Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

Near Jnana Bharathi Campus, Bengaluru-560 056.

(An Autonomous Institution, Aided by Government of Karnataka)

Aided By Govt. of Karnataka

MINI PROJECT REPORT

ON

"CAR RENTAL MANAGEMENT SYSTEM"

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED

BY

SHASHANK M KADIWAL   **SANJAY REVANNA**

1DA18CS141   **1DA18CS137**

UNDER THE GUIDANCE OF

Mrs. Asha
Assoc.Prof, Dept of CSE
Dr.AIT

Department of Computer Science & Engineering

2020-21

Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

Near Jnana Bharathi Campus, Bengaluru-560 056.

(An Autonomous Institution, Aided by Government of Karnataka)

Aided By Govt. of Karnataka

CERTIFICATE

This is to certify that the project entitled "**CAR RENTAL MANAGEMENT SYSTEM**" submitted in the partial fulfillment of the requirement of the 5th semester DBMS laboratory curriculum during the year 2020 is a result of bonafide work carried out by-

SHASHANK M KADIWAL	**SANJAY REVANNA**
1DA18CS141	**1DA18CS137**

Signature of the guides:

_____

Mrs. Asha

Assoc. Prof., Dept of CSE

Dr. AIT

1. Internal Examiner _____

2. External Examiner _____

Dr. Siddaraju,

Head of Department

Department of CSE, Dr.AIT

ACKNOWLEDGEMENT

The sense of contention and elation that accompanies the success of this seminar and the report could be incomplete without mentioning the names of people who have helped me in accomplishing them, people whose constant guidance, support and encouragement resulted in the realization.

We consider ourselves privileged to express our gratitude and respect towards all those who guided us through the project, "**CAR RENTAL MANAGEMENT SYSTEM**".

We take this opportunity to thank **Dr. Nanjundaswamy C, PRINCIPAL Dr. Ambedkar Institute of Technology, Bengaluru** for his support and encouragement.

We are grateful to **Dr. Siddaraju, Head of Department, CSE, Dr. Ambedkar Institute of Technology, Bengaluru** for providing encouragement and support.

We consider ourselves privileged to express our gratitude and respect towards our guide **Mrs. Asha, Assoc.Proffessor, Department of CSE, Dr. Ambedkar Institute of Technology** for constant guidance and support for the completion of the project.

Lastly, we thank all the members of the staff both teaching and non-teaching, friends and last but not the least our parents and family, for helping me directly or indirectly in the completion of the project.

SHASHANK M KADIWAL

SANJAY REVANNA

ABSTRACT

The paper developed an automated system that is used to manage car information and its administration. This was with a view to eliminate the problem of inappropriate data keeping, inaccurate reports, time wastage in storing, processing and reserving information encountered by the traditional car rental system in order to improve the overall efficiency of the organization.

The proposed system was tested using the information collected from Avis Car Rental Company, Bengaluru and compared with the existing traditional car rental system. The design provides excellent car rental management service and improved information structure.

CONTENTS

CHAPTER 1

INTRODUCTION

A Car Rental Management System is looking to develop a state-of-the-art car rental portfolio management system which is able to track their car rental booking history and billing and car details. This system facilitates the owner of the Car Rental Company to retrieve, update and track and delete the bookings and cars efficiently. At the same time, can utilize this system to monitor their financial management.

Currently, different locations of the car rental company have their own separated systems leading to lack of communication and inefficient data sharing. For example, the car rental company located in Mumbai uses simple Microsoft Excel to keep the record of their customers, cars and insurance of their cars which is inconvenient to retrieve and update the cars and customer's information. In the car rental company located at Dharwad, maintain a book-based ledger to keep a record of their customers and cars and the insurance for the same. The main branch of the company located in Bengaluru has to keep the customer and car details of all their branch offices on their own computer system. While each statement serves a distinctive process, there is no coordination, assimilation and representation of data. The systems may have duplicate data which leads to waste of space. The different systems also may have different application programs which cause incompatible files.

Due to these disadvantages of the current system, a car rental management system is proposed. Car Rental Management System is a database management system (DBMS) which is based on computer networks, using the advance database technology to construct, maintain and update various kinds of data in data base system. The DBMS can track and update all the information of the cars available and customers during a particular time span. The major advantages of the DBMS are easy to retrieve and update information, efficient data sharing and communication and reliable backup and security.

Information about cars is done by just writing the car name, car model and insurance details. Whenever a new car is added to the company its information is stored freshly. Bills are generated by recording the usage of the car by the customer. Details of the customer are updated on a written sheet and at last, they all summed up. Car usage details are recorded on the document, which contains the customer information. It is destroyed after sometime time period to decrease the paper load in the office. The admin themselves have to track the car details and customer details which is a tedious job.

CHAPTER 2
LITERATURE REVIEW

2.1 DATABASE MANAGEMENT SYSTEM

Data can be defined as a representation of facts, concepts, or instructions in a formalized manner, which should be suitable for communication, interpretation, or processing by human or electronic machine.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system Let us discuss few examples. An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc. Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application. There are 4 major types of DBMS.Lets look into them in detail.

- **HIERARCHICAL** -In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top down or bottom up) format. Data is represented using a parent-child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent.
- **NETWORK DBMS -**The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.
- **RELATIONAL MODEL**-Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.
- **OBJECT-ORIENTED MODEL**-In Object-oriented Model data stored in the form of objects. The structure which is called classes which display data within it. It defines a database as a collection of objects which stores both data members values and operations.

2.2 SQL – STRUCTURED QUERY LANGUAGE

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system.

2.3 MySQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by **Oracle Company**.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by **MySQL AB, a Swedish company**, and written in C programming language and C++ programming language. Many small and big companies use MySQL.   MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++,and Java languages.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to updates the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users.

2.3 MySQL Workbench

MySQL Workbench is a unified visual database designing or graphical user interface tool used for working with database architects, developers, and Database Administrators. It is developed and maintained by Oracle.   It provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. We can use this Server Administration for creating new physical data models, E-R diagrams, and for SQL development (run queries, etc.). It is available for all major operating systems like Mac OS, Windows, and Linux.

With MySQL Workbench, you use an intuitive, browser-based interface, to:

- Administer the database
- Create tables, views, and other database objects
- Import, export, and view table data
- Run queries and SQL scripts
- Generate reports.

2.4 ENTITY RELATIONSHIP DIAGRAM

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example,
in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

- The properties that are used to describe an entity are known as Attributes; for example, an Employee entity may have a Name, Gender, Date of Birth of his/her attributes.
- If book is regarded as an entity then Author's name, Price, published by, etc. are its various attributes.

A specific entity will have a value for each of its attributes. Thus, an entity has a value for each attribute.

A diagram representing entities and relationships among them is known as entity relationship diagram. The major elements used in ER diagram are entities, attributes, identifiers and relationships that express a reality
for which database is designed.

Fig.2.4.1 ER Diagram

ENTITY TYPE:

It symbolizes anything in the real world that has multiple existence.

- **WEAK ENTITY TYPE:** The weak entity in DBMS do not have a primary key and are dependent on
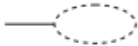
  the parent entity. It mainly depends on other entities.
- **RELATIONSHIP TYPE:** A diamond box is used to represent the relationship between two
  entities. Relationships can be one-to-one, one-to-many or many-to-many.
- **IDENTIFYING RELATIONSHIP TYPE:** The relationship type that is used to relate a weak entity type

  to its owner is shown by double lined diamond shaped box.

ATTRIBUTE:

Entities are represented by means of their properties, called **attributes**. All attributes have values.
For example, a student entity may have name, class, and age as attributes. There exists a domain or range
of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has
to be alphabetic. A student's age cannot be negative, etc.

- **KEY ATTRIBUTE**: A key attribute is one for which each entity has a unique value. It is represented by
  an oval shape with the attribute name underlined.
- **MULTI VALUED ATTRIBUTE**: An entity that has multiple values for that attribute is called multivalued attribute.
- **DERIEVED ATTRIBUTE:** As discussed earlier, an attribute whose value depends upon the value of the stored attribute. It is represented using a dashed oval shape.

In a database system, we deal with various types of keys as follows:

- **CANDIDATE KEY:** Minimal set of attributes that uniquely identifies each occurrence of an entity type.
- **PRIMARY KEY:** Candidate key selected to uniquely identify each occurrence of an entity type.
- **UNIQUE KEY:** Can accept unique of null values.
- **COMPOSITE KEY:** A key that consists of two or more attributes and removal of even one of them
  would result in loss of intended information.

2.5 RELATIONAL SCHEMA

The relational schema is the primary element of the relational database. These databases are managed using language and structure that is consistent with first-order logic. This allows for database management based on entity relationships, making them easy to organize according to volume. Relational schema refers to the meta-data that describes the structure of data within a certain domain. It is the blueprint of a database that outlines the way its structure organizes data into tables. There are two steps to creating a relational database schema: creating the logical schema and creating the physical schema. The logical schema depicts the structure of the database, showing the tables, columns and relationships with other tables in the database and can be created with modeling tools or spreadsheet and drawing software. The physical schema is created by actually generating the tables, columns and relationships in the relational database management software (RDBMS).

2.6 NORMALIZATION

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Here are the most commonly used normal forms.

- First Normal Form
- Second Normal Form
- Third Normal Form
- Boyce & Codd Normal Form.

First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

- It should only have single(atomic) valued attributes/columns.

- Values stored in a column should be of the same domain

- All the columns in a table should have unique names.

- The order in which data is stored, does not matter.

Second Normal Form (2NF)

For a table to be in the Second Normal Form, it should follow the following 4 rules:

- It should be in the First Normal form.

- It should not have Partial Dependency.

Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

- It is in the Second Normal form.

- It doesn't have Transitive Dependency.

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form

- For each functional dependency $(X \rightarrow Y)$, X should be a super key.

2.7 USES OF DBMS

Data that is well organized and integrated is very useful in decision making. We can infer some of the following uses of DBMS.
- Effective and efficient management of data.
- Query processing and management.
- Easy to understand and user friendly.
- Security and integrity of data.
- Better decision making.
- Data sharing and storage.
- Better access to accurate data.
- Ensures error free information.

2.8 APPLICATIONS OF DATABASE

Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers' databases are used to hold administrative information and more specialized data, such as engineering data or economic models.
Databases touch all aspects of our lives. Some of the major areas of application are as follows:

- Banking
- Airlines
- Universities
- E- Commerce
- Human Resources

CHAPTER 3

REQUIREMENT SPECIFICATION

The hardware and software components of a computer system that are required to install and use software efficiently are specified in the SRS. The minimum system requirements need to be met for the programs to run at all times on the system.

3.1 HARDWARE REQUIREMENTS

The hardware requirements specify the necessary hardware which provides us the platform to implement our programs.

- 2.2 GHz processor (Pentium).
- GB RAM (System Memory).
- 20 GB of hard-drive space.
- VGA capable of 1024 x 768 screen resolution.
- Necessary computer peripherals such as keyboard etc.

3.2 SOFTWARE REQUIREMENTS

The software requirement specifies the pre-installed software needed to run the code being implemented in this project.

- Windows Operating System
- MySQL Workbench

CHAPTER 4

DESCRIPTION

A Car Rental Management System is looking to develop a state-of-the-art car rental portfolio management system which is able to track their car rental booking history and billing and car details. This system facilitates the owner of the Car Rental Company to retrieve, update and track and delete the bookings and cars efficiently. At the same time, can utilize this system to monitor their financial management.

Currently, different locations of the car rental company have their own separated systems leading to lack of communication and inefficient data sharing. For example, the car rental company located in Basveshwarnagar uses simple Microsoft Excel to keep the record of their customers, cars and insurance of their cars which is inconvenient to retrieve and update the cars and customer's information. In the car rental company located at Whitefield, maintain a book-based ledger to keep a record of their customers and cars and the insurance for the same. The main branch of the company located in Rajajinagar has to keep the customer and car details of all their branch offices on their own computer system. While each statement serves a distinctive process, there is no coordination, assimilation and representation of data. The systems may have duplicate data which leads to waste of space. The different systems also may have different application programs which cause incompatible files.

Due to these disadvantages of the current system, a car rental management system is proposed. Car Rental Management System is a database management system (DBMS) which is based on computer networks, using the advance database technology to construct, maintain and update various kinds of data in data base system. The DBMS can track and update all the information of the cars available and customers during a particular time span. The major advantages of the DBMS are easy to retrieve and update information, efficient data sharing and communication and reliable backup and security.

4.1 E-R DIAGRAM

Entity	Attributes
Customer	PName, DLNumber, Email, Address, MemberId, PhoneNo
Car	RNo, ModelY, Model, Last_Service, Mileage
Category	CName, No_of_Persons, LCapacity, Cost_Per_Day, CategoryId
Booking	Book_Id, BookD, ReturnD, Amt.
Location	Location_Id, LName, LAddress
Insurance	Insurance_Id, IName, Insurance_Expiry, Coverage_type
Billing	Bill_Id, Total_Amount, Service_Fee, Bill_Date.

Fig. 4.1.1 E-R Diagram
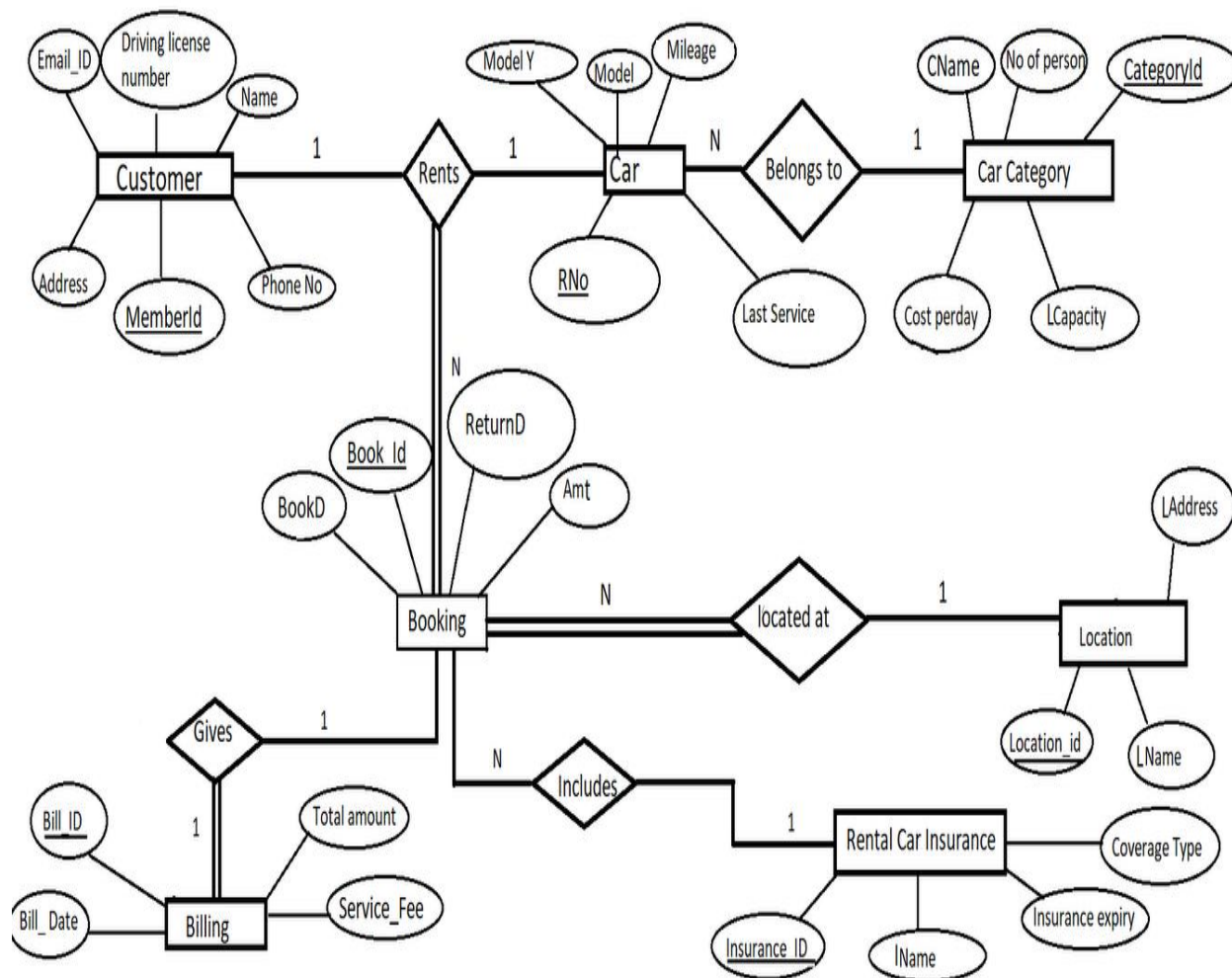
Fig. 4.1.1 E-R diagram for "CAR RENTAL MANAGEMENT SYSTEM"

4.2 RELATIONAL SCHEMA

BOOKING

Amt	ReturnD	BookD	MemberId	Book_Id	Insurance_Id	RNo	Loc_Id

CUSTOMER

PName	MemberId	Email	Address	PhoneNo	DLNumber

CAR

ModelY	Model	Mileage	Last_Service	RNo	CategoryId

CATEGORY

CName	No_of_Persons	LCapacity	Cost_Per_Day	CategoryId

BILL

Bill_Id	Service_Fee	Total_Amount	Bill_Date	Book_Id

INSURANCE

IName	Insurance_Expiry	Coverage_type	Insurance_Id

LOCATION

LName	LAddress	Loc_Id

Fig. 4.2.1 Relational Schema for "CAR RENTAL MANAGEMENT SYSTEM"

BOOKING TABLE:

BOOKING

Amt	ReturnD	BookD	MemberId	Book_Id	Insurance_Id	RNo	Loc_Id

Attributes:

- **Amt:** Every Booking has a basic amount and it cannot be null. It is of type int.
- **ReturnD:** Every Booking has a Return Date and it cannot be null. It is of type date.
- **BookD:** Every Booking has a Booking Date and it cannot be null. It is of type date.
- **MemberId:** This MemberId references Customer MemberId. This is the foreign key in the table. It is of the type int.
- **Book_Id:** Book_Id is uniquely assigned to every Booking. This is the primary key in the table. It is of the type varchar.
- **Insurance_Id:** This Insurance_Id references Insurance Insurance_Id. This is the foreign key in the table. It is of the type varchar**.**
- **RNo:** This RNo references Car RNo. This is the foreign key in the table. It is of the type varchar.
- **Loc_Id:** This Loc_Id references Location Loc_Id. This is the foreign key in the table. It is of the type varchar.

CUSTOMER TABLE:

CUSTOMER

PName	MemberId	Email	Address	PhoneNo	DLNumber

Attributes:

- **PName:** Every Customer must have a name and it cannot be null. It is of the type varchar.
- **MemberId:** MemberId is uniquely assigned to every Customer. This is the primary key in the table. It is of the type int.
- **Email:** Every customer must specify his email and it cannot be null. It is of the type varchar.
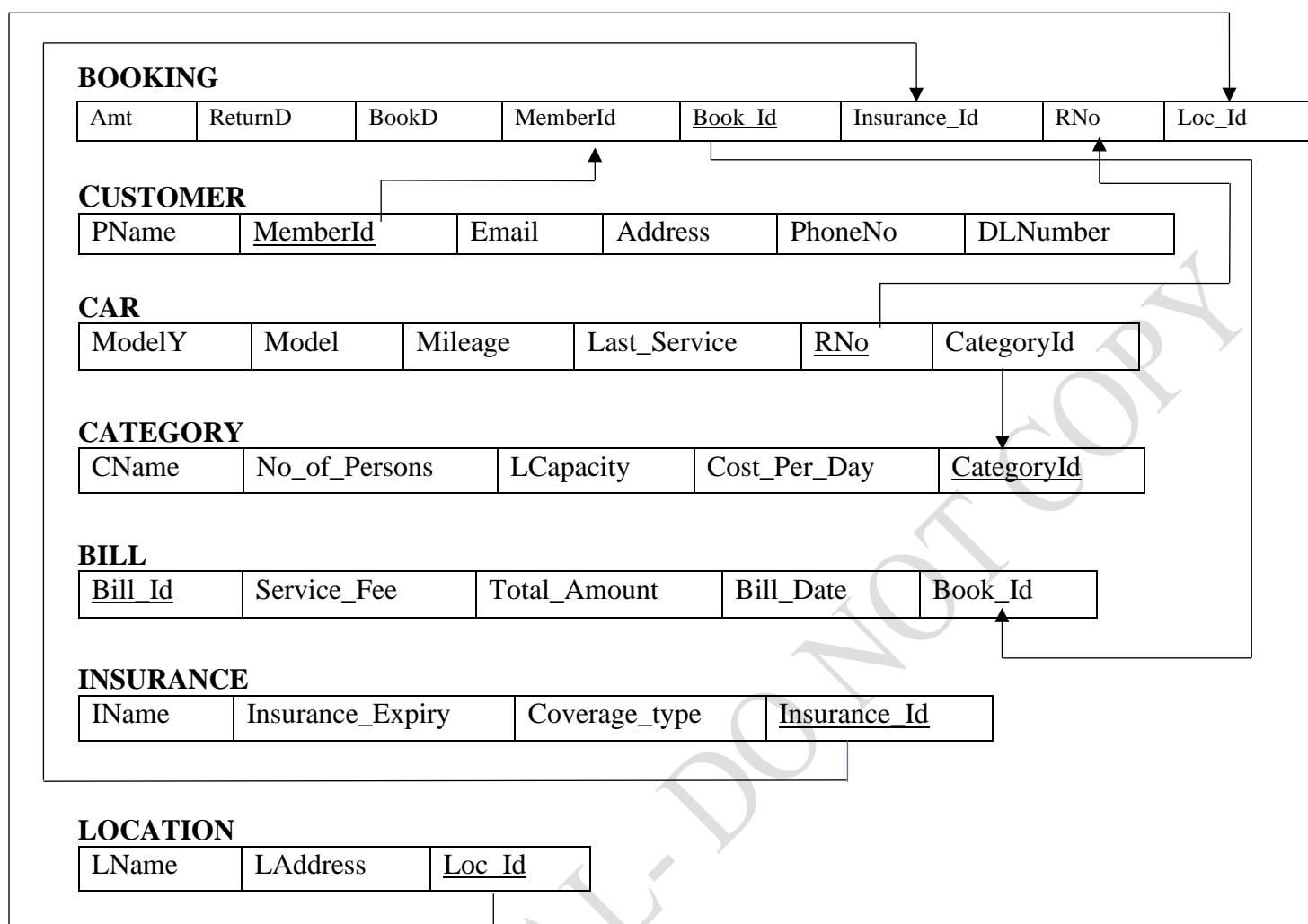- **Address:** Every customer must specify his address and it cannot be null. It is of the type varchar.
- **PhoneNo:** Every customer must specify his phone number and it cannot be null. It is of the type varchar.
- **DLNumber:** Every customer renting a car needs to have a valid driving license and it cannot be null.

CAR TABLE:

CAR

ModelY	Model	Mileage	Last_Service	RNo	CategoryId

Attributes:

- **ModelY:** Every car has a manufacturing year and it cannot be null. It is of the type varchar.
- **Model:** Every car has a name and it cannot be null. It is of the type varchar.
- **Mileage:** Every car has a mileage and it cannot be null. It is of the type varchar.
- **RNo:** RNo is uniquely assigned to every Car. This is the primary key in the table. It is of the type varchar.
- **CategoryId:** This CategoryId references Category CategoryId. This is the foreign key in the table. It is of the type int.

CATEGORY TABLE:

CATEGORY

CName	No_of_Persons	LCapacity	Cost_Per_Day	CategoryId

Attributes:

- **CName:** Every category has a name and it cannot be null. It is of the type varchar;
- **No_of_Persons:** Every category of cars has a number of persons it can hold and it cannot be null. It is of the type int.
- **LCapacity:** Every category of car has a language capacity and it cannot be null. It is of the type varchar.
- **Cost_Per_Day:** Every category of car has a rental cost per day and it cannot be null. It is of the type int.
- **CategoryId:** CategoryId is uniquely assigned to every category of cars. This is the primary key in the table. It is of the type int.

BILL TABLE:

BILL

Bill_Id	Service_Fee	Total_Amount	Bill_Date	Book_Id

Attributes:

- **Bill_Id:** Bill_Id is uniquely assigned to every category of cars. This is the primary key in the table. It is of the type int.
- **Service_Fee:** Every bill has a service fee and it cannot be null. It is of the type varchar.
- **Total_Amount:** Every bill has a total amount and it cannot be null. It is of the type varchar.
- **Bill_Date:** Every bill has a bill date and it cannot be null. It is of the type date.
- **Book_Id:** This Book_Id references Booking Book_Id. This is the foreign key in the table. It is of the type varchar.

INSURANCE TABLE:

INSURANCE

IName	Insurance_Expiry	Coverage_type	Insurance_Id

Attributes:

- **IName:** Every insurance has an insurance company name and it cannot be null. It is of the type varchar.
- **Insurance_Expiry:** Every Insurance has an expiry date and it cannot be null. It is of the type date.
- **Coverage_type:** Every Insurance has a coverage type and it cannot be null. It is of the type varchar.
- **Insurance_Id:** Insurance_Id is uniquely assigned to every insurance. This is the primary key in the table. It is of the type varchar.

LOCATION TABLE:

LOCATION

LName	LAddress	Loc_Id

Attributes:

- **LName:** Every location has a location name and it cannot be null. It is of the type varchar.
- **LAddress:** Every location has a location address and it cannot be null. It is of the type varchar**.**
- **Loc_Id:** Loc_Id is uniquely assigned to every location. This is the primary key in the table. It is of the type int.

CHAPTER 5

CODING

5.1 Table Creation

Category Table:

```
CREATE TABLE CATEGORY (
CName VARCHAR (20),
No_of_Persons INT NOT NULL,
CategoryId INT NOT NULL,
LCapacity INT NOT NULL,
Cost_Per_Day FLOAT NOT NULL,
PRIMARY KEY(CategoryId)
);
```

Field	Type	Null	Key	Default	Extra
CName	varchar(20)	YES		NULL	
No_of_Persons	int	NO		NULL	
CategoryId	int	NO	PRI	NULL	
LCapacity	int	NO		NULL	
Cost_Per_Day	float	NO		NULL	

Car Table:

```
CREATE TABLE CAR (
ModelY INT NOT NULL,
Model VARCHAR (20) NOT NULL,
Last_Service DATE NOT NULL,
Mileage INT NOT NULL,
CategoryId INT,
FOREIGN KEY(CategoryId) REFERENCES CATEGORY(CategoryId),
RNo VARCHAR (20),
PRIMARY KEY (RNo)
);
```

Field	Type	Null	Key	Default	Extra
ModelY	int	NO		NULL	
Model	varchar(20)	NO		NULL	
Last_Service	date	NO		NULL	
Mileage	int	NO		NULL	
CategoryId	int	YES	MUL	NULL	
RNo	varchar(20)	NO	PRI	NULL	

Customer Table:

```
CREATE TABLE CUSTOMER (
PName VARCHAR (20) NOT NULL,
DLNumber VARCHAR (20) NOT NULL,
Email VARCHAR (20) NOT NULL,
Address VARCHAR (40) NOT NULL,
PhoneNo NUMERIC NOT NULL,
MemberId INT,
PRIMARY KEY (MemberId)
);
```

Field	Type	Null	Key	Default	Extra
PName	varchar(20)	NO		NULL	
DLNumber	varchar(20)	NO		NULL	
Email	varchar(20)	NO		NULL	
Address	varchar(40)	NO		NULL	
PhoneNo	decimal(10,0)	NO		NULL	
MemberId	int	NO	PRI	NULL	

Location Table:

```
CREATE TABLE LOCATION (
LName VARCHAR (20) NOT NULL,
Loc_Id INT,
LAddress VARCHAR (40) NOT NULL,
PRIMARY KEY(Loc_Id)
);
```

Field	Type	Null	Key	Default	Extra
LName	varchar(20)	NO		NULL	
Loc_Id	int	NO	PRI	NULL	
LAddress	varchar(40)	NO		NULL	

Insurance Table:

```
CREATE TABLE INSURANCE (
IName VARCHAR (20) NOT NULL,
Coverage_type VARCHAR (20) NOT NULL,
Insurance_Expiry DATE NOT NULL,
InsuranceId VARCHAR (20),
PRIMARY KEY (Insurance_Id)
);
```

Field	Type	Null	Key	Default	Extra
IName	varchar(20)	NO		NULL	
Coverage_type	varchar(20)	NO		NULL	
Insurance_Expiry	date	NO		NULL	
Insurance_Id	varchar(20)	NO	PRI	NULL	

Booking Table:

```
CREATE TABLE BOOKING (
ReturnD DATE NOT NULL,
BookD DATE NOT NULL,
MemberId INT,
FOREIGN KEY (MemberId) REFERENCES CUSTOMER(MemberId),
Loc_Id INT,
FOREIGN KEY (Loc_Id) REFERENCES LOCATION(Loc_Id),
Insurance_Id VARCHAR (20),
FOREIGN KEY (Insurance_Id) REFERENCES INSURANCE(Insurance_Id),
Book_Id VARCHAR (20),
PRIMARY KEY (Book_Id),
RNo VARCHAR (20),
FOREIGN KEY (RNo) REFERENCES CAR(RNo),
Amt FLOAT
);
```

Field	Type	Null	Key	Default	Extra
ReturnD	date	NO		NULL	
BookD	date	NO		NULL	
MemberId	int	YES	MUL	NULL	
Loc_Id	int	YES	MUL	NULL	
Insurance_Id	varchar(20)	YES	MUL	NULL	
Book_Id	varchar(20)	NO	PRI	NULL	
RNo	varchar(20)	YES	MUL	NULL	
Amt	float	YES		NULL	
UsageDays	int	YES		NULL	

Bill Table:

```
CREATE TABLE BILL (
Bill_Id VARCHAR (20),
Service_Fee FLOAT,
Total_Amount FLOAT,
Bill_Date DATE NOT NULL,
Book_Id VARCHAR (20),
PRIMARY KEY (Bill_Id),
FOREIGN KEY (Book_Id) REFERENCES BOOKING(Book_Id)
);
```

Field	Type	Null	Key	Default	Extra
Bill_Id	varchar(20)	NO	PRI	NULL	
Service_Fee	float	YES		NULL	
Total_Amount	float	YES		NULL	
Bill_Date	date	NO		NULL	
Book_Id	varchar(20)	YES	MUL	NULL	

