PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering

Jan – May 2020

UE18CS252

Database Management Systems

Project Report

<Car Sales and Service Dtabase >

PES1201802001 Kavishankar K S

4th Sem Sec. A Roll No.58

PROJECT SUMMARY

The project contains the mini world of Car Sales and Service

When we come across the showroom which sold large number of cars has to maintain the large amount of data with respect to every car has sold and payment and service for the car

The details of the customer who had purchased the car. To make it simpler to store the data of every associated system in the database which has ease of inserting, updating and deleting.

The database mainly contains with the showroom which is associated with all its employee details, customer, car models, service point and payment details.

In the database we store the details of the customer who brought the car at the particular showroom car details identified by the car number, payment made for the car which is identified by the payment Id, service point for the car.

The design of the database was done using an ER model. A schema and an ER diagram have been depicted for a better understanding. Various relations and tables have been shown and how each of them are related. The cardinality of these relationships is also given. The tables were made using the DDL commands.

The every entity directly connected to primary key and there is no week entity in the relation.

Here we used triggers to update the salary scale of the employee who joined to the department recently since as every year the living scale increase. The trigger will make it automatically without any additional requirement of any manipulation.

Introduction

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. ... The **database** management system (DBMS) is the software that interacts with end users, applications, and the **database** itself to capture and analyse the data.

Here we create the database of the car sales and service system which helps in maintaining the database of the every aspect of showroom which contains the details of every customer, Employee's working in the showroom, mechanics working in the service point, car models and the payment details of the every car sold in the showroom.
The normalisation is done to some extent by framing the correct ER model, the functional dependencies of every entity's are defined.

The important entities are:

1) Showroom:
The entity is identified by the unique attribute showroom Id which is the primary key for the entity, where the non key attributes are showroom name, showroom location, number of employees working in the showroom, pin code of the showroom location.

2) Customer:
The primary of the entity is customer id others non key attributes are customer Name, phone, email, pin code, foreign key attributes are showroom id and car number.

3) Car Models:
We identify the car model using primary key attribute car number, other attributes are
Car version, car model, model year, car seats count and showroom ID as the foreign key attribute.

4) Employee's
The employee's working in the showroom are identified using the primary key of the employee table Employee Id and associated non key attributes are EmpName, EmpPhone, EmpAddress,Empsalary and foreign key attribute is showroom ID.

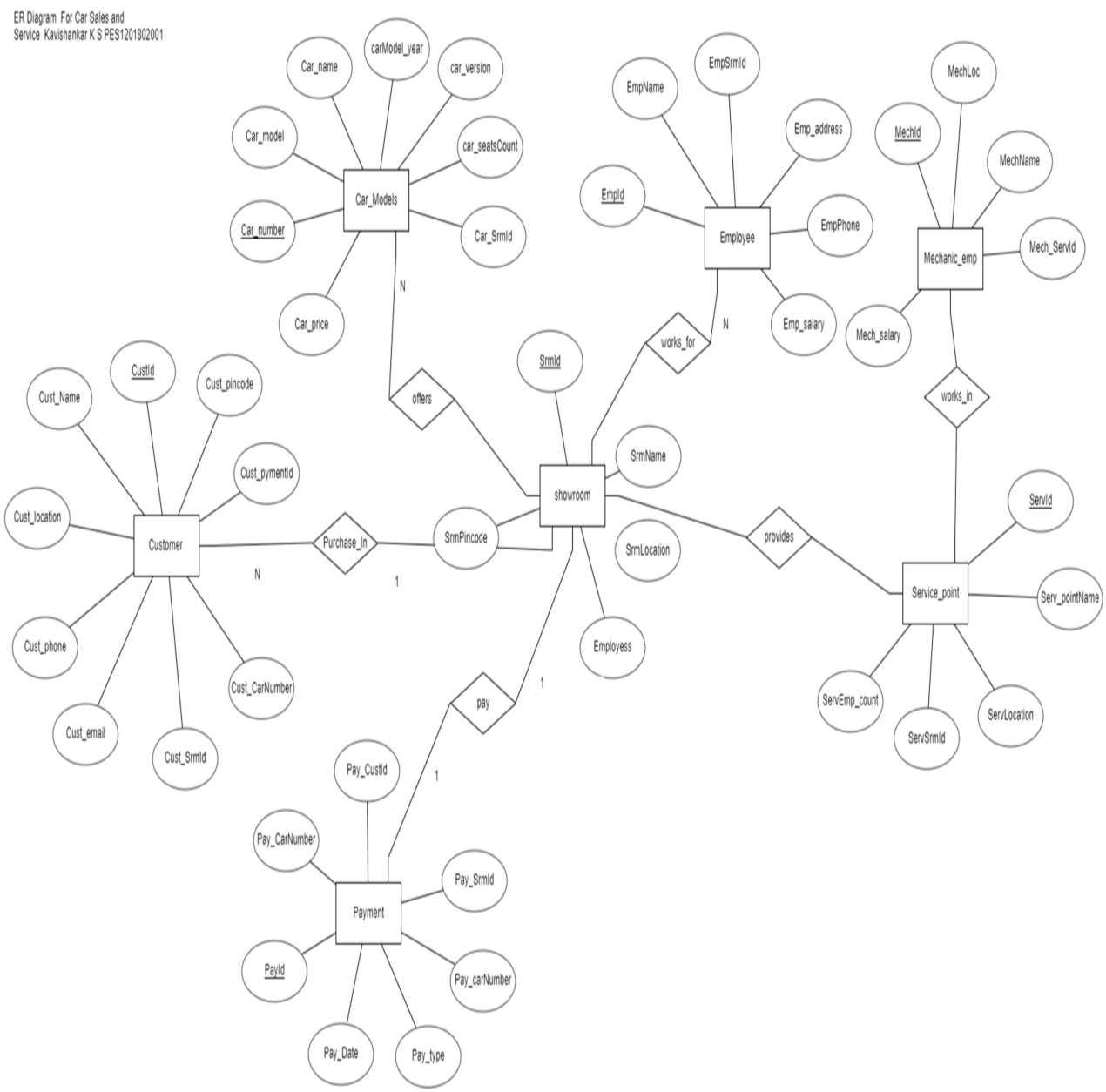
5) Payment:
Payment has the primary key which is payment Id and other attributes are date of payment, type of payment and car number as the froeign key

6)Service Point
We have the associated service point for the every showroom and it contains the sevice Id as the primary key attribute and showroom Id as the foreign key.

7) Mechanic Employee's
The employee's in the service point who are identified by Mechanic Id as the primary key in the database and we store the information about them such as employee name, phone, salary and location and we have servId as the foreign key.

