Foodhub EDA and Revenue Optimization

Context

The number of restaurants in New York is increasing rapidly, and many students and working professionals depend on these restaurants because of their busy schedules. Online food delivery services have become a convenient solution, allowing customers to order food from their favorite restaurants with ease.

FoodHub is a food aggregator platform that provides access to multiple restaurants through a single mobile application. Customers can place orders directly through the app, and once the restaurant confirms the order, the platform assigns a delivery partner. The delivery person picks up the order from the restaurant and delivers it to the customer's location. After receiving the order, the customer can rate the experience in the app. FoodHub generates revenue by charging a fixed commission on each order from the partner restaurants.

Objective

FoodHub has collected data from customer orders placed through its platform. The company wants to analyze this data to understand customer demand, restaurant performance, and operational efficiency. These insights will help improve customer experience, optimize operations, and support better business decisions.

As a Data Scientist at FoodHub, the goal of this analysis is to explore the order data and answer key business questions provided by the team.

Data Description

The dataset contains information related to individual food orders placed through the FoodHub platform. Each record represents a single order and includes details about the customer, restaurant, cuisine, cost, timing, and ratings.

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

Data Source: Kaggle – FoodHub Dataset (https://www.kaggle.com/datasets/tasnimniger/foodhub-data/data)

Let us start by importing the required libraries and Loading the datasets

```
# 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

# Command to tell Python to actually display the graphs
%matplotlib inline

# to restrict the float value to 3 decimal places
pd.set_option('display.float_format', lambda x: '%.2f' % x)

import warnings
warnings.filterwarnings('ignore')

# Loading the datasets
from google.colab import drive
drive.mount('/content/drive')

file_path="/content/drive/My Drive/Datasets/foodhub_order.csv"
df=pd.read_csv(file_path)
```

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

```
# Checking the first 5 rows
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1898,\n  \"fields\":
[\n    {\n      \"column\": \"order_id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 548,\n        \"min\":
1476547,\n        \"max\": 1478444,\n        \"num_unique_values\":
1898,\n        \"samples\": [\n          1477722,\n          1478319,\
n          1477650\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"customer_id\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 113698,\n        \"min\": 1311,\n
\"max\": 405334,\n        \"num_unique_values\": 1200,\n
\"samples\": [\n          351329,\n          49987,\n          345899\
n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"restaurant_name\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 178,\n
\"samples\": [\n          \"Tortaria\",\n          \"Osteria
Morini\",\n          \"Philippe Chow\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cuisine_type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 14,\n        \"samples\": [\n
\"Thai\",\n          \"French\",\n          \"Korean\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cost_of_the_order\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7.483812110049553,\n        \"min\": 4.47,\n        \"max\": 35.41,\n
\"num_unique_values\": 312,\n        \"samples\": [\n          21.29,\
n          7.18,\n          13.34\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"day_of_the_week\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"Weekday\",\n          \"Weekend\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"5\",\n          \"4\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"food_preparation_time\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4,\n        \"min\": 20,\n        \"max\": 35,\n
\"num_unique_values\": 16,\n        \"samples\": [\n          25,\n
23\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"delivery_time\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 4,\n        \"min\": 15,\n
\"max\": 33,\n        \"num_unique_values\": 19,\n        \"samples\":
[\n          20,\n          21\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


Understanding the structure of the data

```
# Checking the shape of the dataset
print("There are", df.shape[0], 'rows and', df.shape[1], "columns.")
```

```
There are 1898 rows and 9 columns.
```

```
# Checking the structure and types of the data
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
```

Observation:

- There are 6 numerical columns in the data and 8 object type columns.
- It can be observed that no columns have less entries (less than 1898 rows) which indicates that there is no missing values in the data.

```
# Checking for missing values in the data
df.isnull().sum()
```

```
order_id                0
customer_id             0
restaurant_name         0
cuisine_type            0
cost_of_the_order       0
day_of_the_week         0
rating                  0
food_preparation_time   0
delivery_time           0
dtype: int64
```

*It is confirmed that there are **no missing value** in the dataset.*

Checking the Statistical Summary of the data

```
# Statistical summary of the Numerical Data
df.describe().T
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5,\n  \"fields\": [\n{\n      \"column\": \"count\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 0.0,\n        \"min\": 1898.0,\n        \"max\": 1898.0,\n        \"num_unique_values\": 1,\n\"samples\": [\n          1898.0\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"mean\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 645876.9529336845,\n
\"min\": 16.498851422550054,\n   \"max\": 1477495.5,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
171168.478398314\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"std\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 50784.71247265651,\n   \"min\": 4.63248077592887,\n
\"max\": 113698.13974303962,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   113698.13974303962\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"min\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 660181.1448777716,\n
\"min\": 4.47,\n   \"max\": 1476547.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   1311.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"25%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 652710.1666554807,\n
\"min\": 12.08,\n   \"max\": 1477021.25,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   77787.75\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"50%\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 648764.1850689455,\n   \"min\": 14.14,\n   \"max\":
1477495.5,\n   \"num_unique_values\": 5,\n   \"samples\": [\
n   128600.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"75%\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 641497.2176487005,\n   \"min\": 22.2975,\n
\"max\": 1477969.75,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   270525.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"max\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 640369.1284858972,\n
\"min\": 33.0,\n   \"max\": 1478444.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   405334.0\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df.describe(include='object').T
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 4,\n \"fields\": [\n
{\n   \"column\": \"count\",\n   \"properties\": {\n
\"dtype\": \"date\",\n   \"min\": \"1898\",\n   \"max\":
\"1898\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n
\"1898\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"unique\",\n   \"properties\": {\n   \"dtype\": \"date\",\n
\"min\": 2,\n   \"max\": 178,\n   \"num_unique_values\": 4,\

```
n        \"samples\": [\n          14\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"top\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"American\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"freq\",\n      \"properties\": {\n
\"dtype\": \"date\",\n        \"min\": \"219\",\n        \"max\":
\"1351\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n
\"584\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Observation:

- Average food preparation time: 27 minutes; Average delivery time: 24 minutes: Preparation takes slightly longer than delivery, suggesting kitchen operations are the main contributor to total order time.
- Average order value: 16.5, with most orders ranging between 12 and 22, showing consistent customer spending.
- Orders are significantly higher on weekends, confirming a clear demand spike during non-working days.
- **Restaurant and cuisine trends:** The platform features 178 unique restaurants, reflecting a diverse food marketplace.Shake Shack is the most frequently ordered restaurant, indicating strong brand preference. There are 14 cuisine types, with American cuisine being the most popular among customers.

Number of orders that are not rated

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

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

Observation:

- There are 736 orders that were not rated, showing a gap in feedback collection. Improving the rating process could provide better insights into customer satisfaction

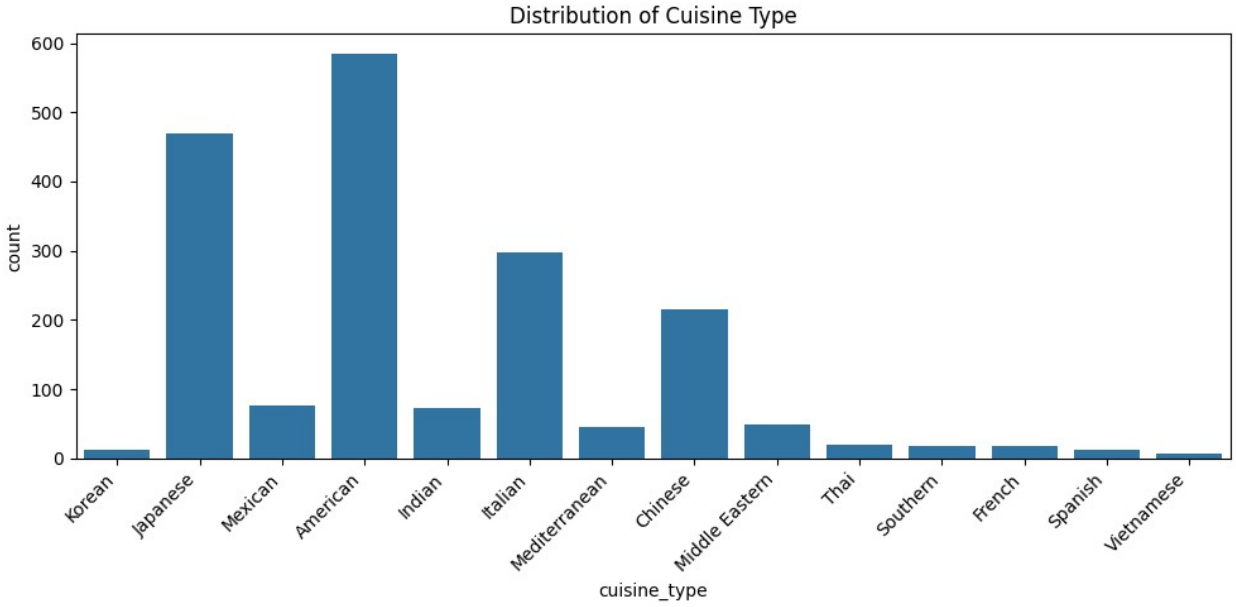
Exploratory Data Analysis (EDA)

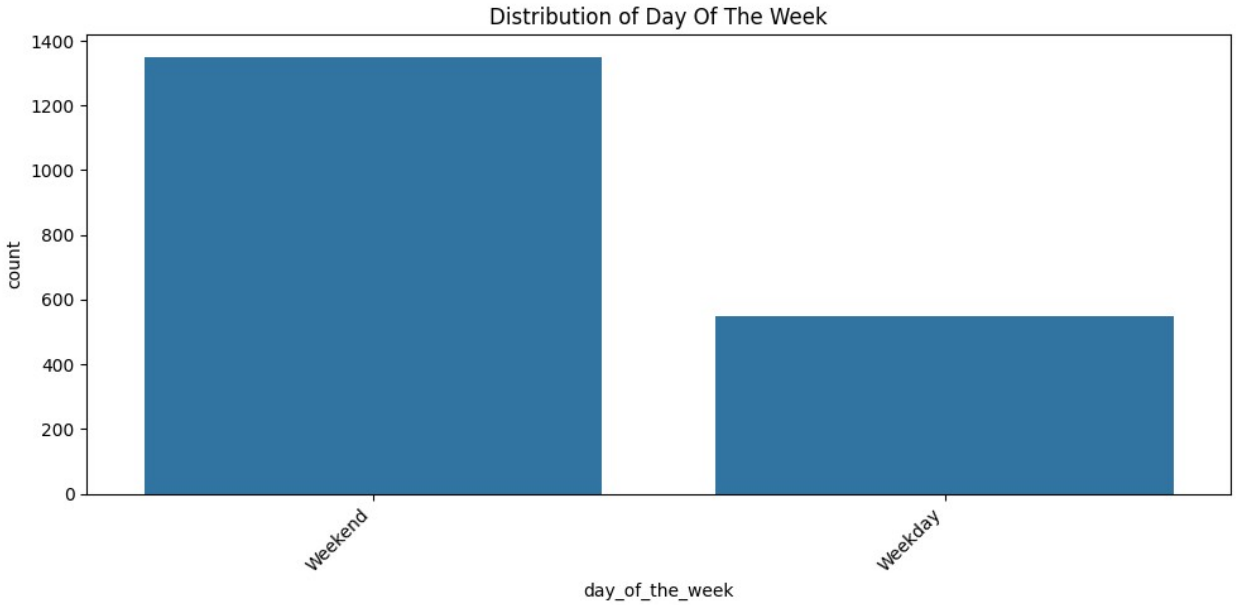
Univariate Analysis

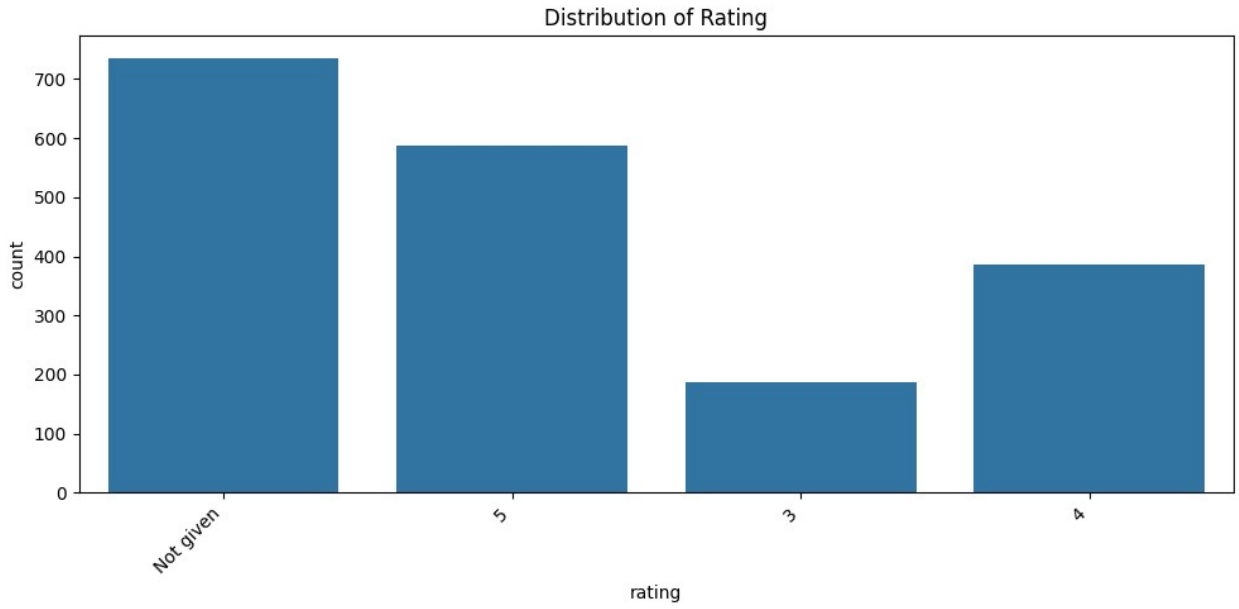
```
# Checking the Unique value in categorical (object) column
for col in ['cuisine_type', 'day_of_the_week', 'rating',
'restaurant_name']:
    print(f"Unique values in {col}:")
    print(df[col].unique())
    print("\n")
```

```
Unique values in cuisine_type:
['Korean' 'Japanese' 'Mexican' 'American' 'Indian' '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']


Unique values in restaurant_name:
['Hangawi' 'Blue Ribbon Sushi Izakaya' 'Cafe 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"]



# Bar Plot
for col in ['cuisine_type', 'day_of_the_week', 'rating']:
    plt.figure(figsize=(10, 5))
    sns.countplot(data=df, x=col)
    plt.title(f'Distribution of {col.replace("_", " ").title()}')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
```


Distribution of Cuisine Type
count
cuisine_type
Korean
Japanese
Mexican
American
Indian
Italian
Mediterranean
Chinese
Middle Eastern
Thai
Southern
French
Spanish
Vietnamese


Distribution of Day Of The Week
count
day_of_the_week
Weekend
Weekday

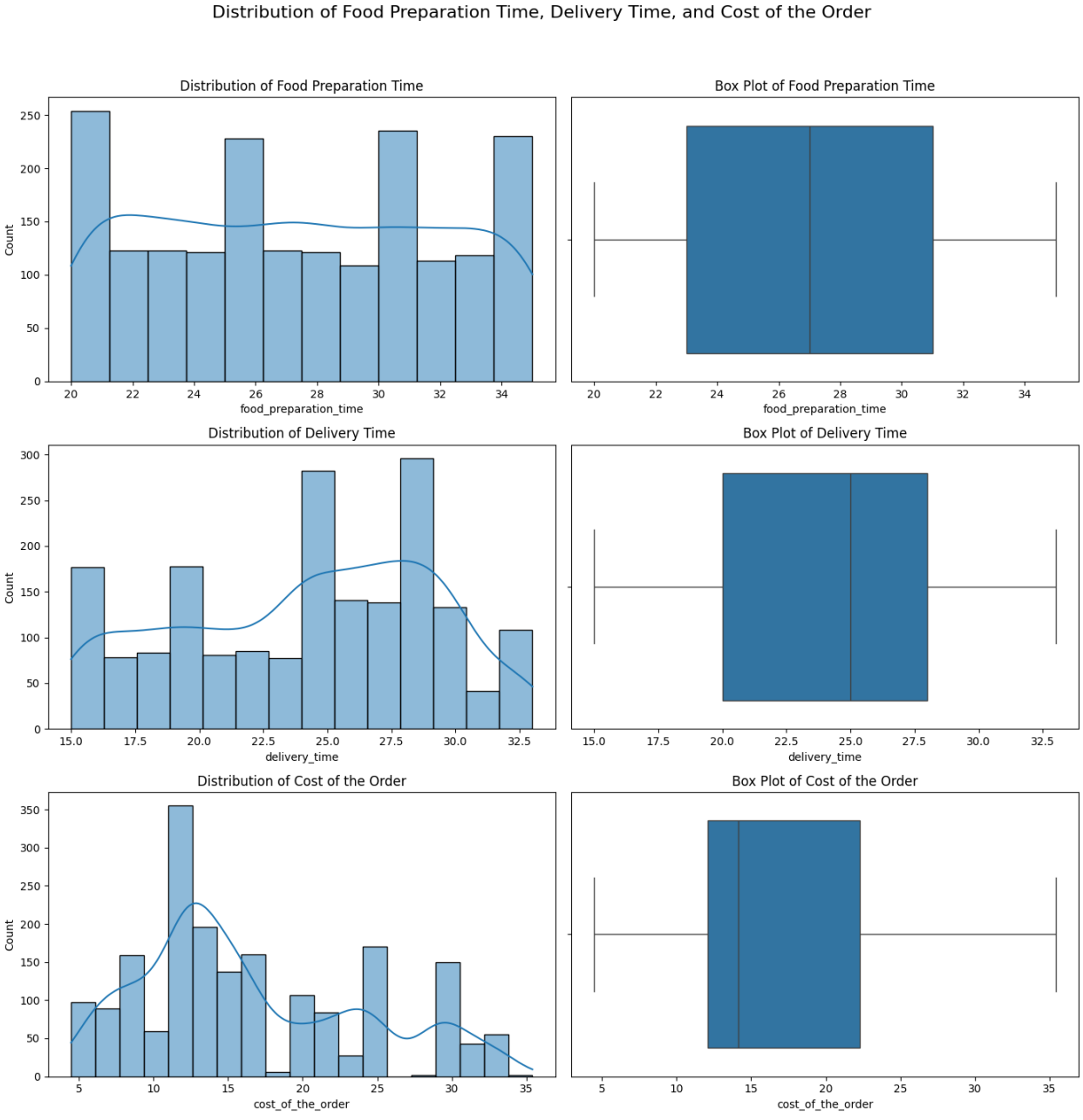


Observation:

- American cuisine and weekend orders drive the majority of revenue.
- A large share of orders have “Not given” ratings, suggesting low customer feedback participation despite generally positive ratings among submitted responses.

```
fig, axes = plt.subplots(3, 2, figsize=(14, 15))
fig.suptitle('Distribution of Food Preparation Time, Delivery Time,
and Cost of the Order', fontsize=16)

# Food Preparation Time - Histogram
sns.histplot(data=df, x='food_preparation_time', kde=True, ax=axes[0,
0])
axes[0, 0].set_title('Distribution of Food Preparation Time')

# Food Preparation Time - Boxplot
sns.boxplot(data=df, x='food_preparation_time', ax=axes[0, 1])
axes[0, 1].set_title('Box Plot of Food Preparation Time')

# Delivery Time - Histogram
sns.histplot(data=df, x='delivery_time', kde=True, ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Delivery Time')

# Delivery Time - Boxplot
sns.boxplot(data=df, x='delivery_time', ax=axes[1, 1])
axes[1, 1].set_title('Box Plot of Delivery Time')

# Cost of the Order - Histogram
sns.histplot(data=df, x='cost_of_the_order', kde=True, ax=axes[2, 0])
axes[2, 0].set_title('Distribution of Cost of the Order')
```

```
# Cost of the Order - Boxplot
sns.boxplot(data=df, x='cost_of_the_order', ax=axes[2, 1])
axes[2, 1].set_title('Box Plot of Cost of the Order')

plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout to prevent title overlap
plt.show()
```



Observation:

- Food preparation time averages 27 minutes, slightly higher than the average delivery time of 24 minutes, making preparation the main contributor to total order time.
- The average order cost is around 16.5, with most orders concentrated in the mid-price range, indicating consistent customer spending behavior.

Which are the top 5 restaurants in terms of the number of orders received?

```
# Get top 5 restaurants with highest number of orders
df['restaurant_name'].value_counts().head(5)
```

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

Which is the most popular cuisine on weekends?

```
# Get most popular cuisine on weekends
df_weekend = df[df['day_of_the_week'] == 'Weekend']
df_weekend['cuisine_type'].value_counts().idxmax()
```

```
{"type":"string"}
```

What % of the orders cost more than 20 dollars?

```
# Get orders that cost above 20 dollars
df_greater_than_20 = df[df['cost_of_the_order']>20]

# Calculate the number of total orders where the cost is above 20 dollars
print('The number of total orders that cost above 20 dollars is:', df_greater_than_20.shape[0])

# Calculate percentage of such orders in the dataset
percentage = (df_greater_than_20.shape[0] / df.shape[0]) * 100

print("Percentage of orders above 20 dollars:", round(percentage, 2), '%')
```

```
The number of total orders that cost above 20 dollars is: 555
Percentage of orders above 20 dollars: 29.24 %
```

Top 3 Most Frequent Customers

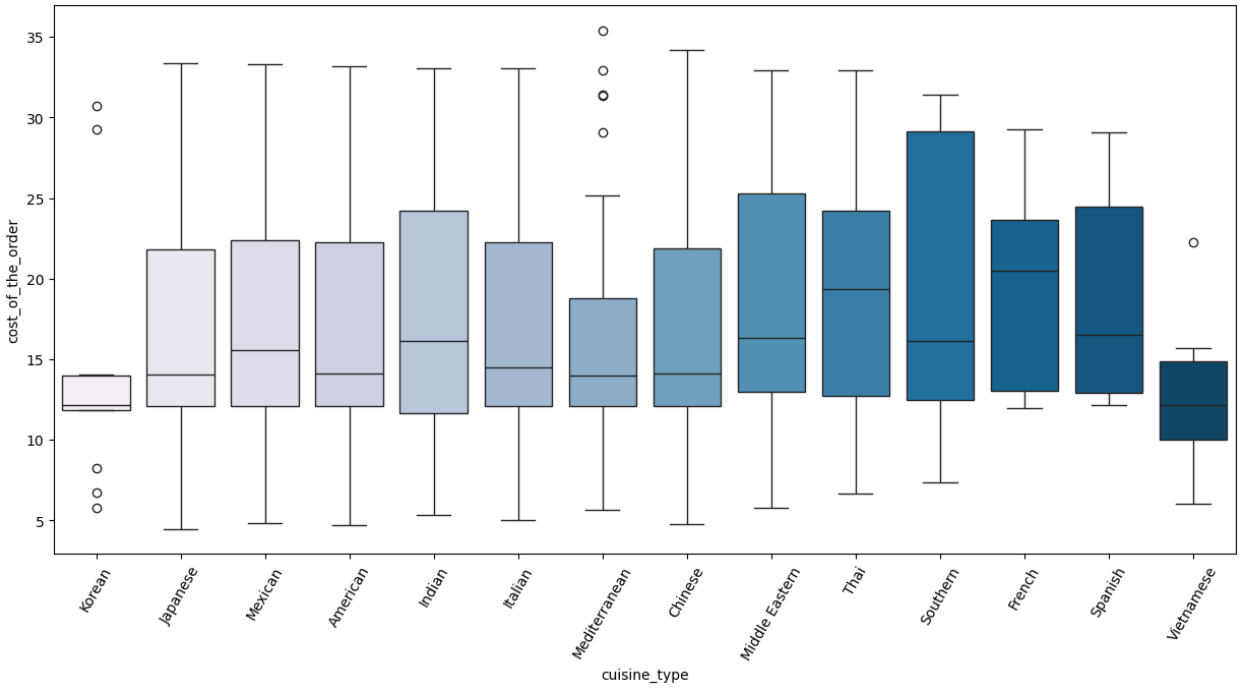
```
# Get the counts of each customer_id
df['customer_id'].value_counts().head(3)
```

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

Multivariate Analysis

Cuisine vs Cost of the order

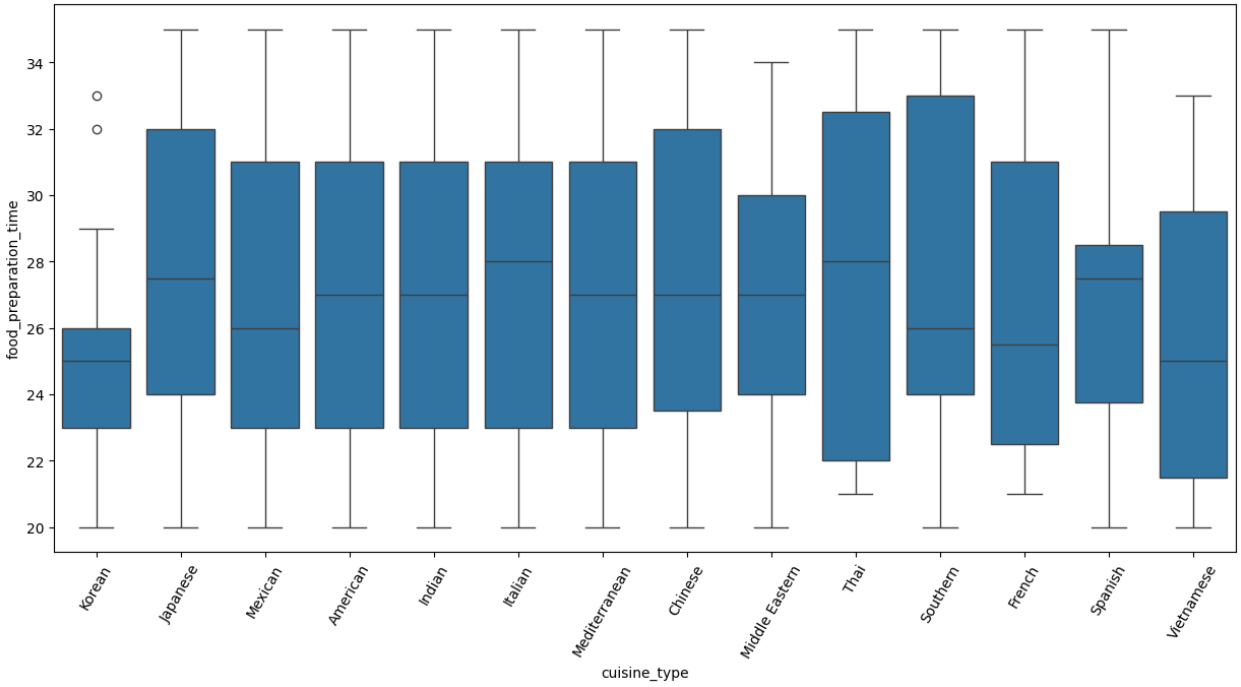
```
# Relationship between cost of the order and cuisine type
plt.figure(figsize=(15,7))
sns.boxplot(x = "cuisine_type", y = "cost_of_the_order", data = df,
palette = 'PuBu')
plt.xticks(rotation = 60)
plt.show()
```



- Higher-end cuisines command higher order values, while some cuisines serve as low-cost options.
- Most cuisines fall within the 12–20 cost range, showing a mid-priced market structure.

Cuisine vs Food Preparation time

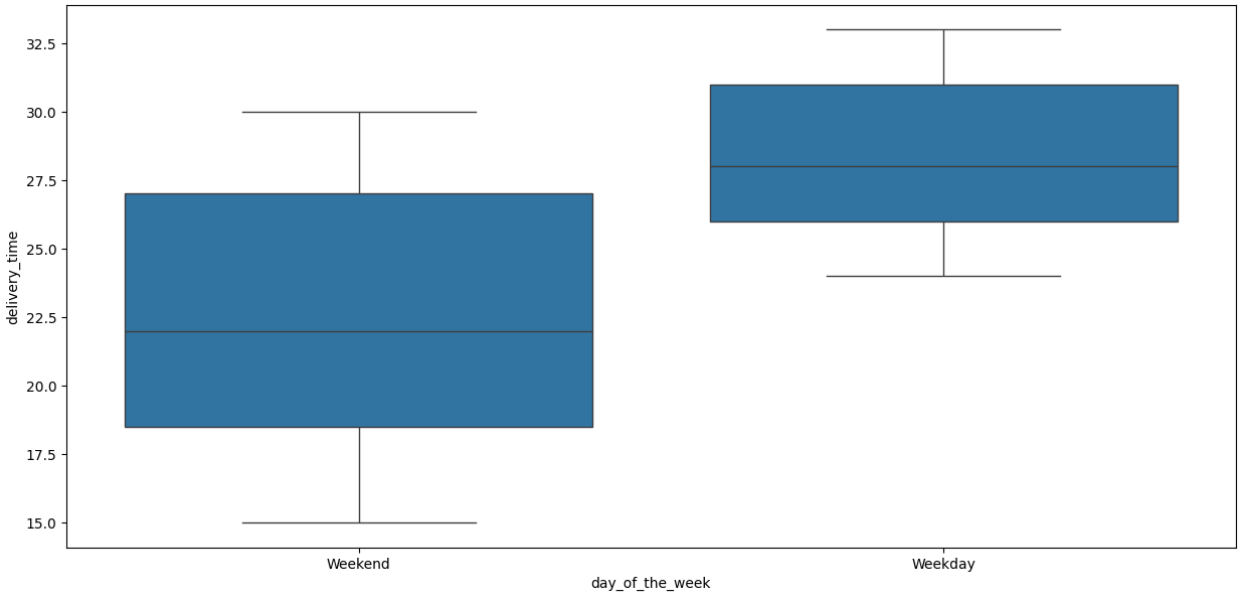
```
# Relationship between food preparation time and cuisine type
plt.figure(figsize=(15,7))
sns.boxplot(x='cuisine_type', y='food_preparation_time', data=df)
plt.xticks(rotation = 60)
plt.show()
```



- Preparation time is operationally stable and not strongly influenced by cuisine type.

Day of the Week vs Delivery time

```
# Relationship between day of the week and delivery time
plt.figure(figsize=(15,7))
sns.boxplot(x='day_of_the_week', y='delivery_time', data=df)
plt.show()
```



- Delivery performance is consistent overall, but weekday deliveries tend to be slightly slower than weekend ones.

Restauratnt Name vs Cost of the order

```
df.groupby(['restaurant_name'])
['cost_of_the_order'].sum().sort_values(ascending = False).head(14)
```

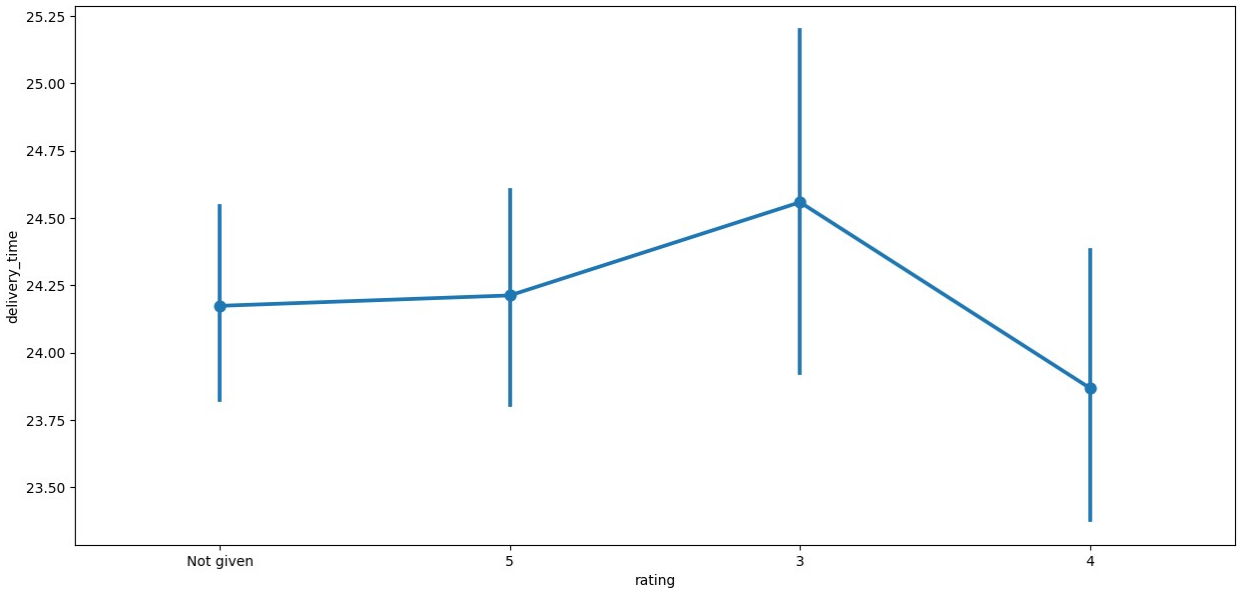
```
restaurant_name
Shake Shack                      3579.53
The Meatball Shop                2145.21
Blue Ribbon Sushi                1903.95
Blue Ribbon Fried Chicken        1662.29
Parm                             1112.76
RedFarm Broadway                  965.13
RedFarm Hudson                    921.21
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
```

- Revenue is concentrated among a few top restaurants, particularly Shake Shack.

Rating vs Delivery time

```
# Relationship between rating and delivery time
plt.figure(figsize=(15, 7))
```

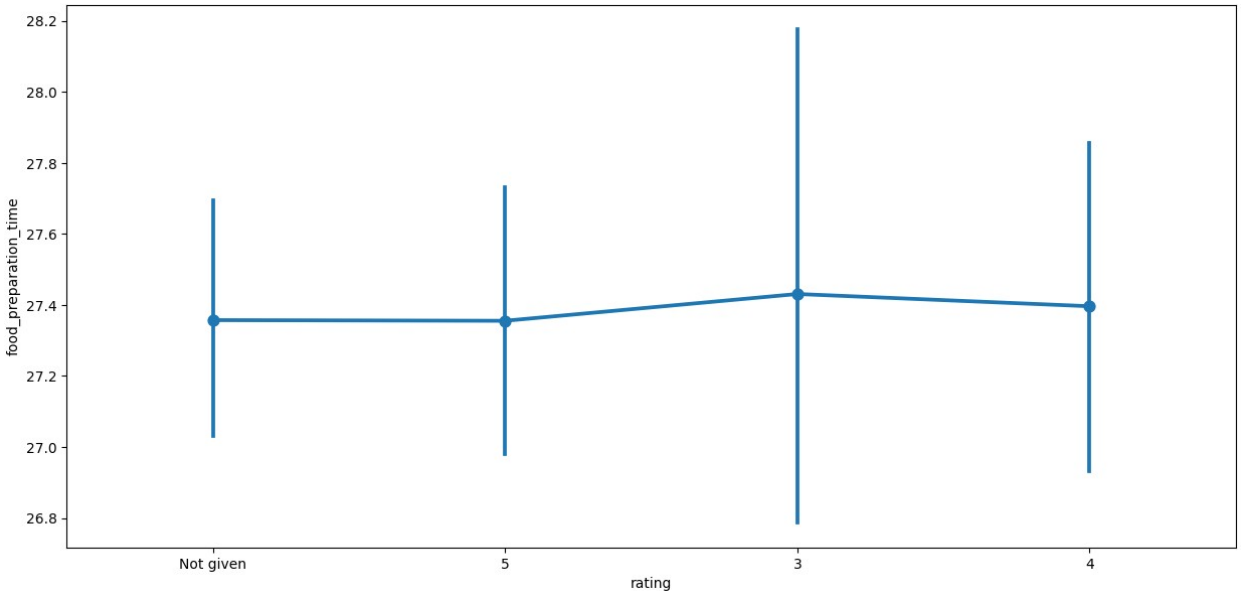
```
sns.pointplot(x = 'rating', y = 'delivery_time', data = df)
plt.show()
```



- Orders with a rating of 3 show the highest average delivery time and the widest variation.
- Delivery time variability is more closely linked to lower ratings than preparation time.

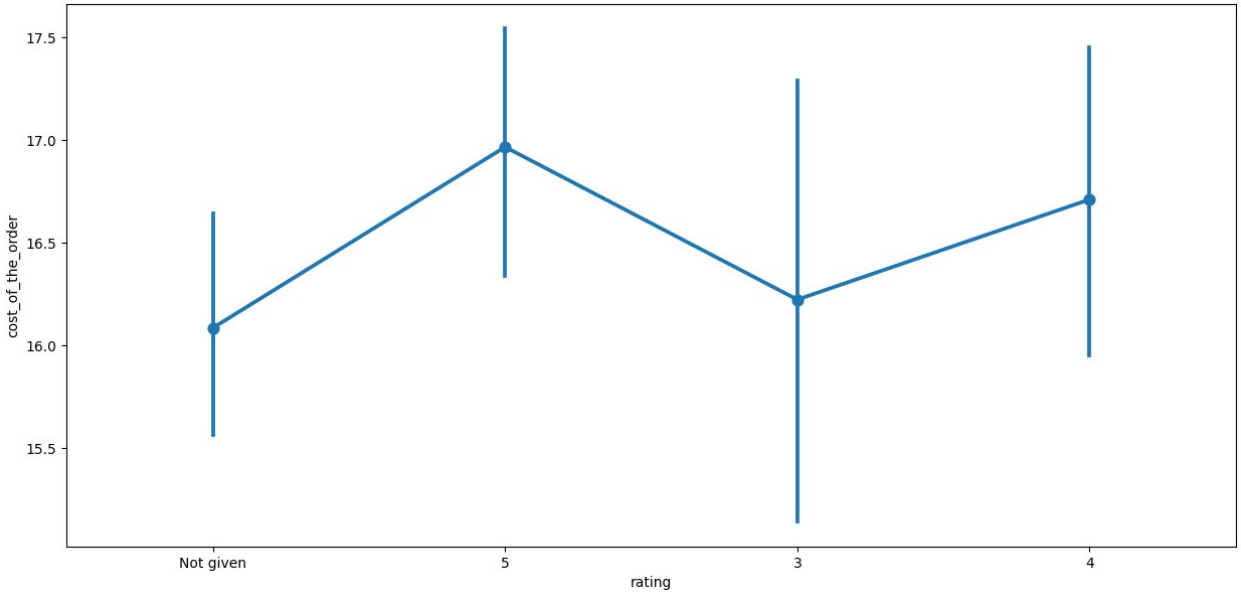
Rating vs Food preparation time

```
# Relationship between rating and food preparation time
plt.figure(figsize=(15, 7))
sns.pointplot(x='rating', y='food_preparation_time', data=df)
plt.show()
```



Rating vs Cost of the order

```
# Relationship between rating and cost of the order
plt.figure(figsize=(15, 7))
sns.pointplot(x='rating', y='cost_of_the_order', data=df)
plt.show()
```

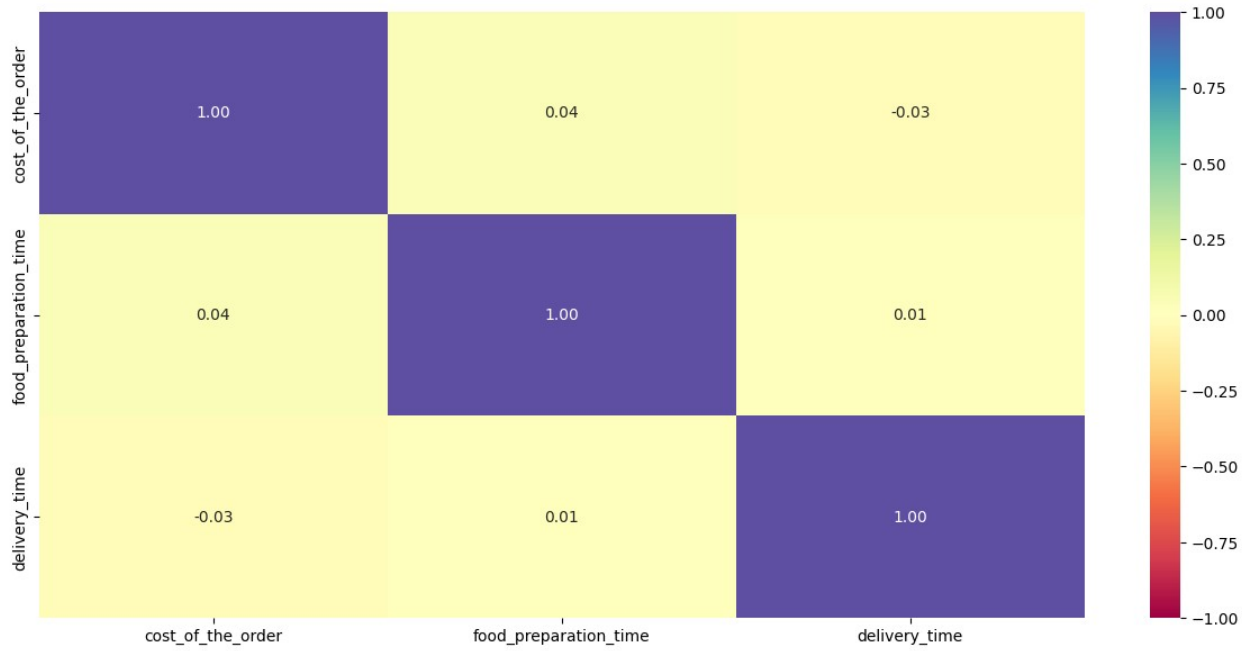


- Customer satisfaction appears to increase with order value.

Correlation Check

```
# Plot the heatmap
col_list = ['cost_of_the_order', 'food_preparation_time',
'delivery_time']
```

```
plt.figure(figsize=(15, 7))
sns.heatmap(df[col_list].corr(), annot=True, vmin=-1, vmax=1,
fmt=".2f", cmap="Spectral")
plt.show()
```



- Correlation values among cost, preparation time, and delivery time are near zero. Operational metrics behave as separate systems, not strongly influencing each other.

Real World Scenario: 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. Finding

the restaurants fulfilling the criteria to get the promotional offer.

```
# 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 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()
```

{"summary":"{\n \"name\": \"df_rating_count\",\n \"rows\": 156,\n
\"fields\": [\n   {\n   \"column\": \"restaurant_name\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 156,\n   \"samples\": [\n   \"Tres
Carnes\",\n   \"The Loop\",\n   \"The Odeon\"\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"rating\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 15,\n   \"min\": 1,\n   \"max\": 133,\n
\"num_unique_values\": 29,\n   \"samples\": [\n   2,\n
13,\n   19\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_rating_count"}

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

# Filter to get the data of restaurants that have rating count 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.groupby(['restaurant_name'])
['rating'].mean().sort_values(ascending =
False).reset_index().dropna()
```

{"summary":"{\n \"name\": \"df_mean_4\",\n \"rows\": 4,\n
\"fields\": [\n   {\n   \"column\": \"restaurant_name\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   \"Blue
Ribbon Fried Chicken\",\n   \"Blue Ribbon Sushi\",\n
\"The Meatball Shop\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"rating\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.1264678402938812,\n   \"min\": 4.219178082191781,\n
\"max\": 4.511904761904762,\n   \"num_unique_values\": 4,\n
\"samples\": [\n   4.328125,\n   4.219178082191781,\n
4.511904761904762\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

Only a small subset of restaurants meets both quality and popularity criteria, indicating that the promotional offer is highly selective and targets consistently high-performing restaurants.

Real World Scenario: 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. Finding the net revenue generated by the company across all orders.

```
#function to determine the revenue
def compute_rev(x):
    if x > 20:
        return x*0.25
    elif x > 5:
        return x*0.15
    else:
        return x*0

df['Revenue'] = df['cost_of_the_order'].apply(compute_rev)
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1898,\n  \"fields\":
[\n    {\n      \"column\": \"order_id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 548,\n        \"min\":
1476547,\n        \"max\": 1478444,\n        \"num_unique_values\":
1898,\n        \"samples\": [\n          1477722,\n          1478319,\
n          1477650\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"customer_id\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 113698,\n        \"min\": 1311,\n
\"max\": 405334,\n        \"num_unique_values\": 1200,\n
\"samples\": [\n          351329,\n          49987,\n          345899\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"restaurant_name\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 178,\n
\"samples\": [\n          \"Tortaria\",\n          \"Osteria
Morini\",\n          \"Philippe Chow\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cuisine_type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 14,\n        \"samples\": [\n
\"Thai\",\n          \"French\",\n          \"Korean\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cost_of_the_order\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7.483812110049553,\n        \"min\": 4.47,\n        \"max\": 35.41,\n
\"num_unique_values\": 312,\n        \"samples\": [\n          21.29,\
n          7.18,\n          13.34\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"day_of_the_week\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
```

```
\"Weekday\",\n          \"Weekend\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"5\",\n          \"4\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"food_preparation_time\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4,\n        \"min\": 20,\n        \"max\": 35,\n
\"num_unique_values\": 16,\n        \"samples\": [\n          25,\n
23\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"delivery_time\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 4,\n        \"min\": 15,\n
\"max\": 33,\n        \"num_unique_values\": 19,\n        \"samples\":
[\n          20,\n          21\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Revenue\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2.295598285490868,\n        \"min\":
0.0,\n        \"max\": 8.8525,\n        \"num_unique_values\": 306,\n
\"samples\": [\n          1.1415,\n          2.3355\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
# get the total revenue and print it
total_rev = df['Revenue'].sum()
print('The net revenue is around', round(total_rev, 2), 'dollars')
```

```
The net revenue is around 6166.3 dollars
```

- The total net revenue generated by the company is approximately $6,166.30 using the defined commission rates.
- Most company revenue is generated from mid- to high-value orders, making higher-priced transactions a key driver of platform earnings.

Real World Scenario: The company wants to analyze the total time required to deliver the food. What percentage of orders take more than 60 minutes to get delivered from the time the order is placed? (The food has to be prepared and then delivered).

```
# Calculate total delivery time and add a new column to the dataframe
df to store the total delivery time
df['total_time'] = df['food_preparation_time'] + df['delivery_time']

# Calculate percentage of orders > 60 mins
total_orders = df.shape[0]
orders_gt_60 = df[df['total_time'] > 60].shape[0]
percentage = (orders_gt_60 / total_orders) * 100
```

```
print(f"Percentage of orders taking more than 60 minutes:
{percentage:.2f}%")
```

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

- About 10.54% of orders take longer than 60 minutes to complete (preparation + delivery). This highlights a potential area for operational improvement.

Real World Scenario: 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?

```
# Get the mean delivery time on weekdays and print it
print('The mean delivery time on weekdays is around',
      round(df[df['day_of_the_week'] == 'Weekday']
['delivery_time'].mean()),
     'minutes')

print('The mean delivery time on weekends is around',
      round(df[df['day_of_the_week'] == 'Weekend']
['delivery_time'].mean()),
     'minutes')
```

```
The mean delivery time on weekdays is around 28 minutes
The mean delivery time on weekends is around 22 minutes
```

- The mean delivery time on weekdays is around 28 minutes, whereas on weekends it is around 22 minutes. Delivery is faster on weekends.

Actionable Insights and Recommendations

Insights

1. **Demand concentration:** American cuisine and a few top restaurants generate the majority of orders and revenue.
2. **Stable spending pattern:** Average order value is about $16.5, with most orders in the mid-price range.
3. **Operational balance:** Preparation time (~27 min) is slightly higher than delivery time (~24 min), but both are consistent.
4. **Delivery impacts satisfaction:** Lower ratings are associated with slower and more variable delivery times.
5. **Low feedback participation:** A large share of orders have no ratings, limiting service quality insights.
6. **High-value orders drive results:** Higher-cost orders contribute more to both revenue and customer satisfaction.

7. **Weekend efficiency:** Deliveries are faster on weekends despite higher demand.
8. **Improvement opportunity:** About 10.5% of orders exceed 60 minutes, indicating operational delays.

Recommendations

- **Improve delivery efficiency**, especially on weekdays, to reduce low-rating incidents.
- **Encourage customer feedback** through incentives and simplified rating processes.
- **Promote high-performing restaurants** and popular cuisines to drive revenue.
- **Increase average order value** through bundles, upselling, and premium offerings.
- **Optimize kitchen operations** to reduce total order completion time.

Conclusion

The Analysis shows stable operations and consistent customer spending, with demand concentrated among a few cuisines and top-performing restaurants. Delivery performance has the strongest impact on customer satisfaction, while higher-value orders drive both revenue and ratings. By improving delivery efficiency, increasing feedback participation, and promoting high-value transactions, the company can enhance customer experience and overall profitability.