|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **Chapter No** | **Chapter Name** | **Page No** |
|  | Acknowledgement |  |
|  | Table of Contents | 1 |
| 1 | Introduction  1.1 About a project | 2  2 |
| 2 | Scope of the project  2.1 Problem statement  2.2 Objectives | 3  3  3 |
| 3 | System Requirements and Specifications  3.1 Functional Requirements  3.2 Non-Functional Requirements  3.3 Software Requirements  3.4Hardware Requirements | 4-6  4  5  6  6 |
| 4 | Methodology  4.1 Schema Diagram  4.2 ER Diagram | 7-8  7  8 |
| 5 | Implementation  5.1 Table Description  5.2 Source Code  5.3 Connections | 9-18  9-10  11-17  18 |
| 6 | Result | 19-33 |
|  | Conclusion | 34 |
|  | References | 35 |

**CHAPTER 01**

**INTRODUCTION**

A car showroom management system serves as the backbone of operations for car

dealerships, providing a comprehensive suite of tools to efficiently manage inventory,

customer relationships, sales processes, and after-sales services. With robust inventory

management capabilities, dealers can effortlessly track the status of each vehicle, from

acquisition to sale, while maintaining crucial details such as make, model, and pricing.

Integrated CRM functionalities enable dealerships to cultivate strong relationships with

customers by storing vital information like contact details, preferences, and purchase

histories, facilitating personalized interactions and targeted marketing efforts. Streamlined

sales processes are facilitated through automation, allowing for seamless lead tracking, quote

generation, negotiation, and contract management. Finance and accounting modules handle

invoicing, payment processing and financial reporting, ensuring smooth financial operations.

* 1. **ABOUT THE PROJECT:**

The Car Showroom Management System is a mini-project designed to streamline the operations of a car dealership using a Database Management System (DBMS). This system efficiently handles various functions including inventory management, customer relations, sales tracking, and employee oversight. By leveraging a relational database, the system ensures accurate record-keeping and quick access to critical information, ultimately enhancing operational efficiency and customer service. It provides a comprehensive platform to manage car models, customer details, sales transactions, and service requests, facilitating better decision-making and improved showroom management.

**CHAPTER 02**

**SCOPE OF THE PROJECT**

**2.1 PROBLEM STATEMENT**:

The Car Showroom Management System aims to address inefficiencies and errors in manual dealership operations by automating inventory, sales, customer, and employee management. Traditional methods are time-consuming and prone to inconsistencies. The project seeks to create a centralized database solution for accurate, real-time information access. This will enhance operational efficiency and improve customer service.

**2.2 OBJECTIVES OF THE PROJECT:**

The objective of this project is to develop a comprehensive database management system (DBMS) for efficiently managing a car showroom's operations. This system helps to streamline processes, maintain accurate records, and provide a user-friendly interface for managing car inventory, customer details, sales transactions, and employee information.

**1. Inventory Management:**

- Track car models, specifications, prices, and availability.

- Update inventory based on sales and new arrivals.

**2. Customer Management:**

- Store customer details, including name, phone number, and contact information.

- Manage customer inquiries and test drive requests.

**3. Sales Management:**

- Record sales transactions, including car details, customer information, and date of sale.

- Generate sales reports and analyze sales trends

**4. Employee Management:**

- Maintain employee records, including personal details, position, and salary.

**CHAPTER 03**

**SYSTEM REQUIREMENTS**

**3.1 FUNCTIONAL REQUIREMENTS:**

**1. User Management:**

- Add, update, and delete employee records.

- Manage user authentication and authorization.

**2. Car Management:**

- Add, update, and delete car records.

- Track car details including model, brand, and price.

**3. Customer Management:**

- Add, update, and delete customer records.

- Store customer contact information and purchase history.

**4. Sales Management:**

- Record sales transactions, linking car and customer records.

- Track date of sale and generate sales reports.

**5. Inventory Management:**

- Update inventory based on sales and new arrivals.

- Monitor available car stock levels.

**6. Data Retrieval:**

- Display records from car, customer, sales, and employee tables.

- Generate reports on inventory, sales, and employee performance.

**7. Data Validation:**

- Ensure data integrity by validating inputs for each field.

**3.2 NON-FUNCTIONAL REQUIREMENTS:**

**1. Performance:**

- System should handle a moderate number of concurrent users efficiently.

- Queries and data retrieval should be optimized for quick response times.

**2. Reliability:**

- Ensure data accuracy and consistency.

- Implement error handling and transaction management to prevent data loss.

**3. Usability:**

- Provide a user-friendly interface for ease of navigation and operation.

- Ensure the system is easy to learn and use for non-technical users.

**4. Scalability:**

- Design the system to handle an increasing number of records and users.

- Allow for future enhancements without significant changes to the system architecture.

**5. Security:**

- Implement user authentication and role-based access control.

- Ensure data is protected from unauthorized access and breaches.

**6. Maintainability:**

- Write clean and well-documented code for ease of maintenance.

- Ensure the system can be easily updated with new features and fixes.

**7. Portability:**

- Ensure the system can run on different operating systems (Windows, macOS, Linux).

- Allow easy migration of the database to different servers if needed.

**8. Backup and Recovery:**

- Implement regular data backup procedures.

- Ensure the system can recover data in case of hardware or software failures.

**3.3 HARDWARE REQUIREMENTS**

**1. Processor:** Dual-core processor or higher (Intel i3 or equivalent)

**2. RAM:** Minimum 4GB RAM (8GB recommended for better performance)

**3. Storage:** Minimum 20GB free disk space

**4. Monitor:** Minimum resolution of 1024x768 pixels

**5. Input Devices:** Keyboard and mouse

**3.4 SOFTWARE REQUIREMENTS**

**1. Operating System:** Windows 7 or higher, macOS, or Linux (Ubuntu preferred)

**2. Database Server:** MySQL Server 5.7 or higher

**3. Programming Language:** Python 3.6 or higher

**4. Python Libraries:**- mysql-connector-python

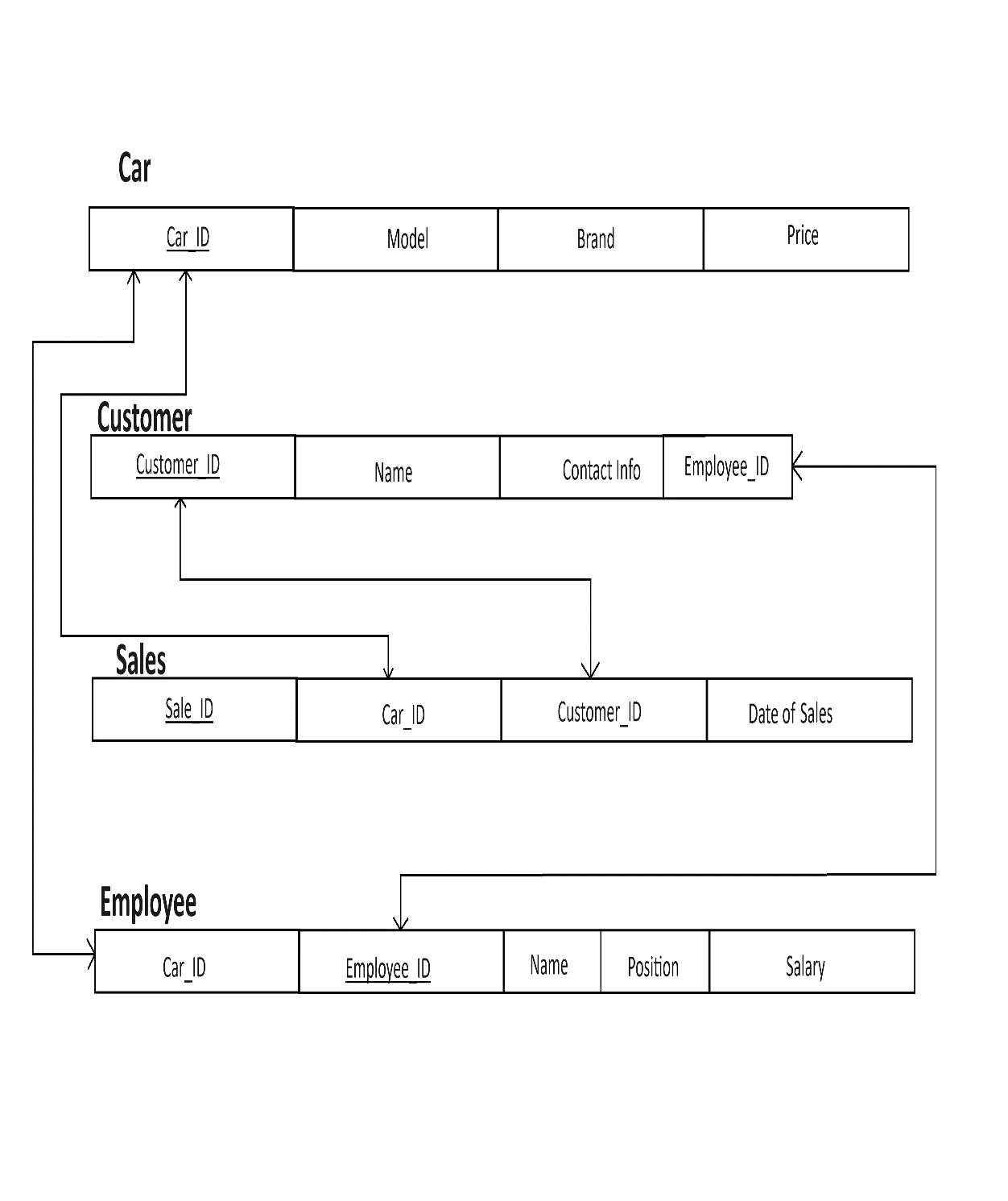
**5. Text Editor/IDE:** Visual Studio Code, PyCharm, or any other preferred text editor/IDE

**6. Web Browser:** Latest version of Google Chrome, Firefox, or any modern browser for accessing documentation and online resources.

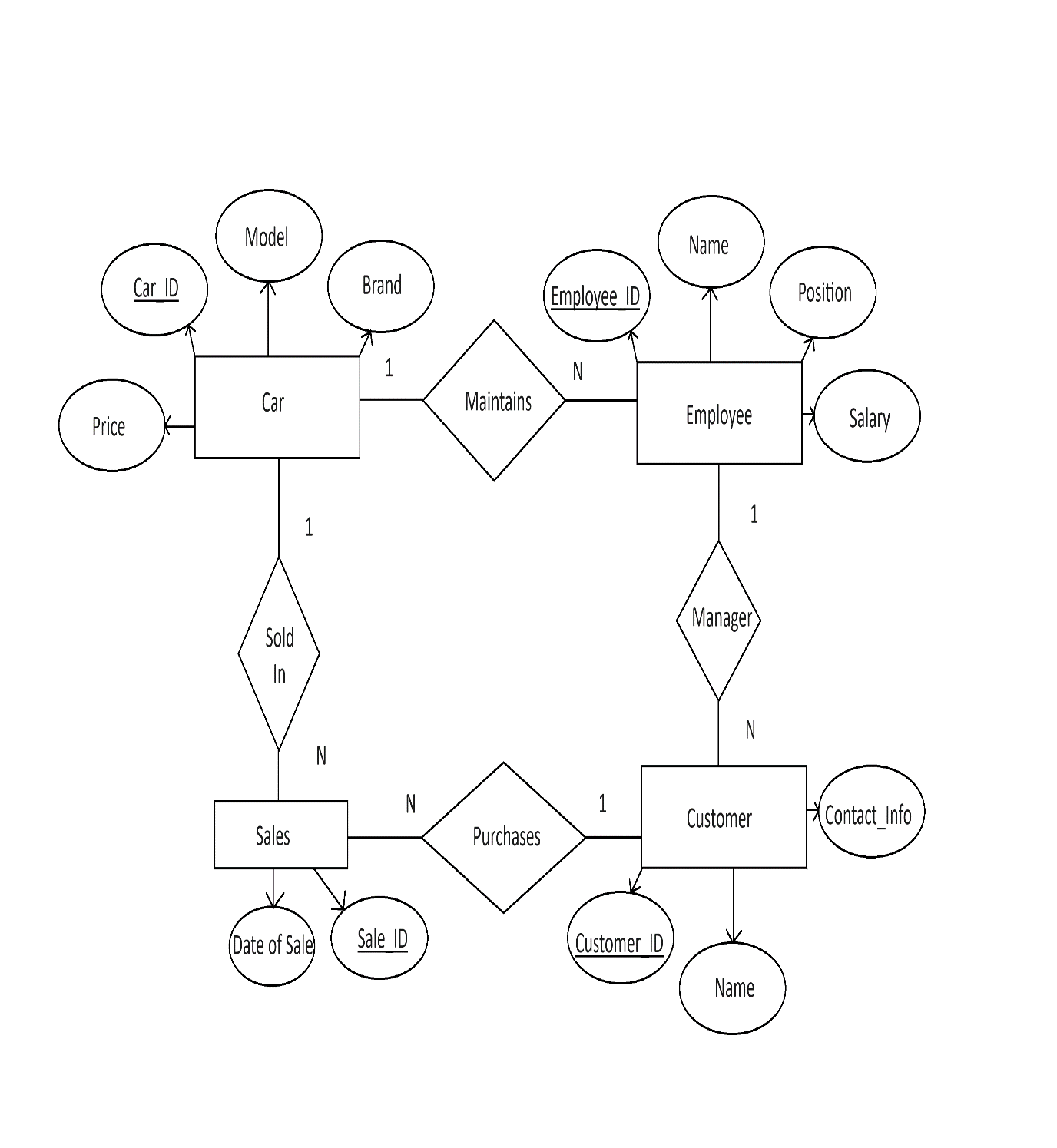
**CHAPTER 04**

**METHODOLOGY**

**4.2 SCHEMA DAIGRAM**



**4.1 ER DIAGRAM**



**CHAPTER 05**

**IMPLEMENTATION**

**5.1 TABLE DESCRIPTION**

car

Stores information about the cars available in the showroom.

* car\_id: INT - Primary Key. Unique identifier for each car.
* model: VARCHAR(50) - Model name of the car.
* brand: VARCHAR(50) - Brand name of the car.
* price: INT - Price of the car.

employee

Stores information about the employees working in the showroom.

* emp\_id: INT - Primary Key. Unique identifier for each employee.
* car\_id: INT - Foreign Key. References car(car\_id). The car that the employee is associated with.
* name: VARCHAR(50) - Name of the employee.
* position: VARCHAR(50) - Position or job title of the employee.
* salary: INT - Salary of the employee.

customer

Stores information about the customers.

* customer\_id: INT - Primary Key. Unique identifier for each customer.
* emp\_id: INT - Foreign Key. References employee(emp\_id). The employee who is handling this customer.
* name: VARCHAR(50) - Name of the customer.
* contact\_info: VARCHAR(100) - Contact information of the customer.

sales

Stores information about the sales transactions.

* sale\_id: INT - Primary Key. Unique identifier for each sale.
* car\_id: INT - Foreign Key. References car(car\_id). The car that is sold.
* customer\_id: INT - Foreign Key. References customer(customer\_id). The customer who bought the car.
* date\_of\_sale: DATE - The date when the sale was made.

Functions Description

add\_record()

Prompts the user to enter data for a specific table (car, employee, customer, or sales) and inserts the data into the corresponding table.

display\_record()

Prompts the user to enter the table name and displays all the records in that table. The column headers are displayed in uppercase.

delete\_record()

Prompts the user to enter the tablname, the column name of the primary key, and the ID of the record to delete. Deletes the specified record from the table.

update\_record()

Prompts the user to enter the table name, the column name of the primary key, the ID of the record to update, the column name to update, and the new value. Updates the specified record in the table.

**5.2 SOURCE CODE**

import mysql.connector as mycon

# Establish connection

con = mycon.connect(host='localhost', user='root', password="chetana123")

cur = con.cursor()

# Create database and tables if they don't exist

cur.execute("CREATE DATABASE IF NOT EXISTS car\_showroom\_management")

cur.execute("USE car\_showroom\_management")

# Creating tables

cur.execute("""

    CREATE TABLE IF NOT EXISTS car (

        car\_id INT PRIMARY KEY,

        model VARCHAR(50),

        brand VARCHAR(50),

        price INT

    )

""")

cur.execute("""

    CREATE TABLE IF NOT EXISTS employee (

        emp\_id INT PRIMARY KEY,

        car\_id INT,

        name VARCHAR(50),

        position VARCHAR(50),

        salary INT,

        FOREIGN KEY (car\_id) REFERENCES car(car\_id)

    )

""")

cur.execute("""

    CREATE TABLE IF NOT EXISTS customer (

        customer\_id INT PRIMARY KEY,

        emp\_id INT,

        name VARCHAR(50),

        contact\_info VARCHAR(100),

        FOREIGN KEY (emp\_id) REFERENCES employee(emp\_id)

    )

""")

cur.execute("""

    CREATE TABLE IF NOT EXISTS sales (

        sale\_id INT PRIMARY KEY,

        car\_id INT,

        customer\_id INT,

        date\_of\_sale DATE,

        FOREIGN KEY (car\_id) REFERENCES car(car\_id),

        FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id)

    )

""")

con.commit()

def add\_record():

    table = input("Enter the table name (car, employee, customer, sales): ")

    if table == "car":

        car\_id = int(input("Enter Car ID: "))

        model = input("Enter Model: ")

        brand = input("Enter Brand: ")

        price = int(input("Enter Price: "))

        query = "INSERT INTO car (car\_id, model, brand, price) VALUES (%s, %s, %s, %s)"

        values = (car\_id, model, brand, price)

    elif table == "employee":

        emp\_id = int(input("Enter Employee ID: "))

        car\_id = int(input("Enter Car ID: "))

        name = input("Enter Name: ")

        position = input("Enter Position: ")

        salary = int(input("Enter Salary: "))

        query = "INSERT INTO employee (emp\_id, car\_id, name, position, salary) VALUES (%s, %s, %s, %s, %s)"

        values = (emp\_id, car\_id, name, position, salary)

    elif table == "customer":

        customer\_id = int(input("Enter Customer ID: "))

        emp\_id = int(input("Enter Employee ID: "))

        name = input("Enter Name: ")

        contact\_info = input("Enter Contact Info: ")

        query = "INSERT INTO customer (customer\_id, emp\_id, name, contact\_info) VALUES (%s, %s,%s, %s)"

        values = (customer\_id, emp\_id, name, contact\_info)

    elif table == "sales":

        sale\_id = int(input("Enter Sale ID: "))

        car\_id = int(input("Enter Car ID: "))

        customer\_id = int(input("Enter Customer ID: "))

        date\_of\_sale = input("Enter Date of Sale (YYYY-MM-DD): ")

        query = "INSERT INTO sales (sale\_id, car\_id, customer\_id, date\_of\_sale) VALUES (%s, %s, %s, %s)"

        values = (sale\_id, car\_id, customer\_id, date\_of\_sale)

    else:

        print("Invalid table name.")

        return

    cur.execute(query, values)

    con.commit()

    print("## Data Saved ##")

def display\_record():

    table = input("Enter the table name (car, employee, customer, sales): ")

    query = "SELECT \* FROM {}".format(table)

    cur.execute(query)

    result = cur.fetchall()

    headers = [desc[0].upper() for desc in cur.description]

    print(f"{'   '.join(headers)}")

    for row in result:

        print(f"{'     '.join(str(item) for item in row)}")

def delete\_record():

    table = input("Enter the table name (car, employee, customer, sales): ")

    column = input("Enter the column name of the primary key: ")

    id = input("Enter the ID of the record to delete: ")

    query = "DELETE FROM {} WHERE {} = %s".format(table, column)

    cur.execute(query, (id,))

    con.commit()

    print("## Record Deleted ##")

def update\_record():

    table = input("Enter the table name (car, employee, customer, sales): ")

    column = input("Enter the column name of the primary key: ")

    id = input("Enter the ID of the record to update: ")

    update\_col = input("Enter the column name to update: ")

    new\_value = input("Enter the new value: ")

    query = "UPDATE {} SET {} = %s WHERE {} = %s".format(table, update\_col, column)

    cur.execute(query, (new\_value, id))

    con.commit()

    print("## Record Updated ##")

choice = None

while choice != 0:

    print("1. ADD RECORD")

    print("2. DISPLAY RECORD")

    print("3. DELETE RECORD")

    print("4. UPDATE RECORD")

    print("0. EXIT")

    choice = int(input("Enter Choice: ")

    if choice == 1:

        add\_record()

    elif choice == 2:

        display\_record()

    elif choice == 3:

        delete\_record()

    elif choice == 4:

        update\_record()

    elif choice == 0:

        con.close()

        print("## Bye!! ##")

    else:

print("## INVALID CHOICE ##")

**5.3 CONNECTIONS**

Car

: This table does not reference any other tables. It stores basic information about the cars.

employee:

* car\_id: This is a foreign key that references the car\_id in the car table. This implies that each employee is associated with a specific car.

customer:

* emp\_id: This is a foreign key that references the emp\_id in the employee table. This indicates which employee is handling a particular customer.

sales:

* car\_id: This is a foreign key that references the car\_id in the car table. This indicates which car was sold in a particular sale.
* customer\_id: This is a foreign key that references the customer\_id in the customer table. This indicates which customer made the purchase.

**Data Integrity**

The use of foreign keys ensures referential integrity between the tables. This means:

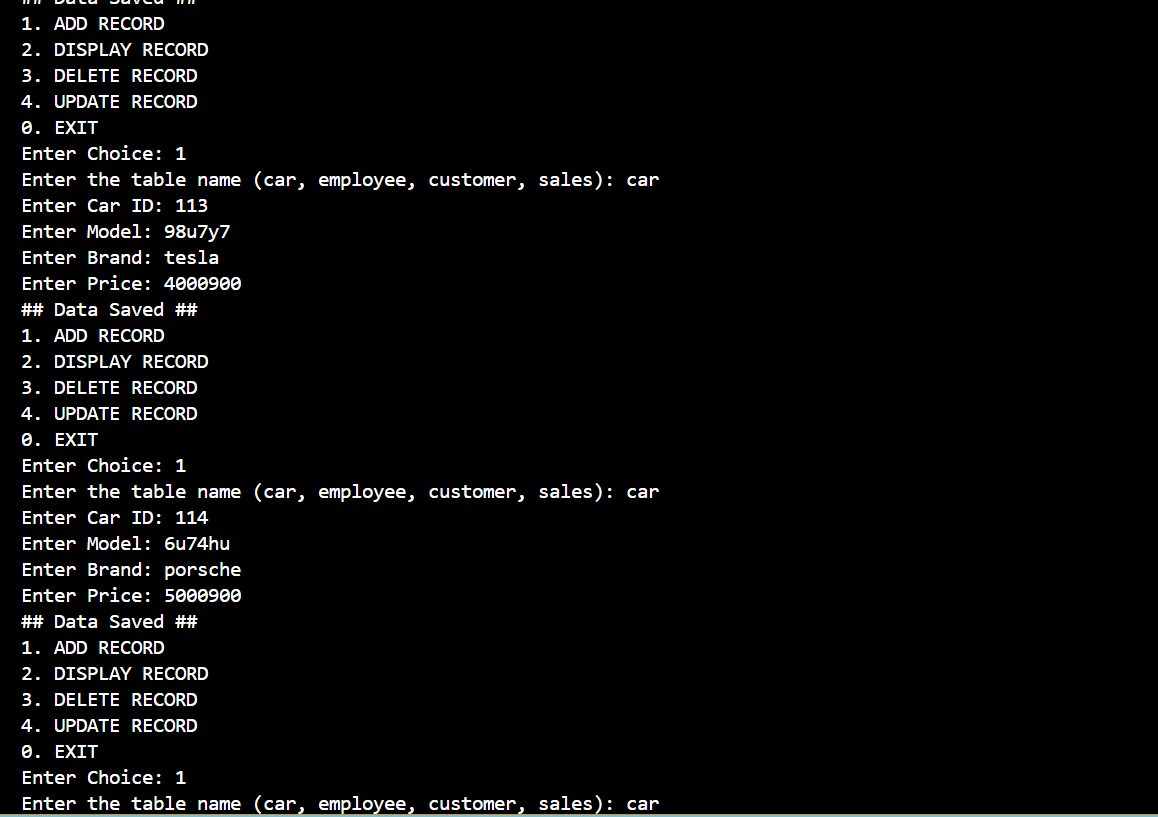
* An employee must be associated with an existing car.
* A customer must be associated with an existing employee.
* A sale must involve an existing car and an existing customer.

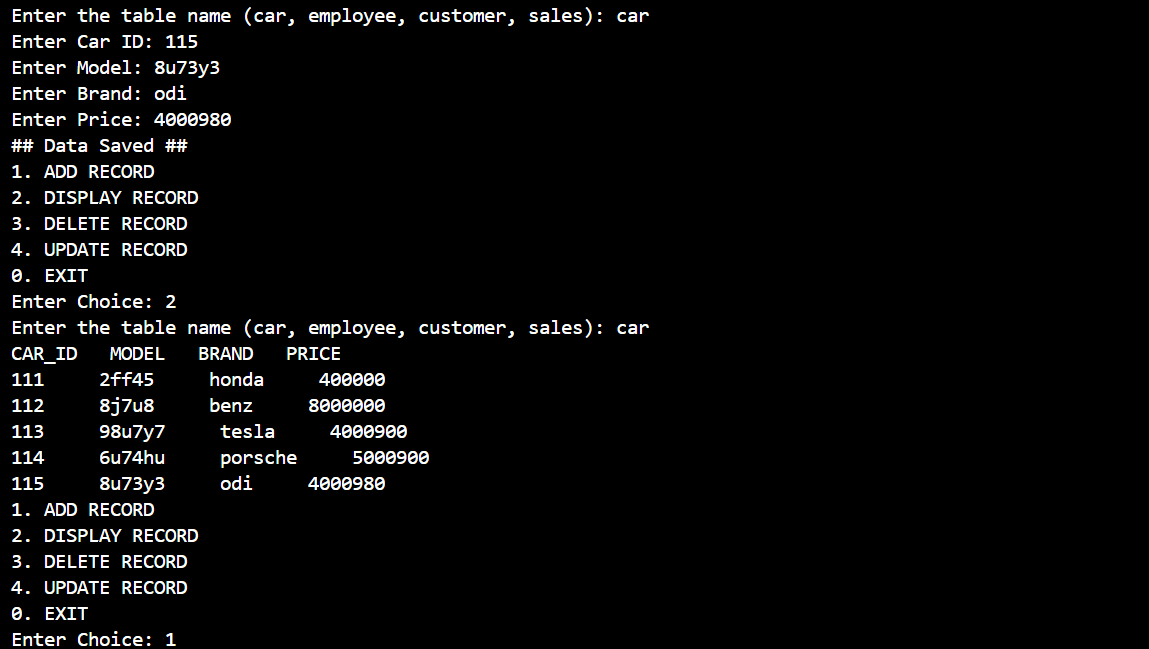
These connections help maintain a consistent and accurate database structure for the car showroom management system.

