Restaurant Rating Analysis

```
Importing libraries
```

In [2]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```python
Data = pd.read_csv('Dataset .csv')
```

In [4]:
```python
Data.head()  #view the first rows
```

Out[4]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Long
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.02
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.0
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.0
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.0
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.0

5 rows × 21 columns

In [5]:
```python
Data.info()    #Get information about the data tpes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Restaurant ID         9551 non-null   int64
 1   Restaurant Name       9551 non-null   object
 2   Country Code          9551 non-null   int64
 3   City                  9551 non-null   object
 4   Address               9551 non-null   object
 5   Locality              9551 non-null   object
 6   Locality Verbose      9551 non-null   object
 7   Longitude             9551 non-null   float64
 8   Latitude              9551 non-null   float64
 9   Cuisines              9542 non-null   object
 10  Average Cost for two  9551 non-null   int64
 11  Currency              9551 non-null   object
 12  Has Table booking     9551 non-null   object
 13  Has Online delivery   9551 non-null   object
 14  Is delivering now     9551 non-null   object
 15  Switch to order menu  9551 non-null   object
 16  Price range           9551 non-null   int64
 17  Aggregate rating      9551 non-null   float64
 18  Rating color          9551 non-null   object
 19  Rating text           9551 non-null   object
 20  Votes                 9551 non-null   int64
dtypes: float64(3), int64(5), object(13)
memory usage: 1.5+ MB
```

In [6]:
```python
Data.describe()   #summary statastics for numerical columns
```

Out[6]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Ag
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	(
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4

In [7]:
```
Data.isnull().sum() #Check for missing values
```

Out[7]:
```
Restaurant ID           0
Restaurant Name         0
Country Code            0
City                    0
Address                 0
Locality                0
Locality Verbose        0
Longitude               0
Latitude                0
Cuisines                9
Average Cost for two    0
Currency                0
Has Table booking       0
Has Online delivery     0
Is delivering now       0
Switch to order menu    0
Price range             0
Aggregate rating        0
Rating color            0
Rating text             0
Votes                   0
dtype: int64
```

Determine the top three most common cuisines in the dataset.

In [8]:
```
cuisine_count = Data['Cuisines'].value_counts()
top_3_cuisine = cuisine_count.head(3)
top_3_cuisine
```

Out[8]:
```
Cuisines
North Indian               936
North Indian, Chinese      511
Chinese                    354
Name: count, dtype: int64
```

Calculate the percentage of restaurants that serve each of the top cuisines.

In [9]:
```

total_restaurants = len(Data) #9551

percent_top_cuisines = (top_3_cuisine/total_restaurants)*100
percent_top_cuisines

```

Out[9]:
```
Cuisines
North Indian               9.800021
North Indian, Chinese      5.350225
Chinese                    3.706418
Name: count, dtype: float64
```

Identify the city with the highest number of restaurants in the dataset.

In [10]:
```
city_count = Data['City'].value_counts()
city_with_highest_restaurants = city_count.idxmax()
city_with_highest_restaurants
```

Out[10]: 'New Delhi'

Calculate the average rating for restaurants in each city.

In [11]:
```
Avg_rating_by_city = Data.groupby('City')['Aggregate rating'].mean()
Avg_rating_by_city
```

Out[11]:
```
City
Abu Dhabi          4.300000
Agra               3.965000
Ahmedabad          4.161905
Albany             3.555000
Allahabad          3.395000
                     ...
Weirton            3.900000
Wellington City    4.250000
Winchester Bay     3.200000
Yorkton            3.300000
��stanbul          4.292857
Name: Aggregate rating, Length: 141, dtype: float64
```

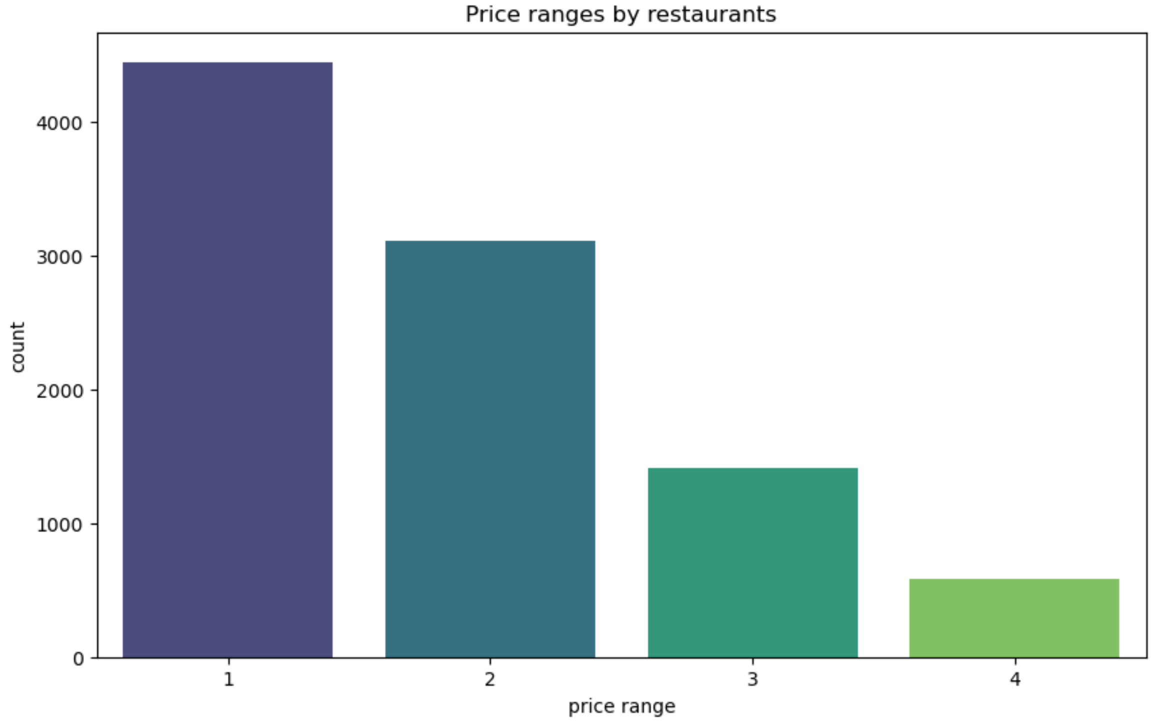
Determine the city with the highest average rating.

In [12]:
```
Avg_rating_by_city = Data.groupby('City')['Aggregate rating'].mean()
highest_avg_rating = Avg_rating_by_city.idxmax()
highest_avg_rating
```

Out[12]: 'Inner City'

Create a histogram or bar chart to visualize the distribution of price ranges among the restaurants

In [13]:
```
plt.figure(figsize=(10,6))
sns.countplot(x = 'Price range',data = Data,palette='viridis')
plt.title('Price ranges by restaurants')
plt.xlabel('price range')
plt.ylabel('count')
plt.show()
```



Calculate the percentage of restaurants in each price range category.

In [14]:
```
price_range_percentage = Data['Price range'].value_counts(normalize=Tr
price_range_percentage
```

Out[14]:
```
Price range
1    46.529159
2    32.593446
3    14.741912
4     6.135483
Name: proportion, dtype: float64
```

Determine the percentage of restaurants that offer online delivery.

In [15]:
```
Percent_onlinedelivery = Data['Has Online delivery'].value_counts(norm
Percent_onlinedelivery
```

Out[15]:
```
Has Online delivery
No     74.337766
Yes    25.662234
Name: proportion, dtype: float64
```

Compare the average ratings of restaurants with and without online delivery.

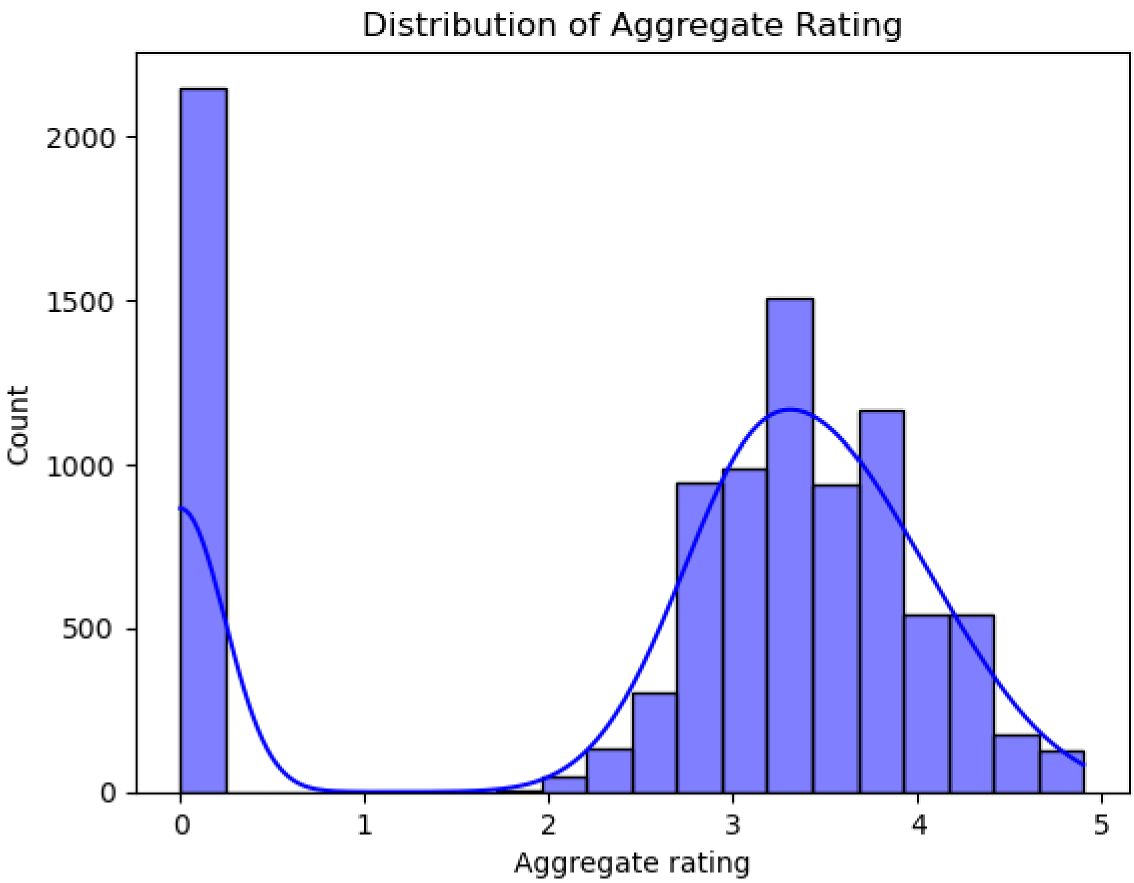
In [16]:
```python
Avg_ratings = Data.groupby('Has Online delivery')['Aggregate rating'].
Avg_ratings
```

Out[16]:
```
Has Online delivery
No     2.465296
Yes    3.248837
Name: Aggregate rating, dtype: float64
```

Analyze the distribution of aggregate ratings and determine the most common rating range.

In [17]:
```python
sns.histplot(Data['Aggregate rating'],bins = 20,kde = True,color = 'bl
plt.title('Distribution of Aggregate Rating')
plt.show
```

Out[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Calculate the average number of votes received by restaurants.

In [18]:
```python
Avg_votes = Data['Votes'].mean()
Avg_votes
```

Out[18]:
```
156.909747670401
```

Identify the most common combinations of cuisines in the dataset.

In [19]:
```python
Data['Combined cuisines'] = Data['Cuisines'].str.split(', ')
common_cuisines = Data['Combined cuisines'].explode().value_counts()

