Project 4 - EDA

Submitted by:

Abdullah Saeed Abbasi

Designing a Marketing Campaign for a restaurant Chain Using Exploratory Data Analysis

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

In [2]:
```
df1 = pd.read_csv('C:\\Users\\avata\\Desktop\\New folder\\Portfolio Project\\zomato_restaurants.csv')

df = df1.copy()
```

Data Cleaning and Preparation:

Identify and handle missing values.

Detect and correct any inconsistencies in the dataset (e.g., data types,mislabeled categories).

Feature engineering (if necessary), like extracting useful information from existing data.

In [3]:
```
df1.dtypes
```

Out[3]:
```
res_id                    int64
name                     object
establishment            object
url                      object
address                  object
city                     object
city_id                   int64
locality                 object
latitude                float64
longitude               float64
zipcode                  object
country_id                int64
locality_verbose         object
cuisines                 object
timings                  object
average_cost_for_two      int64
price_range               int64
currency                 object
highlights               object
aggregate_rating        float64
rating_text              object
votes                     int64
photo_count               int64
opentable_support       float64
delivery                  int64
takeaway                  int64
dtype: object
```

In [5]:
```
df
```

Out[5]:

	res_id	name	establishment	url	address	city	city_id	locality	latitu
0	3400299	Bikanervala	['Quick Bites']	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	Agra	34	Khandari	27.2114
1	3400005	Mama Chicken Mama Franky House	['Quick Bites']	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra	Agra	34	Agra Cantt	27.1605
2	3401013	Bhagat Halwai	['Quick Bites']	https://www.zomato.com/agra/bhagat-halwai-2-sh...	62/1, Near Easy Day, West Shivaji Nagar, Goalp...	Agra	34	Shahganj	27.1829
3	3400290	Bhagat Halwai	['Quick Bites']	https://www.zomato.com/agra/bhagat-halwai-civi...	Near Anjana Cinema, Nehru Nagar, Civil Lines, ...	Agra	34	Civil Lines	27.2056
4	3401744	The Salt Cafe Kitchen & Bar	['Casual Dining']	https://www.zomato.com/agra/the-salt-cafe-kitc...	1C,3rd Floor, Fatehabad Road, Tajganj, Agra	Agra	34	Tajganj	27.1577
...	...	...	...	...	...	...	...	...	
211939	3202251	Kali Mirch Cafe And Restaurant	['Casual Dining']	https://www.zomato.com/vadodara/kali-mirch-caf...	Manu Smriti Complex, Near Navrachna School, Gl...	Vadodara	32	Fatehgunj	22.3369

	res_id	name	establishment	url	address	city	city_id	locality	latitu
211940	3200996	Raju Omlet	['Quick Bites']	https://www.zomato.com/vadodara/raju-omlet-kar...	Mahalaxmi Apartment, Opposite B O B, Karoli Ba...	Vadodara	32	Karelibaug	22.3224
211941	18984164	The Grand Thakar	['Casual Dining']	https://www.zomato.com/vadodara/the-grand-thak...	3rd Floor, Shreem Shalini Mall, Opposite Conqu...	Vadodara	32	Alkapuri	22.3105
211942	3201138	Subway	['Quick Bites']	https://www.zomato.com/vadodara/subway-1-akota...	G-2, Vedant Platina, Near Cosmos, Akota, Vadodara	Vadodara	32	Akota	22.2700
211943	18879846	Freshco's - The Health Cafe	['Café']	https://www.zomato.com/vadodara/freshcos-the-h...	Shop 7, Ground Floor, Opposite Natubhai Circle...	Vadodara	32	Vadiwadi	22.3099

211944 rows × 26 columns

In [6]:
```
# Checking types
df.dtypes
```

Out[6]:
```
res_id                    int64
name                     object
establishment            object
url                      object
address                  object
city                     object
city_id                   int64
locality                 object
latitude                float64
longitude               float64
zipcode                  object
country_id                int64
locality_verbose         object
cuisines                 object
timings                  object
average_cost_for_two      int64
price_range               int64
currency                 object
highlights               object
aggregate_rating        float64
rating_text              object
votes                     int64
photo_count               int64
opentable_support       float64
delivery                  int64
takeaway                  int64
dtype: object
```

In [7]:
```
df.shape
```

Out[7]:
```
(211944, 26)
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['res_id', 'name', 'establishment', 'url', 'address', 'city', 'city_id',
       'locality', 'latitude', 'longitude', 'zipcode', 'country_id',
       'locality_verbose', 'cuisines', 'timings', 'average_cost_for_two',
       'price_range', 'currency', 'highlights', 'aggregate_rating',
       'rating_text', 'votes', 'photo_count', 'opentable_support', 'delivery',
       'takeaway'],
      dtype='object')
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
res_id                       0
name                         0
establishment                0
url                          0
address                    134
city                         0
city_id                      0
locality                     0
latitude                     0
longitude                    0
zipcode                 163187
country_id                   0
locality_verbose             0
cuisines                  1391
timings                   3874
average_cost_for_two         0
price_range                  0
currency                     0
highlights                   0
aggregate_rating             0
rating_text                  0
votes                        0
photo_count                  0
opentable_support           48
delivery                     0
takeaway                     0
dtype: int64
```

Duplicate Checking

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
151527
```

In [11]:
```
# Syntax: df.duplicated(subset=None, keep='first')
# Returns: Series of booleans indicating duplicate rows

# Example: Check for duplicate rows in the DataFrame
duplicate_rows = df.duplicated()

# Count the number of duplicate rows
num_duplicates = duplicate_rows.sum()

# Print the number of duplicate rows
print("Number of duplicate rows:", num_duplicates)

# Optional: Display the duplicate rows
duplicate_data = df[duplicate_rows]
print("Duplicate rows:")
print(duplicate_data)
```

```
Number of duplicate rows: 151527
Duplicate rows:
          res_id                          name      establishment  \
101      3400059          Peshawri - ITC Mughal    ['Fine Dining']
116      3400060          Taj Bano - ITC Mughal    ['Fine Dining']
140      3400017                 Pinch Of Spice  ['Casual Dining']
141      3400018                 Pinch Of Spice  ['Casual Dining']
142      3400850                     Urban Deck  ['Casual Dining']
...          ...                            ...                ...
211937  18855810             Biryani aur Baatein  ['Casual Dining']
211938  18662583                   Wok On Fire  ['Casual Dining']
211939   3202251  Kali Mirch Cafe And Restaurant  ['Casual Dining']
211941  18984164                The Grand Thakar  ['Casual Dining']
211943  18879846     Freshco's - The Health Cafe           ['Café']

                                                      url  \
101     https://www.zomato.com/agra/peshawri-itc-mugha... (https://www.zomato.com/agra/peshawri-itc-mugh
a...)
116     https://www.zomato.com/agra/taj-bano-itc-mugha... (https://www.zomato.com/agra/taj-bano-itc-mugh
a...)
140     https://www.zomato.com/agra/pinch-of-spice-civ... (https://www.zomato.com/agra/pinch-of-spice-ci
v...)
141     https://www.zomato.com/agra/pinch-of-spice-taj... (https://www.zomato.com/agra/pinch-of-spice-ta
j...)
142     https://www.zomato.com/agra/urban-deck-2-civil... (https://www.zomato.com/agra/urban-deck-2-civi
l...)
...                                                   ...
211937  https://www.zomato.com/vadodara/biryani-aur-ba... (https://www.zomato.com/vadodara/biryani-aur-b
a...)
211938  https://www.zomato.com/vadodara/wok-on-fire-fa... (https://www.zomato.com/vadodara/wok-on-fire-f
a...)
211939  https://www.zomato.com/vadodara/kali-mirch-caf... (https://www.zomato.com/vadodara/kali-mirch-ca
f...)
211941  https://www.zomato.com/vadodara/the-grand-thak... (https://www.zomato.com/vadodara/the-grand-tha
k...)
211943  https://www.zomato.com/vadodara/freshcos-the-h... (https://www.zomato.com/vadodara/freshcos-the-
h...)

                                                 address      city  city_id  \
101               ITC Mughal, Fatehabad Road, Tajganj, Agra      Agra       34
116               ITC Mughal, Fatehabad Road, Tajganj, Agra      Agra       34
140     23/453, Opposite Sanjay Cinema, Wazipura Road,...      Agra       34
141                   1076/2, Fatehabad Road, Tajganj, Agra      Agra       34
```

```
142     5th Floor, The P L Palace Hotel, MG Road, Sanj...      Agra       34
...                                                   ...       ...      ...
211937  Shop 14, Atlantis K-10, A Wing, Genda Circle R...  Vadodara       32
211938  Ground Floor 1, Rossette Building, Opposite Se...  Vadodara       32
211939  Manu Smriti Complex, Near Navrachna School, GI...  Vadodara       32
211941  3rd Floor, Shreem Shalini Mall, Opposite Conqu...  Vadodara       32
211943  Shop 7, Ground Floor, Opposite Natubhai Circle...  Vadodara       32

                   locality   latitude  longitude  ... price_range  currency  \
101     ITC Mughal, Tajganj  27.161150  78.043993  ...           4       Rs.
116     ITC Mughal, Tajganj  27.161132  78.044022  ...           4       Rs.
140             Civil Lines  27.201735  78.007625  ...           4       Rs.
141                 Tajganj  27.159649  78.043304  ...           4       Rs.
142             Civil Lines  27.199573  78.003699  ...           4       Rs.
...                     ...        ...        ...  ...         ...       ...
211937             Alkapuri  22.317746  73.168043  ...           2       Rs.
211938            Fatehgunj  22.323357  73.187461  ...           3       Rs.
211939            Fatehgunj  22.336931  73.192356  ...           2       Rs.
211941             Alkapuri  22.310563  73.171163  ...           2       Rs.
211943             Vadiwadi  22.309935  73.158768  ...           2       Rs.

                                               highlights aggregate_rating  \
101     ['Lunch', 'Cash', 'Credit Card', 'Dinner', 'De...              4.4
116     ['Credit Card', 'Lunch', 'Cash', 'Debit Card',...              4.3
140     ['Lunch', 'Delivery', 'Credit Card', 'Dinner',...              4.6
141     ['Delivery', 'Dinner', 'Cash', 'Credit Card', ...              4.6
142     ['Dinner', 'Cash', 'Debit Card', 'Takeaway Ava...              4.3
...                                                   ...              ...
211937  ['Dinner', 'Cash', 'Takeaway Available', 'Debi...              4.1
211938  ['Dinner', 'Cash', 'Debit Card', 'Lunch', 'Tak...              4.0
211939  ['Dinner', 'Cash', 'Lunch', 'Delivery', 'Indoo...              4.1
211941  ['Dinner', 'Cash', 'Debit Card', 'Lunch', 'Tak...              4.0
211943  ['Dinner', 'Cash', 'Takeaway Available', 'Debi...              4.0

       rating_text  votes  photo_count opentable_support delivery  takeaway
101      Very Good    353          154               0.0       -1        -1
116      Very Good     96          205               0.0       -1        -1
140      Excellent    915          105               0.0        1        -1
141      Excellent    965          690               0.0        1        -1
142      Very Good    672          192               0.0        1        -1
...            ...    ...          ...               ...      ...       ...
211937   Very Good    154           96               0.0       -1        -1
211938   Very Good    301          126               0.0        1        -1
```

```
211939   Very Good    243           40               0.0       -1         -1
211941   Very Good    111           38               0.0       -1         -1
211943   Very Good     93           53               0.0        1         -1

[151527 rows x 26 columns]
```

Removing Duplicate and Irrelevant Columns

In [12]:
```
df.drop_duplicates(inplace=True)
df.drop(columns=['currency', 'zipcode'], inplace=True)
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
res_id                    0
name                      0
establishment             0
url                       0
address                  18
city                      0
city_id                   0
locality                  0
latitude                  0
longitude                 0
country_id                0
locality_verbose          0
cuisines                470
timings                1070
average_cost_for_two      0
price_range               0
highlights                0
aggregate_rating          0
rating_text               0
votes                     0
photo_count               0
opentable_support        19
delivery                  0
takeaway                  0
dtype: int64
```

Missing Values Catering

Cusinies & opentable_support

In [14]:
```
df['cuisines'].fillna('Unknown', inplace=True)

df['opentable_support'].fillna(0, inplace=True)
```

Address

We have Longitude and Latitude, by this we can calculate address

In [15]:
```
from geopy.geocoders import Nominatim

# Create a geocoder object
geolocator = Nominatim(user_agent="restaurant_geocoder")

# Function to get address from latitude and longitude
def get_address(lat, lon):
    location = geolocator.reverse((lat, lon), timeout=10)  # Increase timeout to 10 seconds
    return location.address if location else None

# Assuming df is your DataFrame and 'latitude', 'longitude', and 'address' are column names
# Fill missing addresses based on latitude and longitude
df['address'] = df.apply(lambda row: row['address'] if pd.notnull(row['address']) else get_address(row['latit
```

Timings

Few restaruants names are same, so we will first use those to fill the timings and then remove duplicate again and then will remove the remaining missing values

In [16]:
```
def fill_missing_timings(group):
    try:
        mode_value = group.mode().iloc[0]
        return group.fillna(mode_value)
    except IndexError:
        return group

