PES UNIVERSITY, BANGALORE

B. TECH- CSE

DBMS - Mini Project

CAR RENTAL MANAGEMENT SYSTEM

Submitted by:-

Name: **SUNDEEP A**

SRN: **PES1UG20CS445**

Roll No: **48**

Section: **H**

Sem: **5**

Table of Contents

Short Description and Scope of the Project

Description

In our system, Customer can rent a car based on make and a model. Our system provides customer to have different pick-up and drop-off locations and will impose late fee if the rental car is returned beyond the return date and time. The Customers can purchase car rental insurance which is optional. Customer can Rent a car of his choice from a list of cars that are Available. Instead of giving different Renting, Delay Fees to each car. We thought to categorize all the cars into a set of 4 different Car Categories namely ECONOMY, FULL SIZED SUV, STANDARD, MINI VAN and all the cars that belong to the same category will have the same Rental / Delay Fees.
While renting a car the customer needs to specify the booking details like from when to when they are going to rent the car. After they return the Car, we are going to enter the actual return date. Based on all these details we are going to calculate the Amount to be paid by the customer using Functions and updating the payment details using procedure and Cursors.

Scope:

1. This technology allows the company to make its services available to the general public while also keeping track of its performance.
2. A **car rental reservation system** can help manage multiple bookings, move between bookings, track different rental statuses, bill distinct bookings precisely, contact specific customers, and much more.
3. making a company available to customers 24 hours a day, seven days a week.
4. Online systems reduce the time it takes to rent a car and the costs of hiring people to input data into paper-based records.
5. Having all the records in one place it is much easier for you to track your expenses and budget appropriately. This can help with financial planning and decision-making for the future of your business.

