RESTAURANT AND INVENTORY MANAGAMENT SYSTEM

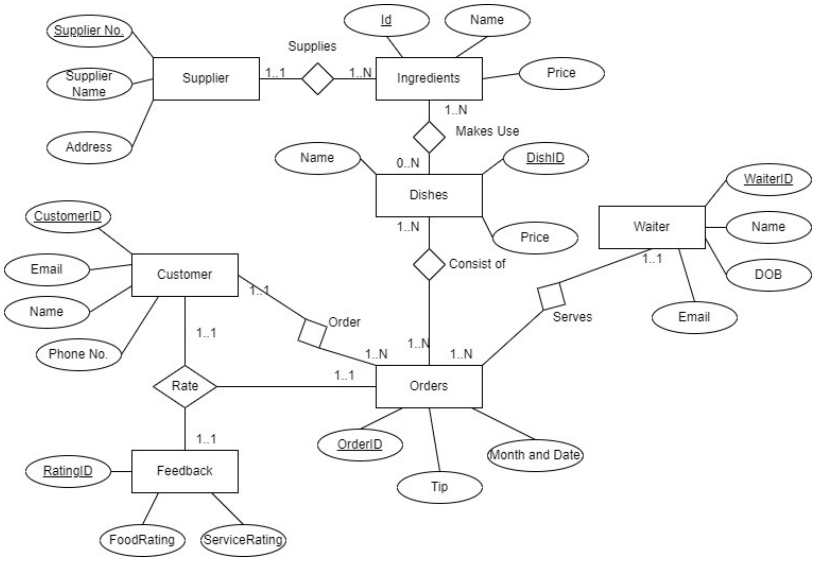
Krishna Barfiwala

I. Introduction:

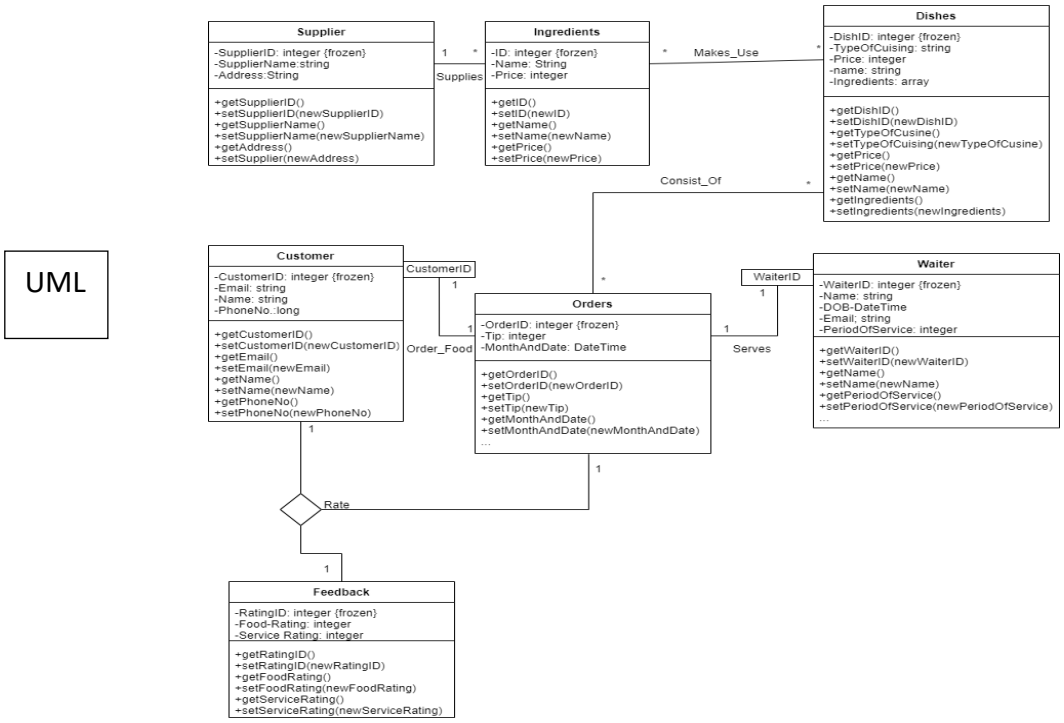
Restaurant business revolves around 2 things, balancing the books and customer satisfaction. Often, when you think about customer satisfaction, the feedback by the customer is not enough for analysis as people do not tend to review their meals openly. To grow and retain customers, it is imperative that any business evaluates several metrics including orders and inventory. Business sales are not always linear in nature, so one must continuously measure changes over time to adjust the rate and quantity at which it replenishes. Inventory control connects the upstream operations of purchasing and manufacturing to the downstream activities of sales and product demand. An insightful question would be- which is the most frequently ordered dish?

The next questions popping up as what ingredients are utilized in the most popular dish and which supplier provides the ingredients for these dishes, which cuisine is best widely ordered at what period in the year. Food is not the only factor to consider while evaluating a restaurant. Additionally, the overall dining experience must be satisfactory. Treatment of customers is equally significant. Therefore, the workers and waiters also play an important role. The performance of the waiters can be evaluated in part based on tips. We can presume that a waiter is giving good service and is crucial to the success of the restaurant if he is successful in collecting a significant tip. Our project will model a Restaurant and Inventory Management System where we will discuss the key aspects that a business can consider for better for increased productivity.

II. Conceptual Data Modelling



ER



III. Mapping Conceptual Model to Relational Model

1. **Supplier**: Supplier supplies ingredients to the restaurant. This is the entity has the following attributes.
   Supplier(SupplierID, SupplierName, Address)
   a. SupplierID: Primary key for supplier. Unique ID for a supplier
   b. Name: Name of the supplier
   c. Address: Address of the supplier

2. **Ingredients**: Refers to the ingredients being used cook the dishes. This entity includes the following attributes.
   Ingredients(ID, Name, Price, *SupplierID*)
   a. ID: Unique ID given for an ingredient
   b. Name: Name of the ingredient
   c. Price: Price of each item
   d. *SupplierID*: Foreign key in Supplier. NOT NULL

3. **Dishes**: The dishes being offered by the restaurant. This entity includes the following attributes.
   Dishes(DishID, Name, Price, TypeOfCuisine)
   a. DishID: Primary key for Dishes. Unique ID for a dish.
   b. Name: Name of the dish.
   c. Price: Price of the dish.
   d. Ingredients: List of ingredients being used in the making of the dish.
   e. Type of Cuisine: The cuisine the dish belongs to.

4. **Customer**: Refers to the customer coming to dine at the restaurant. This entity consists of the following attributes.

Customer(CustomerID, Email, Name, PhoneNo)

a. CutomerID: Primary key for Customer. Unique ID for a customer
b. Name: Name of the customer
c. Email: Email of the customer
d. Phone No: Phone number of the customer

5. **Waiters**: Refers to the waiters working at the restaurant. This entity consists of the following attributes.
   Waiter(WaiterID, Name, Email, DOB, Period of Service)
   a. WaiterID: Primary key for waiters. Unique ID for a waiter
   b. Name: Name of the waiter
   c. Email: Email of the waiter
   d. DOB: Date of Birth of the waiter
   e. Period of Service: How long the waiter has been working at the restaurant.

6. **Orders**: A customer can order a dish which will be served to him by a specific waiter. This entity consists of:
   Orders(OrderID, *CustomerID*, *WaiterID*, Tip, MonthAndDate)
   a. Month and Date: Month and date at which the order is placed
   b. OrderID: Primary key for Order. Id for each order.
   c. Tip: The amount of tip the guest wishes to pay.

7. **Feedback**: Refers to the rating the customer wishes to give for his overall experience. This entity consists of the following attributes.
   Feedback(RatingID, FoodRating, ServiceRating,*OrderID*, *CustomerID*)
   a. FeedbackID: Primary Key for feedback. Unique Id for every feedback.
   b. FoodRating: Rating of the food
   c. ServiceRating: Rating of the service

Relationships

1. Supplies: Supplier Supplies Ingredients for Restaurant.
2. Makes Use: Ingredients used to make Dish.
3. Consists of: Dishes present in the Order.
4. Order Food: Customer Orders food.
5. Gives: Customer gives Feedback for Orders.
6. Serves: Waiter serves the order to the customer.

IV. Implementation of Relation Model via MySQL and NoSQL

"MYSQL"

Q1. To find the top 3 dishes sold: Helps in predicting the most sold ingredients and time of the year to keep the inventory handy. It benefits by preventing the customer loss or customer satisfaction.

```
SELECT C.DISHID, D.NAME, COUNT(C.DISHID) AS COUNT_OF_DISH
FROM RESTAURANT.CONSIST_OF AS C, RESTAURANT.DISHES AS D
WHERE D.DISHID = C.DISHID
GROUP BY DISHID
ORDER BY COUNT(DISHID) DESC
LIMIT 3;
```

DISHID	NAME	COUNT_OF_DISH
10	Chocolate Cake	45
6	Gnochi Stuffed with crab and Chanterelle in Alfr...	41
4	Beef Tenderloin with Mini Red Potatoes	38

Q2. Most popular dishes: The most sold dish may not be the most popular one. A popular dish must be high in rating and high in sales as well. We have mapped the feedback with the order. Every dish comprising the order is mapped in the next day. These dishes not only have high sales, but they are also highly rated.

```
SELECT T.DISHID, D.NAME, COUNT(*) AS ORDER_COUNT
FROM ( SELECT T.ORDERID, T.FOODRATING, C.DISHID
        FROM (SELECT ORDERID, FOODRATING
                FROM RESTAURANT.FEEDBACK
                WHERE FOODRATING > 7) AS T,
        RESTAURANT.CONSIST_OF AS C
        WHERE C.ORDERID = T.ORDERID) AS T,
RESTAURANT.DISHES AS D
WHERE D.DISHID = T.DISHID
GROUP BY T.DISHID
ORDER BY ORDER_COUNT DESC
LIMIT 5;
```

DISHID	NAME	ORDER_COUNT
4	Beef Tenderloin with Mini Red Potatoes	20
10	Chocolate Cake	18
6	Gnochi Stuffed with crab and Chanterelle in Alfr...	18
7	Base with English Muffin and White beans	14

Q3. Total profit loss incurred: To understand the sales and corresponding actions to be taken, if any. This helps to know the success of the business and figure out the financial stability.

```
FROM(SELECT T.DISHID, T.COUNT, SUM(I.PRICE*T.COUNT) AS TOTAL_PRICE_INGREDIENTS
        FROM (SELECT DISHID, COUNT(DISHID) AS COUNT
                FROM RESTAURANT.CONSIST_OF
                GROUP BY DISHID) AS T
        INNER JOIN RESTAURANT.MAKES_USE AS M
            ON M.DISHID = T.DISHID
        INNER JOIN RESTAURANT.INGREDIENTS AS I
            ON I.ID = M.ID
        GROUP BY T.DISHID) AS T,
RESTAURANT.DISHES AS D
WHERE D.DISHID = T.DISHID
```

DISHID	COUNT	DISH_PRICE	TOTAL_PRICE_DISH	TOTAL_PRICE_INGREDIENTS	TOTAL_PROFIT_LOSS
1	29	225.02	6525.58	5438.08	1087.50
2	37	216.65	8016.05	6661.48	1354.57
3	31	134.41	4166.71	3472.31	694.40
4	38	83.04	3155.52	2629.60	525.92
5	29	199.10	5773.90	4811.68	962.22

Q4. Customer retention: Finding the topmost visiting customers is important too. Business owners want to know about the most frequently visiting customers. In times when offering discounts or promotions, we want to give more advantage to the most loyal customers. Hence it is good to know the most visiting customers.

```
/*TOP 10 VISITING CUSTOMERS*/

SELECT O.CUSTOMERID, C.NAME, COUNT(O.CUSTOMERID) AS NUMBER_OF_VISITS
FROM RESTAURANT.ORDERS AS O, RESTAURANT.CUSTOMER AS C
WHERE C.CUSTOMERID = O.CUSTOMERID
GROUP BY O.CUSTOMERID
ORDER BY COUNT(O.CUSTOMERID) DESC
LIMIT 10;
```

CUSTOMERID	NAME	NUMBER_OF_VISITS
62	Antonetta	6
46	Birgitta	5
31	Corrinne	5
41	Stanford	5
97	Hansiain	5

Q5. Supplier With most Grocery Purchase: If a dish is doing well, we would like to have its logistics clear. The ingredients used for preparing the dishes and then the suppliers who provide these ingredients. Then we group by supplier and add the total count of products that are purchased from a single supplier.

```
SELECT S.SUPPLIERID, S.SUPPLIERNAME, T.ID, SUM(T.COUNT_OF_DISH) AS GROCERIES_BOUGHT_PER_STORE
FROM (SELECT M.ID, D.COUNT_OF_DISH
        FROM RESTAURANT.MAKES_USE AS M
        INNER JOIN (SELECT DISHID, COUNT(DISHID) AS COUNT_OF_DISH
                    FROM RESTAURANT.CONSIST_OF
                    GROUP BY DISHID
                    ORDER BY COUNT(DISHID) DESC
                    LIMIT 3) AS D
                ON D.DISHID = M.DISHID
        GROUP BY M.ID) AS T,
RESTAURANT.INGREDIENTS AS I,
RESTAURANT.SUPPLIER AS S
WHERE I.ID = T.ID AND S.SUPPLIERID = I.SUPPLIERID
GROUP BY S.SUPPLIERID
ORDER BY GROCERIES_BOUGHT_PER_STORE DESC;
```

SUPPLIERID	SUPPLIERNAME	ID	GROCERIES_BOUGHT_PER_STORE
5	Mudo	91	79
23	Lajo	87	79
7	Photofeed	42	45
8	Katz	44	45
24	Bluezoom	86	45
26	Tavu	84	41
22	Wikido	89	41

Q6. Evaluate the estimate number of order a waiter serves. While hiring new waiters or laying off existing employees, if the sales is down, this information may be helpful to understand which waiters are doing well and which waiters are not doing well.

```
SELECT W.WAITERID, W.NAME, COUNT(O.ORDERID) AS WAITER_ORDERS
FROM RESTAURANT.ORDERS AS O
INNER JOIN RESTAURANT.WAITER AS W
ON (O.WAITERID = W.WAITERID)
GROUP BY W.WAITERID
```

WAITERID	NAME	WAITER_ORDERS
1	Sean	28
2	Erika	33
3	Auberon	23
4	Kingsley	31
5	Collin	44
6	Dari	20
7	Donovan	21

Q7. Finding the top 10 most visiting customers. This is mostly used in the case of customer retention. Business owners want to know about the most frequently visiting customers.

```
/*TOP 10 VISITING CUSTOMERS*/

SELECT O.CUSTOMERID, C.NAME, COUNT(O.CUSTOMERID) AS NUMBER_OF_VISITS
FROM RESTAURANT.ORDERS AS O, RESTAURANT.CUSTOMER AS C
WHERE C.CUSTOMERID = O.CUSTOMERID
GROUP BY O.CUSTOMERID
ORDER BY COUNT(O.CUSTOMERID) DESC
LIMIT 10;
```

	CUSTOMERID	NAME	NUMBER_OF_VISITS
▶	62	Antonetta	6
	46	Birgitta	5
	31	Corrinne	5
	41	Stanford	5
	97	Hansiain	5

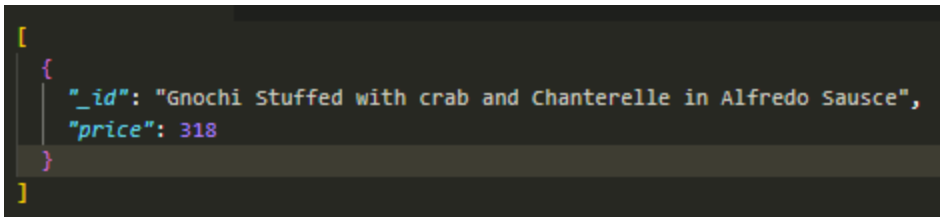
“NOSQL”

Q1. Finding the costliest dish on the menu. We would like to know which is the most expensive item on the menu.

We first grouped the Dishes cluster by price and then sorted the values in descending order. Finally, we have limited the number of output rows to 1 to find the most expensive dish on the menu.

```
use('DMAProject');

//db.Orders.find({}).toArray()

db.Dishes.aggregate([
  {
    /**
     * _id: The id of the group.
     * fieldN: The first field name.
     */
    $group: {
      _id: "$Name",
      price: {$max: "$Price"}
    }
  },
  {
    /**
     * Provide any number of field/order pairs.
     */
    $sort: {
      price: -1
    }
  },
  {
    $limit: 1
  }
])
```

Query



Output

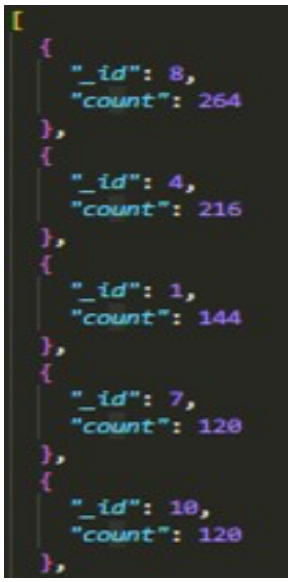
Q2. Top selling dishes in a particular period: This query sorts out the top selling dishes during a certain period (in months). Sometimes, the dishes being ordered can change with time, season etc. For eg, during the winter months, people prefer to order soups and avoid deserts. Vice-versa in the summers. We have tried to find most selling dishes during the summer months.

