PROJECT 02:

UBER COMPETITOR DATABASE

CSE 581: Introduction To Database Management Systems

Date of Submission: 03 May, 2019

By: Sakshi Sanjay Salokhe

SUID: 201026946

Email: ssalokhe@syr.edu

Contents:

Abstract:

In the fast moving world, sometimes, 24 hours of a day fall short. Life has become faster and every minute counts. This gave rise to online booking of cabs. People who do not have cars can book their rides sitting back home and move out just when the driver arrives. For those who own their private vehicles, do not have to worry about parking if they book a cab. This idea of online booking of cabs requires a solid database to operate. When a passenger books a ride, the request has to be sent to all the nearby drivers. The drivers will respond to the request and that ride gets booked.
Before these higher level activities occur, the basic operation here is to store all the data. The driver and the rider, both need to have an account so that we can store the details and their personal information. Apart from the personal information, we need to keep track of certain information in real time as well for the ongoing rides. After this, we also have to take care of the history for various purposes like if someone loses any personal item in the car, or if someone has had a bad experience and wants to lodge a complaint. So maintaining the historical information is also a must. Also, we need to keep a track of the payment details which is highly confidential and cannot be compromised. And other most important thing that should be taken care of is the ratings given by the driver to the customer and vice versa. Database is needed for all these activities that take place in a simple cab booking service. There is data that has to be stored until a person wants to delete his account, data that keeps updating in real time, data that changes on a daily basis and so on.
Therefore, in this project we will be creating an online cab booking service named “IRride” that is a competitor to the Uber service.

Introduction:

IRide is an online cab booking application like uber. In this project, we develop a database for the cab booking application. The database is created so as to store the information about a customer, driver, the car information, ratings, trip history, etc.
The project contains queries so as to create the database for an online cab booking application. We write queries to create the database, to insert and populate the database and to create views and scripts for testing the data.
The database contains tables to store data related to customers, drivers, cars and the trips. However, there are various other tables related to these to break down the tables to their atomic stages and move the database to the 3rd normal form so as to avoid any sort of confusion and dependencies.
We try to divide the tables involved in the database to the lowest possible division so that the values needed can be accessed using foreign keys and it becomes easier for us to access the information as we need it at the particular time.
Apart from the normal storage of data, the database will be designed in a way that it calculates the desired values based on the values needed from other dependent tables and update the table as needed.
Thus the project focuses on developing a database for a cab booking service that is a competitor of uber where we create the tables that are needed to store all the necessary information of all the basic entities involved.

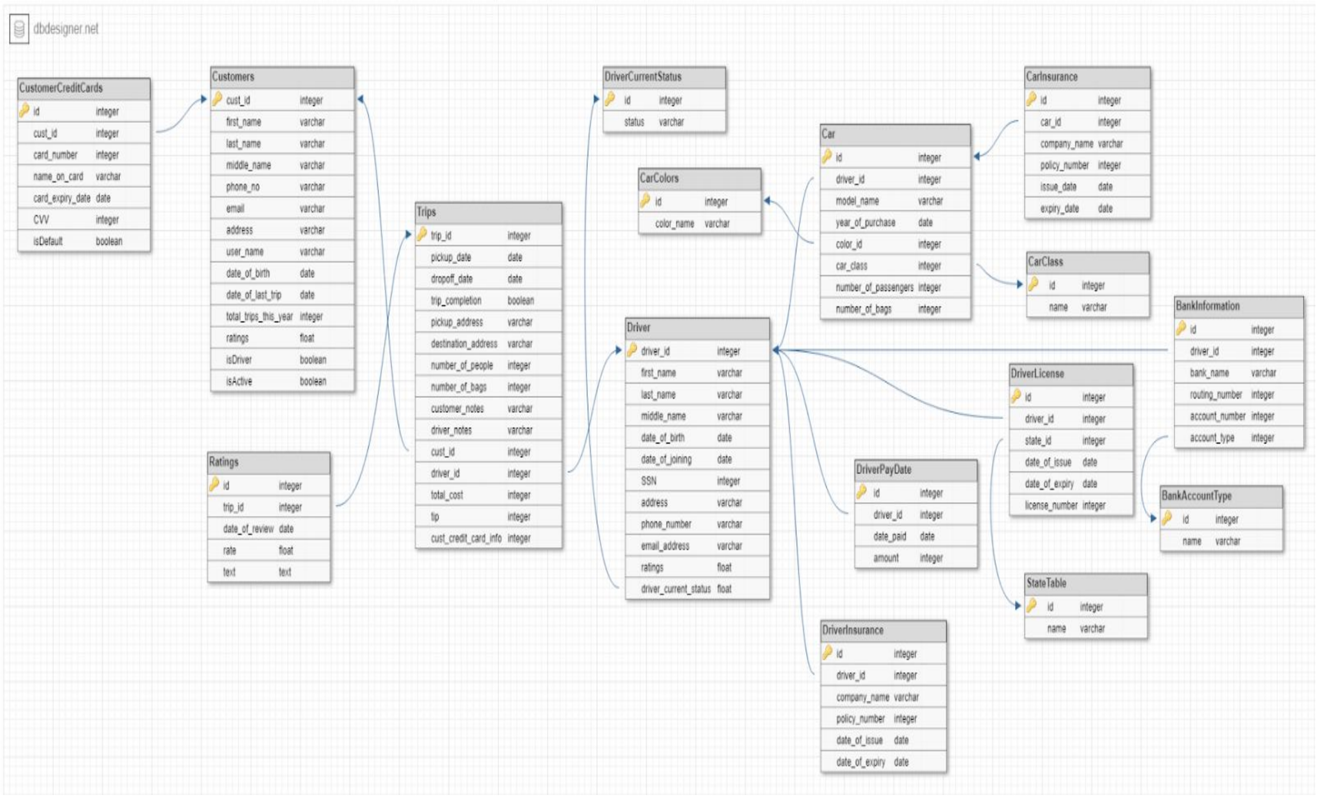
Description:

The tables that will be involved in the database are as follows:

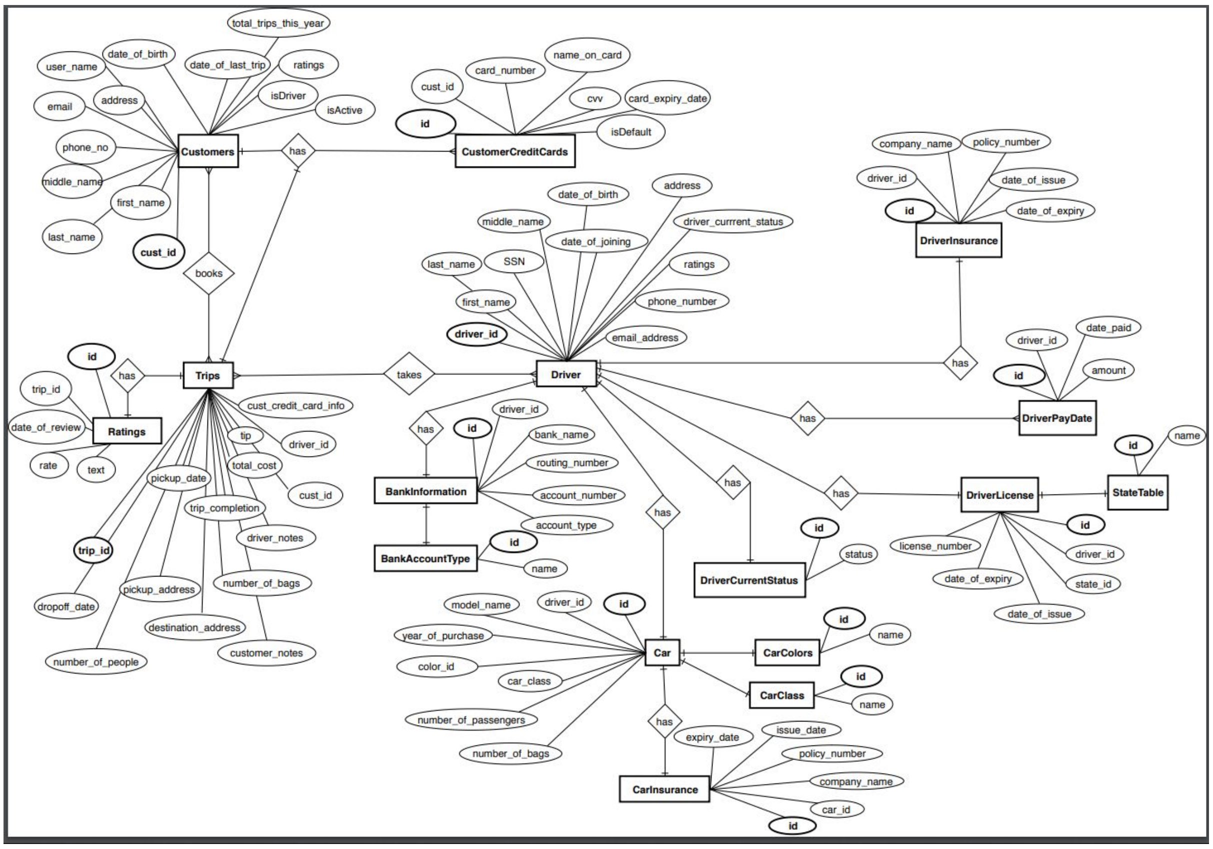
1. **Customers** table stores the personal information of the customers like first name, last name, phone number, address, email address, username, date of birth, isDriver (to check if the customer also works as a driver for the company), date of the last trip taken, average of all ratings till date given by the drivers, payment details, number of trips taken in the current year, if the customer has an active account,etc.
2. Similarly to reduce the tables and get the database into 3rd normal form, we store the credit card information of the customers into another table named **CustomerCreditCards**.
3. **Driver** table stores the personal details of a driver in a way similar to the Customers. We also store the SSN here as we need to have a credit check on a person, before hiring him.
4. When we send the notification about a new ride request to all the nearby drivers, we need to know which drivers are currently online. Due to such reasons, we need to store the current status of the drivers. Therefore we create a new table called **DriverCurrentStatus.** Then we have **DriverLicense** table to store the details of the drivers license, **DriverInsurance** table to store details of the drivers insurance. Since we have to pay the drivers, we maintain the details of their bank account to which we make the deposit. We store these details in **BankInformation** table. We have **DriverPayDate** table where we store the date of payment of each driver and the amount paid. All these tables have the foreign key as **driver_id** so that we know the details of the particular driver.
5. We have **State_Table** that stores the names of states based on the primary keys. Similarly we keep a table for **BankAccountType** to store if the account is checking or savings.
6. We have a table called **Car** for car information like model, year of purchase, color,car class, number of passengers that it can hold, number of bags that can fit in it.
7. We maintain tables to break down the **Cars** table into 3rd normal form. These include **CarColor, CarInsurance, CarClass.**
8. We have a table **Trips** to store the trip information where we store the driver id, the customer id,pickup location, drop off location, pick up time, drop off time,number of people, number of bags, notes if any, total cost, tips,etc.
9. Lastly, we have a **Ratings** table to store the ratings given by a driver to the passenger and vice versa. We do this for each trip therefore, we have **trip_id** as the foreign key.

These are some of the tables that will be included in the database for the uber competitor.

Database Design Diagram:



ER Diagram:


Customers
CustomerCreditCards
Trips
Ratings
Driver
BankInformation
BankAccountType
DriverInsurance
DriverPayDate
DriverLicense
StateTable
DriverCurrentStatus
Car
CarColors
CarClass
CarInsurance
has
books
takes

ER Diagram Entities

1. **Customers**

Column Name	**Datatype**	**Allow Nulls?**
cust_id	Integer (Primary Key)	No
first_name	varchar(255)	No
last_name	varchar(255)	No
middle_name	varchar(255)	Yes
phone_no	Varchar (10)	No
email	varchar(255)	Yes
address	varchar(255)	No
user_name	varchar(255)	No
date_of_birth	date	No
date_of_last_trip	date	No
total_trips_this_year	Integer	No
ratings	float	No
isDriver	bit	No
isActive	bit	No

2. **CustomerCreditCards**

Column Name	**Datatype**	**Allow Nulls?**
id	Integer (Primary Key)	No
cust_id	Integer (Foreign Key)	No
card_number	Varchar (16)	No
name_on_card	Varchar (255)	No
card_expiry_date	date	No
CVV	Int	No
isDefault	bit	No

3. **Trips**

Column Name	**Datatype**	**Allow Nulls?**
trip_id	Integer (Primary Key)	No
pickup_date	date	No
dropoff_date	date	No
trip_completion	Bit	No
pickup_address	varchar(255)	No
destination_address	varchar(255)	No
number_of_people	Int	No
number_of_bags	Int	No
customer_notes	varchar(255)	Yes
driver_notes	varchar(255)	Yes
cust_id	Integer	No
driver_id	Integer	No
total_cost	money	No
tip	money	No
cust_credit_card_info	Int	No

4. **Ratings**

Column Name	**Datatype**	**Allow Nulls?**
id	Integer (Primary Key)	No
trip_id	Integer	No
date_of_review	date	No
rate	float	No
text	varchar(255)	Yes

5. **Driver**

Column Name	**Datatype**	**Allow Nulls?**
driver_id	Integer (Primary Key)	No
first_name	varchar(255)	No
last_name	varchar(255)	No
middle_name	varchar(255)	Yes
date_of_birth	date	No
date_of_joining	date	No
SSN	BigInt	No
address	varchar(255)	No
phone_number	Varchar (10)	No
email_address	varchar(255)	No
ratings	float	No
driver_current_status	Int	No

6. **DriverInsurance**

Column Name	**Datatype**	**Allow Nulls?**
id	Integer (Primary Key)	No
driver_id	Integer (Foreign Key)	No
company_name	varchar(255)	No
policy_number	BigInt	No
date_of_issue	date	No
date_of_expiry	date	No

7. **DriverCurrentStatus**

Column Name	**Datatype**	**Allow Nulls?**
id	Integer (Primary Key)	No
status	varchar(255)	No

8. DriverLicense

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
driver_id	Integer (Foreign Key)	No
state_id	Integer (Foreign Key)	No
date_of_issue	date	No
date_of_expiry	date	No
license_number	BigInt	No

9. StateTable

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
name	varchar(255)	No

10. DriverPayDate

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
driver_id	Integer (Foreign Key)	No
date_paid	date	No
amount	money	No

11. BankInformation

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
driver_id	Integer (Foreign Key)	No
bank_name	varchar(255)	No
routing_number	BigInt	No
account_number	BigInt	No
account_type	Int (Foreign Key)	No

12. BankAccountType

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
name	varchar(255)	No

13. Car

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
driver_id	Integer (Foreign Key)	No
model_name	varchar(255)	No
year_of_purchase	smallInt	No
color_id	Integer (Foreign Key)	No
car_class	Integer (Foreign Key)	No
number_of_passengers	Int	No
number_of_bags	Int	No

14. CarInsurance

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
car_id	Integer (Foreign Key)	No
company_name	varchar(255)	No
policy_number	BigInt	No
issue_date	date	No
expiry_date	date	No

15. CarColors

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
color_name	varchar(255)	No

16. CarClass

Column Name	Datatype	Allow Nulls?
id	Integer (Primary Key)	No
name	varchar(255)	No

Implementation

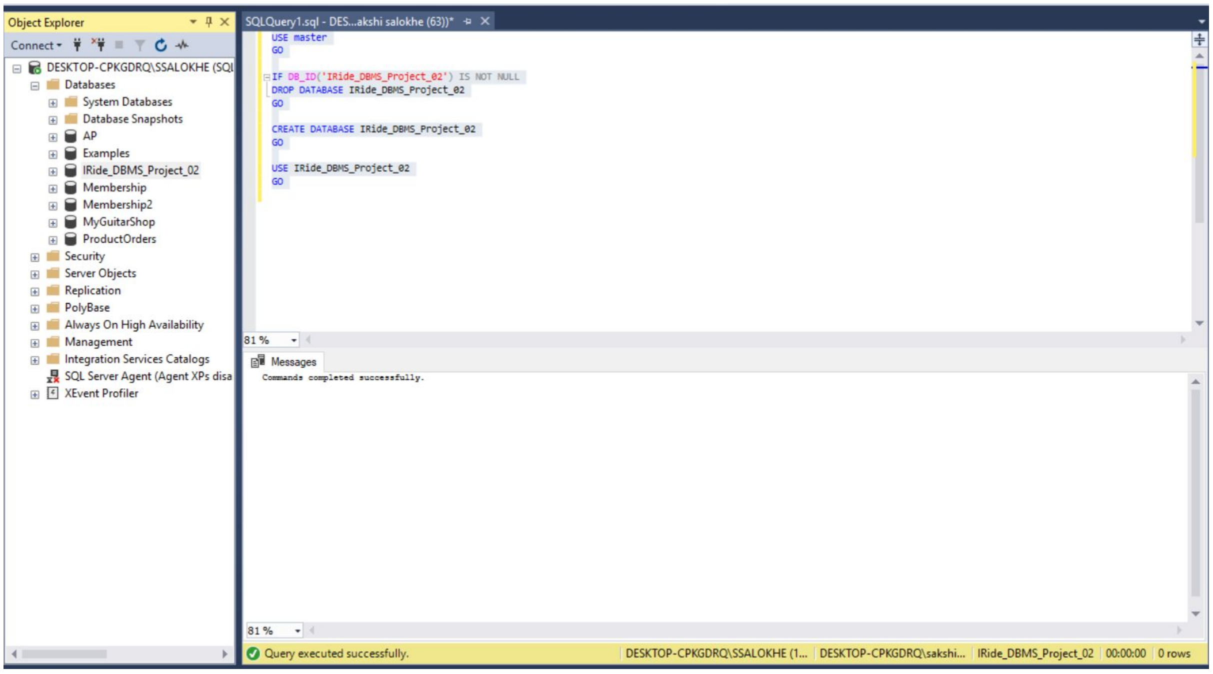
Creating Database:

