Project Foundations for Data Science: FoodHub Data Analysis

Marks: 60

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: 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 [ ]:
```
# 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

# import google drive, necessary to load/read file
from google.colab import drive
drive.mount("/content/drive")
```

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

Understanding the structure of the data

In [ ]:
```
# read the data
df = pd.read_csv('/content/drive/MyDrive/MIT Data Science & AI Course Notes/Week 1: Py

#Create a new column adding the food prep time and delivery time, called total time (f
df['total_time'] = df['food_preparation_time'] + df['delivery_time']

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

Out[ ]:

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

Observations:

The DataFrame has 9 columns as mentioned in the Data Dictionary. Data in each row corresponds to the order placed by a customer.

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

In [ ]:
```
#By running df.shape, we can see how many rows and columns are in the dataset
df.shape
```

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

Observations:

- 1897 Rows
- 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 [ ]:
```
# 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
```

Observations:

order_id = integer

customer_id = integer

restaurant_name = object

cuisine_type = object

cost_of_the_order = float

day_of_the_week = object

rating = object

food_preparation_time = integer

deliver_time = integer

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

In [ ]:
```
#Check for missing values in the data
df.isnull().sum()

#Replace "not given" with nan
df["rating"] = df["rating"].replace(["Not given"], np.nan)

#We can then change the data type of this column to float
df["rating"] = df["rating"].astype(float)
#Run df.head() to ensure the values were replaced
df.head()
```

Out[ ]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	f
0	1477147	337525	Hangawi	Korean	30.75	Weekend	NaN	
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	NaN	
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5.0	
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3.0	
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4.0	

Observations:

There are no explicitly "missing" values, seen by running df.info(), but by looking at the data set we can see that in the rating column, there are some values of "not given". We can replace these using

df["rating"] = df["rating"].replace(["Not given"], np.nan)

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 [ ]:
```
#We can use df.describe() to check the statistical summary of the data
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	
order_id	1898.0	1.477496e+06	548.049724	1476547.00	1477021.25	1477495.50	1.477
customer_id	1898.0	1.711685e+05	113698.139743	1311.00	77787.75	128600.00	2.705
cost_of_the_order	1898.0	1.649885e+01	7.483812	4.47	12.08	14.14	2.229
rating	1162.0	4.344234e+00	0.741478	3.00	4.00	5.00	5.000
food_preparation_time	1898.0	2.737197e+01	4.632481	20.00	23.00	27.00	3.100
delivery_time	1898.0	2.416175e+01	4.972637	15.00	20.00	25.00	2.800

Observations:

Minimum: 20 minutes

Average: Roughly 27.37 minutes

Maximum: 35 minutes

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

In [ ]:
```
#We can use df.isna().sum() to find this information because of our replacement earlie
df.isna().sum()
```

Out[ ]:
```
order_id                   0
customer_id                0
restaurant_name            0
cuisine_type               0
cost_of_the_order          0
day_of_the_week            0
rating                   736
food_preparation_time      0
delivery_time              0
dtype: int64
```

Observations:

736 Orders

Exploratory Data Analysis (EDA)

Univariate Analysis

Order_ID

We can see that there are 1898 unique orders

In [ ]:
```
#We can check how many unique order id's there are
df['order_id'].nunique()
```

Out[ ]:
```
1898
```

Customer_ID

We can see how many customers have placed orders (1200)

In [ ]:
```
#We can check how many unique order id's there are
df['customer_id'].nunique()
```

Out[ ]:
```
1200
```

Restaurant names

We can see how many restaurants have received orders (178)

In [ ]:
```
df['restaurant_name'].nunique()
```

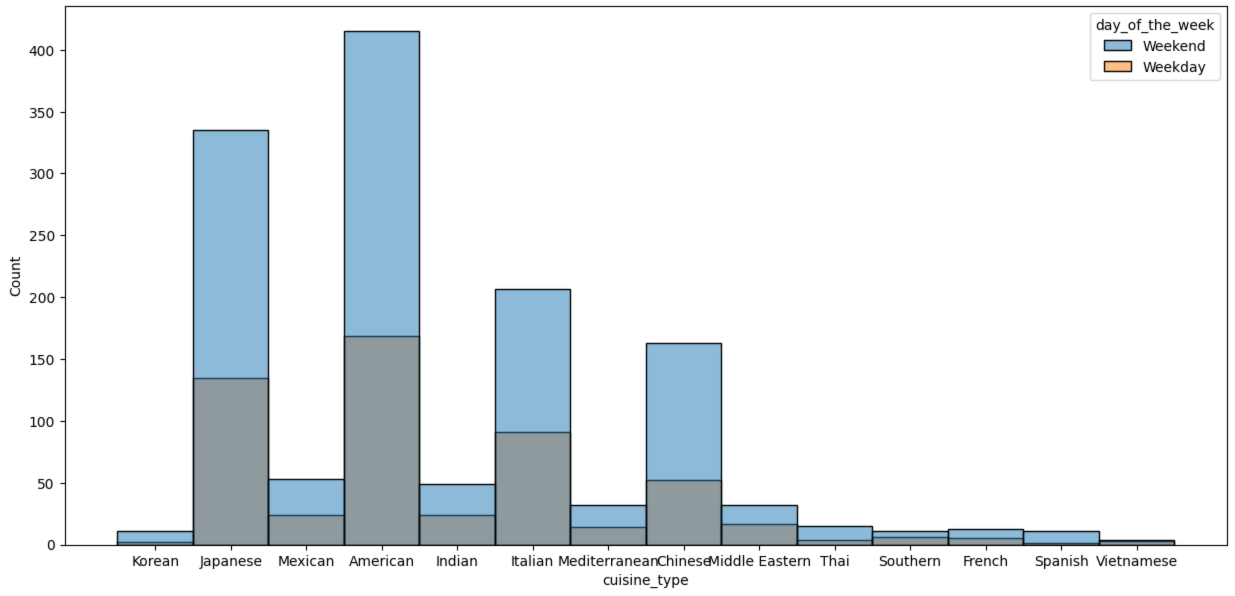
Out[ ]:
```
178
```

Cuisine Type

We can see there are 14 types of cuisines have been ordered by the customers , and plot it in a countplot.

In [ ]:
```
print(df['cuisine_type'].nunique())
plt.figure(figsize = (15, 7))
sns.histplot(data = df, x = "cuisine_type", hue = "day_of_the_week")
```

