Uber Database Design

CSE 581: Introduction to Database

Saumya Sharma | 321573561

Contents

1. Abstract……3
2. Relational Diagram……4
3. Entity Relation Diagram……9
4. Implementation……10
5. Testing Views……18
6. Testing Functions……20
7. Testing Stored Procedures……22
8. Testing Scripts……24
9. Conclusion……25

Abstract

Uber Technologies Inc. is a transportation network company (TNC) headquartered in San Francisco, California. Uber offers services including peer-to-peer ridesharing, ride service hailing, food delivery, and a bicycle-sharing system. The company has operations in 785 metropolitan areas worldwide. In this project, we try to emulate the database used by Uber for their operations.

Uber has 3 main stakeholders:

1. Customer: A person who needs to be picked up at a certain location and dropped off at a certain location
2. Driver: A person who has been hired by Uber to drive their car around a specific area to provide rides for people who request them.
3. Administrator: A person who works on the backend of the system and oversees the operation of business

Relational Diagram

The new database named Uber is defined using

```
CREATE DATABASE Uber;
USE Uber;
```

Customer Table: The customer table has information about the customers who use the Uber application. The following columns have been defined for the Customer Table.

```
CREATE TABLE Customers
(
  customer_id   varchar(15)     AUTO_INCREMENT PRIMARY KEY,
  fname         varchar(50)     NOT NULL,
  lname         varchar(50)     NOT NULL,
  phone_num     varchar(10)     NOT NULL,
  card_num      bigint          NOT NULL,
  home_address  varchar(100)    NULL,
  active        bit             default 1,
  customer_rating_id int        NOT NULL
);
```

The Customers table has two foreign keys :
Card_num which links to the CardDetails table
Customer_rating_id which links to CustomerRatings table

The customer_id is the primary key. No value in customer table is allowed to be null except home_address. The active column says 1 when the customer is active and 0 when inactive.

CardDetails Table: This table contains information about the payment methods used by the customer. One customer can have multiple cards, hence one to many relationship.

```
CREATE TABLE CardDetails
(
  card_num          bigint        NOT NULL,
  card_holder_name  varchar(50) NOT NULL,
  expiry            date          NOT NULL
);
```

The primary key here is the card_num.

Driver Table: This table contains information regarding the drivers in Uber. Below are the columns defined in the table. They contain information such as driver license, bank information, insurance information etc which have their own tables as defined further.

```
CREATE TABLE Driver
(
  driver_id     varchar(15)     AUTO_INCREMENT PRIMARY KEY,
```

```
  fname        varchar(50)      NOT NULL,
  lname        varchar(50)      NOT NULL,
  license_id    varchar(7)       NOT NULL,
  phone_num     varchar(10)      NOT NULL,
  ssn          varchar(12)      NOT NULL,
  home_address  varchar(100)     NULL,
  bank_id       varchar(10)      NOT NULL,
  insurance_id  int              NOT NULL,
  license_plate varchar(10)      NOT NULL,
  driver_status text             NOT NULL,
  driver_rating_id int           NOT NULL,
);
```

The driver_id is the primary key here and has 5 foreign keys namely license_id which links to LicenseInfo, insurance_id which links to InsuranceInfo, bank_id which links to bank_info,license_plate which links to DriverCar and driver_rating_id which links to DriverRatings

Ride Table: This table links the customer and the driver. When a customer books a cab, all the information is recorded in this database.

```
CREATE TABLE Ride
(
  ride_id          varchar(15)    NOT NULL,
  pickup_location    varchar(50)    NOT NULL,
  drop_location      varchar(50)    NOT NULL,
  is_cancelled       bit            default 1,
  customer_id        varchar(15)    NOT NULL,
  driver_id          varchar(15)    NOT NULL,
  total_cost         decimal(8,2)   NOT NULL,
  ride_date          datetime       NOT NULL,
);
```

This table has two foreign keys, customer_id which links to Customers table and driver_id which links to Driver table.

DriverCar Table: This table contains the information about the car owned by the driver. In this database design, one driver owns one car. Below are the columns defined in the database.

```
CREATE TABLE DriverCar
(
  license_plate       varchar(10)     NOT NULL,
  number_of_bags      int             NOT NULL,
  car_model           varchar(50)     NOT NULL,
  number_of_customers varchar(15)     NOT NULL
);
```

CustomerRatings Table: This table contains the rating given by the customer to the driver for a particular ride.

```
CREATE TABLE CustomerRatings
(
  customer_rating_id int    NOT NULL,
  driver_rating     int    CHECK (driver_rating>0 AND driver_rating<6),
  ride_id           varchar(15)    NOT NULL,
);
```

DriverRatings Table: This table contains the rating given by the driver to the customer for a particular ride.

```
CREATE TABLE DriverRatings
(
  driver_rating_id     int    NOT NULL,
  customer_rating      int    CHECK (customer_rating>0 AND customer_rating<6),
  ride_id              varchar(15)    NOT NULL,
);
```

LicenseInfo Table: This table contains information about the driver's license.

```
CREATE TABLE LicenseInfo
(
  license_id          varchar(7) NOT NULL,
  issue_date          date   NOT NULL,
  expiry              date   NOT NULL
);
```

InsuranceInfo Table: This table contains information about the Insurance held by Driver.

```
CREATE TABLE InsuranceInfo
(
  insurance_id        int NOT NULL,
  issue_date          date   NOT NULL,
  expiry              date   NOT NULL,
  company_name        varchar(20) NOT NULL
);
```

BankInfo Table: This table contains bank details about the car driver.

```
CREATE TABLE BankInfo
(
  bank_id             varchar(10)    NOT NULL,
  bank_name           varchar(10)   NOT NULL,
  account_num         varchar(30)   NOT NULL,
  account_type        text   NOT NULL
);
```

