TAXI MANAGEMENT SYSTEM

Name: Manya Smriti
Registeration No. :18BIT0127
Name: Tamanna Srivastava
Registeration No. : 18BIT0473
Slot :F2+TF2
Course: Database Management System
Course Code: ITE1003
Faculty Name: Bimal Kumar Ray
Year:2019
Month:November

ABSTRACT

The main aim of Taxi Management Mini DBMS project is to rent taxi and get payments from respective clients. We aim to demonstrate the use of create, read, update and delete MySQL operations through this project. The project starts by adding a taxi and by adding details of driver using the taxi added. The owner provides taxi to the drivers and ads their expenses on daily basis. Booking scene is where a customers can book a taxi to get o the desired location.

In this project we have tried to analyse data requirements and functional requirements for the taxi management system and implemented our project accordingly The data requirements, apart from data to be stored in the databasehas been taken into account, the necessary integrity constraints that are reasonable for the database are taken under consideration..

DBMS PROJECT --A TECHNICAL TOUR OF UBER TAXI MANAGEMENT SYSTEM

1. Choose a mini world for design and implementation of its a database assigning an appropriate title for the database.

The Uber Taxi Management System

Mini-world:

Some part of the real world about which data is stored in a database.

The mini-world chosen here is the Uber taxi management system data storage in form of database for design and implementation.

2. Write down the data requirements and functional requirements for the database (in approximately 1500 words).

The data requirements, apart from data to be stored in the database should also take into account the necessary integrity constraints that are reasonable for the database under consideration.

The functional requirements should involve at least two different scenarios of removal of old data, at least two different scenarios for modification of existing data and four different scenarios of data retrieval.

Data Requirement for Uber Taxi management system

1) Tax

Taxi_no

Data stored:

• Taxi_no of each taxi

Constraint:

• Taxi_no is primary key and hence cannot be null.
• No two taxi can have same number.
• taxi id can only consist of fixed number of number(say 10)

b) Name

Data stored:

• Name of taxi
•Value is constrained to be of datatype varchar

c) Model Name:
Data stored:
• Model of taxi
•Value is constrained to be of datatype varchar

d) License_No

• License_No of taxi
•Value is constrained to be of datatype number

e) Storage
Data stored
• availablity of storage(yes/no)
•Value is constrained to be of datatype varchar

f) Manufacturer
Data stored
• Manufacturing Company
•Value is constrained to be of datatype varchar

g)Type
Data stored
• Type of taxi(premium/unlimited)
•Value is constrained to be of datatype varchar

h) No_of_seat
Data stored:
•Number of seats available
•Value is constrained to be of datatype number

2) Passenger Data stored:

• Passenger’s id
• Passenger’s Name
• Passenger’s Age
• Reservation status

Constraints:

- The passenger table should have unique id for each tuple.
- Tabular form is conventionally chosen.
- Passenger id cannot be NULL value.
- Passenger id can take only value having datatype as number and that too limited assigned value only. Passenger name can take only value having datatype as varchar and that too limited assigned value only(ex-varchar(20)).

Passenger age can take only value having datatype as number and that too limited assigned value only (ex-age(3)).

Date of travel can take only value having datatype as number and that too limited assigned value only. Reservation status can take only value having datatype as varchar and that too limited assigned value only(ex-varchar(20)).

3) User Data stored:

- Date of Birth
- rating
- ID
- Password
- Mobile_No.

Constraint

- The user table should have unique id for each tuple.

- Tabular form is conventionally chosen.
- Passenger id cannot be NULL value.
- User id can take only value having datatype as number and that too limited assigned value only. Password can take only value having datatype as varchar and that too limited assigned value only(exvarchar(20)).

Mobile_No. can take only value having datatype as number and that too limited assigned value only (ex-number(10)). Date of birth can take only value having datatype as timestamp and that too limited assigned value only.
Rating can take only value having datatype as number and that too limited assigned value only(exnumber(10))

4) Reservation Data stored:

•Date_of_journey

•Pnr

•Starting_Point

•Destination Point

•Fare

•Type_of_booking

•No_of_seats

Constraints:

• The reservation table should have unique pnr for each tuple.

• Tabular form is conventionally chosen.

• Passenger id cannot be NULL value.

• Pnr,fare can take only value having datatype as number and that too limited assigned value only. Starting_point, type_of_booking and destination_point can take only value having datatype as varchar and that too limited assigned value only(ex-varchar(20)).

No. of seats can take only value having datatype as number and that too limited assigned value only (ex-number(10)).

Date of journey can take only value having datatype as timestamp and that too limited assigned value only.

5) Driver Data stored:

•Driver_id

•Ratings

• Phone_no

Constraints:

• The driver table should have unique driver_id for each tuple.

•Tabular form is conventionally chosen.

• Passenger id cannot be NULL value.

• Driver_id,phone_no,rating can take only value having datatype as number and that too limited assigned value only.

6) Employee Data stored:

• Employee's id

- Employee’s Name
- Employee’s Age(for retirement purpose)
- Department number
- Salary

Constraints:

- The Employee table should have unique id for each tuple.
- Relational model is conventionally chosen.
- Tabular form is conventionally chosen.
- Employee id cannot be NULL value.
- Employee id can take only value having datatype as number and that too limited assigned value only. Employee name can take only value having datatype as varchar and that too limited assigned value only(ex-varchar(20)).

Employee age can take only value having datatype as number and that too limited assigned value only (ex-age(3)). Department number can take only value having datatype as number and that too limited assigned value only.

Salary can take only value having datatype as number and that too limited assigned value only.

- Department number is a foreign key

7) Payment Data stored:

- Taxi Id

- base Fare
- Tax Service
- Other charge
- Account_No

Constraints:

- Taxi id is a foreign key.
- Domain Constraint: base Fare, Tax_service and other charge can take only value having datatype as number and that too limited assigned value only.

8) Tracking Data stored:

- Taxi id
- Geolocation of taxi
- Geolocation of passenger
- Id of driver
- Arrival time
- Depart time
- Distance

Constraint:

- Taxi number is a foreign key.
- Relational model is conventionally chosen.
- Tabular form is conventionally chosen.
- distance can take only value having datatype as number and that too limited assigned value only.

Arrival time, Depart time can only take datatype timestamp. Geolocation of taxi,Geolocation of passenger,Taxi id, day can take only value having datatype as varchar and that too limited assigned value only(ex-varchar(20)).

Functional Requirement for Uber Taxi Management System

1) **Making sure availability of taxi** of almost all type at all time. Taxi management system should make sure that taxis could be made available to customers at all time for smooth business. Availabilty of variety of taxis should also be ensured(3-seater,4-seater,6-seater)

2) Assigning **multiple stops** as per customer's demand Uber added multiple stops facility to facilitate customer's demand of multiple stoppage. For instance: While visiting relatives,customer wish to visit stores to purchase certain items and then visit relatives, Uber now provides this facility to customer.

3) **Online Booking** Of Uber Taxi Taxi provide online booking to passengers and ensures confirmation through messages and email. Ubers do certain steps like:

- •Takes passenger's phone number
- •Takes name
- •Takes email id
- •Takes current location
- •Asks for destination
- •Asks for payment mode followed by payment(if mode supports)

4) **Maintaining Customer's database and driver's database Customer** details are kept in account Driver's details along with driver's photo is kept in account. Rating of drivers and passengers are also taken in account.

5) **Stating the reservation status** of passengers.
Message confirming passenger's travel is sent to passengers Messages like "Your Uber is on the way. Mohan(taxi Driver name) 4.75 stars(rating as per customer to driver) will arrive in 5 minutes" is sent once a travel is confirmed.

6) **Updating taxi location** every 5 sec Once a travel is booked , the taxi management system make sure that passenger's get the live location of taxi every 5 sec. This facility is continued till the travel is completed. Travel must be cancelled if passengers requests cancellation. The reservation status of passenger should also be updated as per cancellation or confirm booking.

7) **Assigning work** to each department and employee. The Uber Management System should make sure that employees like technical staffs, engineers and managers should be appointed with a proper verification so that they could be helpful in providing facilities in a

