Project Python Foundations: FoodHub Data Analysis

Marks: 60

Context

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

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

Objective

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

Data Description

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

Data Dictionary

- order_id: Unique ID of the order
- customer_id: ID of the customer who ordered the food
- restaurant_name: Name of the restaurant

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

Let us start by importing the required libraries

```
# 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

```
# I upload the file directly to colab to 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_o
0	1477147	337525	Hangawi	Korean	30.75	
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	
2	1477070	66393	Cafe Habana	Mexican	12.23	
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	

Observations:

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

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

```
df
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day
0	1477147	337525	Hangawi	Korean	30.75	
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	
2	1477070	66393	Cafe Habana	Mexican	12.23	
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	
4	1478249	76942	Dirty Bird to Go	American	11.59	
...	...	...	...	...	...	
1893	1476701	292602	Chipotle Mexican Grill $1.99 Delivery	Mexican	22.31	
1894	1477421	397537	The Smile	American	12.18	
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	
1896	1477513	64151	Jack's Wife Freda	Mediterranean	12.18	

```
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: There are 9 columns within the datafram indexing from 0-8 in order as; order_id, customer_id, restaurant_name, cuisine_type, cost_of_the_order, day_of_the_week, rating, food_preparation_time, and delivery_time.

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

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

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

Observations: 'cost_of_the_order' is the only float, which makes sense as dollar orders typically are rounded out to the cent. We also have object for all string columns and int64 for all integer columns. Need to check on 'rating' column as is shows object in column with a numerical rating.

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

```
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

dtype: int64

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

```
array(['Not given', '5', '3', '4'], dtype=object)
```

```
# Convert 'Not given' to NaN
df['rating'] = df['rating'].replace(['Not given'], np.nan)
# Convert type to float
df['rating'] = df['rating'].astype(float)
```

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

rating	count
5.0	588
4.0	386
3.0	188

dtype: int64

```
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

```
# This was the code I was attempting to use to convert the NaN values in rating to a

#def fill_rating_with_criteria(row, df):
#     if np.isnan(row['rating']):
#         # Define the range for cost_of_the_order
#         cost_lower = row['cost_of_the_order'] - 5
#         cost_upper = row['cost_of_the_order'] + 5

#         # Filter rows that match the criteria
#         matching_rows = df[
#             (df['cuisine_type'] == row['cuisine_type']) &
#             (df['restaurant_name'] == row['restaurant_name']) &
#             (df['day_of_the_week'] == row['day_of_the_week']) &
#             (df['cost_of_the_order'] >= cost_lower) &
#             (df['cost_of_the_order'] <= cost_upper)
#         ]

#         # Calculate the mean rating for matching rows
#         mean_rating = matching_rows['rating'].mean()

#         # Return the rounded mean rating if available, otherwise leave as NaN
#         return round(mean_rating, 1) if not np.isnan(mean_rating) else np.nan
#     else:
#         # If the rating is not NaN, keep the original value
#         return row['rating']

# # Apply the function to fill the NaN values in the 'rating' column
# df['rating'] = df.apply(fill_rating_with_criteria, axis=1, df=df)

# # Check the first few rows to verify
# print(df['rating'].head())
```

Observations: While there are no NaN values present in the data set, there are 736 "Not given" ratings within the 'rating' column. I would consider these to be missing values and values that need to be ammended before performing analytics with the 'rating' column involved. So I converted these values to NaN. I Then provided a a line of code to fill in the NaN values in 'rating' with the mean of rows matched with the cuisine type, restaurant name, day of the week, and cost of the order within plus or minus $5. However, for the final analysis I decided to leave this code out as it would change the STD by about 12% which is too high for my liking.

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]

```
df.describe().T
```

	count	mean	std	min	25%	5
order_id	1898.0	1.477496e+06	548.049724	1476547.00	1477021.25	1477495
customer_id	1898.0	1.711685e+05	113698.139743	1311.00	77787.75	128600
cost_of_the_order	1898.0	1.649885e+01	7.483812	4.47	12.08	14
rating	1162.0	4.344234e+00	0.741478	3.00	4.00	5
food_preparation_time	1898.0	2.737197e+01	4.632481	20.00	23.00	27
delivery_time	1898.0	2.416175e+01	4.972637	15.00	20.00	25

Observations: 'food_preparation_time' shows us the difference between the timestamps of the restaurant's order confirmation and the delivery person's pick-up confirmation. The statistics behind this difference are as follows: minimum = 20.00 minutes, average ~ 27.37 minutes, maximum = 35.00 minutes.

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

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

rating	count
5.0	588
4.0	386
3.0	188

dtype: int64

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

736

Observations: 736 orders are not rated within the 'rating' column.

Exploratory Data Analysis (EDA)

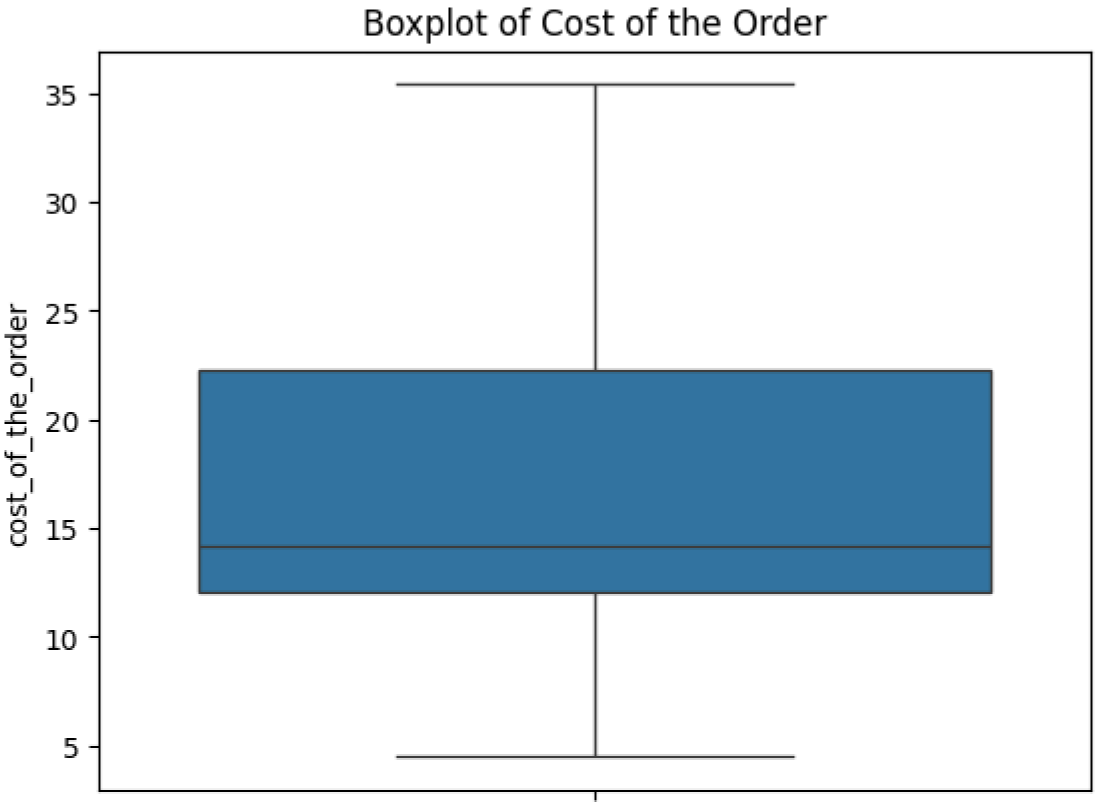
Univariate Analysis

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

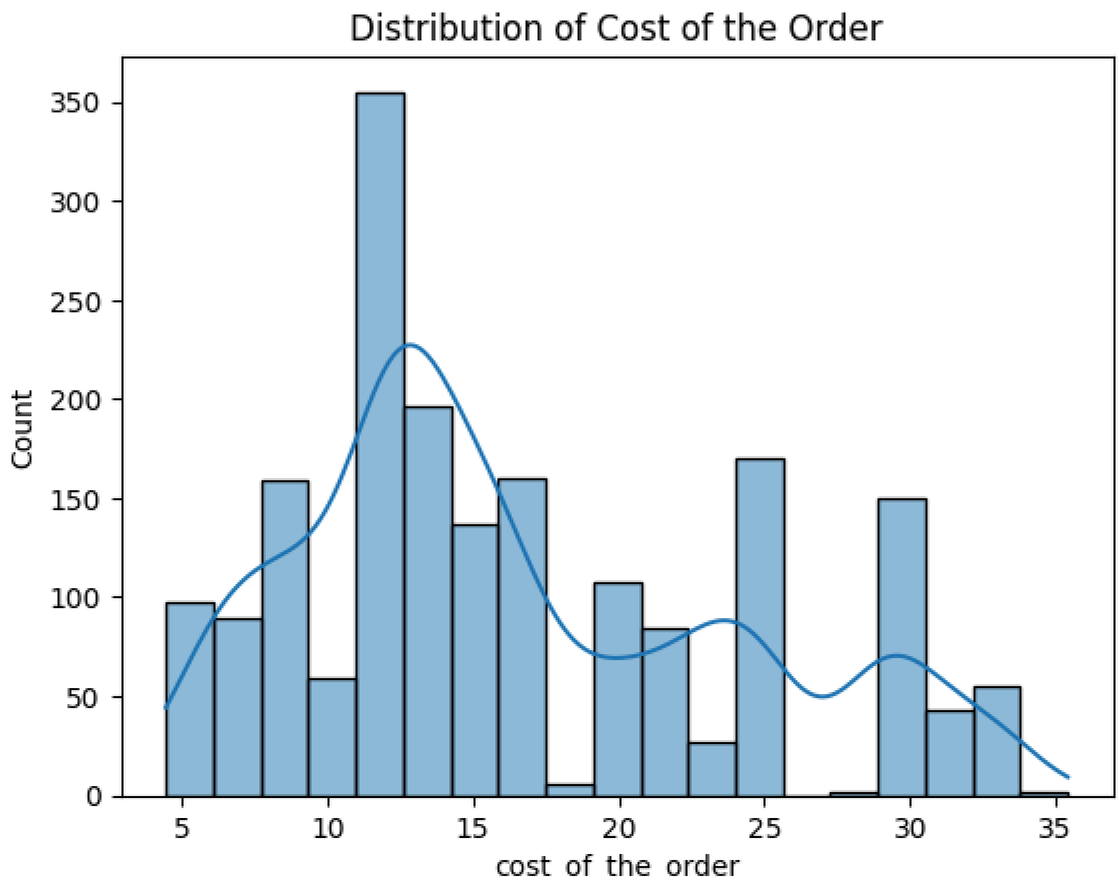
```
df.info()
```

```
<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                 1162 non-null   float64
 7   food_preparation_time  1898 non-null   int64  
 8   delivery_time          1898 non-null   int64  
dtypes: float64(2), int64(4), object(3)
memory usage: 133.6+ KB
```

```
sns.boxplot(df['cost_of_the_order'])
plt.title('Boxplot of Cost of the Order')
plt.show()
```




```
sns.histplot(df['cost_of_the_order'], kde=True)
plt.title('Distribution of Cost of the Order')
plt.show()
```




```
np.floor(df['cost_of_the_order']).value_counts()
```

cost_of_the_order	count
12.0	340
29.0	150
14.0	143
16.0	126
24.0	117
9.0	105
15.0	101
8.0	96
6.0	91
19.0	86
13.0	73
5.0	70
22.0	70
25.0	53
11.0	48
31.0	42
21.0	40
32.0	30
33.0	25
7.0	23
20.0	23
17.0	18
10.0	10
4.0	9
18.0	4
28.0	2
30.0	1
35.0	1
34.0	1

dtype: int64

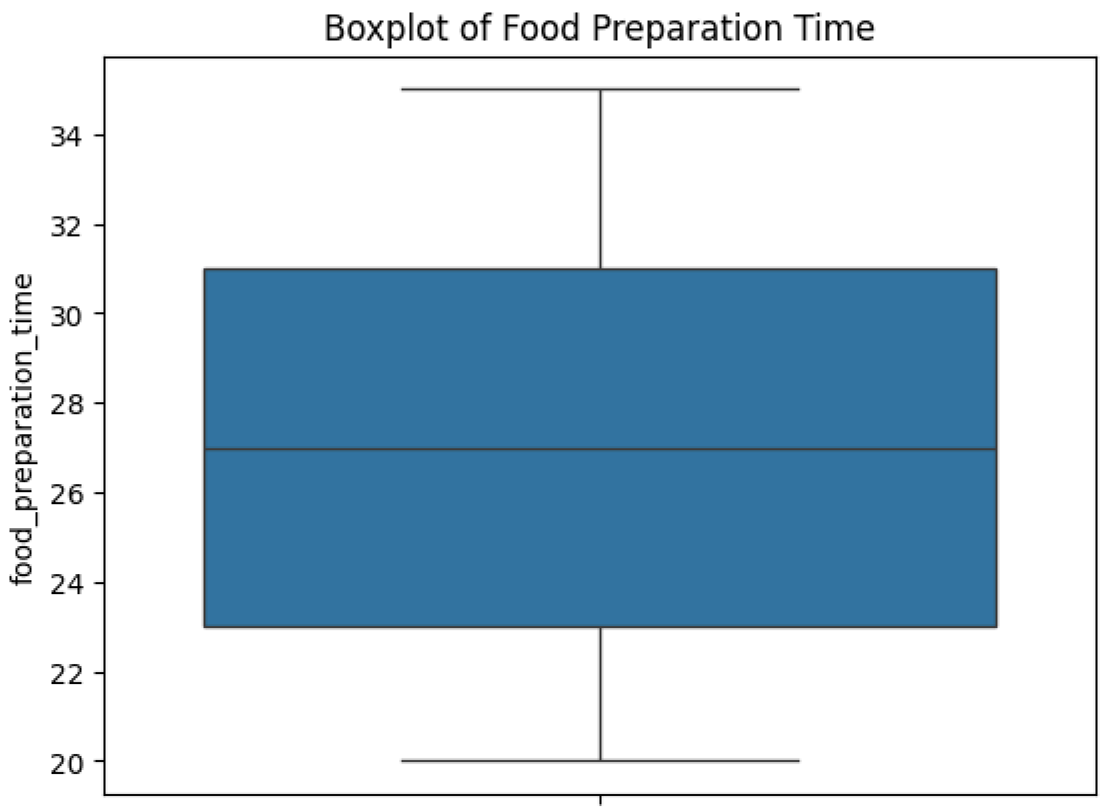
```
df['cost_of_the_order'].describe().T
```

	cost_of_the_order
count	1898.000000
mean	16.498851
std	7.483812
min	4.470000
25%	12.080000
50%	14.140000
75%	22.297500
max	35.410000

dtype: float64

COST_OF_THE_ORDER OBSERVATION: The cost of orders data is skewed slightly to the right with 50% of the orders costing between 4.47USD to 14.14USD and the other 50% of orders costing between 14.14USD and 35.41USD (a slightly larger IQR between the top half of the data). There seem to be no extreme outliers here, but rather a majority of orders between about 5USD and 17USD, a bundle of orders between about 19USD and 25USD, and finally, a group of orders pulling data to the right that cost between about 29USD to 34USD. When rounding this column to the nearest whole number, 12USD is the cost of order 2x more frequently than the second place cost. Following that, 29USD, 14USD, 16USD, and 24USD trail tightly together.

```
sns.boxplot(df['food_preparation_time'])
plt.title('Boxplot of Food Preparation Time')
plt.show()
```




```
df['food_preparation_time'].describe().T
```

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

dtype: float64

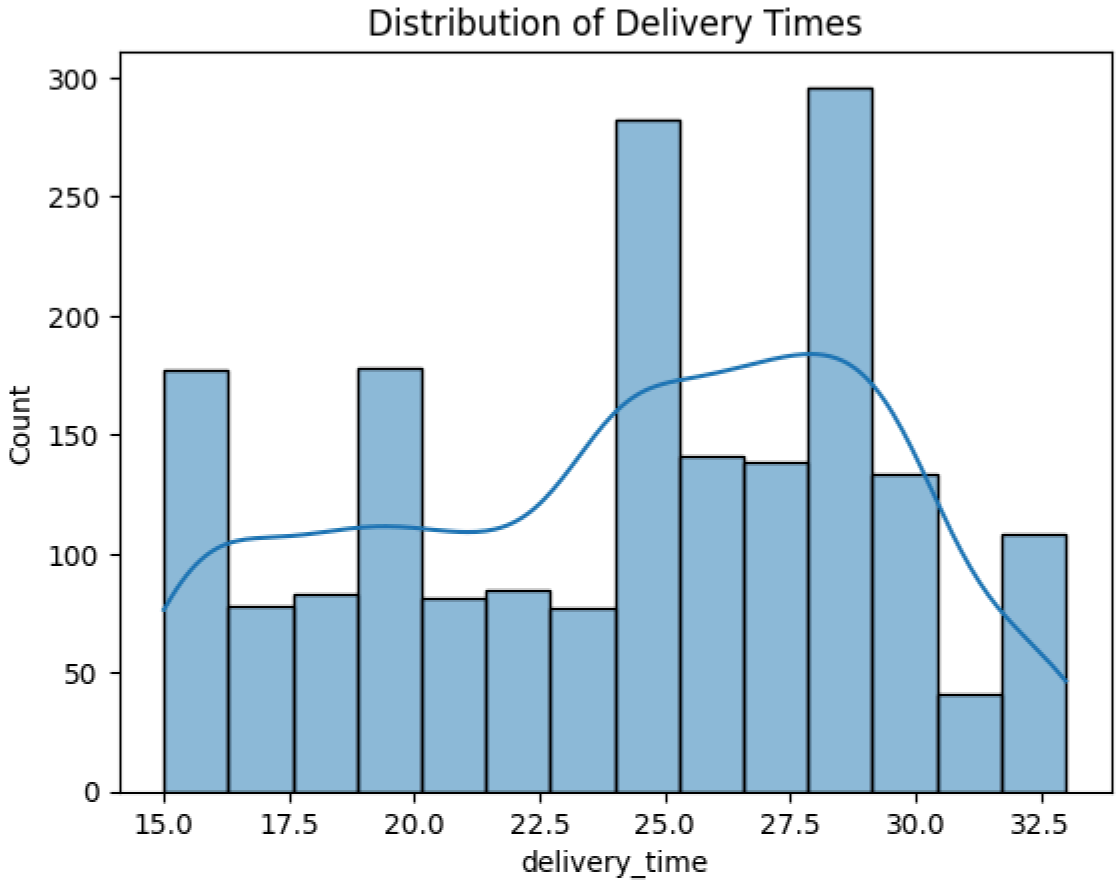
```
df['food_preparation_time'].value_counts()
```

food_preparation_time	count
21	135
23	123
27	123
22	123
28	121
24	121
20	119
30	119
33	118
35	117
31	116
26	115
25	113
34	113
32	113
29	109

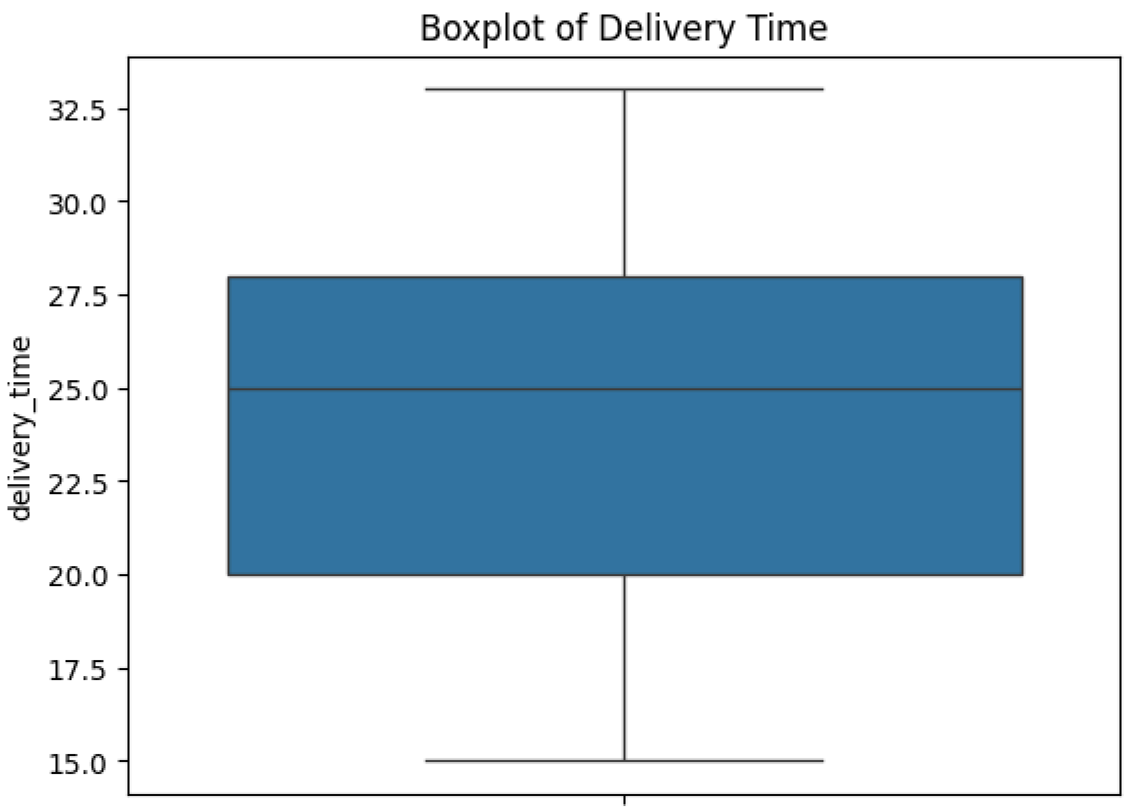
dtype: int64

FOOD_PREPARATION_TIME OBSERVATION: This is a relatively normal distribution of data as the average time to prep (27.37 minutes) and median time to prep (27 minutes) are very close to eachother. With 20 minutes being the minimum and 35 minutes being the maximum, there are no outliers. There is not much use for this information without cross analysis. Possibilities for multivariate analysis with prep time could include rating, restaurant_name, day_of_the_week, and more.

```
sns.histplot(df['delivery_time'], kde=True)
plt.title('Distribution of Delivery Times')
plt.show()
```




```
sns.boxplot(df['delivery_time'])
plt.title('Boxplot of Delivery Time')
plt.show()
```




```
df['delivery_time'].describe().T
```

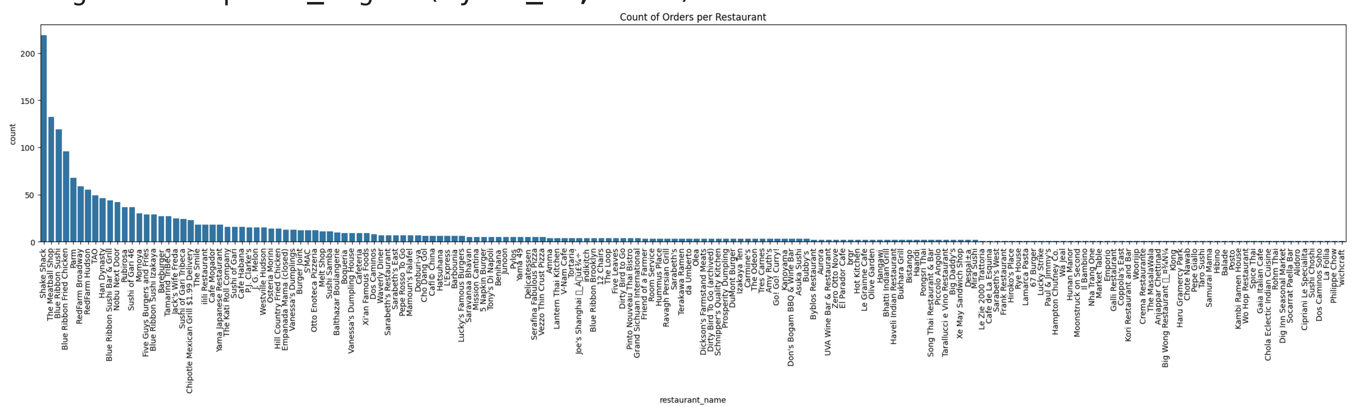
	delivery_time
count	1898.000000
mean	24.161749
std	4.972637
min	15.000000
25%	20.000000
50%	25.000000
75%	28.000000
max	33.000000

dtype: float64

DELIVERY_TIME OBSERVATION: There are no outliers here with a minimum delivery time of 15 minutes and a maximum of 33 minutes. The data is slightly skewed left as there is a slightly larger spread between the minimum delivery time and median delivery time. Again, there is not much to pull from this data without cross analysis using data from (possibly) the rating, restaurant_name, or other columns.

```
plt.figure(figsize=(30,5))
#Using the only index values of the automatically ordered value_counts function
#to order this countplot from highest to lowest with regards to count
sns.countplot(x='restaurant_name', data=df, order=df['restaurant_name'].value_counts
plt.xticks(rotation=90)
plt.title('Count of Orders per Restaurant')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarn
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarn
  fig.canvas.print_figure(bytes_io, **kw)
```




```
df['restaurant_name'].value_counts().head(20)
```

restaurant_name	count
Shake Shack	219
The Meatball Shop	132
Blue Ribbon Sushi	119
Blue Ribbon Fried Chicken	96
Parm	68
RedFarm Broadway	59
RedFarm Hudson	55
TAO	49
Han Dynasty	46
Blue Ribbon Sushi Bar & Grill	44
Nobu Next Door	42
Rubirosa	37
Sushi of Gari 46	37
Momoya	30
Five Guys Burgers and Fries	29
Blue Ribbon Sushi Izakaya	29
Bareburger	27
Tamarind TriBeCa	27
Jack's Wife Freda	25
Sushi of Gari Tribeca	24

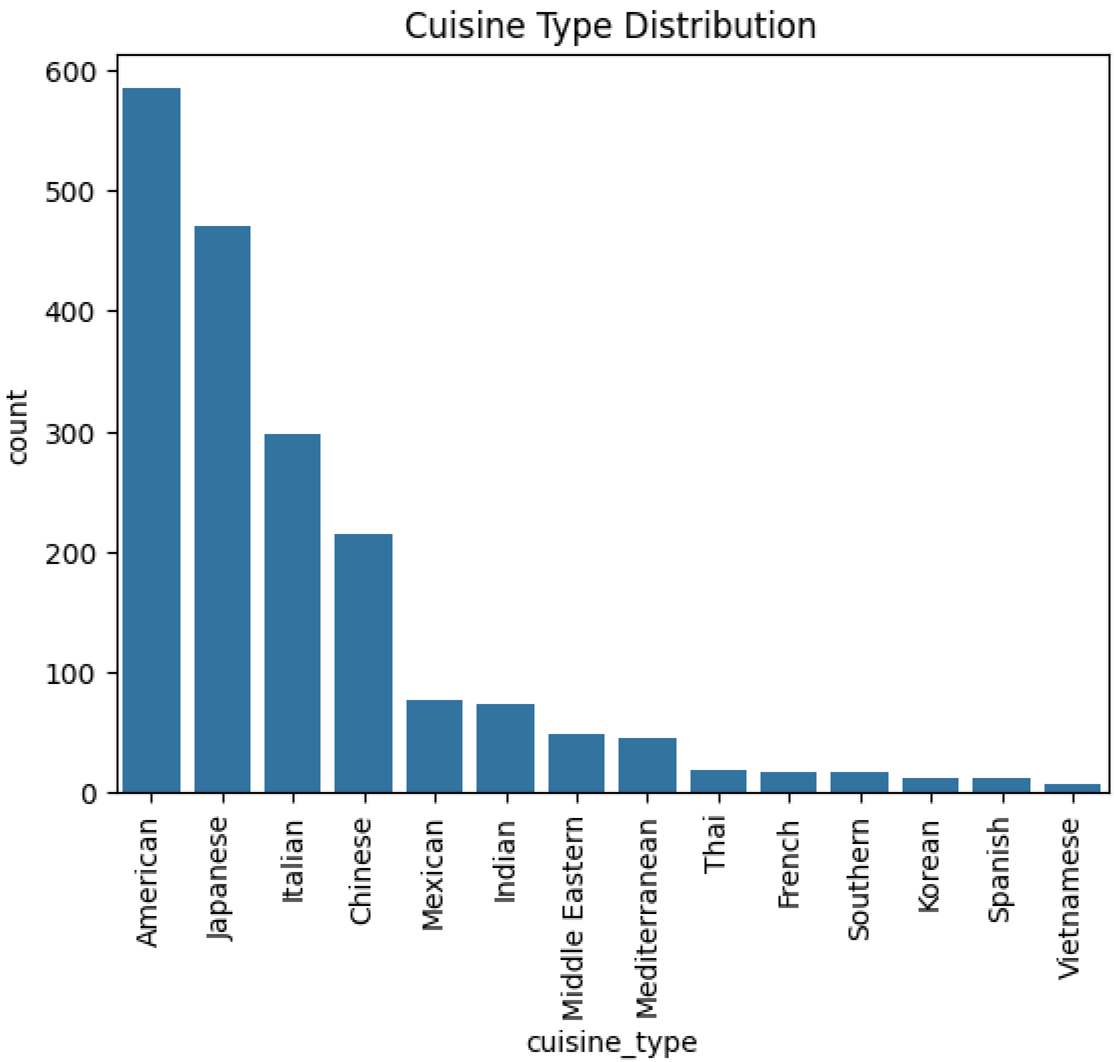
dtype: int64

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

178

RESTAURANT_NAME OBSERVATION: While the countplot is hard to read, it gives a vizualization of how the top few restaurants dominate the order volume in a dataset with 178 restaruants inside of it. The top five restaurants based on order volume in the dataset are: Shake Shack 219, The Meatball Shop 132, Blue Ribbon Sushi 119, Blue Ribbon Fried Chicken 96, Parm 68. With 1898 rows of data, just these top 5 restaurants make up 33.40% of the total orders within the data.

```
sns.countplot(x='cuisine_type', data=df, order=df['cuisine_type'].value_counts().ind
plt.xticks(rotation=90)
plt.title('Cuisine Type Distribution')
plt.show()
```




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

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

dtype: int64

CUISINE_TYPE OBSERVATION: Similar to the name of restaurant column, the cuisine type is dominated by only a few. The top four stand out much further than the rest, and those are: American 584, Japanese 470, Italian 298, and Chinese 215. These four cuisine types account for an astounding 82.56% of the total orders within the data.

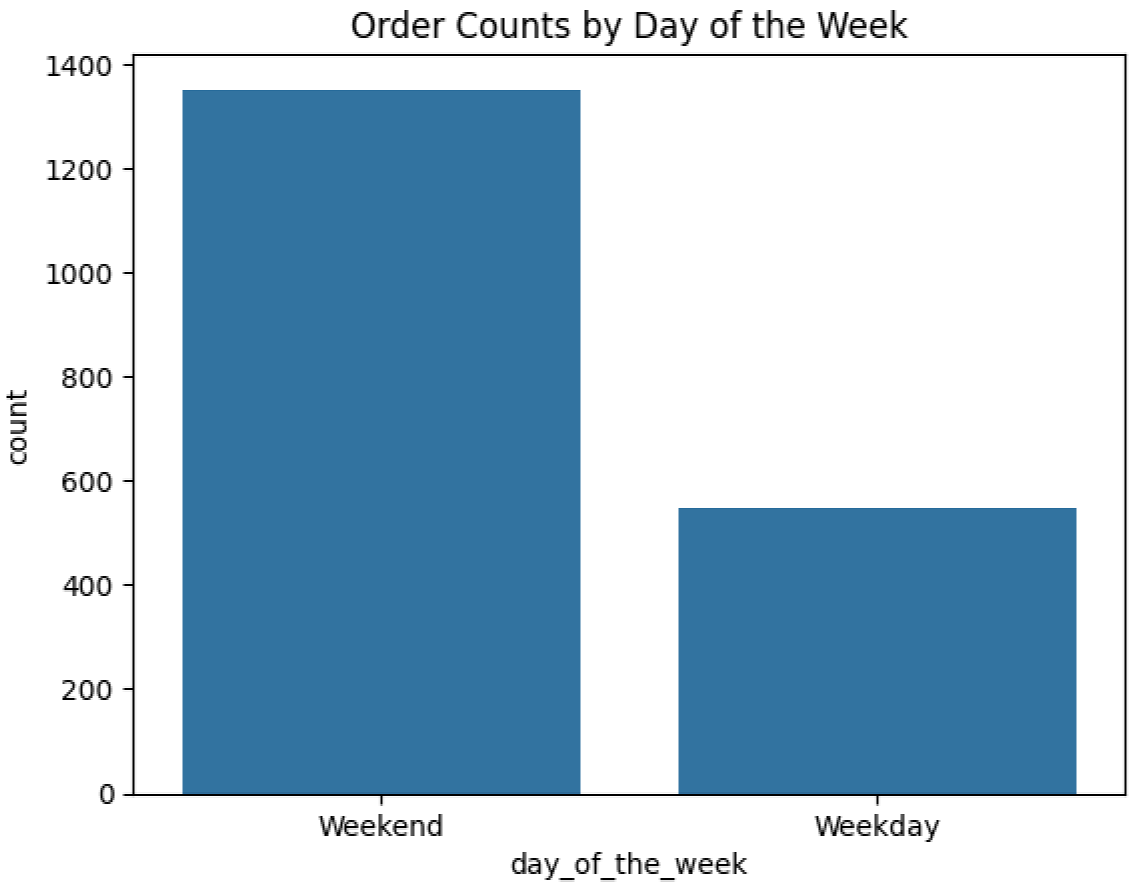
```
df['day_of_the_week'].value_counts()
```

day_of_the_week	count
Weekend	1351
Weekday	547

dtype: int64

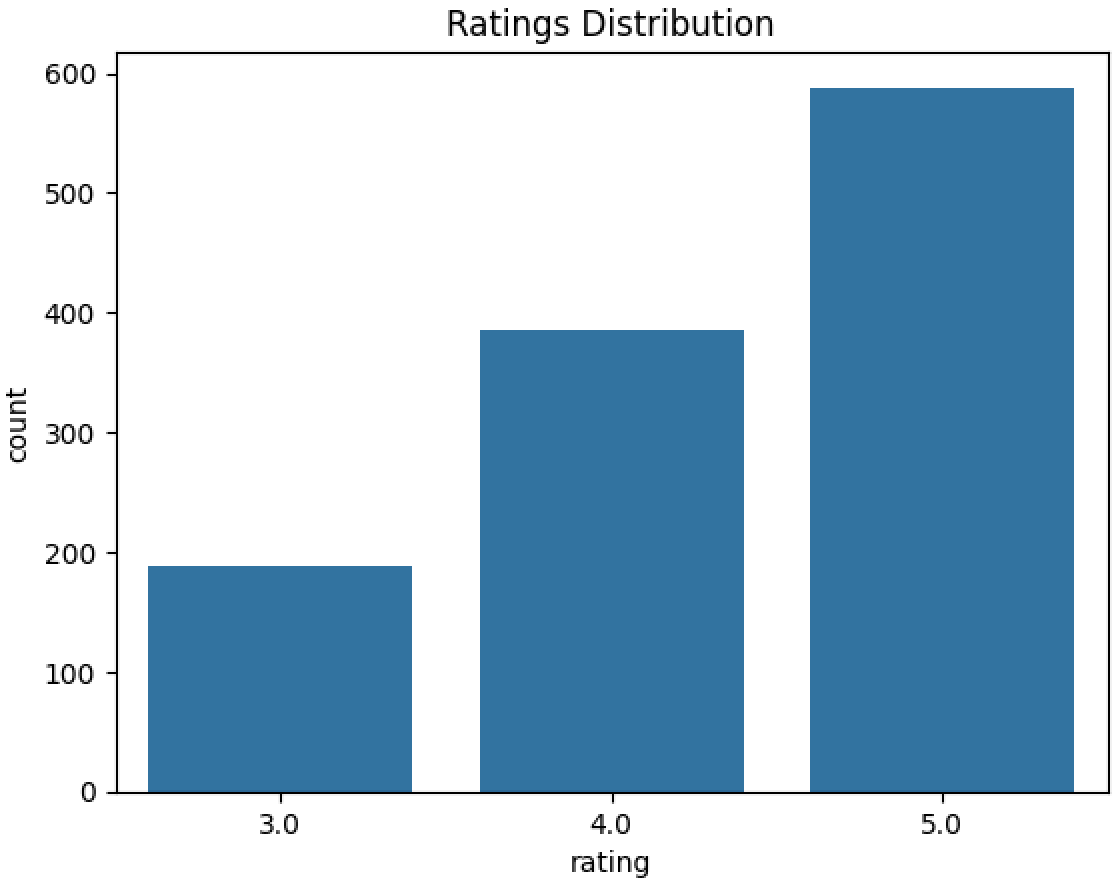
```
sns.countplot(x='day_of_the_week', data=df, order=['Weekend', 'Weekday'])
plt.title('Order Counts by Day of the Week')
```

```
plt.show()
```



DAY_OF_THE_WEEK OBSERVATION: Weekends account for 71.18% of the total orders within this data set. It will be worth looking into the difference in std when doing multivariate analysis between weekends & rating/delivery_time/cost_of_the_order vs. weekdays & rating/delivery_time/cost_of_the_order. My assumption is that weekends will prove to be more profitable at the cost of more poor ratings, longer delivery times, etc.

```
sns.countplot(x='rating', data=df)
plt.title('Ratings Distribution')
plt.show()
```




```
df['rating'].describe().T
```

	rating
count	1162.000000
mean	4.344234
std	0.741478
min	3.000000
25%	4.000000
50%	5.000000
75%	5.000000
max	5.000000

dtype: float64

RATING OBSERVATION: Too many ratings are NaN to consider this column within our final analysis and observations.

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)
```

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

dtype: int64

Observations: The top 5 restaurants in terms of number of orders recieved are Shake Shack, The Meatball Shop, Blue Ribbon, Sushi, Blue Ribbon, Fried Chicken, Parm. These 5 account for around 33.40% of the total orders recieved within this data set.

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

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

cuisine_type	count
American	415
Japanese	335
Italian	207
Chinese	163
Mexican	53

dtype: int64

Observations: American is the most popular with 415 orders on the Weekend, followed by Japanese (335), Italian (207), Chinese (163), and Mexican (53)

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

```
#total number of rows in the df
df.shape[0]
```

```
1898
```

```
#total number of rows in the df with a 'cost_of_the_order' > 20
df[df['cost_of_the_order']>20].shape[0]
```

```
555
```

```
#number of rows with coto>20 divided by total n of rows times 100, rounded to 2nd de
perc_over_20 = round((df[df['cost_of_the_order']>20].shape[0]/df.shape[0])*100,2)
print(f"{perc_over_20}% of the orders cost more than $20.")
```

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

Observations: 29.24% of orders cost more than 20USD.

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

```
#rounding the mean of all values in the delivery_time column to the 2nd decimal
mean_delivery = round(df['delivery_time'].mean(),2)
print(f"The mean order delivery time is {mean_delivery} minutes.")
```

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

Observations: 24.16 minutes is the average order delivery time.

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

```
df['customer_id'].value_counts().head(9)
```

customer_id	count
52832	13
47440	10
83287	9
250494	8
259341	7
82041	7
65009	7
276192	7
97079	6

dtype: int64

```
# Get the customer IDs with exactly 7 orders
customers_with_7_orders = df['customer_id'].value_counts()[df['customer_id'].value_c

# Filter the DataFrame for those customer IDs
df_filtered = df[df['customer_id'].isin(customers_with_7_orders)]

# Group by customer_id and sum the cost_of_the_order for each customer
total_spent_by_each_customer = df_filtered.groupby('customer_id')['cost_of_the_order

# Display the result
print(total_spent_by_each_customer)
```

```
customer_id
65009      99.49
82041     120.92
259341    130.81
276192    146.46
Name: cost_of_the_order, dtype: float64
```

Observations: The top four customers are obvious, but the 5th to 8th customer are all in a tie with 7 orders each. So to decide the 5th customer I've used some code to find who has spent the most out of those four customers tied with 7 orders. The final 5 customers recieving discounts are 52832, 47440, 83287, 250494, and 276192.

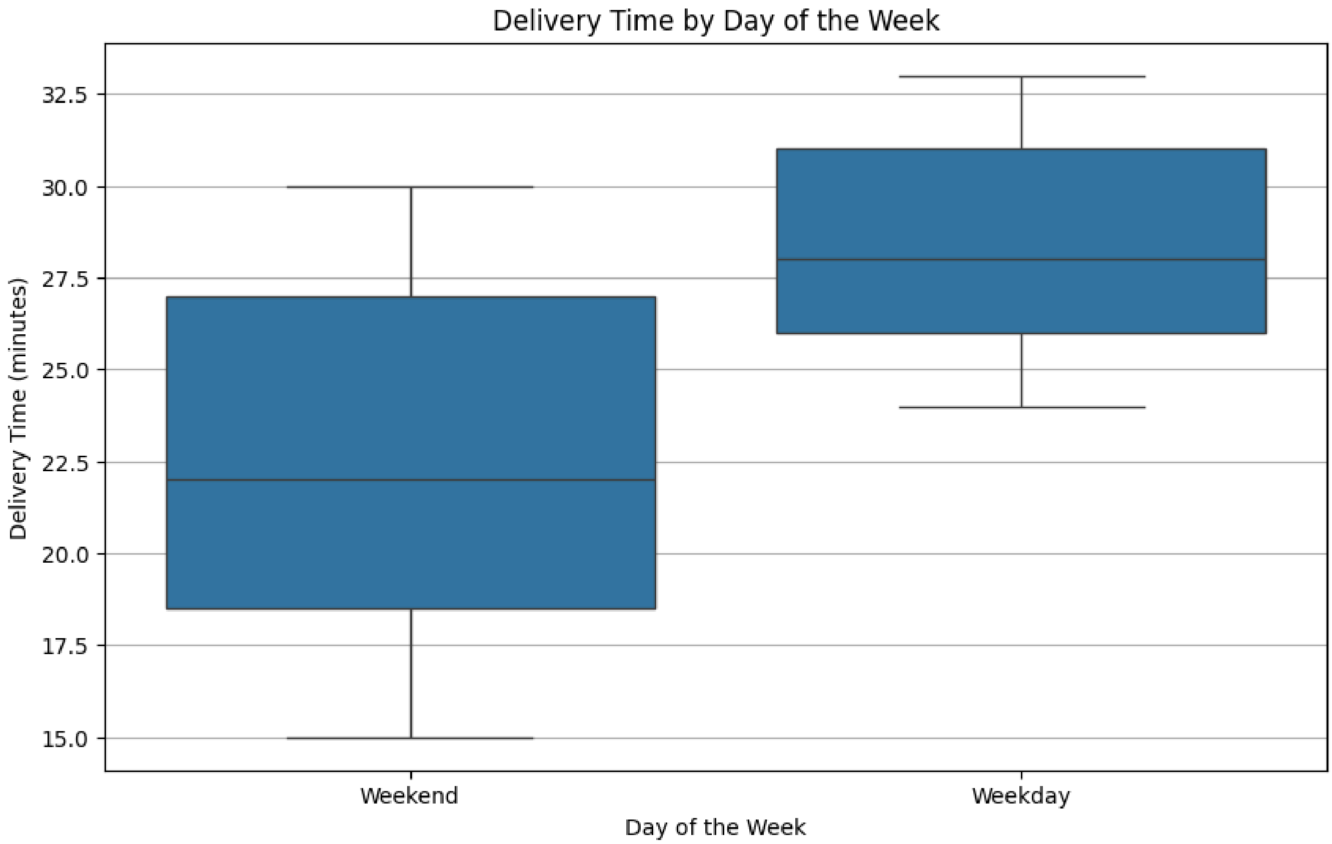
Multivariate Analysis

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]

analyze the data to get a fair idea about the demand of different restaurants which will help them in enhancing their customer experience.

1. Day of the week vs. Delivery time
2. Cost of order vs. Food prep time
3. Cuisine type vs. Food Prep Time vs. Delivery Time
4. Cuisine type vs. Day of the Week vs. Cost of Order
5. (value count top 10)Restaurant Name vs. Day of the Week
6. (value count top 10)Restrauant Name vs. Day of the Week vs. Delivery Time
7. Avg Cost of Order vs. Cuisine Type
8. Avg Delivery Time vs. Cuisine Type
9. Avg Food Prep Time vs. Cuisine Type
10. Avg Total Order to Deliver Time vs. Cuisine Type
11. Total Delivery Time of Each Cuisine vs. Day of the Week
12. Correlation Matrix of numerical columns

```
#Create size plot for the figure
plt.figure(figsize=(10, 6))
#Creating the box plot
sns.boxplot(x='day_of_the_week',
            y='delivery_time',
            data=df)
#Add specifications to the chart
plt.title('Delivery Time by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Delivery Time (minutes)')
plt.grid(axis='y')
plt.show()
```

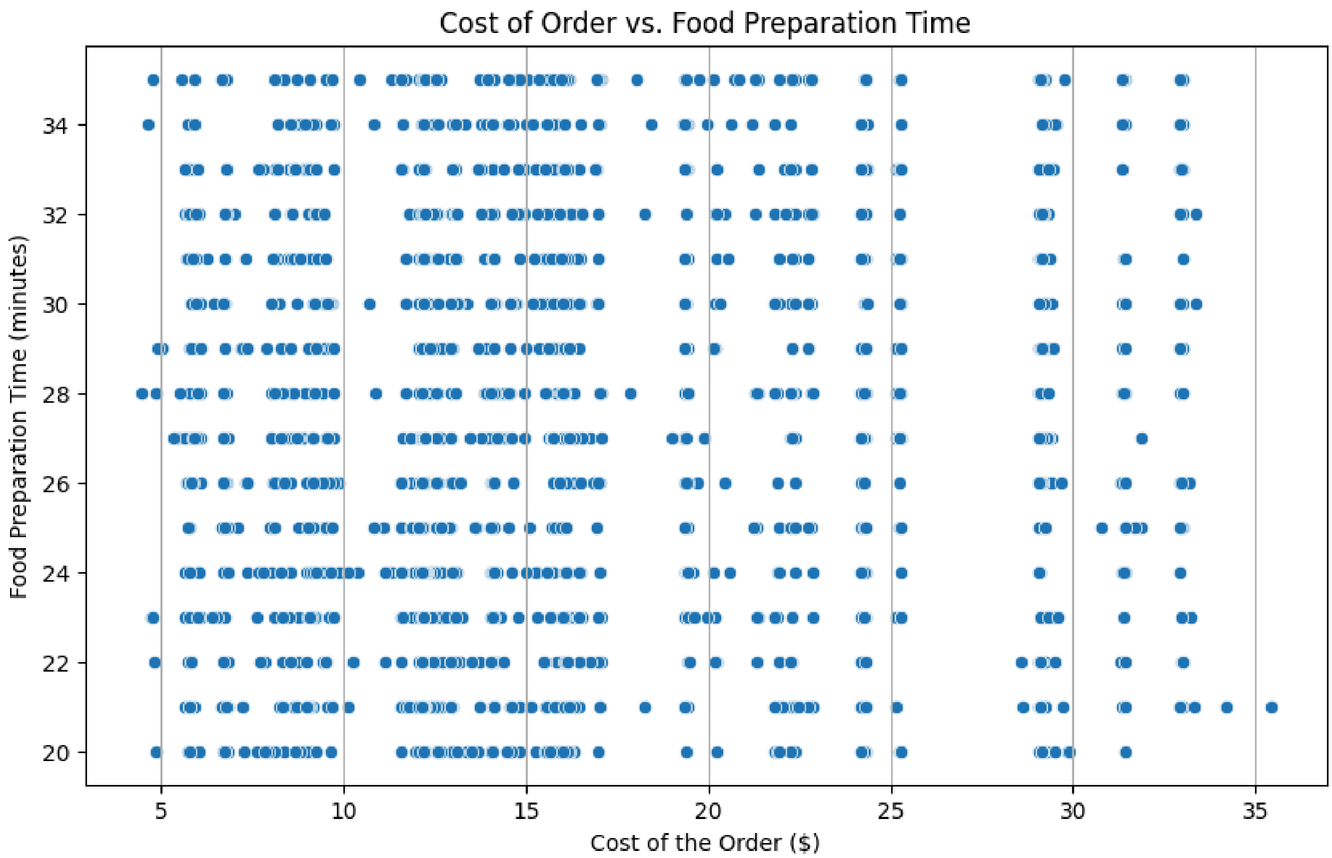



```
# Statistical summary of the delivery time column grouped by the day_of_the_week
# column values
df.groupby('day_of_the_week')['delivery_time'].describe()
```

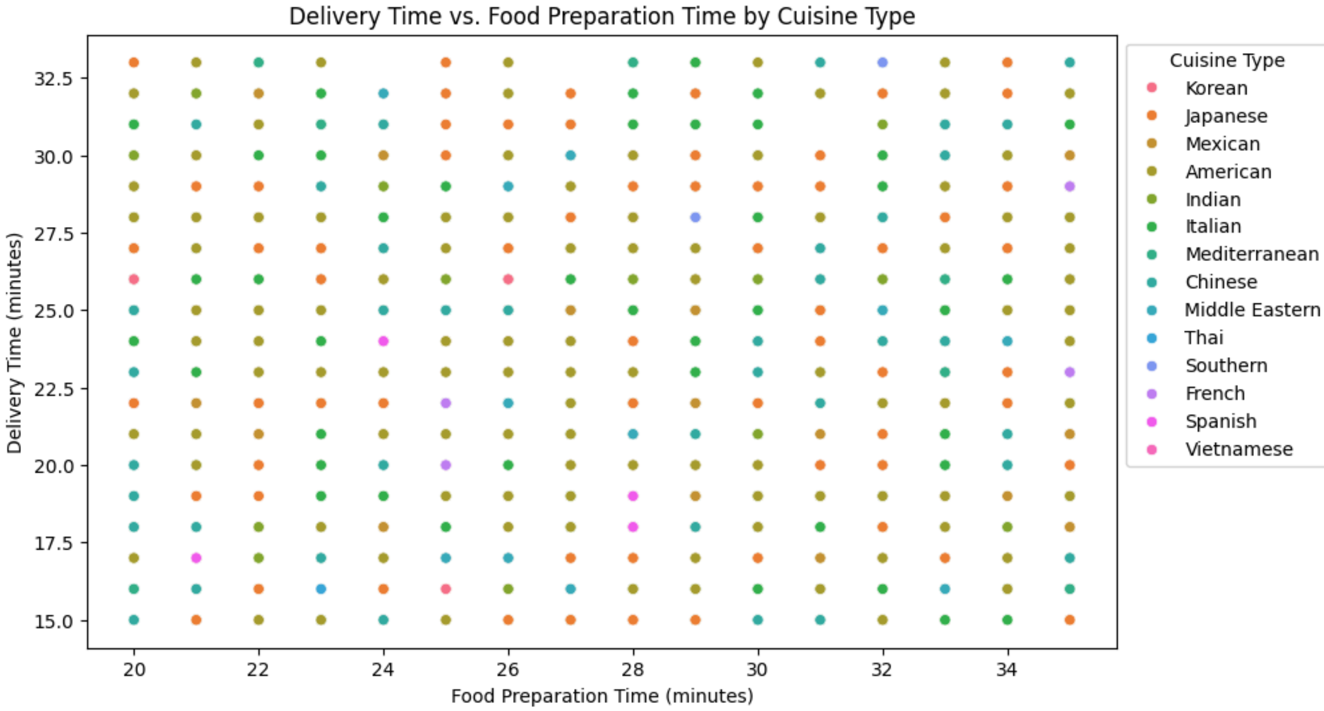
	count	mean	std	min	25%	50%	75%	max
day_of_the_week								
Weekday	547.0	28.340037	2.891428	24.0	26.0	28.0	31.0	33.0
Weekend	1351.0	22.470022	4.628938	15.0	18.5	22.0	27.0	30.0

```
plt.figure(figsize=(10, 6))
#Creating the scatterplot
sns.scatterplot(x='cost_of_the_order',
                y='food_preparation_time',
                data=df)
plt.title('Cost of Order vs. Food Preparation Time')
plt.xlabel('Cost of the Order ($)')
plt.ylabel('Food Preparation Time (minutes)')
```

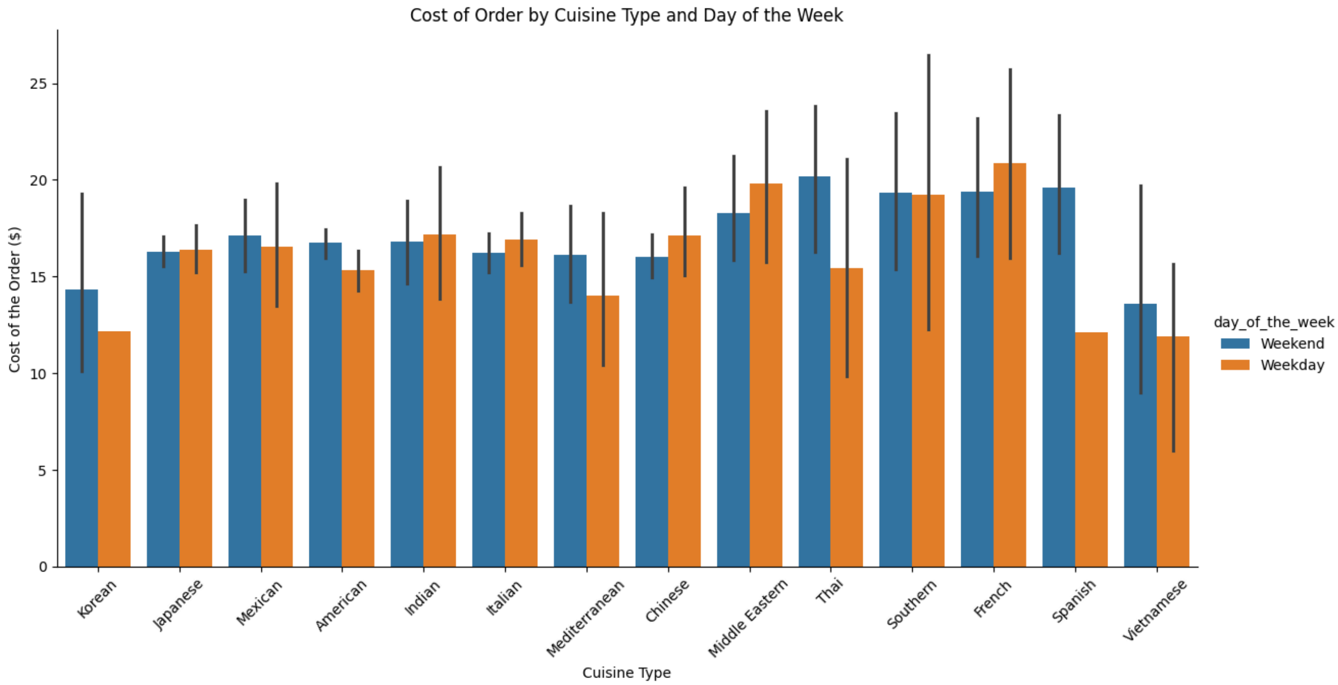
```
plt.grid(axis='x')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='food_preparation_time',
                y='delivery_time',
                hue='cuisine_type',
                data=df)
plt.title('Delivery Time vs. Food Preparation Time by Cuisine Type')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Delivery Time (minutes)')
#Using legend specifics for multivariate analysis clarity
plt.legend(title='Cuisine Type', bbox_to_anchor=(1, 1))
plt.show()
```

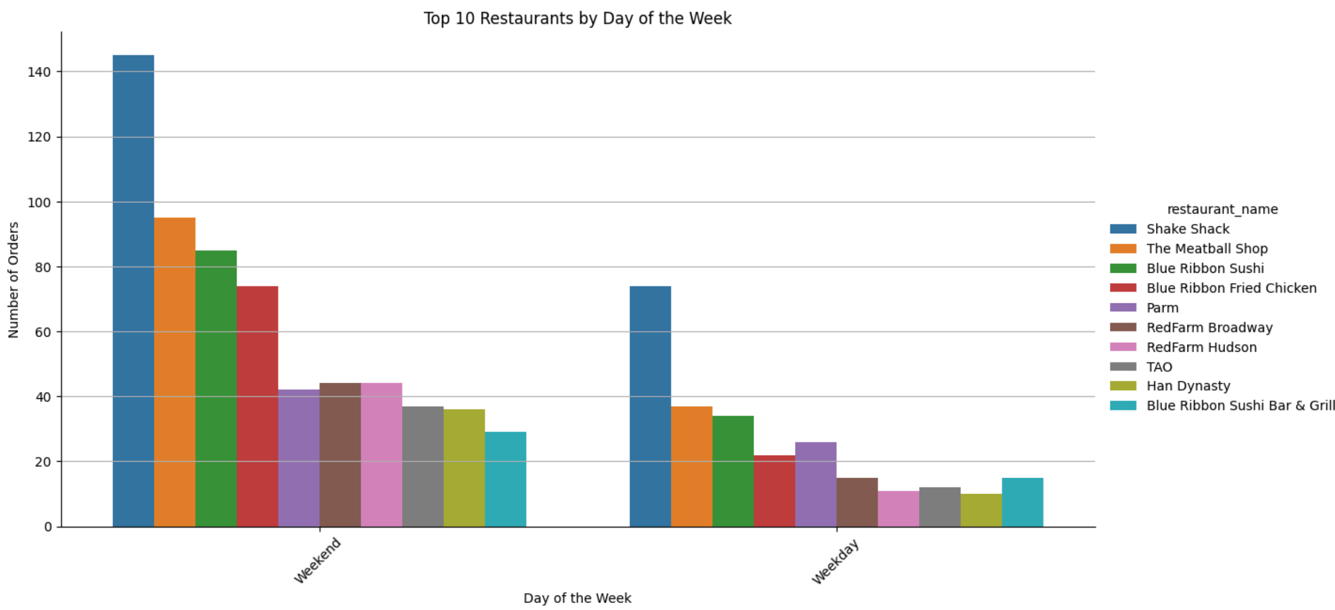



```
sns.catplot(x='cuisine_type',
            y='cost_of_the_order',
            hue='day_of_the_week',
            data=df,
            kind='bar',
            #To spread the plot vertically
            height=6,
            #To spread the plot horizontally
            aspect=2)
plt.title('Cost of Order by Cuisine Type and Day of the Week')
plt.xlabel('Cuisine Type')
plt.ylabel('Cost of the Order ($)')
plt.xticks(rotation=45)
plt.show()
```



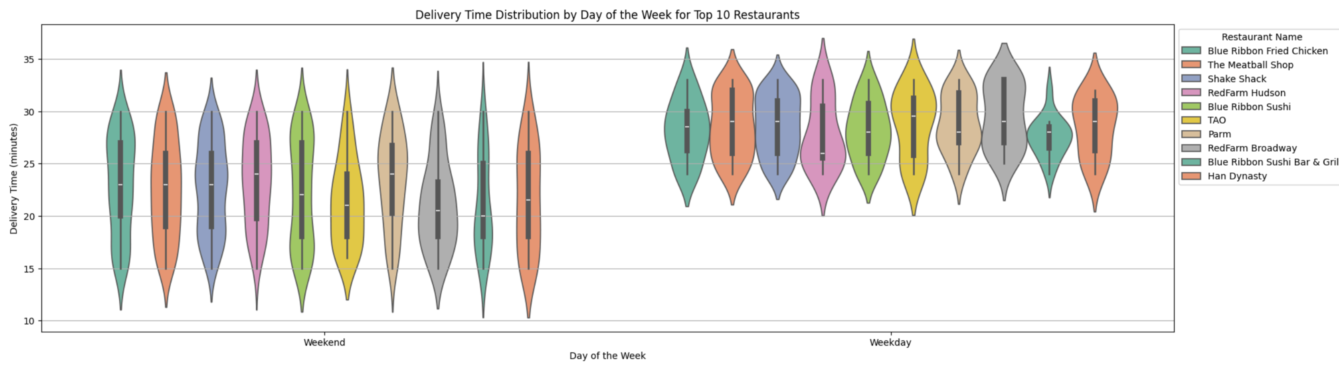

```
#Creating a value to extract only the restaurant name(index) from the top ten
#restaurants with regards to frequency in the dataset
top_restaurants = df['restaurant_name'].value_counts().head(10).index

#Creating a new dataframe that includes all of the data from our original df but
#only for the 'restaurant_name's that are within the top 10 value count
df_top_restaurants = df[df['restaurant_name'].isin(top_restaurants)]

sns.catplot(
    x='day_of_the_week',
    hue='restaurant_name',
    data=df_top_restaurants,
    kind='count',
```

```
    height=6,
    aspect=2,
    hue_order=top_restaurants)
plt.title('Top 10 Restaurants by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



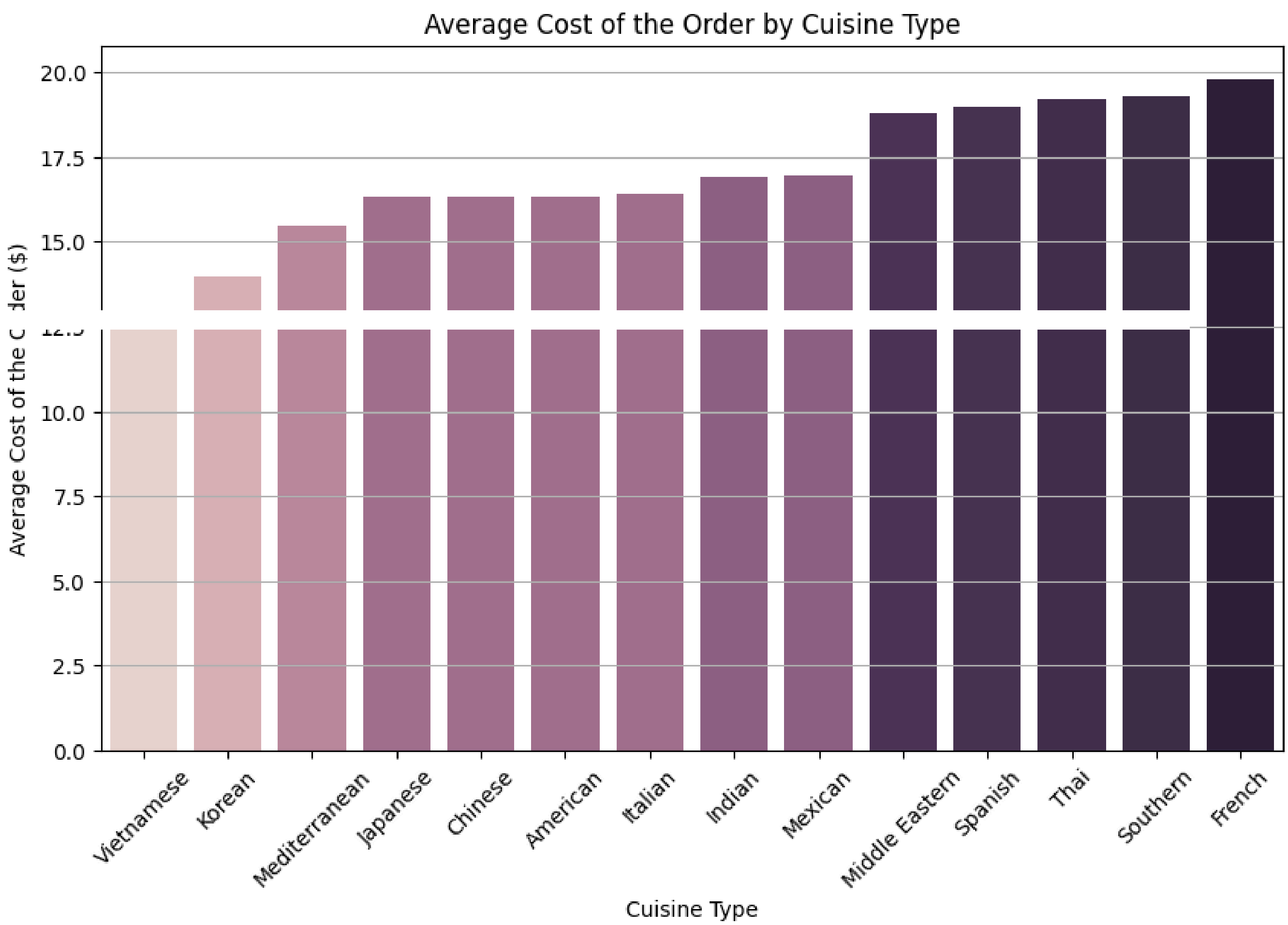

```
plt.figure(figsize=(22, 6))
sns.violinplot(
    x='day_of_the_week',
    y='delivery_time',
    #Using a dataframe created earlier to organize chart amongst the top 10
```

```
    #restaurants based on total order count
    hue=df[df['restaurant_name'].isin(top_restaurants)]['restaurant_name'],
    data=df,
    palette='Set2'
    )
plt.title('Delivery Time Distribution by Day of the Week for Top 10 Restaurants')
plt.xlabel('Day of the Week')
plt.ylabel('Delivery Time (minutes)')
plt.legend(title='Restaurant Name', bbox_to_anchor=(1, 1))
plt.grid(axis='y')
plt.show()
```



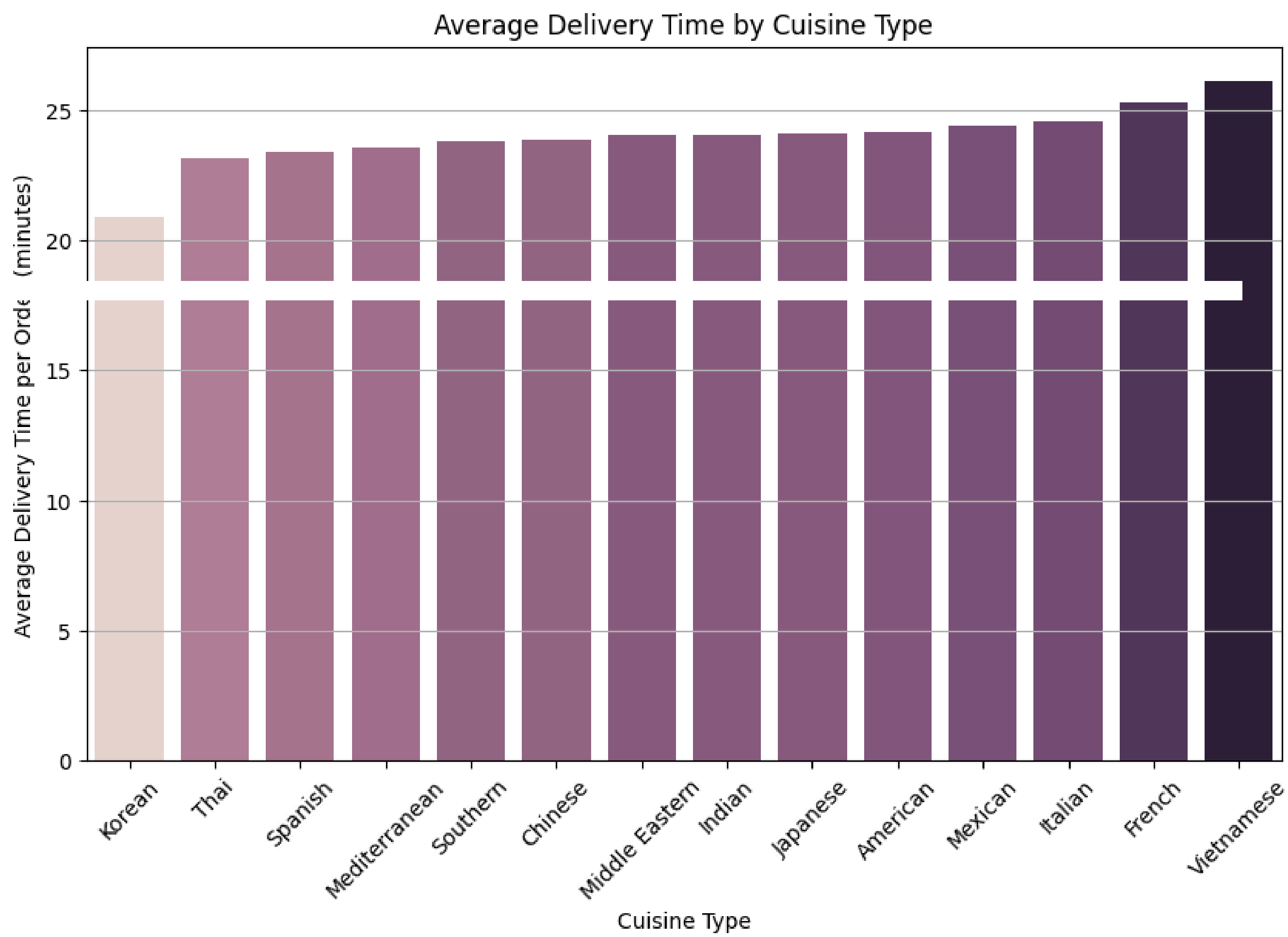

```
#Creating a value to compute the average cost of ordering from each cuisine and
#sorting them in ascending order
avg_cost_by_cuisine = df.groupby('cuisine_type')['cost_of_the_order'].mean().sort_va

plt.figure(figsize=(10, 6))
sns.barplot(
    x=avg_cost_by_cuisine.index,
    y=avg_cost_by_cuisine.values,
    hue=avg_cost_by_cuisine,
```

```
    legend=False
)
plt.title('Average Cost of the Order by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Cost of the Order ($)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



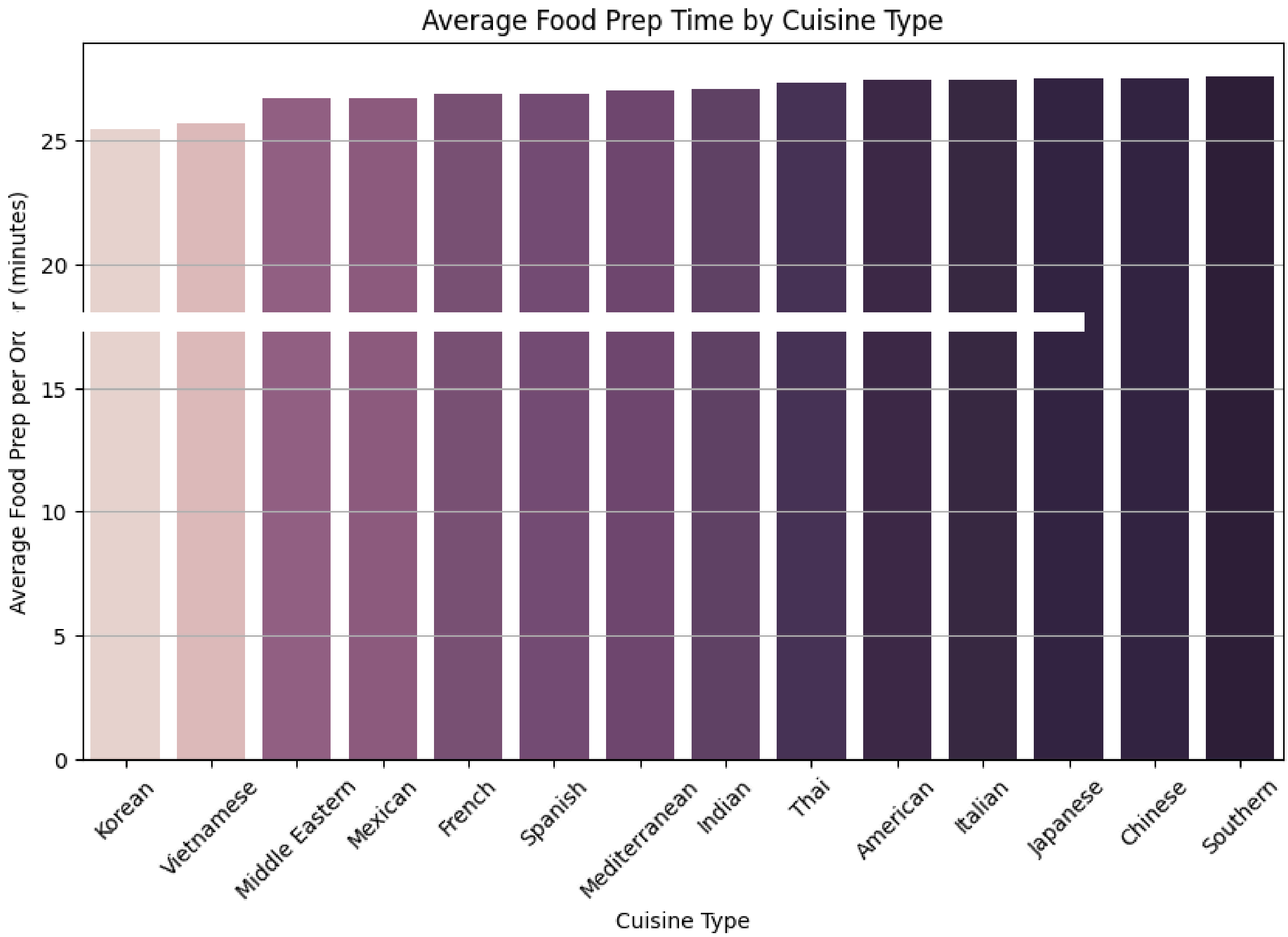

```
avg_dtime_by_cuisine = df.groupby('cuisine_type')['delivery_time'].mean().sort_value

plt.figure(figsize=(10, 6))
sns.barplot(
    x=avg_dtime_by_cuisine.index,
    y=avg_dtime_by_cuisine.values,
    hue=avg_dtime_by_cuisine,
    legend=False
```

```
)
plt.title('Average Delivery Time by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Delivery Time per Order (minutes)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



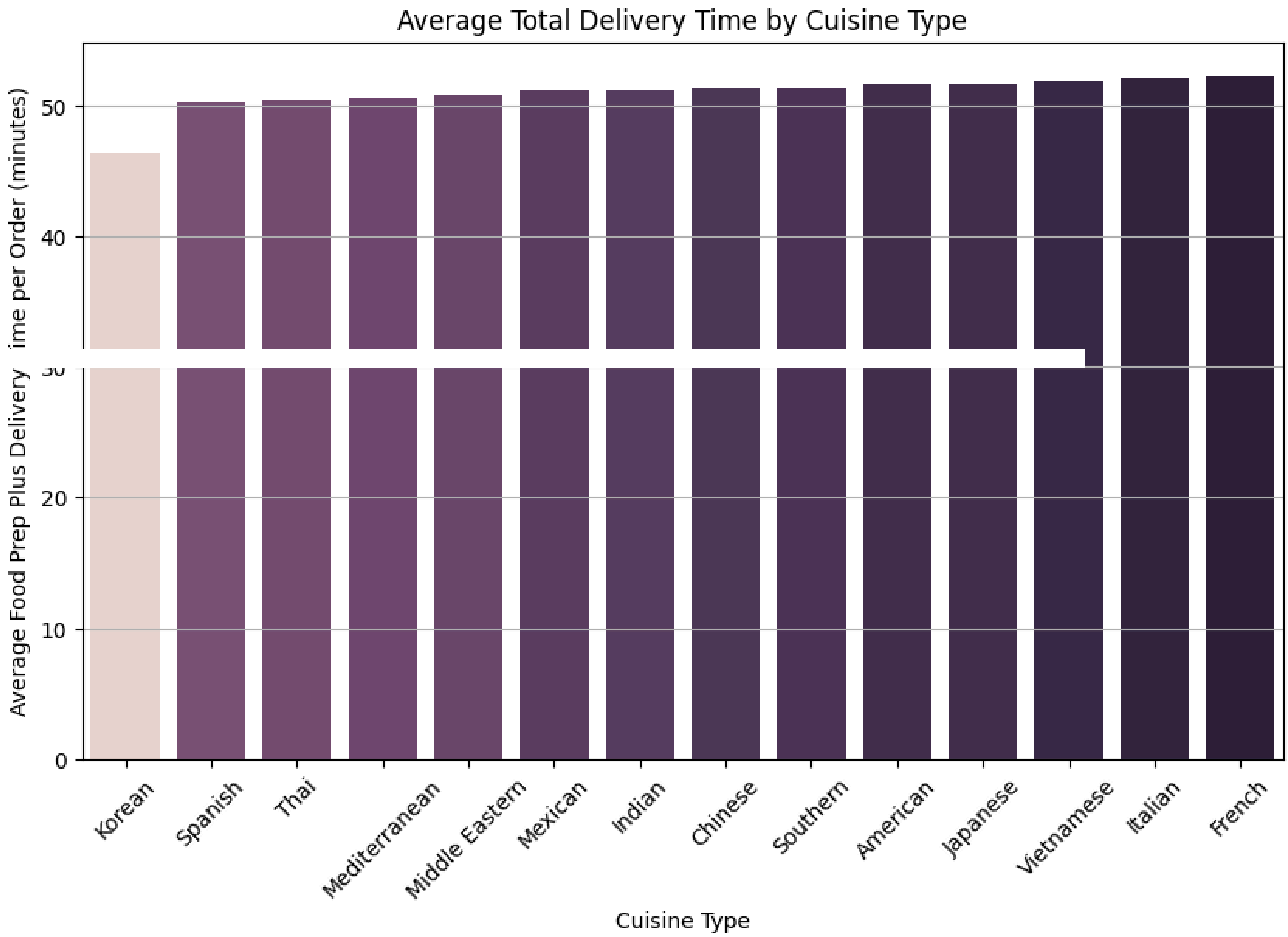

```
avg_preptime_by_cuisine = df.groupby('cuisine_type')['food_preparation_time'].mean()

plt.figure(figsize=(10, 6))
sns.barplot(
    x=avg_preptime_by_cuisine.index,
    y=avg_preptime_by_cuisine.values,
    hue=avg_preptime_by_cuisine,
    legend=False
)
```

```
plt.title('Average Food Prep Time by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Food Prep per Order (minutes)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



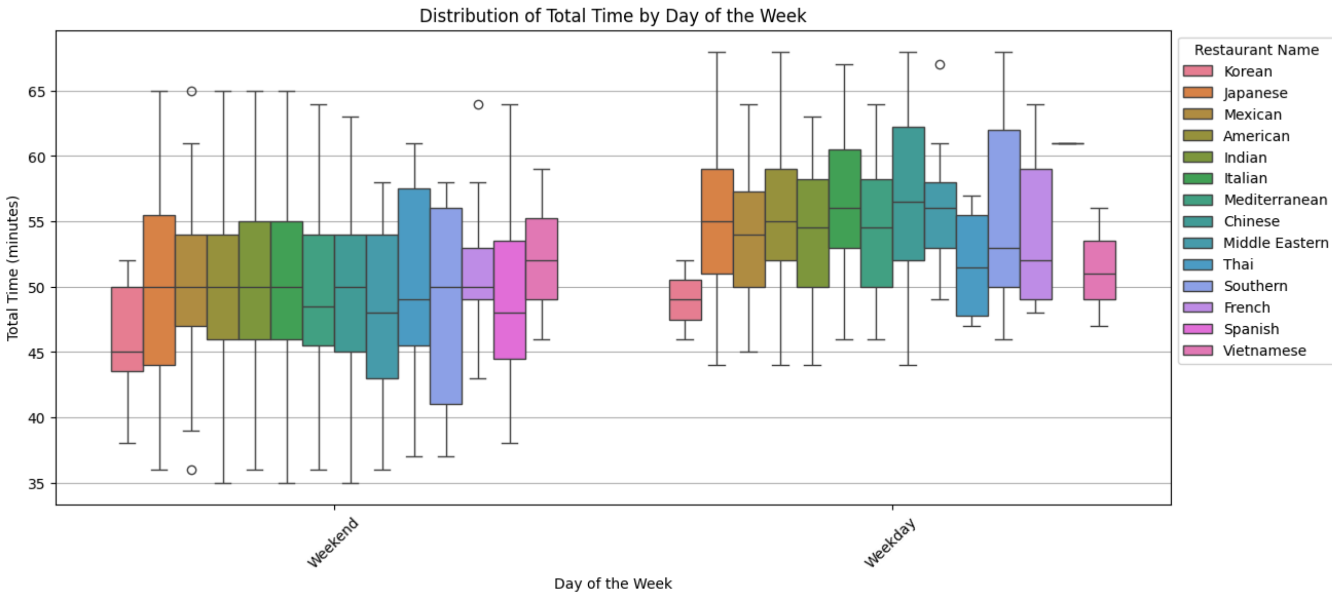

```
avg_dtime_by_cuisine = df.groupby('cuisine_type')['delivery_time'].mean().sort_value
avg_preptime_by_cuisine = df.groupby('cuisine_type')['food_preparation_time'].mean()
total_time_by_cuisine = (avg_dtime_by_cuisine + avg_preptime_by_cuisine).sort_values

plt.figure(figsize=(10, 6))
sns.barplot(
    x=total_time_by_cuisine.index,
    y=total_time_by_cuisine.values,
    hue=total_time_by_cuisine,
    legend=False
```

```
)
plt.title('Average Total Delivery Time by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Food Prep Plus Delivery Time per Order (minutes)')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```




```
#creating a total_time value within our df to call upon
df['total_time'] = df['delivery_time'] + df['food_preparation_time']

plt.figure(figsize=(14, 6))
sns.boxplot(
    x='day_of_the_week',
    y='total_time',
    data=df,
    hue='cuisine_type')
```

```
plt.title('Distribution of Total Time by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Total Time (minutes)')
plt.xticks(rotation=45)
plt.legend(title='Restaurant Name', bbox_to_anchor=(1, 1))
plt.grid(axis='y')
plt.show()
```




```
df.groupby('cuisine_type')['total_time'].describe()
```

cuisine_type	count	mean	std	min	25%	50%	75%	max
American	584.0	51.633562	6.616107	35.0	47.00	51.0	56.00	68.0
Chinese	215.0	51.367442	7.362437	35.0	46.00	51.0	56.50	68.0
French	18.0	52.222222	6.025002	43.0	49.00	50.5	56.75	64.0
Indian	73.0	51.191781	6.520603	36.0	47.00	51.0	56.00	65.0
Italian	298.0	52.050336	6.764496	35.0	47.25	53.0	56.00	67.0
Japanese	470.0	51.642553	6.993440	36.0	47.00	52.0	56.00	68.0
Korean	13.0	46.384615	4.519190	38.0	44.00	45.0	51.00	52.0
Mediterranean	46.0	50.586957	7.206713	36.0	46.25	50.0	55.00	64.0
Mexican	77.0	51.116883	5.891506	36.0	48.00	51.0	55.00	65.0
Middle Eastern	49.0	50.755102	7.221296	36.0	47.00	53.0	56.00	67.0
Southern	17.0	51.411765	8.602753	37.0	46.00	53.0	56.00	68.0
Spanish	12.0	50.333333	8.060378	38.0	45.25	49.0	57.25	64.0
Thai	19.0	50.473684	7.152066	37.0	47.00	49.0	57.00	61.0
Vietnamese	7.0	51.857143	4.740906	46.0	48.50	51.0	55.00	59.0

```
correlation_matrix = df[['rating', 'cost_of_the_order', 'delivery_time', 'food_prepa

print(correlation_matrix)
```

```
                         rating  cost_of_the_order  delivery_time  \
rating                 1.000000           0.033983      -0.009804
cost_of_the_order      0.033983           1.000000      -0.029949
delivery_time         -0.009804          -0.029949       1.000000
food_preparation_time -0.006083           0.041527       0.011094
total_time            -0.011348           0.006358       0.735195

                       food_preparation_time  total_time
rating                             -0.006083   -0.011348
cost_of_the_order                   0.041527    0.006358
delivery_time                       0.011094    0.735195
food_preparation_time               1.000000    0.685970
total_time                          0.685970    1.000000
```

```
#taking the mean of all costs of orders and ranking them
#in descending order
top_10_restaurants = df.groupby('restaurant_name')['cost_of_the_order'].mean().sort_
print(top_10_restaurants)
```

```
restaurant_name
Kambi Ramen House      32.930000
Emporio                31.430000
Bhatti Indian Grill    31.115000
Haru Gramercy Park     29.830000
Lucky Strike           29.250000
Il Bambino             29.250000
Sarabeth's             29.133333
Rohm Thai              29.100000
Klong                  29.050000
67 Burger              29.050000
Name: cost_of_the_order, dtype: float64
```

```
#pythonic version with nlargest function I just found
top_10_restaurants = df.groupby('restaurant_name')['cost_of_the_order'].mean().nlarg
print(top_10_restaurants)
```

```
restaurant_name
Kambi Ramen House      32.930000
Emporio                31.430000
Bhatti Indian Grill    31.115000
Haru Gramercy Park     29.830000
Il Bambino             29.250000
Lucky Strike           29.250000
Sarabeth's             29.133333
Rohm Thai              29.100000
67 Burger              29.050000
Klong                  29.050000
Name: cost_of_the_order, dtype: float64
```

```
df['restaurant_name'].value_counts().head(15)
```

restaurant_name	count
Shake Shack	219
The Meatball Shop	132
Blue Ribbon Sushi	119
Blue Ribbon Fried Chicken	96
Parm	68
RedFarm Broadway	59
RedFarm Hudson	55
TAO	49
Han Dynasty	46
Blue Ribbon Sushi Bar & Grill	44
Nobu Next Door	42
Rubirosa	37
Sushi of Gari 46	37
Momoya	30
Five Guys Burgers and Fries	29

dtype: int64

Observations:

1. **Delivery times during the weekday are an evident problem. While there are almost triple the amount of deliveries on the weekend versus the weekdays, the weekend's delivery time is nearly 6 minutes faster, on average, than during the week. While the spread for delivery times are much higher on the weekend, ranging from 15 to 30 minutes, the weekday delivery times are consistently slower, ranging from 24 to 33 minutes.**
2. Cost of the order does not seem to have any influence on food prep time. There is no helpful information to disclose here to FoodHub or the customer.
3. Cuisine type does not seem to have a correlation with food preparation time and delivery time as they are scattered across the board.
4. The cost of order by cuisine type based on the day of the week shows a relatively even balance as some cuisine types average more expensive orders during the week while others average more expensive orders on the weekends. No particular cuisine jumps out as having

the most expensive order the most often. **In general, this statistic is helpful in pointing out that the delivery times are more volatile than the cost of the orders when comparing them to the day of the week.**

5. The top 10 most ordered from restaurants remains relatively the same in distribution from weekday to weekend, and **the weekend shows consistently more orders for nearly all of the top 10 restaurants.**
6. The top 10 most ordered from restaurants against the frequency of delivery times on weekdays versus weekends highlights what we've already known in that the weekdays have an issue with long delivery times. While all of the top 10 restaurants reach into the same delivery times on the weekends as they do on the weekdays, **the frequency for long delivery times are consistently higher for all of the top 10 during the weekdays.**
7. Average cost of order demonstrates which cuisines typically have higher/lower prices. These could be used for marketing purposes for people on a budget vs. people that want to splurge.
8. Average delivery (part 1)
9. Average prep (part 2)
10. Total time (1+2) demonstrates an interesting statistic that nearly all cuisine types average just above 50 minutes for the total time of order to food prep to completed delivery.
11. Unsurprisingly here, the distribution of total delivery time for each cuisine against the day of the week demonstrates the issue of longer delivery times on the weekdays. Every single cuisine lags in this department except for Vietnamese food which seems to have very slightly better delivery times on the weekday than the weekend.
12. Lastly, I printed a correlation matrix with the 'rating' column to see if we could gain any insight from it. With so many missing values this proved to be unhelpful.
13. I got a last minute thought to check the top 10 highest costs of order on average against the top 10 most popular restauarants, but there are no restaurants in both categories.

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]

```
# First, group by restaurant_name and calculate the rating count and average rating
restaurant_ratings = df.groupby('restaurant_name')['rating'].agg(['count', 'mean'])
```

```
# Now, filter the restaurants that have more than 50 ratings and average rating gre
promotional_restaurants = restaurant_ratings[(restaurant_ratings['count'] > 50) & (
```