

**American International University- Bangladesh**

**Spring2019-2020**

**Project Report**

**Advance Database Management System**

**Project Title: Car Shop Management System**

**Section: B**

|  |  |
| --- | --- |
| Student Name | Student Id |
| A.M. Rakibul Hossain | **16-31399-1** |
| Kh. Shakib Ebna Atiq | **18-36914-1** |
| Md. Rofiqul Islam | **17-34588-2** |
| Taohidul hasan | **17-34698-2** |

Contents

[Introduction: 4](#_Toc37433302)

[Project Proposal: 4](#_Toc37433303)

[Class Diagram: 5](#_Toc37433304)

[Use Case Diagram: 6](#_Toc37433305)

[Activity Diagram: 7](#_Toc37433306)

[User Interface: 8](#_Toc37433307)

[Landing: 8](#_Toc37433308)

[Sign Up: 9](#_Toc37433309)

[Customer Sign In: 10](#_Toc37433310)

[Manager Sign In: 11](#_Toc37433311)

[Owner Sign In: 12](#_Toc37433312)

[Customer Profile: 13](#_Toc37433313)

[Manager Profile: 14](#_Toc37433314)

[Owner Profile: 15](#_Toc37433315)

[Customer Dashboard: 16](#_Toc37433316)

[Manager Dashboard: 17](#_Toc37433317)

[Owner Dashboard: 18](#_Toc37433318)

[Add Car: 19](#_Toc37433319)

[Add Customer: 20](#_Toc37433320)

[Add Manager: 21](#_Toc37433321)

[See My Car: 22](#_Toc37433322)

[Manage Car: 23](#_Toc37433323)

[ER Diagram: 24](#_Toc37433324)

[Normalization 25](#_Toc37433325)

[Schema Diagram: 27](#_Toc37433326)

[Table Creation: 28](#_Toc37433327)

[Owner: 28](#_Toc37433328)

[Manager: 29](#_Toc37433329)

[Customer: 30](#_Toc37433330)

[Car: 31](#_Toc37433331)

[Create User: 32](#_Toc37433332)

[Owner: 32](#_Toc37433333)

[Manger: 33](#_Toc37433334)

[Customer: 33](#_Toc37433335)

[INSERTION 34](#_Toc37433336)

[Owner table 34](#_Toc37433337)

[Manager table 34](#_Toc37433338)

[Customer table 35](#_Toc37433339)

[Car table 35](#_Toc37433340)

[SQL 36](#_Toc37433341)

[Single Row Functions: 36](#_Toc37433342)

[Group Functions: 37](#_Toc37433343)

[Subquery: 38](#_Toc37433344)

[Joining: 39](#_Toc37433345)

[View: 41](#_Toc37433346)

[Synonym: 42](#_Toc37433347)

[PL/SQL 43](#_Toc37433348)

[Function: 43](#_Toc37433349)

[Procedure: 46](#_Toc37433350)

[Record: 48](#_Toc37433351)

[Cursor: 51](#_Toc37433352)

[Trigger: 54](#_Toc37433353)

[Package: 56](#_Toc37433354)

# Introduction:

The “Car Shop Management System” has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

# Project Proposal:

The purpose of the car shop management system is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so their valuable data can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

The main objective of the project on car shop management system is to manage the details of car, customer, payment order. It manages all the information about the car. The purpose of the project is to build an application program to reduce the manual work for managing the car.

Here are three types of user – owner, manager, customer.

Manager can add car, delete car and sell car. Owner can also do that, but the main work is to maintain the manager. User can search any car; they can show everything and can booked a car and can buy it.

# Class Diagram:

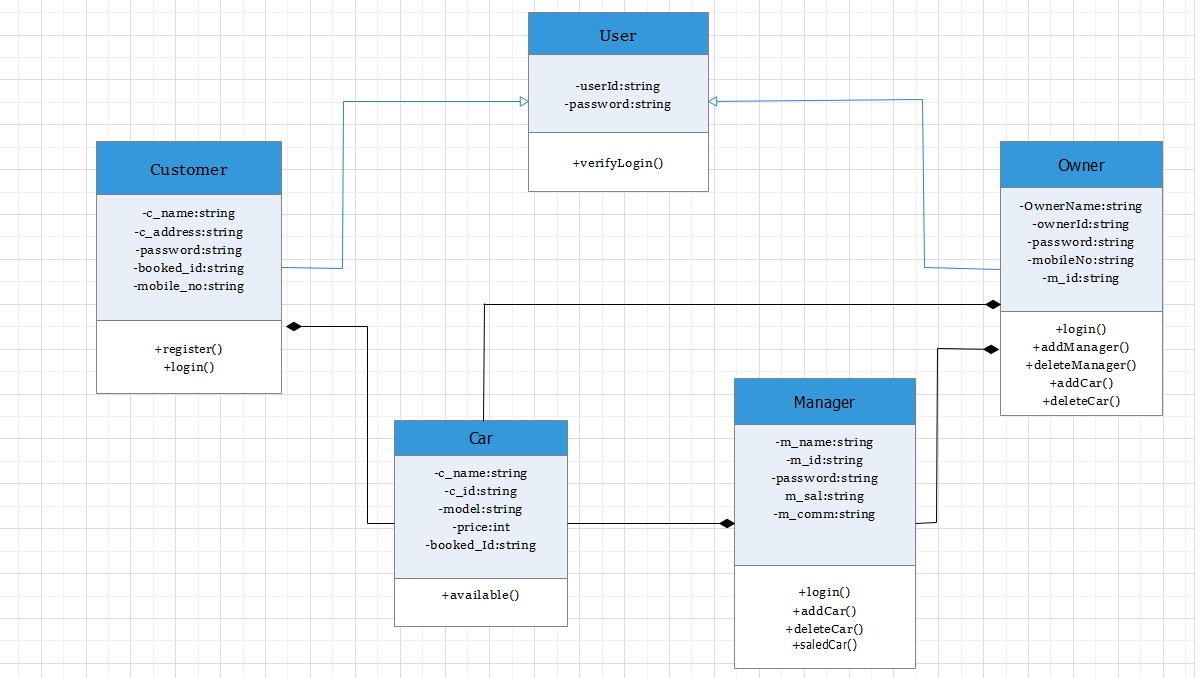


Fig 1: Class diagram.

# Use Case Diagram:

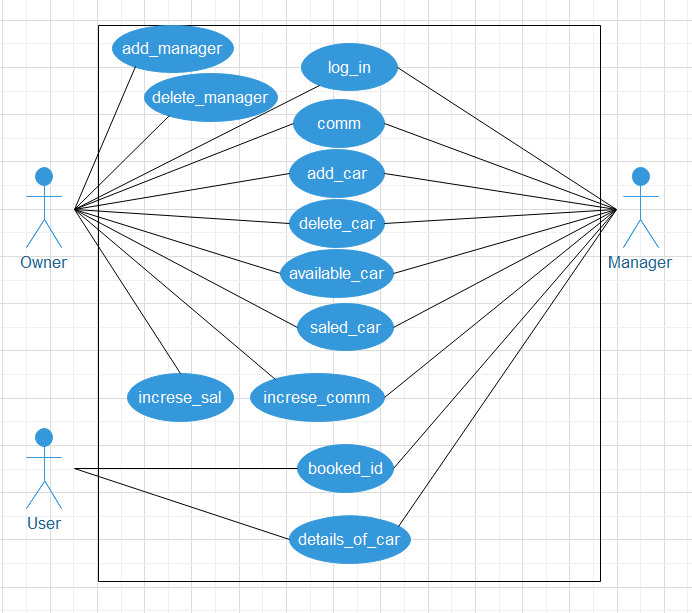


Fig 2: Use Case diagram.

