COGNIFYZ TECHNOLOGIES INTERNSHIP PROGRAM

NAME: PRECIOUS ONYEDEKE

REF: CTI/A1/C54621

Dataset Overview

Total Entries: 9,551 rows

Total Columns: 21 columns

Key Columns in the Dataset

Restaurant ID: A unique identifier for each restaurant.

Restaurant Name: The name of the restaurant.

Country Code: Numerical code representing the country where the restaurant is located.

City: The city where the restaurant is located.

Address: The full address of the restaurant.

Locality: The specific locality within the city.

Locality Verbose: A more descriptive locality name.

Longitude: The geographical longitude of the restaurant.

Latitude: The geographical latitude of the restaurant.

Cuisines: The type(s) of cuisine offered by the restaurant.

Average Cost for two: The average cost for two people dining at the restaurant.

Currency: The currency in which the restaurant charges.

Has Table booking: Indicates whether the restaurant offers table booking (Yes/No).

Has Online delivery: Indicates whether the restaurant offers online delivery (Yes/No).

Is delivering now: Indicates whether the restaurant is currently delivering (Yes/No).

Switch to order menu: This column contains only one unique value (No), so it may not be useful for analysis.

Price range: A numerical value representing the price range (from 1 to 4).

Aggregate rating: The overall rating of the restaurant.

Rating color: A color code associated with the rating.

Rating text: Descriptive text associated with the rating (e.g., "Good", "Average").

Votes: The number of votes the restaurant has received.

In [1]:
```
#importing csv file
import pandas as pd
df = pd.read_csv("Resturant dataset.csv")
df
```

Out[1]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Lo
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
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
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
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
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
...	...	...	...	...	...	...	...	
9546	5915730	Namlı Gurme	208	stanbul	Kemanke Karamustafa Paa Mahallesi, Rıhtım Cadd...	Karak_y	Karak_y, stanbul	28
9547	5908749	Ceviz Aacı	208	stanbul	Kouyolu Mahallesi, Muhittin st_nda Caddesi, No...	Kouyolu	Kouyolu, stanbul	29
9548	5915807	Huqqa	208	stanbul	Kuru_eme Mahallesi, Muallim Naci Caddesi, No 5...	Kuru_eme	Kuru_eme, stanbul	29
9549	5916112	Ak Kahve	208	stanbul	Kuru_eme Mahallesi, Muallim Naci Caddesi, No 6...	Kuru_eme	Kuru_eme, stanbul	29

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Lo
9550	5927402	Walter's Coffee Roastery	208	stanbul	Cafeaa Mahallesi, Bademalt Sokak, No 21/B, Ka...	Moda	Moda, stanbul	29

9551 rows × 21 columns

In [2]:
```
df.describe()
```

Out[2]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggr
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.0
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.6
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.5
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.0
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.5
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.2
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.7
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.9

In [3]:
```
df.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
```

In [4]:
```
#data cleaning
df= df.dropna()
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['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'],
      dtype='object')
```

Level 1

Task 1: Top Cuisines

Determine the top three most common cuisines

In [6]:
```
top_cuisines = df.groupby(['Cuisines'])['Cuisines'].count()
top_cuisines=top_cuisines.sort_values(ascending=False)
```

In [7]:
```
top_cuisines.to_frame()
```

Out[7]:

Cuisines	Cuisines
North Indian	2992
Chinese	855
Fast Food	672
Bakery	621
Cafe	617
...	...
Pub Food	1
Patisserie	1
Indonesian	1
Peruvian	1
Irish	1

119 rows × 1 columns

the top three cuisines are North Indian, Chinese, Fast Food

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

In [8]:
```
def percentage (n):
    new=(n/9551)*100
    new = round(new,1)
    print(f"the percentage is: {new}%")
```

In [9]:
```
percentage(2992)
```

```
the percentage is: 31.3%
```

In [10]:
```
percentage(855)
```

```
the percentage is: 9.0%
```

In [11]:
```
percentage(672)
```

```
the percentage is: 7.0%
```

Task 2: City Analysis

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

In [12]:
```
city=df.groupby(['City'])['Restaurant ID'].count()
city=city.sort_values(ascending=False)
```

In [13]:
```
city
```

Out[13]:
```
City
New Delhi    5473
Gurgaon      1118
Noida        1080
Faridabad     251
Ghaziabad      25
             ... 
Randburg        1
Macedon         1
Lorn            1
Lincoln         1
Forrest         1
Name: Restaurant ID, Length: 140, dtype: int64
```

Calculate the average rating for restaurants in each city.

In [14]:
```
rate=df.groupby(['City'])['Restaurant Name', 'Aggregate rating'].mean()
rate=rate.sort_values(by='Aggregate rating',ascending=False)
```

```
C:\Users\PRECIOUS ONYEDEKE\AppData\Local\Temp\ipykernel_8956\4093208627.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple o
f keys) will be deprecated, use a list instead.
  rate=df.groupby(['City'])['Restaurant Name', 'Aggregate rating'].mean()
```

In [15]:
```
rate
```

Out[15]:

	Aggregate rating
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

140 rows × 1 columns

Determine the city with the highest average rating.

The city with the highest average rating is inner city

Task 3: Price Range Distribution

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

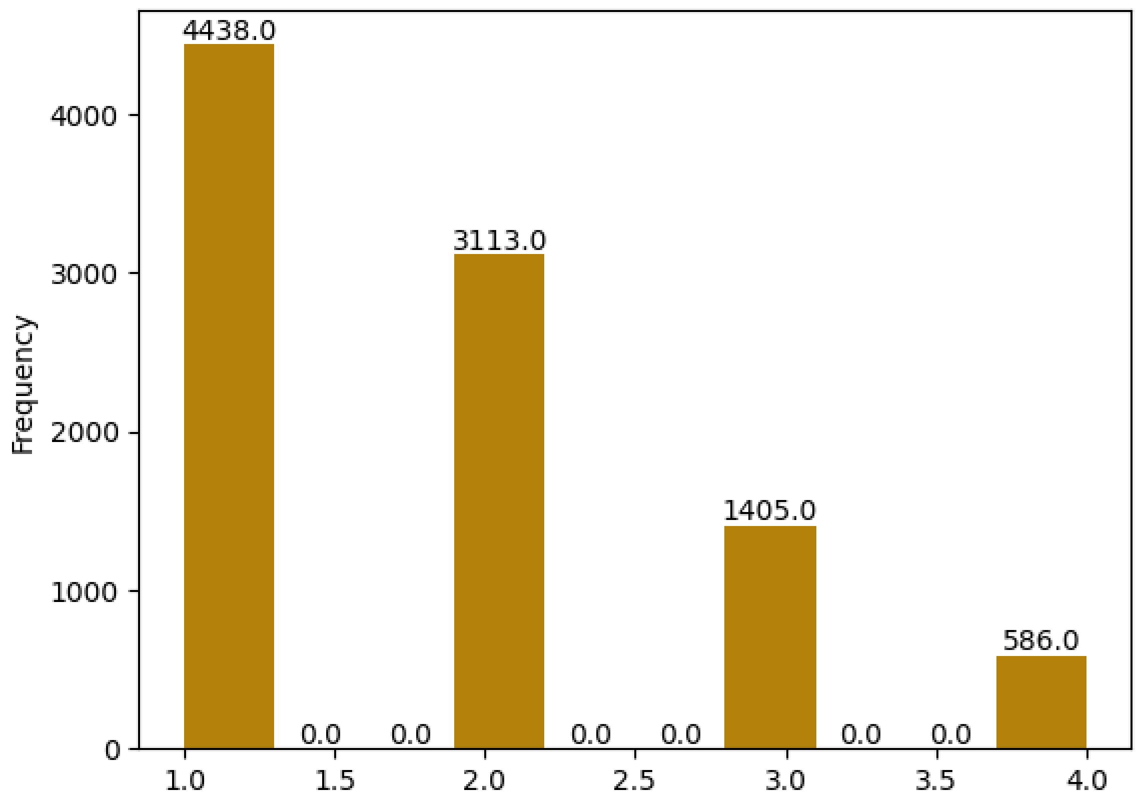
In [16]:
```
price=df.groupby(['Restaurant ID'])['Price range'].mean()
price.unique()
```

Out[16]:
```
array([3., 2., 4., 1.])
```

In [17]:
```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import numpy as np
```

```
C:\Users\PRECIOUS ONYEDEKE\AppData\Roaming\Python\Python39\site-packages\matp
lotlib\projections\__init__.py:63: UserWarning: Unable to import Axes3D. This
may be due to multiple versions of Matplotlib being installed (e.g. as a syst
em package and as a pip package). As a result, the 3D projection is not avail
able.
  warnings.warn("Unable to import Axes3D. This may be due to multiple version
s of "
```

In [18]:
```
ax=price.plot(kind='hist',color ='darkgoldenrod')
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_he
                ha='center', va='center', xytext=(0, 5), textcoords='offset po

plt.show()
```



Calculate the percentage of restaurants in each price range category.

In [19]:
```
#price range = 1
percentage(4438)
```

the percentage is: 46.5%

In [20]:
```
#price range = 2
percentage(3113)
```

the percentage is: 32.6%

In [21]:
```
#price range = 3
percentage(1405)
```

the percentage is: 14.7%

In [22]:
```
#price range = 4
percentage(586)
```

the percentage is: 6.1%

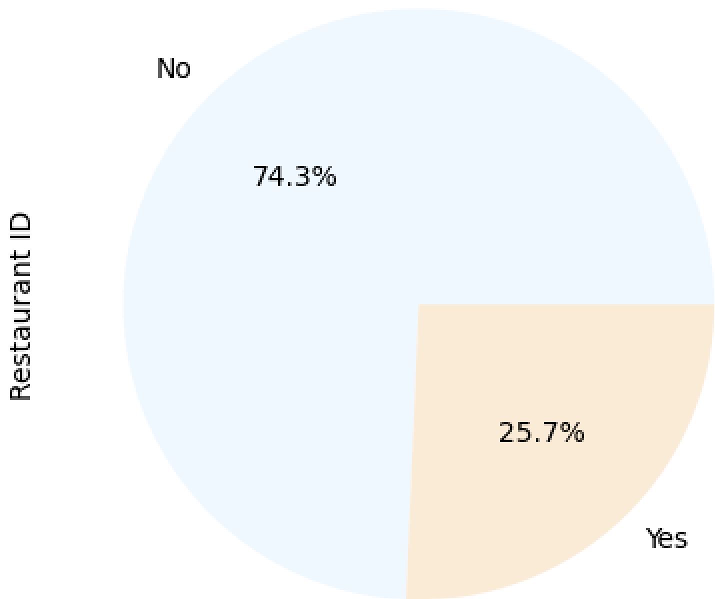
Task 4: Online Delivery

Determine the percentage of restaurants that offer online delivery.

In [23]:
```
online=df.groupby(['Has Online delivery'])['Restaurant ID'].count()
online
```

Out[23]:
```
Has Online delivery
No     7091
Yes    2451
Name: Restaurant ID, dtype: int64
```

In [24]:
```
online.plot(kind='pie', colors = mcolors.CSS4_COLORS, autopct='%1.1f%%')
plt.show()
```

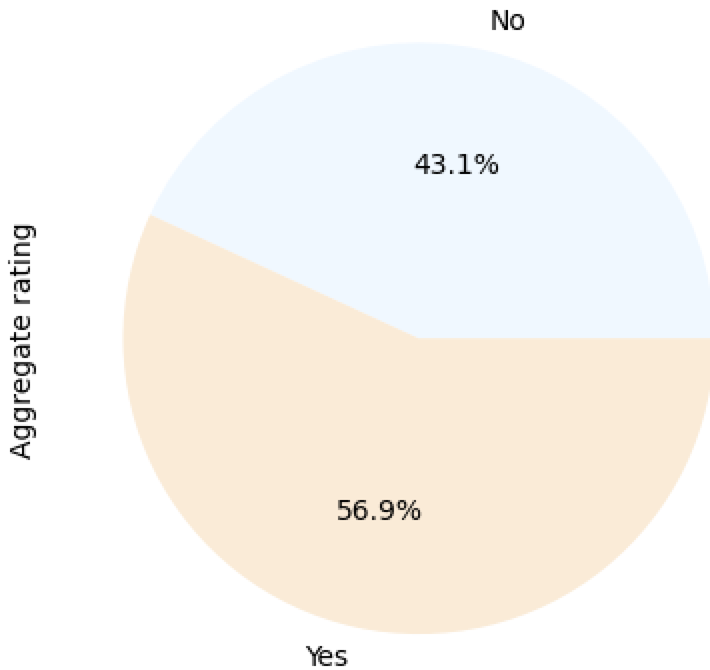


Compare the average ratings of restaurants with and without online delivery.

In [25]:
```
avg_rate=df.groupby(['Has Online delivery'])['Aggregate rating'].mean()
avg_rate
```

Out[25]:
```
Has Online delivery
No     2.463517
Yes    3.248837
Name: Aggregate rating, dtype: float64
```

In [26]:
```
avg_rate.plot(kind='pie', colors = mcolors.CSS4_COLORS, autopct='%1.1f%%')
plt.show()
```



Level 2

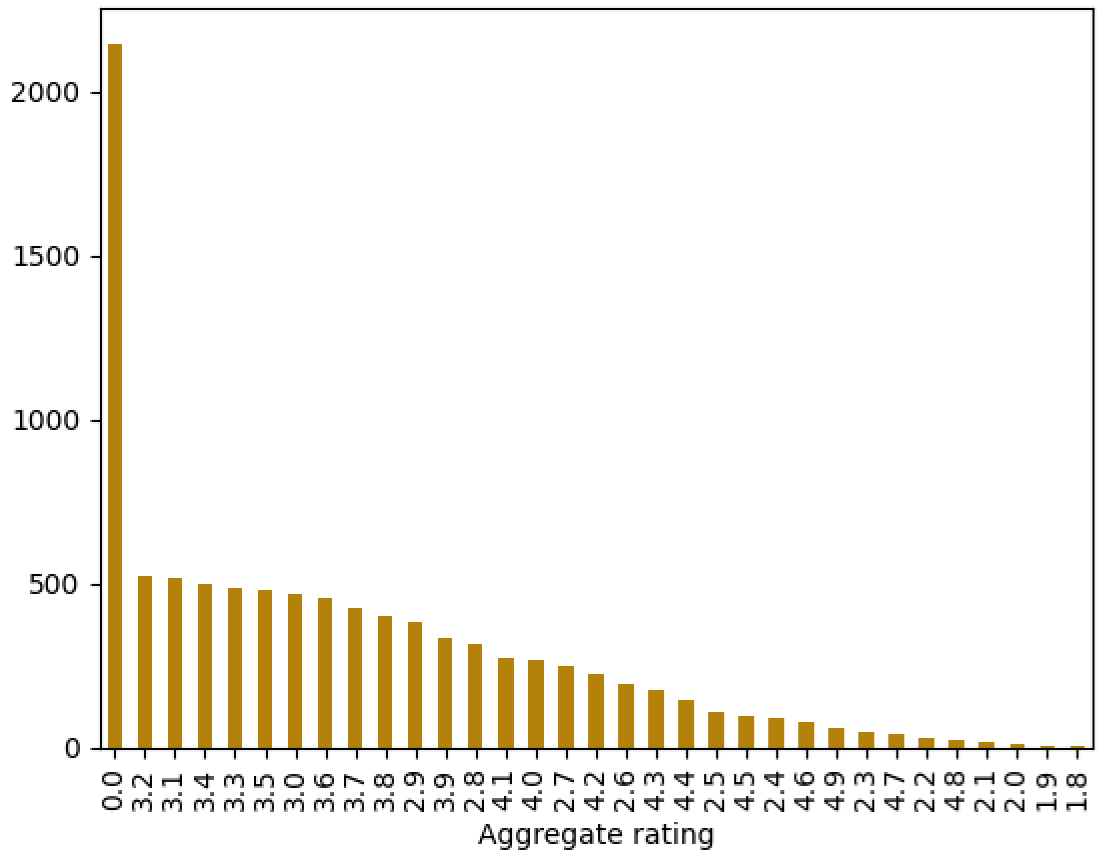
Task 1: Restaurant Ratings

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

In [27]:
```
agg_rate=df.groupby(['Aggregate rating'])['Aggregate rating'].count()
agg_rate=agg_rate.sort_values(ascending=False)
agg_rate
```

Out[27]:
```
Aggregate rating
0.0    2148
3.2     522
3.1     519
3.4     495
3.3     483
3.5     480
3.0     468
3.6     458
3.7     427
3.8     399
2.9     381
3.9     332
2.8     315
4.1     274
4.0     266
2.7     250
4.2     221
2.6     191
4.3     174
4.4     143
2.5     110
4.5      95
2.4      87
4.6      78
4.9      61
2.3      47
4.7      41
2.2      27
4.8      25
2.1      15
2.0       7
1.9       2
1.8       1
Name: Aggregate rating, dtype: int64
```

In [28]:
```
agg_rate.plot(kind='bar',color ='darkgoldenrod')
plt.show()
```



Calculate the average number of votes received by restaurants.

In [29]:
```
avg_vote=df.groupby(['Restaurant ID'])['Votes'].mean()
avg_vote=avg_vote.sort_values(ascending=False)
avg_vote.head(20)
```

Out[29]:
```
Restaurant ID
51705     10934.0
51040      9667.0
308322     7931.0
20404      7574.0
56618      6907.0
20842      5966.0
58882      5705.0
94286      5434.0
54162      5385.0
20870      5288.0
900        5172.0
35217      5145.0
1614       4986.0
301605     4914.0
463        4689.0
20350      4464.0
308022     4385.0
799        4373.0
304262     4085.0
301700     3986.0
Name: Votes, dtype: float64
```

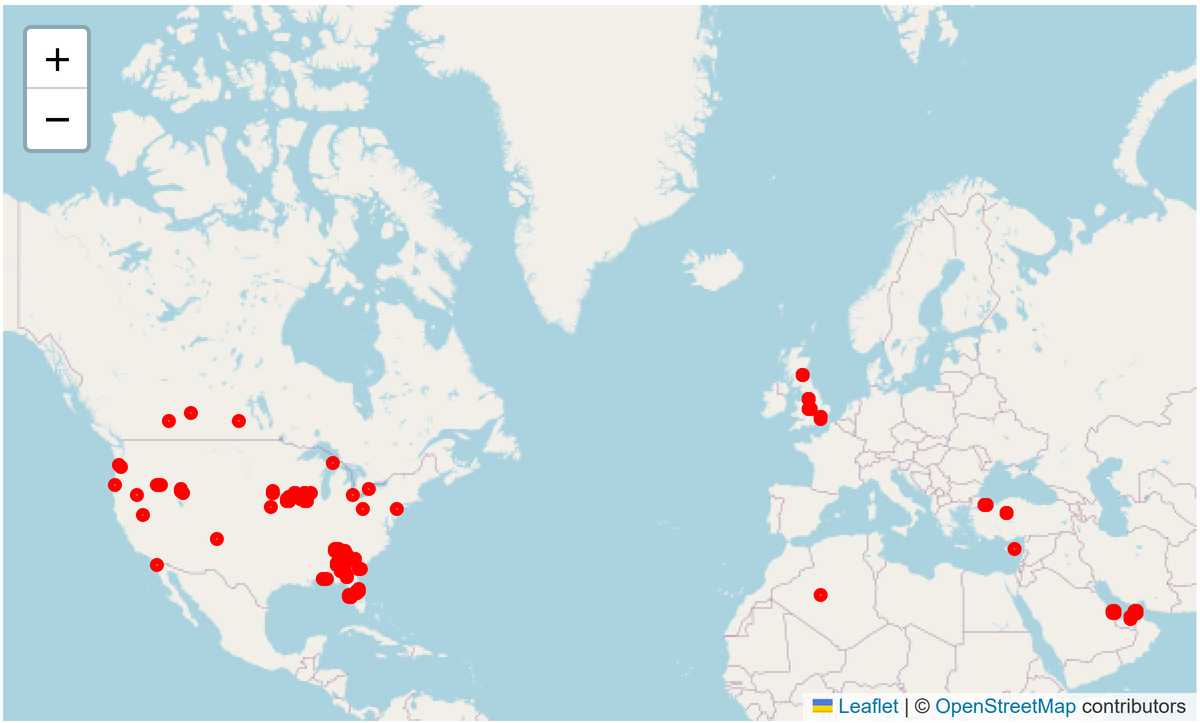
Task 3: Geographic Analysis

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

In [30]:
```
import folium
```

In [31]:
```
_map = folium.Map(location=[df['Latitude'].mean(), df['Longitude'].mean()], zo
for idx, row in df.iterrows():
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=2,  # Increase the radius to make markers larger
        color='red',  # Set the outline color to red
        fill=True,
        fill_color='red',  # Set the fill color to red
        fill_opacity=0.6,
        popup=row['Restaurant Name']
    ).add_to(_map)
_map
```

Out[31]:



Identify any patterns or clusters of restaurants in specific areas.

from the map, more clusters are seen in these continents; Asia, North America and Australia , it means most of the resturants are located there.

Task 4: Restaurant Chains

Identify if there are any restaurant chains present in the dataset.

In [32]:
```
restu_chain=df.groupby(['Restaurant Name'])['Address'].count()
restu_chain=restu_chain.sort_values(ascending=False)
restu_chain.head(50)
```

Out[32]:
```
Restaurant Name
Cafe Coffee Day            83
Domino's Pizza             79
Subway                     63
Green Chick Chop           51
McDonald's                 48
Keventers                  34
Pizza Hut                  30
Giani                      29
Baskin Robbins             28
Barbeque Nation            26
Dunkin' Donuts             22
Barista                    22
Giani's                    22
Pind Balluchi              20
Costa Coffee               20
Wah Ji Wah                 19
Pizza Hut Delivery         19
Twenty Four Seven          19
```

from the result above, it shows that Restaurants has other chains in the dataset.

Analyze the ratings and popularity of different restaurant chains.

In [33]:
```
restu_chain=df.groupby(['Restaurant Name'])['Address','Aggregate rating' ].sum
restu_chain=restu_chain.sort_values(by='Aggregate rating', ascending=False)
restu_chain.head(50)
```

```
C:\Users\PRECIOUS ONYEDEKE\AppData\Local\Temp\ipykernel_8956\3135725658.py:
1: FutureWarning: Indexing with multiple keys (implicitly converted to a tu
ple of keys) will be deprecated, use a list instead.
  restu_chain=df.groupby(['Restaurant Name'])['Address','Aggregate rating'
].sum()
```

Out[33]:

Restaurant Name	Aggregate rating
Domino's Pizza	216.5
Cafe Coffee Day	200.8
Subway	183.2
McDonald's	160.3
Green Chick Chop	136.3
Barbeque Nation	113.2

Level 3

Task 2 : Votes Analysis

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

In [34]:
```
restu_chain=df.groupby(['Restaurant Name'])['Address','Votes' ].sum()
restu_chain=restu_chain.sort_values(by='Votes', ascending=False)
restu_chain
```

```
C:\Users\PRECIOUS ONYEDEKE\AppData\Local\Temp\ipykernel_8956\2522985831.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple o
f keys) will be deprecated, use a list instead.
  restu_chain=df.groupby(['Restaurant Name'])['Address','Votes' ].sum()
```

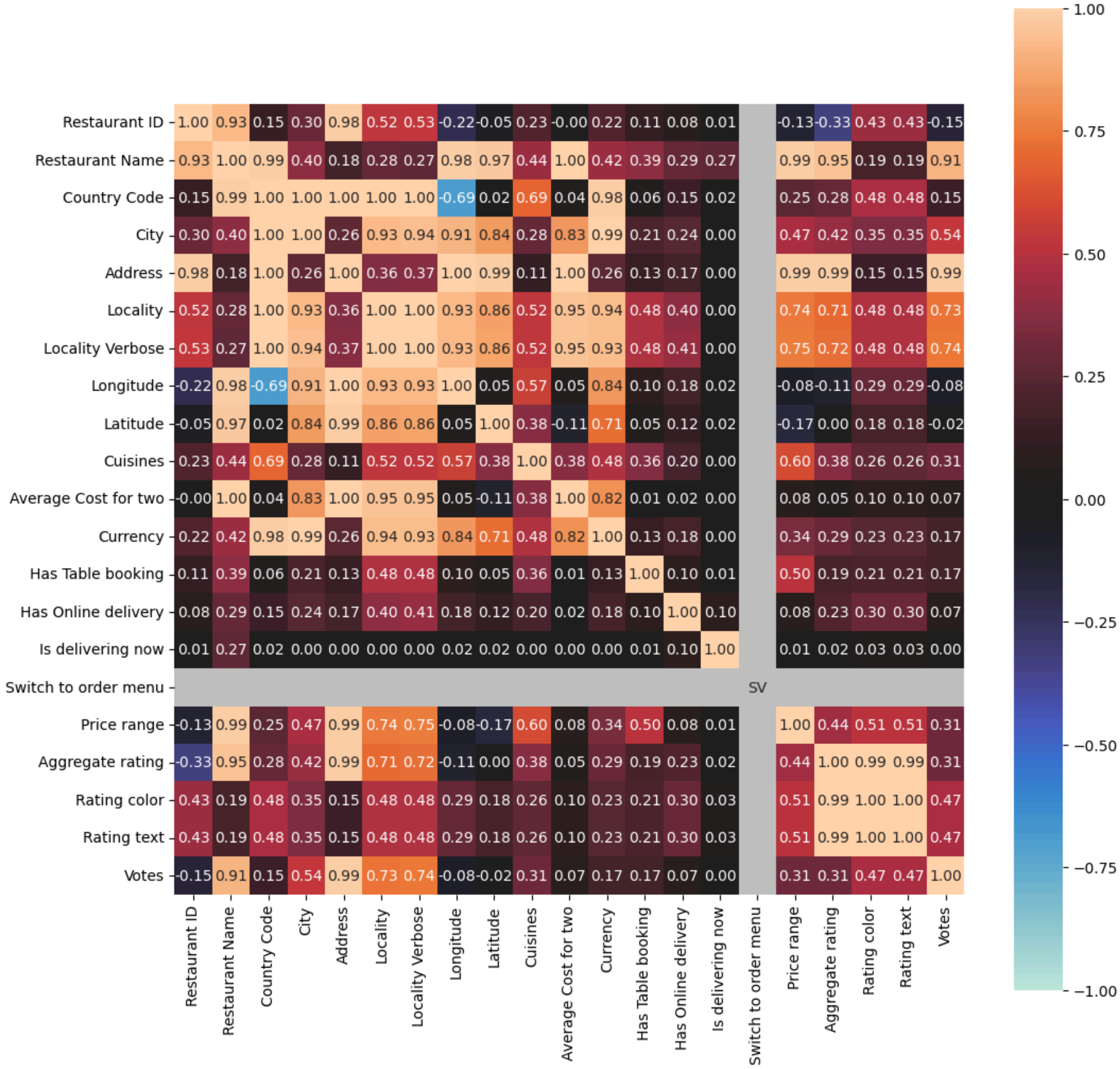
Out[34]:

	Votes
Restaurant Name	
Barbeque Nation	28142
AB's - Absolute Barbecues	13400
Toit	10934
Big Chill	10853
Farzi Cafe	10098
...	...
The Hangout-Deli	0
Foody Goody	0
Foody Dragon	0
Shiv Murti Hotel	0
Rajesh Eating Corner	0

7437 rows × 1 columns

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

In [35]:
```
from dython.nominal import associations
assoc=associations(df, num_num_assoc='pearson', figsize=(12, 12))
correlation = assoc['corr']['Votes']['Aggregate rating']
correlation
```



Out[35]: 0.31347418032500046

There is no correlation between number of votes and the rating of a restaurant.

Task 3: Price Range vs. Online Delivery and Table Booking

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

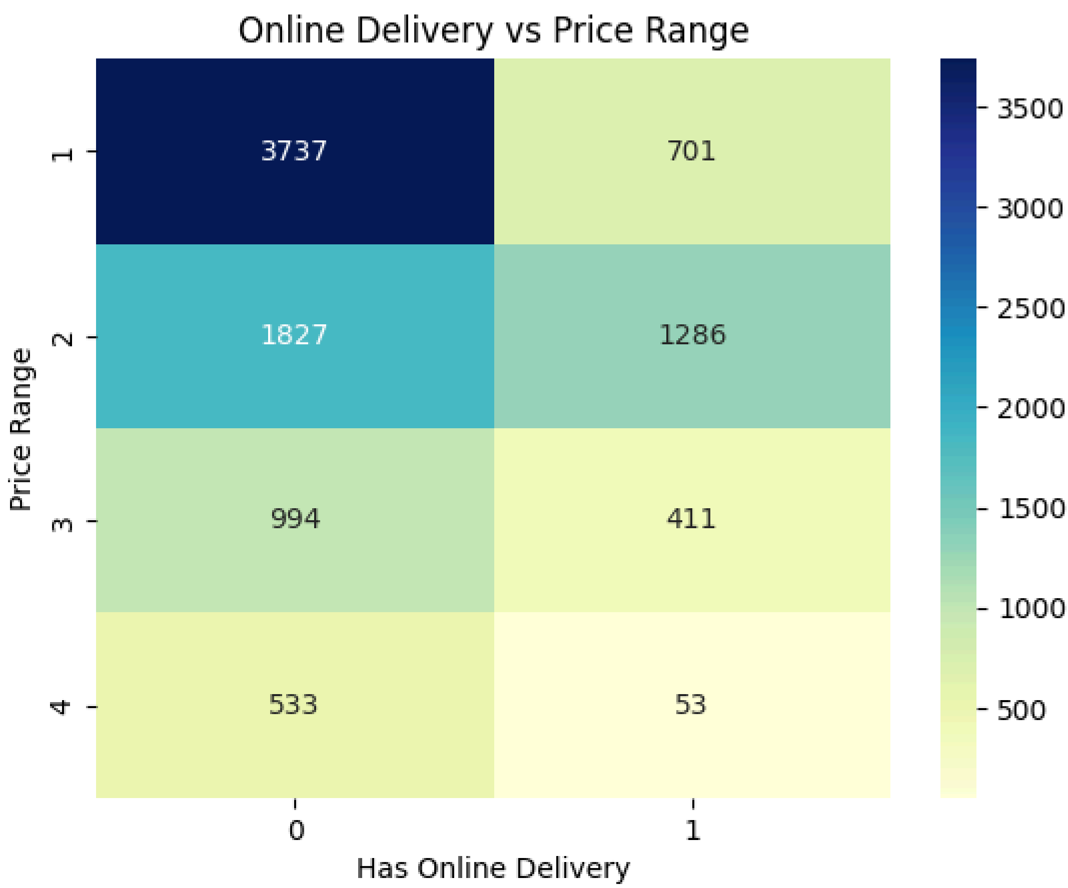
In [36]:
```
df['Price range'] = df['Price range'].astype(int)
df['Has Online delivery'] = df['Has Online delivery'].map({'Yes': 1, 'No': 0})
df['Has Table booking'] = df['Has Table booking'].map({'Yes': 1, 'No': 0})
```

```
C:\Users\PRECIOUS ONYEDEKE\AppData\Local\Temp\ipykernel_8956\145888016.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df['Price range'] = df['Price range'].astype(int)
C:\Users\PRECIOUS ONYEDEKE\AppData\Local\Temp\ipykernel_8956\145888016.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df['Has Online delivery'] = df['Has Online delivery'].map({'Yes': 1, 'No':
0})
C:\Users\PRECIOUS ONYEDEKE\AppData\Local\Temp\ipykernel_8956\145888016.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df['Has Table booking'] = df['Has Table booking'].map({'Yes': 1, 'No': 0})
```

In [37]:
```
online_delivery_ct = pd.crosstab(df['Price range'], df['Has Online delivery'])

# Contingency table for Price range and Table booking
table_booking_ct = pd.crosstab(df['Price range'], df['Has Table booking'])

print(online_delivery_ct)
print(table_booking_ct)
```

```
Has Online delivery     0     1
Price range
1                    3737   701
2                    1827  1286
3                     994   411
4                     533    53
Has Table booking     0    1
Price range
1                  4437    1
2                  2874  239
3                   761  644
4                   312  274
```

In [38]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Visualization for Online delivery
sns.heatmap(online_delivery_ct, annot=True, cmap="YlGnBu", fmt="d")
plt.title('Online Delivery vs Price Range')
plt.xlabel('Has Online Delivery')
plt.ylabel('Price Range')
plt.show()

# Visualization for Table booking
sns.heatmap(table_booking_ct, annot=True, cmap="YlGnBu", fmt="d")
plt.title('Table Booking vs Price Range')
plt.xlabel('Has Table Booking')
plt.ylabel('Price Range')
plt.show()
```



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

For booking services; the result shows that, table bookings are more when in the higher price range and less in the lower price range

For online delivery; the result shows that, online deliveries are more in the lower price range and less in the higher price range.

Therefore, higher priced resturants offer more of table booking compared to online delivery.

In [ ]: