# VEHICLE RENTAL SERVICE DATABASE DESIGN & DEVELOPMENT

**Created By:** Milind Naik

**Guided By:** Dr Junaid Qazi

# Contents

# Introduction

This Article presents a comprehensive database design for a car rental service. The database is meticulously crafted to streamline employee management, reservations, vehicle availability tracking, and customer information handling, ensuring smooth and secure operations. We'll explore the objectives, tables, fields, relationships, and SQL queries essential for running an efficient vehicle rental business.

# Mission

Our mission is to offer reliable, accessible, and secure car rentals with a focus on user privacy and satisfaction.

# Objective

1. **Extensive Vehicle Selection:** Offer a wide range of car rental options to cater to every need and preference, ensuring you have the perfect vehicle for every occasion.

2. **Competitive Pricing:** Attract a diverse customer base by providing rental rates that are reasonable and affordable, making your services accessible to all.

3. **Exceptional Customer Service:** Prioritize excellent support and guidance to ensure a seamless and enjoyable rental experience for every client.

4. **Data-Driven Optimization:** Utilize technology to gain valuable insights that streamline operations and enhance the customer interface, fostering efficiency and convenience.

5. **Secure and Reliable Platform:** Guarantee a safe and dependable system that safeguards client information and maintains the integrity of every transaction, instilling confidence in your services.

# Database Design

## Why Database Design is important?

For the vehicle rental service firm, having a well-designed database is crucial for several reasons aligned with the company's goals of providing a safe, scalable, and efficient platform. Here's why:

1. **Streamlined Data Management**

   **Centralized data storage:** All essential business data, such as customer information, vehicle availability, reservations, payments, and reviews, are organized in a thoughtfully structured database. This centralization enables seamless data processing and querying.

   **Rapid data retrieval:** With optimal indexing and appropriate relationships, the system can instantly fetch relevant information like available vehicles, customer details, and rental history, boosting operational efficiency.

2. **Enhanced Customer Experience**

   **Prompt Responses:** When booking a car, verifying availability, or reviewing past rentals, customers expect timely information. A well-structured database enables real-time data updates, ensuring exceptional customer service.

   **Personalized Recommendations:** By effectively managing customer profiles and preferences, the database can offer individualized suggestions, such as preferred car models or rental history. This heightens customer satisfaction and fosters loyalty.

3. **Flexibility and Scalability**

   **Business Growth Support:** A well-designed database can handle increasing data volumes without performance issues as the business expands, whether by adding more locations or serving a larger fleet. This aligns with the company's goal of providing reliable and easily accessible car rentals.

   **Adaptability for Future Changes:** With the right architecture, additional tables or features (such as more payment options or car classifications) can be incorporated into the database without a complete overhaul. Maintaining competitiveness and the ability to adapt in a rapidly changing market requires this flexibility.

4. **Security and Integrity of Data**

   **Customer Data Protection:** Ensuring a dependable and secure system is a major corporate goal. Customer data is shielded from intrusions and breaches using security features including role-based access, encryption, and user authentication in a well-designed database.

   **Transaction Integrity:** The database ensures data integrity by using constraints and foreign keys appropriately. For instance, it guarantees that payments are accurately matched to finished transactions and that bookings are associated with legitimate clients and automobiles.

5. **Technology-Based Perspectives**

   **Data Insights and Analytics:** The business can monitor crucial performance indicators like revenue, client feedback, and vehicle usage with a relational database. Using this data, the company can identify high-demand car models, refine pricing strategies, and streamline operations to meet technology-driven insight goals.

   **Monitoring Performance:** The database provides valuable information on fleet performance, consumer preferences, and operational efficiency. This data enables the business to make informed decisions that enhance customer satisfaction and boost profitability.

6. **Streamlined Booking Experience**

   **Seamless Reservation Process:** The car rental system offers an intuitive booking interface where customers can effortlessly reserve vehicles and access real-time information on availability, pricing, and rental status. This user-friendly approach directly contributes to customer satisfaction and cost-effectiveness.

   **Preventing Conflicts:** The database's well-designed relationships significantly reduce the risk of double bookings or pricing discrepancies, ensuring smooth operations.

7. **Facilitating Regulatory Compliance**

   **Meticulous Record-Keeping:** The system securely stores rental agreements, client contracts, and transaction records. This comprehensive documentation ensures accurate data is available for audits or disputes, helping the business adhere to legal and regulatory requirements.

# Database Tables and Fields

The following are the key tables and their fields in the database:

## 1. Customer Table

user_id (INT) - PRIMARY KEY, AUTO_INCREMENT

first_name (VARCHAR (50)) - NOT NULL

last_name (VARCHAR (50)) - NOT NULL

email (VARCHAR (100)) - UNIQUE, NOT NULL

password (VARCHAR (255)) - NOT NULL

phone_number (VARCHAR (20)) - Optional

address (TEXT) - Optional

created_at (TIMESTAMP) - DEFAULT CURRENT_TIMESTAMP

## 2. Vehicles Table

vehicle_id (INT) - PRIMARY KEY, AUTO_INCREMENT

model (VARCHAR (50)) - NOT NULL

year (INT) - NOT NULL

vehicle_type (VARCHAR (50)) - NOT NULL

daily_rate (DECIMAL (10, 2)) - NOT NULL

availability_status (VARCHAR (20)) - NOT NULL

branch_id (INT) - FOREIGN KEY REFERENCES Branches(branch_id)

insurance_id (INT) - FOREIGN KEY REFERENCES VehicleInsurance(insurance_id)

## 3. Reservations Table

reservation_id (INT) - PRIMARY KEY, AUTO_INCREMENT

user_id (INT) - FOREIGN KEY REFERENCES Customer(user_id)

vehicle_id (INT) - FOREIGN KEY REFERENCES Vehicles(vehicle_id)

reservation_date (DATE) - NOT NULL

start_date (DATE) - NOT NULL

end_date (DATE) - NOT NULL

total_price (DECIMAL (10, 2)) - NOT NULL

status (VARCHAR (20)) - NOT NULL

## 4. Branches

branch_id (INT)- PRIMARY KEY, AUTO_INCREMENT

branch_name (VARCHAR (100)),

address (TEXT),

city (VARCHAR (50))

state (VARCHAR (50))

zip_code (VARCHAR (10))

phone_number (VARCHAR (20))

email (VARCHAR (100))

## 5. Vehicle Reviews

review_id (INT)- PRIMARY KEY, AUTO_INCREMENT

user_id (INT) - FOREIGN KEY REFERENCES Customer(user_id)

vehicle_id (INT) - FOREIGN KEY REFERENCES Vehicles(vehicle_id)

review_rating (INT)

review_text (TEXT)

review_date -TIMESTAMP DEFAULT CURRENT_TIMESTAMP

## 6. Accident Reports

report_id (INT) - PRIMARY KEY, AUTO_INCREMENT

reservation_id (INT) - FOREIGN KEY, REFERENCES Reservations(reservation_id)

vehicle_id (INT) - FOREIGN KEY, REFERENCES Vehicles(vehicle_id)

insurance_id (INT) - FOREIGN KEY, REFERENCES VehicleInsurance(insurance_id)

accident_date (DATE) - NOT NULL

description (TEXT) – Optional

damage_cost (DECIMAL (10, 2)) - Optional

is_resolved (BOOLEAN) - DEFAULT FALSE

### 7. Employee

employee_id (INT) - PRIMARY KEY

first_name (VARCHAR (50)) - NOT NULL

last_name (VARCHAR (50)) - NOT NULL

branch_id (INT) - FOREIGN KEY, REFERENCES Branches(branch_id)

### 8. Vehicle Insurance

insurance_id (INT) - PRIMARY KEY, AUTO_INCREMENT

provider (VARCHAR (100)) - NOT NULL

policy_number (VARCHAR (50)) - NOT NULL

coverage_amount (DECIMAL (12, 2)) - NOT NULL

valid_from (DATE) - NOT NULL

valid_until (DATE) - NOT NULL

# Database Table Relationships

## Why Database Tables Relationship is important?

Relationships are essential for databases because they define the connections between data stored in different tables, ensuring data consistency and integrity. By separating similar data into multiple tables and enabling complex queries to retrieve related information efficiently, they reduce data redundancy. Establishing relationships also improves scalability, ensuring that the system can grow while maintaining structured and meaningful data exchanges.

## Summary of All Relationships for Rental Database Tables

- **Customer (1) ↔ Reservations (N)**: user_id in **Reservations** references user_id in **Customer**.
- **Customer (1) ↔ VehicleReviews (N)**: user_id in **VehicleReviews** references user_id in **Customer**.
- **Vehicles (1) ↔ Reservations (N)**: vehicle_id in **Reservations** references vehicle_id in **Vehicles**.
- **Vehicles (1) ↔ VehicleReviews (N)**: vehicle_id in **VehicleReviews** references vehicle_id in **Vehicles**.
- **Vehicles (1) ↔ AccidentReports (N)**: vehicle_id in **AccidentReports** references vehicle_id in **Vehicles**.
- **Branches (1) ↔ Vehicles (N)**: branch_id in **Vehicles** references branch_id in **Branches**.
- **Branches (1) ↔ Employees (N)**: branch_id in **Employees** references branch_id in **Branches**.
- **Reservations (1) ↔ AccidentReports (N)**: rental_id in **AccidentReports** references reservation_id in **Reservations**.
- **VehicleInsurance (1) ↔ Vehicles (N)**: insurance_id in **Vehicles** references insurance_id in **VehicleInsurance**.
- **VehicleInsurance (1) ↔ AccidentReports (N)**: insurance_id in **AccidentReports** references insurance_id in **VehicleInsurance**.

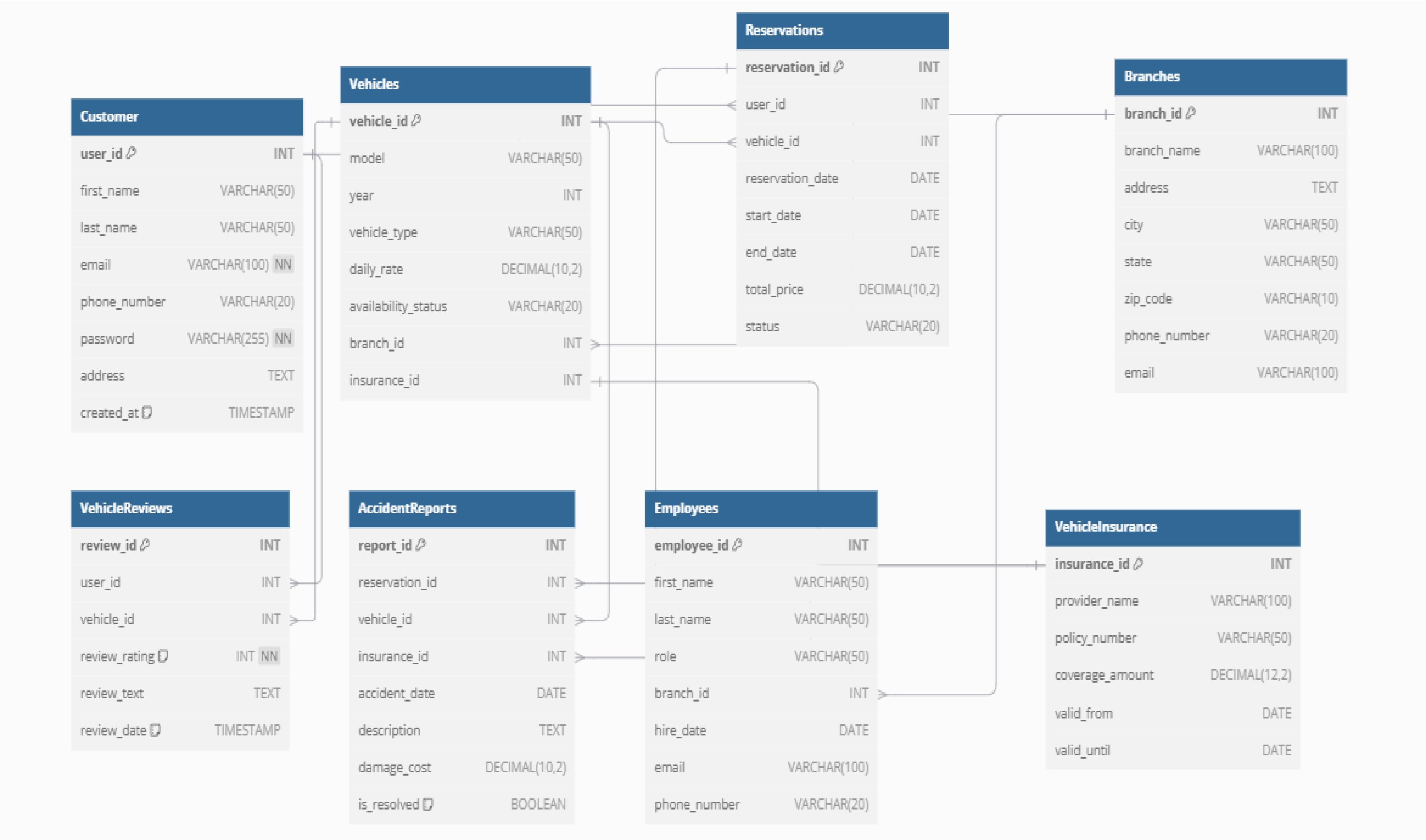
# Entity Relationship Diagram (ER)

## Why Entity Relationship Diagram is Important?

An entity-relationship (ER) diagram is a crucial tool for visualizing the relationships and data structure within a database. It helps stakeholders understand the entities, their attributes, and the connections between them, facilitating effective communication. ER diagrams serve as a blueprint for database design, ensuring optimal structure and data integrity. They simplify database design and maintenance by providing a clear representation of data flows, making it easier to identify and address potential issues or areas for improvement.

## Entity Relationship Diagram within the tables in Database

The ER diagram below illustrates the relationships between the various entities in the vehicle rental system database.



# Database Development

We create a sample database for the rental car system. Below is the sample query run during the development. Moreover, the other queries are mentioned in Appendix B.

**Example:**

**Create Database:**

CREATE DATABASE CarRentalService;

**Create table and Insert query for Customer table:**

1. Customer Table

   CREATE TABLE Customer (

     user_id INT PRIMARY KEY AUTO_INCREMENT,

     first_name VARCHAR(50),

     last_name VARCHAR(50),

     email VARCHAR(100) UNIQUE NOT NULL,

     password VARCHAR(255) NOT NULL,

     phone_number VARCHAR(20),

     address TEXT,

     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

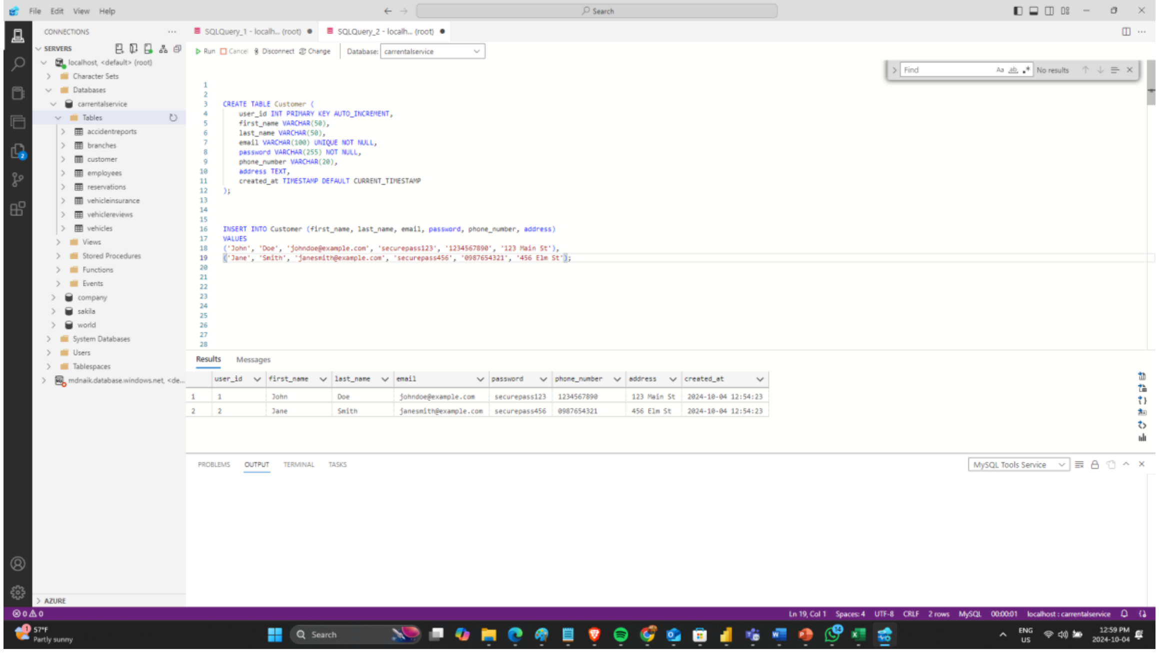
   );

INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)

VALUES

('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),

('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');


SQLQuery_1 - localh... (root)
SQLQuery_2 - localh... (root)
Run
Cancel
Disconnect
Change
Database: carrentalservice
Results
Messages
PROBLEMS
OUTPUT
TERMINAL
TASKS

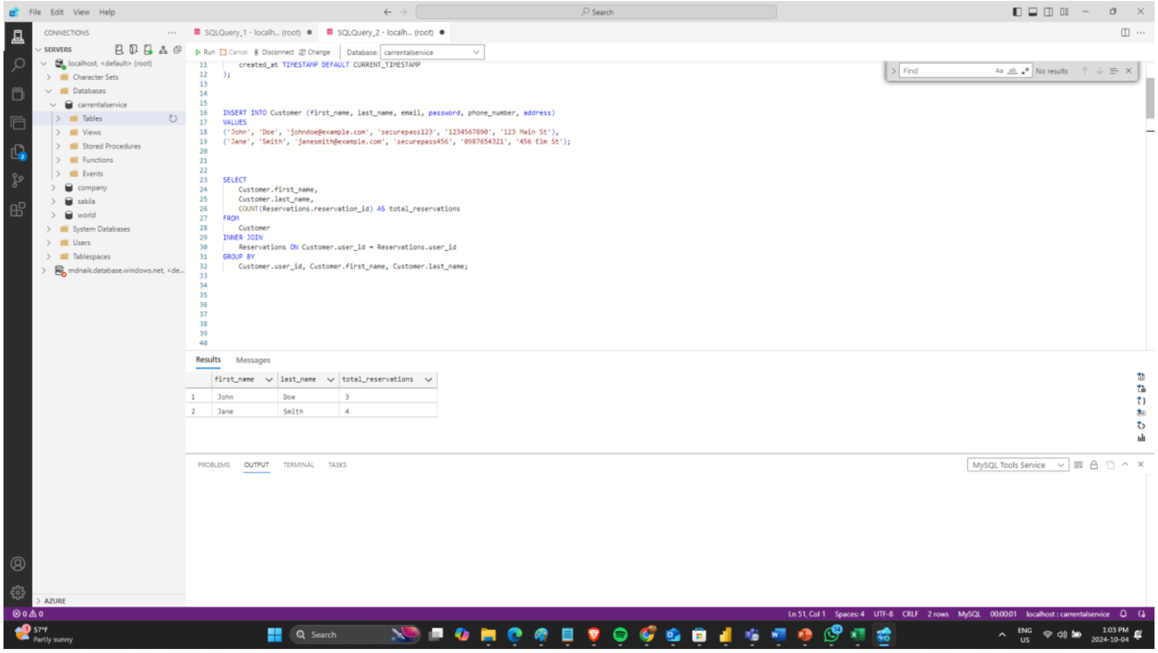
```


CREATE TABLE Customer (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);



INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)
VALUES
('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),
('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');
```

| | user_id | first_name | last_name | email | password | phone_number | address | created_at |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | John | Doe | johndoe@example.com | securepass123 | 1234567890 | 123 Main St | 2024-10-04 12:54:23 |
| 2 | 2 | Jane | Smith | janesmith@example.com | securepass456 | 0987654321 | 456 Elm St | 2024-10-04 12:54:23 |

# Inner Join Query

```
SELECT
    Customer.first_name,
    Customer.last_name,
    COUNT(Reservations.reservation_id) AS total_reservations
FROM
    Customer   INNER JOIN
    Reservations ON Customer.user_id = Reservations.user_id
GROUP BY
    Customer.user_id, Customer.first_name, Customer.last_name;
```

**Explanation:** We are trying to get the number of bookings by customer and displaying their first name and last name by matching the reservation id in reservation table and customer id from customer table.



```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);



INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)
VALUES
('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),
('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');



SELECT
    Customer.first_name,
    Customer.last_name,
    COUNT(Reservations.reservation_id) AS total_reservations
FROM
    Customer
INNER JOIN
    Reservations ON Customer.user_id = Reservations.user_id
GROUP BY
    Customer.user_id, Customer.first_name, Customer.last_name;
```

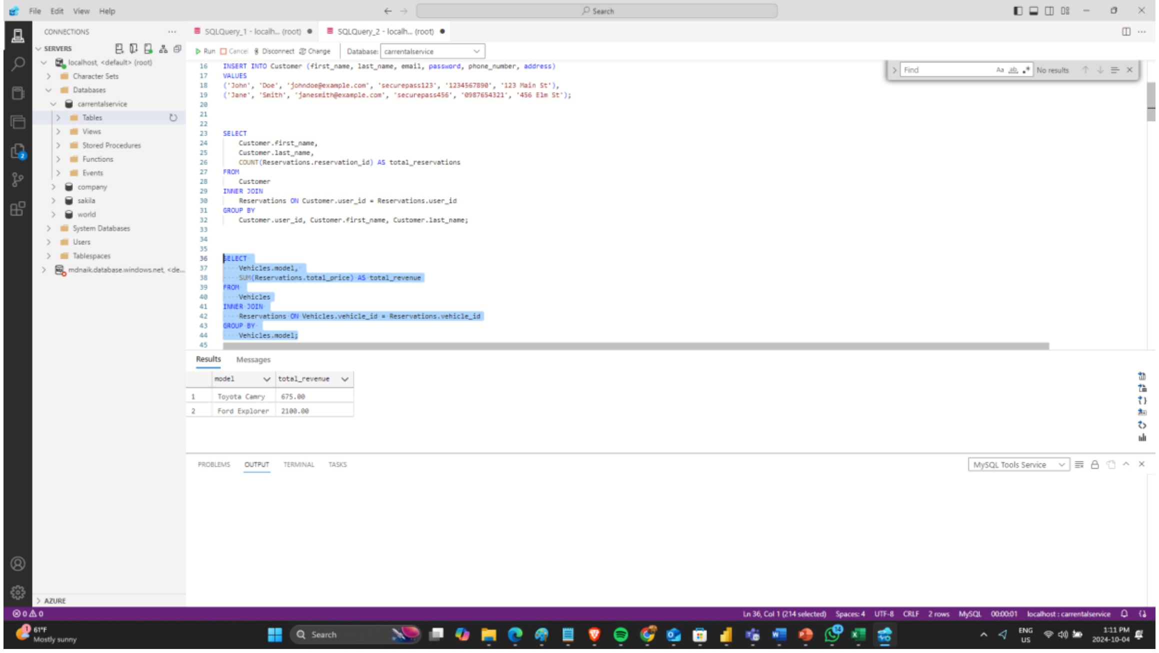
| | first_name | last_name | total_reservations |
|---|---|---|---|
| 1 | John | Doe | 3 |
| 2 | Jane | Smith | 4 |

## Group By Query

```
SELECT
    Vehicles.model,
    SUM(Reservations.total_price) AS total_revenue
FROM
    Vehicles
INNER JOIN
    Reservations ON Vehicles.vehicle_id = Reservations.vehicle_id
GROUP BY
    Vehicles.model;
```

**Explanation:**

We are trying to get the revenue generated by each vehicle using group by.



Database: carrentalservice

```
INSERT INTO Customer (first_name, last_name, email, password, phone_number, address)
VALUES
('John', 'Doe', 'johndoe@example.com', 'securepass123', '1234567890', '123 Main St'),
('Jane', 'Smith', 'janesmith@example.com', 'securepass456', '0987654321', '456 Elm St');



SELECT
    Customer.first_name,
    Customer.last_name,
    COUNT(Reservations.reservation_id) AS total_reservations
FROM
    Customer
INNER JOIN
    Reservations ON Customer.user_id = Reservations.user_id
GROUP BY
    Customer.user_id, Customer.first_name, Customer.last_name;



SELECT
    Vehicles.model,
    SUM(Reservations.total_price) AS total_revenue
FROM
    Vehicles
INNER JOIN
    Reservations ON Vehicles.vehicle_id = Reservations.vehicle_id
GROUP BY
    Vehicles.model;
```

|  | model | total_revenue |
|---|---|---|
| 1 | Toyota Camry | 675.00 |
| 2 | Ford Explorer | 2100.00 |

Ln 36, Col 1 (214 selected)   Spaces: 4   UTF-8   CRLF   2 rows   MySQL   00:00:01   localhost : carrentalservice

# Conclusion

In today's competitive rental car market, a well-designed database is crucial for a company's success and growth. By implementing a structured database system, the business can efficiently manage vital operations like reservations, vehicle inventory, and customer relationships. The database design addresses key objectives, such as offering a diverse fleet, competitive pricing, exceptional customer service, and leveraging data-driven insights to make informed decisions.

The relational database model streamlines processes through centralized data management, real-time information access, and personalized customer experiences. Additionally, it enables scalability, allowing the company to adapt seamlessly to increasing demand and new business opportunities. Implementing robust security protocols safeguards customer information, fostering trust and reliability in the service.

Overall, a thoughtfully designed database system is the backbone of a thriving automobile rental service, enabling efficient operations, data-driven decision-making, and a customer-centric approach.

# Appendix A – Data Dictionary

| Table Name | Field Name | Description | Constraints |
|---|---|---|---|
| **Customer** | user_id | Unique identifier for the customer | PRIMARY KEY, AUTO_INCREMENT |
| | first_name | First name of the customer | NOT NULL |
| | last_name | Last name of the customer | NOT NULL |
| | email | Email address of the customer | UNIQUE, NOT NULL |
| | password | Encrypted password | NOT NULL |
| | phone_number | Contact number of the customer | Optional |
| | address | Residential address of the customer | Optional |
| | created_at | Timestamp of customer creation | DEFAULT CURRENT_TIMESTAMP |
| | is_active | Active status of the customer | DEFAULT TRUE |
| **Vehicles** | vehicle_id | Unique identifier for each vehicle | PRIMARY KEY, AUTO_INCREMENT |
| | model | Vehicle model | NOT NULL |
| | year | Manufacture year of the vehicle | NOT NULL |
| | vehicle_type | Type of vehicle (SUV, Sedan, etc.) | NOT NULL |
| | daily_rate | Rental rate per day | NOT NULL |
| | availability_status | Availability status (available, rented, etc.) | NOT NULL |
| | branch_id | Foreign key referencing Branches table | FOREIGN KEY REFERENCES Branches(branch_id) |
| | insurance_id | Foreign key referencing VehicleInsurance table | FOREIGN KEY REFERENCES VehicleInsurance(insurance_id) |
| **Reservations** | reservation_id | Unique identifier for each reservation | PRIMARY KEY, AUTO_INCREMENT |
| | user_id | Foreign key referencing Customer table | FOREIGN KEY REFERENCES Customer(user_id) |
| | vehicle_id | Foreign key referencing Vehicles table | FOREIGN KEY REFERENCES Vehicles(vehicle_id) |
| | reservation_date | Date the reservation was made | NOT NULL |
| | start_date | Start date of the reservation | NOT NULL |
| | end_date | End date of the reservation | NOT NULL |
| | total_price | Total cost of the reservation | NOT NULL |

| | | | |
|---|---|---|---|
| | status | Status of the reservation (confirmed, canceled, etc.) | NOT NULL |
| **Branches** | branch_id | Unique identifier for each branch | PRIMARY KEY, AUTO_INCREMENT |
| | branch_name | Name of the branch | NOT NULL |
| | address | Address of the branch | NOT NULL |
| | city | City where the branch is located | NOT NULL |
| | state | State where the branch is located | NOT NULL |
| | zip_code | Postal code of the branch | NOT NULL |
| | phone_number | Contact number of the branch | Optional |
| | email | Email address of the branch | Optional |
| **VehicleReviews** | review_id | Unique identifier for each review | PRIMARY KEY, AUTO_INCREMENT |
| | user_id | Foreign key referencing Customer table | FOREIGN KEY REFERENCES Customer(user_id) |
| | vehicle_id | Foreign key referencing Vehicles table | FOREIGN KEY REFERENCES Vehicles(vehicle_id) |
| | review_rating | Rating given to the vehicle (1-5) | CHECK (review_rating BETWEEN 1 AND 5) |
| | review_text | Text review by the customer | Optional |
| | review_date | Date and time of the review | DEFAULT CURRENT_TIMESTAMP |
| **AccidentReports** | report_id | Unique identifier for each accident report | PRIMARY KEY, AUTO_INCREMENT |
| | rental_id | Foreign key referencing Reservations table | FOREIGN KEY REFERENCES Reservations(rental_id) |
| | vehicle_id | Foreign key referencing Vehicles table | FOREIGN KEY REFERENCES Vehicles(vehicle_id) |
| | insurance_id | Foreign key referencing VehicleInsurance table | FOREIGN KEY REFERENCES VehicleInsurance(insurance_id) |
| | accident_date | Date of the accident | NOT NULL |
| | description | Detailed description of the accident | Optional |
| | damage_cost | Estimated cost of damage | Optional |
| | is_resolved | Resolution status of the report | DEFAULT FALSE |
| **Employees** | employee_id | Unique identifier for each employee | PRIMARY KEY, AUTO_INCREMENT |
| | first_name | First name of the employee | NOT NULL |
| | last_name | Last name of the | NOT NULL |

| | | | |
|---|---|---|---|
| | | employee | |
| | role | Role or job position of the employee | NOT NULL |
| | branch_id | Foreign key referencing Branches table | FOREIGN KEY REFERENCES Branches(branch_id) |
| | hire_date | Date the employee was hired | NOT NULL |
| | email | Email address of the employee | Optional |
| | phone_number | Contact number of the employee | Optional |

# Appendix B – SQL Queries

1. Vehicles

```
CREATE TABLE Vehicles (
   vehicle_id INT PRIMARY KEY AUTO_INCREMENT,
   model VARCHAR(50),
   year INT,
   vehicle_type VARCHAR(50),  -- (e.g., 'SUV', 'Sedan', 'Truck')
   daily_rate DECIMAL(10, 2),
   availability_status VARCHAR(20), -- (e.g., 'available', 'rented', 'maintenance')
   branch_id INT,
   FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
FOREIGN KEY (insurance_id) REFERENCES VehicleInsurance(insurance_id)
);

INSERT INTO Vehicles (model, year, vehicle_type, daily_rate, availability_status, branch_id, insurance_id) VALUES
('Toyota Camry', 2020, 'Sedan', 45.99, 'available', 1, 101),
('Ford Explorer', 2019, 'SUV', 65.50, 'rented', 2, 102),
 ('Chevrolet Silverado', 2021, 'Truck', 80.00, 'available', 1, 103),
 ('Honda Civic', 2018, 'Sedan', 40.00, 'maintenance', 3, 104),
('Tesla Model 3', 2022, 'Sedan', 120.00, 'available', 2, 105);
```

2. Reservations

```
CREATE TABLE Reservations (
   reservation_id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT,
   vehicle_id INT,
  reservation_date DATE,
  start_date DATE,
  end_date DATE,
  total_price INT,
  status VARCHAR(20), -- (e.g., 'confirmed', 'cancelled', 'completed')
  FOREIGN KEY (user_id) REFERENCES Users(user_id),
  FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id)
);

INSERT INTO Reservations (user_id, vehicle_id, reservation_date, start_date, end_date, total_price, status)
VALUES
(1, 1, '2024-09-15', '2024-09-20', '2024-09-25', 229.95, 'confirmed'),
(2, 2, '2024-09-18', '2024-09-21', '2024-09-23', 131.00, 'completed'),
(3, 3, '2024-09-20', '2024-09-22', '2024-09-24', 160.00, 'cancelled'),
(4, 4, '2024-09-25', '2024-09-28', '2024-10-01', 240.00, 'confirmed'),
(5, 5, '2024-09-30', '2024-10-02', '2024-10-05', 360.00, 'confirmed');
```

3. Branch

```
CREATE TABLE Branches (
  branch_id INT PRIMARY KEY AUTO_INCREMENT,
  branch_name VARCHAR(100),
  address TEXT,
  city VARCHAR(50),
  state VARCHAR(50),
  zip_code VARCHAR(10),
  phone_number VARCHAR(20),
  email VARCHAR(100)
);
```

```
INSERT INTO Branches (branch_name, address, city, state, zip_code, phone_number, email)
VALUES
('Downtown Branch', '123 Main St', 'Springfield', 'IL', '62701', '217-555-1234', 'downtown@carrental.com'),
('Airport Branch', '456 Airport Rd', 'Springfield', 'IL', '62707', '217-555-5678', 'airport@carrental.com'),
('Eastside Branch', '789 East Blvd', 'Shelbyville', 'IL', '62565', '217-555-9876', 'eastside@carrental.com'),
('Westside Branch', '321 West Dr', 'Ogdenville', 'IL', '62298', '618-555-4321', 'westside@carrental.com'),
('Capitol Branch', '654 Capitol Ave', 'Capital City', 'IL', '62702', '217-555-8765', 'capitol@carrental.com');
```

4. VehicleReviews

```
CREATE TABLE VehicleReviews (
   review_id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT,
   vehicle_id INT,
   review_rating INT CHECK (review_rating BETWEEN 1 AND 5),
   review_text TEXT,
   review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id)
);

INSERT INTO VehicleReviews (user_id, vehicle_id, review_rating, review_text)
VALUES
(1, 1, 5, 'Excellent car! Smooth ride and very comfortable. Highly recommend!'),
(2, 2, 4, 'Great SUV for family trips. Good space but a bit thirsty on gas.'),
(3, 3, 3, 'The truck was okay. Performance was decent, but it had some scratches.'),
(4, 4, 5, 'Absolutely loved this car! Perfect for my weekend getaway.'),
(5, 5, 4, 'The Tesla was amazing! Quick acceleration and great technology. A bit pricey though.');
```

6. Accident reports

```
CREATE TABLE AccidentReports (
  report_id INT PRIMARY KEY AUTO_INCREMENT,
  rental_id INT,
  vehicle_id INT,
  insurance_id INT,
  accident_date DATE,
  description TEXT,
  damage_cost DECIMAL(10, 2),
  is_resolved BOOLEAN DEFAULT 0,
  FOREIGN KEY (rental_id) REFERENCES Rentals(rental_id),
  FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id)
  FOREIGN KEY (insurance_id) REFERENCES VehicleInsurance(vehicle_id)
);
INSERT INTO AccidentReports (rental_id, vehicle_id, insurance_id, accident_date, description, damage_cost, is_resolved)
VALUES
(1, 1, 101, '2024-09-21', 'Minor fender bender, no injuries reported.', 250.00, 0),
(2, 2, 102, '2024-09-22', 'Side collision at an intersection.', 1500.50, 0),
(3, 3, 103, '2024-09-23', 'Rear-end collision, extensive damage.', 3500.75, 1),
(4, 4, 104, '2024-09-24', 'Accident due to slippery road conditions.', 800.00, 0),
(5, 5, 105, '2024-09-25', 'Vehicle rolled over, severe damage.', 5000.00, 1);
```

## 7. EMPLOYEE TABLE

Contains employee details for managing client service and utilizing technology-driven insights.

```
CREATE TABLE Employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  role VARCHAR(50), -- (e.g., 'customer service', 'manager', 'mechanic')
  branch_id INT,
  hire_date DATE,
  email VARCHAR(100),
  phone_number VARCHAR(20),
  FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);


INSERT INTO Employees (first_name, last_name, role, branch_id, hire_date, email, phone_number)
VALUES
('Alice', 'Johnson', 'manager', 1, '2023-01-15', 'alice.johnson@carrental.com', '217-555-0011'),
('Bob', 'Smith', 'customer service', 2, '2023-03-10', 'bob.smith@carrental.com', '217-555-0012'),
('Charlie', 'Brown', 'mechanic', 1, '2023-05-20', 'charlie.brown@carrental.com', '217-555-0013'),
('David', 'Williams', 'customer service', 3, '2023-07-05', 'david.williams@carrental.com', '217-555-0014');
```