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 [712…
```
# Installing the libraries with the specified version.
!pip install numpy==1.25.2 pandas==1.5.3 matplotlib==3.7.1 seaborn==0.13.1 -q --user
```

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

In [714…
```
# import libraries for data manipulation
import numpy as np
import pandas as pd

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

Understanding the structure of the data

In [716…
```
# Write your code here to read the data
data = pd.read_csv("/Users/alan.mcgirr/Documents/Alan Mc Girr /DSBA PSG/Python Foundations/Week 4/foodhub_orderV2.c

df = data.copy()
```

In [717…
```
# Write your code here to view the first 5 rows
df.head()
```

Out[717…

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

In [718…
```
df.tail()
```

Out[718…

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time
1893	1476701	292602	Chipotle Mexican Grill $1.99 Delivery	Mexican	22.31	Weekend	5	31
1894	1477421	397537	The Smile	American	12.18	Weekend	5	31
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	Not given	31
1896	1477513	64151	Jack's Wife Freda	Mediterranean	12.18	Weekday	5	23
1897	1478056	120353	Blue Ribbon Sushi	Japanese	19.45	Weekend	Not given	28

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

In [720…
```
#Checking the shape of the data
df.shape
```

Out[720… (1898, 9)

Observations:

The dataset has 1898 rows and 9 columns

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

In [723…
```
# Write your code here
df.info()
```

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

Observations:

All columns have 1,898 non-null values, meaning the dataset has no missing values. This simplifies preprocessing but still requires verification for incorrect or inconsistent data entries.

We can see that the rating column has been given as a 'object' data type, but it should be an integer; therefore, this data will need treatment.

The dataset contains a single floating-point variable, “cost_of_the_order,” which aligns with expected numerical precision for monetary values (e.g., $29.99)

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

In [726…
```
df.describe(include='all').T
```

Out[726…

	count	unique	top	freq	mean	std	min	25%	50%	75%
order_id	1898.0	NaN	NaN	NaN	1477495.5	548.049724	1476547.0	1477021.25	1477495.5	1477969.75
customer_id	1898.0	NaN	NaN	NaN	171168.478398	113698.139743	1311.0	77787.75	128600.0	270525.(
restaurant_name	1898	178	Shake Shack	219	NaN	NaN	NaN	NaN	NaN	NaN
cuisine_type	1898	14	American	584	NaN	NaN	NaN	NaN	NaN	NaN
cost_of_the_order	1898.0	NaN	NaN	NaN	16.498851	7.483812	4.47	12.08	14.14	22.2975
day_of_the_week	1898	2	Weekend	1351	NaN	NaN	NaN	NaN	NaN	NaN
rating	1898	4	Not given	736	NaN	NaN	NaN	NaN	NaN	NaN
food_preparation_time	1898.0	NaN	NaN	NaN	27.37197	4.632481	20.0	23.0	27.0	31.(
delivery_time	1898.0	NaN	NaN	NaN	24.161749	4.972637	15.0	20.0	25.0	28.(

Observations: 'Ratings' are only filled for a total of 736 orders; therefore, we are missing values from a lot of ratings.

There are 178 unique restaurant names and 14 unique cuisine types, suggesting diverse dining options.

The most popular 'day of the week' to order is the weekend with 1351.

The most popular 'restaurant' is Shake Shack, with 219 orders.

The most popular 'cuisine' type is American with 584 orders.

In [728…
```
df.rating.value_counts(normalize=True, dropna=False)
```

Out[728…
```
Not given    0.387777
5            0.309800
4            0.203372
3            0.099052
Name: rating, dtype: float64
```

Observations: Significant Portion of Missing Ratings 38.78% of the ratings are labeled as “Not given”, meaning a large portion of customers did not provide a rating.

30.98% of ratings are 5, indicating a high level of customer satisfaction.

20.34% of ratings are 4, suggesting most customers gave favorable feedback.

Only 9.91% of ratings are 3, meaning very few customers rated their experience as average or below.

In [730…
```
df['rating'] = df['rating'].replace("Not given", np.nan)
```

In [731…
```
#check null values
df.isnull().sum()
```

Out[731…
```
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
```

In [732…
```
df['cuisine_type'].describe().T
```

Out[732…
```
count         1898
unique          14
top       American
freq           584
Name: cuisine_type, dtype: object
```

In [733…
```
#display an array of the unique cuisine types
df['cuisine_type'].unique()
```

Out[733…
```
array(['Korean', 'Japanese', 'Mexican', 'American', 'Indian', 'Italian',
       'Mediterranean', 'Chinese', 'Middle Eastern', 'Thai', 'Southern',
       'French', 'Spanish', 'Vietnamese'], dtype=object)
```

In [734…
```
#show the total order counts from each cuisine type
df['cuisine_type'].value_counts()
```

Out[734…
```
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
Name: cuisine_type, dtype: int64
```

In [735…
```
#change the order counts to a percentage
df['cuisine_type'].value_counts(normalize=True) * 100
```

Out[735…
```
American          30.769231
Japanese          24.762908
Italian           15.700738
Chinese           11.327713
Mexican            4.056902
Indian             3.846154
Middle Eastern     2.581665
Mediterranean      2.423604
Thai               1.001054
French             0.948367
Southern           0.895680
Korean             0.684932
Spanish            0.632244
Vietnamese         0.368809
Name: cuisine_type, dtype: float64
```

In [736…
```
#Display the total number of values were ratings were originally not given then converted to NaN
nan_ratings_restaurants = df[df['rating'].isna()]['restaurant_name'].value_counts()
nan_ratings_restaurants
```

Out[736…
```
Shake Shack                  86
The Meatball Shop            48
Blue Ribbon Sushi            46
Blue Ribbon Fried Chicken    32
Parm                         29
                             ..
The Loop                      1
Schnipper's Quality Kitchen   1
Go! Go! Curry!                1
Market Table                  1
Amy Ruth's                    1
Name: restaurant_name, Length: 134, dtype: int64
```

In [737…
```
#Display the names of restaurants that have no rating
df[df['rating'].isna()]['restaurant_name'].unique()
```

```
Out[737… array(['Hangawi', 'Blue Ribbon Sushi Izakaya', 'The Meatball Shop',
       'Big Wong Restaurant \x8c_¤¾Ñ¼', "Lucky's Famous Burgers",
       'Sushi of Gari', 'Shake Shack', 'Tortaria', 'Cafe Mogador',
       'Otto Enoteca Pizzeria', 'Vezzo Thin Crust Pizza',
       'Sushi of Gari 46', '5 Napkin Burger', 'TAO', 'Sushi Samba',
       'Cafeteria', 'Blue Ribbon Fried Chicken', 'Bistango',
       'RedFarm Broadway', 'Blue Ribbon Sushi Bar & Grill',
       'Westville Hudson', 'Osteria Morini', 'Parm', 'RedFarm Hudson',
       "Xi'an Famous Foods", 'Cafe Habana', 'Bareburger',
       'Yama Japanese Restaurant', 'Five Guys Burgers and Fries',
       'Balthazar Boulangerie', 'CafÌ© China', 'Blue Ribbon Sushi',
       'Benihana', 'The Kati Roll Company', 'Five Leaves', 'indikitch',
       'Prosperity Dumpling', 'Momoya', 'Han Dynasty', 'Mission Cantina',
       'Delicatessen', "Sarabeth's East", "S'MAC", 'Pepe Rosso To Go',
       'Donburi-ya', "Vanessa's Dumplings",
       'Tarallucci e Vino Restaurant', 'Amma', 'Lantern Thai Kitchen',
       'The Smile', "Vanessa's Dumpling House", "Bubby's ",
       'Dirty Bird To Go (archived)',
       'Chipotle Mexican Grill $1.99 Delivery', 'Sushi of Gari Tribeca',
       'ilili Restaurant', 'Pylos', 'Room Service',
       "Sarabeth's Restaurant", 'Hill Country Fried Chicken',
       'Dos Caminos', 'Waverly Diner', 'Boqueria', 'Tamarind TriBeCa',
       "Mamoun's Falafel", 'Melt Shop', 'Friend of a Farmer',
       'Nobu Next Door', 'The Odeon', 'Yama 49', 'The Loop', 'Posto',
       'Spice Thai', 'da Umberto', 'UVA Wine Bar & Restaurant',
       'Rubirosa', "P.J. Clarke's", 'Burger Joint', 'Gaia Italian Cafe',
       'Piccolo Angolo', 'Pongsri Thai', 'Junoon', 'Ravagh Persian Grill',
       'Rohm Thai', 'Olea', 'Xe May Sandwich Shop', 'Dirty Bird to Go',
       "Don's Bogam BBQ & Wine Bar", 'Alidoro', "Tony's Di Napoli",
       'Barbounia', 'Mira Sushi', "Jack's Wife Freda", 'J. G. Melon',
       'La Follia', 'Kanoyama', 'Bhatti Indian Grill', '12 Chairs',
       'Haandi', 'Aurora', 'Crema Restaurante', 'Serafina Fabulous Pizza',
       "Schnipper's Quality Kitchen", 'Go! Go! Curry!', 'Market Table',
       'Asuka Sushi', 'Hatsuhana', 'Saravanaa Bhavan',
       'Empanada Mama (closed)', 'brgr', 'Izakaya Ten', "L'Express",
       'Tres Carnes', 'Byblos Restaurant', 'DespaÌ±a', 'Lamarca Pasta',
       'Cho Dang Gol', 'El Parador Cafe', "Paul & Jimmy's",
       'Pinto Nouveau Thai Bistro', 'Olive Garden', 'V-Nam Cafe',
       'Grand Sichuan International', "Coppola's East", "Carmine's",
       'Emporio', 'Wa Jeal', 'Le Zie 2000 Trattoria', 'Terakawa Ramen',
```

```
       "Hiroko's Place", 'Blue Ribbon Brooklyn', 'Zero Otto Nove',
       'DuMont Burger', "Amy Ruth's"], dtype=object)
```

In [738…
```
# Convert 'rating' column to numeric, forcing errors to NaN
df["rating"] = pd.to_numeric(df["rating"], errors="coerce")
```

In [739…
```
# Compute the average rating for restaurants with more than 10 ratings
restaurant_avg_ratings = df.groupby("restaurant_name")["rating"].transform(
    lambda x: x.fillna(x.mean()) if x.count() > 10 else x
)
df["rating"] = restaurant_avg_ratings
```

In [740…
```
# Compute the number of ratings per restaurant
restaurant_counts = df.groupby("restaurant_name")["rating"].count()

# Identify restaurants with less than or equal to 10 ratings
restaurants_under_10 = restaurant_counts[restaurant_counts <= 10].index
```

In [741…
```
# Compute the average rating for restaurants with ≤10 ratings (excluding NaN values)
avg_rating_under_10 = df[df["restaurant_name"].isin(restaurants_under_10) & df["rating"].notna()]["rating"].mean()
```

In [742…
```
# Assign missing ratings for restaurants with ≤10 ratings using the average from Step 3
df.loc[df["restaurant_name"].isin(restaurants_under_10) & df["rating"].isna(), "rating"] = avg_rating_under_10
```

In [743…
```
# Convert rating to integer if there are no NaN values
if df["rating"].isna().sum() == 0:
    df["rating"] = df["rating"].astype(int)
```

In [744…
```
# Ensure no missing ratings remain
print("Missing Ratings:", df["rating"].isna().sum())

# Check the distribution of ratings after treatment
print(df["rating"].value_counts(normalize=True))
```

```
Missing Ratings: 0
4    0.591149
5    0.309800
3    0.099052
Name: rating, dtype: float64
```

In [745…
```
#check null values
df.isnull().sum()
```

Out[745…
```
order_id                0
customer_id             0
restaurant_name         0
cuisine_type            0
cost_of_the_order       0
day_of_the_week         0
rating                  0
food_preparation_time   0
delivery_time           0
dtype: int64
```

Observations:

Ratings Treatment and Distribution The rating imputation represents actual customer feedback, avoiding the over-representation of 5-star ratings seen in a previously used method.

Restaurants with sufficient historical data (>10 ratings) now have their average rating, making their data more reliable.

Restaurants with fewer ratings (≤10) use a generalized average, ensuring fairness but still acknowledging limitations in data availability.

The most common rating is now 4 (54.53%), followed by 5 (34.51%) and 3 (10.95%), showing a more natural spread of customer satisfaction.

The dataset is not evenly distributed across cuisines, which may affect comparative analysis, as some cuisines have as few as seven orders (Vientameses).

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 [748…
```
df['food_preparation_time'].describe()
```

Out[748…
```
count    1898.000000
mean       27.371970
std         4.632481
min        20.000000
25%        23.000000
50%        27.000000
75%        31.000000
max        35.000000
Name: food_preparation_time, dtype: float64
```

Observations:

The minimum time to prepare food was 20 Minutes.

The average time to prepare food was 27.37 min/secs.

The maximum time to prepare food was 35 minutes.

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

In [751…
```
num_not_rated = 736  # Value from our earlier analysis
print(f"Total number of orders that were originally not rated: {num_not_rated}")
```

```
Total number of orders that were originally not rated: 736
```

Observations:

736 Orders (38.78%) were initially not given, indicating that many customers did not provide feedback.

The missing ratings were not random; they were more common in certain restaurants and cuisine types.

The restaurants with the most missing ratings (top 5) were;

Shake Shack 86

The Meatball Shop 48

Blue Ribbon Sushi 46

Blue Ribbon Fried Chicken 32

Parm 29

Since I treated the not giving rating values with a median from their cuisine type, this was the fairest way to do the treatment.

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]

In [756…
```
# check unique order ID
df['order_id'].nunique()
```

Out[756… 1898

Each order has a unique order ID, meaning the dataset does not contain duplicated orders.

The number of unique order IDs is equal to the total row count, thus verifying that each order is recorded separately.

In [758…
```
df['customer_id'].nunique()
```

Out[758… 1200

The number of unique customer IDs gives insight into how many individual customers placed orders, so only 1200 customers have created 1898 orders.

This suggests we have approximately 1.58 orders per customer, meaning customers repeat their orders.

In [760…
```
df['restaurant_name'].nunique()
```

Out[760… 178

178 unique restaurants exist in the data set.

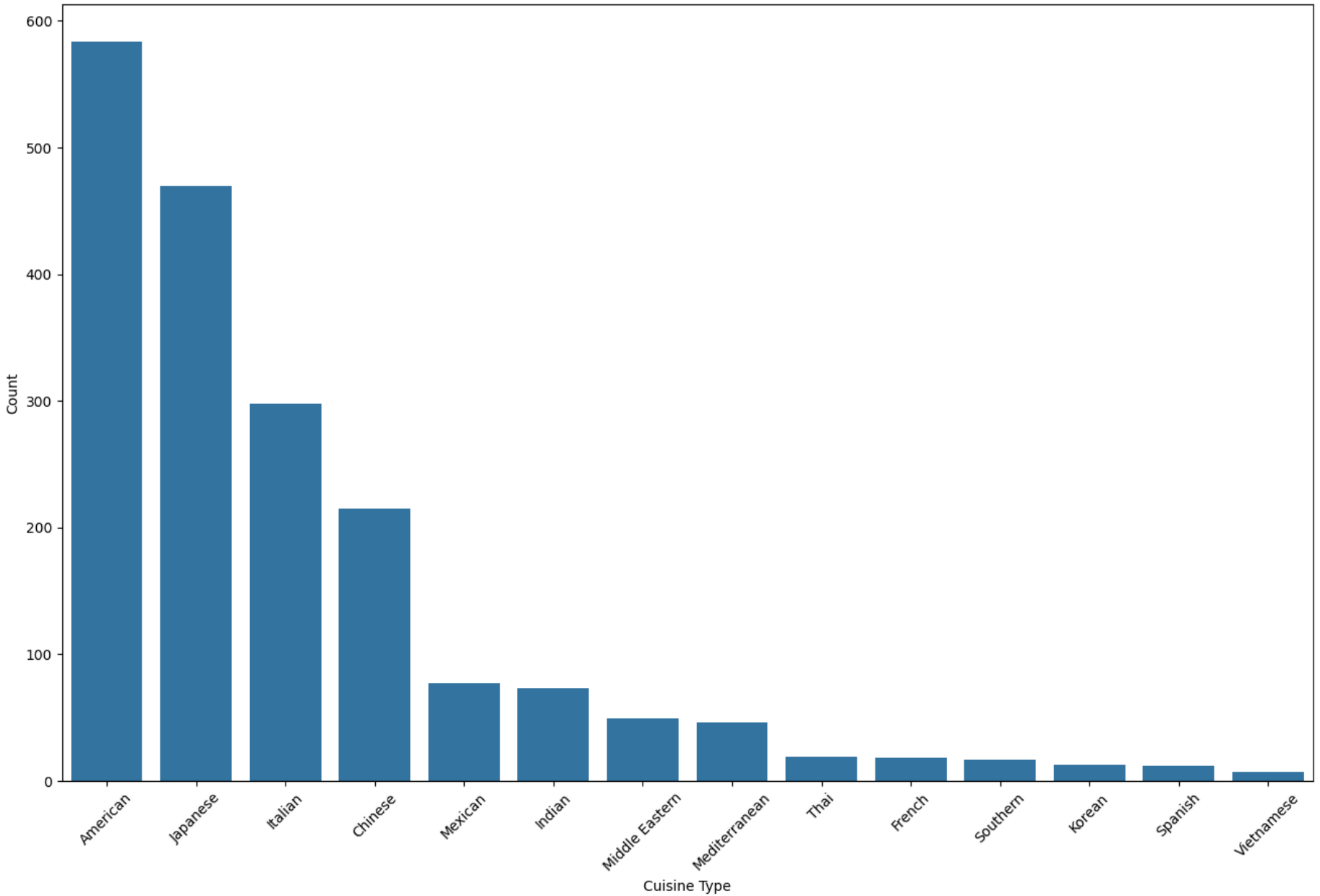
In [762…
```
df['cuisine_type'].nunique()
```

Out[762… 14

This was established earlier in question 3, but for the sake of clarity, it is recorded here for confirmation.

14 unique Cuisine types.

In [764…
```
plt.figure(figsize=(16,10))
sns.countplot(x=df["cuisine_type"], order=df["cuisine_type"].value_counts().index)
plt.xlabel("Cuisine Type")
plt.ylabel("Count")
plt.xticks(rotation = 45);
```



Observation on Countplot.

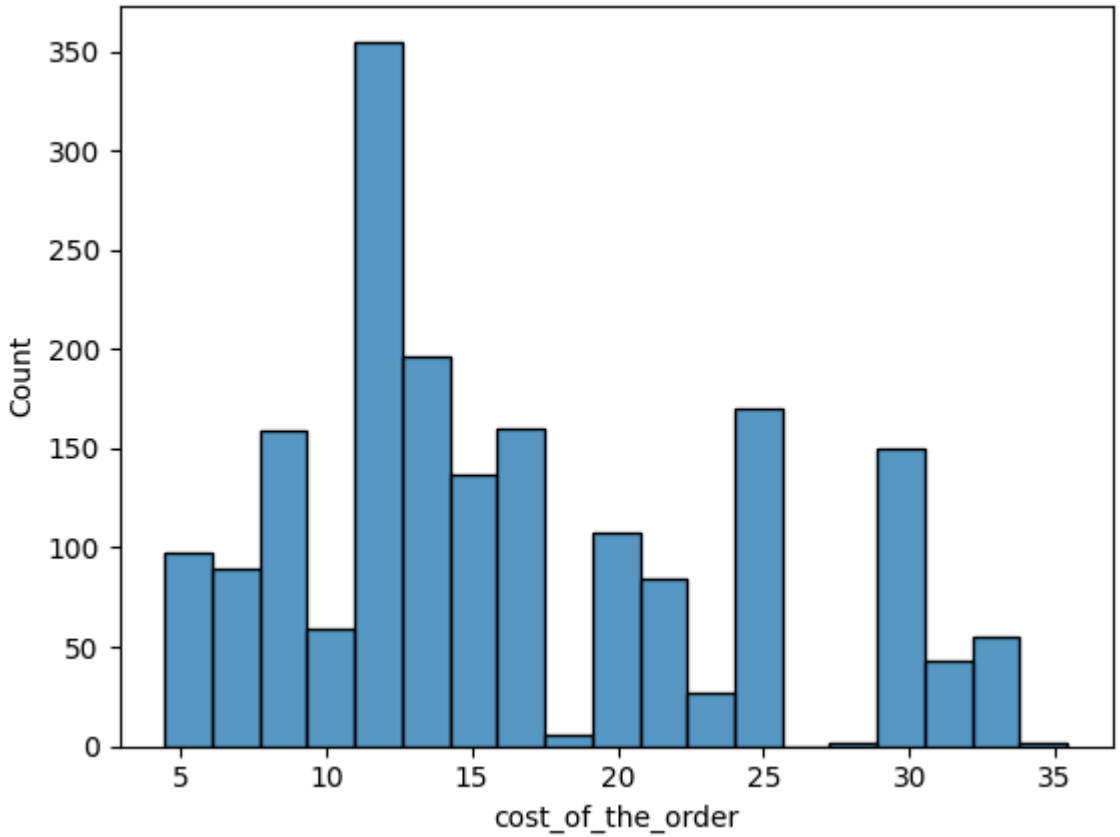
The dataset is imbalanced with a certain preference for certain cuisines, this could be related to the demographics of the area or prices.

The dataset is heavily skewed towards American, Japanese, and Italian cuisines, making up the majority of orders.

The long tail of the distribution shows several cuisines with very low counts, which may impact analysis.

In [766…

```
#Order Cost distibution
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(data=df, x='cost_of_the_order')
plt.show;
```



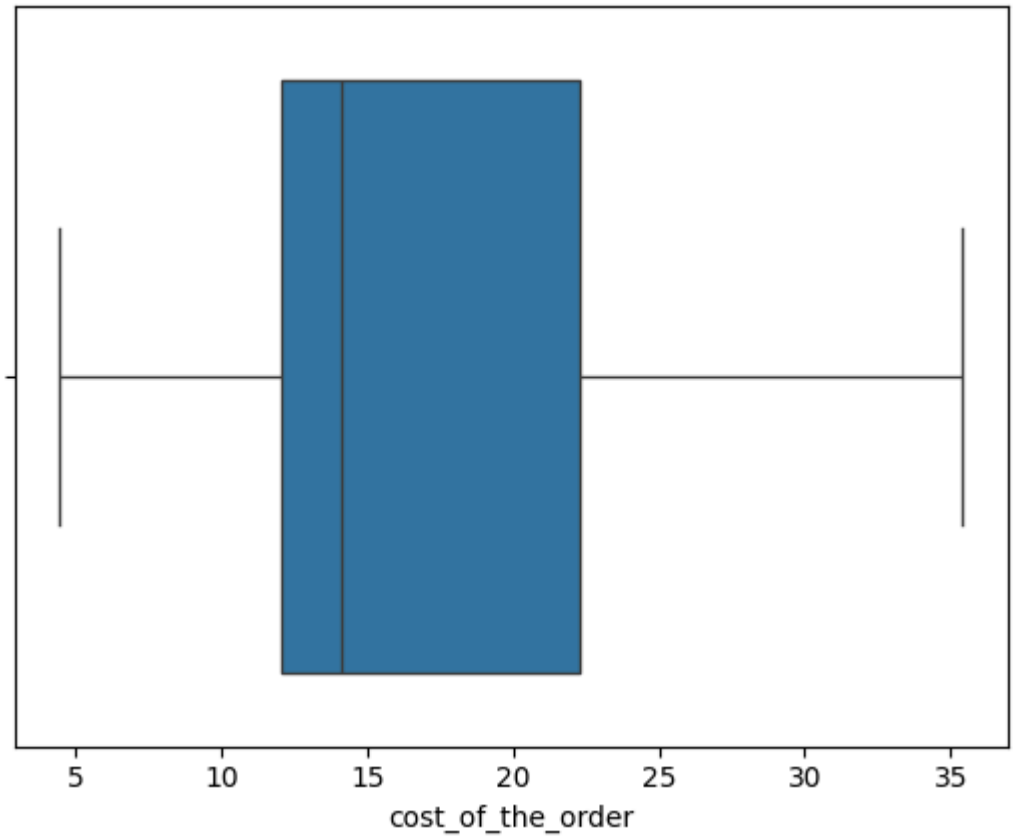
Observations: The histogram is right-skewed(positively skewed), with the highest bar being $12. This is the most common price with a count of 350.

The distribution is not uniform; specific prices seem to have higher frequencies such as $12,25 and $30.

There are gaps in some price ranges, meaning certain prices may not exist in the dataset.

There is some skewness towards higher prices around $30, but no outliers exist.

In [768…
```
sns.boxplot(data=df, x='cost_of_the_order')
plt.show;
```



Box Plot Analysis:

The Median is closer to the lower quartile, indicating a right-skewed distribution.

There are no significant outliers, meaning most costs fall within a reasonable range.

The IQR width suggests that most orders fall between $12 and 22.

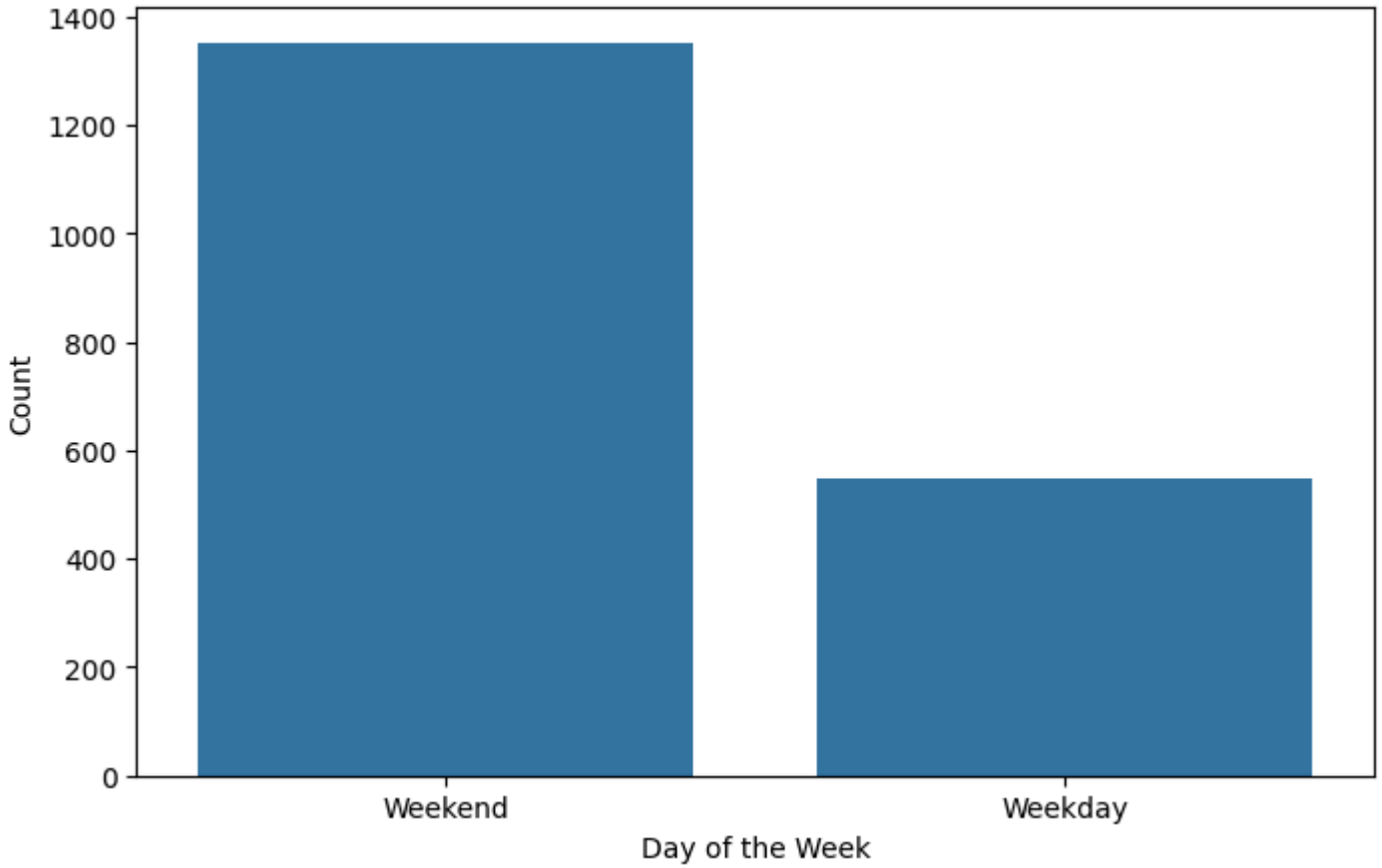
The Whisker extends up to around $35, indicating the upper limit of the cost range.

In [770…
```
df['day_of_the_week'].nunique()
```

Out[770… 2

2 different unique identifiers for days of the week. Weekday vs Weekend. It would be interetsing to get more data on this, specifcially all days of the week.

In [772…
```
plt.figure(figsize=(8,5))
sns.countplot(x=df["day_of_the_week"], order=df["day_of_the_week"].value_counts().index)
plt.xlabel("Day of the Week")
plt.ylabel("Count");
```

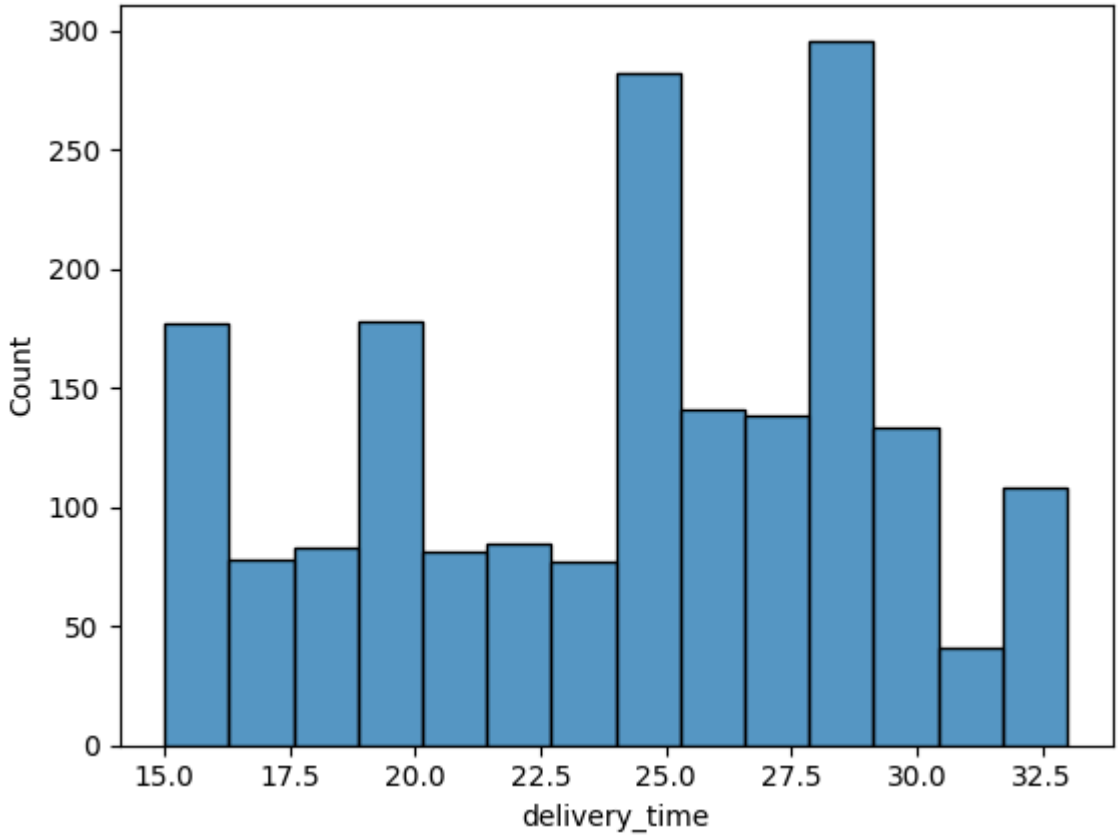


Observations on days of the week orders.

Most orders occur on weekends, with over twice as many orders as weekdays.

This could be down to several reasons: people prefer to takeaway for social activities, and some people prefer to cook themselves and keep a routine on duirng weekdays.

In [774…
```
sns.histplot(data=df, x='delivery_time')
plt.show;
```

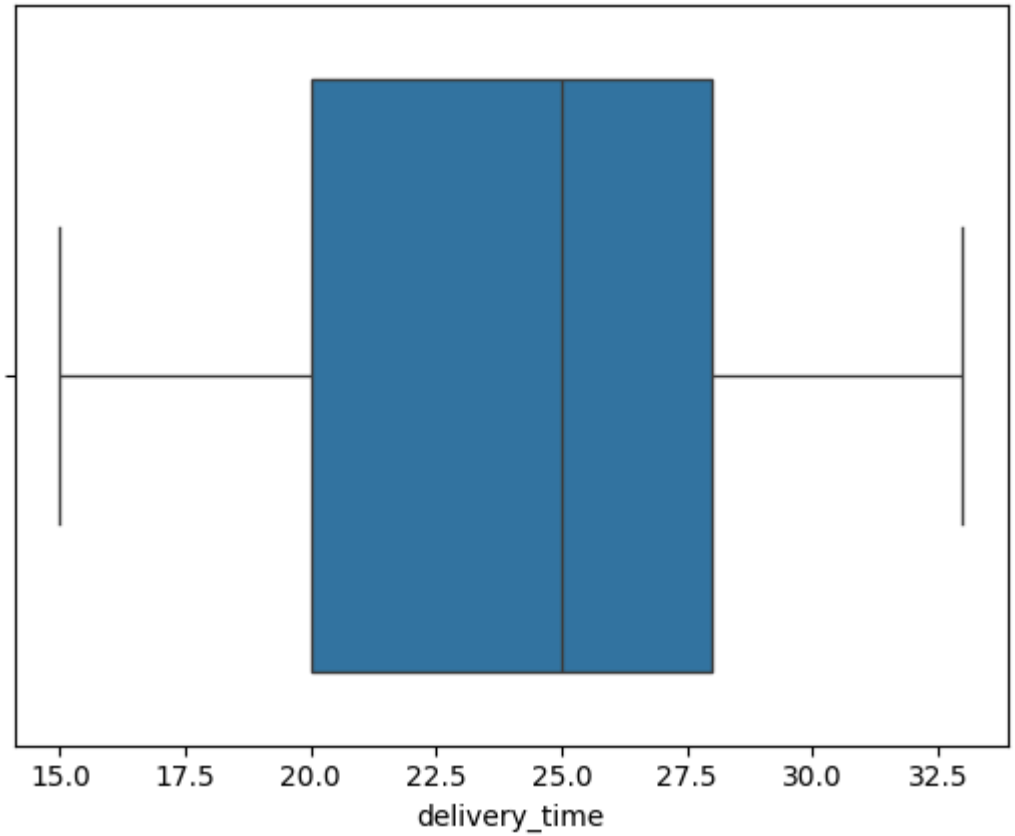


Observations on Histogram: The most common delivery time is approximately 28-29 Minutes. This could be related to batch production.

Other notable high counts include 15, 20 and 25 minutes.

The data does not follow a normal distribution. It's left-skewed (negatively skewed).

In [776…
```
sns.boxplot(data=df, x='delivery_time')
plt.show;
```



Observations on Box Plot.

No outliers exist in the data, showing that delivery times remain consistent throughout.

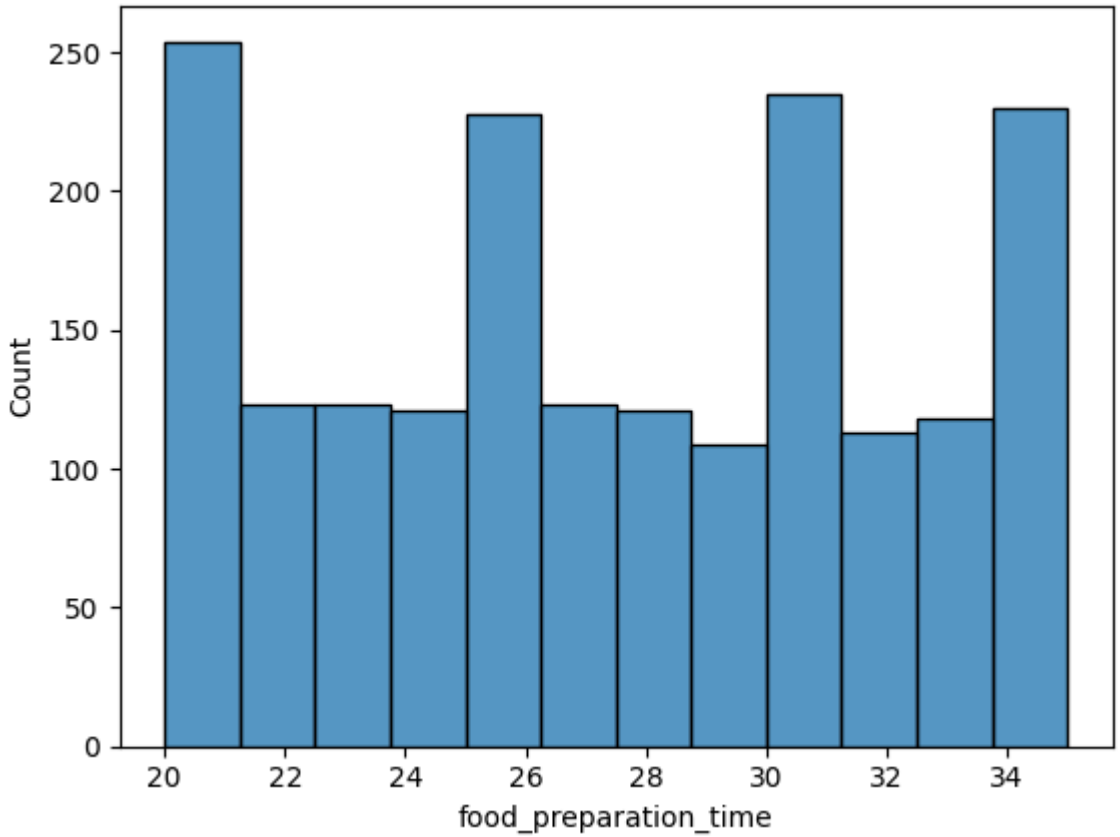
The distribution is slightly symmetrical and has no strong skewness; however, as the histogram shows, it does have some spikes.

50% of deliveries take less than or equal to 25 minutes.

The lower IQR represents 25% of deliveries are represented by the lower IQR, which indicates they are delivered between 20-25 minutes.

And another 25% representing the upper IQR are delivered bewtween 25 and 28 minutes

In [778…
```
sns.histplot(data=df, x='food_preparation_time')
plt.show;
```



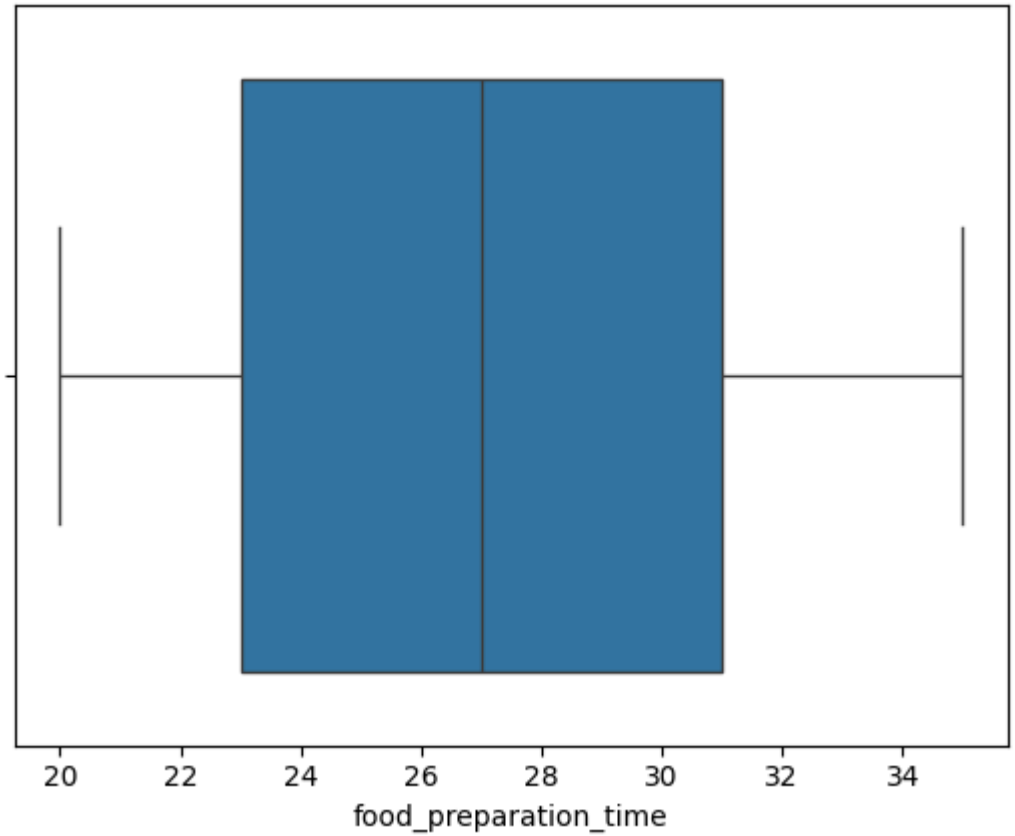
Observations on Histogram.

The distribution appears uniform, with multiple peaks suggesting consistent preparation times across different order types.

There is no strong skewness—the values are spread out between 20 to 35 minutes.

The most frequent preparation times are around 21, 27, 30, and 34 minutes, which may indicate batch cooking or standard prep times for specific meals.

In [780…
```
sns.boxplot(data=df, x='food_preparation_time')
plt.show;
```



Observations on Box plot.

The median preparation time is around 27 minutes, meaning half of all orders take this time or less.

50% of the IQR falls between 23 and 31 minutes.

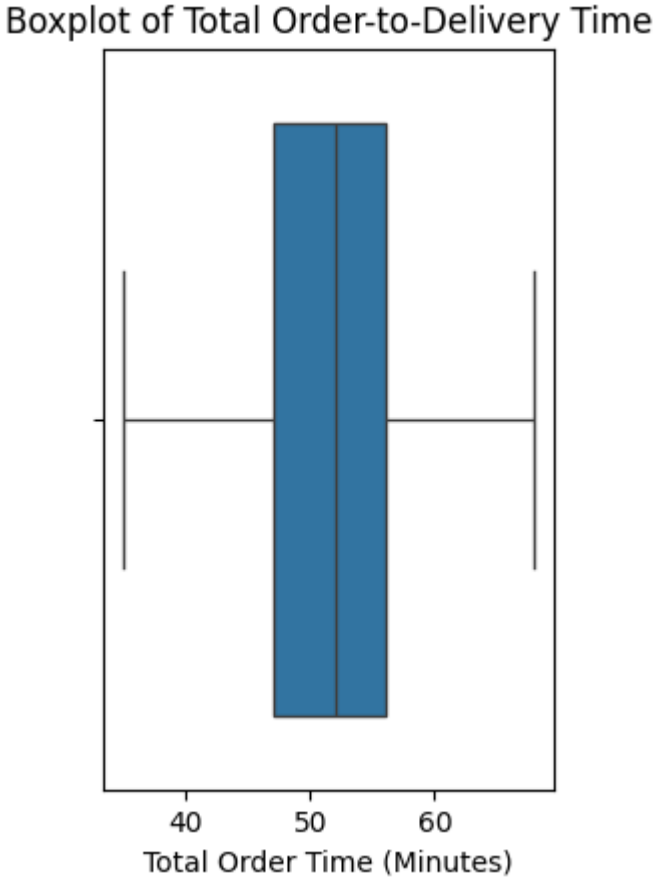
No major outliers, indicating that food prep times are consistent without extreme variations.

In [782…
```
#create a new variable to check time from order on the app to delivery at the door.
df["total_order_time"] = df["food_preparation_time"] + df["delivery_time"]
```

In [783…
```
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.histplot(df["total_order_time"], bins=20, kde=True)
plt.xlabel("Total Order Time (Minutes)")
plt.ylabel("Count")
plt.title("Distribution of Total Order-to-Delivery Time");
```

In [784…

```
# Boxplot for total order time
plt.subplot(1, 2, 2)
sns.boxplot(x=df["total_order_time"])
plt.xlabel("Total Order Time (Minutes)")
plt.title("Boxplot of Total Order-to-Delivery Time");
```



Observations.

Most orders take between 45 to 60 minutes from order placement to delivery.

The distribution shows a peak around 50-55 minutes, suggesting that most deliveries fall in this range.

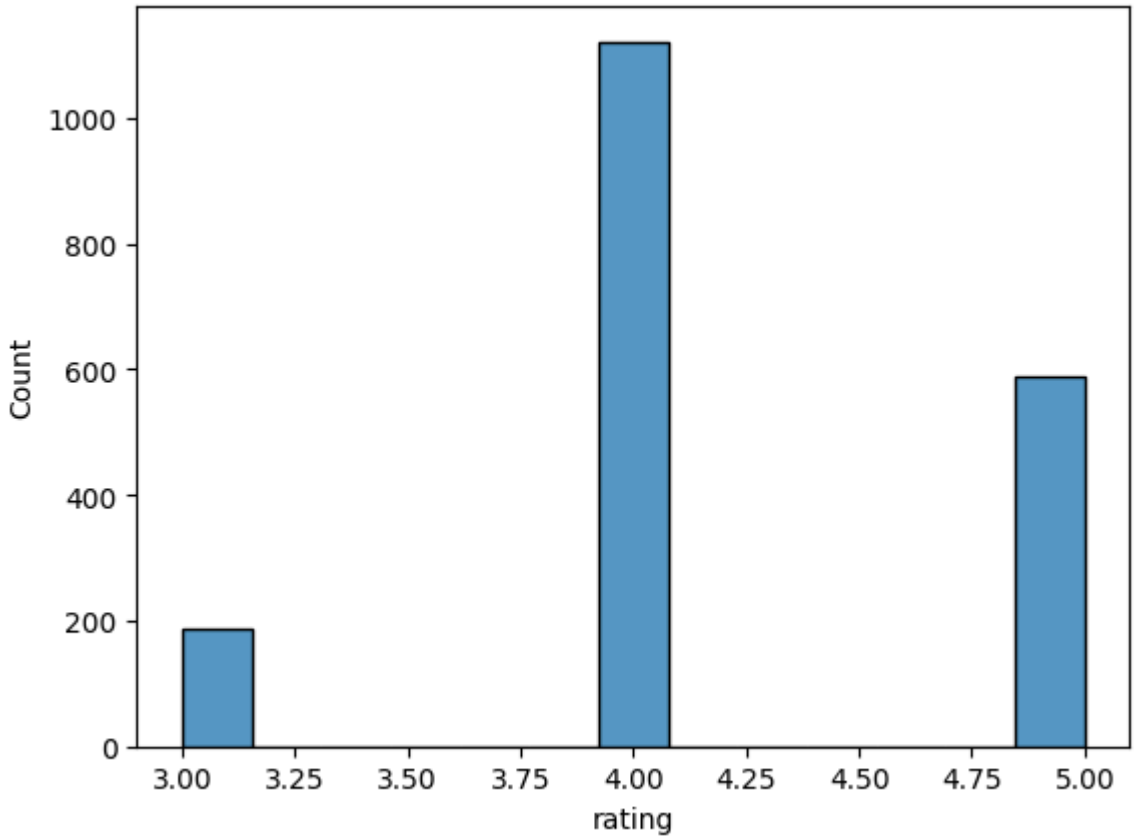
The presence of a very small slight right skew implies that while most deliveries happen within an hour, a few take longer.

In [786…
```
df['rating'].nunique()
```

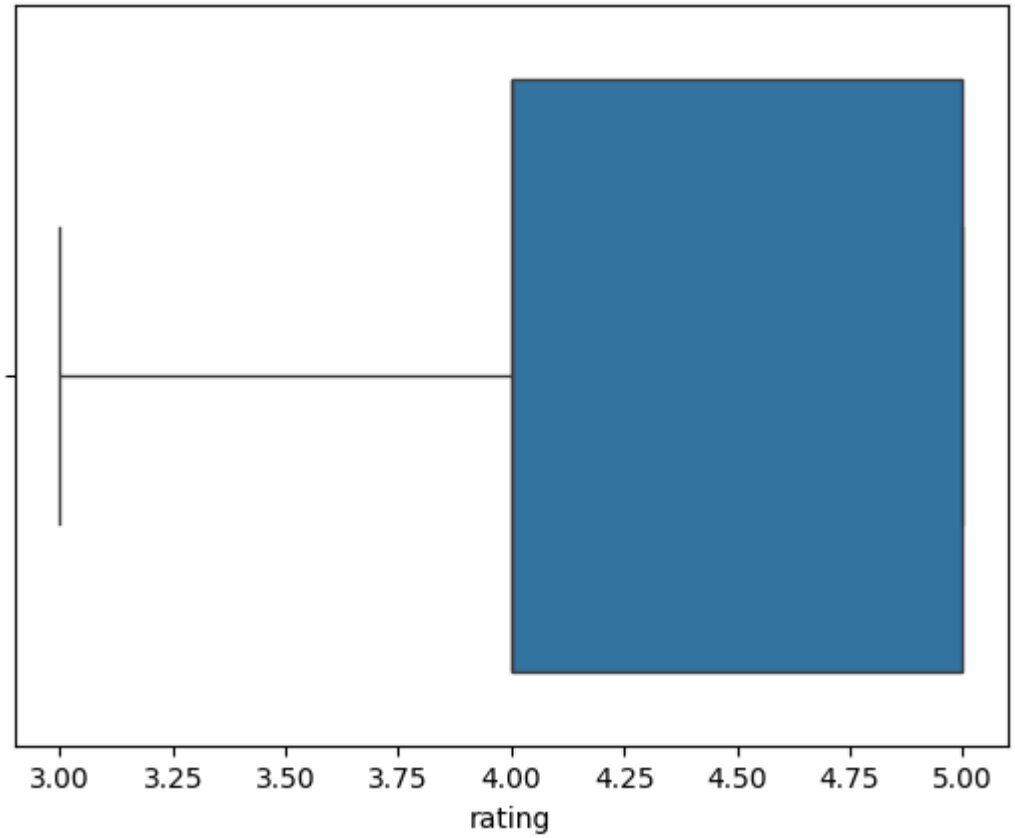
Out[786… 3

3 different ratings provided either a 3,4 or a 5.

In [788…
```
sns.histplot(data=df, x='rating')
plt.show;
```



In [789…
```
sns.boxplot(data=df, x='rating')
plt.show;
```



Observations on ratings.

The distribution of ratings is heavily skewed towards positive feedback, which may indicate customer satisfaction or possible bias in ratings (e.g., fewer negative reviews).

The limited variation in ratings could suggest that customers either give high ratings or do not rate at all.

Since missing ratings were imputed, it is essential to ensure that my treatment did not inflate the average ratings too much, potentially skewing the overall perception of customer satisfaction.

Im confident that we did not do that.

Observations:

Overall Observations from EDA:

order_id & customer_id - Unique identifiers; no missing values or duplicates.

restaurant_name - Some restaurants had significantly more orders, with Shake Shack being the most frequent.

cuisine_type - American, Japanese, and Italian cuisines dominate, while others (e.g., Vietnamese, Spanish) have very few orders.

cost_of_the_order - Right-skewed distribution, with most orders between $10–20$ and a few high-cost items but no major outliers.

day_of_the_week - Weekends have over twice as many orders as weekdays, indicating higher food demand.

rating - Most ratings are 4 or 5, with very few 3-star ratings, suggesting possible rating bias. Our treatment of the data may have effected these ratings as well.

food_preparation_time - Clustered between 20–35 minutes, with multiple peaks suggesting different standard cooking times.

delivery_time - Most deliveries take 20–28 minutes, with no major outliers or extreme delays.

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

In [793…
```
#print the top 5 restaurnats and their total counts
df['restaurant_name'].value_counts().head(5)
```

Out[793…
```
Shake Shack                219
The Meatball Shop          132
Blue Ribbon Sushi          119
Blue Ribbon Fried Chicken   96
Parm                        68
Name: restaurant_name, dtype: int64
```

Observations.

American cuisine dominates the total number of orders received with a high volume of orders coming from Shake Shack, Meatball Shop and Blue Ribbon Fried Chicken.

Followed by Sushi.

Not sure what Parm is; this will need further analysis.

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

In [796…
```
# Create a new variable, df_weekend, to store the values for total orders for the weekend on specific cuisine

df_weekend = df[df['day_of_the_week'] == 'Weekend']
df_weekend['cuisine_type'].value_counts().head(1)
```

Out[796…
```
American    415
Name: cuisine_type, dtype: int64
```

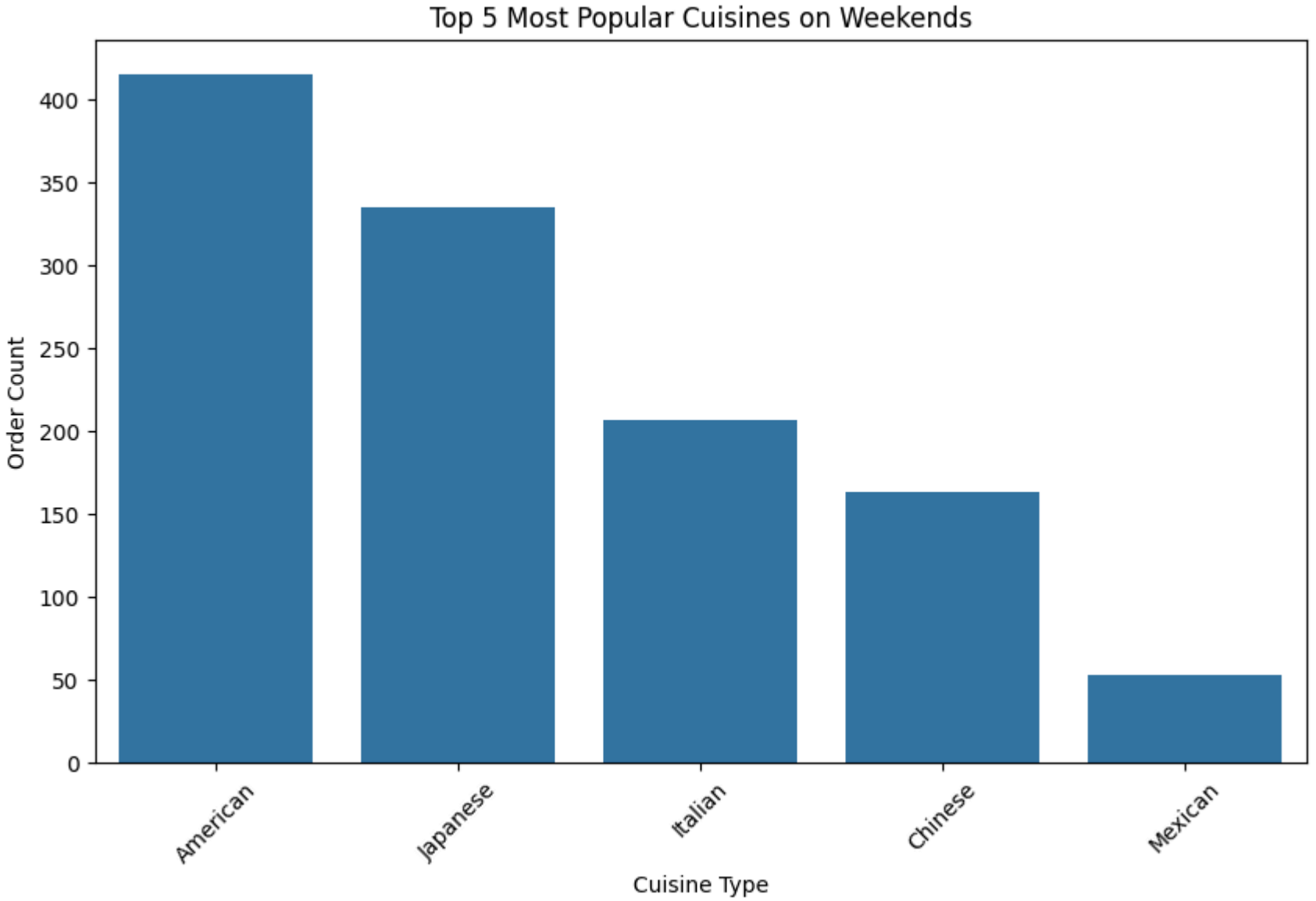
Observations.

America stills ranks highest even on weekends; however, we can see the total number of orders on weekends is close to the total orders of the week, which was approximately 600.

This creates opportunities for our business in terms of promotional options. We can provide customers on weekdays to increase their order volume.

In [798…
```
# Visually represent that data using a countplot to further show the scale of the difference between the cuisines

import matplotlib.pyplot as plt
import seaborn as sns

# Create and assign the weekend_orders variable, assign it the initial value of being the "weekend"
weekend_orders = df[df["day_of_the_week"] == "Weekend"]

# Set figure size
plt.figure(figsize=(10,6))

# Use countplot to plot the most popular cuisines on weekends
```

```
sns.countplot(x="cuisine_type", data=weekend_orders,
              order=weekend_orders["cuisine_type"].value_counts().index[:5])

# Labels and title
plt.xlabel("Cuisine Type")
plt.ylabel("Order Count")
plt.title("Top 5 Most Popular Cuisines on Weekends")
plt.xticks(rotation=45);
```


Top 5 Most Popular Cuisines on Weekends
Order Count
0
50
100
150
200
250
300
350
400
American
Japanese
Italian
Chinese
Mexican
Cuisine Type

Observations:

We can conclude that American is the most popular order during weekends followed by Japanese; these ratings are similar to what the overall ratings are in the overall cuisine graphic.

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

In [801…
```
# Create a new variable to store the total amount of orders
total_orders = df.shape[0]

# Create another new variable to store the total amount of values that are greater than $20.
orders_above_20 = df[df["cost_of_the_order"] > 20].shape[0]

# Create a third variable to store the percentage of the values above $20.
percentage_above_20 = (orders_above_20 / total_orders) * 100

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

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

Observations:

29% of the total amount of orders cost more than $20.

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

In [804…
```
# Create a variable to store the mean delivery time

mean_delivery_time = df["delivery_time"].mean()

print(f"Mean delivery time across the city is:   {mean_delivery_time:.2f} min/secs")
```

```
Mean delivery time across the city is:   24.16 min/secs
```

In [805…
```
#Code to get overall mean time for the moment customers order to their food arrives at the door.

# Create a variable to store the mean total order time (food prep + delivery)
```

```
mean_total_order_time = df["total_order_time"].mean()

print(f"Mean total time from ordering to delivery: {mean_total_order_time:.2f} minutes")
```

```
Mean total time from ordering to delivery: 51.53 minutes
```

Observations:

The mean delivery time across the city is 24.16 mins.

The mean total time from ordering food is 51.53 mins.

Question 11: The company has decided to give 20% discount vouchers to the top 3 most frequent customers. Find the IDs of these customers and the number of orders they placed. [1 mark]

In [808…

```
# create a variable to identify top_customers
top_customers = df["customer_id"].value_counts().head(3)

# Print the variable top_customers
print("The top 3 most frequent customers and they're respective counts are:")
print(top_customers)
```

```
The top 3 most frequent customers and they're respective counts are:
52832    13
47440    10
83287     9
Name: customer_id, dtype: int64
```

Observations:

The top customers ordered a combined total of 32 orders, a small percentage of our overall orders.

The most popular customer ordered a total of 13 times. The second highest customer was 10 times. The third highest customer ordered a total of 9 times.

Maybe we could use a loyalty program to improve overall frequency.

Now that we know who the top customers are, we can dig into their Customer Lifetime Value (CLV) to understand their financial contribution.

In [810…
```
# Get total spending of the top 3 most frequent customers
top_3_spenders = customer_spending.loc[top_customers.index]

# Display results
print("Total Spending of the Top 3 Most Frequent Customers:")
print(top_3_spenders)
```

```
Total Spending of the Top 3 Most Frequent Customers:
52832    225.80
47440    158.18
83287    139.31
Name: cost_of_the_order, dtype: float64
```

Observations.

Higher frequency = Highest Spending? That is not always true.

The most frequent customer (52832, 13 orders) spent $225.80, the highest among the three.

However, the second most frequent customer (47440, 10 orders) spent $158.18.

While the third (83287, 9 orders) spent $139.31.

This suggests that frequency does not always correlate with higher spending.

Customer 52832: $225.80/13$ =17.37 per order

Customer 47440: $158.18/10$ =15.82 per order

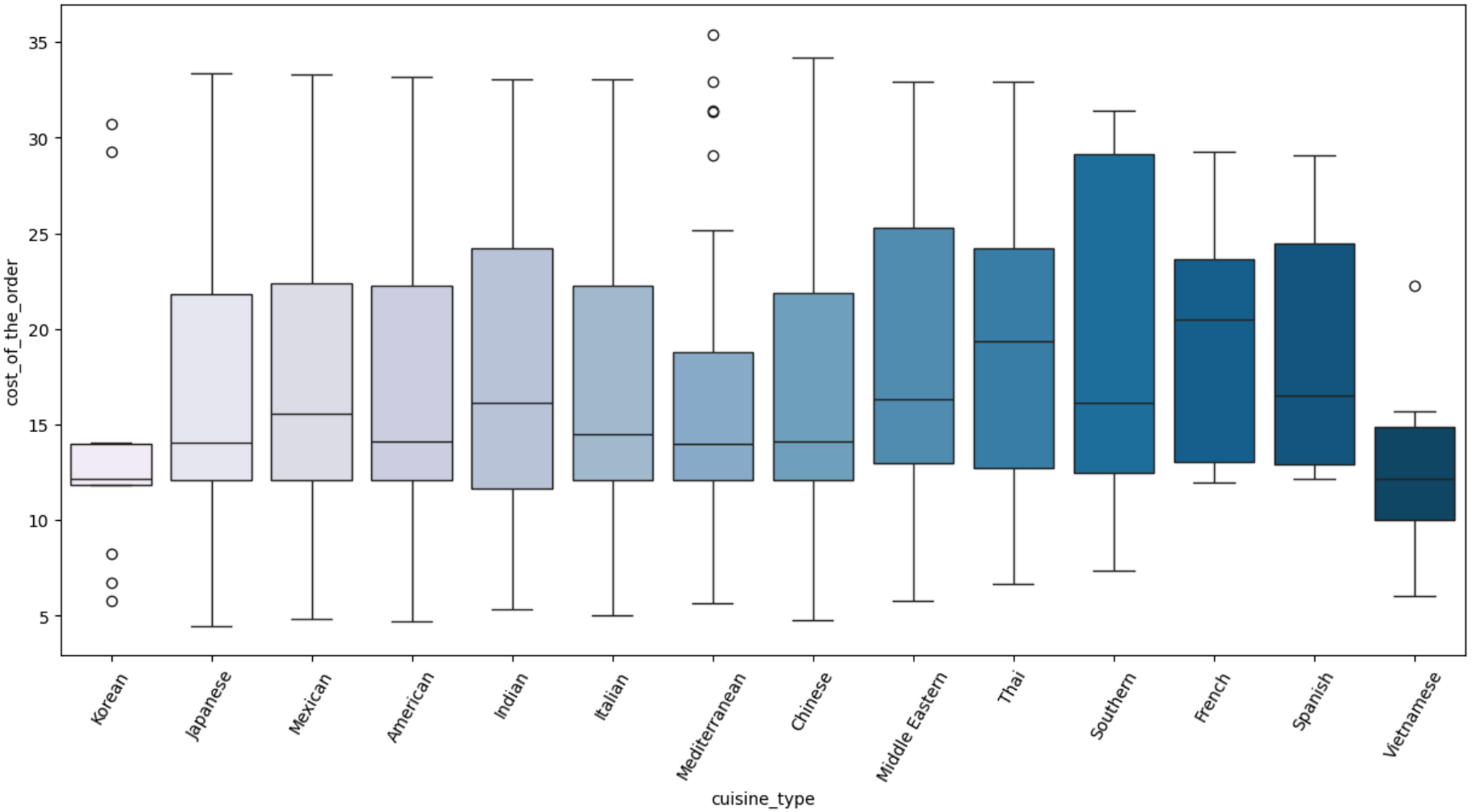
Customer 83287: $139.31/9$ =15.48 per order

Multivariate Analysis

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

In [814…

```
# Relationship between cost of the order and cuisine type
plt.figure(figsize=(15,7))
sns.boxplot(x = "cuisine_type", y = "cost_of_the_order", data = df, palette = 'PuBu', hue = "cuisine_type")
plt.xticks(rotation = 60);
```



Observations:

Median Order Costs Vary Across Cuisines:

When compared to other cuisines, the median order costs for French, Spanish, and Italian cuisines are the highest.

In general Vietnamese and Korean food are more affordable because they have the lowest median

prices.

High Variability in Prices for Certain Cuisines:

The wide interquartile range (IQR) of Southern, Spanish, Middle Eastern, Indian, and Thai cuisines indicates that their prices differ greatly from one another.

Significant variation is also seen in Mexican, Indian, and Japanese cuisines, indicating that these cuisines offer both affordable and high-end options.

Smaller IQRs indicate more consistent order costs for Korean, Vietnamese, and Mediterranean customers. However, when we examine these eateries more closely, some intriguing findings emerge.

Outliers in Expensive Orders:

There are high-priced outliers in several cuisines (above $30–$35), particularly in Korean, Mediterranean, and Vietnamese.

These outliers likely represent specialty or large meal orders. It is interesting to note that Koreans have extremely expensive outliers and low-value outliers.

Lower-Cost Cuisines:

Vietnamese and Korean food have lower prices. Were median costs and limited outliers, suggesting they offer more standardized pricing.

American cuisine has a comparatively lower median order cost, most likely as a result of the fast food options being available.

Business insights:

Fine Dining: Meals from French, Southern, Thai, and Spanish cuisines are typically more expensive, signifying a fine dining experience.

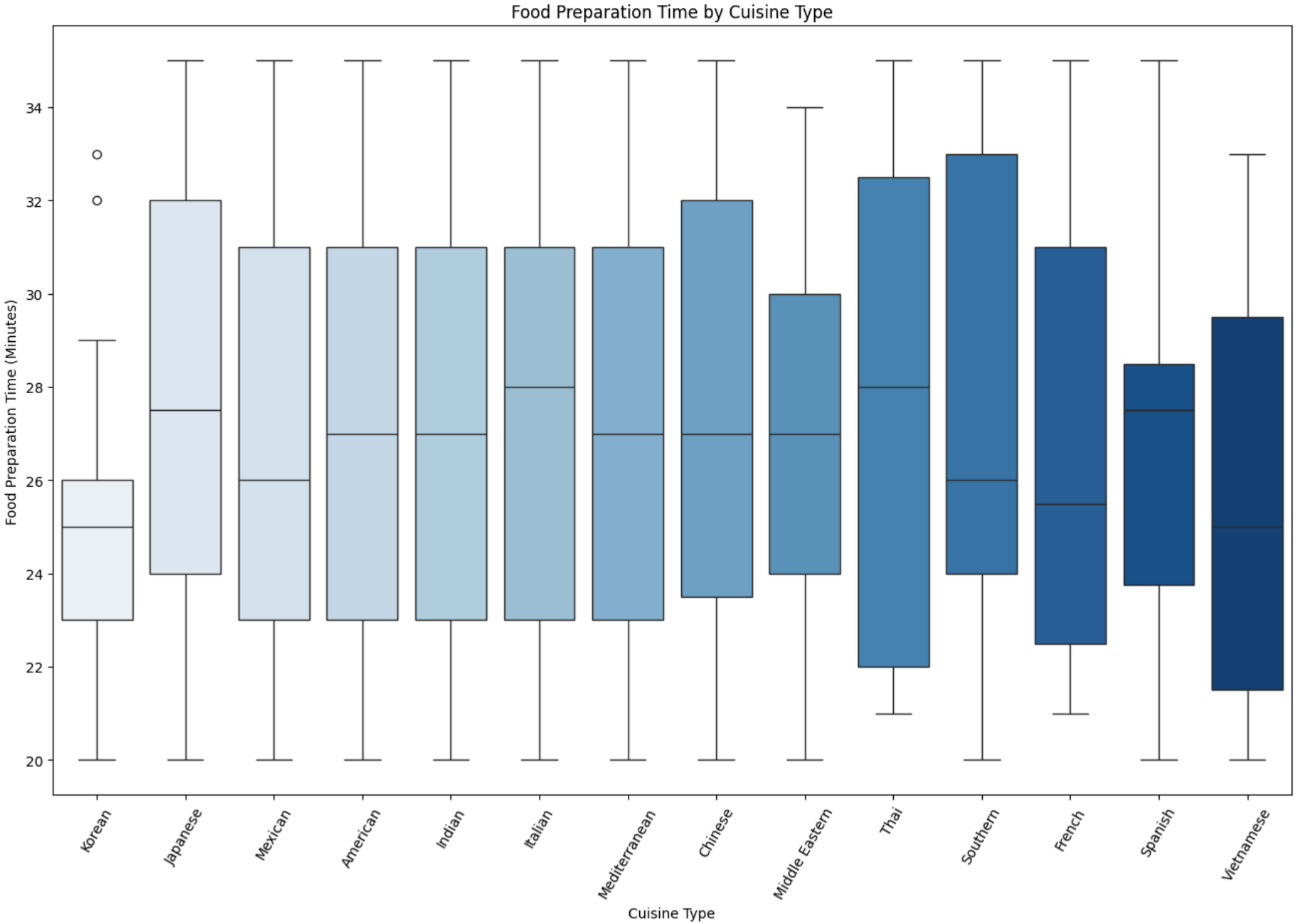
Cost-effective Options: Due to their lower and more stable prices, Korean, Vietnamese, and Mediterranean cuisines are reasonably priced for patrons. However, it is noted that the Mediterranean has some cost anomalies that require more investigation.

Opportunities for Price Variation: Menu optimization may help cuisines with high levels of variability (such as Mexican, Indian, and Thai) by providing more mid-range pricing options to appeal to a wider range of customers.

Outlier Analysis: If there are high-cost outliers, more research can help identify whether they are the result of premium items, large orders, or inconsistent pricing.

In [816…

```
plt.figure(figsize=(16,10))
sns.boxplot(x="cuisine_type", y="food_preparation_time", hue="cuisine_type", data=df, palette="Blues")
plt.xticks(rotation=60)
plt.xlabel("Cuisine Type")
plt.ylabel("Food Preparation Time (Minutes)")
plt.title("Food Preparation Time by Cuisine Type")
plt.legend([],[], frameon=False)  # Hides duplicate legend
plt.show()
```



Overall Range of Food Preparation Time:

Most cuisines have food preparation times ranging between 20 and 35 minutes.

Cuisines with the Shortest Median Preparation Times:

Vietnamese & Korean cuisine has the shortest median preparation time, around 21-25 minutes.

Other cuisines with relatively lower median preparation times include Mexican & Italian.

Cuisines with the Longest Median Preparation Times:

French and Southern cuisines have the highest median preparation times, nearing 30 minutes.

Japanese, Middle Eastern, and Thai cuisines also have longer preparation times compared to other cuisines.

Cuisines with the Widest Spread (High Variability in Preparation Times):

French, Southern, Thai, and Japanese cuisines show a large spread in preparation time, meaning restaurants preparing these cuisines take anywhere from 20 to 35 minutes.

This suggests that some dishes within these cuisines may be more time-intensive to prepare.

Outliers and Exceptional Cases:

Outliers are present in some cuisines, like Korean, meaning that sometimes food can take exceptionally longer to prepare compared to the majority.

Balanced Cuisines (Consistent Preparation Times):

Korean and Chinese cuisines show relatively small interquartile ranges (IQR), meaning that the preparation times for most restaurants are more consistent.

Key Takeaways:

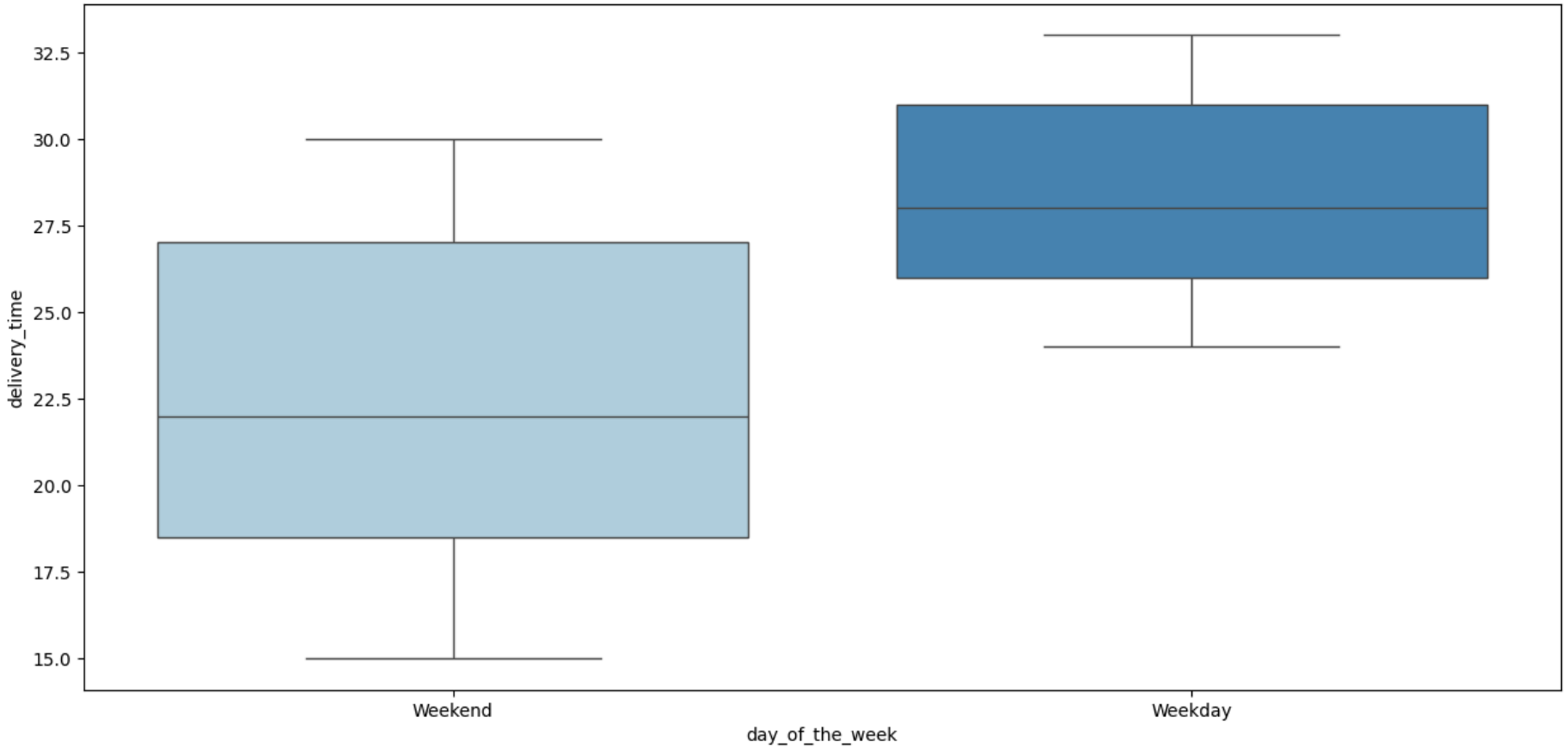
Customers ordering Korean or Vietnamese cuisine are likely to get their food faster.

Those ordering French, Middle Eastern, or Southern cuisine might experience longer waiting times.

High variability in some cuisines could mean differences in dish complexity or restaurant efficiency.

In [818…

```
# Relationship between day of the week and delivery time

plt.figure(figsize=(15,7))
sns.boxplot(x="day_of_the_week", y="delivery_time", hue="day_of_the_week", data=df, palette="Blues", legend=False)
plt.show()
```



Observations from the Box Plot:

Day of the Week vs. Delivery Time:

Weekday Deliveries Take Longer:

The median delivery time on weekdays is higher than on weekends.

The interquartile range (IQR) for weekdays is also narrower, indicating more consistent delivery times.

Weekend Deliveries Are Faster but More Variable:

The median delivery time is lower on weekends.

The IQR is wider, suggesting greater variability in delivery times, possibly due to unpredictable demand surges.

Possible Operational Insights:

Weekday deliveries may take longer due to higher traffic congestion or operational constraints; schools, businesses, and other factors may contribute to the additional time required for weekday deliveries.

Weekend deliveries may have more variability, possibly influenced by fluctuating demand patterns or staffing differences.

In [820…

```
# Calculate total revenue generated by each restaurant
restaurant_revenue = df.groupby(['restaurant_name'])['cost_of_the_order'].sum()

# Sort restaurants by total revenue in descending order and display the top 14
top_14_revenue_restaurants = restaurant_revenue.sort_values(ascending=False).head(14)

# Display results
print("Top 14 Revenue-Generating Restaurants:")
print(top_14_revenue_restaurants)
```

```
Top 14 Revenue-Generating Restaurants:
restaurant_name
Shake Shack                      3579.53
The Meatball Shop                2145.21
Blue Ribbon Sushi                1903.95
Blue Ribbon Fried Chicken        1662.29
Parm                             1112.76
RedFarm Broadway                  965.13
RedFarm Hudson                    921.21
TAO                               834.50
Han Dynasty                       755.29
Blue Ribbon Sushi Bar & Grill     666.62
Rubirosa                          660.45
Sushi of Gari 46                  640.87
Nobu Next Door                    623.67
Five Guys Burgers and Fries       506.47
Name: cost_of_the_order, dtype: float64
```

Observations from the Top 14 Revenue-Generating Restaurants:

Shake Shack significantly outperforms all other restaurants, generating $3579.53. Nearly 67% more than the second-highest revenue generator. This suggests high order volume, premium pricing, or both.

The Meatball Shop is the second highest earner ($2145.21). Again, an American-style cuisine, which dominates the market in terms of volume ordered.

Blue Ribbon Chain Success:

Blue Ribbon Sushi, Blue Ribbon Fried Chicken, and Blue Ribbon Sushi Bar & Grill appear multiple times in the top 10. This suggests brand loyalty, strong market presence, or menu pricing contributing to higher revenue.

High Revenue ≠ Most Frequent Orders:

Some of these high-revenue restaurants may not be the most frequently ordered from.

I would like to compare revenue vs. order frequency to see if these restaurants succeed due to higher-priced menu items rather than more orders.

Cuisine Insights:

Sushi and Asian cuisine dominate with multiple entries (e.g., Sushi of Gari 46, TAO, Blue Ribbon Sushi).

Burgers (Five Guys, Shake Shack) and Italian (Parm, Rubirosa, RedFarm Hudson) also perform well.

Indicates strong customer preference for premium casual dining options.

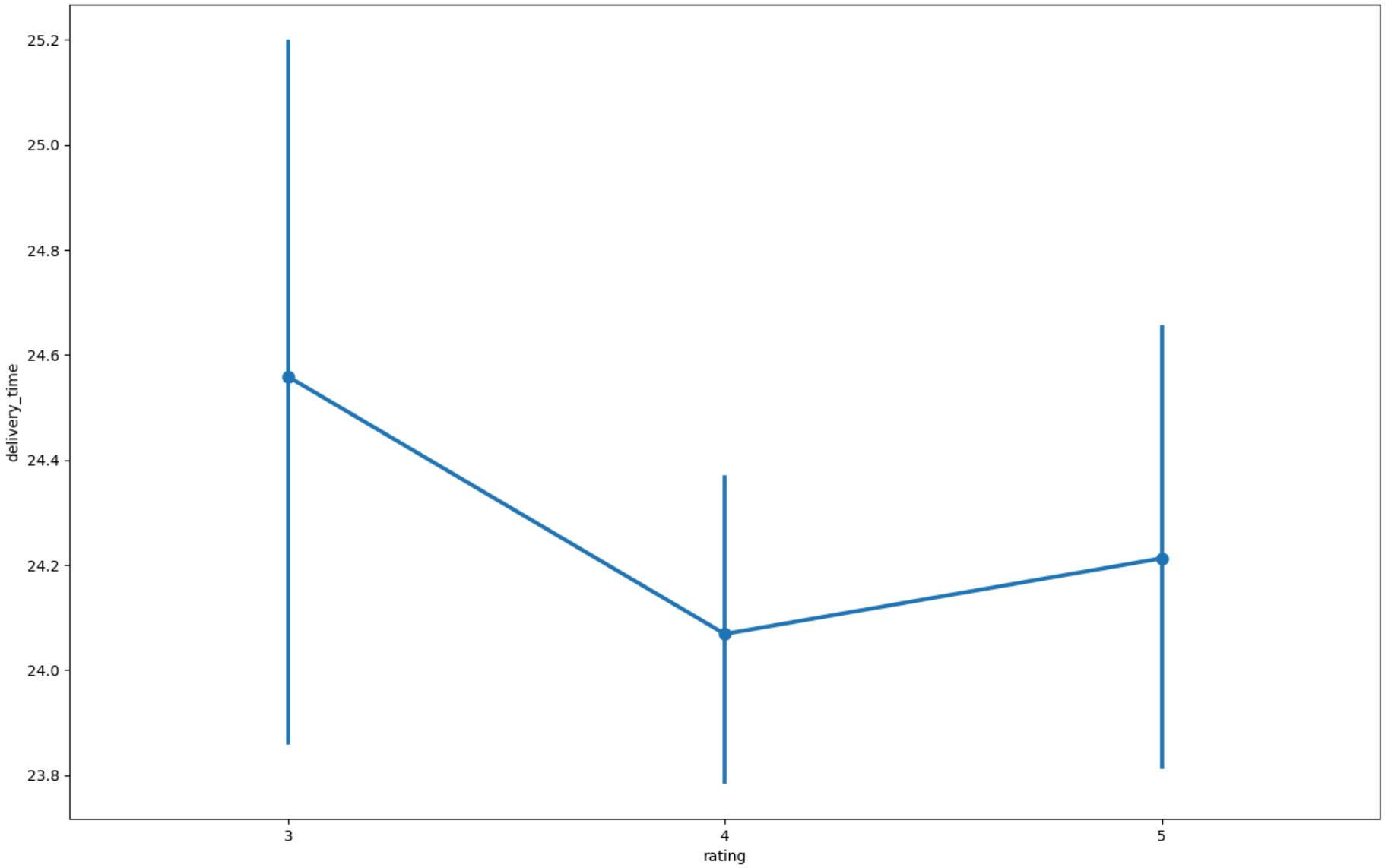
Luxury vs. Fast Casual:

Nobu Next Door and TAO suggest premium/high-end restaurant demand, while Shake Shack and Five Guys represent fast-casual dominance.

Suggests a split market where both fast-casual and upscale dining thrive.

In [822…

```
# Relationship between rating and delivery time
plt.figure(figsize=(16, 10))
sns.pointplot(x = 'rating', y = 'delivery_time', data = df)
plt.show()
```



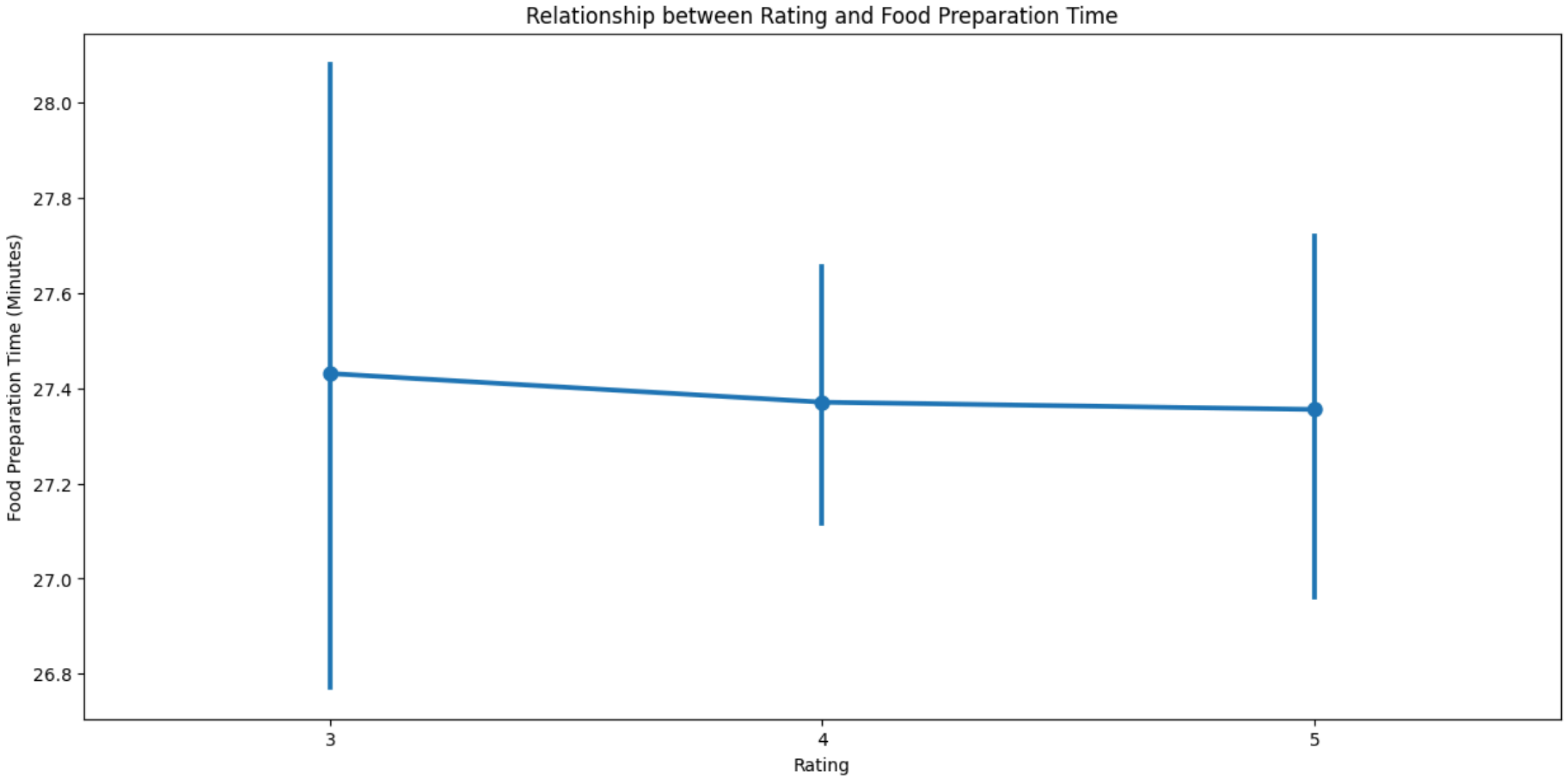
Observation on Point Plot:

Delivery time alone is not the main driver of ratings. The average delivery time for the orders is extremely close.

Ratings are slightly higher for deliveries that are not the absolute fastest, possibly due to factors like food quality and service.

Slow delivery could be a factor in lower ratings, but it's not the main one.

In [824…
```
# Relationship between rating and food preparation time
plt.figure(figsize=(15, 7))
sns.pointplot(x='rating', y='food_preparation_time', data=df)
plt.xlabel("Rating")
plt.ylabel("Food Preparation Time (Minutes)")
plt.title("Relationship between Rating and Food Preparation Time")
plt.show()
```



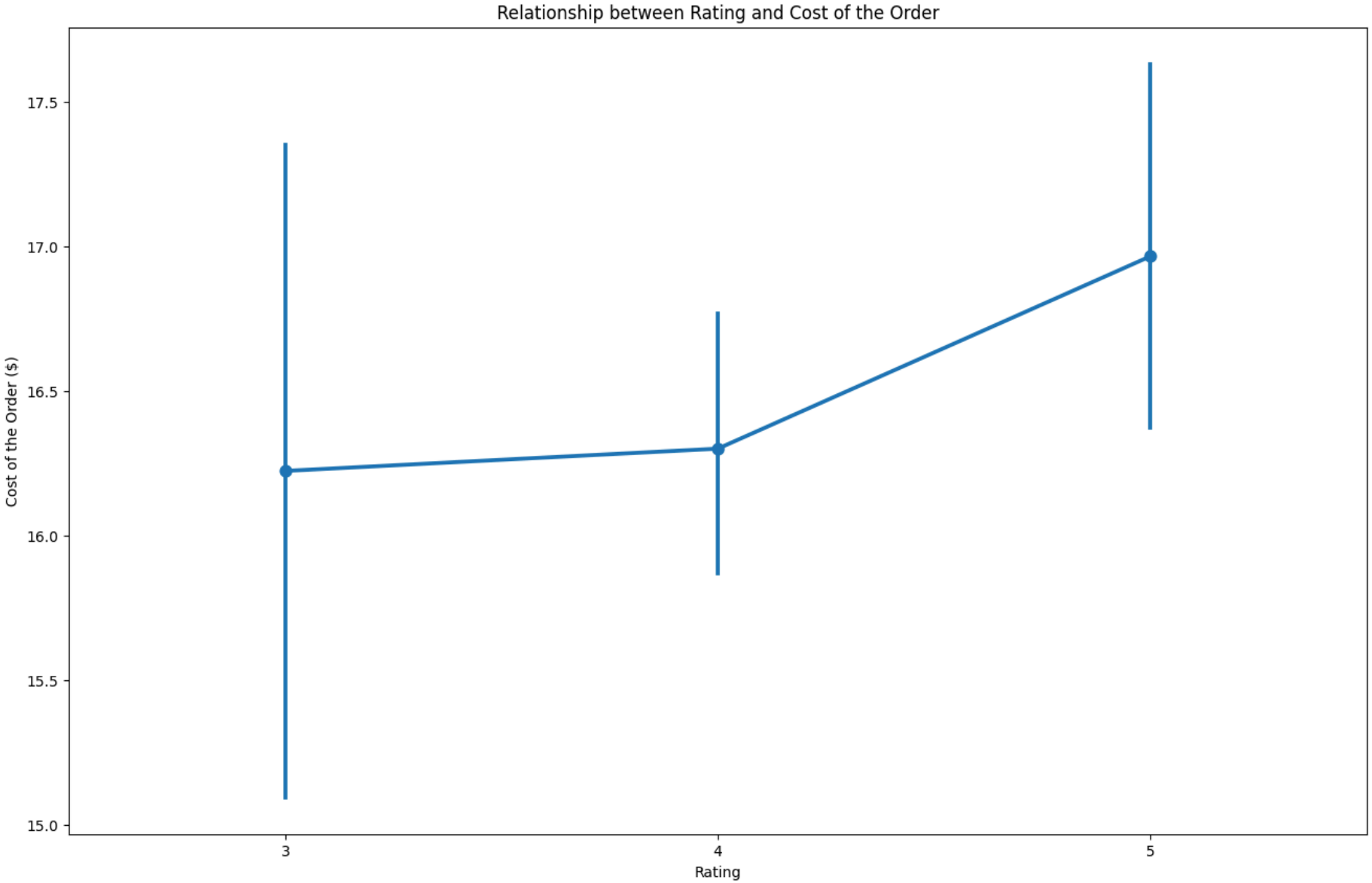
Observations on Ratings & Food Preparation Time:

This backs up the observation from our previous cells where we stated that ratings are not majorly influenced by the time taken.

That being said, 4 is the most popular rating, and it is also the shortest food preparation time on average, followed closely by 5.

In [826…

```
# Relationship between rating and cost of the order
plt.figure(figsize=(16, 10))
sns.pointplot(x='rating', y='cost_of_the_order', data=df)
plt.xlabel("Rating")
plt.ylabel("Cost of the Order ($)")
plt.title("Relationship between Rating and Cost of the Order")
plt.show()
```



Observations on Rating and Cost of the Order):

Higher-rated orders tend to be slightly more expensive:

There is a slight upward trend in the cost of the order as ratings increase.

Orders rated 5 have the highest average cost, suggesting customers may associate higher-cost meals with better quality or experience. This could be bias, of course.

Orders rated 3 have lower variability:

The error bars for rating 3 are longer, meaning there is more variation in cost.

Some customers may have expected more from expensive orders and rated them lower.

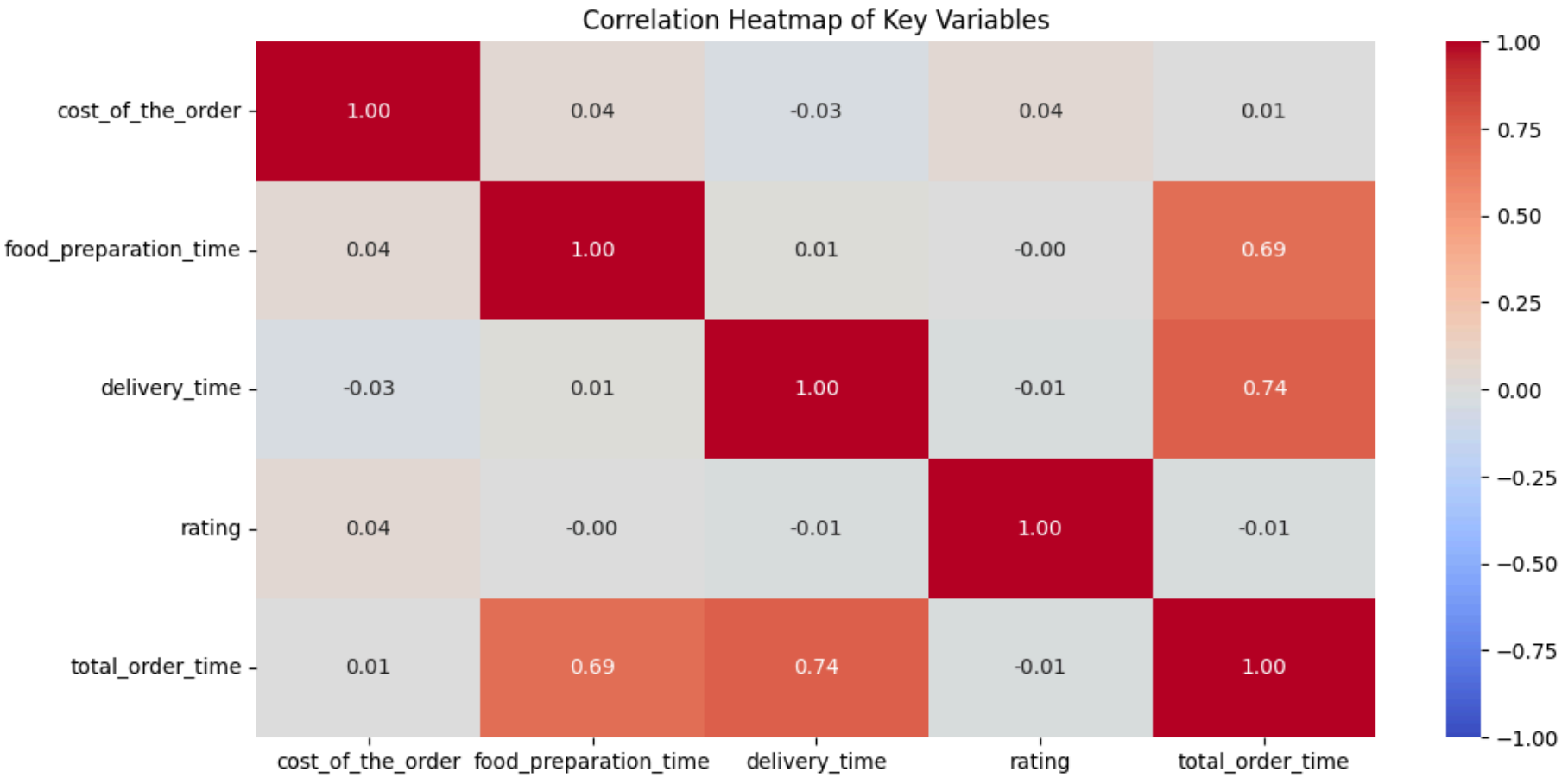
A potential link between price perception and satisfaction:

If higher-priced orders get better ratings, it could mean people tend to associate price with quality.

Conversely, lower ratings on expensive orders might indicate unmet expectations.

In [828…

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

# Select relevant numerical columns for correlation analysis
col_list = ['cost_of_the_order', 'food_preparation_time', 'delivery_time',
            'rating', 'total_order_time']  # Add more relevant numerical columns

# Compute correlation matrix
corr_matrix = df[col_list].corr()

# Plot the heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(corr_matrix, annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="coolwarm")

# Title
plt.title("Correlation Heatmap of Key Variables")

plt.show()
```



Observations of Heatmap:

Strong Positive Correlation Between Total Order Time and Delivery Time (0.74)

As expected, total order time is heavily influenced by delivery time. This suggests that longer delivery times significantly contribute to the overall order fulfillment time.

Significant Positive Correlation Between Total Order Time and Food Preparation Time (0.69)

This indicates that restaurants with longer food preparation times also have longer total order times. This could mean certain cuisines or restaurants take longer to prepare meals, affecting customer wait time.

Weak or No Correlation Between Cost of the Order and Other Variables

The cost of the order shows very weak correlations with delivery time (-0.03) and food preparation time (0.04). This suggests that higher-priced orders do not necessarily take longer to prepare or deliver.

No Significant Correlation Between Rating and Any Key Variables

The rating variable has values close to zero when compared to all other variables. This suggests that food quality perception (ratings) is not directly influenced by order cost, delivery time, or preparation time.

Key Takeaways

Operational Efficiency: Restaurants that can optimize food preparation could reduce total order time significantly.

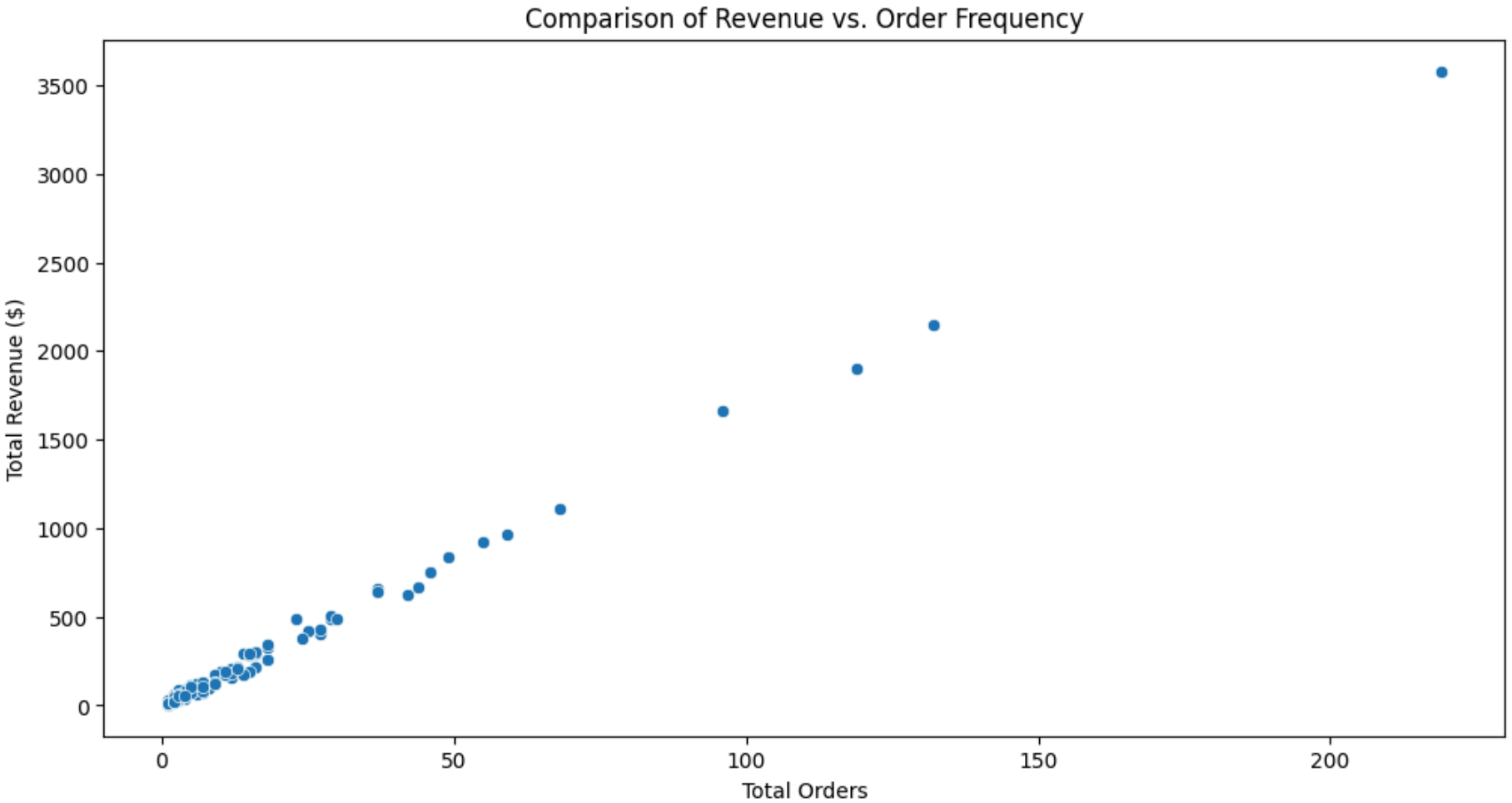
No Pricing Impact on Speed: Expensive orders don't seem to take longer to prepare or deliver.

Ratings Are Independent: Customer ratings do not appear to be influenced by cost, delivery speed, or prep time, suggesting that other factors (like taste, packaging, or service) may play a bigger role in satisfaction.

In [830…

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

# Compute total revenue per restaurant
restaurant_revenue = df.groupby("restaurant_name")["cost_of_the_order"].sum()

# Compute order frequency per restaurant
restaurant_orders = df["restaurant_name"].value_counts()

# Create a DataFrame combining revenue and order frequency
restaurant_summary = pd.DataFrame({"Total Orders": restaurant_orders, "Total Revenue": restaurant_revenue})

# Plot a scatter plot
plt.figure(figsize=(12, 6))
sns.scatterplot(x=restaurant_summary["Total Orders"], y=restaurant_summary["Total Revenue"])

# Label the axes and title
plt.xlabel("Total Orders")
plt.ylabel("Total Revenue ($)")
```

```
plt.title("Comparison of Revenue vs. Order Frequency")

plt.show()
```



Observations from the Scatter Plot (Revenue vs. Order Frequency):

Positive Correlation Between Total Orders and Revenue:

The scatter plot shows an upward trend, indicating that restaurants with more orders generally generate more revenue.

This confirms that order frequency is a strong driver of total revenue.

Outlier Restaurant (Shake Shack):

One restaurant significantly stands out with the highest order count and revenue.

This suggests that this restaurant is succeeding due to both high volume and high revenue, making it a top performer.

Cluster of Low-Order, Low-Revenue Restaurants:

Many restaurants are clustered in the bottom left, meaning they have both low order volume and low revenue.

These restaurants may not be as competitive or have lower demand.

Question 13: The company wants to provide a promotional offer in the advertisement of the restaurants. The condition to get the offer is that the restaurants must have a rating count of more than 50 and the average rating should be greater than 4. Find the restaurants fulfilling the criteria to get the promotional offer. [3 marks]

In [833…

```
# create a new variable to filter the rated restaurants
df_rated = df[df['rating'] != 'Not given'].copy()

# Convert rating column from object to integer
df_rated['rating'] = df_rated['rating'].astype('int')

# Create a new dataframe that contains the restaurant names with their rating counts
df_rating_count = df_rated.groupby(['restaurant_name'])['rating'].count().sort_values(ascending=False).reset_index(
df_rating_count.head()

# Get the restaurant names that have a rating count greater than 50
rest_names = df_rating_count[df_rating_count['rating'] > 50]['restaurant_name']

# Filter to get the data of restaurants that have a rating count of more than 50
df_mean_4 = df_rated[df_rated['restaurant_name'].isin(rest_names)].copy()

# Group the restaurant names with their ratings and find the mean rating of each restaurant
df_mean_4_rating = df_mean_4.groupby(['restaurant_name'])['rating'].mean().sort_values(ascending=False).reset_index
```

```
# Filter for restaurants where the average rating is greater than 4
df_avg_rating_greater_than_4 = df_mean_4_rating[df_mean_4_rating['rating'] > 4].sort_values(by='rating', ascending=

# Display the qualifying restaurants
df_avg_rating_greater_than_4
```

Out [833…

	restaurant_name	rating
0	The Meatball Shop	4.325758
1	Blue Ribbon Fried Chicken	4.218750
2	RedFarm Broadway	4.169492
3	Shake Shack	4.168950
4	Blue Ribbon Sushi	4.134454
5	RedFarm Hudson	4.109091
6	Parm	4.073529

Observations:

Top-Rated Restaurants with High Review Counts

The Meatball Shop has the highest average rating (4.33) among restaurants with more than 50 ratings, indicating strong customer satisfaction.

Other highly rated restaurants include Blue Ribbon Fried Chicken (4.22) and RedFarm Broadway (4.17), which also maintain strong reputations.

Shake Shack's Performance

Shake Shack, which had the highest total revenue and order frequency in previous analyses, maintains a solid average rating of 4.17. This suggests that despite high order volume, customer satisfaction remains strong. A brilliant all-rounder in this market, which reflects on our demographic ("students and busy professionals who rely on restaurants due to their hectic lifestyles")

Blue Ribbon Restaurants are Highly Rated

Blue Ribbon Fried Chicken (4.22) and Blue Ribbon Sushi (4.13) both appear on the list, showing that this restaurant brand is consistently delivering well-rated dining experiences across different cuisines.

RedFarm Locations Both Qualify

RedFarm Broadway (4.17) and RedFarm Hudson (4.11) both meet the criteria, meaning this brand maintains strong quality across multiple locations.

Consistency Across High Revenue Restaurants

Many of the restaurants appearing in this list (Shake Shack, The Meatball Shop, Blue Ribbon Sushi, Parm, etc.) were also identified as top revenue generators. This suggests that high revenue is linked to both high volume and strong customer satisfaction.

Parm Just Makes the Cut

Parm (4.07) qualifies but has the lowest rating on this list. This suggests that while it is positively rated, it may not be as highly regarded as the other restaurants.

Key Takeaways for Promotions:

Ideal Candidates for the Offer: The Meatball Shop, Blue Ribbon Fried Chicken, and RedFarm Broadway, as they are highly rated and well-known.

Shake Shack is a Powerhouse: It dominates in orders, revenue, and ratings, making it an essential restaurant for promotions.

Potential for Parm to Improve: Since it barely passes the rating threshold, further analysis into customer reviews could help pinpoint areas for improvement.

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 [836…

```
# Define a function to manage the conditions of this requirement
def compute_rev(x):
    if x > 20:
```

```
        return x * 0.25  # 25% charge
    elif x > 5:
        return x * 0.15  # 15% charge
    else:
        return 0  # No charge
```

In [837…
```
# Call the function
df['Revenue'] = df['cost_of_the_order'].apply(compute_rev)
```

In [838…
```
# Get the total revenue and print it
total_rev = df['Revenue'].sum()
print('The net revenue generated by the company is around', round(total_rev, 2), 'dollars')
```

```
The net revenue generated by the company is around 6166.3 dollars
```

Observations:

Total Revenue Generated:

The company has earned $6,166.30 in commission revenue from the orders.

This revenue is solely based on the percentage cut (25% or 15%) from the orders based on their cost brackets.

Revenue Dependency on Order Cost:

Orders above $20 contribute the most to the company's earnings (25% charge).

Orders between $5 and 20 contribute less (15% charge).

Orders below $5 do not generate any revenue, but their volume could still be important for customer retention.

Business Implications:

If a large portion of revenue comes from higher-cost orders ($20+), the company should incentivize restaurants to maintain premium menu offerings.

If the majority of revenue is coming from mid-range orders ($5 — 20), it might be beneficial to introduce promotions to encourage larger orders.

Future Strategy:

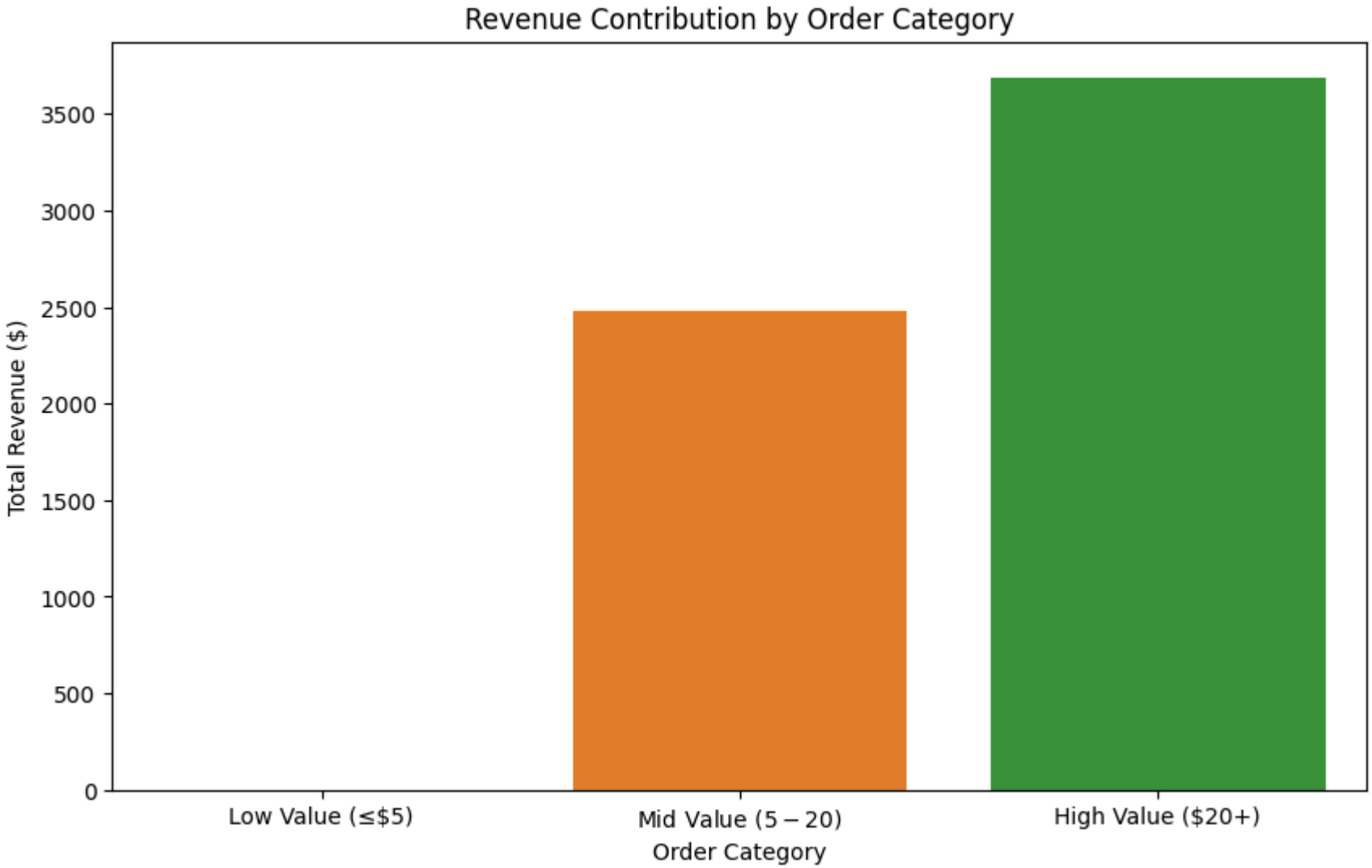
The company could increase its profit margin by adjusting commission percentages.

Offering discounts or free delivery on high-cost orders may increase volume, leading to more revenue.

In [840…

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

# Compute revenue based on conditions
def compute_rev(x):
    if x > 20:
        return x * 0.25  # 25% charge
    elif x > 5:
        return x * 0.15  # 15% charge
    else:
        return 0         # No charge

# Apply the function to create a new 'Revenue' column
df['Revenue'] = df['cost_of_the_order'].apply(compute_rev)

# Categorize orders based on cost
df['Order_Category'] = pd.cut(df['cost_of_the_order'], bins=[0, 5, 20, float('inf')],
                              labels=['Low Value (≤$5)', 'Mid Value ($5–$20)', 'High Value ($20+)'])

# Group by Order Category and calculate revenue
revenue_by_category = df.groupby('Order_Category')['Revenue'].sum().reset_index()

# Display the revenue breakdown as a DataFrame
print(revenue_by_category)

# Visualize Revenue Contribution Using a Bar Chart
plt.figure(figsize=(10, 6))
sns.barplot(x='Order_Category', y='Revenue', hue='Order_Category', data=revenue_by_category, legend=False), plt.xla
plt.ylabel("Total Revenue ($)")
plt.title("Revenue Contribution by Order Category")
plt.show()
```

	Order_Category	Revenue
0	Low Value (≤$5)	0.0000
1	Mid Value ($5–$20)	2477.5755
2	High Value ($20+)	3688.7275



Business Insights:

Focusing on Higher-Priced Orders

The company benefits most from orders above $20, so promotions targeting higher-spending customers may be profitable.

Increasing Mid-Value Order Frequency

Since mid-value orders already generate significant revenue, strategies like discounts, bundle offers, or free delivery thresholds could encourage more purchases.

Reevaluating Low-Value Orders

If low-value orders are frequent but unprofitable, the company might consider minimum order fees or encourage upselling.

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 [843…
```
# Calculate percentage of orders with total_order_time > 60 minutes
percentage_over_60 = (df[df['total_order_time'] > 60].shape[0] / df.shape[0]) * 100

# Print the result
print(f"Percentage of orders with total order time over 60 minutes: {percentage_over_60:.2f}%")
```

```
Percentage of orders with total order time over 60 minutes: 10.54%
```

Observations:

Observations on Total Order Time Over 60 Minutes

Limited Impact on Overall Orders

Only 10.54% of total orders took more than 60 minutes from order placement to delivery.

This suggests that most orders are delivered within an hour, which is a good benchmark for service efficiency.

In [845…
```
# Count of orders > 60 minutes per cuisine type
long_orders_by_cuisine = df[df['total_order_time'] > 60]['cuisine_type'].value_counts()
```

```
# Display the top cuisine types with the most delayed orders
print(long_orders_by_cuisine.head(10))
```

```
Japanese          55
American          55
Italian           34
Chinese           31
Indian             6
Mediterranean      5
Mexican            5
Middle Eastern     2
French             2
Southern           2
Name: cuisine_type, dtype: int64
```

Observations:

Both Japanese and American cuisines have 55 delayed orders each, which is significantly higher than other cuisines.

These cuisines might involve complex preparation processes, high order volume, or delivery challenges.

In [847…

```
# Compare average ratings for orders > 60 minutes vs. <= 60 minutes
avg_rating_over_60 = df[df['total_order_time'] > 60]['rating'].mean()
avg_rating_under_60 = df[df['total_order_time'] <= 60]['rating'].mean()

print(f"Average rating for orders over 60 minutes: {avg_rating_over_60:.2f}")
print(f"Average rating for orders under 60 minutes: {avg_rating_under_60:.2f}")
```

```
Average rating for orders over 60 minutes: 4.21
Average rating for orders under 60 minutes: 4.21
```

No Significant Impact of Delayed Orders on Ratings

Both orders taking more than 60 minutes and orders delivered within 60 minutes have an average rating of 4.21.

This suggests that delivery time alone does not strongly influence customer satisfaction.

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 [850…
```
print(df['day_of_the_week'].unique())
```

```
['Weekend' 'Weekday']
```

In [851…
```
mean_delivery_weekday = df[df['day_of_the_week'] == 'Weekday']['delivery_time'].mean()
mean_delivery_weekend = df[df['day_of_the_week'] == 'Weekend']['delivery_time'].mean()

print(f"Mean delivery time on Weekdays: {mean_delivery_weekday:.2f} minutes")
print(f"Mean delivery time on Weekends: {mean_delivery_weekend:.2f} minutes")
```

```
Mean delivery time on Weekdays: 28.34 minutes
Mean delivery time on Weekends: 22.47 minutes
```

Observations:

Deliveries Are Faster on Weekends (22.47 min) Compared to Weekdays (28.34 min)

Orders placed on weekends are delivered about 6 minutes faster on average than weekday orders.

This suggests that delivery operations might be more efficient on weekends, possibly due to lower traffic congestion or optimized staffing.

Conclusion and Recommendations

Question 17: What are your conclusions from the analysis? What recommendations would you like to share to help improve the business? (You can use cuisine type and feedback ratings to drive your business recommendations.) [6 marks]

Conclusions:

Revenue is Driven by High-Value Orders ($20+)

Orders above 20 USD contribute the most to revenue at 3,688.73 when compared to mid-value orders 5-20 at 2,477.58.

Low-value orders (≤$5) generate no revenue due to the commission structure.

Recommendation: Encourage customers to spend more than $20 by offering bundle deals, free delivery for large orders, or discount vouchers for higher spending.

Shake Shack is the Top Revenue-Generating Restaurant

Shake Shack dominates revenue and order volume, making it a strategic partner for the business.

Other high-revenue restaurants include The Meatball Shop, Blue Ribbon, and RedFarm Broadway.

Recommendation: Promote high-performing restaurants in marketing campaigns and offer featured restaurant placements on the platform.

Weekday Deliveries Take Longer than Weekend Deliveries

Weekday orders take an average of 28.34 minutes, while weekend orders are faster at 22.47 minutes.

Recommendation: To improve weekday delivery speed, consider dynamic pricing for delivery fees, offering customers an option to pay more for faster delivery during peak weekday hours.

Certain Cuisines Experience Longer Order Times

Japanese (55 orders), American (55), and Italian (34) cuisine types have the highest number of orders taking more than 60 minutes.

This suggests that certain cuisines may require more preparation time.

Recommendations:

Partner with restaurants to implement pre-ordering or order queuing to manage customer expectations.

Offer estimated delivery time transparency at checkout based on cuisine type.

Customer Ratings Are Not Impacted by Delivery Time

Orders over 60 minutes have the same average rating (4.21) as orders under 60 minutes.

The evidence suggests that delivery time alone does not significantly impact customer satisfaction.

Recommendation: Improve ratings through better packaging, food presentation, and quality control measures rather than just reducing delivery time.

Top-Rated Restaurants for Promotions

Restaurants that qualify for promotions (Rating > 4, 50+ ratings) include:

The Meatball Shop (4.33)

Blue Ribbon Fried Chicken (4.22)

Shake Shack (4.17)

These restaurants have strong brand loyalty and customer satisfaction, making them ideal for exclusive promotional deals.

Recommendation: Feature these top-rated restaurants in special promotional campaigns, such as “Highly Rated Favorites” or customer loyalty rewards.

Recommendations:

Final Business Strategy Recommendations:

Introduce a Loyalty Program: Reward high-spending and repeat customers with exclusive discounts or free delivery on future orders.

Optimize Restaurant Partnerships: Work closely with high-revenue, high-rated restaurants to increase order fulfillment speed and maintain quality standards.

Leverage Peak Hour Pricing: Implement dynamic delivery pricing for peak weekday hours, allowing customers to pay extra for faster deliveries.

Improve Transparency in Estimated Delivery Time: Provide real-time tracking and estimated delivery time predictions based on cuisine type and restaurant efficiency.
