Project Report

Vehicle Rental System(MySQL Only)

1. Introduction

*In today's fast-paced world, rental services are essential for both individuals and businesses. This project focuses on designing a **Vehicle Rental System** using **MySQL**, a relational database management system. The system replicates core operations typically seen in commercial car/bike rental services, with emphasis on **database modeling**, **data integrity**, and **modular query execution**.*

This project omits frontend and backend components to concentrate solely on database logic, schema design, normalization principles, and performance-optimized queries that replicate a fully functional vehicle rental service backend.

2. Objective

*The main objective of this project is to implement a **relational database system** that can:*

- *Maintain and manage details of vehicles, branches, customers, and staff.*
- *Process rental bookings and prevent double-booking conflicts.*
- *Track payments, damages, discounts, and promotional campaigns.*
- *Generate comprehensive reports that support operational decisions.*

This system simulates the backend of a commercial vehicle rental platform, providing a strong foundation for future integration with web or mobile applications.

3. System Requirements

Software

- ***MySQL Version:** 5.7 or above*
- ***Client Interface:** MySQL Workbench / phpMyAdmin / Terminal*

Optional Tools

- *DB Diagram Designer (like dbdiagram.io or MySQL Workbench)*
- *Document processor for report formatting (MS Word, LaTeX, etc.)*

Constraints

- *Must follow normalization standards (3NF minimum)*
- *Enforce primary and foreign key integrity*
- *No use of server-side or client-side scripting*

4. Database Design Overview

*The database consists of **nine core entities**, each fulfilling a distinct role in the system. The schema is normalized and built to support extendability, ensuring minimal redundancy and high data integrity.*

Table	*Description*
Branches	*Details of different rental locations (e.g., city, area, manager info)*
Customers	*Personal and contact details, unique email and phone constraint*
Vehicles	*Includes make, model, year, type (car/bike), current status, and branch*
Bookings	*Rental transactions with start/end dates, linked to customer and vehicle*
Payments	*Tracks transaction details like amount, status, and payment method*

Damages	*Incident logging with severity, resolution status, and dates*
Promotions	*Discount codes with validity range and discount percentages*
Maintenance	*Scheduled and completed maintenance records for each vehicle*
Pricing	*Hourly and daily rate setup for each vehicle type*

5. Schema Implementation

The `schema.sql` file includes:

- ***DDL Commands** to create all tables with relevant attributes and data types.*
- ***Constraints**: Primary keys, foreign keys, `CHECK`, `ENUM`, and `NOT NULL`.*
- ***Normalization**: The design avoids transitive dependencies and ensures that each field stores only atomic values.*
- ***Scalability**: The schema allows easy addition of new branches, vehicles, and rate adjustments without schema changes.*

Examples:

- *`status ENUM` in `Vehicles` ensures that only allowed values like `'available'`, `'booked'`, and `'maintenance'` are accepted.*
- *Foreign key references between `Bookings → Customers` and `Bookings → Vehicles` ensure relational consistency.*

6. Sample Data

The `sample_data.sql` file simulates real-life data to test all functionalities:

- ***Branches:** 5 diverse locations*
- ***Vehicles:** 8–10 entries (cars and bikes)*
- ***Customers:** Users with realistic contact details*
- ***Bookings & Payments:** Both successful and failed transactions*
- ***Promotions:** Active and expired codes*
- ***Damages & Maintenance:** Simulates wear-and-tear and resolution logs*

.

7. Functional Modules & SQL Operations

*Each module supports **CRUD operations** (Create, Read, Update, Delete) and can be accessed independently via SQL.*

✅ Customers

- *Insert new customers*
- *Fetch customer rental history*
- *Update contact details*
- *Delete inactive users*

✅ Vehicles

- *Add or retire vehicles*
- *Update availability/status*
- *Filter by type, status, or location*
- *Join with pricing for rate display*

✅ *Bookings*

- *Create bookings with validation*
- *Prevent overlapping bookings*
- *View historical and active bookings*
- *Associate with payments and promotions*

✅ *Payments*

- *Insert payment record after booking*
- *Track payment method and status*
- *Join with booking to get payment reports*

✅ *Damage Logs*

- *Insert damage entries (low to critical)*
- *Set resolution status*
- *Link to specific vehicle IDs*

✅ *Reports & Analytics*

- ***Availability Check:*** *Given a date range and type*
- ***Revenue Report:*** *Monthly earnings, branch-wise*
- ***Top Vehicles:*** *Most frequently rented*
- ***Top Customers:*** *Based on total spend*
- ***Promo Usage:*** *Discounted earnings summary*

8. Bonus Features

These value-added modules simulate features from real-world rental systems:

🎟️ *Promotions*

- *Use discount codes during booking*
- *Ensure active date range is respected*
- *Calculate effective billing after discount*

🏢 *Multi-Branch Support*

- *View vehicle inventory branch-wise*
- *Filter availability by location*
- *Track earnings per branch*

🛠️ *Maintenance Schedules*

- *Schedule services every N kilometers or dates*
- *Track maintenance cost per vehicle*
- *View upcoming and overdue maintenance*

📊 *Summary Reports*

- *Export data like:*
    - *Total income for a month*
    - *Most damaged vehicle*
    - *Underutilized vehicles*
    - *Payment methods most used*

9. Setup Instructions

To initialize and run this system on your local MySQL environment:

1. **Run `schema.sql`**
   This creates the complete schema and all constraints.

2. **Run `sample_data.sql`**
   Inserts demo data for realistic testing.

3. **Run `queries.sql`**
   Execute functional and bonus feature queries:

   - *CRUD operations*
   - *Availability checks*
   - *Revenue reports*
   - *Discount application*
   - *Top vehicle and customer reports*

10. Deliverables

File Name	Description
`schema.sql`	*SQL DDL for tables, indexes, constraints*
`sample_data.sql`	*Sample data covering all modules*
`queries.sql`	*SQL for CRUD operations and bonus features*
`Vehicle_Rental_Report.pdf`	*This professional documentation*

`ER_Diagram.png` *(opt)*	*Visual schema diagram for better understanding*

11. Conclusion

*This project highlights the strength of using **MySQL** for designing complex, real-world systems. With features like **normalized schema**, **modular queries**, **data consistency**, and **reporting tools**, this project forms a complete backend for a vehicle rental platform. While no frontend or backend logic is implemented, the project is fully extensible to integrate with web or mobile interfaces.*