A) For the food delivery app ER Diagram in the first assignment, I believe we can make the following normalised tables:

User ID	Name	Contact info	Email	Password

User ID	Address ID

Address ID	Pin code	apartment

Pin code	State	city

Driver ID	Name	phone	email id	Live location?

Restaurant ID	Name	Cuisine

Restaurant ID	Locations/outlets	Timings	Discount coupons

Item ID	Item Name	Description	Category	Price

Restaurant ID	Item ID

Rating ID	User ID	Restaurant ID	Rating

Order ID	Address ID	Driver ID	Order Total	Order Status	Description	Special instruction

Order ID	Item ID

Payment ID	Order ID	Mode of payment	Status

B)

```sql
-- Create Users table
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    ContactInfo VARCHAR(20),
    Email VARCHAR(100) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL
);

-- Create Addresses table
CREATE TABLE Addresses (
    AddressID INT PRIMARY KEY,
    PinCode VARCHAR(10) NOT NULL,
    Apartment VARCHAR(50),
    State VARCHAR(50),
    City VARCHAR(50)
);

-- Create Drivers table
CREATE TABLE Drivers (
    DriverID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Phone VARCHAR(20),
    EmailID VARCHAR(100) UNIQUE NOT NULL,
    Location VARCHAR(50)
);

-- Create Restaurants table
CREATE TABLE Restaurants (
    RestaurantID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Cuisine VARCHAR(50)
);

-- Create Locations table
CREATE TABLE Locations (
    LocationID INT PRIMARY KEY,
    RestaurantID INT,
    Outlet VARCHAR(50),
    Timings VARCHAR(50),
    DiscountCoupons VARCHAR(50),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)
);
```

```sql
-- Create Items table
CREATE TABLE Items (
    ItemID INT PRIMARY KEY,
    ItemName VARCHAR(50) NOT NULL,
    Description VARCHAR(255),
    Category VARCHAR(50),
    Price DECIMAL(10, 2)
);

-- Create RestaurantItems table
CREATE TABLE RestaurantItems (
    RestaurantID INT,
    ItemID INT,
    PRIMARY KEY (RestaurantID, ItemID),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID),
    FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
);

-- Create Ratings table
CREATE TABLE Ratings (
    RatingID INT PRIMARY KEY,
    UserID INT,
    RestaurantID INT,
    Rating INT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)
);

-- Create Orders table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    AddressID INT,
    DriverID INT,
    OrderTotal DECIMAL(10, 2),
    OrderStatus VARCHAR(50),
    Description VARCHAR(255),
    SpecialInstruction VARCHAR(255),
    FOREIGN KEY (AddressID) REFERENCES Addresses(AddressID),
    FOREIGN KEY (DriverID) REFERENCES Drivers(DriverID)
);

-- Create OrderItems table
CREATE TABLE OrderItems (
    OrderID INT,
    ItemID INT,
    PRIMARY KEY (OrderID, ItemID),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
```

```
    FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
);

-- Create Payments table
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY,
    OrderID INT,
    ModeOfPayment VARCHAR(50),
    Status VARCHAR(50),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);

-- Insert mock data into Users table
INSERT INTO Users (UserID, Name, ContactInfo, Email, Password)
VALUES
(1, 'John Doe', '1234567890', 'john@example.com', 'password123'),
(2, 'Jane Smith', '9876543210', 'jane@example.com', 'pass456');

-- Insert mock data into Addresses table
INSERT INTO Addresses (AddressID, PinCode, Apartment, State, City)
VALUES
(1, '12345', 'Apt 101', 'California', 'Los Angeles'),
(2, '54321', 'Apt 202', 'New York', 'New York');

-- Insert mock data into Drivers table
INSERT INTO Drivers (DriverID, Name, Phone, EmailID, Location)
VALUES
(1, 'Driver 1', '1112223333', 'driver1@example.com', 'Location A'),
(2, 'Driver 2', '4445556666', 'driver2@example.com', 'Location B');

-- Insert mock data into Restaurants table
INSERT INTO Restaurants (RestaurantID, Name, Cuisine)
VALUES
(1, 'Restaurant A', 'Italian'),
(2, 'Restaurant B', 'Mexican');

-- Insert mock data into Locations table
INSERT INTO Locations (LocationID, RestaurantID, Outlet, Timings,
DiscountCoupons)
VALUES
(1, 1, 'Outlet 1', '9 AM - 9 PM', 'DISC10'),
(2, 1, 'Outlet 2', '10 AM - 8 PM', 'SAVE20'),
(3, 2, 'Outlet 3', '11 AM - 10 PM', 'MEAL50');

-- Insert mock data into Items table
INSERT INTO Items (ItemID, ItemName, Description, Category, Price)
VALUES
(1, 'Pizza Margherita', 'Classic Italian pizza', 'Main Course', 12.99),
```

```
(2, 'Tacos', 'Spicy Mexican tacos', 'Appetizer', 8.99);

-- Insert mock data into RestaurantItems table
INSERT INTO RestaurantItems (RestaurantID, ItemID)
VALUES
(1, 1),
(2, 2);

-- Insert mock data into Ratings table
INSERT INTO Ratings (RatingID, UserID, RestaurantID, Rating)
VALUES
(1, 1, 1, 4),
(2, 2, 2, 5);

-- Insert mock data into Orders table
INSERT INTO Orders (OrderID, AddressID, DriverID, OrderTotal, OrderStatus,
Description, SpecialInstruction)
VALUES
(1, 1, 1, 25.99, 'Delivered', 'Order for dinner', 'No onions'),
(2, 2, 2, 18.99, 'Pending', 'Lunch order', 'Extra cheese');

-- Insert mock data into OrderItems table
INSERT INTO OrderItems (OrderID, ItemID)
VALUES
(1, 1),
(2, 2);

-- Insert mock data into Payments table
INSERT INTO Payments (PaymentID, OrderID, ModeOfPayment, Status)
VALUES
(1, 1, 'Credit Card', 'Paid'),
(2, 2, 'Cash', 'Pending');
```

I think we can create the tables and populate it with data as written and then we can split the addresses table into the required tables as mentioned above, along with address to user mapping.

C) All the tables are in the first normalized form as they are unimodular, and each cell has only one data. To move the tables to 2NF, it is essential to ensure the absence of partial dependency of data. Also, User ID and Address ID are not in the same table, as these two would be key values, and then Name, Pincode, etc. will depend on only one of the two keys. Such a partial dependency can cause anomalies.

- User Table: User ID serves as the primary key in the User table, with Address ID linked to it. This arrangement avoids partial dependency, ensuring that each attribute in the User table is fully dependent on the primary key.
- Address Table: Address ID, pincode, state, and city are stored in separate tables since state and city are dependent on pincode, which is not a key value. The State, City, and Pincode are placed in a distinct table with pincode as the primary key, preventing partial dependency.
- Driver Table: Driver ID acts as the primary key, and attributes such as Name, Phone, Email, and Location are fully dependent on it. This design guarantees the absence of partial dependency in the Driver table.
- Restaurant Table: The segregation of Restaurant ID, Name, and Cuisine from Locations/Outlets prevents partial dependency. This structure ensures that if a restaurant changes its name, it only needs updating in one place, avoiding anomalies.
- Item Table: Item ID is the primary key, and attributes like Item Name, Description, Category, and Price are fully dependent on it. This design eliminates partial dependency, ensuring that each non-key attribute depends directly on the primary key.
- Rating Table: User ID, Restaurant ID, and RatingID together form the composite primary key, preventing partial dependency.
- Order Table: Order ID serves as the primary key, and attributes like Address ID, Driver ID, Order Total, Order Status, Description, and Special Instruction are fully dependent on it.
- Order_Item Table: The Order_Item table establishes a mapping between Order ID and Item ID.
- Payment Table: Payment ID serves as the primary key, and attributes like Order ID, Mode of Payment, and Status are dependent on it.

I have tried to ensure that a non-key value does not depend on another to achieve 3NF. I have used VARCHAR data type for the Phone (storing phone numbers as strings). This decision was made for flexibility, as phone numbers can have varying formats. However, if storage efficiency is a concern, and phone numbers need to be treated as numeric for certain reasons then we should store it as a number. This would also ensure that the phone number is legitimate and doesn't have unwanted characters.