ER DIAGRAM

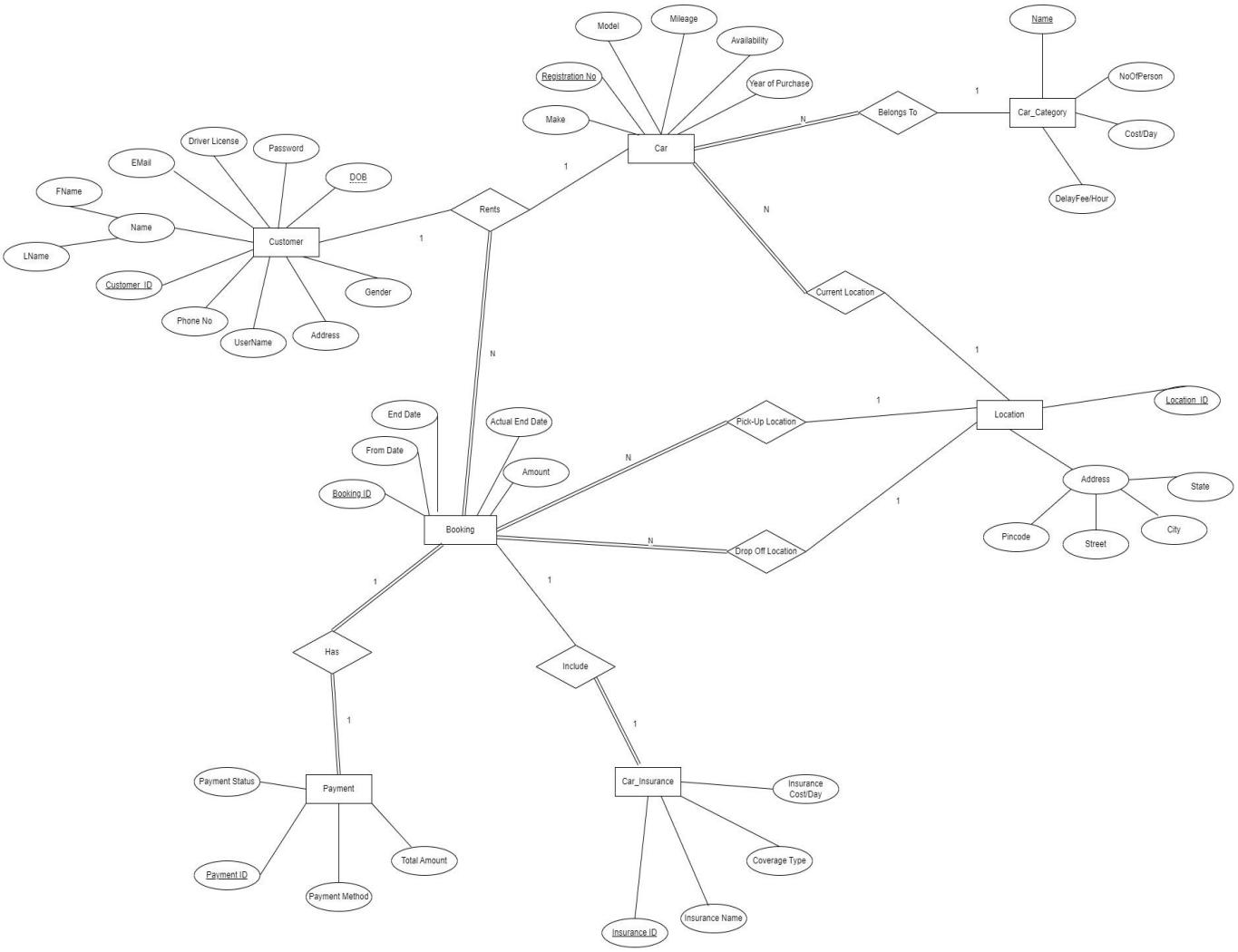


Fig 1: ER Diagram

Relational Schema

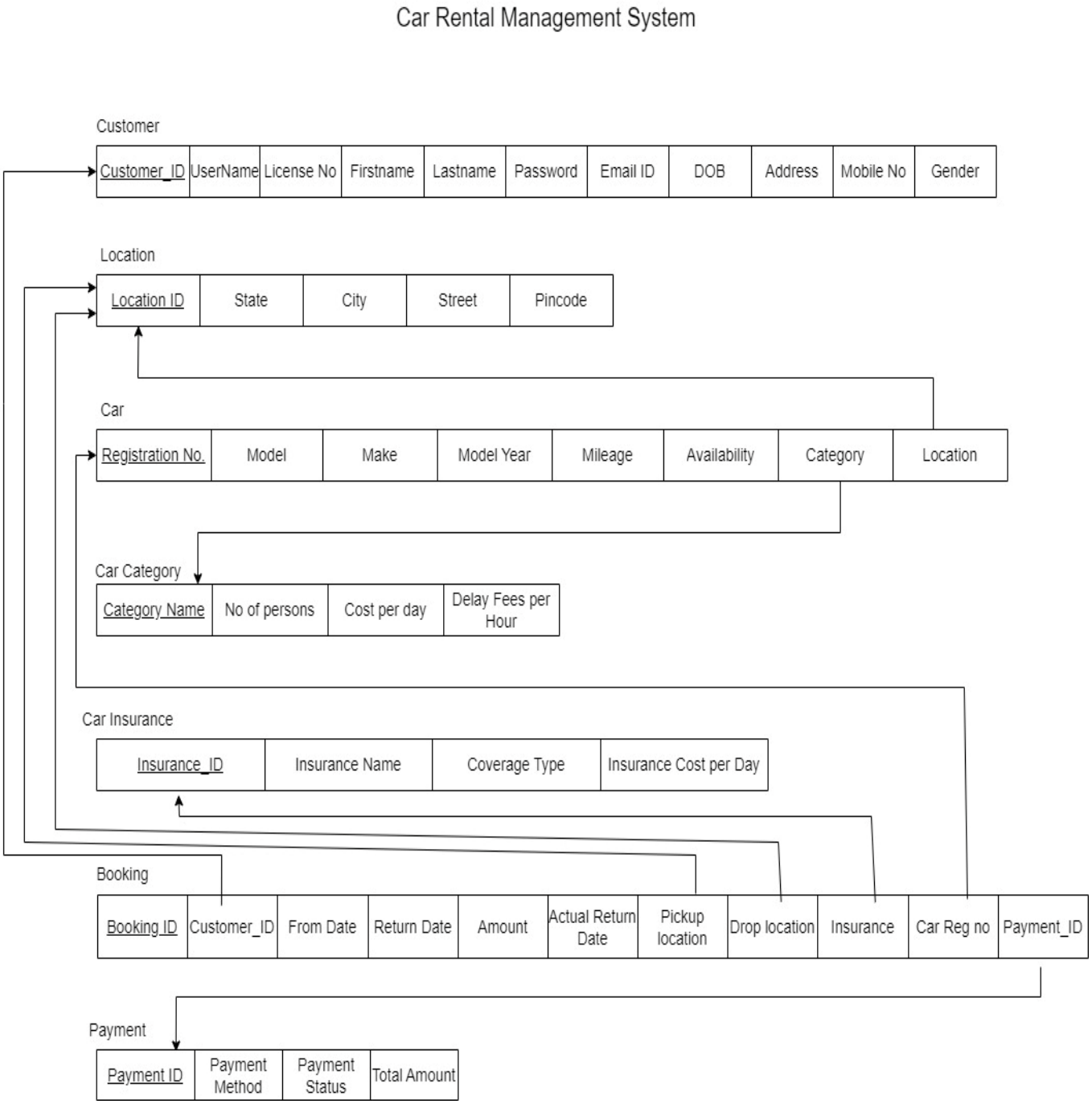


Fig 2: Relation Schema

DDL statements - Building the database

Customer

```
MariaDB [pes1ug20cs445_car_rental_project]> create table Customer(
    ->     Customer_ID int not null auto_increment primary key,
    ->     UserName varchar(35) not null unique,
    ->     LicenseNo varchar(35) not null,
    ->     Firstname varchar(35) not null,
    ->     Lastname varchar(35) not null,
    ->     Password varchar(50) not null,
    ->     email varchar(35),
    ->     DOB date not null,
    ->     Address varchar(50),
    ->     phone_no char(10) not null,
    ->     Gender varchar(10) not null,
    ->     constraint check_phone_customer check(char_length(phone_no) = 10)
    -> );
Query OK, 0 rows affected (0.018 sec)

MariaDB [pes1ug20cs445_car_rental_project]> alter table Customer auto_increment = 5000;
Query OK, 0 rows affected (0.025 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig 3.1 : Structure of Customer Table

Location

```
MariaDB [pes1ug20cs445_car_rental_project]> create table Location(
    ->     Location_ID int not null auto_increment primary key,
    ->     state varchar(35) not null,
    ->     city varchar(35) not null,
    ->     Area varchar(35) not null,
    ->     pincode int(5)
    ->     );
Query OK, 0 rows affected (0.027 sec)

MariaDB [pes1ug20cs445_car_rental_project]> alter table Location auto_increment = 6000;
Query OK, 0 rows affected (0.023 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig 3.2 : Structure of Location Table

Car Category

```
MariaDB [pes1ug20cs445_car_rental_project]> create table car_category(
    ->     category_name varchar(35) not null primary key,
    ->     no_of_persons int not null,
    ->     cost_per_day double not null,
    ->     Delay_Fee_per_hour double not null
    -> );
Query OK, 0 rows affected (0.033 sec)
```

Fig 3.3: Structure of Car Category Table

Car Details

```
MariaDB [pes1ug20cs445_car_rental_project]> create table car_detail(
    ->     Registration_No char(6) not null primary key,
    ->     Model varchar(35) not null,
    ->     Make varchar(35) not null,
    ->     Model_Year int(4) not null,
    ->     Mileage double not null,
    ->     Availability boolean default true,
    ->     Category varchar(35) not null,
    ->     Location int not null,
    ->     foreign key(Category) references car_category(category_name),
    ->     foreign key(Location) references Location(Location_ID)
    -> );
Query OK, 0 rows affected (0.042 sec)
```

Fig 3.4: Structure of Car Category Table

Car Insurance

```
MariaDB [pes1ug20cs445_car_rental_project]> create table Car_Insurance(
    ->     Insurance_ID int not null auto_increment primary key,
    ->     Insurance_Name varchar(35) not null,
    ->     Coverage_Type varchar(35) not null,
    ->     Insurance_Cost_Per_Day double not null
    ->     );
Query OK, 0 rows affected (0.032 sec)

MariaDB [pes1ug20cs445_car_rental_project]> alter table Car_Insurance auto_increment = 7000;
Query OK, 0 rows affected (0.057 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig 3.5: Structure of Car Insurance Table

Payment

```
MariaDB [pes1ug20cs445_car_rental_project]> create table Payment(
    ->     Payment_ID int not null auto_increment primary key,
    ->     Total_Amount double not null,
    ->     Payment_Method varchar(35) not null, -- card, gpay etc..
    ->     Payment_status varchar(30) -- partially paid, fully paid etc..
    -> );
Query OK, 0 rows affected (0.025 sec)

MariaDB [pes1ug20cs445_car_rental_project]> alter table Payment auto_increment=8000;
Query OK, 0 rows affected (0.018 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig 3.6: Structure of Payment Table

Booking Details

```
MariaDB [pes1ug20cs445_car_rental_project]> create table Booking_Details(
    ->     Booking_ID int not null auto_increment primary key,
    ->     Customer_ID int not null,
    ->     From_Date date not null,
    ->     Return_Date datetime not null,
    ->     Amount double,
    ->     Actual_Return_Date datetime not null,
    ->     Pickup_Location int not null,
    ->     Drop_Location int not null,
    ->     Insurance int,
    ->     Car_Reg_No char(6) not null,
    ->     Payment_ID int,
    ->     foreign key(Customer_ID) references Customer(Customer_ID),
    ->     foreign key(Insurance) references Car_Insurance(Insurance_ID),
    ->     foreign key(Pickup_Location) references Location(Location_ID),
    ->     foreign key(Drop_Location) references Location(Location_ID),
    ->     foreign key(Car_Reg_No) references car_detail(Registration_No),
    ->     foreign key(Payment_ID) references Payment(Payment_ID)
    ->     );
Query OK, 0 rows affected (0.079 sec)

MariaDB [pes1ug20cs445_car_rental_project]> alter table Booking_Details auto_increment=9000;
Query OK, 0 rows affected (0.025 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig 3.7 : Structure of Booking Details Table

Populating the Database

Insertion from a CSV File:

Car Category:

```
MariaDB [pes1ug20cs445_car_rental_project]> LOAD DATA INFILE "C:\\Users\\HP\\Desktop\\PESU\\SEM - 5\\PES1UG20CS445\\DBMS\\Project\\Data2.csv" INTO TABLE car_category
    -> FIELDS TERMINATED BY ","
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 4 rows affected, 4 warnings (0.042 sec)
Records: 4  Deleted: 0  Skipped: 0  Warnings: 4
```

Fig 4.1: Inserting into Car Category Using CSV File

Using Insert Operation : Method 1 :

Customer Table :

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into Customer(UserName,LicenseNo,Firstname,Lastname,Password,email,DOB,Address,Phone_no,Gender)
    -> values
    ->     ("123@123","DL-04056789778","H","Hemanth","123@123","Hemanth.123@gmail.com","2000-02-04","Church Street,Bengaluru,India","8322335011","Male"),
    ->     ("124@124","DL-04056789999","A","Revanth","124@124","Revanth@gmail.com","2002-03-23","Church Street,Bengaluru,India","7886870547","Male"),
    ->     ("125@125","DL-04075789999","B","Anil","125@125","Anil@gmail.com","2002-04-08","Jayanagar,Bengaluru,India","9845123645","Male"),
    ->     ("126@126","DL-04075781234","C","Sunitha","126@126","Sunitha@gmail.com","2001-07-21","Majestic,Bengaluru,India","7643123456","Female"),
    ->     ("127@127","DL-79975781234","F","Suman","127@127","Suman@gmail.com","2001-05-22","White Field,Bengaluru,India","2345312672","Male"),
    ->     ("128@128","DL-79975781265","I","Sumantha","128@128","Sumantha@gmail.com","2000-03-01","K.R Puram,Bengaluru,India","9745312768","Female"),
    ->     ("129@129","DL-12398781234","P","Pushpika","129@129","Pushpika@gmail.com","1995-10-10","Kengeri,Bengaluru,India","9623915687","Female"),
    ->     ("130@130","DL-7994531234","O","Teja","130@130","Teja@gmail.com","2003-09-06","Jayanagar,Bengaluru,India","7532167985","Female"),
    ->     ("131@131","DL-77777777777","MS","Dhoni","131@131","Dhoni@gmail.com","19877-02-18","Church Street,Bengaluru,India","6754287906","Male"),
    ->     ("132@132","DL-79975782734","Virat","Kohli","132@132","Kohli@gmai.com","1988-10-18","MG Rd,Bengaluru,India","4538329876","Male");
Query OK, 10 rows affected, 1 warning (0.013 sec)
Records: 10  Duplicates: 0  Warnings: 1
```

Fig 4.2: Inserting Data into Customer Table

Location Table :

```
MariaDB [pes1ug20cs445_car_rental_project]> Insert into Location(state,city,Area,pincode)
    -> values
    ->     ("Karnataka","Bengaluru","Kempegowda International Airport","583101"),
    ->     ("Karnataka","Bengaluru","Pes University","560085"),
    ->     ("Karnataka","Bengaluru","Palace Grounds","560063"),
    ->     ("Karnataka","Bengaluru","KSR Railway Station","560085" ),
    ->     ("Karnataka","Bengaluru","Majestic Bus Station","560034"),
    ->     ("Karnataka","Bengaluru","Church Street","560007");
Query OK, 6 rows affected (0.013 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Fig 4.3: Inserting Data into Location Table

Car Details Table:

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into car_detail(Registration_No,Model,Make,Model_Year,Mileage,Category,Location)
    -> values
    ->     ('DA6523','CIVIC','HONDA',2014,9,'ECONOMY',6003),
    ->     ('FA1252','BOLT','CHEVORLET',2015,7,'ECONOMY',6001),
    ->     ('GQ2146','INSIGHT','HONDA',2016,6.5,'ECONOMY',6002),
    ->     ('VR2341','COROLLA','TOYOTA',2014,12.356,'ECONOMY',6004),
    ->     ('KS1683','TIAGO','TATA',2014,8,'STANDARD',6005),
    ->     ('HNX1890','PRIUS','TOYOTA',2015,7.8,'STANDARD',6002),
    ->     ('UI1289','TRIBER','RENAULT',2017,6,'MINI VAN',6003),
    ->     ('OP9867','ERTIGA','MARUTHI',2018,8,'MINI VAN',6005),
    ->     ('UI7745','INNOVA CRYSTA','TOYATA',2020,8,'FULL SIZE SUV',6004);
Query OK, 9 rows affected, 1 warning (0.044 sec)
Records: 9  Duplicates: 0  Warnings: 1
```

Fig 4.4: Inserting Data into Car Details Table

Car Insurance:

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into Car_Insurance(Insurance_Name,Coverage_Type,Insurance_Cost_Per_Day)
    -> values
    ->     ('Collision Damage Waiver','Bodywork of the Car,Additional parts',3),
    ->     ('Personal Accident','IF you get Injured',2),
    ->     ('Roadside Assistance','IF the car breaks Down',2);
Query OK, 3 rows affected, 1 warning (0.041 sec)
Records: 3  Duplicates: 0  Warnings: 1
```

Fig 4.3: Inserting Data into Car Insurance Table

Payment:

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into Payment(Total_Amount,Payment_Method,Payment_status)
    -> values
    -> (0,"Debit",0),
    -> (1000,"Debit",1),
    -> (2000,"Credit",0),
    -> (1500,"Cash",1),
    -> (10000,"UPI",1);
Query OK, 5 rows affected (0.041 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Fig 4.4: Inserting Data into Payment Table

Booking:

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into Booking_Details(Customer_ID,From_Date,Return_Date,Amount,Actual_Return_Date,Pickup_Location,Drop_Location,Insurance,
Car_Reg_No,Payment_ID)
    -> values
    ->     (5000,"2022-11-07","2022-11-10 07:00",0,"2022-11-11 07:00",6000,6000,7000,"KS1683",8000),
    ->     (5001,"2022-11-08","2022-11-11 08:00",0,"2022-11-11 10:00",6001,6001,7001,"UI7745",8000),
    ->     (5002,"2022-10-07","2022-10-14 15:00",0,"2022-10-15 10:00",6002,6002,7000,"DA6523",8000),
    ->     (5003,"2022-10-20","2022-10-28 17:00",0,"2022-10-28 17:00",6003,6000,7000,"HNX189",8000),
    ->     (5004,"2022-10-29","2022-11-05 10:00",0,"2022-11-05 13:00",6004,6000,7002,"OP9867",8000);
Query OK, 5 rows affected (0.008 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Fig 4.5: Inserting Data into Booking Table

Using Insert Operation : Method 2 :

Inserting Few Extra Rows into Car Details Table

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into car_detail values
    ->     ("RA9867","FIG0","FORD",2018,8,1,"STANDARD",6004),
    ->     ('OU7023','CRUZE','CHEVROLET',2016,7,1,'MINI VAN',6002);
Query OK, 2 rows affected (0.040 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Fig 4.6: Inserting Data Into Car Details Using Variation 2.

Join Queries

Q1. Getting Car details along with its current Location.

Query:

select Registration_No,Model,Area,city,state from car_detail join location WHERE car_detail.Location = location.Location_ID;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Registration_No,Model,Area,city,state from car_detail join location
 WHERE car_detail.Location = location.Location_ID;
+-----------------+---------------+----------------------+-----------+-----------+
| Registration_No | Model         | Area                 | city      | state     |
+-----------------+---------------+----------------------+-----------+-----------+
| FA1252          | BOLT          | Pes University       | Bengaluru | Karnataka |
| GQ2146          | INSIGHT       | Palace Grounds       | Bengaluru | Karnataka |
| HNX189          | PRIUS         | Palace Grounds       | Bengaluru | Karnataka |
| OU7023          | CRUZE         | Palace Grounds       | Bengaluru | Karnataka |
| DA6523          | CIVIC         | KSR Railway Station  | Bengaluru | Karnataka |
| UI1289          | TRIBER        | KSR Railway Station  | Bengaluru | Karnataka |
| RA9867          | FIG0          | Majestic Bus Station | Bengaluru | Karnataka |
| UI7745          | INNOVA CRYSTA | Majestic Bus Station | Bengaluru | Karnataka |
| VR2341          | COROLLA       | Majestic Bus Station | Bengaluru | Karnataka |
| KS1683          | TIAGO         | Church Street        | Bengaluru | Karnataka |
| OP9867          | ERTIGA        | Church Street        | Bengaluru | Karnataka |
+-----------------+---------------+----------------------+-----------+-----------+
11 rows in set (0.002 sec)
```

Fig 5.1: Car Details along with its current Location

Q2. Display the First Name and Last Name of the User Who have done a booking.

Query :

select Firstname,Lastname from customer join booking_details where customer.Customer_ID = booking_details.Customer_ID;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Firstname,Lastname from customer join booking_details where
customer.Customer_ID = booking_details.Customer_ID;
+-----------+----------+
| Firstname | Lastname |
+-----------+----------+
| H         | Hemanth  |
| A         | Revanth  |
| B         | Anil     |
| C         | Sunitha  |
| F         | Suman    |
+-----------+----------+
5 rows in set (0.001 sec)
```

Fig 5.2: Displaying the FirstName and LastName of the Users who have done a Booking

Q3. Display Car Details along with along with the category to which it belongs, No of persons it can hold, Cost per day

Query :

select Model,Make,Category,cost_per_day,no_of_persons from car_detail join car_category WHERE car_detail.Category = car_category.category_name;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Model,Make,Category,cost_per_day,no_of_persons from
car_detail join car_category WHERE car_detail.Category = car_category.category_name;
+---------------+-----------+---------------+--------------+---------------+
| Model         | Make      | Category      | cost_per_day | no_of_persons |
+---------------+-----------+---------------+--------------+---------------+
| CIVIC         | HONDA     | ECONOMY       |           30 |             5 |
| BOLT          | CHEVORLET | ECONOMY       |           30 |             5 |
| INSIGHT       | HONDA     | ECONOMY       |           30 |             5 |
| COROLLA       | TOYOTA    | ECONOMY       |           30 |             5 |
| INNOVA CRYSTA | TOYATA    | FULL SIZE SUV |           60 |             8 |
| ERTIGA        | MARUTHI   | MINI VAN      |           70 |             7 |
| CRUZE         | CHEVROLET | MINI VAN      |           70 |             7 |
| TRIBER        | RENAULT   | MINI VAN      |           70 |             7 |
| PRIUS         | TOYOTA    | STANDARD      |           38 |             5 |
| TIAGO         | TATA      | STANDARD      |           38 |             5 |
| FIG0          | FORD      | STANDARD      |           38 |             5 |
+---------------+-----------+---------------+--------------+---------------+
11 rows in set (0.001 sec)
```

Fig 5.3: Display Car Details along with along with the category to which it belongs, No of persons it can hold, Cost per day

Q4. Display the cars that have been selected by a customer.

Query :

select Firstname,Lastname,Make,Model,Category from customer join (booking_details join car_detail) where customer.Customer_ID = booking_details.Customer_ID and Registration_No = Car_Reg_No;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Firstname,Lastname,Make,Model,Category from custome
r join (booking_details join car_detail) where customer.Customer_ID = booking_details.Customer_ID and
Registration_No = Car_Reg_No;
+-----------+----------+---------+---------------+---------------+
| Firstname | Lastname | Make    | Model         | Category      |
+-----------+----------+---------+---------------+---------------+
| H         | Hemanth  | TATA    | TIAGO         | STANDARD      |
| A         | Revanth  | TOYATA  | INNOVA CRYSTA | FULL SIZE SUV |
| B         | Anil     | HONDA   | CIVIC         | ECONOMY       |
| C         | Sunitha  | TOYOTA  | PRIUS         | STANDARD      |
| F         | Suman    | MARUTHI | ERTIGA        | MINI VAN      |
+-----------+----------+---------+---------------+---------------+
5 rows in set (0.001 sec)
```

Fig 5.4: Display the cars that have been selected by a customer.

Aggregate Functions

Q1. Calculate the number of cars per each category.

Query:

select category,count(Registration_No) from car_detail group by category;

```
MariaDB [pes1ug20cs445_car_rental_project]> select category,count(Registration_No) from car_detail
group by category;
+---------------+------------------------+
| category      | count(Registration_No) |
+---------------+------------------------+
| ECONOMY       |                      4 |
| FULL SIZE SUV |                      1 |
| MINI VAN      |                      3 |
| STANDARD      |                      3 |
+---------------+------------------------+
4 rows in set (0.001 sec)
```

Fig 6.1: Number of cars per each Category

Q2. Calculate the Number of people that are from the same Area.

Query :

select Address, count(Firstname) from customer group by Address;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Address, count(Firstname) from customer group
by Address;
+------------------------------+------------------+
| Address                      | count(Firstname) |
+------------------------------+------------------+
| Church Street,Bengaluru,India |                3 |
| Jayanagar,Bengaluru,India     |                2 |
| K.R Puram,Bengaluru,India     |                1 |
| Kengeri,Bengaluru,India       |                1 |
| Majestic,Bengaluru,India      |                1 |
| MG Rd,Bengaluru,India         |                1 |
| White Field,Bengaluru,India   |                1 |
+------------------------------+------------------+
7 rows in set (0.001 sec)
```

Fig 6.2: Calculate the Number of people that are from the same Area

Q3. Average Amount per Transaction:

Query :

select AVG(Total_Amount) from payment;

```
MariaDB [pes1ug20cs445_car_rental_project]> select AVG(Total_Amount) from payment;
+-------------------+
| AVG(Total_Amount) |
+-------------------+
|              2900 |
+-------------------+
1 row in set (0.001 sec)
```

Fig 6.3: Average Amount per Transaction

Q4 . Number of car of a particular Model_year

Query :

select Model_Year, count(Registration_No) from car_detail group by Model_Year;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Model_Year, count(Registration_No) from
car_detail group by Model_Year;
+------------+-----------------------+
| Model_Year | count(Registration_No) |
+------------+-----------------------+
|       2014 |                     3 |
|       2015 |                     2 |
|       2016 |                     2 |
|       2017 |                     1 |
|       2018 |                     2 |
|       2020 |                     1 |
+------------+-----------------------+
6 rows in set (0.001 sec)
```

Fig 6.4: Number of Cars of a Particular Model Year.

Set Operations

Q1. Display the Customer First name and Last Name who have returned the Car on or before the Return Date

Query :

select Firstname,Lastname from customer join booking_details where customer.Customer_ID = booking_details.Customer_ID and Return_Date = Actual_Return_Date

UNION

select Firstname,Lastname from customer join booking_details where customer.Customer_ID = booking_details.Customer_ID and Return_Date > Actual_Return_Date;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Firstname,Lastname from customer join booking_details where
customer.Customer_ID = booking_details.Customer_ID and Return_Date = Actual_Return_Date
    -> UNION
    -> select Firstname,Lastname from customer join booking_details where customer.Customer_ID = booking_detail
s.Customer_ID and Return_Date > Actual_Return_Date;
+-----------+----------+
| Firstname | Lastname |
+-----------+----------+
| C         | Sunitha  |
+-----------+----------+
1 row in set (0.002 sec)
```

Fig 7.1: Display the Customer First name and Last Name who have returned the Car on or before the Return Date

Q2. Display the First Name and Last Name of the Customer who have booked a car and are Male.

Query:

select Firstname,Lastname from customer where Gender = "Male"

INTERSECT

select Firstname,Lastname from customer join booking_details where customer.Customer_ID = booking_details.Customer_ID;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Firstname,Lastname from customer where Gender = "Male"
    -> INTERSECT
    -> select Firstname,Lastname from customer join booking_details where customer.Customer_ID = booking_detail
s.Customer_ID;
+-----------+----------+
| Firstname | Lastname |
+-----------+----------+
| H         | Hemanth  |
| A         | Revanth  |
| B         | Anil     |
| F         | Suman    |
+-----------+----------+
4 rows in set (0.001 sec)
```

Fig 7.2: Display the First Name and Last Name of the Customer who have booked a car and are Male.

Q3. Display the Customer Firstname and Last name where the Amount paid >1250 and is not Male

Query:

select Firstname,Lastname from customer join (booking_details join payment) where customer.Customer_ID = booking_details.Customer_ID and booking_details.Payment_ID = payment.Payment_ID and Total_Amount > 1250

EXCEPT

select Firstname,Lastname from customer where Gender="Male";

```
MariaDB [pes1ug20cs445_car_rental_project]> select Firstname,Lastname from customer join (booking_details join
payment) where customer.Customer_ID = booking_details.Customer_ID and booking_details.Payment_ID = payment.Paym
ent_ID and Total_Amount > 1250
    -> EXCEPT
    -> select Firstname,Lastname from customer where Gender="Male";
+-----------+----------+
| Firstname | Lastname |
+-----------+----------+
| C         | Sunitha  |
+-----------+----------+
1 row in set (0.001 sec)
```

Fig 7.3: Display the Customer Firstname and Last name where the Amount paid >1250 and is not Male

Q4. Display the Car that were Booked and the Model_Year >2017

Query :

select Make,Model,Model_Year from car_detail join booking_details where Registration_No = Car_Reg_No

EXCEPT

select Make,Model,Model_Year from car_detail where Model_Year <= 2017;

```
MariaDB [pes1ug20cs445_car_rental_project]> select Make,Model,Model_Year from car_detail join booking_details
where Registration_No = Car_Reg_No
    -> EXCEPT
    -> select Make,Model,Model_Year from car_detail where Model_Year <= 2017;
+---------+--------------+------------+
| Make    | Model        | Model_Year |
+---------+--------------+------------+
| MARUTHI | ERTIGA       |       2018 |
| TOYATA  | INNOVA CRYSTA |       2020 |
+---------+--------------+------------+
2 rows in set (0.004 sec)
```

Fig 7.4: Display the Cars that were booked and their Model_Year > 2017

Functions and Procedures

Function:

To calculate the Amount that has to be Paid by a customer based on the Number of Days a car has been Booked and if there is a delay in returning the car.

```
MariaDB [pes1ug20cs445_car_rental_project]> DELIMITER $$
MariaDB [pes1ug20cs445_car_rental_project]> CREATE FUNCTION Total_Cost(From_date DATE,Return_date DATETIME,Actual_Return DATETIME,Category VARCHAR(35))
    -> RETURNS float(5)
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE Booking_Days INT(5);
    -> DECLARE Extra_Hours int(5);
    -> DECLARE Total float(5);
    -> SET Booking_Days = DATEDIFF(Return_date,From_date);
    -> SET Extra_Hours = HOUR(TIMEDIFF(Actual_Return,Return_date));
    -> IF category = "ECONOMY" THEN
    -> SET Total = 3000*Booking_Days + 250*Extra_Hours;
    -> ELSEIF category = "FULL SIZE SUV" THEN
    -> SET Total = 6000*Booking_Days + 450*Extra_Hours;
    -> ELSEIF category = "MINI VAN" THEN
    -> SET Total = 7000*Booking_Days + 550*Extra_Hours;
    -> ELSEIF category = "STANDARD" THEN
    -> SET Total = 4000*Booking_Days + 300*Extra_Hours;
    -> ELSE
    -> SET Total=0;
    -> END IF;
    -> RETURN Total;
    -> END; $$
Query OK, 0 rows affected (0.043 sec)

MariaDB [pes1ug20cs445_car_rental_project]>
MariaDB [pes1ug20cs445_car_rental_project]> DELIMITER ;
MariaDB [pes1ug20cs445_car_rental_project]>
```

Fig 8.1: Creating a Function to Calculate the amount that has to be paid by each customer.

Function Call

Creating a View to Store the result from the Function so that we can update the Amount in Booking Details Table.

```
MariaDB [pes1ug20cs445_car_rental_project]> create view BookingAmount as select b.Booking_ID as Booking_ID,Total_Cost(b.From_Date,b.Return_Date,b.Actual_Return_Date,cat.cat
egory_name) as Amount
    -> from (( booking_details as b inner join car_detail as car on b.Car_Reg_No = car.Registration_No )
    -> inner join car_category as cat on car.Category = cat.category_name);
Query OK, 0 rows affected (0.043 sec)

MariaDB [pes1ug20cs445_car_rental_project]> select * from BookingAmount;
+------------+--------+
| Booking_ID | Amount |
+------------+--------+
|       9005 |  19200 |
|       9006 |  18900 |
|       9007 |  25750 |
|       9008 |  32000 |
|       9009 |  50650 |
+------------+--------+
5 rows in set (0.002 sec)
```

Fig 8.2: Creating a View to store the Result that has been generated by the function, so that it can be Later used to update the Booking Details Table.

Updated the Amount Field in Booking Details Table :

```
MariaDB [pes1ug20cs445_car_rental_project]> UPDATE booking_details AS b INNER JOIN BookingAmount AS b1 ON b.Booking_ID = b1.Booking_ID SET b.Amount = b1.Amount;
Query OK, 5 rows affected (0.042 sec)
Rows matched: 5  Changed: 5  Warnings: 0

MariaDB [pes1ug20cs445_car_rental_project]> select * from booking_details;
+------------+------------+------------+---------------------+--------+---------------------+-----------------+---------------+-----------+------------+------------+
| Booking_ID | Customer_ID | From_Date  | Return_Date         | Amount | Actual_Return_Date  | Pickup_Location | Drop_Location | Insurance | Car_Reg_No | Payment_ID |
+------------+------------+------------+---------------------+--------+---------------------+-----------------+---------------+-----------+------------+------------+
|       9005 |       5000 | 2022-11-07 | 2022-11-10 07:00:00 |  19200 | 2022-11-11 07:00:00 |            6000 |          6000 |      7000 | KS1683     |       8000 |
|       9006 |       5001 | 2022-11-08 | 2022-11-11 08:00:00 |  18900 | 2022-11-11 10:00:00 |            6001 |          6001 |      7001 | UI7745     |       8001 |
|       9007 |       5002 | 2022-10-07 | 2022-10-14 15:00:00 |  25750 | 2022-10-15 10:00:00 |            6002 |          6002 |      7000 | DA6523     |       8002 |
|       9008 |       5003 | 2022-10-20 | 2022-10-28 17:00:00 |  32000 | 2022-10-28 17:00:00 |            6003 |          6000 |      7000 | HNX189     |       8003 |
|       9009 |       5004 | 2022-10-29 | 2022-11-05 10:00:00 |  50650 | 2022-11-05 13:00:00 |            6004 |          6000 |      7002 | OP9867     |       8004 |
+------------+------------+------------+---------------------+--------+---------------------+-----------------+---------------+-----------+------------+------------+
5 rows in set (0.000 sec)
```

Fig 8.3: Updating the Amount in Booking Details Table

Procedure:

Viewing the Values in Booking_details table :

```
MariaDB [pes1ug20cs445_car_rental_project]> select Booking_ID,Payment_ID,Amount from booking_details;
+------------+------------+--------+
| Booking_ID | Payment_ID | Amount |
+------------+------------+--------+
|       9005 |       8000 |  19200 |
|       9006 |       8001 |  18900 |
|       9007 |       8002 |  25750 |
|       9008 |       8003 |  32000 |
|       9009 |       8004 |  50650 |
|       9013 |       8005 |  42000 |
+------------+------------+--------+
6 rows in set (0.000 sec)
```

Fig 8.4: Viewing the Amount Details corresponding to Respective Booking IDs and Payment IDs

Initial Values in Payment Table:

```
+------------+--------------+----------------+
| Payment_ID | Total_Amount | Payment_Method |
+------------+--------------+----------------+
|       8000 |            0 | Debit          |
|       8001 |            0 | Debit          |
|       8002 |            0 | Credit         |
|       8003 |            0 | Cash           |
|       8004 |            0 | UPI            |
|       8005 |            0 | Credit         |
+------------+--------------+----------------+
6 rows in set (0.000 sec)
```

Fig 8.5: Initial Values in Payment Table

Creating Procedure:

```
MariaDB [pes1ug20cs445_car_rental_project]> DELIMITER $$
MariaDB [pes1ug20cs445_car_rental_project]> CREATE PROCEDURE Update_Payment()
    -> BEGIN
    -> DECLARE done INT DEFAULT 0;
    -> DECLARE Amt double;
    -> DECLARE pay_id int(11);
    -> DECLARE booking_cursor CURSOR FOR SELECT Amount,Payment_ID FROM booking_details;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    -> OPEN booking_cursor;
    -> label: LOOP
    -> FETCH booking_cursor INTO Amt,pay_id;
    -> UPDATE Payment set Total_Amount = Amt where Payment_ID = pay_id;
    -> IF done = 1 THEN LEAVE label;
    -> END IF;
    -> END LOOP;
    -> CLOSE booking_cursor;
    -> END $$
Query OK, 0 rows affected (0.044 sec)

MariaDB [pes1ug20cs445_car_rental_project]>
MariaDB [pes1ug20cs445_car_rental_project]> DELIMITER ;
```

Fig 8.6: Creating a Procedure to Update the Payment info in the Payment Table

Procedure Call:

```
MariaDB [pes1ug20cs445_car_rental_project]> call Update_Payment();
Query OK, 6 rows affected (0.021 sec)
```

Fig 8.7: Calling the Procedure to Update the Payment Amount in Payment Table

Values in Payment Table After Procedure Call:

Payment_ID	Total_Amount	Payment_Method
8000	19200	Debit
8001	18900	Debit
8002	25750	Credit
8003	32000	Cash
8004	50650	UPI
8005	42000	Credit

6 rows in set (0.000 sec)

Fig 8.8: Values in the Payment Table After Procedure Call

Triggers and Cursors

Triggers:

Set the availability of a particular car to 0 when that car is booked.

Before Creating Trigger:

```
MariaDB [pes1ug20cs445_car_rental_project]> select * from car_detail;
+-----------------+---------------+-----------+------------+---------+--------------+---------------+----------+
| Registration_No | Model         | Make      | Model_Year | Mileage | Availability | Category      | Location |
+-----------------+---------------+-----------+------------+---------+--------------+---------------+----------+
| DA6523          | CIVIC         | HONDA     |       2014 |       9 |            1 | ECONOMY       |     6003 |
| FA1252          | BOLT          | CHEVORLET |       2015 |       7 |            1 | ECONOMY       |     6001 |
| GQ2146          | INSIGHT       | HONDA     |       2016 |     6.5 |            1 | ECONOMY       |     6002 |
| HNX189          | PRIUS         | TOYOTA    |       2015 |     7.8 |            1 | STANDARD      |     6002 |
| KS1683          | TIAGO         | TATA      |       2014 |       8 |            1 | STANDARD      |     6005 |
| OP9867          | ERTIGA        | MARUTHI   |       2018 |       8 |            1 | MINI VAN      |     6005 |
| OU7023          | CRUZE         | CHEVROLET |       2016 |       7 |            1 | MINI VAN      |     6002 |
| RA9867          | FIG0          | FORD      |       2018 |       8 |            1 | STANDARD      |     6004 |
| UI1289          | TRIBER        | RENAULT   |       2017 |       6 |            1 | MINI VAN      |     6003 |
| UI7745          | INNOVA CRYSTA | TOYATA    |       2020 |       8 |            1 | FULL SIZE SUV |     6004 |
| VR2341          | COROLLA       | TOYOTA    |       2014 |  12.356 |            1 | ECONOMY       |     6004 |
+-----------------+---------------+-----------+------------+---------+--------------+---------------+----------+
11 rows in set (0.001 sec)
```

Fig 9.1: Initial Values in the Car Details Table

Inserting the Trigger:

```
MariaDB [pes1ug20cs445_car_rental_project]> drop trigger if exists booking_done;
Query OK, 0 rows affected, 1 warning (0.000 sec)

MariaDB [pes1ug20cs445_car_rental_project]> delimiter $$
MariaDB [pes1ug20cs445_car_rental_project]> create trigger booking_done
    -> after insert
    -> on booking_details for each row
    -> begin
    ->
    -> update car_detail set Availability =0 where Registration_No = new.Car_Reg_No;
    ->
    -> end $$
Query OK, 0 rows affected (0.053 sec)

MariaDB [pes1ug20cs445_car_rental_project]> delimiter ;
MariaDB [pes1ug20cs445_car_rental_project]>
```

Fig 9.2: Creating a Trigger to set the Availability of a Particular car to 0 When It is booked.

Inserting data into Booking Details Table:

```
MariaDB [pes1ug20cs445_car_rental_project]> insert into Booking_Details(Customer_ID,From_Date,Return_Date,Amount,
Actual_Return_Date,Pickup_Location,Drop_Location,Insurance,Car_Reg_No,Payment_ID)
    -> values
    -> (5005,"2022-11-14","2022-11-20 9:00",0,"2022-11-20 9:00",6000,6003,7001,"UI1289",8000);
Query OK, 1 row affected (0.044 sec)
```

Fig 9.3: Inserting Data into Booking Details to check if Trigger Works Properly.

Checking Car Details after Inserting a new Booking Detail:

