Restaurant Data Analysis🍽️ 📊 🔍

level 1 Task 1: Data Exploration and Preprocessing🔍 📊 🧹

1. Dataset Overview📂 🔍

Objective🎯 :

- The first step in any data analysis project is to understand the structure and contents of the dataset. We begin by loading the dataset and examining its shape (number of rows and columns) and the first few records.

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
file_path = ('D:/cognifyz/Dataset .csv')
df = pd.read_csv(file_path)
```

In [5]:
```
print("Number of rows and columns:", df.shape)
print("First few rows of the dataset:\n", df.head())
```

```
Number of rows and columns: (9551, 21)
First few rows of the dataset:
    Restaurant ID         Restaurant Name  Country Code              City  \
0         6317637        Le Petit Souffle           162       Makati City
1         6304287        Izakaya Kikufuji           162       Makati City
2         6300002  Heat - Edsa Shangri-La           162  Mandaluyong City
3         6318506                    Ooma           162  Mandaluyong City
4         6314302             Sambo Kojin           162  Mandaluyong City

                                        Address  \
0  Third Floor, Century City Mall, Kalayaan Avenu...
1  Little Tokyo, 2277 Chino Roces Avenue, Legaspi...
2  Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...
3  Third Floor, Mega Fashion Hall, SM Megamall, O...
4  Third Floor, Mega Atrium, SM Megamall, Ortigas...

                                  Locality  \
0   Century City Mall, Poblacion, Makati City
1  Little Tokyo, Legaspi Village, Makati City
2  Edsa Shangri-La, Ortigas, Mandaluyong City
3      SM Megamall, Ortigas, Mandaluyong City
4      SM Megamall, Ortigas, Mandaluyong City

                                Locality Verbose   Longitude   Latitude  \
0  Century City Mall, Poblacion, Makati City, Mak...  121.027535  14.565443
1  Little Tokyo, Legaspi Village, Makati City, Ma...  121.014101  14.553708
2  Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...  121.056831  14.581404
3  SM Megamall, Ortigas, Mandaluyong City, Mandal...  121.056475  14.585318
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]
```

Results 📊✅:

- Shape: The dataset consists of 9,551 rows and 21 columns.
- Sample Data: A glimpse at the first few rows reveals that the dataset contains information about restaurants, including their names, locations, cuisines, ratings, and more.

Insight 💡 :

- The dataset is quite comprehensive, covering various aspects of restaurant data, which will be useful for detailed analysis and modeling.

2. Missing Values Analysis🔍 ❓ 📉

Objective🎯 :

- Identify missing values in each column to understand the completeness of the data and handle any missing values appropriately.

In [6]:
```
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 Restaurant ID           0
Restaurant Name          0
Country Code             0
City                     0
Address                  0
Locality                 0
Locality Verbose         0
Longitude                0
Latitude                 0
Cuisines                 9
Average Cost for two     0
Currency                 0
Has Table booking        0
Has Online delivery      0
Is delivering now        0
Switch to order menu     0
Price range              0
Aggregate rating         0
Rating color             0
Rating text              0
Votes                    0
dtype: int64
```

Results 📊 ✅ :

- The only column with missing values is Cuisines, which has 9 missing entries.

3. Handling Missing Values 🛠️❓📈:

Action 🔧:

- Replace the missing values in the Cuisines column with 'Unknown', ensuring that these missing values do not disrupt further analysis.

In [7]:
```
df['Cuisines'] = df['Cuisines'].fillna('Unknown')
missing_values_after = df.isnull().sum()
print("Missing values after handling:\n", missing_values_after)
```

```
Missing values after handling:
 Restaurant ID           0
Restaurant Name          0
Country Code             0
City                     0
Address                  0
Locality                 0
Locality Verbose         0
Longitude                0
Latitude                 0
Cuisines                 0
Average Cost for two     0
Currency                 0
Has Table booking        0
Has Online delivery      0
Is delivering now        0
Switch to order menu     0
Price range              0
Aggregate rating         0
Rating color             0
Rating text              0
Votes                    0
dtype: int64
```

Conclusion 🔚✅:

After handling the missing values, the dataset has no missing entries, making it ready for further analysis.

3. Data Type Conversion 🔄📊🔢:

Objective 🎯:

- Check the data types of each column to ensure they are appropriate for analysis. If necessary, perform data type conversion.

In [8]:
```
print("Data types before conversion:\n", df.dtypes)
```

```
Data types before conversion:
 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
```

In [9]:
```
print("Data types after conversion:\n", df.dtypes)
```

```
Data types after conversion:
 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
```

Results 📊✅:

- The data types are generally appropriate for the columns, with integers and floats used for numerical data and objects (strings) used for categorical data.

Conclusion ⬅️ END ✅:

No data type conversion is needed. The dataset is well-structured with the correct data types for further analysis.

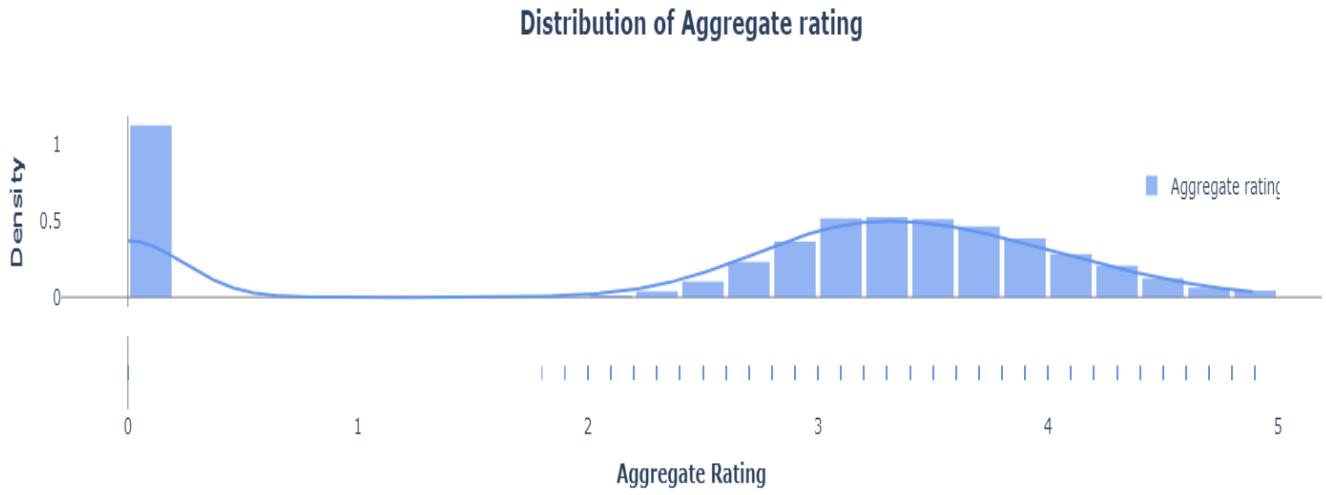
4. Target Variable Analysis ("Aggregate Rating") 🎯 📊 ⭐:

Objective 🎯:

- Analyze the distribution of the target variable (Aggregate rating), which will help understand its spread and identify any potential class imbalances

In [10]:
```
import plotly.graph_objects as go
import plotly.figure_factory as ff
target_variable = 'Aggregate rating'
hist_data = [df[target_variable]]
group_labels = [target_variable]
fig = ff.create_distplot(
    hist_data,
    group_labels,
    bin_size=0.2,
    colors=['cornflowerblue']
)
fig.update_layout(
    title={
        'text': f'Distribution of {target_variable}',
        'y': 0.9,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top',
        'font': dict(size=20, weight='bold')
    },
    xaxis_title={
        'text': 'Aggregate Rating',
        'font': dict(size=16, weight='bold')
    },
    yaxis_title={
        'text': 'Density',
        'font': dict(size=16, weight='bold')
    },
    xaxis=dict(
        tickfont=dict(size=14),
        showgrid=True,
        zeroline=True,
        zerolinecolor='gray',
        zerolinewidth=1
    ),
    yaxis=dict(
        tickfont=dict(size=14),
        showgrid=True,
        zeroline=True,
```

```
        zerolinecolor='gray',
        zerolinewidth=1
    ),
    bargap=0.1,
    plot_bgcolor='white',
    paper_bgcolor='white',
    hovermode='closest',
    legend=dict(
        x=0.85,
        y=0.85,
        font=dict(size=14)
    )
)
fig.show()
```



In [11]:
```
class_counts = df[target_variable].value_counts()
print("Class distribution:\n", class_counts)
```

```
Class distribution:
 Aggregate rating
0.0    2148
3.2     522
3.1     519
3.4     498
3.3     483
3.5     480
3.0     468
3.6     458
3.7     427
3.8     400
2.9     381
3.9     335
2.8     315
4.1     274
4.0     266
2.7     250
4.2     221
2.6     191
4.3     174
4.4     144
2.5     110
4.5      95
2.4      87
4.6      78
4.9      61
2.3      47
4.7      42
2.2      27
4.8      25
2.1      15
2.0       7
1.9       2
1.8       1
Name: count, dtype: int64
```

Results 📊 ✅:

- Distribution: The distribution of Aggregate rating shows a heavy concentration of ratings around 0.0 and in the range of 3.0 to 4.0.
- Class Imbalance: The most frequent ratings are 0.0, followed by ratings between 3.0 and 4.0, indicating potential class imbalance issues that might need to be addressed during modeling.

Insight 💡 :

- The presence of a significant number of restaurants with an aggregate rating of 0.0 could indicate either unrated or poorly rated restaurants. This imbalance may require special handling during model training, such as resampling techniques or adjusting the class weights.

Conclusion 🔚✅:

The data exploration and preprocessing steps have successfully cleaned and prepared the dataset for further analysis. The target variable shows some imbalance, which needs to be considered in future modeling steps.

Task 2: Descriptive Analysis 📊📝

1. Statistical Summary of Numerical Columns 📊📈🔢

Objective 🎯:

- Calculate basic statistical measures like mean, median, and standard deviation for the numerical columns to get an overview of the data distribution.

In [12]:
```
numerical_stats = df.describe()
print("Basic statistical measures for numerical columns:\n", numerical_stats)
```

```
Basic statistical measures for numerical columns:
       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
```

In [13]:
```
numerical_df = df.select_dtypes(include='number')
median_values = numerical_df.median()
print("Median values for numerical columns:\n", median_values)
```

```
Median values for numerical columns:
 Restaurant ID         6.004089e+06
Country Code           1.000000e+00
Longitude              7.719196e+01
Latitude               2.857047e+01
Average Cost for two   4.000000e+02
Price range            2.000000e+00
Aggregate rating       3.200000e+00
Votes                  3.100000e+01
dtype: float64
```

Results 📊✅:

- Key Statistics 🔑📊:

- Mean: The average values provide insight into central tendencies, such as an average Aggregate rating of 2.67.
- Standard Deviation: High standard deviation values for Average Cost for two indicate a wide range of restaurant pricing.
- Median: Median values for most numerical columns are close to their mean, suggesting a relatively normal distribution, except for Votes, which has a high variance.

Insight 💡:

- The distribution of the Aggregate rating and Votes indicates a skew in the data, with some restaurants having significantly higher ratings and votes than others.

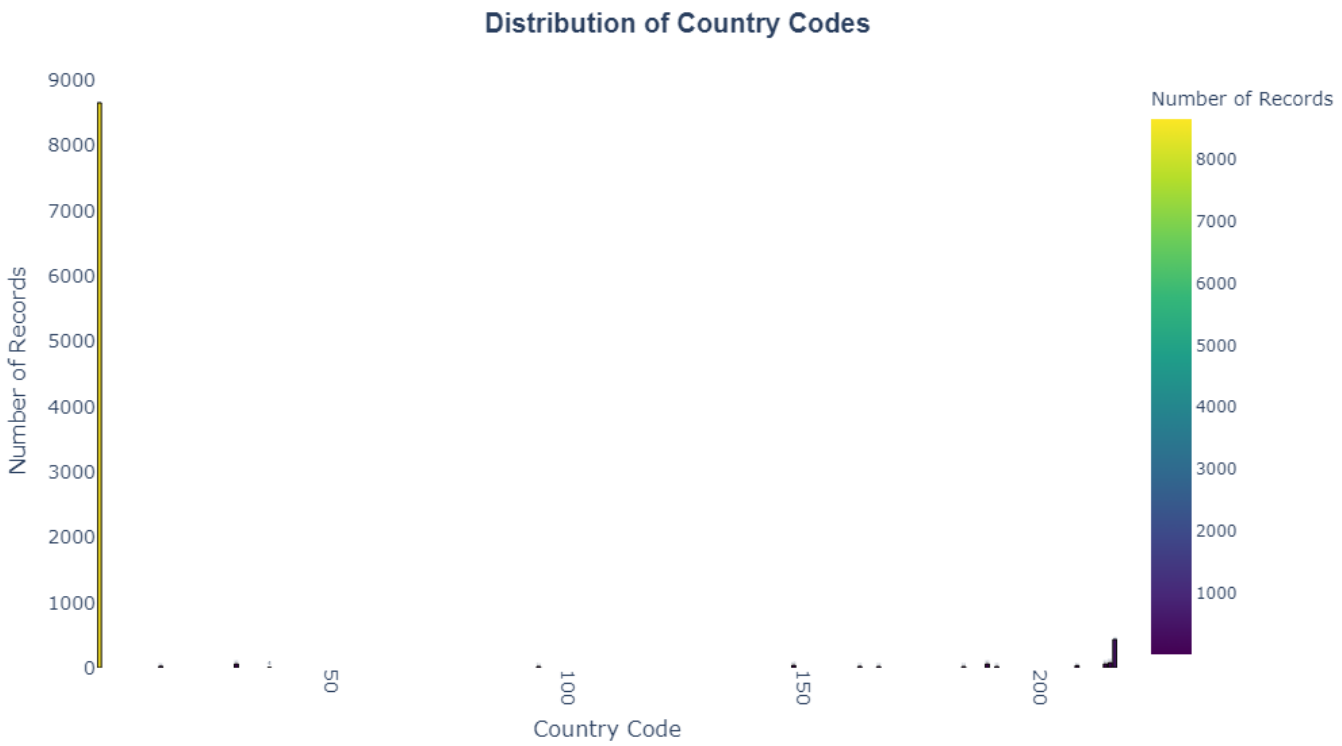
2. Distribution of Categorical Variables 📊📉🔢

Objective 🎯:

- Explore the distribution of key categorical variables such as Country Code, City, and Cuisines to understand the spread of data across different categories.

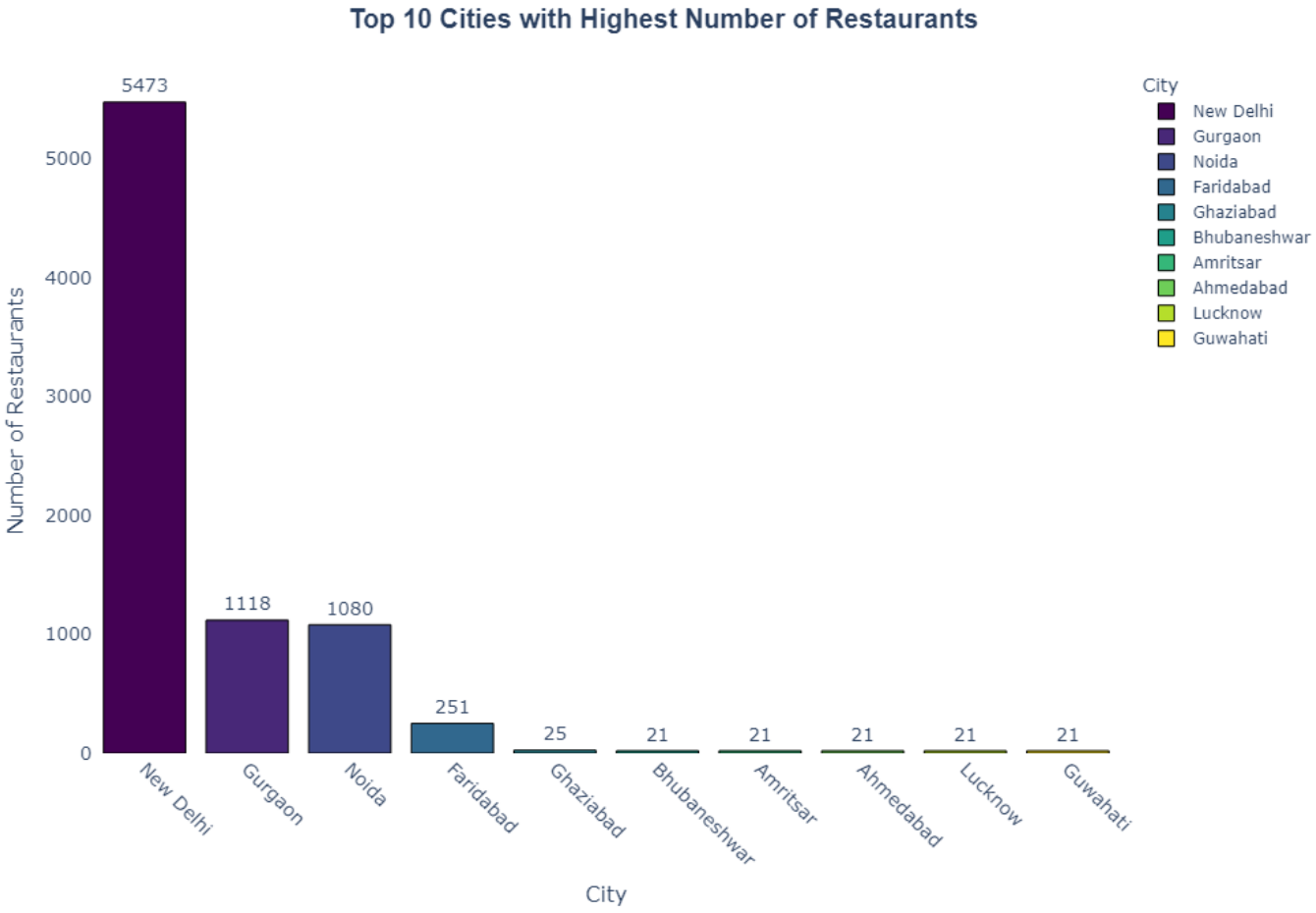
In [61]:
```
import pandas as pd
import plotly.express as px
file_path = 'D:/cognifyz/Dataset .csv'
df = pd.read_csv(file_path)
country_code_distribution = df['Country Code'].value_counts()
country_code_df = pd.DataFrame({
    'Country Code': country_code_distribution.index,
    'Count': country_code_distribution.values
})
fig = px.bar(
    country_code_df,
    x='Country Code',
    y='Count',
```

```
    text='Count',  # Add text annotations
    title='Distribution of Country Codes',
    color='Count',  # Color bars by count
    color_continuous_scale=px.colors.sequential.Viridis,
    labels={'Count': 'Number of Records'},
fig.update_layout(
    title={'text': 'Distribution of Country Codes', 'x': 0.5, 'xanchor': 'center',
    xaxis_title='Country Code',
    yaxis_title='Number of Records',
    xaxis=dict(
        tickangle=90,
        tickfont=dict(size=14),
        title_font=dict(size=16),
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        tickfont=dict(size=14),
        title_font=dict(size=16),
        showgrid=False,
        zeroline=False ]
    ),
    paper_bgcolor='white',
    plot_bgcolor='white',
    width=1000,
    height=600,
    margin=dict(l=50, r=20, t=60, b=100)
)
fig.update_traces(
    texttemplate='%{text}',
    textposition='outside',
    textfont_size=14,
    marker=dict(line=dict(color='black', width=1))
)
fig.show()
```


Distribution of Country Codes
Number of Records
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
50
100
150
200
Country Code
Number of Records
8000
7000
6000
5000
4000
3000
2000
1000

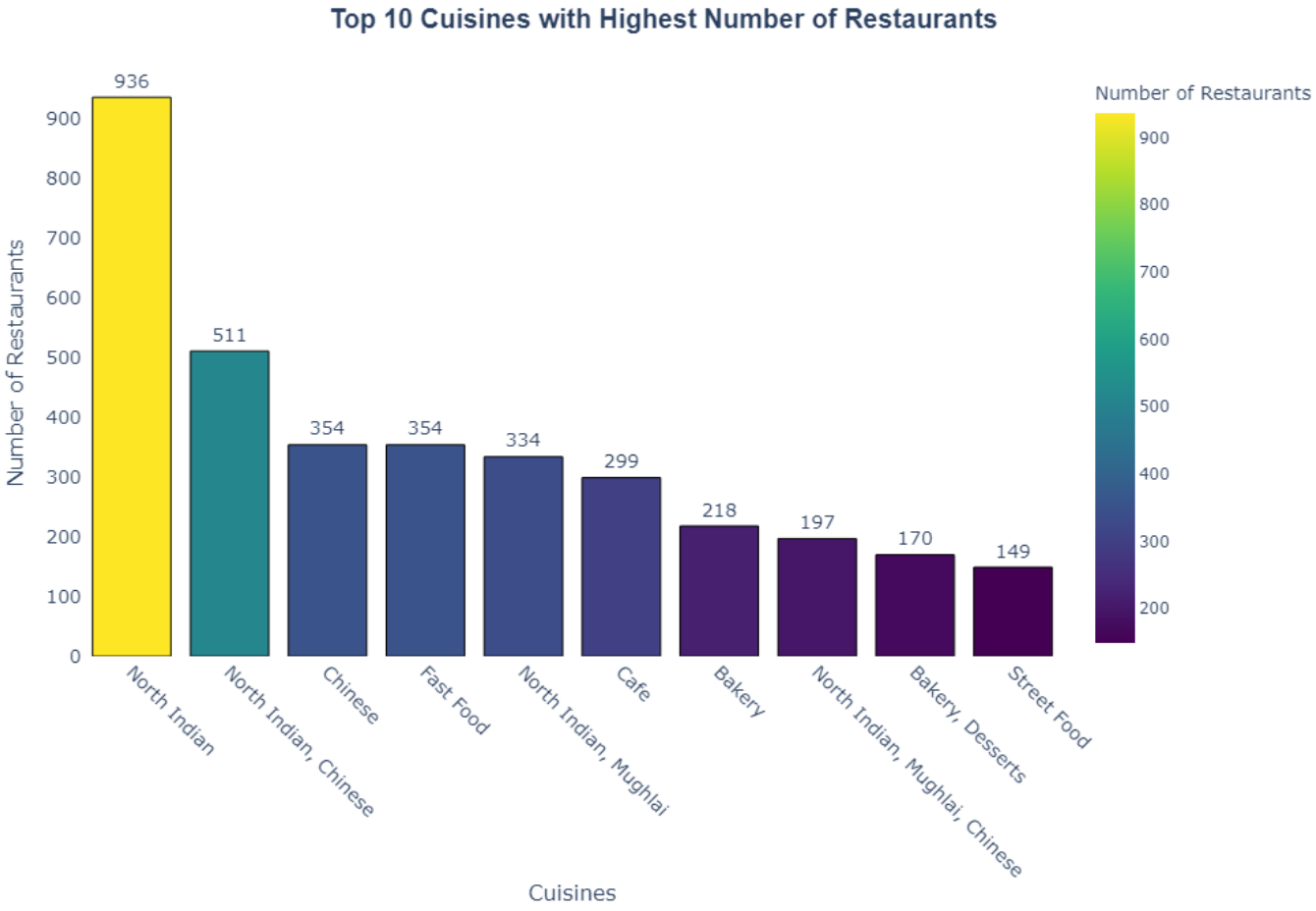
In [56]:
```
import pandas as pd
import plotly.express as px
file_path = 'D:/cognifyz/Dataset .csv'
df = pd.read_csv(file_path)
city_distribution = df['City'].value_counts()
top_cities = city_distribution.head(10)
top_cities_df = pd.DataFrame({
    'City': top_cities.index,
    'Count': top_cities.values
})
fig = px.bar(
    top_cities_df,
    x='City',
    y='Count',
    text='Count',
    title='Top 10 Cities with Highest Number of Restaurants',
    color='City',
    color_discrete_sequence=px.colors.sequential.Viridis,
```

```
    labels={'Count': 'Number of Restaurants'},
)
fig.update_layout(
    title={'text': 'Top 10 Cities with Highest Number of Restaurants', 'x': 0.5, 'x
    xaxis_title='City',
    yaxis_title='Number of Restaurants',
    xaxis=dict(
        tickangle=45,
        tickfont=dict(size=14),
        title_font=dict(size=16),
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        tickfont=dict(size=14),
        title_font=dict(size=16),
        showgrid=False,
        zeroline=False
    ),
    paper_bgcolor='white',
    plot_bgcolor='white',
    width=1000,
    height=700,
    margin=dict(l=50, r=20, t=60, b=100)
)
fig.update_traces(
    texttemplate='%{text}',
    textposition='outside',
    textfont_size=14,
    marker=dict(line=dict(color='black', width=1))
)
fig.show()
```


Top 10 Cities with Highest Number of Restaurants
Number of Restaurants
City
5473
1118
1080
251
25
21
21
21
21
21
0
1000
2000
3000
4000
5000
New Delhi
Gurgaon
Noida
Faridabad
Ghaziabad
Bhubaneshwar
Amritsar
Ahmedabad
Lucknow
Guwahati

In [58]:
```
cuisines_distribution = df['Cuisines'].value_counts()
top_cuisines = cuisines_distribution.head(10)
top_cuisines_df = pd.DataFrame({
    'Cuisines': top_cuisines.index,
    'Count': top_cuisines.values
})
fig = px.bar(
    top_cuisines_df,
    x='Cuisines',
    y='Count',
    text='Count',
    title='Top 10 Cuisines with Highest Number of Restaurants',
```

```
    color='Count',
    color_continuous_scale=px.colors.sequential.Viridis,
    labels={'Count': 'Number of Restaurants'},
)
fig.update_layout(
    title={'text': 'Top 10 Cuisines with Highest Number of Restaurants', 'x': 0.5,
    xaxis_title='Cuisines',
    yaxis_title='Number of Restaurants',
    xaxis=dict(
        tickangle=45,
        tickfont=dict(size=14),
        title_font=dict(size=16),
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        tickfont=dict(size=14),
        title_font=dict(size=16),
        showgrid=False,
        zeroline=False
    ),
    paper_bgcolor='white',
    plot_bgcolor='white',
    width=1000,
    height=700,
    margin=dict(l=50, r=20, t=60, b=100)
)
fig.update_traces(
    texttemplate='%{text}',
    textposition='outside',
    textfont_size=14,
    marker=dict(line=dict(color='black', width=1))
)
fig.show()
```


Top 10 Cuisines with Highest Number of Restaurants
Number of Restaurants
Cuisines
936
511
354
354
334
299
218
197
170
149
North Indian
North Indian, Chinese
Chinese
Fast Food
North Indian, Mughlai
Cafe
Bakery
North Indian, Mughlai, Chinese
Bakery, Desserts
Street Food

Results 📈 ✅:

- Country Code: The dataset has a dominant presence of one country code, with the vast majority of entries linked to a single country.
- Top Cities: The top cities with the highest number of restaurants are heavily populated and urban areas, reflecting their restaurant density.
- Top Cuisines: Popular cuisines like North Indian, Chinese, and Italian are among the top in the dataset, reflecting common preferences.

Insight 💡 :

- The distribution of categorical variables highlights the concentration of restaurant data in certain cities and cuisines, suggesting these are key areas for focused analysis.

Task 3: Geospatial Analysis 🌍 📍 🔍

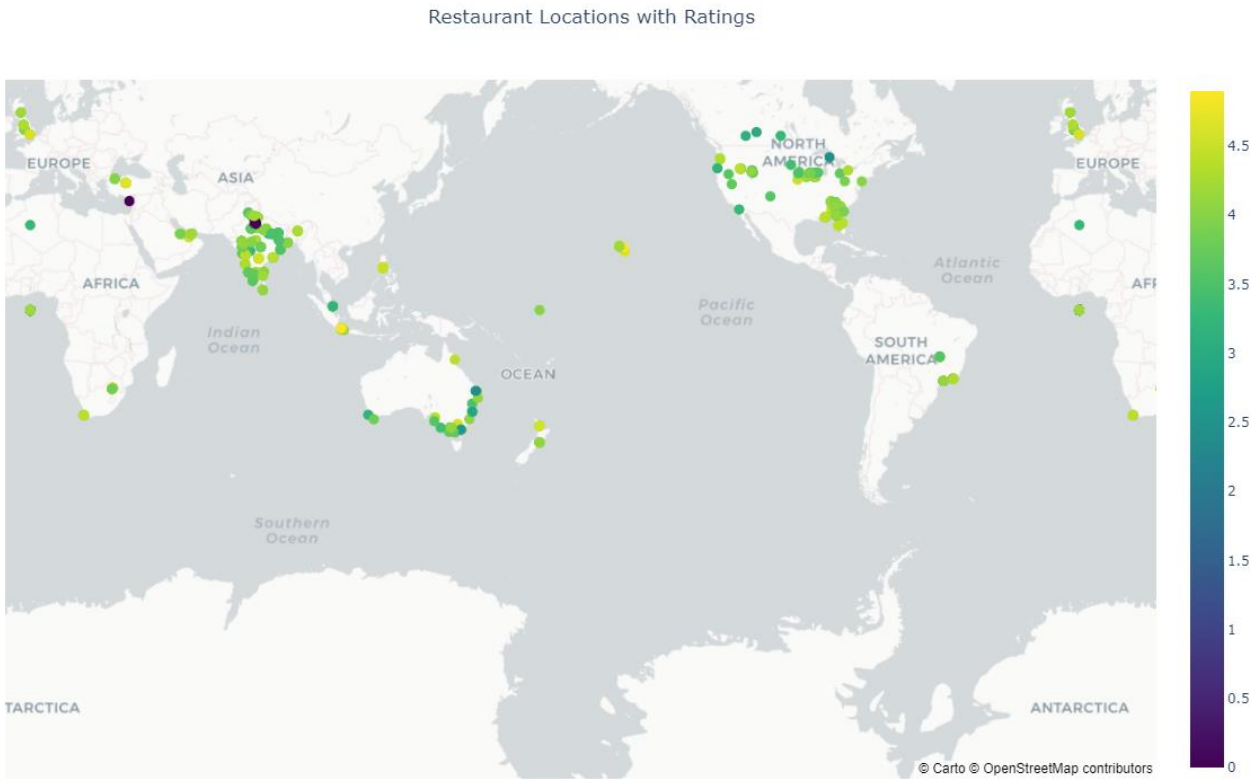
1. Visualizing the Distribution of Aggregate Ratings 📊 🌐 ⭐

Objective 🎯 :

- To visually understand the distribution of restaurant ratings and identify any trends or anomalies.

In [31]:
```
heatmap = go.Scattermapbox(
    lat=df['Latitude'],
    lon=df['Longitude'],
    mode='markers',
    marker=dict(
        size=5,
        color='rgba(255, 0, 0, 0.5)',  # Heatmap color
        opacity=0.5
    ),
    text=df.apply(lambda row: f"Restaurant: {row['Restaurant Name']}<br>Address: {r
    hoverinfo='text'
)
markers = go.Scattermapbox(
    lat=df['Latitude'],
    lon=df['Longitude'],
    mode='markers',
    marker=dict(
        size=10,
        color=df['Aggregate rating'],
        colorscale='Viridis',
        showscale=True
    ),
    text=df.apply(lambda row: f"Restaurant: {row['Restaurant Name']}<br>Address: {r
    hoverinfo='text'
)
fig = go.Figure(data=[heatmap, markers])
fig.update_layout(
    title={'text': 'Restaurant Locations with Ratings', 'x': 0.5},
    mapbox=dict(
        style='carto-positron',
        center=dict(lat=df['Latitude'].mean(), lon=df['Longitude'].mean()),
        zoom=10
    ),
    coloraxis_colorbar=dict(
```

```
        title='Rating',
        tickvals=[df['Aggregate rating'].min(), df['Aggregate rating'].max()],
        ticktext=[f"{df['Aggregate rating'].min()}", f"{df['Aggregate rating'].max(
    ),
    showlegend=False,
    width=1200,
    height=800
)
fig.show()
```



Results 📈 ✅:

- The histogram shows that most restaurants have ratings clustered around 0.0 and between 3.0 and 4.0. There is a significant drop in the number of restaurants with ratings above 4.5.

Insights 💡:

- This distribution suggests that while many restaurants receive average ratings, very few are rated as exceptional (above 4.5). The high number of 0.0 ratings could indicate unrated restaurants or those with very low customer satisfaction.

Conclusion 🔚 ✅:

The distribution indicates a potential focus area for improving customer experience, as many restaurants are not reaching the higher rating tiers. Additionally, the presence of many 0.0 ratings suggests that some restaurants may need strategies to encourage more customer feedback.

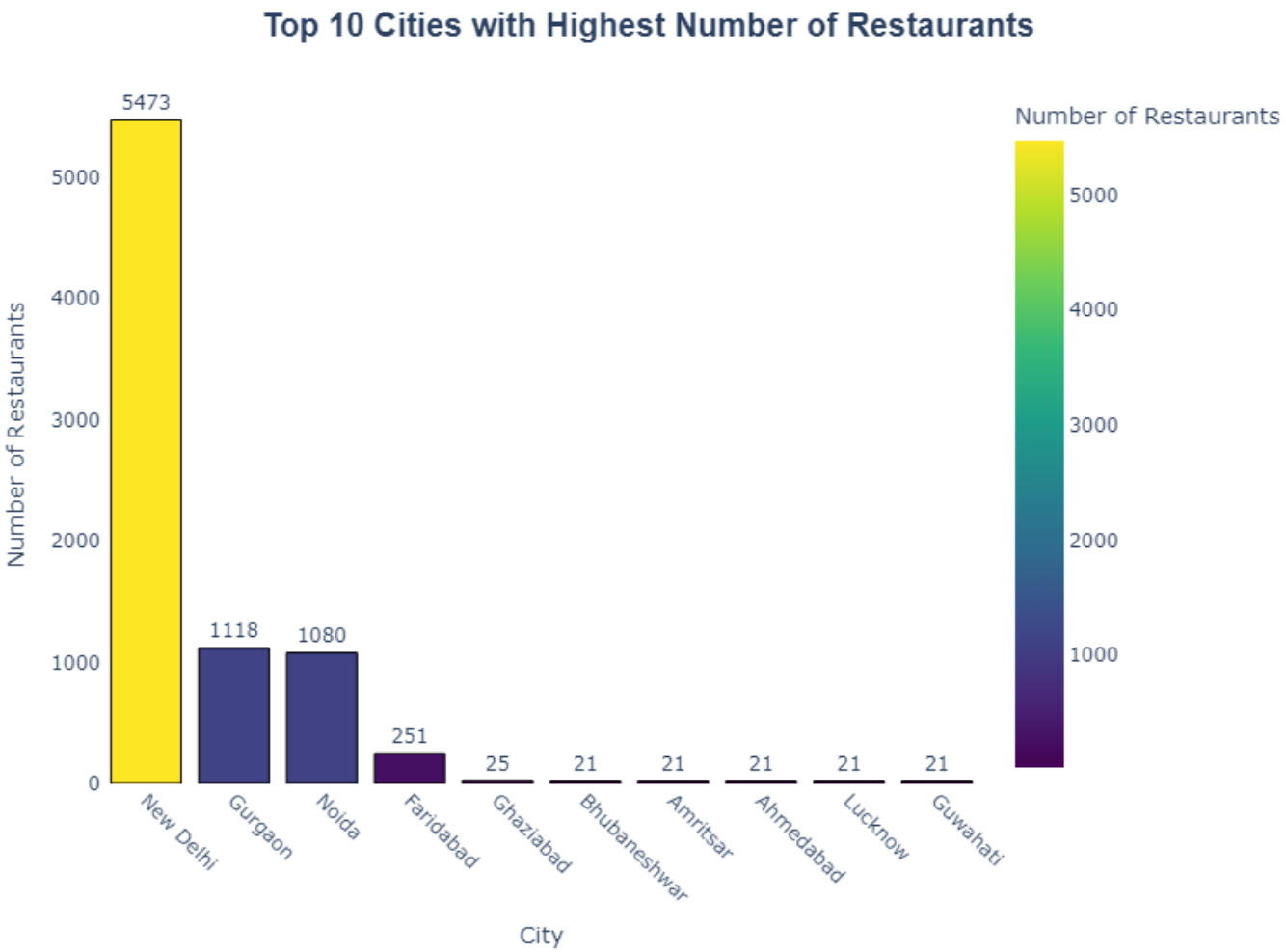
2. Top 10 Cities by Number of Restaurants 🏙️ 🔟 🍽️

Objective 🎯:

- Identify the cities with the most restaurants in the dataset to understand geographical distribution.

In [62]:
```
top_cities = df['City'].value_counts().head(10)
top_cities_df = pd.DataFrame({
    'City': top_cities.index,
    'Count': top_cities.values
})
fig = px.bar(
    top_cities_df,
    x='City',
    y='Count',
    text='Count',
    title='Top 10 Cities with Highest Number of Restaurants',
    color='Count',
    color_continuous_scale=px.colors.sequential.Viridis,
    labels={'Count': 'Number of Restaurants'},
)
fig.update_layout(
    title={'text': 'Top 10 Cities with Highest Number of Restaurants', 'x': 0.5, 'x
    xaxis_title='City',
    yaxis_title='Number of Restaurants',
    xaxis=dict(
        tickangle=45,
        tickfont=dict(size=12),
        title_font=dict(size=14),
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        tickfont=dict(size=12),
```

```
        title_font=dict(size=14),
        showgrid=False,
        zeroline=False
    ),
    paper_bgcolor='white',
    plot_bgcolor='white',  \
    width=800,
    height=600,
    margin=dict(l=50, r=20, t=60, b=80)
)
fig.update_traces(
    texttemplate='%{text}',
    textposition='outside',
    textfont_size=12,
    marker=dict(line=dict(color='black', width=1))
)
fig.show()
```



Results 📈 ✅ :

- New Delhi, Gurgaon, and Noida are the top three cities with the most restaurants, with New Delhi leading by a significant margin.

Insights 💡:

- These cities represent major urban centers with a high concentration of dining establishments, likely due to higher population density and disposable income.

Conclusion 🔚✅:

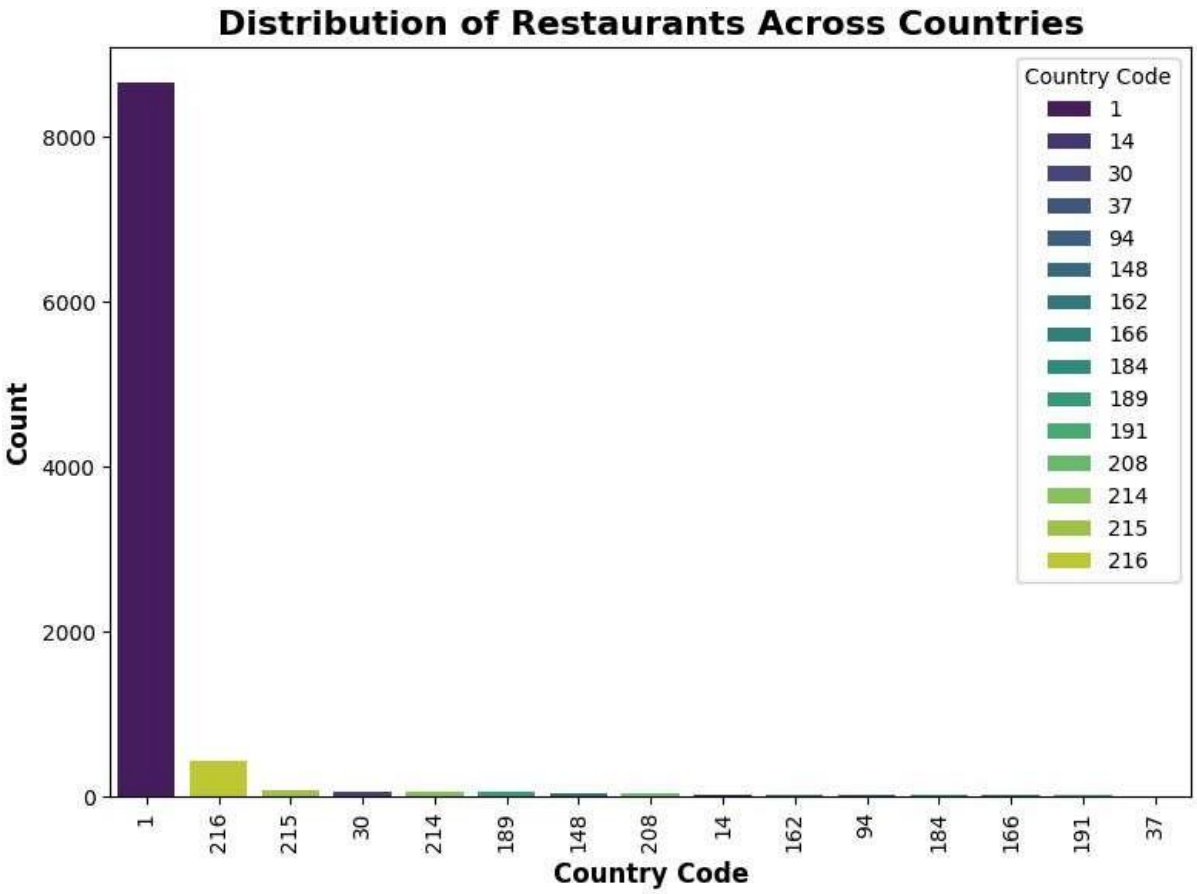
Marketing and business strategies should consider these cities as primary targets due to their high restaurant density. Understanding consumer preferences in these cities could yield valuable insights for expanding restaurant chains.

3. Country-wise Distribution of Restaurants🌍📊🍽️

Objective🎯:

- Understand the distribution of restaurants across different countries in the dataset.

In [64]:
```
palette = sns.color_palette('viridis', len(df['Country Code'].value_counts()))
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Country Code', order=df['Country Code'].value_counts().in
plt.title('Distribution of Restaurants Across Countries', fontsize=16, weight='bold
plt.xlabel('Country Code', fontsize=12, weight='bold')
plt.ylabel('Count', fontsize=12, weight='bold')
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()
```



Results 📈✅:

- The chart reveals a heavy concentration of restaurants in a single country, which likely dominates the dataset. Other countries have much smaller shares.

Insights 💡:

- The dataset is skewed towards one country, indicating that the analysis may primarily reflect the restaurant industry in that specific country rather than being globally representative.

Conclusion 🔚✅:

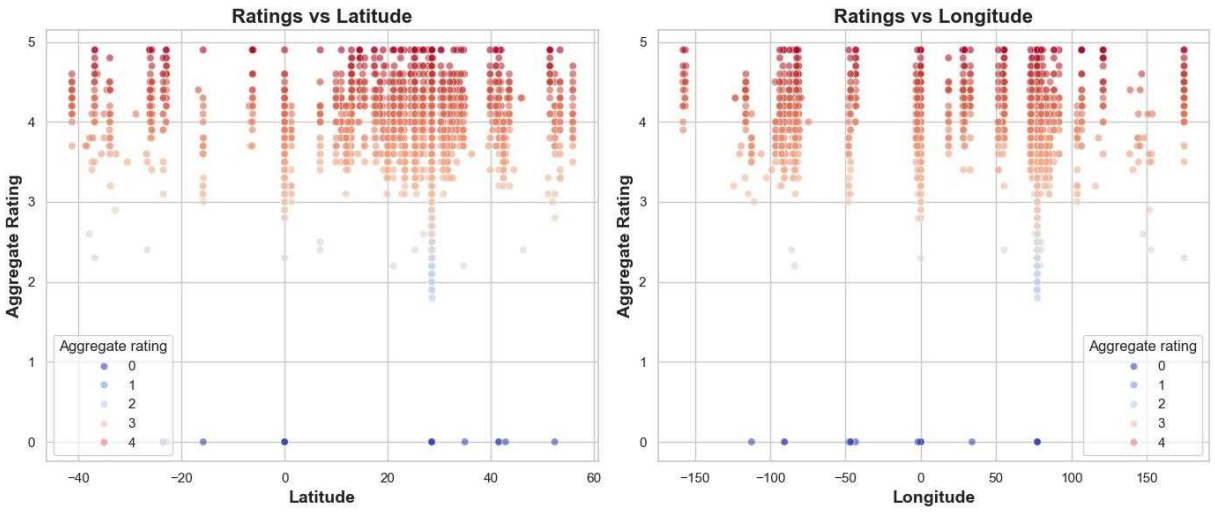
If the goal is to understand global trends, additional data from more countries would be necessary. For now, the focus should be on analyzing the market dynamics in the dominant country represented in the dataset.

4. Ratings vs. Latitude and Longitude 🌎

Objective 🎯:

- Examine how restaurant ratings are distributed geographically across latitude and longitude.

In [41]:
```
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.scatterplot(data=df, x='Latitude', y='Aggregate rating', hue='Aggregate rating'
plt.title('Ratings vs Latitude', fontsize=16, weight='bold')
plt.xlabel('Latitude', fontsize=14, weight='bold')
plt.ylabel('Aggregate Rating', fontsize=14, weight='bold')
plt.subplot(1, 2, 2)
sns.scatterplot(data=df, x='Longitude', y='Aggregate rating', hue='Aggregate rating
plt.title('Ratings vs Longitude', fontsize=16, weight='bold')
plt.xlabel('Longitude', fontsize=14, weight='bold')
plt.ylabel('Aggregate Rating', fontsize=14, weight='bold')
plt.tight_layout()
plt.show()
```



Insights 💡 :

- The scatter plots reveal that ratings are distributed across different latitudes and longitudes, with varying densities. There are some geographic areas where restaurants tend to have higher or lower ratings.

Conclusion 🔚 ✅ :

Geographic distribution of ratings can provide insights into regional trends in restaurant quality. Areas with lower ratings might indicate potential market opportunities for improving restaurant quality or addressing local consumer preferences.

5. Correlation Matrix 🎳

Objective 🎯 :

- Examine the relationships between latitude, longitude, and aggregate ratings.

In [42]:
```
correlation_matrix = df[['Latitude', 'Longitude', 'Aggregate rating']].corr()
print("Correlation matrix:\n", correlation_matrix)
```

```
Correlation matrix:
                  Latitude  Longitude  Aggregate rating
Latitude          1.000000   0.043207          0.000516
Longitude         0.043207   1.000000         -0.116818
Aggregate rating  0.000516  -0.116818          1.000000
```

Insights 💡 :

- The correlation matrix helps in understanding how latitude and longitude relate to ratings. For example, if latitude and longitude show strong correlations with ratings, this might indicate that location plays a significant role in restaurant ratings.

Conclusion 🔚 ✅ :

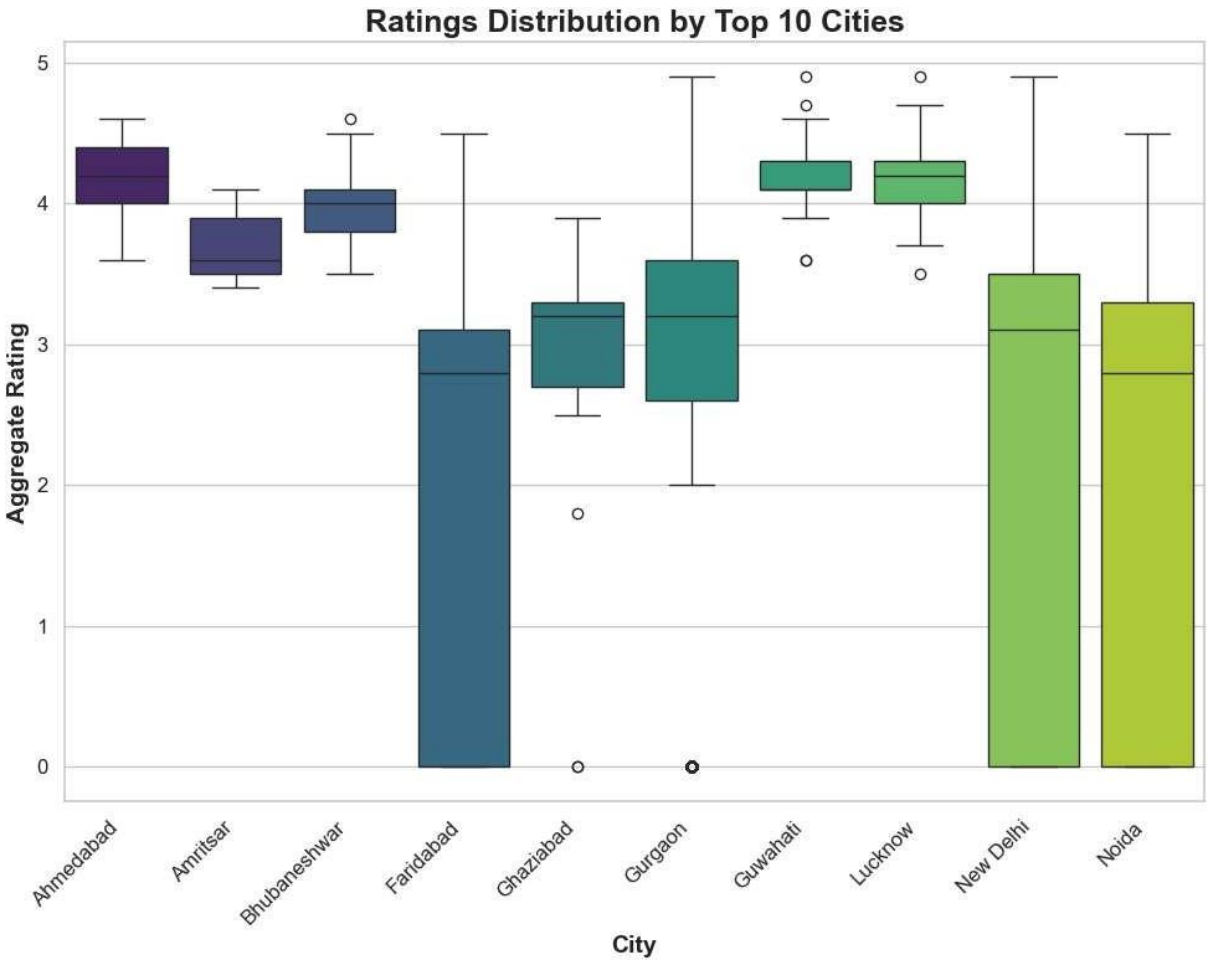
Correlation analysis provides valuable insights into the relationships between geographic coordinates and ratings, which can be useful for targeted marketing or location-based improvements.

6. Ratings Distribution by Top 10 Cities 🗺️

Objective 🎯 :

- Compare the distribution of ratings across the top 10 cities with the highest number of restaurants.

In [43]:
```
top_cities = df['City'].value_counts().head(10).index
filtered_df = df[df['City'].isin(top_cities)]
palette = sns.color_palette('viridis', len(top_cities))
plt.figure(figsize=(10, 8))
sns.boxplot(data=filtered_df, x='City', y='Aggregate rating', palette=palette, hue=
plt.title('Ratings Distribution by Top 10 Cities', fontsize=18, weight='bold')
plt.xlabel('City', fontsize=14, weight='bold')
plt.ylabel('Aggregate Rating', fontsize=14, weight='bold')
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



Insights 💡 :

- The boxplot displays the spread and central tendency of ratings in the top 10 cities. Some cities have a wider range of ratings compared to others, indicating varying levels of restaurant quality.

Conclusion ⬅️ END ✅ :

Understanding rating distributions by city helps identify which cities have more consistent restaurant quality and which have greater variability. This can aid in strategic decisions regarding market entry or quality control.

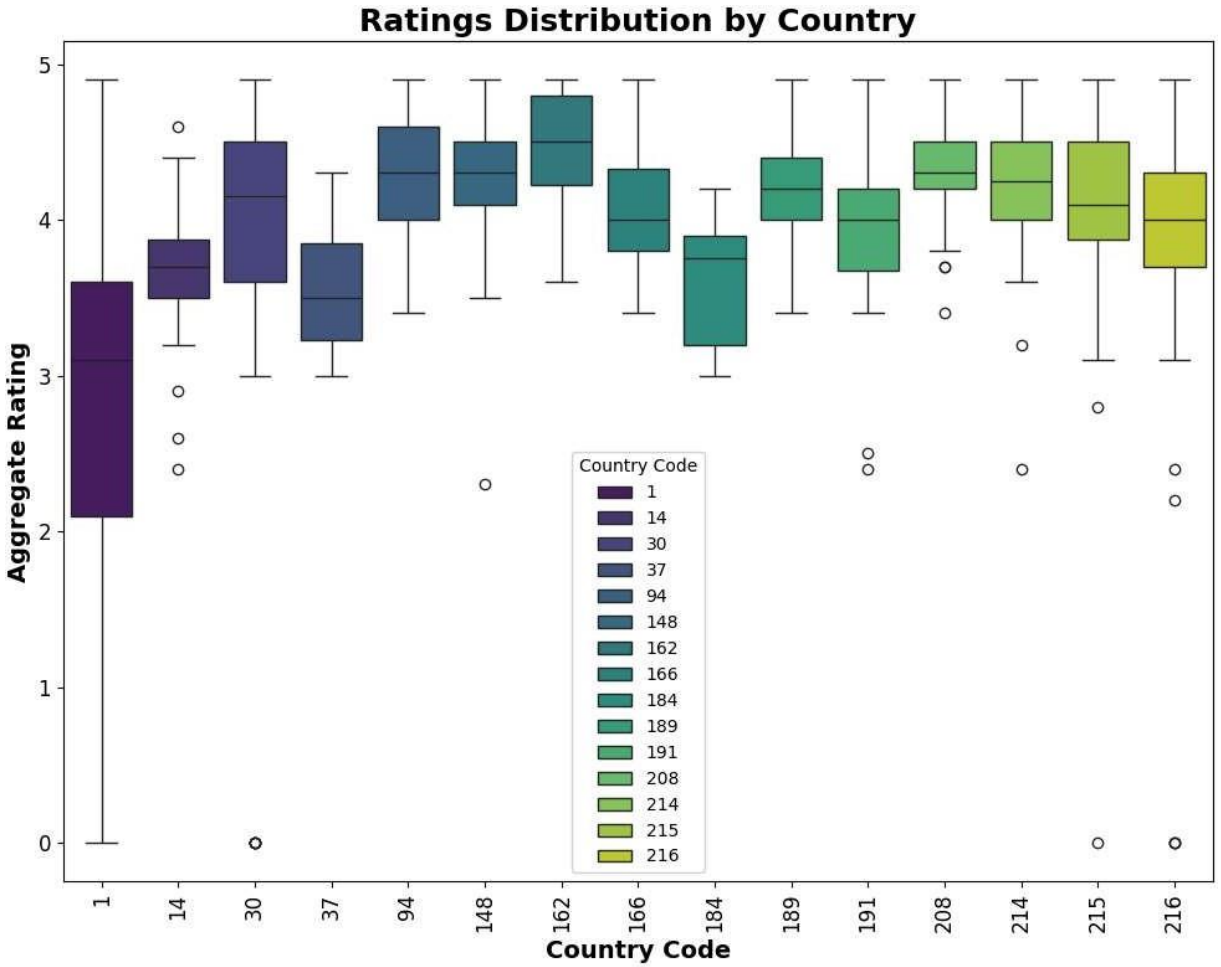
7. Ratings Distribution by Country 🌎

Objective 🎯 :

- Analyze how ratings vary across different countries.

In [40]:
```
num_countries = df['Country Code'].nunique()
palette = sns.color_palette('viridis', n_colors=num_countries)
```

```
plt.figure(figsize=(10, 8))
sns.boxplot(data=df, x='Country Code', y='Aggregate rating', palette=palette, hue='
plt.title('Ratings Distribution by Country', fontsize=18, weight='bold')
plt.xlabel('Country Code', fontsize=14, weight='bold')
plt.ylabel('Aggregate Rating', fontsize=14, weight='bold')
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



Insights 💡 :

- The boxplot illustrates the variability in ratings across different countries. Some countries might show higher ratings on average, while others may exhibit a broader range of ratings

Conclusion 🔚 ✅:

Analyzing ratings by country provides insights into international market differences in restaurant quality. It helps in understanding regional preferences and quality standards

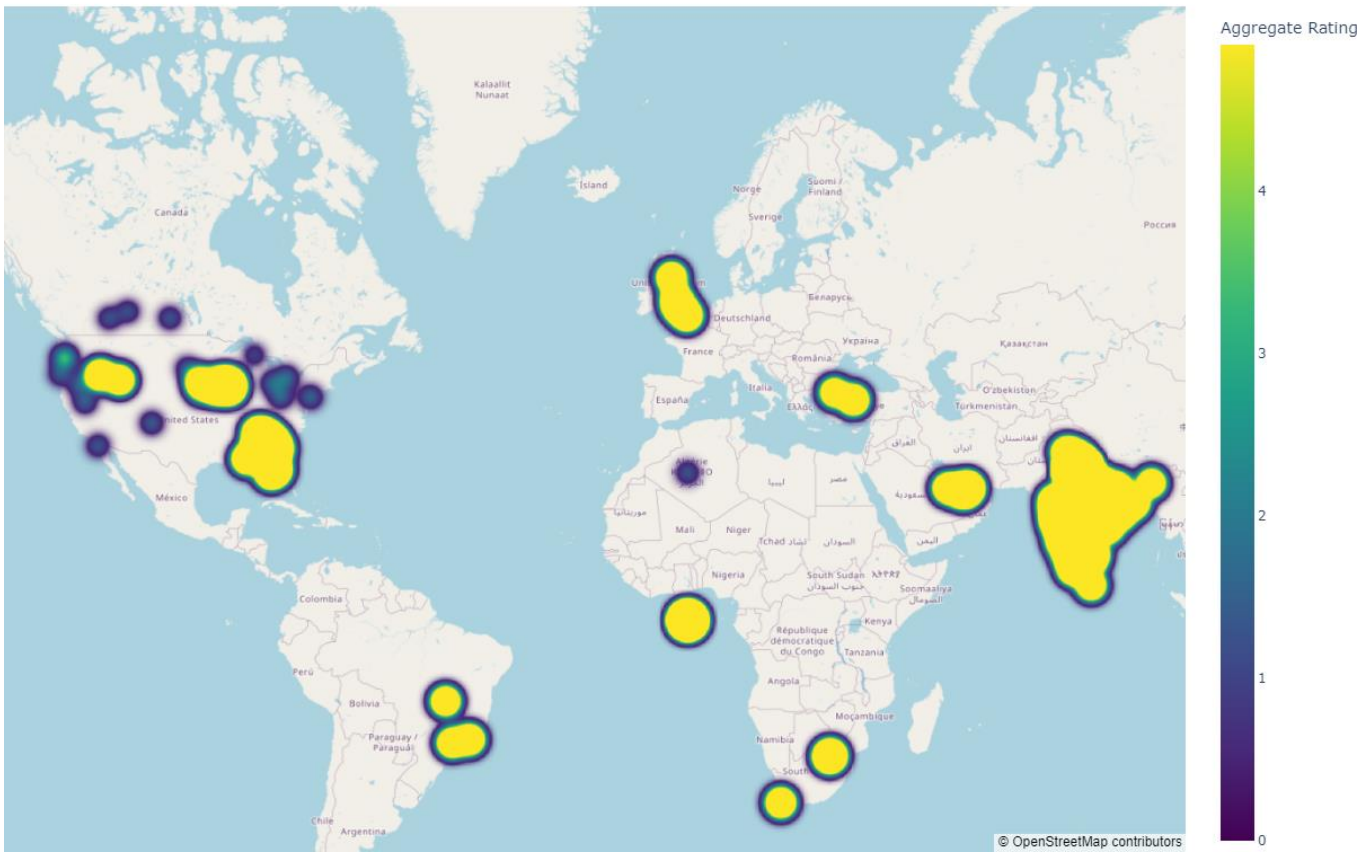
8. Restaurant Heatmap with Ratings 🗺️

Objective🎯:

- Create a heatmap to visualize the density of restaurant ratings geographically.

In [38]:
```
fig = px.density_mapbox(
    df,
    lat='Latitude',
    lon='Longitude',
    z='Aggregate rating',
    radius=20,
    color_continuous_scale='Viridis',
    mapbox_style='open-street-map',
    center=dict(lat=df['Latitude'].mean(), lon=df['Longitude'].mean()),
    zoom=11,
    title='Restaurant Heatmap with Ratings'
)
fig.update_layout(
    width=1200,
    height=800,
    title={'x': 0.5, 'xanchor': 'center'},
    margin=dict(l=0, r=0, t=50, b=0),
    coloraxis_colorbar=dict(
        title='Aggregate Rating',
        tickvals=[0, 1, 2, 3, 4, 5],
        ticktext=['0', '1', '2', '3', '4', '5']
    )
)
fig.show()
```

Restaurant Heatmap with Ratings



Insights 💡 :

- The heatmap visualizes the concentration of restaurant ratings, showing where higher or lower ratings are more prevalent. Areas with intense color indicate high rating density.

Conclusion ⬅️ END ✅:

The heatmap provides a spatial understanding of restaurant ratings, which can be useful for identifying high-performing and underperforming regions. This can guide targeted improvements or marketing strategies.

Overall Conclusion 📊

The analysis of restaurant ratings revealed a concentration of ratings between 0.0 and 4.0, highlighting variations across different cities and countries. The scatterplots of ratings versus latitude and longitude demonstrated geographic patterns in restaurant quality, indicating that location significantly influences ratings. The boxplots for the top cities and countries further illustrated diverse rating distributions, with some regions showing broader variability than others. These insights underscore the importance of understanding geographic and regional trends in restaurant performance, suggesting that targeted marketing and operational strategies could enhance customer satisfaction and business outcomes.