UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT ON MAHINDRA CAR SHOWROOM MANAGEMENT SYSTEM

Program name: BCA

Subject name/Code: Data Management System (23CAT-251/23CAP-252)

Submitted by:

Name: Vansh Kumar

ID: 23BCA10471

Section: 23BCA- 4(B)

Submitted to:

Name: Mr. Arvinder Singh

ACKNOWLEDGEMENT

We would like to express our sincere gratitude for the support and guidance we received throughout the development of our project, Mahindra Car Showroom Management System, carried out as part of our Database Management System (DBMS) coursework.

We are especially thankful to our mentor, Mr. Arvinder Singh, for his continuous encouragement and expert guidance. His in-depth knowledge of database concepts and SQL programming greatly contributed to the successful implementation of our project. His insights helped us understand the core principles of database design and apply them effectively in building a functional and efficient management system.

We are also grateful to our college and faculty members for providing a learning environment that encouraged practical application of theoretical concepts. Their support enabled us to approach the project with confidence and clarity.

Additionally, we appreciate the feedback and encouragement from our peers, which helped us identify areas of improvement and refine our system further.

This project has been a valuable learning experience, allowing us to enhance our technical skills and better understand the real-world applications of DBMS. We are truly thankful to everyone who contributed to the completion of this project.

INTRODUCTION

Overview

The Mahindra Car Showroom Management System is a database project developed to manage and automate the core functions of a Mahindra car dealership. It is designed to handle various tasks such as managing car inventory, customer details, sales records, staff information, and service bookings efficiently.

The objective of this system is to reduce manual effort, eliminate errors, and provide quick access to accurate data through a structured and well-organized database. Built using SQL and DBMS concepts, the project demonstrates practical knowledge of data modeling, normalization, and relational database design. This system enhances operational efficiency and offers a scalable solution for real-world showroom management needs.

Importance of DBMS in Car Showrooms

DBMS is the foundation of the Mahindra Car Showroom Management System, ensuring efficient handling of data related to cars, customers, sales, employees, and services. It allows structured storage and easy retrieval of information using SQL queries. DBMS helps reduce data redundancy, maintain data integrity, and ensure consistency across the system. With features like data security, backup, and multi-user access, it supports smooth daily operations and quick decision-making. By applying concepts like normalization and relational design, the system becomes more scalable and reliable, ultimately enhancing the overall management of the showroom.

Key Objectives:

1. Store and manage vehicle inventory with specifications, prices, and stock levels.
2. Maintain customer records including personal details and purchase history.
3. Track sales transactions, invoices, and payment details efficiently.
4. Manage employee data, roles, and performance records.
5. Log service and test drive bookings linked to customer and vehicle data.
6. Generate operational reports on sales, inventory, and customer insights.
7. Ensure data integrity through normalization and relational design.
8. Enable secure and efficient data access using SQL queries.

SYSTEM FUNCTIONALITY

The Mahindra Car Showroom Management System enables efficient management of vehicle inventory, customer records, sales transactions, employee data, and service bookings. It allows real-time updates, easy data retrieval, and automated report generation. The system ensures data integrity, accuracy, and security while streamlining daily operations of the showroom.

- Cars
- Customers
- Employees
- Sales
- Test-drives
- Services
- Inventory
- Showrooms

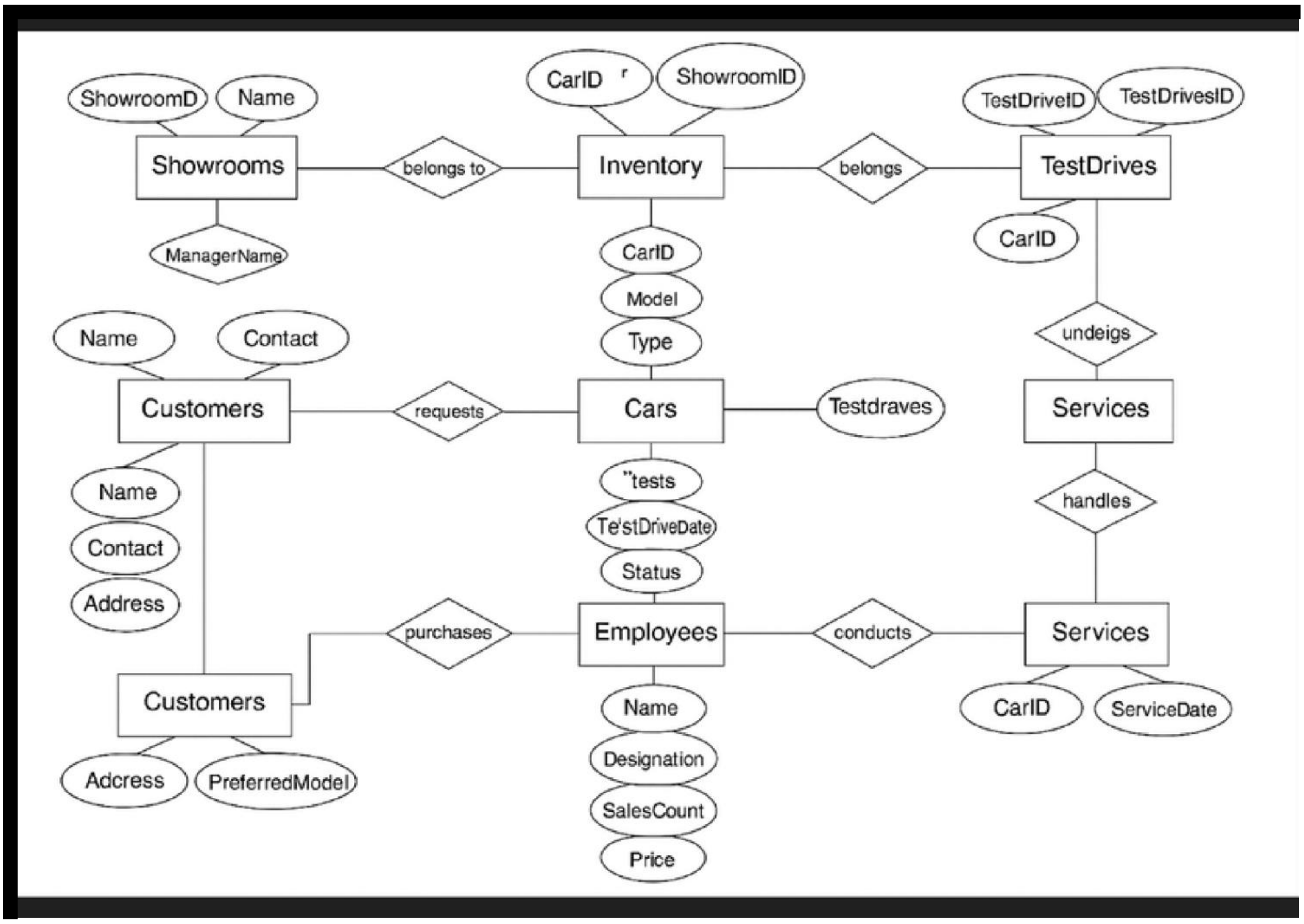
TECHNOLOGIES USED:

DBMS: MySQL

Tools:

MySQL Workbench

ER Diagram :



✵ Expected Benefits

1. Improved Data Organization: Centralized data storage ensures easy access and better organization of vehicle, customer, and sales records.
2. Data Integrity: Ensures accuracy, consistency, and reliability through normalization and validation rules.
3. Enhanced Security: Provides secure data access, backup, and user authentication.
4. Efficient Data Retrieval: Fast querying and retrieval of data for reports, transactions, and inventory management.
5. Reduced Redundancy: Minimizes duplicate data and ensures data consistency across the system.
6. Scalability: Can handle growing data needs as the business expands.
7. Real-time Updates: Ensures up-to-date information for decision-making and operational efficiency.

Techniques Used in Designing Relational Schema:

1. Entity-Relationship (ER) Modeling
   - The first step is creating an ER diagram that defines all the entities (such as Car, Customer, Employee, Sale, Service, etc.) and their relationships. This helps identify the tables and relationships between them.
2. Normalization
   - Normalization (1NF, 2NF, and 3NF) is applied to ensure that the relational schema is free from redundancy and anomalies. It ensures that each table contains only relevant data, improves data integrity, and optimizes storage.
3. Primary Keys and Foreign Keys
   - Primary keys are used to uniquely identify each record in a table (e.g., Car_ID for the Car table, Customer_ID for the Customer table).
   - Foreign keys establish relationships between tables (e.g., linking a Sales table to a Customer table through Customer_ID).
4. Relational Integrity Constraints
   - Integrity constraints like NOT NULL, UNIQUE, CHECK, and DEFAULT are used to enforce business rules at the database level, ensuring the accuracy and consistency of the data.
5. One-to-Many and Many-to-Many Relationships
   - For example, a one-to-many relationship between Customer and Sales (one customer can make multiple purchases) or a many-to-many relationship between Car and Service (a car can be serviced multiple times, and a service can be linked to multiple cars).
   - These are implemented using join tables (e.g., a Car_Service table for the many-to-many relationship).
6. Join Operations
   - The relational schema supports JOIN operations (inner, outer joins) to combine data from multiple tables when generating reports or querying combined data from related entities.
7. Indexes
   - Indexes are created on frequently queried fields, like Car_ID, Customer_ID, and Sales_ID, to speed up data retrieval operations.
8. Stored Procedures and Triggers

  - Stored procedures can automate complex tasks like updating stock levels after a sale, while triggers might be used to update related tables automatically (e.g., after inserting a new sale, updating inventory levels).

9. Data Consistency and Constraints
   - The use of consistency constraints ensures that data entered into tables is valid, for example, ensuring that the sale date or the test drive date is not in the future.

Software Requirements:

1. Database Management System (DBMS):
   - MySQL or SQL Server (for database creation, management, and querying).
   - Alternatively, Oracle or PostgreSQL can be used for relational database management.
2. Programming Languages:
   - SQL (Structured Query Language) for writing queries to interact with the database.
   - Optional: PHP or Java for developing the front-end or integrating with the database.
3. Database Design Tool:
   - MySQL Workbench or pgAdmin for designing and managing the database schema.
   - ER/Studio or DBDesigner for creating Entity-Relationship diagrams.
4. Operating System:
   - Windows 10 or Linux (Ubuntu, CentOS) for hosting the database and running the development environment.
5. Web Browser:
   - Google Chrome, Mozilla Firefox, or any other modern browser for accessing any web-based user interface (if developed).
6. IDE (Integrated Development Environment):
   - Visual Studio Code or Eclipse (for writing and debugging any code or scripts, if required).
7. Backup and Recovery Tools:

- XAMPP (for local server simulation), or native DBMS backup tools for managing database backups.

Hardware Requirements:

1. Processor (CPU):
    - Intel Core i3 or equivalent (minimum) for smooth operation of the DBMS and development tools.
    - Intel Core i5 or higher is recommended for better performance.
2. RAM:
    - 4 GB RAM minimum.
    - 8 GB RAM or more recommended for handling larger datasets or multi-user environments.
3. Hard Drive (Storage):
    - At least 20 GB of free disk space for storing the DBMS, tools, and project files.
    - SSD recommended for faster read/write speeds.
4. Network:
    - Internet Connection (for downloading necessary software, updates, and for hosting if it's a cloud-based project).
    - Local Area Network (LAN) for multi-user access, if required.
5. Display:
    - 1024x768 screen resolution or higher for clear view and easy navigation in IDEs and DBMS tools.
6. Peripherals:
    - Mouse and Keyboard (standard input devices).
    - Printer (optional, for printing reports or documentation).

Performance Enhancements:

1. Indexing:

  - Use indexes on frequently queried fields (like Car_ID, Customer_ID, and Sales_ID) to speed up search and retrieval operations. This minimizes query execution time, especially for large datasets.

2. Query Optimization:
    - Optimize SQL queries by using proper joins, avoiding unnecessary subqueries, and selecting only necessary columns. Use EXPLAIN statements to analyze and improve query performance.
3. Database Normalization:
    - Apply proper normalization (up to 3NF) to minimize redundancy and improve storage efficiency. This also reduces the overall size of the database, enhancing query performance.
4. Use of Stored Procedures:
    - Implement stored procedures for common operations like updating stock levels, generating reports, or inserting records. Stored procedures run on the server side, reducing the need for frequent communication between the application and database.
5. Caching:
    - Implement caching mechanisms (e.g., Memcached or Redis) to store frequently accessed data temporarily. This reduces the need to query the database repeatedly, speeding up data retrieval.

Aim/Overview Of The Project

The Mahindra Car Showroom Management System aims to automate and streamline showroom operations, including inventory management, customer records, sales transactions, and service bookings. By utilizing a relational database, the system ensures efficient data management, real-time updates, and accurate reporting, enhancing decision-making and operational efficiency.

Objective:

The primary objective of the Mahindra Car Showroom Management System is to efficiently manage and streamline the core operations of a car showroom using a relational database system. The system is designed to:

1. Track Inventory: Monitor and manage the stock of various car models across multiple showrooms, ensuring availability and quick access to product details.
2. Customer Management: Store and manage customer information, including contact details and preferred car models, for personalized service and follow-ups.

3. Sales Management: Record sales transactions, associating each sale with specific cars, customers, and employees, and generate accurate sales reports.
4. Employee Management: Track employee details and sales performance, helping to evaluate staff productivity and reward high performers.
5. Test Drives: Manage and record test drive bookings, ensuring efficient scheduling and follow-up.
6. Service Management: Track services provided to cars, including service types, costs, and service dates, ensuring timely maintenance of vehicles.
7. Reporting and Insights: Generate reports on sales, inventory status, employee performance, test drive statistics, and more to aid in decision-making and business analysis.

The system enhances operational efficiency, reduces manual errors, and provides real-time insights, making it easier to manage day-to-day activities at the car showroom.

Entity-Relationship (ER) Diagram Overview

Here is the Entity-Relationship (ER) Diagram for the Mahindra Car Showroom

Management System based on the provided SQL code:

Entities:

1. Showrooms
    - Attributes: ShowroomID (PK), Name, Location, ManagerName
2. Cars
    - Attributes: CarID (PK), Model, Type, Engine, Price
3. Inventory
    - Attributes: CarID (FK), ShowroomID (FK), Quantity
    - Primary Key: (CarID, ShowroomID)
4. Customers
    - Attributes: CustomerID (PK), Name, Contact, Address, PreferredModel
5. Employees
    - Attributes: EmployeeID (PK), Name, Designation, SalesCount
6. Sales
    - Attributes: SaleID (PK), CarID (FK), CustomerID (FK), EmployeeID (FK), SaleDate, Price

  - Foreign Keys: CarID, CustomerID, EmployeeID

7. TestDrives
   - Attributes: TestDriveID (PK), CustomerID (FK), CarID (FK), TestDriveDate, Status
   - Foreign Keys: CustomerID, CarID
8. Services
   - Attributes: ServiceID (PK), CarID (FK), ServiceDate, Description, Cost
   - Foreign Key: CarID

Relationships:

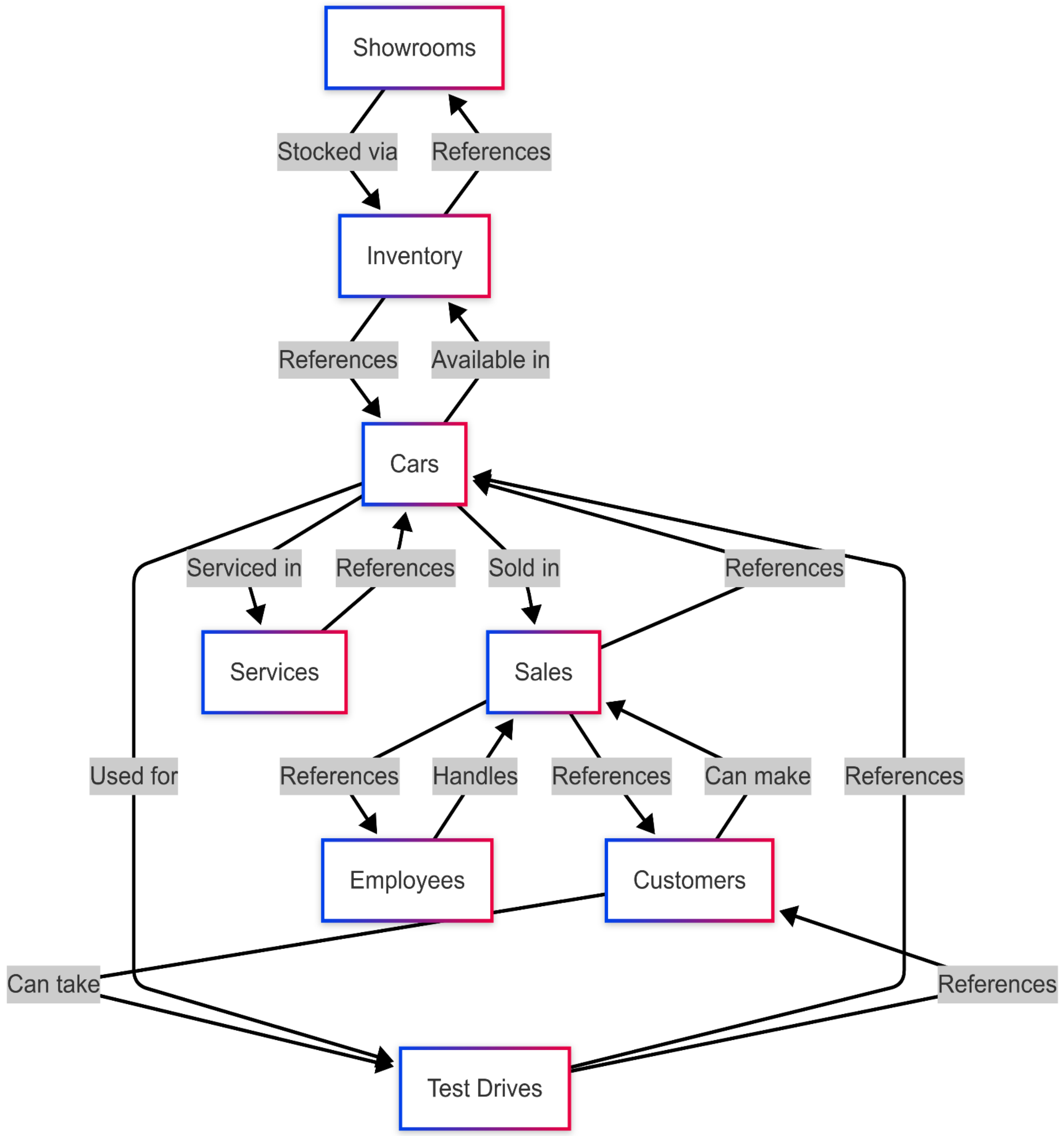
- Showrooms to Inventory
  A Showroom can have many Inventory records (one-to-many). Each inventory record relates to one Car and one Showroom.
- Cars to Inventory
  A Car can be present in multiple Inventory records across different Showrooms (one-to-many).
- Customers to Sales
  A Customer can make multiple Sales (one-to-many). Each sale is linked to a specific Car and an Employee (Salesperson).
- Employees to Sales
  An Employee can process multiple Sales (one-to-many).
- Cars to TestDrives
  A Car can be test-driven by multiple Customers (one-to-many).
- Customers to TestDrives
  A Customer can request multiple Test Drives (one-to-many).
- Cars to Services
  A Car can undergo multiple Services (one-to-many). Each service is associated with a specific car.

Entity-Relationship Diagram (ERD) Representation:

The ERD includes the following components:

- Rectangles for entities (Showrooms, Cars, Customers, Employees, etc.)
- Ovals for attributes of each entity.
- Diamonds for relationships (e.g., "Makes" for the Sales relationship between Customers and Cars).
- Lines connecting entities and relationships with cardinality (one-to-many, many-to-many, etc.)

E – R Diagram


Showrooms
Stocked via
References
Inventory
References
Available in
Cars
Serviced in
References
Sold in
References
Services
Sales
Used for
References
Handles
References
Can make
References
Employees
Customers
Can take
References
Test Drives

RELATIONAL SCHEMA:

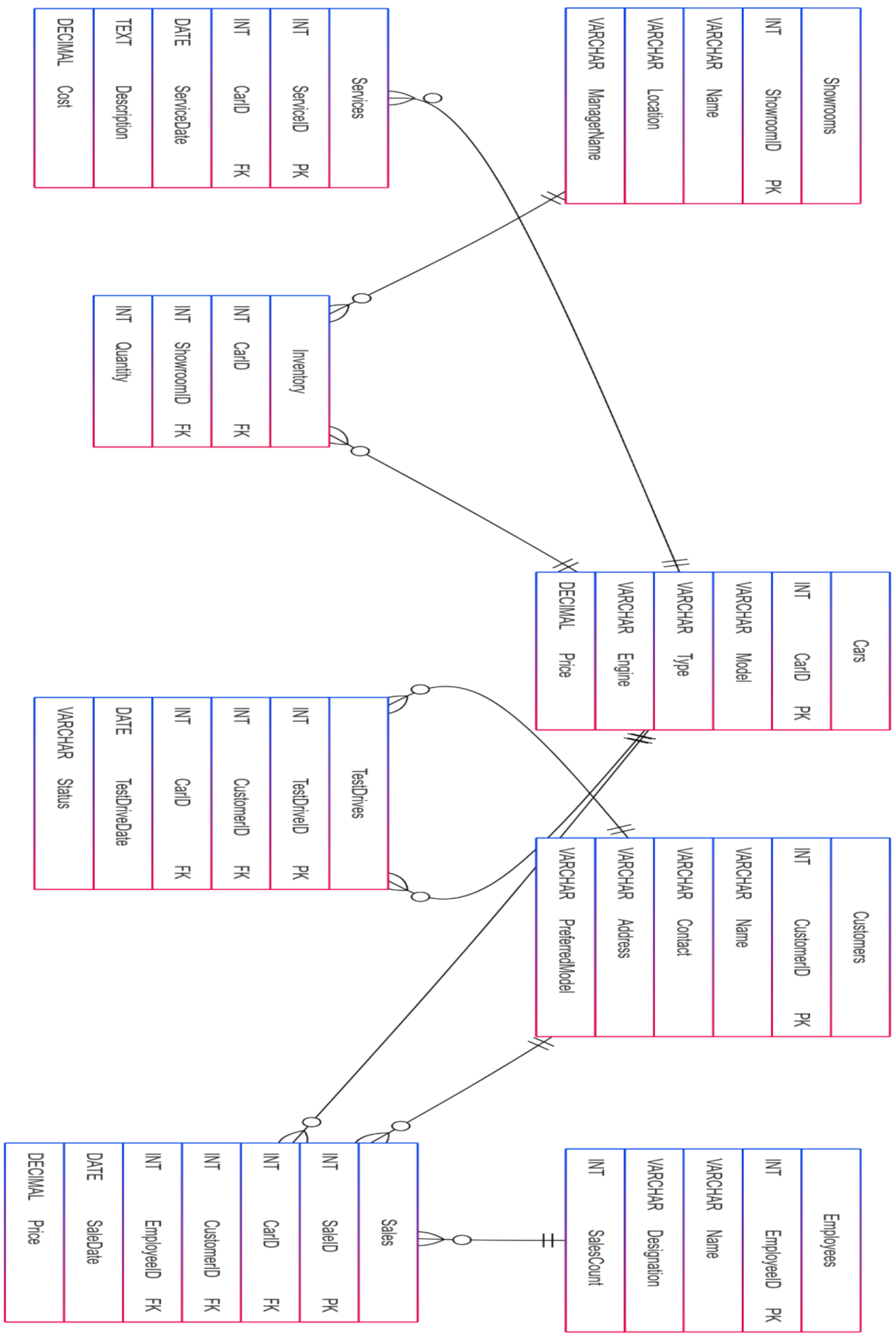


TABLE creation:

```
-- 1. Showrooms Table
CREATE TABLE Showrooms (
    ShowroomID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Location VARCHAR(100) NOT NULL,
    ManagerName VARCHAR(50)
);

-- 2. Cars Table
CREATE TABLE Cars (
    CarID INT PRIMARY KEY,
    Model VARCHAR(50) NOT NULL,
    Type VARCHAR(30) NOT NULL,
    Engine VARCHAR(30) NOT NULL,
    Price DECIMAL(10,2) NOT NULL
);

-- 3. Inventory Table
CREATE TABLE Inventory (
    CarID INT,
    ShowroomID INT,
    Quantity INT NOT NULL,
    PRIMARY KEY (CarID, ShowroomID),
    FOREIGN KEY (CarID) REFERENCES Cars(CarID),
    FOREIGN KEY (ShowroomID) REFERENCES
Showrooms(ShowroomID)
);

-- 4. Customers Table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Contact VARCHAR(20),
    Address VARCHAR(100),
    PreferredModel VARCHAR(50)
);

-- 5. Employees Table
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Designation VARCHAR(30),
    SalesCount INT DEFAULT 0
);
```

```
-- 6. Sales Table
CREATE TABLE Sales (
    SaleID INT PRIMARY KEY,
    CarID INT,
    CustomerID INT,
    EmployeeID INT,
    SaleDate DATE,
    Price DECIMAL(10,2),
    FOREIGN KEY (CarID) REFERENCES Cars(CarID),
    FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID),
    FOREIGN KEY (EmployeeID) REFERENCES
Employees(EmployeeID)
);

-- 7. Test Drives Table
CREATE TABLE TestDrives (
    TestDriveID INT PRIMARY KEY,
    CustomerID INT,
    CarID INT,
    TestDriveDate DATE,
    Status VARCHAR(20),
    FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID),
    FOREIGN KEY (CarID) REFERENCES Cars(CarID)
);

-- 8. Services Table
CREATE TABLE Services (
    ServiceID INT PRIMARY KEY,
    CarID INT,
    ServiceDate DATE,
    Description TEXT,
    Cost DECIMAL(10,2),
    FOREIGN KEY (CarID) REFERENCES Cars(CarID)
);



-- DATA INSERTIONS

-- Showrooms
INSERT INTO Showrooms VALUES
```

```sql
(1, 'Mahindra AutoZone', 'Mumbai', 'Rohit Sharma'),
(2, 'Mahindra MaxDrive', 'Delhi', 'Anjali Mehta'),
(3, 'Mahindra Elite Motors', 'Bangalore', 'Ravi Verma'),
(4, 'Mahindra SpeedHub', 'Chennai', 'Sunita Rao'),
(5, 'Mahindra DriveX', 'Pune', 'Aman Kapoor'),
(6, 'Mahindra ApexAuto', 'Hyderabad', 'Pooja Yadav'),
(7, 'Mahindra RapidCars', 'Ahmedabad', 'Kunal Bhatia'),
(8, 'Mahindra Zoom Motors', 'Jaipur', 'Neeraj Thakur'),
(9, 'Mahindra KingAuto', 'Lucknow', 'Riya Sinha'),
(10, 'Mahindra StarDrive', 'Kolkata', 'Saurav Ghosh');

-- Cars
INSERT INTO Cars VALUES
(1, 'Thar', 'SUV', 'Diesel', 1500000.00),
(2, 'XUV700', 'SUV', 'Petrol', 2000000.00),
(3, 'Scorpio-N', 'SUV', 'Diesel', 1800000.00),
(4, 'Bolero', 'MPV', 'Diesel', 1000000.00),
(5, 'XUV300', 'Compact SUV', 'Petrol', 1300000.00),
(6, 'Marazzo', 'MPV', 'Diesel', 1400000.00),
(7, 'Alturas G4', 'Luxury SUV', 'Diesel', 2700000.00),
(8, 'KUV100 NXT', 'Micro SUV', 'Petrol', 700000.00),
(9, 'eXUV300', 'Electric SUV', 'Electric', 1800000.00),
(10, 'BE.05', 'Concept EV', 'Electric', 2500000.00);
```

```
-- Inventory
INSERT INTO Inventory VALUES
(1, 1, 10),
(2, 2, 8),
(3, 3, 7),
(4, 4, 6),
(5, 5, 12),
(6, 6, 4),
(7, 7, 3),
(8, 8, 9),
(9, 9, 5),
(10, 10, 6);

-- Customers
INSERT INTO Customers VALUES
(1, 'Akash Malhotra', '9876543210', 'New Delhi', 'XUV700'),
(2, 'Sneha Reddy', '9823456789', 'Hyderabad', 'Thar'),
(3, 'Rahul Jain', '9012345678', 'Mumbai', 'Scorpio-N'),
(4, 'Priya Nair', '9567890123', 'Chennai', 'XUV300'),
(5, 'Karan Mehta', '9234567890', 'Pune', 'Bolero'),
(6, 'Ravneet Singh', '9812345670', 'Ludhiana', 'Alturas G4'),
(7, 'Megha Bansal', '9801234567', 'Jaipur', 'Marazzo'),
```

```sql
(8, 'Vikas Anand', '9786543210', 'Bangalore', 'BE.05'),
(9, 'Tanvi Kapoor', '9767890123', 'Ahmedabad', 'KUV100 NXT'),
(10, 'Siddharth Das', '9745678901', 'Kolkata', 'eXUV300');

-- Employees
INSERT INTO Employees VALUES
(1, 'Manoj Kumar', 'Sales Executive', 5),
(2, 'Divya Sharma', 'Sales Manager', 10),
(3, 'Nikhil Joshi', 'Support Staff', 3),
(4, 'Anita Das', 'Sales Executive', 6),
(5, 'Rajeev Kapoor', 'Technician', 2),
(6, 'Deepika Rani', 'Customer Relations', 4),
(7, 'Sandeep Chauhan', 'Sales Executive', 5),
(8, 'Arjun Verma', 'Inventory Manager', 1),
(9, 'Bhavna Tyagi', 'Finance Officer', 0),
(10, 'Mohit Yadav', 'Technician', 2);

-- Sales
INSERT INTO Sales VALUES
(1, 1, 1, 1, '2025-01-10', 1500000.00),
(2, 2, 2, 2, '2025-02-12', 2000000.00),
(3, 3, 3, 3, '2025-03-15', 1800000.00),
(4, 4, 4, 4, '2025-04-18', 1000000.00),
(5, 5, 5, 5, '2025-05-20', 1300000.00),
(6, 6, 6, 6, '2025-06-22', 1400000.00),
(7, 7, 7, 7, '2025-07-25', 2700000.00),
(8, 8, 8, 8, '2025-08-10', 2500000.00),
(9, 9, 9, 9, '2025-09-14', 700000.00),
(10, 10, 10, 10, '2025-10-05', 1800000.00);

-- Test Drives
INSERT INTO TestDrives VALUES
(1, 1, 1, '2025-01-05', 'Completed'),
(2, 2, 2, '2025-02-10', 'Pending'),
(3, 3, 3, '2025-03-10', 'Completed'),
(4, 4, 4, '2025-04-12', 'Completed'),
(5, 5, 5, '2025-05-15', 'Completed'),
(6, 6, 6, '2025-06-01', 'Completed'),
(7, 7, 7, '2025-07-03', 'Pending'),
(8, 8, 8, '2025-08-08', 'Completed'),
(9, 9, 9, '2025-09-12', 'Completed'),
(10, 10, 10, '2025-10-01', 'Pending');

-- Services
INSERT INTO Services VALUES
(1, 1, '2025-01-25', 'Oil Change and Engine Check', 3000.00),
(2, 2, '2025-02-28', 'AC Service', 2500.00),
(3, 3, '2025-03-30', 'Full Service', 5000.00),
(4, 4, '2025-04-25', 'Battery Replacement', 3500.00),
(5, 5, '2025-05-28', 'Brake Pad Replacement', 2800.00),
(6, 6, '2025-06-15', 'Suspension Check', 4000.00),
(7, 7, '2025-07-10', 'Interior Cleaning', 1500.00),
(8, 8, '2025-08-18', 'Tire Alignment', 2200.00),
(9, 9, '2025-09-21', 'Software Update', 1200.00),
(10, 10, '2025-10-10', 'General Inspection', 1800.00);
```

Observation

The Mahindra Showroom database is well-structured, capturing key relationships between showrooms, cars, customers, employees, and services. Each table is designed with clear primary and foreign keys, ensuring data integrity. One-to-many relationships are commonly used, such as one showroom holding many cars or one employee handling many sales. The use of foreign keys like CarID, CustomerID, and ShowroomID efficiently links tables like Inventory, Sales, and TestDrives. This structure allows for effective tracking of sales, services, and customer interactions. Additionally, data insertion reflects real-world scenarios, enabling practical analysis through SQL queries. The schema supports easy expansion and provides a solid foundation for generating insights into performance, customer preferences, and inventory distribution.

Limitation :

 No Login or User Role System – There's no table for managing different user roles (admin, employee, customer) or secure login/authentication.

 No Service Center or Technician Mapping – Services are linked only to cars, not to specific technicians or service centers.

 No Date of Purchase in Customer Table – The customer table doesn’t record when the customer made their first purchase.

 No Car Availability Status – The availability (in stock, out of stock) of cars isn’t explicitly tracked.

 Limited Employee Details – Employee contact, joining date, and department aren’t stored.

 No Payment or Billing Records – There is no dedicated table for tracking payment methods, receipts, or billing details.

CONCLUSION:

The Mahindra Showroom database offers a strong foundation for managing core operations of a car dealership network. It efficiently handles data related to showrooms, cars, inventory, customers, employees, sales, test drives, and services. The relationships between tables are well-defined using primary and foreign keys, ensuring data consistency and ease of access. One-to-many relationships dominate the structure, which reflects real-world scenarios like one showroom having multiple cars or one employee handling many sales. The database supports important operations such as tracking vehicle availability, recording sales and services, and monitoring customer preferences and interactions.

However, despite its strengths, the database has some limitations. It lacks advanced features such as user authentication, detailed employee information, and service technician assignments. There’s also no provision for tracking payments, invoices, or car status (like booked or available). These features are essential for a more complete dealership management system.

In conclusion, while the current schema is suitable for basic operations and analysis, it would benefit from further enhancements to support real-time business needs, improve decision-making, and offer better customer experience. Adding modules for billing, user roles, and detailed service tracking would significantly improve the system's efficiency, scalability, and usability in a real-world business environment.

SUMMARY:

The Mahindra Showroom database is a structured and relational system designed to manage the core functionalities of a multi-branch car dealership. It comprises eight key tables: Showrooms, Cars, Inventory, Customers, Employees, Sales, TestDrives, and Services. Each table is well-defined with appropriate primary keys and foreign key constraints, ensuring logical data flow and integrity across the system.

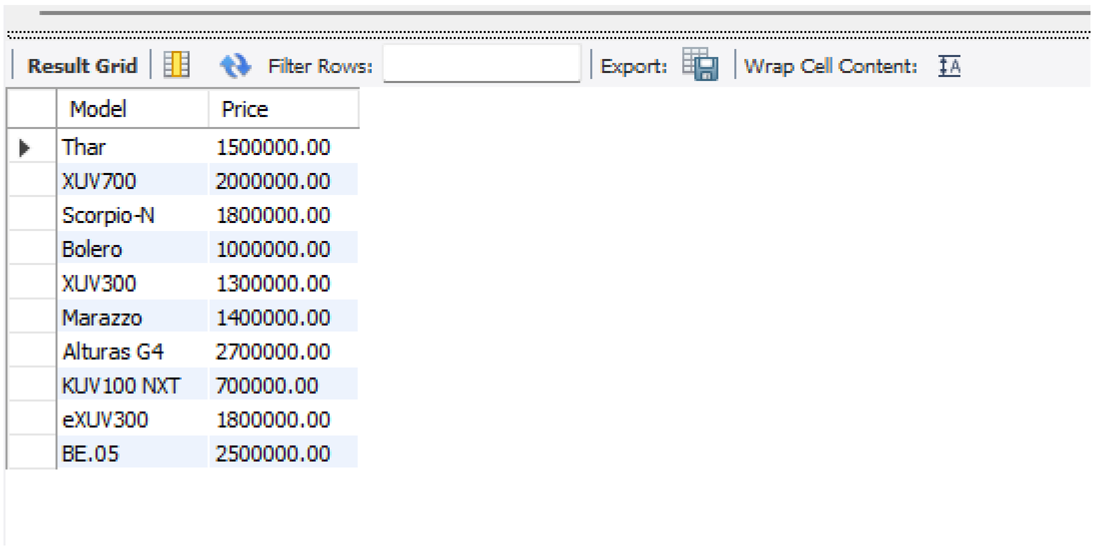
The relationships between tables are mostly one-to-many, such as one showroom managing many cars through inventory, or one employee handling many sales. The Inventory table acts as a junction to track car availability across showrooms, while Sales records capture transaction details by linking customers, employees, and car models. Similarly, TestDrives and Services maintain historical records of customer engagement and post-sale service.

The schema supports various analytical queries like total car stock per showroom, average service cost, top-performing employees, and customer preferences. However, the database lacks some practical features. There's no user authentication system, no detailed technician or payment tracking, and no advanced status tracking for car availability.

In conclusion, this database lays a strong foundation for showroom operations but requires additional modules to support real-time dealership needs, advanced reporting, and enhanced customer service

-- 1) View all car models with prices

SELECT Model, Price FROM Cars;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Model	Price
Thar	1500000.00
XUV700	2000000.00
Scorpio-N	1800000.00
Bolero	1000000.00
XUV300	1300000.00
Marazzo	1400000.00
Alturas G4	2700000.00
KUV100 NXT	700000.00
eXUV300	1800000.00
BE.05	2500000.00

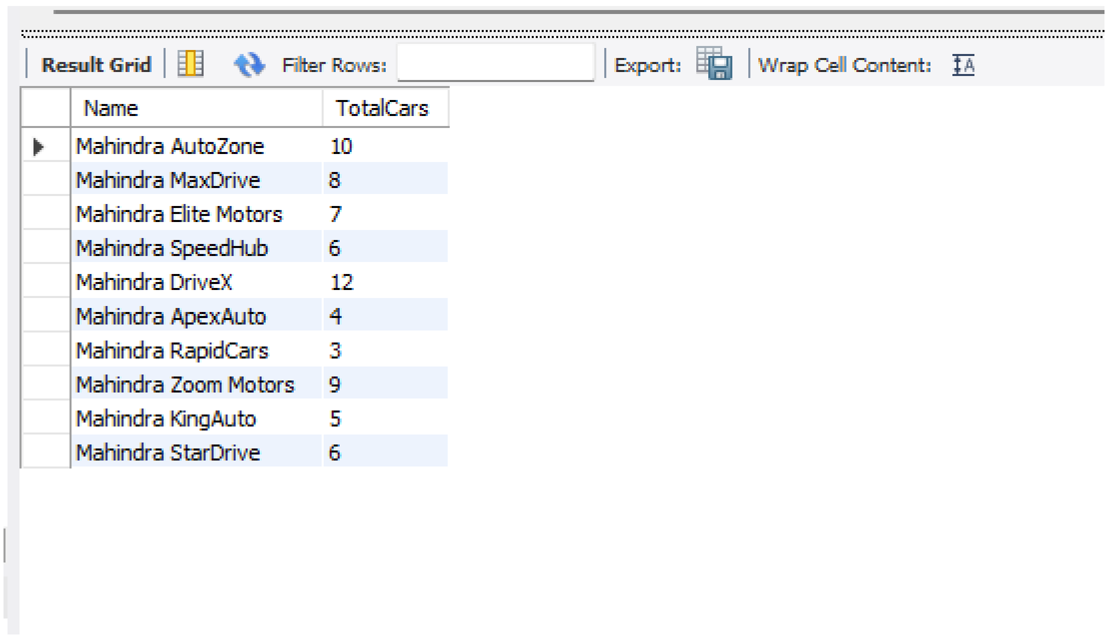
-- 2) View total number of cars in each showroom

SELECT Showrooms.Name, SUM(Inventory.Quantity) AS TotalCars

FROM Inventory

JOIN Showrooms ON Inventory.ShowroomID = Showrooms.ShowroomID

GROUP BY Showrooms.Name;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	TotalCars
Mahindra AutoZone	10
Mahindra MaxDrive	8
Mahindra Elite Motors	7
Mahindra SpeedHub	6
Mahindra DriveX	12
Mahindra ApexAuto	4
Mahindra RapidCars	3
Mahindra Zoom Motors	9
Mahindra KingAuto	5
Mahindra StarDrive	6

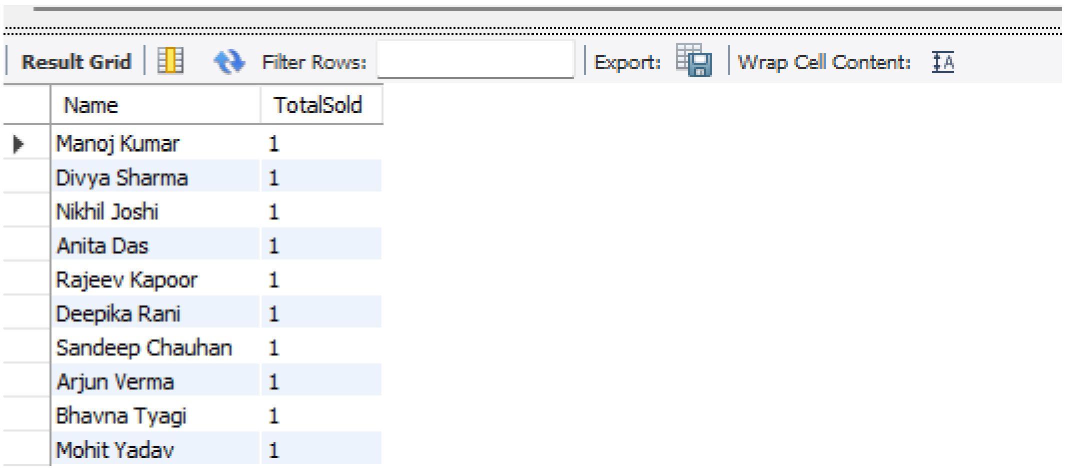
```
-- 3) View cars sold by each employee

SELECT Employees.Name, COUNT(Sales.SaleID) AS TotalSold

FROM Sales

JOIN Employees ON Sales.EmployeeID = Employees.EmployeeID

GROUP BY Employees.Name;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	TotalSold
Manoj Kumar	1
Divya Sharma	1
Nikhil Joshi	1
Anita Das	1
Rajeev Kapoor	1
Deepika Rani	1
Sandeep Chauhan	1
Arjun Verma	1
Bhavna Tyagi	1
Mohit Yadav	1

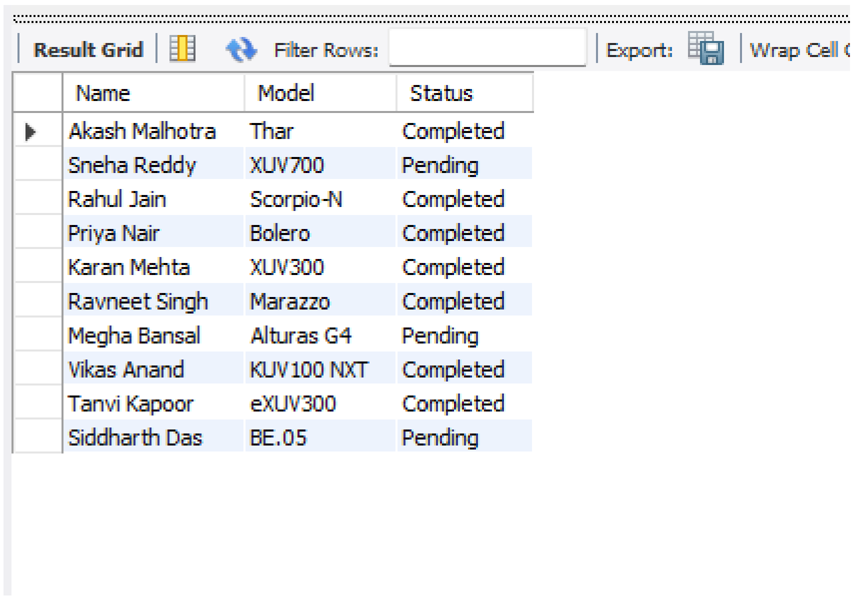
```
-- 4) Retrieve customers who took a test drive with car model

SELECT Customers.Name, Cars.Model, TestDrives.Status

FROM TestDrives

JOIN Customers ON TestDrives.CustomerID = Customers.CustomerID

JOIN Cars ON TestDrives.CarID = Cars.CarID;
```



Result Grid | Filter Rows: | Export: | Wrap Cell

Name	Model	Status
Akash Malhotra	Thar	Completed
Sneha Reddy	XUV700	Pending
Rahul Jain	Scorpio-N	Completed
Priya Nair	Bolero	Completed
Karan Mehta	XUV300	Completed
Ravneet Singh	Marazzo	Completed
Megha Bansal	Alturas G4	Pending
Vikas Anand	KUV100 NXT	Completed
Tanvi Kapoor	eXUV300	Completed
Siddharth Das	BE.05	Pending

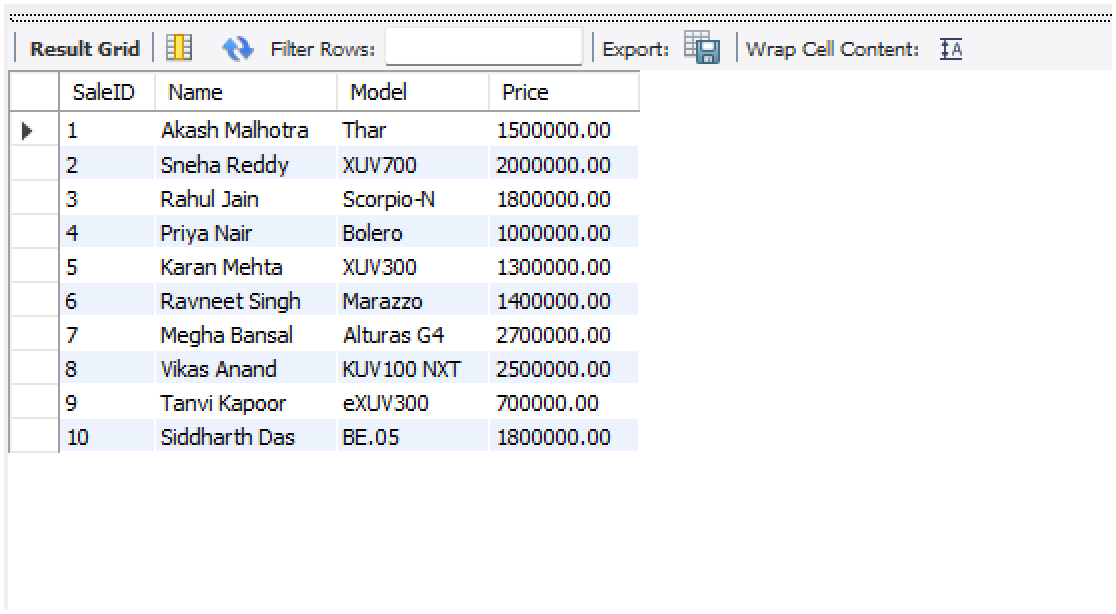
-- 5) Retrieve all sales details with car, customer, and price

SELECT Sales.SaleID, Customers.Name, Cars.Model, Sales.Price

FROM Sales

JOIN Cars ON Sales.CarID = Cars.CarID

JOIN Customers ON Sales.CustomerID = Customers.CustomerID;



SaleID	Name	Model	Price
1	Akash Malhotra	Thar	1500000.00
2	Sneha Reddy	XUV700	2000000.00
3	Rahul Jain	Scorpio-N	1800000.00
4	Priya Nair	Bolero	1000000.00
5	Karan Mehta	XUV300	1300000.00
6	Ravneet Singh	Marazzo	1400000.00
7	Megha Bansal	Alturas G4	2700000.00
8	Vikas Anand	KUV100 NXT	2500000.00
9	Tanvi Kapoor	eXUV300	700000.00
10	Siddharth Das	BE.05	1800000.00

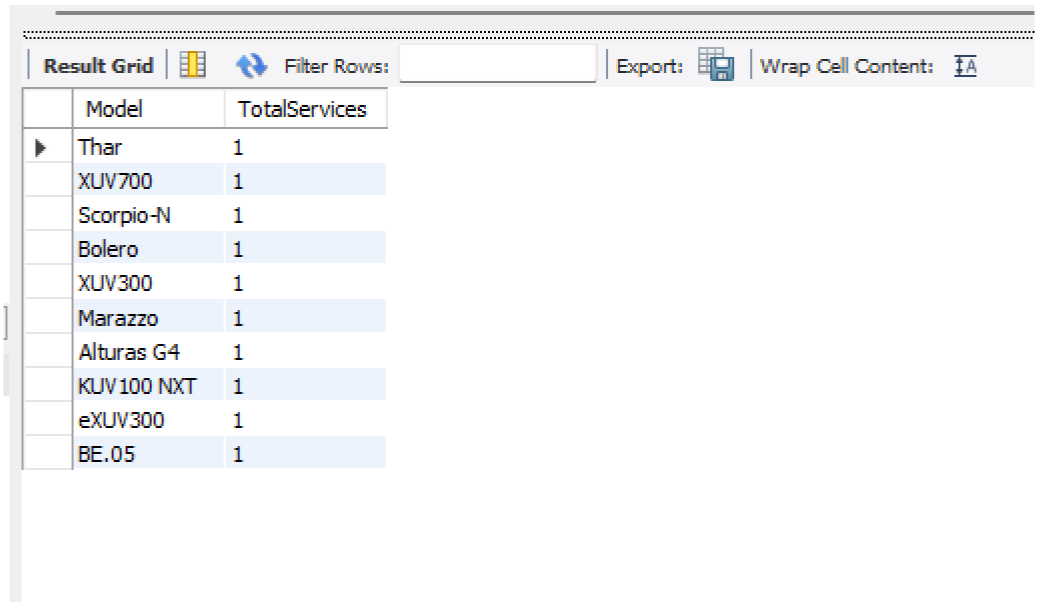
-- 6) Count services done per car

SELECT Cars.Model, COUNT(Services.ServiceID) AS TotalServices

FROM Services

JOIN Cars ON Services.CarID = Cars.CarID

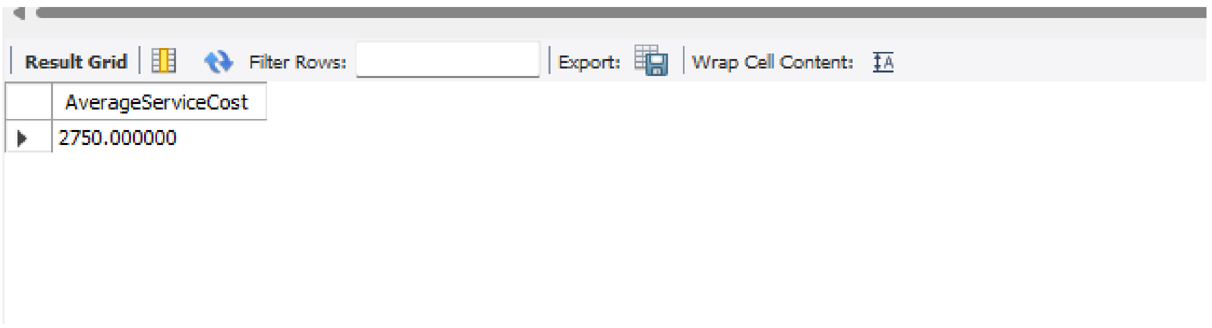
GROUP BY Cars.Model;


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Model	TotalServices
Thar	1
XUV700	1
Scorpio-N	1
Bolero	1
XUV300	1
Marazzo	1
Alturas G4	1
KUV100 NXT	1
eXUV300	1
BE.05	1

-- 7) Average cost of services

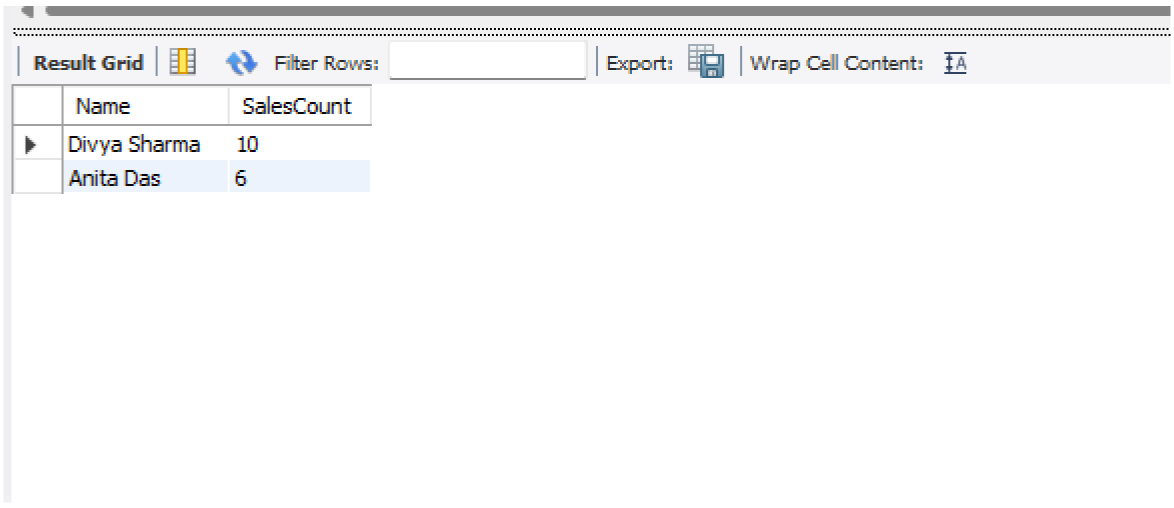
SELECT AVG(Cost) AS AverageServiceCost FROM Services;


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AverageServiceCost
2750.000000

-- 8) Get all employees with more than 5 sales

SELECT Name, SalesCount FROM Employees WHERE SalesCount > 5;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	SalesCount
Divya Sharma	10
Anita Das	6

-- 9) Showrooms having more than 8 cars in stock

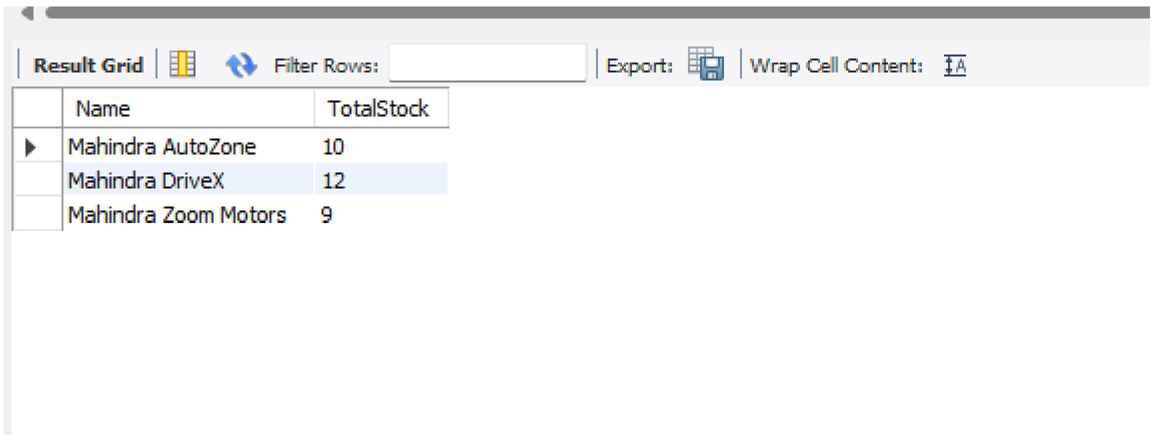
SELECT Showrooms.Name, SUM(Inventory.Quantity) AS TotalStock

FROM Inventory

JOIN Showrooms ON Inventory.ShowroomID = Showrooms.ShowroomID

GROUP BY Showrooms.Name

HAVING SUM(Inventory.Quantity) > 8;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	TotalStock
Mahindra AutoZone	10
Mahindra DriveX	12
Mahindra Zoom Motors	9

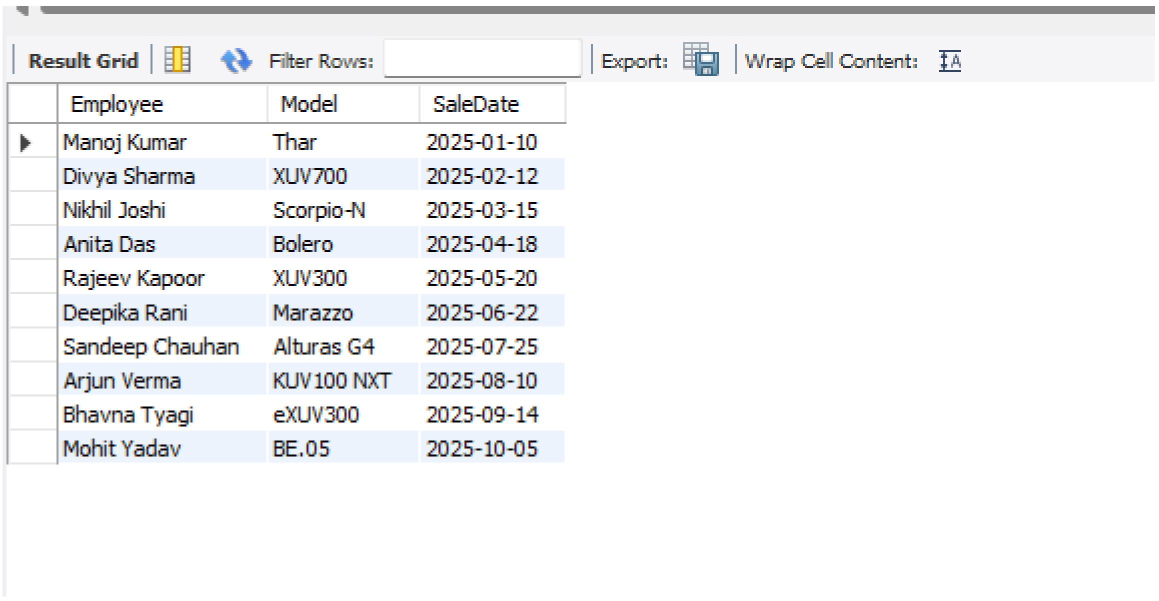
-- 10) INNER JOIN: Sales with employee and car

SELECT Employees.Name AS Employee, Cars.Model, Sales.SaleDate

FROM Sales

JOIN Employees ON Sales.EmployeeID = Employees.EmployeeID

JOIN Cars ON Sales.CarID = Cars.CarID;



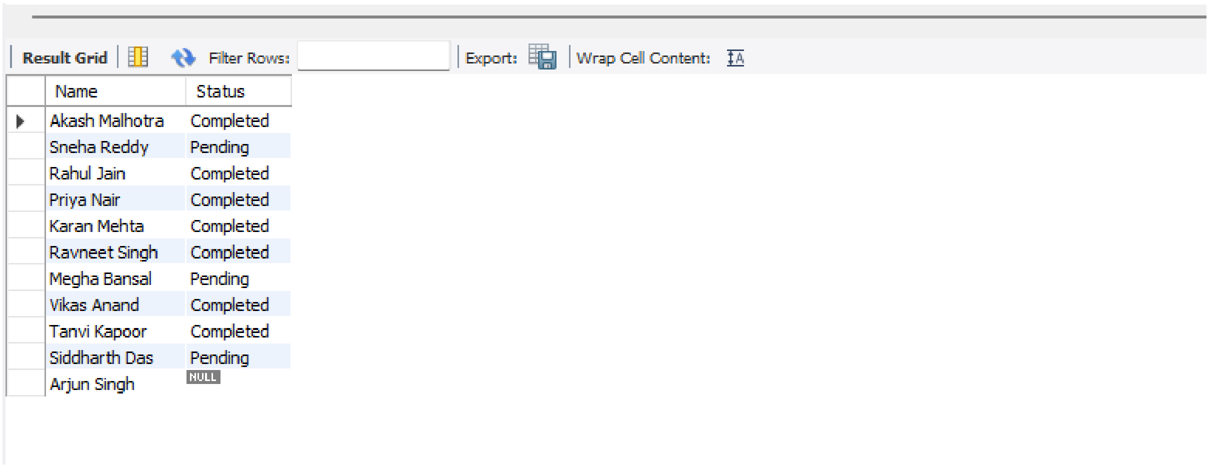
Employee	Model	SaleDate
Manoj Kumar	Thar	2025-01-10
Divya Sharma	XUV700	2025-02-12
Nikhil Joshi	Scorpio-N	2025-03-15
Anita Das	Bolero	2025-04-18
Rajeev Kapoor	XUV300	2025-05-20
Deepika Rani	Marazzo	2025-06-22
Sandeep Chauhan	Alturas G4	2025-07-25
Arjun Verma	KUV100 NXT	2025-08-10
Bhavna Tyagi	eXUV300	2025-09-14
Mohit Yadav	BE.05	2025-10-05

-- 11) LEFT JOIN: All customers even if they didn't take a test drive

SELECT Customers.Name, TestDrives.Status

FROM Customers

LEFT JOIN TestDrives ON Customers.CustomerID = TestDrives.CustomerID;



Name	Status
Akash Malhotra	Completed
Sneha Reddy	Pending
Rahul Jain	Completed
Priya Nair	Completed
Karan Mehta	Completed
Ravneet Singh	Completed
Megha Bansal	Pending
Vikas Anand	Completed
Tanvi Kapoor	Completed
Siddharth Das	Pending
Arjun Singh	NULL

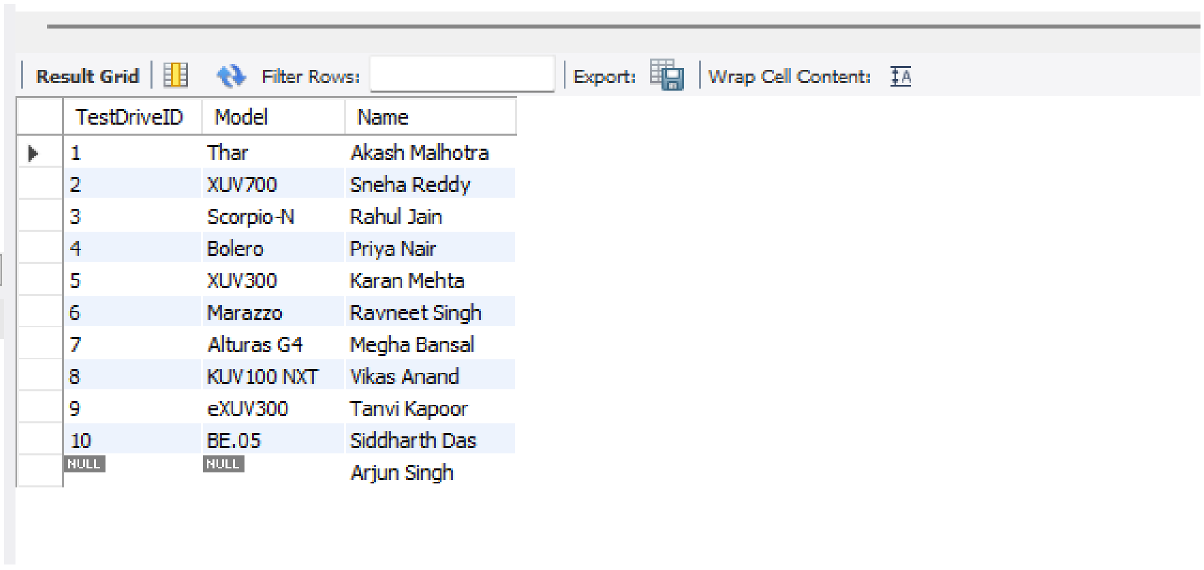
-- 12) RIGHT JOIN: All test drives and their customers

SELECT TestDrives.TestDriveID, Cars.Model, Customers.Name

FROM TestDrives

RIGHT JOIN Cars ON TestDrives.CarID = Cars.CarID

RIGHT JOIN Customers ON TestDrives.CustomerID = Customers.CustomerID;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

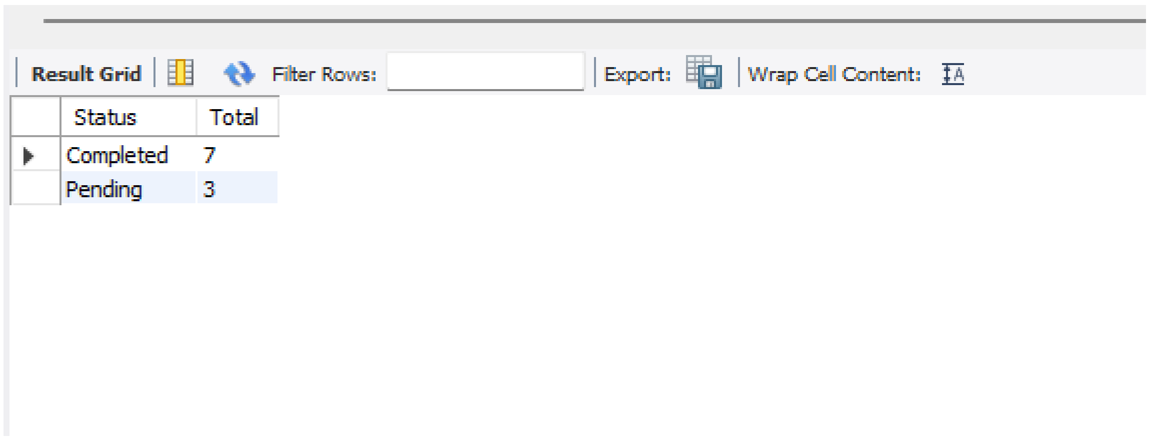
	TestDriveID	Model	Name
▶	1	Thar	Akash Malhotra
	2	XUV700	Sneha Reddy
	3	Scorpio-N	Rahul Jain
	4	Bolero	Priya Nair
	5	XUV300	Karan Mehta
	6	Marazzo	Ravneet Singh
	7	Alturas G4	Megha Bansal
	8	KUV100 NXT	Vikas Anand
	9	eXUV300	Tanvi Kapoor
	10	BE.05	Siddharth Das
	NULL	NULL	Arjun Singh

-- 13) Group by test drive status

SELECT Status, COUNT(*) AS Total

FROM TestDrives

GROUP BY Status;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Status	Total
▶	Completed	7
	Pending	3