Project Title: Automobile User Interface

**Index:**

1. Introduction……………………………………………….. 3
2. Components of an Automobile User Interface………….….3
3. Hardware Components
4. Software Components
5. Essential Elements of a car user Interface…………………4
6. Car Inventory Management
7. Salesperson Management
8. Customer Management
9. Sales Record Management
10. User Login System

4.SQL Query……………………………………………………...5

5.Instructions…………………………………………………......22

6. Diagrams …………………………………………………..... 23

7.ER schema………………………………………………………24

8.conclusion…………………………………………………….. 24

9. References…………………………………………………..... 24

**1. Introduction**

This project focuses on the design and implementation of a database management system (DBMS) for MacDonald's Fine Automobiles. The objective is to develop a comprehensive database to manage inventory, salespersons, customers, and sales records. This database will help streamline business operations by providing efficient data storage, retrieval, and manipulation. The project consists of creating an Entity-Relationship diagram, designing the schema, and implementing SQL scripts for database creation, data manipulation, and stored procedures.

**2. Components of the Automobile Database System**

a) Hardware Components

- Server: The database will be hosted on a dedicated SQL Server, which provides the necessary processing power and storage for the database.

- Workstations: Users will access the database through client machines connected to the server.

b) Software Components

- DBMS: Microsoft SQL Server will be used as the DBMS for managing the data.

- Development Tools: SQL Server Management Studio (SSMS) will be utilized for database development, management, and querying.

- Backup and Security Tools: Tools such as SQL Server Backup and native security features will be used to protect the data.

**3. Essential Elements of the Automobile Database System**

a) Car Inventory Management

- Attributes: CarID, Year, Make, Model, Type, Price

- Functionality: This component allows users to view, add, update, and delete car records from the inventory.

b) Salesperson Management

- Attributes: SalesPersonID, FirstName, LastName, Phone, Email, HireDate

- Functionality: This module manages all the details related to salespersons, including their contact information and hiring details.

c) Customer Management

- Attributes: CustomerID, FirstName, LastName, Phone, Email, Address, City, State, Zip

- Functionality: This module manages customer details, allowing users to add, update, or delete customer records.

d) Sales Record Management

- Attributes: SalesID, SalesPersonID, CarID, CustomerID, SalesPrice

- Functionality: This module records sales transactions, linking salespersons, cars, and customers.

e) User Login System

-Attributes: UserID, Username, Password, Email, Role

- Functionality: This module allows access to users, admin to login into portal

**4. SQL Queries**

The following SQL queries are essential for interacting with the database:

- View Records: Queries to display data from the CarInventory, SalesPerson, Customer, and SalesRecord tables.

- Insert Records: Queries to add new entries to each table.

- Update Records: Queries to modify existing entries in each table.

- Delete Records: Queries to remove entries from each table.

Stored procedures and triggers are also implemented to automate specific tasks and ensure data integrity.

**Query:**

USE CarDealershipsDB;

GO

-- Create the UserLogin table

CREATE TABLE UserLogin (

UserID INT IDENTITY(1,1) PRIMARY KEY, -- Primary Key, Auto-incremented

Username NVARCHAR(50) UNIQUE NOT NULL, -- Unique and Required

Password NVARCHAR(255) NOT NULL, -- Required

Email NVARCHAR(100) UNIQUE NOT NULL, -- Unique and Required

Role NVARCHAR(50) -- Optional

);

GO

-- Create the Customer table

CREATE TABLE Customer (

CustomerID INT IDENTITY(1,1) PRIMARY KEY,

FirstName NVARCHAR(50),

LastName NVARCHAR(50),

PhoneNumber NVARCHAR(15),

Email NVARCHAR(100),

Address NVARCHAR(100),

City NVARCHAR(50),

State NVARCHAR(2),

Zip NVARCHAR(10)

);

GO

-- Create the SalesPerson table with UserID as a foreign key

CREATE TABLE SalesPerson (

SalesPersonID INT IDENTITY(1,1) PRIMARY KEY,

FirstName NVARCHAR(50),

LastName NVARCHAR(50),

PhoneNumber NVARCHAR(15),

Email NVARCHAR(100),

HireDate DATE,

UserID INT FOREIGN KEY REFERENCES UserLogin(UserID) -- Foreign Key from UserLogin table

);

GO

-- Create the CarInventory table

CREATE TABLE CarInventory (

CarID INT IDENTITY(1,1) PRIMARY KEY,

Year INT,

Make NVARCHAR(50),

Model NVARCHAR(50),

Type NVARCHAR(50),

Price DECIMAL(10, 2)

);

GO

-- Create the RecordSales table

CREATE TABLE RecordSales (

SaleID INT IDENTITY(1,1) PRIMARY KEY,

SalesPersonID INT FOREIGN KEY REFERENCES SalesPerson(SalesPersonID),

CarID INT FOREIGN KEY REFERENCES CarInventory(CarID),

CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID),

SalesPrice DECIMAL(10, 2)

);

GO

USE CarDealershipsDB;

GO

-- Insert sample data into UserLogin table

INSERT INTO UserLogin (Username, Password, Email, Role)

VALUES

('admin', HASHBYTES('SHA2\_256', 'AdminPassword123'), 'admin@example.com', 'Admin'),

('johndoe', HASHBYTES('SHA2\_256', 'John123!'), 'johndoe@example.com', 'User'),

('janedoe', HASHBYTES('SHA2\_256', 'Jane123!'), 'janedoe@example.com', 'User'),

('samsmith', HASHBYTES('SHA2\_256', 'SamSmith123!'), 'samsmith@example.com', 'User'),

('emilyjones', HASHBYTES('SHA2\_256', 'EmilyJones123!'), 'emilyjones@example.com', 'User'),

('robertbrown', HASHBYTES('SHA2\_256', 'RobertBrown123!'), 'robertbrown@example.com', 'User');

GO

-- Insert sample data into Customer table

INSERT INTO Customer (FirstName, LastName, PhoneNumber, Email, Address, City, State, Zip)

VALUES

('John', 'Doe', '555-1234', 'john.doe@example.com', '123 Maple St', 'Springfield', 'IL', '62701'),

('Jane', 'Smith', '555-5678', 'jane.smith@example.com', '456 Oak St', 'Chicago', 'IL', '60601'),

('Emily', 'Johnson', '555-8765', 'emily.johnson@example.com', '789 Pine St', 'Peoria', 'IL', '61614'),

('Michael', 'Brown', '555-4321', 'michael.brown@example.com', '101 Cedar St', 'Naperville', 'IL', '60540'),

('Sarah', 'Davis', '555-0987', 'sarah.davis@example.com', '202 Birch St', 'Rockford', 'IL', '61101'),

('David', 'Miller', '555-6543', 'david.miller@example.com', '303 Elm St', 'Evanston', 'IL', '60201');

GO

-- Insert sample data into SalesPerson table

INSERT INTO SalesPerson (FirstName, LastName, PhoneNumber, Email, HireDate, UserID)

VALUES

('Alice', 'Green', '555-1122', 'alice.green@example.com', '2023-01-15', 2), -- Assign UserID

('Bob', 'White', '555-2233', 'bob.white@example.com', '2022-06-21', 3), -- Assign UserID

('Charlie', 'Black', '555-3344', 'charlie.black@example.com', '2021-09-10', 4), -- Assign UserID

('Diana', 'Blue', '555-4455', 'diana.blue@example.com', '2020-03-18', 5), -- Assign UserID

('Edward', 'Red', '555-5566', 'edward.red@example.com', '2019-11-25', 6), -- Assign UserID

('Fiona', 'Yellow', '555-6677', 'fiona.yellow@example.com', '2018-07-30', 1); -- Assign UserID

GO

-- Insert sample data into CarInventory table

INSERT INTO CarInventory (Year, Make, Model, Type, Price)

VALUES

(2022, 'Toyota', 'Camry', 'Sedan', 25000.00),

(2021, 'Honda', 'Civic', 'Sedan', 22000.00),

(2023, 'Ford', 'F-150', 'Truck', 35000.00),

(2020, 'Chevrolet', 'Malibu', 'Sedan', 20000.00),

(2019, 'Nissan', 'Rogue', 'SUV', 24000.00),

(2022, 'Tesla', 'Model 3', 'Sedan', 40000.00);

GO

-- Insert sample data into RecordSales table

INSERT INTO RecordSales (SalesPersonID, CarID, CustomerID, SalesPrice)

VALUES

(1, 1, 1, 24000.00),

(2, 2, 2, 21000.00),

(3, 3, 3, 36000.00),

(4, 4, 4, 19000.00),

(5, 5, 5, 26000.00),

(6, 6, 6, 39000.00);

GO

USE CarDealershipsDB;

GO

-- =============================================

-- Routine: ViewUserLogin

-- Description: Retrieves all records from the UserLogin table.

