Group Member Names: Raymond Li, David Xiao, Shadman Rakib

**Entities, Attributes, Primary and Foreign Keys:**

Entity: Customer

Attributes:

* Customer\_ID (Primary Key)
* Name
  + First\_name
  + last\_name
* Contact Information
  + Phone\_number
  + Email

Reasoning:

Customer requires Name, Contact Information.

Entity: Customer\_Addresses

Attributes:

* Address\_ID (Primary Key)
* Address
  + Postal\_code
  + Street\_address
  + City
  + State
* Customer\_ID (Foreign Key)

Reasoning:

Customer Addresses is a separate entity because it is a 1:M relationship and a customer can have multiple addresses. It needs address information like postal code, street address, city, state, etc.

Entity: Restaurant

Attributes:

* Restaurant\_ID (Primary Key)
* Name
* Location
  + Postal\_code
  + Street\_address
  + City
  + State
* Contact Information
  + Phone\_number
  + Email

Reasoning:

Restaurant needs a name, location, and contact information.

Entity: Menu\_Items

Attributes:

* MenuItem\_ID (Primary Key)
* Name
* Description
* Price
* Availability

Reasoning:

Restaurant has its menu and we do not want to include quantity because quantity is tied to the order. Menu items need Name description, price, and availability.

Entity: Restaurant\_Menu\_Items

Attributes:

* (Restaurant\_ID, MenuItem\_ID) (Primary Key)
* Restaurant\_ID (Foreign Key)
* MenuItem\_ID (Foreign Key)

Reasoning:

Restaurants can have many menu items. Menu items can also have many restaurants. We need to use a M:M relationship. For example, many restaurants can serve the same menu item. Then, there are fast food restaurants that share the same company which means a menu item can have multiple restaurants.

Entity: Cuisine\_Types

Attributes:

* Cuisine\_ID (Primary Key)
* Cuisine\_name

Reasoning:

Cuisine Type only needs the name such as Italian, Chinese, Korean, Mediterranean, etc.

Entity: Restaurant\_Cuisines

Attributes:

* (Restaurant\_ID, Cuisine\_ID) (Primary Key)
* Restaurant\_ID (Foreign Key)
* Cuisine\_ID (Foreign Key)

Reasoning:

Restaurants can have many cuisines. Cuisines can have many restaurants. For example, a buffet restaurant can serve Italian and Chinese food. A cuisine can also have many restaurants because many restaurants are based on the same cuisine.

Entity: Delivery\_Personnel

Attributes:

* Personnel\_ID (Primary Key)
* Name
  + First\_name
  + Last\_name
* Contact Information
  + Phone\_number
  + Email
* Availability

Reasoning:

The delivery personnel needs a name, contact information, and availability.

Entity: Payment\_Method

Attributes:

* Payment\_ID (Primary Key)
* Method\_Type
* Masked\_Details
* Provider\_Method\_ID
* Last\_Used
* Customer\_ID (Foreign Key)

Reasoning:

Payment method needs payment type can be cash, credit, PayPal, etc., It contains masked details of the payment method. Ex: \*\*\*\* \*\*\*\* \*\*\*\*\*\* 1231 To handle payment method details securely rather than storing the method/card details. We use Provider\_Method\_ID to hold a payment token instead from the provider to process the payment. Last\_Used to keep track of the payment method’s last used time. It links to the customer as a 1:M relationship because a customer can have multiple payment methods at once.

Entity: Orders

Attributes:

* Order\_ID (Primary Key)
* Created\_At
* Delivery\_time
* Real\_time\_location
* Special\_Instructions
* Status
* Charges
  + OrderItem\_Price
  + Taxes
  + Delivery Fee
* Customer\_ID (Foreign Key)
* Restaurant\_ID (Foreign Key)
* Personnel\_ID (Foreign Key)
* Delivery\_Address\_ID (Foreign Key)

Reasoning:

You need delivery time, real time location, instructions on what to do with the order for the delivery personnel, the status of the order, and the charges so you can see the price in different areas (before tax, after tax). Then you need customers, restaurants, personnel so you can see which restaurant the food is from, which customer ordered, and who is delivering the food.

Entity: Payments  
Attributes:

* Payment\_ID (Primary Key)
* Transaction\_ID
* Amount
* Payment\_Status
* Timestamp
* Order\_ID (Foreign Key)
* Payment\_Method\_ID (Foreign Key)

Reasoning:  
Separate Payment entity to handle each payment rather than storing it all in payment method or order. Contains primary key of Payment\_ID. Transaction\_ID is for external reference to transactions, such as transaction\_ID of a vendor. Amount is total amount billed, Payment\_Status is the current status of the payment. Timestamp to track when the payment was started, Order\_ID foreign key to link this to an order, Payment\_Method\_ID to link the payment method used.

Entity: Order\_Item

Attributes:

* OrderItem\_ID (Primary Key)
* Quantity
* MenuItem\_ID (Foreign Key)
* Order\_ID (Foreign Key)

Reasoning:

Order\_Item is separate from Menu\_Item because it handles quantity. If the database only had Menu\_Item, it would not make sense to have a quantity of 0 to Menu\_Item if a restaurant decides to add a new item to the menu. It also links to Orders because it calculates the quantity of the food the customer wants to buy.

**Alternate Primary and Foreign Key Format:**

Primary Keys:

Customer: Customer\_ID

Customer\_Addresses: Address\_ID

Restaurant: Restaurant\_ID

Menu\_Items: MenuItem\_ID

Restaurant\_Menu\_Items: (Restaurant\_ID, MenuItem\_ID)

Cuisine\_Types: Cuisine\_ID

Restaurant\_Cuisines: (Restaurant\_ID, Cuisine\_ID)

Delivery\_Personnel: Personnel\_ID

Payment\_Method: Payment\_ID

Orders: Order\_ID

Order\_Item: OrderItem\_ID

Foreign Keys:

Customer\_Addresses.Customer\_ID → Customer.Customer\_ID

Restaurant\_Menu\_Items.Restaurant\_ID → Restaurant.Restaurant\_ID

Restaurant\_Menu\_Items.MenuItem\_ID → Menu\_Items.MenuItem\_ID

Restaurant\_Cuisines.Restaurant\_ID → Restaurant.Restaurant\_ID

Restaurant\_Cuisines.Cuisine\_ID → Cuisine\_Types.Cuisine\_ID

Payment\_Method.Customer\_ID → Customer.Customer\_ID

Orders.Customer\_ID → Customer.Customer\_ID

Orders.Restaurant\_ID → Restaurant.Restaurant\_ID

Orders.Personnel\_ID → Delivery\_Personnel.Personnel\_ID

Order\_Item.MenuItem\_ID → Menu\_Items.MenuItem\_ID

Order\_Item.Order\_ID → Orders.Order\_ID