

Project Python Foundations: FoodHub Data Analysis

Context

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

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

Objective

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

Data Description

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

Data Dictionary

- `order_id`: Unique ID of the order
- `customer_id`: ID of the customer who ordered the food
- `restaurant_name`: Name of the restaurant
- `cuisine_type`: Cuisine ordered by the customer
- `cost_of_the_order`: Cost of the order

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

Let us start by importing the required libraries

```
In [1]: # Installing the libraries with the specified version.
!pip install numpy==1.25.2 pandas==1.5.3 matplotlib==3.7.1 seaborn==0.13.1 -q
```

Note:

- After running the above cell, kindly restart the runtime (for Google Colab) or notebook kernel (for Jupyter Notebook), and run all cells sequentially from the next cell.
- On executing the above line of code, you might see a warning regarding package dependencies. This error message can be ignored as the above code ensures that all necessary libraries and their dependencies are maintained to successfully execute the code in **this notebook**.

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

pd.set_option('display.float_format', lambda x: '%.2f' % x) #to display values
```

Understanding the structure of the data

```
In [3]: # 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')
```

```
In [4]: # Write your code here to read the data
data = pd.read_csv('foodhub_order.csv')
data_orig = data.copy()
```

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

Out [5]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week
0	1477147	337525	Hangawi	Korean	30.75	Weekend
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday

The dataset has been loaded properly.

- Dataset consists of several columns displaying the various attributes related to each order
- order_id column display the unique order Id for each order
- customer_id column display the ID assigned to each customer who has ordered the food
- restaurant_name and cuisine_type columns displays the name of the restaurant and the corresponding cuisine_type for which order was placed
- cost_of_the_order column provides information on the price of the order that was placed
- day_of_the_week provides details of the day when was it placed like weekend or weekday
- rating column displays the rating given by the customer for that particular order
- food_preparation_time and delivery_time columns provides information on time taken by restaurant to prepare the order and time taken by the food aggregator company to deliver the order to the customer

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

In [6]: `# Write your code here`
`data.shape`

Out [6]: (1898, 9)

Observations:

- The Dataset has 1898 number of rows with 9 columns

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

In [7]: `# Write your code here`
`data.info()`

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

Observations:

- There are 4 int64, 4 object and 1 float64 datatypes in the dataset.
- We can observe that there are no null values in the dataset.

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

```
In [8]: # Write your code here
data.isnull().sum()
```

```
Out[8]: 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:

- As per the above, there are no null values in any of the columns in the dataset. But we observed from the first 5 rows of the dataset that there are some rows that are mentioned as 'Not Given' in the 'rating' column which means that the actual values are missing or not provided or not captured. And those number can be seen from the value_counts() function

```
In [9]: data['rating'].value_counts()
```

```
Out[9]: Not given      736
5              588
4              386
3              188
Name: rating, dtype: int64
```

As there are 736 rows which do not have any ratings mentioned and this is around 40% of the total rating rows. Hence we can replace these values with NaN and then apply appropriate treatment of those missing/null values if needed. As we do not know the exact reason of these 'Not given' values, we will explore the imputation method using mean and median

```
In [10]: ## Creating a copy of the original data to perform the imputation without
## changing the actual dataset and check which method to be applied if needed
data_copy= data.copy()
data_copy_mean=data.copy()
data_copy_median=data.copy()

## Replacing 'Not given' with 'nan'
data_copy['rating']=data_copy['rating'].replace('Not given',np.nan)
data_copy_mean['rating']=data_copy_mean['rating'].replace('Not given',np.nan)
data_copy_median['rating']=data_copy_median['rating'].replace('Not given',np.nan)

## Converting the rating column type to int type for further calculations/analysis
data_copy['rating'] = pd.to_numeric(data_copy['rating'], errors='coerce')
data_copy_mean['rating'] = pd.to_numeric(data_copy_mean['rating'], errors='coerce')
data_copy_median['rating'] = pd.to_numeric(data_copy_median['rating'], errors='coerce')

## Imputing the null values with mean and median
data_copy_mean['rating']= data_copy_mean['rating'].fillna(value=data_copy_mean['rating'].mean())
data_copy_median['rating']= data_copy_median['rating'].fillna(value=data_copy_median['rating'].median())
```

The following block describes the dataframes with original, mean and median imputed rating

```
In [11]: data_copy.describe().T
```

```
Out[11]:
```

	count	mean	std	min	25%	50%	75%	max
order_id	1898.00	1477495.50	548.05	1476547.00	1477021.25	1477495.50	1477970.00	1478017.00
customer_id	1898.00	171168.48	113698.14	1311.00	77787.75	128600.00	271168.48	271168.48
cost_of_the_order	1898.00	16.50	7.48	4.47	12.08	14.14	16.50	16.50
rating	1162.00	4.34	0.74	3.00	4.00	5.00	5.00	5.00
food_preparation_time	1898.00	27.37	4.63	20.00	23.00	27.00	27.00	27.00
delivery_time	1898.00	24.16	4.97	15.00	20.00	25.00	25.00	25.00

```
In [12]: data_copy_mean.describe().T
```

Out[12]:

	count	mean	std	min	25%	50%	
order_id	1898.00	1477495.50	548.05	1476547.00	1477021.25	1477495.50	147
customer_id	1898.00	171168.48	113698.14	1311.00	77787.75	128600.00	27
cost_of_the_order	1898.00	16.50	7.48	4.47	12.08	14.14	
rating	1898.00	4.35	0.58	3.00	4.00	4.36	
food_preparation_time	1898.00	27.37	4.63	20.00	23.00	27.00	
delivery_time	1898.00	24.16	4.97	15.00	20.00	25.00	

In [13]: data_copy_median.describe().T

Out[13]:

	count	mean	std	min	25%	50%	
order_id	1898.00	1477495.50	548.05	1476547.00	1477021.25	1477495.50	147
customer_id	1898.00	171168.48	113698.14	1311.00	77787.75	128600.00	27
cost_of_the_order	1898.00	16.50	7.48	4.47	12.08	14.14	
rating	1898.00	4.47	0.67	3.00	4.00	5.00	
food_preparation_time	1898.00	27.37	4.63	20.00	23.00	27.00	
delivery_time	1898.00	24.16	4.97	15.00	20.00	25.00	

In [14]: data_copy['rating'].value_counts(dropna=False)

Out[14]:

```

NaN      736
5.00     588
4.00     386
3.00     188
Name: rating, dtype: int64

```

In [15]: data_copy_median['rating'].value_counts()

Out[15]:

```

5.00     1061
4.00      627
3.00      188
4.50       22
Name: rating, dtype: int64

```

By imputing the 'Not given' values based on mean and median, grouped by cuisine type resulted in imbalance in distribution of ratings for instance, 5 rating under original data had 588 counts while after imputing with median we have 1061 counts which shows that the customers are extremely happy/satisfied which seems to be an overestimate

While imputation with mean results in pulling down the median value from 5 to 4.36 (as illustrated in statistical summary of the original and mean imputed dataframe.

Hence for the given dataset we would not prefer to perform stastictical analysis on imputed values

Observations:

- Overall, we could observe that the dataset has 1898 rows with 9 columns with no null values present in the data.
- However from the top 5 rows of the data shows that rating column has "Not given" mentioned instead of rating. And the value_counts shows that we have around 736 rows of "Not given" in this column which is around 40%.
- As per our analysis and observation, we would not prefer to perform imputation on rating missing values

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 [16]: # Write your code here
data.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	max
order_id	1898.00	NaN	NaN	NaN	1477495.50	548.05	1476547.00	1477495.50
customer_id	1898.00	NaN	NaN	NaN	171168.48	113698.14	1311.00	711684.80
restaurant_name	1898	178	Shake Shack	219	NaN	NaN	NaN	NaN
cuisine_type	1898	14	American	584	NaN	NaN	NaN	NaN
cost_of_the_order	1898.00	NaN	NaN	NaN	16.50	7.48	4.47	165.00
day_of_the_week	1898	2	Weekend	1351	NaN	NaN	NaN	NaN
rating	1898	4	Not given	736	NaN	NaN	NaN	NaN
food_preparation_time	1898.00	NaN	NaN	NaN	27.37	4.63	20.00	35.00
delivery_time	1898.00	NaN	NaN	NaN	24.16	4.97	15.00	35.00

Observations:

The minimum it takes for food to be prepared is **20 mins** and it takes maximum of **35 mins** to prepare food. And on an average, it takes **27.37 mins** to prepare food

- The above table displays the descriptive analysis for all columns
- On an average, delivery takes around 24 mins to deliver the order once the food is prepared and the food preparation itself takes around 27 mins to be prepared once the order is placed
- Average cost of the order is 16.5 and around 50% of the orders cost less than the average.
- Shake Shack is the most popular restaurant and American is the most preferred cuisine
- Around 71% of the orders are placed on weekends

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

```
In [17]: # Write the code here
data['rating'].value_counts() ## This is used to count the no of unique values
```

```
Out[17]: Not given    736
         5          588
         4          386
         3          188
         Name: rating, dtype: int64
```

Observations:

- As we can observe that there are **736** orders that are not rated by the customers

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]

Order_ID

```
In [18]: # check unique order ID
data['order_id'].nunique()
```

```
Out[18]: 1898
```

Customer_ID

```
In [19]: # check unique customer ID
data['customer_id'].nunique()
```

```
Out[19]: 1200
```

This shows that there are repiting customers who have ordered more than once from this food aggregator company app online

```
In [20]: data['customer_id'].value_counts().head(10)
```

```
Out[20]: 52832    13
         47440    10
         83287     9
         250494     8
         259341     7
         82041     7
         65009     7
         276192     7
         97079     6
         97991     6
         Name: customer_id, dtype: int64
```

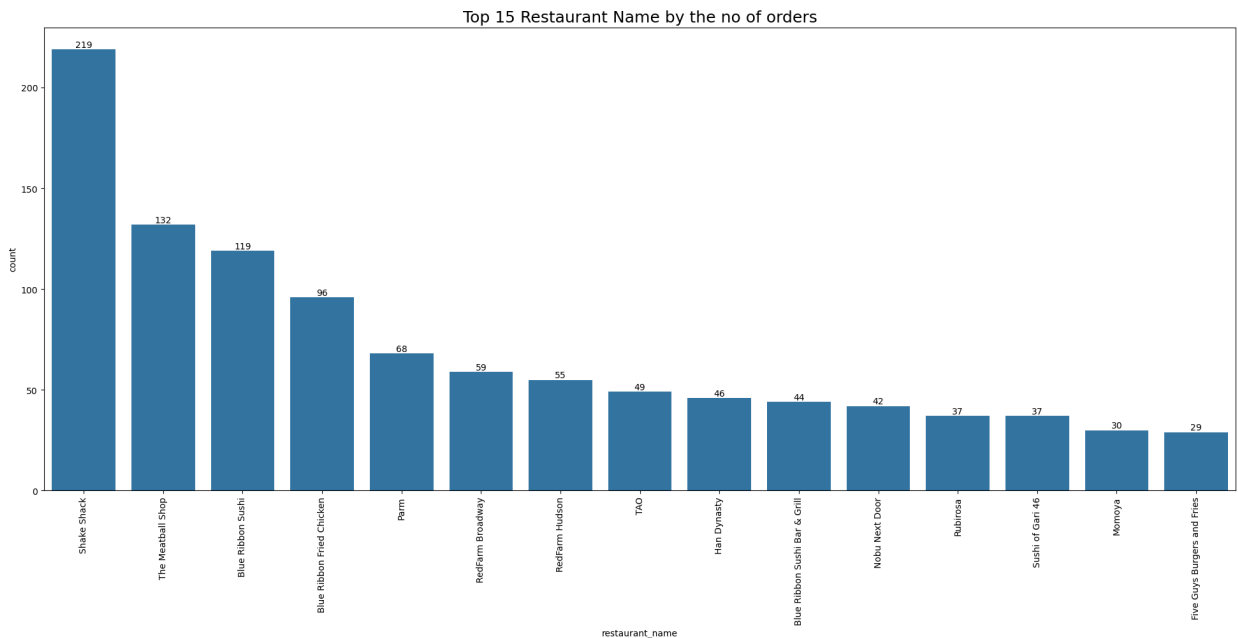
This shows that the customers with id **'52832'** and **'47440'** are top 2 repeat customers which shows some good engagement

Lets explore the categorical variables

```
In [21]: # check unique restaurant name
data['restaurant_name'].nunique()
```

Out[21]: 178

```
In [22]: plt.figure(figsize=(20,8))
plot = sns.countplot(data = data, x='restaurant_name',order=data['restaurant_name'].nlargest(15,'orders'))
plt.tight_layout()
for container in plot.containers:
    plot.bar_label(container)
plt.xticks(rotation=90);
plt.title('Top 15 Restaurant Name by the no of orders',fontsize = 18);
```



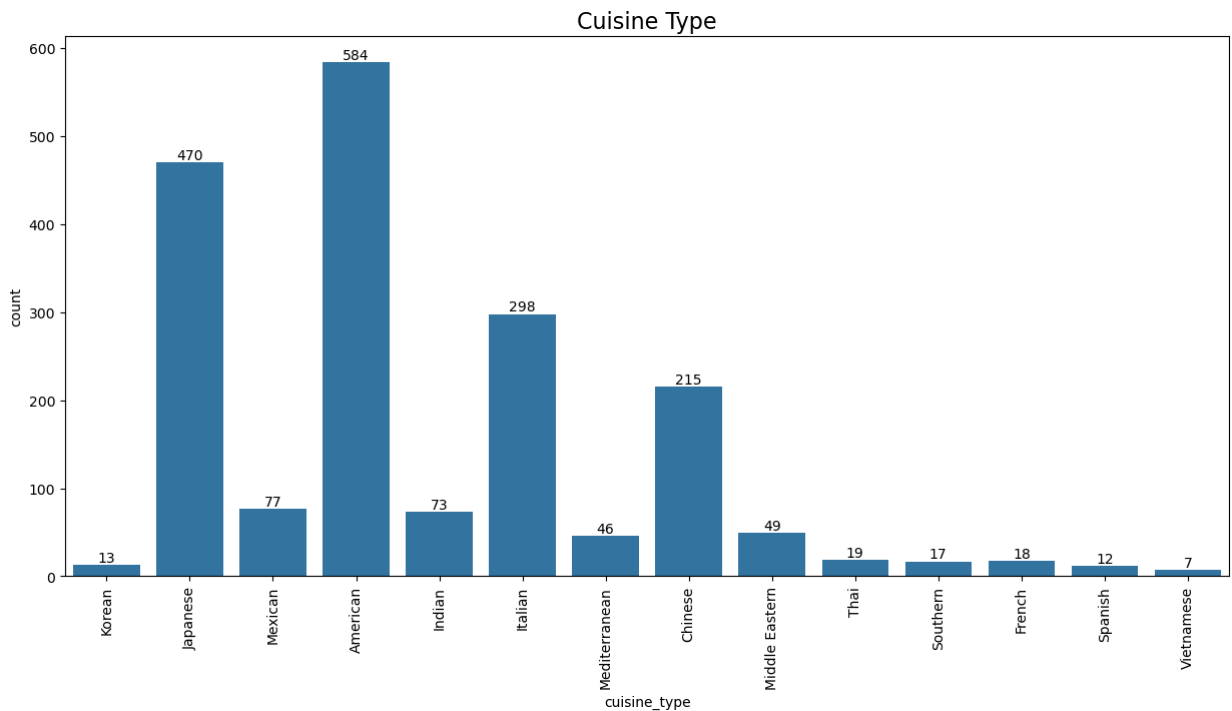
- As there are 178 restaurants to be analysed in a graph which is not easy to interpret, hence we displayed the top 15 restaurants as per the no of orders. And we observe that **'Shake Shack'** is the top restaurant which has received the highest no of orders followed by **'The Meatball Shop'**

```
In [23]: # check unique cuisine type
data['cuisine_type'].nunique()
```

Out[23]: 14

```
In [24]: plt.figure(figsize=(15,7))
plot = sns.countplot(data = data, x='cuisine_type');
for container in plot.containers:
    plot.bar_label(container)
```

```
plt.xticks(rotation=90);
plt.title('Cuisine Type',fontsize = 16);
```

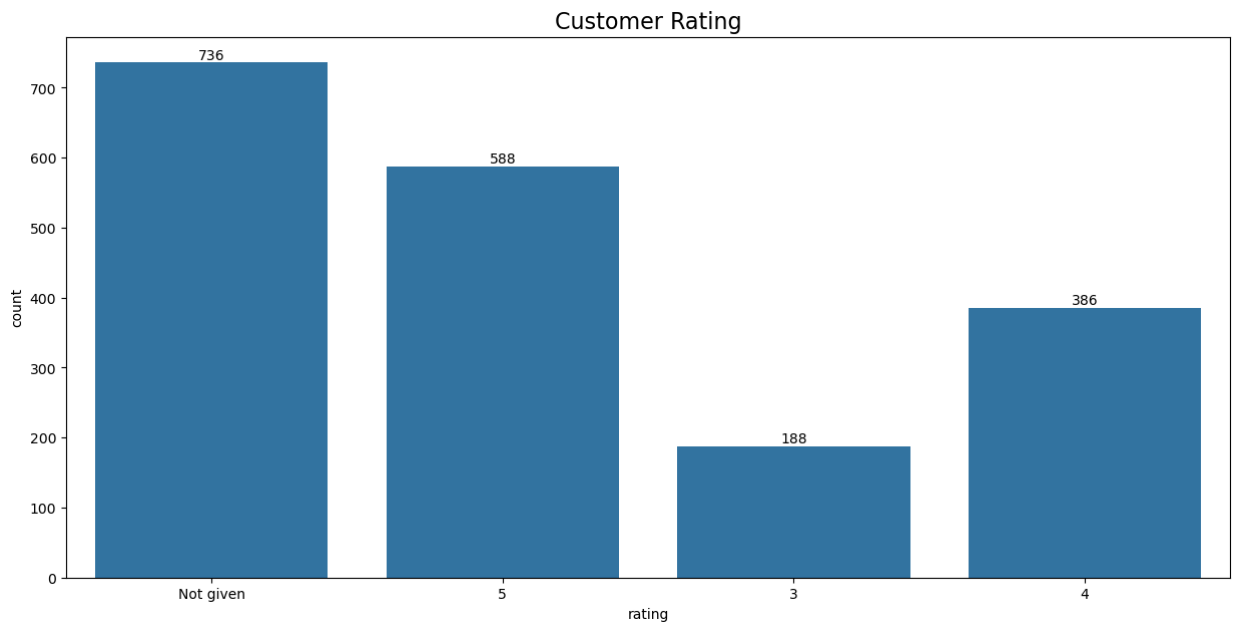


The top 3 cuisine types are **American, Japanese and Italian**. American being the most popular of all. The bottom 3 cuisine types are **Vietnamese, Spanish and Korean**.

```
In [25]: # check unique rating
data['rating'].unique()
```

Out[25]: 4

```
In [26]: plt.figure(figsize=(15,7))
plot = sns.countplot(data = data, x='rating');
for container in plot.containers:
    plot.bar_label(container)
plt.title('Customer Rating',fontsize = 16);
```

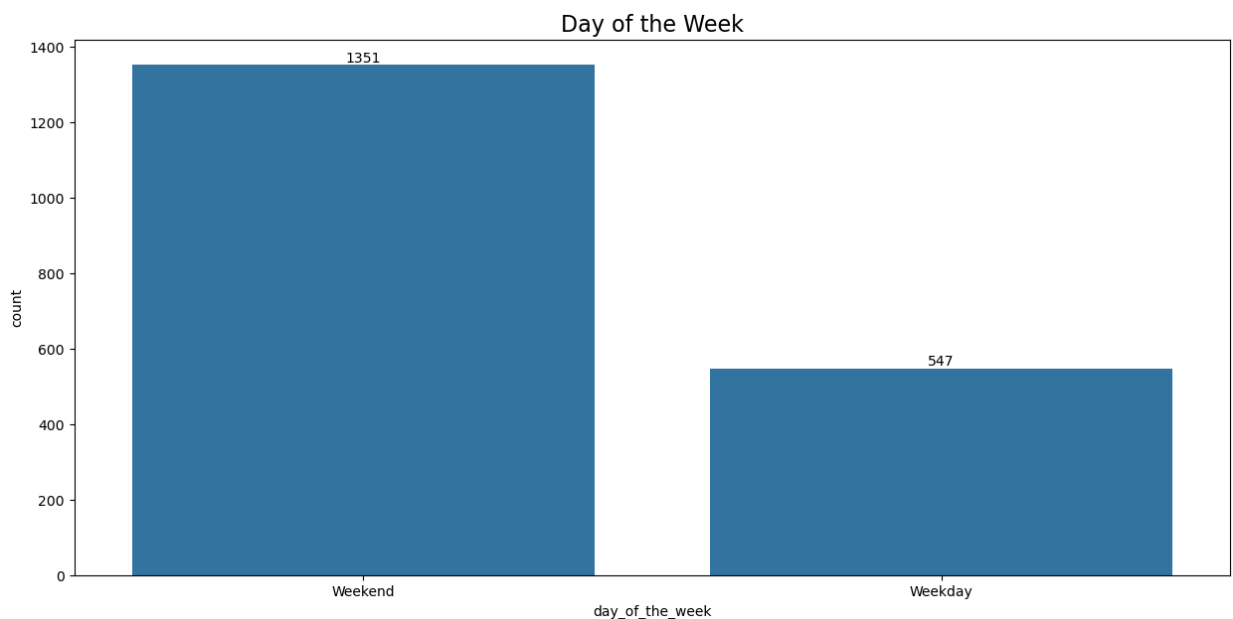


- We observe that there are no ratings less than 3 and most of the orders that are rated are highly rated (4 and 5) which shows that customers who have given ratings are satisfied and happy

```
In [27]: # check unique values for the day of the week  
data['day_of_the_week'].nunique()
```

Out[27]: 2

```
In [28]: plt.figure(figsize=(15,7))  
plot = sns.countplot(data = data, x='day_of_the_week');  
for container in plot.containers:  
    plot.bar_label(container)  
plt.title('Day of the Week', fontsize = 16);
```

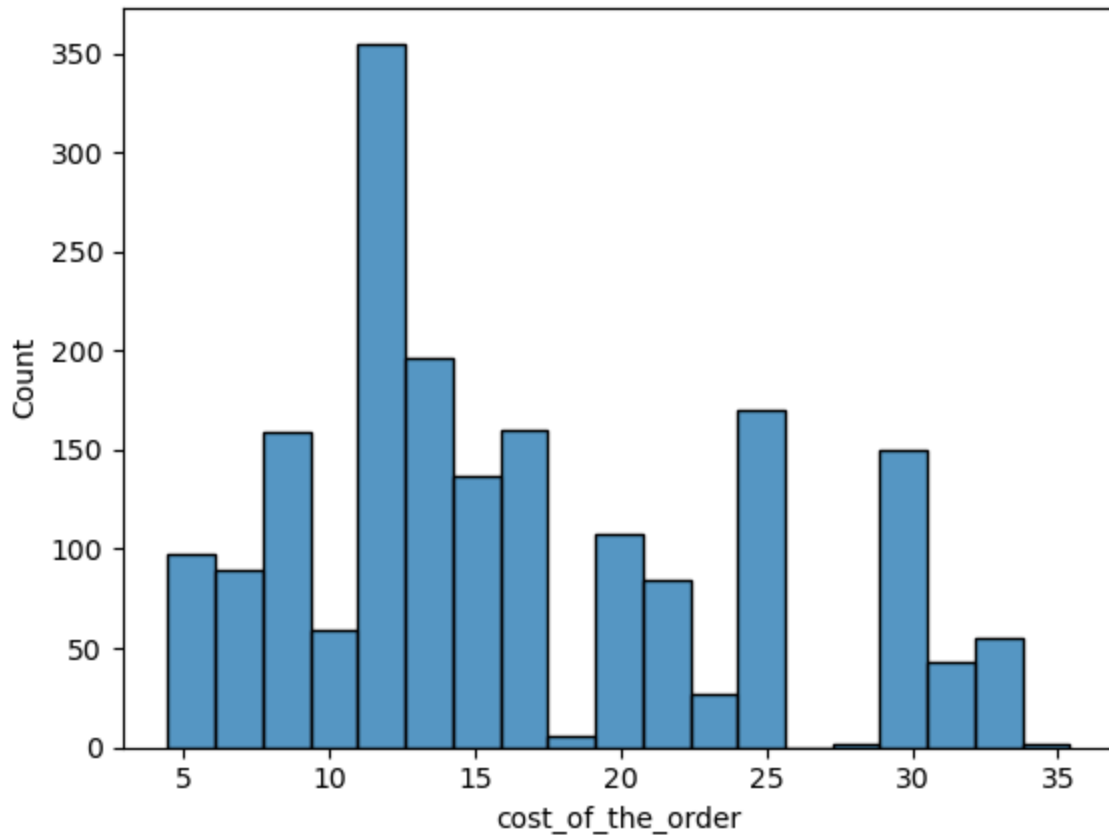


- We can observe that more orders are placed on weekend as compared to Weekday. They are more than double of the orders that are placed on weekday.

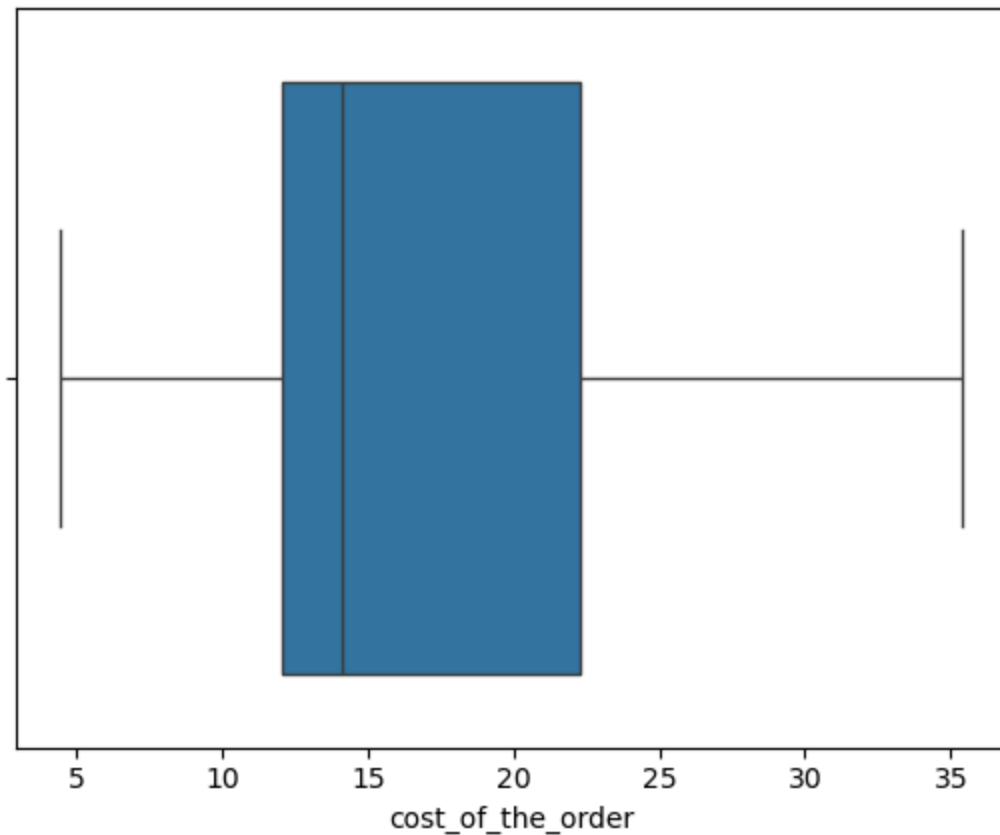
Lets explore numerical variables

Cost of the Order

```
In [29]: # Write the code here  
sns.histplot(data = data, x='cost_of_the_order');
```



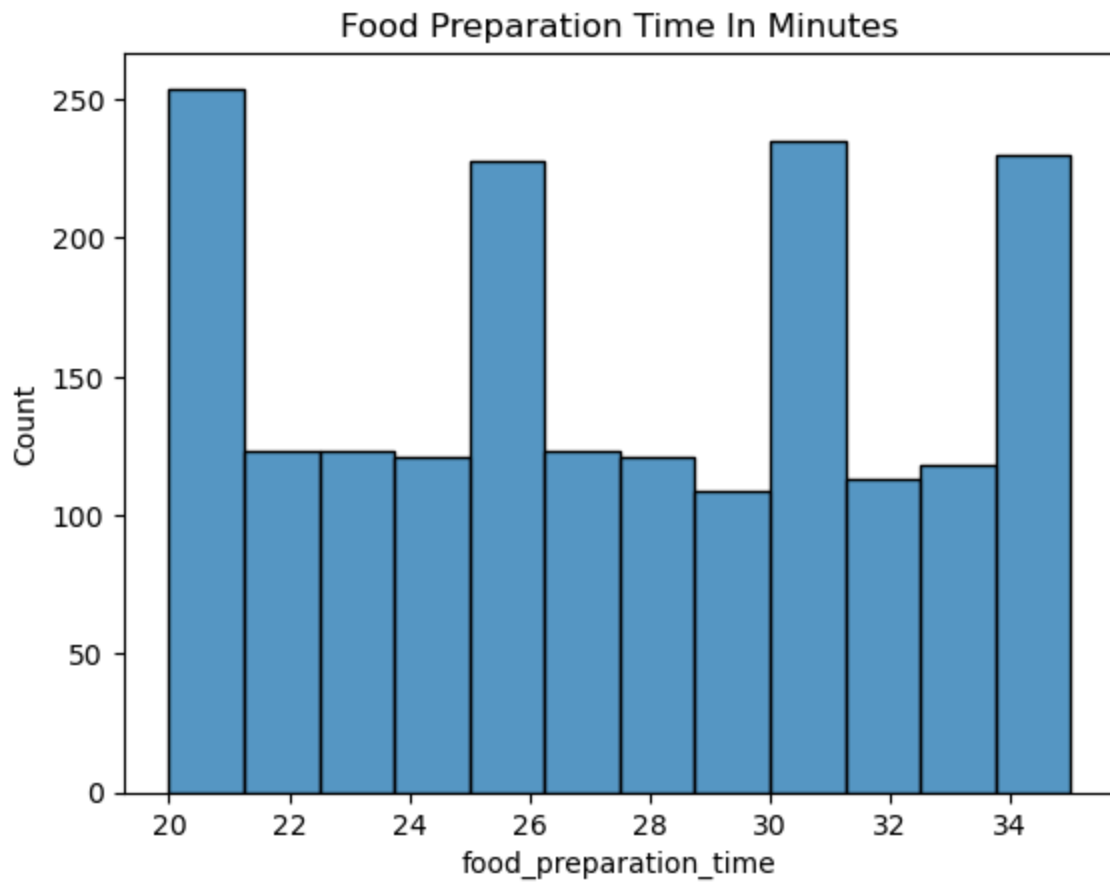
```
In [30]: sns.boxplot(data = data, x='cost_of_the_order');
```



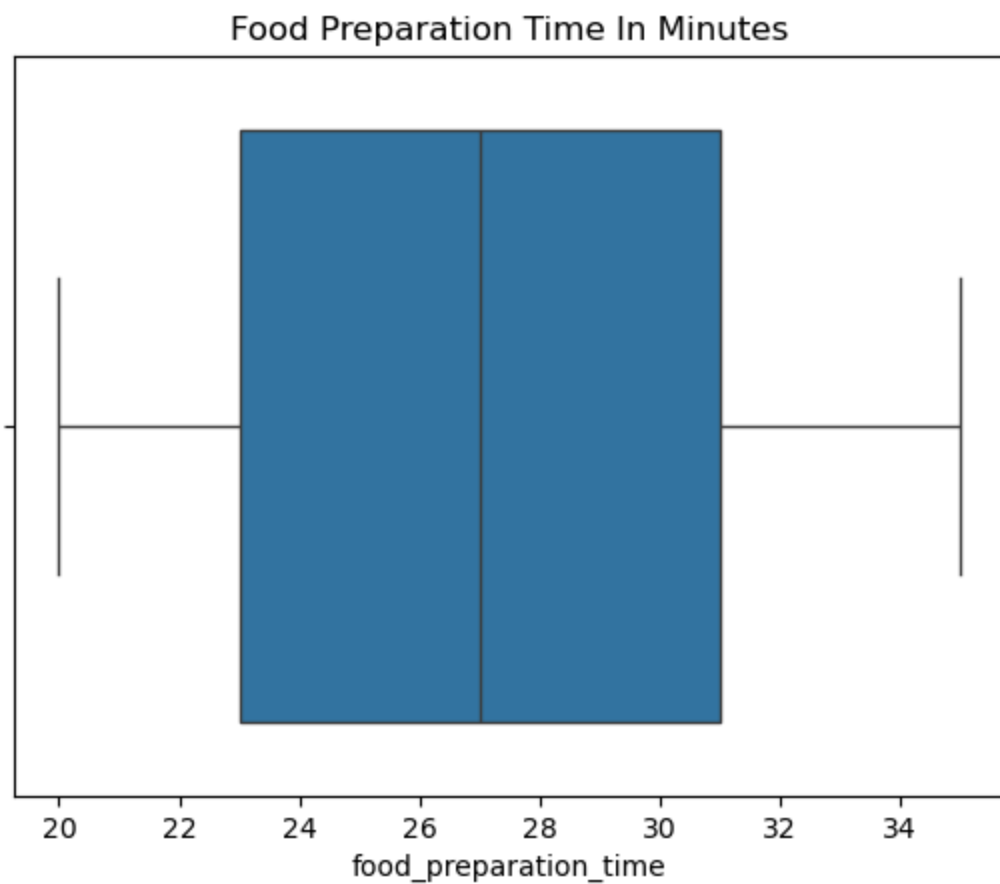
- From the histplot, we can see that there are multiple peaks which shows that there are different pricing groups of orders
- The most common order cost appears to be around 10-12.
- The order costs range approximately from 4 to 35
- As per the boxplot, we can observe that the cost of the order distribution is right skewed.
- The median cost of the order is around 15, meaning half of the orders cost less than or equal to 15, and half cost more.
- The box (from Q1 to Q3) represents the middle 50% of the data, roughly ranging from 13 to 23.
- There appear to be no outliers in this plot, as there are no individual points beyond the whiskers.
- IQR of the plot = $Q3 - Q1 = 10.22$

Food Preparation Time

```
In [31]: sns.histplot(data = data, x ='food_preparation_time');  
plt.title('Food Preparation Time In Minutes');
```



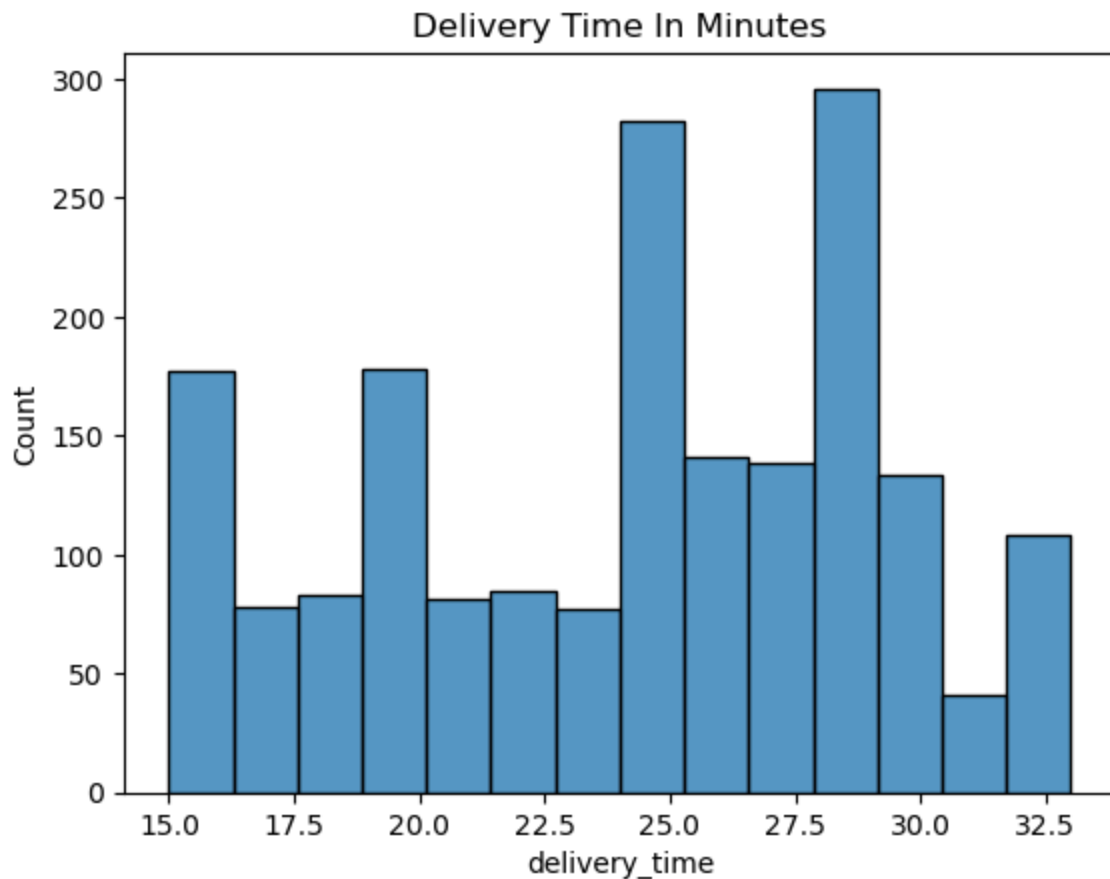
```
In [32]: sns.boxplot(data = data, x = 'food_preparation_time');  
plt.title('Food Preparation Time In Minutes');
```



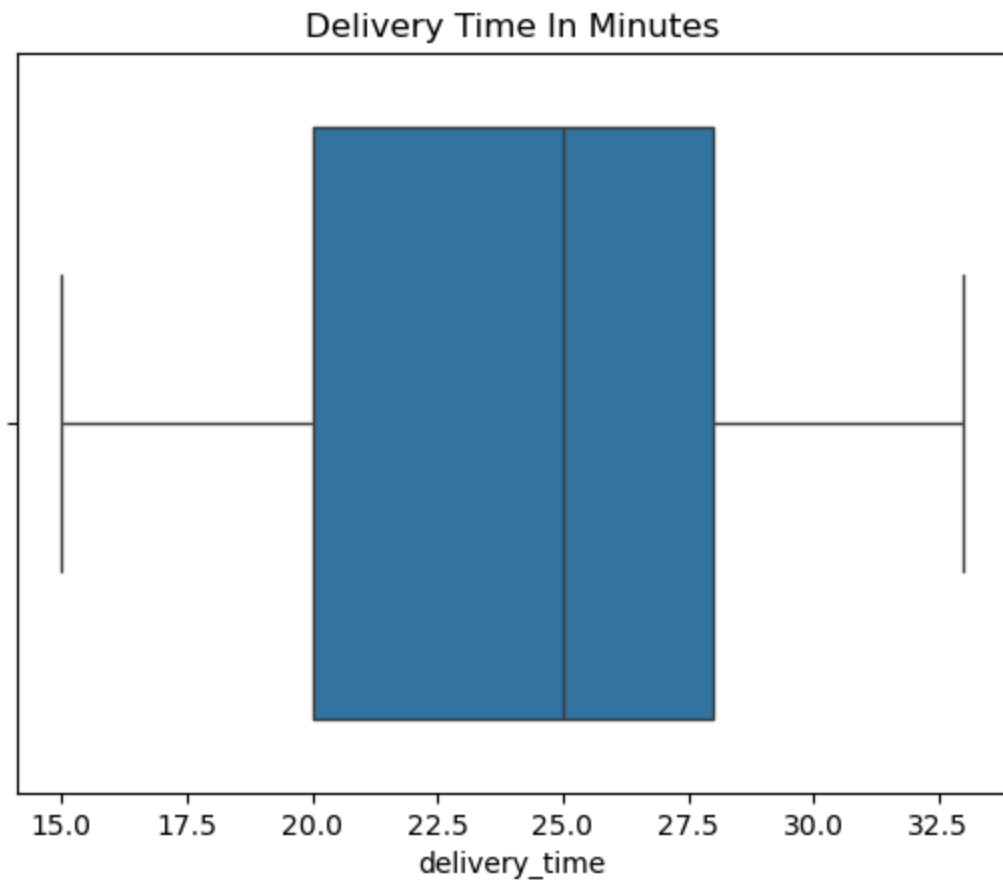
- The food_preparation_time seems to be of uniform distribution typically ranging from 20 to 35 mins.
- From the histplot, we can see that there are higher frequencies for some of the values of the food preparation time
- From the boxplot, it seems there are no outliers present in this data

Delivery Time

```
In [33]: sns.histplot(data=data, x='delivery_time');  
plt.title('Delivery Time In Minutes');
```



```
In [34]: sns.boxplot(data=data, x='delivery_time');  
plt.title('Delivery Time In Minutes');
```



- The distribution seems to be of slightly left skewed distribution
- The overall range of delivery times is approximately from 15.0 minutes to 32.5 minutes.
- The histogram shows that the most frequent delivery times occur between 27.5 and 30.0, with a particularly high count around the 27.5 to 28.75 minute mark. Another frequent range of delivery times is between 24.0 and 26.5 minutes. The least frequent delivery times are between 30.0 and 31.25 minutes.
- The median is at 25 mins which says that 50% of the deliveries take less than or equal to 25 mins
- As per the boxplot, there are no outliers present in the data for Delivery time
- The box represents the interquartile range (IQR), which contains the middle 50% of the delivery times which ranges from approximately 20.0 minutes (Q1) to about 28.0 minutes (Q3).

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

```
In [35]: # Write the code here
data['restaurant_name'].value_counts().head(5)
```

```
Out[35]: Shake Shack      219
The Meatball Shop    132
Blue Ribbon Sushi    119
Blue Ribbon Fried Chicken  96
Parm                  68
Name: restaurant_name, dtype: int64
```

Observations:

- The top restaurant that has received the maximum number of orders received is **Shake Shack** followed by **The Meatball Shop, Blue Ribbon Sushi, Blue Ribbon Fried Chicken and Parm**

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

```
In [36]: ## This gives all the unique values of cuisine type with their counts ordered
data[data['day_of_the_week']=='Weekend']['cuisine_type'].value_counts().head(5)
```

```
Out[36]: American      415
Japanese      335
Italian       207
Chinese       163
Mexican        53
Name: cuisine_type, dtype: int64
```

```
In [37]: ## This will give directly the most ordered cuisine type on weekend
data.loc[data['day_of_the_week']=='Weekend', 'cuisine_type'].mode()[0]
```

```
Out[37]: 'American'
```

Observations:

American is the most popular cuisine on the weekends that has been ordered the most

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

```
In [38]: # Write the code here
## This is an one liner code for calculating the percentage of the orders that
(((data['cost_of_the_order']>20).sum()/len(data))*100).round(2)
```

```
Out[38]: 29.24
```

```
In [39]: # Get orders that cost above 20 dollars
data_greater_than_20 = data[data['cost_of_the_order']>20]

# Calculate percentage of such orders in the dataset
percentage = (data_greater_than_20.shape[0] / data.shape[0]) * 100

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

Percentage of orders above 20 dollars is 29.24 %

Observations:

The percentage of orders received that cost more than 20 dollars is **29.24%**

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

```
In [40]: # Write the code here
print('The mean delivery time for this dataset is',(data['delivery_time'].mean)
```

The mean delivery time for this dataset is 24.16 minutes

Observations:

On an average an order takes around **24 mins** to be delivered to the customer

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

```
In [41]: # Write the code here
data['customer_id'].value_counts().head(3)
```

```
Out[41]: 52832    13
         47440    10
         83287     9
         Name: customer_id, dtype: int64
```

Observations:

- The top customer with id '**52832**' has ordered 13 times followed by the second frequent customer with id '**47440**' ordering 10 times . The third frequent customer is customer Id '**83287**' who had ordered 9 times through the app online

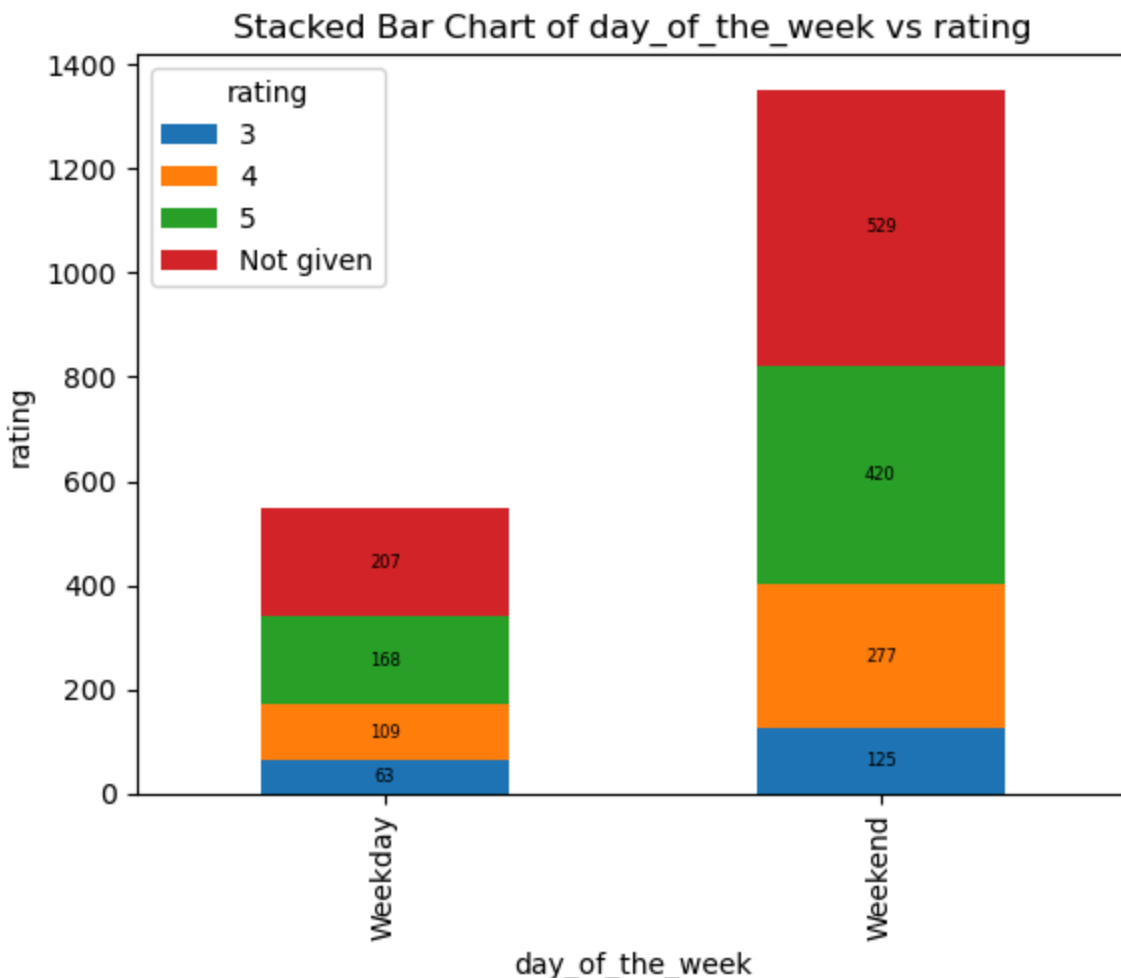
Multivariate Analysis

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

- We are going to skip the order id and customer id from the multivariate analysis as those do not add much essence to the analysis and also do not have any kind of pattern or category in them
- Similarly for the case of restaurant, we are going to ignore as the no of datapoints are large to be displayed and interpreted in a graph

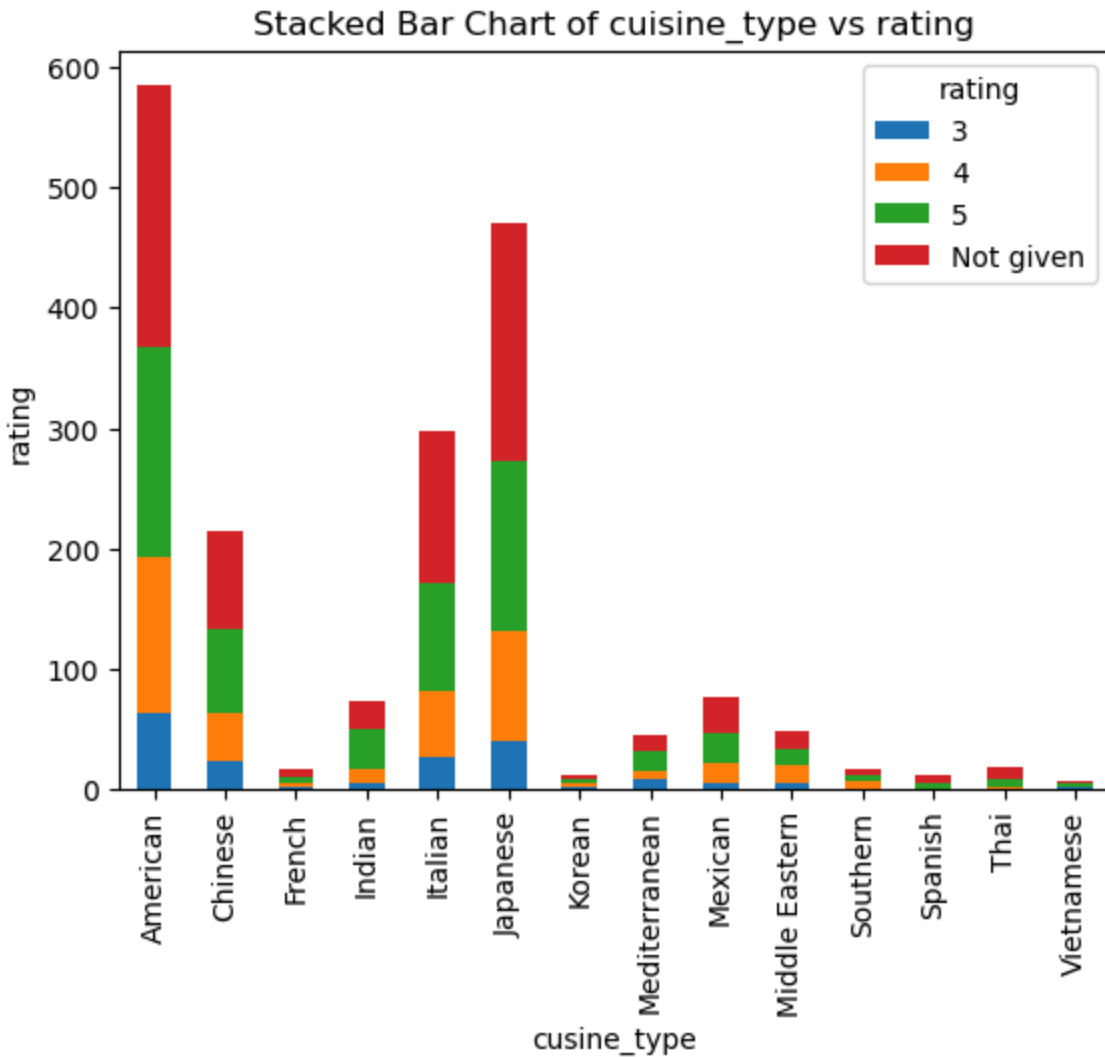
Lets explore the relationship between various categorical-categorical variables

```
In [42]: crosstab = pd.crosstab(data['day_of_the_week'], data['rating'])
plot = crosstab.plot(kind='bar', stacked=True)
plt.xlabel('day_of_the_week')
plt.ylabel('rating')
plt.title('Stacked Bar Chart of day_of_the_week vs rating')
plt.legend(title='rating')
for bar in plot.containers:
    plot.bar_label(bar, label_type='center', fontsize = 6)
plt.show()
```



- This shows that the ratings distribution percentage is similar on orders placed both on weekends or weekdays

```
In [43]: crosstab = pd.crosstab(data['cuisine_type'], data['rating'])
plot = crosstab.plot(kind='bar', stacked=True)
plt.xlabel('cuisine_type')
plt.ylabel('rating')
plt.title('Stacked Bar Chart of cuisine_type vs rating')
plt.legend(title='rating')
plt.figure(figsize=(20,8))
plt.show()
```



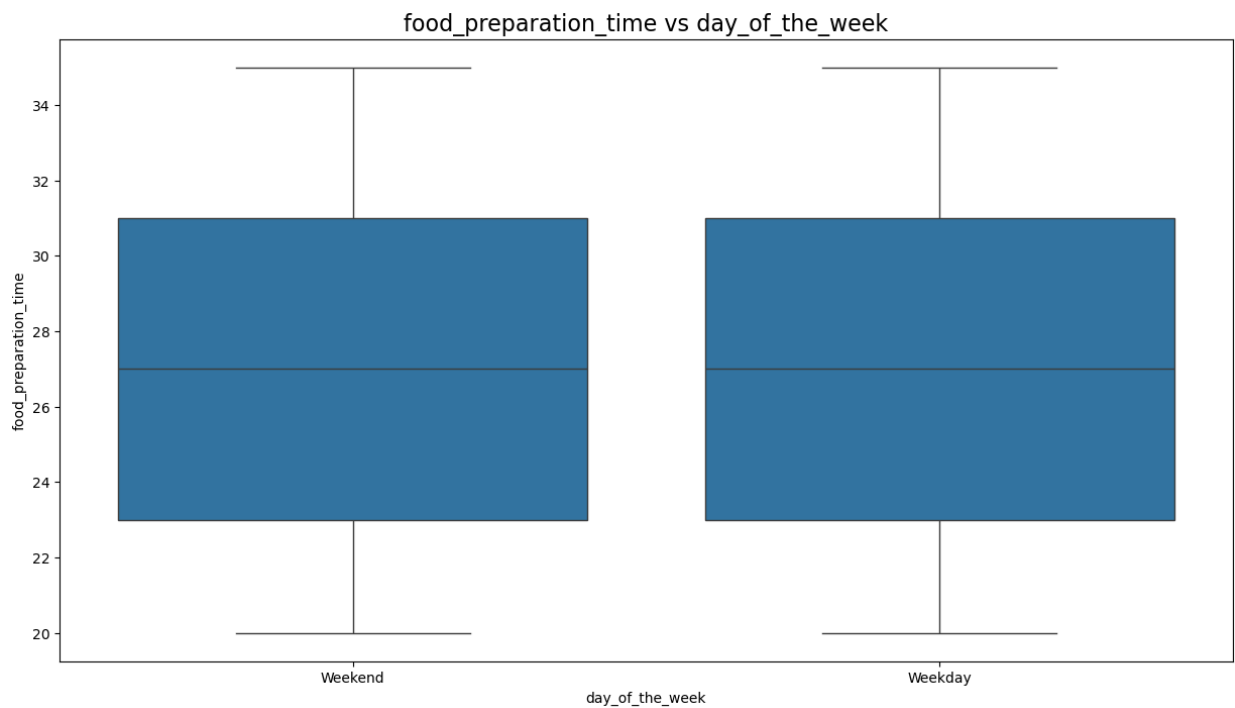
<Figure size 2000x800 with 0 Axes>

- Orders are not rated across all the cuisine types but in various proportions like Japanese and Italian have around 42% of their orders not rated
- We can observe that the orders for Southern, Spanish, Thai, Vietnamese, French cuisines got good rating. And overall also, 3 ratings are less across all cuisine types

Lets explore the relationship between categorical and numerical variables

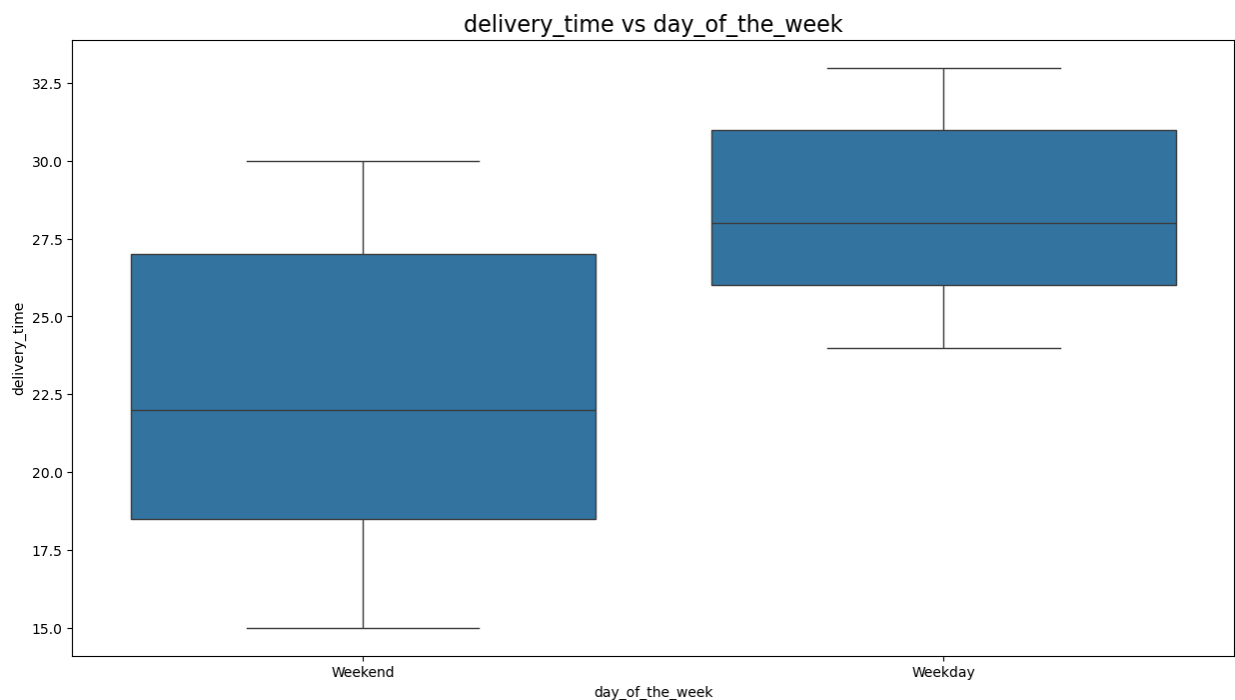
```
In [44]: # Write the code here
plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='day_of_the_week', y='food_preparation_time')
plt.title('food_preparation_time vs day_of_the_week',fontsize = 16)
```

```
Out[44]: Text(0.5, 1.0, 'food_preparation_time vs day_of_the_week')
```



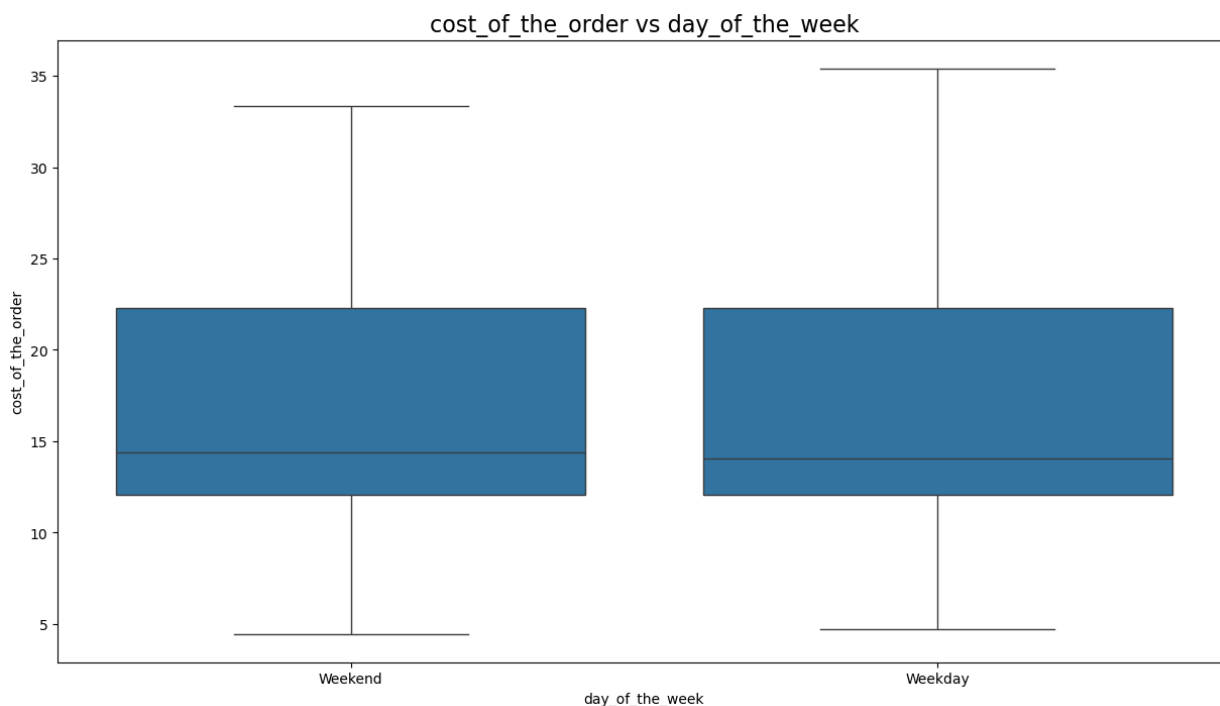
- Similar to the rating , food preparation time also doesnt differ much whether it is an order placed and prepared on a weekday or weekend
- This also shows that despite the higher no of orders placed on weekends, the preparation time is maintained across both weekends and weekdays on an overall basis

```
In [45]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='day_of_the_week', y='delivery_time')
plt.title('delivery_time vs day_of_the_week',fontsize = 16)
plt.show()
```



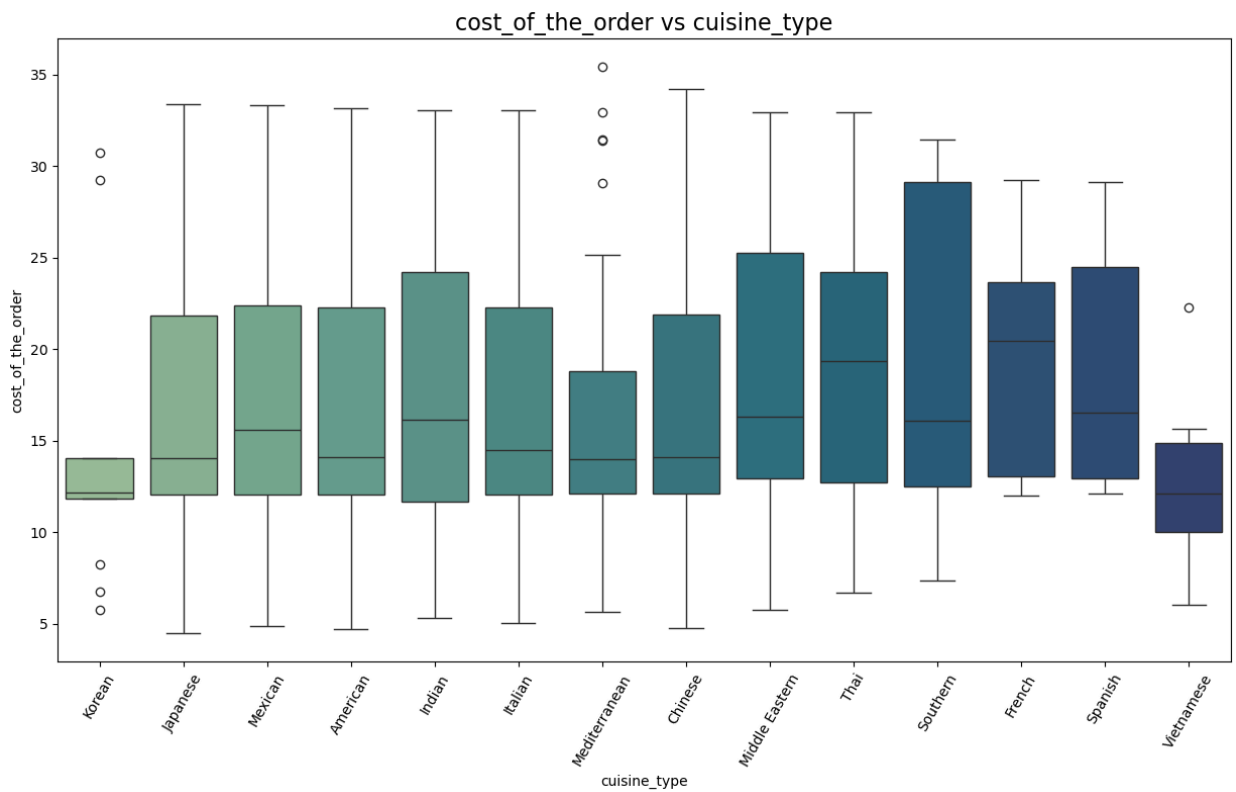
- This boxplot shows that the delivery time takes lesser time on weekends rather than on weekdays. Though there are not many orders on weekdays as compared to weekends still the delivery time is high ,probably due to high traffic on weekdays or other factors.
- 50% of the deliveries takes less than or around 22 mins to get delivered on weekends whereas the same is around 28 mins for weekdays
- And also the maximum time taken for delivery for order is 33 mins for weekday vs 30 mins for weekend

```
In [46]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='day_of_the_week', y='cost_of_the_order')
plt.title('cost_of_the_order vs day_of_the_week',fontsize = 16)
plt.show()
```

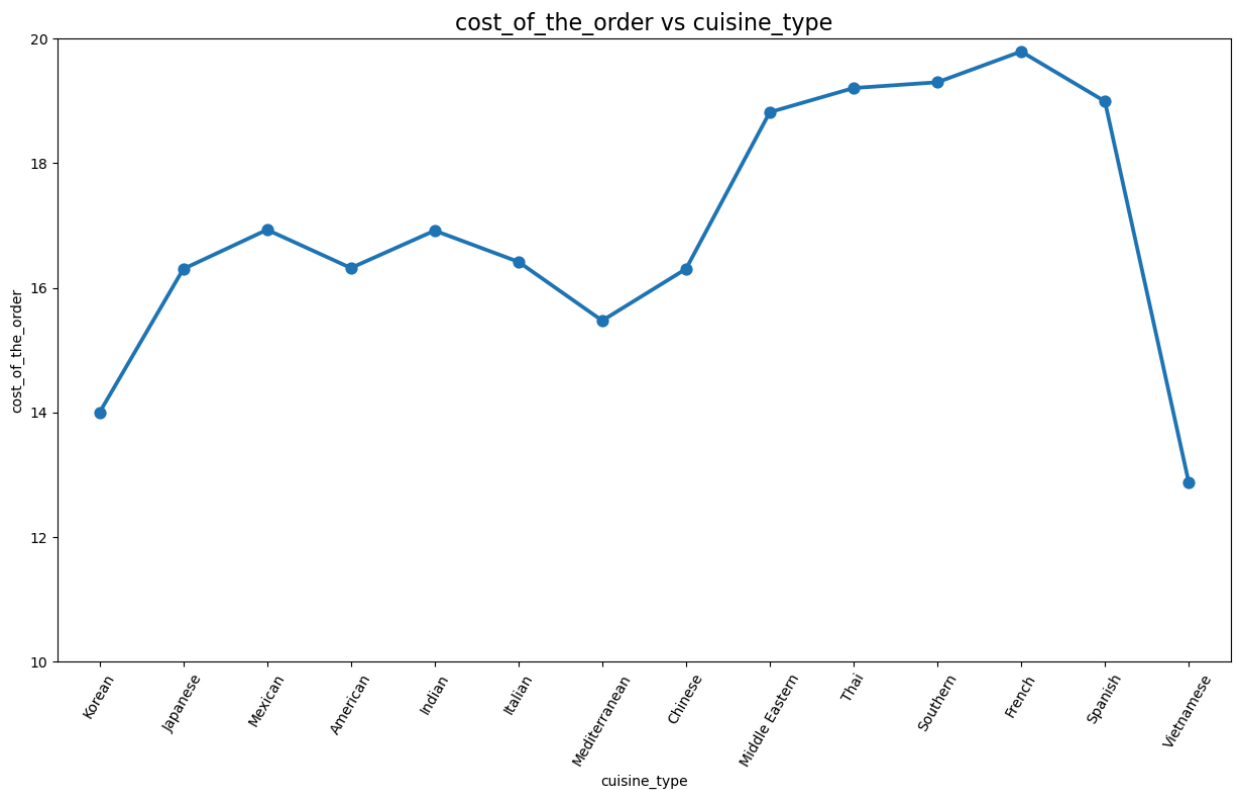


- The distribution of the order cost is quite similar for both weekends and weekdays and both of them are right skewed. However the median is slightly lower for weekday as compared to weekend which shows that 50% of the orders placed on weekdays are of slightly lesser price than that of weekends
- The most expensive order is placed on weekday.

```
In [47]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='cuisine_type', y='cost_of_the_order',palette = 'crest')
plt.title('cost_of_the_order vs cuisine_type',fontsize = 16)
plt.xticks(rotation=60);
plt.show()
```

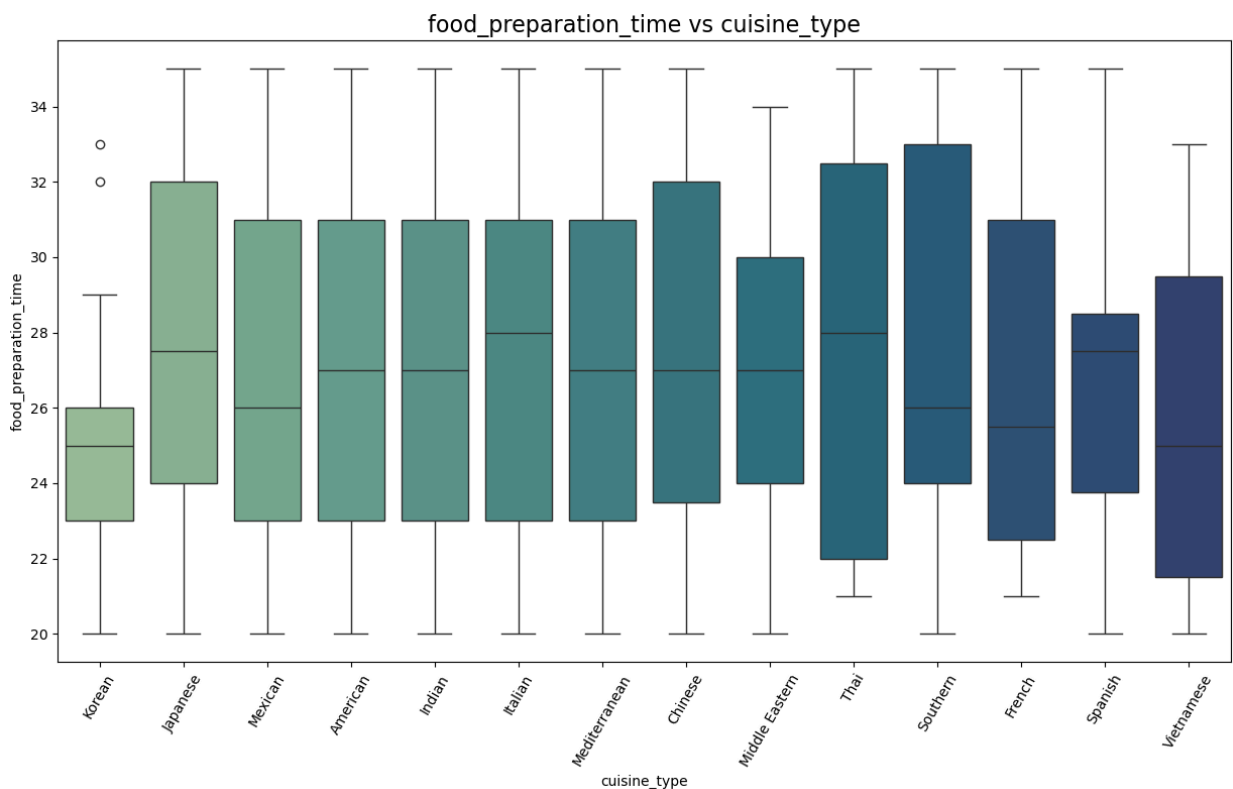


```
In [48]: plt.figure(figsize=(15,8))
sns.pointplot(data=data, x='cuisine_type', y='cost_of_the_order', errorbar=('c
plt.xticks(rotation=60);
plt.title('cost_of_the_order vs cuisine_type', fontsize = 16)
plt.ylim(10,20)
plt.show()
```

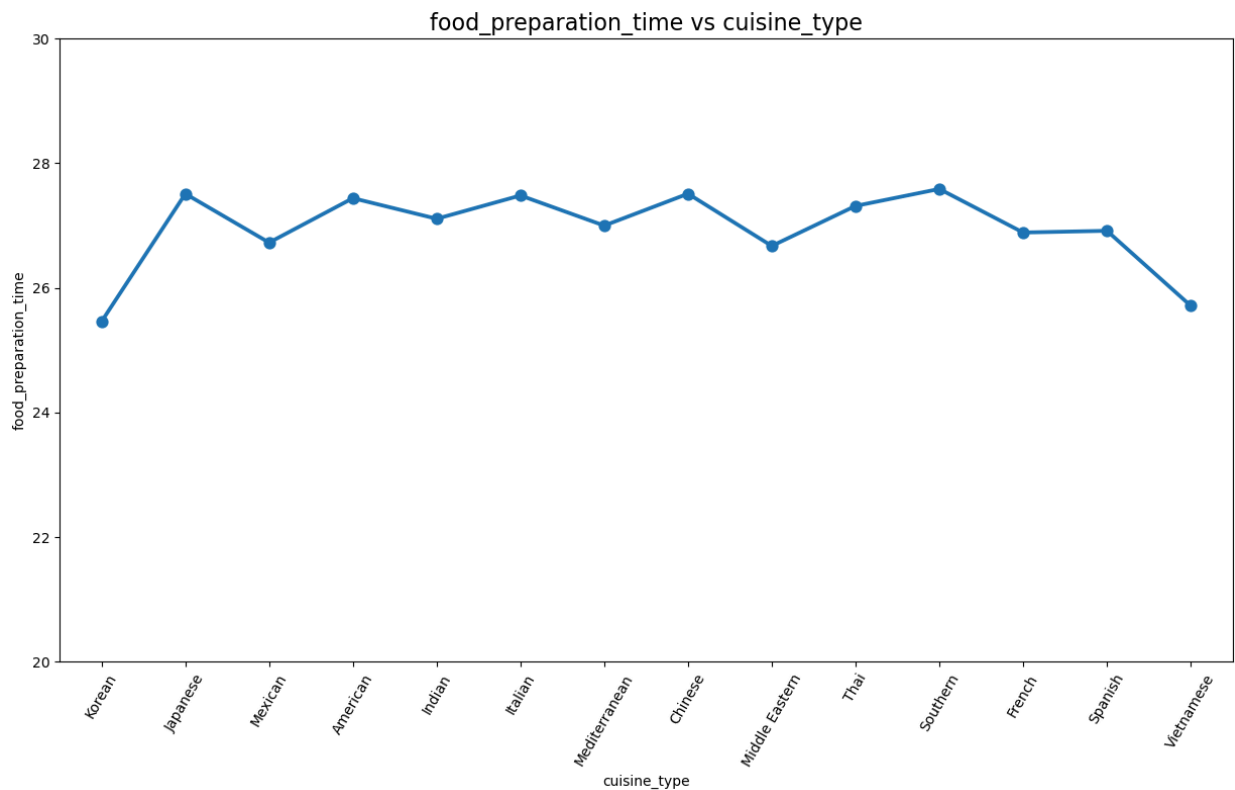


- As per the boxplot, there are outliers present for Korean, Mediterranean and Vietnamese cuisines. Korean and Mediterranean cuisines have some unique high cost dishes or may be due to some addons but they don't seem to be very extreme so we do not need to treat or ignore them
- Southern cuisine has the maximum range of cost for 50% of their orders i.e higher IQR ranging from approx. 12 to 27
- Median values are higher for French and Thai than rest which shows that 50% of these cuisine orders are approx. below 20 and 18 respectively
- Average cost of French, Southern, Thai, Spanish and Middle Eastern orders are higher than the rest of the cuisine types while Vietnamese and Korean have lesser average order cost.

```
In [49]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='cuisine_type', y='food_preparation_time', palette = 'magma')
plt.title('food_preparation_time vs cuisine_type', fontsize = 16)
plt.xticks(rotation=60);
plt.show()
```

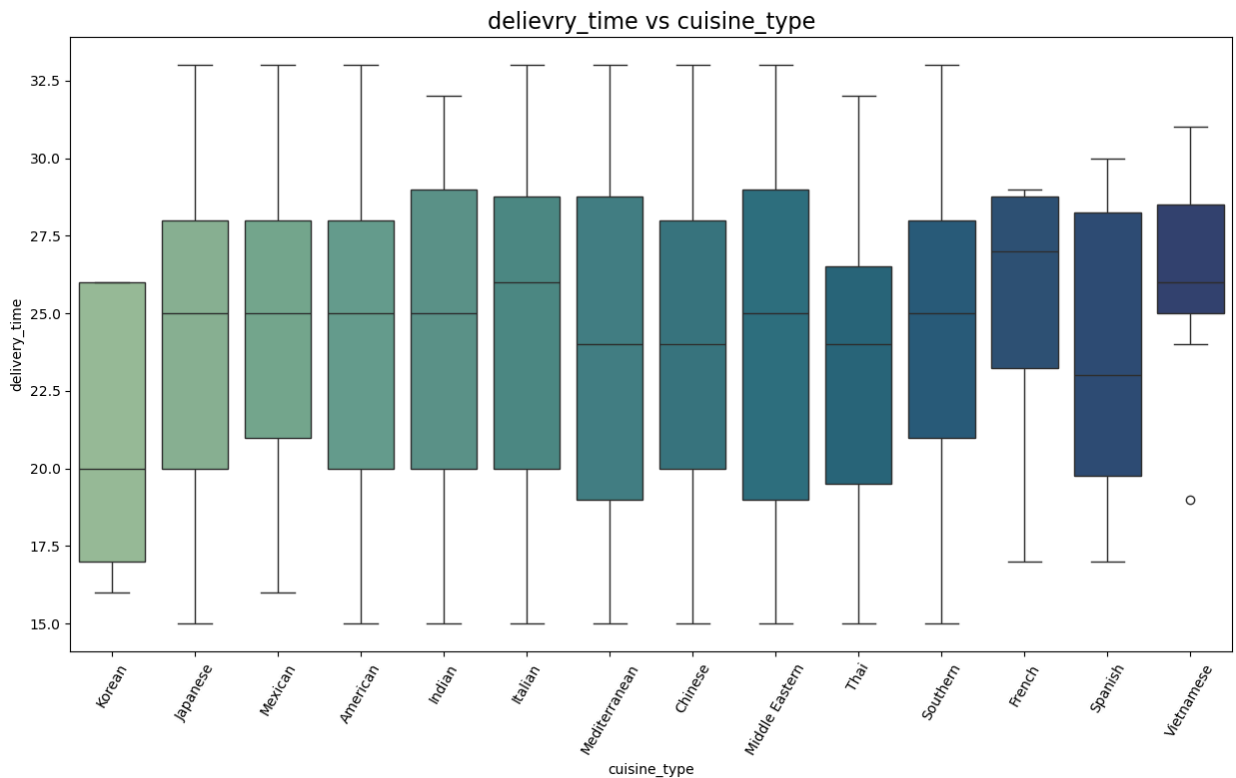


```
In [50]: plt.figure(figsize=(15,8))
sns.pointplot(data=data, x='cuisine_type', y='food_preparation_time', errorbar='ci')
plt.xticks(rotation=60);
plt.title('food_preparation_time vs cuisine_type', fontsize = 16)
plt.ylim(20,30)
plt.show()
```

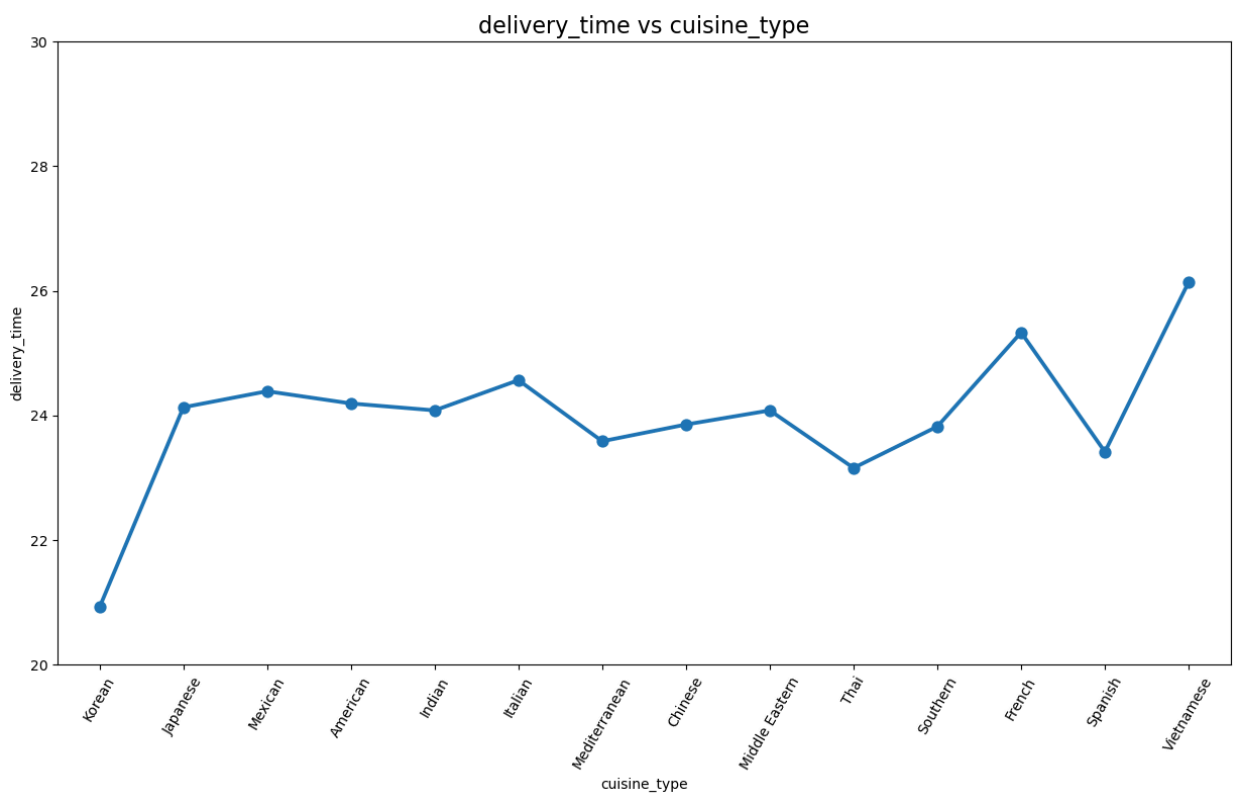


- There are outliers present in the Korean cuisine which means that there are few Korean cuisine orders which take more than the normal distribution
- Most of the boxes are of more or less nearly symmetrical distribution except few skewed distributions like French, southern, Thai, Korean etc.
- On an average, the food preparation time does not vary much across cuisines as seen from the point plot

```
In [51]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='cuisine_type', y='delivery_time', palette = 'crest', l
plt.title('delievry_time vs cuisine_type', fontsize = 16)
plt.xticks(rotation=60);
plt.show()
```



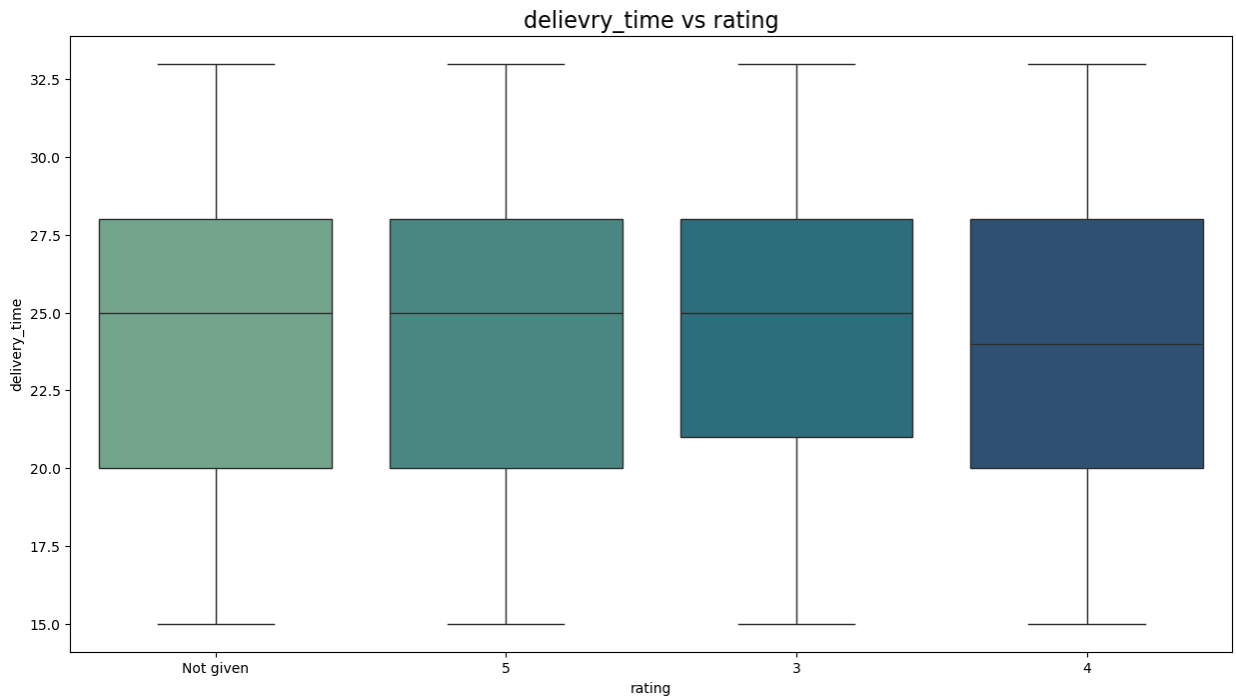
```
In [52]: plt.figure(figsize=(15,8))
sns.pointplot(data=data, x='cuisine_type', y='delivery_time', errorbar=('ci', False))
plt.xticks(rotation=60);
plt.title('delivery_time vs cuisine_type', fontsize = 16)
plt.ylim(20,30)
plt.show()
```



- Vietnamese has one outlier below Q1

- Median value for Korean is the lowest i.e around 20 while the same for French , Vietnamese are higher as per the boxplot
- But overall, all the delivery times range are more or less similar except Korean which has lower delivery times

```
In [53]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='rating', y='delivery_time',palette = 'crest',hue='ra
plt.title('delievry_time vs rating',fontsize = 16)
plt.show()
```

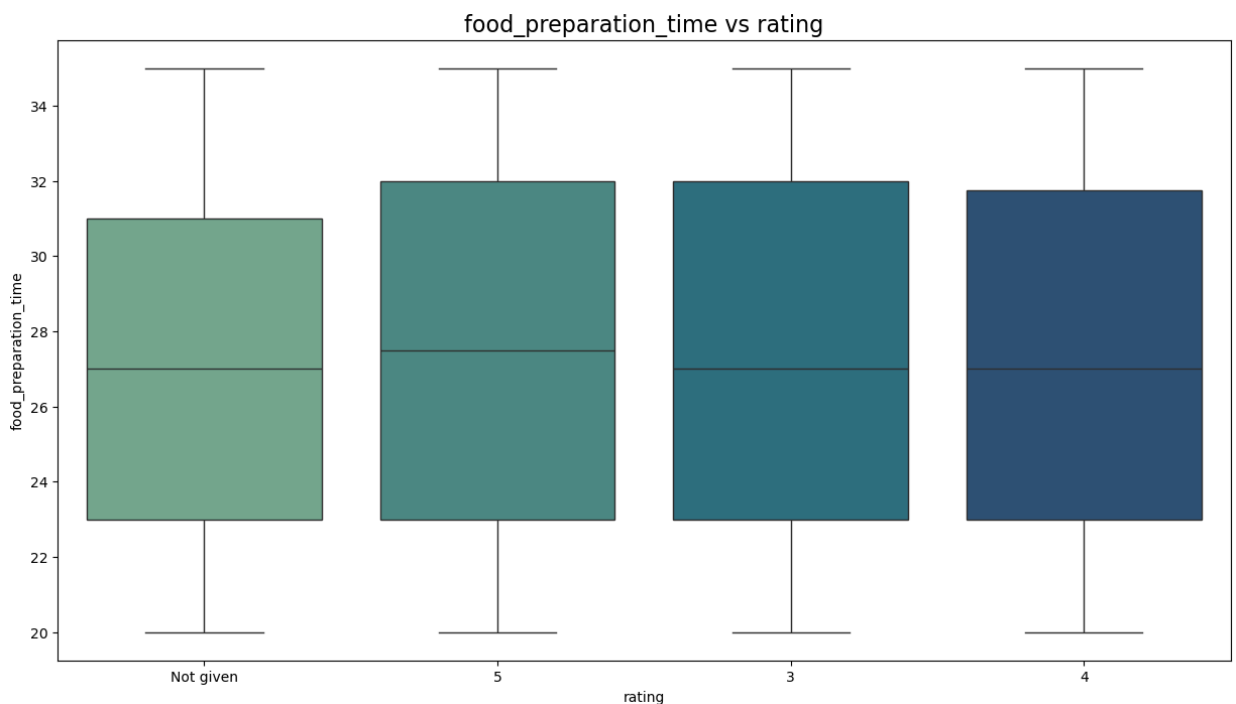


```
In [54]: plt.figure(figsize=(15, 8))
sns.pointplot(x = 'rating', y = 'delivery_time', data = data,errorbar=('ci',Fa
#sns.pointplot(x = [['3','4','5','Not given']], y = 'delivery_time', data = da
plt.title('delievry_time vs rating',fontsize = 16)
plt.show()
```

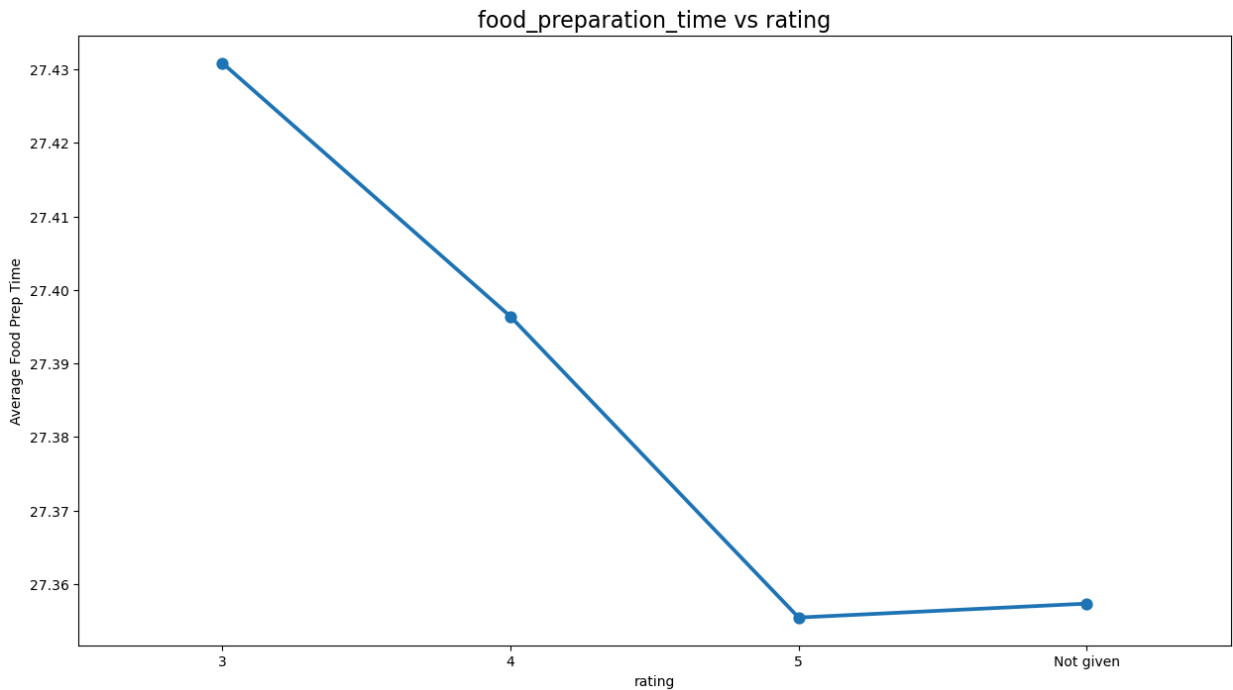


- Ratings do not seem to depend much on the delivery time as the distributions are more or less same in the boxplot
- Though the average delivery time is higher for rating 3, goes down for 4 but again it goes up for rating 5 and not rated indicating randomness in delivery time with respect to rating. Moreover, the difference is quite low, approximately 1 min. That shows no correlation between delivery time and rating.

```
In [55]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='rating', y='food_preparation_time',palette = 'crest')
plt.title('food_preparation_time vs rating',fontsize = 16)
plt.show()
```



```
In [56]: plt.figure(figsize=(15, 8))
sns.pointplot(x = 'rating', y = 'food_preparation_time', data = data,errorbar=
plt.title('food_preparation_time vs rating',fontsize = 16)
plt.ylabel('Average Food Prep Time')
plt.show()
```

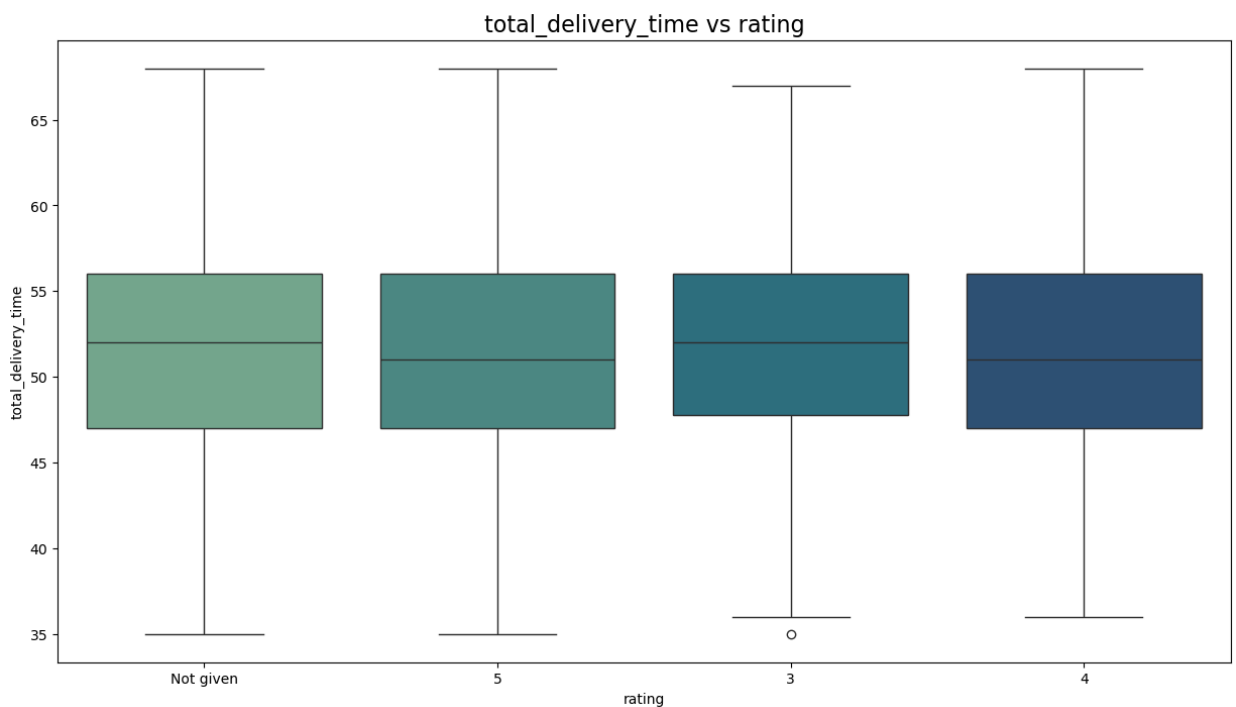


- The box plot shows the distributions are of nearly symmetric and uniform
- The rating does not seem to depend hugely on food preparation time as the distribution is very similar across all rating with nearly same median and quartile values.
- 50% of the orders take around less than or equal to 27 mins for rating 3,4 and 5
- Even though the lineplot shows a huge rise for rating 3 but the value difference is within seconds for average food preparation time

```
In [57]: data_copy=data.copy() ## Creating a copy of the data to calculate the total de
data_copy['total_delivery_time'] = data_copy['delivery_time'] + data_copy['food
plt.figure(figsize=(15, 8))
sns.pointplot(x = 'rating', y = 'total_delivery_time', data = data_copy,errorb
plt.title('total_delivery_time vs rating',fontsize = 16)
plt.show()
```

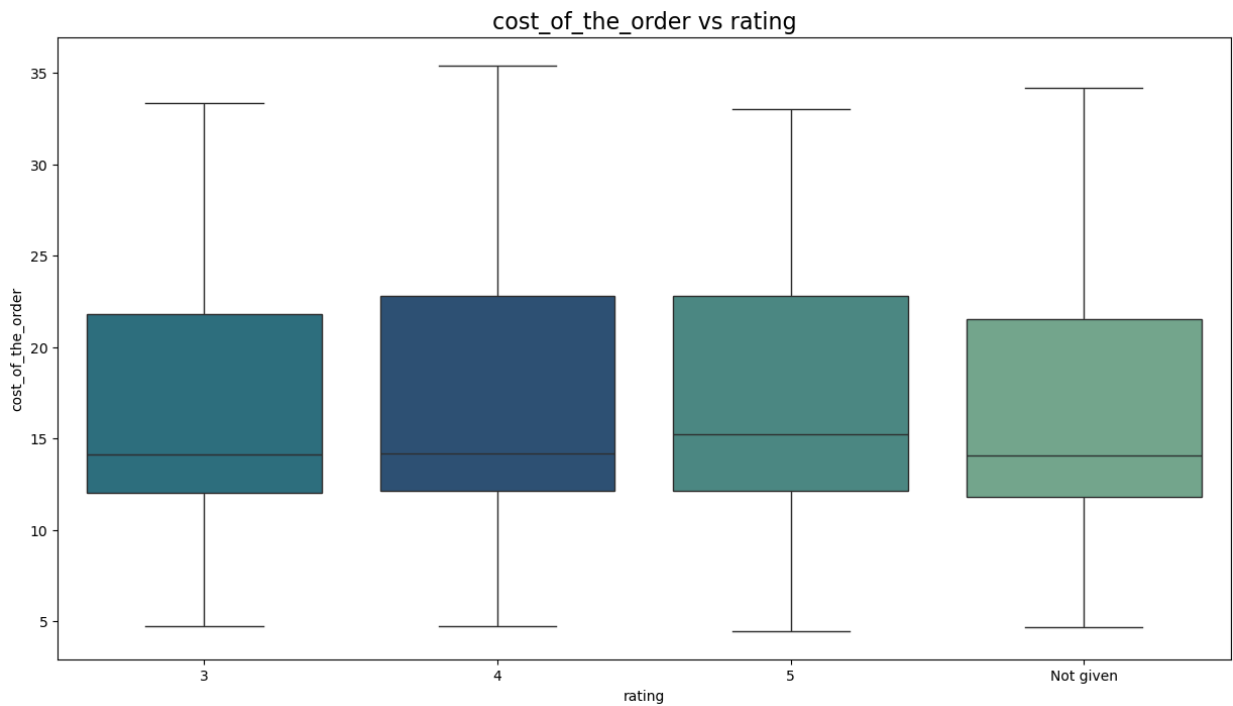


```
In [58]: plt.figure(figsize=(15,8))
sns.boxplot(data=data_copy, x='rating', y='total_delivery_time',palette = 'crest')
plt.title('total_delivery_time vs rating',fontsize = 16)
plt.show()
```

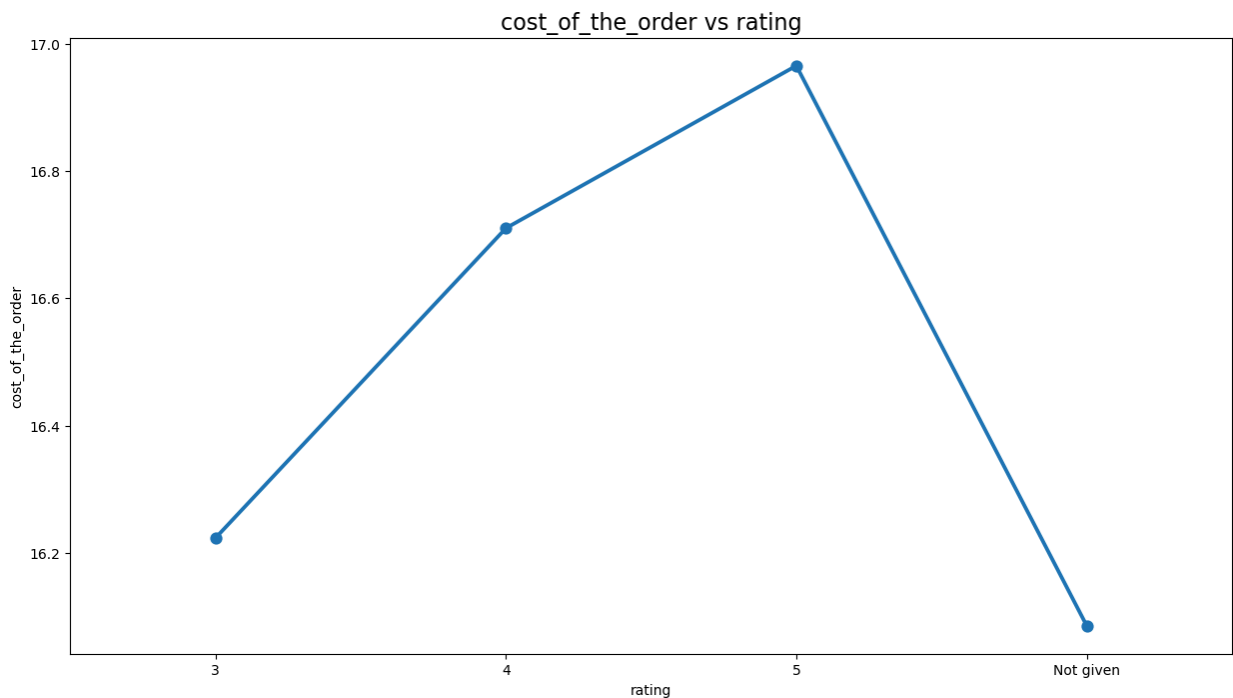


- This confirms that the rating do not depend on the delivery and food prepartaion time as the time difference between various ratings is hardly within mins.

```
In [59]: plt.figure(figsize=(15,8))
sns.boxplot(data=data, x='rating', y='cost_of_the_order',palette = 'crest',hue:
plt.title('cost_of_the_order vs rating',fontsize = 16)
plt.show()
```



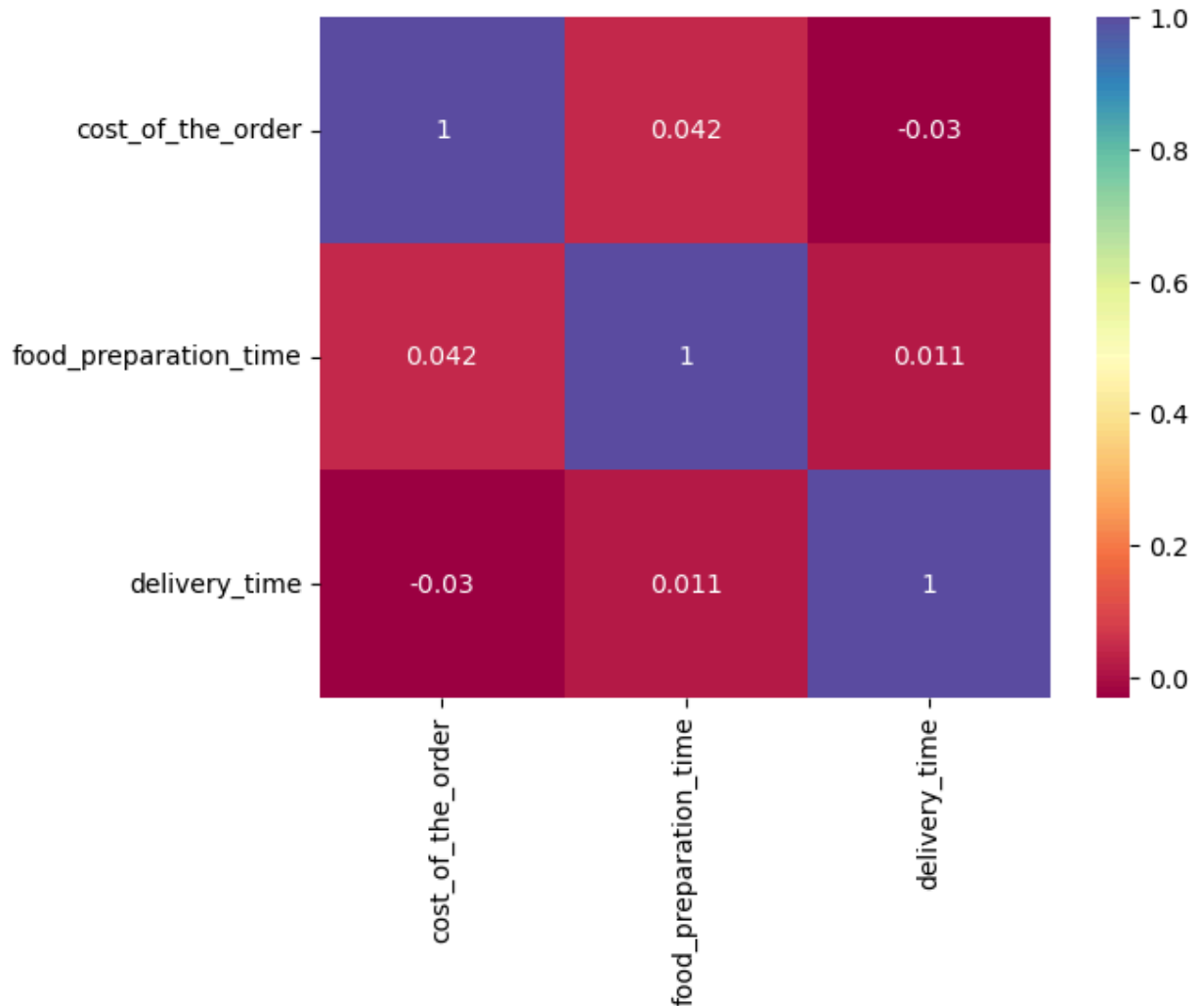
```
In [60]: plt.figure(figsize=(15, 8))
sns.pointplot(x = 'rating', y = 'cost_of_the_order', data = data,errorbar=('ci
plt.title('cost_of_the_order vs rating',fontsize = 16)
plt.show()
```



- Median Cost of the order is less for ratings 3 and 4 i.e below or equal to 15 while the same for rating 5 is slightly above 15
- 50% of the orders cost ranges from 13 to 22 for rating 3 , 13 to 23 for rating 4 and 5.
- In the box plot the box for not rated is very similar to rating 3 as in their Q1, Q2 and Q3 values are similar
- On an average the cost of the order is in increasing trend but the slope is not very high

Lets check the correlation between the numuerical variables

```
In [61]: num_corr = data[['cost_of_the_order', 'food_preparation_time', 'delivery_time']]
sns.heatmap(data = num_corr, annot = True, cmap = 'Spectral');
```



- From the heatmap we can observe that there is no correlation among these variables

Lets check the top restaurants as per revenue generation

```
In [62]: # This is to filter the dataframe for those rows with valid rating values and
rated_data = data[data['rating'] != 'Not given'].copy()
rated_data['rating'] = rated_data['rating'].astype('int')# This is to convert
```

```
In [63]: # This function selects the top N or bottom N restaurants as per revenue gener
# number of orders and ratings
def process_selective_restaurant(data, rated_data, top = True, quantity = 10):
    top_res_revenue = data.groupby('restaurant_name').agg(num_orders=('restaurant
        total_order_cost=('cost_of_the_order', 'sum'),avg_prep_time=('
    if top:
        # Top restaurant by revenue
        top_res_revenue = data.groupby('restaurant_name').agg(num_orders=('res
            total_order_cost=('cost_of_the_order', 'sum'),avg_prep_time=('
```

```

# storing them in a list to preserve the order
ordered_restaurants = top_res_revenue['restaurant_name'].tolist()

# computing the average rating of those top restaurants
data_rating = rated_data[rated_data['restaurant_name'].isin(ordered_restau
data_rating['restaurant_name'] = pd.Categorical(
    data_rating['restaurant_name'],
    categories=ordered_restaurants,
    ordered=True
)
# sort by this categorical order
data_rating = data_rating.sort_values('restaurant_name')
#data_rating mean
data_revenue_mean_rating = data_rating.groupby(['restaurant_name'])['rating

# combining the average rating with the revenue for the top restaurants
combined_df = pd.concat([top_res_revenue, data_revenue_mean_rating['rating
combined_df['avg_cost_per_order'] = combined_df['total_order_cost']/combine
return combined_df

```

In [64]: processed_top15 = process_selective_restaurant(data,rated_data, top=True, quan
processed_top15

Out[64]:

	restaurant_name	num_orders	total_order_cost	avg_prep_time	rating	avg_cost_per_order
0	Shake Shack	219	3579.53	27.95	4.28	16.34
1	The Meatball Shop	132	2145.21	27.18	4.51	16.25
2	Blue Ribbon Sushi	119	1903.95	27.92	4.22	16.00
3	Blue Ribbon Fried Chicken	96	1662.29	27.20	4.33	17.32
4	Parm	68	1112.76	27.31	4.13	16.36
5	RedFarm Broadway	59	965.13	27.59	4.24	16.36
6	RedFarm Hudson	55	921.21	27.16	4.18	16.75
7	TAO	49	834.50	26.78	4.36	17.03
8	Han Dynasty	46	755.29	27.41	4.43	16.42
9	Blue Ribbon Sushi Bar & Grill	44	666.62	26.30	4.59	15.15
10	Rubirosa	37	660.45	28.24	4.12	17.85
11	Sushi of Gari 46	37	640.87	27.54	4.24	17.32
12	Nobu Next Door	42	623.67	27.76	4.35	14.85
13	Five Guys Burgers and Fries	29	506.47	25.62	4.56	17.46
14	Momoya	30	492.13	27.30	4.27	16.40

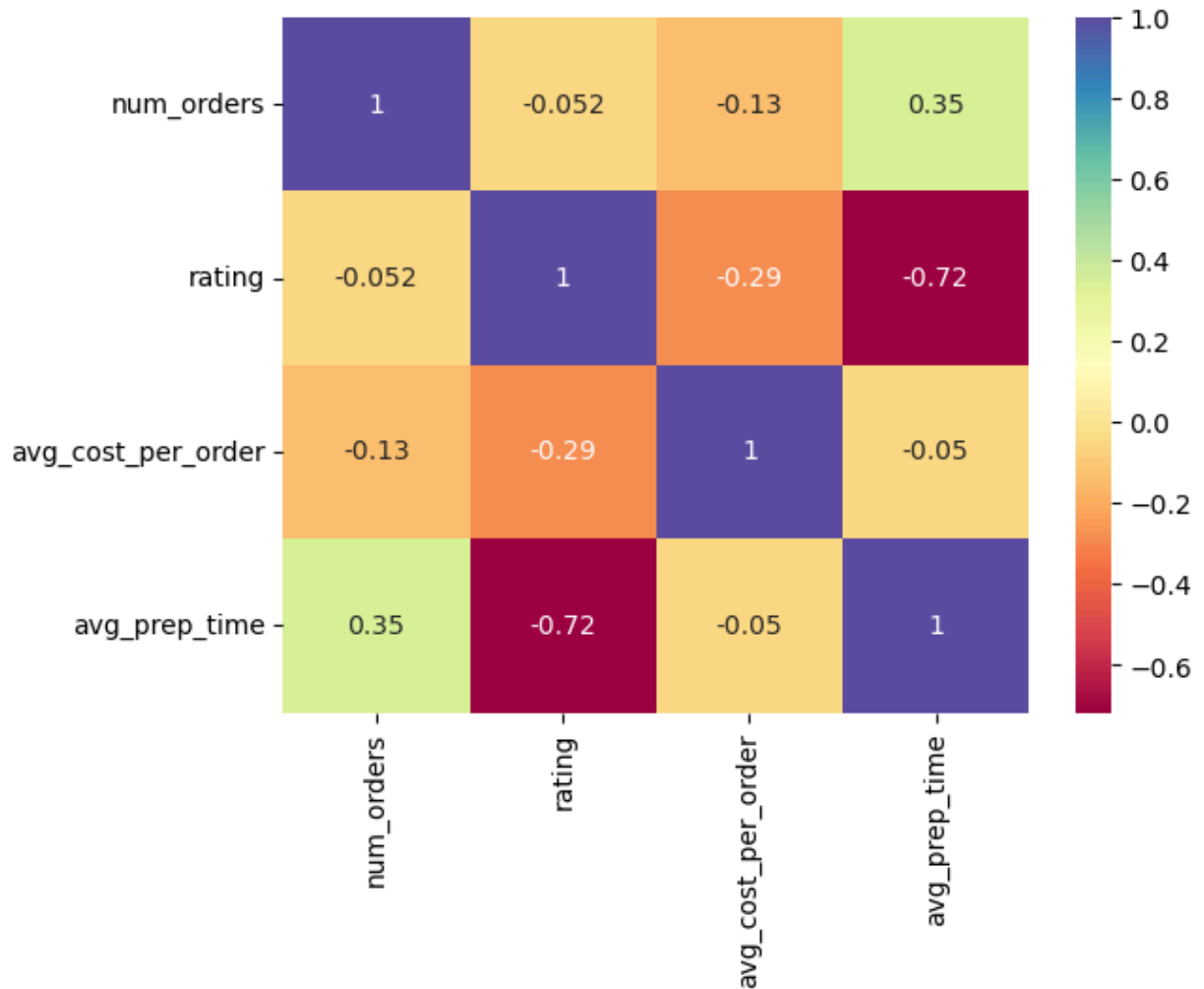
- The combined dataframe obtained indicates that very high rating (in the range of 4.5-5.0) doesnt result in higher revenue, since most of the restaurant have average rating

less than 4.5 except The Meatball Shop and Five Guys Burgers&Fries.

- Still, none of the top restaurant have very low rating either (average rating being more than 4.0.)

```
In [65]: col_list = processed_top15[['num_orders', 'rating', 'avg_cost_per_order', 'avg_prep_time']]
sns.heatmap(data= col_list, annot = True, cmap = 'Spectral')
```

Out[65]: <Axes: >



- Performing multivariate analysis on the top 15 restaurants we found that average cost per order slightly impacts negatively, the number of orders (-0.13) as well as the rating (-0.29). Hence, the top restaurant may still get better rating if they regulate the cost of the order, but that will impact the revenue generation (necessitating an optimal tradeoff between these two factors).
- The 'avg_prep_time' also adversely impact the rating of the top 15 restaurant (-0.72).

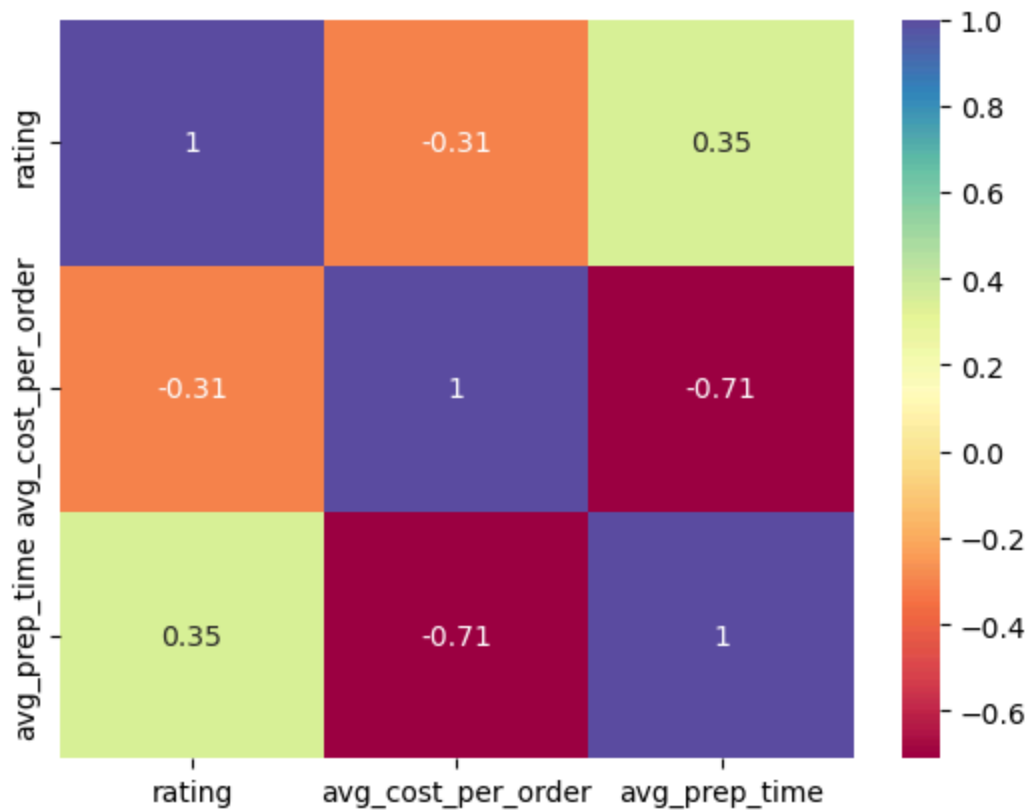
```
In [66]: processed_bottom15 = process_selective_restaurant(data, rated_data, top=False, c
processed_bottom15
```

Out[66]:

	restaurant_name	num_orders	total_order_cost	avg_prep_time	rating	avg_cost_per_order
0	Rye House	1	12.13	26.00	4.00	12.13
1	Frank Restaurant	1	12.08	20.00	4.00	12.08
2	Samurai Mama	1	11.64	21.00	5.00	11.64
3	Lamarca Pasta	1	9.22	23.00	NaN	9.22
4	Balade	1	9.22	25.00	5.00	9.22
5	Gaia Italian Cafe	1	8.78	25.00	NaN	8.78
6	Chola Eclectic Indian Cuisine	1	8.73	30.00	5.00	8.73
7	Woorijip	1	8.25	26.00	3.00	8.25
8	'wichcraft	1	8.10	28.00	5.00	8.10
9	La Follia	1	8.05	28.00	NaN	8.05
10	Market Table	1	6.79	21.00	NaN	6.79
11	Wa Jeal	1	6.74	32.00	NaN	6.74
12	Cipriani Le Specialita	1	5.92	35.00	5.00	5.92
13	Big Wong Restaurant _x¼Ñ¼	1	5.92	34.00	NaN	5.92
14	Hunan Manor	1	5.72	31.00	5.00	5.72

In [67]: `col_list_bottom = processed_bottom15[['rating', 'avg_cost_per_order', 'avg_prep_time']]`
`sns.heatmap(data= col_list_bottom, annot = True, cmap = 'Spectral')`

Out[67]: `<Axes: >`



- Performing multivariate analysis on the bottom 15 restaurants we found that average cost per order slightly impacts negatively the rating (-0.31). Hence, irrespective of the revenue generated by the restaurants if they regulate the cost of the order they might get better ratings. But since lower cost of order, impact the revenue generation it necessitates an optimal tradeoff between these two factors.
- The 'avg_prep_time' also adversely impact the rating of the bottom 15 restaurant (-0.71).

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 [68]: rated_data = data[data['rating'] != 'Not given'].copy() # This is to filter the
rated_data['rating'] = rated_data['rating'].astype('int') # Converting the col

In [69]: data_rating_count = rated_data.groupby(['restaurant_name'])['rating'].count().reset_index()
data_rating_count
```

Out [69]:

	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
...
151	Frank Restaurant	1
152	Socarrat Paella Bar	1
153	El Parador Cafe	1
154	Lucky Strike	1
155	'wichcraft	1

156 rows x 2 columns

```
In [70]: rest_names = data_rating_count[data_rating_count['rating']>50]['restaurant_name']
data_avg_more_than_4 = rated_data[rated_data['restaurant_name'].isin(rest_names)]

data_mean_rating = data_avg_more_than_4.groupby(['restaurant_name'])['rating'].mean()
df_avg_rating_greater_than_4 = data_mean_rating[data_mean_rating['rating'] > 4]
df_avg_rating_greater_than_4
```

Out [70]:

	restaurant_name	rating
0	The Meatball Shop	4.51
1	Shake Shack	4.28
2	Blue Ribbon Sushi	4.22
3	Blue Ribbon Fried Chicken	4.33

```
In [71]: # Using one liner as an alternative
rated_data.groupby('restaurant_name')['rating'].agg(['count', 'mean']).reset_index()
```

Out [71]:

	restaurant_name	count	mean
16	Blue Ribbon Fried Chicken	64	4.33
17	Blue Ribbon Sushi	73	4.22
117	Shake Shack	133	4.28
132	The Meatball Shop	84	4.51

Observations:

There are **four restaurants** which has no of ratings **more than 50** and their average rating is greater than 4

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 [72]: #function to determine the revenue
def compute_revenue(x):
    if x > 20:
        return x*0.25
    elif x > 5:
        return x*0.15
    else:
        return x*0

data['net_revenue'] = data['cost_of_the_order'].apply(compute_revenue)
data.head()
data['net_revenue']
```

```
Out[72]: 0      7.69
1      1.81
2      1.83
3      7.30
4      1.74
...
1893   5.58
1894   1.83
1895   6.30
1896   1.83
1897   2.92
Name: net_revenue, Length: 1898, dtype: float64
```

```
In [73]: total_revenue = data['net_revenue'].sum()
print("The Total revenue generated by the company across all the orders is around", total_revenue)

The Total revenue generated by the company across all the orders is around 6166.3 dollars
```

```
In [74]: print("No of orders with zero revenue is", data['net_revenue'].isnull().sum())

No of orders with zero revenue is 0
```

Observations:

The company is earning revenue on all the orders as all of them are atleast greater than 5 dollars. And the total revenue generated from all the orders is around **6166 dollars**

Question 15: The company wants to analyze the total time required to deliver the food. What percentage of orders take more than 60 minutes to get delivered from the time the order is placed? (The food has to be prepared and then delivered.) [2 marks]

```
In [75]: # Write the code here
data['total_deliver_time'] = data['food_preparation_time'] + data['delivery_time']
```

```
percentage_of_orders = (((data['total_deliver_time'] > 60).sum() / len(data)) * 100)
print(f"The percentage of orders that take more than 60 mins to get delivered is {percentage_of_orders}%")
```

The percentage of orders that take more than 60 mins to get delivered is 10.54%

Observations:

Around **10%** of orders take more than 60 mins to get delivered to the customer from the time the order is placed.

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 [76]: # Write the code here
print('The mean delivery time on weekdays is around',
      round(data[data['day_of_the_week'] == 'Weekday']['delivery_time'].mean(),
            'minutes'))
```

The mean delivery time on weekdays is around 28 minutes

```
In [77]: print('The mean delivery time on weekdays is around',
              round(data[data['day_of_the_week'] == 'Weekend']['delivery_time'].mean(),
                    'minutes'))
```

The mean delivery time on weekends is around 22 minutes

Observations:

The mean delivery time is around **28 mins on weekdays** and around **22 mins on weekends** and this shows that the delivery takes higher time on weekdays than weekends.

Conclusion and Recommendations

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

Conclusions:

- There are 1898 rows and 9 columns in the food hub order dataset
- Shake Shack is the top restaurant which has received the highest no of orders. The Meatball Shop and Blue Ribbon Sushi are the next largest restaurants.
- The top 3 cuisine types are American, Japanese and Italian. American being the most popular of all. The bottom 3 cuisine types are Vietnamese, Spanish and Korean.
- The average cost of the order is 16.5 dollars and 50% of the orders are priced between 12 to 22.

- Most of the ratings are on the higher side and there are no ratings below 3. But there are 736 orders which are not rated which might indicate that the customers either didn't bother or didn't want to rate because of less engagement or satisfaction.
- The percentage of orders received cost above 20 dollars is around 29%.
- On an average an order takes ~24 mins to be delivered to the customer.
- Average cost of French, Southern, Thai, Spanish and Middle Eastern orders are higher than the rest of the cuisine types.
- Average rating of Spanish and Thai orders are higher as compared to others while Vietnamese and Korean have received lowest average of ratings.
- No of orders placed on weekends are higher as compared to weekdays
- The mean delivery time is around 28 mins on weekdays and around 22 mins on weekends and this shows that the delivery takes higher time on weekdays than weekends.
- Around 10% of the orders take more than 60 mins to get delivered to the customer from the time the order is placed.
- There is no much correlation among the variables that means ratings do not have a high dependency on any of the variables when full dataset is considered.
- While performing multivariate analysis on the top 15 and bottom 15 restaurants as per revenue generation, we observed that rating is slightly negatively correlated with the average cost of the order and also strongly negatively correlated with the average food preparation time. But since these analysis were performed on a very small subset of the dataset, we cannot conclude the above inferences can be made on the entire dataset.
- While expanding the analysis to top and bottom 100 restaurants such strong correlations weakens drastically.

Recommendations for the Food Aggregator firm:

- Strengthen the collaboration with existing restaurants which serves top cuisine types like American, Japanese, Italian and also explore new collaboration for these top performing cuisine types.
- Collaborate with those restaurants that provides French, Spanish and Thai cuisines as generally these have higher average costs and good ratings.
- The company should prioritize revisiting collaboration with low performing restaurants (low revenue).
- The company might look at improving their delivery times specifically for weekdays to improve their business.

Recommendations for the Restaurants:

- Based on our multivariate analysis on a small subset of the dataset, i.e. the top 15 and bottom 15 restaurant based on total revenue; they can improve their rating if they regulate the cost of the order, but that will impact the revenue generation (necessitating an optimal tradeoff between these two factors).

- Similarly, the top 15 and bottom 15 restaurant can also improve their rating if they reduce their food preparation time.
 - Note: These recommendations are specifically for the edge cases with a smaller dataset, hence recommendations for each restaurant should be unique .
-