Relational Database Design Final Project

My Client’s background and my questions to the client:
Imagine a car repair shop “BestCar” asked me to build a database for them. In order to get information to build the database, I want ask the client two categories of questions:

- Interests & Entities: The interest of the business? What information (attributes) to record? What might be the possible entities of business? etc.
- The relationship of the entities: cardinalities (one to many, many to many, or one to one), especially, participation (mandatory or optional).

As per my client, they recently hired a data analyst who discovered some *data redundancy and inconsistency issues* within the client’s current database. The analyst suggested organizing the data in a manner that facilitates efficient updates and retrieval, while also minimizing redundancy.

Create Entity Relationship Model

Assume after several rounds of discussion, I confirmed that the client approximately want to the following entities, the attributes, and relationships among entities:

Entities, attributes, and identifiers
Here is what I summarized about the entities. (* indicate the primary key):

- Customers: *CustomerID, FirstName, LastName, Email, Address, Phone, PaymentInfo
- Cars: *CustomerID, *Car#, CarName, CarBrand, CarModel, CarColor, CarYear, CustomerName, CustomerPhone,
- Staffs: *EmployeeID, SSN, FirstName, LastName, DOB, Address, Email, Phone
- Visits: *VisitID, Date, Time, CustomerID, Car#, ServiceID, ServiceName, ServicePrice, ServiceDescription, Staff, Bill

Relationships among entities

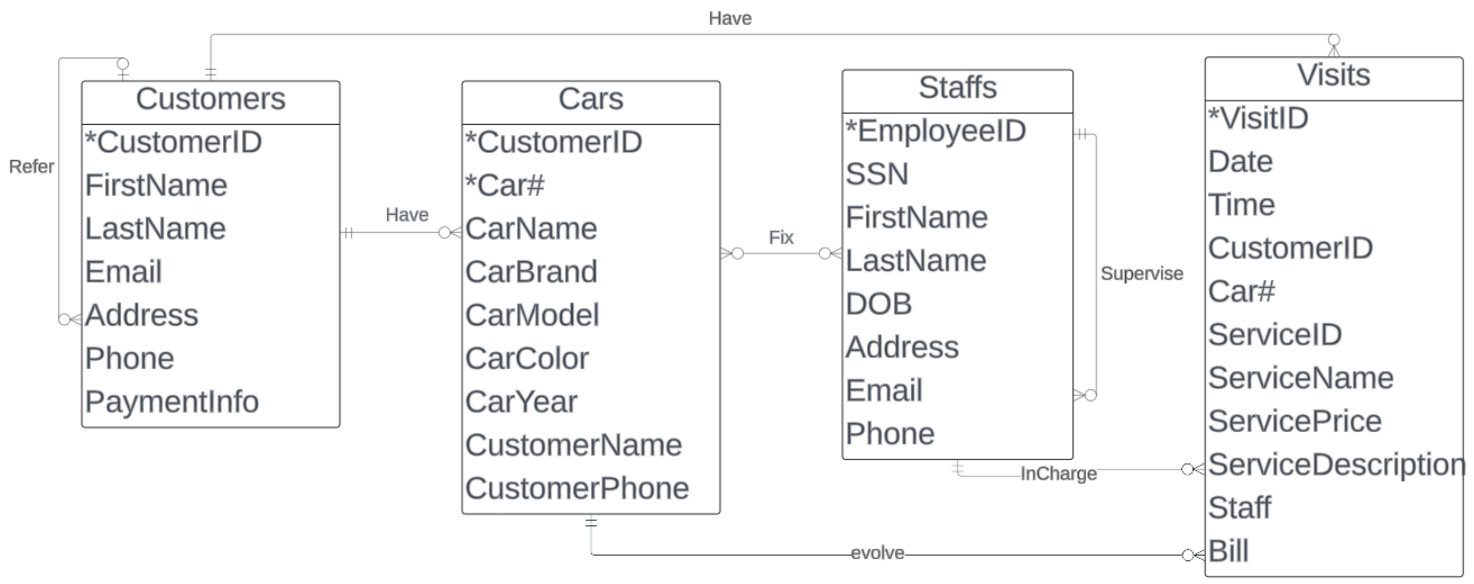
Also, assume after I confirmed with client, the relationships are follows:

- A customer may have one or more cars; A car must belong to one and only one customer.
- A customer may have one or more visits; A visit must be done by one and only one customer.
- A car may have one or more visits; A visit must involve one and only one car.
- A car may be fixed with one or more staff. A staff may fix one or more cars;.

- A staff must be in charge of one or more visits; A visit must be charged by one and only one staff.
- A staff must have one and only one supervisor; A staff may supervise one or more staff.
- A customer may be referred by only one customer; A customer may refer to one or more customers.

Create the Entity Relationship Diagram (ERD)

Having the relationship, we can create the entity relationship diagram.



Convert ERD to a Relational Model

Here is the relational model I converted from the ERD:

- Customers (<u>CustomerID</u>, FirstName, LastName, Email, Address, Phone, PaymentInfo, ReferredByCustomerID(fk)).
- Cars (<u>CustomerID(fk)</u>, <u>Car#</u>, CarName, CarBrand, CarColor, CarYear, CustomerName, CustomerPhone).
- Staff (<u>EmployeeID</u>, SSN, FirstName, LastName, DOB, Address, Email, Phone, SupervisorID(fk)).
- Visit (<u>VisitID</u>, Date, Time, CustomerID(fk), Car#(fk), ServiceID, ServiceName, ServicePrice, ServiceDescription, EmployeeID(fk), Bill).
- Cars_Staff(<u>CustomerID(fk)</u>, <u>Car#(fk)</u>,<u>EmployeeID(fk)</u>)

Normalize the Relational Model to 3NF

Functional Dependencies:

Before normalization, we should ask the client about the Functional Dependencies of the relations. Assume here is what we got from the client:

- Customers (CustomerID, FirstName, LastName, Email, Address, Phone, PaymentInfo, ReferredByCustomerID(fk)).
    - FD1: CustomerID → FirstName, LastName, Email, Address, Phone, PaymentInfo, ReferredByCustomerID
- Cars (CustomerID(fk), Car#, CarName, CarBrand, CarColor, CarYear, CustomerName, CustomerPhone).
    - FD1: CustomerID, Car# → CarName, CarBrand, CarColor, CarYear, CustomerName, CustomerPhone
    - FD2: CustomerID → CustomerName, CustomerPhone
- Staff (EmployeeID, SSN, FirstName, LastName, DOB, Address, Email, Phone, SupervisorID(fk)).
    - FD1: EmployeeID → SSN, FirstName, LastName, DOB, Address, Email, Phone, SupervisorID
- Visit (VisitID, Date, Time, CustomerID(fk), Car#(fk), ServiceID, ServiceName, ServicePrice, ServiceDescription, EmployeeID(fk), Bill).
    - FD1: VisitID → Date, Time, CustomerID, Car#, ServiceID, ServiceName, ServicePrice, ServiceDescription, EmployeeID, Bill
    - FD2: ServiceID → ServiceName, ServicePrice, ServiceDescription.
- Cars_Staff(CustomerID(fk), Car#(fk), EmployeeID(fk))
    - There is no non-primary-key attribute.

Now we can *normalize this Relational Model to 3NF*:

Here is the normalization process:

- Customers, Staff, and Cars_Staff relations are in 3NF, because they are in 1NF; they have no partial functional dependencies so they are in 2NF; and they have no transitive functional dependencies so they are in 3NF.
- Cars relation is in 1NF. However, it is not in 2NF because FD2: CustomerID → CustomerName, CustomerPhone. CustomerID as part of the primary key, determines non-primary-key attributes. This leads to a partial functional dependency. We need to normalize Cars to 2NF:
    - Create a new relation to put CustomerID, CustomerName, CustomerPhone. Since Customer relation has these attributes, we can simply remove them from Cars, and keep CustomerID as a foreign key. Now, Cars relation is:
    - Cars (CustomerID(fk), Car#, CarName, CarBrand, CarColor, CarYear).

        - FD1: CustomerID, Car# → CarName, CarBrand, CarColor, CarYear

- Cars relation now is in 2NF. It's also in 3NF because it is in 1NF and 2NF, and there are no transitive functional dependencies.
- Visit relation is in 1NF, and 2NF. However, it is not in 3NF because of FD2: ServiceID → ServiceName, ServicePrice, ServiceDescription. VisitID → ServiceID, and ServiceID → ServiceName, ServicePrice, ServiceDescription is a transitive functional dependency. We need to normalize Visit to 3NF:
    - Create a new relation to put ServiceID, ServiceName, ServicePrice, ServiceDescription and modify visit. Now, Service and Visit relations are:
    - Service (ServiceID, ServiceName, ServicePrice, ServiceDescription)
        - FD1: ServiceID → ServiceName, ServicePrice, ServiceDescription
    - Visit (VisitID, Date, Time, CustomerID(fk), Car#(fk), ServiceID(fk), EmployeeID(fk), Bill).
        - FD1: VisitID → Date, Time, CustomerID, Car, ServiceID, EmployeeID, Bill

Finalize the relational model in 3NF for further implementation

- Customers (CustomerID, FirstName, LastName, Email, Address, Phone, PaymentInfo, ReferredByCustomerID(fk)).
    - FD1: CustomerID → FirstName, LastName, Email, Address, Phone, PaymentInfo, ReferredByCustomerID
- Cars (CustomerID(fk), Car#, CarName, CarBrand, CarColor, CarYear).
    - FD1: CustomerID, Car# → CarName, CarBrand, CarColor, CarYear
- Staff (EmployeeID, SSN, FirstName, LastName, DOB, Address, Email, Phone, SupervisorID(fk)).
    - FD1: EmployeeID → SSN, FirstName, LastName, DOB, Address, Email, Phone, SupervisorID
- Service (ServiceID, ServiceName, ServicePrice, ServiceDescription)
    - FD1: ServiceID → ServiceName, ServicePrice, ServiceDescription
- Visit (VisitID, Date, Time, CustomerID(fk), Car#(fk), ServiceID(fk), EmployeeID(fk), Bill).
    - FD1: VisitID → Date, Time, CustomerID, Car, ServiceID, EmployeeID, Bill
- Cars_Staff(CustomerID(fk), Car#(fk), EmployeeID(fk))
    - There is no non-primary-key attribute