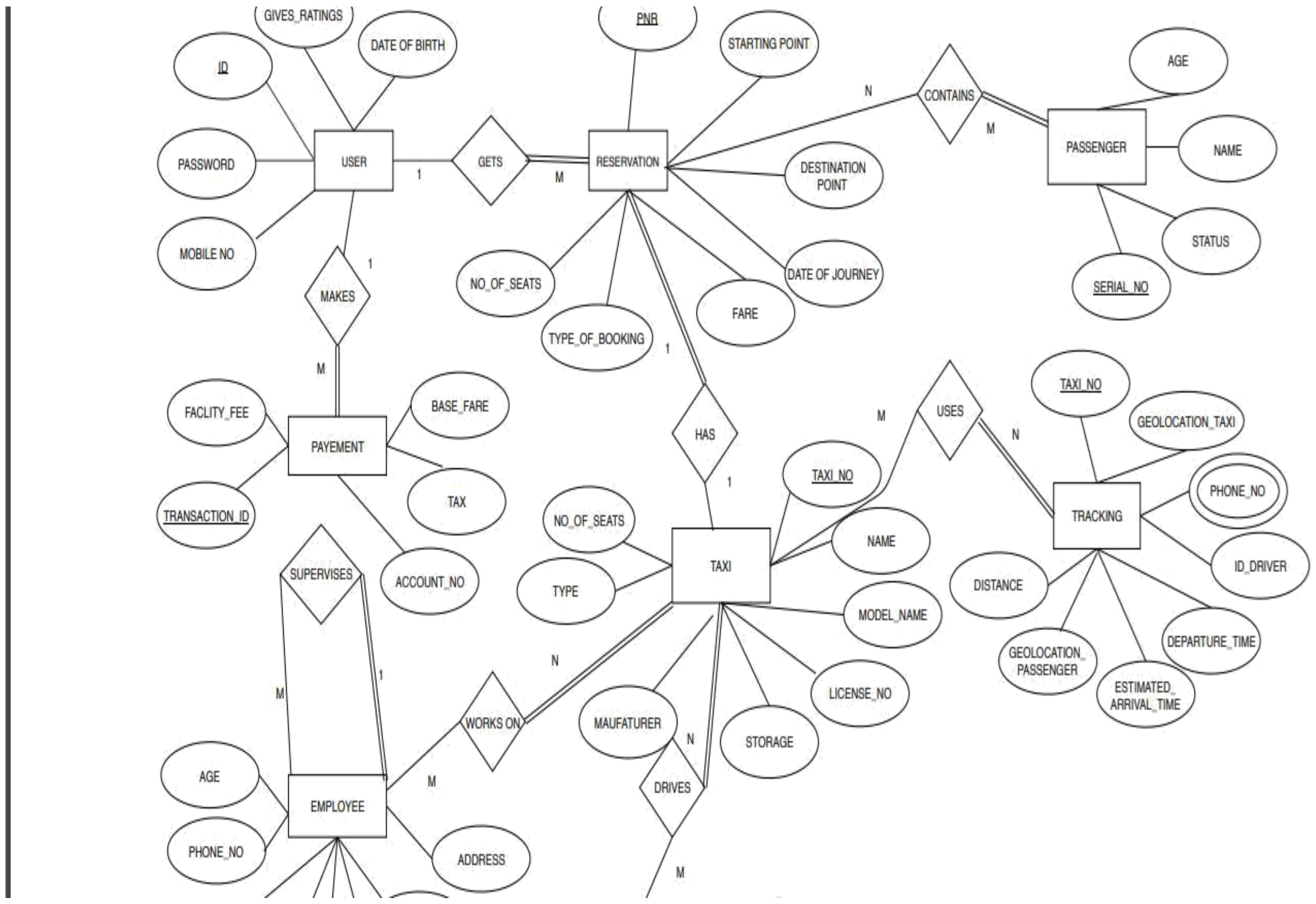
desired manner. Employees at ground level like that for sanitation and cleaniness must also be appointed in a different manner.

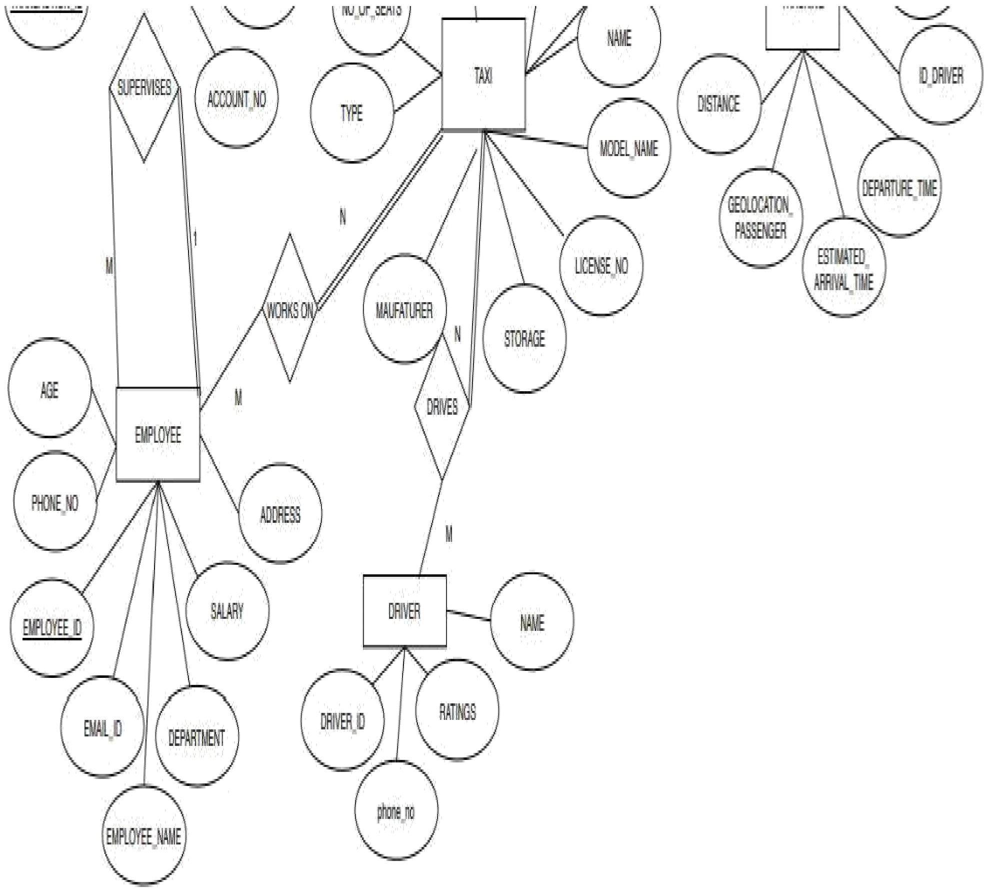
8) **Good Communication System** Uber's Customer Care number helps customers 24*7 . Hence a good communication support system is required.

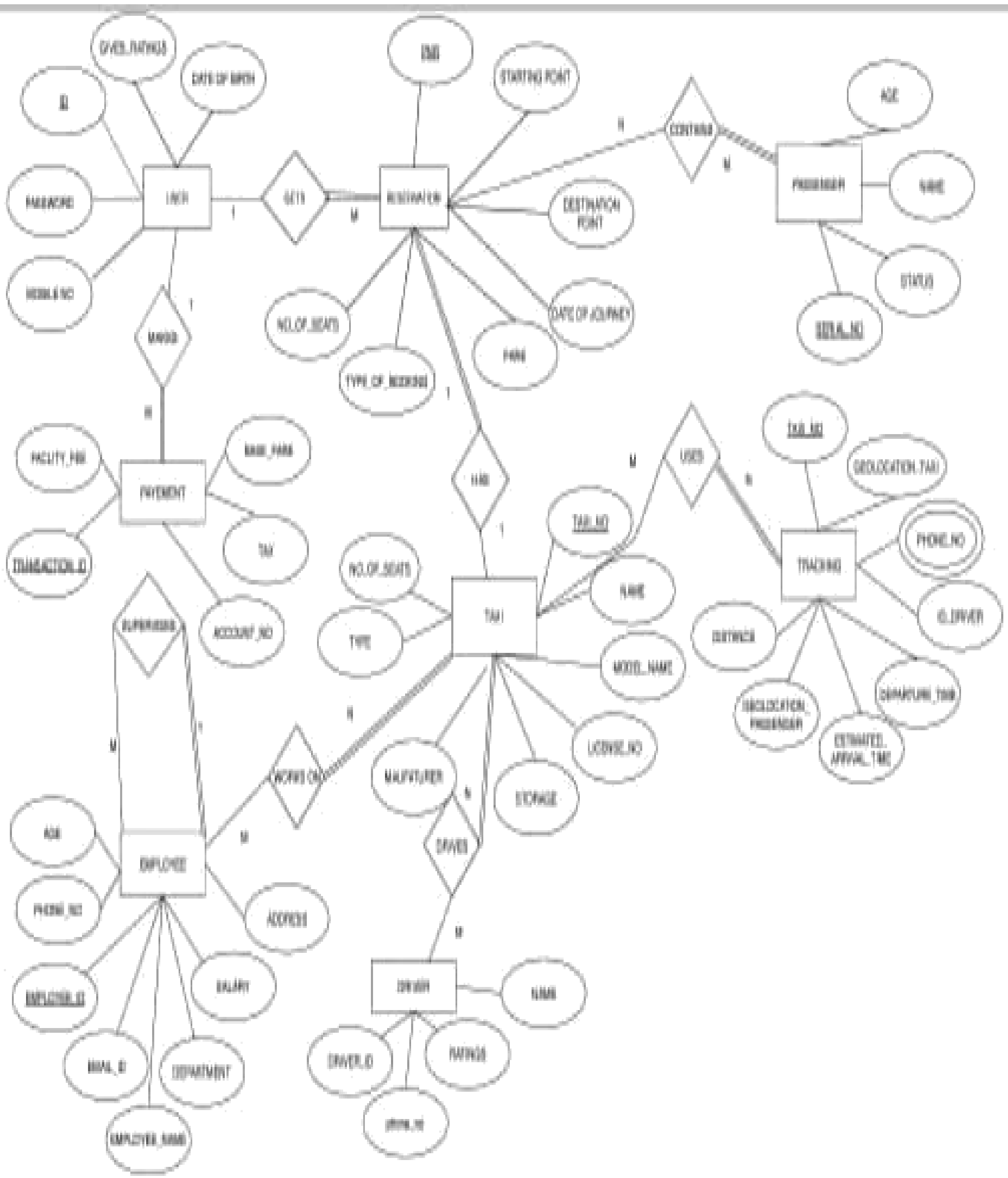
9) **Deciding taxi's fare** Travel cost per kilometer is to be decided by management system in such a way that both customer and company runs in profit and smoothly.

