TAXI SERVICE

04.24.2018

—DATABASE DESIGN (CS6360.002)

Sandeep Govindaraj (SXG175630)
Mrunmayee Sanjay Jangam (mxj171430)

Requirements

Taxi Service Scope

We have designed a database for an online taxi service company which spans over multiple locations. We have our own assumptions on the services offered to the customer and employees through an application as follows:

- Taxi Request
    - Customer's request for a taxi through an application, a driver is assigned to the request on the server side based on the location, rating etc.
    - Driver communicates with the customer, picks up the customer and drops him at the destination.
    - Customer provides feedback for the driver and the vehicle which will be used as a criterion in assigning the appropriate taxi.
- Various trip attributes such as wait time, miles covered, busy times (based on booking timestamp) etc. are stored for better operational analysis of the business.
- Track Gasoline receipts, maintenance charges incurred for every vehicle which contributes in devising a profitable business
- Drivers can drive any vehicle provided they have license to do so provided the vehicle is available.
- Drivers utilization can be computed based on the recorded trip history which should be used for better management of resources.
- Vehicle service history are audited for better maintenance of the same.
- Vehicle’s insurance details are tracked so that information regarding insurance renewal can be informed in advance.
- Customers trips are audited to track total trips, miles travelled etc. which is a valuable information which can be utilized for a reward system.
- The taxi service can be owned by multiple people, the net income is shared by all the owners and the owners decide on the employee's share over each trip on a daily a basis.

Service Structure

The company covers services in multiple locations and services are online through an application. For every location, vehicles and drivers are mapped to that location and can change location whenever a destination of a trip is outside it’s currently assigned area. Customers book taxi’s through an application with the trip details and valuable feedbacks from the customer are recorded. Customers payments are directed to a single account, from where drivers are paid daily by the owners based on their contributionfor the day. Customers cumulative trip history is monitored for a rewards system. Drivers contributionand efforts are tracked daily, which will be used by owner to decide on the drivers take away percentageover his trips. Vehicle’s maintenance and insurance history are tracked and readily available to ownerfor better asset management. Expenses on Gasoline, car services, car insurances, driver’s salaries are alltracked to generate net income and further devise plans on optimizations.

Driver's Responsibilities & Benefits

On completion of every trip, driver gets tip from the customer and part of the money paid for that trip based on the share % defined by owner for that driver for every trip. Based on driver's quality of service, customer provides feedbacks which is considered as a metric to evaluate drivers. Evaluation of drivers is vital, which puts them in front of other drivers for an increment and get advantage over other drivers when assigning a driver to a trip. Drivers are supposed to turn in bills on gasoline to the owner at the end of each day for reimbursements.

Customer's Value & Rewards

On completion of every trip, customer completes more trip and travels more miles with the taxi service and so are provided with rewards on future trips as configured by the owner via the application. Moreover, feedbacks provided by the customer are of great value and are important for the owner to analyze driver's performance and it affects the benefits for that driver. Customers feedback over the vehicle is important for the owner to analyze the condition of vehicles etc.

Asset Management

Vehicles and drivers are the most important aspects of a taxi service, asset management is a vital part of any business. Vehicle maintenance is monitored by keeping track of the service dates and complaints over the vehicle by customers time to time. Owners keep track of customer feedback for drivers, to better assess them for their contribution.

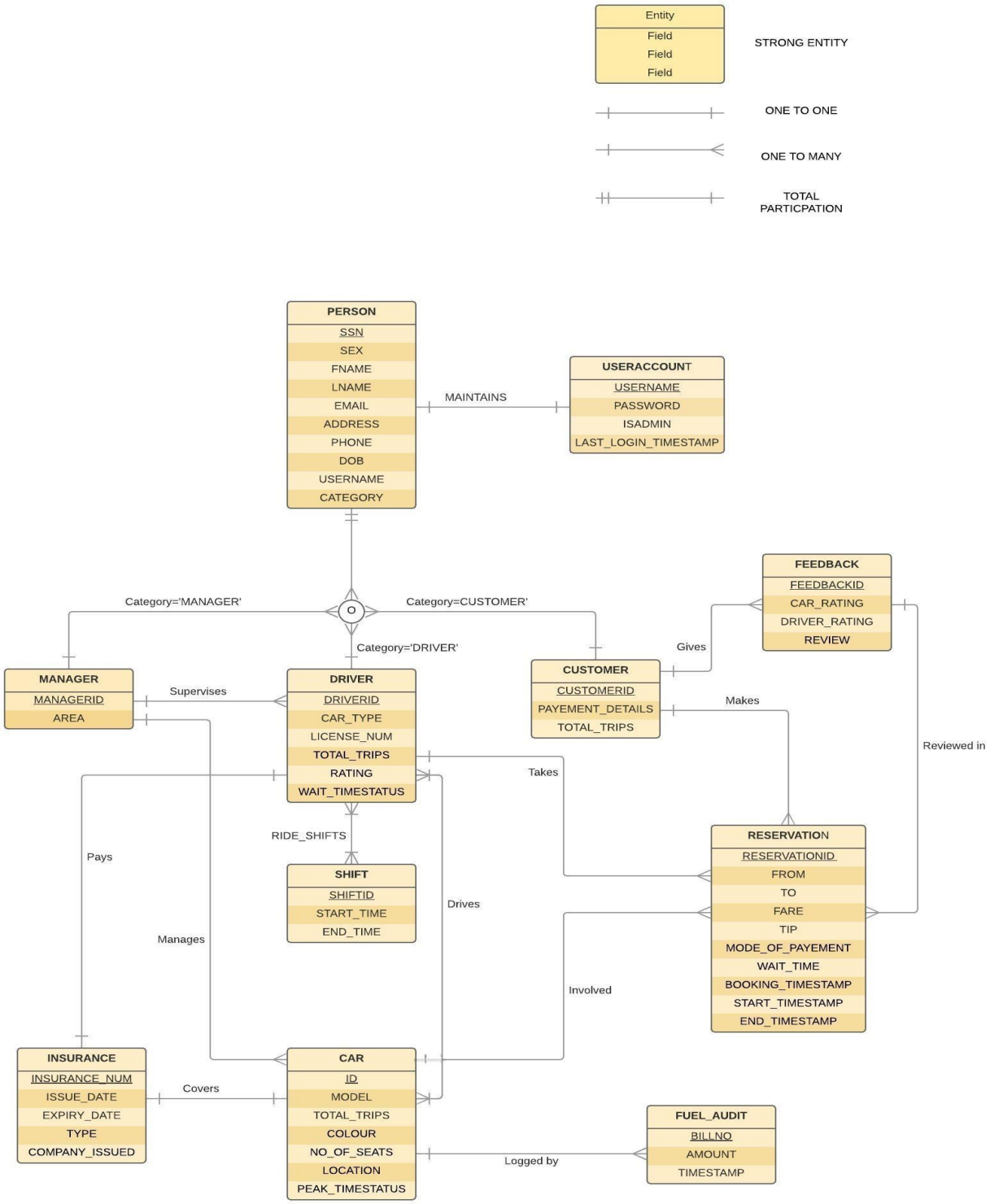
Owner's Responsibilities & Benefits

Owner is responsible to fix the static variables such as proportion of money from every trip to be given to a driver. Owner's will be intimated by the application about expiry of insurance for vehicles, and it's the sole duty of the owners to renew and update the insurance details in the application. Owners can view reports on net income, driver utilization, fuel expenses, repair expenses day wise which enables them to have a clear picture of flow of money within the business. The net income is shared by all the owners based on their shares over the business.

Auditing

Every business application requires proper auditing for recovery and reporting. We have audited various key performance indicators such as rating given for drivers, rating for vehicles and expenses on fuel, insurance and service which prove to be valuable in creating reports to measure net income, utilization of drivers, asset evaluation etc.

ER-Diagram


Entity
Field
Field
Field
STRONG ENTITY
ONE TO ONE
ONE TO MANY
TOTAL PARTICPATION
PERSON
SSN
SEX
FNAME
LNAME
EMAIL
ADDRESS
PHONE
DOB
USERNAME
CATEGORY
MAINTAINS
USERACCOUNT
USERNAME
PASSWORD
ISADMIN
LAST_LOGIN_TIMESTAMP
Category='MANAGER'
O
Category=CUSTOMER'
Category='DRIVER'
FEEDBACK
FEEDBACKID
CAR_RATING
DRIVER_RATING
REVIEW
Gives
MANAGER
MANAGERID
AREA
Supervises
DRIVER
DRIVERID
CAR_TYPE
LICENSE_NUM
TOTAL_TRIPS
RATING
WAIT_TIMESTATUS
CUSTOMER
CUSTOMERID
PAYEMENT_DETAILS
TOTAL_TRIPS
Makes
Reviewed in
Takes
RIDE_SHIFTS
Pays
SHIFT
SHIFTID
START_TIME
END_TIME
Drives
RESERVATION
RESERVATIONID
FROM
TO
FARE
TIP
MODE_OF_PAYEMENT
WAIT_TIME
BOOKING_TIMESTAMP
START_TIMESTAMP
END_TIMESTAMP
Manages
Involved
INSURANCE
INSURANCE_NUM
ISSUE_DATE
EXPIRY_DATE
TYPE
COMPANY_ISSUED
Covers
CAR
ID
MODEL
TOTAL_TRIPS
COLOUR
NO_OF_SEATS
LOCATION
PEAK_TIMESTATUS
Logged by
FUEL_AUDIT
BILLNO
AMOUNT
TIMESTAMP

Modeling of Requirements as ER-Diagram

The requirements can be summarized/ derived from ERD as

- Adding a Category attribute to the Person entity to categorize which type of person he/she is, Either Manager or Driver or Customer.
- The Person entity will have total participation and the entities such as Manager, Driver and Customer are derived from this.
- Each Person will have at most one user account (1:1) .
- Entities such as Customer, Driver and Car collectively involved in a Reservation. All these are mapped as (1:M) since one Customer can book one or more Reservations, each driver can be allocated with many Reservations and a single Car can be evenly involved in all the Reservations.
- Car could have one or more allocated drivers(normally or depends on the shifts) and drivers can drive multiple Cars. So, it's M:N relationship.
- Each Driver can have multiple shifts and each Shift can be allocated to multiple Drivers(M:N)
- Each Customer writes a feedback regarding their recent Reservation. One Customer can write more than one feedback (1:M) and each feedback will be assigned to the particular Reservation(1:1).
- Owner maintains a 1:M relationship with both Cars and the drivers. He can own more Cars(1:M) and supervise the drivers(1:M).
- Insurance upholds a 1:1 relationship with the driver and Car as driver is the one who pays for the insurance and ensure that the Car's license is not expired.
- Car logs audit for gasoline that it had consumed and services undertaken during a tenure. It maintains a 1:M relationship.

Mapping of ERD in Relational Schema

The following tables has been mapped using **OPTION 8A (Multiple relations-Superclass and subclasses).**

Note: We have to create a separate table for a relation that has M:N binary relationship types. In our case, DEDICATED_SHIFTS and DRIVES relations will have a separate tables which hold foreign keys of {Driver,Vehicle} for RIDE_SHIFTS and {Driver,Shift} for DRIVES respectively.

I. PERSON

SSN	SEX	FNAME	LNAME	EMAIL	ADDRESS	PHONE	DOB	USERNAME	CATEGORY

- Primary Key : SSN
- Foreign Key : FOREIGN KEY (USERNAME) REFERENCES USERACCOUNT(USERNAME)

II. DRIVER

DRIVERID	MID	DSSN	CAR_TYPE	TOTAL_TRIPS	LICENSE_NUM	RATING

- Primary Key : DRIVERID
- Foreign KeyS : FOREIGN KEY (DSSN) REFERENCES PERSON(SSN),FOREIGN KEY (MID) REFERENCES MANAGER (MANAGERID)

III. CUSTOMER

CUSTOMERID	CSSN	PAYEMENT_DETAILS	TOTAL_TRIPS

- Primary Key : CUSTOMERID
- Foreign Keys : FOREIGN KEY (CSSN) REFERENCES PERSON(SSN)

IV. MANAGER

MANAGERID	MSSN	AREA

- Primary Key : MANAGERID
- Foreign Keys : FOREIGN KEY (MSSN) REFERENCES PERSON(SSN)

V. Shift

SHIFTID	START_TIME	END_TIME

- Primary Key : SHIFTID
- Foreign Keys : None

VI. UserAccount

USERNAME	PASSWORD	ISADMIN	LAST_LOGIN_TIMESTAMP

- Primary Key : Username
- Foreign Keys : None

VII. RESERVATION

RESERVATIONID	FROM	TO	FARE	TIP	MODE_OF_ PAYEMENT	CARID	CID

DID	WAIT_TIME	BOOKING_TIMESTAMP	START_TIMESTAMP	END_TIMESTAMP

- Primary Key : RESERVATIONID
- Foreign Keys : FOREIGN KEY (CARID) REFERENCES CAR(ID), FOREIGN KEY (CID) REFERENCES CUSTOMER (CUSTOMERID), FOREIGN KEY (DID) REFERENCES DRIVER (DRIVERID)

VIII. CAR

ID	MID	MODEL	TOTAL_TRIPS	COLOUR	NO_0F_SEATS	LOCATION

- Primary Key : ID
- Foreign Keys : FOREIGN KEY (MID) REFERENCES MANAGER (MANAGERID)

IX. FEEDBACK

FEEDBACKID	DRIVER_RATING	CAR_RATING	REVIEW	CID	RID

- Primary Key : FEEDBACKID
- Foreign Keys : FOREIGN KEY (CID) REFERENCES CUSTOMER(CUSTOMERID), FOREIGN KEY (RID) REFERENCES RESERVATION (RESERVATIONID)

X. FUEL_AUDIT

BILLNO	CARID	AMOUNT	TIMESTAMP

- Primary Key : BILLNO
- Foreign Keys : FOREIGN KEY (CARID) REFERENCES CAR(ID)

XI. INSURANCE

INSURANCE_NUM	ISSUE_DATE	EXPIRY_DATE	TYPE	COMPANY_ISSUED

- Primary Key : INSURANCE_NUM
- Foreign Keys : None

XII. DRIVES

DID	CARID

- Primary Key : DID,VID
- Foreign Keys : FOREIGN KEY (DID) REFERENCES DRIVER(DRIVERID), FOREIGN KEY (CARID) REFERENCES CAR(ID0)

XIII. DEDICATED SHIFTS

DID	SID

- Primary Key : DID,SID
- Foreign Keys : FOREIGN KEY (DID) REFERENCES DRIVER(DRIVERID), FOREIGN KEY (SID) REFERENCES SHIFT(SHIFTID)

Normalization of Relational Schema

The following Functional Dependencies exists in the relational schema (Note: To ensure consistency, the foreign attributes won't have the same name that we mentioned in the tables above. Instead, it will have the same name as the primary key).

- USERACCOUNT {USERNAME -> PASSWORD, ISADMIN, LAST_LOGIN_TIMESTAMP}
- PERSON {SSN -> SEX, FNAME,LNAME, EMAIL, PHONE, DOB, ADDRESS, CATEGORY, USERNAME}
- MANAGER{MANAGERID -> SSN, AREA}
- RESERVATION {RESERVATIONID -> FROM, TO, TIP,FARE, MODE_OF_PAYEMENT, TOTAL_MILES, WAIT_TIME, BOOKING_TIMESTAMP, START_TIMESTAMP, END_TIMESTAMP, DRIVERID, CUSTOMERID, CARID}
- DRIVER {DRIVERID -> SSN, MANAGERID, CAR_TYPE, TOTAL_TRIPS, LICENSE_NUM, RATING,WAIT_TIMESTATUS}
- CUSTOMER {CUSTOMERID -> SSN, PAYEMENT_DETAILS, TOTAL_TRIPS}
- CAR {CARID -> MODEL, MANAGERID, TOTAL_TRIPS, COLOUR, NO_OF_SEATS, LOCATION, INSURANCE_NUM,PEAK_TIMESTATUS}
- SHIFT {SHIFTID -> START_TIME, END_TIME}
- FEEDBACK {FEEDBACKID -> CUSTOMERID, CAR_RATING, DRIVER_RATING, REVIEW, RESERVATIONID}
- INSURANCE {INSURANCE_NUM -> EXPIRY_DATE, ISSUE_DATE, TYPE, COMPANNY_ISSUED}
- FUEL_AUDIT {BILLNO -> CARID, AMOUNT, TIMESTAMP}

- The above functional dependencies cause the schema to be in third normal form. The **candidate keys** are **{ SHIFTID, FEEDBACKID, BILLNO}**
- As the closure of**{ SHIFTID, FEEDBACKID, BILLNO}** can lead to all the attributes in the database.

DDL's to create Relations in DB and Add Constraints

The following DDLs has to be executed in tandem.

I. USERACCOUNT

```
CREATE TABLE USERACCOUNT(
USERNAME VARCHAR(40) NOT NULL,
PASSWORD VARCHAR(50),
ISADMIN VARCHAR(1),
LAST_LOGIN_TIMESTAMP DATE,
PRIMARY KEY (USERNAME));
```

II. PERSON

```
CREATE TABLE PERSON(
SSN INTEGER NOT NULL,
SEX VARCHAR(1),
FNAME VARCHAR(50),
LNAME VARCHAR(50),
EMAIL VARCHAR(50),
ADDRESS VARCHAR(100),
PHONE INTEGER,
DOB  DATE,
USERNAME VARCHAR(40),
CATEGORY VARCHAR(20),
```

```
PRIMARY KEY (SSN),
FOREIGN KEY (USERNAME) REFERENCES USERACCOUNT (USERNAME) ON DELETE CASCADE);
```

III. INSURANCE

```
CREATE TABLE INSURANCE (
INSURANCE_NUM INTEGER NOT NULL,
ISSUE_DATE DATE,
EXPIRY_DATE DATE,
TYPE VARCHAR(20),
COMPANY_ISSUED VARCHAR(10),
PRIMARY KEY (INSURANCE_NUM));
```

IV. MANAGER

```
CREATE TABLE MANAGER(
MANAGERID INTEGER NOT NULL,
AREA VARCHAR(50),
MSSN INTEGER,
PRIMARY KEY (MANAGERID),
FOREIGN KEY (MSSN) REFERENCES PERSON (SSN) ON DELETE CASCADE);
```

V. DRIVER

```
CREATE TABLE DRIVER(
DRIVERID INTEGER NOT NULL,
MID INTEGER,
DSSN INTEGER,
CAR_TYPE VARCHAR(20),
TOTAL_TRIPS INTEGER,
```

```
LICENSE_NUM INTEGER,
RATING DECIMAL(2,1),
WAIT_TIMESTATUS VARCHAR(1),
PRIMARY KEY (DRIVERID),
FOREIGN KEY (DSSN) REFERENCES PERSON(SSN) ON DELETE CASCADE,
FOREIGN KEY (MID) REFERENCES MANAGER(MANAGERID) ON DELETE CASCADE);
```

VI. CUSTOMER

```
CREATE TABLE CUSTOMER(
CUSTOMERID INTEGER NOT NULL,
CSSN INTEGER,
PAYEMENT_DETAILS VARCHAR(50),
TOTAL_TRIPS INTEGER,
PRIMARY KEY (CUSTOMERID),
FOREIGN KEY (CSSN) REFERENCES PERSON(SSN) ON DELETE CASCADE);
```

VII. CAR

```
CREATE TABLE CAR(
ID INTEGER NOT NULL,
MID INTEGER,
INS_NO INTEGER,
MODEL VARCHAR(10),
TOTAL_TRIPS INTEGER,
COLOUR VARCHAR(20),,
NO_OF_SEATS INTEGER,
LOCATION VARCHAR(20),
PEAK_TIMESTATUSVARCHAR(1),
PRIMARY KEY (ID),
```

```
FOREIGN KEY (MID) REFERENCES MANAGER(MANAGERID) ON DELETE CASCADE,
FOREIGN KEY (INS_NO) REFERENCES INSURANCE (INSURANCE_NUM) ON DELETE CASCADE);
```

VIII. RESERVATION

```
CREATE TABLE RESERVATION(
RESRVATIONID INTEGER NOT NULL,
FROM VARCHAR(50),
TO VARCHAR(50),
FARE DECIMAL(10,2),
TIP DECIMAL(10,2),
 MODE_OF_PAYEMENT VARCHAR(20),
CARID INTEGER,
CID INTEGER,
DID INTEGER,
WAIT_TIME DATE,
BOOKING_TIMESTAMP DATE,
START_TIMESTAMP DATE,
END_TIMESTAMP DATE,
PRIMARY KEY (RESRVATIONID),
FOREIGN KEY (CARID) REFERENCES CAR(ID) ON DELETE CASCADE,
FOREIGN KEY (CID) REFERENCES CUSTOMER(CUSTOMERID) ON DELETE CASCADE,
FOREIGN KEY (DID) REFERENCES DRIVER(DRIVERID) ON DELETE CASCADE);
```

IX. FEEDBACK

```
CREATE TABLE FEEDBACK(
FEEDBACKID INTEGER NOT NULL,
DRIVER_RATING DECIMAL(2,1),
CAR_RATING DECIMAL(2,1),
REVIEW VARCHAR(100),
```

```
CID INTEGER,
RID INTEGER,
PRIMARY KEY (FEEDBACKID),
FOREIGN KEY (CID) REFERENCES CUSTOMER(CUSTOMERID) ON DELETE CASCADE,
FOREIGN KEY (RID) REFERENCES RESERVATION(TRIID) ON DELETE CASCADE);
```

X. FUEL_AUDIT

```
CREATE TABLE FUEL_AUDIT(
BILLNO INTEGER NOT NULL,
CARID INTEGER,
Amount DECIMAL(8,2),
Timestamp DATE,
PRIMARY KEY (BILLNO),
FOREIGN KEY (CARID) REFERENCES CAR(ID) ON DELETE CASCADE);
```

XI. SHIFT

```
CREATE TABLE SHIFT(
SHIFTID INTEGER NOT NULL,
START_TIME DATE,
END_TIME DATE,
PRIMARY KEY(SHIFTID));
```

XII. DRIVES

```
CREATE TABLE DRIVES(
DID INTEGER NOT NULL,
CARID INTEGER NOT NULL,
PRIMARY KEY(DID,CARID),
FOREIGN KEY (DID) REFERENCES DRIVER(DRIVRID) ON DELETE CASCADE,
FOREIGN KEY (CARID) REFERENCES CAR(ID) ON DELETE CASCADE);
```

XIII. DEDICATED_SHIFTS

```
CREATE TABLE DEDICATED_SHIFTS(
DID INTEGER NOT NULL,
SID INTEGER NOT NULL,
PRIMARY KEY(DID,SID),
FOREIGN KEY (DID) REFERENCES DRIVER(DRIVERID) ON DELETE CASCADE,
FOREIGN KEY (SID) REFERENCES Shift(SHIFTID) ON DELETE CASCADE);
```

PL/SQL – TRIGGERS

Trigger 1: Update No_of_Trips for Driver and Passenger

➢ Once the customer booked a reservation, the respective entries will get populated into RESERVATION table. We executing the below trigger to keep track of how many no of trips that a particular driver and a particular customer made.

```
CREATE OR REPLACE TRIGGER INCREMENT_TOTAL_TRIPS_
   AFTER INSERT OR UPDATE OF END_TIMESTAMP ON RESERVATION
   FOR EACH ROW
BEGIN
   IF :NEW.END_TIMESTAMP IS NOT NULL THEN
      UPDATE DRIVER SET TOTAL_TRIPS = NO_OF_TRIPS+1 WHERE
      DRIVERID=:NEW.DID;
      UPDATE CUSTOMER
      SET
         TOTAL_TRIPS = TOTAL_TRIPS + 1,
         WHERE CUSTOMERID = :NEW.CID;
```

```
    END IF;
END;
```

Trigger 2: Update the current location of the vehicle

➢ Once the trip completes, we have to update the current location of the vehicle so that it could be available for next trips that are nearby. To implement this, we are using below trigger.

```
create or replace TRIGGER UPDATE_LOCATION
    AFTER INSERT OR UPDATE OF END_TIME_STAMP ON RESERVATION
    FOR EACH ROW
BEGIN
    IF :NEW.END_TIMESTAMP IS NOT NULL THEN
        UPDATE CAR
        SET
            LOCATION = :NEW.TO
            WHERE ID IN (SELECT CARID FROM DRIVES WHERE DID = :NEW.DID);
    END IF;
END;
```

Trigger 3: Update the fare during peak hours

➢ update the fare to 1.5x times the Normal fare during peak hours when the peak time status of the car is set to ‘Y’

```
CREATE OR REPLACE TRIGGER INCREASE_FARE
    AFTER INSERT OR UPDATE OF PEAK_TIME_STATUS ON CAR
    FOR EACH ROW
BEGIN
    IF :NEW.PEAK_TIME_STATUS IS NOT NULL AND :NEW.PEAK_TIME_STATUS='Y'
        THEN
        UPDATE RESERVATION
        SET
        FARE = 1.5*FARE
        WHERE CARID = :NEW.ID;
```

```
    END IF;
END;
```

Trigger 4: Update the wait time

➢ Update the wait time to 5 minutes when the driver reaches the location of the customer and the customer is yet to board.

```
CREATE OR REPLACE TRIGGER SET_WAITTIME
    AFTER INSERT OR UPDATE OF WAIT_TIME_STATUS ON DRIVER
    FOR EACH ROW
BEGIN
    IF :NEW.WAIT_TIME_STATUS IS NOT NULL AND :NEW.WAIT_TIME_STATUS='Y'
        THEN
        UPDATE RESERVATION
        SET
        WAIT_TIME = 5
        WHERE DID = :NEW.DRIVERID;
    END IF;
END;
```

PL/SQL – PROCEDURES

Procedure 1: Top 5 drivers

➢ Fetching the 5 top rated drivers on the average driver rating basis.

```
1.  CREATE OR REPLACE PROCEDURE TOP_RATED_DRIVERS IS
2.  CURSOR DRIVER_DETAILS IS
3.  SELECT * FROM
4.        (SELECT
5.        DRIVER.DRIVERID AS DRIVER_ID,
6.        PERSON.FNAME AS FNAME,
7.        PERSON.LNAME AS LNAME,
8.        DRIVER.RATING AS RATING
9.        FROM PERSON,DRIVER
10.       WHERE DRIVER.DSSN = PERSON.SSN
11.       ORDER BY DRIVER.RATING DESC)
12.       WHERE ROWNUM <= 5;
13.       DRIVER_DETAILS_ROW   DRIVER_DETAILS%ROWTYPE;
14. BEGIN
15.    OPEN DRIVER_DETAILS;
16.       LOOP
17.          FETCH DRIVER_DETAILS INTO DRIVER_DETAILS_ROW;
18.          EXIT WHEN DRIVER_DETAILS%NOTFOUND;
19.          DBMS_OUTPUT.PUT_LINE('Driver Name: '||DRIVER_DETS_ROW.FNAME||',
    '||DRIVER_DETS_ROW.LNAME);
20.          DBMS_OUTPUT.PUT_LINE('Average Rating: '||DRIVER_DETS_ROW.RATING);
21.       END LOOP;
22.    CLOSE DRIVER_DETS;
23. END;
```

Procedure 2: DISCOUNT FOR CUSTOMERS

➢ Fetching the 5 top prime customers who have travelled most trips within specified date range in order to provide them discounts.

```
1.  CREATE OR REPLACE PROCEDURE DISCOUNT_FOR_CUSTOMERS (START_DATE IN DATE,
    END_DATE IN  DATE) IS
1.  CURSOR CUSTOMER_DETAILS IS
2.  SELECT * FROM
3.        (SELECT
4.        CUSTOMER.CUSTOMERID AS CUSTOMER_ID,
5.        PERSON.FNAME AS FNAME,
6.        PERSON.LNAME AS LNAME,
7.        COUNT(RESERVATIONID) AS TRIPS_TRAVELLED,
8.        RESERVATION.END_TIMESTAMP AS END_TIMESTAMP
9.        FROM PERSON,CUSTOMER,RESERVATION
10.       WHERE CUSTOMER.CSSN = PERSON.SSN
11.       AND RESERVATION.CID=CUSTOMER.ID
12.  AND (RESERVATION.END_TIME_STAMP BETWEEN START_DATE AND END_DATE);
13.  ORDER BY RESERVATION.TRIPS_TRAVELLED DESC)
14.       WHERE ROWNUM <= 10;
15.       CUSTOMER_DETAILS_ROW   CUSTOMER_DETAILS%ROWTYPE;
16. BEGIN
17.    OPEN CUSTOMER_DETAILS;
18.       LOOP
19.          FETCH CUSTOMER_DETAILS INTO CUSTOMER_DETAILS_ROW;
20.          EXIT WHEN CUSTOMER_DETAILS%NOTFOUND;
21.  DBMS_OUTPUT.PUT_LINE('CUSTOMER NAME: '||CUSTOMER_DETAILS_ROW.FNAME||',
    '||CUSTOMER_DETAILS_ROW.LNAME);
22.       DBMS_OUTPUT.PUT_LINE('TOTAL TRIPS: '||CUSTOMER_DETAILS_ROW.TRIPS_TRAVELLED);
23.       END LOOP;
24.
25.    CLOSE DRIVER_DETS;
26. END;
```