*Technical Report for Car Rental System Database Project*

**1. Introduction**

This project aims to design and implement a database system for a Car Rental company. The database stores information about customers, cars, employees, rental contracts, and invoices. The goal is to create an efficient and normalized database that supports basic operations such as renting cars, managing contracts, and tracking payments.

**2. Requirements Analysis**

The client requires a system to handle the following:

Customer information (personal details and driver license info).

Car details (model, type, daily rental price, and availability status).

Employee details (name and job title).

Rental contracts (linking customers, cars, and employees with rental dates and total price).

Invoices for payments related to contracts.

**3. Entity-Relationship Diagram (ERD) Design**

We started by identifying the main entities:

Customers: Represent the people who rent cars.

Cars: Represents vehicles available for rent.

Employees: Staff managing rental contracts and customer service.

Contracts: Records of car rental transactions.

Invoices: Records of payments related to rental contracts.

Each entity has specific attributes. For example, Customers have CustomerID (Primary Key), Name, Phone, Address, and LicenseNumber.

Relationships

One customer can have multiple rental contracts (1:N).

One car can be rented in multiple contracts over time (1:N).

One employee can manage multiple contracts (1:N).

Each contract has one invoice (1:1).

The ERD was drawn using standard notation with entities as rectangles, attributes as ovals, primary keys underlined, and relationships showing cardinality.

**4. Mapping ERD to Relational Schema**

Each entity translates into a database table with attributes as columns:

Primary keys ensure unique identification of records.

Foreign keys maintain relationships between tables, such as CustomerID in Contracts referencing Customers.

For example, the Contracts table includes foreign keys CustomerID, CarID, and EmployeeID referencing their respective tables.

**5. Normalization**

The database design follows normalization rules to eliminate redundancy and ensure data integrity. Each table represents one entity or relationship, and no redundant data is stored across tables.

6. SQL Implementation

We wrote SQL scripts to:

Create the database and tables with the specified columns, primary keys, and foreign keys.

Insert at least 10 rows of sample data into each table to simulate real-world records.

Define user privileges:

A read-only user for data querying.

An admin user with full privileges.

Example SQL snippet for creating the Customers table:

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(50),

Phone VARCHAR(15),

Address VARCHAR(100),

LicenseNumber VARCHAR(20)

);

**7. Sample Queries**

We wrote at least five SQL queries to demonstrate database operations, including:

Text search (e.g., finding customers by name).

Aggregate functions (e.g., calculating total rental prices).

Sorting results in ascending or descending order.

Joining tables to retrieve combined information such as contracts with customer and car details.

**8. Conclusion**

This project successfully designed and implemented a relational database system for a car rental service, covering all essential requirements. The database is well-structured, normalized, and includes sample data and user access controls. The system can be extended with additional features like maintenance tracking or penalty management if needed.