10)**Taking payments** and ensuring traveller's security Taking payments and ensuring traveller's safety and security of money and individual.

3.Draw an ER/EER diagram based on the data requirements. Indicate key constraints, cardinality constraints and participation constraints on diagram.

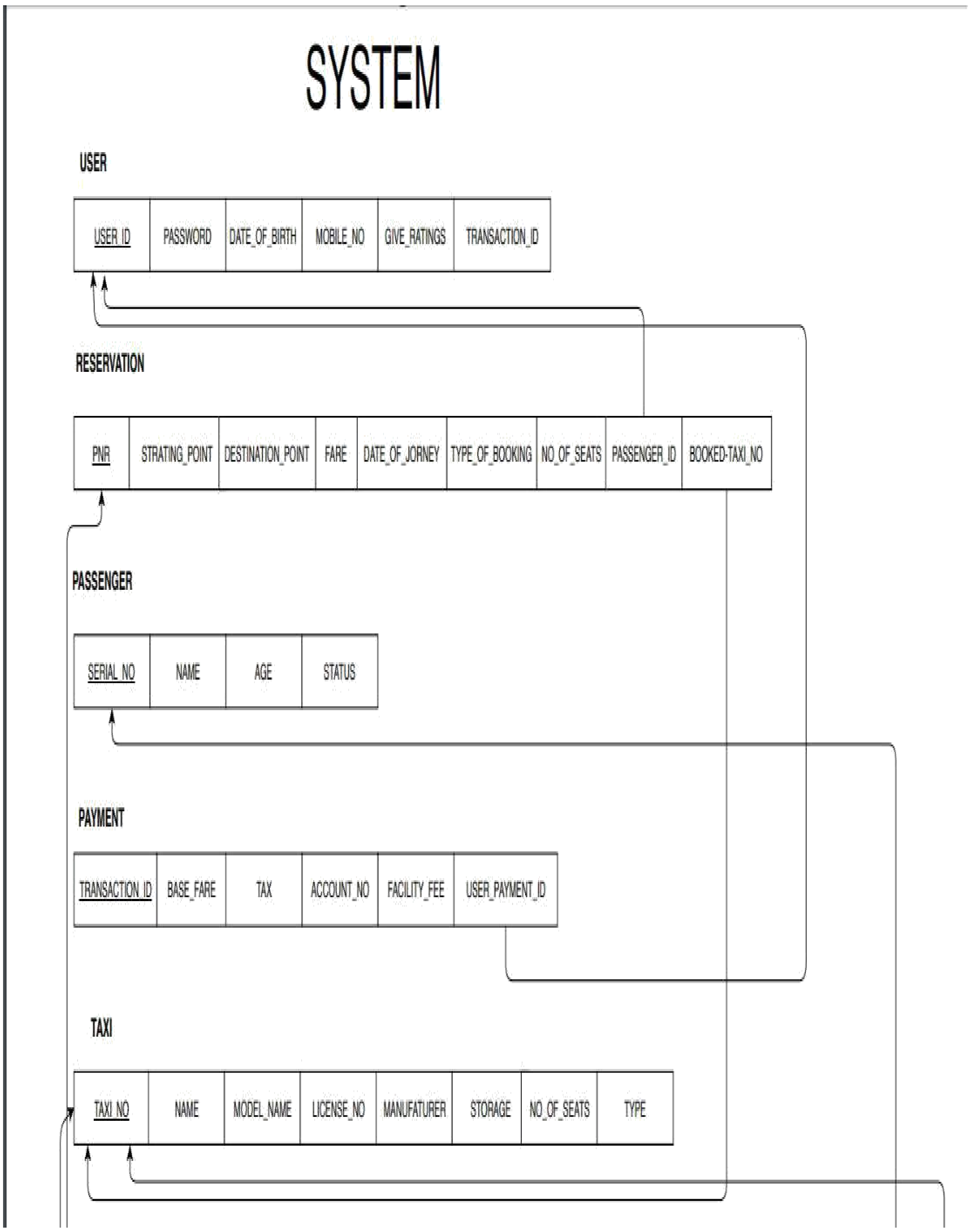


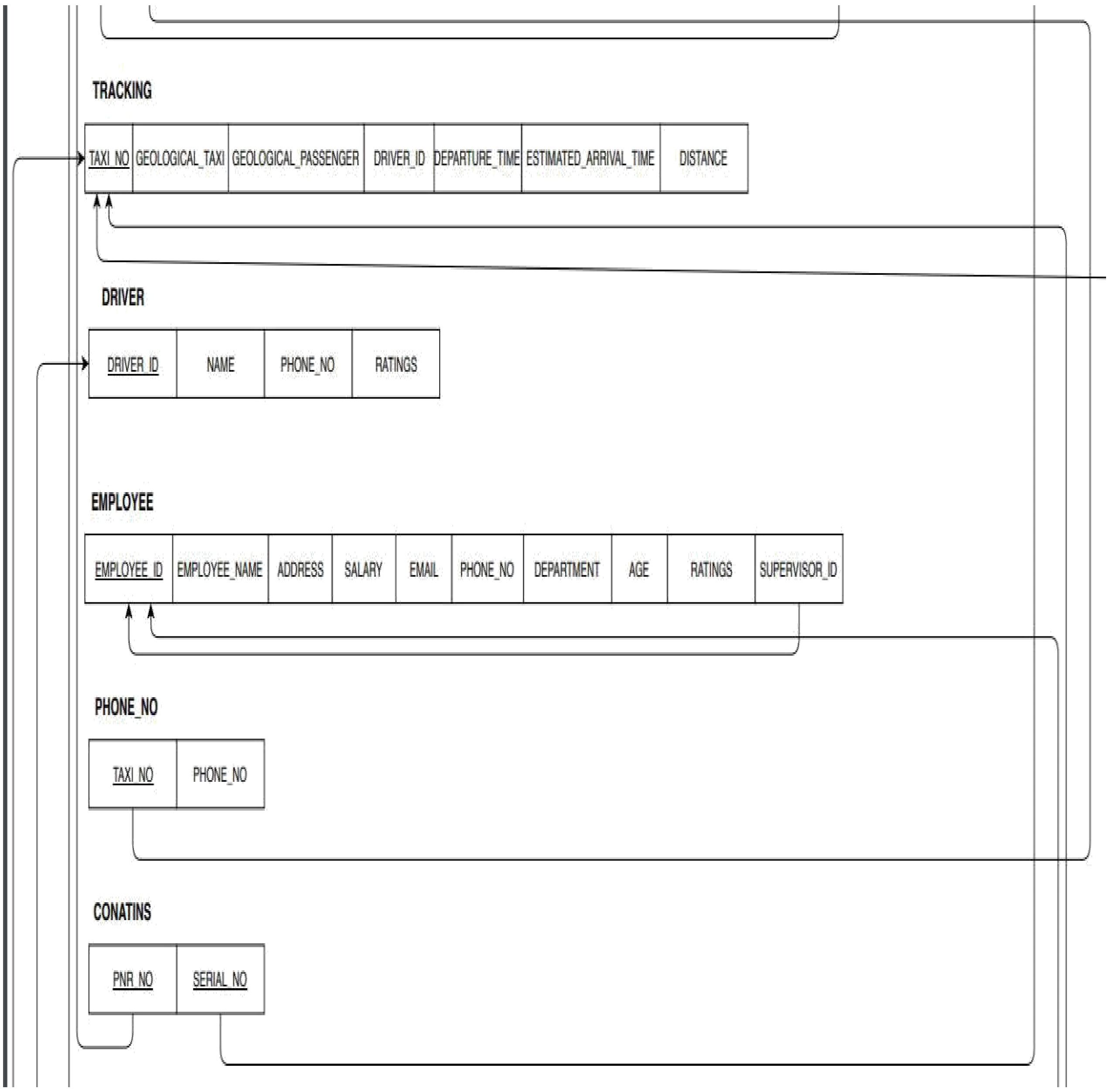

NO_OF_SEATS
TAXI
NAME
SUPERVISES
ACCOUNT_NO
TYPE
DISTANCE
ID_DRIVER
MODEL_NAME
DEPARTURE_TIME
GEOLOCATION_PASSENGER
ESTIMATED_ARRIVAL_TIME
N
1
M
LICENSE_NO
WORKS ON
MAUFATURER
STORAGE
N
AGE
M
DRIVES
EMPLOYEE
PHONE_NO
ADDRESS
M
SALARY
DRIVER
NAME
EMPLOYEE_ID
RATINGS
DRIVER_ID
EMAIL_ID
DEPARTMENT
phone_no
EMPLOYEE_NAME

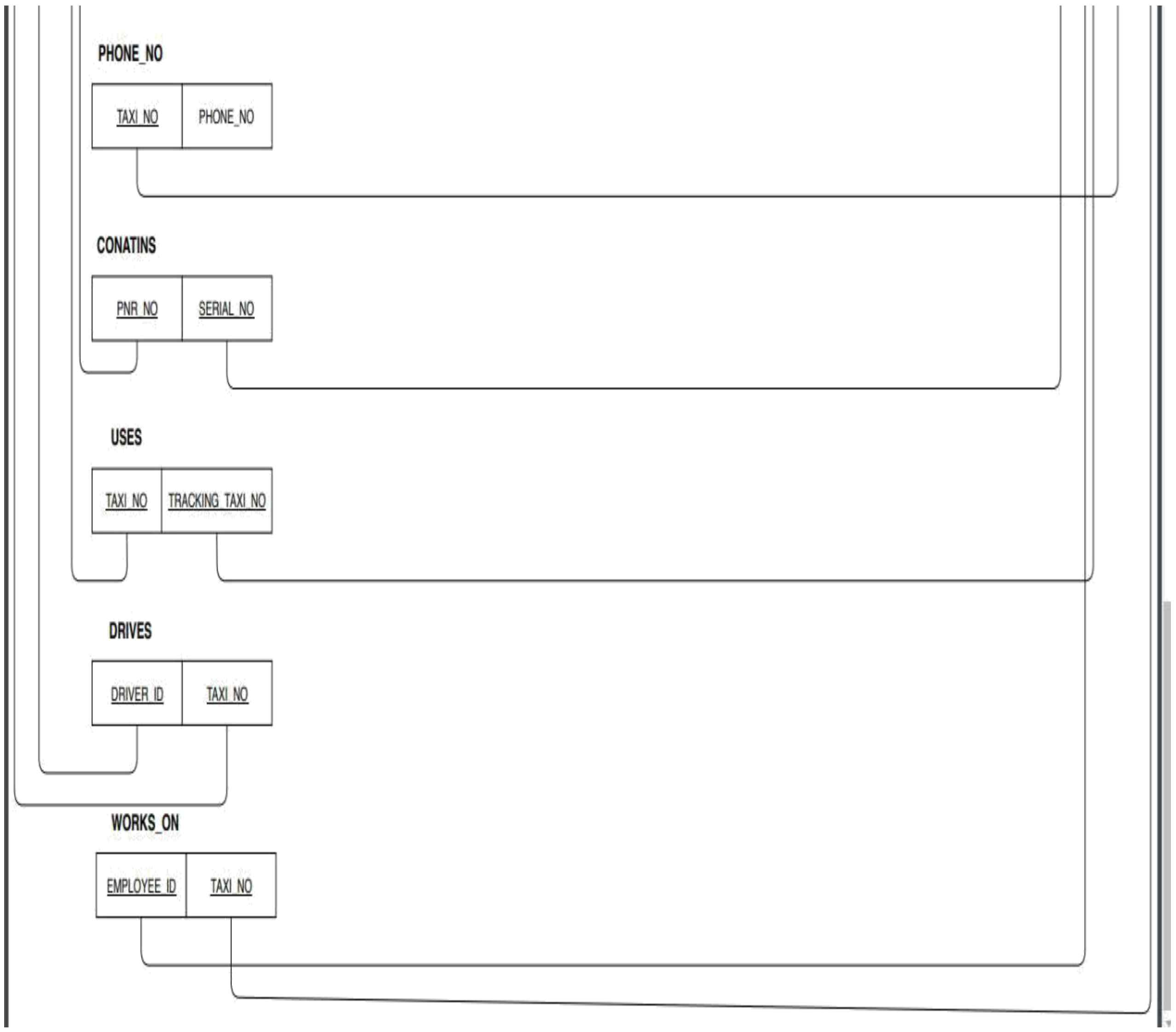


Question 4:

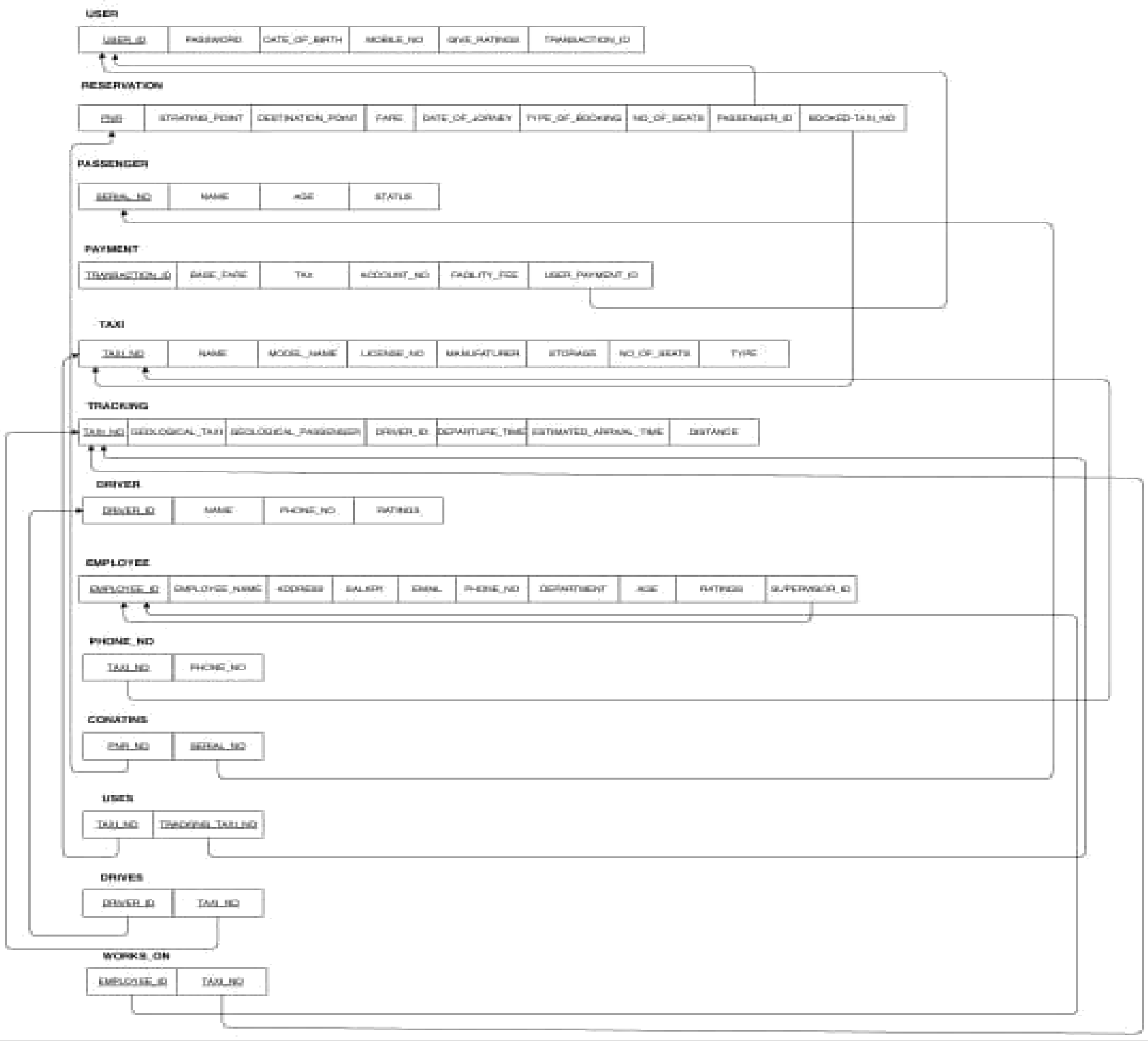
4. Convert the ER/EER diagram into a relational database schema diagram.


SYSTEM
USER
USER_ID
PASSWORD
DATE_OF_BIRTH
MOBILE_NO
GIVE_RATINGS
TRANSACTION_ID
RESERVATION
PNR
STRATING_POINT
DESTINATION_POINT
FARE
DATE_OF_JORNEY
TYPE_OF_BOOKING
NO_OF_SEATS
PASSENGER_ID
BOOKED-TAXI_NO
PASSENGER
SERIAL_NO
NAME
AGE
STATUS
PAYMENT
TRANSACTION_ID
BASE_FARE
TAX
ACCOUNT_NO
FACILITY_FEE
USER_PAYMENT_ID
TAXI
TAXI_NO
NAME
MODEL_NAME
LICENSE_NO
MANUFATURER
STORAGE
NO_OF_SEATS
TYPE


TRACKING
TAXI_NO
GEOLOGICAL_TAXI
GEOLOGICAL_PASSENGER
DRIVER_ID
DEPARTURE_TIME
ESTIMATED_ARRIVAL_TIME
DISTANCE
DRIVER
DRIVER_ID
NAME
PHONE_NO
RATINGS
EMPLOYEE
EMPLOYEE_ID
EMPLOYEE_NAME
ADDRESS
SALARY
EMAIL
PHONE_NO
DEPARTMENT
AGE
RATINGS
SUPERVISOR_ID
PHONE_NO
TAXI_NO
PHONE_NO
CONATINS
PNR_NO
SERIAL_NO


PHONE_NO
TAXI_NO
PHONE_NO
CONATINS
PNR_NO
SERIAL_NO
USES
TAXI_NO
TRACKING_TAXI_NO
DRIVES
DRIVER_ID
TAXI_NO
WORKS_ON
EMPLOYEE_ID
TAXI_NO

TAXI MANAGEMENT SYSTEM



5)

Implement the relational database schema incorporating appropriate (based on data requirements) integrity constraints and enter necessary sample data into the tables.