```
MariaDB [pes1ug20cs445_car_rental_project]> select * from car_detail;
+-----------------+---------------+-----------+------------+---------+--------------+---------------+----------+
| Registration_No | Model         | Make      | Model_Year | Mileage | Availability | Category      | Location |
+-----------------+---------------+-----------+------------+---------+--------------+---------------+----------+
| DA6523          | CIVIC         | HONDA     |       2014 |       9 |            1 | ECONOMY       |     6003 |
| FA1252          | BOLT          | CHEVORLET |       2015 |       7 |            1 | ECONOMY       |     6001 |
| GQ2146          | INSIGHT       | HONDA     |       2016 |     6.5 |            1 | ECONOMY       |     6002 |
| HNX189          | PRIUS         | TOYOTA    |       2015 |     7.8 |            1 | STANDARD      |     6002 |
| KS1683          | TIAGO         | TATA      |       2014 |       8 |            1 | STANDARD      |     6005 |
| OP9867          | ERTIGA        | MARUTHI   |       2018 |       8 |            1 | MINI VAN      |     6005 |
| OU7023          | CRUZE         | CHEVROLET |       2016 |       7 |            1 | MINI VAN      |     6002 |
| RA9867          | FIGO          | FORD      |       2018 |       8 |            1 | STANDARD      |     6004 |
| UI1289          | TRIBER        | RENAULT   |       2017 |       6 |            0 | MINI VAN      |     6003 |
| UI7745          | INNOVA CRYSTA | TOYATA    |       2020 |       8 |            1 | FULL SIZE SUV |     6004 |
| VR2341          | COROLLA       | TOYOTA    |       2014 |  12.356 |            1 | ECONOMY       |     6004 |
+-----------------+---------------+-----------+------------+---------+--------------+---------------+----------+
11 rows in set (0.000 sec)
```

Fig 9.4: Checking the Car Details to Check if the Availability of a Booked Car has been made 0 or not.

Cursors:

Viewing the Values in Booking_details table :

```
MariaDB [pes1ug20cs445_car_rental_project]> select Booking_ID,Payment_ID,Amount from booking_details;
+------------+------------+--------+
| Booking_ID | Payment_ID | Amount |
+------------+------------+--------+
|       9005 |       8000 |  19200 |
|       9006 |       8001 |  18900 |
|       9007 |       8002 |  25750 |
|       9008 |       8003 |  32000 |
|       9009 |       8004 |  50650 |
|       9013 |       8005 |  42000 |
+------------+------------+--------+
6 rows in set (0.000 sec)
```

Fig 9.5: Viewing the Amount Details corresponding to Respective Booking IDs and Payment IDs

Initial Values in Payment Table:

```
+------------+--------------+----------------+
| Payment_ID | Total_Amount | Payment_Method |
+------------+--------------+----------------+
|       8000 |            0 | Debit          |
|       8001 |            0 | Debit          |
|       8002 |            0 | Credit         |
|       8003 |            0 | Cash           |
|       8004 |            0 | UPI            |
|       8005 |            0 | Credit         |
+------------+--------------+----------------+
6 rows in set (0.000 sec)
```

Fig 9.6: Initial Values in Payment Table

Creating Cursor:

```
MariaDB [pes1ug20cs445_car_rental_project]> DELIMITER $$
MariaDB [pes1ug20cs445_car_rental_project]> CREATE PROCEDURE Update_Payment()
    -> BEGIN
    -> DECLARE done INT DEFAULT 0;
    -> DECLARE Amt double;
    -> DECLARE pay_id int(11);
    -> DECLARE booking_cursor CURSOR FOR SELECT Amount,Payment_ID FROM booking_details;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    -> OPEN booking_cursor;
    -> label: LOOP
    -> FETCH booking_cursor INTO Amt,pay_id;
    -> UPDATE Payment set Total_Amount = Amt where Payment_ID = pay_id;
    -> IF done = 1 THEN LEAVE label;
    -> END IF;
    -> END LOOP;
    -> CLOSE booking_cursor;
    -> END $$
Query OK, 0 rows affected (0.044 sec)

MariaDB [pes1ug20cs445_car_rental_project]>
MariaDB [pes1ug20cs445_car_rental_project]> DELIMITER ;
```

Fig 9.7: Creating Cursor to Update the Payment Details in the Payment Table.

Implementing the Cursor:

```
MariaDB [pes1ug20cs445_car_rental_project]> call Update_Payment();
Query OK, 6 rows affected (0.021 sec)
```

Fig 9.8: Implementing the Cursor

Values in Payment Table After Using Cursor:

Payment_ID	Total_Amount	Payment_Method
8000	19200	Debit
8001	18900	Debit
8002	25750	Credit
8003	32000	Cash
8004	50650	UPI
8005	42000	Credit

6 rows in set (0.000 sec)

Fig 9.9: Payment Values in Payment Table After Calling the Cursor.

Developing a Frontend

CREATE:

Select on which Table You want to Perform Create Operation:

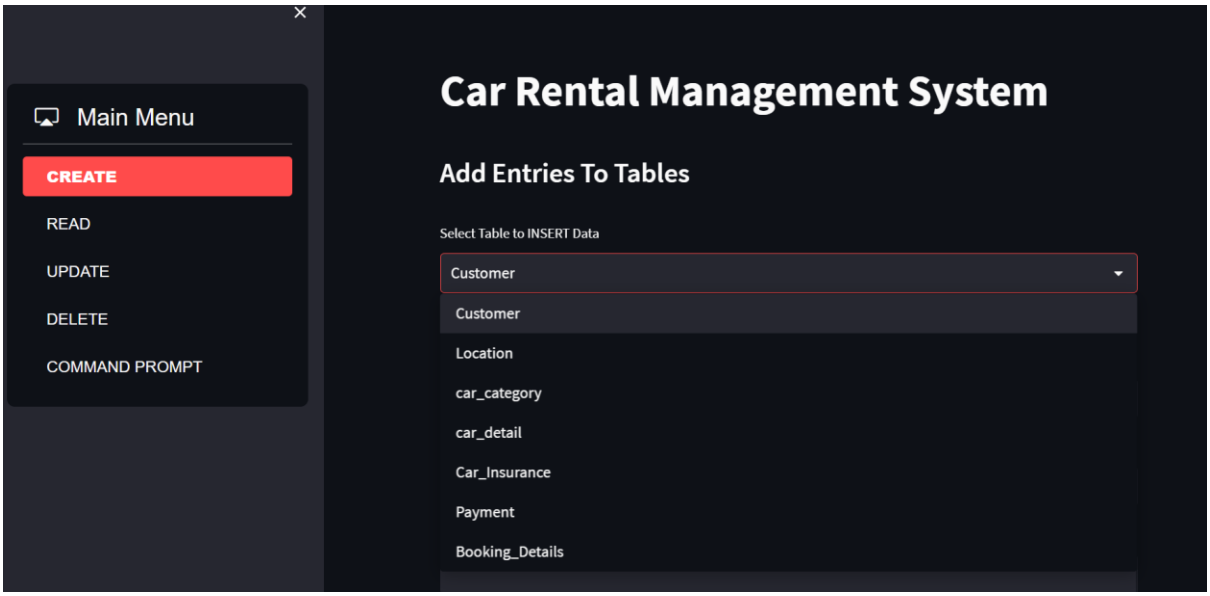


Fig 10.1: You can Select Table of your choice to Perform any Operation

Let's Select Location Table:

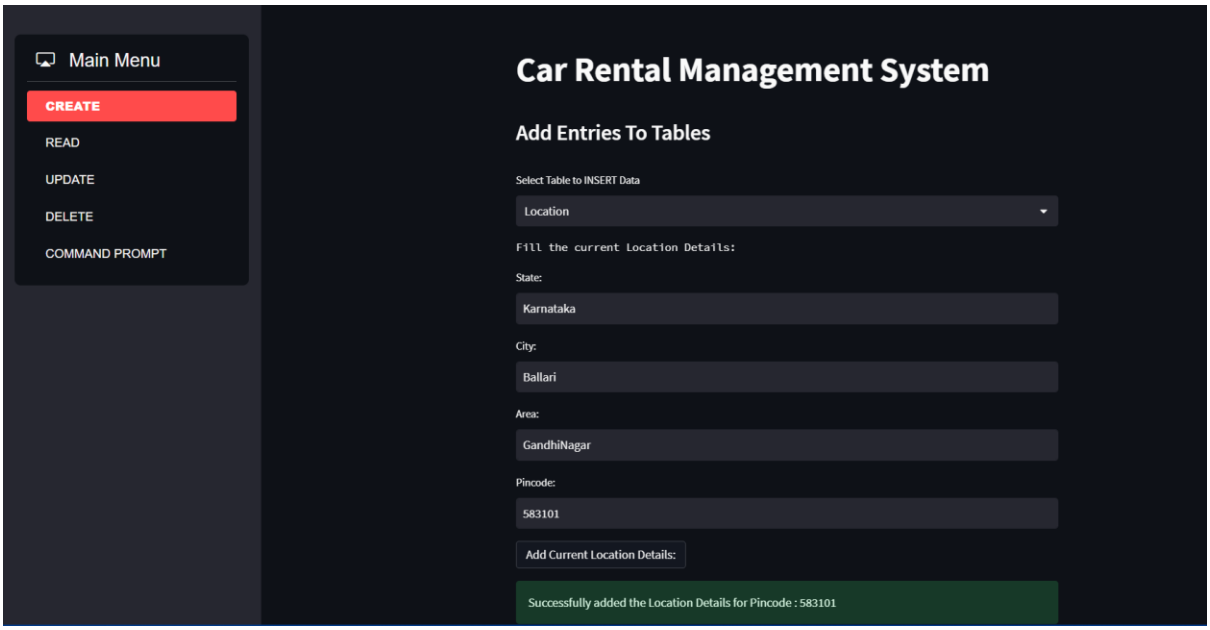


Fig 10.2: Creating a new Record in Location Table

View/Read:

You can select from which table you want to display all its contents:

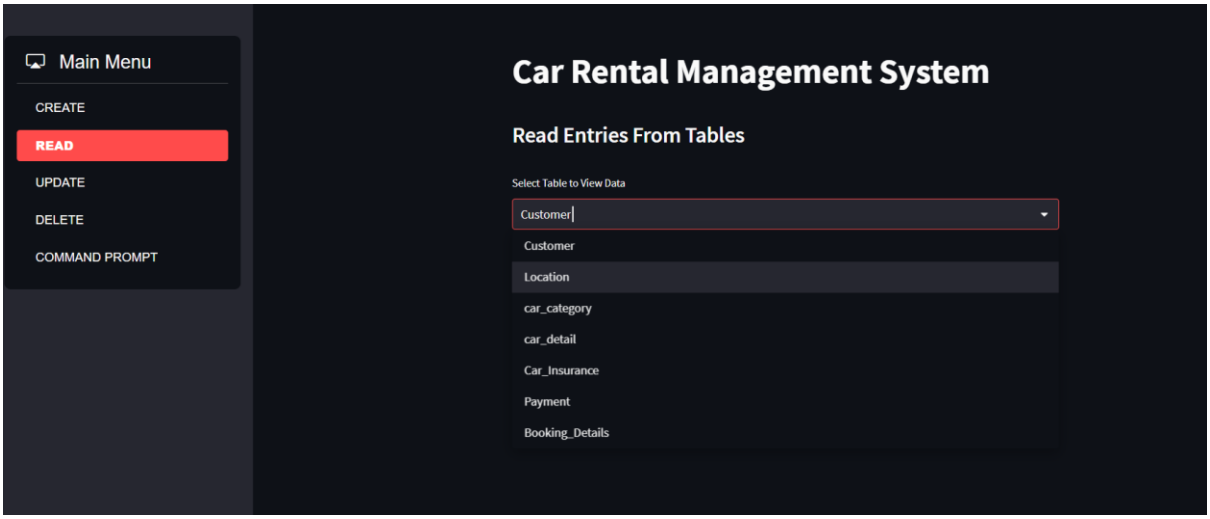
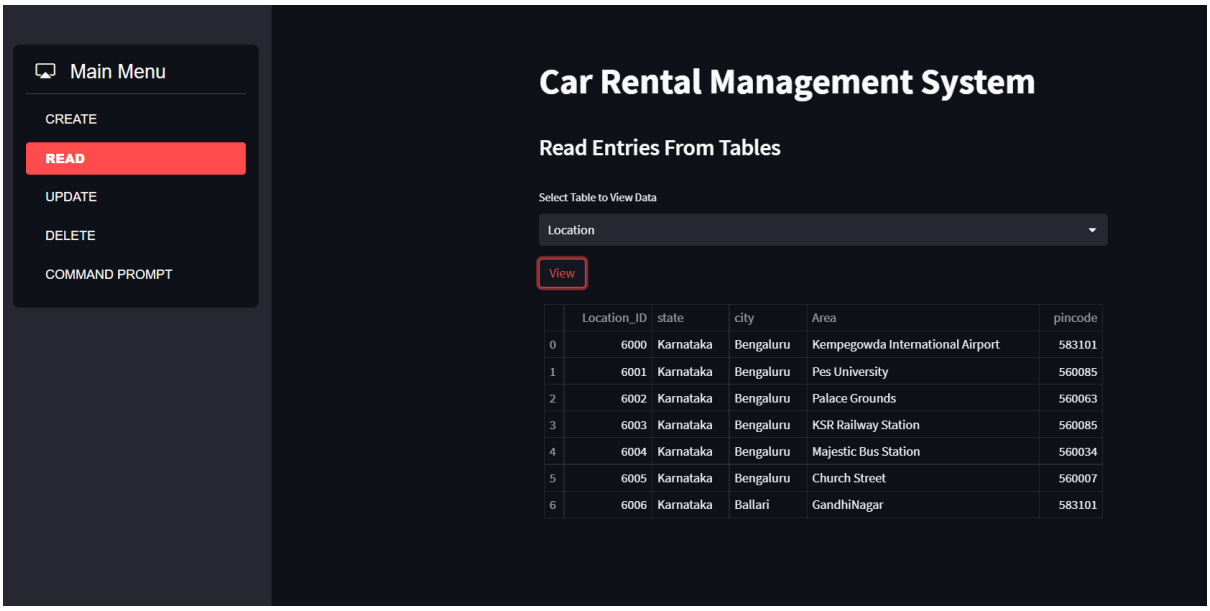


Fig 10.3: View Data from Table of your Choice

Lets check if the new Location will be displayed:



	Location_ID	state	city	Area	pincode
0	6000	Karnataka	Bengaluru	Kempegowda International Airport	583101
1	6001	Karnataka	Bengaluru	Pes University	560085
2	6002	Karnataka	Bengaluru	Palace Grounds	560063
3	6003	Karnataka	Bengaluru	KSR Railway Station	560085
4	6004	Karnataka	Bengaluru	Majestic Bus Station	560034
5	6005	Karnataka	Bengaluru	Church Street	560007
6	6006	Karnataka	Ballari	GandhiNagar	583101

Fig 10.4: On viewing the Location Table we can see that the new Record has been Successfully added.

Update:

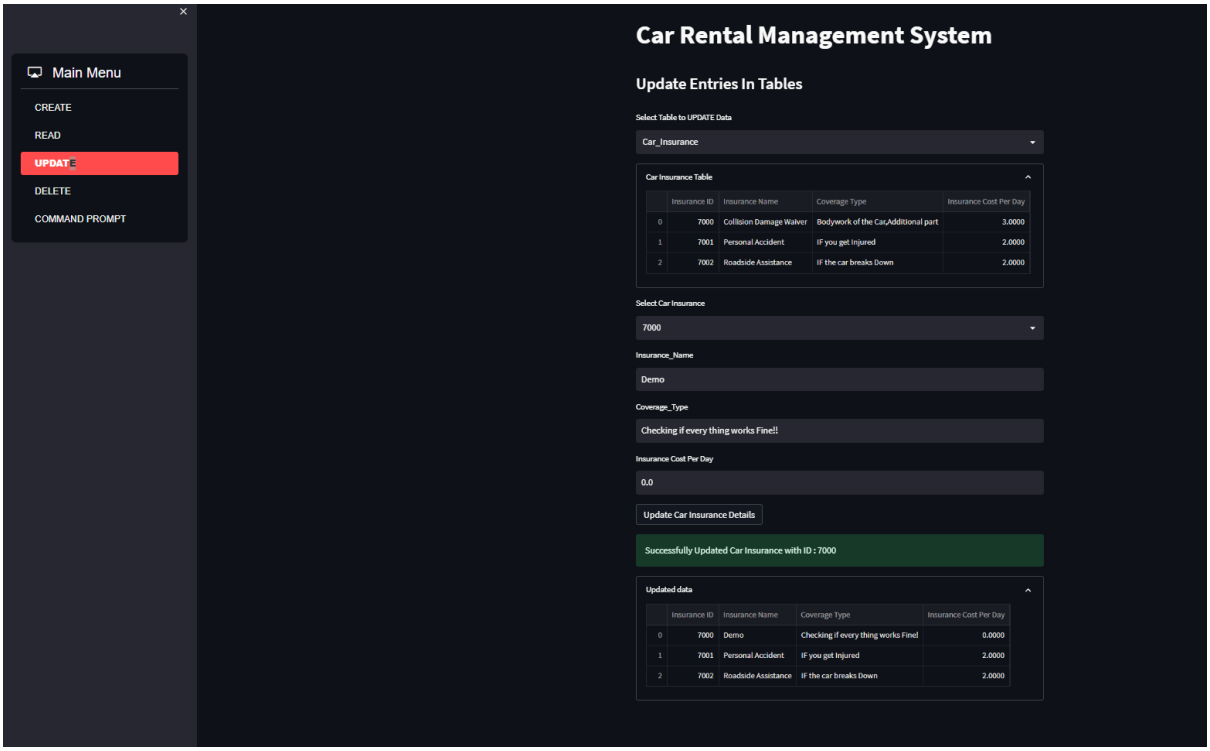


Fig 10.5: Updating the Car Insurance Table

Delete:

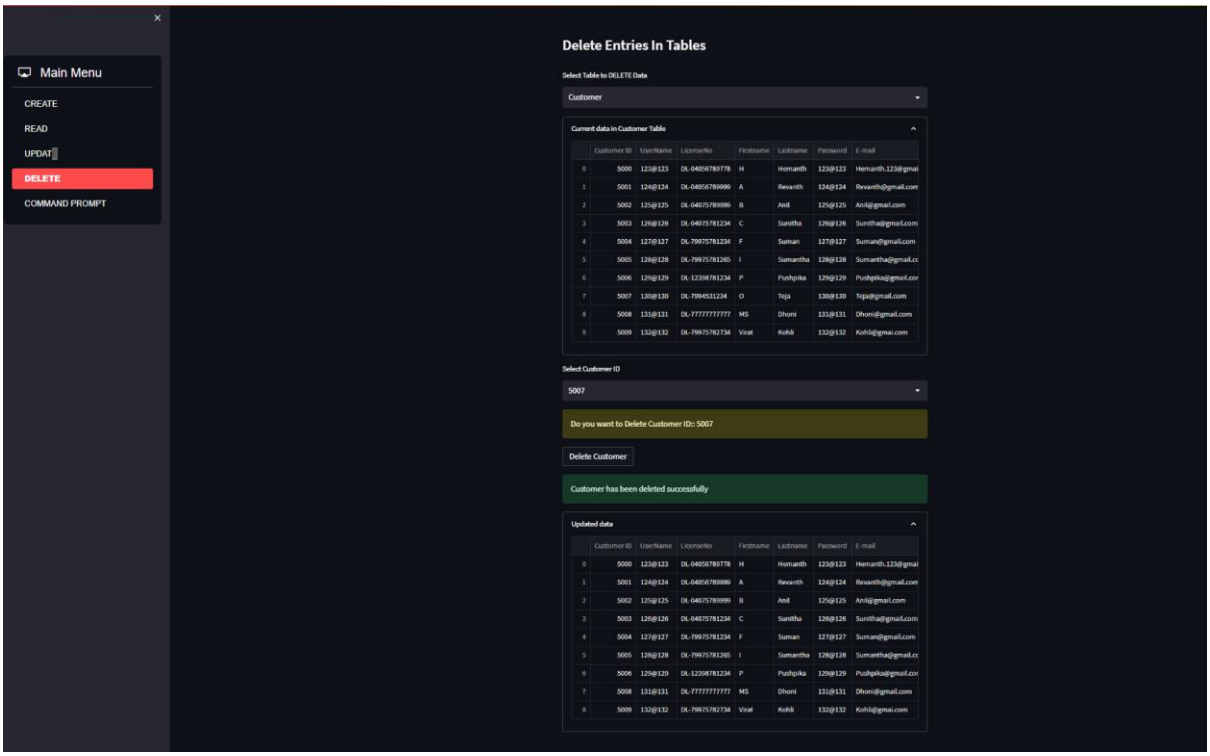


Fig 10.6: Deleting Customer with ID 5007 from Customer Table.

Command Prompt:

Select Operation From Command Prompt

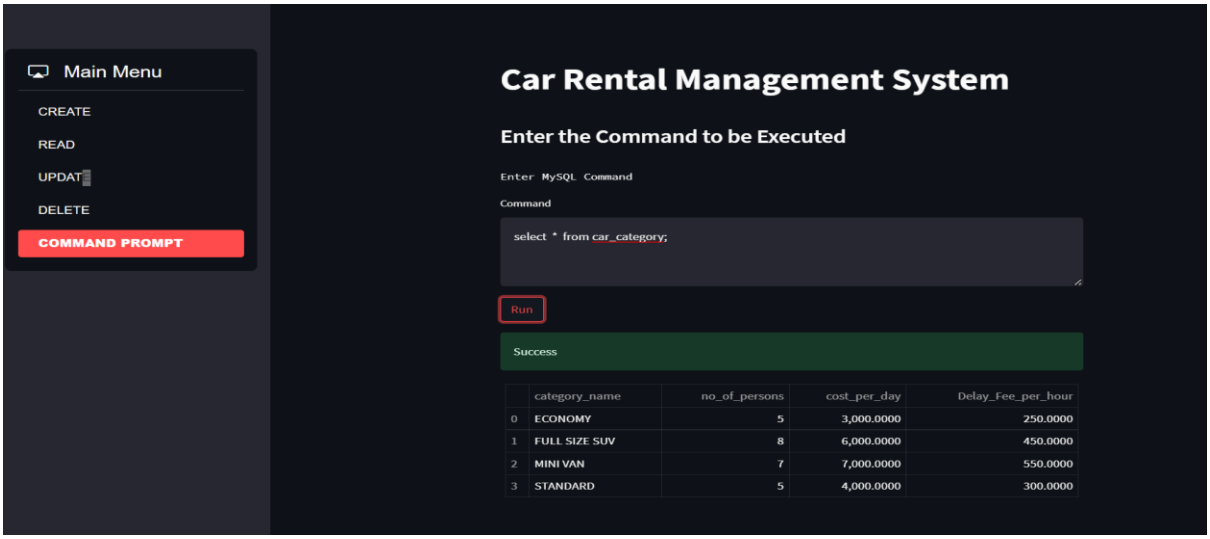


Fig 10.7: Performing Select Operation from Front-End Window

Update operation using Command Prompt:

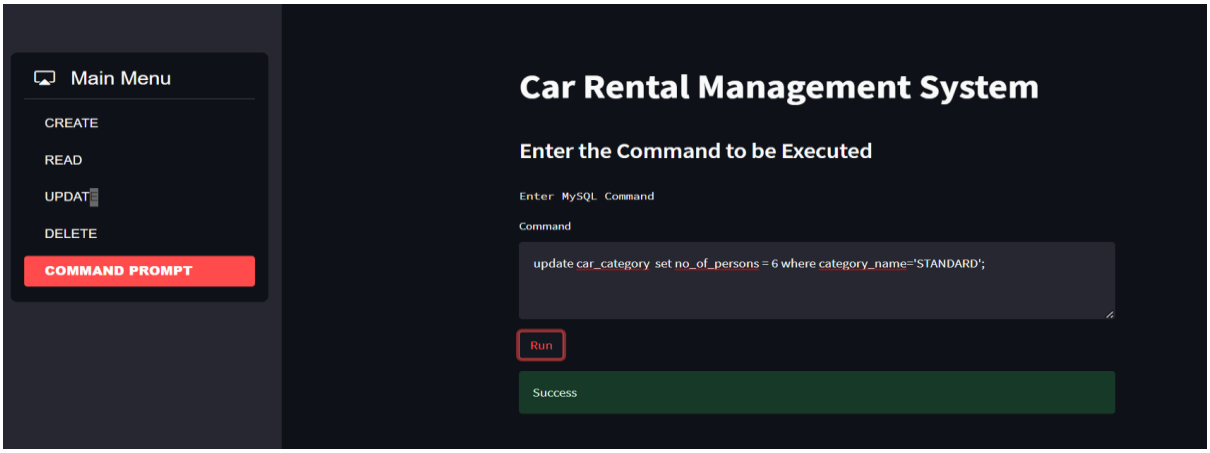


Fig 10.8: Update Operation Using Front-End Window

Displaying the car_Category table to check if the result has been updated.

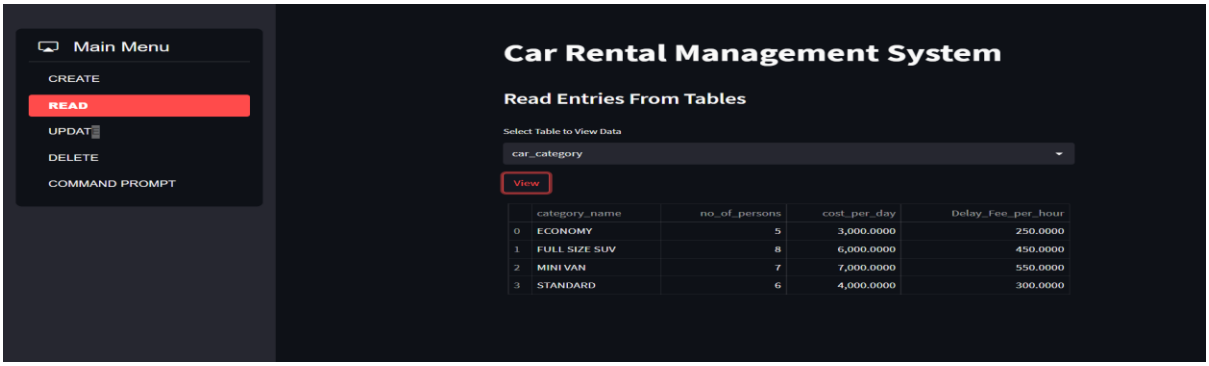


Fig 10.9: Viewing the Car Category table to check if the value has been updated Successfully