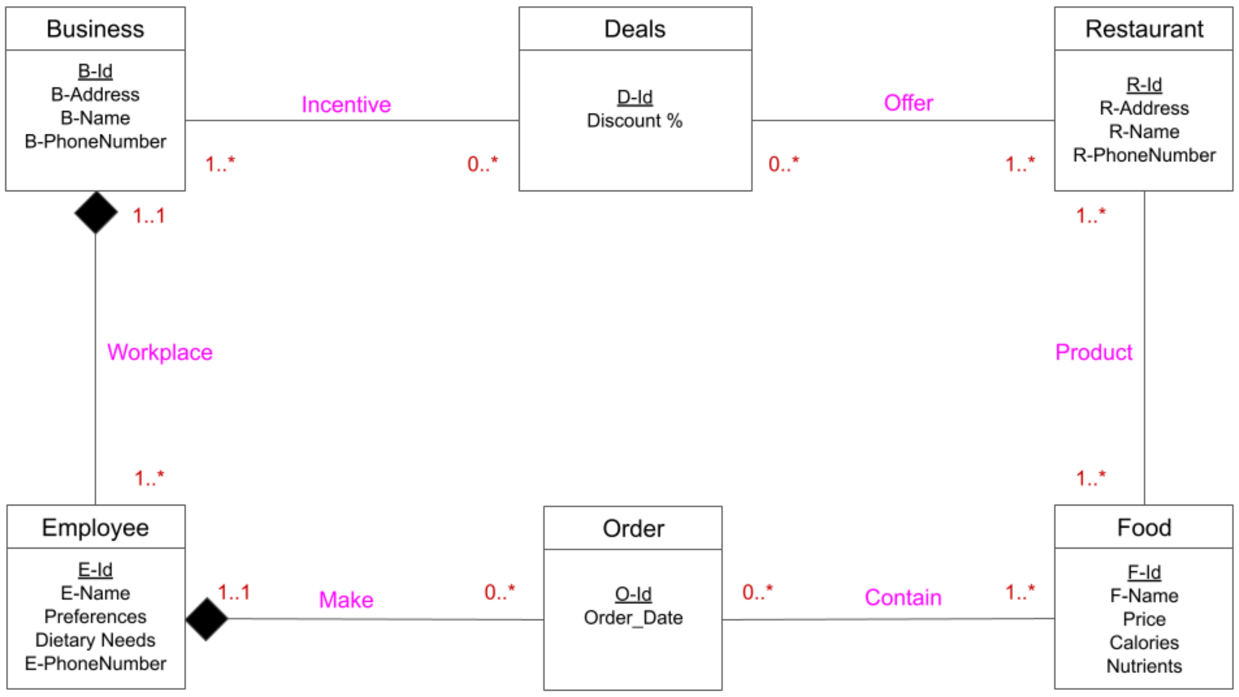
1. **UML Diagram:**



2. **Assumptions:**
   a. Employee
      i. Employee is an entity because we have to know which people ordered what food, as well as their dietary preferences and needs.
      ii. Employees must belong to exactly one business - BizBites is a platform that connects businesses with restaurants, so it wouldn't make sense to offer services to anyone who does not belong to a registered business. Additionally, employees cannot be registered with more than one business to prevent anyone from potentially doubling up on benefits.
      iii. Employees can make any number of orders - But restaurants have the right to turn down orders that are too big for them to make in time. However, there is no strict limit considering each restaurant will have a different threshold.
   b. Business
      i. Business is an entity because we have to know how many orders are placed by the employees from a particular business, and so we know the address that the food should be delivered to.
      ii. A business must have at least one employee - It wouldn't make sense to provide services to a business without any employees, so we require them to have at least one employee.
      iii. A business can have any number of deals offered to it - A business can have no deals offered to it, or have multiple deals offered to it by multiple restaurants.
   c. Restaurant

        i. Restaurant is an entity because we have to know where food should be ordered from. Without knowing contact information and address, businesses and restaurants can't be connected with each other.
        ii. Restaurants must serve at least one food as a product - It wouldn't make sense for a restaurant not to provide any food. Therefore, we require them to serve at least one food item.
        iii. Restaurants can offer any number of deals - Restaurants may not offer deals if businesses do not place enough orders. On the other hand, they may offer multiple deals depending on a business's level of patronage.
    d. Deals
        i. Deals is its own entity because we need to know which restaurant it's being offered by and what business it's being offered to.
        ii. Deals must be offered by at least one restaurant - It wouldn't make sense for a deal to not be offered by any restaurant as that would make it useless. Additionally, the same deal can be offered by several restaurants (i.e. Holiday Discounts).
        iii. Deals must be offered to at least one business - It wouldn't make sense for a deal to not be offered to any business as that would make it useless. Additionally, the same deal can be offered to several businesses.
    e. Food
        i. Food is its own entity because we need to know which restaurant it can be ordered from as well as its calories and nutritional value to make dietary recommendations.
        ii. Food must belong to at least one restaurant because it has to be ordered from somewhere. Food can also belong to multiple restaurants (i.e. same restaurant in different locations).
        iii. Food can have an order placed for it any number of times - Meaning multiple people can order the same food item. However, a restaurant has the right to turn down orders that they may not be able to prepare on time.
    f. Order
        i. Order is its own entity so we can maintain user history for employees. We want to store what foods were ordered at what date.
        ii. Each order must belong to exactly one employee - It doesn't make sense for an order to be placed by nobody, and we don't want individual orders to be associated with more than one employee for the sake of maintaining proper user history.
        iii. Each order must contain at least one food item - It doesn't make sense for an order to be placed for nothing. Additionally, orders can contain multiple food items to prevent users from having to make a separate order for every food item they want to eat.

3. **Diagram Requirements:**
    a. Our UML diagram involves at least 5 entities with at most one entity representing user information.
    b. Our UML diagram involves at least two types of relationships.

4. **3NF Normalization Rational:**

   Each primary key (Employee ID, Business ID, Deal ID, etc) is unique, making sure that no two records in the table are redundant. Given our schema a single column primary key will work

   a. Employee:
      i. Stores each employee's unique ID, name, preferences, dietary needs, and phone number.
      ii. Associates each employee with a specific business using a foreign key.
   b. Business:
      i. Captures unique business information like ID, name, address, and phone number.
      ii. Each business can have multiple employees but must have at least one.
   c. Restaurant:
      i. Holds restaurant-specific details such as ID, name, address, and phone number.
      ii. Each restaurant can offer multiple food items and deals.
   d. Deals:
      i. Represents discounts and offers made by restaurants to businesses.
      ii. Each deal is linked to a specific restaurant and a specific business.
   e. Food:
      i. Contains details of food items, including ID, name, price, calories, nutrients, and restaurant affiliation.
      ii. Each food item is linked to a restaurant using a foreign key.
   f. Orders:
      i. Records each order with details like order ID, employee ID, food ID, and the date of the order.
      ii. Links employees to food items ordered and captures order history.
5. Logical Design(Relational Schema)
   a. Business(B-Id:INT [PK], B-Address:VARCHAR(255), B-Name:VARCHAR(100), B-PhoneNumber:VARCHAR(30))
   b. Employee(E-Id:INT [PK], E-Name:VARCHAR(100), Preferences:TEXT, Dietary_Needs:TEXT, E-PhoneNumber:VARCHAR(30), B-Id:INT [FK to Business.B-Id])
   c. Restaurant(R-Id:INT [PK], R-Address:VARCHAR(255), R-Name:VARCHAR(100), R-PhoneNumber:VARCHAR(30))
   d. Deals(D-Id:INT [PK], Discount_Percentage:DECIMAL(5,2), R-Id:INT [FK to Restaurant.R-Id], B-Id:INT [FK to Business.B-Id])
   e. Food(F-Id:INT [PK], F-Name:VARCHAR(100), Price:DECIMAL(10,2), Calories:INT, Nutrients:TEXT, R-Id:INT [FK to Restaurant.R-Id])
   f. Orders(O-Id:INT [PK], E-Id:INT [FK to Employee.E-Id], F-Id:INT [FK to Food.F-Id], Order_Date:DATE)