A. Title Page

Lewis University
CPSC 50900: Database Systems
Spring 2025 Term Project

Car Sales and Customer Management Database

The Car Sales and Customer Management Database is designed to help car dealerships efficiently manage their sales, customer interactions, and inventory.

Keshetty Sreeja,sreejakeshetty@lewisu.edu

Work products stored in the Github repository https://github.com/s-keshetty/CarDealershipData

Table of Contents

Schedule of Milestones

Here is a schedule that shows when each milestone is due and what sections comprise it.

Deadline	Sections for which you must demonstrate significant progress
February 4 at 11:59pm	a. Title page b. Initial proposal c. Data sources d. Alternative ways to store the data r. Activity Log – at least six entries covering the first two weeks
February 18 at 11:59pm	e. Conceptual and logical models f. Physical model g. Populate the database with data r. Activity Log – at least six entries covering the past two weeks
March 4 at 11:59pm	h. Data manipulation language (DML) scripts i. Indexes j. Views l. Transactions m. Security r. Activity Log – at least six entries covering the past two weeks

The remaining sections – Triggers, Locking and Concurrency, Backup, and Programming, will be turned in with the final report, which is due March 16 at 11:59pm.

B. Initial Proposal

Description: You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?

Rubric: Your response to each of these six questions will be graded out of 3 points.

- *3 points: clear, complete descriptions that convey the importance and meaning of your data*
- *2 points: mostly clear descriptions, although some additional data would have helped in some sections*
- *1 point: necessary details are lacking in many of your responses.*

You will also earn 2 additional points for coming up with a descriptive title for your project.

As you consider various ideas for your project, keep in mind that your database is going to have to store data for at least 8 different types of things. Each of these different "types of things" will become a table in the database you design and build. So, the idea can't be so narrow that you can't identify at least eight different types of things in it that you'd store data about.

Total points possible: 20

This project involves creating a relational database for a car dealership that stores information about cars,customers,orders,salepersons,features,fuel types,engine types and car feature mapping.The database has structured data and includes the following details:

- **Customers:** Customer ID, name, phone number
- **Orders:** Order ID, customer ID, car ID, order date
- **Salespersons:** Salesperson ID, name, contact info, number of sales
- **Features:** Feature ID, feature name (e.g., GPS, sunroof, parking sensors)
- **Car Features:** Mapping table linking cars and features
- **Fuel Types:** Fuel ID, fuel type (gasoline, diesel, electric, etc.)
- **Engine Types:** Engine ID, engine type (V6, V8, electric, etc.)
- **Cars:** Car ID, make, fuel type, engine type, price

I am interested in this database project because it involves the automobile industry which is highly data driven and dealerships rely on efficient inventory management,ent systems that have customer tracking and monitoring sales.

By organizing this data into a structured database,dealerships can improve customer experience and optimize the inventory by making the decision-making based on sales and demand forecasting much more efficient.

This data is important because it helps car dealerships with:

- Improving customer service by tracking customer interactions
- Determining the midels that are selling well
- Understand which car features are in high demand in the current market.
- Improve the delearship efficiency by managing salesperson performance and customer history.

And as we know that data is highly valuable hence proper data management allows dealerships to make good business decisions and enhance customer satisfaction.
The data is collected from the following:

- public datasets available online(eg.cars.csv)
- fictional datasets(eg.orders.xml)
- manual data entry for specific dealership records(salespersons and customer details)

The data has been formatted into csv,xml and json.
The primary users of this would be:
1.Car dealership managers-to track inventory and customer preferences.

2.Salepersons-to access customer history regarding payments or to optimize sale strategies.

3.Automotive analysts-to analyze current market trends and demand patterns.

This database can be integrated into a dealership management system which:

Tracks salesperson performance based on the number of orders completed,automates the oreder processing for customers and allows inventory forecasting to improve the dealership efficiency.
This application can also be served as an internal customer relationship management tool.

C. Data Sources

Description: Gather your data in text files. The text files may be csv, tab-delimited, xml, json, or some other custom format. Not all the files need be of the same type. Identify what each file contains by indicating where it came from, explaining in detail how it is structured, and describing how you will

reorganize the data into a relational database. Post your data files to your GitHub repository, and provide samples of the data in your Word doc.

Rubric: Your work will be graded as follows:

- *5 points: you gathered multiple data files that contain the data that will populate your databases. If you do not use multiple data files, you will not receive credit.*
- *5 points: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.*
- *3 points: you identify which fields you plan to include in your database, including their data types and any constraints you expect to impose on the data or steps you'll have to take to clean up the data.*
- *2 points: you post the data files to your GitHub account and make it possible for me to see them.*

Total points possible: 15

To create the relational database,I have collected data and saved in multiple formats:

File Name	Description
cars.csv	Stores car details (car_id, make, fuel_id, engine_id, price).
customers.json	Stores customer details (customer_id, name, phone).
orders.xml	Stores orders (order_id, customer_id, car_id, order date).
salespersons.csv	Stores salesperson details (salesperson_id, name, contact info, num_sales).
features.csv	Stores feature details (feature_id, feature_name).
car_features.csv	Mapping table linking cars & features (car_id, feature_id).
fuel_types.json	Stores fuel type details (fuel_id, fuel_type).
engine_types.xml	Stores engine type details (engine_id, engine_type).

Each of these files represents an entity in the database.

The data sources and structure:
cars.csv-(table format)

Source: github(EDA-on-Automobile-Dataset/Automobile_data.csv at master · rushabh-mehta/EDA-on-Automobile-Dataset)

Contains: car details like ID,fuel type,engine type,make and price.

Structure: this file has one row per car.

Eg:

car_id,make,fuel_id,engine_id,price

1,Toyota,1,1,25000

2,Ford,1,2,28000

Customers.json

Source:Fictional data

Eg:

```
[
  {"customer_id": 1, "name": "John Doe", "phone": "123-456-7890"},
  {"customer_id": 2, "name": "Jane Smith", "phone": "987-654-3210"}
]
```

Orders.xml

Source: simulated orders for car purchases

Structure: represents hierarchical relationships

Eg:

```
<orders>
  <order>
    <order_id>1</order_id>
    <customer_id>1</customer_id>
    <car_id>3</car_id>
    <order_date>2024-01-01</order_date>
  </order>
</orders>
```

Salepersons.csv

Source: manually enterd records

Structure: stores salepersons detials like contact info and performance matrics.

Eg:
salesperson_id,name,contact_info,num_sales

1,Mike Johnson,555-555-5555,20

2,Anna Davis,444-444-4444,30

Features.csv

Source: this dataset is manually created

Structure: stores feature names

Eg:

feature_id,feature_name

1,GPS

2,Leather Seats

Car_features.csv(bridge table for cars and features)

Source:derived from cars.csv and features.csv forming many-to-many relationship

Structure:Links car_id with featire_id allowing multiple features per car.

Eg:

car_feature_id,car_id,feature_id

1,1,1

2,1,2

3,2,3

Engine_types.xml

Source:manually generated datset

Structure:stored engine types in hierarchial structure.

Eg:

<engine_types>

  <engine_type>

<engine_id>1</engine_id>

<engine_name>DOHC</engine_name>

</engine_type>

</engine_types>

Each file contributes fields that define the structure or relational database.

Below are a few entities and their constraints:
Customers:

Customer_id→(PK)

Name→not null

Phone→unique,not null

Orders:

Order_id→(PK)

Customer_id→(FK)(customers)

Car_id→(FK)cars

Order_date→not null

Car features:

Car_id→(FK)cars

Feature_id→(FK)features

So, every table in the datbase must have primary key to ensure the integrity of data.Foreign keys are for establishing relationships between tables.

Data cleaning :
Primary keys fields remove the dupilcate entries.

If order dates are inconsistent or incorrect the dealership's sales records will be unreliable. Hence, data type constraints are applied.

D. Alternative Ways to Store the Data

Description: We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data

using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.

Rubric: Your work will be graded as follows

- *5 points for clearly describing how your data could be stored using one alternative to relational databases and what the advantages and disadvantages of that approach would be.*
- *5 points for clearly describing how your data could be stored using another alternative to relational databases and what the advantages and disadvantages of that approach would be.*

Total points possible: 10

A hierarchical database organizes data in a tree-like structure with parent-child relationships. The data is arranged in levels, similar to a file system where each record has a single parent but can have multiple children.

Structure:
Root node: car dealership

Branch_1=customers

- Customer ID → Name, Phone
- Orders (Child Nodes) → Order ID, Car ID, Order Date

Branch_2=cars

- Car ID → Make, Price, Fuel Type, Engine Type
- Features (Child Nodes) → Feature ID, Feature Name

Every customer has their orders stored as child nodes and each car has its features stored as child nodes.

Implementation(XML format):

```
<dealership>
  <customers>
    <customer>
      <customer_id>1</customer_id>
      <name>John Doe</name>
      <phone>123-456-7890</phone>
```

```
    <orders>
      <order>
        <order_id>1</order_id>
        <car_id>3</car_id>
        <order_date>2024-01-01</order_date>
      </order>
    </orders>
  </customer>
 </customers>
 <cars>
  <car>
    <car_id>3</car_id>
    <make>Honda</make>
    <fuel_type>Gasoline</fuel_type>
    <engine_type>V6</engine_type>
    <price>25000</price>
    <features>
      <feature>
        <feature_id>1</feature_id>
        <feature_name>GPS</feature_name>
      </feature>
    </features>
  </car>
 </cars>
</dealership>
```

Advantages:

Quick data retieval-as relationships are predefined,queries are optimized for quick lookups.

Data integrity-Parent-child relationships make sure that data integrity is strong.

Disadvantages:
Limited flexibility- the one-to-many relationship structure makes it difficult to handle complex relationships(many-to-many)

Difficult to modify-Changing the structure like adding a new level may require root reconstruction.

NoSQL:

A document-based NoSQL database stores data in JSON-like format, eliminating the need for rigid table structures found in relational databases. Each document contains all relevant data in a nested format, making data retrieval faster and more efficient.

Orders collection-every order is a document


```
{
  "order_id": 1,
  "customer": {
    "customer_id": 1,
    "name": "John Doe",
    "phone": "123-456-7890"
  },
  "car": {
    "car_id": 3,
    "make": "Honda",
    "fuel_type": "Gasoline",
    "engine_type": "V6",
    "price": 25000
  },
```

"order_date": "2024-01-01"

}

Cars collection-every car is a document

{

"car_id": 3,

"make": "Honda",

"fuel_type": "Gasoline",

"engine_type": "V6",

"price": 25000,

"features": ["GPS", "Sunroof", "Backup Camera"]

}

Advantages:

Scalability- NoSQL datbases are horizontally scalable so they can be distributed across multiple servers.

High performance for reads- as the data is stored in single document, retrieval needs fewer queries.

Disadvantages:

Less support for transactions- making sure that data consistency across multiple documents is a tough.

Data redundancy-customer or car data may be duplicated in many documents.

E. Conceptual and Logical Models

Description: First, come up with a conceptual model. The conceptual model identifies the entity sets and the relationships among them. For each relationship, identify the connectivity and the participation (optional or mandatory).

Now that you know the entity sets, the next step is to develop the logical model by adding attributes. For each entity set, identify the attributes that describe the entity set. This may include references to other entity sets that are involved in relationships. Then, identify the functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine, like this:

determinant, or, determinants → attributes, they, determine

This becomes the basis for identifying your entity sets, which will become your tables when we move to the physical model in the next section. The attributes listed on the left of the arrows are candidates to become your primary key attributes. Attributes that are references to other entity sets are candidates to become the foreign keys.

For entity sets that have multi-attribute determinants, replace them with surrogate keys. This makes it easier to identify each entity in the set and to define foreign keys.

Then apply normalization to make sure that your design satisfies First, Second, and Third Normal forms. For 1st Normal Form, make sure that all attributes are indivisible. This may require adding an entity set that lists values that appear in comma-separated lists as individual entities. For 2nd Normal Form, make sure there are no partial dependencies (this won't be a problem if all your entity sets have single-attribute determinants). Finally, make sure all your entity sets are in 3rd Normal Form. This means that you have to split transitive dependencies into separate entity sets and add relationships between the original entity set and the new ones.

Finally, draw the logical model as an ERD. At this point, your design will have entity sets, their relationships, and their attributes. M:N relationships are acceptable at this point, as we'll remove them in the physical model.

Rubric: Your work will be graded as follows:

- *5 points for identifying all entity sets*
- *5 points for writing each relationship between entity sets as two sentences and correctly identifying their connectivity and participation.*
- *5 points for adding attributes to entity sets and writing the functional dependencies correctly. Replace multi-attribute determinants with surrogate keys.*
- *4 points for performing the normalization steps. Make sure your design is in 3rd Normal Form.*
- *5 points for drawing the ERD for the logical model. At this point, the ERD will show entity sets, relationships, attributes, and primary identifiers. The design may include M:N relationships at this point. We'll get rid of those in the physical model.*

Total points possible: 24

The conceptual model focuses on identifying entity sets and their relationships, without going into attributes or table structures. Below, is the definition of the main entities, their relationships, and specify connectivity (1:1, 1:M, M:N) and participation (mandatory or optional).

These are the 8 main entities in the Car Sales and Customer Management Database:

Customers – People who purchase cars.
Orders – Tracks sales transactions.
Cars – Vehicles available for sale.
Salespersons – Employees who assist in selling cars.

Features – Various features available in cars (e.g., GPS, Sunroof).
Car Features – A bridge table linking cars and features (many-to-many relationship).
Fuel Types – Types of fuel used by cars (Gasoline, Diesel, Electric, etc.).
Engine Types – Types of engines used by cars (V6, Electric, etc.).

Each relationship is described, as required:

1 Customers & Orders (1:M, Mandatory)

- A customer can place multiple orders over time.
- Each order must be linked to exactly one customer.

2 Orders & Cars (M:1, Mandatory)

- Each order must contain a car that the customer is purchasing.
- A car can appear in multiple orders if it is sold multiple times.

3 Salespersons & Orders (1:M, Optional)

- A salesperson can handle multiple orders and assist customers.
- Some orders may not be linked to a salesperson (e.g., online purchases).

4 Cars & Features (M:N, Optional, Handled by Car_Features Table)

- A car can have multiple features, such as GPS, Sunroof, or Parking Sensors.
- Each feature can belong to multiple cars (e.g., "GPS" exists in both Toyota and BMW models).

5 Cars & Fuel Types (M:1, Mandatory)

- A car must have exactly one fuel type (e.g., Gasoline, Diesel, Electric).
- A fuel type can be used by multiple cars (e.g., Gasoline cars from Toyota, Ford, etc.).

6 Cars & Engine Types (M:1, Mandatory)

- A car must have exactly one engine type (e.g., V6, Electric).
- An engine type can be used by multiple cars (e.g., "V6" engines are found in Ford and Audi models).

