DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By:

PES2201800368 NITISH S

The CAR RENTAL DATABASE SYSTEM is a system for managing a RENTAL COMAPNY where CUSTOMERS reserve CARS which are available for a particular duration they wish to reserve.

All the assumptions made with regards to the Database are kept in mind and proceeded with the DATABASE design.

The **ER diagram** contains the information about the Entity Types, Relationships and the Structural Constraints . The **Logical Design** where the Relational Schema is built along with the appropriate choice of Primary Keys and Foreign Keys for each Relation. The **Physical Design** which contains the DDL statements to create the database and its tables applying all the required constraints.

The **Triggers** which are implemented help in taking care of violations if any exist while entering the values and also help in storing details about when updation/insertion/deletion is performed on the Database for future references. The **Retrieval Queries** written in SQL help in obtaining the information from the Database whenever required so that exhaustive search can be avoided.

This CAR RENTAL DATABASE SYSTEM when clubbed along with a proper website or application can behave as an efficient RENTAL COMPANY / WEBSITE for renting cars by the customers for the required duration at reasonable rates.

The Backend i.e. the Database is also developed in a very organized way which takes care of all the requirements to be fulfilled and also the constraints which have to be taken care of and ensures that there are no clashes of any sort among the values in the Database.

INTRODUCTION

The CAR RENTAL DATABASE SYSTEM is a system for managing a RENTAL COMAPNY where CUSTOMERS reserve CARS which are available for a particular duration they wish to reserve.

The ENTITIES TYPES in this DATABASE SYSTEM are:

1) CARS --- Contains details of the different Cars available for renting along with the unique Car NO i.e VIN, Brand, Model and Color of the Car.

2) CATEGORIES ---- Table which contains the details of the Type or Segment , a particular Car Belongs To along with a unique Category ID allotted to each Category.

E.g Hatchback, Sedan , SUV.

3) CUSTOMERS ---- Table which contains information of the Customers who have registered themselves with the Rental Company along with their Address, Phone No, Name and a unique ID which is provided to each Customer by the Rental Company

4) LOCATIONS ---- Table which contains the location at which a particular car is present and is reserved at. Each location is given a distinct Location ID and is stored along with the Address i.e Street NO, Street, City, State, Country and also Phone NO of the Location which can have multiple values.

5) RESERVATIONS --- It contains the details of the Reservations which are made along with a unique Reservation ID, ID of the Customer who has made the Reservation, VIN of the car which is Reserved, Location ID at which the Car is reserved, duration of the reservation i.e the Pickup Date and the Return Date and the Amount charged for the Reservation.

The following Assumptions are made during the Design of the DATABASE and also while designing the Relationships among the various Entity types: (REQUIREMENT ANALYSIS):

1) A Car can be rented multiple times or never be rented at all, but a Reservation ID includes only one Car at a particular time {Pickup Date, Return Date }.

2) The same Customer can have more than one Reservations on different dates and with different Cars.

3) A Car is rented by only one Customer at a particular time. If the same Customer wants to make another Reservation , he/she can do so and a new Registration ID will be provided.

4) A rental belongs to no more than one Location. Rental's Location is identified as the Pickup and Return Location of the Car allotted when the Reservation ID was issued.

The following assumptions are kept in mind and proceeded with the DATABASE design in the following Pages.

The ER diagram contains the information about the Entity Types, Relationships and their Structural Constraints .

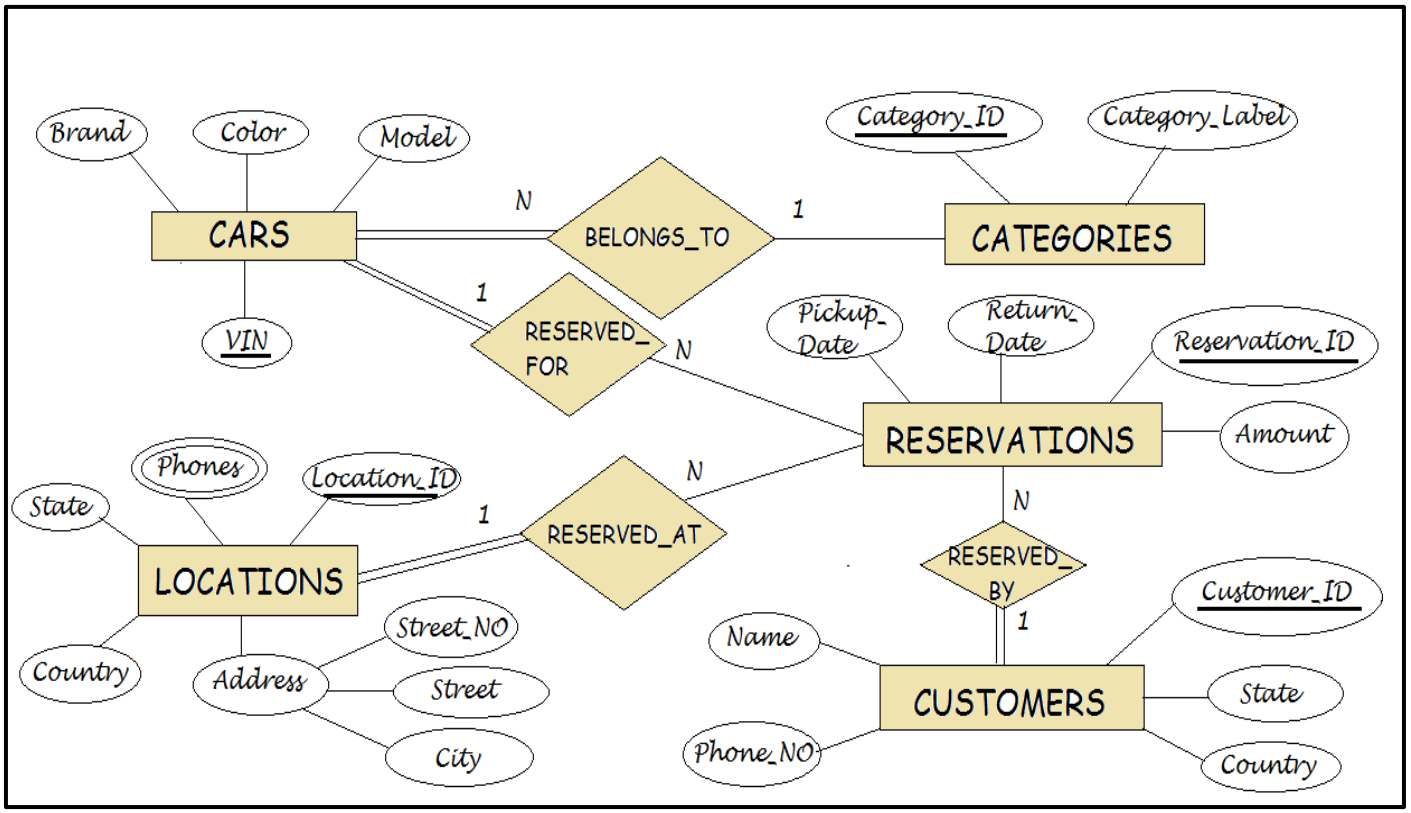
It is followed by the Logical Design where the Relational Schema is built along with the appropriate Primary Keys and Foreign Keys chosen for each Relation.

This Is followed by the Physical Design which contains the DDL statements to create the database and its tables applying all the required constraints.

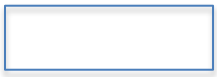
A few Triggers are implemented and some useful Retrieval Queries in SQL are also written to the Car Rental Database System.

DATA MODEL

ER DIAGRAM FOR THE CarRental Database:-



INDEX:-



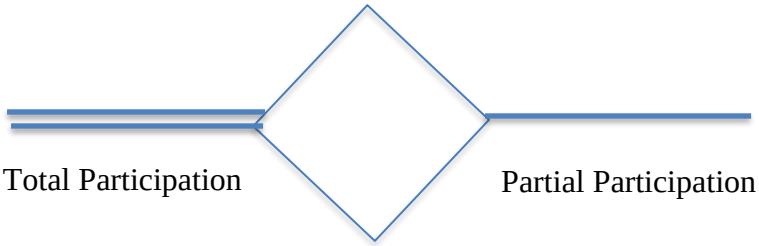
ENTITY TYPE

ATTRIBUTE OF AN ENTITY TYPE

PRIMARY KEY ATTRIBUTE OF AN ENTITY

MULTIVALUED ATTRIBUTE

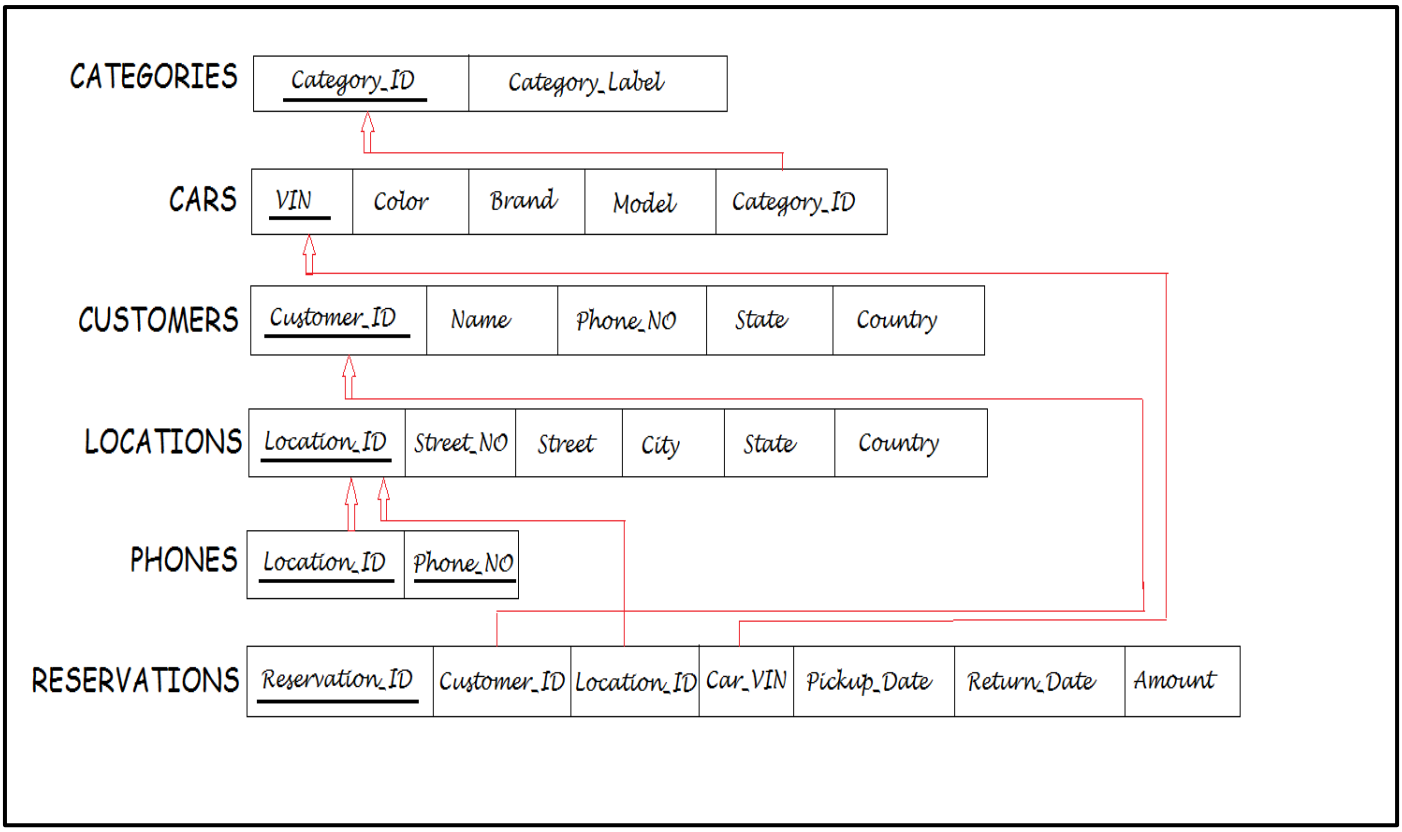
RELATIONSHIP



Total Participation

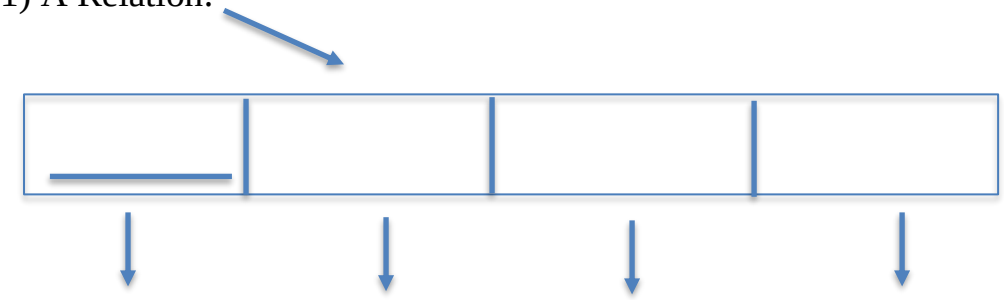
Partial Participation

The ***RELATIONAL SCHEMA*** created following all the rules of the ER Diagram to Relational Schema Mapping.



INDEX:-

1) A Relation:



PRIMARY KEY ATTRIBUTE OF THE REALTION

ATTRIBUTES OF THE RELATION

2) The ARROWS represent the ***Referential Integrity Constraints.*** (i.e Foreign Key mapping to the Primary Key).

FD AND NORMALIZATION

FUNCTIONAL DEPENDENCIES:-

- **VIN** -----> { Model, Color, Brand, Category_ID }
- **Category_ID** -----> { Category_Label }
- **Location_ID** ----> { Street_NO, Street, City, State, Country}
- **Customer_ID** -----> { Name, Mobile_NO, State, Country}
- **Registration_ID** ----> { Pickup_Date, Return_Date, Amount, Customer_ID, Location_ID, Car_VIN }

NORMALIZATION:-

On converting the ER Model to the Relational Schema following all the rules of ER Model to Relational Schema mapping, the relations obtained are already in the normal form.

However, under the following circumstances the relations may violate the Normal Forms:

VIOLATION OF 2nd NORMAL FORM:-

If Relations were to be in the following way:

CARS { VIN, Color, Brand, Model, Category_ID } with VIN as the PK,

RESERVATIONS { Reservation_ID, Car_VIN, Customer_ID, Location_ID, Amount} with Reservation_ID as the PK,

And instead of having Pickup_Date and Return_Date as attributes in the RESERVATIONS relation, if there existed a relation as follows:

RESERVATION_DURATION { Reservation_ID, Car_VIN, Duration } with Reservation_ID and Car_VIN together as the COMPOSITE KEY,

