NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI

Department of Computer Application

Cab Booking

PL/SQL

PROJECT WORK

Submitted By :

SHIVAM TIWARI

205118070

Under the guidance of

Dr. Ms. R. Eswari

NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI

CERTIFICATE

*This is to certify that **Shivam Tiwari** student of 2nd semester MCA (batch 2018-2021) of National Institute of Technology, Tiruchirappalli has successfully completed the project **Cab Booking** in PL/SQL under the guidance of **Dr. Ms. R. Eswari.***

Signature

INDEX

ABSTRACT

In this PL/SQL project, there will be provided modules where one can get a cab of different facilities accordingly nearby him. User will choose here his location and destiny on the modules; this application will book the available cab present in database.

Initially this application will consider particular city to travel in with different available places within the city. Laterwards its area of service will be stretched to cover the entire state and so on.

The language in use will be **PL/SQL** providing above mentioned featured in it. The database server which will handle the data is **ORACLE**.

Few schema entities will be :

1. Cab

2. Customer

3. Driver

4. Owner

5. Cab_ride

6. Payment_type

Concepts Used

- Procedures:
    - ➢Reg_driver
    - ➢Reg_cab
    - ➢Reg_owner
    - ➢Reg_car_model
    - ➢Reg_driver_cab
    - ➢Reg_payment_type
    - ➢Reg_cab_ride
    - ➢Reg_distance_map
    - ➢Reg_customer
    - ➢Reg_cab_ride_history
    - ➢Rem_driver
    - ➢Rem_owner
    - ➢Rem_car_model
    - ➢Rem_customer
    - ➢Free_cab
    - ➢Free_all_cab

- Functions:
    - ➢Cab_booking

- Triggers:
    - ➢Rem_driver
    - ➢Bef_reg_driver
    - ➢Pay_tri
    - ➢Post_cab
    - ➢Rem_cab_ride
    - ➢Cus_tri
    - ➢Rem_customer
    - ➢Bef_reg_cab
    - ➢Rem_cab
    - ➢Bef_reg_owner
    - ➢Rem_owner
    - ➢Rem_car_model

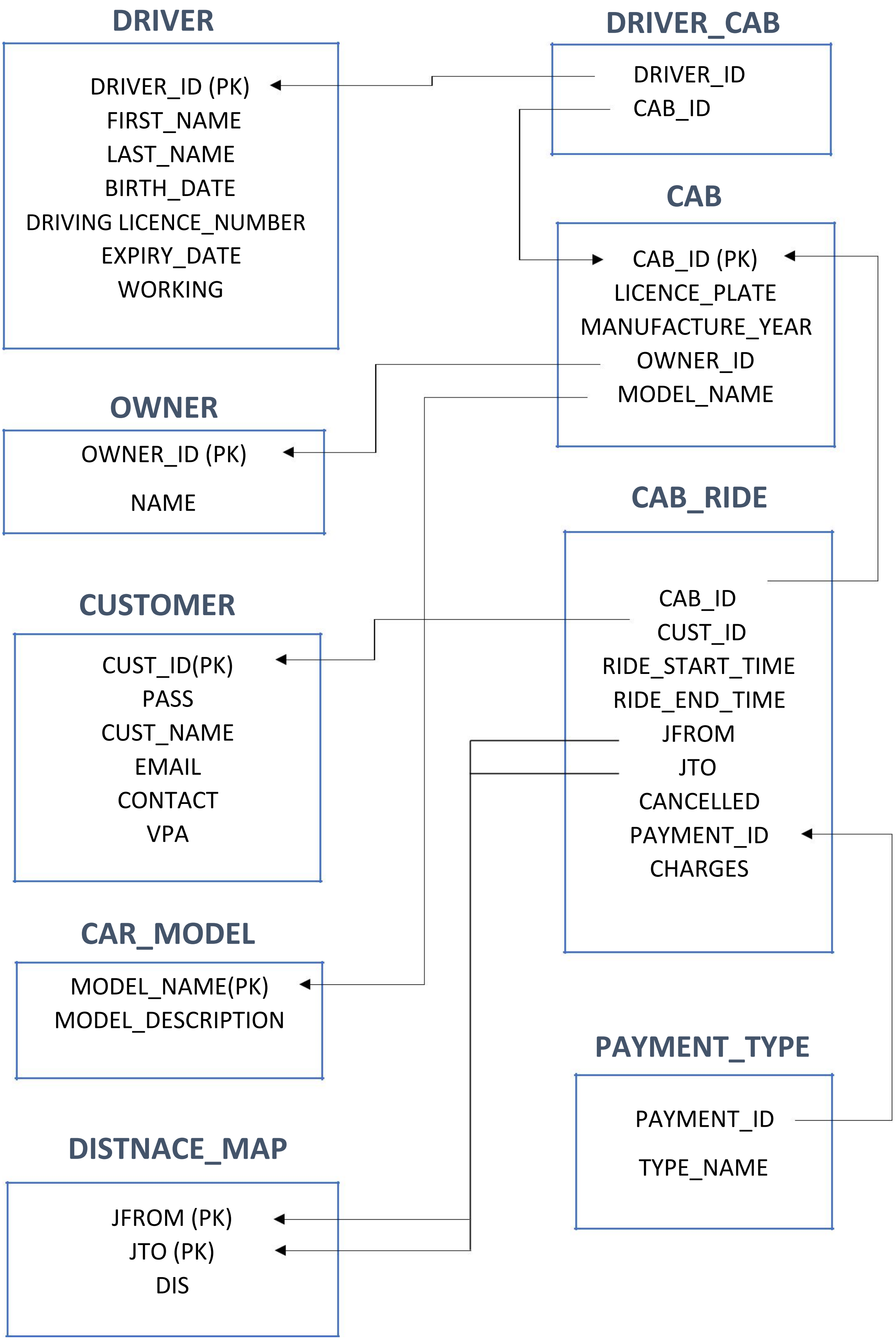
- Sequences:
    - ➢Payid
    - ➢Cusid
    - ➢Drivid
    - ➢Cabid
    - ➢Ownid