5.2 Value Insertion

Category Table:

INSERT INTO CATEGORY VALUES('HATCHBACK',4,1,230,850);
INSERT INTO CATEGORY VALUES('SEDAN',5,2,400,1050);
INSERT INTO CATEGORY VALUES('SUV',7,3,500,1500);
INSERT INTO CATEGORY VALUES('LUXURY',5,4,450,2000);
INSERT INTO CATEGORY VALUES('CABRIOLET',5,5,300,5000);

SELECT * FROM CATEGORY;

CName	No_of_Persons	CategoryId	LCapacity	Cost_Per_Day
HATCHBACK	4	1	230	850
SEDAN	5	2	400	1050
SUV	7	3	500	1500
LUXURY	5	4	450	2000
CABRIOLET	5	5	300	5000
NULL	NULL	NULL	NULL	NULL

Car Table:

INSERT INTO CAR VALUES (2015,'I20', STR_TO_DATE ('10-01-2020', '%d-%m-%Y'),50000,1,'KA-02-SK-7600');
INSERT INTO CAR VALUES (2016,'POLO', STR_TO_DATE ('10-03-2020', '%d-%m-%Y'),40000,1,'KA-02-SR-8055');
INSERT INTO CAR VALUES (2014,'HONDA CITY', STR_TO_DATE ('24-12-2019', '%d-%m-%Y'),60000,2,'KA-03-MP-1234');
INSERT INTO CAR VALUES (2017,'COROLLA', STR_TO_DATE ('15-10-2019', '%d-%m-%Y'),30000,2,'KA-04-MH-7860');
INSERT INTO CAR VALUES (2018,'BREZZA', STR_TO_DATE ('03-02-2020', '%d-%m-%Y'),20000,3,'KA-02-MH-4722');
INSERT INTO CAR VALUES (2017,'INNOVA', STR_TO_DATE ('07-05-2020', '%d-%m-%Y'),35000,3,'KA-05-CN-3210');
INSERT INTO CAR VALUES (2019,'AUDI A4', STR_TO_DATE ('15-12-2020', '%d-%m-%Y'),10000,4,'KA-01-LX-007');

SELECT * FROM CAR;

ModelY	Model	Last_Service	Mileage	CategoryId	RNo
2019	AUDI A4	2020-12-15	10000	4	KA-01-LX-007
2018	BREZZA	2020-02-03	20000	3	KA-02-MH-4722
2015	I20	2020-01-10	50000	1	KA-02-SK-7600
2016	POLO	2020-03-10	40000	1	KA-02-SR-8055
2014	HONDA CITY	2019-12-24	60000	2	KA-03-MP-1234
2017	COROLLA	2019-10-15	30000	2	KA-04-MH-7860
2017	INNOVA	2020-05-07	35000	3	KA-05-CN-3210
NULL	NULL	NULL	NULL	NULL	NULL

Customer Table:

INSERT INTO CUSTOMER VALUES ('RAJKUMAR','KA02 1234',
'rajkumar@abc.com','RAJAJINAGAR','6362319011',1001);
INSERT INTO CUSTOMER VALUES ('ROHAN','KA02 4321',
'rohan@abc.com','BASAVESHWARNAGAR','7349726140',1002);
INSERT INTO CUSTOMER VALUES ('TUSHARA','KA01 4567', 'tushara@abc.com','CHANDRA
LAYOUT','9844655540',1003);
INSERT INTO CUSTOMER VALUES ('SPANDANA','KA03 0678',
'spandana@abc.com','HEBBAL','8904575820',1004);
INSERT INTO CUSTOMER VALUES ('SRIVATSA','KA04 9867',
'srivatsa@xyz.com','INDIRANAGAR','8296414286 ',1005);
INSERT INTO CUSTOMER VALUES ('RAHUL','KA41 4532',
'rahul@xyz.com','KENGERI','9663131168',1006);
INSERT INTO CUSTOMER VALUES ('SHASHANK','KA02 0510',
'shashank@xyz.com','BASAVESHWARNAGAR','9513801692',1007
);

SELECT * FROM CUSTOMER;

PName	DLNumber	Email	Address	PhoneNo	MemberId
RAJKUMAR	KA02 1234	rajkumar@abc.com	RAJAJINAGAR	6362319011	1001
ROHAN	KA02 4321	rohan@abc.com	BASAVESHWARNAGAR	7349726140	1002
TUSHARA	KA01 4567	tushara@abc.com	CHANDRA LAYOUT	9844655540	1003
SPANDANA	KA03 0678	spandana@abc.com	HEBBAL	8904575820	1004
SRIVATSA	KA04 9867	srivatsa@xyz.com	INDIRANAGAR	8296414286	1005
RAHUL	KA41 4532	rahul@xyz.com	KENGERI	9663131168	1006
SHASHANK	KA02 0510	shashank@xyz.com	BASAVESHWARNAGAR	9513801692	1007
NULL	NULL	NULL	NULL	NULL	NULL

Location Table:

INSERT INTO LOCATION VALUES ('CRMS-NAGARBHAVI',101,'NAGARBHAVI');
INSERT INTO LOCATION VALUES ('CRMS-RAJAJINAGAR',102,'RAJAJINAGAR');
INSERT INTO LOCATION VALUES ('CRMS-WHITEFIELD',103,'WHITEFIELD');
INSERT INTO LOCATION VALUES ('CRMS-MAJESTIC',104,'MAJESTIC');

SELECT * FROM LOCATION;

LName	Loc_Id	LAddress
CRMS-NAGARBHAVI	101	NAGARBHAVI
CRMS-RAJAJINAGAR	102	RAJAJINAGAR
CRMS-WHITEFIELD	103	WHITEFIELD
CRMS-MAJESTIC	104	MAJESTIC
NULL	NULL	NULL

Insurance Table:

INSERT INTO INSURANCE VALUES ('EDELWEISS','CTP', STR_TO_DATE ('10-01-2021', '%d-%m-%Y'),'E01');
INSERT INTO INSURANCE VALUES ('ICICI','CI', STR_TO_DATE ('07-03-2021', '%d-%m-%Y'),'I01');
INSERT INTO INSURANCE VALUES ('KOTAK','FTO', STR_TO_DATE ('15-03-2021', '%d-%m-%Y'),'K01');
INSERT INTO INSURANCE VALUES ('EDELWEISS','CTP', STR_TO_DATE ('18-05-2021', '%d-%m-%Y'),'E02');
INSERT INTO INSURANCE VALUES ('ICICI','CI', STR_TO_DATE ('19-10-2021', '%d-%m-%Y'),'I02');
INSERT INTO INSURANCE VALUES ('KOTAK','FTO', STR_TO_DATE ('02-06-2021', '%d-%m-%Y'),'K02');
INSERT INTO INSURANCE VALUES ('EDELWEISS','CTP', STR_TO_DATE ('05-10-2021', '%d-%m-%Y'),'E03');

SELECT * FROM INSURANCE;

IName	Coverage_type	Insurance_Expiry	Insurance_Id
EDELWEISS	CTP	2021-01-10	E01
EDELWEISS	CTP	2021-05-18	E02
EDELWEISS	CTP	2021-10-05	E03
ICICI	CI	2021-03-07	I01
ICICI	CI	2021-10-19	I02
KOTAK	FTO	2021-03-15	K01
KOTAK	FTO	2021-06-02	K02
NULL	NULL	NULL	NULL

Booking Table:

INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('09-03-2020', '%d-%m-%Y'), STR_TO_DATE ('13-03-2020', '%d-%m-%Y'),1002,101,'E01','CRMS01','KA-02-SK-7600');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('17-04-2020', '%d-%m-%Y'), STR_TO_DATE ('24-04-2020', '%d-%m-%Y'),1003,102,'E02','CRMS02','KA-03-MP-1234');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('21-05-2020', '%d-%m-%Y'), STR_TO_DATE ('22-05-2020', '%d-%m-%Y'),1001,103,'E03','CRMS03','KA-01-LX-007');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('15-11-2020', '%d-%m-%Y'), STR_TO_DATE ('20-11-2020', '%d-%m-%Y'),1004,101,'I01','CRMS04','KA-04-MH-7860');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('02-03-2020', '%d-%m-%Y'), STR_TO_DATE ('06-03-2020', '%d-%m-%Y'),1007,102,'E01','CRMS05','KA-02-SK-7600');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('02-10-2020', '%d-%m-%Y'), STR_TO_DATE ('07-10-2020', '%d-%m-%Y'),1005,104,'K01','CRMS06','KA-05-CN-3210');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('12-12-2020', '%d-%m-%Y'), STR_TO_DATE ('24-12-2020', '%d-%m-%Y'),1006,101,'K02','CRMS07','KA-02-MH-4722');
INSERT INTO BOOKING (ReturnD, BookD, MemberId, Loc_Id, Insurance_Id, Book_Id, RNo) VALUES (STR_TO_DATE ('28-11-2020', '%d-%m-%Y'), STR_TO_DATE ('01-12-2020', '%d-%m-%Y'),1007,103,'I02','CRMS08','KA-02-SR-8055');

SELECT * FROM BOOKING;

ReturnD	BookD	MemberId	Loc_Id	Insurance_Id	Book_Id	RNo	Amt
2020-03-09	2020-03-13	1002	101	E01	CRMS01	KA-02-SK-7600	NULL
2020-04-17	2020-04-24	1003	102	E02	CRMS02	KA-03-MP-1234	NULL
2020-05-21	2020-05-22	1001	103	E03	CRMS03	KA-01-LX-007	NULL
2020-11-15	2020-11-20	1004	101	I01	CRMS04	KA-04-MH-7860	NULL
2020-03-02	2020-03-06	1007	102	E01	CRMS05	KA-02-SK-7600	NULL
2020-10-02	2020-10-07	1005	104	K01	CRMS06	KA-05-CN-3210	NULL
2020-12-12	2020-12-24	1006	101	K02	CRMS07	KA-02-MH-4722	NULL
2020-11-28	2020-12-01	1007	103	I02	CRMS08	KA-02-SR-8055	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Bill Table:

INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B101',200, STR_TO_DATE ('13-03-2020', '%d-%m-%Y'),'CRMS01');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B102',200, STR_TO_DATE ('24-04-2020', '%d-%m-%Y'),'CRMS02');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B103',200, STR_TO_DATE ('22-05-2020', '%d-%m-%Y'),'CRMS03');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B104',200, STR_TO_DATE ('20-11-2020', '%d-%m-%Y'),'CRMS04');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B105',200, STR_TO_DATE ('06-03-2020', '%d-%m-%Y'),'CRMS05');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B106',200, STR_TO_DATE ('07-10-2020', '%d-%m-%Y'),'CRMS06');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B107',200, STR_TO_DATE ('24-12-2020', '%d-%m-%Y'),'CRMS07');
INSERT INTO BILL (Bill_Id, Service_Fee, Bill_Date, Book_Id) VALUES ('B108',200, STR_TO_DATE ('01-12-2020', '%d-%m-%Y'),'CRMS08');

SELECT * FROM BILL;

Bill_Id	Service_Fee	Total_Amount	Bill_Date	Book_Id
B101	200	NULL	2020-03-13	CRMS01
B102	200	NULL	2020-04-24	CRMS02
B103	200	NULL	2020-05-22	CRMS03
B104	200	NULL	2020-11-20	CRMS04
B105	200	NULL	2020-03-06	CRMS05
B106	200	NULL	2020-10-07	CRMS06
B107	200	NULL	2020-12-24	CRMS07
B108	200	NULL	2020-12-01	CRMS08
NULL	NULL	NULL	NULL	NULL

5.3 QUERIES

The most common operation in SQL, the query, makes use of the declarative. SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO Syntax provided in some databases.

Queries allow the user to describe desired data, leaving the database management system (DBMS) to carry out planning, optimizing, and performing the physical operations necessary to produce that result, normally immediately following the SELECT keyword. An asterisk ("*****") can be used to specify that the query should return all columns of the queried tables. Select is the most complex statement in SQL, with optional keywords and clauses that include:

- The **FROM** clause, which indicates the table(s) to retrieve data from. The **FROM** clause can include optional JOIN subclauses to specify the rules for joining tables.
- The **WHERE** clause includes a comparison predicate, which restricts the rows returned by the query. The **WHERE** clause eliminates all rows from the result set where the comparison predicate does not evaluate to True.
- The **GROUP BY** clause projects rows having common values into a smaller set of rows. **GROUP BY** is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The **WHERE** clause is applied before the GROUP BY clause.
- The **HAVING** clause includes a predicate used to filter rows resulting from the **GROUP BY** clause. Because it acts on the results of the **GROUP BY** clause, aggregation functions can be used in the **HAVING** clause predicate.
- The **ORDER BY** clause identifies which column[s] to use to sort the resulting data, and in which direction
  to sort them (ascending or descending). Without an **ORDER BY** clause, the order of rows returned by an SQL query is undefined.
- The **DISTINCT** keyword eliminates duplicate data.

QUERY 1: Calculate the amount in the booking table for every booking based on the number of days of car rented.

SOLUTION:
CREATE TABLE RESULT AS (SELECT CAR. RNo, CAR. CategoryId, CATEGORY. Cost_Per_Day FROM CATEGORY JOIN CAR ON CATEGORY. CategoryId=CAR. CategoryId);
CREATE TABLE RESULT2 (SELECT BOOKING. RNo, RESULT. Cost_Per_Day *DATEDIFF (BookD, ReturnD) AS Amt FROM RESULT JOIN BOOKING ON BOOKING. RNo=RESULT. RNo);
UPDATE BOOKING AS t1
JOIN RESULT2 AS t2 ON t1. RNo= t2. RNo
SET t1. Amt = t2. Amt;
SELECT * FROM BOOKING;

ReturnD	BookD	MemberId	Loc_Id	Insurance_Id	Book_Id	RNo	Amt
2020-03-09	2020-03-13	1002	101	E01	CRMS01	KA-02-SK-7600	3400
2020-04-17	2020-04-24	1003	102	E02	CRMS02	KA-03-MP-1234	7350
2020-05-21	2020-05-22	1001	103	E03	CRMS03	KA-01-LX-007	2000
2020-11-15	2020-11-20	1004	101	I01	CRMS04	KA-04-MH-7860	5250
2020-03-02	2020-03-06	1007	102	E01	CRMS05	KA-02-SK-7600	3400
2020-10-02	2020-10-07	1005	104	K01	CRMS06	KA-05-CN-3210	7500
2020-12-12	2020-12-24	1006	101	K02	CRMS07	KA-02-MH-4722	18000
2020-11-28	2020-12-01	1007	103	I02	CRMS08	KA-02-SR-8055	2550
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

QUERY 2: Calculate the total amount in the bill table for every bill which includes amount from booking table and service fee.

SOLUTION:
UPDATE BILL AS A
JOIN BOOKING AS B ON A. Book_Id = B. Book_Id
SET A. Total_Amount = B. Amt + A. Service_Fee;
SELECT * FROM BILL;

Bill_Id	Service_Fee	Total_Amount	Bill_Date	Book_Id
B101	200	3600	2020-03-13	CRMS01
B102	200	7550	2020-04-24	CRMS02
B103	200	2200	2020-05-22	CRMS03
B104	200	5450	2020-11-20	CRMS04
B105	200	3600	2020-03-06	CRMS05
B106	200	7700	2020-10-07	CRMS06
B107	200	18200	2020-12-24	CRMS07
B108	200	2750	2020-12-01	CRMS08
NULL	NULL	NULL	NULL	NULL

QUERY 3: Delete category from category table of which car category is not present in the car table.

SOLUTION:

    DELETE FROM CATEGORY
    WHERE CategoryId NOT IN (SELECT CategoryId FROM CAR CATEGORY);
    SELECT * FROM CATEGORY;

	CName	No_of_Persons	CategoryId	LCapacity	Cost_Per_Day
▶	HATCHBACK	4	1	230	850
	SEDAN	5	2	400	1050
	SUV	7	3	500	1500
	LUXURY	5	4	450	2000
*	NULL	NULL	NULL	NULL	NULL

QUERY 4: Update Email Id of customer whose DL Number is "KA04 9867".

SOLUTION:

    UPDATE CUSTOMER
    SET Email = "srivatsa@abc.com"
    WHERE DLNumber="KA04 9867";
    SELECT * FROM CUSTOMER;

	PName	DLNumber	Email	Address	PhoneNo	MemberId
▶	RAJKUMAR	KA02 1234	rajkumar@abc.com	RAJAJINAGAR	6362319011	1001
	ROHAN	KA02 4321	rohan@abc.com	BASAVESHWARNAGAR	7349726140	1002
	TUSHARA	KA01 4567	tushara@abc.com	CHANDRA LAYOUT	9844655540	1003
	SPANDANA	KA03 0678	spandana@abc.com	HEBBAL	8904575820	1004
	SRIVATSA	KA04 9867	srivatsa@abc.com	INDIRANAGAR	8296414286	1005
	RAHUL	KA41 4532	rahul@xyz.com	KENGERI	9663131168	1006
	SHASHANK	KA02 0510	shashank@xyz.com	BASAVESHWARNAGAR	9513801692	1007
*	NULL	NULL	NULL	NULL	NULL	NULL

QUERY 5: Retrieve the count of Insurance from each company.

SOLUTION:

    SELECT IName,
    COUNT(Insurance_Id) FROM INSURANCE
    GROUP BY IName;

	IName	COUNT(Insurance_Id)
▶	EDELWEISS	3
	ICICI	2
	KOTAK	2

CONCLUSION

The project Car Rental Management System (CRMS) is for computerizing the working in a car rental company. The software takes care of all the requirements of an average car rental company and is capable to provide easy and effective storage of information related to cars and customers of the car rental company.

It generates bills, provides car details including car insurance and service details. It also provides customer details such as driving license number, membership id, name etc. The system also provides the facility of backup as per requirement.

BIBLIOGRAPHY

Few of the book(s) and websites that were instrumental in helping us to complete this project are as mentioned below.

BOOK

1. Fundamental of Database System by Elmasri and Navathe ,5th Edition, Addison-Wesley,2007.
2. Database System Concepts by Avi Silberschatz, Henry F Korth, and S. Sudharshan,1996.
3. Concepts of Database Management by Philip J. Pratt,2008
4. Modern Database Management by Jeffery A Hoffer,2010

URL

1. https://www.w3schools.com
2. https://www.youtube.com
3. https://www.google.co.in
4. https://www.wikipedia.org