s-project-learnernotebook-fullcode

July 10, 2023

1 Project Foundations for Data Science: FoodHub Data Analysis

Marks: 40

1.0.1 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.

1.0.2 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.

1.0.3 Data Description

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

1.0.4 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

1.0.5 Let us start by importing the required libraries

[ ]:
```
# 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
```

1.0.6 Understanding the structure of the data

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

[ ]:
```
# read the data
df = pd.read_csv('foodhub_order.csv')
# returns 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

```
   delivery_time
0             20
1             23
2             28
3             15
4             24
```

[ ]:
```
from google.colab import files
import pandas as pd
```

[ ]:
```
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving foodhub_order.csv to foodhub_order.csv
```

[ ]:
```
file_name = list(uploaded.keys())[0]
```

[ ]:
```
# Reading the data from CSV fille
df = pd.read_csv(file_name)
```

[ ]:
```
df.head()
```

[ ]:
```
   order_id  customer_id           restaurant_name cuisine_type  \
0   1477147       337525                   Hangawi       Korean
1   1477685       358141  Blue Ribbon Sushi Izakaya     Japanese
2   1477070        66393               Cafe Habana      Mexican
3   1477334       106968  Blue Ribbon Fried Chicken     American
4   1478249        76942          Dirty Bird to Go     American

   cost_of_the_order day_of_the_week     rating  food_preparation_time  \
0              30.75         Weekend  Not given                     25
1              12.08         Weekend  Not given                     25
2              12.23         Weekday          5                     23
3              29.20         Weekend          3                     25
4              11.59         Weekday          4                     25

   delivery_time
0             20
1             23
2             28
3             15
4             24
```

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

1.0.7 Question 1: How many rows and columns are present in the data?

[ ]:
```
# Write your code here
# This code checks the numbers of rows and columns
num_rows, num_cols = df.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_cols)
```

```
Number of rows: 1898
Number of columns: 9
```

Observations: The dataset contains a total of 1898 rows. The dataset consists of 9 columns.

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

[ ]:
```
# Use info() to print a concise summary of the DataFrame
# This code displays the data types of the columns
print(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
None
```

Observations: The different types of data the ‘oder_id and ’customer_id’ are numbers(integers). The’restaurant_name’and ’cuisine_type’contain text(strings). The’cost_of_the_order’is anumber with decimal points(float) The’day_of_the_week’and’rating’are also text(strings) The’food_preparation_time’and’delivery_time’are numbers(integars)

1.0.9 Question 3: Are there any missing values in the data? If yes, treat them using an appropriate method

[ ]:
```
missing_values = df.isnull().sum()
print(missing_values)
```

```
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
```

There are no missing values in the dataset.

1.0.10 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?

[ ]:
```
# Write your code here
summary = df['food_preparation_time'].describe()
minimum_time = summary['min']
average_time = summary['mean']
maximum_time = summary['max']
print("Minimum preparation time:", minimum_time)
print("average prepartion time:", average_time)
print("Maximum preparation time:", maximum_time)
```

```
Minimum preparation time: 20.0
average prepartion time: 27.371970495258168
Maximum preparation time: 35.0
```

Observations: The minimum time it takes for food to be prepared once an order is placed is 20.0 units. On average, the food preparation time is approximately 27.37 units.The maximum time is takes for foood to be prepared onnce an order is placed is 35.o units.

1.0.11 Question 5: How many orders are not rated?

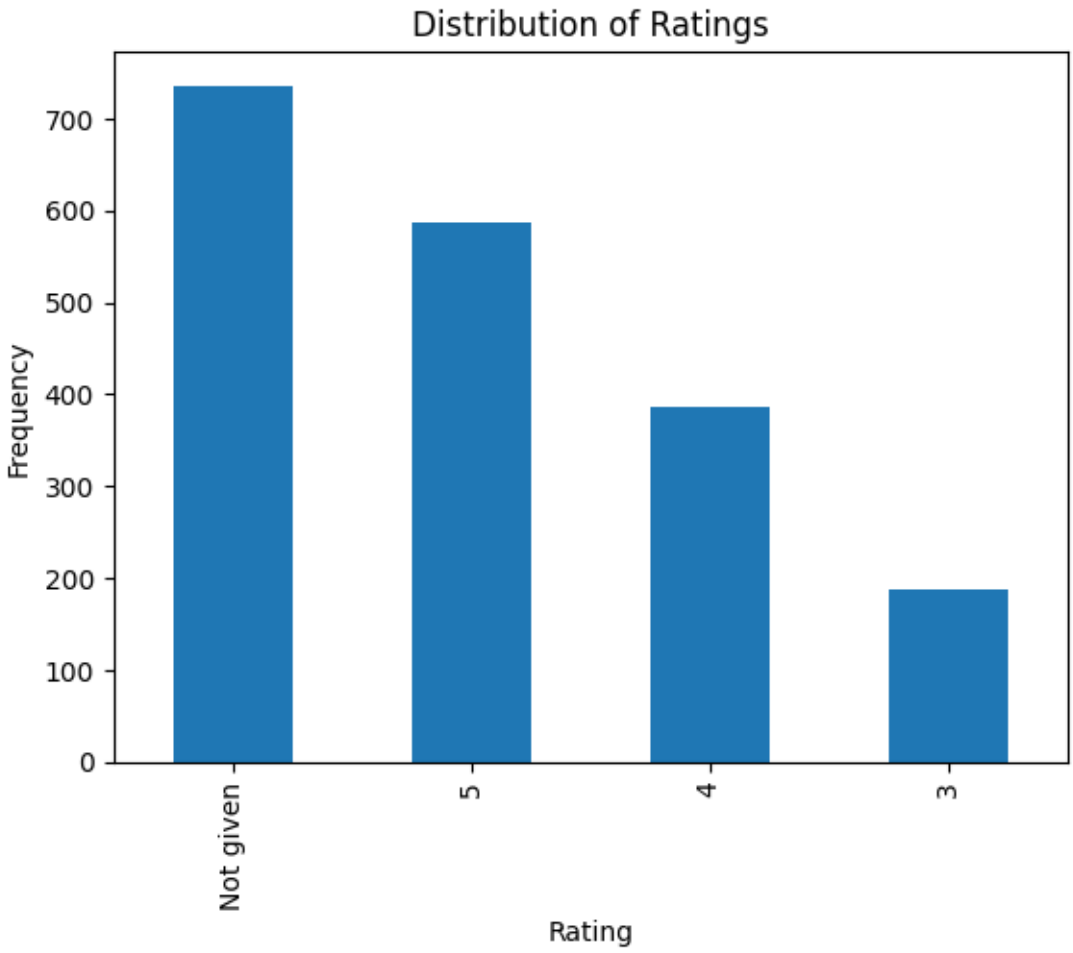
[ ]:
```
# Write the code here
rating_counts = df['rating'].value_counts()
unrated_orders = rating_counts.get('Not rated', 0)
print("Number of unrated orders:", unrated_orders)
```

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

Observations: There are no unrated orders.

1.0.12 Exploratory Data Analysis (EDA)

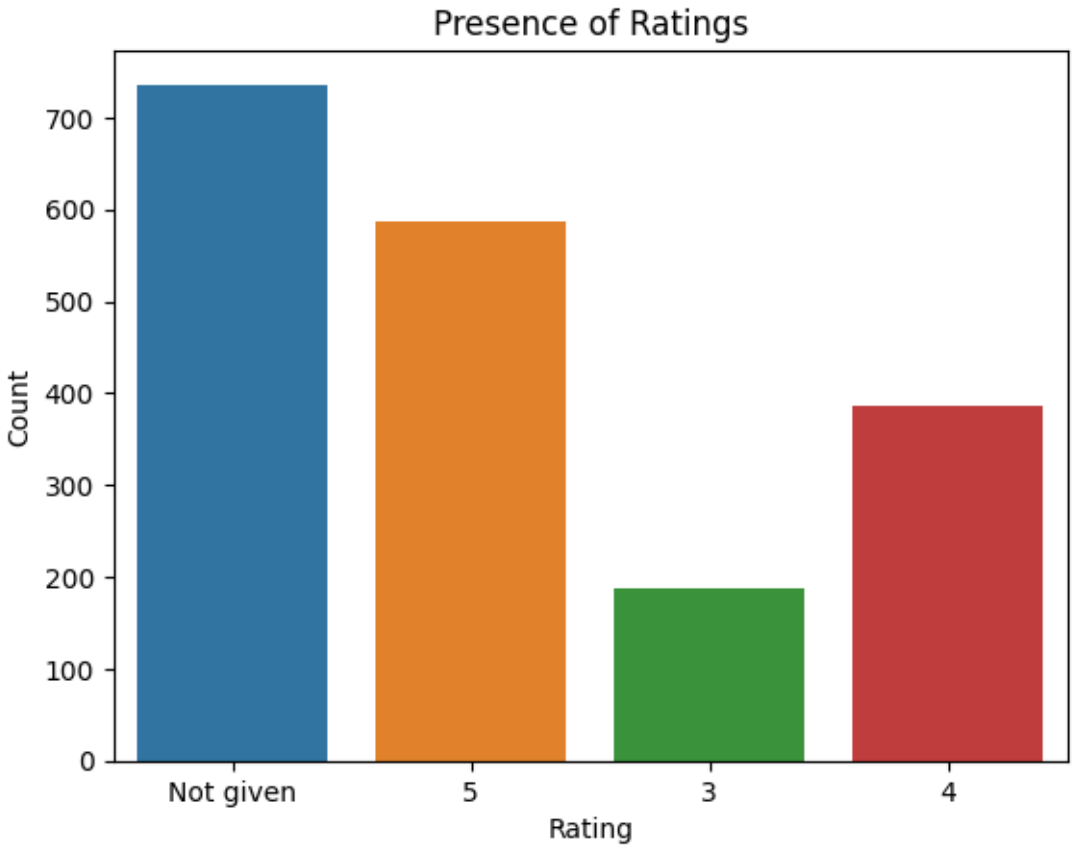
[ ]:
```
import matplotlib.pyplot as plt
df['rating'].value_counts().plot(kind='bar')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Distribution of Ratings')
plt.show()
```



1.0.13 Univariate Analysis

[ ]:
```
import seaborn as sns
sns.countplot(x='rating', data=df)
plt.xlabel('Rating')
plt.ylabel('Count')
```

```
plt.title('Presence of Ratings')
plt.show()
```



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

[ ]:
```
# Write the code here
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[ ]:
```
# Iterating over each variable and creating visualizations
for column in df.columns:
    plt.figure(figsize=(8, 6))
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

```
<Figure size 800x600 with 0 Axes>
```

1.0.15 Question 7: Which are the top 5 restaurants in terms of the number of orders received?

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

Observations: unfortunetly I could not get the code to work on this question.

1.0.16 Question 8: Which is the most popular cuisine on weekends?

[ ]:
```
# Data for weekend orders
weekend_orders = df[df['day_of_the_week'].isin(['Saturday', 'Sunday'])]
```

[ ]:
```
# Count the occurrences of each cuisine type
cuisine_counts = weekend_orders['cuisine_type'].value_counts()
```

[ ]:
```
print(df.columns)
```

```
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')
```

[ ]:
```
# Count the occurrences of each cuisine type
cuisine_counts = weekend_orders['cuisine_type'].value_counts()
```

[ ]:
```
# Check if there are any weekend orders
if len(cuisine_counts) == 0:
    print("No weekend orders found in the dataset.")
```

```
No weekend orders found in the dataset.
```

Observations: No weekend order found in dataset.

1.0.17 Question 9: What percentage of the orders cost more than 20 dollars?

[ ]:
```
# Calculate the percentage of orders that cost more than $20
cost_gt_20 = (df['cost_of_the_order'] > 20).sum()
percentage_gt_20 = (cost_gt_20 / len(df)) * 100
```

[ ]:
```
# Print the percentage
print("Percentage of orders costing more than $20:", percentage_gt_20)
```

```
Percentage of orders costing more than $20: 29.24130663856691
```

Observations: Approximately 29.24% of the orders in the dataset have a cost greater than $20. This indicates that a significant portion of the orders falls into the higher cost range.

1.0.18 Question 10: What is the mean order delivery time?

[ ]:
```
# Calculate mean() function from pandas
mean_delivery_time = df['delivery_time'].mean()
print("Mean order delivery time:", mean_delivery_time)
```

```
Mean order delivery time: 24.161749209694417
```

Observations: The mean order delivery time is approximately 24.16.The mean delivery time can be used as a benchmark for evaluating and improving the delivery performance.

1.0.19 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

[ ]:
```
# Count the number of orders for each customer
customer_orders = df['customer_id'].value_counts()
```

[ ]:
```
# Get the top 3 most frequent customers
top_customers = customer_orders.nlargest(3)
```

[ ]:
```
# Get the customer IDs and the corresponding number of orders
top_customer_ids = top_customers.index
top_customer_order_counts = top_customers.values
```

[ ]:
```
# Print the customer IDs and the number of orders
for i in range(len(top_customer_ids)):
    print("Customer ID:", top_customer_ids[i])
    print("Number of Orders:", top_customer_order_counts[i])
    print()
```

```
Customer ID: 52832
Number of Orders: 13
```

```
Customer ID: 47440
Number of Orders: 10

Customer ID: 83287
Number of Orders: 9
```

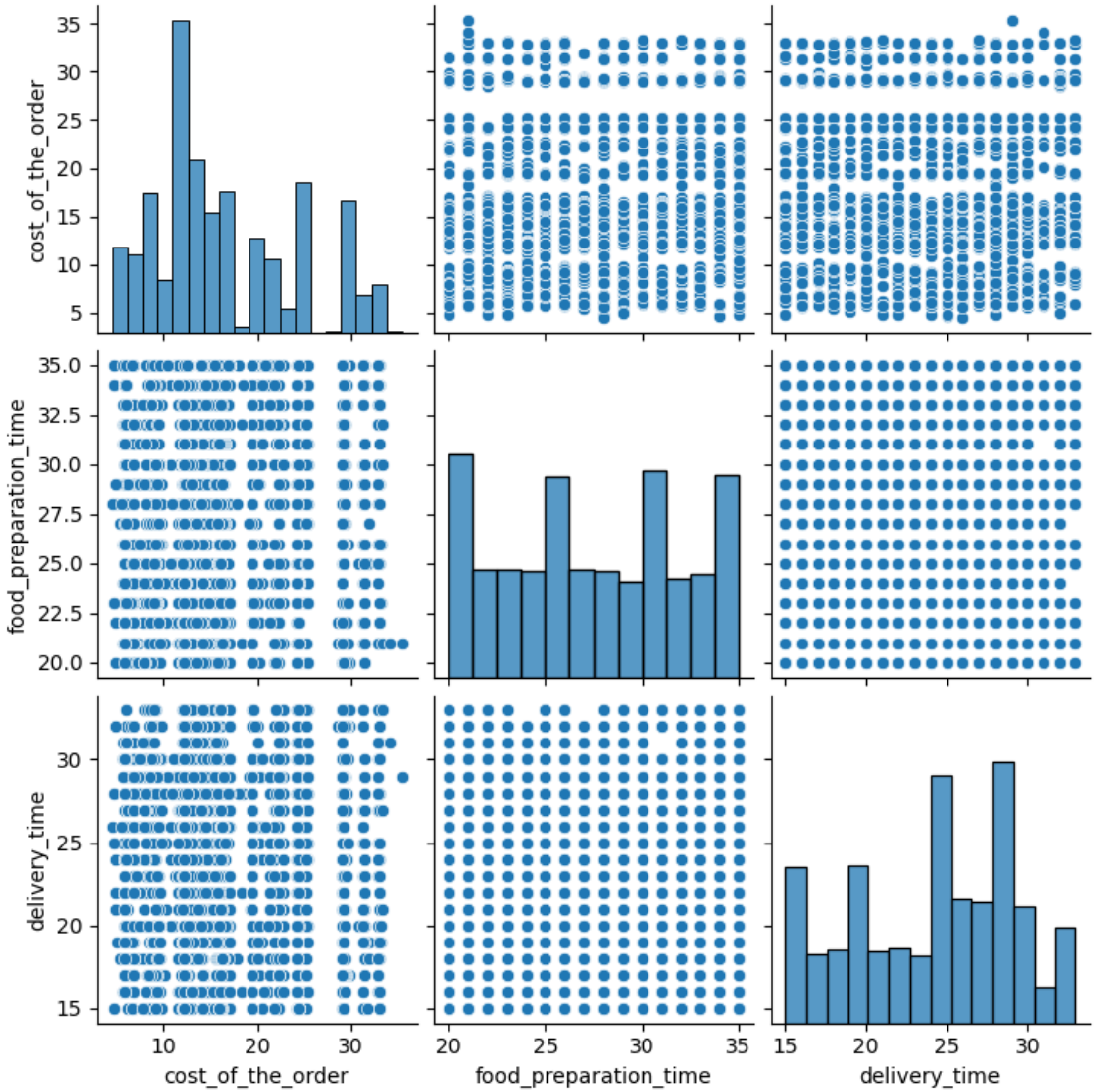
Observations: The customer with ID 52832 has placed the highest number of orders, with a total of 13 orders.The customer with ID 47440 is the second most frequent customer, with 10 orders.The customer with ID 83287 is the third most frequent customer, with 9 orders.

1.0.20 Multivariate Analysis

1.0.21 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)

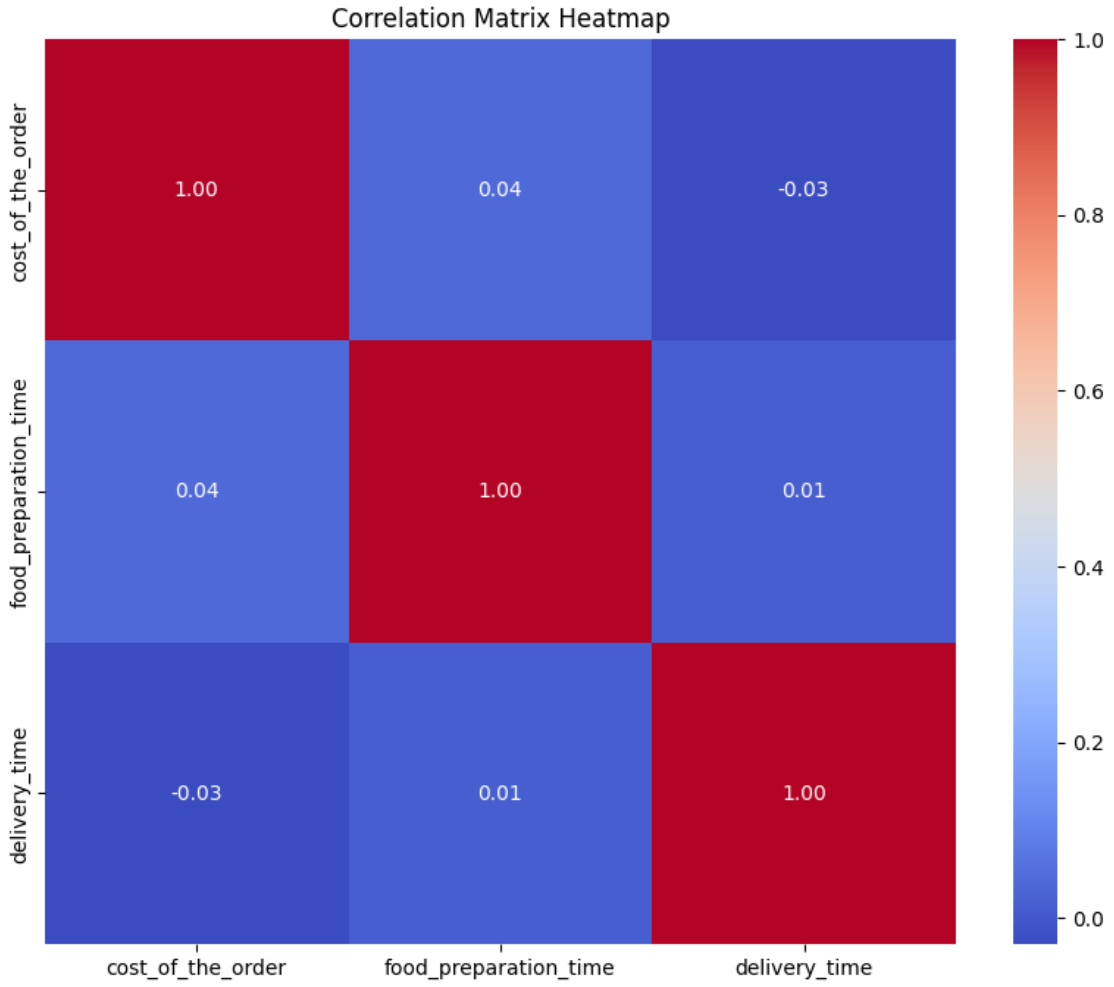
[ ]:
```
# Select relevant numerical variables for pairplot
numerical_vars = ['cost_of_the_order', 'food_preparation_time', 'delivery_time']
```

[ ]:
```
# Create pair plot
sns.pairplot(df[numerical_vars])
plt.show()
```



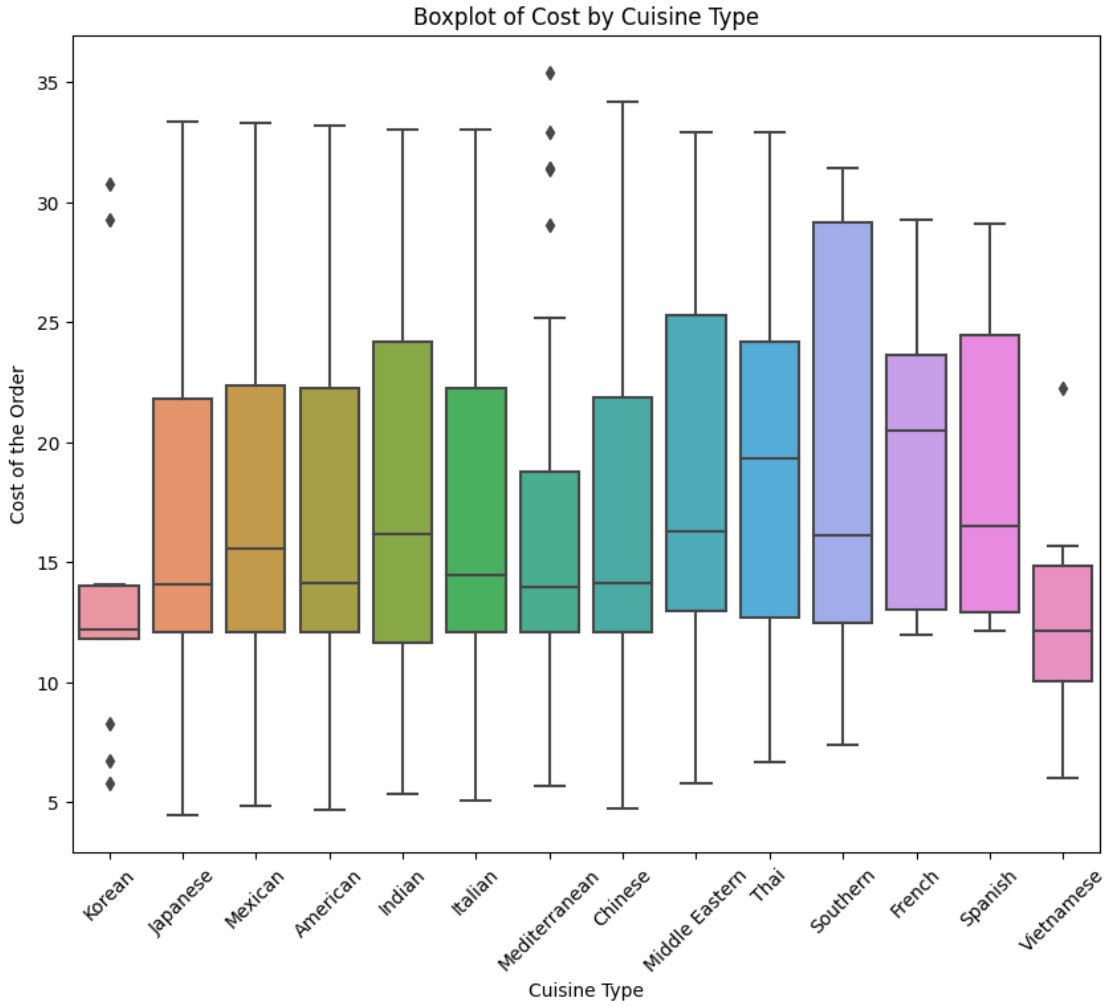
[ ]:
```
# Compute correlation matrix
corr_matrix = df[numerical_vars].corr()
```

[ ]:
```
# Create heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
plt.show()
```



[ ]:
```
# Select a relevant numerical variable and categorical variable
numerical_var = 'cost_of_the_order'
categorical_var = 'cuisine_type'
```

[ ]:
```
# Create boxplot
plt.figure(figsize=(10, 8))
sns.boxplot(data=df, x=categorical_var, y=numerical_var)
plt.title('Boxplot of Cost by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Cost of the Order')
plt.xticks(rotation=45)
plt.show()
```



1.0.22 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

Unfortunately I could not figure this code out.

1.0.23 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

[ ]:
```
# Calculate the net revenue for each order
df['net_revenue'] = df['cost_of_the_order']
```

[ ]:
```
# Apply the charges based on order cost
df.loc[df['cost_of_the_order'] > 20, 'net_revenue'] *= 0.75
df.loc[(df['cost_of_the_order'] > 5) & (df['cost_of_the_order'] <= 20),␣
↪'net_revenue'] *= 0.85
```

[ ]:
```
# Calculate the total net revenue
total_net_revenue = df['net_revenue'].sum()
```

[ ]:
```
# Print the total net revenue
print("Total net revenue generated by the company: $", total_net_revenue)
```

```
Total net revenue generated by the company: $ 25148.517
```

Observations: Based on the calculation, the total net revenue generated by the company across all orders is $25,148.517. This represents the amount of revenue earned by the company after applying the charges based on the order cost.

1.0.24 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)

[ ]:
```
# Calculate the percentage of orders that take more than 60 minutes to deliver
orders_gt_60_minutes = (df['food_preparation_time'] + df['delivery_time']) > 60
percentage_gt_60_minutes = (orders_gt_60_minutes.sum() / len(df)) * 100
```

[ ]:
```
# Print the percentage
print("Percentage of orders taking more than 60 minutes to deliver:",␣
↪percentage_gt_60_minutes)

# Print the percentage
print("Percentage of orders taking more than 60 minutes to deliver:",␣
↪percentage_gt_60_minutes)
```

```
Percentage of orders taking more than 60 minutes to deliver: 10.537407797681771
```

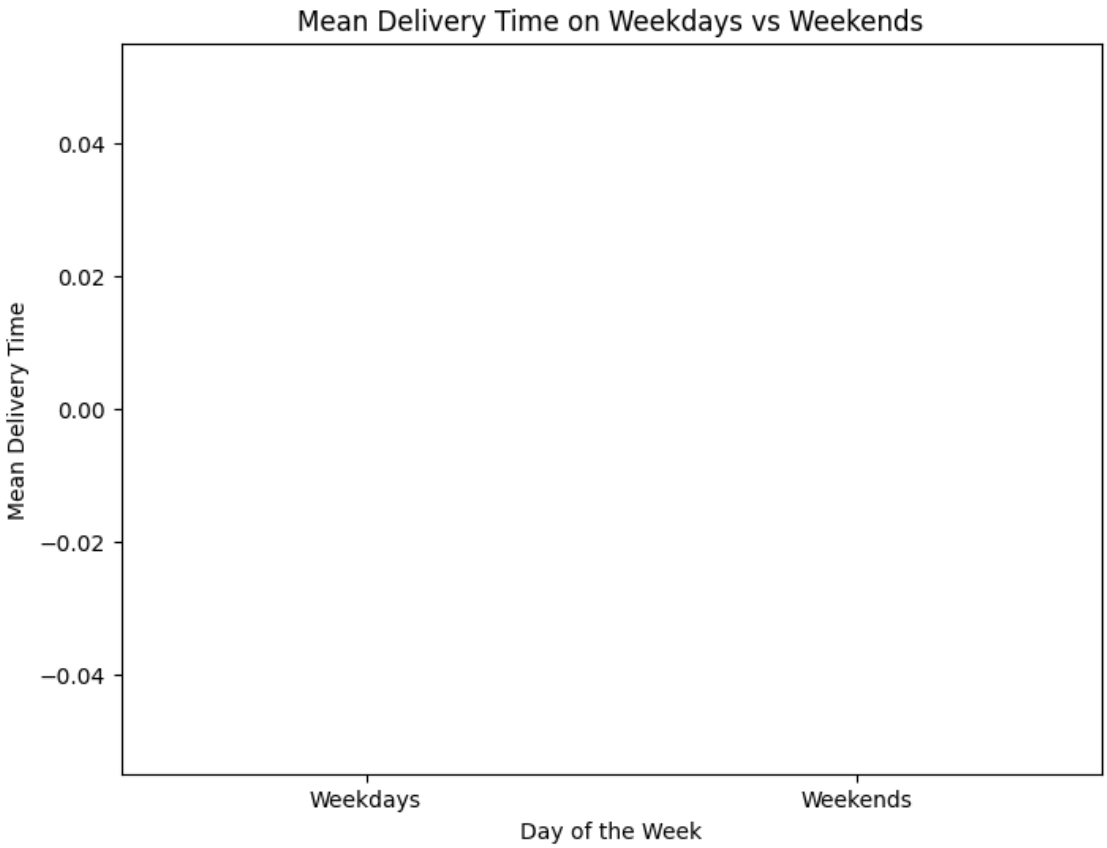
Observations: Approximately 10.54% of the orders in the dataset experience delivery times exceeding 60 minutes.This indicates that a significant portion of orders may face longer delivery times.

1.0.25 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?

[ ]:
```
# Filter the data for weekday and weekend orders
weekday_orders = df[df['day_of_the_week'].isin(['Monday', 'Tuesday',␣
↪'Wednesday', 'Thursday', 'Friday'])]
weekend_orders = df[df['day_of_the_week'].isin(['Saturday', 'Sunday'])]
```

[ ]:
```
# Calculate the mean delivery time for weekdays and weekends
mean_delivery_time_weekday = weekday_orders['delivery_time'].mean()
mean_delivery_time_weekend = weekend_orders['delivery_time'].mean()
```

[ ]:
```
# Create a bar plot to visualize the mean delivery time
plt.figure(figsize=(8, 6))
sns.barplot(x=['Weekdays', 'Weekends'], y=[mean_delivery_time_weekday,␣
↪mean_delivery_time_weekend])
plt.xlabel('Day of the Week')
plt.ylabel('Mean Delivery Time')
plt.title('Mean Delivery Time on Weekdays vs Weekends')
plt.show()
```



Observations:

1.0.26 Conclusion and Recommendations

1.0.27 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)

Conlusions: 1. Popular Cuisine Types: The analysis identified the popularity of different cuisine types among customers based on the dataset.2. Feedback Ratings: The analysis provided insights into customer feedback ratings and their impact on the business.

1.0.28 Recommendations:

1. Focus on Promoting Popular Cuisines: Allocate resources and marketing efforts towards the promotion and expansion of the most popular cuisine types to attract more customers.2.Enhance Quality Control: Address areas for improvement based on customer feedback to ensure consistent food quality, taste, and presentation.3.Optimize Delivery Process: Streamline operations, improve coordination between the kitchen and delivery staff, and implement strategies to minimize delivery times and ensure timely service.4.Implement Loyalty and Rewards Program: Introduce a loyalty program to incentivize repeat orders, reward frequent customers, and build customer loyalty.5.Strengthen Online Presence and Marketing: Develop a strong online presence, engage with customers through social media platforms, and utilize targeted digital marketing campaigns to reach a wider audience.

-