# Activity Diagram:

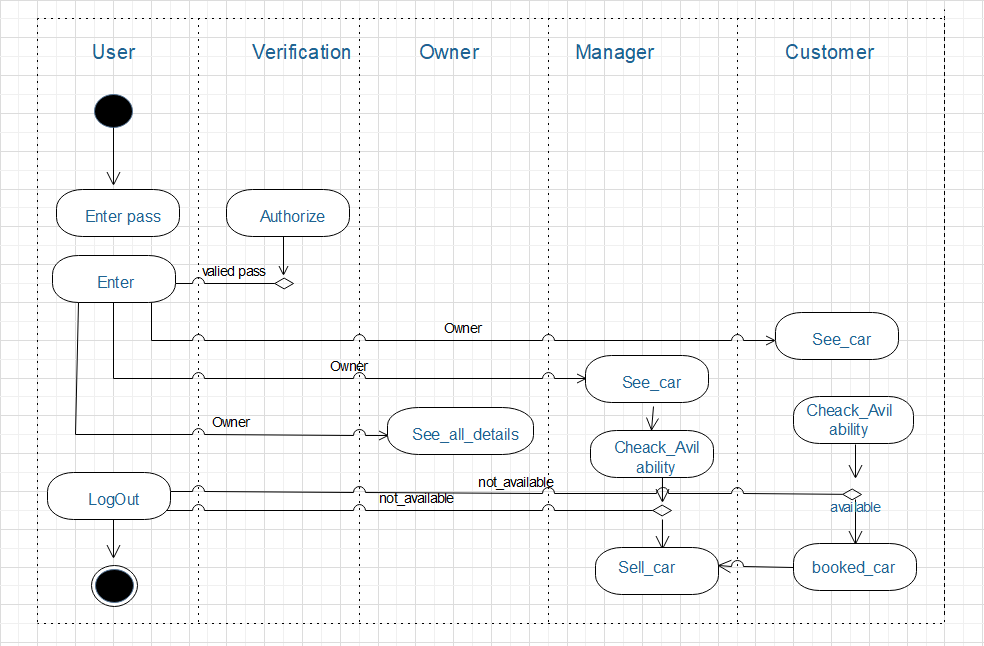
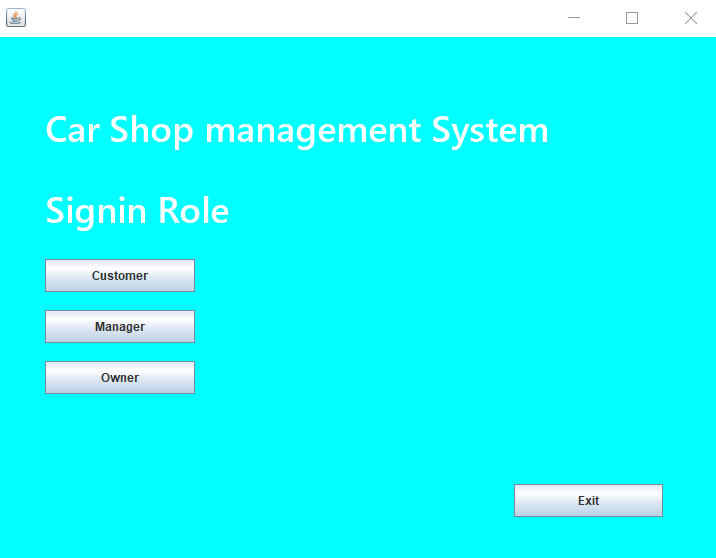


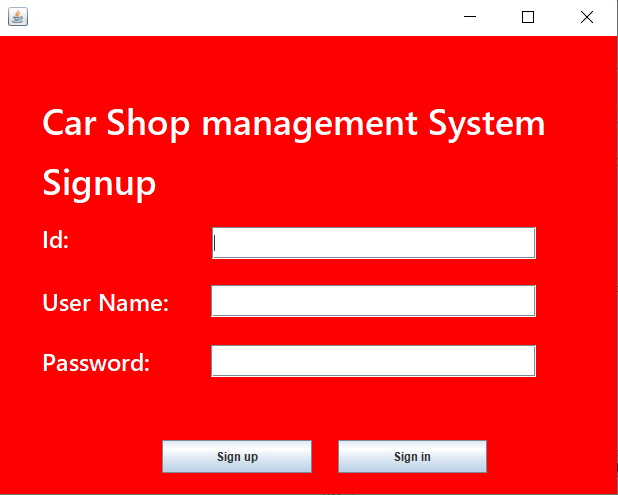
Fig 3: Activity diagram.

# User Interface:

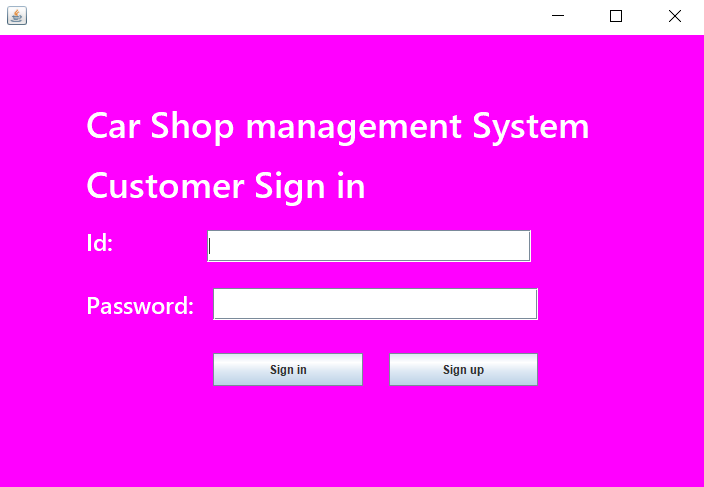
## Landing:



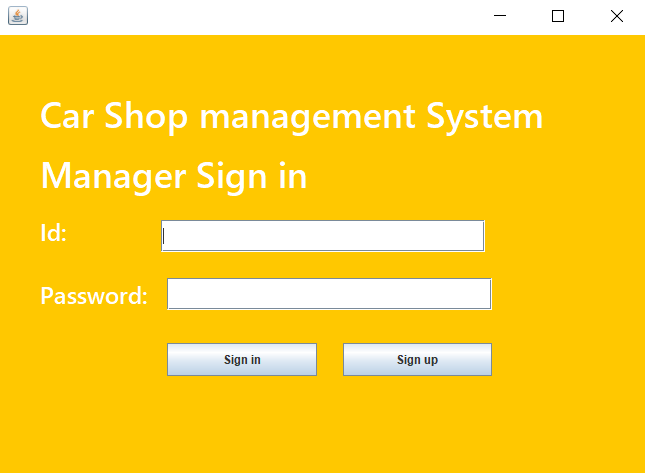
## Sign Up:



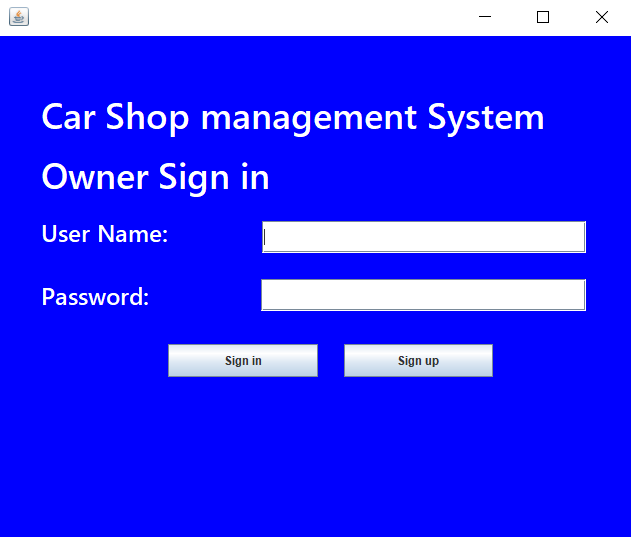
## Customer Sign In:



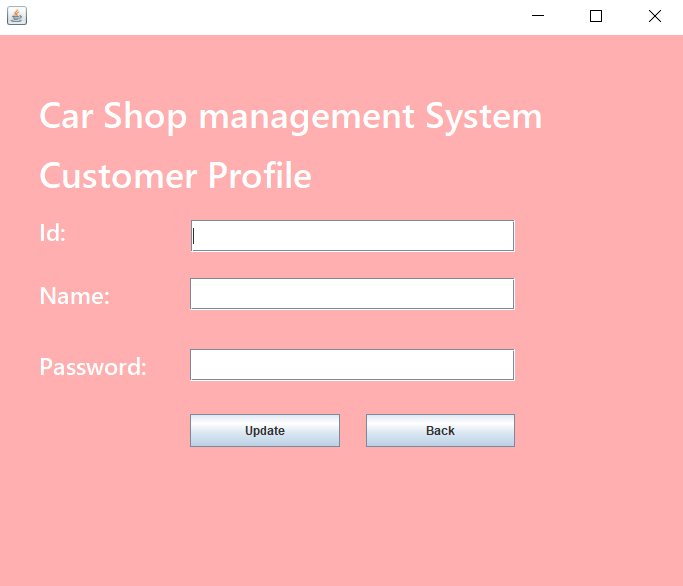
## Manager Sign In:



