Project Python Foundations: FoodHub Data Analysis

Marks: 60

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: 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 [ ]:

```
# 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 [ ]:

```
# 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 [ ]:

```
# read the data
df = pd.read_csv('/content/drive/MyDrive/Data Analytics Course/Week 9- PROJECT 2/fo
# returns the first 5 rows
df.head()
```

Out[ ]:

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

Observations:

The DataFrame has 9 columns as mentioned in the Data Dictionary. Data in each row corresponds to the order placed by a customer.

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

In [ ]:

```
# Find number of rows and columns:
df.shape
```

Out[ ]:

```
(1898, 9)
```

Observations:

There are 1898 rows and 9 columns present in the data.

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

In [ ]:

```
# Use info() to print a concise summary of the DataFrame
df.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:

Order_id, Customer_id, Food_preparation_time, and Delivery_time are all integer datatype. Restaurant_name, Cuisine_type, Day_of_the_week, and Rating are all object datatype. Cost_of_the_order is a float datatype.

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

In [ ]:

```
# Find missing values
df.isnull().sum()
```

Out[ ]:

	0
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:

There are no missing values in the data, therefore no treatment is needed. Additionally, there are no duplicates in the dataset.

In [ ]:

```
# Find duplicates
df.duplicated().sum()
```

Out[ ]:

```
0
```

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 [ ]:

```
# Find statistical summary of data
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	
order_id	1898.0	1.477496e+06	548.049724	1476547.00	1477021.25	14774
customer_id	1898.0	1.711685e+05	113698.139743	1311.00	77787.75	1286
cost_of_the_order	1898.0	1.649885e+01	7.483812	4.47	12.08	
food_preparation_time	1898.0	2.737197e+01	4.632481	20.00	23.00	.
delivery_time	1898.0	2.416175e+01	4.972637	15.00	20.00	.

Observations:

Once an order has been placed, it takes a minimum of 20 minutes, average of 27 minutes, and a maximum of 35 minutes for food to be prepared.

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

In [ ]:

```
# Find the unique ratings in dataset
unique_ratings = df['rating'].unique()
print(unique_ratings)
```

```
['Not given' '5' '3' '4']
```

In [ ]:

```
# Count how many are "not given"
df['rating'].isin(['Not given']).sum()
```

Out[ ]:

```
736
```

Observations:

736 orders are not rated.

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 [ ]:  # List the columns for univariate analysis
         df.columns
```

```
Out[ ]:  Index(['order_id', 'customer_id', 'restaurant_name', 'cuisine_type',
                'cost_of_the_order', 'day_of_the_week', 'rating',
                'food_preparation_time', 'delivery_time'],
               dtype='object')
```

```
In [ ]:  # Count of restaurant names ranked by number of orders
         restaurant_counts = df['restaurant_name'].value_counts()
         print(restaurant_counts)

         # Spacing for output display
         print( )
         print("-"*50)
         print( )

         #Statistical summary of each restaurant
         print(df['restaurant_name'].describe())
```

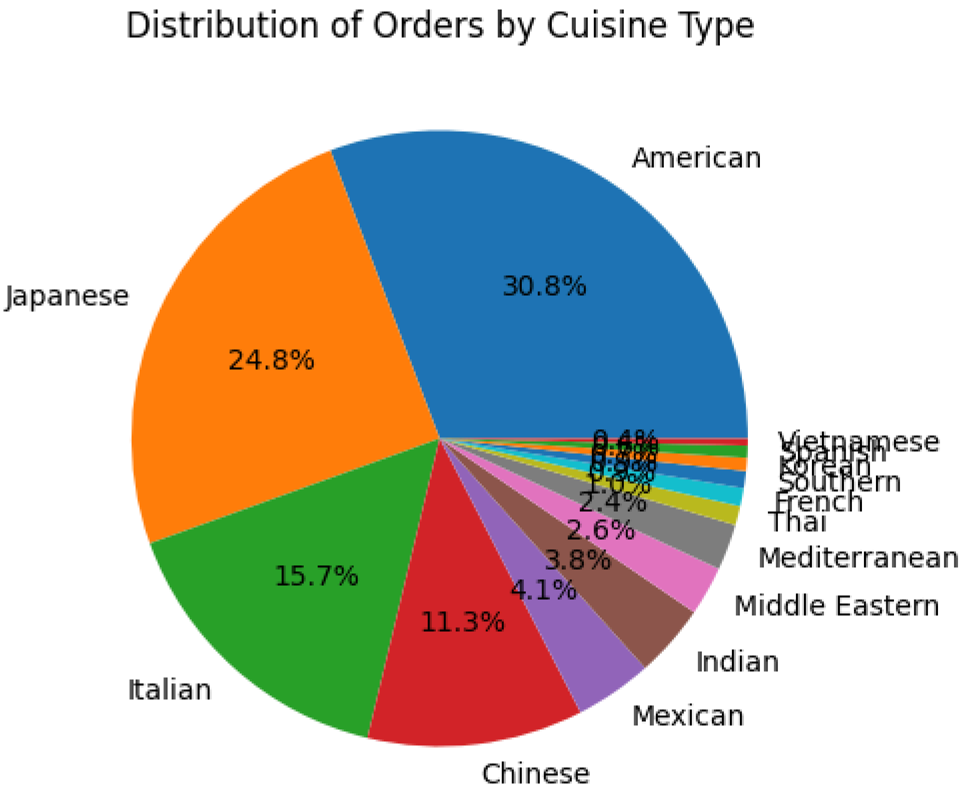
```
restaurant_name
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
                            ...
Sushi Choshi                   1
Dos Caminos Soho               1
La Follia                      1
Philippe Chow                  1
'wichcraft                     1
Name: count, Length: 178, dtype: int64

--------------------------------------------------

count            1898
unique            178
top       Shake Shack
freq              219
Name: restaurant_name, dtype: object
```

Observation: The top 3 restaurants account for nearly 1/4 of orders on FoodHub (470 out of 1898 indivudual orders).

In [ ]:

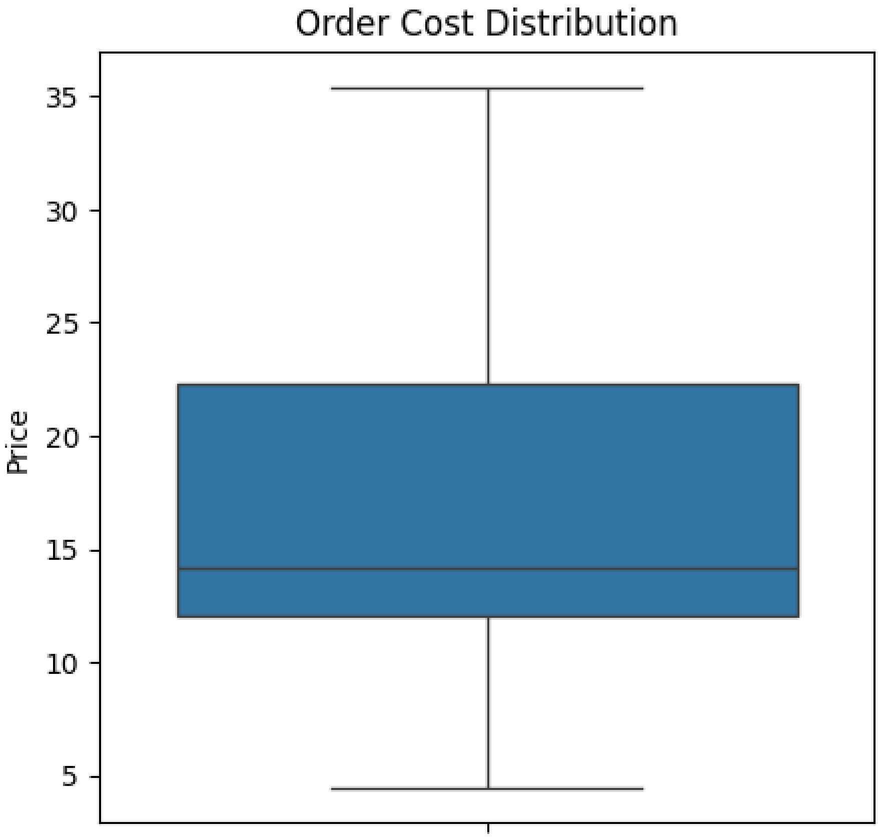
```
# Count of cuisine type, ranked by order counts
cuisine_type_counts = df['cuisine_type'].value_counts()
print(cuisine_type_counts)

#Spacing in output display
print( )
print("-"*50)
print( )

# Statistical summary of each cuisine type
print(df['cuisine_type'].describe())

# Create pie chart to see the distribution of orders among cuisine type
plt.figure(figsize=(5, 5))
plt.pie(cuisine_type_counts, labels=cuisine_type_counts.index, autopct='%1.1f%%')
plt.title('Distribution of Orders by Cuisine Type')
plt.show()
```

```
cuisine_type
American          584
Japanese          470
Italian           298
Chinese           215
Mexican            77
Indian             73
Middle Eastern     49
Mediterranean      46
Thai               19
French             18
Southern           17
Korean             13
Spanish            12
Vietnamese          7
Name: count, dtype: int64

