Introduction to Python Project : FoodHub Data Analysis

Problem Statement

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 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 improve its business.

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:** Price paid per 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 [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Understanding the structure of the data

In [3]:

```
# uncomment and run the following lines for Google Colab
from google.colab import drive
drive.mount('/content/drive')
```

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

In [4]:

```
# read the data
path = '/content/drive/MyDrive/foodhub_order.csv'
df = pd.read_csv(path)
```

In [5]:

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

Out[5]:

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

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

In [5]:

```
df.shape
```

Out[5]:

```
(1898, 9)
```

Observations:

- **There are `1898 rows` & `9 columns` in the food data set.**

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

In [ ]:

```
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:

- **The `order_id`, `customer_id`, `cost_of_the_order`, `food_preparation_time`, and `delivery_time` columns are numeric column while the rest are categorical in nature.**
- **All the columns have `1898 observations`, which means none of the columns has null values.**

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

In [ ]:

```
df.isnull().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                  0
food_preparation_time   0
delivery_time           0
dtype: int64
```

Observations:

- **As we can see, all the columns has `0 missing value`. It means there are no missing values in the data.**

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 [ ]:

```
df.describe()
```

Out[ ]:

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

Observations:

- **The *minimum*, *average*, and *maximum* `food_preparation_time` are *20 minutes*, around *27 minutes* and *35 minutes* respectively.**

- `cost_of_the_order` **varies from around 4 dollars to 35 dollars & average cost of the order is $16.**
- **The *minumum*, *average*, and *maximum* `delivery_time` are *15 minutes*, *25 minutes* and *33 minutes* respectively.**

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

In [ ]:

```
# Method1:
df['rating'].value_counts()
```

Out[ ]:

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

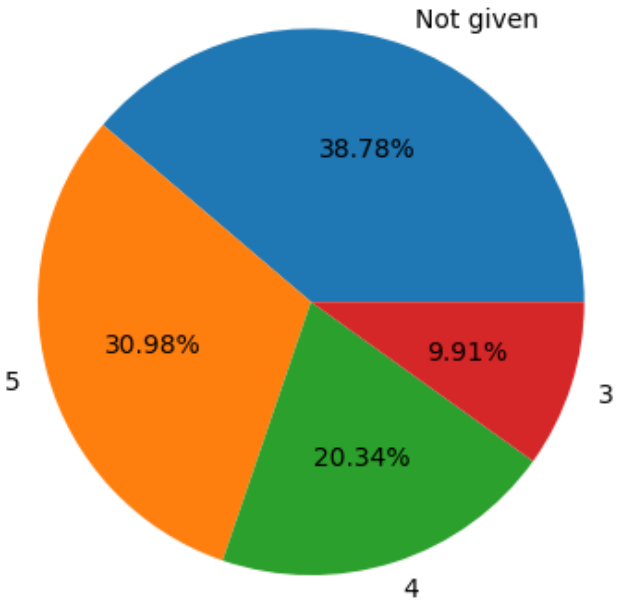
In [ ]:

```
# Method2:
not_rated = df['rating'].value_counts()['Not given']
print(f"There are {not_rated} orders are not rated.")
```

```
There are 736 orders are not rated.
```

In [ ]:

```
plt.pie(df['rating'].value_counts(), labels=df['rating'].value_counts().index, autopct='
%0.2f%%');
```



Observations:

- `736 observations` **are comes under** `Not given` **category in the rating column, it means 736 orders are not rated.**
- **Approximately** `39%` **of the orders are not rated by users.**
- **Around** `31% (588 observations)` **of the orders are rated 5 by users.**

Exploratory Data Analysis (EDA)

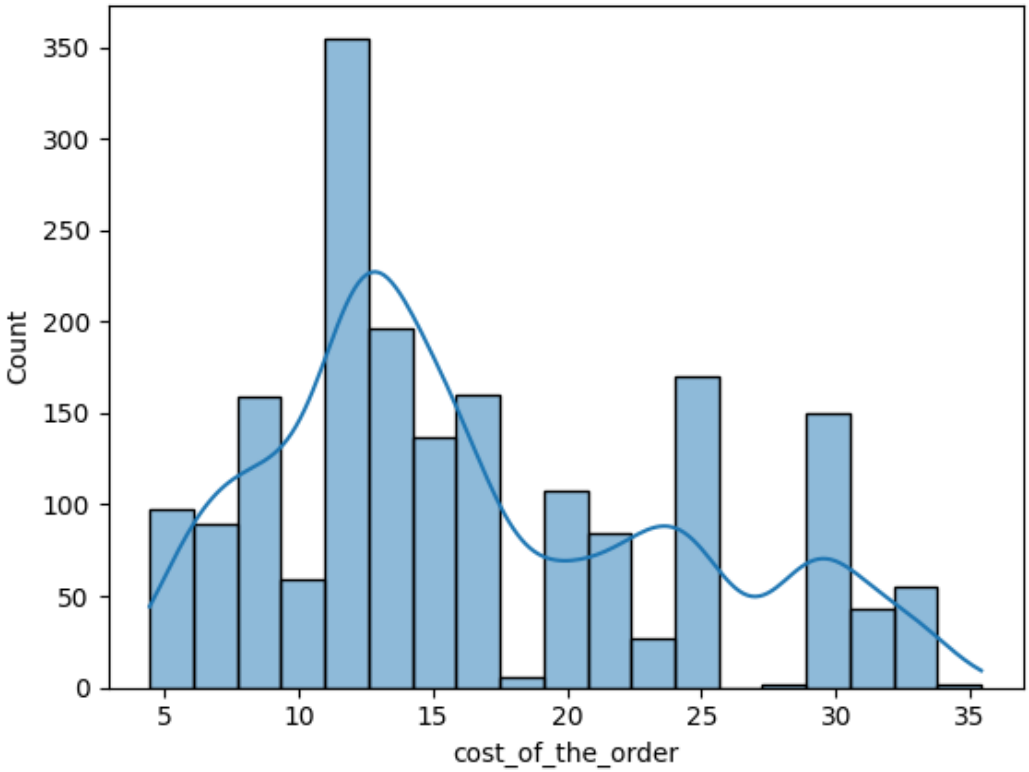
Univariate Analysis

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

Distribution of `cost_of_the_order` *(Numerical Column)*

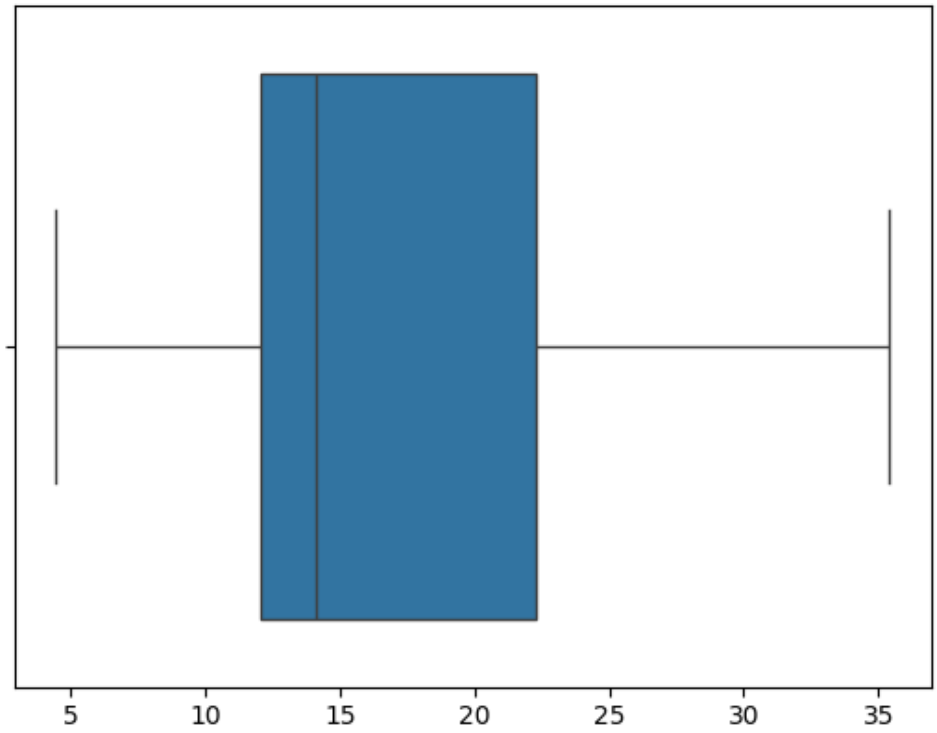
In [ ]:

```
sns.histplot(data=df, x='cost_of_the_order', kde=True);
```



In [ ]:

```
sns.boxplot(data=df, x='cost_of_the_order');
```



cost_of_the_order

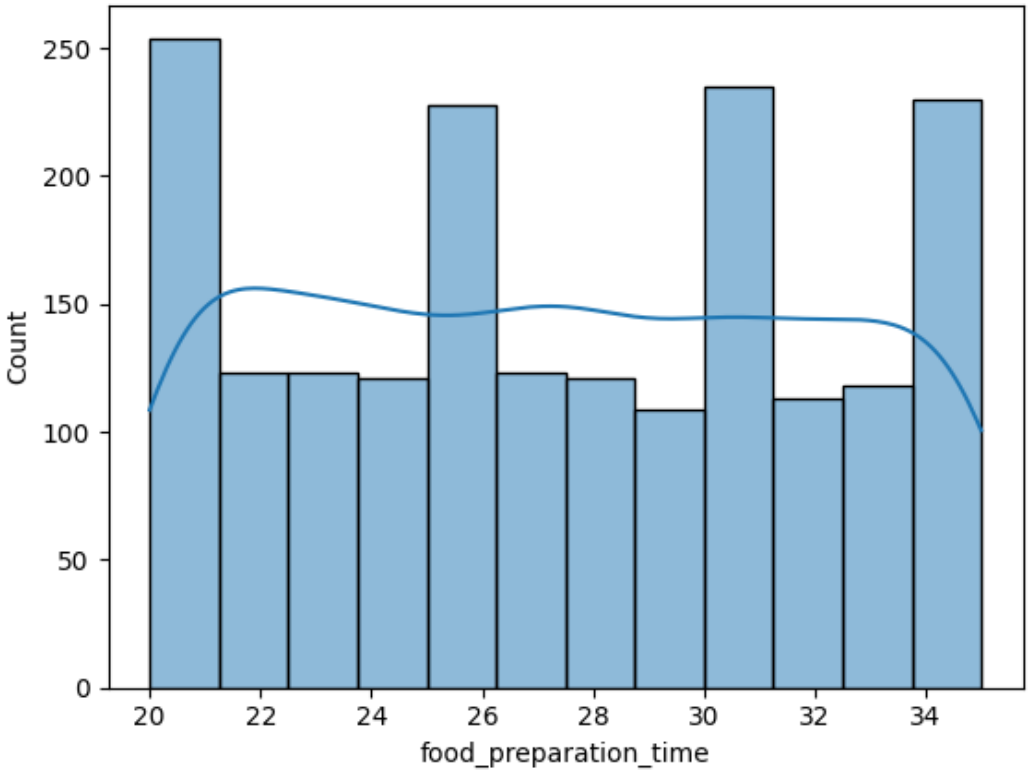
Observations:

- **The `cost_of_the_order` has a right-skewed distribution with no outlier.**
- **The median of `cost_of_the_order` is nearly 14 dollars.**

Distribution of `food_preparation_time` *(Numerical Columns)*

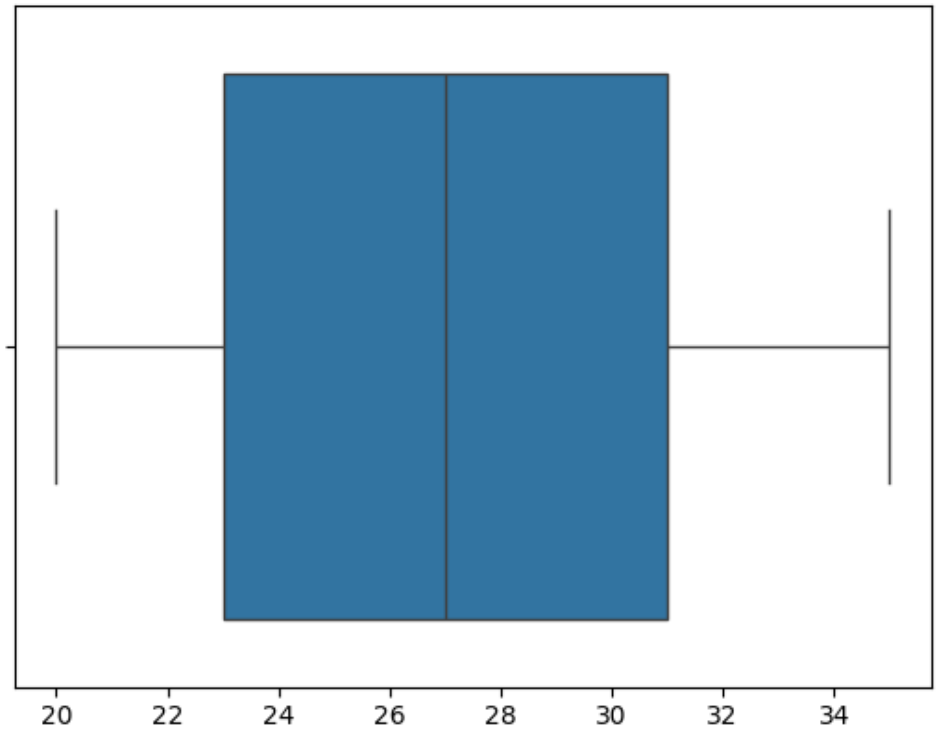
In [ ]:

```
sns.histplot(data=df, x='food_preparation_time', kde=True);
```



In [ ]:

```
sns.boxplot(data=df, x='food_preparation_time');
```



food_preparation_time

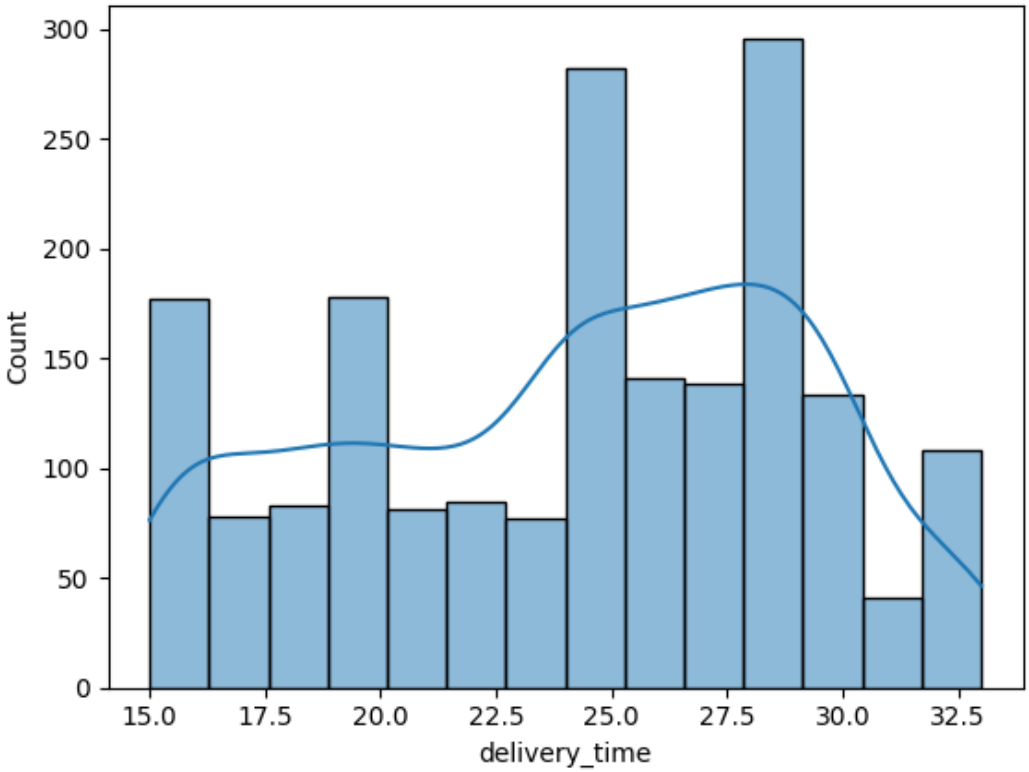
Observations:

- **The `food_preparation_time` distribution looks like evenly distributed.**
- **The median of `food_preparation_time` is around 27 minutes.**

Distribution of `delivery_time` *(Numerical Column)*

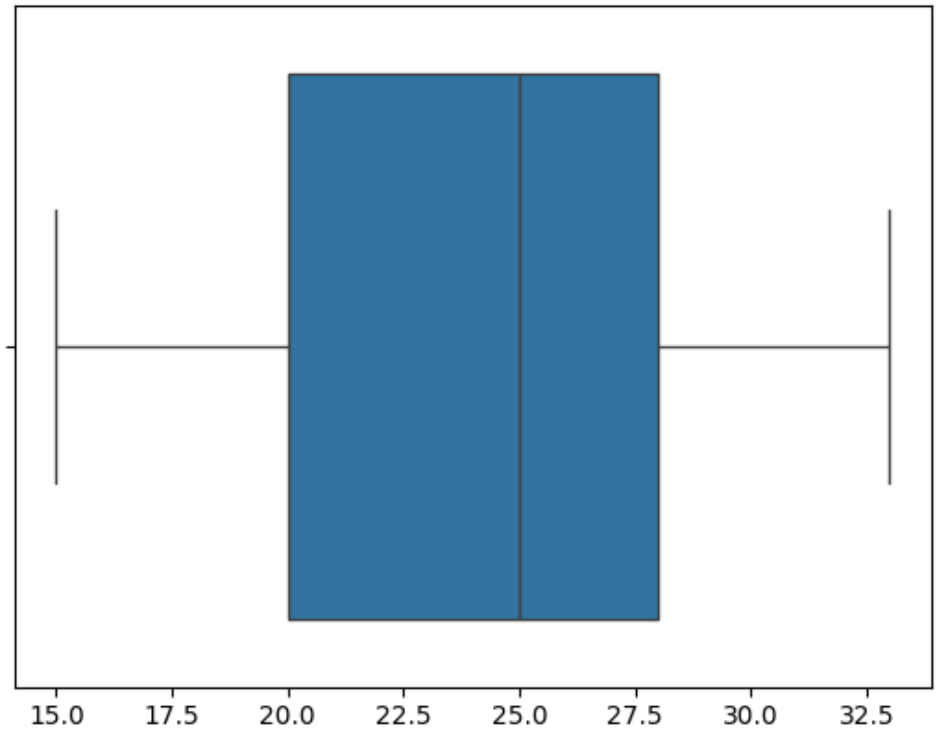
In [ ]:

```
sns.histplot(data=df, x='delivery_time', kde=True);
```



In [ ]:

```
sns.boxplot(data=df, x='delivery_time');
```



delivery_time

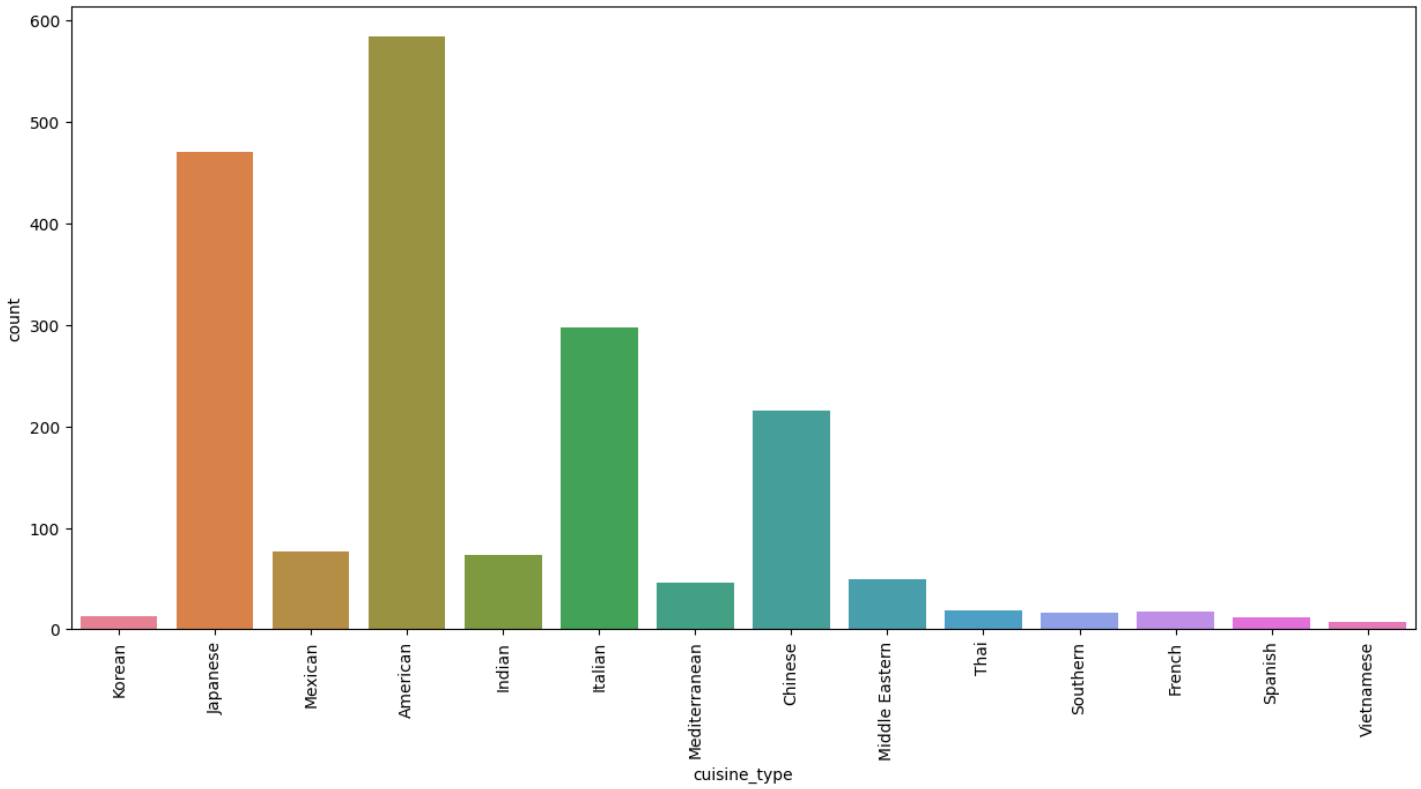
Observations:

- **Distribution looks like evenly distributed with no outliers.**
- **The median of `delivery_time` is 25 minutes.**

Distribution of `cuisine_type` *(Categorical Column)*

In [ ]:

```
plt.figure(figsize=(15,7))
sns.countplot(data=df, x='cuisine_type', hue='cuisine_type')
plt.xticks(rotation=90);
```



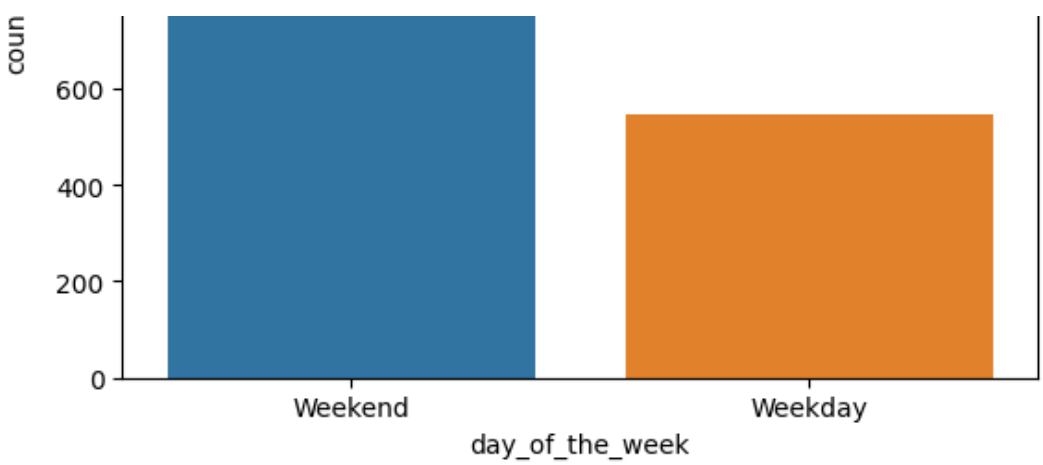
Observations:

- **The highest orders have been received for `American` cuisine, followed by `Japanese` and `Italian` cuisine.**
- **`Vietnamese` cuisine has received the least number of orders.**

Distribution of `day_of_the_week` *(Categorical Column)*

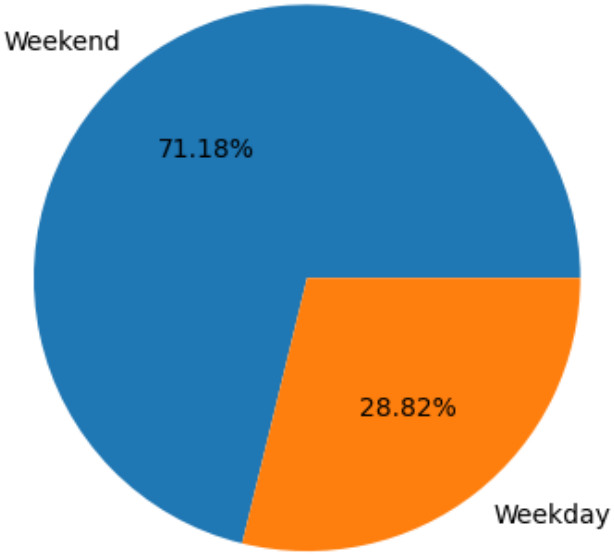
In [ ]:

```
sns.countplot(data=df, x='day_of_the_week', hue='day_of_the_week');
```



In [ ]:

```
plt.pie(df['day_of_the_week'].value_counts(), labels=df['day_of_the_week'].value_counts(
).index, autopct='%0.2f%%');
```



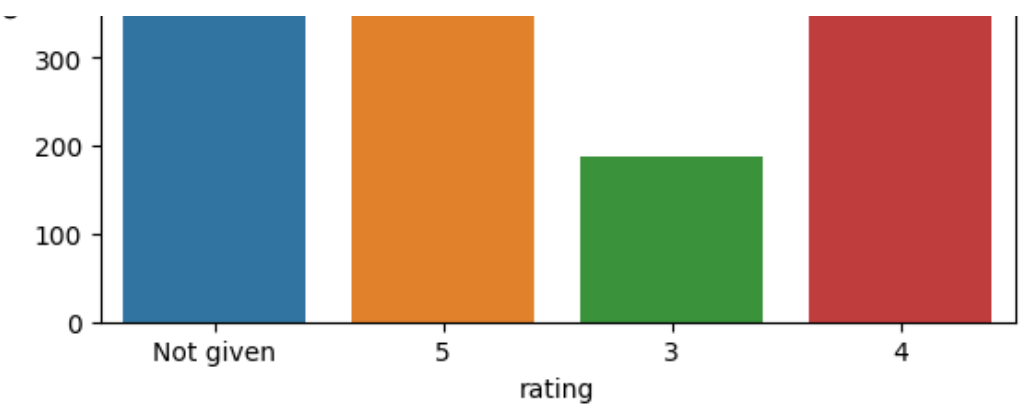
Observations:

- **The restaurant receives more orders on Weekends than the Weekdays.**
- **About `71%` of the orders are placed by users on Weekends.**

Distribution of `rating` *(Categorical Column)*

In [ ]:

```
sns.countplot(data=df, x='rating', hue='rating');
```



Observations:

- **We can see that `over 700 orders` have not been rated, indicating that users are not interested in reviewing their orders.**

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

We are using three types of methods or format

- **Method 1: Using pandas series**
- **Method 2: Using pandas data frame**
- **Method 3: Using graph**

In [ ]:

```
# Format 1: return output in pandas series
df['restaurant_name'].value_counts().head()
```

Out[ ]:

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

In [ ]:

```
# Format 2: Return output in Data Frame (for more readable)
pd.DataFrame(df['restaurant_name'].value_counts().head()).reset_index()
```

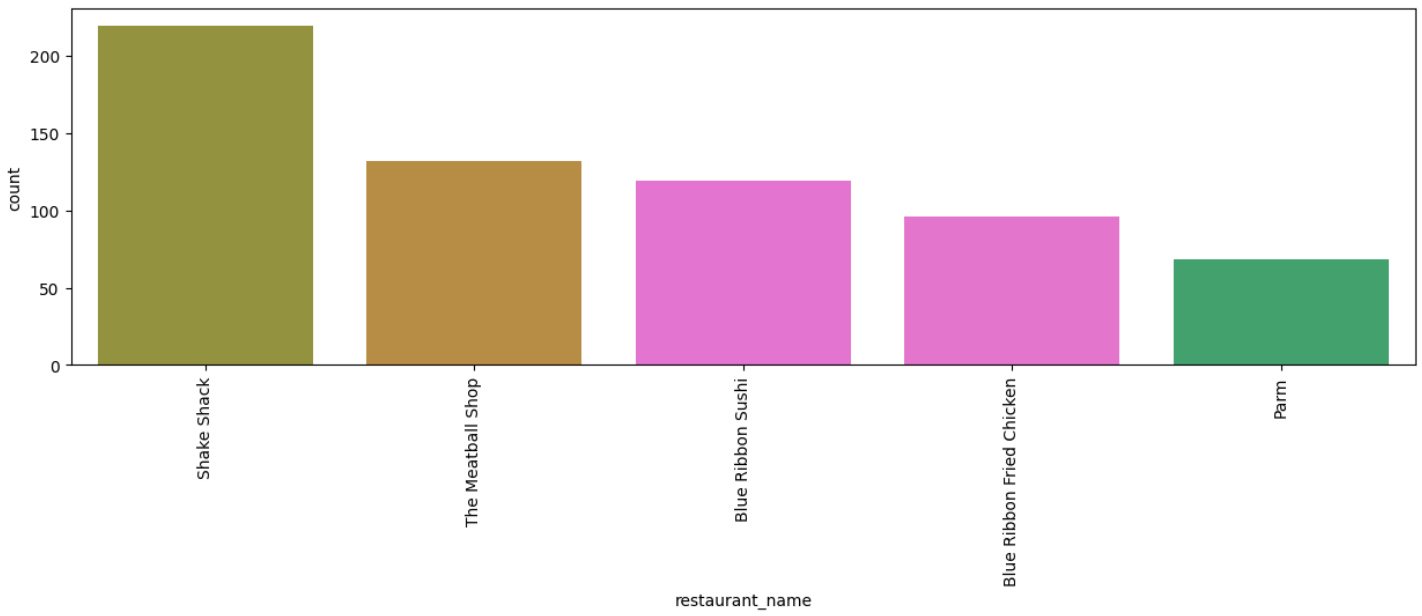
Out[ ]:

	restaurant_name	count
0	Shake Shack	219
1	The Meatball Shop	132
2	Blue Ribbon Sushi	119
3	Blue Ribbon Fried Chicken	96
4	Parm	68

In [ ]:

```
# Format 3: Using graph
plt.figure(figsize=(15,4))
sns.countplot(data=df.sort_values('restaurant_name', ascending=False), x='restaurant_nam
e', hue='restaurant_name', order=df['restaurant_name'].value_counts().index[:5])
```

```
plt.xticks(rotation=90);
```



Extra Calculations

In [ ]:

```
# Fetch top 5 restaurant's name from data set
top_5_rest = df['restaurant_name'].value_counts().head().index
top_5_rest
```

Out[ ]:

```
Index(['Shake Shack', 'The Meatball Shop', 'Blue Ribbon Sushi',
       'Blue Ribbon Fried Chicken', 'Parm'],
      dtype='object', name='restaurant_name')
```

In [ ]:

```
# fetch top 5 restaurant's data from the original data frame
filtered_df = df[df['restaurant_name'].isin(top_5_rest)]
filtered_df
```

Out[ ]:

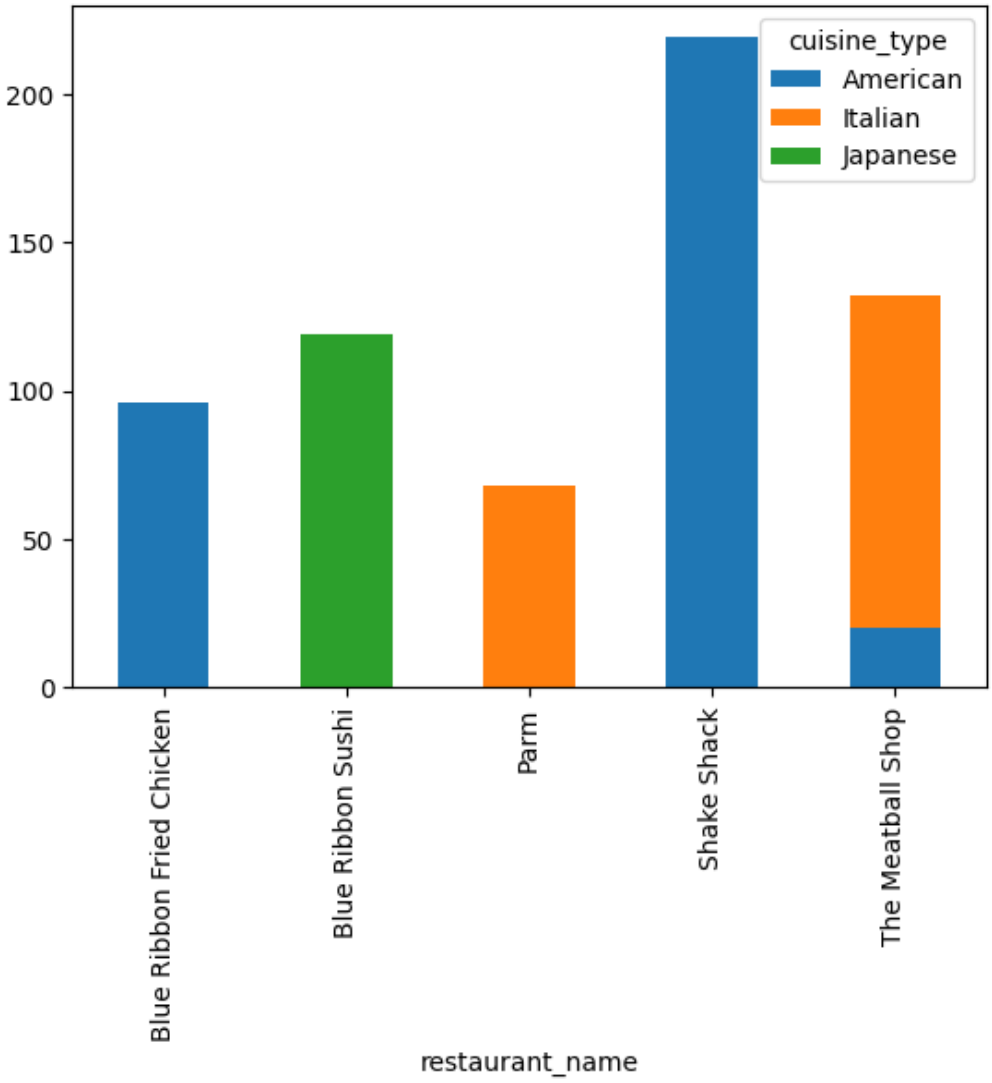
	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25
6	1477894	157711	The Meatball Shop	Italian	6.07	Weekend	Not given	28
12	1476966	129969	Blue Ribbon Fried Chicken	American	24.30	Weekend	5	23
15	1477414	66222	Shake Shack	American	16.20	Weekend	5	33
19	1477354	67487	Blue Ribbon Sushi	Japanese	16.20	Weekend	4	35
...	...	...	...	...	...	...	...	..
1887	1476873	237616	Shake Shack	American	5.82	Weekend	Not given	26
1888	1477353	106324	The Meatball Shop	Italian	16.20	Weekend	5	21
1891	1476981	138586	Shake Shack	American	5.82	Weekend	Not given	22
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	Not given	31
			Blue Ribbon				Not	

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time
1897	1478056	120353	[illegible]	Japanese	19.45	Weekend	[illegible]	28

634 rows × 9 columns

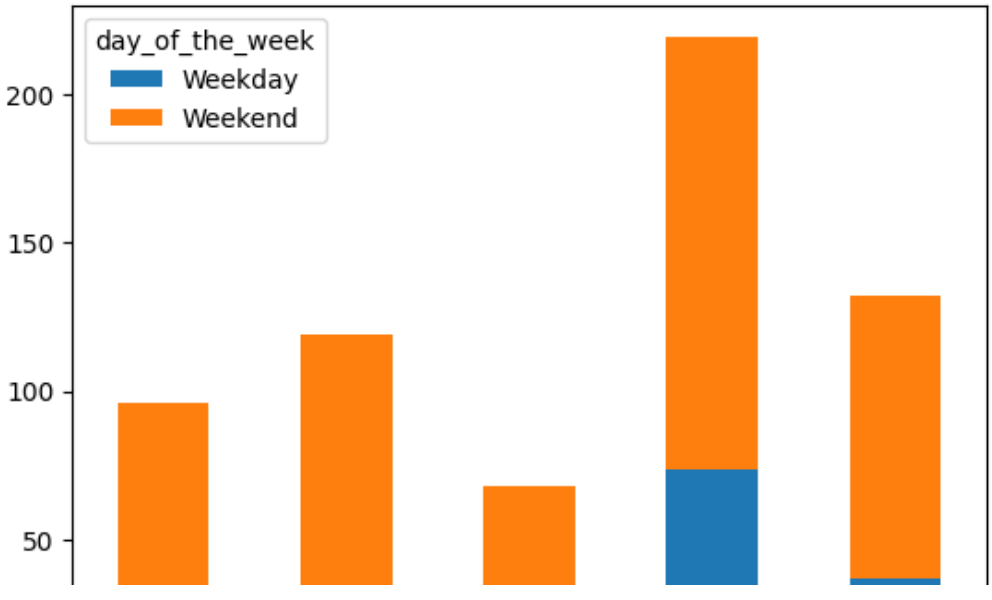
In [ ]:

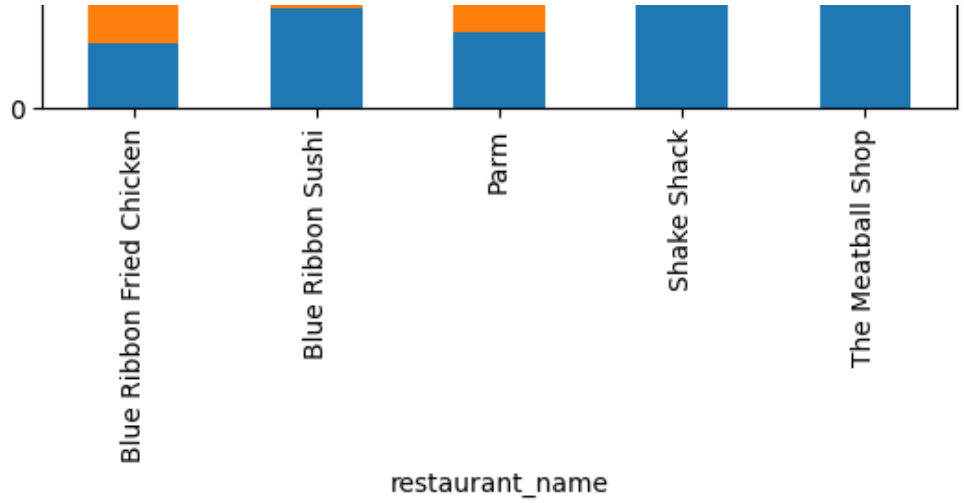
```
pd.crosstab(filtered_df['restaurant_name'], filtered_df['cuisine_type']).plot.bar(stacked=True);
```



In [ ]:

```
pd.crosstab(filtered_df['restaurant_name'],  filtered_df['day_of_the_week']).plot.bar(stacked=True);
```





In [ ]:

```
top_5_rest_info = {"Restaurant Name":[], "Total Orders":[], "Weekends":[], "Weekdays":[]}
for rest_name in top_5_rest:
  top_5_rest_info["Restaurant Name"].append(rest_name)
  top_5_rest_info["Total Orders"].append(df[df['restaurant_name']==rest_name].shape[0])
  top_5_rest_info["Weekends"].append(df[df['restaurant_name']==rest_name]['day_of_the_week'].value_counts()['Weekend'])
  top_5_rest_info["Weekdays"].append(df[df['restaurant_name']==rest_name]['day_of_the_week'].value_counts()['Weekday'])

top_5_df = pd.DataFrame(top_5_rest_info)
top_5_df
```

Out[ ]:

	Restaurant Name	Total Orders	Weekends	Weekdays
0	Shake Shack	219	145	74
1	The Meatball Shop	132	95	37
2	Blue Ribbon Sushi	119	85	34
3	Blue Ribbon Fried Chicken	96	74	22
4	Parm	68	42	26

Observations:

- `Shake Shack` **restaurant received the highest number of orders, followed by** `The Meatball Shop`, `Blue Ribbon Sushi`, `Blue Ribbon Fried Chicken` **and** `Parm`.
- **Except** `The Meatball Shop` **restaurant all four restaurants ordered one type of cuisine only.**

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

There are two methods

- **Method 1: Using pandas data frame**
- **Method 2: Using graph**

In [ ]:

```
# Method 1: Using pandas
df[df['day_of_the_week']=='Weekend']['cuisine_type'].value_counts().head(1)
```

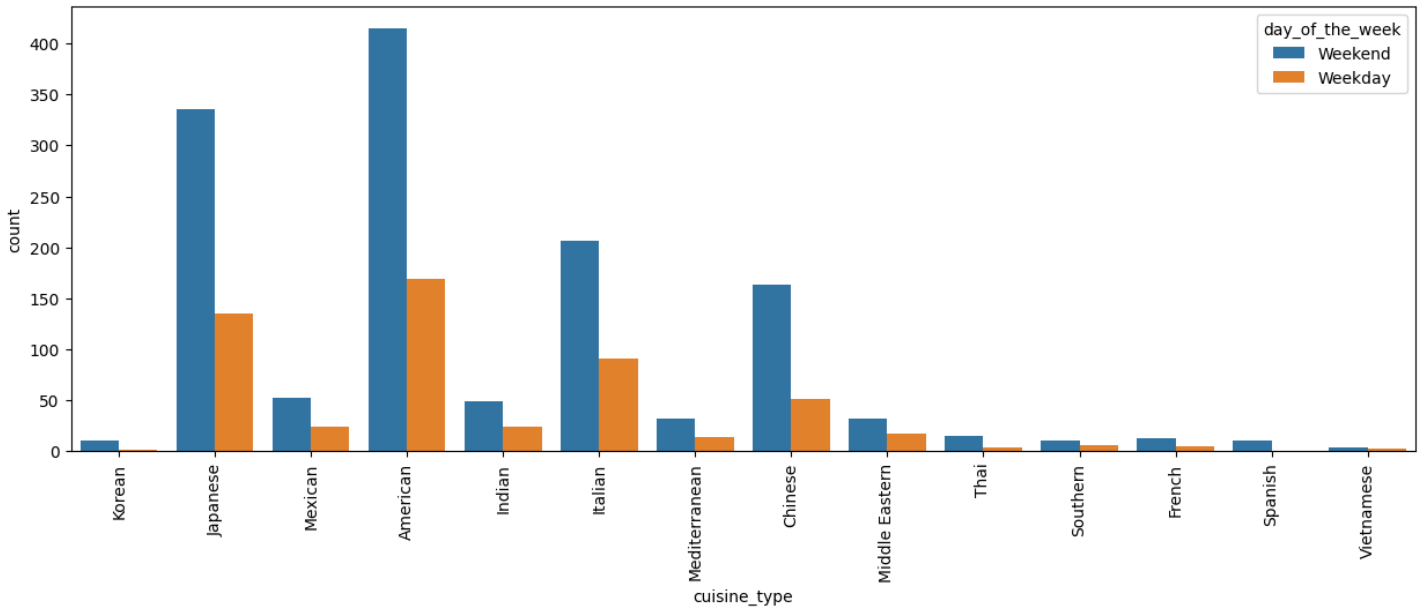
Out[ ]:

```
cuisine_type
American    415
Name: count, dtype: int64
```

Name: count, dtype: int64

In [ ]:

```
# Method 2: Using graph
plt.figure(figsize=(15,5))
sns.countplot(data=df, x='cuisine_type', hue='day_of_the_week')
plt.xticks(rotation=90);
```



Observations:

- **The `most popular` cuisine on `weekends` is `American` cuisine, followed by `Japanese`, `Italian` and other cusines.**
- **The consistently favourite and popular cuisine is `American`. It remains the top choice both on weekends and weekdays, in this dataset.**

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

In [ ]:

```
get_percentage = df[df['cost_of_the_order'] > 20].shape[0] / df.shape[0] * 100
print(f"Percentage of the orders cost more than $20 is {round(get_percentage,2)}%.")
```

```
Percentage of the orders cost more than $20 is 29.24%.
```

Observations:

- **Percentage of the `cost_of_the_order` more than $20 is 29.24%**

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

In [ ]:

```
mean_del_time = df["delivery_time"].mean()
print(f"The mean order delivery time is {round(mean_del_time, 2)} seconds.")
```

```
The mean order delivery time is 24.16 seconds.
```

Observations:

- **The mean order `delivery_time` is 24.16 seconds.**

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]

We are using three types of methods

- **Method 1: Using pandas series**
- **Method 2: Using pandas data frame**
- **Method 3: Using graph**

In [ ]:

```
# Method 1: Return output in pandas series
df['customer_id'].value_counts().head(3)
```

Out[ ]:

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

In [ ]:

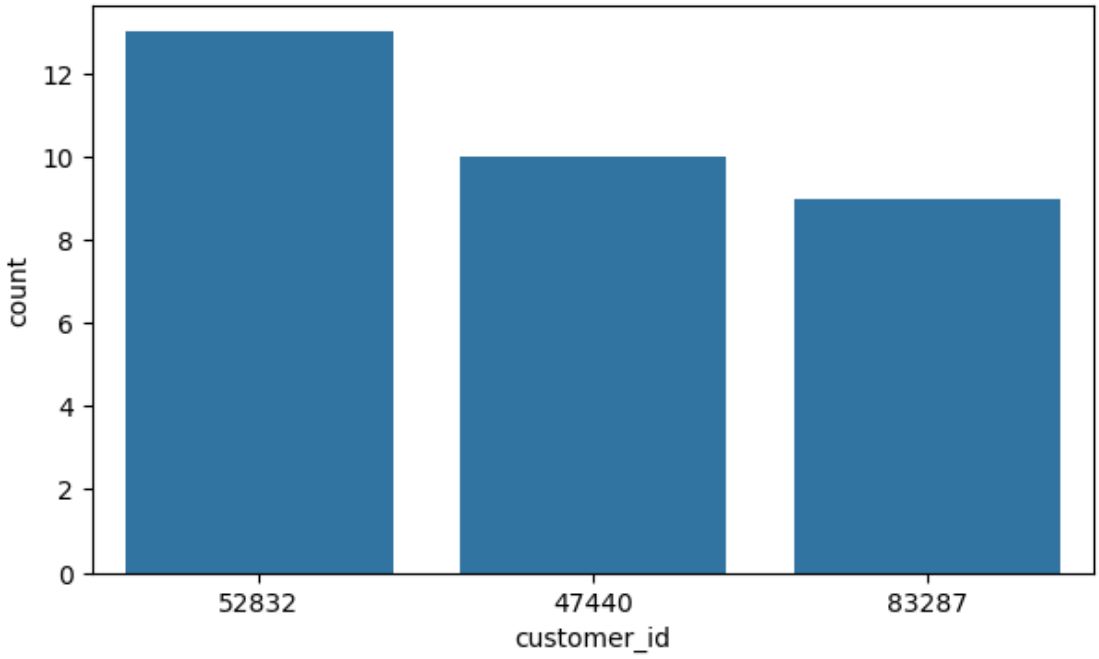
```
# Method 2: Return output in Data Frame (for more readable)
pd.DataFrame(df['customer_id'].value_counts()).head(3).reset_index()
```

Out[ ]:

	customer_id	count
0	52832	13
1	47440	10
2	83287	9

In [ ]:

```
# Method 3: Using Graph
plt.figure(figsize=(7,4))
sns.countplot(data=df.sort_values('customer_id', ascending=False), x='customer_id', order
=df['customer_id'].value_counts().index[:3]);
```



Observations:

- **customer_id `52832` placed highest number of orders i.e. 13, followed by the customer_id `47440 (count 10)` and customer_id `83287 (count 9)`.**

Multivariate Analysis

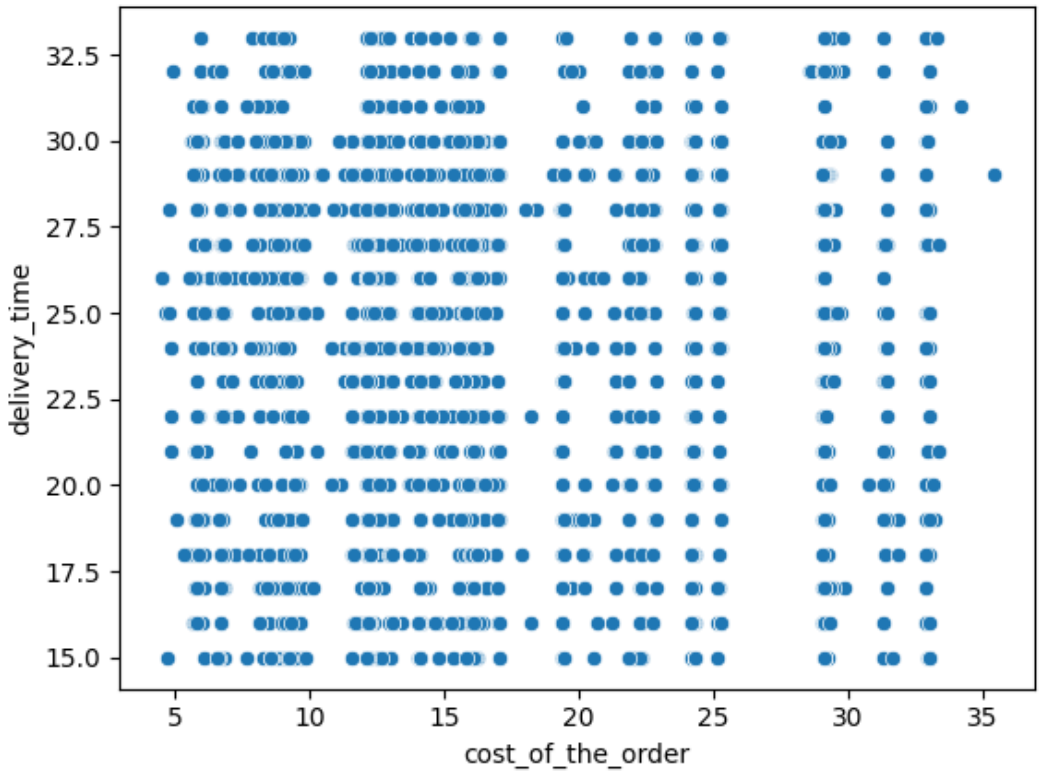
Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. (It is a good idea to explore relations between numerical variables as well as relations between numerical and categorical variables) [10 marks]

Multivariate Analysis for *(Numerical Variables vs Numerical Variables)*

Distribution of `cost_of_the_order` vs `delivery_time` (Numerical Column vs Numerical column)

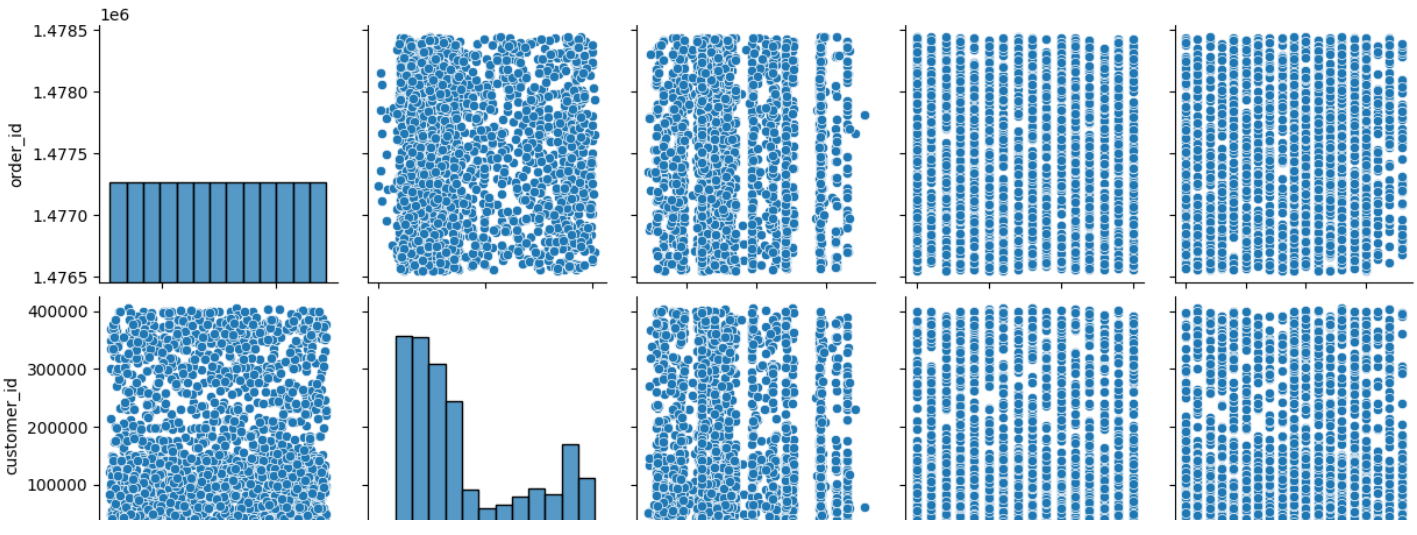
In [ ]:

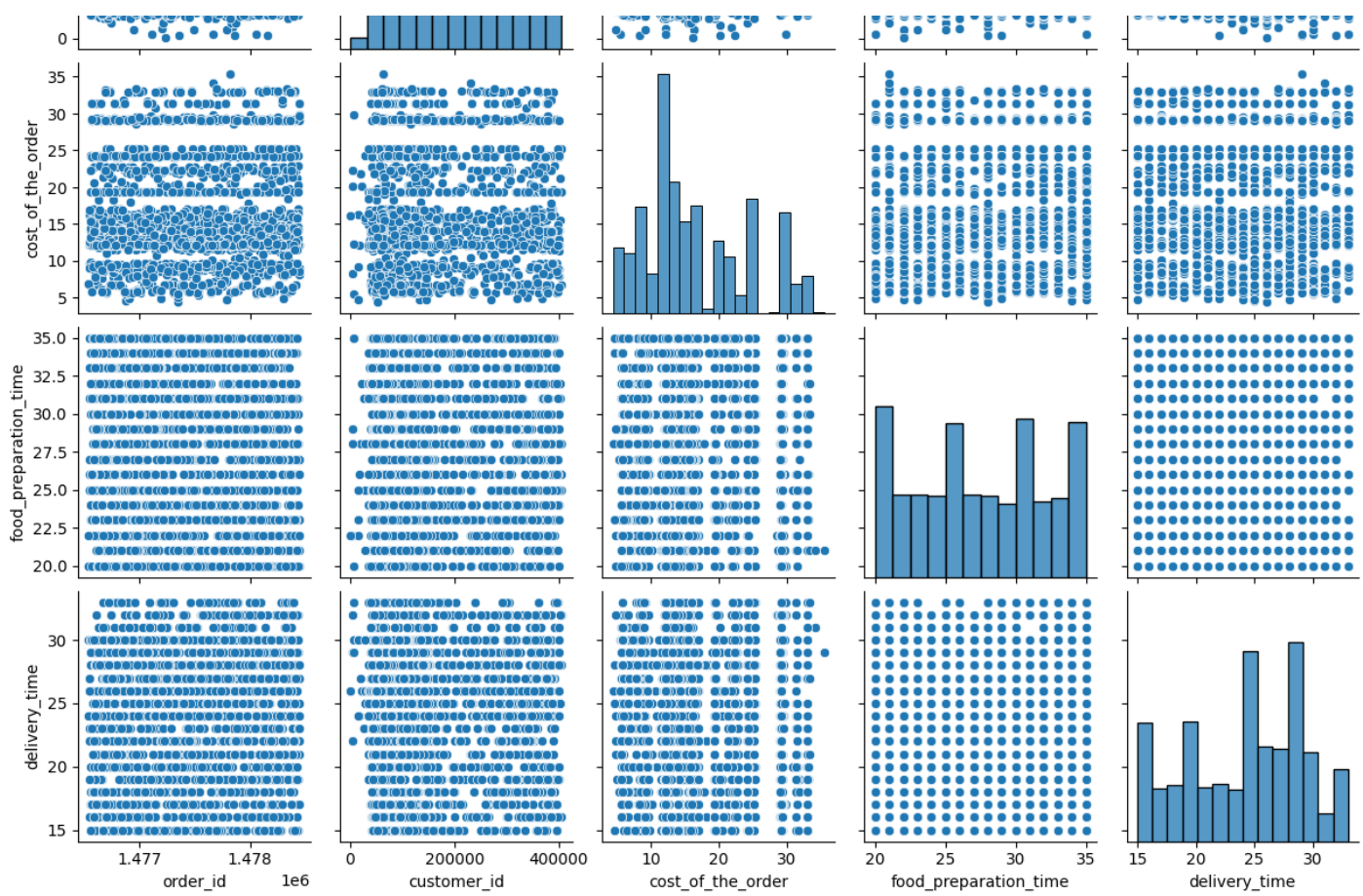
```
sns.scatterplot(data=df, x='cost_of_the_order', y='delivery_time');
```



In [ ]:

```
sns.pairplot(data=df);
```

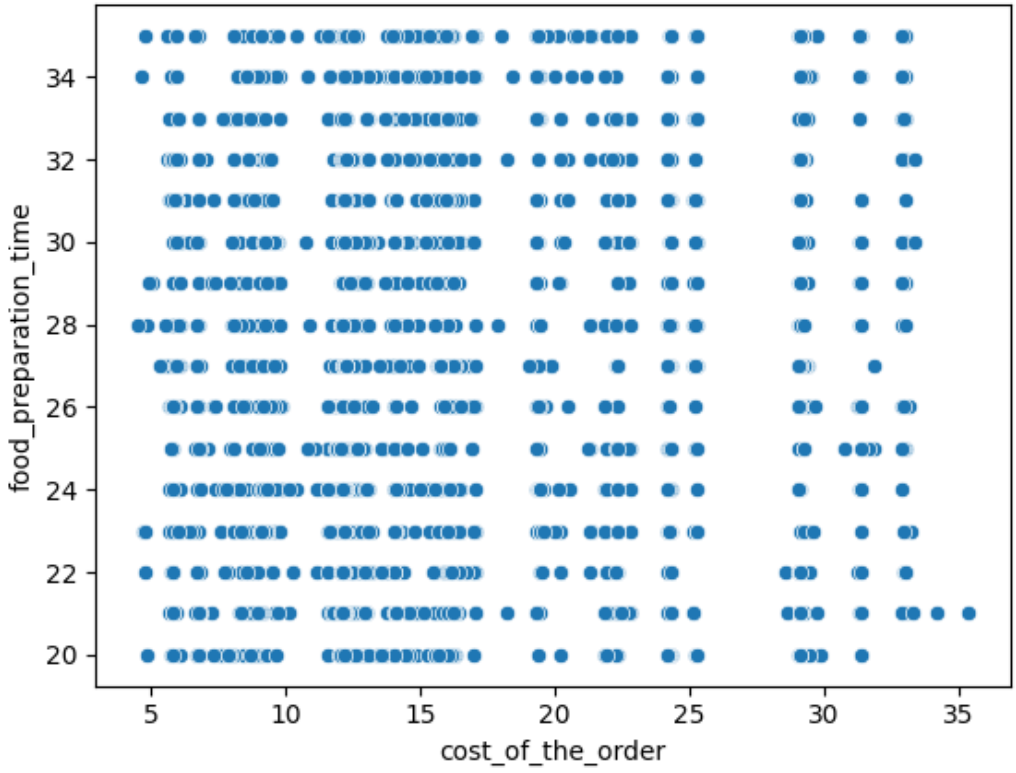




Distribution of `cost_of_the_order` ***vs*** `food_preparation_time` ***(Numerical Column vs Numerical Column)***

In [ ]:

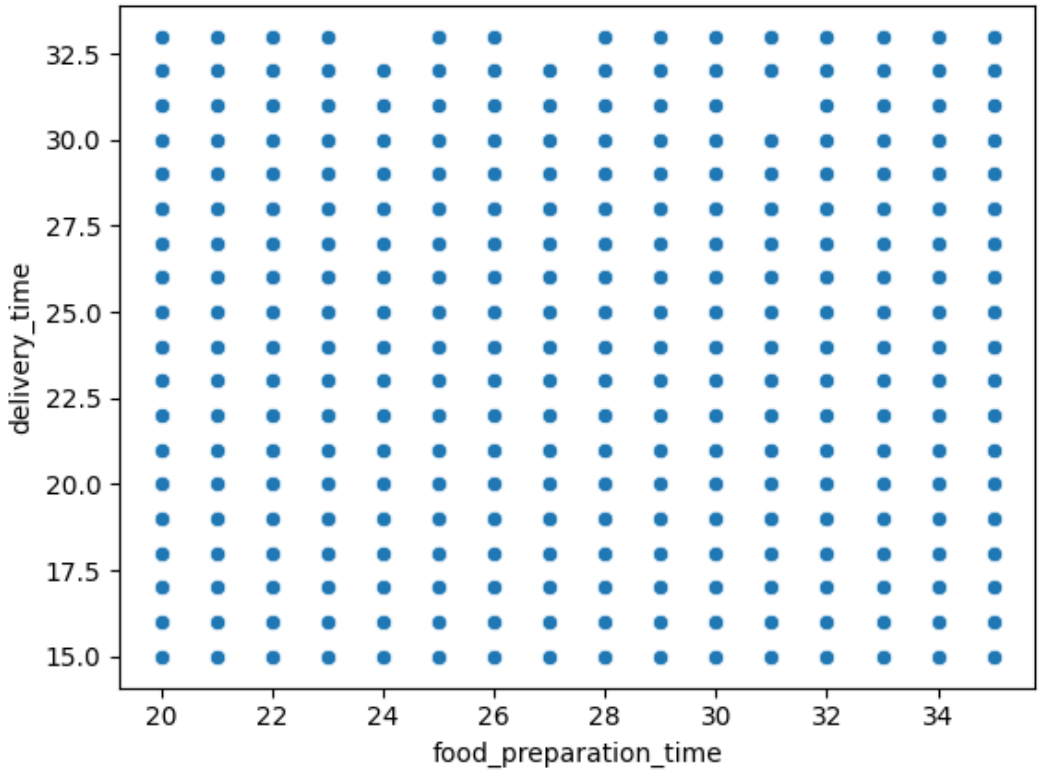
```
sns.scatterplot(data=df, x='cost_of_the_order', y='food_preparation_time');
```



Distribution of `food_preparation_time` ***vs*** `delivery_time` ***(Numerical Column vs Numerical Column)***

In [ ]:

```
sns.scatterplot(data=df, x='food_preparation_time', y='delivery_time');
```



Observations:

- **There are no correlation between** `cost_of_the_order` **vs** `delivery_time`**,** `cost_of_the_order` **vs** `food_preparation_time` **and** `food_preparation_time` **vs** `delivery_time`**.**
- **We are finding correlation among numerical columns so there is no need of** `order_id` **&** `customer_id` **columns.**

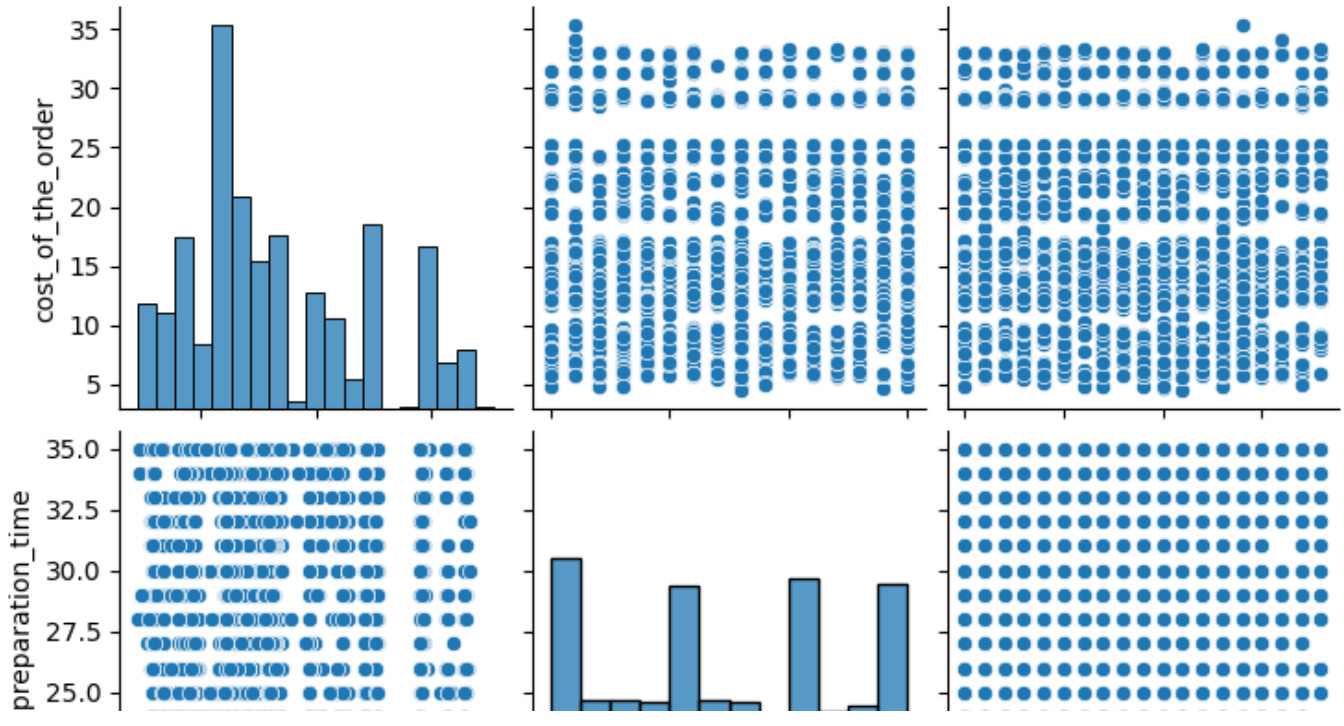
Distribution of *`All Numerical Columns (pairplot)`*

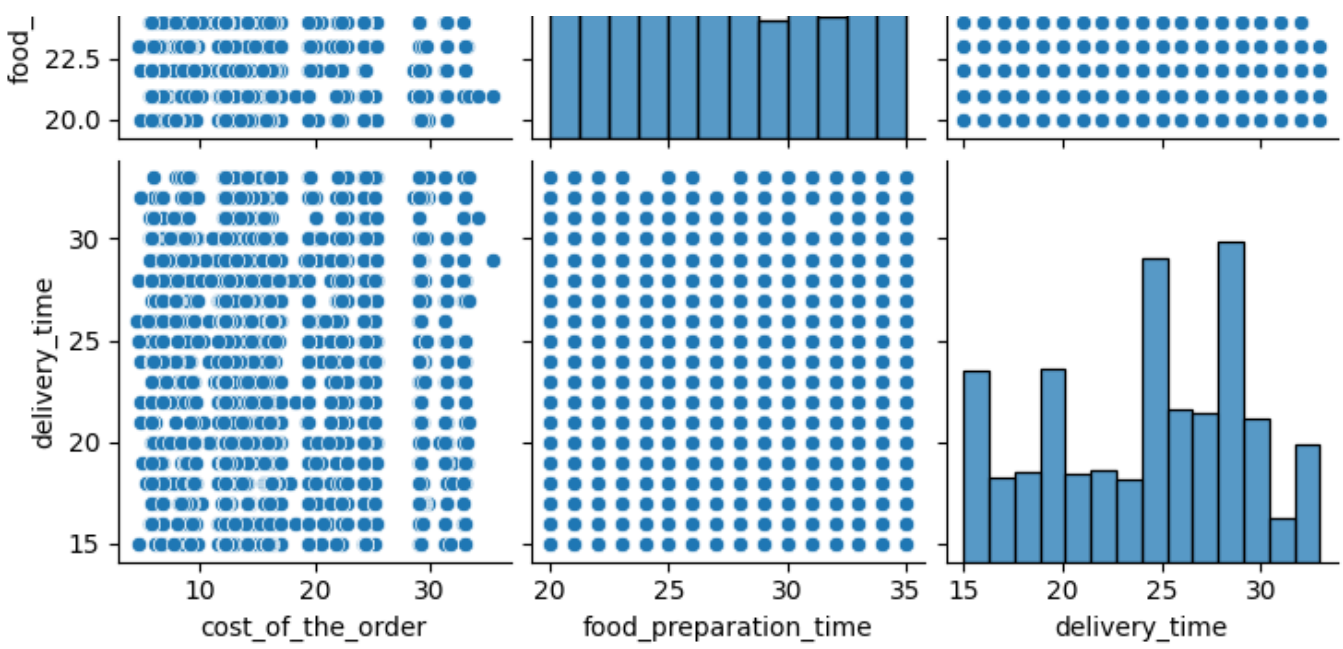
In [ ]:

```
numerical_columns = ['cost_of_the_order', 'food_preparation_time', 'delivery_time']
```

In [ ]:

```
sns.pairplot(data=df[numerical_columns]);
```





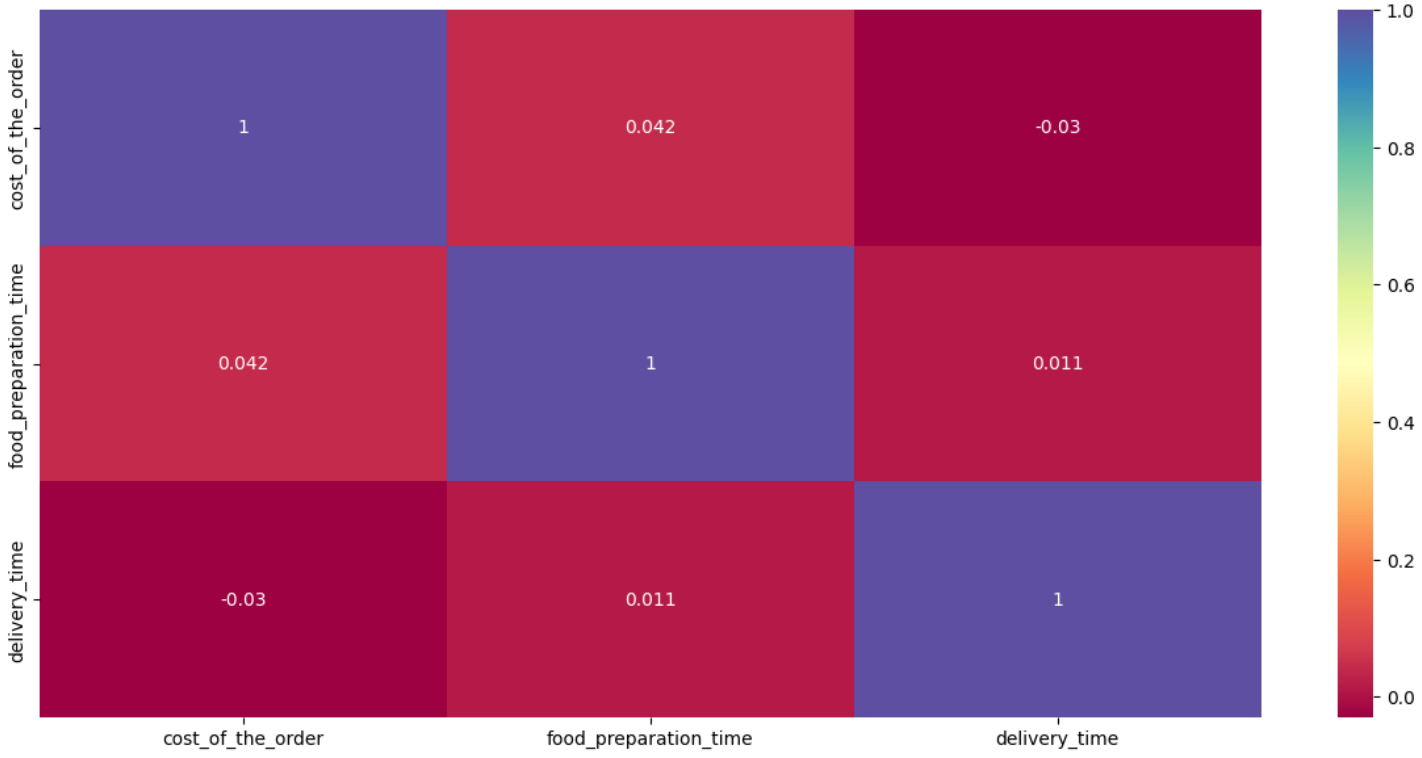
Observations:

- **As we can see, there is no correlation among them.**
- **Let's try another plot.**

Distribution of *All Numerical Columns (heatmap plot)*

In [ ]:

