

Level 3

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
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
```

```
df = pd.read_csv("Dataset .csv")
df
```

	Restaurant ID	Restaurant Name	Country Code
City \			
0	6317637	Le Petit Souffle	162
Makati City			
1	6304287	Izakaya Kikufuji	162
Makati City			
2	6300002	Heat - Edsa Shangri-La	162
Mandaluyong City			
3	6318506	Ooma	162
Mandaluyong City			
4	6314302	Sambo Kojin	162
Mandaluyong City			
...
...			
9546	5915730	Naml \ Gurme	208
00istanbul			
9547	5908749	Ceviz A00ac \	208
00istanbul			
9548	5915807	Huqqa	208
00istanbul			
9549	5916112	A000k Kahve	208
00istanbul			
9550	5927402	Walter's Coffee Roastery	208
00istanbul			

	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	Kemanke00 Karamustafa Pa00a Mahallesi, R\ht\m ...
9547	Ko00uyolu Mahallesi, Muhittin 00sto_nda00 Cadd...

9548 Kuru0_e00me Mahallesi, Muallim Naci Caddesi, N...
 9549 Kuru0_e00me Mahallesi, Muallim Naci Caddesi, N...
 9550 Cafea00a Mahallesi, Bademalt\ Sokak, No 21/B, ...

	Locality \
0	Century City Mall, Poblacion, Makati City
1	Little Tokyo, Legaspi Village, Makati City
2	Edsa Shangri-La, Ortigas, Mandaluyong City
3	SM Megamall, Ortigas, Mandaluyong City
4	SM Megamall, Ortigas, Mandaluyong City
...	...
9546	Karak0_y
9547	Ko00uyolu
9548	Kuru0_e00me
9549	Kuru0_e00me
9550	Moda

	Locality Verbose	Longitude \
0	Century City Mall, Poblacion, Makati City, Mak...	121.027535
1	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101
2	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831
3	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475
4	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508
...
9546	Karak0_y, 00stanbul	28.977392
9547	Ko00uyolu, 00stanbul	29.041297
9548	Kuru0_e00me, 00stanbul	29.034640
9549	Kuru0_e00me, 00stanbul	29.036019
9550	Moda, 00stanbul	29.026016

	Latitude	Cuisines ...	
Currency \			
0	14.565443	French, Japanese, Desserts ...	Botswana
Pula(P)			
1	14.553708	Japanese ...	Botswana
Pula(P)			
2	14.581404	Seafood, Asian, Filipino, Indian ...	Botswana
Pula(P)			
3	14.585318	Japanese, Sushi ...	Botswana
Pula(P)			
4	14.584450	Japanese, Korean ...	Botswana
Pula(P)			
...
..			
9546	41.022793	Turkish ...	Turkish
Lira(TL)			
9547	41.009847	World Cuisine, Patisserie, Cafe ...	Turkish
Lira(TL)			
9548	41.055817	Italian, World Cuisine ...	Turkish

Lira(TL)
 9549 41.057979 Restaurant Cafe ... Turkish
 Lira(TL)
 9550 40.984776 Cafe ... Turkish
 Lira(TL)

	Has Table booking	Has Online delivery	Is delivering now	\
0	Yes	No	No	
1	Yes	No	No	
2	Yes	No	No	
3	No	No	No	
4	Yes	No	No	
...	
9546	No	No	No	
9547	No	No	No	
9548	No	No	No	
9549	No	No	No	
9550	No	No	No	

	Switch to order menu	Price range	Aggregate rating	Rating color
\				
0	No	3	4.8	Dark Green
1	No	3	4.5	Dark Green
2	No	4	4.4	Green
3	No	4	4.9	Dark Green
4	No	4	4.8	Dark Green
...
9546	No	3	4.1	Green
9547	No	3	4.2	Green
9548	No	4	3.7	Yellow
9549	No	4	4.0	Green
9550	No	2	4.0	Green

	Rating text	Votes
0	Excellent	314
1	Excellent	591
2	Very Good	270
3	Excellent	365
4	Excellent	229
...

```

9546    Very Good    788
9547    Very Good   1034
9548         Good    661
9549    Very Good    901
9550    Very Good    591

```

```
[9551 rows x 21 columns]
```

```
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
```

```
#Drop rows with missing values
```

```
df.dropna(inplace=True)
```

```
#After handling missing values
```

```
df.shape
```

```
(9542, 21)
```

Task 1: Predictive Modeling

```
# Considering some potential features
selected_features = ['Average Cost for two', 'Price range', 'Votes']

X = df[selected_features]
y = df['Aggregate rating']

# Split the dataset into training and testing sets (80% train, 20%
test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Build the regression model (Linear Regression as an example)
model = LinearRegression()
model.fit(X_train, y_train)

LinearRegression()

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

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

# Mean Squared Error
mse

1.7193421524563215

# Root Mean Squared Error
rmse

1.3112368788500122

# R^2 Score
r2

0.24920612400080278

# Initialize and train different regression models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(n_estimators=100,
random_state=42)
}

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)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
results[name] = {'MSE': mse, 'RMSE': rmse, 'R^2': r2}

# Print results
print("Regression Model Performance:")
for name, metrics in results.items():
    print(f"Model: {name}")
    print(f"MSE: {metrics['MSE']:.2f}")
    print(f"RMSE: {metrics['RMSE']:.2f}")
    print(f"R^2: {metrics['R^2']:.2f}")
    print("-----")

```

```

Regression Model Performance:
Model: Linear Regression
MSE: 1.72
RMSE: 1.31
R^2: 0.25

```

```

-----
Model: Decision Tree
MSE: 0.20
RMSE: 0.44
R^2: 0.91

```

```

-----
Model: Random Forest
MSE: 0.14
RMSE: 0.37
R^2: 0.94
-----

```

Task 2: Customer Preference Analysis

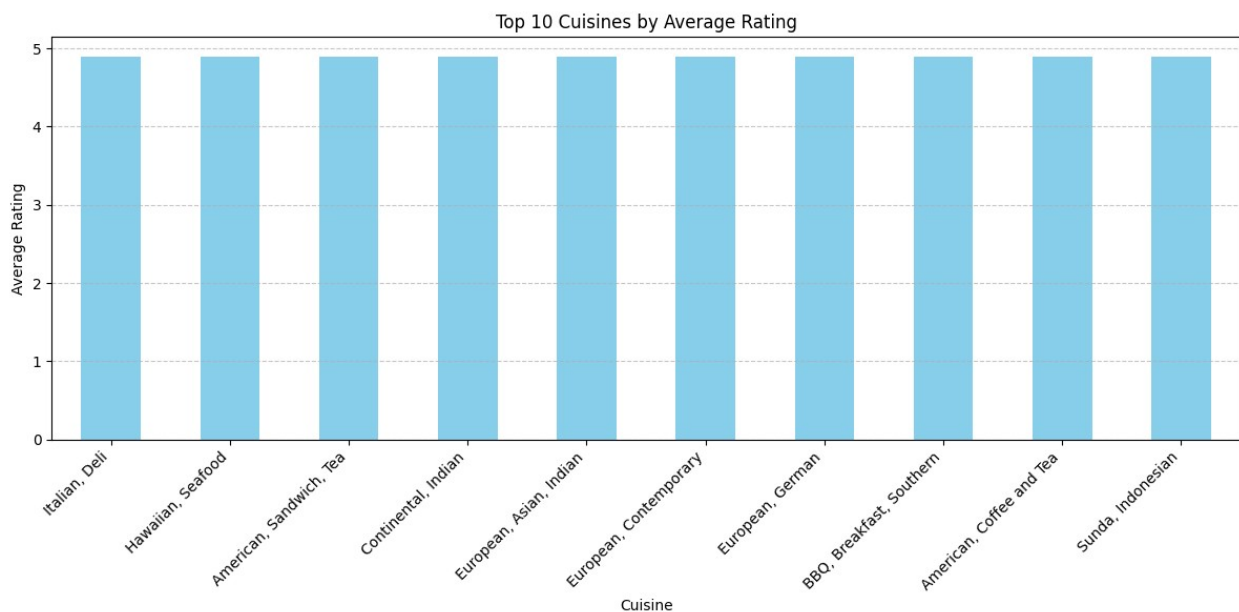
```

# Group the data by cuisine and calculate the average rating for each
cuisine
avg_rating_by_cuisine = df.groupby('Cuisines')['Aggregate
rating'].mean().sort_values(ascending=False)

# Plot the top cuisines by average rating
plt.figure(figsize=(12, 6))
avg_rating_by_cuisine.head(10).plot(kind='bar', color='skyblue')
plt.title('Top 10 Cuisines by Average Rating')
plt.xlabel('Cuisine')
plt.ylabel('Average Rating')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)

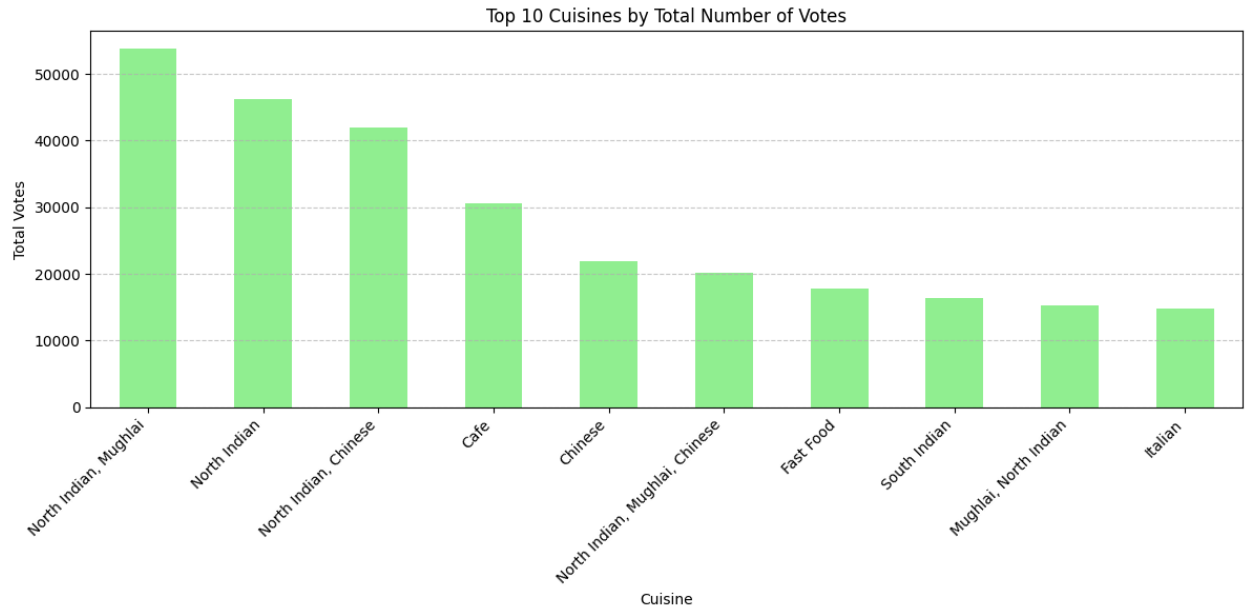
```

```
plt.tight_layout()
plt.show()
```

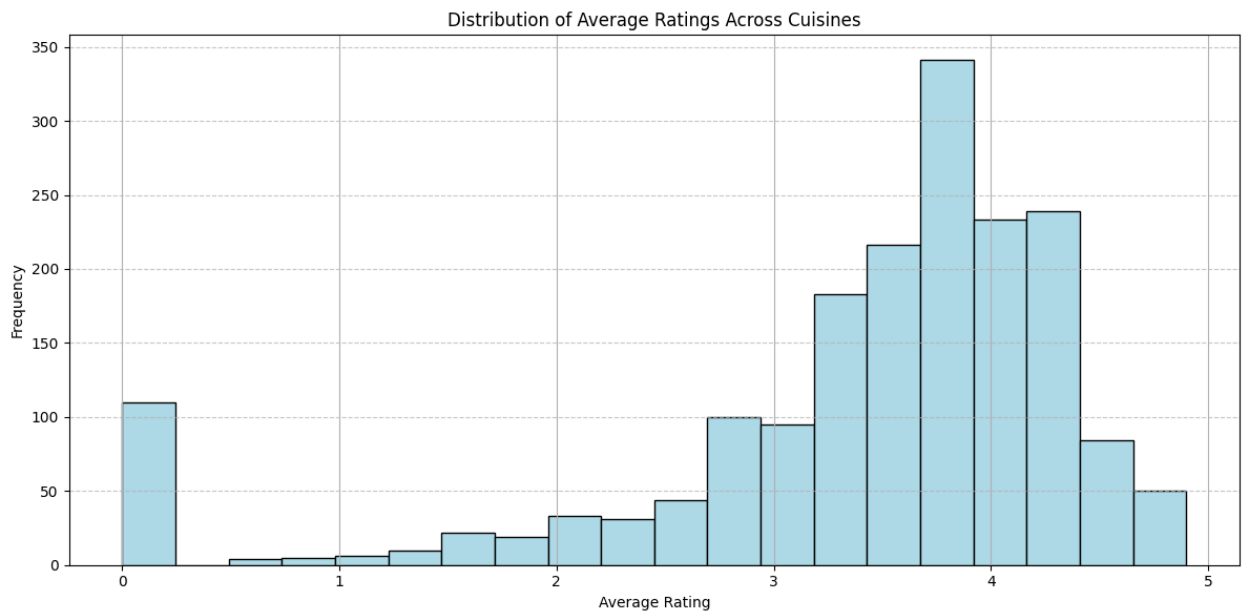


```
# Group the data by cuisine and calculate the total number of votes
for each cuisine
total_votes_by_cuisine = df.groupby('Cuisines')
['Votes'].sum().sort_values(ascending=False)

# Plot the top cuisines by total number of votes
plt.figure(figsize=(12, 6))
total_votes_by_cuisine.head(10).plot(kind='bar', color='lightgreen')
plt.title('Top 10 Cuisines by Total Number of Votes')
plt.xlabel('Cuisine')
plt.ylabel('Total Votes')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

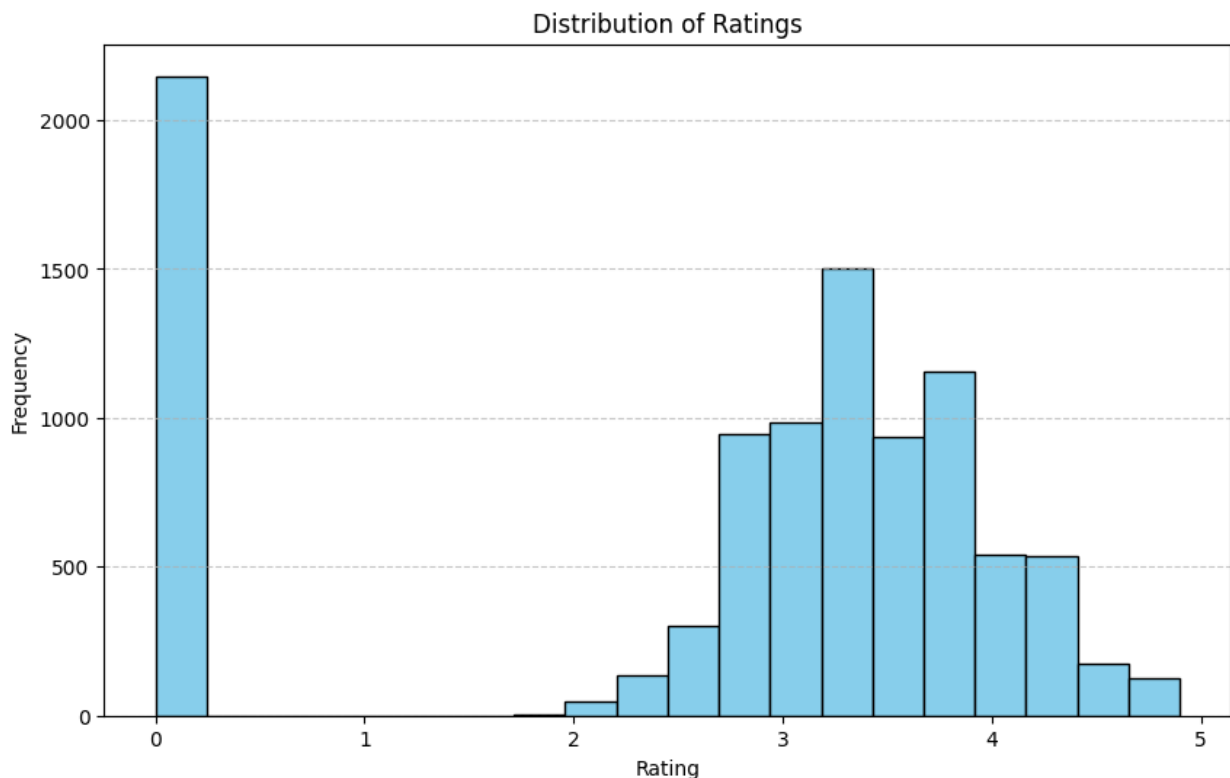


```
# Plot the distribution of average ratings across cuisines
plt.figure(figsize=(12, 6))
avg_rating_by_cuisine.hist(bins=20, color='lightblue',
edgecolor='black')
plt.title('Distribution of Average Ratings Across Cuisines')
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

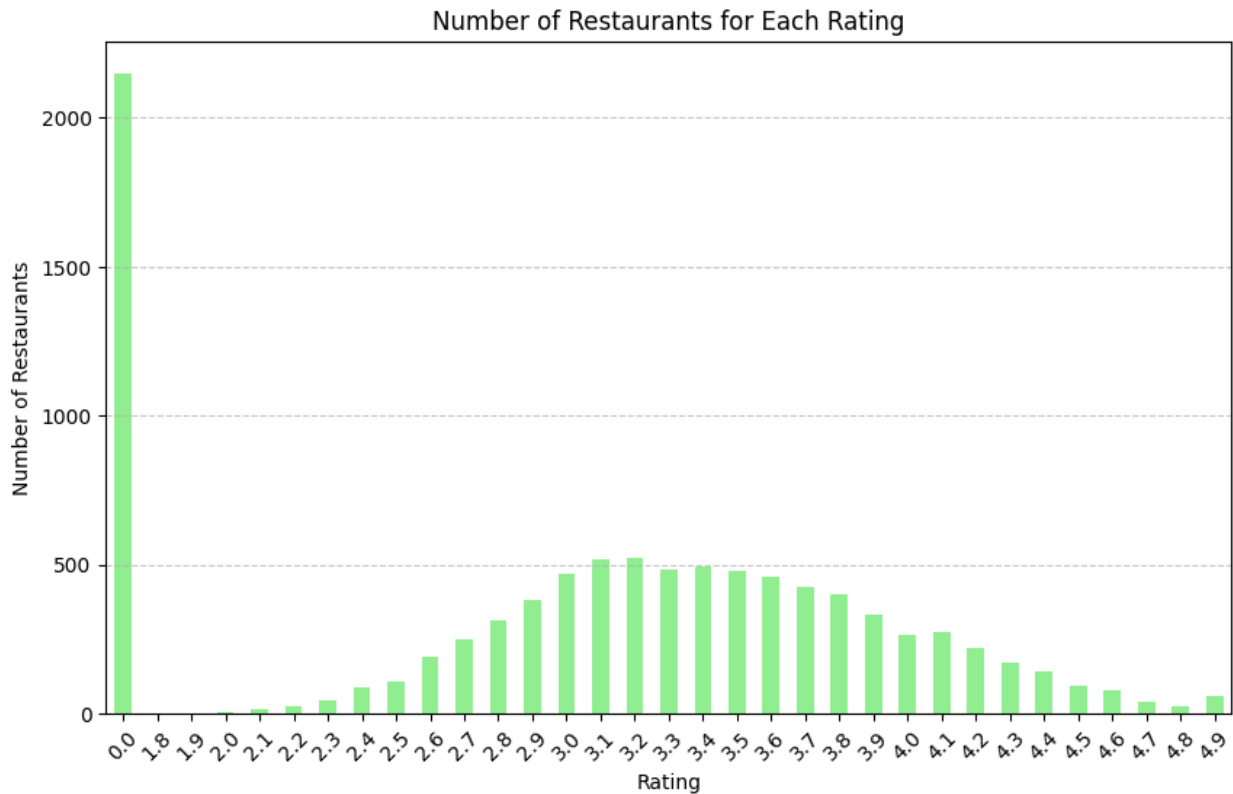


Task 3: Data Visualization

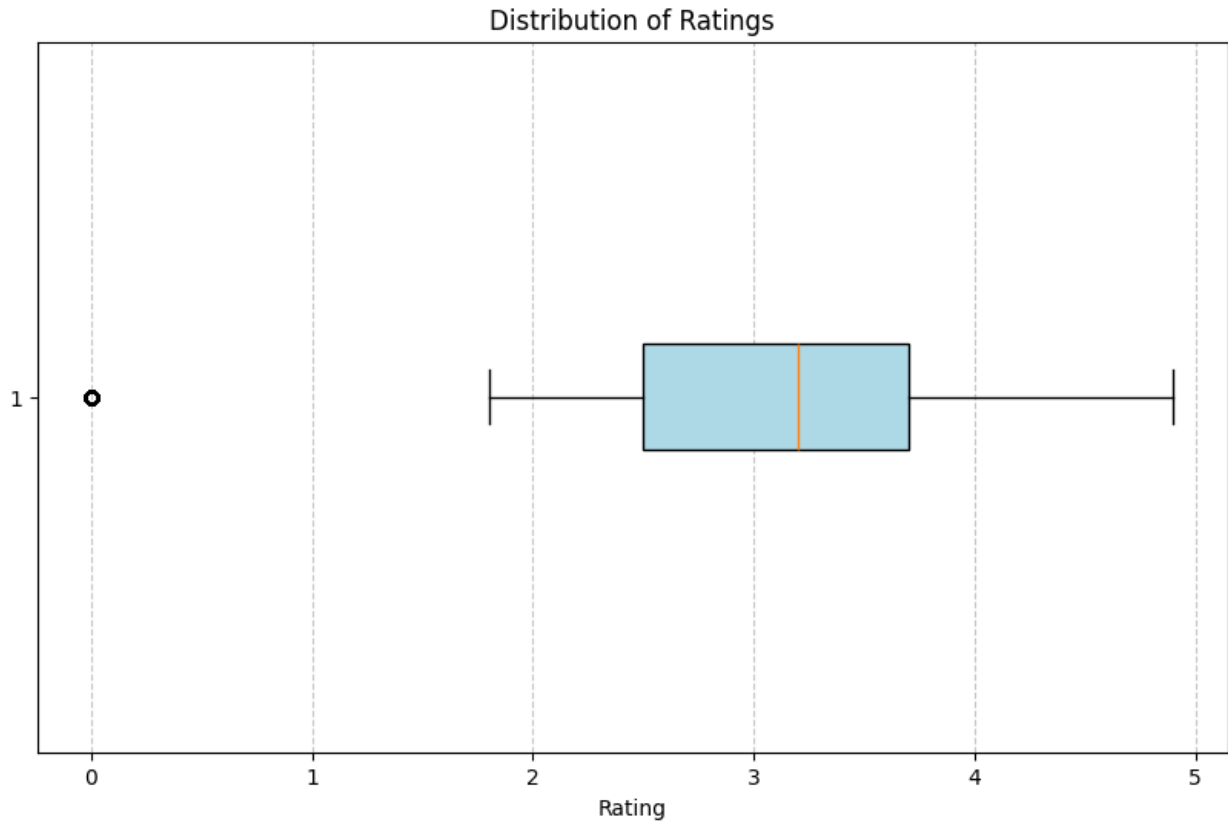
```
# Create a histogram to represent the distribution of ratings
plt.figure(figsize=(10, 6))
plt.hist(df['Aggregate rating'], bins=20, color='skyblue',
         edgecolor='black')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



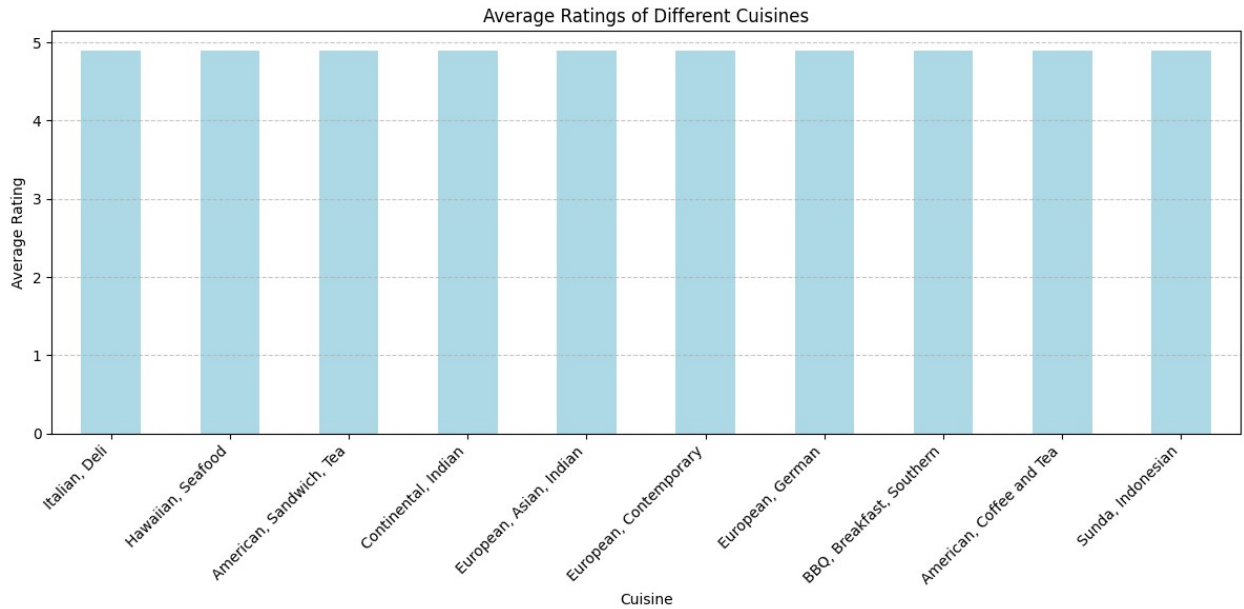
```
# Create a bar plot to represent the number of restaurants for each
rating
plt.figure(figsize=(10, 6))
df['Aggregate rating'].value_counts().sort_index().plot(kind='bar',
                                                         color='lightgreen')
plt.title('Number of Restaurants for Each Rating')
plt.xlabel('Rating')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
# Create a box plot to represent the distribution of ratings
plt.figure(figsize=(10, 6))
plt.boxplot(df['Aggregate rating'], vert=False, patch_artist=True,
            boxprops=dict(facecolor='lightblue'))
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```

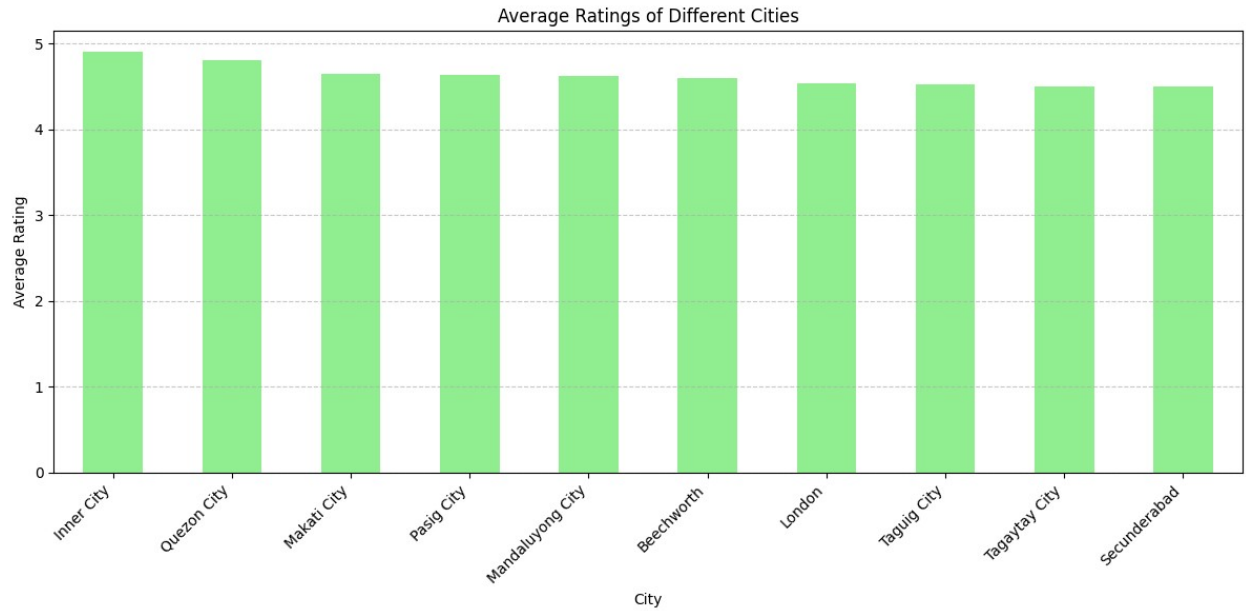


```
# Plot the average ratings of different cuisines
plt.figure(figsize=(12, 6))
avg_rating_by_cuisine.head(10).plot(kind='bar', color='lightblue')
plt.title('Average Ratings of Different Cuisines')
plt.xlabel('Cuisine')
plt.ylabel('Average Rating')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



```
# Calculate the average rating for each city
avg_rating_by_city = df.groupby('City')['Aggregate
rating'].mean().sort_values(ascending=False)

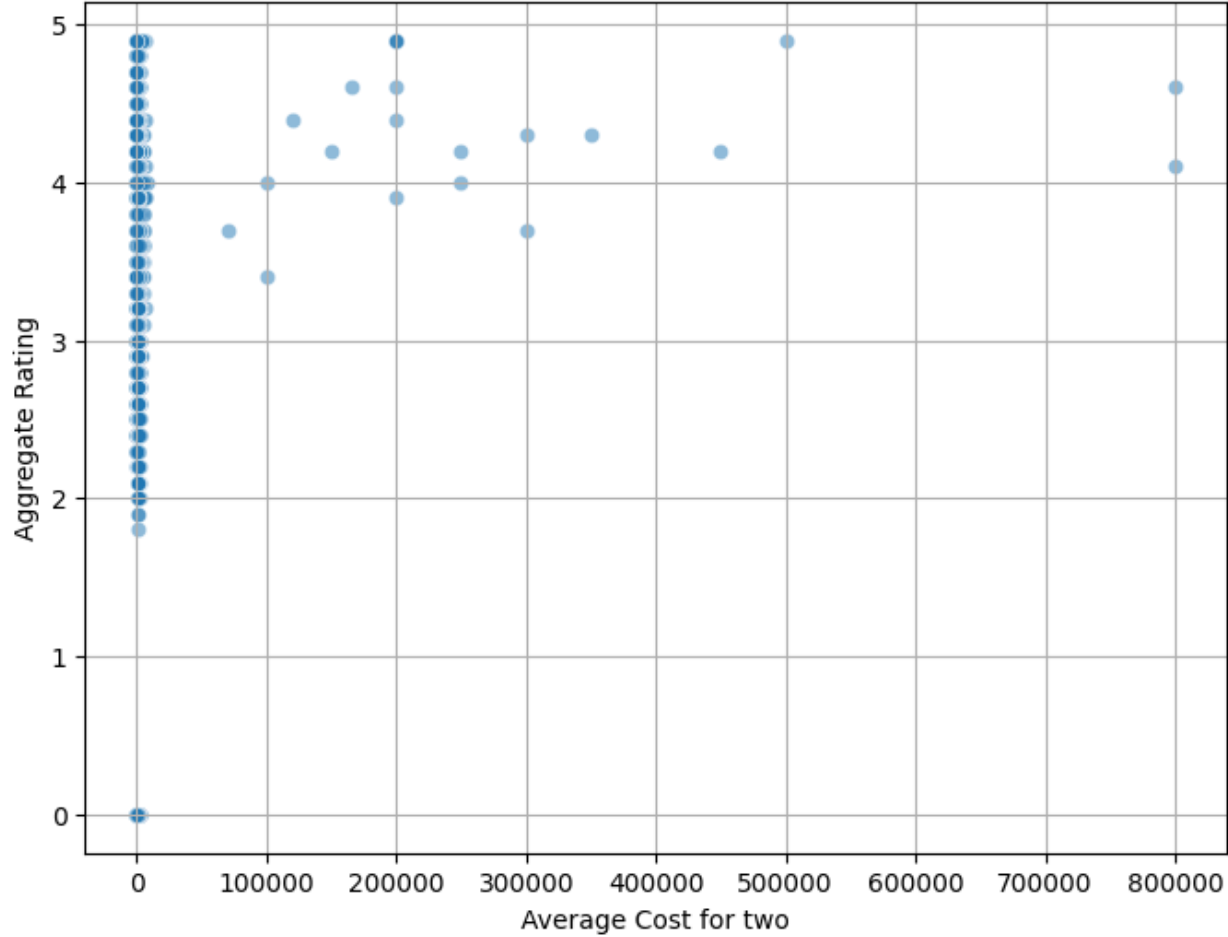
# Plot the average ratings of different cities
plt.figure(figsize=(12, 6))
avg_rating_by_city.head(10).plot(kind='bar', color='lightgreen')
plt.title('Average Ratings of Different Cities')
plt.xlabel('City')
plt.ylabel('Average Rating')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

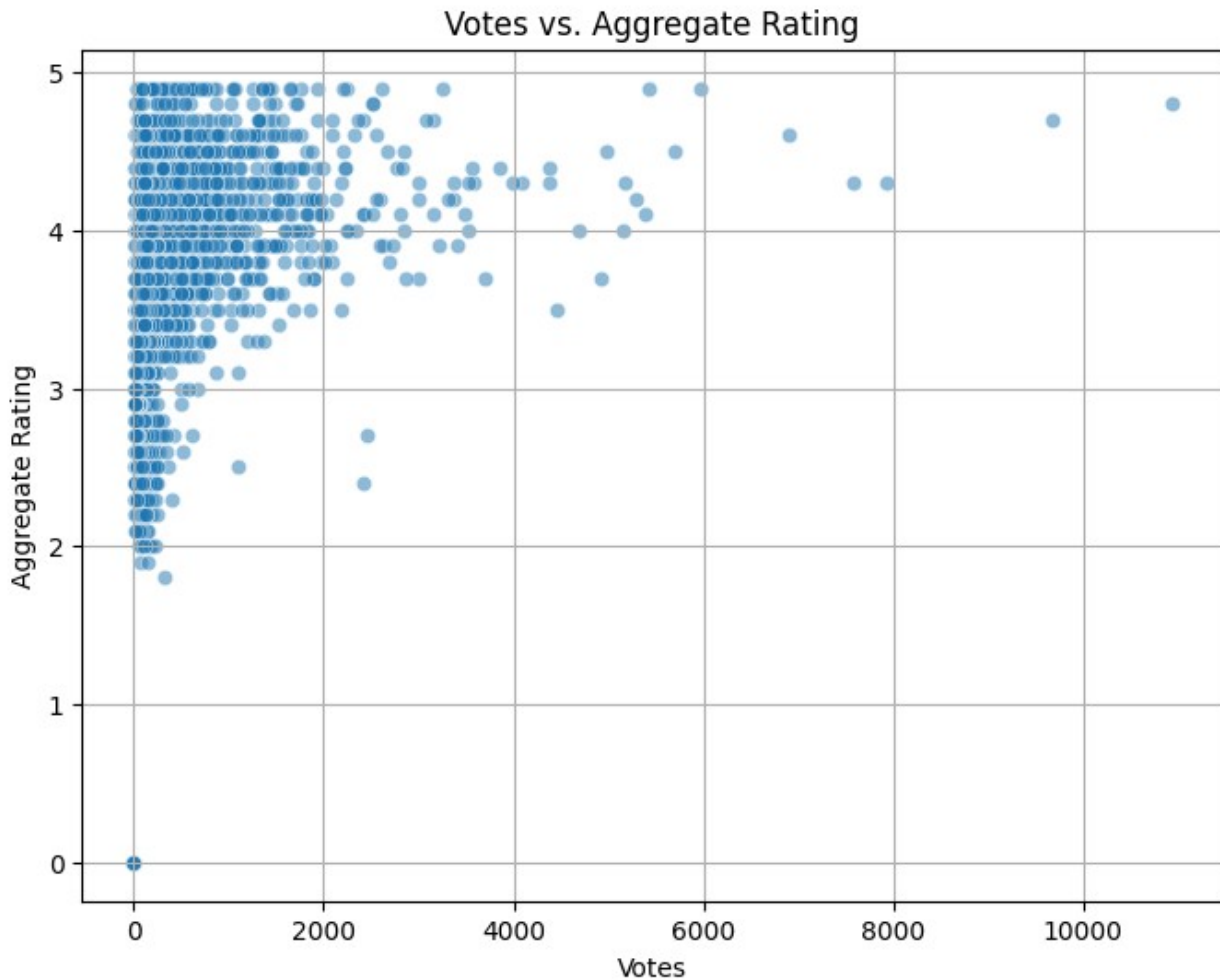


```
# Visualize numerical features vs. target variable (aggregate rating)
num_features = ['Average Cost for two', 'Votes'] # Add more numerical
features if needed

for feature in num_features:
    plt.figure(figsize=(8, 6))
    sns.scatterplot(data=df, x=feature, y='Aggregate rating',
alpha=0.5)
    plt.title(f'{feature} vs. Aggregate Rating')
    plt.xlabel(feature)
    plt.ylabel('Aggregate Rating')
    plt.grid(True)
    plt.show()
```

Average Cost for two vs. Aggregate Rating





```
# Visualize categorical features vs. target variable (aggregate
rating)
cat_features = ['Price range', 'Has Table booking', 'Has Online
delivery'] # Add more categorical features if needed

for feature in cat_features:
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df, x=feature, y='Aggregate rating')
    plt.title(f'{feature} vs. Aggregate Rating')
    plt.xlabel(feature)
    plt.ylabel('Aggregate Rating')
    plt.grid(True)
    plt.show()
```

