed])
VALUES
        (1, 23, '1/1/2019', '1/2/2019', '1/2/2019', 2, 1, 'Y'),
        (2, 23, '2/2/2019', '2/2/2019', '2/2/2019', 1, 1, 'Y'),
        (3, 24, '3/14/2019', '3/14/2019', '3/14/2019', 1, 1, 'Y'),
        (4, 23, '3/15/2019', '3/15/2019', '3/15/2019', 3, 2, 'Y'),
        (5, 24, '5/2/2019', '5/3/2019', '5/3/2019', 1, 1, 'Y')
GO

INSERT INTO [dbo].[TripPickUpAddress]
        ([TripID], [AddressLine1], [AddressLine2], [City], [State], [ZipCode])
VALUES
        (31, '456 WestCott', '', 'Syracuse','NY','13210'),
        (32, '349 James st', '', 'Newark','NJ','13021'),
        (33, '990 Westcott','','Buffalo','NY','10210'),
        (34, '204 Euclid Ave','','Manchester','NH','03217'),
        (35, '990 Westcott','','Buffalo','NY','10210')
GO

INSERT INTO [dbo].[TripDropOffAddress]
        ([TripID], [AddressLine1], [AddressLine2], [City], [State], [ZipCode])
VALUES
        (31, '349 James st', '', 'Newark','NJ','13021'),
        (32, '990 Westcott','','Buffalo','NY','10210'),
        (33, '204 Euclid Ave','','Manchester','NH','03217'),
        (34, '990 Westcott','','Buffalo','NY','10210'),
        (35, '456 WestCott', '', 'Syracuse','NY','13210')
GO

INSERT INTO CustomerAddress
        ([AddressID], [CustomerID], [AddressLine1], [AddressLine2], [City], [State],
[ZipCode],
          [CellPhoneNumber], [HomePhoneNumber], [BusinessPhoneNumber], [Email])
VALUES
        ('CA0601', 1, '456 WestCott', '', 'Syracuse','NY','13210', '3154500501', '',
'', 'JAnderson@gmail.com'),
        ('CA0602', 2, '349 James st', '', 'Newark','NJ','13021', '2019045064', '',
'', 'MKenneth@gmail.com'),
        ('CA0603', 3, '990 Westcott','','Buffalo','NY','10210', '6065458150', '',
'', 'SClaire@gmail.com'),
```

```
        ('CA0604', 4, '204 Euclid Ave','','Manchester','NH','03217', '4199051205',
'', '', 'TRobinson@gmail.com'),
        ('CA0605', 5, '990 Westcott','','Buffalo','NY','10210', '3154500502', '',
'', 'PWilliams@gmail.com')
GO

INSERT INTO CustomerPaymentInfo
        ([CustomerID], [CreditCard])
VALUES
        (1, '4013686575532315'),
        (2, '4013978943831895'),
        (3, '4013821300893092'),
        (4, '4013889611805402'),
        (5, '4013477115573393')
GO

INSERT INTO [dbo].[DriverRating]
        ([DriverID], [TripID], [ReviewDate], [Score], [RatingText])
VALUES
        (23, 31, '1/2/2019', '5', ''),
        (23, 32, '2/2/2019', '5', ''),
        (24, 33, '3/14/2019', '5', ''),
        (23, 34, '3/15/2019', '4', ''),
        (24, 35, '5/3/2019', '5', '')
GO

INSERT INTO CustomerSummary
        ([CustomerID], [Active], [LastTripDate], [TripsNumberThisYear])
VALUES
        (1, 'Y', '1/2/2019', 1),
        (2, 'Y', '2/2/2019', 1),
        (3, 'Y', '3/14/2019', 1),
        (4, 'Y', '3/15/2019', 1),
        (5, 'Y', '5/3/2019', 1)
GO

INSERT INTO [dbo].[TripPayment]
        ([TripID], [CustomerID], [CostPaid], [Tip], [CreditCard])
VALUES
        (31, 1, '20.01', '2.00', '4013686575532315'),
        (32, 2, '18.09', '1.80', '4013978943831895'),
        (33, 3, '25.34', '2.53', '4013821300893092'),
        (34, 4, '34.87', '3.48', '4013889611805402'),
        (35, 5, '55.67', '5.56', '4013477115573393')
```

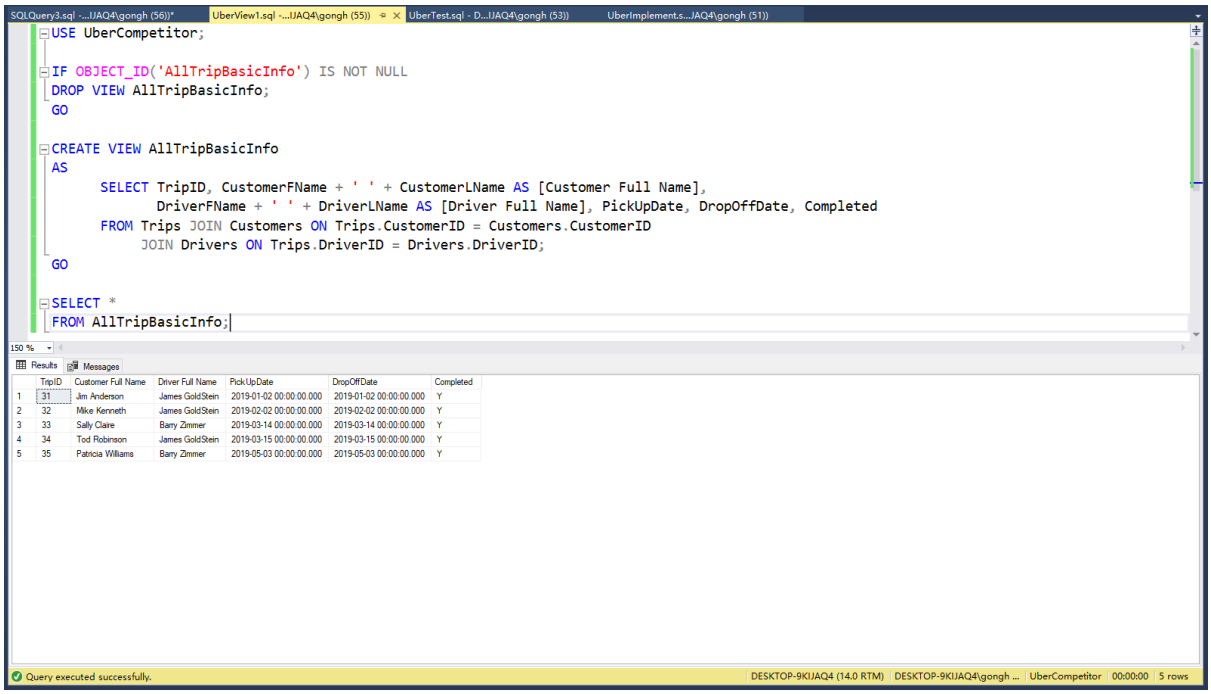
```sql
GO

INSERT INTO [dbo].[DriverDetail]
          ([TripID], [DriverID],[DriverNotes])
VALUES
          (31, 23, 'VERY KIND!'),
          (32, 23, ''),
          (33, 24, ''),
          (34, 23, ''),
          (35, 24, '')
GO

INSERT INTO [dbo].[CustomerDetail]
          ([TripID], [CustomerID],[CustomerNotes])
VALUES
          (31, 1, ''),
          (32, 2, ''),
          (33, 3, ''),
          (34, 4, ''),
          (35, 5, 'VERY GOOD!')
GO

INSERT INTO [dbo].[TripsLinking]
          ([CustomerID], [DriverID])
VALUES
          (1, 23),
          (2, 23),
          (3, 24),
          (4, 23),
          (5, 24)
GO

INSERT INTO [dbo].[DriverAddress]
          ([DriverID], [AddressLine1], [AddressLine2], [City], [State], [ZipCode])
VALUES
          (21, '1707 Lakebend Way', '', 'Dover', 'DE', '19901'),
          (22, '2181 Walker Ave', '', 'Dallas', 'TX', '75219'),
          (23, '1790 Lakepoint', '', 'Helena', 'MT', '59601'),
          (24, '2347 Ludwick', '', 'Cheyenne', 'WY', '82001'),
          (25, '1261 Northwood', '', 'Charleston', 'WV', '25302')
GO

INSERT INTO [dbo].[DriverLicense]
          ([DriverID], [State], [DateOfIssue], [DateOfExpiry], [LicenseNumber])
```

```sql
VALUES
        (21, 'DE', '8/1/2013', '8/1/2023', 'JHYRWVH675IK'),
        (22, 'TX', '8/1/2013', '8/1/2023', 'JYSXWQK368RT'),
        (23, 'MT', '10/3/2015', '10/3/2025', 'MKGVEWL873LO'),
        (24, 'WY', '11/4/2014', '11/4/2024', 'NJTCEUM456VF'),
        (25, 'WV', '5/20/2015', '5/20/2025', 'MKYVENC346TH')
GO

INSERT INTO [dbo].[DriverInsurance]
        ([DriverID], [Company], [PolicyNumber], [DateOfIssue], [DateOfExpiry])
VALUES
        (21, 'GEICO', '468468368632', '8/2/2013', '8/2/2023'),
        (22, 'STATEFARM', '874632986510', '8/2/2013', '8/2/2023'),
        (23, 'GEICO', '876427036518', '10/4/2015', '10/4/2025'),
        (24, 'STATEFARM', '194827403717', '11/5/2014', '11/5/2024'),
        (25, 'STATEFARM', '843754699273', '5/21/2015', '5/21/2025')
GO

INSERT INTO [dbo].[BankAccount]
        ([DriverID], [BankName], [RAnumber], [AccountType])
VALUES
        (21, 'BOA', '47629640', 'checking'),
        (22, 'Chase', '86397637', 'checking'),
        (23, 'BOA', '72394372', 'savings'),
        (24, 'Citi', '64564332', 'savings'),
        (25, 'Citi', '76543654', 'checking')
GO

INSERT INTO [dbo].[PaymentRecord]
        ([DriverID], [TripID], [PickUpDate], [Cost], [Tip])
VALUES
        (23, 31, '1/2/2019', '20.01', '2.00'),
        (23, 32, '2/2/2019', '18.09', '1.80'),
        (24, 33, '3/14/2019', '25.34', '2.53'),
        (23, 34, '3/15/2019', '34.87', '3.48'),
        (24, 35, '5/3/2019', '55.67', '5.56')
GO

INSERT INTO [dbo].[CustomerRating]
        ([TripID], [CustomerID], [LeftDate], [Score], [RatingText])
VALUES
        (31, 1, '1/2/2019', '5', ''),
        (32, 2, '2/2/2019', '5', ''),
        (33, 3, '3/14/2019', '5', ''),
```

```sql
            (34, 4, '3/15/2019', '4', ''),
            (35, 5, '5/3/2019', '5', '')
GO

INSERT INTO [dbo].[CarInfo]
            ([DriverID], [Make], [Model], [Year], [Color], [CarClass],
[PassengersNumber], [BagsNumber])
VALUES
            (21, 'Audi', 'A6', '2012', 'Black', 'regular', 4, 6),
            (22, 'Buick', 'GL8', '2013', 'Black', 'SUV', 4, 6),
            (23, 'Audi', 'A6', '2013', 'Black', 'regular', 6, 8),
            (24, 'Ford', 'Focus', '2014', 'Black', 'SUV', 4, 6),
            (25, 'Kia', 'K5', '2013', 'White', 'regular', 4, 6)
GO
```

Design – Business Reports / Testing - Reports

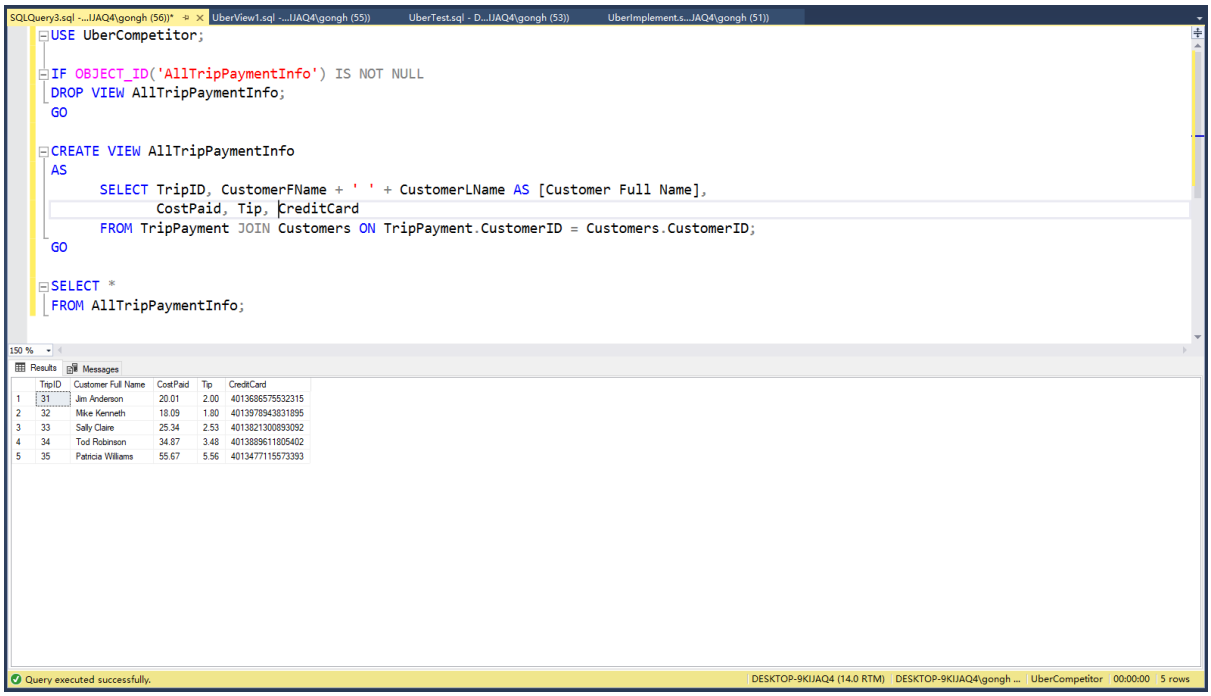
This view will show the basic information of all the trips.




```
USE UberCompetitor;

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

CREATE VIEW AllTripBasicInfo
AS
      SELECT TripID, CustomerFName + ' ' + CustomerLName AS [Customer Full Name],
             DriverFName + ' ' + DriverLName AS [Driver Full Name], PickUpDate,
DropOffDate, Completed
      FROM Trips JOIN Customers ON Trips.CustomerID = Customers.CustomerID
           JOIN Drivers ON Trips.DriverID = Drivers.DriverID;
GO

SELECT *
FROM AllTripBasicInfo;
```

Report: This part includes the most basic and helpful information between each other. But it doesn't show the exact time.

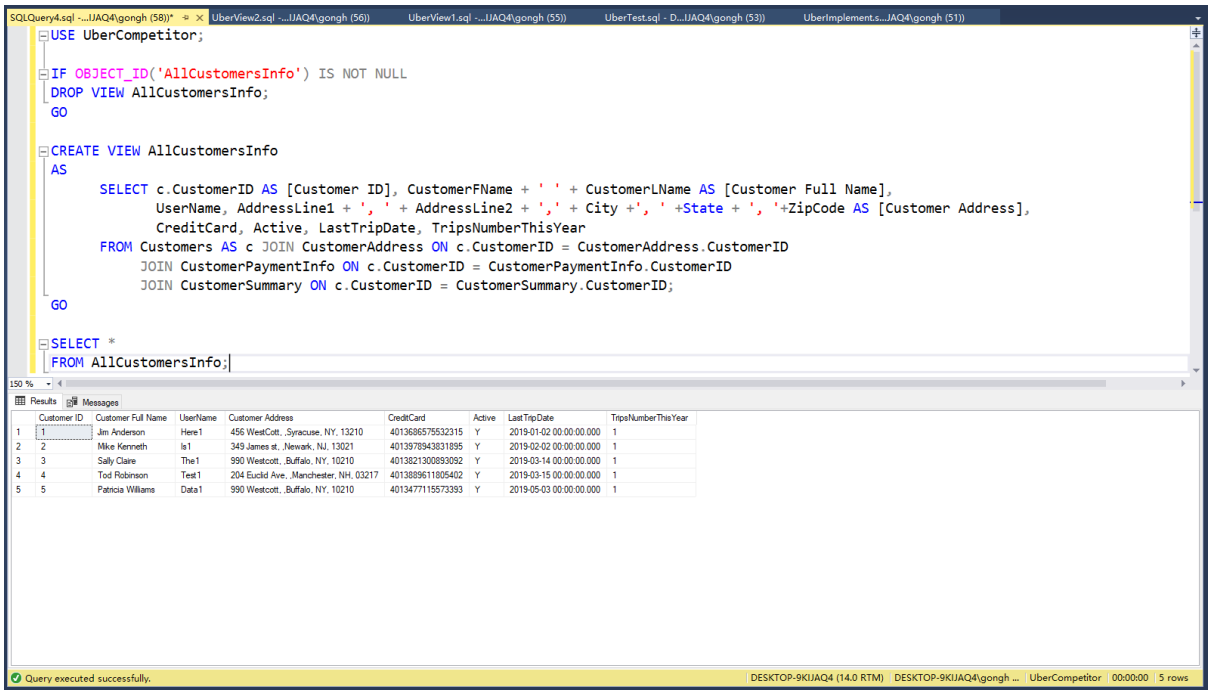
This view will show the payment information of all the trips.




```
USE UberCompetitor;

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

CREATE VIEW AllTripPaymentInfo
AS
      SELECT TripID, CustomerFName + ' ' + CustomerLName AS [Customer Full Name],
             CostPaid, Tip, CreditCard
      FROM TripPayment JOIN Customers ON TripPayment.CustomerID =
Customers.CustomerID;
GO

SELECT *
FROM AllTripPaymentInfo;
```

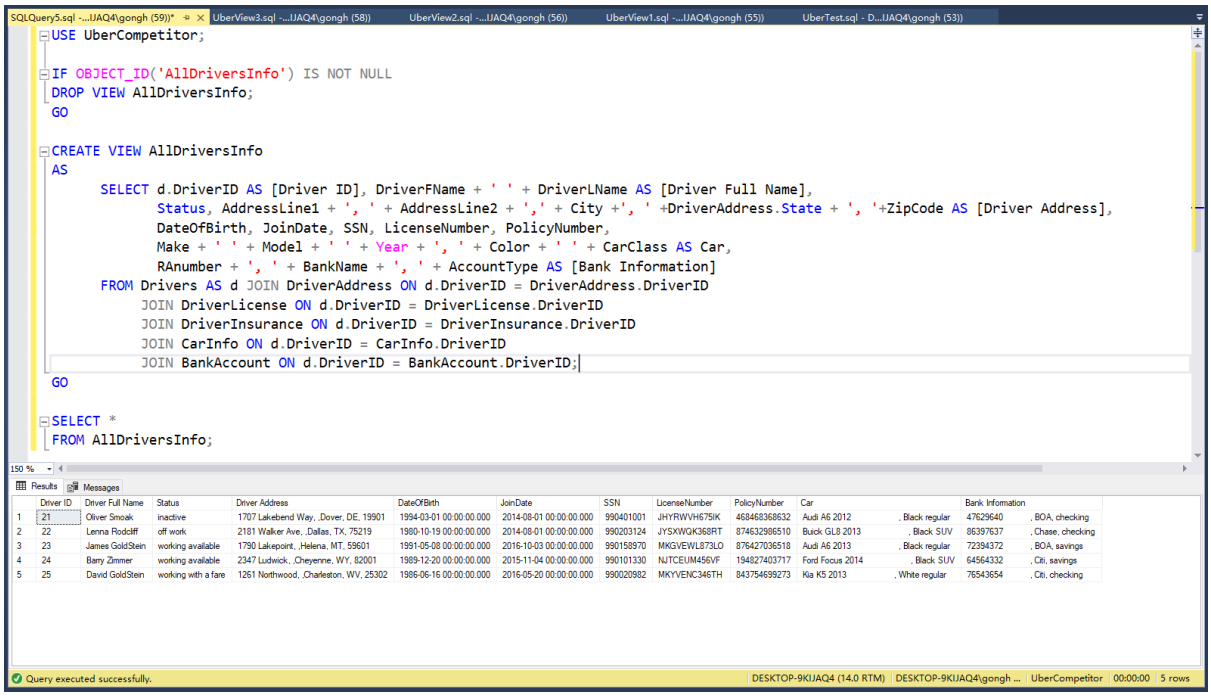
Report: We can see all the payment information here. Till now, all transactions have tips, which are about 10%.

This view will show us the whole information of all the customers.




```
USE UberCompetitor;

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

CREATE VIEW AllCustomersInfo
AS
      SELECT c.CustomerID AS [Customer ID], CustomerFName + ' ' + CustomerLName AS
[Customer Full Name],
             UserName, AddressLine1 + ', ' + AddressLine2 + ',' + City +', ' +State +
', '+ZipCode AS [Customer Address],
              CreditCard, Active, LastTripDate, TripsNumberThisYear
      FROM Customers AS c JOIN CustomerAddress ON c.CustomerID =
CustomerAddress.CustomerID
           JOIN CustomerPaymentInfo ON c.CustomerID = CustomerPaymentInfo.CustomerID
           JOIN CustomerSummary ON c.CustomerID = CustomerSummary.CustomerID;
GO

SELECT *
FROM AllCustomersInfo;
```

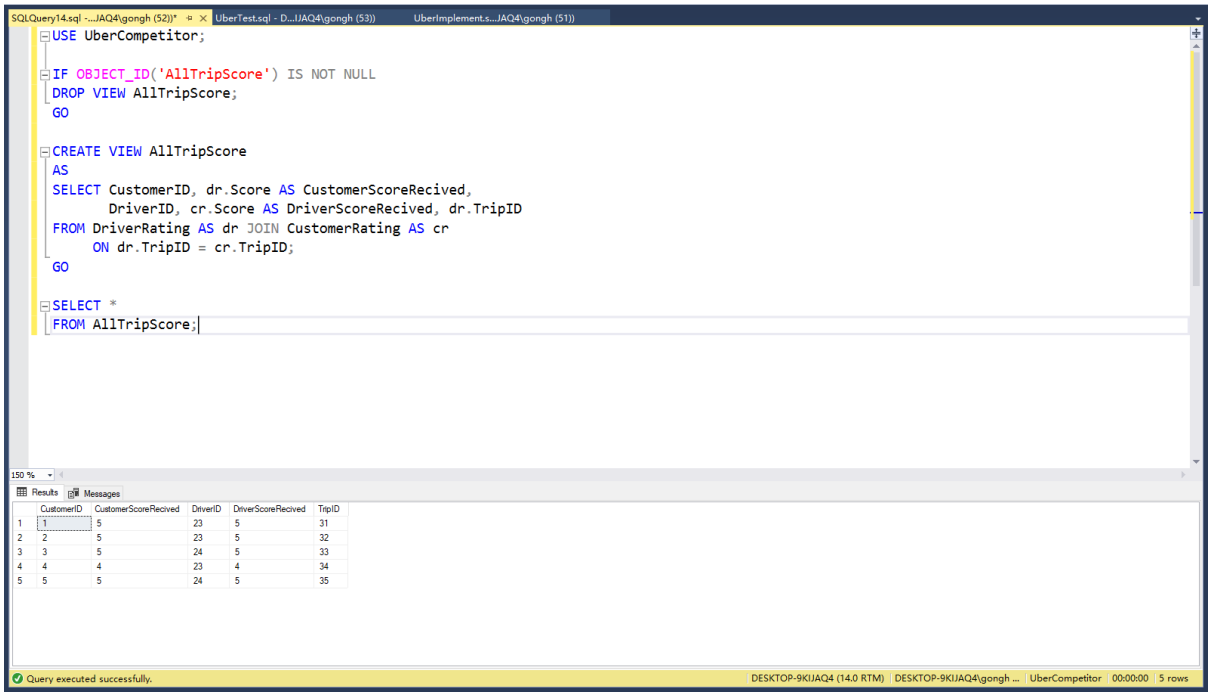
This view will show us the whole information of all the drivers.



	Driver ID	Driver Full Name	Status	Driver Address	DateOfBirth	JoinDate	SSN	LicenseNumber	PolicyNumber	Car	Bank Information
1	21	Oliver Smoak	inactive	1707 Lakebend Way, ,Dover, DE, 19901	1994-03-01 00:00:00.000	2014-08-01 00:00:00.000	990401001	JHYRWVH675IK	468468368632	Audi A6 2012 , Black regular	47629640 , BOA, checking
2	22	Lenna Rodcliff	off work	2181 Walker Ave, ,Dallas, TX, 75219	1980-10-19 00:00:00.000	2014-08-01 00:00:00.000	990203124	JYSXWQK368RT	874632986510	Buick GL8 2013 , Black SUV	86397637 , Chase, checking
3	23	James GoldStein	working available	1790 Lakepoint, ,Helena, MT, 59601	1991-05-08 00:00:00.000	2016-10-03 00:00:00.000	990158970	MKGVEWL873LO	876427036518	Audi A6 2013 , Black regular	72394372 , BOA, savings
4	24	Barry Zimmer	working available	2347 Ludwick, ,Cheyenne, WY, 82001	1989-12-20 00:00:00.000	2015-11-04 00:00:00.000	990101330	NJTCEUM456VF	194827403717	Ford Focus 2014 , Black SUV	64564332 , Citi, savings
5	25	David GoldStein	working with a fare	1251 Northwood, ,Charleston, WV, 25302	1986-06-16 00:00:00.000	2016-05-20 00:00:00.000	990020982	MKYVENC346TH	843754699273	Kia K5 2013 , White regular	76543654 , Citi, checking

```
USE UberCompetitor;

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

CREATE VIEW AllDriversInfo
AS
     SELECT d.DriverID AS [Driver ID], DriverFName + ' ' + DriverLName AS [Driver
Full Name],
            Status, AddressLine1 + ', ' + AddressLine2 + ',' + City +', '
+DriverAddress.State + ', '+ZipCode AS [Driver Address],
             DateOfBirth, JoinDate, SSN, LicenseNumber, PolicyNumber,
             Make + ' ' + Model + ' ' + Year + ', ' + Color + ' ' + CarClass AS Car,
             RAnumber + ', ' + BankName + ', ' + AccountType AS [Bank Information]
     FROM Drivers AS d JOIN DriverAddress ON d.DriverID = DriverAddress.DriverID
          JOIN DriverLicense ON d.DriverID = DriverLicense.DriverID
          JOIN DriverInsurance ON d.DriverID = DriverInsurance.DriverID
          JOIN CarInfo ON d.DriverID = CarInfo.DriverID
          JOIN BankAccount ON d.DriverID = BankAccount.DriverID;
GO

SELECT *
FROM AllDriversInfo;
```

This view will show all the feedbacks with each other, for each trip.

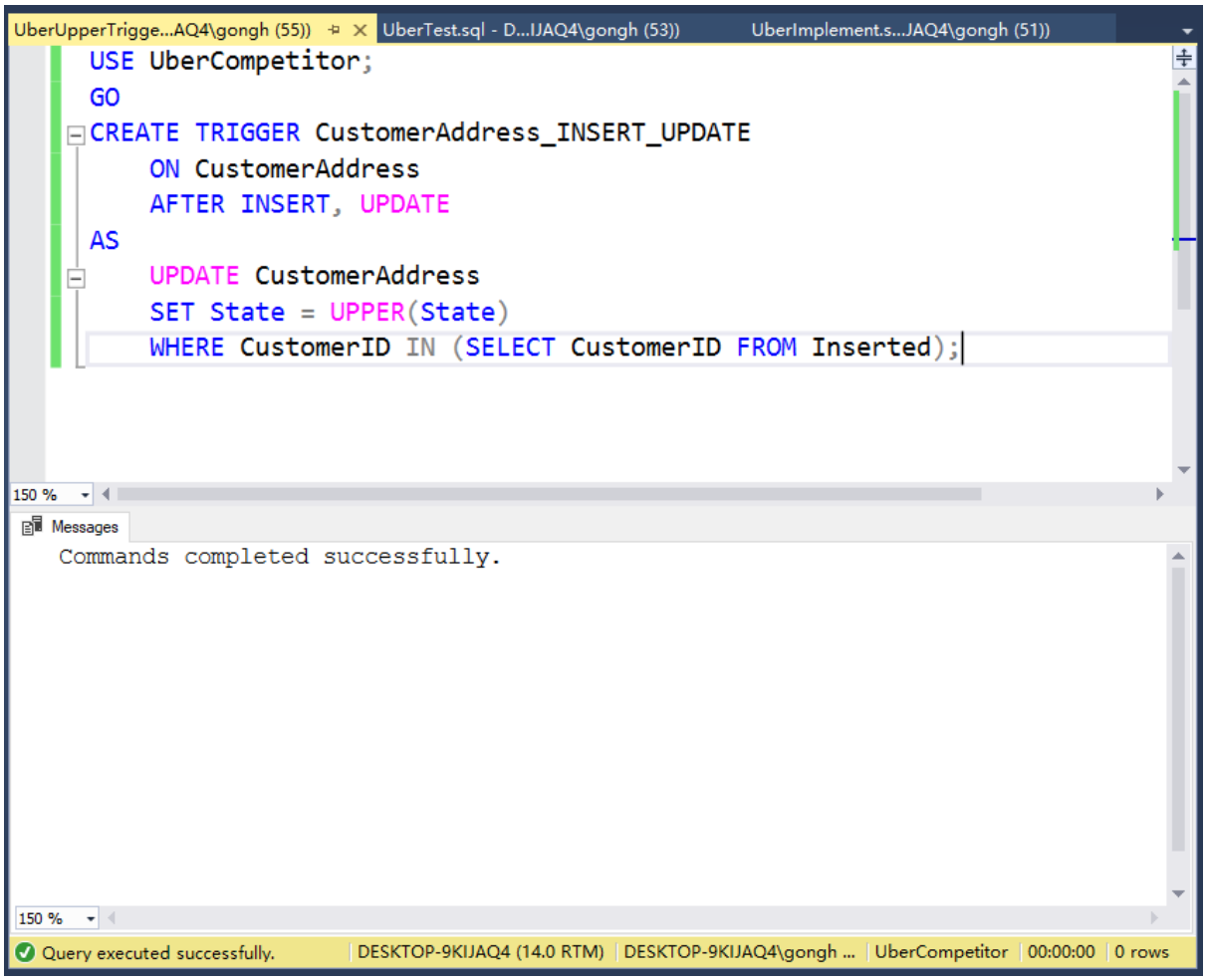



```sql
USE UberCompetitor;

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

CREATE VIEW AllTripScore
AS
SELECT CustomerID, dr.Score AS CustomerScoreRecived,
       DriverID, cr.Score AS DriverScoreRecived, dr.TripID
FROM DriverRating AS dr JOIN CustomerRating AS cr
     ON dr.TripID = cr.TripID;
GO

SELECT *
FROM AllTripScore;
```

Report: From the information given now, we can see both the customer and driver have a bad experience with each other.

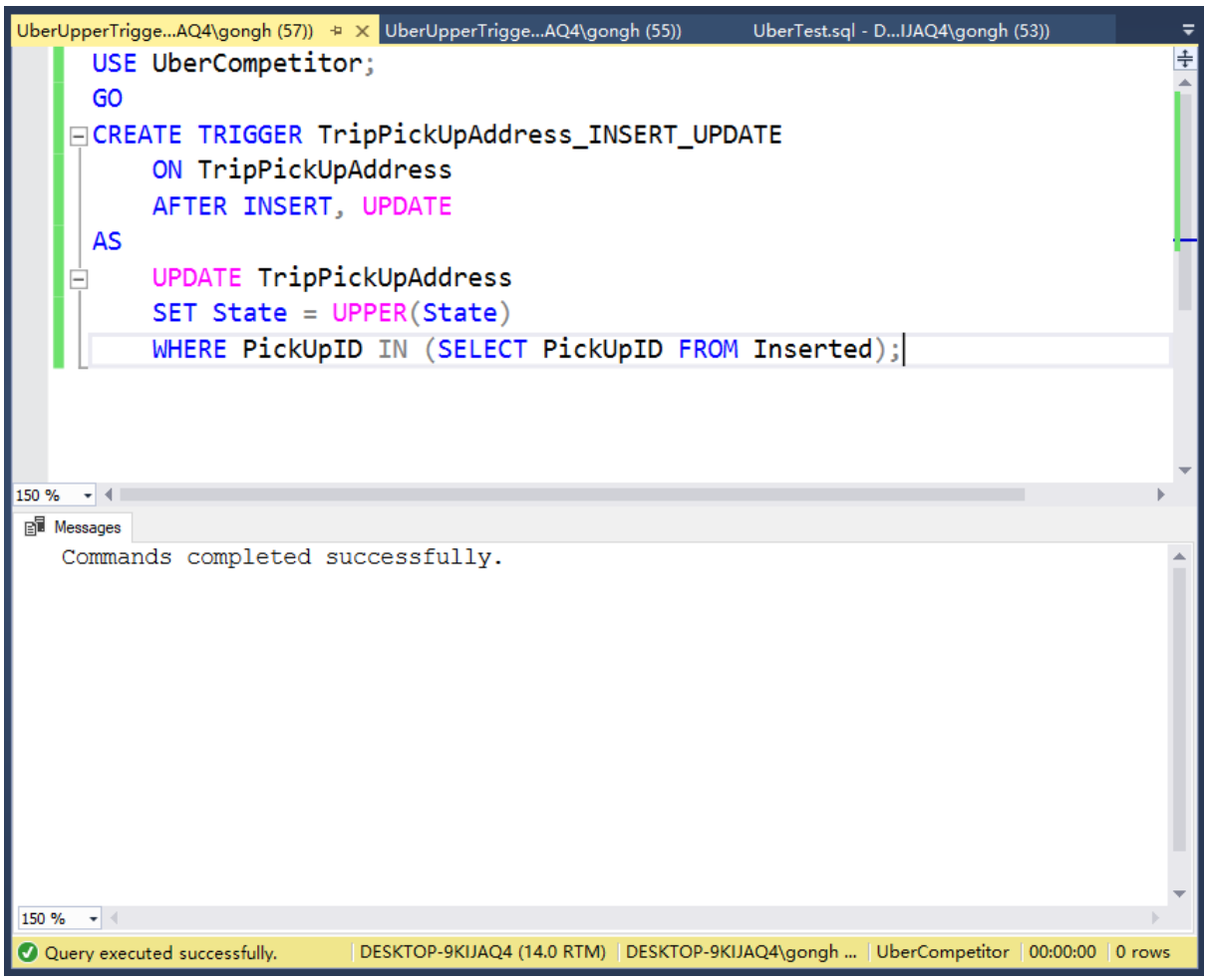
Design – Triggers

This trigger will help fix the lower problem when the new state insert into address table.



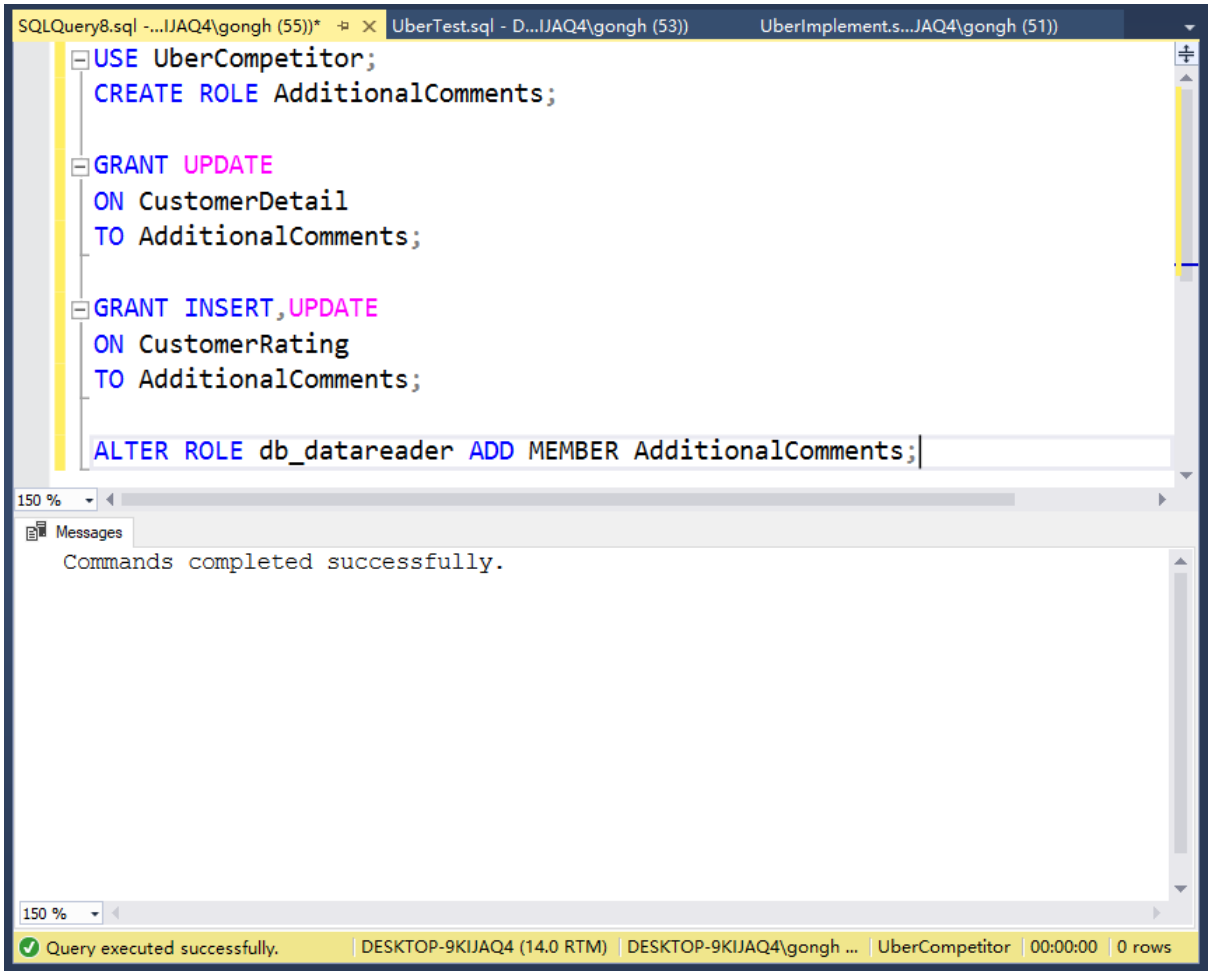

```
USE UberCompetitor;
GO
CREATE TRIGGER CustomerAddress_INSERT_UPDATE
    ON CustomerAddress
    AFTER INSERT, UPDATE
AS
    UPDATE CustomerAddress
    SET State = UPPER(State)
    WHERE CustomerID IN (SELECT CustomerID FROM Inserted);
```

This trigger have the same function, but on the TripPickUpAddress table. The same way can be used in any table which have 'state' column.

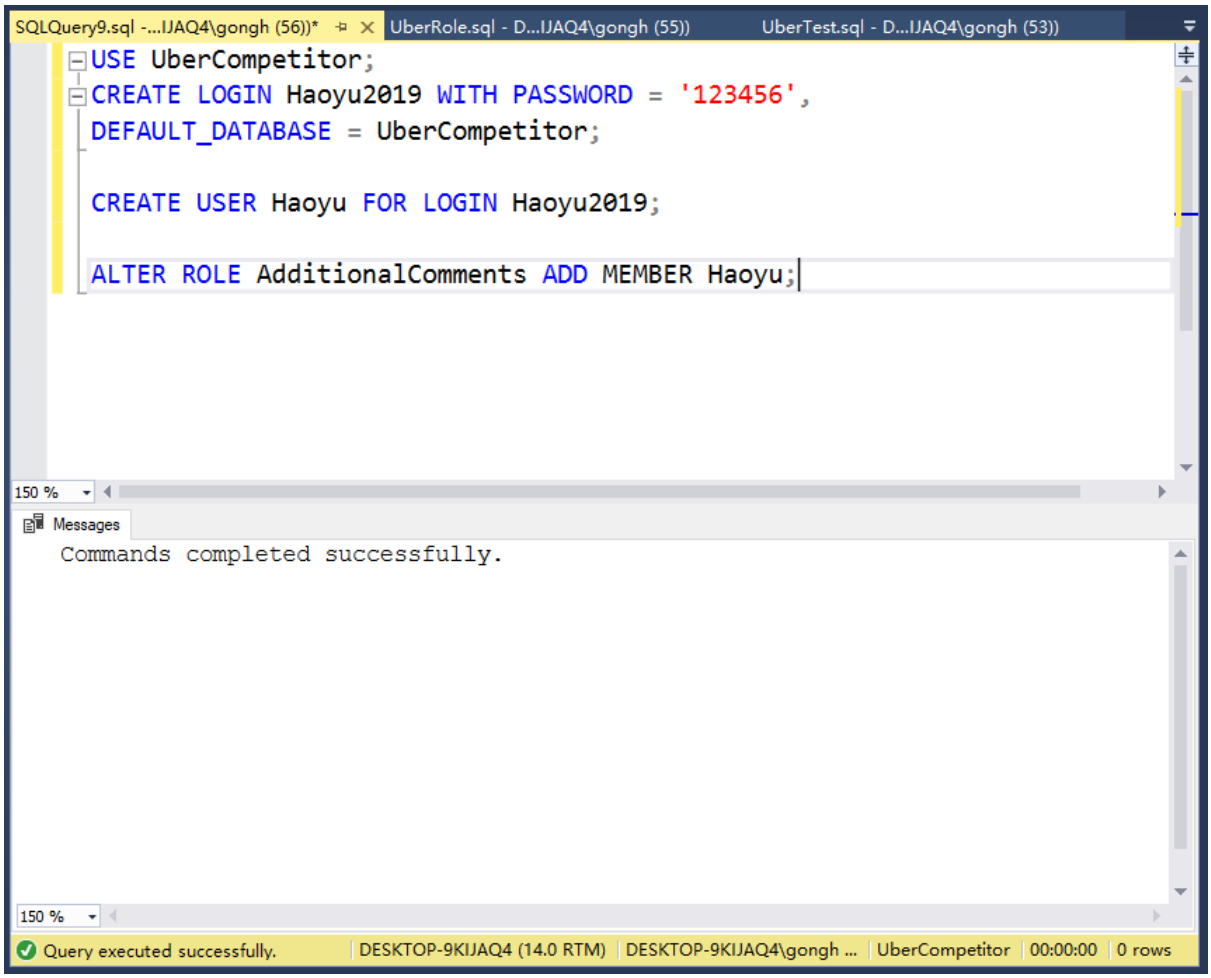



```
USE UberCompetitor;
GO
CREATE TRIGGER TripPickUpAddress_INSERT_UPDATE
    ON TripPickUpAddress
    AFTER INSERT, UPDATE
AS
    UPDATE TripPickUpAddress
    SET State = UPPER(State)
    WHERE PickUpID IN (SELECT PickUpID FROM Inserted);
```

Design – Security Levels

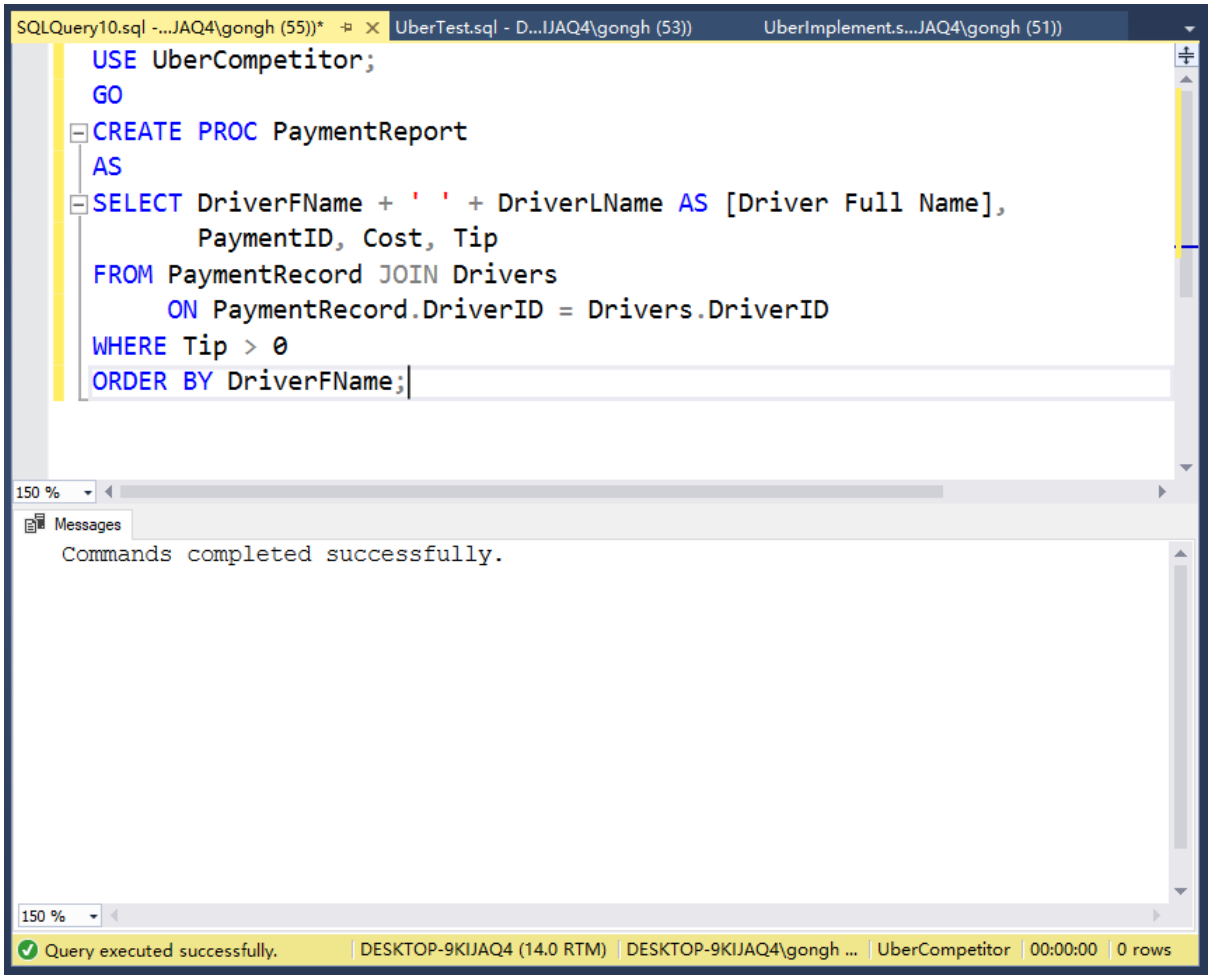



```sql
USE UberCompetitor;
CREATE ROLE AdditionalComments;

GRANT UPDATE
ON CustomerDetail
TO AdditionalComments;

GRANT INSERT,UPDATE
ON CustomerRating
TO AdditionalComments;

ALTER ROLE db_datareader ADD MEMBER AdditionalComments;
```



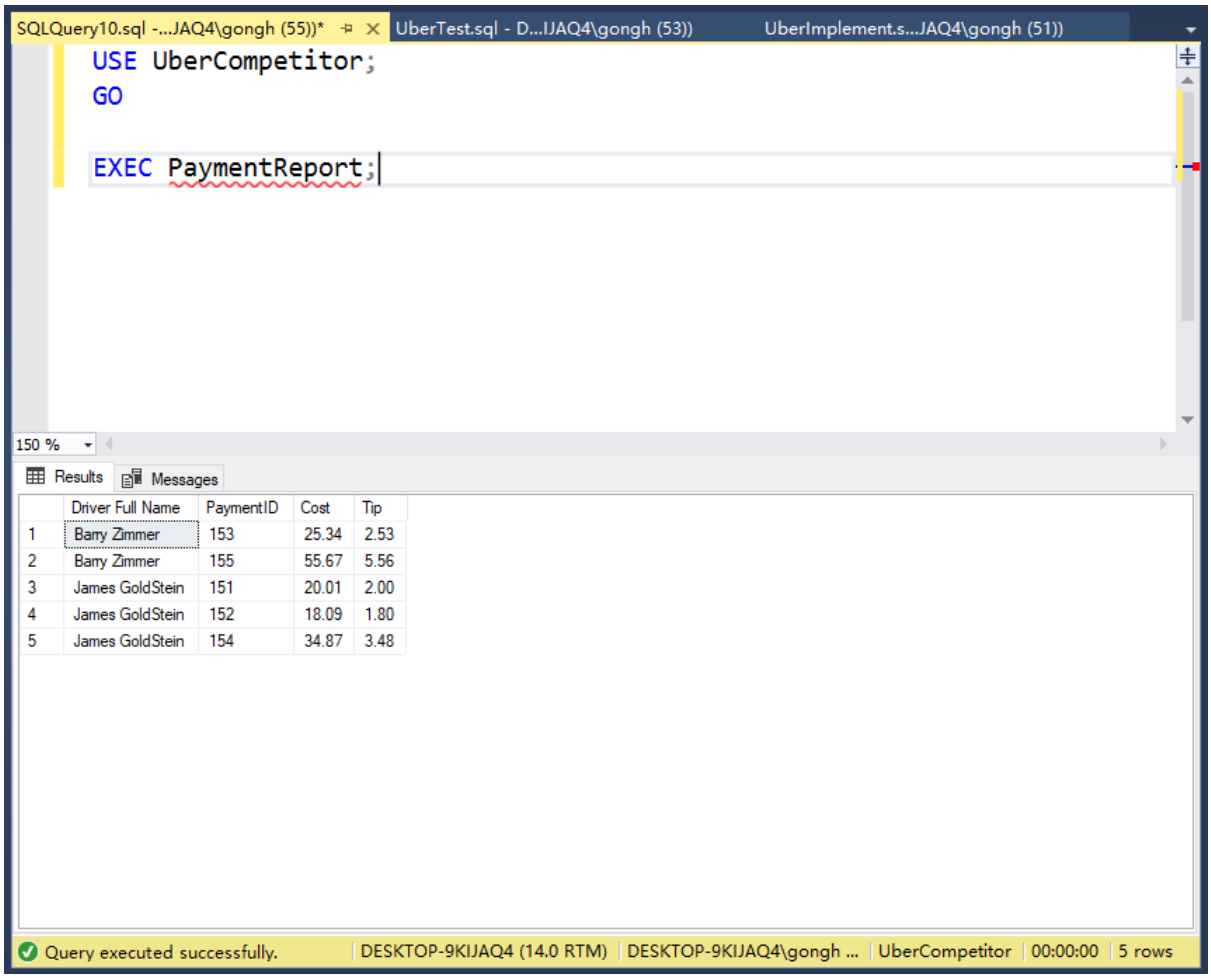

```sql
USE UberCompetitor;
CREATE LOGIN Haoyu2019 WITH PASSWORD = '123456',
DEFAULT_DATABASE = UberCompetitor;

CREATE USER Haoyu FOR LOGIN Haoyu2019;

ALTER ROLE AdditionalComments ADD MEMBER Haoyu;
```

Design – Performance and efficiency

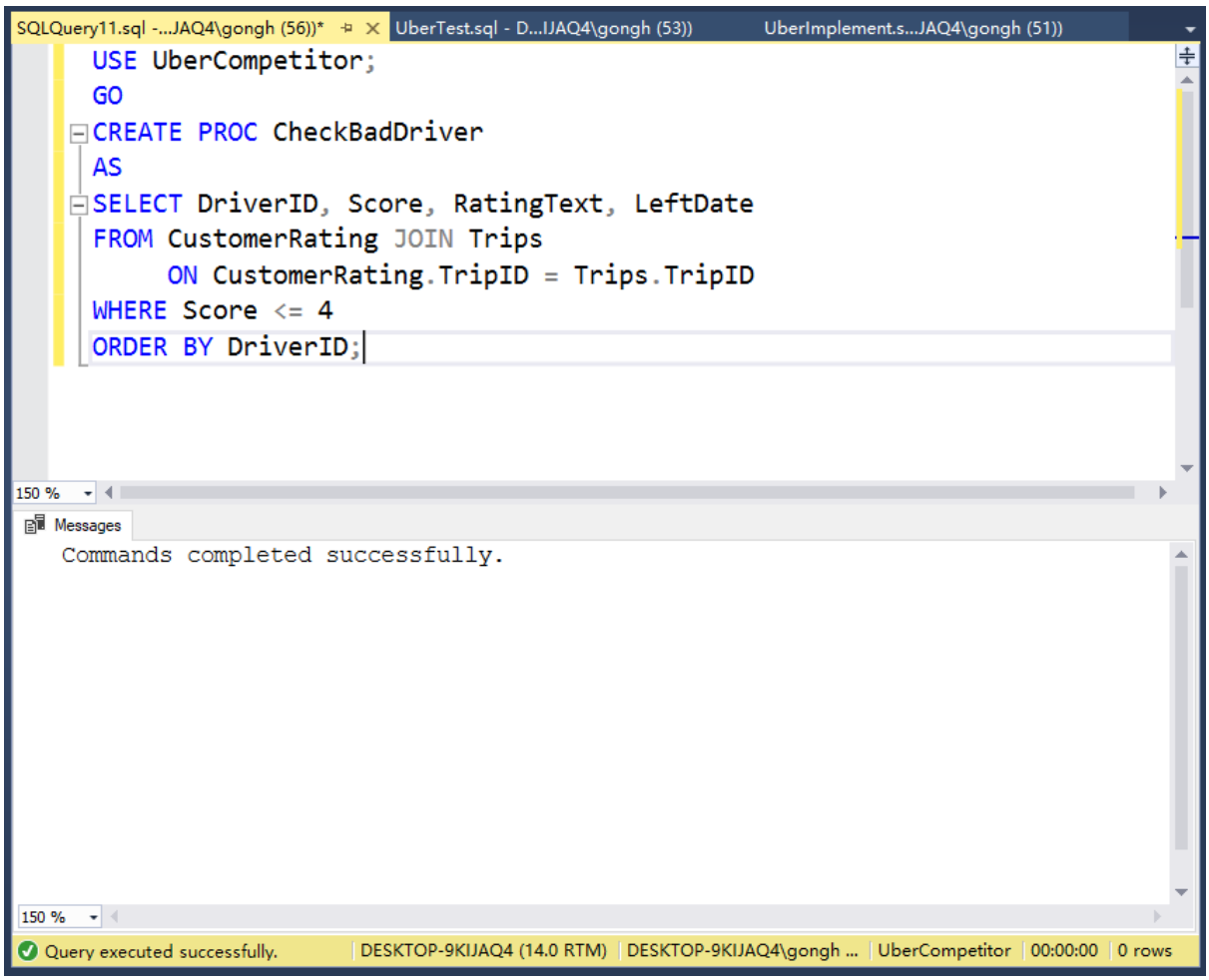



```sql
USE UberCompetitor;
GO
CREATE PROC PaymentReport
AS
SELECT DriverFName + ' ' + DriverLName AS [Driver Full Name],
      PaymentID, Cost, Tip
FROM PaymentRecord JOIN Drivers
    ON PaymentRecord.DriverID = Drivers.DriverID
WHERE Tip > 0
ORDER BY DriverFName;
```

This SP will show those record paid tips.

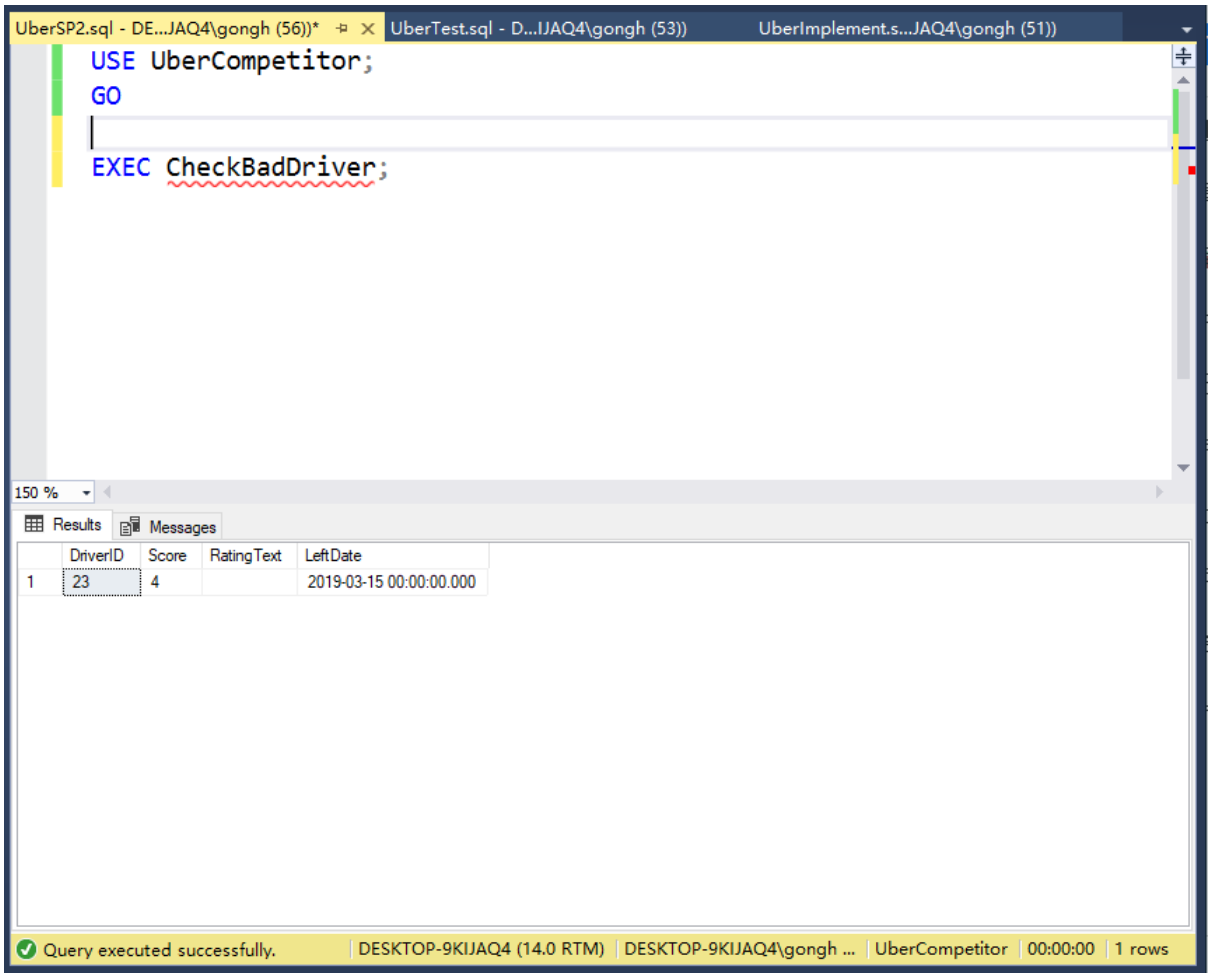


The results are above.

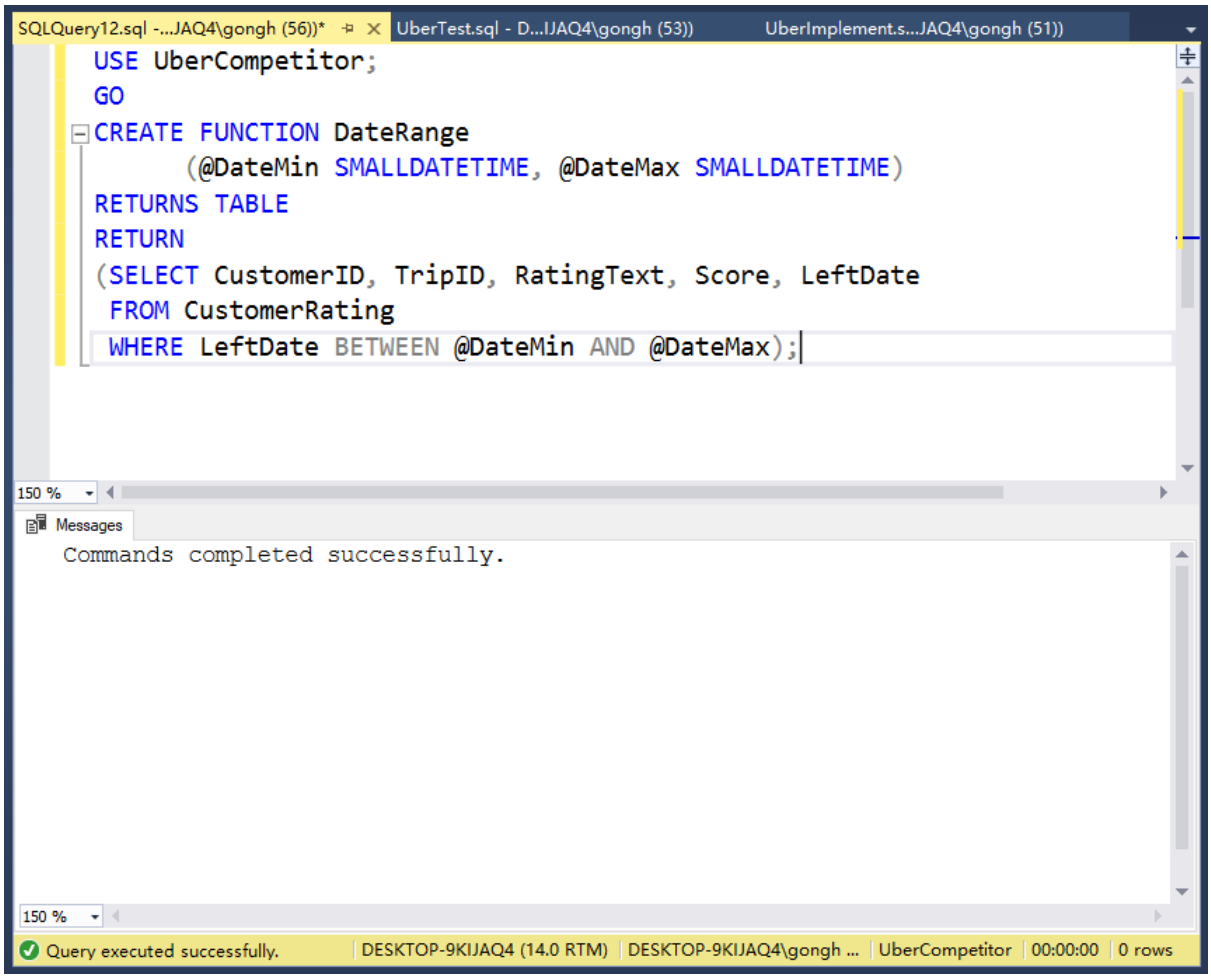



```sql
USE UberCompetitor;
GO
CREATE PROC CheckBadDriver
AS
SELECT DriverID, Score, RatingText, LeftDate
FROM CustomerRating JOIN Trips
    ON CustomerRating.TripID = Trips.TripID
WHERE Score <= 4
ORDER BY DriverID;
```

This SP will show the low score driver.

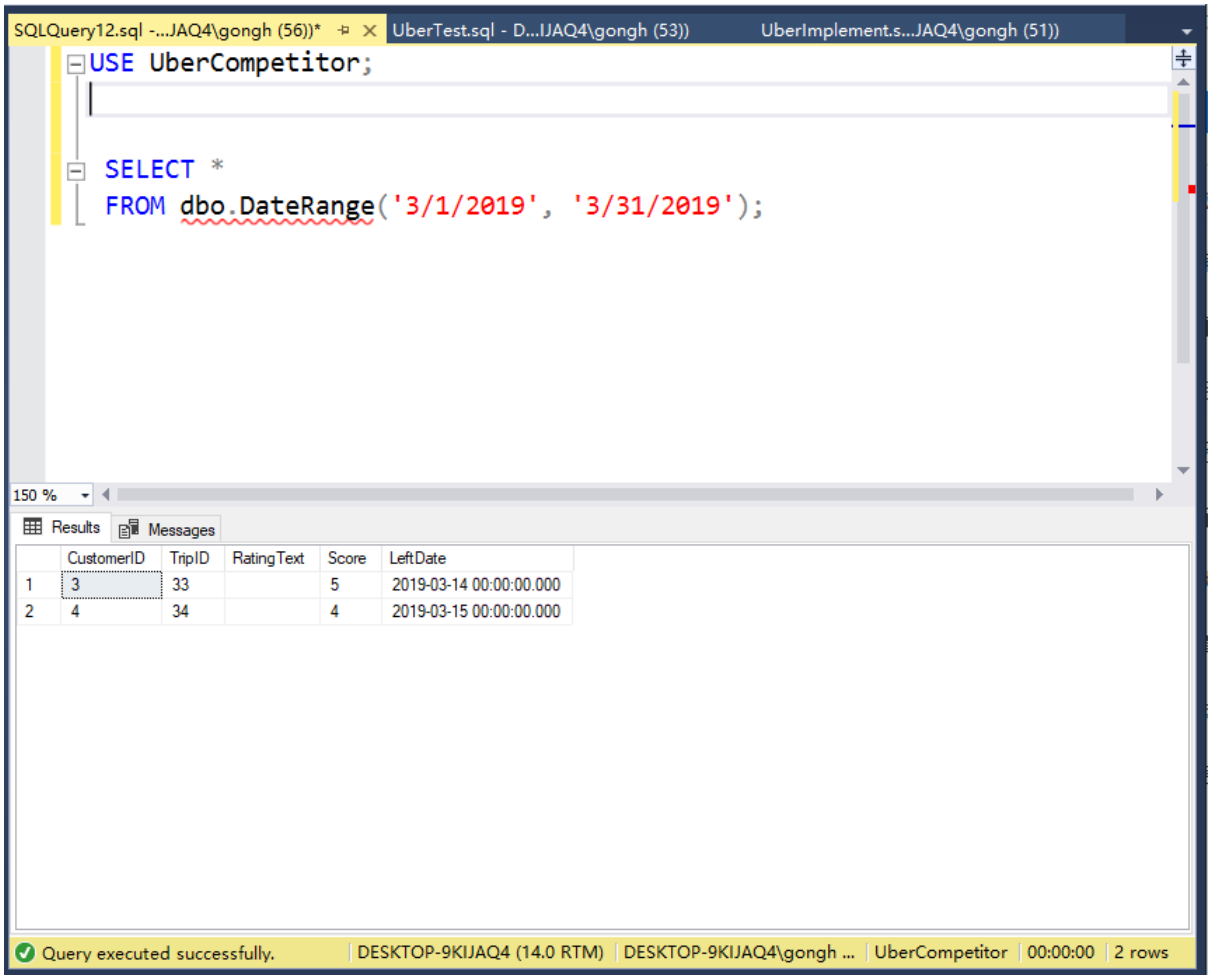



```sql
USE UberCompetitor;
GO

EXEC CheckBadDriver;
```

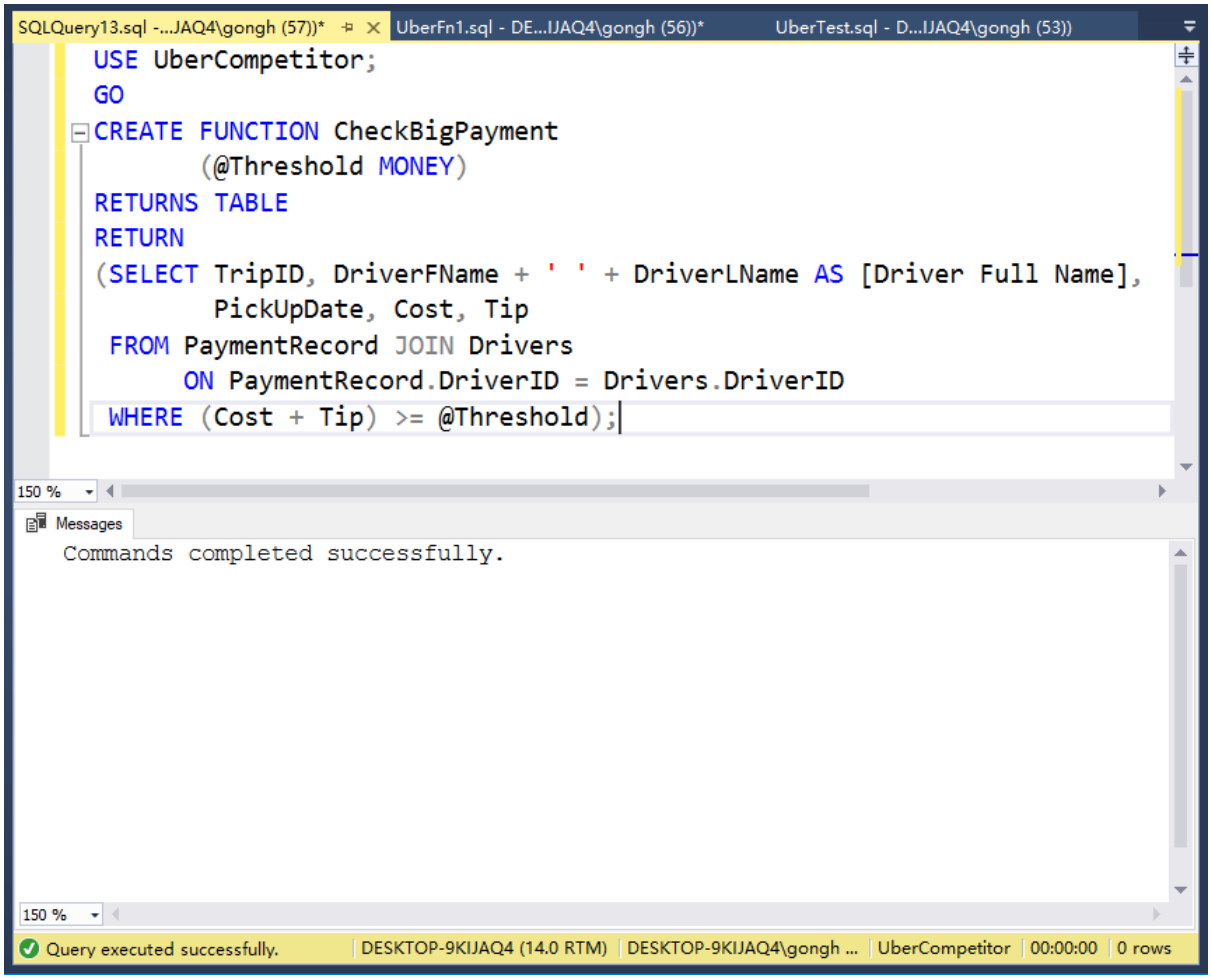



```
USE UberCompetitor;
GO
CREATE FUNCTION DateRange
      (@DateMin SMALLDATETIME, @DateMax SMALLDATETIME)
RETURNS TABLE
RETURN
(SELECT CustomerID, TripID, RatingText, Score, LeftDate
 FROM CustomerRating
 WHERE LeftDate BETWEEN @DateMin AND @DateMax);
```

This function will find the trip record in the particular range we want.

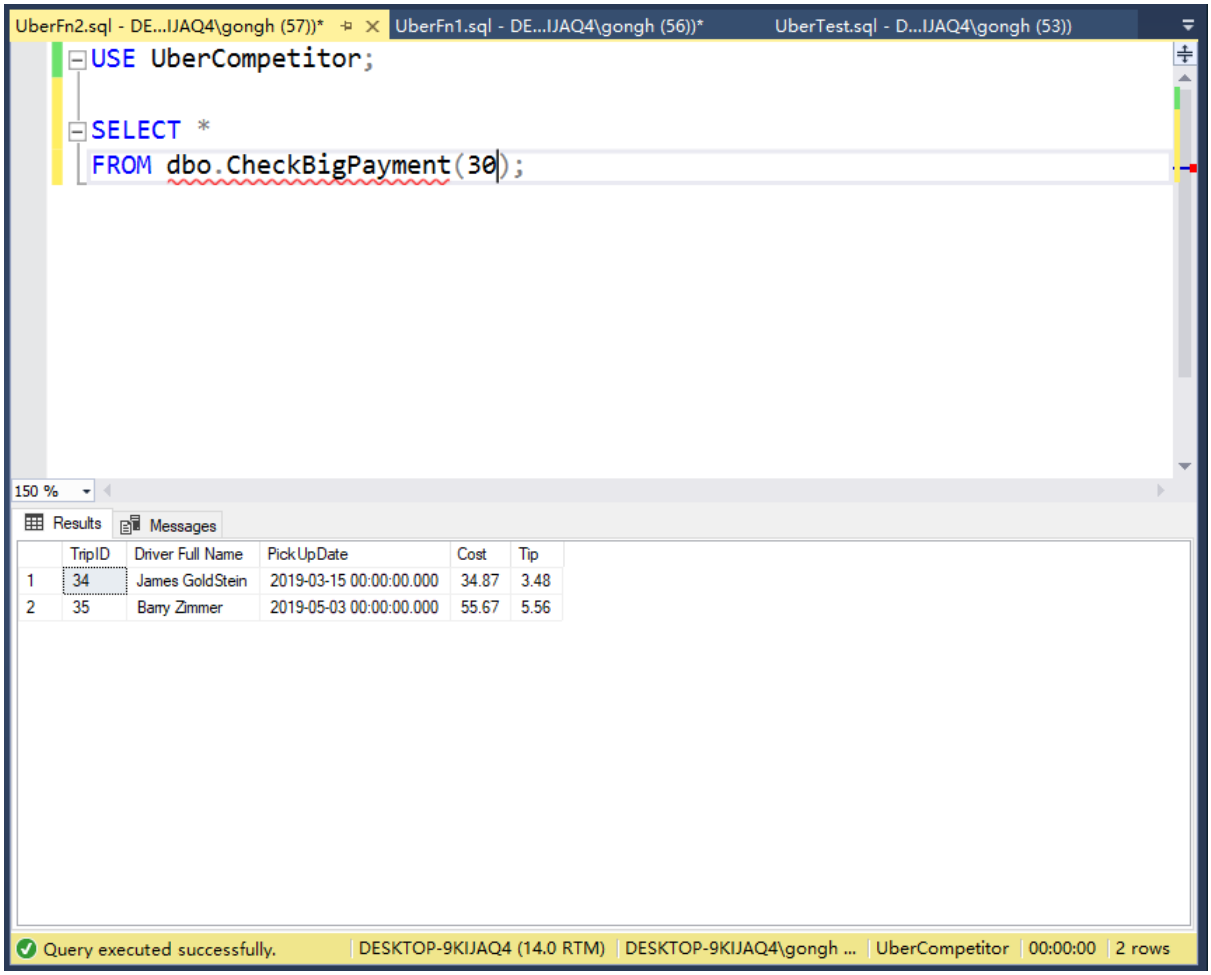



```sql
USE UberCompetitor;


SELECT *
FROM dbo.DateRange('3/1/2019', '3/31/2019');
```




```sql
USE UberCompetitor;
GO
CREATE FUNCTION CheckBigPayment
       (@Threshold MONEY)
RETURNS TABLE
RETURN
(SELECT TripID, DriverFName + ' ' + DriverLName AS [Driver Full Name],
        PickUpDate, Cost, Tip
 FROM PaymentRecord JOIN Drivers
      ON PaymentRecord.DriverID = Drivers.DriverID
 WHERE (Cost + Tip) >= @Threshold);
```

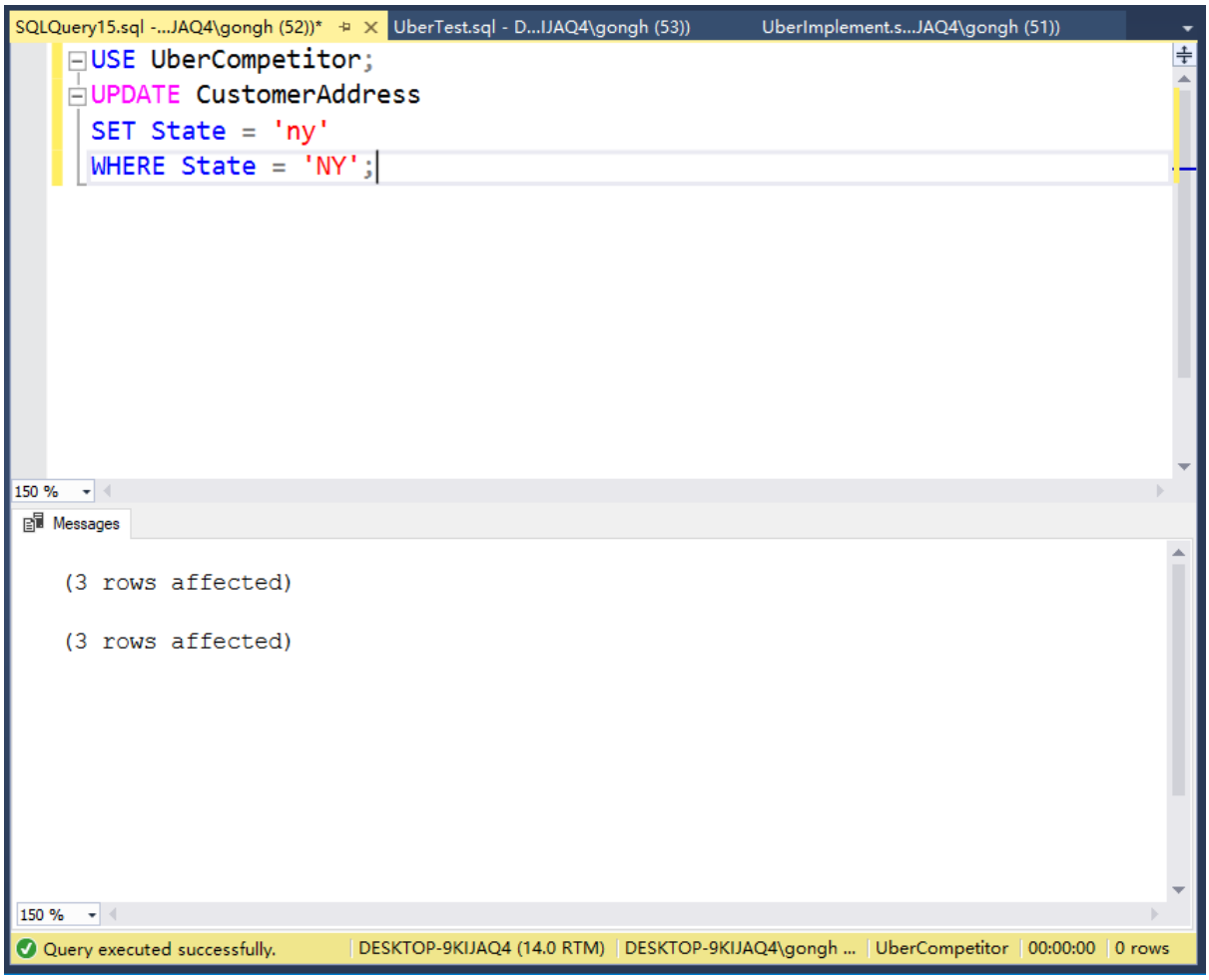
This function will find those total payment bigger or equal with the amount we want.

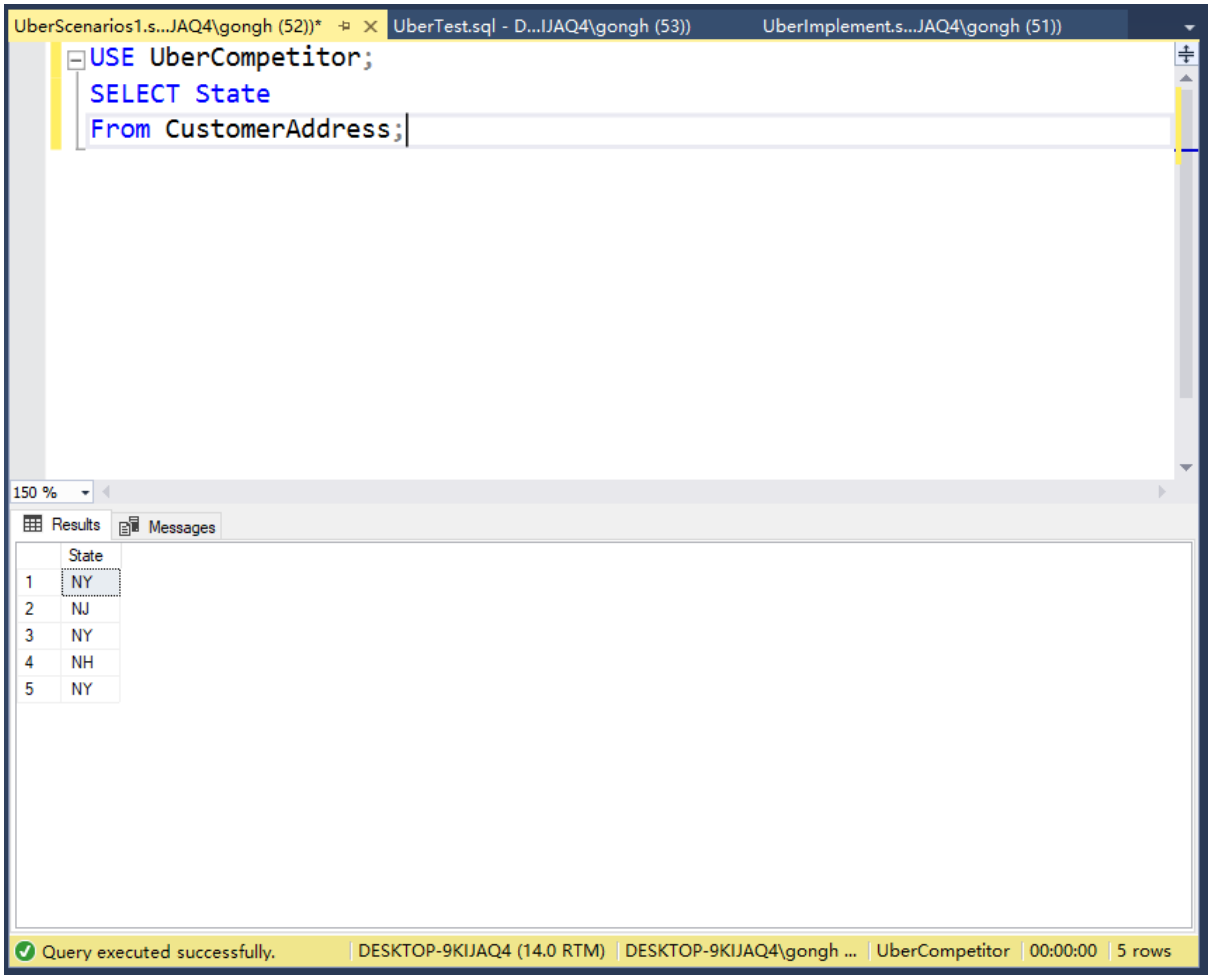



```sql
USE UberCompetitor;

SELECT *
FROM dbo.CheckBigPayment(30);
```

Testing – Scenarios

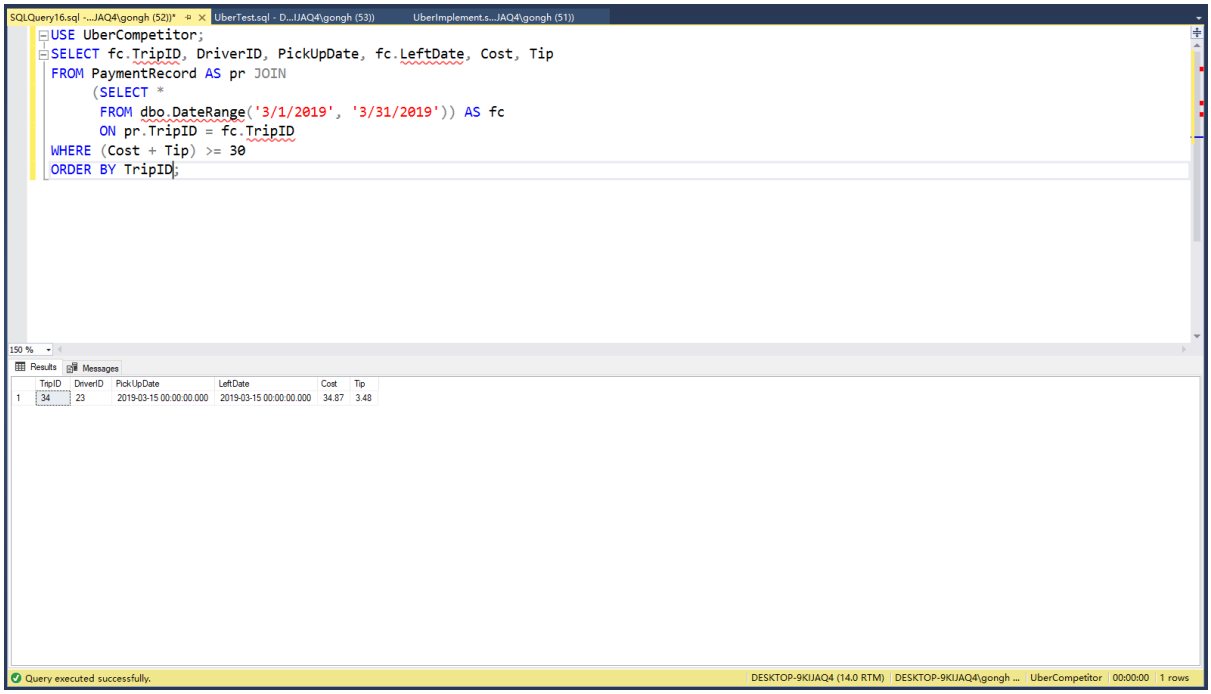
1. Change the 'NY' into 'ny' on purpose, see what's going to happen.(Try the CustomerAddress_INSERT_UPDATE trigger)






```sql
USE UberCompetitor;
SELECT State
From CustomerAddress;
```

2. Find those total payment is bigger or equal to 30, from those trip happened in the March (3.1~3.31). Testing the function 'DateRange'




```sql
USE UberCompetitor;
SELECT fc.TripID, DriverID, PickUpDate, fc.LeftDate, Cost, Tip
FROM PaymentRecord AS pr JOIN
     (SELECT *
      FROM dbo.DateRange('3/1/2019', '3/31/2019')) AS fc
      ON pr.TripID = fc.TripID
WHERE (Cost + Tip) >= 30
ORDER BY TripID;
```

Conclusion – Analysis and remarks

In this project, I finished all the necessary parts, such as designing, coding, and testing. All those queries are fully functional.

All the things of this database are right, but it doesn't mean that it is perfect. Obviously, it is just a simple structure. The real database of Uber is definitely more complex than this. I am happy I finished it, and it worked. However, several places deserve to be changed.

a. Those time should be more specific to hour : minute : second
b. The scenarios of testing part are kind of easy, especially the first one. But to be honest, those are all I could think of.
c. E/R diagram. Even though I make the distance larger, it is still not easy to understand. But I have an excel version about it, PKs and FKs.