We first join the “Orders” cluster with “Consist_Of” cluster using $lookup. Further we join the “Dishes” cluster to this. The resulting BSON will have the Month_And_Date values in an object. To access the month in the Month_And_Date value, we have added that value as an additional field to the BSON. DishID from Consist_Of has also been added as an additional field. To consider the months 4-8, we have used $match. The resulting BSON is grouped using the “did” and the count is calculated. In the resulting BSON, we can see the id of the dish and the number of times it's been ordered between the months April to August.

```
use('DMAProject');

db.Consist_Of.aggregate(
  [
    {
      $lookup: {
        from: "Orders",
        localField: "OrderID",
        foreignField: "OrderID",
        as: "Orders"
      }
    },

    {
      $unwind: "$Orders"
    },

    {
      $lookup: {
        from: "Dishes",
        localField: "DishID",
        foreignField: "DishId",
        as: "Dishes"
      }
    },

    {$unwind: "$Dishes"},

    {
      $addFields: {
        month: {"$month":"$Orders.Month_And_Date"},
        cuisine: "$Dishes.Type of Cuisine"
```

```
      }
    },

    {
      $match: {
        month: {"$gte": 4, "$lt":8}
      }
    },

    {
      $group: {
        _id: "$cuisine",
        count: {
          $sum: 1
        }
      }
    },
  ]
).toArray()


]).toArray()
```

Query



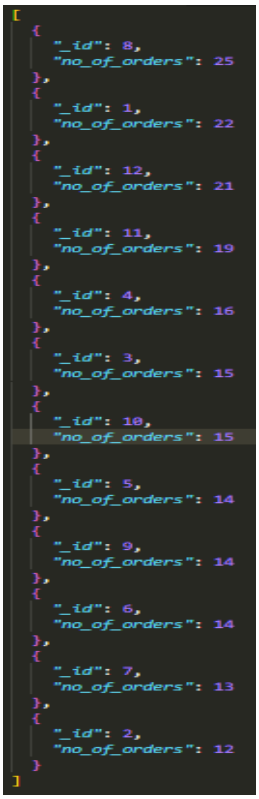
Output

Q3. In this query, we have tried to find the months with maximum productivity. Certain times of the year, such as Christmas/new year period, the influx of customers increases, and we restaurant owners need to be prepared for it.

We have used the ‘Orders’ cluster in this. We again need to access the month and therefore added a ‘month’ field for accessing the month in the Month_And_Date field. Then we have grouped by this ‘month’ field to get the number of orders per month.

```
use('DMAProject');

db.Orders.aggregate([

    {
      $addFields: {
        month: {"$month":"$Month_And_Date"}
      }
    },

    {
      $group: {
        _id: "$month",
        no_of_orders: {
          $sum: 1
        }
      }
    },

    {
      $sort:{
        no_of_orders: -1
      }
    }

  ]
)
```

Query



Output

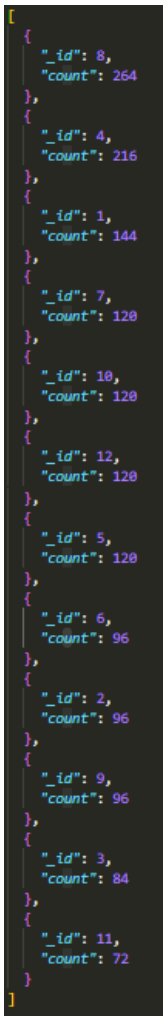
Q4. This query sorts out the top selling dishes during a certain period of time. We have considered this range in terms of months. Sometimes, the dishes being ordered can change with time, season etc. For eg, during the winter months, people prefer to order soups and avoid deserts. Vicer-versa in the summers. We have tried to find most selling dishes during the summer months(We have assumed the months to be from 4-8 i.e April-August)

