**Project: Car Rental Service**

I interviewed a staff member at a local car rental business to better understand the day to day operations and what information is essential to track in a car rental system. Based on the interview, I identified the following key data elements the system needs to capture:

**Customer Details:**Full Name  
Contact Number  
Email Address  
Driver’s License Number  
Address

**Car Information:**Make and Model  
Year  
Plate Number  
Daily Rental Rate  
Vehicle Status (Available, Rented, Maintenance)

**Branch Locations: (if multiple locations exist)**Location Number  
Location Address

**Rental Transactions:**Rental Start and End Dates  
Car linked to the rental  
Employee who handled the transaction  
Total Cost of Rental  
Payment Status (Paid, Pending, Unpaid)

**Payments:**Date of Payment  
Amount Paid  
Payment Method (Credit Card, Cash, etc.)

**Employee Details:**Name  
Role (Manager, Front Desk, etc.)  
Contact Information  
Assigned Branch

**Branch Information:**Branch Name  
Phone Number  
Address

🔹 Documentation of Process:

To better understand how to structure the car rental system, I conducted an informal interview with someone who works at a small independent car rental business. I asked them to walk me through the process of renting a car, step-by-step, and to explain what information they rely on in their system to serve customers and manage operations.

The employee described the typical workflow: a customer walks in or calls to check availability, their information is collected or retrieved from the system, they choose a car, a rental contract is created with pickup and return dates, payment is recorded, and the car is marked as rented. When the car is returned, the system updates the status and calculates the total cost based on the daily rate and number of days used.

What stood out was how important it was to have clean and connected records. The employee mentioned how frustrating it can be when information is scattered, especially when dealing with multiple cars, employees, or late returns. They also noted that while their business only has one location, it’s helpful to be able to assign employees and cars to a branch in case they expand in the future.

Based on this conversation, I determined that my system needed to include core entities such as Customer, Car, Rental, Payment, Employee, and Branch. The interview helped me make realistic choices about what to include and focus on simplicity while allowing room for scalability.

**Challenges**  
Throughout the development of my car rental database project, I encountered several challenges that helped me understand how technical and design decisions affect the structure and usability of a database. While building the system in Microsoft Access and preparing to export it to MySQL, I faced obstacles related to relationship mapping, data consistency, SQL syntax conversion, and maintaining simplicity while preparing for future scalability.

One of the first issues I encountered was setting up proper foreign key relationships. Microsoft Access requires that primary keys exist and be set correctly before foreign keys can be created. Early on, I forgot to make some of my ID fields “Not Null” or set them as primary keys, which caused Access to reject relationship connections. I had to go back into each table’s design view and double-check that my key fields were properly configured, which slowed down my initial progress but helped me learn the importance of enforcing integrity early in the process.

Another major challenge was ensuring sample data consistency. When entering test records into the tables, I ran into errors because some foreign key values didn’t exist in the related tables. For example, a rental couldn’t reference a car that hadn’t been entered yet, or a payment couldn’t link to a rental that didn’t exist. I had to carefully plan the order in which I entered records — starting with customers, cars, employees, and branches, then moving on to rentals and finally payments. This made me realize how critical it is to understand relational dependencies before inserting data.

I also had difficulty translating my Access database into MySQL SQL scripts. Access and MySQL handle constraints and data types differently. For instance, Access allows certain defaults without declaring them explicitly, while MySQL is stricter about NOT NULL and requires that every primary and foreign key is clearly defined with matching data types. I had to rewrite many parts of my script to fix issues like missing key constraints or incompatible data types, especially when moving from Access' AutoNumber to MySQL's AUTO\_INCREMENT.

Another challenge was figuring out how much complexity to include in the system. While the project focuses on a small, single-branch business, I chose to include a Branch table so the system could scale in the future. This added extra relationships, such as connecting employees and cars to branches. I had to think carefully about how to structure those links without overwhelming the design or introducing redundancy. Keeping the system simple, but still flexible for future needs, was a balancing act.

Finally, designing the Chen-style ER diagram by hand was more time-consuming than I expected. It required translating my digital relationships into Chen notation using diamonds for relationships and ensuring I labeled everything clearly with appropriate cardinality. Although it took extra effort, this step helped me visually confirm that my data model matched my understanding of the system’s real-world flow.

Overall, these challenges made the project more meaningful and gave me a better understanding of how to build functional, scalable, and logically sound databases. I feel more confident now in using both Access and MySQL to design relational systems and troubleshoot real-world data design problems.