Importing Libraries

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
Data = pd.read_csv('C:/Users/Ankit Pandey/OneDrive/Desktop/zomato.csv', header=0, encoding='latin1')
```

In [3]:
```python
Data.head() #View the first few rows
```

Out[3]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range	Aggregate rating	Rating color	Rating text	Votes
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...	121.027535	14.565443	French, Japanese, Desserts	...	Botswana Pula(P)	Yes	No	No	No	3	4.8	Dark Green	Excellent	314
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...	121.014101	14.553708	Japanese	...	Botswana Pula(P)	Yes	No	No	No	3	4.5	Dark Green	Excellent	591
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...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)	Yes	No	No	No	4	4.4	Green	Very Good	270
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...	121.056475	14.585318	Japanese, Sushi	...	Botswana Pula(P)	No	No	No	No	4	4.9	Dark Green	Excellent	365
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...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)	Yes	No	No	No	4	4.8	Dark Green	Excellent	229

5 rows × 21 columns

In [4]:
```python
Data.info() #Get information about the data types
```

```
<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 [5]:
```python
Data.describe()  #Summary statistics for numerical columns
```

Out[5]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	156.909748
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	430.169145
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.500000	5.000000
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.200000	31.000000
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.700000	131.000000
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000

In [6]:
```python
Data.isnull().sum()  #Check for missing values
```

Out[6]:
```
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]:
```python
top_cuisines = Data['Cuisines'].value_counts().head(3)
print(top_cuisines)
```

```
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]:
```python
total_restaurants = len(Data)

percentage_per_cuisine = (top_cuisines/ total_restaurants) * 100

print(percentage_per_cuisine)
```

```
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]:
```python
city_with_highest_restaurants = Data['City'].value_counts().idxmax()

print(f"The city with the highest number of restaurants is: {city_with_highest_restaurants}")
```

```
The city with the highest number of restaurants is: New Delhi
```

Calculate the average rating for restaurants in each city.

In [13]:
```python
average_rating_per_city =Data.groupby('City')['Aggregate rating'].mean().sort_values(ascending=False)

print(average_rating_per_city)
```

```
City
Inner City          4.900000
Quezon City         4.800000
Makati City         4.650000
Pasig City          4.633333
Mandaluyong City    4.625000
                      ...   
New Delhi           2.438845
Montville           2.400000
Mc Millan           2.400000
Noida               2.036204
Faridabad           1.866932
Name: Aggregate rating, Length: 141, dtype: float64
```

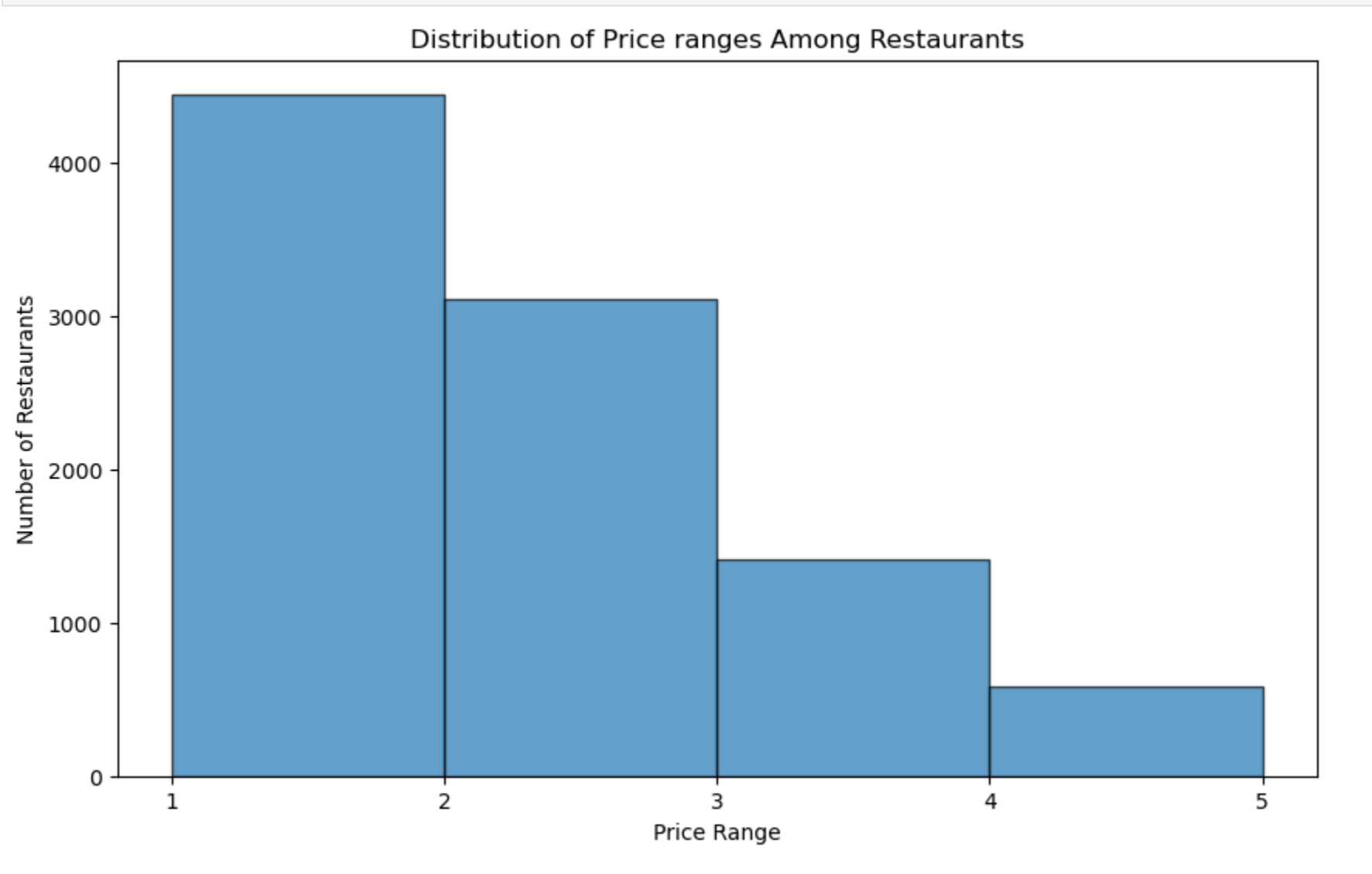
Determine the city with the highest average rating.

In [14]:
```python
city_highest_average_rating = Data.groupby('City')['Aggregate rating'].mean().idxmax()

print(f"The city with the highest average rating is: {city_highest_average_rating}")
```

```
The city with the highest average rating is: Inner City
```

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

In [15]:
```python
plt.figure(figsize=(10,6))
plt.hist(Data['Price range'], bins= [1,2,3,4,5], edgecolor ='black', alpha= 0.7)
plt.title('Distribution of Price ranges Among Restaurants')
plt.xlabel('Price Range')
plt.ylabel('Number of Restaurants')
plt.xticks([1,2,3,4,5])
plt.show()
```



Calculate the percentage of restaurants in each price range category.

In [6]:
```python
price_range_percentage = Data['Price range'].value_counts(normalize=True) *100

print(price_range_percentage)
```

```
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 [7]:
```python
online_delivery_percentage = Data['Has Online delivery'].value_counts(normalize=True) *100

print(online_delivery_percentage)
```

```
Has Online delivery
No     74.337766
Yes    25.662234
Name: proportion, dtype: float64
```

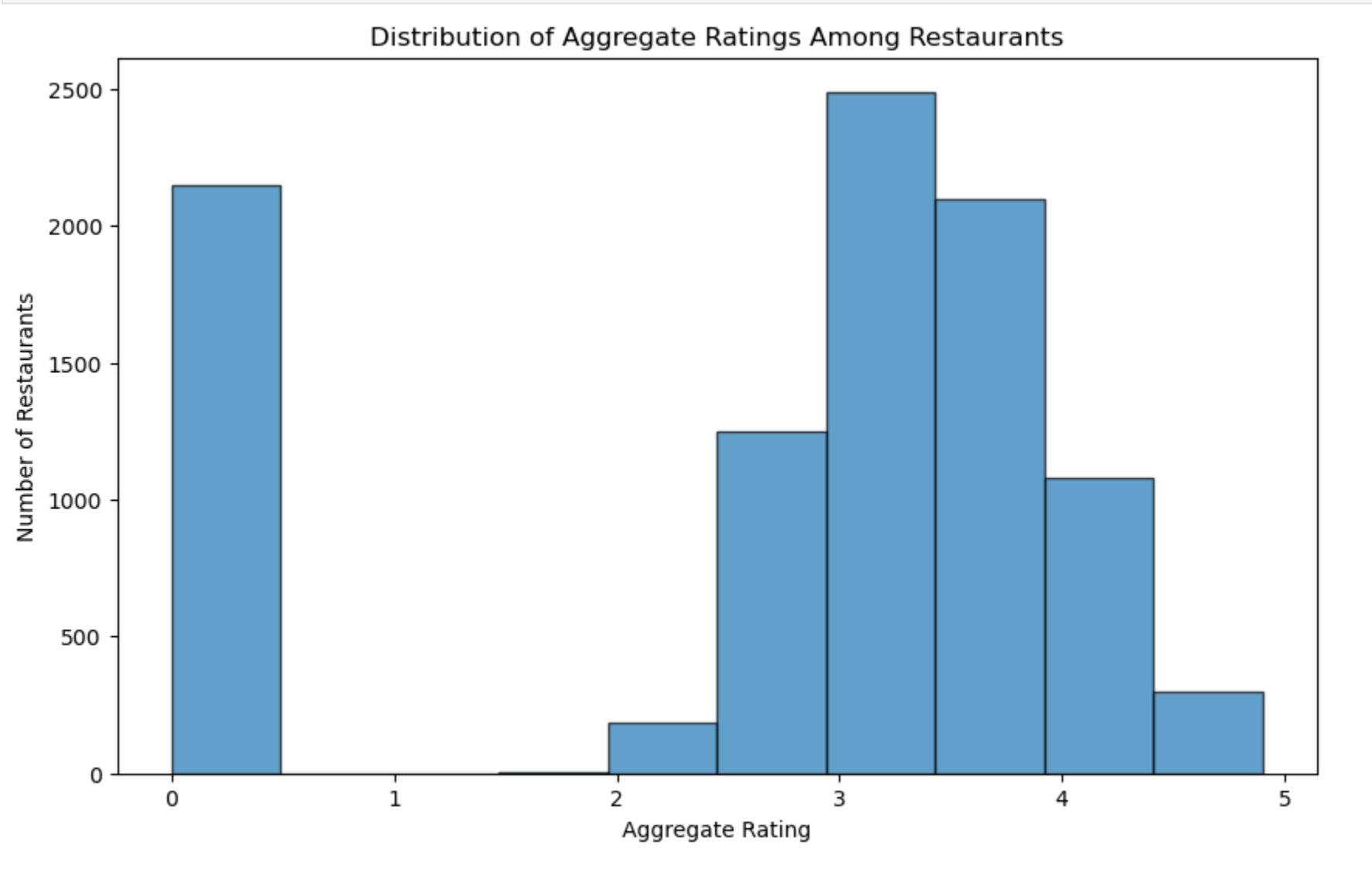
Compare the average ratings of restaurants with and without online delivery.

In [9]:
```python
Avg_ratings = Data.groupby('Has Online delivery')['Aggregate rating'].mean()

print(Avg_ratings)
```

```
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 [4]:
```python
plt.figure(figsize=(10,6))
plt.hist(Data['Aggregate rating'], bins=10, edgecolor='black', alpha=0.7)
plt.title('Distribution of Aggregate Ratings Among Restaurants')
plt.xlabel('Aggregate Rating')
plt.ylabel('Number of Restaurants')
plt.show()
```



Calculate the average number of votes received by restaurants.

In [5]:
```python
average_votes = Data['Votes'].mean()

print(f"The average number of votes received by restutants is: {average_votes:2f}")
```

```
The average number of votes received by restutants is: 156.909748
```

Identify the most common combination of cuisines in the dataset.

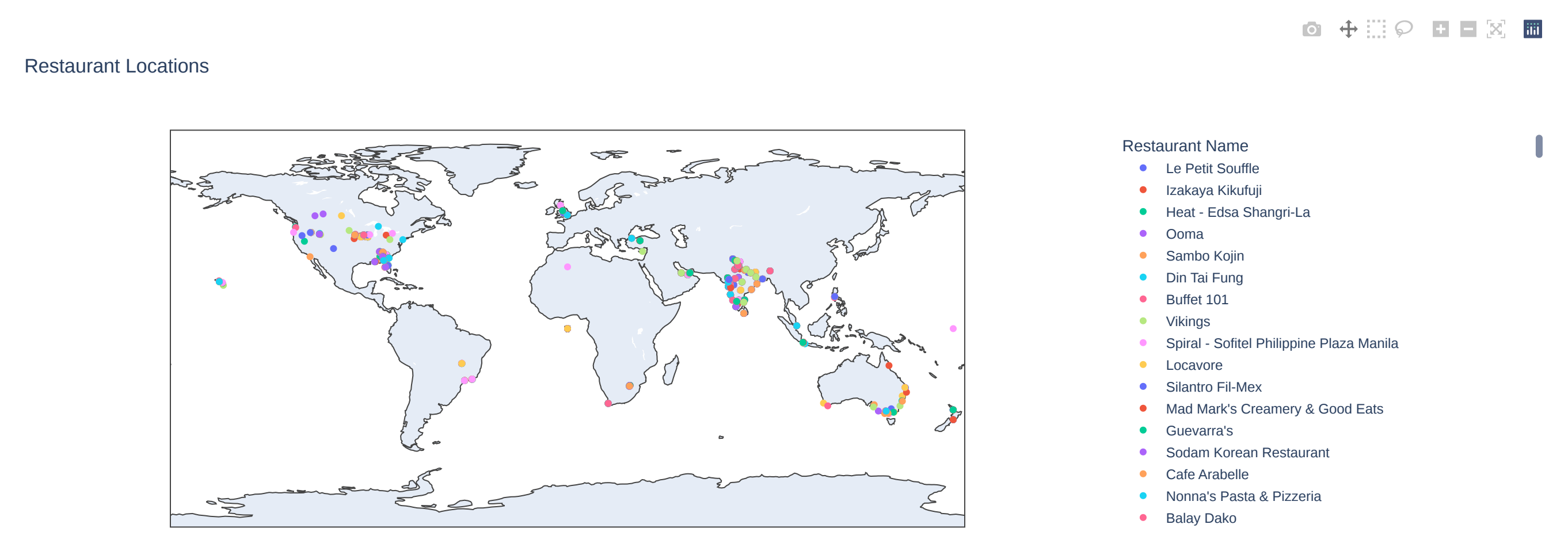
In [6]:
```python
most_common_cuisines = Data['Cuisines'].value_counts().idxmax()

print(f"The most common combination of cuisine is: {most_common_cuisines}")
```

```
The most common combination of cuisine is: North Indian
```

Plot the locations if restaurants on a map using logitude and latitude coordinates.

In [7]:
```python
import plotly.express as px  #For Geograph map
```

In [9]:
```python
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 restaurants chains present in the dataset.

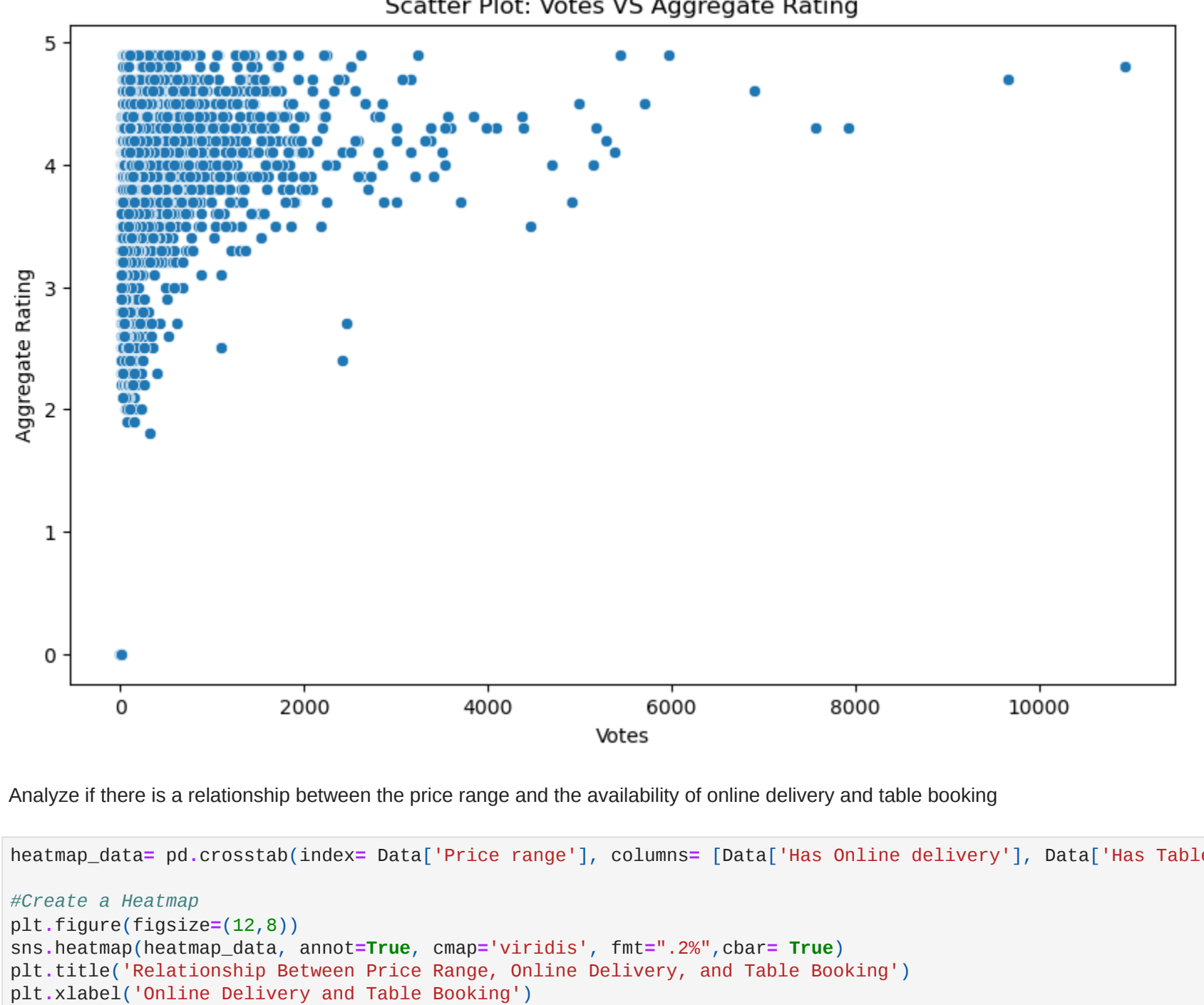
In [10]:
```python
restaurant_chains= Data[Data.duplicated(subset=['Restaurant Name'], keep=False)].sort_values(by=['Restaurant Name'])

if not restaurant_chains.empty:
    print("Potential restaurant chain found in the dataset:")
    print(restaurant_chains[['Restaurant Name', 'City','Address']])
else:
    print("No resturant chains found in the dataset.")
```

```
Potential restaurant chain found in the dataset:
        Restaurant Name       City  \
751    10 Downing Street     Bhopal   
2333   10 Downing Street     Indore   
8848  221 B Baker Street      Noida   
8498  221 B Baker Street      Noida   
8039  221 B Baker Street      Noida   
...                  ...        ...   
1965                Zizo    Gurgaon   
5927      Zooby's Kitchen  New Delhi   
8975      Zooby's Kitchen      Noida   
1560               buí©no    Gurgaon   
1870               buí©no    Gurgaon   

                                                Address  
751   Third Floor, DB City Mall, Maharana Pratap Nag...  
2333  Second Floor, Malhar Mega Mall, AB Road, Schem...  
8848                     PG 30, TOT Mall, Sector 62, Noida  
8498          21, Jalvayu Vihar Market, Sector 25, Noida  
8039  10, Brahmaputra Shoping Complex, Sector 29, Noida  
...                                                 ...  
1965                 Palm Spring Plaza, Sector 54, Gurgaon  
5927                        New Friends Colony, New Delhi  
8975  Shop 31, Amrapali Princely Estate, Sector 76, ...  
1560  2nd Floor, Near Lifestyle, MGF Metropolis Mall...  
1870  108, Vijay Vihar, Silokhera Road, Near BPTP Pa...  

[2839 rows x 3 columns]
```

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

In [11]:
```python
correlation = Data[['Votes','Aggregate rating']].corr()
print(f"Correlation between Votes and Aggregate rating:\n{correlation}")

#Create a Scatter plot
plt.figure(figsize=(10,6))
sns.scatterplot(x='Votes', y='Aggregate rating', data= Data)
plt.title('Scatter Plot: Votes VS Aggregate Rating')
plt.xlabel('Votes')
plt.ylabel('Aggregate Rating')
plt.show()
```

```
Correlation between Votes and Aggregate rating:
                     Votes  Aggregate rating
Votes             1.000000          0.313691
Aggregate rating  0.313691          1.000000
```



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

In [15]:
```python
heatmap_data= pd.crosstab(index= Data['Price range'], columns= [Data['Has Online delivery'], Data['Has Table booking']], normalize='index')

#Create a Heatmap
plt.figure(figsize=(12,8))
sns.heatmap(heatmap_data, annot=True, cmap='viridis', fmt=".2%",cbar= True)
plt.title('Relationship Between Price Range, Online Delivery, and Table Booking')
plt.xlabel('Online Delivery and Table Booking')
plt.ylabel('Price Range')
plt.show()
```