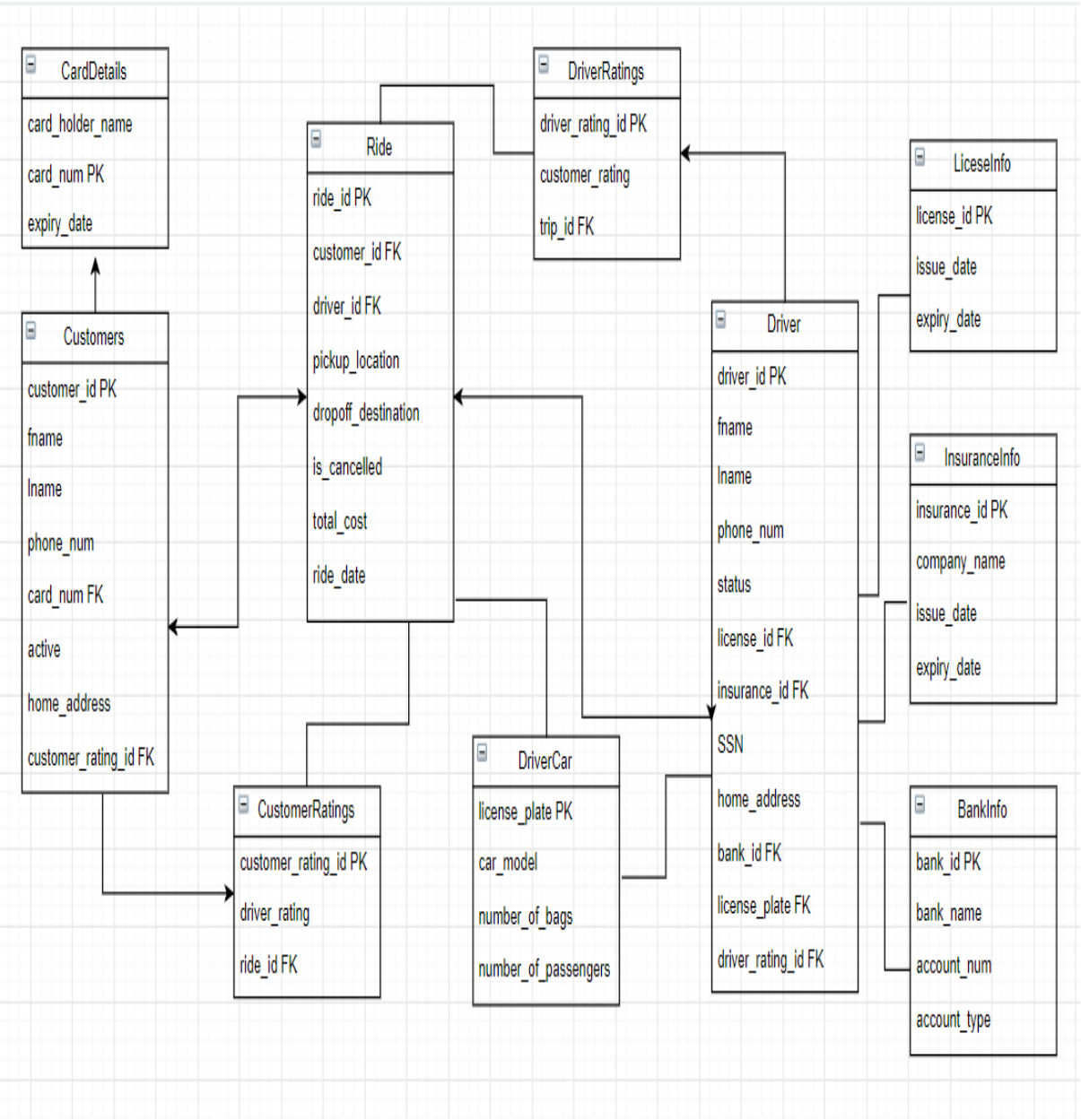
Adding Constrains:

Add the primary keys to the tables:

```
ALTER TABLE Customers ADD PRIMARY KEY (customer_id);
ALTER TABLE Driver ADD PRIMARY KEY (driver_id);
ALTER TABLE Ride ADD PRIMARY KEY (ride_id);
ALTER TABLE CardDetails ADD PRIMARY KEY (card_num);
ALTER TABLE DriverCar ADD PRIMARY KEY (license_plate);
ALTER TABLE CustomerRatings ADD PRIMARY KEY (customer_rating_id);
ALTER TABLE DriverRatings ADD PRIMARY KEY (driver_rating_id);
ALTER TABLE LicenseInfo ADD PRIMARY KEY (license_id);
ALTER TABLE InsuranceInfo ADD PRIMARY KEY (insurance_id);
ALTER TABLE BankInfo ADD PRIMARY KEY (bank_id);
```

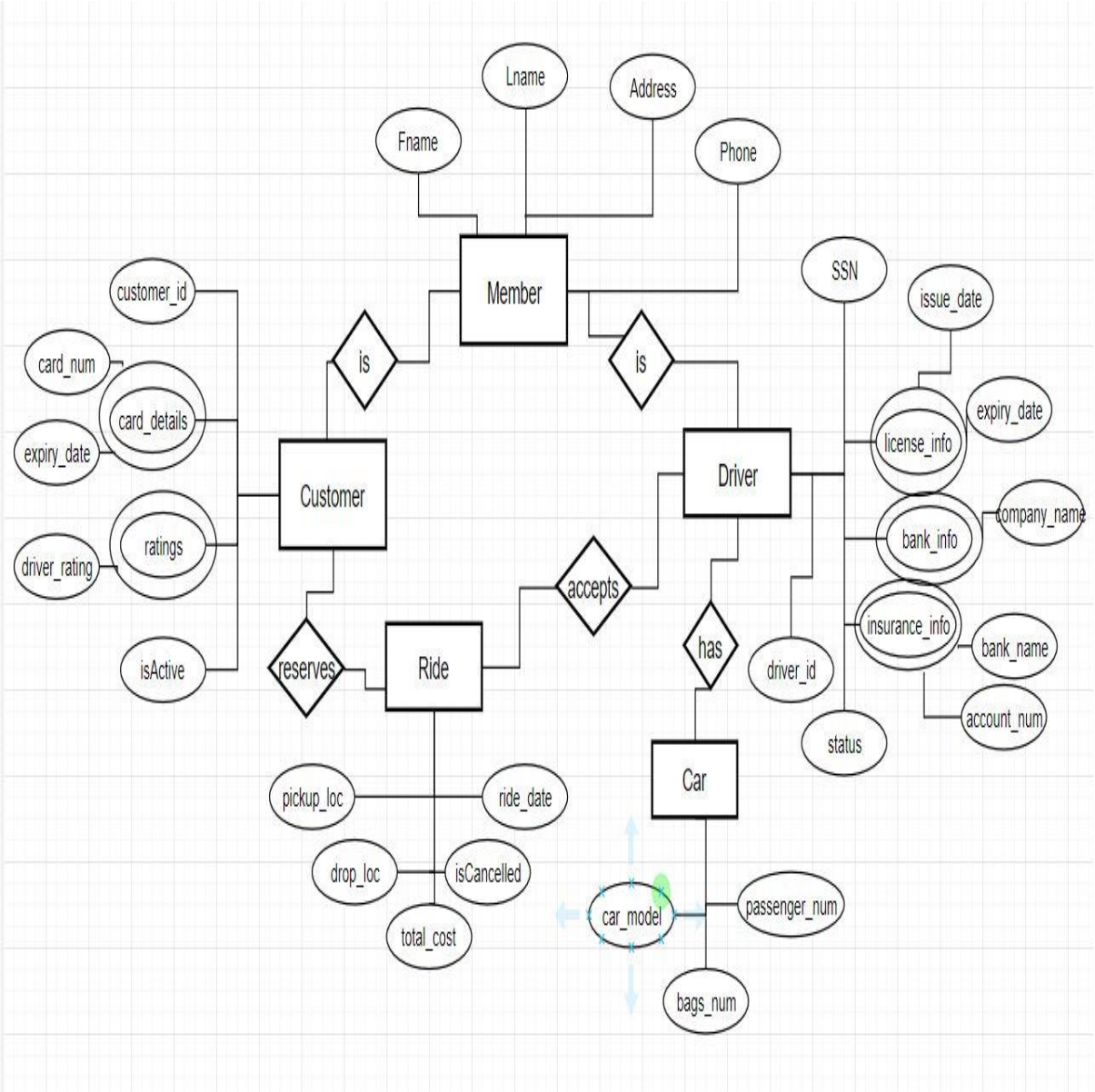
Add foreign keys to the tables:

```
ALTER TABLE Customers ADD CONSTRAINT FK_CustomerRatings FOREIGN KEY
(customer_rating_id) REFERENCES CustomerRatings (customer_rating_id);
ALTER TABLE Customers ADD CONSTRAINT FK_CardDetails FOREIGN KEY (card_num)
REFERENCES CardDetails (card_num);
ALTER TABLE Driver ADD CONSTRAINT FK_LicenseInfo FOREIGN KEY (license_id)
REFERENCES LicenseInfo (license_id);
ALTER TABLE Driver ADD CONSTRAINT FK_InsuranceInfo FOREIGN KEY
(insurance_id) REFERENCES InsuranceInfo (insurance_id);
ALTER TABLE Driver ADD CONSTRAINT FK_BankInfo FOREIGN KEY (bank_id)
REFERENCES BankInfo (bank_id);
ALTER TABLE Driver ADD CONSTRAINT FK_DriverRatings FOREIGN KEY
(driver_rating_id) REFERENCES DriverRatings (driver_rating_id);
ALTER TABLE Ride ADD CONSTRAINT FK_Customer FOREIGN KEY (customer_id)
REFERENCES Customers (customer_id);
ALTER TABLE Ride ADD CONSTRAINT FK_Driver FOREIGN KEY (driver_id)
REFERENCES Driver (driver_id);
ALTER TABLE CustomerRatings ADD CONSTRAINT FK_Customer_Ratings FOREIGN
KEY (ride_id) REFERENCES Ride (ride_id);
ALTER TABLE DriverRatings ADD CONSTRAINT FK_Driver_Ratings FOREIGN KEY
(ride_id) REFERENCES Ride (ride_id);
```



Customers to CardDetails (One to Many)
Customers to Ride (Many to Many)
Customers to CustomerRatings (One to Many)
Ride to CustomerRatings(One to One)
Ride to DriverCar(One to One)
Ride to Driver(Many to Many)
Ride to DriverRatings (One to One)
Driver to DriverRatings(One to Many)
Driver to LicenseInfo(One to One)
Driver to InsuranceInfo (One to One)
Driver to BankInfo (One to One)

Entity Relational Diagram



Inherited Relations:
The Customer IS A Member
The Driver IS A Member

The Customer Reserves a Ride

The Driver HAS a Car

The Driver ACCEPTS a customer Ride

Implementation

Inserting the Values to database:

```
INSERT INTO Customers (customer_id, fname, lname, phone_num,
card_num,home_address,active,customer_rating_id)
VALUES
  (1, 'Barbara', 'Litt', '4129990000', 1234567891011121,'34 ABC Street',1,2),      -- 1
  (2, 'James', 'Bond', '4129990001', 1123456789101112,'45 QAC Street',1,1),        -- 2
  (3, 'Kim', 'Kardashian', '4129990002', 2112345678910111,'98 WEN Street',1,4),    -- 3
  (4, 'Hana', 'Ko', '4129990003', 1211234567891011,'26 KOL Street',1,3),           -- 4
  (5, 'Patrick', 'Yang', '4129990004', 1121123456789101,'04 DER Street',1,6),      -- 5
  (6, 'Donald', 'Trump', '4129990005', 1112112345678910,'27 RED Street',1,5),      -- 6
  (7, 'Kendall', 'Jenner', '4129990006', 1011121123456789,'86 OPW Street',1,8),    -- 7
  (8, 'Victoria', 'Beckham', '4129990007', 9101112112345678,'89 GUI Street',1,7);  -- 8


INSERT INTO CardDetails  (card_num,card_holder_name,expiry)
VALUES
(1234567891011121,'Barbara Litt',CAST('2024-01-01' AS DATE)),      --1
(1123456789101112,'James Bond',CAST('2024-01-01' AS DATE)),       --2
(2112345678910111,'Kim Kardashian',CAST('2024-01-01' AS DATE)),           --3
(1211234567891011,'Harvey Specter',CAST('2024-01-01' AS DATE)),           --4
(1121123456789101,'Mike Ross',CAST('2024-01-01' AS DATE)),         --5
(1112112345678910,'Kim Jhon Un',CAST('2024-01-01' AS DATE)),     --6
(1011121123456789,'Spongebob',CAST('2024-01-01' AS DATE)),        --7
(9101112112345678,'Karlie Kloss',CAST('2024-01-01' AS DATE));      --8


INSERT INTO Driver (driver_id, fname, lname, license_id,
phone_num,ssn,home_address,bank_id,insurance_id,license_plate,driver_status,driver_r
ating_id)
VALUES
  (1, 'Rheanna', 'Westbrook', 'A1234','3154366685','345-374-3456','456 SED
Street','123EDF',8, '2FAST4U', 'AVAILABLE',5), --1
  (2, 'Loraine', 'Rowland', 'B903','6754366685','785-374-4567','457 JTG
Street','456TEF',7, '3HUA172', 'AVAILABLE',4),   --2
  (3, 'Nova', 'Denman', 'C456','8354366685','955-374-8654','904 LOI Street','905HTI',6,
'BB1B001', 'INACTIVE',3),      --3
  (4, 'Steven', 'Spilburgh', 'Q782','8904366685','115-374-1234','367 ERT
Street','345OPT',5, 'XCG6033', 'AVAILABLE',2),   --4
  (5, 'Mauricio', 'Wood', 'Y768','1344366685','905-374-0987','956 OPT
Street','356IPT',4, 'BFEJ681', 'AVAILABLE',1),    --5
  (6, 'Tom', 'Riddle', 'P092','3244366685','565-374-3456','893 TOP Street','789IOP',3,
'BMB6781', 'OFF-WORK',6),       --6
  (7, 'John', 'Deer', 'A760','5674366685','985-374-3456','278 EWR Street','345POY',2,
'GHG5269', 'AVAILABLE',7),    --7
  (8, 'Ritz', 'Carlton', 'E321','9874366685','565-374-4567','123 TYU Street','907TIP',1,
'5A0J230', 'AVAILABLE',8);     --8
```

```
INSERT INTO Ride (ride_id, pickup_location, drop_location, is_cancelled,
customer_id,driver_id,total_cost,ride_date)
VALUES
  (1, 'Westfield', 'Hotel Pensylvania', 0,3,8,34.6,convert(datetime,'18-06-12 10:34:09
PM',5)), --1
  (2, 'Jamaica', 'Rowland', 0,6,7,7.8,convert(datetime,'18-06-12 10:34:09 PM',5)),   --2
  (3, 'Nova', 'Westbrook', 0,8,6,89.1,convert(datetime,'18-06-12 10:34:09 PM',5)),      --3
  (4, 'Side step', 'Stairs', 0,1,5,12.4,convert(datetime,'18-06-12 10:34:09 PM',5)),   --4
  (5, 'Bus stop', 'Forest', 1,4,4,25.7,convert(datetime,'18-06-12 10:34:09 PM',5)),    --5
  (6, 'Red Light', 'Desert', 0,2,3,9.0,convert(datetime,'18-06-12 10:34:09 PM',5)),       --6
  (7, 'Cros walk', 'Hotel Taj', 1,5,2,45.5,convert(datetime,'18-06-12 10:34:09 PM',5)),    --
7
  (8, 'Hotel Ritz', 'Hotel Leela', 0,7,1,32.8,convert(datetime,'18-06-12 10:34:09 PM',5));
--8

INSERT INTO DriverCar (license_plate, number_of_bags, car_model,
number_of_customers) VALUES
  ('2FAST4U',3, 'Ferrari-A3', 2),   --1
  ('3HUA172',7, 'BMW-E Series', 4),   --2
  ('BB1B001',3, 'Audi A8', 1),   --3
  ('XCG6033',3, 'Camry D7', 3),    --4
  ('BFEJ681',7, 'Ferrari-R3', 5),   --5
  ('BMB6781',7, 'Lamborghini-S1', 5),  --6
  ('GHG5269',7, 'Volkswagen-S3', 6),    --7
  ('5A0J230',7, 'Audi-A10', 2);  --8

  INSERT INTO CustomerRatings (customer_rating_id, driver_rating, ride_id ) VALUES
  (5,4, 3),   --1
  (2,5, 4),   --2
  (1,4, 5),   --3
  (8,5, 6),    --4
  (6,4, 7),   --5
  (7,4, 8),  --6
  (3,5, 1),    --7
  (4,5, 2);  --8

  INSERT INTO DriverRatings (driver_rating_id, customer_rating, ride_id ) VALUES
  (5,5, 3),   --1
  (2,4, 4),   --2
  (1,3, 5),   --3
  (8,4, 6),    --4
  (6,5, 7),   --5
  (7,5, 8),  --6
  (3,4, 1),    --7
  (4,5, 2);  --8
```

```
INSERT INTO LicenseInfo (license_id, issue_date, expiry ) VALUES
('A1234',CAST('2014-01-01' AS DATE), CAST('2024-01-01' AS DATE)),   --1
('B903',CAST('2014-10-02' AS DATE), CAST('2024-10-02' AS DATE)),   --2
('C456',CAST('2012-11-01' AS DATE), CAST('2022-11-01' AS DATE)),   --3
('Q782',CAST('2015-01-07' AS DATE), CAST('2025-01-07' AS DATE)),    --4
('Y768',CAST('2013-09-10' AS DATE), CAST('2023-09-10' AS DATE)),   --5
('P092',CAST('2014-08-17' AS DATE), CAST('2024-08-17' AS DATE)),  --6
('A760',CAST('2013-02-28' AS DATE), CAST('2023-02-28' AS DATE)),    --7
('E321',CAST('2009-05-23' AS DATE), CAST('2019-05-23' AS DATE));  --8


INSERT INTO InsuranceInfo (insurance_id, issue_date, expiry,company_name )
VALUES
(5,CAST('2014-01-01' AS DATE), CAST('2034-01-01' AS DATE),'ABC Alliance'),   --1
(2,CAST('2014-10-02' AS DATE), CAST('2034-10-02' AS DATE),'DEF Alliance'),   --2
(1,CAST('2012-11-01' AS DATE), CAST('2032-11-01' AS DATE),'GHI Alliance'),   --3
(8,CAST('2015-01-07' AS DATE), CAST('2035-01-07' AS DATE),'JKL Alliance'),    --4
(6,CAST('2013-09-10' AS DATE), CAST('2033-09-10' AS DATE),'MNO Alliance'),   --5
(7,CAST('2014-08-17' AS DATE), CAST('2034-08-17' AS DATE),'PQR Alliance'),  --6
(3,CAST('2013-02-28' AS DATE), CAST('2033-02-28' AS DATE),'STU Alliance'),    --7
(4,CAST('2009-05-23' AS DATE), CAST('2029-05-23' AS DATE),'VWX Alliance');  --8


INSERT INTO BankInfo (bank_id, bank_name, account_num,account_type ) VALUES
('123EDF','Chase Bank','123-456-6789','Checking'),   --1
('456TEF','BOFA Bank','456-456-6789','Saving'),    --2
('905HTI','BOFA Bank','754-456-6789','Saving'),    --3
('345OPT','Chase Bank','268-456-6789','Checking'),     --4
('356IPT','Chase Bank','908-456-6789','Checking'),    --5
('789IOP','AXA Bank','654-456-6789','Saving'),   --6
('345POY','AXA Bank','890-456-6789','Saving'),     --7
('907TIP','Key Bank','535-456-6789','Saving');  --8
```

```sql
select * from Customers;
```