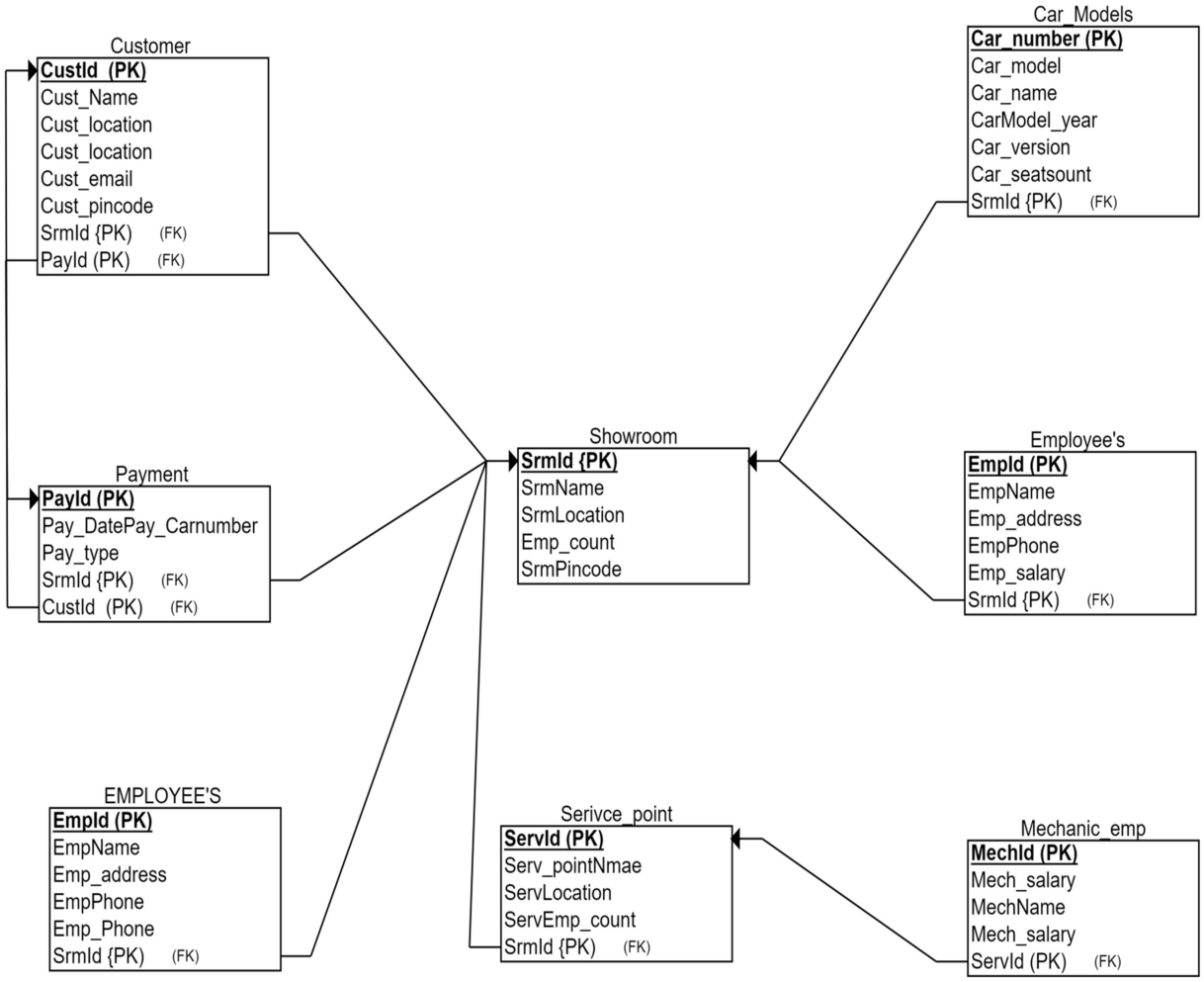
Data Model

ER Diagram



Relational Schema Diagram

Relational Schema Diagram Kavishankar K S PES1201802001



Type of relations:

One-one binary relation(1:1)

- ➔ Customer and car
- ➔ Payment and car number

One-many binary relation (1:N)

- ➔ Showroom and car models
- ➔ Showroom and customer
- ➔ Showroom and employee's
- ➔ Service point and mechanics

Many-many binary relation(N:N)

- ➔ Showroom and Service point.

FD and Normalization

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

A functional dependency, denoted by X → Y, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], they must also have t1[Y] = t2[Y].

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

$$X \rightarrow Y$$

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

In this library we can fetch information about all customer like customer location, contact details by using the customer Id attribute.

Here the customer Id is the determinant and customer details are dependent.

Using customer ID determining the name of the customer.

```
mysql> select custName
    -> from customer
    -> where custId="CUS20201";
+----------+
| custName |
+----------+
| RAJESH   |
+----------+
```

Showroom

SrmId ->{SrmName,SrmLocation,Emp_count,SrmPincode}

In the showroom relation the SrmId is the primary key

Customer

CustId ->{CustName,Custlocation,CustEmail,CustPhone,CustPincode,SrmId,PayId}

Where the CustId is the primary key and SrmId, PayId are foreign key.

Car Model

Car_number-> {Car_Model,Car_name,CarModelYear,CarVersion,CarSeatsCount,SrmId}

Car_number is the primary key.

Employee's

EmpId->{EmpName,EmpAdress,EmpPhone,EmpSalary,SrmId}

Where EmpId is the primary key.

Payment

PayId->{Pay_date,PayCarNumber,PayType,PaySrmId}

Service Point

ServId ->{ServPointName,ServLocation,ServEmpCount,SrmId}

Primary key is ServId.

Mechanic Employees

MechId ->{MechName,MechSalary,MechServId}

Primary key is MechId.

Violations of Normalization:

1NF -> The normalization violation occurs when a tuple holds more than one value for an attribute and therefore the atomicity is lost. Atomicity here refers to values in the table not being further divided

We know that the address is the composite attribute which violates the 1NF in the relation because it is associated with the location, zip code, street but to make it simpler we have created the separate column to store the zip code of the entity.

2NF

➔ For 2NF to not be violated, it must satisfy 1NF. And all non-key attributes must be fully functionally dependent on the Primary Key.

Here, 2NF is not violated as all the non-key attributes are fully dependent on the primary keys of their relation.

If we introduce the Employee Count to the customer table it violates the 2NF because employee table has nothing to do with the employee count and no primary key of the customer entity will give the information about the employee count.

3NF

➔ For 3NF to not be violated, it must satisfy 2NF. The other condition is there should be no transitive dependency for non-prime attributes. That means non-prime attributes should not be dependent on other non-prime attributes in a given table. So, a transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and

Y → Z (where it is not the case that Y → X).

No prime attribute dependence on the other non-prime attributes in the relation transitively.

Ex->In the relation customer location depends on the pin code which not an primary attribute and pincode depends on the Customer Id which is primary attribute so it makes the violation of 3NF in the relation to avoid this create table for the location attribute containing the pincode as the primary key to avoid the 3NF violation.

DDL

```
1)create table showroom(
    SrmId char(8) not null,
    SrmName  varchar(15) not null,
    SrmLocation varchar(15) not null,
    Emp_count int not null,
    Srmpincode int not null,
    primary key(SrmId)
   );

2)create table Employee (
   EmpId char(8) not null,
```

```
   EmpName varchar(15) not null,
 3)  Emp_address varchar(20),
   EmpPhone int,
   EmpSrmId char(8) not null,
   primary key(EmpId)
   foreign key(EmpSrmId) references Showroom(SrmId)
   );

4)create table CarModels(
   Car_number char(10) not null,
   Car_model varchar(15) not null,
   CarModelYear int not null,
   CarVersion varchar(10) not null,
    CarSeatsCount int not null,
    CarSrmId char(8) not null,
    Primary key(Car_number),
    foreign key(CarSrmId) references Showroom(SrmId)
    ):

 5) Create Table service_point (
    ServId char(8) not null,
    ServpointName varchar(15) not null,
    ServLocation varchar(25) not null,
    ServEmpCount int not null,
    ServSrmId char(8) not null,
    Primary key(ServId),
   Foreign key(ServSrmId) references Showroom(SrmId)
 );

6) Create Table MechanicEmp(
   MechId char(8) not null,
   MechName varchar(15) not null,
   MechSalary int not null,
   MechPhone int,
   MechAddress varchar(25),
   MechServId char(8) not null,
   Primary key(MechId),
   Foreign key(MechSevId) references Showroom(SrmId)
    );
7) Create Table customer (
    CustId char(8) not null,
```

```
    CustName varchar(15) not null,
    CustLocation varchar(25) not null,
    CustEmail varchar(15),
    CustPhone int not null,
    CustPincode int not null,
    CustCarNumber char(10) not null,
    CustSrmId char(8) not null,
    Primary key(CustId),
    Foreign key(CustCarNumber) references CarModels(Car_Number),
    Foreign key(CustSrmId) references Showroom(SrmId)
    );
8) Create Table payment (
    PayId char(8) not null,
    PayDate Date,
    PayType varchar(15) not null,
    PayCarNumber char(10) not null,
    Primary key(PayID),
   Foreign key(Paycarnumber) references CarModels(Car_Number)
  );
```

Triggers

Triggers are the SQL codes that are automatically executed in response to certain events on a particular table. These are used to maintain the integrity of the data. A trigger in SQL works similar to a real-world trigger.

Syntax:

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

Explanation of syntax:

1. create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.

2. [before | after]: This specifies when the trigger will be executed.

3. {insert | update | delete}: This specifies the DML operation.

4. on [table_name]: This specifies the name of the table associated with the trigger.

5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.

6. [trigger_body]: This provides the operation to be performed as trigger is fire

BEFORE and AFTER of Trigger:

BEFORE triggers run the trigger action before the triggering statement is run.
AFTER triggers run the trigger action after the triggering statement is run.

Here we have created trigger event named as trig which on inserting the data into mechanic employee salary table will update the salary to divide by 0.6 times.

```
mysql> create trigger trig
    -> before insert
    -> on mechanicemp
    -> for each row
    -> set new.mechsalary=new.mechsalary/0.6;
Query OK, 0 rows affected (3.71 sec)
```

After inserting the salary the event triggered salary

```
mysql> insert into mechanicemp values("MEC20186","AKASH","23000","9932432365","GANDHI BAZAR SHIVAMOGGA","Mahi1234");
Query OK, 1 row affected (0.21 sec)

mysql> select * from mechanicemp;
+----------+----------+-----------+-----------+--------------------------+-----------+
| MechId   | MechName | MechSalary | MECHPHONE  | MECHADDRESS              | MechServId |
+----------+----------+-----------+-----------+--------------------------+-----------+
| MEC20181 | RAMKUMAR |      20000 | 9932124323 | SHARAVATHINAGAR SHIVAMOGGA | Mahi1234  |
| MEC20182 | Rakesh   |      24000 | 9900124543 | puradal SHIVAMOGGA         | Mahi1234  |
| MEC20183 | Kiran    |      22000 | 8900175443 | govindapura SHIVAMOGGA     | Mahi1234  |
| MEC20184 | Bhuvan   |      23200 | 7801232445 | Navule SHIVAMOGGA          | Mahi1234  |
| MEC20185 | MANOJ    |      25000 | 9923345312 | DURGIGUDI SHIVAMOGGA       | MAHI1234  |
| MEC20186 | AKASH    |      38333 | 9932432365 | GANDHI BAZAR SHIVAMOGGA    | Mahi1234  |
+----------+----------+-----------+-----------+--------------------------+-----------+
```

SQL Queries

1)Advanced Query

a) Car Models sold between the year 2014 to 2015

```
mysql> select *
    -> from carmodels
    -> where carmodelyear BETWEEN "2014" and "2017";
+------------+-----------+--------------+------------+---------------+----------+
| Car_number | car_model | CarModelYear | carVersion | CarSeatsCount | CarSrmId |
+------------+-----------+--------------+------------+---------------+----------+
| KA14FE4352 | ALTURAS   |         2014 | TOP        |             5 | MAHI1234 |
| KA14SD3291 | THAR      |         2017 | TOP        |             7 | MAHI1234 |
| KA14SH3231 | TUV300    |         2015 | MIDDLE     |             7 | MAHI1234 |
| KA14VF2352 | ALTURAS   |         2014 | TOP        |             5 | MAHI1234 |
+------------+-----------+--------------+------------+---------------+----------+
```

Order By

Ordering customer in ascending order according to their names.

```
mysql> select *
    -> from customer
    -> order by custname asc;
+----------+-----------+----------------------+--------------------+-----------+------------+--------------+----------+
| CustId   | CustName  | CustLocation         | CUSTEMAIL          | CUSTPHONE | CustPincode | CustCarNumber | CustSrmId |
+----------+-----------+----------------------+--------------------+-----------+------------+--------------+----------+
| CUS20202 | GURUPRASAD | RR NAGAR SHIMOGA     | GURUP@GMAIL.COM    | 945358732 | 577243     | KA14ER5423   | MAHI1234 |
| CUS20203 | PRAKASH   | NAGARAHALLI SHIMOGA  | PRAKASH@GMAIL.COM  | 8753582334 | 577208     | KA14BR3291   | MAHI1234 |
| CUS20204 | PRASHANTH | GURUPURA SHIMOGA     | PRASHANTH@GMAIL.COM | 9002582334 | 577207     | KA14FE4352   | MAHI1234 |
| CUS20201 | RAJESH    | GOPI CIRCLE SHIMOGA  | RAJESH12@GMAIL.COM | 9902358732 | 577203     | KA14BG5421   | MAHI1234 |
| CUS20205 | SRENIVAS  | HANUMATHAPURA SHIMOGA | SRENIVAS@GMAIL.COM | 9121382334 | 577347     | KA14GH3421   | MAHI1234 |
+----------+-----------+----------------------+--------------------+-----------+------------+--------------+----------+
5 rows in set (0.00 sec)
```

Correlated nested Queries

Selecting mechanic employee having salary greater than 22000.

```
mysql> select mechname,mechsalary
    ->  from mechanicemp
    ->  where mechId in ( select mechId
    ->  from mechanicemp
    ->  where mechsalary>22000);
+----------+------------+
| mechname | mechsalary |
+----------+------------+
| Rakesh   |      24000 |
| Bhuvan   |      23200 |
| MANOJ    |      25000 |
+----------+------------+
```

Selecting customer name, phone and customer car number where customer car number is KA14BR3291

```
mysql> SELECT CUSTNAME
    -> SELECT CUSTNAM^C
mysql> SELECT CUSTNAME,CUSTCARNUMBER,CUSTPHONE
    -> FROM CUSTOMER
    -> WHERE CUSTCARNUMBER IN (SELECT CUSTCARNUMBER
    -> FROM CUSTOMER
    -> WHERE CUSTCARNUMBER="KA14BR3291");
+----------+--------------+------------+
| CUSTNAME | CUSTCARNUMBER | CUSTPHONE  |
+----------+--------------+------------+
| PRAKASH  | KA14BR3291    | 8753582334 |
+----------+--------------+------------+
```

Aggregate Queries

Selecting the Mechanic Employee name and phone number in Service department who has maximum pay scale.

```
mysql> SELECT MECHNAME,MECHPHONE
    -> FROM MECHANICEMP
    -> WHERE MECHSALARY=ANY(SELECT MAX(MECHSALARY)
    -> FROM MECHANICEMP);
+----------+------------+
| MECHNAME | MECHPHONE  |
+----------+------------+
| MANOJ    | 9923345312 |
+----------+------------+
```

Selecting the selling rate of the car based on the version

```
mysql> SELECT CARVERSION,COUNT(*) AS TOTAL_VERSION
    -> FROM CARMODELS
    -> GROUP BY CARVERSION
    -> ORDER BY TOTAL_VERSION ASC;
+------------+---------------+
| CARVERSION | TOTAL_VERSION |
+------------+---------------+
| LOW        |             1 |
| MIDDLE     |             4 |
| TOP        |            10 |
+------------+---------------+
```

Sold car based on the model

```
mysql> SELECT  CAR_MODEL,COUNT(*) AS TOTAL
    -> FROM CARMODELS
    -> GROUP BY CAR_MODEL
    -> ORDER BY TOTAL ASC;
+-----------+-------+
| CAR_MODEL | TOTAL |
+-----------+-------+
| XUV100    |     1 |
| BOLERO    |     1 |
| SCORPIO   |     1 |
| KUV100    |     2 |
| ALTURAS   |     2 |
| TUV300    |     3 |
| THAR      |     5 |
```

Conclusion

In our daily life we may come across many type of the databases. In this some has to be maintained and some are not. We use the database in many ways like track monthly expenditure, yearly current bill, transportation charges. In many mobile application they track the user to customise the client requirements, personalization of web pages and more for things.

In this we store the database of the showroom which sells the cars .The project is very helpful in maintaining database of car selling showroom where we connect every showroom data with the customer and sold car details which is effective in bringing every data with each other. Thus the showroom easily can maintain the data of the evry car has been sold and service for the car has been offered by the showroom.

Future works:

- ➔ Framing the front end
- ➔ Collecting the more data
- ➔ Automating the more data like location based on gps.
- ➔ Sending the updates to customer about the service and loan payment.
- ➔ Developing the chat bot for the customer basic queries.