We first join the “Orders” cluster with “Consist_Of” cluster using $lookup. Further we join the “Dishes” cluster to this. The resulting BSON will have the Month_And_Date values in an object. To access the month in the Month_And_Date value, we have added that value as an additional field to the BSON. DishID from Consist_Of has also been added as an additional field. To consider the months 4-8, we have used $match. The resulting BSON is grouped using the “did” and the count is calculated. In the resulting BSON, we can see the id of the dish and the number of times it's been ordered between the months April to August.

```
use('DMAProject');


db.Orders.aggregate([

  {$lookup:
    {
      from: "Consist_Of",
      localField: "OrderID",
      foreignField: "OrderID",
      as: "Consists"
    }
  },

  {
    $unwind: "$Consists"
  },

  {$lookup:
    {
      from: "Dishes",
      localField: "DishID",
      foreignField: "DishID",
      as: "Dish_Details"
    }
  },

  {
    $unwind: "$Dish_Details"
  },

  {
  $addFields: {
    month: {"$month": "$Month_And_Date"},
    did: "$Consists.DishID"
  }},

  {
  $match: {
    month: {"$gte": 4, "$lt":8}
  }},
```

```
  {
    $group: {
      _id: "$did",
      count: {
        $sum: 1
      }
    }
  },

  {
    $sort:{
      count: -1
    }
  }


]).toArray()
```

Query



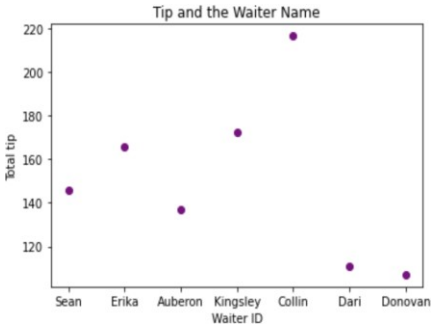
Output

V. Database access via Python

```
In [1]: host= os.getenv( 'Local' )
        conn= pymysql.connect(
            host=host,
            port=int (3306),
            user= "root",
            passwd='IE6700user',
            db="restaurant",
            charset='utf8mb4')
```

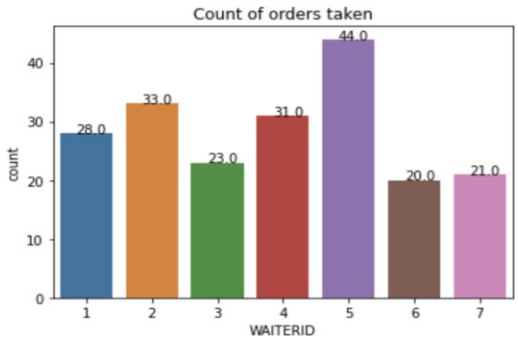
Connecting to MYSQL

```
In [227]: #Scatter plot for waiter id and the tip
          import matplotlib.pyplot as plt
          plt.scatter(x=grouped['NAME'], y=grouped['TIP'], color='purple')
          ax.set_facecolor('white')
          plt.xlabel('Waiter ID')
          plt.ylabel('Total tip (in $)')
          plt.title('Tip and the Waiter Name')
          plt.show()
```



Tip and Waiter Name

```
In [152]: #how many times a waiter has taken order
          import seaborn as sns
          a=sns.countplot(data=orders, x="WAITERID")
          for p in a.patches:
             a.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
          a.set_title('Count of orders taken')
```



Orders vs Waiter ID

- From the previous 2 graphs, we can clearly induce that Collin is the best performing waiter. He has collected the most tip of all waiters and taken the most orders.

- From graph 1, we can see that Kingsley and Erika earn almost similar amounts in terms of tip. From graph II, we can induce that Kingsley has earned the same amount of tip in fewer number of orders. From this, we can induce that Kingsley is performing better than Erika because Kingsley is earning higher tip on average in comparison to Erika.

VI. Summary and Recommendation:

- This model helps in connecting the upstream operations of purchasing and manufacturing to the downstream activities of sales and product demand. A dish with poor sales could indicate that there is potential for improvement or that the menu needs to be altered.

- For the most selling dish in the respective period, the owner needs to be well prepared with its ingredients so that there is enough satisfaction from the customers.

- This model draws few conclusions based on customer reviews, which might be fallacious and lead to flawed decisions.