THEN

If Model of the Car was also included as an attribute in the RESERVATION_DURATION relation, then

(Car_VIN ----> Model) but the Key of the Relation is { Reservation_ID, Car_VIN }.

There exists a **PARTIAL DEPENDENCY** and hence the Relation would not satisfy the **2nd NORMAL FORM.**

VIOLATION OF 3rd NORMAL FORM:-

If instead of including just the Category_ID as foreign key in the CARS relation, if we include Category_Label also, in other words if CARS and CATEGORIES relation are combined into a single relation,

R: CARS { VIN, Color, Brand, Model, Category_ID, Category_Label } with VIN as the PRIMARY KEY of the relation, then according the above functional dependency,

VIN ---> { Color, Brand, Model, Category_ID, Category_Label }

But also Category_ID ---> { Category_Label } .

We observe that there exists a **TRANSITIVE DEPENDENCY** in this case. Hence, there is a violation of the **3RD NORMAL FORM.**

In order to remove the violation, we decompose the relation into two relations ensuring the Lossless - Join Property and Dependency Preservation Property. The two relations are:

R1: CARS { VIN, Color, Brand, Model, Category_ID } with VIN as the Primary Key and Category_ID as the Foreign Key.

AND

R2: CATEGORIES { Category_ID, Category_Label } with Category_ID as the Primary Key.

TEST FOR LOSSLESS DECOMPOSITION:

The final table applying the Functional Dependencies: i.e
VIN -----> { Model, Color, Brand, Category_ID }
Category_ID -----> { Category_Label }

R	VIN	Color	Brand	Model	Category_ID	Category_Label
R1	~~b11~~ a1	~~b12~~ a2	~~b13~~ a3	~~b14~~ a4	~~b15~~ a5	~~b16~~ a6
R2	b21	b22	b23	b24	~~b25~~ a1	~~b26~~ a2

One row of the table contains all a's. Therefore, such a decomposition is valid and lossless.

DDL

CREATING THE DATABASE:

```
create database CarRental;
```

CREATING THE TABLES IN THE DATABASE NAMED 'CarRental':

```
CREATE TABLE Categories(
   Category_ID int not null,
   Category_Label varchar(10),
   PRIMARY KEY (Category_ID)
);

CREATE TABLE Cars (
   VIN varchar(20) not null,
   Color varchar(10),
   Brand varchar(20),
   Model varchar(20),
   Category_ID int,
   PRIMARY KEY (VIN)
);

CREATE TABLE Locations (
   Location_ID int not null ,
   Country varchar(20),
   State varchar(20),
   Street_NO varchar(10),
   Street varchar(20),
   City varchar(20),
   PRIMARY KEY (Location_ID)
);

CREATE TABLE Phones (
   Phone_NO character(13) not null,
   Location_ID int not null,
   PRIMARY KEY (Location_ID , Phone_NO)
);

CREATE TABLE Customers (
   Customer_ID int not null,
   Customer_Name varchar(20) not null,
   Phone_NO character(13) not null,
   State varchar(20),
   Country varchar(20),
   PRIMARY KEY (Customer_ID)
);

CREATE TABLE Reservations (
   Reservation_ID int not null ,
   Car_VIN varchar(10) not null,
   Cust_ID int not null,
   Location_ID int not null,
   Pickup_date DATE not null,
   Return_date DATE not null,
   Amount float not null,
   PRIMARY KEY (Reservation_ID)
);

CREATE TABLE Customers_Audit (
    Customer_ID int not null,
   Customer_Name varchar(20) not null,
   Action VARCHAR(50) ,
   UpdateDate DATETIME,
   PRIMARY KEY (Customer_ID)
);
```

Adding Check Constraints and Referential Integrity Constraints to the Created Tables:

1) ALTER TABLE Cars ADD CONSTRAINT cat_id
FOREIGN KEY (Category_ID) references Categories (Category_ID)
on delete cascade on update cascade;

2) ALTER TABLE Phones ADD CONSTRAINT loc_id
FOREIGN KEY (Location_ID) references Locations (Location_ID)
on delete cascade on update cascade;

3) ALTER TABLE Reservations ADD CONSTRAINT cust_id
FOREIGN KEY (Cust_ID) references Customers (Customer_ID)
on delete cascade on update cascade;

4) ALTER TABLE Reservations ADD CONSTRAINT car_vin
FOREIGN KEY (Car_VIN) references Cars (VIN)
on delete cascade on update cascade;

5) ALTER TABLE Reservations ADD CONSTRAINT location_id
FOREIGN KEY (Location_ID) references Locations (Location_ID)
on delete cascade on update cascade;

6) ALTER TABLE Categories
ADD CONSTRAINT CHECK (Category_ID > 0 AND Category_ID < 4);

7) ALTER TABLE Reservations
ADD CONSTRAINT CHECK (Return_date >= Pickup_date);

8) ALTER TABLE Locations
ADD CONSTRAINT CHECK (Location_ID > 0 AND Location_ID < 11);

Examples of Statements to enter values into the Created Tables:

INSERT INTO Categories VALUES (1,'Hatchback');

INSERT INTO Categories VALUES (2,'PrimeSedan');

INSERT INTO Categories VALUES (3,'SUV');

INSERT INTO Cars VALUES 'KA04P3633','Golden','Suzuki','Maruti800',1);

INSERT INTO Cars VALUES ('KA02MH4543','Mustard','Suzuki','WagonR',1);

INSERT INTO Customers VALUES (23,'Satish','080984543029','Karnataka','India');

INSERT INTO Customers VALUES (123,'Suresh','9197657320198','TamilNadu','India');

INSERT INTO Locations VALUES (1,'India','Karnataka','21A','Vijaynagar','Bangalore');

INSERT INTO Locations VALUES (2,'India','Karnataka','9B','MG Road','Mysuru');

INSERT INTO Phones VALUES ('0802435261780',1);

INSERT INTO Phones VALUES ('0802435261781',1);

INSERT INTO Phones VALUES ('0794516273901',2);

INSERT INTO Reservations VALUES (1265,'KA04P3633',23,1,'2020-08-02','2020-08-02',450);

INSERT INTO Reservations VALUES (1288,'KA02MH4543',706,3,'2020-07-23','2020-07-25',3200);

INSERT INTO Reservations VALUES (243,'DL88RF2167',980,4,'2020-07-12','2020-07-18',7820);

DATABASE STATE

VIN	Color	Brand	Model	Category_ID
DL88RF2167	White	Volkswagen	Ameo	2
GJ69NB4765	Black	Tata	Indigo	2
KA02MH4543	Mustard	Suzuki	WagonR	1
KA03X127	Black	Honda	Civic	2
KA04P3633	Golden	Suzuki	Maruti800	1
KA07NH1232	Blue	Maruti	NULL	1
MH01Y4325	Red	Audi	Q5	3
MH02R5617	Blue	Maruti	Brezza	3
PY09HU1567	Blue	Hyundai	Verna	2
PY09HU657	Brown	Mahindra	Scorpio	3
TN64GF8901	White	Tata	Hexa	3
TN77A129	Blue	Hyundai	I20	1
TS45MN32	White	BMW	3Series	2
NULL	NULL	NULL	NULL	NULL

CARS Table

Category_ID	Category_Label
1	Hatchback
2	PrimeSedan
3	SUV
NULL	NULL

CATEGORIES Table

Customer_ID	Customer_Name	Phone_NO	State	Country
23	Satish	9107654302918	Karnataka	India
65	Anupama	9109903316316	Maharashtra	India
123	Suresh	9197657320198	TamilNadu	India
435	Nitish	9809034598042	Karnataka	India
632	Sandeep Bhat	9107654302918	TamilNadu	India
684	Sathvik Saya	0988769543109	Pondicherry	India
705	Supreet Ronad	9809789096514	Maharashtra	India
706	Shashank GS	9107654302118	Gujarat	India
897	Manjula	9189980913517	Telangana	India
980	Tripti	2198765432190	Delhi	India
NULL	NULL	NULL	NULL	NULL

CUSTOMERS Table

Phone_NO	Location_ID
0802435261780	1
0802435261781	1
0794516273900	2
0794516273901	2
0092837190829	3
8903241561728	4
8903241561729	4
8802134253678	5
6789087654351	6
6789087654352	6
1007865431928	7
1906578421980	8
9702341562710	9
9702341562718	9
9702341562719	9

PHONES Table

Location_ID	Country	State	Street_NO	Street	City
1	India	Karnataka	21A	Vijaynagar	Bangalore
2	India	Karnataka	9B	MG Road	Mysuru
3	India	Maharashtra	77	ALP Colony	Mumbai
4	India	Gujarat	66J	Margosa Road	Gandhinagar
5	India	Maharashtra	89K	ECity	Thane
6	India	TamilNadu	6D	RRNagar	Chennai
7	India	Telangana	88P	Kalasipalya	Hyderabad
8	India	Telangana	8C	Madivala	Warangal
9	India	Pondicherry	98R	HST Layout	Auroville
10	India	Karnataka	34K	Subbaih Road	Chitradurga
NULL	NULL	NULL	NULL	NULL	NULL

LOCATIONS Table

Reservation_ID	Car_VIN	Cust_ID	Location_ID	Pickup_date	Return_date	Amount
90	GJ69NB4765	123	4	2020-08-12	2020-08-13	1200
190	TN64GF8901	897	8	2020-06-12	2020-06-12	1090
243	DL88RF2167	980	4	2020-07-12	2020-07-18	7820
789	KA02MH4543	705	2	2020-05-19	2020-05-20	1380
1265	KA04P3633	23	1	2020-08-02	2020-08-02	450
1288	KA02MH4543	706	3	2020-07-23	2020-07-25	3200
1980	DL88RF2167	23	4	2020-06-18	2020-06-18	420
2309	KA02MH4543	23	1	2020-09-13	2020-09-14	1380
2318	PY09HU657	684	9	2020-06-01	2020-06-01	1200
4536	TN77A129	632	6	2020-05-18	2020-05-18	760
5687	MH01Y4325	435	4	2020-05-27	2020-05-31	10000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

RESERVATIONS Table

TRIGGERS

1) Trigger to store any update made to a column in the Customers Table.

```
CREATE TRIGGER Before_Update
   BEFORE UPDATE ON Customers
   FOR EACH ROW
   INSERT INTO Customers_Audit
   SET
    action = 'UPDATE',
    Customer_ID = OLD.Customer_ID,
    Customer_Name = OLD.Customer_Name,
    UpdateDate = NOW();
```

E.g. for working of the trigger:

```
Use CarRental;
UPDATE Customers
SET Phone_NO='9107654302918'
 WHERE Customer_ID=23;
```

(Note:
On updating the value as done above, the details namely the Customer_ID and Customer_Name for which update is made, the action and the Time of Update is stored into the Customers_Audit table).

```
SELECT * FROM Customers_Audit;
```

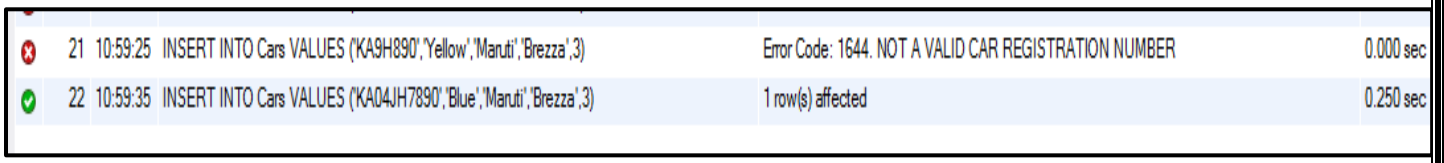
Customer_ID	Customer_Name	Action	UpdateDate
23	Satish	UPDATE	2020-05-19 22:32:15
632	Sandeep Bhat	UPDATE	2020-05-19 20:36:00
NULL	NULL	NULL	NULL

2) Trigger to check if a car VIN is valid before entering into the database.

```
delimiter $$
CREATE TRIGGER Car_Insert_Value
    BEFORE INSERT on Cars
    FOR EACH ROW
   BEGIN
   IF NEW.VIN not rlike '^[A-Z]{2}[0-9]{2}[A-Z]{1,2}[0-9]{1,4}$'
   THEN
   signal sqlstate '45000' set message_text = 'NOT A VALID CAR REGISTRATION
NUMBER' ;

   END IF;
   END;
```

Eg: **KA04JH7890** ----- Valid VIN , so gets entered into the table.
Whereas **K9H897** ------ Displays error as 'NOT A VALID CAR REGISTRATION NUMBER'.



✖	21	10:59:25	INSERT INTO Cars VALUES ('KA9H890','Yellow','Maruti','Brezza',3)	Error Code: 1644. NOT A VALID CAR REGISTRATION NUMBER	0.000 sec
✔	22	10:59:35	INSERT INTO Cars VALUES ('KA04JH7890','Blue','Maruti','Brezza',3)	1 row(s) affected	0.250 sec

SQL QUERIES

--- NESTED QUERIES: ---

1) Retrieve names of customers who have reserved car no KA02MH4543.

```
SELECT C.Customer_Name
FROM Customers as C
WHERE EXISTS ( SELECT *
                FROM Reservations as R
               WHERE R.Car_VIN = 'KA02MH4543' AND R.Cust_ID = C.Customer_ID);
```

Customer_Name
Supreet Ronad
Shashank GS
Satish

2) Retrieve names and phone number of customers who have no reservations.

```
SELECT C.Customer_Name , C.Phone_NO
FROM Customers as C
WHERE NOT EXISTS ( SELECT *
                    FROM Reservations as R
                    WHERE C.Customer_ID = R.Cust_ID);
```

Customer_Name	Phone_NO
Anupama	9109903316316

3) Find all car number and their model which have no reservations.

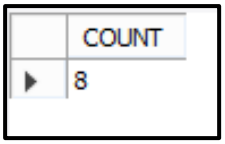
```
SELECT C.VIN , C.Model
FROM Cars as C
WHERE NOT EXISTS ( SELECT *
                    FROM Reservations as R
                    WHERE R.Car_VIN = C.VIN);
```

VIN	Model
KA03X127	Civic
KA04JH7890	Brezza
KA07NH1232	NULL
MH02R5617	Brezza
PY09HU1567	Verna
TS45MN32	3Series
NULL	NULL

--- QUERIES USING AGGREGATE FUNCTIONS: ---

1) Find the number of reservations whose amount is greater than 1000.

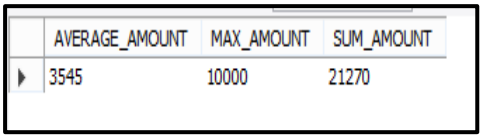
```
SELECT COUNT(*)
FROM Reservations
WHERE Amount>1000;
```



	COUNT
▶	8

2) Find total amount, average amount and highest amount among all the reservations.

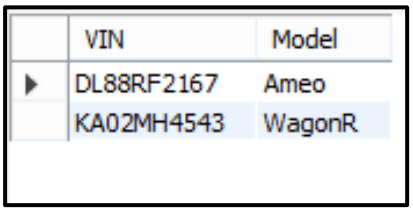
```
SELECT avg(Amount), MAX(Amount), SUM(Amount)
FROM Reservations
WHERE Location_ID = 4 OR Location_ID = 1;
```



	AVERAGE_AMOUNT	MAX_AMOUNT	SUM_AMOUNT
▶	3545	10000	21270

3) For each car which has more than one reservation, retrieve the car number and model.

```
SELECT VIN , Model
FROM Cars, Reservations
WHERE VIN = Car_VIN
GROUP BY VIN, Model
HAVING COUNT(*) > 1;
```



	VIN	Model
▶	DL88RF2167	Ameo
	KA02MH4543	WagonR

--- JOIN QUERIES: ---

1) Return the customer name and their phone number of all customers who have made atleast one reservations.

```
SELECT Distinct Customer_Name, Phone_NO, Reservation_ID, Amount
FROM Customers LEFT OUTER JOIN Reservations ON Customer_ID = Cust_ID
ORDER BY 1;
```

Customer_Name	Phone_NO	Reservation_ID	Amount
Anupama	9109903316316	NULL	NULL
Manjula	9189980913517	190	1090
Nitish	9809034598042	5687	10000
Sandeep Bhat	9107654302918	4536	760
Sathvik Saya	0988769543109	2318	1200
Satish	9107654302918	1265	450
Satish	9107654302918	1980	420
Satish	9107654302918	2309	1380
Shashank GS	9107654302118	1288	3200
Supreet Ronad	9809789096514	789	1380
Suresh	9197657320198	90	1200
Tripti	2198765432190	243	7820

2) Find the count of cars under each category which are reserved.

```
SELECT Categories.Category_ID, Category_Label, Count(1) as
No_of_reservations_of_each_category
FROM Cars LEFT OUTER JOIN Categories ON Cars.Category_ID =
Categories.Category_ID
WHERE EXISTS (SELECT Distinct VIN
                FROM Cars LEFT OUTER JOIN Reservations ON VIN = Car_VIN )
                GROUP BY Category_ID, Category_Label ;
```

Category_ID	Category_Label	No_of_reservations_of_each_category
1	Hatchback	4
2	PrimeSedan	5
3	SUV	5

CONCLUSION

The CAR RENTAL DATABASE SYSTEM is a system for managing a RENTAL COMAPNY where CUSTOMERS reserve CARS which are available for a particular duration they wish to reserve.

This DATABASE SYSTEM when clubbed along with a proper website or application can behave as an efficient RENTAL COMPANY / WEBSITE for renting cars by the customers for the required duration at reasonable rates.

The Backend i.e. the Database is also developed in a very organized way which takes care of all the requirements to be fulfilled and also the constraints which have to be taken care of and ensures that there are no clashes of any sort among the values in the Database. Though some of the clashes which may occur are taken care of, there are some others which may occur and are out of the scope of our course.

Some limitations of this system are:

1) No way to prevent the same Car being reserved during the same Duration more than once.

2) A particular Car may always be present at the same Location and the Customer should choose the Car only if he/she is willing to reserve the car from that Location.

These limitations can be taken care of and can be clubbed with a efficient website/application and also other improvements like automatic rate calculation based on the Car type, duration of booking and other factors, making sure that a already reserved Car during a particular time is not displayed to the Customer while he is booking, listing out Cars only at the Location in which the Customer has registered with etc.

Overall, this Database Development would serve to be very useful when combined along with a RENTAL COMPANY website/application and thereby deploy it for an efficient real world application of the Database.