Importing Libraries

In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Overview:

Explore the basic characteristics of the dataset, including dimensions, data types, and missing values.

In [18]:
```
df= pd.read_csv(r"C:\Users\aryan\OneDrive\Desktop\DS PROJECTS\BIG PROJECT\ZOMATO ANALYSIS (2)\Indian-Resturants.csv")
df
```

Out[18]:

	res_id	name	establishment	url	address	city	city_id	localit
0	3400299	Bikanervala	['Quick Bites']	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	Agra	34	Khanda
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	Agr Cant
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	Shahgan
3	3400290	Bhagat Halwai	['Quick Bites']	https://www.zomato.com/agra/bhagat-halwai-civi...	Near Anjana Cinema, Nehru Nagar, Civil Lines, ...	Agra	34	Civil Line
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	Tajgan
...	...	...	...	...	...	...	...	..
211939	3202251	Kali Mirch Cafe And Restaurant	['Casual Dining']	https://www.zomato.com/vadodara/kali-mirch-caf...	Manu Smriti Complex, Near Navrachna School, GI...	Vadodara	32	Fatehgun
					Mahalaxmi Apartment,			

	res_id	name	establishment	url	address	city	city_id	locality
211940	3200996	Raju Omlet	['Quick Bites']	https://www.zomato.com/vadodara/raju-omlet-kar...	Opposite B O B, Karoli Ba...	Vadodara	32	Karelibaug
211941	18984164	The Grand Thakar	['Casual Dining']	https://www.zomato.com/vadodara/the-grand-thak...	3rd Floor, Shreem Shalini Mall, Opposite Conqu...	Vadodara	32	Alkapur
211942	3201138	Subway	['Quick Bites']	https://www.zomato.com/vadodara/subway-1-akota...	G-2, Vedant Platina, Near Cosmos, Akota, Vadodara	Vadodara	32	Akota
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	Vadiwad

211944 rows × 26 columns

Size of dataset

In [19]:
```
print("Rows, Columns:", df.shape)
```
```
Rows, Columns: (211944, 26)
```

First 5 rows

In [20]:
```
print("----- HEAD -----")
print(df.head())
```

```
----- HEAD -----
    res_id                          name      establishment  \
0  3400299                    Bikanervala    ['Quick Bites']
1  3400005  Mama Chicken Mama Franky House    ['Quick Bites']
2  3401013                   Bhagat Halwai    ['Quick Bites']
3  3400290                   Bhagat Halwai    ['Quick Bites']
4  3401744     The Salt Cafe Kitchen & Bar  ['Casual Dining']

                                                 url  \
0  https://www.zomato.com/agra/bikanervala-khanda...
1  https://www.zomato.com/agra/mama-chicken-mama-...
2  https://www.zomato.com/agra/bhagat-halwai-2-sh...
3  https://www.zomato.com/agra/bhagat-halwai-civi...
4  https://www.zomato.com/agra/the-salt-cafe-kitc...

                                             address  city  city_id  \
0  Kalyani Point, Near Tulsi Cinema, Bypass Road,...  Agra       34
1         Main Market, Sadar Bazaar, Agra Cantt, Agra  Agra       34
2  62/1, Near Easy Day, West Shivaji Nagar, Goalp...  Agra       34
3  Near Anjana Cinema, Nehru Nagar, Civil Lines, ...  Agra       34
4         1C,3rd Floor, Fatehabad Road, Tajganj, Agra  Agra       34

      locality   latitude  longitude  ... price_range  currency  \
0     Khandari  27.211450  78.002381  ...           2       Rs.
1   Agra Cantt  27.160569  78.011583  ...           2       Rs.
2     Shahganj  27.182938  77.979684  ...           1       Rs.
3  Civil Lines  27.205668  78.004799  ...           1       Rs.
4      Tajganj  27.157709  78.052421  ...           3       Rs.

                                          highlights aggregate_rating  \
0  ['Lunch', 'Takeaway Available', 'Credit Card',...              4.4
1  ['Delivery', 'No Alcohol Available', 'Dinner',...              4.4
2  ['No Alcohol Available', 'Dinner', 'Takeaway A...              4.2
3  ['Takeaway Available', 'Credit Card', 'Lunch',...              4.3
4  ['Lunch', 'Serves Alcohol', 'Cash', 'Credit Ca...              4.9

   rating_text  votes  photo_count opentable_support delivery  takeaway
0    Very Good    814          154               0.0       -1        -1
1    Very Good   1203          161               0.0       -1        -1
2    Very Good    801          107               0.0        1        -1
3    Very Good    693          157               0.0        1        -1
4    Excellent    470          291               0.0        1        -1

[5 rows x 26 columns]
```

Column names

In [21]:
```
print("----- COLUMNS -----")
print(list(df.columns))
```

```
----- COLUMNS -----
['res_id', 'name', 'establishment', 'url', 'address', 'city', 'city_id', 'locality', 'latitude', 'longitude', 'zipcode', 'country_id', 'locality_verbose', 'cuisines', 'timings', 'average_cost_for_t
wo', 'price_range', 'currency', 'highlights', 'aggregate_rating', 'rating_text', 'votes', 'photo_count', 'opentable_support', 'delivery', 'takeaway']
```

Data types and non-null counts

In [22]:
```
print("----- INFORMATION -----")
print(df.describe())
```

```
----- INFORMATION -----
            res_id       city_id      latitude     longitude  country_id  \
count  2.119440e+05  211944.000000  211944.000000  211944.000000    211944.0
mean   1.349411e+07    4746.785434      21.499758      77.615276         1.0
std    7.883722e+06    5568.766386      22.781331       7.500104         0.0
min    5.000000e+01       1.000000       0.000000       0.000000         1.0
25%    3.301027e+06      11.000000      15.496071      74.877961         1.0
50%    1.869573e+07      34.000000      22.514494      77.425971         1.0
75%    1.881297e+07   11306.000000      26.841667      80.219323         1.0
max    1.915979e+07   11354.000000   10000.000000      91.832769         1.0

       average_cost_for_two    price_range  aggregate_rating          votes  \
count         211944.000000  211944.000000     211944.000000  211944.000000
mean             595.812229       1.882535          3.395937     378.001864
std              606.239363       0.892989          1.283642     925.333370
min                0.000000       1.000000          0.000000     -18.000000
25%              250.000000       1.000000          3.300000      16.000000
50%              400.000000       2.000000          3.800000     100.000000
75%              700.000000       2.000000          4.100000     362.000000
max            30000.000000       4.000000          4.900000   42539.000000

         photo_count  opentable_support       delivery  takeaway
count  211944.000000           211896.0  211944.000000  211944.0
mean      256.971224                0.0      -0.255907      -1.0
std       867.668940                0.0       0.964172       0.0
min         0.000000                0.0      -1.000000      -1.0
25%         3.000000                0.0      -1.000000      -1.0
50%        18.000000                0.0      -1.000000      -1.0
75%       128.000000                0.0       1.000000      -1.0
max     17702.000000                0.0       1.000000      -1.0
```

Basic statistics for numeric columns

In [23]:

```
print("----- DESCRIBE (numeric) -----")
print(df.describe())
```

```
----- DESCRIBE (numeric) -----
            res_id       city_id      latitude     longitude  country_id  \
count  2.119440e+05  211944.000000  211944.000000  211944.000000    211944.0
mean   1.349411e+07    4746.785434      21.499758      77.615276         1.0
std    7.883722e+06    5568.766386      22.781331       7.500104         0.0
min    5.000000e+01       1.000000       0.000000       0.000000         1.0
25%    3.301027e+06      11.000000      15.496071      74.877961         1.0
50%    1.869573e+07      34.000000      22.514494      77.425971         1.0
75%    1.881297e+07   11306.000000      26.841667      80.219323         1.0
max    1.915979e+07   11354.000000   10000.000000      91.832769         1.0

       average_cost_for_two    price_range  aggregate_rating          votes  \
count         211944.000000  211944.000000     211944.000000  211944.000000
mean             595.812229       1.882535          3.395937     378.001864
std              606.239363       0.892989          1.283642     925.333370
min                0.000000       1.000000          0.000000     -18.000000
25%              250.000000       1.000000          3.300000      16.000000
50%              400.000000       2.000000          3.800000     100.000000
75%              700.000000       2.000000          4.100000     362.000000
max            30000.000000       4.000000          4.900000   42539.000000

         photo_count  opentable_support       delivery  takeaway
count  211944.000000           211896.0  211944.000000  211944.0
mean      256.971224                0.0      -0.255907      -1.0
std       867.668940                0.0       0.964172       0.0
min         0.000000                0.0      -1.000000      -1.0
25%         3.000000                0.0      -1.000000      -1.0
50%        18.000000                0.0      -1.000000      -1.0
75%       128.000000                0.0       1.000000      -1.0
max     17702.000000                0.0       1.000000      -1.0
```

Basic Statistics for all columns (shows top values for objects)

In [24]:

```
print("----- DESCRIBE (all) -----")
print(df.describe(include='all').T)
```

```
----- DESCRIBE (all) -----
                   count unique  \
res_id          211944.0    NaN
name              211944  41100
establishment     211944     27
url               211944  55568
address           211810  50657
city              211944     99
city_id         211944.0    NaN
locality          211944   3731
latitude        211944.0    NaN
```

```
longitude           211944.0    NaN
zipcode                48757   1311
country_id          211944.0    NaN
locality_verbose      211944   3910
cuisines              210553   9382
timings               208070   7740
average_cost_for_two  211944.0    NaN
price_range         211944.0    NaN
currency              211944      1
highlights            211944  31455
aggregate_rating    211944.0    NaN
rating_text           211944     39
votes               211944.0    NaN
photo_count         211944.0    NaN
opentable_support   211896.0    NaN
delivery            211944.0    NaN
takeaway            211944.0    NaN

                                                           top  \
res_id                                                     NaN
name                                             Domino's Pizza
establishment                                  ['Quick Bites']
url                   https://www.zomato.com/chennai/3bs-buddies-bar...
address                          Laxman Jhula, Tapovan, Rishikesh
city                                                   Chennai
city_id                                                    NaN
locality                                           Civil Lines
latitude                                                   NaN
longitude                                                  NaN
zipcode                                                      0
country_id                                                 NaN
locality_verbose                          Ana Sagar Lake, Ajmer
cuisines                                          North Indian
timings                                         11 AM to 11 PM
average_cost_for_two                                       NaN
price_range                                                NaN
currency                                                   Rs.
highlights            ['Dinner', 'Takeaway Available', 'Lunch', 'Cas...
aggregate_rating                                           NaN
rating_text                                          Very Good
votes                                                      NaN
photo_count                                                NaN
opentable_support                                          NaN
delivery                                                   NaN
takeaway                                                   NaN

                        freq            mean            std   min  \
res_id                   NaN  13494112.348106  7883721.972533  50.0
name                    3108             NaN             NaN   NaN
establishment          64390             NaN             NaN   NaN
url                      169             NaN             NaN   NaN
address                  299             NaN             NaN   NaN
city                   11630             NaN             NaN   NaN
city_id                  NaN     4746.785434     5568.766386   1.0
locality                3660             NaN             NaN   NaN
latitude                 NaN       21.499758       22.781331   0.0
longitude                NaN       77.615276        7.500104   0.0
zipcode                 7100             NaN             NaN   NaN
country_id               NaN             1.0             0.0   1.0
locality_verbose        1760             NaN             NaN   NaN
cuisines               15996             NaN             NaN   NaN
timings                26605             NaN             NaN   NaN
average_cost_for_two     NaN      595.812229      606.239363   0.0
price_range              NaN        1.882535        0.892989   1.0
currency              211944             NaN             NaN   NaN
highlights              3352             NaN             NaN   NaN
aggregate_rating         NaN        3.395937        1.283642   0.0
rating_text            65451             NaN             NaN   NaN
votes                    NaN      378.001864       925.33337 -18.0
photo_count              NaN      256.971224       867.66894   0.0
opentable_support        NaN             0.0             0.0   0.0
delivery                 NaN       -0.255907        0.964172  -1.0
takeaway                 NaN            -1.0             0.0  -1.0

                            25%         50%         75%         max
res_id                3301027.0  18695734.0  18812974.0  19159790.0
name                        NaN         NaN         NaN         NaN
establishment               NaN         NaN         NaN         NaN
url                         NaN         NaN         NaN         NaN
address                     NaN         NaN         NaN         NaN
city                        NaN         NaN         NaN         NaN
city_id                    11.0        34.0     11306.0     11354.0
locality                    NaN         NaN         NaN         NaN
latitude              15.496071   22.514494   26.841667     10000.0
longitude             74.877961   77.425971   80.219323   91.832769
```

```
zipcode                 NaN      NaN      NaN      NaN
country_id              1.0      1.0      1.0      1.0
locality_verbose        NaN      NaN      NaN      NaN
cuisines                NaN      NaN      NaN      NaN
timings                 NaN      NaN      NaN      NaN
average_cost_for_two  250.0    400.0    700.0  30000.0
price_range             1.0      2.0      2.0      4.0
currency                NaN      NaN      NaN      NaN
highlights              NaN      NaN      NaN      NaN
aggregate_rating        3.3      3.8      4.1      4.9
rating_text             NaN      NaN      NaN      NaN
votes                  16.0    100.0    362.0  42539.0
photo_count             3.0     18.0    128.0  17702.0
opentable_support       0.0      0.0      0.0      0.0
delivery               -1.0     -1.0      1.0      1.0
takeaway               -1.0     -1.0     -1.0     -1.0
```

Missing values summary (counts and percent)

In [25]:

```
miss = df.isnull().sum().sort_values(ascending=False)
miss_pct = (df.isnull().mean()*100).sort_values(ascending=False)
print("----- MISSING VALUES (top 20) -----")
print(pd.concat([miss, miss_pct], axis=1, keys=['missing_count','missing_percent']).head(20))
```

```
----- MISSING VALUES (top 20) -----
                      missing_count  missing_percent
zipcode                      163187        76.995338
timings                        3874         1.827841
cuisines                       1391         0.656305
address                         134         0.063224
opentable_support                48         0.022647
city                              0         0.000000
name                              0         0.000000
establishment                     0         0.000000
url                               0         0.000000
res_id                            0         0.000000
longitude                         0         0.000000
latitude                          0         0.000000
locality                          0         0.000000
city_id                           0         0.000000
locality_verbose                  0         0.000000
average_cost_for_two              0         0.000000
price_range                       0         0.000000
country_id                        0         0.000000
currency                          0         0.000000
highlights                        0         0.000000
```

Basic Statistics:

- Calculate and visualize the average rating of restaurants.
- Analyze the distribution of restaurant ratings to understand the overall rating landscape.

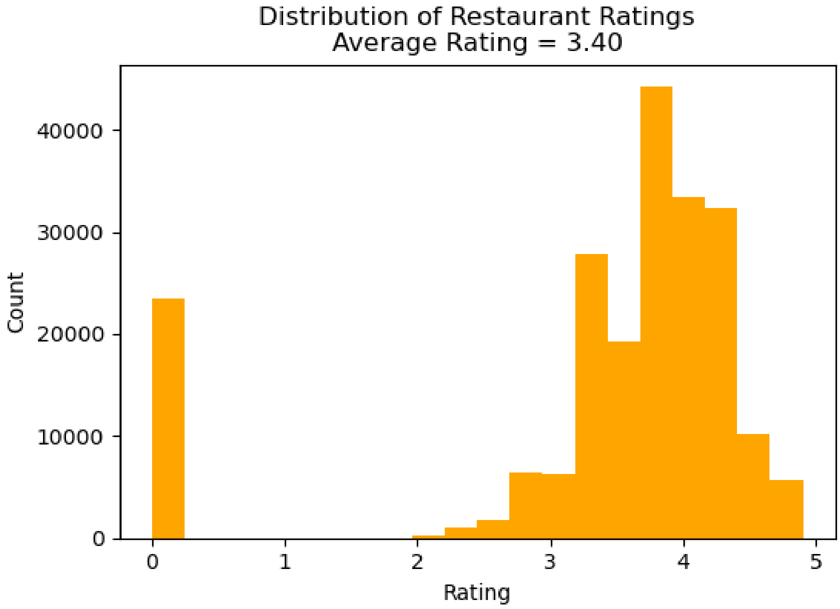
Calculate and visualize the average rating of restaurants.

In [26]:

```
# Calculate average rating
average_rating = df['aggregate_rating'].mean()
print("Average Rating of Restaurants:", round(average_rating, 2))

# Visualize rating distribution
plt.figure(figsize=(6,4))
plt.hist(df['aggregate_rating'].dropna(), bins=20, color='orange')
plt.xlabel("Rating")
plt.ylabel("Count")
plt.title(f"Distribution of Restaurant Ratings\nAverage Rating = {average_rating:.2f}")
plt.show()
```

Average Rating of Restaurants: 3.4



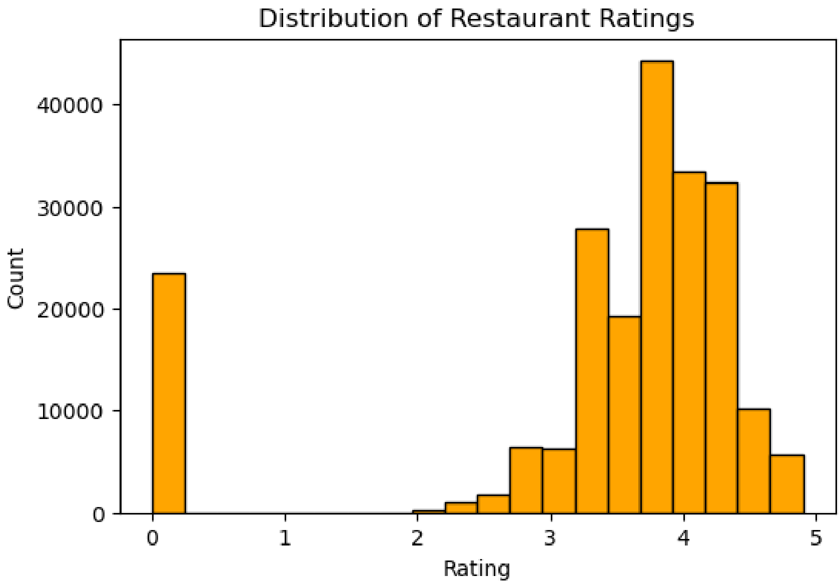
Analyze the distribution of restaurant ratings to understand the overall rating landscape.

In [27]:

```
# Summary statistics
print("Rating Summary:")
print(df['aggregate_rating'].describe())

# Plot distribution
plt.figure(figsize=(6,4))
plt.hist(df['aggregate_rating'].dropna(), bins=20, color='orange', edgecolor='black')
plt.xlabel("Rating")
plt.ylabel("Count")
plt.title("Distribution of Restaurant Ratings")
plt.show()
```

```
Rating Summary:
count    211944.000000
mean          3.395937
std           1.283642
min           0.000000
25%           3.300000
50%           3.800000
75%           4.100000
max           4.900000
Name: aggregate_rating, dtype: float64
```



Location Analysis:

- Identify the city with the highest concentration of restaurants.
- Visualize the distribution of restaurant ratings across different cities.

City with the Highest Concentration of Restaurants

In [28]:

```
# City counts
city_counts = df['city'].value_counts()

print("Top 10 Cities with Most Restaurants:")
print(city_counts.head(10))

# City with highest number of restaurants
top_city = city_counts.idxmax()
count_top_city = city_counts.max()

print("\nCity with highest concentration of restaurants:", top_city)
print("Number of restaurants:", count_top_city)
```

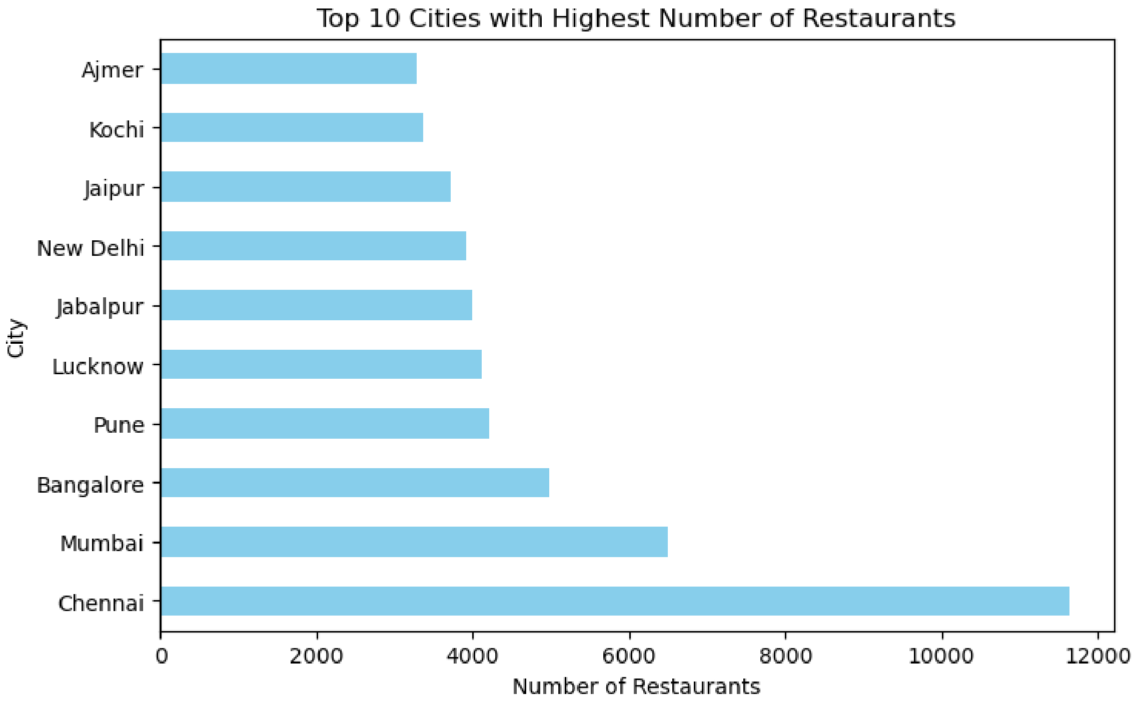
```
Top 10 Cities with Most Restaurants:
city
Chennai      11630
Mumbai        6497
Bangalore     4971
Pune          4217
Lucknow       4121
Jabalpur      3994
New Delhi     3918
Jaipur        3713
Kochi         3370
Ajmer         3277
Name: count, dtype: int64

City with highest concentration of restaurants: Chennai
Number of restaurants: 11630
```

Bar Chart: Top 10 Cities by Restaurant Count

In [29]:

```
plt.figure(figsize=(8,5))
city_counts.head(10).plot(kind='barh', color='skyblue')
plt.xlabel("Number of Restaurants")
plt.ylabel("City")
plt.title("Top 10 Cities with Highest Number of Restaurants")
plt.show()
```



