Project Report

Car Commerce Center

by

Rishi Joshi, Roll number: T055

Mayank Bhardwaj, Roll number: T043

Course: DBMS

AY: 2023-24

Table of Contents

I. Storyline

Database Requirements:

User Management:

- Registration: Users should be able to register for an account on CCC by providing essential information such as email, password, and contact details.
- User Profiles: Registered users should have personalized profiles where they can manage their information and preferences.
- Authentication: The database must support secure authentication mechanisms to verify user identities during login.
- Role-Based Access Control: Differentiate between buyers and sellers, each with distinct privileges and capabilities within the system.

Car Listings Management:

- Car Information: Sellers should be able to create listings for their cars, providing details such as make, model, year, mileage, price, and photos.
- Categorization: Cars should be categorized based on attributes like type (e.g., sedan, SUV), fuel type (e.g., petrol, diesel), and transmission type (e.g., automatic, manual).
- Search and Filter: Users should be able to search for cars based on various criteria such as make, model, year, price range, etc., with filtering options to refine search results.
- Featured Listings: Allow sellers to promote certain listings as featured or highlighted for increased visibility.

Transaction Management:

- Buying Process: Enable users to initiate purchases directly through CCC, facilitating communication and negotiation between buyers and sellers.
- Order Tracking: Provide a mechanism for users to track the status of their orders, from initiation to completion.

- Payment Integration: Integrate payment gateways to facilitate secure online transactions, allowing users to make payments for purchased cars directly through CCC.

Security:

- Data Encryption: Implement encryption techniques to safeguard sensitive user data, such as passwords and financial information.
- Access Control: Enforce strict access controls to ensure that only authorised users can view, modify, or delete data within the database.
- Secure Communication: Utilise secure communication protocols (e.g., HTTPS) to protect data transmission between clients and the server.

Performance and Scalability:

- Efficient Queries: Optimise database queries to ensure fast retrieval of car listings and user information, even as the database grows.
- Scalability: Design the database to scale seamlessly with increasing user and listing volumes, ensuring consistent performance under high loads.

II. Components of Database Design

Entities:

Car Entity:

- CarID (Primary Key)
- Model
- Company
- MakeYear
- RCNo
- Insurance
- UserEmail (Foreign Key)
- EnginePower

- Colour
- CarTypeID (Foreign Key)
- FuelTypeID (Foreign Key)
- TransmissionID (Foreign Key)

CarType Entity:

- CarTypeID (Primary Key)
- Type

FuelType Entity:

- FuelTypeID (Primary Key)
- FuelType

Transmission Entity:

- TransmissionID (Primary Key)
- Transmission

User Entity:

- Email (Primary Key)
- Password
- PhoneNo

Seller Entity:

- AadharNo (Primary Key)
- Name
- Address
- PhoneNo
- DOB
- Email (Foreign Key)
- DLNo

Buyer Entity:

- AadharNo (Primary Key)
- Name

- Address
- PhoneNo
- DOB
- Email (Foreign Key)
- DLNo

Order Entity:

- OrderID (Primary Key)
- CarID (Foreign Key)
- UserEmail (Foreign Key)
- SellerAadharNo (Foreign Key)
- BuyerAadharNo (Foreign Key)
- Date
- Price

In this schema:

- Each entity represents a distinct concept within the car commerce system.
- Attributes within each entity describe the properties or characteristics of that concept.
- Primary keys uniquely identify each record within the entity.
- Foreign keys establish relationships between entities, linking records together.

Relationships among various entities:

Car - CarType (Many-to-One):

- Cars can have only one CarType.
- CarType can be associated with many Cars.
- Participation: Mandatory on the Car side (each Car must have a CarType), optional on the CarType side (a CarType may not be associated with any Cars).
- Cardinality: One CarType to many Cars.

Car - FuelType (Many-to-One):

- Cars can have only one FuelType.
- FuelType can be associated with many Cars.
- Participation: Mandatory on the Car side (each Car must have a FuelType), optional on the FuelType side (a FuelType may not be associated with any Cars).
- Cardinality: One FuelType to many Cars.

Car - Transmission (Many-to-One):

- Cars can have only one Transmission type.
- Transmission type can be associated with many Cars.
- Participation: Mandatory on the Car side (each Car must have a Transmission type), optional on the Transmission side (a Transmission type may not be associated with any Cars).
- Cardinality: One Transmission type to many Cars.

User - Seller (One-to-One):

- Each Seller is a User.
- Each User can be a Seller.
- Participation: Mandatory on both sides (each Seller must be a User, and each User must be a Seller).
- Cardinality: One User to one Seller.

User - Buyer (One-to-One):

- Each Buyer is a User.
- Each User can be a Buyer.
- Participation: Mandatory on both sides (each Buyer must be a User, and each User must be a Buyer).
- Cardinality: One User to one Buyer.

Seller - Order (One-to-Many):

- Each Seller can have multiple Orders.
- Each Order is associated with one Seller.
- Participation: Mandatory on the Order side (each Order must have a Seller), optional on the Seller side (a Seller may not have any Orders).
- Cardinality: One Seller to many Orders.

Buyer - Order (One-to-Many):

- Each Buyer can have multiple Orders.
- Each Order is associated with one Buyer.
- Participation: Mandatory on the Order side (each Order must have a Buyer), optional on the Buyer side (a Buyer may not have any Orders).
- Cardinality: One Buyer to many Orders.

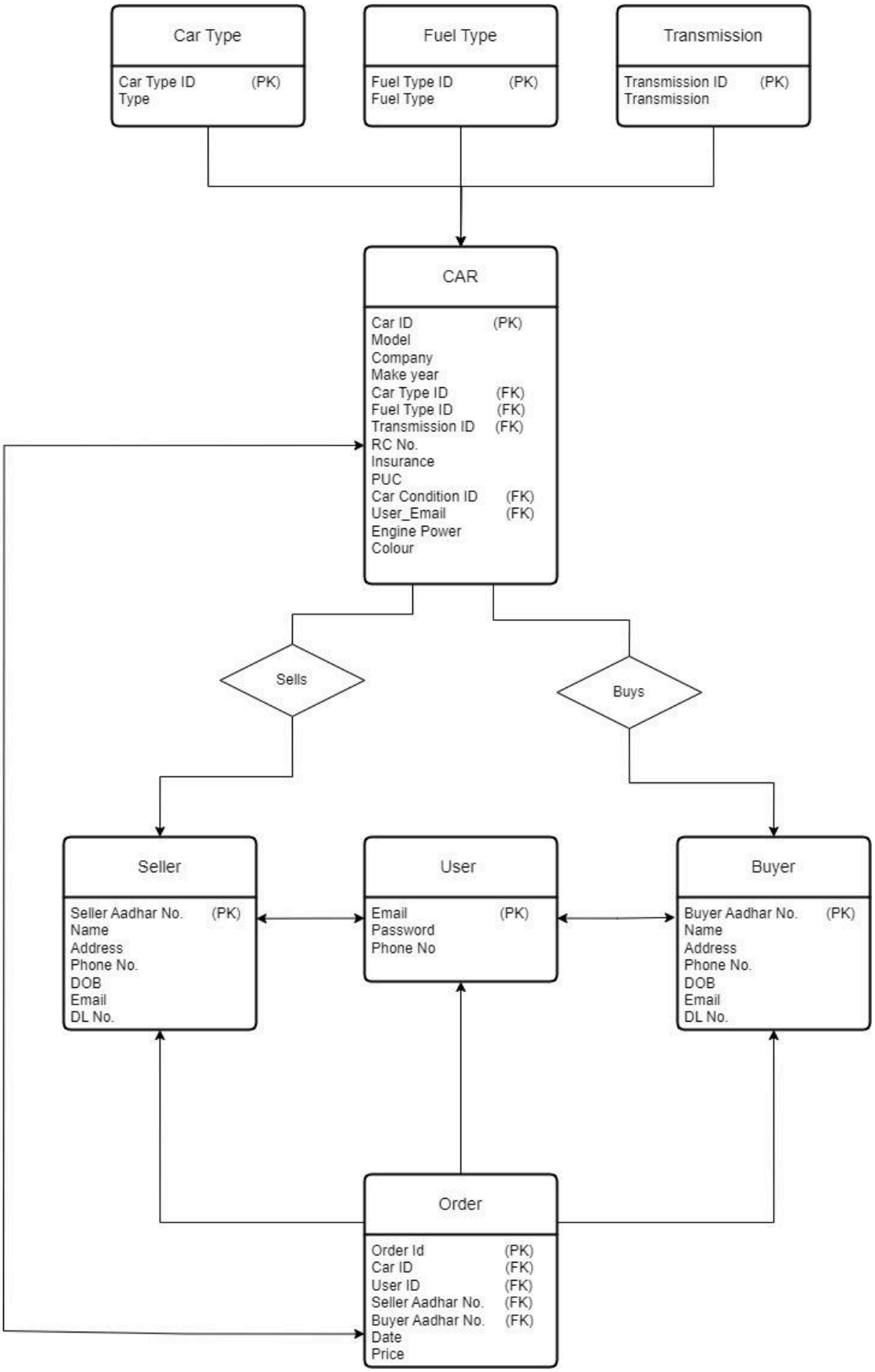
Car - Order (One-to-One):

- Each Car is associated with only one Order.
- Each Order is associated with only one Car.
- Participation: Mandatory on both sides (each Car must have one Order, and each Order must have one Car).
- Cardinality: One Car to one Order.

User - Order (One-to-Many):

- Each User can create multiple Orders.
- Each Order is associated with one User.
- Participation: Mandatory on the Order side (each Order must have a User), optional on the User side (a User may not have any Orders).
- Cardinality: One User to many Orders.