Entities and Attributes

1. Customers
    - PK: customer_id
    - Attributes: name (not null), phone number (unique)
2. Orders
    - PK: order_id
    - FK: customer_id (references Customers), car_id (references Cars), salesperson_id (optional, references Salespersons)
    - Attributes: order_date (required)
3. Cars
    - PK: car_id
    - FK: fuel_id (references Fuel Types), engine_id (references Engine Types)
    - Attributes: make (brand), price (greater than zero)
4. Salespersons
    - PK: salesperson_id
    - Attributes: name, contactinfo, numsales
5. Features
    - PK: feature_id
    - Attributes: feature_name
6. Car Features
    - PK: carfeature_id
    - FK: car_id (references Cars), feature_id (references Features)
7. Fuel Types
    - PK: fuel_id
    - Attributes: fuel_type
8. Engine Types
    - PK: engine_id

- Attributes: engine_type

A functional dependency shows how one attribute determines another.

1. customer_id → name, phone

- The customer_id uniquely determines the name and phone of a customer.

2. order_id → customer_id, car_id, salesperson_id, order_date

- The order_id uniquely determines which customer_id placed the order, which car_id was purchased, which salesperson_id handled the sale (if any), and the order_date.

3. car_id → make, fuel_id, engine_id, price

- The car_id uniquely determines the car's make, fuel_id, engine_id, and price.

4. salesperson_id → name, contact_info, num_sales

- The salesperson_id uniquely determines the salesperson's name, contact_info, and their num_sales count.

5. feature_id → feature_name

- The feature_id uniquely determines the feature_name.

6. car_feature_id → car_id, feature_id

- The car_feature_id uniquely determines which car_id has which feature_id.

7. fuel_id → fuel_type

- The fuel_id uniquely determines the fuel_type.

8. engine_id → engine_type

- The engine_id uniquely determines the engine_type.

Normalistaion: the process of organising data in relational database to remove redundancy.

Normalisation process:

(1NF)Ensuring atomicity:

Cars table previously had fuel_type and engine_type as text, leading to redundancy.

Solution:

- Replaced the text values with foreign keys(IDs)
- Duplicate values in the cars table linked to already existing tables(fuel_type and engine_type).

Final normalised tables which are 3NF compliant.:

Cars table:

A	B	C	D	E
make	fuel-type	engine-typ	price	car_id
alfa-romer	1	1	13495	1
alfa-romer	1	1	16500	2
alfa-romer	2	2	16500	3

As fuel_type depends on fuel_id not car_id and engine_type depends on engine_id not car_id. That is why fuel and engine types have separate tables.

Fuel_types table:

fuel_id| fuel_type

---

1 | Gasoline

2 | Diesel

Engine types table:

<engine_types>

<engine_type>

<engine_id>1</engine_id>

<engine_name>DOHC</engine_name>

</engine_type>

<engine_type>

<engine_id>2</engine_id>

<engine_name>OHV</engine_name>

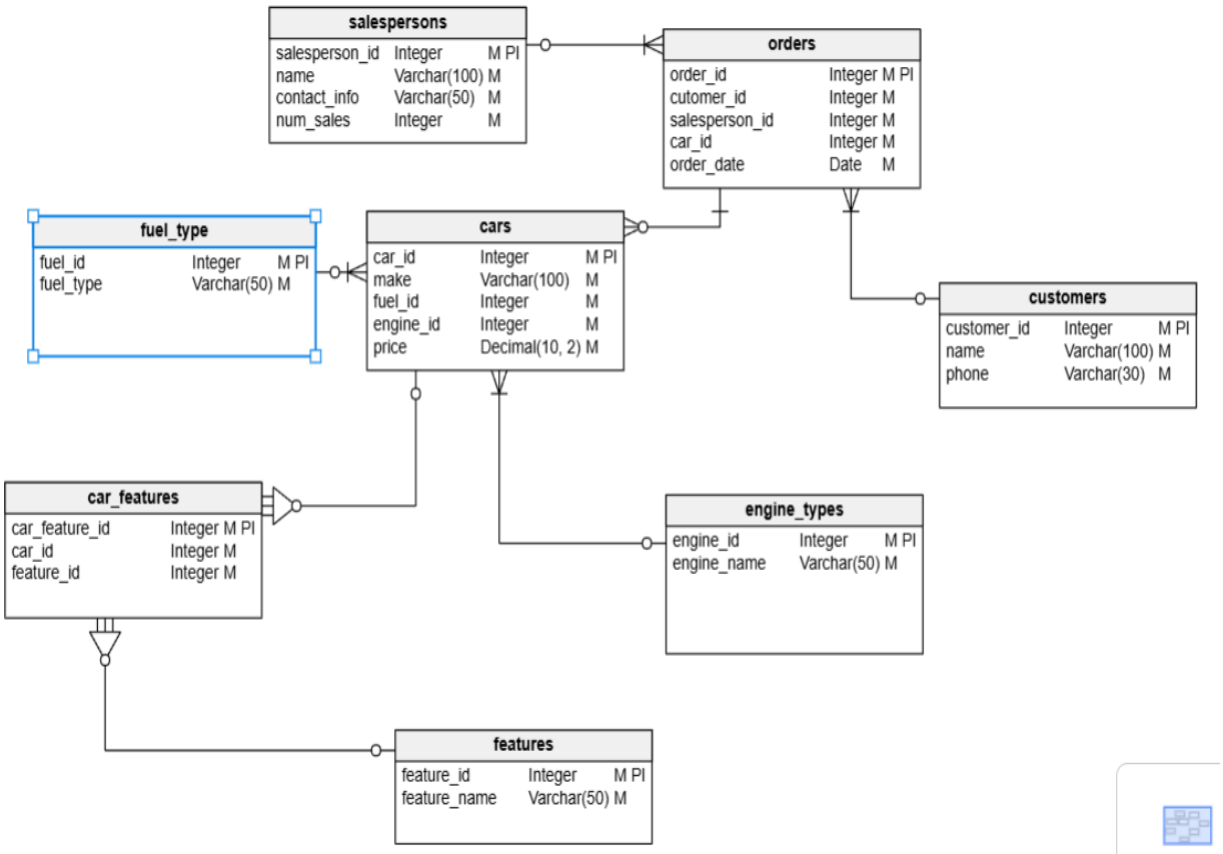
</engine_type>

</engine_types>

Most importantly car_features table is created that acts as a bridge table, as a car can have many features and a feature can be in many cars.

Car_features table:

A	B	C
car_featur	car_id	feature_id
1	1	1
2	2	3



Screenshot of the logical model of car dealership database(bridge entity (car_features)has been included in logical model it self.)

F. Physical Model

Description: This is where you will complete your database design. Add data types, including size constraints, uniqueness constraints, and auto-incrementing for all attributes. Implement relationships using foreign keys. Replace many-to-many relationships with two one-to-many relationships using bridge entity sets. Add additional entity sets that you think could be helpful for storing the acceptable values of particular attributes. (For example, if you were storing student data, valid student statuses might include Good Standing, Graduated, On Probation, Expelled. Put those in a table and create a relationship back to the student table). Draw the ERD for the physical model.

Using the final ERD, write the SQL DDL statements needed to create the database, its tables, and the relationships among them. Run these statements in MySQL to build your database. Provide screen shots that show the database you built in MySQL, including its tables and descriptions of some of the tables. To show a list of databases and a list of the tables in a particular database, use the show command. To see a description for a table, use the describe command.

Rubric: Your work will be graded as follows:

- *3 points for introducing bridge entity sets (if necessary)*
- *3 points for adding data types and other constraints on the data.*
- *3 points for introducing other entity sets and their relationships that help enforce what values can be assigned to particular attributes (if necessary)*
- *5 points for drawing the ERD for the physical model. If you used Vertabelo, the resulting ERD must be free of errors and warnings*
- *6 points for generating the SQL scripts that build the database and then running the script in mysql. Demonstrate that the script built the database and its tables with screenshots that show that you ran the show and describe commands.*

You will be penalized 4 points if your database doesn't have at least 8 appropriately defined tables.

Total points possible: 20

1.Relationship Between Customers and Orders (1:M)

- A customer can place multiple orders.
- Each order is placed by one customer.

Cardinality: One-to-Many (1:M)

- One Customer → Many Orders

2.Relationship Between Orders and Cars (1:M)

- One Car can be linked to many Orders (the same car can be ordered multiple times).
- Each Order is linked to only one Car,a car can be part of multiple orders (if resold).

Cardinality: One-to-many (1:M)

3. Relationship Between Salespersons and Orders (1:M)

- A salesperson handles multiple orders.
- Each order is handled by one salesperson.

Cardinality: One-to-Many (1:M)

- One Salesperson → Many Orders

4.Relationship Between Cars and Fuel Types (1:M)

- A fuel type (e.g., Gasoline, Diesel) is used in multiple cars.
- Each car has one fuel type.

Cardinality: One-to-Many (1:M)

- One Fuel Type → Many Cars

5.Relationship Between Cars and Engine Types (1:M)

- An engine type (e.g., V6, DOHC) is used in multiple cars.
- Each car has one engine type.

Cardinality: One-to-Many (1:M)

- One Engine Type → Many Cars

6.Relationship Between Cars and Features (M:N)

- A car can have multiple features (GPS, Leather Seats, etc.).
- A feature can belong to multiple cars.

Cardinality: Many-to-Many (M:N)

- Many Cars → Many Features
- Solution: Use a Bridge Table (Car_Features)

The physical model represents the final structure of the Car Dealership Database, incorporating data types, constraints, foreign keys, and bridge tables to maintain data integrity. The model includes eight main tables along with foreign key relationships that enforce business rules.

1.Data Types & Constraints

- **Primary Keys (PK)** are defined using AUTO_INCREMENT where applicable.
- **Foreign Keys (FK)** establish relationships between tables.
- **VARCHAR types** are used for text-based attributes like make, name, and contact_info, with appropriate length constraints.
- **DECIMAL(10,2)** is used for the price attribute to store currency values.

2. Bridge Table for Many-to-Many Relationship

- Car Features (car_features)
    - A car can have multiple features, and a feature can belong to multiple cars.

- This is resolved by a many-to-many (M:N) relationship using the car_features table.

Steps to Implement in MySQL

1. Create the database and use it:

```
CREATE DATABASE car_dealer_db;
USE car_dealer_db;
```

2. Run the SQL script using:

```
source C:/path/to/car.sql;
```

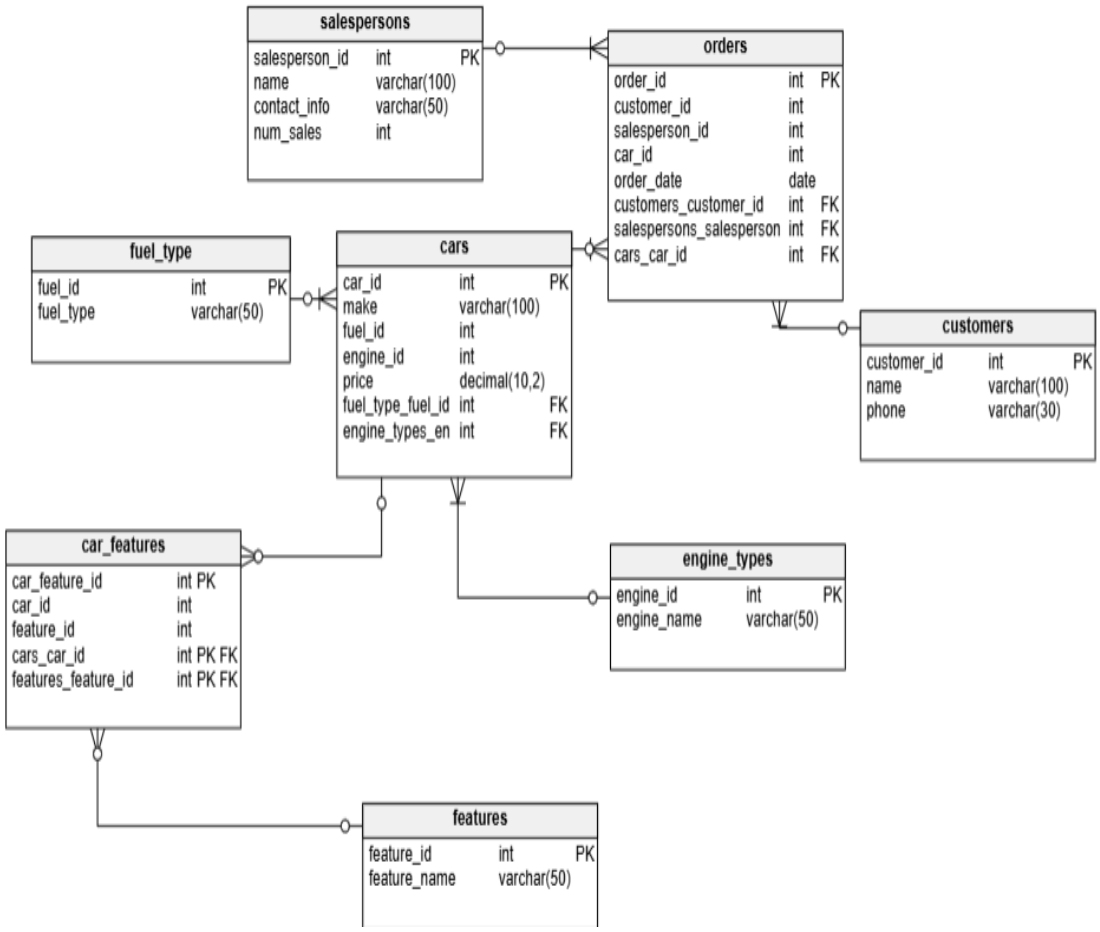
Verify that the tables exist:

```
SHOW TABLES;
```

Check table structures:

```
DESCRIBE cars;
DESCRIBE orders;
DESCRIBE customers;
```

Physical model



```
MariaDB [(none)]> USE carDB;
Database changed
MariaDB [carDB]> CREATE TABLE customers (
    ->     customer_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100) NOT NULL,
    ->     phone VARCHAR(30) NOT NULL
    -> );
Query OK, 0 rows affected (0.007 sec)

MariaDB [carDB]>
MariaDB [carDB]> CREATE TABLE salespersons (
    ->     salesperson_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100) NOT NULL,
    ->     contact_info VARCHAR(50) NOT NULL,
    ->     num_sales INT NOT NULL DEFAULT 0
    -> );
Query OK, 0 rows affected (0.007 sec)

MariaDB [carDB]>
MariaDB [carDB]> CREATE TABLE cars (
    ->     car_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     make VARCHAR(100) NOT NULL,
    ->     fuel_id INT NOT NULL,
    ->     engine_id INT NOT NULL,
    ->     price DECIMAL(10,2) NOT NULL
    -> );
Query OK, 0 rows affected (0.007 sec)

MariaDB [carDB]> CREATE TABLE orders (
    ->     order_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_id INT NOT NULL,
    ->     salesperson_id INT NOT NULL,
    ->     car_id INT NOT NULL,
    ->     order_date DATE NOT NULL,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE,
    ->     FOREIGN KEY (salesperson_id) REFERENCES salespersons(salesperson_id) ON DELETE CASCADE,
    ->     FOREIGN KEY (car_id) REFERENCES cars(car_id) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.027 sec)

MariaDB [carDB]> SHOW TABLES;
+-----------------+
| Tables_in_cardb |
+-----------------+
| cars            |
| customers       |
| orders          |
| salespersons    |
+-----------------+
4 rows in set (0.000 sec)
```

```
MariaDB [carDB]> CREATE TABLE fuel_types (
    ->     fuel_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     fuel_type VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.008 sec)

MariaDB [carDB]> CREATE TABLE engine_types (
    ->     engine_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     engine_name VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.007 sec)

MariaDB [carDB]> CREATE TABLE features (
    ->     feature_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     feature_name VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.015 sec)

MariaDB [carDB]> CREATE TABLE car_features (
    ->     car_feature_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     car_id INT NOT NULL,
    ->     feature_id INT NOT NULL,
    ->     FOREIGN KEY (car_id) REFERENCES cars(car_id),
    ->     FOREIGN KEY (feature_id) REFERENCES features(feature_id)
    -> );
Query OK, 0 rows affected (0.029 sec)

MariaDB [carDB]> SHOW TABLES;
+-----------------+
| Tables_in_cardb |
+-----------------+
| car_features    |
| cars            |
| customers       |
| engine_types    |
| features        |
| fuel_types      |
| orders          |
| salespersons    |
+-----------------+
8 rows in set (0.001 sec)

MariaDB [carDB]> INSERT INTO fuel_types (fuel_id, fuel_type) VALUES
    -> (1, 'Gasoline'),
    -> (2, 'Diesel'),
    -> (3, 'Electric'),
    -> (4, 'Hybrid'),
    -> (5, 'Ethanol'),
    -> (6, 'Natural Gas'),
    -> (7, 'Hydrogen'),
```

```
    -> (8, 'Propane');
Query OK, 8 rows affected (0.004 sec)
Records: 8  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO engine_types (engine_id, engine_name) VALUES
    -> (1, 'DOHC'),
    -> (2, 'OHV'),
    -> (3, 'SOHC'),
    -> (4, 'Inline-4'),
    -> (5, 'V6'),
    -> (6, 'V8');
Query OK, 6 rows affected (0.003 sec)
Records: 6  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO cars (car_id, make, fuel_id, engine_id, price) VALUES
    -> (1, 'alfa-romero', 1, 1, 13495),
    -> (2, 'alfa-romero', 1, 1, 16500),
    -> (3, 'alfa-romero', 2, 2, 16500),
    -> (4, 'audi', 1, 2, 13950),
    -> (5, 'audi', 1, 2, 17450),
    -> (6, 'audi', 1, 3, 15250),
    -> (7, 'audi', 5, 2, 17710),
    -> (8, 'audi', 3, 4, 18920),
    -> (9, 'audi', 4, 2, 23875),
    -> (10, 'audi', 1, 2, 16000),
    -> (11, 'bmw', 1, 2, 16430),
    -> (12, 'bmw', 1, 2, 16925),
    -> (13, 'bmw', 8, 6, 20970),
    -> (14, 'bmw', 7, 3, 21105),
    -> (15, 'bmw', 1, 3, 24565),
    -> (16, 'bmw', 1, 5, 30760),
    -> (17, 'bmw', 1, 2, 41315),
    -> (18, 'bmw', 1, 2, 36880),
    -> (19, 'chevrolet', 1, 5, 5151),
    -> (20, 'chevrolet', 5, 3, 6295),
    -> (21, 'chevrolet', 1, 3, 6575),
    -> (22, 'dodge', 1, 3, 5572),
    -> (23, 'dodge', 1, 3, 6377),
    -> (24, 'dodge', 2, 3, 7957),
    -> (25, 'dodge', 1, 6, 6229),
    -> (26, 'dodge', 1, 3, 6692),
    -> (27, 'dodge', 1, 3, 7609),
    -> (28, 'dodge', 5, 1, 8558),
    -> (29, 'dodge', 1, 3, 8921),
    -> (30, 'dodge', 4, 3, 12964);
Query OK, 30 rows affected (0.003 sec)
Records: 30  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO customers (customer_id, name, phone) VALUES
    -> (1, 'John Doe', '123-456-7890'),
```

```
    -> (8, 'Propane');
Query OK, 8 rows affected (0.004 sec)
Records: 8  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO engine_types (engine_id, engine_name) VALUES
    -> (1, 'DOHC'),
    -> (2, 'OHV'),
    -> (3, 'SOHC'),
    -> (4, 'Inline-4'),
    -> (5, 'V6'),
    -> (6, 'V8');
Query OK, 6 rows affected (0.003 sec)
Records: 6  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO cars (car_id, make, fuel_id, engine_id, price) VALUES
    -> (1, 'alfa-romero', 1, 1, 13495),
    -> (2, 'alfa-romero', 1, 1, 16500),
    -> (3, 'alfa-romero', 2, 2, 16500),
    -> (4, 'audi', 1, 2, 13950),
    -> (5, 'audi', 1, 2, 17450),
    -> (6, 'audi', 1, 3, 15250),
    -> (7, 'audi', 5, 2, 17710),
    -> (8, 'audi', 3, 4, 18920),
    -> (9, 'audi', 4, 2, 23875),
    -> (10, 'audi', 1, 2, 16000),
    -> (11, 'bmw', 1, 2, 16430),
    -> (12, 'bmw', 1, 2, 16925),
    -> (13, 'bmw', 8, 6, 20970),
    -> (14, 'bmw', 7, 3, 21105),
    -> (15, 'bmw', 1, 3, 24565),
    -> (16, 'bmw', 1, 5, 30760),
    -> (17, 'bmw', 1, 2, 41315),
    -> (18, 'bmw', 1, 2, 36880),
    -> (19, 'chevrolet', 1, 5, 5151),
    -> (20, 'chevrolet', 5, 3, 6295),
    -> (21, 'chevrolet', 1, 3, 6575),
    -> (22, 'dodge', 1, 3, 5572),
    -> (23, 'dodge', 1, 3, 6377),
    -> (24, 'dodge', 2, 3, 7957),
    -> (25, 'dodge', 1, 6, 6229),
    -> (26, 'dodge', 1, 3, 6692),
    -> (27, 'dodge', 1, 3, 7609),
    -> (28, 'dodge', 5, 1, 8558),
    -> (29, 'dodge', 1, 3, 8921),
    -> (30, 'dodge', 4, 3, 12964);
Query OK, 30 rows affected (0.003 sec)
Records: 30  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO customers (customer_id, name, phone) VALUES
    -> (1, 'John Doe', '123-456-7890'),
```

```
MariaDB [carDB]> INSERT INTO orders (order_id, customer_id, salesperson_id, car_id, or
    -> (1, 1, 3, 2, '2024-02-01'),
    -> (2, 2, 5, 5, '2024-02-05'),
    -> (3, 3, 7, 7, '2024-02-08'),
    -> (4, 4, 1, 10, '2024-02-12'),
    -> (5, 5, 2, 12, '2024-02-15'),
    -> (6, 6, 6, 14, '2024-02-18'),
    -> (7, 7, 4, 18, '2024-02-20'),
    -> (8, 8, 9, 20, '2024-02-22'),
    -> (9, 9, 8, 25, '2024-02-25'),
    -> (10, 10, 10, 30, '2024-02-28');
Query OK, 10 rows affected (0.013 sec)
Records: 10  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO features (feature_id, feature_name) VALUES
    -> (1, 'GPS'),
    -> (2, 'Leather Seats'),
    -> (3, 'Sunroof'),
    -> (4, 'Parking Sensors'),
    -> (5, 'Bluetooth Audio'),
    -> (6, 'Backup Camera'),
    -> (7, 'Keyless Entry'),
    -> (8, 'Adaptive Cruise Control'),
    -> (9, 'Blind Spot Monitoring'),
    -> (10, 'Heated Seats');
Query OK, 10 rows affected (0.003 sec)
Records: 10  Duplicates: 0  Warnings: 0

MariaDB [carDB]> INSERT INTO car_features (car_feature_id, car_id, feature_id) VALUES
    -> (1, 1, 1),
    -> (2, 2, 3),
    -> (3, 3, 5),
    -> (4, 4, 7),
    -> (5, 5, 2),
    -> (6, 6, 4),
    -> (7, 7, 6),
    -> (8, 8, 8);
Query OK, 8 rows affected (0.003 sec)
Records: 8  Duplicates: 0  Warnings: 0

MariaDB [carDB]> SHOW TABLES;
+----------------+
| Tables_in_cardb |
+----------------+
| car_features    |
| cars            |
| customers       |
| engine_types    |
| features        |
| fuel_types      |
```

```
    -> (8, 8, 8);
Query OK, 8 rows affected (0.003 sec)
Records: 8  Duplicates: 0  Warnings: 0

MariaDB [carDB]> SHOW TABLES;
+-----------------+
| Tables_in_cardb |
+-----------------+
| car_features    |
| cars            |
| customers       |
| engine_types    |
| features        |
| fuel_types      |
| orders          |
| salespersons    |
+-----------------+
8 rows in set (0.001 sec)

MariaDB [carDB]> SELECT*FROM car_features;
+----------------+--------+------------+
| car_feature_id | car_id | feature_id |
+----------------+--------+------------+
|              1 |      1 |          1 |
|              2 |      2 |          3 |
|              3 |      3 |          5 |
|              4 |      4 |          7 |
|              5 |      5 |          2 |
|              6 |      6 |          4 |
|              7 |      7 |          6 |
|              8 |      8 |          8 |
+----------------+--------+------------+
8 rows in set (0.000 sec)

MariaDB [carDB]> SELECT*FROM cars;
+--------+-------------+---------+-----------+----------+
| car_id | make        | fuel_id | engine_id | price    |
+--------+-------------+---------+-----------+----------+
|      1 | alfa-romero |       1 |         1 | 13495.00 |
|      2 | alfa-romero |       1 |         1 | 16500.00 |
|      3 | alfa-romero |       2 |         2 | 16500.00 |
|      4 | audi        |       1 |         2 | 13950.00 |
|      5 | audi        |       1 |         2 | 17450.00 |
|      6 | audi        |       1 |         3 | 15250.00 |
|      7 | audi        |       5 |         2 | 17710.00 |
|      8 | audi        |       3 |         4 | 18920.00 |
|      9 | audi        |       4 |         2 | 23875.00 |
|     10 | audi        |       1 |         2 | 16000.00 |
|     11 | bmw         |       1 |         2 | 16430.00 |
|     12 | bmw         |       1 |         2 | 16925.00 |
```

```
|              6 |       6 |           4 |
|              7 |       7 |           6 |
|              8 |       8 |           8 |
+----------------+---------+-------------+
8 rows in set (0.000 sec)

MariaDB [carDB]> SELECT*FROM cars;
+--------+-------------+---------+-----------+----------+
| car_id | make        | fuel_id | engine_id | price    |
+--------+-------------+---------+-----------+----------+
|      1 | alfa-romero |       1 |         1 | 13495.00 |
|      2 | alfa-romero |       1 |         1 | 16500.00 |
|      3 | alfa-romero |       2 |         2 | 16500.00 |
|      4 | audi        |       1 |         2 | 13950.00 |
|      5 | audi        |       1 |         2 | 17450.00 |
|      6 | audi        |       1 |         3 | 15250.00 |
|      7 | audi        |       5 |         2 | 17710.00 |
|      8 | audi        |       3 |         4 | 18920.00 |
|      9 | audi        |       4 |         2 | 23875.00 |
|     10 | audi        |       1 |         2 | 16000.00 |
|     11 | bmw         |       1 |         2 | 16430.00 |
|     12 | bmw         |       1 |         2 | 16925.00 |
|     13 | bmw         |       8 |         6 | 20970.00 |
|     14 | bmw         |       7 |         3 | 21105.00 |
|     15 | bmw         |       1 |         3 | 24565.00 |
|     16 | bmw         |       1 |         5 | 30760.00 |
|     17 | bmw         |       1 |         2 | 41315.00 |
|     18 | bmw         |       1 |         2 | 36880.00 |
|     19 | chevrolet   |       1 |         5 |  5151.00 |
|     20 | chevrolet   |       5 |         3 |  6295.00 |
|     21 | chevrolet   |       1 |         3 |  6575.00 |
|     22 | dodge       |       1 |         3 |  5572.00 |
|     23 | dodge       |       1 |         3 |  6377.00 |
|     24 | dodge       |       2 |         3 |  7957.00 |
|     25 | dodge       |       1 |         6 |  6229.00 |
|     26 | dodge       |       1 |         3 |  6692.00 |
|     27 | dodge       |       1 |         3 |  7609.00 |
|     28 | dodge       |       5 |         1 |  8558.00 |
|     29 | dodge       |       1 |         3 |  8921.00 |
|     30 | dodge       |       4 |         3 | 12964.00 |
+--------+-------------+---------+-----------+----------+
30 rows in set (0.000 sec)

MariaDB [carDB]> SELECT*FROM customers;
+-------------+-----------------+--------------+
| customer_id | name            | phone        |
+-------------+-----------------+--------------+
|           1 | John Doe        | 123-456-7890 |
|           2 | Jane Smith      | 987-654-3210 |
|           3 | Michael Johnson | 555-123-4567 |
```

```
MariaDB [carDB]> SELECT*FROM customers;
+-------------+-----------------+--------------+
| customer_id | name            | phone        |
+-------------+-----------------+--------------+
|           1 | John Doe        | 123-456-7890 |
|           2 | Jane Smith      | 987-654-3210 |
|           3 | Michael Johnson | 555-123-4567 |
|           4 | Emily Davis     | 444-987-6543 |
|           5 | Chris Brown     | 111-222-3333 |
|           6 | Lisa White      | 666-777-8888 |
|           7 | Robert Green    | 999-888-7777 |
|           8 | Sophia Adams    | 333-444-5555 |
|           9 | Daniel Carter   | 222-333-4444 |
|          10 | Olivia Wilson   | 777-666-5555 |
|          11 | Jack Martinez   | 321-654-9870 |
|          12 | Emma Garcia     | 741-852-9630 |
|          13 | Noah Hernandez  | 369-258-1470 |
|          14 | Mia Lopez       | 852-741-3690 |
|          15 | James Wright    | 159-357-9510 |
|          16 | Charlotte Hall  | 456-789-1230 |
|          17 | Benjamin Allen  | 789-456-1230 |
|          18 | Lucas Young     | 951-753-1590 |
|          19 | Harper King     | 357-159-7530 |
|          20 | Elijah Scott    | 258-147-3690 |
|          21 | Aiden Baker     | 654-789-3210 |
|          22 | Madison Adams   | 741-963-8520 |
|          23 | William Nelson  | 369-147-2580 |
|          24 | Evelyn Carter   | 147-258-3690 |
|          25 | Sofia Mitchell  | 357-753-9510 |
|          26 | Henry Perez     | 753-951-3570 |
|          27 | Lily Roberts    | 258-369-1470 |
|          28 | Mason Turner    | 951-357-7530 |
|          29 | Ella Collins    | 123-456-7890 |
|          30 | Samuel Stewart  | 789-123-4560 |
+-------------+-----------------+--------------+
30 rows in set (0.000 sec)

MariaDB [carDB]> SELECT*FROM engine_types;
+-----------+-------------+
| engine_id | engine_name |
+-----------+-------------+
|         1 | DOHC        |
|         2 | OHV         |
|         3 | SOHC        |
|         4 | Inline-4    |
|         5 | V6          |
|         6 | V8          |
+-----------+-------------+
6 rows in set (0.000 sec)
```

```
MariaDB [carDB]> SELECT*FROM fuel_types;
+---------+-------------+
| fuel_id | fuel_type   |
+---------+-------------+
|       1 | Gasoline    |
|       2 | Diesel      |
|       3 | Electric    |
|       4 | Hybrid      |
|       5 | Ethanol     |
|       6 | Natural Gas |
|       7 | Hydrogen    |
|       8 | Propane     |
+---------+-------------+
8 rows in set (0.000 sec)

MariaDB [carDB]> SELECT*FROM orders;
+----------+-------------+----------------+--------+------------+
| order_id | customer_id | salesperson_id | car_id | order_date |
+----------+-------------+----------------+--------+------------+
|        1 |           1 |              3 |      2 | 2024-02-01 |
|        2 |           2 |              5 |      5 | 2024-02-05 |
|        3 |           3 |              7 |      7 | 2024-02-08 |
|        4 |           4 |              1 |     10 | 2024-02-12 |
|        5 |           5 |              2 |     12 | 2024-02-15 |
|        6 |           6 |              6 |     14 | 2024-02-18 |
|        7 |           7 |              4 |     18 | 2024-02-20 |
|        8 |           8 |              9 |     20 | 2024-02-22 |
|        9 |           9 |              8 |     25 | 2024-02-25 |
|       10 |          10 |             10 |     30 | 2024-02-28 |
+----------+-------------+----------------+--------+------------+
10 rows in set (0.000 sec)

MariaDB [carDB]> SELECT*FROM salespersons;
+----------------+-----------------+-------------+-----------+
| salesperson_id | name            | contact_info | num_sales |
+----------------+-----------------+-------------+-----------+
|              1 | Mike Johnson    | 555-555-5555 |        20 |
|              2 | Anna Davis      | 444-444-4444 |        30 |
|              3 | Chris Lee       | 333-222-1111 |        15 |
|              4 | Emma Taylor     | 111-333-5555 |        25 |
|              5 | Liam Wilson     | 222-555-7777 |        18 |
|              6 | Sophia Martinez | 999-888-7777 |        12 |
|              7 | William Clark   | 888-999-1111 |        22 |
|              8 | Olivia Harris   | 777-666-5555 |        28 |
|              9 | Benjamin White  | 666-777-8888 |        16 |
|             10 | Ava King        | 321-654-9870 |        19 |
+----------------+-----------------+-------------+-----------+
10 rows in set (0.000 sec)

MariaDB [carDB]>
```

The SQL script:

```
CREATE TABLE customers (
    ->    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    ->    name VARCHAR(100) NOT NULL,
    ->    phone VARCHAR(30) NOT NULL
CREATE TABLE salespersons (
    ->    salesperson_id INT AUTO_INCREMENT PRIMARY KEY,
    ->    name VARCHAR(100) NOT NULL,
    ->    contact_info VARCHAR(50) NOT NULL,
    ->    num_sales INT NOT NULL DEFAULT 0
    -> );
CREATE TABLE cars (
    ->    car_id INT AUTO_INCREMENT PRIMARY KEY,
    ->    make VARCHAR(100) NOT NULL,
    ->    fuel_id INT NOT NULL,
    ->    engine_id INT NOT NULL,
    ->    price DECIMAL(10,2) NOT NULL
    -> );
CREATE TABLE fuel_types (
    ->    fuel_id INT AUTO_INCREMENT PRIMARY KEY,
    ->    fuel_type VARCHAR(50) NOT NULL
    -> );
```

```
CREATE TABLE orders (
    ->     order_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_id INT NOT NULL,
    ->     salesperson_id INT NOT NULL,
    ->     car_id INT NOT NULL,
    ->     order_date DATE NOT NULL,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON
DELETE CASCADE,
    ->     FOREIGN KEY (salesperson_id) REFERENCES salespersons(salesperson_id)
ON DELETE CASCADE,
    ->     FOREIGN KEY (car_id) REFERENCES cars(car_id) ON DELETE CASCADE
    -> );
CREATE TABLE engine_types (
    ->     engine_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     engine_name VARCHAR(50) NOT NULL
    -> );
CREATE TABLE features (
    ->     feature_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     feature_name VARCHAR(50) NOT NULL
    -> );
CREATE TABLE car_features (
    ->     car_feature_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     car_id INT NOT NULL,
    ->     feature_id INT NOT NULL,
    ->     FOREIGN KEY (car_id) REFERENCES cars(car_id),
    ->     FOREIGN KEY (feature_id) REFERENCES features(feature_id)
```

->);

G. Populate the database with data

Description: You built the database in section F, and it now exists in mysql. Now populate it with your data. Take your original data source or sources and generate insert statements from them. Store the insert statements in a text file, and then use the mysql source command to run these insert statements to populate the various table structures. Generating the necessary insert statements may require writing Python scripts or manipulating Excel databases to convert the data from your original data sources.

Rubric: Your work will be grades as follows:

- *Explain step-by-step and very clearly how you created the required SQL statements from your initial data. Write it as a set of instructions. 5 points*
- *Show the file of insert statements that you ran in MySQL. You may do this either by including the listing in this report or by identifying the file in your GitHub that contains the insert statements. Make sure I have access to your GitHub repository. 4 points*
- *Show screenshots of the data in your MySQL database. To do this, run select statements for each table and show screen shots of what is displayed: 5 points*

Total points possible: 14

First use this sql command:
CREATE DATABASE car_dealer;

USE car_dealer;

To insert we must convert the file contents into sql format like this:
Engine types:

INSERT INTO engine_types (engine_id, engine_name) VALUES

(1, 'DOHC'),

(2, 'OHV'),

(3, 'SOHC'),

(4, 'Inline-4'),

(5, 'V6'),

(6, 'V8');

Customers:

INSERT INTO customers (customer_id, name, phone) VALUES

(1, 'John Doe', '123-456-7890'),

(2, 'Jane Smith', '987-654-3210'),

(3, 'Michael Johnson', '555-123-4567'),

(4, 'Emily Davis', '444-987-6543'),

(5, 'Chris Brown', '111-222-3333'),

(6, 'Lisa White', '666-777-8888'),

(7, 'Robert Green', '999-888-7777'),

(8, 'Sophia Adams', '333-444-5555'),

(9, 'Daniel Carter', '222-333-4444'),

(10, 'Olivia Wilson', '777-666-5555'),

(11, 'Jack Martinez', '321-654-9870'),

(12, 'Emma Garcia', '741-852-9630'),

(13, 'Noah Hernandez', '369-258-1470'),

(14, 'Mia Lopez', '852-741-3690'),

(15, 'James Wright', '159-357-9510'),

(16, 'Charlotte Hall', '456-789-1230'),

(17, 'Benjamin Allen', '789-456-1230'),

(18, 'Lucas Young', '951-753-1590'),

(19, 'Harper King', '357-159-7530'),

(20, 'Elijah Scott', '258-147-3690'),

(21, 'Aiden Baker', '654-789-3210'),

(22, 'Madison Adams', '741-963-8520'),

(23, 'William Nelson', '369-147-2580'),

(24, 'Evelyn Carter', '147-258-3690'),

(25, 'Sofia Mitchell', '357-753-9510'),

(26, 'Henry Perez', '753-951-3570'),

(27, 'Lily Roberts', '258-369-1470'),

(28, 'Mason Turner', '951-357-7530'),

(29, 'Ella Collins', '123-456-7890'),

(30, 'Samuel Stewart', '789-123-4560');

Features:

INSERT INTO features (feature_id, feature_name) VALUES

(1, 'GPS'),

(2, 'Leather Seats'),

(3, 'Sunroof'),

(4, 'Parking Sensors'),

(5, 'Bluetooth Audio'),

(6, 'Backup Camera'),

(7, 'Keyless Entry'),

(8, 'Adaptive Cruise Control'),

(9, 'Blind Spot Monitoring'),

(10, 'Heated Seats');

Salespersons:

INSERT INTO salespersons (salesperson_id, name, contact_info, num_sales) VALUES

(1, 'Mike Johnson', '555-555-5555', 20),

(2, 'Anna Davis', '444-444-4444', 30),

(3, 'Chris Lee', '333-222-1111', 15),

(4, 'Emma Taylor', '111-333-5555', 25),

(5, 'Liam Wilson', '222-555-7777', 18),

(6, 'Sophia Martinez', '999-888-7777', 12),

(7, 'William Clark', '888-999-1111', 22),

(8, 'Olivia Harris', '777-666-5555', 28),

(9, 'Benjamin White', '666-777-8888', 16),

(10, 'Ava King', '321-654-9870', 19);

Fuel types:

INSERT INTO fuel_types (fuel_id, fuel_type) VALUES

(1, 'Gasoline'),

(2, 'Diesel'),

(3, 'Electric'),

(4, 'Hybrid'),

(5, 'Ethanol'),

(6, 'Natural Gas'),

(7, 'Hydrogen'),

(8, 'Propane');

Car features:

INSERT INTO car_features (car_feature_id, car_id, feature_id) VALUES

(1, 1, 1),

(2, 2, 3),

(3, 3, 5),

(4, 4, 7),

(5, 5, 2),

(6, 6, 4),

(7, 7, 6),

(8, 8, 8);

Cars:

INSERT INTO cars (car_id, make, fuel_id, engine_id, price) VALUES

(1, 'alfa-romero', 1, 1, 13495),

(2, 'alfa-romero', 1, 1, 16500),

(3, 'alfa-romero', 2, 2, 16500),

(4, 'audi', 1, 2, 13950),

(5, 'audi', 1, 2, 17450),

(6, 'audi', 1, 3, 15250),

(7, 'audi', 5, 2, 17710),

(8, 'audi', 3, 4, 18920),

(9, 'audi', 4, 2, 23875),

(10, 'audi', 1, 2, 16000),

(11, 'bmw', 1, 2, 16430),

(12, 'bmw', 1, 2, 16925),

(13, 'bmw', 8, 6, 20970),

(14, 'bmw', 7, 3, 21105),

(15, 'bmw', 1, 3, 24565),

(16, 'bmw', 1, 5, 30760),

(17, 'bmw', 1, 2, 41315),

(18, 'bmw', 1, 2, 36880),

(19, 'chevrolet', 1, 5, 5151),

(20, 'chevrolet', 5, 3, 6295),

(21, 'chevrolet', 1, 3, 6575),

(22, 'dodge', 1, 3, 5572),

(23, 'dodge', 1, 3, 6377),

(24, 'dodge', 2, 3, 7957),

(25, 'dodge', 1, 6, 6229),

(26, 'dodge', 1, 3, 6692),

(27, 'dodge', 1, 3, 7609),

(28, 'dodge', 5, 1, 8558),

(29, 'dodge', 1, 3, 8921),

(30, 'dodge', 4, 3, 12964);

Orders:

INSERT INTO orders (order_id, customer_id, salesperson_id, car_id, order_date)
VALUES

(1, 1, 3, 2, '2024-02-01'),

(2, 2, 5, 5, '2024-02-05'),

(3, 3, 7, 7, '2024-02-08'),

(4, 4, 1, 10, '2024-02-12'),

(5, 5, 2, 12, '2024-02-15'),

(6, 6, 6, 14, '2024-02-18'),

(7, 7, 4, 18, '2024-02-20'),

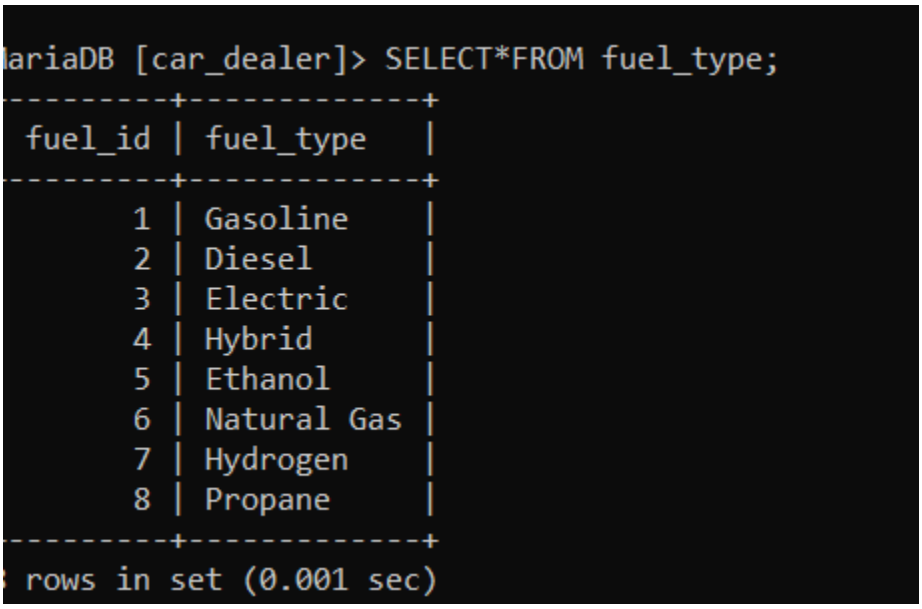
(8, 8, 9, 20, '2024-02-22'),

(9, 9, 8, 25, '2024-02-25'),

(10, 10, 10, 30, '2024-02-28');

In the shell we must run this: source “sql file name”

Make sure that the file name is correct.



```
+-----------+-------------+
| engine_id | engine_name |
+-----------+-------------+
|         1 | DOHC        |
|         2 | OHV         |
|         3 | SOHC        |
|         4 | Inline-4    |
|         5 | V6          |
|         6 | V8          |
+-----------+-------------+
6 rows in set (0.000 sec)
```

```
MariaDB [car_dealer]> SELECT*FROM customers;
+-------------+-----------------+--------------+
| customer_id | name            | phone        |
+-------------+-----------------+--------------+
|           1 | John Doe        | 123-456-7890 |
|           2 | Jane Smith      | 987-654-3210 |
|           3 | Michael Johnson | 555-123-4567 |
|           4 | Emily Davis     | 444-987-6543 |
|           5 | Chris Brown     | 111-222-3333 |
|           6 | Lisa White      | 666-777-8888 |
|           7 | Robert Green    | 999-888-7777 |
|           8 | Sophia Adams    | 333-444-5555 |
|           9 | Daniel Carter   | 222-333-4444 |
|          10 | Olivia Wilson   | 777-666-5555 |
|          11 | Jack Martinez   | 321-654-9870 |
|          12 | Emma Garcia     | 741-852-9630 |
|          13 | Noah Hernandez  | 369-258-1470 |
|          14 | Mia Lopez       | 852-741-3690 |
|          15 | James Wright    | 159-357-9510 |
|          16 | Charlotte Hall  | 456-789-1230 |
|          17 | Benjamin Allen  | 789-456-1230 |
|          18 | Lucas Young     | 951-753-1590 |
|          19 | Harper King     | 357-159-7530 |
|          20 | Elijah Scott    | 258-147-3690 |
|          21 | Aiden Baker     | 654-789-3210 |
|          22 | Madison Adams   | 741-963-8520 |
|          23 | William Nelson  | 369-147-2580 |
|          24 | Evelyn Carter   | 147-258-3690 |
|          25 | Sofia Mitchell  | 357-753-9510 |
|          26 | Henry Perez     | 753-951-3570 |
|          27 | Lily Roberts    | 258-369-1470 |
|          28 | Mason Turner    | 951-357-7530 |
|          29 | Ella Collins    | 123-456-7890 |
|          30 | Samuel Stewart  | 789-123-4560 |
+-------------+-----------------+--------------+
30 rows in set (0.000 sec)
```

```
MariaDB [car_dealer]> SELECT*FROM salespersons;
+---------------+-----------------+-------------+-----------+
| salesperson_id | name           | contact_info | num_sales |
+---------------+-----------------+-------------+-----------+
|              1 | Mike Johnson    | 555-555-5555 |        20 |
|              2 | Anna Davis      | 444-444-4444 |        30 |
|              3 | Chris Lee       | 333-222-1111 |        15 |
|              4 | Emma Taylor     | 111-333-5555 |        25 |
|              5 | Liam Wilson     | 222-555-7777 |        18 |
|              6 | Sophia Martinez | 999-888-7777 |        12 |
|              7 | William Clark   | 888-999-1111 |        22 |
|              8 | Olivia Harris   | 777-666-5555 |        28 |
|              9 | Benjamin White  | 666-777-8888 |        16 |
|             10 | Ava King        | 321-654-9870 |        19 |
+---------------+-----------------+-------------+-----------+
10 rows in set (0.000 sec)
```

```
+------------+--------------------------+
| feature_id | feature_name             |
+------------+--------------------------+
|          1 | GPS                      |
|          2 | Leather Seats            |
|          3 | Sunroof                  |
|          4 | Parking Sensors          |
|          5 | Bluetooth Audio          |
|          6 | Backup Camera            |
|          7 | Keyless Entry            |
|          8 | Adaptive Cruise Control  |
|          9 | Blind Spot Monitoring    |
|         10 | Heated Seats             |
+------------+--------------------------+
0 rows in set (0.077 sec)
```

```
MariaDB [carDB]> SELECT*FROM cars;
+--------+-------------+---------+-----------+----------+
| car_id | make        | fuel_id | engine_id | price    |
+--------+-------------+---------+-----------+----------+
|      1 | alfa-romero |       1 |         1 | 13495.00 |
|      2 | alfa-romero |       1 |         1 | 16500.00 |
|      3 | alfa-romero |       2 |         2 | 16500.00 |
|      4 | audi        |       1 |         2 | 13950.00 |
|      5 | audi        |       1 |         2 | 17450.00 |
|      6 | audi        |       1 |         3 | 15250.00 |
|      7 | audi        |       5 |         2 | 17710.00 |
|      8 | audi        |       3 |         4 | 18920.00 |
|      9 | audi        |       4 |         2 | 23875.00 |
|     10 | audi        |       1 |         2 | 16000.00 |
|     11 | bmw         |       1 |         2 | 16430.00 |
|     12 | bmw         |       1 |         2 | 16925.00 |
|     13 | bmw         |       8 |         6 | 20970.00 |
|     14 | bmw         |       7 |         3 | 21105.00 |
|     15 | bmw         |       1 |         3 | 24565.00 |
|     16 | bmw         |       1 |         5 | 30760.00 |
|     17 | bmw         |       1 |         2 | 41315.00 |
|     18 | bmw         |       1 |         2 | 36880.00 |
|     19 | chevrolet   |       1 |         5 |  5151.00 |
|     20 | chevrolet   |       5 |         3 |  6295.00 |
|     21 | chevrolet   |       1 |         3 |  6575.00 |
|     22 | dodge       |       1 |         3 |  5572.00 |
|     23 | dodge       |       1 |         3 |  6377.00 |
|     24 | dodge       |       2 |         3 |  7957.00 |
|     25 | dodge       |       1 |         6 |  6229.00 |
|     26 | dodge       |       1 |         3 |  6692.00 |
|     27 | dodge       |       1 |         3 |  7609.00 |
|     28 | dodge       |       5 |         1 |  8558.00 |
|     29 | dodge       |       1 |         3 |  8921.00 |
|     30 | dodge       |       4 |         3 | 12964.00 |
+--------+-------------+---------+-----------+----------+
30 rows in set (0.011 sec)
```

```
MariaDB [carDB]> SELECT*FROM orders;
+----------+-------------+----------------+--------+------------+
| order_id | customer_id | salesperson_id | car_id | order_date |
+----------+-------------+----------------+--------+------------+
|        1 |           1 |              3 |      2 | 2024-02-01 |
|        2 |           2 |              5 |      5 | 2024-02-05 |
|        3 |           3 |              7 |      7 | 2024-02-08 |
|        4 |           4 |              1 |     10 | 2024-02-12 |
|        5 |           5 |              2 |     12 | 2024-02-15 |
|        6 |           6 |              6 |     14 | 2024-02-18 |
|        7 |           7 |              4 |     18 | 2024-02-20 |
|        8 |           8 |              9 |     20 | 2024-02-22 |
|        9 |           9 |              8 |     25 | 2024-02-25 |
|       10 |          10 |             10 |     30 | 2024-02-28 |
+----------+-------------+----------------+--------+------------+
10 rows in set (0.002 sec)
```

```
MariaDB [carDB]> SELECT*FROM car_features;
+----------------+--------+------------+
| car_feature_id | car_id | feature_id |
+----------------+--------+------------+
|              1 |      1 |          1 |
|              2 |      2 |          3 |
|              3 |      3 |          5 |
|              4 |      4 |          7 |
|              5 |      5 |          2 |
|              6 |      6 |          4 |
|              7 |      7 |          6 |
|              8 |      8 |          8 |
+----------------+--------+------------+
8 rows in set (0.010 sec)
```

H. Data Manipulation Language (DML) Scripts

Description: Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about

the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.

Rubric: Your work will be graded as follows:

- *1 point each for the two insert statements*
- *1 point each for the two update statements*
- *1 point for the delete statement*
- *1 point for the simple select statement*
- *2 points each for the 2 join statements*
- *2 points each for the two that use summary statements*
- *2 points for the multi-table query*
- *2 points for the query of your choice.*
- *6 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

Total points possible: 24

a.Insert statements:

1.Insert a new customer:

```
INSERT INTO customers (name, phone)
VALUES ('SREEJA', '108-999-180');
```

```
|          26 | Henry Perez     | 753-951-3570 |
|          27 | Lily Roberts    | 258-369-1470 |
|          28 | Mason Turner    | 951-357-7530 |
|          29 | Ella Collins    | 123-456-7890 |
|          30 | Samuel Stewart  | 789-123-4560 |
+-------------+-----------------+--------------+
29 rows in set (0.000 sec)

MariaDB [carDB]> INSERT INTO customers(name,phone)
    -> VALUES ('SREEJA','108-999-180');
Query OK, 1 row affected (0.009 sec)

MariaDB [carDB]> SELECT*FROM customers;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    23
Current database: carDB

+-------------+-----------------+--------------+
| customer_id | name            | phone        |
+-------------+-----------------+--------------+
|           2 | Jane Smith      | 987-654-3210 |
|           3 | Michael Johnson | 555-123-4567 |
|           4 | Emily Davis     | 444-987-6543 |
|           5 | Chris Brown     | 111-222-3333 |
|           6 | Lisa White      | 666-777-8888 |
|           7 | Robert Green    | 999-888-7777 |
|           8 | Sophia Adams    | 333-444-5555 |
|           9 | Daniel Carter   | 222-333-4444 |
|          10 | Olivia Wilson   | 777-666-5555 |
|          11 | Jack Martinez   | 321-654-9870 |
|          12 | Emma Garcia     | 741-852-9630 |
|          13 | Noah Hernandez  | 369-258-1470 |
|          14 | Mia Lopez       | 852-741-3690 |
|          15 | James Wright    | 159-357-9510 |
|          16 | Charlotte Hall  | 456-789-1230 |
|          17 | Benjamin Allen  | 789-456-1230 |
|          18 | Lucas Young     | 951-753-1590 |
|          19 | Harper King     | 357-159-7530 |
|          20 | Elijah Scott    | 258-147-3690 |
|          21 | Aiden Baker     | 654-789-3210 |
|          22 | Madison Adams   | 741-963-8520 |
|          23 | William Nelson  | 369-147-2580 |
|          24 | Evelyn Carter   | 147-258-3690 |
|          25 | Sofia Mitchell  | 357-753-9510 |
|          26 | Henry Perez     | 753-951-3570 |
|          27 | Lily Roberts    | 258-369-1470 |
|          28 | Mason Turner    | 951-357-7530 |
|          29 | Ella Collins    | 123-456-7890 |
|          30 | Samuel Stewart  | 789-123-4560 |
|          33 | SREEJA          | 108-999-180  |
+-------------+-----------------+--------------+
30 rows in set (0.057 sec)
```

Insert into a new car:

INSERT INTO cars (make, fuel_id, engine_id, price)

VALUES ('Toyota Corolla', 1, 2, 22000.00);

```
    -> INSERT INTO (make,fuel_id,engine_id,price)
    -> VALUES('Toyota Corolla',1,2,22000.00);
ERROR 1064 (42000): You have an error in your SQL syntax; che
rsion for the right syntax to use near 'INSERT INTO (make,fue
VALUES('Toyota Corolla',1,2,22000.00)' at line 5
MariaDB [carDB]> INSERT INTO cars (make,fuel_id,engine_id,pri
    -> VALUES('Toyota Corolla',1,2,22000.00);
Query OK, 1 row affected (0.019 sec)

MariaDB [carDB]> SELECT*FROM cars;
+--------+----------------+---------+-----------+----------+
| car_id | make           | fuel_id | engine_id | price    |
+--------+----------------+---------+-----------+----------+
|      1 | alfa-romero    |       1 |         1 | 13495.00 |
|      2 | alfa-romero    |       1 |         1 | 16500.00 |
|      3 | alfa-romero    |       2 |         2 | 16500.00 |
|      4 | audi           |       1 |         2 | 13950.00 |
|      5 | audi           |       1 |         2 | 17450.00 |
|      6 | audi           |       1 |         3 | 15250.00 |
|      7 | audi           |       5 |         2 | 17710.00 |
|      8 | audi           |       3 |         4 | 18920.00 |
|      9 | audi           |       4 |         2 | 23875.00 |
|     10 | audi           |       1 |         2 | 16000.00 |
|     11 | bmw            |       1 |         2 | 16430.00 |
|     12 | bmw            |       1 |         2 | 16925.00 |
|     13 | bmw            |       8 |         6 | 20970.00 |
|     14 | bmw            |       7 |         3 | 21105.00 |
|     15 | bmw            |       1 |         3 | 24565.00 |
|     16 | bmw            |       1 |         5 | 30760.00 |
|     17 | bmw            |       1 |         2 | 41315.00 |
|     18 | bmw            |       1 |         2 | 36880.00 |
|     19 | chevrolet      |       1 |         5 |  5151.00 |
|     20 | chevrolet      |       5 |         3 |  6295.00 |
|     21 | chevrolet      |       1 |         3 |  6575.00 |
|     22 | dodge          |       1 |         3 |  5572.00 |
|     23 | dodge          |       1 |         3 |  6377.00 |
|     24 | dodge          |       2 |         3 |  7957.00 |
|     25 | dodge          |       1 |         6 |  6229.00 |
|     26 | dodge          |       1 |         3 |  6692.00 |
|     27 | dodge          |       1 |         3 |  7609.00 |
|     28 | dodge          |       5 |         1 |  8558.00 |
|     29 | dodge          |       1 |         3 |  8921.00 |
|     30 | dodge          |       4 |         3 | 12964.00 |
|     31 | Toyota Corolla |       1 |         2 | 22000.00 |
+--------+----------------+---------+-----------+----------+
31 rows in set (0.002 sec)
```

b.Update statements:

1.Update a customer’s phone number:

UPDATE customers

SET phone = '789-825-154'

WHERE customer_id = 5;

```
+-------------+-----------------+--------------+
0 rows in set (0.009 sec)

ariaDB [carDB]> UPDATE customers
    -> SET phone='789-825-154'
    -> WHERE customer_id=5;
uery OK, 1 row affected (0.005 sec)
ows matched: 1  Changed: 1  Warnings: 0

ariaDB [carDB]> SELECT*FROM customers;
+-------------+-----------------+--------------+
| customer_id | name            | phone        |
+-------------+-----------------+--------------+
|           2 | Jane Smith      | 987-654-3210 |
|           3 | Michael Johnson | 555-123-4567 |
|           4 | Emily Davis     | 444-987-6543 |
|           5 | Chris Brown     | 789-825-154  |
|           6 | Lisa White      | 666-777-8888 |
|           7 | Robert Green    | 999-888-7777 |
|           8 | Sophia Adams    | 333-444-5555 |
|           9 | Daniel Carter   | 222-333-4444 |
|          10 | Olivia Wilson   | 777-666-5555 |
|          11 | Jack Martinez   | 321-654-9870 |
|          12 | Emma Garcia     | 741-852-9630 |
|          13 | Noah Hernandez  | 369-258-1470 |
|          14 | Mia Lopez       | 852-741-3690 |
|          15 | James Wright    | 159-357-9510 |
|          16 | Charlotte Hall  | 456-789-1230 |
|          17 | Benjamin Allen  | 789-456-1230 |
|          18 | Lucas Young     | 951-753-1590 |
|          19 | Harper King     | 357-159-7530 |
|          20 | Elijah Scott    | 258-147-3690 |
|          21 | Aiden Baker     | 654-789-3210 |
|          22 | Madison Adams   | 741-963-8520 |
|          23 | William Nelson  | 369-147-2580 |
|          24 | Evelyn Carter   | 147-258-3690 |
|          25 | Sofia Mitchell  | 357-753-9510 |
|          26 | Henry Perez     | 753-951-3570 |
|          27 | Lily Roberts    | 258-369-1470 |
|          28 | Mason Turner    | 951-357-7530 |
|          29 | Ella Collins    | 123-456-7890 |
|          30 | Samuel Stewart  | 789-123-4560 |
|          33 | SREEJA          | 108-999-180  |
+-------------+-----------------+--------------+
0 rows in set (0.000 sec)
```

2.Update the price of a car:

UPDATE cars

SET price = 25000.00

WHERE car_id = 5;

```
riaDB [carDB]> UPDATE cars
    -> SET price=25000.00
    -> WHERE car_id=5;
uery OK, 1 row affected (0.007 sec)
ows matched: 1  Changed: 1  Warnings: 0

riaDB [carDB]> SELECT*FROM cars;
+--------+----------------+---------+-----------+----------+
| car_id | make           | fuel_id | engine_id | price    |
+--------+----------------+---------+-----------+----------+
|      1 | alfa-romero    |       1 |         1 | 13495.00 |
|      2 | alfa-romero    |       1 |         1 | 16500.00 |
|      3 | alfa-romero    |       2 |         2 | 16500.00 |
|      4 | audi           |       1 |         2 | 13950.00 |
|      5 | audi           |       1 |         2 | 25000.00 |
|      6 | audi           |       1 |         3 | 15250.00 |
|      7 | audi           |       5 |         2 | 17710.00 |
|      8 | audi           |       3 |         4 | 18920.00 |
|      9 | audi           |       4 |         2 | 23875.00 |
|     10 | audi           |       1 |         2 | 16000.00 |
|     11 | bmw            |       1 |         2 | 16430.00 |
|     12 | bmw            |       1 |         2 | 16925.00 |
|     13 | bmw            |       8 |         6 | 20970.00 |
|     14 | bmw            |       7 |         3 | 21105.00 |
|     15 | bmw            |       1 |         3 | 24565.00 |
|     16 | bmw            |       1 |         5 | 30760.00 |
|     17 | bmw            |       1 |         2 | 41315.00 |
|     18 | bmw            |       1 |         2 | 36880.00 |
|     19 | chevrolet      |       1 |         5 |  5151.00 |
|     20 | chevrolet      |       5 |         3 |  6295.00 |
|     21 | chevrolet      |       1 |         3 |  6575.00 |
|     22 | dodge          |       1 |         3 |  5572.00 |
|     23 | dodge          |       1 |         3 |  6377.00 |
|     24 | dodge          |       2 |         3 |  7957.00 |
|     25 | dodge          |       1 |         6 |  6229.00 |
|     26 | dodge          |       1 |         3 |  6692.00 |
|     27 | dodge          |       1 |         3 |  7609.00 |
|     28 | dodge          |       5 |         1 |  8558.00 |
|     29 | dodge          |       1 |         3 |  8921.00 |
|     30 | dodge          |       4 |         3 | 12964.00 |
|     31 | Toyota Corolla |       1 |         2 | 22000.00 |
+--------+----------------+---------+-----------+----------+
 rows in set (0.000 sec)
```

c.Delete statements:

1.Remove a customer:

DELETE FROM customers

WHERE customer_id = 5;

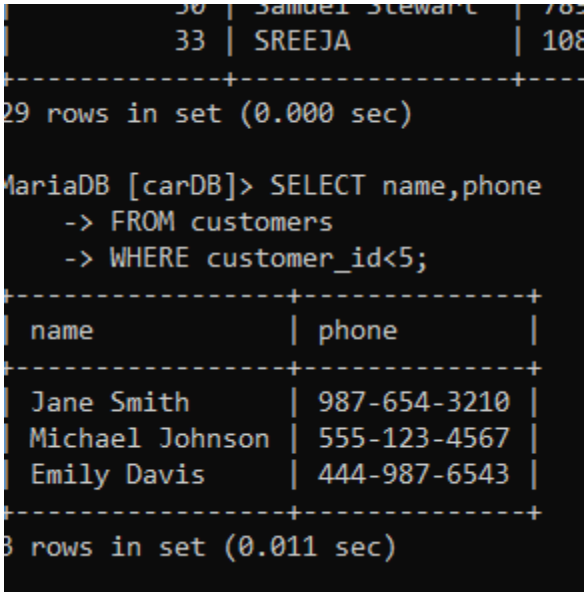
```
1 rows in set (0.000 sec)

ariaDB [carDB]> DELETE FROM customers
    -> WHERE customer_id=5;
uery OK, 1 row affected (0.009 sec)

ariaDB [carDB]> SELECT*FROM customers;
+-------------+-----------------+--------------+
| customer_id | name            | phone        |
+-------------+-----------------+--------------+
|           2 | Jane Smith      | 987-654-3210 |
|           3 | Michael Johnson | 555-123-4567 |
|           4 | Emily Davis     | 444-987-6543 |
|           6 | Lisa White      | 666-777-8888 |
|           7 | Robert Green    | 999-888-7777 |
|           8 | Sophia Adams    | 333-444-5555 |
|           9 | Daniel Carter   | 222-333-4444 |
|          10 | Olivia Wilson   | 777-666-5555 |
|          11 | Jack Martinez   | 321-654-9870 |
|          12 | Emma Garcia     | 741-852-9630 |
|          13 | Noah Hernandez  | 369-258-1470 |
|          14 | Mia Lopez       | 852-741-3690 |
|          15 | James Wright    | 159-357-9510 |
|          16 | Charlotte Hall  | 456-789-1230 |
|          17 | Benjamin Allen  | 789-456-1230 |
|          18 | Lucas Young     | 951-753-1590 |
|          19 | Harper King     | 357-159-7530 |
|          20 | Elijah Scott    | 258-147-3690 |
|          21 | Aiden Baker     | 654-789-3210 |
|          22 | Madison Adams   | 741-963-8520 |
|          23 | William Nelson  | 369-147-2580 |
|          24 | Evelyn Carter   | 147-258-3690 |
|          25 | Sofia Mitchell  | 357-753-9510 |
|          26 | Henry Perez     | 753-951-3570 |
|          27 | Lily Roberts    | 258-369-1470 |
|          28 | Mason Turner    | 951-357-7530 |
|          29 | Ella Collins    | 123-456-7890 |
|          30 | Samuel Stewart  | 789-123-4560 |
|          33 | SREEJA          | 108-999-180  |
+-------------+-----------------+--------------+
9 rows in set (0.000 sec)
```

d.Select specific customer details:

SELECT name, phone

FROM customers

WHERE customer_id < 5;



e.Select statements with joins:

1. List orders with customers name:

SELECT orders.order_id, customers.name, orders.order_date

FROM orders

JOIN customers ON orders.customer_id = customers.customer_id;

```
MariaDB [carDB]> SELECT orders.order_id, customers.name, orders.order_date
    -> FROM orders
    -> JOIN customers ON orders.customer_id = customers.customer_id;
+----------+-----------------+------------+
| order_id | name            | order_date |
+----------+-----------------+------------+
|        2 | Jane Smith      | 2024-02-05 |
|        3 | Michael Johnson | 2024-02-08 |
|        4 | Emily Davis     | 2024-02-12 |
|        6 | Lisa White      | 2024-02-18 |
|        7 | Robert Green    | 2024-02-20 |
|        8 | Sophia Adams    | 2024-02-22 |
|        9 | Daniel Carter   | 2024-02-25 |
|       10 | Olivia Wilson   | 2024-02-28 |
+----------+-----------------+------------+
8 rows in set (0.006 sec)
```

2.Show cars slod with salesperson details:

SELECT cars.make, salespersons.name

FROM orders

JOIN cars ON orders.car_id = cars.car_id

JOIN salespersons ON orders.salesperson_id = salespersons.salesperson_id;

```
MariaDB [carDB]> SELECT cars.make, salespersons.name
    -> FROM orders
    -> JOIN cars ON orders.car_id = cars.car_id
    -> JOIN salespersons ON orders.salesperson_id = salespersons.salesperson_id;
+-----------+-----------------+
| make      | name            |
+-----------+-----------------+
| audi      | Liam Wilson     |
| audi      | William Clark   |
| audi      | Mike Johnson    |
| bmw       | Sophia Martinez |
| bmw       | Emma Taylor     |
| chevrolet | Benjamin White  |
| dodge     | Olivia Harris   |
| dodge     | Ava King        |
+-----------+-----------------+
8 rows in set (0.006 sec)

MariaDB [carDB]>
```

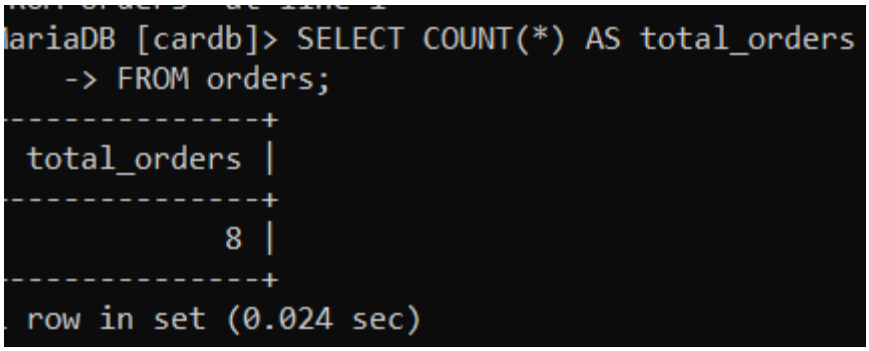
f.Queries with summary functions:

Use functions like COUNT, SUM, AVG, MAX, MIN.

1.Count total orders:

SELECT COUNT(*) AS total_orders

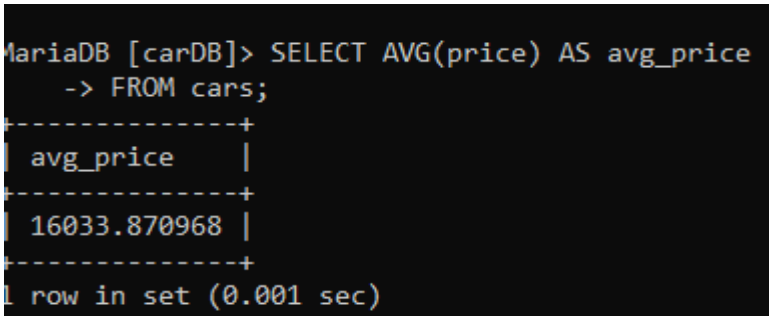
FROM orders;



2.Calculate average car price:

SELECT AVG(price) AS avg_price

FROM cars;



g.Multi-table query

1.Show all order details including car,customer and salesperson:

SELECT orders.order_id, customers.name AS customer_name, cars.make AS car_model, salespersons.name AS salesperson_name

FROM orders

JOIN customers ON orders.customer_id = customers.customer_id

JOIN cars ON orders.car_id = cars.car_id

JOIN salespersons ON orders.salesperson_id = salespersons.salesperson_id;

```
MariaDB [carDB]> SELECT orders.order_id, customers.name AS customer_name, cars.make AS car_model, salespersons.name A
S salesperson_name
    -> FROM orders
    -> JOIN customers ON orders.customer_id = customers.customer_id
    -> JOIN cars ON orders.car_id = cars.car_id
    -> JOIN salespersons ON orders.salesperson_id = salespersons.salesperson_id;
+----------+-----------------+-----------+-----------------+
| order_id | customer_name   | car_model | salesperson_name |
+----------+-----------------+-----------+-----------------+
|        2 | Jane Smith      | audi      | Liam Wilson     |
|        3 | Michael Johnson | audi      | William Clark   |
|        4 | Emily Davis     | audi      | Mike Johnson    |
|        6 | Lisa White      | bmw       | Sophia Martinez |
|        7 | Robert Green    | bmw       | Emma Taylor     |
|        8 | Sophia Adams    | chevrolet | Benjamin White  |
|        9 | Daniel Carter   | dodge     | Olivia Harris   |
|       10 | Olivia Wilson   | dodge     | Ava King        |
+----------+-----------------+-----------+-----------------+
```

SELECT DISTINCT orders.customer_id

-> FROM orders

-> INNER JOIN customers

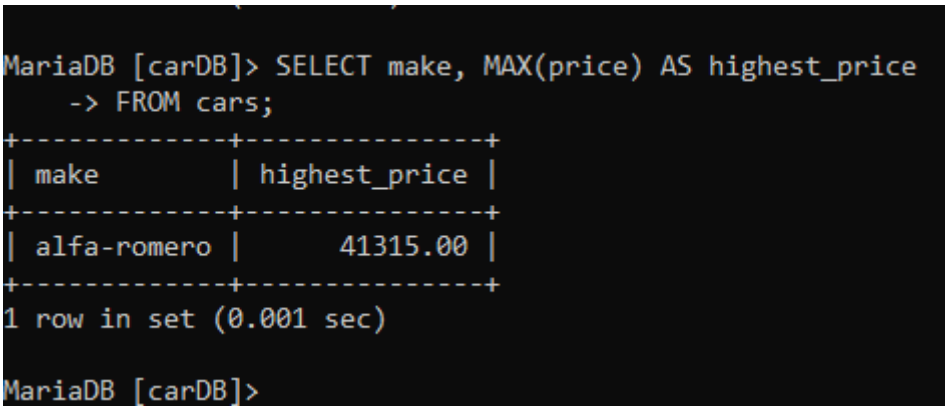
-> ON orders.customer_id = customers.customer_id;

```
MariaDB [cardb]> SELECT DISTINCT orders.customer_id
    -> FROM orders
    -> INNER JOIN customers
    -> ON orders.customer_id = customers.customer_id;
+-------------+
| customer_id |
+-------------+
|           2 |
|           3 |
|           4 |
|           6 |
|           7 |
|           8 |
|           9 |
|          10 |
+-------------+
8 rows in set (0.012 sec)
```

h.Custom query :finding expensive car sold:

SELECT make, MAX(price) AS highest_price

FROM cars;



I. Indexes

Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.

Rubric: Your work will be graded as follows:

- *3 points for clearly defining at least three indexes and explaining why you chose them.*
- *3 points for showing the sql needed to generate the indexes*
- *2 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

Total points possible: 8

Indexes in MYSQL are used to speed data retrieval.Which allows MYSQL to quickly locate rows instead of scanning the hwole table.

Indexes are used when:

- On columns used in WHERE,JOIN and ORDER BY chances.
- On columns most searched or filtered.

Performance is improved by adding index on columns used in frequent queries.

#1: customer_id in the orders table because:

- This column is often used to join the orders and customers tables.
- Indexing speeds up these joins and searches by customer_id.

SQL to create index:

CREATE INDEX idx_customer_id ON orders(customer_id);

```
Database changed
MariaDB [cardb]> CREATE INDEX idx_customer_id ON orders(customer_id);
ERROR 1061 (42000): Duplicate key name 'idx_customer_id'
MariaDB [cardb]> SHOW INDEX FROM orders;
+--------+------------+---------------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name            | Seq_in_index | Column_name   | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+---------------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| orders |          0 | PRIMARY             |            1 | order_id      | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | salesperson_id      |            1 | salesperson_id | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | idx_customer_id     |            1 | customer_id   | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | idx_car_salesperson |            1 | car_id        | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | idx_car_salesperson |            2 | salesperson_id | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
+--------+------------+---------------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
5 rows in set (0.001 sec)
```

#2: make in the cars table because:

- It is seacrhed to find cars by brand.
- Indexing improves search performance for car makes.

SQL to create index:

CREATE INDEX idx_make ON cars(make);

```
ERROR 1061 (42000): Duplicate key name 'idx_make'
MariaDB [cardb]> SHOW INDEX FROM cars;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| cars  |          0 | PRIMARY  |            1 | car_id      | A         |          31 |     NULL | NULL   |      | BTREE      |         |               |
| cars  |          1 | idx_make |            1 | make        | A         |          15 |     NULL | NULL   |      | BTREE      |         |               |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
2 rows in set (0.003 sec)
```

#3: composite index on car_id and salesperson_id in the orders table:

- Composite index cover multiple columns for multi-column searches.
- Used for filtering both car and salesperson.

SQL to create composite index:

CREATE INDEX idx_car_salesperson ON orders(car_id, salesperson_id);

```
MariaDB [cardb]> SHOW INDEX FROM orders;
+--------+------------+---------------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name            | Seq_in_index | Column_name   | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+---------------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| orders |          0 | PRIMARY             |            1 | order_id      | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | salesperson_id      |            1 | salesperson_id | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | idx_customer_id     |            1 | customer_id   | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | idx_car_salesperson |            1 | car_id        | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
| orders |          1 | idx_car_salesperson |            2 | salesperson_id | A         |           8 |     NULL | NULL   |      | BTREE      |         |               |
+--------+------------+---------------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
5 rows in set (0.001 sec)
```

```
MariaDB [cardb]> EXPLAIN SELECT * FROM orders WHERE customer_id = 15;
+------+-------------+--------+------+-----------------+-----------------+---------+-------+------+-------+
| id   | select_type | table  | type | possible_keys   | key             | key_len | ref   | rows | Extra |
+------+-------------+--------+------+-----------------+-----------------+---------+-------+------+-------+
|    1 | SIMPLE      | orders | ref  | idx_customer_id | idx_customer_id | 4       | const | 1    |       |
+------+-------------+--------+------+-----------------+-----------------+---------+-------+------+-------+
1 row in set (0.000 sec)
```

Usage of EXPALIN to measure performance improvements

- **Before the index**: You will likely see ALL under the type column (full table scan – slow).
- **After the index**: You should see ref under type (indexed access – fast)

J. Views

Description: Add two views to your database to provide easy access to combinations of data from multiple tables.

Rubric: Your work will be graded as follows:

- *3 points for including the SQL for generating the two views in your Word document*
- *3 points for including screenshots for the data contained in each view in your Word document*
- *3 points for explaining why each view is a valuable addition to your database*

Total points possible: 9

Views are virtual tables in MySQL that present the results of a query. They simplify data access by combining multiple tables or presenting only essential columns.

#1: Customer Order Summary

- It provides a **summary** of customer orders by joining the customers and orders tables.

SQL to create view:

CREATE VIEW customer_order_summary AS

-> SELECT customers.customer_id, customers.name, orders.order_id, orders.order_date

-> FROM customers

-> JOIN orders ON customers.customer_id = orders.customer_id;

Query OK, 0 rows affected (0.040 sec)

```
MariaDB [cardb]> CREATE VIEW customer_order_summary AS
    -> SELECT customers.customer_id, customers.name, orders.order_id, orders.order_date
    -> FROM customers
    -> JOIN orders ON customers.customer_id = orders.customer_id;
Query OK, 0 rows affected (0.040 sec)

MariaDB [cardb]> SELECT * FROM customer_order_summary;
+-------------+-----------------+----------+------------+
| customer_id | name            | order_id | order_date |
+-------------+-----------------+----------+------------+
|           2 | Jane Smith      |        2 | 2024-02-05 |
|           3 | Michael Johnson |        3 | 2024-02-08 |
|           4 | Emily Davis     |        4 | 2024-02-12 |
|           6 | Lisa White      |        6 | 2024-02-18 |
|           7 | Robert Green    |        7 | 2024-02-20 |
|           8 | Sophia Adams    |        8 | 2024-02-22 |
|           9 | Daniel Carter   |        9 | 2024-02-25 |
|          10 | Olivia Wilson   |       10 | 2024-02-28 |
+-------------+-----------------+----------+------------+
8 rows in set (0.020 sec)
```

#2: Car Sales Details

- This provides a detailed view of car sales, combining data from cars, salespersons, and orders.

SQL ro create view:

CREATE VIEW car_sales_details AS

SELECT cars.make, cars.price, salespersons.name AS salesperson_name, orders.order_date

FROM orders

JOIN cars ON orders.car_id = cars.car_id

JOIN salespersons ON orders.salesperson_id = salespersons.salesperson_id;

```
MariaDB [cardb]> CREATE VIEW car_sales_details AS
    -> SELECT cars.make, cars.price, salespersons.name AS salesperson_name, orders.order_date
    -> FROM orders
    -> JOIN cars ON orders.car_id = cars.car_id
    -> JOIN salespersons ON orders.salesperson_id = salespersons.salesperson_id;
Query OK, 0 rows affected (0.006 sec)

MariaDB [cardb]>  SELECT * FROM car_sales_details;
+-----------+----------+-----------------+------------+
| make      | price    | salesperson_name | order_date |
+-----------+----------+-----------------+------------+
| audi      | 25000.00 | Liam Wilson      | 2024-02-05 |
| audi      | 17710.00 | William Clark    | 2024-02-08 |
| audi      | 16000.00 | Mike Johnson     | 2024-02-12 |
| bmw       | 21105.00 | Sophia Martinez  | 2024-02-18 |
| bmw       | 36880.00 | Emma Taylor      | 2024-02-20 |
| chevrolet |  6295.00 | Benjamin White   | 2024-02-22 |
| dodge     |  6229.00 | Olivia Harris    | 2024-02-25 |
| dodge     | 12964.00 | Ava King         | 2024-02-28 |
+-----------+----------+-----------------+------------+
```

K. Stored Programs (Stored Procedures, Stored Functions, Triggers)

Description: Add a stored procedure, stored function or trigger to a table and demonstrate using it.

Rubric: Your work will be graded as follows:

- *3 points for including the SQL for the stored program (procedure, function, or trigger in your Word document*
- *3 points for clearly explaining the purpose of the stored program*
- *3 points for a screenshot and explanation that shows the stored program in action.*

Total points possible: 9

Stored Programs in MySQL include procedures, functions, and triggers. They automate repetitive tasks and enforce business rules.

Add a New Order

- It simplifies adding new orders by bundling multiple steps into one reusable program.

SQL:

DELIMITER $$

MariaDB [cardb]>

MariaDB [cardb]> CREATE PROCEDURE AddOrder(IN cust_id INT, IN car_id INT, IN salesperson_id INT)

-> BEGIN

-> INSERT INTO orders (customer_id, car_id, salesperson_id, order_date)

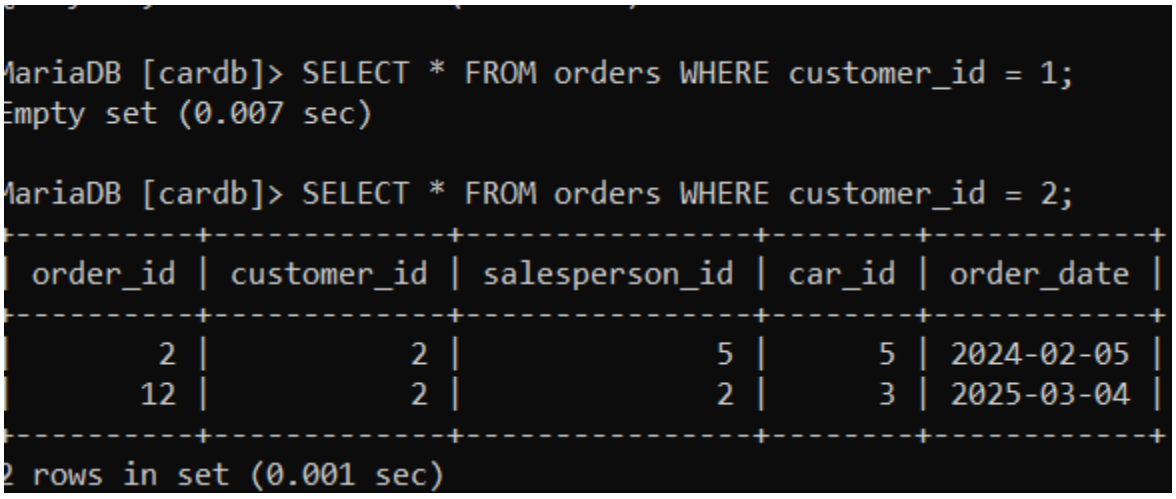
-> VALUES (cust_id, car_id, salesperson_id, CURDATE());

-> END$$

Query OK, 0 rows affected (0.015 sec)

CALL AddOrder(1, 3, 2);

- This is calling a stored procedure named AddOrder.
- Parameters:
  - 2 → customer_id
  - 3 → car_id
  - 2 → salesperson_id



```
MariaDB [cardb]> SELECT * FROM orders WHERE customer_id = 1;
Empty set (0.007 sec)

MariaDB [cardb]> SELECT * FROM orders WHERE customer_id = 2;
+----------+-------------+----------------+--------+------------+
| order_id | customer_id | salesperson_id | car_id | order_date |
+----------+-------------+----------------+--------+------------+
|        2 |           2 |              5 |      5 | 2024-02-05 |
|       12 |           2 |              2 |      3 | 2025-03-04 |
+----------+-------------+----------------+--------+------------+
2 rows in set (0.001 sec)
```

Trigger: Automatically Update Car Count

- It updates the car stock after a sale.

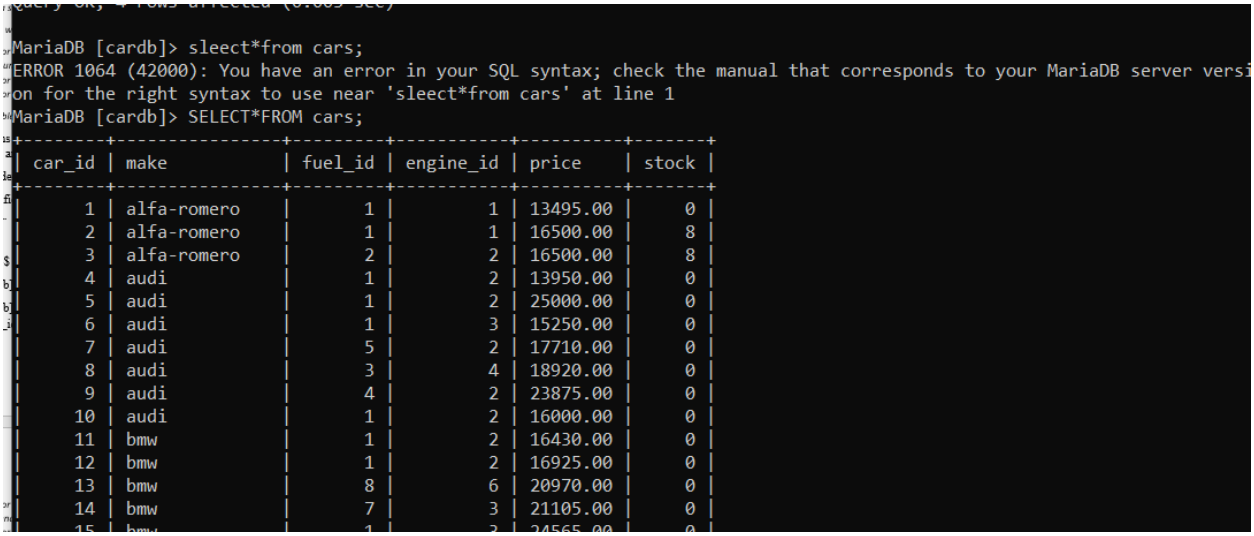
SQL:

UPDATE cars

-> SET stock = 10

-> WHERE car_id =

MariaDB [cardb]> CALL AddOrder(2, 3, 4);



```
MariaDB [cardb]> sleect*from cars;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server versi
on for the right syntax to use near 'sleect*from cars' at line 1
MariaDB [cardb]> SELECT*FROM cars;
+--------+----------------+---------+-----------+----------+-------+
| car_id | make           | fuel_id | engine_id | price    | stock |
+--------+----------------+---------+-----------+----------+-------+
|      1 | alfa-romero    |       1 |         1 | 13495.00 |     0 |
|      2 | alfa-romero    |       1 |         1 | 16500.00 |     8 |
|      3 | alfa-romero    |       2 |         2 | 16500.00 |     8 |
|      4 | audi           |       1 |         2 | 13950.00 |     0 |
|      5 | audi           |       1 |         2 | 25000.00 |     0 |
|      6 | audi           |       1 |         3 | 15250.00 |     0 |
|      7 | audi           |       5 |         2 | 17710.00 |     0 |
|      8 | audi           |       3 |         4 | 18920.00 |     0 |
|      9 | audi           |       4 |         2 | 23875.00 |     0 |
|     10 | audi           |       1 |         2 | 16000.00 |     0 |
|     11 | bmw            |       1 |         2 | 16430.00 |     0 |
|     12 | bmw            |       1 |         2 | 16925.00 |     0 |
|     13 | bmw            |       8 |         6 | 20970.00 |     0 |
|     14 | bmw            |       7 |         3 | 21105.00 |     0 |
```

car_id 2: The stock value is 8, which means there are 8 units of this car available.

L. Transactions

Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?

Rubric: Your work will be graded as follows:

- *5 points for clearly explaining the importance of transactions to ensuring ACID behavior*
- *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

Total points possible: 8

Transactions are crucial for maintaining the integrity and consistency of data in a database. They ensure that the database adheres to the ACID properties:

1. **Atomicity**:
    - Ensures that all operations within a transaction are completed successfully. If any operation fails, the entire transaction is rolled back, leaving the database in its previous state.
    - Example: If you're transferring money between two bank accounts, both the debit and credit operations must succeed. If one fails, neither should be applied.
2. **Consistency**:

- Guarantees that a transaction brings the database from one valid state to another, maintaining all predefined rules and constraints.
- Example: If a transaction violates a database constraint (like a foreign key constraint), the entire transaction is rolled back to maintain consistency.

3. **Isolation**:
    - Ensures that transactions are executed independently of one another, preventing interference and maintaining data integrity.
    - Example: If two transactions are trying to update the same data simultaneously, isolation ensures that one transaction's changes are not visible to the other until it is complete.
4. **Durability**:
    - Ensures that once a transaction is committed, its changes are permanent and will survive any subsequent system failures.
    - Example: Once a transaction to update a customer's order is committed, the changes will persist even if the system crashes immediately afterward.

SQL:

```
START TRANSACTION;

INSERT INTO orders (customer_id, car_id, salesperson_id, order_date)

VALUES (2, 3, 4, CURDATE());


UPDATE cars

SET stock = stock - 1

WHERE car_id = 3;

COMMIT;
```

Before(cars table):

```
MariaDB [cardb]> SELECT*FROM cars;
+--------+----------------+---------+-----------+----------+-------+
| car_id | make           | fuel_id | engine_id | price    | stock |
+--------+----------------+---------+-----------+----------+-------+
|      1 | alfa-romero    |       1 |         1 | 13495.00 |     0 |
|      2 | alfa-romero    |       1 |         1 | 16500.00 |     8 |
|      3 | alfa-romero    |       2 |         2 | 16500.00 |     1 |
|      4 | audi           |       1 |         2 | 13950.00 |     0 |
|      5 | audi           |       1 |         2 | 25000.00 |     0 |
|      6 | audi           |       1 |         3 | 15250.00 |     0 |
|      7 | audi           |       5 |         2 | 17710.00 |     0 |
|      8 | audi           |       3 |         4 | 18920.00 |     0 |
|      9 | audi           |       4 |         2 | 23875.00 |     0 |
|     10 | audi           |       1 |         2 | 16000.00 |     0 |
|     11 | bmw            |       1 |         2 | 16430.00 |     0 |
|     12 | bmw            |       1 |         2 | 16925.00 |     0 |
|     13 | bmw            |       8 |         6 | 20970.00 |     0 |
|     14 | bmw            |       7 |         3 | 21105.00 |     0 |
|     15 | bmw            |       1 |         3 | 24565.00 |     0 |
|     16 | bmw            |       1 |         5 | 30760.00 |     0 |
|     17 | bmw            |       1 |         2 | 41315.00 |     0 |
```

After(cars table):

```
ows matched: 1  Changed: 1  Warnings: 0

ariaDB [cardb]> COMMIT;
uery OK, 0 rows affected (0.004 sec)

ariaDB [cardb]> SELECT * FROM cars;
--------+----------------+---------+-----------+----------+-------+
 car_id | make           | fuel_id | engine_id | price    | stock |
--------+----------------+---------+-----------+----------+-------+
      1 | alfa-romero    |       1 |         1 | 13495.00 |     0 |
      2 | alfa-romero    |       1 |         1 | 16500.00 |     8 |
      3 | alfa-romero    |       2 |         2 | 16500.00 |     0 |
      4 | audi           |       1 |         2 | 13950.00 |     0 |
      5 | audi           |       1 |         2 | 25000.00 |     0 |
      6 | audi           |       1 |         3 | 15250.00 |     0 |
      7 | audi           |       5 |         2 | 17710.00 |     0 |
      8 | audi           |       3 |         4 | 18920.00 |     0 |
      9 | audi           |       4 |         2 | 23875.00 |     0 |
     10 | audi           |       1 |         2 | 16000.00 |     0 |
     11 | bmw            |       1 |         2 | 16430.00 |     0 |
     12 | bmw            |       1 |         2 | 16925.00 |     0 |
     13 | bmw            |       8 |         6 | 20970.00 |     0 |
     14 | bmw            |       7 |         3 | 21105.00 |     0 |
     15 | bmw            |       1 |         3 | 24565.00 |     0 |
     16 | bmw            |       1 |         5 | 30760.00 |     0 |
     17 | bmw            |       1 |         2 | 41315.00 |     0 |
     18 | bmw            |       1 |         2 | 36880.00 |     0 |
```

A new order has been added to the orders list.

M. Database Security

Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.

Rubric: Your work will be graded as follows:

- *4 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
- *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
- *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

Total points possible: 12

Database Security

In a database system like cardb, different users interact with the database based on their job responsibilities. It is essential to define specific privileges for each user to maintain **data security**, **data integrity**, and **performance**. Below are the key users:

a) Admin User

- **Role**: Manages the entire database (full access).
- **Privileges**:
    - Create, delete, and modify all tables.
    - Manage user permissions.

b) Salesperson User

- **Role**: Records new customer orders and retrieves information about cars.
- **Privileges**:
    - **SELECT** – To view customer and car information.
    - **INSERT** – To add new orders.
- **Restrictions**: No permission to **DELETE** or **ALTER** records for safety reasons.

c) Manager User

- **Role**: Oversees sales and customer information.
- **Privileges**:

- **SELECT** – To view all data.
- **UPDATE** – To modify order records.
- **INSERT** – To add new sales records.

- **Restrictions**: No access to **DROP** or manage **users**.

2. SQL GRANT Statements

a) Create Admin User and Grant Full Privileges

```
CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin_password';
GRANT ALL PRIVILEGES ON cardb.* TO 'admin_user'@'localhost';
FLUSH PRIVILEGES;
```

b) Create Salesperson User with Limited Privileges

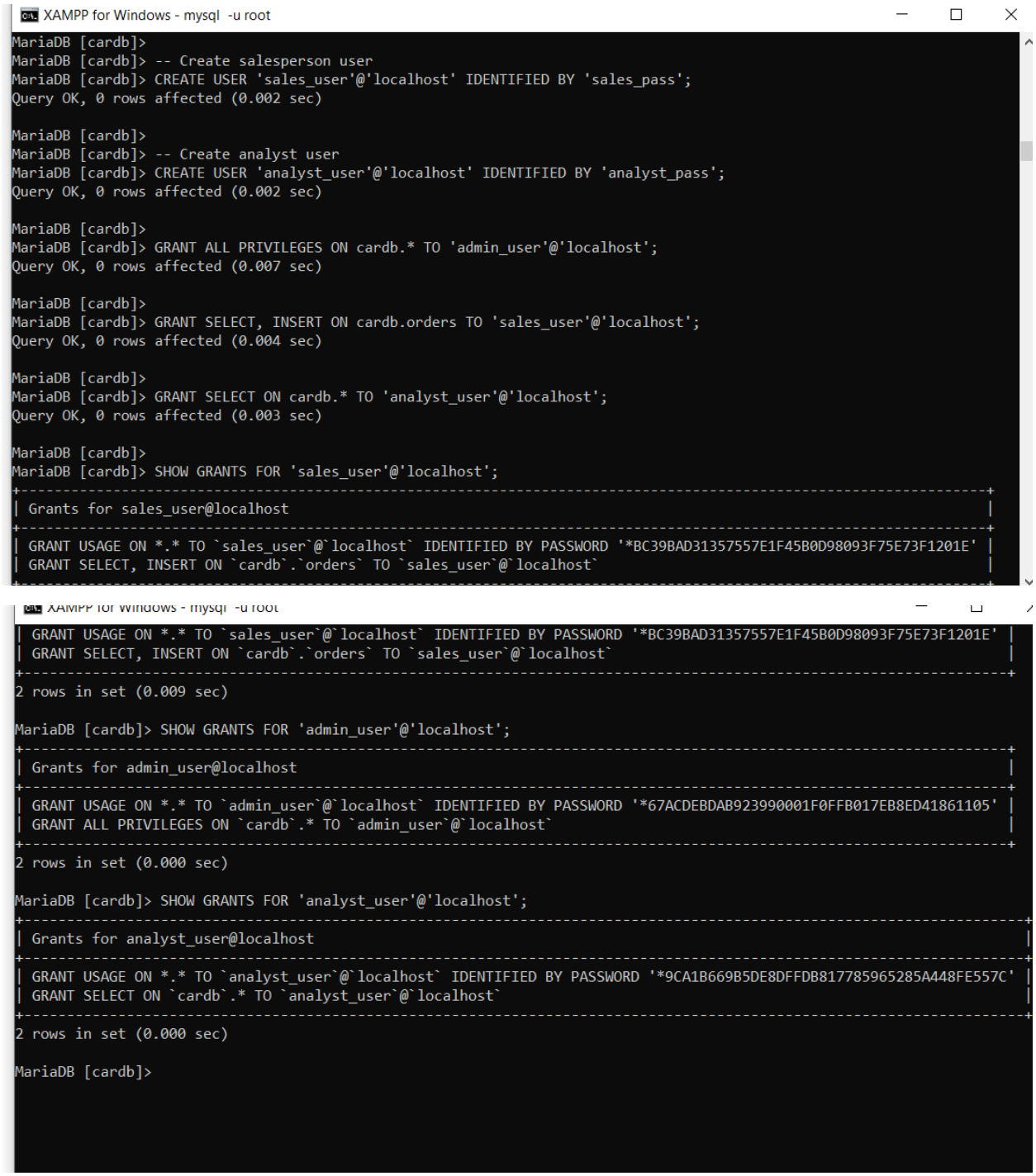
```
CREATE USER 'sales_user'@'localhost' IDENTIFIED BY 'sales_password';
GRANT SELECT, INSERT ON cardb.orders TO 'sales_user'@'localhost';
GRANT SELECT ON cardb.cars TO 'sales_user'@'localhost';
FLUSH PRIVILEGES;
```

The sales_user can:

- View car details (via SELECT).
- Insert new orders but **cannot delete or modify existing records**.

c) Create Manager User with Moderate Privileges

```
CREATE USER 'manager_user'@'localhost' IDENTIFIED BY 'manager_password';
GRANT SELECT, INSERT, UPDATE ON cardb.* TO 'manager_user'@'localhost';
FLUSH PRIVILEGES;
```



Test Salesperson's Restricted Access

1. Log in as the **sales_user**:
2. mysql -u sales_user -p
3. INSERT INTO orders (customer_id, car_id, salesperson_id, order_date)

4. VALUES (1, 3, 2, CURDATE());

Failed Action: Attempt to delete a record:

```
Query OK, 0 rows affected (0.013 sec)

MariaDB [(none)]> EXIT;
Bye

sreej@LAPTOP-IFR6TN80 c:\xampp
# mysql -u sales_user -p
Enter password: *************
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE cardb;
Database changed
MariaDB [cardb]> DELETE FROM orders WHERE order_id = 1;
ERROR 1142 (42000): DELETE command denied to user 'sales_user'@'localhost' for table `cardb`.`orders`
```

User Privileges and their Importance:

1. **Data Security**: Restricting access prevents accidental or malicious changes to sensitive data.
2. **Integrity**: Ensure that only authorized users can update or insert new records.
3. **Compliance**: Follows the principle of least privilege—giving users only the access they need.
4. **Performance**: Reducing the scope of user access can improve query efficiency and database stability.

N. Locking and Concurrent Access

Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.

Rubric: Your work will be graded as follows:

- *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
- *3 points for providing a screenshot and accompanying explanation of locking tables.*

Total points possible: 5

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

O. Backing Up Your Database

Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?

Rubric: Your work will be graded as follows:

- *6 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
- *2 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

Total points possible: 8

ENTER YOUR WORK ON DATABASE BACKUPS HERE

P. Programming

Description: Write a Python, Java, or PHP program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.

Rubric: Your work will be graded as follows:

- *10 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
- *2 points for posting the code to GitHub*
- *6 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

Total points possible: 18

ENTER YOUR PYTHON, PHP, or JAVA DATABASE PROGRAMMING WORK HERE

Q. Suggested Future Work

Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.

Rubric: Your work will be graded as follows:

- *3 points for clearly describing the limitations of your databases*

- *3 points for explaining how you would address these shortcomings*
- *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
- *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

Total points possible: 12

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

R. Activity Log

Description: As an appendix, the team will keep a frequently updated diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.

If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.

Total points possible: 24

Week 1

- Wed**:** Researched database models and finalized the Car Sales and Customer Management Database. Identified 8 key entities and assigned data formats (CSV, JSON, XML). Planned the overall database structure.
- Fri: Collected and cleaned car data from an automobile dataset, removing unnecessary columns. Created cars.csv with car details including car_id, make, fuel_id, engine_id, and price.
- Sun: Developed customers.json with customer_id, name, and phone. Began structuring orders.xml to link customers and car purchases.

Week 2

- Mon**:** Created salespersons.csv with salesperson ID, name, contact info, and number of sales. Checked data for completeness and consistency.
- Tue**:** Worked on features.csv listing car features and car_features.csv for car-feature mapping. Ensured proper relationships between car_id and feature_id.

- Sat Finalized fuel_types.json and engine_types.xml with structured identifiers. Reviewed all files for formatting and consistency.

Week 3

- Identify entity sets and relationships, including connectivity and participation.
- Develop the logical model by adding attributes and identifying functional dependencies.
- Apply normalization to ensure the design satisfies First, Second, and Third Normal Forms.

Week 4

- Draw the logical model as an ERD, showing entity sets, relationships, attributes, and primary identifiers.
- Add data types, size constraints, uniqueness constraints, and auto-incrementing for all attributes in the physical model.
- Generate SQL DDL statements to create the database, its tables, and relationships, and run these statements in MySQL.

Week 5

- Defined at least three indexes and explained why I chose them.
- Showed the SQL needed to generate the indexes.
- Explained how I would demonstrate the performance improvement afforded by the indexes.
- Added 2 views to the database to provide easy access to combinations of data from multiple tables.
- Included the SQL for generating the two views in the Word document.
- Included screenshots for the data contained in each view in the Word document.
- Explained why each view is a valuable addition to the database.

Week 6

- Added a stored procedure, stored function, or trigger to a table and demonstrate using it.
- Included the SQL for the stored program in the Word document.

- Explained the purpose of the stored program.
- Provided a screenshot and explanation that shows the stored program in action.
- Explained the importance of transactions to ensuring ACID behavior.
- Include a screenshot and accompanying explanation of a MySQL transaction.
- Identified the different kinds of users who will use my database.
- Wrote GRANT statements to define the privileges for these different kinds of users.