1)User table

create table user (user_id number(8) primary key,password varchar(8),date_of_birth date,mobile_no number(10),gives_rating number(2),transaction_id number(8));

desc user;

Name	Null?	Type
USER_ID	NOT NULL	NUMBER(8)
PASSWORD		VARCHAR2(8)
DATE_OF_BIRTH		DATE
MOBILE_NO		NUMBER(10)
GIVES_RATING		NUMBER(2)
TRANSACTION_ID		NUMBER(8)

Insertion:

insert into user values(1234,4321,to_date('25-11-01','dd-mm-yy'),9967543214,3,1457);

USER_ID	PASSWORD	DATE_OF_B	MOBILE_NO	GIVES_RATING	TRANSACTION_ID
1234	4321	25-NOV-01	9967543214	3	1457
1355	1233	18-NOV-13	9865876511	5	1238

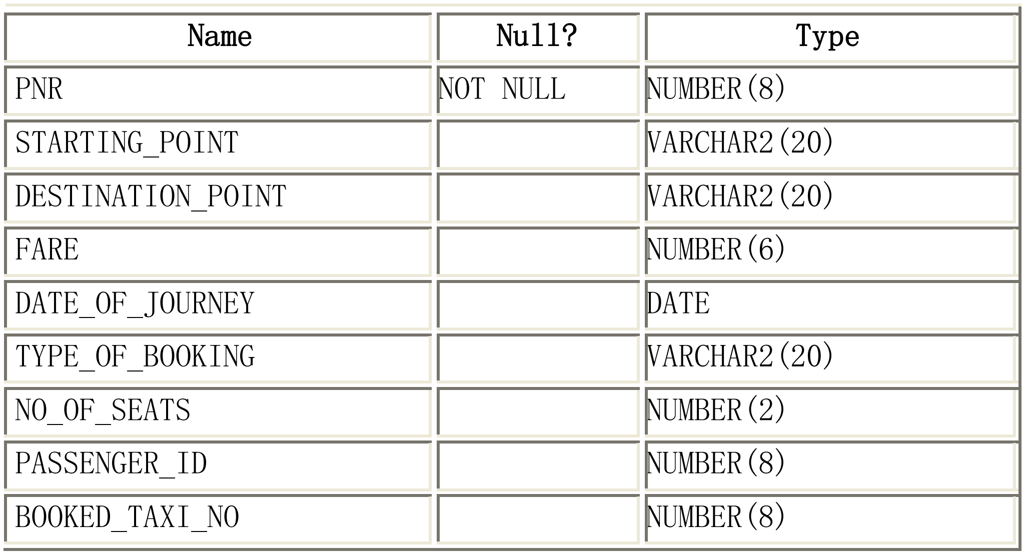
1) Reservation table

```
create table reservation(pnr number(8) primary key, starting_point
varchar(20),destination_point varchar(20), fare
number(6),date_of_journey date,type_of_booking
varchar(20),no_of_seats number(2));
```

```
desc reservation;
```

Later, after making taxi table, booked_taxi no is inserted by

```
ALTER TABLE reservation
ADD booked_taxi_no references taxi;
```



Name	Null?	Type
PNR	NOT NULL	NUMBER(8)
STARTING_POINT		VARCHAR2(20)
DESTINATION_POINT		VARCHAR2(20)
FARE		NUMBER(6)
DATE_OF_JOURNEY		DATE
TYPE_OF_BOOKING		VARCHAR2(20)
NO_OF_SEATS		NUMBER(2)
PASSENGER_ID		NUMBER(8)
BOOKED_TAXI_NO		NUMBER(8)

Insertion:

//Done after taxi insertion

```
insert into reservation
values(1789,"katpadi","Malaikodi",1800,to_date('13-08-19','dd-mm-
yy'), "Uber Premium",4,1234,4567);
```

```
select * from reservation;
```

PNR	STARTING_POINT	DESTINATION_POINT	FARE	DATE_OF_JOURNEY	TYPE_OF_BOOKING	NO_OF_SEATS	PASSENGER_ID	BOOKED_TAXI_NO
1789	katpadi	Malaikodi	1800	13-AUG-19	Uber Premium	4	1234	4567
1456	katpadi	kathi	1599	15-AUG-19	Uber Unlimited	6	1355	1453

2) Passenger table

create table passenger(serial_no number(8) primary key,name varchar(25),age number(3),status varchar(12));

desc passenger;

Name	Null?	Type
SERIAL_NO	NOT NULL	NUMBER(8)
NAME		VARCHAR2(25)
AGE		NUMBER(3)
STATUS		VARCHAR2(12)

Insertion:

insert into passenger values(2345,'joy',18,'confirm');

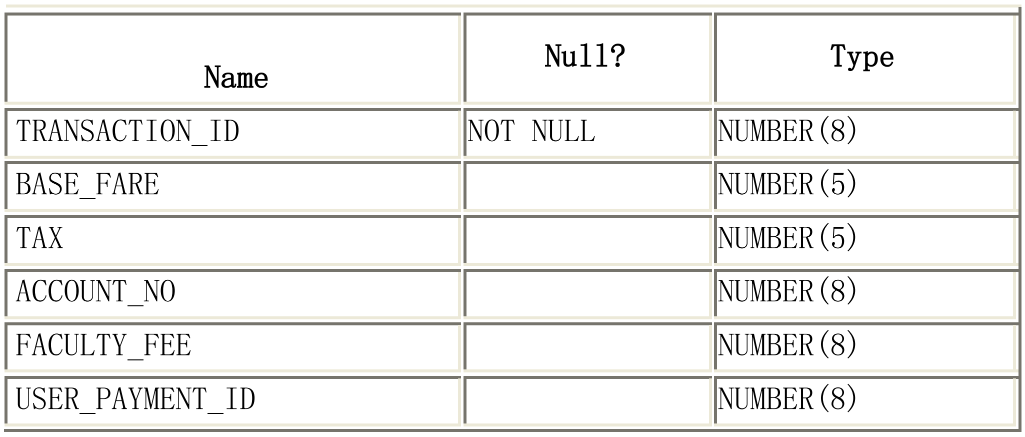
select * from passenger;

SERIAL_NO	NAME	AGE	STATUS
2345	joy	18	confirm
1111	jesika	20	notConfirm

3) Payment table

create table payment(transaction_id number(8) primary key,base_fare number(5),tax number(5),account_no number(8),faculty_fee number(8),user_payment_id references user);

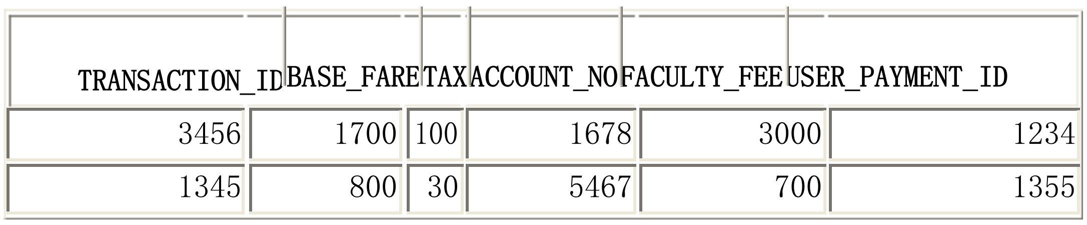
desc payment;



Name	Null?	Type
TRANSACTION_ID	NOT NULL	NUMBER(8)
BASE_FARE		NUMBER(5)
TAX		NUMBER(5)
ACCOUNT_NO		NUMBER(8)
FACULTY_FEE		NUMBER(8)
USER_PAYMENT_ID		NUMBER(8)

Insertion:

insert into payment values(3456,1700,100,1678,3000,1234);
select * from payment;



TRANSACTION_ID	BASE_FARE	TAX	ACCOUNT_NO	FACULTY_FEE	USER_PAYMENT_ID
3456	1700	100	1678	3000	1234
1345	800	30	5467	700	1355

4) Taxi Table

create table taxi(taxi_no number(8) primary key, name varchar(20),model_name varchar(20),license_no number(8),manufacturer varchar(20),storage varchar(20),no_of_seats number(8),type varchar(8));

desc taxi;

Name	Null?	Type
TAXI_NO	NOT NULL	NUMBER(8)
NAME		VARCHAR2(20)
MODEL_NAME		VARCHAR2(20)
LICENSE_NO		NUMBER(8)
MANUFACTURER		VARCHAR2(20)
STORAGE		VARCHAR2(20)
NO_OF_SEATS		NUMBER(8)
TYPE		VARCHAR2(8)

Insertion:

insert into taxi values(4567,'hackney carriage','TX4',1897,'Worship Company','Yes',4,'premium');

select * from taxi;

TAXI_NO	NAME	MODEL_NAME	LICENSE_NO	MANUFACTURER	STORAGE	NO_OF_SEATS	TYPE

4567	hackney carriage	TX4	1897	Worship Company	Yes	4	premium
1453	hackney carriage	TX3	1456	Worship Company	No	4	premium

5) Tracking Table

create table tracking(taxi_no number(8) primary key, geological_taxi varchar(20),geological_passenger varchar(20),driver_id number(9),departure_time timestamp(0),estimated_arrival_time timestamp(0),distance number(4));

desc tracking;

Name	Null?	Type
TAXI_NO	NOT NULL	NUMBER(8)
GEOLOGICAL_TAXI		VARCHAR2(20)
GEOLOGICAL_PASSENGER		VARCHAR2(20)
DRIVER_ID		NUMBER(9)
DEPARTURE_TIME		TIMESTAMP(0)
ESTIMATED_ARRIVAL_TIME		TIMESTAMP(0)
DISTANCE		NUMBER(4)

Insertion:

insert into tracking values(4567,'Nellore','katpadi',6666,to_timestamp('07:35','hh24:mi'),to_timestamp('07:37','hh24:mi'),160);

select * from tracking;

TAXI	GEOLOGICAL_TAXI	GEOLOGICAL_PASSENGER	DRIVER_ID	DEPARTURE_TIME	ESTIMATED_ARRIVAL_TIME	DISTANCE

_NO						
1001	Nellore	katpadi	6666	01-OCT-19 07.35.00 AM	01-OCT-19 07.37.00 AM	160
1002	koimbatore	nellore	1456	01-OCT-19 07.35.00 AM	01-OCT-19 07.39.00 AM	200

6) Driver table

create table driver(driver_id number(8) primary key, name varchar(20),phone_no number(10),ratings number(2));

desc driver;

Name	Null?	Type
DRIVER_ID	NOT NULL	NUMBER(8)
NAME		VARCHAR2(20)
PHONE_NO		NUMBER(10)
RATINGS		NUMBER(2)

Insertion:

insert into driver values(6666,'raghu',9934564122,5);

select * from driver;

DRIVER_ID	NAME	PHONE_NO	RATINGS
6666	raghu	9934564122	5
7777	rani	9934509122	4

7) Employee Table

create table employee(employee_id number(8) primary key,employee_name varchar(20),address varchar(20),salary number(7),email varchar(40),phone_no number(10),department varchar(20),age number(3),ratings number(2),supervisor_id references employee);

desc employee;

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(8)
EMPLOYEE_NAME		VARCHAR2(20)
ADDRESS		VARCHAR2(20)
SALARY		NUMBER(7)
EMAIL		VARCHAR2(40)
PHONE_NO		NUMBER(10)
DEPARTMENT		VARCHAR2(20)
AGE		NUMBER(3)
RATINGS		NUMBER(2)
SUPERVISOR_ID		NUMBER(8)

Insertion:

insert into employee values(8765,'mitra','Dhanushkodi',10000,'mitra@gmail.com',9876785432,'taxi_service',29,5,8765);

select * from employee;

EMPLOYEE_ID	EMPLOYEE_NAME	ADDRESS	SALARY	EMAIL	PHONE_NO	DEPARTMENT	AGE	RATINGS	SUPERVISOR_ID

8765	mitra	Dhanushkodi	10000	mitra@gmail.com	9876785432	taxi_service	29	5	8765
9089	maya	nellore	9999	maya@gmail.com	9076785632	taxi_service	36	5	9089

8) Phone_no table

create table phone_no(taxi_no references taxi not null,phone_no number(10));

desc phone_no;

Name	Null?	Type
TAXI_NO	NOT NULL	NUMBER(8)
PHONE_NO		NUMBER(10)

Insertion:

insert into phone_no values(4567,9876985345);

select * from phone_no;

TAXI_NO	PHONE_NO
4567	9876985345
1453	9934267853

9) Contains table

create table contains(pnr_no references reservation not null, serial_no references passenger not null);

desc contains;

Name	Null?	Type
PNR_NO	NOT NULL	NUMBER(8)
SERIAL_NO	NOT NULL	NUMBER(8)

Insertion:

insert into contains values(1789,2345);

select * from contains;

PNR_NO	SERIAL_NO
1789	2345
1456	1111

10) Uses table

create table uses(taxi_no references taxi not null,tracking_taxi_no references tracking not null);

desc uses;

Name	Null?	Type
TAXI_NO	NOT NULL	NUMBER(8)
TRACKING_TAXI_NO	NOT NULL	NUMBER(8)

Insertion:

insert into uses values(4567,1001);

select * from uses;

TAXI_NO	TRACKING_TAXI_NO
4567	1001
1453	1002

11) Drives Table

create table drives(driver_id references driver not null, taxi_no references tracking not null);

desc drives;

Name	Null?	Type
DRIVER_ID	NOT NULL	NUMBER(8)
TAXI_NO	NOT NULL	NUMBER(8)

Insertion:

insert into drives values(6666,1001);

select * from drives;

DRIVER_ID	TAXI_NO
6666	1001
7777	1002

12) Works_on Table

```
create table works_on(employee_id references employee
not null,taxi_no references tracking not null);
```

```
desc works_on;
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(8)
TAXI_NO	NOT NULL	NUMBER(8)

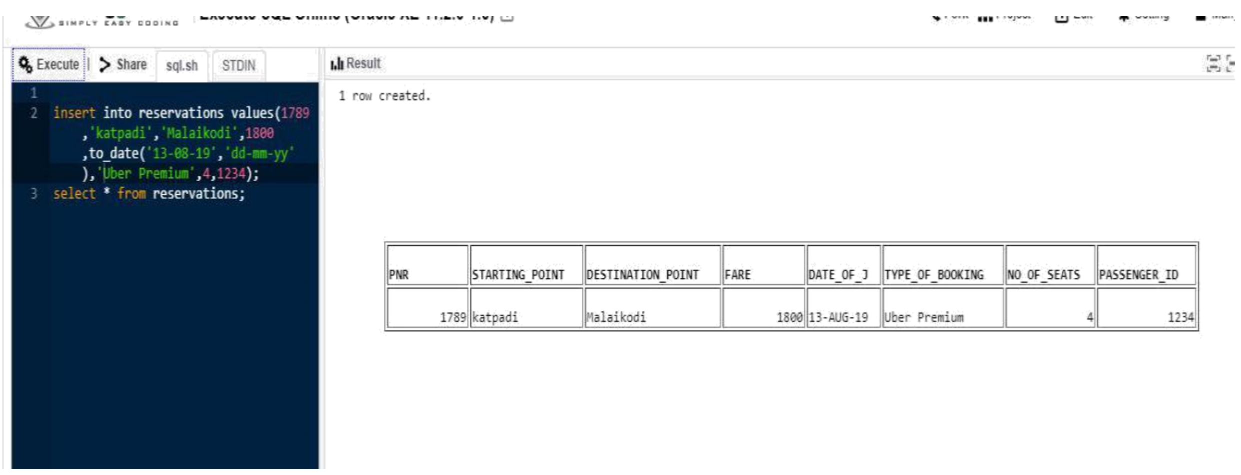
Insertion:

```
insert into works_on values(8765,1001);
```

```
select * from works_on;
```

EMPLOYEE_ID	TAXI_NO
8765	1001
9089	1002

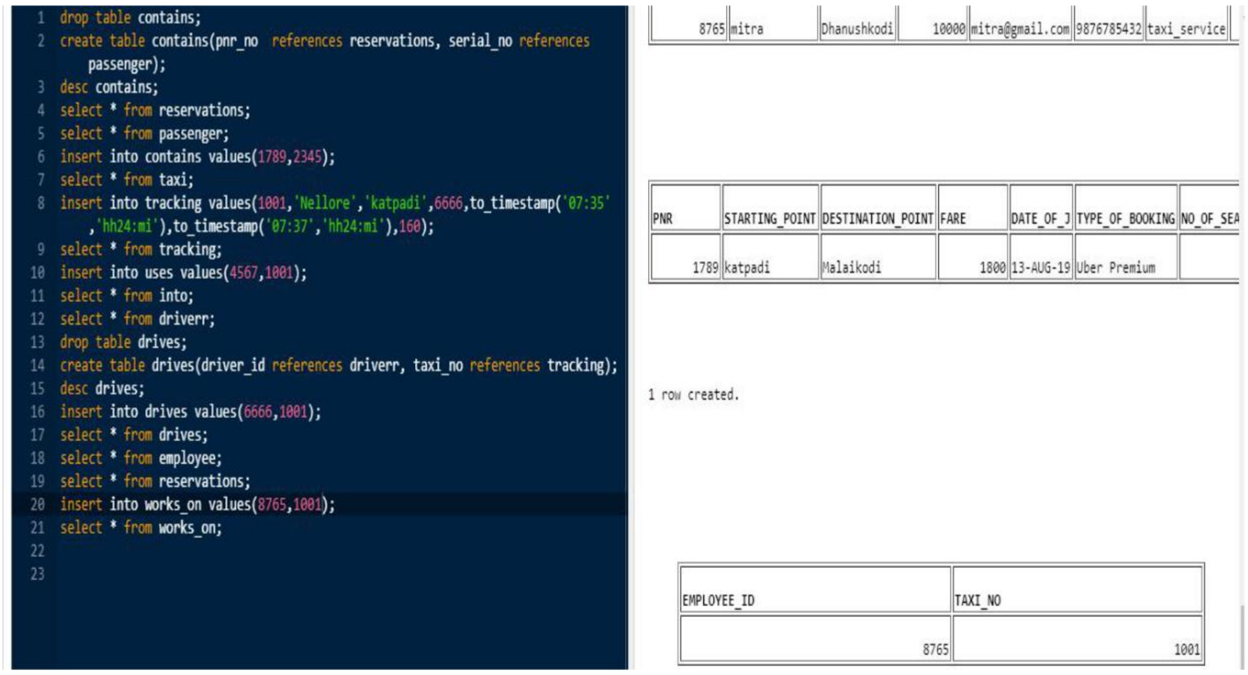
Some screens



```
insert into reservations values(1789
    ,'katpadi','Malaikodi',1800
    ,to_date('13-08-19','dd-mm-yy'
    ),'Uber Premium',4,1234);
select * from reservations;
```

```
1 row created.
```

PNR	STARTING_POINT	DESTINATION_POINT	FARE	DATE_OF_J	TYPE_OF_BOOKING	NO_OF_SEATS	PASSENGER_ID
1789	katpadi	Malaikodi	1800	13-AUG-19	Uber Premium	4	1234



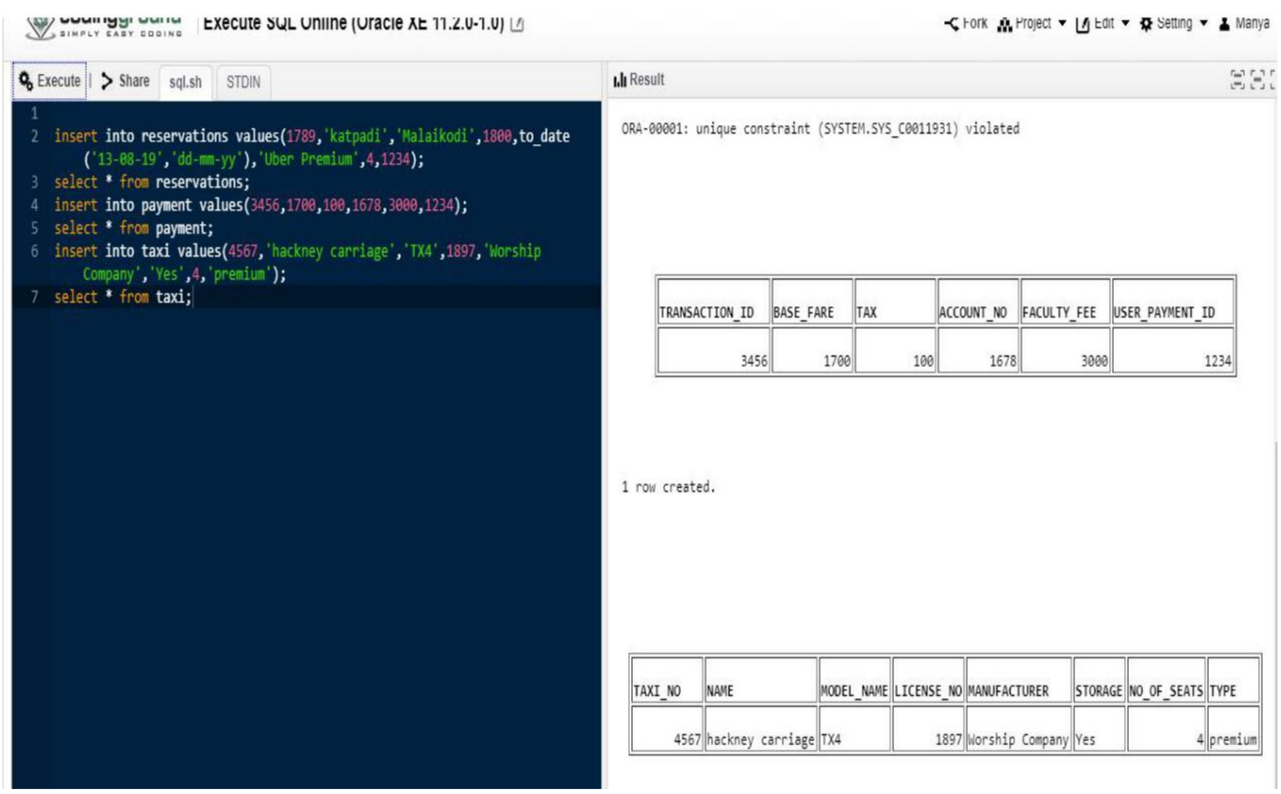
```sql
drop table contains;
create table contains(pnr_no  references reservations, serial_no references
    passenger);
desc contains;
select * from reservations;
select * from passenger;
insert into contains values(1789,2345);
select * from taxi;
insert into tracking values(1001,'Nellore','katpadi',6666,to_timestamp('07:35'
    ,'hh24:mi'),to_timestamp('07:37','hh24:mi'),160);
select * from tracking;
insert into uses values(4567,1001);
select * from into;
select * from driverr;
drop table drives;
create table drives(driver_id references driverr, taxi_no references tracking);
desc drives;
insert into drives values(6666,1001);
select * from drives;
select * from employee;
select * from reservations;
insert into works_on values(8765,1001);
select * from works_on;
```

8765	mitra	Dhanushkodi	10000	mitra@gmail.com	9876785432	taxi_service

PNR	STARTING_POINT	DESTINATION_POINT	FARE	DATE_OF_J	TYPE_OF_BOOKING	NO_OF_SEA
1789	katpadi	Malaikodi	1800	13-AUG-19	Uber Premium	

```
1 row created.
```

EMPLOYEE_ID	TAXI_NO
8765	1001



```sql

insert into reservations values(1789,'katpadi','Malaikodi',1800,to_date
    ('13-08-19','dd-mm-yy'),'Uber Premium',4,1234);
select * from reservations;
insert into payment values(3456,1700,100,1678,3000,1234);
select * from payment;
insert into taxi values(4567,'hackney carriage','TX4',1897,'Worship
    Company','Yes',4,'premium');
select * from taxi;
```

```
ORA-00001: unique constraint (SYSTEM.SYS_C0011931) violated
```

TRANSACTION_ID	BASE_FARE	TAX	ACCOUNT_NO	FACULTY_FEE	USER_PAYMENT_ID
3456	1700	100	1678	3000	1234

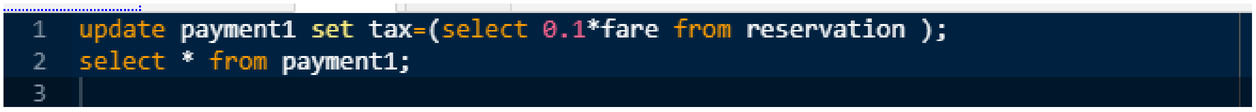
```
1 row created.
```

TAXI_NO	NAME	MODEL_NAME	LICENSE_NO	MANUFACTURER	STORAGE	NO_OF_SEATS	TYPE
4567	hackney carriage	TX4	1897	Worship Company	Yes	4	premium

6. Write down the necessary SQL statements for implementation of functional requirements through SQL query, delete and update statement.

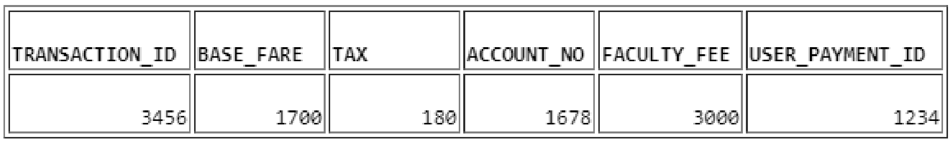
UPDATE COMMANDS

1) Update the tax on the ride and set it as 10% of fare.



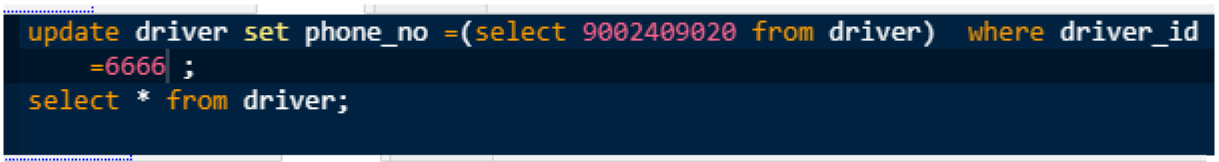
```
update payment1 set tax=(select 0.1*fare from reservation );
select * from payment1;
```

```
1 row updated.
```

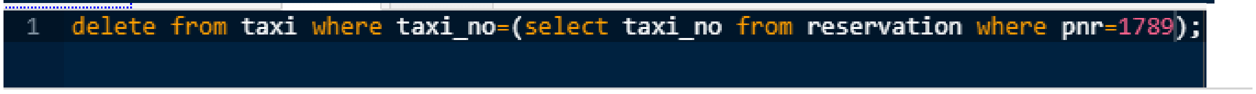


TRANSACTION_ID	BASE_FARE	TAX	ACCOUNT_NO	FACULTY_FEE	USER_PAYMENT_ID
3456	1700	180	1678	3000	1234

2)Update the phone no of a driver.

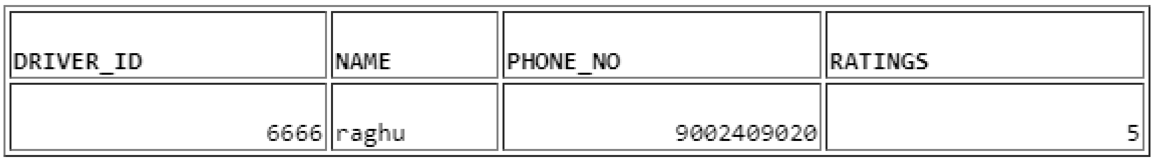


```
update driver set phone_no =(select 9002409020 from driver)  where driver_id
    =6666 ;
select * from driver;
```



```
delete from taxi where taxi_no=(select taxi_no from reservation where pnr=1789);
```

```
1 row updated.
```



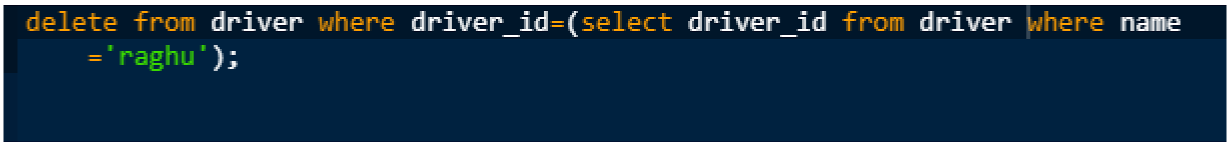
DRIVER_ID	NAME	PHONE_NO	RATINGS
6666	raghu	9002409020	5

Delete commands

1)Delete the taxi details of taxi that got into accident during a ride.

```
1 row deleted.
```

2) Delete the record of driver who has left the company.

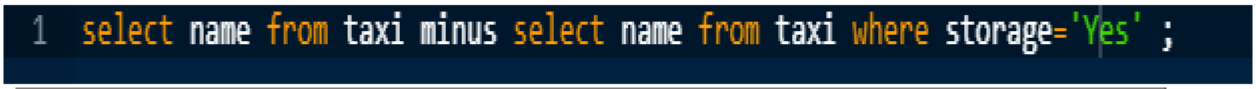


```
delete from driver where driver_id=(select driver_id from driver where name
    ='raghu');
```

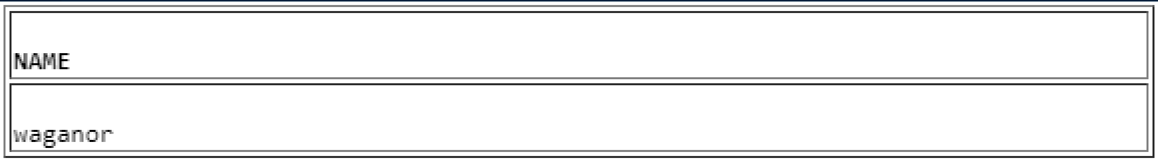
```
1 row deleted.
```

Selection commands

1)Find the taxi names that do not have storage .

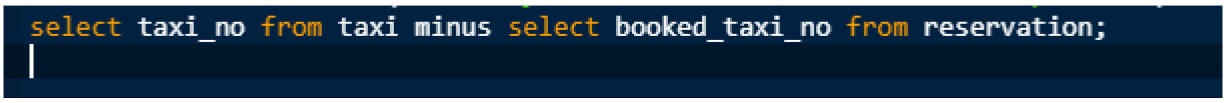


```
select name from taxi minus select name from taxi where storage='Yes' ;
```

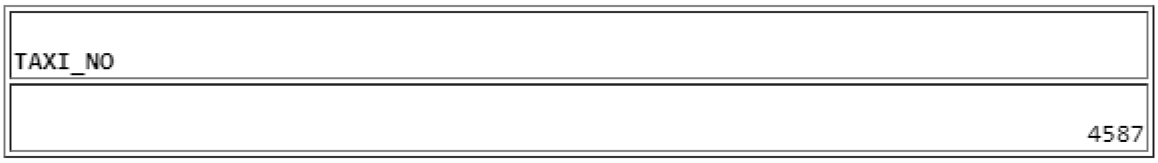


```
NAME
waganor
```

2)Retrieve the taxi number of taxis which have not been reserved.

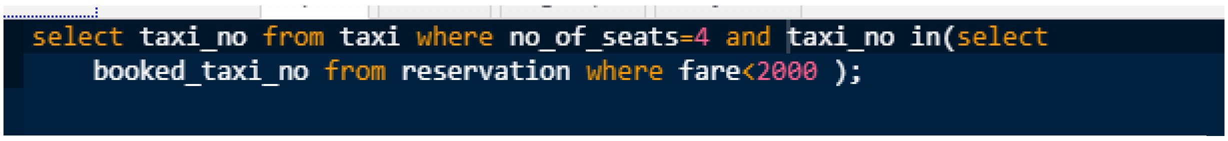


```
select taxi_no from taxi minus select booked_taxi_no from reservation;
```

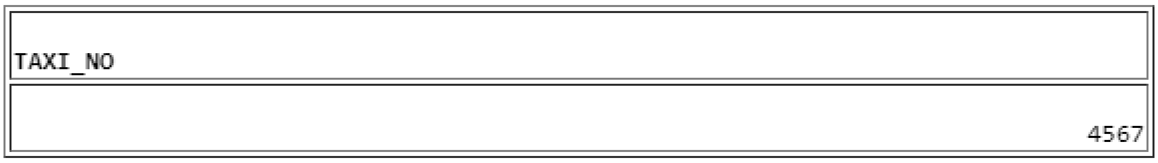


```
TAXI_NO
                                                                      4587
```

2) Find the taxi names that are 4 seaters and the base fare is less than 2000 for each case.

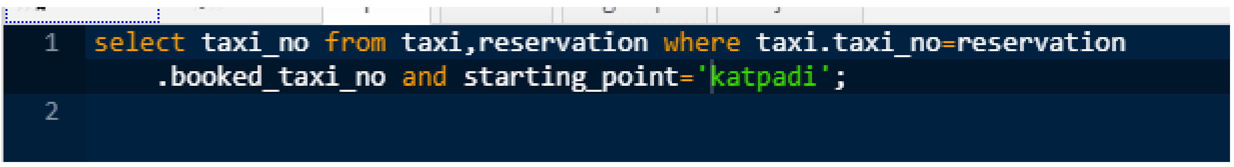


```
select taxi_no from taxi where no_of_seats=4 and taxi_no in(select
    booked_taxi_no from reservation where fare<2000 );
```

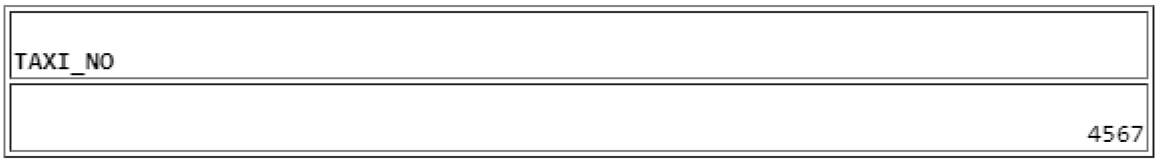


```
TAXI_NO
                                                                      4567
```

4)Retrieve the taxi record of all thoes taxis which are reserved from Katpadi.



```
select taxi_no from taxi,reservation where taxi.taxi_no=reservation
    .booked_taxi_no and starting_point='katpadi';
```



```
TAXI_NO
4567
```

7) Define and implement one PL/SQL function and one PL/SQL procedure appropriate for the database under
consideration

PL/SQL procedure

```
declare
modell varchar(20);
cursor pas_cur is select taxi_no  from taxi  WHERE model='&modell';
pas_rec       pas_cur%rowtype;
begin
open pas_cur;
loop fetch pas_cur into pas_rec;
exit when pas_cur%notfound;
dbms_output.put_line(pas_rec.taxi_no);
end loop;
close pas_cur;
end;
 /
```

PL/SQL function

```
create or replace function emp_age(id number) return number is
agee  employee.age%type;
cursor pas_cur is select age  from employee  WHERE employee_id=id;
pas_rec       pas_cur%rowtype;
begin
open pas_cur;
loop fetch pas_cur into pas_rec;
exit when pas_cur%notfound;
dbms_output.put_line(pas_rec.age);
end loop;
return agee;
close pas_cur;
```

```
end;
 /
```

8. Define two business rules appropriate for the database under consideration and implement the rules using trigger.

```
CREATE TABLE Salgrade (
  Grade              NUMBER,
  Losal              NUMBER,
  Hisal              NUMBER,
  rating_classification    NUMBER)

CREATE OR REPLACE TRIGGER Salary_check
BEFORE INSERT OR UPDATE OF salary,rating ON employee
FOR EACH ROW
DECLARE
  Minsal             NUMBER;
  Maxsal             NUMBER;
  Salary_out_of_range   EXCEPTION;
BEGIN

/* Retrieve the minimum and maximum salary for the
employee's new job classification from the SALGRADE
table into MINSAL and MAXSAL: */

SELECT Minsal, Maxsal INTO Minsal, Maxsal FROM Salgrade
  WHERE Job_classification = :new.rating;


/* If the employee's new rating is less than or greater
than the rating_classification's limits, the exception is
raised.  The exception message is returned and the
pending INSERT or UPDATE statement that fired the
trigger is rolled back:*/

  IF (:new.salary< Minsal OR :new.salary > Maxsal) THEN
    RAISE Salary_out_of_range;
  END IF;
EXCEPTION
  WHEN Salary_out_of_range THEN
    Raise_application_error (-20300,'salary either too high or too low');
  WHEN NO_DATA_FOUND THEN
    Raise_application_error(-20322,
      'Invalid Rating_Classification ');
END;


CREATE TABLE Company_holidays (Day DATE);
CREATE OR REPLACE TRIGGER Emp_permit_changes
BEFORE INSERT OR DELETE OR UPDATE ON Employee
DECLARE
  Dummy           INTEGER;
  Not_on_weekends   EXCEPTION;
  Not_on_holidays   EXCEPTION;
  Non_working_hours EXCEPTION;
BEGIN
/* check for weekends: */
  IF (TO_CHAR(Sysdate, 'DY') = 'SAT' OR
    TO_CHAR(Sysdate, 'DY') = 'SUN') THEN
    RAISE Not_on_weekends;
  END IF;
/* check for company holidays:*/
  SELECT COUNT(*) INTO Dummy FROM Company_holidays
    WHERE TRUNC(Day) = TRUNC(Sysdate);
  /* TRUNC gets rid of time parts of dates: */
  IF dummy > 0 THEN
    RAISE Not_on_holidays;
  END IF;
/* Check for work hours (8am to 6pm): */
  IF (TO_CHAR(Sysdate, 'HH24') < 8 OR
     TO_CHAR(Sysdate, 'HH24') > 18) THEN
     RAISE Non_working_hours;
  END IF;
EXCEPTION
  WHEN Not_on_weekends THEN
    Raise_application_error(-20324,'May not change '
      ||'employee table during the weekend');
  WHEN Not_on_holidays THEN
    Raise_application_error(-20325,'May not change '
      ||'employee table during a holiday');
  WHEN Non_working_hours THEN
    Raise_application_error(-20326,'May not change '
    ||'Emp_tab table during non-working hours');
```

```
END;
```