```
from IPython.display import Image
Image('/content/restaurant.png')
```

📊 Restaurant Data Analysis:

In this project,

We perform an exploratory data analysis (EDA) on a restaurant dataset to uncover valuable insights into dining trends across various cities and cuisines. The dataset includes information about restaurants such as their name, location, cuisines served, cost, ratings, votes, and services like online delivery and table booking.

Our goal is to analyze the data to answer key business questions, such as:

Which cuisines are most popular?

Which cities have the highest concentration of restaurants?

How do average ratings vary across cities?

What are the pricing trends in the restaurant industry?

How do services like online delivery or table booking impact ratings?

The analysis includes:

Identifying top cuisines and their distribution.

City-wise analysis of restaurant counts and average ratings.

Price range distribution across restaurants.

Visualizations using Seaborn and Matplotlib for better insights.

By the end of this project, we aim to derive meaningful patterns and trends that can help stakeholders in the food and restaurant industry make informed decisions.

```
# import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns


# Load the dataset

data= pd.read_csv("/content/Dataset .csv")

# Display the first five rows " to get an overview."
data.head()
```

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency
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	French, Japanese, Desserts	...	Botswana Pula(P)
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	Japanese	...	Botswana Pula(P)
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	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)
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	Japanese, Sushi	...	Botswana Pula(P)
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)

5 rows × 21 columns

```
# Display basic information about the dataset such as  column names, data types, and non-null counts.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant ID         9551 non-null   int64  
 1   Restaurant Name       9551 non-null   object 
 2   Country Code          9551 non-null   int64  
 3   City                  9551 non-null   object 
 4   Address               9551 non-null   object 
 5   Locality              9551 non-null   object 
 6   Locality Verbose      9551 non-null   object 
 7   Longitude             9551 non-null   float64
 8   Latitude              9551 non-null   float64
 9   Cuisines              9542 non-null   object 
 10  Average Cost for two  9551 non-null   int64  
 11  Currency              9551 non-null   object 
 12  Has Table booking     9551 non-null   object 
 13  Has Online delivery   9551 non-null   object 
 14  Is delivering now     9551 non-null   object 
 15  Switch to order menu  9551 non-null   object 
 16  Price range           9551 non-null   int64  
 17  Aggregate rating      9551 non-null   float64
 18  Rating color          9551 non-null   object 
 19  Rating text           9551 non-null   object 
 20  Votes                 9551 non-null   int64  
dtypes: float64(3), int64(5), object(13)
memory usage: 1.5+ MB
```

```
# Check for missing values in each column.
data.isnull().sum()
```

	0
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

```
data.describe()
```

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	156.909748
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	430.169145
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.500000	5.000000
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.200000	31.000000
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.700000	131.000000
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000

```
# Manually define irrelevant columns based on project requirements
irrelevant_cols = ["Address", "Locality Verbose", "Switch to order menu"]


# Drop irrelevant columns
data_cleaned = data.drop(columns=irrelevant_cols, errors="ignore")



# Fill missing values for categorical columns with mode (most frequent value)
cat_cols = data_cleaned.select_dtypes(include="object").columns
data_cleaned[cat_cols] = data_cleaned[cat_cols].fillna(data_cleaned[cat_cols].mode().iloc[0])

# Check if missing values are handled
 # Should print 0 if all missing values are handled
data_cleaned.isnull().sum().sum()
```

```
np.int64(0)
```

Level 1

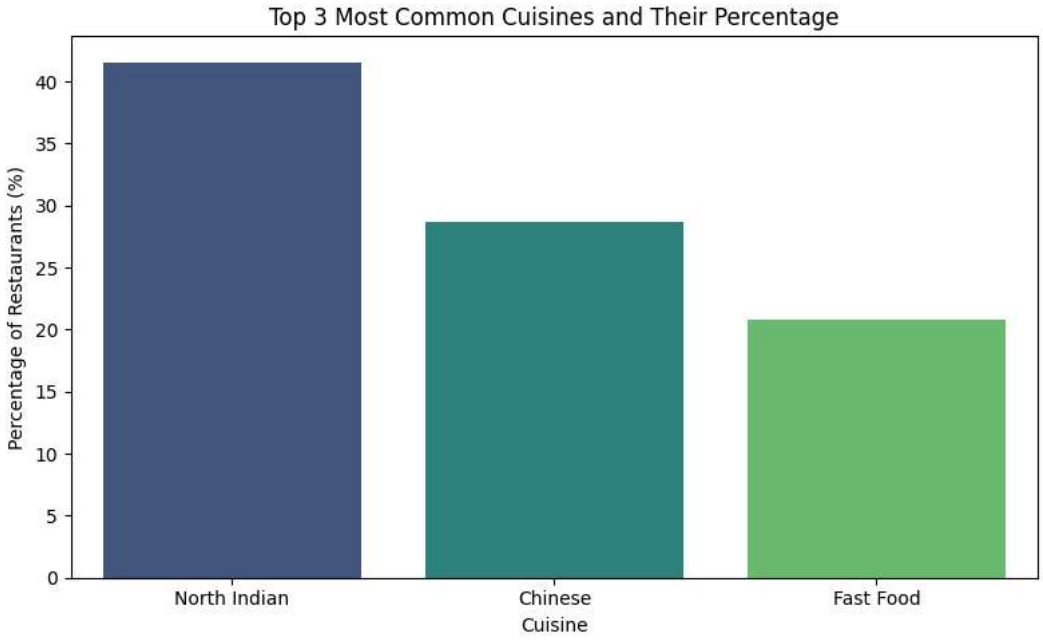
Task 1: Top Cuisines

Determine the top three most common cuisines in the dataset.

Calculate the percentage of restaurants that serve each of the top cuisines.

```
# Split the 'Cuisines' column to account for multiple cuisines per restaurantfrom collections import Counter
from collections import Counter

# Flatten the list of cuisines and count occurrences
all_cuisines = data_cleaned["Cuisines"].dropna().str.split(", ").sum()
cuisine_counts = Counter(all_cuisines)

# Get the top three cuisines
top_cuisines = cuisine_counts.most_common(3)
print("Top 3 Cuisines:", top_cuisines)

# Convert to DataFrame for easier calculation
top_cuisines_data = pd.DataFrame(top_cuisines, columns=["Cuisine", "Count"])

# Calculate the percentage of restaurants serving each of these cuisines
total_restaurants = len(data_cleaned)
top_cuisines_data["Percentage"] = (top_cuisines_data["Count"] / total_restaurants) * 100

# Display the DataFrame
print(top_cuisines_data)

# Visualization
plt.figure(figsize=(8, 5))
sns.barplot(x="Cuisine", y="Percentage", data=top_cuisines_data, palette="viridis")
plt.title("Top 3 Most Common Cuisines and Their Percentage")
plt.ylabel("Percentage of Restaurants (%)")
plt.xlabel("Cuisine")
plt.tight_layout()
plt.show()
```

```
Top 3 Cuisines: [('North Indian', 3969), ('Chinese', 2735), ('Fast Food', 1986)]
        Cuisine  Count  Percentage
0  North Indian   3969   41.555858
1       Chinese   2735   28.635745
2     Fast Food   1986   20.793634
<ipython-input-14-69949343788d>:24: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x="Cuisine", y="Percentage", data=top_cuisines_data, palette="viridis")
```



TOP 3 CUISINES

	Cuisine	Count	Percentage
1	North Indian	3969	41.55
2	Chinese	2735	28.63
3	Fast Food	1986	20.79

Level1

Task 2:City Analysis

Identify the city with the highest number of restaurants in the dataset.

Calculate the average rating for restaurants in each city.

Determine the city with the highest average rating.

```
# Identify the city with the highest number of restaurants
city_counts = data_cleaned["City"].value_counts()

# City with the highest number of restaurants
top_city = city_counts.idxmax()
print(f"City with the highest number of restaurants: {top_city} ({city_counts.max()} restaurants)")

# Calculate the average rating for restaurants in each city
city_avg_rating = data_cleaned.groupby("City")["Aggregate rating"].mean().sort_values(ascending=False)
print(f"The average rating for restaurants in each city")
print(data_cleaned.groupby("City")["Aggregate rating"].mean().sort_values(ascending=False).to_string())


# Identify the city with the highest average rating
highest_avg_rating_city = city_avg_rating.idxmax()
highest_avg_rating = city_avg_rating.max()
print(f"City with the highest average rating: {highest_avg_rating_city} ({highest_avg_rating:.2f})")

# Convert to DataFrame for visualization
top_10_cities = city_counts.head(10)  # Top 10 cities with most restaurants
city_avg_rating_df = city_avg_rating.head(10)  # Top 10 cities by average rating

# Plot top 10 cities with most restaurants
plt.figure(figsize=(10, 5))
sns.barplot(x=top_10_cities.index, y=top_10_cities.values, palette="mako")
plt.xticks(rotation=45)
plt.title("Top 10 Cities with Most Restaurants")
plt.xlabel("City")
plt.ylabel("Number of Restaurants")
plt.show()

# Plot top 10 cities with highest average ratings
plt.figure(figsize=(10, 5))
sns.barplot(x=city_avg_rating_df.index, y=city_avg_rating_df.values, palette="coolwarm")
plt.xticks(rotation=45)
plt.title("Top 10 Cities with Highest Average Restaurant Ratings")
plt.xlabel("City")
plt.ylabel("Average Rating")
plt.ylim(0, 5)  # Ratings are between 0-5
plt.show()
```

```
City with the highest number of restaurants: New Delhi (5473 restaurants)
The average rating for restaurants in each city
City
Inner City                4.900000
Quezon City               4.800000
Makati City               4.650000
Pasig City                4.633333
Mandaluyong City          4.625000
Beechworth                4.600000
London                    4.535000
Taguig City               4.525000
Secunderabad              4.500000
Lincoln                   4.500000
Tagaytay City             4.500000
Orlando                   4.475000
Tampa Bay                 4.410000
Rest of Hawaii            4.410000
Palm Cove                 4.400000
Tanunda                   4.400000
Bangalore                 4.375000
Dubai                     4.370000
Pasay City                4.366667
Jakarta                   4.356250
Hyderabad                 4.344444
Chennai                   4.315000
Ankara                    4.305000
Tangerang                 4.300000
Abu Dhabi                 4.300000
Vineland Station          4.300000
Vernonia                  4.300000
Mohali                    4.300000
Sandton                   4.300000
Randburg                  4.300000
Clatskanie                4.300000
��stanbul                  4.292857
Auckland                  4.275000
Rio de Janeiro            4.265000
Boise                     4.260000
Kolkata                   4.255000
San Juan City             4.250000
Wellington City           4.250000
Goa                       4.245000
Des Moines                4.235000
Pune                      4.220000
Panchkula                 4.200000
Athens                    4.200000
Johannesburg              4.200000
Bandung                   4.200000
Pensacola                 4.200000
Lucknow                   4.195238
Guwahati                  4.190476
Pretoria                  4.190000
Cedar Rapids/Iowa City    4.165000
Ahmedabad                 4.161905
Savannah                  4.155000
Coimbatore                4.135000
Jaipur                    4.130000
Augusta                   4.130000
Macon                     4.115000
Dalton                    4.110000
Cape Town                 4.110000
East Ballina              4.100000
Auckland                  4.100000
Trentham East             4.100000
Edinburgh                 4.090000
Mumbai                    4.085000
Kochi                     4.080000
Doha                      4.060000
Chandigarh                4.050000
Dehradun                  4.050000
Manchester                4.045000
Gainesville               4.035000
Sharjah                   4.030000
Columbus                  4.030000
Vadodara                  4.025000
Vizag                     4.005000
Princeton                 4.000000
Bhubaneshwar              3.980952
Ludhiana                  3.980000
```

Task 2:City Analysis

City with the highest number of restaurants: ***New Delhi*** (5473 restaurants)

The average rating for restaurants in each city

City with the highest average rating: ***Inner City*** (4.90)

Level 1

Task 3: Price Range Distribution

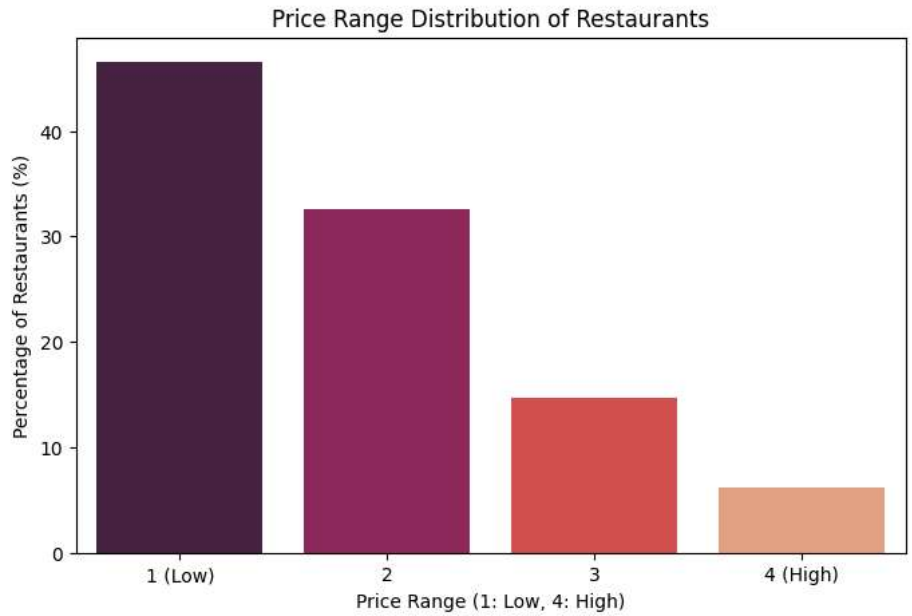
Create a histogram or bar chart to visualize the distribution of price range samong the restaurants.

Calculate the percentage of restaurants in each price range category.

```
# Count the number of restaurants in each price range
price_counts = data_cleaned["Price range"].value_counts().sort_index()

# Calculate the percentage of restaurants in each price range
price_percentage = (price_counts / len(data_cleaned)) * 100

# Convert to DataFrame for visualization
price_data = pd.DataFrame({"Price Range": price_counts.index, "Percentage": price_percentage.values})
```

```
# Print the percentage breakdown
print("Percentage of Restaurants in Each Price Range:")
print(price_data)

# Plot the distribution of price ranges
plt.figure(figsize=(8, 5))
sns.barplot(x=price_data["Price Range"], y=price_data["Percentage"], palette="rocket")
plt.title("Price Range Distribution of Restaurants")
plt.xlabel("Price Range (1: Low, 4: High)")
plt.ylabel("Percentage of Restaurants (%)")
plt.xticks(ticks=[0, 1, 2, 3], labels=["1 (Low)", "2", "3", "4 (High)"])
plt.show()
```

```
Percentage of Restaurants in Each Price Range:
   Price Range  Percentage
0            1   46.529159
1            2   32.593446
2            3   14.741912
3            4    6.135483
/tmp/ipython-input-16-6462441d0877.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

sns.barplot(x=price_data["Price Range"], y=price_data["Percentage"], palette="rocket")
```




```
Birmingham                3.680000
Waterloo                  3.650000
Monroe                    3.600000
Victor Harbor             3.600000
Ojo Caliente              3.600000
Lakeview                  3.600000
Dicky Beach               3.600000
Lorn                      3.600000
Singapore                 3.575000
Albany                    3.555000
Dubuque                   3.535000
Noida                     2.036204
Faridabad                 1.866932
City with the highest average rating: Inner City (4.90)
```

Percentage of Restaurants in Each Price Range:

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

                                                    palette="mako")
```

Price Range	percentage
1	46.53
2	32.59
3	14.74
4	6.13

Level 1

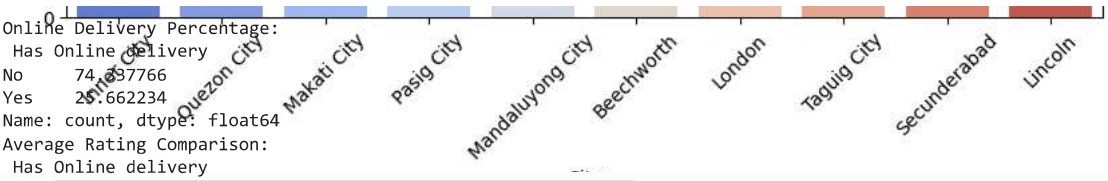
Task 4: Online Delivery

Determine the percentage of restaurants that offer online delivery.

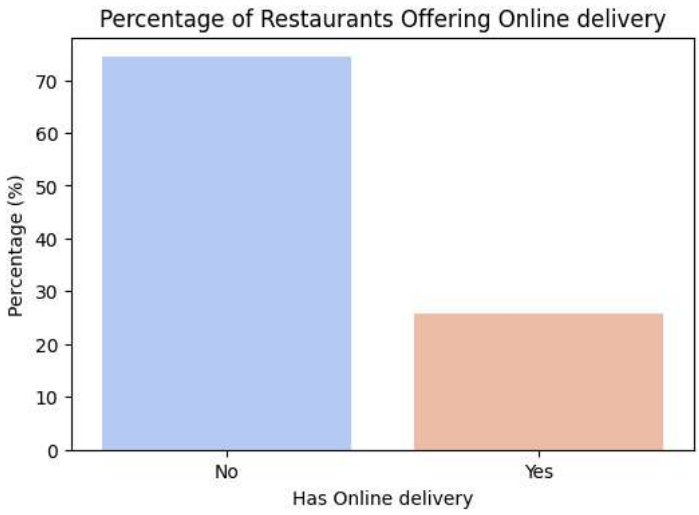
Compare the average ratings of restaurants with and without online delivery.

```
# Count the number of restaurants offering online delivery

online_delivery_counts = data["Has Online delivery"].value_counts()

# Calculate the percentage of restaurants offering online delivery
```

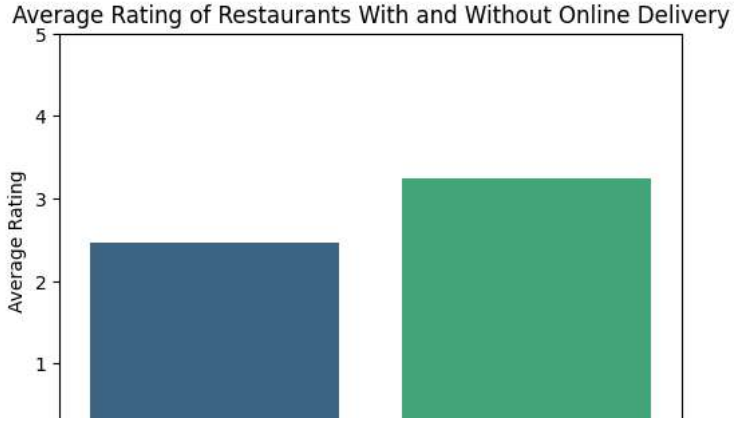
```
online_delivery_percentage = (online_delivery_counts / len(data)) * 100
print("Online Delivery Percentage:\n", online_delivery_percentage)

# Compare the average rating of restaurants with and without online delivery

avg_rating_online = data.groupby("Has Online delivery")["Aggregate rating"].mean()
print("Average Rating Comparison:\n", avg_rating_online)

# Visualization - Online Delivery Percentage
plt.figure(figsize=(6, 4))
sns.barplot(x=online_delivery_percentage.index, y=online_delivery_percentage.values, palette="coolwarm")
plt.xticks([0, 1], ["No", "Yes"])
plt.title("Percentage of Restaurants Offering Online delivery")
plt.ylabel("Percentage (%)")
plt.xlabel("Has Online delivery")
plt.show()

# Visualization - Average Rating Comparison
plt.figure(figsize=(6, 4))
sns.barplot(x=avg_rating_online.index, y=avg_rating_online.values, palette="viridis")
plt.xticks([0, 1], ["No", "Yes"])
plt.title("Average Rating of Restaurants With and Without Online Delivery")
plt.ylabel("Average Rating")
plt.xlabel("Has Online delivery")
plt.ylim(0, 5)  # Ratings are between 0-5
plt.show()
```




```
Online Delivery Percentage:
 Has Online delivery
No     74.337766
Yes    25.662234
Name: count, dtype: float64
Average Rating Comparison:
 Has Online delivery
No     2.465296
Yes    3.248837
Name: Aggregate rating, dtype: float64
<ipython-input-17-6b4710c4a37a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.barplot(x=online_delivery_percentage.index, y=online_delivery_percentage.values, palette="coolwarm")
```




```
<ipython-input-17-6b4710c4a37a>:26: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.barplot(x=avg_rating_online.index, y=avg_rating_online.values, palette="viridis")
```



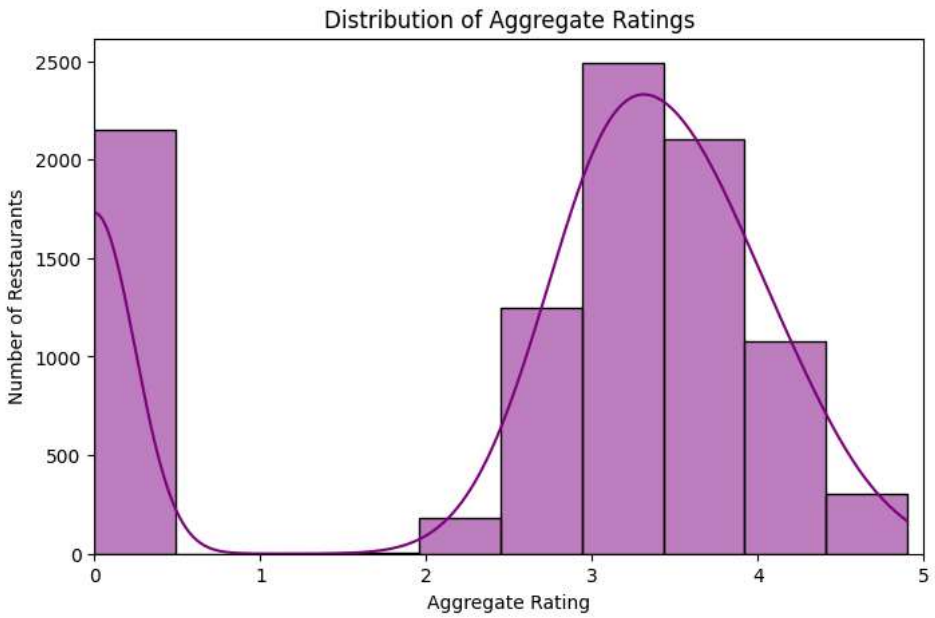
Level 2

Task 1 :Restaurant Ratings

Analyze the distribution of aggregate ratings and determine the most common rating range.

Calculate the average number of votes received by restaurants.

```
# Analyze the distribution of aggregate ratings

plt.figure(figsize=(8, 5))
sns.histplot(data["Aggregate rating"], bins=10, kde=True, color="purple")
plt.title("Distribution of Aggregate Ratings")
plt.xlabel("Aggregate Rating")
plt.ylabel("Number of Restaurants")
plt.xlim(0, 5)  # Ratings are between 0-5
plt.show()

# Determine the most common rating range

rating_bins = pd.cut(data["Aggregate rating"], bins=[0, 1, 2, 3, 4, 5], labels=["0-1", "1-2", "2-3", "3-4", "4-5"])
rating_counts = rating_bins.value_counts().sort_index()
print("Most Common Rating Range:\n", rating_counts)

# Calculate the average number of votes received by restaurants

avg_votes = data["Votes"].mean()
print(f"Average Number of Votes Received by Restaurants: {avg_votes:.2f}")

# Visualization - Most Common Rating Range

plt.figure(figsize=(8, 5))
sns.barplot(x=rating_counts.index, y=rating_counts.values, palette="magma")
plt.title("Most Common Aggregate Rating Range")
plt.xlabel("Rating Range")
plt.ylabel("Number of Restaurants")
plt.show()
```




```
Most Common Rating Range:
 Aggregate rating
0-1       0
1-2      10
2-3    1891
3-4    4388
4-5    1114
Name: count, dtype: int64
Average Number of Votes Received by Restaurants: 156.91
<ipython-input-18-4c87c2947b7e>:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.barplot(x=rating_counts.index, y=rating_counts.values, palette="magma")
```

Most Common Rating Range:

Aggregate Rating	No of Restaurants
0-1	0
1-2	10
2-3	1891
3-4	4388
4-5	1114

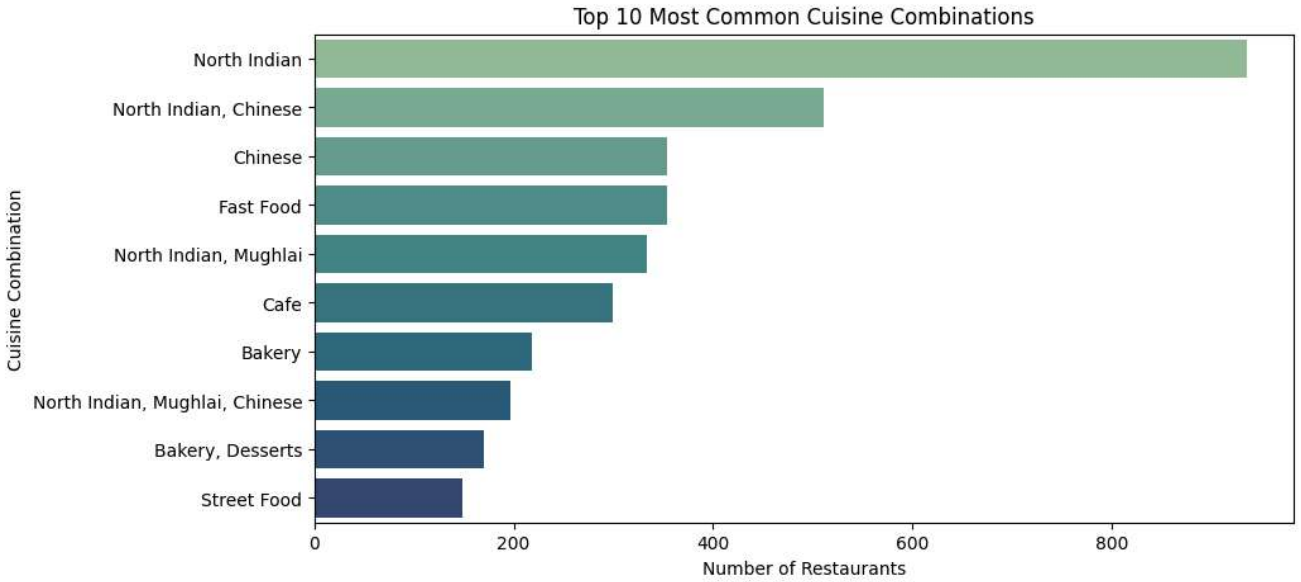
Level 2

Task: Cuisine Combination

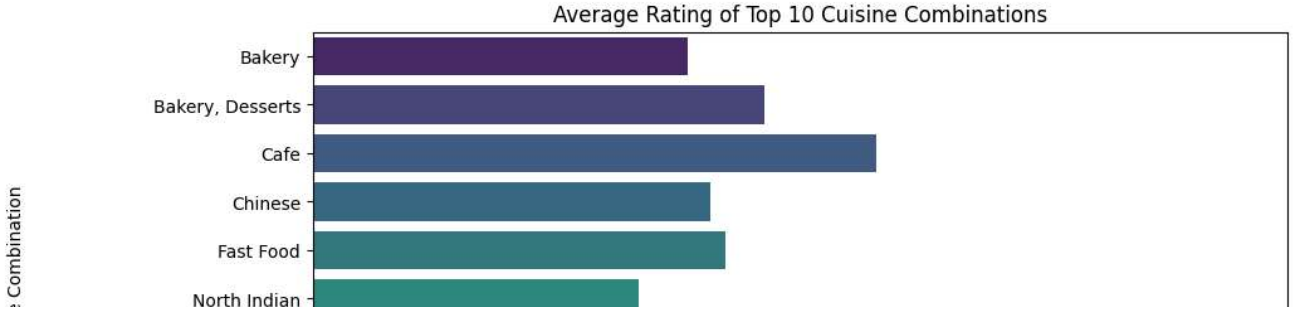
Identify the most common combinations of cuisines in the dataset. Determine if certain cuisine combinations tend to have higher ratings.

```
# Count occurrences of each cuisine combination

cuisine_combinations = data["Cuisines"].value_counts().head(10)  # Top 10 most common combinations
print("Top 10 Most Common Cuisine Combinations:\n", cuisine_combinations)

# Calculate average rating for top cuisine combinations
top_cuisine_ratings = data[data["Cuisines"].isin(cuisine_combinations.index)].groupby("Cuisines")["Aggregate rating"].mean()

# Visualization - Most Common Cuisine Combinations
plt.figure(figsize=(10, 5))
sns.barplot(y=cuisine_combinations.index, x=cuisine_combinations.values, palette="crest")
plt.title("Top 10 Most Common Cuisine Combinations")
plt.xlabel("Number of Restaurants")
plt.ylabel("Cuisine Combination")
plt.show()

# Visualization - Average Rating of Top Cuisine Combinations
plt.figure(figsize=(10, 5))
sns.barplot(y=top_cuisine_ratings.index, x=top_cuisine_ratings.values, palette="viridis")
plt.title("Average Rating of Top 10 Cuisine Combinations")
plt.xlabel("Average Rating")
plt.ylabel("Cuisine Combination")
plt.xlim(0, 5)  # Ratings are between 0-5
plt.show()
```

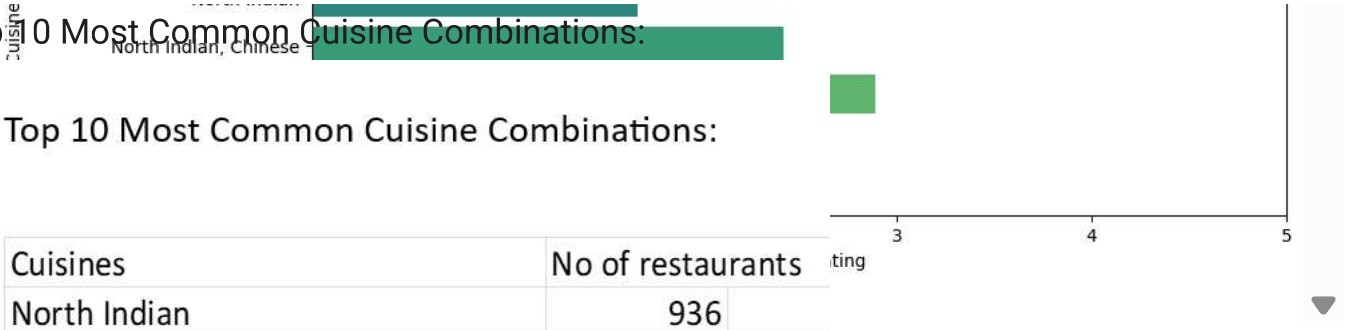
```
Top 10 Most Common Cuisine Combinations:
 Cuisines
North Indian                      936
North Indian, Chinese             511
Chinese                           354
Fast Food                         354
North Indian, Mughlai             334
Cafe                              299
Bakery                            218
North Indian, Mughlai, Chinese    197
Bakery, Desserts                  170
Street Food                       149
Name: count, dtype: int64
<ipython-input-19-0d4921a3771e>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(y=cuisine_combinations.index, x=cuisine_combinations.values, palette="crest")
```




```
<ipython-input-19-0d4921a3771e>:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(y=top_cuisine_ratings.index, x=top_cuisine_ratings.values, palette="viridis")
```



Top 10 Most Common Cuisine Combinations:



Top 10 Most Common Cuisine Combinations:

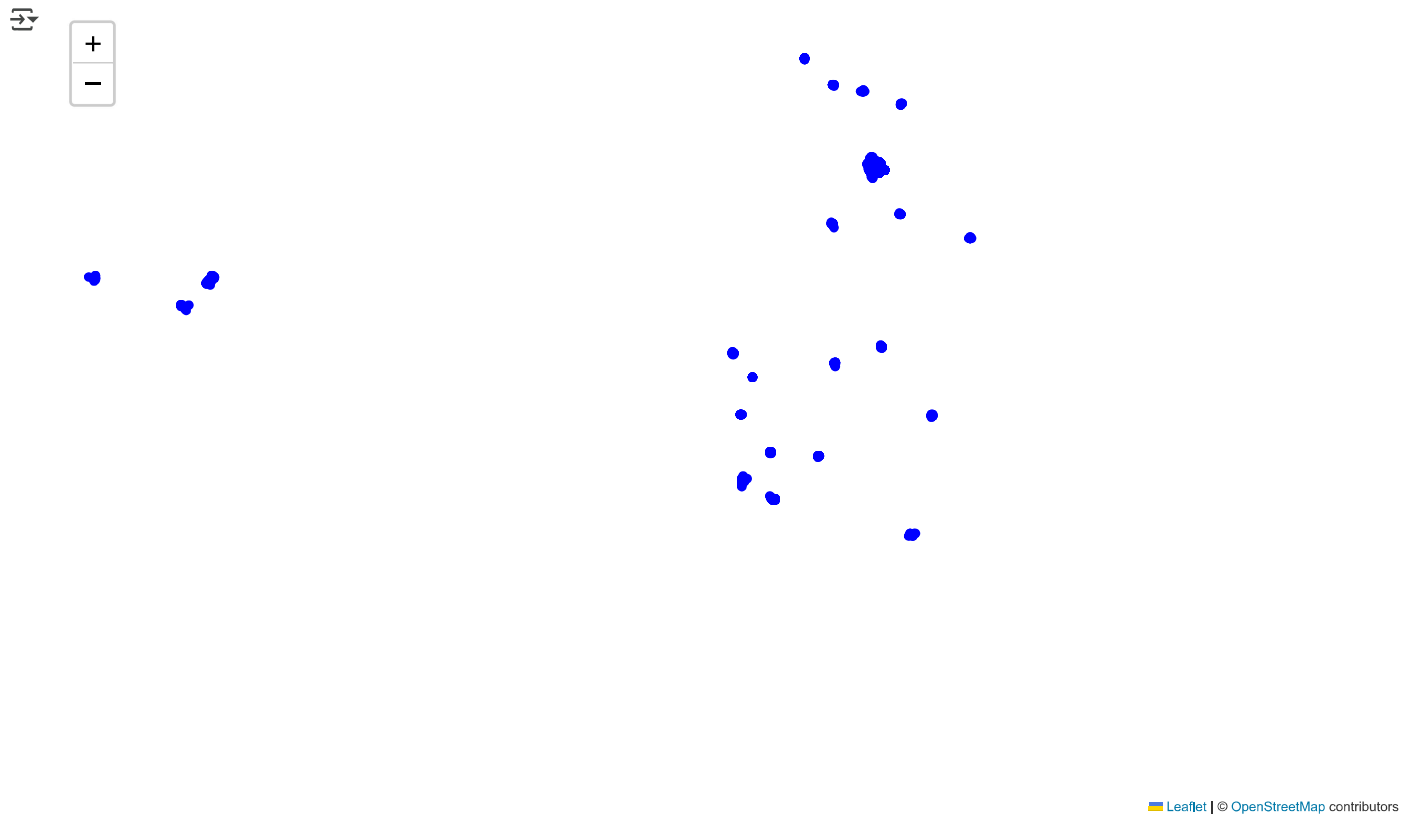
Cuisines	No of restaurants
North Indian	936
North Indian, Chinese	511
Chinese	354
Fast food	354
North Indian, Mughlai	334
Café	299
Bakery	218
North Indian, Mughlai,Chinese	197
Bakery, Desserts	170
Street Food	149

Level 2

Task 3: Geographic Analysis

Plot the locations of restaurants on a map using longitude and latitude coordinates. Identify any patterns or clusters of restaurants in specific areas.

```
import folium
from folium.plugins import HeatMap

# Create a base map centered around the mean latitude and longitude
map_center = [data_cleaned["Latitude"].mean(), data_cleaned["Longitude"].mean()]
restaurant_map = folium.Map(location=map_center, zoom_start=5)

# Add restaurant locations to the map
for _, row in data_cleaned.iterrows():
    folium.CircleMarker(
        location=[row["Latitude"], row["Longitude"]],
        radius=2,
        color="blue",
        fill=True,
        fill_color="blue",
        fill_opacity=0.5,
    ).add_to(restaurant_map)

# Display the map
restaurant_map
```




```
# Create a heatmap to identify clusters
heatmap_data = data_cleaned[["Latitude", "Longitude"]].dropna().values
restaurant_heatmap = folium.Map(location=map_center, zoom_start=5)
HeatMap(heatmap_data, radius=10).add_to(restaurant_heatmap)

# Display the heatmap
restaurant_heatmap
```

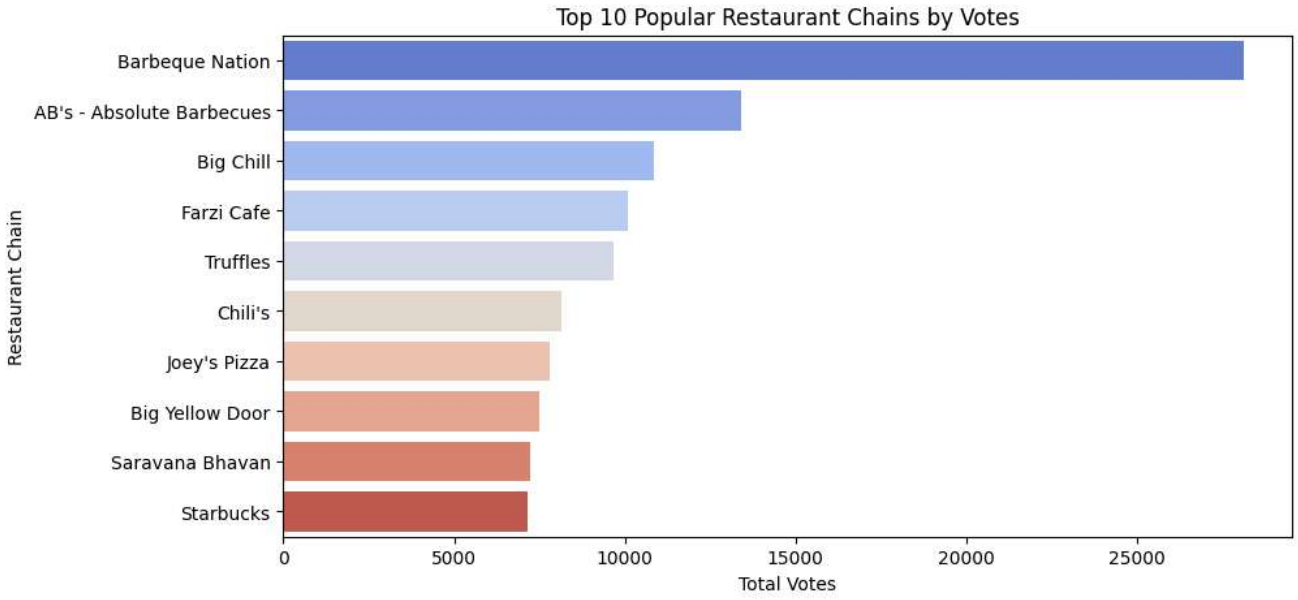
Level 2

Task 4: Restaurant Chains

Identify if there are any restaurant chains present in the dataset. Analyze the ratings and popularity of different restaurant chains.

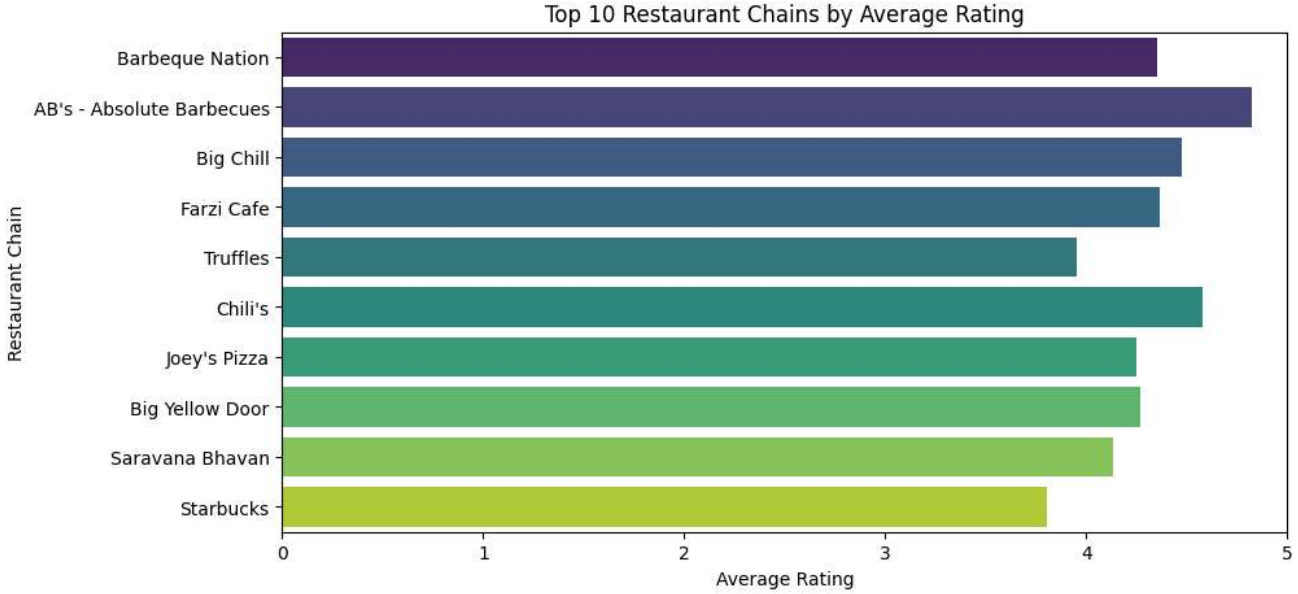
```
# Identify restaurant chains (restaurants with the same name appearing multiple times)
chain_counts = data_cleaned["Restaurant Name"].value_counts()
restaurant_chains = chain_counts[chain_counts > 1]  # Only consider names appearing more than once

print("Top 10 Restaurant Chains:\n", restaurant_chains.head(10))

# Calculate average rating and votes for each chain
chain_analysis = data_cleaned[data_cleaned["Restaurant Name"].isin(restaurant_chains.index)].groupby("Restaurant Name").agg(
    {"Aggregate rating": "mean", "Votes": "sum"}  # Average rating, total votes
).sort_values(by="Votes", ascending=False)  # Sort by popularity (votes)

print("\nTop 10 Popular Chains by Votes:\n", chain_analysis.head(10))

# Visualization - Top 10 Restaurant Chains by Votes
plt.figure(figsize=(10, 5))
sns.barplot(y=chain_analysis.head(10).index, x=chain_analysis["Votes"].head(10), palette="coolwarm")
plt.title("Top 10 Popular Restaurant Chains by Votes")
plt.xlabel("Total Votes")
plt.ylabel("Restaurant Chain")
plt.show()

# Visualization - Top 10 Chains by Average Rating
plt.figure(figsize=(10, 5))
sns.barplot(y=chain_analysis.head(10).index, x=chain_analysis["Aggregate rating"].head(10), palette="viridis")
plt.title("Top 10 Restaurant Chains by Average Rating")
plt.xlabel("Average Rating")
plt.ylabel("Restaurant Chain")
plt.xlim(0, 5)  # Ratings are between 0-5
plt.show()
```

```
Top 10 Restaurant Chains:
 Restaurant Name
Cafe Coffee Day     83
Domino's Pizza      79
Subway              63
Green Chick Chop    51
McDonald's          48
Keventers           34
Pizza Hut           30
Giani               29
Baskin Robbins      28
Barbeque Nation     26
Name: count, dtype: int64

Top 10 Popular Chains by Votes:
                          Aggregate rating  Votes
Restaurant Name
Barbeque Nation                   4.353846  28142
AB's - Absolute Barbecues         4.825000  13400
Big Chill                         4.475000  10853
Farzi Cafe                        4.366667  10098
Truffles                          3.950000   9682
Chili's                           4.580000   8156
Joey's Pizza                      4.250000   7807
Big Yellow Door                   4.266667   7511
Saravana Bhavan                   4.133333   7238
Starbucks                         3.805556   7139
<ipython-input-22-887817d0dd4e>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(y=chain_analysis.head(10).index, x=chain_analysis["Votes"].head(10), palette="coolwarm")
```



Top 10 Popular Chains by Votes:

```
<ipython-input-22-887817d0dd4e>:24: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(y=chain_analysis.head(10).index, x=chain_analysis["Aggregate rating"].head(10), palette="viridis")
```



Restaurant Name	Aggregate rating	Votes
Barbeque Nation	4.353846	28142
AB's - Absolute Barbecues	4.825	13400
Big Chill	4.475	10853
Farzi Cafe	4.366667	10098
Truffles	3.95	9682
Chili's	4.58	8156
Joey's Pizza	4.25	7807
Big Yellow Door	4.266667	7511
Saravana Bhavan	4.133333	7238

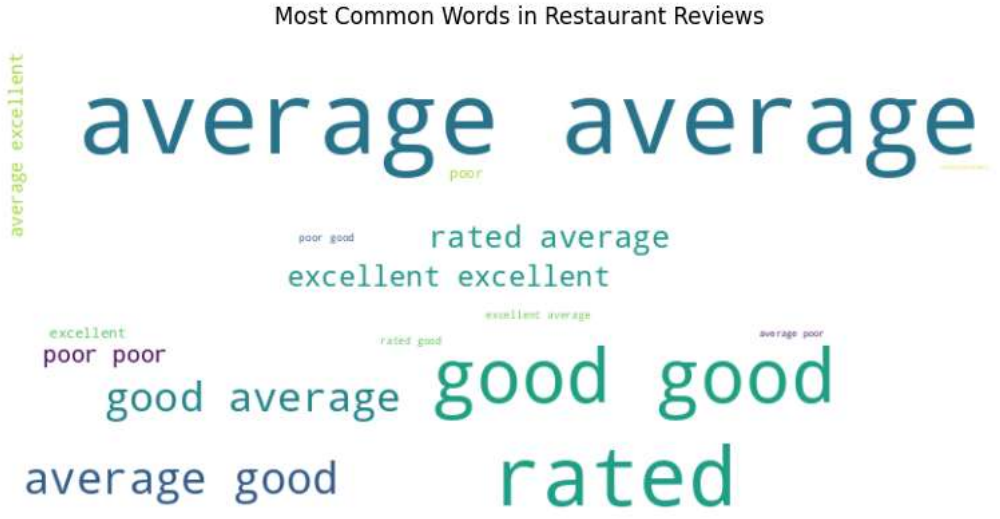
Level 3

Task 1: Restaurant Reviews

Analyze the text reviews to identify the most common positive and negative keywords. Calculate the average length of reviews and explore if there is a relationship between review length and rating.

```
from collections import Counter
import re
from wordcloud import WordCloud

# Assuming there is a 'Reviews' column with customer reviews
data["Reviews"] = data_cleaned["Rating text"].fillna("")  # Replace NaN values with empty strings

# Function to clean text (remove special characters, numbers, etc.)
def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r"[^\w\s]", "", text)  # Remove punctuation
    text = re.sub(r"\d+", "", text)  # Remove numbers
    return text

data["Reviews"] = data_cleaned["Rating text"].apply(clean_text)

# Tokenization - Split words for frequency analysis
all_words = " ".join(data["Reviews"]).split()
word_counts = Counter(all_words)

# Get most common words
common_words = word_counts.most_common(20)
print("Most Common Words in Reviews:\n", common_words)

# Visualization - Word Cloud of Reviews
wordcloud = WordCloud(width=800, height=400, background_color="white").generate(" ".join(all_words))
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Most Common Words in Restaurant Reviews")
plt.show()
```

```
Most Common Words in Reviews:
 [('average', 3737), ('good', 3179), ('not', 2148), ('rated', 2148), ('very', 1079), ('excellent', 301), ('poor', 186)]
```



Most Common Words in Reviews

Words in Reviews	Total To of Reviews
Average	3737
Good	3179
Not	2148
Rated	2148
Very	10779
Excellent	301
Poor	186

```
# Define basic positive and negative words
positive_words = {"good", "great", "excellent", "amazing", "delicious", "tasty", "awesome", "fantastic"}
negative_words = {"bad", "worst", "terrible", "disgusting", "poor", "awful", "horrible", "bland"}

# Count occurrences of positive and negative words in reviews
positive_count = sum(word_counts[word] for word in positive_words if word in word_counts)
negative_count = sum(word_counts[word] for word in negative_words if word in word_counts)

print(f"Positive Words Count: {positive_count}")
print(f"Negative Words Count: {negative_count}")
```

```
Positive Words Count: 3480
Negative Words Count: 186
```

Positive Words Count: 3480

Negative Words Count: 186

Level 3

Task2: Votes Analysis

Identify the restaurants with the highest and lowest number of votes. Analyze if there is a correlation between the number of votes and the rating of a restaurant.

```
# 1. Restaurant with Highest Votes
max_votes_row = data_cleaned[data_cleaned["Votes"] == data_cleaned["Votes"].max()]
print("🏆 Restaurant with Highest Votes:")
print(max_votes_row[["Restaurant Name", "City", "Votes", "Aggregate rating"]])

# 2. Restaurant with Lowest Non-zero Votes (Fixing the error)
non_zero_votes_data_cleaned = data_cleaned[data_cleaned["Votes"] > 0]
min_votes_row = non_zero_votes_data_cleaned[non_zero_votes_data_cleaned["Votes"] == non_zero_votes_data_cleaned["Votes"].min()]
print("\n🍽️ Restaurant with Lowest Non-zero Votes:")
print(min_votes_row[["Restaurant Name", "City", "Votes", "Aggregate rating"]])

# 3. Correlation between Votes and Rating
correlation = data_cleaned["Votes"].corr(data_cleaned["Aggregate rating"])
print(f"\n📈 Correlation between Votes and Aggregate Rating: {correlation:.2f}")
```

```
🏆 Restaurant with Highest Votes:
     Restaurant Name       City  Votes  Aggregate rating
728            Toit  Bangalore  10934               4.8

🍽️ Restaurant with Lowest Non-zero Votes:
            Restaurant Name            City  Votes  Aggregate rating
58    Quiosque Chopp Brahma  Rio de Janeiro      1               0.0
412             Nosh Mahal       Pocatello      1               0.0
871   Aggarwal Sweet Corner       Faridabad      1               0.0
878            Aapki Rasoi       Faridabad      1               0.0
888         Popcorn Fusion       Faridabad      1               0.0
...                    ...             ...    ...               ...
9100    Chocolate Fountain           Noida      1               0.0
9101            McDonald's           Noida      1               0.0
9102             Tea Trails          Noida      1               0.0
9109          Bread & Pasta          Noida      1               0.0
9112              The Grand          Noida      1               0.0

[483 rows x 4 columns]

📈 Correlation between Votes and Aggregate Rating: 0.31
```

Restaurant with Highest Votes:10934

Restaurant Name.....Toiet

City.....Banglore

Aggregate Rating.....4.8

Correration between Votes and Aggregate Rating: 0.31

```
# 1. Most common rating
common_rating = data_cleaned["Aggregate rating"].mode()[0]
print(f"Most Common Aggregate Rating: {common_rating}")

# 2. Average number of votes
average_votes = data_cleaned["Votes"].mean()
print(f"Average Number of Votes: {average_votes:.2f}")

# 3. Distribution of Ratings
plt.figure(figsize=(10, 5))
sns.countplot(x="Aggregate rating", data=data_cleaned, palette="coolwarm")
plt.title("Distribution of Aggregate Ratings")
plt.xlabel("Rating")
plt.ylabel("Number of Restaurants")
plt.show()

# 4. Distribution of Votes
plt.figure(figsize=(10, 5))
sns.histplot(data_cleaned["Votes"], bins=30, kde=True, color="skyblue")
plt.title("Distribution of Votes")
plt.xlabel("Votes")
```