DATABASE DESIGN FOR UBER

CS6360.003

UBER-1

Team:

Fatemeh Hajiheidari(fxh200004)
Lipi Vikram Thakker(lxt190004)
Monish Alur Gowdru (mxa190025)

TABLE OF CONTENT

Requirements:

Uber is a ride Transportation Network Company. It provides peer-to-peer ridesharing service to the customers. The company does not own most of the cars, they let customers join as drivers with a personal registered vehicle. But, in case if anyone wants to join and does not own a car, the company sponsors the employee. It acts as a broker, receiving commissions from each booking.

Uber mainly has two types of users:

1. Driver:

2. Customer

Major Services:

1. User can sign up by providing the basic information, creating login id and password.

2. User can register as a driver and provide rides.

3. Driver provides information about the car including the make, model, number of seats, and insurance.

4. Customer has the flexibility to choose the type of ride(for example UberX or UberXL). The estimated fare price is also visible to the customer based on the type of ride chosen. The customer gets allotted to a ride according to the availability of the driver in the current location.

5. Customer can apply promo code if available while making the booking.

6. Customer can book a cab from the current location to destination.

7. Driver can make a choice of whether to accept or deny the ride.

8. Once the cab is confirmed, customer and driver can communicate with each other via messaging.

9. After reaching the destination, driver ends the ride and estimated fare is deducted by payment option chosen by the customer at the beginning of the ride.

10. Customer and driver can give reviews and rate their ride experience.

We need to create database design based on the following requirements of UBER database:

DRIVER: To qualify as an Uber driver, a user must be at least 21 years of age, have a valid driver's license, SSN, clean driving record and registered vehicle under insurance.

CUSTOMER: User above the age of 18, can have an Uber account who needs a ride from current location to destination location.

A **USER** entity has been created which can be either CUSTOMER or a DRIVER, both are registered with Uber.

It stores the personal information about the user including name, date of birth, address, email, and phone number. Each user is identified uniquely with the User ID generated by Uber when the user registers. Note that drivers can also be a customer when not riding their vehicle.

The **CUSTOMER** is uniquely identified with customer ID, which is derived from the UserID.

The **DRIVER** is identified by a driver ID which is derived from user ID. Driver information including driver's license number, SSN, join date are also stored.

Driver's **VEHICLE** information must be stored to help customers locate the car when they are booking a ride and to help companies to understand the condition of the car and keep track of their employees. Details include, Vehicle Identification Number (VIN), registration number, make, condition, color, capacity, insurance number, last inspected, purchased date, mileage, manufacture year.

We need to know the driver's license details. Creating **DL_DETAILS** would keep track of the expiry date of DL, what state issued the license along with DL number.

Driver can make a choice of hitting the road to provide service at the time of their convenience. The database will store **SHIFT** details, that provides information about the date, login time, logout time, referring to each driver. One driver can have multiple shifts during a day.

Whenever a customer requests a ride, details will be stored in **TRIP** with a trip ID which will contain the information about fare, distance from pickup address to drop off address, booking date. Will assume that a customer is assigned a driver who is near to pick up location (might be the one looking to provide rides or one who's drop off location of a customer is near to pick up location).

Will consider the two possibilities trips:

COMPLETED TRIP: The database stores information about the pickup time, drop off time, duration of the ride, final deductible amount paid by customer's account after including the tip.

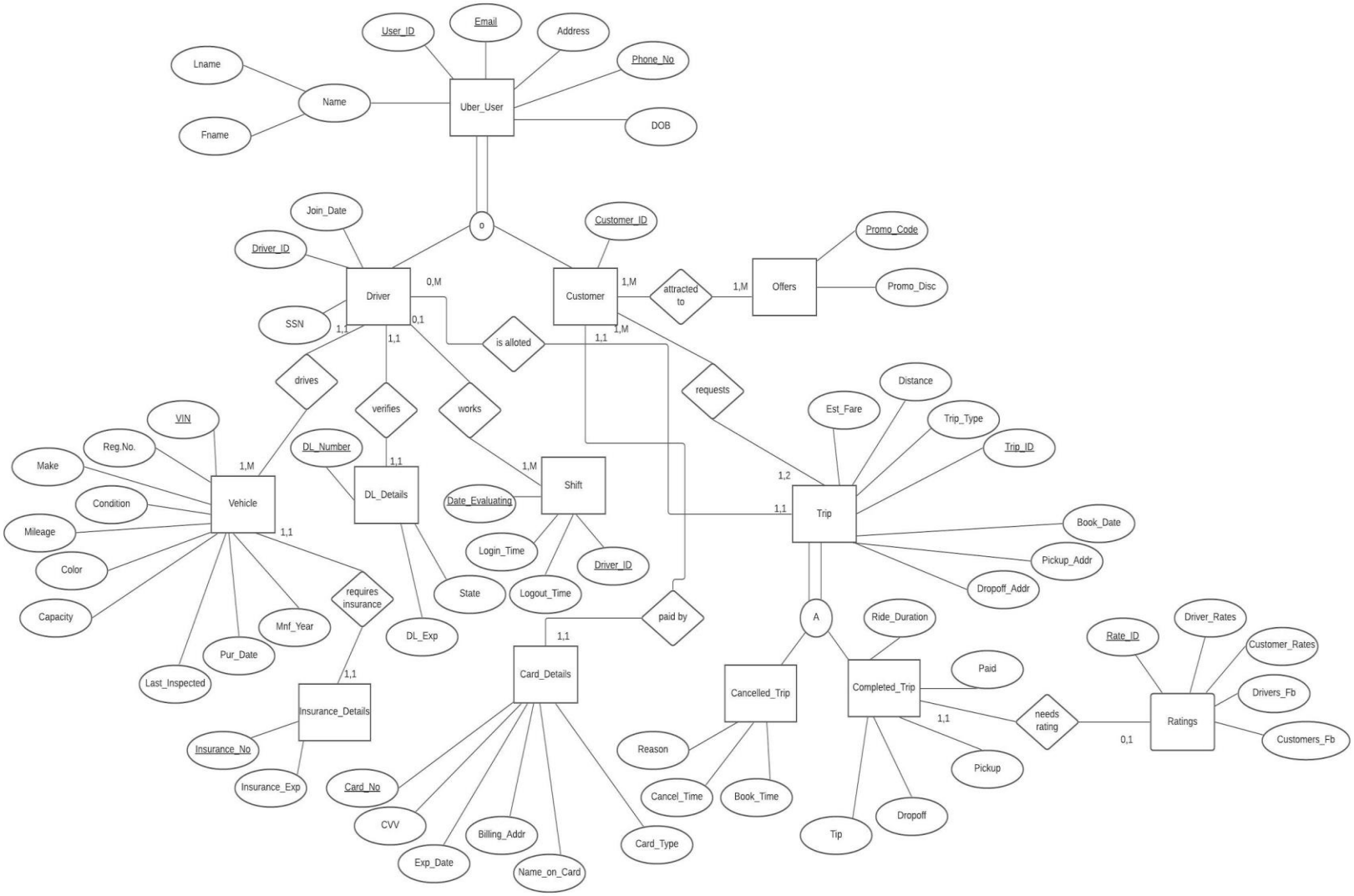
CANCELLED TRIP: To analyze the situation and provide feedback about the user, when the trip is cancelled, make sure to store the data about booking time, cancelation time and reason for cancellation.

While requesting for a trip, customers can enter promo code, if there are any **OFFERS** listed under the customer. Usually, providing discounts would help an organization to attract customers.

Modeling of Requirements as ER Diagram:

Considerations for relationships:

- A driver can drive one vehicle(in the case of changing the vehicle driver should update vehicle's information), at the same time multiple drivers can drive one registered vehicle.
- A driver can possess only one driver's license.
- A driver can login in one shift a day, and there can be multiple drivers logged in at the same shift.
- Trip can only be accepted by one driver, once the driver is assigned, the trip will not be available for other users.
- Vehicle can be registered with only one insurance company for a duration of time.
- Trip can be shared with two customers at the max.
- Customer can have access to multiple promo codes, at the same time multiple customers can use the same promo code.
- For a trip amount is deducted by the customer's registered card with Uber.
- For a trip users may or may not give the ratings.
- Customers have at least one offer(welcome offer).


User_ID
Email
Address
Phone_No
Lname
Name
Fname
Uber_User
DOB
Join_Date
Driver_ID
o
Customer_ID
Promo_Code
Promo_Disc
Driver
0,M
Customer
1,M
attracted to
1,M
Offers
SSN
1,1
0,1
is alloted
1,1
1,M
drives
verifies
works
requests
VIN
Reg.No.
Make
Condition
Mileage
Color
Capacity
Last_Inspected
Pur_Date
Mnf_Year
1,M
Vehicle
1,1
requires insurance
1,1
Insurance_Details
Insurance_No
Insurance_Exp
DL_Number
1,1
DL_Details
DL_Exp
State
1,M
Shift
Date_Evaluating
Login_Time
Logout_Time
Driver_ID
paid by
1,1
Card_Details
Card_No
CVV
Exp_Date
Billing_Addr
Name_on_Card
Card_Type
Distance
Est_Fare
Trip_Type
Trip_ID
1,2
1,1
Trip
Book_Date
Pickup_Addr
Dropoff_Addr
A
Ride_Duration
Cancelled_Trip
Completed_Trip
Paid
Reason
Cancel_Time
Book_Time
1,1
needs rating
0,1
Pickup
Tip
Dropoff
Rate_ID
Driver_Rates
Customer_Rates
Ratings
Drivers_Fb
Customers_Fb

Mapping of ERD to Relational Schema

Mapping of generalization and specialization:

Uber_User has two subclasses Driver and Customer. The specialization has total participation. Therefore can use 8a or 8b to create the relation. Used 8a to create Uber_User with common attributes, customer and driver have their attributes.

M:N Relationships:

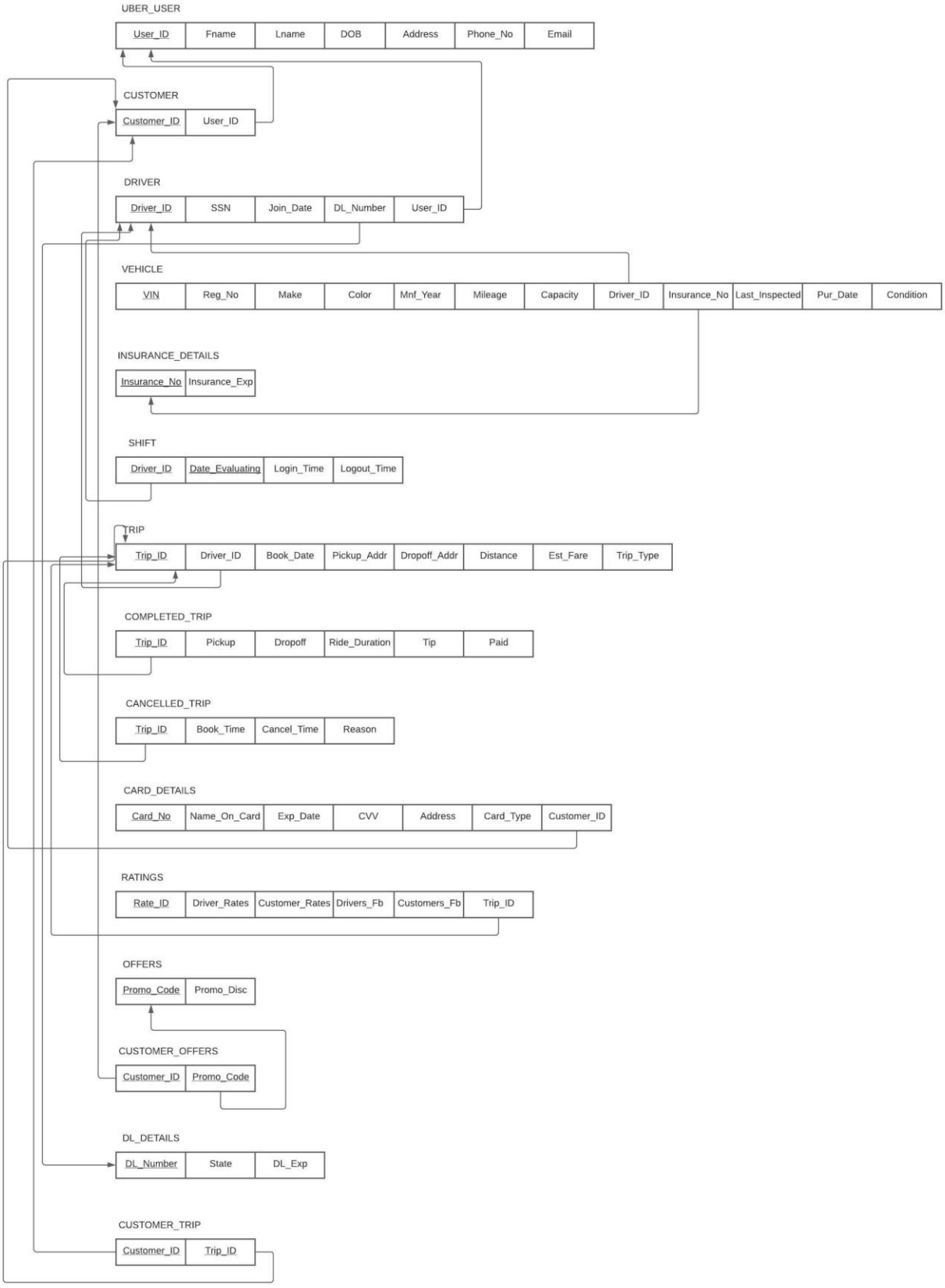
Relations involved in Requests and attracted_to have M:N relationship and thus a new relation is created, which has foreign key, primary keys of the entities related using the relationship and has attributes of the relationship. The key value for this relation is the combination of primary keys of each of the participating entities.

1:1 Relationships:

Relations involved in Verifies, Requires_Insurance, Paid_By, Needs_Rating are having 1:1 relationship.Used foreign key approach, which says to include primary key of either of the entity as foreign key in other entity relation. It is recommended to include foreign key in the relation which has total participation.

1:N Relationships:

Relations involved in Drives, Works and Is_Alloted have 1:N relationship. Identify the relation that represents the participating entity type at N-side of the relationship and include foreign key as the primary key of the relation of other entity participating in the relationship.


UBER_USER
User_ID
Fname
Lname
DOB
Address
Phone_No
Email
CUSTOMER
Customer_ID
User_ID
DRIVER
Driver_ID
SSN
Join_Date
DL_Number
User_ID
VEHICLE
VIN
Reg_No
Make
Color
Mnf_Year
Mileage
Capacity
Driver_ID
Insurance_No
Last_Inspected
Pur_Date
Condition
INSURANCE_DETAILS
Insurance_No
Insurance_Exp
SHIFT
Driver_ID
Date_Evaluating
Login_Time
Logout_Time
TRIP
Trip_ID
Driver_ID
Book_Date
Pickup_Addr
Dropoff_Addr
Distance
Est_Fare
Trip_Type
COMPLETED_TRIP
Trip_ID
Pickup
Dropoff
Ride_Duration
Tip
Paid
CANCELLED_TRIP
Trip_ID
Book_Time
Cancel_Time
Reason
CARD_DETAILS
Card_No
Name_On_Card
Exp_Date
CVV
Address
Card_Type
Customer_ID
RATINGS
Rate_ID
Driver_Rates
Customer_Rates
Drivers_Fb
Customers_Fb
Trip_ID
OFFERS
Promo_Code
Promo_Disc
CUSTOMER_OFFERS
Customer_ID
Promo_Code
DL_DETAILS
DL_Number
State
DL_Exp
CUSTOMER_TRIP
Customer_ID
Trip_ID

Normalization of Relational Schema

1. Uber_User: {**User_ID** →Fname, Lname, DOB, Address, Phone_No, Email}
2. Customer: {**Customer_ID** → *User_ID*}
3. Driver: {**Driver_ID** → SSN, Join_Date, *DL_Number, User_ID*}
4. Vehicle: {**VIN**→Reg_No, Make, Color, Mnf_Year, Mileage, Capacity, *Driver_ID*, Insurance_No, Last_Inspected, Pur_Date, Condition}
5. Insurance_Info: {**Insurance_No**→Insurance_Exp}
6. DL_Details: {**DL_Number**→DL_Exp, State}
7. Shift: {**Driver_ID, Date_Evaluating**→ login time, logout time}
8. Trip: {**Trip_ID** → *Driver_ID,* Book_Date, Pickup_Addr, Dropoff_Addr, Distance, Est_Fare, Trip_Type}
9. Cancelled_Trip: {**Trip_ID** → Book_Time, Cancel_Time, Reason}
10. Complete Trip: {**Trip_ID** → Pickup, Dropoff, Ride_Duration, Tip, Paid}
11. Card_Details: {**Card_No** → Name_on_Card, Exp_Date, CVV, Billing_Addr, Card_Type, *Trip_ID*}
12. Ratings: {**Rate_ID** → Driver_Rates, Customer_Rates, Drivers_Fb, Customers_Fb, *Trip_ID*}
13. Offers: {**Promo_Code** → Promo_Disc, *Customer_ID*}
14. Customer_Trip: {**Customer_ID, Trip_ID**}
15. Customer_Offers: {**Customer_ID, Promo_Code**}

The above schema is in 3NF.

As the schema is already in 3NF relational schema would be the same as above.

SQL Statements to Create Relations in DB and Add Constraints

```
CREATE TABLE UBER_USER(
USER_ID         VARCHAR(15) NOT NULL,
FNAME           VARCHAR(25)  NOT NULL,
LNAME           VARCHAR(25)  NOT NULL,
DOB             DATE         NOT NULL,
ADDRESS         VARCHAR(25)  NOT NULL,
PHONE_NO        INTEGER      NOT NULL UNIQUE,
EMAIL           VARCHAR(25)  NOT NULL UNIQUE,
PRIMARY KEY(USER_ID));

CREATE TABLE CUSTOMER(
CUSTOMER_ID         VARCHAR(15)  NOT NULL,
USER_ID             VARCHAR(15) NOT NULL,
PRIMARY KEY(CUSTOMER_ID));

CREATE TABLE DRIVER(
DRIVER_ID       VARCHAR(15)  NOT NULL,
SSN             CHAR(9)      NOT NULL,
JOIN_DATE       DATE         NOT NULL,
DL_NUMBER       CHAR(8)      NOT NULL,
USER_ID         VARCHAR(15) NOT NULL,
PRIMARY KEY(DRIVER_ID));
```

```
--Not needed to define as foregin key leave it as it is
CREATE TABLE VEHICLE(
VIN             CHAR(17)        NOT NULL,
REG_NO          CHAR(7)         NOT NULL,
MAKE            VARCHAR(25)  NOT NULL,
COLOR           VARCHAR(25)  NOT NULL,
MNF_YEAR        INTEGER                 NOT NULL,
MILEAGE                 INTEGER                 NOT NULL,
CAPACITY        INTEGER         NOT NULL,
DRIVER_ID       VARCHAR(15)  NOT NULL,
INSURANCE_NO    CHAR(9)          NOT NULL,
LAST_INSPECTED  INTEGER          NOT NULL,
PUR_DATE        DATE             NOT NULL,
CONDITION       VARCHAR(25)  NOT NULL,
PRIMARY KEY(VIN));

CREATE TABLE INSURANCE_DETAILS(
INSURANCE_NO CHAR(9)        NOT NULL,
INSURANCE_EXP DATE          NOT NULL,
PRIMARY KEY(INSURANCE_NO));

CREATE TABLE DL_DETAILS(
DL_NUMBER               CHAR(8)         NOT NULL,
STATE                   VARCHAR(25)  NOT NULL,
DL_EXP                  DATE            NOT NULL,
PRIMARY KEY(DL_NUMBER));
```

```
CREATE TABLE SHIFT(
DRIVER_ID           VARCHAR(15)  NOT NULL,
DATE_EVALUATING     DATE         NOT NULL,
LOGIN_TIME          TIMESTAMP    NOT NULL,
LOGOUT_TIME         TIMESTAMP    NOT NULL,
PRIMARY KEY(DRIVER_ID,DATE_EVALUATING));

CREATE TABLE TRIP(
TRIP_ID             VARCHAR(15)  NOT NULL,
DRIVER_ID           VARCHAR(15)  NOT NULL,
BOOK_DATE           DATE         NOT NULL,
PICKUP_ADDR         VARCHAR(25)  NOT NULL,
DROPOFF_ADDR        VARCHAR(25)  NOT NULL,
DISTANCE            INTEGER      NOT NULL,
EST_FARE            INTEGER      NOT NULL,
TRIP_TYPE           VARCHAR(25)  NOT NULL,
PRIMARY KEY(TRIP_ID));

CREATE TABLE CANCELLED_TRIP(
TRIP_ID             VARCHAR(15)  NOT NULL,
BOOK_TIME           TIMESTAMP    NOT NULL,
CANCEL_TIME         TIMESTAMP    NOT NULL,
REASON              VARCHAR(50)  NOT NULL,
PRIMARY KEY(TRIP_ID));
```

```
CREATE TABLE COMPLETED_TRIP(
TRIP_ID          VARCHAR(15)  NOT NULL,
PICKUP           TIMESTAMP    NOT NULL,
DROPOFF          TIMESTAMP    NOT NULL,
RIDE_DURATION    INTEGER      NOT NULL,
TIP              INTEGER      NOT NULL,
PAID             INTEGER      NOT NULL,
PRIMARY KEY(TRIP_ID));

CREATE TABLE CARD_DETAILS(
CARD_NO          CHAR(12)     NOT NULL,
NAME_ON_CARD     VARCHAR(25)  NOT NULL,
EXP_DATE         DATE         NOT NULL,
CVV              CHAR(3)      NOT NULL,
BILLING_ADDR     VARCHAR(50)  NOT NULL,
CARD_TYPE        VARCHAR(25)  NOT NULL,
CUSTOMER_ID      VARCHAR(15)  NOT NULL,
PRIMARY KEY(CARD_NO));

CREATE TABLE RATINGS(
RATE_ID          VARCHAR(15)  NOT NULL,
DRIVER_RATES     INTEGER      NOT NULL,
CUSTOMER_RATES   INTEGER      NOT NULL,
DRIVERS_FB       VARCHAR(25)  NOT NULL,
CUSTOMERS_FB     VARCHAR(25)  NOT NULL,
TRIP_ID          VARCHAR(15)  NOT NULL,
```

```
PRIMARY KEY(RATE_ID));

CREATE TABLE OFFERS(
PROMO_CODE      CHAR(8)      NOT NULL,
PROMO_DISC      INTEGER      NOT NULL,
PRIMARY KEY(PROMO_CODE));

CREATE TABLE CUSTOMER_OFFERS(
CUSTOMER_ID     VARCHAR(15)  NOT NULL,
PROMO_CODE      CHAR(8)      NOT NULL,
PRIMARY KEY(CUSTOMER_ID, PROMO_CODE));

CREATE TABLE CUSTOMER_TRIP(
CUSTOMER_ID     VARCHAR(15)  NOT NULL,
TRIP_ID         VARCHAR(15)  NOT NULL,
PRIMARY KEY(CUSTOMER_ID, TRIP_ID));
```

```
ALTER TABLE DRIVER ADD CONSTRAINT FKDL FOREIGN KEY (DL_NUMBER) REFERENCES DL_DETAILS(DL_NUMBER);

ALTER TABLE VEHICLE ADD CONSTRAINT FKDID FOREIGN KEY (DRIVER_ID) REFERENCES DRIVER(DRIVER_ID);

ALTER TABLE VEHICLE ADD CONSTRAINT FKINSURANCE FOREIGN KEY (INSURANCE_NO) REFERENCES INSURANCE_DETAILS(INSURANCE_NO);

ALTER TABLE SHIFT ADD CONSTRAINT FKDRIVERID FOREIGN KEY (DRIVER_ID) REFERENCES DRIVER(DRIVER_ID);

--ALTER TABLE OFFERS ADD CONSTRAINT FKCID FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID);
```

```
ALTER TABLE TRIP ADD CONSTRAINT FKDRIVEID FOREIGN KEY (DRIVER_ID) REFERENCES DRIVER(DRIVER_ID);

ALTER TABLE TRIP ADD CONSTRAINT FKC_ID FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID);

ALTER TABLE CARD_DETAILS ADD CONSTRAINT FKTID FOREIGN KEY (TRIP_ID) REFERENCES TRIP(TRIP_ID);

ALTER TABLE RATINGS ADD CONSTRAINT FKT_ID FOREIGN KEY (TRIP_ID) REFERENCES TRIP(TRIP_ID);

ALTER TABLE customer_offers ADD CONSTRAINT FKCID FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID);

ALTER TABLE customer_offers ADD CONSTRAINT FKPC FOREIGN KEY (PROMO_CODE) REFERENCES OFFERS(PROMO_CODE);

ALTER TABLE customer_trip ADD CONSTRAINT FK_CuID FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID);

ALTER TABLE customer_trip ADD CONSTRAINT FK_TID FOREIGN KEY (TRIP_ID) REFERENCES TRIP(TRIP_ID);

ALTER TABLE CUSTOMER ADD CONSTRAINT FKUID FOREIGN KEY (USER_ID) REFERENCES UBER_USER(USER_ID);

ALTER TABLE DRIVER ADD CONSTRAINT FK_UID FOREIGN KEY (USER_ID) REFERENCES UBER_USER(USER_ID);
```

Stored Procedures:

1. Stored Procedure to Calculate Average Ratings of all Drivers:

```
create or replace PROCEDURE Average_Rating AS

CURSOR Driver_Rating IS SELECT AVG(R.Customer_Rates) as Avg_Rating, T.Driver_ID FROM

Trip T, Ratings R WHERE T.Trip_ID=R.Trip_ID GROUP BY T.Driver_ID;

thisRating Driver_Rating%ROWTYPE;

BEGIN

OPEN Driver_Rating;

LOOP

FETCH Driver_Rating INTO thisRating;

EXIT WHEN (Driver_Rating%NOTFOUND);

dbms_output.put_line(thisRating.Avg_Rating || ' is the Average rating for the driver ID:'

|| thisRating.Driver_ID);

END LOOP;

CLOSE Driver_Rating;

END;

begin

Average_Rating;

End;
```

```
171 create or replace PROCEDURE Average_Rating AS
172 CURSOR Driver_Rating IS SELECT AVG(R.Customer_Rates) as Avg_Rating, T.Driver_ID FROM
173 Trip T, Ratings R WHERE T.Trip_ID=R.Trip_ID GROUP BY T.Driver_ID;
174 thisRating Driver_Rating%ROWTYPE;
175 BEGIN
176 OPEN Driver_Rating;
177 LOOP
178 FETCH Driver_Rating INTO thisRating;
179 EXIT WHEN (Driver_Rating%NOTFOUND);
180 dbms_output.put_line(thisRating.Avg_Rating || ' is the Average rating for the driver ID:'
181 || thisRating.Driver_ID);
182 END LOOP;
183 CLOSE Driver_Rating;
184 END;
185
186 begin
187 Average_Rating;
188 End;
189
190
191
192
193
194
195
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

```
Procedure AVERAGE_RATING compiled

Elapsed: 00:00:00.200



5 is the Average rating for the driver ID:U143
5 is the Average rating for the driver ID:U333
3 is the Average rating for the driver ID:U621
5 is the Average rating for the driver ID:U595
5 is the Average rating for the driver ID:U186


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.013
```

2. Stored Procedure to Calculate Total Fare for a given Ride:

```
create or replace PROCEDURE Calculate_Fare (base_fare IN number, tax IN number, cost_per_mile IN number) AS

CURSOR Total_Fare IS

SELECT

CT.Trip_ID as TID, CT.Tip as TIP, T.Distance as Dist

FROM

Trip T,

Completed_Trip CT

WHERE

T.Trip_ID=CT.Trip_ID;

thisTrip Total_Fare %rowtype;

thisTotalFare Completed_Trip.Paid%TYPE;

BEGIN

OPEN Total_Fare ;

LOOP

FETCH Total_Fare INTO thisTrip;

EXIT WHEN (Total_Fare %NOTFOUND);

thisTotalFare:= (base_fare + tax + cost_per_mile*thisTrip.Dist + thisTrip.TIP);

dbms_output.put_line(thisTotalFare || ' is the total fare for the Trip ID:' || thisTrip.TID);

END LOOP;

CLOSE Total_Fare;

END;


Begin
```

Calculate_Fare(5,10,1);

End;

```
252 create or replace PROCEDURE Calculate_Fare (base_fare IN number, tax IN number, cost_per_mile IN number) AS
253 CURSOR Total_Fare IS
254 SELECT
255 CT.Trip_ID as TID, CT.Tip as TIP, T.Distance as Dist
256 FROM
257 Trip T,
258 Completed_Trip CT
259 WHERE
260 T.Trip_ID=CT.Trip_ID;
261 thisTrip
262 Total_Fare %rowtype;
263 thisTotalFare Completed_Trip.Paid%TYPE;
264 BEGIN
265 OPEN
266 Total_Fare ;
267 LOOP
268 FETCH
269 Total_Fare INTO thisTrip;
270 EXIT WHEN
271 (Total_Fare %NOTFOUND);
272 thisTotalFare:= (base_fare + tax + cost_per_mile*thisTrip.Dist + thisTrip.TIP);
273 dbms_output.put_line(thisTotalFare || ' is the total fare for the Trip ID:' || thisTrip.TID);
274 END LOOP;
275 CLOSE
276 Total_Fare;
277 END;
278
279
280
281 Begin
282 Calculate_Fare(5,10,1);
283 END;
284
285
286
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

```
25 is the total fare for the Trip ID:T500
48 is the total fare for the Trip ID:T600
40 is the total fare for the Trip ID:T700
33 is the total fare for the Trip ID:T800
29 is the total fare for the Trip ID:T900


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009
```

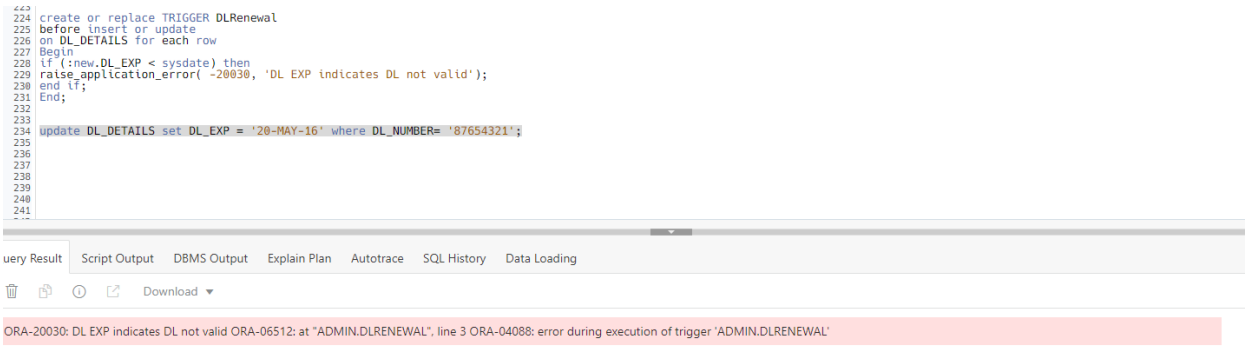
Triggers:

1. Trigger to check if the driver license is expired

```
create or replace TRIGGER DLRenewal
before insert or update
on DL_DETAILS for each row
Begin
if (:new.DL_EXP < sysdate) then
raise_application_error( -20030, 'DL EXP indicates DL not valid');
end if;
End;
```

After compiling, run the below query:

```
update DL_DETAILS set DL_EXP = '20-MAY-16' where DL_NUMBER= '87654321';
```

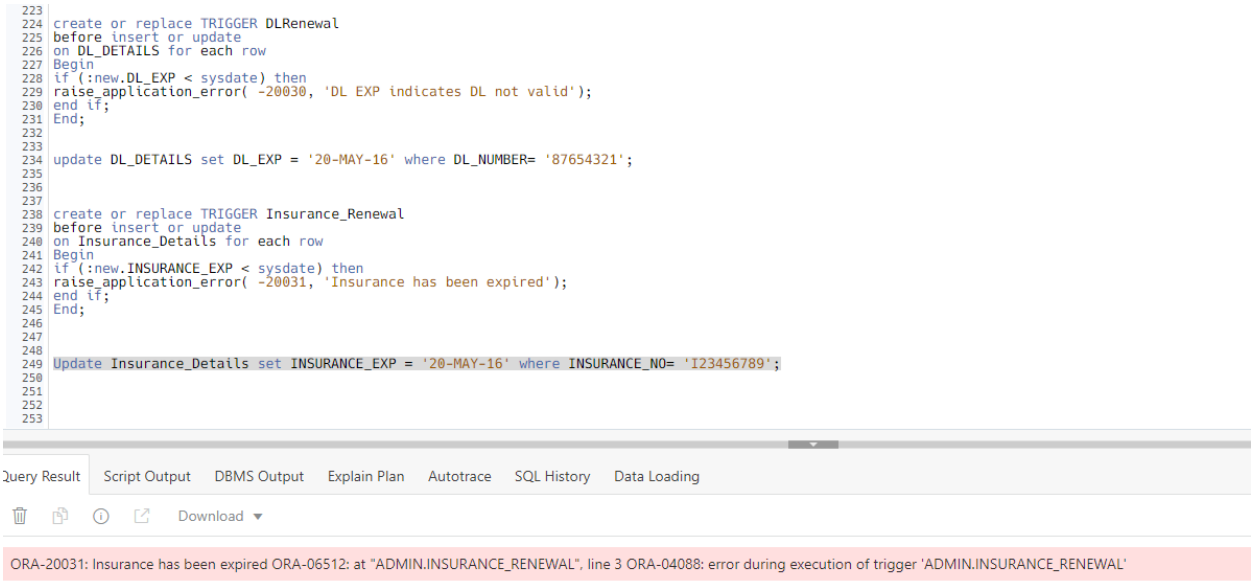


2. Trigger to check if the Insurance of a vehicle is expired

```
create or replace TRIGGER Insurance_Renewal
before insert or update
on DL_DETAILS for each row
Begin
if (:new.INSURANCE_EXP < sysdate) then
raise_application_error( -20031, 'Insurance has been expired');
end if;
End;
```

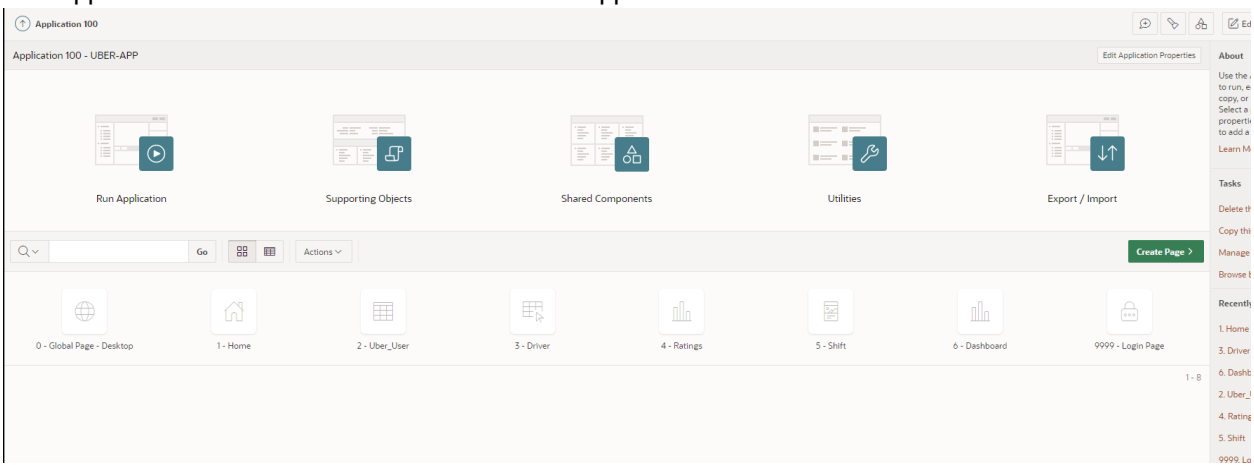
After compiling, run the below query:

```
Update Insurance_Details set INSURANCE_EXP = '20-MAY-16' where INSURANCE_NO= 'I23456789';
```

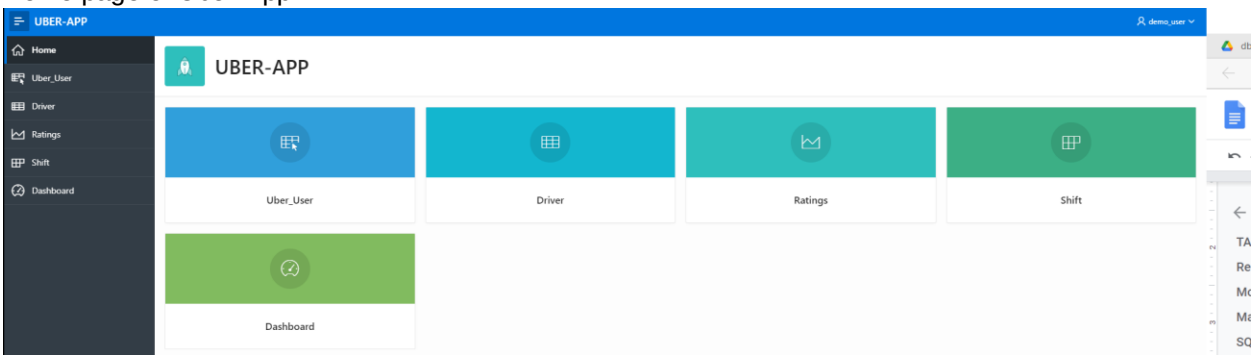


APEX:

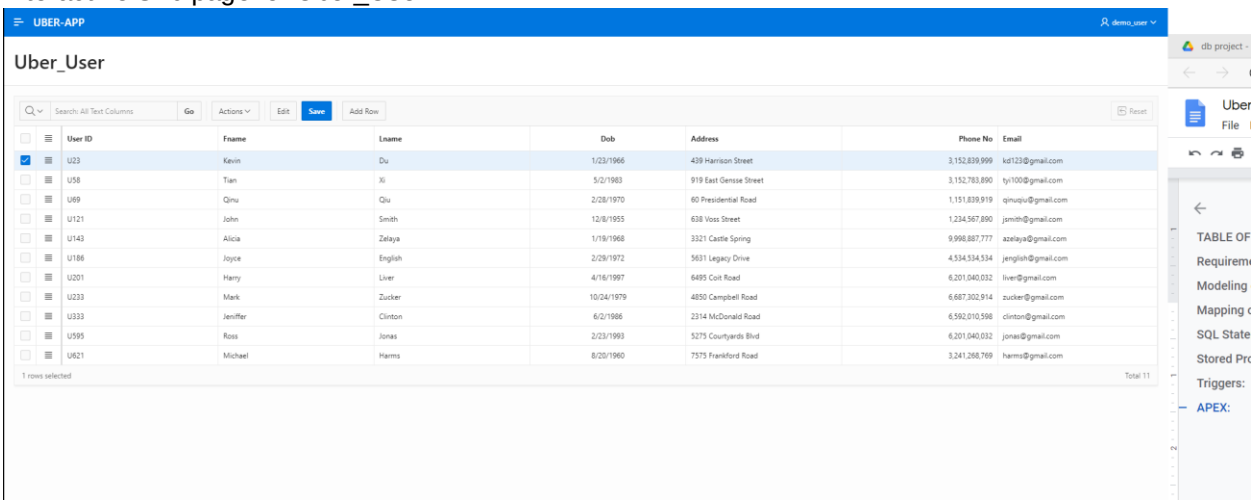
Run application and enter credentials to view the app:



Home page of Uber-App:



Interactive Grid page for Uber_User:



Interactive Report page for Driver:

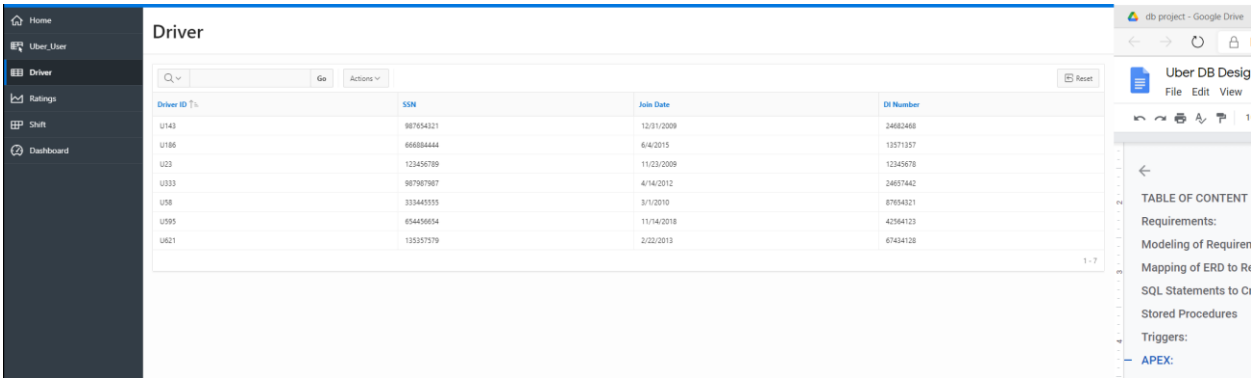
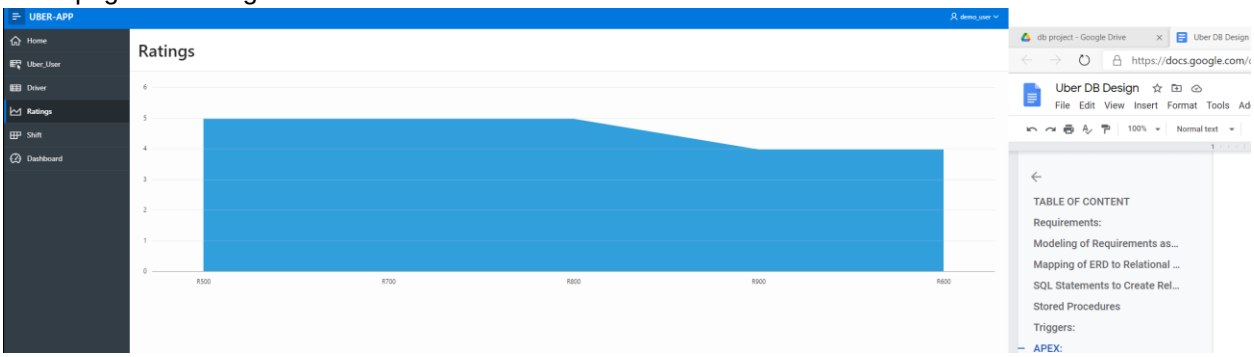


Chart page for Ratings:



Cards page for shift:

Dashboard page for multiple tables: