Database Design
Final Project Report

Course Number - CS6360.001

Team Number - 13

Project Title - Uber-3

Team Members

Soumya Mukhija (sxm200134)

Praticha Suhas Damale (psd190006)

Adithya Sundararajan Iyer (asi200000)

Table of Contents

Introduction

How Uber Works

Uber is a mobility-as-a-service provider, which means that they deliver the service of mobility to their customer base. Uber is responsible for helping people travel from a source to their choice of destination, providing a multitude of options for their comfort and convenience.

The system consists of a car, a driver, and a customer. The driver is assumed to always own the car that they drive. A customer makes a trip request, choosing their pickup location, drop location, and the type of car that they wish to travel in (*Premium*, which includes Black and Black SUV, or *Economy*, which includes UberX, UberXL, Comfort, and Uber Pet). A randomly selected driver near the pickup location can choose to either accept or decline that request. If the request is declined, it is forwarded to another randomly chosen driver in that area, and the process repeats itself.

Once the request is accepted by a driver, they can view the route to the pickup location of the customer. Once they reach the pickup point, the customer has 5 minutes to arrive, after which the ride is auto-canceled. After picking up the customer, the driver can view the route to the drop location on their app interface.

When the customer is dropped off, they see an option to rate the driver on their application and add a tip. Similarly, the driver gets an option to rate the customer as well. Both these ratings are used to calculate the average total rating of the respective customer and the driver.

Once the trip is complete, the customer can choose to pay with their card, cash, Paypal, or Uber money (which is virtual cash that is pre-added by the customer using their card). One can choose to save multiple cards in their account for easy payment. All completed trips must be paid for.

Uber also utilizes the concept of Uber Points. Each time a customer makes a trip, they earn some points based on the car type chosen and the fare paid (The number of points is twice the fare for *Economy* cars, and thrice the fare for *Premiums*). The aggregate of these points is used to determine the type of a customer, which can be among Blue, Gold, Diamond, and Platinum. The better the type of a customer, the more perks they receive while traveling with Uber.

Data Requirements and Interaction of Entities

Using the above data, we have extracted the main components of the system and used that information to build an initial Entity-Relationship (ER) diagram.

1. **Person**: A Person can be a Driver, Customer, or both. All Persons on Uber provide their email IDs, passwords, full names, and uniquely identifying contact numbers. They can choose to provide an online display picture link for easy recognition by others, their date of birth, and sex.

    *Note that even though overlaps between a Driver and a Customer are allowed, they both view different interfaces on the application.

    a. **Customer:** This is a subset of the Person entity. Each Person on the Customer interface has a few attributes that distinguish them from the Driver interface, including a customer rating, a unique referral link, the customer type, wallet balance (the amount of Uber money added and not yet used), and their Uber points.

    Customers are allowed to make and cancel trip requests. They can also have a list of saved cards or saved addresses for faster access to the Uber services. Every time a customer makes a trip, the database system automatically adds the trip and payment information, all of which can be viewed by the customer.

    b. **Driver**: Similar to above, each Driver has a few attributes distinguishing them from the Customer subset. These attributes include their SSN, address, information about their driver's license, and Driver's Rating.

    Each driver is assumed to own at least 1 car. Drivers are employees in Uber, so they have also linked salary accounts in the application where their pay is dispensed.

2. **Cars:** A car has several properties including a uniquely identifying Vehicle Identification Number (VIN), model, brand, color, a unique license plate number, insurance information, and capacity. It would also have a type, which a user has the flexibility to choose while requesting a ride. Premium cars are more expensive and worth more Uber points than Comfort.

3. **Trip Requests:** A customer makes a trip request, choosing from options such as car type, pickup location, and drop-off location. Based on these choices, they are shown an estimated rate for the ride. After booking, the customer can cancel the ride up to 5 minutes before the driver arrives. Upon cancelation, customers have to provide a reason from a list of options.

On the other hand, a driver can either accept or decline a trip request made by the customer. If they decline it, the request is forwarded to another randomly chosen driver in that area meeting the customer's preferences of car type. A trip can be either complete or canceled, but not both.

a. **Canceled Trips:**
    i. This is a disjoint subset of Trip Requests.
    ii. Trips canceled by the customers only consist of a cancellation reason, a penalty, and a timestamp in the database.
b. **Completed Trips:**
    i. A trip that is completed implies that a customer was provided service by the driver.
    ii. They were picked up and dropped off at their choice of location in their choice of car type.
    iii. Because of this, the Completed Trips entity contains a larger amount of information, including tips provided, timestamps of different milestones during the trip, ratings provided by both parties, the ride charge, and a total fare (including taxes) and discount applied.
    iv. Completed trips also have payment details and car details added to them.

4. **Payment History**: Payment is added every time a trip is completed, and then each payment entry is stored in the database. The attributes of this entity include an identifying transaction ID, the total amount paid including taxes and tips, payment mode (Cash/Card/Paypal/Uber Money) a Paypal ID (in the case of Paypal payment). The user also has an option to make payment from a saved card.

5. **Saved Cards**: These are cards that the user has saved in the database for easy access while making payments. These consist of a card nickname given by the user, type of card (Debit/Credit), card network (Visa/Mastercard etc.), full name on the card, card number, and expiration date. Cards are uniquely identified by the card numbers.

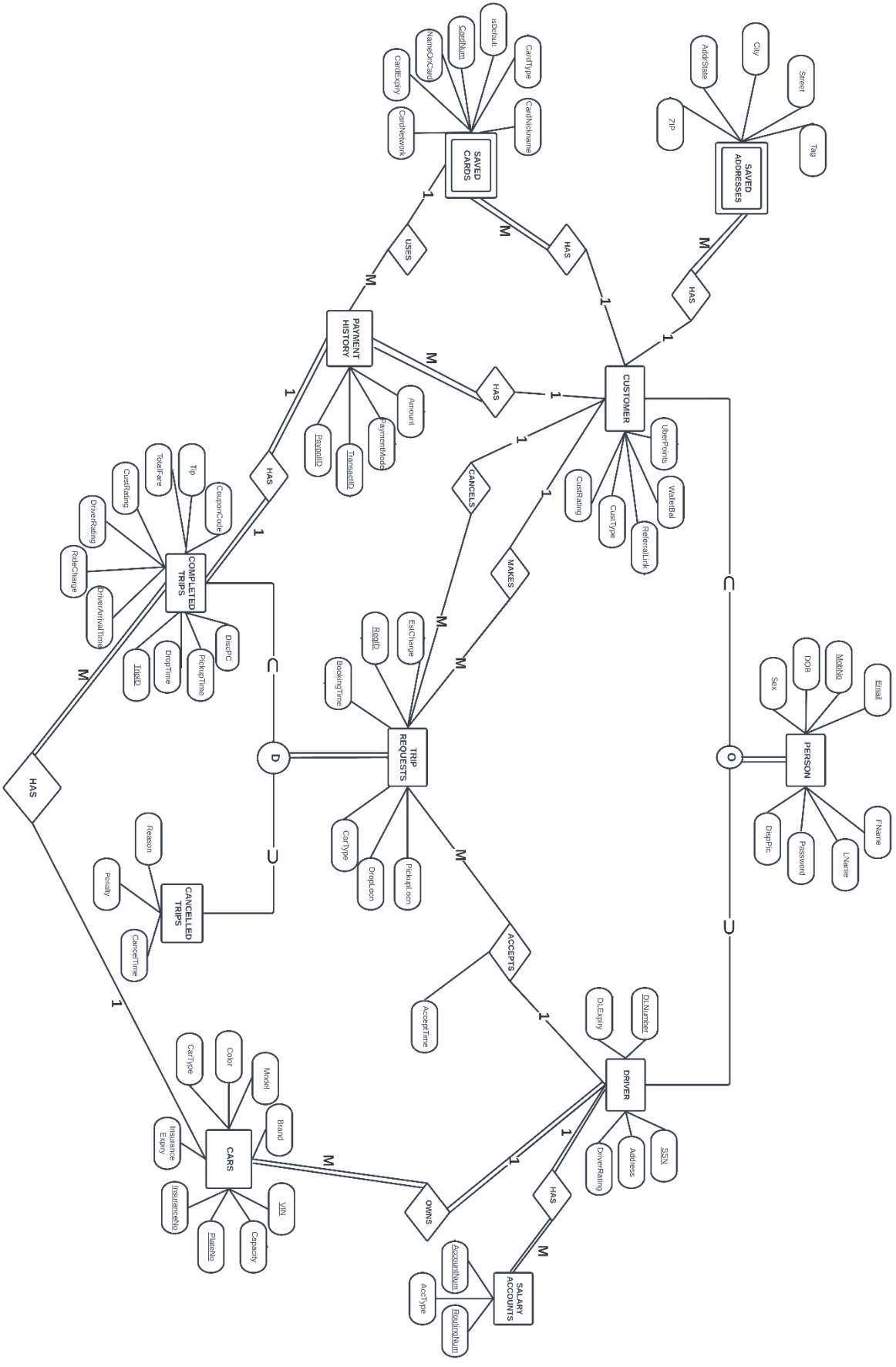
   Saved cards are used by the customer while making a payment, so they are linked with both the Customer and the Payment History entities. A saved card would not exist without a customer, so it is a weak entity of Customer in the system.

6. **Saved Addresses:** These are a list of addresses that a customer might go frequently to. They appear on the top of the application as soon as a person enters the Customer interface. Each of these addresses is assigned a unique tag by the user (such as *'Home'*), and they have a street, city, state, and zip code.

   A saved address would not exist without a customer, so it is a weak entity of Customer in the system.

7. **Salary Accounts**: Drivers are employees of Uber, so they each have a salary account linked with their profile that they receive their payments in. This entity includes a unique account ID, an account number, a routing number and the account type (savings/checking). A driver can have multiple salary accounts.

(Enhanced) Entity-Relationship Diagram



Relational Schema and Normalization

This section consists of 4 parts:

1. Mapping EER Diagram into Relational Schema
2. Informal Design Guidelines for Relational Databases
3. Functional Dependencies and Successive Normalization
4. 3NF Normalized Relational Schema

Mapping EER Diagram into Relational Schema

*Note - The primary keys have been underlined, and the foreign keys have been *italicized*.

1. **PERSON** (MobNo, FName, LName, Email, Password, DOB, Sex, DispPic)

2. **DRIVER** (SSN, Address, DLNumber, DLExpiry, DriverRating, *MobNo*)
   Foreign-Key(MobNo) REFERENCES PERSON(MobNo)

3. **CARS** (VIN, Brand, Model, Color, PlateNo, InsuranceNo, InsuranceExpiry, Capacity, CarType, *SSN*)
   Foreign-Key(SSN) REFERENCES DRIVER(SSN)

4. **SALARY_ACCOUNTS** (AccountNum, RoutingNum, AccType, *SSN*)
   Foreign-Key(SSN) REFERENCES DRIVER(SSN)

5. **CUSTOMER** (*MobNo*, CustType, UberPoints, WalletBal, ReferralLink, CustRating)
   Foreign-Key(MobNo) REFERENCES PERSON(MobNo)

6. **SAVED_ADDRESSES** (*MobNo*, Tag, Street, City, AddrState, ZIP)
   Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)

7. **SAVED_CARDS** (CardNum, NameOnCard, CardExpiry, CardType, CardNetwork, CardNickname, isDefault, *MobNo*)
   Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)

8. **TRIP_REQUESTS** (ReqID, BookingTime, CarType, PickupLocn, DropLocn, EstCharge, *MobNo*)
   Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)

9. **ACCEPTS** (*RequestID*, *SSN*, AcceptTime)
   Foreign-Key(RequestID) REFERENCES TRIP_REQUESTS(ReqID)
   Foreign-Key(SSN) REFERENCES DRIVER(SSN)

10. **CANCELLED_TRIPS** (*RequestID*, CancelTime, Reason, Penalty)
    Foreign-Key(RequestID) REFERENCES TRIP_REQUESTS(ReqID)

11. **COMPLETE_TRIPS** (TripID, DriverArrivalTime, PickupTime, DropTime, CouponCode, DiscPC, RideCharge, TotalFare, Tip, CustRating, DriverRating, *SSN*, *RequestID*, *VIN*)
    Foreign-Key(SSN) REFERENCES DRIVER(SSN)
    Foreign-Key(RequestID) REFERENCES TRIP_REQUESTS(ReqID)
    Foreign-Key(VIN) REFERENCES CARS(VIN)

12. **PAYMENT_HISTORY** (TransactID, Amount, PaymentMode, PaypalID, *CardNum*, *TripID*, *MobNo*)
    Foreign-Key(CardNum) REFERENCES SAVED_CARDS(CardNum)
    Foreign-Key(TripID) REFERENCES COMPLETE_TRIPS(TripID)
    Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)

Informal Design Guidelines for Relational Databases

➔ Semantics of the attributes are easily explainable, and each tuple in a relationship would represent one entity/relationship instance.

➔ There is no wasteful or redundant storage of information.

➔ The insertion, updation, and deletion anomalies will be handled by normalizing the table to 3NF form.

➔ Relations have been designed to reduce the occurrence of NULL values.

Functional Dependencies and Successive Normalization

All attributes would contain atomic values. None of the entities or relationships have multivalued, composite or nested attributes. Hence the designed schema is in 1NF.

PERSON

PERSON {MobNo, FName, LName, Email, Password, DOB, Sex, DispPic}

FD1: MobNo → FName, LName, Email, Password, DOB, Sex, DispPic

There is no partial or transitive dependency. So we can say it is already in 3NF.

DRIVER

DRIVER {SSN, Address, DLNumber, DLExpiry, DriverRating, MobNo}

FD2: SSN → Address, DLNumber, DriverRating, MobNo

FD3: DLNum → DLExpiry

There is no partial dependency. So we can say it is in 2NF.

Thus we see that transitive dependency exists. This violates 3NF. To eliminate this transitive dependency and normalize to 3NF, we create the following new tables:

DL_INFO {DLNum, DLExpiry}

DRIVER {SSN, Address, DLNumber, DriverRating, MobNo}

CARS

CARS {VIN, Brand, Model, Color, PlateNo, InsuranceNo, InsuranceExpiry, Capacity, CarType, SSN}

FD4: VIN → Brand, Model, Color, PlateNo, InsuranceNo, Capacity, CarType, SSN

FD5: InsuranceNo → InsuranceExpiry

There is no partial dependency. So we can say it is in 2NF.

Thus we see that transitive dependency exists. This violates 3NF. To eliminate this transitive dependency and normalize to 3NF, we create the following new tables:

INSURANCE {InsuranceNo, InsuranceExpiry}

CARS {VIN, Brand, Model, Color, PlateNo, InsuranceNo, Capacity, CarType, SSN}

SALARY_ACCOUNTS

SALARY_ACCOUNTS {AccountNum, RoutingNum, AccType, SSN}

FD6: AccountNum, RoutingNum → AccType, SSN

There is no partial or transitive dependency. So we can say it is already in 3NF.

CUSTOMER

CUSTOMER {MobNo, CustType, UberPoints, WalletBal, ReferralLink, CustRating}

FD7: MobNo → CustType, UberPoints, WalletBal, ReferralLink, CustRating

There is no partial or transitive dependency. So we can say it is already in 3NF.

SAVED ADDRESSES

SAVED_ADDRESSES {MobNo, Tag, Street, City, AddrState, ZIP}

FD8: MobNo, Tag → Street, City, AddrState, ZIP

There is no partial or transitive dependency. So we can say it is already in 3NF.

SAVED_CARDS

SAVED_CARDS {CardNum, NameOnCard, CardExpiry, CardType, CardNetwork, CardNickname, isDefault, MobNo}

FD9: CardNum → NameOnCard, CardExpiry, CardType, CardNetwork, CardNickname, isDefault, MobNo

There is no partial or transitive dependency. So we can say it is already in 3NF.

TRIP_REQUESTS

SAVED_CARDS {ReqID, BookingTime, CarType, PickupLocn, DropLocn, EstCharge, MobNo}

FD10: ReqID → BookingTime, CarType, PickupLocn, DropLocn, EstCharge, MobNo

There is no partial or transitive dependency. So we can say it is already in 3NF.

ACCEPTS

ACCEPTS {RequestID, SSN, AcceptTime}

FD11: RequestID, SSN → AcceptTime

There is no partial or transitive dependency. So we can say it is already in 3NF.

CANCELLED_TRIPS

CANCELLED_TRIPS {RequestID, CancelTime, Reason, Penalty}

FD12: RequestID → CancelTime, Reason, Penalty

There is no partial or transitive dependency. So we can say it is already in 3NF.

COMPLETE_TRIPS

COMPLETE_TRIPS {TripID, DriverArrivalTime, PickupTime, DropTime, CouponCode, DiscPC, RideCharge, TotalFare, Tip, CustRating, DriverRating, SSN, RequestID, VIN}

FD13: TripID → DriverArrivalTime, PickupTime, DropTime, CouponCode, RideCharge, TotalFare, Tip, CustRating, DriverRating, SSN, RequestID, VIN

FD14: CouponCode → DiscPC

There is no partial dependency. So we can say it is in 2NF.

Thus we see that transitive dependency exists. This violates 3NF. To eliminate this transitive dependency and normalize to 3NF, we create the following new tables:

COUPONS {CouponCode, DiscPC}

COMPLETE_TRIPS {TripID, DriverArrivalTime, PickupTime, DropTime, CouponCode, RideCharge, TotalFare, Tip, CustRating, DriverRating, SSN, RequestID, VIN}

PAYMENT_HISTORY

PAYMENT_HISTORY {TransactID, Amount, PaymentMode, PaypalID, CardNum, TripID, MobNo}

FD15: TransactID → Amount, PaymentMode, PaypalID, CardNum, TripID, MobNo

There is no partial or transitive dependency. So we can say it is already in 3NF.

3NF Normalized Relational Schema

*Note - The primary keys have been underlined, and the foreign keys have been *italicized*.

1. **PERSON** (MobNo, FName, LName, Email, Password, DOB, Sex, DispPic)
2. **DL_INFO** {DLNum, DLExpiry}
3. **DRIVER** (SSN, Address, DriverRating, *DLNumber*, *MobNo*)
   Foreign-Key(MobNo) REFERENCES PERSON(MobNo)
   Foreign-Key(DLNumber) REFERENCES DL_INFO(DLNum)
4. **INSURANCE** {InsuranceNo, InsuranceExpiry}
5. **CARS** (VIN, Brand, Model, Color, PlateNo, Capacity, CarType, *InsuranceNo*, *SSN*)
   Foreign-Key(SSN) REFERENCES DRIVER(SSN)
   Foreign-Key(InsuranceNo) REFERENCES INSURANCE(InsuranceNo)
6. **SALARY_ACCOUNTS** (AccountNum, RoutingNum, AccType, *SSN*)
   Foreign-Key(SSN) REFERENCES DRIVER(SSN)
7. **CUSTOMER** (*MobNo*, CustType, UberPoints, WalletBal, ReferralLink, CustRating)
   Foreign-Key(MobNo) REFERENCES PERSON(MobNo)
8. **SAVED_ADDRESSES** (*MobNo*, Tag, Street, City, AddrState, ZIP)
   Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)
9. **SAVED_CARDS** (CardNum, NameOnCard, CardExpiry, CardType, CardNetwork, CardNickname, isDefault, *MobNo*)
   Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)
10. **TRIP_REQUESTS** (ReqID, BookingTime, CarType, PickupLocn, DropLocn, EstCharge, *MobNo*)
    Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)
11. **ACCEPTS** (*RequestID*, *SSN*, AcceptTime)
    Foreign-Key(RequestID) REFERENCES TRIP_REQUESTS(ReqID)
    Foreign-Key(SSN) REFERENCES DRIVER(SSN)

12. **CANCELLED_TRIPS** (*RequestID*, CancelTime, Reason, Penalty)
    Foreign-Key(RequestID) REFERENCES TRIP_REQUESTS(ReqID)

13. **COUPONS** {CouponCode, DiscPC}

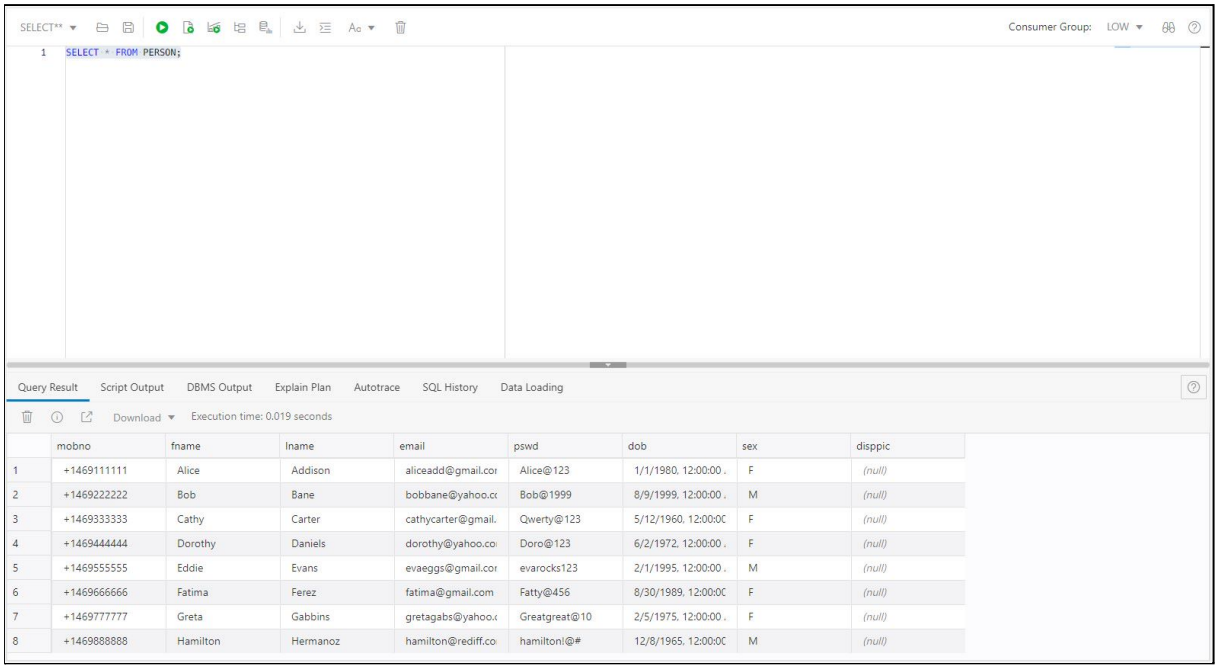
14. **COMPLETE_TRIPS** (TripID, DriverArrivalTime, PickupTime, DropTime, RideCharge, TotalFare, Tip, CustRating, DriverRating, *SSN*, *RequestID*, *VIN*, *CouponCode*)
    Foreign-Key(SSN) REFERENCES DRIVER(SSN)
    Foreign-Key(RequestID) REFERENCES TRIP_REQUESTS(ReqID)
    Foreign-Key(VIN) REFERENCES CARS(VIN)
    Foreign-Key(CouponCode) REFERENCES COUPONS(CouponCode)

15. **PAYMENT_HISTORY** (TransactID, Amount, PaymentMode, PaypalID, *CardNum*, *TripID*, *MobNo*)
    Foreign-Key(CardNum) REFERENCES SAVED_CARDS(CardNum)
    Foreign-Key(TripID) REFERENCES COMPLETE_TRIPS(TripID)
    Foreign-Key(MobNo) REFERENCES CUSTOMER(MobNo)

Table Creation Queries (With Screenshots)

1. PERSON

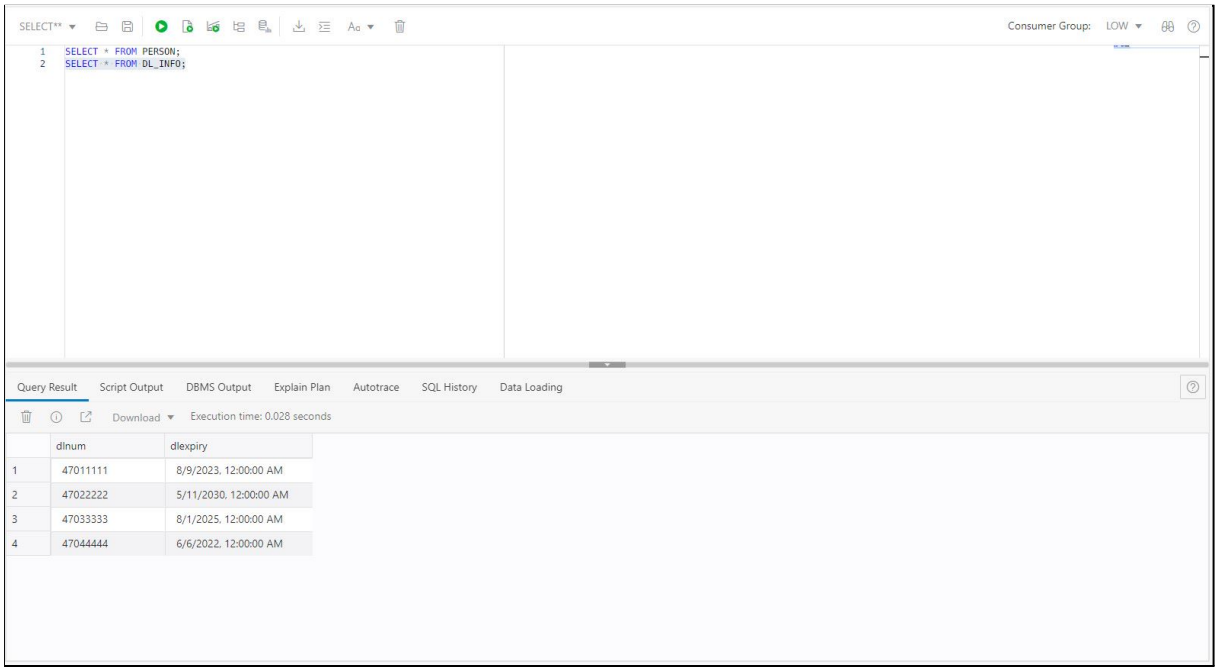
```
CREATE TABLE Person
(
  MobNo      varchar(13)          ,
  FName      varchar(50)          NOT NULL,
  LName      varchar(50)          NOT NULL,
  Email      varchar(50)          NOT NULL UNIQUE,
  Pswd           char(30)             NOT NULL,
  DOB            DATE                 ,
  Sex            char(1)          ,
  DispPic        varchar(200)     ,

  PRIMARY KEY(MobNo)
);
```



	mobno	fname	lname	email	pswd	dob	sex	disppic
1	+1469111111	Alice	Addison	aliceadd@gmail.cor	Alice@123	1/1/1980, 12:00:00 .	F	(null)
2	+1469222222	Bob	Bane	bobbane@yahoo.cc	Bob@1999	8/9/1999, 12:00:00 .	M	(null)
3	+1469333333	Cathy	Carter	cathycarter@gmail.	Qwerty@123	5/12/1960, 12:00:0(	F	(null)
4	+1469444444	Dorothy	Daniels	dorothy@yahoo.co	Doro@123	6/2/1972, 12:00:00 .	F	(null)
5	+1469555555	Eddie	Evans	evaeggs@gmail.cor	evarocks123	2/1/1995, 12:00:00 .	M	(null)
6	+1469666666	Fatima	Ferez	fatima@gmail.com	Fatty@456	8/30/1989, 12:00:0(	F	(null)
7	+1469777777	Greta	Gabbins	gretagabs@yahoo.(	Greatgreat@10	2/5/1975, 12:00:00 .	F	(null)
8	+1469888888	Hamilton	Hermanoz	hamilton@rediff.co	hamilton!@#	12/8/1965, 12:00:0(	M	(null)

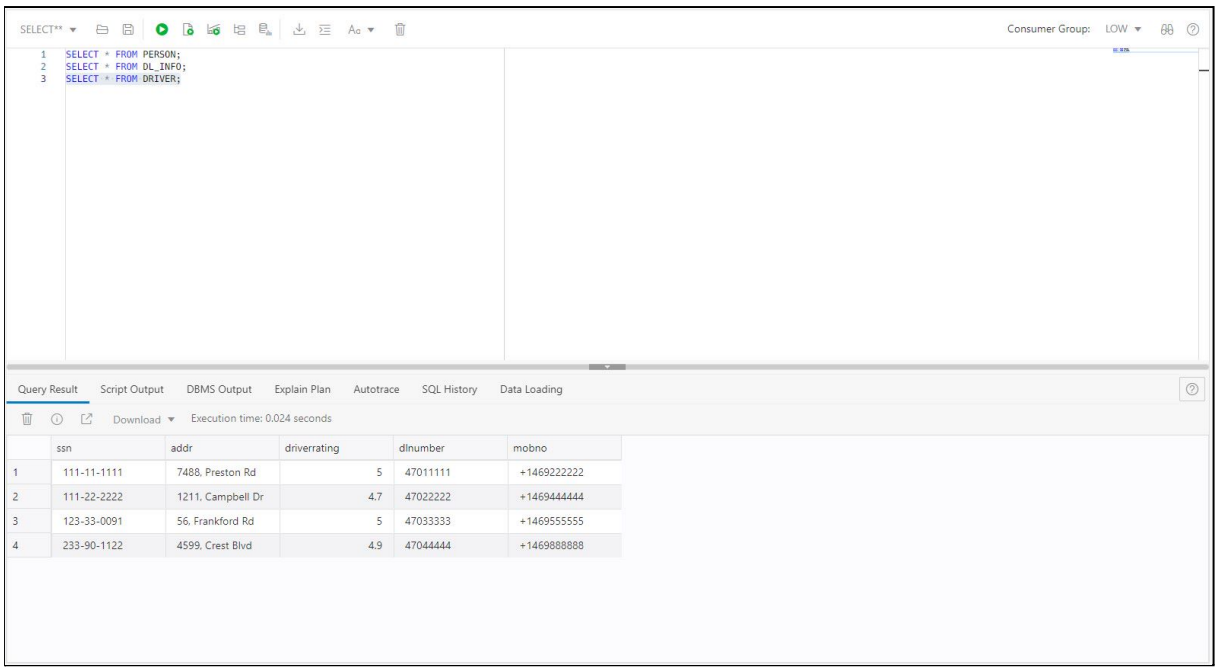
2. DL_INFO

```
CREATE TABLE DL_Info
(
    DLNum    char(8) ,
    DLExpiry     DATE         NOT NULL,

    PRIMARY KEY(DLNum)
);
```



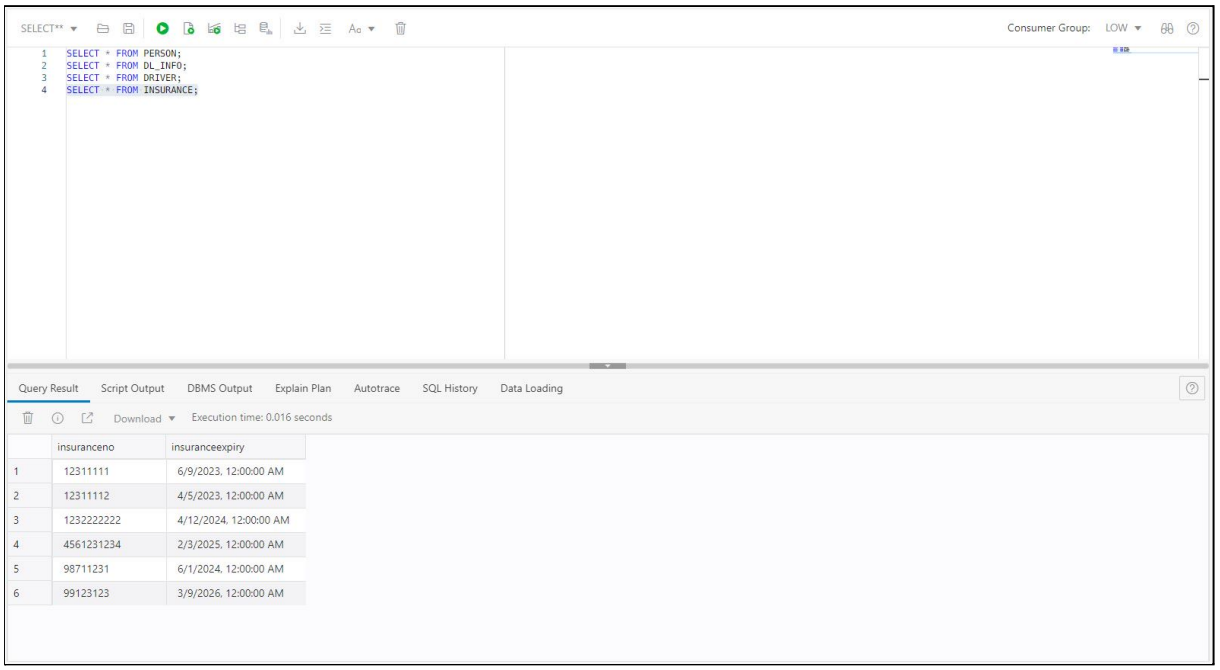
3. DRIVER

```
CREATE TABLE Driver
(
  SSN            char(11)         ,
  Addr           varchar(100)         NOT NULL,
  DriverRating       decimal(2,1)     ,
  DLNumber       char(8)      NOT NULL UNIQUE,
  MobNo      varchar(13)      NOT NULL,


  PRIMARY KEY(SSN),
  FOREIGN KEY (MobNo) REFERENCES Person(MobNo) ON DELETE  CASCADE,
  FOREIGN KEY (DLNumber) REFERENCES DL_INFO(DLNum) ON DELETE CASCADE
);
```



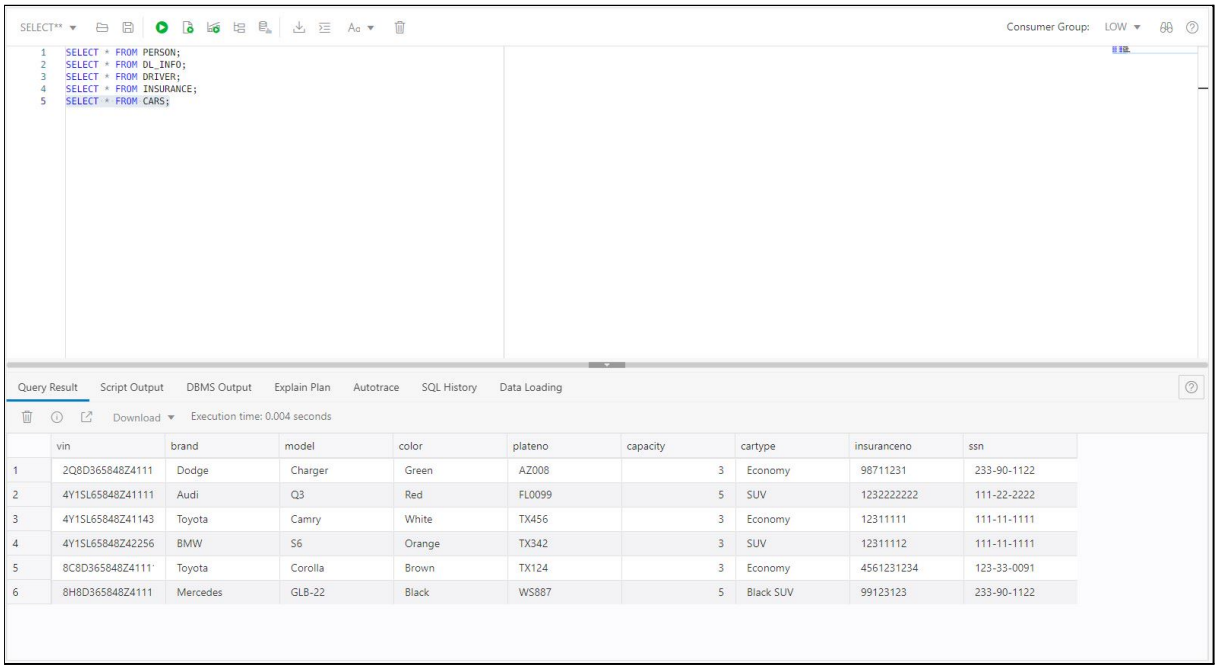
4. INSURANCE

```
CREATE TABLE INSURANCE
(
    InsuranceNo     varchar(15)     ,
    InsuranceExpiry DATE            NOT NULL,

    PRIMARY KEY(InsuranceNo)
);
```



5. CARS

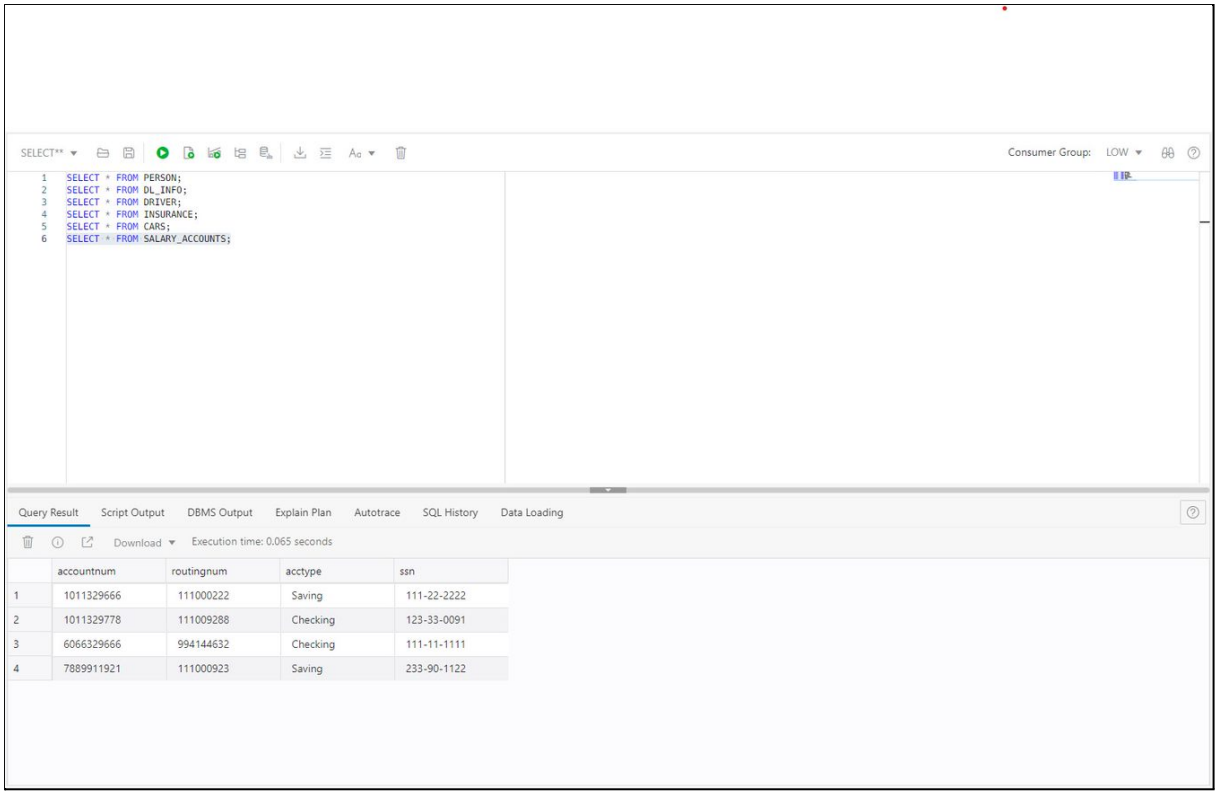
```
CREATE TABLE Cars
(
  VIN            char(17)         ,
  Brand                  varchar(50)      NOT NULL,
  Model      varchar(50)      NOT NULL,
  Color      varchar(20)      NOT NULL,
  PlateNo        varchar(20)      NOT NULL UNIQUE,
  Capacity               int      DEFAULT 3    NOT NULL ,
  CarType        varchar(20)      NOT NULL,
  InsuranceNo    varchar(15)      NOT NULL UNIQUE,
  SSN            char(11)         NOT NULL ,

  PRIMARY KEY(VIN),
  FOREIGN KEY (SSN) REFERENCES Driver(SSN) ON DELETE CASCADE,
  FOREIGN KEY (InsuranceNo) REFERENCES INSURANCE(InsuranceNo) ON DELETE CASCADE
);
```



	vin	brand	model	color	plateno	capacity	cartype	insuranceno	ssn
1	2Q8D365848Z4111	Dodge	Charger	Green	AZ008	3	Economy	98711231	233-90-1122
2	4Y1SL65848Z41111	Audi	Q3	Red	FL0099	5	SUV	1232222222	111-22-2222
3	4Y1SL65848Z41143	Toyota	Camry	White	TX456	3	Economy	12311111	111-11-1111
4	4Y1SL65848Z42256	BMW	S6	Orange	TX342	3	SUV	12311112	111-11-1111
5	8C8D365848Z4111	Toyota	Corolla	Brown	TX124	3	Economy	4561231234	123-33-0091
6	8H8D365848Z4111	Mercedes	GLB-22	Black	WS887	5	Black SUV	99123123	233-90-1122

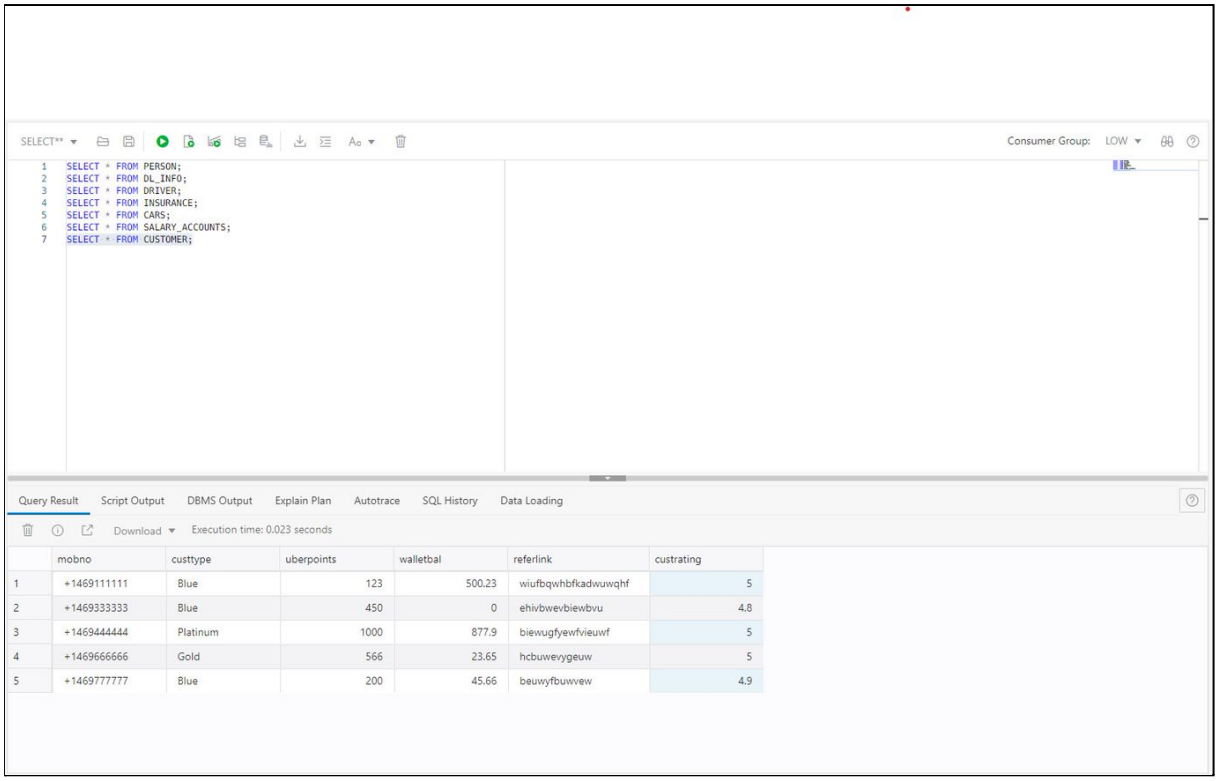
6. SALARY_ACCOUNTS

```
CREATE TABLE Salary_Accounts
(
    AccountNum       char(10)         NOT NULL,
    RoutingNum       char(9)      NOT NULL,
    AccType      varchar(10)      NOT NULL,
    SSN          char(11)         NOT NULL,

    PRIMARY KEY(AccountNum,RoutingNum),
    FOREIGN KEY (SSN) REFERENCES Driver(SSN) ON DELETE CASCADE
);
```



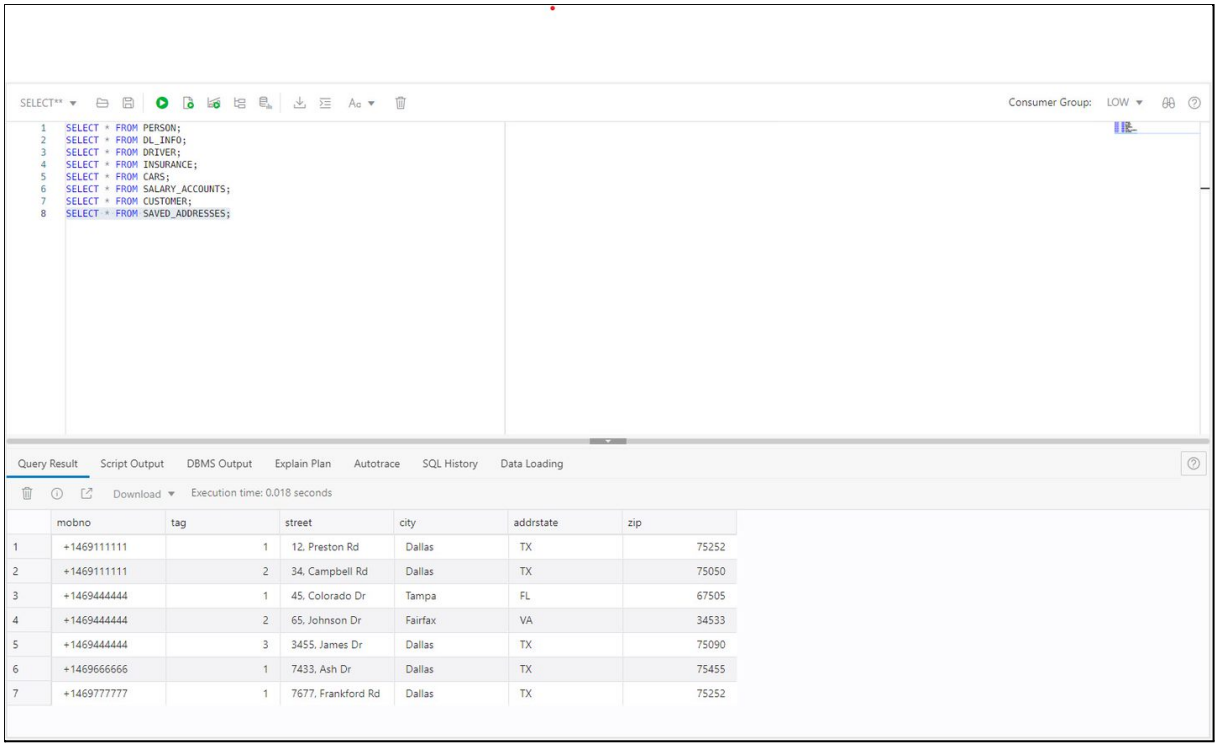
7. CUSTOMER

```
CREATE TABLE Customer
(
    MobNo    varchar(13)      ,
    CustType     varchar(15)      DEFAULT 'Blue' NOT NULL ,
    UberPoints   int          DEFAULT 0 NOT NULL,
    WalletBal    decimal(10,2)    DEFAULT 0 NOT NULL,
    ReferLink    varchar(70)      NOT NULL,
    CustRating   decimal(2,1)             ,

    PRIMARY KEY(MobNo),
    FOREIGN KEY (MobNo) REFERENCES Person(MobNo) ON DELETE CASCADE
);
```



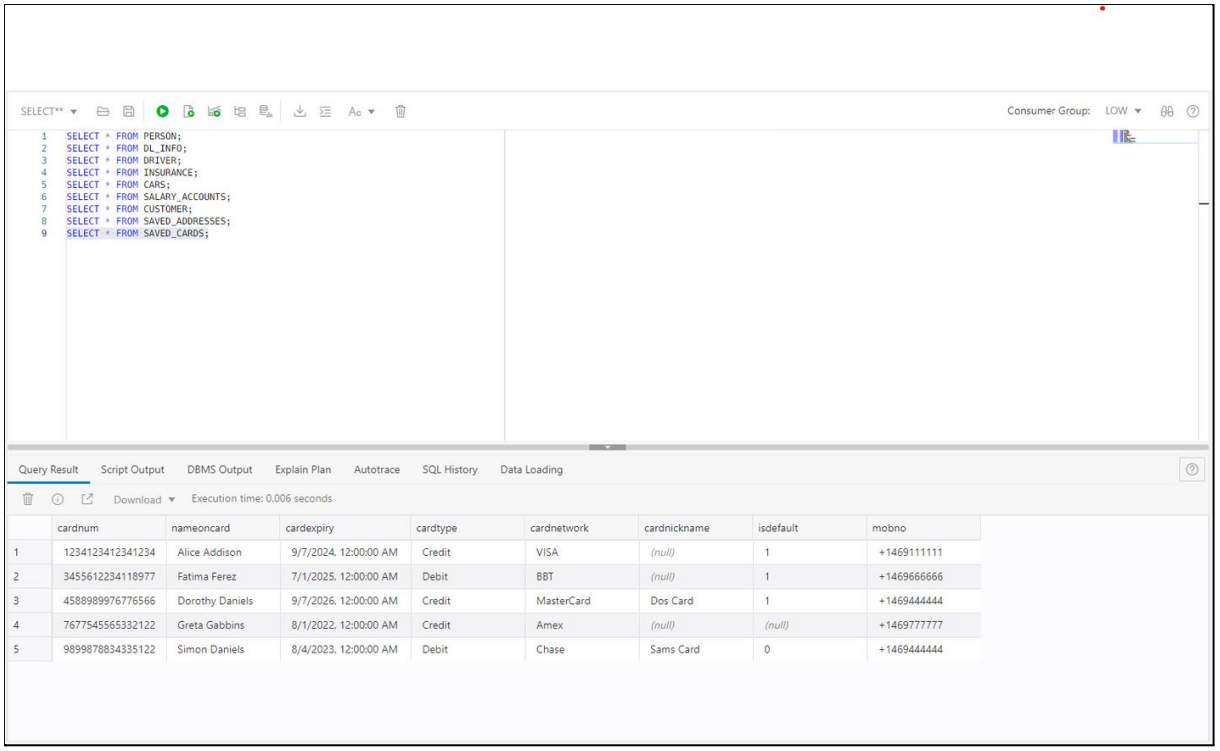
8. SAVED_ADDRESSES

```
CREATE TABLE Saved_Addresses
(
    MobNo    varchar(13)      NOT NULL,
    Tag      int          NOT NULL,
    Street       varchar(30)      NOT NULL,
    City         varchar(30)      NOT NULL,
    AddrState    varchar(30)      NOT NULL,
    ZIP      int          NOT NULL,

    PRIMARY KEY (MobNo,Tag),
    FOREIGN KEY (MobNo) REFERENCES Customer(MobNo) ON DELETE CASCADE
);
```



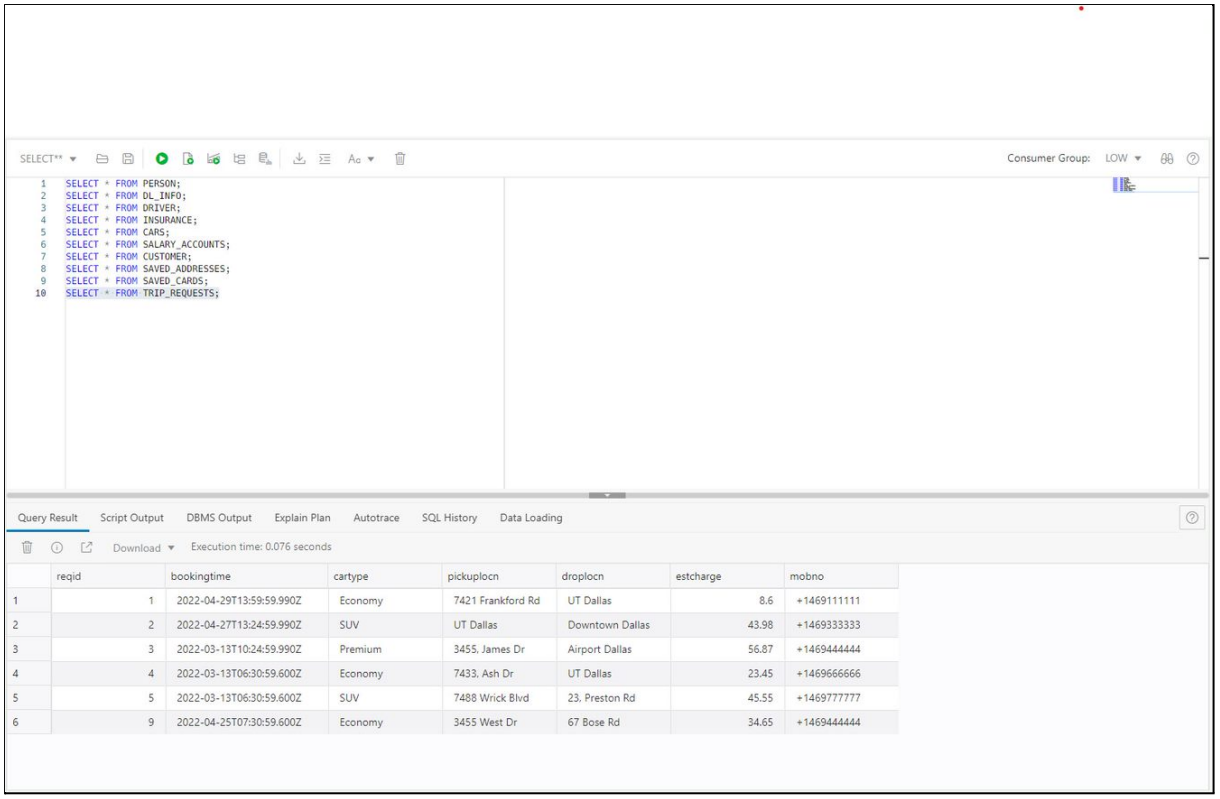
9. SAVED_CARDS

```
CREATE TABLE Saved_Cards
(
    CardNum             varchar(16)     ,
    NameOnCard      varchar(50)     NOT NULL,
    CardExpiry      DATE            NOT NULL,
    CardType            varchar(50)     NOT NULL,
    CardNetwork     varchar(30)     NOT NULL,
    CardNickname    varchar(30),
    isDefault       CHAR(1),
    MobNo       varchar(13)     NOT NULL,

    PRIMARY KEY(CardNum),
    FOREIGN KEY (MobNo) REFERENCES Customer(MobNo) ON DELETE CASCADE

);
```



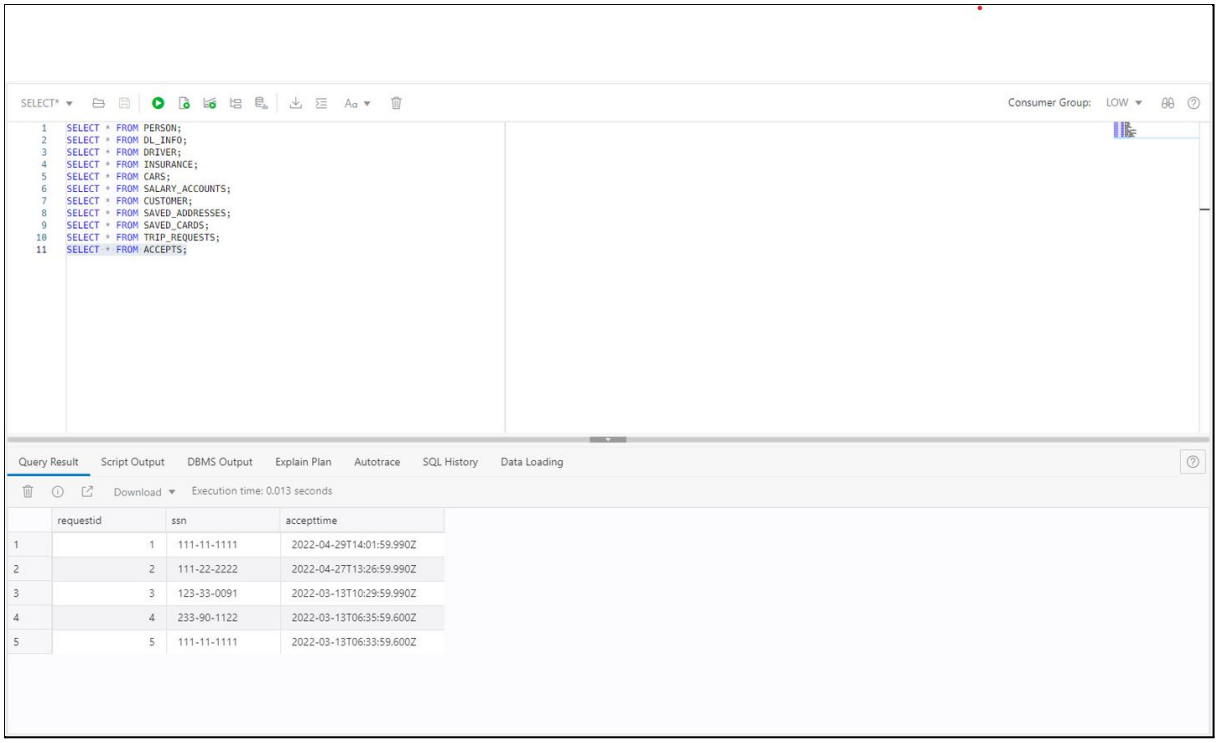
10. TRIP_REQUESTS

```
CREATE TABLE Trip_Requests
(
    ReqID               int GENERATED BY DEFAULT ON NULL AS IDENTITY  ,
    BookingTime     TIMESTAMP       NOT NULL,
    CarType             varchar(50)     NOT NULL,
    PickupLocn          varchar(50)     NOT NULL,
    DropLocn        varchar(50)     NOT NULL,
    EstCharge           decimal(5,2)        NOT NULL,
    MobNo       varchar(13)     NOT NULL,

    PRIMARY KEY(ReqID),
    FOREIGN KEY (MobNo) REFERENCES Customer(MobNo) ON DELETE CASCADE

);
```



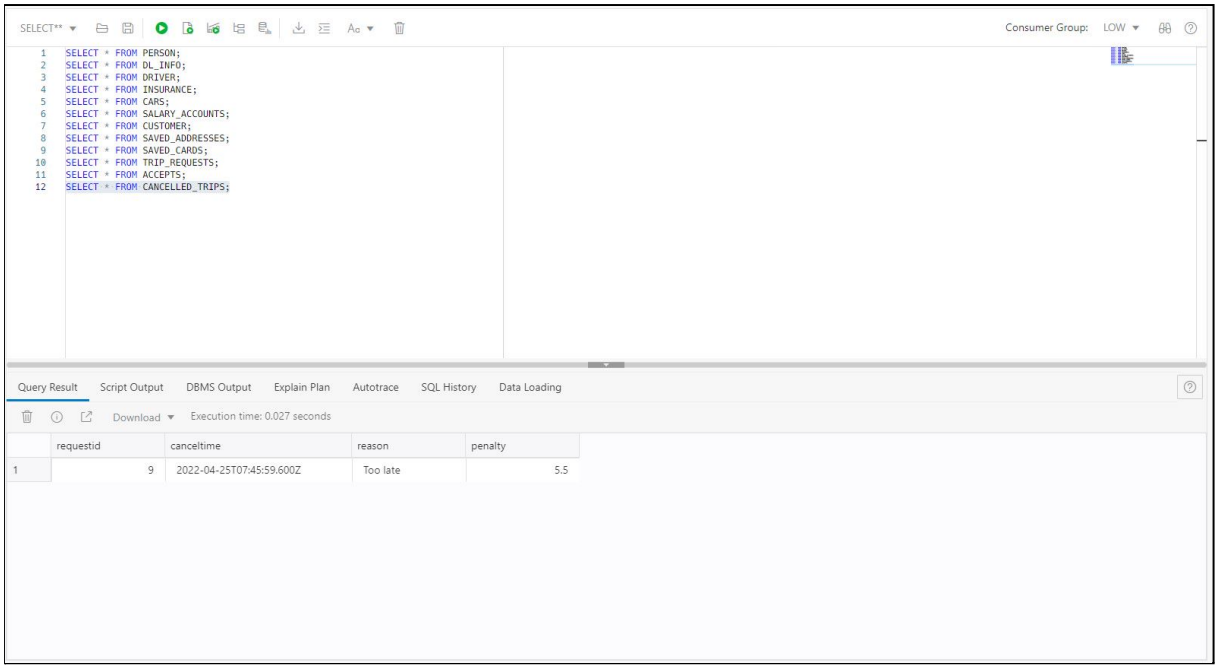
11. ACCEPTS

```
CREATE TABLE Accepts
(
    RequestID   int     ,
    SSN     char(11)    ,
    AcceptTime  TIMESTAMP   NOT NULL,

    PRIMARY KEY(RequestID,SSN),
    FOREIGN KEY (RequestID) REFERENCES Trip_Requests(ReqID) ON DELETE CASCADE,
    FOREIGN KEY (SSN) REFERENCES Driver(SSN)
);
```



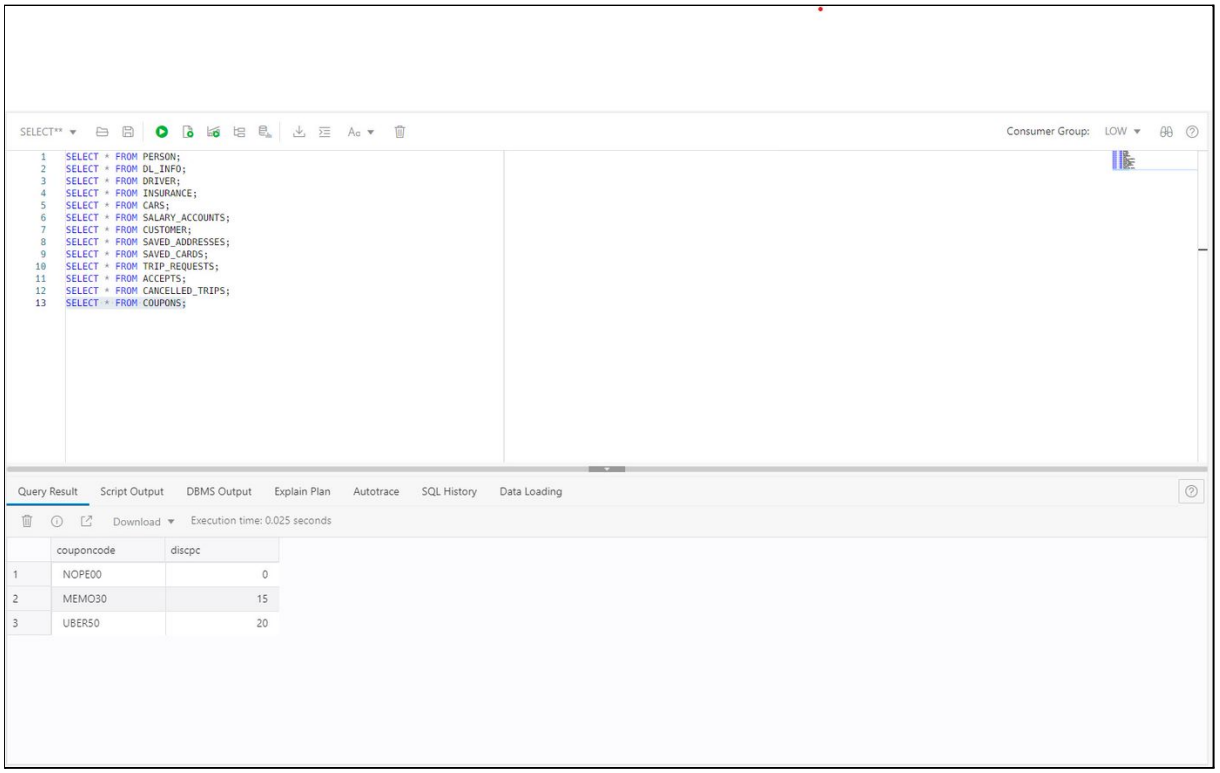
12. CANCELLED_TRIPS

```
CREATE TABLE Cancelled_Trips
(
    RequestID   int          NOT NULL,
    CancelTime  TIMESTAMP        NOT NULL,
    Reason              varchar(30)      NOT NULL,
    Penalty decimal(5,2)         NOT NULL,

    PRIMARY KEY(RequestID),
    FOREIGN KEY (RequestID) REFERENCES Trip_Requests(ReqID) ON DELETE CASCADE
);
```



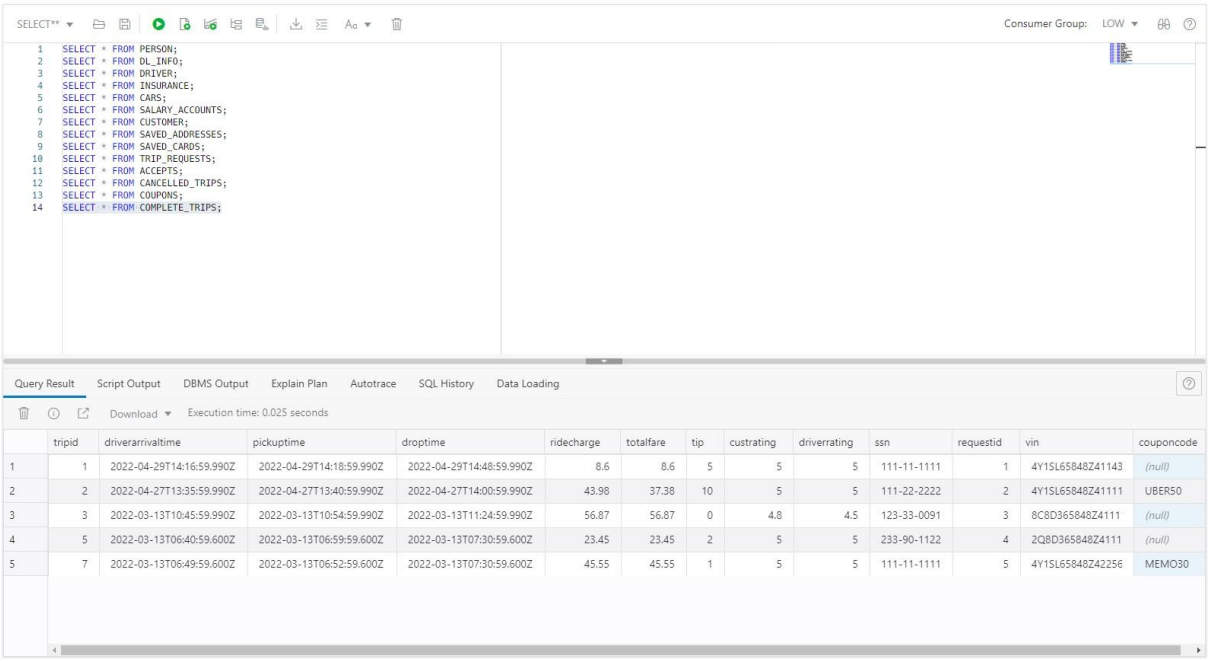
13. COUPONS

```
CREATE TABLE Coupons
(
    CouponCode  varchar(15),
    DiscPC  int,

    PRIMARY KEY(CouponCode)
);
```



14. COMPLETE_TRIPS

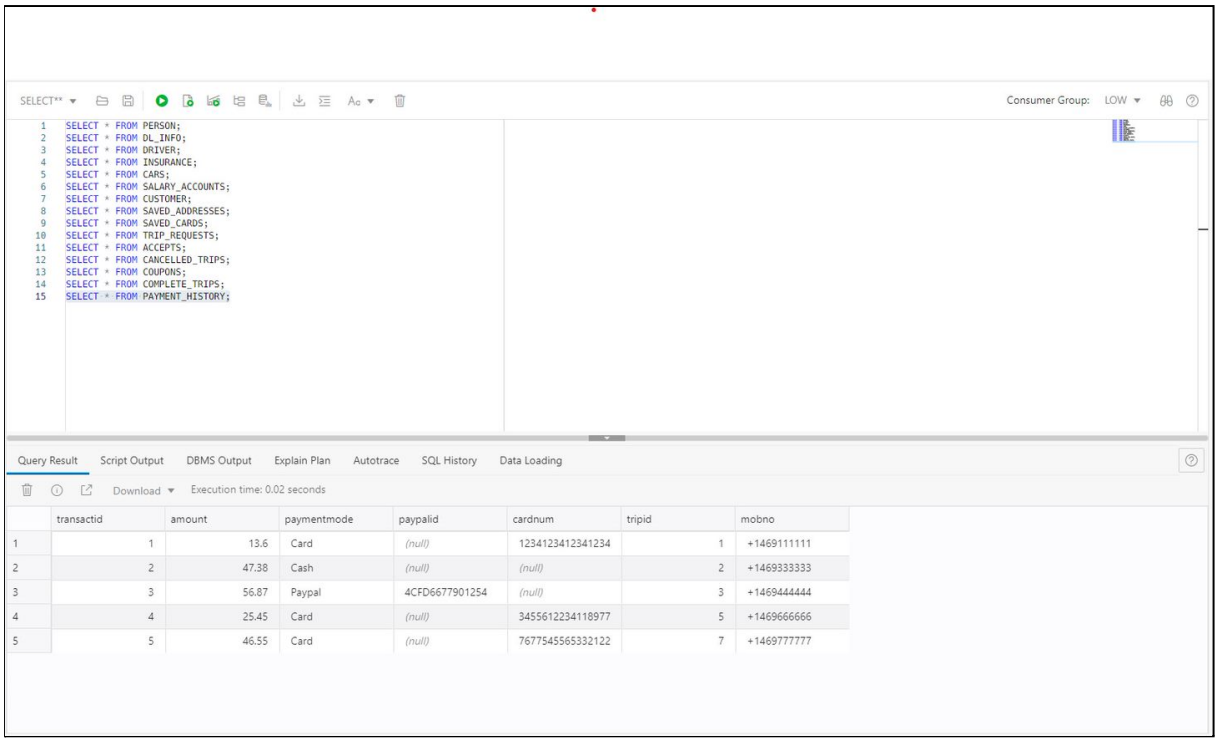
```
CREATE TABLE Complete_Trips
(
    TripID          int GENERATED BY DEFAULT ON NULL AS IDENTITY ,
    DriverArrivalTime   TIMESTAMP       NOT NULL,
    PickupTime      TIMESTAMP       NOT NULL,
    DropTime        TIMESTAMP       NOT NULL,
    RideCharge      decimal(5,2),
    TotalFare       decimal(5,2)        NOT NULL,
    Tip         decimal(5,2)        NOT NULL,
    CustRating      decimal(2,1)        NOT NULL,
    DriverRating        decimal(2,1)        NOT NULL,
    SSN         char(11)        NOT NULL,
    RequestID       int         NOT NULL,
    VIN         char(17)        NOT NULL,
    CouponCode      varchar(15),

    PRIMARY KEY(TripID),
    FOREIGN KEY (SSN) REFERENCES Driver(SSN),
    FOREIGN KEY (RequestID) REFERENCES Trip_Requests(ReqID)  ,
    FOREIGN KEY (VIN) REFERENCES Cars(VIN) ,
    FOREIGN KEY (CouponCode) REFERENCES COUPONS(CouponCode)
);
```



	tripid	driverarrivaltime	pickuptime	droptime	ridecharge	totalfare	tip	custrating	driverrating	ssn	requestid	vin	couponcode
1	1	2022-04-29T14:16:59.990Z	2022-04-29T14:18:59.990Z	2022-04-29T14:48:59.990Z	8.6	8.6	5	5	5	111-11-1111	1	4Y1SL65848Z41143	(null)
2	2	2022-04-27T13:35:59.990Z	2022-04-27T13:40:59.990Z	2022-04-27T14:00:59.990Z	43.98	37.38	10	5	5	111-22-2222	2	4Y1SL65848Z41111	UBER50
3	3	2022-03-13T10:45:59.990Z	2022-03-13T10:54:59.990Z	2022-03-13T11:24:59.990Z	56.87	56.87	0	4.8	4.5	123-33-0091	3	8C8D365848Z4111	(null)
4	5	2022-03-13T06:40:59.600Z	2022-03-13T06:59:59.600Z	2022-03-13T07:30:59.600Z	23.45	23.45	2	5	5	233-90-1122	4	2Q8D365848Z4111	(null)
5	7	2022-03-13T06:49:59.600Z	2022-03-13T06:52:59.600Z	2022-03-13T07:30:59.600Z	45.55	45.55	1	5	5	111-11-1111	5	4Y1SL65848Z42256	MEMO30

15. PAYMENT_HISTORY

```
CREATE TABLE Payment_History
(
  TransactID        int GENERATED BY DEFAULT ON NULL AS IDENTITY,
  Amount         decimal(6,2)         NOT NULL,
  PaymentMode    varchar(20)      NOT NULL,
  PaypalID       varchar(20),
  CardNum        varchar(16)      ,
  TripID         int          NOT NULL,
  MobNo      varchar(13)      NOT NULL,

  PRIMARY KEY(TransactID),
  FOREIGN KEY (CardNum) REFERENCES Saved_Cards(CardNum) ON DELETE SET NULL,
  FOREIGN KEY (TripID) REFERENCES Complete_Trips(TripID) ON DELETE SET NULL,
  FOREIGN KEY (MobNo) REFERENCES Customer(MobNo) ON DELETE SET NULL
);
```

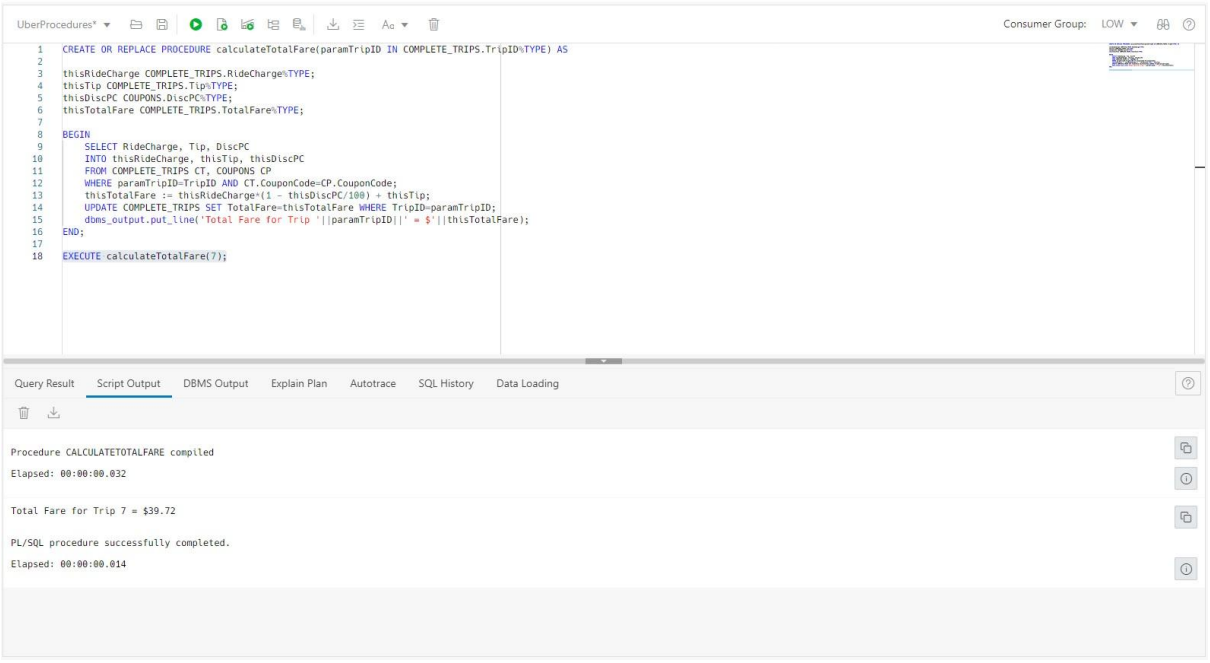


PL/SQL Procedures & Triggers

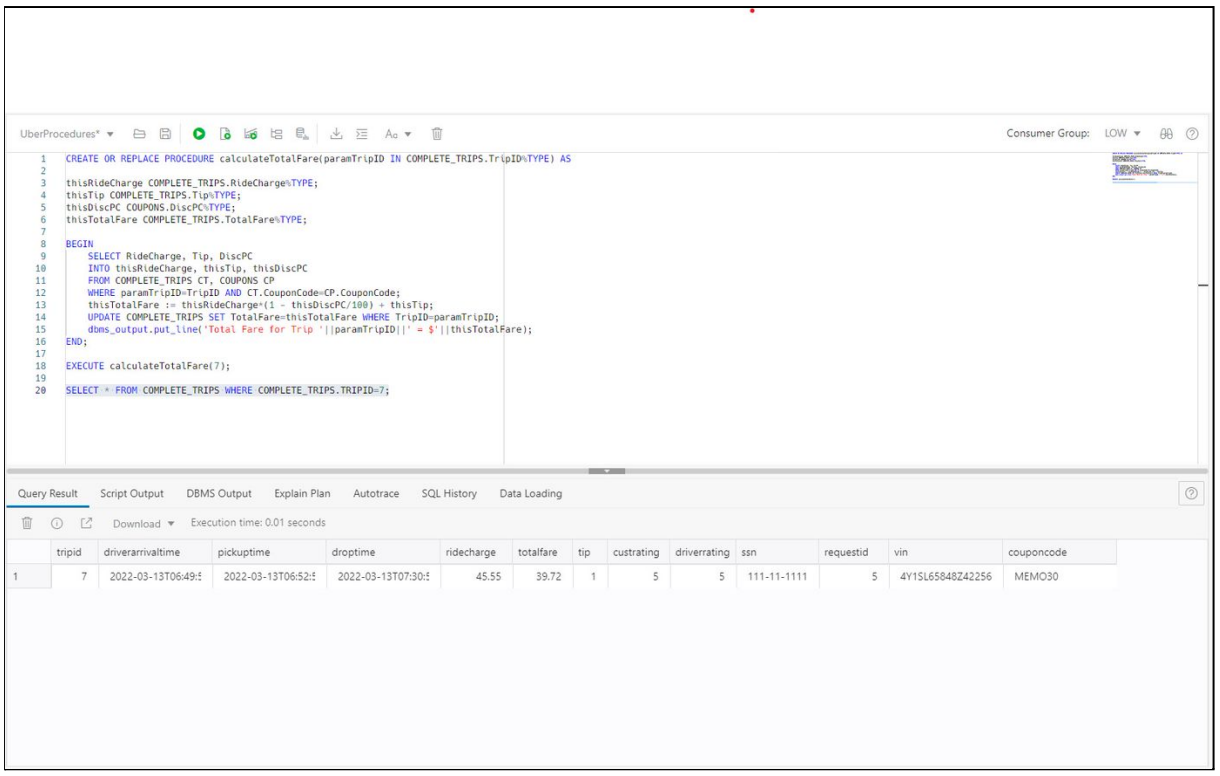
Stored Procedures

1. Procedure to compute the total fare of a completed trip.

```
CREATE OR REPLACE PROCEDURE calculateTotalFare(paramTripID IN
COMPLETE_TRIPS.TripID%TYPE) AS

thisRideCharge COMPLETE_TRIPS.RideCharge%TYPE;
thisTip COMPLETE_TRIPS.Tip%TYPE;
thisDiscPC COUPONS.DiscPC%TYPE;
thisTotalFare COMPLETE_TRIPS.TotalFare%TYPE;

BEGIN
    SELECT RideCharge, Tip, DiscPC
    INTO thisRideCharge, thisTip, thisDiscPC
    FROM COMPLETE_TRIPS CT, COUPONS CP
    WHERE paramTripID=TripID AND CT.CouponCode=CP.CouponCode;
    thisTotalFare := thisRideCharge*(1 - thisDiscPC/100) + thisTip;
    UPDATE COMPLETE_TRIPS SET TotalFare=thisTotalFare WHERE
TripID=paramTripID;
    dbms_output.put_line('Total Fare for Trip '||paramTripID||' =
$'||thisTotalFare);
END;
```



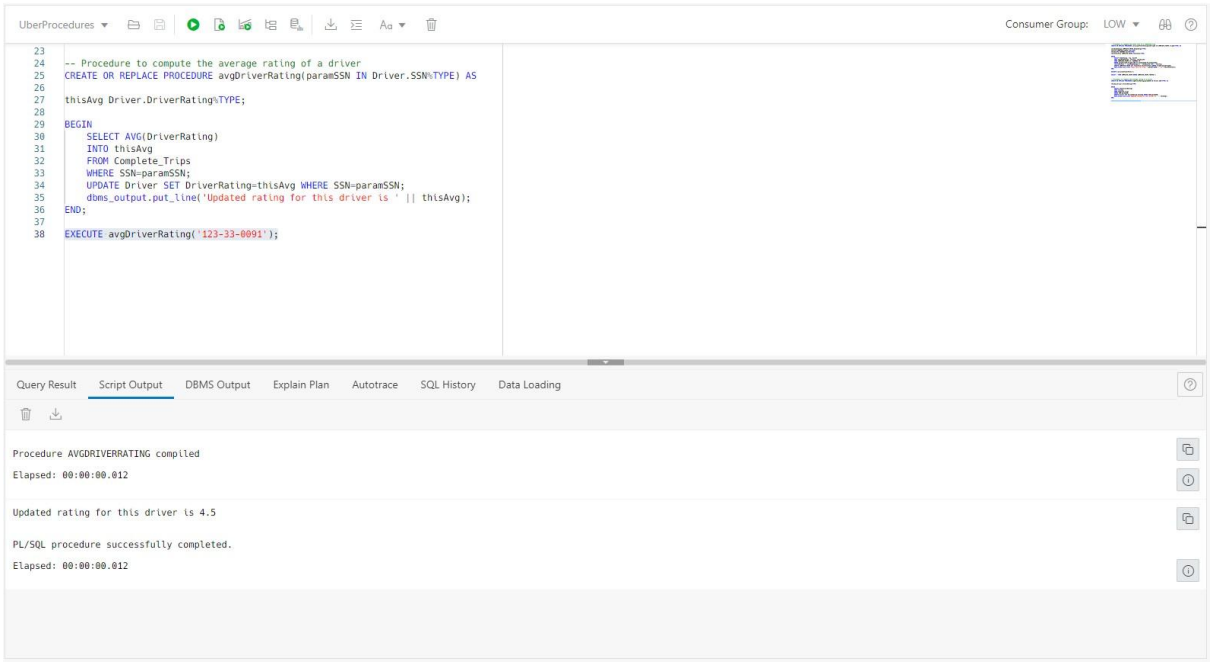
Compilation and Execution of procedure



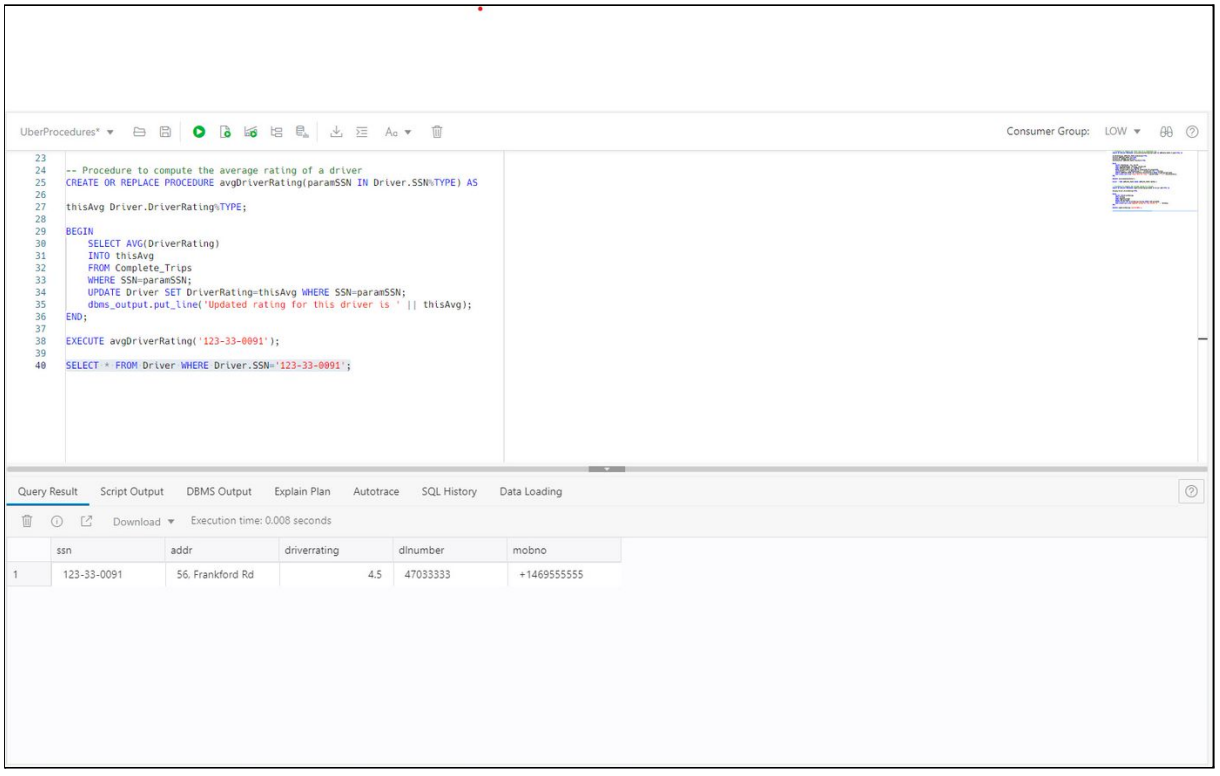
Updated value in table after procedure execution

2. Procedure to compute the average rating of a driver.

```
CREATE OR REPLACE PROCEDURE avgDriverRating(paramSSN IN Driver.SSN%TYPE) AS

thisAvg Driver.DriverRating%TYPE;

BEGIN
    SELECT AVG(DriverRating)
    INTO thisAvg
    FROM Complete_Trips
    WHERE SSN=paramSSN;
    UPDATE Driver SET DriverRating=thisAvg WHERE SSN=paramSSN;
    dbms_output.put_line('Updated rating for this driver is ' || thisAvg);
END;
```



Compilation and Execution of procedure

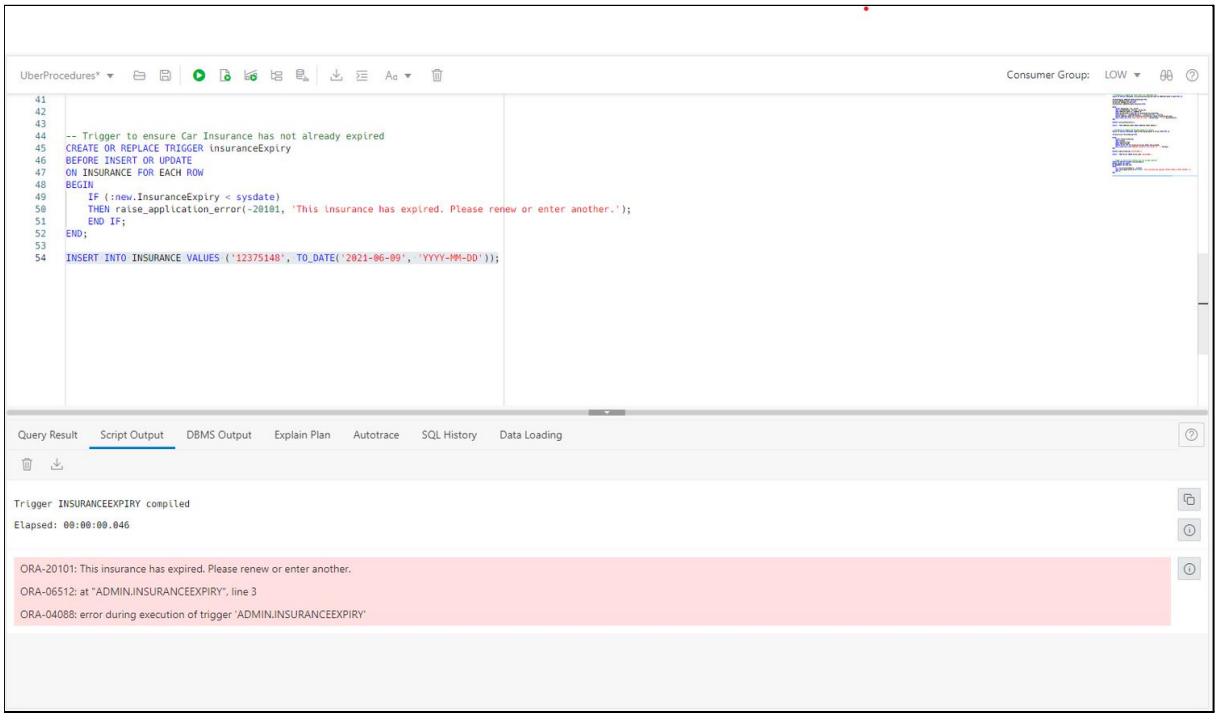


Updated value in table after procedure execution

Triggers

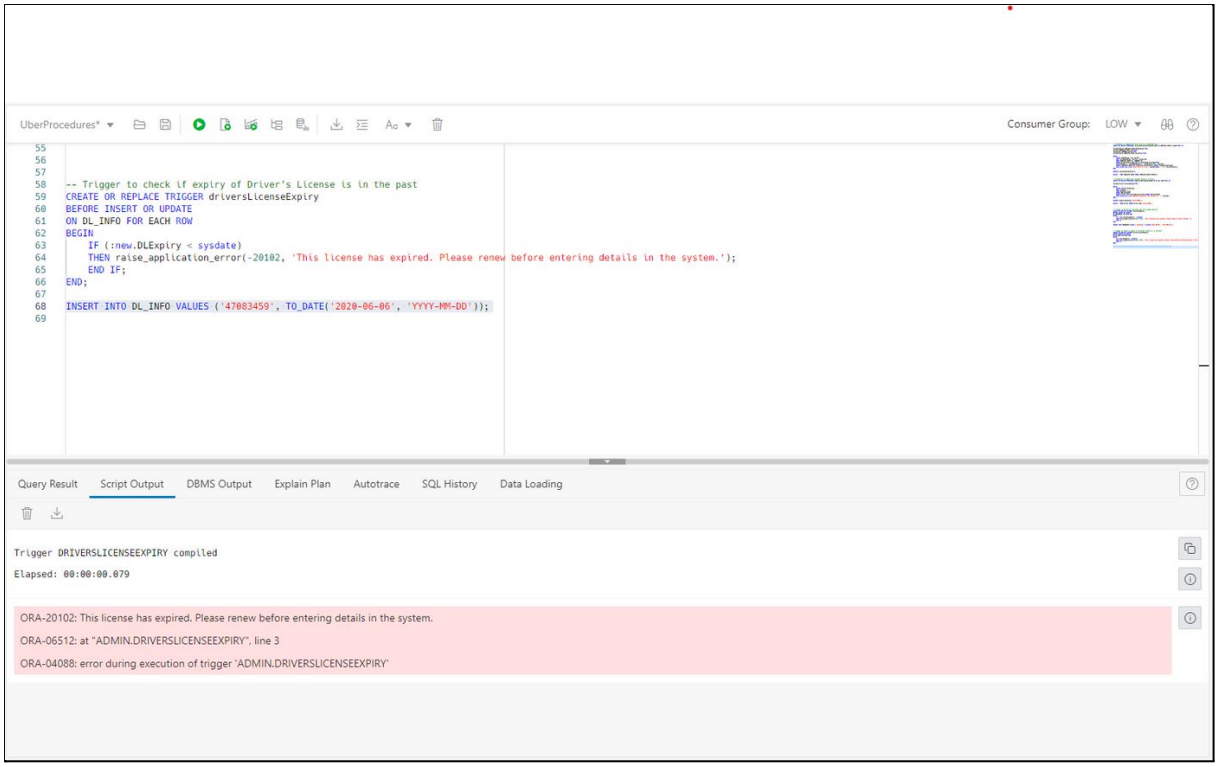
1. Trigger to ensure Car Insurance has not already expired

```
CREATE OR REPLACE TRIGGER insuranceExpiry
BEFORE INSERT OR UPDATE
ON INSURANCE FOR EACH ROW
BEGIN
    IF (:new.InsuranceExpiry < sysdate)
     THEN raise_application_error(-20101, 'This insurance has expired. Please
renew or enter another.');
    END IF;
END;
```



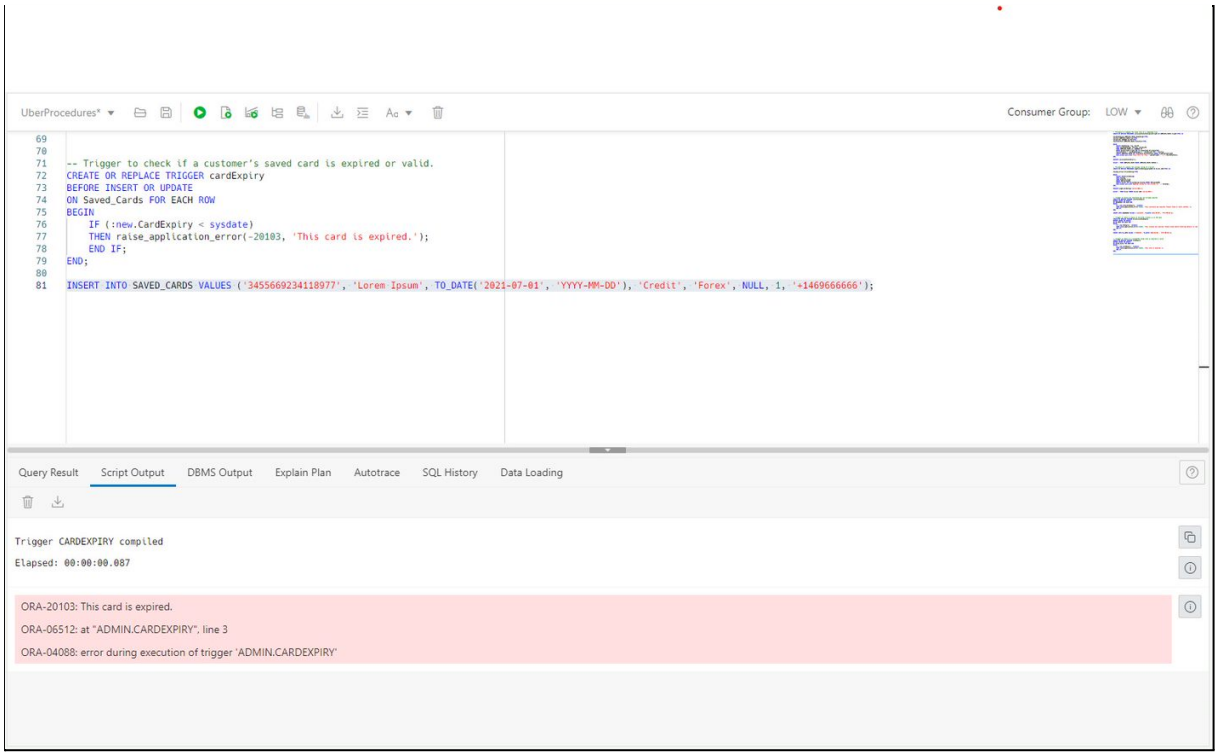
2. Trigger to check if expiry of Driver's License is in the past

```
CREATE OR REPLACE TRIGGER driversLicenseExpiry
BEFORE INSERT OR UPDATE
ON DL_INFO FOR EACH ROW
BEGIN
    IF (:new.DLExpiry < sysdate)
     THEN raise_application_error(-20102, 'This license has expired. Please
renew before entering details in the system.');
    END IF;
END;
```



3. Trigger to check if a customer's saved card is expired or valid.

```
CREATE OR REPLACE TRIGGER cardExpiry
BEFORE INSERT OR UPDATE
ON Saved_Cards FOR EACH ROW
BEGIN
    IF (:new.CardExpiry < sysdate)
    THEN raise_application_error(-20103, 'This card is expired.');
    END IF;
END;
```



THANK YOU!