FoodHub Project

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

In [1]:
```
#import libraries needed for data manipulation

import numpy as np
import pandas as pd

#import libraries needed for data visualization

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

In [2]:
```
#import dataset named foodhub_order.csv

df = pd.read_csv('foodhub_order.csv')

#read first five rows of dataset

df.head()
```

Out[2]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
0	1477147	337525	Hangawi	Korean	30.75	Weekend	Not given	25	20
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25	23
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23	28
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25	15
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25	24

Observations:

Dataframe has 9 columns corresponding to different variables of dataset, further explained by the data dictionary above. Each row detais an order placed by a customer.

Question 1: Write the code to check the shape of the dataset and write your observations based on that.

In [3]:
```
#check shape of foodhub_order.csv dataset

df.shape
```

Out[3]:
```
(1898, 9)
```

Observations:

- The DataFrame has 1898 rows and 9 columns.

Question 2: Write the observations based on the below output from the info() method.

In [4]:
```
# use info() to print a concise summary of the DataFrame

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
```

In [5]:
```
# double check for null values in the dataset

df.isnull().sum()
```

Out[5]:
```
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:

- There are no missing values in the dataset, with a total of 1898 non-null observations in each of the columns.
- Using info() tells us the datatypes of all 9 variables in the foodhub_order.csv dataset.
  - 4 are integer type: 'order_id', 'customer_id', 'food_preparation_time', 'delivery_time'
  - 1 is float type: 'cost_of_the_order'
  - 4 are object type: 'restaurant_name', 'cuisine_type', 'day_of_the_week', 'rating'
- Total memory usage is approximately 133.6 KB.
- There are no missing values in the dataset.

Question 3: 'restaurant_name', 'cuisine_type', 'day_of_the_week' are object types. Write the code to convert the mentioned features to 'category' and write your observations on the same.

In [6]:
```
# converting "objects" to "category" reduces the data space required to store the dataframe
# below is code to convert 'restaurant_name', 'cuisine_type', 'day_of_the_week' into categorical data
# 'ratings' must stay an object type since the values are mixed.

df['restaurant_name'] = df.restaurant_name.astype('category')
df['cuisine_type'] = df.cuisine_type.astype('category')
df['day_of_the_week'] = df.day_of_the_week.astype('category')

# use info() to print a concise summary of the DataFrame

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   category
 3   cuisine_type           1898 non-null   category
 4   cost_of_the_order      1898 non-null   float64 
 5   day_of_the_week        1898 non-null   category
 6   rating                 1898 non-null   object  
 7   food_preparation_time  1898 non-null   int64   
 8   delivery_time          1898 non-null   int64   
dtypes: category(3), float64(1), int64(4), object(1)
memory usage: 102.7+ KB
```

Observations:

- 'restaurant_name', 'cuisine_type' and 'day_of_the_week' are now converted into categorical values.
- Total memory usage has decreased from 133.6+ to 102.7+ KB.

Question 4: Write the code to find the summary statistics and write your observations based on that.

In [7]:
```
# get the summary statistics of the numerical data
df.describe()
```

Out[7]:

	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

	order_id	customer_id	cost_of_the_order	food_preparation_time	delivery_time
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:

- Order ID and Customer ID are just identifiers for each order.
- The cost of an order ranges from 4.47 to 35.41 dollars, with an average order costing around 16 dollars and a standard deviation of 7.5 dollars. The cost of 75% of the orders are below 23 dollars. This indicates that most of the customers prefer low-cost food compared to the expensive ones.
- Food preparation time ranges from 20 to 35 minutes, with an average of around 27 minutes and a standard deviation of 4.6 minutes. The spread is not very high for the food preparation time.
- Delivery time ranges from 15 to 33 minutes, with an average of around 24 minutes and a standard deviation of 5 minutes. The spread is not too high for delivery time either.
- For the variables cost_of_the_order, food_preparation_time, and delivery time, the mean and 50% percentile are close in value, indicating a normal non-skewed distribution for those variables.

Question 5: How many orders are not rated?

In [8]:
```
#count total orders that have a rating (non-null)

df[df['rating']=="Not given"].shape[0]
```

Out[8]:
```
736
```

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

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

In [10]:
```
#second way to calculate it to confirm

df.loc[df.rating == 'Not given','rating'].count()
```

Out[10]:
```
736
```

Observations:

Out of 1898 customer orders, 736 have an entry "Not given," i.e. there is no existing rating.

Question 6: Explore all the variables and provide observations on the distributions of all the relevant variables in the dataset. (UNIVARIATE ANALYSIS)

In [11]:
```
df.corr()

# Refer back to Question 4 describe function for 5 main statistics (min, median, max, mean, std)
```

Out[11]:

	order_id	customer_id	cost_of_the_order	food_preparation_time	delivery_time
order_id	1.000000	-0.013960	0.021171	-0.007026	0.009690
customer_id	-0.013960	1.000000	-0.001322	-0.030330	-0.009885
cost_of_the_order	0.021171	-0.001322	1.000000	0.041527	-0.029949
food_preparation_time	-0.007026	-0.030330	0.041527	1.000000	0.011094
delivery_time	0.009690	-0.009885	-0.029949	0.011094	1.000000

In [12]:
```
# define a function to plot a boxplot and a histogram along the same scale

def histbox(data, feature, figsize=(12, 7), kde=False, bins=None):
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (12,7))
    kde: whether to show the density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (box, hist) = plt.subplots(
        nrows=2,                                            # Number of rows of the subplot grid = 2
                                                                # boxplot first then histogram created below
        sharex=True,                                        # x-axis same among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},        # boxplot 1/3 height of histogram
        figsize=figsize,                                    # figsize defined above as (12, 7)
    )
    # defining boxplot inside function, so when using it say histbox(df, 'cost'), df: data and cost: feature

    sns.boxplot(
        data=data, x=feature, ax=box, showmeans=True, color="chocolate"
    )  # showmeans makes mean val on boxplot have star, ax =
    sns.histplot(
        data=data, x=feature, kde=kde, ax=hist, bins=bins, color = "darkgreen"
    ) if bins else sns.histplot(
        data=data, x=feature, kde=kde, ax=hist, color = "darkgreen"
    )  # For histogram if there are bins in potential graph

    # add vertical line in histogram for mean and median
```

```
    hist.axvline(
        data[feature].mean(), color="purple", linestyle="--"
    )  # Add mean to the histogram
    hist.axvline(
        data[feature].median(), color="black", linestyle="-"
    )  # Add median to the histogram
```

In [13]:

```
# define a function to create labeled barplots

def bar(data, feature, perc=False, n=None):
    """
    Barplot with percentage at the top

    data: dataframe
    feature: dataframe column
    perc: whether to display percentages instead of count (default is False)
    n: displays the top n category levels (default is None, i.e., display all levels)
    """

    total = len(data[feature])  # length of the column
    count = data[feature].nunique()
    if n is None:
        plt.figure(figsize=(count + 1, 5))
    else:
        plt.figure(figsize=(n + 1, 5))

    plt.xticks(rotation=90, fontsize=15)
    ax = sns.countplot(
        data=data,
        x=feature,
        palette="Paired",
        order=data[feature].value_counts().index[:n].sort_values(),
    )

    for p in ax.patches:
        if perc == True:
            label = "{:.1f}%".format(
                100 * p.get_height() / total
            )  # percentage of each class of the category
        else:
            label = p.get_height()  # count of each level of the category

        x = p.get_x() + p.get_width() / 2  # width of the plot
        y = p.get_height()  # height of the plot

        ax.annotate(
            label,
            (x, y),
            ha="center",
            va="center",
            size=12,
            xytext=(0, 5),
```

```
            textcoords="offset points",
        )  # annotate the percentage

    plt.show()  # show the plot
```

In [14]:
```
# check unique order ID

df['order_id'].value_counts().shape
```

Out[14]: (1898,)

Observations:

- There are 1898 unique orders. As mentioned earlier, 'order_id' is just an identifier for the orders.

In [15]:
```
# check unique customer ID

df['customer_id'].value_counts().shape
```

Out[15]: (1200,)

Observations:

- There are 1200 unique customers. We can see that there are some customers who have placed more than one order.
- Let's check the top 5 customers' IDs who have ordered most frequently.

In [16]:
```
df['customer_id'].value_counts().head()
```

Out[16]:
```
52832     13
47440     10
83287      9
250494     8
65009      7
Name: customer_id, dtype: int64
```

Observations:

- Customer with ID 52832 has ordered 13 times.

In [17]:
```
# check unique restaurant name

df['restaurant_name'].value_counts().shape
```

Out[17]: (178,)

Observations:

- There are 178 unique restaurants.
- Let's check the number of orders that get served by the restaurants.

In [18]:
```
df['restaurant_name'].value_counts()
```

Out[18]:
```
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
                            ...
Klong                          1
Kambi Ramen House              1
Il Bambino                     1
Hunan Manor                    1
Lamarca Pasta                  1
Name: restaurant_name, Length: 178, dtype: int64
```

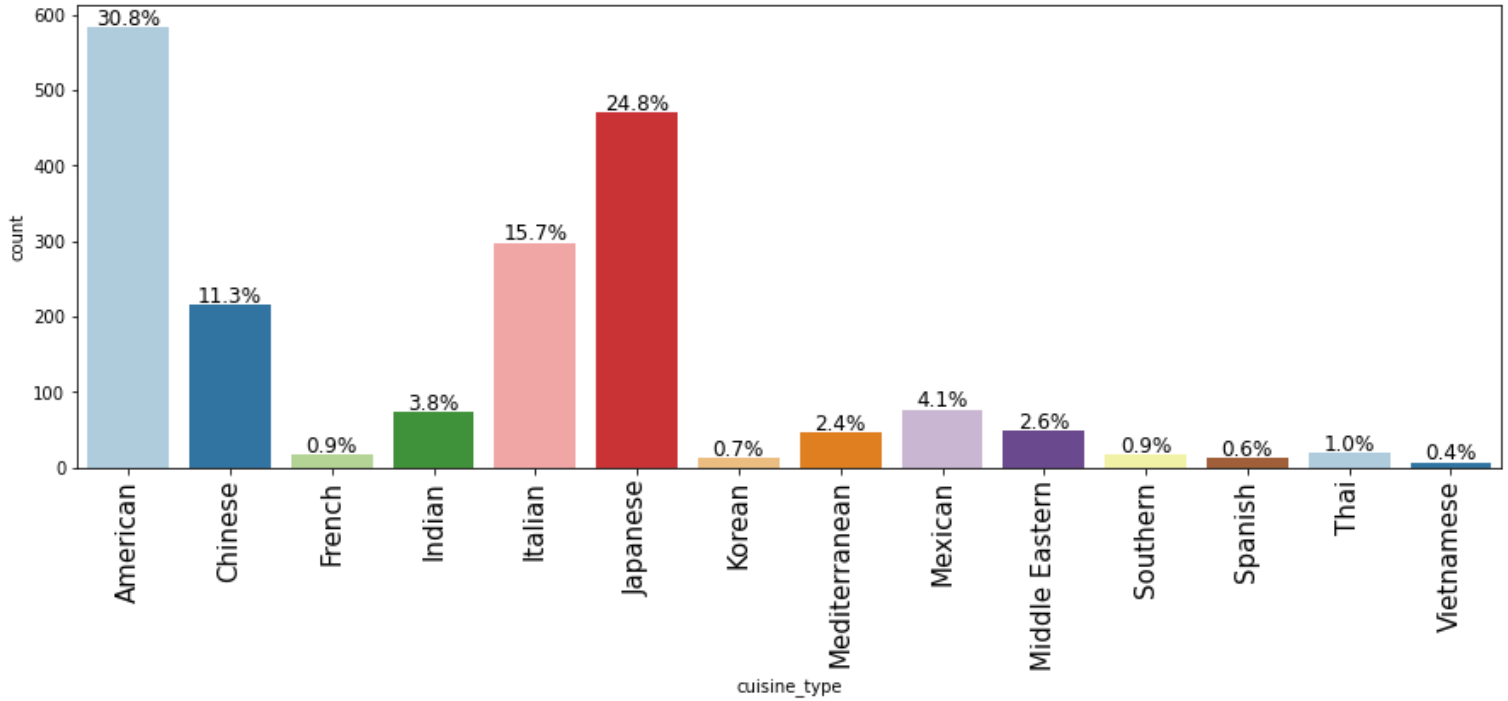
Observations:

- The restaurant that has received maximum number of orders is Shake Shack

In [19]:
```
# check unique cuisine type

df['cuisine_type'].value_counts().shape
```

Out[19]:
```
(14,)
```

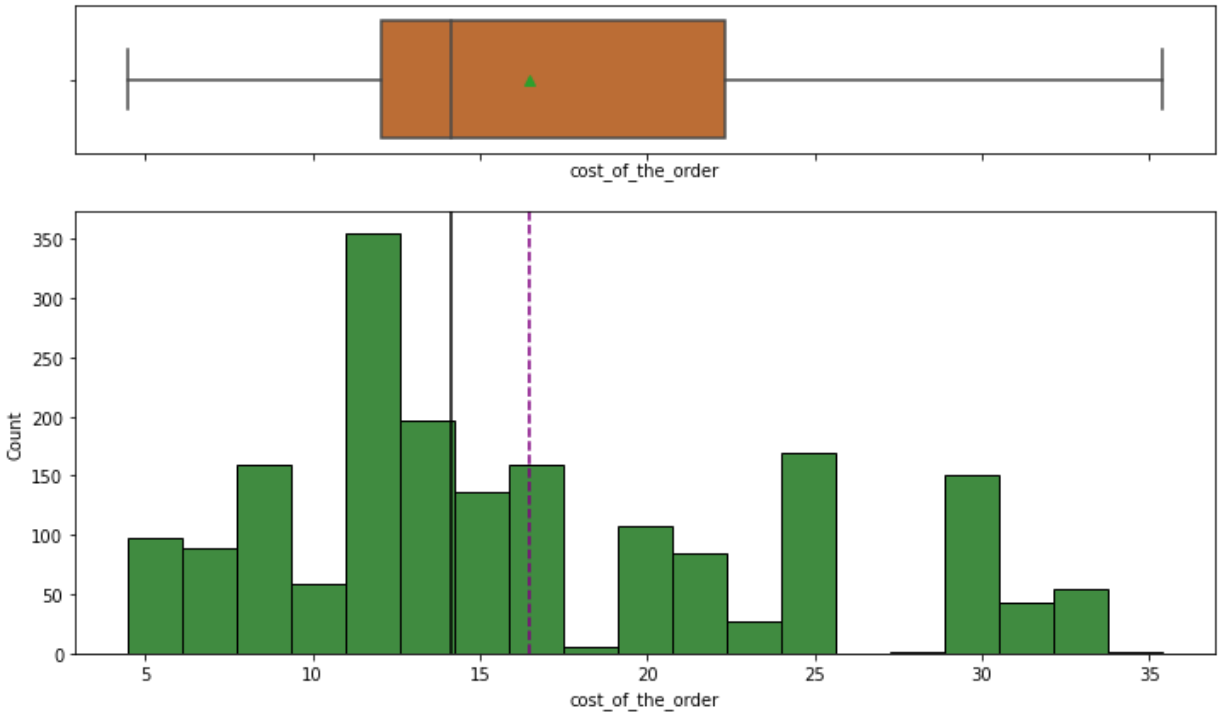
In [20]:
```
bar(df, 'cuisine_type', perc=True)
```



Observations:

- There are 14 unique cuisines in the dataset.
- The distribution of cuisine types show that cuisine types are not equally distributed.
- The most frequent cuisine type is American followed by Japanese and Italian.
- Vietnamese appears to be the least popular of all the cuisines.

In [21]:
```
histbox(df, 'cost_of_the_order')
```



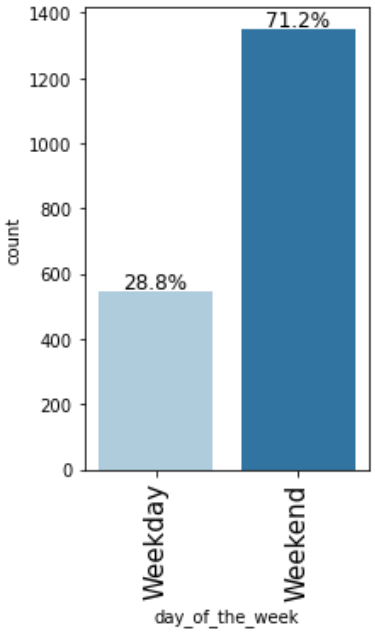
Observations:

- The average cost of the order is greater than the median cost indicating that the distribution for the cost of the order is right-skewed.
- The mode of the distribution indicates that a large chunk of people prefer to order food that costs around 10-12 dollars.
- There are few orders that cost greater than 30 dollars. These orders might be for some expensive meals.

In [22]:
```
# check the unique values

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

Out[22]:
```
Weekend    1351
Weekday     547
Name: day_of_the_week, dtype: int64
```

In [23]:
```
bar(df, 'day_of_the_week', perc=True)
```



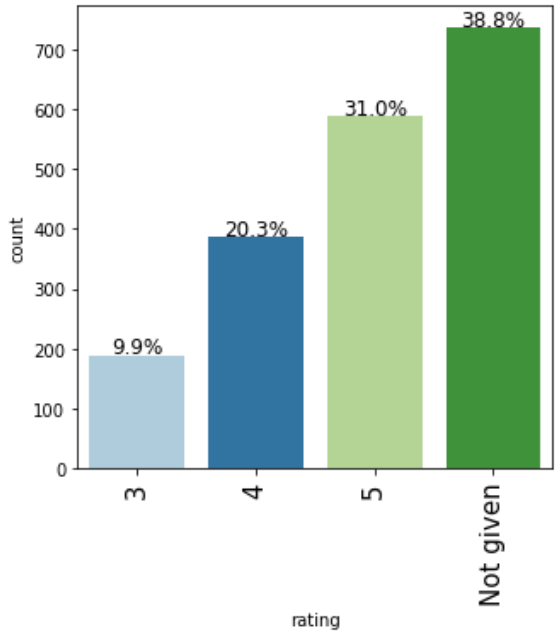
Observations:

- The 'day_of_the_week' columns consists of 2 unique values - Weekday and Weekend
- The distribution shows that around 71% of all orders are placed on weekends.

In [24]:
```
# check the unique values

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

Out[24]:
```
Not given    736
5            588
4            386
3            188
Name: rating, dtype: int64
```

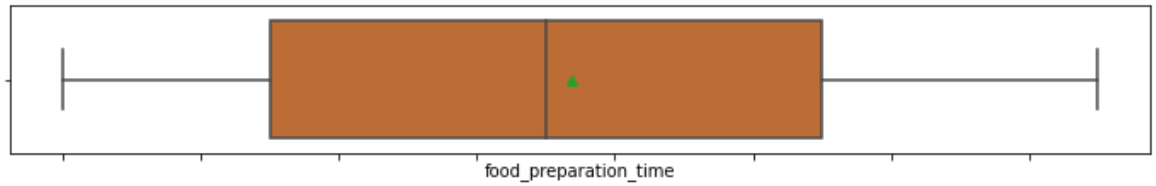
In [25]:
```
bar(df, 'rating', perc=True)
```

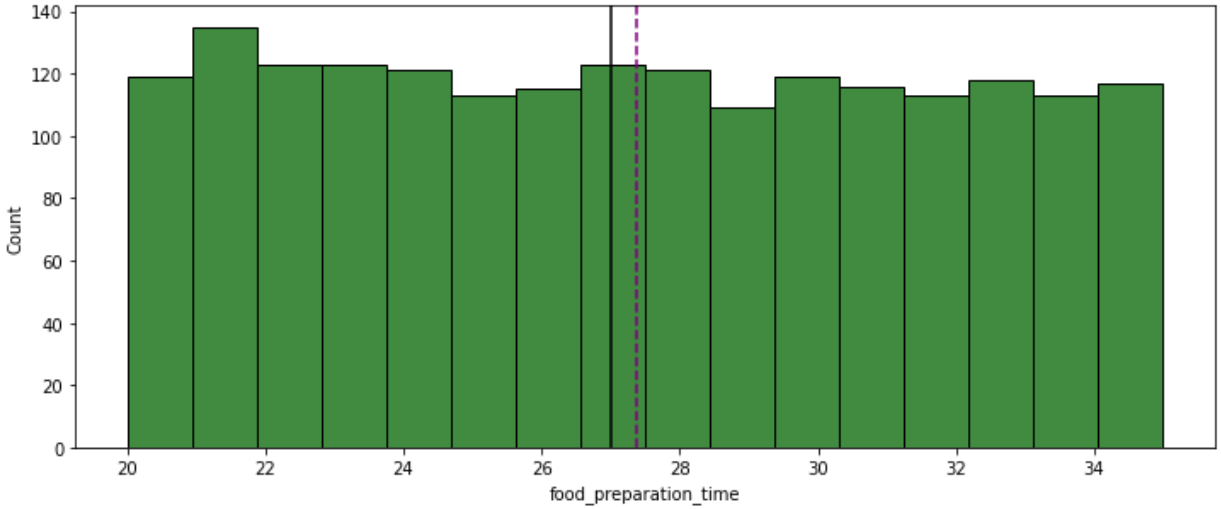


Observations:

- The distribution of 'rating' shows that the most frequent rating category is 'not given' (around 39%), followed by a rating of 5 (around 31%).
- Only 10% orders have been rated 3.

In [26]:
```
histbox(df, 'food_preparation_time', bins = 16)
```





In [27]:
```
# calculate mean food prep time for each cuisine type

df.groupby(by = ["cuisine_type"])['food_preparation_time'].mean()
```

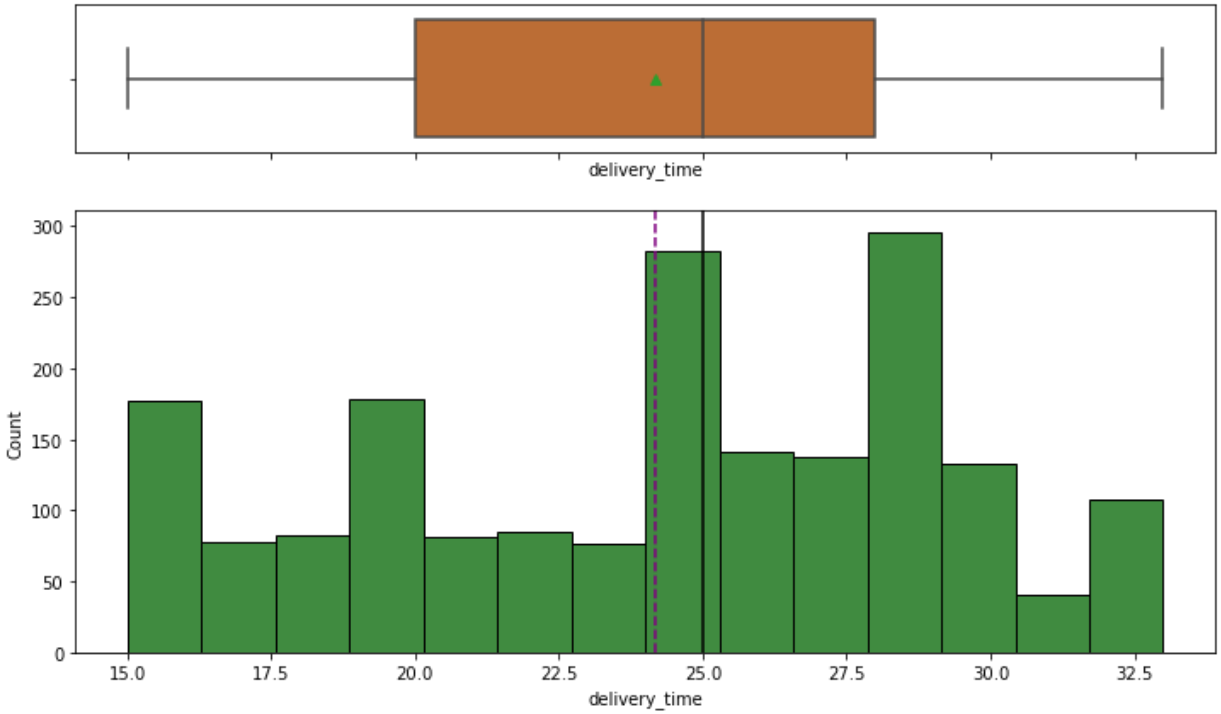
Out[27]:
```
cuisine_type
American          27.440068
Chinese           27.511628
French            26.888889
Indian            27.109589
Italian           27.483221
Japanese          27.510638
Korean            25.461538
Mediterranean     27.000000
Mexican           26.727273
Middle Eastern    26.673469
Southern          27.588235
Spanish           26.916667
Thai              27.315789
Vietnamese        25.714286
Name: food_preparation_time, dtype: float64
```

Observations:

- The average food preparation time is almost equal to the median food preparation time indicating that the distribution is nearly symmetrical.
- The food preparation time is pretty evenly distributed between 20 and 35 minutes.

- There are no outliers in this column.

In [28]:
```
histbox(df, 'delivery_time')
```



Observations:

- The average delivery time is a bit smaller than the median delivery time indicating that the distribution is a bit left-skewed.
- Comparatively more number of orders have delivery time between 24 and 30 minutes.
- There are no outliers in this column.

Question 7: Write the code to find the top 5 restaurants that have received the highest number of orders.

In [29]:
```
df['restaurant_name'].value_counts()[:5]
```

Out[29]:
```
Shake Shack                219
The Meatball Shop          132
Blue Ribbon Sushi          119
Blue Ribbon Fried Chicken   96
Parm                        68
Name: restaurant_name, dtype: int64
```

Observations:

- Top 5 popular restaurants that have received the highest number of orders **'Shake Shack', 'The Meatball Shop', 'Blue Ribbon Sushi', 'Blue Ribbon Fried Chicken' and 'Parm'**.
- Almost 33% of the orders in the dataset are from these restaurants.

Question 8: Write the code to find the most popular cuisine on weekends.

In [30]:
```
df.groupby(by = ["day_of_the_week"])['cuisine_type'].value_counts()
```

Out[30]:
```
day_of_the_week  cuisine_type
Weekday          American          169
                 Japanese          135
                 Italian            91
                 Chinese            52
                 Indian             24
                 Mexican            24
                 Middle Eastern     17
                 Mediterranean      14
                 Southern            6
                 French              5
                 Thai                4
                 Vietnamese          3
                 Korean              2
                 Spanish             1
Weekend          American          415
                 Japanese          335
                 Italian           207
                 Chinese           163
                 Mexican            53
                 Indian             49
                 Mediterranean      32
                 Middle Eastern     32
                 Thai               15
                 French             13
                 Korean             11
                 Southern           11
                 Spanish            11
                 Vietnamese          4
Name: cuisine_type, dtype: int64
```

In [31]:
```
# second way to calculate it

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

Out[31]:
```
American    415
Name: cuisine_type, dtype: int64
```

Observations:

- On weekends, the most popular cuisine type is American with 415 customer orders.

Question 9: Write the code to find the number of total orders where the cost is above 20 dollars. What is the percentage of such orders in the dataset?

In [32]:
```
Q9 = df[df['cost_of_the_order'] > 20]

# total orders where the cost is above 20 dollars

print('The number of total orders that cost above 20 dollars is:', Q9.shape[0])

percentage = (Q9.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 %
```

In [33]:
```
# second way to calculate it to confirm

df.loc[df.cost_of_the_order > 20.0,'cost_of_the_order'].count()
```

Out[33]:
```
555
```

Observations:

- There are a total of 555 orders that cost above 20 dollars.
- The percentage of such orders in the dataset is around 29.24%.

Question 10: Write the code to find the mean delivery time based on this dataset.

In [34]:
```
round(df['delivery_time'].mean(), 2)
```

Out[34]:
```
24.16
```

Observations:

- Cross checking against Question 4's describe function, the mean delivery time is ~ 24.16 minutes.

Question 11: Suppose the company has decided to give a free coupon of 15 dollars to the customer who has spent the maximum amount on a single order. Write the code to find the ID of the customer along with the order details.

In [35]:
```
df.groupby(by = ["cost_of_the_order"])['order_id'].max()
```

Out[35]:
```
cost_of_the_order
4.47     1477349
```

```
4.66     1476877
4.71     1477788
4.75     1477790
4.80     1477787
          ...
33.22    1478143
33.32    1478329
33.37    1477700
34.19    1477665
35.41    1477814
Name: order_id, Length: 312, dtype: int64
```

In [36]:
```
# display details for order with the maximum cost

df.loc[df['order_id'] == 1477814]
```

Out[36]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
573	1477814	62359	Pylos	Mediterranean	35.41	Weekday	4	21	29

In [37]:
```
# second way to calculate to confirm

df[df['cost_of_the_order'] == df['cost_of_the_order'].max()]
```

Out[37]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
573	1477814	62359	Pylos	Mediterranean	35.41	Weekday	4	21	29

Observations:

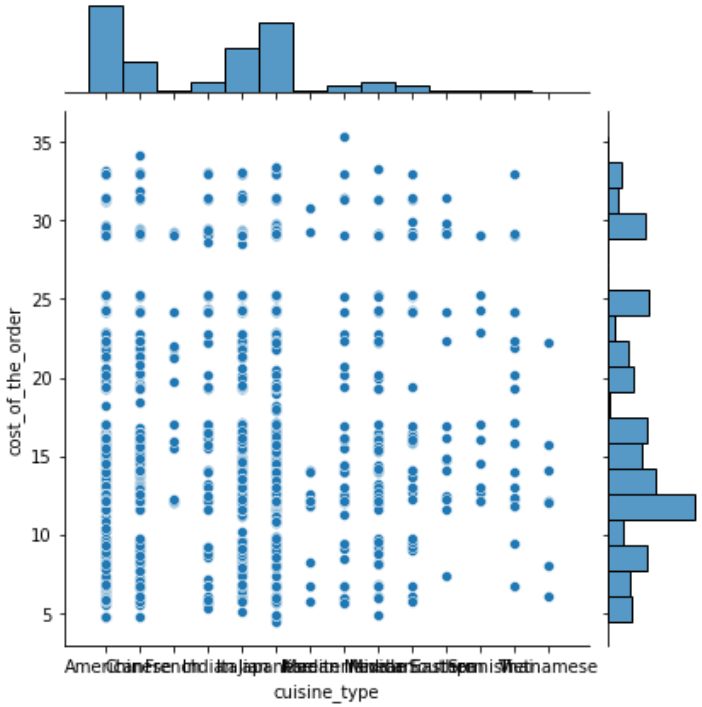
- The customer_id of the customer who has spent the maximum amount on a single order is '62359'.
- The order details are:

> The order_id is '1477814'.
>
> The customer ordered at 'Pylos' which is a Mediterranean restaurant.
>
> The cost of the order was around 35 dollars.
>
> The order was placed on a weekend.
>
> The food preparation time and delivery time for the order were 21 minutes and 29 minutes respectively.
>
> The rating given by the customer is 4.

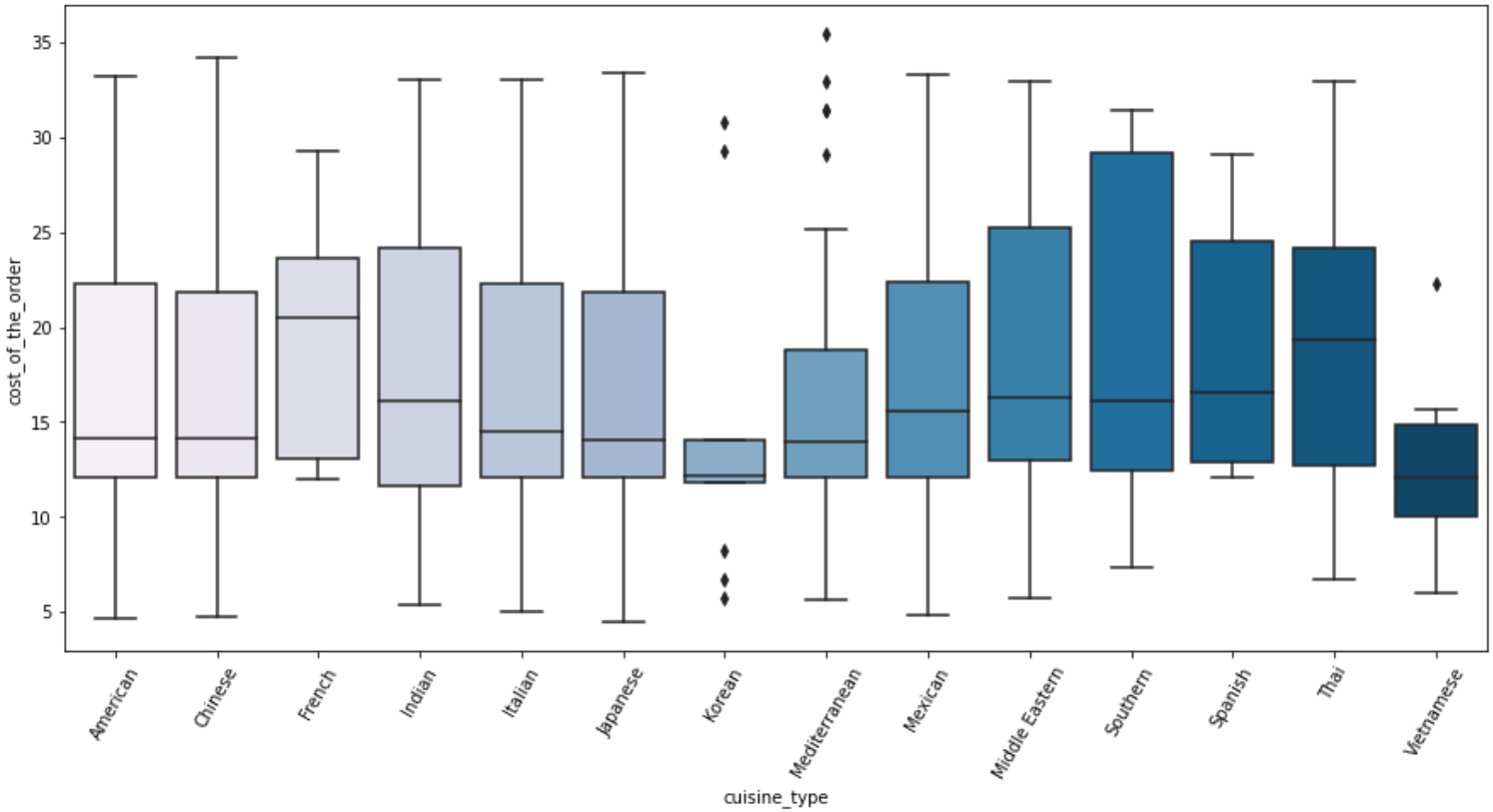
Question 12: Perform bivariate/multivariate analysis to explore relationships between the important variables in the dataset. (MULTIVARIATE ANALYSIS)

In [38]:
```
# numerical variables to explore:  cost, food prep time, delivery tine
# categorical from above: day of week, rating, cuisine type, restaurant

# Relationship between cost of the order and cuisine type

plt.figure(figsize=(15,7))
sns.jointplot(x = "cuisine_type", y = "cost_of_the_order", data = df, palette = 'PuBu')
plt.show()
```

```
<Figure size 1080x504 with 0 Axes>
```



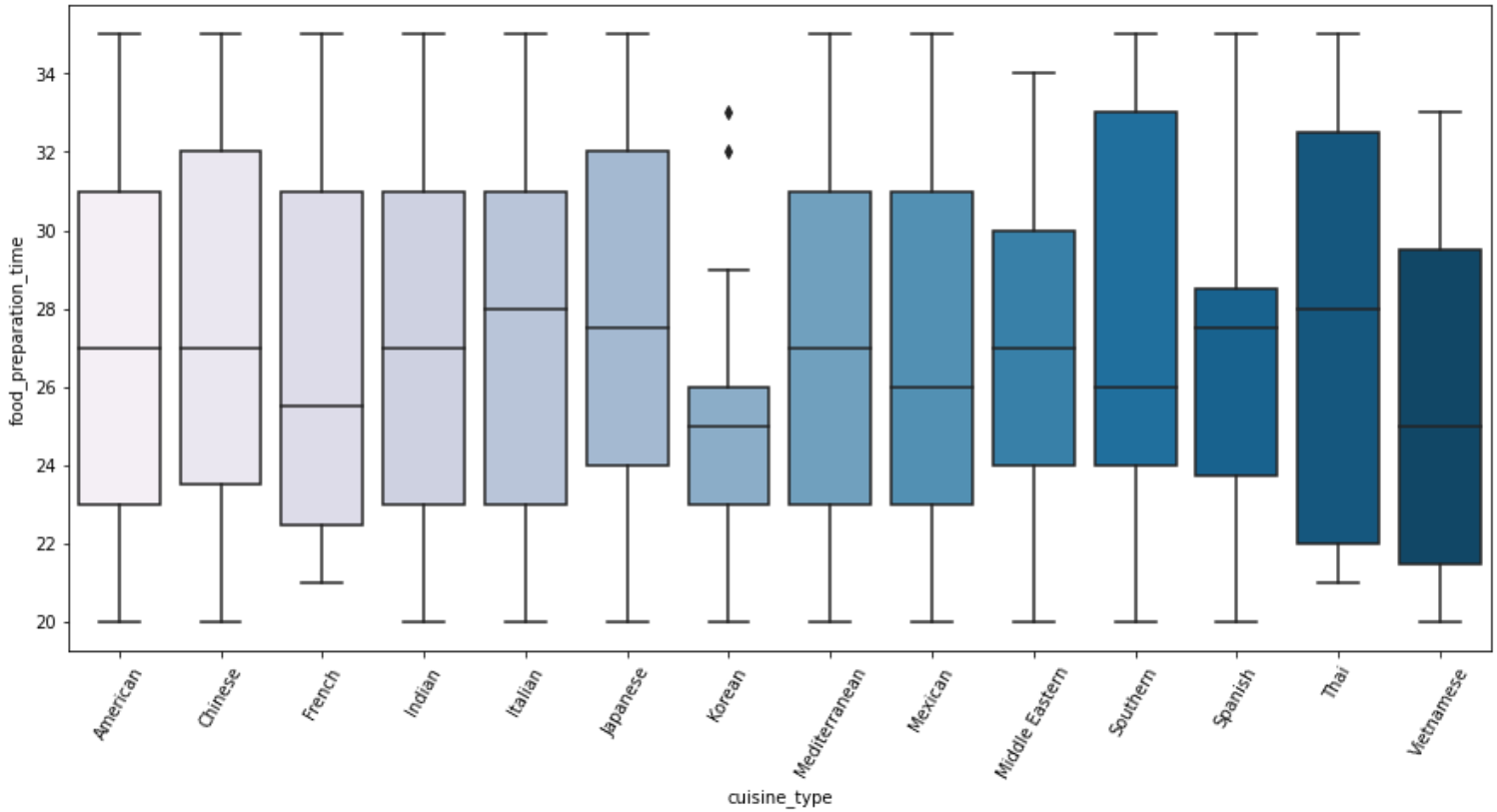
In [39]:
```
# Relationship between cost of the order and cuisine type

plt.figure(figsize=(15,7))
sns.boxplot(x = "cuisine_type", y = "cost_of_the_order", data = df, palette = 'PuBu')
plt.xticks(rotation = 60)
plt.show()
```



Observations:

- Vietnamese and Korean cuisines cost less compared to other cuisines.
- The boxplots for Italian, American, Chinese, Japanese cuisines are quite similar. This indicates that the quartile costs for these cuisines are quite similar.
- Outliers are present for the cost of Korean, Mediterranean and Vietnamese cuisines.
- French and Spanish cuisines are costlier compared to other cuisines.

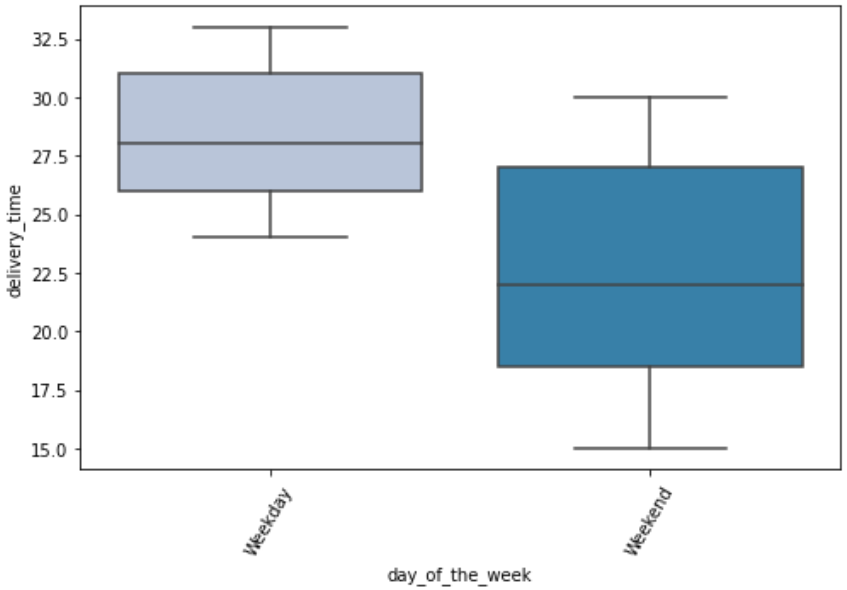
In [40]:
```
# Relationship between food preparation time and cuisine type

plt.figure(figsize=(15,7))
sns.boxplot(x = "cuisine_type", y = "food_preparation_time", data = df, palette = 'PuBu')
plt.xticks(rotation = 60)
plt.show()
```



Observations:

- Food preparation time is very consistent for most of the cuisines.
- The median food preparation time lies between 24 and 30 minutes for all the cuisines.
- Outliers are present for the food preparation time of Korean cuisine.
- Korean cuisine takes less time compared to the other cuisines.

In [41]:
```
# Relationship between day of the week and delivery time

plt.figure(figsize=(8,5))
sns.boxplot(x = "day_of_the_week", y = "delivery_time", data = df, palette = 'PuBu')
plt.xticks(rotation = 60)
plt.show()
```



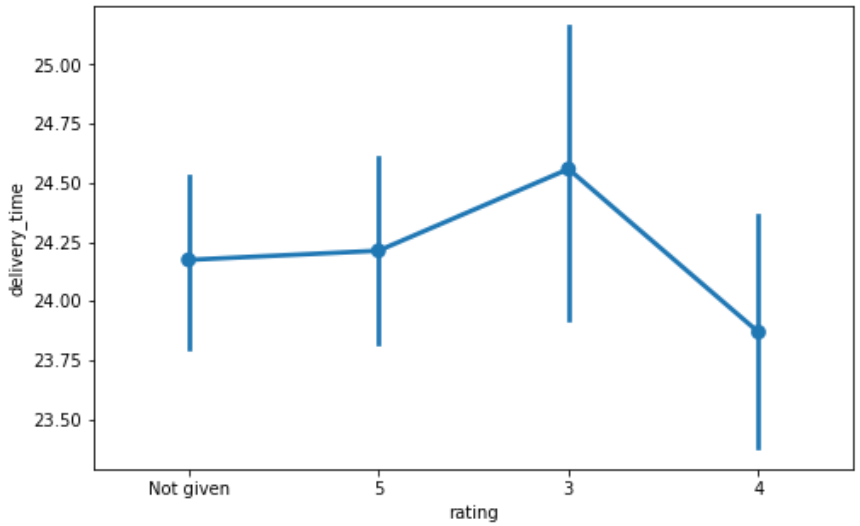
Observations:

- The delivery time for all the orders over the weekends is less compared to weekdays. This could be due to the dip in traffic over the weekends.

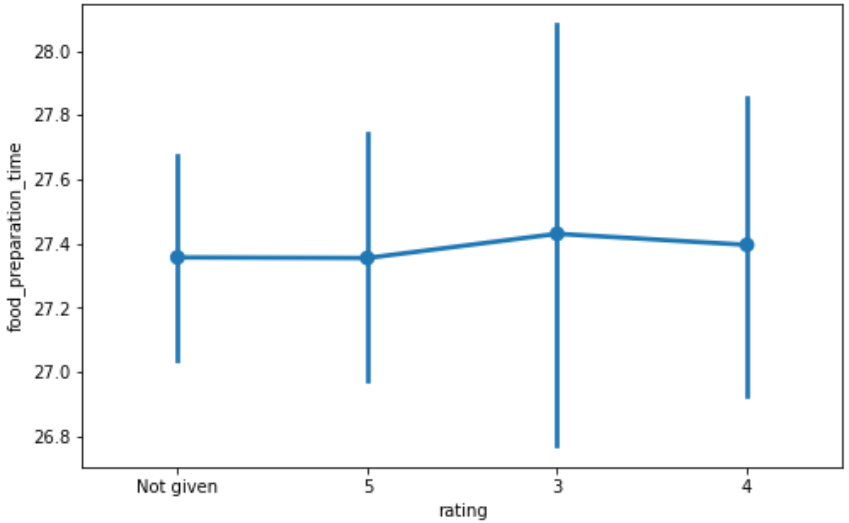
In [42]:
```
# Calculate revenue generated by top 14 restaurants (ALL GENERATE OVER $500):

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

Out[42]:
```
restaurant_name
Shake Shack                      3579.53
The Meatball Shop                2145.21
Blue Ribbon Sushi                1903.95
Blue Ribbon Fried Chicken        1662.29
Parm                             1112.76
RedFarm Broadway                  965.13
RedFarm Hudson                    921.21
TAO                               834.50
Han Dynasty                       755.29
Blue Ribbon Sushi Bar & Grill     666.62
Rubirosa                          660.45
Sushi of Gari 46                  640.87
Nobu Next Door                    623.67
Five Guys Burgers and Fries       506.47
Name: cost_of_the_order, dtype: float64
```

In [43]:
```
# Relationship between rating and delivery time

plt.figure(figsize=(8, 5))
sns.pointplot(x = 'rating', y = 'delivery_time', data = df)
plt.show()
```



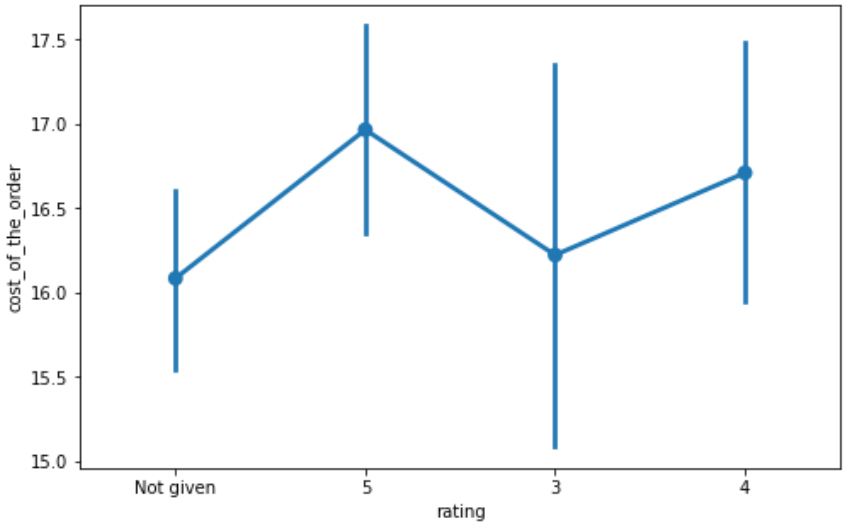
In [44]:
```
# Relationship between rating and food preparation time

plt.figure(figsize=(8, 5))
sns.pointplot(x = 'rating', y = 'food_preparation_time', data = df)
plt.show()
```



Observations:

- It seems like delivery time does play a role in low ratings, but food preparation time does not influence them.

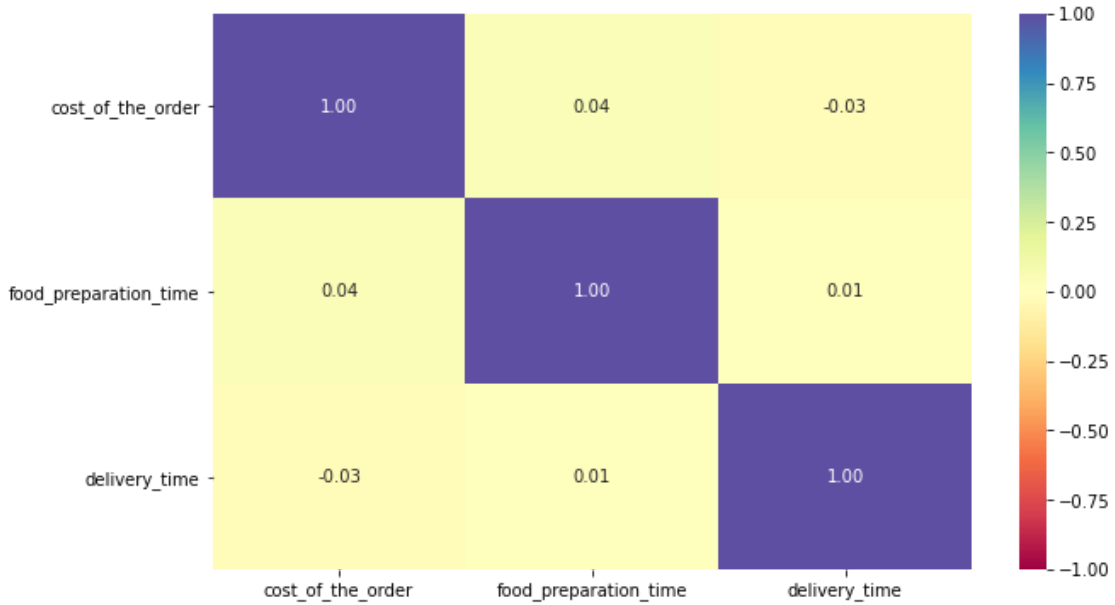
In [45]:
```
# Relationship between rating and cost of the order

plt.figure(figsize=(8, 5))
sns.pointplot(x = 'rating', y = 'cost_of_the_order', data = df)
plt.show()
```



Observations:

- It seems that high-cost orders have been rated well and low-cost orders have not been rated.

In [46]:
```
# plot the heatmap: 3 numerical variables, correlations ___ No correlation found between 3 variables.

col_list = ['cost_of_the_order', 'food_preparation_time', 'delivery_time']
plt.figure(figsize=(10, 6))
sns.heatmap(df[col_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectral")
plt.show()
```

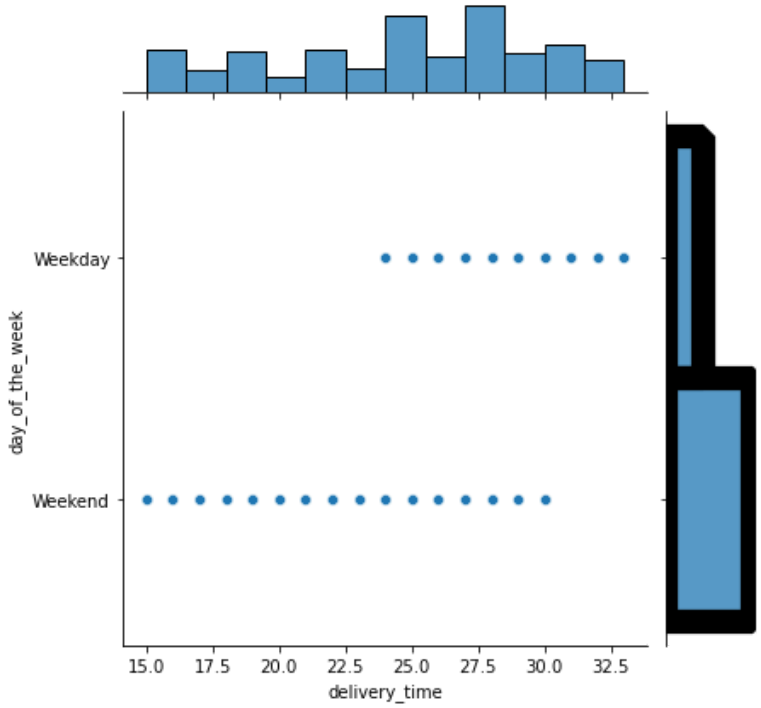


In [47]:
```
# Want to create a new dataframe only using data from the 5 restaurants with the most orders
# Look at Question 7 for top 5

df2 = df[df["restaurant_name"] == "Shake Shack" ]
df3 = df[df["restaurant_name"] == "The Meatball Shop"]
df4 = df[df["restaurant_name"] == "Blue Ribbon Sushi"]
df5 = df[df["restaurant_name"] == "Blue Ribbon Fried Chicken"]
df6 = df[df["restaurant_name"] == "Parm"]
```

In [48]:
```
# Merge 5 dataframes from above to create subset of data from the 5 restaurants with the most orders

top_restaurants = pd.concat([df2, df3, df4, df5, df6], axis = 0, join = 'inner')
```

In [49]:
```
(sns.jointplot(data=top_restaurants , x='delivery_time', y='day_of_the_week'))
```

Out[49]:
```
<seaborn.axisgrid.JointGrid at 0x7f95720757c0>
```



Question 13: Suppose 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. Write the code to find the restaurants fulfilling the criteria to get the promotional offer.

From Question 5, 736 orders are not rated out of 1898. To clean data, drop orders that have "Not given" as rating.

In [50]:
```
df.drop(df.loc[df['rating']=="Not given"].index, inplace=True)
```

In [51]:
```
# Convert rating variable into an integer
df['rating'] = df.rating.astype('int64')
```

In [52]:
```
# create a dataframe that contains the restaurant names with their rating counts
Q13 = df.groupby(['restaurant_name'])['rating'].count().sort_values(ascending = False).reset_index()
Q13.head()
```

Out[52]:

	restaurant_name	rating
0	Shake Shack	133
1	The Meatball Shop	84
2	Blue Ribbon Sushi	73

	restaurant_name	rating
3	Blue Ribbon Fried Chicken	64
4	RedFarm Broadway	41

In [53]:
```
# get names of restaurants with more than 50 ratings

Q13b = Q13[Q13['rating'] > 50]['restaurant_name']

# filter so that it only displays those restaurants with more than 50 ratings

finalQ = df[df['restaurant_name'].isin(Q13b)].copy()

# find the mean rating of the restaurants - reset index at end is so it displays neater

finalQ.groupby(finalQ['restaurant_name'])['rating'].mean().sort_values(ascending = False).reset_index()[:5]
```

Out[53]:

	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
4	'wichcraft	NaN

In [54]:
```
# SECOND WAY TO DO QUESTION 13:


# creating a new set filtering restaurant names by number of orders

q_thirteen = df['restaurant_name'].value_counts()

# creating a new set filtering restaurant names by average rating

thirteen = df.groupby('restaurant_name')['rating'].mean()
```

In [55]:
```
# using 'loc' to filter so that it only displays restaurants ratings above 50, with an average rating above 4

result = pd.concat([q_thirteen.loc[q_thirteen > 50], thirteen.loc[thirteen > 4.0]], axis=1, join='inner')
display(result)
```

	restaurant_name	rating
Shake Shack	133	4.278195

	restaurant_name	rating
The Meatball Shop	84	4.511905
Blue Ribbon Sushi	73	4.219178
Blue Ribbon Fried Chicken	64	4.328125

Observations:

- The restaurants fulfilling the criteria to get the promotional offer are: **'The Meatball Shop', 'Blue Ribbon Fried Chicken', 'Shake Shack' and 'Blue Ribbon Sushi'**.

Question 14: Suppose 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. Write the code to find the net revenue generated on all the orders given in the dataset.

In [56]:
```
# Reset index because we dropped values in Question 13 - replaces rows where rating = "not given"
# and switches all variables back to original type (Inplace = True so it doesn't return dataframe after reset)

df.reset_index(inplace=True)
```

In [57]:
```
# define cost for if statements below

cost = df['cost_of_the_order']

# check sum of all orders in dataset

cost.sum()
```

Out[57]: 19476.01

In [58]:
```
def get_revenue(cost):                    # define function to calculate revenue with cost as the variable                    #
    revenue = 0                           # initialize revenue as 0
    for c in cost:                        # for loop to establish conditions later for calculations
        if (c > 5 and c < 20):            # 15% revenue for orders $5-$20
            revenue += c * 0.15           # iterate: keeps going until end of dataset
        elif(c > 20):                     # 25% revenue for orders over $20
            revenue += c * 0.25           # iterate: keeps going until end of dataset
    return revenue

print(round(get_revenue(cost),2))                  # print final sum of revenue after all iterations are complete.
```

```
3865.57
```

In [59]:
```
# SECOND WAY TO DO QUESTION 14:

# add a new column named revenue to the dataframe df that stores what the company charges for each order
```

```
df['revenue'] = [order_cost * 0.25 if order_cost > 20 else
                         order_cost * 0.15 if order_cost > 5 else
                         0
                         for order_cost in df['cost_of_the_order']]

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

```
The net revenue is around 3865.57 dollars
```

Question 15: Suppose the company wants to analyze the total time required to deliver the food. Write the code to find out the percentage of orders that have more than 60 minutes of total delivery time.

In [60]:
```
# Add new column to dataframe to store total time

df['total_time'] = df['food_preparation_time'] + df['delivery_time']

# count number of orders with total time over 60

df[df['total_time'] > 60].count()[:1]
```

Out[60]:
```
index    119
dtype: int64
```

In [61]:
```
# find the percentage of orders that have more than 60 minutes of total delivery time

# find the percentage of orders that have more than 60 minutes of total delivery time
print ('The percentage of orders that have more than 60 minutes of total delivery time is',
       round(df[df['total_time'] > 60].shape[0] / df.shape[0] * 100, 2),'%')
```

```
The percentage of orders that have more than 60 minutes of total delivery time is 10.24 %
```

Question 16: Suppose the company wants to analyze the delivery time of the orders on weekdays and weekends.Write the code to find the mean delivery time on weekdays and weekends. Write your observations on the results.

In [62]:
```
round(df.groupby('day_of_the_week')['delivery_time'].mean(), 2)
```

Out[62]:
```
day_of_the_week
Weekday    28.31
Weekend    22.44
Name: delivery_time, dtype: float64
```

Observations:

- The mean delivery time on weekdays is around 28 minutes whereas the mean delivery time on weekends is around 22 minutes.
- This could be due to the dip of traffic volume in the weekends.

Conclusions:

- Around 80% of the orders are for American, Japanese, Italian and Chinese cuisines. Thus, it seems that these cuisines are quite popular among customers of FoodHub.
- Shake Shack is the most popular restaurant that has received the highest number of orders.
- Order volumes increase on the weekends compared to the weekdays.
- Delivery time over the weekends is less compared to the weekdays. This could be due to the dip in traffic volume over the weekends.
- Around 39% of the orders have not been rated.

Business Recommendations:

- FoodHub should integrate with restaurants serving American, Japanese, Italian and Chinese cuisines as these cuisines are very popular among FoodHub customers.
- FoodHub should provide promotional offers to top-rated popular restaurants like Shake Shack that serve most of the orders.
- As the order volume is high during the weekends, more delivery persons should be employed during the weekends to ensure timely delivery of the order. Weekend promotional offers should be given to the customers to increase the food orders during weekends.
- Customer Rating is a very important factor to gauge customer satisfaction. The company should investigate the reason behind the low count of ratings. They can redesign the rating page in the app and make it more interactive to lure the customers to rate the order.
- Around 11% of the total orders have more than 60 minutes of total delivery time. FoodHub should try to minimize such instances in order to avoid customer dissatisfaction. They can provide some reward to the punctual delivery persons.