df['timings'] = df.groupby('name')['timings'].transform(fill_missing_timings)
```

In [17]:
```
df.duplicated().sum()
```

Out[17]: 6

In [18]:
```
df.drop_duplicates(inplace=True)
```

In [19]:
```
df.isnull().sum()
```

Out[19]:
```
res_id                    0
name                      0
establishment             0
url                       0
address                   0
city                      0
city_id                   0
locality                  0
latitude                  0
longitude                 0
country_id                0
locality_verbose          0
cuisines                  0
timings                 818
average_cost_for_two      0
price_range               0
highlights                0
aggregate_rating          0
rating_text               0
votes                     0
photo_count               0
opentable_support         0
delivery                  0
takeaway                  0
dtype: int64
```

In [20]:
```
df.shape
```

Out[20]:
```
(60411, 24)
```

In [21]:
```
df.dropna(subset=['timings'], inplace=True)
```

In [22]:
```
df.shape
```

Out[22]:
```
(59593, 24)
```

Exploratory Data Analysis:

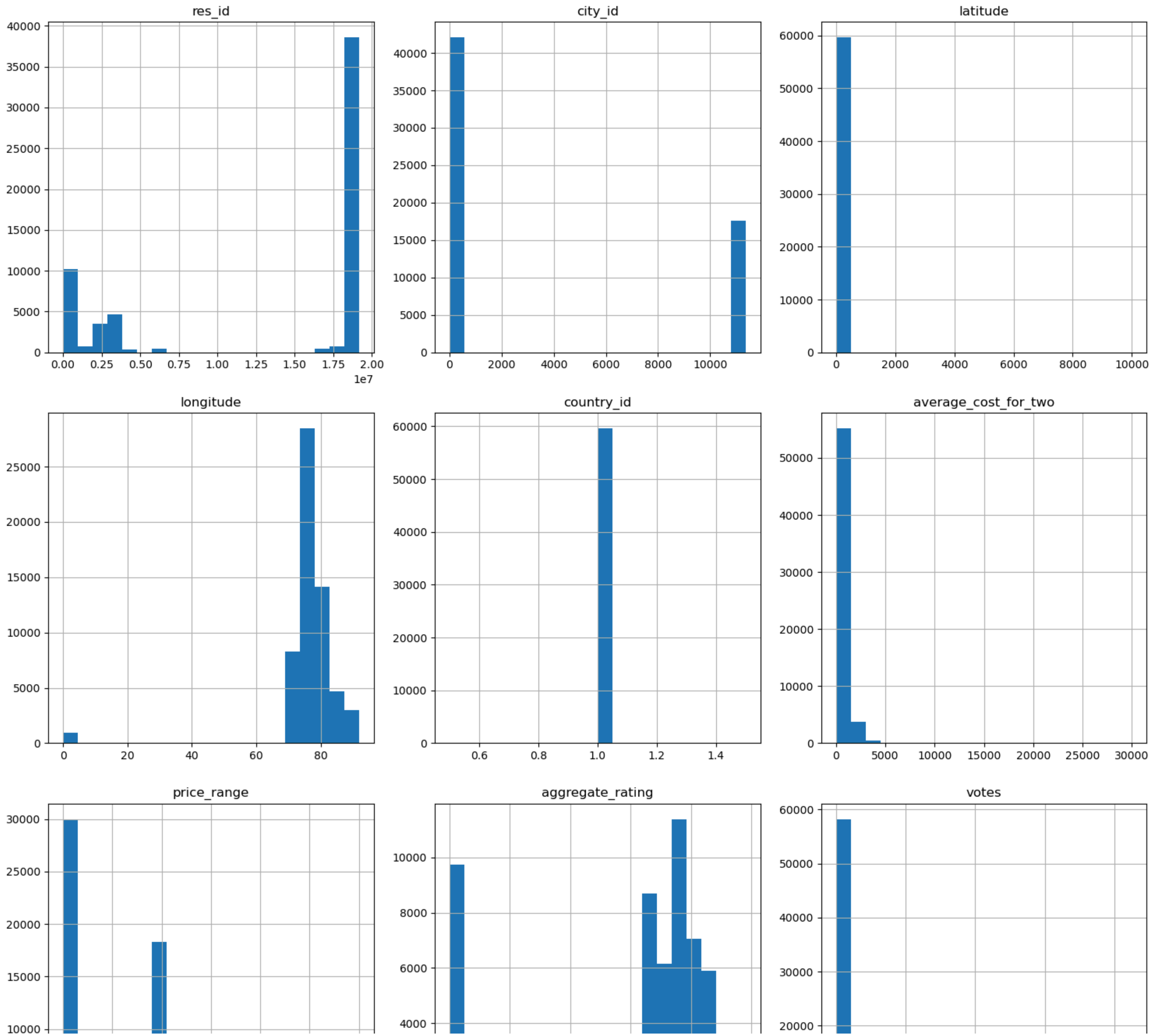
Descriptive Statistics: Summarize the central tendency, dispersion, and shape of the dataset's distribution.

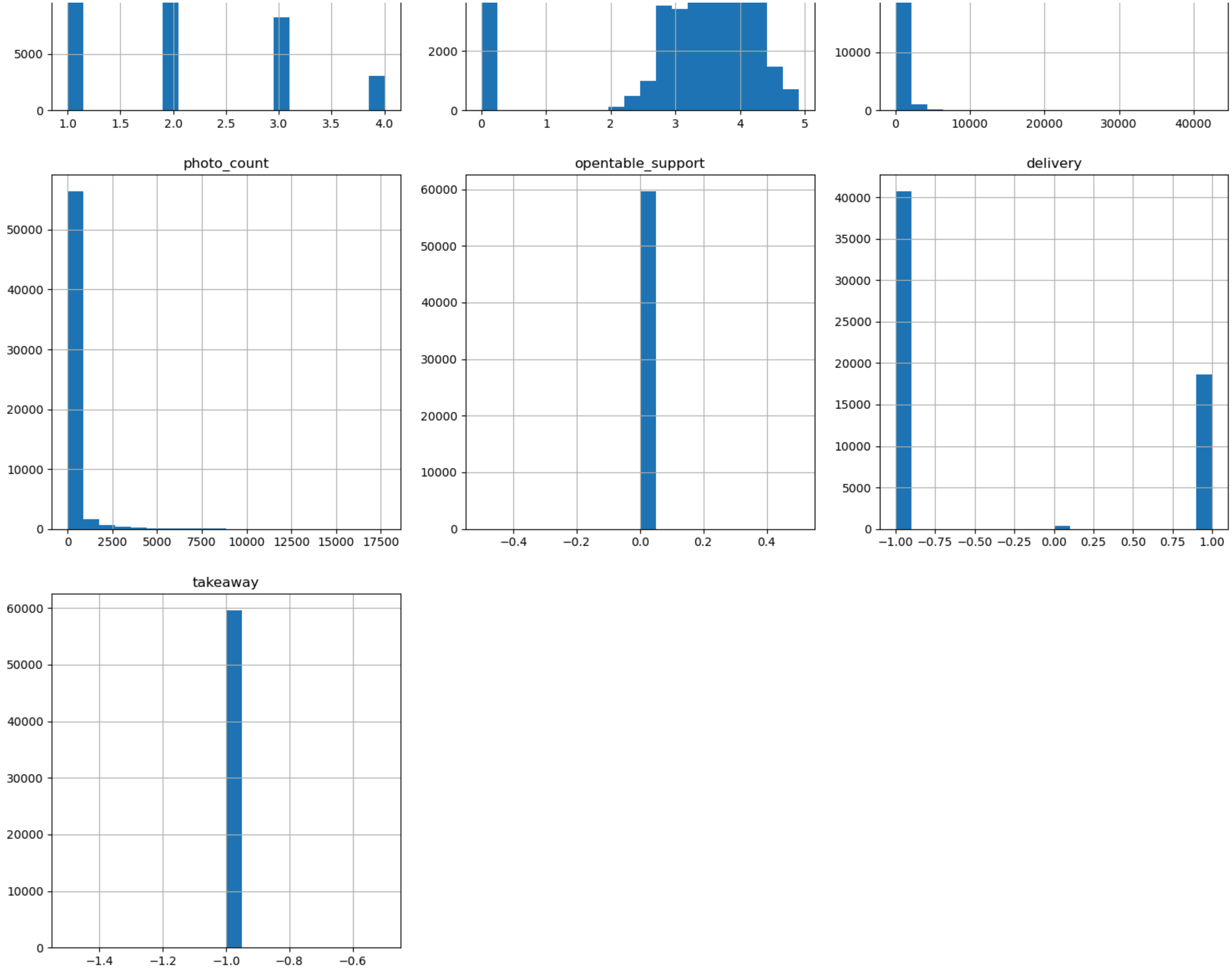
In [23]:
```
df.describe()
```

Out[23]:

	res_id	city_id	latitude	longitude	country_id	average_cost_for_two	price_range	aggregate_rating	
count	5.959300e+04	59593.000000	59593.000000	59593.000000	59593.0	59593.000000	59593.000000	59593.000000	595
mean	1.302184e+07	3340.680399	21.352361	76.586707	1.0	542.449080	1.737738	3.050964	2
std	8.156788e+06	5145.063815	41.463961	10.616694	0.0	596.214095	0.882172	1.427605	7
min	5.000000e+01	1.000000	0.000000	0.000000	1.0	0.000000	1.000000	0.000000	-
25%	3.000001e+06	7.000000	16.479014	74.748321	1.0	200.000000	1.000000	3.000000	
50%	1.869037e+07	25.000000	22.319499	77.127395	1.0	400.000000	1.000000	3.500000	
75%	1.885787e+07	11294.000000	26.745870	79.931594	1.0	600.000000	2.000000	4.000000	2
max	1.915979e+07	11354.000000	10000.000000	91.832769	1.0	30000.000000	4.000000	4.900000	425

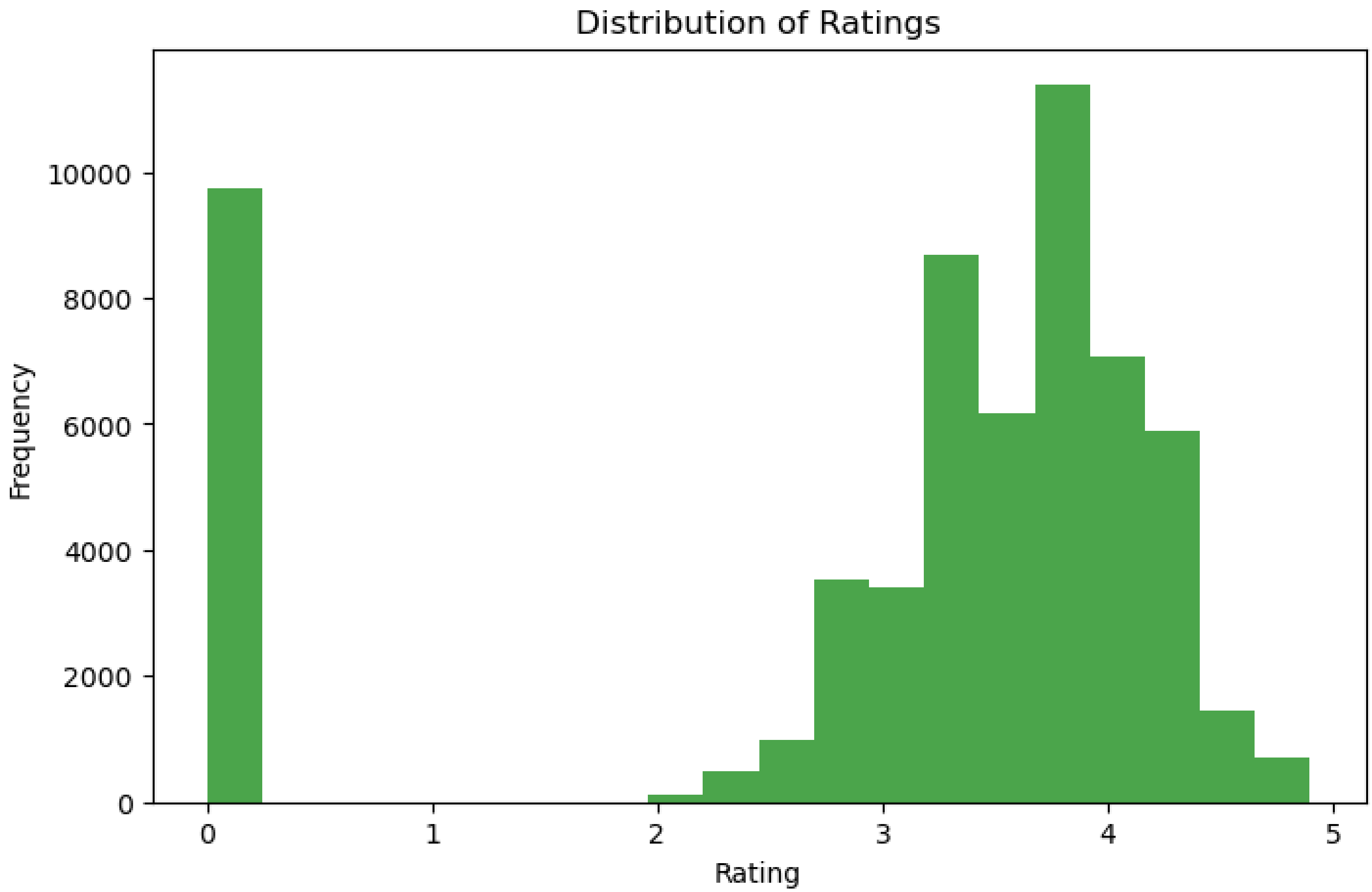
In [24]:
```
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

# Plot histograms for numerical columns
num_plots = len(numerical_cols)
num_rows = ((num_plots - 1) // 3) + 1
plt.figure(figsize=(15, 5 * num_rows))  # Set figure size dynamically
for i, col in enumerate(numerical_cols, start=1):
    plt.subplot(num_rows, 3, i)
    df[col].hist(bins=20)
    plt.title(col)
plt.tight_layout()  # Adjust layout
plt.show()
```

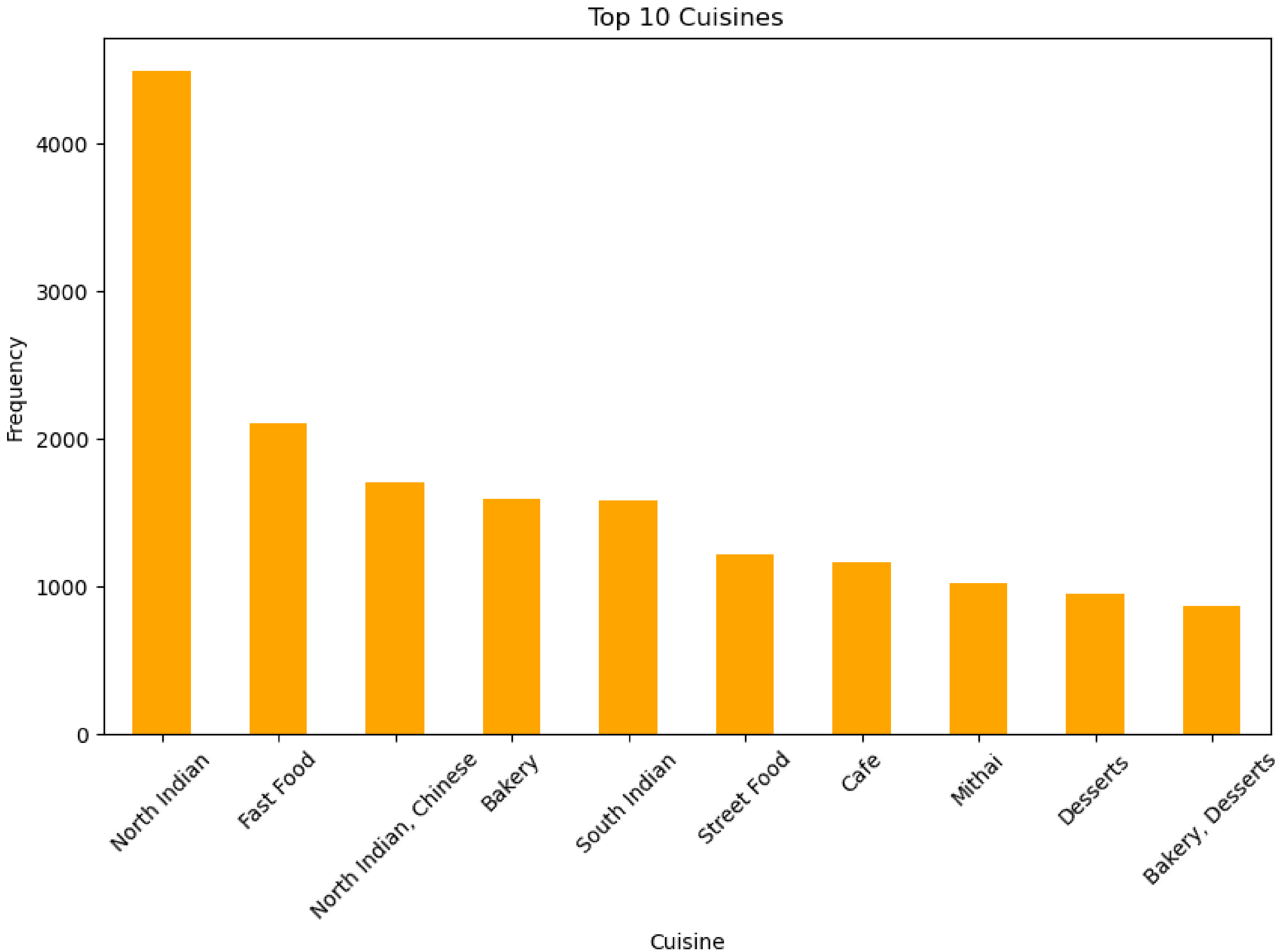

res_id
0 5000 10000 15000 20000 25000 30000 35000 40000
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
1e7
city_id
0 5000 10000 15000 20000 25000 30000 35000 40000
0 2000 4000 6000 8000 10000
latitude
0 10000 20000 30000 40000 50000 60000
0 2000 4000 6000 8000 10000
longitude
0 5000 10000 15000 20000 25000
0 20 40 60 80
country_id
0 10000 20000 30000 40000 50000 60000
0.6 0.8 1.0 1.2 1.4
average_cost_for_two
0 10000 20000 30000 40000 50000
0 5000 10000 15000 20000 25000 30000
price_range
10000 15000 20000 25000 30000
aggregate_rating
4000 6000 8000 10000
votes
20000 30000 40000 50000 60000


photo_count
opentable_support
delivery
takeaway

Distribution Analysis: Analyze the distribution of key variables (e.g., ratings, price range, cuisines).

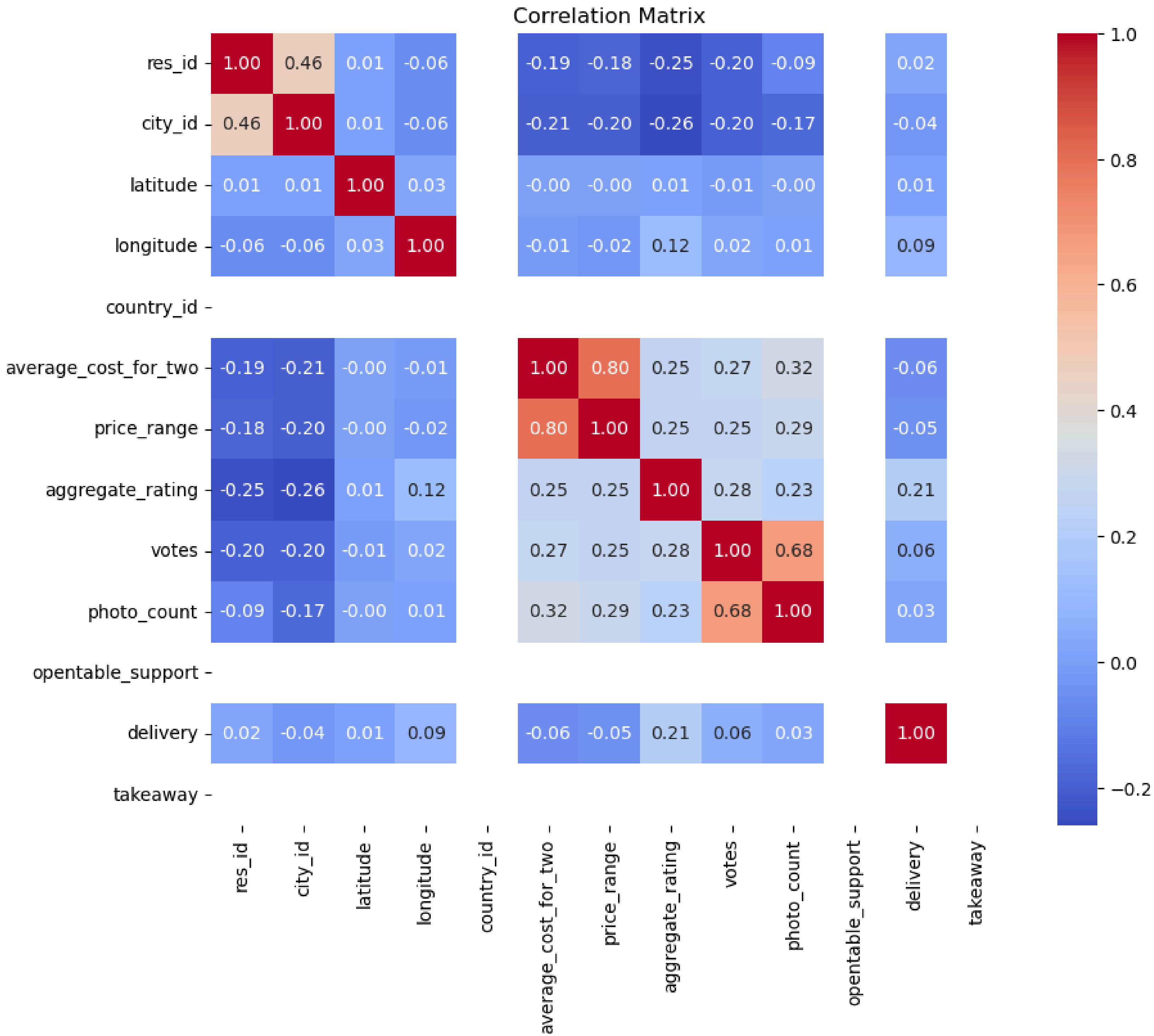
In [25]:
```
# Visualize distribution of ratings
plt.figure(figsize=(8, 5))
plt.hist(df['aggregate_rating'], bins=20, color='green', alpha=0.7)
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()

# Analyze distribution of cuisines
top_cuisines = df['cuisines'].value_counts().head(10)
plt.figure(figsize=(10, 6))
top_cuisines.plot(kind='bar', color='orange')
plt.title('Top 10 Cuisines')
plt.xlabel('Cuisine')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```


Distribution of Ratings
Frequency
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Rating


Top 10 Cuisines
Frequency
0
1000
2000
3000
4000
North Indian
Fast Food
North Indian, Chinese
Bakery
South Indian
Street Food
Cafe
Mithai
Desserts
Bakery, Desserts
Cuisine

Correlation Analysis: Examine the relationships between different variables.

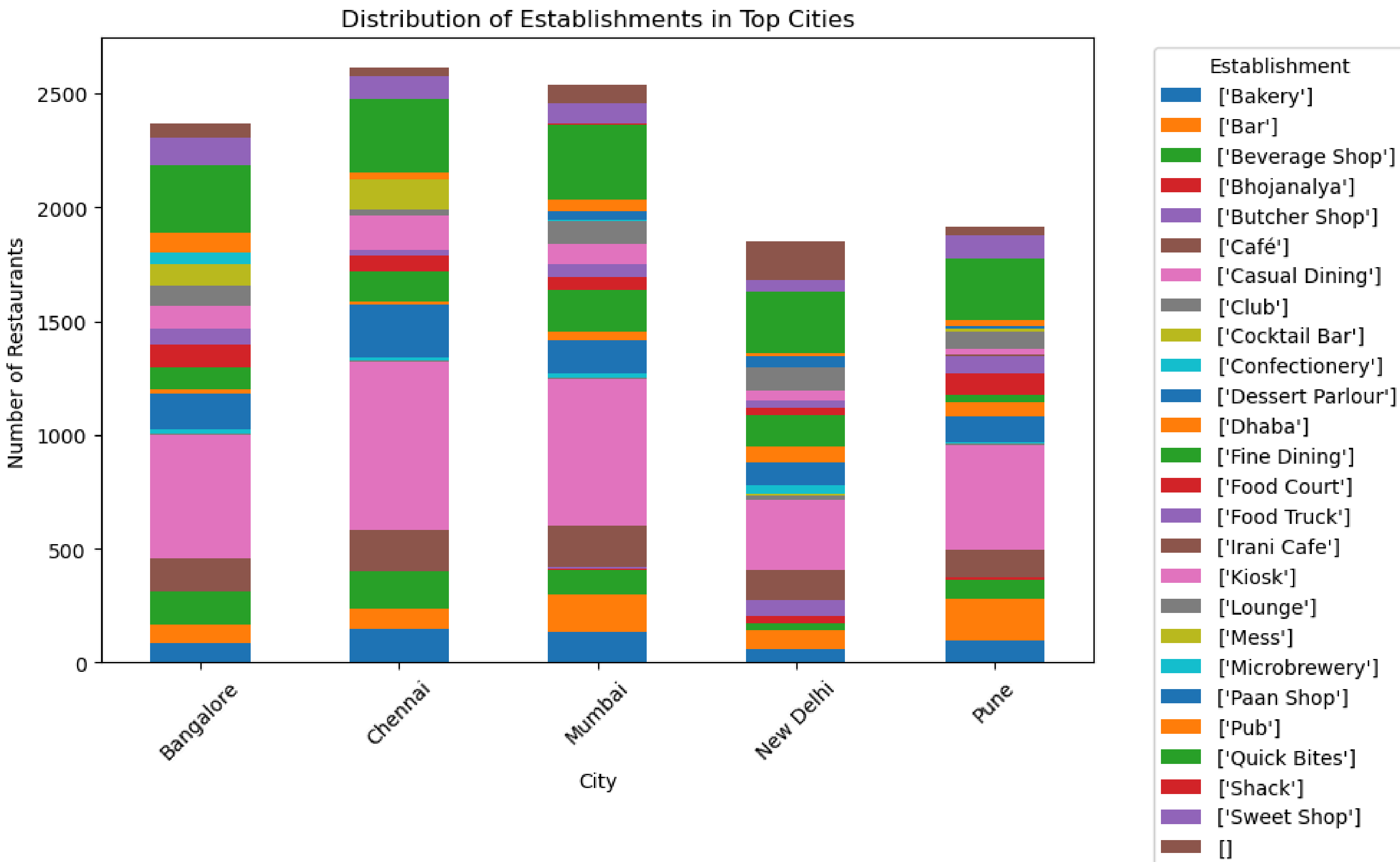
In [26]:
```
numeric_df = df.select_dtypes(include='number').corr() # Select only numeric columns & Calculate correlation

# Visualize correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
res_id
city_id
latitude
longitude
country_id
average_cost_for_two
price_range
aggregate_rating
votes
photo_count
opentable_support
delivery
takeaway
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Regional Analysis:

Compare the restaurant trends and customer preferences across different cities or regions.

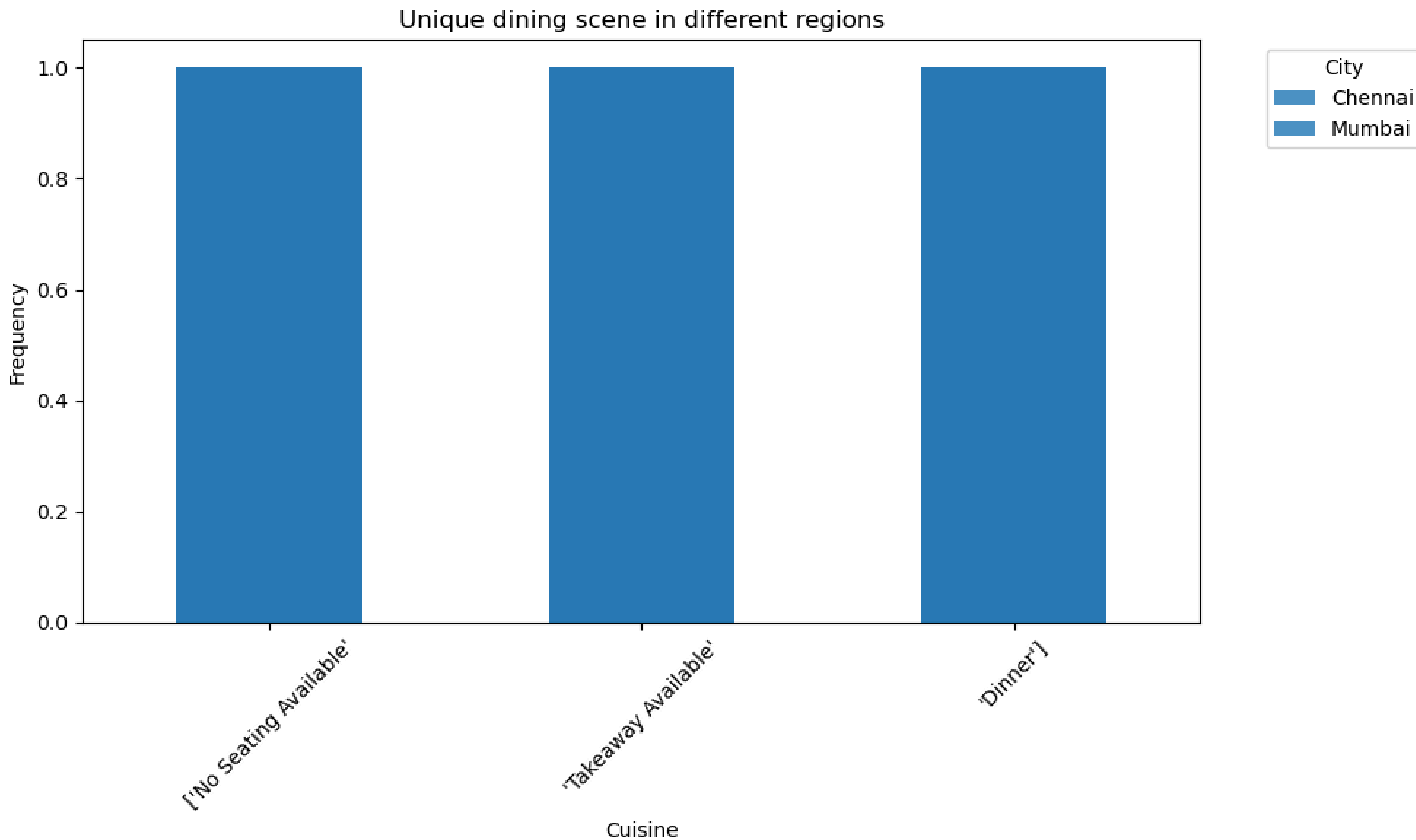
In [27]:
```
top_cities = df['city'].value_counts().nlargest(5).index.tolist()  # Change 5 to the number of top cities you

# Step 2: Filter Data
filtered_df = df[df['city'].isin(top_cities)]

# Step 3: Aggregate Establishments
city_establishment_counts = filtered_df.groupby('city')['establishment'].value_counts().unstack().fillna(0)

# Step 4: Plot
city_establishment_counts.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Distribution of Establishments in Top Cities')
plt.xlabel('City')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45)
plt.legend(title='Establishment', bbox_to_anchor=(1.05, 1), loc='upper left')  # Adjust legend location
plt.subplots_adjust(right=0.75)  # Adjust space for the legend
plt.tight_layout()
plt.show()
```



Distribution of Establishments in Top Cities
Number of Restaurants
0
500
1000
1500
2000
2500
Bangalore
Chennai
Mumbai
New Delhi
Pune
City
Establishment
['Bakery']
['Bar']
['Beverage Shop']
['Bhojanalya']
['Butcher Shop']
['Café']
['Casual Dining']
['Club']
['Cocktail Bar']
['Confectionery']
['Dessert Parlour']
['Dhaba']
['Fine Dining']
['Food Court']
['Food Truck']
['Irani Cafe']
['Kiosk']
['Lounge']
['Mess']
['Microbrewery']
['Paan Shop']
['Pub']
['Quick Bites']
['Shack']
['Sweet Shop']
[]

Identify unique characteristics of the dining scene in each region.

In [28]:
```
# Step 1: Filter data for the top 5 cities and exclude entries with certain keywords in highlights
top_cities = df['city'].value_counts().nlargest(5).index.tolist()
exclude_keywords = ['Credit Card', 'Debit Card', 'Cash', 'Digital Payments Accepted']

plt.figure(figsize=(10, 6))

for city in top_cities:
    city_data = df[(df['city'] == city) & (~df['highlights'].str.contains('|'.join(exclude_keywords)))]
    if not city_data.empty:
        city_cuisine_counts = city_data['highlights'].str.split(', ').explode().value_counts().nlargest(5)
        city_cuisine_counts.plot(kind='bar', label=city, alpha=0.8)

plt.title('Unique dining scene in different regions')
plt.xlabel('Cuisine')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.legend(title='City', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



Unique dining scene in different regions
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
['No Seating Available'
'Takeaway Available'
'Dinner']
Cuisine
City
Chennai
Mumbai

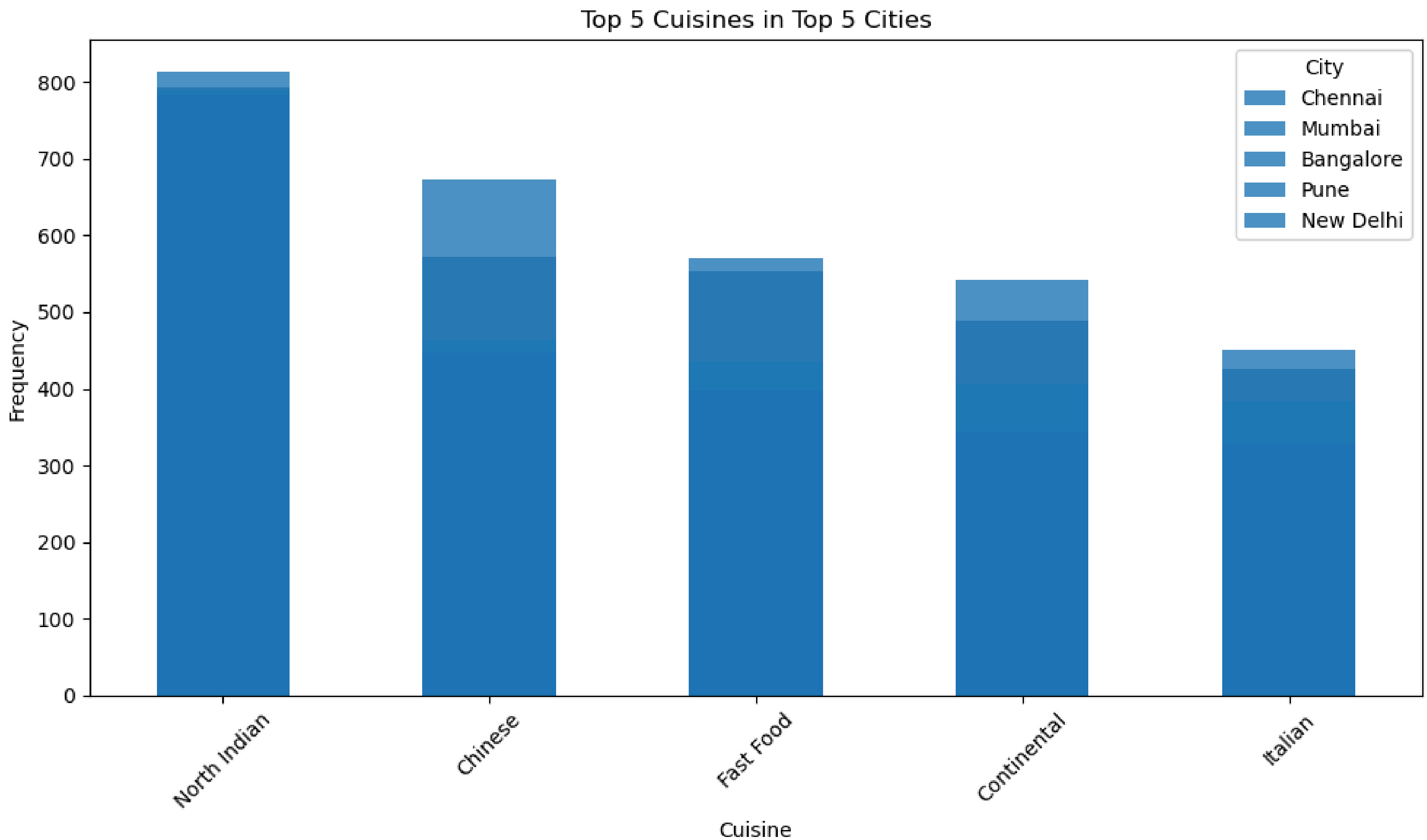
Customer Preference Analysis:

Analyze the types of cuisines that are popular in different regions.

In [29]:
```
top_cuisines = df['cuisines'].str.split(', ', expand=True).stack().value_counts().nlargest(5)
top_cuisines
```

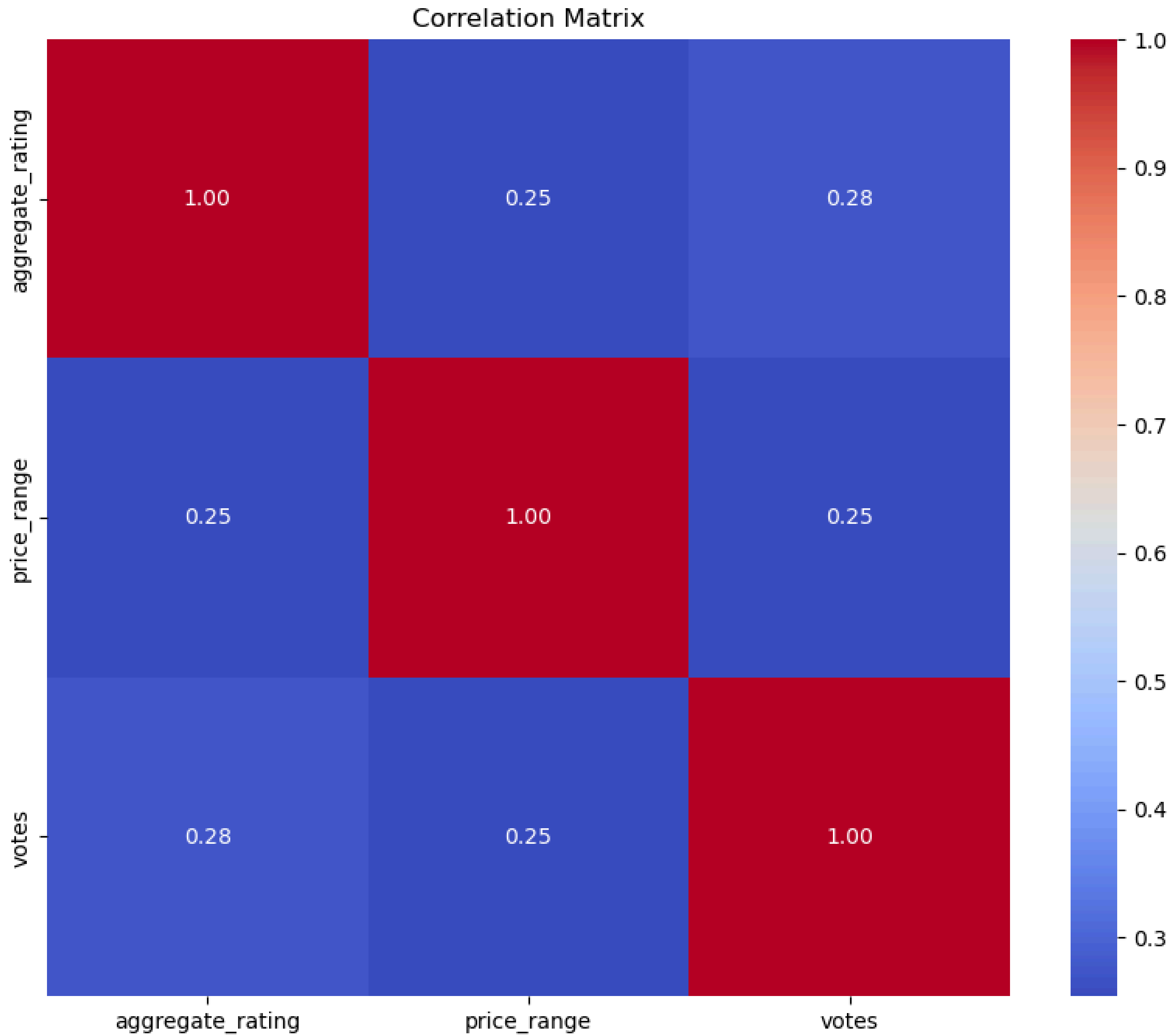
Out[29]:
```
North Indian    21023
Chinese         13971
Fast Food       13039
Desserts         7703
Beverages        7410
Name: count, dtype: int64
```

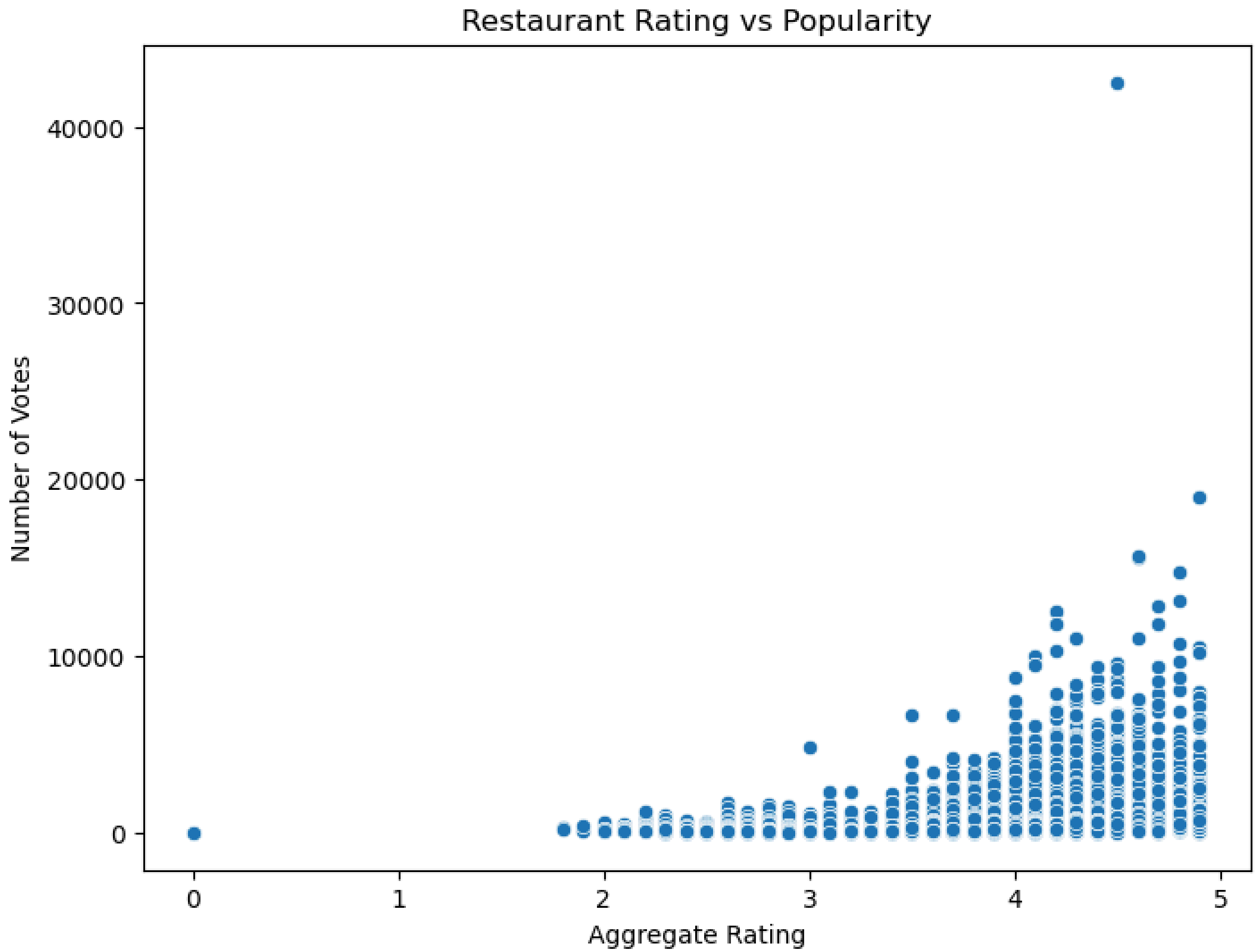
In [30]:
```
# Step 1: Filter data for the top 5 cities
top_cities = df['city'].value_counts().nlargest(5).index.tolist()
filtered_df = df[df['city'].isin(top_cities)]

# Step 2: Extract cuisines from filtered data
cuisine_series = filtered_df['cuisines'].str.split(', ').apply(pd.Series).stack()

# Step 3: Count the occurrence of each cuisine
top_cuisines_per_city = cuisine_series.value_counts().nlargest(5)

# Step 4: Visualize the top cuisines for each city
plt.figure(figsize=(10, 6))
for city in top_cities:
    city_cuisines = filtered_df.loc[filtered_df['city'] == city, 'cuisines'].str.split(', ')
    cuisine_counts = city_cuisines.explode().value_counts().nlargest(5)
    cuisine_counts.plot(kind='bar', label=city, alpha=0.8)
plt.title('Top 5 Cuisines in Top 5 Cities')
plt.xlabel('Cuisine')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.legend(title='City')
plt.tight_layout()
plt.show()
```

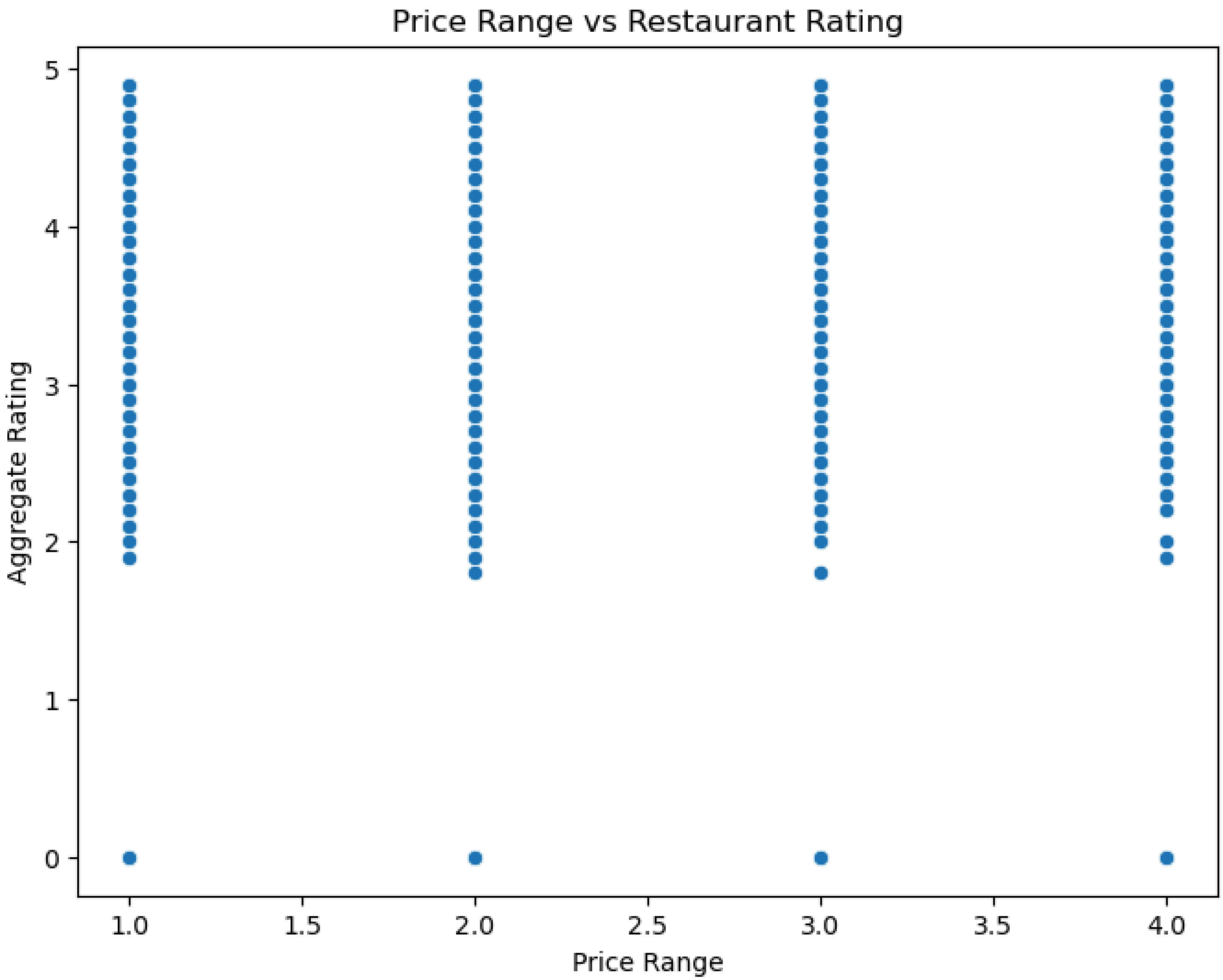

Top 5 Cuisines in Top 5 Cities
Frequency
0
100
200
300
400
500
600
700
800
North Indian
Chinese
Fast Food
Continental
Italian
Cuisine
City
Chennai
Mumbai
Bangalore
Pune
New Delhi

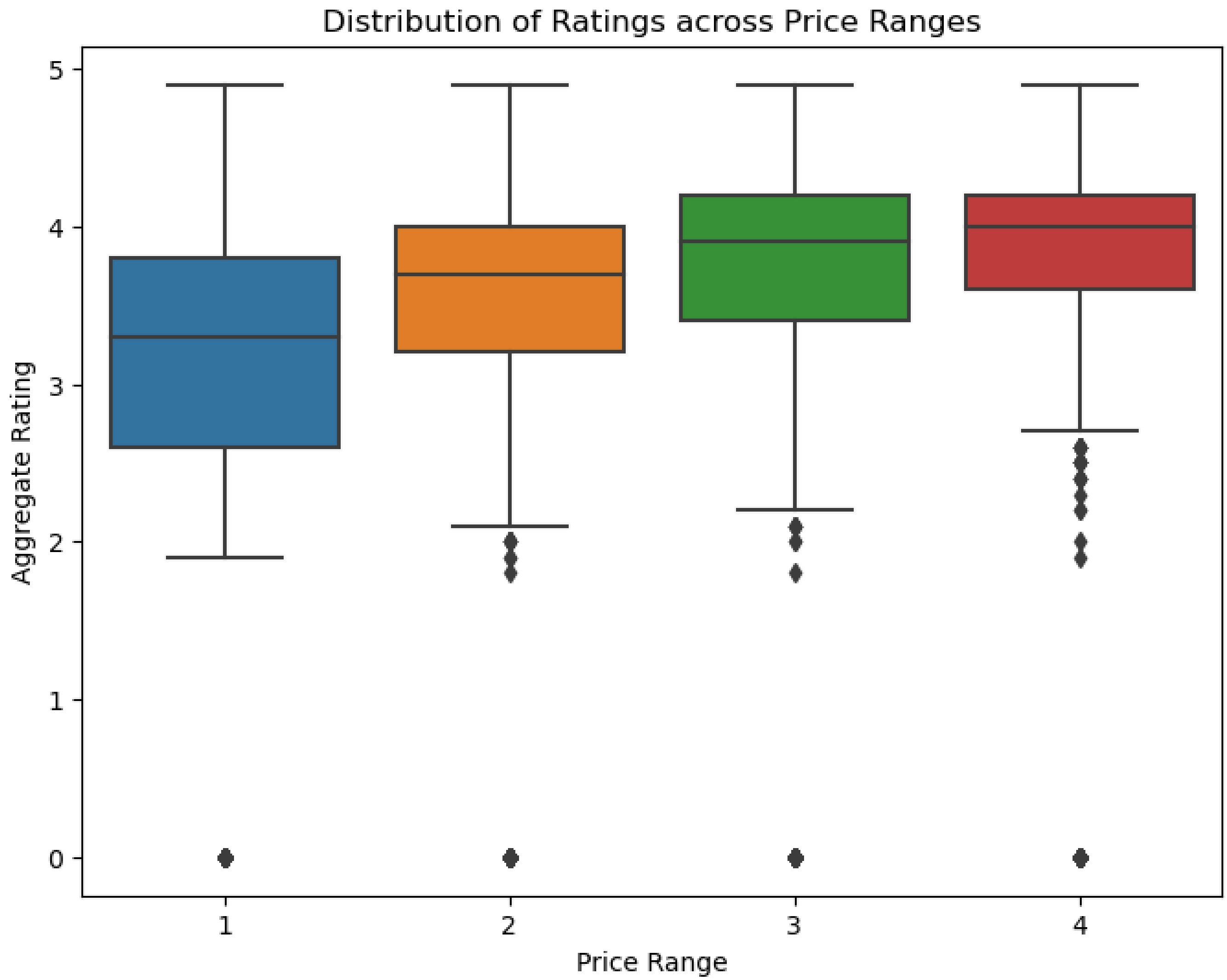
Examine the relationship between restaurant ratings, price range, and popularity.

In [31]:
```
# Correlation Analysis
correlation_matrix = df[['aggregate_rating', 'price_range', 'votes']].corr()

# Visualize correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# Scatter plot: aggregate_rating vs votes
plt.figure(figsize=(8, 6))
sns.scatterplot(x='aggregate_rating', y='votes', data=df)
plt.title('Restaurant Rating vs Popularity')
plt.xlabel('Aggregate Rating')
plt.ylabel('Number of Votes')
plt.show()

# Scatter plot: price_range vs aggregate_rating
plt.figure(figsize=(8, 6))
sns.scatterplot(x='price_range', y='aggregate_rating', data=df)
plt.title('Price Range vs Restaurant Rating')
plt.xlabel('Price Range')
plt.ylabel('Aggregate Rating')
plt.show()

# Box plot: aggregate_rating across different price ranges
plt.figure(figsize=(8, 6))
sns.boxplot(x='price_range', y='aggregate_rating', data=df)
plt.title('Distribution of Ratings across Price Ranges')
plt.xlabel('Price Range')
plt.ylabel('Aggregate Rating')
plt.show()
```


Correlation Matrix
aggregate_rating
price_range
votes
1.00
0.25
0.28
0.25
1.00
0.25
0.28
0.25
1.00
aggregate_rating
price_range
votes
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3


Restaurant Rating vs Popularity
Number of Votes
40000
30000
20000
10000
0
Aggregate Rating
0
1
2
3
4
5


Price Range vs Restaurant Rating
Aggregate Rating
Price Range
0
1
2
3
4
5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Distribution of Ratings across Price Ranges
Aggregate Rating
Price Range
0
1
2
3
4
5
1
2
3
4

Competitive Analysis:

Identify major competitors in each region based on cuisine, pricing, and ratings.

In [32]:
```
filtered_result = df.loc[(df['cuisines'] != 'Unknown') & (df['average_cost_for_two'] != 0),
                         ['city', 'cuisines', 'average_cost_for_two', 'aggregate_rating']]

duplicate_cities = filtered_result.drop_duplicates(subset='city')

duplicate_cities.nlargest(20, 'aggregate_rating')
```

Out[32]:

	city	cuisines	average_cost_for_two	aggregate_rating
15114	Amritsar	Fast Food, Italian	500	4.9
19630	Bangalore	Continental, North Indian, Chinese, European, ...	2100	4.9
27257	Bhubaneshwar	Tex-Mex, Fast Food	700	4.9
129079	Mangalore	Ice Cream, Desserts, Beverages, Fast Food	250	4.9
134885	Thane	Modern Indian, North Indian, Chinese, Momos, A...	1600	4.9
151290	Nashik	Continental, Indian, Chinese	1000	4.9
173048	Rajkot	North Indian, Gujarati, South Indian, Continental	700	4.9
5936	Ajmer	Continental, Beverages, South Indian, Fast Foo...	600	4.8
11147	Allahabad	North Indian	200	4.8
24601	Bhopal	Street Food, South Indian, Fast Food, Desserts...	400	4.8
33460	Chennai	North Indian, European, Mediterranean, Contine...	1500	4.8
134905	Navi Mumbai	Italian, Continental, Mexican	1600	4.8
193113	Trichy	Arabian, Chinese, BBQ, Rolls	500	4.8
29812	Chandigarh	European, Continental, North Indian, Finger Fo...	1600	4.7
45090	Coimbatore	Biryani, South Indian	700	4.7
53885	New Delhi	Asian, Chinese, Thai, Japanese	2500	4.7
146468	Nagpur	Cafe, Chinese, Fast Food, Beverages	500	4.7
186818	Surat	Beverages, North Indian	250	4.7
114338	Kolkata	Italian, Chinese, Finger Food	1300	4.6
132502	Meerut	North Indian	100	4.6

Analyze the strengths and weaknesses of these competitors.

Strengths:

1. **High Aggregate Ratings:** Competitors across various cities, including Amritsar, Bangalore, Bhubaneshwar, New Delhi, and others, boast exceptionally high aggregate ratings (ranging from 4.6 to 4.9), indicating high customer satisfaction and quality food service.

2. **Diverse Cuisine Offerings:** Many competitors offer a wide range of cuisines, such as Continental, North Indian, Chinese, and more, appealing to a broader customer base with varying preferences.
3. **Reasonable Average Cost for Two:** Despite delivering high-quality food and service, several competitors maintain a reasonable average cost for two, catering to budget-conscious customers while offering value for money.
4. **Consistency Across Multiple Cities:** Competitors with branches in multiple cities demonstrate consistency in delivering high-quality food and service across different locations, showcasing strong brand management and operational efficiency.

Weaknesses:

1. **Limited Analysis Scope:** The provided data lacks insights into specific weaknesses of individual competitors, such as customer complaints or operational challenges, necessitating further data and context for a comprehensive analysis.
2. **Potential Operational Challenges:** Competitors may face challenges in maintaining consistency in quality across locations, managing operational costs, or adapting to changing consumer preferences, impacting their overall performance.
3. **External Factors:** External factors such as economic conditions or unforeseen events like pandemics can also impact the strengths and weaknesses of competitors in the restaurant industry, adding uncertainty to their operational environment.

In conclusion, while competitors exhibit strengths such as high ratings, diverse cuisine offerings, and reasonable pricing, a detailed analysis of weaknesses would require additional data and context to identify specific improvement areas or potential challenges they mav encounter.

THE END