-- =============================================

CREATE PROCEDURE ViewUserLogin

AS

BEGIN

SELECT \* FROM UserLogin;

END;

GO

-- =============================================

-- Routine: AddUserLogin

-- Description: Inserts a new record into the UserLogin table.

-- =============================================

CREATE PROCEDURE AddUserLogin

@Username NVARCHAR(50),

@Password NVARCHAR(255), -- Should be hashed before calling this procedure

@Email NVARCHAR(100),

@Role NVARCHAR(50)

AS

BEGIN

-- Insert a new record with hashed password

INSERT INTO UserLogin (Username, Password, Email, Role)

VALUES (@Username, @Password, @Email, @Role);

END;

GO

-- =============================================

-- Routine: UpdateUserLogin

-- Description: Updates an existing record in the UserLogin table.

-- =============================================

CREATE PROCEDURE UpdateUserLogin

@UserID INT,

@Username NVARCHAR(50),

@Password NVARCHAR(255), -- Should be hashed before calling this procedure

@Email NVARCHAR(100),

@Role NVARCHAR(50)

AS

BEGIN

-- Update an existing record with hashed password

UPDATE UserLogin

SET Username = @Username,

Password = @Password,

Email = @Email,

Role = @Role

WHERE UserID = @UserID;

END;

GO

-- =============================================

-- Routine: DeleteUserLogin

-- Description: Deletes a record from the UserLogin table.

-- =============================================

CREATE PROCEDURE DeleteUserLogin

@UserID INT

AS

BEGIN

-- Delete a record

DELETE FROM UserLogin

WHERE UserID = @UserID;

END;

GO

USE CarDealershipsDB;

GO

-- =============================================

-- Routine: ViewCustomers

-- Description: Retrieves all records from the Customer table.

-- =============================================

CREATE PROCEDURE ViewCustomers

AS

BEGIN

SELECT \* FROM Customer;

END;

GO

-- =============================================

-- Routine: AddCustomer

-- Description: Inserts a new record into the Customer table.

-- =============================================

CREATE PROCEDURE AddCustomer

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@PhoneNumber NVARCHAR(15),

@Email NVARCHAR(100),

@Address NVARCHAR(100),

@City NVARCHAR(50),

@State NVARCHAR(2),

@Zip NVARCHAR(10)

AS

BEGIN

INSERT INTO Customer (FirstName, LastName, PhoneNumber, Email, Address, City, State, Zip)

VALUES (@FirstName, @LastName, @PhoneNumber, @Email, @Address, @City, @State, @Zip);

END;

GO

-- =============================================

-- Routine: UpdateCustomer

-- Description: Updates an existing record in the Customer table.

-- =============================================

CREATE PROCEDURE UpdateCustomer

@CustomerID INT,

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@PhoneNumber NVARCHAR(15),

@Email NVARCHAR(100),

@Address NVARCHAR(100),

@City NVARCHAR(50),

@State NVARCHAR(2),

@Zip NVARCHAR(10)

AS

BEGIN

UPDATE Customer

SET FirstName = @FirstName,

LastName = @LastName,

PhoneNumber = @PhoneNumber,

Email = @Email,

Address = @Address,

City = @City,

State = @State,

Zip = @Zip

WHERE CustomerID = @CustomerID;

END;

GO

-- =============================================

-- Routine: DeleteCustomer

-- Description: Deletes a record from the Customer table.

-- =============================================

CREATE PROCEDURE DeleteCustomer

@CustomerID INT

AS

BEGIN

DELETE FROM Customer

WHERE CustomerID = @CustomerID;

END;

GO

USE CarDealershipsDB;

GO

-- =============================================

-- Routine: ViewCarInventory

-- Description: Retrieves all records from the CarInventory table.

-- =============================================

CREATE PROCEDURE ViewCarInventory

AS

BEGIN

SELECT \* FROM CarInventory;

END;

GO

-- =============================================

-- Routine: AddCarInventory

-- Description: Inserts a new record into the CarInventory table.

-- =============================================

CREATE PROCEDURE AddCarInventory

@Year INT,

@Make NVARCHAR(50),

@Model NVARCHAR(50),

@Type NVARCHAR(50),

@Price DECIMAL(10, 2)

AS

BEGIN

INSERT INTO CarInventory (Year, Make, Model, Type, Price)

VALUES (@Year, @Make, @Model, @Type, @Price);

END;

GO

-- =============================================

-- Routine: UpdateCarInventory

-- Description: Updates an existing record in the CarInventory table.

-- =============================================

CREATE PROCEDURE UpdateCarInventory

@CarID INT,

@Year INT,

@Make NVARCHAR(50),

@Model NVARCHAR(50),

@Type NVARCHAR(50),

@Price DECIMAL(10, 2)

AS

BEGIN

UPDATE CarInventory

SET Year = @Year,

Make = @Make,

Model = @Model,

Type = @Type,

Price = @Price

WHERE CarID = @CarID;

END;

GO

-- =============================================

-- Routine: DeleteCarInventory

-- Description: Deletes a record from the CarInventory table.

-- =============================================

CREATE PROCEDURE DeleteCarInventory

@CarID INT

AS

BEGIN

DELETE FROM CarInventory

WHERE CarID = @CarID;

END;

GO

USE CarDealershipsDB;

GO

-- =============================================

-- Routine: ViewSalesPersons

-- Description: Retrieves all records from the SalesPerson table.

-- =============================================

CREATE PROCEDURE ViewSalesPersons

AS

BEGIN

SELECT \* FROM SalesPerson;

END;

GO

-- =============================================

-- Routine: AddSalesPerson

-- Description: Inserts a new record into the SalesPerson table.

-- =============================================

CREATE PROCEDURE AddSalesPerson

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@PhoneNumber NVARCHAR(15),

@Email NVARCHAR(100),

@HireDate DATE,

@UserID INT -- New parameter for UserID

AS

BEGIN

INSERT INTO SalesPerson (FirstName, LastName, PhoneNumber, Email, HireDate, UserID)

VALUES (@FirstName, @LastName, @PhoneNumber, @Email, @HireDate, @UserID);

END;

GO

-- =============================================

-- Routine: UpdateSalesPerson

-- Description: Updates an existing record in the SalesPerson table.

-- =============================================

CREATE PROCEDURE UpdateSalesPerson

@SalesPersonID INT,

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@PhoneNumber NVARCHAR(15),

@Email NVARCHAR(100),

@HireDate DATE,

@UserID INT -- New parameter for UserID

AS

BEGIN

UPDATE SalesPerson

SET FirstName = @FirstName,

LastName = @LastName,

PhoneNumber = @PhoneNumber,

Email = @Email,

HireDate = @HireDate,

UserID = @UserID

WHERE SalesPersonID = @SalesPersonID;

END;

GO

-- =============================================

-- Routine: DeleteSalesPerson

-- Description: Deletes a record from the SalesPerson table.

-- =============================================

CREATE PROCEDURE DeleteSalesPerson

@SalesPersonID INT

AS

BEGIN

DELETE FROM SalesPerson

WHERE SalesPersonID = @SalesPersonID;

END;

GO

----Record Sales

USE CarDealershipsDB;

GO

-- =============================================

-- Routine: ViewRecordSales

-- Description: Retrieves all records from the RecordSales table.

-- =============================================

CREATE PROCEDURE ViewRecordSales

AS

BEGIN

SELECT \* FROM RecordSales;

END;

GO

-- =============================================

-- Routine: AddRecordSales

-- Description: Inserts a new record into the RecordSales table.

-- =============================================

CREATE PROCEDURE AddRecordSales

@SalesPersonID INT,

@CarID INT,

@CustomerID INT,

@SalesPrice DECIMAL(10, 2)

AS

BEGIN

INSERT INTO RecordSales (SalesPersonID, CarID, CustomerID, SalesPrice)

VALUES (@SalesPersonID, @CarID, @CustomerID, @SalesPrice);

END;

GO

-- =============================================

-- Routine: UpdateRecordSales

-- Description: Updates an existing record in the RecordSales table.

-- =============================================

CREATE PROCEDURE UpdateRecordSales

@SaleID INT,

@SalesPersonID INT,

@CarID INT,

@CustomerID INT,

@SalesPrice DECIMAL(10, 2)

AS

BEGIN

UPDATE RecordSales

SET SalesPersonID = @SalesPersonID,

CarID = @CarID,

CustomerID = @CustomerID,

SalesPrice = @SalesPrice

WHERE SaleID = @SaleID;

END;

GO

-- =============================================

-- Routine: DeleteRecordSales

-- Description: Deletes a record from the RecordSales table.