```
plt.figure(figsize=(15,7))
sns.heatmap(df[numerical_columns].corr(), annot=True, cmap='Spectral')
plt.xticks();
```



Observations:

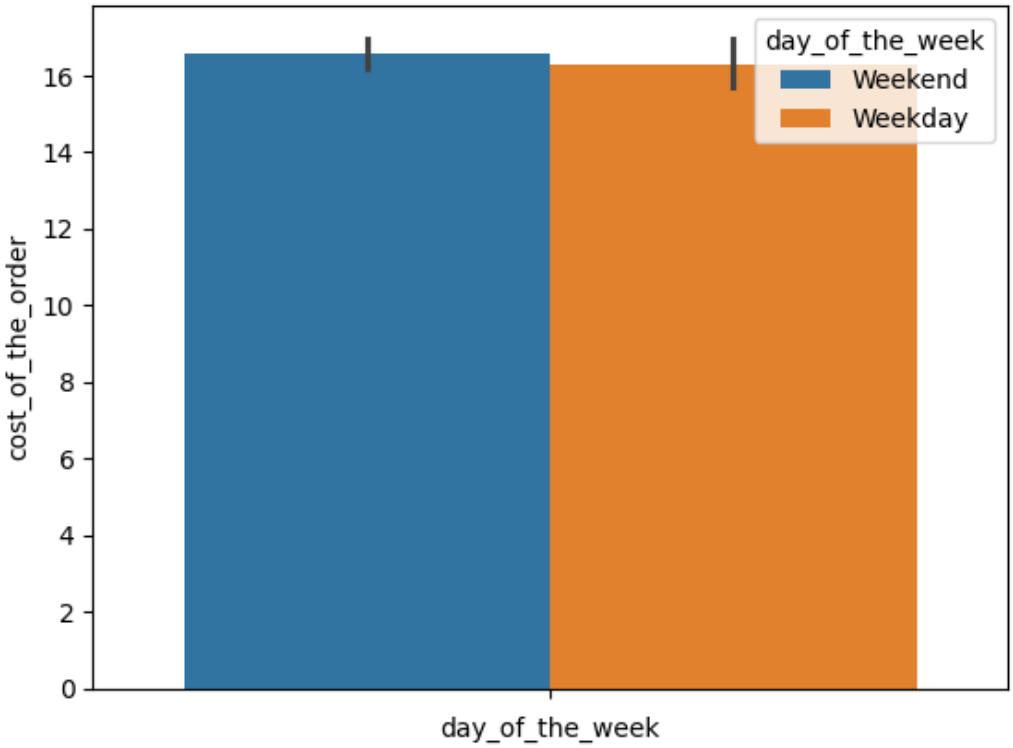
- **We can clearly seen that, there is very weak or no such relationship between any of the other variables.**

Multivariate Analysis for *(Numerical Variables vs Categorical Variables)*

Distribution of `cost_of_the_order` vs `day_of_the_week` (Numerical Column & Categorical column)

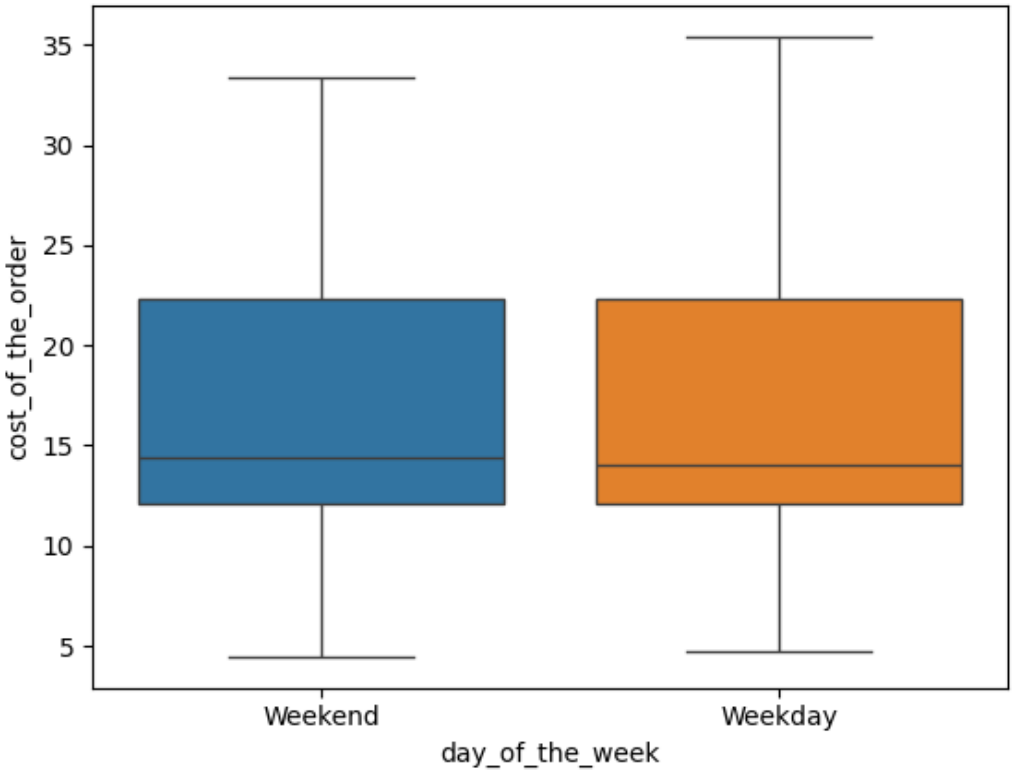
In [ ]:

```
sns.barplot(data=df, y='cost_of_the_order', hue='day_of_the_week')
plt.xlabel("day_of_the_week");
```



In [26]:

```
sns.boxplot(data=df, x="day_of_the_week", y='cost_of_the_order', hue='day_of_the_week');
```



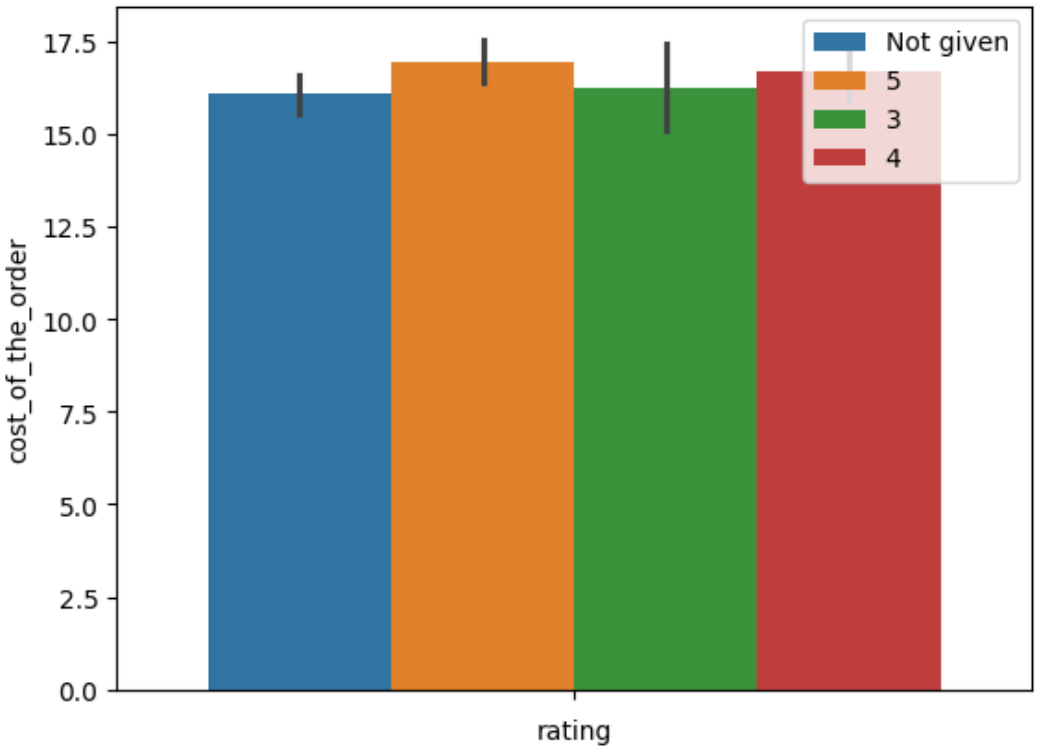
Observations:

- `cost_of_the_order` **are almost same in both Weekend & Weekdays.**

Distribution of `cost_of_the_order` vs `rating` (Numerical Column & Categorical column)

In [ ]:

```
sns.barplot(data=df, y='cost_of_the_order', hue='rating')
plt.xlabel("rating")
plt.legend(loc='upper right');
```



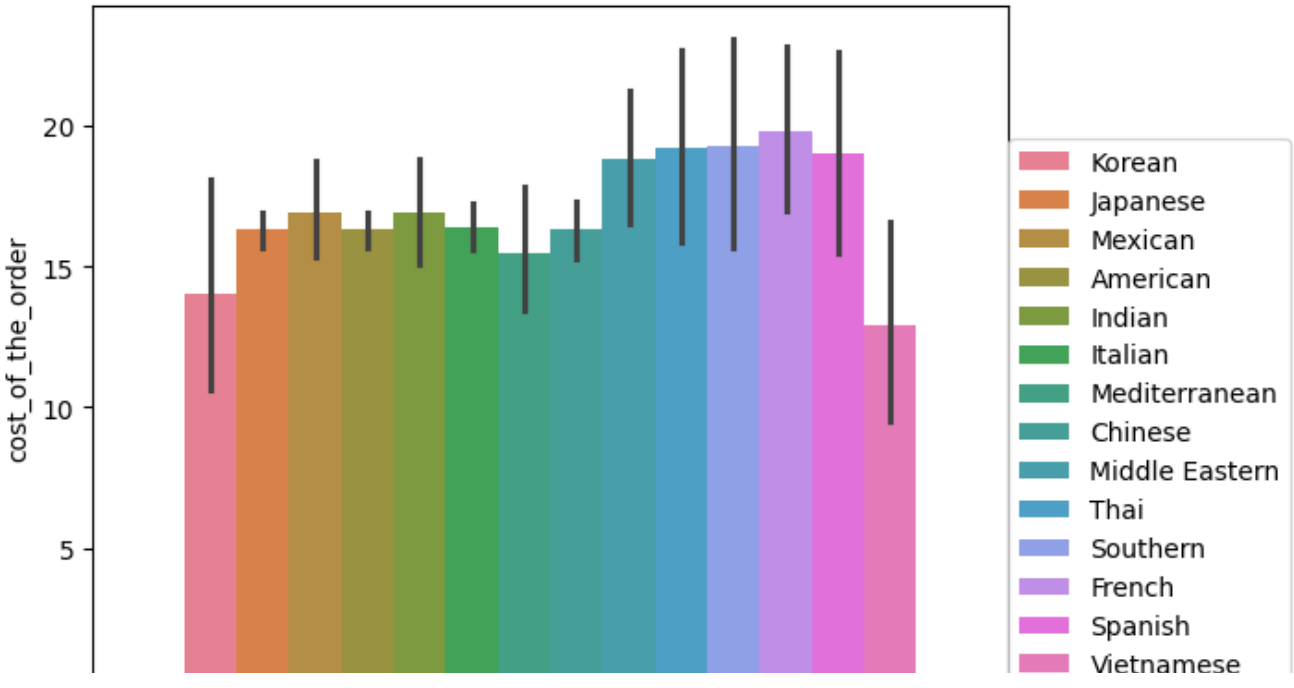
Observations:

- **There is no correlation between the `cost_of_the_order` and the `rating`.**

Distribution of `cost_of_the_order` vs `cuisine_type` (Numerical Column & Categorical column)

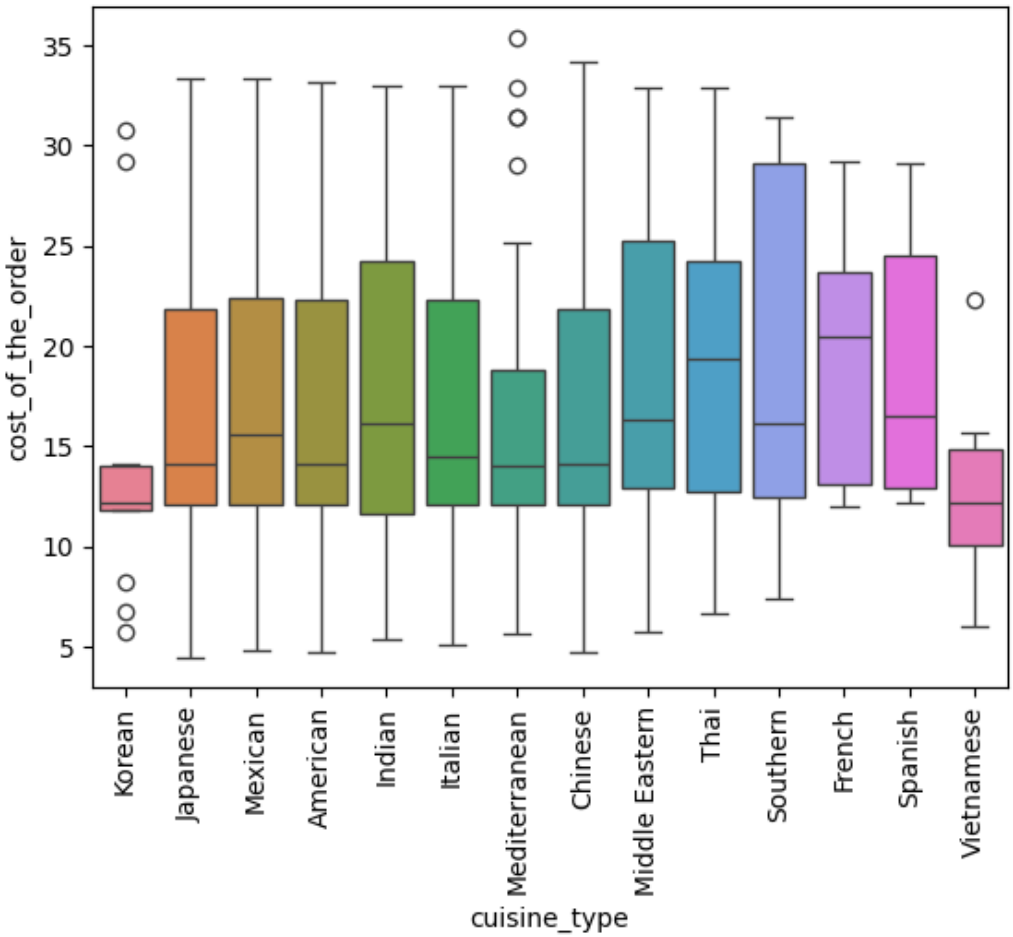
In [ ]:

```
sns.barplot(data=df, y='cost_of_the_order', hue='cuisine_type')
plt.xlabel("Cuisine Type")
plt.legend(loc=[1,0]);
```



In [31]:

```
sns.boxplot(data=df, x="cuisine_type", y='cost_of_the_order', hue='cuisine_type')
plt.xticks(rotation=90);
```



In [ ]:

```
df.groupby('cuisine_type')['cost_of_the_order'].agg(['max', 'min', 'mean']).sort_values(
'mean', ascending=False)
```

Out[ ]:

cuisine_type	max	min	mean
French	29.25	11.98	19.793889
Southern	31.43	7.38	19.300588
Thai	32.93	6.69	19.207895
Spanish	29.10	12.13	18.994167
Middle Eastern	32.93	5.77	18.820612
Mexican	33.32	4.85	16.933117
Indian	33.03	5.34	16.919726
Italian	33.03	5.05	16.418691
American	33.18	4.71	16.319829
Chinese	34.19	4.75	16.305209
Japanese	33.37	4.47	16.304532
Mediterranean	35.41	5.67	15.474783
Korean	30.75	5.77	14.001538

	max	min	mean
Vietnamese	22.26	6.01	12.882857

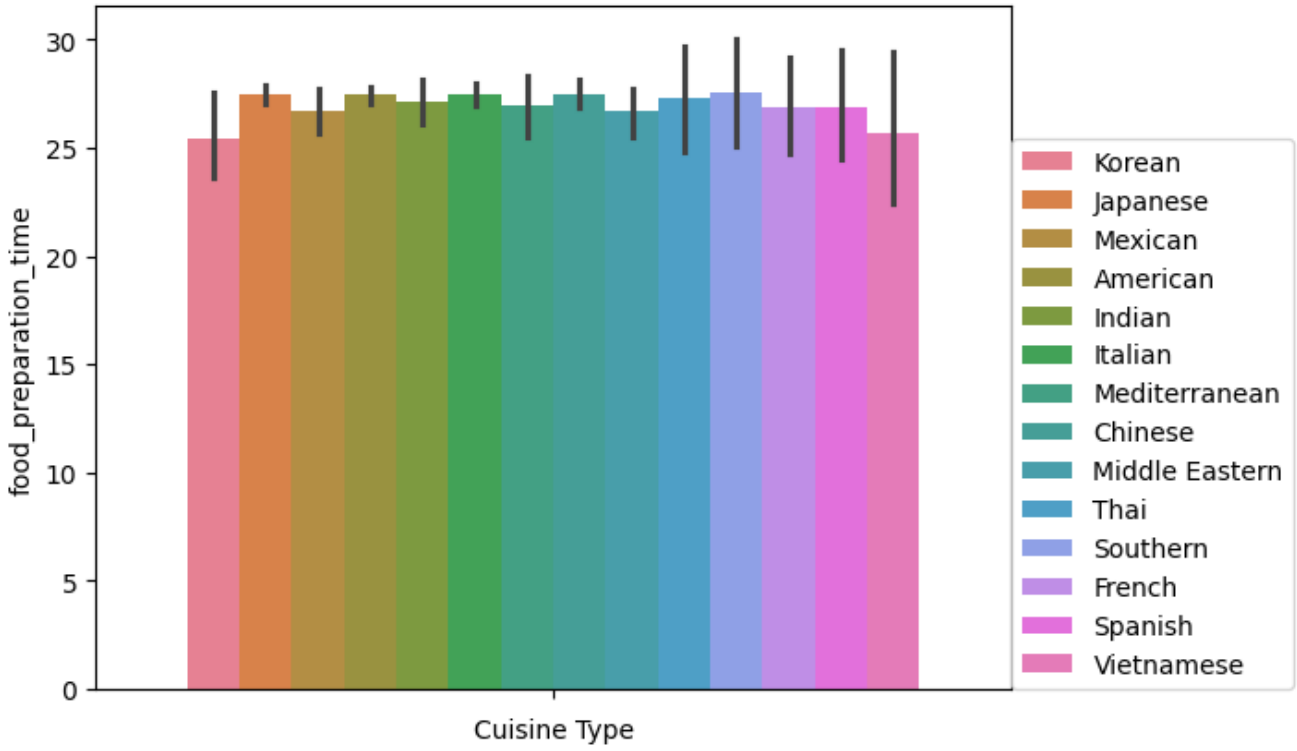
Observations:

- **There is correlation between the `cost_of_the_order` and the `cuisine_type`.**
- **The most expensive `cuisine_type` on average is `French`, followed by `Southern`, `Thai` and others.**
- **The least expensive `cuisine_type` on average is `Vietnamese`, followed by `Korean`, `Mediterranean`.**

Distribution of `food_preparation_time` vs `cuisine_type` (Numerical Column & Categorical column)

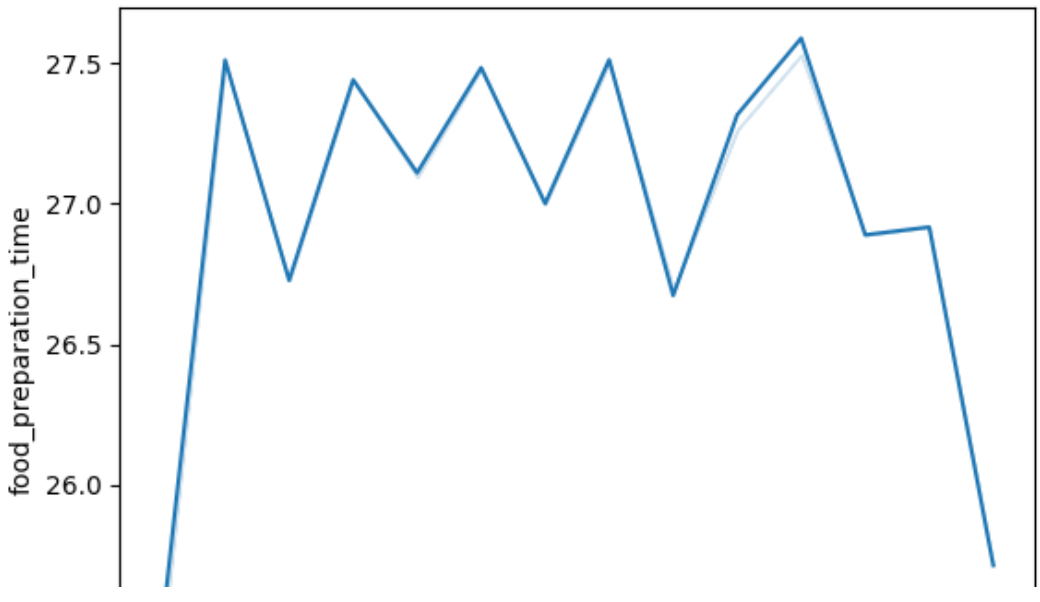
In [ ]:

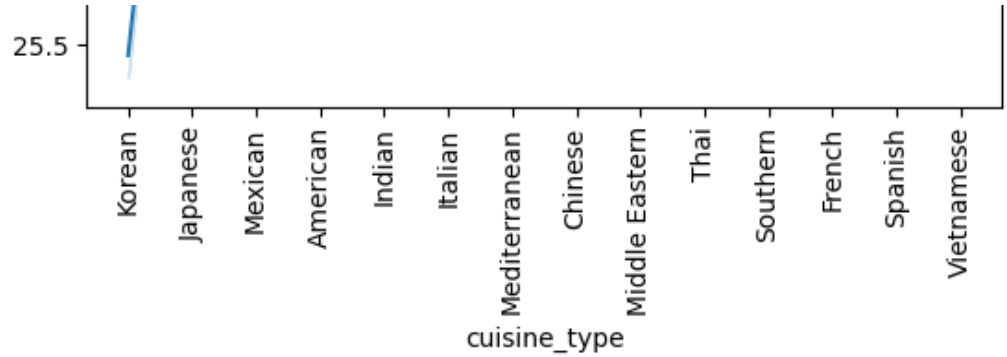
```
sns.barplot(data=df, y='food_preparation_time', hue='cuisine_type')
plt.xlabel("Cuisine Type")
plt.legend(loc=[1,0]);
```



In [22]:

```
sns.lineplot(data=df, y='food_preparation_time', x='cuisine_type', errorbar=("ci", False)
)
plt.xticks(rotation=90);
```





In [ ]:

```
df.groupby('cuisine_type')['food_preparation_time'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[ ]:

cuisine_type	max	min	mean
Southern	35	20	27.588235
Chinese	35	20	27.511628
Japanese	35	20	27.510638
Italian	35	20	27.483221
American	35	20	27.440068
Thai	35	21	27.315789
Indian	35	20	27.109589
Mediterranean	35	20	27.000000
Spanish	35	20	26.916667
French	35	21	26.888889
Mexican	35	20	26.727273
Middle Eastern	34	20	26.673469
Vietnamese	33	20	25.714286
Korean	33	20	25.461538

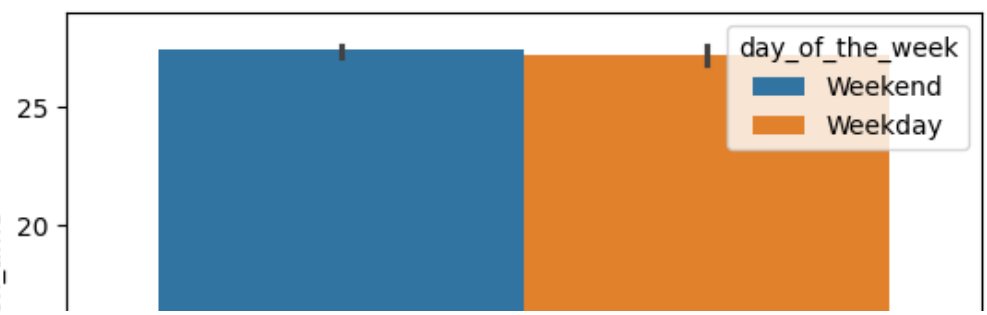
Observations:

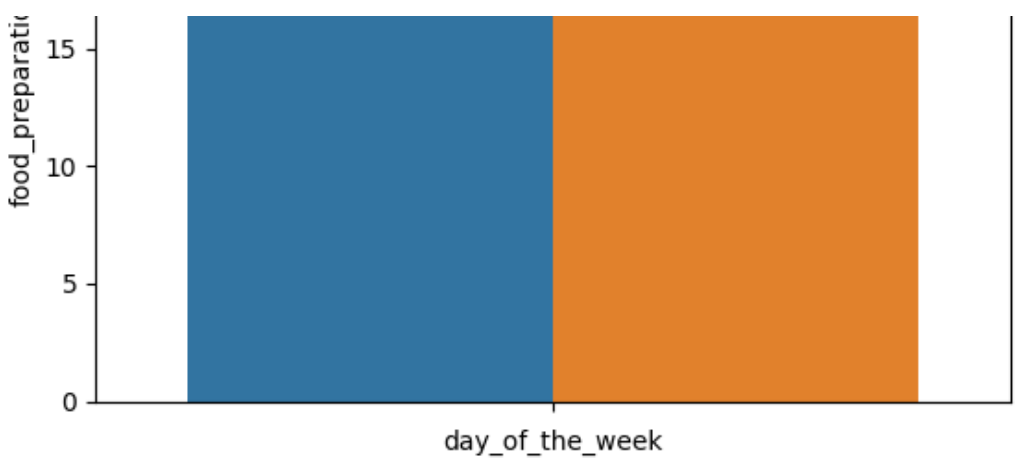
- **All cuisine have nearly the same average** `food_preparation_time`.

Distribution of *`food_preparation_time`* ***vs*** *`day_of_the_week`* ***(Numerical Column & Categorical column)***

In [ ]:

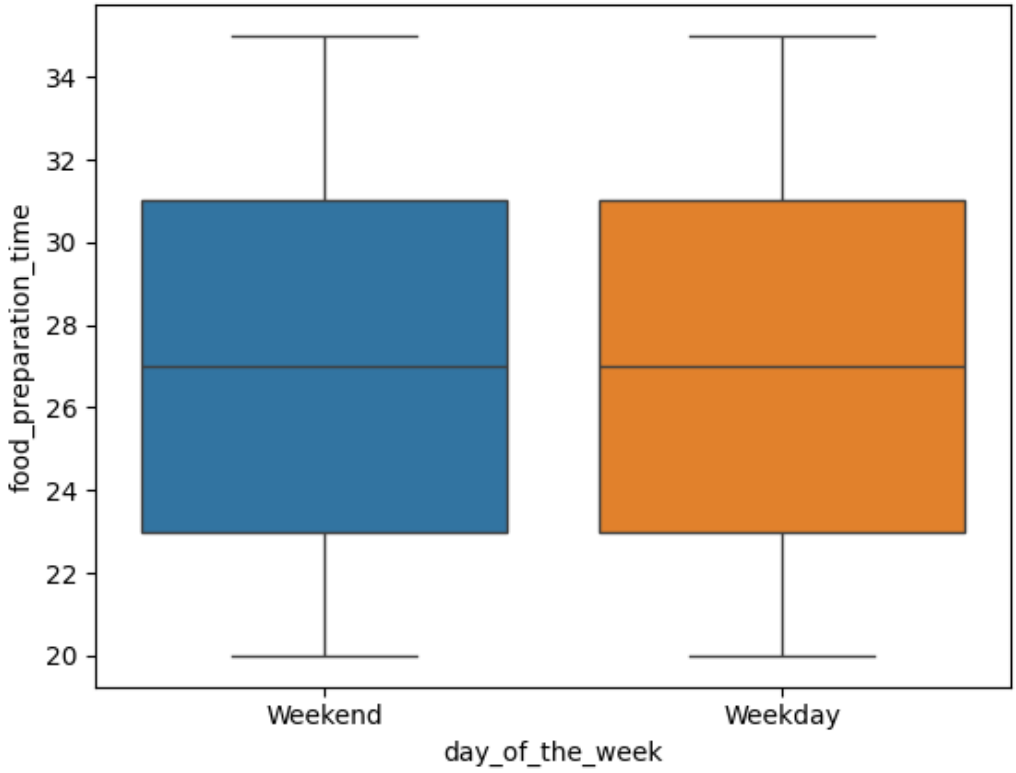
```
sns.barplot(data=df, y='food_preparation_time', hue='day_of_the_week')
plt.xlabel("day_of_the_week");
```





In [27]:

```
sns.boxplot(data=df, x="day_of_the_week", y='food_preparation_time', hue='day_of_the_wee
k');
```



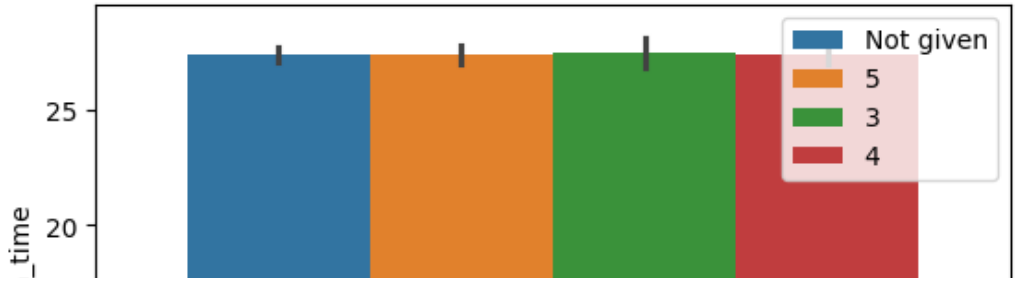
Observations:

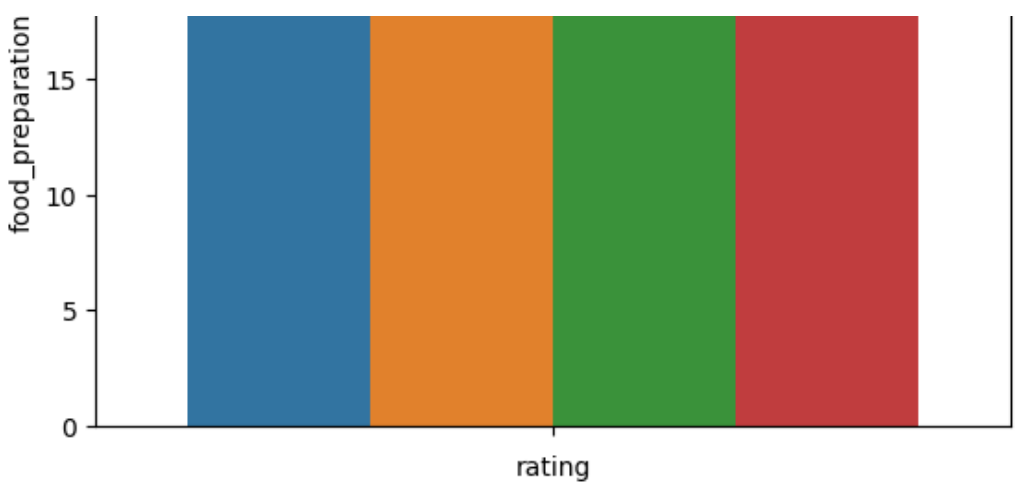
- `food_preparation_time` **is the same on both Weekends & Weekdays.**

Distribution of *`food_preparation_time`* ***vs*** *`rating`* ***(Numerical Column & Categorical column)***

In [ ]:

```
sns.barplot(data=df, y='food_preparation_time', hue='rating')
plt.xlabel("rating")
plt.legend(loc='upper right');
```





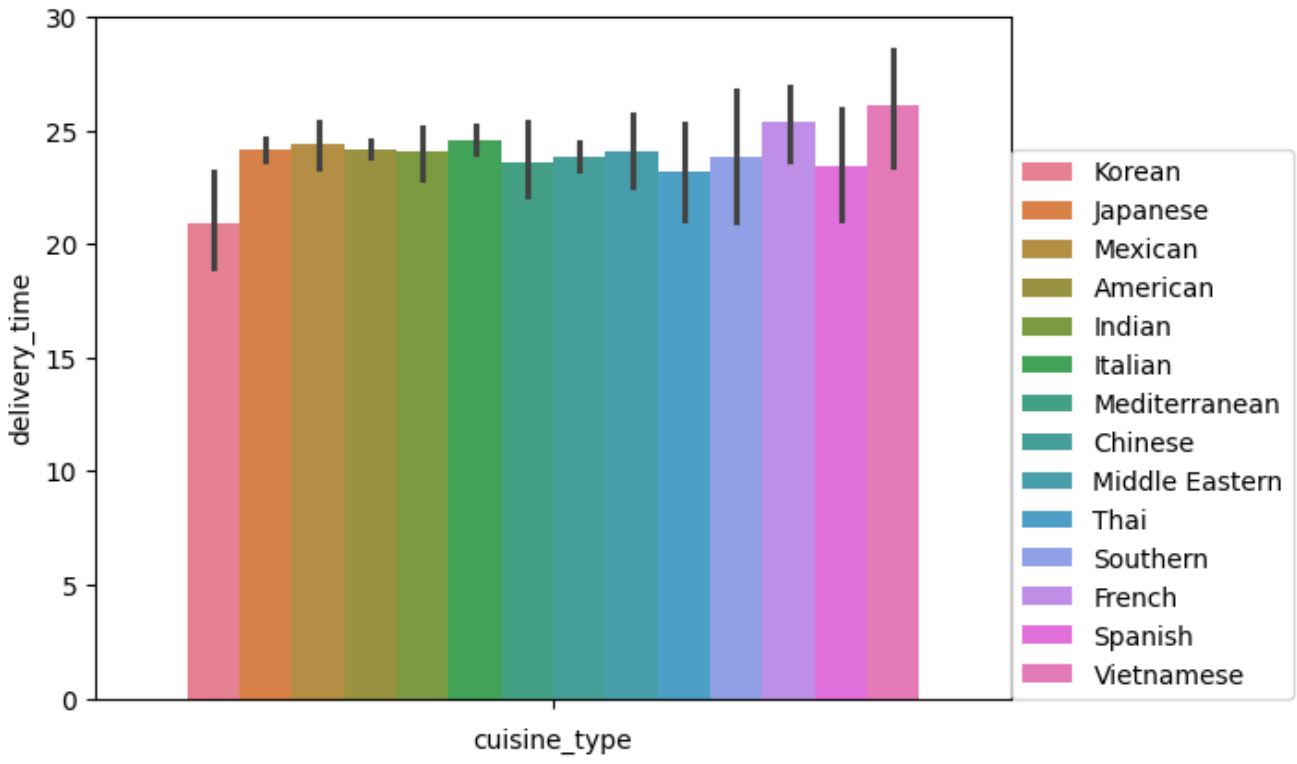
Observations:

- **There is no correlation between the `food_preparation_time` and the `rating`.**

Distribution of `delivery_time` vs `cuisine_type` (Numerical Column & Categorical column)

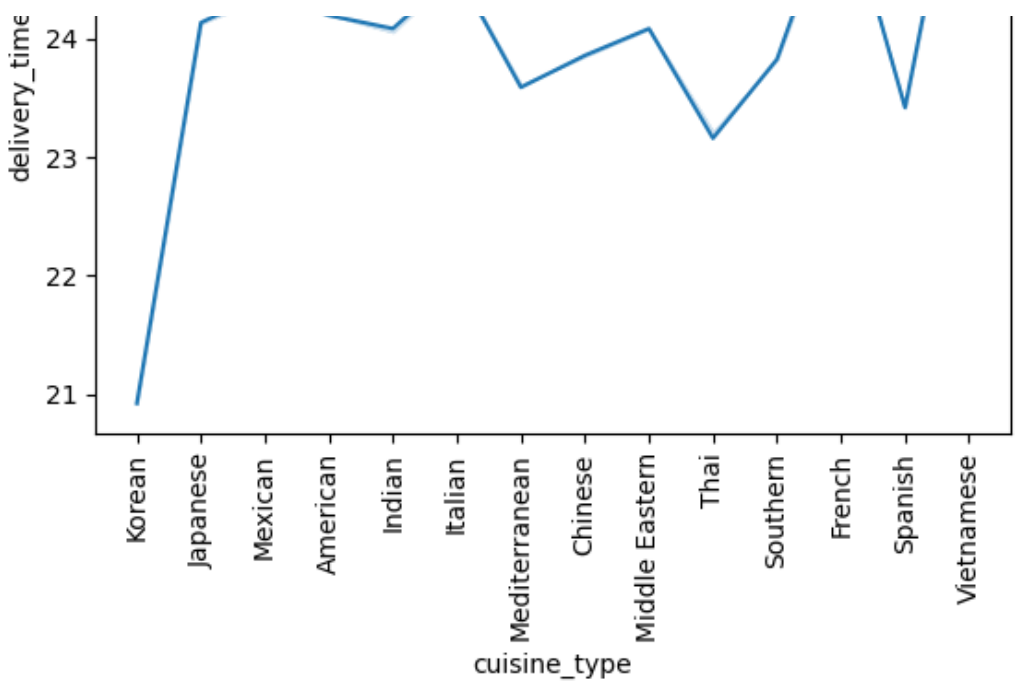
In [ ]:

```
sns.barplot(data=df, y='delivery_time', hue='cuisine_type')
plt.xlabel("cuisine_type")
plt.legend(loc=[1,0]);
```



In [24]:

```
sns.lineplot(data=df, x='cuisine_type', y='delivery_time', errorbar=("ci", False))
plt.xticks(rotation=90);
```



In [ ]:

```
df.groupby('cuisine_type')['delivery_time'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[ ]:

cuisine_type	max	min	mean
Vietnamese	31	19	26.142857
French	29	17	25.333333
Italian	33	15	24.567114
Mexican	33	16	24.389610
American	33	15	24.193493
Japanese	33	15	24.131915
Indian	32	15	24.082192
Middle Eastern	33	15	24.081633
Chinese	33	15	23.855814
Southern	33	15	23.823529
Mediterranean	33	15	23.586957
Spanish	30	17	23.416667
Thai	32	15	23.157895
Korean	26	16	20.923077

Observations:

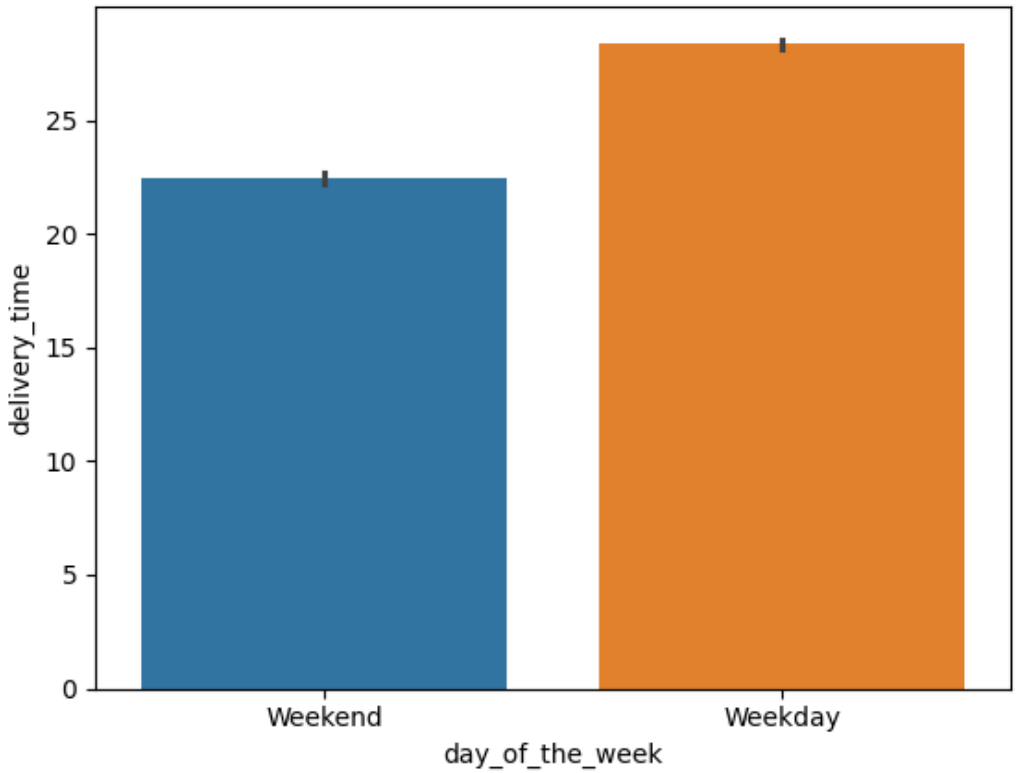
- **As per graph & table there is correlation between the `delivery_time` and the `cuisine_type`.**
- **`Vietnamese` cuisine takes more `delivery_time` on average, followed by `French`, `Italian` and others.**
- **`Korean` cuisine takes the least `delivery_time` on average.**

Distribution of `delivery_time` vs `day_of_the_week` (Numerical Column & Categorical column)

In [ ]:

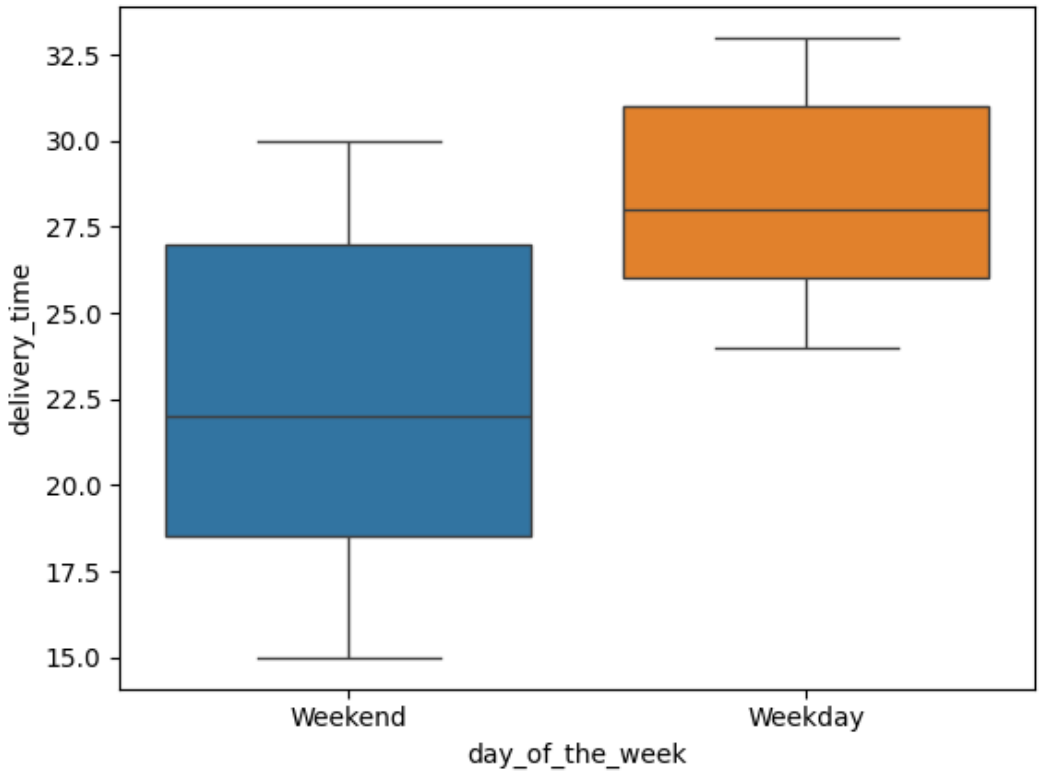
```
sns.barplot(data=df, x='day of the week', y='delivery time', hue='day of the week')
```

```
plt.xlabel("day_of_the_week");
```



In [28]:

```
sns.boxplot(data=df, x="day_of_the_week", y='delivery_time', hue='day_of_the_week');
```



In [ ]:

```
df.groupby('day_of_the_week')['delivery_time'].agg(['max', 'min', 'mean']).sort_values('
mean', ascending=False)
```

Out[ ]:

	max	min	mean
day_of_the_week			
Weekday	33	24	28.340037

```
Weekend    30    15    22.470022
           max   min   mean
```

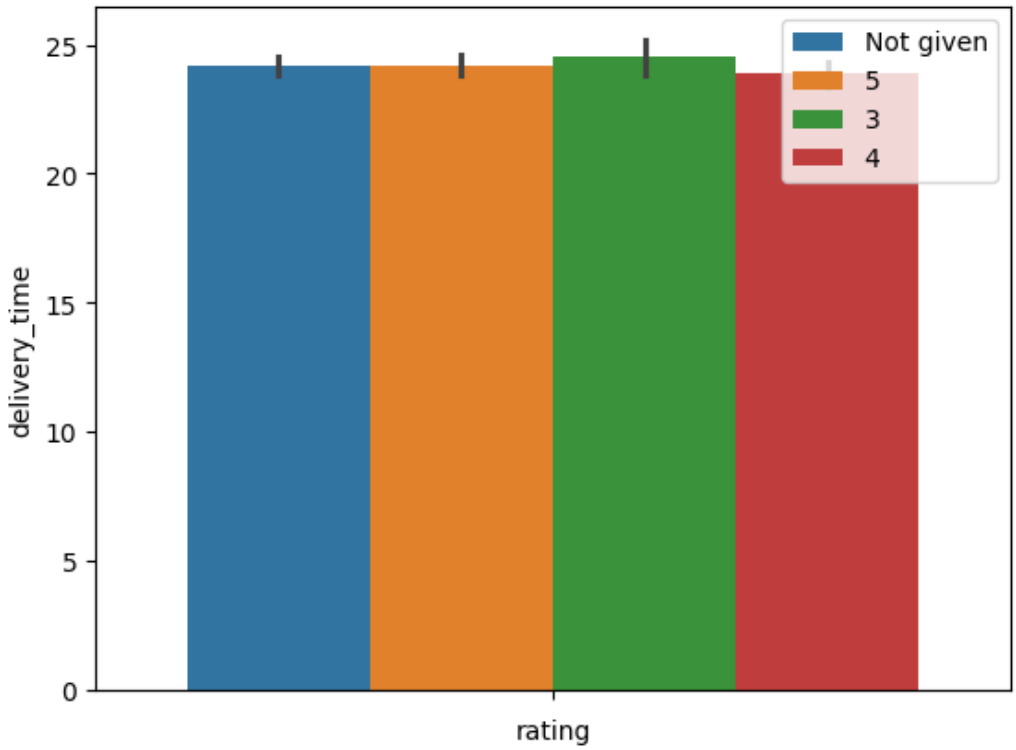
Observations:

- `delivery_time` **depends on** `day_of_the_week`**. It is shorter on weekends compared to weekdays.**

Distribution of *`delivery_time`* ***vs*** *`rating`* ***(Numerical Column & Categorical column)***

In [ ]:

```
sns.barplot(data=df, y='delivery_time', hue='rating')
plt.xlabel("rating")
plt.legend(loc='upper right');
```



Observations:

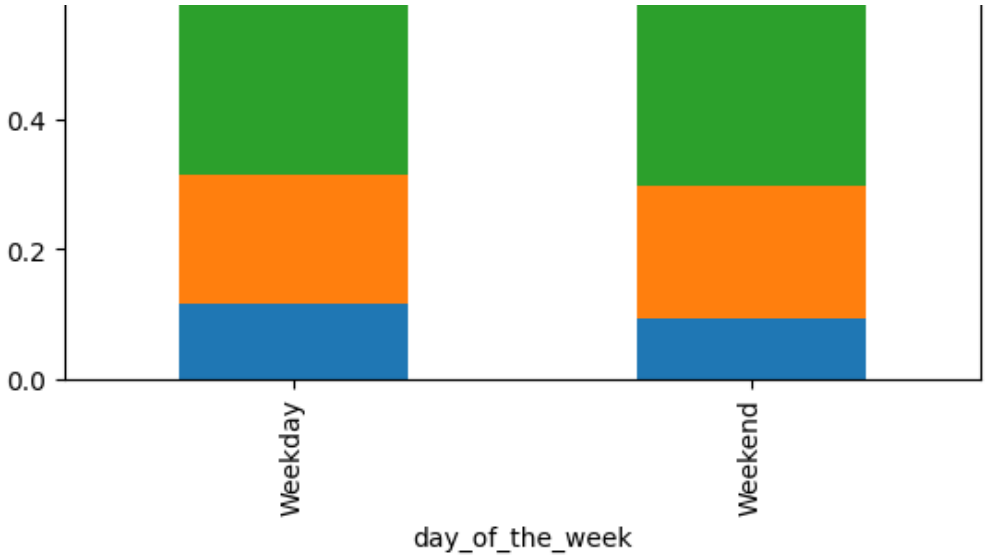
- **There is no correlation between the** `delivery_time` **and the** `rating`**.**

Multivariate Analysis for *(Categorical Variables vs Categorical Variables)*

Distribution of *`day_of_the_week`* ***vs*** *`rating`* ***(Categorical Column & Categorical column)***

In [ ]:

```
pd.crosstab(df['day_of_the_week'], df['rating'], normalize = 'index').plot.bar(stacked =
True)
plt.legend(loc='upper right');
```



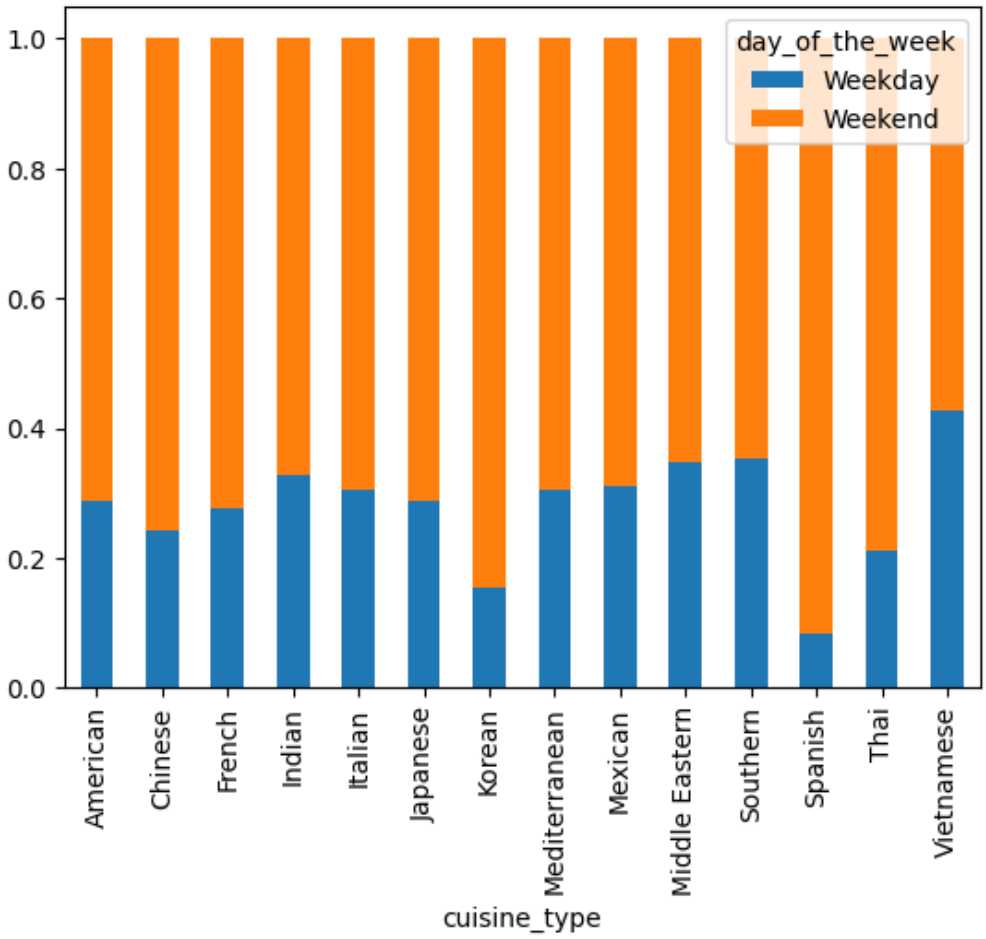
Observations:

- `Rating` **values are not affected by whether it is a Weekday or a Weekend.**

Distribution of *`cuisine_type`* ***vs*** *`day_of_the_week`* ***(Categorical Column & Categorical column)***

In [ ]:

```
pd.crosstab(df['cuisine_type'], df['day_of_the_week'], normalize = 'index').plot.bar(sta
cked = True);
```



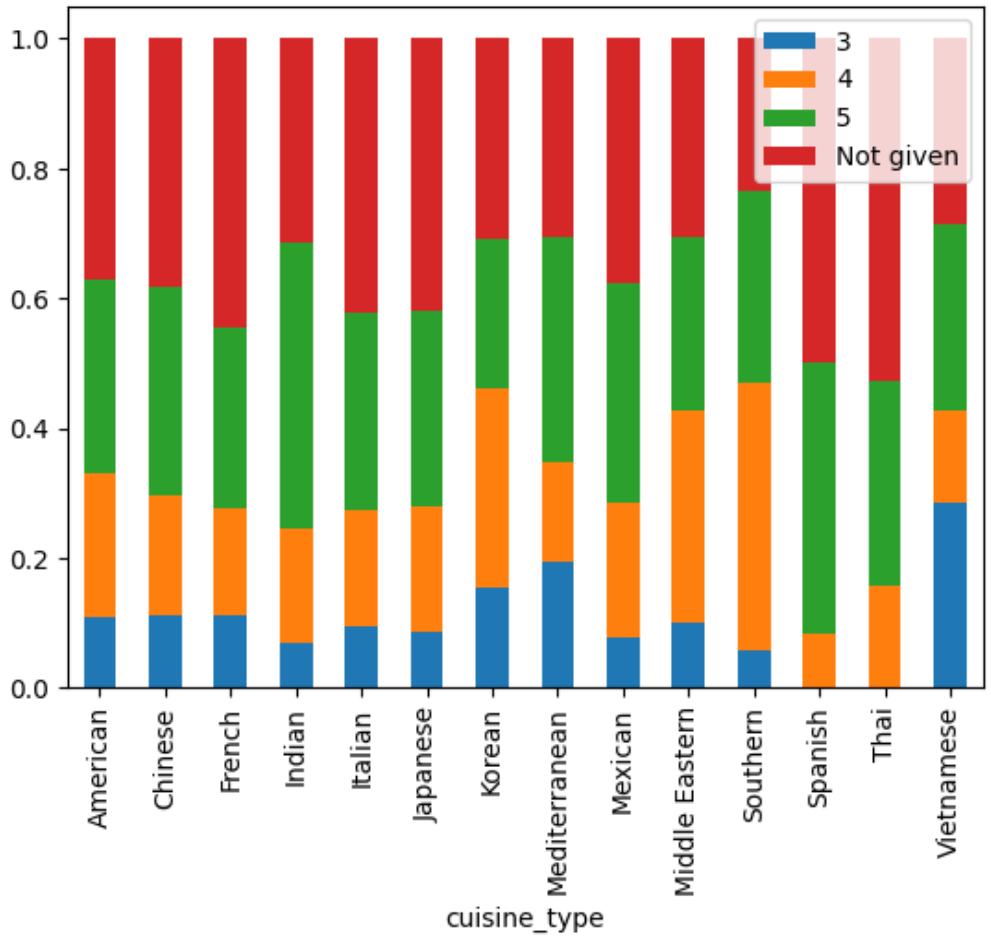
Observations:

- **There is correlation between the** `day_of_the_week` **and the** `cuisine_type`**, more cusines are ordered on Weekends than on Weekdays.**

Distribution of `cuisine_type` vs `rating` (Categorical Column & Categorical column)

In [ ]:

```
pd.crosstab(df['cuisine_type'], df['rating'], normalize = 'index').plot.bar(stacked = Tr
ue)
plt.legend(loc='upper right');
```



Observations:

- **As expected, there is no relationship between `cuisine_type` and `rating`.**

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]

We can achieve this requirements in 5 steps:

- **Step 1: to remove not rating observations from the dataset,**
- **Step 2. to calculate number of rating from the new data frame,**
- **Step 3. to convert rating data type from `object` to `int`,**
- **Step 4. to count number of ratings & average rating for all restaurants,**
- **Step 5. Apply both the conditions (i.e. restaurants have more than 50 ratings and average rating greater than 4).**

In [6]:

```
# Step1: We need to remove not rating observations from the dataset
new_df = df[df['rating'] != 'Not given'].reset_index()
new_df
```

Out[6]:

Out[6]:

	index	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_prepara
0	2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	
1	3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	
2	4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	
3	5	1477224	147468	Tamarind TriBeCa	Indian	25.22	Weekday	3	
4	7	1477859	89574	Barbounia	Mediterranean	5.97	Weekday	3	
...	...	...	...	...	...	...	...	...	
1157	1889	1478190	94152	RedFarm Broadway	Chinese	8.68	Weekday	3	
1158	1890	1477316	164776	TAO	Japanese	15.67	Weekend	5	
1159	1893	1476701	292602	Chipotle Mexican Grill $1.99 Delivery	Mexican	22.31	Weekend	5	
1160	1894	1477421	397537	The Smile	American	12.18	Weekend	5	
1161	1896	1477513	64151	Jack's Wife Freda	Mediterranean	12.18	Weekday	5	

1162 rows × 10 columns

In [7]:

```
# Step2: We have to calculate number of rating from the new data frame
rating_rest_df = new_df[['restaurant_name', 'rating']]
rating_rest_df
```

Out[7]:

	restaurant_name	rating
0	Cafe Habana	5
1	Blue Ribbon Fried Chicken	3
2	Dirty Bird to Go	4
3	Tamarind TriBeCa	3
4	Barbounia	3
...	...	...
1157	RedFarm Broadway	3
1158	TAO	5
1159	Chipotle Mexican Grill $1.99 Delivery	5
1160	The Smile	5
1161	Jack's Wife Freda	5

1162 rows × 2 columns

In [10]:

```
# Step 3: We can convert rating object into integer
rating_rest_df.loc[:, 'rating'] = rating_rest_df.loc[:, 'rating'].astype("int")
```

In [11]:

```
# Step4: We have to count number of ratings & avg rating for all restaurants
rest_rating_group = rating_rest_df.groupby('restaurant_name')['rating'].agg(['count', 'm
```

```
ean'])
rest_rating_group
```

Out[11]:

restaurant_name	count	mean
'wichcraft	1	5.0
12 Chairs	2	4.5
5 Napkin Burger	2	4.0
67 Burger	1	5.0
Amma	2	4.5
...	...	...
Zero Otto Nove	1	4.0
brgr	1	3.0
da Umberto	1	5.0
ilili Restaurant	13	4.153846
indikitch	2	4.5

156 rows × 2 columns

In [ ]:

```
# Step4: Apply both the conditions
# first condition: filter those restaurant whose number of rating count is greater that 5
0
# Second condition: filter out those restaurant whose avg rating is greater than 4
rest_50_rating = rest_rating_group[(rest_rating_group['count'] > 50) & (rest_rating_grou
p['mean'] > 4)].sort_values('count', ascending=False)
rest_50_rating
```

Out[ ]:

restaurant_name	count	mean
Shake Shack	133	4.278195
The Meatball Shop	84	4.511905
Blue Ribbon Sushi	73	4.219178
Blue Ribbon Fried Chicken	64	4.328125

Observations:

- **The resturants that have more than 50 ratings & average ratings are grater than 4 are (sort by count i.e. total number of ratings):**
  1. **Shake Shack**
  2. **The Meatball Shop**
  3. **Blue Ribbon Sushi**
  4. **Blue Ribbon Fried Chicken**

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]

We can achieve this requirements in two steps:

- **Step 1: Create new column `revenue` in the data frame & apply conditions**
- **Step 2: after validation we can find sum of `revenue` column**

In [ ]:

```
# Step 1: Create new column revenue in the data frame & apply conditions
df["revenue"] = df['cost_of_the_order'].apply(lambda cost: cost*0.25 if cost > 20 else (
cost*0.15 if cost > 5 else 0))
df.sort_values('revenue', ascending=False)
```

Out[ ]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_ti
573	1477814	62359	Pylos	Mediterranean	35.41	Weekday	4	
1646	1477665	231061	Han Dynasty	Chinese	34.19	Weekday	Not given	
1762	1477700	60039	Blue Ribbon Sushi	Japanese	33.37	Weekday	3	
1831	1476970	275689	Nobu Next Door	Japanese	33.37	Weekend	4	
1370	1478329	116992	Tres Carnes	Mexican	33.32	Weekday	4	
...	...	...	...	...	...	...	...	
1569	1477786	145389	RedFarm Hudson	Chinese	4.75	Weekend	3	
889	1477787	14869	Shake Shack	American	4.80	Weekend	4	
624	1477349	52327	Nobu Next Door	Japanese	4.47	Weekend	5	
542	1477788	270444	P.J. Clarke's	American	4.71	Weekend	Not given	
1695	1478302	318665	Blue Ribbon Sushi Bar & Grill	Japanese	4.90	Weekday	4	

1898 rows × 10 columns

In [ ]:

```
# Step 2: after validation we can find sum of revenue column
total_revenue =df['revenue'].sum()
print(f"Net revenue generated by the company across all orders is {round(total_revenue,2)
} dollars.")
```

```
Net revenue generated by the company across all orders is 6166.3 dollars.
```

In [ ]:

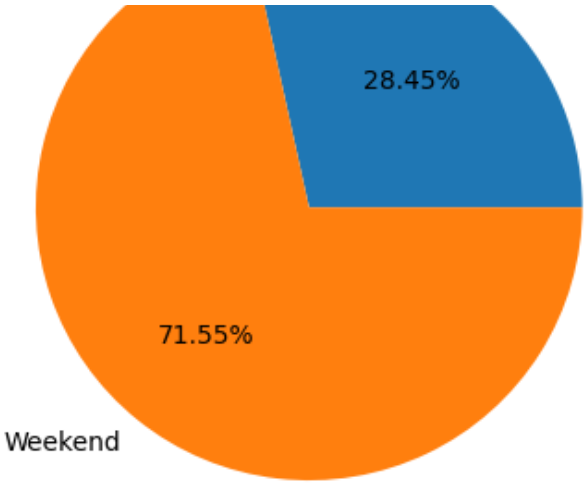
```
# Generate revenue on Weekend & Weekdays
df.groupby('day_of_the_week')['revenue'].sum()
```

Out[ ]:

```
day_of_the_week
Weekday    1754.3345
Weekend    4411.9685
Name: revenue, dtype: float64
```

In [ ]:

```
plt.pie(df.groupby('day_of_the_week')['revenue'].sum(), labels=df.groupby('day_of_the_we
ek')['revenue'].sum().index, autopct='%0.2f%%');
```



Observations:

- **Total `revenue` generated by the company across all orders is around `$6166`.**
- **As expected, the `revenue` generated by the company across all orders is higher on weekends compared to weekdays due to the higher order count on weekends.**
- **Around `72%` of the `revenue` is generated from Weekends's orders.**

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]

We can achieve this requirements in two steps:

- **Step 1: Create new column `total_time_required` in the data frame then,**
- **Step 2: Find percentage after applying conditions**

In [ ]:

```
# Step 1: Create new column total_time_required in the data frame
df["total_time_required"] = df['food_preparation_time'] + df['delivery_time']
df.head()
```

Out[ ]:

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

In [ ]:

```
# Step 2: Find percentage after applying conditons
percentage_ = df[df['total_time_required'] > 60].shape[0] / df.shape[0] * 100
print(f"Total percentage of orders take more than an hour (60 minutes) to get delivered f
rom the time the order is placed is {round(percentage_, 2)} %.")
```

```
Total percentage of orders take more than an hour (60 minutes) to get delivered from the
```

```
time the order is placed is 10.54 %.
```

Observations:

- `total_time_required` **column is sum of** `food_preparation_time` **&** `delivery_time` **columns.**
- **Total percentage of orders take** `more than an hour` **to get delivered from the time the order is placed is around** `10.54%`

Question 16: The company wants to analyze the delivery time of the orders on weekdays and weekends. How does the mean delivery time vary during weekdays and weekends? [2 marks]

In [ ]:

```
# Let's see average mean delivery time
df['delivery_time'].describe()
```

Out[ ]:

```
count    1898.000000
mean       24.161749
std         4.972637
min        15.000000
25%        20.000000
50%        25.000000
75%        28.000000
max        33.000000
Name: delivery_time, dtype: float64
```

In [ ]:

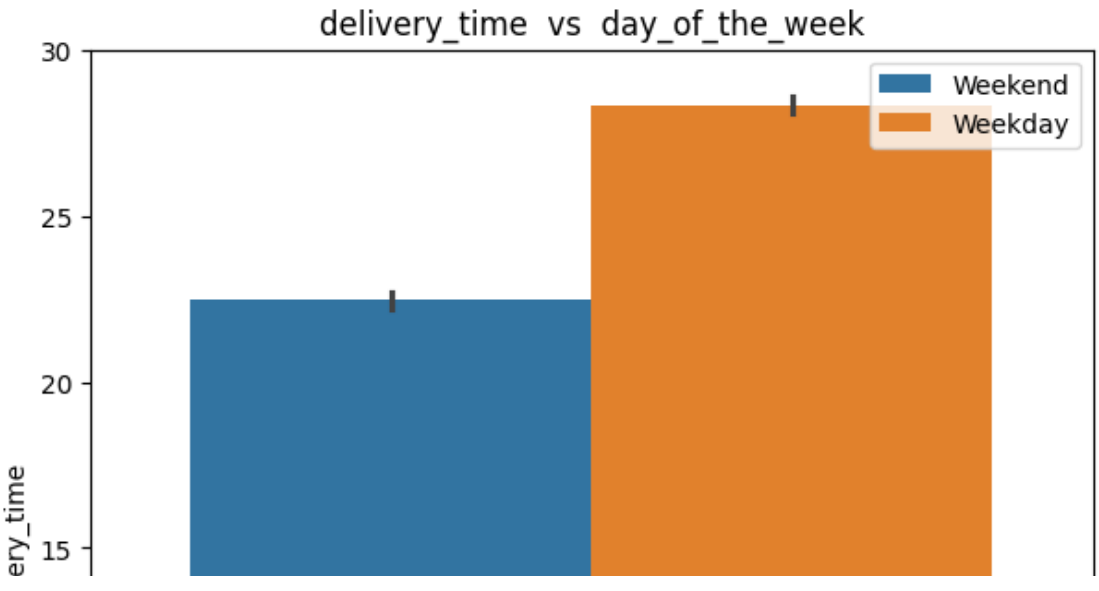
```
# Let's see average mean delivey time on Weekdays & Weekends
df.groupby('day_of_the_week')['delivery_time'].mean()
```

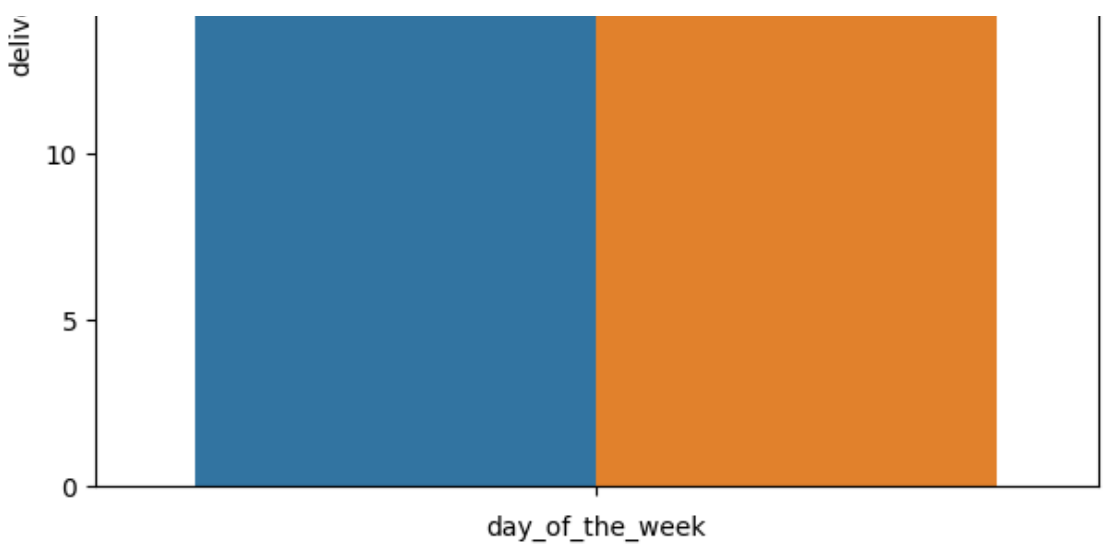
Out[ ]:

```
day_of_the_week
Weekday    28.340037
Weekend    22.470022
Name: delivery_time, dtype: float64
```

In [ ]:

```
# Plot 1: delivery_time vs day_of_the_week
plt.figure(figsize=(7,7))
sns.barplot(data=df, y='delivery_time', hue='day_of_the_week')
plt.title('delivery_time  vs  day_of_the_week')
plt.xlabel("day_of_the_week")
plt.legend(loc='upper right');
```





In [ ]:

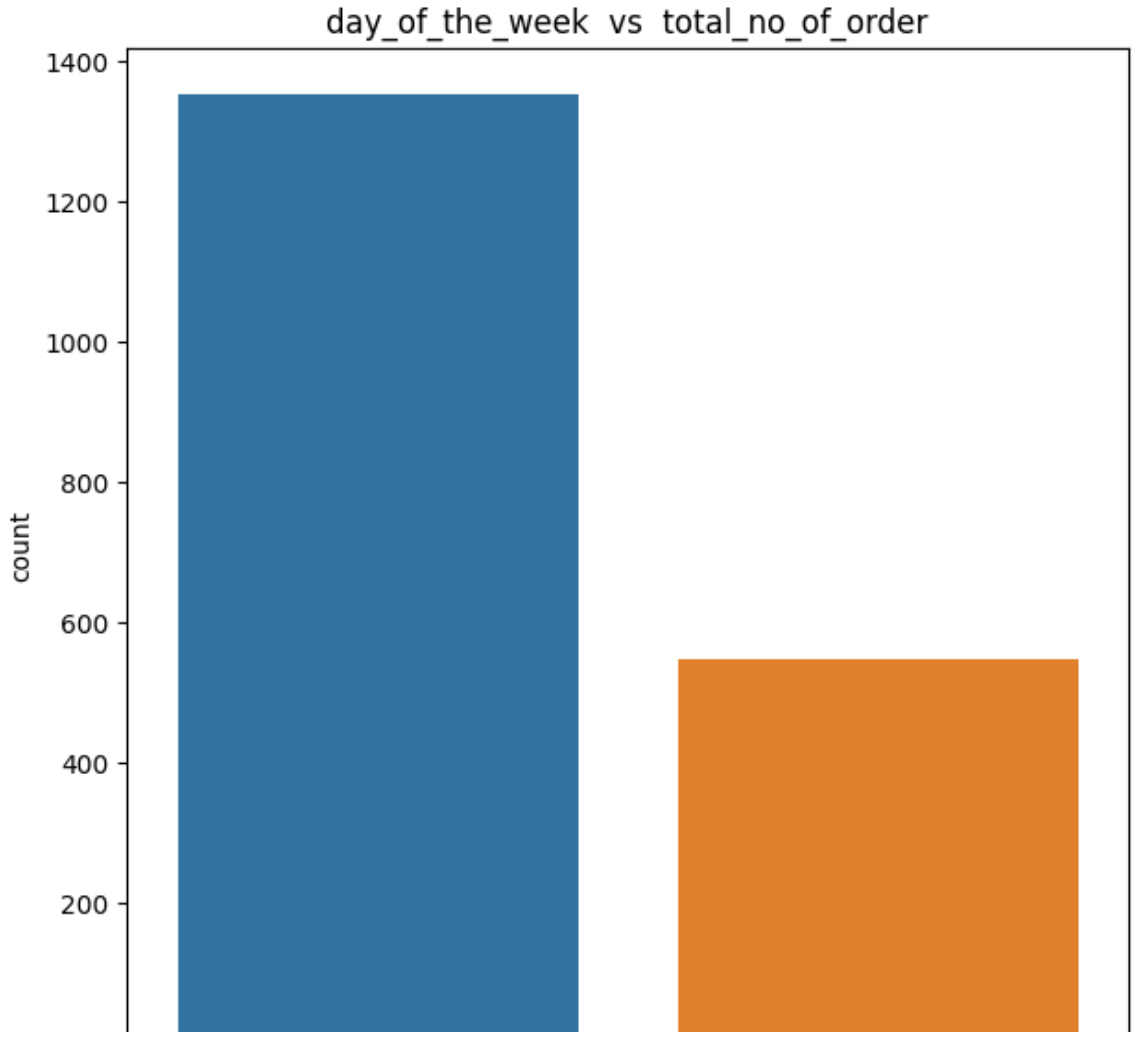
```
# Let's see total number of orders on Weekdays & Weekends
df['day_of_the_week'].value_counts()
```

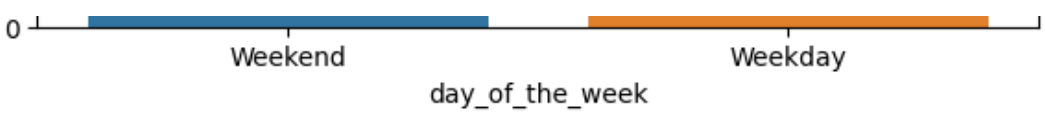
Out[ ]:

```
day_of_the_week
Weekend    1351
Weekday     547
Name: count, dtype: int64
```

In [ ]:

```
# Plot 2: day_of_the_week vs total_no_of_orders
plt.figure(figsize=(7,7))
sns.countplot(data=df, x='day_of_the_week', hue='day_of_the_week');
plt.title('day_of_the_week  vs  total_no_of_orders')
plt.xlabel("day_of_the_week");
```





Observations:

- **The average mean `delivery_time` is around 24 minutes.**
- **The average mean `delivery_time` on `Weekdays are 28 minutes` while on `Weekends are 22 minutes`. This is a strange or curious observation.**
- **As we can see there are difference between Weekdays & Weekends average delivery time, let's find what are factors.**
- **The number of orders on `Weekdays (547 orders)` is less than the number of orders on `Weekends (1351 orders)`.**
- **There is a `probability` of `heavier traffic` on `Weekdays` compared to `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:

- **About `71%` of the orders are placed by users on `Weekends`.**
- **Approximately `39%` of the orders are `not rated` by users.**
- **`Shake Shack` restaurant received the highest number of orders, followed by `The Meatball Shop`, `Blue Ribbon Sushi`, `Blue Ribbon Fried Chicken` and `Parm`.**
- **As expected, the number of orders on `weekends` is `higher than` on `weekdays` for the top 5 restaurants as well.**
- **Except `The Meatball Shop` restaurant all four restaurants out of the top 5 restaurants ordered one type of cuisine only.**
- **The most popular cuisine on weekends is `American` cuisine.**
- **The consistently favourite and popular cuisine is `American`. It remains the top choice both on weekends and weekdays, in this dataset.**
- **The most expensive `cuisine_type` on average is `French`, followed by `Southern`, `Thai` and others.**
- **The least expensive `cuisine_type` on average is `Vietnamese`, followed by `Korean`, `Mediterranean`.**
- **All cuisine have nearly the same average `food_preparation_time`.**
- **As expected, the revenue generated by the company across all orders is higher on weekends compared to weekdays due to the higher order count on weekends.**
- **Around 72% of the `revenue` is generated from Weekend's orders.**
- **The average mean `delivery_time` on `Weekdays are 28 minutes` while on `Weekends are 22 minutes`. This is a strange or curious observation.**
- **The number of orders on `Weekdays (547 orders)` is less than the number of orders on `Weekends (1351 orders)`.**
- **There is a `probability` of `heavier traffic` on `Weekdays` compared to `Weekends`.**

Recommendations:

- **The company should focus on `encouraging customers to rate their orders`, as around `39%` of the orders are `not rated`. By understanding customer satisfaction, the company can improve its services accordingly.**
- **The company should offer extra discounts, special offers, or rewards to their `most frequent customers`.**
- **The Company should `give offers` on `weekdays orders` like give promotional offers etc on weekdays so that number of orders will increase.**

- **The company should also focus on `reducing the mean delivery_time`, especially on `weekdays`, as `high delivery times` may be causing `low orders`.**
- **The highest number of orders has been received for `American` cuisine, followed by `Japanese` and `Italian cuisines`. The company should focus on improving the `total_time_required` (the sum of `food_preparation_time` and `delivery_time`) on both weekdays and weekends.**
- **The company should provide extra offers on cuisines that have received the least number of orders, such as `Vietnamese`, `Korean`, `Spanish`, `French` etc.**
- **The company should advertise most famous cusines (like `American`, `Japanese` etc) as well as top restaurents (like `Shake Shack`, `Blue Ribbon Fried Chicken` etc).**

In [ ]:

```
```