USE master
GO

IF DB_ID('IRide_DBMS_Project_02') IS NOT NULL
DROP DATABASE IRide_DBMS_Project_02
GO

CREATE DATABASE IRide_DBMS_Project_02
GO

USE IRide_DBMS_Project_02
GO



```
USE master
GO

IF DB_ID('IRide_DBMS_Project_02') IS NOT NULL
DROP DATABASE IRide_DBMS_Project_02
GO

CREATE DATABASE IRide_DBMS_Project_02
GO

USE IRide_DBMS_Project_02
GO
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows

Creating Tables in the new database

```
CREATE TABLE Customers(
        cust_id int IDENTITY,
        first_name varchar(255) NOT NULL,
        last_name varchar(255) NOT NULL,
        middle_name varchar(255) NULL,
        phone_no varchar(10) NOT NULL,
        email varchar(255) NULL,
        address varchar(255) NOT NULL,
        user_name varchar(255) NOT NULL,
        date_of_birth date NOT NULL,
        date_of_last_trip date NOT NULL,
        total_trips_this_year int NOT NULL,
        ratings float NOT NULL,
        isDriver bit Not Null,
        isActive bit NOT NULL,

        PRIMARY KEY (cust_id)
);
GO

CREATE TABLE CustomerCreditCards(
        id int IDENTITY,
        cust_id int NOT NULL,
        card_number varchar(16) NOT NULL,
        name_on_card varchar(255) NOT NULL,
        card_expiry_date date NOT NULL,
        CVV Int NOT NULL,
        isDefault bit Not Null,

        PRIMARY KEY(id),

        FOREIGN KEY(cust_id)
        REFERENCES Customers(cust_id),
);
GO

CREATE TABLE DriverCurrentStatus(
        id int IDENTITY,
        status varchar(255) NOT NULL,

        PRIMARY KEY (id),
);
```

```
GO

CREATE TABLE Driver(
        driver_id int IDENTITY,
        first_name varchar(255) NOT NULL,
        last_name varchar(255) NOT NULL,
        middle_name varchar(255) NULL,
        date_of_birth date NOT NULL,
        date_of_joining date NOT NULL,
        SSN BigInt NOT NULL,
        address varchar(255) NOT NULL,
        phone_number varchar(10) NOT NULL,
        email_address varchar(255) NULL,
        ratings float NOT NULL,
        driver_current_status Int Not Null,

        PRIMARY KEY (driver_id),
        FOREIGN KEY(driver_current_status)
        REFERENCES DriverCurrentStatus(id),
);
GO

CREATE TABLE Trips(
        trip_id int IDENTITY,
        pickup_date date NOT NULL,
        dropoff_date date NOT NULL,
        trip_completion bit NOT NULL,
        pickup_address varchar(255) NOT NULL,
        destination_address varchar(255) NOT NULL,
        number_of_people Int NOT NULL,
        number_of_bags Int NOT NULL,
        customer_notes varchar(255) NULL,
        driver_notes varchar(255) NULL,
        cust_id Int NOT NULL,
        driver_id Int NOT NULL,
        total_cost money NOT NULL,
        tip money NOT NULL,
        cust_credit_card_info Int NOT NULL,
        PRIMARY KEY(trip_id),
        FOREIGN KEY(cust_id)
        REFERENCES Customers(cust_id),
        FOREIGN KEY(driver_id)
        REFERENCES Driver(driver_id),

);
GO
```

```
CREATE TABLE Ratings(
        id int IDENTITY,
        trip_id int NOT NULL,
        date_of_review date NOT NULL,
        rate float NOT NULL,
        text varchar(255) NULL,

        PRIMARY KEY(id),

        FOREIGN KEY(trip_id)
        REFERENCES Trips(trip_id), /* take card infor for the cust_id with isDefault = 1*/
);
GO

CREATE TABLE DriverInsurance(
        id int IDENTITY,
        driver_id int NOT NULL,
        company_name varchar(255) NOT NULL,
        policy_number BigInt NOT NULL,
        date_of_issue date NOT NULL,
        date_of_expiry date NOT NULL,

        PRIMARY KEY(id),

        FOREIGN KEY(driver_id)
        REFERENCES Driver(driver_id),

);
GO

CREATE TABLE DriverPayDate(
        id int IDENTITY,
        driver_id int NOT NULL,
        date_paid date NOT NULL,
        amount money NOT NULL,

        PRIMARY KEY(id),

        FOREIGN KEY(driver_id)
        REFERENCES Driver(driver_id),

);
GO
```

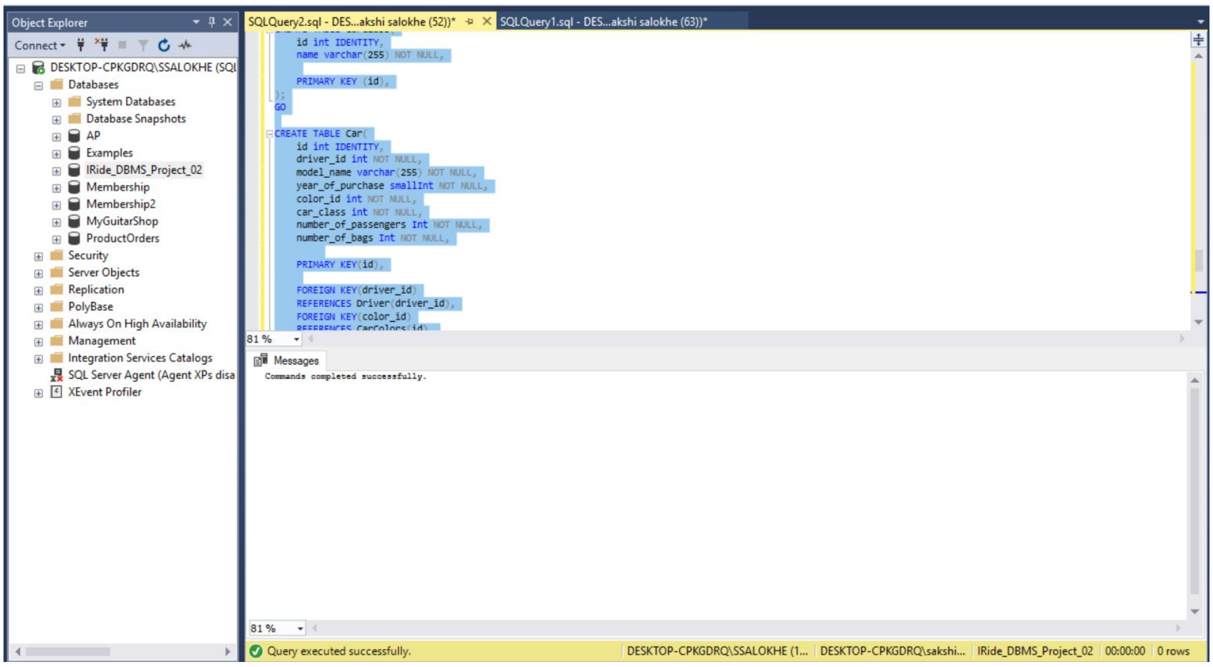
```
CREATE TABLE StateTable(
        id int IDENTITY,
        name varchar(255) NOT NULL,

        PRIMARY KEY (id),
);
GO

CREATE TABLE DriverLicense(
        id int IDENTITY,
        driver_id int NOT NULL,
        state_id int NOT NULL,
        license_number BigInt NOT NULL,
        date_of_issue date NOT NULL,
        date_of_expiry date NOT NULL,

        PRIMARY KEY(id),

        FOREIGN KEY(driver_id)
        REFERENCES Driver(driver_id),
        FOREIGN KEY(state_id)
        REFERENCES StateTable(id),

);
GO

CREATE TABLE BankAccountType(
        id int IDENTITY,
        name varchar(255) NOT NULL,

        PRIMARY KEY (id),
);
GO

CREATE TABLE BankInformation(
        id int IDENTITY,
        driver_id int NOT NULL,
        bank_name varchar(255) NOT NULL,
        routing_number BigInt NOT NULL,
        account_number BigInt NOT NULL,
        account_type Int NOT NULL,

        PRIMARY KEY(id),

        FOREIGN KEY(driver_id)
        REFERENCES Driver(driver_id),
        FOREIGN KEY(account_type)
        REFERENCES BankAccountType(id),

);
GO

CREATE TABLE CarColors(
        id int IDENTITY,
        color_name varchar(255) NOT NULL,

        PRIMARY KEY (id),
);
GO

CREATE TABLE CarClass(
        id int IDENTITY,
        name varchar(255) NOT NULL,

        PRIMARY KEY (id),
);
GO

CREATE TABLE Car(
        id int IDENTITY,
        driver_id int NOT NULL,
        model_name varchar(255) NOT NULL,
        year_of_purchase smallInt NOT NULL,
        color_id int NOT NULL,
        car_class int NOT NULL,
        number_of_passengers Int NOT NULL,
        number_of_bags Int NOT NULL,

        PRIMARY KEY(id),

        FOREIGN KEY(driver_id)
        REFERENCES Driver(driver_id),
        FOREIGN KEY(color_id)
        REFERENCES CarColors(id),
        FOREIGN KEY(car_class)
        REFERENCES CarClass(id),

);
GO

CREATE TABLE CarInsurance(
        id int IDENTITY,
```

car_id **int NOT NULL**,
company_name **varchar(255) NOT NULL**,
policy_number **BigInt NOT NULL**,
date_of_issue **date NOT NULL**,
date_of_expiry **date NOT NULL**,

PRIMARY KEY(id),

FOREIGN KEY(car_id)
REFERENCES Car(id),

);
GO



```
    id int IDENTITY,
    name varchar(255) NOT NULL,

    PRIMARY KEY (id),
);
GO

CREATE TABLE Car(
    id int IDENTITY,
    driver_id int NOT NULL,
    model_name varchar(255) NOT NULL,
    year_of_purchase smallInt NOT NULL,
    color_id int NOT NULL,
    car_class int NOT NULL,
    number_of_passengers Int NOT NULL,
    number_of_bags Int NOT NULL,

    PRIMARY KEY(id),

    FOREIGN KEY(driver_id)
    REFERENCES Driver(driver_id),
    FOREIGN KEY(color_id)
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows

Inserting values in the tables of our database

```
INSERT INTO
Customers
(first_name,last_name,middle_name,phone_no,email,address,user_name,date_of_birth,date
_of_last_trip,total_trips_this_year,ratings,isDriver,isActive)
VALUES
('Sakshi Sanjay','Salokhe',",'3152438254','ssalokhe@syr.edu','105 Concord Pl, Syracuse,
NY', 'ssalokhe', '1995-11-19', '2019-04-21',0,5.0,0,1),

('Pushkar Mahendra','Tatiya',",'3153963628','pmtatiya@syr.edu','105 Concord Pl, Syracuse,
NY', 'pmtatiya', '1995-08-02', '2019-03-21',0,5.0,1,1),

('Shruti','Salokhe',",'3152438434','sshrutis@syr.edu','318 Westcott Street, Syracuse, NY',
'sshrutis', '2000-01-25', '2018-12-13',0,5.0,0,1),

('Vishnu','Menon','Kailas','8152438252','kmvishnu@gmail.com','105 Concord Pl, NJ',
'kmvishnu', '1992-12-19', '2019-04-21',0,5.0,1,1),

('Kruti','Kalmath',",'8004557895','kruti@gmail.com','New Brunswick, NY', 'kruti', '1994-05-19',
'2019-04-15',0,5.0,0,1),

('Chaitali','P','C','3152438000','cpulkund@syr.edu','111 Trinity Pl, Syracuse, NY', 'cpulkund',
'1995-09-18', '2019-04-02',0,5.0,0,1),

('Saurabh','Pohnerkar',",'3152336254','spohner@syr.edu','105 South Beach Street,
Syracuse, NY', 'saurabh', '1996-01-05', '2018-11-21',0,5.0,1,1),

('Souradeepta','Biswas',",'8155168254','soura@syr.edu','222 Broadway, NY', 'soura',
'1990-02-28', '2019-04-01',0,5.0,0,1),

('Siddharth','Kumar','S','3152008254','skumar@syr.edu','Frederick, Maryland', 'skumar',
'1995-06-19', '2019-03-11',0,5.0,0,1),

('Anushka Atul','Patil',",'3152448004','apatil@syr.edu','Boston, MA', 'apatil', '1997-09-04',
'2019-01-17',0,5.0,0,1);

INSERT INTO
CustomerCreditCards
(cust_id,card_number,name_on_card,card_expiry_date,CVV,isDefault)
VALUES
(1,'1234123412341234','Sakshi Salokhe','2022-11-01',123,1),

(1,'1234123412344321','Sakshi Salokhe', '2025-01-01',432,0),
```

(2,'2345234523452345','Pushkar Tatiya','2022-01-01',234,1),

(2,'2345234523455432','Pushkar M Tatiya','2024-01-01',543,0),

(2,'2345234554325432','Pushkar Tatiya','2018-12-01',503,0),

(3,'3456345634563456','Shruti Salokhe','2023-10-01',943,1),

(4,'4567456745674567','Vishnu KM','2024-05-21',643,1),

(5,'5678567856785678','Kruti Kalmath','2021-01-01',533,1),

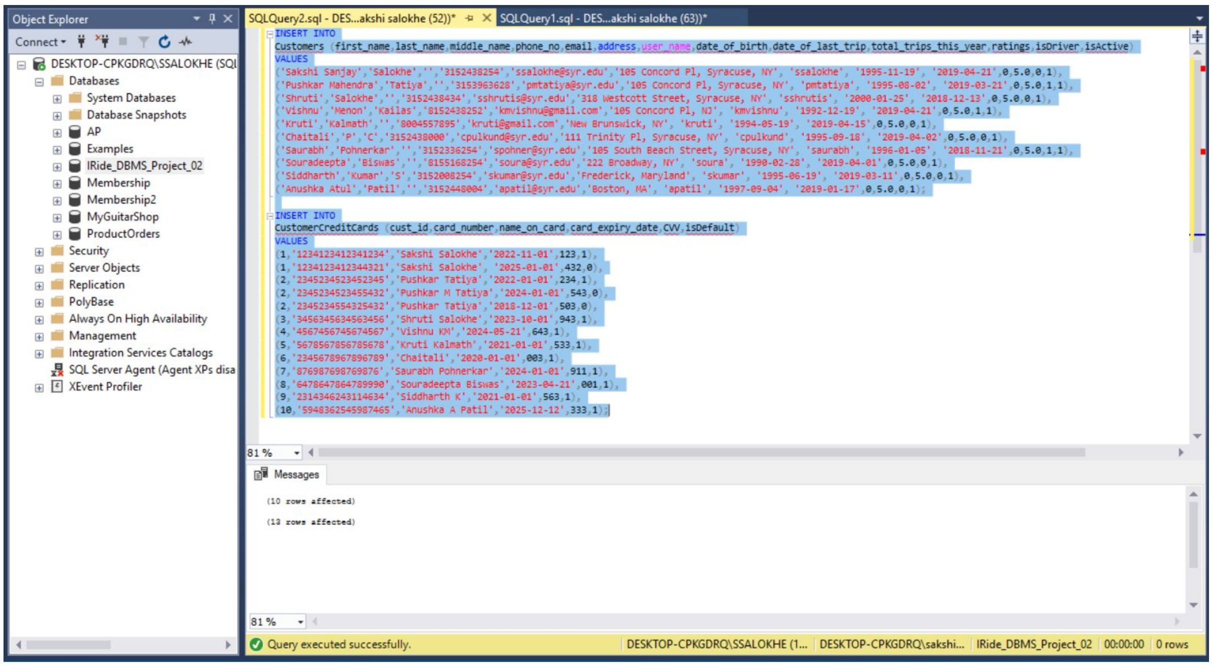
(6,'2345678967896789','Chaitali','2020-01-01',003,1),

(7,'876987698769876','Saurabh Pohnerkar','2024-01-01',911,1),

(8,'6478647864789990','Souradeepta Biswas','2023-04-21',001,1),

(9,'2314346243114634','Siddharth K','2021-01-01',563,1),

(10,'5948362545987465','Anushka A Patil','2025-12-12',333,1);