Out[ ]:
```
14
<Axes: xlabel='cuisine_type', ylabel='Count'>
```

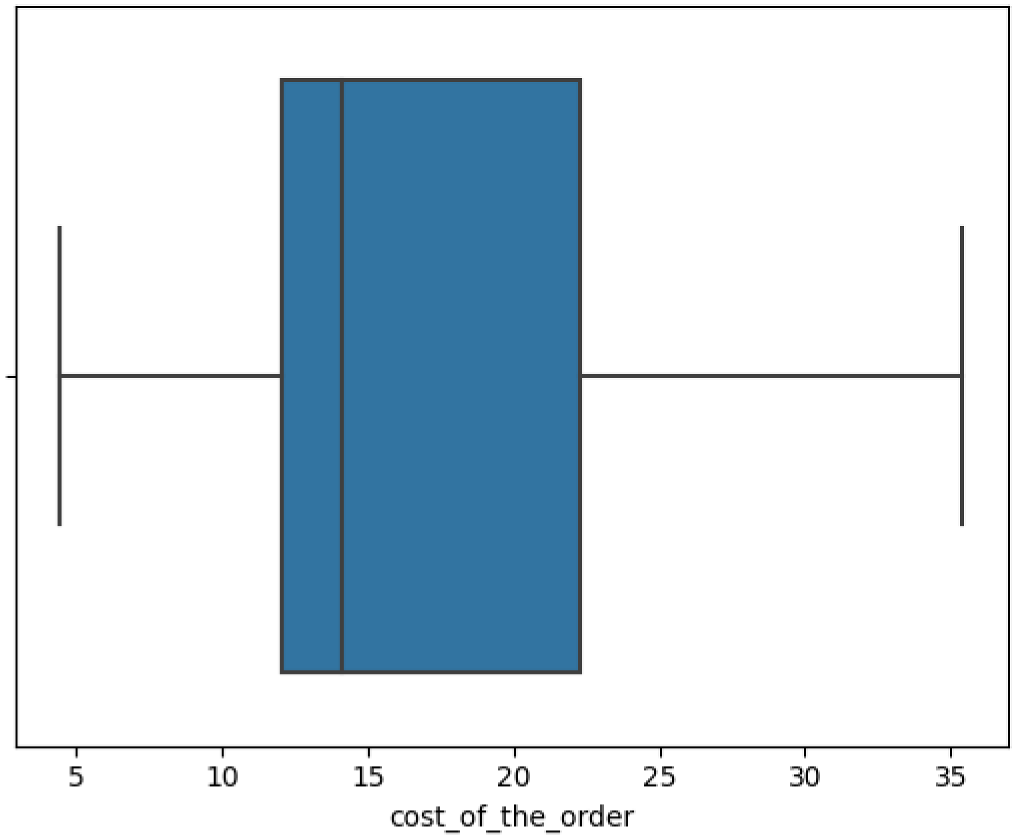


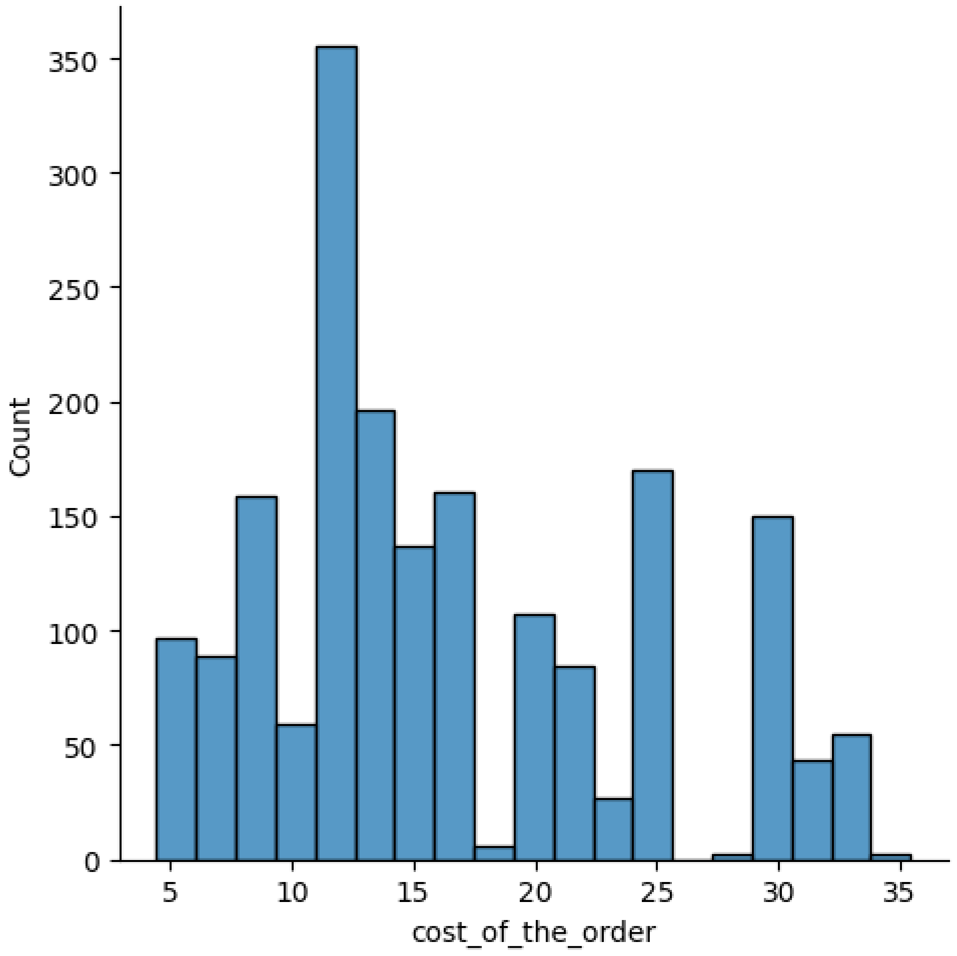
Cost Of the Order

We can see plot order costs in boxplot and displot

In [ ]:
```
sns.boxplot(data=df, x = "cost_of_the_order")
plt.show()
```

```
sns.displot(data=df, x = "cost_of_the_order")
plt.show()
```

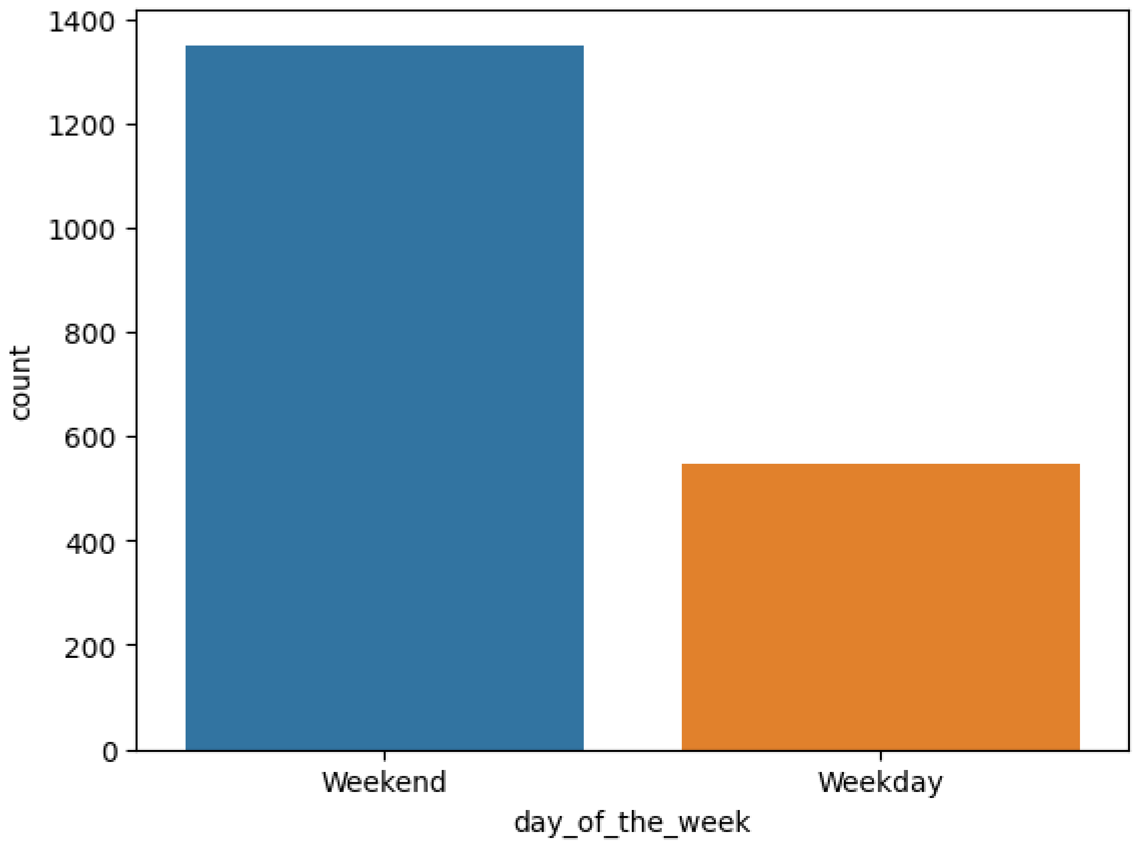




Day Of The Week

We can see how many people ordered on weekdays vs weekends by plotting it in a countplot. There are significantly more orders on the weekends than weekdays

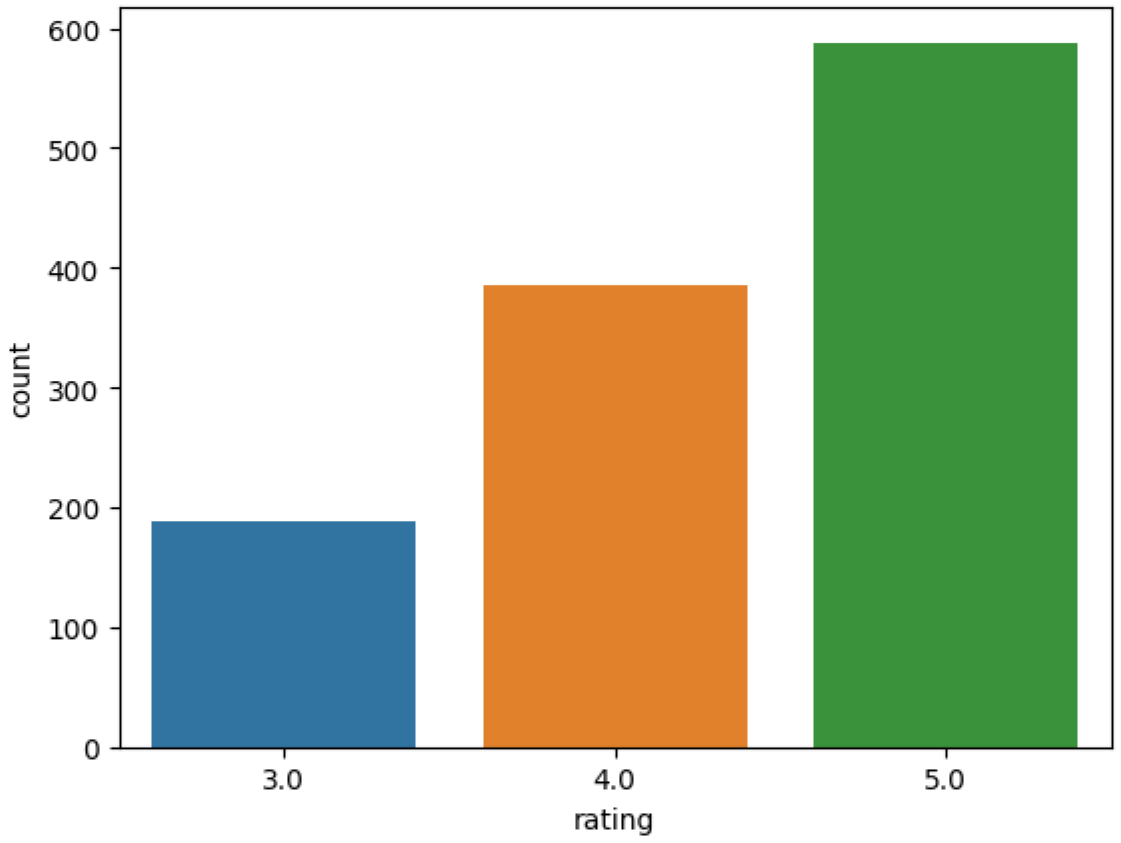
```
In [ ]: sns.countplot(data = df, x = 'day_of_the_week')
        plt.show()
```



Rating

We can see ratings in a countplot as well

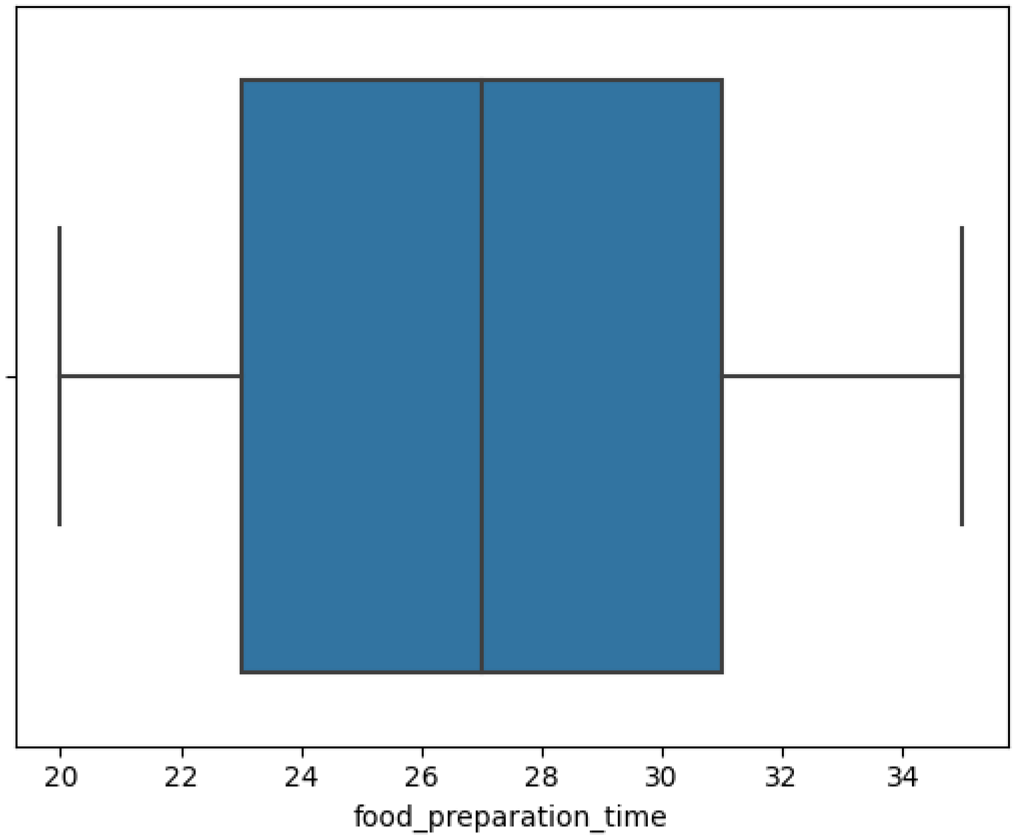
```
In [ ]: sns.countplot(data = df, x = 'rating')
        plt.show()
```

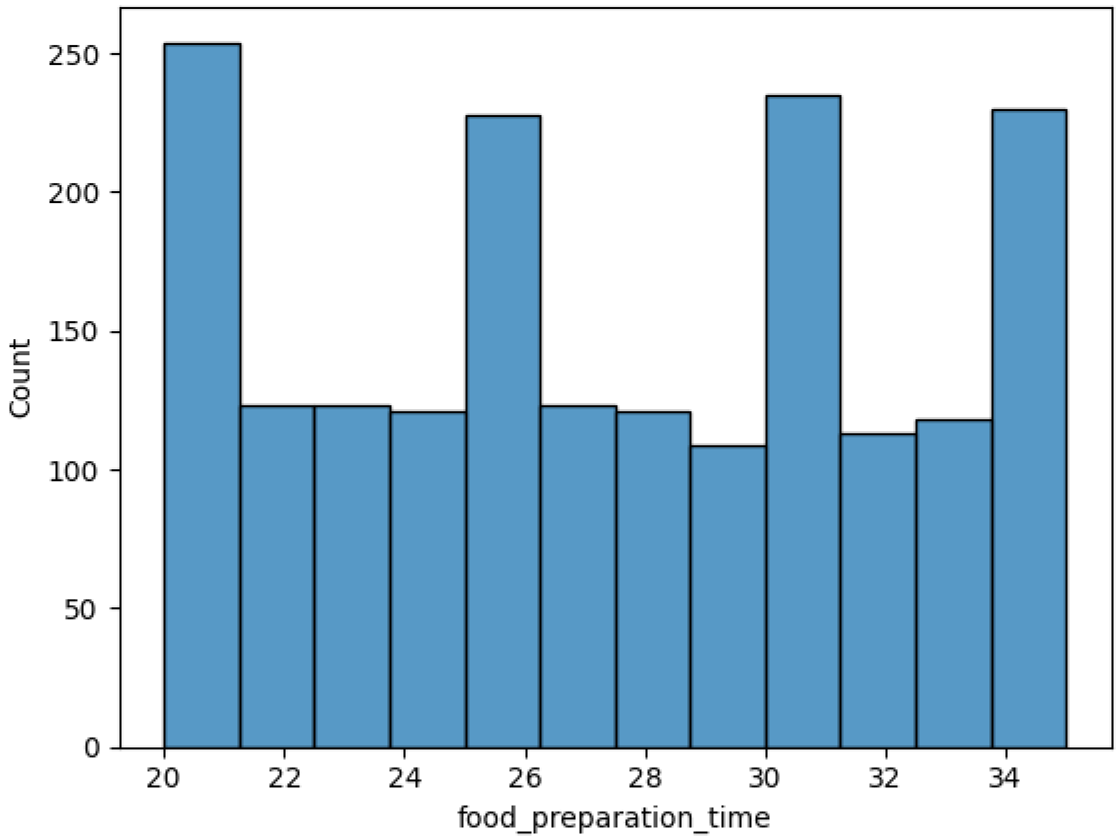


Food Preparation Time

We can see plot food preparation time in both a histplot and a boxplot

In [ ]:
```
sns.boxplot(data=df, x = "food_preparation_time")
plt.show()
sns.histplot(data=df, x = "food_preparation_time")
plt.show()
```

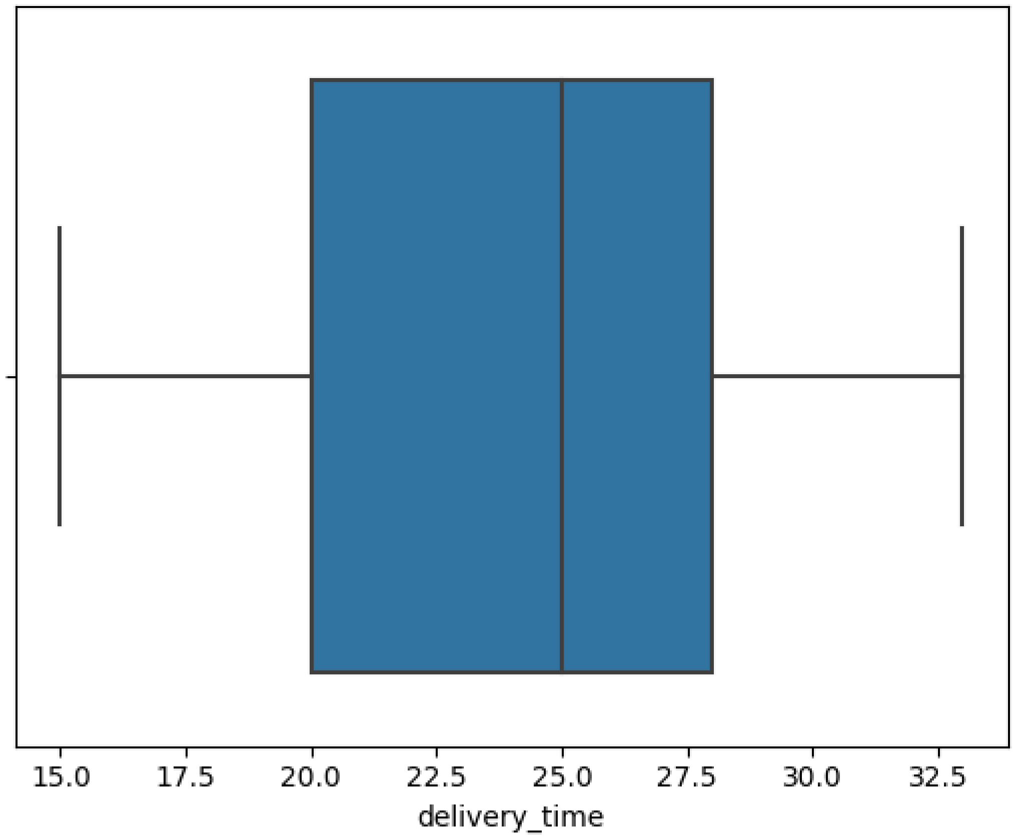


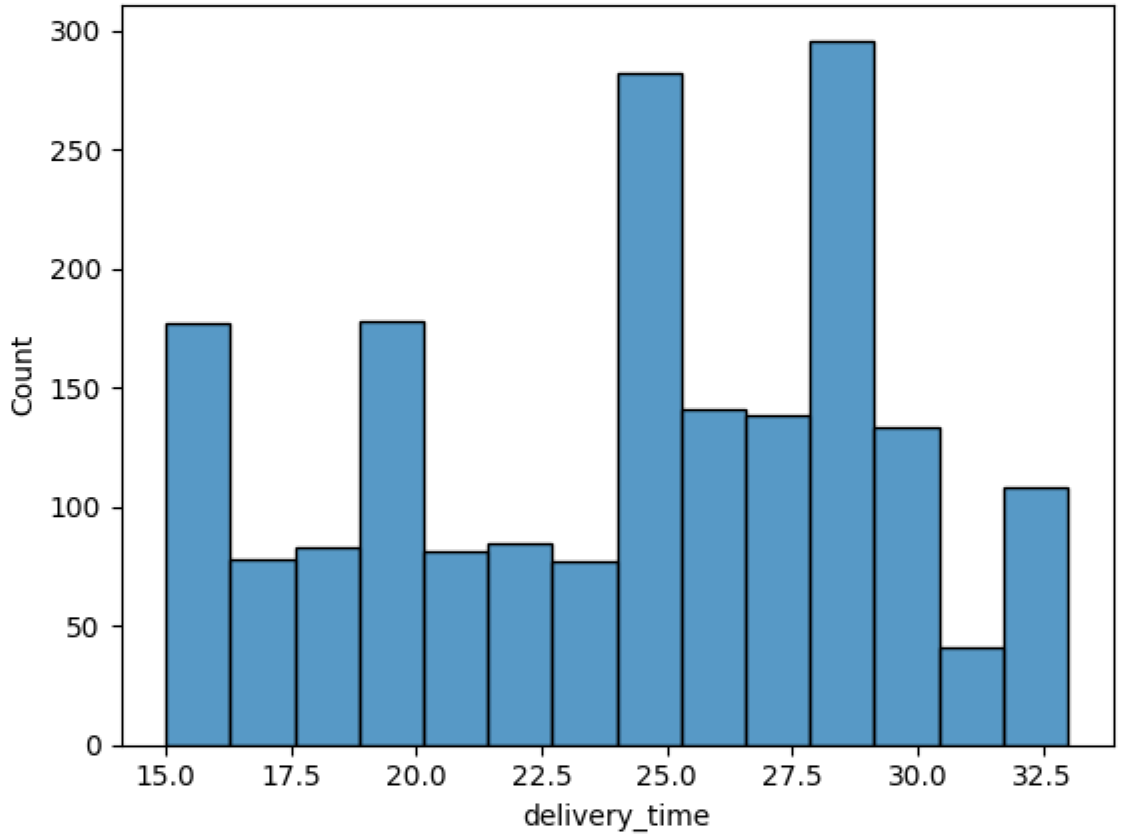


Delivery Time

We can see plot delivery time in both a histplot and a boxplot

In [ ]:
```
sns.boxplot(data=df, x = "delivery_time")
plt.show()
sns.histplot(data=df, x = "delivery_time")
plt.show()
```





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

customer_id:

We can see that the boxplot and histplot show us that the data is skewed towards the right, and there are few, if any, outliers. The countplot is much more useful, showing us that most customers only placed one order, fewers placed two, and fewer still placed more. Someone placed as many as 7 orders, while a few placed 5 and two placed 6.

restaurant_name:

We can see that most restaurants have less than 50 orders, although there are a few that have between 50-100 orders, and only one that has more than 150 orders (Shake Shack has 219 orders).

cuisine_type:

Looking at the histogram, we can see that American cuisine is overwhelmingly the most ordered food, at just under 600 orders. Japanese, Italian and Chinese are the only other cuisines with over 100 orders, with just under 500, just under 300, and just over 200 orders, respectively. All other cuisines have less than 100 order.

cost_of_the_order

Looking at the box plot and histogram, we can see that most orders fall between 10-25 dollars, although there are some outliers with few orders that are cheaper than 10 and some that are 30 or more expensive. To this end, we can see that the data is skewed to the right.

day_of_the_week

Looking at a histogram, we can see that people overwhelmingly order more food on the weekends, with more than double the orders on weekends than weekdays (under 1400 orders on weekends vs under 600 orders on weekdays)

rating

By looking at a histogram, we can see that most restaurants got 5 star ratings, with just under 600 orders having 5 stars. Just under 400 orders got 4 stars, and just under 200 orders got 3 stars, and no restaurant got a rating lower than 3.

food_preparation_time

We can see by looking at both histograms and boxplots that the most orders take either 20-21 minutes, 25-26 minutes, 30-31 minutes or 33-34 minutes, and fewer orders fall in between those intervals. The median is around 27 minutes.

delivery_time

We can see by looking at both the histogram and box plot that these graphs are positively skewed to the left, with most orders taking 24+ minutes to deliver and the average being ~25 minutes. The orders range from taking as little as 15 minutes to as long as almost 33 minutes. The most amount of orders take, in order: 28 minutes 24-25 minutes 19 minutes 15-16 minutes

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

In [ ]:
```
# We can see that the plot is rather cluttered, so we can use value_counts to
# get more useful info
df["restaurant_name"].value_counts()
```

Out[ ]:
```
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
                            ...
Sushi Choshi                   1
Dos Caminos Soho               1
La Follia                      1
Philippe Chow                  1
'wichcraft                     1
Name: restaurant_name, Length: 178, dtype: int64
```

Observations:

Shake Shack: 219

The Meatball Shop: 132

Blue Ribbon Sushi: 119

Blue Ribbon Fried Chicken: 96

Parm: 68

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

In [ ]:
```
# Get most popular cuisine on weekends
df_weekend = df[df['day_of_the_week'] == 'Weekend']
#this will check the unique value counts for cuisines on weekends
df_weekend['cuisine_type'].value_counts()
```

Out[ ]:
```
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
```

Observations: It's clear by looking at the value counts that, similar to the entire data set, American cuisine is the most popular on the weekends, followed by Japanese and Italian.

