Final project data base

Idea : The project idea is to create a system to manage a car rental business. This involves keeping track of customers, available cars, rental agreements, and the locations (branches) where cars are available. The system would likely allow customers to browse and book cars, record rental details, and manage the return process. It aims to streamline the rental operations for the business and provide a better experience for customers.

ERD: Entities: These are the main objects or concepts about which you want to store information. In this diagram, the entities are represented by rectangles:

\* Customer: Represents individuals who rent cars. Attributes include ID (likely the primary key), Address, Name, and Phone.

\* Car: Represents the vehicles available for rent. Attributes include ID (likely the primary key), Plate Number, Year, Model, and Status.

\* Employee: Represents the staff members. Attributes include ID (likely the primary key), Name, and Position.

\* Branch: Represents the physical locations of the rental service. Attributes include ID (likely the primary key), Location, and Name.

\* Rental: Represents the act of a customer renting a car. It acts as a relationship with its own attributes (like Date).

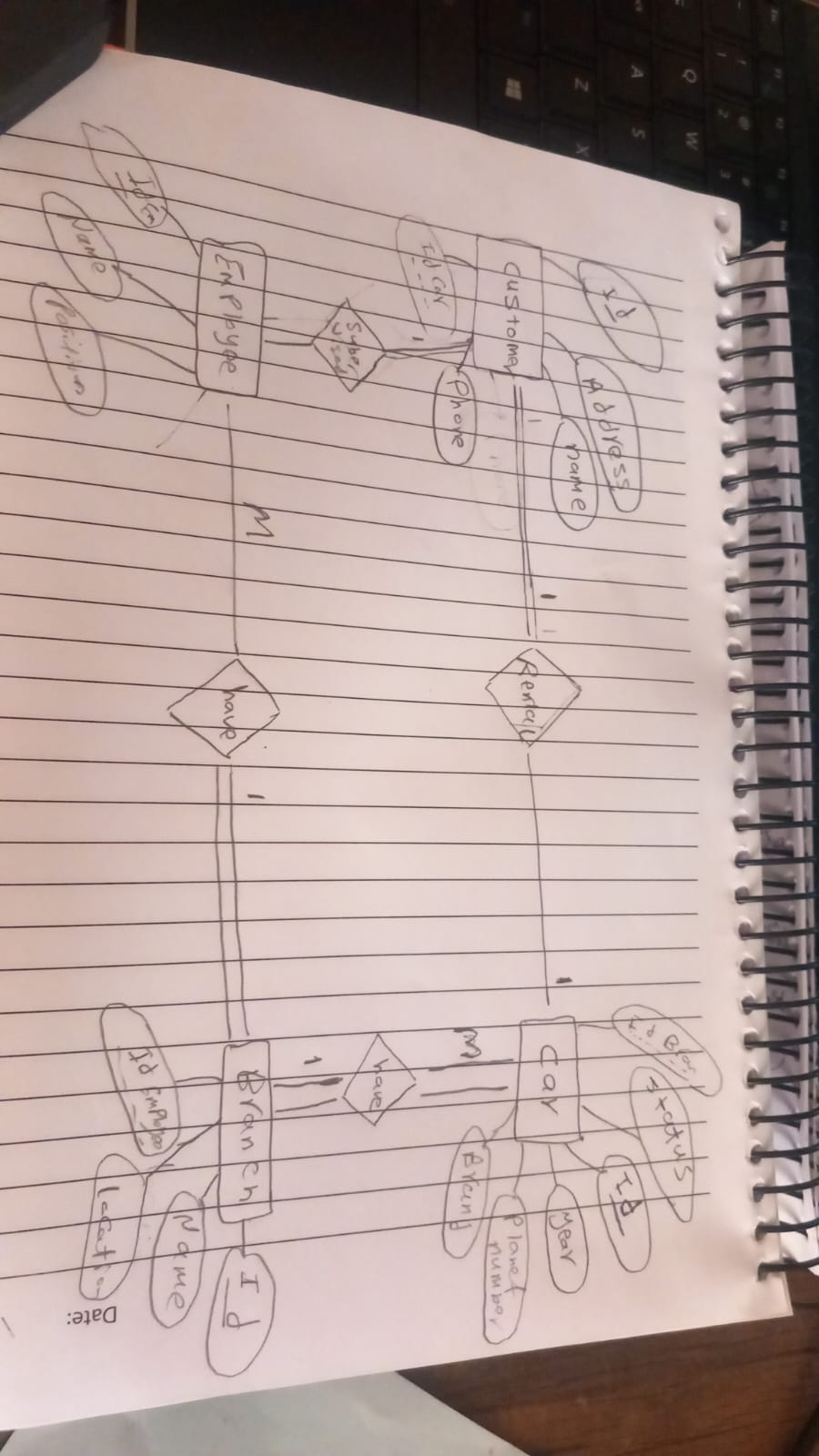
Relationships: These show how the entities are connected to each other. Relationships are represented by diamonds:

\* Rents: Connects the "Customer" entity to the "Car" entity through the "Rental" entity. The "1:M" notation indicates that one customer can rent many cars (over time, through different rentals), and one car can be rented by many customers (over time, through different rentals). The "Rental" entity itself holds information specific to each rental transaction (like the date).

\* Works at: Connects the "Employee" entity to the "Branch" entity. The "M:1" notation indicates that many employees can work at one branch, but each employee works at only one branch (in this simplified model).

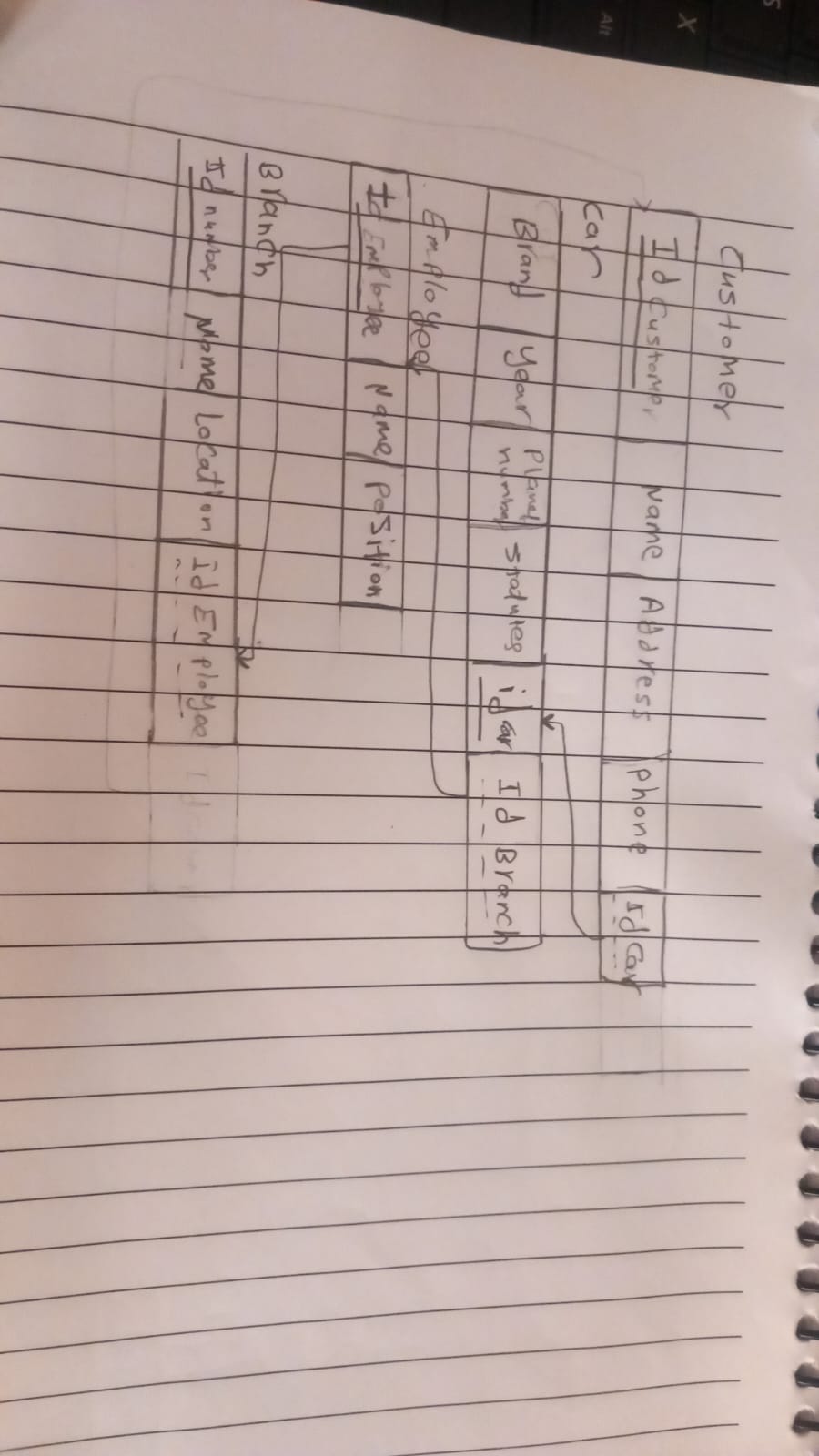
\* Has: Connects the "Branch" entity to the "Car" entity. The "1:M" notation indicates that one branch can have many cars, but each car belongs to only one branch (in this simplified model).

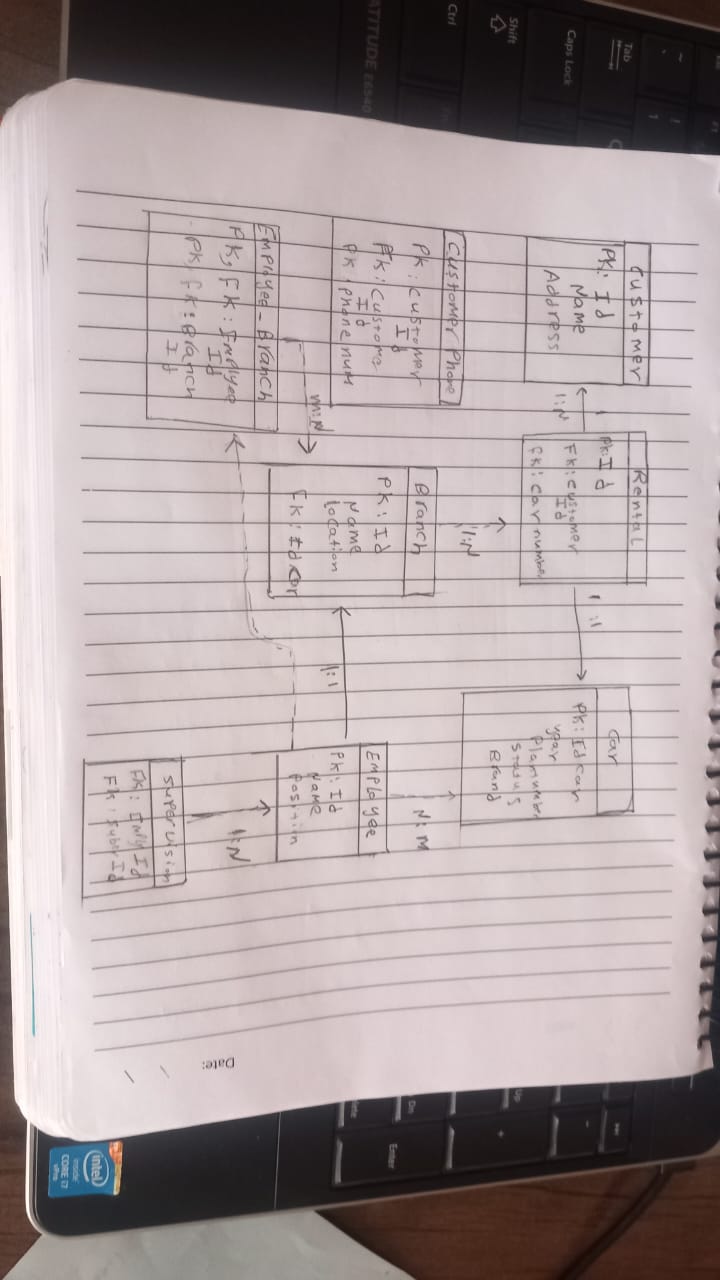
In summary: This ER diagram models a car rental system. Customers rent cars. Employees work at branches. Branches have cars. The "Rental" entity captures the details of each rental transaction, linking customers and cars.



shows a basic database design for a car rental system. It outlines different tables (like Customer, Car, Employee, Branch, and Car-Branch) and the information (columns) each table will hold. It also shows how these tables are related to each other using arrows, indicating connections between different pieces of information (like a customer renting a car or an employee working at a branch).

Essentially, it's a plan for organizing information about customers, cars, employees, and branches so that the rental process can be managed efficiently





the database design appears to adhere to the principles of the Third Normal Form (3NF).

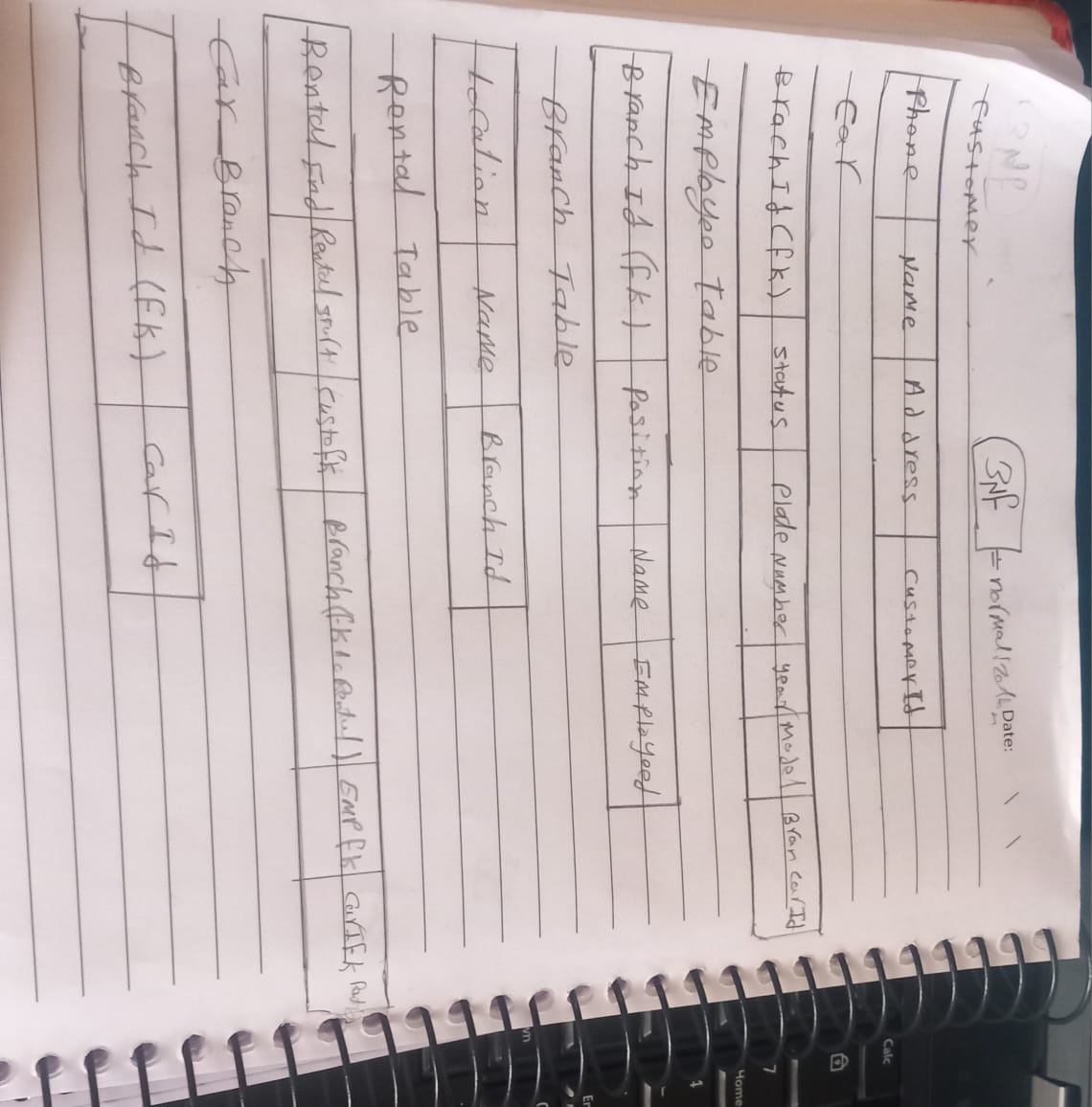
In simple terms, 3NF aims to organize data efficiently to reduce redundancy and improve data management. It achieves this by ensuring that:

\* Each table holds information about a single subject or entity. For example, the Customer table is solely about customer details, and the Car table is only about car information.

\* All information in a table directly relates to the primary identifier (primary key) of that table. For instance, the customer's name, address, and phone number directly describe the customer identified by the CustomerID.

\* There are no hidden dependencies where one piece of non-key information depends on another piece of non-key information. For example, the employee's position depends on the employee, not on the branch they work at. Branch-specific information (like location) is kept in the Branch Table.

By following 3NF, this car rental database design avoids unnecessary repetition of data, making it easier to update information without causing inconsistencies across the database

QUERS RESULT