-- =============================================

CREATE PROCEDURE DeleteRecordSales

@SaleID INT

AS

BEGIN

DELETE FROM RecordSales

WHERE SaleID = @SaleID;

END;

GO

USE CarDealershipsDB;

GO

-- =============================================

-- 1NF: Check for Atomic Values and Primary Keys

-- =============================================

-- CarInventory table: Ensure CarID is the primary key

SELECT CarID FROM CarInventory;

-- SalesPerson table: Ensure SalesPersonID is the primary key

SELECT SalesPersonID FROM SalesPerson;

-- Customer table: Ensure CustomerID is the primary key

SELECT CustomerID FROM Customer;

-- RecordSales table: Ensure SaleID is the primary key

SELECT SaleID FROM RecordSales;

-- =============================================

-- 2NF: Check for Partial Dependencies

-- =============================================

-- CarInventory table: Check all attributes depend on CarID

SELECT CarID, Year, Make, Model, Type, Price FROM CarInventory;

-- SalesPerson table: Check all attributes depend on SalesPersonID

SELECT SalesPersonID, FirstName, LastName, PhoneNumber, Email, HireDate FROM SalesPerson;

-- Customer table: Check all attributes depend on CustomerID

SELECT CustomerID, FirstName, LastName, PhoneNumber, Email, Address, City, State, Zip FROM Customer;

-- RecordSales table: Check all attributes depend on SaleID

SELECT SaleID, SalesPersonID, CarID, CustomerID, SalesPrice FROM RecordSales;

-- =============================================

-- 3NF: Check for Transitive Dependencies

-- =============================================

-- CarInventory table: Ensure no transitive dependencies

SELECT CarID, Year, Make, Model, Type, Price FROM CarInventory;

-- SalesPerson table: Ensure no transitive dependencies

SELECT SalesPersonID, FirstName, LastName, PhoneNumber, Email, HireDate FROM SalesPerson;

-- Customer table: Ensure no transitive dependencies

SELECT CustomerID, FirstName, LastName, PhoneNumber, Email, Address, City, State, Zip FROM Customer;

-- RecordSales table: Ensure no transitive dependencies

SELECT SaleID, SalesPersonID, CarID, CustomerID, SalesPrice FROM RecordSales;

-- =============================================

-- Functional Dependencies Verification

-- =============================================

-- CarInventory: CarID → Year, Make, Model, Type, Price

-- SalesPerson: SalesPersonID → FirstName, LastName, PhoneNumber, Email, HireDate

-- Customer: CustomerID → FirstName, LastName, PhoneNumber, Email, Address, City, State, Zip

-- RecordSales: SaleID → SalesPersonID, CarID, CustomerID, SalesPrice

-- Example verification for Customer table

SELECT CustomerID, City, State, Zip

FROM Customer

WHERE City IS NOT NULL

AND State IS NOT NULL

AND Zip IS NOT NULL;

-- =============================================

-- Foreign Key Constraint Verification

-- =============================================

-- Ensure foreign key constraints are in place for SalesPerson table

ALTER TABLE SalesPerson

ADD CONSTRAINT FK\_SalesPerson\_UserLogin

FOREIGN KEY (SalesPersonID) REFERENCES UserLogin(UserID);

GO

-- Ensure foreign key constraints are in place for RecordSales table

ALTER TABLE RecordSales

ADD CONSTRAINT FK\_RecordSales\_SalesPerson

FOREIGN KEY (SalesPersonID) REFERENCES SalesPerson(SalesPersonID);

ALTER TABLE RecordSales

ADD CONSTRAINT FK\_RecordSales\_Car

FOREIGN KEY (CarID) REFERENCES CarInventory(CarID);

ALTER TABLE RecordSales

ADD CONSTRAINT FK\_RecordSales\_Customer

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID);

GO

-------Testing Script---

USE CarDealershipDB;

GO

-- Adding 6 customers to the Customer table

EXEC AddCustomer 'John', 'Doe', '123-456-7890', 'john.doe@example.com', '123 Elm St', 'New York', 'NY', '10001';

EXEC AddCustomer 'Jane', 'Smith', '987-654-3210', 'jane.smith@example.com', '456 Oak St', 'Los Angeles', 'CA', '90001';

EXEC AddCustomer 'Alice', 'Johnson', '555-123-4567', 'alice.johnson@example.com', '789 Pine St', 'Chicago', 'IL', '60601';

EXEC AddCustomer 'Bob', 'Brown', '444-321-6789', 'bob.brown@example.com', '101 Maple St', 'Houston', 'TX', '77001';

EXEC AddCustomer 'Eve', 'Davis', '333-654-9870', 'eve.davis@example.com', '202 Birch St', 'Miami', 'FL', '33101';

EXEC AddCustomer 'Charlie', 'Miller', '222-789-0123', 'charlie.miller@example.com', '303 Cedar St', 'Seattle', 'WA', '98101';

-- Viewing all customers in the Customer table

EXEC ViewCustomers;

-- Updating a customer's information

EXEC UpdateCustomer 1, 'John', 'Doe', '111-222-3333', 'john.new@example.com', '123 Elm St', 'New York', 'NY', '10001';

-- Deleting a customer from the Customer table

EXEC DeleteCustomer 6;

USE CarDealershipsDB;

GO

-- =============================================

-- Test Script for SalesPerson

-- =============================================

-- Viewing all salespersons in the SalesPerson table

EXEC ViewSalesPersons;

-- Updating a salesperson's information

EXEC UpdateSalesPerson 1, 'Michael', 'Johnson', '555-123-4567', 'michael.johnson@newemail.com', '2023-01-15', 1;

-- Deleting a salesperson from the SalesPerson table

EXEC DeleteSalesPerson 6;

-- Viewing all salespersons again to confirm changes

EXEC ViewSalesPersons;

-- Adding 6 cars to the CarInventory table

EXEC AddCarInventory 2020, 'Toyota', 'Camry', 'Sedan', 24000.00;

EXEC AddCarInventory 2019, 'Honda', 'Civic', 'Sedan', 22000.00;

EXEC AddCarInventory 2021, 'Ford', 'F-150', 'Truck', 35000.00;

EXEC AddCarInventory 2018, 'Chevrolet', 'Malibu', 'Sedan', 20000.00;

EXEC AddCarInventory 2022, 'Tesla', 'Model 3', 'Electric', 45000.00;

EXEC AddCarInventory 2021, 'BMW', 'X5', 'SUV', 60000.00;

-- Viewing all cars in the CarInventory table

EXEC ViewCarInventory;

-- Updating a car's information

EXEC UpdateCarInventory 1, 2020, 'Toyota', 'Camry', 'Hybrid', 26000.00;

-- Adding 6 sales records to the RecordSales table

EXEC AddRecordSales 1, 1, 1, 24000.00;

EXEC AddRecordSales 2, 2, 2, 22000.00;

EXEC AddRecordSales 3, 3, 3, 35000.00;

EXEC AddRecordSales 4, 4, 4, 20000.00;

EXEC AddRecordSales 5, 5, 5, 45000.00;

EXEC AddRecordSales 1, 6, 1, 60000.00;

-- Viewing all sales records in the RecordSales table

EXEC ViewRecordSales;

-- Updating a sales record

EXEC UpdateRecordSales 1, 1, 1, 1, 25000.00;

-- Deleting a sales record from the RecordSales table

EXEC DeleteRecordSales 6;

1. **Instructions:**

1. Database Setup:

- Install Microsoft SQL Server and SQL Server Management Studio (SSMS).

- Execute the provided SQL scripts to create the database and tables.

2. Running Stored Procedures/Routines:

- Use SSMS to execute stored procedures for adding, updating, deleting, and viewing records.

- Test each procedure with sample data to ensure functionality.

3. Backup and Security:

- Set up automated backups using SQL Server tools.

- Implement user access control to protect sensitive data.

The Automobile database and all required tables will be created by this script, along with sample data to populate them. This scheme can be further expanded and altered to meet more precise needs.

**5. Explanation**

The database schema is designed with normalization in mind, reducing redundancy and ensuring data integrity. The Entity-Relationship diagram outlines the relationships between tables, with primary and foreign keys defined to enforce referential integrity. The tables are structured to minimize duplication and optimize query performance.

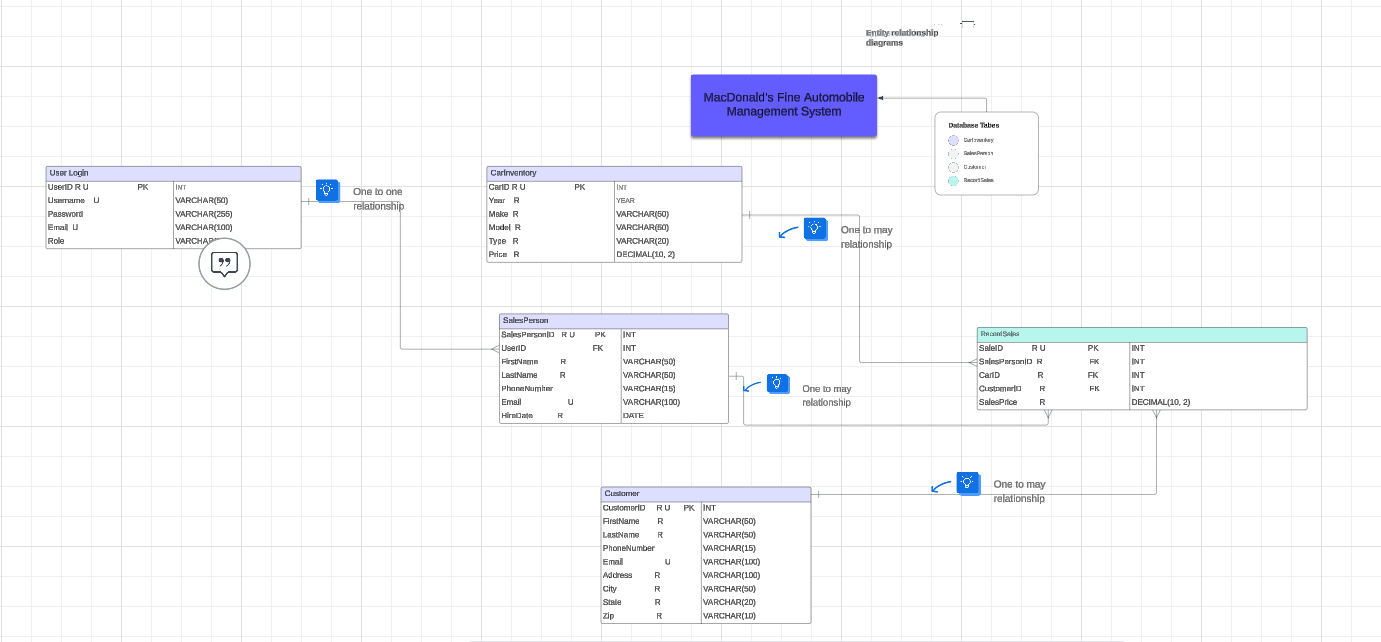
**6. Diagrams**

The Entity-Relationship (E/R) diagram represents the relationships between the entities in the database. Key elements include:

- Entities: UserLogin, CarInventory, SalesPerson, Customer, SalesRecord

- Relationships: Each RecordSales links a SalesPerson, CarInventory, and Customer, ensuring a clear and logical connection between these tables.

The diagram is developed using Lucidchart, following best practices for database design.



**7. ER Schema**

+-------------------+ +-------------------+ +-----------------------+

| UserLogin | | SalesPerson | | Customer |

+-------------------+ +-------------------+ +-----------------------+

| PK UserID |<-----> | PK SalesPersonID| | PK CustomerID |

| U Username | | FK UserID | | FK UserID |

| Password | | FirstName | | FirstName |

| U Email | | LastName | | LastName |

| Role | | Phone | | Phone |

+--------------------+ | Email | | Email |

+---------------------+ | City |

| State |

| ZIP |

+------------------------+

+-------------------+ +-----------------------+

| CarInventory | | RecordSales |

+-------------------+ +-----------------------+

| PK CarID |<---> | PK SalesID |

| Year | | FK SalesPersonID |

| Make | | FK CarID |

| Model | | FK CustomerID |

| Type | | SalesPrice |

| Price | +------------------------+

+-------------------+

**8. Conclusion**

This project successfully developed a comprehensive DBMS for managing Fine Automobiles' business operations. By implementing efficient data storage, retrieval, and manipulation techniques, the database will streamline inventory, sales, and customer management. The use of SQL Server ensures that the system is scalable, secure, and reliable.

**9. References**

- Microsoft SQL Server Documentation: https://docs.microsoft.com/en-us/sql/sql-server/

- Database Design Concepts by Elmasri & Navathe

- SQL Server Management Studio Guide: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms