Functional Dependencies

1. Customers Table:

- Customer ID → Name, Email, Phone Number, Address
- Email → Customer ID, Name, Phone Number, Address

2. Restaurants Table:

- Restaurant ID → Name, Address, Contact Information, Opening Hours, Cuisine Type, Ratings

3. Menus Table:

- (Dish Name, Restaurant ID) → Description, Price

4. Orders Table:

- Order ID → Customer ID, Restaurant ID, Order Date, Order Item, Quantity, Delivery Address, Total Amount, Payment Status
- Customer ID → Delivery Address

5. Delivery Personnel Table:

- DeliveryPerson ID → Name, Phone Number

6. Payments Table:

- Payment ID → Order ID, Payment Method, PaymentDateTime
- OrderID → PaymentID, PaymentMethod, PaymentDateTime

7. Reviews and Ratings Table:

- ReviewID → CustomerID, RestaurantID, Rating, ReviewText, ReviewDateTime

8. Coupons Table:

- CouponID → CouponCode, DiscountAmount, ExpirationDate, MinimumOrderValue

9. Feedback for Delivery Personnel Table:

- FeedbackID → DeliveryPersonID, CustomerID, Rating, Comments, FeedbackDateTime

Normal Forms

1. Customers Table

- **1NF**: All attributes are atomic.
- **2NF**: Not in 2NF because `Email → CustomerID` creates a partial dependency, as `Email` (non-key) determines `CustomerID`.

2. Restaurants Table

- **BCNF**: The primary key determines all other attributes.

3. Menus Table

- **BCNF**: The primary key determines all other attributes.

4. Orders Table

- **1NF**: All attributes are atomic.
- **2NF**: Not in 2NF because of partial dependency of `CustomerID → DeliveryAddress`.

5. Delivery Personnel Table

- **BCNF**: The primary key determines all other attributes.

6. Payments Table

- **3NF**: It is in 3NF because there are no transitive dependencies.
- **BCNF**: Not in BCNF because `OrderID → PaymentID` violates BCNF.

7. Reviews and Ratings Table

- **BCNF**: The primary key determines all other attributes.

8. Coupons Table

- **BCNF**: The primary key determines all other attributes.

9. Feedback for Delivery Personnel Table

- **BCNF**: The primary key determines all other attributes.

Anomalies

1. Update Anomalies

- **Customers Table**: If a customer's email is updated but the email-related information is stored in multiple rows (e.g., due to partial dependencies), it could result in different email addresses for the same customer.
    - **Example**: Suppose the email for `CustomerID` is stored in multiple records. If the email address is updated in one place but not another, this inconsistency can lead to different email addresses being stored for the same customer.
- **Orders Table**: If a customer's delivery address is updated, and the delivery address is stored with each order, the database would need to update the address in multiple rows. Failing to update all instances could lead to inconsistent delivery addresses for the same customer.
    - **Example**: If the address for `CustomerID` changes, all rows in the `Orders` table that include this customer would need to be updated, or else different rows may contain different delivery addresses.

2. Delete Anomalies

- **Customers Table**: If a customer's email or address is tied directly to their orders, deleting an order might also delete the customer's email or address.
    - **Example**: Deleting a customer's order can lead to the loss of the customer's contact details if all orders for that customer are removed.

- **Payments Table**: If a payment is deleted, information about the associated order could also be deleted, since the table was not properly normalized (due to the `OrderID → PaymentID`).
    - **Example**: Deleting a payment could result in the loss of order-related data, creating gaps in the order history.

3. Insert Anomalies

- **Customers Table**: If a new customer is being added, but the design requires an order to be associated with the customer, then it would be impossible to insert the customer's details without also inserting an order.
    - **Example**: A customer can't be added unless an order is also added.
- **Orders Table**: If the schema requires the `DeliveryAddress` and `CustomerID` to be provided for every new order, but the customer doesn't have a registered address yet, this leads to an incomplete record being inserted.
    - **Example**: If a customer hasn't provided a delivery address yet, you can't insert their order without violating the schema's requirements for an address.

Redundancies

- **Redundancy in Customer Data**:

Storing customer details (like `Email`, `PhoneNumber`, and `Address`) within the `Orders`table leads to repeated entries for every order the customer places. This redundancy can cause storage inefficiency and increase the risk of data anomalies.

- **Redundancy in Payments**:

  In the `Payments` table, the `OrderID` determines `PaymentID`, which leads to repeated data. For example, associating the same order with multiple payment records is a form of redundancy.

- **Redundancy in Menus**:

  If the menu items (like `DishName`) are repeated for multiple orders from the same restaurant, this redundancy can increase the size of the database and cause problems.