

## cognifyz-level-3

May 4, 2024

```
[1]: import warnings
warnings.filterwarnings("ignore")
```

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

```
[3]: df = pd.read_csv("C:/Users/rishi/OneDrive/Documents/cognifyz internship/Dataset_1_1.csv")
```

```
[5]: 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
```

## TASK 1 : PREDICTIVE MODELING

```
[6]: from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.metrics import mean_squared_error, r2_score
```

```
[7]: df = pd.get_dummies(df, columns=['Has Table booking', 'Has Online delivery'],
     ↪drop_first=True)
```

```
[8]: features = ['Average Cost for two', 'Votes', 'Price range', 'Has Table
     ↪booking_Yes', 'Has Online delivery_Yes']
     target = 'Aggregate rating'

     X = df[features]
     y = df[target]
```

```
[9]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
     ↪random_state=42)
```

```
[10]: models = {
        "Linear Regression": LinearRegression(),
        "Decision Tree": DecisionTreeRegressor(),
        "Random Forest": RandomForestRegressor()
    }
```

```
[11]: for model_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)
        print(f"Model: {model_name}")
        print(f"Mean Squared Error: {mse}")
        print(f"R-squared: {r2}")
        print("-----")
```

```
Model: Linear Regression
Mean Squared Error: 1.6764802747031446
R-squared: 0.26344464090219477
-----
```

```
Model: Decision Tree
Mean Squared Error: 0.20870488595952855
R-squared: 0.9083062863649817
```

```

-----
Model: Random Forest
Mean Squared Error: 0.13412489834172028
R-squared: 0.9410727259051496
-----

```

## Task 2: Customer Preference Analysis

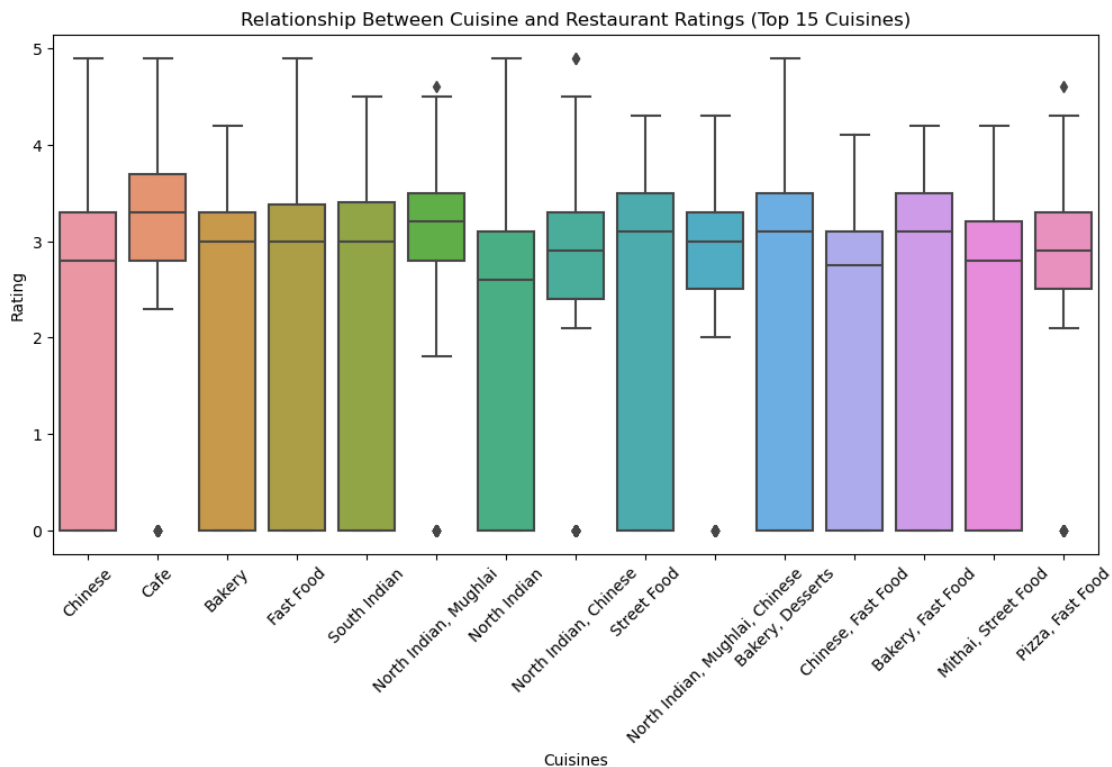
```

[12]: top_n = 15
      top_cuisines = df['Cuisines'].value_counts().nlargest(top_n).index

[13]: df_filtered = df[df['Cuisines'].isin(top_cuisines)]

[15]: plt.figure(figsize=(12, 6))
      sns.boxplot(data=df_filtered, x='Cuisines', y='Aggregate rating')
      plt.title(f'Relationship Between Cuisine and Restaurant Ratings (Top {top_n} Cuisines)')
      plt.xlabel('Cuisines')
      plt.ylabel('Rating')
      plt.xticks(rotation=45)
      plt.show()

```

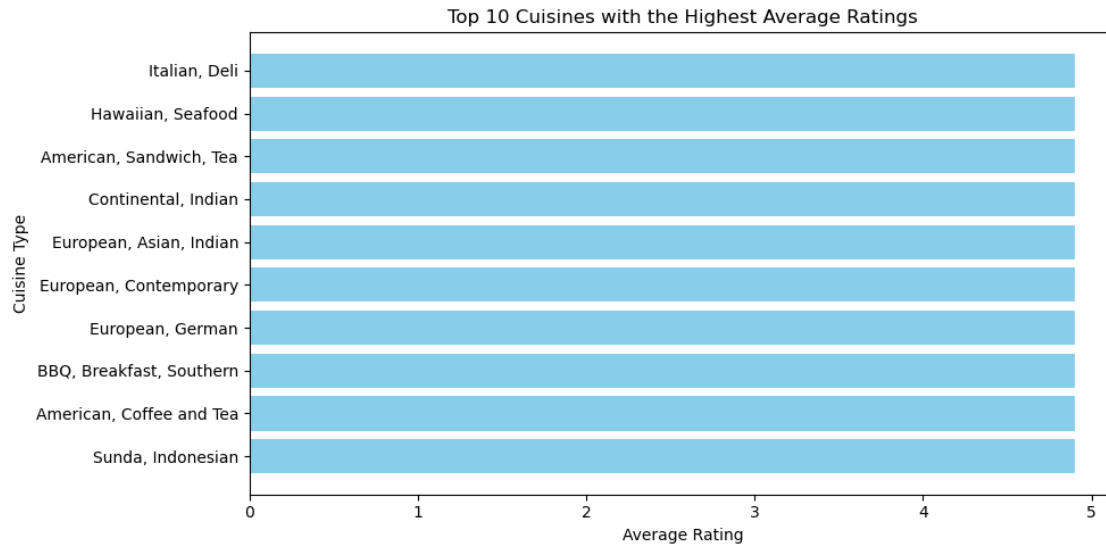


```
[16]: # Identifying most popular cuisines based on number of votes
top_cuisines = df.groupby('Cuisines')['Votes'].sum().nlargest(10)

# Display result
print("Top Cuisines by Number of Votes:")
print(top_cuisines)
```

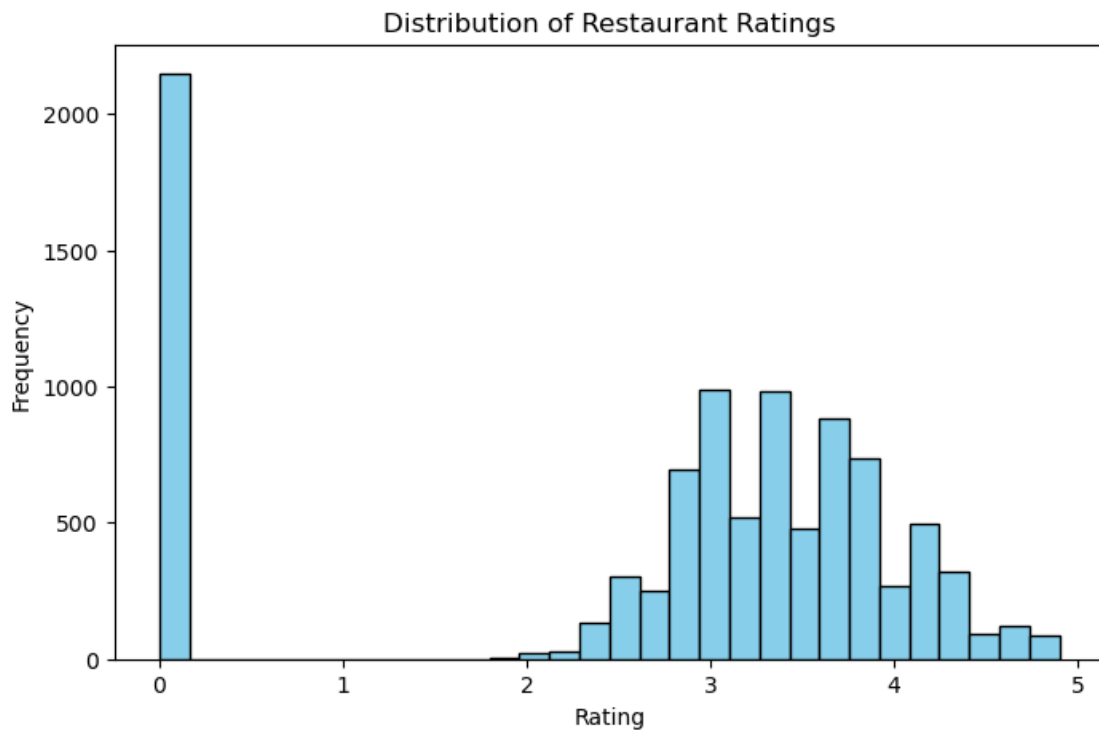
```
Top Cuisines by Number of Votes:
Cuisines
North Indian, Mughlai          53747
North Indian                   46241
North Indian, Chinese          42012
Cafe                           30657
Chinese                        21925
North Indian, Mughlai, Chinese  20115
Fast Food                      17852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

```
[17]: cuisine_ratings = df.groupby('Cuisines')['Aggregate rating'].mean().
      ↪reset_index()
cuisine_ratings = cuisine_ratings.sort_values(by='Aggregate rating',
      ↪ascending=False)
plt.figure(figsize=(10, 5))
plt.barh(cuisine_ratings['Cuisines'][:10], cuisine_ratings['Aggregate rating'][:
      ↪10], color='skyblue')
plt.xlabel('Average Rating')
plt.ylabel('Cuisine Type')
plt.title('Top 10 Cuisines with the Highest Average Ratings')
plt.gca().invert_yaxis() # To display the highest rating at the top
plt.tight_layout()
plt.show()
```

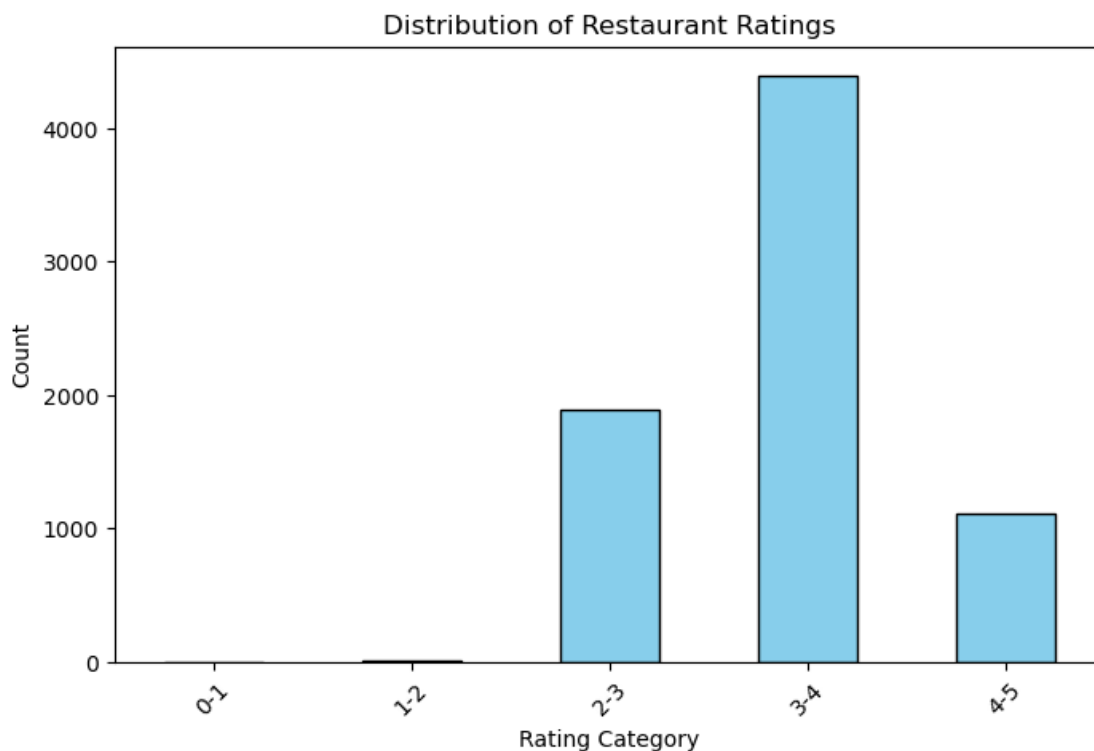


### TASK 3 : DATA VISUALIZATION

```
[18]: plt.figure(figsize=(8, 5))
plt.hist(df['Aggregate rating'], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Distribution of Restaurant Ratings')
plt.show()
```

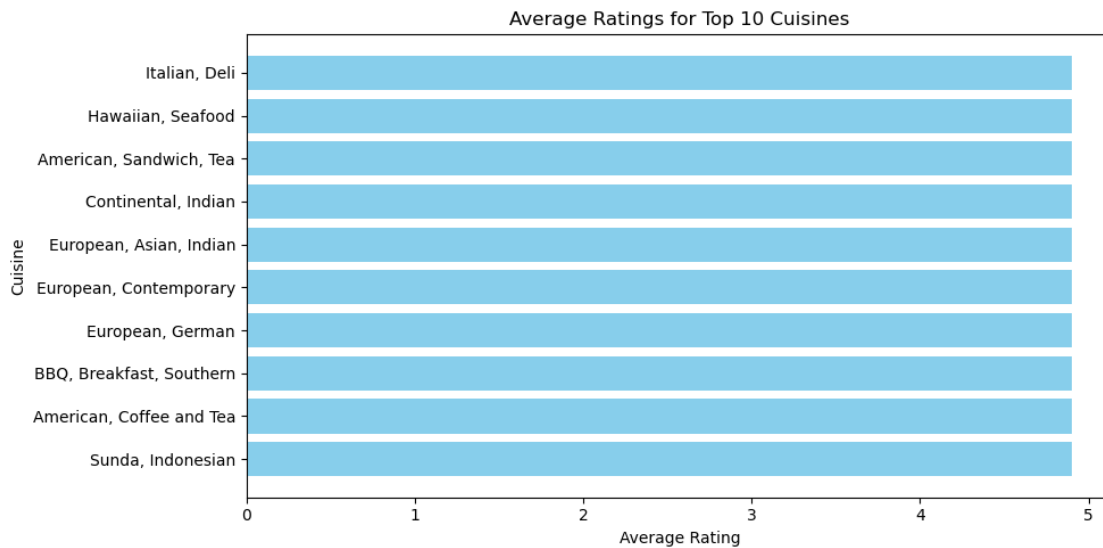


```
[19]: bins = [0, 1, 2, 3, 4, 5]
labels = ['0-1', '1-2', '2-3', '3-4', '4-5']
df['Rating Category'] = pd.cut(df['Aggregate rating'], bins=bins, labels=labels)
rating_counts = df['Rating Category'].value_counts().sort_index()
plt.figure(figsize=(8, 5))
rating_counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('Rating Category')
plt.ylabel('Count')
plt.title('Distribution of Restaurant Ratings')
plt.xticks(rotation=45)
plt.show()
```

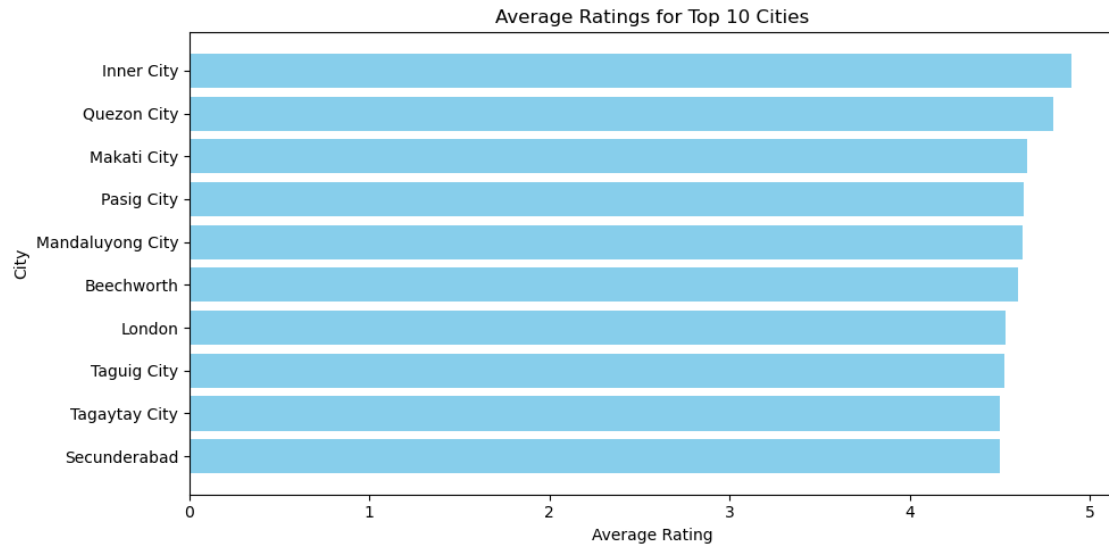


```
[20]: cuisine_ratings = df.groupby('Cuisines')['Aggregate rating'].mean().
      ↪reset_index()
cuisine_ratings = cuisine_ratings.sort_values(by='Aggregate rating',
      ↪ascending=False)
plt.figure(figsize=(10, 5))
plt.barh(cuisine_ratings['Cuisines'][:10], cuisine_ratings['Aggregate rating'][:
      ↪10], color='skyblue')
plt.ylabel('Cuisine')
```

```
plt.xlabel('Average Rating')
plt.title('Average Ratings for Top 10 Cuisines')
plt.gca().invert_yaxis() # To display the highest rating at the top
plt.tight_layout()
plt.show()
```



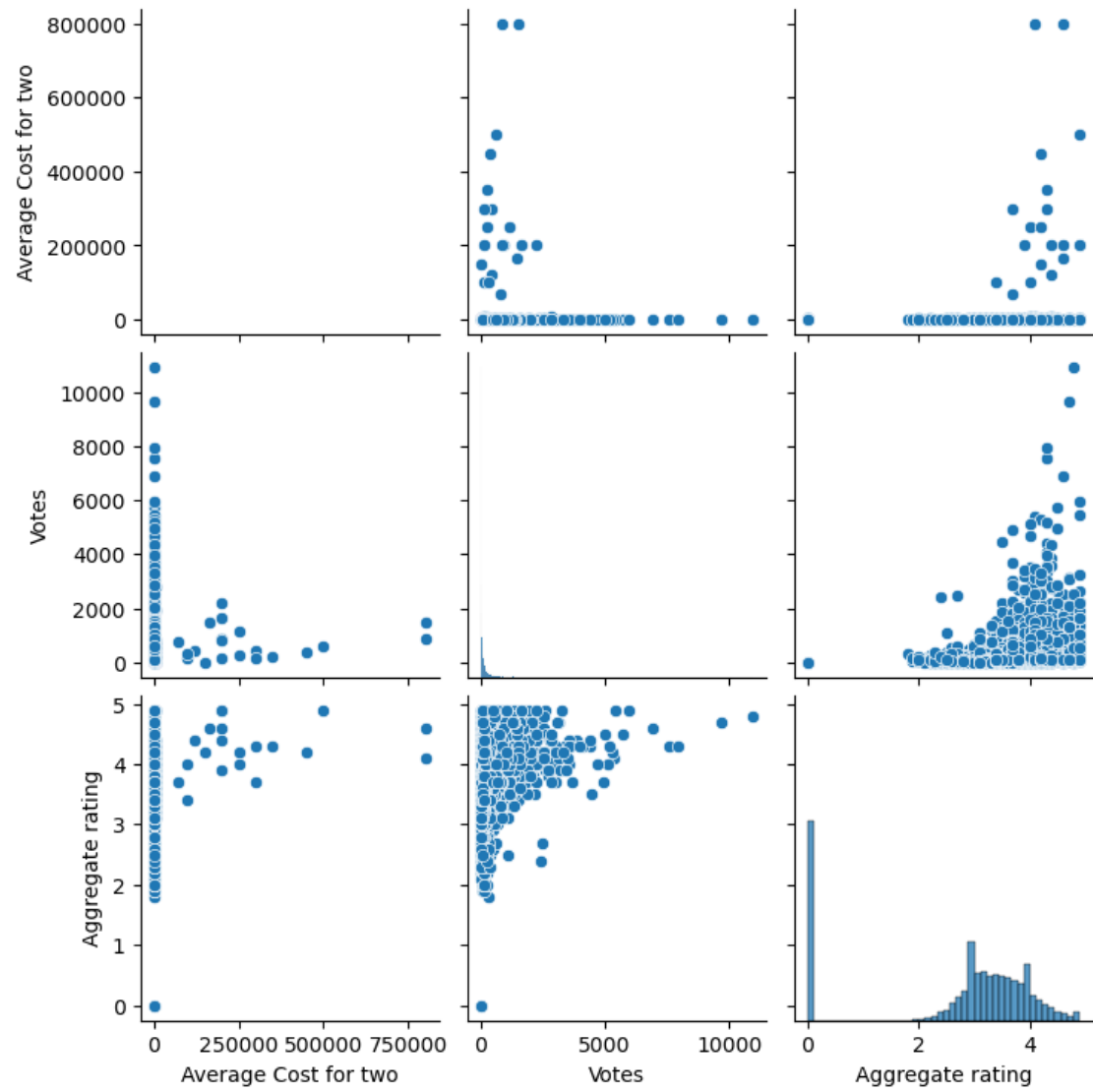
```
[21]: city_ratings = df.groupby('City')['Aggregate rating'].mean().reset_index()
city_ratings = city_ratings.sort_values(by='Aggregate rating', ascending=False)
plt.figure(figsize=(10, 5))
plt.barh(city_ratings['City'][:10], city_ratings['Aggregate rating'][:10],
         color='skyblue')
plt.ylabel('City')
plt.xlabel('Average Rating')
plt.title(f'Average Ratings for Top 10 Cities')
plt.gca().invert_yaxis() # To display the highest rating at the top
plt.tight_layout()
plt.show()
```



```
[22]: sns.pairplot(data=df, vars=['Average Cost for two', 'Votes', 'Aggregate_
    ↳rating'])
plt.suptitle("Relationship Between Features and Rating", y=1.02)
plt.show()
```



Relationship Between Features and Rating



[ ]: