About This Project :

In this project, we aim to analyze Zomato restaurant data to identify key factors that contribute to the success of restaurants, as measured by their ratings. By exploring various features such as location, cuisine, pricing, and service offerings, we aim to provide insights that can help restaurant owners and Zomato users make informed decisions

Project Flow: -

1. **Data collection and Data loading**
2. **Data Preprocessing - Handling missing values, Handling outlier, duplicates, Handling categorical(lastly)**
3. **EDA - Exploratory Data Analysis - Formulate 10-15 questions - based on given problem statement**
4. **Observation - answer to these 10-15 questions**
5. **Recommendations - sumaarization based on acquired answers**
6. **Conclusion - 4-5 point**

1. Data collection and Data loading

In [ ]:

```
## importing libraries
import os ## optional library - used to import paths for different files
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

## warning libarary
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
## load the dataset
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/EDA DATASETS/Indian-Resturants.csv")
```

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [ ]:

```
## showing the data
df.head(2)
```

Out[ ]:

	res_id	name	establishment	url	address	city	city_id	locality	latitude	longitude	...	price_range	currency	highlights	aggre
0	3400299	Bikanervala	['Quick Bites']	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	Agra	34	Khandari	27.211450	78.002381	...	2	Rs.	['Lunch', 'Takeaway Available', 'Credit Card',...	
1	3400005	Mama Chicken Mama Franky House	['Quick Bites']	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra	Agra	34	Agra Cantt	27.160569	78.011583	...	2	Rs.	['Delivery', 'No Alcohol Available', 'Dinner',...	

2 rows × 26 columns

Data Overview:

Explore the basic characteristics of the dataset, including dimensions, data types, and missing values

In [ ]:

```
rows=df.shape[0]
columns=df.shape[1]
print(f"This dataset containes {rows} and {columns}")
```

```
This dataset containes 211944 and 26
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211944 entries, 0 to 211943
Data columns (total 26 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   res_id                211944 non-null  int64
```

```
 0   res_id                211944 non-null  int64
 1   name                  211944 non-null  object
 2   establishment         211944 non-null  object
 3   url                   211944 non-null  object
 4   address               211810 non-null  object
 5   city                  211944 non-null  object
 6   city_id               211944 non-null  int64
 7   locality              211944 non-null  object
 8   latitude              211944 non-null  float64
 9   longitude             211944 non-null  float64
 10  zipcode               48757 non-null   object
 11  country_id            211944 non-null  int64
 12  locality_verbose      211944 non-null  object
 13  cuisines              210553 non-null  object
 14  timings               208070 non-null  object
 15  average_cost_for_two  211944 non-null  int64
 16  price_range           211944 non-null  int64
 17  currency              211944 non-null  object
 18  highlights            211944 non-null  object
 19  aggregate_rating      211944 non-null  float64
 20  rating_text           211944 non-null  object
 21  votes                 211944 non-null  int64
 22  photo_count           211944 non-null  int64
 23  opentable_support     211896 non-null  float64
 24  delivery              211944 non-null  int64
 25  takeaway              211944 non-null  int64
dtypes: float64(4), int64(9), object(13)
memory usage: 42.0+ MB
```

2. Data Preprocessing

In [ ]:

```
## missing values
df.isnull().sum()
```

Out[ ]:

	0
res_id	0
name	0
establishment	0
url	0
address	134
city	0
city_id	0
locality	0
latitude	0
longitude	0
zipcode	163187
country_id	0
locality_verbose	0
cuisines	1391
timings	3874
average_cost_for_two	0
price_range	0
currency	0
highlights	0
aggregate_rating	0
rating_text	0
votes	0
photo_count	0
opentable_support	48
delivery	0
takeaway	0

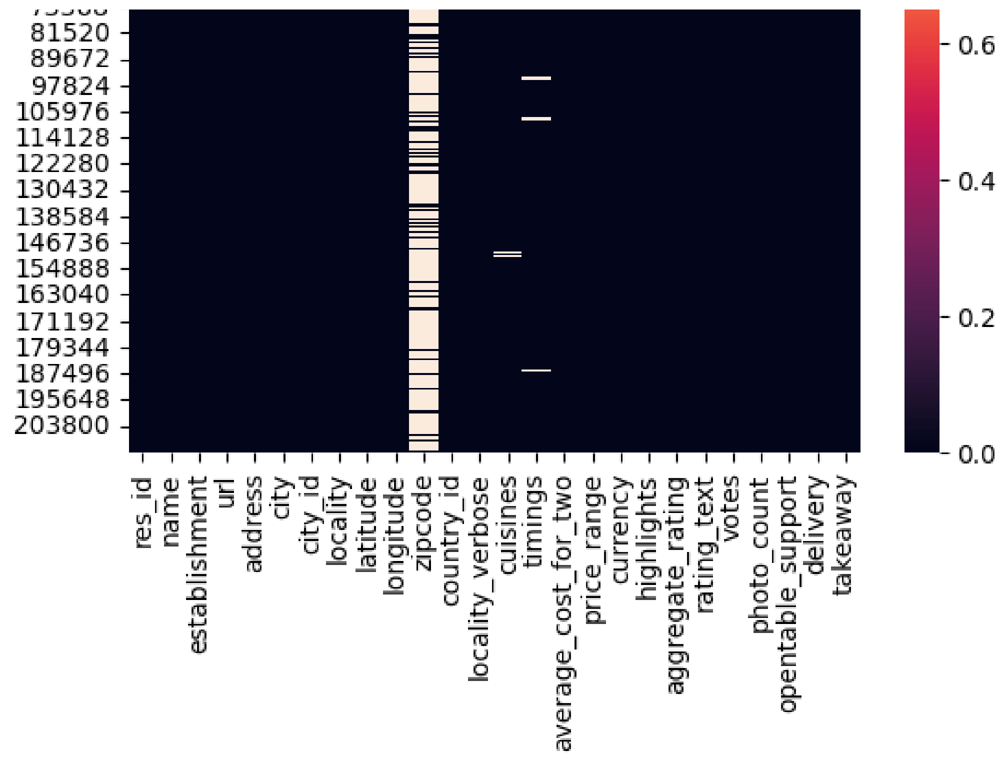
dtype: int64

In [ ]:

```
sns.heatmap(df.isnull())
```

Out[ ]:

<Axes: >



Handling missing values - we will check the percentage of missing values:-

- **if missing values are greater than 25% we will drop the column (default you are domain expert)**
- **if less than 25% values are missing then we will cap the values using statistical method**

In [ ]:

```
## percentage of missing values in each column
(df.isnull().sum()/len(df))*100
```

Out[ ]:

	0
res_id	0.000000
name	0.000000
establishment	0.000000
url	0.000000
address	0.063224
city	0.000000
city_id	0.000000
locality	0.000000
latitude	0.000000
longitude	0.000000
zipcode	76.995338
country_id	0.000000
locality_verbose	0.000000
cuisines	0.656305
timings	1.827841
average_cost_for_two	0.000000
price_range	0.000000
currency	0.000000
highlights	0.000000
aggregate_rating	0.000000
rating_text	0.000000
votes	0.000000
photo_count	0.000000
opentable_support	0.022647
delivery	0.000000
takeaway	0.000000

dtype: float64

In [ ]:

```
## first we will drop zipcode column as it is missing more than 76% of values
df.drop("zipcode", axis = 1, inplace = True)
```

In [ ]:

```
## Required capping column names = address,cuisines, timings, opentable_support
```

In [ ]:

```
df.dtypes
```

Out[ ]:

	0
res_id	int64
name	object
establishment	object
url	object
address	object
city	object
city_id	int64
locality	object
latitude	float64
longitude	float64
country_id	int64
locality_verbose	object
cuisines	object
timings	object
average_cost_for_two	int64
price_range	int64
currency	object
highlights	object
aggregate_rating	float64
rating_text	object
votes	int64
photo_count	int64
opentable_support	float64
delivery	int64
takeaway	int64

dtype: object

In [ ]:

```
##for objects, we will use this to cross check the frequency of varriables
df['timings'].value_counts()
#It counts how many times each unique value appears in the column.
```

Out[ ]:

timings	count
11 AM to 11 PM	26605
10 AM to 10 PM	5419
11 AM to 10 PM	4933
11 AM to 11 PM (Mon-Sun)	4063
10 AM to 11 PM	3949
...	...
10:30 AM to 9 PM (Mon-Sat), Sun Closed	1
6pm – 11:30pm (Mon),6pm – 11pm (Tue-Sun)	1
12 Noon to 11:45 PM, 12 Midnight to 12:30 AM	1
1 PM to 12:30 AM (Mon-Sun)	1
10am – 11pm (Mon-Wed),10:30am – 11pm (Thu-Sun)	1

7740 rows × 1 columns

dtype: int64

In [ ]:

```
## Handling missing in categorical variable
list_of_cols_cat = ["address","cuisines", "timings"]
for i in list_of_cols_cat:
    df[i] = df[i].fillna(df[i].mode()[0])
```

In [ ]:

```
df["opentable_support" ].value_counts()
```

Out[ ]:

opentable_support	count
0.0	211896

dtype: int64

In [ ]:

```
## opentable_support column has only 0 as number and it will not be helpful for analysis i will drop the column
df.drop("opentable_support", axis = 1, inplace = True)
```

In [ ]:

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

Out[ ]:

	0
res_id	0
name	0
establishment	0
url	0
address	0
city	0
city_id	0
locality	0
latitude	0
longitude	0
country_id	0
locality_verbose	0
cuisines	0
timings	0
average_cost_for_two	0
price_range	0
currency	0
highlights	0
aggregate_rating	0
rating_text	0
votes	0
photo_count	0
delivery	0
takeaway	0

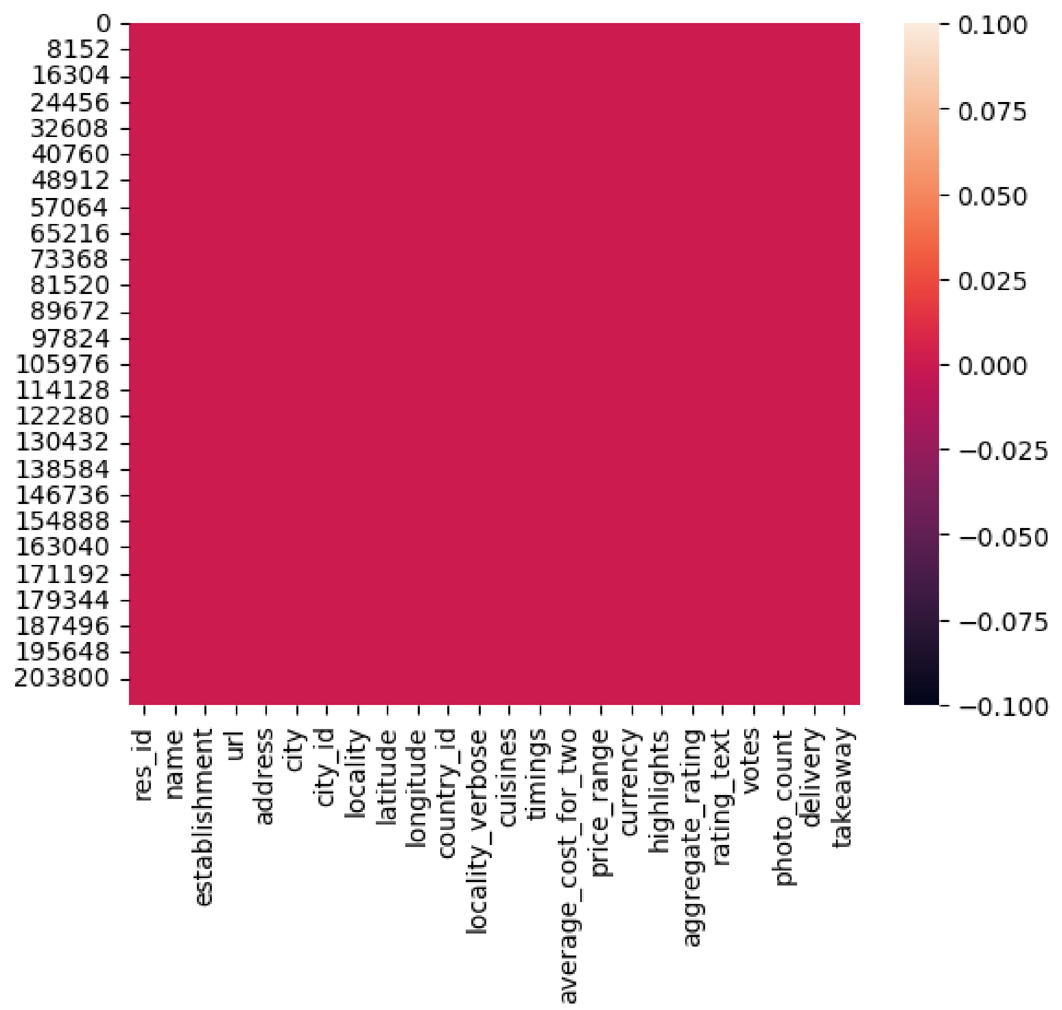
dtype: int64

In [ ]:

```
sns.heatmap(df.isnull())
```

Out[ ]:

<Axes: >



In [ ]:

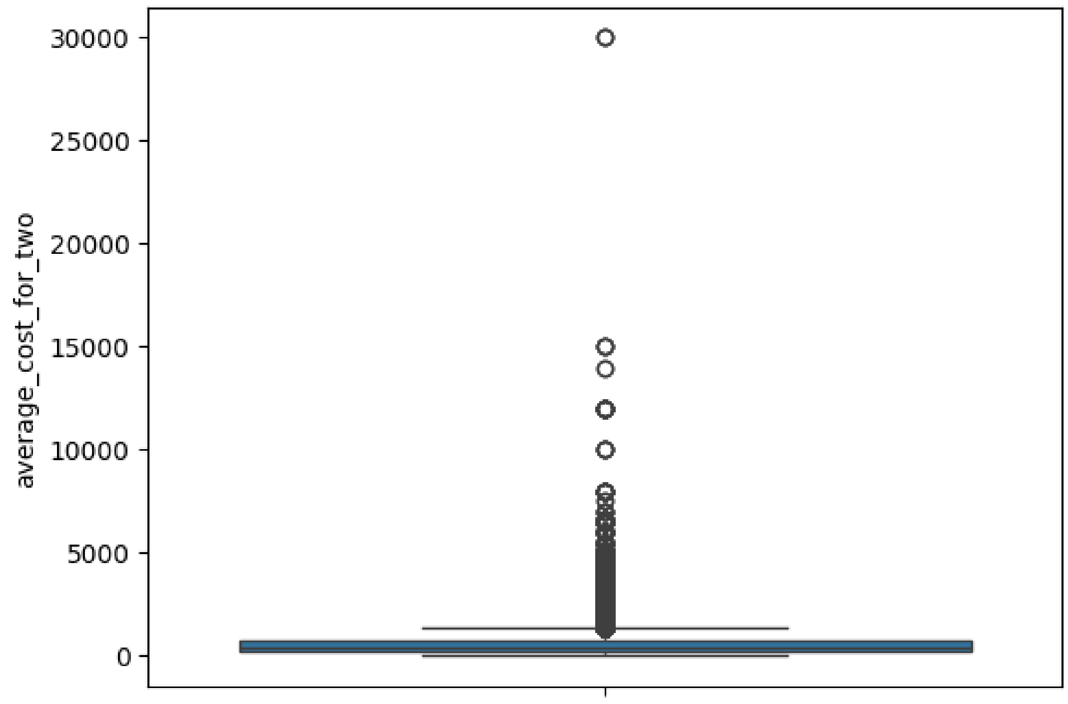
```
df.info()
```

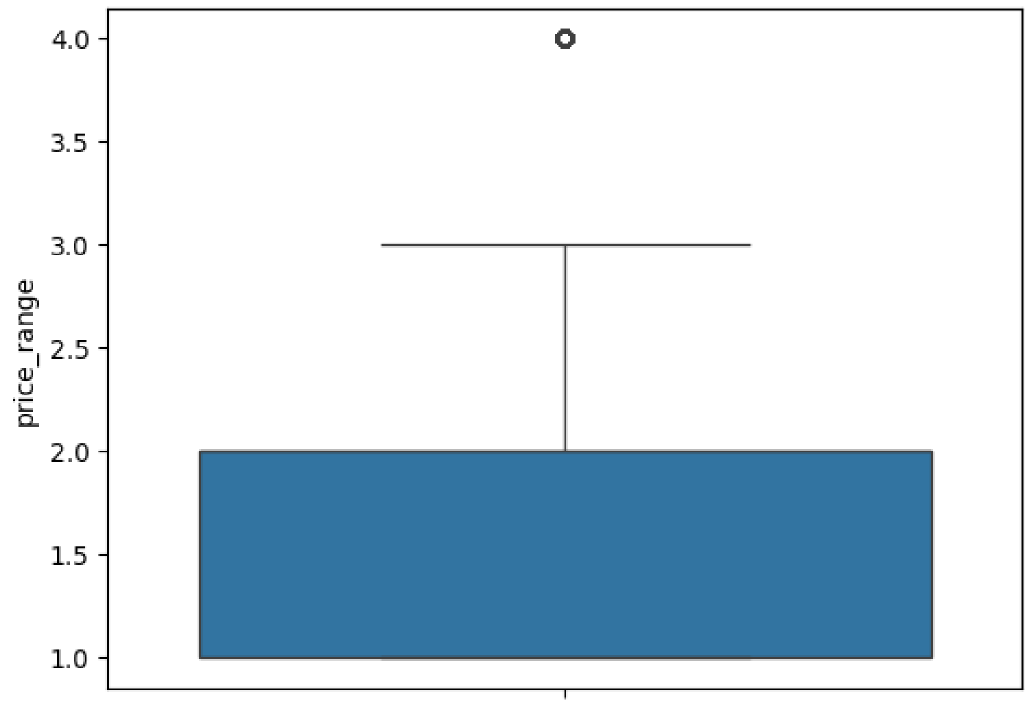
<class 'pandas.core.frame.DataFrame'>

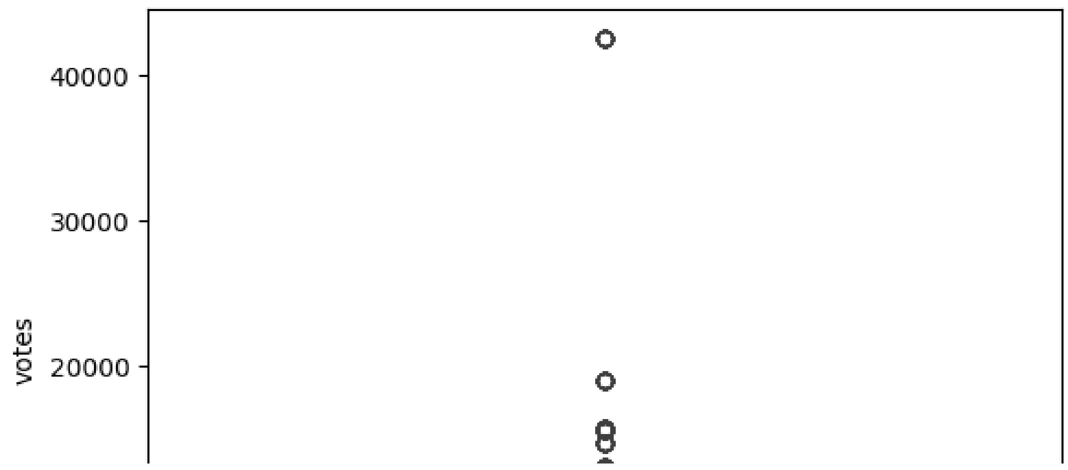
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211944 entries, 0 to 211943
Data columns (total 24 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   res_id                211944 non-null  int64  
 1   name                  211944 non-null  object 
 2   establishment         211944 non-null  object 
 3   url                   211944 non-null  object 
 4   address               211944 non-null  object 
 5   city                  211944 non-null  object 
 6   city_id               211944 non-null  int64  
 7   locality              211944 non-null  object 
 8   latitude              211944 non-null  float64
 9   longitude             211944 non-null  float64
 10  country_id            211944 non-null  int64  
 11  locality_verbose      211944 non-null  object 
 12  cuisines              211944 non-null  object 
 13  timings               211944 non-null  object 
 14  average_cost_for_two  211944 non-null  int64  
 15  price_range           211944 non-null  int64  
 16  currency              211944 non-null  object 
 17  highlights            211944 non-null  object 
 18  aggregate_rating      211944 non-null  float64
 19  rating_text           211944 non-null  object 
 20  votes                 211944 non-null  int64  
 21  photo_count           211944 non-null  int64  
 22  delivery              211944 non-null  int64  
 23  takeaway              211944 non-null  int64  
dtypes: float64(3), int64(9), object(12)
memory usage: 38.8+ MB
```

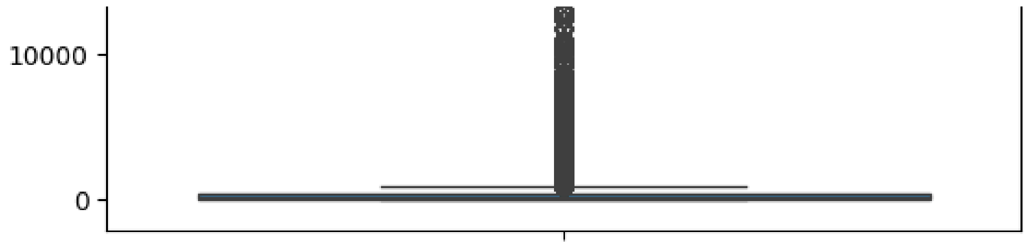
In [ ]:

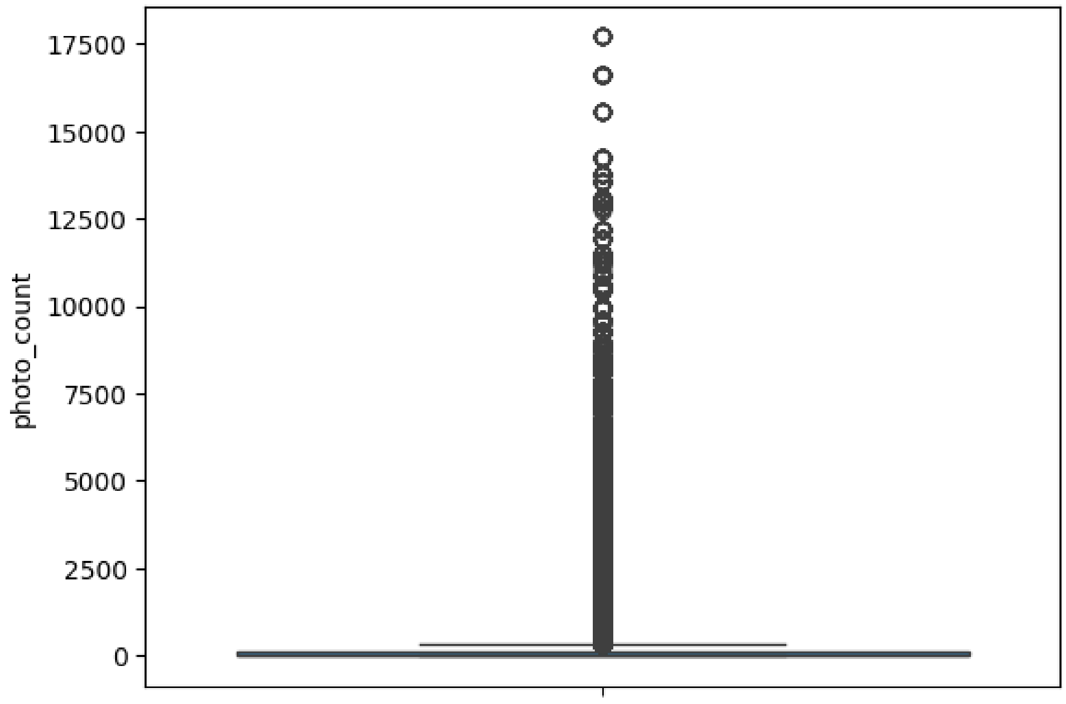
```python
num_col1 = ["average_cost_for_two", "price_range", "votes", "photo_count"]
for i in num_col1:
    sns.boxplot(df[i])
    plt.show()
```











Handling Outliers

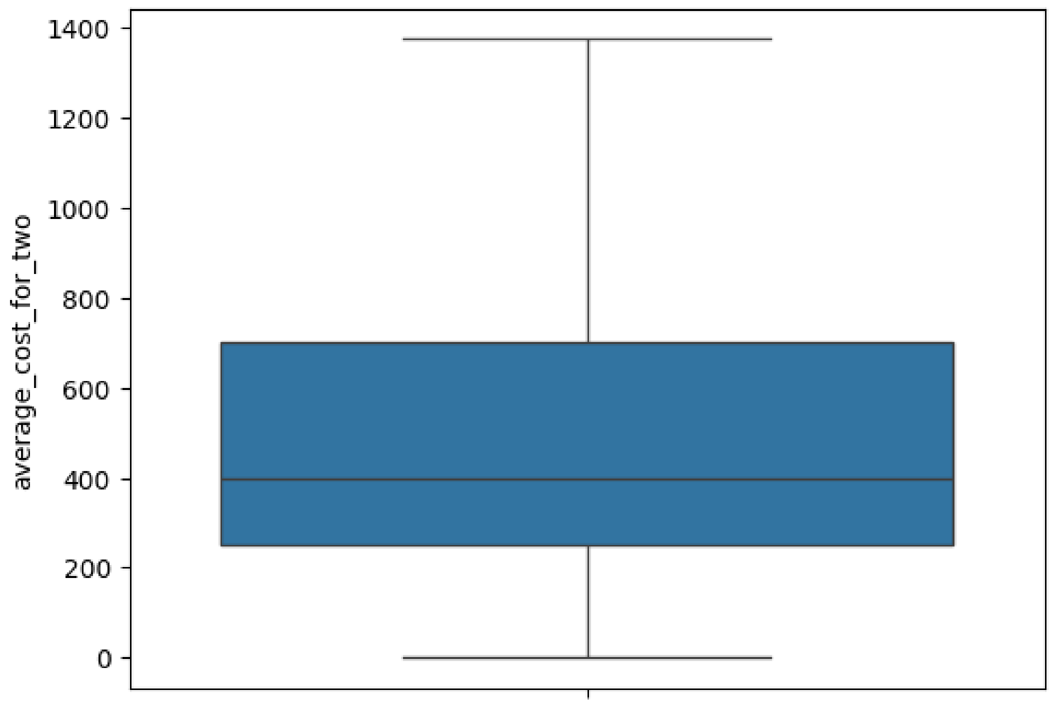
In [ ]:

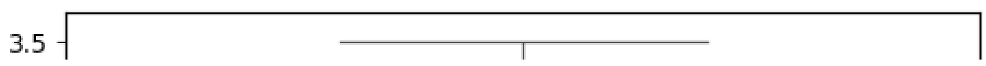
```
## IQR
num_col1 = ["average_cost_for_two", "price_range", "votes", "photo_count"]
for i in num_col1:
    q1 = df[i].quantile(0.25)
    q3 = df[i].quantile(0.75)
    iqr = q3 - q1
    print(f"Q1:{q1}")
    print(f"Q3:{q3}")
    print(f"IQR:{iqr}")

## formulate UL and lower LL
    UL = q3+1.5*iqr
    LL = q1-1.5*iqr
    df[i] = np.where(df[i]>UL,UL,
                    np.where(df[i]<LL,LL,
                            df[i]))
```

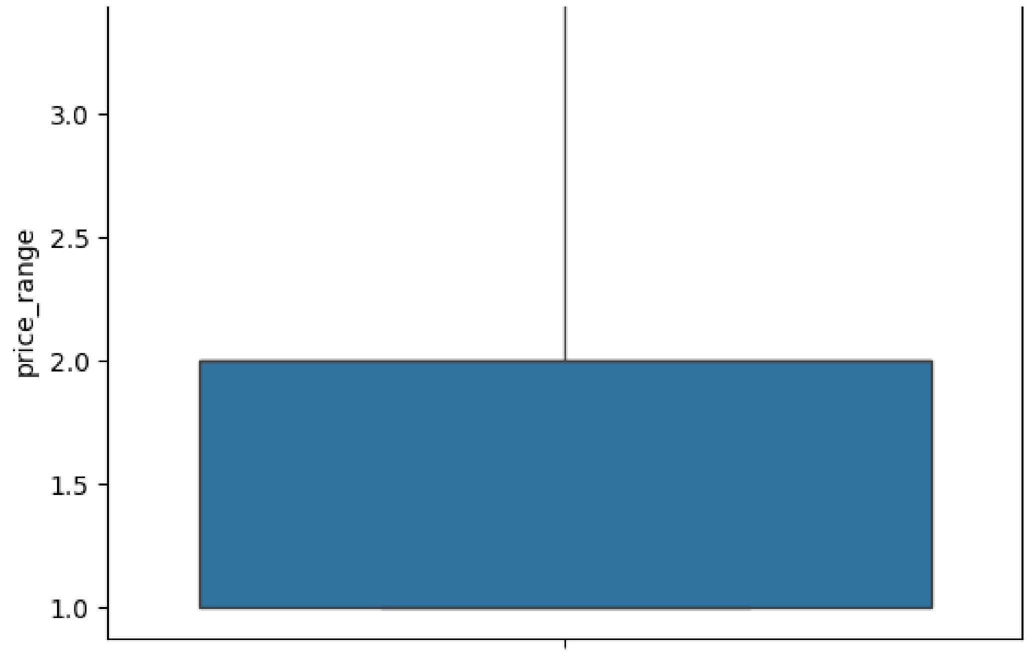
```
Q1:250.0
Q3:700.0
IQR:450.0
Q1:1.0
Q3:2.0
IQR:1.0
Q1:16.0
Q3:362.0
IQR:346.0
Q1:3.0
Q3:128.0
IQR:125.0
```

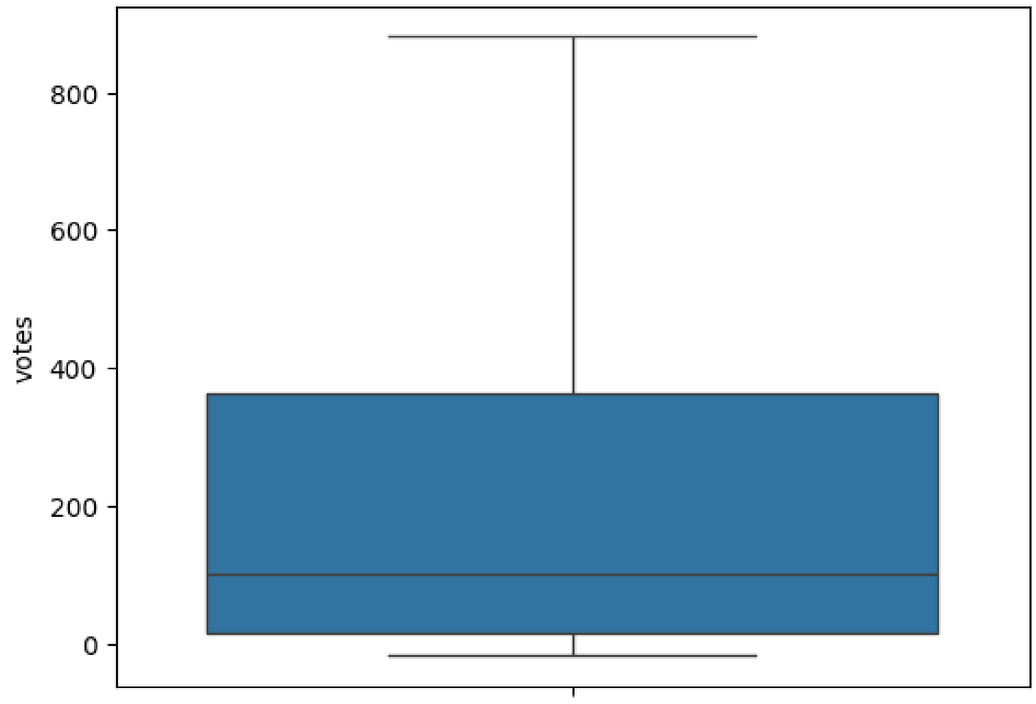
In [ ]:

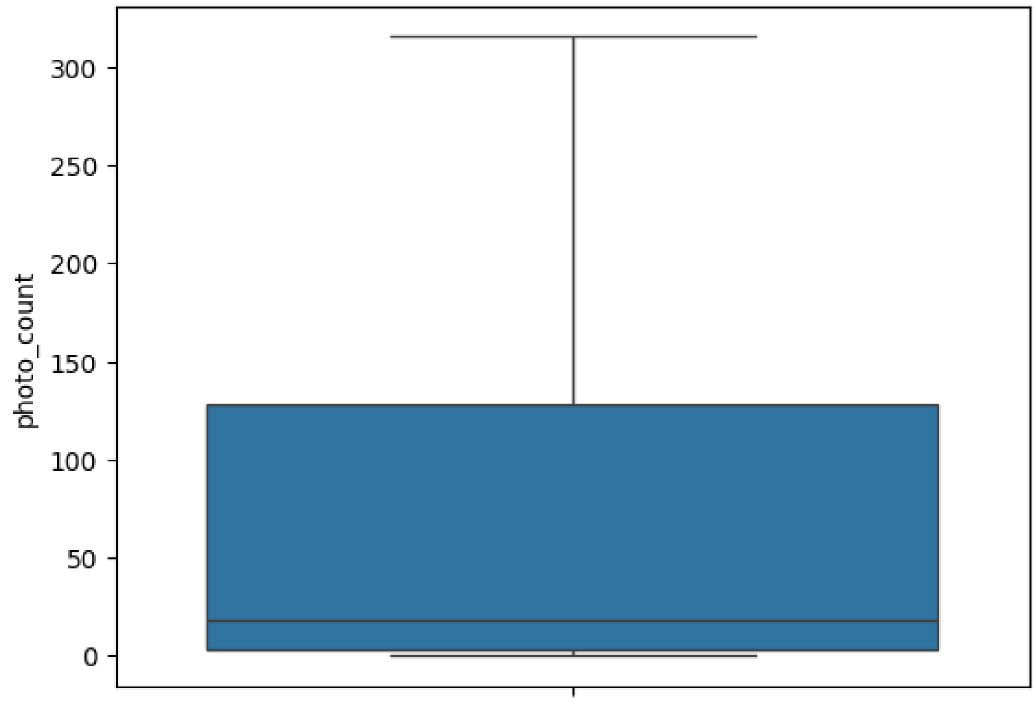
```
num_col1 = ["average_cost_for_two", "price_range", "votes", "photo_count"]
for i in num_col1:
    sns.boxplot(df[i])
    plt.show()
```











In [ ]:

```
data_preprocessd = df.copy()
df.to_csv("data_preprocessd.csv")
```

Basic Statistics:

Calculate and visualize the average rating of restaurants. Analyze the distribution of restaurant ratings to understand the overall rating landscape.

1. **Statistical Analysis**
2. **Univariate - analysis using single column/ feature in dataset**
3. **Bivariate analysis - analysis using two features/columns in dataset**
4. **Mulitivariate analysis - analysis using more than two features/columns in dataset**

In [ ]:

```
## summarize dataset
df.describe()
```

Out[ ]:

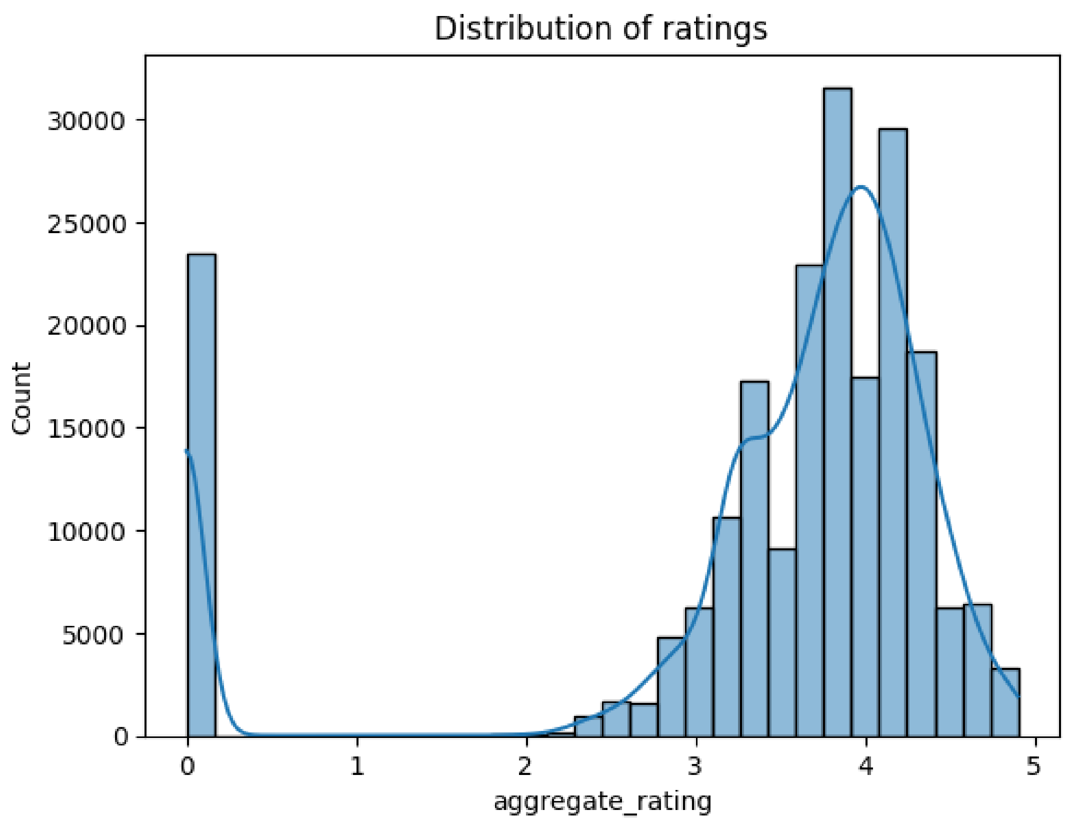
	res_id	city_id	latitude	longitude	country_id	average_cost_for_two	price_range	aggregate_rating	votes	photo_count	delivery	ta
count	2.119440e+05	211944.000000	211944.000000	211944.000000	211944.0	211944.000000	211944.000000	211944.000000	211944.000000	211944.000000	211944.000000	2
mean	1.349411e+07	4746.785434	21.499758	77.615276	1.0	535.667332	1.852848	3.395937	239.495282	83.740210	-0.255907	
std	7.883722e+06	5568.766386	22.781331	7.500104	0.0	378.328401	0.828051	1.283642	294.654997	115.748161	0.964172	

	res_id	city_id	latitude	longitude	country_id	average_cost_for_two	price_range	aggregate_rating	votes	photo_count	delivery	ta
min	5.000000e+01	1.000000	0.000000	0.000000	1.0	0.000000	1.000000	0.000000	-18.000000	0.000000	-1.000000	
25%	3.301027e+06	11.000000	15.496071	74.877961	1.0	250.000000	1.000000	3.300000	16.000000	3.000000	-1.000000	
50%	1.869573e+07	34.000000	22.514494	77.425971	1.0	400.000000	2.000000	3.800000	100.000000	18.000000	-1.000000	
75%	1.881297e+07	11306.000000	26.841667	80.219323	1.0	700.000000	2.000000	4.100000	362.000000	128.000000	1.000000	
max	1.915979e+07	11354.000000	10000.000000	91.832769	1.0	1375.000000	3.500000	4.900000	881.000000	315.500000	1.000000	

In [ ]:

```
## Average rating
print("Average Rating given by customers:", df["aggregate_rating"].mean())
## plot the ratings
sns.histplot(df["aggregate_rating"], bins = 30, kde = True)
plt.title("Distribution of ratings")
plt.show()
```

```
Average Rating given by customers: 3.3959366625146266
```

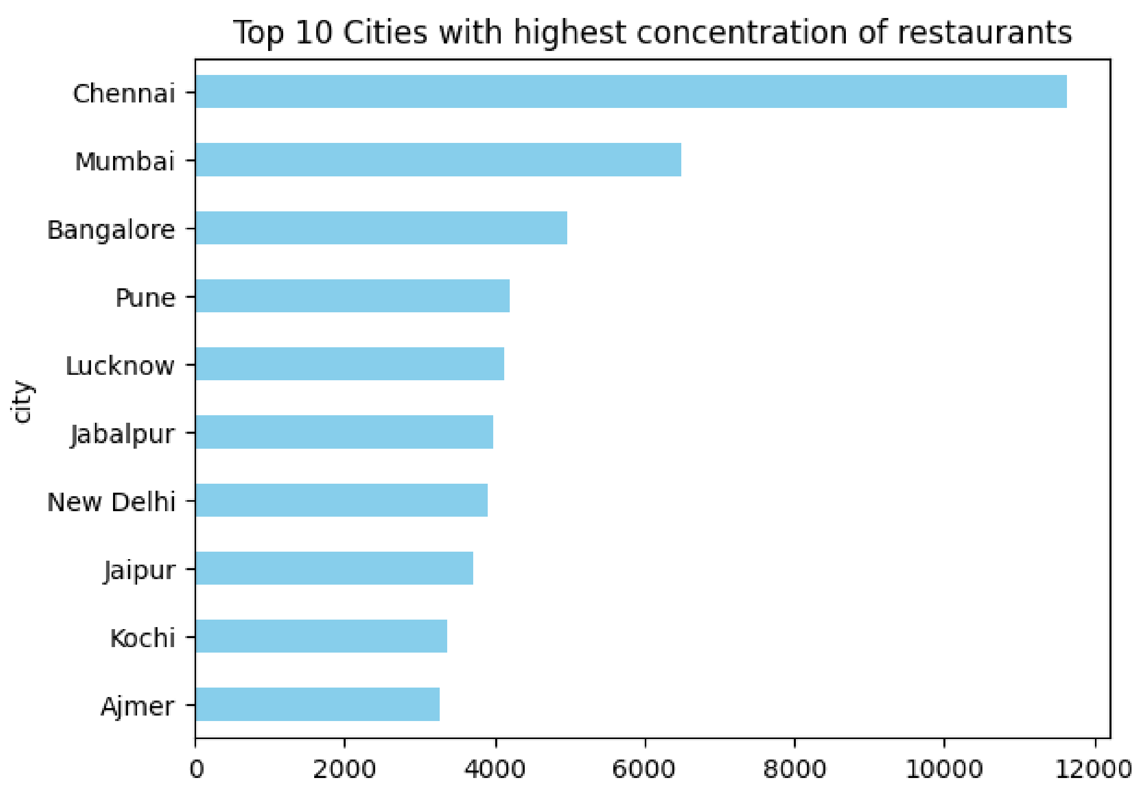


Location analysis

1. **Identify the city with the highest concentration of restaurants.**
2. **Visualize the distribution of restaurant ratings across different cities**

In [ ]:

```
city_count= df["city"].value_counts().sort_values(ascending=False).head(10)
city_count.iloc[::-1].plot(kind = "barh", color = "skyblue")
plt.title("Top 10 Cities with highest concentration of restaurants")
plt.show()
```



In [ ]:

```
df['city'].unique()
```

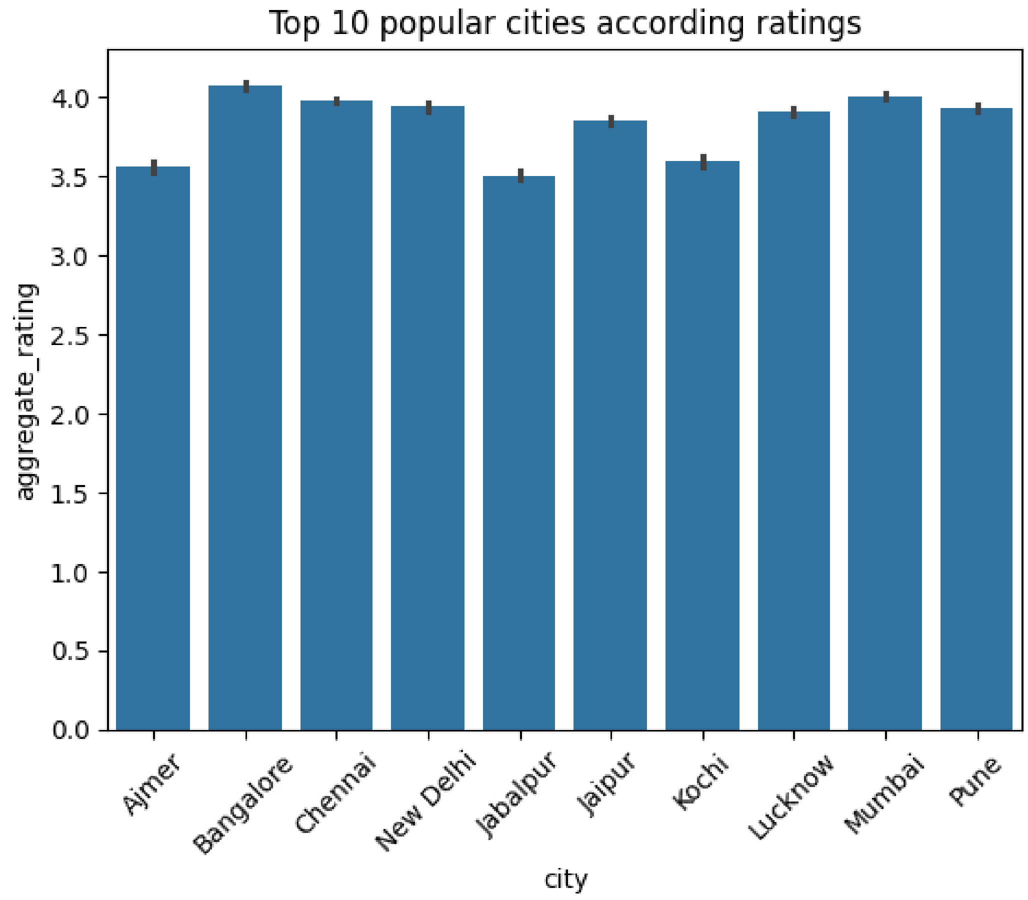
Out[ ]:

```
array(['Agra', 'Ahmedabad', 'Gandhinagar', 'Ajmer', 'Alappuzha',
       'Allahabad', 'Amravati', 'Amritsar', 'Aurangabad', 'Bangalore',
       'Bhopal', 'Bhubaneshwar', 'Chandigarh', 'Mohali', 'Panchkula',
       'Zirakpur', 'Nayagaon', 'Chennai', 'Coimbatore', 'Cuttack',
       'Darjeeling', 'Dehradun', 'New Delhi', 'Gurgaon', 'Noida',
       'Faridabad', 'Ghaziabad', 'Greater Noida', 'Dharamshala',
       'Gangtok', 'Goa', 'Gorakhpur', 'Guntur', 'Guwahati', 'Gwalior',
```

```
        'Haridwar', 'Hyderabad', 'Secunderabad', 'Indore', 'Jabalpur',
        'Jaipur', 'Jalandhar', 'Jammu', 'Jamnagar', 'Jamshedpur', 'Jhansi',
        'Jodhpur', 'Junagadh', 'Kanpur', 'Kharagpur', 'Kochi', 'Kolhapur',
        'Kolkata', 'Howrah', 'Kota', 'Lucknow', 'Ludhiana', 'Madurai',
        'Manali', 'Mangalore', 'Manipal', 'Udupi', 'Meerut', 'Mumbai',
        'Thane', 'Navi Mumbai', 'Mussoorie', 'Mysore', 'Nagpur',
        'Nainital', 'Nasik', 'Nashik', 'Neemrana', 'Ooty', 'Palakkad',
        'Patiala', 'Patna', 'Puducherry', 'Pune', 'Pushkar', 'Raipur',
        'Rajkot', 'Ranchi', 'Rishikesh', 'Salem', 'Shimla', 'Siliguri',
        'Srinagar', 'Surat', 'Thrissur', 'Tirupati', 'Trichy',
        'Trivandrum', 'Udaipur', 'Varanasi', 'Vellore', 'Vijayawada',
        'Vizag', 'Vadodara'], dtype=object)
```

In [ ]:

```
## Rating vs city
#since so many cities are there hence, we are creating a dataframe of all city column which will iterate through each city and fil
ter only which appears in city_count.index ie top 10
sns.barplot(x = "city", y = "aggregate_rating", data = df[df["city"].isin(city_count.index)])
plt.xticks(rotation = 45)
plt.title("Top 10 popular cities according ratings")
plt.show()
```



Cuisine Analysis:

Determine the most popular cuisines among the listed restaurants. Investigate if there's a correlation between the variety of cuisines offered and restaurant ratings

In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['res_id', 'name', 'establishment', 'url', 'address', 'city', 'city_id',
       'locality', 'latitude', 'longitude', 'country_id', 'locality_verbose',
       'cuisines', 'timings', 'average_cost_for_two', 'price_range',
       'currency', 'highlights', 'aggregate_rating', 'rating_text', 'votes',
       'photo_count', 'delivery', 'takeaway'],
      dtype='object')
```

In [ ]:

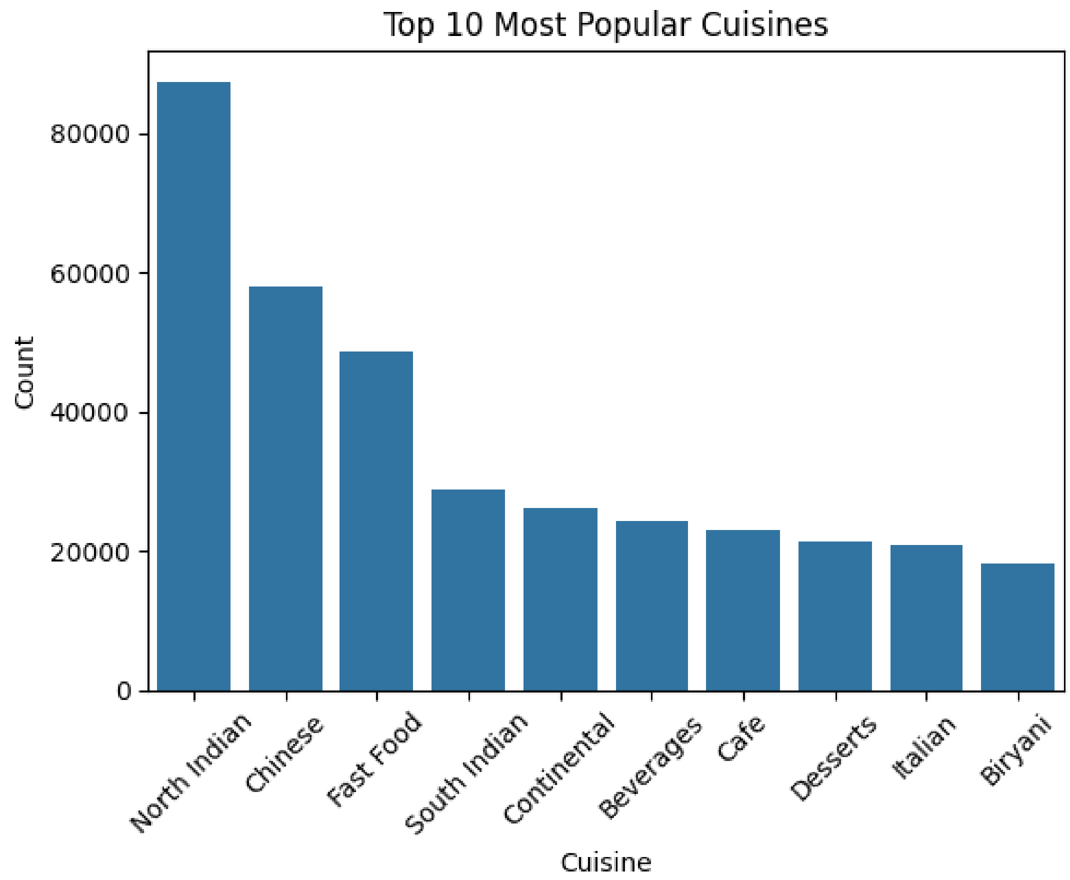
```
from collections import Counter
all_cuisines=df['cuisines'].dropna().str.split(",")
flat_cuisine_list = [cuisine.strip() for sublist in all_cuisines for cuisine in sublist]
cuisine_count=Counter(flat_cuisine_list)
cuisine_df=pd.DataFrame(cuisine_count.items(),columns=['Cuisine','Count']).sort_values(by='Count',ascending=False)
cuisine_df.head(10)
```

Out[ ]:

	Cuisine	Count
0	North Indian	87356
7	Chinese	57989
8	Fast Food	48584
1	South Indian	28895
10	Continental	26126
16	Beverages	24382
13	Cafe	23140
4	Desserts	21437
11	Italian	20920
35	Biryani	18315

In [ ]:

```
plt.figure(figsize=(6, 5))
sns.barplot(x='Cuisine', y='Count', data=cuisine_df.head(10))
plt.xticks(rotation=45)
plt.title("Top 10 Most Popular Cuisines")
plt.tight_layout()
plt.show()
```



North Indian,Chinese and Fast food cuisines dominate the restaurant scene, suggesting strong cultural preference and market demand.

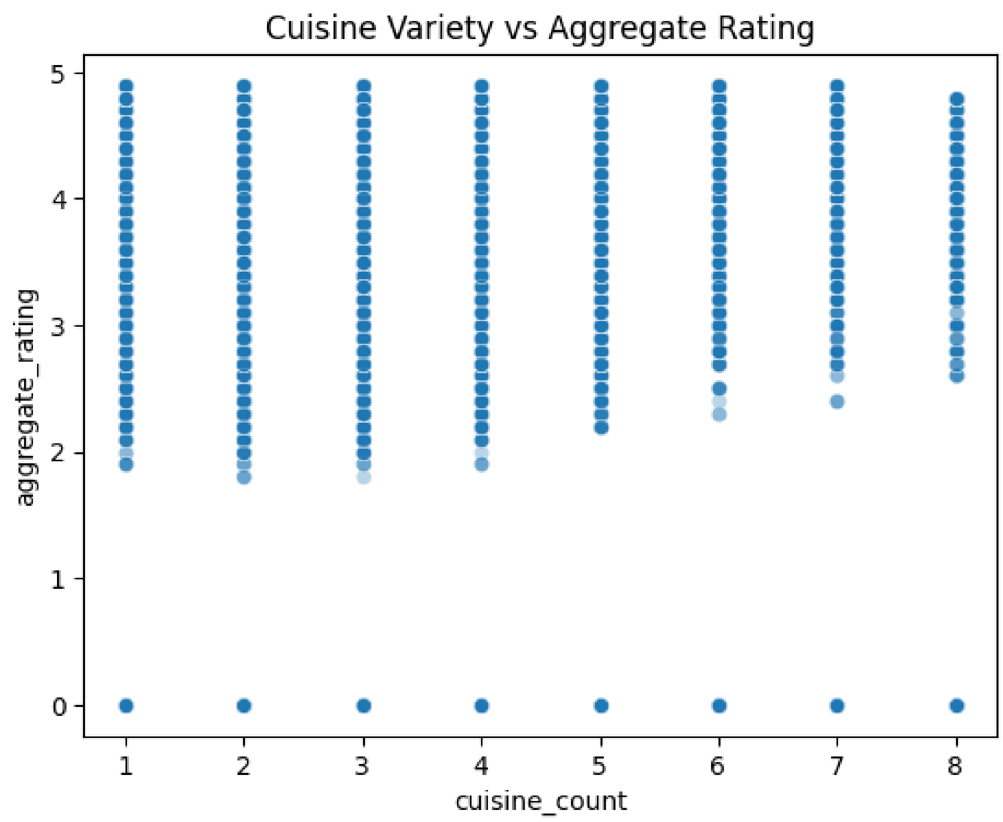
In [ ]:

```
# Add a new column for number of cuisines offered
df['cuisine_count'] = df['cuisines'].apply(lambda x: len(x.split(', ')) if pd.notnull(x) else 0)

# Check correlation
correlation = df['cuisine_count'].corr(df['aggregate_rating'])
print(f"Correlation is {correlation}")
```

```
Correlation is 0.23970696171223177
```

In [ ]:

```
sns.scatterplot(x='cuisine_count', y='aggregate_rating', data=df, alpha=0.3)
plt.title("Cuisine Variety vs Aggregate Rating")
plt.show()
```



There is a slight positive correlation (0.23) between the number of cuisines offered and restaurant ratings. This suggests that restaurants offering a wider variety of cuisines tend to have marginally better ratings, but the effect is not strong. Quality may still matter more than quantity.

Price Range and Rating:

1. **Analyze the relationship between price range and restaurant ratings.**
2. **Visualize the average cost for two people in different price categories**

In [ ]:

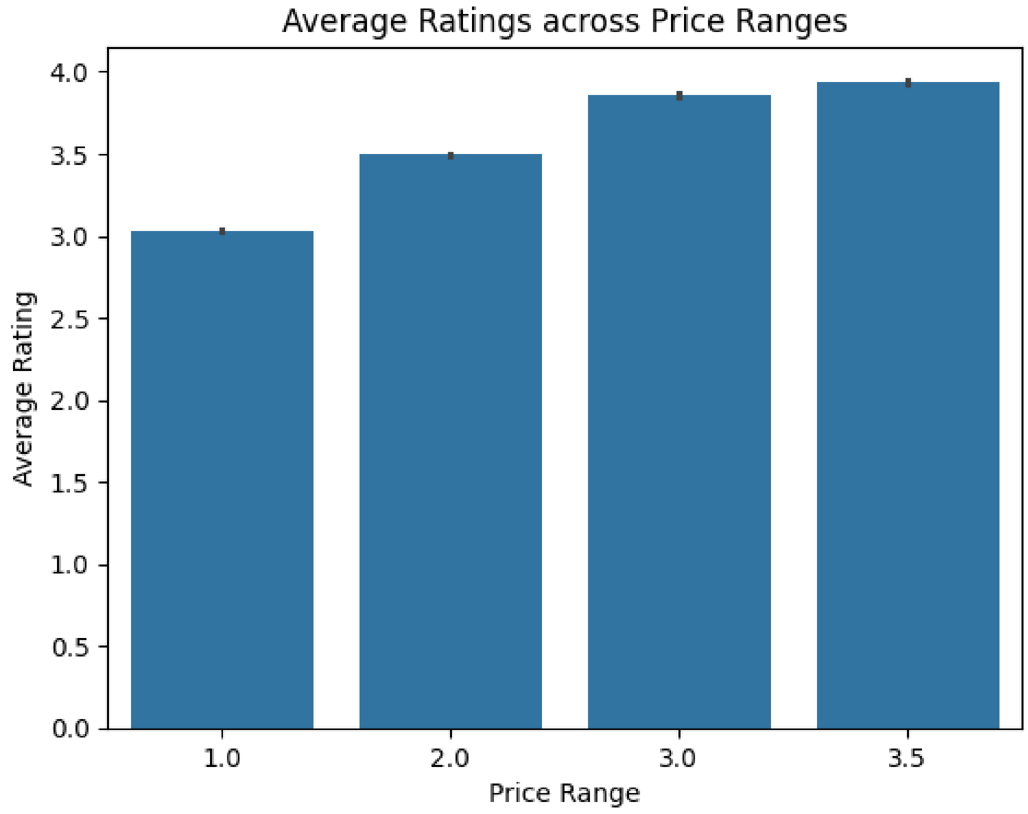
```
df['price_range'].unique()
```

Out[ ]:

```
array([2. , 1. , 3. , 3.5])
```

In [ ]:

```
sns.barplot(x='price_range', y='aggregate_rating', data=df)
plt.title("Average Ratings across Price Ranges")
plt.xlabel("Price Range")
plt.ylabel("Average Rating")
plt.show()
```



Price Range 3 and 3.5 (expensive) have high ratings between 3.5 to 4.0 indicating good value and affect of price ranges on ratings of restaurants.

Price Range 1 and 2(inexpensive to moderate) show slightly low ratings between 3.0 go 3.5, but not drastically different. It might indicate that people are ready to pay for high prices provided the quality of food should be good, which leads their satisfaction and better ratings, may be value for money.

In [ ]:

```
sns.barplot(x='price_range', y='average_cost_for_two', data=df)
plt.title("Average Cost for Two across Price Ranges")
plt.xlabel("Price Range")
plt.ylabel("Avg. Cost for Two")
plt.show()
```

As expected, the average cost for two increases with the price range, confirming price categorization is consistent.

Online Order and Table Booking:

Investigate the impact of online order availability on restaurant ratings. Analyze the distribution of restaurants that offer table booking.

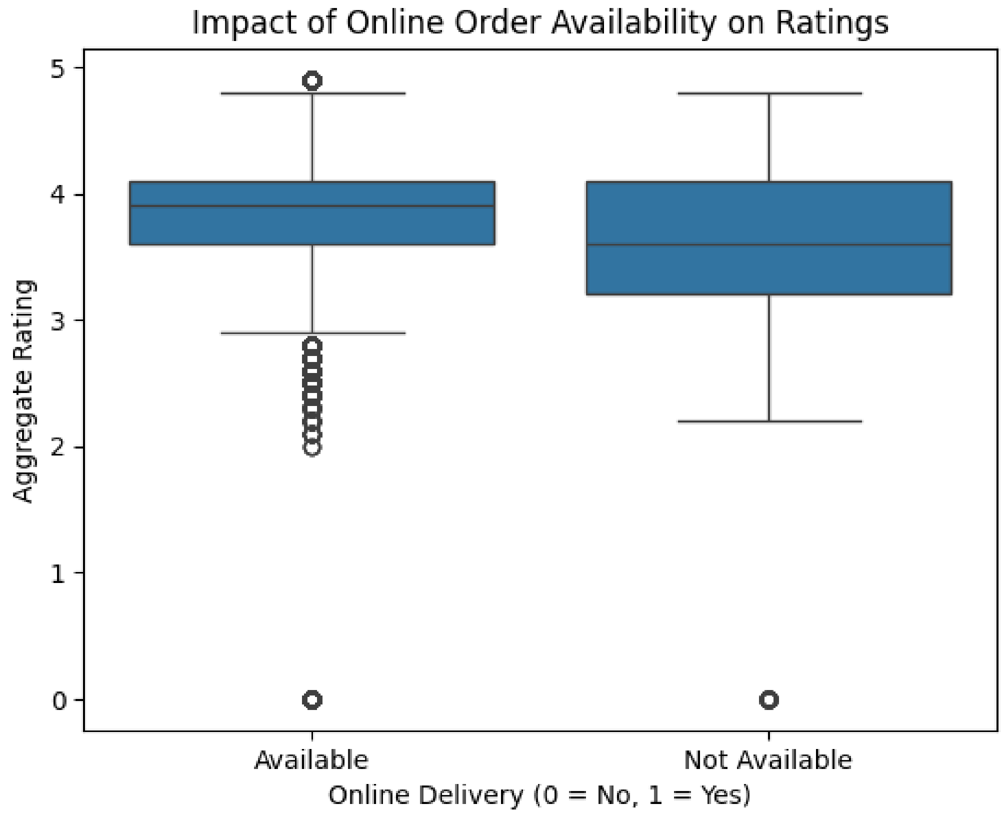
In [ ]:

```
#to not prevent the loss of data, as to remove -1 , we will have to entire rows, hence, creating different dataset
df_cleaned = df[df['delivery'] != -1]
```

In [ ]:

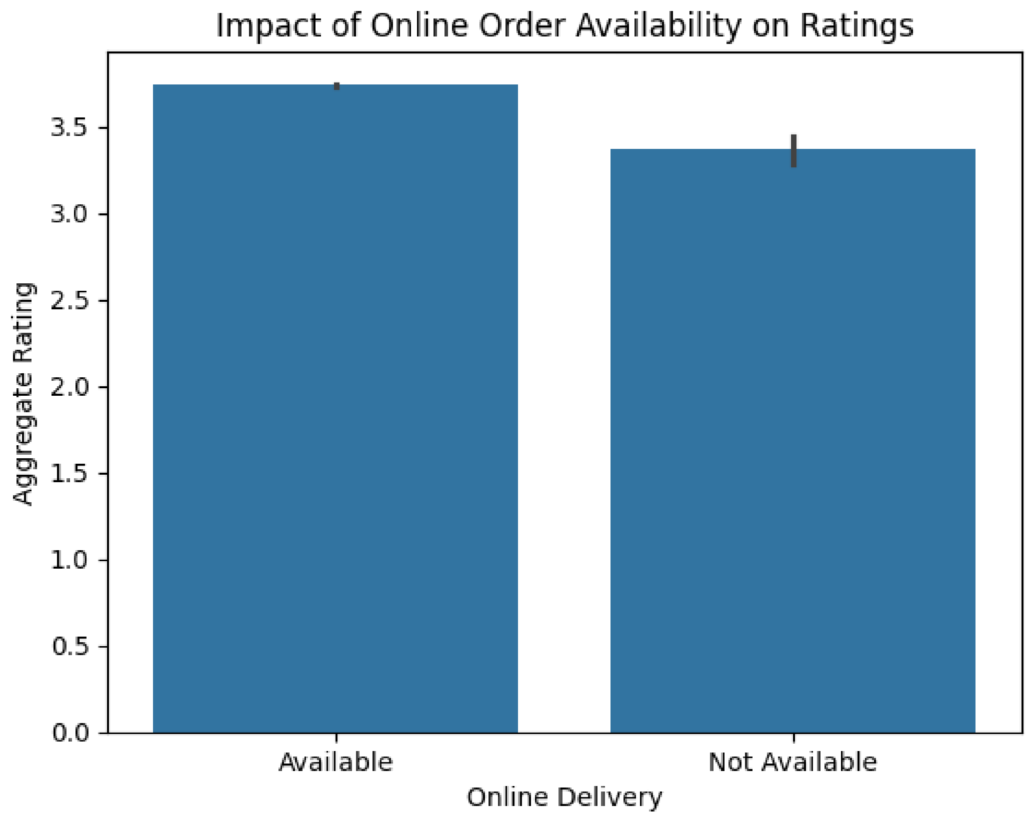
```
df_cleaned['delivery_mode'] = df_cleaned['delivery'].map({1: 'Available', 0: 'Not Available'})
sns.boxplot(x='delivery_mode', y='aggregate_rating', data=df_cleaned)
plt.title("Impact of Online Order Availability on Ratings")
plt.xlabel("Online Delivery (0 = No, 1 = Yes)")
```

```
plt.ylabel("Aggregate Rating")
plt.show()
```



In [ ]:

```
df_cleaned['delivery_mode'] = df_cleaned['delivery'].map({1: 'Available', 0: 'Not Available'})
sns.barplot(x='delivery_mode', y='aggregate_rating', data=df_cleaned)
plt.title("Impact of Online Order Availability on Ratings")
plt.xlabel("Online Delivery")
plt.ylabel("Aggregate Rating")
plt.show()
```



Restaurants with online ordering available tend to have slightly higher aggregate ratings compared to those without. Median Rating: The median rating for restaurants with online ordering is slightly higher than those without,indicating that the middle 50% of restaurants with online ordering have higher ratings.

Can not perform analysis on tabel booking as the column has been dropped because it only contained on value which was "0".

Top Restaurant Chains:

Identify and visualize the top restaurant chains based on the number of outlets. Explore the ratings of these top chains.

In [ ]:

```
top_res_chains = df['name'].value_counts().head(10)
top_res_chains
```

Out[ ]:

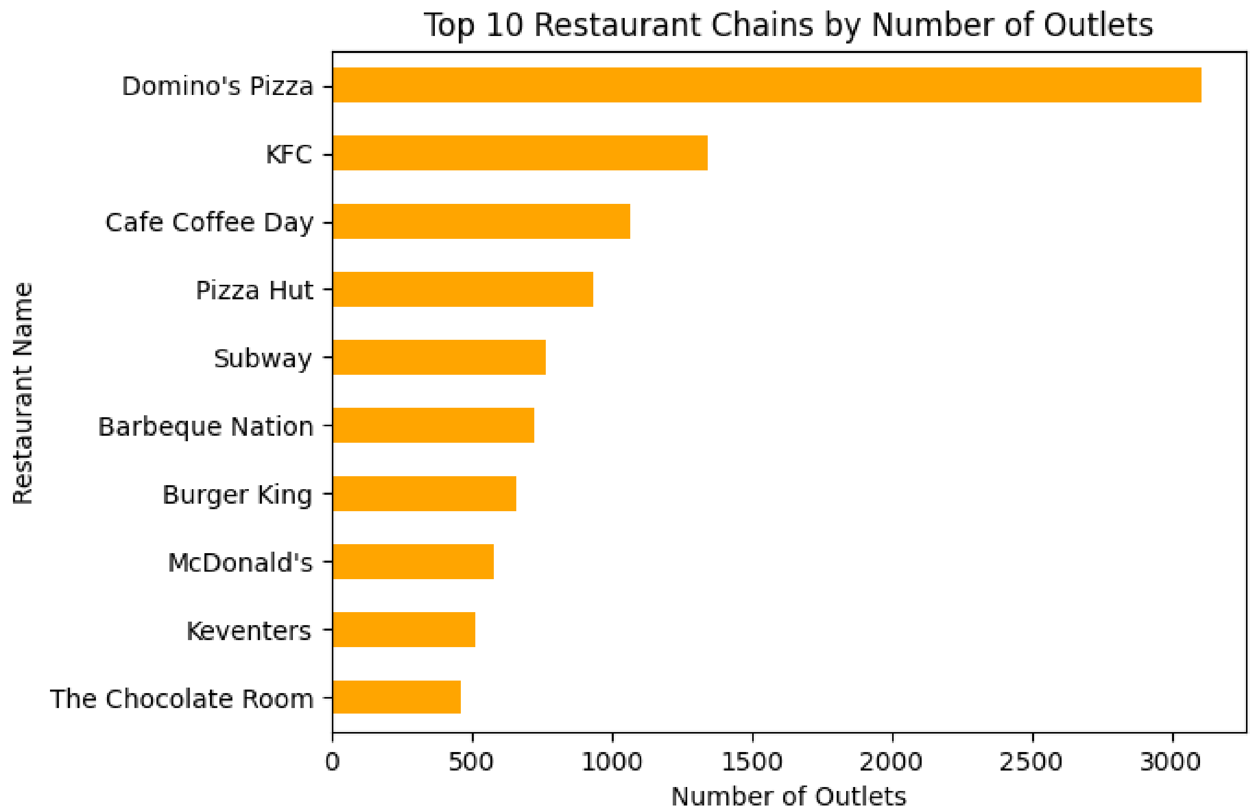
	count
name	
Domino's Pizza	3108
KFC	1343
Cafe Coffee Day	1068
Pizza Hut	936
Subway	766
Barbeque Nation	725
Burger King	658
McDonald's	578

name	count
McDonald's	578
Keventers	512
The Chocolate Room	461

dtype: int64

In [ ]:

```
top_res_chains.iloc[::-1].plot(kind='barh', color='orange')
plt.title('Top 10 Restaurant Chains by Number of Outlets')
plt.xlabel('Number of Outlets')
plt.ylabel('Restaurant Name')
plt.show()
```



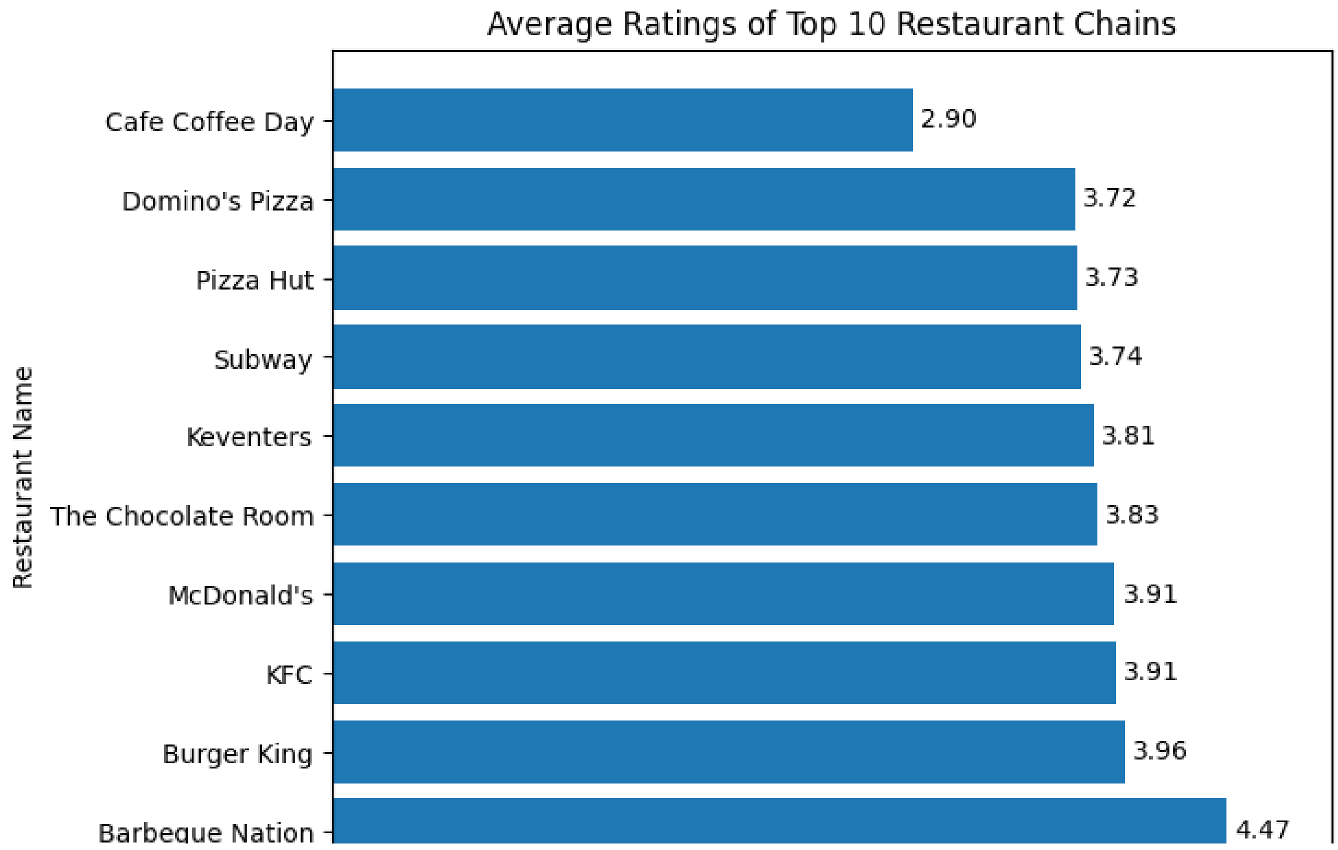
In [ ]:

```
top_rest_chains = df['name'].value_counts().head(10).index.tolist()
df_top = df[df['name'].isin(top_rest_chains)]
top_chains_avg_rating = df_top.groupby('name')['aggregate_rating'].mean().reset_index()
top_chains_avg_rating = top_chains_avg_rating.sort_values(by='aggregate_rating', ascending=False)
print(top_chains_avg_rating)
```

```
                 name  aggregate_rating
0     Barbeque Nation          4.472966
1         Burger King          3.964438
4                 KFC          3.913924
6           McDonald's          3.912457
9  The Chocolate Room          3.825380
5           Keventers          3.809570
8              Subway          3.742950
7           Pizza Hut          3.726389
3      Domino's Pizza          3.716216
2     Cafe Coffee Day          2.904963
```

In [ ]:

```
fig, ax = plt.subplots(figsize=(7, 6))
bars = ax.barh(top_chains_avg_rating['name'], top_chains_avg_rating['aggregate_rating'])
ax.bar_label(bars, fmt='%.2f', padding=3)
ax.set_title('Average Ratings of Top 10 Restaurant Chains')
ax.set_xlabel('Average Rating')
ax.set_ylabel('Restaurant Name')
ax.set_xlim(0, 5)
plt.show()
```



1)The top restaurants chains with maximum number of outlets are 1. Dominos Pizza 2. KFC and 3. Coffee café Day. 2.)Barbeque has low number of outlets between 600 to 1000, still, it has high rating of 4.47.Similary for Burger King and McDonald's. 3)KFC has many outlets and also has an average rating of 3.97. 4)This shows that number of outlets not necessarily promise high ratings.

Word Cloud for Reviews:

1. **Create a word cloud based on customer reviews to identify common positive and negative sentiments.**
2. **Analyze frequently mentioned words and sentiments**

In [ ]:

```
df.head(2)
```

Out[ ]:

	res_id	name	establishment	url	address	city	city_id	locality	latitude	longitude	...	price_range	currency	highlights	aggre
0	3400299	Bikanervala	['Quick Bites']	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	Agra	34	Khandari	27.211450	78.002381	...	2.0	Rs.	['Lunch', 'Takeaway Available', 'Credit Card',...	
1	3400005	Mama Chicken Mama Franky House	['Quick Bites']	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra	Agra	34	Agra Cantt	27.160569	78.011583	...	2.0	Rs.	['Delivery', 'No Alcohol Available', 'Dinner',...	

2 rows × 25 columns

In [ ]:

```
# !pip install wordcloud
```

In [ ]:

```
from wordcloud import WordCloud
```

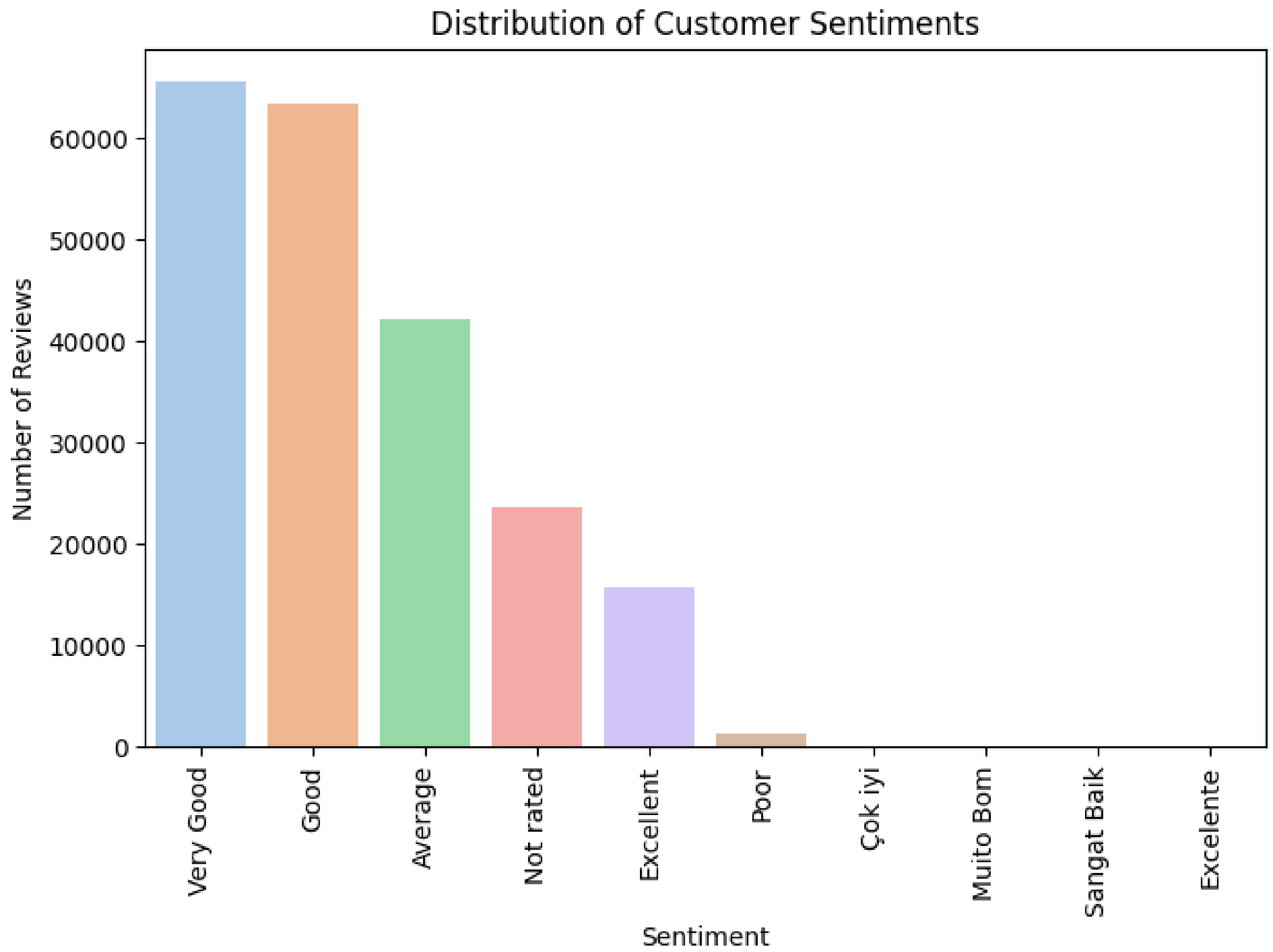
In [ ]:

```
if "rating_text" in df.columns:
    review_txt = ' '.join(df["rating_text"].dropna().tolist())
    wordcloud = WordCloud(width = 800, height = 400, background_color = "white").generate(review_txt)
    plt.figure(figsize = (12,8))
    plt.imshow(wordcloud, interpolation =  "bilinear")
    plt.axis("off")
    plt.title("Word cloud of Customer Reviews")
    plt.show()
```

In [ ]:

```
sentiment_counts = df['rating_text'].value_counts().reset_index().head(10)
sentiment_counts.columns = ['Sentiment', 'Count']
plt.figure(figsize=(8, 5))
```

```
sns.barplot(data=sentiment_counts, x='Sentiment', y='Count', palette='pastel')
plt.title("Distribution of Customer Sentiments")
plt.xlabel("Sentiment")
plt.ylabel("Number of Reviews")
plt.xticks(rotation=90)
plt.show()
```



Maximum number of reviews(more then 60000) are of "Very good" and "good". Very few reviews are of "Poor" and Significant number of reviews between 10000 to 20000 are for "Excellent". This means People are actually to an extent happy with the services of Restaurants which might be the reason of success of restaurants and Zomato collectively.

Restaurant Features:

1. **Analyze the distribution of restaurants based on features like Wi-Fi, Alcohol availability, etc.**
2. **Investigate if the presence of certain features correlates with higher ratings**

In [ ]:

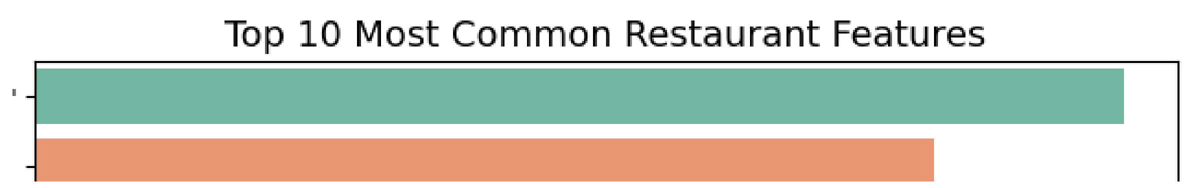
```
from collections import Counter
#already in list format in highlight column so not splitting
valid_highlights = df['highlights'].dropna()
flat_list = []
for features in valid_highlights:
    for item in features:
        flat_list.append(item)
feature_counts = Counter(flat_list)
# DataFrame
feature_df = pd.DataFrame(feature_counts.items(), columns=['Feature', 'Count']).sort_values(by='Count', ascending=False)
feature_df.head(10)
```

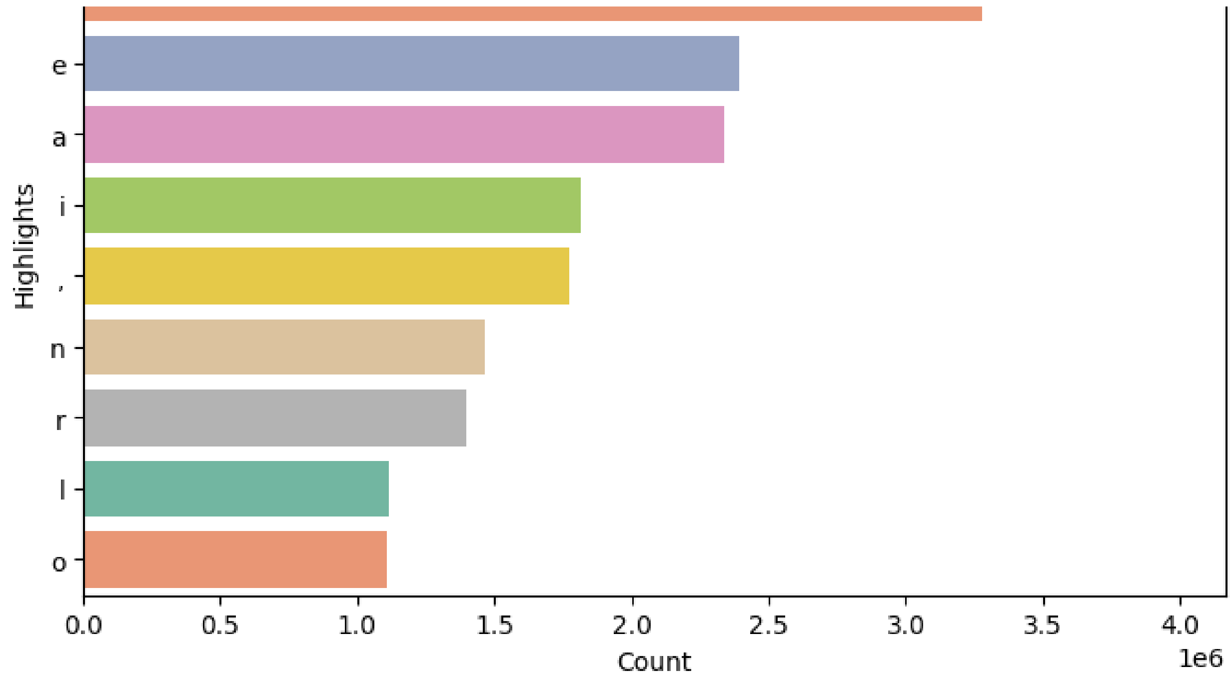
Out[ ]:

	Feature	Count
1	'	3968606
8		3282254
12	e	2390015
10	a	2335973
17	i	1812147
7	,	1774428
4	n	1465283
21	r	1399987
18	l	1116039
26	o	1106925

In [ ]:

```
plt.figure(figsize=(8, 5))
sns.barplot(x='Count', y='Feature', data=feature_df.head(10), palette='Set2')
plt.title("Top 10 Most Common Restaurant Features", fontsize=14)
plt.xlabel("Count")
plt.ylabel("Highlights")
plt.show()
```





In [ ]:

```
df['has_wifi'] = df['highlights'].apply(lambda x: 'Wifi' in x if isinstance(x, list) else False)
```

In [ ]:

```
df.groupby('has_wifi')['aggregate_rating'].mean()
```

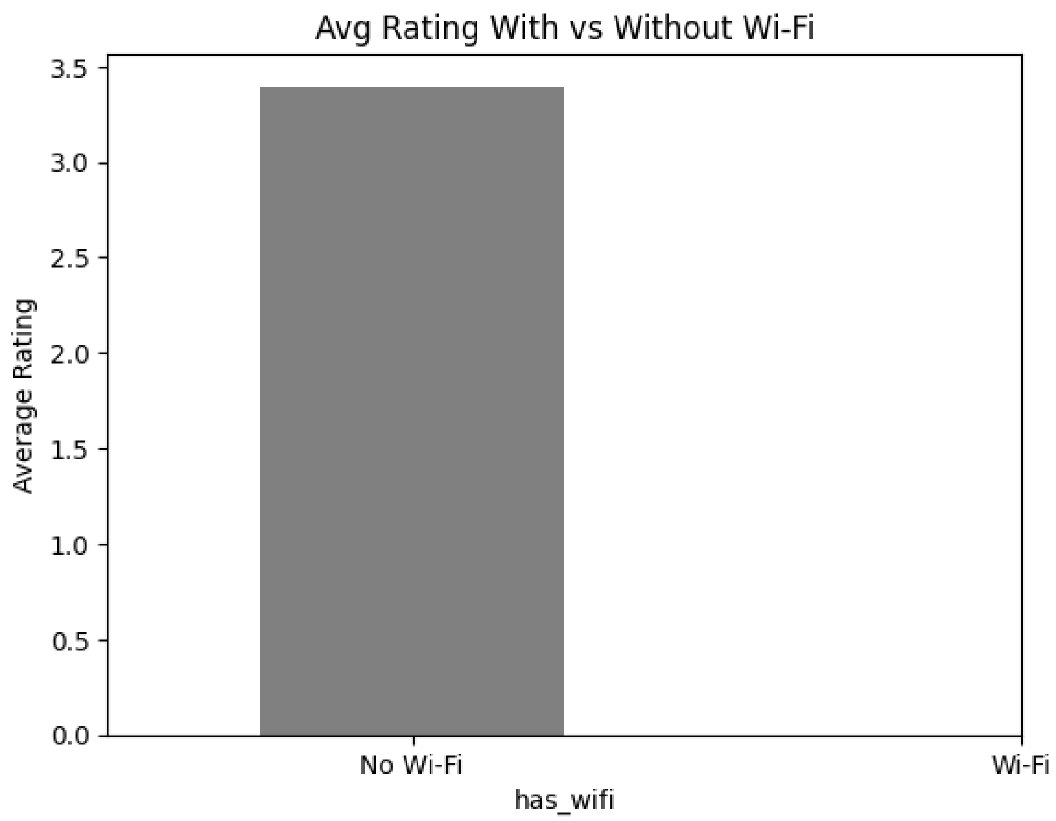
Out[ ]:

	aggregate_rating
has_wifi	
False	3.395937

dtype: float64

In [ ]:

```
df.groupby('has_wifi')['aggregate_rating'].mean().plot(kind='bar', color=['gray', 'skyblue'])
plt.title("Avg Rating With vs Without Wi-Fi")
plt.xticks([0,1], ['No Wi-Fi', 'Wi-Fi'], rotation=0)
plt.ylabel("Average Rating")
plt.show()
```



In [ ]:

```
df['has_air_conditioner'] = df['highlights'].apply(lambda x: 'Air Conditioned' in x if isinstance(x, list) else False)
```

In [ ]:

```
df.groupby('has_air_conditioner')['aggregate_rating'].mean()
```

Out[ ]:

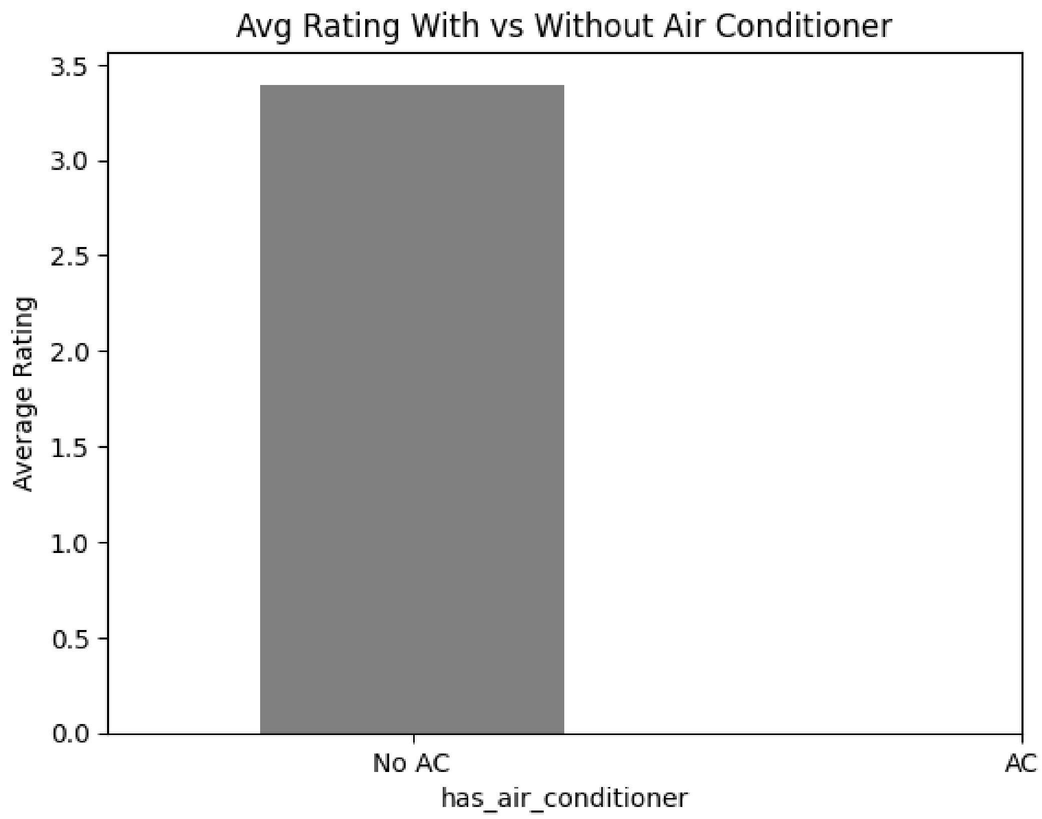
	aggregate_rating
has_air_conditioner	
False	3.395937

dtype: float64

In [ ]:

```
df.groupby('has_air_conditioner')['aggregate_rating'].mean().plot(kind='bar', color=['gray', 'skyblue'])
plt.title("Avg Rating With vs Without Air Conditioner")
```

```
plt.xticks([0,1], ['No AC', 'AC'], rotation=0)
plt.ylabel("Average Rating")
plt.show()
```



1)Most common features are "Cash" payment mode, "Takeaway" and "Indoor Sitting". 2)People have rated restaurants with facilites like having Wi-fi and Air conditioner more.

In [ ]:

```
## Abstract syntax library - It segregates different literals for a common data type. Functions as eval but it has more
# strong bond with data type
'''import ast

df["highlights"] = df["highlights"].apply(ast.literal_eval)

## seaparte unique features for each list
all_feat = set([j for i in df["highlights"] for j in i])

## One hot encoding
for i in all_feat:
    df[i] = df["highlights"].apply(lambda x: 1 if i in x else 0)'''
```

Out[ ]:

```
'import ast\n\ndf["highlights"] = df["highlights"].apply(ast.literal_eval)\n\n## seaparte unique features for each list\nall_feat = set([j for i in df["highlights"] for j in i])\n\n## One hot encoding \nfor i in all_feat:\n    df[i] = df["highlights"].apply(lambda x: 1 if i in x else 0)'
```

In [ ]:

```
#df.columns
```

In [ ]:

```
#df["Wifi"].value_counts()
```

In [ ]:

```
## if restaurats have wifi
'''def plot(df, x):
    data = df.groupby(x)["aggregate_rating"].mean()
    sns.barplot(data)
    plt.title(f"{x} vs aggregate_rating")
    plt.show()'''
```

Out[ ]:

```
'def plot(df, x):\n    data = df.groupby(x)["aggregate_rating"].mean()\n    sns.barplot(data)\n    plt.title(f"{x} vs aggregate_rating")\n    plt.show()'
```

In [ ]:

```
#plot(df=df, x="Wifi")
```

In [ ]:

```
## No Alcohol Available
'''no_alco_rating = df.groupby("No Alcohol Available")["aggregate_rating"].mean()
sns.barplot(no_alco_rating)'''
```

```
  File "<ipython-input-310-b793f7b0004a>", line 2
    '''no_alco_rating = df.groupby("No Alcohol Available")["aggregate_rating"].mean()
    ^
SyntaxError: incomplete input
```

In [ ]:

```
#data_preprocessd.info()
```

In [ ]:

```
#plot(df=df, x="Table booking for Groups")
```

In [ ]:

```
'''num_col = data_preprocessd.select_dtypes(include = ["int","float64"])
num_col.head(2)'''
```

In [ ]:

```
## Correlation - its multivariate analysis
plt.figure(figsize=(14,12))
sns.heatmap(num_col.corr(), annot = True, cmap = "viridis")
plt.title("Correlation between features")
plt.show()
```

In [ ]:

```
#saving data to prevent any mishap
new_data = pd.read_csv("data_preprocessd.csv")
new_data.head()
```