LENOVO-PC\SQLEXP...- dbo.Customers | LENOVO-PC\SQLEXP...S.Uber - dbo.Ride | SQLQuery1.sql - L...VO-PC\Saumya (53))*

```sql
select * from Customers;
```

100 %

Results | Messages

	customer_id	fname	lname	phone_num	card_num	home_address	active	customer_rating_id
1	1	Barbara	Litt	4129990000	1234567891011121	34 ABC Street	1	2
2	2	James	Bond	4129990001	1123456789101112	45 QAC Street	1	1
3	3	Kim	Kardashian	4129990002	2112345678910111	98 WEN Street	1	4
4	4	Hana	Ko	4129990003	1211234567891011	26 KOL Street	1	3
5	5	Patrick	Yang	4129990004	1121123456789101	04 DER Street	1	6
6	6	Donald	Trump	4129990005	1112112345678910	27 RED Street	1	5
7	7	Kendall	Jenner	4129990006	1011121123456789	86 OPW Street	1	8
8	8	Victoria	Beckham	4129990007	9101112112345678	89 GUI Street	1	7

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```sql
select * from CustomerRatings;
```

LENOVO-PC\SQLEXP...- dbo.Customers | LENOVO-PC\SQLEXP...S.Uber - dbo.Ride | SQLQuery1.sql - L...VO-PC\Saumya (53))*

```sql
select * from CustomerRatings;
```

100 %

Results | Messages

	customer_rating_id	driver_rating	ride_id
1	1	4	5
2	2	5	4
3	3	5	1
4	4	5	2
5	5	4	3
6	6	4	7
7	7	4	8
8	8	5	6

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```sql
select * from Driver;
```

LENOVO-PC\SQLEXP...- dbo.Customers | LENOVO-PC\SQLEXP...S.Uber - dbo.Ride | SQLQuery1.sql - L...VO-PC\Saumya (53))*

100 %

Results | Messages

	driver_id	fname	lname	license_id	phone_num	ssn	home_address	bank_id	insurance_id	license_plate	driver_status	dr
1	1	Rheanna	Westbrook	A1234	3154366685	345-374-3456	456 SED Street	123EDF	8	2FAST4U	AVAILABLE	5
2	2	Loraine	Rowland	B903	6754366685	785-374-4567	457 JTG Street	456TEF	7	3HUA172	AVAILABLE	4
3	3	Nova	Denman	C456	8354366685	955-374-8654	904 LOI Street	905HTI	6	BB1B001	INACTIVE	3
4	4	Steven	Spilburgh	Q782	8904366685	115-374-1234	367 ERT Street	345OPT	5	XCG6033	AVAILABLE	2
5	5	Mauricio	Wood	Y768	1344366685	905-374-0987	956 OPT Street	356IPT	4	BFEJ681	AVAILABLE	1
6	6	Tom	Riddle	P092	3244366685	565-374-3456	893 TOP Street	789IOP	3	BMB6781	OFF-WORK	6
7	7	John	Deer	A760	5674366685	985-374-3456	278 EWR Street	345POY	2	GHG5269	AVAILABLE	7

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```sql
select * from DriverCar;
```

LENOVO-PC\SQLEXP...- dbo.Customers | LENOVO-PC\SQLEXP...S.Uber - dbo.Ride | SQLQuery1.sql - L...VO-PC\Saumya (53))*

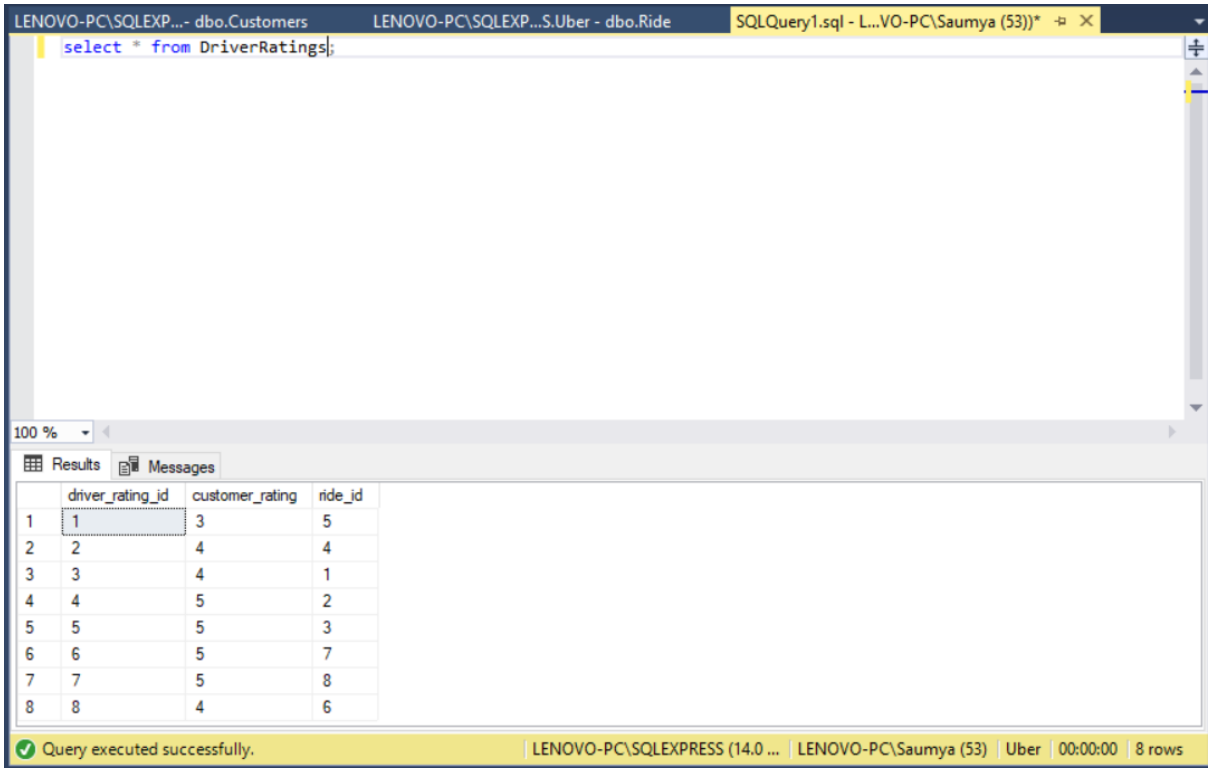
100 %

Results | Messages

	license_plate	number_of_bags	car_model	number_of_customers
1	2FAST4U	3	Ferrari-A3	2
2	3HUA172	7	BMW-E Series	4
3	5A0J230	7	Audi-A10	2
4	BB1B001	3	Audi A8	1
5	BFEJ681	7	Ferrari-R3	5
6	BMB6781	7	Lamborghini-S1	5
7	GHG5269	7	Volkswagen-S3	6
8	XCG6033	3	Camry D7	3

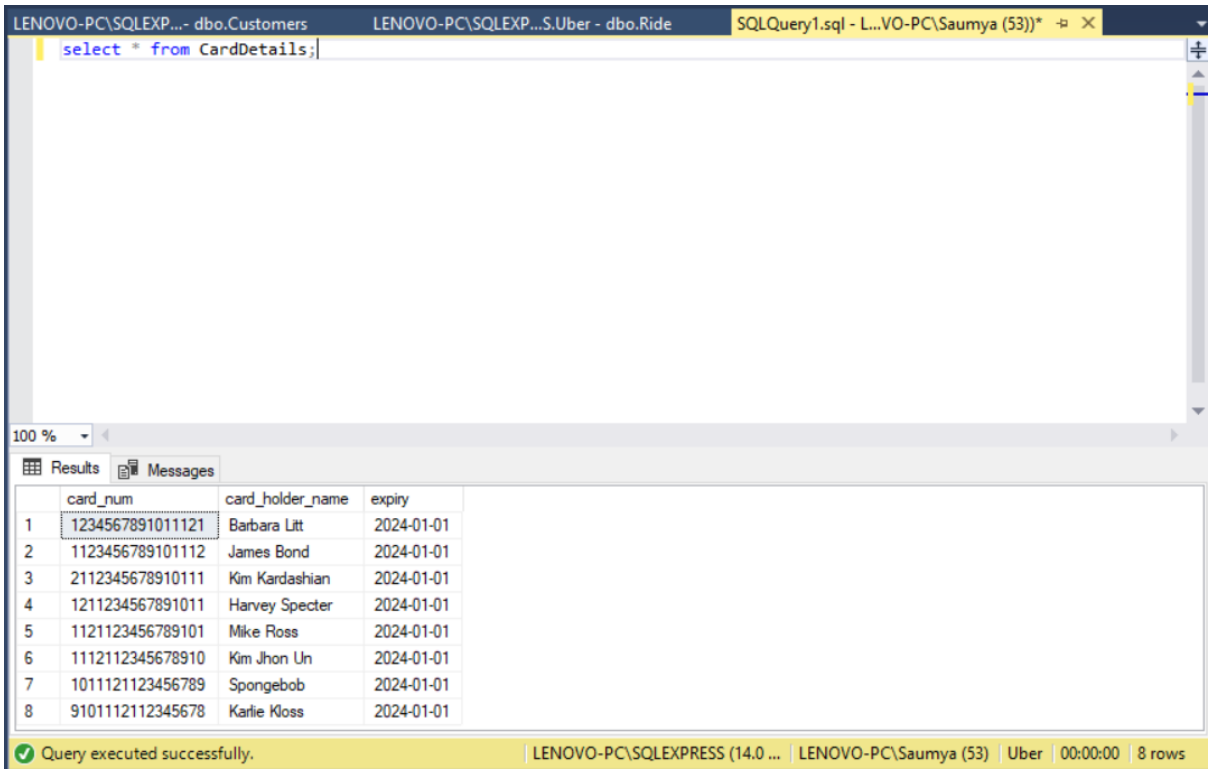
Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```sql
select * from DriverRatings;
```



	driver_rating_id	customer_rating	ride_id
1	1	3	5
2	2	4	4
3	3	4	1
4	4	5	2
5	5	5	3
6	6	5	7
7	7	5	8
8	8	4	6

```sql
select * from CardDetails;
```



	card_num	card_holder_name	expiry
1	1234567891011121	Barbara Litt	2024-01-01
2	1123456789101112	James Bond	2024-01-01
3	2112345678910111	Kim Kardashian	2024-01-01
4	1211234567891011	Harvey Specter	2024-01-01
5	1121123456789101	Mike Ross	2024-01-01
6	1112112345678910	Kim Jhon Un	2024-01-01
7	1011121123456789	Spongebob	2024-01-01
8	9101112112345678	Karlie Kloss	2024-01-01

```sql
select * from LicenseInfo;
```

LENOVO-PC\SQLEXP...- dbo.Customers | LENOVO-PC\SQLEXP...S.Uber - dbo.Ride | SQLQuery1.sql - L...VO-PC\Saumya (53))*

100 %

Results | Messages

	license_id	issue_date	expiry
1	A1234	2014-01-01	2024-01-01
2	B903	2014-10-02	2024-10-02
3	C456	2012-11-01	2022-11-01
4	Q782	2015-01-07	2025-01-07
5	Y768	2013-09-10	2023-09-10
6	P092	2014-08-17	2024-08-17
7	A760	2013-02-28	2023-02-28
8	E321	2009-05-23	2019-05-23

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```sql
select * from InsuranceInfo;
```

LENOVO-PC\SQLEXP...- dbo.Customers | LENOVO-PC\SQLEXP...S.Uber - dbo.Ride | SQLQuery1.sql - L...VO-PC\Saumya (53))*

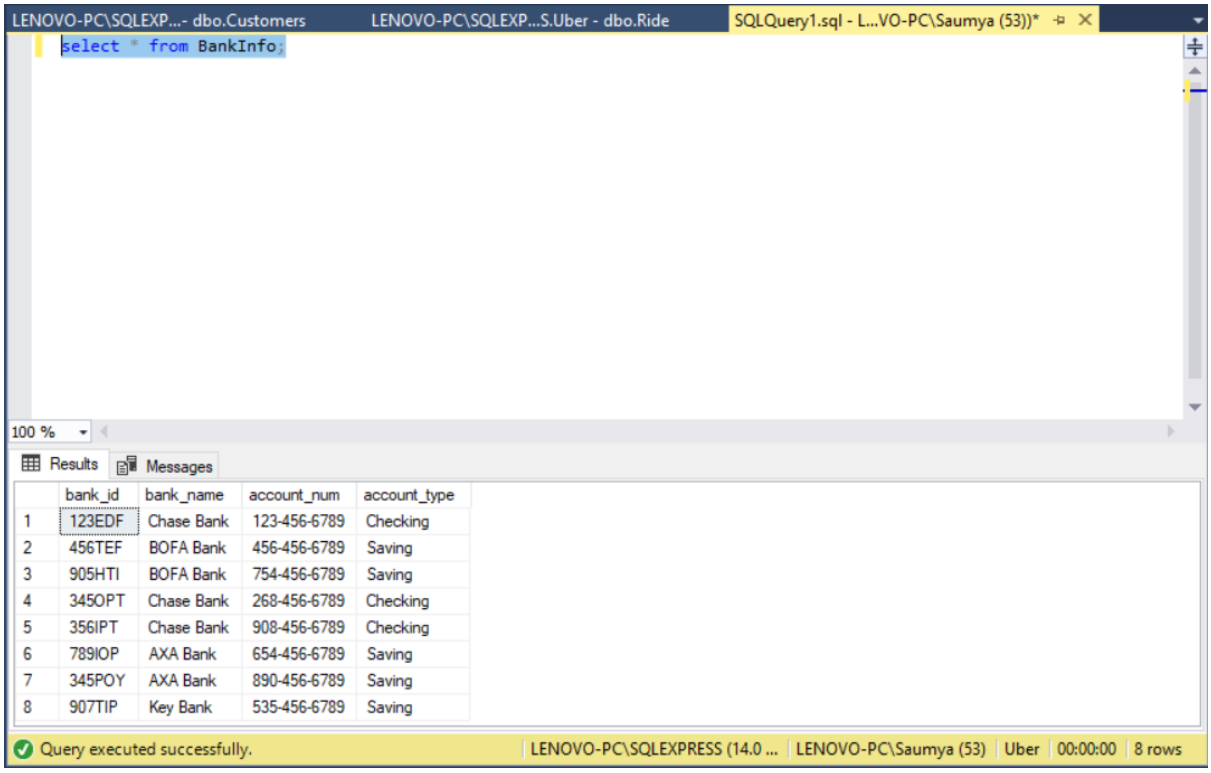
100 %

Results | Messages

	insurance_id	issue_date	expiry	company_name
1	1	2012-11-01	2032-11-01	GHI Alliance
2	2	2014-10-02	2034-10-02	DEF Alliance
3	3	2013-02-28	2033-02-28	STU Alliance
4	4	2009-05-23	2029-05-23	VWX Alliance
5	5	2014-01-01	2034-01-01	ABC Alliance
6	6	2013-09-10	2033-09-10	MNO Alliance
7	7	2014-08-17	2034-08-17	PQR Alliance
8	8	2015-01-07	2035-01-07	JKL Alliance

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```sql
select * from BankInfo;
```



	bank_id	bank_name	account_num	account_type
1	123EDF	Chase Bank	123-456-6789	Checking
2	456TEF	BOFA Bank	456-456-6789	Saving
3	905HTI	BOFA Bank	754-456-6789	Saving
4	345OPT	Chase Bank	268-456-6789	Checking
5	356IPT	Chase Bank	908-456-6789	Checking
6	789IOP	AXA Bank	654-456-6789	Saving
7	345POY	AXA Bank	890-456-6789	Saving
8	907TIP	Key Bank	535-456-6789	Saving

Testing Views

Creating simple views from this database:

```
create view
CustomerNames
as select fname+' '+lname as FullName from Customers;
select * from CustomerNames;
```

Results | Messages

	FullName
1	Barbara Litt
2	James Bond
3	Kim Kardashian
4	Hana Ko
5	Patrick Yang
6	Donald Trump
7	Kendall Jenner
8	Victoria Beckham

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```
create view
DriverNames
as select fname+' '+lname as FullName from Driver;
select * from DriverNames;
```

Results | Messages

	FullName
1	Rheanna Westbrook
2	Loraine Rowland
3	Nova Denman
4	Steven Spilburgh
5	Mauricio Wood
6	Tom Riddle
7	John Deer
8	Ritz Carlton

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

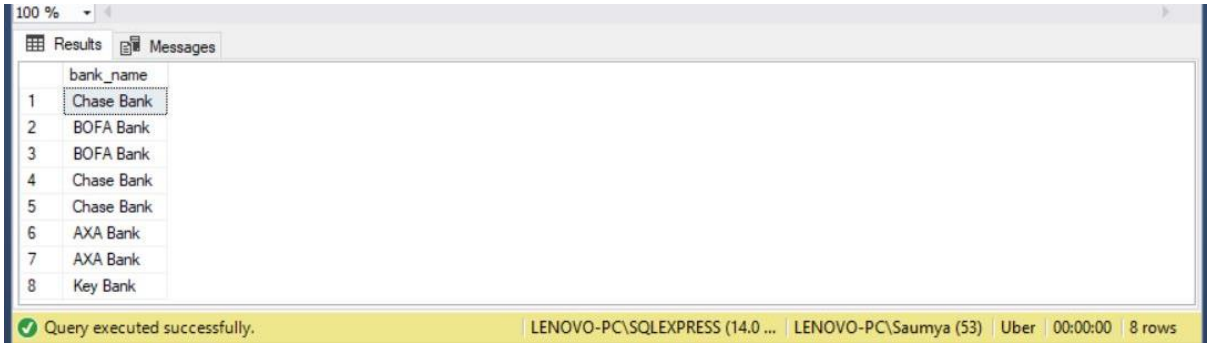
```
create view
Cars
as select car_model from DriverCar;
select * from Cars;
```

Results | Messages

	car_model
1	Ferrari-A3
2	BMW-E Series
3	Audi-A10
4	Audi A8
5	Ferrari-R3
6	Lamborghini-S1
7	Volkswagen-S3
8	Camry D7

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```
create view
BankNames
as select bank_name from BankInfo;
select * from BankNames;
```



100 %

Results | Messages

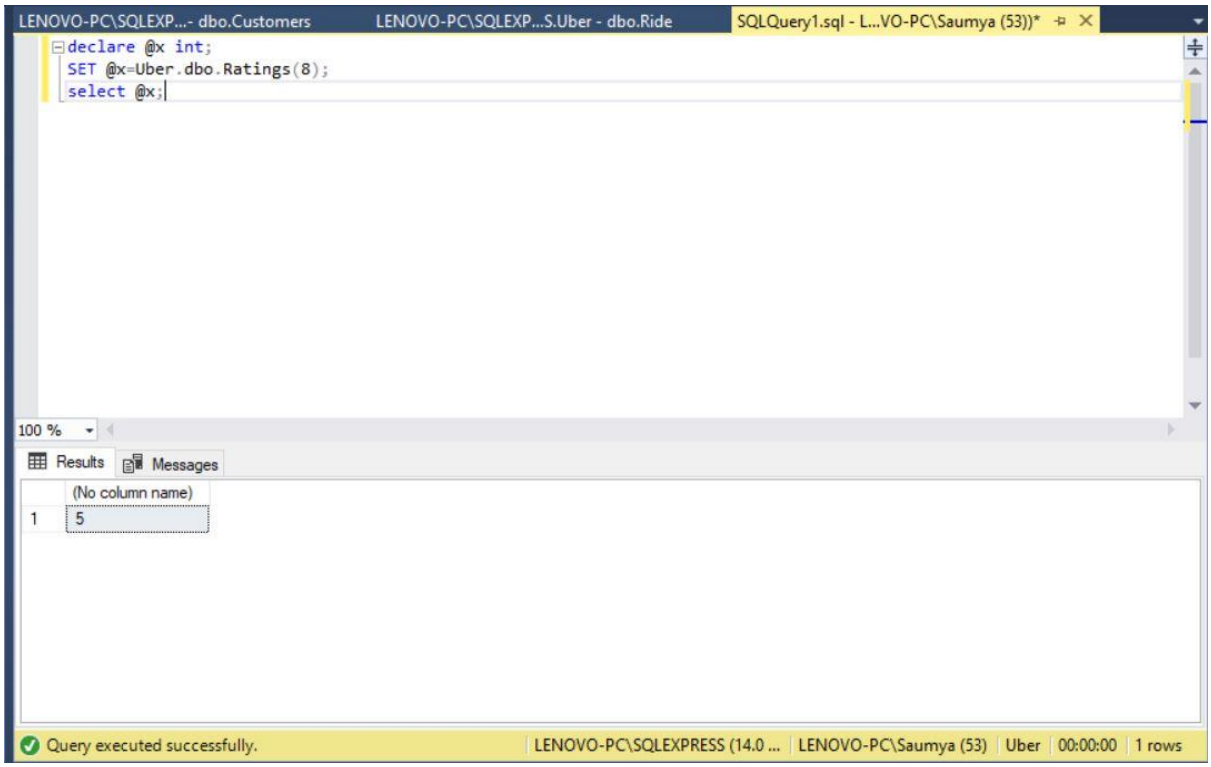
	bank_name
1	Chase Bank
2	BOFA Bank
3	BOFA Bank
4	Chase Bank
5	Chase Bank
6	AXA Bank
7	AXA Bank
8	Key Bank

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

Testing Functions

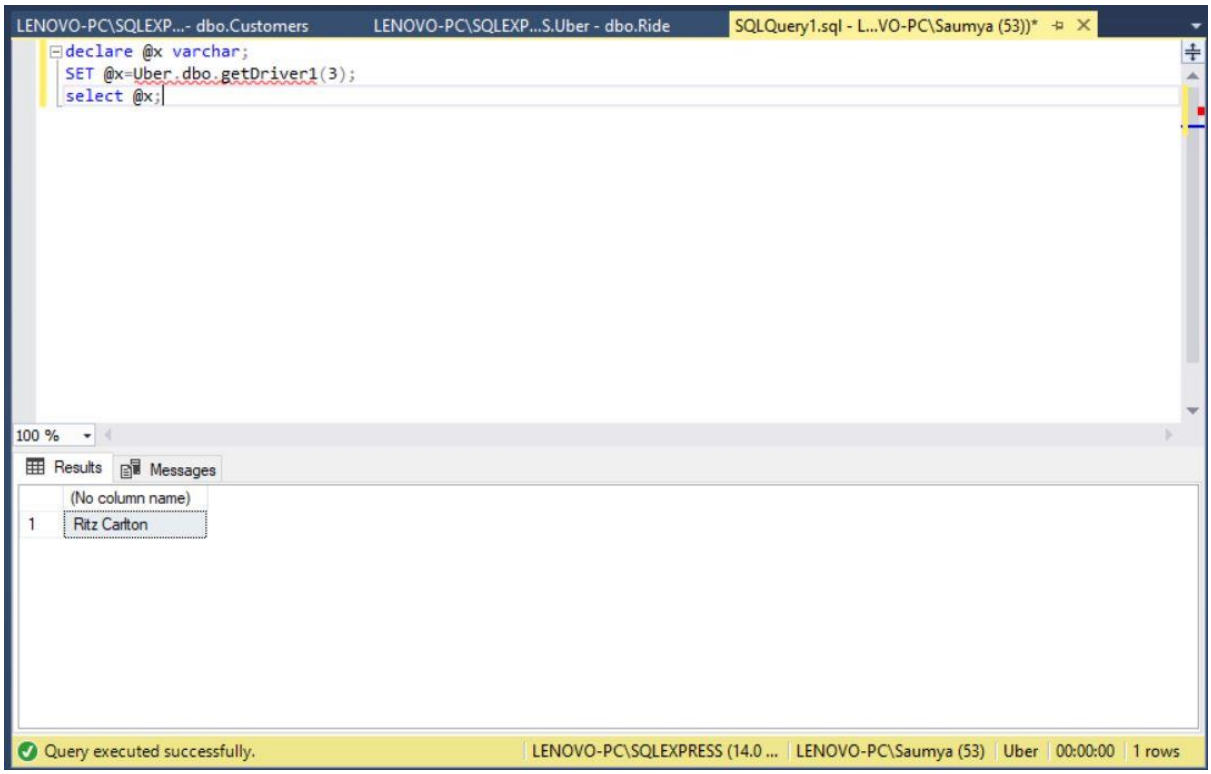
Function gives rating given by the customer to the driver with a given position.

```
CREATE FUNCTION Ratings (@id INT)
RETURNS INT
BEGIN
  DECLARE @RATING INT;
    SELECT @RATING = driver_rating FROM CustomerRatings WHERE
customer_rating_id = @id;
  RETURN @RATING;
END;

declare @x int;
SET @x=Uber.dbo.Ratings(8);
select @x;
```



Function gives the Driver Name of a Customer with given customer_id

```
CREATE FUNCTION getDriver1 (@id VARCHAR)
RETURNS VARCHAR
BEGIN
  DECLARE @full_name VARCHAR;
   SELECT @full_name = Driver.fname+' '+Driver.lname FROM Customers JOIN Ride
          on Customers.customer_id = Ride.customer_id JOIN Driver on Ride.driver_id =
Driver.driver_id
          WHERE Customers.customer_id = @id;
  RETURN @full_name;
END;

declare @x varchar;
SET @x=Uber.dbo.getDriver1(8);
select @x;
```



Testing Stored Procedures

Below are a few stored procedures:

This procedure gets customer details

```
CREATE PROC CustomerDetails
AS
 SELECT *  FROM Uber.dbo.Customers order by fname;
```

Results | Messages

	customer_id	fname	lname	phone_num	card_num	home_address	active	customer_rating_id
1	1	Barbara	Litt	4129990000	1234567891011121	34 ABC Street	1	2
2	6	Donald	Trump	4129990005	1112112345678910	27 RED Street	1	5
3	4	Hana	Ko	4129990003	1211234567891011	26 KOL Street	1	3
4	2	James	Bond	4129990001	1123456789101112	45 QAC Street	1	1
5	7	Kendall	Jenner	4129990006	1011121123456789	86 OPW Street	1	8
6	3	Kim	Kardashian	4129990002	2112345678910111	98 WEN Street	1	4
7	5	Patrick	Yang	4129990004	1121123456789101	04 DER Street	1	6
8	8	Victoria	Beckham	4129990007	9101112112345678	89 GUI Street	1	7

Query executed successfully. | LENOVO-PC\SQLEXPRESS (14.0 ... | LENOVO-PC\Saumya (53) | Uber | 00:00:00 | 8 rows

```
EXEC CustomerDetails;
GO
```

This procedure gets driver license

```
CREATE PROC DriverLicense
AS
select * from Uber.dbo.LicenseInfo;
```

Results | Messages

	license_id	issue_date	expiry
1	A1234	2014-01-01	2024-01-01
2	B903	2014-10-02	2024-10-02
3	C456	2012-11-01	2022-11-01
4	Q782	2015-01-07	2025-01-07
5	Y768	2013-09-10	2023-09-10
6	P092	2014-08-17	2024-08-17
7	A760	2013-02-28	2023-02-28
8	E321	2009-05-23	2019-05-23

```
EXEC DriverLicense;
GO
```

This procedure gives names of drivers starting from ‘S’

```
CREATE PROC DriversNamesBeginningFromS
AS
select fname from Uber.dbo.Driver where fname LIKE 'S%';
```

Results | Messages

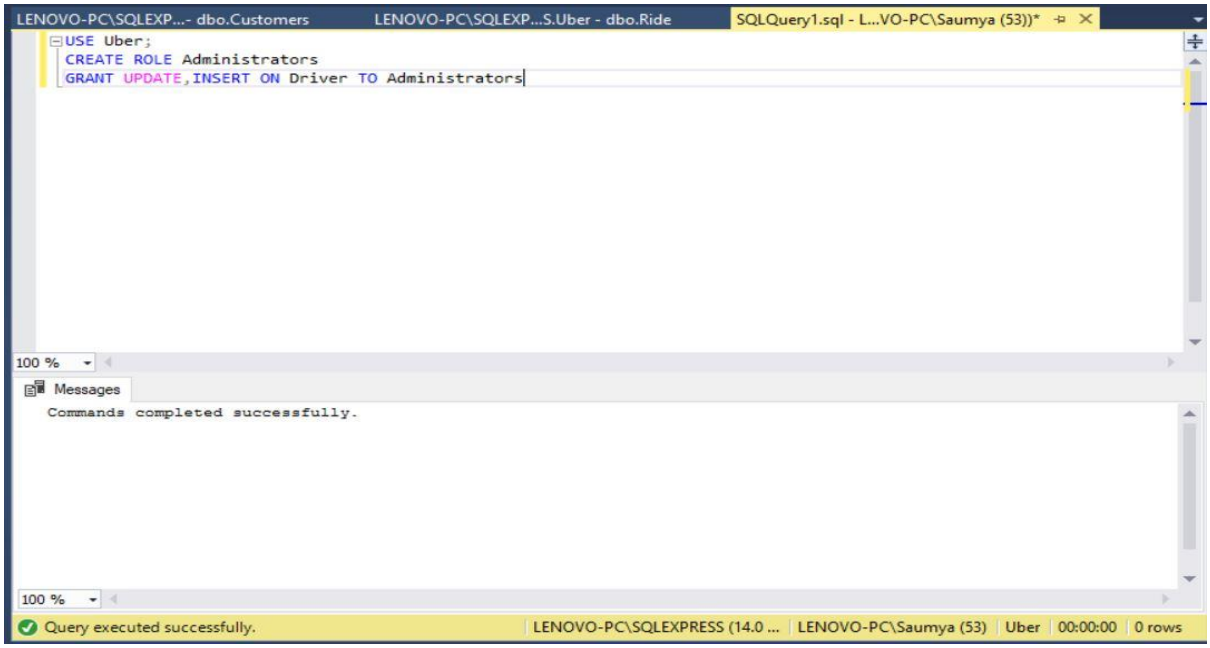
	fname
1	Steven

```
EXEC DriversNamesBeginningFromS;
GO
```

Testing Scripts

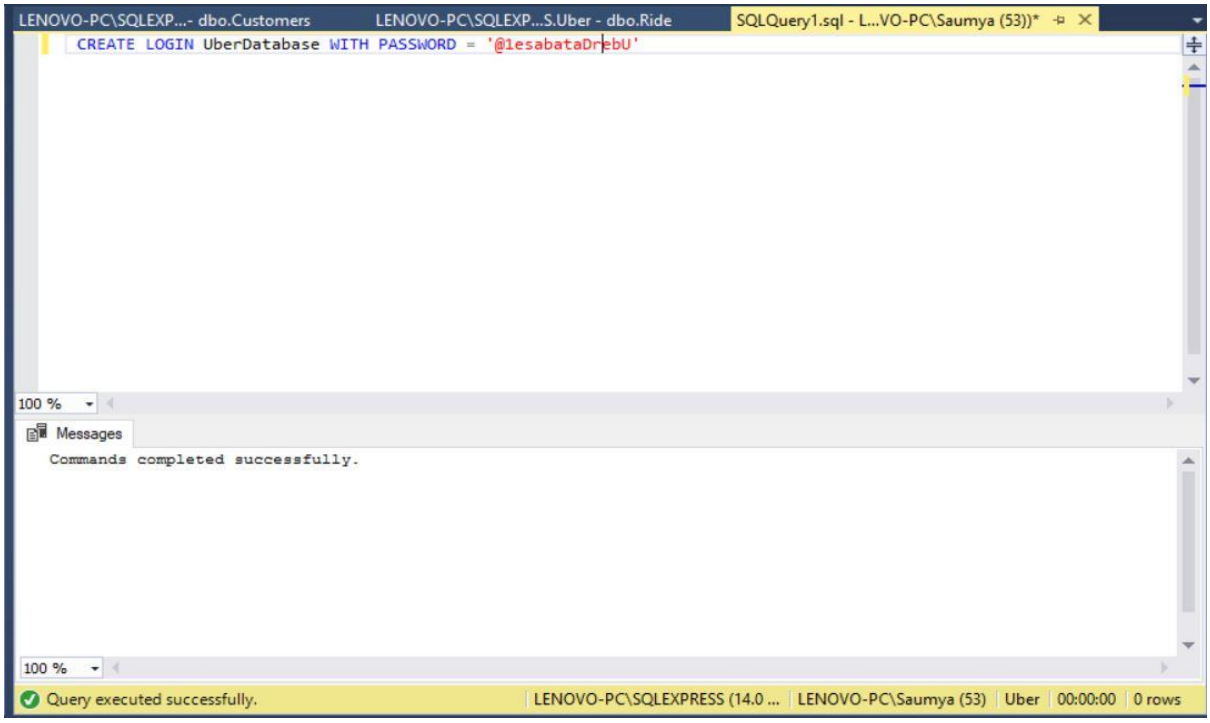
This query creates Admin role and gives update and insert access to Driver

USE Uber;
CREATE ROLE Administrators
GRANT UPDATE,INSERT ON Driver TO Administrators



This query creates Login Uber Database with Password

CREATE LOGIN UberDatabase WITH PASSWORD = '@1esabataDrebU'



Conclusion

The above design is limited and does not consider the driver working shifts and schedules. The database model presented in this report is focused only on the most important functionalities. The basic design consists of a Customer, Driver and the Administrator which we have successfully implemented.
We have implemented a few simple queries, views, functions and stored procedures. The above model can be extended to create more complex queries and procedures.