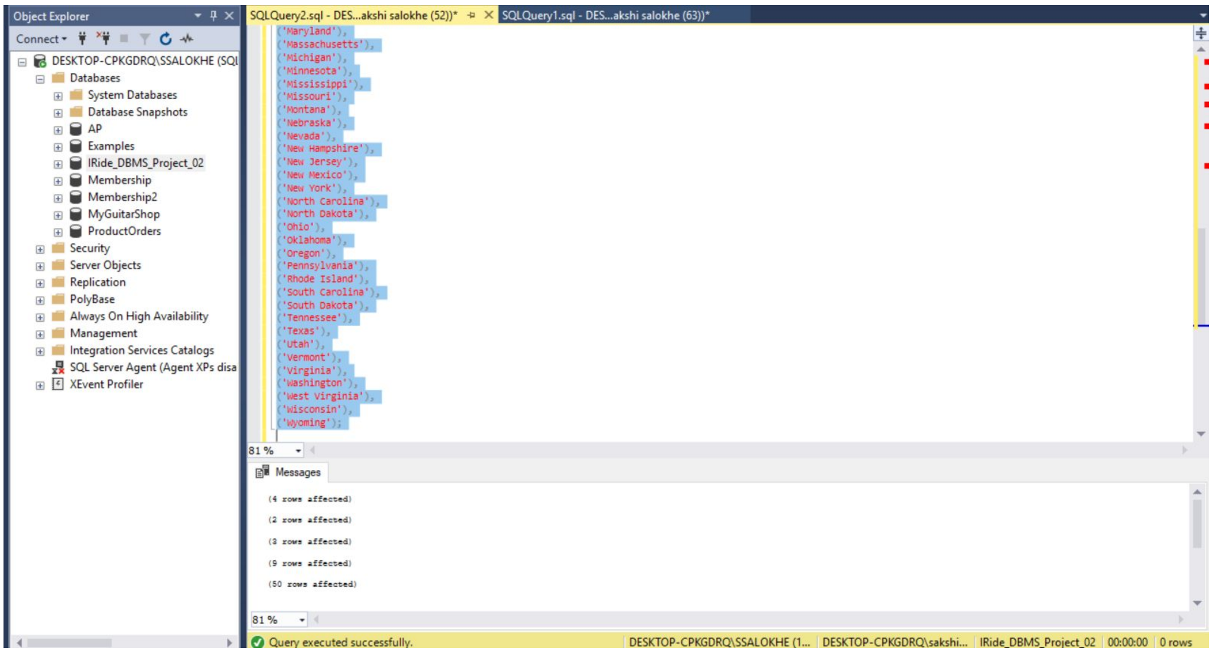


```sql
INSERT INTO
Customers (first_name,last_name,middle_name,phone_no,email,address,user_name,date_of_birth,date_of_last_trip,total_trips_this_year,ratings,isDriver,isActive)
VALUES
('Sakshi Sanjay','Salokhe','','3152438254','ssalokhe@syr.edu','105 Concord Pl, Syracuse, NY', 'ssalokhe', '1995-11-19', '2019-04-21',0,5.0,0,1),
('Pushkar Mahendra','Tatiya','','3153963628','pmtatiya@syr.edu','105 Concord Pl, Syracuse, NY', 'pmtatiya', '1995-08-02', '2019-03-21',0,5.0,1,1),
('Shruti','Salokhe','','3152438434','sshrutis@syr.edu','318 Westcott Street, Syracuse, NY', 'sshrutis', '2000-01-25', '2018-12-13',0,5.0,0,1),
('Vishnu','Menon','Kailas','8152438252','kmvishnu@gmail.com','105 Concord Pl, NJ', 'kmvishnu', '1992-12-19', '2019-04-21',0,5.0,1,1),
('Kruti','Kalmath','','8004557895','kruti@gmail.com','New Brunswick, NY', 'kruti', '1994-05-19', '2019-04-15',0,5.0,0,1),
('Chaitali','P','C','3152438000','cpulkund@syr.edu','111 Trinity Pl, Syracuse, NY', 'cpulkund', '1995-09-18', '2019-04-02',0,5.0,0,1),
('Saurabh','Pohnerkar','','3152336254','spohner@syr.edu','105 South Beach Street, Syracuse, NY', 'saurabh', '1996-01-05', '2018-11-21',0,5.0,1,1),
('Souradeepta','Biswas','','8155168254','soura@syr.edu','222 Broadway, NY', 'soura', '1990-02-28', '2019-04-01',0,5.0,0,1),
('Siddharth','Kumar','S','3152008254','skumar@syr.edu','Frederick, Maryland', 'skumar', '1995-06-19', '2019-03-11',0,5.0,0,1),
('Anushka Atul','Patil','','3152448004','apatil@syr.edu','Boston, MA', 'apatil', '1997-09-04', '2019-01-17',0,5.0,0,1);

INSERT INTO
CustomerCreditCards (cust_id,card_number,name_on_card,card_expiry_date,CVV,isDefault)
VALUES
(1,'1234123412341234','Sakshi Salokhe','2022-11-01',123,1),
(1,'1234123412344321','Sakshi Salokhe', '2025-01-01',432,0),
(2,'2345234523452345','Pushkar Tatiya','2022-01-01',234,1),
(2,'2345234523455432','Pushkar M Tatiya','2024-01-01',543,0),
(2,'2345234554325432','Pushkar Tatiya','2018-12-01',503,0),
(3,'3456345634563456','Shruti Salokhe','2023-10-01',943,1),
(4,'4567456745674567','Vishnu KM','2024-05-21',643,1),
(5,'5678567856785678','Kruti Kalmath','2021-01-01',533,1),
(6,'2345678967896789','Chaitali','2020-01-01',003,1),
(7,'876987698769876','Saurabh Pohnerkar','2024-01-01',911,1),
(8,'6478647864789990','Souradeepta Biswas','2023-04-21',001,1),
(9,'2314346243114634','Siddharth K','2021-01-01',563,1),
(10,'5948362545987465','Anushka A Patil','2025-12-12',333,1);
```

```
(10 rows affected)

(13 rows affected)
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows

```
INSERT INTO
DriverCurrentStatus (status)
VALUES

('inActive'),
('Idle/Available'),
('Busy / onTrip'),
('offline');

INSERT INTO
BankAccountType (name)
VALUES

('Checking'),
('Savings');

INSERT INTO
CarClass (name)
VALUES

('Regular Sedan'),
('Luxury Sedan'),
('SUV');

INSERT INTO
CarColors (color_name)
VALUES

('Red'),
('Yellow'),
('Blue'),
('Green'),
('Orange'),
('Brown'),
('Grey'),
('White'),
('Black');

INSERT INTO
StateTable (name)
VALUES

('Alabama'),
('Alaska'),
('Arizona'),
('Arkansas'),
('California'),
('Colorado'),
('Connecticut'),
('Delaware'),
('Florida'),
('Georgia'),
('Hawaii'),
('Idaho'),
('Illinois'),
('Indiana'),
('Iowa'),
('Kansas'),
('Kentucky'),
('Louisiana'),
('Maine'),
('Maryland'),
('Massachusetts'),
('Michigan'),
('Minnesota'),
('Mississippi'),
('Missouri'),
('Montana'),
('Nebraska'),
('Nevada'),
('New Hampshire'),
('New Jersey'),
('New Mexico'),
('New York'),
('North Carolina'),
('North Dakota'),
('Ohio'),
('Oklahoma'),
('Oregon'),
('Pennsylvania'),
('Rhode Island'),
('South Carolina'),
('South Dakota'),
('Tennessee'),
('Texas'),
('Utah'),
('Vermont'),
('Virginia'),
('Washington'),
('West Virginia'),
('Wisconsin'),
('Wyoming');
```



```
('Maryland'),
('Massachusetts'),
('Michigan'),
('Minnesota'),
('Mississippi'),
('Missouri'),
('Montana'),
('Nebraska'),
('Nevada'),
('New Hampshire'),
('New Jersey'),
('New Mexico'),
('New York'),
('North Carolina'),
('North Dakota'),
('Ohio'),
('Oklahoma'),
('Oregon'),
('Pennsylvania'),
('Rhode Island'),
('South Carolina'),
('South Dakota'),
('Tennessee'),
('Texas'),
('Utah'),
('Vermont'),
('Virginia'),
('Washington'),
('West Virginia'),
('Wisconsin'),
('Wyoming');
```

```
(4 rows affected)

(2 rows affected)

(3 rows affected)

(9 rows affected)

(50 rows affected)
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows

INSERT INTO
Driver(first_name,last_name,middle_name,date_of_birth,date_of_joining,SSN, address,phone_number,email_address, ratings,driver_current_status)
VALUES

('Pushkar Mahendra','Tatiya','','1995-08-02','2019-01-01',123123123,'105 Concord Place, Syracuse, NY','3152438254','pmtatiya@syr.edu',5.0,4),

('Vishnu','Menon','Kailas', '1992-12-19', '2017-12-01',234234234,'105 Concord Pl, NJ','8152438252','kmvishnu@gmail.com',5.0,1),

('Saurabh','Pohnerkar','','1996-01-05', '2018-01-21',345345345,'105 South Beach Street, Syracuse, NY','3152336254','spohner@syr.edu',5.0,2),

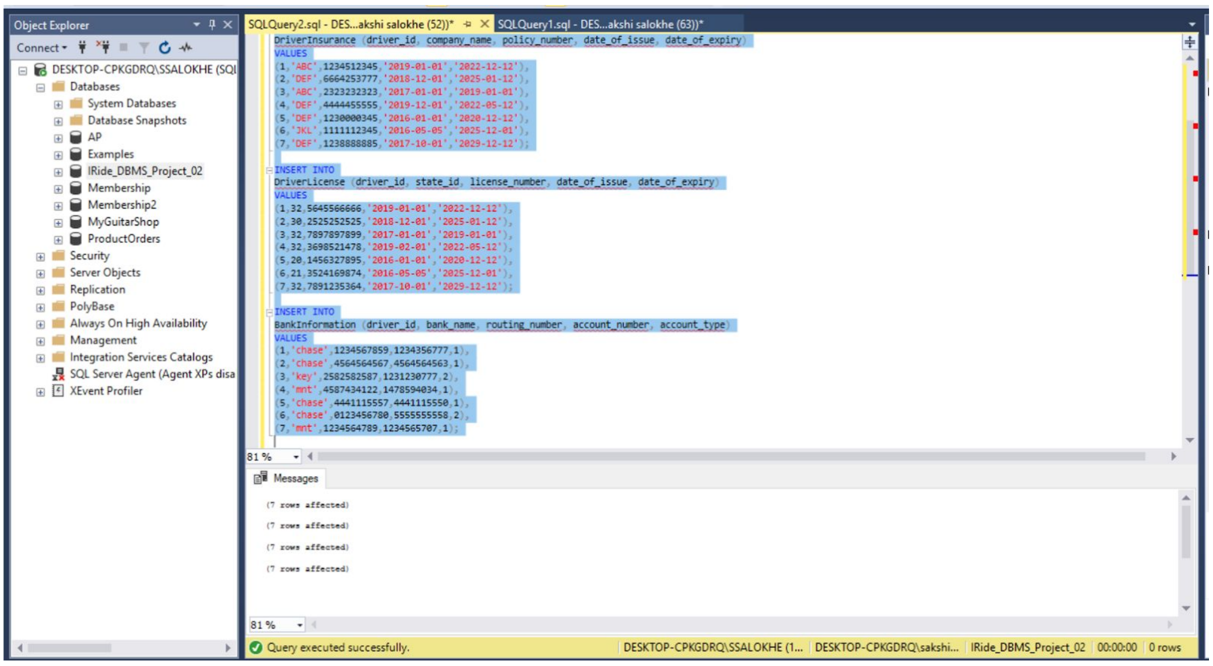
('Prashant','Kamath','','1993-06-21','2018-07-08',456456456,'Green Lakes Park, Syracuse, NY','8974587955','pkamat@gmail.com',5.0,4),

('Ritesh','Deshmukh' , 'M', '1985-02-19','2018-08-08',567567568,'Frederick,Maryland','4859632145','rdeshmukh@gmail.com',5.0,2),

('Shah','Khan', 'Rukh', '1971-11-02','2018-06-08',875487544,'Boston, MA', '5654228888','skhan@gmail.com',5.0,4),

('Aishwarya','Rai','','1974-11-01','2019-04-08',199999990,'New York City','7908096677','arai@gmail.com',5.0,1);

```
INSERT INTO
DriverInsurance (driver_id, company_name, policy_number, date_of_issue, date_of_expiry)
VALUES

(1,'ABC',1234512345,'2019-01-01','2022-12-12'),
(2,'DEF',6664253777,'2018-12-01','2025-01-12'),
(3,'ABC',2323232323,'2017-01-01','2019-01-01'),
(4,'DEF',4444455555,'2019-12-01','2022-05-12'),
(5,'DEF',1230000345,'2016-01-01','2020-12-12'),
(6,'JKL',1111112345,'2016-05-05','2025-12-01'),
(7,'DEF',1238888885,'2017-10-01','2029-12-12');

INSERT INTO
DriverLicense (driver_id, state_id, license_number, date_of_issue, date_of_expiry)
VALUES

(1,32,5645566666,'2019-01-01','2022-12-12'),
(2,30,2525252525,'2018-12-01','2025-01-12'),
(3,32,7897897899,'2017-01-01','2019-01-01'),
(4,32,3698521478,'2019-02-01','2022-05-12'),
(5,20,1456327895,'2016-01-01','2020-12-12'),
(6,21,3524169874,'2016-05-05','2025-12-01'),
(7,32,7891235364,'2017-10-01','2029-12-12');



INSERT INTO
BankInformation (driver_id, bank_name, routing_number, account_number, account_type)
VALUES

(1,'chase',1234567859,1234356777,1),
(2,'chase',4564564567,4564564563,1),
(3,'key',2582582587,1231230777,2),
(4,'mnt',4587434122,1478594034,1),
(5,'chase',4441115557,4441115550,1),
(6,'chase',0123456780,5555555558,2),
(7,'mnt',1234564789,1234565707,1);
```



```
DriverInsurance (driver_id, company_name, policy_number, date_of_issue, date_of_expiry)
VALUES
(1,'ABC',1234512345,'2019-01-01','2022-12-12'),
(2,'DEF',6664253777,'2018-12-01','2025-01-12'),
(3,'ABC',2323232323,'2017-01-01','2019-01-01'),
(4,'DEF',4444455555,'2019-12-01','2022-05-12'),
(5,'DEF',1230000345,'2016-01-01','2020-12-12'),
(6,'JKL',1111112345,'2016-05-05','2025-12-01'),
(7,'DEF',1238888885,'2017-10-01','2029-12-12');

INSERT INTO
DriverLicense (driver_id, state_id, license_number, date_of_issue, date_of_expiry)
VALUES
(1,32,5645566666,'2019-01-01','2022-12-12'),
(2,30,2525252525,'2018-12-01','2025-01-12'),
(3,32,7897897899,'2017-01-01','2019-01-01'),
(4,32,3698521478,'2019-02-01','2022-05-12'),
(5,20,1456327895,'2016-01-01','2020-12-12'),
(6,21,3524169874,'2016-05-05','2025-12-01'),
(7,32,7891235364,'2017-10-01','2029-12-12');

INSERT INTO
BankInformation (driver_id, bank_name, routing_number, account_number, account_type)
VALUES
(1,'chase',1234567859,1234356777,1),
(2,'chase',4564564567,4564564563,1),
(3,'key',2582582587,1231230777,2),
(4,'mnt',4587434122,1478594034,1),
(5,'chase',4441115557,4441115550,1),
(6,'chase',0123456780,5555555558,2),
(7,'mnt',1234564789,1234565707,1);
```

```
(7 rows affected)

(7 rows affected)

(7 rows affected)

(7 rows affected)
```

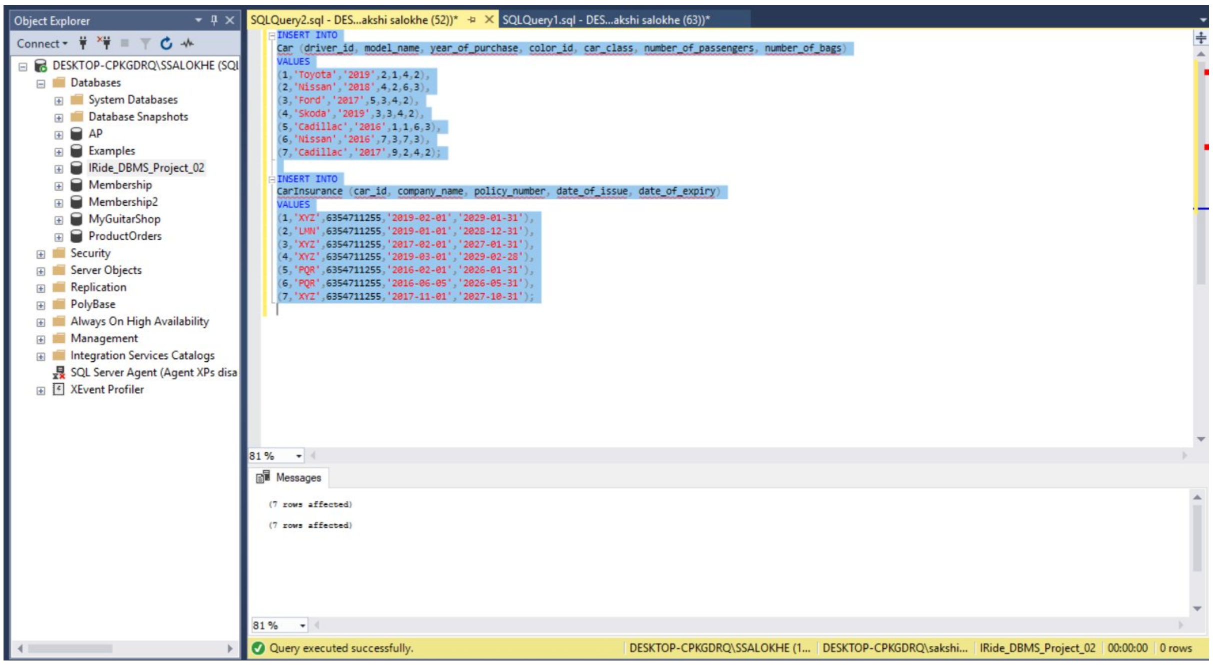
Query executed successfully.

INSERT INTO
Car (driver_id, model_name, year_of_purchase, color_id, car_class, number_of_passengers, number_of_bags)
VALUES

(1,'Toyota','2019',2,1,4,2),
(2,'Nissan','2018',4,2,6,3),
(3,'Ford','2017',5,3,4,2),
(4,'Skoda','2019',3,3,4,2),
(5,'Cadillac','2016',1,1,6,3),
(6,'Nissan','2016',7,3,7,3),
(7,'Cadillac','2017',9,2,4,2);

INSERT INTO
CarInsurance (car_id, company_name, policy_number, date_of_issue, date_of_expiry)
VALUES

(1,'XYZ',6354711255,'2019-02-01','2029-01-31'),
(2,'LMN',6354711255,'2019-01-01','2028-12-31'),
(3,'XYZ',6354711255,'2017-02-01','2027-01-31'),
(4,'XYZ',6354711255,'2019-03-01','2029-02-28'),
(5,'PQR',6354711255,'2016-02-01','2026-01-31'),
(6,'PQR',6354711255,'2016-06-05','2026-05-31'),
(7,'XYZ',6354711255,'2017-11-01','2027-10-31');



```
INSERT INTO
Car (driver_id, model_name, year_of_purchase, color_id, car_class, number_of_passengers, number_of_bags)
VALUES
(1,'Toyota','2019',2,1,4,2),
(2,'Nissan','2018',4,2,6,3),
(3,'Ford','2017',5,3,4,2),
(4,'Skoda','2019',3,3,4,2),
(5,'Cadillac','2016',1,1,6,3),
(6,'Nissan','2016',7,3,7,3),
(7,'Cadillac','2017',9,2,4,2);

INSERT INTO
CarInsurance (car_id, company_name, policy_number, date_of_issue, date_of_expiry)
VALUES
(1,'XYZ',6354711255,'2019-02-01','2029-01-31'),
(2,'LMN',6354711255,'2019-01-01','2028-12-31'),
(3,'XYZ',6354711255,'2017-02-01','2027-01-31'),
(4,'XYZ',6354711255,'2019-03-01','2029-02-28'),
(5,'PQR',6354711255,'2016-02-01','2026-01-31'),
(6,'PQR',6354711255,'2016-06-05','2026-05-31'),
(7,'XYZ',6354711255,'2017-11-01','2027-10-31');
```

```
(7 rows affected)

(7 rows affected)
```

```
Query executed successfully.   DESKTOP-CPKGDRQ\SSALOKHE (1...   DESKTOP-CPKGDRQ\sakshi...   IRide_DBMS_Project_02   00:00:00   0 rows
```

INSERT INTO
Trips
(pickup_date,dropoff_date,trip_completion,pickup_address,destination_address,number_of_people,number_of_bags,customer_notes,driver_notes,cust_id,driver_id,total_cost,tip,cust_credit_card_info)
VALUES

('2019-04-21','2019-04-21',1,'Slutzker Center','105 Concord Place',3,0,",",1,7,7.6,1.5,1),

('2019-04-21','2019-04-21',1,'Hendrick Chapel', 'Destiny USA',2,0,",",4,4,14.7,1.5,7),

('2019-04-15','2019-04-15',1,'Alto cinco', '111 TrinityPl',1,1,",",5,7,7.6,2.5,8),

('2019-04-02','2019-04-02',1,'Recess Caffe','Bird Library',2,0,'meet me at the east entrance','sure.',6,1,10.0,3.5,9),

('2019-04-01','2019-04-01',1,'Als Pub','Crouse Hospital',1,2,'come to the parking place','sure sir',8,2,12.3,1.5,11),

('2019-03-21','2019-03-21',1,'105 Concord Place', 'Recess Caffe',4,2,",",2,5,7.6,1.0,3),

('2019-03-11','2019-03-11',1,'Bird Library','Hendrick Chapel',4,2,",",9,6,3.2,2.0,12),

('2019-01-17','2019-01-17',1,'Crouse Hospital','111 TrinityPl',1,0,",",10,1,11.0,1.5,13),

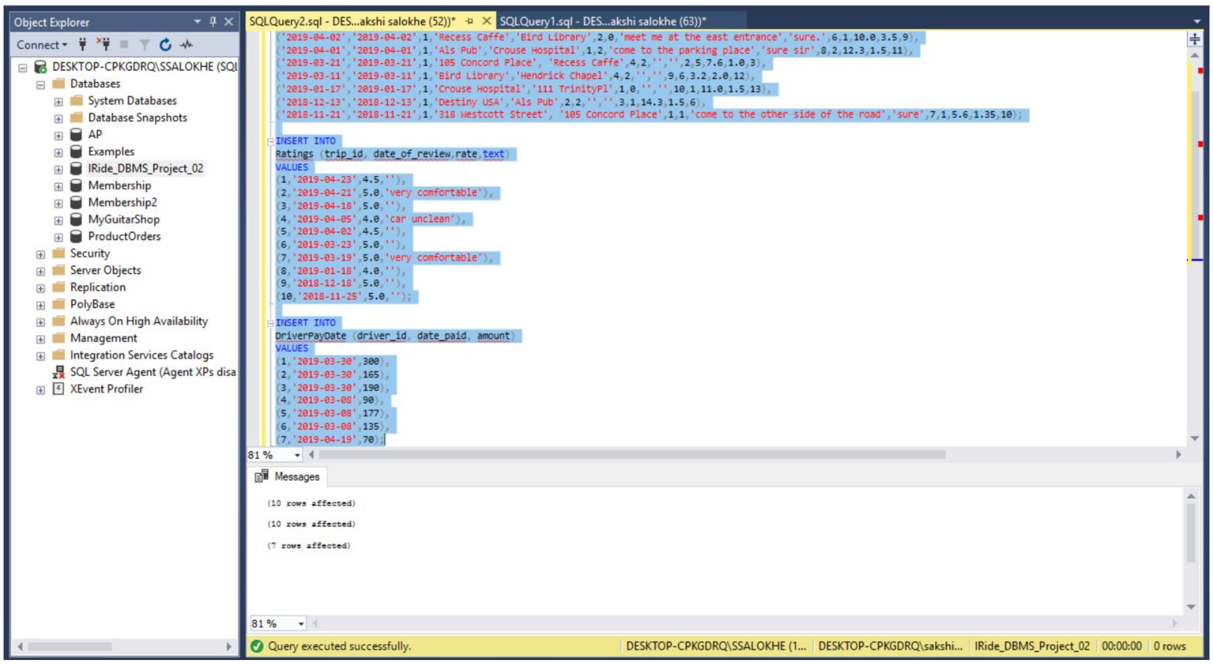
('2018-12-13','2018-12-13',1,'Destiny USA','Als Pub',2,2,",",3,1,14.3,1.5,6), ('2018-11-21','2018-11-21',1,'318 Westcott Street', '105 Concord Place',1,1,'come to the other side of the road','sure',7,1,5.6,1.35,10);

INSERT INTO
Ratings (trip_id, date_of_review,rate,text)
VALUES

(1,'2019-04-23',4.5,''),
(2,'2019-04-21',5.0,'very comfortable'),
(3,'2019-04-18',5.0,''),
(4,'2019-04-05',4.0,'car unclean'),
(5,'2019-04-02',4.5,''),
(6,'2019-03-23',5.0,''),
(7,'2019-03-19',5.0,'very comfortable'),
(8,'2019-01-18',4.0,''),
(9,'2018-12-18',5.0,''),
(10,'2018-11-25',5.0,'');

INSERT INTO
DriverPayDate (driver_id, date_paid, amount)
VALUES

(1,'2019-03-30',300),
(2,'2019-03-30',165),
(3,'2019-03-30',190),
(4,'2019-03-08',90),
(5,'2019-03-08',177),
(6,'2019-03-08',135),
(7,'2019-04-19',70);



```
('2019-04-02','2019-04-02',1,'Recess Caffe','Bird Library',2,0,'meet me at the east entrance','sure.',6,1,10.0,3.5,9),
('2019-04-01','2019-04-01',1,'Als Pub','Crouse Hospital',1,2,'come to the parking place','sure sir',8,2,12.3,1.5,11),
('2019-03-21','2019-03-21',1,'105 Concord Place', 'Recess Caffe',4,2,'','',2,5,7.6,1.0,3),
('2019-03-11','2019-03-11',1,'Bird Library','Hendrick Chapel',4,2,'','',9,6,3.2,2.0,12),
('2019-01-17','2019-01-17',1,'Crouse Hospital','111 TrinityPl',1,0,'','',10,1,11.0,1.5,13),
('2018-12-13','2018-12-13',1,'Destiny USA','Als Pub',2,2,'','',3,1,14.3,1.5,6),
('2018-11-21','2018-11-21',1,'318 Westcott Street', '105 Concord Place',1,1,'come to the other side of the road','sure',7,1,5.6,1.35,10);

INSERT INTO
Ratings (trip_id, date_of_review,rate,text)
VALUES
(1,'2019-04-23',4.5,''),
(2,'2019-04-21',5.0,'very comfortable'),
(3,'2019-04-18',5.0,''),
(4,'2019-04-05',4.0,'car unclean'),
(5,'2019-04-02',4.5,''),
(6,'2019-03-23',5.0,''),
(7,'2019-03-19',5.0,'very comfortable'),
(8,'2019-01-18',4.0,''),
(9,'2018-12-18',5.0,''),
(10,'2018-11-25',5.0,'');

INSERT INTO
DriverPayDate (driver_id, date_paid, amount)
VALUES
(1,'2019-03-30',300),
(2,'2019-03-30',165),
(3,'2019-03-30',190),
(4,'2019-03-08',90),
(5,'2019-03-08',177),
(6,'2019-03-08',135),
(7,'2019-04-19',70);
```

```
(10 rows affected)

(10 rows affected)

(7 rows affected)
```

Query executed successfully.

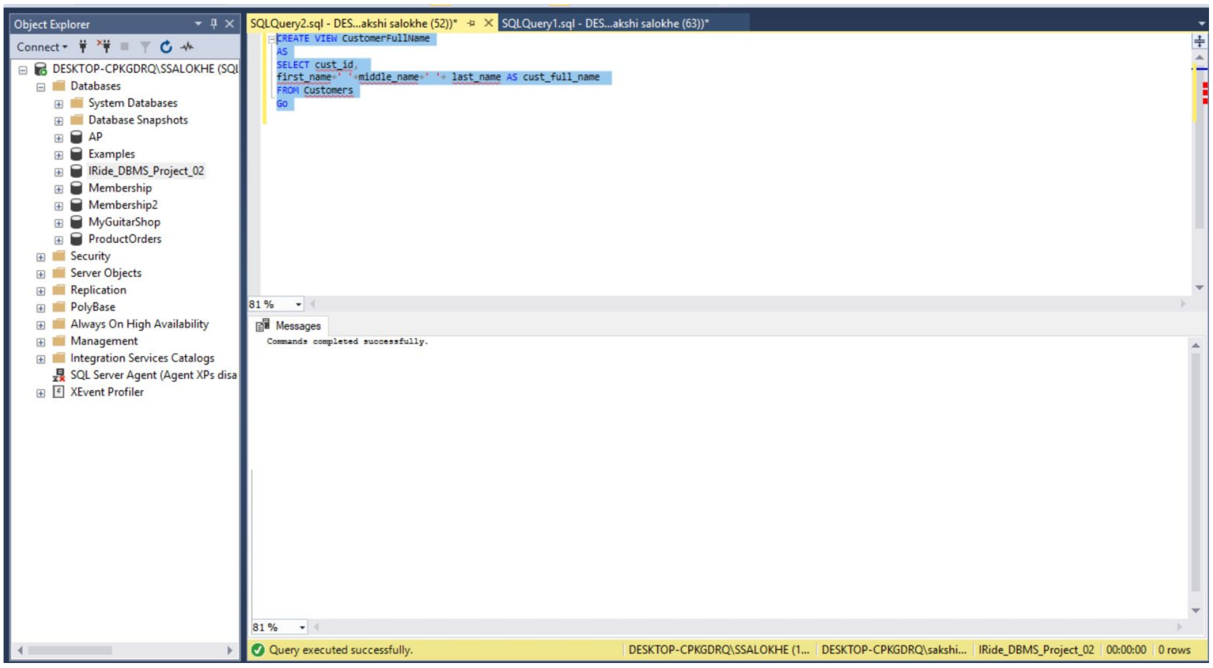
Remarks:
Thus we created the new database **IRide_DBMS_Project_02** and created the necessary tables and inserted values into the tables using mysql queries.

Views

View 1: To get the full name of a customer from the columns in the Customers table (first_name, last_name and middle_name columns).

```
CREATE VIEW CustomerFullName
AS
        SELECT cust_id,
                first_name+' '+middle_name+' '+ last_name AS cust_full_name
        FROM Customers
Go

SELECT * FROM CustomerFullName
```

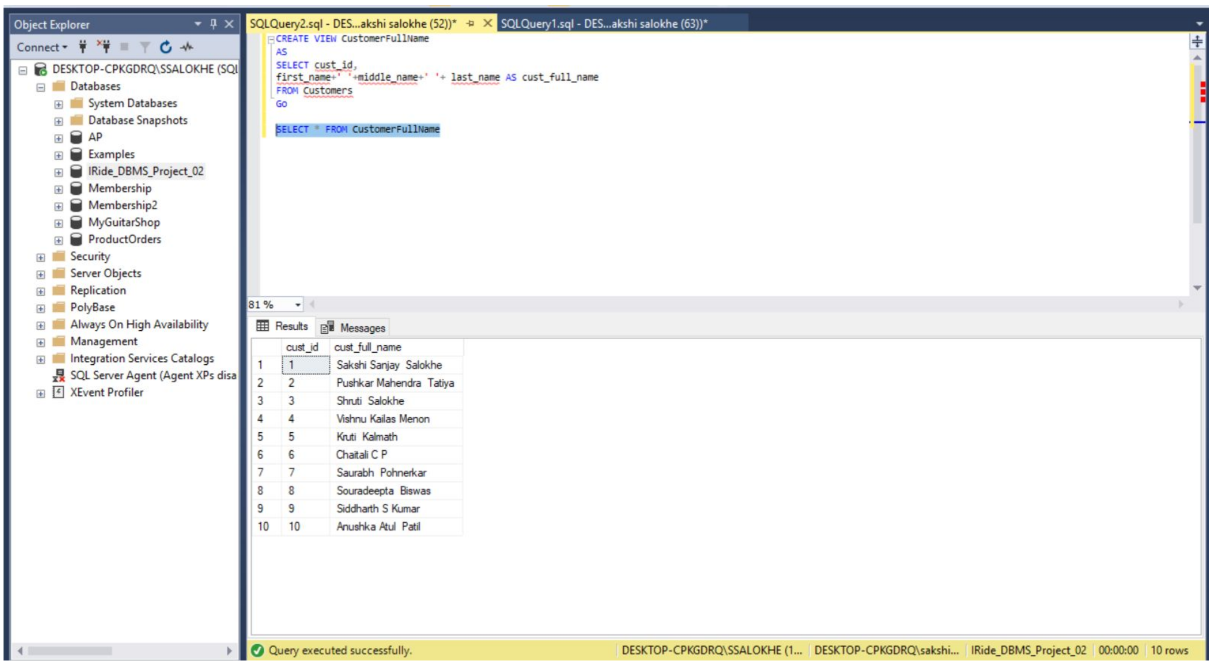
Screenshot:



```
CREATE VIEW CustomerFullName
AS
SELECT cust_id,
first_name+' '+middle_name+' '+ last_name AS cust_full_name
FROM Customers
Go
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows



```sql
CREATE VIEW CustomerFullName
AS
SELECT cust_id,
first_name+' '+middle_name+' '+ last_name AS cust_full_name
FROM Customers
Go

SELECT * FROM CustomerFullName
```

	cust_id	cust_full_name
1	1	Sakshi Sanjay Salokhe
2	2	Pushkar Mahendra Tatiya
3	3	Shruti Salokhe
4	4	Vishnu Kailas Menon
5	5	Kruti Kalmath
6	6	Chaitali C P
7	7	Saurabh Pohnerkar
8	8	Souradeepta Biswas
9	9	Siddharth S Kumar
10	10	Anushka Atul Patil

Query executed successfully. DESKTOP-CPKGDRQ\SSALOKHE (1... DESKTOP-CPKGDRQ\sakshi... IRide_DBMS_Project_02 00:00:00 10 rows

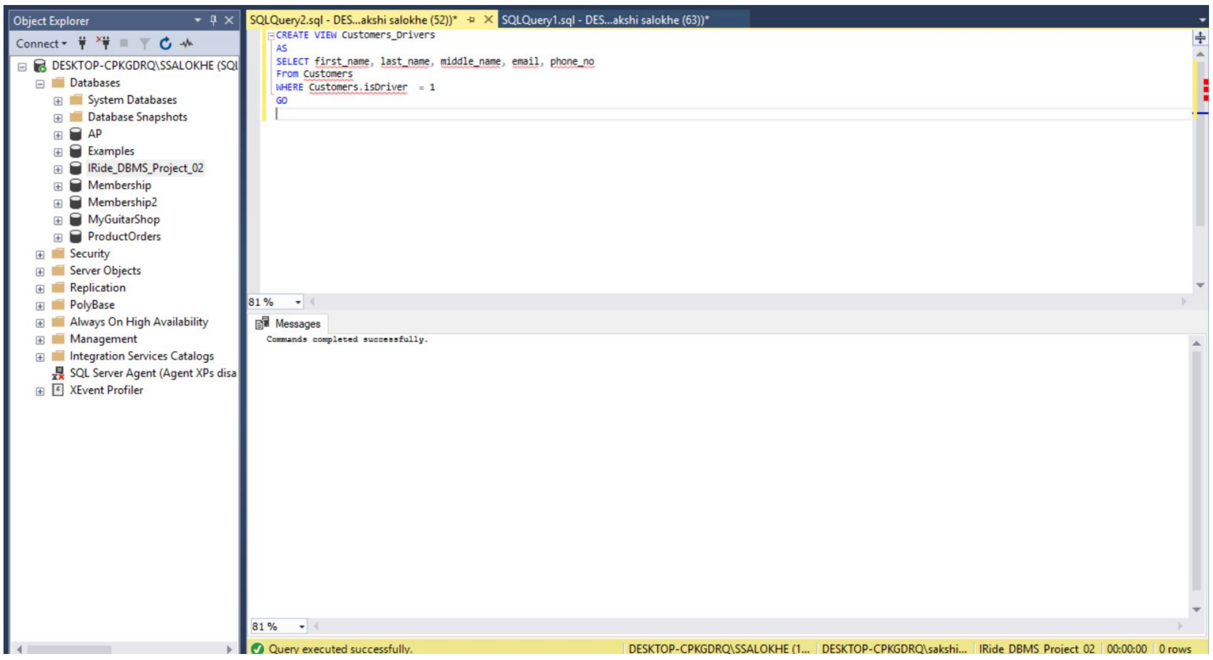
Remark:
Thus we created a view to join customers first name, middle name and last name and to get his full name.

View 2: To find and display the details of the customers that also work as drivers (by checking if the isDriver column value for that customer is 1).

```
CREATE VIEW Customers_Drivers
AS
        SELECT first_name, last_name, middle_name, email, phone_no
        From Customers
        WHERE Customers.isDriver  = 1
GO
```

```
Select * From Customers_Drivers;
```

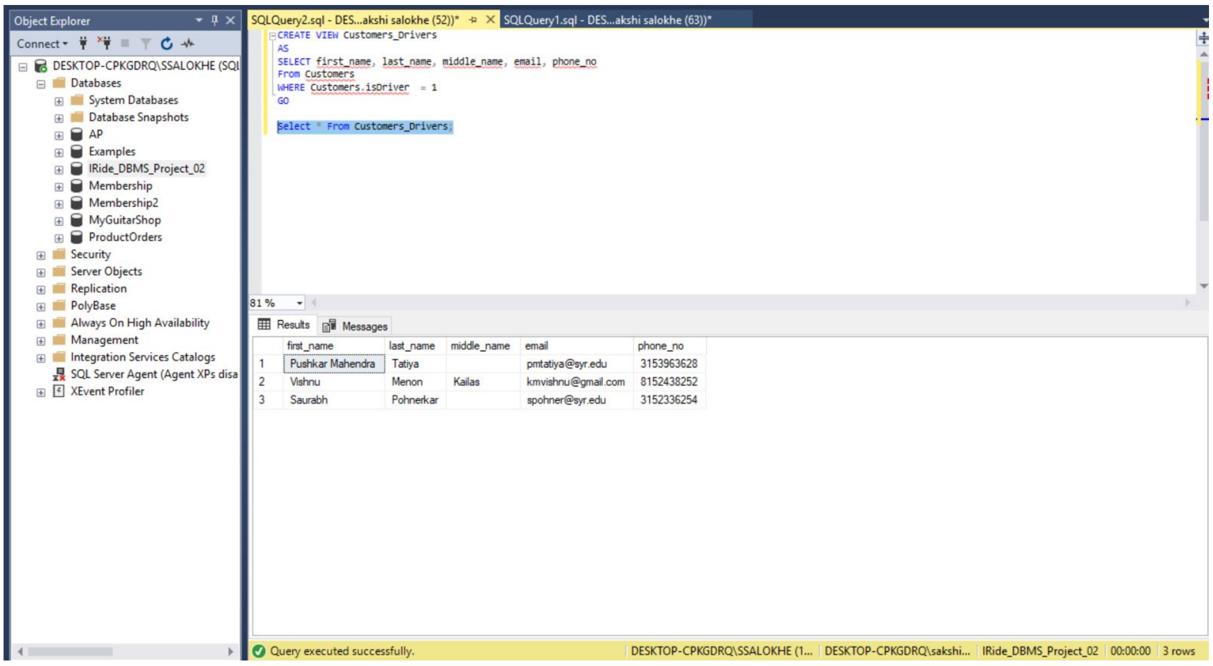
Screenshot:



```
CREATE VIEW Customers_Drivers
AS
SELECT first_name, last_name, middle_name, email, phone_no
From Customers
WHERE Customers.isDriver  = 1
GO
```

```
Commands completed successfully.
```

```
Query executed successfully.   DESKTOP-CPKGDRQ\SSALOKHE (1...   DESKTOP-CPKGDRQ\sakshi...   IRide_DBMS_Project_02   00:00:00   0 rows
```



```
CREATE VIEW Customers_Drivers
AS
SELECT first_name, last_name, middle_name, email, phone_no
From Customers
WHERE Customers.isDriver  = 1
GO

Select * From Customers_Drivers;
```

	first_name	last_name	middle_name	email	phone_no
1	Pushkar Mahendra	Tatiya		pmtatiya@syr.edu	3153963628
2	Vishnu	Menon	Kailas	kmvishnu@gmail.com	8152438252
3	Saurabh	Pohnerkar		spohner@syr.edu	3152336254

```
Query executed successfully.    DESKTOP-CPKGDRQ\SSALOKHE (1...    DESKTOP-CPKGDRQ\sakshi...    IRide_DBMS_Project_02    00:00:00    3 rows
```

Remark:
Thus we created a view to display the users that are customers but also work as drivers.

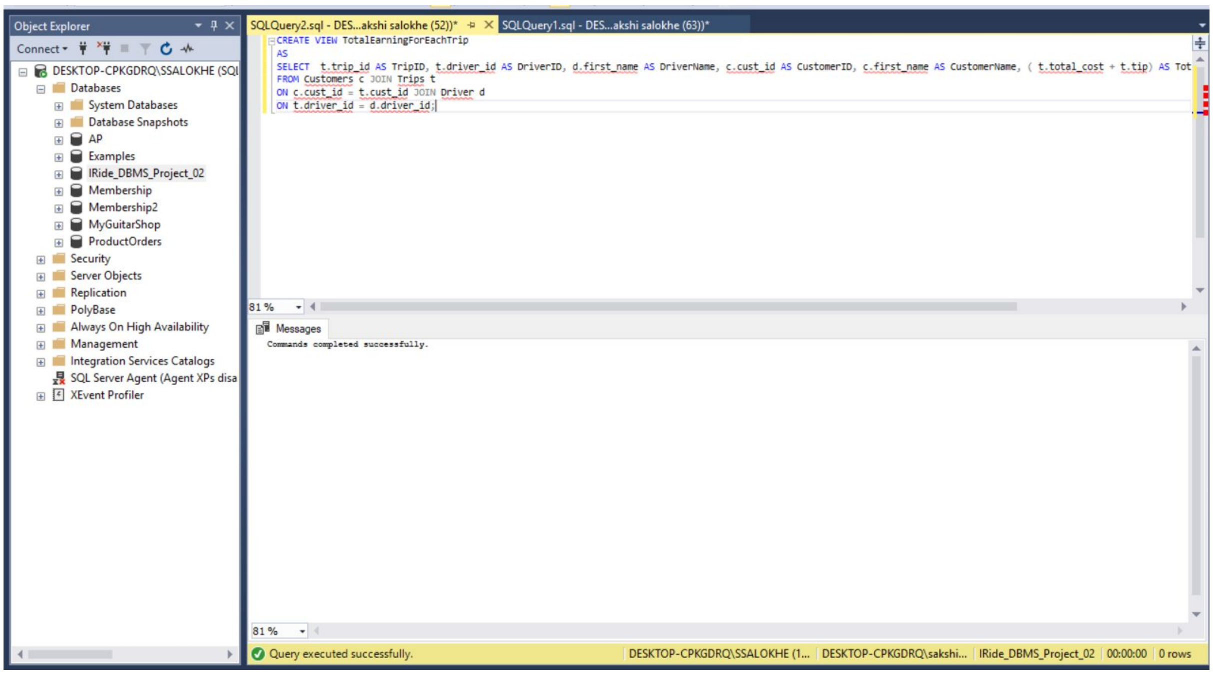
View 3: To calculate the total amount earned in that particular trip (tip + total cost of the trip).

CREATE VIEW TotalEarningForEachTrip
AS

> **SELECT** t.trip_id AS TripID, t.driver_id **AS** DriverID, d.first_name **AS** DriverName, c.cust_id **AS** CustomerID, c.first_name **AS** CustomerName, (t.total_cost + t.tip) **AS** TotalEarning
> **FROM** Customers c **JOIN** Trips t
> **ON** c.cust_id = t.cust_id **JOIN** Driver d
> **ON** t.driver_id = d.driver_id;

Select * **From** TotalEarningForEachTrip;

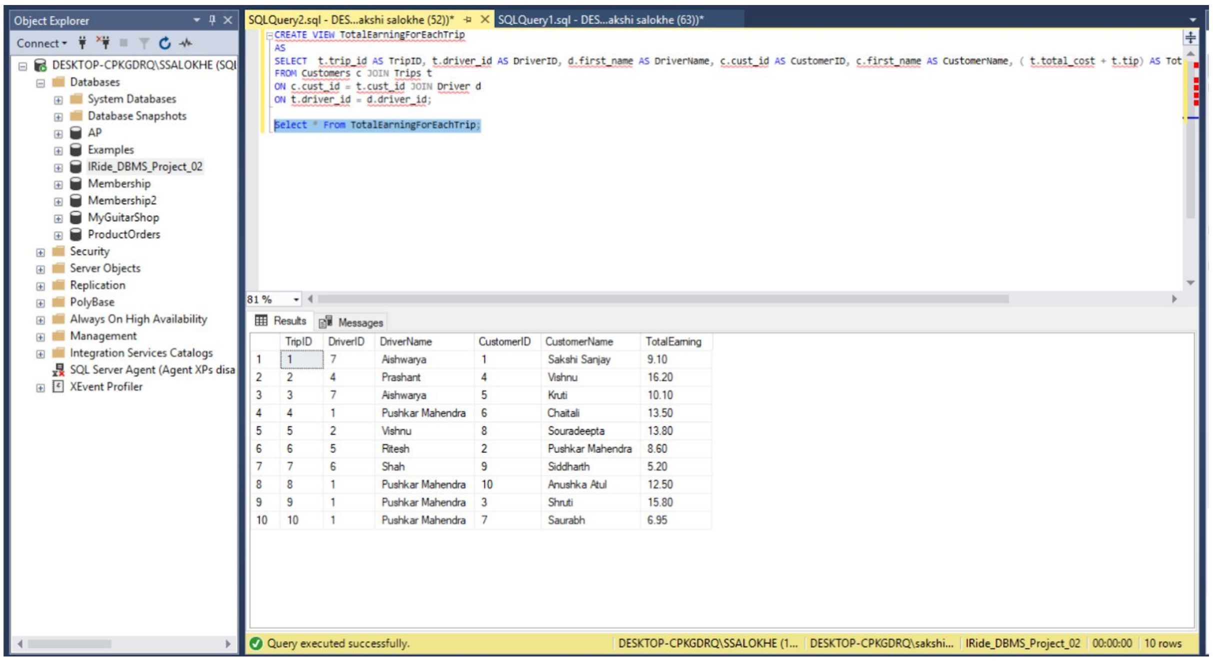
Screenshots:



```
CREATE VIEW TotalEarningForEachTrip
AS
SELECT  t.trip_id AS TripID, t.driver_id AS DriverID, d.first_name AS DriverName, c.cust_id AS CustomerID, c.first_name AS CustomerName, ( t.total_cost + t.tip) AS Tot
FROM Customers c JOIN Trips t
ON c.cust_id = t.cust_id JOIN Driver d
ON t.driver_id = d.driver_id;
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows



```sql
CREATE VIEW TotalEarningForEachTrip
AS
SELECT  t.trip_id AS TripID, t.driver_id AS DriverID, d.first_name AS DriverName, c.cust_id AS CustomerID, c.first_name AS CustomerName, ( t.total_cost + t.tip) AS Tot
FROM Customers c JOIN Trips t
ON c.cust_id = t.cust_id JOIN Driver d
ON t.driver_id = d.driver_id;

Select * From TotalEarningForEachTrip;
```

	TripID	DriverID	DriverName	CustomerID	CustomerName	TotalEaming
1	1	7	Aishwarya	1	Sakshi Sanjay	9.10
2	2	4	Prashant	4	Vishnu	16.20
3	3	7	Aishwarya	5	Kruti	10.10
4	4	1	Pushkar Mahendra	6	Chaitali	13.50
5	5	2	Vishnu	8	Souradeepta	13.80
6	6	5	Ritesh	2	Pushkar Mahendra	8.60
7	7	6	Shah	9	Siddharth	5.20
8	8	1	Pushkar Mahendra	10	Anushka Atul	12.50
9	9	1	Pushkar Mahendra	3	Shruti	15.80
10	10	1	Pushkar Mahendra	7	Saurabh	6.95

Query executed successfully.

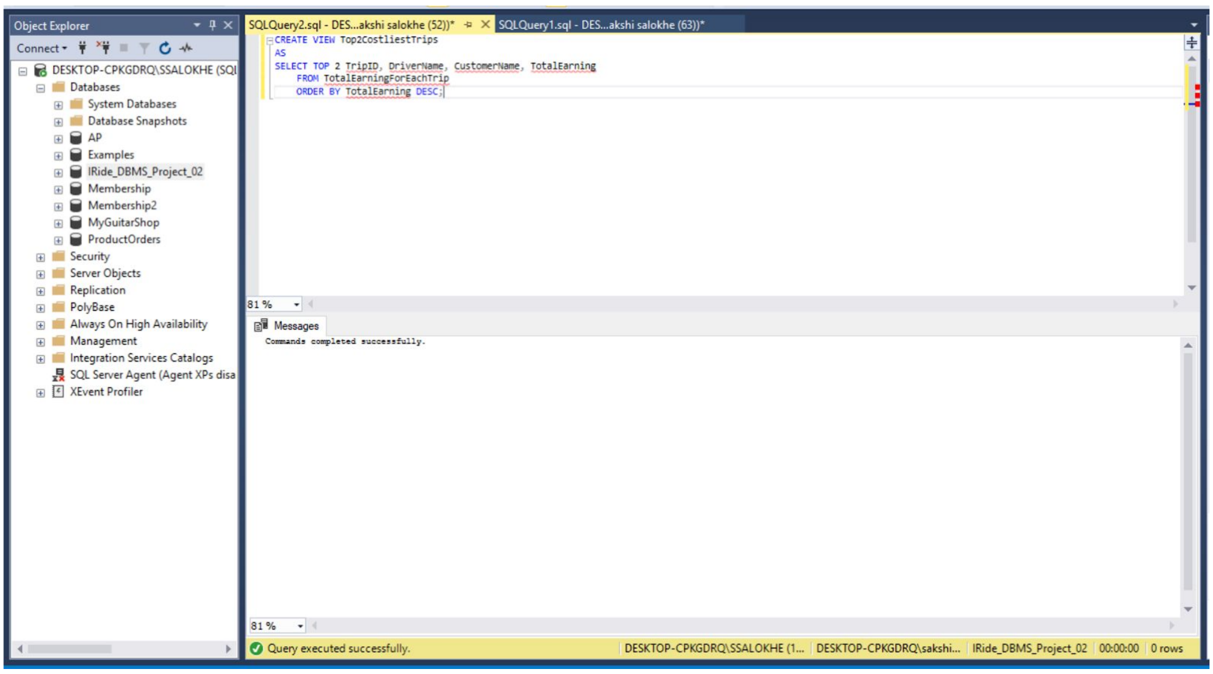
Remark:

Thus we calculate the total trip cost as the sum of trip cost and tip given by a customer to the driver.

View 4: From the total amount earned in the last trip found in the previous view, we create a new view to find the top 2 costliest trips.

```
CREATE VIEW Top2CostliestTrips
AS
SELECT TOP 2 TripID, DriverName, CustomerName, TotalEarning
        FROM TotalEarningForEachTrip
        ORDER BY TotalEarning DESC;

SELECT * FROM Top2CostliestTrips
```

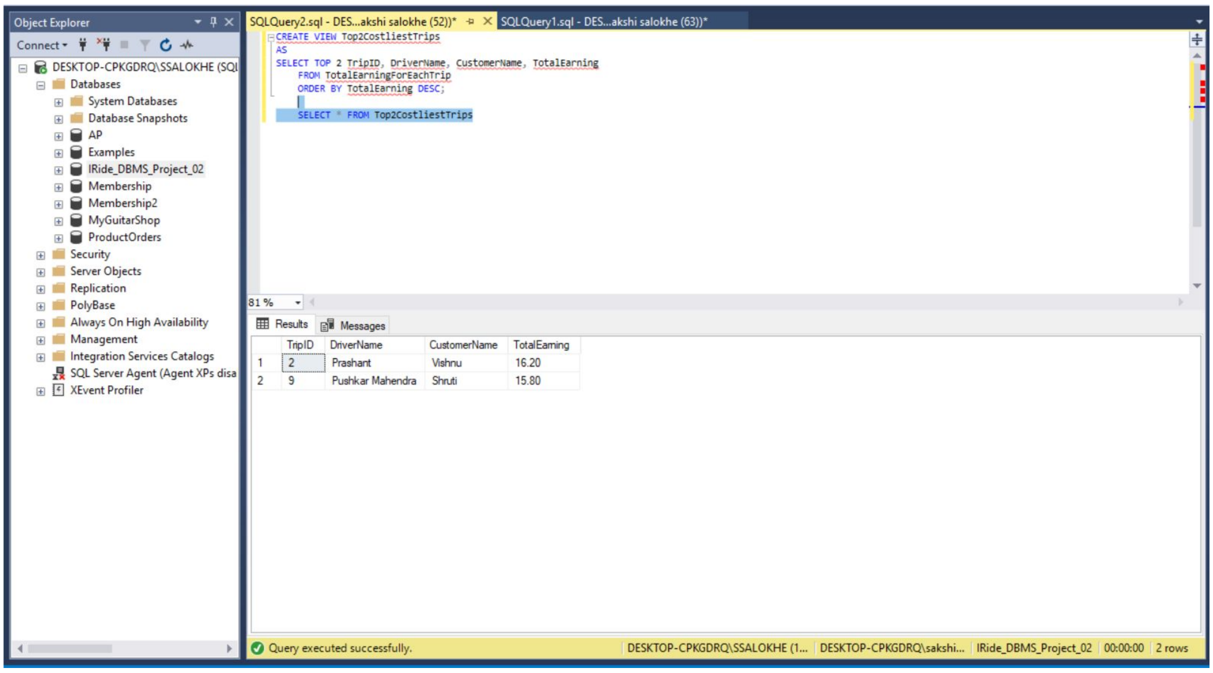
Screenshot:



```
CREATE VIEW Top2CostliestTrips
AS
SELECT TOP 2 TripID, DriverName, CustomerName, TotalEarning
    FROM TotalEarningForEachTrip
    ORDER BY TotalEarning DESC;
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows



```sql
CREATE VIEW Top2CostliestTrips
AS
SELECT TOP 2 TripID, DriverName, CustomerName, TotalEarning
    FROM TotalEarningForEachTrip
    ORDER BY TotalEarning DESC;

    SELECT * FROM Top2CostliestTrips
```

	TripID	DriverName	CustomerName	TotalEarning
1	2	Prashant	Vishnu	16.20
2	9	Pushkar Mahendra	Shruti	15.80

```
Query executed successfully.    DESKTOP-CPKGDRQ\SSALOKHE (1...  DESKTOP-CPKGDRQ\sakshi...  IRide_DBMS_Project_02  00:00:00  2 rows
```

Remark:
Thus we created a view to find the top 2 costliest trips.

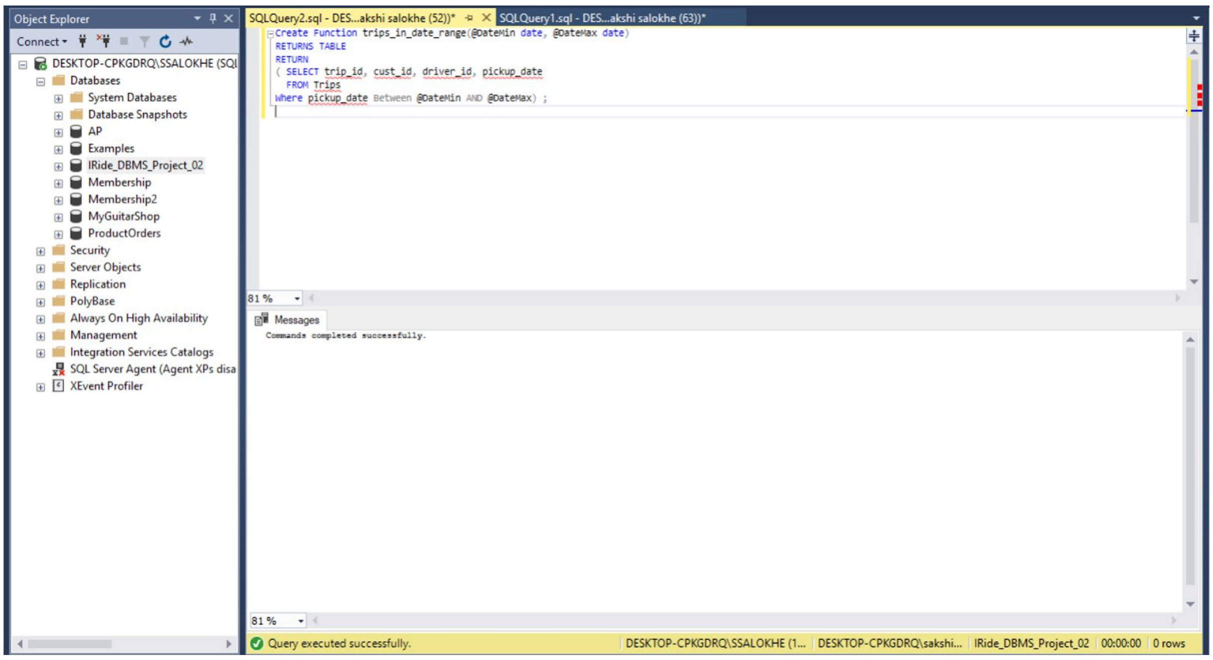
Functions

Function 1: To find and display the details of the trips in a given range of dates.

Create Function trips_in_date_range(@DateMin date, @DateMax date)
RETURNS TABLE
RETURN
    (**SELECT** trip_id, cust_id, driver_id, pickup_date
      FROM Trips
      Where pickup_date Between @DateMin AND @DateMax) ;

Select * **FROM** trips_in_date_range('2019-01-01', '2019-04-22');

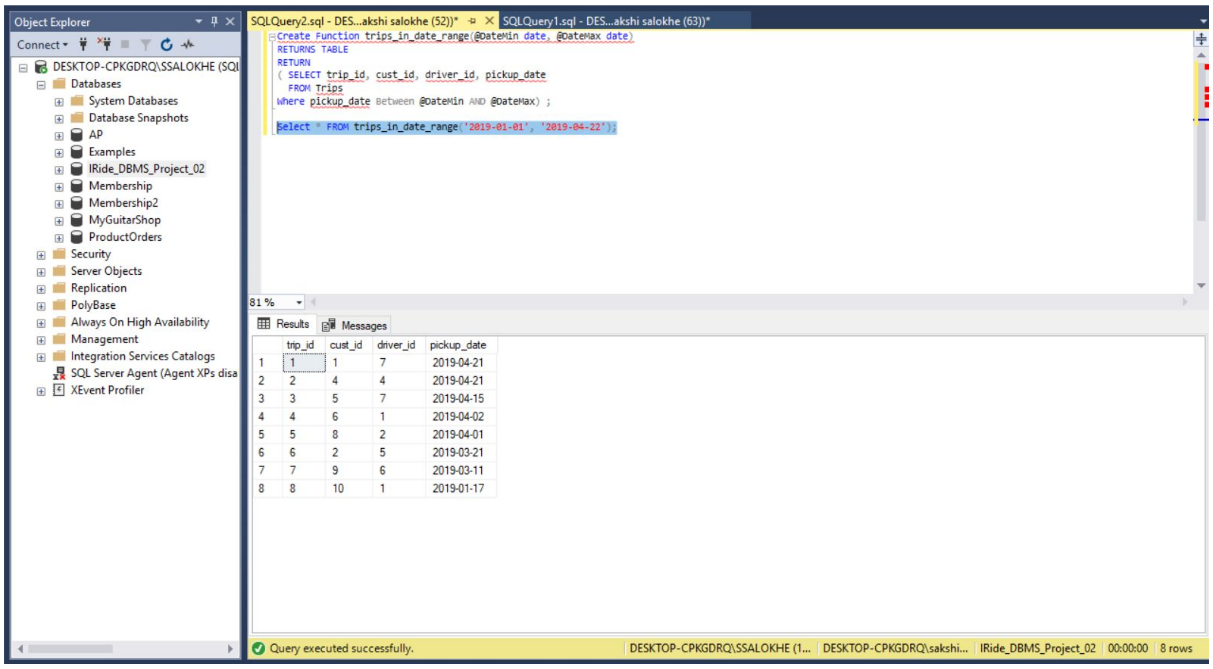
Screenshot:



```
Create Function trips_in_date_range(@DateMin date, @DateMax date)
RETURNS TABLE
RETURN
( SELECT trip_id, cust_id, driver_id, pickup_date
  FROM Trips
Where pickup_date Between @DateMin AND @DateMax) ;
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows



```sql
Create Function trips_in_date_range(@DateMin date, @DateMax date)
RETURNS TABLE
RETURN
( SELECT trip_id, cust_id, driver_id, pickup_date
  FROM Trips
Where pickup_date Between @DateMin AND @DateMax) ;

Select * FROM trips_in_date_range('2019-01-01', '2019-04-22');
```

	trip_id	cust_id	driver_id	pickup_date
1	1	1	7	2019-04-21
2	2	4	4	2019-04-21
3	3	5	7	2019-04-15
4	4	6	1	2019-04-02
5	5	8	2	2019-04-01
6	6	2	5	2019-03-21
7	7	9	6	2019-03-11
8	8	10	1	2019-01-17

```
Query executed successfully.    DESKTOP-CPKGDRQ\SSALOKHE (1...  DESKTOP-CPKGDRQ\sakshi...  IRide_DBMS_Project_02  00:00:00  8 rows
```

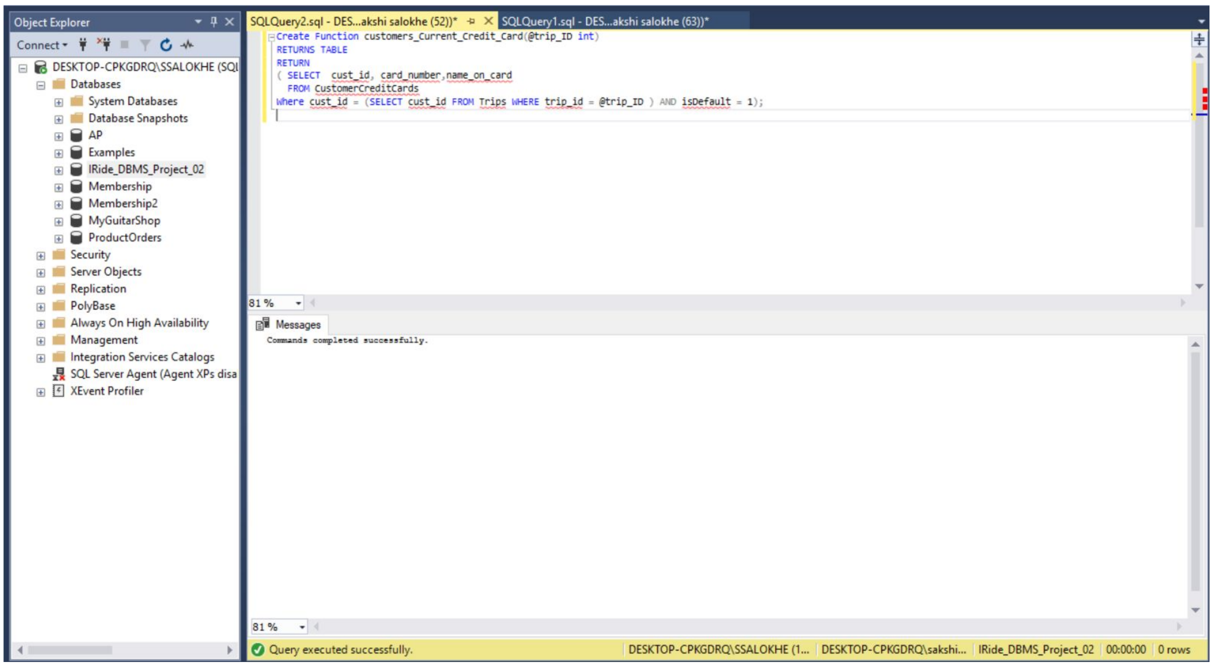
Remark:
Thus we have written a function that displays a table with trip id, customer id, driver id and pickup date within a given range as the user input.

Function 2: To find the credit card that is to be used to deduct money from for that trip. (customers_Current_Credit_card = 1).

Create Function customers_Current_Credit_Card(@trip_ID int)
RETURNS TABLE
RETURN
(**SELECT** cust_id, card_number,name_on_card
FROM CustomerCreditCards
Where cust_id = (**SELECT** cust_id **FROM** Trips WHERE trip_id = @trip_ID) **AND** isDefault = 1);

Select * **FROM** customers_Current_Credit_Card(1);

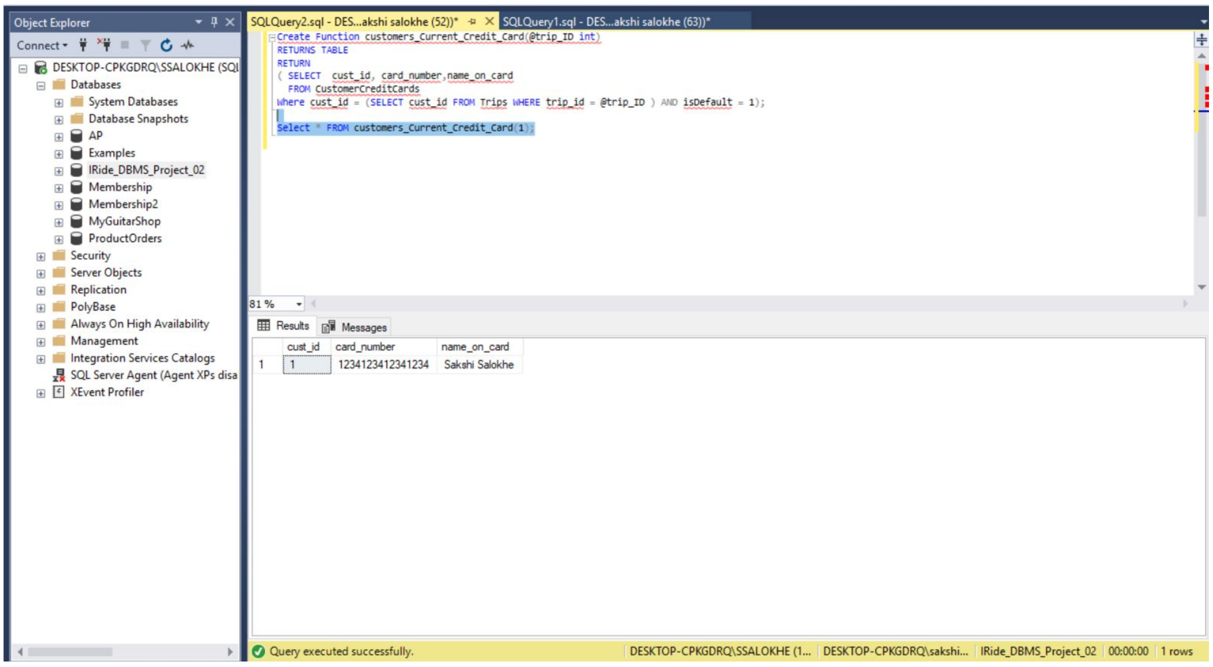
Screenshot:



```
Create Function customers_Current_Credit_Card(@trip_ID int)
RETURNS TABLE
RETURN
( SELECT  cust_id, card_number,name_on_card
  FROM CustomerCreditCards
Where cust_id = (SELECT cust_id FROM Trips WHERE trip_id = @trip_ID ) AND isDefault = 1);
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows



```sql
Create Function customers_Current_Credit_Card(@trip_ID int)
RETURNS TABLE
RETURN
( SELECT  cust_id, card_number,name_on_card
  FROM CustomerCreditCards
Where cust_id = (SELECT cust_id FROM Trips WHERE trip_id = @trip_ID ) AND isDefault = 1);

Select * FROM customers_Current_Credit_Card(1);
```

```
   cust_id  card_number       name_on_card
1  1        1234123412341234  Sakshi Salokhe
```

Query executed successfully.

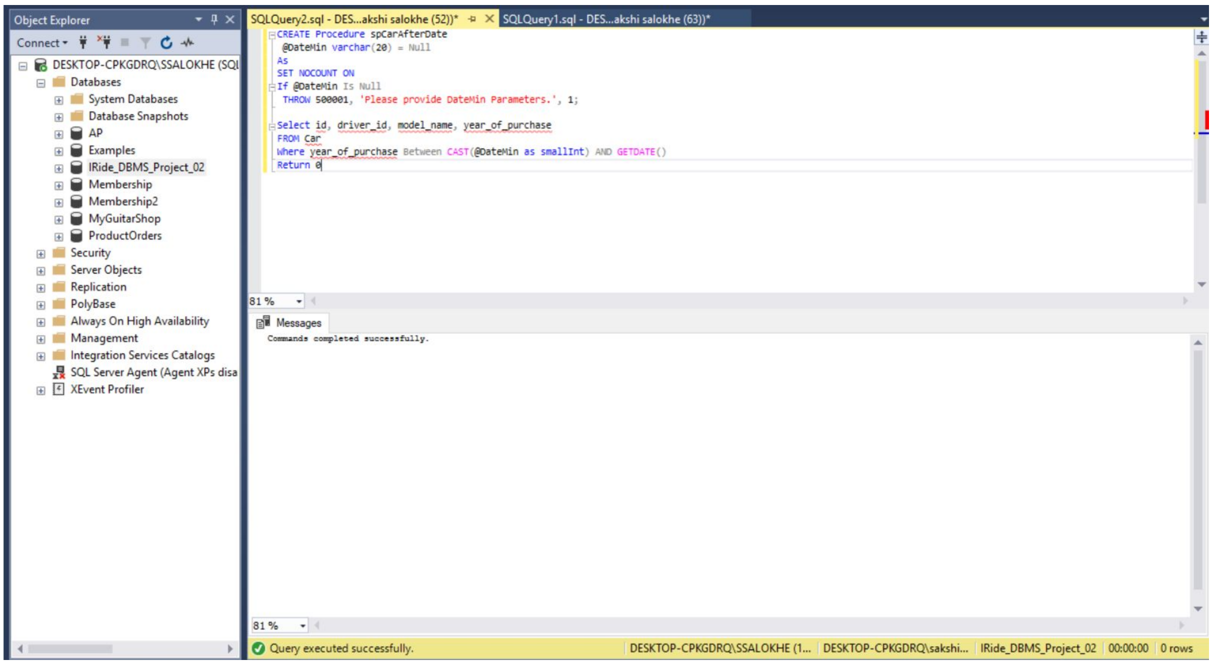
Remark:
Thus we created a function to return a table. We pass the trip id as a parameter. We take the customer in this trip and match that customer id in the table where the card details are stored. If a customer has multiple cards, we select the one that is currently in use which is marked by isDefault = 1.

Stored Procedures

1: To find the cars purchased after a given year.

```
CREATE Procedure spCarAfterDate
 @DateMin varchar(20) = Null
As
        SET NOCOUNT ON
        If @DateMin Is Null
                THROW 500001, 'Please provide DateMin Parameters.', 1;

        Select id, driver_id, model_name, year_of_purchase
        FROM Car
        Where year_of_purchase Between CAST(@DateMin as smallInt) AND GETDATE()
        Return 0

BEGIN
        EXEC spCarAfterDate '2018';
END
```

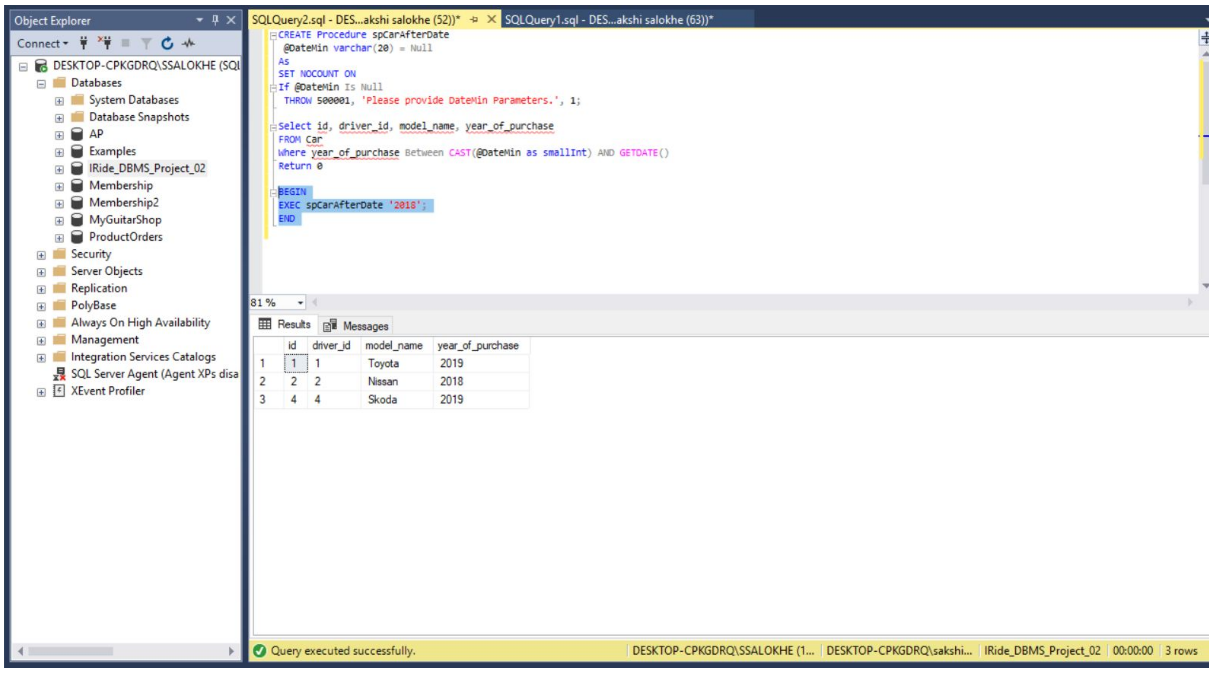
Screenshot:



```
CREATE Procedure spCarAfterDate
 @DateMin varchar(20) = Null
As
SET NOCOUNT ON
If @DateMin Is Null
 THROW 500001, 'Please provide DateMin Parameters.', 1;

Select id, driver_id, model_name, year_of_purchase
FROM Car
Where year_of_purchase Between CAST(@DateMin as smallInt) AND GETDATE()
Return 0
```

```
Commands completed successfully.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows



```sql
CREATE Procedure spCarAfterDate
 @DateMin varchar(20) = Null
As
SET NOCOUNT ON
If @DateMin Is Null
 THROW 500001, 'Please provide DateMin Parameters.', 1;

Select id, driver_id, model_name, year_of_purchase
FROM Car
Where year_of_purchase Between CAST(@DateMin as smallInt) AND GETDATE()
Return 0

BEGIN
EXEC spCarAfterDate '2018';
END
```

	id	driver_id	model_name	year_of_purchase
1	1	1	Toyota	2019
2	2	2	Nissan	2018
3	4	4	Skoda	2019

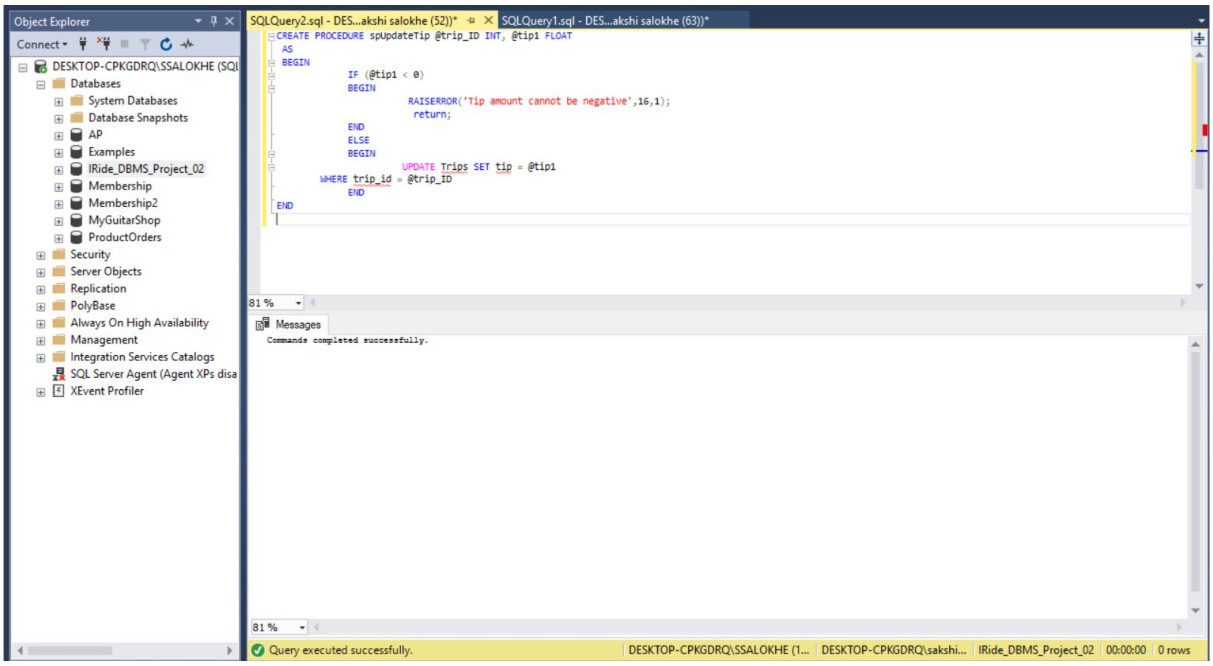
Query executed successfully.

Remark:
Thus we created a stored procedure to get the details of a car purchased after the given year.

2: To update the tip amount given by a customer if he wants to.

```
CREATE PROCEDURE spUpdateTip @trip_ID INT, @tip1 FLOAT
 AS
 BEGIN
            IF (@tip1 < 0)
            BEGIN
                      RAISERROR('Tip amount cannot be negative',16,1);
                       return;
            END
            ELSE
            BEGIN
                     UPDATE Trips SET tip = @tip1
                      WHERE trip_id = @trip_ID
            END
END

BEGIN
EXEC spUpdateTip 1,3.8;
END

select * from Trips where trip_id = 1
```

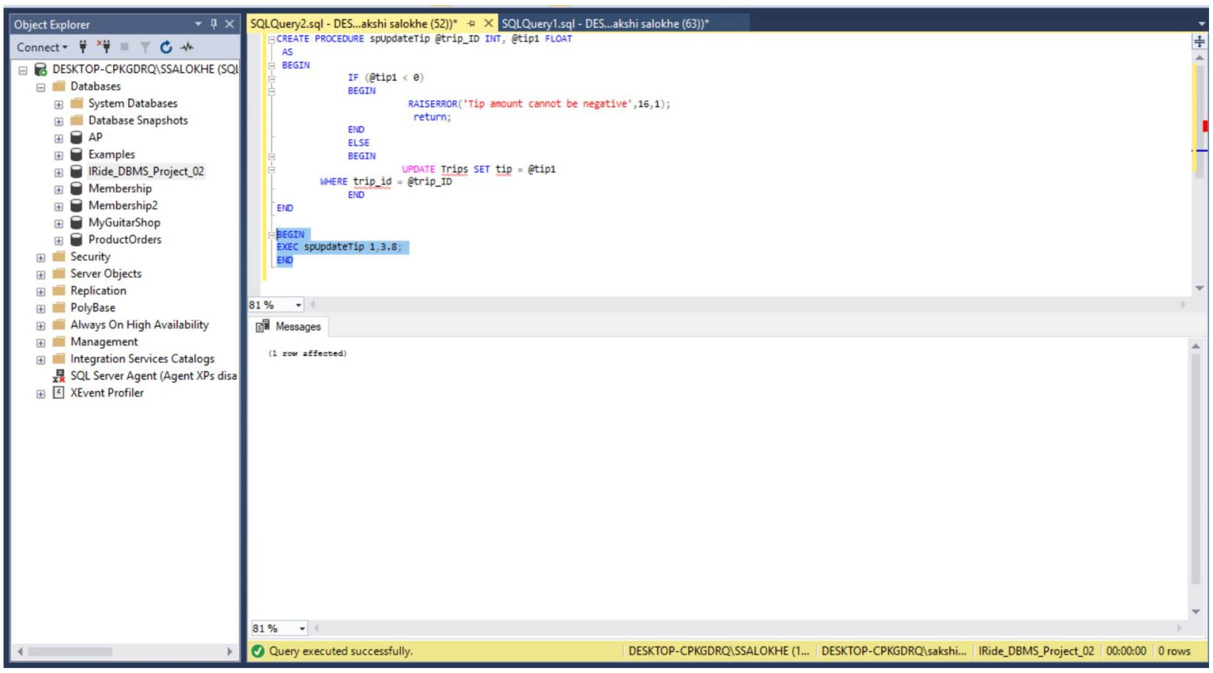
Screenshot:



```
CREATE PROCEDURE spUpdateTip @trip_ID INT, @tip1 FLOAT
 AS
 BEGIN
            IF (@tip1 < 0)
            BEGIN
                      RAISERROR('Tip amount cannot be negative',16,1);
                       return;
            END
            ELSE
            BEGIN
                     UPDATE Trips SET tip = @tip1
     WHERE trip_id = @trip_ID
            END
END
```

```
Commands completed successfully.
```

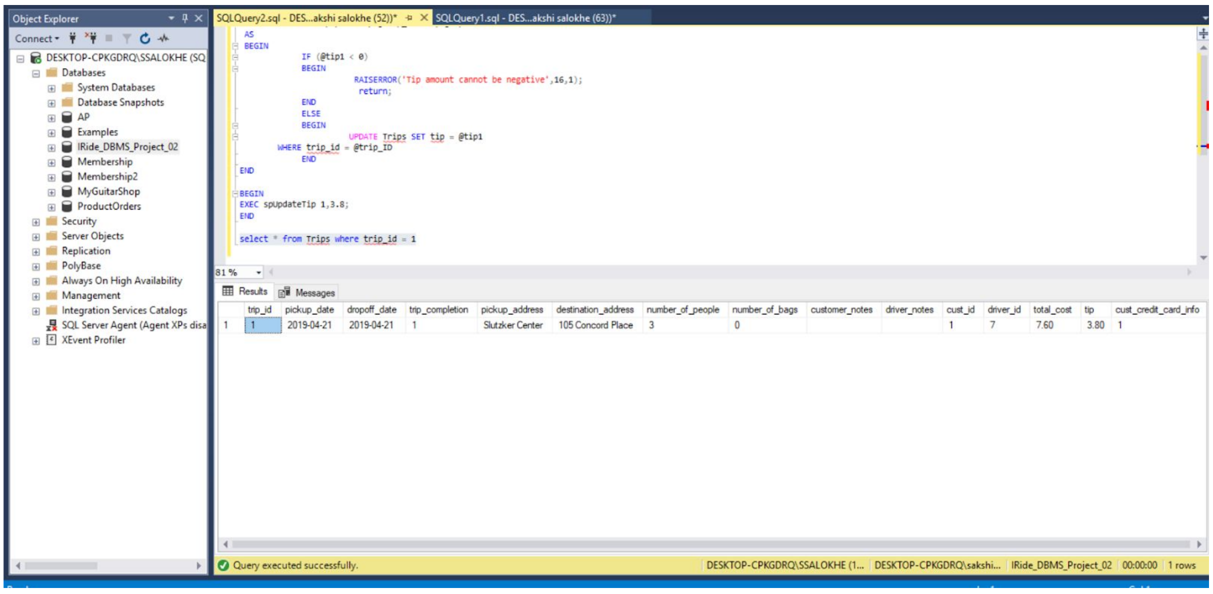
Query executed successfully.



```
CREATE PROCEDURE spUpdateTip @trip_ID INT, @tip1 FLOAT
 AS
 BEGIN
             IF (@tip1 < 0)
             BEGIN
                         RAISERROR('Tip amount cannot be negative',16,1);
                          return;
             END
             ELSE
             BEGIN
                         UPDATE Trips SET tip = @tip1
        WHERE trip_id = @trip_ID
             END
END

BEGIN
EXEC spUpdateTip 1,3.8;
END
```

```
(1 row affected)
```

Query executed successfully.



```
 AS
 BEGIN
             IF (@tip1 < 0)
             BEGIN
                         RAISERROR('Tip amount cannot be negative',16,1);
                          return;
             END
             ELSE
             BEGIN
                         UPDATE Trips SET tip = @tip1
        WHERE trip_id = @trip_ID
             END
END

BEGIN
EXEC spUpdateTip 1,3.8;
END

select * from Trips where trip_id = 1
```

	trip_id	pickup_date	dropoff_date	trip_completion	pickup_address	destination_address	number_of_people	number_of_bags	customer_notes	driver_notes	cust_id	driver_id	total_cost	tip	cust_credit_card_info
1	1	2019-04-21	2019-04-21	1	Slutzker Center	105 Concord Place	3	0			1	7	7.60	3.80	1

Query executed successfully.

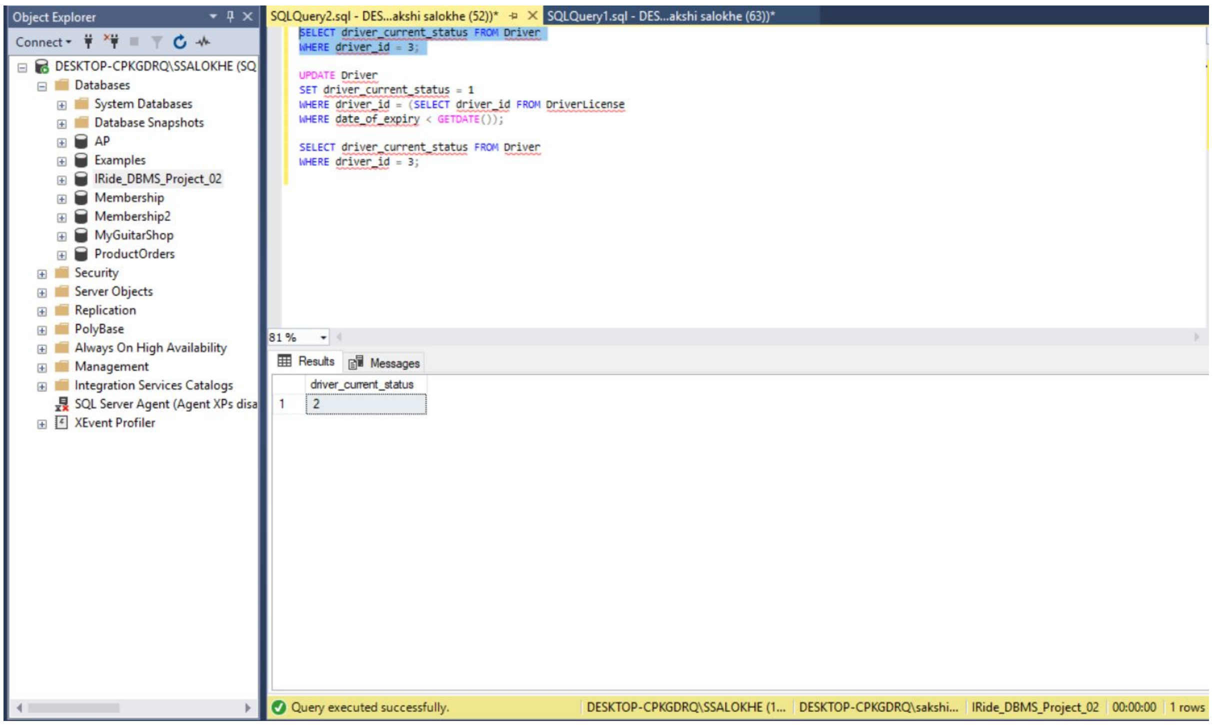
Remark:
Thus we created a stored procedure to update the tip amount by a customer for a particular trip.

Scripts

1: to make a driver inactive if his license has expired and he did not renew it or fails to update the information and latest documents.

```
SELECT driver_current_status FROM Driver
WHERE driver_id = 3;

UPDATE Driver
SET driver_current_status = 1
WHERE driver_id = (SELECT driver_id FROM DriverLicense
                   WHERE date_of_expiry < GETDATE());

SELECT driver_current_status FROM Driver
WHERE driver_id = 3;
```

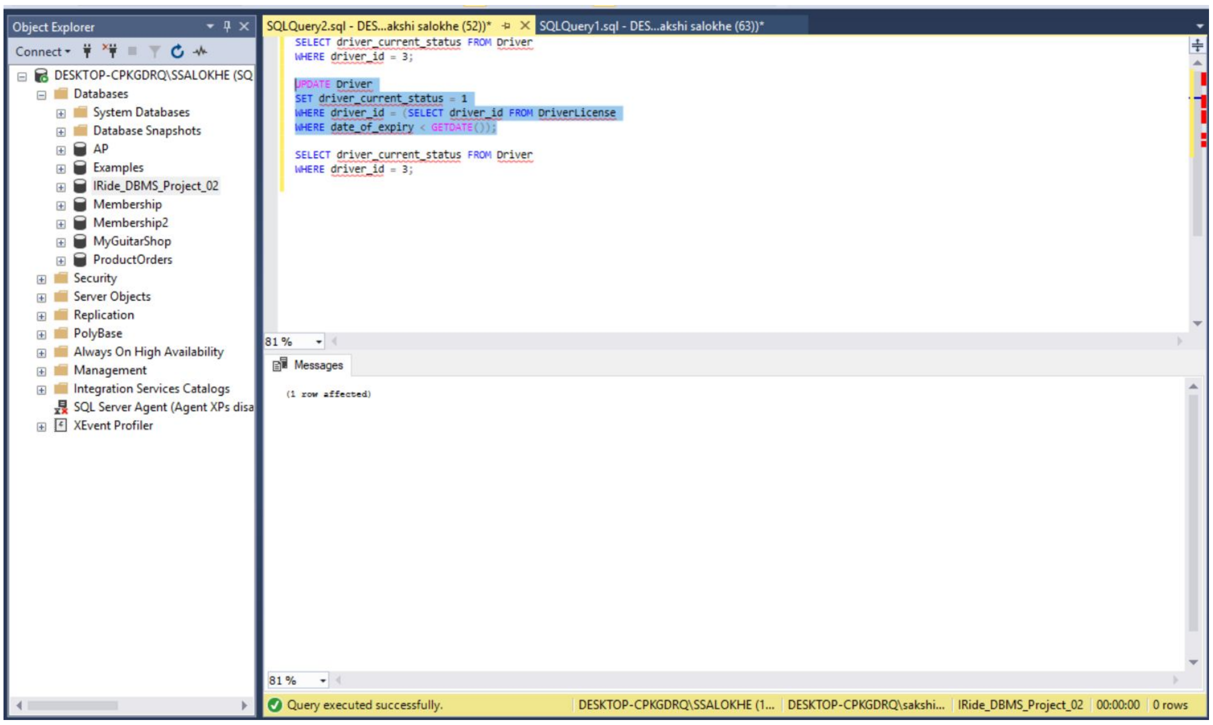
Screenshot:



```
SELECT driver_current_status FROM Driver
WHERE driver_id = 3;

UPDATE Driver
SET driver_current_status = 1
WHERE driver_id = (SELECT driver_id FROM DriverLicense
WHERE date_of_expiry < GETDATE());

SELECT driver_current_status FROM Driver
WHERE driver_id = 3;
```

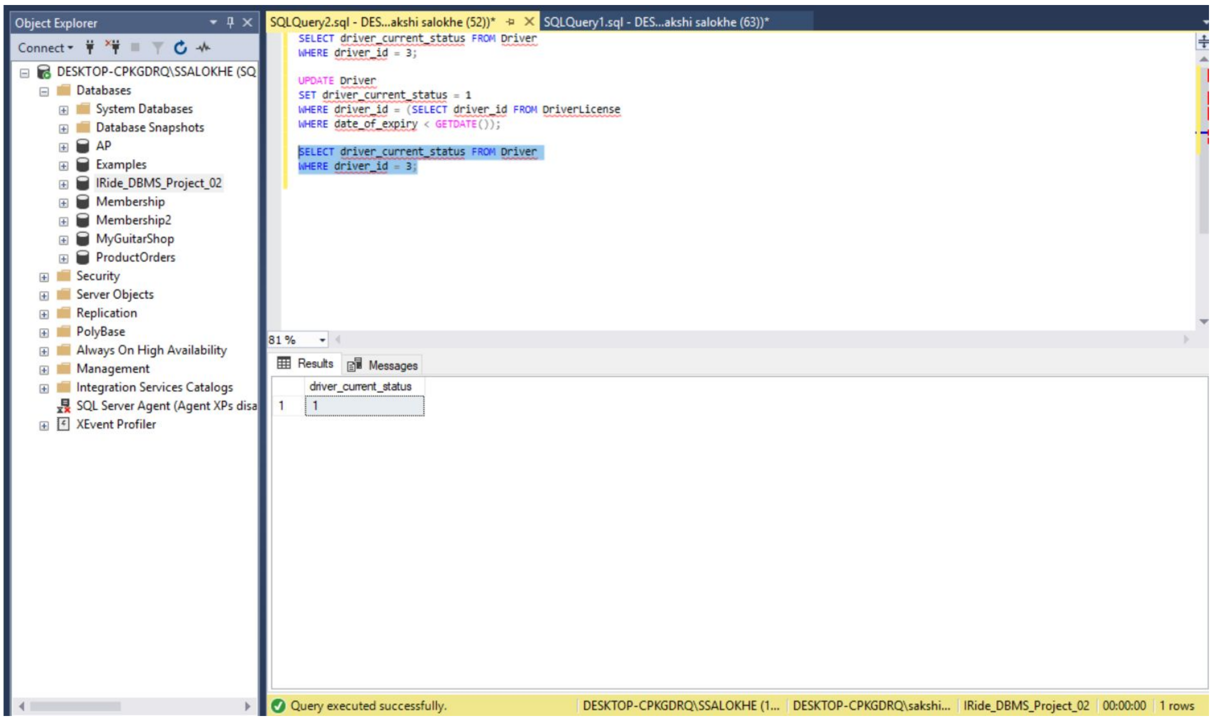
```
   driver_current_status
1  2
```

```
Query executed successfully.   DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 1 rows
```


```sql
SELECT driver_current_status FROM Driver
WHERE driver_id = 3;

UPDATE Driver
SET driver_current_status = 1
WHERE driver_id = (SELECT driver_id FROM DriverLicense
WHERE date_of_expiry < GETDATE());

SELECT driver_current_status FROM Driver
WHERE driver_id = 3;
```

```
(1 row affected)
```

```
Query executed successfully.    DESKTOP-CPKGDRQ\SSALOKHE (1...    DESKTOP-CPKGDRQ\sakshi...    IRide_DBMS_Project_02    00:00:00    0 rows
```


```sql
SELECT driver_current_status FROM Driver
WHERE driver_id = 3;

UPDATE Driver
SET driver_current_status = 1
WHERE driver_id = (SELECT driver_id FROM DriverLicense
WHERE date_of_expiry < GETDATE());

SELECT driver_current_status FROM Driver
WHERE driver_id = 3;
```

	driver_current_status
1	1

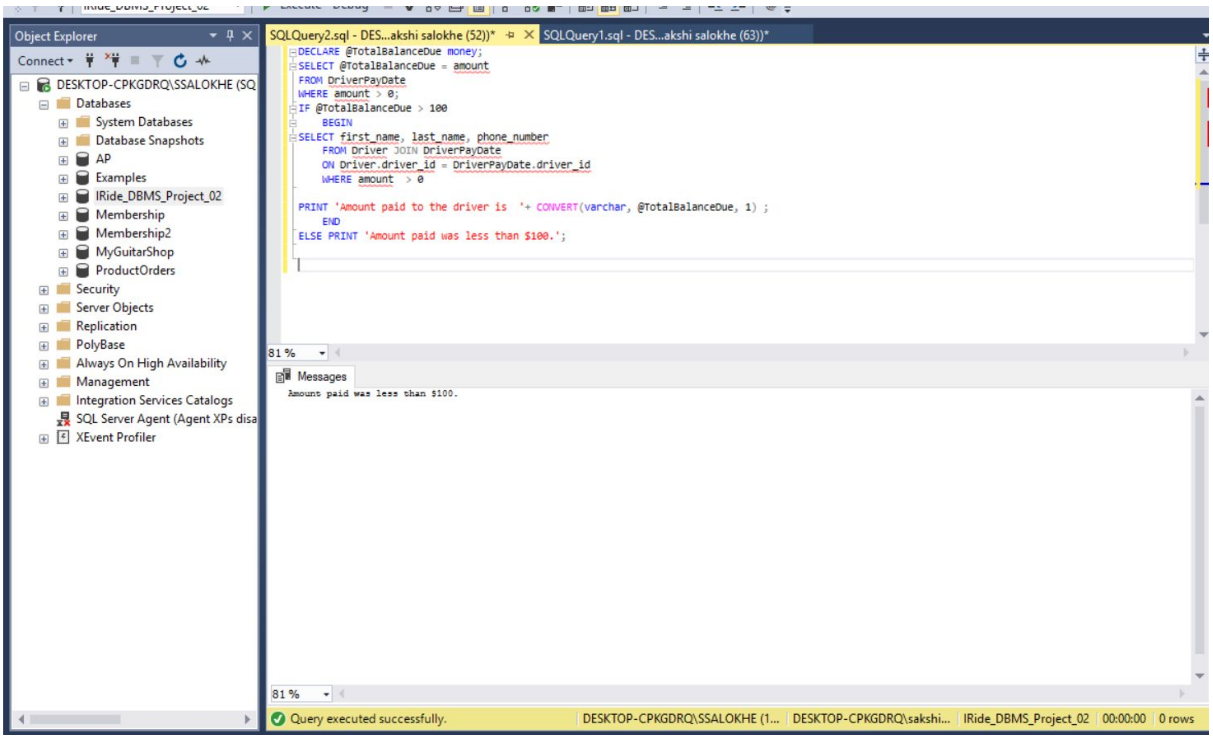
```
Query executed successfully.    DESKTOP-CPKGDRQ\SSALOKHE (1...    DESKTOP-CPKGDRQ\sakshi...    IRide_DBMS_Project_02    00:00:00    1 rows
```

Remark:
Thus we have written a script where we make a driver **inactive (set value to 1)** whose license has expired.

2: To find details of driver that got paid more than $100 in that pay cycle.

```
DECLARE @TotalBalanceDue money;
        SELECT @TotalBalanceDue = amount
        FROM DriverPayDate
        WHERE amount > 0;
IF @TotalBalanceDue > 100
        BEGIN
        SELECT first_name, last_name, phone_number
                FROM Driver JOIN DriverPayDate
                ON Driver.driver_id = DriverPayDate.driver_id
                WHERE amount  > 0

PRINT 'Amount paid to the driver is  '+ CONVERT(varchar, @TotalBalanceDue, 1) ;
        END
ELSE PRINT 'Amount paid was less than $100.';
```

Screenshot:



SQLQuery2.sql - DES...akshi salokhe (52))*

```
DECLARE @TotalBalanceDue money;
SELECT @TotalBalanceDue = amount
FROM DriverPayDate
WHERE amount > 0;
IF @TotalBalanceDue > 100
    BEGIN
SELECT first_name, last_name, phone_number
    FROM Driver JOIN DriverPayDate
    ON Driver.driver_id = DriverPayDate.driver_id
    WHERE amount  > 0

PRINT 'Amount paid to the driver is  '+ CONVERT(varchar, @TotalBalanceDue, 1) ;
    END
ELSE PRINT 'Amount paid was less than $100.';
```

Messages

```
Amount paid was less than $100.
```

Query executed successfully. | DESKTOP-CPKGDRQ\SSALOKHE (1... | DESKTOP-CPKGDRQ\sakshi... | IRide_DBMS_Project_02 | 00:00:00 | 0 rows

Remark:

Thus we have written a script that gives details of the drivers that got paid more than $100 and gives a message for those who got paid lesser than $100.

Project Analysis

1. In this project, we created a database **IRide_DBMS_Project_02** for a online cab booking service that is a competitor of Uber. In this database, we created tables to store the information of customers, drivers, their payment details, bank information to deposit paychecks, addresses, phone numbers, email addresses. We also created tables to store details of trips taken by customers, tips given, cost of trips, car information of a driver, insurances, licenses, etc.
2. Relevant data types were assigned to the tables to maintain data integrity. The tables were created in a way to get the database in 3rd normal form to avoid any confusion and redundancy of data.
3. Initially, when the database design was created, it was found that there were many columns that should not have been included or were included in a wrong way. We had to take care for the values and redesign the database. After redesigning, values were inserted in the tables.
4. The tables were then populated using data from real life. Since there are no application and real trips involved, care had to be taken of the date the driver joined IRide, the date when trips were taken, etc.
5. Since there is no real-time trip information, care had to be taken of the data that was inserted. The dates could not be allowed to collide with each other. For example, the drivers trip date cannot be before his joining date.
6. Separate tables were created for storing state names with ids, bank account types, drivers current status, etc. These values can be used in any tables without any dependency. Thus in case, the scope and number of tables increase in the future, some things will remain constant and we will not have to make changes in the core structure.
7. After getting the database to the 3rd normal form, values were inserted in the tables. Then to test the data we created various **views, scripts, functions and stored procedures**. The results that we got after executing them were exactly what was expected. Thus we could evaluate the correctness of the database and the inserted values.

Remarks

1. The **IRide database** was designed for online cab booking service similar to Uber. The database design was created on online tool **dbdesigner.net** and **draw.io** to create the Database Design and the ER Diagram respectively. When the database was designed, it was not in the 3rd normal form. But as the columns were divided and disintegrated to the atomic sizes, the final design of the database was created in the **3rd normal form**. Thus there was no data redundancy issues in the final design.
2. After finalizing the database design, **queries were written to create the database**. After the successful creation of the database, **sql queries were written to create the tables in the database**. **Relevant data types were assigned** to each column of all the tables.
3. **After the successful creation of the database tables, the tables were populated with relevant data**. Care was taken while inserting values in the tables so that no dates overlap and all the data seems to come from real applications.
4. Once the database was ready, validity and relevance of the database and data was checked using **views, functions, scripts and stored procedures.**
5. Views were created to:
    a. Display full names of the customers instead of first, middle and last names.
    b. Extract information of the customers who also work as drivers (isDriver = 1 column).
    c. Calculate total earning from each trip i.e. the sum of the trip cost and the tip given.
    d. Find the top 2 costliest trips using the 3rd view.
6. Functions were created to:
    a. Find the trips that occured in a given range of dates.
    b. Find the credit card that was used to pay for that particular trip. This was done to find the credit card when a user has multiple cards in his account.
7. Stored Procedures were created to:
    a. Find the details of cars that were purchased after a given year.
    b. Update tip amount by a customer.
8. Scripts were written to:
    a. Make a driver inactive if his license got expired.
    b. Find the detail of driver if the amount paid to him as salary was greater than $100. If the amount was less than $100, then a message is displayed.
9. Thus the data was tested properly and the results were exactly what was expected. Thus the **validity of data and the database** and its security was ensured.