➡Importing Libraries

In [88]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

print("Libraries imported...!")
```

```
Libraries imported...!
```

In [102…
```
dataframe = pd.read_csv("C:/Users/OCS/OneDrive/Documents/Projects/Cognifyz/Dataset .csv")
print("Dataset loaded...:)")
```

```
Dataset loaded...:)
```

➡To check total number of rows and columns

In [3]:
```
dataframe.shape
```

Out[3]:
```
(9551, 21)
```

➡To show the top 5 rows of our data

In [4]:
```
dataframe.head(5)
```

Out[4]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Curr
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	...	Bots Pu
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	...	Bots Pu
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	...	Bots Pu
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	...	Bots Pu
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	...	Bots Pu

5 rows × 21 columns

To show the bottom 5 rows of our data

In [5]:
```
dataframe.tail(5)
```

Out[5]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cu
9546	5915730	Namlı Gurme	208	��stanbul	Kemanke�� Karamustafa Pa��a Mahallesi, Rıhtım ...	Karak�_y	Karak�_y, ��stanbul	28.977392	41.022793	-
9547	5908749	Ceviz A��acı	208	��stanbul	Ko��uyolu Mahallesi, Muhittin ��st�_nda�� Cadd...	Ko��uyolu	Ko��uyolu, ��stanbul	29.041297	41.009847	C Pat
9548	5915807	Huqqa	208	��stanbul	Kuru�_e��me Mahallesi, Muallim Naci Caddesi, N...	Kuru�_e��me	Kuru�_e��me, ��stanbul	29.034640	41.055817	(
9549	5916112	A���k Kahve	208	��stanbul	Kuru�_e��me Mahallesi, Muallim Naci Caddesi, N...	Kuru�_e��me	Kuru�_e��me, ��stanbul	29.036019	41.057979	Res
9550	5927402	Walter's Coffee Roastery	208	��stanbul	Cafea��a Mahallesi, Bademaltı Sokak, No 21/B, ...	Moda	Moda, ��stanbul	29.026016	40.984776	

5 rows × 21 columns

➡DATA CLEANING

a. Showing all the rows,columns,data_type to see that there is no null values present in our dataset

In [6]:
```
dataframe.info()
```

```
<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
```

b. This method returns description of the data in the DataFrame (i.e. count, mean, std, etc)

In [7]:
```
dataframe.describe()
```

Out[7]:

	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

This method will print all the columns name.

In [8]:
```python
for i in dataframe.columns:
    print(i)
```

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

---LEVEL - 1---

->TASK_1 - Top Cuisines

In [10]:
```python
cuisine_counts = dataframe["Cuisines"].value_counts()

cuisine_counts_df = cuisine_counts.reset_index()
cuisine_counts_df.columns = ['Cuisines', 'Counts']
print(cuisine_counts_df)
```

```
                                Cuisines  Counts
0                           North Indian     936
1                  North Indian, Chinese     511
2                                Chinese     354
3                              Fast Food     354
4                  North Indian, Mughlai     334
...                                  ...     ...
1820     World Cuisine, Patisserie, Cafe       1
1821                     Burger, Izgara       1
1822                   Desserts, B�_rek       1
1823  Restaurant Cafe, Turkish, Desserts       1
1824            Restaurant Cafe, Desserts       1

[1825 rows x 2 columns]
```

In [12]:
```python
# a. To print "3 top cuisines" of the Restaurant

top_3_cuisines = cuisine_counts.head(3)

#creating a dataframe "top_3_cuisines_df" to print the Count as the column name

top_3_cuisines_df = top_3_cuisines.reset_index()
top_3_cuisines_df.columns = ['Cuisines' , 'Count']
print(top_3_cuisines_df)
```

```
               Cuisines  Count
0          North Indian    936
1  North Indian, Chinese    511
2               Chinese    354
```

In [13]:
```
# b. Calculate the percentage of restaurants that serve each of the top cuisines.

total_restaurants = dataframe.shape[0]
top_3_cuisines_percentage = (top_3_cuisines / total_restaurants) * 100

#To print the percentage as the column name

top_3_cuisines_df = top_3_cuisines_percentage.reset_index()
top_3_cuisines_df.columns = ['Cuisines', 'Percentage']
print(top_3_cuisines_df)
```

```
               Cuisines  Percentage
0          North Indian    9.800021
1  North Indian, Chinese    5.350225
2               Chinese    3.706418
```

->TASK_2 - City Analysis

In [14]:
```
# a. To print the city with the highest number of restaurants in the dataset.

# Counts the occurences of Cities

city_counts = dataframe['City'].value_counts().reset_index(name='Count')

# Creating a dataframe "top_city"

top_city = city_counts.sort_values(by = 'Count' , ascending=False).iloc[0]

print(f"The city with the highest number of restaurants is '{top_city['City']}' with '{top_city['Count']}' resta
```

```
The city with the highest number of restaurants is 'New Delhi' with '5473' restaurants.
```

In [15]:
```
# b. Calculate the "average rating" for restaurants in each city

average_ratings = dataframe.groupby('City')['Aggregate rating'].mean().reset_index().round(1)

print(average_ratings)
```

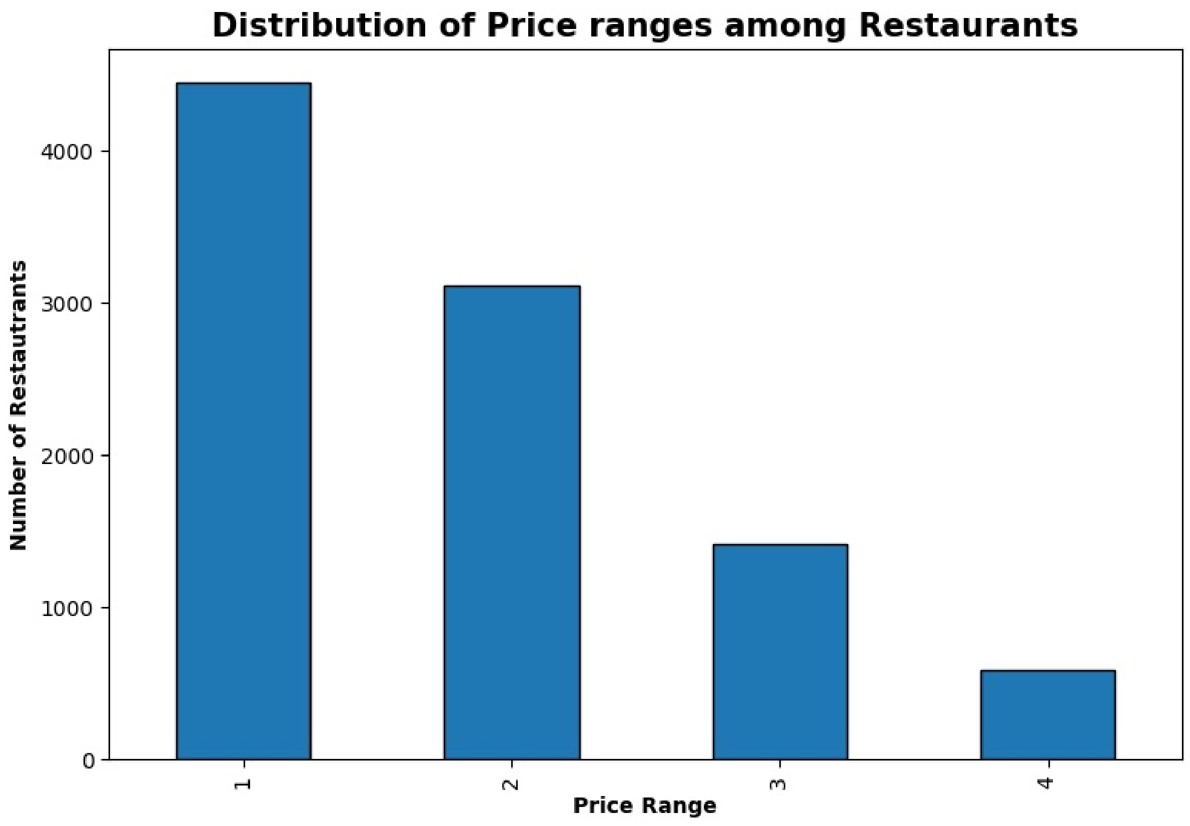
```
               City  Aggregate rating
0         Abu Dhabi               4.3
1              Agra               4.0
2         Ahmedabad               4.2
3            Albany               3.6
4         Allahabad               3.4
..              ...               ...
136         Weirton               3.9
137 Wellington City               4.2
138  Winchester Bay               3.2
139         Yorkton               3.3
140       ��stanbul               4.3

[141 rows x 2 columns]
```

In [16]:
```
# c. Determine the city with the "highest average rating"

top_city = average_ratings.sort_values(by = 'Aggregate rating', ascending=False).iloc[0]

#rounding off it to two digits

print(f"The city with the highest average rating is '{top_city['City']}' with an average rating of {top_city['A
```

```
The city with the highest average rating is 'Inner City' with an average rating of 4.9.
```

->TASK_3 - Price Range Distribution

In [17]:
```
# a. Create a "bar chart" to visualize the distribution of price ranges among the restaurants.

plt.figure(figsize=(9, 6))
price_counts = dataframe['Price range'].value_counts()
price_counts.plot(kind = 'bar', edgecolor = 'black')
plt.title("Distribution of Price ranges among Restaurants", fontsize = 15, fontweight = 'bold')
plt.xlabel("Price Range", fontweight = 'bold')
plt.ylabel("Number of Restautrants", fontweight = 'bold')
plt.show()
```



In [18]:
```python
# b. Calculate the percentage of restaurants in each price range category.

# "price_counts_percentage" will calculate the %
price_counts_percentage = dataframe['Price range'].value_counts(normalize = True).round(3) * 100

# Creating a new dataframe "price_counts_percentage_df" to show the results in Percentage column

price_counts_percentage_df = price_counts_percentage.reset_index()
price_counts_percentage_df.columns = ['Price range' , 'Percentage']

# To show % symbol in my column

price_counts_percentage_df['Percentage'] = price_counts_percentage_df['Percentage'].astype(str) + '%'

print(price_counts_percentage_df)
```

```
   Price range Percentage
0            1      46.5%
1            2      32.6%
2            3      14.7%
3            4       6.1%
```

->TASK_4 - Online Delivery

In [19]:
```python
# a. Determine the percentage of restaurants that offer online delivery.

# Converting the 'Has Online delivery' column values into binary values
dataframe['Has Online delivery'] = dataframe['Has Online delivery'].map({'Yes': 1, 'No': 0, 'YES': 1, 'NO': 0,

total_restaurants = dataframe.shape[0]
online_delivery_count = dataframe['Has Online delivery'].sum()

#Calculate Percentage
online_delivery_percentage = (online_delivery_count / total_restaurants) * 100

print(f"Percentage of Restaurants that offer online delivery: {online_delivery_percentage:.2f}%")
```

```
Percentage of Restaurants that offer online delivery: 25.66%
```

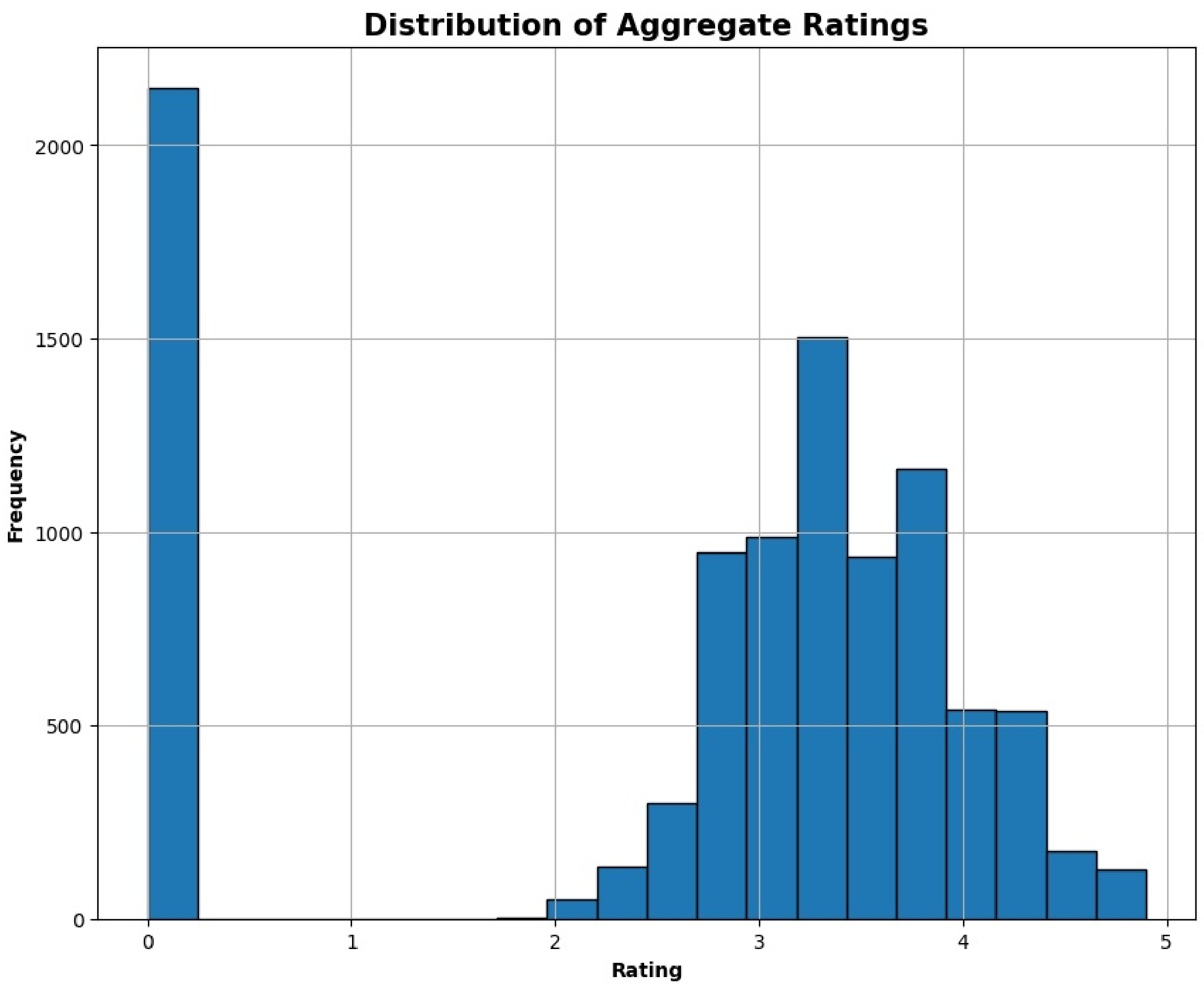
In [20]:
```python
# b. Compare the average ratings of restaurants with and without online delivery.

average_ratings = dataframe.groupby('Has Online delivery')['Aggregate rating'].mean().reset_index().round(2)
average_ratings.columns = ['Online delivery', 'Average rating']
print(average_ratings)
```

```
   Online delivery  Average rating
0                0            2.47
1                1            3.25
```

---LEVEL - 2 ---

-> TASK_1 - Restaurants Ratings

```
In [39]: # a. Analyze the distribution of aggregate ratings and determine the most common rating range.

         # Plotting a "Histogram"

         plt.figure(figsize=(10, 8))
         plt.hist(dataframe['Aggregate rating'], bins= 20, edgecolor='black')
         plt.title('Distribution of Aggregate Ratings', fontsize = 15, fontweight = 'bold')
         plt.xlabel('Rating', fontsize = 10, fontweight = 'bold')
         plt.ylabel('Frequency',  fontsize = 10, fontweight = 'bold')
         plt.grid(True)
         plt.show()
```



```
In [38]: # b. Calculate the average number of votes received by restaurants

         average_vote_counts = dataframe['Votes'].mean()
         print(f"The average number of votes received by restaurants is: {average_vote_counts:.2f}")
```

```
The average number of votes received by restaurants is: 156.91
```

-> TASK_2 - Cuisine Combination

```
In [48]: # a. Identify the most common combinations of cuisines in the dataset.

         # Split the cuisines by ',' and count combinations
         dataframe['Cuisines'] = dataframe['Cuisines'].dropna().str.split(', ').apply(lambda x:', '.join(sorted(x)))
         combination_counts = dataframe['Cuisines'].value_counts().reset_index()
         combination_counts.columns = ['Cuisine Combination', 'Count']
         most_common_combinations_df = combination_counts.head(10)

         print(most_common_combinations_df)
```

```
              Cuisine Combination  Count
0                    North Indian    936
1           Chinese, North Indian    616
2           Mughlai, North Indian    394
3                       Fast Food    354
4                         Chinese    354
5  Chinese, Mughlai, North Indian    306
6                            Cafe    299
7                          Bakery    218
8                Bakery, Desserts    181
9              Chinese, Fast Food    159
```

In [53]:
```
# b. Determine if certain cuisine combinations tend to have higher ratings.

dataframe['Cuisines'] = dataframe['Cuisines'].dropna().str.split(', ').apply(lambda x: ', '.join(sorted(x)))
cuisine_ratings = dataframe.groupby('Cuisines')['Aggregate rating'].mean().reset_index()
cuisine_ratings_sorted = cuisine_ratings.sort_values(by='Aggregate rating', ascending=False)
print(cuisine_ratings_sorted.head(10))
```

```
                              Cuisines  Aggregate rating
1342                     World Cuisine               4.9
31               American, BBQ, Sandwich             4.9
132   American, Healthy Food, Mexican                4.9
94              American, Coffee and Tea             4.9
84          American, Caribbean, Seafood             4.9
908              Contemporary, European              4.9
1033                   European, German              4.9
158              American, Sandwich, Tea             4.9
412               Bar Food, Burger, Steak            4.9
317            BBQ, Breakfast, Southern              4.9
```

-> TASK_3 - Restaurant Chains

In [76]:
```
# a. Identify if there are any restaurant chains present in the dataset

restaurant_counts = dataframe['Restaurant Name'].value_counts()
chains = restaurant_counts[restaurant_counts > 1]
chains.columns = ['Restaurant Name', 'Count']

print(chains)
```

```
Restaurant Name
Cafe Coffee Day     83
Domino's Pizza      79
Subway              63
Green Chick Chop    51
McDonald's          48
                    ..
San Carlo            2
Gymkhana             2
Dishoom              2
Timboo Cafe          2
D�_vero��lu          2
Name: count, Length: 734, dtype: int64
```

In [82]:
```
# b. Analyze the ratings and popularity of different restaurant chains

restaurant_counts = dataframe['Restaurant Name'].value_counts()
chains = restaurant_counts[restaurant_counts > 1].index
chain_df = dataframe[dataframe['Restaurant Name'].isin(chains)]
chain_analysis = chain_df.groupby('Restaurant Name').agg(avg_rating=('Aggregate rating', 'mean'),total_votes=('

print("Ratings and Popularity of Restaurant Chains: \n")
print(chain_analysis)
```

```
Ratings and Popularity of Restaurant Chains:

          Restaurant Name  avg_rating  total_votes
0       10 Downing Street    4.000000          670
1      221 B Baker Street    3.366667          215
2      34 Parkstreet Lane    3.050000           31
3    34, Chowringhee Lane    2.791667          777
4          4700BC Popcorn    3.500000          176
..                    ...         ...          ...
729                 Zaika    2.850000          245
730      Zaika Kathi Rolls    1.500000           16
731                  Zizo    3.866667         1371
732        Zooby's Kitchen    3.150000           52
733               bu��no    3.750000          117

[734 rows x 3 columns]
```

---LEVEL-3 ---

->TASK_1 - Votes Analysis

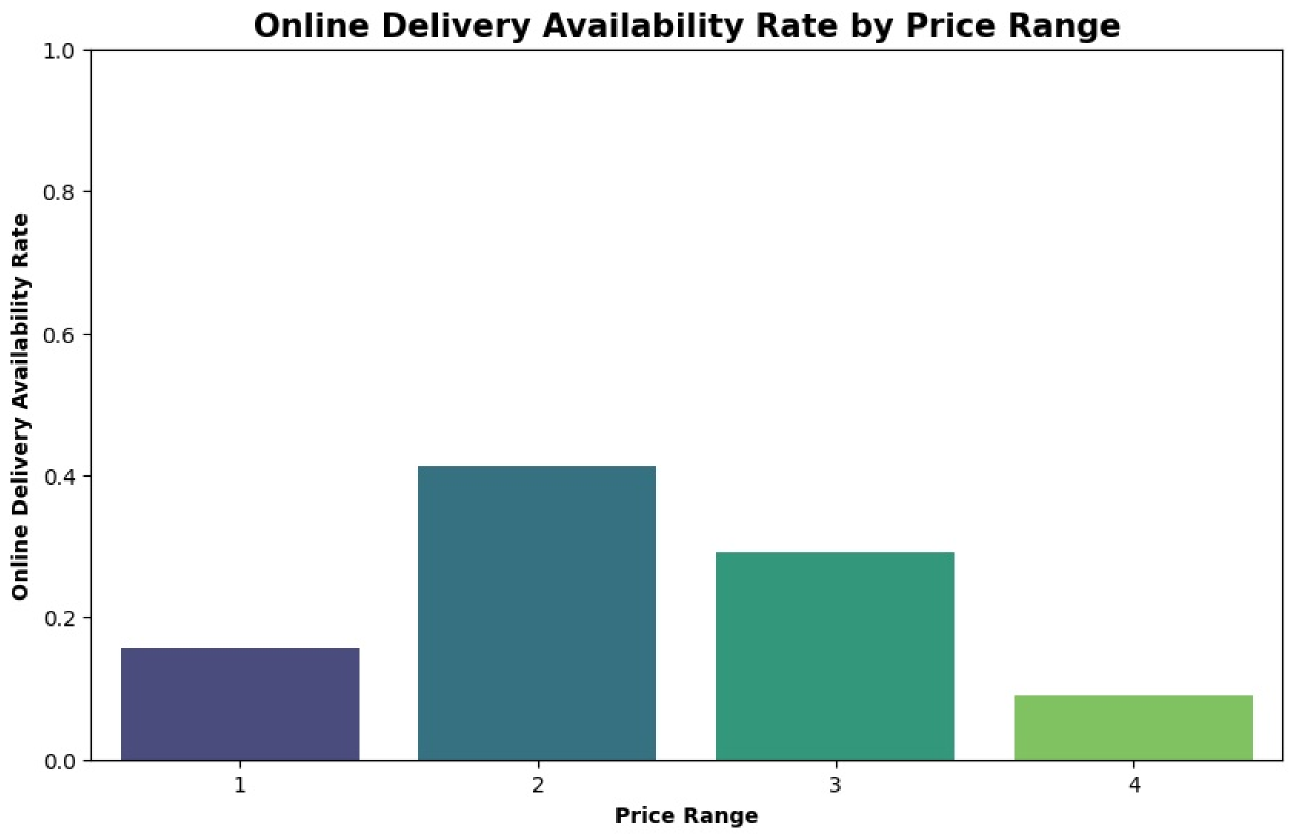
In [86]:
```
# Identify the restaurants with the highest and lowest number of votes.

highest_votes = dataframe[dataframe['Votes'] == dataframe['Votes'].max()]
lowest_votes = dataframe[dataframe['Votes'] == dataframe['Votes'].min()]

print("Restaurant(s) with the Highest Number of Votes:\n")
print(highest_votes[['Restaurant Name', 'Votes']])

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

```
Restaurant(s) with the Highest Number of Votes:

    Restaurant Name  Votes
728            Toit  10934

Restaurant(s) with the Lowest Number of Votes:

                 Restaurant Name  Votes
69              Cantinho da Gula      0
874                The Chaiwalas      0
879            Fusion Food Corner      0
880                Punjabi Rasoi      0
887                Baskin Robbin      0
...                          ...    ...
9044               6 Packs Momos      0
9098                   Cafe' Wow      0
9099  Chef's Basket Pop Up Caf��      0
9103             The Hangout-Deli      0
9111                    Platters      0

[1094 rows x 2 columns]
```

->TASK_2 - Price Range vs. Online Delivery and Table Booking

In [103…
```
# Analyze if there is a relationship between the price range and the availability of online delivery and table

dataframe['Has Online delivery'] = dataframe['Has Online delivery'].apply(lambda x: 1 if x == 'Yes' else 0)
dataframe['Has Table booking'] = dataframe['Has Table booking'].apply(lambda x: 1 if x == 'Yes' else 0)

price_range_summary = dataframe.groupby('Price range').agg(online_delivery_rate=('Has Online delivery', 'mean')

print("Summary Statistics by Price Range:")
print(price_range_summary)

# To visualize the results we have to plot the bar plot for Online Delivery

plt.figure(figsize=(10, 6))
sns.barplot(x='Price range', y='online_delivery_rate', data=price_range_summary, palette='viridis')
plt.xlabel('Price Range', fontsize = 10, fontweight = 'bold')
plt.ylabel('Online Delivery Availability Rate', fontsize = 10, fontweight = 'bold')
plt.title('Online Delivery Availability Rate by Price Range', fontsize = 15, fontweight = 'bold')
plt.ylim(0, 1)
plt.show()

# To visualize the results we have to plot the bar plot for Online Delivery

plt.figure(figsize=(10, 6))
sns.barplot(x='Price range', y='table_booking_rate', data=price_range_summary, palette='viridis')
plt.xlabel('Price Range', fontsize = 10, fontweight = 'bold')
plt.ylabel('Table Booking Availability Rate', fontsize = 10, fontweight = 'bold')
plt.title('Table Booking Availability Rate by Price Range', fontsize = 15, fontweight = 'bold')
plt.ylim(0, 1)
plt.show()
```

```
Summary Statistics by Price Range:
   Price range  online_delivery_rate  table_booking_rate
0            1              0.157741            0.000225
1            2              0.413106            0.076775
2            3              0.291903            0.457386
3            4              0.090444            0.467577
```


Online Delivery Availability Rate by Price Range
Online Delivery Availability Rate
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
Price Range

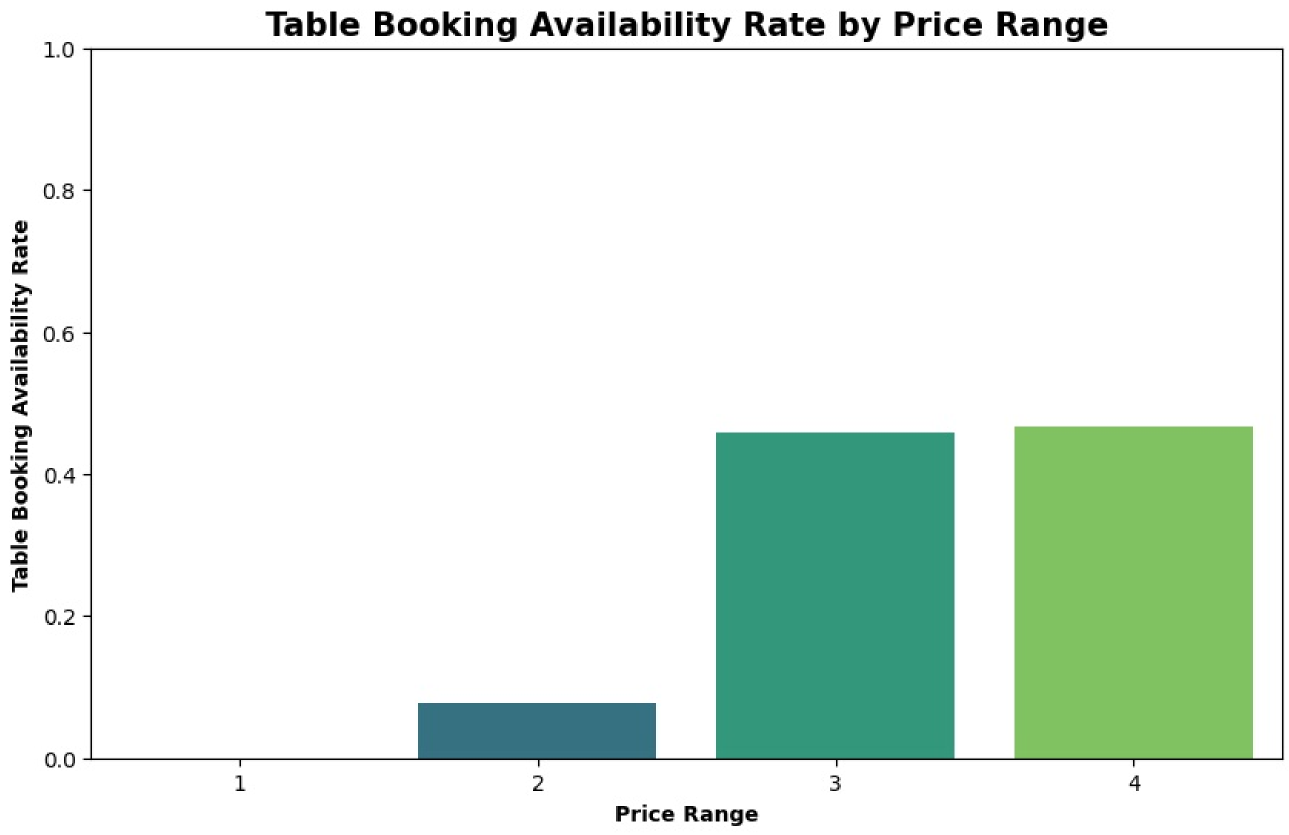


Table Booking Availability Rate by Price Range
Table Booking Availability Rate
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
Price Range