Project Foundations for Data Science: 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 [1]:

```python
# 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 [2]:

```python
# read the data
df = pd.read_csv('foodhub_order.csv')
# returns the first 5 rows
df.head()
```

Out[2]:

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

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 [3]:
```
# Check the shape of the dataset
df.shape # Fill in the blank
```

Out[3]:
```
(1898, 9)
```

Observations:

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

In [4]:
```
# Checking For Missing Data Types
df.dtypes
```

Out[4]:
```
order_id                   int64
customer_id                int64
restaurant_name           object
cuisine_type              object
cost_of_the_order        float64
day_of_the_week           object
rating                    object
food_preparation_time      int64
delivery_time              int64
dtype: object
```

Observations:

There Are No Missing Values In The Data.

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

In [7]:
```
# Check for missing values
print(df.isnull().sum())

# Treat missing values (if any) - Example using mean imputation
for col in df.columns:
    if df[col].isnull().any():
        mean_val = df[col].mean()
        mydata[col].fillna(mean_val, inplace=True)

# Verify if missing values are handled
print(df.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
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

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 [8]:
```python
# Describing The Data As A Whole.
df.describe()
```

Out[8]:

	order_id	customer_id	cost_of_the_order	food_preparation_time	delivery_time
count	1.898000e+03	1898.000000	1898.000000	1898.000000	1898.000000
mean	1.477496e+06	171168.478398	16.498851	27.371970	24.161749
std	5.480497e+02	113698.139743	7.483812	4.632481	4.972637
min	1.476547e+06	1311.000000	4.470000	20.000000	15.000000
25%	1.477021e+06	77787.750000	12.080000	23.000000	20.000000
50%	1.477496e+06	128600.000000	14.140000	27.000000	25.000000
75%	1.477970e+06	270525.000000	22.297500	31.000000	28.000000
max	1.478444e+06	405334.000000	35.410000	35.000000	33.000000

Observations:

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

In [10]:
```python
# Count the occurrences of each rating
rating_counts = df['rating'].value_counts()

# Display the result
print(rating_counts)
```

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

Observations:

There Are 736 Orders Not Rated

Exploratory Data Analysis (EDA)

Univariate Analysis

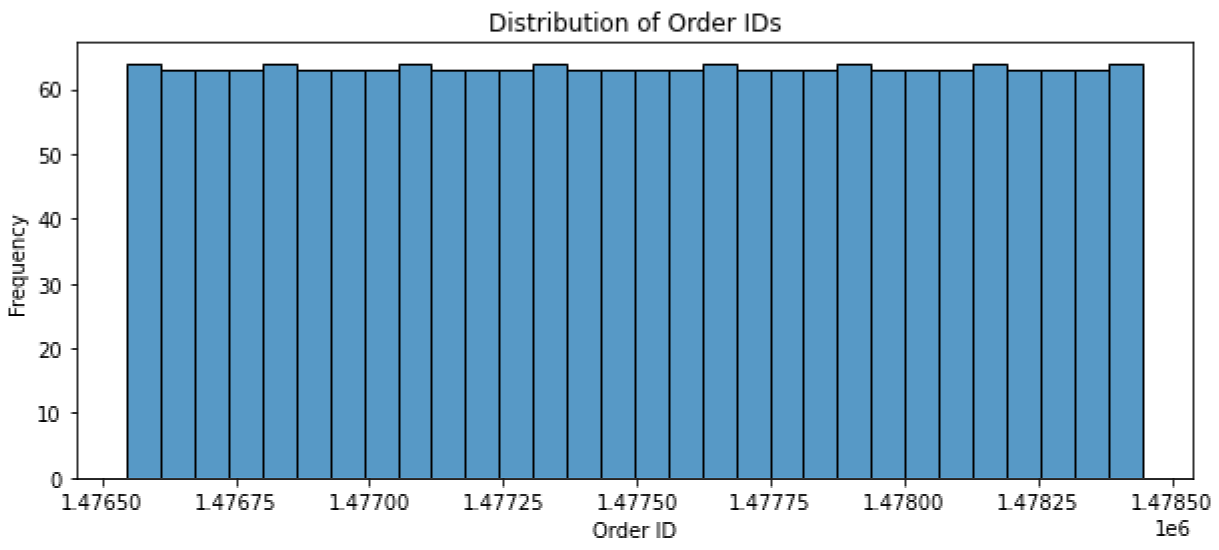
Question 6: Explore all the variables and provide observations on their distributions. (Generally, histograms, boxplots, countplots, etc. are used for univariate exploration.) [9 marks]

Order ID

In [11]:

```
# Check unique order IDs
unique_order_ids = df['order_id'].nunique()
print(f"Number of unique order IDs: {unique_order_ids}")

# Check for duplicate order IDs
duplicate_order_ids = df['order_id'].duplicated().sum()
print(f"Number of duplicate order IDs: {duplicate_order_ids}")

# Plot the frequency of order IDs
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 4))
sns.histplot(df['order_id'], bins=30, kde=False)
plt.title('Distribution of Order IDs')
plt.xlabel('Order ID')
plt.ylabel('Frequency')
plt.show()
```

```
Number of unique order IDs: 1898
Number of duplicate order IDs: 0
```



In [ ]:

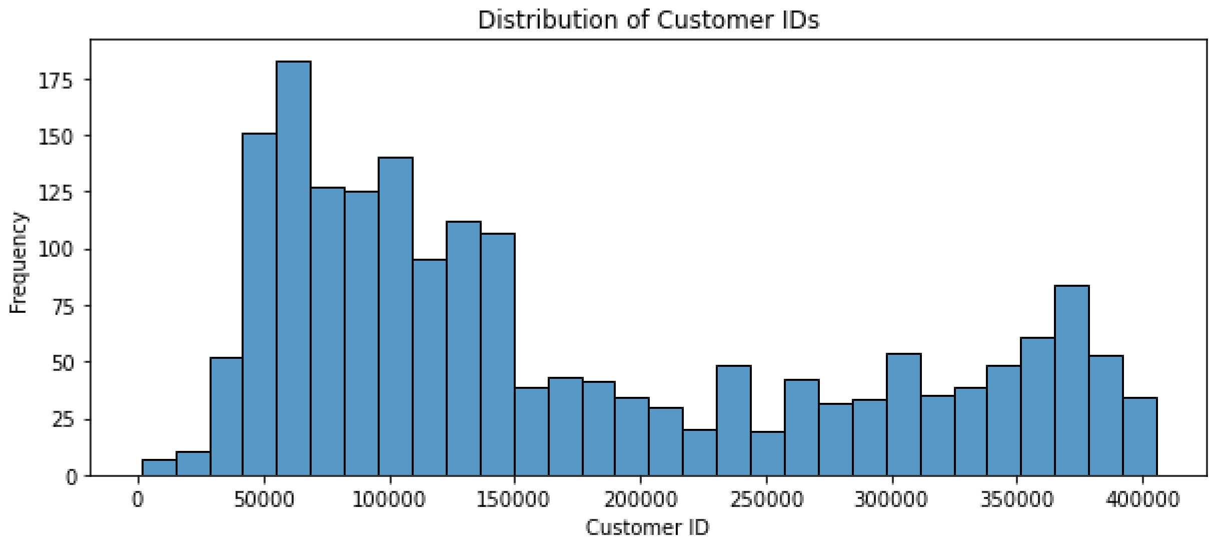
```
Observation:
```

Customer ID

In [12]:

```python
# Check unique customer IDs
unique_customer_ids = df['customer_id'].nunique()
print(f"Number of unique customer IDs: {unique_customer_ids}")

# Check for duplicate customer IDs
duplicate_customer_ids = df['customer_id'].duplicated().sum()
print(f"Number of duplicate customer IDs: {duplicate_customer_ids}")

# Plot the frequency of customer IDs
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 4))
sns.histplot(df['customer_id'], bins=30, kde=False)
plt.title('Distribution of Customer IDs')
plt.xlabel('Customer ID')
plt.ylabel('Frequency')
plt.show()
```

```
Number of unique customer IDs: 1200
Number of duplicate customer IDs: 698
```



