Blue text on a black background

Description automatically generated

**SQL Databases (CPSC – 500 – 1)**

**Project Report**

**Final Group Project –**

**Database design & Implementation**

**CPSC-500-1 - SQL Databases**

**Professor Abbas Hamze**

**Group – 6**

**Kelechi C, Uchendu - (NF1002048)**

**Renato Hiroyuki Oshiro - (NF1011996)**

**Heta Chavda - (NF1014555)**

**Report: Implementation of Car Dealership Database**

**Content**

1. **Introduction 4**
2. **Objective 4**
3. **Overview 4**
4. **Database Design 5**
5. **Database Schema 7**
6. **Database Implementation 7**
7. **DML Statetments 16**
8. **Data Retrieval 24**
9. **Conclusion 30**
10. **Introduction**

This report presents the design and implementation of a relational database for the car dealership system. The main objective of the project was to implement an efficient database for storing and managing information about customers, cars, sales, services, and other relevant entities. The developed database will serve as a platform for various business operations, such as the management of inventories, analyses of sales, and management of customer relationships.

1. **Objective**

The focus of this report is to design and implement a robust, scalable, and efficient database system for a car dealership. The system will provide ease of operation in carrying out the major business processes, including inventory management, sales tracking, customer relationship management, and service scheduling.

1. **Overview**

This report will discuss the following major aspects related to the database project:

**1. Database Design*:***

Entity-Relationship (ER) Diagram: A diagrammatic representation of the entities and their relationships within the car dealership system.

Data Model: Detailed description of the database schema, including tables, attributes, and data types.

Normalization: Explanation of the normalization process used to eliminate redundancy and ensure data integrity.

**2. Database Implementation:**

Data Definition Language (DDL): Creation of database tables using CREATE TABLE statements.

Data Manipulation Language (DML): The insertion, updating, and deletion of data are done by INSERT, UPDATE, and DELETE statements, respectively.

Data Query Language (DQL): Data retrieval by the SELECT statement along with JOIN, WHERE , GROUP BY, and HAVING for complex queries.

**3. Data Preparation for Machine Learning:**

Data Cleaning and Preprocessing: How to handle missing values, outliers, and inconsistencies.

Feature Engineering: How new features are generated from already existing features to better the performance of the models.

Data Extraction: SQL queries to extract relevant data for machine learning tasks.

**4.Results and Analysis:**

Database Performance: Evaluation of query performance and database scalability.

Data Quality Assessment: Analysis of data accuracy and completeness.

1. **Database Design**

Entity-Relationship Diagram (ERD):

A diagram of a company

Description automatically generated

ERD describes the following entities and their relationships:

Customers: Storing information about customers, including their name, contact details, and address;

Cars: Maintaining the details about cars, such as make, model, year, price, and VIN;

Dealership: Name of the dealership, address, and phone number;

Employees: Information, including name, contact details, position, and department;

Inventory: Tracks the inventory level at each dealership.

Sales: This table records sales transactions, including customer, employee, car, sale date, and price.

Services: This table keeps track of the service history for cars, including the service date, description, and cost.

Payments: This table contains payments made by customers, including the payment method, date, and amount.

Warranty: This table stores warranty information for cars, including start and end dates and coverage details.

1. **Database Schema:**

The database schema was designed based on the ERD and normalized to minimize data redundancy and ensure data integrity. The following tables were created:

Customers (CustomerID, FirstName, LastName, PhoneNumber, Email, AddressID)

Cars (CarID, Make, Model, Year, Price, VIN)

Dealerships (DealershipID, DealershipName, AddressID, Phone)

Employees (EmployeeID, FirstName, LastName, PhoneNumber, Email, Position, Department, DealershipID)

Inventory (InventoryID, CarID, DealershipID, StockQuantity)

Sales (SaleID, CustomerID, EmployeeID, CarID, SaleDate, SalePrice)

Services (ServiceID, CarID, CustomerID, ServiceDate, ServiceDescription, Cost)

Payments (PaymentID, SaleID, PaymentMethod, PaymentDate, PaymentAmount)

Warranty (WarrantyID, CarID, CustomerID, WarrantyStartDate, WarrantyEndDate, CoverageDetails)

1. **Database Implementation**

The database was implemented by using SQL, specifically MySQL. DDL statements were used to create the tables and DML statements were used to populate the tables with sample data.

**DDL Statements:**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

PhoneNumber VARCHAR(20) NOT NULL UNIQUE,

Email VARCHAR(100) NOT NULL UNIQUE,

AddressID INT, FOREIGN KEY (AddressID) REFERENCES Address(AddressID)

);

**EXPLANATION**

CustomerID*:*

Serves as the Primary Key, uniquely identifying each customer.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

*FirstName and LastName:*

Store the customer's first and last names

Defined as NOT NULL to ensure these fields are always populated

*PhoneNumber and Email:*

Capture the customer’s contact details

Defined as NOT NULL UNIQUE to ensure no duplicate entries and maintain data integrity

AddressID:

A foreign key linking to the Address table (AddressID) to associate each customer with their address.

Supports a **relational database model** by avoiding duplication of address data.

**CREATING ADDRESS TABLE**

Create Address Table

CREATE TABLE Address ( AddressID INT PRIMARY KEY AUTO\_INCREMENT,

Street VARCHAR(255) NOT NULL,

City VARCHAR(100) NOT NULL,

State VARCHAR(100) NOT NULL,

PostalCode VARCHAR(20) NOT NULL);

**EXPLANATION**

AddressID:

Serves as the Primary Key for uniquely identifying each address.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

Street, City, State, PostalCode:

These columns store the complete address details:

Street: Specific address line.

City: The city where the address is located.

State: The state or region.

PostalCode: ZIP or postal code for precise location identification.

Defined as NOT NULL to ensure these fields are always populated.

**CREATING CAR TABLE**

Create Car Table

CREATE TABLE Cars (

CarID INT PRIMARY KEY AUTO\_INCREMENT,

Make VARCHAR(100) NOT NULL,

Model VARCHAR(100) NOT NULL,

Year INT NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

VIN VARCHAR(50) NOT NULL UNIQUE);

**EXPLANATION**

CarID:

Serves as the Primary Key, uniquely identifying each car.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

Make and Model:

Store the brand (e.g., Toyota) and model (e.g., Corolla) of the car.

Defined as NOT NULL to ensure all cars have this information

Year:

Represents the manufacturing year of the car

Defined as NOT NULL UNIQUE for data accuracy.

Price:

Stores the car's price as a decimal value, ensuring precision for monetary amounts (e.g., $25,000.50).

Defined as NOT NULL to always provide pricing details.

VIN:

Stores the Vehicle Identification Number, a globally unique identifier for each car.

Defined as NOT NULL UNIQUE to ensure no two cars share the same VIN, preventing duplication.

**CREATING DEALERSHIP TABLE**

Create Dealership Table

CREATE TABLE Dealership (

DealershipID INT PRIMARY KEY AUTO\_INCREMENT,

DealershipName VARCHAR(255) NOT NULL,

AddressID INT,

Phone VARCHAR(20) NOT NULL,

FOREIGN KEY (AddressID) REFERENCES Address(AddressID));

**EXPLANATION**

DealershipID:

Serves as the Primary Key, uniquely identifying each dealership.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

DealershipName:

Stores the name of the dealership branch (e.g., "ABC Motors").

Defined as NOT NULL to ensure every dealership has a name.

AddressID

A foreign key linking to the Address table (AddressID), associating each dealership with a specific location

Ensures consistency and reusability of address data.

Phone:

Stores the dealership’s contact phone number.

Defined as NOT NULL to ensure this critical contact information is always available.

**CREATING EMPLOYEE TABLE**

Create Employee Table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

PhoneNumber VARCHAR(20) NOT NULL UNIQUE,

Email VARCHAR(100) NOT NULL UNIQUE,

Position VARCHAR(100) NOT NULL,

Department VARCHAR(20),

DealershipID INT, FOREIGN KEY (DealershipID) REFERENCES Dealership(DealershipID));

**EXPLANATION**

EmployeeID:

Serves as the Primary Key, uniquely identifying each employee.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

FirstName and LastName:

Store the first and last names of employees.

Defined as NOT NULL to ensure these fields are always populated.

PhoneNumber and Email

Store the employee's contact information

Defined as NOT NULL UNIQUE to prevent duplicate entries and ensure data integrity.

Position and Department:

Position: The role or job title of the employee (e.g., Sales Manager, Mechanic).

Department: (Optional) Indicates the department within the dealership (e.g., Sales, Service).

DealershipID:

Employee table ensures that all employee details, roles, and branch affiliations are centrally managed

Linking to the Dealership table allows for easy assignment of staff to specific branches, supporting branch-specific reporting and management.

**CREATING INVENTORY TABLE**

Create Inventory Table

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY AUTO\_INCREMENT,

CarID INT NOT NULL,

DealershipID INT NOT NULL,

StockQuantity INT NOT NULL, FOREIGN KEY (CarID) REFERENCES Cars(CarID),

FOREIGN KEY (DealershipID) REFERENCES Dealership(DealershipID));

**EXPLANATION**

InventoryID:

Serves as the Primary Key, uniquely identifying each inventory record.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

CarID:

A foreign key referencing the Cars table (CarID).

Links each inventory record to a specific car model.

DealershipID:

A foreign key referencing the Dealership table (DealershipID).

Associates each inventory record with a specific dealership branch.

StockQuantity:

Stores the number of units available for a specific car at a given dealership

Defined as NOT NULL to ensure inventory data is complete

**CREATING SALES TABLE**

Create Sales Table

CREATE TABLE Sales (

SaleID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT NOT NULL,

EmployeeID INT NOT NULL,

CarID INT NOT NULL,

SaleDate DATE NOT NULL,

SalePrice DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

FOREIGN KEY (CarID) REFERENCES Cars(CarID));

**EXPLANATION**

SaleID:

Serves as the Primary Key, uniquely identifying each sale.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

CustomerID:

A foreign key referencing the Customers table (CustomerID).

Links the sale to the customer who made the purchase.

EmployeeID:

A foreign key referencing the Employee table

Identifies the employee responsible for facilitating the sale.

CarID:

A foreign key referencing the Cars table (CarID).

Specifies the car being sold in the transaction.

SaleDate:

Records the date the transaction occurred.

Defined as NOT NULL to ensure every sale has a timestamp.

SalePrice:

Stores the final price at which the car was sold.

Defined as DECIMAL(10, 2) to maintain precision for monetary values

**CREATING SERVICES TABLE**

Create Services Table

CREATE TABLE Services (

ServiceID INT PRIMARY KEY AUTO\_INCREMENT,

CarID INT NOT NULL,

CustomerID INT NOT NULL,

ServiceDate DATE NOT NULL,

ServiceDescription TEXT NOT NULL,

Cost DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (CarID) REFERENCES Cars(CarID),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID));

**EXPLANATION**

ServiceID:

Serves as the Primary Key, uniquely identifying each service.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

CustomerID:

A foreign key referencing the Customers table (CustomerID).

Associates the service with the customer who owns the car.

CarID:

A foreign key referencing the Cars table (CarID).

Links the service record to the specific car being serviced.

ServiceDate:

Records the date the service was performed.

Defined as NOT NULL to ensure every service has a timestamp.

ServiceDescription:

Provides details of the service performed (e.g., "Oil Change" or "Brake Repair").

Stored as TEXT to allow for comprehensive descriptions.

Cost:

Stores the monetary cost of the service.

Defined as DECIMAL(10, 2) for precision in financial records.

**CREATING PAYMENTS TABLE**

Create Payments Table

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY AUTO\_INCREMENT,

SaleID INT,

PaymentMethod VARCHAR(50) NOT NULL,

PaymentDate DATE NOT NULL,

PaymentAmount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (SaleID) REFERENCES Sales(SaleID));

**EXPLANATION**

PaymentID:

Serves as the Primary Key, uniquely identifying each payment record.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

SaleID:

A foreign key referencing the Sales table (SaleID).

Links the payment to the corresponding sale transaction.

PaymentMethod:

Stores the method of payment (e.g., "Credit Card," "Bank Transfer," "Cash").

Defined as VARCHAR(50) NOT NULL to ensure a valid payment method is always recorded.

PaymentDate:

Records the date the payment was made.

Defined as NOT NULL to ensure every payment has a timestamp.

PaymentAmount:

Stores the amount paid in the transaction.

Defined as DECIMAL(10, 2) for precision in financial records.

**CREATING WARRANTY TABLE**

Create Warranty Table

CREATE TABLE Warranty (

WarrantyID INT PRIMARY KEY AUTO\_INCREMENT,

CarID INT NOT NULL,

CustomerID INT NOT NULL,

WarrantyStartDate DATE NOT NULL,

WarrantyEndDate DATE NOT NULL,

CoverageDetails TEXT NOT NULL,

FOREIGN KEY (CarID) REFERENCES Cars(CarID),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID));

**EXPLANATION**

WarrantyID:

Serves as the Primary Key, uniquely identifying each Warranty record.

Uses AUTO\_INCREMENT to automatically generate sequential IDs.

CarID:

A foreign key referencing the Car table (CarID).

Links the warranty to a specific car covered under it.

CustomerID:

A foreign key referencing the Customers table

Associates the warranty with the customer who purchased the car.

WarrantyStartDate and WarrantyEndDate:

Records the date the payment was made.

Both are defined as NOT NULL to ensure complete warranty coverage information.

CoverageDetails:

Provides details about what is covered under the warranty (e.g., parts, labour).

Stored as TEXT to allow for comprehensive descriptions.

1. **DML Statements**

**POPULATING CUSTOMER TABLE**

INSERT INTO Customers (FirstName, LastName, PhoneNumber, Email, AddressID)VALUES ('John', 'Doe', '123-456-7890', 'john.doe@example.com', 1),

('Jane', 'Smith', '234-567-8901', 'jane.smith@example.com', 2),

('Emily', 'Brown', '345-678-9012', 'emily.brown@example.com', 3),

('Michael', 'Johnson', '456-789-0123', 'michael.johnson@example.com', 4),

('Sarah', 'Taylor', '567-890-1234', 'sarah.taylor@example.com', 5),('Chris', 'Wilson', '678-901-2345', 'chris.wilson@example.com', 6),

('Jessica', 'Moore', '789-012-3456', 'jessica.moore@example.com', 7),

('Daniel', 'Davis', '890-123-4567', 'daniel.davis@example.com', 8),

('Sophia', 'Martin', '901-234-5678', 'sophia.martin@example.com', 9),

('Oliver', 'Garcia', '123-345-6789', 'oliver.garcia@example.com', 10);

**EXPLANATION**

INSERT INTO Customers (FirstName, LastName, PhoneNumber, Email, AddressID):

Specifies the target table (Customers) and the columns (FirstName, LastName, PhoneNumber, Email, AddressID) where data is to be inserted.

VALUES*:*

Provides details for multiple customers, including:

FirstName and LastName: The customer's full name (e.g., "John Doe").

PhoneNumber: The customer's phone number (e.g., "123-456-7890").

Email: The customer's email address (e.g., "john.doe@example.com").

AddressID: A foreign key linking the customer to their address in the Address table (e.g., 1 for "123 Main St").

Bulk Insertion:

Efficiently inserts 10 rows of customer data in a single query.

**POPULATING ADDRESS TABLE**

INSERT INTO Address (Street, City, State, PostalCode)VALUES

('123 Main St', 'Niagara Falls', 'Ontario', 'L2G1C5'),

('456 Elm St', 'Toronto', 'Ontario', 'M5G1X8'),

('789 Pine St', 'Hamilton', 'Ontario', 'L8P3B3'),

('101 Maple Ave', 'St. Catharines', 'Ontario', 'L2R2C1'),

('200 Oak Rd', 'Mississauga', 'Ontario', 'L5M2G1'),

('55 Birch Ln', 'Brantford', 'Ontario', 'N3T1H4'),

('67 Walnut Dr', 'Burlington', 'Ontario', 'L7P4N3'),

('305 Cedar Ct', 'Oshawa', 'Ontario', 'L1G5V7'),

('400 Spruce St', 'London', 'Ontario', 'N6A2K3'),

('222 Willow Way', 'Kingston', 'Ontario', 'K7L5E6');

**EXPLANATION**

INSERT INTO Address (Street, City, State, PostalCode):

Specifies the target table (Address) and the columns (Street, City, State, PostalCode) to insert data into.

VALUES:

Lists the address details for multiple records, including:

Street: The specific address line (e.g., "123 Main St").

City: The city of the address (e.g., "Niagara Falls").

State: The state or province (e.g., "Ontario").

PostalCode: The postal or ZIP code (e.g., "L2G1C5").

Bulk Insertion:

Inserts multiple rows in one query for efficiency.

**POPULATING CARS TABLE**

INSERT INTO Cars (Make, Model, Year, Price, VIN)VALUES

('Toyota', 'Corolla', 2022, 25000.00, '1HGCM82633A123456'),

('Honda', 'Civic', 2021, 22000.00, '1HGCM82633A654321'),

('Ford', 'F-150', 2023, 40000.00, '1FTRX12W67KD12345'),

('Chevrolet', 'Malibu', 2020, 18000.00, '2C3CDXBG3GH765432'),

('Nissan', 'Rogue', 2022, 28000.00, 'JN8AT2MV3LW123456'),

('Hyundai', 'Elantra', 2023, 22000.00, 'KMHDH4AE1EU876543'),

('Kia', 'Sorento', 2021, 30000.00, '5XYKTDA26BG123456'),

('Mazda', 'CX-5', 2023, 32000.00, 'JM3KE2BE1F0123456'),

('Tesla', 'Model 3', 2023, 45000.00, '5YJ3E1EA8KF123456'),

('Volkswagen', 'Jetta', 2022, 20000.00, '3VW4A7AU2KM123456');

**EXPLANATION**

INSERT INTO Cars (Make, Model, Year, Price, VIN):

Specifies the target table (Cars) and the columns (Make, Model, Year, Price, VIN) for the data insertion.

VALUES:

Provides details for multiple cars, including:

Make: Manufacturer of the car (e.g., "Toyota").

Model: Specific car model (e.g., "Corolla").

Year: Manufacturing year of the car (e.g., 2022).

Price: Cost of the car, defined as a decimal (e.g., $25,000.00).

VIN: Unique Vehicle Identification Number to ensure the car's global traceability

Bulk Insertion:

Efficiently inserts 10 rows of car data in a single query.

**POPULATING DEALERSHIP TABLE**

INSERT INTO Dealership (DealershipName, AddressID, Phone)VALUES

('Niagara Cars', 1, '905-123-4567'),

('Toronto Auto', 2, '416-234-5678'),

('Hamilton Wheels', 3, '905-345-6789'),

('Mississauga Motors', 5, '905-456-7890'),

('London Dealership', 9, '519-567-8901');

**EXPLANATION**

INSERT INTO Dealership (DealershipName, AddressID, Phone):

Specifies the target table (Dealership) and the columns (DealershipName, AddressID, Phone) to insert data into.

VALUES:

Provides details for multiple dealerships:

DealershipName: The name of the dealership (e.g., "Niagara Cars").

AddressID: A foreign key linking to the Address table to associate each dealership with its location.

Phone: The dealership's contact number (e.g., "905-123-4567").

Bulk Insertion:

Efficiently inserts 5 rows of dealership data in a single query.

**POPULATING EMPLOYEES TABLE**

INSERT INTO Employees (FirstName, LastName, PhoneNumber, Email, Position, Department, DealershipID)VALUES

('Mike', 'Johnson', '345-678-9012', 'mike.johnson@example.com', 'Sales Manager', 'Sales', 1),

('Anna', 'Lee', '567-890-1234', 'anna.lee@example.com', 'Sales Associate', 'Sales', 1),

('Mark', 'Chen', '678-901-2345', 'mark.chen@example.com', 'Finance Manager', 'Finance', 2),

('Laura', 'Kim', '789-012-3456', 'laura.kim@example.com', 'Sales Associate', 'Sales', 3),

('David', 'King', '890-123-4567', 'david.king@example.com', 'Service Manager', 'Services', 4),

('Mary', 'Doe', '801-801-8012', 'mary.doe@example.com', 'Recruiter', 'HR', 2);

**EXPLANATION**

INSERT INTO Employees (FirstName, LastName, PhoneNumber, Email, Position, Department, DealershipID):

Specifies the target table (Employees) and the columns (FirstName, LastName, PhoneNumber, Email, Position, Department, DealershipID) for data insertion.

VALUES:

Provides details for multiple employees:

FirstName and LastName: Full name of the employee (e.g., "Mike Johnson").

PhoneNumber and Email: Contact information for each employee.

Position: Job title (e.g., "Sales Manager").

Department: Department the employee works in (e.g., "Sales").

DealershipID: Foreign key linking the employee to the specific dealership branch.

Bulk Insertion:

Inserts 6 rows of employee data in a single query, covering various roles and dealership assignments.

**POPULATING INVENTORY TABLE**

INSERT INTO Inventory (CarID, DealershipID, StockQuantity)VALUES

(1, 1, 10),

(2, 2, 5),

(3, 3, 8),

(4, 4, 4),

(5, 5, 6),

(6, 1, 3),

(7, 2, 7),

(8, 3, 9),

(9, 4, 2),

(10, 5, 1);

**EXPLANATION**

INSERT INTO Inventory (CarID, DealershipID, StockQuantity):

Specifies the target table (Inventory) and the columns (CarID, DealershipID, StockQuantity) for the data insertion.

VALUES:

Provides details for inventory records:

CarID: A foreign key referencing the Cars table, linking each inventory record to a specific car.

DealershipID: A foreign key referencing the Dealership table, linking the inventory to a specific dealership branch.

StockQuantity: The number of units available for that car at the respective dealership.

Bulk Insertion:

Inserts 10 rows of inventory data in a single query, distributing cars across dealerships.

**POPULATING SALES TABLE**

INSERT INTO Sales (CustomerID, EmployeeID, CarID, SaleDate, SalePrice)VALUES

(1, 1, 1, '2024-01-01', 25000.00),

(2, 2, 2, '2024-01-05', 22000.00),

(3, 3, 3, '2024-01-10', 40000.00),

(4, 4, 4, '2024-01-15', 18000.00),

(5, 5, 5, '2024-01-20', 28000.00);

**EXPLANATION**

INSERT INTO Sales (CustomerID, EmployeeID, CarID, SaleDate, SalePrice):

Specifies the target table (Sales) and the columns (CustomerID, EmployeeID, CarID, SaleDate, SalePrice) to insert data into.

VALUES:

Provides details for multiple sales transactions:

CustomerID: A foreign key linking to the Customers table, identifying the buyer.

EmployeeID: A foreign key linking to the Employees table, identifying the sales representative responsible for the transaction.

CarID: A foreign key linking to the Cars table, identifying the car sold.

SaleDate: The date of the transaction (e.g., '2024-01-01').

SalePrice: The price at which the car was sold (e.g., $25,000.00).

Bulk Insertion:

Inserts 5 rows of sales data in a single query.

**POPULATING SALES TABLE**

INSERT INTO Services (CarID, CustomerID, ServiceDate, ServiceDescription, Cost)VALUES

(1, 1, '2024-02-01', 'Oil Change', 50.00),

(2, 2, '2024-02-05', 'Tire Rotation', 80.00),

(3, 3, '2024-02-10', 'Brake Replacement', 250.00),

(4, 4, '2024-02-15', 'Battery Replacement', 200.00),

(5, 5, '2024-02-20', 'Wheel Alignment', 100.00);

**EXPLANATION**

INSERT INTO Services (CarID, CustomerID, ServiceDate, ServiceDescription, Cost):

Specifies the target table (Services) and the columns (CarID, CustomerID, ServiceDate, ServiceDescription, Cost) for the data insertion.

VALUES:

Provides details for multiple sales transactions:

CarID: A foreign key linking to the Cars table, identifying the car being serviced.

CustomerID: A foreign key linking to the Customers table, identifying the car owner.

ServiceDate: The date when the service was performed (e.g., '2024-02-01').

ServiceDescription: A detailed description of the service performed (e.g., "Oil Change").

Cost: The cost of the service (e.g., $50.00).

Bulk Insertion:

Inserts 5 rows of service data in a single query.

**POPULATING PAYMENTS TABLE**

INSERT INTO Payments (SaleID, PaymentMethod, PaymentDate, PaymentAmount)VALUES

(1, 'Credit Card', '2024-01-01', 25000.00),

(2, 'Debit Card', '2024-01-05', 22000.00),

(3, 'Bank Transfer', '2024-01-10', 40000.00),

(4, 'Cash', '2024-01-15', 18000.00),

(5, 'Credit Card', '2024-01-20', 28000.00),

(NULL, 'Credit Card', '2024-02-01', 50.00),

(NULL, 'Debit Card', '2024-02-05', 80.00),

(NULL, 'Bank Transfer', '2024-02-10', 250.00),

(NULL, 'Cash', '2024-02-15', 200.00),

(NULL, 'Credit Card', '2024-02-20', 100.00);

**EXPLANATION**

INSERT INTO Payments (SaleID, PaymentMethod, PaymentDate, PaymentAmount):

Specifies the target table (Payments) and the columns (SaleID, PaymentMethod, PaymentDate, PaymentAmount) to insert data into.

VALUES:

Provides details for multiple payment transactions:

SaleID: A foreign key linking to the Sales table. For service payments (not linked to car sales), this value is NULL.

PaymentMethod: The method of payment used (e.g., "Credit Card", "Cash").

PaymentDate: The date of payment (e.g., '2024-01-01’).

PaymentAmount: The amount paid (e.g., $25,000.00).

Mixed Data:

Includes payments for.

Car Sales: Payments linked to car purchases (e.g., SaleID 1–5).

Services: Payments not linked to car sales, represented by SaleID = NULL.

Bulk Insertion:

Efficiently inserts 10 rows of payment data in a single query.

**POPULATING WARRANTY TABLE**

INSERT INTO Warranty (CarID, CustomerID, WarrantyStartDate, WarrantyEndDate, CoverageDetails)VALUES

(1, 1, '2024-01-01', '2026-01-01', 'Bumper-to-Bumper Coverage'),

(2, 2, '2024-01-05', '2026-01-05', 'Powertrain Warranty'),

(3, 3, '2024-01-10', '2026-01-10', 'Extended Warranty'),

(4, 4, '2024-01-15', '2026-01-15', 'Basic Warranty'),

(5, 5, '2024-01-20', '2026-01-20', 'Roadside Assistance Included');

**EXPLANATION**

INSERT INTO Warranty (CarID, CustomerID, WarrantyStartDate, WarrantyEndDate, CoverageDetails):

Specifies the target table (Warranty) and the columns (CarID, CustomerID, WarrantyStartDate, WarrantyEndDate, CoverageDetails) for the data insertion.

VALUES:

Provides details for multiple warranty records:

CarID: A foreign key linking to the Cars table, identifying the car covered under the warranty.

CustomerID: A foreign key linking to the Customers table, identifying the buyer associated with the warranty.

WarrantyStartDate and WarrantyEndDate: Define the validity period of the warranty (e.g., '2024-01-01' to '2026-01-01').

CoverageDetails: Specifies the type of coverage provided (e.g., "Bumper-to-Bumper Coverage").

Bulk Insertion:

Inserts 5 rows of warranty data in a single query.

1. **Data Retrieval**

Get all customer information and their corresponding car purchases

SELECT c.FirstName, c.LastName, c.Email, ca.Make, ca.Model, s.SaleDate, s.SalePrice

FROM customers as c

JOIN Sales s ON c.CustomerID = s.CustomerID

JOIN Cars ca ON s.CarID = ca.CarID;

**EXPLANATION**

SELECT Clause:

Specifies the columns to retrieve.

c.FirstName, c.LastName, c.Email: Retrieves customer details from the Customers table

ca.Make, ca.Model: Retrieves car details from the Cars table.

s.SaleDate, s.SalePrice: Retrieves sales transaction details from the Sales table.

FROM Clause:

Begins with the Customers (c) table as the base table:

JOIN Clauses:

JOIN Sales s ON c.CustomerID = s.CustomerID:

Links the Customers table to the Sales table using the CustomerID foreign key.

Ensures only customers with purchase records are included.

JOIN Cars ca ON s.CarID = ca.CarID:

Links the Sales table to the Cars table using the CarID foreign key.

Retrieves details about the cars purchased.

Data Relationship:

Connects Customers, Sales, and Cars tables to generate a complete view of customer purchase history.

**OUTPUT TO RETRIEVAL QUERY 1**

A screenshot of a computer

Description automatically generated

**Get total sales by dealership**

SELECT d.DealershipName, SUM(s.SalePrice) AS TotalSales

FROM Sales as s

JOIN Inventory as i ON s.CarID = i.CarID

JOIN Dealership as d ON i.DealershipID = d.DealershipID

GROUP BY d.DealershipName;

**EXPLANATION**

SELECT Clause:

Specifies the columns to retrieve.

d.DealershipName: Retrieves the name of the dealership

SUM(s.SalePrice) AS TotalSales: Calculates the total revenue generated by each dealership by summing the sale prices from the Sales table.

FROM Clause:

Starts with the Sales (s) table as the base table:

JOIN Clauses:

JOIN Inventory i ON s.CarID = i.CarID:

Links the Sales table to the Inventory table using the CarID foreign key.

Ensures that each sale is tied to an inventory record.

JOIN Dealership d ON i.DealershipID = d.DealershipID:

Links the Inventory table to the Dealership table using the DealershipID foreign key.

Associates each inventory item with the dealership where it is stocked.

GROUP BY Clause

GROUP BY d.DealershipName:

Groups the results by dealership name so that total sales for each dealership can be calculated.

A screenshot of a computer

Description automatically generated

**Find customers who purchased a specific car model**

SELECT c.FirstName, c.LastName, ca.Make, ca.Model

FROM Customers c

JOIN Sales s ON c.CustomerID = s.CustomerID

JOIN Cars ca ON s.CarID = ca.CarID

WHERE ca.Model = 'Corolla';

**EXPLANATION**

SELECT Clause:

Specifies the columns to retrieve.

c.FirstName, c.LastName: Retrieves the customer's first and last names from the Customers table.

ca.Make, ca.Model: Retrieves the car's make (e.g., Toyota) and model (e.g., Corolla) from the Cars table.

FROM Clause:

Begins with the Customers (c) table as the base:

JOIN Clauses:

JOIN Sales s ON c.CustomerID = s.CustomerID:

Links the Customers table to the Sales table via CustomerID, ensuring only customers with purchases are included.

JOIN Cars ca ON s.CarID = ca.CarID:

Links the Sales table to the Cars table via CarID, providing details about the car purchased.

WHERE Clause:

ca.Model = 'Corolla’: Filters the results to include only customers who purchased the specific car model "Corolla.".



**Retrieve inventory details**

SELECT d.DealershipName, ca.Make, ca.Model, i.StockQuantity

FROM Inventory I

JOIN Cars ca ON i.CarID = ca.CarID

JOIN Dealership d ON i.DealershipID = d.DealershipID;

**EXPLANATION**

SELECT Clause:

Specifies the columns to retrieve.

d.DealershipName: Retrieves the name of the dealership from the Dealership table.

ca.Make, ca.Model: Retrieves the car's make (e.g., Toyota) and model (e.g., Corolla) from the Cars table.

FROM Clause:

Begins with the Inventory (i) table, which holds car stock information for various dealerships:

JOIN Clauses:

JOIN Cars ca ON i.CarID = ca.CarID:

Links the Inventory table to the Cars table via CarID, associating inventory data with car details.

JOIN Dealership d ON i.DealershipID = d.DealershipID:

Links the Inventory table to the Dealership table via DealershipID, associating inventory data with dealership locations.

A screenshot of a computer

Description automatically generated

**Aggregate services cost by customer**

SELECT c.FirstName, c.LastName, SUM(s.Cost) AS TotalServiceCost

FROM Customers c

JOIN Services s ON c.CustomerID = s.CustomerID

GROUP BY c.FirstName, c.LastName;

**EXPLANATION**

SELECT Clause:

Specifies the columns to retrieve.

c.FirstName, c.LastName: Retrieves the customer's first and last names from the Customers table.

SUM(s.Cost) AS TotalServiceCost: Calculates the total cost of services for each customer by summing the Cost column from the Services table.

FROM Clause:

Begins with the Customers (c) table, which contains customer information:

JOIN Clauses:

JOIN Services s ON c.CustomerID = s.CustomerID:

Links the Customers table to the Services table via CustomerID, ensuring each service is associated with the correct customer.

GROUP BY Clause:

GROUP BY c.FirstName, c.LastName’:

Groups the results by customer name so that the total service cost is calculated for each individual.

A screenshot of a table

Description automatically generated

1. **CONCLUSION**

This database serves as the backbone of an efficient car dealership system, empowering stakeholders with actionable insights and paving the way for data-driven growth including but not limited to the following areas.

**Integrated System**: The database successfully connects core entities such as customers, cars, employees, inventory, sales, services, payments, and warranties, ensuring seamless data flow across all dealership operations.

**Enhanced Decision-Making**: Enables real-time insights into inventory management, dealership sales performance, service revenue, and customer behavior for strategic decision-making.

**Scalability and Reusability**: Designed with a normalized structure to minimize redundancy and ensure data integrity, making it scalable for future expansion (e.g., additional dealerships or service centres).

**Machine Learning Readiness**: The database structure allows easy extraction of meaningful datasets for predictive modeling, such as customer purchase trends, service frequency, and inventory demand forecasting.

**Operational Efficiency**: Supports operational goals like tracking sales, managing warranties, monitoring stock levels, and ensuring after-sales service quality.