```
Level 1 Task 1

import pandas as pd

df = pd.read_csv(r/XLS 1.csv)
print(df)
print("Number of Rows:", df.shape[0])
print("Number of Columns:", df.shape[1])
# Check for missing values
missing_values = df.isnull().sum()
# Display missing values
print("Missing Values:\n", missing_values)
# Example: Drop rows with missing values
df_cleaned = df.dropna()
print(df_cleaned)
# Display data types
print("Data Types:\n", df.dtypes)
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(df['Aggregate rating'], bins=20, kde=True)
plt.title('Distribution of Aggregate Rating')
plt.show()

from imblearn.over_sampling import SMOTE

X = df.drop('Aggregate rating', axis=1)
y = df['Aggregate rating']

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
       File "<ipython-input-1-9d53971a77d5>", line 5
         df = pd.read_csv(r/XLS 1.csv)
                           ^
     SyntaxError: invalid decimal literal
```

```
import warnings
warnings.filterwarnings('ignore')


#Importing the libraries for various use case

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set (color_codes = True)
%matplotlib inline


#Load the dataset
df = pd.read_csv("/XLS 1.csv")
print (df)
```

```
           Has Table booking N  Has Online delivery Y  Has Table booking N.1  \
     0                  7100.0                 2451.0                 8393.0
     1                     NaN                    NaN                    NaN
     2                     NaN                    NaN                    NaN
     3                     NaN                    NaN                    NaN
     4                     NaN                    NaN                    NaN
     ...                   ...                    ...                    ...
     9546                  NaN                    NaN                    NaN
     9547                  NaN                    NaN                    NaN
     9548                  NaN                    NaN                    NaN
     9549                  NaN                    NaN                    NaN
     9550                  NaN                    NaN                    NaN

           Has Table booking Y  total_restaurants  Restaurant ID  \
     0                  1158.0             9551.0        6317637
     1                     NaN                NaN        6304287
     2                     NaN                NaN        6300002
     3                     NaN                NaN        6318506
     4                     NaN                NaN        6314302
     ...                   ...                ...            ...
     9546                  NaN                NaN        5915730
     9547                  NaN                NaN        5908749
     9548                  NaN                NaN        5915807
     9549                  NaN                NaN        5916112
     9550                  NaN                NaN        5927402

                    Restaurant Name  Country Code             City  \
     0             Le Petit Souffle           162       Makati City
```

```
1            Izakaya Kikufuji           162       Makati City
2      Heat - Edsa Shangri-La           162  Mandaluyong City
3                       Ooma            162  Mandaluyong City
4                Sambo Kojin            162  Mandaluyong City
...                       ...           ...               ...
9546              Naml? Gurme           208         ??stanbul
9547             Ceviz A??ac?           208         ??stanbul
9548                    Huqqa           208         ??stanbul
9549              A???k Kahve           208         ??stanbul
9550  Walter's Coffee Roastery           208         ??stanbul

                                             Address  ...  \
0     Third Floor, Century City Mall, Kalayaan Avenu...  ...
1     Little Tokyo, 2277 Chino Roces Avenue, Legaspi...  ...
2     Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...  ...
3     Third Floor, Mega Fashion Hall, SM Megamall, O...  ...
4     Third Floor, Mega Atrium, SM Megamall, Ortigas...  ...
...                                                 ...  ...
9546  Kemanke?? Karamustafa Pa??a Mahallesi, R?ht?m ...  ...
9547  Ko??uyolu Mahallesi, Muhittin ??st?_nda?? Cadd...  ...
9548  Kuru?_e??me Mahallesi, Muallim Naci Caddesi, N...  ...
9549  Kuru?_e??me Mahallesi, Muallim Naci Caddesi, N...  ...
9550  Cafea??a Mahallesi, Bademalt? Sokak, No 21/B, ...  ...

              Currency Has Table booking  Has Online delivery  \
0     Botswana Pula(P)               Yes                   No
1     Botswana Pula(P)               Yes                   No
2     Botswana Pula(P)               Yes                   No
3     Botswana Pula(P)                No                   No
```

Double-click (or enter) to edit

```
#Explore the data set and identify the number of rows and columns

df.head()
df.info()
df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Has Table booking N   1 non-null      float64
 1   Has Online delivery Y 1 non-null      float64
 2   Has Table booking N.1 1 non-null      float64
 3   Has Table booking Y   1 non-null      float64
 4   total_restaurants     1 non-null      float64
 5   Restaurant ID         9551 non-null   int64
 6   Restaurant Name       9551 non-null   object
 7   Country Code          9551 non-null   int64
 8   City                  9551 non-null   object
 9   Address               9551 non-null   object
 10  Locality              9551 non-null   object
 11  Locality Verbose      9551 non-null   object
 12  Longitude             9551 non-null   float64
 13  Latitude              9551 non-null   float64
 14  Cuisines              9542 non-null   object
 15  Average Cost for two  9551 non-null   int64
 16  Currency              9551 non-null   object
 17  Has Table booking     9551 non-null   object
 18  Has Online delivery   9551 non-null   object
 19  Is delivering now     9551 non-null   object
 20  Switch to order menu  9551 non-null   object
 21  Price range           9551 non-null   int64
 22  Aggregate rating      9551 non-null   float64
 23  Rating color          9551 non-null   object
 24  Rating text           9551 non-null   object
 25  Votes                 9551 non-null   int64
dtypes: float64(8), int64(5), object(13)
memory usage: 1.9+ MB
(9551, 26)
```

```
#Check the missing value in each column and handle them accordingly

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

```
Has Table booking N      9550
Has Online delivery Y    9550
Has Table booking N.1    9550
Has Table booking Y      9550
total_restaurants        9550
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
```

```
#There are only 9 missing value in cuisine column which very low in copare to the dataset
#So, we can just ignore or just replace with 'Not Specified'

df ['Cuisines'].fillna('Not Specified',inplace = True)
```

Start coding or generate with AI.

```
   Has Table booking N  Has Online delivery Y  Has Table booking N.1  \
0               7100.0                 2451.0                 8393.0

   Has Table booking Y  total_restaurants  Restaurant ID   Restaurant Name  \
0               1158.0             9551.0        6317637  Le Petit Souffle

   Country Code         City  \
0           162  Makati City

                                          Address  ...          Currency  \
0  Third Floor, Century City Mall, Kalayaan Avenu...  ...  Botswana Pula(P)

  Has Table booking  Has Online delivery  Is delivering now  \
0               Yes                   No                 No

  Switch to order menu  Price range Aggregate rating Rating color Rating text  \
0                   No            3              4.8   Dark Green   Excellent

  Votes
0   314

[1 rows x 26 columns]
```

```
#Check it again

df.isnull().sum()
#Now there is no missing value
```

```
Has Table booking N      9550
Has Online delivery Y    9550
Has Table booking N.1    9550
Has Table booking Y      9550
total_restaurants        9550
Restaurant ID               0
Restaurant Name             0
Country Code                0
City                        0
Address                     0
Locality                    0
Locality Verbose            0
Longitude                   0
Latitude                    0
Cuisines                    0
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
```

```
#Checking for duplicates

dup = df.duplicated().sum()
print (f'Number of Duplicated Rows are {dup}')
```

```
Number of Duplicated Rows are 0
```

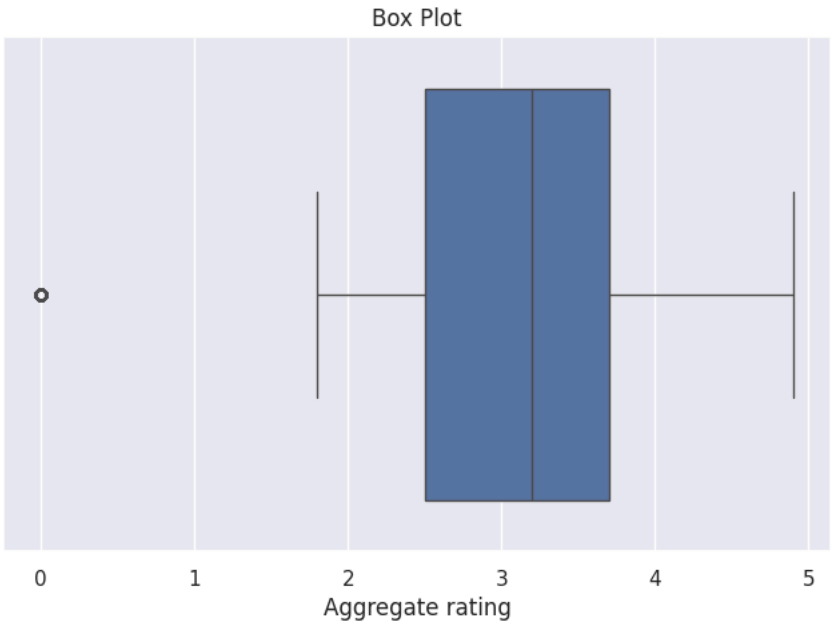
```
#Perform data type conversion if neccessary analyze the distribution of the target variable ("Aggregate Rating") and
#identify any class imbalances.
df.info()

#No need to do any data type conversion here
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Has Table booking N   1 non-null      float64
 1   Has Online delivery Y 1 non-null      float64
 2   Has Table booking N.1 1 non-null      float64
 3   Has Table booking Y   1 non-null      float64
 4   total_restaurants     1 non-null      float64
 5   Restaurant ID         9551 non-null   int64  
 6   Restaurant Name       9551 non-null   object 
 7   Country Code          9551 non-null   int64  
 8   City                  9551 non-null   object 
 9   Address               9551 non-null   object 
 10  Locality              9551 non-null   object 
 11  Locality Verbose      9551 non-null   object 
 12  Longitude             9551 non-null   float64
 13  Latitude              9551 non-null   float64
 14  Cuisines              9551 non-null   object 
 15  Average Cost for two  9551 non-null   int64  
 16  Currency              9551 non-null   object 
 17  Has Table booking     9551 non-null   object 
 18  Has Online delivery   9551 non-null   object 
 19  Is delivering now     9551 non-null   object 
 20  Switch to order menu  9551 non-null   object 
 21  Price range           9551 non-null   int64  
 22  Aggregate rating      9551 non-null   float64
 23  Rating color          9551 non-null   object 
 24  Rating text           9551 non-null   object 
 25  Votes                 9551 non-null   int64  
dtypes: float64(8), int64(5), object(13)
memory usage: 1.9+ MB
```

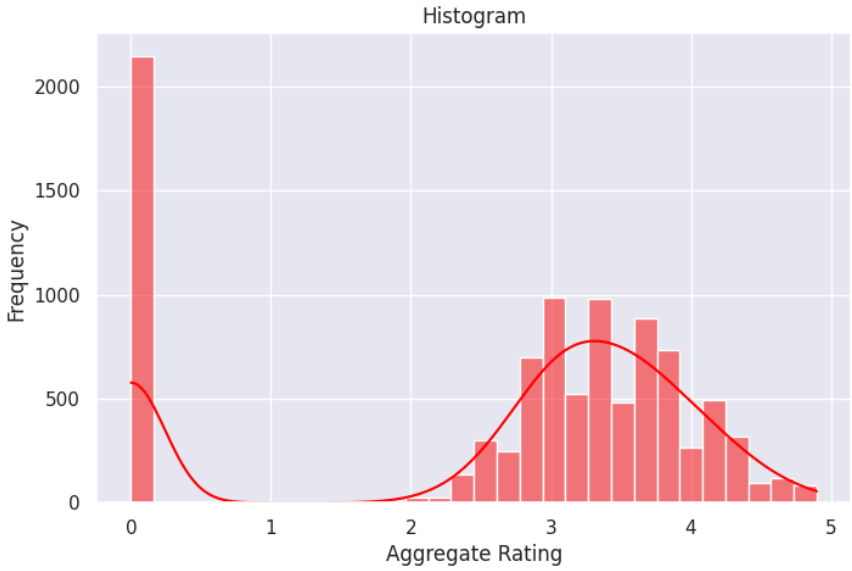
```
#Target variable "Aggregate Rating"
target = "Aggregate rating"


#Descriptive Statistics

print (df[target].describe())
```

```
count    9551.000000
mean        2.666370
std         1.516378
min         0.000000
25%         2.500000
50%         3.200000
75%         3.700000
max         4.900000
Name: Aggregate rating, dtype: float64
```

```
#Box plot

plt.figure (figsize = (8,5))
sns.boxplot (x = df[target])
plt.title ('Box Plot')
plt. xlabel ("Aggregate rating")
plt.show()
```




```
#Histogram

plt.figure (figsize =(8,5))
sns.histplot (df[target],bins =30, kde = True, color ='red')
plt.title ('Histogram')
plt.xlabel ('Aggregate Rating')
plt.ylabel('Frequency')
plt.show()

#No class imbalance
```



Start coding or generate with AI.

Level 1 - Task 2:- Task: Descriptive Analysis

```
 # 1. Calculate basic statistical measures (mean,median, standard deviation, etc.) for numerical columns.


df.describe()
```

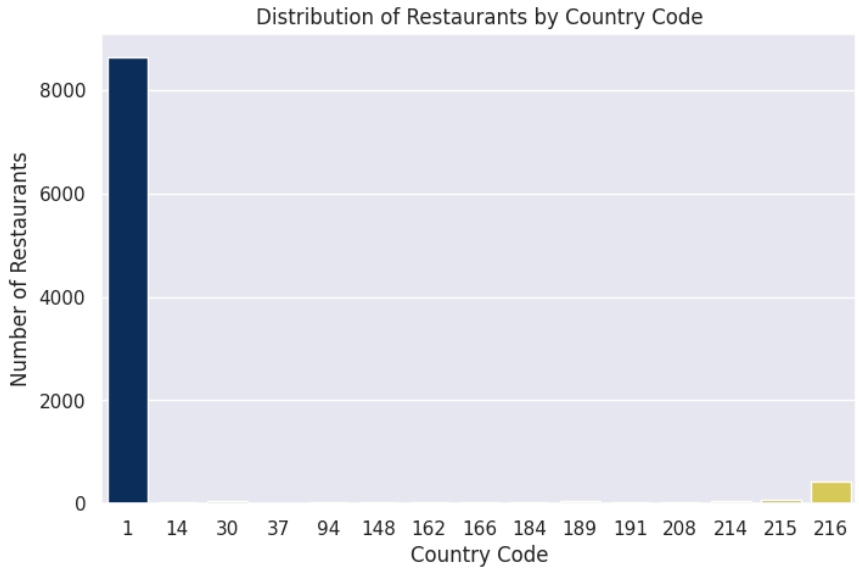
	Has Table booking N	Has Online delivery Y	Has Table booking N.1	Has Table booking Y	total_restaurants	Restaurant ID	Count Co
count	1.0	1.0	1.0	1.0	1.0	9.551000e+03	9551.0000
mean	7100.0	2451.0	8393.0	1158.0	9551.0	9.051128e+06	18.3656
std	NaN	NaN	NaN	NaN	NaN	8.791521e+06	56.7505
min	7100.0	2451.0	8393.0	1158.0	9551.0	5.300000e+01	1.0000
25%	7100.0	2451.0	8393.0	1158.0	9551.0	3.019625e+05	1.0000
50%	7100.0	2451.0	8393.0	1158.0	9551.0	6.004089e+06	1.0000
75%	7100.0	2451.0	8393.0	1158.0	9551.0	1.835229e+07	1.0000

```
df[["Average Cost for two", "Price range", "Aggregate rating", "Votes"]].describe()
```

	Average Cost for two	Price range	Aggregate rating	Votes
count	9551.000000	9551.000000	9551.000000	9551.000000
mean	1199.210763	1.804837	2.666370	156.909748
std	16121.183073	0.905609	1.516378	430.169145
min	0.000000	1.000000	0.000000	0.000000
25%	250.000000	1.000000	2.500000	5.000000
50%	400.000000	2.000000	3.200000	31.000000
75%	700.000000	2.000000	3.700000	131.000000
max	800000.000000	4.000000	4.900000	10934.000000

2. Explore the distribution of categorical variables like "Country Code," "City," and"Cuisines." Identify the top cuisines and cities with the highest number of restaurants.

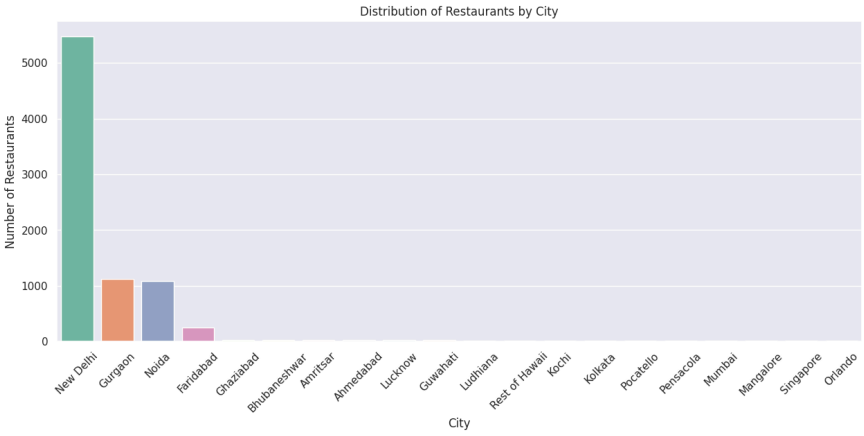
```
# Explore the distribution of "Country Code"

plt.figure(figsize=(8, 5))

sns.countplot(x='Country Code', data=df, palette='cividis')

plt.title('Distribution of Restaurants by Country Code')
plt.xlabel('Country Code')
plt.ylabel('Number of Restaurants')
plt.show()
```



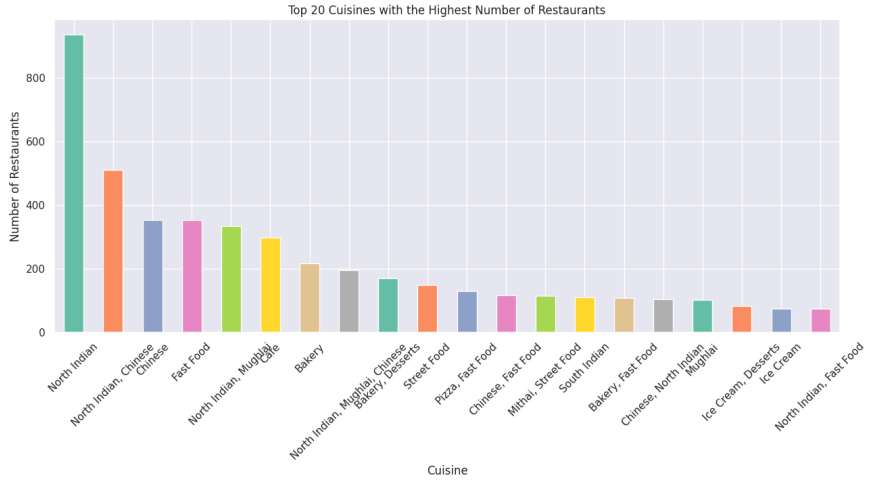
```
# Top Countries with the highest number of restaurants

top_countries = df["Country Code"].value_counts().head()

print('Top 5 Countries with the Highest Number of Restaurants:')
print(top_countries)
```

```
Top 5 Countries with the Highest Number of Restaurants:
1      8652
216     434
215      80
30       60
214      60
Name: Country Code, dtype: int64
```

```
# Explore the distribution of "City"

plt.figure(figsize=(15, 6))

sns.countplot(x='City', data=df, order=df['City'].value_counts().head(20).index, palette='Set2')

plt.title('Distribution of Restaurants by City')
plt.xlabel('City')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45)
plt.show()
```




```
# Explore the distribution of "Cuisines"

plt.figure(figsize=(15, 6))

cuisines_count = df['Cuisines'].value_counts()
cuisines_count.head(20).plot(kind='bar', color=sns.color_palette("Set2"))

plt.title('Top 20 Cuisines with the Highest Number of Restaurants')
plt.xlabel('Cuisine')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45)
plt.show()
```




```
# Top cities with the highest number of restaurants

top_cities = df['City'].value_counts().head(10)

print('Top 10 Cities with the Highest Number of Restaurants:')
print(top_cities)
```

```
Top 10 Cities with the Highest Number of Restaurants:
New Delhi       5473
Gurgaon         1118
Noida           1080
Faridabad        251
Ghaziabad         25
Bhubaneshwar      21
Amritsar          21
Ahmedabad         21
Lucknow           21
Guwahati          21
Name: City, dtype: int64
```

```
# Top cuisines with the highest number of restaurants

top_cuisines = cuisines_count.head(10)
print('Top 10 Cuisines with the Highest Number of Restaurants:')
print(top_cuisines)
```

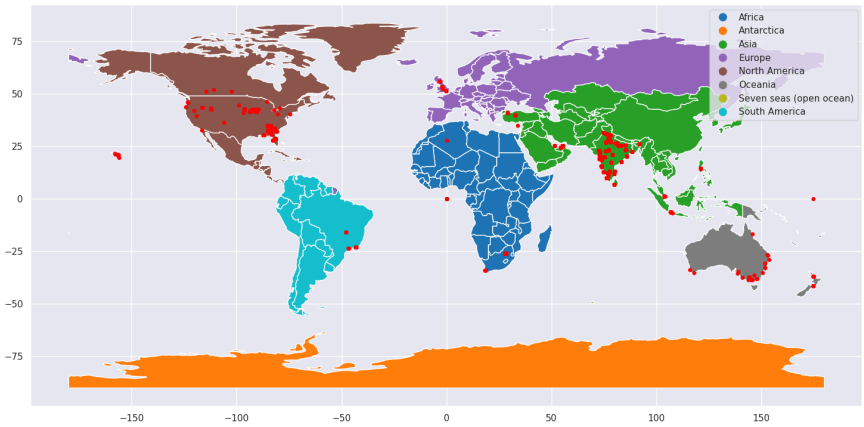
```
Top 10 Cuisines with the Highest Number of Restaurants:
North Indian                      936
North Indian, Chinese             511
Chinese                           354
Fast Food                         354
North Indian, Mughlai             334
Cafe                              299
Bakery                            218
North Indian, Mughlai, Chinese    197
Bakery, Desserts                  170
Street Food                       149
Name: Cuisines, dtype: int64
```

Level 1 - Task 3:- Task: Geospatial Analysis

1. Visualize the locations of restaurants on a map using latitude and longitude information.

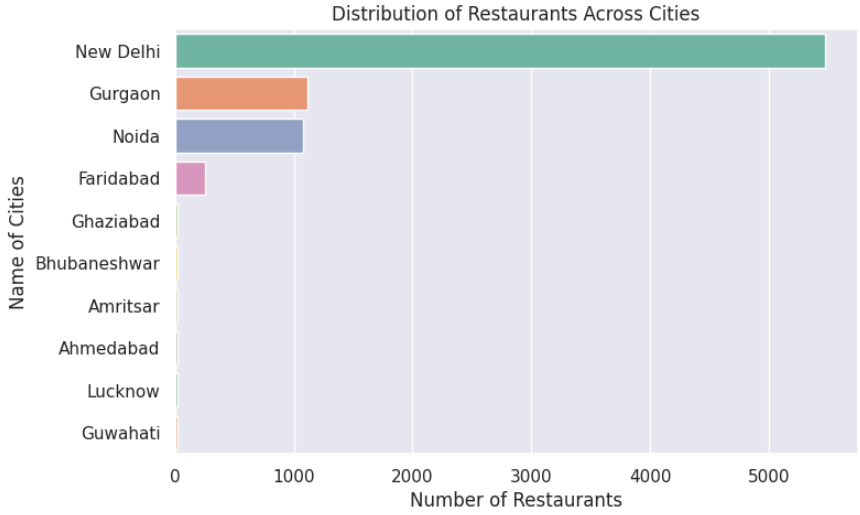
```
# Locations of restaurants on a map using latitude and longitude information
# Import the necessary libraries

from shapely.geometry import Point
import geopandas as gpd
from geopandas import GeoDataFrame
```

```
# Create Point geometry from latitude and longitude using Shapely
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.Longitude, df.Latitude))

# Create a base map of the world using Geopandas
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

# Create a map that fits the screen and plots the restaurant locations
gdf.plot(ax=world.plot("continent", legend = True, figsize=(18, 15)), marker='o', color='red', markersize=15)

plt.show()
```

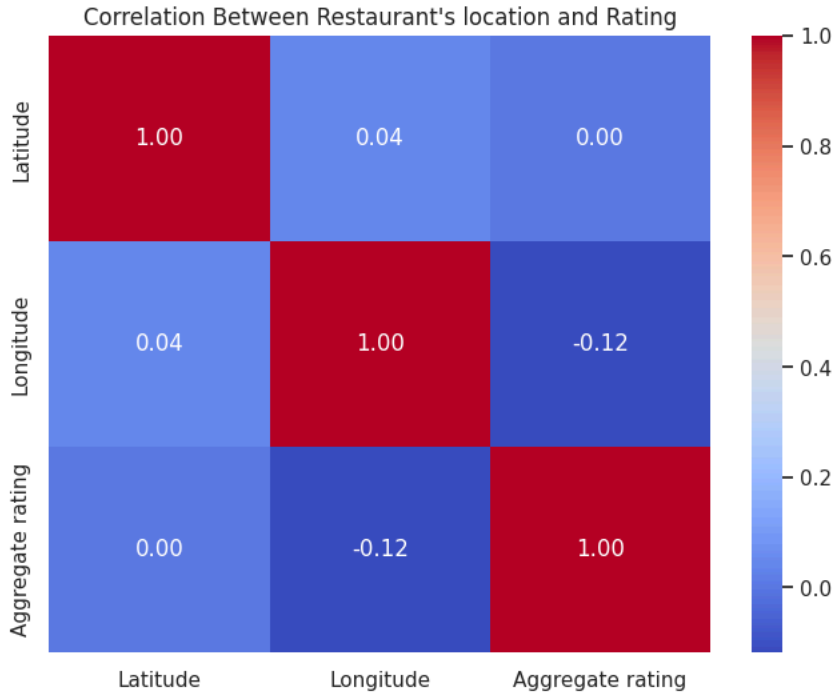


2. Analyze the distribution of restaurants across different cities or countries. Determine if there is any correlation between the restaurant's location and its rating.

```
# Distribution of restaurants across different cities or countries

plt.figure(figsize=(8, 5))

# There are many cities names present in the data, so i select only the top 10 cities
sns.countplot(y = df['City'], order=df.City.value_counts().head(10).index, palette='Set2')

plt.xlabel('Number of Restaurants')
plt.ylabel('Name of Cities')
plt.title('Distribution of Restaurants Across Cities')

plt.show()
```




```
# Checking correlation between the restaurant's location and its rating

plt.figure(figsize=(8, 6))

# Calculate the correlation between latitude, longitude, and ratings
correlation_matrix = df[['Latitude', 'Longitude', 'Aggregate rating']].corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Between Restaurant's location and Rating")

plt.show()
```



Saving this Dataset for next Level Tasks

OBSERVATION : The Restaurant Dataset has various attributes such as Restaurant Ids, Restaurant Name, City, Country, Types of Cuisines, etc. **bold text** There are 9551 Rows and 21 Columns.

In this Dataset 9 missing values from the "Cuisines" Column. So, it can be replaced by Not Specified.

In this Dataset, No Duplicates are present.

No need to do any data type conversion & No Class Imbalance.

After exploring the distribution of categorical variables like "Country Code", "City," and "Cuisines. I've found that the majority of restaurants are located in Country Code 1, followed by the second-highest concentration in Country Code 216. And 5473 Restaurant are located in Delhi, followed by 1118 in Gurgaon and 1080 in Noida.

Top Cuisines are "North Indian", "Chinese", "Fast Food".

USA and India have the most number of restaurants.

Level 2 - Task 1:- Task: Table Booking and Online Delivery

1. Determine the percentage of restaurants that offer table booking and online delivery.

```
# Check for value counts

df["Has Table booking"].value_counts()
```

```
No     8393
Yes    1158
Name: Has Table booking, dtype: int64
```

```
df["Has Online delivery"].value_counts()
```

```
No     7100
Yes    2451
Name: Has Online delivery, dtype: int64
```

```
print("Table Booking : ", round((1158/(8393+1158)) *100, 2),"%")

print("Online Delivery : ", round((2451/(7100+2451)) *100, 2),"%")
```

```
Table Booking :  12.12 %
Online Delivery :  25.66 %
```

```
# Another way to find the percentage


# Number of restaurants offering table booking & online delivery

table_booking = df['Has Table booking'].value_counts().get('Yes')

online_delivery = df['Has Online delivery'].value_counts().get('Yes')

# Calculate the percentage for table booking * online delivery

percentage_table_booking = (table_booking / len(df)) * 100

percentage_online_delivery = (online_delivery / len(df)) * 100

print(f"Percentage of Restaurants offers table booking: {percentage_table_booking:.2f} %\n")
print(f"Percentage of Restaurants offers online delivery: {percentage_online_delivery:.2f} %")
```

```
Percentage of Restaurants offers table booking: 12.12 %

Percentage of Restaurants offers online delivery: 25.66 %
```

2. Compare the average ratings of restaurants with table booking and those without.

```
# Filter the DataFrame for rows with 'Yes' & 'No' in the 'Table Booking' column

df_with_table_booking = df[df['Has Table booking'] == 'Yes']

df_without_table_booking = df[df['Has Table booking'] == 'No']

# After filtering rows with and without table booking

print("Rows With Table Booking :", df_with_table_booking.shape)

print("Rows Without Table Booking :", df_without_table_booking.shape)
```

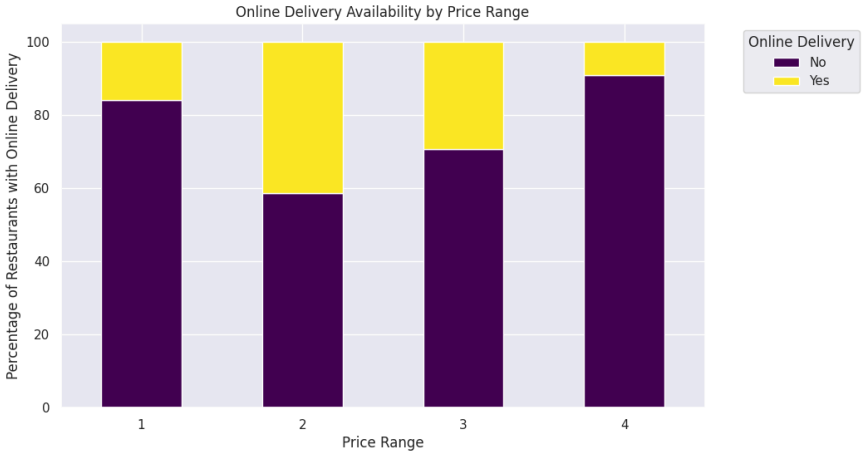
```
Rows With Table Booking : (1158, 26)
Rows Without Table Booking : (8393, 26)
```

```
# Average Ratings of Restaurants

print("Average Ratings:- ")
print(" With Table Booking : ", round(df_with_table_booking["Aggregate rating"].mean(),2))
print(" Without Table Booking : ", round(df_without_table_booking["Aggregate rating"].mean(),2))
```

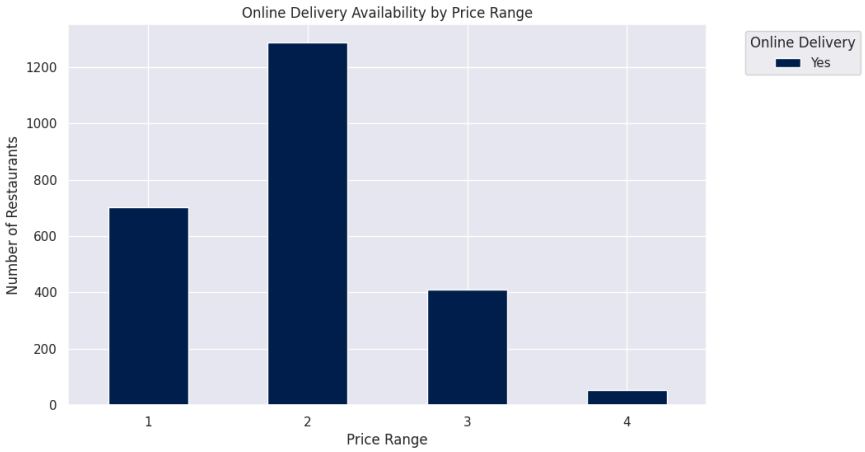
```
Average Ratings:-
 With Table Booking :  3.44
 Without Table Booking :  2.56
```

3. Analyze the availability of online delivery among restaurants with different price ranges.

```
# Group by 'Price range' and calculate the percentage of restaurants with online delivery

Online_Delivery_by_price_range = df.groupby('Price range')['Has Online delivery'].value_counts(normalize=True).unstack() * 100

Online_Delivery_by_price_range.plot(kind='bar', stacked=True, colormap='viridis', figsize=(10, 6))

plt.title('Online Delivery Availability by Price Range')
plt.xlabel('Price Range')
plt.ylabel('Percentage of Restaurants with Online Delivery')
plt.xticks(rotation = 0)
plt.legend(title='Online Delivery', bbox_to_anchor=(1.05, 1))

plt.show()
```




```
# Taking only those restaurant with online Delivery available

Online_Delivery_Yes = df[df['Has Online delivery'] == 'Yes']

# Group by 'Price range' and calculate the percentage of restaurants with online delivery

Online_Delivery_counts = Online_Delivery_Yes.groupby(['Price range', 'Has Online delivery']).size().unstack()

Online_Delivery_counts.plot(kind='bar', stacked=True, colormap='cividis', figsize=(10, 6))
plt.title('Online Delivery Availability by Price Range')
plt.xlabel('Price Range')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation = 0)
plt.legend(title='Online Delivery', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```



From the above 1st graph we can see that most of the restaurant do not have the online delivery services. In price range 1 less than 20 % are available, In price range 2 around 40 % are availabe, In price range 3 it look like 30 % are available and In price range 4 only 10 % are available.

From the above 2nd graph, we can analyze, people used to buy from the Price range 2 and very less number of people buy food from Price range 4 may be because of its costliest in price compare to others.

Level 2 - Task 2:- Task: Price Range Analysis

1. Determine the most common price range among all the restaurants.

```
df["Price range"].value_counts()
```

```
1    4444
2    3113
3    1408
4     586
Name: Price range, dtype: int64
```

```
most_common = df["Price range"].mode()[0]

print("Most Common Price range among all the restaurant : ", most_common)
```

```
Most Common Price range among all the restaurant :  1
```

2. Calculate the average rating for each price range. & Identify the color that represents the highest average rating among different price ranges.

```
Avg_Rating_by_price_range = df.groupby('Price range')['Aggregate rating'].mean().round(2)

print("Average Rating for each price range :")
print(Avg_Rating_by_price_range)
```

```
Average Rating for each price range :
Price range
1    2.00
2    2.94
3    3.68
4    3.82
Name: Aggregate rating, dtype: float64
```

Double-click (or enter) to edit

```
# Find the price range with the highest average rating
highest_avg_rating_color = Avg_Rating_by_price_range.idxmax()

plt.bar(Avg_Rating_by_price_range.index, Avg_Rating_by_price_range, color='skyblue', width=0.5)

plt.bar(highest_avg_rating_color, Avg_Rating_by_price_range[highest_avg_rating_color], color='green', width=0.5)

plt.xlabel('Price Range')
plt.ylabel('Average Rating')
plt.title('Average Rating by Price Range')

plt.show()
```

Price range 4 get the highest average rating, which is 3.82, followed by price range 3, 2 and 1

Level 2 - Task 3:- Task: Feature Engineering

1. Extract additional features from the existing columns, such as the length of the restaurant name or address.

```
# Extract the length of the restaurant name and address and create new columns

df['Restaurant Name Length'] = df['Restaurant Name'].apply(lambda x: len(str(x)))

df['Address Length'] = df['Address'].apply(lambda x: len(str(x)))

df[['Restaurant Name', 'Restaurant Name Length', 'Address', 'Address Length']]
```

	Restaurant Name	Restaurant Name Length	Address	Address Length
0	Le Petit Souffle	16	Third Floor, Century City Mall, Kalayaan Avenu...	71
1	Izakaya Kikufuji	16	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	67
2	Heat - Edsa Shangri-La	22	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	56
3	Ooma	4	Third Floor, Mega Fashion Hall, SM Megamall, O...	70
4	Sambo Kojin	11	Third Floor, Mega Atrium, SM Megamall, Ortigas...	64
...	...	...	...	...
9546	Naml? Gurme	11	Kemanke?? Karamustafa Pa??a Mahallesi, R?ht?m ...	103
9547	Ceviz A??ac?	12	Ko??uyolu Mahallesi, Muhittin ?? st?_nda?? Cadd...	77

2. Create new features like "Has Table Booking"or "Has Online Delivery" by encoding categorical variables.

```
# Creating new features "Has Table Booking" and "Has Online Delivery"

df['Has Table Booking'] = df['Has Table booking'].apply(lambda x: 1 if x == 'Yes' else 0)

df['Has Online Delivery'] = df['Has Online delivery'].apply(lambda x: 1 if x == 'Yes' else 0)


df[['Has Table booking', 'Has Table Booking', 'Has Online delivery', 'Has Online Delivery']]
```

	Has Table booking	Has Table Booking	Has Online delivery	Has Online Delivery
0	Yes	1	No	0
1	Yes	1	No	0
2	Yes	1	No	0
3	No	0	No	0
4	Yes	1	No	0
...	...	...	...	...
9546	No	0	No	0
9547	No	0	No	0
9548	No	0	No	0
9549	No	0	No	0
9550	No	0	No	0

9551 rows × 4 columns

Two new columns added, 'Restaurant Name length' and 'Address Length' from the length of the restaurant name or address And also two new binary column added by encoding categorical variables, 'Has Table booking' and 'Has Online delivery

OBSERVATION : Percentage of Restaurants offers table booking is 12.12 % &Percentage of Restaurants offers online delivery is 25.66 %.

Average Ratings With Table Booking is 3.44 & Without Table Booking is 2.56.

Most of the restaurant do not have the online delivery services. In price range 1 less than 20 % are available, In price range 2 around 40 % are availabe, In price range 3 it look like 30 % are available and In price range 4 only 10 % are available.

People mostly buy from the Price range 2 and very less number of people buy food from Price range 4 may be because of its costliest in price compare to others.

Most Common Price range among all the restaurant is 1.

Price range 4 get the highest average rating, which is 3.82, followed by price range 3, 2 and 1.

Level 3 - Task 1:- Task: Predictive Modeling

1. Build a regression model to predict the aggregate rating of a restaurant based on available features.

   Split the dataset into training and testing sets and evaluate the model's performance using appropriate metrics.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


# Convert categorical variables to numeric
df1 = pd.get_dummies(df, columns=['Has Table booking', 'Has Online delivery'], drop_first=True)


X = df1[['Average Cost for two', 'Votes', 'Price range', 'Has Table booking_Yes', 'Has Online delivery_Yes']]
y = df1['Aggregate rating']


# Split the dataset into training and testing sets (80% training, 20% testing)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Initialize and train the linear regression model

model = LinearRegression()
model.fit(X_train, y_train)

# Predict ratings on the testing set
y_pred = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Model: Linear Regression")
print("Mean Squared Error (MSE):", mse)
print("R-squared (R2) Score:", r2)
```

```
Model: Linear Regression
Mean Squared Error (MSE): 1.6764802747031442
R-squared (R2) Score: 0.2634446409021949
```

2. Experiment with different algorithms (e.g.,linear regression, decision trees, random forest) and compare their performance.

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor


# Initialize and train different regression models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42)
}

# Evaluate models
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results[name] = {'MSE': mse, 'R2 Score': r2}

# Display results
results_df = pd.DataFrame(results)
print(results_df)
```

```
           Linear Regression  Decision Tree  Random Forest
MSE                 1.676480       0.203498       0.133938
R2 Score            0.263445       0.910594       0.941155
```

Level 3 - Task 2:- Task: Customer Preference Analysis

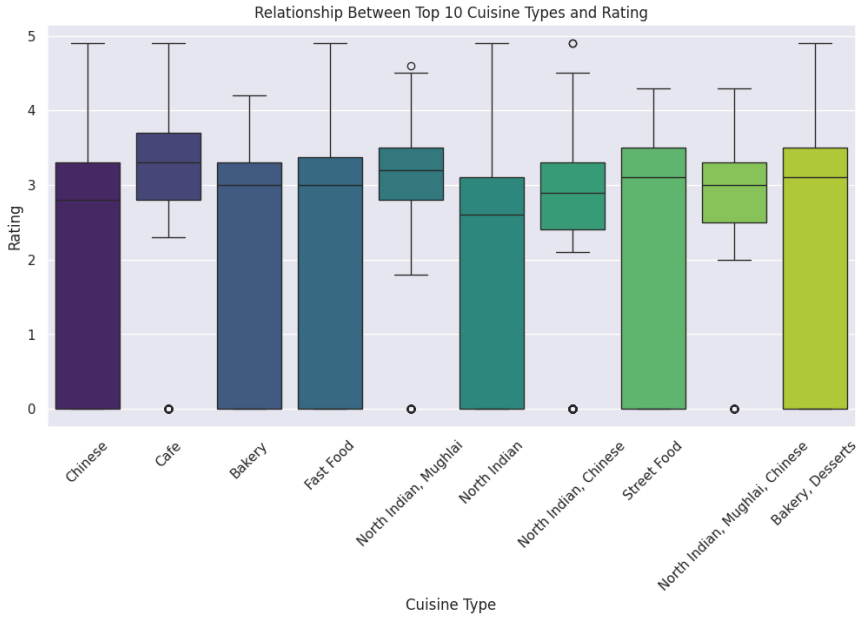
1. Analyze the relationship between the type of cuisine and the restaurant's rating.

```
# Split cuisines into individual cuisine types
cuisines = df1['Cuisines']


cuisines.value_counts().head(10)
```

```
North Indian                      936
North Indian, Chinese             511
Chinese                           354
Fast Food                         354
North Indian, Mughlai             334
Cafe                              299
Bakery                            218
North Indian, Mughlai, Chinese    197
Bakery, Desserts                  170
Street Food                       149
Name: Cuisines, dtype: int64
```

```
# Get the top 10 most common cuisines
top_10_cuisines = cuisines.value_counts().head(10).index


# Create a DataFrame with cuisine types and corresponding ratings
cuisine_ratings = pd.DataFrame({'Cuisine': cuisines, 'Rating': df1['Aggregate rating']})
```

```
# Filter cuisine_ratings DataFrame to include only the top 10 cuisines
cuisine_ratings_top_10 = cuisine_ratings[cuisine_ratings['Cuisine'].isin(top_10_cuisines)]


# Plot the relationship between the top 20 cuisine types and rating
plt.figure(figsize=(12, 6))
sns.boxplot(x='Cuisine', y='Rating', data=cuisine_ratings_top_10, palette='viridis')
plt.title('Relationship Between Top 10 Cuisine Types and Rating')
plt.xlabel('Cuisine Type')
plt.ylabel('Rating')
plt.xticks(rotation=45)
plt.show()
```

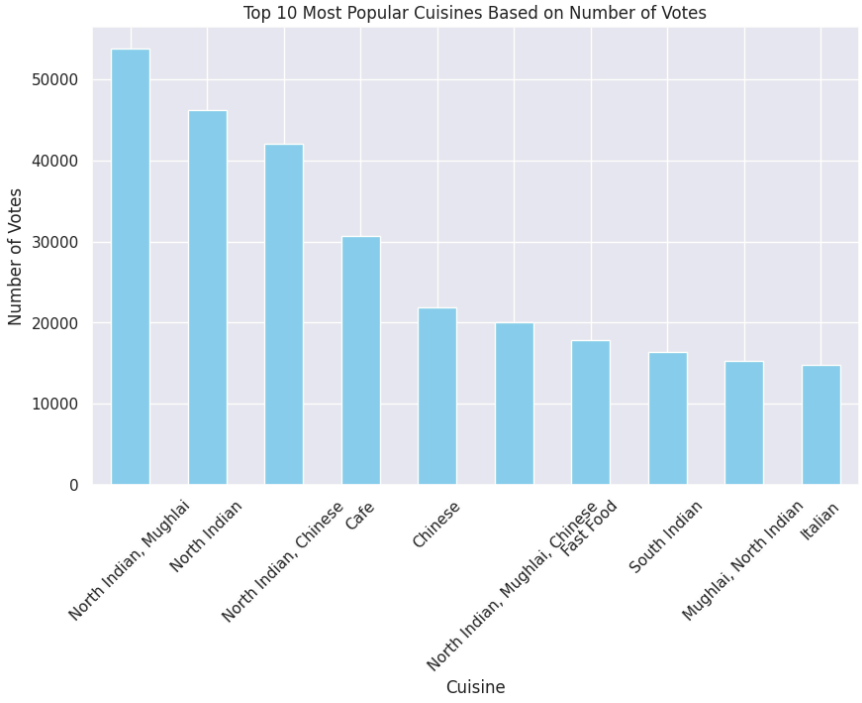


2. Identify the most popular cuisines among customers based on the number of votes.

```
# Create a DataFrame with cuisine types and corresponding votes
cuisine_votes = pd.DataFrame({'Cuisine': cuisines, 'Votes': df1['Votes']})


# Group by cuisine and sum the votes for each cuisine
cuisine_votes_sum = cuisine_votes.groupby('Cuisine')['Votes'].sum()


# Sort cuisines based on the total votes in descending order
popular_cuisines = cuisine_votes_sum.sort_values(ascending=False)


# Display the top 10 most popular cuisines
print("Top 10 Most Popular Cuisines Based on Number of Votes:")
print(popular_cuisines.head(10))
```

```
Top 10 Most Popular Cuisines Based on Number of Votes:
Cuisine
North Indian, Mughlai            53747
North Indian                     46241
North Indian, Chinese            42012
Cafe                             30657
Chinese                          21925
North Indian, Mughlai, Chinese   20115
Fast Food                        17852
South Indian                     16433
Mughlai, North Indian            15275
Italian                          14799
Name: Votes, dtype: int64
```

```
# Plotting the bar plot
plt.figure(figsize=(10, 6))
popular_cuisines.head(10).plot(kind='bar', color='skyblue')
plt.title('Top 10 Most Popular Cuisines Based on Number of Votes')
plt.xlabel('Cuisine')
plt.ylabel('Number of Votes')
plt.xticks(rotation=45)
plt.show()
```

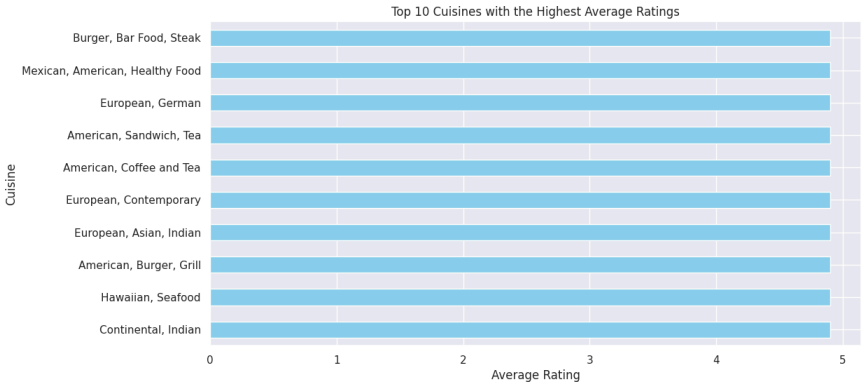


3. Determine if there are any specific cuisines that tend to receive higher ratings.

```
# Create a DataFrame with cuisine types and corresponding ratings
cuisine_ratings = pd.DataFrame({'Cuisine': cuisines, 'Rating': df1['Aggregate rating']})


# Calculate the average rating for each cuisine
average_rating_by_cuisine = cuisine_ratings.groupby('Cuisine')['Rating'].mean()


# Sort cuisines based on the average rating in descending order
sorted_cuisines_by_rating = average_rating_by_cuisine.sort_values(ascending=False)


# Display the top 10 cuisines with the highest average ratings
print("Top 10 Cuisines with the Highest Average Ratings:")
print(sorted_cuisines_by_rating.head(10))
```

```
Top 10 Cuisines with the Highest Average Ratings:
Cuisine
Continental, Indian               4.9
Hawaiian, Seafood                 4.9
American, Burger, Grill           4.9
European, Asian, Indian           4.9
European, Contemporary            4.9
American, Coffee and Tea          4.9
American, Sandwich, Tea           4.9
European, German                  4.9
Mexican, American, Healthy Food   4.9
Burger, Bar Food, Steak           4.9
Name: Rating, dtype: float64
```

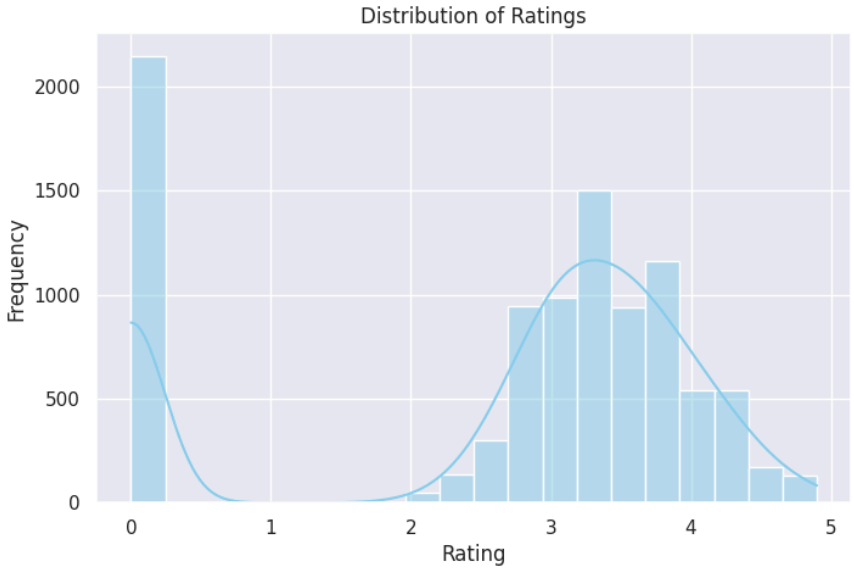
```
# Plot the graph
plt.figure(figsize=(12, 6))
sorted_cuisines_by_rating.head(10).plot(kind='barh', color='skyblue')
plt.title('Top 10 Cuisines with the Highest Average Ratings')
plt.xlabel('Average Rating')
plt.ylabel('Cuisine')
plt.show()
```



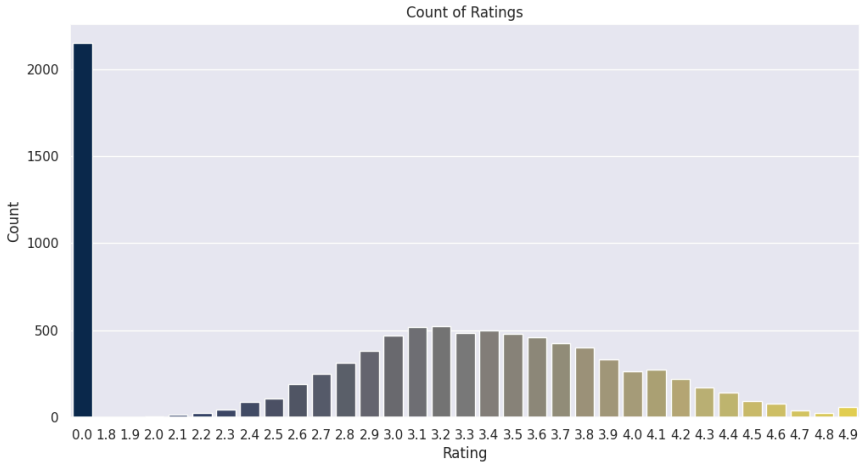
Level 3 - Task 3:- Task: Data Visualization

1. Create visualizations to represent the distribution of ratings using different charts (histogram, barplot, etc.).

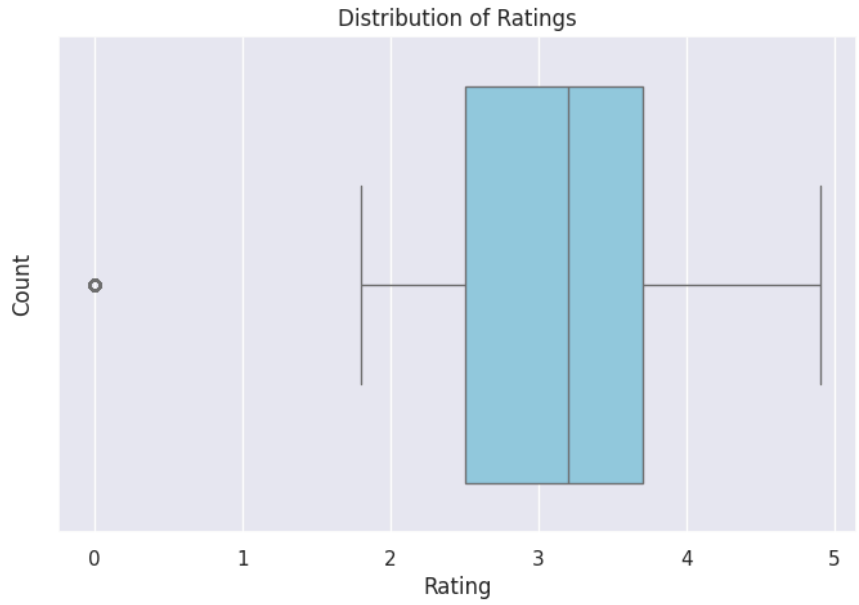
```
# Create a histogram to visualize the distribution of ratings
plt.figure(figsize=(8, 5))
sns.histplot(df1['Aggregate rating'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



```
# Create a bar plot to visualize the count of ratings
plt.figure(figsize=(12, 6))
sns.countplot(x='Aggregate rating', data=df1, palette='cividis')
plt.title('Count of Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



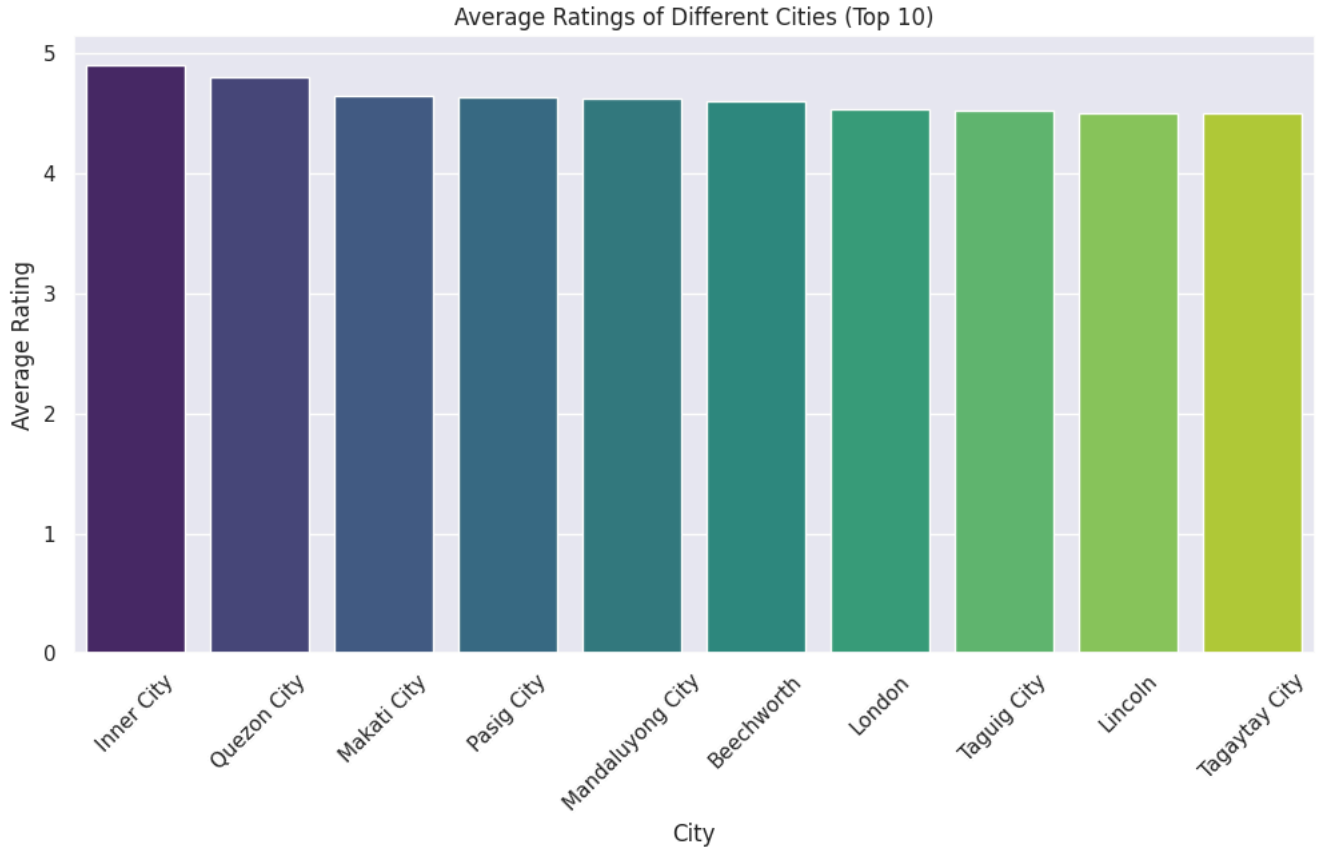

```
# Create a box plot to visualize the distribution of ratings
plt.figure(figsize=(8, 5))
sns.boxplot(x='Aggregate rating', data=df1, color='skyblue')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



2. Compare the average ratings of different cuisines or cities using appropriate visualizations.

```
# Group data by city and calculate the average rating for each city
average_rating_by_city = df1.groupby('City')['Aggregate rating'].mean().sort_values(ascending=False)
```

```
# Plot the average ratings of different cities (top 10 cities)
plt.figure(figsize=(12, 6))
sns.barplot(x=average_rating_by_city.head(10).index, y=average_rating_by_city.head(10).values, palette='viridis')
plt.title('Average Ratings of Different Cities (Top 10)')
plt.xlabel('City')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



3. Visualize the relationship between various features and the target variable to gain insights.

```
# Pair plot: Pairwise relationships between features and Aggregate Rating

features = ['Average Cost for two', 'Votes', 'Price range', 'Has Table booking_Yes', 'Has Online delivery_Yes', 'Aggregate rating']
sns.pairplot(df1[features])
plt.show()
```