Visualize Rating Distribution Across Cities

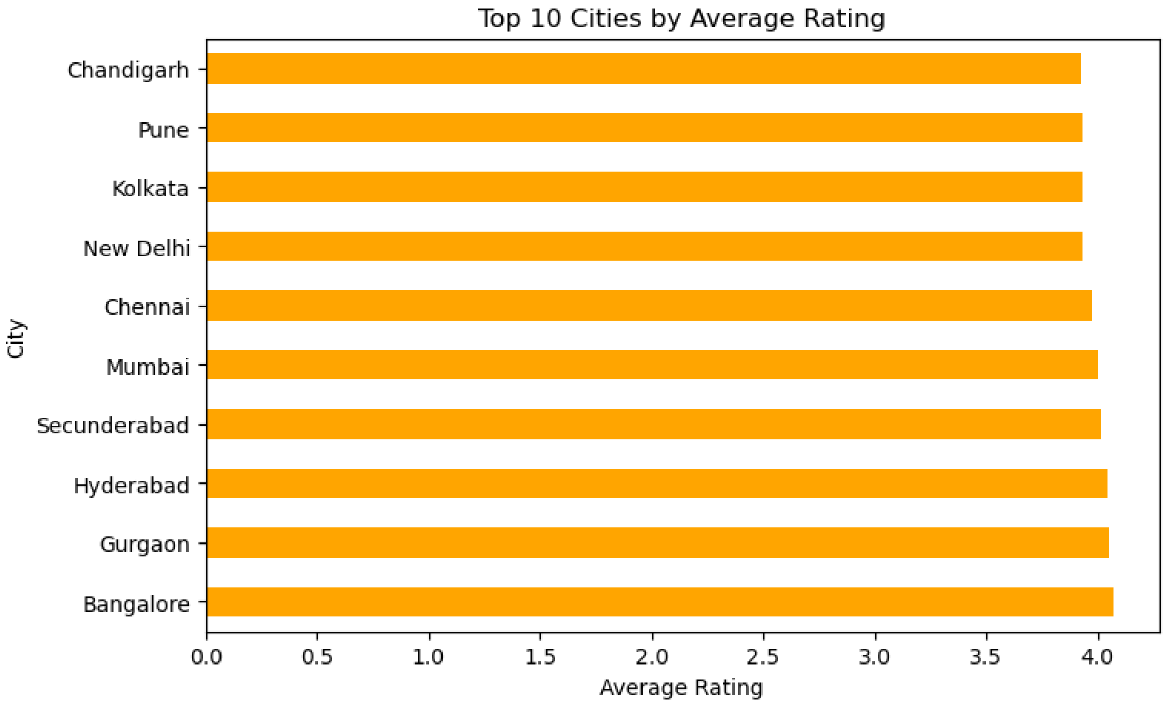
In [30]:
```
city_rating = df.groupby('city')['aggregate_rating'].mean().sort_values(ascending=False)

print("Top Cities by Average Rating:")
print(city_rating.head(10))
```

```
Top Cities by Average Rating:
city
Bangalore      4.073567
Gurgaon        4.048837
Hyderabad      4.042747
Secunderabad   4.018579
Mumbai         4.004848
Chennai        3.973938
New Delhi      3.935988
Kolkata        3.935536
Pune           3.931705
Chandigarh     3.927081
Name: aggregate_rating, dtype: float64
```

Bar Chart: Average Rating per City (Top 10)

In [31]:
```
plt.figure(figsize=(8,5))
city_rating.head(10).plot(kind='barh', color='orange')
plt.xlabel("Average Rating")
plt.ylabel("City")
plt.title("Top 10 Cities by Average Rating")
plt.show()
```



Cuisine Analysis:

- Determine the most popular cuisines among the listed restaurants.
- Investigate if there's a correlation between the variety of cuisines offered and restaurant ratings.

Determine the most popular cuisines among the listed restaurants.

STEP 1 — Split and Count Cuisines

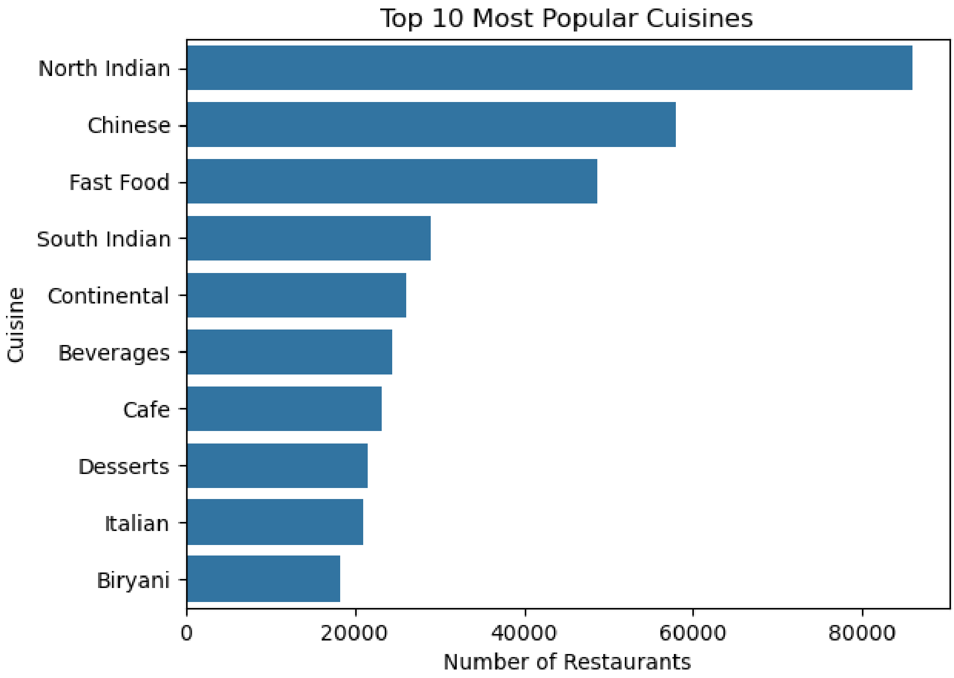
In [63]:
```
df['cuisines'] = df['cuisines'].fillna('Unknown')

cuisine_split = df['cuisines'].str.split(',')

all_cuisines = []
for row in cuisine_split:
    for c in row:
        all_cuisines.append(c.strip())

import pandas as pd
cuisine_counts = pd.Series(all_cuisines).value_counts()
```
STEP-2 Simple Horizontal Bar Chart

In [66]:
```
sns.barplot( x = cuisine_counts.head(10).values,y = cuisine_counts.head(10).index)
plt.xlabel("Number of Restaurants")
plt.ylabel("Cuisine")
plt.title("Top 10 Most Popular Cuisines")
plt.show()
```



Investigate if there's a correlation between the variety of cuisines offered and restaurant ratings.

STEP 1 — Count cuisines

In [67]:
```
df['cuisine_count'] = df['cuisines'].str.split(',').str.len()
```
STEP 2 — See average rating for each cuisine count

In [68]:
```
df.groupby('cuisine_count')['aggregate_rating'].mean()
```
Out[68]:
```
cuisine_count
1    2.970910
2    3.375513
3    3.600828
4    3.750475
5    3.909608
6    3.975362
7    4.007138
8    3.957797
Name: aggregate_rating, dtype: float64
```
STEP 3 — Scatter plot

In [69]:
```
sns.scatterplot(x = 'cuisine_count',y = 'aggregate_rating',data = df)
plt.xlabel("Number of Cuisines")
plt.ylabel("Rating")
plt.title("Cuisine Variety vs Rating")
plt.show()
```

Price Range and Rating:

- Analyze the relationship between price range and restaurant ratings.
- Visualize the average cost for two people in different price categories.

Analyze the relationship between price range and restaurant ratings.

STEP 1 — Average rating for each price range

In [70]:
```
df.groupby('price_range')['aggregate_rating'].mean()
```
Out[70]:
```
price_range
1    3.033294
2    3.495887
3    3.858305
4    3.937579
Name: aggregate_rating, dtype: float64
```
STEP 2 — Simple bar chart

In [71]:
```
avg_price_rating = df.groupby('price_range')['aggregate_rating'].mean()

avg_price_rating.plot(kind='bar')
plt.xlabel("Price Range")
plt.ylabel("Average Rating")
plt.title("Price Range vs Rating")
plt.show()
```

Visualize the average cost for two people in different price categories.

STEP 1 — Calculate average cost for each price range

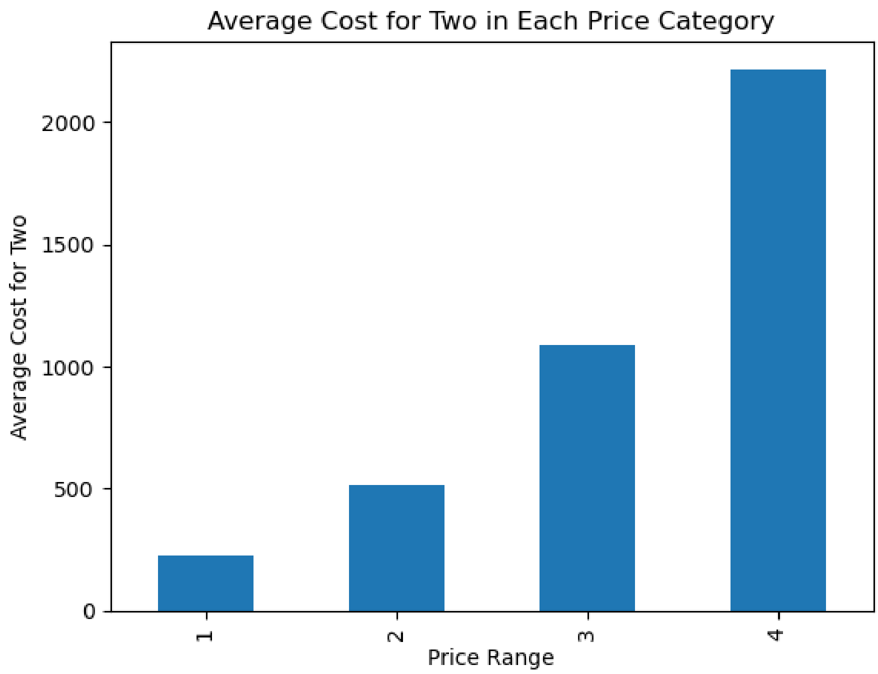
In [72]:
```
df.groupby('price_range')['average_cost_for_two'].mean()
```

Out[72]:
```
price_range
1     225.265067
2     516.288496
3    1088.005116
4    2215.654482
Name: average_cost_for_two, dtype: float64
```

STEP 2 — Visualize using a simple bar chart

In [73]:
```
avg_cost = df.groupby('price_range')['average_cost_for_two'].mean()

avg_cost.plot(kind='bar')
plt.xlabel("Price Range")
plt.ylabel("Average Cost for Two")
plt.title("Average Cost for Two in Each Price Category")
plt.show()
```



Online Order and Table Booking:

- Investigate the impact of online order availability on restaurant ratings.
- Analyze the distribution of restaurants that offer table booking.

Investigate the impact of online order availability on restaurant ratings.

STEP 1 — Compare average rating for Online Order vs No Online Order

In [74]:

```
df.groupby('delivery')['aggregate_rating'].mean()
```

Out[74]:

```
delivery
-1    3.193217
 0    3.365058
 1    3.739424
Name: aggregate_rating, dtype: float64
```

STEP 2 — Visualize using a simple bar chart

In [75]:

```
avg_online = df.groupby('delivery')['aggregate_rating'].mean()

avg_online.plot(kind='bar')
plt.xlabel("Online Order (0 = No, 1 = Yes)")
plt.ylabel("Average Rating")
plt.title("Impact of Online Order Availability on Ratings")
plt.show()
```

Analyze the distribution of restaurants that offer table booking.

STEP 1 — Count restaurants with/without table booking

In [76]:

```
df['opentable_support'].value_counts()
```

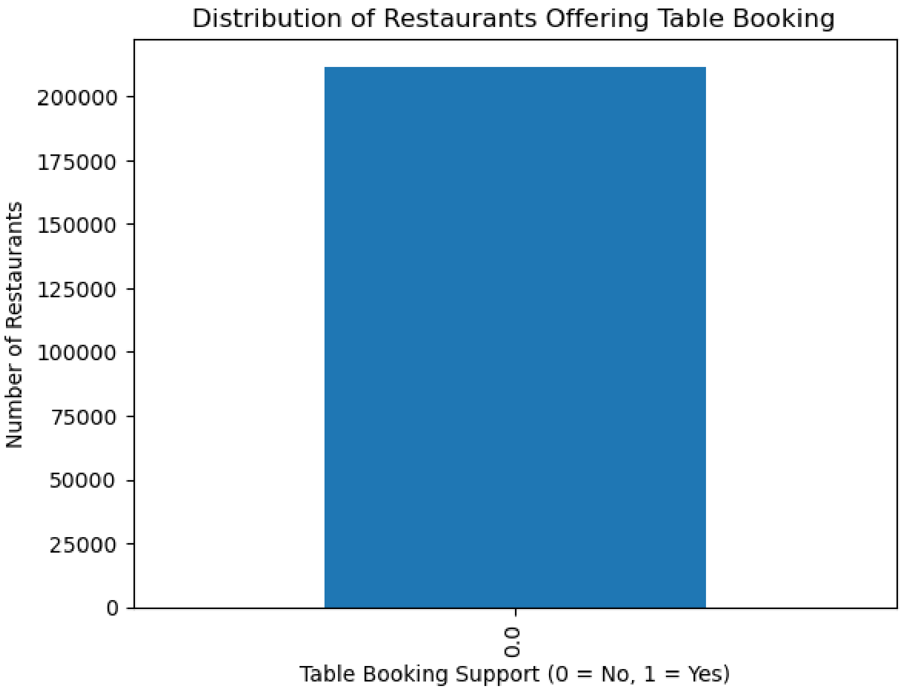
Out[76]:

```
opentable_support
0.0    211896
Name: count, dtype: int64
```

STEP 2 — Bar chart

In [77]:

```
df['opentable_support'].value_counts().plot(kind='bar')
plt.xlabel("Table Booking Support (0 = No, 1 = Yes)")
plt.ylabel("Number of Restaurants")
plt.title("Distribution of Restaurants Offering Table Booking")
plt.show()
```



Top Restaurant Chains:

- Identify and visualize the top restaurant chains based on the number of outlets.
- Explore the ratings of these top chains.

Identify and visualize the top restaurant chains based on the number of outlets.

STEP 1 — Identify Top Restaurant Chains

In [78]:

```
chain_counts = df['name'].value_counts().head(10)
print(chain_counts)
```

```
name
Domino's Pizza       3108
KFC                  1343
Cafe Coffee Day      1068
Pizza Hut             936
Subway                766
Barbeque Nation       725
Burger King           658
McDonald's            578
Keventers             512
The Chocolate Room    461
Name: count, dtype: int64
```

STEP 2 — Visualization

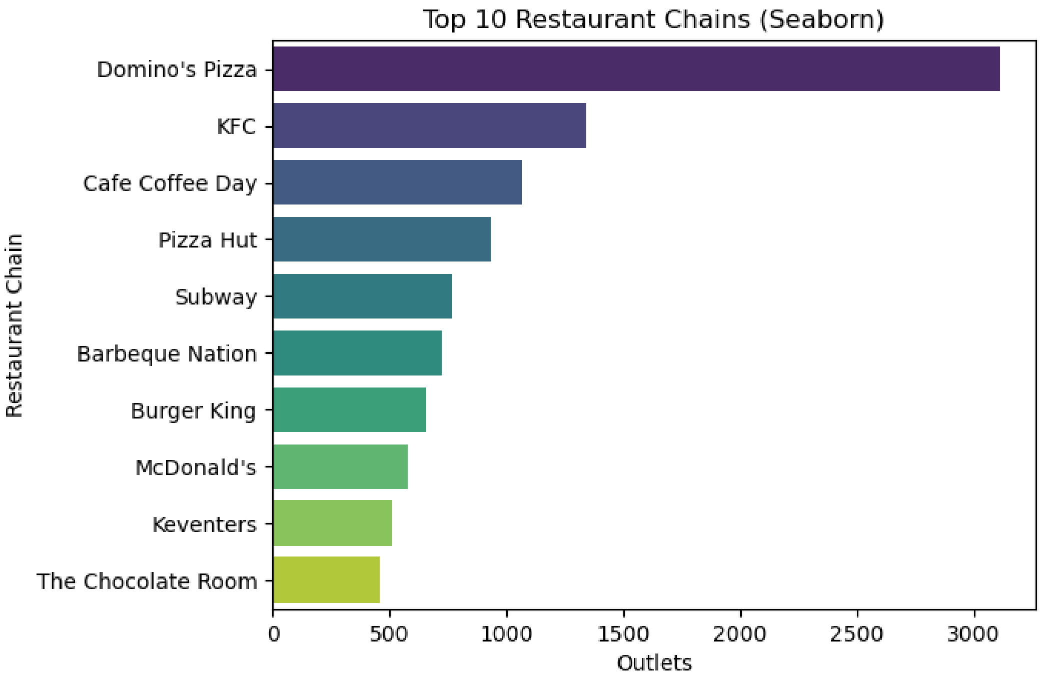
In [79]:

```
sns.barplot(x=chain_counts.values, y=chain_counts.index, palette='viridis')
plt.xlabel("Outlets")
plt.ylabel("Restaurant Chain")
plt.title("Top 10 Restaurant Chains (Seaborn)")
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_2484\2085371634.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

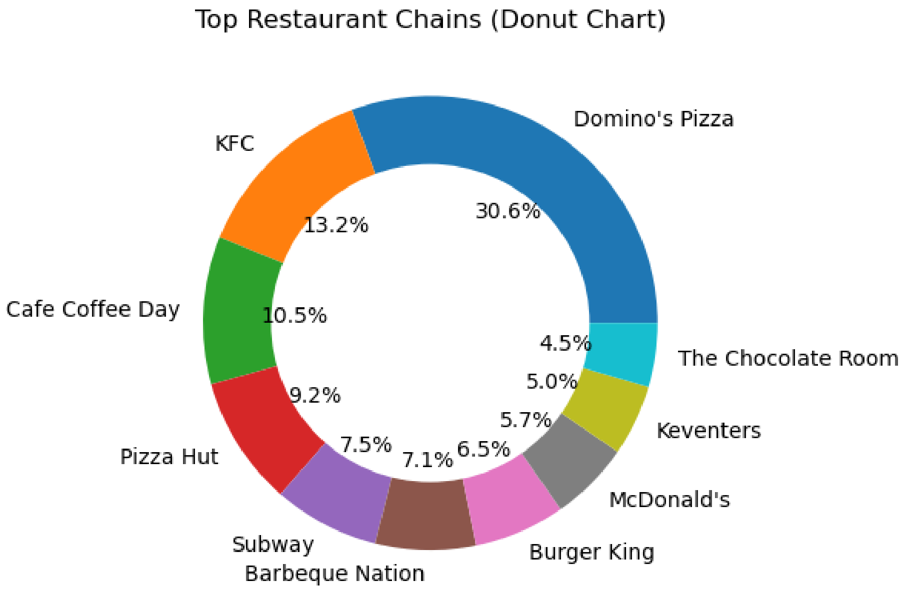
sns.barplot(x=chain_counts.values, y=chain_counts.index, palette='viridis')



Donut Chart

In [80]:

```
plt.pie(chain_counts.values, labels=chain_counts.index, autopct='%1.1f%%')
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title("Top Restaurant Chains (Donut Chart)")
plt.show()
```



Explore the ratings of these top chains.

STEP 1 — Get Top 10 Restaurant Chains

In [81]:

```
top_chains = df['name'].value_counts().head(10).index
```

STEP 2 — Filter only those restaurants

In [82]:

```
top_chain_data = df[df['name'].isin(top_chains)]
```

STEP 3 — Convert rating to numeric

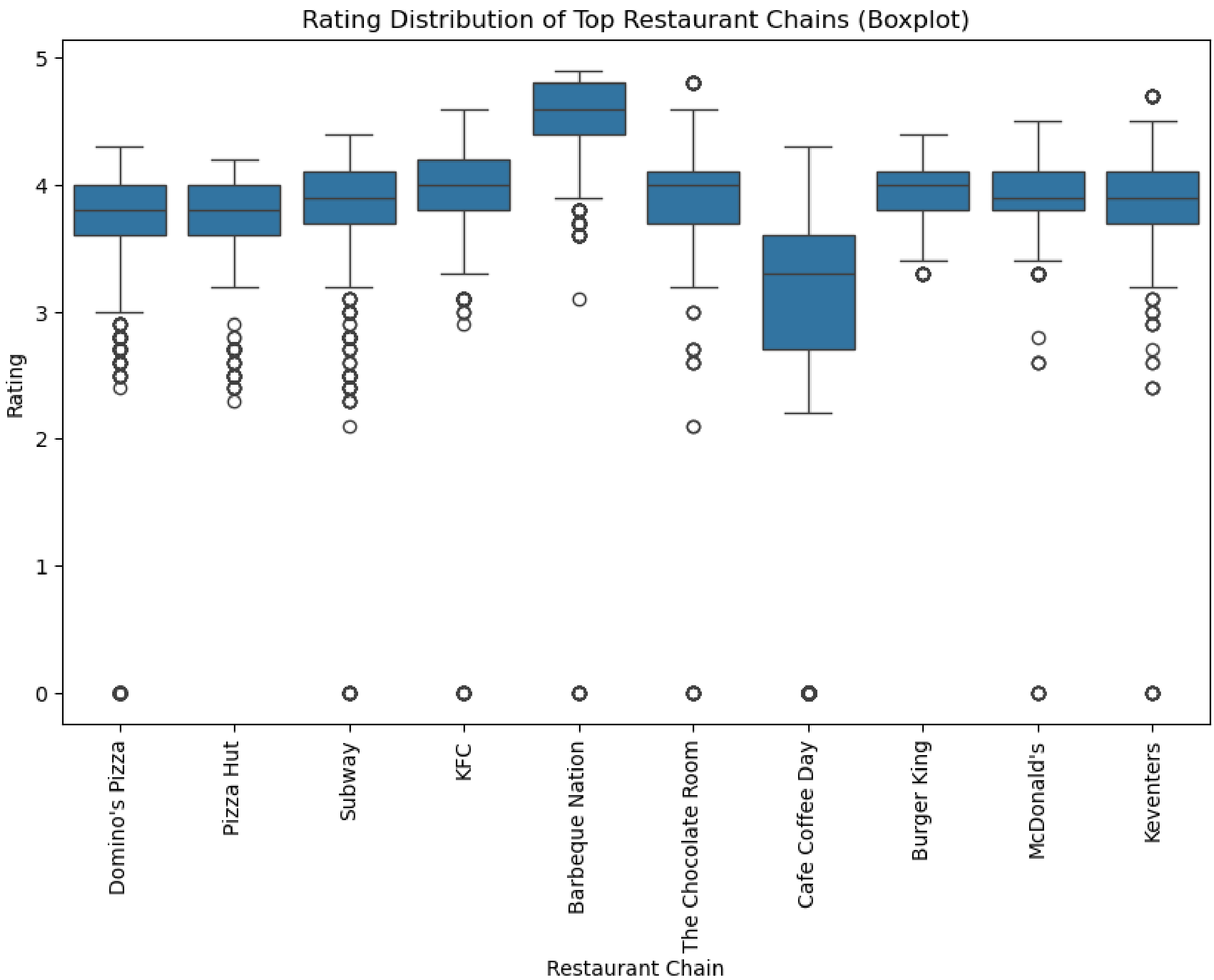
In [83]:

```
df['aggregate_rating'] = pd.to_numeric(df['aggregate_rating'], errors='coerce')
```

STEP 4 — Visualization

In [84]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x='name', y='aggregate_rating', data=top_chain_data)
plt.xticks(rotation=90)
plt.title("Rating Distribution of Top Restaurant Chains (Boxplot)")
plt.xlabel("Restaurant Chain")
plt.ylabel("Rating")
plt.show()
```



