Level 3

Task 01: Predictive Modeling

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

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

In [2]:
```
df=pd.read_csv('Dataset .csv')
df.head()
```

Out[2]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Long
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...	121.0
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...	121.0
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...	121.0
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...	121.0
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...	121.0

5 rows × 21 columns

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]:
```
df.duplicated().sum()
```

Out[4]:
```
0
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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
```

In [6]:
```
df.describe()
```

Out[6]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	A
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	955
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	

In [7]:
```
df=pd.get_dummies(df,columns=['Has Table booking','Has Online delivery'],dr
```

In [8]:
```
df.head()
```

Out[8]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Lon
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...	121.0
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...	121.0
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...	121.0
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...	121.0
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...	121.0

5 rows × 21 columns

In [9]:
```
features=['Average Cost for two','Votes','Price range','Has Table booking_Ye
target='Aggregate rating'
X=df[features]
y=df[target]
```

In [10]:
```
X.head()
```

Out[10]:

	Average Cost for two	Votes	Price range	Has Table booking_Yes	Has Online delivery_Yes
0	1100	314	3	True	False
1	1200	591	3	True	False
2	4000	270	4	True	False
3	1500	365	4	False	False
4	1500	229	4	True	False

In [11]:
```
y.head()
```

Out[11]:
```
0    4.8
1    4.5
2    4.4
3    4.9
4    4.8
Name: Aggregate rating, dtype: float64
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, ra
```

In [13]:
```
X_train.shape,X_test.shape
```

Out[13]:
```
((7640, 5), (1911, 5))
```

In [14]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(X_train,y_train)

print('Intercept:',model.intercept_)
print('coefficient:',model.coef_)
```

```
Intercept: 1.2330394199256802
coefficient: [ 1.38686965e-06  6.72786991e-04  6.54436449e-01 -2.40525652e
-01
  6.51947553e-01]
```

In [15]:
```
train_pred=model.predict(X_train)
from sklearn.metrics import r2_score
print('Train R2:',model.score(X_train,y_train))
test_pred=model.predict(X_test)
from sklearn.metrics import mean_squared_error
print('Test MSE:',mean_squared_error(y_test,test_pred))
from sklearn.metrics import r2_score
print('Test R2:',model.score(X_test,y_test))
from sklearn.model_selection import cross_val_score
print('Cross Validation Score',cross_val_score(model,X,y,cv=5).mean())
```

```
Train R2: 0.26269853224419226
Test MSE: 1.6764802747031446
Test R2: 0.26344464090219477
Cross Validation Score -3.5495414688881306
```

In [16]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, cross_val_score

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

print('Feature importances:', model.feature_importances_)
train_pred = model.predict(X_train)
print('Train R2:', r2_score(y_train, train_pred))
test_pred = model.predict(X_test)
print('Test MSE:', mean_squared_error(y_test, test_pred))
print('Test R2:', r2_score(y_test, test_pred))
print('Cross Validation Score:', cross_val_score(model, X, y, cv=5).mean())
```

```
Feature importances: [0.01752289 0.97342699 0.0046246  0.00129971 0.003125
82]
Train R2: 0.991032937894125
Test MSE: 0.20589155960334507
Test R2: 0.9095423108120364
Cross Validation Score: 0.8970734273110328
```

In [17]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, cross_val_score

model = RandomForestRegressor()
model.fit(X_train, y_train)
print('Feature importances:', model.feature_importances_)
print('Number of trees in the forest:', len(model.estimators_))

train_pred = model.predict(X_train)
print('Train R2:', r2_score(y_train, train_pred))
test_pred = model.predict(X_test)
print('Test MSE:', mean_squared_error(y_test, test_pred))
print('Test R2:', r2_score(y_test, test_pred))
print('Cross Validation Score:', cross_val_score(model, X, y, cv=5).mean())
```

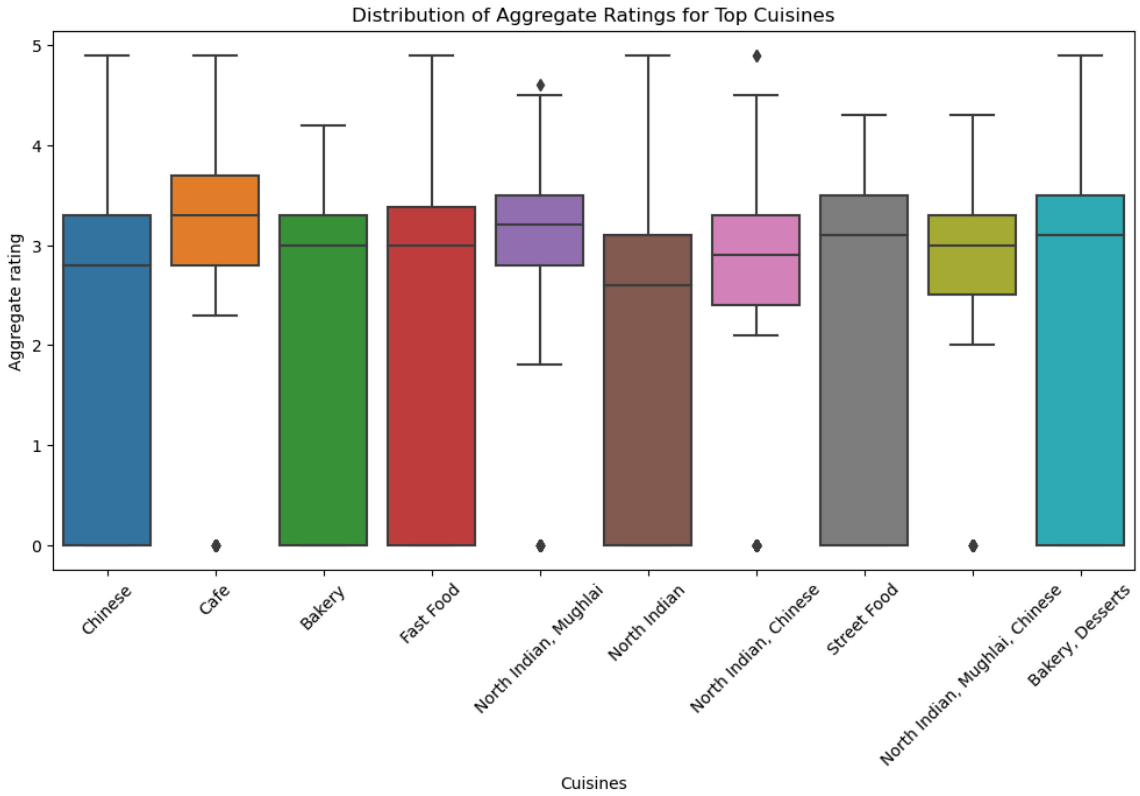
```
Feature importances: [0.01846568 0.97135586 0.00485877 0.00175649 0.003563
19]
Number of trees in the forest: 100
Train R2: 0.985430189785516
Test MSE: 0.13431278311617523
Test R2: 0.9409901794298907
Cross Validation Score: 0.9308511151708372
```

Task 02: Customer Preference Analysis

In [18]:
```
df.columns
```

Out[18]:
```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Addres
s',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisine
s',
       'Average Cost for two', 'Currency', 'Is delivering now',
       'Switch to order menu', 'Price range', 'Aggregate rating',
       'Rating color', 'Rating text', 'Votes', 'Has Table booking_Yes',
       'Has Online delivery_Yes'],
      dtype='object')
```

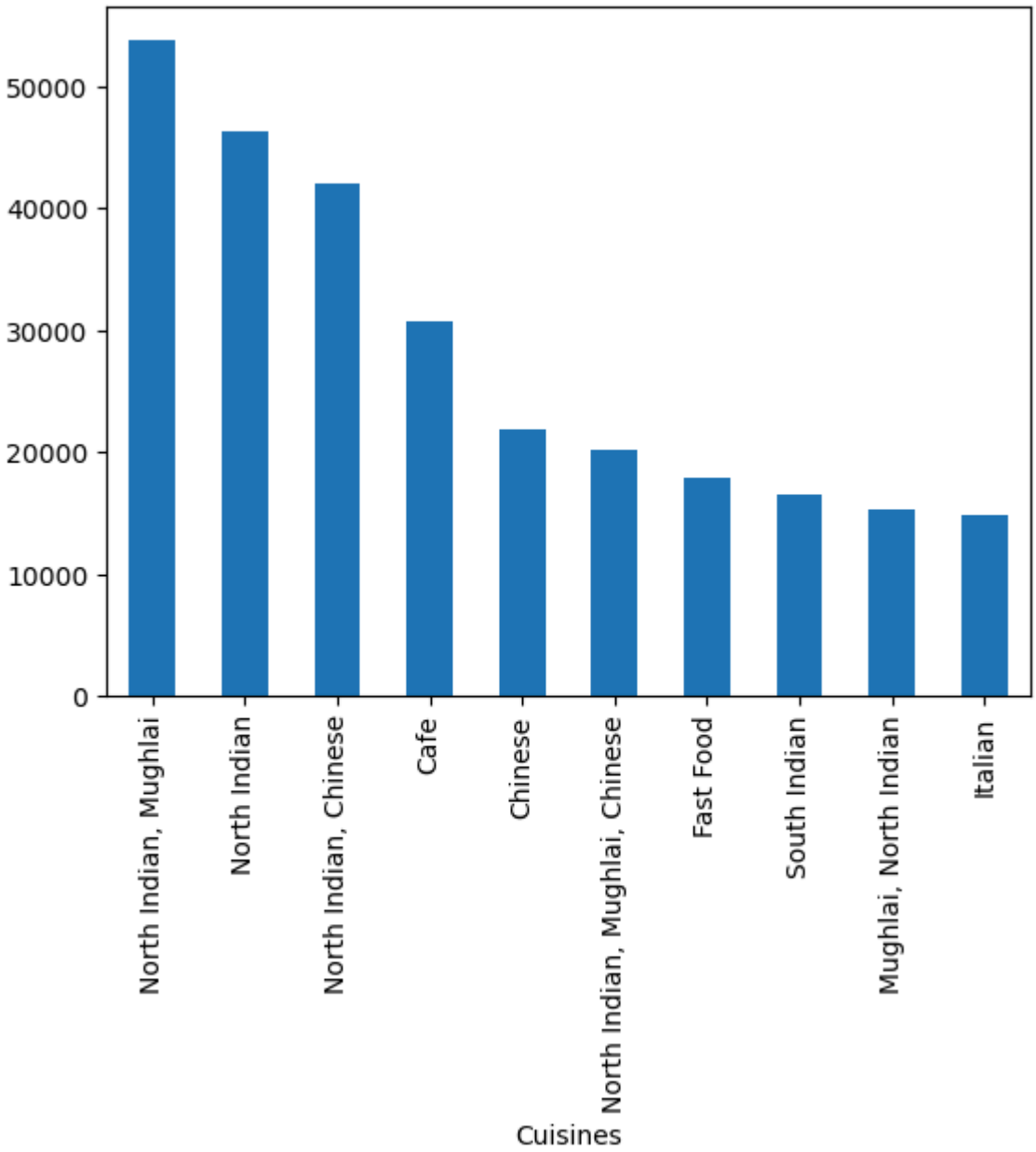
In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns

top_n = 10
top_cuisines = df['Cuisines'].value_counts().nlargest(top_n).index
df_filtered = df[df['Cuisines'].isin(top_cuisines)]

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_filtered, x='Cuisines', y='Aggregate rating')
plt.xticks(rotation=45)
plt.title('Distribution of Aggregate Ratings for Top Cuisines')
plt.show()
```



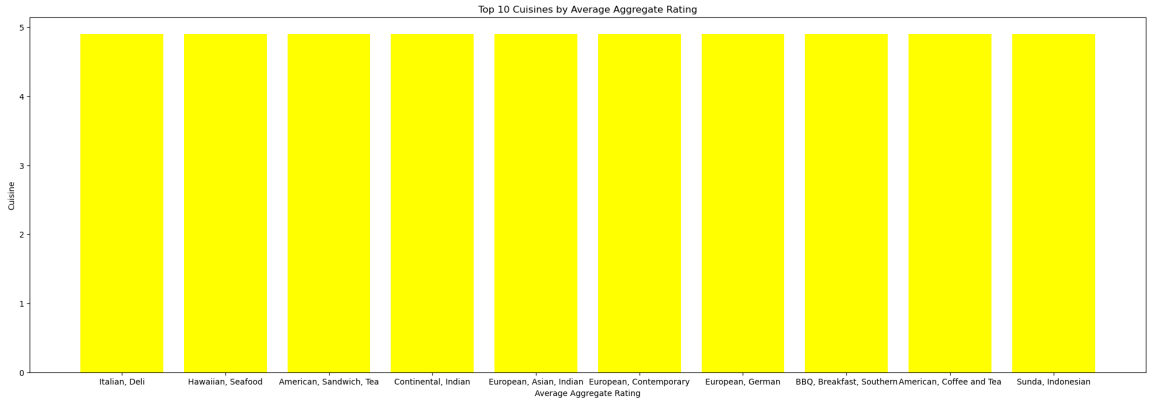
In [20]:
```
top_cuisines=df.groupby('Cuisines')['Votes'].sum().nlargest(10)
top_cuisines
```

Out[20]:
```
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
```

In [21]:
```
top_cuisines.plot(kind='bar')
plt.show()
```



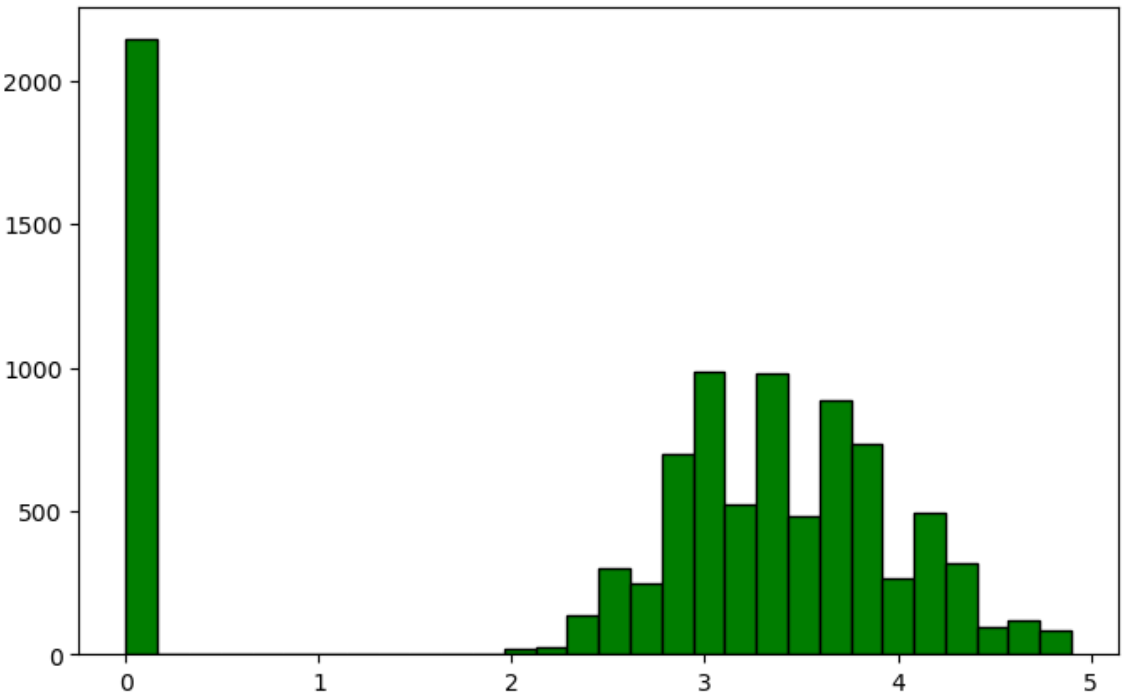
In [22]:
```
cuisine_rating = df.groupby('Cuisines')['Aggregate rating'].mean().reset_inc
cuisine_rating = cuisine_rating.sort_values(by='Aggregate rating', ascending
plt.figure(figsize=(25, 8))
plt.bar(cuisine_rating['Cuisines'][:10], cuisine_rating['Aggregate rating']
plt.xlabel('Average Aggregate Rating')
plt.ylabel('Cuisine')
plt.title('Top 10 Cuisines by Average Aggregate Rating')
plt.show()
```



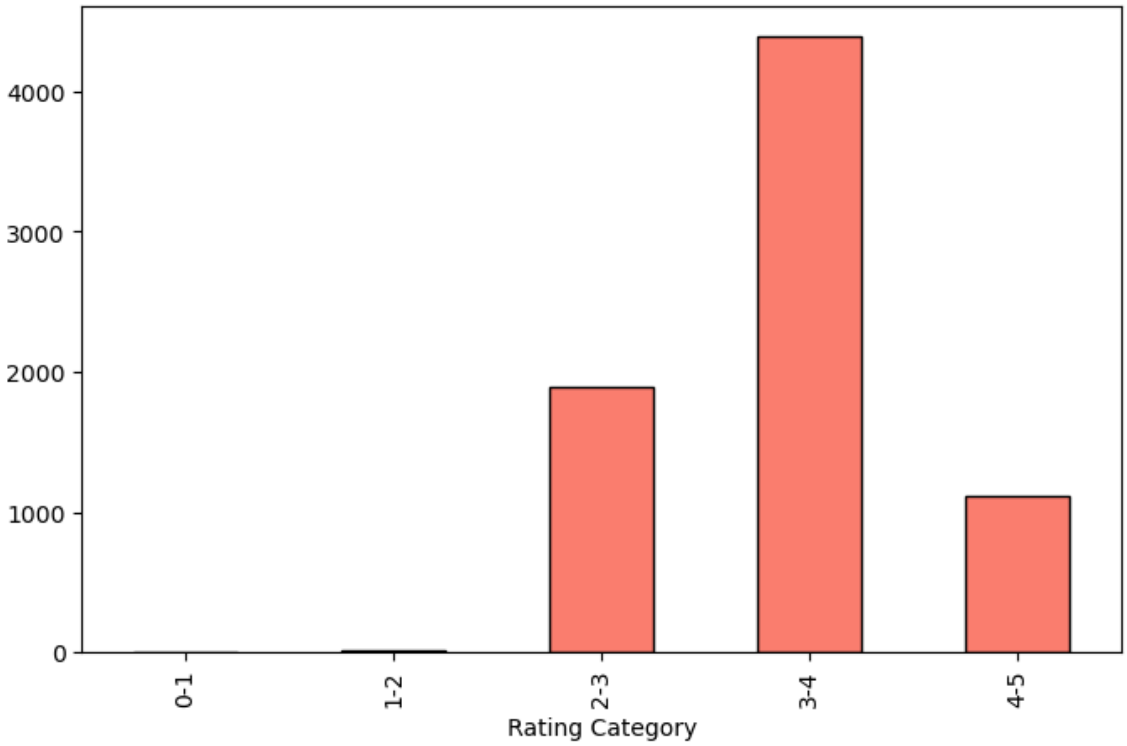
Task 03: Data Visualization

Type *Markdown* and LaTeX: α^2

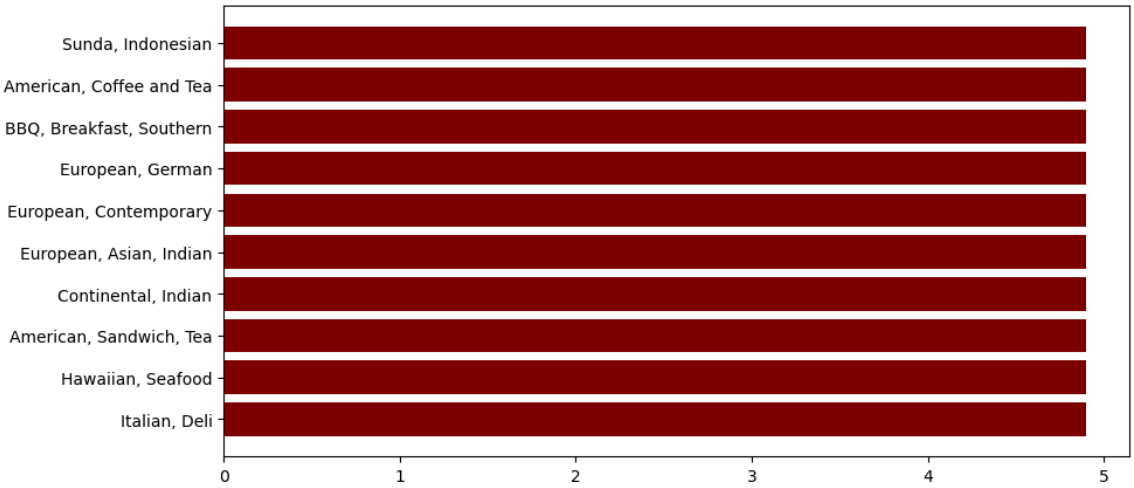
In [23]:
```
plt.figure(figsize=(8,5))
plt.hist(df['Aggregate rating'],bins=30,color='green',edgecolor='black')
plt.show()
```



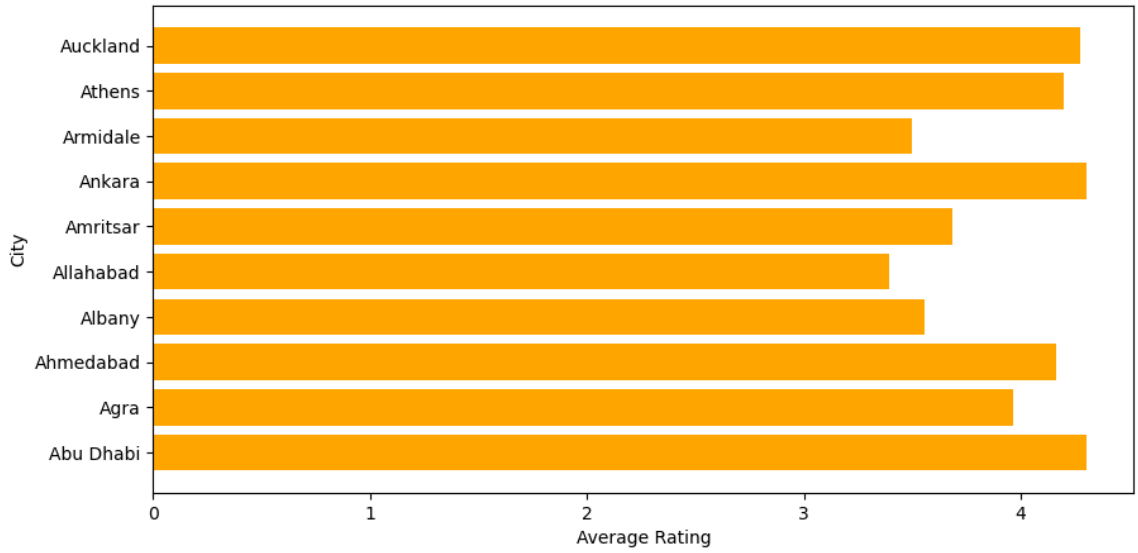
In [24]:
```
bins=[0,1,2,3,4,5]
labels=['0-1','1-2','2-3','3-4','4-5']
df['Rating Category']=pd.cut(df['Aggregate rating'],bins=bins,labels=labels)
rating_counts=df['Rating Category'].value_counts().sort_index()
plt.figure(figsize=(8,5))
rating_counts.plot(kind='bar',color='salmon',edgecolor='black')
plt.show()
```



In [25]:
```
cuisine_rating=df.groupby('Cuisines')['Aggregate rating'].mean().reset_index()
cuisine_rating=cuisine_rating.sort_values(by='Aggregate rating',ascending=False)
plt.figure(figsize=(10,5))
plt.barh(cuisine_rating['Cuisines'][:10],cuisine_rating['Aggregate rating'])
plt.show()
```



In [29]:
```
city_ratings=df.groupby('City')['Aggregate rating'].mean().reset_index()
plt.figure(figsize=(10,5))
plt.barh(city_ratings['City'][:10],city_ratings['Aggregate rating'][:10],co
plt.xlabel('Average Rating')
plt.ylabel('City')
plt.show()
```



In [32]:
```
sns.pairplot(data=df, vars=['Average Cost for two','Votes','Aggregate rating
plt.suptitle('Relationship Between Features and Rating')
plt.show()
```