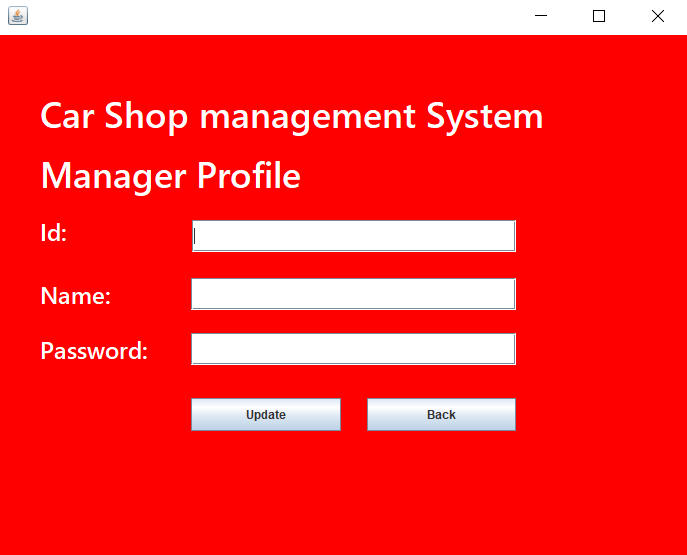
## Owner Sign In:



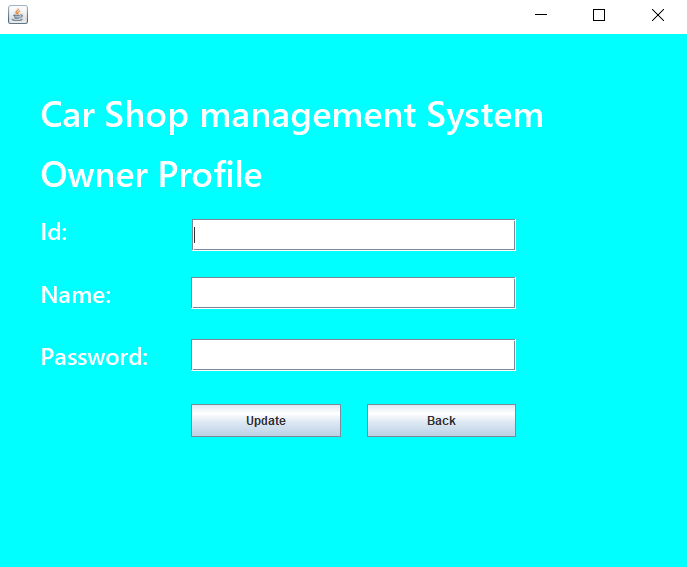
## Customer Profile:



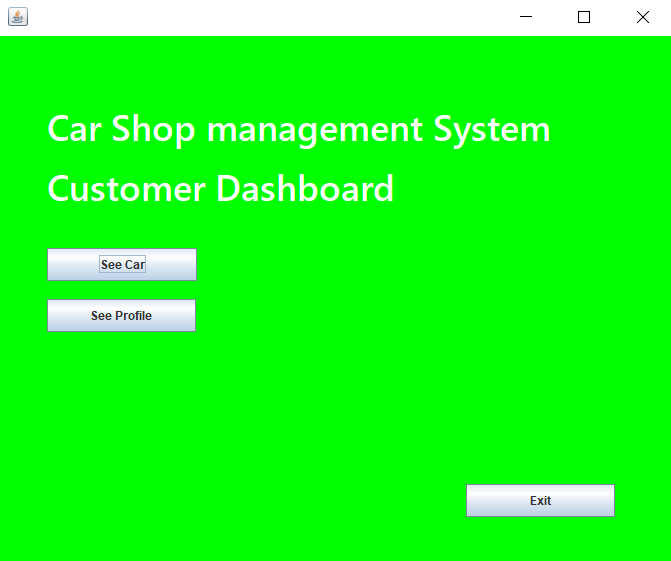
## Manager Profile:



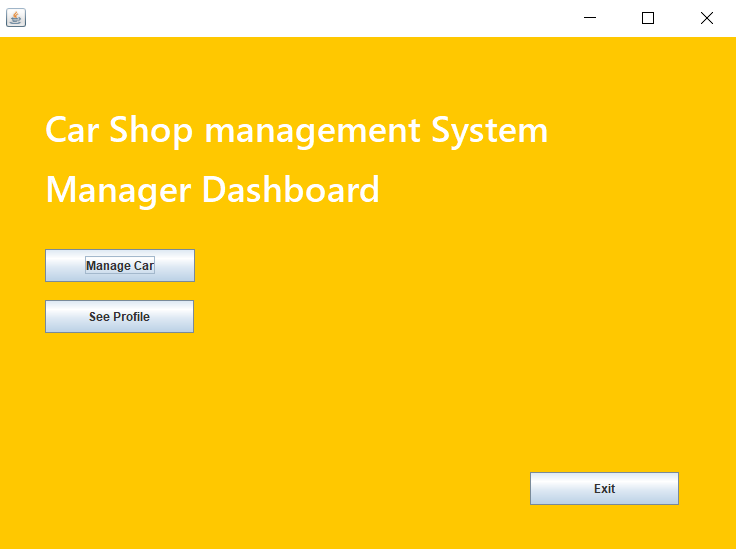
## Owner Profile:



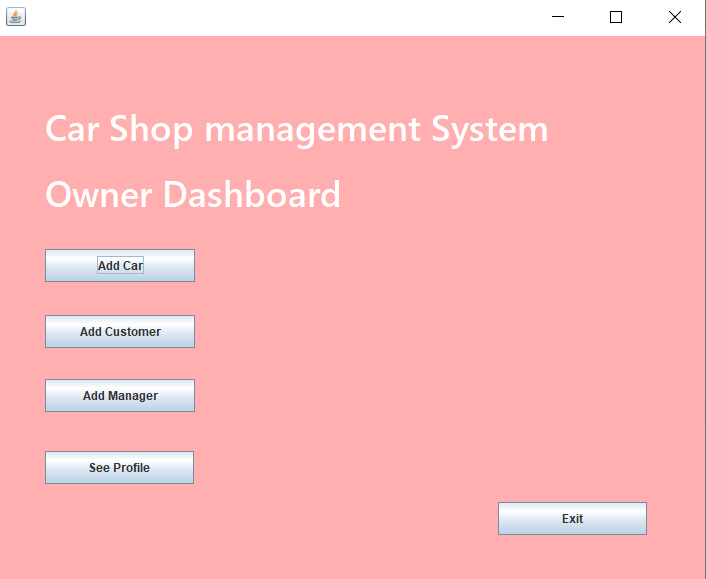
## Customer Dashboard:



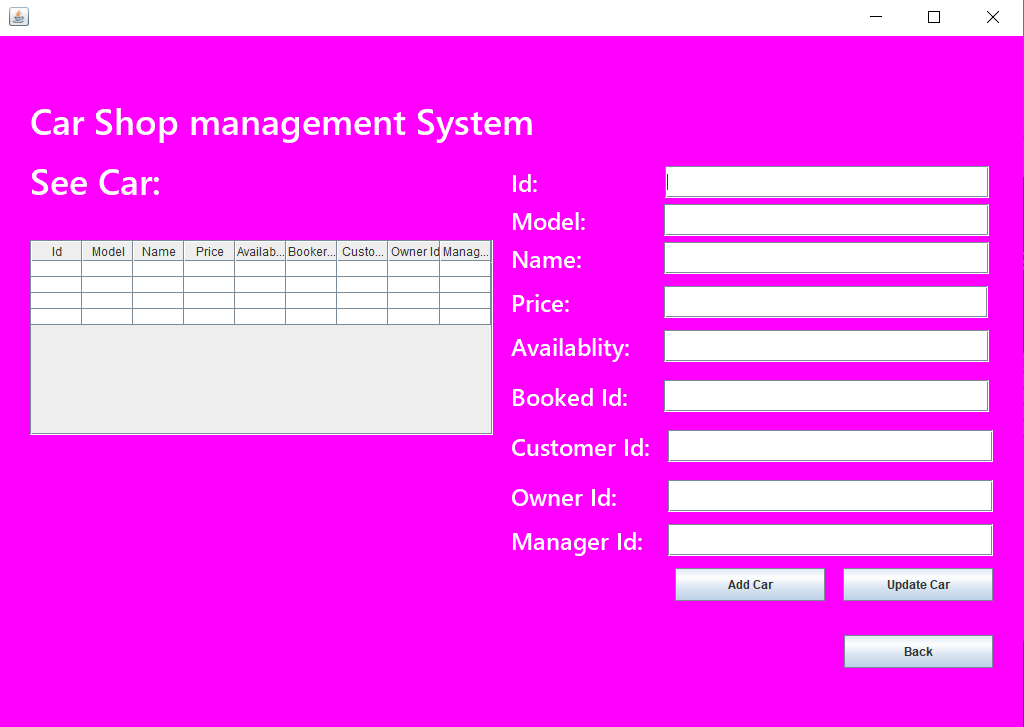
## Manager Dashboard:



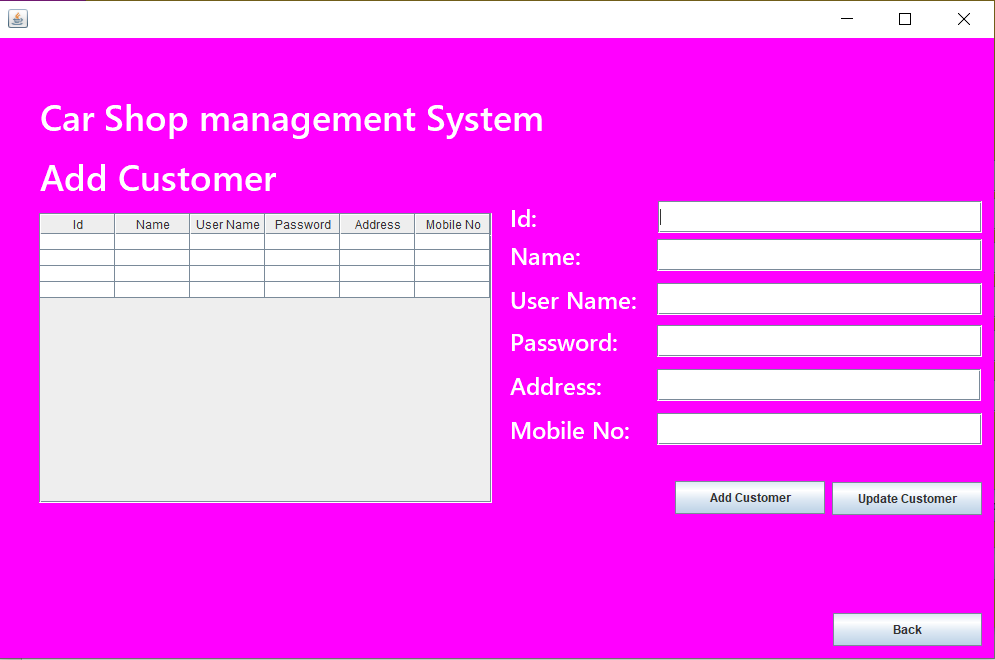
## Owner Dashboard:



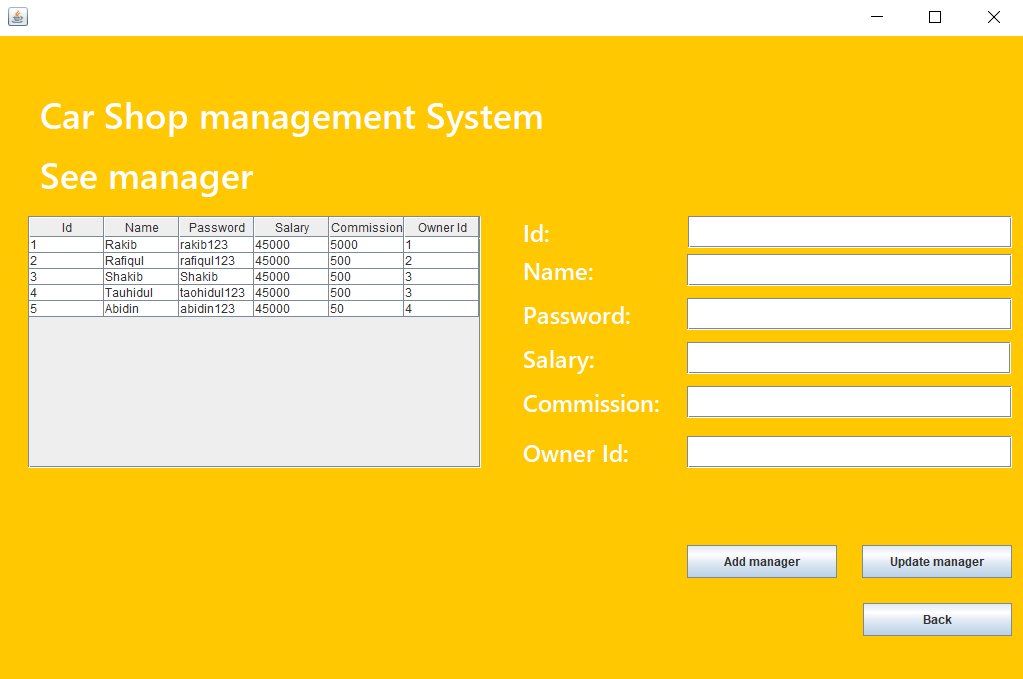
## Add Car:



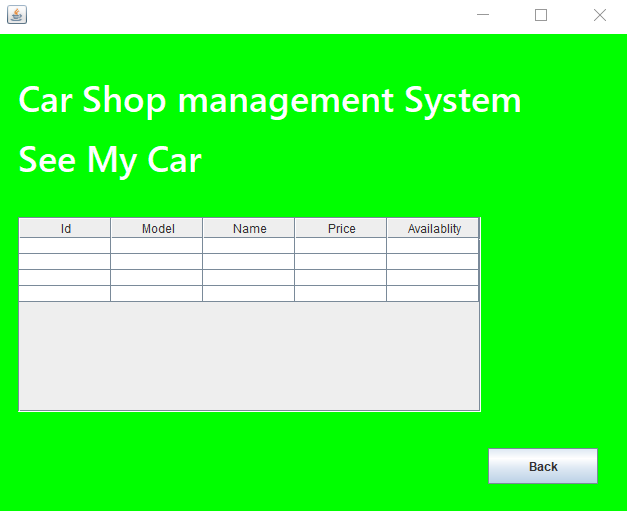
## Add Customer:



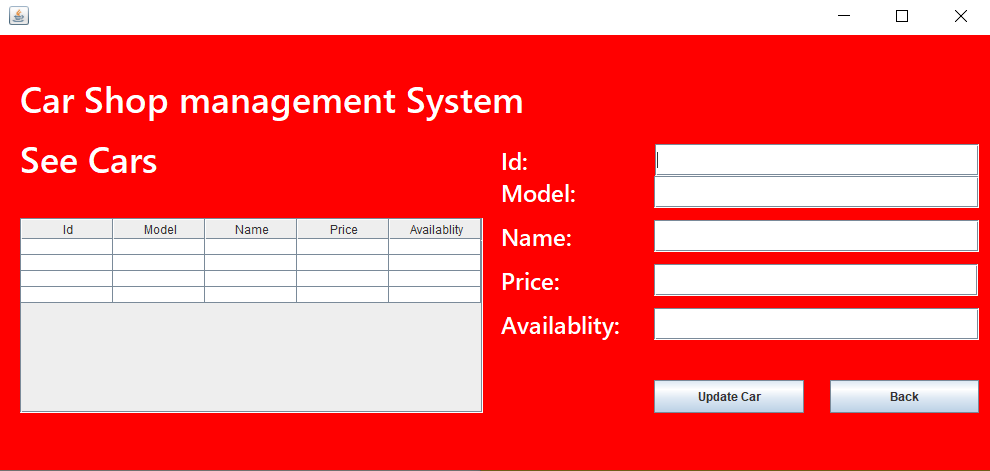
## Add Manager:



## See My Car:



## Manage Car:



# ER Diagram:

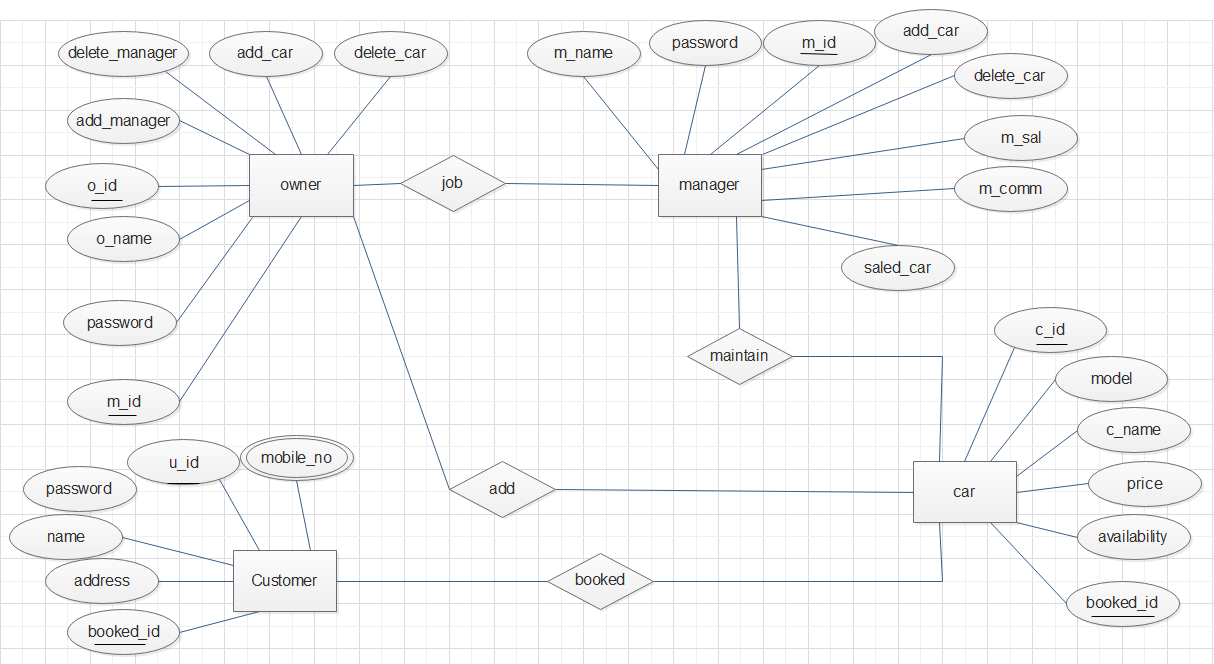


Fig 4: ER Diagram

# Normalization

**JOB**(#o\_id, o\_name, o\_password, m\_name, m\_password, #m-id, m\_sal, m\_comm)

**1NF:-**

There is no multi-valued attribute

**2NF:-**

1. #o\_id, o\_name, o\_password

2. #m\_id, m\_name, m\_password, m\_sal, m\_comm

**3NF:-**

1. #o\_id, o\_name, o\_password

2. #m\_id, m\_name, m\_password, m\_sal, m\_comm

**MAINTAIN**(m\_name, password, #m\_id, m\_sal, m\_comm, c\_id, model, c\_name, price, availability, booked\_id)

**1NF:-**

There is no multi-valued attribute

**2NF:-**

1. #m\_id, m\_name, m\_password, m\_sal, m\_comm

2. #c\_id, model, c\_name, price, availability, booked\_id

**3NF:-**

1. #m\_id, m\_name, m\_password, m\_sal, m\_comm

2. #c\_id, model, c\_name, price, availability, booked\_id

**ADD**(#o\_id, o\_name,password, #c\_id, model, c\_name, price, availability, booked\_id)

**1NF:-**

There is no multi-valued attribute

**2NF:-**

1. #o\_id, o\_name, o\_password

2. #c\_id, model, c\_name, price, availability, booked\_id

**3NF:-**

1. #o\_id, o\_name, o\_password

2. #c\_id, model, c\_name, price, availability, booked\_id

**Booked**(#c\_id, model, c\_name, price, availability, booked\_id, booked\_id, address, name, password, #u\_id, mobile\_no)

**1NF:-**

mobile\_no is an multi-valued attribute

**2NF:-**

1. #c\_id, model, c\_name, price, availability, booked\_id

2. #u\_id, address, name, password, mobile\_no

**3NF:-**

1. #c\_id, model, c\_name, price, availability, booked\_id

2. #u\_id, address, name, password, mobile\_no

# Schema Diagram:

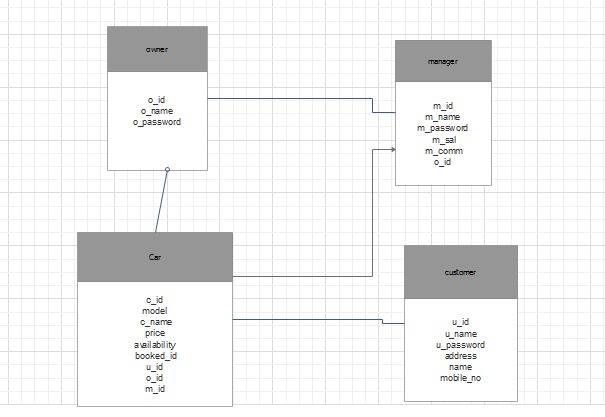


Fig 5: Schema Diagram

# Table Creation:

## Owner:

CREATE TABLE owner(

o\_id number(10),

o\_name varchar(30),

o\_password varchar(20),

constraint pk\_owner primary key(o\_id)

)

CREATE SEQUENCE seq\_o\_id

start with 1

increment by 1

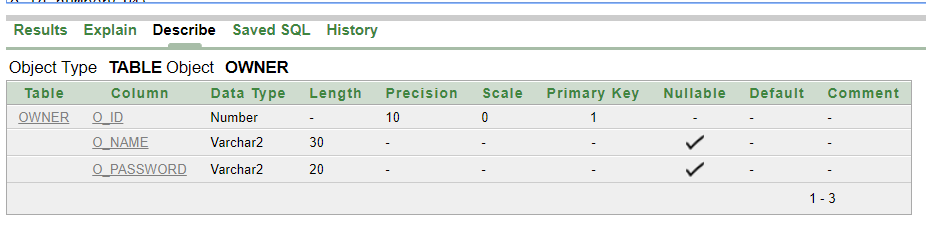
nomaxvalue

nominvalue

Nocycle;

INSERT INTO owner VALUES (seq\_o\_id.nextval, 'Rakib', 'rakib123');

Desc owner;



## Manager:

CREATE TABLE manager(

m\_id number(10),

m\_name varchar(30),

m\_password varchar(20),

m\_sal number(10),

m\_comm number(10),

o\_id number(10),

constraint pk\_manager primary key(m\_id),

constraint fk\_man\_own foreign key(o\_id) references owner(o\_id)

)

CREATE SEQUENCE seq\_m\_id

start with 1

increment by 1

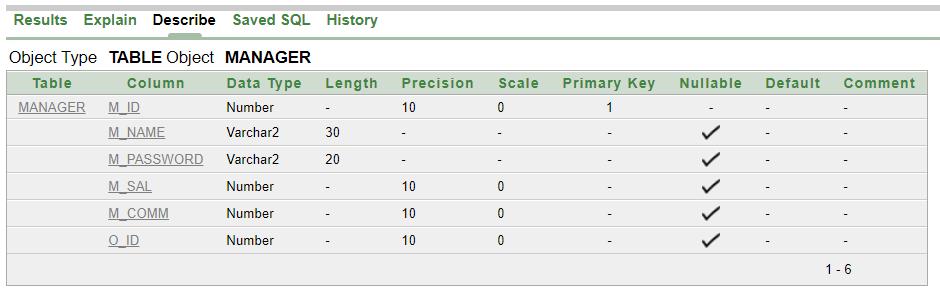
nomaxvalue

nominvalue

Nocycle;

INSERT INTO manager VALUES (seq\_m\_id.nextval, 'Rakib', 'rakib123', '45000', '5000', '1');

Desc manager;



## Customer:

CREATE TABLE customer(

u\_id number(10),

u\_name varchar2(30),

u\_password varchar2(20),

address varchar2(10),

name varchar2(10),

mobile\_no number(10),

constraint pk\_customer primary key(u\_id)

)

CREATE SEQUENCE seq\_u\_id

start with 4

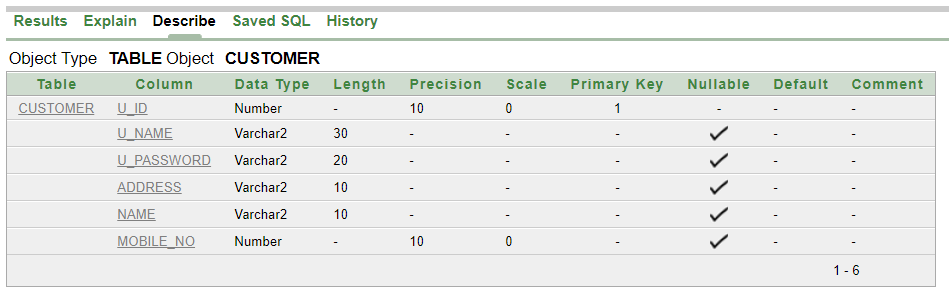
increment by 1

nomaxvalue

nominvalue

Nocycle;

INSERT INTO customer VALUES (seq\_u\_id.nextval, 'Rakib', 'rakib123', 'dhanmondi', 'Dhaka', '546548432');



## Car:

CREATE TABLE car(

c\_id number(10),

c\_name varchar(30),

price number(10),

model varchar2(20),

availability varchar2(20),

booked\_id number(10),

m\_id number(10),

o\_id number(10),

u\_id number(10),

constraint pk\_car primary key(c\_id),

constraint fk\_car\_man foreign key(m\_id) references manager(m\_id),

constraint fk\_car\_own foreign key(o\_id) references owner(o\_id),

constraint fk\_car\_cus foreign key(u\_id) references customer(u\_id)

)

CREATE SEQUENCE seq\_c\_id

start with 1

increment by 1

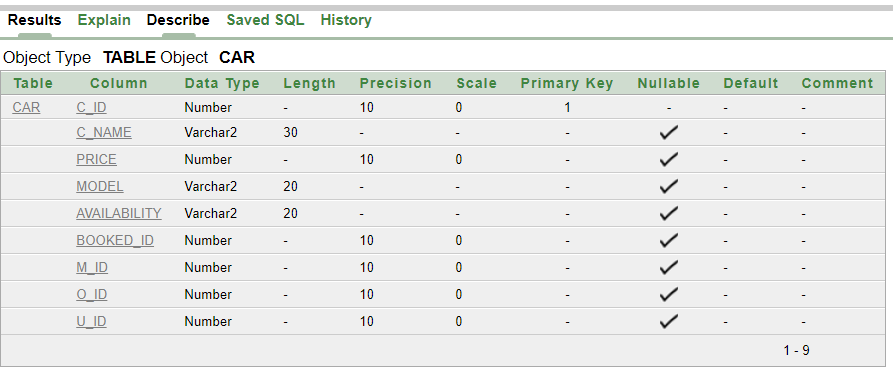
nomaxvalue

nominvalue

Nocycle;

INSERT INTO car VALUES (seq\_c\_id.nextval, 'Ferari', '5000000', 'r1', 'yes', '1', '1', '1', '6');

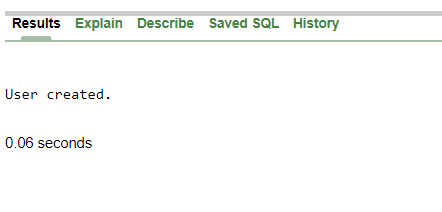
Desc car;



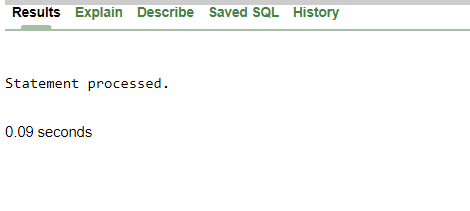
## Create User:

### Owner:

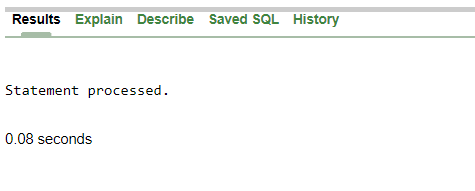
1. **CREATE USER owner IDENTIFIED BY owner;**



1. **GRANT UNLIMITED TABLESPACE TO owner;**

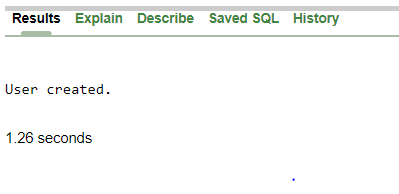


1. **GRANT create table, create sequence, create view TO owner;**

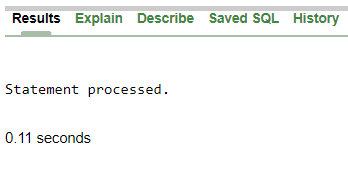


## Manger:

1. CREATE USER manager IDENTIFIED BY manager;

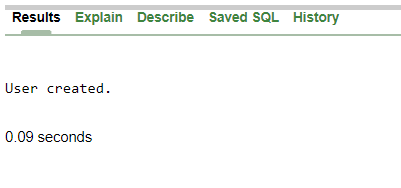


1. GRANT create table, create sequence, create view TO manager;



## Customer:

1. CREATE USER customer IDENTIFIED BY customer;



# INSERTION

## Owner table

INSERT INTO owner VALUES (seq\_o\_id.nextval, 'Rakib', 'rakib123');

INSERT INTO owner VALUES (seq\_o\_id.nextval, 'Rafiqul', 'rafiqul123');

INSERT INTO owner VALUES (seq\_o\_id.nextval, ‘Shakib’, 'shakib123');

INSERT INTO owner VALUES (seq\_o\_id.nextval, ‘Tauhidul', 'taohidul123');

INSERT INTO owner VALUES (seq\_o\_id.nextval, 'Abidin', ‘abidin123');

select \* from owner;



## Manager table

INSERT INTO manager VALUES (seq\_m\_id.nextval, 'Rakib', 'rakib123', '45000', '5000', '1');

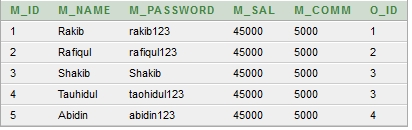
INSERT INTO manager VALUES (seq\_m\_id.nextval, 'Rafiqul', 'rafiqul123', '45000', '500', '2');

INSERT INTO manager VALUES (seq\_m\_id.nextval, 'Shakib', 'Shakib', '45000', '500', '3');

INSERT INTO manager VALUES (seq\_m\_id.nextval, 'Tauhidul', 'taohidul123', '45000', '500', '3');

INSERT INTO manager VALUES (seq\_m\_id.nextval, 'Abidin', 'abidin123', '45000', '50', '4');

select \* from manager;



## Customer table

INSERT INTO customer VALUES (seq\_u\_id.nextval, 'Rakib', 'rakib123', 'dhanmondi', 'Dhaka', '546548432');

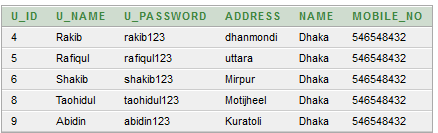
INSERT INTO customer VALUES (seq\_u\_id.nextval, 'Rafiqul', 'rafiqul123', 'uttara', 'Dhaka', '546548432');

INSERT INTO customer VALUES (seq\_u\_id.nextval, 'Shakib', 'shakib123', 'Mirpur', 'Dhaka', '546548432');

INSERT INTO customer VALUES (seq\_u\_id.nextval, 'Taohidul', 'taohidul123', 'Motijheel', 'Dhaka', '546548432');

INSERT INTO customer VALUES (seq\_u\_id.nextval, 'Abidin', 'abidin123', 'Kuratoli', 'Dhaka', '546548432');

select \* from customer;



## Car table

INSERT INTO car VALUES (seq\_c\_id.nextval, 'Ferari', '5000000', 'r1', 'yes', '1', '1', '1', '6');

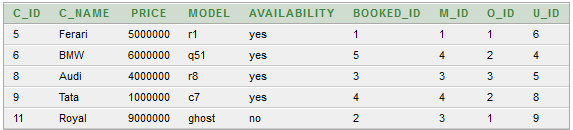
INSERT INTO car VALUES (seq\_c\_id.nextval, 'BMW', '6000000', 'q51', 'yes', '5', '4', '2', '4');

INSERT INTO car VALUES (seq\_c\_id.nextval, 'Audi', '4000000', 'r8', 'yes', '3', '3', '3', '5');

INSERT INTO car VALUES (seq\_c\_id.nextval, 'Tata', '1000000', 'c7', 'yes', '4', '4', '2', '8');

INSERT INTO car VALUES (seq\_c\_id.nextval, 'Royal', '9000000', 'ghost', 'no', '2', '3', '1', '9');

select \* from car;

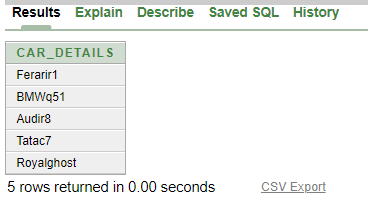


# SQL

## Single Row Functions:

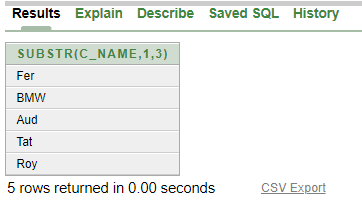
1. Display car’s name and model as “car\_details”.

select concat(c\_name,model) as car\_details from car;



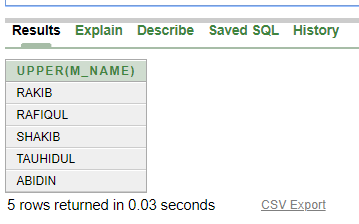
1. Display first 3 letters of all the cars.

select substr(c\_name,1,3) from car;



1. Display all the manager’s name in upper case.

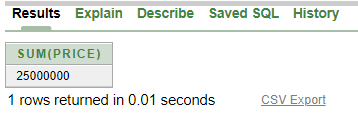
select upper(m\_name) from manager;



## Group Functions:

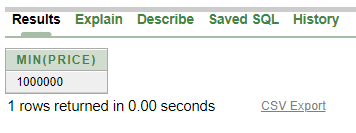
1. Calculate the sum of price of all the cars.

select sum(price) from car;



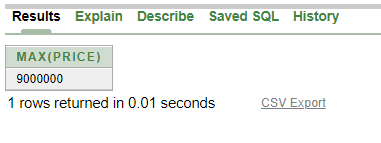
1. Find out the lowest price from the cars.

select min(price) from car;



1. Find out the highest price from the cars.

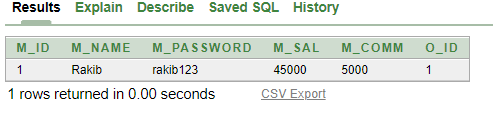
select max(price) from car;



## Subquery:

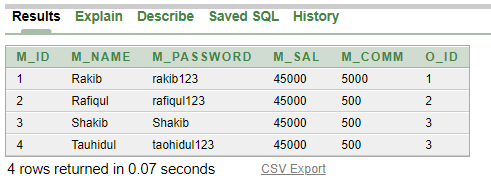
1. Find out the managers data whose commission is more than the manager id 4.

select \* from manager where m\_comm> (select m\_comm from manager where m\_id=4)



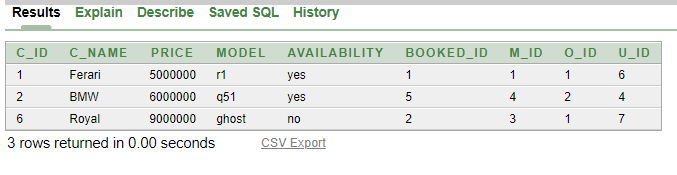
1. Find out the managers whose commission is more than the manager name Abidin.

select \* from manager where m\_comm> (select m\_comm from manager where m\_name='Abidin')



1. Find out the cars which price is more than the model r8;

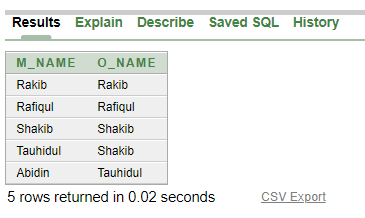
select \* from car where price> (select price from car where model='r8')



## Joining:

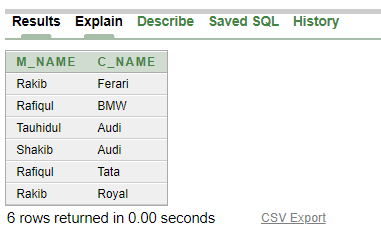
1. Display all the managers and their respective owners.

select m.m\_name, o.o\_name from manager m, owner o where m.o\_id = o.o\_id;



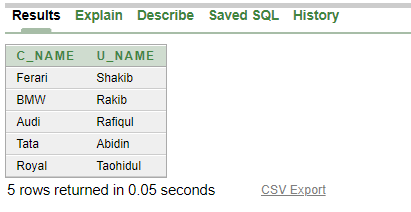
1. Display all the managers and their assigned cars.

select m.m\_name, c.c\_name from manager m, car c where m.o\_id = c.o\_id;



1. Display all the customers and their owned cars.

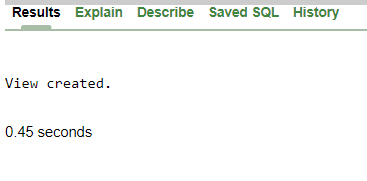
select c.c\_name, cu.u\_name from customer cu, car c where cu.u\_id = c.u\_id;



## View:

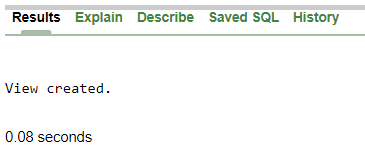
1. Create a view to display customer name and their id.

create view view\_customer as select u\_name, u\_id from customer;



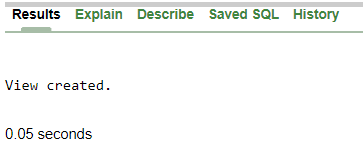
1. Create a view to display car name and their id.

create view view\_car as select c\_name, c\_id from car;



1. Create a view to display manager and their id.

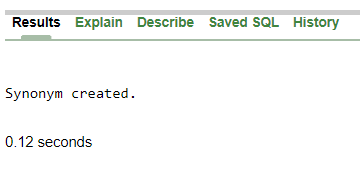
create view view\_manager as select m\_name, m\_id from manager;



## Synonym:

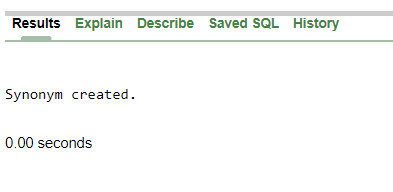
1. Create a synonym for manager table.

create synonym syn\_man for manager;



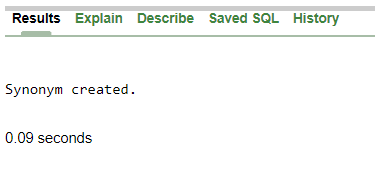
1. Create a synonym for customer table.

create synonym syn\_cus for customer;



1. Create a synonym for owner table.

create synonym syn\_own for owner;



# PL/SQL

## Function:

1. Create a function to display data from manager table.

CREATE OR REPLACE FUNCTION selectMsg(p\_name IN VARCHAR2)

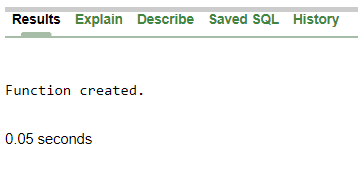
RETURN VARCHAR2

IS

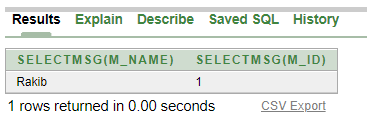
BEGIN

RETURN (p\_name);

END;



SELECT selectMsg(m\_name), selectMsg(m\_id) FROM manager WHERE m\_id=1;



1. Create/replace a function to show the number of total customers.

CREATE OR REPLACE FUNCTION totalCustomer

RETURN NUMBER

IS

total NUMBER := 0;

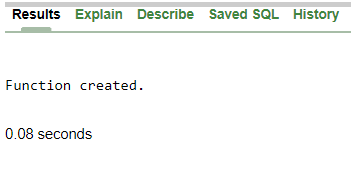
BEGIN

SELECT count(\*) INTO total

FROM customer;

RETURN total;

END;



DECLARE

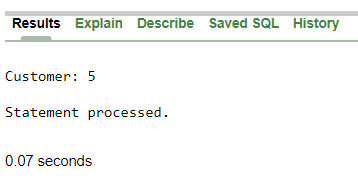
customer NUMBER(10);

BEGIN

customer := totalCustomer();

dbms\_output.put\_line('Customer: '||customer);

END;



1. Create a function to display total number of cars.

create or replace function totalCar

return number as

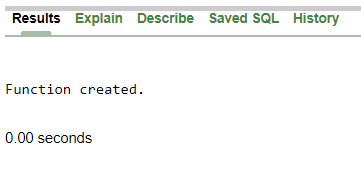
totalnum number(10);

begin

select count(c\_id) into totalnum from car;

return totalnum;

end;



declare

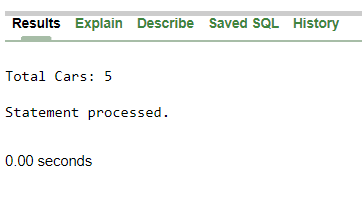
totalnum number(10);

begin

totalnum:=totalCar();

dbms\_output.put\_line('Total Cars: '||totalnum);

end;



## Procedure:

1. Create a procedure to update customer.

CREATE OR REPLACE PROCEDURE updateCustomer

(id IN NUMBER,

username IN VARCHAR2,

password IN VARCHAR2,

address IN VARCHAR2,

state IN VARCHAR2,

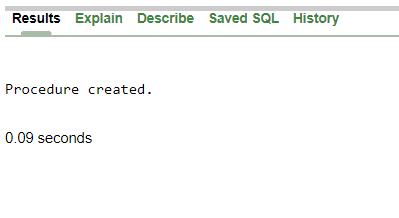
phone IN NUMBER)

IS

BEGIN

UPDATE customer SET u\_name=username, u\_password=password, address=address, name=state, mobile\_no=phone WHERE u\_id=id;

END;



1. Create a procedure to print car details.

CREATE OR REPLACE PROCEDURE print\_car

(id IN NUMBER)

IS

car\_details car%ROWTYPE;

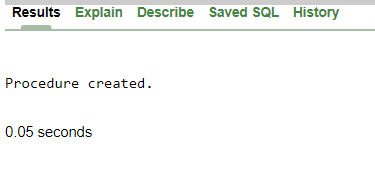
BEGIN

SELECT \* INTO car\_details

FROM car WHERE c\_id=id;

dbms\_output.put\_line(car\_details.c\_name || ' price is: ' || car\_details.price || ' and Model is: ' || car\_details.model);

END;



1. Create a procedure to update owner.

CREATE OR REPLACE PROCEDURE updateOwner

(id IN NUMBER,

username IN VARCHAR2,

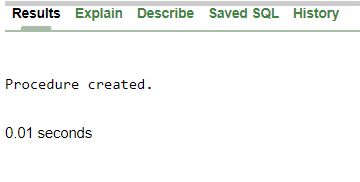
password IN VARCHAR2)

IS

BEGIN

UPDATE owner SET o\_name=username, o\_password=password WHERE o\_id=id;

END;



## Record:

1. Create a user defined record to display car name and car model whose id is 6.

DECLARE

TYPE rec\_car IS RECORD(

car\_name car.c\_name%TYPE,

car\_model car.model%TYPE

);

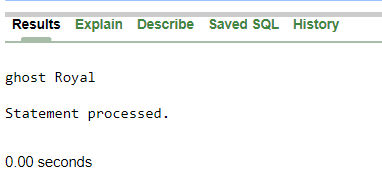
var rec\_car;

BEGIN

SELECT c\_name, model INTO var.car\_name, var.car\_model FROM car WHERE c\_id=6;

dbms\_output.put\_line(var.car\_model ||' '||var.car\_name);

END;



1. Create a cursor based record to display manager name and manager salary.

DECLARE

CURSOR c\_manager IS

SELECT m\_name, m\_sal FROM manager;

var\_man c\_manager%ROWTYPE;

BEGIN

OPEN c\_manager;

LOOP

FETCH c\_manager INTO var\_man;

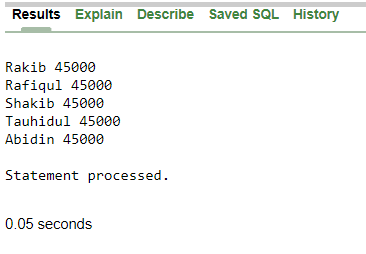
EXIT WHEN c\_manager%NOTFOUND;

dbms\_output.put\_line(var\_man.m\_name || ' ' || var\_man.m\_sal);

END LOOP;

CLOSE c\_manager;

END;



1. Create a cursor based record to display all customer’s name and their address.

DECLARE

CURSOR c\_customer IS

SELECT u\_name, address FROM customer;

var\_cus c\_customer%ROWTYPE;

BEGIN

OPEN c\_customer;

LOOP

FETCH c\_customer INTO var\_cus;

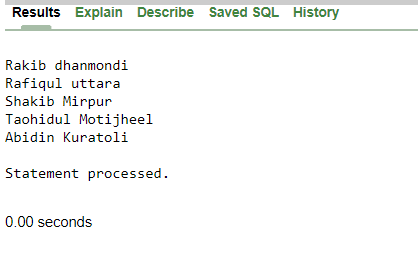
EXIT WHEN c\_customer%NOTFOUND;

dbms\_output.put\_line(var\_cus.u\_name || ' ' || var\_cus.address);

END LOOP;

CLOSE c\_customer;

END;



## Cursor:

1. Create an implicit cursor to add 200tk for each manager in their commission.

DECLARE

total NUMBER(2);

BEGIN

UPDATE manager

SET m\_comm = m\_comm + 200;

IF sql%NOTFOUND THEN

dbms\_output.put\_line('No Manager Updated');

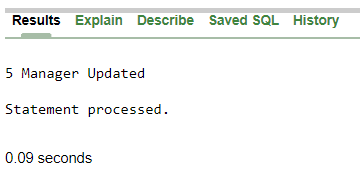
ELSIF sql%FOUND THEN

total := sql%ROWCOUNT;

dbms\_output.put\_line(total||' Manager Updated');

END IF;

END;



1. Create an explicit cursor to display owner id and name.

DECLARE

c\_id owner.o\_id%TYPE;

c\_name owner.o\_name%TYPE;

CURSOR cursor\_owner IS

SELECT o\_id, o\_name FROM owner;

BEGIN

OPEN cursor\_owner;

LOOP

FETCH cursor\_owner INTO c\_id, c\_name;

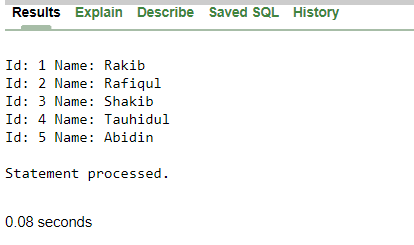
EXIT WHEN cursor\_owner%NOTFOUND;

dbms\_output.put\_line('Id: '||c\_id||' Name: '||c\_name);

END LOOP;

CLOSE cursor\_owner;

END;



1. Create a cursor using for loop to display car name and price.

DECLARE

CURSOR c\_car IS

SELECT c\_name, price FROM car;

BEGIN

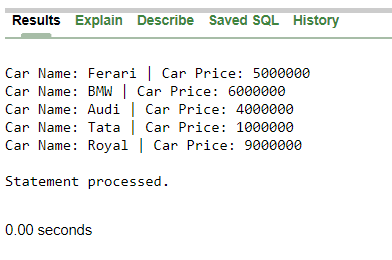
FOR loop\_index IN c\_car

LOOP

dbms\_output.put\_line('Car Name: '||loop\_index.c\_name || ' | Car Price: '||loop\_index.price);

END LOOP;

END;



## Trigger:

1. Create a trigger for any new customer inserted into the customer table.

create trigger customer\_t

after insert on customer

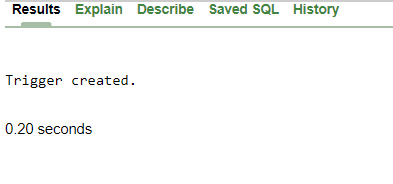
for each row

when (new.u\_id > 0)

begin

dbms\_output.put\_line('New Customer Added');

end;



1. Create a trigger for any new manager inserted into the manager table.

create trigger manager\_t

after insert on manager

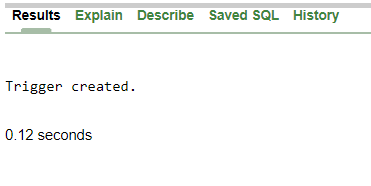
for each row

when (new.m\_id > 0)

begin

dbms\_output.put\_line('New Manager Added');

end;



1. Create a trigger for any new car inserted into the car table.

create trigger car\_t

after insert on car

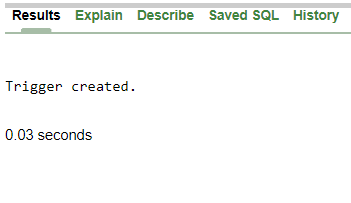
for each row

when (new.m\_id > 0)

begin

dbms\_output.put\_line('New Car Added');

end;



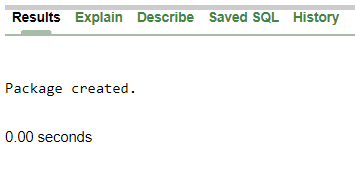
## Package:

1. Create a package which has a procedure to update car model.

CREATE OR REPLACE PACKAGE pkg\_car IS

PROCEDURE update\_model(model in varchar);

END pkg\_car;



CREATE OR REPLACE PACKAGE BODY pkg\_car IS

PROCEDURE update\_model(model in varchar)

as

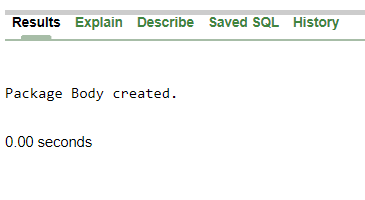
begin

update car set model=model;

dbms\_output.put\_line(model);

end;

END pkg\_car;

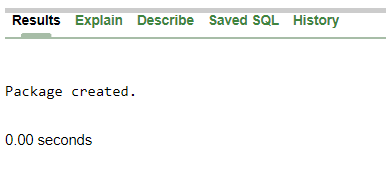


1. Create a package with a procedure to update salary of the managers.

CREATE OR REPLACE PACKAGE pkg\_manager IS

PROCEDURE update\_salary(salary in number);

END pkg\_manager;



CREATE OR REPLACE PACKAGE BODY pkg\_manager IS

PROCEDURE update\_salary(salary in number)

as

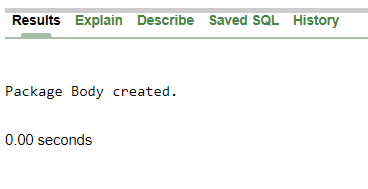
begin

update manager set m\_sal=salary;

dbms\_output.put\_line(salary);

end;

END pkg\_manager;

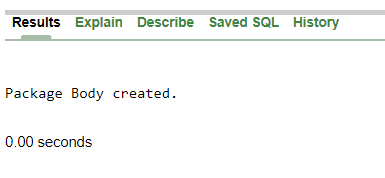


1. Create a package with a procedure to update owner name.

CREATE OR REPLACE PACKAGE pkg\_owner IS

PROCEDURE update\_name(name in varchar);

END pkg\_owner;



CREATE OR REPLACE PACKAGE BODY pkg\_owner IS

PROCEDURE update\_name(name in varchar)

as

begin

update owner set o\_name=name;

dbms\_output.put\_line(name);

end;

END pkg\_owner;