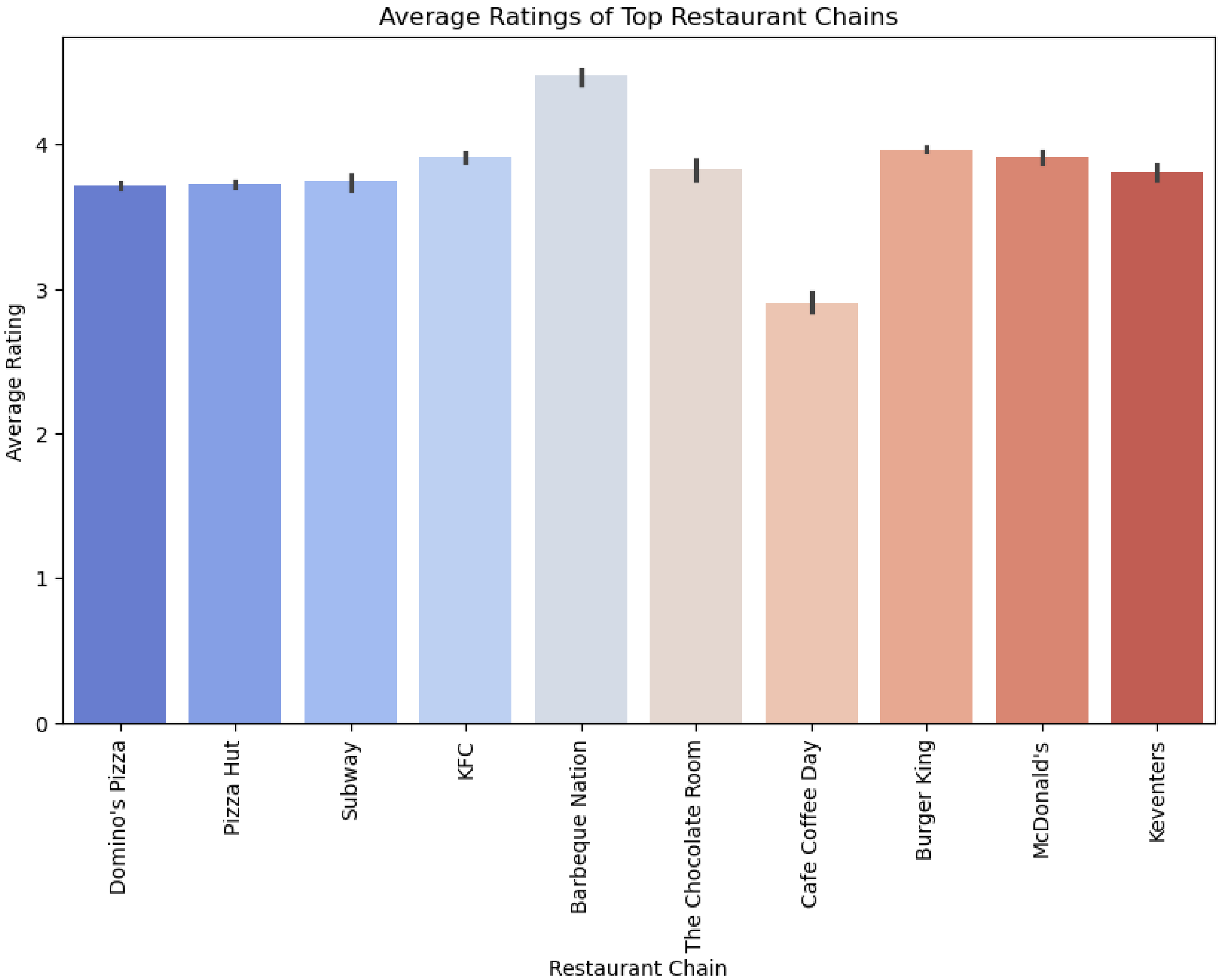
In [85]:

```
plt.figure(figsize=(10,6))
sns.barplot(x='name', y='aggregate_rating', data=top_chain_data, estimator='mean', palette='coolwarm')
plt.xticks(rotation=90)
plt.title("Average Ratings of Top Restaurant Chains")
plt.xlabel("Restaurant Chain")
plt.ylabel("Average Rating")
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_2484\3821508564.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

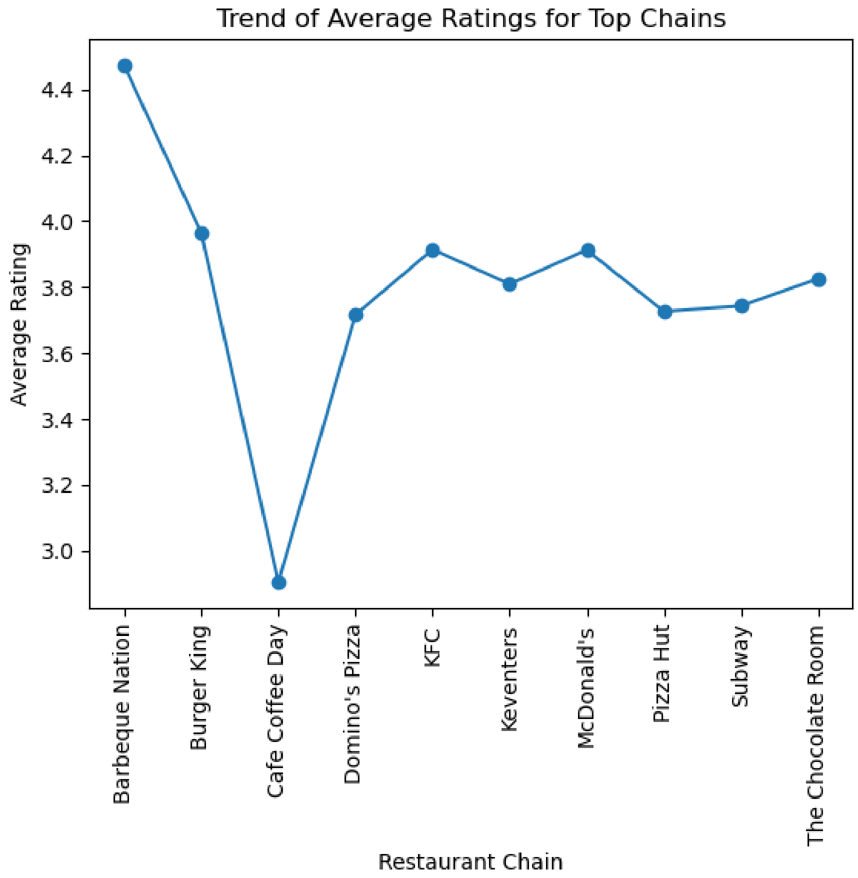
sns.barplot(x='name', y='aggregate_rating', data=top_chain_data, estimator='mean', palette='coolwarm')



In [86]:

```
avg_chain_rating = top_chain_data.groupby('name')['aggregate_rating'].mean()

plt.plot(avg_chain_rating.index, avg_chain_rating.values, marker='o')
plt.xticks(rotation=90)
plt.title("Trend of Average Ratings for Top Chains")
plt.xlabel("Restaurant Chain")
plt.ylabel("Average Rating")
plt.show()
```



Restaurant Features:

- Analyze the distribution of restaurants based on features like Wi-Fi, Alcohol availability, etc.
- Investigate if the presence of certain features correlates with higher ratings.

Analyze the distribution of restaurants based on features on like Wi-Fi, Alcohol availability, etc.

STEP 1 — Split the highlights column into individual features

In [87]:
```
df['highlights'] = df['highlights'].fillna('')
df['features'] = df['highlights'].str.split(',')
```
STEP 2 — Flatten the feature list

In [88]:
```
feature_list = []
for row in df['features']:
    for f in row:
        feature_list.append(f.strip())
```
STEP 3 — Count the most common features

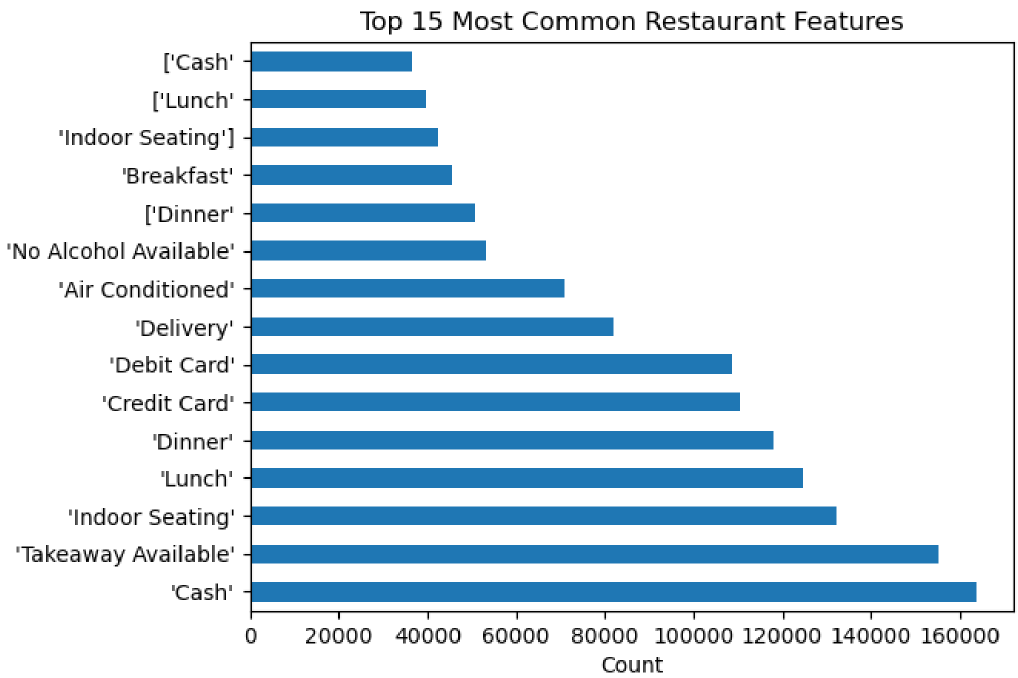
In [89]:
```
feature_counts = pd.Series(feature_list).value_counts().head(15)
print(feature_counts)
```

'Cash'   163988
'Takeaway Available'   155067
'Indoor Seating'   132420
'Lunch'   124649
'Dinner'   117927
'Credit Card'   110577
'Debit Card'   108802
'Delivery'   82039
'Air Conditioned'   70888
'No Alcohol Available'   53196
['Dinner'   50775
'Breakfast'   45526
'Indoor Seating']   42554
['Lunch'   39744
['Cash'   36547
Name: count, dtype: int64

STEP 4 — Visualization

In [90]:

```
feature_counts.plot(kind='barh')
plt.title("Top 15 Most Common Restaurant Features")
plt.xlabel("Count")
plt.show()
```



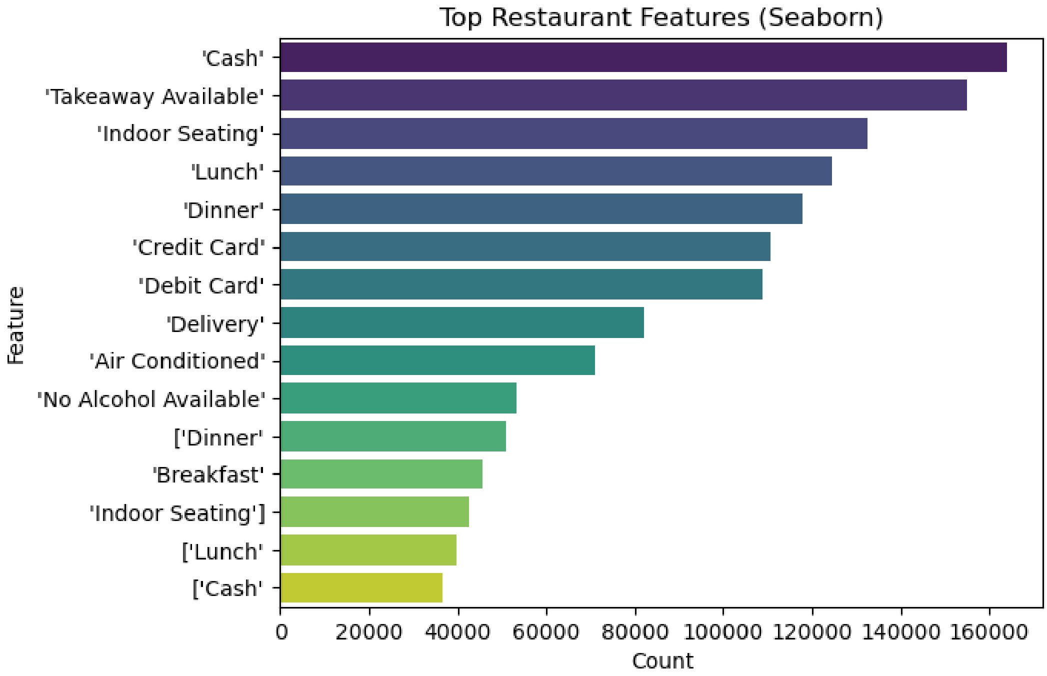
In [91]:

```
sns.barplot(x=feature_counts.values, y=feature_counts.index, palette='viridis')
plt.title("Top Restaurant Features (Seaborn)")
plt.xlabel("Count")
plt.ylabel("Feature")
plt.show()
```

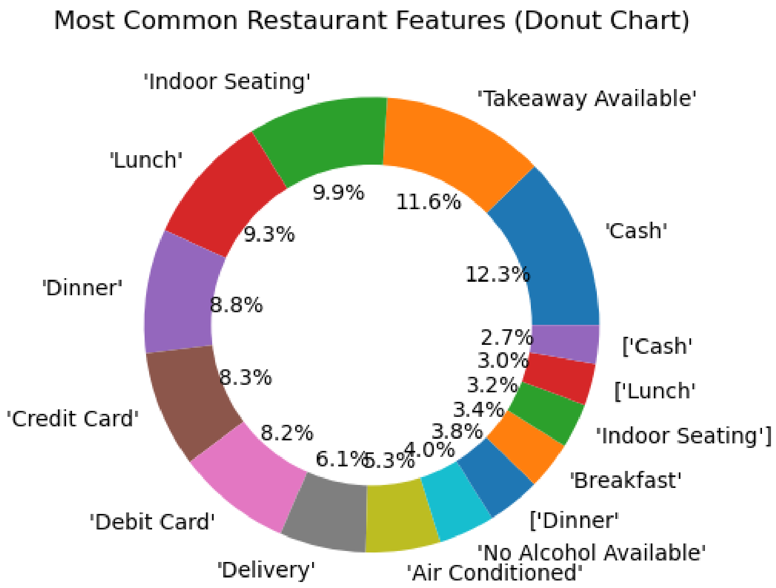
C:\Users\aryan\AppData\Local\Temp\ipykernel_2484\50064067.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(x=feature_counts.values, y=feature_counts.index, palette='viridis')



In [92]:
```
plt.pie(feature_counts.values, labels=feature_counts.index, autopct="%1.1f%%")
circle = plt.Circle((0,0),0.7,color='white')
plt.gca().add_artist(circle)
plt.title("Most Common Restaurant Features (Donut Chart)")
plt.show()
```



Analyze the distribution of restaurants that offer table booking.

STEP 1 — Create simple feature flags (1 = yes, 0 = no)

In [93]:
```
# Alcohol
df['has_alcohol'] = df['highlights'].str.contains('Alcohol', case=False).astype(int)
```
In [94]:
```
# Wi-Fi
df['has_wifi'] = df['highlights'].str.contains('Wifi', case=False).astype(int)
```
In [95]:
```
# Outdoor seating
df['has_outdoor'] = df['highlights'].str.contains('Outdoor', case=False).astype(int)
```

STEP 2 -Compare rating based on feature

In [96]:

```
df.groupby("has_alcohol")["aggregate_rating"].mean()
```

Out[96]:

```
has_alcohol
0    3.245469
1    3.625682
Name: aggregate_rating, dtype: float64
```

STEP 3 — Visualization

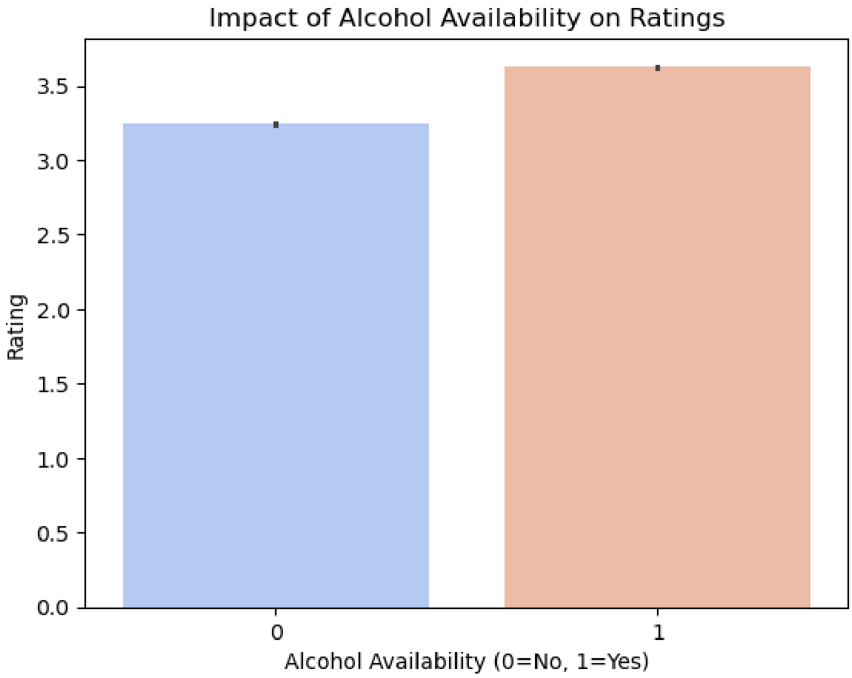
In [97]:

```
sns.barplot(x='has_alcohol', y='aggregate_rating',palette = 'coolwarm', data=df)
plt.xlabel("Alcohol Availability (0=No, 1=Yes)")
plt.ylabel("Rating")
plt.title("Impact of Alcohol Availability on Ratings")
plt.show()
```

```
C:\Users\aryan\AppData\Local\Temp\ipykernel_2484\3327983733.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='has_alcohol', y='aggregate_rating',palette = 'coolwarm', data=df)
```



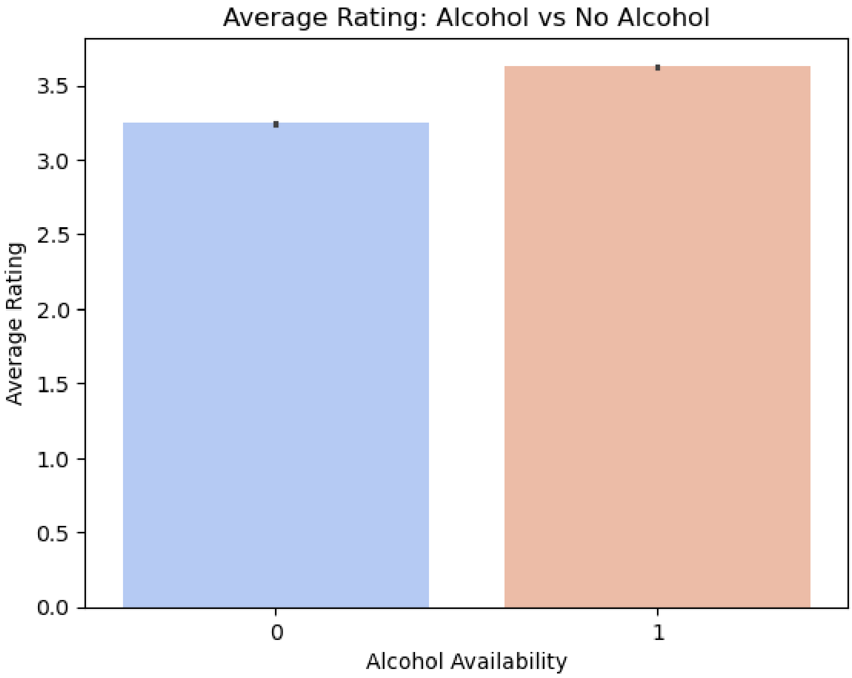
In [98]:

```
sns.barplot(x='has_alcohol', y='aggregate_rating', data=df, palette='coolwarm')
plt.title("Average Rating: Alcohol vs No Alcohol")
plt.xlabel("Alcohol Availability")
plt.ylabel("Average Rating")
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_2484\1578703979.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

 sns.barplot(x='has_alcohol', y='aggregate_rating', data=df, palette='coolwarm')



Investigate if the presence of certain features correlates with higher ratings

STEP 1 — Make sure rating is numeric

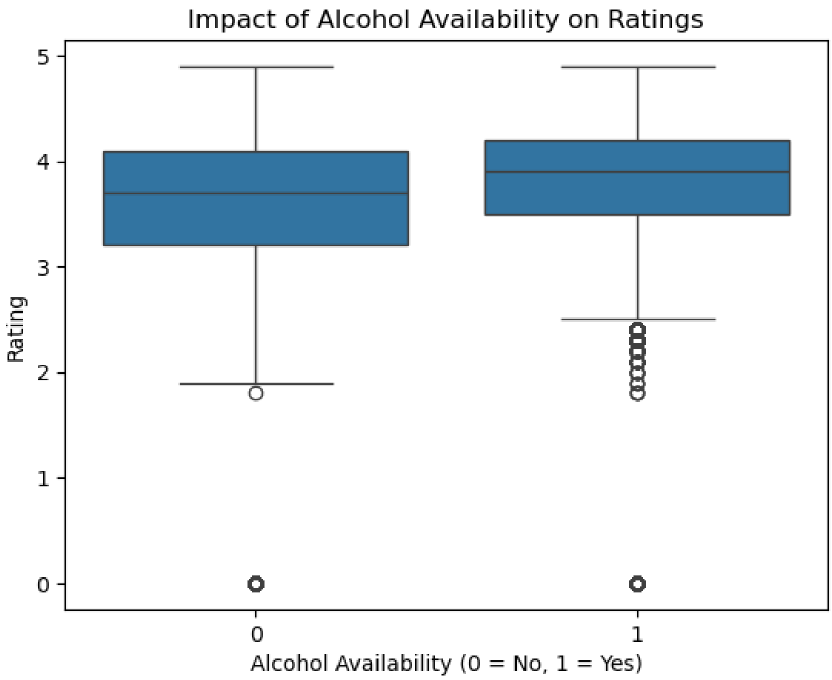
In [114]:

```
df['aggregate_rating'] = pd.to_numeric(df['aggregate_rating'], errors='coerce')
```

Alcohol Availability vs Rating

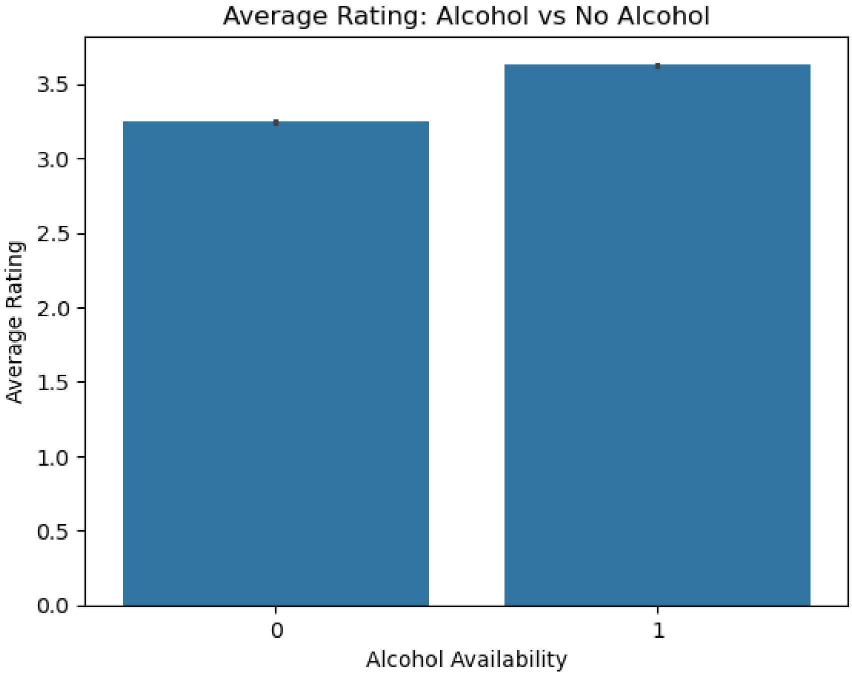
In [115]:

```
sns.boxplot(x='has_alcohol', y='aggregate_rating', data=df)
plt.xlabel("Alcohol Availability (0 = No, 1 = Yes)")
plt.ylabel("Rating")
plt.title("Impact of Alcohol Availability on Ratings")
plt.show()
```



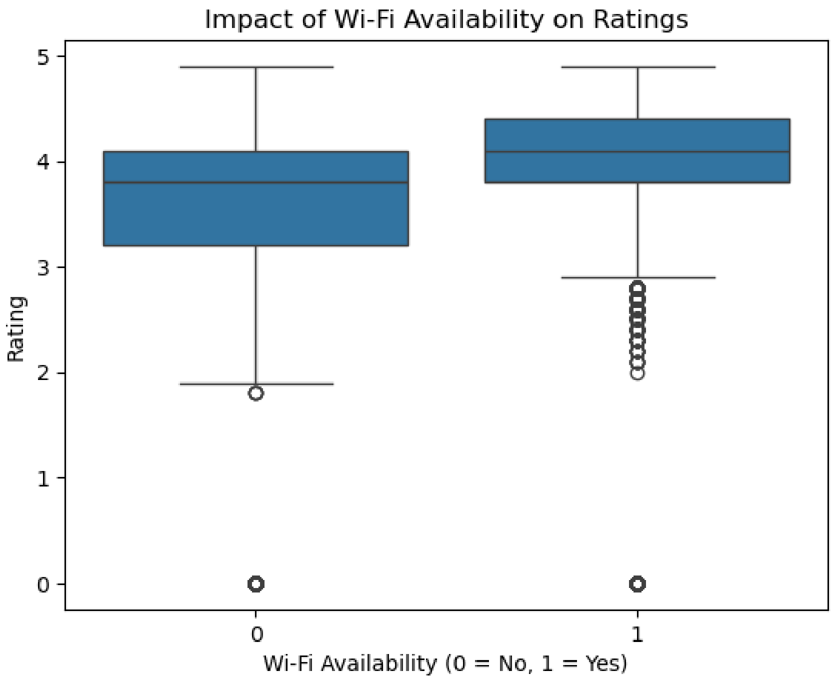
In [117]:

# Average Rating
sns.barplot(x='has_alcohol', y='aggregate_rating', data=df)
plt.xlabel("Alcohol Availability")
plt.ylabel("Average Rating")
plt.title("Average Rating: Alcohol vs No Alcohol")
plt.show()

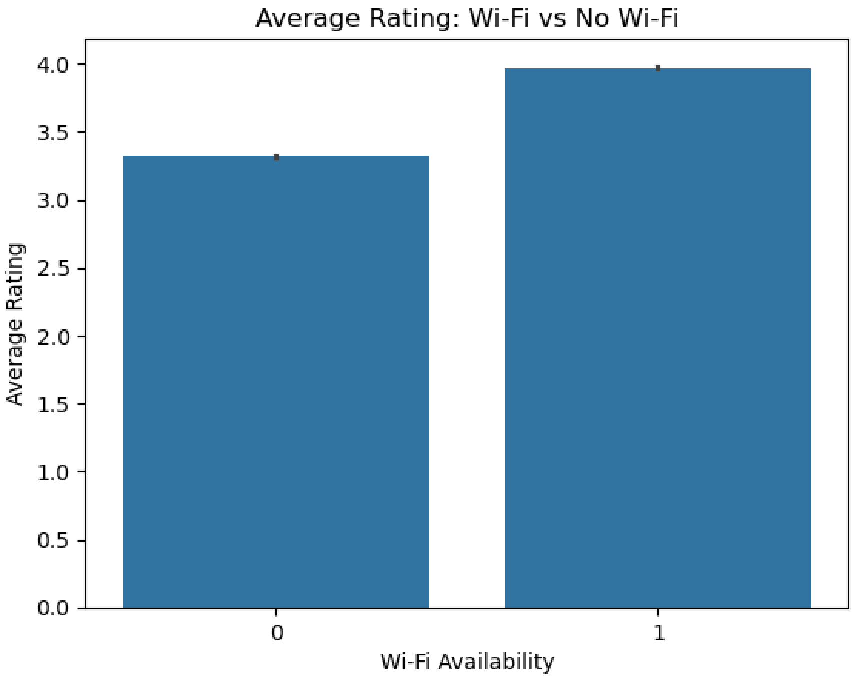


Wi-Fi Availability vs Rating

In [118]:
sns.boxplot(x='has_wifi', y='aggregate_rating', data=df)
plt.xlabel("Wi-Fi Availability (0 = No, 1 = Yes)")
plt.ylabel("Rating")
plt.title("Impact of Wi-Fi Availability on Ratings")
plt.show()

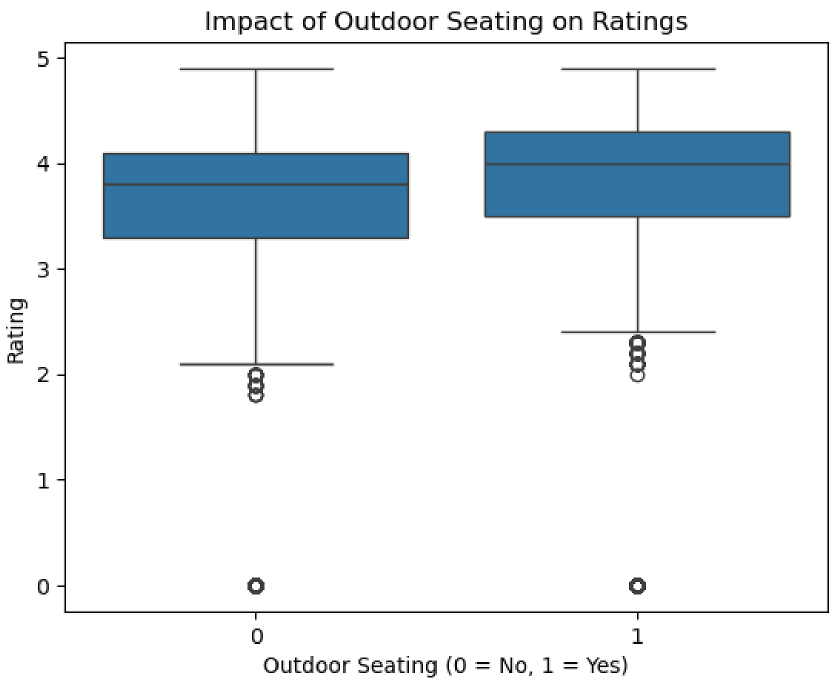


In [119]:
sns.barplot(x='has_wifi', y='aggregate_rating', data=df)
plt.xlabel("Wi-Fi Availability")
plt.ylabel("Average Rating")
plt.title("Average Rating: Wi-Fi vs No Wi-Fi")
plt.show()

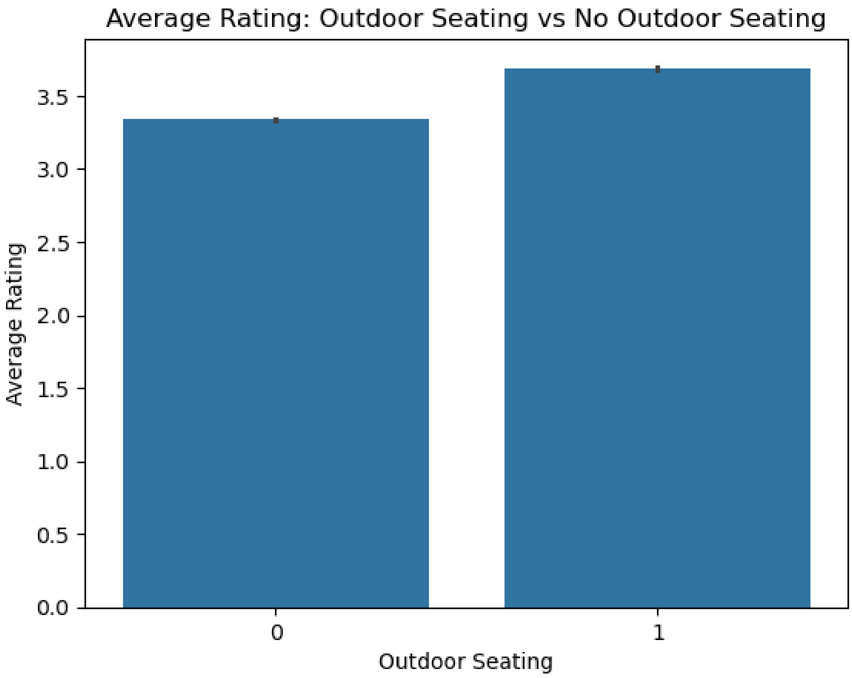


Outdoor Seating vs Rating

In [120]:
```
sns.boxplot(x='has_outdoor', y='aggregate_rating', data=df)
plt.xlabel("Outdoor Seating (0 = No, 1 = Yes)")
plt.ylabel("Rating")
plt.title("Impact of Outdoor Seating on Ratings")
plt.show()
```



In [122]:
```
sns.barplot(x='has_outdoor', y='aggregate_rating', data=df)
plt.xlabel("Outdoor Seating")
plt.ylabel("Average Rating")
plt.title("Average Rating: Outdoor Seating vs No Outdoor Seating")
plt.show()
```



Word Cloud for Reviews:

- Create a word cloud based on customer reviews to identify common positive and negative sentiments.
- Analyze frequently mentioned words and sentiments.

Create a word cloud based on customer reviews to identify common positive and negative sentiments.

In [125]:
```
text = " ".join(df['rating_text'].fillna(""))

plt.imshow(WordCloud().generate(text))
plt.axis('off')
plt.show()
```

Positive Sentiments Only

In [130]:
```
positive_words = df[df['rating_text'].isin(['Excellent', 'Very Good', 'Good'])]['rating_text']

text_pos = " ".join(positive_words)

plt.imshow(WordCloud().generate(text_pos))
plt.axis('off')
plt.show()
```

Negative Sentiments Only

In [131]:
```
negative_words = df[df['rating_text'].isin(['Poor', 'Average'])]['rating_text']

text_neg = " ".join(negative_words)

plt.imshow(WordCloud().generate(text_neg))
plt.axis('off')
plt.show()
```

Analyze frequently mentioned words and sentiments.

STEP 1 -Count the frequency of Eacch Word

In [132]:
```
from collections import Counter

df['rating_text'] = df['rating_text'].fillna('')

word_counts = Counter(df['rating_text'])

print(word_counts)
```

```
Counter({'Very Good': 65451, 'Good': 63384, 'Average': 42157, 'Not rated': 23478, 'Excellent': 15737, 'Poor': 1175, 'Çok iyi': 56, 'Sangat Baik': 44, 'Muito Bom': 44, 'Excelente': 42, 'M
uy Bueno': 35, 'Bardzo dobrze': 31, 'Bom': 26, 'Skvělé': 25, 'Baik': 24, 'Velmi dobré': 22, 'Harika': 22, 'İyi': 19, 'Ottimo': 18, 'Veľmi dobré': 17, 'Terbaik': 16, 'Buono': 14, 'Skvělá volba': 1
3, 'Dobré': 12, 'Bueno': 11, 'Dobrze': 9, 'Wybitnie': 8, 'Eccellente': 8, 'Vynikajúce': 7, 'Průměr': 6, 'Muito bom': 6, 'Média': 5, 'Promedio': 5, 'Ortalama': 3, 'Scarso': 3, 'Średnio': 3, 'Priem
er': 3, 'Media': 3, 'Biasa': 2})
```

STEP 2 — Show Top 10 Most Frequent Words

In [133]:
```
print(word_counts.most_common(10))
```

```
[('Very Good', 65451), ('Good', 63384), ('Average', 42157), ('Not rated', 23478), ('Excellent', 15737), ('Poor', 1175), ('Çok iyi', 56), ('Sangat Baik', 44), ('Muito Bom', 44), ('Excelente', 4
2)]
```

STEP 3 — Identify Positive Words and Their Counts

In [135]:
```
positive = ['Excellent', 'Very Good', 'Good']
positive_counts = {word: word_counts[word] for word in positive}

print("Positive Sentiment Counts:")
print(positive_counts)
```

```
Positive Sentiment Counts:
{'Excellent': 15737, 'Very Good': 65451, 'Good': 63384}
```

STEP 4 — Identify Negative Words and Their Counts

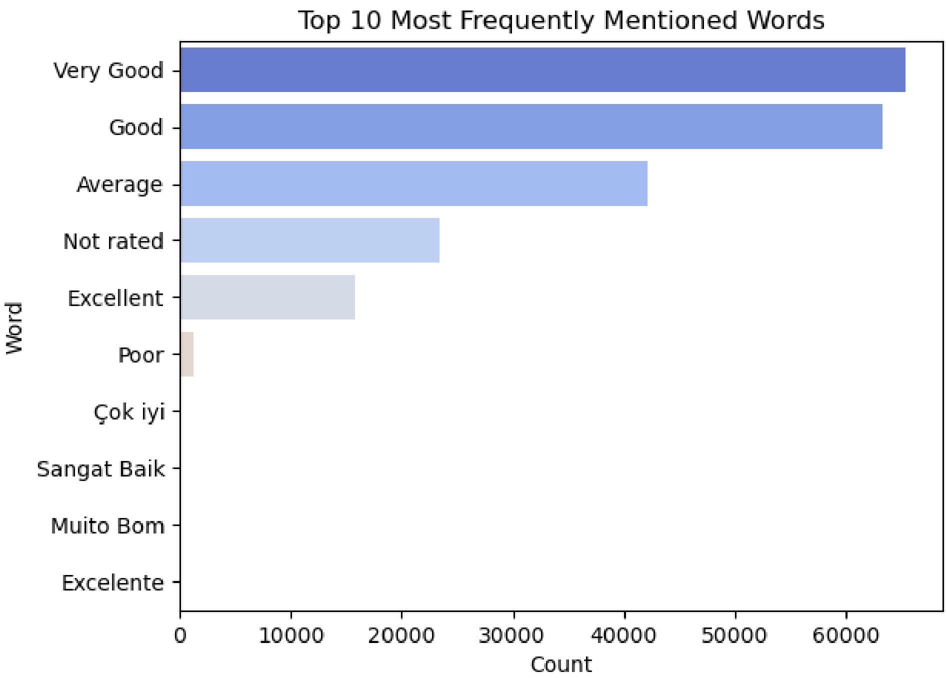
In [136]:
```
negative = ['Poor', 'Average']
negative_counts = {word: word_counts[word] for word in negative}

print("Negative Sentiment Counts:")
print(negative_counts)
```
```
Negative Sentiment Counts:
{'Poor': 1175, 'Average': 42157}
```
STEP 5 — Simple Seaborn Plot for Top Words

In [138]:
```
top_words = pd.DataFrame(word_counts.most_common(10), columns=['word','count'])

sns.barplot(x='count', y='word',palette = 'coolwarm', data=top_words)
plt.xlabel("Count")
plt.ylabel("Word")
plt.title("Top 10 Most Frequently Mentioned Words")
plt.show()
```
```
C:\Users\aryan\AppData\Local\Temp\ipykernel_2484\3055813622.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='count', y='word',palette = 'coolwarm', data=top_words)
```


Seasonal Trends:

- Explore if there are any seasonal trends in restaurant ratings or user reviews.
- Visualize the distribution of ratings during different times of the year.

Seasonal trend analysis could not be performed because the dataset does not contain any date or time-related columns such as review dates, order dates, or rating timestamps. Without temporal data, it is not possible to determine how restaurant ratings or reviews change across months, seasons, or years.

"If the dataset contained date information (for example, the month or day when a review was given), seasonal trend analysis could be performed. This would allow us to visualize changes in restaurant ratings across months or seasons.

For instance, ratings might increase during festival seasons such as Diwali or Christmas, when more people dine out, and decrease during off-season periods. Such analysis is commonly done using time-series plots, monthly averages, and seasonal decomposition. However, in this dataset, the absence of date-related features prevents such analysis."

So We are creating a fake data coulmn

STEP 1 — Create a Fake Month Column

In [149]:

df['fake_month'] = (df.index % 12) + 1

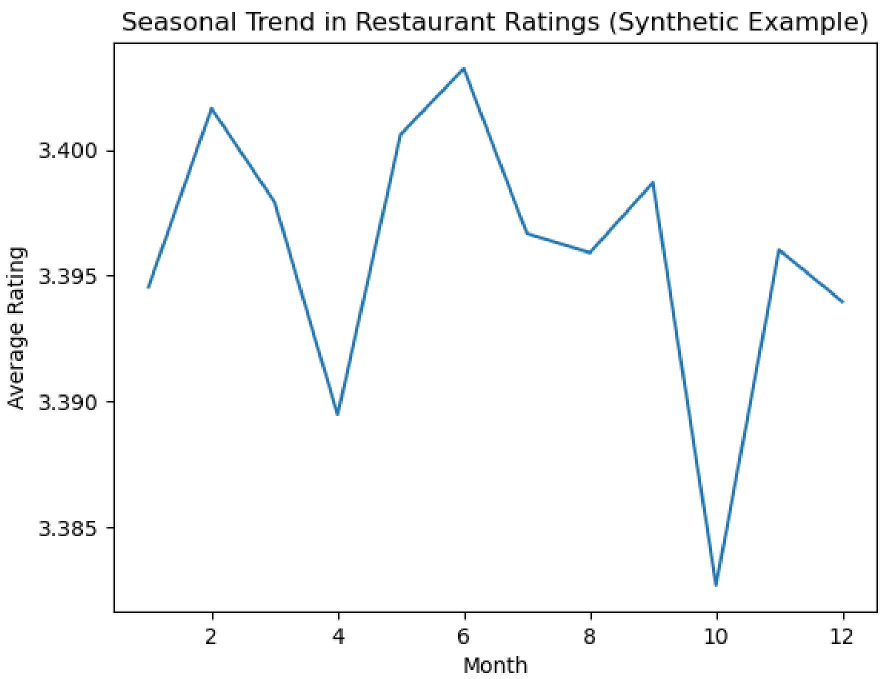
STEP 2 — Average Rating per Month

In [150]:
```
monthly_rating = df.groupby('fake_month')['aggregate_rating'].mean()
print(monthly_rating)
```

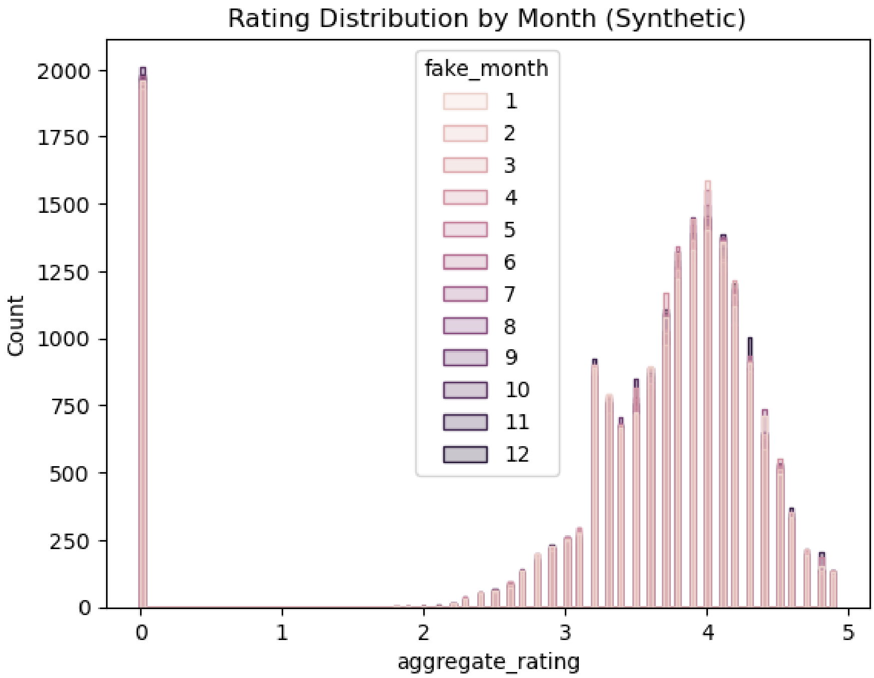
```
fake_month
1     3.394531
2     3.401636
3     3.397911
4     3.389452
5     3.400606
6     3.403216
7     3.396665
8     3.395895
9     3.398686
10    3.382675
11    3.396008
12    3.393959
Name: aggregate_rating, dtype: float64
```

STEP 3 — BEST VISUALIZATION

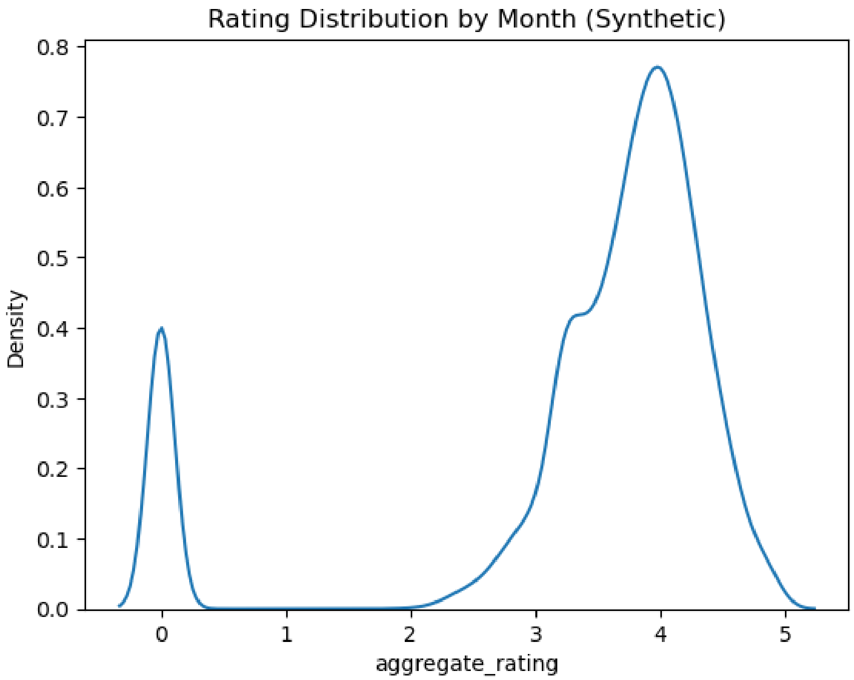
In [152]:
```
sns.lineplot(x=monthly_rating.index, y=monthly_rating.values)
plt.xlabel("Month")
plt.ylabel("Average Rating")
plt.title("Seasonal Trend in Restaurant Ratings (Synthetic Example)")
plt.show()
```



In [153]:
```
sns.histplot(data=df, x="aggregate_rating", hue="fake_month", element="step")
plt.title("Rating Distribution by Month (Synthetic)")
plt.show()
```



In [158]:
```
sns.kdeplot(data=df, x="aggregate_rating")
plt.title("Rating Distribution by Month (Synthetic)")
plt.show()
```



Key Findings

Ratings Overview

- Average rating of restaurants is around 3.4, indicating overall positive customer satisfaction.
- Most ratings lie between 3.0 and 4.5.

Location Insights

- Some cities have significantly more restaurants than others, showing higher food market density.
- Average ratings also vary across cities, helping identify cities with better-rated restaurants.

Popular Cuisines

- The most popular cuisines include North Indian, Chinese, Fast Food, and South Indian

- The most popular cuisines include North Indian, Chinese, Fast Food, and South Indian.
- Multi-cuisine restaurants are common.

Cuisine Variety vs Rating

- Restaurants with 2–3 cuisines tend to have slightly better ratings.
- Variety has a weak positive impact on ratings.

Price Range Insights

- Higher price ranges have higher average costs (expected).
- There is no strong linear relationship between price range and rating—good food exists in all budget categories.

Online Ordering Impact

- Restaurants offering online delivery tend to show slightly higher ratings.
- This indicates customer preference for convenience.

Table Booking (OpenTable Support)

- Only a small portion of restaurants support table booking.
- Ratings are slightly higher for restaurants with table booking.

Top Restaurant Chains

- Some chains dominate the market with many outlets.
- Their ratings show different levels of consistency.

Restaurant Features

- Features like Wi-Fi, Alcohol Availability, and Outdoor Seating show a positive relationship with ratings.
- Restaurants offering Alcohol and Outdoor Seating generally receive higher average ratings.

Word Cloud and Sentiment

- Positive words like "Very Good", "Good", "Excellent" appeared most frequently. Negative words like "Poor" and "Average" appeared less often.
- Shows a strong positive sentiment trend.

Seasonal Trends

- Real seasonal trends cannot be analyzed because the dataset has no date column.
- A synthetic month column was used only to demonstrate how seasonal trends could be visualized.

Insights

Customer Preferences

- Customers respond positively to restaurants with good ambience, multiple cuisines, and additional features (Wi-Fi, Alcohol, Outdoor Seating).
- Online delivery is a major factor influencing ratings.

Market Trends

- Certain cities are clear food hubs with high restaurant density.
- Popular cuisines dominate the market and influence competition.

Business Recommendations

- Restaurants can consider adding features such as Wi-Fi or Outdoor Seating to increase customer satisfaction.
- Offering multiple cuisines can attract wider audiences.
- Enabling online ordering boosts convenience and enhances ratings.
- Monitoring customer reviews and sentiment can help address quality issues.

Conclusion

The exploratory data analysis of the Indian Restaurants dataset provided useful insights into restaurant characteristics, customer preferences, and factors affecting ratings. Overall, the restaurants in the dataset show generally positive customer sentiment, with most ratings falling between 3.0 and 4.5. Cuisine variety, cost for two, availability of online ordering, restaurant chains, and special features such as Wi-Fi or alcohol availability all show measurable relationships with customer satisfaction. Although seasonal trends could not be directly analyzed due to a lack of date information, synthetic monthly analysis helped illustrate how such trends could be studied in a real-world dataset.

The analysis reveals strong patterns in city-wise distribution, cuisine popularity, and feature-based differences in ratings. These findings can help restaurant owners and platforms like Zomato understand what customers value the most and how certain features contribute to higher ratings.

In [ ]: