

Project Python Foundations: FoodHub Data Analysis

Context

The number of restaurants in New York is increasing day by day. Lots of students and busy professionals rely on those restaurants due to their hectic lifestyles. Online food delivery service is a great option for them. It provides them with good food from their favorite restaurants. A food aggregator company FoodHub offers access to multiple restaurants through a single smartphone app.

The app allows the restaurants to receive a direct online order from a customer. The app assigns a delivery person from the company to pick up the order after it is confirmed by the restaurant. The delivery person then uses the map to reach the restaurant and waits for the food package. Once the food package is handed over to the delivery person, he/she confirms the pick-up in the app and travels to the customer's location to deliver the food. The delivery person confirms the drop-off in the app after delivering the food package to the customer. The customer can rate the order in the app. The food aggregator earns money by collecting a fixed margin of the delivery order from the restaurants.

Objective

The food aggregator company has stored the data of the different orders made by the registered customers in their online portal. They want to analyze the data to get a fair idea about the demand of different restaurants which will help them in enhancing their customer experience. Suppose you are hired as a Data Scientist in this company and the Data Science team has shared some of the key questions that need to be answered. Perform the data analysis to find answers to these questions that will help the company to improve the business.

Data Description

The data contains the different data related to a food order. The detailed data dictionary is given below.

Data Dictionary

- `order_id`: Unique ID of the order
- `customer_id`: ID of the customer who ordered the food
- `restaurant_name`: Name of the restaurant
- `cuisine_type`: Cuisine ordered by the customer
- `cost_of_the_order`: Cost of the order

- `day_of_the_week`: Indicates whether the order is placed on a weekday or weekend (The weekday is from Monday to Friday and the weekend is Saturday and Sunday)
- `rating`: Rating given by the customer out of 5
- `food_preparation_time`: Time (in minutes) taken by the restaurant to prepare the food. This is calculated by taking the difference between the timestamps of the restaurant's order confirmation and the delivery person's pick-up confirmation.
- `delivery_time`: Time (in minutes) taken by the delivery person to deliver the food package. This is calculated by taking the difference between the timestamps of the delivery person's pick-up confirmation and drop-off information

Let us start by importing the required libraries

```
In [28]: # Installing the Libraries with the specified version.
!pip install numpy==1.25.2 pandas==1.5.3 matplotlib==3.7.1 seaborn==0.13.1 -q --
```

Note: After running the above cell, kindly restart the notebook kernel and run all cells sequentially from the start again.

```
In [2]: # import libraries for data manipulation
import numpy as np
import pandas as pd

# import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

Understanding the structure of the data

```
In [3]: # uncomment and run the following lines for Google Colab
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call `drive.mount("/content/drive", force_remount=True)`.


```
In [4]: # Write your code here to read the data

#data = pd.read_csv('/content/drive/MyDrive/FoodHub/foodhub_oder.csv')
data = pd.read_csv("/content/drive/MyDrive/FoodHub/foodhub_order.csv")
```

```
In [5]: # Write your code here to view the first 5 rows
#To get the first 5 rows, the head() function will be used
data.head()
```

Out[5]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_v
0	1477147	337525	Hangawi	Korean	30.75	Weel
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weel
2	1477070	66393	Cafe Habana	Mexican	12.23	Wee
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weel
4	1478249	76942	Dirty Bird to Go	American	11.59	Wee



Question 1: How many rows and columns are present in the data? [0.5 mark]

In [6]: *# Write your code here*
#To identify the number of rows and columns, the shape attributes will be used

```
print(data.shape)

print("There are", data.shape[0], 'rows and', data.shape[1], "columns.")
```

(1898, 9)

There are 1898 rows and 9 columns.

Observations:

The data has 1898 rows and 9 columns

Question 2: What are the datatypes of the different columns in the dataset? (The info() function can be used) [0.5 mark]

In [7]: *# Write your code here*
#To understand different data types in the dataset, the info() function will be

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   order_id              1898 non-null   int64
1   customer_id           1898 non-null   int64
2   restaurant_name       1898 non-null   object
3   cuisine_type          1898 non-null   object
4   cost_of_the_order     1898 non-null   float64
5   day_of_the_week       1898 non-null   object
6   rating                1898 non-null   object
7   food_preparation_time 1898 non-null   int64
8   delivery_time         1898 non-null   int64
dtypes: float64(1), int64(4), object(4)
memory usage: 133.6+ KB
```

Observations:

There are 5 numeric (int and float) columns in the data and 4 object type columns

numerical columns: order_id, customer_id, food_preparation_time, delivery_time and 'cost of the order'

Object type: restaurant_name, cuisine_type, day_of_the_week and rating

Question 3: Are there any missing values in the data? If yes, treat them using an appropriate method. [1 mark]

```
In [8]: # Write your code here
#The missing data can be identified using isnull() and sum()

print(data.isnull().sum())
```

```
order_id          0
customer_id       0
restaurant_name   0
cuisine_type      0
cost_of_the_order 0
day_of_the_week   0
rating            0
food_preparation_time 0
delivery_time     0
dtype: int64
```

Observations:

The dataset has no missing values in any of the columns

Question 4: Check the statistical summary of the data. What is the minimum, average, and maximum time it takes for food to be prepared once an order is placed? [2 marks]

```
In [9]: # Write your code here
#To get the statistical summary of the data of numerical attributes, the describ
```

```
print(data.describe().T)
```

```
#We can derive from the above invocation.. but the option to get 'food preparation time'  
print("\n\nFood Preparation Time stats\n", data['food_preparation_time'].describe())
```

	count	mean	std	min \
order_id	1898.0	1.477496e+06	548.049724	1476547.00
customer_id	1898.0	1.711685e+05	113698.139743	1311.00
cost_of_the_order	1898.0	1.649885e+01	7.483812	4.47
food_preparation_time	1898.0	2.737197e+01	4.632481	20.00
delivery_time	1898.0	2.416175e+01	4.972637	15.00

	25%	50%	75%	max
order_id	1477021.25	1477495.50	1.477970e+06	1478444.00
customer_id	77787.75	128600.00	2.705250e+05	405334.00
cost_of_the_order	12.08	14.14	2.229750e+01	35.41
food_preparation_time	23.00	27.00	3.100000e+01	35.00
delivery_time	20.00	25.00	2.800000e+01	33.00

Food Preparation Time stats

count	1898.000000
mean	27.371970
std	4.632481
min	20.000000
25%	23.000000
50%	27.000000
75%	31.000000
max	35.000000

Name: food_preparation_time, dtype: float64

Observations:

Once the order is placed, Food preparation time are as follows

Minimum: 20

Average Time: 27.37

Max Time: 35

Question 5: How many orders are not rated? [1 mark]

```
In [10]: # Write the code here  
print("The unique values of the ratings are as follows\n", data['rating'].unique())  
print (" \nValue Counts", data['rating'].value_counts())  
print (" \nValue Counts Percentage", data['rating'].value_counts(normalize=True))
```

The unique values of the ratings are as follows
['Not given' '5' '3' '4']

```
Value Counts rating
Not given      736
5              588
4              386
3              188
Name: count, dtype: int64
```

```
Value Counts Percentage rating
Not given      38.777661
5              30.979979
4              20.337197
3              9.905163
Name: proportion, dtype: float64
```

Observations:

As shown above --> How many orders are not rated = 736 (38.77%)... .

data.info():

6 rating 1898 non-null object

This show that the ratings is currently stored as non numerical (categorical column) as an object type.

If a transaction is not rated by customer, it will be stored as "Not given"

Based on the data, the rating column has 4 distinct values...3, 4, 5 and 'Not given'...

Many customers have not rated the transactions ('Not given') (around 38.77%)... and those who rated seems to be happy customers since they rated as 5 (30.977%) and 4 (20.337%)... Among the rated customers,9.905% customers rated as 3...

As shown above --> How many orders are not rated = 736 (38.77%)... . This indicates that a large portion of customers have opted out from providing ratings.

Exploratory Data Analysis (EDA)

Univariate Analysis

Question 6: Explore all the variables and provide observations on their distributions. (Generally, histograms, boxplots, countplots, etc. are used for univariate exploration.) [9 marks]

```
In [11]: # Write the code here
# As part of univariate, we will perform the analysis on each column/attribute

#1. Order_Id
#We will check if there are any duplicate order ids
print("Duplicate order Ids: ", data['order_id'].duplicated().sum())
```

```

#2. Customer_id
#Now we will do analysis on customer_id
print("\nRepeated Customers: ", data['customer_id'].duplicated().sum())
print("Unique Customers: ", data['customer_id'].nunique())

top_customers = data['customer_id'].value_counts().head(5)
print("Top 5 Customers by Number of Orders:\n")
print(top_customers)

#Plot the top 5 customers
sns.barplot(x=top_customers.index.astype(str), y=top_customers.values)
plt.title('Top 5 Customers by Number of Orders')
plt.xlabel('Customer ID')
plt.ylabel('Number of Orders')
plt.show()

#3. Restaurant name related
#Let us see how many unique restaurant names are part of Foodhub
print("\n The number of unique restaurants on the FoodHub platform:", data['rest

#Plot the top 5 restaurants
top_restaurants = data['restaurant_name'].value_counts().head(10)
top_restaurants.plot(kind='bar', figsize=(10, 6))
plt.title('Top 10 Restaurants by Number of Orders')
plt.xlabel('Restaurant Name')
plt.ylabel('Number of Orders')
plt.show()

#4. Cuisine Type
#Let us see how many unique cuisine types are part of Foodhub
print("\n The number of unique cuisine types on the FoodHub platform:", data['cu

# Plot countplot for cuisine type
plt.figure(figsize=(15, 5))
plt.xticks(rotation=45)
sns.countplot(x='cuisine_type', data=data, order=data['cuisine_type'].value_coun
plt.title('Count of Orders by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Number of Orders')
plt.tight_layout()
plt.show()

#5. Let us try to analyze cost of the order now... by having Histogram and box p
#Histogram for cost of the order
plt.figure(figsize=(6, 4))
sns.histplot(data['cost_of_the_order'], bins=30, kde=True)
plt.title('Distribution of Cost of the Order')
plt.xlabel('Cost ($)')
plt.tight_layout()
plt.show()

# Boxplot for Cost of the Order
plt.figure(figsize=(6, 2))
sns.boxplot(x=data['cost_of_the_order'])
plt.title('Boxplot of Cost of the Order')
plt.xlabel('Cost ($)')
plt.tight_layout()
plt.show()

```

#6. Day of the week

```
print("\n The unique values in day of the week:", data['day_of_the_week'].unique)
```

#countplot

```
plt.figure(figsize=(6, 4))
sns.countplot(x='day_of_the_week', data=data)
plt.title('Count: Orders by Day of the Week')
plt.tight_layout()
plt.show()
```

#7. Ratings

```
print("\n The unique values in 'rating' column:", data['rating'].unique())
```

#Countplot: Rating

```
plt.figure(figsize=(6, 4))
sns.countplot(x='rating', data=data, order=['Not given', '3', '4', '5'])
plt.title('Count: Customer Ratings')
plt.tight_layout()
plt.show()
```

*#8. Let us try to analyze 'Food Preparation Time'... by having Histogram and box
Histogram*

```
plt.figure(figsize=(6, 4))
sns.histplot(data['food_preparation_time'], bins=30, kde=True)
plt.title('Distribution of Food Preparation Time')
plt.xlabel('Time (minutes)')
plt.tight_layout()
plt.show()
```

Boxplot

```
plt.figure(figsize=(6, 4))
sns.boxplot(x=data['food_preparation_time'])
plt.title('Boxplot of Food Preparation Time')
plt.xlabel('Time (minutes)')
plt.tight_layout()
plt.show()
```

#9. Histogram and Boxplot for Delivery Time

Histogram

```
plt.figure(figsize=(6, 4))
sns.histplot(data['delivery_time'], bins=30, kde=True)
plt.title('Distribution of Delivery Time')
plt.xlabel('Time (minutes)')
plt.tight_layout()
plt.show()
```

Boxplot

