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

```
# 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 --user
```

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

```
# 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

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

```
Mounted at /content/drive
```

```
# Write your code here to read the data
df = pd.read_csv('/content/drive/MyDrive/Python Course Test Project/foodhub_order.csv')
```

```
#Custom Color Set
ElleSet = [
    (102/255, 194/255, 165/255),  # Muted Green
    (214/255, 95/255, 95/255),    # Muted Red
    (141/255, 160/255, 203/255),  # Soft Blue
    (130/255, 198/255, 226/255),  # Muted Blue
    (166/255, 216/255, 84/255),   # Lime Green
    (230/255, 196/255, 148/255),  # Beige
    (179/255, 179/255, 179/255),  # Neutral Gray
    (255/255, 217/255, 47/255)    # Yellow
]

# Function to apply ElleSet globally in Seaborn
def use_ElleSet():
    sns.set_palette(ElleSet)
```

```
import os
```

```
os.environ["PYTHONWARNINGS"] = "ignore::FutureWarning"
```

```
# Write your code here to view the first 5 rows
df.head()
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
0	1477147	337525	Hangawi	Korean	30.75	Weekend	Not given	25	20
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25	23
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23	28
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25	15
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25	24

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

```
# Write your code here
df.shape
```

```
(1898, 9)
```

Observations:

There are 1898 rows and 9 columns within the dataset.

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

```
# Write your code here
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:

The dataset contains four object, four integer, and one float data types across its columns.

Recommended cleaning:

- customer_id and order_id are described as integers. These can be converted to category data type to ensure they are treated as categorical identifiers rather than numerical values. This will prevent uninteded mathmatical operation and ensure accurate analysis. Additionally, category type will ensure certain Seaborn visualization output is optimized.
- restaurant_name, cuisine_type,and day_of_the_week have all been described as objects. Converting them to categorical data type will optimize visualizations as described above improving data analysis outcomes.
- rating is stored as an object. This should be converted to a numerical value type to allow for calculations such as average rating, most frequent rating, and performing other statistical analysis. 'Not Provided' response should remain captured as Nan.

```
# Convert customer_id and order_id data types
df[['customer_id','order_id']] = df[['customer_id','order_id']].astype('category')
```

```
# Convert restaurant_name, cuisine_type, day_of_the_week data types
df[['restaurant_name', 'cuisine_type','day_of_the_week']] = df[['restaurant_name', 'cuisine_type','day_of_the_week']].astype('category')
```

```
# Convert rating to numerical value type, int
# Replace 'Not given' with NaN and then convert to numeric
df['rating'] = pd.to_numeric(df['rating'], errors='coerce').astype('Int64')
```

```
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   category
 1   customer_id            1898 non-null   category
 2   restaurant_name        1898 non-null   category
 3   cuisine_type           1898 non-null   category
 4   cost_of_the_order      1898 non-null   float64
 5   day_of_the_week        1898 non-null   category
 6   rating                 1162 non-null   Int64
 7   food_preparation_time  1898 non-null   int64
 8   delivery_time          1898 non-null   int64
dtypes: Int64(1), category(5), float64(1), int64(2)
memory usage: 203.4 KB
```

Actions:

- customer_id and order_id converted from int to category data type.
- restaurant_name, cuisine_type, and day_of_the_week converted from object to category data types.
- rating converted to int, maintaining Not Provided results as Nan.

Updated summary below.

```
df.describe(include = 'all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
order_id	1898.0	1898.0	1476547.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
customer_id	1898.0	1200.0	52832.0	13.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
restaurant_name	1898	178	Shake Shack	219	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cuisine_type	1898	14	American	584	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cost_of_the_order	1898.0	NaN	NaN	NaN	16.498851	7.483812	4.47	12.08	14.14	22.2975	35.41
day_of_the_week	1898	2	Weekend	1351	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rating	1162.0	<NA>	<NA>	<NA>	4.344234	0.741478	3.0	4.0	5.0	5.0	5.0
food_preparation_time	1898.0	NaN	NaN	NaN	27.37197	4.632481	20.0	23.0	27.0	31.0	35.0
delivery_time	1898.0	NaN	NaN	NaN	24.161749	4.972637	15.0	20.0	25.0	28.0	33.0

Observations:

- Customer ID indicates 13 unique orders = top repeat order count by a single customer id
- Shack Shake is top perforing Restaurant
- American is top cuisine type
- Mean cost of order = 16$ with a range = 4.47 to 35.41
- Mean Food Prep = 27 minutes with a range = 20 min to 35 min
- Mean Delivery Time = 24 minutes with a range = 15 min to 33 min
- Combined Overall Mean Wait Time = 51 minutes

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

```
# write your code here
df.isnull().sum()
```

	0
order_id	0
customer_id	0
restaurant_name	0
cuisine_type	0
cost_of_the_order	0
day_of_the_week	0
rating	736
food_preparation_time	0
delivery_time	0

dtype: int64

Observations:

There are 736 missing values in 'rating' column.

?Are the missing values Missing Completely at Random (MCAR) or are they related to other factors that may demonstrate there is a pattern or relationship between the missing ratings and other variables in the dataset?

Areas there is most likely to be a correlation:

- food prep time & delivery time
- reviewed as: overall wait time
- cost of the order
- day of the week

Food Prep Time and Delivery Time Reviewed as Overall Wait Time

```python
# Locate Missing Rankings
df.loc[df['rating'].isnull()==True]
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
0	1477147	337525	Hangawi	Korean	30.75	Weekend	<NA>	25	20
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	<NA>	25	23
6	1477894	157711	The Meatball Shop	Italian	6.07	Weekend	<NA>	28	21
10	1477895	143926	Big Wong Restaurant _¤¾Ñ¼	Chinese	5.92	Weekday	<NA>	34	28
14	1478198	62667	Lucky's Famous Burgers	American	12.13	Weekday	<NA>	23	30
...	...	...	...	...	...	...	...	...	...
1887	1476873	237616	Shake Shack	American	5.82	Weekend	<NA>	26	30
1891	1476981	138586	Shake Shack	American	5.82	Weekend	<NA>	22	28
1892	1477473	97838	Han Dynasty	Chinese	29.15	Weekend	<NA>	29	21
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	<NA>	31	24
1897	1478056	120353	Blue Ribbon Sushi	Japanese	19.45	Weekend	<NA>	28	24

736 rows × 9 columns

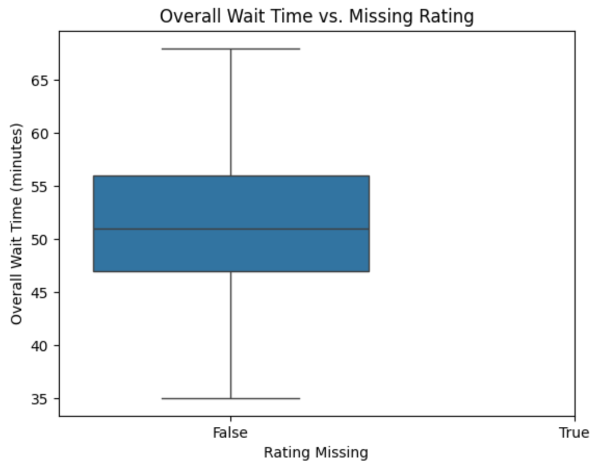
```python
# Create a new column 'overall_wait_time'
df['overall_wait_time'] = df['food_preparation_time'] + df['delivery_time']

# Display the updated DataFrame
display(df)
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time	overall_wait_time
0	1477147	337525	Hangawi	Korean	30.75	Weekend	<NA>	25	20	45
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	<NA>	25	23	48
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23	28	51
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25	15	40
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25	24	49
...	...	...	...	...	...	...	...	...	...	...
1893	1476701	292602	Chipotle Mexican Grill $1.99 Delivery	Mexican	22.31	Weekend	5	31	17	48
1894	1477421	397537	The Smile	American	12.18	Weekend	5	31	19	50
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	<NA>	31	24	55
1896	1477513	64151	Jack's Wife Freda	Mediterranean	12.18	Weekday	5	23	31	54
1897	1478056	120353	Blue Ribbon Sushi	Japanese	19.45	Weekend	<NA>	28	24	52

1898 rows × 10 columns

```python
 # View Boxplot Overall Wait Time
sns.boxplot(x=df['rating'].isnull().astype(str), y=df['overall_wait_time'])
plt.title('Overall Wait Time vs. Missing Rating')
plt.xlabel('Rating Missing')
plt.ylabel('Overall Wait Time (minutes)')
plt.xticks([0, 1], ['False', 'True'])
plt.show()
```



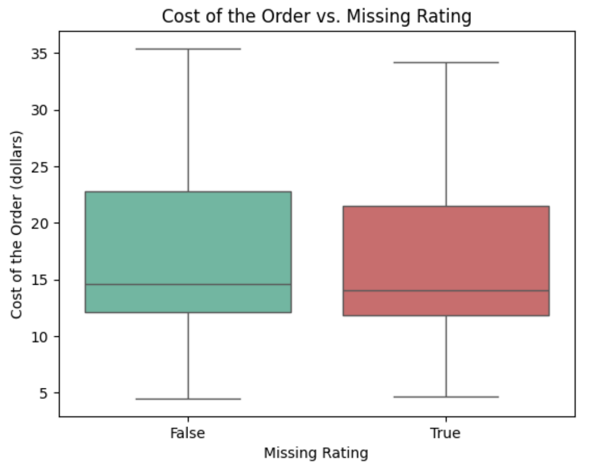
Observations:

The median overall wait time is slightly higher for orders without ratings, but the difference is not significant.

NOTE TO INSTRUCTOR: I reran this calculation after making changes to df. I did not save original. I do not want to risk resetting and messing up other parts of my submittal. I am leaving it as is now but originally it did successfully show true responses as well.

Cost of Order Review

```
# View BoxPlot Cost of Order
num_colors = len(df['rating'].isnull().unique()) # Dynamically get number of needed colors
sns.boxplot(x=df['rating'].isnull(), y=df['cost_of_the_order'], hue=df['rating'].isnull(), palette=ElleSet[:num_colors], legend=False) # Use required colors
plt.title('Cost of the Order vs. Missing Rating')
plt.xlabel('Missing Rating')
plt.ylabel('Cost of the Order (dollars)')
plt.show()
```



Observation

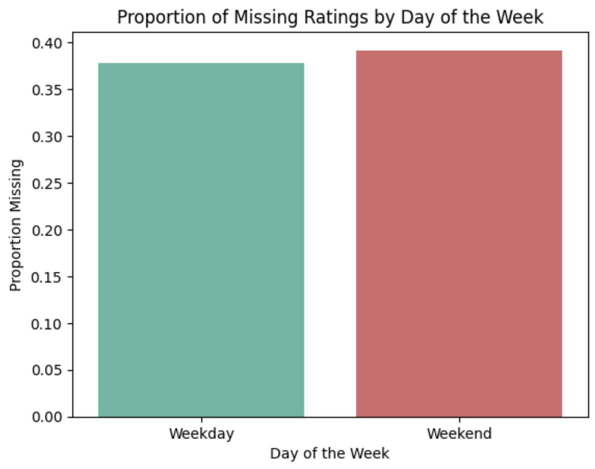
The median cost of the order is slightly lower for orders without ratings, but the difference is not significant.

Day of the Week Review

```
# View Day of the Week
# Group data by day of the week and calculate the proportion of missing ratings
day_missing_prop = df.groupby('day_of_the_week', observed=False)['rating'].apply(lambda x: x.isnull().mean())

# Create the bar plot with ElleSet palette and addressing warnings
num_colors = len(day_missing_prop.index.unique())  # Get the number of unique days
sns.barplot(x=day_missing_prop.index, y=day_missing_prop.values, hue=day_missing_prop.index, palette=ElleSet[:num_colors], dodge=False)
```

```
plt.title('Proportion of Missing Ratings by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Proportion Missing')
plt.show()
```



Observation

There is an insignificant difference in the proportion of missing ratings between weekdays and weekends.

Findings

Missing Value Treatment

Based upon this review, the removal of the missing data is unlikely to lead to the loss of valuable information. The missing ratings appear to be missing at random and will be removed from the dataset.

```
# Delete Rankings with Missing Values
df.dropna(subset=['rating'], inplace=True)
```

```
# Confirm Clean
df.isnull().sum()
```

	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
overall_wait_time	0

dtype: int64

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]

```
# Get the summary statistics of the numerical data
summary_stats = df.describe()

# Extract and print the minimum, average, and maximum food preparation time
min_prep_time = summary_stats.loc['min', 'food_preparation_time']
avg_prep_time = summary_stats.loc['mean', 'food_preparation_time']
max_prep_time = summary_stats.loc['max', 'food_preparation_time']

print(f"Minimum food preparation time: {min_prep_time} minutes")
print(f"Average food preparation time: {avg_prep_time} minutes")
print(f"Maximum food preparation time: {max_prep_time} minutes")
```

```
Minimum food preparation time: 20.0 minutes
Average food preparation time: 27.381239242685027 minutes
Maximum food preparation time: 35.0 minutes
```

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

```
df['rating'].value_counts()
```

rating	count
5	588
4	386
3	188

dtype: Int64

```
num_unrated_orders = df['rating'].isnull().sum()

# Print the result
print("Number of unrated orders:", num_unrated_orders)
```

```
Number of unrated orders: 0
```

Observations:

There are currently '0' unrated orders. The original data set contained 736 orders that were not rated; however, the missing rankings have been cleaned from the data while addressing Question no. 3.

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]

```
# Explore Variables
df.describe(include = 'all').T
```

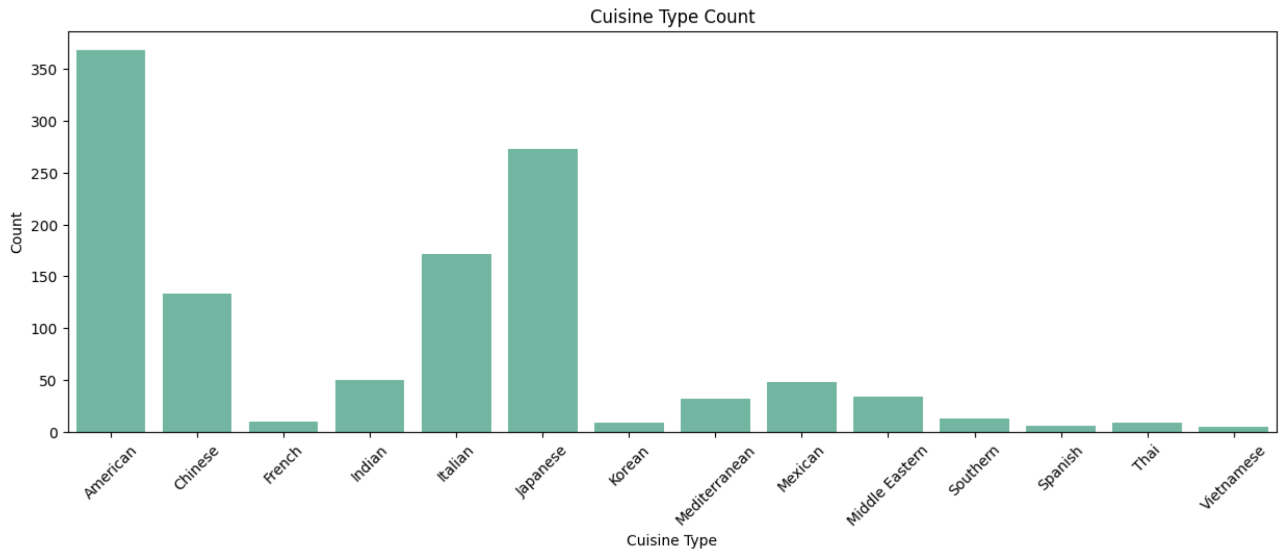
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
order_id	1162.0	1162.0	1476547.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
customer_id	1162.0	859.0	52832.0	7.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
restaurant_name	1162	156	Shake Shack	133	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cuisine_type	1162	14	American	368	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cost_of_the_order	1162.0	NaN	NaN	NaN	16.760766	7.572578	4.47	12.13	14.6	22.75	35.41
day_of_the_week	1162	2	Weekend	822	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rating	1162.0	<NA>	<NA>	<NA>	4.344234	0.741478	3.0	4.0	5.0	5.0	5.0
food_preparation_time	1162.0	NaN	NaN	NaN	27.381239	4.677922	20.0	23.0	27.0	32.0	35.0
delivery_time	1162.0	NaN	NaN	NaN	24.154045	4.930999	15.0	20.0	25.0	28.0	33.0
overall_wait_time	1162.0	NaN	NaN	NaN	51.535284	6.767522	35.0	47.0	51.0	56.0	68.0

Observations

- Customer ID indicates 7 unique orders = top repeat order count by a single customer id
- There are 14 unique cuisine types and 156 unique restaurant names
- Shack Shake is top performing restaurant
- American is top cuisine type
- Mean cost of order = 14.60$ with a range = 4.47 to 35.41
- Mean Food Prep = 27 minutes with a range = 20 min to 35 min
- Mean Delivery Time = 24 minutes with a range = 15 min to 33 min
- Combined Overall Mean Wait Time = 51 minutes

Cuisine Type

```python
# View Cuisine Type Data
plt.figure(figsize = (15,5))
sns.countplot(data = df, x = 'cuisine_type', color=ElleSet[0]);
plt.title('Cuisine Type Count');
plt.xlabel('Cuisine Type');
plt.ylabel('Count');
plt.xticks(rotation=45);
plt.show()
```

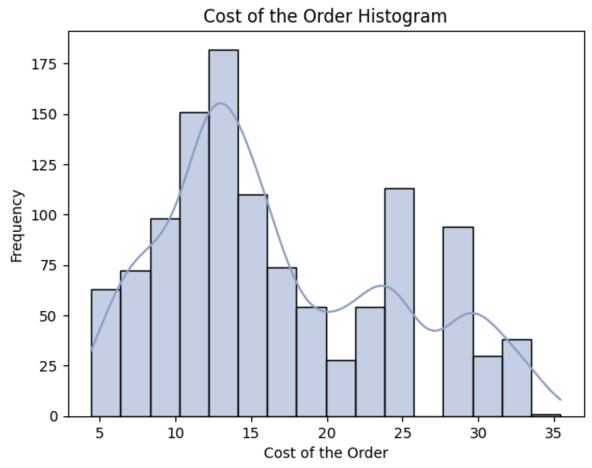


Observations

- American cuisine type has the highest order count.
- Japanese follows in second place.
- The majority of cuisine types are =<50 orders per restaurant.
- There are only 4 restaurants with orders above 50. These are: Chinesse, Italian, Japaneses and American - in progressive order.

Cost of the Order

```python
## Histogram for the cost of order
sns.histplot(data=df,x='cost_of_the_order', color=ElleSet[2], kde=True)
plt.title('Cost of the Order Histogram')
plt.xlabel('Cost of the Order')
plt.ylabel('Frequency')
plt.show()
```



Observations

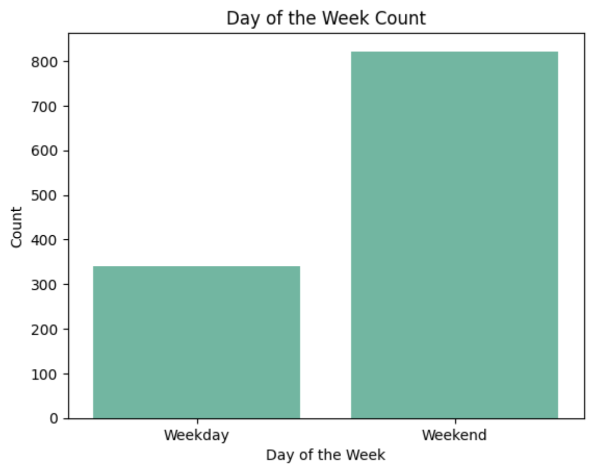
- Order costs between 12 and 14 dollars are most frequent, peaking at 175 orders.
- Smaller frequency peaks occur around 25 dollars (110 orders) and 25 dollars (110 orders) and 29 dollars (100 orders).
- Overall, the distribution of order costs is left-skewed, indicating most orders are lower in price.

Day of the Week

```
df['day_of_the_week'].nunique()
```

```
2
```

```
sns.countplot(data = df, x = 'day_of_the_week', color=ElleSet[0]);
plt.title('Day of the Week Count');
plt.xlabel('Day of the Week');
plt.ylabel('Count');
plt.show()
```



Observations

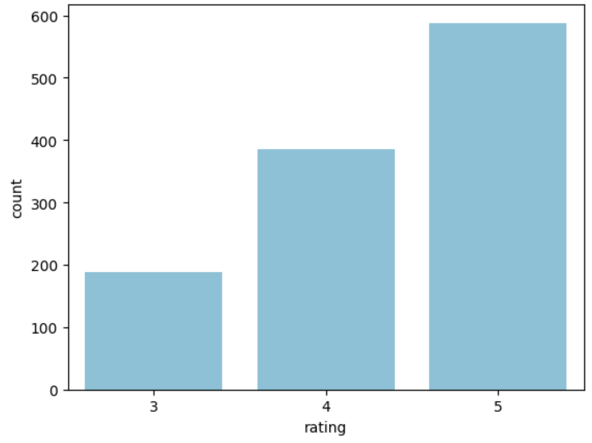
- Higher weekend demand: The data clearly shows a preference for ordering food on weekends, likely due to increased leisure time and social gatherings.
- Weekday opportunity: This suggests a potential opportunity to boost weekday sales through targeted promotions or menu adjustments catered to busy professionals or students.
- Resource allocation: FoodHub should ensure sufficient staff and delivery resources are available during peak weekend hours to maintain service quality
- Customer behavior: This pattern reflects customer behavior and lifestyle choices, indicating a need for FoodHub to align its strategies with these trends.
- Weekend marketing focus: It would be beneficial to focus marketing efforts on weekend-specific deals or promotions to further capture this demand.

Service Rating

```
# Unique Values
df['rating'].unique()
```

```
<IntegerArray>
[5, 3, 4]
Length: 3, dtype: Int64
```

```
sns.countplot(data = df, x = 'rating', color=ElleSet[3]);
```



Observations:

- Most ratings are positve, concentrated at 5 with no responses below 3. However, there were 736 unrated orders highlighting a need to improve feedback response rates.
- Despite high satisfaction, investigating the reasons for 3 and 4-star ratings, as well as the low overall response rate, can reveal areas for improvement and enhance customer loyalty.

Food Preperation

```
df.describe()[['food_preparation_time']]
```

	food_preparation_time
count	1162.000000
mean	27.381239
std	4.677922
min	20.000000
25%	23.000000
50%	27.000000
75%	32.000000
max	35.000000

Observations:

- Median food prep time is 27 minutes. Range is from 20 minutes up to 35 minutes. *While most orders are prepared within a reasonable timeframe, FoodHub should investigate the factors contributing to the longest preparation times (approaching 35 minutes) to identify potential bottlenecks and ensure consistently efficient service.

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

```
df['restaurant_name'].value_counts().head(5)
```

	count
restaurant_name	
Shake Shack	133
The Meatball Shop	84
Blue Ribbon Sushi	73
Blue Ribbon Fried Chicken	64
RedFarm Broadway	41

dtype: int64

Observations:

- Shake Shake significantly outperforms other restaurants in quantity of orders.

The top 5 restaurants are primarily comfort food type (with the exception of the sushi restaurant).

- The high order volume for Shake Shack and the dominance of comfort food restaurants in the top 5 suggest a strong customer preference for familiar, convenient, and potentially indulgent food options. This trend could indicate a market opportunity for FoodHub to partner with more comfort food establishments and leverage Shake Shack's popularity through promotions or strategic collaborations.

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

```
df_weekend = df[df['day_of_the_week'] == 'Weekend']
df_weekend['cuisine_type'].value_counts().idxmax()
```

```
'American'
```

```
df_weekday = df[df['day_of_the_week'] == 'Weekday']
df_weekday['cuisine_type'].value_counts().idxmax()
```

```
'American'
```

Observations:

- American is consistently the most popular cuisines, outperforming others on weedays and weekends.
- This consistent demand suggests a broad appeal across different customer segments and occasions, making it a reliable and potentially profitable focus for FoodHub and its partner restaurants.

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

```
# Get orders that cost above 20 dollars
df_greater_than_20 = df[df['cost_of_the_order']>20] ## Write the appropriate column name to get the orders having cost above $20

# Calculate the number of total orders where the cost is above 20 dollars
print('The number of total orders that cost above 20 dollars is:', df_greater_than_20.shape[0])

# Calculate percentage of such orders in the dataset
percentage = (df_greater_than_20.shape[0] / df.shape[0]) * 100

print("Percentage of orders above 20 dollars:", round(percentage, 2), '%')
```

```
The number of total orders that cost above 20 dollars is: 356
Percentage of orders above 20 dollars: 30.64 %
```

Observations:

- Price sensitivity of FoodHub customers: With nearly 70 percent of orders costing less than 20 dollars, there is a clear indication that FoodHub customers are price-conscious and tend to favor more affordable options. This suggests a strong demand for value-driven meals and promotions within that price range.
- Potential for higher-priced offerings: While the majority of orders fall below 20 dollars, the remaining 30.64 percent (356 orders) demonstrate a segment of customers willing to spend more. This presents a potential opportunity for FoodHub to explore and expand its offerings in the higher price range, potentially by partnering with premium restaurants or introducing special menu items.

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

```
df['delivery_time'].mean()
```

```
24.15404475043029
```

Observations:

- Mean delivery time is roughly 24 minutes.
- Balanced operational efficiency: The relatively equal distribution of time between delivery and food preparation suggests a balanced operational flow, where neither stage is significantly bottlenecking the overall process. This indicates a degree of efficiency in both kitchen operations and delivery logistics.
- Opportunities for holistic optimization: While the balance is positive, both delivery and food preparation present opportunities for optimization to reduce the total wait time. Improvements in either area could have a significant impact on overall efficiency and customer experience.
- Potential customer experience trade-offs: Reducing wait times in one area might impact the other. For example, speeding up delivery could put pressure on food preparation, potentially leading to quality compromises. FoodHub needs to carefully consider these trade-offs when implementing optimization strategies to ensure a balanced and positive customer experience.

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]

```
# Get the counts of  each customer_id
df['customer_id'].value_counts().head(5)
```

	count
customer_id	
52832	7
47440	7
65009	6

```
250494    6
276192    5

dtype: int64
```

Observations:

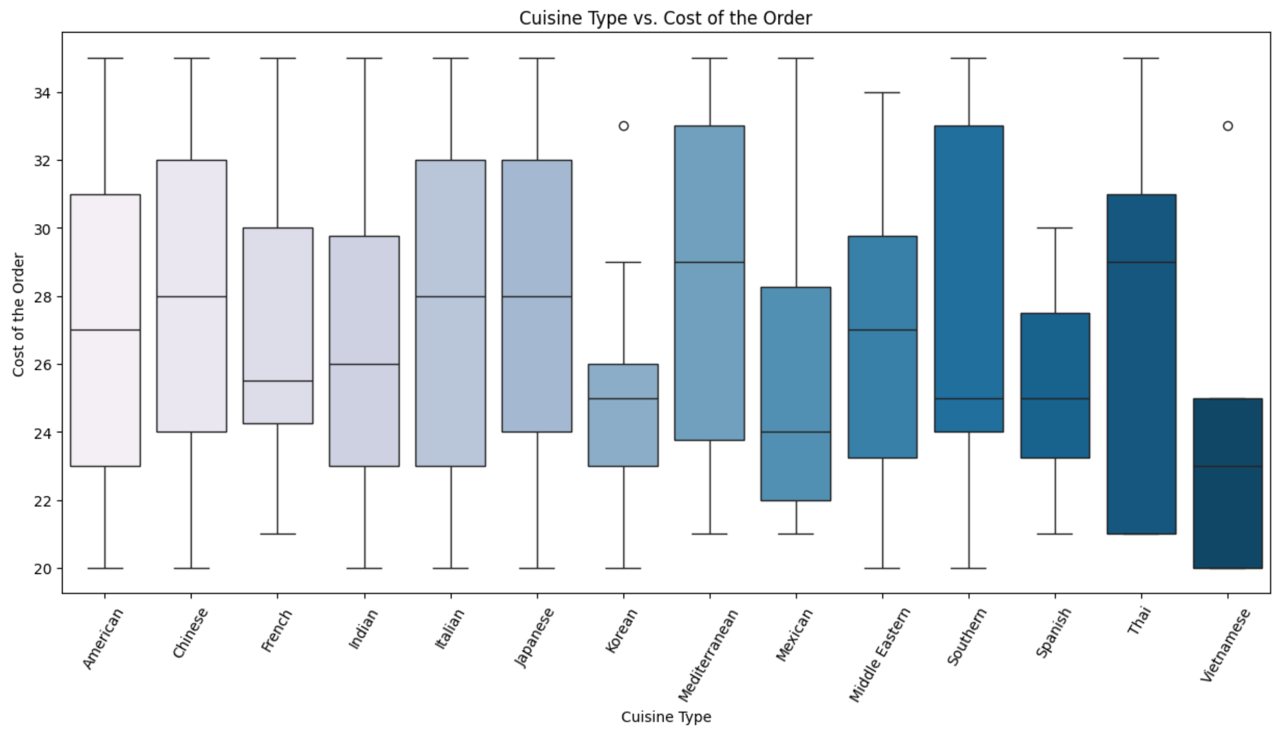
This result has changed since removing the missing data for rankings, as previously advised. While more frequent customers were identified in the original dataset, we are now providing the company with the most frequent customers who also rated their service.Perhaps this is ok?

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]

Cuisine vs. Cost of the Order

```python
# Relationship between cost of the order and cuisine type
plt.figure(figsize=(15,7))
sns.boxplot(x="cuisine_type", y="food_preparation_time", data=df, palette='PuBu', hue="cuisine_type")
plt.title('Cuisine Type vs. Cost of the Order')
plt.xlabel('Cuisine Type')
plt.ylabel('Cost of the Order')
plt.xticks(rotation = 60)
plt.show()
```



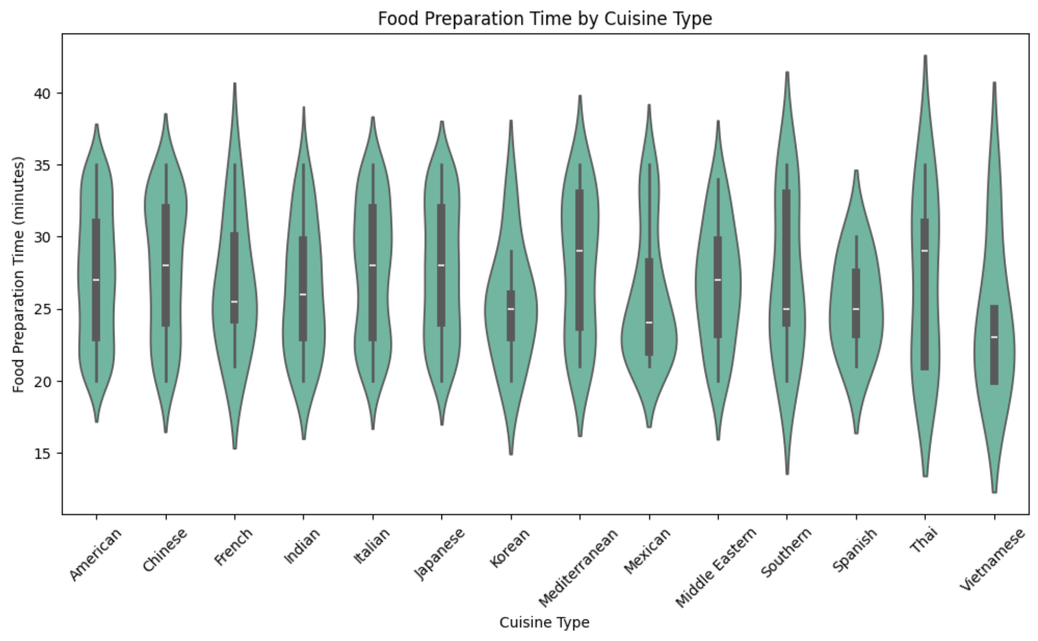
Observations

- Some cuisines (e.g., Southern, Middle Eastern, and Thai) have a wider range in order costs, while others (e.g., French, Korean, and Vietnamese) are more consistent. Japanese and Italian cuisines show higher order values, indicating a willingness to pay more for these premium options.
- FoodHub could analyze price sensitivity within each category—cuisines with higher variability may benefit from dynamic pricing or special offers.

Cusines vs Food Preperation Time

```python
plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='cuisine_type', y='food_preparation_time', color=ElleSet[0])
```

```python
plt.xlabel('Cuisine Type')
plt.ylabel('Food Preparation Time (minutes)')
plt.xticks(rotation=45)
plt.title("Food Preparation Time by Cuisine Type")
plt.show()
```



Observations

- Across all cuisine types, the majority of food preparation times fall between 20-35 minutes, with some outliers extending beyond 40 minutes.
- French, Indian, and Southern cuisines show slightly higher variability, meaning some restaurants may take significantly longer to prepare meals
- Encourage restaurants with longer prep times to optimize kitchen efficiency to reduce delivery wait times and improve customer satisfaction.

Day of the Week vs. Delivery Time

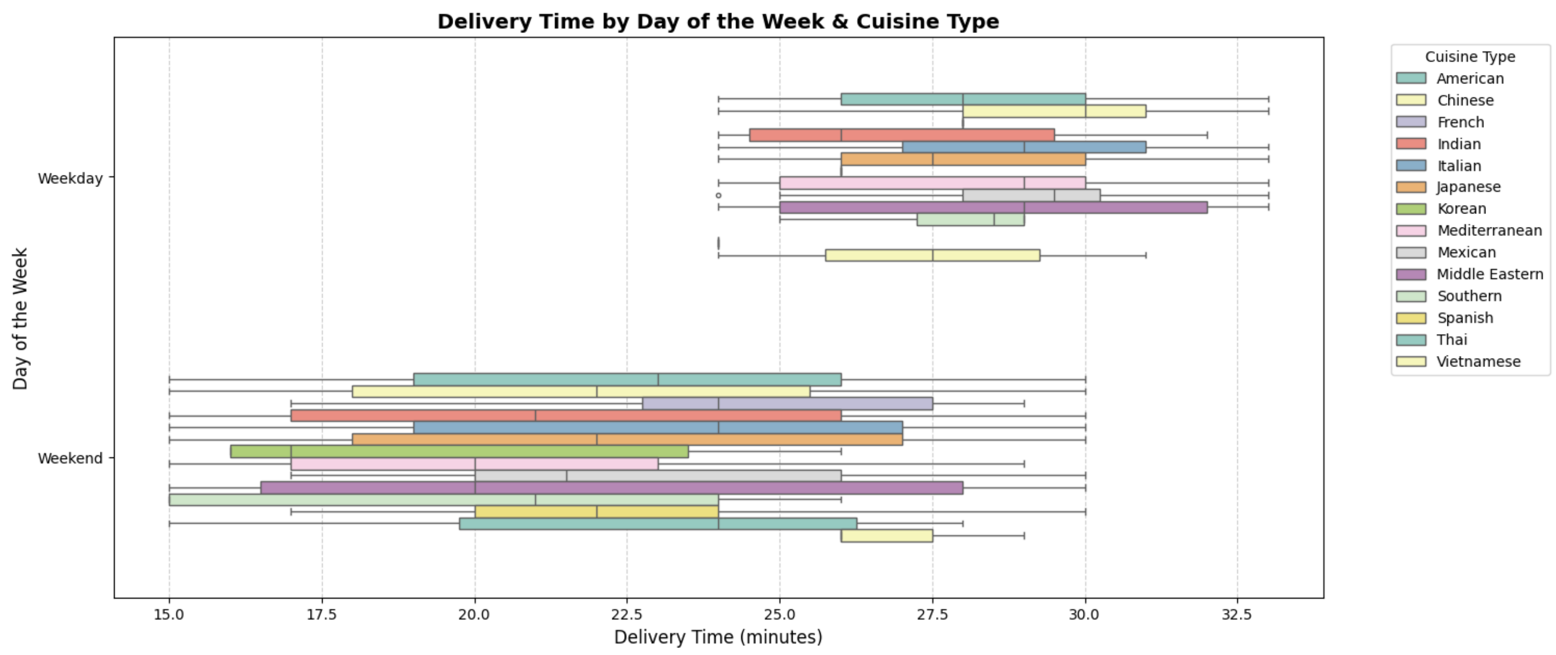
```python
# Relationship between day of the week and delivery time
plt.figure(figsize=(15,7))

sns.boxplot(
    x="delivery_time",
    y="day_of_the_week",
    data=df,
    palette="Set3",
    hue="cuisine_type",
    dodge=True,  # Separates cuisine types better
    width=0.6,  # Reduces overlapping
    fliersize=3  # Reduces size of outlier markers
)

plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.xlabel("Delivery Time (minutes)", fontsize=12)
plt.ylabel("Day of the Week", fontsize=12)
plt.title("Delivery Time by Day of the Week & Cuisine Type", fontsize=14, fontweight="bold")

# Move legend outside
plt.legend(title="Cuisine Type", bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```



Observations

- Weekend deliveries show greater variability, meaning some orders arrive very quickly, while others take significantly longer.
- However, the median delivery time on weekends is lower than on weekdays, suggesting that while some delays occur, a larger portion of orders arrive faster overall.
- FoodHub could investigate peak-hour bottlenecks on weekends to identify if specific cuisines or locations are experiencing delays.
- Optimize driver allocation for weekend demand, ensuring high-traffic areas are well covered.

Revenue by Restaurant

```
df.groupby(['restaurant_name'], observed=False)['cost_of_the_order'].sum().sort_values(ascending = False).head(14)
```

restaurant_name	cost_of_the_order
Shake Shack	2225.20
The Meatball Shop	1495.65
Blue Ribbon Sushi	1170.66
Blue Ribbon Fried Chicken	1130.59
RedFarm Broadway	680.69
Parm	655.52
RedFarm Hudson	566.86
Rubirosa	450.66
TAO	421.16
Momoya	365.78
Nobu Next Door	362.57
Five Guys Burgers and Fries	334.79
Chipotle Mexican Grill $1.99 Delivery	327.91
Han Dynasty	324.91

dtype: float64

```
# Grouping dataset to calculate total revenue per restaurant
df["total_revenue"] = df.groupby("restaurant_name", observed=True)["cost_of_the_order"].transform("sum")
```

```
#  Calculate and add total revenue per restaurant to the DataFrame
df['total_revenue'] = df.groupby('restaurant_name', observed=True)['cost_of_the_order'].transform('sum')
```

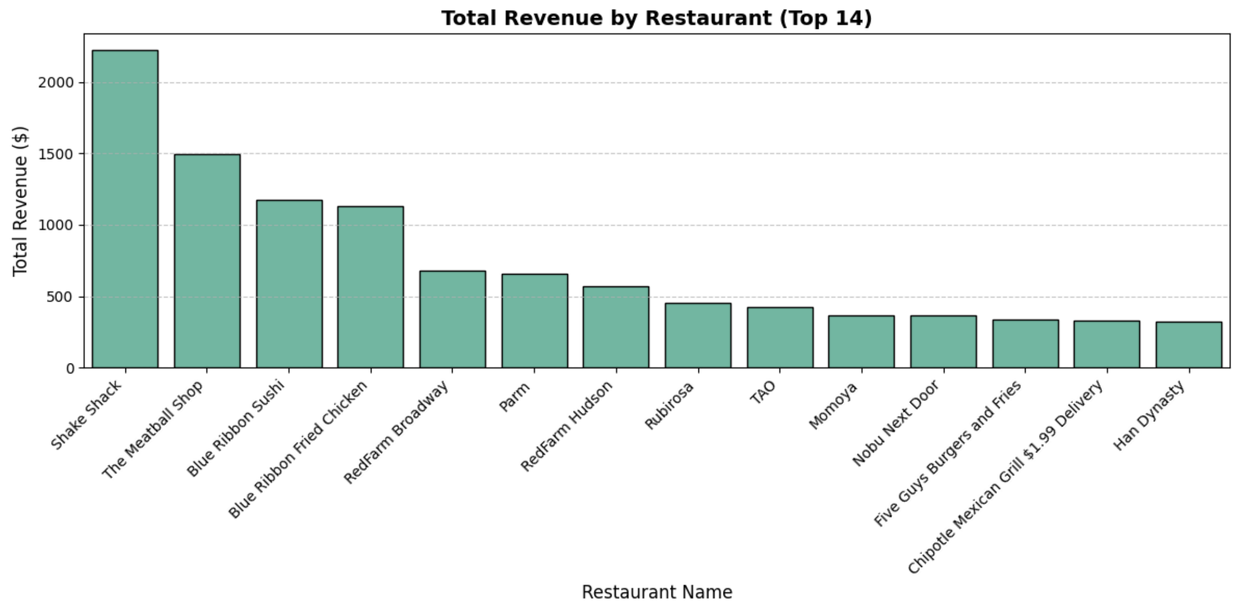
```
# Get the top 14 restaurants by total revenue
top_14_restaurants = df.groupby('restaurant_name', observed=True)['total_revenue'].first().nlargest(14).index

# Filter the DataFrame to include only the top 14 restaurants
filtered_df = df[df['restaurant_name'].isin(top_14_restaurants)]

# Plotting total revenue by restaurant for the top 14, with x-axis sorted
plt.figure(figsize=(12, 6))

# Use Seaborn to create the bar plot
sns.barplot(x='restaurant_name', y='total_revenue', data=filtered_df, order=top_14_restaurants, color=ElleSet[0], edgecolor="black")

plt.xlabel("Restaurant Name", fontsize=12)
plt.ylabel("Total Revenue ($)", fontsize=12)
plt.title("Total Revenue by Restaurant (Top 14)", fontsize=14, fontweight="bold")
plt.xticks(rotation=45, ha='right')  # Rotate labels for readability
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()  # Adjust subplot params for a tight layout
plt.show()
```



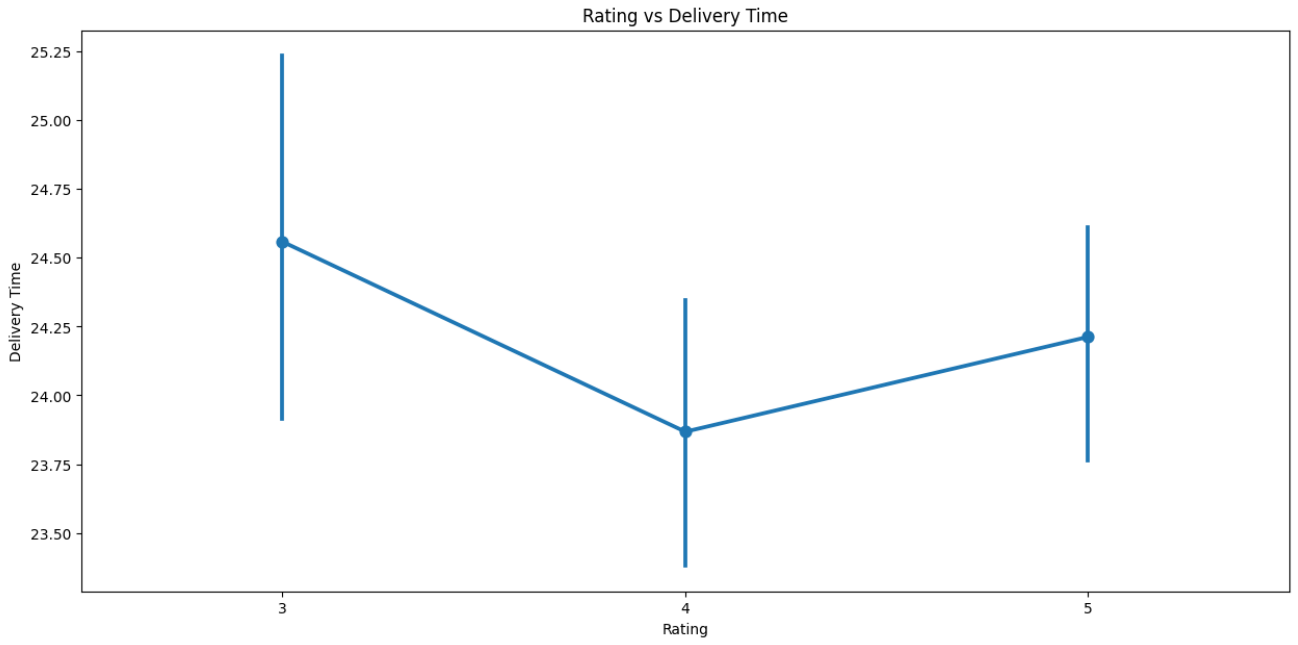
Observations

- Shake Shack's dominance should be leveraged for expansion.
- Mid-tier brands like Blue Ribbon Sushi & The Meatball Shop could grow further with targeted support.
- Low-performing major brands (Five Guys & Chipotle) need strategic improvements to remain competitive.

Rating vs. Delivery Time

```
# Convert 'rating' to integer, replace missing values and 'not given'
df['rating'] = pd.to_numeric(df['rating'], errors='coerce') #Convert to numeric, setting errors='coerce' to handle 'Not given' values
df['rating'] = df['rating'].fillna(0).astype(int,errors='ignore') #fill Nan with 0 and convert to integer

# Relationship between rating and delivery time
plt.figure(figsize=(15, 7))
sns.pointplot(x = 'rating', y = 'delivery_time', data = df)
plt.title("Rating vs Delivery Time")
plt.xlabel("Rating")
plt.ylabel("Delivery Time")
plt.show()
```

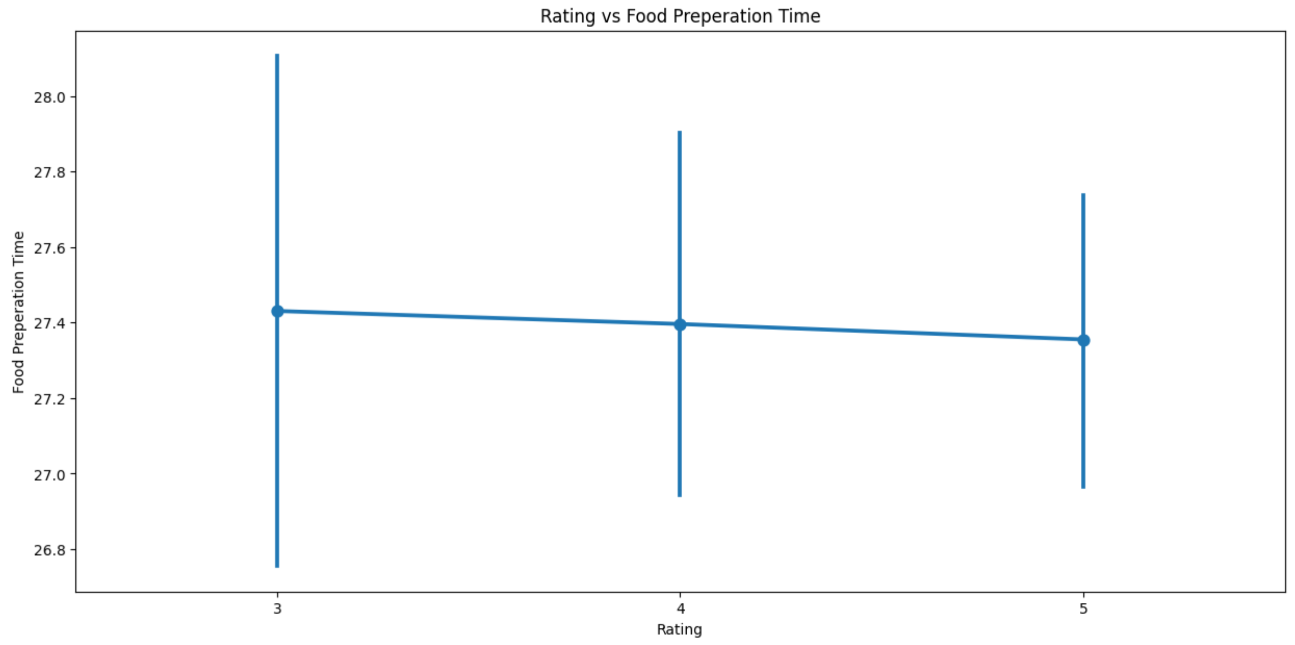


Observation

- Orders with shorter delivery times tend to receive higher ratings, suggesting that customers value speed of service when evaluating their experience.
- However, the highest-rated orders (5-star) do not correspond to the absolute shortest delivery times, indicating that other factors—such as food quality, accuracy, or customer service—also play a role in customer satisfaction.
- Encourage restaurants to focus on factors beyond speed, such as packaging quality, order accuracy, and communication with customers, to improve ratings even when delivery is slightly longer.

Rating vs. Food Prep Time

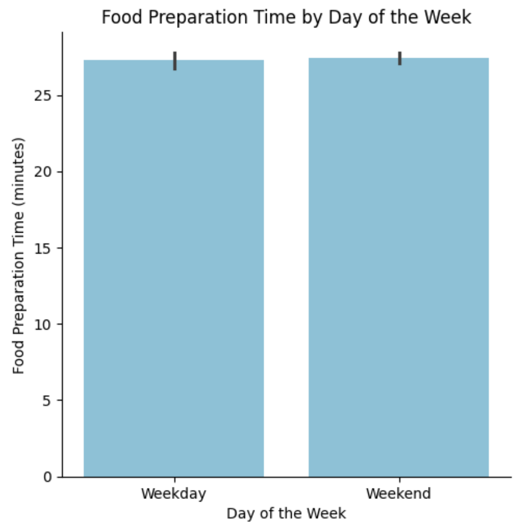
```
# Relationship between rating and food preparation time
plt.figure(figsize=(15, 7))
sns.pointplot(x = 'rating', y = 'food_preparation_time', data = df)
plt.title("Rating vs Food Preperation Time")
plt.xlabel("Rating")
plt.ylabel("Food Preperation Time")
plt.show()
```



Observations

- Food preparation time remains relatively stable across all rating levels (3, 4, and 5 stars), with only a slight decrease in prep time for higher-rated orders.
- The variance in prep time is minimal, indicating that longer preparation times do not significantly impact customer ratings.
- Since prep time does not have a strong correlation with ratings, other factors like food quality, order accuracy, and delivery time likely play a bigger role in customer satisfaction.
- Rather than focusing solely on reducing prep time, FoodHub should encourage restaurants to enhance order accuracy, packaging, and communication to improve customer ratings.

```
sns.catplot(data=df, x='day_of_the_week', y='food_preparation_time', kind='bar', color=ElleSet[3]);
plt.title('Food Preparation Time by Day of the Week');
plt.xlabel('Day of the Week');
plt.ylabel('Food Preparation Time (minutes)');
plt.show()
```



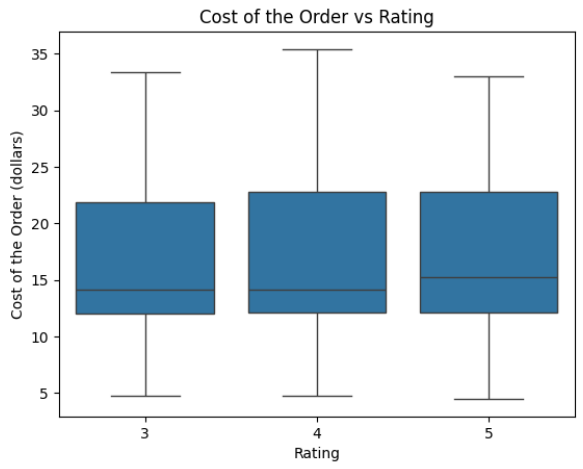
Observations

- Food preparation times are nearly identical on weekdays and weekends, with no significant difference in average prep times.

- The slight variance is minimal, suggesting that kitchen efficiency remains stable regardless of the day.
- Since prep time remains consistent, delays in overall order fulfillment are likely due to external factors such as delivery bottlenecks or order surges on weekends.
- Focus on optimizing delivery logistics during peak weekend hours, rather than improving kitchen speed, to enhance customer experience.

Rating vs. Cost of the Order

```
#  Relationship between rating and cost of the order
sns.boxplot(x='rating', y='cost_of_the_order', data=df)
plt.title('Cost of the Order vs Rating')
plt.xlabel('Rating')
plt.ylabel('Cost of the Order (dollars)')
plt.show()
```

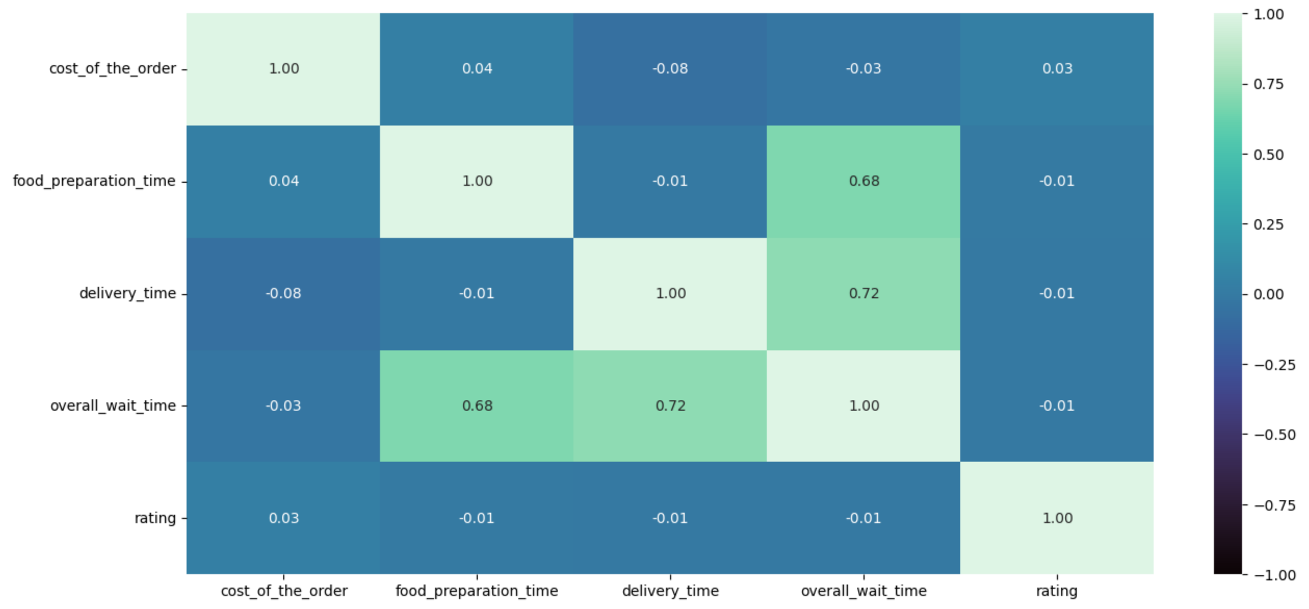


Observations

- The median cost of the order is consistent across ratings.
- Order cost does not significantly impact customer ratings—the median cost remains relatively stable across 3-star, 4-star, and 5-star ratings.
- The distribution of order cost is consistent, suggesting that customers do not rate orders based on price alone.

Correlation among variables

```
# Plot the heatmap
col_list = ['cost_of_the_order', 'food_preparation_time', 'delivery_time', 'overall_wait_time', 'rating']
plt.figure(figsize=(15, 7))
sns.heatmap(df[col_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="mako")
plt.show()
```



Observations

- Overall wait time has a strong positive correlation with both food preparation time (0.68) and delivery time (0.72), confirming that delays in either process contribute to longer overall wait times.
- Cost of the order has little to no correlation with other variables, suggesting that price does not impact delivery speed, wait times, or ratings.
- Customer ratings show no strong correlation with food preparation time, delivery time, or overall wait time, implying that factors beyond timing (such as food quality, order accuracy, and packaging) play a significant role in customer satisfaction.
- Focus on optimizing both food preparation and delivery times to reduce overall wait time, as both contribute significantly to extended waits.
- Improve service elements beyond speed, such as ensuring order accuracy and food quality, since ratings do not strongly correlate with timing metrics.
- Consider testing promotions based on wait time expectations, setting realistic delivery expectations for longer-prep cuisines.

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]

```python
# Filter restaurants with more than 50 ratings
filtered_restaurants = df.groupby('restaurant_name', observed=False).filter(lambda x: len(x) > 50)

# Calculate average rating for each restaurant
restaurant_ratings = filtered_restaurants.groupby('restaurant_name', observed=False)['rating'].mean()

# Filter restaurants with average rating greater than 4
promotional_restaurants = restaurant_ratings[restaurant_ratings > 4].index.tolist()

# Print the promotional restaurants
print("Restaurants for Promotional Offer:")
for restaurant in promotional_restaurants:
    print(restaurant)
```

```
Restaurants for Promotional Offer:
Blue Ribbon Fried Chicken
Blue Ribbon Sushi
Shake Shack
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]

```python
# Determine the revenue
def compute_rev(x):
    if x > 20:
        return x*0.25
    elif x > 5:
        return x*0.15
    else:
        return x*0

df['Net_Revenue'] = df['cost_of_the_order'].apply(compute_rev)
df.head()
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time	overall_wait_time	total_revenue	Net_Revenue
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23	28	51	136.41	1.8345
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25	15	40	1130.59	7.3000
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25	24	49	30.99	1.7385
5	1477224	147468	Tamarind TriBeCa	Indian	25.22	Weekday	3	20	24	44	322.81	6.3050
7	1477859	89574	Barbounia	Mediterranean	5.97	Weekday	3	33	30	63	57.05	0.8955

```python
# Total Net Revenue
total_rev = df['Net_Revenue'].sum()
print('The net revenue is around', round(total_rev, 2), 'dollars')
```

```
The net revenue is around 3865.57 dollars
```

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]

```python
# Calculate total delivery time and add a new column to the dataframe df to store the total delivery time
df['total_time'] = df['food_preparation_time'] + df['delivery_time']

# Filter for orders taking more than 60 minutes
orders_over_60_minutes = df[df['total_time'] > 60]

percentage = (orders_over_60_minutes.shape[0] / df.shape[0]) * 100

print("Percentage of orders taking greater than 60 minutes from time order is placed:", round(percentage, 2), '%')
```

```
Percentage of orders taking greater than 60 minutes from time order is placed: 10.24 %
```

Observations:

- Over 10% of all orders exceed 60 minutes from the time they are placed to delivery, which may negatively impact customer satisfaction.
- Extended wait times may be linked to certain cuisine types, peak hours, or restaurant inefficiencies, requiring further investigation.
- Despite longer wait times, customer ratings do not strongly correlate with wait time, meaning customers may tolerate longer waits if the food quality and service meet expectations.

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]

```python
# Weekday and Weekend Delivery Times
print('The mean delivery time on weekdays is around',
      round(df[df['day_of_the_week'] == 'Weekday']['delivery_time'].mean()),
     'minutes')

print('The mean delivery time on weekends is around',
      round(df[df['day_of_the_week'] == 'Weekend']['delivery_time'].mean()),
     'minutes')
```

```
The mean delivery time on weekdays is around 28 minutes
The mean delivery time on weekends is around 22 minutes
```

Observations:

- Weekday deliveries take somewhat longer (28 minutes) compared to weekends (22 minutes), a 6-minute difference on average.
- The faster weekend delivery times may be due to less traffic congestion, higher driver availability, or fewer simultaneous orders per restaurant.
- The longer weekday delivery times suggest potential bottlenecks during lunch or dinner rush hours, urban congestion, or fewer delivery drivers available.
- Investigate weekday rush-hour patterns to determine whether certain time slots (e.g., lunch or dinner) experience higher delivery delays.
- Encourage restaurants to adjust fulfillment workflows on weekdays, such as prioritizing high-volume time slots with extra kitchen staff. Optimize driver availability on weekdays by offering incentives for working during peak hours to minimize delays.

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:

1. Comfort food, led by American cuisine, dominates revenue.
    - Shake Shack is the clear leader, reinforcing the demand for fast-casual comfort food.
2. Weekends have higher order volume but longer wait times.
    - Despite shorter delivery times, order fulfillment struggles due to peak demand.
3. There is a clear mix of price sensitivity.

A large segment (30%) orders over $20, while others remain cost-conscious.

4. High-rated restaurants are not necessarily the fastest.
   - Delivery time does not strongly impact ratings—food quality, accuracy, and experience play a bigger role.
5. Five Guys & Chipotle underperform despite brand strength.
   - Potential issues with menu appeal, pricing, or delivery efficiency need further analysis.

Recommendations:

1. Expand high-performing brands & optimize Shake Shack's model.
   - Onboard additional Shake Shack locations and apply its best practices to other mid-tier brands.
2. Develop targeted pricing strategies.
   - Introduce value deals for budget-conscious customers and premium offerings for high-spending customers.
3. Improve weekend fulfillment strategies.
   - Increase driver availability and streamline order prep workflows to reduce weekend delays.
4. Enhance restaurant & delivery insights for underperformers.
   - Provide data-backed recommendations to Five Guys & Chipotle to improve revenue performance.
5. Optimize customer experience beyond speed.
   - Encourage high-rated restaurants to improve order accuracy & quality rather than focusing solely on speed.