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

In [ ]:
```
# 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 -
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 52.0 MB/s eta 0:00:
00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.1/12.1 MB 45.9 MB/s eta 0:00:
00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.8/294.8 kB 16.0 MB/s eta 0:0
0:00
  WARNING: The scripts f2py, f2py3 and f2py3.10 are installed in '/root/.loc
al/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
warning, use --no-warn-script-location.
ERROR: pip's dependency resolver does not currently take into account all th
e packages that are installed. This behaviour is the source of the following
dependency conflicts.
cudf-cu12 24.6.1 requires pandas<2.2.3dev0,>=2.0, but you have pandas 1.5.3
which is incompatible.
google-colab 1.0.0 requires pandas==2.2.2, but you have pandas 1.5.3 which i
s incompatible.
mizani 0.11.4 requires pandas>=2.1.0, but you have pandas 1.5.3 which is inc
ompatible.
plotnine 0.13.6 requires pandas<3.0.0,>=2.1.0, but you have pandas 1.5.3 whi
ch is incompatible.
xarray 2024.9.0 requires pandas>=2.1, but you have pandas 1.5.3 which is inc
ompatible.
```

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

In [ ]:
```
# import libraries for data manipulation
import numpy as np
import pandas as pd

# import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [ ]:
```
# mount google drive to access files
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

Understanding the structure of the data

In [ ]:
```
# Creating a dataframe with the data read in from the customer's data
data = pd.read_csv('/content/drive/MyDrive/uoftaiml/proj1/foodhub_order.csv'
```

In [ ]:
```
# Creating a copy of the dataframe to avoid changes on original file
df = data.copy()
```

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

Out[ ]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	da
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	

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

In [ ]:
```
# Rows and Columns present in data
df.shape
```

Out[ ]: (1898, 9)

Observations:

There are 1,898 rows and 9 columns in the data, for a total of 17,082 records

Question 2: What are the datatypes of the different columns in the dataset? [0.5 mark]

In [ ]:
```
# Displaying the datatypes of columns in the dataset
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:

Of the 9 columns, there are 4 integer, 4 object and 1 float datatypes

Type object: restaurant_name, cuisine_type, day_of_the_week and rating

Type integer: order_id, customer_id, food_preparation_time and delivery_time

Type float: cost_of_the_order

For the purposes of this analysis, I am removing the order_id and customer_id columns as this is more a general analysis of demand at restaurants rather than a customer-level analysis. I would change the datatypes to object for those columns if they were included in the analysis, because these variables are more categorical in nature. I considered changing day_of_the_week to a date/time datatype but the information contained in that column is text/object related. The other columns are of the appropriate datatype for this analysis.

In [ ]:
```
# Code to remove the order_id and customer_id columns
df.drop(['order_id', 'customer_id'], axis=1, inplace=True)
```

In [ ]:
```
# Code to verify that those columns have been dropped
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   restaurant_name        1898 non-null   object
 1   cuisine_type           1898 non-null   object
 2   cost_of_the_order      1898 non-null   float64
 3   day_of_the_week        1898 non-null   object
 4   rating                 1898 non-null   object
 5   food_preparation_time  1898 non-null   int64
 6   delivery_time          1898 non-null   int64
dtypes: float64(1), int64(2), object(4)
memory usage: 103.9+ KB
```

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

It does not appear that there are missing values as each column has the same amount of rows with no null values. However, the rating column appears to have values of "Not Given", which can be considered a valid response to a rating question and will be considered part of the analysis. Rating should be changed to int to aggregate on the numerical values, although Not Given could be excluded or changed to int. It appears that text values are formatted properly.

In [ ]:
```
# Checking for missing values in the data
df.isnull().sum()
```

Out[ ]:

	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

In [ ]:
```
# check for string inconsistencies in the dataframe

for column in df.select_dtypes(include=['object']):
  print(f"Unique values in {column}: {df[column].unique()}")
```

```
Unique values in restaurant_name: ['Hangawi' 'Blue Ribbon Sushi Izakaya' 'Ca
fe Habana'
 'Blue Ribbon Fried Chicken' 'Dirty Bird to Go' 'Tamarind TriBeCa'
 'The Meatball Shop' 'Barbounia' 'Anjappar Chettinad' 'Bukhara Grill'
 'Big Wong Restaurant \x8c_¤¾Ñ¼' 'Empanada Mama (closed)' 'Pylos'
 "Lucky's Famous Burgers" 'Shake Shack' 'Sushi of Gari' 'RedFarm Hudson'
 'Blue Ribbon Sushi' 'Five Guys Burgers and Fries' 'Tortaria'
 'Cafe Mogador' 'Otto Enoteca Pizzeria' 'Vezzo Thin Crust Pizza'
 'Sushi of Gari 46' 'The Kati Roll Company' 'Klong' '5 Napkin Burger'
 'TAO' 'Parm' 'Sushi Samba' 'Haru Gramercy Park'
 'Chipotle Mexican Grill $1.99 Delivery' 'RedFarm Broadway' 'Cafeteria'
 'DuMont Burger' "Sarabeth's East" 'Hill Country Fried Chicken' 'Bistango'
 "Jack's Wife Freda" "Mamoun's Falafel" 'Prosperity Dumpling'
 'Blue Ribbon Sushi Bar & Grill' 'Westville Hudson' 'Blue Ribbon Brooklyn'
 'Nobu Next Door' 'Osteria Morini' 'Haandi' 'Benihana' 'Han Dynasty'
 'Chote Nawab' 'Mission Cantina' "Xi'an Famous Foods" 'Rubirosa'
 "Joe's Shanghai \x8e_À\x8eü£¾÷´" 'Bareburger' 'The Odeon' 'Pongsri Thai'
 'Yama Japanese Restaurant' 'Momoya' 'Balthazar Boulangerie' 'CafÌ© China'
 'Boqueria' 'Song Thai Restaurant & Bar' 'Five Leaves'
 'Pinto Nouveau Thai Bistro' "Amy Ruth's" 'Pepe Giallo' 'indikitch'
 'Yama 49' 'Piccolo Angolo' 'Pepe Rosso To Go' "L'Express" 'Amma'
 'Delicatessen' "S'MAC" "Vanessa's Dumplings" 'Bhatti Indian Grill'
 'Taro Sushi' 'Donburi-ya' 'Hatsuhana' 'Samurai Mama' 'Waverly Diner'
 'Tarallucci e Vino Restaurant' "P.J. Clarke's" 'Lantern Thai Kitchen'
 'ilili Restaurant' 'The Smile' "Vanessa's Dumpling House" "Bubby's "
 'Woorijip' 'Dirty Bird To Go (archived)' 'Haveli Indian Restaurant'
 'Dos Caminos' 'da Umberto' 'Sushi of Gari Tribeca' 'Burger Joint'
 'Room Service' "Sarabeth's Restaurant" 'Xe May Sandwich Shop' 'Hibino'
 'Mira Sushi' 'Melt Shop' 'J. G. Melon' 'Hummus Place' 'Saravanaa Bhavan'
 'Friend of a Farmer' 'The Loop' 'Balade' 'Posto' 'Terakawa Ramen'
 'Kambi Ramen House' 'Wo Hop Restaurant' 'Spice Thai'
 "Dickson's Farmstand Meats" 'UVA Wine Bar & Restaurant'
 'Serafina Fabulous Pizza' 'Gaia Italian Cafe'
 'Chola Eclectic Indian Cuisine' 'Hot Kitchen' 'Junoon'
 'Ravagh Persian Grill' 'Rohm Thai' 'Dig Inn Seasonal Market' 'Olea'
 'Cho Dang Gol' 'El Parador Cafe' 'Socarrat Paella Bar'
 "Don's Bogam BBQ & Wine Bar" 'Alidoro' "Tony's Di Napoli"
 'Cipriani Le Specialita' 'Sushi Choshi' 'Kanoyama' 'V-Nam Cafe'
 'Zero Otto Nove' 'Dos Caminos Soho' 'Go! Go! Curry!' 'La Follia'
 'Izakaya Ten' '12 Chairs' 'Philippe Chow' 'The MasalaWala' 'brgr'
 "Carmine's" 'Asuka Sushi' 'Aurora' "Sarabeth's" 'Crema Restaurante'
 "Big Daddy's" 'Moonstruck on Second' 'Cafe de La Esquina' 'Olive Garden'
 '67 Burger' 'Tres Carnes' "Schnipper's Quality Kitchen" 'Nha Trang One'
 'Market Table' 'Galli Restaurant' 'Hampton Chutney Co.'
 'Byblos Restaurant' 'Grand Sichuan International' 'Le Grainne Cafe'
 'Il Bambino' 'Kori Restaurant and Bar' 'DespaÌ±a' 'Lamarca Pasta'
 'Lucky Strike' "Paul & Jimmy's" 'Hunan Manor' "Coppola's East" 'Emporio'
 'Wa Jeal' 'Le Zie 2000 Trattoria' 'Rye House' "Hiroko's Place"
 'Frank Restaurant' "Sarabeth's West" "'wichcraft"]
Unique values in cuisine_type: ['Korean' 'Japanese' 'Mexican' 'American' 'In
dian' 'Italian'
 'Mediterranean' 'Chinese' 'Middle Eastern' 'Thai' 'Southern' 'French'
 'Spanish' 'Vietnamese']
Unique values in day_of_the_week: ['Weekend' 'Weekday']
Unique values in rating: ['Not given' '5' '3' '4']
```

In [ ]:
```
#  check for duplicates in the data
df.duplicated().sum()
```

Out[ ]:
```
0
```

Observations:

The code has confirmed that there is no missing data

There appears to be extra characters in two of the restaurant names, which I will confirm whether those should be included in the dataset.

Discovered later in the analysis, there are two restaurants that appear to be duplicates and are in the top 10 of restaurants and should be combined before analysis proceeds. Should also check for other duplicate values in the data and address.

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 [ ]:
```
# Checking the statistical summary of numerical columns
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%
cost_of_the_order	1898.0	16.498851	7.483812	4.47	12.08	14.14	22.297
food_preparation_time	1898.0	27.371970	4.632481	20.00	23.00	27.00	31.000
delivery_time	1898.0	24.161749	4.972637	15.00	20.00	25.00	28.000

Observations:

The minimum, average and maximum time it takes for food to be prepared are 20 minutes, approximately 27 minutes and 35 minutes respectively.

In [ ]:
```
# Checking the statistical summary of the categorical columns and proportion
df.describe(include='object').T
```

Out[ ]:

	count	unique	top	freq
restaurant_name	1898	178	Shake Shack	219
cuisine_type	1898	14	American	584
day_of_the_week	1898	2	Weekend	1351
rating	1898	4	Not given	736

Observations

Shake Shack has the most occurrences of the restaurants, American cuisine is the most popular, restaurants are busiest on the weekends and Not Given appears the most for ratings.

In [ ]:
```
df['day_of_the_week'].value_counts(normalize=True)
```

Out[ ]:

day_of_the_week	proportion
Weekend	0.711802
Weekday	0.288198

dtype: float64

Observations

The weekend makes up over 70% of business

In [ ]:
```
df['rating'].value_counts(normalize=True)
```

Out[ ]:

rating	proportion
Not given	0.387777
5	0.309800
4	0.203372
3	0.099052

dtype: float64

Observations

Not given makes up approximately 40% of the ratings, followed by 5's with 30%, 4's with 20% and 3's with around 10%. There are no 1's or 2's or other numbers in the dataset.

Note:

There are 178 restaurants in the dataset, I chose to focus on the top 10 in sales, orders initially for insights.

In [ ]:
```
# List the top 10 restaurants by proportion of sales

# Calculate the proportion of orders for each restaurant
```

```
restaurant_proportions = df['restaurant_name'].value_counts(normalize=True)

# Get the top 10 restaurants by proportion
top_10_restaurants = restaurant_proportions.head(10)

# Print the top 10 restaurants and their proportion of sales
top_10_restaurants
```

Out[ ]:

restaurant_name	proportion
Shake Shack	0.115385
The Meatball Shop	0.069547
Blue Ribbon Sushi	0.062698
Blue Ribbon Fried Chicken	0.050580
Parm	0.035827
RedFarm Broadway	0.031085
RedFarm Hudson	0.028978
TAO	0.025817
Han Dynasty	0.024236
Blue Ribbon Sushi Bar & Grill	0.023182

dtype: float64

Observations

Shake Shack has 12% of all sales with the rest of the top 10 accounting for 2 - 7% of sales.

Blue Ribbon Sushi appears to duplicated in the restaurant name and would like to confirm with domain expert before combining or merging.

In [ ]:
```
# list the top 10 restaurants by frequency in sales

# Group by restaurant name and count the number of orders
restaurant_counts = df.groupby('restaurant_name')['restaurant_name'].count()

# Sort the counts in descending order and get the top 10
top_10_restaurants = restaurant_counts.sort_values(ascending=False).head(10)

top_10_restaurants
```

Out[ ]:

restaurant_name	restaurant_name
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

dtype: int64

Observations

Of the 1898 orders, Shack Shack had the most at 219. The top 4 restaurants in orders accounted for approximately 20% of orders.

In [ ]:
```
# what are the cuisines by proportion
df['cuisine_type'].value_counts(normalize=True)
```

Out[ ]:

cuisine_type	proportion
American	0.307692
Japanese	0.247629
Italian	0.157007
Chinese	0.113277
Mexican	0.040569
Indian	0.038462
Middle Eastern	0.025817
Mediterranean	0.024236
Thai	0.010011
French	0.009484
Southern	0.008957
Korean	0.006849
Spanish	0.006322
Vietnamese	0.003688

dtype: float64

Observations

American cuisine is the most popular, accounting for over 30% of orders. The top 4 cuisines account for approximately 75% of orders.

In [ ]:

```
# what are the total sales and what is the proportion and amount that the to

# Calculate total sales
total_sales = df['cost_of_the_order'].sum()

# Group by restaurant name and sum the cost of the order
restaurant_sales = df.groupby('restaurant_name')['cost_of_the_order'].sum()

# Sort the sales in descending order and get the top 10
top_10_restaurant_sales = restaurant_sales.sort_values(ascending=False).head

# Calculate the proportion of sales for the top 10 restaurants
top_10_sales_proportion = top_10_restaurant_sales.sum() / total_sales

# Calculate the dollar amount of sales for the top 10 restaurants
top_10_sales_amount = top_10_restaurant_sales.sum()

# Print the total sales, proportion, and dollar amount contributed by the to
print(f"Total sales: ${total_sales:.2f}")
```

```
print(f"\nProportion of sales contributed by top 10 restaurants: {top_10_sal
print(f"Dollar amount of sales contributed by top 10 restaurants: ${top_10_s

# Calculate the proportion and dollar amount for each of the top 10 restaura
for restaurant, sales in top_10_restaurant_sales.items():
  proportion = sales / total_sales
  print(f"\n{restaurant}: Proportion: {proportion:.2%}, Dollar amount: ${sal
```

```
Total sales: $31314.82

Proportion of sales contributed by top 10 restaurants: 46.45%
Dollar amount of sales contributed by top 10 restaurants: $14546.49

Shake Shack: Proportion: 11.43%, Dollar amount: $3579.53

The Meatball Shop: Proportion: 6.85%, Dollar amount: $2145.21

Blue Ribbon Sushi: Proportion: 6.08%, Dollar amount: $1903.95

Blue Ribbon Fried Chicken: Proportion: 5.31%, Dollar amount: $1662.29

Parm: Proportion: 3.55%, Dollar amount: $1112.76

RedFarm Broadway: Proportion: 3.08%, Dollar amount: $965.13

RedFarm Hudson: Proportion: 2.94%, Dollar amount: $921.21

TAO: Proportion: 2.66%, Dollar amount: $834.50

Han Dynasty: Proportion: 2.41%, Dollar amount: $755.29

Blue Ribbon Sushi Bar & Grill: Proportion: 2.13%, Dollar amount: $666.62
```

Observations

The top 10 restaurants contributed over 45% of total sales, led by Shake Shack

Observations:

The minimum, average and maximum time it takes for food to be prepared are 20 minutes, approximately 27 minutes and 31 minutes respectively.

The cost of the orders range in value from around 5.00 to 35.00, with the majority of the values between 5.00 and 18.00, around $12.00 appearing the most frequently. Right skew

The preparation time ranged from 20 to 35 minutes, with an average time of 27 minutes. Symmetrical over a wide range.

The delivery time ranged from 15 to approximately 33 minutes, the average time being 25 minutes. Left skew

Of the categorical variables, the restaurant Shake Shack, American cuisine and weekend orders appeared most frequently. In this data, Shake Shack had 219 sales, accounting for almost 12% of sales, the weekend accounted for approximately 71% of sales, there are 4 cuisines that account for approximately 75% of sales (American, Japanese, Chinese and Italian in that order), and rating of Not Given appeared in approximately 40% of the data, followed by 5 in approximately 30% of the data.

Visually there are no outliers in the data. Would binning further help the analysis, expose outliers?

With 40% of the ratings being "Not Given" it brings into question how effective the analysis would be on this variable as is. Should the column be dropped or the values imputed using mode? Also the values of the ratings are incomplete though not missing (values = not given, 5,4,3--can it be assumed that out of 1898 orders there were no 1,2? Also, impute with mode will result in approximately 70% of the ratings being a 5; also begs the question whether this should be used in the analysis.

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

In [ ]:
```
# what are the count of orders that are rated as "Not Given"

not_given_count = df[df['rating'] == 'Not given'].shape[0]
print(f"The number of orders not rated is: {not_given_count}")
```

```
The number of orders not rated is: 736
```

Observations:

736 out of the 1898 orders or 40% of ratings are listed as not given, I question whether including rating in this analysis would be insightful as not given is not a rating per se.

Exploratory Data Analysis (EDA)

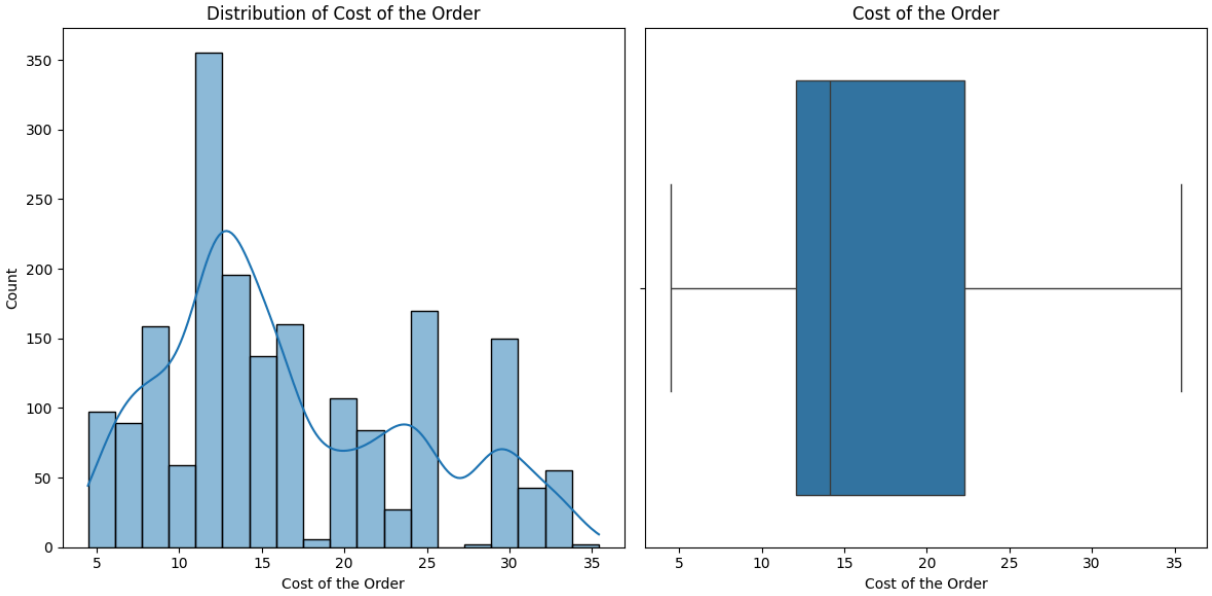
Univariate Analysis

Question 6: Explore all the variables and provide observations on their distributions. [9 marks]

Will first look at numerical distributions, using histograms, boxplots and density plots

In [ ]:
```
# a histplot and boxplot of the cost of the order column

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['cost_of_the_order'], kde=True)
```

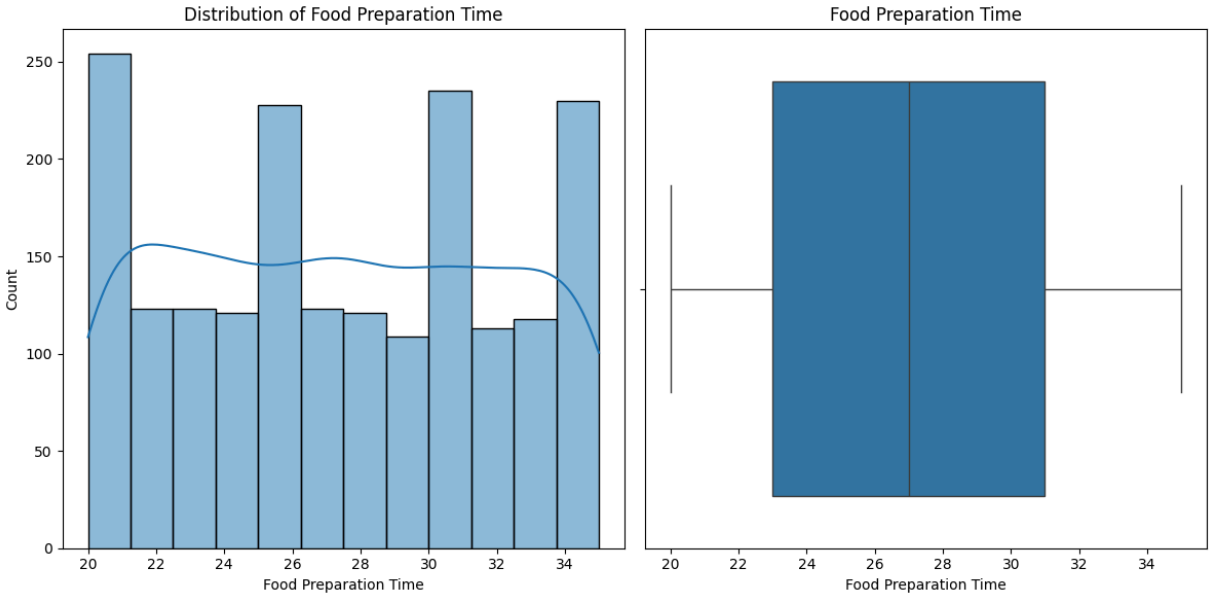
```
plt.title('Distribution of Cost of the Order')
plt.xlabel('Cost of the Order')

plt.subplot(1, 2, 2)
sns.boxplot(x=df['cost_of_the_order'])
plt.title('Cost of the Order')
plt.xlabel('Cost of the Order')

plt.tight_layout()
plt.show()
```



Observations

A majority of the orders are in the 5 - 20 range, most frequently around 12, with average price 14.

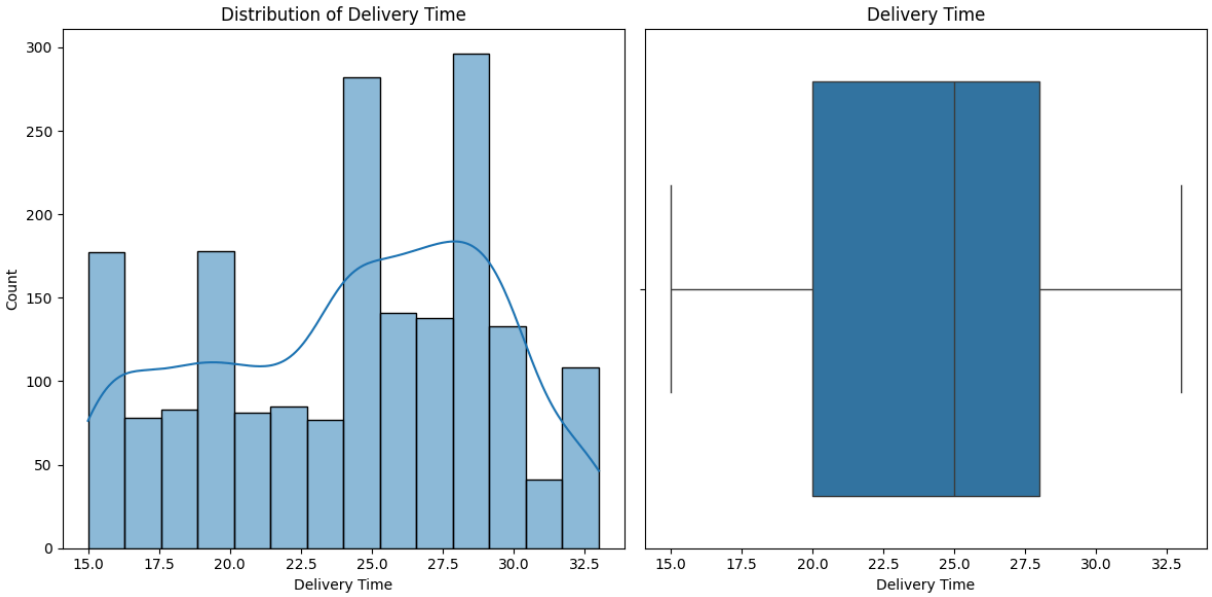
In [ ]:
```
# a histplot and boxplot of the food preparation time column

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['food_preparation_time'], kde=True)
plt.title('Distribution of Food Preparation Time')
plt.xlabel('Food Preparation Time')

plt.subplot(1, 2, 2)
sns.boxplot(x=df['food_preparation_time'])
plt.title('Food Preparation Time')
plt.xlabel('Food Preparation Time')

plt.tight_layout()
plt.show()
```



Observations

Food prep time are generally evenly distributed across times, although the 20, 26, 31 and 35 minute times appear the most frequently and are similar in frequency. Average prep time of 27 minutes.

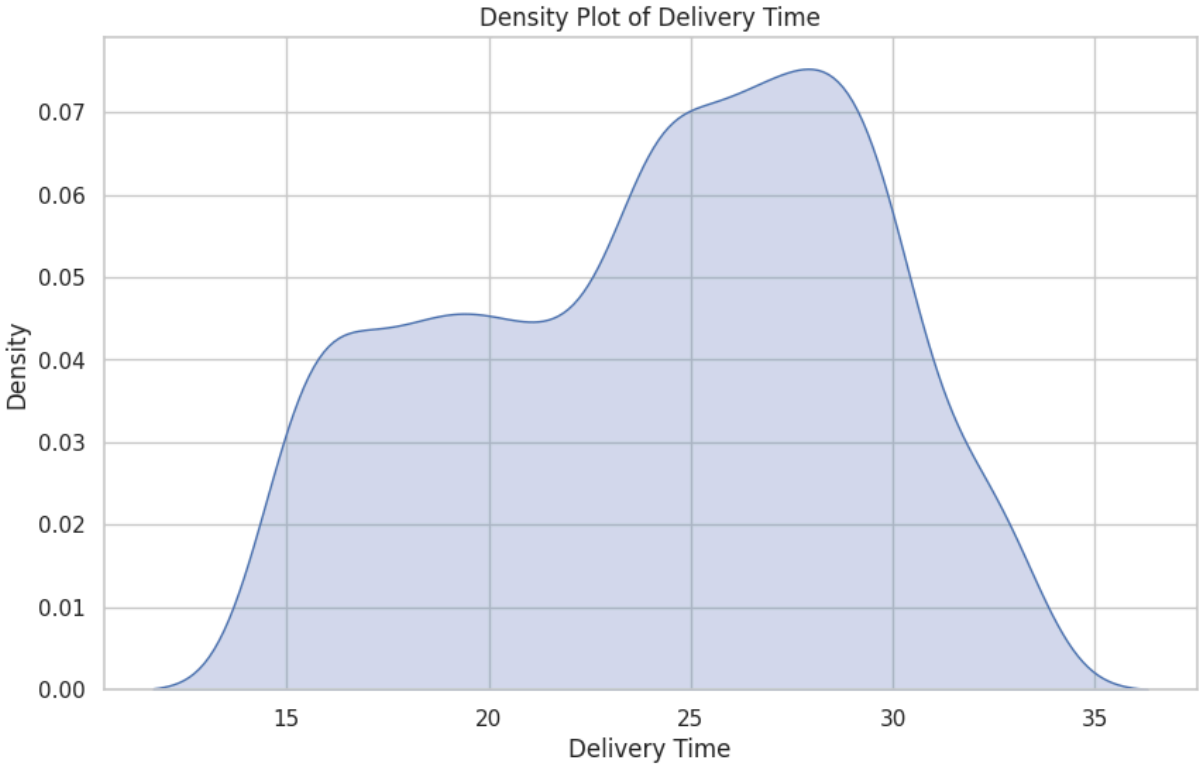
In [ ]:
```
# a histplot and boxplot of the delivery time column

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['delivery_time'], kde=True)
plt.title('Distribution of Delivery Time')
plt.xlabel('Delivery Time')

plt.subplot(1, 2, 2)
sns.boxplot(x=df['delivery_time'])
plt.title('Delivery Time')
plt.xlabel('Delivery Time')

plt.tight_layout()
plt.show()
```



Observations

Delivery time averaged around 25 minutes, with an average prep time of 27 minutes the average completion time (del time + prep time) should be around 52 minutes. Delivery times of 25 and 28 minutes appeared the most frequently.

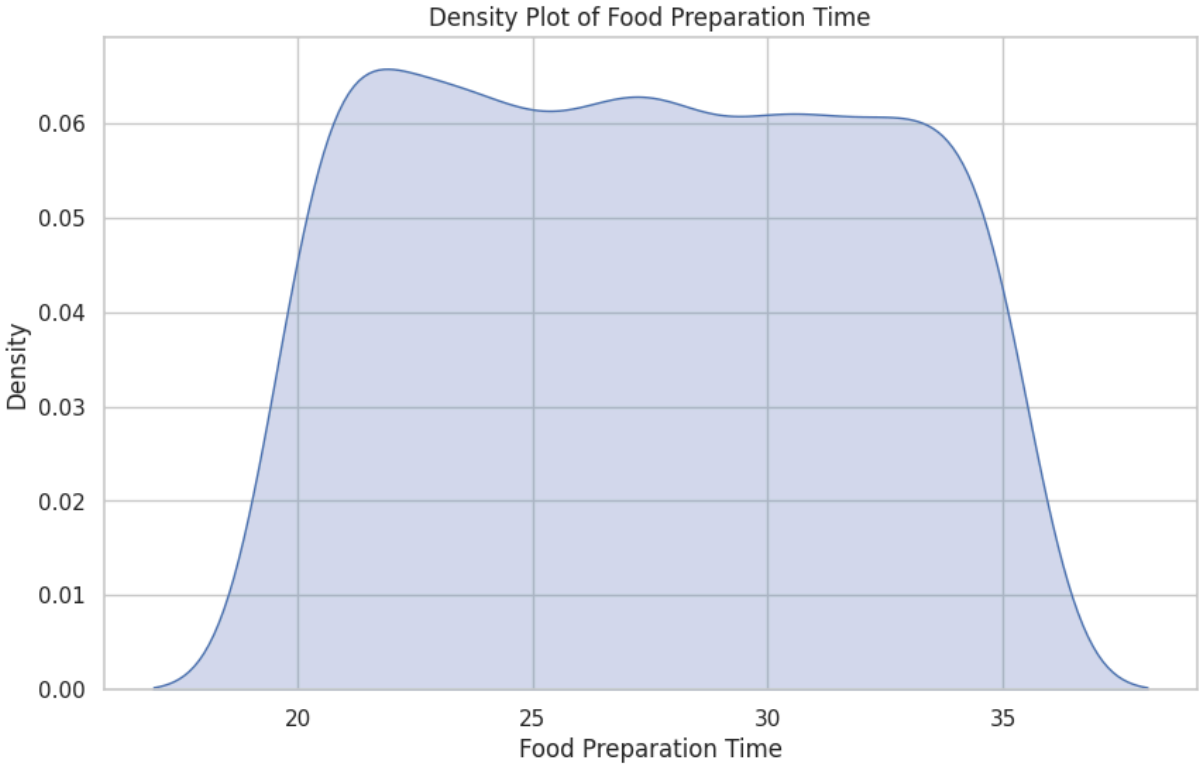
In [ ]:
```
# density plot of delivery time
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Plot the density of delivery times
sns.kdeplot(data=df, x='delivery_time', fill=True)

plt.title('Density Plot of Delivery Time')
plt.xlabel('Delivery Time')
plt.ylabel('Density')
plt.grid(True)
plt.show()
```



Observations

Delivery time frequency is highest around 27 minutes.

In [ ]:
```
# density plot of food prepartion time
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Plot the density of delivery times
sns.kdeplot(data=df, x='food_preparation_time', fill=True)

plt.title('Density Plot of Food Preparation Time')
plt.xlabel('Food Preparation Time')
plt.ylabel('Density')
plt.grid(True)
plt.show()
```



Observations

The food prep time is evenly distributed between 10 and 37 minutes.

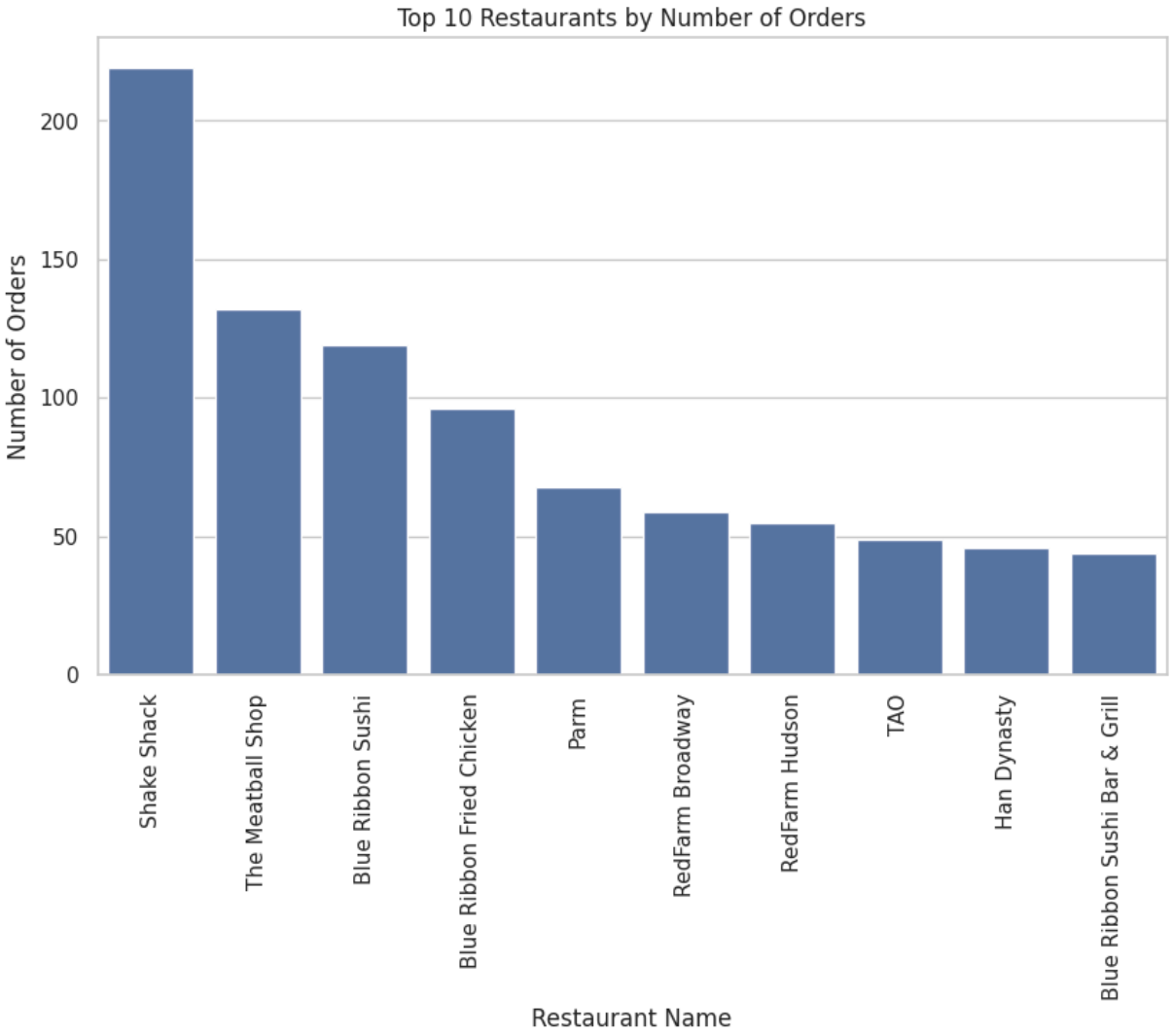
In [ ]:
```
# density plot of order cost
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Plot the density of delivery times
sns.kdeplot(data=df, x='cost_of_the_order', fill=True)

plt.title('Density Plot of Order Cost')
plt.xlabel('Order cost')
plt.ylabel('Density')
plt.grid(True)
plt.show()
```

Observations

The order cost is distributed between 2 and 36, with the highest frequency around 12. The amounts of 24 and 30 have smaller, but similar frequencies.

Next will look at distributions of individual categorical variables visually, along with distributions of multivariate variables

In [ ]:
```
# a countplot of the top 10 restaurants by number of orders

# Group by restaurant name and count the number of orders
restaurant_counts = df.groupby('restaurant_name')['restaurant_name'].count()

# Sort the counts in descending order and get the top 10
top_10_restaurants = restaurant_counts.sort_values(ascending=False).head(10)

# Create a countplot of the top 10 restaurants
plt.figure(figsize=(10, 6))
sns.countplot(x='restaurant_name', data=df, order=top_10_restaurants.index)
plt.xticks(rotation=90)
plt.xlabel('Restaurant Name')
plt.ylabel('Number of Orders')
plt.title('Top 10 Restaurants by Number of Orders')
plt.show()
```



In [ ]:
```
# the top 10 restaurants make up what proportion of total orders and sales

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

# Group by restaurant name and count the number of orders
restaurant_orders = df.groupby('restaurant_name')['restaurant_name'].count()

# Sort the orders in descending order and get the top 10
top_10_restaurant_orders = restaurant_orders.sort_values(ascending=False).he

# Calculate the total number of orders for the top 10 restaurants
top_10_orders_total = top_10_restaurant_orders.sum()

# Calculate the proportion of orders for the top 10 restaurants
top_10_orders_proportion = top_10_orders_total / total_orders

# Calculate total sales
total_sales = df['cost_of_the_order'].sum()

# Group by restaurant name and sum the cost of the order
restaurant_sales = df.groupby('restaurant_name')['cost_of_the_order'].sum()
```

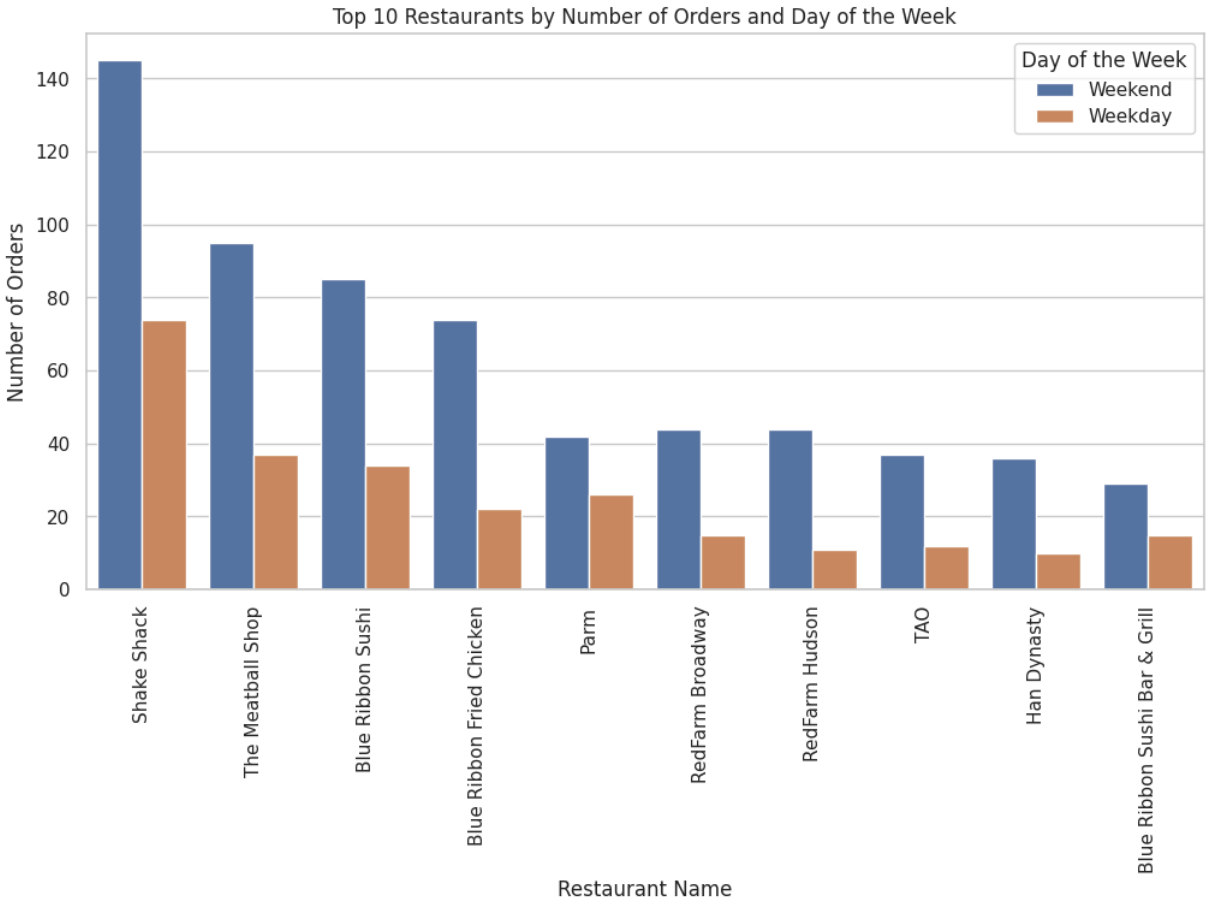
```
# Sort the sales in descending order and get the top 10
top_10_restaurant_sales = restaurant_sales.sort_values(ascending=False).head

# Calculate the proportion of sales for the top 10 restaurants
top_10_sales_proportion = top_10_restaurant_sales.sum() / total_sales

# Print the results
print(f"Top 10 restaurants make up {top_10_orders_proportion:.2%} of total o
print(f"Top 10 restaurants make up {top_10_sales_proportion:.2%} of total sa
```

```
Top 10 restaurants make up 46.73% of total orders.
Top 10 restaurants make up 46.45% of total sales.
```

The top 10 restaurants contribute over 45% of orders and sales.

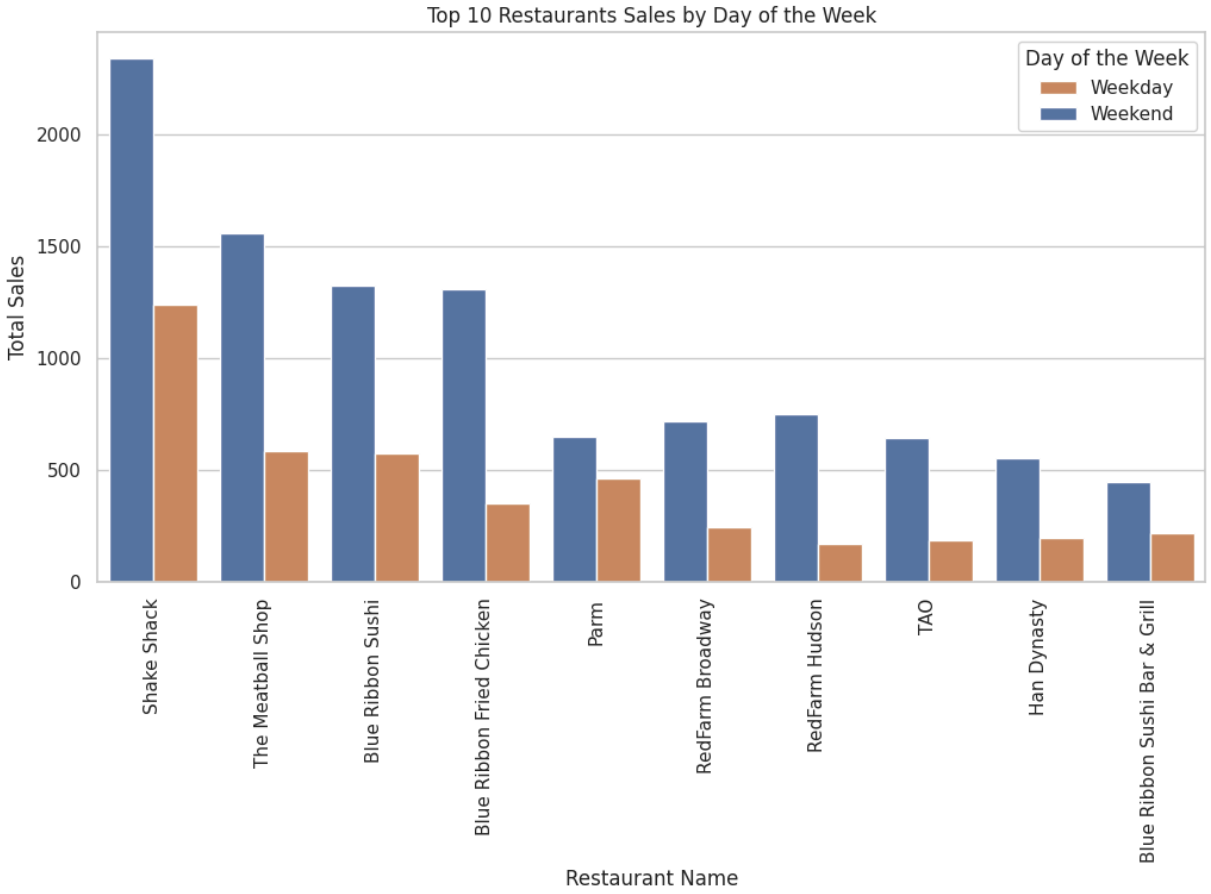
In [ ]:

```
# a countplot of the top 10 restaurants with their number of orders during t

# Group by restaurant name and sum the cost of the order
restaurant_sales = df.groupby('restaurant_name')['cost_of_the_order'].sum()

# Sort the sales in descending order and get the top 10
top_10_restaurant_sales = restaurant_sales.sort_values(ascending=False).head

# Create a countplot of the top 10 restaurants with sale information hue by
plt.figure(figsize=(12, 6))
sns.countplot(x='restaurant_name', hue='day_of_the_week', data=df[df['restau
plt.xticks(rotation=90)
plt.xlabel('Restaurant Name')
plt.ylabel('Number of Orders')
plt.title('Top 10 Restaurants by Number of Orders and Day of the Week')
plt.legend(title='Day of the Week')
plt.show()
```

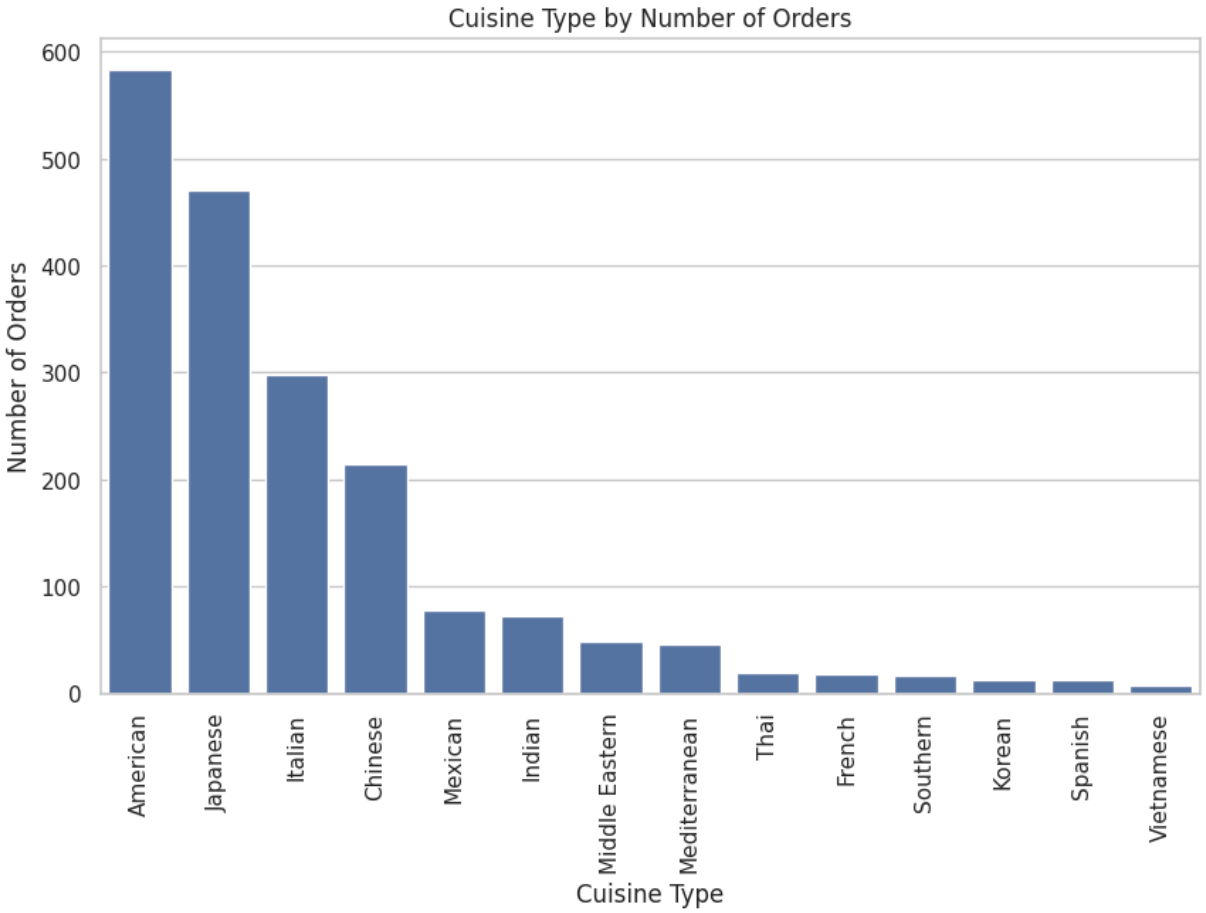


In [ ]:
```
# barplot that shows sales by day of the week for top 10 restaurants

# Group by restaurant name and sum the cost of the order
restaurant_sales = df.groupby('restaurant_name')['cost_of_the_order'].sum()

# Sort the sales in descending order and get the top 10
top_10_restaurant_sales = restaurant_sales.sort_values(ascending=False).head

# Create a bar chart of the top 10 restaurants with sale information hue by
plt.figure(figsize=(12, 6))
chart = sns.barplot(x='restaurant_name', y='cost_of_the_order', hue='day_of_
                    data=df[df['restaurant_name'].isin(top_10_restaurant_sale
                    order=top_10_restaurant_sales.index,
                    estimator=sum, ci=None)  # ci=None removes the confidence
plt.xticks(rotation=90)
plt.xlabel('Restaurant Name')
plt.ylabel('Total Sales')
plt.title('Top 10 Restaurants Sales by Day of the Week')
handles, labels = chart.get_legend_handles_labels()
chart.legend(handles[::-1], labels[::-1], title='Day of the Week') # Reverse

plt.show()
```

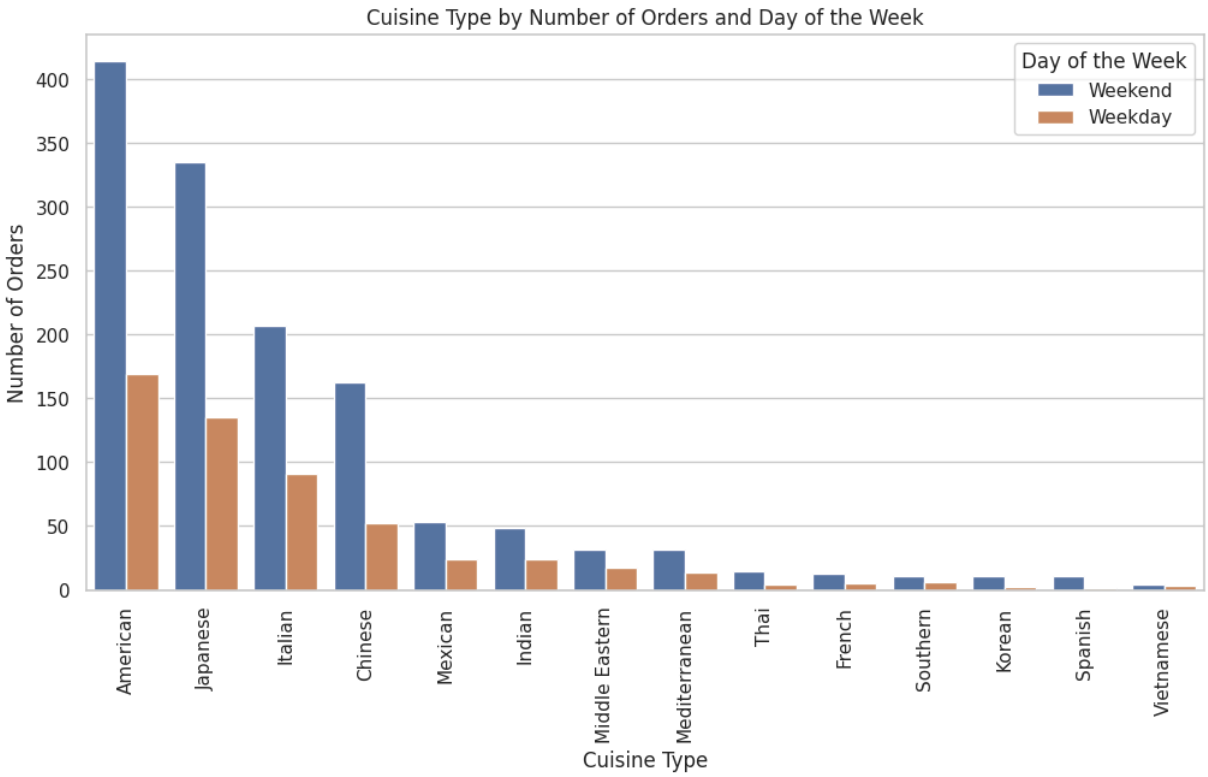
```
<ipython-input-56-51645025fbb2>:11: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  chart = sns.barplot(x='restaurant_name', y='cost_of_the_order', hue='day_of_the_week',
```



In [ ]:
```
# Observations on Cuisine

# Group by cuisine type and count the number of orders
cuisine_counts = df.groupby('cuisine_type')['cuisine_type'].count()

# Sort the counts in descending order
cuisine_counts_sorted = cuisine_counts.sort_values(ascending=False)

# Create a countplot of cuisine types
plt.figure(figsize=(10, 6))
sns.countplot(x='cuisine_type', data=df, order=cuisine_counts_sorted.index)
plt.xticks(rotation=90)
plt.xlabel('Cuisine Type')
plt.ylabel('Number of Orders')
plt.title('Cuisine Type by Number of Orders')
plt.show()
```

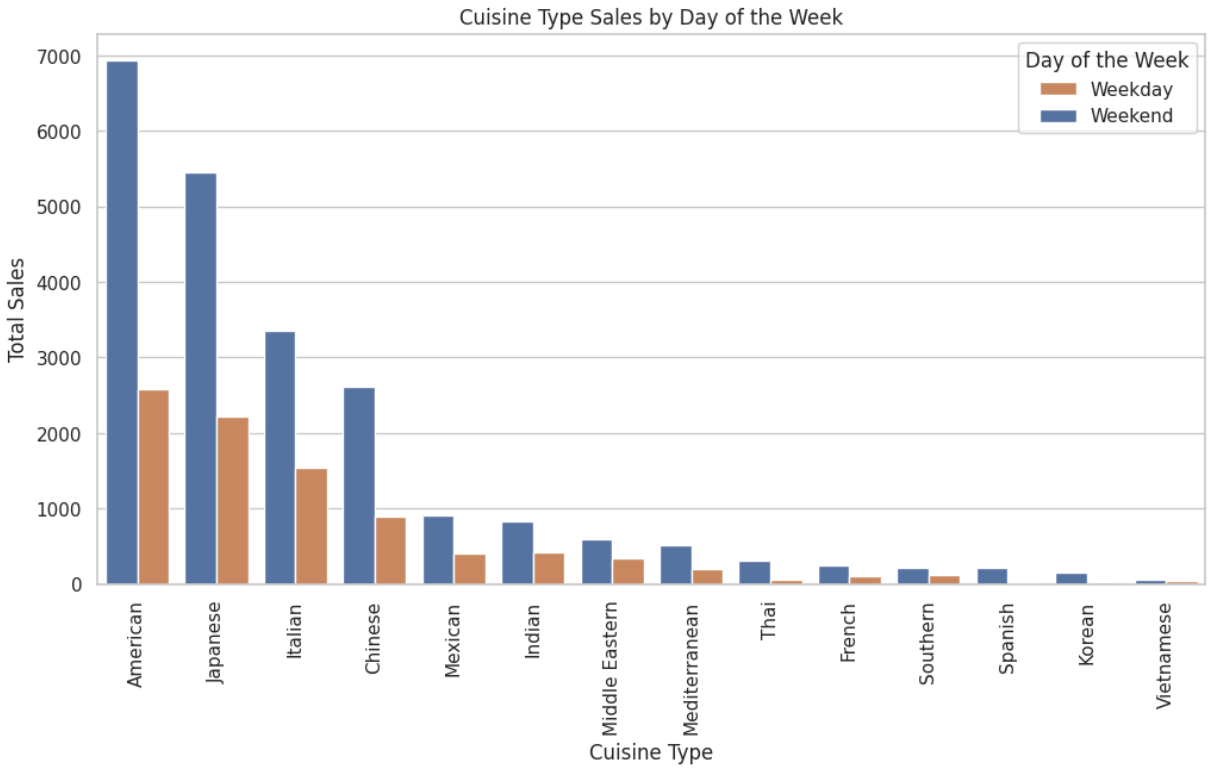


In [ ]:
```
# a countplot of number of orders by cuisine type and day of the week

# Group by cuisine type and count the number of orders
cuisine_counts = df.groupby('cuisine_type')['cuisine_type'].count()

# Sort the counts in descending order
cuisine_counts_sorted = cuisine_counts.sort_values(ascending=False)

# Create a countplot of cuisine types hue = day of the week
plt.figure(figsize=(12, 6))
sns.countplot(x='cuisine_type', hue='day_of_the_week', data=df, order=cuisir
plt.xticks(rotation=90)
plt.xlabel('Cuisine Type')
plt.ylabel('Number of Orders')
plt.title('Cuisine Type by Number of Orders and Day of the Week')
plt.legend(title='Day of the Week')
plt.show()
```



In [ ]:
```
# countplot of cuisine type, total sales by day of the week

# Group by cuisine type and sum the cost of the order
cuisine_sales = df.groupby('cuisine_type')['cost_of_the_order'].sum()

# Sort the sales in descending order
cuisine_sales_sorted = cuisine_sales.sort_values(ascending=False)

# Create a bar chart of the cuisines with sale information hue by day of the
plt.figure(figsize=(12, 6))
chart = sns.barplot(x='cuisine_type', y='cost_of_the_order', hue='day_of_the
                    data=df,
                    order=cuisine_sales_sorted.index,
                    estimator=sum, ci=None)  # ci=None removes the confidence
plt.xticks(rotation=90)
plt.xlabel('Cuisine Type')
plt.ylabel('Total Sales')
plt.title('Cuisine Type Sales by Day of the Week')
handles, labels = chart.get_legend_handles_labels()
chart.legend(handles[::-1], labels[::-1], title='Day of the Week') # Reverse

plt.show()
```

```
<ipython-input-59-b938d78b86cc>:11: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  chart = sns.barplot(x='cuisine_type', y='cost_of_the_order', hue='day_of_t
he_week',
```



In [ ]:
```
# what percentage of total count of cuisines are made up by the top 4 cuisin

# Group by cuisine type and count the number of orders
cuisine_counts = df.groupby('cuisine_type')['cuisine_type'].count()

# Sort the counts in descending order
cuisine_counts_sorted = cuisine_counts.sort_values(ascending=False)

# Get the top 4 cuisines
top_4_cuisines = cuisine_counts_sorted.head(4)

# Calculate the percentage of total orders made up by the top 4 cuisines
percentage_top_4 = (top_4_cuisines.sum() / cuisine_counts.sum()) * 100

print(f"The top 4 cuisines make up {percentage_top_4:.2f}% of the total cuis
```

```
The top 4 cuisines make up 82.56% of the total cuisine orders and sales.
```

Observations:

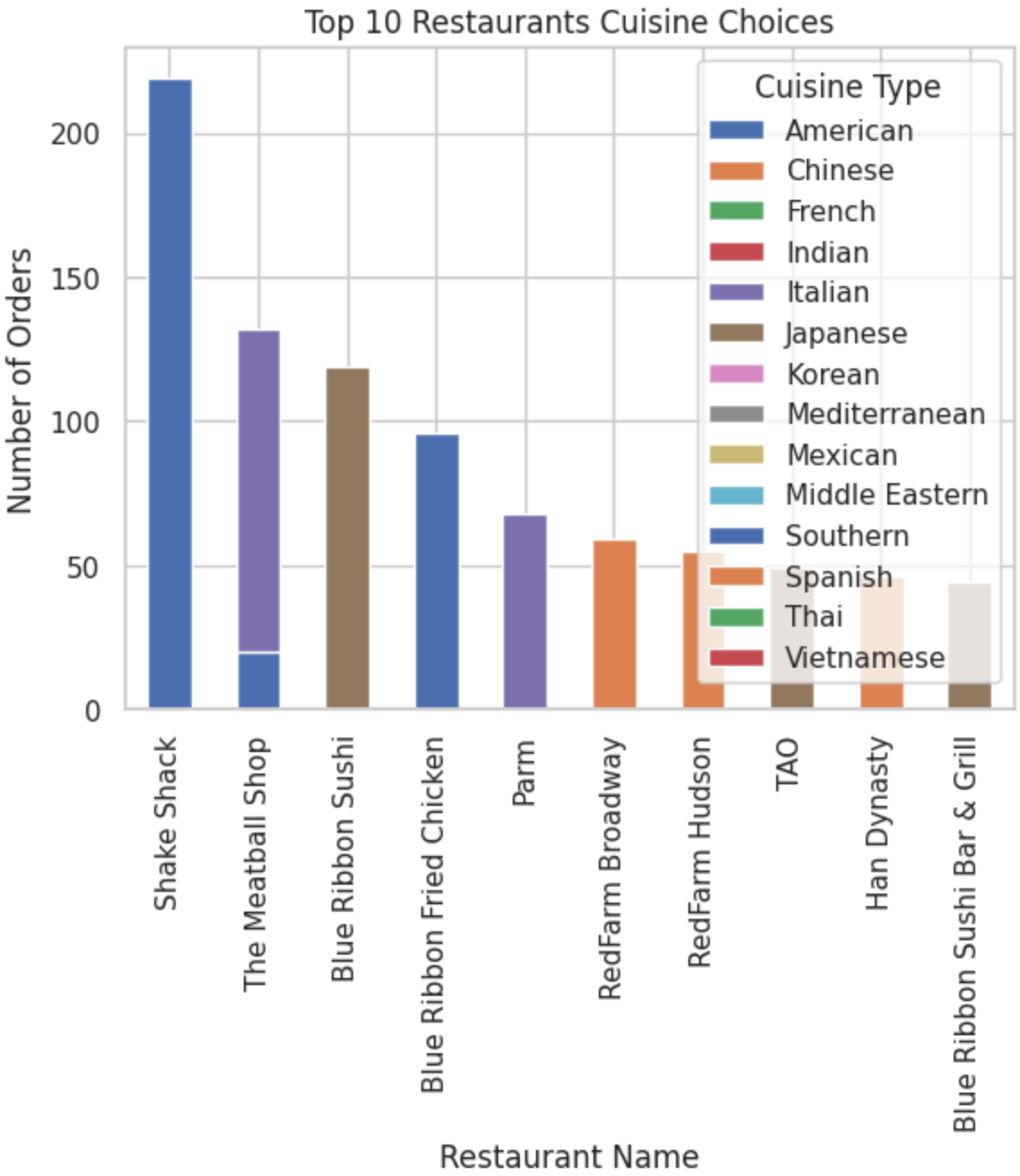
The top 10 restaurants make the majority of their sales and orders on the weekend, serving the top 4 cuisines.

The top 4 cuisines account for over 80% of orders and sales.

In [ ]:
```
# a chart that shows the top 10 restaurants choices of cuisine

# Group by restaurant name and cuisine type, then count the number of orders
restaurant_cuisine_counts = df.groupby(['restaurant_name', 'cuisine_type'])[

# Sort the restaurants by total number of orders in descending order and get
```

```
top_10_restaurants = restaurant_cuisine_counts.sum(axis=1).sort_values(ascer

# Filter the data to only include the top 10 restaurants
top_10_restaurant_cuisine_counts = restaurant_cuisine_counts.loc[top_10_rest

# Create a stacked bar chart
plt.figure(figsize=(15, 6))
top_10_restaurant_cuisine_counts.plot(kind='bar', stacked=True)
plt.xticks(rotation=90)
plt.xlabel('Restaurant Name')
plt.ylabel('Number of Orders')
plt.title('Top 10 Restaurants Cuisine Choices')
plt.legend(title='Cuisine Type')
plt.show()
```

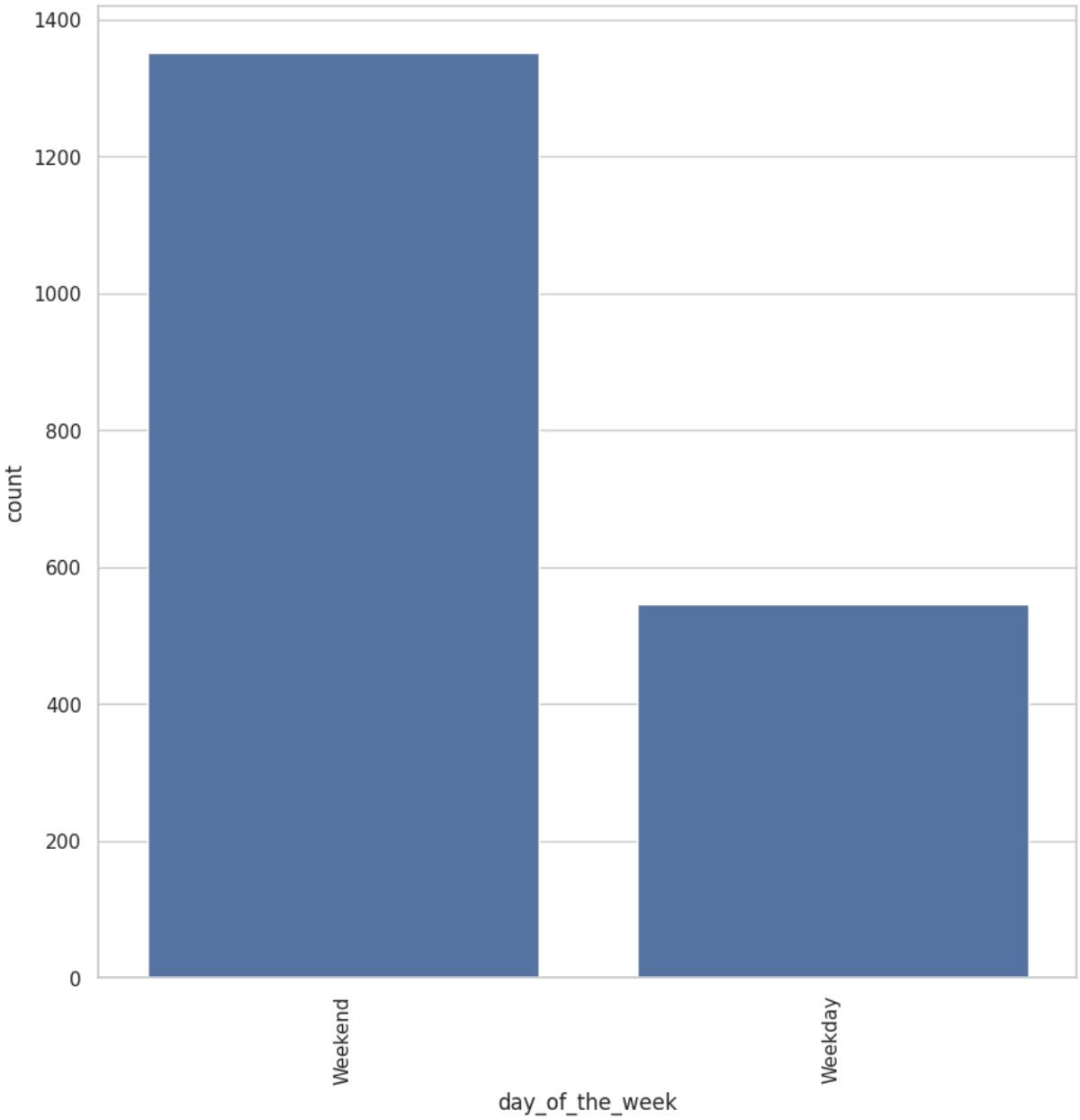
```
<Figure size 1500x600 with 0 Axes>
```



Observations:

The top 10 restaurants serve the top 4 cuisines

In [ ]:
```
#observations on day of the week
plt.figure(figsize=(10, 10))
sns.countplot(data=df,x='day_of_the_week')
plt.xticks(rotation=90)
plt.show()
```

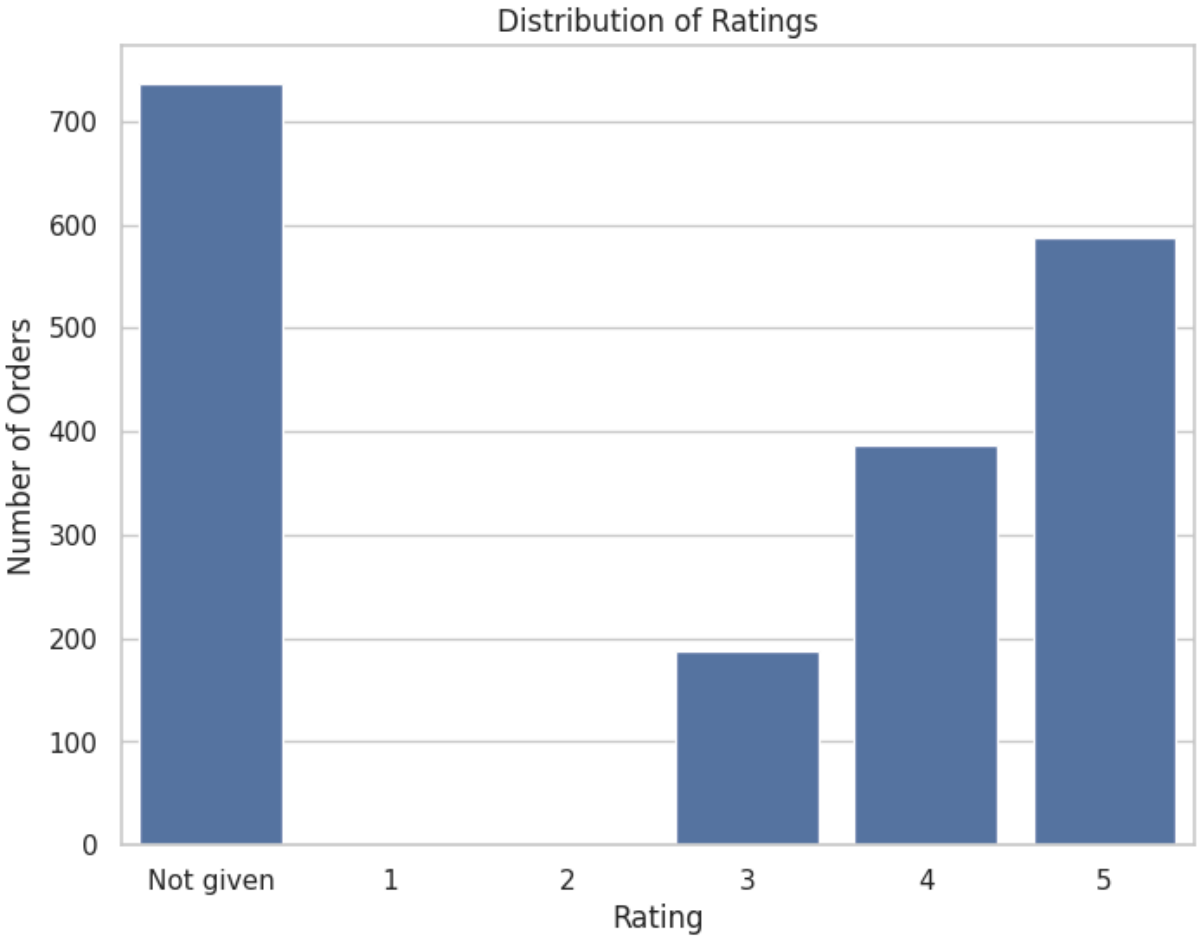


Observations

The weekend is twice as busy as the week

In [ ]:
```
# Observations on ratings
```

```
# Define the order of ratings on the x-axis
rating_order = ['Not given', 1, 2, 3, 4, 5]

# Create the countplot with the specified order
plt.figure(figsize=(8, 6))
sns.countplot(x='rating', data=df, order=rating_order)
plt.xlabel('Rating')
plt.ylabel('Number of Orders')
plt.title('Distribution of Ratings')
plt.show()
```



In [ ]:
```
#  the count of ratings for the top 10 restaurants by day of the week

# Group by restaurant name and sum the cost of the order
restaurant_sales = df.groupby('restaurant_name')['cost_of_the_order'].sum()

# Sort the sales in descending order and get the top 10
top_10_restaurant_sales = restaurant_sales.sort_values(ascending=False).head

# Filter the DataFrame to include only the top 10 restaurants
top_10_df = df[df['restaurant_name'].isin(top_10_restaurant_sales.index)]

# Group by restaurant name, day of the week, and rating, then count the numb
rating_counts = top_10_df.groupby(['restaurant_name', 'day_of_the_week', 'ra

rating_counts
```

Out[ ]:

restaurant_name	day_of_the_week	rating 3	4	5	Not given
Blue Ribbon Fried Chicken	**Weekday**	1	6	8	7
	Weekend	10	15	24	25
Blue Ribbon Sushi	**Weekday**	5	3	7	19
	Weekend	11	22	25	27
Blue Ribbon Sushi Bar & Grill	**Weekday**	0	1	7	7
	Weekend	2	4	8	15
Han Dynasty	**Weekday**	1	2	1	6
	Weekend	2	5	12	17
Parm	**Weekday**	1	4	8	13
	Weekend	8	12	6	16
RedFarm Broadway	**Weekday**	2	4	2	7
	Weekend	6	11	16	11
RedFarm Hudson	**Weekday**	0	1	5	5
	Weekend	9	9	10	16
Shake Shack	**Weekday**	7	22	18	27
	Weekend	16	28	42	59
TAO	**Weekday**	1	3	1	7
	Weekend	2	9	12	14
The Meatball Shop	**Weekday**	2	6	22	7
	Weekend	8	15	31	41

Observations

Most of the ratings for the top 10 restaurants occur on the weekend, when they are the busiest.

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

In [ ]:
```
# top 5 restaurants in terms of number of orders received

top_5_restaurants = df.groupby('restaurant_name')['restaurant_name'].count()
top_5_restaurants
```

Out[ ]:

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

dtype: int64

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

In [ ]:
```
# most popular cuisine on weekends

weekend_cuisine = df[df['day_of_the_week'] == 'Weekend'].groupby('cuisine_ty
weekend_cuisine
```

Out[ ]:

cuisine_type	cuisine_type
American	415

dtype: int64

Observations:

The top 3 restaurants account for approximately 500 of the 1900 sales, approximately 20% of sales.

American cuisine is the most popular on the weekend

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

In [ ]:
```
# what percentage of the orders cost more than 20 dollars?

# Calculate the percentage of orders that cost more than $20
percentage_orders_over_20 = (len(df[df['cost_of_the_order'] > 20]) / len(df)

# Print the result
print(f"{percentage_orders_over_20:.2f}% of the orders cost more than $20.")
```

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

Observations

Approximately 30% of orders are over $20

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

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

# Print the result
print(f"The average order delivery time is {mean_delivery_time:.2f} minutes.
```

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

Observations:

The average order delivery time is approximately 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]

Observations

I had previously removed the customer_id and order_id columns to focus on analyzing the restaurant domain, and would like to analyze the customer domain in a more indepth analysis in the future.

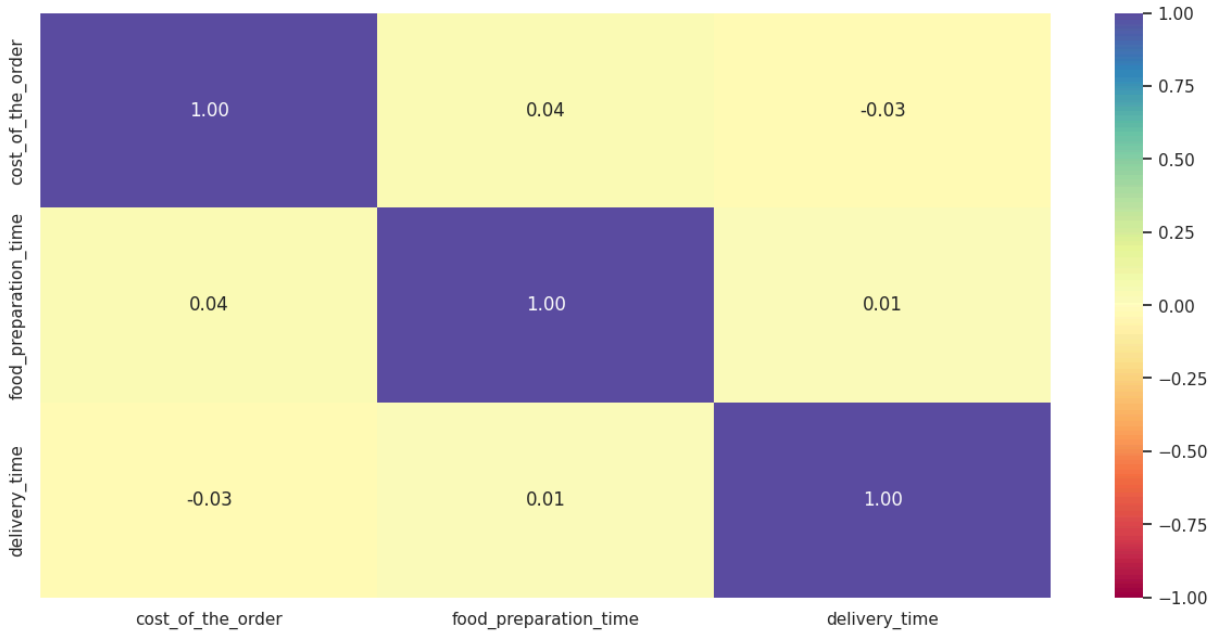
Multivariate Analysis

Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. [10 marks]

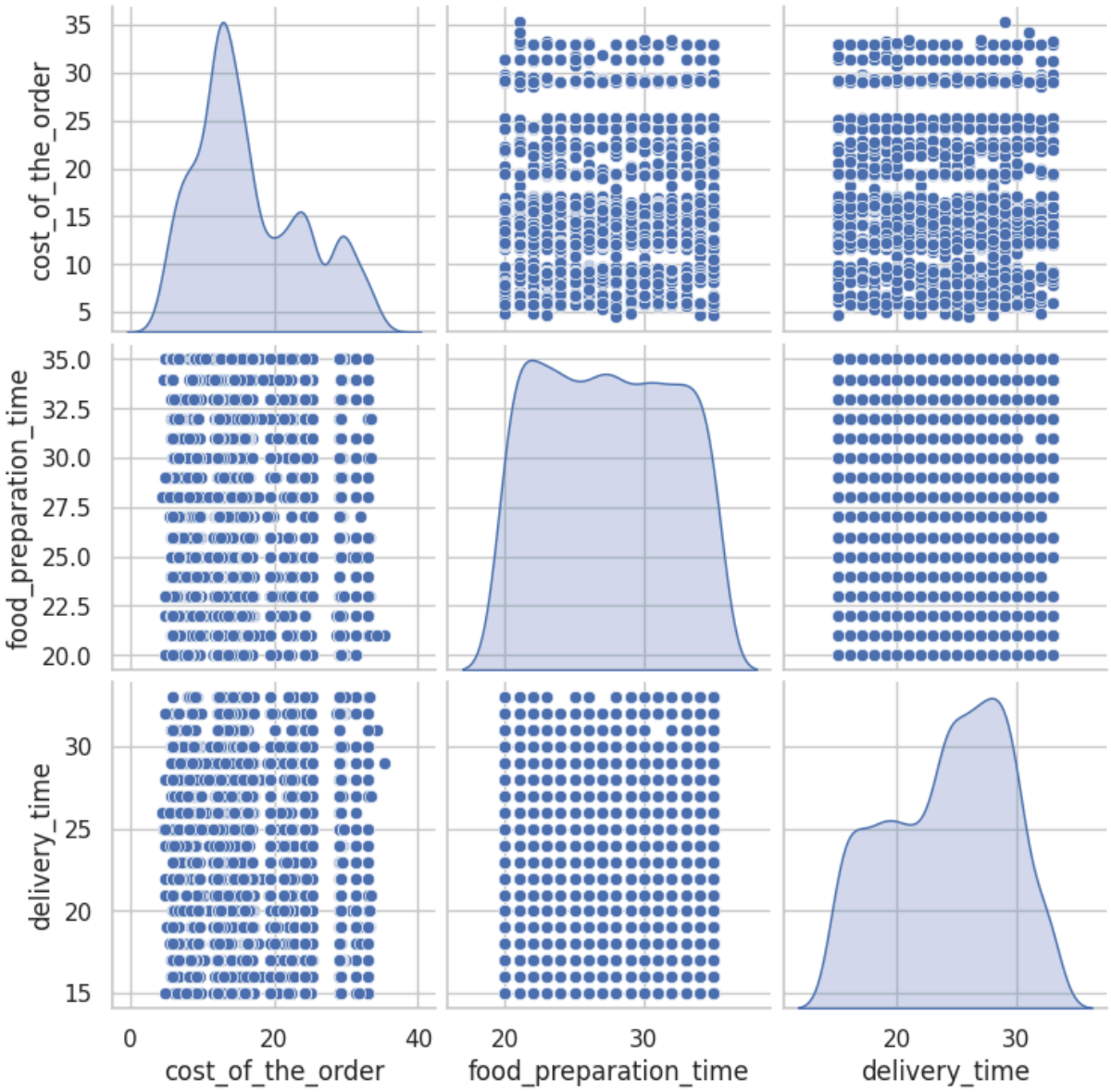
First will perform analysis on numerical variables with a heatmap correlation and pairplot scatterplot

In [ ]:
```
# Check for correlation among numerical variables using a heatmap
num_var = ['cost_of_the_order','food_preparation_time','delivery_time']

corr = df[num_var].corr()

# plot the heatmap

plt.figure(figsize=(15, 7))
```

```
sns.heatmap(corr, annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectral")
plt.show()
```



In [ ]:
```
#check for correlation of numerical var using a pairplot
sns.pairplot(data=df[num_var], diag_kind="kde")
plt.show()
```



Observations:

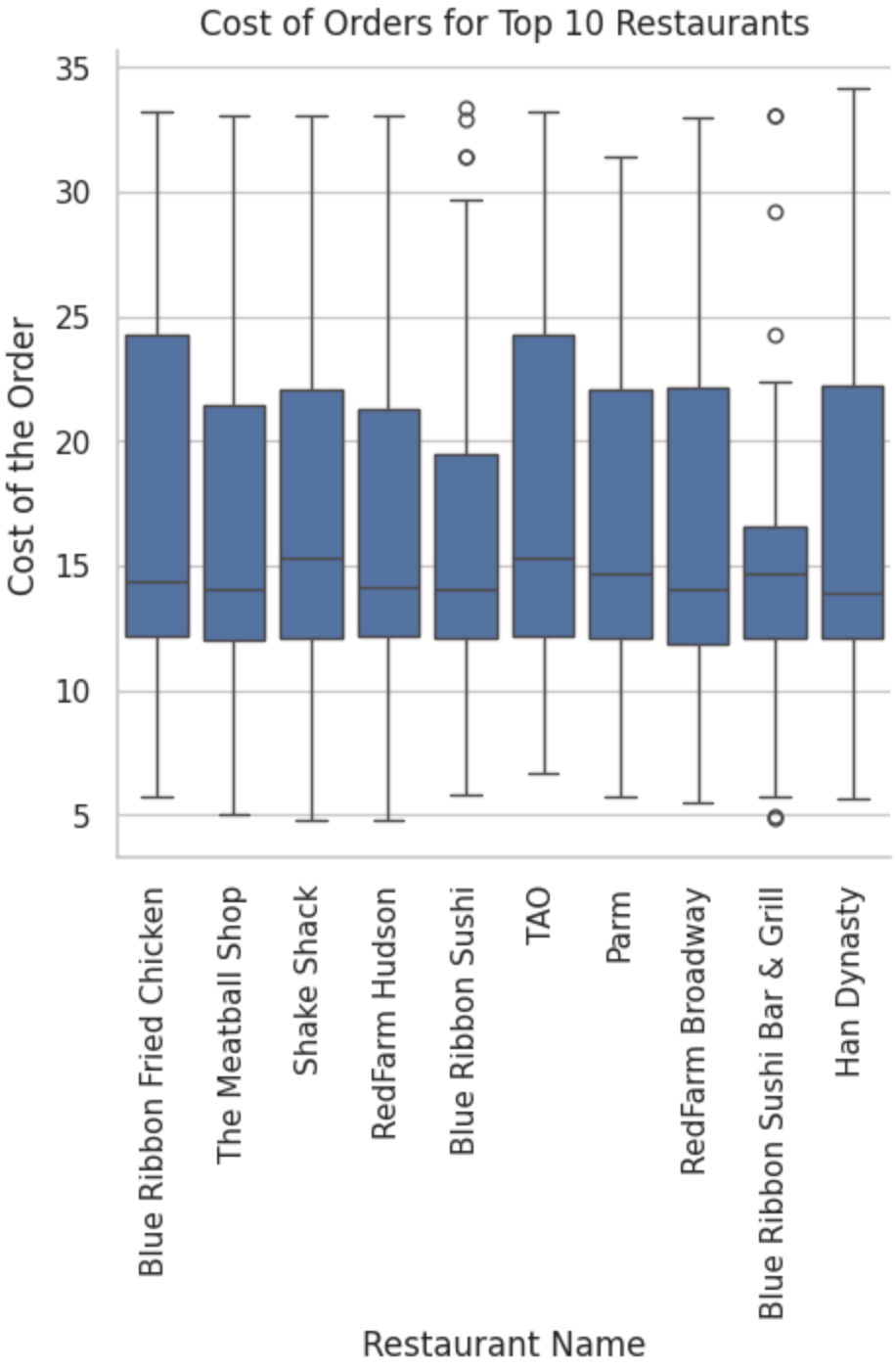
There does not appear to be any correlation between the numerical variables

Perform analysis between numerical and categorical variables using catplots

In [ ]:
```
# observation of top 10 restaurants and order cost

# Group by restaurant name and count the number of orders
restaurant_counts = df.groupby('restaurant_name')['restaurant_name'].count()

# Sort the counts in descending order and get the top 10
top_10_restaurants = restaurant_counts.sort_values(ascending=False).head(10)

# Filter the DataFrame to include only the top 10 restaurants
df_top_10 = df[df['restaurant_name'].isin(top_10_restaurants.index)]
```

```
plt.figure(figsize=(15, 8))
sns.catplot(x='restaurant_name', y='cost_of_the_order', kind='box', data=df_
plt.xticks(rotation=90)
plt.xlabel('Restaurant Name')
plt.ylabel('Cost of the Order')
plt.title('Cost of Orders for Top 10 Restaurants')
plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```



Observations

Outliers appear for suspected duplicate restaurant, Blue Ribbon Sushi, in both instances. Blue Ribbon Sushi Bar & Grill appears to have the lowest cost meals, TAO the highest generally although they all max out around 35, except for Blue Ribbon Sushi that max out at 23 and 30 respectively

In [ ]:
```
# what types of cuisine does blue ribbon sushi and blue Ribbon sushi bar and

# Filter the DataFrame to include only rows for Blue Ribbon Sushi and Blue R
blue_ribbon_restaurants = df[df['restaurant_name'].str.contains('Blue Ribbon

# Get the unique cuisine types for these restaurants
cuisine_types = blue_ribbon_restaurants['cuisine_type'].unique()

# Print the unique cuisine types
print(f"Blue Ribbon Sushi and Blue Ribbon Sushi Bar and Grill serve the foll
```

```
Blue Ribbon Sushi and Blue Ribbon Sushi Bar and Grill serve the following cu
isine types: ['Japanese']
```

Observations:

There appears to be cost of order outliers in 2 of the top 10 restaurants. Further exercise would be to determine the frequency and proportion of outliers in the dataset to help determine if and how they should be handled.

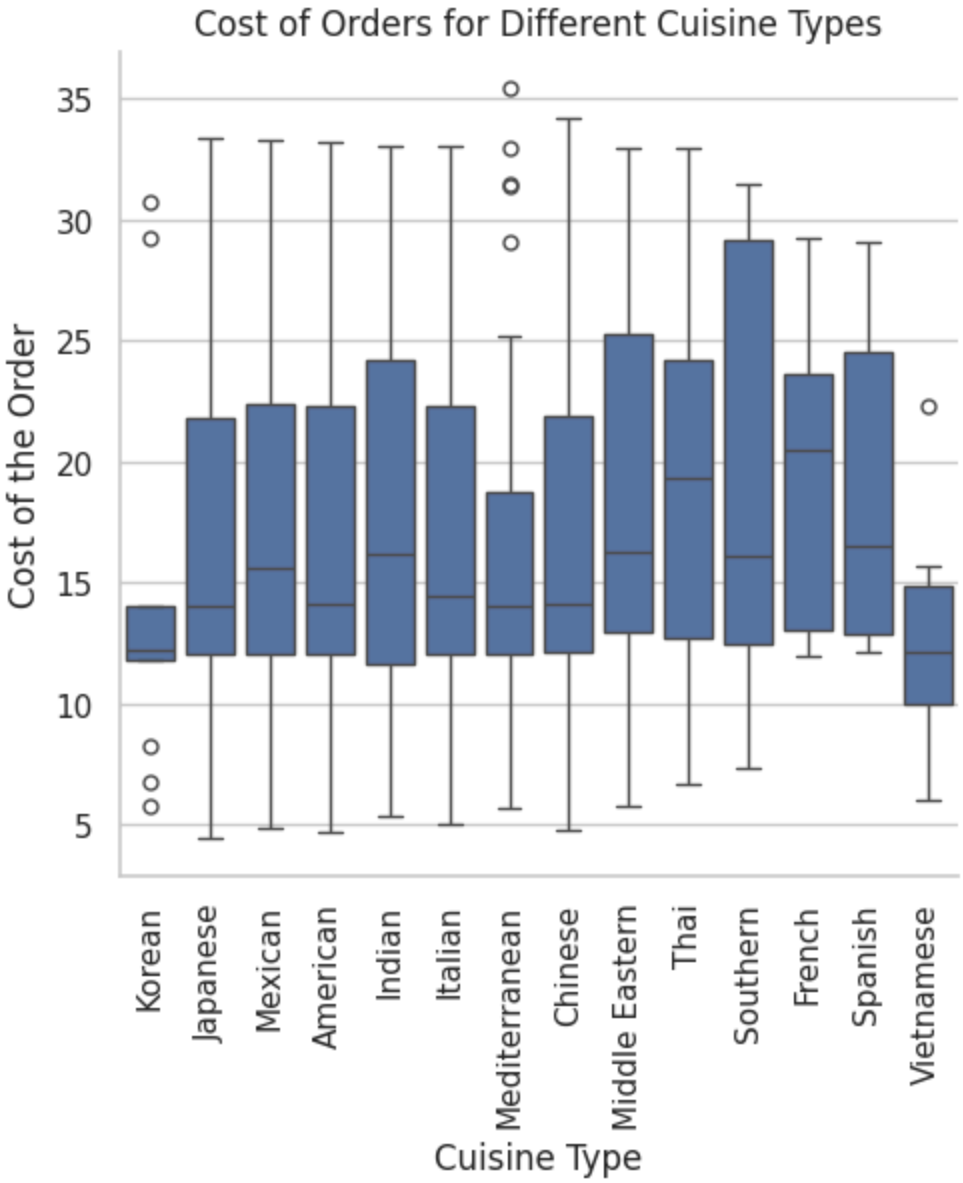
Outliers appear for suspected duplicate restaurant, Blue Ribbon Sushi, in both instances. Blue Ribbon Sushi Bar & Grill appears to have the lowest cost meals, TAO the highest generally although they all max out around 35, except for Blue Ribbon Sushi that max out at 23 and 30 respectively

Blue Ribbon Sushi Bar and Grill (Blue Ribbon Sushi) cost of order minimums are similar although one serves less expensive cuisine than the other. However, they both serve Japanese cuisine so it can be assumed that it is the same restaurant (though comparing restaurant information would further determine truth) and if it is determined that they are duplicates, then they should be combined, treated as a duplicate in the beginning of the analysis. They are also the only two with outliers in the top 10.

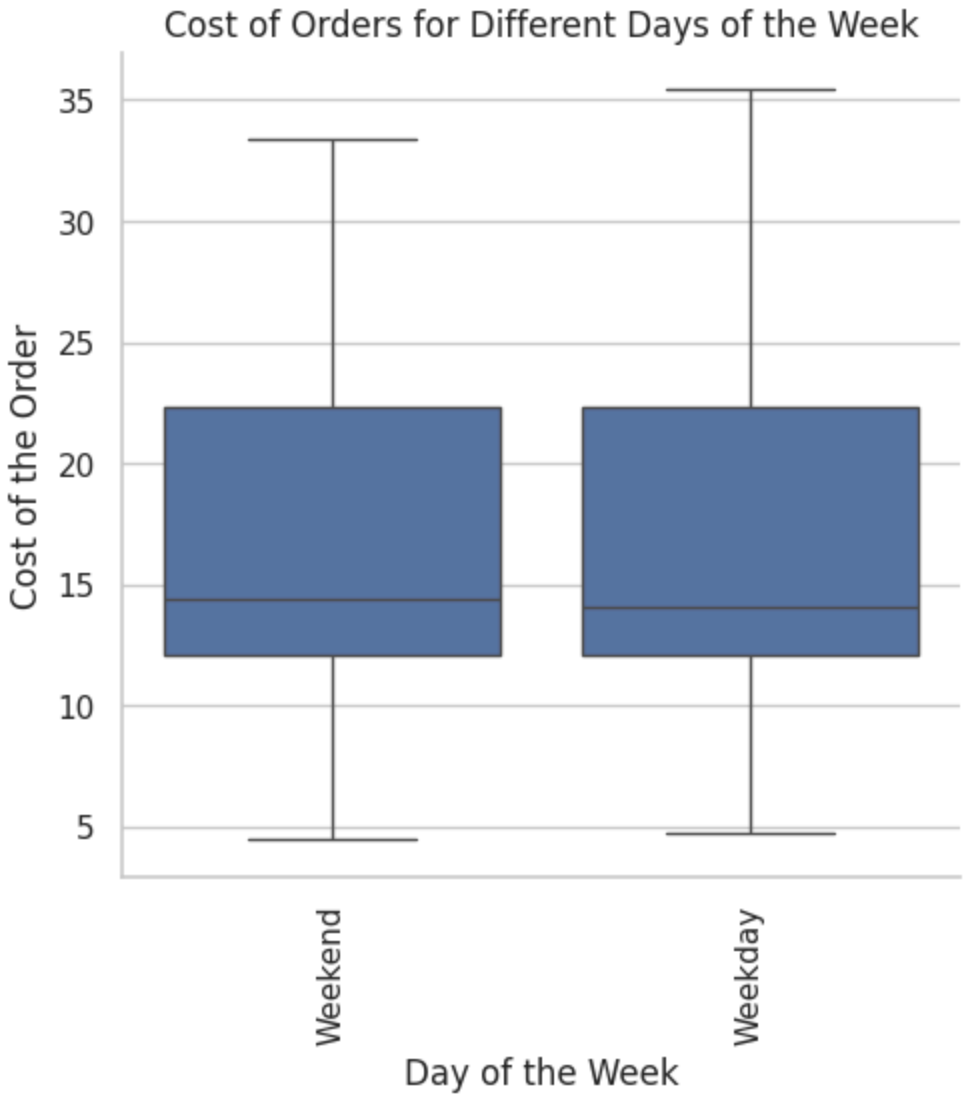
In [ ]:
```
# Create catplots for cuisine, day of the week, and rating compared to cost

plt.figure(figsize=(15, 8))
sns.catplot(x='cuisine_type', y='cost_of_the_order', kind='box', data=df)
plt.xticks(rotation=90)
plt.xlabel('Cuisine Type')
plt.ylabel('Cost of the Order')
plt.title('Cost of Orders for Different Cuisine Types')
plt.show()


plt.figure(figsize=(15, 8))
sns.catplot(x='day_of_the_week', y='cost_of_the_order', kind='box', data=df)
```

```
plt.xticks(rotation=90)
plt.xlabel('Day of the Week')
plt.ylabel('Cost of the Order')
plt.title('Cost of Orders for Different Days of the Week')
plt.show()

plt.figure(figsize=(15, 8))
sns.catplot(x='rating', y='cost_of_the_order', kind='box', data=df)
plt.xticks(rotation=90)
plt.xlabel('Rating')
plt.ylabel('Cost of the Order')
plt.title('Cost of Orders for Different Ratings')
plt.show()
```

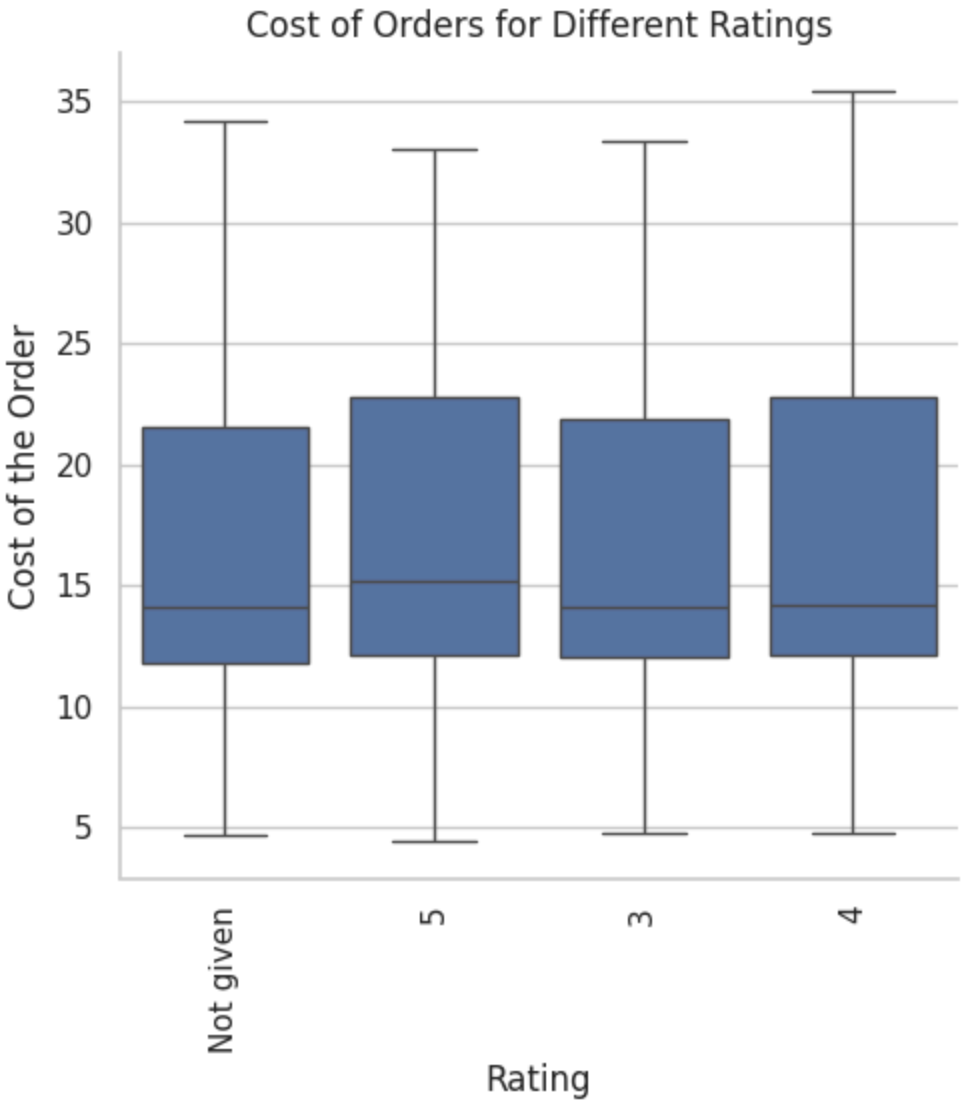
```
<Figure size 1500x800 with 0 Axes>
```




```
<Figure size 1500x800 with 0 Axes>
```



<Figure size 1500x800 with 0 Axes>



Observations

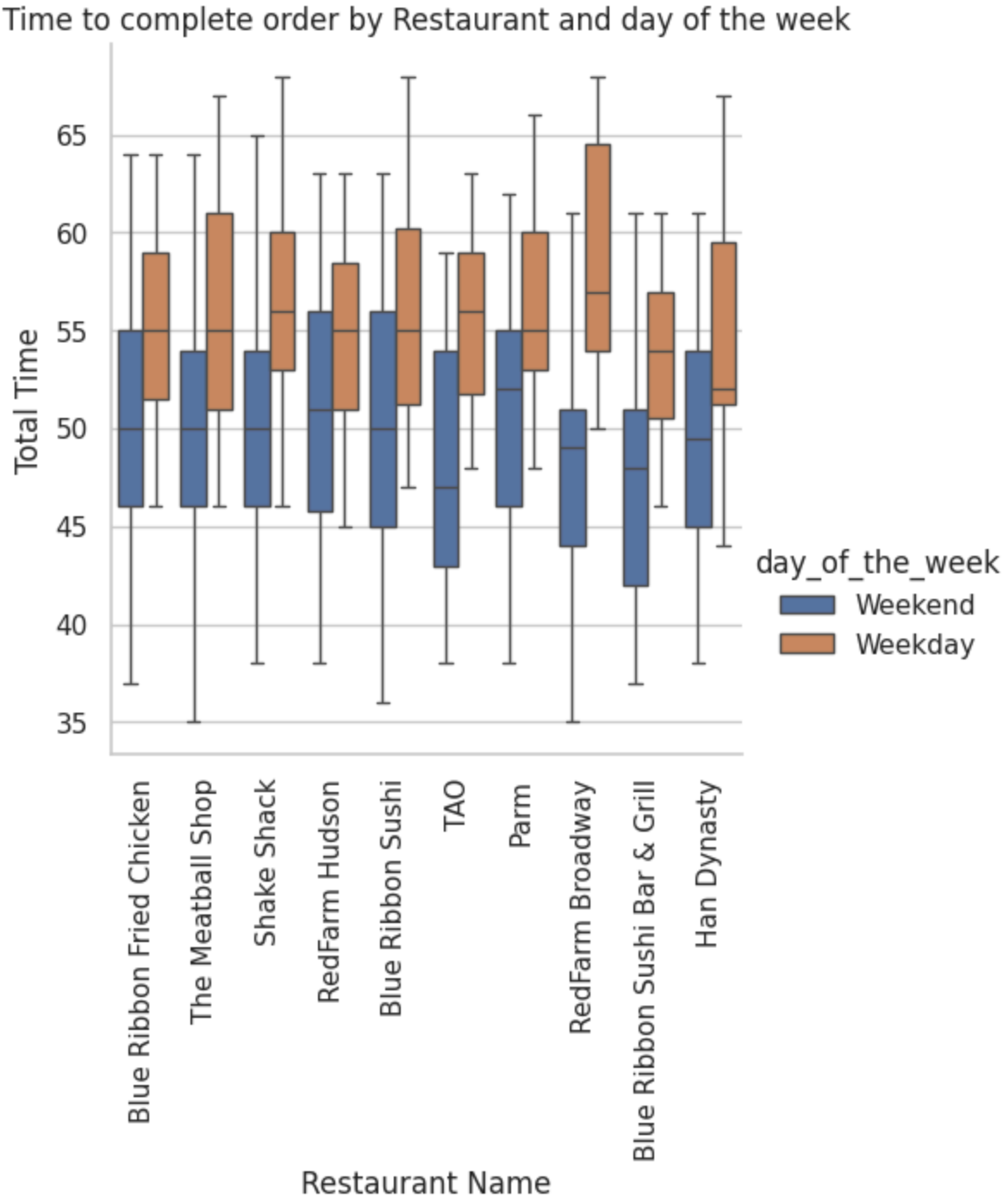
Korean and Vietnamese cuisines are generally the least expensive, with Southern the most expensive and Thai the highest average cost

Ratings and Day of the Week are generally not influenced by order cost.

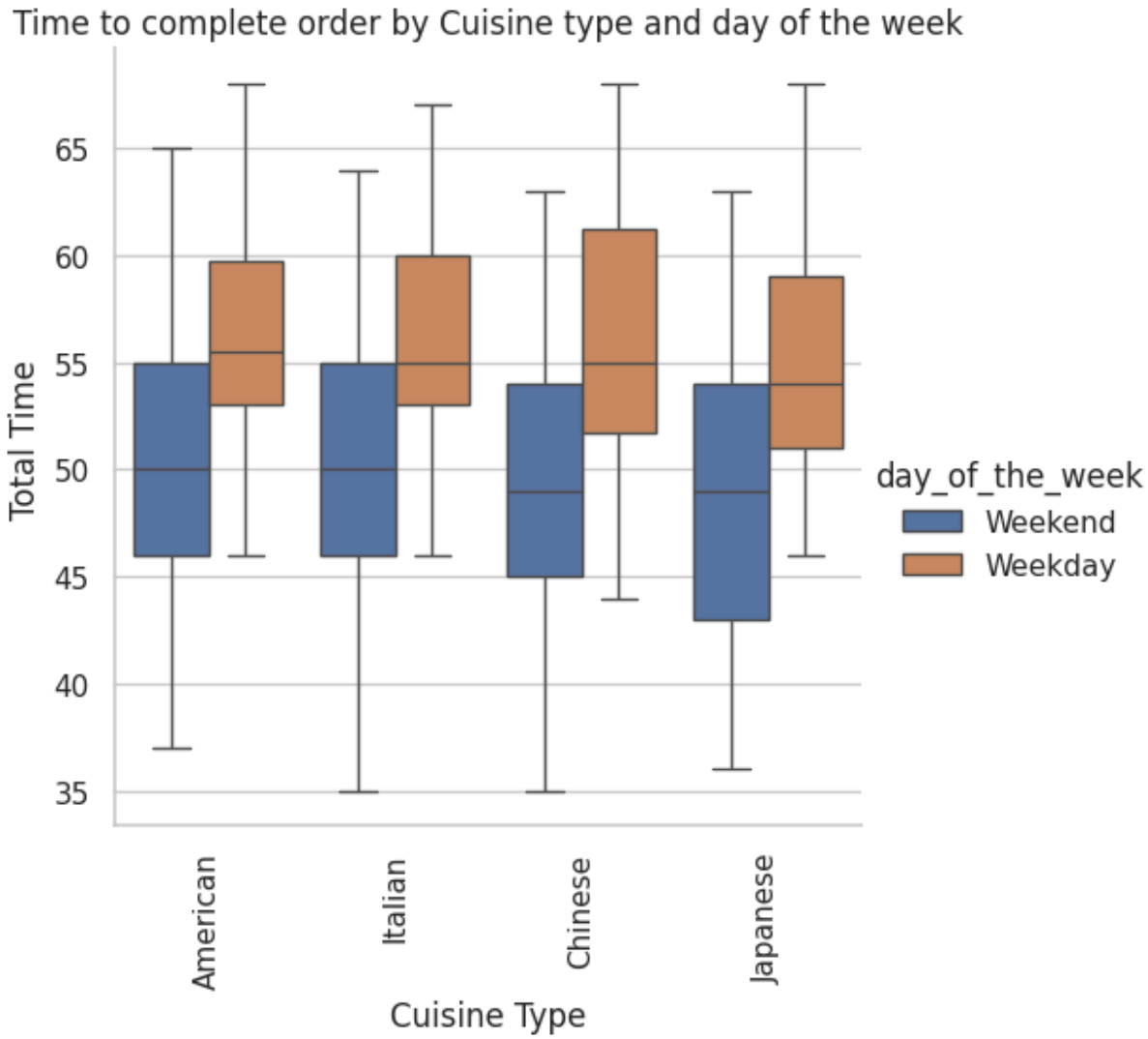
In [ ]:
```
# a catplot of the top 10 restaurants, day of the week, cuisine type and rat

# Group by restaurant name and count the number of orders
restaurant_counts = df.groupby('restaurant_name')['restaurant_name'].count()

# Sort the counts in descending order and get the top 10
top_10_restaurants = restaurant_counts.sort_values(ascending=False).head(10)

# Filter the DataFrame to include only the top 10 restaurants
df_top_10 = df[df['restaurant_name'].isin(top_10_restaurants.index)]

# Calculate the total time using preparation time and delivery time, in minu
```

```
df_top_10['total_time'] = df_top_10['food_preparation_time'] + df_top_10['de


# Create a catplot for restaurant name, day of the week, cuisine type, and r
plt.figure(figsize=(15, 8))
sns.catplot(x='restaurant_name', y='total_time', hue='day_of_the_week', kinc
plt.xticks(rotation=90)
plt.xlabel('Restaurant Name')
plt.ylabel('Total Time')
plt.title('Time to complete order by Restaurant and day of the week')
plt.show()

plt.figure(figsize=(15, 8))
sns.catplot(x='cuisine_type', y='total_time', hue='day_of_the_week', kind='b
plt.xticks(rotation=90)
plt.xlabel('Cuisine Type')
plt.ylabel('Total Time')
plt.title('Time to complete order by Cuisine type and day of the week')
plt.show()

plt.figure(figsize=(15, 8))
sns.catplot(x='rating', y='total_time', hue='day_of_the_week', kind='box', c
plt.xticks(rotation=90)
plt.xlabel('Rating')
plt.ylabel('Total Time')
plt.title('Time to complete order by Rating and day of the week, Not Given t
plt.show()
```

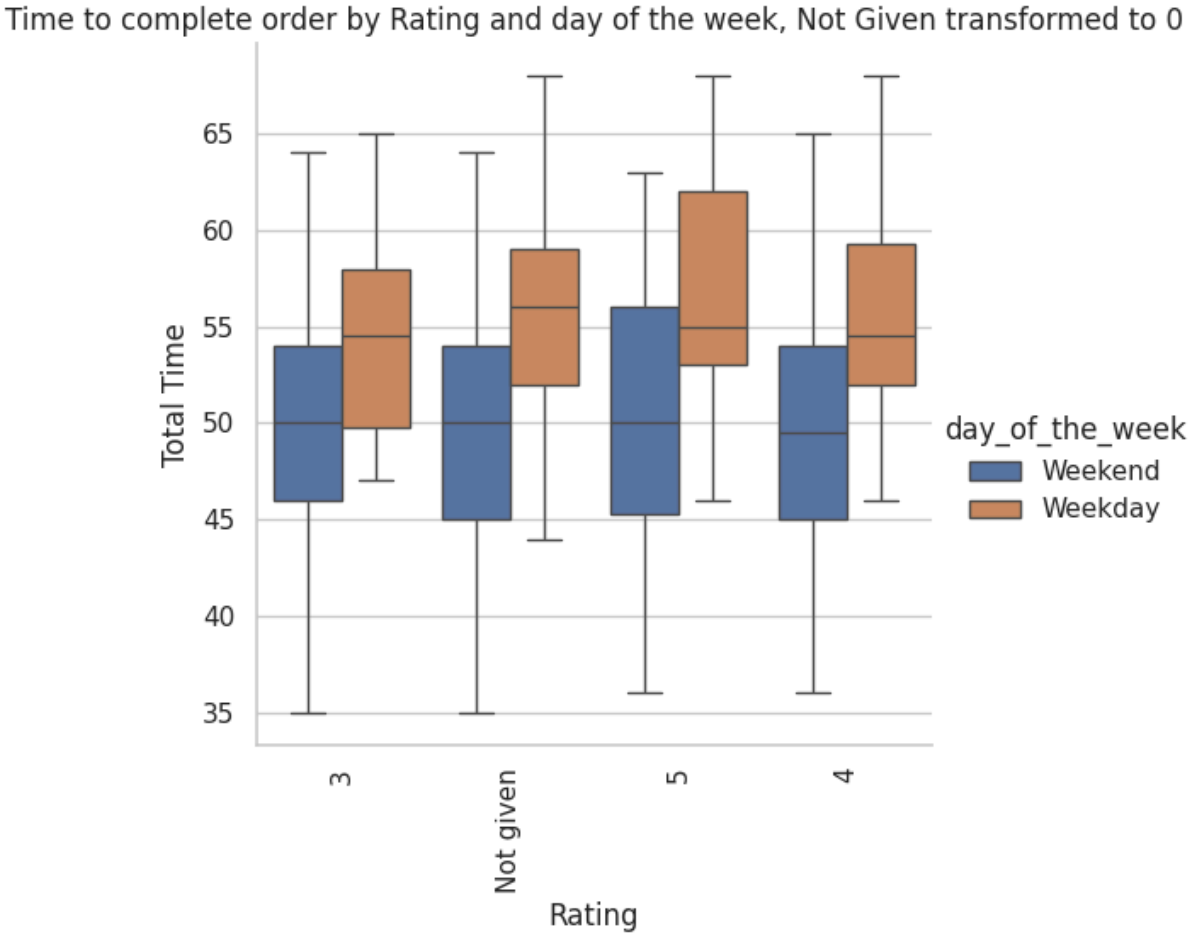
```
<ipython-input-74-689a94694a2b>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_top_10['total_time'] = df_top_10['food_preparation_time'] + df_top_10
['delivery_time']
<Figure size 1500x800 with 0 Axes>
```



<Figure size 1500x800 with 0 Axes>



<Figure size 1500x800 with 0 Axes>



Observations

RedFarm Broadway takes the longest to complete a meal on the weekday, although has one of the faster services on the weekend. Although TAO is one of the most expensive restaurants, they have one of the faster services on the weekend.

For the top 4 cuisines, it generally takes longer to complete an order on the weekday, on average more than 5 minutes, than the weekend.

Ratings do not appear to be influenced by the total time it takes to complete an order which implies that customers generally don't rate the service based on the time it takes to complete an order.

In [ ]:
```
# what is the most popular cuisine?

# Group by cuisine type and count the number of orders
cuisine_counts = df.groupby('cuisine_type')['cuisine_type'].count()

# Find the cuisine with the maximum number of orders
most_popular_cuisine = cuisine_counts.idxmax()

print(f"The most popular cuisine is: {most_popular_cuisine}")
```

```
The most popular cuisine is: American
```

Observations:

There does not appear to be any correlation between the numerical variables

On average, Korean cuisine is the least expensive and Southern is the most expensive. Of the top 10 restaurants, TAO and Blue Ribbon Fried Chicken are the most expensive and Blue Ribbon Sushi are the least expensive

American is the most popular cuisine and meals are competitively priced compared to other options.

Day of the week the order was completed or how a customer rated the purchase don't appear to be influenced by the cost of the order. Hypothesize that a customer does not rate based on cost of meal, rates on other factors. Very limited correlation between day of the week and cost of order

Considering the top 10 restaurants, the time to complete an order generally takes longer during the week. Though orders on the weekend outnumber weekday orders by more than 2 to 1.

Of the cuisines, Korean, Vietnamese and Mediteranean have cost outliers

Japanese cuisine on average takes less time to complete on any week day. American and Italian cuisine completion averages are similar.

Of the top 10 restaurants, Red Farm Broadway takes the longest to complete (complete=prep+del time) a meal on a weekday, while offering the quickest service on the weekend. TAO generally takes less time during the weekend and weekday to complete meals compared to other restaurants, and it's in the top 10 and one of the most expensive restaurants to purchase from, and serves Japanese cuisine.

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]

The rating column needs to be converted to int and it has a string value, not given. Options convert not given to binary (0), remove not given from the column.

In [ ]:
```
# Convert 'rating' column to numeric before calculating the mean
# This is done to handle any potential non-numeric values in the 'rating' co
# errors='coerce' will replace invalid parsing with NaN.
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')

# Recalculate restaurant_ratings with the now numeric 'rating' column
restaurant_ratings = df.groupby('restaurant_name')['rating'].agg(['count', '

# Filter restaurants with more than 50 rating counts and average rating grea
promotional_restaurants = restaurant_ratings[(restaurant_ratings['count'] >

# Print the restaurants fulfilling the criteria
print("Restaurants fulfilling the criteria for promotional offer:")
print(promotional_restaurants.index.tolist())
```

```
Restaurants fulfilling the criteria for promotional offer:
['Blue Ribbon Fried Chicken', 'Blue Ribbon Sushi', 'Shake Shack', 'The Meatb
all Shop']
```

In [ ]:
```
# what is the datatype and contents of the rating column

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

```
float64
[nan  5.  3.  4.]
```

Observations:

Blue Ribbon Fried Chicken, Blue Ribbon Sushi, Shake Shack and the Meatball Shop, all the in the top 10 restaurants, are eligible for the promotional offer.

The rating column was converted to float and Not Given converted to nan values.

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 [ ]:
```
# Output number of orders greater than 20 dollars multiplied by 25% and ther

# Calculate the revenue from orders costing more than $20
orders_over_20 = df[df['cost_of_the_order'] > 20]
revenue_over_20 = len(orders_over_20) * (orders_over_20['cost_of_the_order']


# Calculate the revenue from orders costing more than $5
orders_over_5 = df[(df['cost_of_the_order'] > 5) & (df['cost_of_the_order']
revenue_over_5 = len(orders_over_5) * (orders_over_5['cost_of_the_order'].me

# Calculate the total net revenue
```

```
total_revenue = revenue_over_20 + revenue_over_5

# Print the result
print(f"The net revenue generated by the company across all orders is: ${tot
```

```
The net revenue generated by the company across all orders is: $6166.30
```

Observations:

The company has generated revenue over $6,000 based on this revenue generation equation.

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 [ ]:
```
# What percentage of orders take more than 60 minutes to get delivered from

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

# Calculate the percentage of orders that take more than 60 minutes
percentage_orders_over_60 = (len(df[df['total_delivery_time'] > 60]) / len(d

# Print the result
print(f"{percentage_orders_over_60:.2f}% of the orders take more than 60 mir
```

```
10.54% of the orders take more than 60 minutes to get delivered.
```

Observations:

Approximately 11% of the orders take more than 60 minutes to get delivered.

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 [ ]:
```
# how does the mean delivery time varies over day of the week

# Group by day of the week and calculate the mean delivery time
mean_delivery_time_by_day = df.groupby('day_of_the_week')['delivery_time'].m

# Print the result
print("Mean delivery time by day of the week:")
print(mean_delivery_time_by_day)
```

```
Mean delivery time by day of the week:
day_of_the_week
Weekday    28.340037
Weekend    22.470022
Name: delivery_time, dtype: float64
```

Observations:

The average delivery time is 6 minutes longer during the weekday, around ~28 min. This is surprising as one could hypothesize that the delivery times would be longer on the weekend, when most orders occur by a factor of more than 2 to 1. What could influence 6 minutes longer delivery time on the weekday when there are less orders? Though ~28 min. represents approximately 8% of the data on that variable, with high frequency in the data and lies toward IQR3 portion of data.

In [ ]:
```
# Count the number of orders with a delivery time of 28 minutes
count_28_minutes = len(df[df['delivery_time'] == 28])

# Calculate the percentage of orders with a delivery time of 29 minutes
percentage_28_minutes = (count_28_minutes / len(df)) * 100

# Print the results
print(f"Number of orders with a delivery time of 28 minutes: {count_28_minut
print(f"Percentage of orders with a delivery time of 28 minutes: {percentage
```

```
Number of orders with a delivery time of 28 minutes: 148
Percentage of orders with a delivery time of 28 minutes: 7.80%
```

Observations:

The percentage of orders with an average delivery time of 28 minutes is around 8%. 148 orders out of 1898 orders takes 28 minutes to deliver.

In [ ]:
```
# what is the average complete time, preparation time + delivery time for al

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

# Calculate the average total time for all orders
average_total_time = df['total_time'].mean()

print(f"The average complete time (preparation time + delivery time) for all
```

```
The average complete time (preparation time + delivery time) for all orders
is: 51.53 minutes.
```

Observations

The average completion time for an order is approximately 52 minutes.

Conclusion and Recommendations

Question 17: What are your conclusions from the analysis? What recommendations would you like to share to help improve the business? [6 marks]

Conclusions:

- We have analyzed the foodhub dataset from the perspective of sales occurring either on a weekday or weekend, performing descriptive statistics and visual distributions analysis on individual numeric variables and multiple numeric and categorical variables.
- The objective of the analysis is to get a fair idea about the demand of different restaurants which will help them in enhancing their customer experience.
- The top 10 restaurants were analyzed and compared, and we learned that American cuisine is the most popular, especially on the weekend where restaurants are busiest, and is competitively priced compared to other cuisines. Shake Shack is the most popular restaurant in the group based on significant sales and orders, specializing in American cuisine. The top 10 restaurants sell the top 4 cuisines.
- 10% of orders take more than 60 minutes to deliver to the customer, although longer order completion times do not to appear to affect how a customer rates the service.
- Using the current revenue generation equation, calculated the revenue generated ($6000) from the orders contained in the dataset.
- Orders take longer to complete during the week, and delivery times are 6 minutes longer than the weekend, despite the weekend having over 2 times more orders than the week.
- The top ten restaurants combine for approximately 45% of all sales and orders, complete significantly more orders during the weekend than the week, hence most of the revenue is generated on the weekend.
- The top 4 cuisines account for 80% of sales, orders.
- Hypothesize that weekday orders take longer to complete, based on delivery time, due to factors unaccounted for at this time. It appears that restaurants have optimized revenue generation on the weekend, with more orders generating more revenue than the week, with shorter delivery times than the week.

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

Recommendations:

- Target restaurants that have high sales, orders to further analyze what are most popular cuisines and at what time.
- Target restaurant customers through further analysis of purchasing patterns. Key is to include customer demographic information in the dataset.
- Southern cuisine is the most expensive across orders, with an average price over 19 per order and Blue Ribbon Fried Chicken, the number 4 restaurant in orders serve this cuisine as a specialty. This is a great opportunity to perform and measure a rewards campaign that influences customer behavior on selecting Southern cuisine to increase foodhub's revenue in addition to increasing brand awareness for Blue Ribbon Fried Chicken, and other restaurants that serve this cuisine.
- TAO, even though it is one of the most expensive restaurants, is one of the most popular, serving popular Japanese cuisine. This is another great opportunity to perform and measure a rewards campaign that influences customer behavior on selecting Japanese cuisine to increase foodhub's revenue in addition to increasing brand awareness for TAO and other restaurants that serve this cuisine.
- Without hesitation, a rewards campaign should be performed to continue to influence customer behavior and increase brand awareness for the category leaders; American cuisine through Shake Shack and Italian cuisine through the Meatball Shop and Parm. These cuisine and restaurant leaders contribute significantly to foodhub's current revenue.
- The rating system needs further definition and accurate collection from customer responses to be able to learn how meal cost, service, cuisine and other factors affect ratings. This will aid in performing and measuring restaurant customer reward campaigns.
- Perform and measure reward campaigns for restaurants and customers based on criteria of cost and cuisine that will increase restaurant participation across all restaurants, and also increase customer visits.
- While keeping the day of the week variable, create a day/time variable and accurately collect this information to learn how factors change over time.
- Verify restaurant information that appears to be duplicated, and address outliers in cost based on the restaurant or cuisine being served.
- Attempt to determine what factors are influencing the increased time it takes to deliver a meal during the week, if these factors can be adjusted to improve delivery times, if that is a management objective.

In [ ]:
```
#!pip install nbconvert
#!jupyter nbconvert /content/drive/MyDrive/uoftaiml/proj1/Michael_Adams_Note
#!jupyter nbconvert /content/drive/MyDrive/uoftaiml/proj1/Michael_Adams_Note
#!jupyter nbconvert --to pdf /content/drive/MyDrive/uoftaiml/proj1/Michael_A
```