III. Entity Relationship Diagram



IV. Relational Model

Relational Model represented with tables:

Car table:

Field	Data Type	Key
CarID	INT	PK
Model	VARCHAR	
Company	VARCHAR	
MakeYear	INT	
RCNo	VARCHAR	
Insurance	VARCHAR	
UserEmail	VARCHAR	FK
EnginePower	INT	
Colour	VARCHAR	
CarTypeID	INT	FK
FuelTypeID	INT	FK
TransmissionID	INT	FK

CarType Table:

Field	Data Type	Key
CarTypeID	INT	PK
Type	VARCHAR	

FuelType Table:

Field	Data Type	Key
FuelTypeID	INT	PK
FuelType	VARCHAR	

Transmission Table:

Field	Data Type	Key
TransmissionID	INT	PK
Transmission	VARCHAR	

User Table:

Field	Data Type	Key
Email	VARCHAR	PK
Password	VARCHAR	
PhoneNo	VARCHAR	

Seller Table:

Field	Data Type	Key
AadharNo	VARCHAR	PK
Name	VARCHAR	
Address	VARCHAR	
PhoneNo	VARCHAR	
DOB	DATE	
Email	VARCHAR	FK
DLNo	VARCHAR	

Buyer Table:

Field	Data Type	Key
AadharNo	VARCHAR	PK
Name	VARCHAR	
Address	VARCHAR	
PhoneNo	VARCHAR	
DOB	DATE	
Email	VARCHAR	FK
DLNo	VARCHAR	

Order Table:

Field	Data Type	Key
OrderID	INT	PK
CarID	INT	FK
UserEmail	VARCHAR	FK
SellerAadharNo	VARCHAR	FK
BuyerAadharNo	VARCHAR	FK
Date	DATE	
Price	DECIMAL	

In this relational model, each table represents a distinct entity, and the fields within each table represent the attributes of that entity. Primary keys (PK) uniquely identify records within each table, while foreign keys (FK) establish relationships between tables. This model effectively organises the database schema and its relationships in a tabular format.

V. Normalization

The database schema consists of five tables: Car, User, Seller, Buyer, and Orders. The tables represent entities and relationships related to CCC.

1. First Normal Form (1NF):

The schema is satisfying 1NF. Each table has a primary key, and there are no repeating groups within the tables. All cells hold atomic values.

2. Second Normal Form (2NF):

The schema is satisfying 2NF. Each non-key attribute in the tables appears to be fully dependent on the primary key. There are no partial dependencies.

3. Third Normal Form (3NF):

The schema seems to exhibit some violations of 3NF, particularly in the Orders table. The Order table includes attributes such as UserEmail, SellerAadharNo, and BuyerAadharNo, which are transitively dependent on the OrderID. To achieve 3NF, these attributes should be moved to separate tables.

Issues Identified:

Transitive Dependency in Orders Table:
The Orders table has transitive dependencies involving UserEmail, SellerAadharNo, and BuyerAadharNo. These attributes should be moved to separate tables to achieve 3NF.

```
1
2   CREATE TABLE CarType (
3       CarTypeID INT PRIMARY KEY,
4       Type VARCHAR(50)
5   );
6
7   CREATE TABLE FuelType (
8       FuelTypeID INT PRIMARY KEY,
9       FuelType VARCHAR(50)
10  );
11
12  CREATE TABLE Transmission (
13      TransmissionID INT PRIMARY KEY,
14      Transmission VARCHAR(50)
15  );
```

These tables are designed to store specific attributes related to car types, fuel types, and transmission types, respectively. Each table features a primary key field (CarTypeID, FuelTypeID, TransmissionID) to uniquely identify each record, alongside a descriptive field (Type, FuelType, Transmission) to capture the corresponding type information.

```
18 •  ALTER TABLE Car
19    ADD COLUMN CarTypeID INT,
20    ADD COLUMN FuelTypeID INT,
21    ADD COLUMN TransmissionID INT,
22    ADD FOREIGN KEY (CarTypeID) REFERENCES CarType(CarTypeID),
23    ADD FOREIGN KEY (FuelTypeID) REFERENCES FuelType(FuelTypeID),
24    ADD FOREIGN KEY (TransmissionID) REFERENCES Transmission(TransmissionID);
25
26    -- Drop redundant columns from the Car table
27    ALTER TABLE Car
28    DROP COLUMN CarType,
29    DROP COLUMN FuelType,
30    DROP COLUMN Transmission;
```

To further enhance the normalization of our database schema, we have implemented alterations to the existing Car table structure. These alterations aim to reduce redundancy and improve data integrity by segregating attributes into separate tables and establishing foreign key relationships.

Firstly, we have expanded the Car table by adding three new columns: CarTypeID, FuelTypeID, and TransmissionID. These columns will store references to the corresponding types of cars, fuel, and transmission stored in the CarType, FuelType, and Transmission tables, respectively. Concurrently, foreign key constraints have been applied to enforce referential integrity, ensuring that each entry in the Car table references a valid entry in the associated tables.

Following the addition of these columns and establishment of foreign key relationships, redundant columns (CarType, FuelType, Transmission) have been removed from the Car table. By eliminating redundant data and establishing normalized relationships, we optimize data storage and maintain consistency within our database.

VI. SQL Queries

Creating Tables:

```
create database car;
use car;

CREATE TABLE Car (
        CarID INT PRIMARY KEY,
        Model VARCHAR(255),
        Company VARCHAR(255),
        MakeYear INT,
        CarType VARCHAR(50),
        FuelType VARCHAR(50),
        Transmission VARCHAR(50),
        RCNo VARCHAR(50),
        Insurance VARCHAR(50),
        UserEmail VARCHAR(255),
        EnginePower INT,
        Colour VARCHAR(50)
);

CREATE TABLE User (
        Email VARCHAR(255) PRIMARY KEY,
        Password VARCHAR(255),
        PhoneNo VARCHAR(20)
);

CREATE TABLE Seller (
        AadharNo VARCHAR(20) PRIMARY KEY,
        Name VARCHAR(255),
        Address VARCHAR(255),
        PhoneNo VARCHAR(20),
        DOB DATE,
        Email VARCHAR(255),
        DLNo VARCHAR(50)
);

CREATE TABLE Buyer (
        AadharNo VARCHAR(20) PRIMARY KEY,
        Name VARCHAR(255),
        Address VARCHAR(255),
        PhoneNo VARCHAR(20),
        DOB DATE,
        Email VARCHAR(255),
        DLNo VARCHAR(50)
);
```

```
CREATE TABLE Orders (
        OrderID INT PRIMARY KEY,
        CarID INT,
        UserEmail VARCHAR(255),
        SellerAadharNo VARCHAR(20),
        BuyerAadharNo VARCHAR(20),
        OrderDate DATE,
        Price DECIMAL(10, 2)
);
```

Tables_in_car
buyer
car
cartype
fueltype
orders
seller
transmission
user

Normalization:

```
CREATE TABLE CarType (
        CarTypeID INT PRIMARY KEY,
        Type VARCHAR(50)
);

CREATE TABLE FuelType (
        FuelTypeID INT PRIMARY KEY,
        FuelType VARCHAR(50)
);

CREATE TABLE Transmission (
        TransmissionID INT PRIMARY KEY,
        Transmission VARCHAR(50)
);


ALTER TABLE Car
ADD COLUMN CarTypeID INT,
ADD COLUMN FuelTypeID INT,
ADD COLUMN TransmissionID INT,
ADD FOREIGN KEY (CarTypeID) REFERENCES CarType(CarTypeID),
ADD FOREIGN KEY (FuelTypeID) REFERENCES FuelType(FuelTypeID),
ADD FOREIGN KEY (TransmissionID) REFERENCES Transmission(TransmissionID);
```

```
-- Drop redundant columns from the Car table
ALTER TABLE Car
DROP COLUMN CarType,
DROP COLUMN FuelType,
DROP COLUMN Transmission;
```