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

In [ ]:
```
df_greater_than_20 = df[df["cost_of_the_order"]>20]
print(df_greater_than_20.shape[0])
fraction = df_greater_than_20.shape[0] / df.shape[0]
percentage = fraction * 100
print(percentage)
```

```
555
29.24130663856691
```

Observations:

We can see that a total of 555 orders cost more than 20 dollars, and by dividing that by the total number of orders, that the percentage of orders costing more than 20 dollars is roughly 29.24%

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

In [ ]:
```
#By using the df['column_name'].mean() function, we can find this data
df['delivery_time'].mean()
```

Out[ ]:
```
24.161749209694417
```

Observations:

Roughly 24.16 minutes

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 [ ]:
```
#By taking the value counts and getting the top 3 with head(3), we will find the
#3 customers who placed the most orders
```

```
df['customer_id'].value_counts().head(3)
```

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

Observations:

Customer 52832 placed 13 orders

Customer 47440 placed 10 orders

Customerr 83287 placed 9 orders

Multivariate Analysis

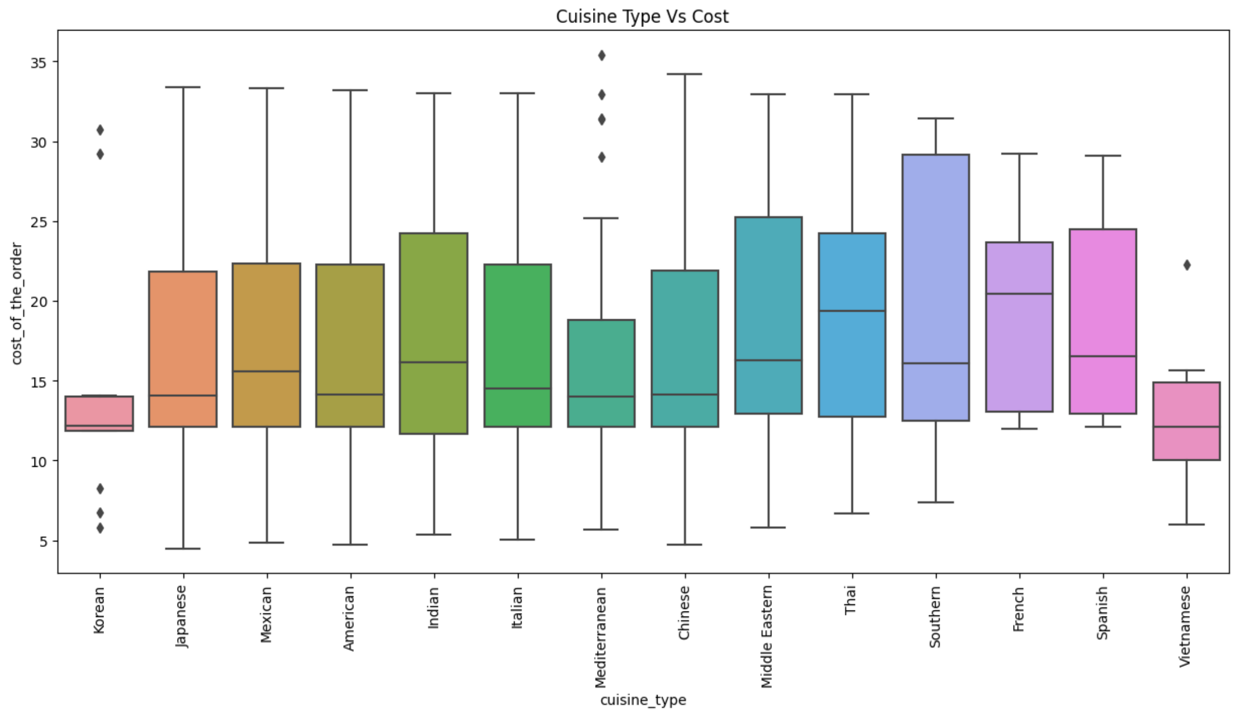
Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. (It is a good idea to explore relations between numerical variables as well as relations between numerical and categorical variables) [10 marks]

Cuisine Vs Cost

We can see the relationship between the cuisine type and cost with a boxplot, which shows us that Korean and Vietnamese meals tend to be the cheapest (with a few outliers), while Southern, Spanish and Italian foods tend to be the most expensive, with Mediterranean food having a few outliers with some of the most expensive meals in the data set.

In [ ]:
```
plt.figure(figsize = (15, 7))
plt.xticks(rotation = 90)
plt.title("Cuisine Type Vs Cost")
sns.boxplot(x = "cuisine_type", y = "cost_of_the_order", data = df)
```

Out[ ]:
```
<Axes: title={'center': 'Cuisine Type Vs Cost'}, xlabel='cuisine_type', ylabel='cost_
of_the_order'>
```

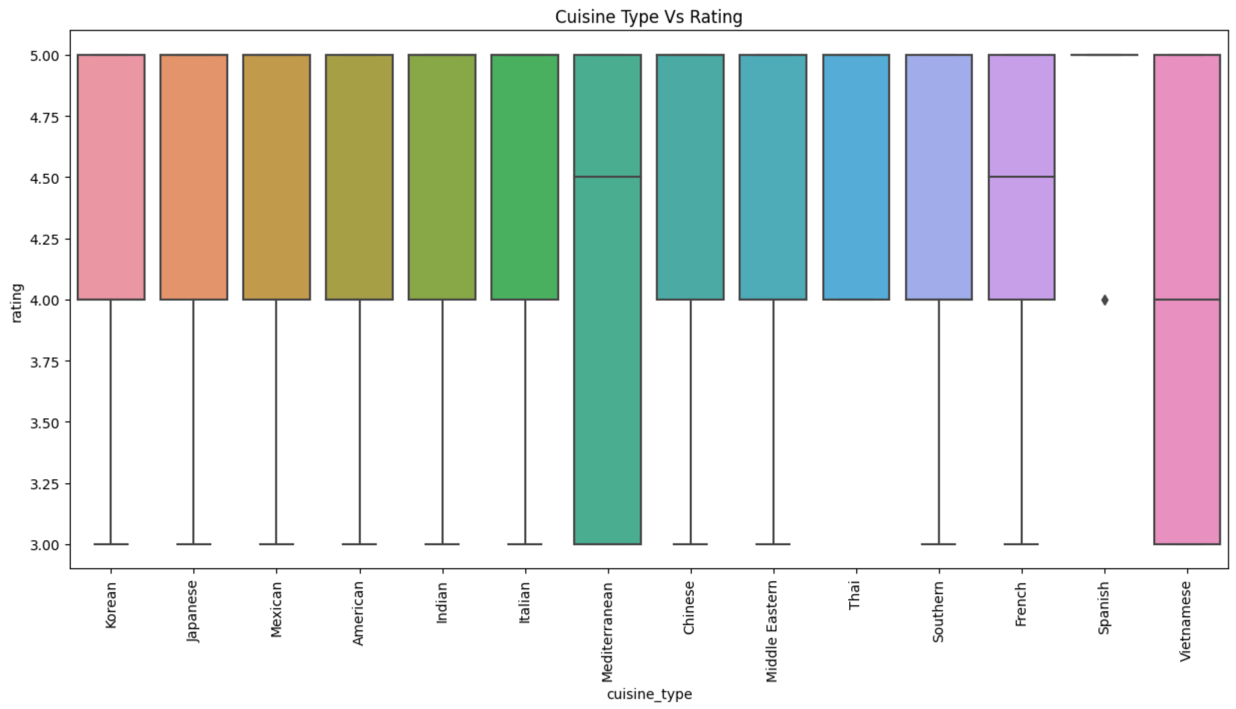


Cuisine Vs Rating

We can see the relationship between the cuisine type and rating with a histogram. We see that most cuisines have only 4 or 5 star ratings, with only Mediterranean and Vietnamese cuisines having any 3 star ratings at all.

In [ ]:
```python
plt.figure(figsize = (15, 7))
plt.xticks(rotation = 90)
plt.title("Cuisine Type Vs Rating")
sns.boxplot(data = df, x = "cuisine_type", y = "rating")
```

Out[ ]:
```
<Axes: title={'center': 'Cuisine Type Vs Rating'}, xlabel='cuisine_type', ylabel='rat
ing'>
```

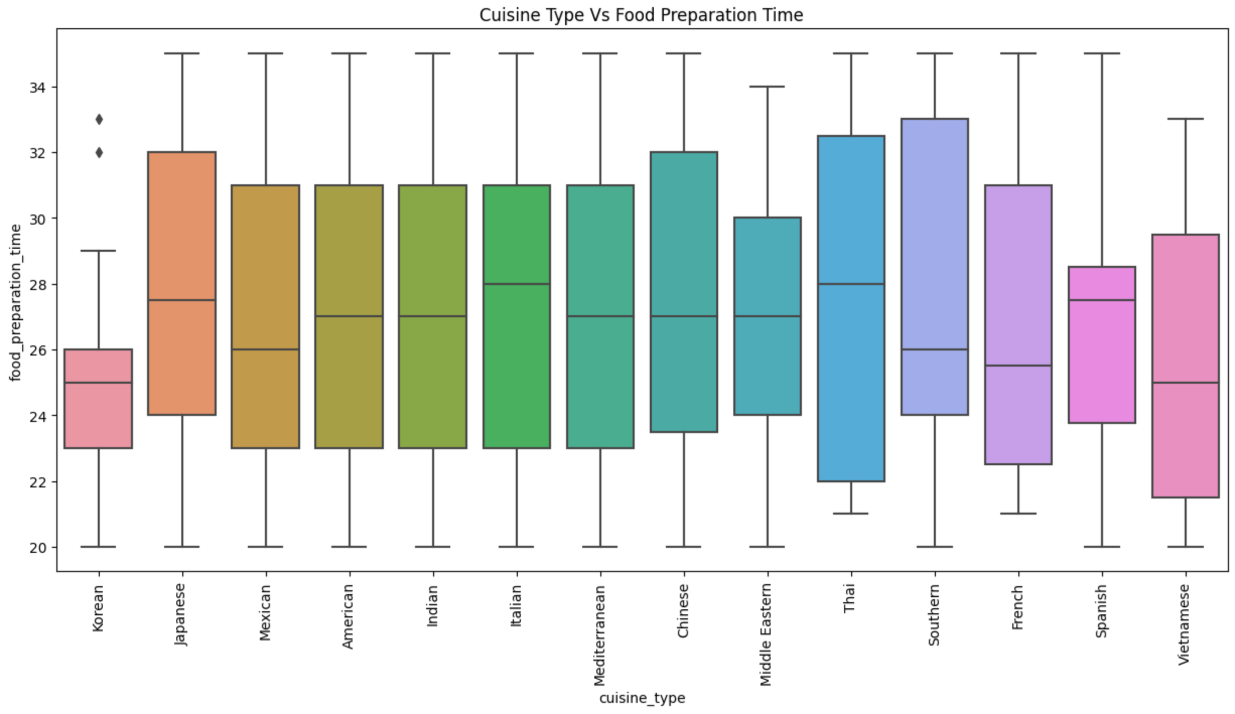


Cuisine VS Food Preparation Time

We can see relationship between the cuisine type and food preparation type with a histogram, seeing that generally Southern, Thai, Chinese and Japanese meals take the longest to prepare, and Korean meals take the shortest amount of time (with a few outliers). All other cuisines stay around the same 23-32 minute range in terms of food preparation.

In [ ]:
```
plt.figure(figsize = (15, 7))
plt.xticks(rotation = 90)
plt.title("Cuisine Type Vs Food Preparation Time")
sns.boxplot(data = df, x = "cuisine_type", y = "food_preparation_time")
```

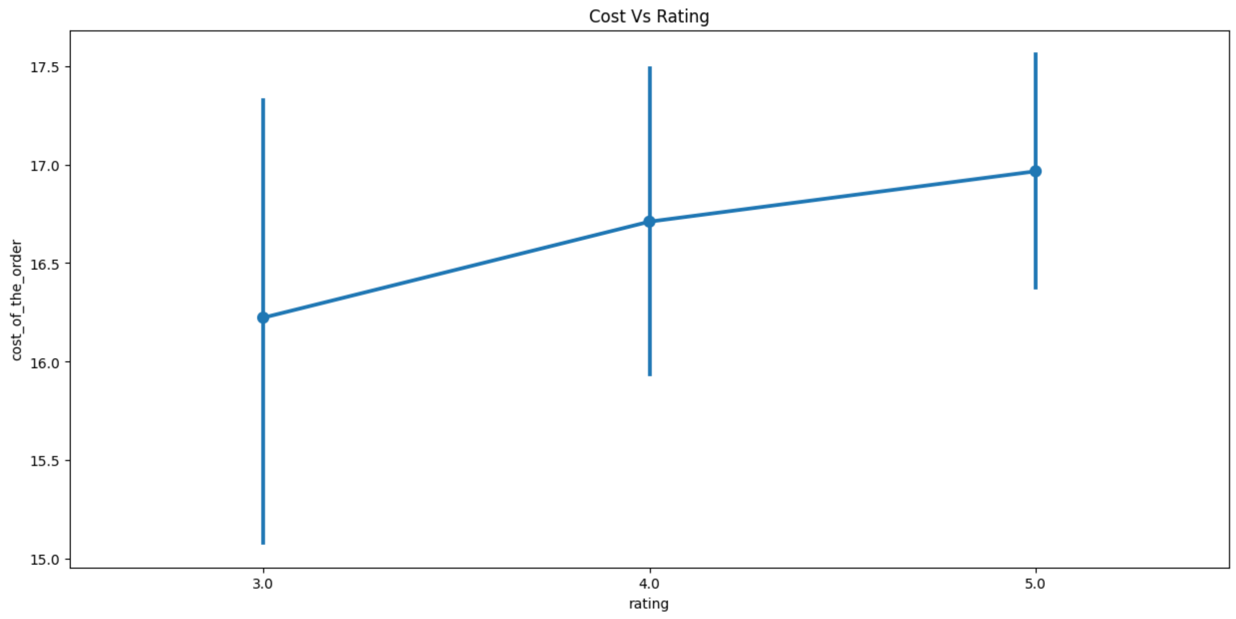
Out[ ]:
```
<Axes: title={'center': 'Cuisine Type Vs Food Preparation Time'}, xlabel='cuisine_typ
e', ylabel='food_preparation_time'>
```



Rating Vs Cost

We can see the relationship between the cost and rating with a pointplot. Looking at the visualization, we see that higher rated meals tend to cost a little bit more, but only by less than a dollar.

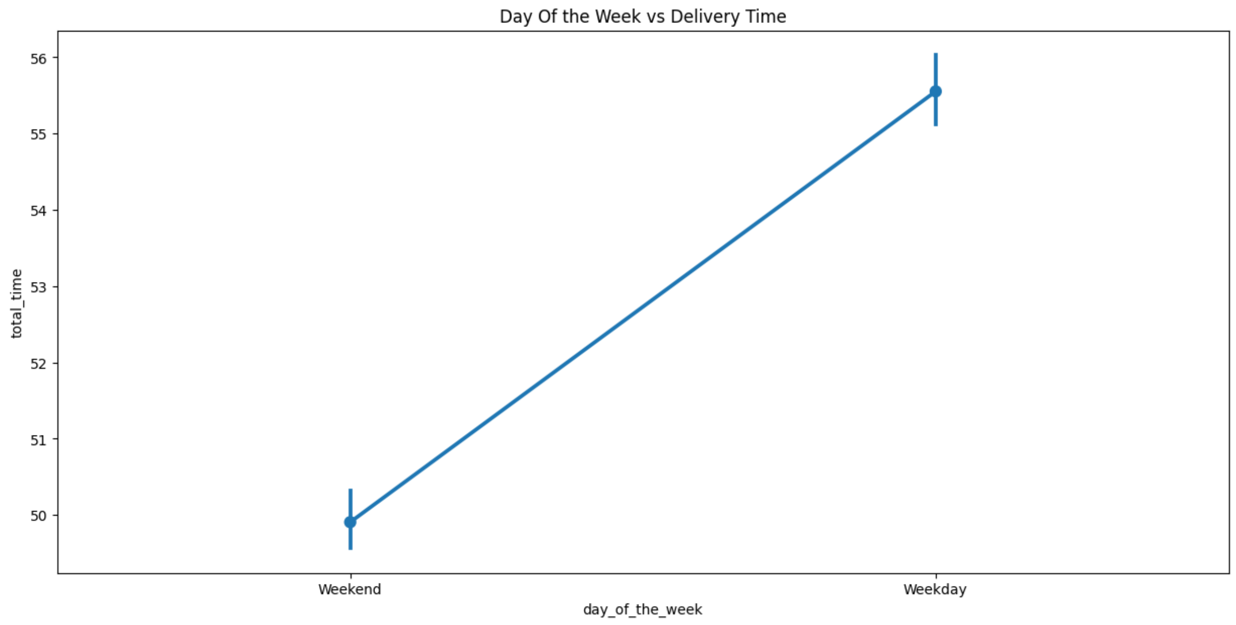
In [ ]:
```
plt.figure(figsize = (15, 7))
plt.title("Cost Vs Rating")
sns.pointplot(data = df, x = "rating", y = "cost_of_the_order")
plt.show()
```



Day of The Week Vs Delivery Time

We can see relationship between the weekends and weekdays and whether the delivery time might change, on average. Looking at the pointplot, we can see that the delivery time is much higher on weekdays than the weekends, perhaps because of rush hour or the work week.

In [ ]:
```
plt.figure(figsize = (15, 7))
plt.title("Day Of the Week vs Delivery Time")
sns.pointplot(data = df, x = "day_of_the_week", y = "total_time")
plt.show()
```

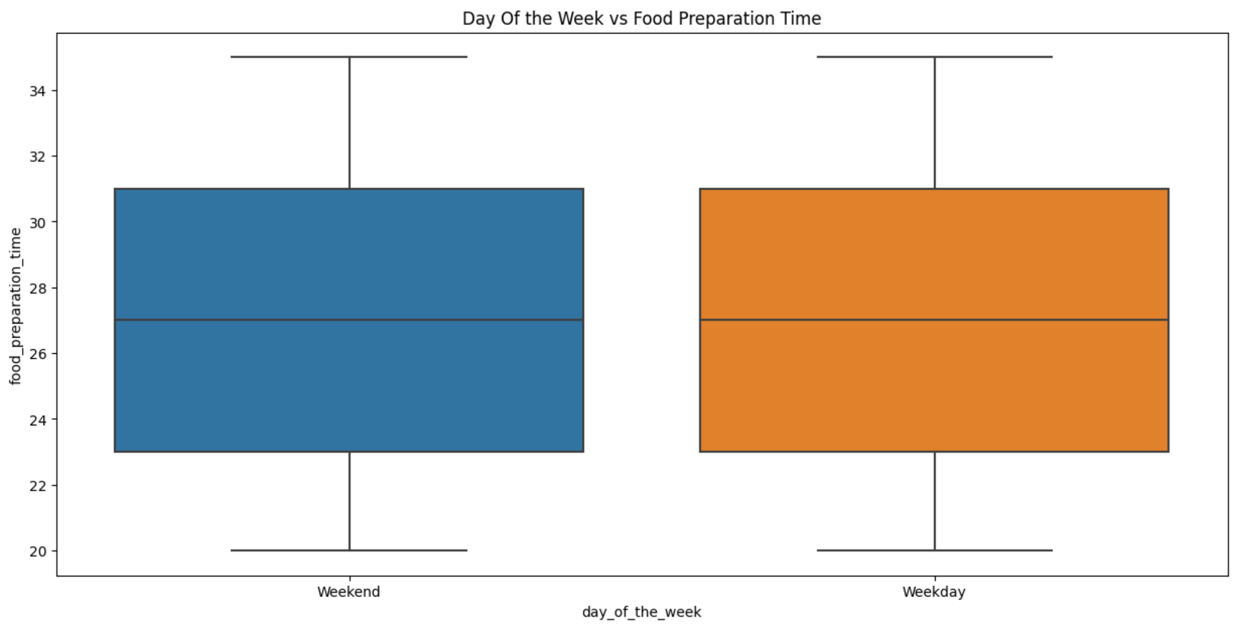


Day Of The Week Vs Prep Time

We can look at a boxplot to see if the prep time may change depending on whether it's a weekday or weekend. Looking at the boxplot, we can see that there is no noticeable change in prep time depending on the day of the week

the order is placed, so it looks like there are no specific days when restaurants get significantly more orders and their prep time suffers as a result.

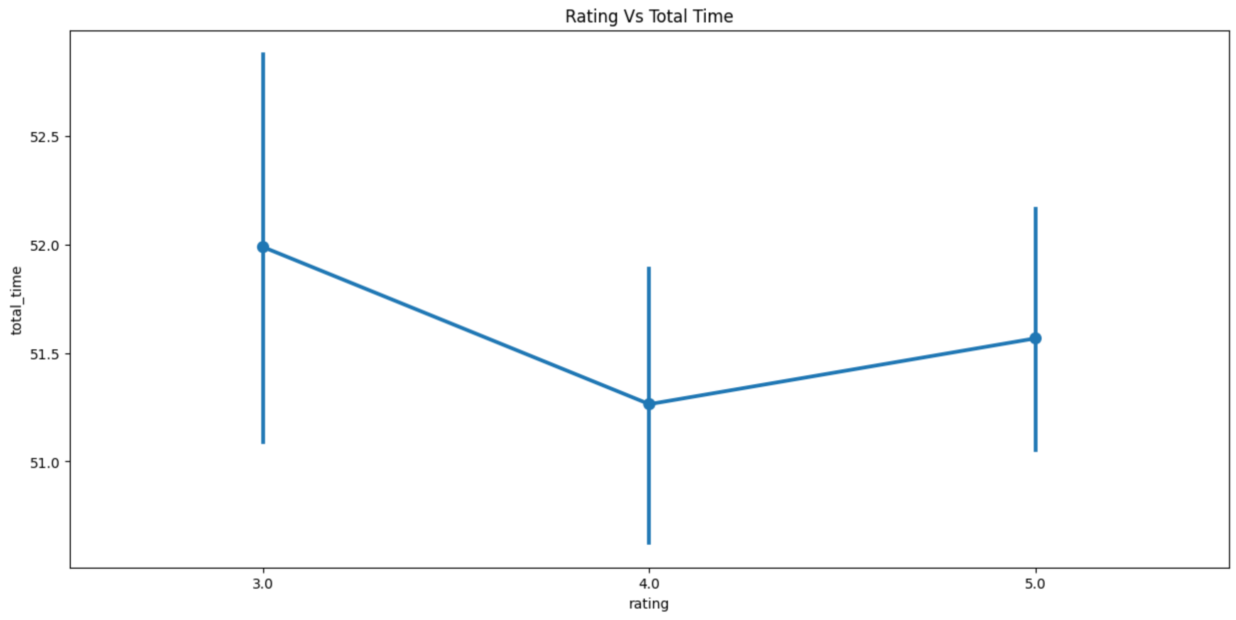
In [ ]:
```
plt.figure(figsize = (15, 7))
plt.title("Day Of the Week vs Food Preparation Time")
sns.boxplot(data = df, x = "day_of_the_week", y = "food_preparation_time")
plt.show()
```



Rating Vs Total Time

We can see if the rating may change depending on how long it takes for someone to get their order once they've placed it. By looking at the pointplot, we can see that, as we may anticipate, orders that take the longest on average have the lowest ratings (not by much). However, we can see that between 4.0 ratings and 5.0 ratings we don't see the total time it takes for the customer to get their order causing much of an issue. This leads us to believe that the difference between a 4.0 and 5.0 rating is based on the quality of the food, among other possible factors.

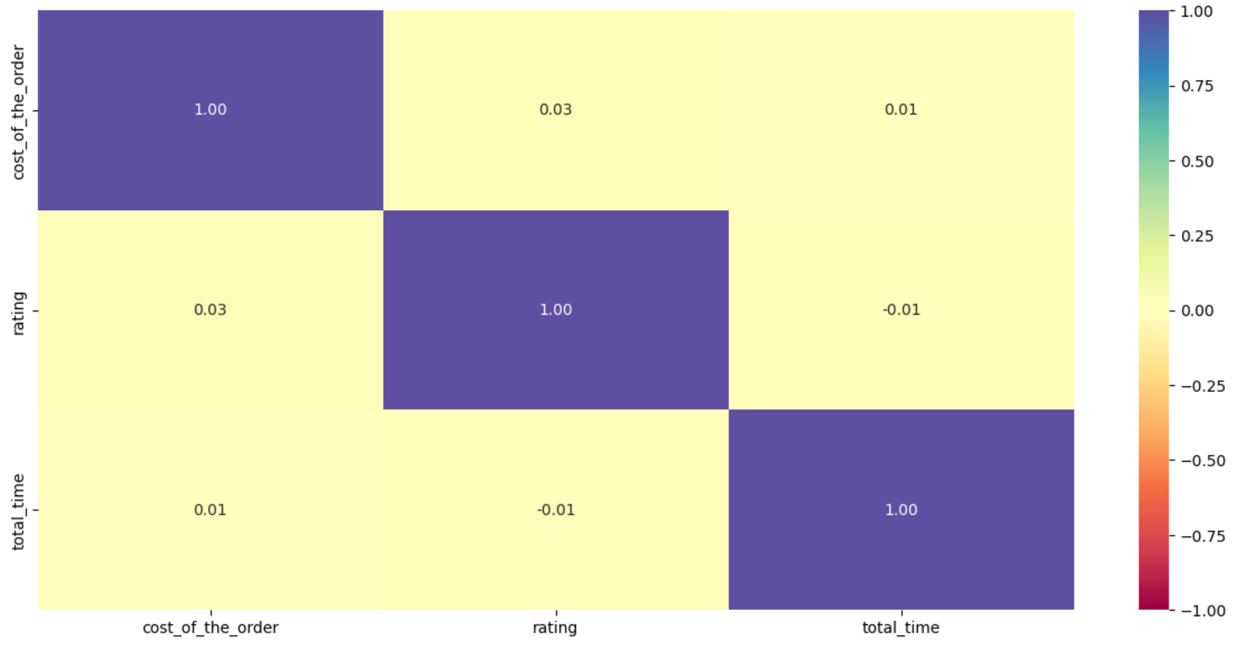
In [ ]:
```
plt.figure(figsize = (15, 7))
plt.title("Rating Vs Total Time")
sns.pointplot(data = df, x = "rating", y = "total_time")
plt.show()
```



Correlation among Variables

We can also plot a heatmap to see if there are any possible correlations between cost, rating and total time. Looking at the heatmap, we see that the correlations are between 0.03 and 0.01, so there is almost no linear association between these variables given the dataset

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



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

the restaurants fulfilling the criteria to get the promotional offer. [3 marks]

In [ ]:
```
#We get rid of the restaurants that have no ratings
df_has_rating = df[df['rating'] != np.nan].copy()

#We can then create a dataframe with only restaurant names and their rating counts, sc
df_count = df_has_rating.groupby(['restaurant_name'])['rating'].count().sort_values(as

#We then get the names of the restaurants with ratings > 50
more_than_50 = df_count[df_count['rating'] > 50]['restaurant_name']

#We then get the data of those restaurants from the list of restaurants with a rating
df_mean_rating = df_has_rating[df_has_rating['restaurant_name'].isin(more_than_50)].cc

#We then do another group by with restaurant names and ratings to find the mean rating
#above criteria
df_mean_rating.groupby(['restaurant_name'])['rating'].mean().sort_values(ascending=Fal
```

Out[ ]:
```
restaurant_name
The Meatball Shop              4.511905
Blue Ribbon Fried Chicken      4.328125
Shake Shack                    4.278195
Blue Ribbon Sushi              4.219178
Name: rating, dtype: float64
```

Observations:

We can see that 4 restaurants fit the criteria

Meatball Shop has an average rating of 4.51

Blue Ribbon Fried Chicken has an average rating of 4.33

Shake Shack has an average rating of 4.28

Blue Ribbon Sushi has an average rating of 4.22

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 [45]:
```
#We can write a function to calculate the revenue depending on the cost of the order,
#then apply it to a the costs of the order to create a new revenue column
def revenue(cost):
  if cost > 20:
    revenue = round(cost * 0.25, 2)
    return revenue
  elif cost > 5:
    revenue = round(cost * 0.15, 2)
    return revenue
  else:
    return 0

#Create a new revenue column, then calculate the sum of the entire column to get the t
```

```
df['revenue'] = df['cost_of_the_order'].apply(revenue)
total_revenue = df['revenue'].sum()
print(total_revenue)
```

```
4412.13
6166.5
```

Observations:

We see that the total revenue is 6166.50 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 [ ]:

```
#Create a new column adding the food prep time and delivery time, called total time
df['total_time'] = df['food_preparation_time'] + df['delivery_time']

#We find the total_times that are exceeding an hour, as that is what we are looking fo
df_longer_than_60 = df[df['total_time'] > 60]

#Divide the number of total_time > 60 orders by the total orders, multiply by 100 to g
fraction = (df_longer_than_60.shape[0] / df.shape[0])
percentage = round(fraction * 100, 2)
print(percentage)
```

```
10.54
```

Observations:

Roughly 10.54% of orders take longer than an hour to get delivered 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 [ ]:

```
#We can calculate the delivery times by doing a conditional in our reference of the
#day_of_the_week column, and applying the mean() function at the end of the statement
#We do this for both weekdays and weekends to find the mean delivery times for both
print(df[df['day_of_the_week'] == 'Weekday']['delivery_time'].mean())
print(df[df['day_of_the_week'] == 'Weekend']['delivery_time'].mean())
```

```
28.340036563071298
22.4700222057735
```

Observations:

Weekdays: Mean delivery time is around 28.34 minutes

Weekends: Mean delivery time is around 22 minutes

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:

- I think what's important to a company like this is knowing A) What sells the most? What cuisines are customers ordering from the most, so we can place more of a focus on marketing those cuisines. To find out this information, we can plot cuisine type vs the new revenue column we created in question 14.

B) When do we sell the most? When are customers placing more orders. This information is useful so we can either:

- focus more on this time frame, perhaps offering deals on delivery or discounts to drive more sales as this is when customers are more active on the app
- try to drive more sales to the other time frame by again, offering deals on delivery or discounts to make ordering in that time frame more appealing

After making the graphs, we can see a few things:

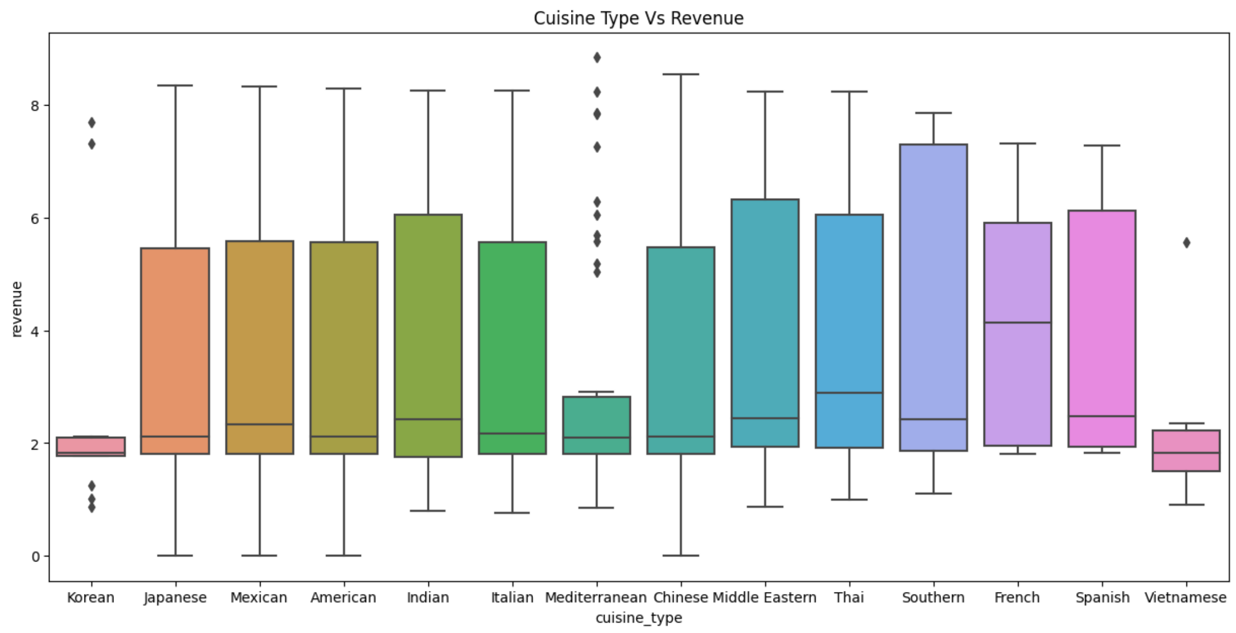
We can see that the revenue on the weekend is significantly higher than on the weekdays. We can respond to this by trying to drive more activity to the app during the week (deals, discounts, etc.) to try to get more engagement during that time. Or, we can focus on promoting more expensive cuisines on the weekends, as that's when customers are more active. We could also do both.

By looking at the cuisine vs revenue graph (which looks quite similar to the cuisine vs cost graph), we see that Southern, Middle Eastern, Thai and Indian meals are making us the most money per order, so we may want to add some kind of promotions or display those types of cuisines first on the app, to drive more sales to those cuisines.

In [46]:
```
weekend_revenue = df[df['day_of_the_week'] == 'Weekend']['revenue'].sum()
weekday_revenue = df[df['day_of_the_week'] == 'Weekday']['revenue'].sum()
print(weekend_revenue)
print(weekday_revenue)

plt.figure(figsize = (15, 7))
plt.title("Cuisine Type Vs Revenue")
sns.boxplot(data = df, x = "cuisine_type", y = "revenue")
plt.show()
```

```
4412.13
1754.3700000000001
```



Recommendations:

There are two approaches that might make sense, based on the data:

Approach 1: Lean into the trends we're seeing in our user base to drive revenue

- Provide deals, discounts, etc. on the weekend, similar to something like Uber Eats or PostMates, because we get significantly more orders/activity during that timeframe
- We know from the graphs that we sell American, Japanese, Chinese and Italian foods the most, so we promote those foods as much as we can, offering discounts or putting the closest restaurants that offer those cuisines on the home page of the user to make them as accessible as possible

Approach 2: Try and drive more sales to the underutilized days of the week/cuisines to drive revenue

- We can try to drive more activity to the app during the week (deals, discounts, etc.) when we get less orders to try and get more people active on the app during the week as opposed to the weekend.

- We can also focus on promoting more expensive cuisines on the weekends, as that's when customers are more active. We can tell from the data that people are likely going to order American, Japanese, Italian and Chinese food regardless, just looking at the order counts by cuisine type. However, Southern, Middle Eastern, French, Indian, Spanish and Thai cuisines tend to make us more money, and have significantly lower order counts when we look at the cuisine_type countplot. To that end, we should place less emphasis on American, Japanese, Chinese and Italian restaurants during the weekend (people are likely to order those anyway if that's what they really want) and place more emphasis on these less-ordered cuisines in an effort to increase revenue.

Conclusion:

In general, I would personally recommend approach 1, as it seems the safer option. Although there may be more upside with the second approach, there is a chance that people just don't order out as much on the weekdays, or they simply won't order the less-ordered cuisines because they're more expensive or they don't like them as much. If that's the case, approach 2 won't help us increase revenue much at all, and may even hurt us by placing less of an emphasis on the time we get the most orders (weekend) and the cuisines we sell the most (American, Italian, etc.).

However, I would recommend they do some focus testing with each approach to see what works, because if approach 2 goes well, that seems like the approach that would drive revenue the most.