CAB BOOKING

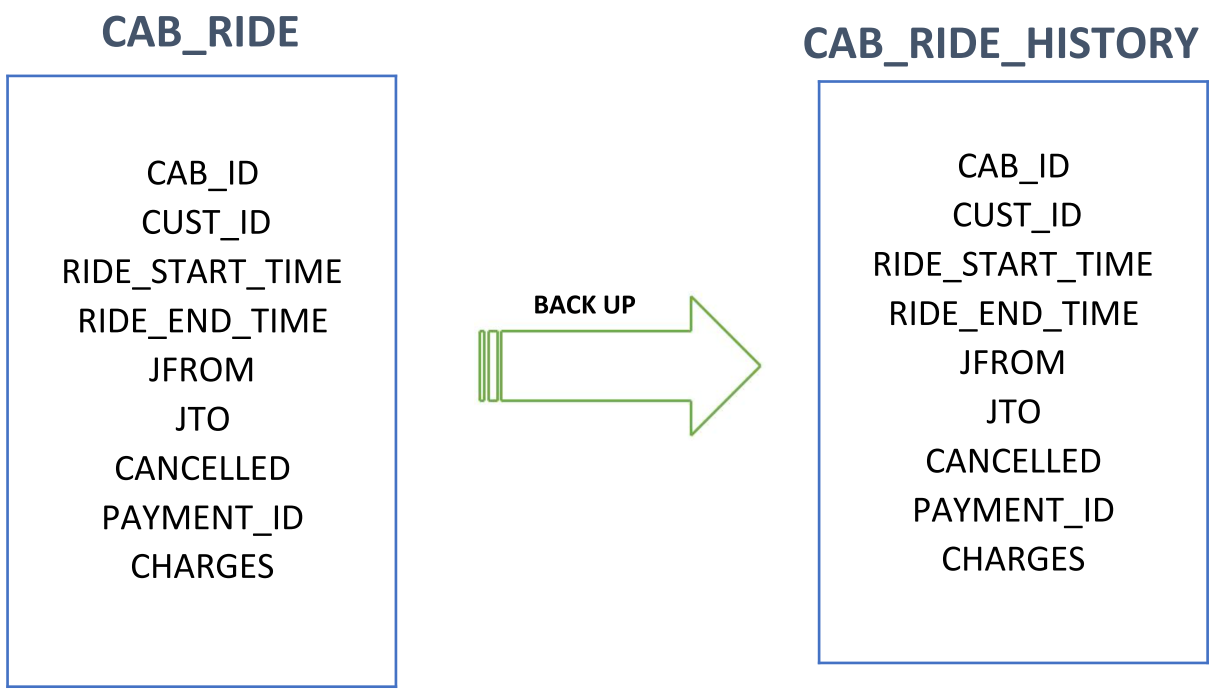
Functional Dependencies

APRIL 16, 2019

DBMS PL/SQL


DRIVER
DRIVER_ID (PK)
FIRST_NAME
LAST_NAME
BIRTH_DATE
DRIVING LICENCE_NUMBER
EXPIRY_DATE
WORKING
DRIVER_CAB
DRIVER_ID
CAB_ID
CAB
CAB_ID (PK)
LICENCE_PLATE
MANUFACTURE_YEAR
OWNER_ID
MODEL_NAME
OWNER
OWNER_ID (PK)
NAME
CAB_RIDE
CAB_ID
CUST_ID
RIDE_START_TIME
RIDE_END_TIME
JFROM
JTO
CANCELLED
PAYMENT_ID
CHARGES
CUSTOMER
CUST_ID(PK)
PASS
CUST_NAME
EMAIL
CONTACT
VPA
CAR_MODEL
MODEL_NAME(PK)
MODEL_DESCRIPTION
PAYMENT_TYPE
PAYMENT_ID
TYPE_NAME
DISTNACE_MAP
JFROM (PK)
JTO (PK)
DIS

There is one more table which is used to make history back up of rides made till now. If any deletion is made from cab_ride table all the entries will be backed up in this table for future references.



CAB BOOKING

Table Creation

APRIL 16, 2019

DBMS PL/SQL

1. DRIVER:

```
            SQL> create table driver(driver_id int PRIMARY KEY,first_name varchar(128),last_name
varchar(128),birth_date date,driving_licence_number varchar(128),expiry_date date,working
char(1));
```

2. CAR_MODEL:

```
            SQL> create table car_model(model_name

            varchar(64) PRIMARY KEY,model_description varchar(500));
```

3. OWNER:

```
            SQL> create table owner(owner_id int PRIMARY KEY,name varchar(100));
```

4. CAB:

```
            SQL> create table cab (cab_id int PRIMARY KEY,

               licence_plate varchar(32),

               manufacture_year int,

               owner_id int NOT NULL,

                model_name varchar2(64)

               );


             alter table cab

             add constraint fk123 FOREIGN KEY (model_name)

             references car_model(model_name);


             alter table cab

             add constraint f.. FOREIGN KEY (owner_id)

            references owner(owner_id);
```

5. PAYMENT_TYPE:

```
SQL> create table payment_type(payment_id int,type_name varchar(128));
```

```
SQL> alter table payment_type
      add constraint fkp FOREIGN KEY (payment_id) references cab_ride(payment_id);
```

6. CAB_RIDE:

```
SQL> create table cab_ride(cab_id int,cust_id int,ride_start_time timestamp,
   ride_end_time timestamp,jfrom varchar(200),
   jto varchar(200),cancelled char(1),payment_id int,charges int);
```

```
 SQL> alter table cab_ride
add constraint fk_cab FOREIGN KEY (cust_id) REFERENCES customer(cust_id);
```

```
SQL> alter table cab_ride
   add constraint fk_cab21 FOREIGN KEY (cab_id) REFERENCES cab(cab_id);
```

```
SQL> alter table cab_ride
 add constraint up UNIQUE(payment_id);
```

7. DISTANCE_MAP:

```
SQL> create table distance_map(jfrom varchar(30),jto varchar(30),dis number);
```

```
SQL> alter table distance_map
   add constraint pk PRIMARY KEY (jfrom,jto);
```

8. CUSTOMER:

```
        SQL> create table customer (cust_id int PRIMARY KEY,pass varchar(10),cust_name
varchar(100),email varchar(100),contact number,vpa varchar(100));
```

9. CAB_RIDE_HISTORY:

```
        SQL> create table cab_ride_history(cab_id int,cust_id int,ride_start_time timestamp,
          ride_end_time timestamp,jfrom varchar(200),
          jto varchar(200),cancelled char(1),payment_id int,charges int);
```

10. DRIVER_CAB

```
        SQL> create table driver_cab(driver_id number PRIMARY KEY,cab_id number);

        SQL> alter table driver_cab
          add constraint fk FOREIGN KEY (driver_id) references driver(driver_id);
```

CAB BOOKING

Description of Tables

APRIL 16, 2019

DBMS PL/SQL

1. SQL> desc **driver**;

```
Name                                    Null?       Type
---------------------------------------- -------- --,--------------------------
DRIVER_ID                               NOT NULL   NUMBER(38)
FIRST_NAME                                         VARCHAR2(128)
LAST_NAME                                          VARCHAR2(128)
BIRTH_DATE                                         DATE
DRIVING_LICENCE_NUMBER                             VARCHAR2(128)
EXPIRY_DATE                                        DATE
WORKING                                            CHAR(1)
```

2. SQL> desc **cab**;

```
Name                                    Null?       Type
---------------------------------------- -------- ----------------------------
CAB_ID                                  NOT NULL   NUMBER(38)
LICENCE_PLATE                                      VARCHAR2(32)
MANUFACTURE_YEAR                                   NUMBER(38)
OWNER_ID                                NOT NULL   NUMBER(38)
MODEL_NAME                                         VARCHAR2(64)
```

3. SQL> desc **owner**;

```
Name                                    Null?       Type
---------------------------------------- -------- ----------------------------
OWNER_ID                                NOT NULL NUMBER(38)
NAME                                               VARCHAR2(100)
```

4. SQL> desc **car_model**;

```
Name                                    Null?      Type
---------------------------------------- -------- ----------------------------
MODEL_NAME                              NOT NULL VARCHAR2(64)
MODEL_DESCRIPTION                                 VARCHAR2(500)
```

5. SQL> desc **driver_cab**;

```
Name                                    Null?      Type
---------------------------------------- -------- ----------------------------
DRIVER_ID                                          NUMBER(38)
CAB_ID                                             NUMBER(38)
```

6. SQL> desc **payment_type**;

```
Name                                    Null?      Type
---------------------------------------- -------- ----------------------------
PAYMENT_ID                              NOT NULL   NUMBER(38)
TYPE_NAME                                          VARCHAR2(128)
```

7. SQL> desc **cab_ride**;

```
Name                                    Null?      Type
---------------------------------------- -------- ----------------------------
CAB_ID                                             NUMBER(38)
CUST_ID                                            NUMBER(38)
RIDE_START_TIME                                    TIMESTAMP(6)
RIDE_END_TIME                                      TIMESTAMP(6)
JFROM                                              VARCHAR2(200)
```

```
JTO                                         VARCHAR2(200)
CANCELLED                                   CHAR(1)
PAYMENT_ID                                  NUMBER(38)
CHARGES                                     NUMBER(38)
```

8. SQL> desc **distance_map**;

```
Name                                   Null?          Type
--------------------------------------- -------- ---------------------------
JFROM    VARCHAR2(30)    JTO    VARCHAR2(30)    DIS
NUMBER
```

9. SQL> desc **customer**;

```
 Name                                  Null?           Type
--------------------------------------- -------- ---------------------------
CUST_ID                                NOT NULL NUMBER(38)
PASS                                            VARCHAR2(10)
CUST_NAME                                        VARCHAR2(100)
EMAIL                                            VARCHAR2(100)
CONTACT                                           NUMBER
VPA                                               VARCHAR2(100)
```

10. SQL> desc **cab_ride_history**;

```
Name                                   Null?          Type
--------------------------------------- -------- ---------------------------
CAB_ID                                          NUMBER(38)
CUST_ID                                         NUMBER(38)
RIDE_START_TIME                                 TIMESTAMP(6)
```

RIDE_END_TIME	TIMESTAMP(6)
JFROM	VARCHAR2(200)
JTO	VARCHAR2(200)
CANCELLED	CHAR(1)
PAYMENT_ID	NUMBER(38)
CHARGES	NUMBER(38)

CAB BOOKING

PROCEDURES

APRIL 16, 2019

DBMS PL/SQL

1. Procedure for Insertion into driver table

```
SQL> create or replace procedure reg_driver(fname driver.first_name%type,lname
driver.last_name%type,bday driver.birth_date%type,lic_no driver.driving_licence_number%type,exp
driver.expiry_date%type,status driver.working%type)
  as
  begin
  insert into driver(first_name,last_name,birth_date,driving_licence_number,expiry_date,working)
values(fname,lname,bday,lic_no,exp,status);
  end;
  /
```

2. Procedure for Insertion into cab table

```
SQL> create or replace procedure reg_cab(plate cab.licence_plate%type,year
cab.manufacture_year%type,owner cab.owner_id%type,mname car_model.model_name%type) as
  begin
  insert into cab(licence_plate,manufacture_year,owner_id,model_name)
values(plate,year,owner,mname);
  end;
  /
```

3. Procedure for Insertion into owner table

```
SQL> create or replace procedure reg_owner(name owner.name%type)
  as
  begin
  insert into owner(name) values(name);
  end;
  /
```

4. Procedure for Insertion into car_model table

```
SQL> create or replace procedure reg_car_model(name car_model.model_name%type,descr
car_model.model_description%type) as
      begin
       insert into car_model values(name,descr);
       end;
       /
```

5. Procedure for Insertion into driver_cab table

```
SQL> create or replace procedure reg_driver_cab(d_id driver_cab.driver_id%type,c_id
driver_cab.cab_id%type)
   as
   begin
   insert into driver_cab values(d_id,c_id);
   end;
   /
```

6. Procedure for Insertion into payment_type table

```
SQL> create or replace procedure reg_payment_type(pid payment_type.payment_id%type,name
payment_type.type_name%type)
   as
   begin
   insert into payment_type values(pid,name);
   end;
   /
```

7. Procedure for Insertion into cab_ride table

```
SQL> create or replace procedure reg_cab_ride(customer_id customer.cust_id%type,password
customer.pass%type,jf cab_ride.jfrom%type,jt cab_ride.jto%type)
  as
  p customer.pass%type;
  cursor c is select pass from customer where cust_id=customer_id;
  begin
  open c;
  fetch c into p;
  close c;
  if(p=password) then
        if(cab_booking(customer_id,jf,jt)=0) then
                dbms_output.put_line('No cab is available right now.. wait for sometimes...');
        end if;
  else
        dbms_output.put_line('Wrong user credentials');
  end if;
/
```

8. Procedure for Insertion into distance_map table

```
SQL> create or replace procedure reg_distance_map(p1
distance_map.jfrom%type,p2 distance_map.jto%type,d distance_map.dis%type)
  as
  begin
  insert into distance_map values(p1,p2,d);
  insert into distance_map values(p2,p1,d);
  end;
  /
```

9. Procedure for Insertion into customer tableSQL> create or replace procedure reg_customer

```
 (
 password customer.pass%type,
 cust_name customer.cust_name%type,
 email customer.email%type,
 contact customer.contact%type,
 vpa customer.vpa%type
 )
 as
begin
insert into customer(pass,cust_name,email,contact,vpa)
values(cust_name,password,email,contact,vpa);
end;
/
```

10. Procedure for Insertion into cab_ride_history table

```
SQL> create or replace procedure reg_cab_ride_history(cbid
  cab_ride.cab_id%type, cust_id cab_ride.cust_id%type,
  st cab_ride.ride_start_time%type,
  et cab_ride.ride_end_time%type,
  jfrom cab_ride.jfrom%type,
  jto cab_ride.jto%type,
  cancelled cab_ride.cancelled%type,
  ptype cab_ride.payment_id%type,
  charges cab_ride.charges%type)
  as
  begin
  insert into cab_ride_history
  values(cbid,cust_id,st,et,jfrom,jto,cancelled,ptype,charges); end;
  /
```

11. Procedure for Removing Driver record.

```
SQL> create or replace procedure rem_driver (id driver.driver_id%type) as
        Begin
        delete from driver where driver_id=id;
        end;
          /
```

