price-range-analysis

February 11, 2024

Importing Library and dataset:

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# Set the style for Seaborn plots
sns.set(style="whitegrid")
```

[3]:
```
df = pd.read_csv("/content/Dataset .csv")
df.head()
```

[3]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude
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…	121.027535	14.565443
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…	121.014101	14.553708
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…	121.056831	14.581404
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…	121.056475	14.585318
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas…	SM Megamall, Ortigas, Mandaluyong City			

```
4  SM Megamall, Ortigas, Mandaluyong City, Mandal…  121.057508  14.584450

                          Cuisines  …          Currency Has Table booking  \
0       French, Japanese, Desserts  …  Botswana Pula(P)               Yes
1                         Japanese  …  Botswana Pula(P)               Yes
2  Seafood, Asian, Filipino, Indian  …  Botswana Pula(P)               Yes
3                  Japanese, Sushi  …  Botswana Pula(P)                No
4                 Japanese, Korean  …  Botswana Pula(P)               Yes

  Has Online delivery Is delivering now Switch to order menu Price range  \
0                  No                No                    No           3
1                  No                No                    No           3
2                  No                No                    No           4
3                  No                No                    No           4
4                  No                No                    No           4

   Aggregate rating  Rating color Rating text Votes
0               4.8    Dark Green   Excellent   314
1               4.5    Dark Green   Excellent   591
2               4.4         Green   Very Good   270
3               4.9    Dark Green   Excellent   365
4               4.8    Dark Green   Excellent   229

[5 rows x 21 columns]
```

[4]:
```
df.describe()
```

[4]:
```
       Restaurant ID  Country Code    Longitude     Latitude  \
count   9.551000e+03   9551.000000  9551.000000  9551.000000
mean    9.051128e+06     18.365616    64.126574    25.854381
std     8.791521e+06     56.750546    41.467058    11.007935
min     5.300000e+01      1.000000  -157.948486   -41.330428
25%     3.019625e+05      1.000000    77.081343    28.478713
50%     6.004089e+06      1.000000    77.191964    28.570469
75%     1.835229e+07      1.000000    77.282006    28.642758
max     1.850065e+07    216.000000   174.832089    55.976980

       Average Cost for two  Price range  Aggregate rating         Votes
count           9551.000000  9551.000000       9551.000000   9551.000000
mean            1199.210763     1.804837          2.666370    156.909748
std            16121.183073     0.905609          1.516378    430.169145
min                0.000000     1.000000          0.000000      0.000000
25%              250.000000     1.000000          2.500000      5.000000
50%              400.000000     2.000000          3.200000     31.000000
75%              700.000000     2.000000          3.700000    131.000000
max           800000.000000     4.000000          4.900000  10934.000000
```

[5]:
```
df.dtypes
```

[5]:
```
Restaurant ID             int64
Restaurant Name          object
Country Code              int64
City                     object
Address                  object
Locality                 object
Locality Verbose         object
Longitude               float64
Latitude                float64
Cuisines                 object
Average Cost for two      int64
Currency                 object
Has Table booking        object
Has Online delivery      object
Is delivering now        object
Switch to order menu     object
Price range               int64
Aggregate rating        float64
Rating color              object
Rating text               object
Votes                     int64
dtype: object
```

Most common Price range:

[6]:
```
# Most common price range
most_common_price_range = df['Price range'].mode()[0]
print("\033[1mMost Common Price Range:\033[0m", most_common_price_range)
```

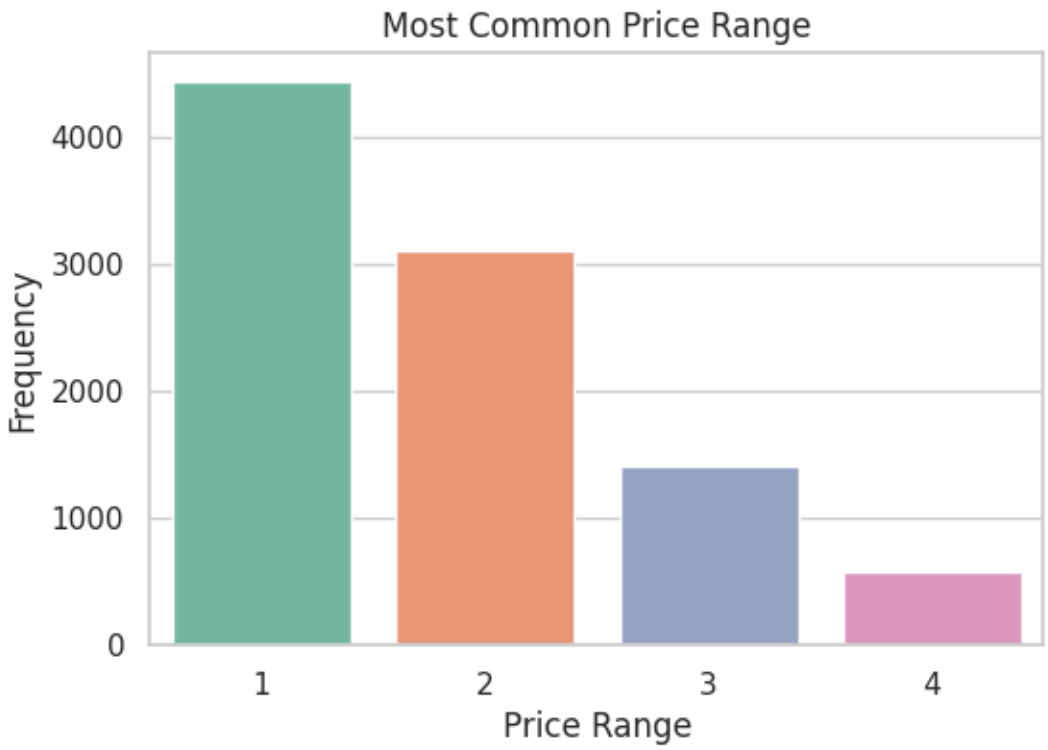
Most Common Price Range: 1

[7]:
```
# Most Common Price Range
plt.figure(figsize=(6, 4))
sns.countplot(x='Price range', data=df, palette="Set2")
plt.title('Most Common Price Range')
plt.xlabel('Price Range')
plt.ylabel('Frequency')
plt.show()
```

```
<ipython-input-7-03261245352d>:3: 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.countplot(x='Price range', data=df, palette="Set2")
```



Average rating for each Price range:

[8]:
```
# Average rating for each price range
avg_rating_by_price_range = df.groupby('Price range')['Aggregate rating'].mean()
print("\033[1mAverage Rating for Each Price Range:\033[0m")
print(avg_rating_by_price_range)
```

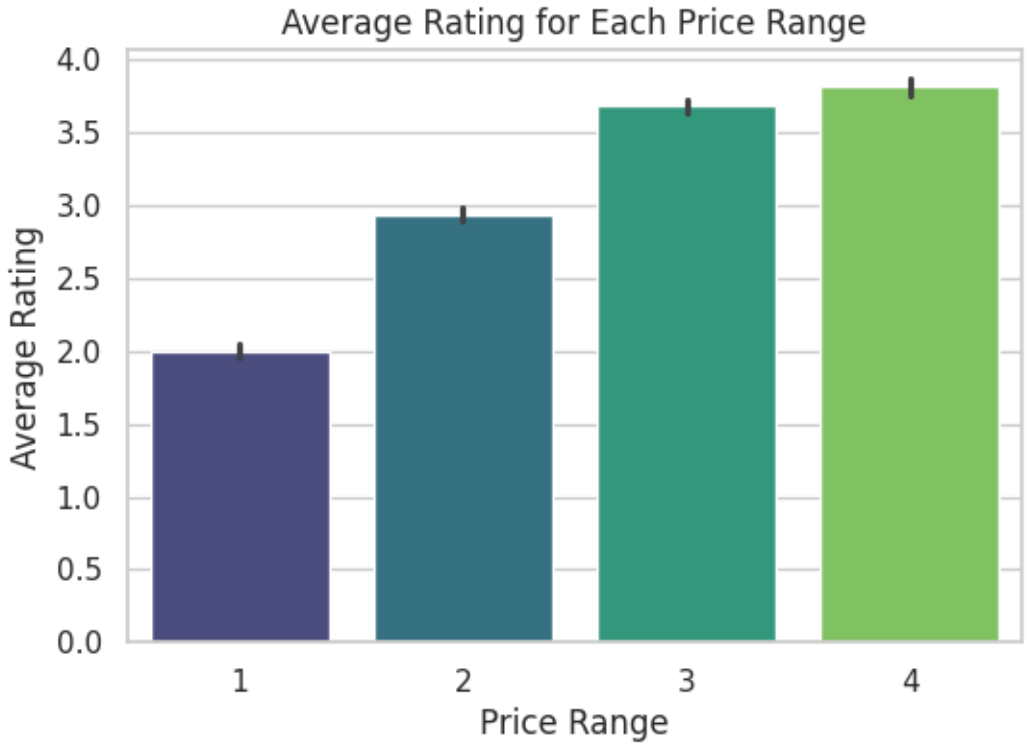
```
Average Rating for Each Price Range:
Price range
1    1.999887
2    2.941054
3    3.683381
4    3.817918
Name: Aggregate rating, dtype: float64
```

[9]:
```
# Average Rating for Each Price Range
plt.figure(figsize=(6, 4))
sns.barplot(x='Price range', y='Aggregate rating', data=df, palette="viridis")
plt.title('Average Rating for Each Price Range')
plt.xlabel('Price Range')
plt.ylabel('Average Rating')
plt.show()
```

```
<ipython-input-9-afcb92e3c940>:3: 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='Price range', y='Aggregate rating', data=df, palette="viridis")
```



Identify color with highest average rating:

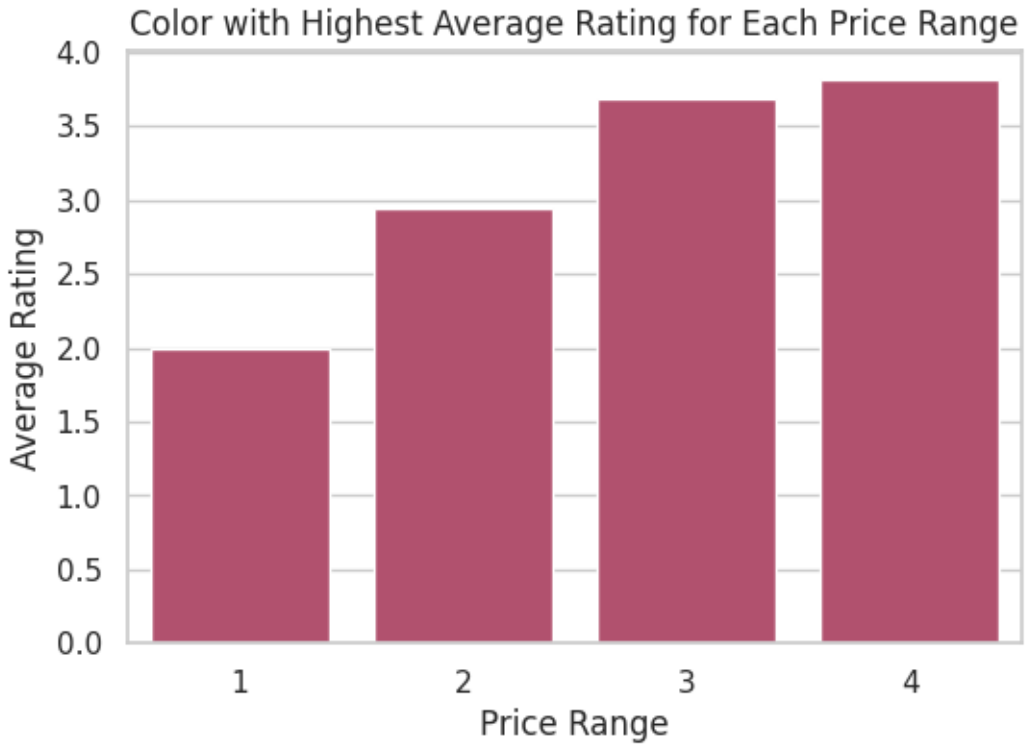
[10]:
```
# Identify color with highest average rating
highest_avg_rating_color = df.groupby('Price range')['Aggregate rating'].
  ↪idxmax()
color_representing_highest_rating = df.loc[highest_avg_rating_color]['Rating␣
  ↪color'].iloc[0]
print("\033[1mColor Representing Highest Average Rating:\033[0m",␣
  ↪color_representing_highest_rating)
```

Color Representing Highest Average Rating: Dark Green

[11]:
```
# Identify Color with Highest Average Rating
avg_rating_by_price_range = df.groupby('Price range')['Aggregate rating'].mean()
```

```
highest_avg_rating_color = df.groupby('Price range')['Aggregate rating'].
  ↪idxmax().apply(lambda x: df.loc[x]['Rating color'])

plt.figure(figsize=(6, 4))
sns.barplot(x=avg_rating_by_price_range.index, y=avg_rating_by_price_range.
  ↪values, hue=highest_avg_rating_color, palette="flare", legend=False)
plt.title('Color with Highest Average Rating for Each Price Range')
plt.xlabel('Price Range')
plt.ylabel('Average Rating')
plt.show()
```



Top Cuisines by Average Cost:

[12]:
```
top_cuisines_by_cost = df.groupby('Cuisines')['Average Cost for two'].mean().
  ↪sort_values(ascending=False).head(10)
print("\033[1mTop Cuisines by Average Cost:\033[0m")
print(top_cuisines_by_cost)
```

```
Top Cuisines by Average Cost:
Cuisines
Asian, Indonesian, Western    800000.0
French, Western               350000.0
Cafe, Western                 300000.0
```

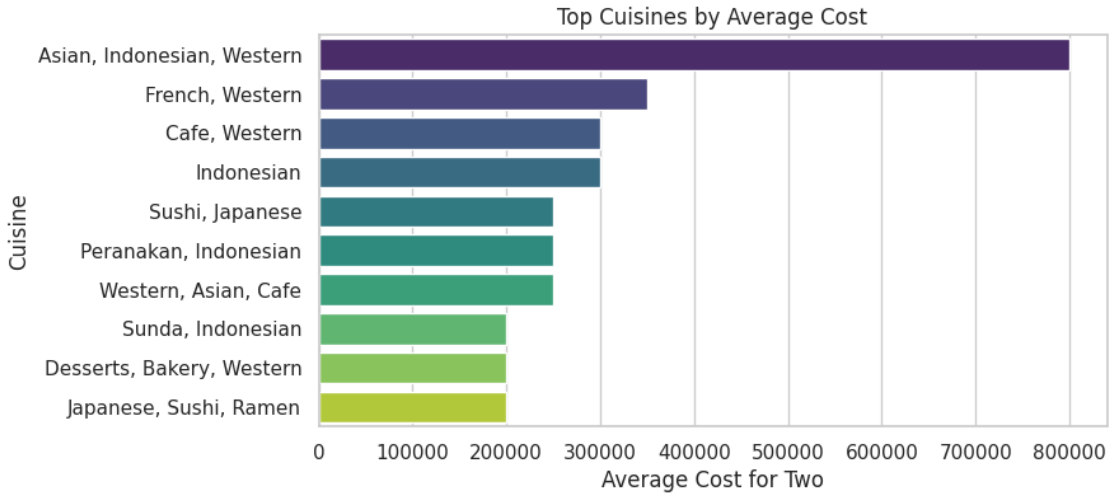
```
Indonesian                    300000.0
Sushi, Japanese               250060.0
Peranakan, Indonesian         250000.0
Western, Asian, Cafe          250000.0
Sunda, Indonesian             200000.0
Desserts, Bakery, Western     200000.0
Japanese, Sushi, Ramen        200000.0
Name: Average Cost for two, dtype: float64
```

[13]:
```
# Top Cuisines by Average Cost
top_cuisines_by_cost = df.groupby('Cuisines')['Average Cost for two'].mean().
 ↪sort_values(ascending=False).head(10)
plt.figure(figsize=(8, 4))
sns.barplot(x=top_cuisines_by_cost.values, y=top_cuisines_by_cost.index,␣
 ↪palette="viridis")
plt.title('Top Cuisines by Average Cost')
plt.xlabel('Average Cost for Two')
plt.ylabel('Cuisine')
plt.show()
```

```
<ipython-input-13-707431cccf41>:4: 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=top_cuisines_by_cost.values, y=top_cuisines_by_cost.index,
palette="viridis")
```



Country-wise Analysis:

[14]:
```
country_analysis = df.groupby('Country Code')['Aggregate rating'].describe()
print("\033[1mCountry-wise Analysis:\033[0m")
print(country_analysis)
```

Country-wise Analysis:

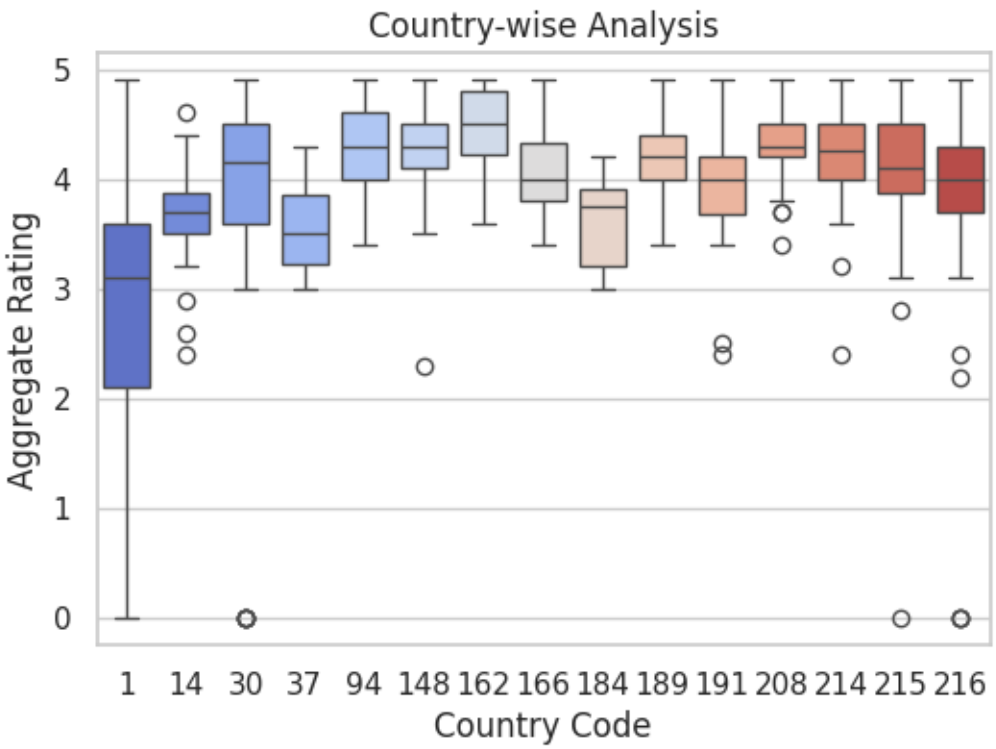
Country Code	count	mean	std	min	25%	50%	75%	max
1	8652.0	2.523324	1.510986	0.0	2.100	3.10	3.600	4.9
14	24.0	3.658333	0.523298	2.4	3.500	3.70	3.875	4.6
30	60.0	3.763333	1.253195	0.0	3.600	4.15	4.500	4.9
37	4.0	3.575000	0.561991	3.0	3.225	3.50	3.850	4.3
94	21.0	4.295238	0.428341	3.4	4.000	4.30	4.600	4.9
148	40.0	4.262500	0.433050	2.3	4.100	4.30	4.500	4.9
162	22.0	4.468182	0.345566	3.6	4.225	4.50	4.800	4.9
166	20.0	4.060000	0.418519	3.4	3.800	4.00	4.325	4.9
184	20.0	3.575000	0.390512	3.0	3.200	3.75	3.900	4.2
189	60.0	4.210000	0.331765	3.4	4.000	4.20	4.400	4.9
191	20.0	3.870000	0.590361	2.4	3.675	4.00	4.200	4.9
208	34.0	4.300000	0.347284	3.4	4.200	4.30	4.500	4.9
214	60.0	4.233333	0.421726	2.4	4.000	4.25	4.500	4.9
215	80.0	4.087500	0.633331	0.0	3.875	4.10	4.500	4.9
216	434.0	4.004378	0.527230	0.0	3.700	4.00	4.300	4.9

[15]:
```
# Country-wise Analysis
plt.figure(figsize=(6, 4))
sns.boxplot(x='Country Code', y='Aggregate rating', data=df, palette="coolwarm")
plt.title('Country-wise Analysis')
plt.xlabel('Country Code')
plt.ylabel('Aggregate Rating')
plt.show()
```

```
<ipython-input-15-83886cdd378d>:3: 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.boxplot(x='Country Code', y='Aggregate rating', data=df,
palette="coolwarm")
```

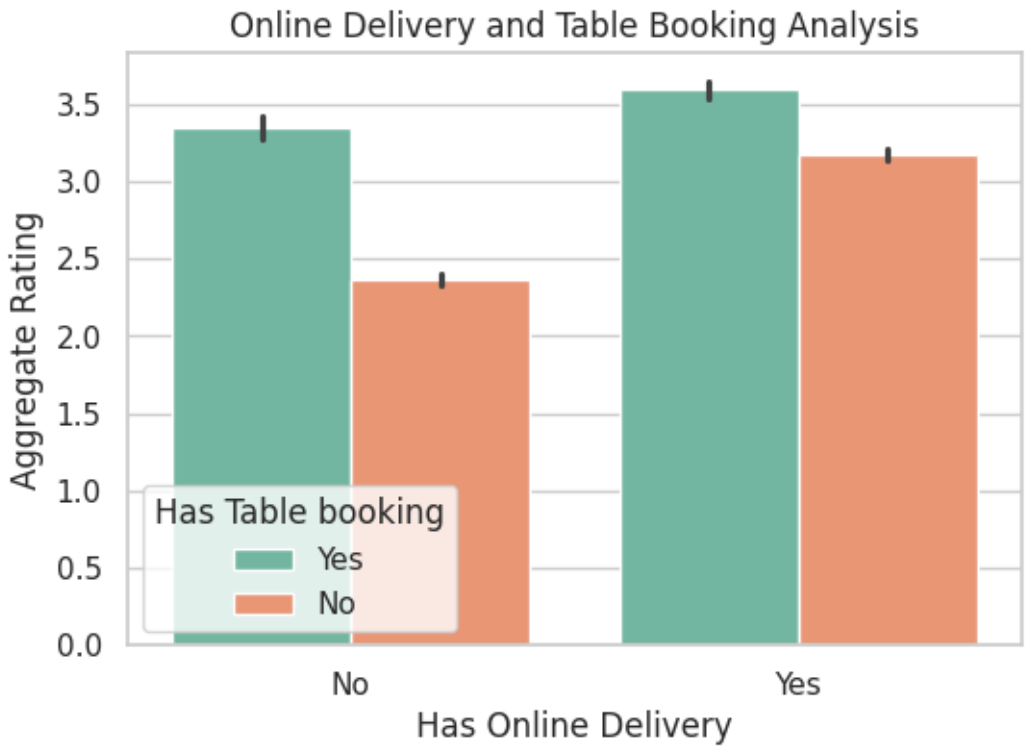


Online Delivery and Table Booking:

[16]:
```
delivery_booking_analysis = df.groupby(['Has Online delivery', 'Has Table␣
↪booking'])['Aggregate rating'].mean()
print("\033[1mOnline Delivery and Table Booking Analysis:\033[0m")
print(delivery_booking_analysis)
```

```
Online Delivery and Table Booking Analysis:
Has Online delivery  Has Table booking
No                   No                  2.365062
                     Yes                 3.349378
Yes                  No                  3.173958
                     Yes                 3.595862
Name: Aggregate rating, dtype: float64
```

[17]:
```
# Online Delivery and Table Booking Analysis
plt.figure(figsize=(6, 4))
sns.barplot(x='Has Online delivery', y='Aggregate rating', hue='Has Table␣
↪booking', data=df, palette="Set2")
plt.title('Online Delivery and Table Booking Analysis')
plt.xlabel('Has Online Delivery')
plt.ylabel('Aggregate Rating')
plt.show()
```



Locality Analysis:

[18]:
```
top_localities_by_rating = df.groupby('Locality')['Aggregate rating'].mean().
  ↪sort_values(ascending=False).head(10)
print("\033[1mTop Localities by Average Rating:\033[0m")
print(top_localities_by_rating)
```

Top Localities by Average Rating:

```
Locality
Pondok Aren                          4.9
Venetian Village, Al Maqtaa          4.9
Hotel Clarks Amer, Malviya Nagar     4.9
Bebek                                4.9
DIFC                                 4.9
Beak Street, Soho                    4.9
The Milk District                    4.9
Kenwood                              4.9
Paia                                 4.9
Taman Impian Jaya Ancol, Ancol       4.9
Name: Aggregate rating, dtype: float64
```

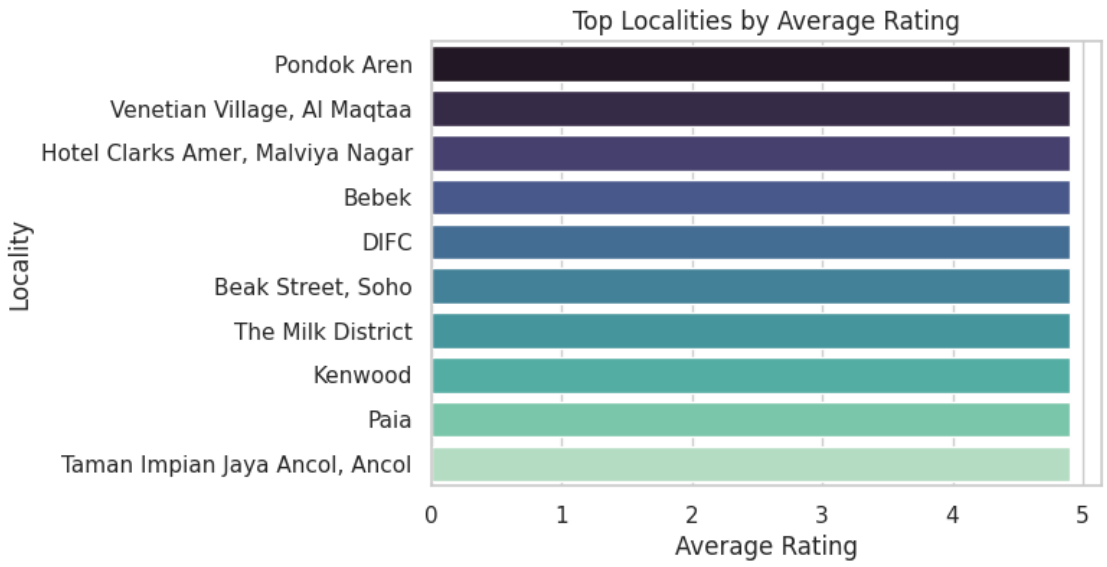
[19]:
```
# Top Localities by Average Rating
```

```
top_localities_by_rating = df.groupby('Locality')['Aggregate rating'].mean().
  ↪sort_values(ascending=False).head(10)
plt.figure(figsize=(6, 4))
sns.barplot(x=top_localities_by_rating.values, y=top_localities_by_rating.
  ↪index, palette="mako")
plt.title('Top Localities by Average Rating')
plt.xlabel('Average Rating')
plt.ylabel('Locality')
plt.show()
```

```
<ipython-input-19-b3378c6707ca>:4: 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=top_localities_by_rating.values,
y=top_localities_by_rating.index, palette="mako")
```



Correlation Analysis:

[20]:
```
correlation_matrix = df[['Average Cost for two', 'Price range', 'Aggregate␣
  ↪rating']].corr()
print("\033[1mCorrelation Matrix:\033[0m")
print(correlation_matrix)
```

```
Correlation Matrix:
                      Average Cost for two  Price range  Aggregate rating
Average Cost for two              1.000000     0.075083          0.051792
```

```
Price range                 0.075083      1.000000          0.437944
Aggregate rating            0.051792      0.437944          1.000000
```

[21]:
```
# Correlation Matrix
correlation_matrix = df[['Average Cost for two', 'Price range', 'Aggregate␣
 ↪rating']].corr()
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title('Correlation Matrix')
plt.show()
```