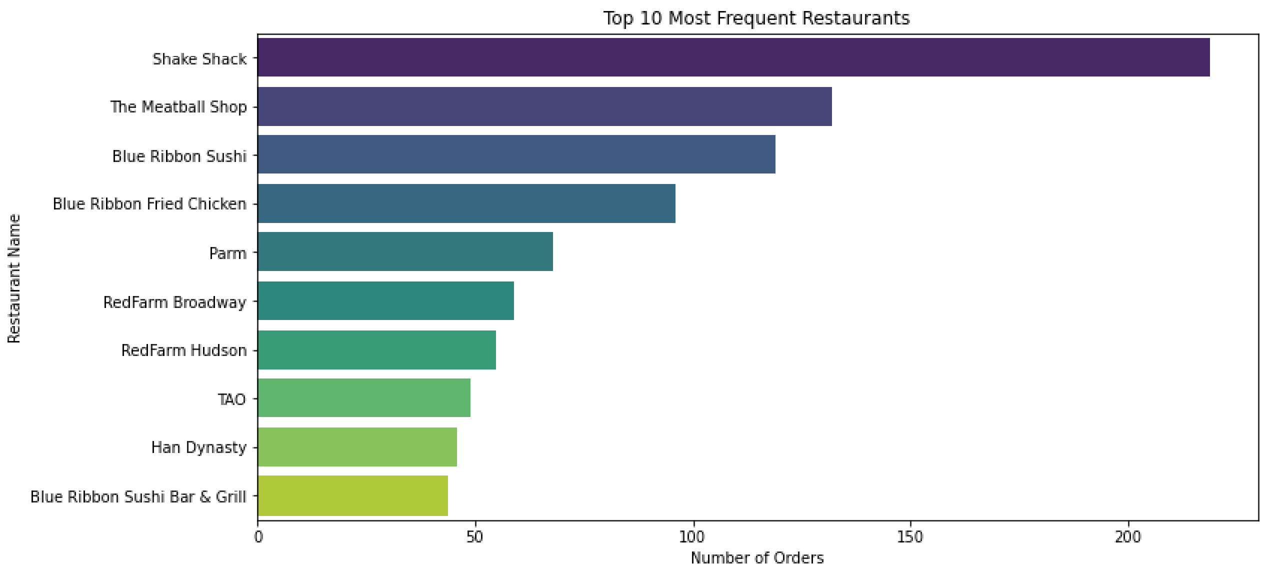
Observations:

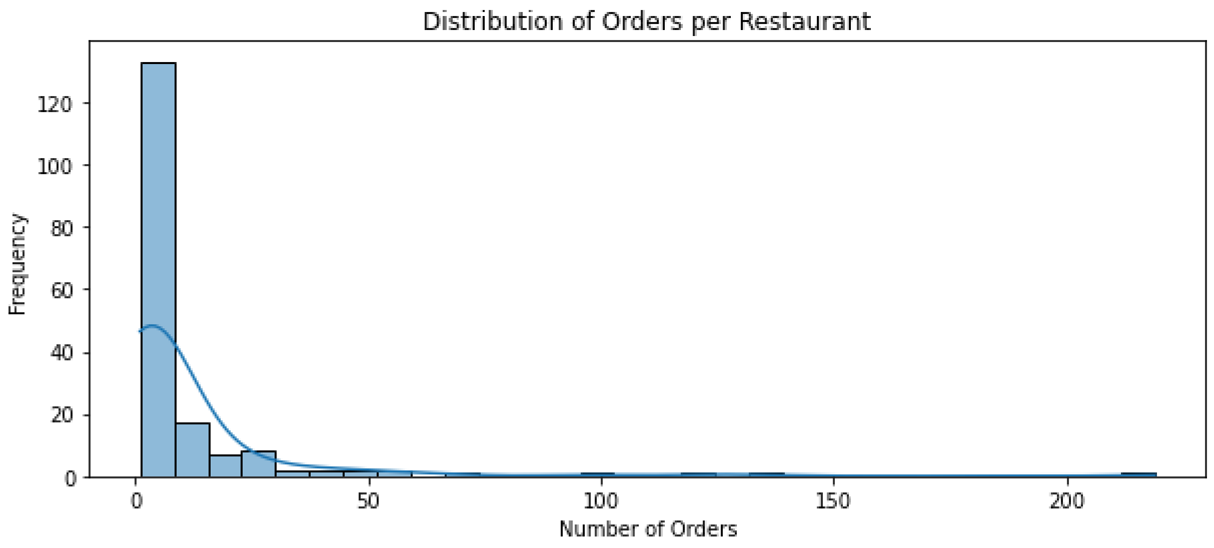
Resturant Name

In [13]:

```python
# Check unique restaurant names
unique_restaurant_names = df['restaurant_name'].nunique()
print(f"Number of unique restaurant names: {unique_restaurant_names}")

# Check the frequency of each restaurant
restaurant_counts = df['restaurant_name'].value_counts()

# Plot the top 10 most frequent restaurants
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))
sns.barplot(y=restaurant_counts.head(10).index, x=restaurant_counts.head(10).values, pa
plt.title('Top 10 Most Frequent Restaurants')
plt.xlabel('Number of Orders')
plt.ylabel('Restaurant Name')
plt.show()

# Plot the distribution of restaurant orders
plt.figure(figsize=(10, 4))
sns.histplot(restaurant_counts, bins=30, kde=True)
plt.title('Distribution of Orders per Restaurant')
plt.xlabel('Number of Orders')
plt.ylabel('Frequency')
plt.show()
```

```
Number of unique restaurant names: 178
```



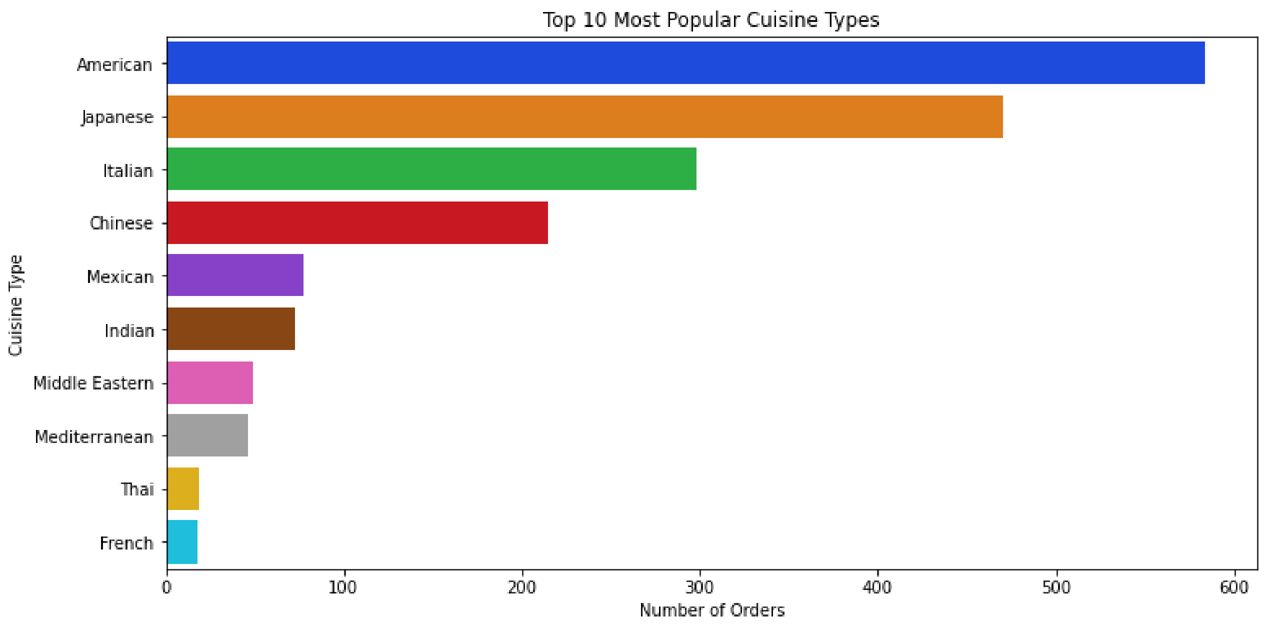


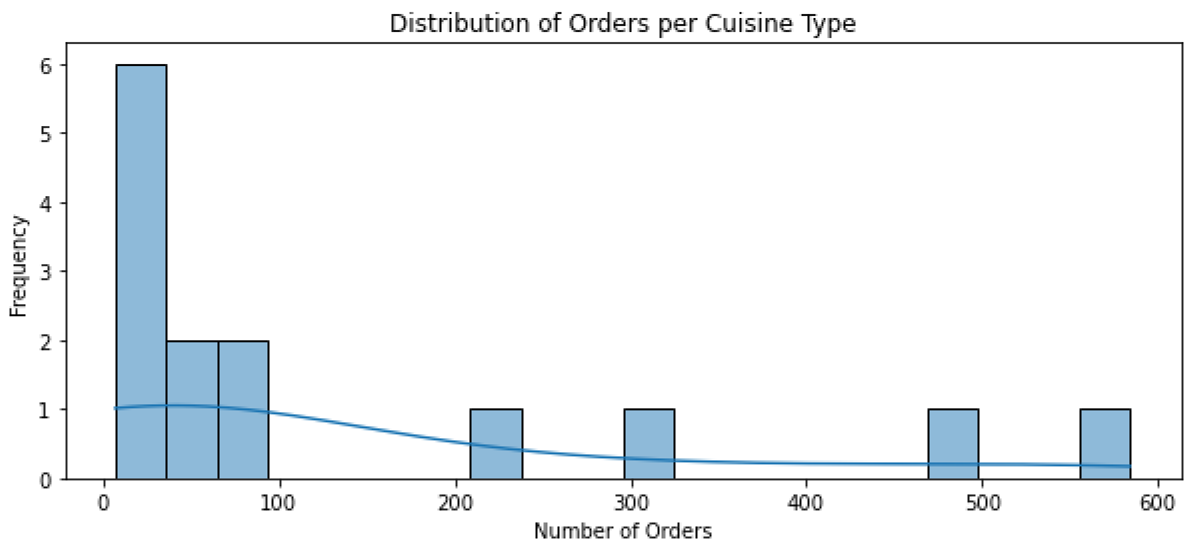
Observations: There are 178 unique restaurants names. Shake Shack, The Meatball Shop and Blue Ribbon Sushi have the most with 119 orders. Used the Histogram to show which resturants receive the fewest/highest orders. The Skewed Distribution showed where resturants have more orders compared to others.