12. Procedure for Removing Owner record.

```
SQL> create or replace procedure rem_ Owner (id owner.owner_id%type) as
        Begin
        delete from owner where owner_id=id;
        end;
          /
```

13. Procedure for Removing Cab record.

```
SQL> create or replace procedure rem_cab(id cab.cab_id%type) as
        Begin
        delete from cab where cab_id=id;
        end;
          /
```

14. Procedure for Removing Car_model record.

```
SQL> create or replace procedure rem_car_model(name car_model.model_name%type) as
        Begin
        delete from car_model where
        model_name=name; end;
          /
```

15. Procedure for Removing Customer record.

```
SQL> create or replace procedure rem_customer(id customer.cust_id%type) as
        Begin
        delete from customer where cust_id=id;
        end;
          /
```

16. Procedure for freeing up a particular cab after the ride.

```
SQL> create or replace procedure free_cab(id cab.cab_id%type)
        as Begin
        delete from cab_ride where cab_id=id;
        end;
          /
```

17. Procedure for freeing up all the cabs after the ride.

```
SQL> create or replace procedure free_all_cab as
        Begin
        delete from cab_ride;
        end;
```

CAB BOOKING

FUNCTIONS

APRIL 16, 2019

DBMS PL/SQL

```
SQL> create or replace function cab_booking (customer_id customer.cust_id%type,jf
cab_ride.jfrom%type,jt cab_ride.jto%type) return number as

   z number;

   o number;

   name customer.cust_name%type;

   charge cab_ride.charges%type;

   cid cab.cab_id%type;

   t int;

   cursor c is select cab_id from cab where cab_id not in (select cab_id from cab_ride);

   begin

   open c;

   fetch c into cid;

  select dis into t from distance_map where jfrom=jf and

  jto=jt; charge := t;

  z := 0;

  o := 1;

  t := (t*3)/2;

  if(c%NOTFOUND) then return z;

  else

  insert into cab_ride(cab_id,cust_id,ride_start_time,ride_end_time,jfrom,jto,cancelled,charges)
values(cid,customer_id,sysdate,sysdate + interval '30' minute,jf,jt,'N',charge);

  dbms_output.put_line('Cab booked');

  select cust_name into name from customer where cust_id=customer_id;

  dbms_output.put_line('Customer: '||name);

  dbms_output.put_line('Cab ID: '||cid);

  dbms_output.put_line('From: '||jf);

  dbms_output.put_line('To: '||jt);

  dbms_output.put_line('To pay: '||charge);

  end if;

  close c;

  return o;     end;

  /
```

CAB BOOKING

TRIGGERS

APRIL 16, 2019

DBMS PL/SQL

Driver Triggers:

1. 'Driver' removal will require removing his record from 'driver_cab' table first.

```
SQL> create or replace trigger rem_driver
   before delete on driver
   for each row
   begin
   delete from driver_cab where driver_cab.driver_id=:old.driver_id;
   end;
   /
```

2. Getting driver id while registering the driver.

```
SQL> create or replace trigger bef_reg_driver
before insert on driver
for each row
begin
select drivid.NEXTVAL into :new.driver_id
from dual;
end;
/
```

Cab_ride Triggers:

3. Generating payment id using payid sequence.

```
SQL> create or replace trigger pay_tri
   before insert on cab_ride
   for each row
   begin
   select payid.NEXTVAL into :new.payment_id from dual;
   end;
   /
```

4. Inserting into payment_type table after booking of cab.

```
SQL> create or replace trigger post_cab
   after insert on cab_ride
   for each row
   begin
   insert into payment_type values(:new.payment_id,'&type_name');
   end;
   /
```

5. for Deleting a record from cab_ride table , cab_ride_history should be updated and data should be removed from payment_type table.

```
SQL> create or replace trigger rem_cab_ride
  before delete on cab_ride
   for each row
   declare
   cbid cab_ride.cab_id%type;
   cid cab_ride.cust_id%type;
   s cab_ride.ride_start_time%type;
  e  cab_ride.ride_end_time%type;  j
  cab_ride.jfrom%type;
  t cab_ride.jto%type;
  c cab_ride.cancelled%type;
  pid    cab_ride.payment_id%type;
  charges cab_ride.charges%type;
```

```
begin
cbid := :old.cab_id;
cid := :old.cust_id;
s := :old.ride_start_time;
e := :old.ride_end_time;
j := :old.jfrom;
t := :old.jto;
c := :old.cancelled;
pid := :old.payment_id;
charges := :old.charges;
reg_cab_ride_history(cbid,cid,s,e,j,t,c,pid,charges);
delete from payment_type where payment_id = pid;
end;
/
```

Customer Triggers:

6. Generating customer id using cusid sequence.

```
SQL> create or replace trigger cus_tri
  before insert on customer
  for each row
  begin
  select cusid.NEXTVAL into :new.cust_id
  from dual;
  end;
  /
```

7. Customer removal will require removing his/her record from 'cab_ride' table first .

```
SQL> create or replace trigger rem_customer
   before delete on customer
   for each row
   begin
   delete from cab_ride where cab_ride.cust_id = :old.cust_id;
   end;
   /
```

Cab Triggers:

8. Getting cab id while registering the cab.

```
SQL> create or replace trigger bef_reg_cab
before insert on cab
for each row
begin
select cabid.NEXTVAL into :new.cab_id
from dual;
end;
/
```

9. Removing cab from DB will need to first delete its record from Driver_cab Table.

```
SQL> create or replace trigger rem_cab
   before delete on cab
   for each row
   begin
   delete from driver_cab where driver_cab.cab_id=:old.cab_id;
   end; /
```

Owner Triggers:

10. Getting owner id while registering the owner

```
SQL> create or replace trigger bef_reg_owner before insert on owner
for each row
begin
select ownid.NEXTVAL into :new.owner_id
from dual;
end;   /
```

11. Removing an owner from DB will need to first delete its record from cab Table

```
SQL> create or replace trigger rem_owner
before delete on owner
for each row
begin
delete from cab where cab.owner_id=:old.owner_id;
end;
/
```

Car_model Triggers:

12. Removing car model from DB will need to first delete its record from cab Table

```
SQL> create or replace trigger rem_car_model
   before delete on car_model
   for each row
   begin
   delete from cab where
   cab.model_name=:old.model_name; end; /
```

CAB BOOKING

SEQUENCES

APRIL 16, 2019

DBMS PL/SQL

1. **ID generation for payment_id:**

```
SQL> create sequence payid start with 12321;
```

2. **ID generation for customer_id:**

```
SQL> create sequence cusid start with 32123;
```

3. **ID generation for driver_id:**

```
SQL> create sequence drivid start with 10000;
```

4. **ID generation for cab_id:**

```
SQL> create sequence cabid start with 20000;
```

5. **ID generation for owner_id:**

```
SQL> create sequence ownid start with 30000;
```

CAB BOOKING

Query for Insertion into Tables

APRIL 16, 2019

DBMS PL/SQL

1. Driver:

```
exec reg_driver('kartik','gupta','10-jun-1997','IND10001','15-sep-2019','y');
exec reg_driver('sahil','goel','23-jan-1996','IND10002','20-oct-2021','y');
```

2. Owner:

```
exec reg_owner('Shivam');
exec reg_owner('Mukesh');
```

3. Car_model:

```
exec reg_car_model('Hatchback','Doors Opening
upwards'); exec reg_car_model('Sedan','Four Doors');
```

4. Cab:

```
exec reg_cab('s101',1998,30001,'Sedan');
exec reg_cab('s102',1998,30002,'Hatchback');
exec reg_cab('s103',2013,30001,'Hatchback');
```

5. Customer:

```
exec reg_customer('1234','ashu','ashu@gmail.com',9876543210,'ashu@oksbi');
exec reg_customer('1234','shobhit','shobhit@gmail.com',9874563210,'shobhit@oksbi');
exec reg_customer('14','mohit','mohit@gmail.com',9887563210,'mohit@oksbi');
```

6. Driver_cab;

```
exec reg_driver_cab(10000,20000);
exec reg_driver_cab(10001,20001);
```

7. Distance_map:

```
exec reg_distance_map('gumti','jareeb',5);

exec reg_distance_map('gumti','rawatpur',5);

exec reg_distance_map('jareeb','rawatpur',10);
```