#The explode() function is then used to transform each element of the

most_common_cuisines = common_cuisines.head(3)
most_common_cuisines
```

Out[19]:
```
Combined cuisines
North Indian    3960
Chinese         2735
Fast Food       1986
Name: count, dtype: int64
```

Determine if certain cuisine combinations tend to have higher ratings.

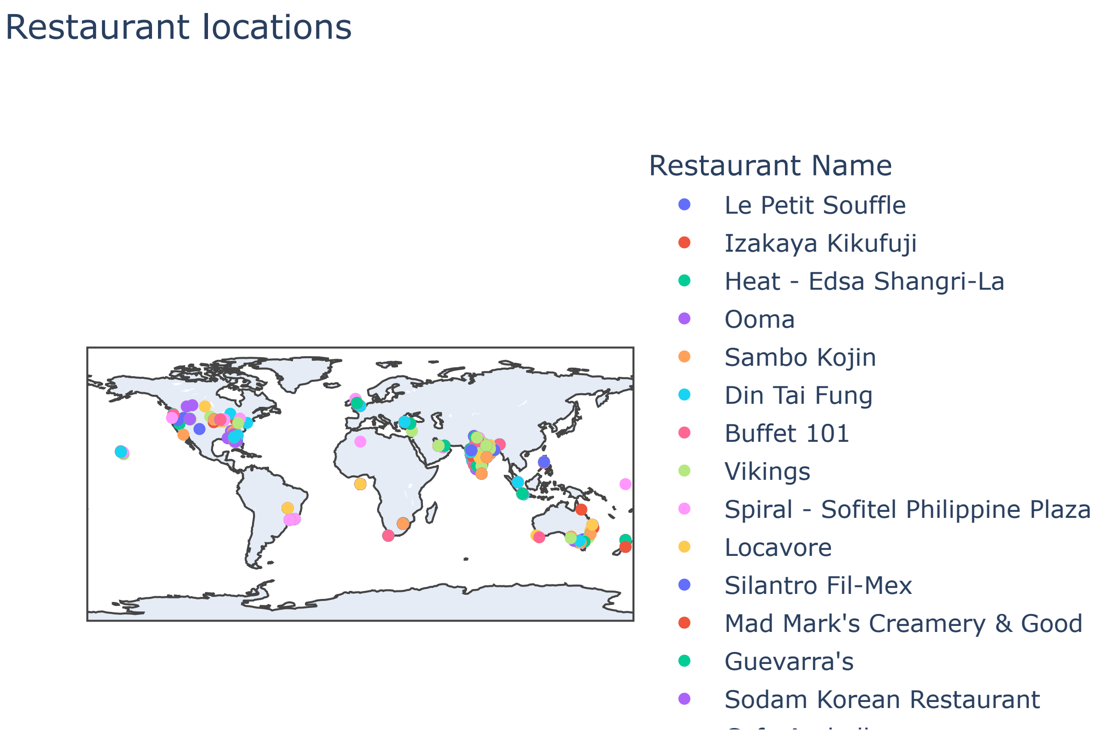
In [20]:
```python
Data['Combined cuisines'] = Data['Cuisines'].str.split(', ')
common_cuisines = Data.explode('Combined cuisines')
Avg_rating_cusisine = common_cuisines.groupby('Combined cuisines')['Ag
Avg_rating_cusisine
```

Out[20]:
```
Combined cuisines
Afghani          1.971429
African          3.525000
American         3.661538
Andhra           3.870000
Arabian          3.385714
                   ...
Turkish Pizza    4.325000
Vegetarian       4.073913
Vietnamese       3.923810
Western          4.140000
World Cuisine    4.300000
Name: Aggregate rating, Length: 145, dtype: float64
```

In [21]:
```python
import plotly.express as px  #for Geograph map
```

Plot the locations of restaurants on a map using longitude and latitude coordinates.

```
In [32]: fig = px.scatter_geo(Data,
                             lat ='Latitude',
                             lon = 'Longitude',
                             color = 'Restaurant Name',
                             hover_name = 'Restaurant Name',
                             title = 'Restaurant locations')

         fig.show()
```



Identify if there are any restaurant chains present in the dataset

In [23]:
```
#Clean and normalize the restaurant names for better comparison
Data['Cleaned Name'] = Data['Restaurant Name'].str.lower().str.strip()

# Find restaurant chains by counting the occurrences of each name
chain_counts = Data['Cleaned Name'].value_counts()

# Display the names that occur more than once (potential chains)
potential_chains =chain_counts[chain_counts > 1]
potential_chains
```

Out[23]:
```
Cleaned Name
cafe coffee day        83
domino's pizza         79
subway                 63
green chick chop       51
mcdonald's             48
                       ..
jack po!tato's          2
metro fast food         2
the mirch masala        2
punjabi chicken         2
south indian corner     2
Name: count, Length: 742, dtype: int64
```

Analyze the ratings and popularity of different restaurant chains.

In [24]:
```
Data['Cleaned Name'] = Data['Restaurant Name'].str.lower().str.strip()

#calculating avg rating for restaurant chain
Avg_rating_chain = Data.groupby('Cleaned Name')['Aggregate rating'].me

#Calculate total num of votes per each chain
total_chain = Data.groupby('Cleaned Name')['Votes'].sum()

#combined the results into  a new dataframe
chain_analysis_df = pd.DataFrame({
    'Average Rating': Avg_rating_chain,
    'Total Votes': total_chain})

# Sort the DataFrame by average rating in descending order

chain_analysis_df = chain_analysis_df.sort_values(by='Average Rating',

chain_analysis_df
```

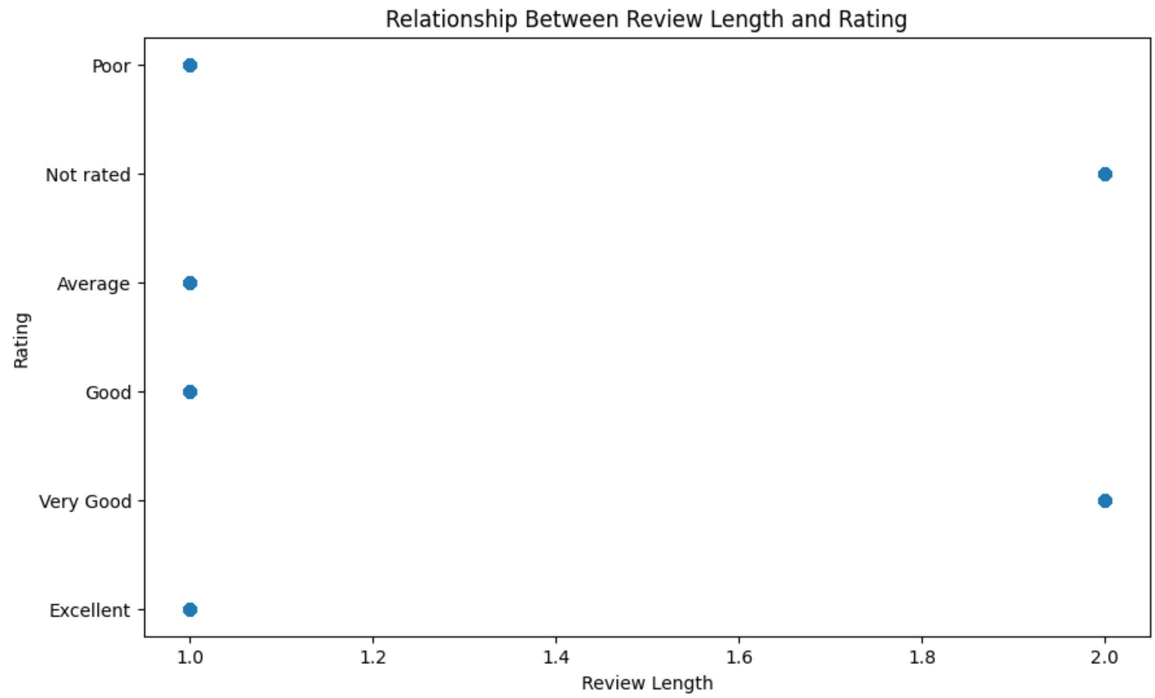
Out[24]:

Cleaned Name	Average Rating	Total Votes
braseiro da g��vea	4.9	40
masala library	4.9	408
milse	4.9	754
solita	4.9	162
miann	4.9	281
...	...	...
m cr��me	0.0	0
m&s coffee cafe	0.0	3
sunrise bakery	0.0	0
royal bakery	0.0	2
khalsa eating point	0.0	0

7433 rows × 2 columns

Calculate the average length of reviews and explore if there is a relationship between review length and rating..

In [47]:
```
# Calculate the length of each review
Data['Review Length'] = Data['Rating text'].apply(lambda x: len(str(x)

# Calculate the average review length
average_review_length = Data['Review Length'].mean()
average_review_length

# Explore the relationship between review length and rating
plt.figure(figsize=(10, 6))
plt.scatter(Data['Review Length'], Data['Rating text'], alpha=0.5)
plt.title('Relationship Between Review Length and Rating')
plt.xlabel('Review Length')
plt.ylabel('Rating')
plt.show()
```



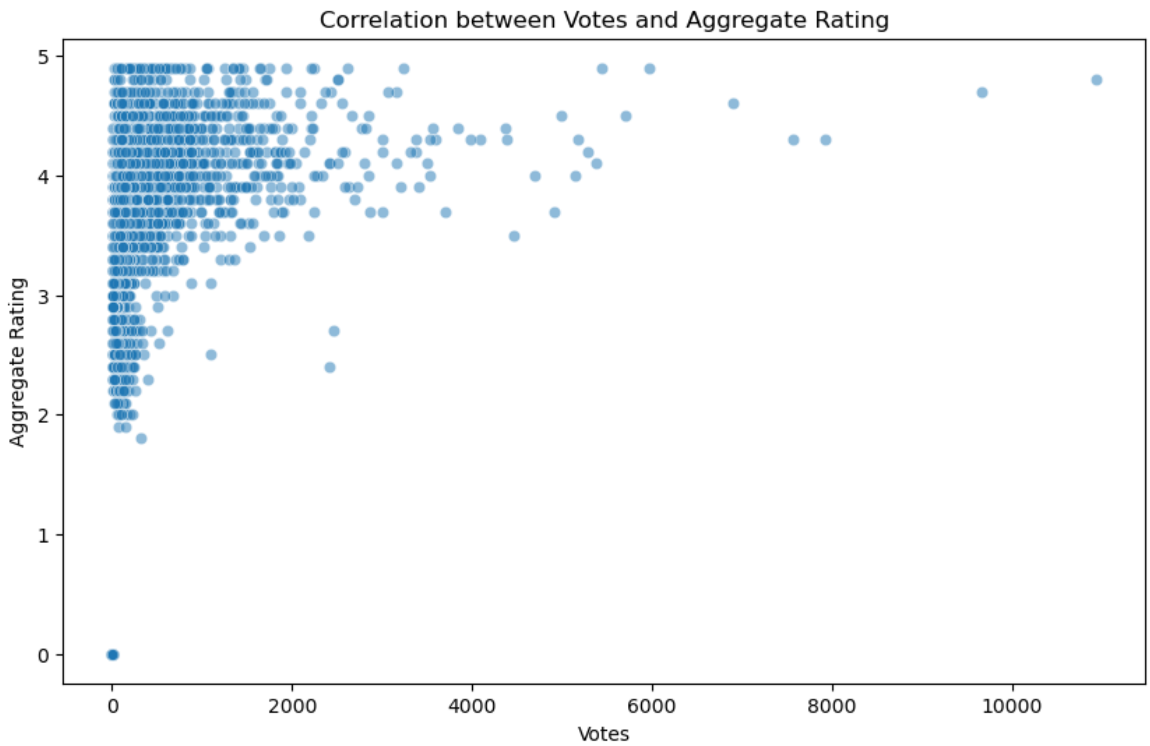
Identify the restaurants with the highest and lowest number of votes.

In [25]:
```
highest_votes =Data.loc[Data['Votes'].idxmax()]
lowest_votes = Data.loc[Data['Votes'].idxmin()]

print("Restaurant with the Highest Votes:")
print(highest_votes[['Restaurant Name', 'Votes']])

print("\nRestaurant with the Lowest Votes:")
print(lowest_votes[['Restaurant Name', 'Votes']])
```

```
Restaurant with the Highest Votes:
Restaurant Name     Toit
Votes              10934
Name: 728, dtype: object

Restaurant with the Lowest Votes:
Restaurant Name    Cantinho da Gula
Votes                             0
Name: 69, dtype: object
```

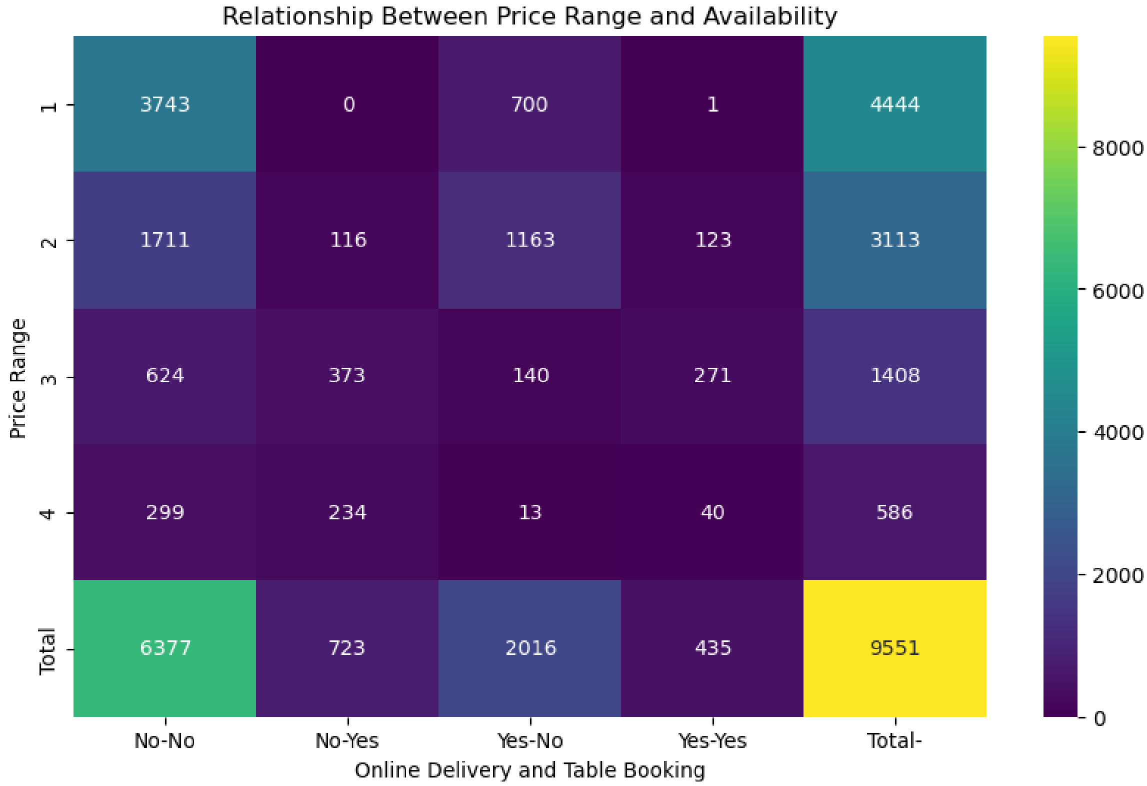
Analyze if there is a correlation between the number of votes and the rating of a restaurant.

In [26]:
```
correlation = Data['Votes'].corr(Data['Aggregate rating'])
correlation

# Create a scatter plot to visualize the relationship

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Votes', y='Aggregate rating', data=Data, alpha=0.5)
plt.title('Correlation between Votes and Aggregate Rating')
plt.xlabel('Votes')
plt.ylabel('Aggregate Rating')
plt.show()
```



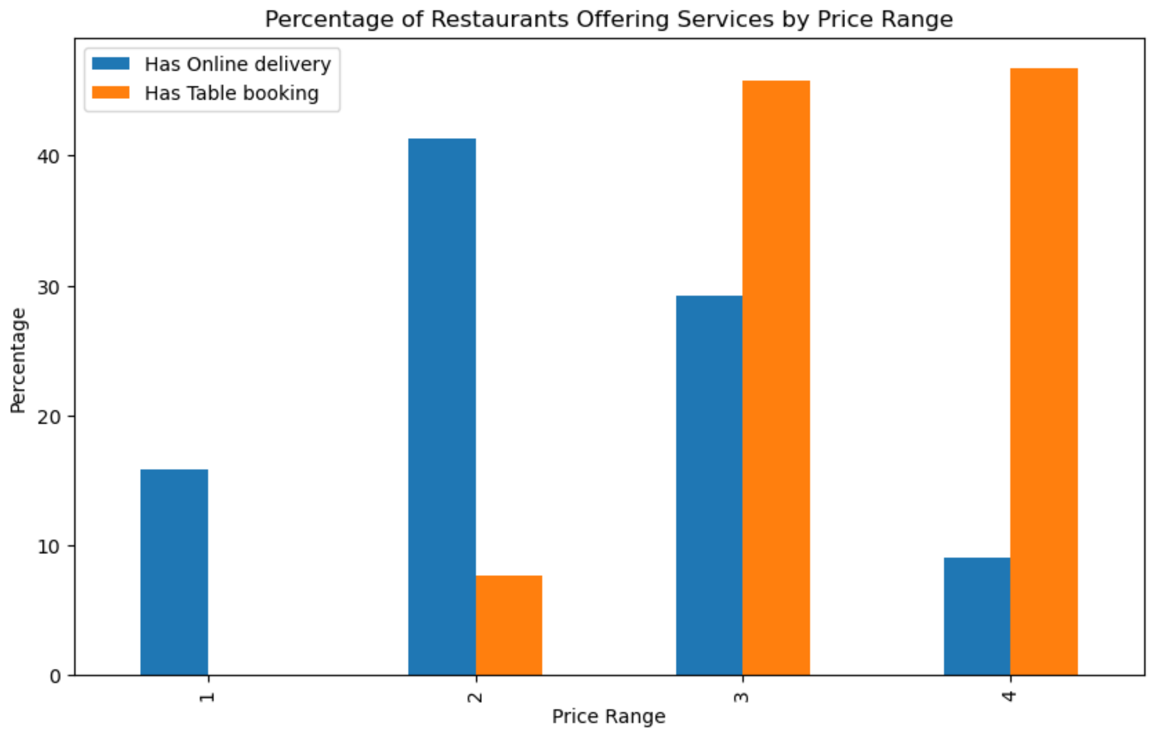
Analyze if there is a relationship between the price range and the availability of online delivery and table booking

```
In [27]: cross_tab = pd.crosstab(index=Data['Price range'], columns=[Data['Has
         cross_tab
         # Create a heatmap to visualize the relationships
         plt.figure(figsize=(10, 6))
         sns.heatmap(cross_tab, annot=True, cmap='viridis', fmt='d', cbar=True)
         plt.title('Relationship Between Price Range and Availability')
         plt.xlabel('Online Delivery and Table Booking')
         plt.ylabel('Price Range')
         plt.show()
```



Determine if higher-priced restaurants are more likely to offer these services.

In [28]:
```
# Group the data by Price range and calculate the percentage of restau
service_percentage_by_price_range = Data.groupby('Price range')[['Has 
service_percentage_by_price_range

# Plot the results
service_percentage_by_price_range.plot(kind='bar', figsize=(10, 6))
plt.title('Percentage of Restaurants Offering Services by Price Range'
plt.xlabel('Price Range')
plt.ylabel('Percentage')
plt.show()
```



In [ ]: