Question 01:

Consider a scenario where you're tasked with designing an ERD for a bus ticketing system. This system facilitates the sale of tickets for bus transportation services.

Entities:

- **Customer:** Represents individuals who purchase tickets. Attributes may include CustomerID, Name, Email, Phone, etc.
- **Ticket:** Represents the ticket purchased by a customer for a bus journey. Attributes may include TicketID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, etc.
- **Bus Route:** Represents the specific route for bus journeys. Attributes may include RouteID, DepartureLocation, ArrivalLocation, Distance, etc.
- **Bus:** Represents the individual buses available for booking. Attributes may include BusID, BusNumber, Model, Capacity, etc.
- **Payment:** Represents the payment details for each ticket transaction. Attributes may include PaymentID, Amount, PaymentMethod, etc.

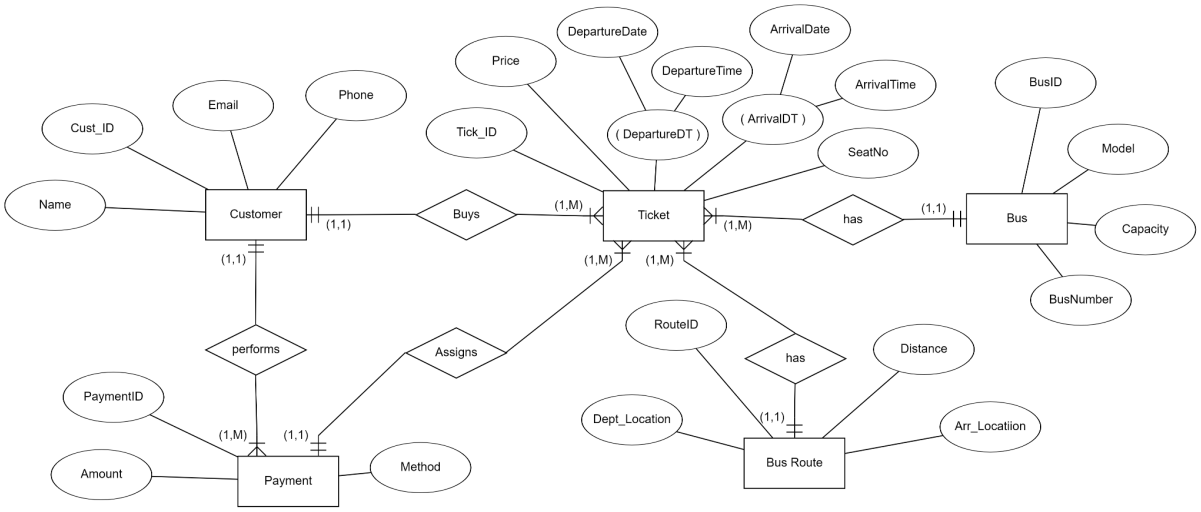
Relationships:

- **Customer - Ticket (1:M):** A customer can purchase multiple tickets, but each ticket is associated with only one customer.
- **Ticket - Bus Route (M:1):** Each ticket corresponds to a specific bus route, but a bus route can have multiple tickets associated with it.
- **Ticket - Bus (M:1):** Each ticket is for a specific bus, but each bus can have multiple tickets associated with it.
- **Customer - Payment (1:M):** A customer can make multiple payments, but each payment is associated with only one customer.
- **Payment - Ticket (1:1 or 1:M):** Each payment is associated with one or more tickets, depending on whether a customer purchases multiple tickets in a single transaction or not. I'm going to assume it is **(1:M)**.

Functional Dependencies:

- CustomerID → Name, Email, Phone
- Email → CustomerID
- Phone → CustomerID
- TicketID → Price, DepartureDateTime, ArrivalDateTime, SeatNumber, CustomerID, RouteID, BusID
- RouteID → DepartureLocation, ArrivalLocation, Distance
- BusID → BusNumber, Model, Capacity
- PaymentID → Amount, PaymentMethod, CustomerID, TicketID

The ER Diagram for this scenario is given as under:



Now for normalization of the FDs, we first find all Candidate keys. First we separate RHS into multiple parts and then find the closure for all Combinations of keys. We get:

- **[DepartureLocation, ArrivalLocation, Distance, BusNumber, Model, Capacity, PaymentID].**

Now the current form is in 1NF. To convert to 2NF, find the minimal cover of the FDs, which includes the FDs:

- CustomerID → Name
- CustomerID → Email
- CustomerID → Phone
- Email → CustomerID
- Phone → CustomerID
- TicketID → Price
- TicketID → DepartureDateTime
- TicketID → ArrivalDateTime
- TicketID → SeatNumber
- TicketID → CustomerID
- TicketID → RouteID
- TicketID → BusID
- PaymentID → Amount
- PaymentID → PaymentMethod
- PaymentID → TicketID

The table is not in 2NF. The FD [PaymentID → Amount] is a partial dependency (i.e., LHS is a proper subset of some CK), the table is split. Thus we get the Functional Dependencies:

- CustomerID → Phone.
- Email → Phone.
- Phone → Email, CustomerID, Name.
- PaymentID → TicketID, Amount, PaymentMethod.

- TicketID → CustomerID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, RouteID, BusID.
- All the other attributes can go to different tables as there are no dependencies between them. For 3NF, we remove the transitive dependencies. So, we get tables:

Relation 01:

- CustomerID → Email
- Email → Phone
- Phone → CustomerID, Name

Relation 02:

- Email → CustomerID
- CustomerID → Email

Relation 03:

- Phone → CustomerID
- CustomerID → Phone

Relation 03:

- TicketID → CustomerID, BusID, RouteID, SeatNumber, ArrivalDateTime, DepartureDateTime, Price

Relation 04:

- PaymentID → TicketID, PaymentMethod, Amount

For conversion into BCNF, find merged minimal cover of FDs, which contains:

- CustomerID → Name, Email, Phone.
- Email → CustomerID.
- Phone → CustomerID.
- TicketID → Price, DepartureDateTime, ArrivalDateTime, SeatNumber, CustomerID, RouteID, BusID.
- PaymentID → Amount, PaymentMethod, TicketID.

The FD [CustomerID → Name, Email, Phone] violates BCNF as the LHS is not superkey. Table is split into the two below:

- (CustomerID, Name, Email, Phone)
- (CustomerID, TicketID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, RouteID, BusID, DepartureLocation, ArrivalLocation, Distance, BusNumber, Model, Capacity, PaymentID, Amount, PaymentMethod)

The FD [PaymentID → TicketID, Amount, PaymentMethod] violates BCNF as the LHS is not superkey. Table is split into the two below:

- (PaymentID, TicketID, Amount, PaymentMethod, CustomerID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, RouteID, BusID).
- (DepartureLocation, ArrivalLocation, Distance, BusNumber, Model, Capacity, PaymentID).

The FD [TicketID → CustomerID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, RouteID, BusID] violates BCNF as the LHS is not superkey. Table is split into the two below:

- (TicketID, CustomerID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, RouteID, BusID)
- (PaymentID, TicketID, Amount, PaymentMethod)

So, for BCNF we get the final tables:

Relation 01:

- CustomerID → Phone.
- Email → Phone.
- Phone → Email, CustomerID, Name.

The attributes DepartureLocation. ArrivalLocation, Distance, BusNumber, Model, Capacity, PaymentID can be put in separate tables.

Relation 02:

- TicketID → CustomerID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, RouteID, BusID.

Relation 03:

- PaymentID → TicketID, Amount, PaymentMethod.

This represents the final BCNF form of the FDs.