INDEX

- Problem Statement and Objectives
- Importing The necessary Libraries
- Loading the dataset
- Basic Exploration
- Order Analysis
- Customer Behaviour
- Restaurant Performance
- Demand Patterns
- Operational Efficiency
- Customer Insights

LIST OF TABLE

LIST OF FIGURES

PROBLEM STATEMENT AND 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.

```
import os
```

```
os.getcwd()
```

IMPORTING THE NECESSARY LIBRARIES

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

LOADING THE DATASET

```
data=pd.read_csv("C:\\Users\\suhan\\OneDrive\\Desktop\\All abt 4sem\\Word files\\bootcamp_project\\2-foodhub_order_New.csv")
```

```
data
```

	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	25.0	20
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25.0	?
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23.0	28
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25.0	15
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25.0	24
...	...	...	...	...	...	...	...	...	...
1893	1476701	292602	Chipotle Mexican Grill $1.99 Delivery	Mexican	22.31	Weekend	5	31.0	17
1894	1477421	397537	The Smile	American	12.18	Weekend	5	31.0	19
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	Not given	31.0	24
1896	1477513	64151	Jack's Wife Freda	Mediterranean	12.18	Weekday	5	23.0	31
1897	1478056	120353	Blue Ribbon Sushi	Japanese	19.45	Weekend	Not given	28.0	24

1898 rows × 9 columns

Displaying all the Dataset.

BASIC EXPLORATION

1. DISPLAYING THE TOP 5 ROWS

```
data.head()
```

	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	25.0	20
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25.0	?
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23.0	28
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25.0	15
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25.0	24

In head section of the dataset, Rating of the first two order_id are writtwn as "not given" which is a string but it should be in integer format and the delivery time of second order is missing and written as "?" in the data set.

2. DISPLAYING THE LAST 5 ROWS

```
data.tail()
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
1893	1476701	292602	Chipotle Mexican Grill $1.99 Delivery	Mexican	22.31	Weekend	5	31.0	17
1894	1477421	397537	The Smile	American	12.18	Weekend	5	31.0	19
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	Not given	31.0	24
1896	1477513	64151	Jack's Wife Freda	Mediterranean	12.18	Weekday	5	23.0	31
1897	1478056	120353	Blue Ribbon Sushi	Japanese	19.45	Weekend	Not given	28.0	24

In tail section of dataset, the last 5 rows are displayed and in which the ratings of two restaurant is written as "Not Given".

INFORMATION ABOUT THE DATA IN BRIEF

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   order_id               1898 non-null   int64  
 1   customer_id            1898 non-null   int64  
 2   restaurant_name        1898 non-null   object 
 3   cuisine_type           1895 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  1896 non-null   float64
 8   delivery_time          1898 non-null   object 
dtypes: float64(2), int64(2), object(5)
memory usage: 133.6+ KB
```

Rating and delivery time should be in integer type.

3. CHECKING THE SHAPE OF DATA

```
data.shape
```

```
(1898, 9)
```

There are total 1898 columns and 9 rows in the given dataset.

4. CHECKING THE DATATYPE OF EACH FEATURES

```
data.dtypes
```

```
order_id                int64
customer_id             int64
restaurant_name        object
cuisine_type           object
cost_of_the_order     float64
day_of_the_week        object
rating                 object
food_preparation_time  float64
delivery_time          object
dtype: object
```

In given datatypes, there are total 3 datatypes are present in which two are of interger type(int64) another two are float(float64) and rest 4 are in object type(object) which is also known as string datatype.

OBSERVATION: Rating and delivery time should be in numerical format but it is present in object datatyoe.

5. CHECK THE STATISTICAL SUMMARY

```
data.describe()
```

	order_id	customer_id	cost_of_the_order	food_preparation_time
count	1.898000e+03	1898.000000	1898.000000	1896.000000
mean	1.477496e+06	171168.478398	80.722007	27.371835
std	5.480497e+02	113698.139743	2798.141333	4.634211
min	1.476547e+06	1311.000000	0.000000	20.000000
25%	1.477021e+06	77787.750000	12.080000	23.000000
50%	1.477496e+06	128600.000000	14.160000	27.000000
75%	1.477970e+06	270525.000000	22.310000	31.000000
max	1.478444e+06	405334.000000	121920.000000	35.000000

"describe()" function shows us the statistical summary of the data.

In the given data set, according to "describe()" function, the minimum cost of the corder is 0.

6. CHECK THE NULL VALUES

```
data.isnull().sum().sum()
```

This shows the total sum of the null values including all the columns and rows of the table which 5 in numbers.

```
data.isnull().sum()
```

```
order_id                0
customer_id             0
restaurant_name         0
cuisine_type            3
cost_of_the_order       0
day_of_the_week         0
rating                  0
food_preparation_time   2
delivery_time           0
dtype: int64
```

There are total 5 NULL values(column wise) present in the data, in which 3 are present in Cuisine_type and another 2 in Food_Preparation_Time.

```
data[data.isnull().sum(axis=1)>0]
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
11	1478437	221206	Empanada Mama (closed)	NaN	8.10	Weekend	5	23.0	22
51	1477883	91817	Blue Ribbon Fried Chicken	NaN	29.39	Weekend	Not given	27.0	28
95	1477027	164016	Blue Ribbon Fried Chicken	NaN	16.39	Weekend	Not given	27.0	22
140	1477376	370372	Blue Ribbon Fried Chicken	American	11.59	Weekday	Not given	NaN	24
188	1477872	300670	Shake Shack	American	13.39	Weekend	Not given	NaN	22

To fetch all the rowshaving atleast one Null Value

```
data.isnull().sum()/len(data)*100
```

```
order_id                0.000000
customer_id             0.000000
restaurant_name         0.000000
cuisine_type            0.158061
cost_of_the_order       0.000000
day_of_the_week         0.000000
rating                  0.000000
food_preparation_time   0.105374
delivery_time           0.000000
dtype: float64
```

Check for the percentage wise missing values in columns.

7. CHECK THE DUPLICATE VALUES

```
data.duplicated().sum()
```

Duplicated data is absent in the given dataset.

8. CHECK THE ANOMALIES AND WRONG ENTRIES

Let's begin with finding the unique values present in each rows of dataset, and then check for tjose entries which is wrong entered and non usable.

Then convert that specific values to nan for futher processing and finding outliers.

```
data['food_preparation_time'].unique()
```

```
array([25., 23., 20., 28., 33., 21., 29., 34., 24., 30., 35., 32., 31.,
       27., 22., 26., nan])
```

It shows how many unique values are present in the column of food_preparation_time, in which we can see that there is a unique value present in it named as "nan".

```
data['rating'].unique()
```

```
array(['Not given', '5', '3', '4'], dtype=object)
```

Showing the Unique values are present in the column of rating, in which we can see that there is a unique value present in it named as "Not Given" which is differnt from other integer type values.

```
data[data['rating']=='Not given']
```

	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	25.0	20
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25.0	?
6	1477894	157711	The Meatball Shop	Italian	6.07	Weekend	Not given	28.0	21
10	1477895	143926	Big Wong Restaurant □_¤¾Ñ¼	Chinese	5.92	Weekday	Not given	34.0	28
14	1478198	62667	Lucky's Famous Burgers	American	12.13	Weekday	Not given	23.0	30
...	...	...	...	...	...	...	...	...	...
1887	1476873	237616	Shake Shack	American	5.82	Weekend	Not given	26.0	30
1891	1476981	138586	Shake Shack	American	5.82	Weekend	Not given	22.0	28
1892	1477473	97838	Han Dynasty	Chinese	29.15	Weekend	Not given	29.0	21
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	Weekday	Not given	31.0	24
1897	1478056	120353	Blue Ribbon Sushi	Japanese	19.45	Weekend	Not given	28.0	24

736 rows × 9 columns

Printing those values whose rating is "Not Given" in the dataset. There is total 736 rows and in which data is present as "Not Given".

```
data['delivery_time'].unique()
```

```
array(['20', '?', '28', '15', '24', '21', '30', '26', '22', '17', '23',
       '25', '16', '29', '27', '18', '31', '32', '19', '33'], dtype=object)
```

The only unique value present in the delivery_time column is "?" which is a wrong entry.

```
data[data['delivery_time']=='?']
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25.0	?
180	1476808	84700	Pepe Giallo	Italian	14.60	Weekday	3	32.0	?

There are total 2 entries in which delivery_time is wrong entered.

```
data['delivery_time'].replace('?',np.nan,inplace=True)
data['delivery_time'].unique()
```

```
array(['20', nan, '28', '15', '24', '21', '30', '26', '22', '17', '23',
       '25', '16', '29', '27', '18', '31', '32', '19', '33'], dtype=object)
```

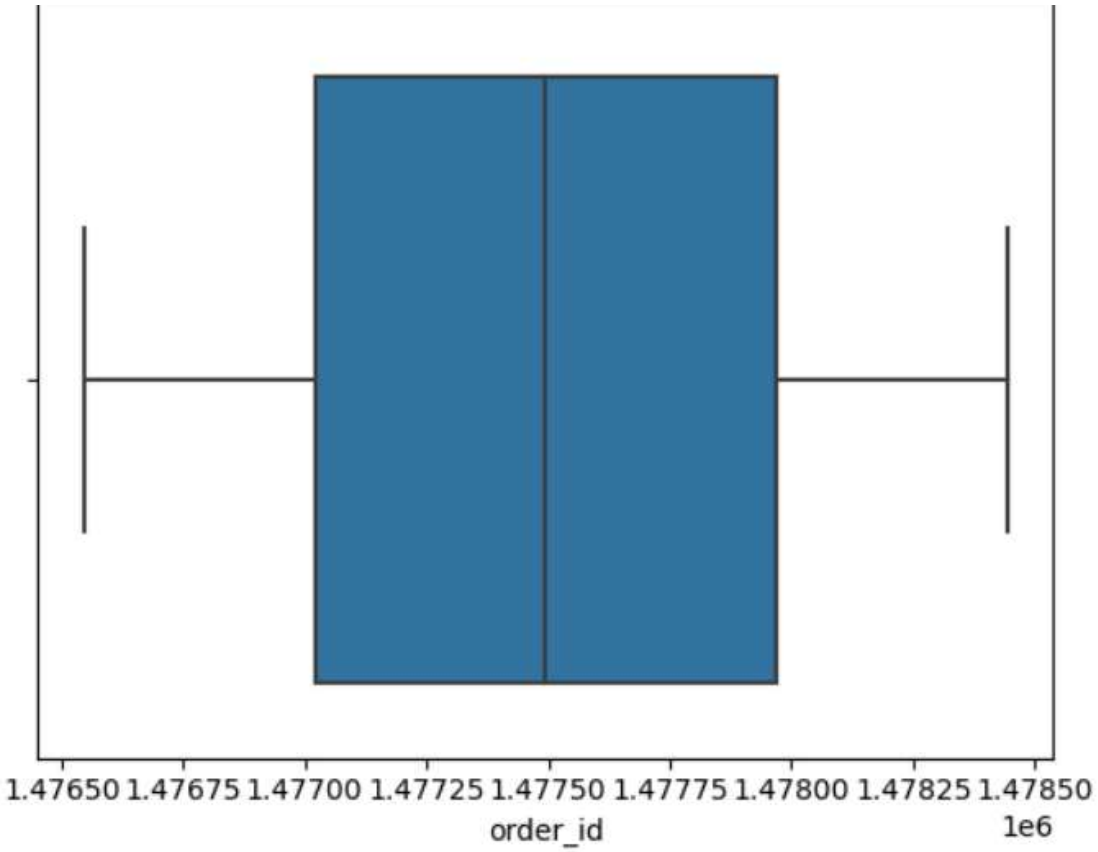
```
data['rating']=data['rating'].replace('Not given',np.nan)
data['rating'].unique()
```

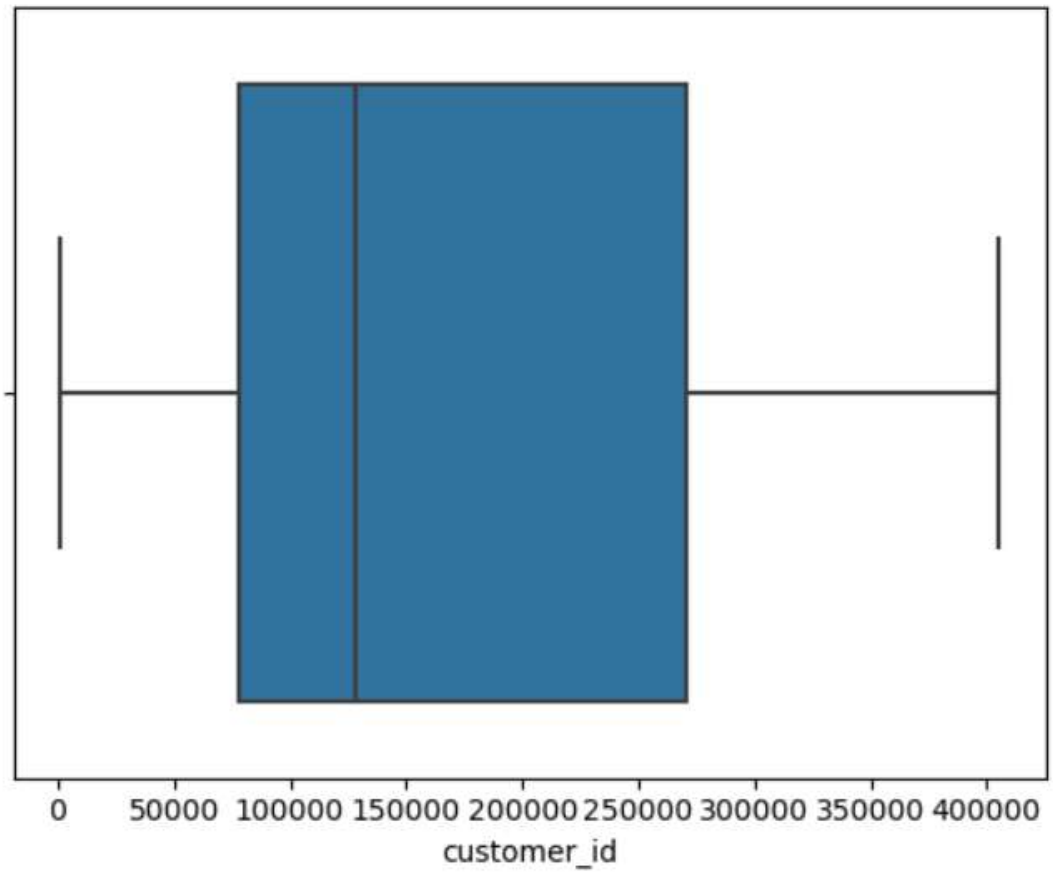
```
array([nan, '5', '3', '4'], dtype=object)
```

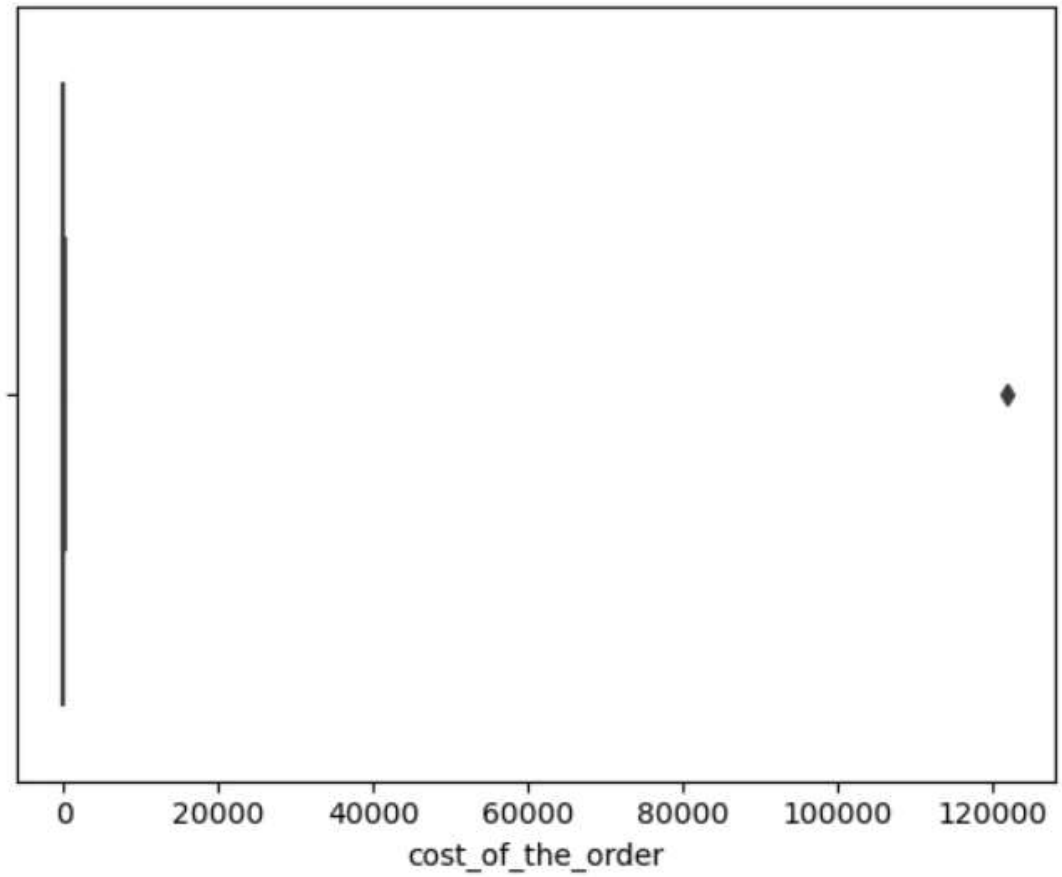
Replacing the wrong entries into "nan" and changing it into main dataset for further treatment.

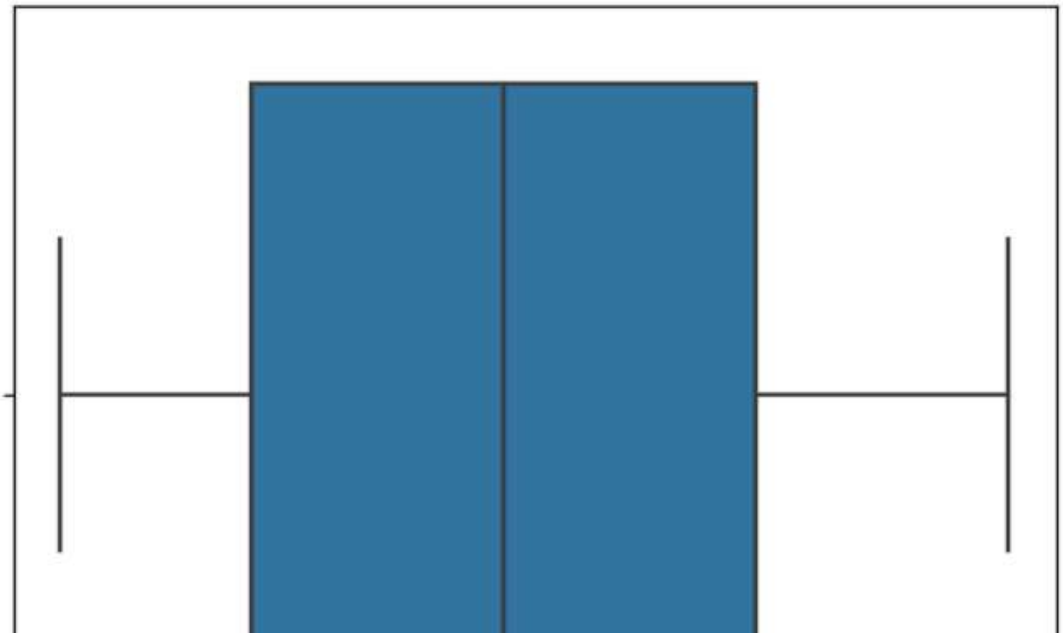
9. CHECK THE OUTLIERS AND THE AUTHENTICITY

```
for i in data.columns:
    if data[i].dtype != 'object':
        sns.boxplot(data=data,x=i)
        plt.show()
```


1.47650 1.47675 1.47700 1.47725 1.47750 1.47775 1.47800 1.47825 1.47850
order_id
1e6


0 50000 100000 150000 200000 250000 300000 350000 400000
customer_id





Checking the outliers by creating boxplot of columns for ouliers treatment.

Observation:- According to graph, only one outlier is present in the above graph which is in cost_of_the_order as clearly shown in the graph.

10. DO THE NECESSARY DATA CLEANING STEPS LIKE DROPPING DUPLICATES, UNNECESSARY COLUMNS, NULL VALUE IMPUTATION, OUTLIERS TREATMENT ETC.

def remove_outliers(col): sorted(col) Q1,Q3=col.quantile([0.25,0.75]) IQR=Q3-Q1

Defining a function named "remove_outliers" in which we use quantile to set up the lower and upper bound to change the lower and upper outliers entries to lower and upper bound respectively.

```
median1=data['cost_of_the_order'].median()
median2=data['customer_id'].median()
median3=data['order_id'].median()
median4=data['food_preparation_time'].median()
median6=data['delivery_time'].median()

data['cost_of_the_order'].replace(np.nan,median1,inplace=True)
data['customer_id'].replace(np.nan,median2,inplace=True)
data['order_id'].replace(np.nan,median3,inplace=True)
data['food_preparation_time'].replace(np.nan,median4,inplace=True)
data['delivery_time'].replace(np.nan,median6,inplace=True)
```

Finding the median of all the numeric column of the data which is cost_of_the_order, customer_id, order_id, food_preparation_time, delivery_time and replacing the nan values with median and manipulating the main dataset with it.

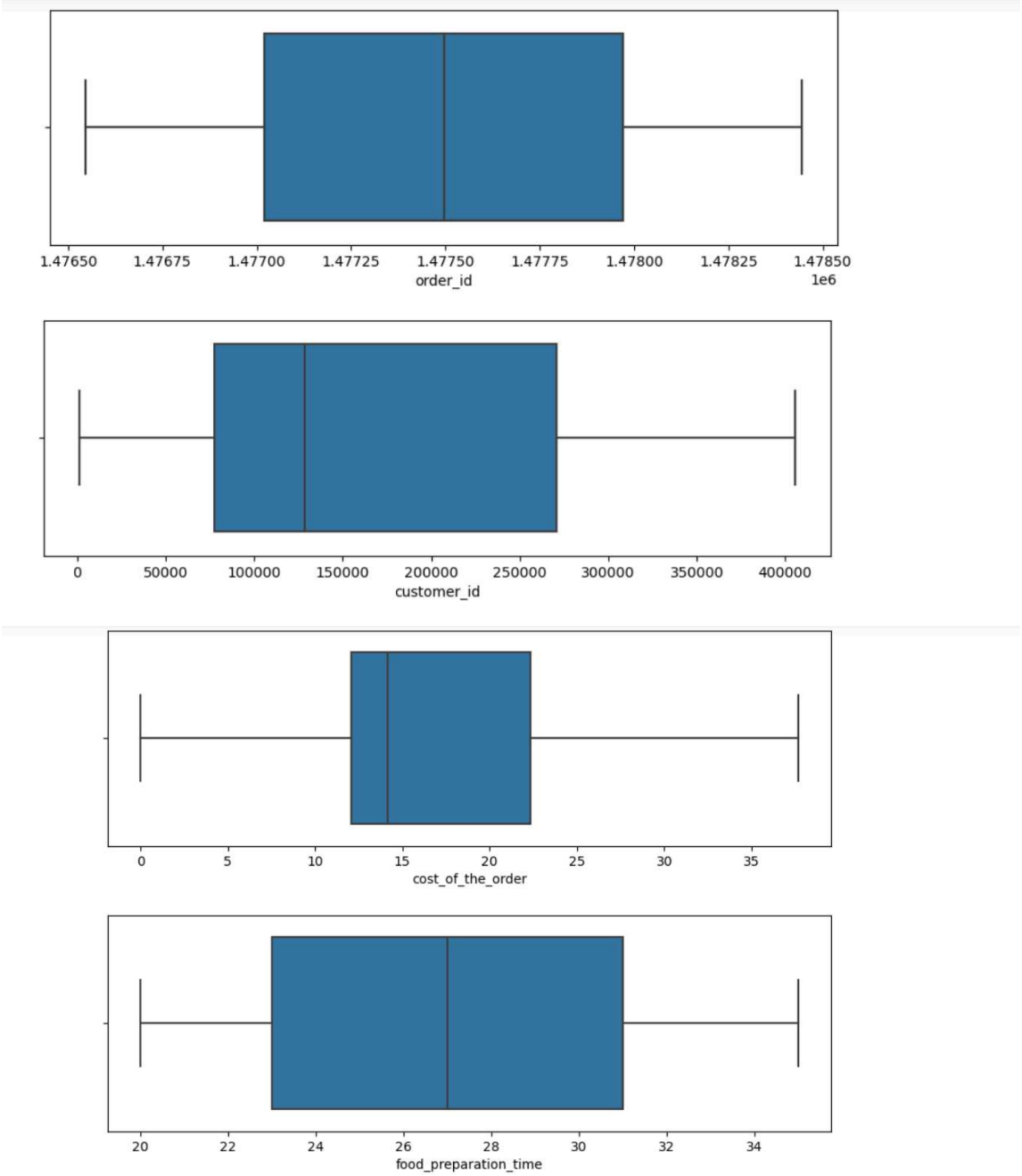
```
mode1=data['restaurant_name'].mode().values[0]
mode2=data['cuisine_type'].mode().values[0]
mode3=data['day_of_the_week'].mode().values[0]

data['restaurant_name']=data['restaurant_name'].replace(np.nan,mode1)
data['cuisine_type']=data['cuisine_type'].replace(np.nan,mode2)
data['day_of_the_week']=data['day_of_the_week'].replace(np.nan,mode3)
```

Finding the mode of all the categorical column of the data which is restaurant_name, cuisine_type, day_of_the_week and replace the nan values of categorical data with mode.

```
ll,ul=remove_outliers(data['cost_of_the_order'])
data['cost_of_the_order']=np.where(data['cost_of_the_order']>ul,ul,data['
cost_of_the_order'])
data['cost_of_the_order']=np.where(data['cost_of_the_order']
<ll,ll,data['cost_of_the_order'])
```

Handling the outliers present in cost_of_the_order with lower and upper limit.

```
for i in
['order_id','customer_id','cost_of_the_order','food_preparation_time']:
    plt.figure(figsize=(10,3))
    sns.boxplot(data=data,x=i)
    plt.show()
```



Plotting the graph using Boxplot to check whether the error is handled.

```
data.isnull().sum()
```

```
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
```

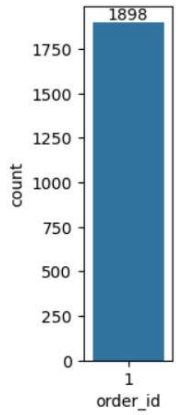
Again checking the null values of data set which shows there is no null values present in the dataset except rating column.

Observation:- Total 736 rating of resturants are not provided in dataset and substituting those values to null and then median can lead to error. So, we will not convert those data into median and work on only those data which is provided.

ORDER ANALYSIS

a. What is the total number of orders in the dataset?

```
plt.figure(figsize=(1,4))
tno=sns.countplot(data=data,x=data['order_id'].value_counts())
for bars in tno.containers:
    tno.bar_label(bars)
```



The total number of orders in the dataset is 1898.

b. What is the average cost of the order?

```
average_cost = data['cost_of_the_order'].mean()
average_cost
```

The average cost of the order is 16.50580874604847.

c. How many unique customers have placed orders?

```
unique_customers = data['customer_id'].nunique()
unique_customers
```

There are total 1200 unique customers who places order online.

d. Which restaurant has recieved the highest number of orders?

```
restaurant_order_counts = data['restaurant_name'].value_counts()
top_restaurant = restaurant_order_counts.idxmax()
top_order_count = restaurant_order_counts.max()
top_restaurant,top_order_count
```

Restaurant named "Shake Shack" has recieved total 219 orders which is highest in number.

CUSTOMER BEHAVIOUR

a. What is the average rating given by customers?

```
data['rating'].unique()
```

```
data['rating']=data['rating'].replace(np.nan,0)
data['rating']=data['rating'].astype('int')
```

```
data.dtypes
```

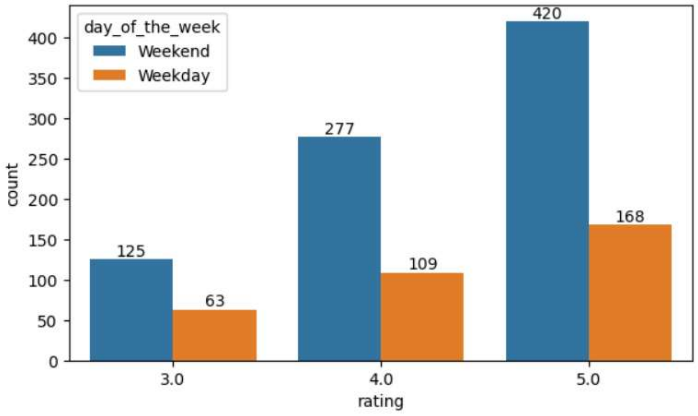
```
data['rating']=data['rating'].replace(0,np.nan)
data['rating'].mean(skipna=True)
```

Again finding the unique values of the rating column and replacing the nan to '0' to convert its datatype to 'int'. Without using the "Not Given" values.

The average rating given by customers is 4.344234079173838.

b. How does the rating vary between weekdays and weekends?

```
plt.figure(figsize=(7,4))
bx = sns.countplot(data=data,x="rating",hue="day_of_the_week")
for bars in bx.containers:
    bx.bar_label(bars)
```



This countplot shows the rating vary between weekdays and weekends.

c. Which Cuisine type is ordered the most?

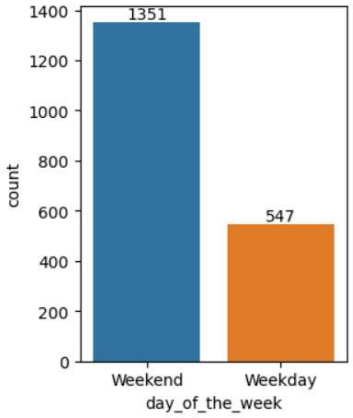
```
cuisine_type=data['cuisine_type'].value_counts()
top_cuisine_type=cuisine_type.idxmax()
most_cuisine_type=cuisine_type.max()
top_cuisine_type,most_cuisine_type
```

"American" Cuisine is ordered the most which is total 585 in numbers.

d. What is the distribution of orders across different days of the week?

```
data['day_of_the_week'].unique()
```

```
plt.figure(figsize=(3,4))
ax=sns.countplot(data=data,x='day_of_the_week')
for bars in ax.containers:
    ax.bar_label(bars)
```



The above countplot shows the distribution of orders across different days of the week.

RESTAURANT PERFORMANCE

a. What is the average food preparation time for each restaurant?

```
data.groupby(['restaurant_name'],as_index=False)
['food_preparation_time'].mean().sort_values(by='food_preparation_time',a
scending=False)
```

	restaurant_name	food_preparation_time
39	Cipriani Le Specialita	35.0
140	Sushi Choshi	35.0
83	Kambi Ramen House	35.0
85	Klong	35.0
148	Taro Sushi	35.0
...	...	...
96	Market Table	21.0
43	Despaİ±a	20.5
3	67 Burger	20.0
58	Frank Restaurant	20.0
68	Haru Gramercy Park	20.0

178 rows × 2 columns

b. Which restaurant has the shortest average food preparation time?

```
data.groupby(['restaurant_name'],as_index=False)
['food_preparation_time'].mean().sort_values(by='food_preparation_time',a
scending=True).head()
```

	restaurant_name	food_preparation_time
68	Haru Gramercy Park	20.0
3	67 Burger	20.0
58	Frank Restaurant	20.0
43	Despaİ±a	20.5
132	Sarabeth's West	21.0

c. How does the average delivery time compare across different restaurants?

```
data['delivery_time']=data['delivery_time'].astype('int')
```

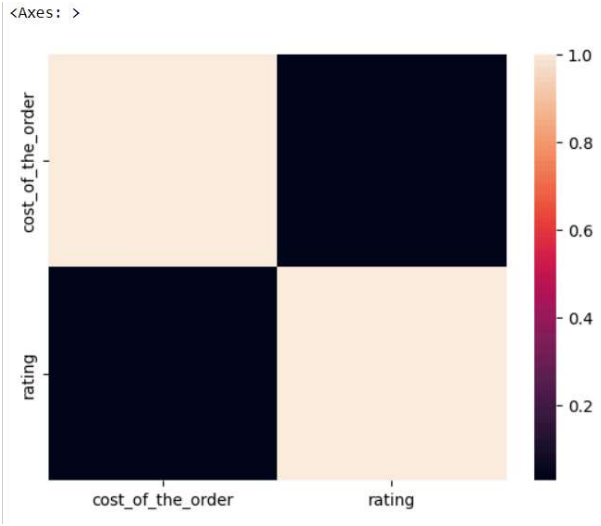
```
data.groupby(['restaurant_name'],as_index=False)
['delivery_time'].mean().sort_values(by='delivery_time',ascending=False)
```

	restaurant_name	delivery_time
132	Sarabeth's West	33.0
148	Taro Sushi	32.0
68	Haru Gramercy Park	32.0
58	Frank Restaurant	31.0
64	Haandi	30.5
...	...	...
61	Galli Restaurant	16.0
152	The MasalaWala	15.0
110	Paul & Jimmy's	15.0
71	Hibino	15.0
60	Gaia Italian Cafe	15.0

178 rows × 2 columns

d. Is there a correlation between the cost of the order and the rating given?

```
data[['cost_of_the_order', 'rating']].corr()
sns.heatmap(data=data[['cost_of_the_order','rating']].corr())
```



The above graph shows the correlation between the cost of the order and the rating given.

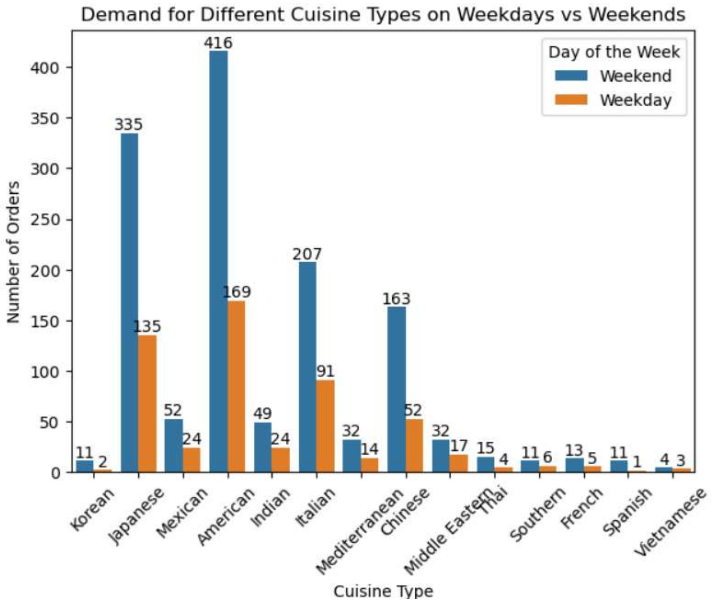
DEMAND PATTERNS

a. How does the demand for different cuisine types vary on weekdays versus weekends?

```
weekday_demand = data[data['day_of_the_week'] == 'Weekday']
['cuisine_type'].value_counts()
weekend_demand = data[data['day_of_the_week'] == 'Weekend']
['cuisine_type'].value_counts()
```

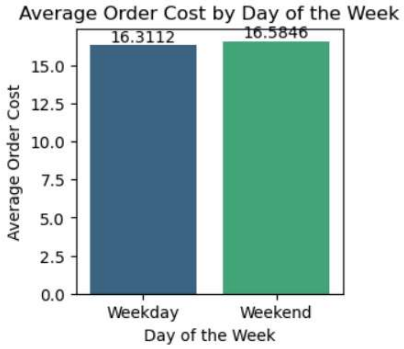
```
plt.figure(figsize=(7, 5))
ax=sns.countplot(data=data, x='cuisine_type', hue='day_of_the_week')
plt.title('Demand for Different Cuisine Types on Weekdays vs Weekends')
plt.xlabel('Cuisine Type')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
```

```
plt.legend(title='Day of the Week')
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



b. Which day of the week has the highest average order cost?

```
average_cost_by_day = data.groupby('day_of_the_week')
['cost_of_the_order'].mean().reset_index()
plt.figure(figsize=(3, 3))
ax= sns.barplot(data=average_cost_by_day, x='day_of_the_week',
y='cost_of_the_order', palette='viridis')
plt.title('Average Order Cost by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Order Cost')
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```

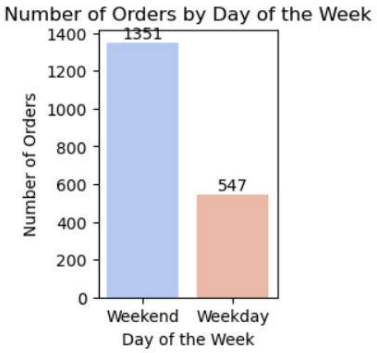


According to plotted graph above, there is a minor difference between weekday and weekend. The average order cost of the weekend is little hgher than weekdays.

c. What is the most common day for orders to be placed?

```
most_common_order_day = data['day_of_the_week'].value_counts().idxmax()
most_common_order_day
```

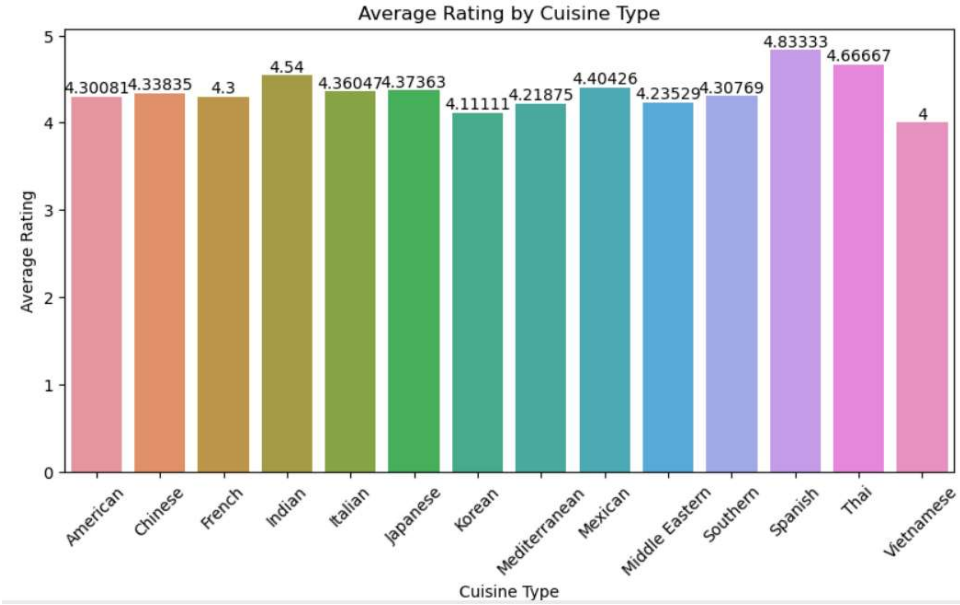
```
order_count_by_day = data['day_of_the_week'].value_counts()
plt.figure(figsize=(2,3))
ax=sns.barplot(x=order_count_by_day.index, y=order_count_by_day.values,
palette='coolwarm')
plt.title('Number of Orders by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



The most common day for orders to be placed is on weekends.

d. How does the average rating vary by ciusine type?

```
average_rating_by_cuisine = data.groupby('cuisine_type')
['rating'].mean().reset_index()
plt.figure(figsize=(10, 5))
ax=sns.barplot(data=average_rating_by_cuisine, x='cuisine_type',
y='rating')
plt.title('Average Rating by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



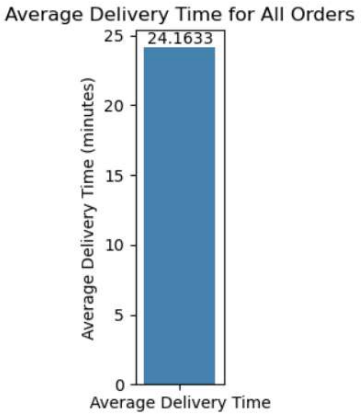
The above barplot shows the Average rating by Cuisine Type in which Spanish Cuisine have the highest average rating.

OPERATIONAL EFFICIENCY

a. What is the average of delivery time of all the orders?

```
adt=data['delivery_time'].mean()
```

```
plt.figure(figsize=(1, 4))
ax=sns.barplot(x=['Average Delivery Time'], y=[adt], palette='Blues_d')
plt.title('Average Delivery Time for All Orders')
plt.ylabel('Average Delivery Time (minutes)')
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



The average of delivery time of all the orders is 24.1633.

b. Which restaurant has the longest average delivery time?

```
average_delivery_time_by_restaurant = data.groupby('restaurant_name')
['delivery_time'].mean().reset_index()
longest_delivery_time_restaurant =
average_delivery_time_by_restaurant.loc[average_delivery_time_by_restaura
nt['delivery_time'].idxmax()]
longest_delivery_time_restaurant
```

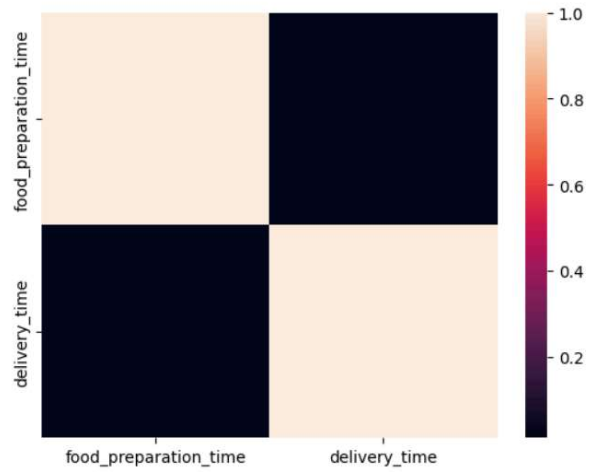
```
restaurant_name    Sarabeth's West
delivery_time                33.0
Name: 132, dtype: object
```

"Sarabeth's West" restaurant has the longest average delivery time which is 33.0.

c. Is there a relationship between food preparation time and delivery time?

```
data[['food_preparation_time','delivery_time']].corr()
sns.heatmap(data=data[['food_preparation_time','delivery_time']].corr())
```

```
<Axes: >
```



The above heatmap shows the relationship between food preparation time and delivery time

d. How does the delivery time impact customer ratings?

```
correlation_delivery_rating = data[['delivery_time',
'rating']].corr().iloc[0, 1]
correlation_delivery_rating

plt.figure(figsize=(13, 6))
ax=sns.barplot(data=data, x='delivery_time', y='rating')
plt.title('Impact of Delivery Time on Customer Ratings')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Customer Rating')
for bars in ax.containers:
    ax.bar_label(bars)
```

```
plt.show()
```

The delivery time impact customer ratings shown by given barplot.

OR

```
sns.heatmap(data=data[['delivery_time','rating']].corr())
```

```
<Axes: >
```

It can also be shown using the heatmap.

CUSTOMER INSIGHTS

a. What is the repeat order rate(number of customers who have placed more than one order)?

```
data2=data[data['customer_id'].duplicated()]
data3=data2['customer_id'].value_counts()
```

```
data3[data3.values>1].head(10)
```

```
52832     12
47440      9
83287      8
250494     7
259341     6
65009      6
276192     6
82041      6
60052      5
97991      5
Name: customer_id, dtype: int64
```

```
data3[data3.values>1].shape
```

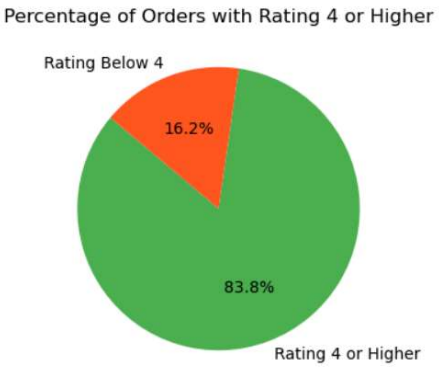
```
(149,)
```

The repeat order rate(number of customers who have placed more than one order) is 149.

b. What is the percentage of orders recieve a rating of 4 or higher?

```
(data[data['rating'] >= 4].shape[0] / data.shape[0]) * 100
```

```
rating_counts = [data[data['rating'] >= 4].shape[0], data[data['rating']
< 4].shape[0]]
rating_labels = ['Rating 4 or Higher', 'Rating Below 4']
plt.figure(figsize=(4, 4))
plt.pie(rating_counts, labels=rating_labels, autopct='%1.1f%%', colors=
['#4CAF50', '#FF5722'], startangle=140)
plt.title('Percentage of Orders with Rating 4 or Higher')
plt.show()
```



The percentage of orders recieve a rating of 4 or higher is 51.31717597471022. The above pie chart shows the rating 4 or higher.