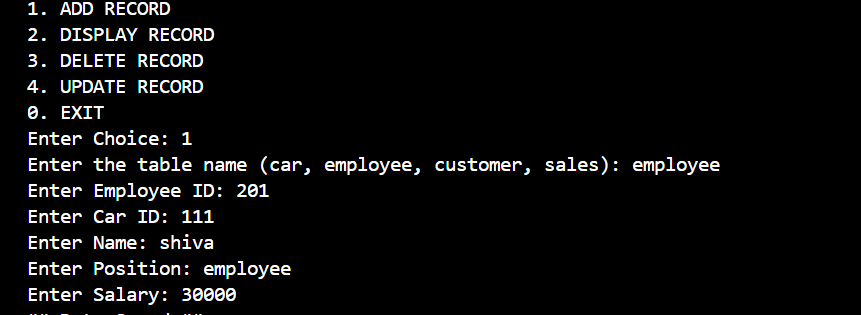
**CHAPTER 06**

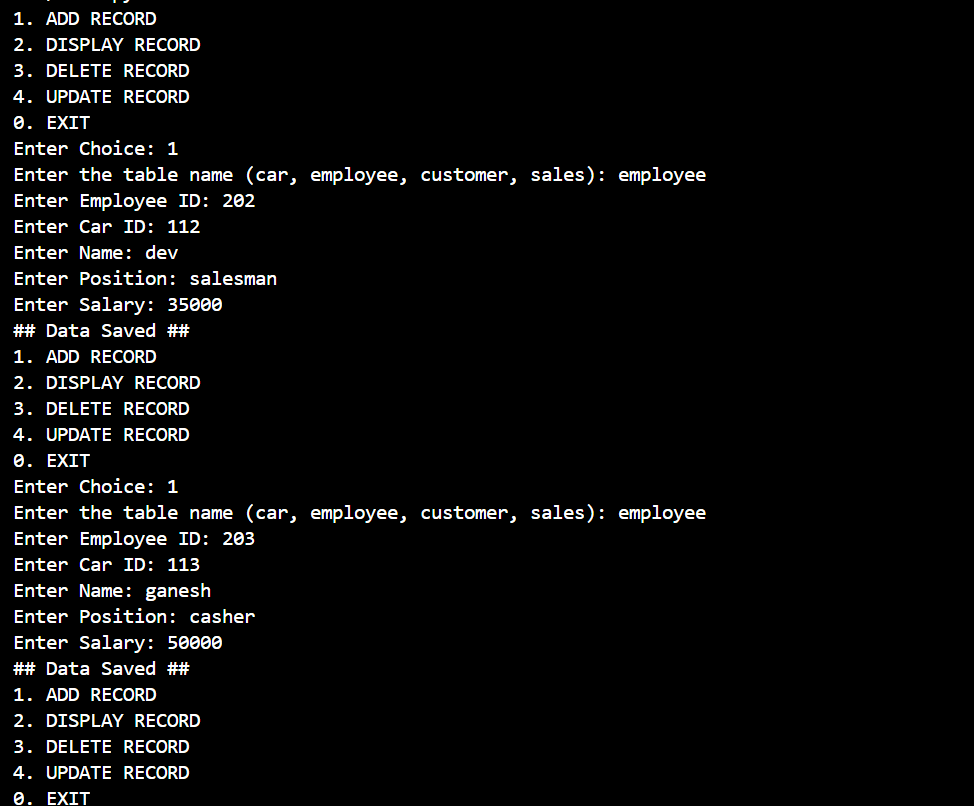
**RESULT**

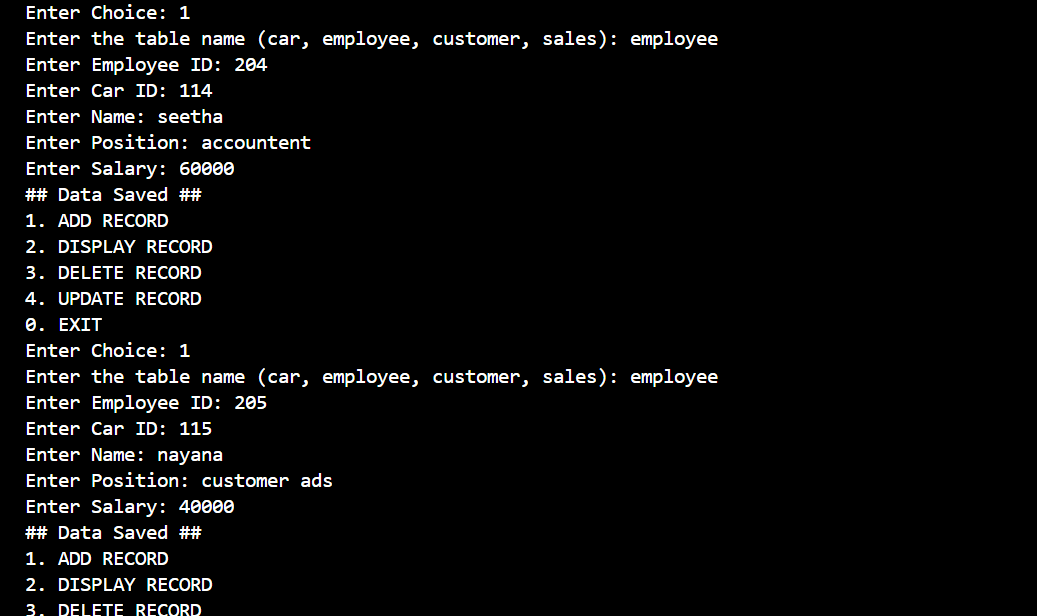


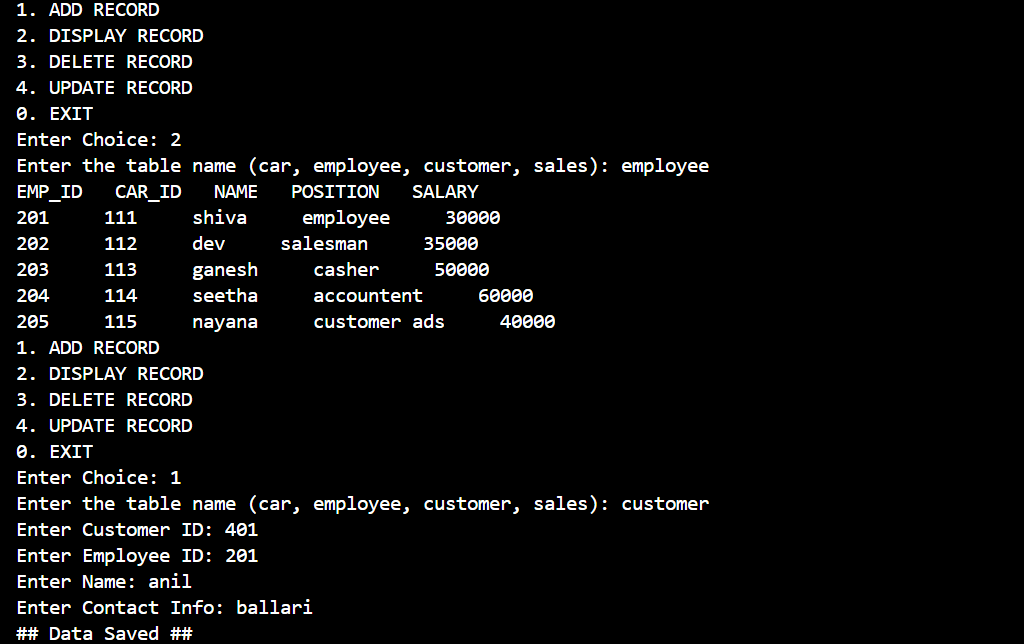


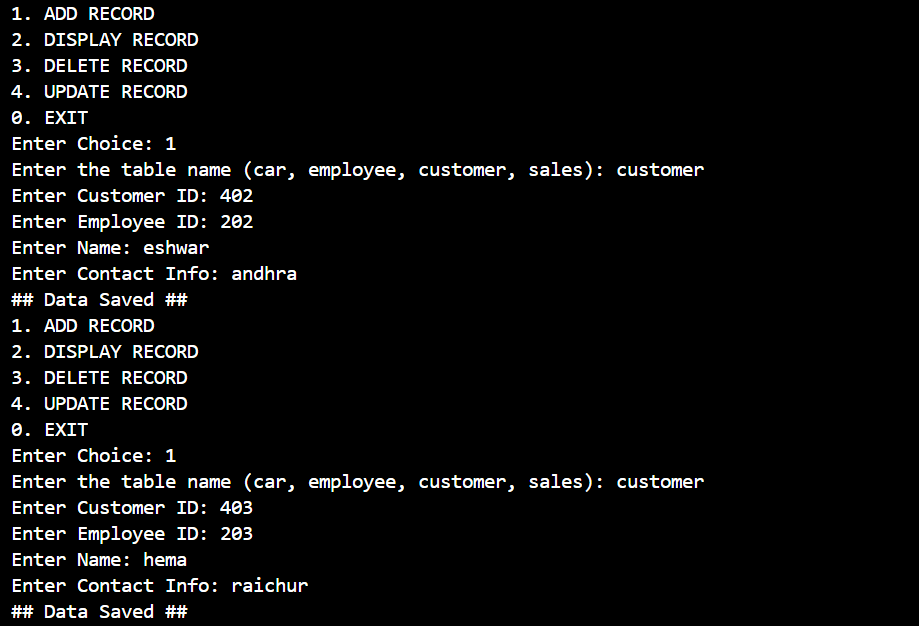


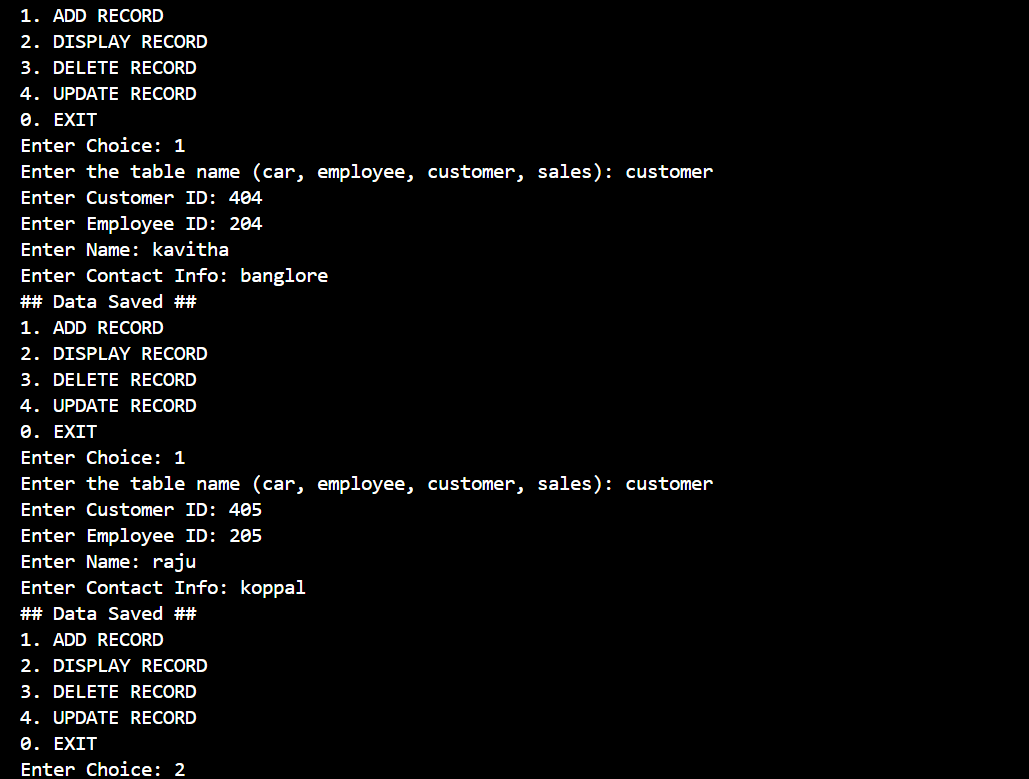


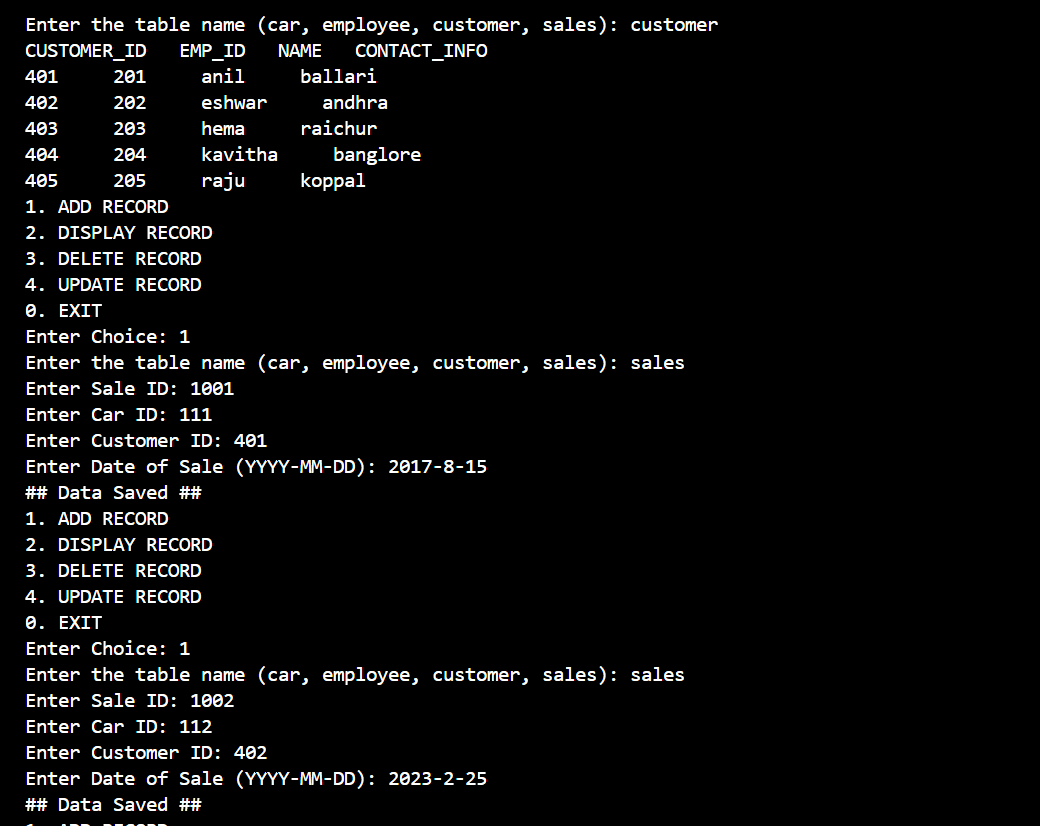


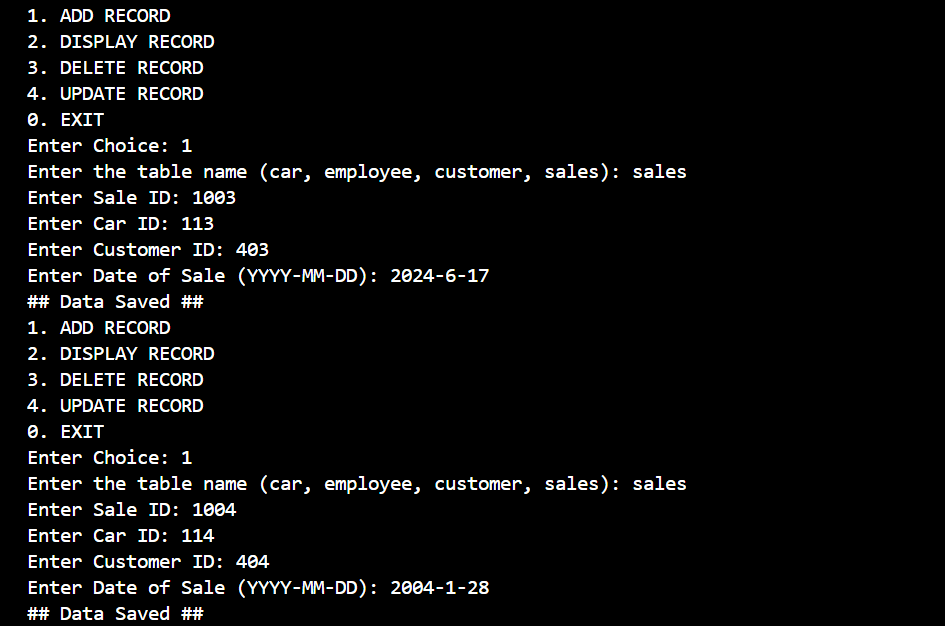


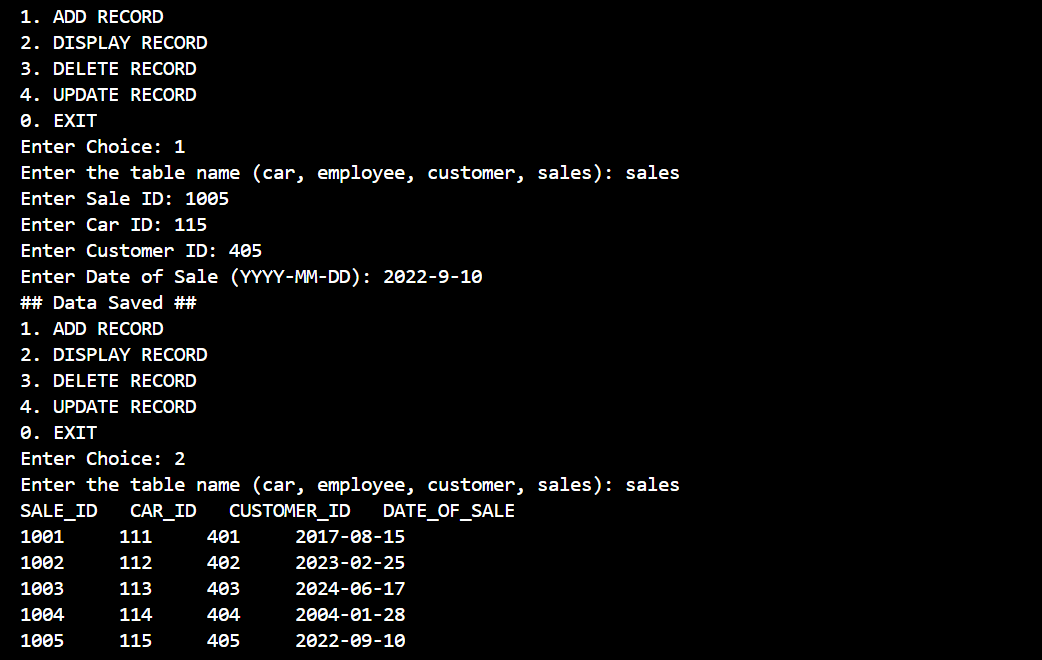


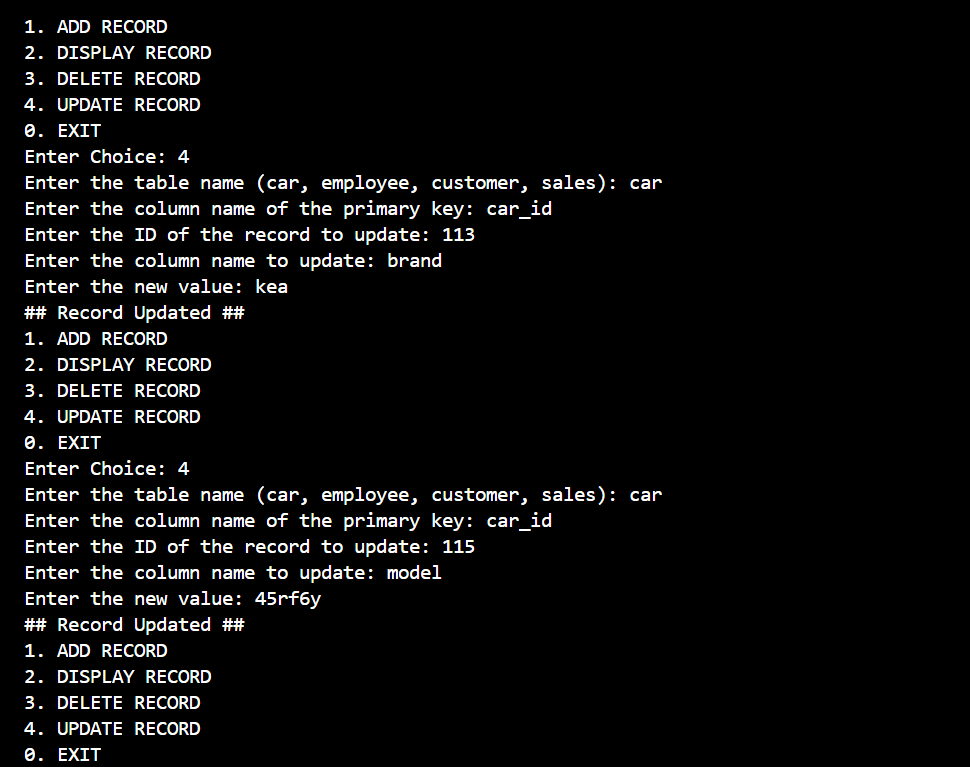


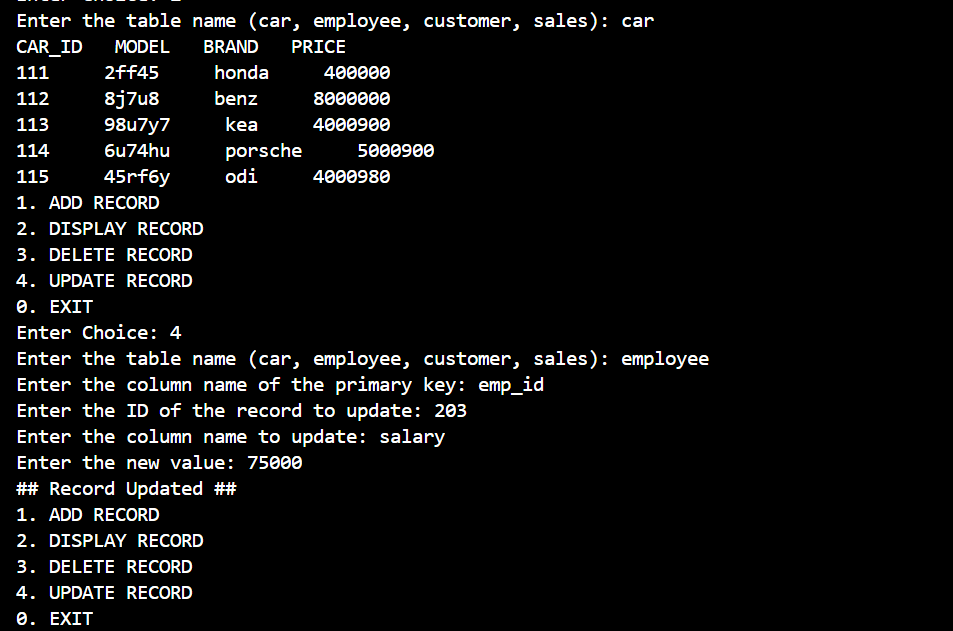


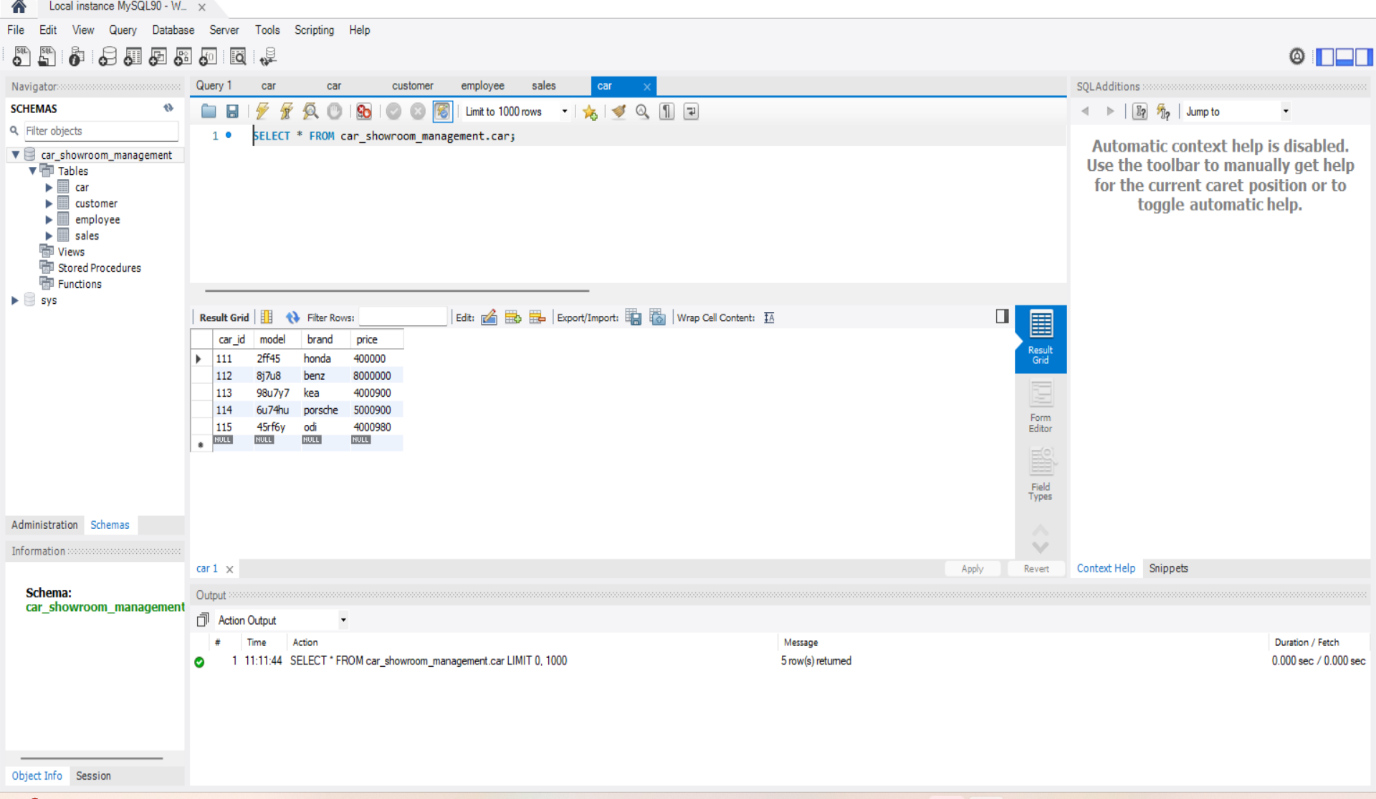


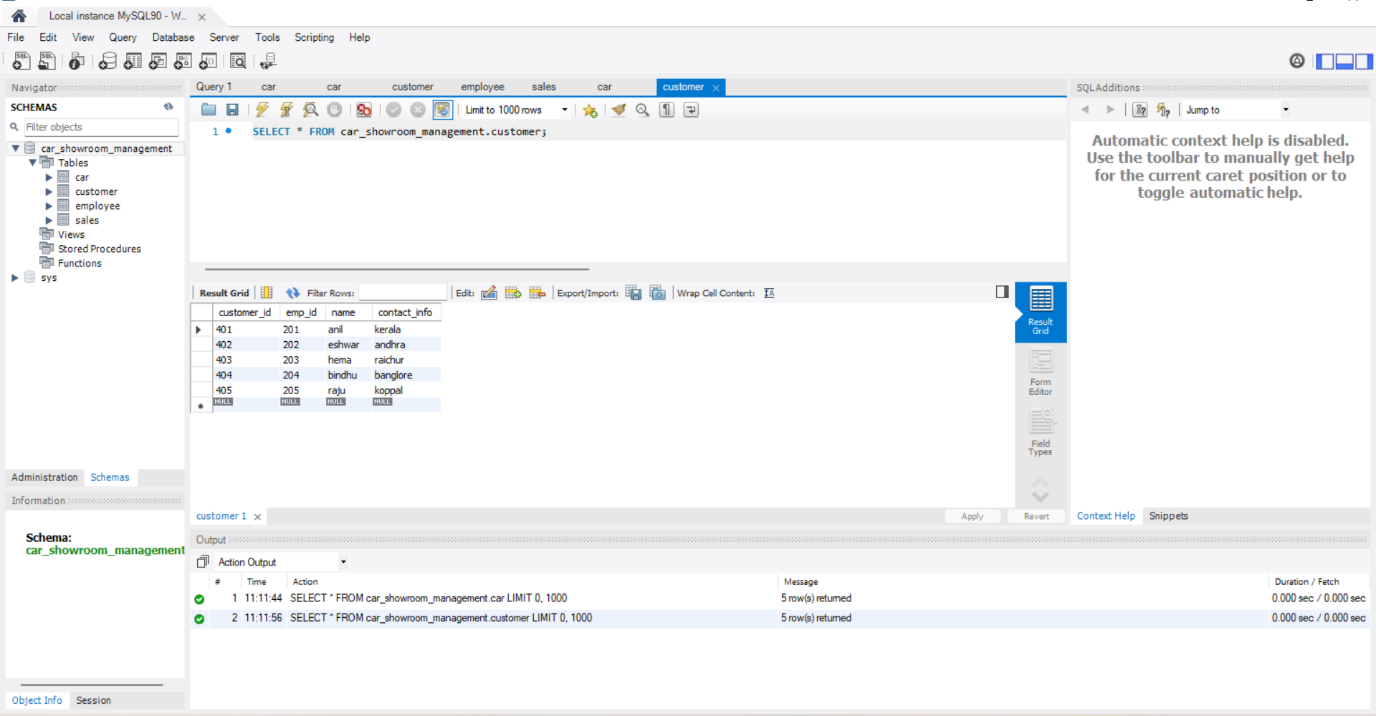


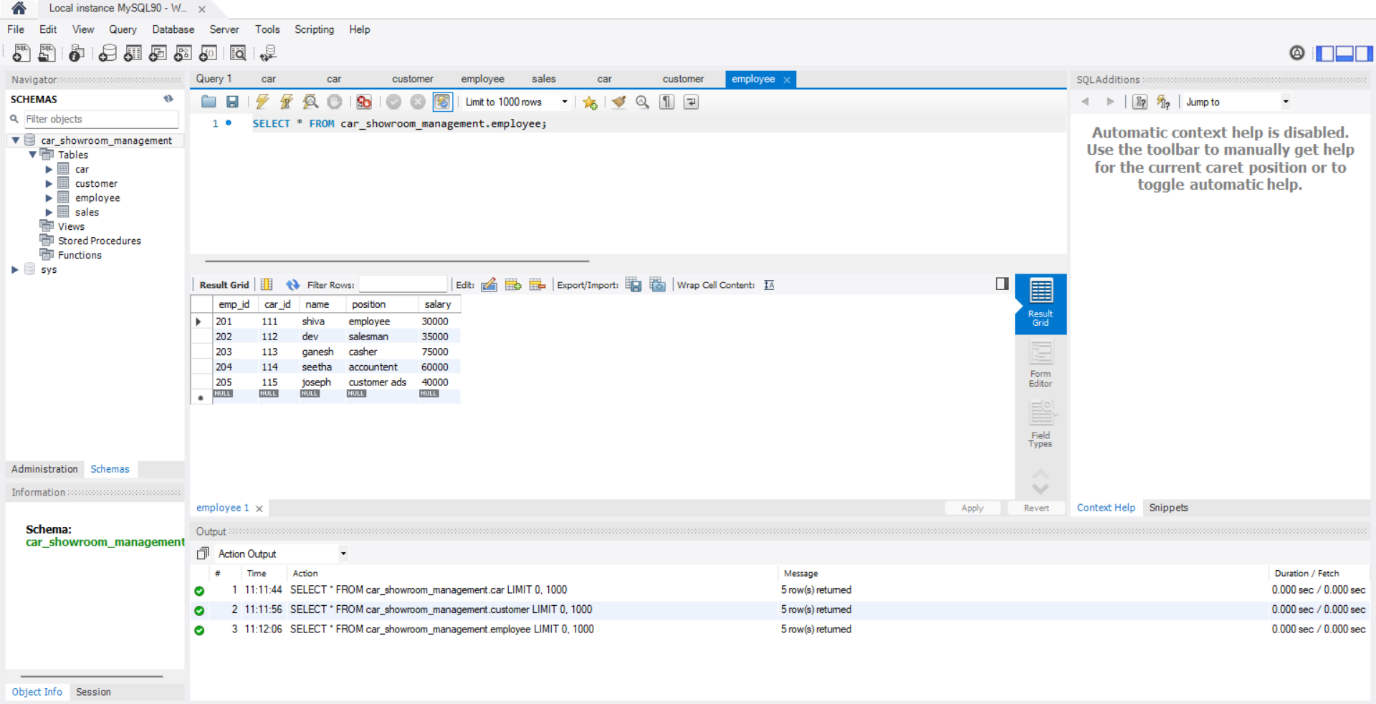


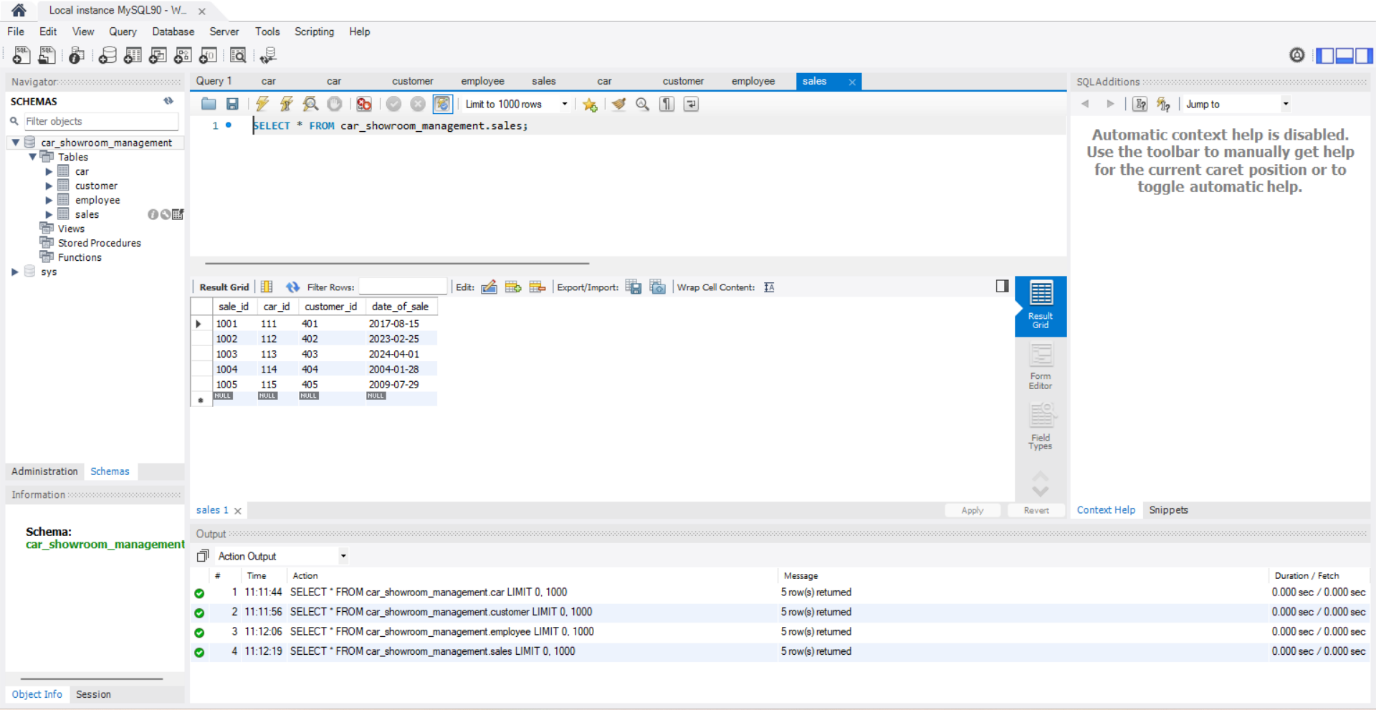


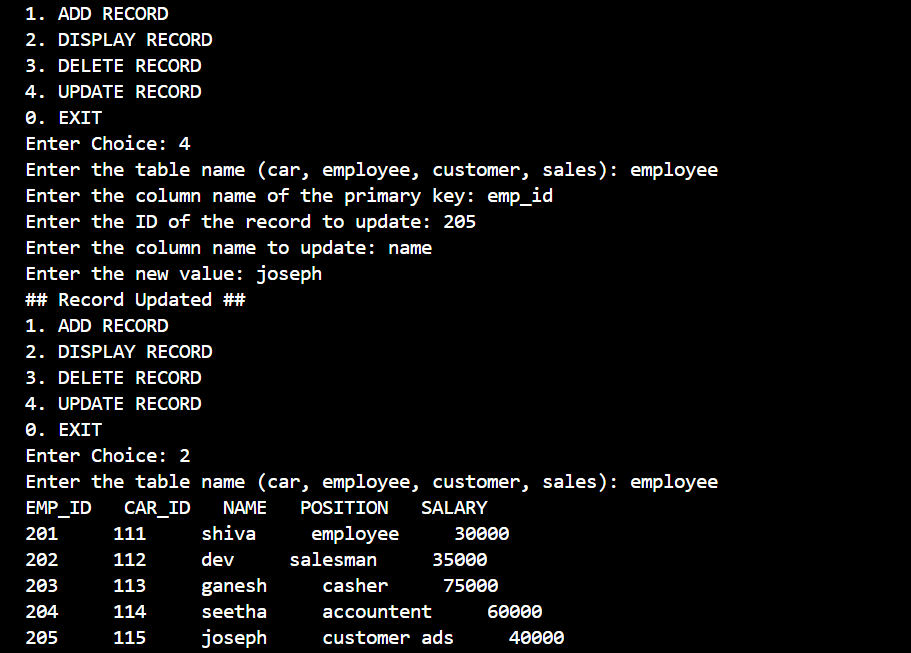


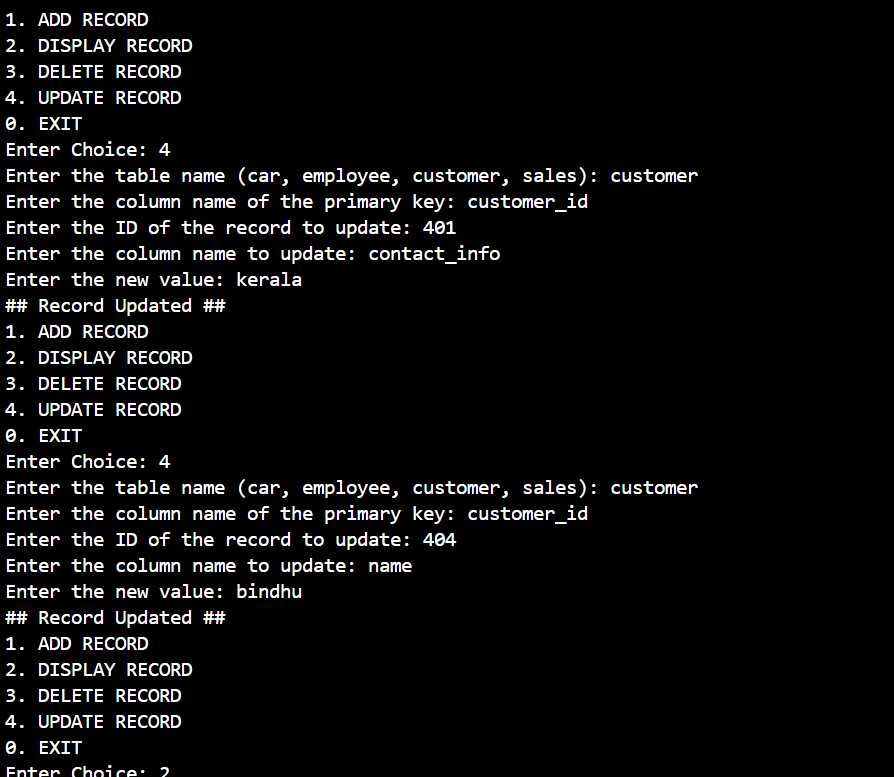


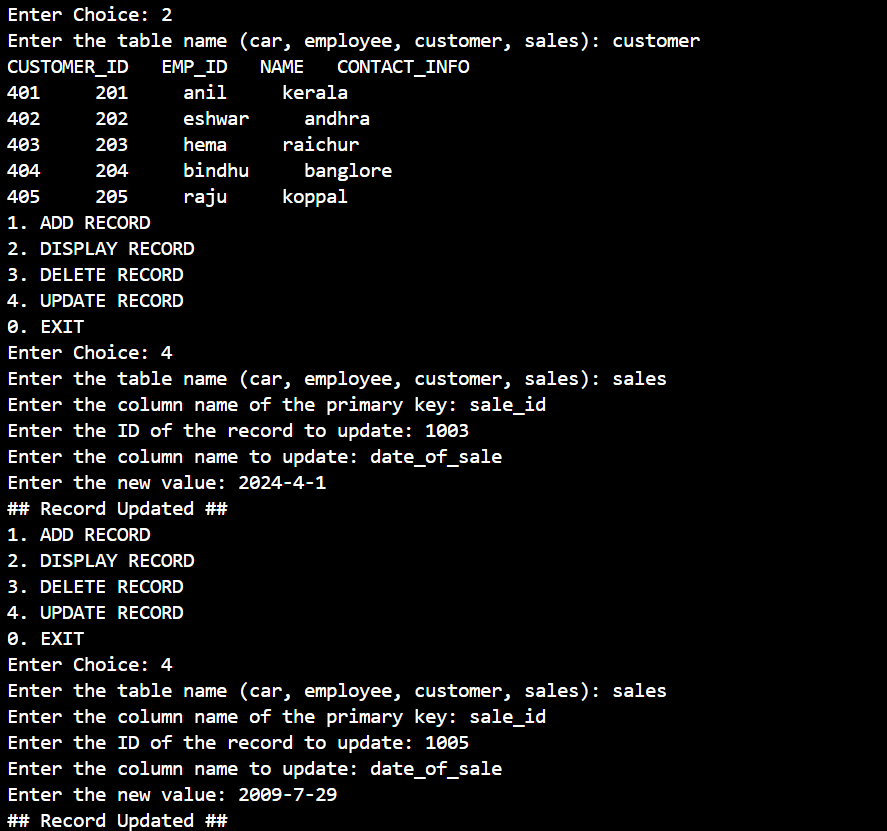


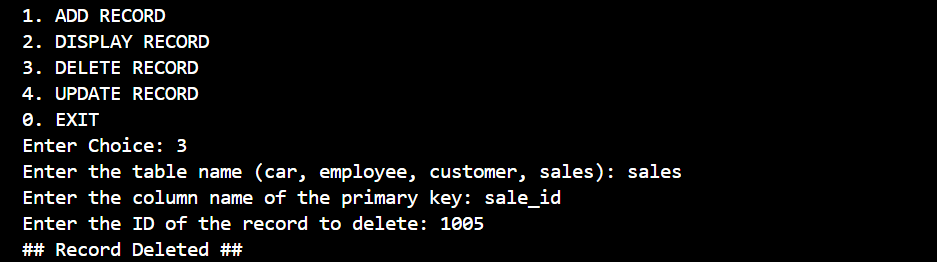


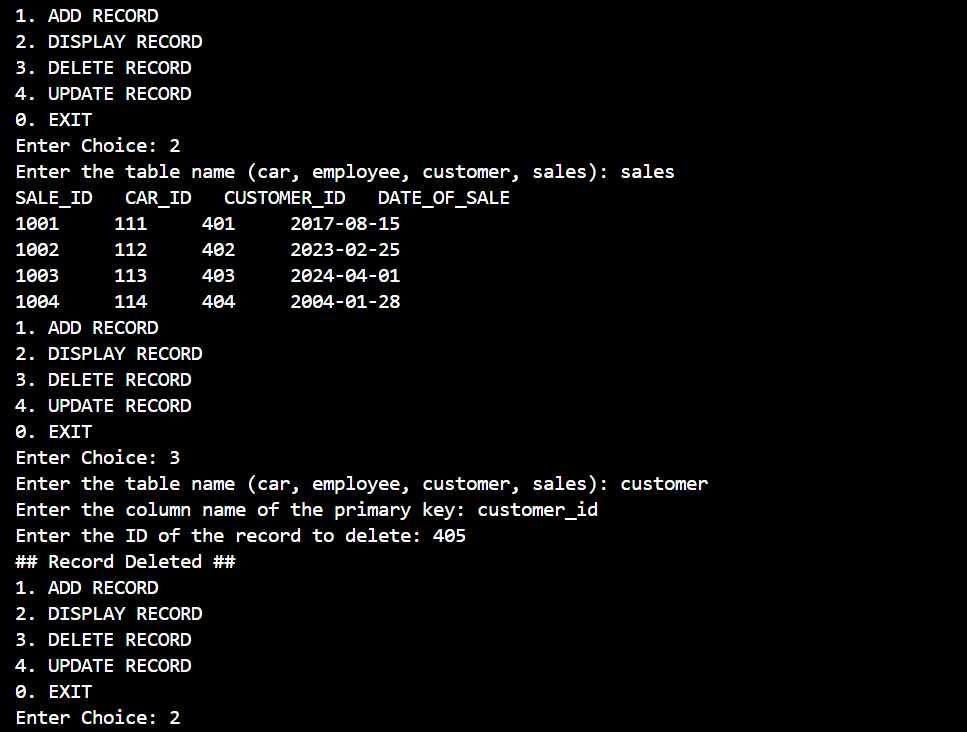


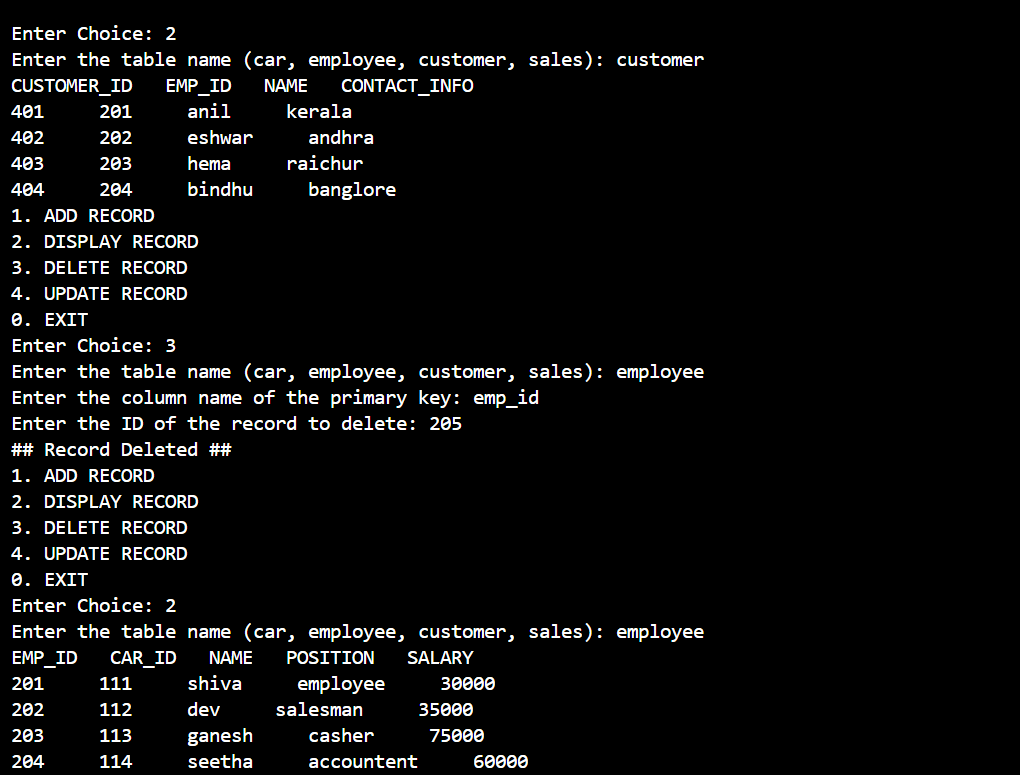


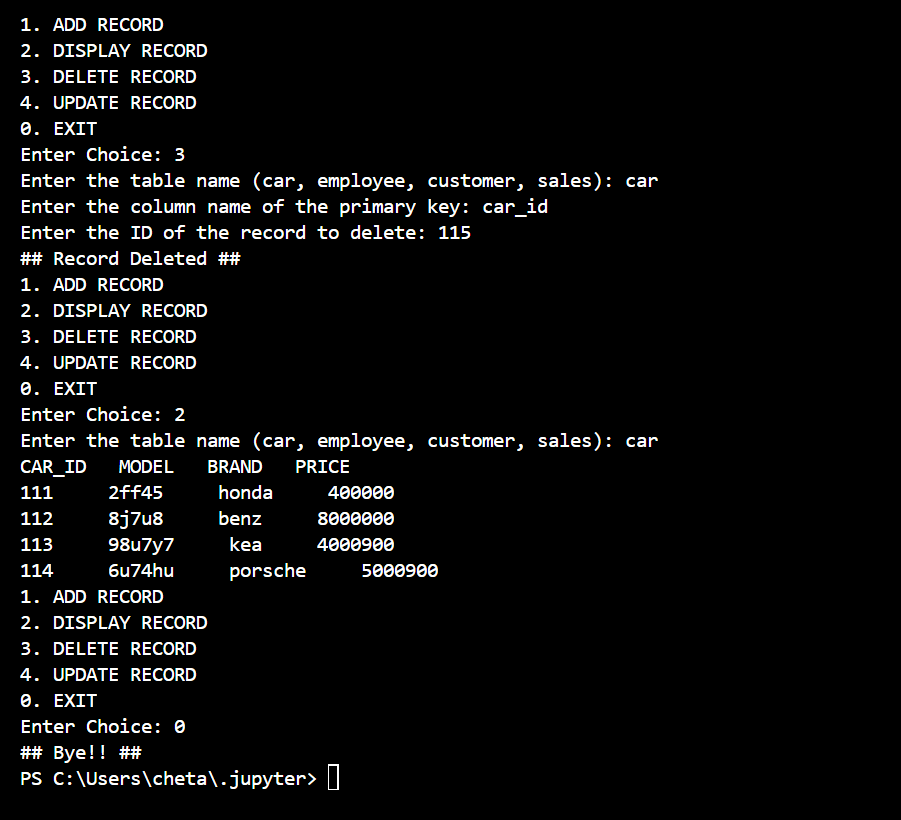


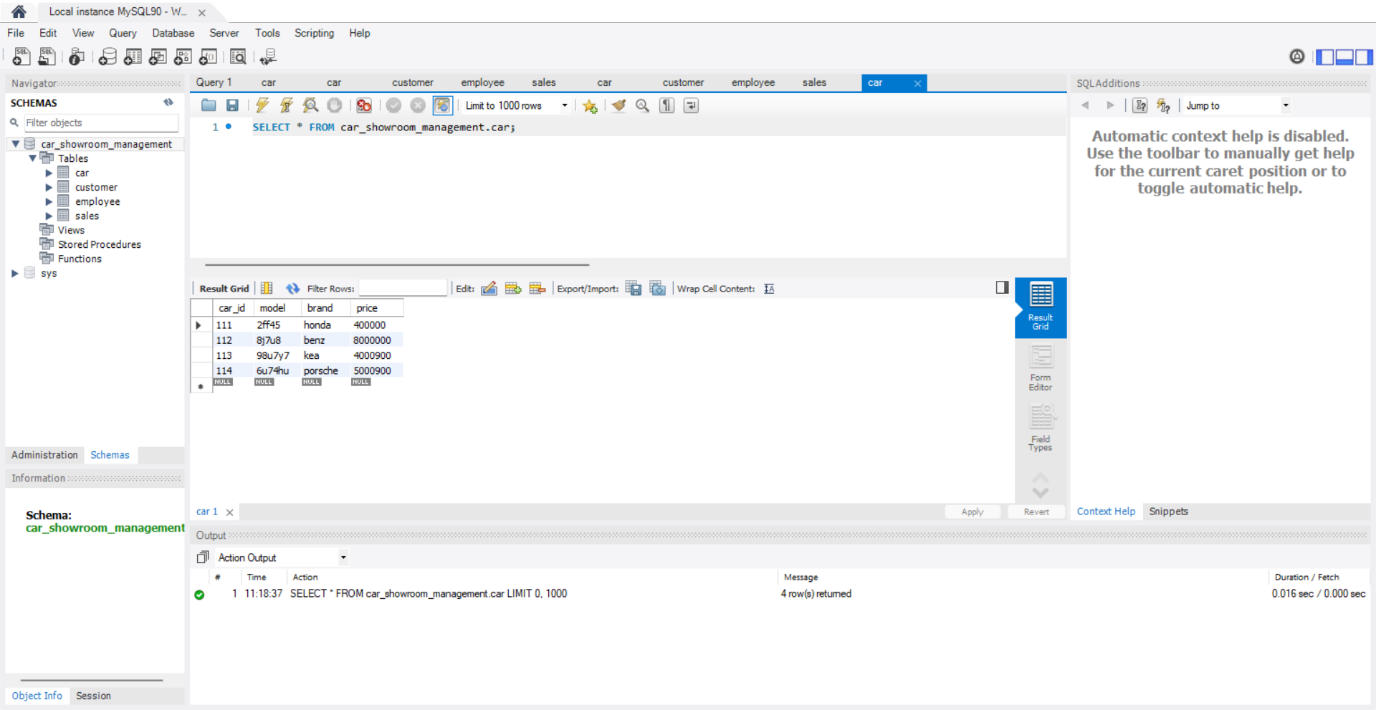


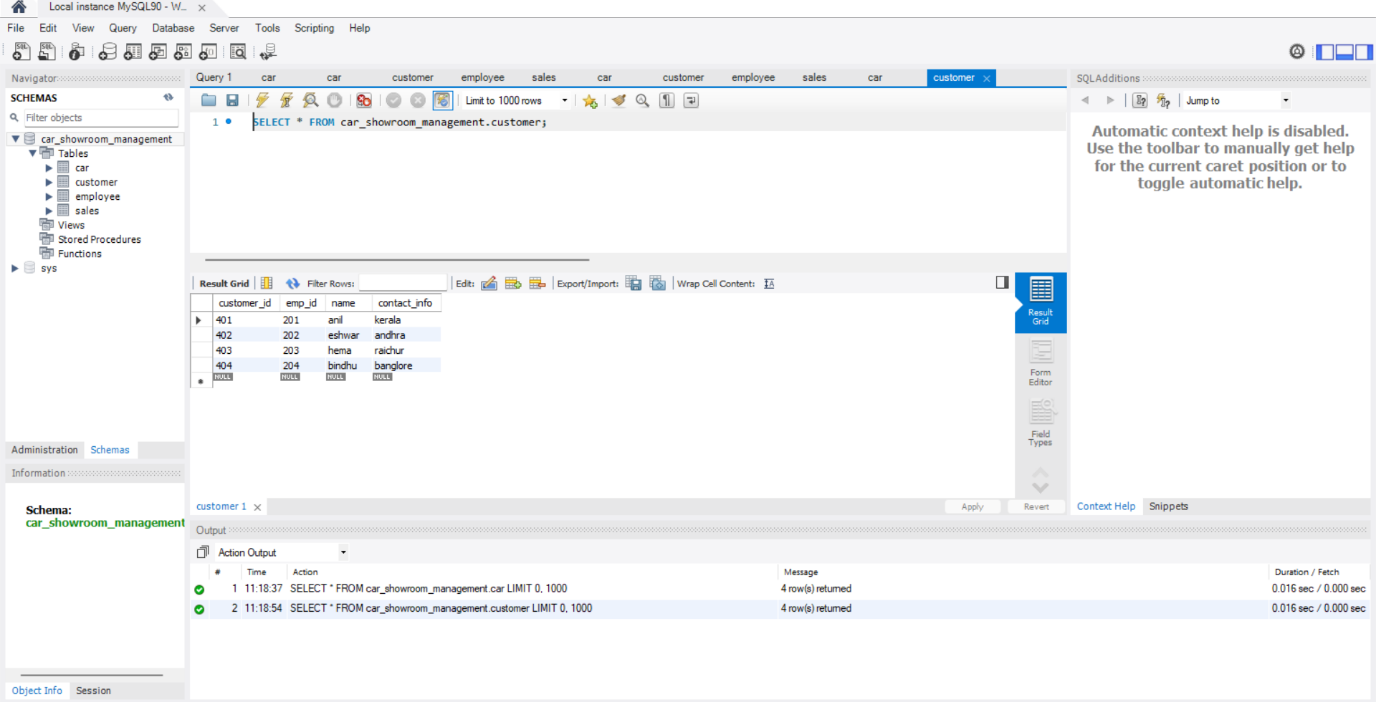


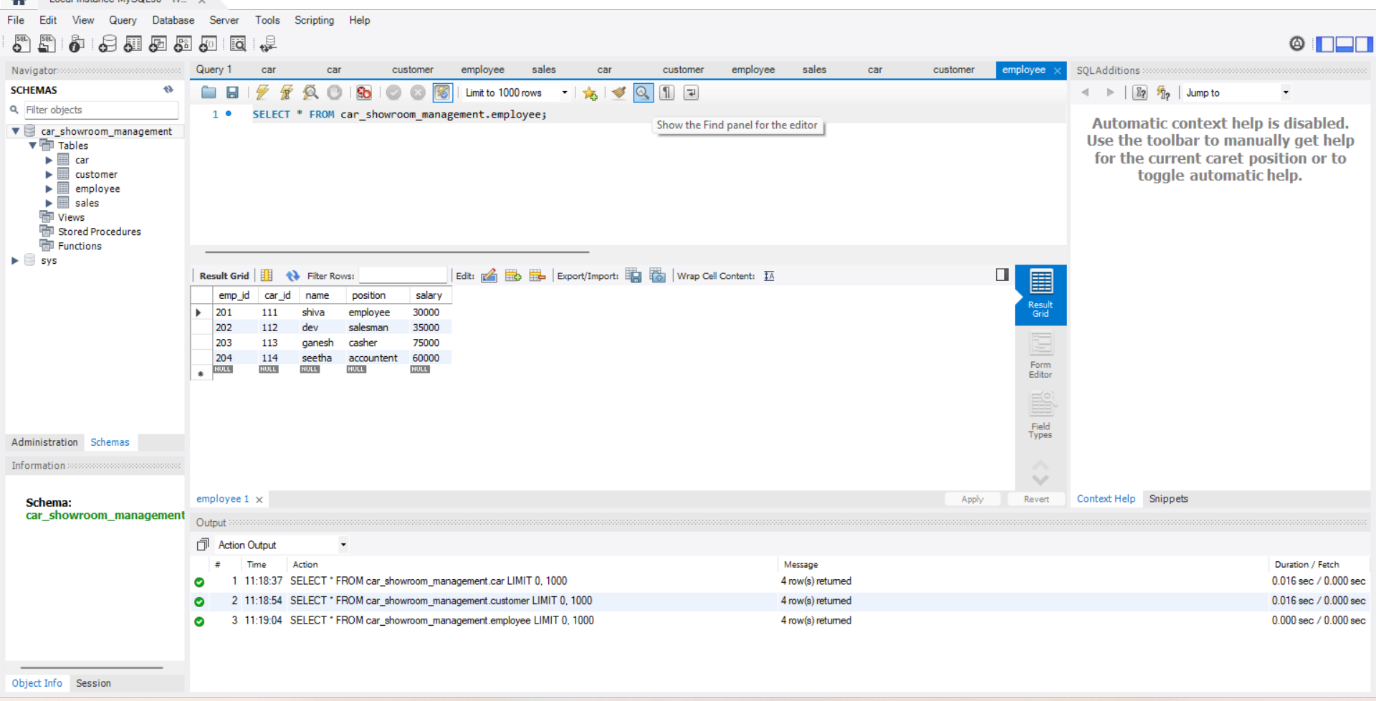


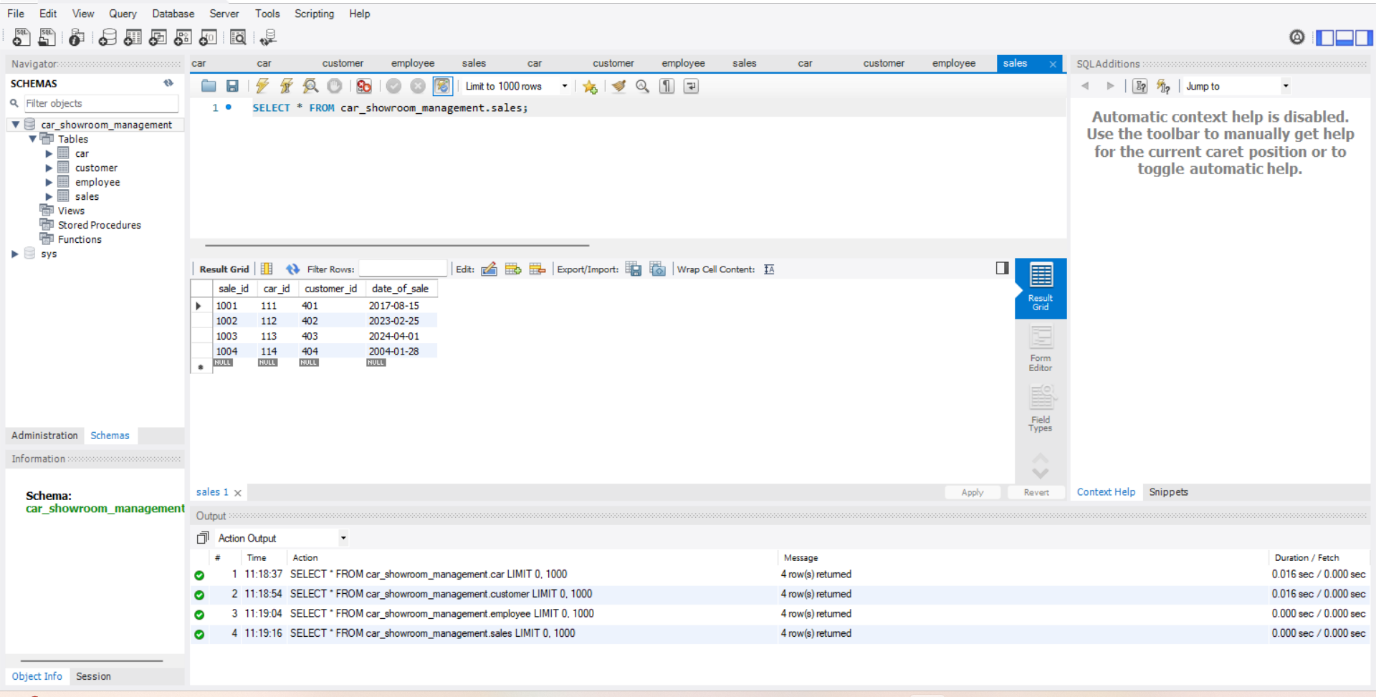












**CONCLUSION**

The Car Showroom Management System project successfully demonstrates how to create a comprehensive database solution for managing various aspects of a car showroom using MySQL. This system includes detailed table structures, effective use of foreign keys for relational integrity, and robust functions for CRUD operations.

The project starts with the creation of a dedicated database named car\_showroom\_management and involves the design of four tables (car, employee, customer, sales). These tables are linked through foreign keys, ensuring that relationships between cars, employees, customers, and sales are consistently maintained, thus preventing orphan records and maintaining referential integrity.

The system offers functionality for adding, displaying, deleting, and updating records, ensuring that all necessary information is captured and properly linked. The user-friendly, menu-driven interface guides users through various operations, making it easy to interact with the database. The system's prompts and feedback ensure that users are informed of successful operations or errors, enhancing the overall user experience.

The modular structure of the code allows for easy updates and scalability, enabling the addition of new tables or features with minimal changes to the existing codebase. Proper indexing and constraints ensure that the system can handle increasing amounts of data without performance degradation.

Looking ahead, there are several potential enhancements that could further improve the system. Implementing user authentication and authorization would secure the system and restrict access based on user roles, such as admin and sales personnel. Adding features for advanced reporting and analytics would help management make informed decisions..

In conclusion, the Car Showroom Management System provides a solid foundation for managing a car showroom's operations efficiently. By leveraging MySQL for database management, the system ensures data consistency, integrity, and ease of use. This project not only meets current needs but also sets the stage for future enhancements and scalability, making it a valuable tool for any car showroom.

**REFERENCES**

* <https://chatgpt.com/c/735bf98e-aec3-4e26-83eb-a2cddffdd082>
* <https://projectworlds.in/free-projects/php-projects/car-selling-and-showroom-management-system-using-php-and-mysql/>
* https://www.freeprojectz.com/python-django-mysql-project-download/car-showroom-management-system.