Populating the Tables:

```
-- Populate CarType table
INSERT INTO CarType (CarTypeID, Type)
VALUES (1, 'Hatchback'),
    (2, 'Sedan'),
    (3, 'SUV'),
    (4, 'MPV');

-- Populate FuelType table
INSERT INTO FuelType (FuelTypeID, FuelType)
VALUES (1, 'Petrol'),
    (2, 'Diesel'),
    (3, 'Electric');

-- Populate Transmission table
INSERT INTO Transmission (TransmissionID, Transmission)
VALUES (1, 'Manual'),
    (2, 'Automatic');

-- Populate User table
INSERT INTO User (Email, Password, PhoneNo)
VALUES ('user1@gmail.com', 'password1', '9876543210'),
    ('user2@gmail.com', 'password2', '9876543211'),
    ('user3@gmail.com', 'password3', '9876543212'),
    ('user4@gmail.com', 'password4', '9876543213'),
    ('user5@gmail.com', 'password5', '9876543214');

-- Populate Seller table
INSERT INTO Seller (AadharNo, Name, Address, PhoneNo, DOB, Email, DLNo)
VALUES ('123456789012', 'Ramesh Sharma', '123, ABC Street, XYZ City', '9876543215', '1980-05-10',
'seller1@gmail.com', 'DL123456'),
    ('234567890123', 'Suresh Kumar', '456, DEF Street, XYZ City', '9876543216', '1975-08-20',
'seller2@gmail.com', 'DL234567'),
    ('345678901234', 'Priya Singh', '789, GHI Street, XYZ City', '9876543217', '1988-02-15', 'seller3@gmail.com',
'DL345678'),
    ('456789012345', 'Neha Patel', '012, JKL Street, XYZ City', '9876543218', '1992-11-30', 'seller4@gmail.com',
'DL456789'),
    ('567890123456', 'Amit Verma', '345, MNO Street, XYZ City', '9876543219', '1983-07-25',
'seller5@gmail.com', 'DL567890');
```

```
-- Populate Buyer table
INSERT INTO Buyer (AadharNo, Name, Address, PhoneNo, DOB, Email, DLNo)
VALUES ('678901234567', 'Anjali Gupta', '901, PQR Street, XYZ City', '9876543220', '1986-04-05',
'buyer1@gmail.com', 'DL678901'),
    ('789012345678', 'Manoj Reddy', '234, STU Street, XYZ City', '9876543221', '1990-09-15',
'buyer2@gmail.com', 'DL789012'),
    ('890123456789', 'Shalini Shah', '567, VWX Street, XYZ City', '9876543222', '1982-12-25',
'buyer3@gmail.com', 'DL890123'),
    ('901234567890', 'Rajesh Singhania', '890, YZA Street, XYZ City', '9876543223', '1978-06-20',
'buyer4@gmail.com', 'DL901234'),
    ('012345678901', 'Deepak Sharma', '123, BCD Street, XYZ City', '9876543224', '1995-03-10',
'buyer5@gmail.com', 'DL012345');

-- Populate Car table
INSERT INTO Car (CarID, Model, Company, MakeYear, CarTypeID, FuelTypeID, TransmissionID, RCNo, Insurance,
UserEmail, EnginePower, Colour)
VALUES (1, 'Maruti Swift', 'Maruti Suzuki', 2019, 1, 1, 1, 'RC1234', 'Yes', 'seller1@gmail.com', 1200, 'Red'),
    (2, 'Hyundai i20', 'Hyundai', 2020, 1, 1, 2, 'RC2345', 'Yes', 'seller2@gmail.com', 1400, 'Blue'),
    (3, 'Honda City', 'Honda', 2021, 2, 1, 2, 'RC3456', 'Yes', 'seller3@gmail.com', 1500, 'White'),
    (4, 'Mahindra XUV500', 'Mahindra', 2018, 3, 2, 2, 'RC4567', 'Yes', 'seller4@gmail.com', 2000, 'Black'),
    (5, 'Toyota Innova Crysta', 'Toyota', 2020, 4, 2, 1, 'RC5678', 'Yes', 'seller5@gmail.com', 2200, 'Silver');

-- Populate Orders table
INSERT INTO Orders (OrderID, CarID, UserEmail, SellerAadharNo, BuyerAadharNo, OrderDate, Price)
VALUES (1, 1, 'buyer1@gmail.com', '123456789012', '678901234567', '2024-03-20', 650000),
    (2, 2, 'buyer2@gmail.com', '234567890123', '789012345678', '2024-03-21', 750000),
    (3, 3, 'buyer3@gmail.com', '345678901234', '890123456789', '2024-03-22', 850000),
    (4, 4, 'buyer4@gmail.com', '456789012345', '901234567890', '2024-03-23', 950000),
    (5, 5, 'buyer5@gmail.com', '567890123456', '012345678901', '2024-03-24', 1050000);
```

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
1	Maruti Swift	Maruti Suzuki	2019	RC1234	Yes	seller1@gmail.com	1200	Red	1	1	1
2	Hyundai i20	Hyundai	2020	RC2345	Yes	seller2@gmail.com	1400	Blue	1	1	2
3	Honda City	Honda	2021	RC3456	Yes	seller3@gmail.com	1500	White	2	1	2
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

OrderID	CarID	UserEmail	SellerAadharNo	BuyerAadharNo	OrderDate	Price
1	1	buyer1@gmail.com	123456789012	678901234567	2024-03-20	650000.00
2	2	buyer2@gmail.com	234567890123	789012345678	2024-03-21	750000.00
3	3	buyer3@gmail.com	345678901234	890123456789	2024-03-22	850000.00
4	4	buyer4@gmail.com	456789012345	901234567890	2024-03-23	950000.00
5	5	buyer5@gmail.com	567890123456	012345678901	2024-03-24	1050000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

AadharNo	Name	Address	PhoneNo	DOB	Email	DLNo
012345678901	Deepak Sharma	123, BCD Street, XYZ City	9876543224	1995-03-10	buyer5@gmail.com	DL012345
678901234567	Anjali Gupta	901, PQR Street, XYZ City	9876543220	1986-04-05	buyer1@gmail.com	DL678901
789012345678	Manoj Reddy	234, STU Street, XYZ City	9876543221	1990-09-15	buyer2@gmail.com	DL789012
890123456789	Shalini Shah	567, VWX Street, XYZ City	9876543222	1982-12-25	buyer3@gmail.com	DL890123
901234567890	Rajesh Singhania	890, YZA Street, XYZ City	9876543223	1978-06-20	buyer4@gmail.com	DL901234
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL queries:

1. **Retrieve all cars from the Car table:**

SELECT * FROM Car;

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
1	Maruti Swift	Maruti Suzuki	2019	RC1234	Yes	seller1@gmail.com	1200	Red	1	1	1
2	Hyundai i20	Hyundai	2020	RC2345	Yes	seller2@gmail.com	1400	Blue	1	1	2
3	Honda City	Honda	2021	RC3456	Yes	seller3@gmail.com	1500	White	2	1	2
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. **Retrieve distinct car types from the Car table:**

SELECT * FROM Car WHERE MakeYear > 2015;

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
1	Maruti Swift	Maruti Suzuki	2019	RC1234	Yes	seller1@gmail.com	1200	Red	1	1	1
2	Hyundai i20	Hyundai	2020	RC2345	Yes	seller2@gmail.com	1400	Blue	1	1	2
3	Honda City	Honda	2021	RC3456	Yes	seller3@gmail.com	1500	White	2	1	2
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. **Retrieve cars with a specific fuel type:**

SELECT * FROM Car WHERE FuelTypeID = '2';

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. **Retrieve cars ordered by their make year in descending order:**

SELECT * FROM Car ORDER BY MakeYear DESC;

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
3	Honda City	Honda	2021	RC3456	Yes	seller3@gmail.com	1500	White	2	1	2
2	Hyundai i20	Hyundai	2020	RC2345	Yes	seller2@gmail.com	1400	Blue	1	1	2
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1
1	Maruti Swift	Maruti Suzuki	2019	RC1234	Yes	seller1@gmail.com	1200	Red	1	1	1
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. **Retrieve the count of cars for each car type:**

SELECT CarTypeID, COUNT(*) AS TotalCars FROM Car GROUP BY CarTypeID;

CarTypeID	TotalCars
1	2
2	1
3	1
4	1

6. **Retrieve the average engine power of cars:**

SELECT AVG(EnginePower) AS AvgEnginePower FROM Car;

AvgEnginePower
1660.0000

7. **Retrieve cars along with their company and model:**

SELECT Company, Model FROM Car;

Company	Model
Maruti Suzuki	Maruti Swift
Hyundai	Hyundai i20
Honda	Honda City
Mahindra	Mahindra XUV500
Toyota	Toyota Innova Crysta

8. **Retrieve sellers along with their email and phone number:**

SELECT Email, PhoneNo FROM Seller;

Email	PhoneNo
seller1@gmail.com	9876543215
seller2@gmail.com	9876543216
seller3@gmail.com	9876543217
seller4@gmail.com	9876543218
seller5@gmail.com	9876543219

9. **Retrieve buyers along with their names and addresses:**

SELECT Name, Address FROM Buyer;

Name	Address
Deepak Sharma	123, BCD Street, XYZ City
Anjali Gupta	901, PQR Street, XYZ City
Manoj Reddy	234, STU Street, XYZ City
Shalini Shah	567, VWX Street, XYZ City
Rajesh Singhania	890, YZA Street, XYZ City

10. **Insert a new car into the Car table:**

INSERT INTO Car (CarID, Model, Company, MakeYear, CarTypeID, FuelTypeID, TransmissionID, RCNo, Insurance, UserEmail, EnginePower, Colour) VALUES (11, 'Maruti Swift', 'Maruti Suzuki', 2021, 1,1,1, 'RC5678', 'Yes', 'buyer1@gmail.com', 1200, 'Red');

11	Maruti Swift	Maruti Suzuki	2021	RC5678	Yes	buyer1@gmail.com	1200	Red	1	1	1

11. **Update the make year of a specific car:**

UPDATE Car SET MakeYear = 2019 WHERE CarID = 2;

2	Hyundai i20	Hyundai	2019

12. **Delete a specific order from the Orders table:**

DELETE FROM Orders WHERE OrderID = 3;

OrderID	CarID	UserEmail	SellerAadharNo	BuyerAadharNo	OrderDate	Price
1	1	buyer1@gmail.com	123456789012	678901234567	2024-03-20	650000.00
2	2	buyer2@gmail.com	234567890123	789012345678	2024-03-21	750000.00
4	4	buyer4@gmail.com	456789012345	901234567890	2024-03-23	950000.00
5	5	buyer5@gmail.com	567890123456	012345678901	2024-03-24	1050000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

13. **Retrieve orders made by a specific user:**

SELECT * FROM Orders WHERE UserEmail = 'buyer1@gmail.com';

OrderID	CarID	UserEmail	SellerAadharNo	BuyerAadharNo	OrderDate	Price
1	1	buyer1@gmail.com	123456789012	678901234567	2024-03-20	650000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

14. **Retrieve orders placed after a specific date:**

SELECT * FROM Orders WHERE OrderDate > '2023-01-01';

OrderID	CarID	UserEmail	SellerAadharNo	BuyerAadharNo	OrderDate	Price
1	1	buyer1@gmail.com	123456789012	678901234567	2024-03-20	650000.00
2	2	buyer2@gmail.com	234567890123	789012345678	2024-03-21	750000.00
4	4	buyer4@gmail.com	456789012345	901234567890	2024-03-23	950000.00
5	5	buyer5@gmail.com	567890123456	012345678901	2024-03-24	1050000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

15. **Join Car and Orders tables to retrieve details of ordered cars:**

SELECT c.Model, c.Company, o.OrderDate, o.Price FROM Car c JOIN Orders o ON c.CarID = o.CarID;

Model	Company	OrderDate	Price
Maruti Swift	Maruti Suzuki	2024-03-20	650000.00
Hyundai i20	Hyundai	2024-03-21	750000.00
Mahindra XUV500	Mahindra	2024-03-23	950000.00
Toyota Innova Crysta	Toyota	2024-03-24	1050000.00

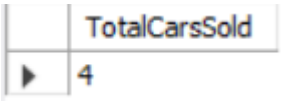
16. **Retrieve sellers who have sold cars with a price greater than 50000:**

SELECT s.Name, s.Email FROM Seller s JOIN Orders o ON s.AadharNo = o.SellerAadharNo WHERE o.Price > 50000;

Name	Email
Ramesh Sharma	seller1@gmail.com
Suresh Kumar	seller2@gmail.com
Neha Patel	seller4@gmail.com
Amit Verma	seller5@gmail.com

17. Retrieve the total number of cars sold:

SELECT COUNT(*) AS TotalCarsSold FROM Orders;



TotalCarsSold
4

18. Select cars along with their respective sellers:

SELECT Car.*, Seller.Name AS SellerName FROM Car
INNER JOIN Seller ON Car.UserEmail = Seller.Email;

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID	SellerName
1	Maruti Swift	Maruti Suzuki	2019	RC1234	Yes	seller1@gmail.com	1200	Red	1	1	1	Ramesh Sharma
2	Hyundai i20	Hyundai	2019	RC2345	Yes	seller2@gmail.com	1400	Blue	1	1	2	Suresh Kumar
3	Honda City	Honda	2021	RC3456	Yes	seller3@gmail.com	1500	White	2	1	2	Priya Singh
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2	Neha Patel
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1	Amit Verma

19. Find the oldest car in the database:

SELECT * FROM Car ORDER BY MakeYear ASC LIMIT 1;

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

20. Select cars with insurance status 'Yes':

SELECT * FROM Car WHERE Insurance = 'Yes';

CarID	Model	Company	MakeYear	RCNo	Insurance	UserEmail	EnginePower	Colour	CarTypeID	FuelTypeID	TransmissionID
1	Maruti Swift	Maruti Suzuki	2019	RC1234	Yes	seller1@gmail.com	1200	Red	1	1	1
2	Hyundai i20	Hyundai	2019	RC2345	Yes	seller2@gmail.com	1400	Blue	1	1	2
3	Honda City	Honda	2021	RC3456	Yes	seller3@gmail.com	1500	White	2	1	2
4	Mahindra XUV500	Mahindra	2018	RC4567	Yes	seller4@gmail.com	2000	Black	3	2	2
5	Toyota Innova Crysta	Toyota	2020	RC5678	Yes	seller5@gmail.com	2200	Silver	4	2	1
11	Maruti Swift	Maruti Suzuki	2021	RC5678	Yes	buyer1@gmail.com	1200	Red	1	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

VI. Project demonstration

Tools/software/ libraries used:

MySql:

MySQL is a popular open-source relational database management system (RDBMS) known for its reliability, scalability, and performance.
MySQL is used for managing structured data in databases. It supports SQL for querying and manipulating data.
You can use MySQL Workbench, a visual tool, or command-line interface (CLI) to interact with MySQL databases, create tables, run queries, and perform administrative tasks.

Eclipse:

Eclipse is a powerful integrated development environment (IDE) primarily used for Java development, but it supports various programming languages through plugins.
You can create Java projects, web applications, and other software projects in Eclipse.
Eclipse provides features such as code editing, debugging, refactoring, version control integration (like Git), and more.

Java database connectivity (JDBC):

Java Database Connectivity (JDBC) serves as a vital component in Java programming, facilitating seamless interaction between Java applications and relational databases. JDBC enables developers to perform database operations such as querying, updating, and managing data within Java applications

MySQL Connector:

The MySQL Connector/J is the official JDBC driver for MySQL databases provided by Oracle Corporation. It allows Java applications to connect and interact with MySQL databases using the JDBC API.

Demonstration of Project:

The Car Commerce Center project is a Java-based application designed to manage car listings and facilitate interactions between car sellers and buyers. The application allows users to perform various tasks such as adding new cars to the inventory, deleting existing listings, searching for cars based on specific criteria, and viewing all available cars. It employs MySQL as the backend database management system to store and manage car-related data.

Features:

User Authentication: Users can log in to the system using their email and password credentials. The application verifies user credentials against the database records stored in the MySQL database.

Car Management:

- Add Car: Authenticated users can add new cars to the inventory by providing details such as model, company, make year, car type, fuel type, transmission, RC number, insurance, engine power, and color.
- Delete Car: Users can remove cars from the inventory by specifying the unique CarID associated with each listing. The system ensures that only authorized users can delete their own car listings.

Search Cars:

Users can search for cars based on specific keywords such as model, company, or color. The application performs a database query to retrieve matching car listings and displays them to the user.

Show All Cars:

Users can view a comprehensive list of all available cars in the inventory. The application fetches data from the MySQL database and presents it in a tabular format, including details such as CarID, model, company, make year, color, and user email.

Logout:

Users can log out of the system, terminating their session and returning to the login screen.

Screenshots:

```
Welcome to Car Commerce Center!
Enter your email: user1@gmail.com
Enter your password: password1
Login successful!
```

After login:

```
Select an option:
1. Add Car
2. Delete Car
3. Search Cars
4. Show All Cars
5. Logout
```

Lets press 4

```
List of all cars available:
-----------------------------------------------------------
CarID | Model              | Company          | MakeYear   |
-----------------------------------------------------------
1     | Maruti Swift       | Maruti Suzuki    | 2019       |
2     | Hyundai i20        | Hyundai          | 2019       |
3     | Honda City         | Honda            | 2021       |
4     | Mahindra XUV500    | Mahindra         | 2018       |
5     | Toyota Innova Crysta | Toyota           | 2020
11    | Maruti Swift       | Maruti Suzuki    | 2021       |
-----------------------------------------------------------
```

Lets press 5

```
Select an option:
1. Add Car
2. Delete Car
3. Search Cars
4. Show All Cars
5. Logout
5
धन्यवाद
```

VII. Self -Learning Beyond Classroom

Database Management with MySQL:

Individuals working on this project would likely gain practical experience in setting up and managing a MySQL database. They would learn about creating tables, defining relationships between tables, writing SQL queries to retrieve and manipulate data, and understanding database normalization principles.

Java JDBC Programming:

The project involves using Java Database Connectivity (JDBC) to interact with the MySQL database from a Java application. Individuals would learn how to establish database connections, create prepared statements, execute queries, handle result sets, and manage transactions programmatically.

Project Management and Collaboration:

Working on a project like the Car Commerce Center may involve collaboration with team members, task allocation, and adhering to project timelines and milestones. Individuals would learn about effective project management practices and teamwork.

VIII. Learning from the Project

Hands-on Application of Database Concepts:

We gained practical experience in database design, normalization, and querying by creating and managing a MySQL database for storing car information. This project allowed us to translate conceptual data models into a functioning database schema.

Deepened Understanding of Java Programming:

Implementing the Car Commerce Center in Java involved applying object-oriented programming principles, JDBC for database connectivity. We strengthened our Java skills through real-world application.

Problem-Solving and Troubleshooting Skills:
Throughout the project, we encountered various challenges such as debugging code, handling exceptions, and resolving database-related issues. By overcoming these obstacles, we developed resilience, creativity, and analytical thinking.

Project Management and Time Management:

Coordinating tasks, managing deadlines, and collaborating with team members (if applicable) required effective project management skills. We learned to prioritize tasks, allocate resources efficiently, and adapt to changing project requirements.

Personal Growth and Confidence:

Successfully completing a project like the Car Commerce Center instilled a sense of accomplishment and confidence in our abilities. It encouraged us to take on more complex challenges and pursue further learning and development opportunities.

IX. Challenges Faced

Normalization:

Ensuring that database tables are properly normalized according to normalization forms like First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF). Understanding these forms and applying them correctly was a challenge.

Integration with JDBC:

Integrating JDBC (Java Database Connectivity) to establish a connection between the Java application and the MySQL database could have posed challenges, especially for those less familiar with JDBC APIs and SQL syntax. Handling exceptions, managing connections, and executing queries securely and efficiently required additional effort.

X. Conclusion

The key takeaways from this project are manifold. Firstly, we gained a deeper understanding of database management systems, particularly MySQL, and how to interact with them using Java Database Connectivity (JDBC). This project enhanced our knowledge of SQL queries, database design principles, and normalization techniques. Additionally, we learned about user authentication and authorization processes, implementing them securely within the application. The project also honed our skills in handling user inputs, error handling, and implementing a menu-driven interface for user interaction. Moreover, it provided insights into the importance of data integrity, security measures, and best practices for developing robust, scalable software systems. Overall, this project served as a comprehensive learning experience,

integrating theoretical concepts with practical application, and equipping us with valuable skills for future endeavors in software development and database management.