--------------------------------------------------

count         1898
unique          14
top       American
freq           584
Name: cuisine_type, dtype: object
```



Observation: American food is the top cuisine type from customers on FoodHub followed by Japanese, Italian, and Chinese food, together making up 82.56% of orders.

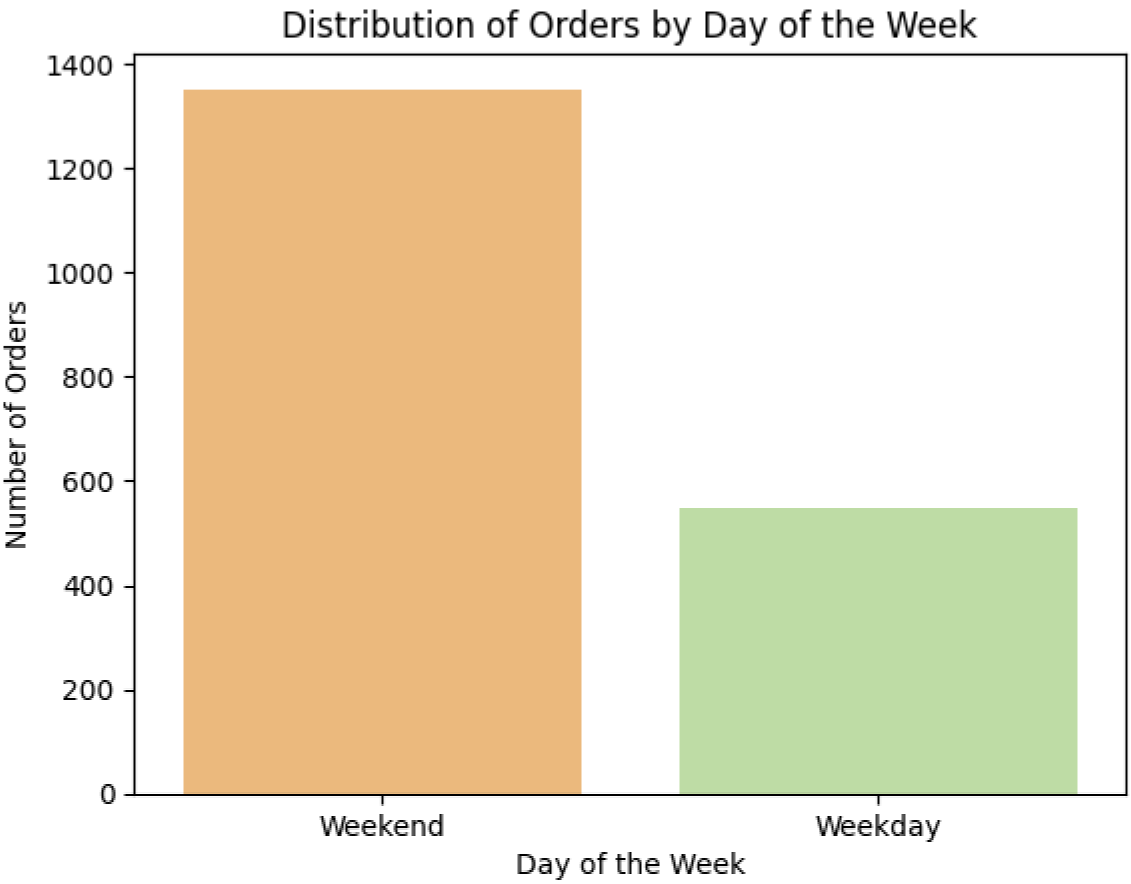
```
In [ ]: # Box plot for cost of the order
        plt.figure(figsize=(5, 5))
        sns.boxplot(y = df['cost_of_the_order'])
        plt.title('Order Cost Distribution')
        plt.ylabel('Price')
        plt.show()

        # Statistical analysis of 'cost_of_the_order'
        print(df['cost_of_the_order'].describe())
```



```
count    1898.000000
mean       16.498851
std         7.483812
min         4.470000
25%        12.080000
50%        14.140000
75%        22.297500
max        35.410000
Name: cost_of_the_order, dtype: float64
```

Observation: Order costs average 16.50 with a minimum order of 4.47 and maximum order of 35.41. Half of the orders range between 12.08 and 22.30. (Numbers are in USD)

In [ ]:
```
# Histogram of day_of_the_week distribution among orders
sns.countplot(x='day_of_the_week', data=df, palette='Spectral')
plt.title('Distribution of Orders by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.show()

#Statistical summary of day_of_the_week
print(df['day_of_the_week'].describe())

print()
print('-'*50)
print()

# Find the percentage of orders based on day of the week
print((df['day_of_the_week'].value_counts(normalize=True))*100)
```

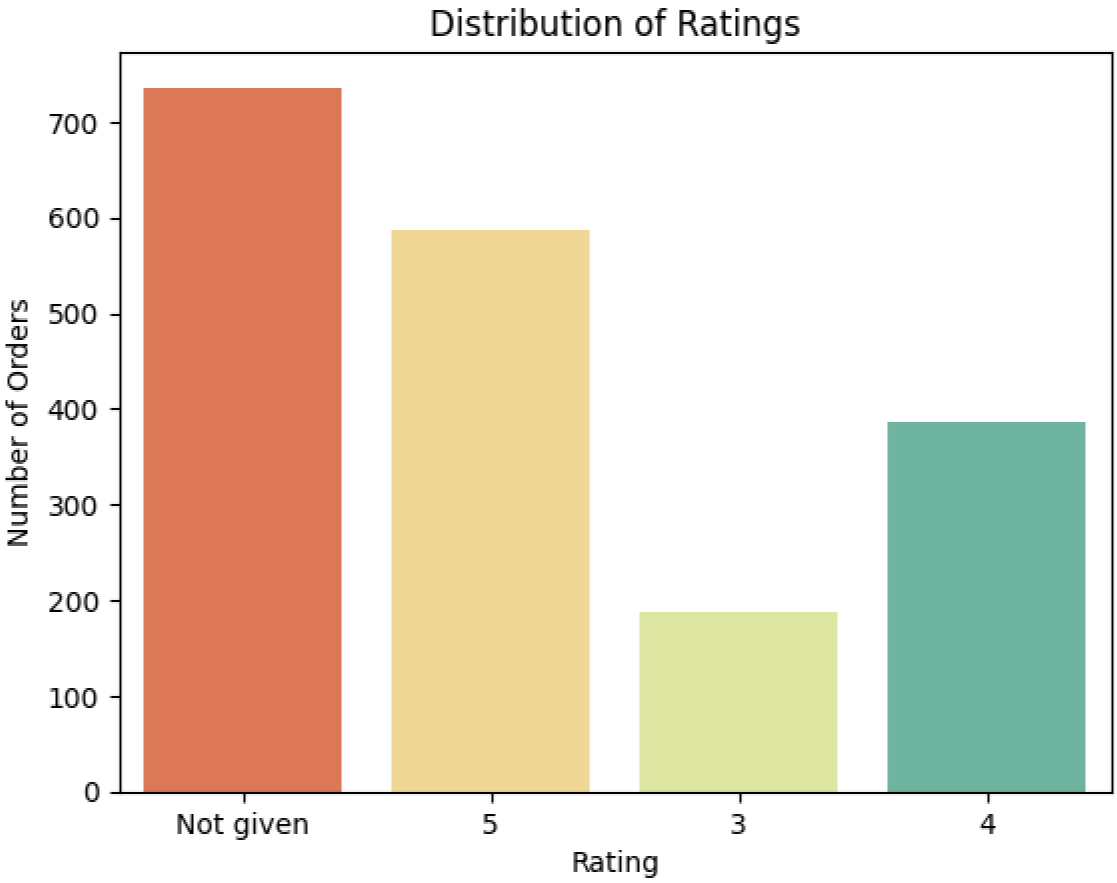


```
count        1898
unique          2
top       Weekend
freq         1351
Name: day_of_the_week, dtype: object

--------------------------------------------------

day_of_the_week
Weekend    71.18019
Weekday    28.81981
Name: proportion, dtype: float64
```

Observation: 71% of orders are placed on the weekends.

In [ ]:

```
# Check the distribution of ratings
sns.countplot(x='rating', data=df, palette='Spectral')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Number of Orders')
plt.show()

# Count for each rating
rating_counts = df['rating'].value_counts()
print(rating_counts)

#Spacing in output
print( )
print("-"*50)
print( )

#Statistical summary of rating
print(df['rating'].describe())
```

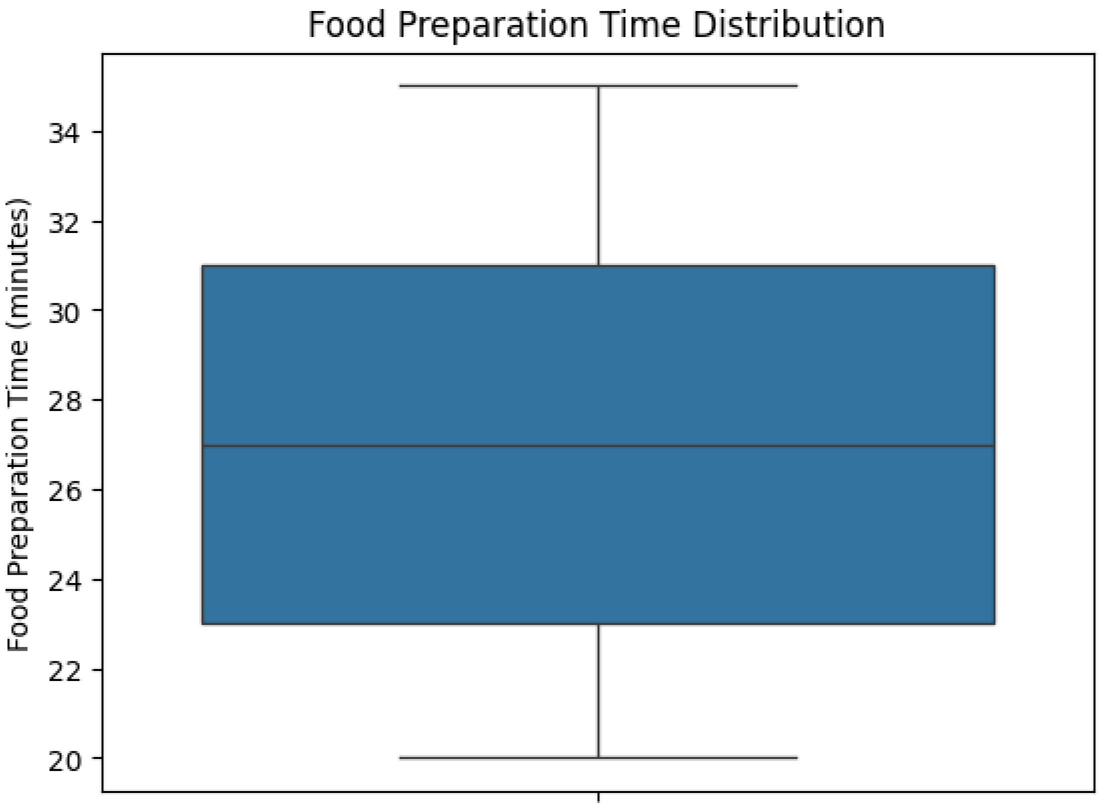


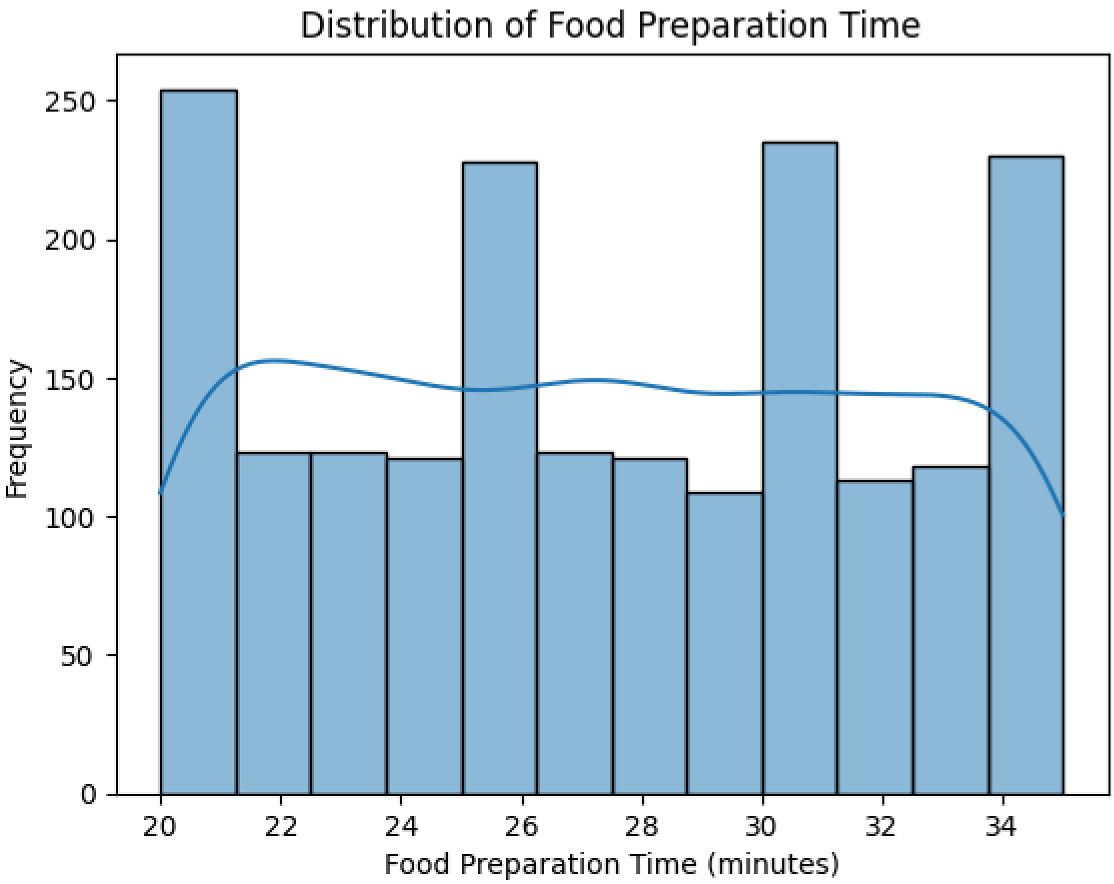
```
rating
Not given    736
5            588
4            386
3            188
Name: count, dtype: int64

--------------------------------------------------

count          1898
unique            4
top       Not given
freq            736
Name: rating, dtype: object
```

Observation: More than 1/3 of orders are not rated. Of orders that are rated, 50.60% of reviews are 5 stars, 33.29% are 4 stars, and 16.18% are 3 stars.

In [ ]:
```
# Box plot for food preparation time
sns.boxplot(y=df['food_preparation_time'])
plt.title('Food Preparation Time Distribution')
plt.ylabel('Food Preparation Time (minutes)')
plt.show()
```

```
# Histogram for food preparation time
sns.histplot(df['food_preparation_time'], kde=True)
plt.title('Distribution of Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Frequency')
plt.show()

# Statistical analysis of 'food_preparation_time'
print(df['food_preparation_time'].describe())
```



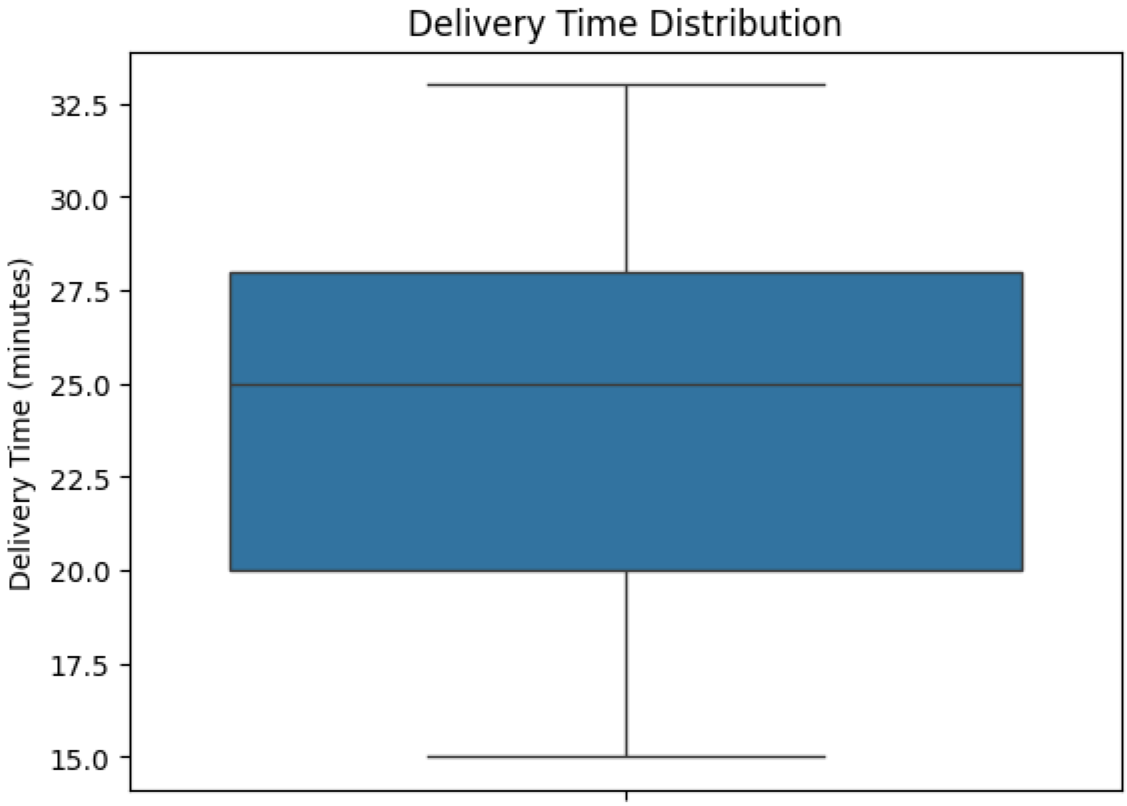


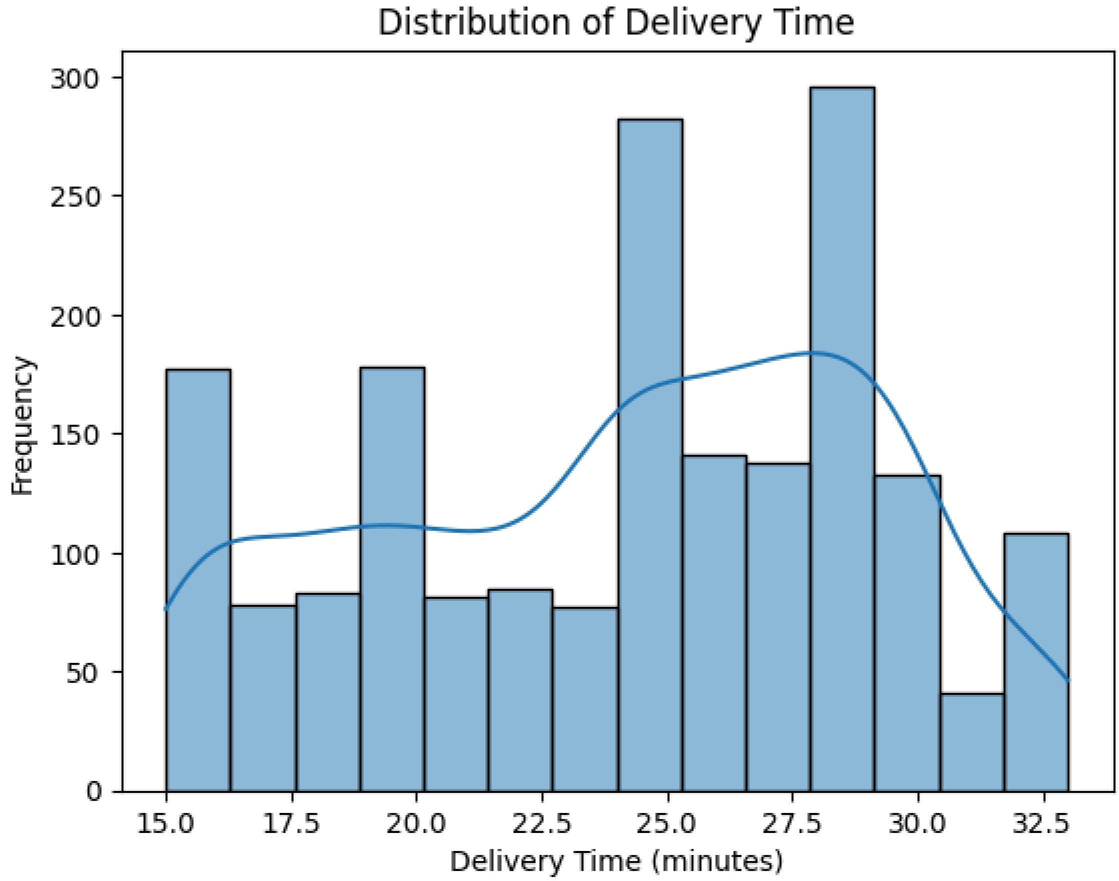
```
count    1898.000000
mean       27.371970
std         4.632481
min        20.000000
25%        23.000000
50%        27.000000
75%        31.000000
max        35.000000
Name: food_preparation_time, dtype: float64
```

Observations: The food preparation time is relatively evenly distributed amoungst the orders ranging from 20-35 minutes with 50% of the orders being prepared between 23-31 minutes and averaging 27.37 minutes.

In [ ]:
```
# Box plot for delivery time
sns.boxplot(y=df['delivery_time'])
plt.title('Delivery Time Distribution')
plt.ylabel('Delivery Time (minutes)')
plt.show()

# Histogram for delivery time
sns.histplot(df['delivery_time'], kde=True)
plt.title('Distribution of Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Frequency')
plt.show()
```

```python
# Statistical analysis of delivery time
print(df['delivery_time'].describe())
```


Delivery Time Distribution


Distribution of Delivery Time

```
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
```

Observation: Delivery time also does not have outliers. It averages 24.16 minutes to deliver food with 50% of deliveries happening within 20-28 minutes. The quickest deliveries happen in 15 minutes and the slowest take 33 minutes.

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

In [ ]:

```
# Count the number of orders per restaurant, then show only the top 5 (the head)
top_5_restaurants = df['restaurant_name'].value_counts().head()
top_5_restaurants
```

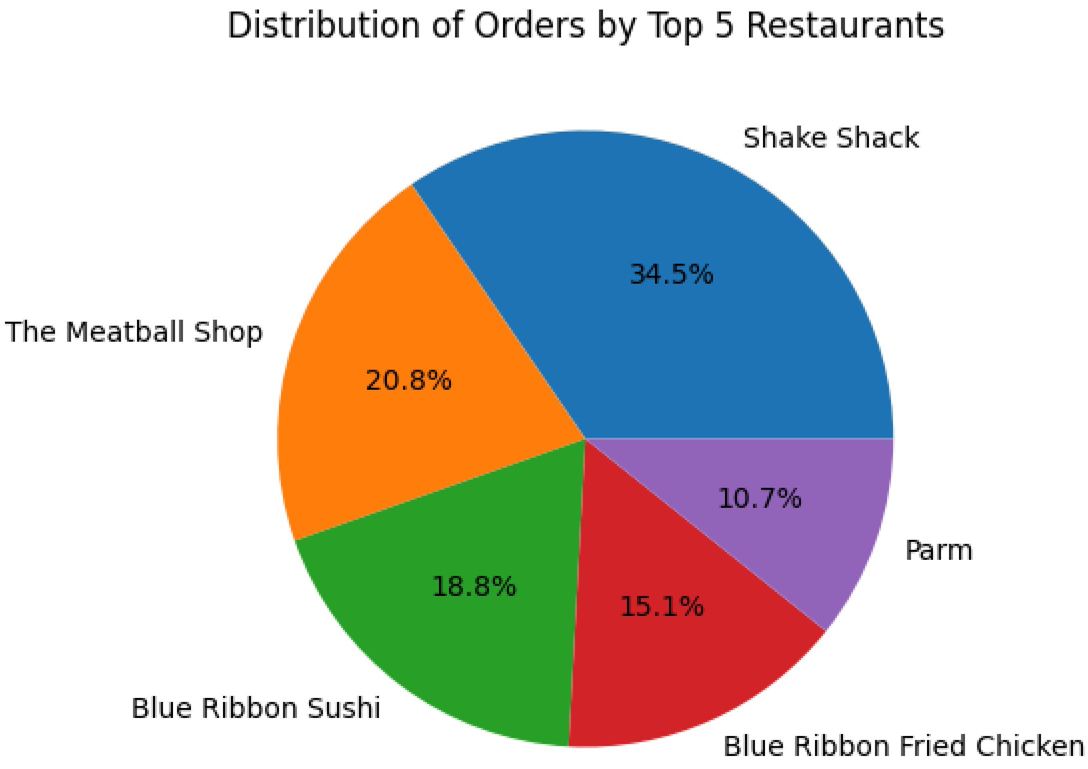
Out[ ]:

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

dtype: int64

In [ ]:

```
# Create a pie chart to show the distribution of orders amongst the top 5 restauran
plt.figure(figsize=(5, 5))
plt.pie(top_5_restaurants, labels=top_5_restaurants.index, autopct='%1.1f%%')
plt.title('Distribution of Orders by Top 5 Restaurants')
plt.show()
```



Observations:

The top 5 restaurants on FoodHub are Shake Shack, The Meatball Shop, Blue Ribbon Sushi, BLue Ribbon Fried Chicken, and Parm.

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

In [ ]:

```
# Find number of orders based on orders that fall under Weekend for day_of_the_week
popular_cuisine_weekend = df[df['day_of_the_week'] == 'Weekend']['cuisine_type'].va
popular_cuisine_weekend
```

Out[ ]:

	count
cuisine_type	
American	415

dtype: int64

Observations:

American food is the most popular cuisine on weekends.

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

```
In [ ]: # Find orders where cost_of_the_order is more than 20
        orders_above_20 = df[df['cost_of_the_order'] > 20]

        # Turn that number into a percentage
        percentage_above_20 = (len(orders_above_20) / len(df)) * 100
        print(f"{percentage_above_20:.2f}% of orders cost more than $20.")
```

```
29.24% of orders cost more than $20.
```

Observations:

29.24% of orders cost more than 20USD. The majority of orders on FoodHub are less than 20.

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

```
In [ ]: # Calculate mean delivery time
        mean_delivery_time = df['delivery_time'].mean()
        print(f"The mean order delivery time is {mean_delivery_time:.2f} minutes.")
```

```
The mean order delivery time is 24.16 minutes.
```

Observations:

The mean order delivery time is 24.16 minutes.

Question 11: The company has decided to give 20% discount vouchers to the top 5 most frequent customers. Find the IDs of these customers and the number of orders they placed. [1 mark]

```
In [ ]: # Count the number of orders per customer and display only the top 5
        top_5_customers = df['customer_id'].value_counts().head()
        print ('The customer ids for the top 5 customers are', top_5_customers.index.tolist
```

```
The customer ids for the top 5 customers are [52832, 47440, 83287, 250494, 259341]
```

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]

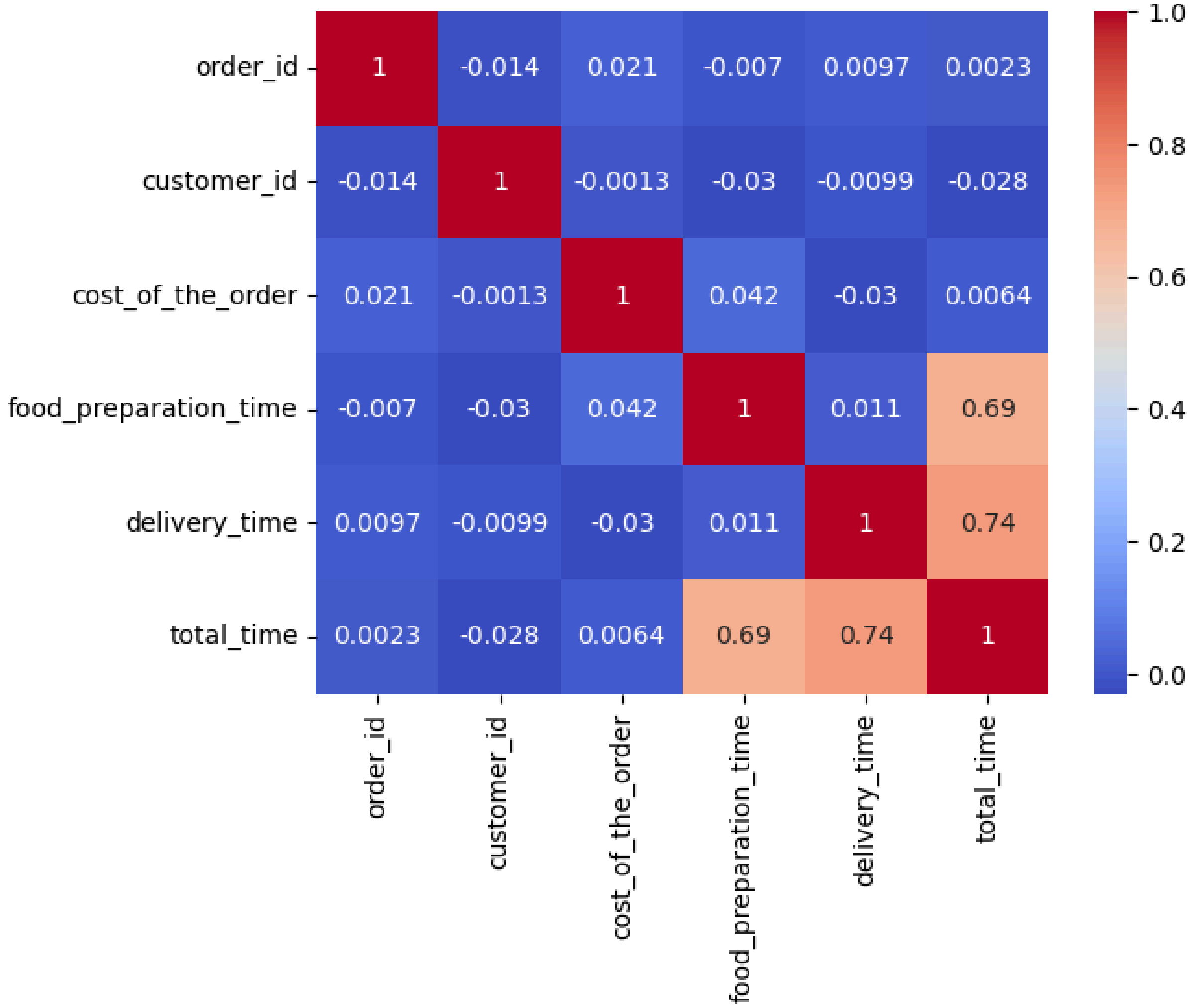
In [ ]:

```
# Check the column names and data types
df.info()
```

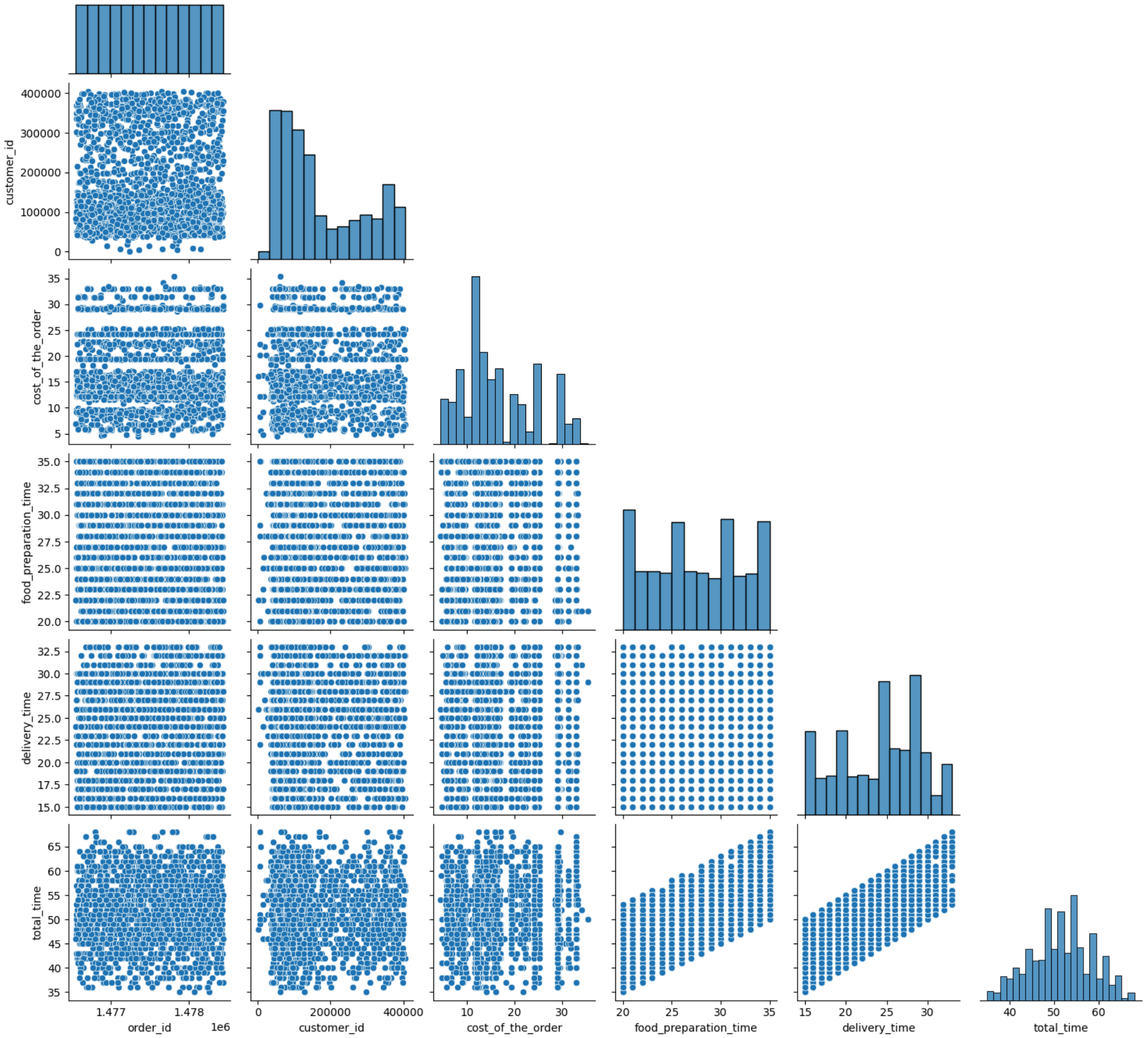
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   order_id               1898 non-null   int64  
 1   customer_id            1898 non-null   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  
 9   total_time             1898 non-null   int64  
dtypes: float64(1), int64(5), object(4)
memory usage: 148.4+ KB
```

In [ ]:

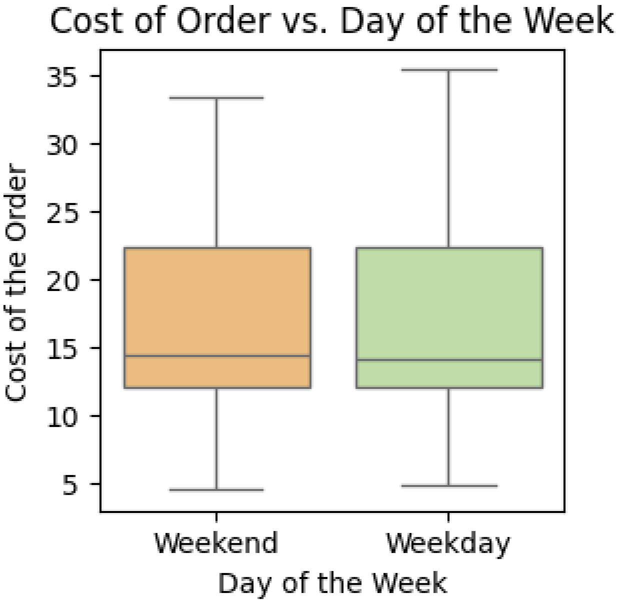
```
# Create an sns heatmap to compare each value against others
numerical_df = df.select_dtypes(include=np.number)  # Select only numerical columns
sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm');
```

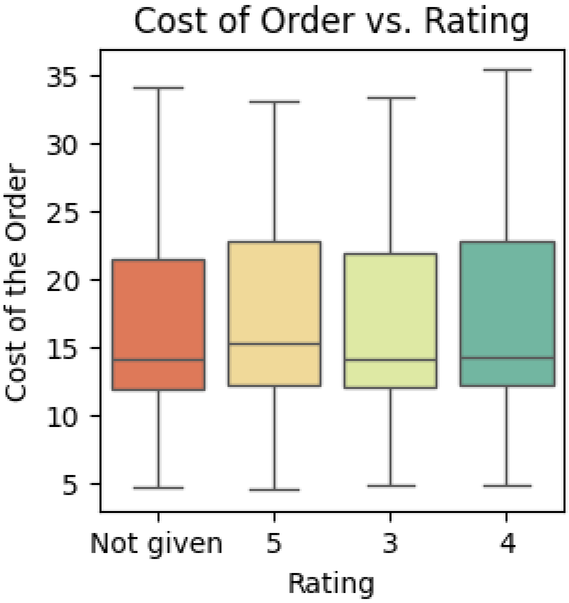


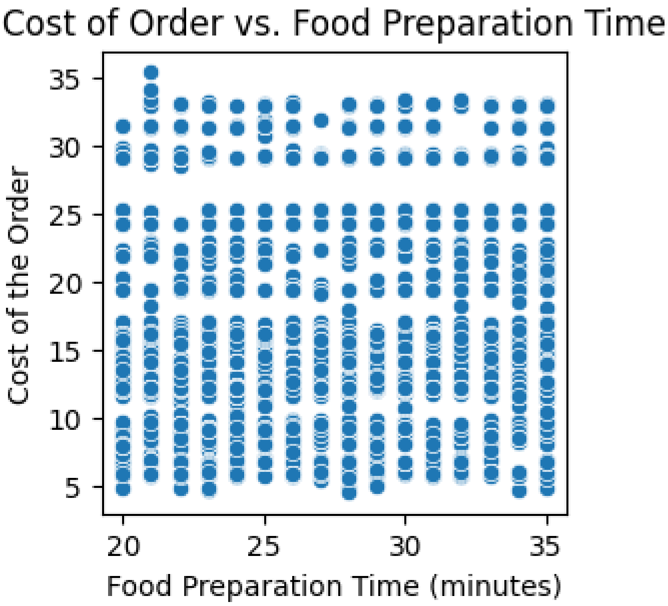
In [ ]:
```
# Create a pairplot to compare each value against the others
sns.pairplot(data=df, corner=True);
```

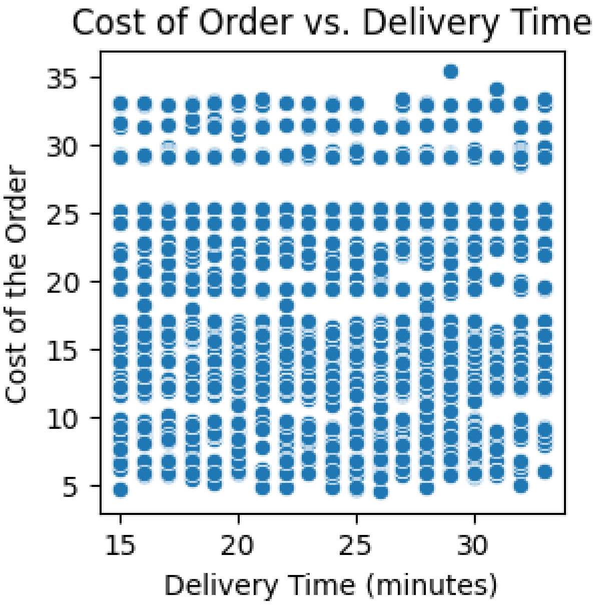


In [ ]:
```
# Compare relations between cost_of_the_order to other variables

# cost_of_the_order vs. day_of_the_week
plt.figure(figsize=(3,3))
sns.boxplot(x='day_of_the_week', y='cost_of_the_order', data=df, palette='Spectral'
plt.title('Cost of Order vs. Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Cost of the Order')
plt.show()

# cost_of_the_order vs. rating
plt.figure(figsize=(3,3))
sns.boxplot(x='rating', y='cost_of_the_order', data=df, palette='Spectral')
plt.title('Cost of Order vs. Rating')
plt.xlabel('Rating')
plt.ylabel('Cost of the Order')
plt.show()

# cost_of_the_order vs. food_preparation_time
plt.figure(figsize=(3,3))
sns.scatterplot(x='food_preparation_time', y='cost_of_the_order', data=df)
plt.title('Cost of Order vs. Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Cost of the Order')
plt.show()

# cost_of_the_order vs. delivery_time
plt.figure(figsize=(3,3))
sns.scatterplot(x='delivery_time', y='cost_of_the_order', data=df)
plt.title('Cost of Order vs. Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Cost of the Order')
plt.show()
```





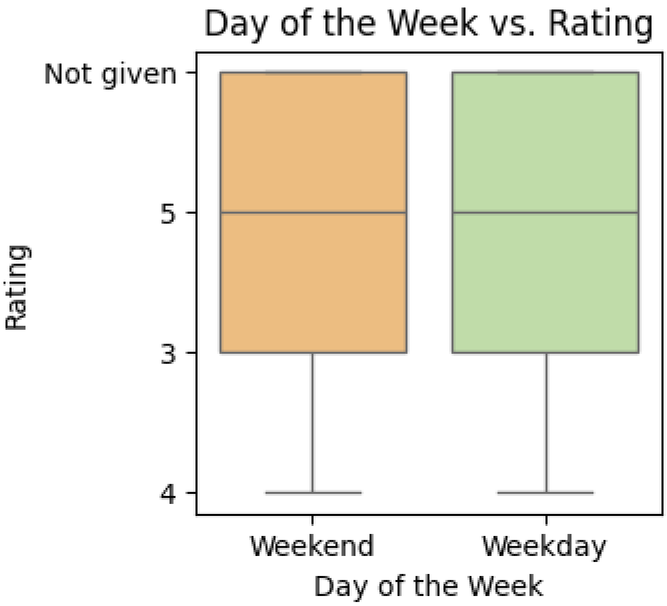


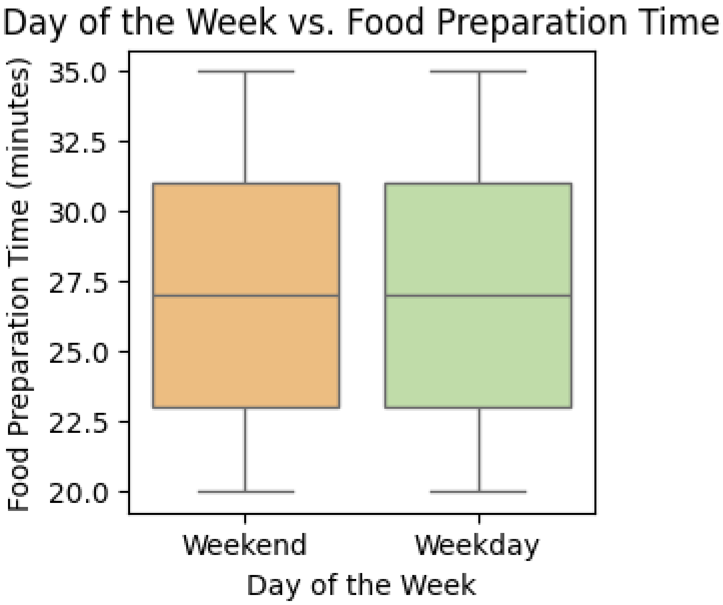


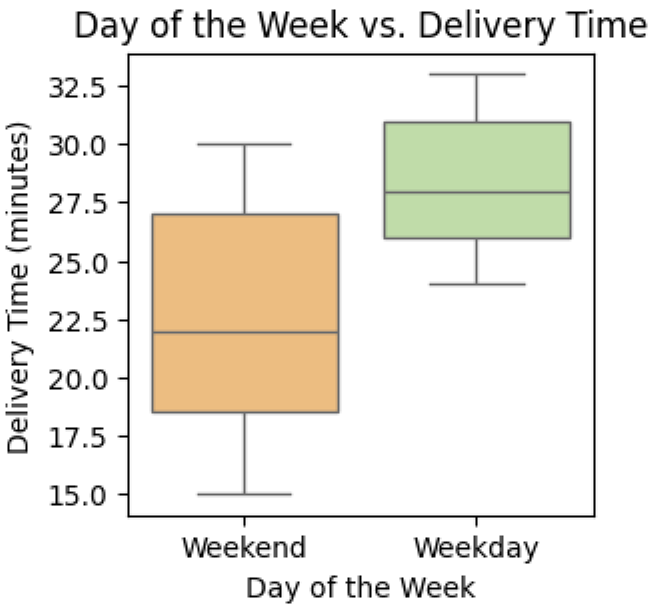
In [ ]:

```
# Compare relations between day_of_the_week and other variables

# day_of_the_week vs. rating
plt.figure(figsize=(3,3))
sns.boxplot(x='day_of_the_week', y='rating', data=df, palette='Spectral')
plt.title('Day of the Week vs. Rating')
plt.xlabel('Day of the Week')
plt.ylabel('Rating')
plt.show()

# day_of_the_week vs. food_preparation_time
plt.figure(figsize=(3,3))
sns.boxplot(x='day_of_the_week', y='food_preparation_time', data=df, palette='Spect
plt.title('Day of the Week vs. Food Preparation Time')
plt.xlabel('Day of the Week')
plt.ylabel('Food Preparation Time (minutes)')
plt.show()
```

```
# day_of_the_week vs. delivery_time
plt.figure(figsize=(3,3))
sns.boxplot(x='day_of_the_week', y='delivery_time', data=df, palette='Spectral')
plt.title('Day of the Week vs. Delivery Time')
plt.xlabel('Day of the Week')
plt.ylabel('Delivery Time (minutes)')
plt.show()

# Average delivery time grouped by day of the week
plt.figure(figsize=(3,3))
average_delivery_time_by_day = df.groupby('day_of_the_week')['delivery_time'].mean(
print(average_delivery_time_by_day)
```







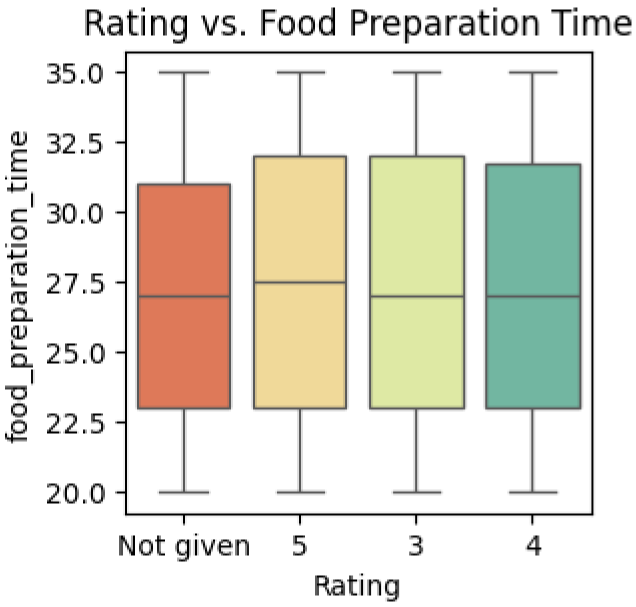
```
day_of_the_week
Weekday    28.340037
Weekend    22.470022
Name: delivery_time, dtype: float64
<Figure size 300x300 with 0 Axes>
```

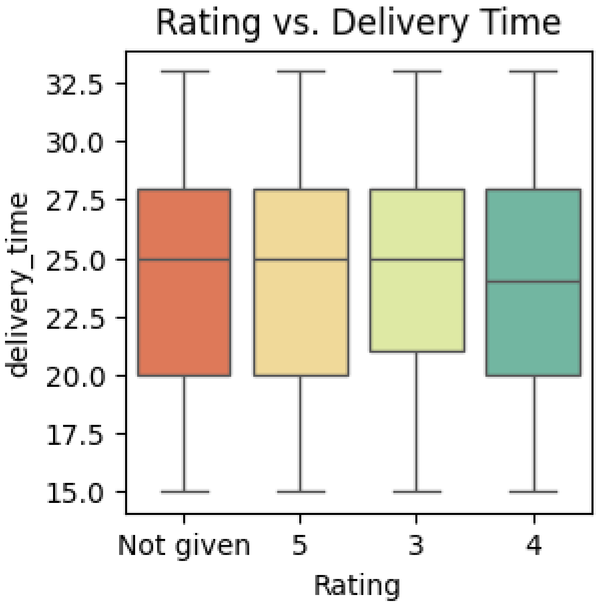
In [ ]:

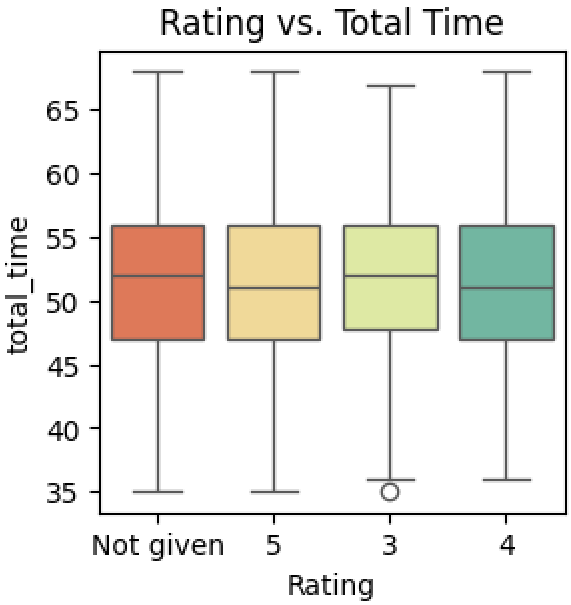
```
# Compare relations between rating and other variables

# Create a boxplot to compare rating vs. food_preparation_time
plt.figure(figsize=(3,3))
sns.boxplot(x='rating', y='food_preparation_time', data=df, palette='Spectral')
plt.title('Rating vs. Food Preparation Time')
plt.xlabel('Rating')

# Create a boxplot to compare rating vs. delivery_time
plt.figure(figsize=(3,3))
sns.boxplot(x='rating', y='delivery_time', data=df, palette='Spectral')
plt.title('Rating vs. Delivery Time')
plt.xlabel('Rating')

# Create a boxplot to compare rating vs. total_time
plt.figure(figsize=(3,3))
# Calculate total time and add it to the DataFrame
if 'total_time' not in df.columns:
    df['total_time'] = df['food_preparation_time'] + df['delivery_time']
sns.boxplot(x='rating', y='total_time', data=df, palette='Spectral')
plt.title('Rating vs. Total Time')
plt.xlabel('Rating')
```

Out[ ]:

```
Text(0.5, 0, 'Rating')
```


Rating vs. Food Preparation Time
food_preparation_time
35.0
32.5
30.0
27.5
25.0
22.5
20.0
Not given
5
3
4
Rating


Rating vs. Delivery Time
delivery_time
32.5
30.0
27.5
25.0
22.5
20.0
17.5
15.0
Not given
5
3
4
Rating

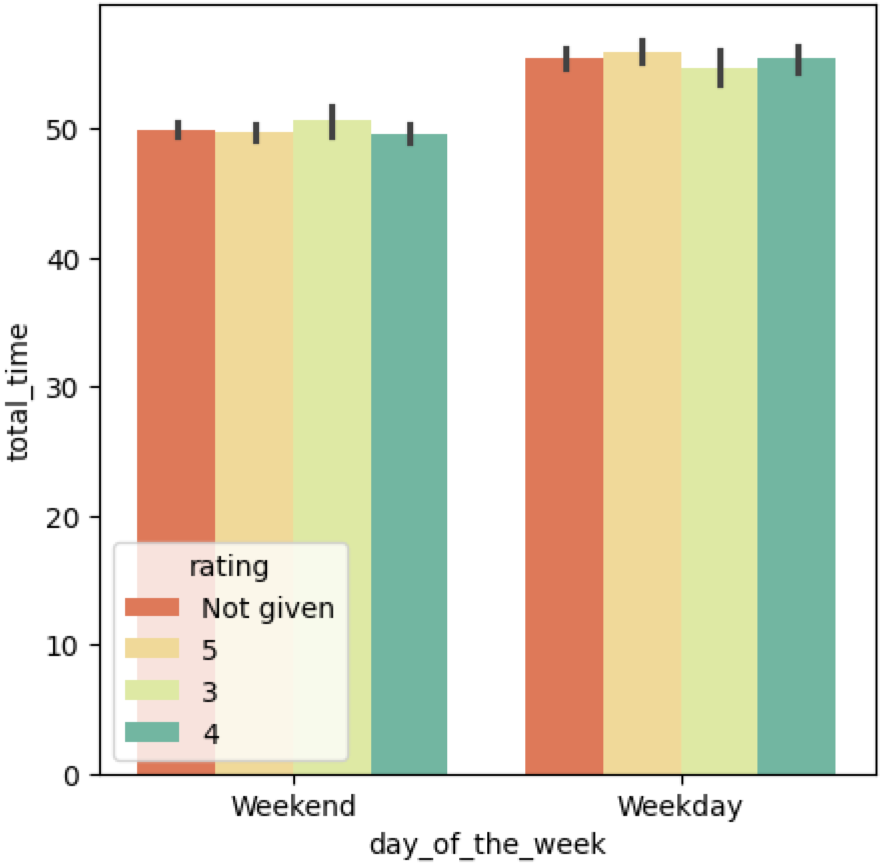

Rating vs. Total Time
total_time
65
60
55
50
45
40
35
Not given
5
3
4
Rating

In [ ]:

```
# Create a graph comparing day_of_the_week based on total_time with a hue of rating
plt.figure(figsize=(5,5))
sns.barplot(x='day_of_the_week', y='total_time', data=df, hue='rating', palette='Sp
```

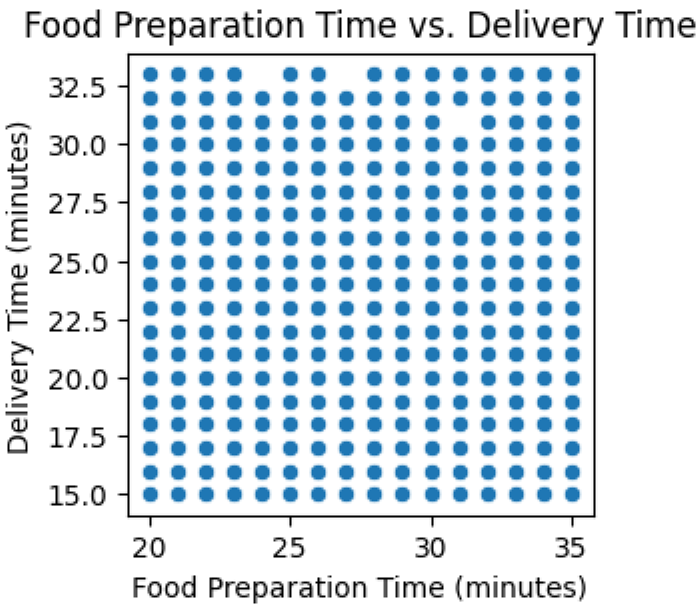
Out[ ]:

```
<Axes: xlabel='day_of_the_week', ylabel='total_time'>
```



In [ ]:

```
# Compare food_preparation_time and delivery time
plt.figure(figsize=(3,3))
sns.scatterplot(x='food_preparation_time', y='delivery_time', data=df)
plt.title('Food Preparation Time vs. Delivery Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Delivery Time (minutes)')
plt.show()
```



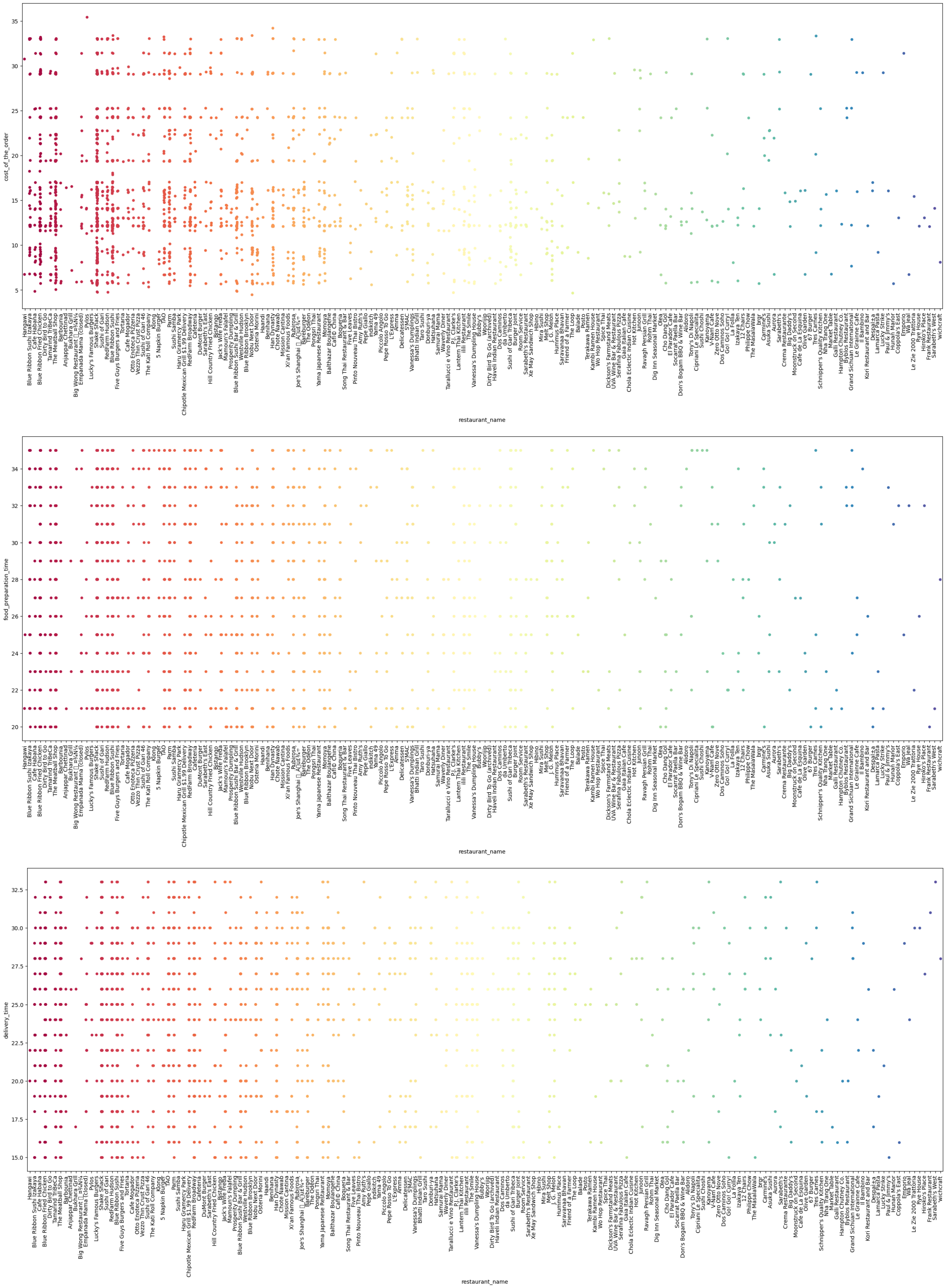
Observation: Amongst numerical values, there does not seem to be huge correlations. The exception being the relationship between day of the week and delivery time. During the week days, delivery time takes an average of 5.87 minutes longer than it does on the weekend.

In [179…

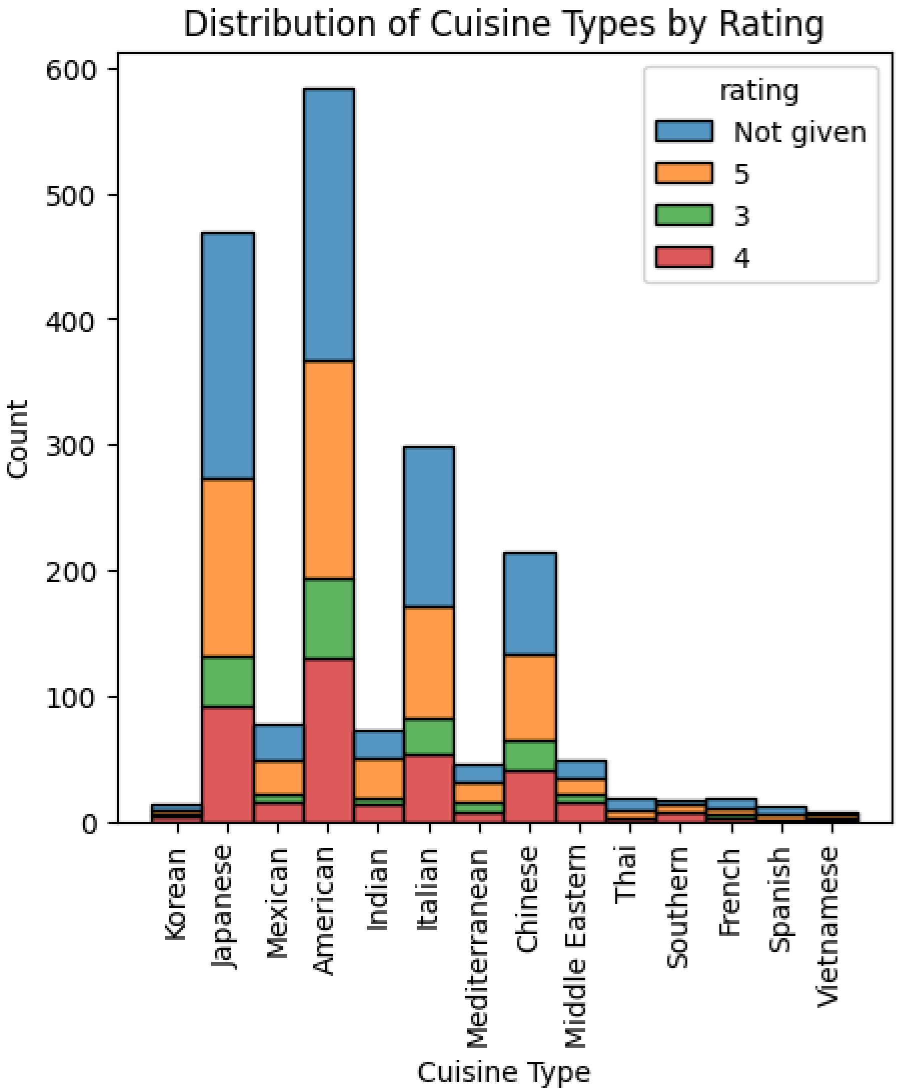
```
# NOTE TO THE PROJECT GRADERS: These plotly graphs do not show in the html file. Th
import plotly.express as px

# Compare restaurant_name to cost_of_order
fig = px.scatter(df, x="restaurant_name", y="cost_of_the_order", title="Restaurant
fig.show()


# Compare restaurant_name to food_preparation_time
fig = px.scatter(df, x="restaurant_name", y="food_preparation_time", title="Restaur
fig.show()


# Compare restaurant_name to delivery_time
fig = px.scatter(df, x="restaurant_name", y="delivery_time", title="Restaurant vs D
fig.show()
```

In [186…

```
# HTML friendly version:

# Compare restaurant_name to cost_of_order using sns strip plot
plt.figure(figsize=(30, 10))
sns.stripplot(x='restaurant_name', y='cost_of_the_order', data=df, jitter=True, pal
plt.xticks(rotation=90)
plt.show()

# Compare restaurant_name to food_preparation_time using sns strip plot
plt.figure(figsize=(30, 10))
sns.stripplot(x='restaurant_name', y='food_preparation_time', data=df, jitter=True,
plt.xticks(rotation=90)
plt.show()

# Compare restaurant_name to delivery_time using sns strip plot
plt.figure(figsize=(30, 10))
sns.stripplot(x='restaurant_name', y='delivery_time', data=df, jitter=True, palette
plt.xticks(rotation=90)
plt.show()
```



```
In [ ]: # Create a histogram comparing number of orders with cuisine types and using rating
        plt.figure(figsize=(5, 5))
        sns.histplot(data=df, x='cuisine_type', hue='rating', multiple='stack')
        plt.title('Distribution of Cuisine Types by Rating')
        plt.xlabel('Cuisine Type')
        plt.xticks(rotation=90)
        plt.show()

        # Print the number of orders per cuisine per rating
        df.groupby(['cuisine_type', 'rating']).size()
```



Out[ ]:

		0
cuisine_type	**rating**	
American	**3**	64
	4	130
	5	174
	Not given	216
Chinese	**3**	24
	4	40
	5	69
	Not given	82
French	**3**	2
	4	3
	5	5
	Not given	8
Indian	**3**	5
	4	13
	5	32
	Not given	23
Italian	**3**	28
	4	54
	5	90
	Not given	126
Japanese	**3**	40
	4	91
	5	142
	Not given	197
Korean	**3**	2
	4	4
	5	3
	Not given	4
Mediterranean	**3**	9

		0
cuisine_type	rating	
	4	7
	5	16
	Not given	14
Mexican	3	6
	4	16
	5	26
	Not given	29
Middle Eastern	3	5
	4	16
	5	13
	Not given	15
Southern	3	1
	4	7
	5	5
	Not given	4
Spanish	4	1
	5	5
	Not given	6
Thai	4	3
	5	6
	Not given	10
Vietnamese	3	2
	4	1
	5	2
	Not given	2

dtype: int64

Observation: Restaurants have relatively even distribution amongst cost of order, food prep time per order, and delivery time per order. Here we can break down each individual order to the respected restaurant.

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 [ ]:

```
# Group rating data by restaurant name
restaurant_ratings = df.groupby('restaurant_name')['rating']
rating_counts = restaurant_ratings.count()

# Convert 'rating' to numeric before calculating the mean to avoid potential errors
average_ratings = restaurant_ratings.apply(pd.to_numeric, errors='coerce').groupby(

# Filter restaurants based on the criteria
promotional_restaurants = average_ratings.index[(rating_counts > 50) & (average_rat

# Create a DataFrame with restaurant names, rating counts, and average ratings
promotional_df = pd.DataFrame({'rating_count': rating_counts[promotional_restaurant
promotional_df
```

Out[ ]:

	rating_count	average_rating
restaurant_name		
Blue Ribbon Fried Chicken	96	4.328125
Blue Ribbon Sushi	119	4.219178
Parm	68	4.128205
RedFarm Broadway	59	4.243902
RedFarm Hudson	55	4.176471
Shake Shack	219	4.278195
The Meatball Shop	132	4.511905

Observations: The restaurants that fit the criteria to receive a promotional offer are Blue Ribbon Fried Chicken, Blue Ribbon Sushi, Parm, RedFarm Broadway, RedFarm Hudson, Shake Shack, and The Meatball Shop.

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 [ ]:

```
# Find orders with cost_of_the_order > 20. Total the sum of the orders and multiply
orders_above_20 = df[df['cost_of_the_order'] > 20]
revenue_above_20 = orders_above_20['cost_of_the_order'].sum() * 0.25
```

```
# Filter orders with cost > 5 but <= 20. Multiply by .15
orders_above_5 = df[(df['cost_of_the_order'] > 5) & (df['cost_of_the_order'] <= 20)
revenue_above_5 = orders_above_5['cost_of_the_order'].sum() * 0.15

# Calculate the total revenue
total_revenue = revenue_above_20 + revenue_above_5
total_revenue = round(total_revenue, 2)

print(f"Total revenue generated by the company: ${total_revenue:.2f}")
```

```
Total revenue generated by the company: $6166.30
```

Observations:

Total revenue generated by the company is $6166.30

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 [ ]:

```
# Calculate total time
df['total_time'] = df['food_preparation_time'] + df['delivery_time']

# Calculate the percentage of orders with total time greater than 60 minutes
percentage_orders_over_60 = (len(df[df['total_time'] > 60]) / len(df)) * 100
print(f"{percentage_orders_over_60:.2f}% of orders take more than 60 minutes to get
```

```
10.54% of orders take more than 60 minutes to get delivered from the time the order
was placed.
```

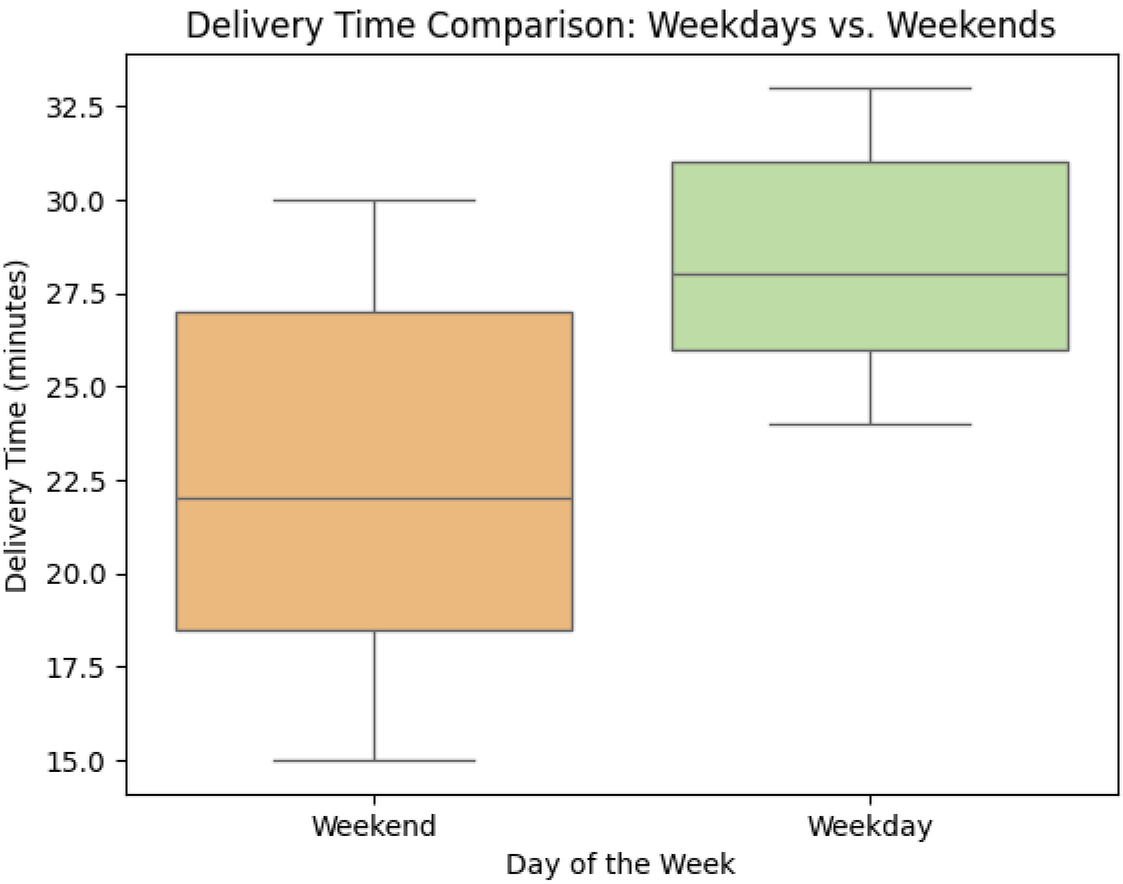
Observations:

10.54% of orders take more than 60 minutes to get delivered from the time the order was placed.

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 [ ]:

```
#Boxplot delivery time weekday vs. weekend
sns.boxplot(x='day_of_the_week', y='delivery_time', data=df, palette='Spectral')
plt.title('Delivery Time Comparison: Weekdays vs. Weekends')
plt.xlabel('Day of the Week')
plt.ylabel('Delivery Time (minutes)')
plt.show()

# Show mean delivery times
mean_delivery_time = df.groupby('day_of_the_week')['delivery_time'].mean()
mean_delivery_time
```



Out[ ]:

	delivery_time
day_of_the_week	
Weekday	28.340037
Weekend	22.470022

dtype: float64

Observations: Weekday deliveries are 5.87 minutes slower on average.

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:

- Weekday deliveries are taking longer than weekend deliveries by nearly 6 minutes, even though 71% of orders are placed on weekends.

Recommendations:

- FoodHub could bring in more revenue if there were more orders placed on the 5 weekdays. Currently only 29% of orders are being placed on those days and it could be attributed to the fact that delivery time takes longer. However, if there were discounts available to the top 5 popular cuisine types (American, Japanese, Italian, Chinese, and Mexican), it could encourage more customers to purchse during the weekday.
