FoodHub Data Analysis

Project Python Foundations

POST GRADUATE PROGRAM IN AI & MACHINE LEARNING: BUSINESS APPLICATIONS

Saroj Ekka
Aug 24th 2025

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- Data Overview
- EDA - Univariate Analysis
- EDA - Multivariate Analysis
- Appendix

Executive Summary

📊 Key Conclusions (based on FoodHub csv data)

1. **Order Volume & Restaurants**
   - Dataset covers **1,898 orders** across **178 restaurants**.
   - Average order cost is **~$16.5**.

2. **Ratings**
   - Many customers did not provide ratings (**736 "Not given" ≈ 39%**).
   - Among those who rated:
     - **5 stars → 588 orders**
     - **4 stars → 386 orders**
     - **3 stars → 188 orders**
   - Ratings are skewed positively but feedback coverage is low.

3. **Operational Times**
   - **Average preparation time ≈ 27 minutes**.
   - **Average delivery time ≈ 24 minutes**.
   - Only **10.5% of orders** take longer than **60 minutes total** (prep + delivery).

4. **Weekday vs Weekend**
   - **Weekdays:** Mean delivery time ≈ **28 minutes**.
   - **Weekends:** Mean delivery time ≈ **22 minutes**.
     - Faster deliveries on weekends.

5. **Revenue**
   - Net revenue (commission model) is **~$6,166**.
   - Orders above $20 contribute more significantly due to the 25% commission.

6. **Promotional Offers**
   - Only **4 restaurants** qualify for the "high engagement + high rating" promotion:
     - The Meatball Shop, Blue Ribbon Fried Chicken, Shake Shack, Blue Ribbon Sushi.

Executive Summary

🎯 Actionable Insights

1. **Ratings & Feedback**
   - Nearly **40% of customers did not provide ratings**. Lack of feedback is limiting insights into quality.
   - Restaurants with >50 reviews and avg rating >4 are strong brand ambassadors.

2. **Delivery Efficiency**
   - Weekend deliveries are **~20% faster** than weekdays, likely due to traffic/staffing factors.
   - Only 1 in 10 orders exceed 60 minutes, showing good operational performance.

3. **Revenue Drivers**
   - Majority of revenue is coming from orders **> $20** due to higher commission.
   - Smaller orders ($5–20) are frequent but less profitable.

4. **Restaurant Engagement**
   - Only a handful of restaurants are consistently high-rated and high-volume.
   - Long-tail restaurants may need quality improvements or visibility boosts.

Executive Summary

✅ Recommendations

1. **Boost Customer Ratings**
   - Encourage ratings with **small discounts or reward points** for feedback.
   - This reduces “Not given” cases and helps identify issues more clearly.
2. **Improve Weekday Delivery**
   - Partner with delivery providers to handle weekday traffic better.
   - Consider **dynamic delivery fee discounts** to spread orders more evenly across time slots.
3. **Revenue Optimization**
   - Highlight **premium combo offers** above $20 in the app → drives higher commission revenue.
   - For smaller orders, push **“add-on items”** (desserts, drinks) to bump order value above $20.
4. **Promotional Campaigns**
   - Promote the **top 4 high-rated restaurants** in ads.
   - Create a **“FoodHub Certified Best Rated” badge** to attract more customers.
5. **Support for Underperforming Restaurants**
   - Provide insights/training to restaurants with **avg ratings ≤3.5**.
   - Offer them app-based promotions to increase order volume and engagement.

Executive Summary - Q17

(Conclusion based on **cuisine type and feedback ratings**)

📊 Conclusions

- **Spanish and Thai cuisines** have the **highest average ratings** (>4.6), but with **very low order counts**
  - underexposed but well-loved.
- **Indian, Mexican, Japanese, Italian, and Chinese cuisines** have both **high ratings (~4.3–4.5)** and **strong order counts**,
  - showing they are the **core demand drivers**.
- **American cuisine** has the **largest order count (584 orders)** but only an average rating of **~4.3**
  - indicating room for quality improvement.

✅ Recommendations

1. **Promote Niche High-rated Cuisines**
   - Increase visibility of **Spanish and Thai restaurants** through promotions and app highlights → convert customer satisfaction into higher order volume.
2. **Strengthen Core Cuisines**
   - Focus on **Indian, Mexican, Japanese, Italian, and Chinese** cuisines with combo deals and targeted ads → maximize revenue from already popular, high-rated cuisines.
3. **Improve American Cuisine Quality**
   - Since **American cuisine has the largest volume but lower relative rating**, work with these restaurants on **quality upgrades** (taste, service consistency). Even a small rating improvement will significantly impact overall customer satisfaction.

Business Problem Overview and Solution Approach

Overview

FoodHub, an online food aggregator, wants to **analyze customer orders and restaurant performance** to gain insights that can improve customer experience and optimize business decisions.

Key challenges include:

- Identifying **which restaurants and cuisines are most in demand**.
- Understanding the **patterns of customer orders** (weekday vs weekend).
- Analyzing **food preparation time and delivery time** to measure efficiency.
- Studying **customer satisfaction through ratings** and its relationship with order cost, preparation, and delivery times.
- Helping FoodHub **strategically improve partnerships** with restaurants and allocate delivery resources better.

The **business goal** is to analyze the data to get a fair idea about the demand of different restaurants which will help them in enhancing their customer experience. In short

- Enhance **customer satisfaction**.
- Improve **delivery efficiency**.
- Support **restaurant selection & promotion strategies**.
- Ultimately, **increase revenue** through better service and operational efficiency.

***Note**: You can use more than one slide if needed*

Business Problem Overview and Solution Approach

Solution Approach / Methodology

The problem can be solved through **Exploratory Data Analysis (EDA), statistical insights, and visualization techniques**:

Step 1: Data Understanding & Cleaning

- Load and explore dataset structure.
- Handle missing values (e.g., ratings not provided).
- Remove duplicates and validate data consistency.
- Convert categorical variables (weekday/weekend, cuisine type, etc.) if needed.

Step 2: Univariate Analysis

- Distribution of **order cost**.
- Most popular **restaurants** and **cuisines**.
- Distribution of **ratings**.
- Typical **food preparation time** and **delivery time**.

Step 3: Bivariate / Multivariate Analysis

- Relationship between **order cost and rating**.
- Effect of **food preparation time and delivery time** on **customer rating**.
- Order demand **weekday vs weekend**.
- Correlation between **restaurant popularity and average rating**.
- Comparison of **different cuisines** in terms of cost, preparation time, and ratings.

Step 4: Key Insights Extraction

- Identify **top-performing restaurants** (high demand + high ratings + low prep/delivery time).
- Find **bottlenecks** (restaurants with high preparation/delivery times leading to lower ratings).
- Determine **cuisine preferences** across customers.
- Recommend strategies for **better delivery resource allocation** (weekend spikes, peak-hour orders).

Step 5: Business Recommendations

- Promote restaurants with **high customer satisfaction**.
- Support restaurants with **high prep times** by suggesting process improvements
- Optimize delivery logistics to **reduce delivery times**.
- Launch **marketing campaigns** based on cuisine and order patterns.

Data Overview - FoodHub Dataset

Dataset Summary

- **Total Records:** *1898*
- **Total Datafields:** 9

Fields Description

- **order_id** → Unique identifier for each order
- **customer_id** → Unique identifier for each customer
- **restaurant_name** → Name of the restaurant
- **cuisine_type** → Cuisine ordered (e.g., Italian, Chinese, Indian)
- **cost_of_the_order** → Cost (in $) of the order
- **day_of_the_week** → Weekday or Weekend order
- **rating** → Rating given by the customer (1–5, may have missing values)
- **food_preparation_time** → Time (minutes) taken by restaurant to prepare food
- **delivery_time** → Time (minutes) taken by delivery person to deliver order

Key Points

- Covers **order lifecycle** from placement → preparation → delivery → customer feedback
- Mix of **categorical, numerical, and ID variables**
- Useful for analyzing:
  ✅ **Customer demand patterns**
  ✅ **Restaurant efficiency**
  ✅ **Delivery performance**
  ✅ **Customer satisfaction**

Data Questions 1 - 2

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

```
[11] # Write your code here
     df.shape
```

```
(1898, 9)
```

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

```
# 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
```

Data Questions 3

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

```
# Write your code
df.isnull().sum()
```

	0
order_id	0
customer_id	0
restaurant_name	0
cuisine_type	0
cost_of_the_order	0
day_of_the_week	0
rating	0
food_preparation_time	0
delivery_time	0

dtype: int64

Observations:

No missing value found

Data Questions 4

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]

Double-click (or enter) to edit

```
# Write your code here
df.describe()
```

	order_id	customer_id	cost_of_the_order	food_preparation_time	delivery_time
count	1.898000e+03	1898.000000	1898.000000	1898.000000	1898.000000
mean	1.477496e+06	171168.478398	16.498851	27.371970	24.161749
std	5.480497e+02	113698.139743	7.483812	4.632481	4.972637
min	1.476547e+06	1311.000000	4.470000	20.000000	15.000000
25%	1.477021e+06	77787.750000	12.080000	23.000000	20.000000
50%	1.477496e+06	128600.000000	14.140000	27.000000	25.000000
75%	1.477970e+06	270525.000000	22.297500	31.000000	28.000000
max	1.478444e+06	405334.000000	35.410000	35.000000	33.000000

Observations:

```
Food preparation time ( once order is placed)
minimum: 20 min
average: 27.37 min
maximum: 35 min
```

Data Questions 5

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

```
# Write the code
df['rating'].value_counts()
```

rating	count
Not given	736
5	588
4	386
3	188

dtype: int64

Observations:

No rated are 736

Univariate Analysis

Summary

- **Categorical columns**
  - **IDs, restaurant_name, cuisine_type, day_of_the_week**
  - Use **countplots/frequency tables**.

- **Numerical columns**
  - **cost_of_the_order, rating, food_preparation_time, delivery_time)**
  - Use **histograms, boxplots, and descriptive statistics**.

- **ID variables:** Not meaningful for univariate analysis, except customer_id for frequency of repeat orders.

Univariate Analysis (column wise)

1. Order_id

- **Type:** Categorical (ID variable, unique).
- **Analysis:** Not meaningful for univariate statistics since it's just an identifier.

2. customer_id

- **Type:** Categorical (ID variable, can repeat).
- **Analysis:**
  - Countplot can show how many orders each customer placed.
  - Most customers likely have few orders, while some may be frequent users.

3. restaurant_name

- **Type:** Categorical.
- **Analysis:**
  - Frequency distribution of orders per restaurant.
  - Identify top restaurants with the highest demand.
  - Long-tail distribution likely (few very popular restaurants, many with fewer orders).

4. cuisine_type

- **Type:** Categorical.
- **Analysis:**
  - Countplot / bar chart to show most popular cuisines.
  - Helps identify customer preferences (e.g., Indian vs Italian vs Chinese).

5. cost_of_the_order

- **Type:** Numerical (continuous).
- **Analysis:**
  - Histogram / boxplot to see spending pattern.
  - Summary statistics (min, mean, median, max).
  - Check for outliers (very high-cost orders).

6. day_of_the_week

- **Type:** Categorical (Weekday / Weekend).
- **Analysis:**
  - Countplot to compare order frequency on weekdays vs weekends.
  - Insight: Higher demand expected on weekends.

Univariate Analysis (column wise)

7. rating

- **Type:** Numerical (discrete, 1–5).
- **Analysis:**
  - Histogram to check rating distribution.
  - Summary stats (average rating).
  - Check how many customers didn’t give a rating (missing values).

8. food_preparation_time

- **Type:** Numerical (continuous).
- **Analysis:**
  - Histogram / boxplot to check distribution.
  - Summary stats (min, mean, max).
  - Outliers: Some restaurants may take unusually long.

9. delivery_time

- **Type:** Numerical (continuous).
- **Analysis:**
  - Histogram to understand delivery speed distribution.
  - Summary stats (min, mean, max).
  - Outliers may indicate traffic/weather issues.

Univariate Analysis - Questions 6

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

```
[16] df['order_id'].nunique()
```

```
1898
```

```
[17] df['customer_id'].nunique()
```

```
1200
```

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

```
178
```

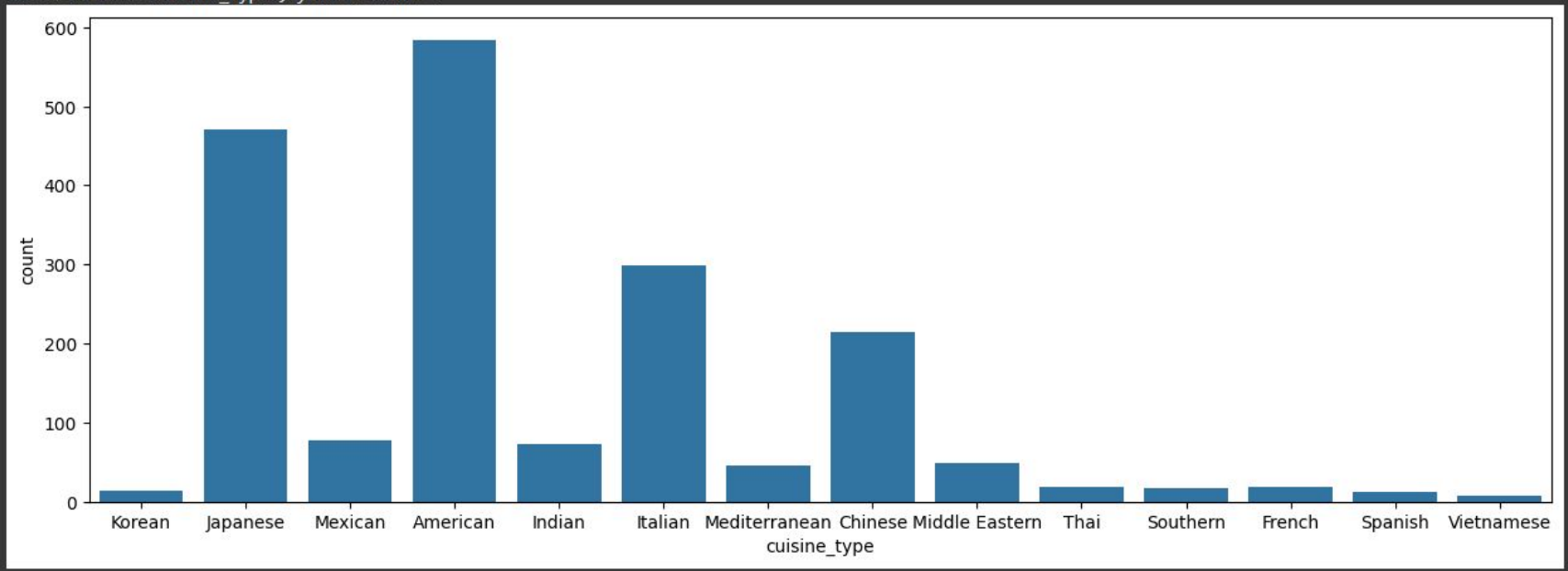
```
[19] df['cuisine_type'].nunique()
```

```
14
```

Univariate Analysis - Questions 6

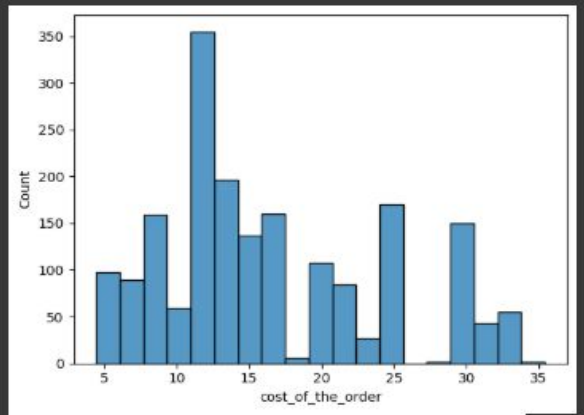
```
[20] plt.figure(figsize = (15,5))
     sns.countplot(data = df, x = 'cuisine_type') ## Create a countplot for cuisine type.
```

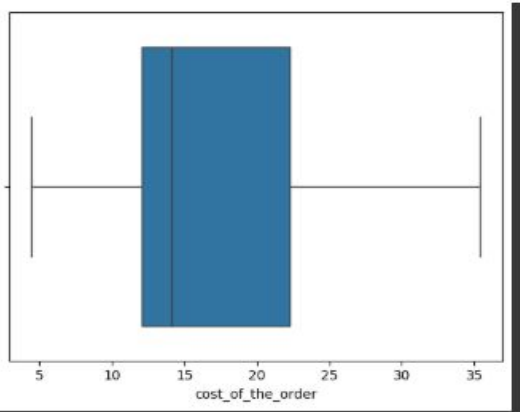
```
<Axes: xlabel='cuisine_type', ylabel='count'>
```



Univariate Analysis - Questions 6

```
sns.histplot(data=df,x='cost_of_the_order') ## Histogram for the cost of order
plt.show()
sns.boxplot(data=df,x='cost_of_the_order') ## Boxplot for the cost of order
plt.show()
```

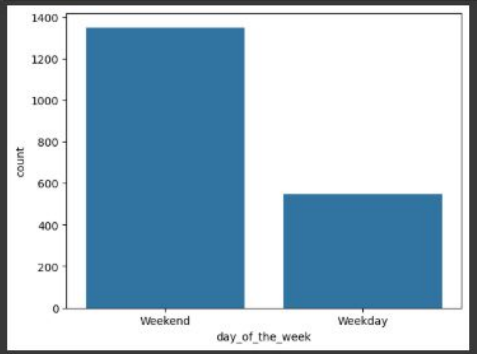





```
[27] # # Check the unique values
df['day_of_the_week'].nunique() ## Complete the code to check unique values for the 'day_of_the_week' column
```

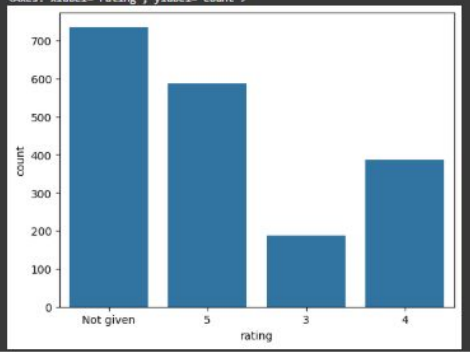
```
2
```

```
# Write the code here
sns.countplot(data = df, x = 'day_of_the_week')
plt.show()
```



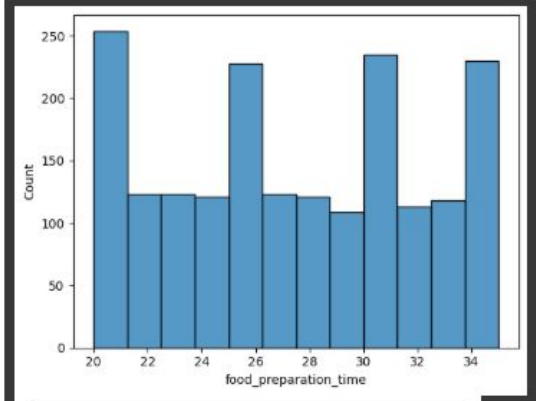

```
[24] # Check the unique values
df['rating'].nunique() ## Complete the code to check unique values for the 'rating' column
sns.countplot(data = df, x = 'rating') ## Complete the code to plot bar graph for 'rating' column
```

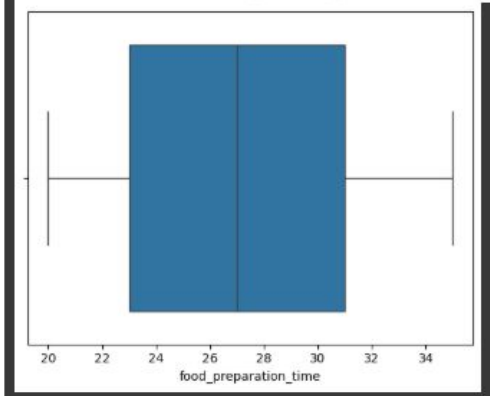
```
<Axes: xlabel='rating', ylabel='count'>
```



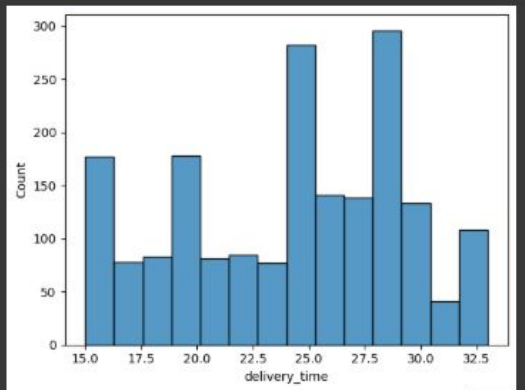
Univariate Analysis - Questions 6

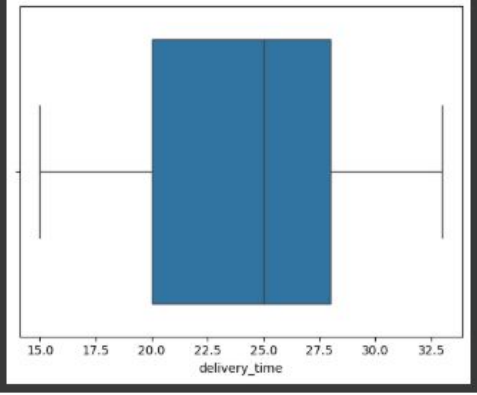
```
sns.histplot(data=df,x='food_preparation_time') ## Complete the code to plot the histogram for the cost of order
plt.show()
sns.boxplot(data=df,x='food_preparation_time') ## Complete the code to plot the boxplot for the cost of order
plt.show()
```





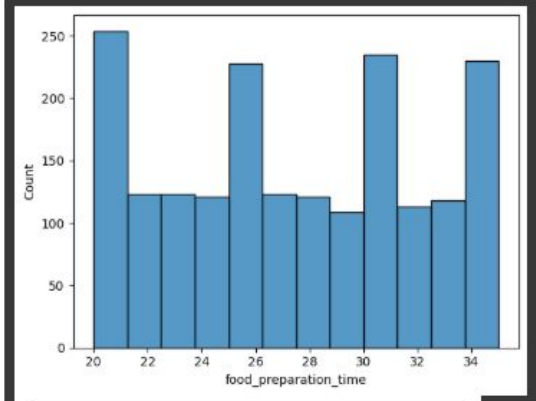

```
sns.histplot(data=df,x='delivery_time') ## Complete the code to plot the histogram for the delivery time
plt.show()
sns.boxplot(data=df,x='delivery_time') ## Complete the code to plot the boxplot for the delivery time
plt.show()
```

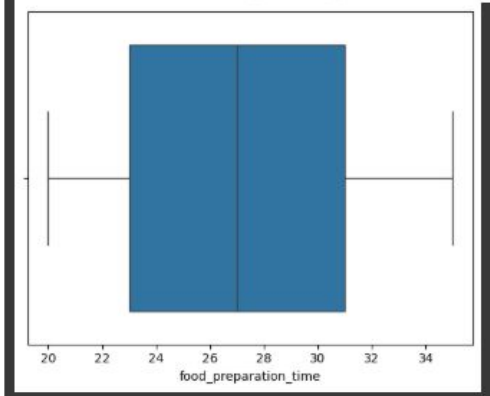




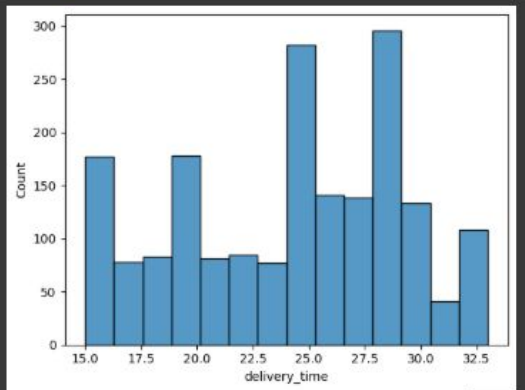
Univariate Analysis - Questions 6

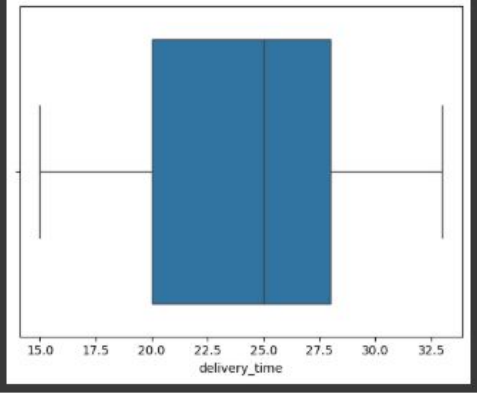
```
sns.histplot(data=df,x='food_preparation_time') ## Complete the code to plot the histogram for the cost of order
plt.show()
sns.boxplot(data=df,x='food_preparation_time') ## Complete the code to plot the boxplot for the cost of order
plt.show()
```






```
sns.histplot(data=df,x='delivery_time') ## Complete the code to plot the histogram for the delivery time
plt.show()
sns.boxplot(data=df,x='delivery_time') ## Complete the code to plot the boxplot for the delivery time
plt.show()
```





Univariate Analysis - Questions 7 -8

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

```
# Write the code here
df['restaurant_name'].value_counts()
```

restaurant_name	count
Shake Shack	219
The Meatball Shop	132
Blue Ribbon Sushi	119
Blue Ribbon Fried Chicken	96
Parm	68
...	...
Rye House	1
Hiroko's Place	1
Frank Restaurant	1
Sarabeth's West	1
'wichcraft	1

178 rows × 1 columns

dtype: int64

```
#### Observations:
Yop5 restaurant are
Shake Shack
The Meatball Shop
Blue Ribbon Sushi
Blue Ribbon Fried Chicken
Parm
```

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

```
# Write the code here
# Get most popular cuisine on weekends
df_weekend = df[df['day_of_the_week'] == 'Weekend']
df_weekend['cuisine_type'].value_counts() ## Complete the code to check unique values for the
```

cuisine_type	count
American	415
Japanese	335
Italian	207
Chinese	163
Mexican	53
Indian	49
Middle Eastern	32
Mediterranean	32
Thai	15
French	13
Korean	11
Southern	11
Spanish	11
Vietnamese	4

dtype: int64

Observations:

most popular for weekend is American

Univariate Analysis - Questions 9-11

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

```
[32] # Write the code here
     # Get orders that cost above 20 dollars
     df_greater_than_20 = df[df['cost_of_the_order']>20] ## Write the appropriate column name to get the orders having cost above $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 %
```

Observations: 29.24%

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

```
[33] # Write the code here
     # Get the mean delivery time
     mean_del_time = df['delivery_time'].mean()  ## Write the appropriate function to obtain the mean delivery time

     print('The mean delivery time for this dataset is', round(mean_del_time, 2), 'minutes')
```

```
The mean delivery time for this dataset is 24.16 minutes
```

Observations: mean is 24.16

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]

```
# Write the code here
 #Get the counts of  each customer_id
df['customer_id'].value_counts().head(3)  ## Write the appropriate column name to get the top 5 cmost frequent customers
```

customer_id	count
52832	13
47440	10
83287	9

dtype: int64

Observations: Above are top 3 customer ID's

Multivariate Analysis

- Summary
  - **Numerical × Numerical** → scatter plots, correlation heatmaps.
  - **Categorical × Numerical** → boxplots, grouped bar charts.
  - **Categorical × Categorical** → stacked bar charts, heatmaps.

Multivariate Analysis - Column relationships

1. **Restaurant Demand vs. Cuisine Type**
   - **Columns:** `restaurant_name` × `cuisine_type`
   - **Analysis:**
     - Which cuisines are served by which restaurants?
     - Do certain restaurants specialize in one cuisine or offer multiple?
     - Visualization: Grouped bar chart or heatmap.

2. **Order Cost vs. Rating**
   - **Columns:** `cost_of_the_order` × `rating`
   - **Analysis:**
     - Do higher-cost orders receive better ratings?
     - Is there a threshold where very high cost leads to lower ratings?
     - Visualization: Boxplot or scatter plot with regression line.

3. **Preparation Time vs. Delivery Time vs. Rating**
   - **Columns:** `food_preparation_time` × `delivery_time` × `rating`
   - **Analysis:**
     - Do longer preparation/delivery times lead to lower ratings?
     - Correlation between prep time and delivery time (are they linked?).
     - Visualization: Bubble chart (x=prep time, y=delivery time, size/color = rating).

4. **Day of the Week vs. Number of Orders vs. Cost**
   - **Columns:** `day_of_the_week` × `order count` × `cost_of_the_order`
   - **Analysis:**
     - Are orders higher on weekends compared to weekdays?
     - Do customers spend more on weekends?
     - Visualization: Bar chart (avg cost per order by day).

Multivariate Analysis - Column relationships

5. Cuisine Type vs. Average Delivery Time

- **Columns:** `cuisine_type` × `delivery_time`
- **Analysis:**
  - Does delivery time vary by cuisine? (e.g., fast food vs. full meals).
  - Helps optimize delivery expectations.
  - Visualization: Boxplot.

6. Restaurant Popularity vs. Customer Ratings

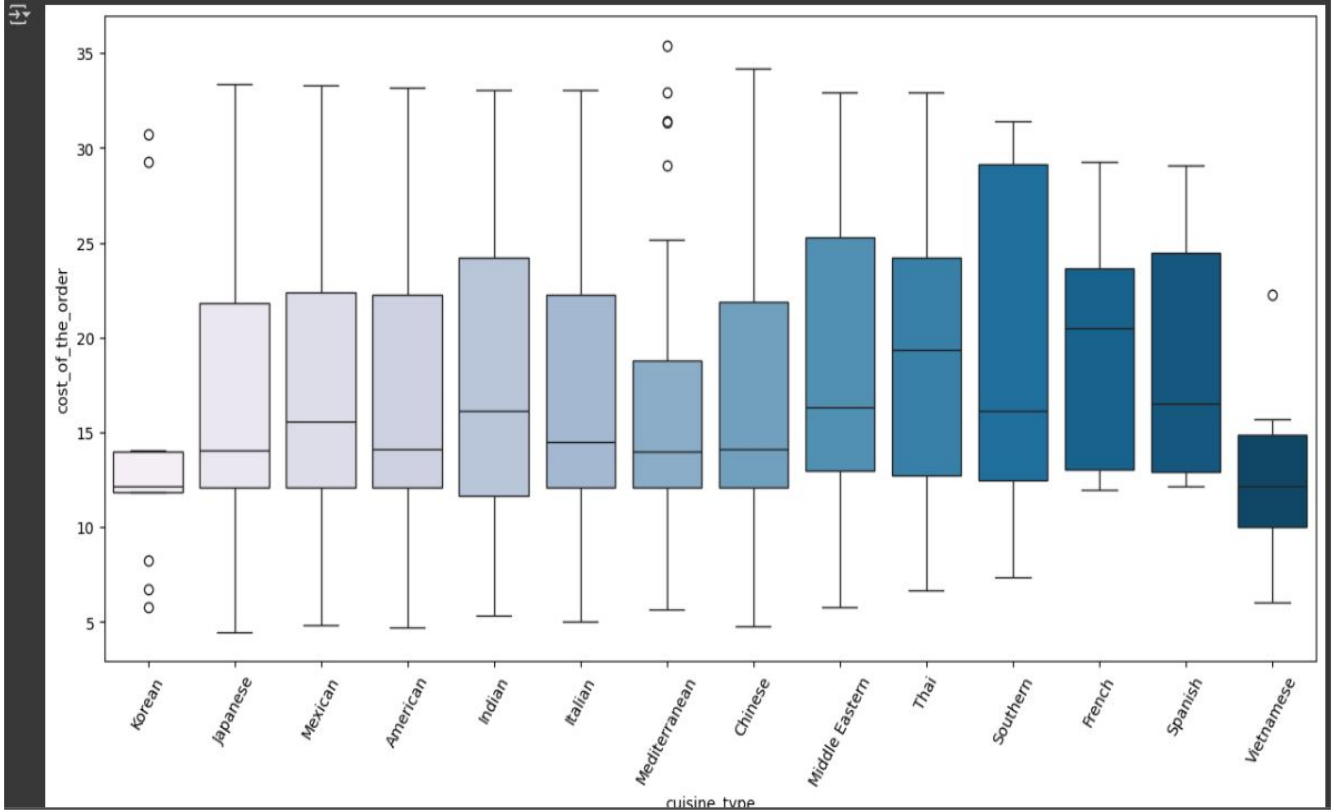
- **Columns:** `restaurant_name` × `rating`
- **Analysis:**
  - Do popular restaurants always have higher ratings?
  - Detect restaurants with high orders but low customer satisfaction.
  - Visualization: Scatter plot (x=order count, y=avg rating).

7. Cost vs. Prep Time vs. Delivery Time

- **Columns:** `cost_of_the_order` × `food_preparation_time` × `delivery_time`
- **Analysis:**
  - Are expensive meals associated with higher prep times?
  - Does delivery time increase with higher-cost orders?
  - Visualization: 3D scatter plot or pairplot.

Multivariate Analysis - Questions 12

```
# Write the code# 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)
plt.show()
```

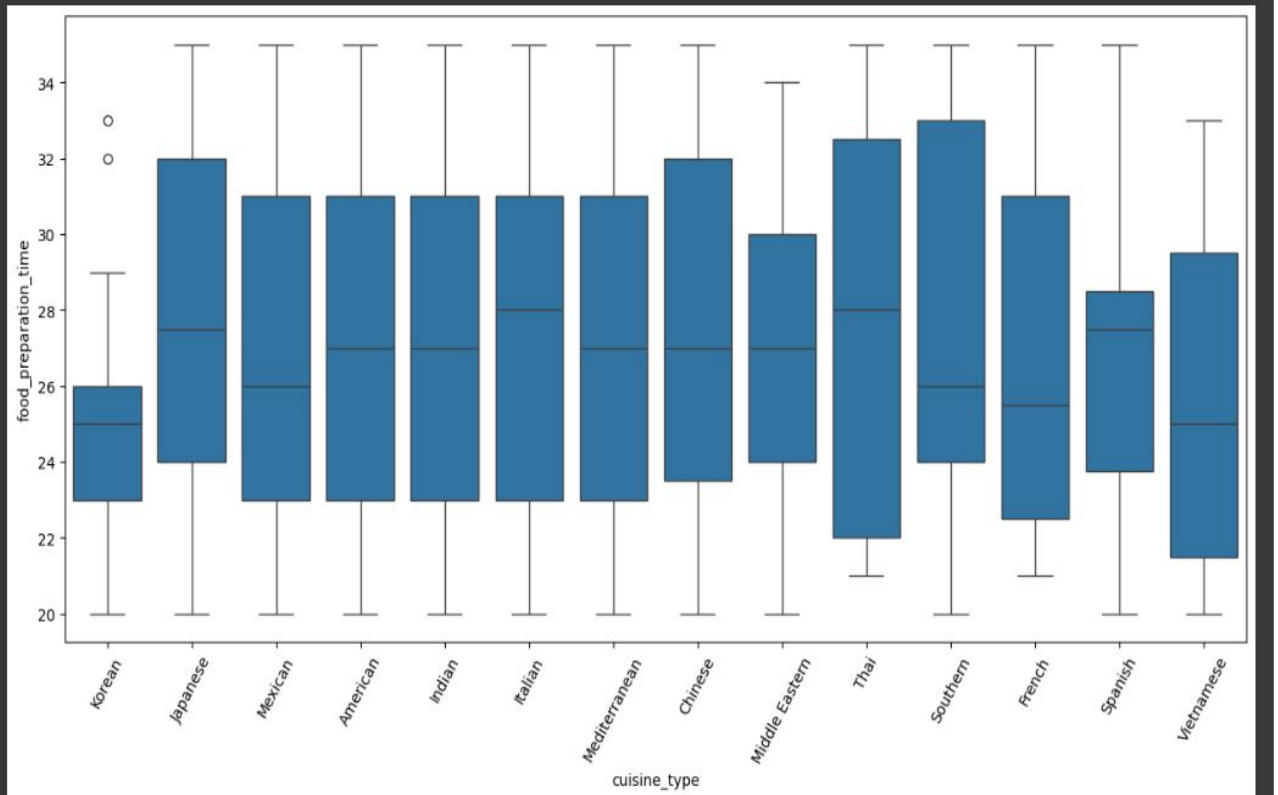


Based on Cuisine type vs cost

- Korean is narrow spread of cost whereas southern has most wide spread
- Vietnamese has also less expensive on average vs French has more expensive
- Mediterranean has more outliner means more premium dishes
- 3 Budget friendly cuisines are Vietnamese, Korean, Mediterranean
- 3 Premium cuisines are French, Spanish & Thai
- 4 Most wide spread range cuisines as per cost are Southern, Italian, Middle Eastern & Spanish

Multivariate Analysis - Questions 12

```
[39] # 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()
```

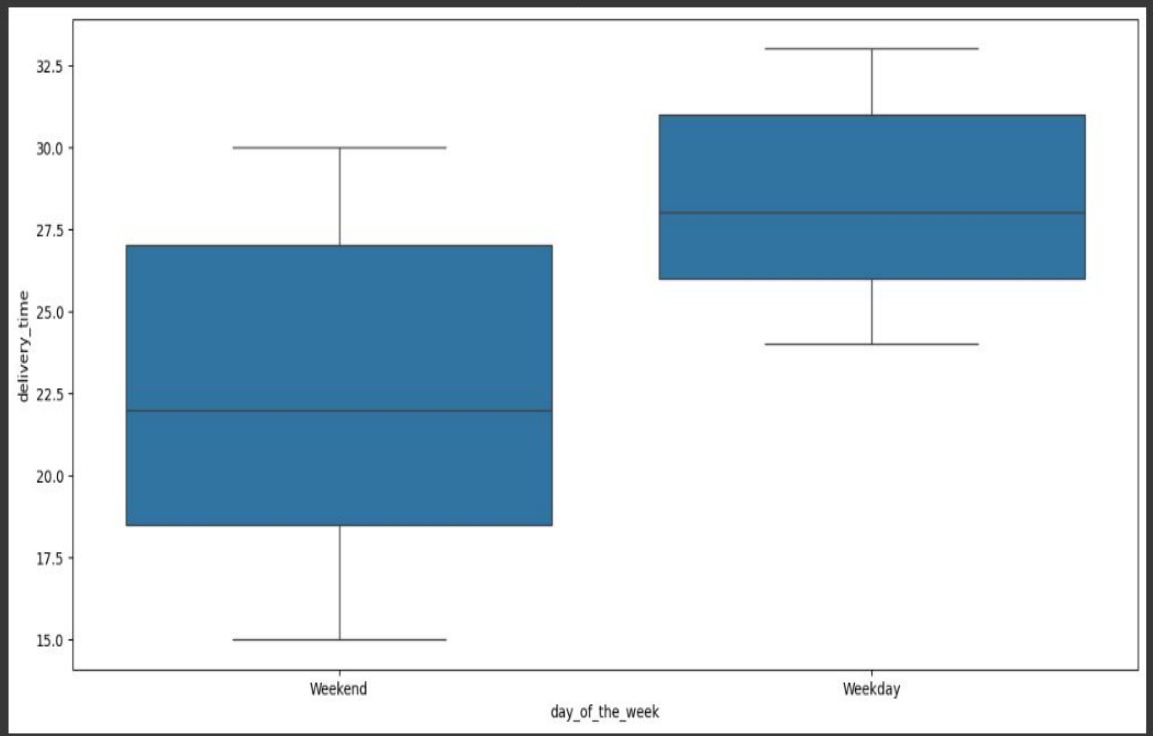


Based on food preparation time and cuisine type

- Vietnamese/Korean are the fastest preparation time
- Italian/Thai takes longer
- Only Korean has some outlier means some dishes takes longer
- Cuisines with wider boxes (e.g., thai → inconsistent preparation times across restaurants.
- Narrower boxes (e.g., Korean) → consistent preparation times.
- Average time for most of is close to 27 min :
  - Italian, Thai,Spanish
  - Japanese,Chinese,
  - American,Mediterranean
  - ,Indian,Middle Eastern , Southern

Multivariate Analysis - Questions 12

```
#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) ## Complete the code to visualize the relationship between day of the week and delivery time using boxplot
plt.show()
```



Based on day of the week and delivery time

- Weekend delivery is faster than weekdays
- Weekend deliveries are faster (avg ~22.5 mins) compared to weekdays (avg ~28.3 mins).
- Weekdays also have a narrower time range (24–33 mins), while weekends show more flexibility (15–30 mins).
-

Multivariate Analysis - Questions 12

```
#Run the below code and write your observations on the revenue generated by the restaurants.
df.groupby(['restaurant_name'])['cost_of_the_order'].sum().sort_values(ascending = False).head(14)
```

restaurant_name	cost_of_the_order
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

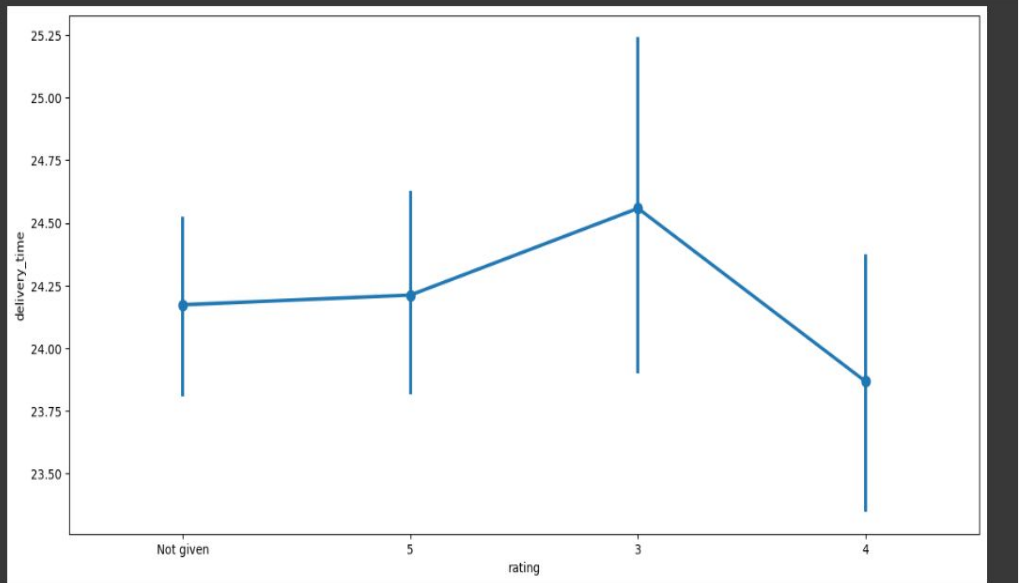
dtype: float64

Based on revenue generated by the restaurants.

- Top most revenue generation restaurant is Shake Shack
- Least revenue is coming from Five Guys Burgers and Fries
- Top three to focus to main revenue are Shake Shack, The Meatball Shop , Blue Ribbon Sushi
- Least 3 to focus on improving/study root cause are Sushi of Gari, Nobu Next Door, Five Guys Burgers and Fries

Multivariate Analysis - Questions 12

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



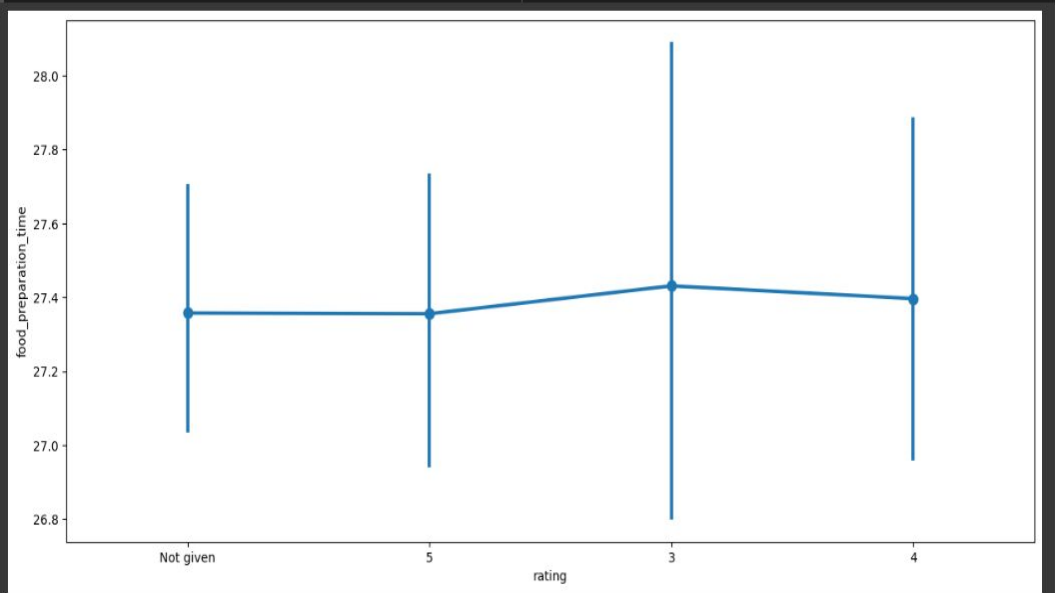

```
[14] # Replace 'Not given' with NaN
     df['rating'] = df['rating'].replace('Not given', pd.NA)

     # Convert rating column to numeric
     df['rating'] = pd.to_numeric(df['rating'])

     # Now check correlation
     correlation = df['rating'].corr(df['delivery_time'])
     print("Correlation between rating & delivery_time:", correlation)

     Correlation between rating & delivery_time: -0.009804253948165316
```

Based on rating vs delivery time

- Don’t see any strong correlation between rating and delivery time

Multivariate Analysis - Questions 12

```
#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)  ## Complete the code to visualize the relationship between rating and food preparation time us
plt.show()
```

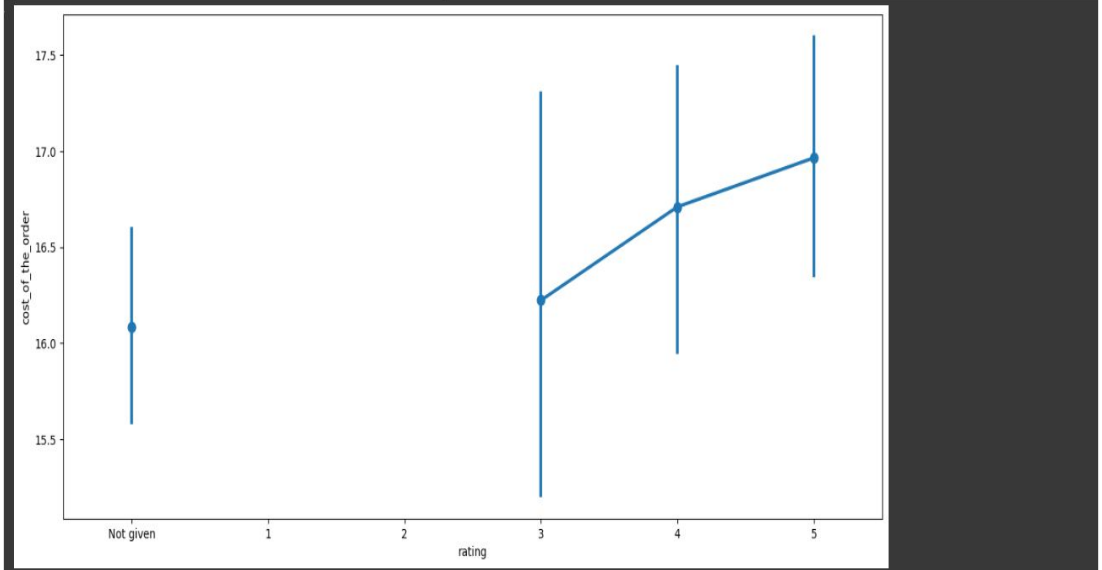


Based on Rating vs Food preparation time

- Average food preparation time is very similar across ratings:
  - Rating 3 → ~27.43 minutes
  - Rating 4 → ~27.40 minutes
  - Rating 5 → ~27.36 minutes
- There is no significant correlation between ratings and food preparation time.

Multivariate Analysis - Questions 12

```
#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,order=["Not given", 1, 2, 3, 4, 5])  ## Complete the code to visualize the relationship between rating and cost of the order using pointplot
plt.show()
```

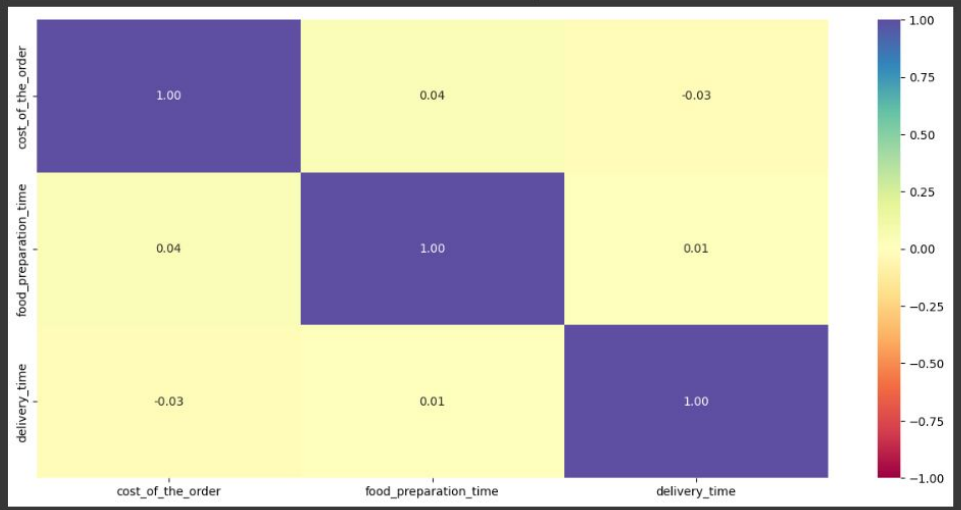


Based on Rating vs Cost of the order

- Apart from not given rating , Average order cost by rating:
  - Rating 3 → ~$16.22
  - Rating 4 → ~$16.71
  - Rating 5 → ~$16.97
- There is a slight upward trend: higher-rated orders tend to have marginally higher costs.
- However, the difference is small (only ~$0.7 between rating 3 and 5).
- This suggests that order cost has minimal influence on customer ratings — other factors (taste, service, delivery) are likely more impactful.

Multivariate Analysis - Questions 12

```
#Correlation among variables
# 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()
```



Based on Correlation among order cost vs Food prep time vs delivery time

- Cost of the order vs Food preparation time
  - 0.04 (very weak positive correlation)
- Cost of the order vs Delivery time
  - -0.03 (very weak negative correlation)
- Food preparation time vs Delivery time
  - 0.01 (almost no correlation)
- None of the variables show any meaningful correlation with each other.
- Order cost, preparation time, and delivery time are essentially independent in this dataset.
- This suggests that operational times (prep & delivery) are not strongly linked to how much an order costs.

Multivariate Analysis - Questions 13

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]

```
[32] # Write the code here
     # 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()
     print(df_rating_count.head())

     # Get the restaurant names that have rating count more than 50
     rest_names = df_rating_count[df_rating_count['rating'] >50 ]['restaurant_name'] ## Complete the code to get the restaurant names having rating count more than 50

     # 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_rating = df_mean_4.groupby(['restaurant_name'])['rating'].mean().sort_values(ascending = False).reset_index().dropna() ## Complete the code to find the mean rating

     # filter for average rating greater than 4
     df_avg_rating_greater_than_4 = df_mean_4_rating[df_mean_4_rating['rating'] > 4].sort_values(by='rating', ascending=False).reset_index(drop=True)  ## Complete the code to find restaurants with rating > 4

     df_avg_rating_greater_than_4
```

```
                restaurant_name  rating
0                   Shake Shack     133
1                The Meatball Shop      84
2                Blue Ribbon Sushi      73
3          Blue Ribbon Fried Chicken      64
4                RedFarm Broadway      41
```

	restaurant_name	rating
0	The Meatball Shop	4.511905
1	Blue Ribbon Fried Chicken	4.328125
2	Shake Shack	4.278195
3	Blue Ribbon Sushi	4.219178

Next steps: Generate code with df_avg_rating_greater_than_4 | View recommended plots | New interactive sheet

Observations:

Restaurant for promotion offer are base don criteria of rating count > 50 and avg rating > 4 are

The Meatball Shop
Blue Ribbon Fried Chicken
Shake Shack
Blue Ribbon Sushi

Multivariate Analysis - Questions 14

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]

```
# Write the code here
#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) ## Write the apprpriate column name to compute the revenue
df.head()

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

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

Multivariate Analysis - Questions 15

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]

```
# Write the code here
# 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']

## Write the code below to find the percentage of orders that have more than 60 minutes of total delivery time (see Question 9 for reference)
perc_over_60 = (df[df['total_time'] > 60].shape[0] / df.shape[0]) * 100

# Calculate percentage of orders taking more than 60 minutes
print("Percentage of orders taking more than 60 minutes:", round(perc_over_60, 2), "%")
```

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

Observations:

Percentage of orders taking more than 60 minutes: 10.54 %

Multivariate Analysis - Questions 16

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]

```
[35] # Write the code here
     # 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')


     ## Write the code below to get the mean delivery time on weekends and print it
     print('The mean delivery time on weekdays 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 weekdays is around 22 minutes
```

Observations:

Mean delivery time on weekdays ≈ 28 minutes Mean delivery time on weekends ≈ 22 minutes

APPENDIX

NOTE:

- Added my notebook as slides in appendix (19 slide after this).
- Conversion to ipynb → html → has disturbed the formatting but ideas is to attach my whole work here
-

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

In [2]:

```
# 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 [9]:

```
# uncomment and run the below code snippets if the dataset is present in the Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive In [5]:

```
from google.colab import drive drive.mount('/content/drive')
```

Mounted at /content/drive In [16]:

```
# Write your code here to read the data
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/PYF-FoodHub Data Analysis/foodhub_order.csv') In [17]:
# Write your code here to view the first 5 rows df.head()
```

Out[17]:

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

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

In [11]:

```
# Write your code here df.shape
```

Out[11]:

(1898, 9)

This is formatted as code **Observations:**

Total Rows: 1898 Total fields/columns: 9

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

In [12]:

```
# 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
```

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

In [13]:

```
# Write your code df.isnull().sum()
```

Out[13]:

Observations:

	0
# This is formatted as code Data types are float64(1), int64(4), object(4)	
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:**

No missing value found

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 [14]:

```
# Write your code here df.describe()
```

Out[14]:

	order_id	customer_id	cost_of_the_order	food_preparation_time	delivery_time
count	1.898000e+03	1898.000000	1898.000000	1898.000000	1898.000000
mean	1.477496e+06	171168.478398	16.498851	27.371970	24.161749
std	5.480497e+02	113698.139743	7.483812	4.632481	4.972637
min	1.476547e+06	1311.000000	4.470000	20.000000	15.000000
25%	1.477021e+06	77787.750000	12.080000	23.000000	20.000000
50%	1.477496e+06	128600.000000	14.140000	27.000000	25.000000
75%	1.477970e+06	270525.000000	22.297500	31.000000	28.000000
max	1.478444e+06	405334.000000	35.410000	35.000000	33.000000

Observations:

Food preparation time (once order is placed) minimum: 20 min
average: 27.37 min maximum: 35 min

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

In [15]:

 # Write the code **df['rating'].value_counts()**

Out[15]:

rating	count
Not given	736
5	588
4	386
3	188

dtype: int64

Observations:

No rated are 736

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 [16]:

df['order_id'].nunique()

Out[16]:

df['cuisine_type'].nuniqueO

Out[19]:

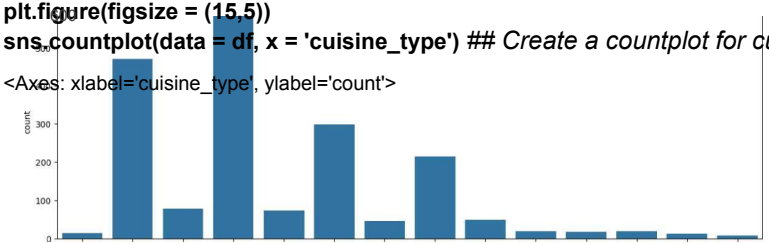
14

In [20]:

plt.figure(figsize = (15,5))

sns.countplot(data = df, x = 'cuisine_type') ## *Create a countplot for cuisine type.* **Out[20]:**

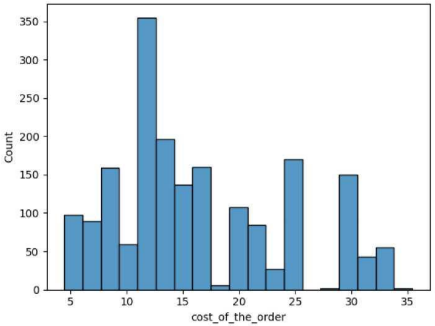
<Axes: xlabel='cuisine_type', ylabel='count'>

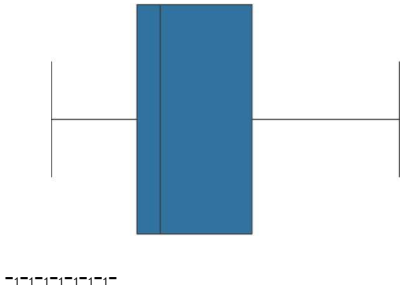


In [21]:

sns.histplot(data=df,x='cost_of_the_order') ## *Histogram for the cost of order* **plt.show()**

sns.boxplot(data=df,x='cost_of_the_order') ## *Boxplot for the cost of order* **plt.show()**





In [27]:

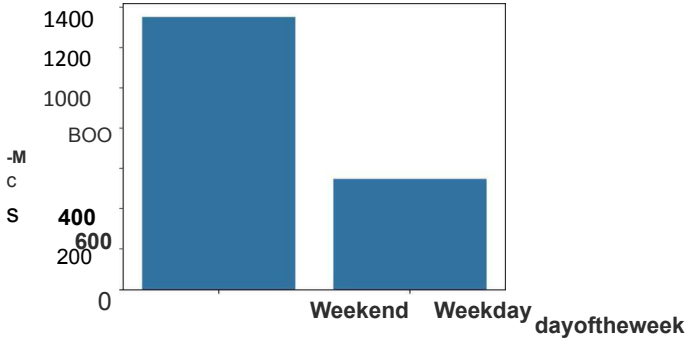
```
#    # Check the unique values
df['day_of_the_week'].nunique() ## Complete the code to check unique values for the 'day_of_the_week' column
```

Out[27]:

2

In [28]:

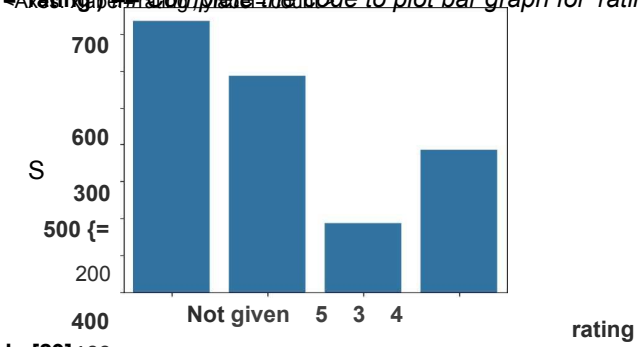
```
#    Write the code here
sns.countplot(data = df, x = 'day_of_the_week') plt.show()
```



In [24]:

 # Check the unique values

df['rating'].nunique() *## Complete the code to check unique values for the 'rating' column* **sns.countplot(data = df, x = 'rating')** *## Complete the code to plot bar graph for 'rating' column* **Out[24]:**

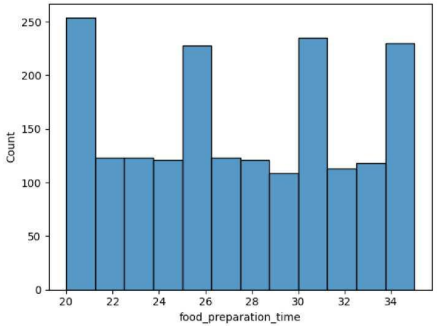


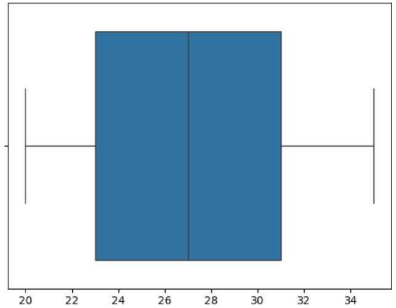
In [29]:

sns.histplot(data=df,x='food_preparation_time') *## Complete the code to plot the histogram for the cost of order* **plt.show()**

sns.boxplot(data=df,x='food_preparation_time') *## Complete the code to plot the boxplot for the cost of order* **plt.show()**

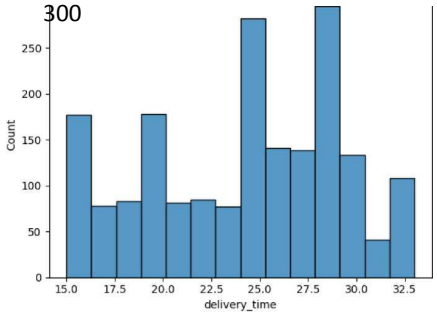
0

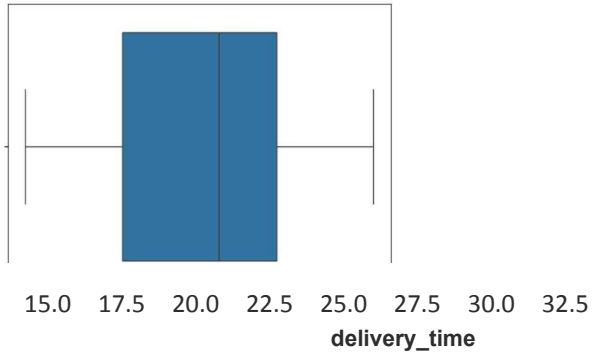




In [26]:

```
sns.histplot(data=df,x='delivery_time') ## Complete the code to plot the histogram for the delivery time plt.show()
sns.boxplot(data=df,x='delivery_time') ## Complete the code to plot the boxplot for the delivery time plt.show()
```





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

In [30]:

 # Write the code here **df['restaurant_name'].value_counts()**

Out[30]:

restaurant_name	count
Shake Shack	219
The Meatball Shop	132
Blue Ribbon Sushi	119
Blue Ribbon Fried Chicken	96
Parm	68
Rye House	1
Hiroko's Place	1
Frank Restaurant	1
Sarabeth's West	1
'wichcraft	1

178 rows x 1 columns

Observations:
Yop5 restaurant are Shake Shack
The Meatball Shop Blue Ribbon
Sushi Blue Ribbon Fried Chicken
Parm

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

In [31]:

```
# Write the code here
# Get most popular cuisine on weekends df_weekend =
df[df['day_of_the_week'] == 'Weekend']
df_weekend['cuisine_type'].value_counts() ## Complete the code to check unique values for the
```

Out[31]:

cuisine_type	count
American	415
Japanese	335
Italian	207
Chinese	163
Mexican	53
Indian	49
Middle Eastern	32
Mediterranean	32
Thai	15
French	13
Korean	11
Southern	11
Spanish	11
Vietnamese	4

dtype: int64 **Observations:**

most popular for weekend is American

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

In [32]:

```
#    Write the code here
#    Get orders that cost above 20 dollars
df_greater_than_20 = df[df['cost_of_the_order']>20] ## Write the appropriate column name to get the orders having cost above $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), '%')
```

In [35]:

```
# Write the code here
#Get the counts of each customer_id
df['customer_id'].value_counts().head(3) ## Write the appropriate column name to get the top 5 cmost frequent customers
```

Out[35]:

customer_id	
52832	13
47440	10
83287	9

dtype: int64

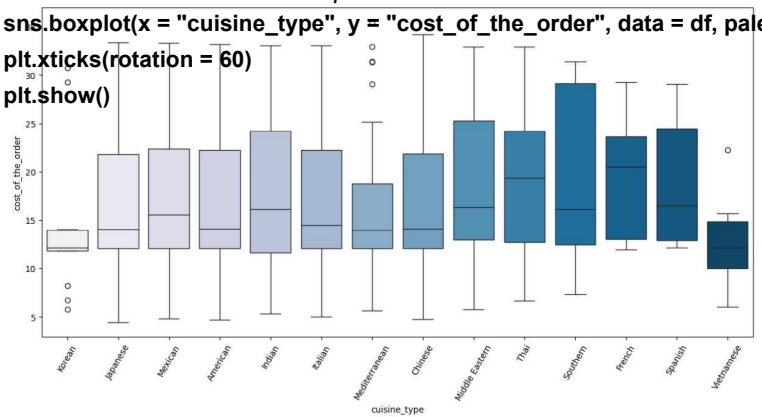
Observations: Above are top 3 customer ID's

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]

Multivariate Analysis

In [6]:

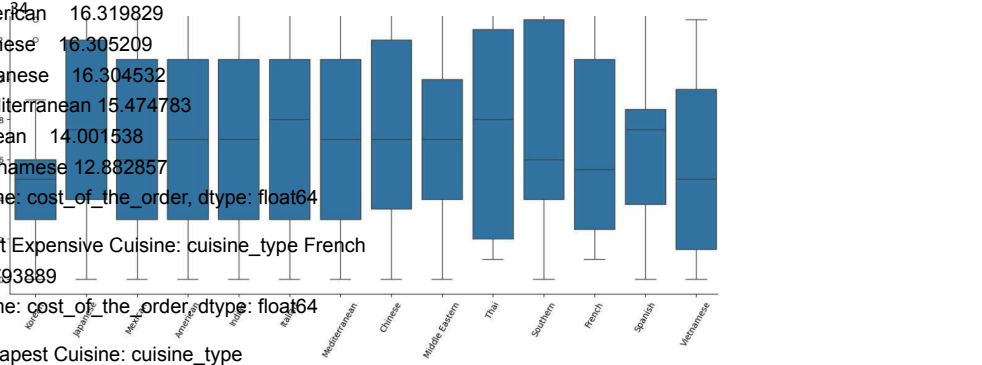
```
# Write the code# Relationship between cost of the order and cuisine type
plt.figure(figsize=(15,7))
sns.boxplot(x = "cuisine_type", y = "cost_of_the_order", data = df, palette = 'PuBu', hue = "cuisine_type")
plt.xticks(rotation = 60)
plt.show()
```



In [7]:

```
#    Group by cuisine type and calculate average order cost
avg_cost_by_cuisine = df.groupby('cuisine_type')['cost_of_the_order'].meanO-sort_values(ascending=False) printfAverage cost per

cuisineAn", avg_cost_by_cuisine)

#    Find the most expensive and cheapest cuisines most_expensive =
avg_cost_by_cuisine.head(1) cheapest = avg_cost_by_cuisine.tail(1)

print('\nMost Expensive CuisineAn", most_expensive) print('\nCheapest
CuisineAn", cheapest)
```

```
Average cost per cuisine: cuisine_type
French    19.793889
Southern    19.300588
Thai    19.207895
Spanish    18.994167
Middle Eastern 18.820612
Mexican    16.933117
Indian 16.919726
Italian    16.418691
American    16.319829
Chinese    16.305209
Japanese    16.304532
Mediterranean 15.474783
Korean    14.001538
Vietnamese 12.882857
Name: cost_of_the_order, dtype: float64

Most Expensive Cuisine: cuisine_type French
19.793889
Name: cost_of_the_order, dtype: float64

Cheapest Cuisine: cuisine_type
Vietnamese    12.882857
Name: cost_of_the_order, dtype: float64
```



In [11]:

 # Group by cuisine and calculate min, avg, max preparation time **prep_time_stats =**
df.groupby('cuisine_type')['food_preparation_time'].agg(min_time='min', avg_time='mean',
median='median', max_time='max'
).sort_values(by='median', ascending=False)
print(prep_time_stats)

cuisine_type	min_time	avg_time	median	max_time
Italian	20	27.332143	27.5	35
Thai	21	27.315789	28.0	35
Spanish	20	26.916667	27.5	35
Japanese	20	27.510638	27.5	35
Chinese	20	27.511628	27.0	35
American	20	27.440068	27.0	35
Mediterranean	20	27.000000	27.0	35
Indian	20	27.109589	27.0	35
Middle Eastern	20	26.673469	27.0	34
Southern	20	27.588235	26.0	35
Mexican	20	26.727273	26.0	35
French	21	26.888889	25.5	35
Korean	20	25.461538	25.0	33
Vietnamese	20	25.714286	25.0	33

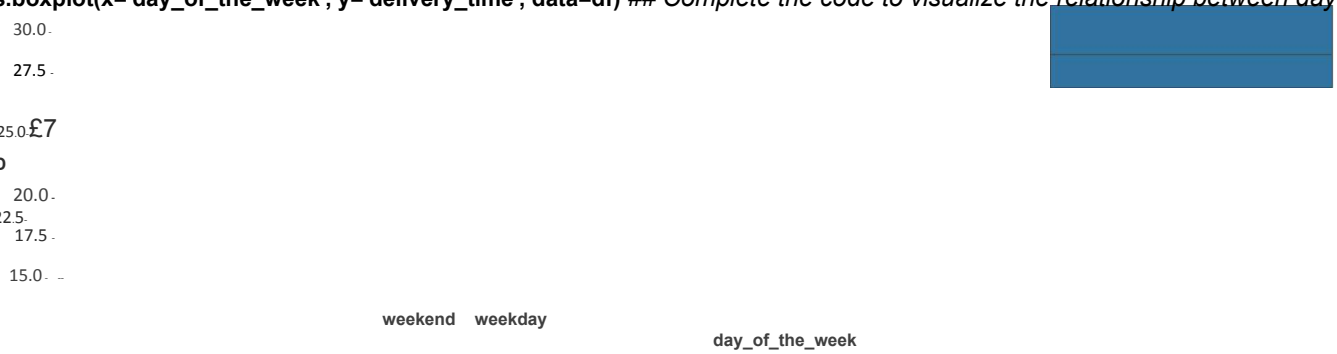
In [7]:

#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) *## Complete the code to visualize the relationship between day of the week a* **plt.show()**



In [8]:

 # Group by day_of_the_week and calculate min, avg, max delivery time **delivery_time_stats =**
df.groupby('day_of_the_week')['delivery_time'].agg(min_time='min', avg_time='mean',
max_time='max'
)

print(delivery_time_stats)

Out[9]:

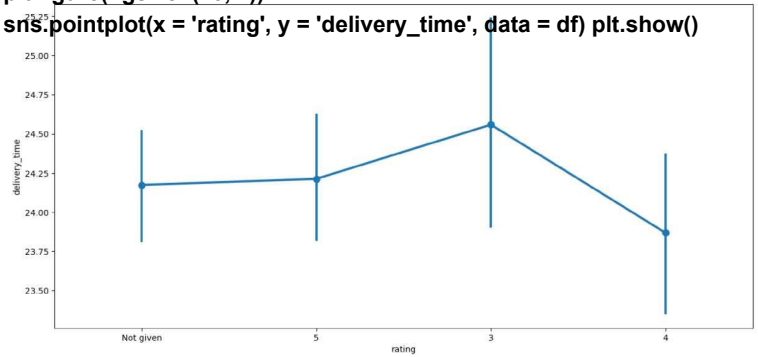
restaurant_name	cost of the order
Shake Shack	3579.53
The Meatball Shop	2145.21
Blue Ribbon Sushi	1903.95
Blue Ribbon Fried Chicken	1662.29
Parm	1112.76
RedFarm Broadway	965.13
RedFarm Hudson	921.21
TAO	834.50
Han Dynasty	755.29
Blue Ribbon Sushi Bar & Grill	666.62
Rubirosa	660.45
Sushi of Gari 46	640.87
Nobu Next Door	623.67
Five Guys Burgers and Fries	506.47

dtype: float64 In [10]:

 # Relationship between rating and delivery time

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

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



In [14]:

 # Replace 'Notgiven' with NaN

df['rating'] = df['rating'].replace('Not given', pd.NA)

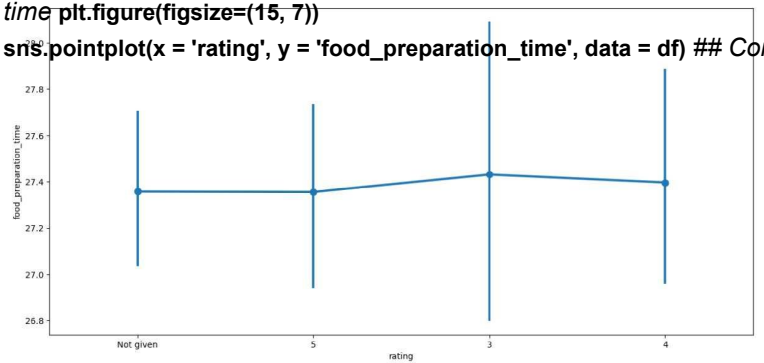
 # Convert rating column to numeric **df['rating'] = pd.to_numeric(df['rating'])**

Correlation between rating & delivery_time: -0.009804253948165316

In [19]:

#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) *## Complete the code to visualize the relationship between rating and food* **plt.show()**

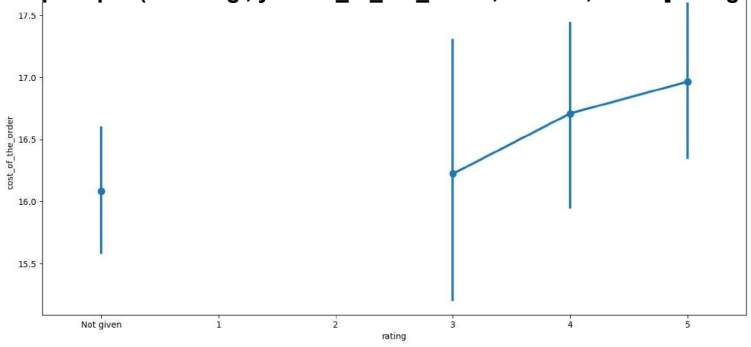


In [26]:

#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,order=["Not given", 1,2, 3, 4, 5]) *## Complete the code to visualize the relationship b* **plt.show()**

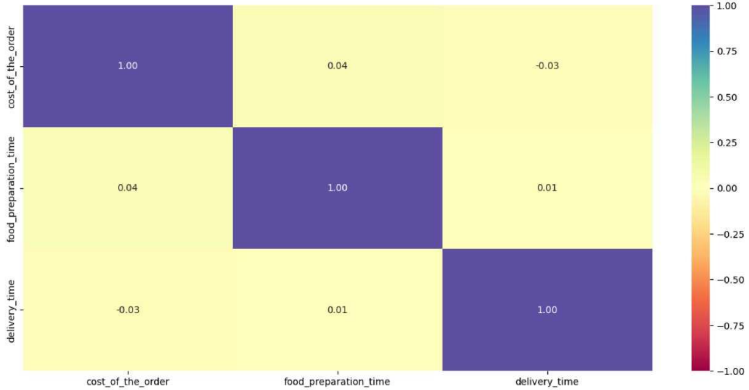


In [27]:

#Correlation among variables # 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()



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 [32]:

```
#    Write the code here
#    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).resetJndexO print(df_rating_count.head())

#    Get the restaurant names that have rating count more than 50
rest_names = df_rating_count[df_rating_count['rating'] >50 ]['restaurant_name'] ## Complete the code to get the restaurant names having r

#    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_rating = df_mean_4.groupby(['restaurant_name'])['rating'].mean().sort_values(ascending = False).resetJndex().dropnaO ## Co
```

	restaurant_name	rating
0	Shake Shack	133
1	The Meatball Shop	84
2	Blue Ribbon Sushi	73
3	Blue Ribbon Fried Chicken	64
4	RedFarm Broadway	41

Out[32]:

	restaurant_name	rating
0	The Meatball Shop	4.511905
1	Blue Ribbon Fried Chicken	4.328125
2	Shake Shack	4.278195
3	Blue Ribbon Sushi	4.219178

Observations:

Restaurant for promotion offer are base don criteria of rating count > 50 and avg rating > 4 are The Meatball Shop Blue Ribbon Fried Chicken Shake Shack Blue Ribbon Sushi

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 [33]:

```
#    Write the code here #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) ## Write the apprpriate column name to compute the revenue df.head()

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

The net revenue is around 6166.3 dollars

Observations:

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]

Percentage of orders taking more than 60 minutes: 10.54 %

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 [65]:

```
# Write the code here
# 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')

## Write the code below to get the mean delivery time on weekends and print it print('The mean delivery time on weekdays 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 weekdays is around 22 minutes

Observations:

Mean delivery time on weekdays » 28 minutes Mean delivery time on weekends » 22 minutes

Conclusion and Recommendations

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:

Great Learning

Happy Learning !