```
plt.figure(figsize=(6, 4))
sns.boxplot(x=data['delivery_time'])
plt.title('Boxplot of Delivery Time')
plt.xlabel('Time (minutes)')
plt.tight_layout()
plt.show()
```


Duplicate order Ids: 0

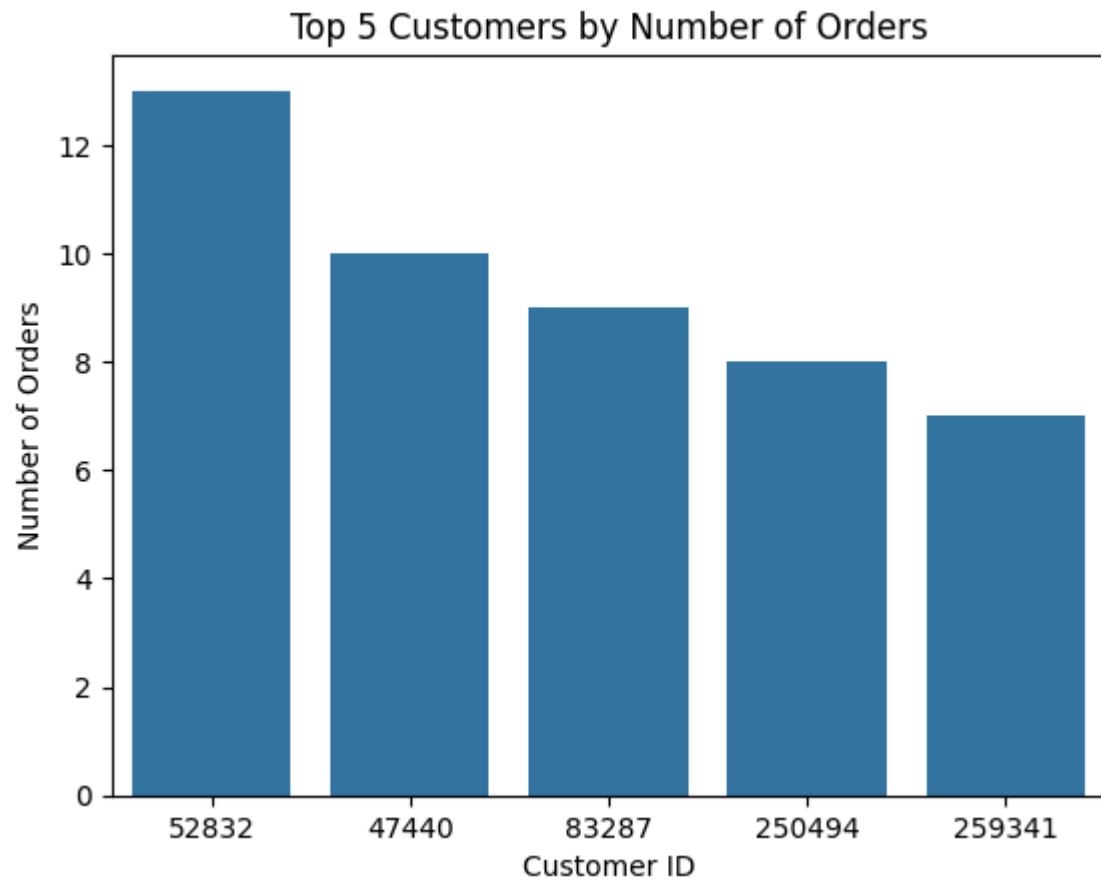
Repeated Customers: 698

Unique Customers: 1200

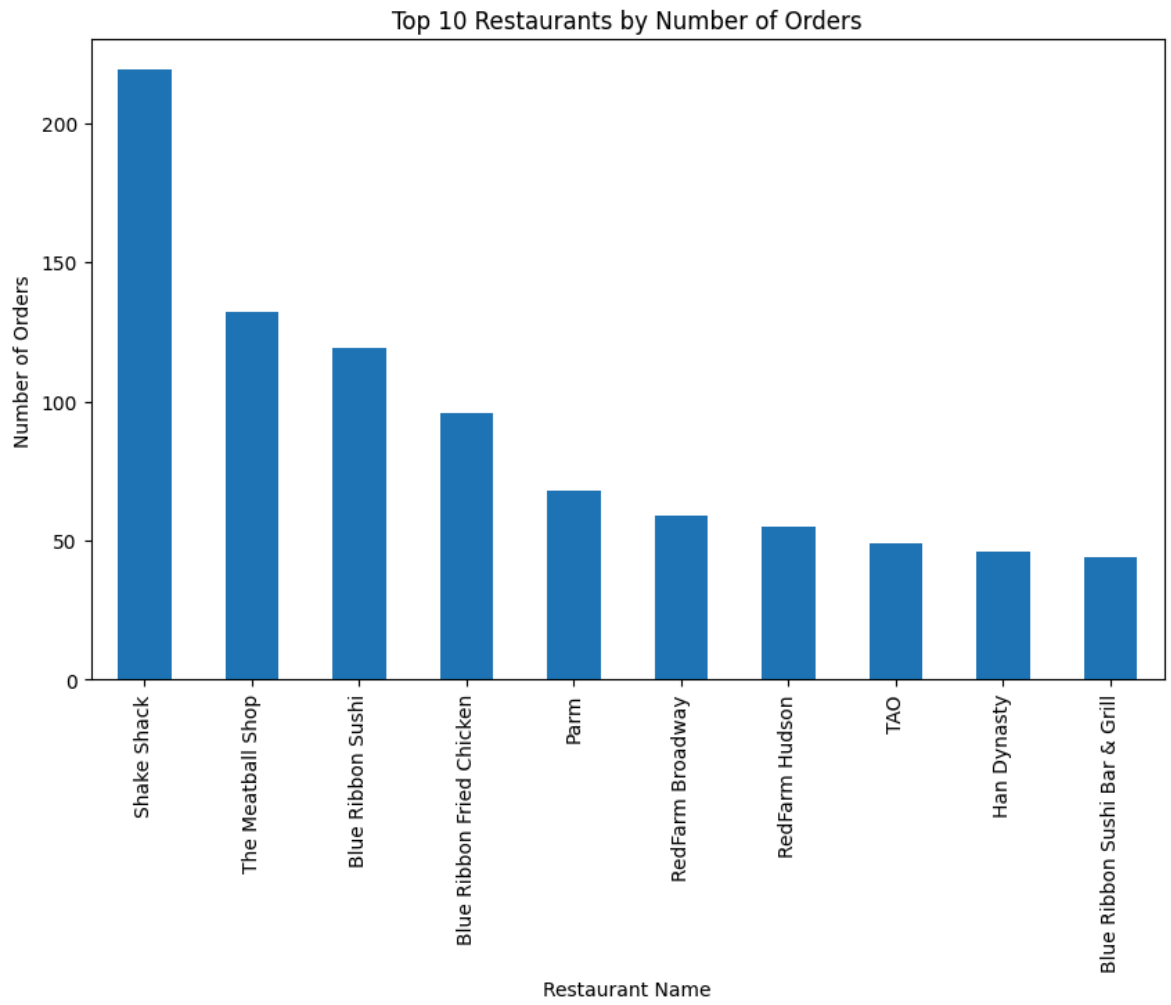
Top 5 Customers by Number of Orders:

customer_id	
52832	13
47440	10
83287	9
250494	8
259341	7

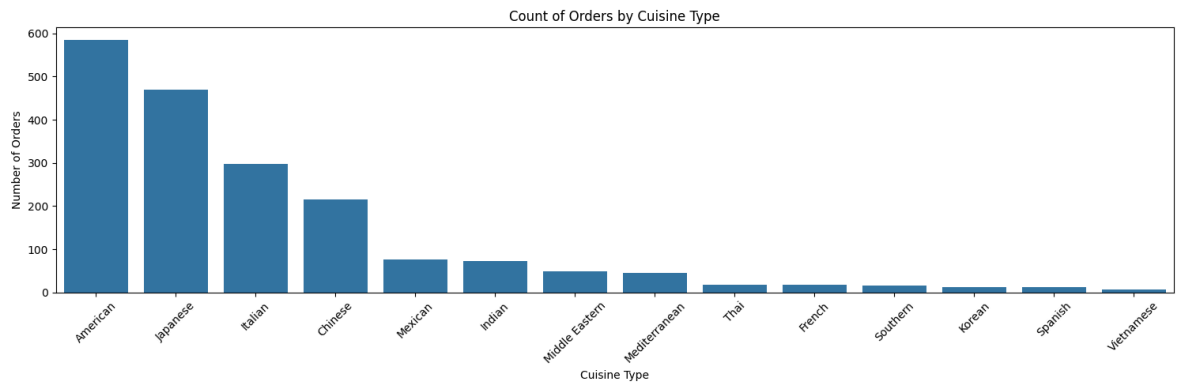
Name: count, dtype: int64

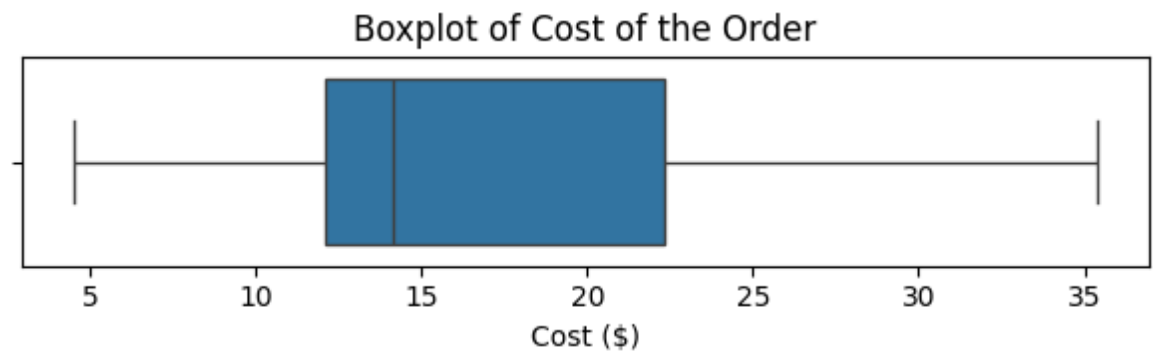
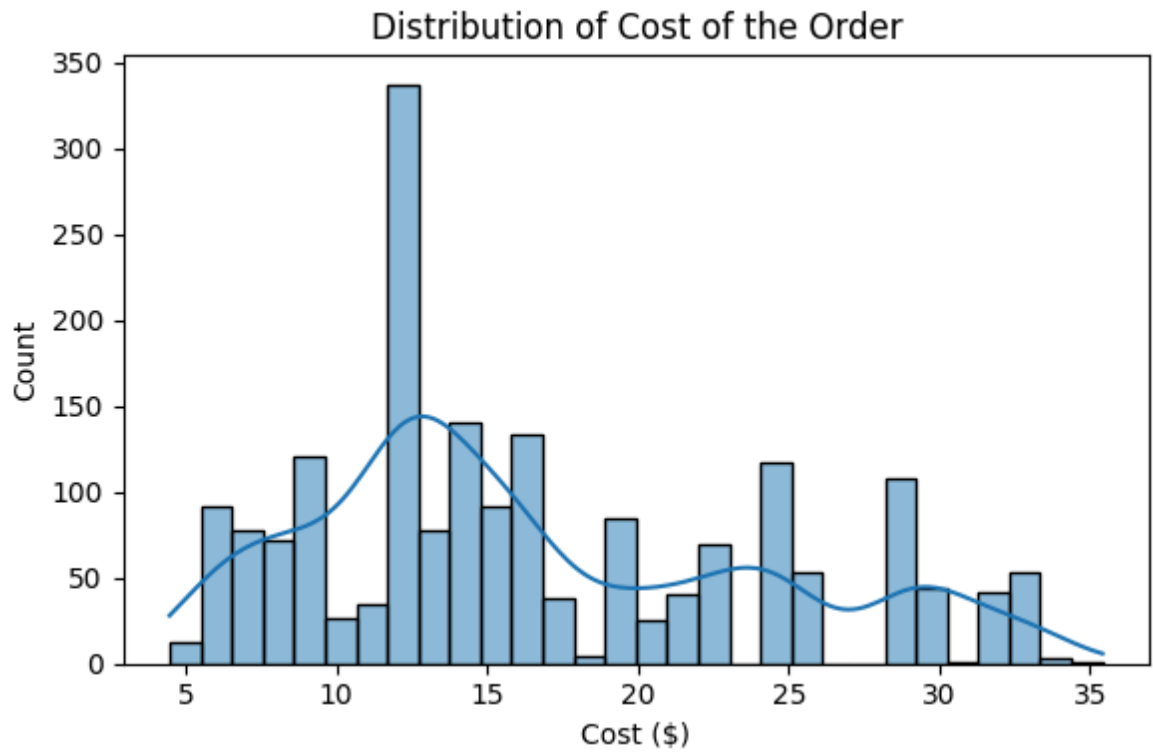


The number of unique restaurants on the FoodHub platform: 178

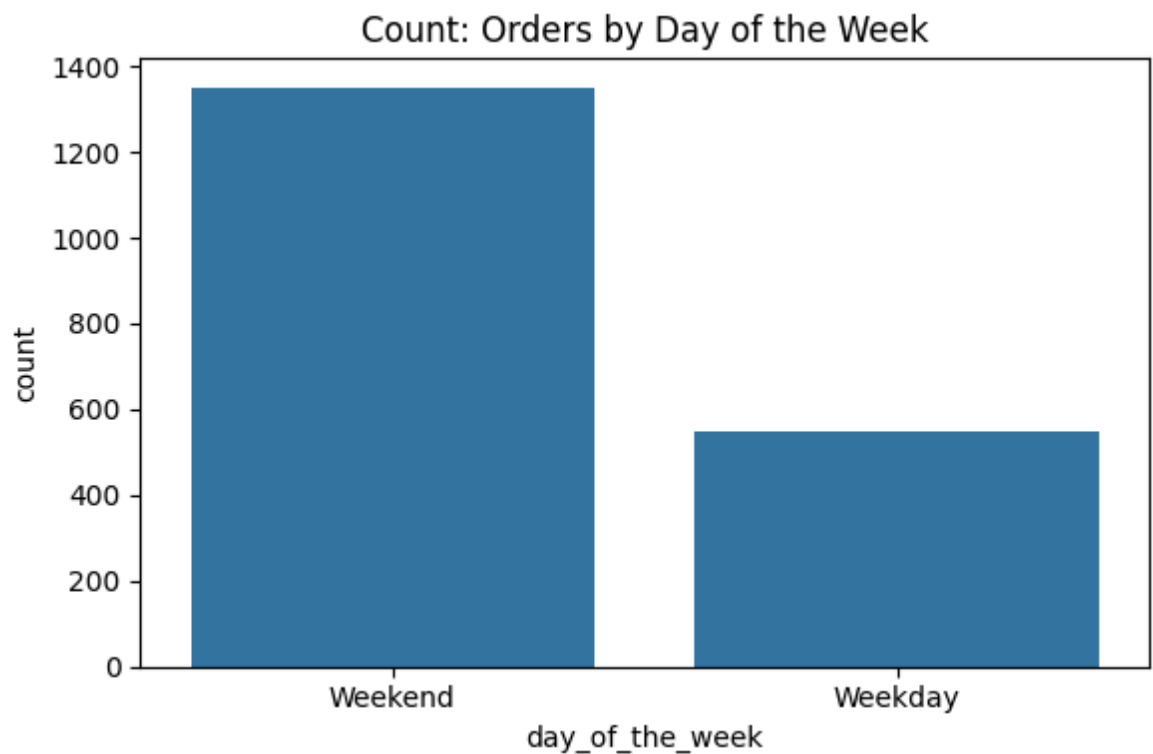


The number of unique cuisine types on the FoodHub platform: 14



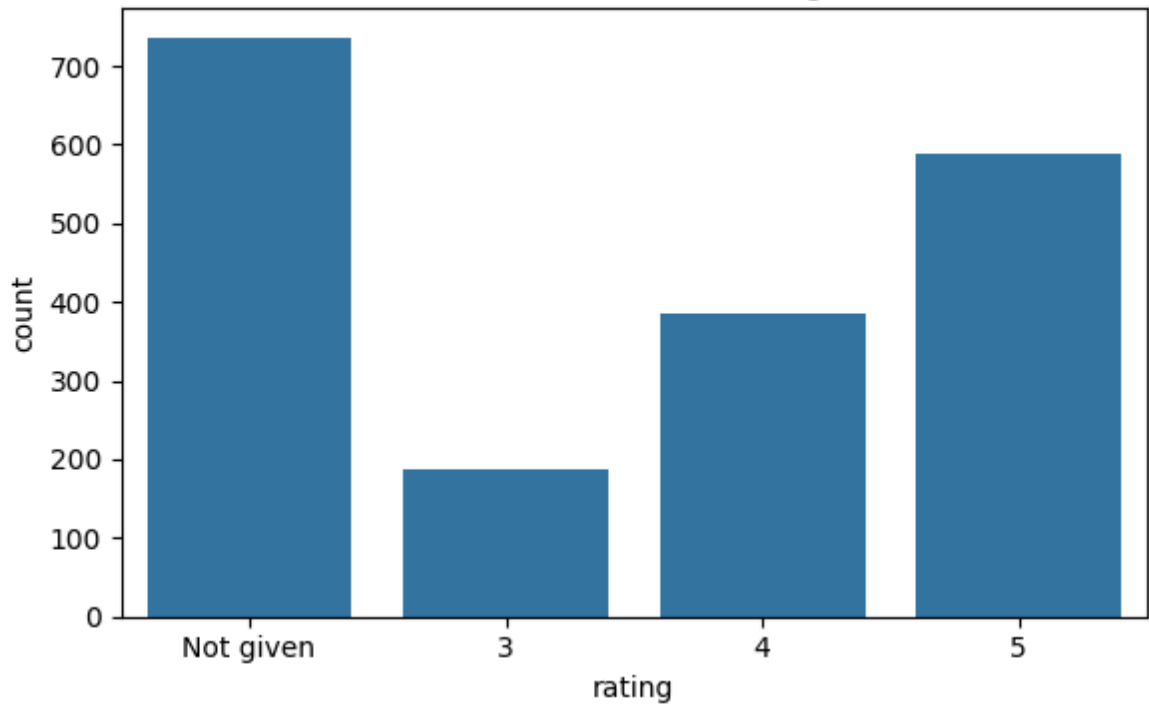


The unique values in day of the week: ['Weekend' 'Weekday']

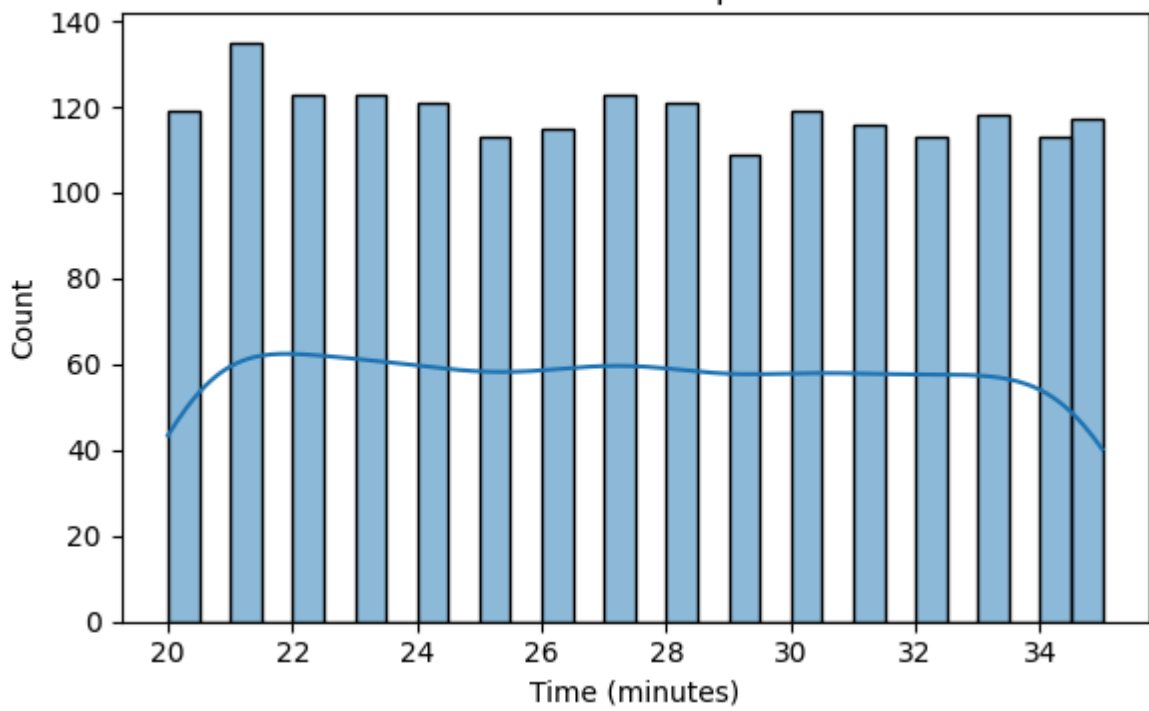


The unique values in 'rating' column: ['Not given' '5' '3' '4']

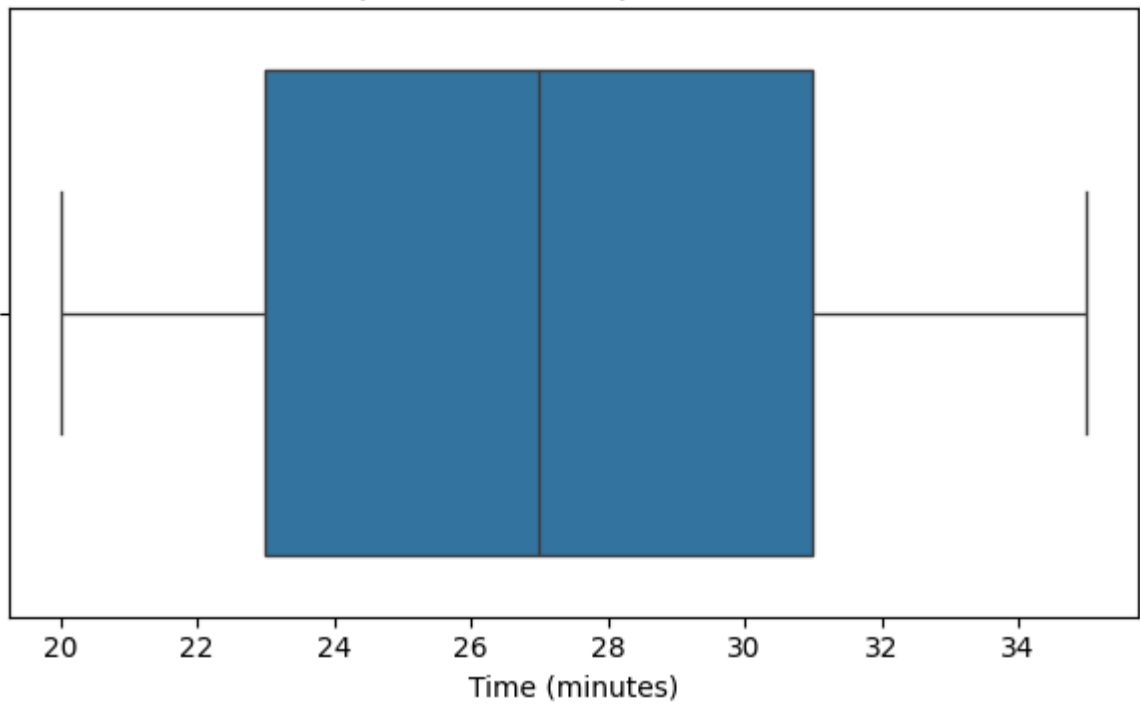
Count: Customer Ratings



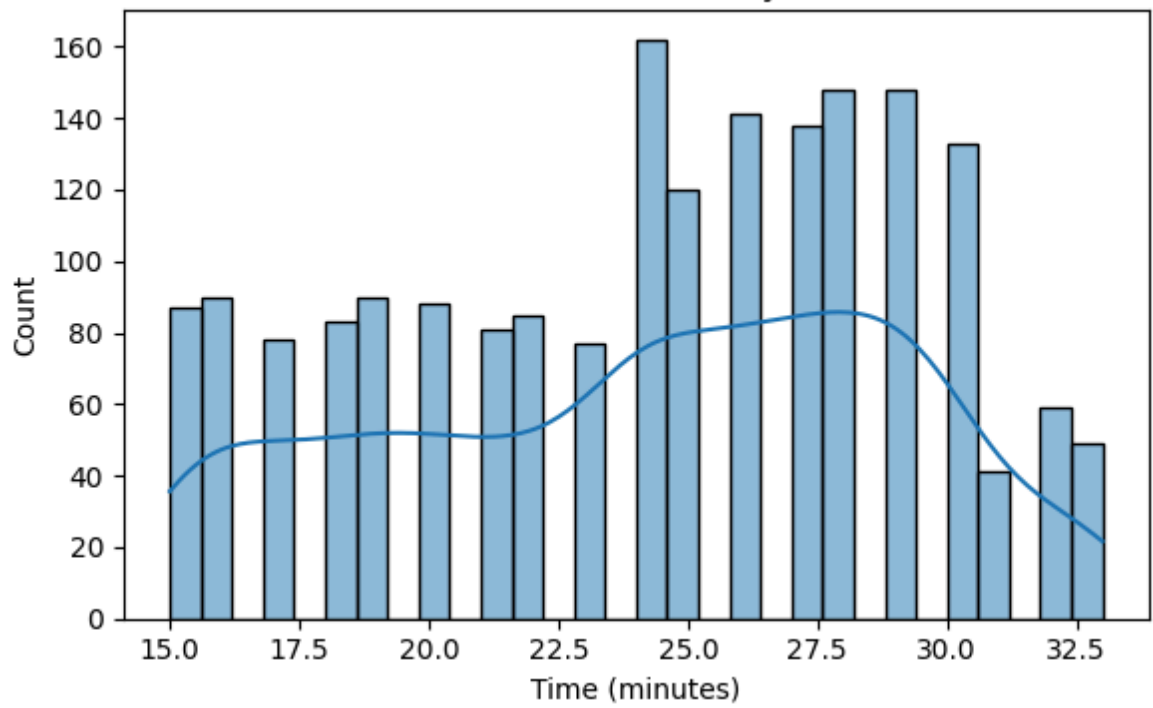
Distribution of Food Preparation Time

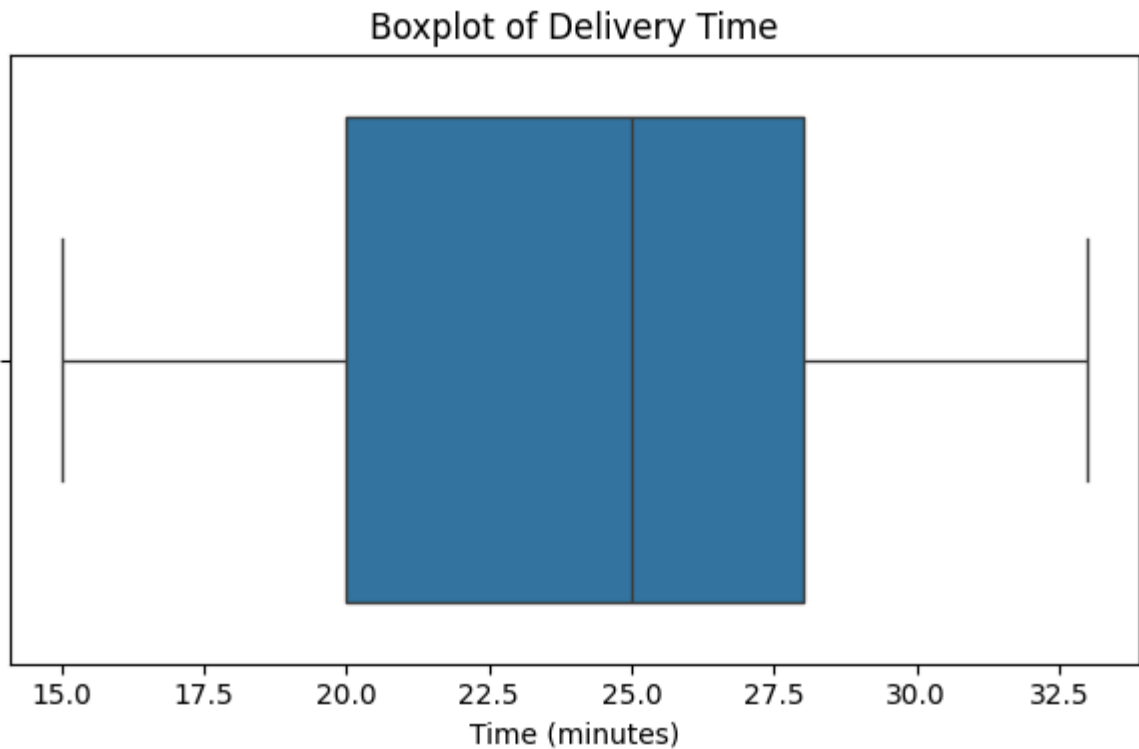


Boxplot of Food Preparation Time



Distribution of Delivery Time





Observation:

1. Order_Id: In the beginning we noticed that there is no missing data for any of the column. In addition we can now notice that there are 0 duplicate order Ids. So From the data sanity perspective, this looks good.
2. Customer_Id: The Food hub has 1200 unique customers and it also has some repeated customers (698) which is good sign... These repeated customers are placing orders multiple times and the top 5 repeated customers are shown in the bar plot.
3. Restaurant_Name: The number of unique restaurants on the FoodHub platform: 178 and 'Shake Shack' restaurant stands out in terms of number of orders. The top restaurants are shown in the bar plot.
4. Cuisine_Type: There are 14 unique types of cuisines available on the Foodhub platform. and American, Japanese and Italian are most frequently ordered cuisines.. The least frequent being Vietnamese, Spanish. The number of orders for each cuisine_type is shown using count plot.
5. Cost of the Order: The majority of orders are in the range of 10-20. However from box plot, it can be observed that it is right skewed meaning there are some orders more expensive (over 30). Also the distribution is NOT symmetrical.
6. Days of the Week: The data is managed through 'weekday' versus 'weekend'. The unique values in this column are 'weekday' and 'weekend' only. The countplot shows that there are more orders on 'weekends' compared to 'weekday'
7. Rating: The unique values in 'rating' column: ['Not given' '5' '3' '4'] The count plot shows that many customers have rated the transactions with 5. So among the rated

orders 5 is most the common followed by 4 and 3. This shows that the customer satisfaction is high but many users are not rating their orders.

8. Food_Preparation_Time: The median prep time is around 27 mins... The distribution peaks around 25-30 mins. it is fairly symmetric with slightly right skewed.

9. delivery_time: The median delivery is around 24-25 minutes.. The box plot shows slightly left skewed

Question 7: Which are the top 5 restaurants in terms of the number of orders received? [1 mark]

```
In [12]: # Write the code here
#Top 5 restaurants can be found by value_counts()

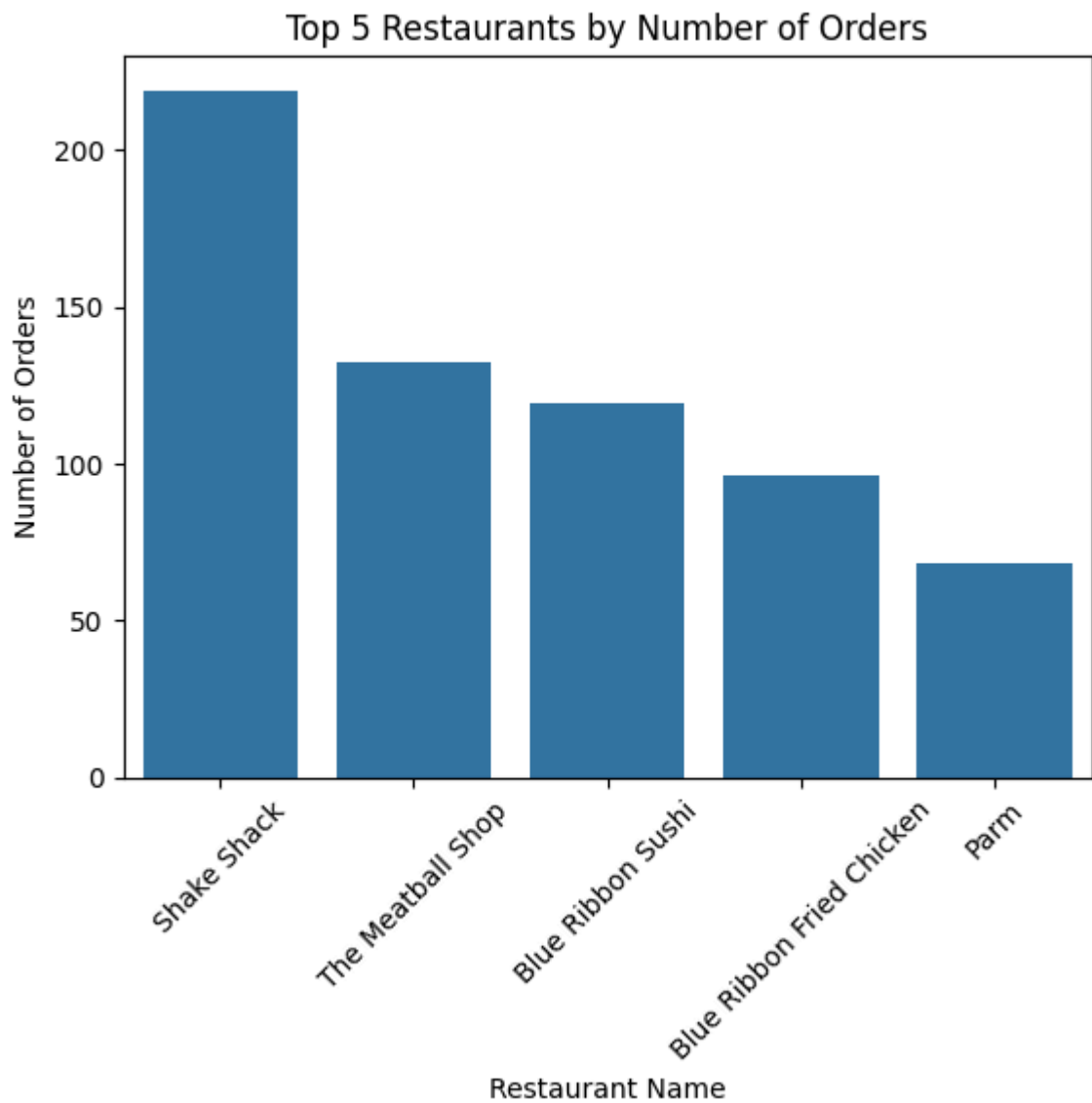
top_restaurants = data['restaurant_name'].value_counts().head(5)
print("Top 5 Restaurants by Number of Orders:")
print(top_restaurants)

#This can also be understood by plotting them in barplot
sns.barplot(x=top_restaurants.index, y=top_restaurants.values)
plt.title('\nTop 5 Restaurants by Number of Orders')
plt.xlabel('Restaurant Name')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.show();
```

Top 5 Restaurants by Number of Orders:

restaurant_name	
Shake Shack	219
The Meatball Shop	132
Blue Ribbon Sushi	119
Blue Ribbon Fried Chicken	96
Parm	68

Name: count, dtype: int64



Observations:

Top 5 Restaurants by Number of Orders: restaurant_name Shake Shack 219

The Meatball Shop 132

Blue Ribbon Sushi 119

Blue Ribbon Fried Chicken 96

Parm 68

Shake Shack leads the chart with 219 orders, followed by The Meatball Shop, Blue Ribbon Sushi, Blue Ribbon Fred Chicken and Parm.... These restaurants are the most in-demand in FoodHub platform.

Question 8: Which is the most popular cuisine on weekends? [1 mark]


```
In [13]: # Write the code here
#To answer this, we need to get the data of the weekend.
print("The unique values of the the days are as follows\n", data['day_of_the_week'].value_counts())

# Filter data for weekends only
weekend_data = data[data['day_of_the_week'] == 'Weekend']

# Find the most popular cuisine on weekends
weekend_cuisine_counts = weekend_data['cuisine_type'].value_counts()

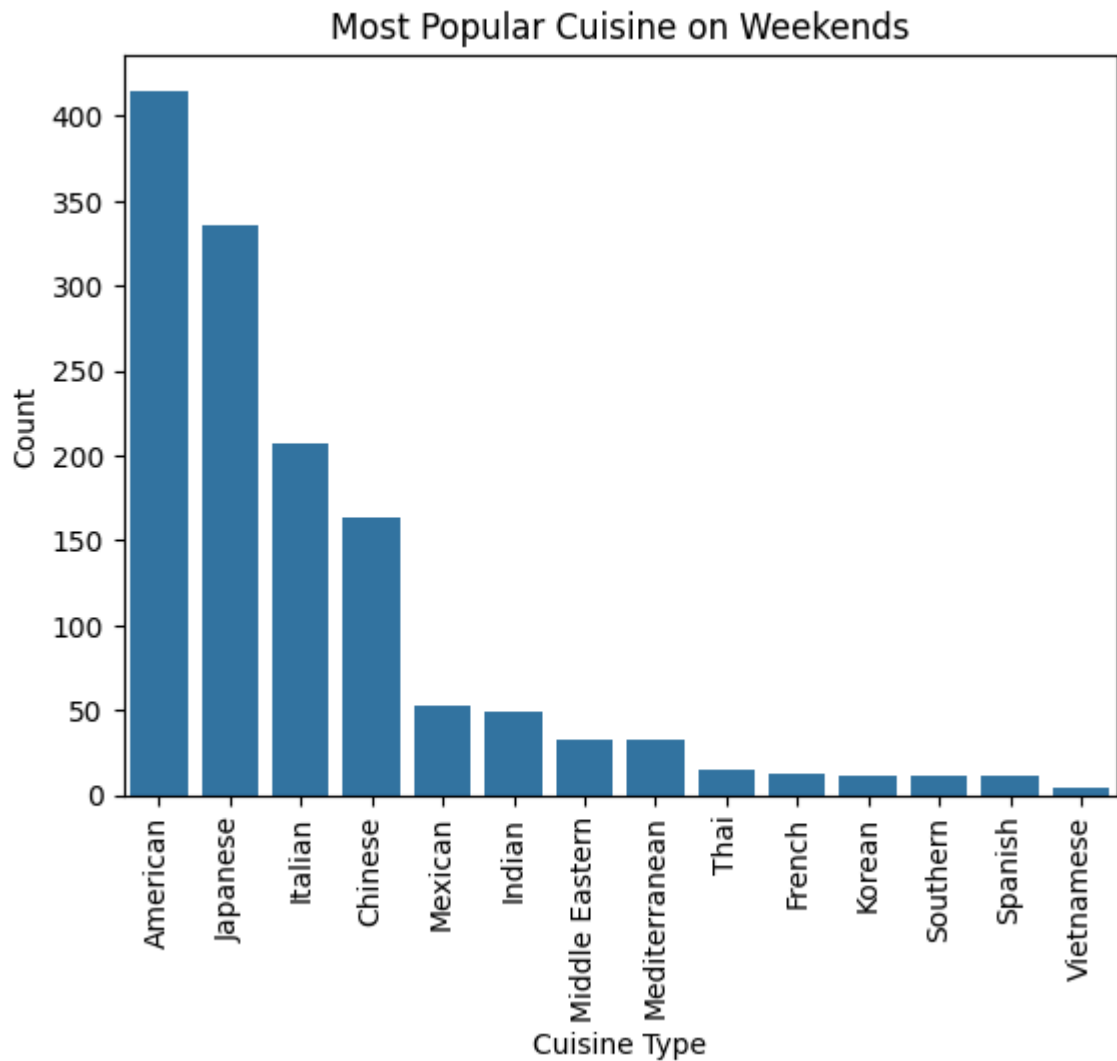
#Now we are going to have the count plot to understand the most popular cuisine
sns.countplot(data=weekend_data, x='cuisine_type', order=weekend_cuisine_counts)
plt.title('Most Popular Cuisine on Weekends')
plt.xlabel('Cuisine Type')
plt.ylabel('Count')
plt.xticks(rotation=90);

#This can also be noticed by just printing the weekend_cuisine_counts
print("\nCuisine order count on weekends:")
print(weekend_cuisine_counts)
```

The unique values of the the days are as follows
 ['Weekend' 'Weekday']

Value Counts day_of_the_week
 Weekend 1351
 Weekday 547
 Name: count, dtype: int64

Cuisine order count on weekends:
 cuisine_type
 American 415
 Japanese 335
 Italian 207
 Chinese 163
 Mexican 53
 Indian 49
 Middle Eastern 32
 Mediterranean 32
 Thai 15
 French 13
 Korean 11
 Southern 11
 Spanish 11
 Vietnamese 4
 Name: count, dtype: int64



Observations:

The most popular cuisine on weekend is American, based on the number of orders (415)...(American cuisine followed by Japanese and Italian). This indicates that there is a strong demand for American food on weekends by Foodhub customers.

Question 9: What percentage of the orders cost more than 20 dollars? [2 marks]

```
In [14]: # Write the code here
orders_gt_20 = data[data['cost_of_the_order'] > 20].shape[0]
orders_le_20 = data.shape[0] - orders_gt_20

print("\nTotal number of Orders: ", data.shape[0], "\nOrders greater than 20: ",
      orders_gt_20)

orders_gt_20_percentage = (orders_gt_20 / data.shape[0]) * 100
print("\nPercentage of orders greater than 20: ", round(orders_gt_20_percentage, 2))

#This can be represented in Pie chart as well
labels = ['Orders_less_than_equal_20', 'Orders_more_than_20']
sizes = [orders_le_20, orders_gt_20]

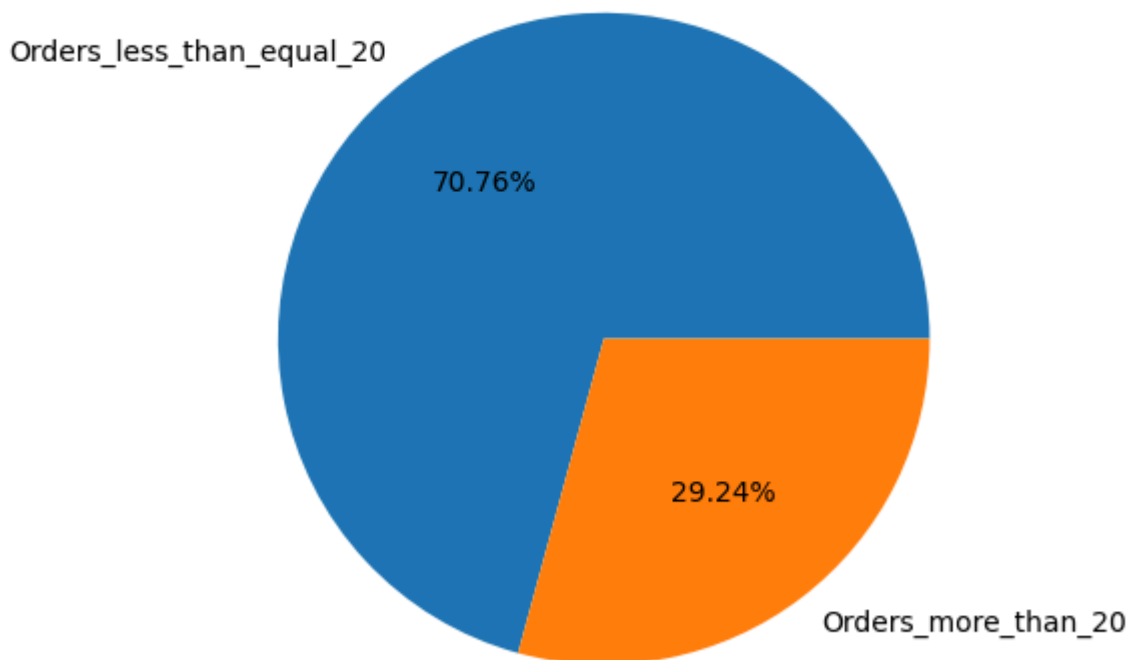
plt.pie(sizes, labels=labels, autopct='%1.2f%%')
plt.title('\nPercentage of Orders Above and Below $20')
```

```
plt.tight_layout()
plt.show()
```

Total number of Orders: 1898
Orders greater than 20: 555
Orders less than equal 20: 1343

Percentage of orders greater than 20: 29.24 %

Percentage of Orders Above and Below \$20



Observations:

The total number of orders can be derived from `data.shape[0] = 1898`.

Out of which 555 orders are greater than 20; which means 29.24% of the orders are above 20... This indicates that nearly one-third of the orders fall into orders greater than 20.

Question 10: What is the mean order delivery time? [1 mark]

```
In [15]: # Write the code here
mean_delivery_time = data['delivery_time'].mean()
print("Mean Delivery Time:", round(mean_delivery_time, 2))

#This can also be observed from mean parameter of describe
print("\nDelivery Time stats:\n", data['delivery_time'].describe())
```

Mean Delivery Time: 24.16

Delivery Time stats:

```
count    1898.000000
mean      24.161749
std        4.972637
min       15.000000
25%       20.000000
50%       25.000000
75%       28.000000
max       33.000000
```

Name: delivery_time, dtype: float64

Observations:

The mean order delivery time: 24.16

Question 11: The company has decided to give 20% discount vouchers to the top 3 most frequent customers. Find the IDs of these customers and the number of orders they placed. [1 mark]

```
In [16]: # Write the code here
#Top 3 customers can be found by using value_counts() and head...
top_3_customers = data['customer_id'].value_counts().head(3).reset_index()

top_3_customers.columns = ['customer_id', 'Number of Orders']

print("Top 3 customers eligible for 20% discount:\n")
top_3_customers
```

Top 3 customers eligible for 20% discount:

```
Out[16]:
```

	customer_id	Number of Orders
0	52832	13
1	47440	10
2	83287	9

Observations:

The top 3 customers eligible for 20% discount (along with their number of orders) are as follows:

- "Customer with customer_id 52832 with 13 orders"
- "Customer with customer_id 47440 with 10 orders"
- "Customer with customer_id 83287 with 9 orders"

Multivariate Analysis

Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. (It is a good idea to explore relations between numerical variables as well as relations between numerical and categorical variables) [10 marks]

```
In [29]: # Write the code here

# 1. We will start by plotting the heatmap to understand the Correlation between
numerical_cols = ['order_id', 'customer_id', 'cost_of_the_order', 'food_preparat
plt.figure(figsize=(8, 6))
sns.heatmap(data[numerical_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap: Numerical Variables')
plt.tight_layout()
plt.show()

#pairplot
sns.pairplot(data[numerical_cols])
plt.suptitle('Pairplot: Numerical Variables', y=1.02)
plt.tight_layout()
plt.show()

# 2. Boxplot: cost_of_the_order vs cuisine_type
# Compute median cost_of_the_order time for each cuisine
plot_order = data['cuisine_type'].value_counts().index
medians = data.groupby('cuisine_type')['cost_of_the_order'].median().reindex(plot_order)
print("\n Median cost of Various Cuisines:\n",medians)

plt.figure(figsize=(12, 6))
ax = sns.boxplot(x='cuisine_type', y='cost_of_the_order', data=data, order=plot_order)
plt.title('Cost of Order by Cuisine Type')
plt.xticks(rotation=60)

# Annotate the median value above each box
for i, cuisine in enumerate(medians.index):
    median_val = medians[cuisine]
    ax.text(i, median_val + 0.5, f'{median_val:.0f}', ha='center', va='bottom',

plt.tight_layout()
plt.show()

## Boxplot: Food Preparation Time vs Cuisine Type
# Compute median preparation time for each cuisine
plot_order = data['cuisine_type'].value_counts().index
medians = data.groupby('cuisine_type')['food_preparation_time'].median().reindex(plot_order)
print("\n Median Preparation Time for various Cuisines:\n",medians)

plt.figure(figsize=(12, 6))
ax = sns.boxplot(x='cuisine_type', y='food_preparation_time', data=data, order=plot_order)
plt.title('Food Preparation Time by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Preparation Time (minutes)')
plt.xticks(rotation=60)

# Annotate the median value above each box
for i, cuisine in enumerate(medians.index):
    median_val = medians[cuisine]
    ax.text(i, median_val + 0.5, f'{median_val:.0f}', ha='center', va='bottom',
```

```

plt.tight_layout()
plt.show();

# 3. Boxplot: delivery_time vs day_of_the_week
plt.figure(figsize=(6, 4))
sns.boxplot(x='day_of_the_week', y='delivery_time', data=data)
plt.title('\nDelivery Time by Day of the Week')
plt.tight_layout()
plt.show()

#. Now Let us evaluate ratings vs other parameters
# Replace 'Not given' with np.nan and convert to float
data['rating_cleaned'] = data['rating'].replace('Not given', np.nan).astype(float)

# 5. Rating vs Delivery time
plt.figure(figsize=(8, 4))

#intentionally included 'Not given' in the plot to understand the delivery_time
sns.boxplot(x='rating', y='delivery_time', data=data, order=['Not given', '3', '4', '5'])
plt.title('Rating vs Delivery Time')
plt.xlabel('Rating')
plt.ylabel('Delivery Time')
plt.tight_layout()
plt.show()

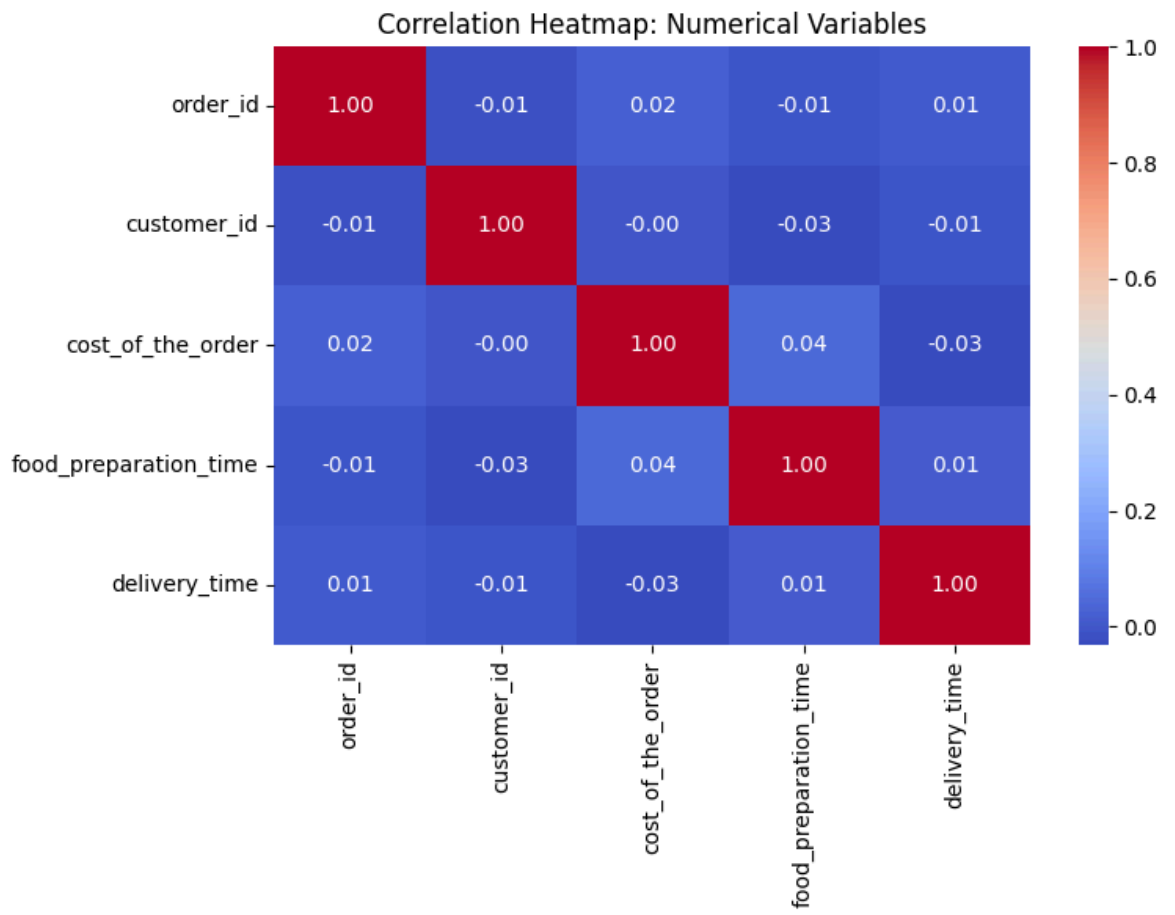
# 6. Rating vs Cost of the Order
sns.boxplot(x='rating', y='cost_of_the_order', data=data, order=['Not given', '3', '4', '5'])
plt.title('Rating vs Cost of the order')
plt.xlabel('Rating')
plt.ylabel('Cost of the Order')
plt.tight_layout()
plt.show()

# 7. Rating vs Preparation Time
sns.boxplot(x='rating', y='food_preparation_time', data=data, order=['Not given', '3', '4', '5'])
plt.title('Rating vs Food Preparation Time')
plt.xlabel('Rating')
plt.ylabel('Food Preparation Time')
plt.tight_layout()
plt.show()

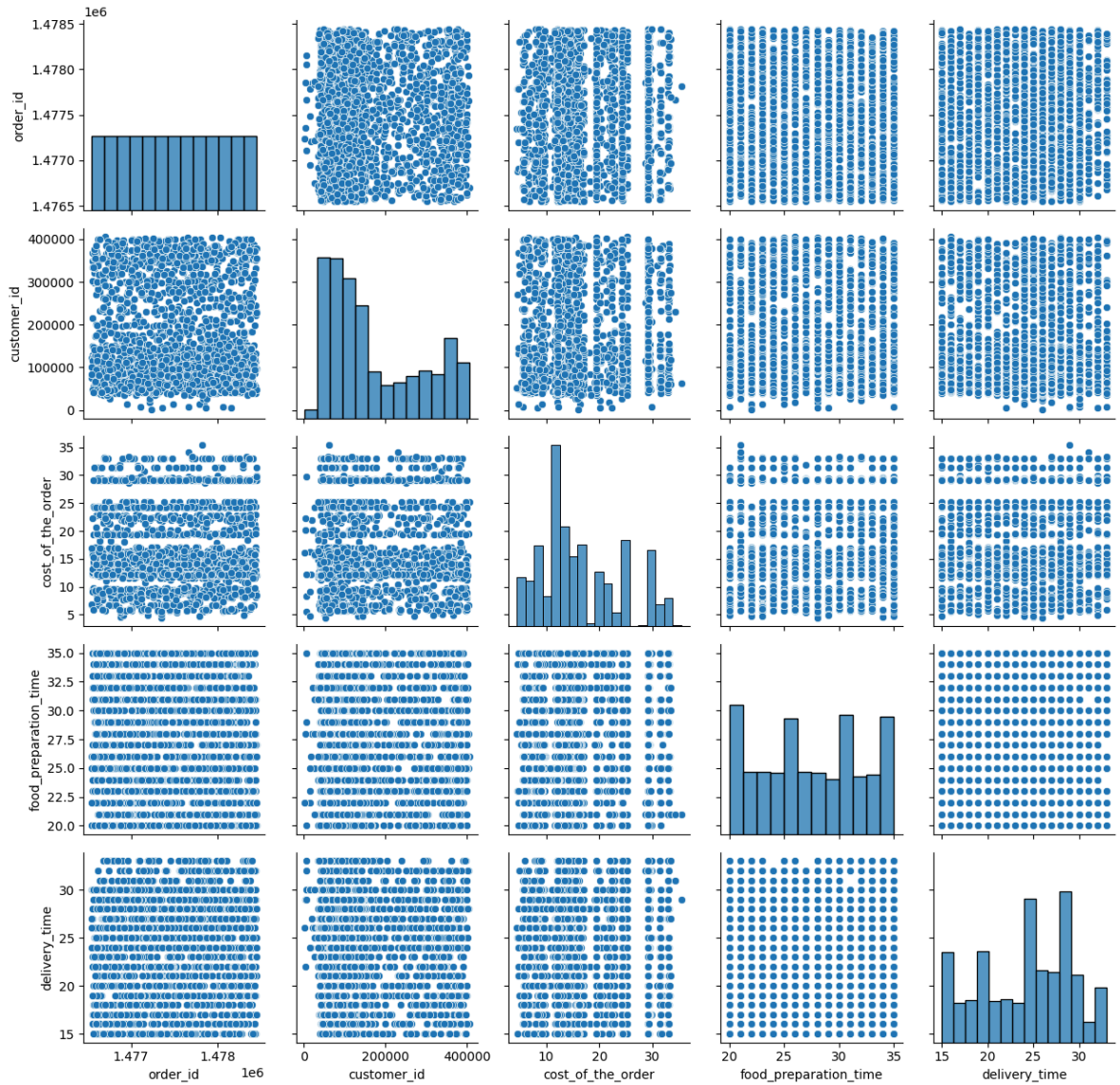
#8. Let us see from Revenue perspective...
# Group by restaurant and sum cost to find total revenue per restaurant
revenue_by_restaurant = data.groupby('restaurant_name')['cost_of_the_order'].sum()

# Plot the top 15 revenue-generating restaurants
plt.figure(figsize=(10, 5))
sns.barplot(x=revenue_by_restaurant.values, y=revenue_by_restaurant.index)
plt.title('Top 15 Revenue-Generating Restaurants')
plt.xlabel('Total Order Revenue')
plt.ylabel('Restaurant Name')
plt.tight_layout()
plt.show();

```



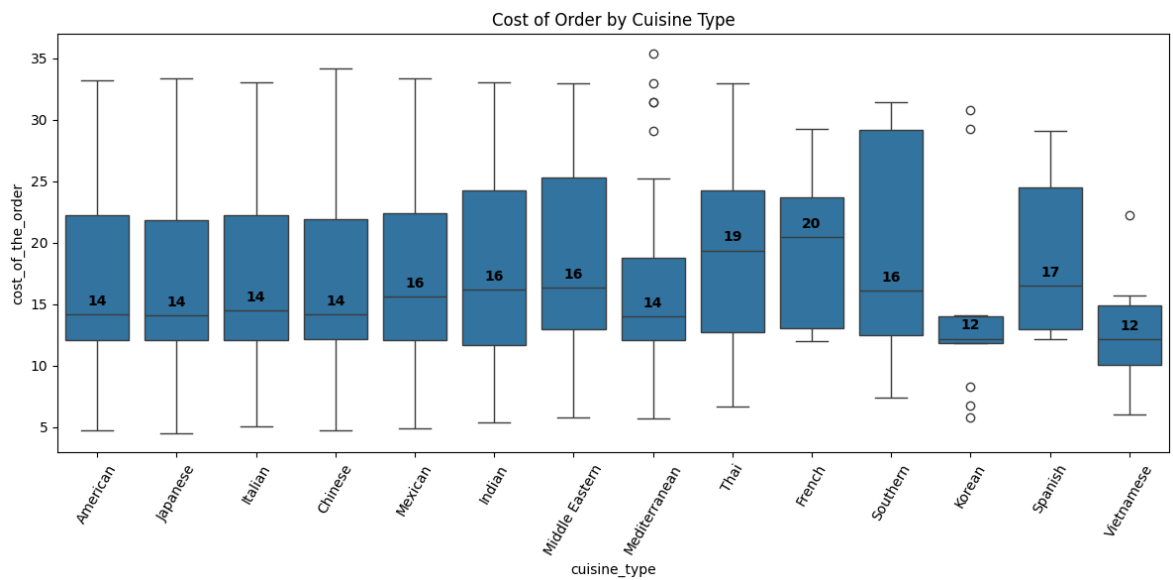
Pairplot: Numerical Variables



Median cost of Various Cuisines:
cuisine_type

American	14.120
Japanese	14.070
Italian	14.480
Chinese	14.120
Mexican	15.570
Indian	16.150
Middle Eastern	16.300
Mediterranean	13.995
Thai	19.350
French	20.470
Southern	16.110
Korean	12.180
Spanish	16.520
Vietnamese	12.130

Name: cost_of_the_order, dtype: float64

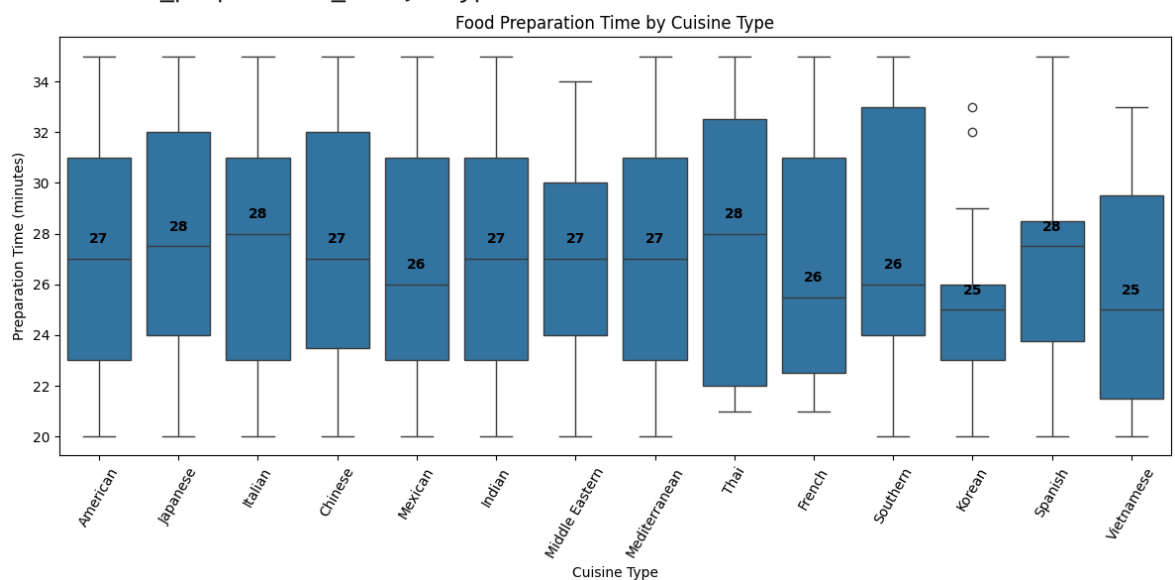


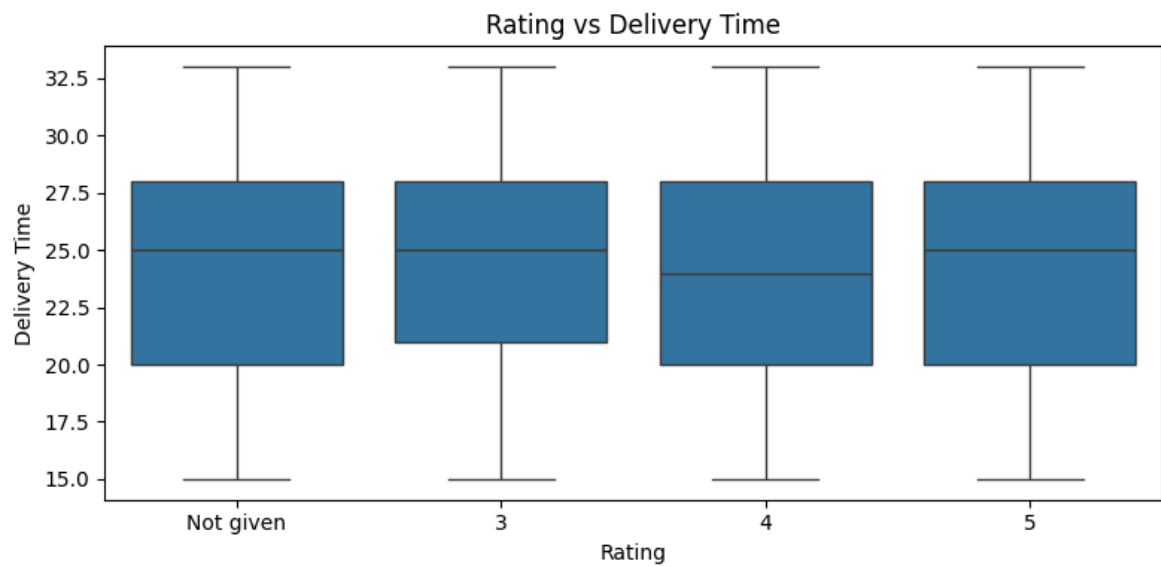
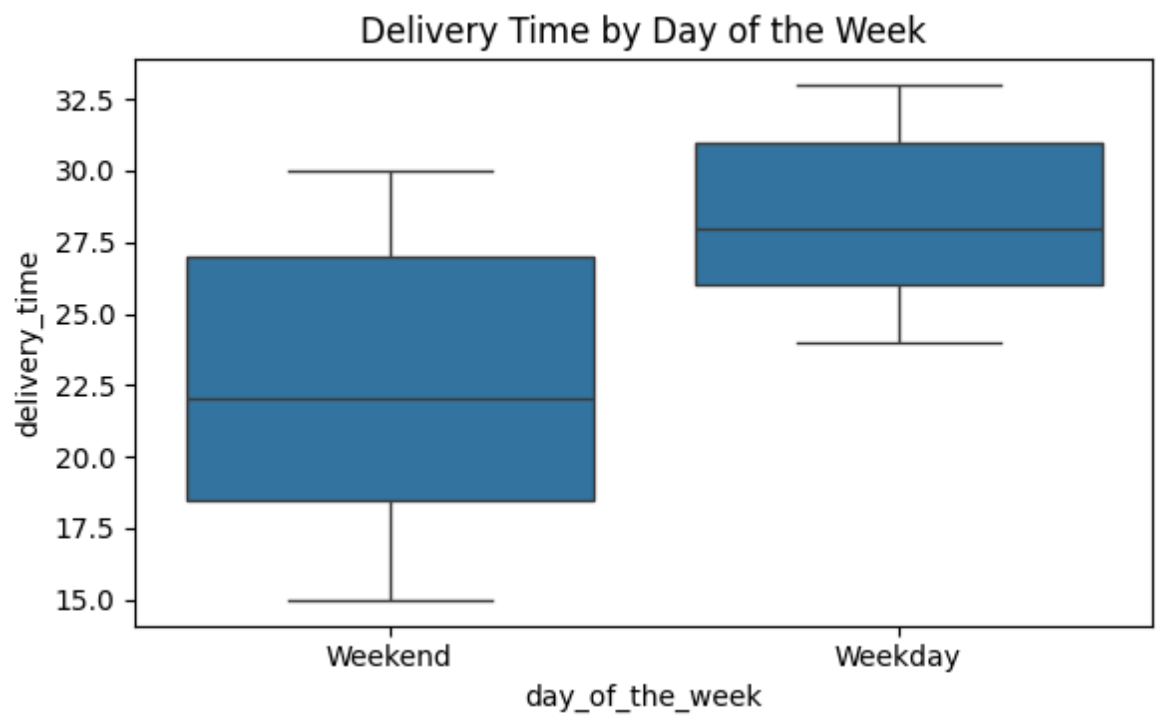
Median Preparation Time for various Cuisines:

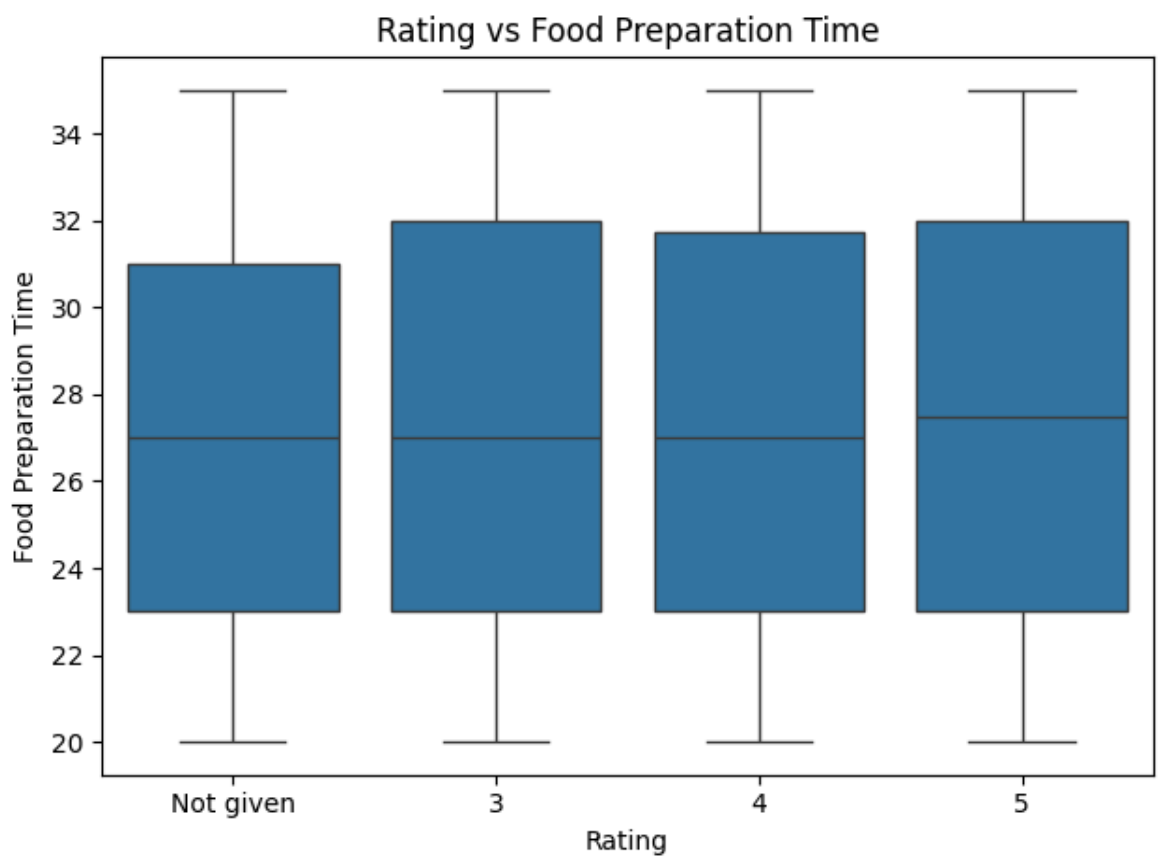
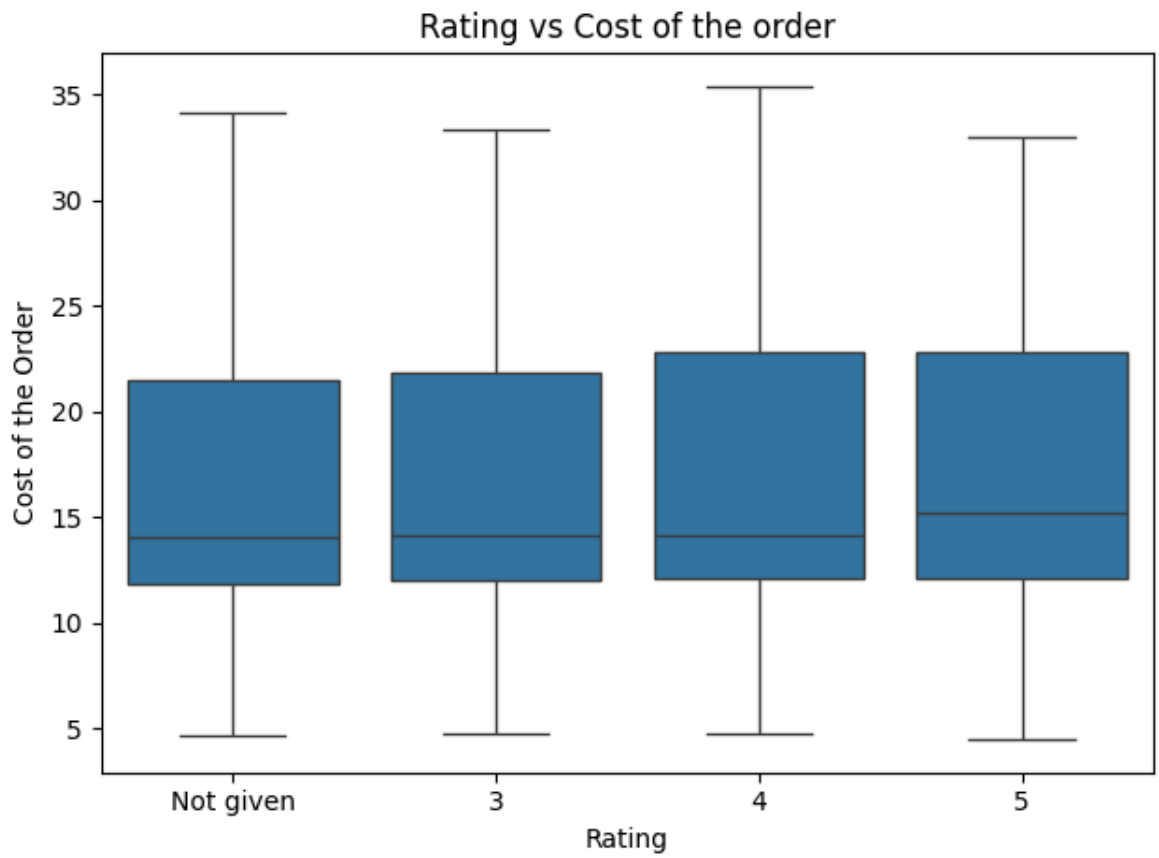
cuisine_type

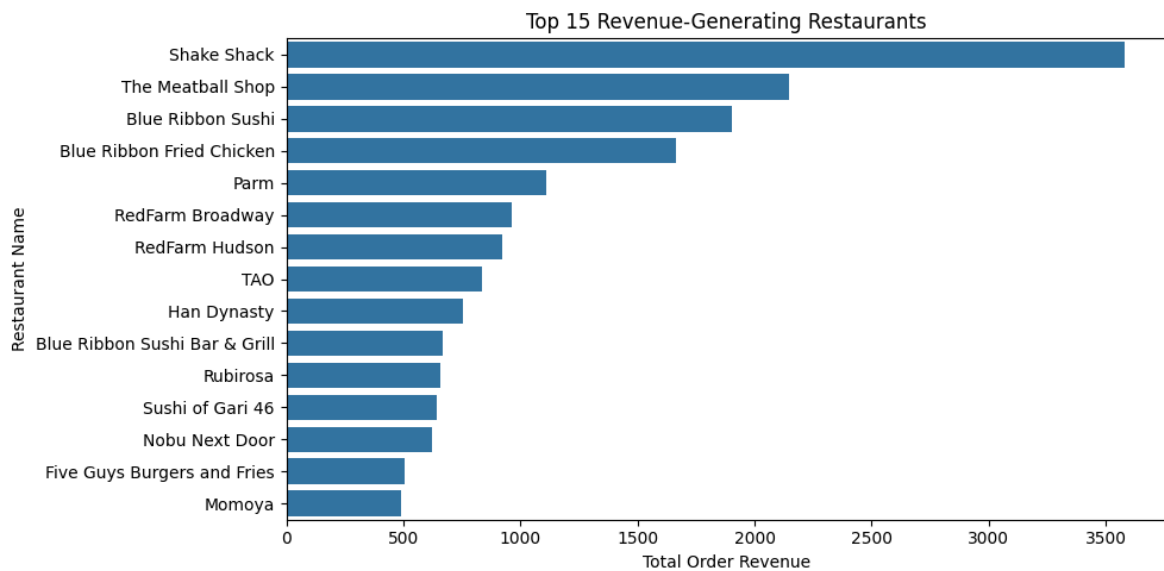
American	27.0
Japanese	27.5
Italian	28.0
Chinese	27.0
Mexican	26.0
Indian	27.0
Middle Eastern	27.0
Mediterranean	27.0
Thai	28.0
French	25.5
Southern	26.0
Korean	25.0
Spanish	27.5
Vietnamese	25.0

Name: food_preparation_time, dtype: float64









Observations:

1. Correlation Heatmap (Numerical vs Numerical) ... Very low or no correlation among the numeric params. Anyway the customer_id and order_id are used as ids... The pairplot also showed similar info

2. Deliveries are generally faster and more consistent on weekends

3. Cuisine Type Vs Cost of the order:

Note: Median values are rounded to nearest whole number in the plot for clarity; exact values used in observations.

- a. French cuisine has the highest median cost (approx 20.47) followed by Thai (19.35)

- b. Vietnamese (12.13) and Korean (12.18) are the lowest median-cost cuisines (~\$12), suggesting more budget-friendly meals

- c. Most cuisines fall in the 13–16 range...

4. Cuisine Type Vs Food Preparation Time

- a. From the box plot, it shows that Italian, Thai cuisines have highest average prep time (28)

- b. Vietnamese and Koreans have the lowest average prep time (25)

- c. Overall variation of preparation_time among different cuisine_type is small indicating that the food_prep_time is consistent across different cuisine_type

5. Ratings vs Delivery Time: For some reason, the rating (4) has lower median delivery time compared to all others. This indicates that the customers value timely delivery but not necessarily give the highest rating(5) just for speedy delivery. There may be other positive factors like food taste/quality to get 5 ratings

6. Ratings vs Cost of the order: The rating 5 has slightly higher median cost compared to other ratings but the spread (variation) across all ratings is very small

7. Ratings vs Preparation Time: Strangely the median food preparation time for rating 5 is slightly higher (by 1-2 mins) than for other ratings. Ratings 3,4 and 'Not given' all have similar (lower) median prep times. which suggests that long prep time by few mins doesnt harm the ratings
8. From the revenue perspective, it seems like "Shake Shack" stands out as the highest revenue contrinutor with more than 3500 in total order value. This is followed by 'The Meatball shop' (more thn 2000), Blue Ribbon Sushi(more than 1500) etc

Question 13: The company wants to provide a promotional offer in the advertisement of the restaurants. The condition to get the offer is that the restaurants must have a rating count of more than 50 and the average rating should be greater than 4. Find the restaurants fulfilling the criteria to get the promotional offer. [3 marks]

```
In [18]: # Write the code here
# From the earlier questions, it is understood that rating column is saved as Ob
# So first this needs to be converted to 'nan'...
data['rating_cleaned'] = data['rating'].replace('Not given', np.nan)

#since all the values belongs to ratings are numeric now, this can be converted
data['rating_cleaned'] = data['rating_cleaned'].astype(float)

#Identify the restaurants which have more than 50 ratings
restaurant_rating_stats = data.groupby('restaurant_name')['rating_cleaned'].agg(
restaurant_rating_stats.columns = ['restaurant_name', 'rating_count', 'average_r

#Filter based on the criteria (rating_count > 50 and average_rating > 4)
discount_offer_eligible_restaurants = restaurant_rating_stats[(restaurant_rating

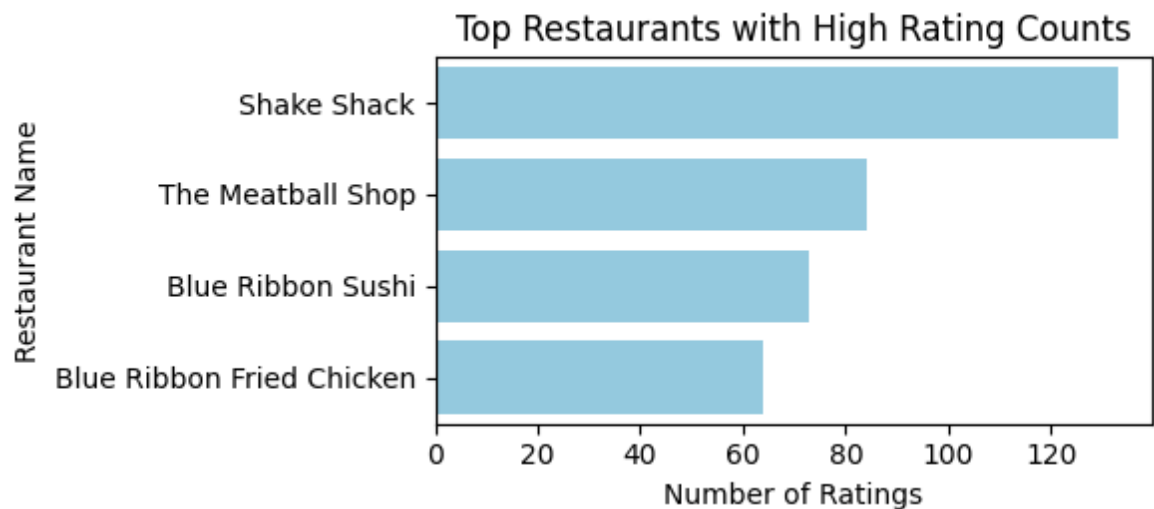
#This is to provide the list in ascending order (of the rating_count) with reset
discount_offer_eligible_restaurants_sorted = discount_offer_eligible_restaurant
discount_offer_eligible_restaurants_sorted = discount_offer_eligible_restaurant

print("\nRestaurants fulfilling the criteria to get the promotional offer are as
print(discount_offer_eligible_restaurants_sorted)

# Simple horizontal bar chart using Seaborn
plt.figure(figsize=(6, 3))
sns.barplot(
    data=discount_offer_eligible_restaurants_sorted,
    x='rating_count',
    y='restaurant_name',
    color='skyblue'
)
plt.title('\nTop Restaurants with High Rating Counts')
plt.xlabel('Number of Ratings')
plt.ylabel('Restaurant Name')
plt.tight_layout()
plt.show()
```

Restaurants fulfilling the criteria to get the promotional offer are as follows:

	restaurant_name	rating_count	average_rating
0	Shake Shack	133	4.278195
1	The Meatball Shop	84	4.511905
2	Blue Ribbon Sushi	73	4.219178
3	Blue Ribbon Fried Chicken	64	4.328125



Observations:

The 4 restaurants that qualify for the promotional offer are as follows: SHake Shack The Meatball shop Blue Ribbon Sushi Blue Ribbon Fried Chicken

'Shake Shack' being the highest number of ratings (133) while the 'Meatball Shop' being the highest average rating (4.5119).

All the four restaurants mentioned above are doing good in terms of number of ratings as well as good average rating.

Question 14: The company charges the restaurant 25% on the orders having cost greater than 20 dollars and 15% on the orders having cost greater than 5 dollars. Find the net revenue generated by the company across all orders. [3 marks]

```
In [19]: # Write the code here
#To calculate revenue generated by company, we can introduce a column order_revenue
# To calculate the revenue, the 'calculate_order_revenue()' function will be defined

#Function to calculate the revenue
def calculate_order_revenue(order_cost):
    if order_cost > 20:
        return order_cost * 0.25 # 25% discount for orders greater than 20
    elif order_cost > 5:
        return order_cost * 0.15 # 15% discount for orders greater than 5
    else:
        return 0 # No discount for orders less than or equal to 5 or 20

#The above function can be applied for each row as follows and the result will be stored in a new column
data['order_revenue'] = data['cost_of_the_order'].apply(calculate_order_revenue)
```

```
#Sum total revenue
total_revenue = data['order_revenue'].sum()

print("Total Revenue Generated by the Company:", round(total_revenue, 2))
```

Total Revenue Generated by the Company: 6166.3

Observations:

The company generates revenue by taking a commission from each order. Based on the rules provided:

Orders > 20 are charged 25%

Orders between 5 and 20 are charged 15%

The total revenue generated from all orders is \$6,166.30.

Question 15: The company wants to analyze the total time required to deliver the food. What percentage of orders take more than 60 minutes to get delivered from the time the order is placed? (The food has to be prepared and then delivered.) [2 marks]

```
In [20]: # Write the code here
# Calculate total time from order to delivery
#we will calculate the 'total delivery time' by taking food_preparation_time and

data['total_delivery_time'] = data['food_preparation_time'] + data['delivery_time']

# Find percentage of orders taking more than 60 minutes
orders_above_60 = data[data['total_delivery_time'] > 60].shape[0]
total_orders = data.shape[0]
percentage_above_60 = (orders_above_60 / total_orders) * 100
print("Number of orders taking total delivery time more than 60 mins:", orders_above_60)
print(f"Percentage of orders taking total delivery time more than 60 minutes: {percentage_above_60}%")

#This can be represented in Pie chart as follows

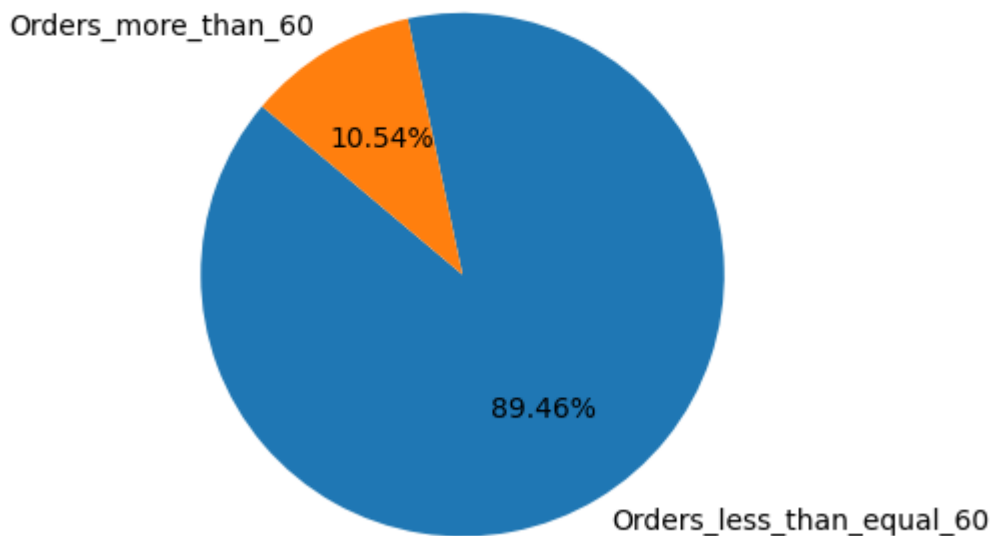
labels = ['Orders_less_than_equal_60', 'Orders_more_than_60']
sizes = [total_orders - orders_above_60, orders_above_60]

# Create pie chart
plt.figure(figsize=(6, 4))
plt.pie(sizes, labels=labels, autopct='%1.2f%%', startangle=140)
plt.title('\nDistribution of Orders by Total Delivery Time')
plt.tight_layout()
plt.show()
```

Number of orders taking total delivery time more than 60 mins: 200 and total number of orders: 1898

Percentage of orders taking total delivery time more than 60 minutes: 10.54%

Distribution of Orders by Total Delivery Time



Observations:

There are 200 orders which takes more than 60 mins of total delivery time (food_preparation_time + delivery_time) out of total number of orders 1898.

This is about 10.54% of orders exceeding 60 mins in total delivery time. This long of total delivery time may impact customer satisfaction.

Question 16: The company wants to analyze the delivery time of the orders on weekdays and weekends. How does the mean delivery time vary during weekdays and weekends? [2 marks]

```
In [21]: # Write the code here
# All weekdays are represented using weekday...
print("The unique values of the the days are as follows\n", data['day_of_the_week'].unique())

#Calculate mean delivery time for weekday and weekend...
mean_delivery_time_weekday = data[data['day_of_the_week'] != 'Weekend']['delivery_time'].mean()
mean_delivery_time_weekend = data[data['day_of_the_week'] == 'Weekend']['delivery_time'].mean()

print("\nMean Delivery Time on Weekdays:", round(mean_delivery_time_weekday, 2))
print("Mean Delivery Time on Weekends:", round(mean_delivery_time_weekend, 2))

#To think beyond the "delivery_time" and take a point from the earlier question
#Let us see how this compares for weekday vs weekend..
mean_total_delivery_time_weekday = data[data['day_of_the_week'] != 'Weekend']['total_delivery_time'].mean()
mean_total_delivery_time_weekend = data[data['day_of_the_week'] == 'Weekend']['total_delivery_time'].mean()

print("\nMean Total Delivery Time on Weekdays:", round(mean_total_delivery_time_weekday, 2))
print("Mean Total Delivery Time on Weekends:", round(mean_total_delivery_time_weekend, 2))
```


The unique values of the the days are as follows
['Weekend' 'Weekday']

Mean Delivery Time on Weekdays: 28.34

Mean Delivery Time on Weekends: 22.47

Mean Total Delivery Time on Weekdays: 55.55

Mean Total Delivery Time on Weekends: 49.91

Observations:

The mean delivery time on weekdays is higher than on weekends. Even if we do the analysis on total delivery time (food_preparation_time + delivery time), the similar gap exists.

This may be due to, more delivery personnael + food preparation staff are available on weekends compared to weekdays.

In addition the factor that may be helping is the traffic conditions on weekend may be lower compared to weekedays.

Conclusion and Recommendations

Question 17: What are your conclusions from the analysis? What recommendations would you like to share to help improve the business? (You can use cuisine type and feedback ratings to drive your business recommendations.) [6 marks]

Conclusions:

The FoodHub has 1200 unique customers. with 698 customers ordering more than once (repeated customers)

There are 178 unique restaurants that are partnered with FoodHub and 'ShakeShack' stands out interms of numbers of orders. Small group of restaurants are contributing in generating FoodHub revenue.

Around 14 unique types of cuisines are offered from Foodhub platform. American, Japanese and Italian are most frequently ordered cuisnes.. The least frequent being Vietnamese, Spanish.

The majority of orders are in the ranged of 10-20... This seems like economically within the budget ranges for middle class members.

The customers are ordering more during weekends compared to weekdays.

Many customers are showing their satisfaction by providing 5 rating. This proves that the customer satisfaction is high but a large portion of users are skipping and not providing the rating.

The median food preparation time is around 27 mins irrespective of the cuisine type. So the restaurants that are offered through FoodHub have consistency in their food preparation process.

The median delivery is around 24-25 minutes.. The food preparation time generally consistent irrespective of the cuisine_type, which suggests that variation in delivery time is more likely due to external factors (traffic, logistics) than restaurant performance.

The food preparation time or delivery time doesn't strongly influence the ratings. It was observed that even the orders with 5 ratings have slightly higher food preparation time...

Recommendations:

- We can start from customer rating perspective. As many orders are not rated, the customers can be encouraged to provide the feedback by providing some small rewards for providing the feedback...
 - Increase the marketing efforts on already highly demanded cuisine types like American, Japanese, Italian etc. The Foodhub can mix and match offers with low selling cuisines with these high demand ones so that more business can be extracted...
 - Offer discounts through high performing restaurants like 'Shake Shack' and start displaying those apps in FoodHub app...
 - In the analysis, it was identified that the food preparation time is consistent for different cuisine types, need to concentrate on how the food delivery time can be reduced. may be optimizing the routes/or improving algorithms for identifying delivery members closer to customer/restaurants
-