Cuisine

In [15]:
```
# Check unique cuisine types
unique_cuisine_types = df['cuisine_type'].nunique()
print(f"Number of unique cuisine types: {unique_cuisine_types}")

# Check the frequency of each cuisine type
cuisine_counts = df['cuisine_type'].value_counts()

# Plot the top 10 most popular cuisine types
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))
sns.barplot(y=cuisine_counts.head(10).index, x=cuisine_counts.head(10).values, palette=
plt.title('Top 10 Most Popular Cuisine Types')
plt.xlabel('Number of Orders')
plt.ylabel('Cuisine Type')
plt.show()

# Plot the distribution of cuisine types
plt.figure(figsize=(10, 4))
sns.histplot(cuisine_counts, bins=20, kde=True)
plt.title('Distribution of Orders per Cuisine Type')
plt.xlabel('Number of Orders')
plt.ylabel('Frequency')
plt.show()
```

```
Number of unique cuisine types: 14
```





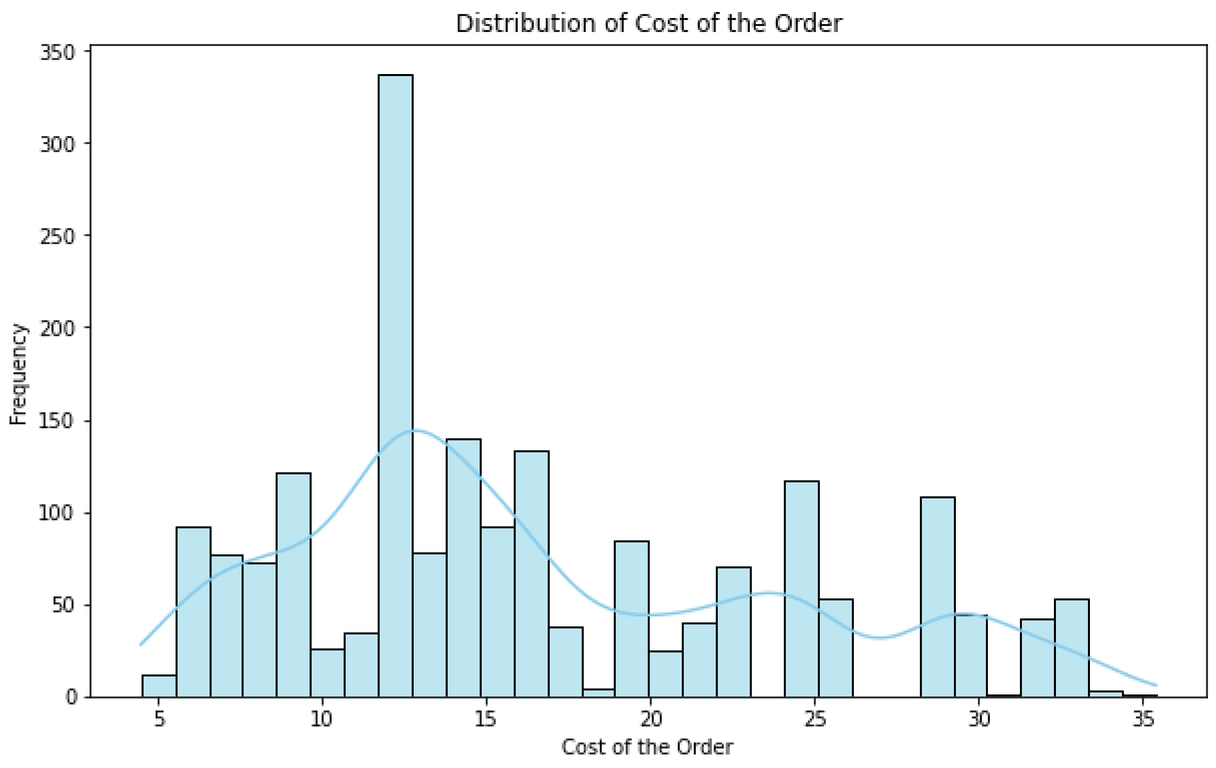
Observation: Bar Plot Shows which type of foods are more popular than most which is American. The histogram shows the distribution where only a few cuisines dominate the number of orders.

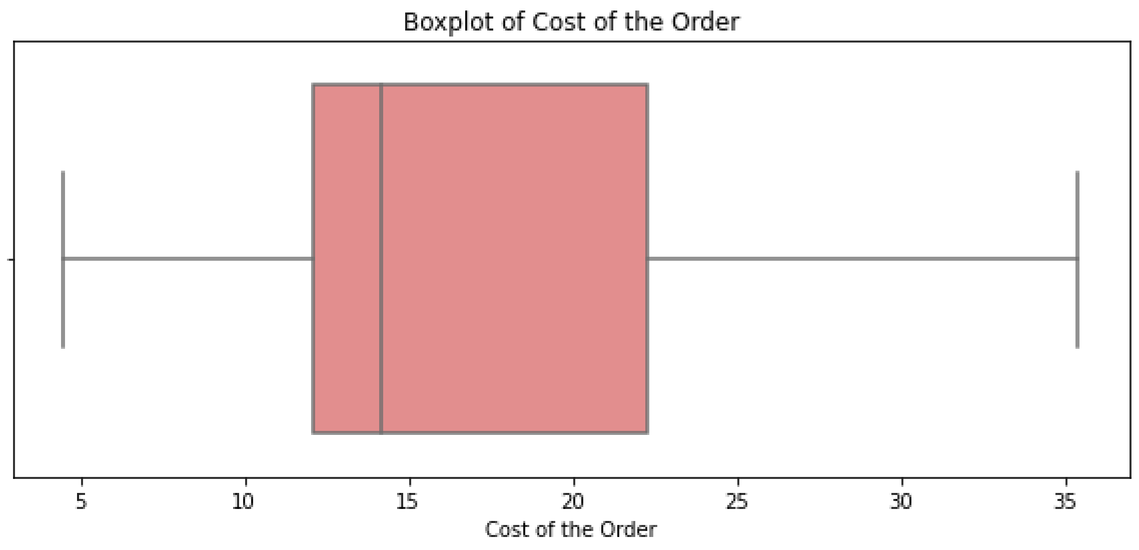
Cost Of The Order

In [16]:

```
# Summary statistics for cost of the order
cost_summary = df['cost_of_the_order'].describe()
print(cost_summary)

# Plotting the distribution of cost of the order
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.histplot(df['cost_of_the_order'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Cost of the Order')
plt.xlabel('Cost of the Order')
plt.ylabel('Frequency')
plt.show()

# Boxplot to identify outliers
plt.figure(figsize=(10, 4))
sns.boxplot(x=df['cost_of_the_order'], color='lightcoral')
plt.title('Boxplot of Cost of the Order')
plt.xlabel('Cost of the Order')
plt.show()
```

```
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
```





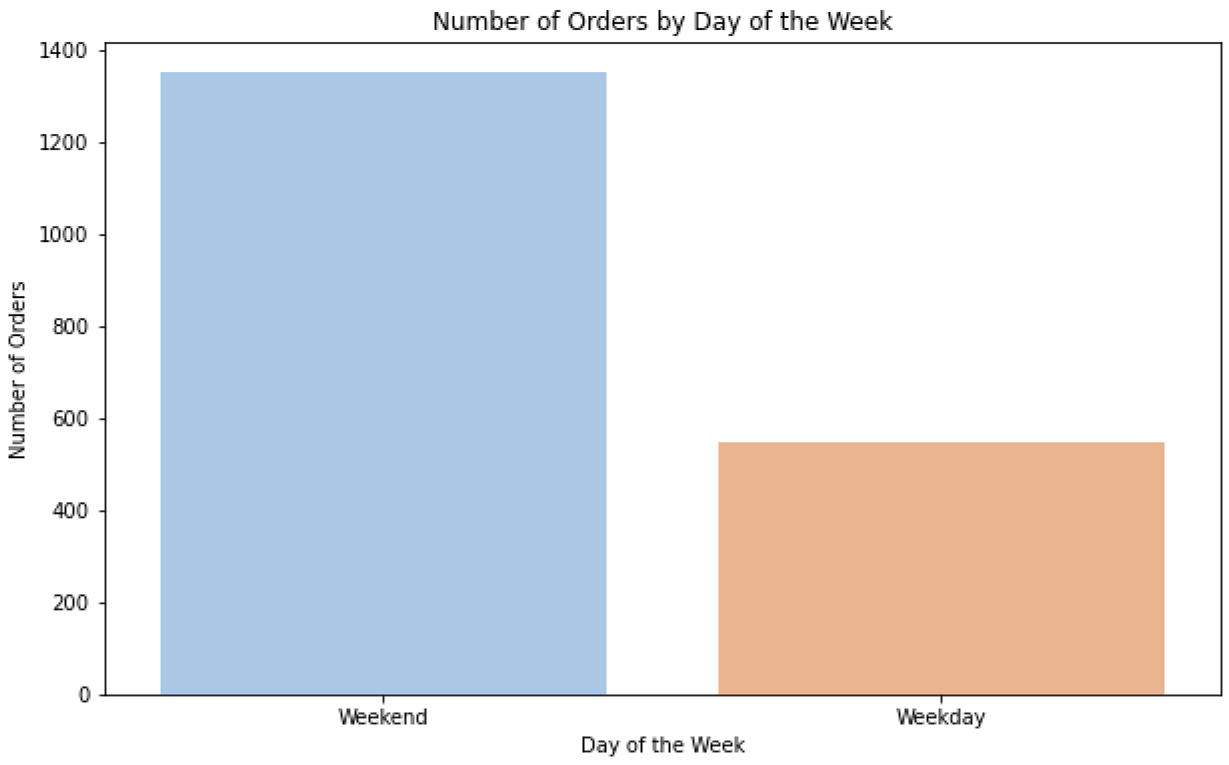
Observations: The histogram shows the distribution of the cost of orders from certain restaurants to others.
The boxplot shows the percentiles and median cost of orders for all restaurants. The code df.desribe() gives the information the boxplot is showing.

Day Of The Week

In [17]:
```
# Check unique days of the week
unique_days = df['day_of_the_week'].nunique()
print(f"Number of unique day categories: {unique_days}")

# Check the frequency of orders for each day
day_counts = df['day_of_the_week'].value_counts()
```

```
# Plot the frequency of orders by day of the week
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x=day_counts.index, y=day_counts.values, palette='pastel')
plt.title('Number of Orders by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.show()
```

```
Number of unique day categories: 2
```



In [18]:

```
# Check the frequency of orders for each day
day_counts = df['day_of_the_week'].value_counts()

# Display the counts
print(day_counts)
```

```
Weekend    1351
Weekday     547
Name: day_of_the_week, dtype: int64
```

The Weekend is typically where most customers go make orders from these resturants. The Weekend is around 1350 orders and weekday being around 500 orders.

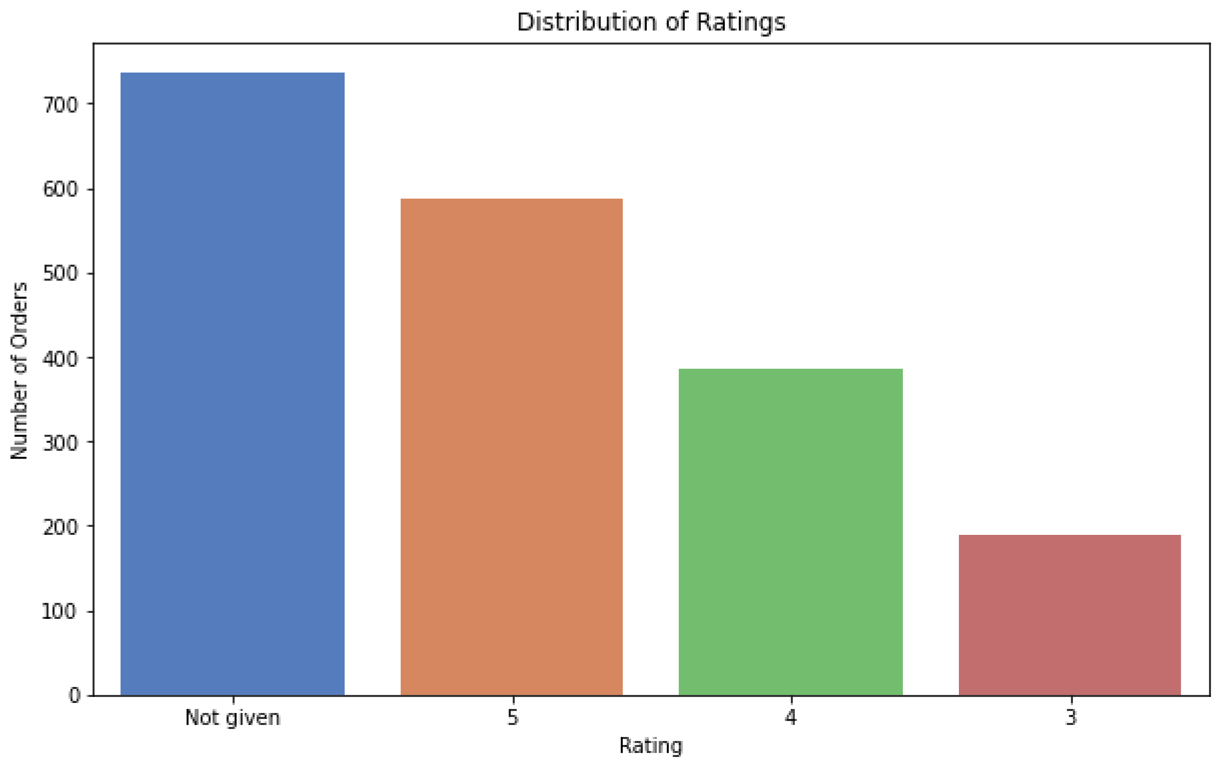
Rating

In [19]:

```
# Check unique ratings
unique_ratings = df['rating'].nunique()
print(f"Number of unique ratings: {unique_ratings}")

# Check the frequency of each rating
rating_counts = df['rating'].value_counts()
```

```
# Plot the frequency of ratings
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x=rating_counts.index, y=rating_counts.values, palette='muted')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Number of Orders')
plt.show()
```

```
Number of unique ratings: 4
```



In [20]:

```
# Check the frequency of orders for each day
ratings_counts = df['rating'].value_counts()

# Display the counts
print(ratings_counts)
```

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

Observation: Many people don't leave a rating compared to rating stores a 3,4, of 5.

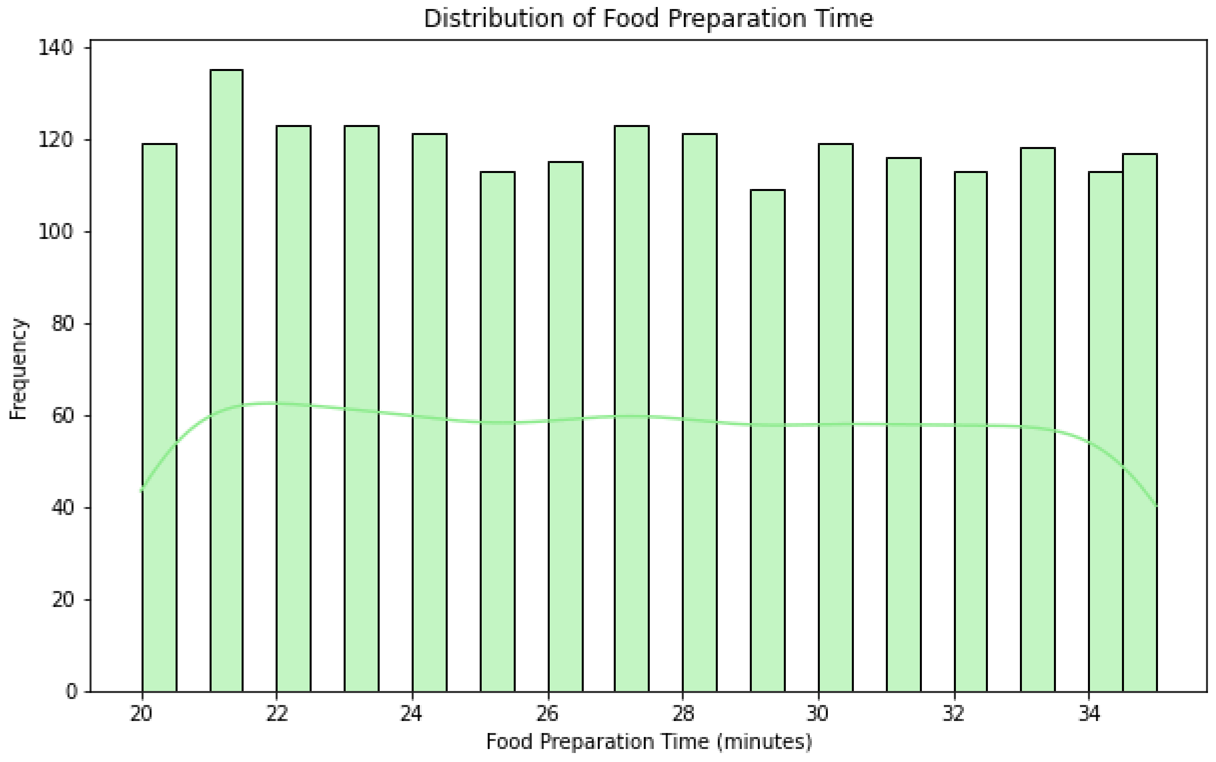
Food Preparation Time

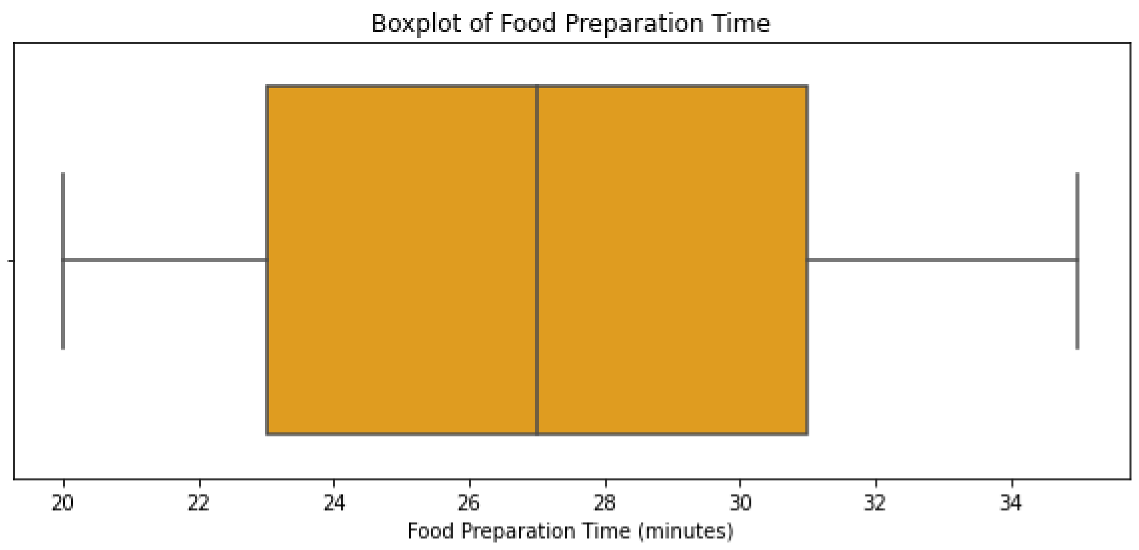
In [21]:

```
# Summary statistics for food preparation time
prep_time_summary = df['food_preparation_time'].describe()
print(prep_time_summary)
```

```python
# Plotting the distribution of food preparation time
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.histplot(df['food_preparation_time'], bins=30, kde=True, color='lightgreen')
plt.title('Distribution of Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Frequency')
plt.show()

# Boxplot to identify outliers
plt.figure(figsize=(10, 4))
sns.boxplot(x=df['food_preparation_time'], color='orange')
plt.title('Boxplot of Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.show()
```

```
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
```





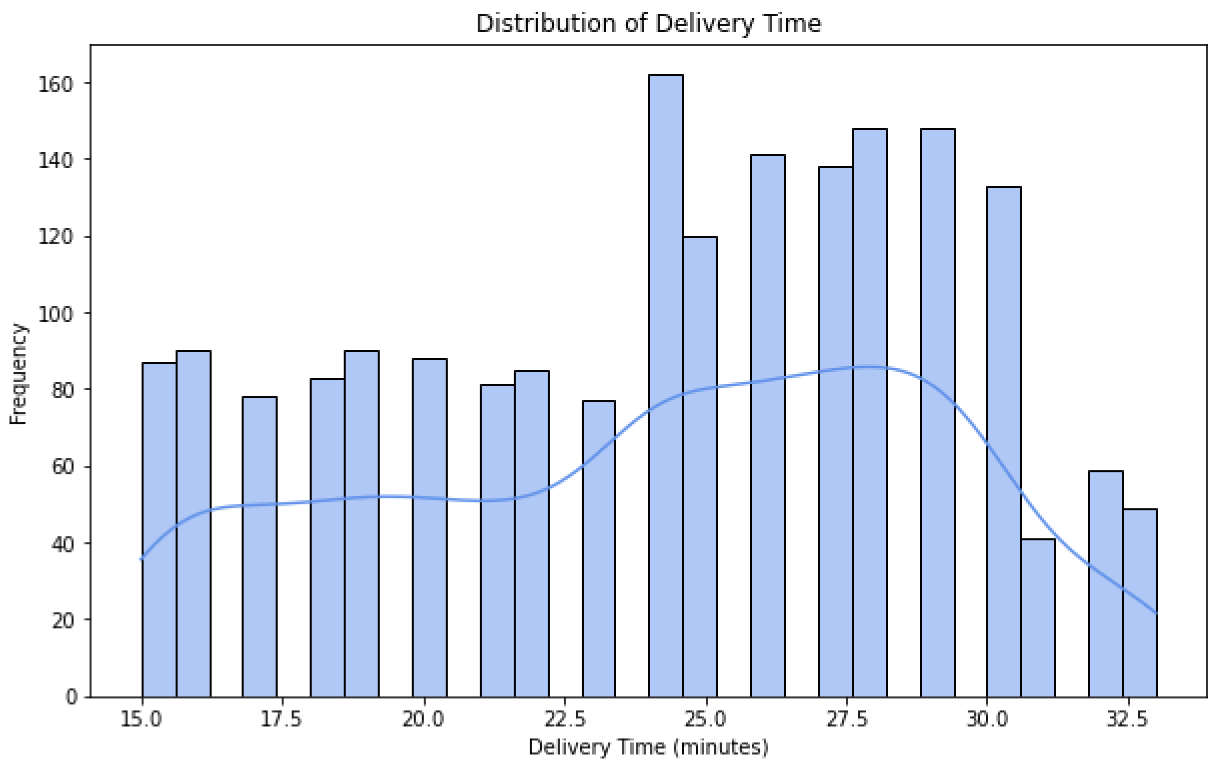
Observation: Most foods are prepared between 25 to 30 minutes with 27 being peak. Boxplot spreads the data with the median being at 27

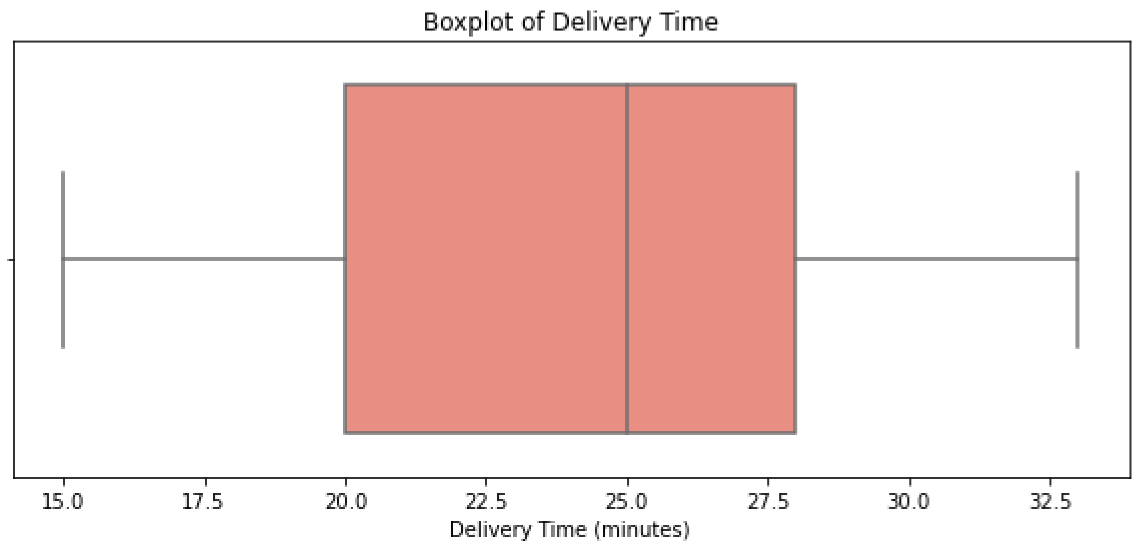
Delivery Time

In [22]:

```
# Summary statistics for delivery time
delivery_time_summary = df['delivery_time'].describe()
print(delivery_time_summary)

# Plotting the distribution of delivery time
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.histplot(df['delivery_time'], bins=30, kde=True, color='cornflowerblue')
plt.title('Distribution of Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Frequency')
plt.show()

# Boxplot to identify outliers
plt.figure(figsize=(10, 4))
sns.boxplot(x=df['delivery_time'], color='salmon')
plt.title('Boxplot of Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.show()
```

```
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
```





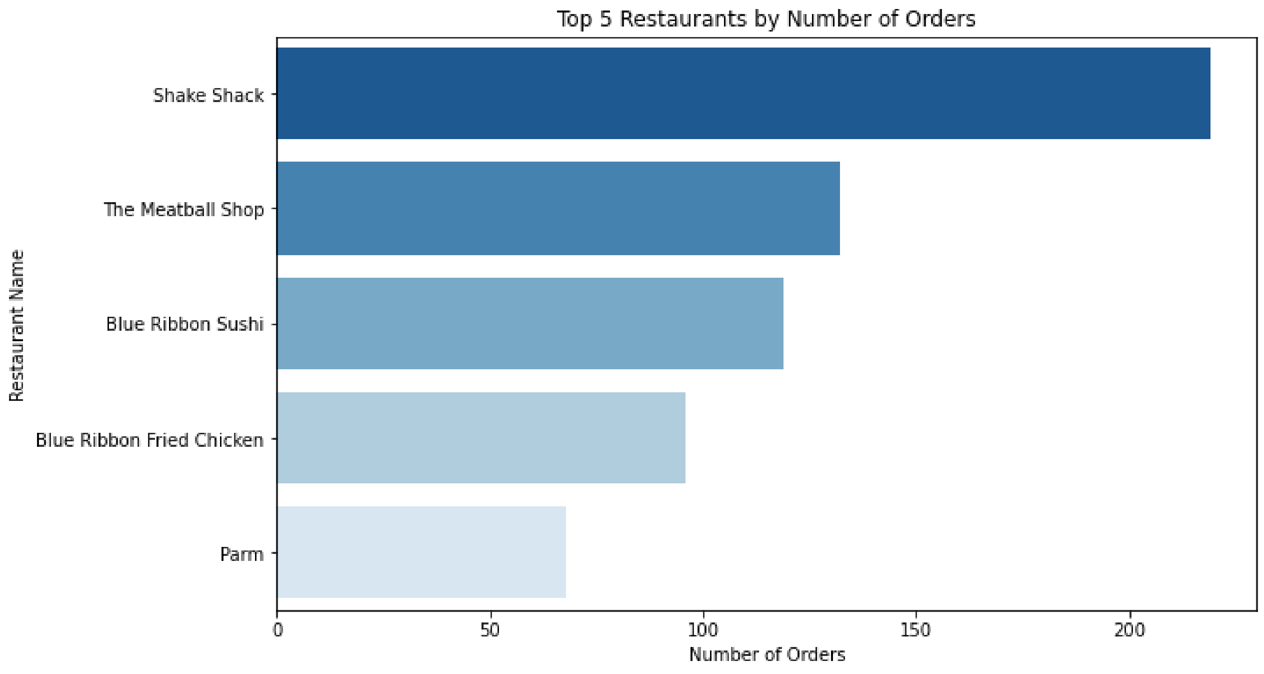
Delivery times are between 20 mintues to 27.5 with 25 being the most common. Boxplot shows the median being at 25.

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

In [23]:
```
# Top 5 restaurants based on the number of orders received
top_5_restaurants = df['restaurant_name'].value_counts().head(5)

# Display the top 5 restaurants
print(top_5_restaurants)

# Plotting the top 5 restaurants
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.figure(figsize=(10, 6))
sns.barplot(y=top_5_restaurants.index, x=top_5_restaurants.values, palette='Blues_r')
plt.title('Top 5 Restaurants by Number of Orders')
plt.xlabel('Number of Orders')
plt.ylabel('Restaurant Name')
plt.show()
```

```
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
Name: restaurant_name, dtype: int64
```



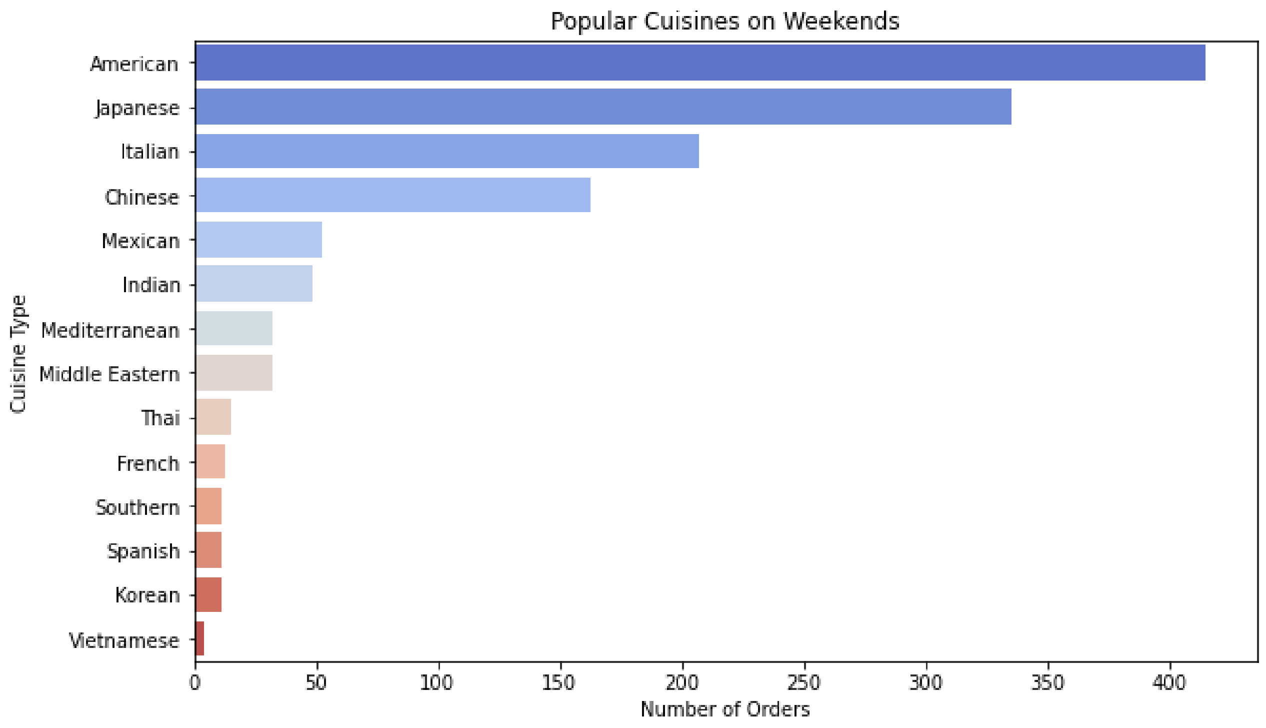
Observations:

Shake Shack, The Meatball Shop, Blue Ribbon Sushi, Blue Ribbon Fried Chicken and Parm are the top 5.

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

```
In [24]:
# Filter data for weekends
weekend_data = df[df['day_of_the_week'] == 'Weekend']

# Find the most popular cuisine on weekends
most_popular_cuisine_weekend = weekend_data['cuisine_type'].value_counts().idxmax()
most_popular_cuisine_count = weekend_data['cuisine_type'].value_counts().max()

# Display the result
print(f"The most popular cuisine on weekends is {most_popular_cuisine_weekend} with {mo

# Plotting the top cuisines on weekends
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(y=weekend_data['cuisine_type'].value_counts().index,
            x=weekend_data['cuisine_type'].value_counts().values,
            palette='coolwarm')

plt.title('Popular Cuisines on Weekends')
plt.xlabel('Number of Orders')
plt.ylabel('Cuisine Type')
plt.show()
```

```
The most popular cuisine on weekends is American with 415 orders.
```



Observations:

American

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

In [26]:
```
# Calculate the number of orders that cost more than $20
orders_above_20 = df[df['cost_of_the_order'] > 20].shape[0]

# Calculate the total number of orders
total_orders = df.shape[0]

# Calculate the percentage
percentage_above_20 = (orders_above_20 / total_orders) * 100

# Display the result
print(f"Percentage of orders costing more than $20: {percentage_above_20:.2f}%")
```

```
Percentage of orders costing more than $20: 29.24%
```

Observations:

About 29% of orders cost more than $20

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

In [27]:

```
# Calculate the mean order delivery time
mean_delivery_time = df['delivery_time'].mean()

# Display the result
print(f"Mean Order Delivery Time: {mean_delivery_time:.2f} minutes")
```

```
Mean Order Delivery Time: 24.16 minutes
```

Observations:

Average delivery time is about 24 minutes

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 [28]:

```
# Find the top 3 most frequent customers
top_3_customers = df['customer_id'].value_counts().head(3)

# Display the result
print(top_3_customers)
```

```
52832    13
47440    10
83287     9
Name: customer_id, dtype: int64
```

Observation

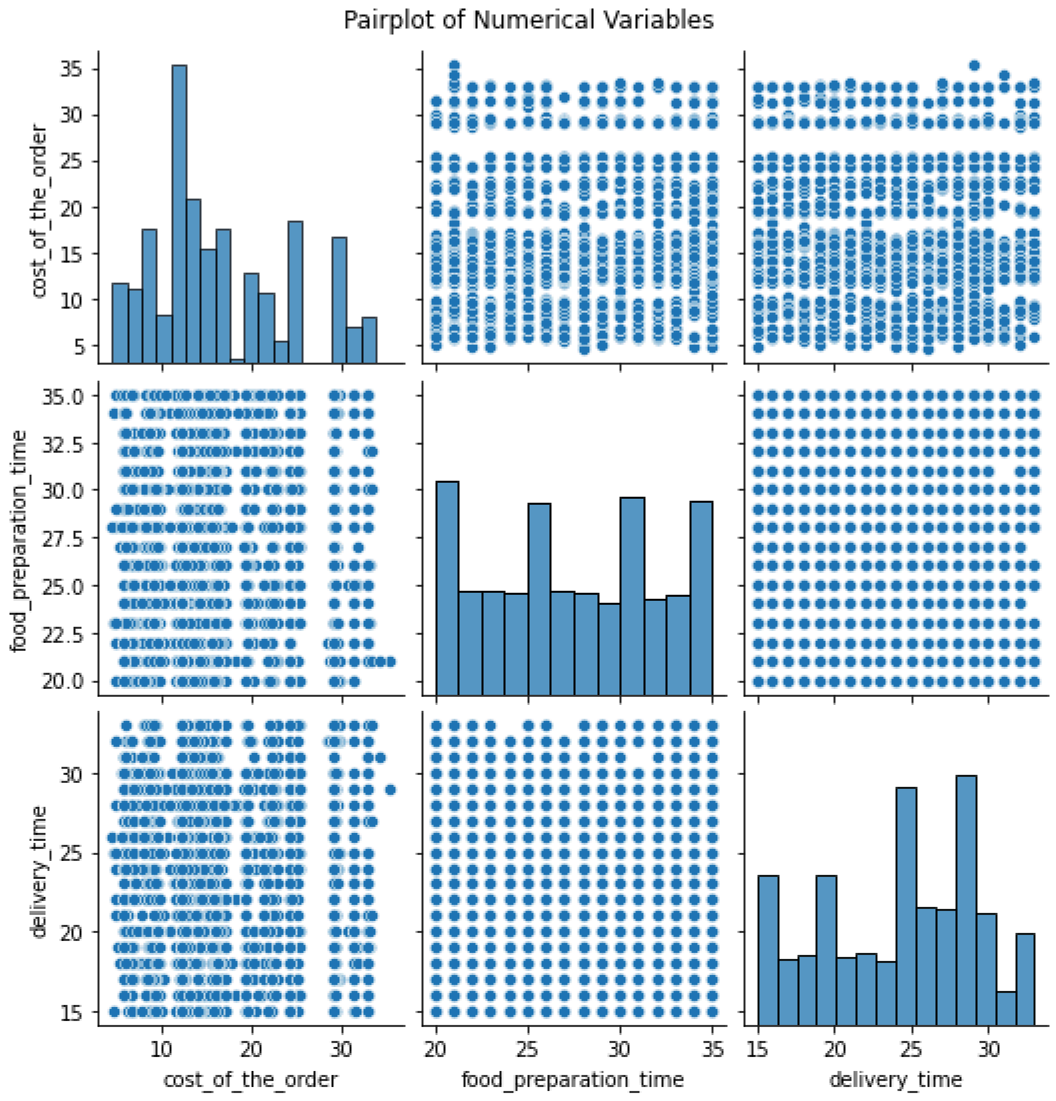
These are the top three customers and the amount of times they ordered

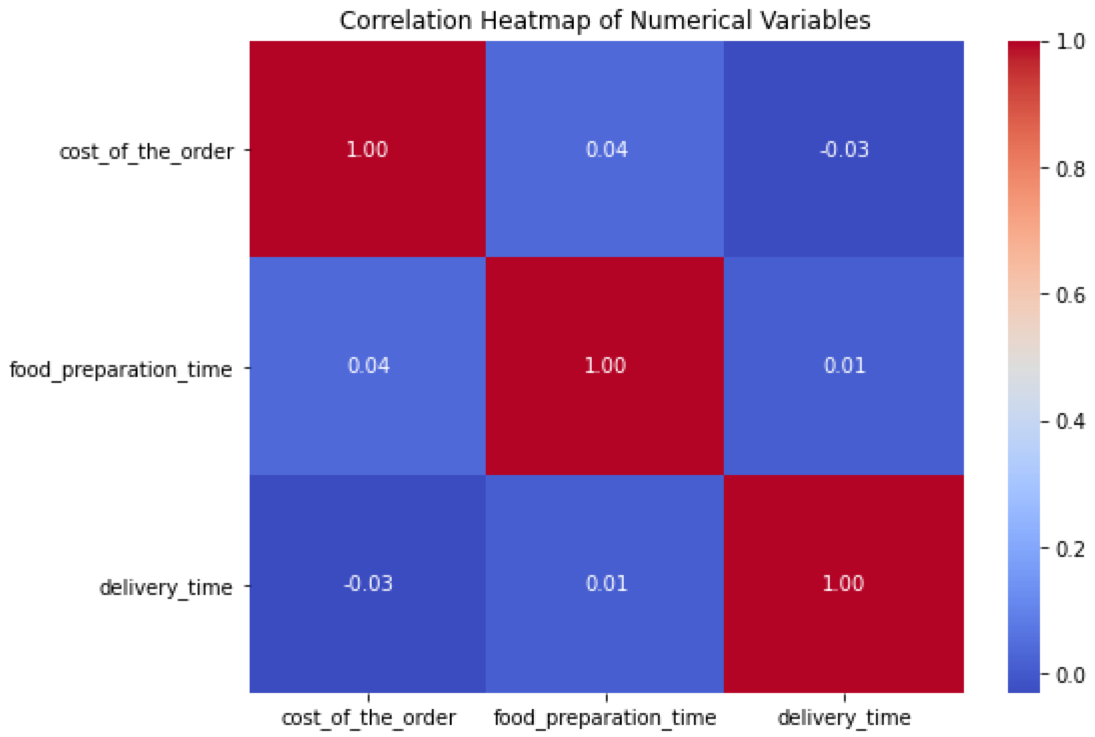
Multivariate Analysis

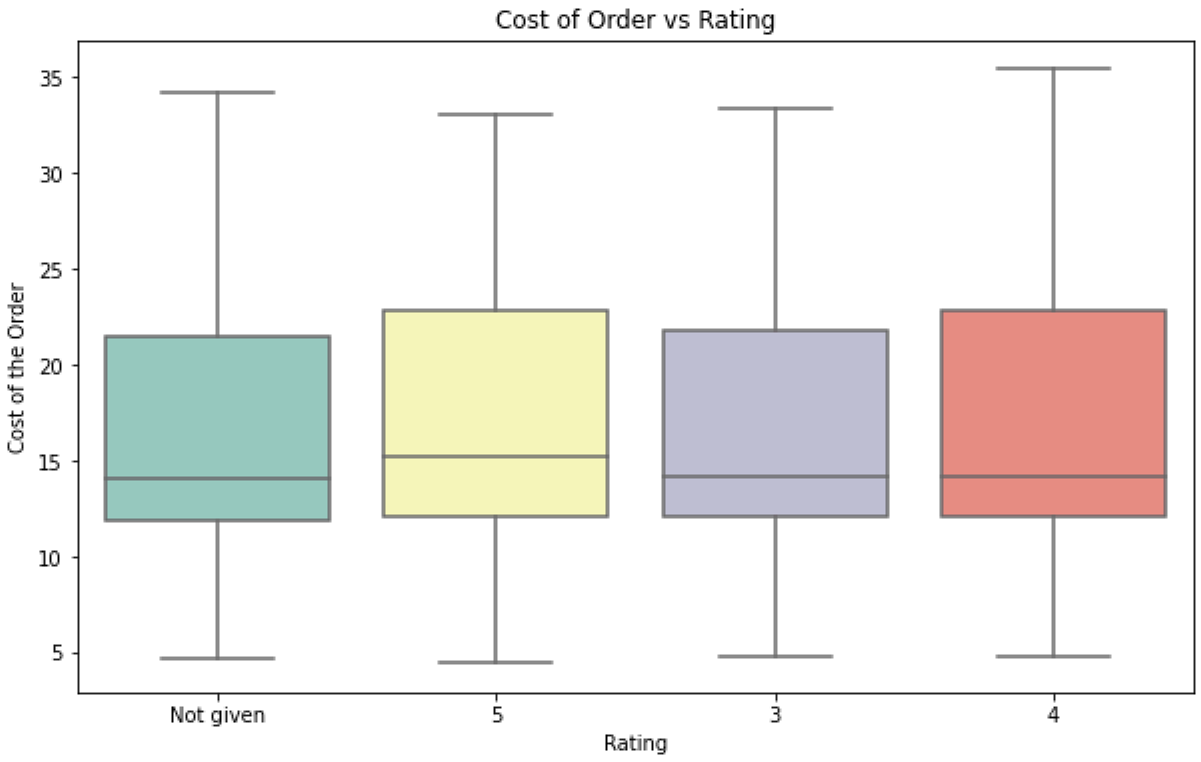
Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. (It is a good idea to explore relations between numerical variables as well as relations between numerical and categorical variables) [10 marks]

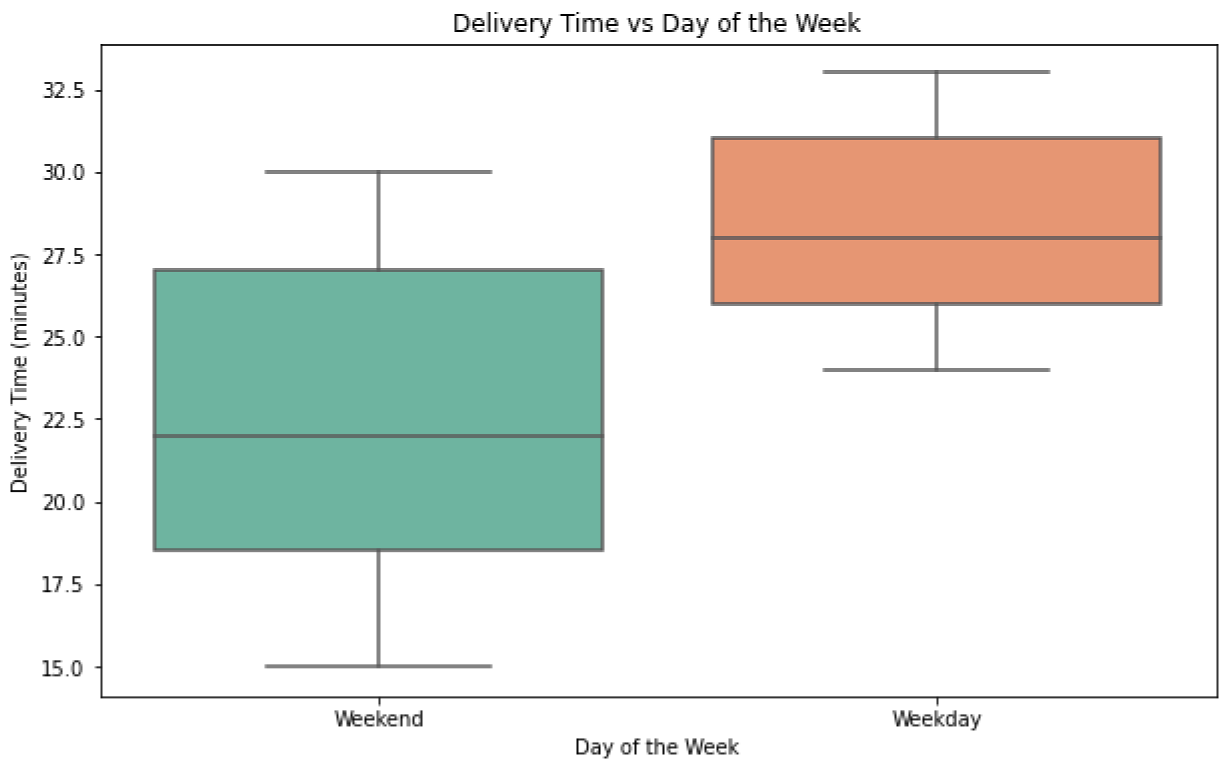
In [29]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Pairplot to explore relationships between numerical variables
sns.pairplot(df[['cost_of_the_order', 'food_preparation_time', 'delivery_time']], diag_
plt.suptitle('Pairplot of Numerical Variables', y=1.02)
plt.show()

# Heatmap to visualize correlation between numerical variables
plt.figure(figsize=(8, 6))
correlation_matrix = df[['cost_of_the_order', 'food_preparation_time', 'delivery_time']
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Variables')
plt.show()

# Boxplot to explore the relationship between cost and rating
plt.figure(figsize=(10, 6))
sns.boxplot(x='rating', y='cost_of_the_order', data=df, palette='Set3')
plt.title('Cost of Order vs Rating')
plt.xlabel('Rating')
plt.ylabel('Cost of the Order')
plt.show()

# Boxplot to explore the relationship between day of the week and delivery time
plt.figure(figsize=(10, 6))
sns.boxplot(x='day_of_the_week', y='delivery_time', data=df, palette='Set2')
plt.title('Delivery Time vs Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Delivery Time (minutes)')
plt.show()
```




Correlation Heatmap of Numerical Variables
cost_of_the_order
food_preparation_time
delivery_time
1.00
0.04
-0.03
0.04
1.00
0.01
-0.03
0.01
1.00
cost_of_the_order
food_preparation_time
delivery_time
1.0
0.8
0.6
0.4
0.2
0.0


Cost of Order vs Rating
Cost of the Order
35
30
25
20
15
10
5
Not given
5
3
4
Rating



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 [32]:

```
# Fixing the SettingWithCopyWarning by using .copy()
df_filtered = df[df['rating'] != 'Not given'].copy()

# Converting rating to numeric
df_filtered.loc[:, 'rating'] = pd.to_numeric(df_filtered['rating'])

# Group by restaurant to get rating count and average rating
restaurant_ratings = df_filtered.groupby('restaurant_name')['rating'].agg(['count', 'me

# Apply the promotional offer conditions
eligible_restaurants = restaurant_ratings[(restaurant_ratings['count'] > 50) & (restaur

# Display eligible restaurants
print(eligible_restaurants)
```

```
                           count      mean
restaurant_name
Blue Ribbon Fried Chicken     64  4.328125
Blue Ribbon Sushi             73  4.219178
Shake Shack                  133  4.278195
The Meatball Shop             84  4.511905
```

Observations:

Four restaurants are eligible for the promotion offer. Count shows the average rating and the amount of ratings that restaurants have received. Shake Shack has the most ratings, while The Meatball Shop has the highest average.

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 [33]:

```
# Calculate the revenue from orders costing more than $20 (25% charge)
revenue_above_20 = df[df['cost_of_the_order'] > 20]['cost_of_the_order'].apply(lambda x

# Calculate the revenue from orders costing more than $5 but less than or equal to $20
revenue_between_5_and_20 = df[(df['cost_of_the_order'] > 5) & (df['cost_of_the_order']

# Total revenue
net_revenue = revenue_above_20 + revenue_between_5_and_20

# Display the net revenue
print(f"Net Revenue Generated by the Company: ${net_revenue:.2f}")
```

```
Net Revenue Generated by the Company: $6166.30
```

Observations:

The total revenue from both fees was $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 [34]:

```
# Calculate the total time for each order (preparation time + delivery time)
df['total_delivery_time'] = df['food_preparation_time'] + df['delivery_time']

# Calculate the number of orders that took more than 60 minutes
orders_above_60 = df[df['total_delivery_time'] > 60].shape[0]

# Calculate the total number of orders
total_orders = df.shape[0]

# Calculate the percentage of orders taking more than 60 minutes
percentage_above_60 = (orders_above_60 / total_orders) * 100

# Display the result
print(f"Percentage of orders taking more than 60 minutes: {percentage_above_60:.2f}%")
```

```
Percentage of orders taking more than 60 minutes: 10.54%
```

Observations:

Code shows that 10% of orders take over an hour

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 [35]:
```
# Calculate the mean delivery time for weekdays and weekends
mean_delivery_time_by_day = df.groupby('day_of_the_week')['delivery_time'].mean()

# Display the result
print(mean_delivery_time_by_day)
```

```
day_of_the_week
Weekday    28.340037
Weekend    22.470022
Name: delivery_time, dtype: float64
```

Observations:

It takes 28 minutes to deliver an order during any weekday. It takes 22 minutes to deliver an order during the weekend.

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:

Weekends are the most common time when customers order. American Cuisine are the most common. Delivery orders are on average 28 mintues furing weekdays but faster on weekend at 22 minutes(estimated). Most Customers Don't Provide Feedback based on the data with 736.

Recommendations:

Try getting more customer feedback due to majority no rating. Though 10 percent of orders are over an hour. Trying brining that time down which could have an impact on customer satisfactory. Revenues is goind good with 29.24% costing more than $20. Hire More Drivers to lower delivery time during weekdays and weekends.
