```
from google.colab import files
uploaded = files.upload()
```

Choose Files Dataset .csv

- **Dataset .csv**(text/csv) - 2249716 bytes, last modified: 6/30/2025 - 100% done

```
Saving Dataset .csv to Dataset  (1).csv
```

```
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
df = pd.read_csv("Dataset .csv")
df.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

Level 1

Task 1

```
df['Cuisines'].isnull().sum()
df['Cuisines'] = df['Cuisines'].fillna('Unknown')
```

```
all_cuisines = ','.join(df['Cuisines']).split(',')  # Split all cuisines
all_cuisines = [c.strip() for c in all_cuisines if c.strip()]  # Clean whitespace
# Count occurrences
cuisine_counter = Counter(all_cuisines)
# Get top 3
top_3_cuisines = cuisine_counter.most_common(3)
top_3_cuisines
```

```
[('North Indian', 3960), ('Chinese', 2735), ('Fast Food', 1986)]
```

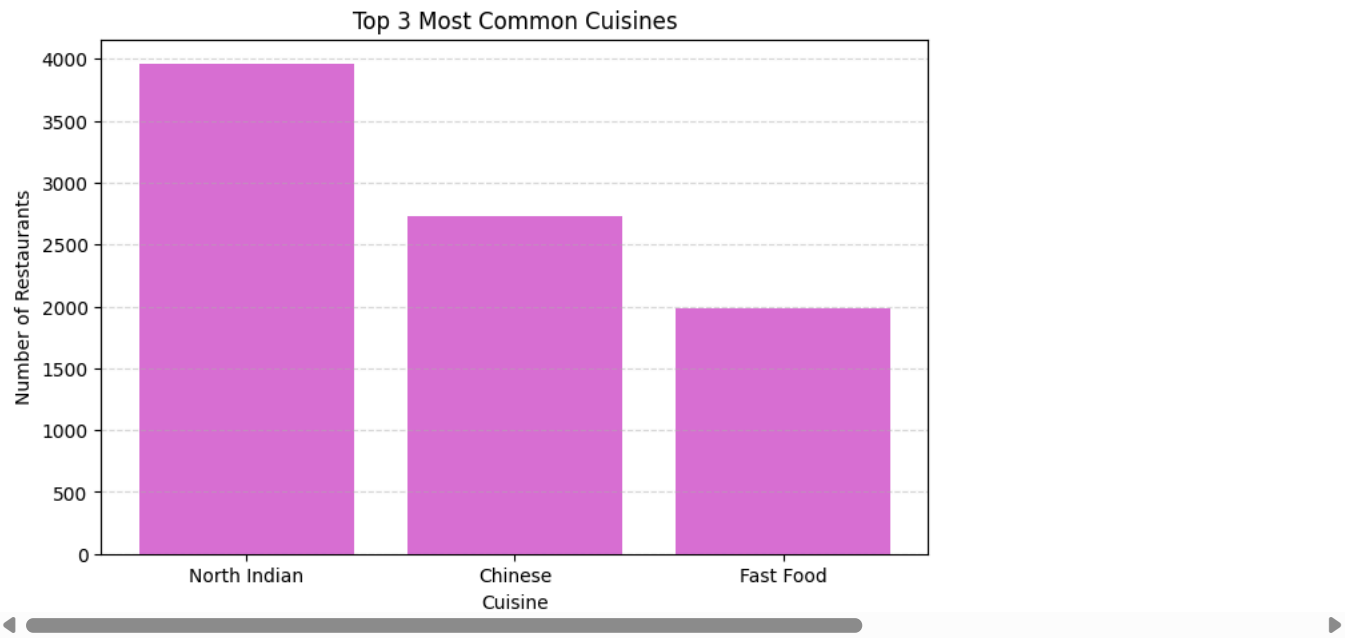
What can I help you build?

```
total_restaurants = len(df)
print("Total restaurants:", total_restaurants)
for cuisine, _ in top_3_cuisines:
```

```
    count = df['Cuisines'].str.contains(cuisine, case=False).sum()
    percentage = (count / total_restaurants) * 100
    print(f"{cuisine}: {count} restaurants, {percentage:.2f}%")
```

```
Total restaurants: 9551
North Indian: 3960 restaurants, 41.46%
Chinese: 2733 restaurants, 28.61%
Fast Food: 1987 restaurants, 20.80%
```

```
labels = [item[0] for item in top_3_cuisines]
counts = [df['Cuisines'].str.contains(c, case=False).sum() for c in labels]
plt.figure(figsize=(8, 5))
plt.bar(labels, counts, color='orchid')
plt.title('Top 3 Most Common Cuisines')
plt.xlabel('Cuisine')
plt.ylabel('Number of Restaurants')
plt.grid(True, axis='y', linestyle='--', alpha=0.5)
plt.show()
```



Task 2

```
# STEP 1: Handle missing values (if any)
df['City'] = df['City'].fillna('Unknown')
df['Aggregate rating'] = pd.to_numeric(df['Aggregate rating'], errors='coerce')
```

```
# STEP 2: City with the highest number of restaurants
city_counts = df['City'].value_counts()
top_city = city_counts.idxmax()
top_city_count = city_counts.max()
print(f"City with the highest number of restaurants: {top_city} ({top_city_count} restaurants)")
```

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

```
# STEP 3: Average rating for each city
city_avg_rating = df.groupby('City')['Aggregate rating'].mean().sort_values(ascending=False)
# Display top 10 cities by average rating
city_avg_rating.head(10)
```

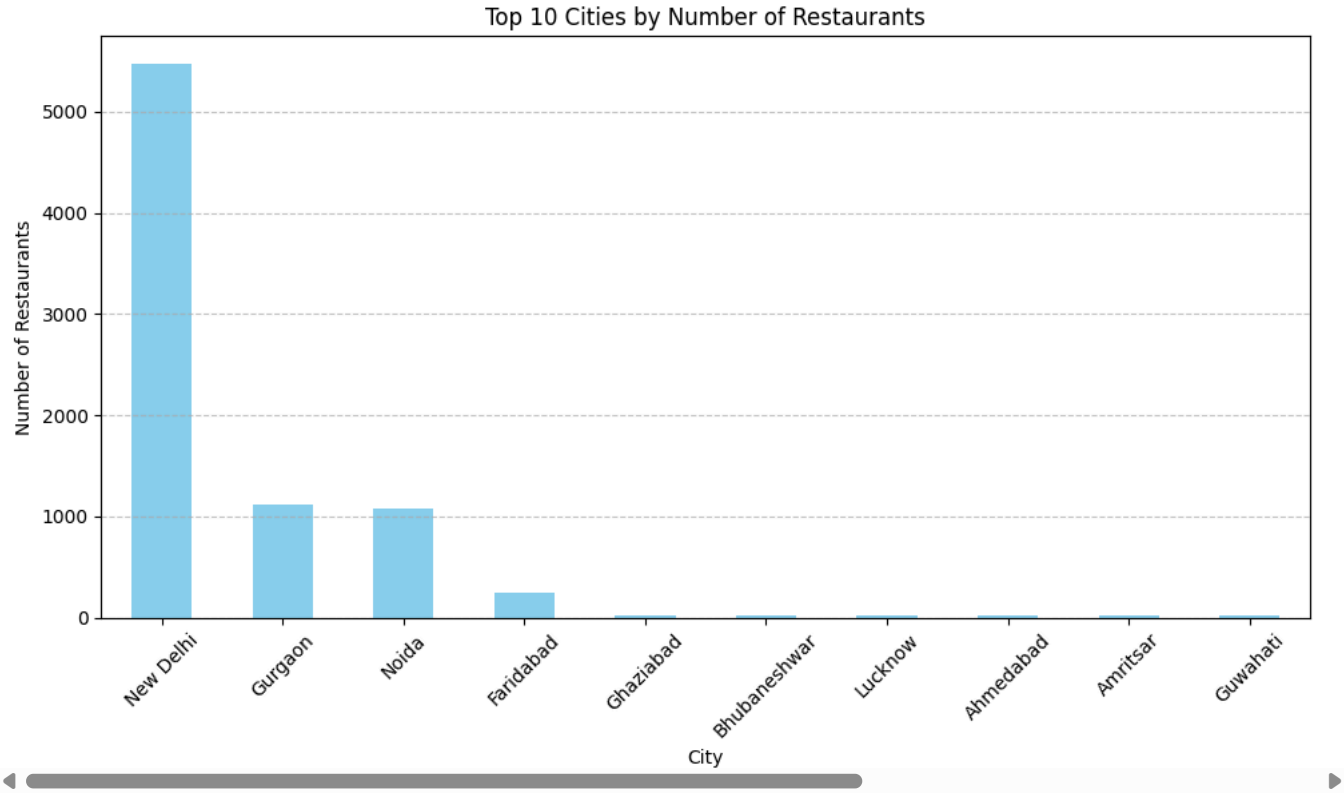
City	Aggregate rating
Inner City	4.900000
Quezon City	4.800000
Makati City	4.650000
Pasig City	4.633333
Mandaluyong City	4.625000
Beechworth	4.600000
London	4.535000
Taguig City	4.525000
Secunderabad	4.500000
Lincoln	4.500000

dtype: float64

```
# STEP 4: City with highest average rating
highest_rated_city = city_avg_rating.idxmax()
highest_rating = city_avg_rating.max()
print(f"City with the highest average rating: {highest_rated_city} ({highest_rating:.2f})")
```

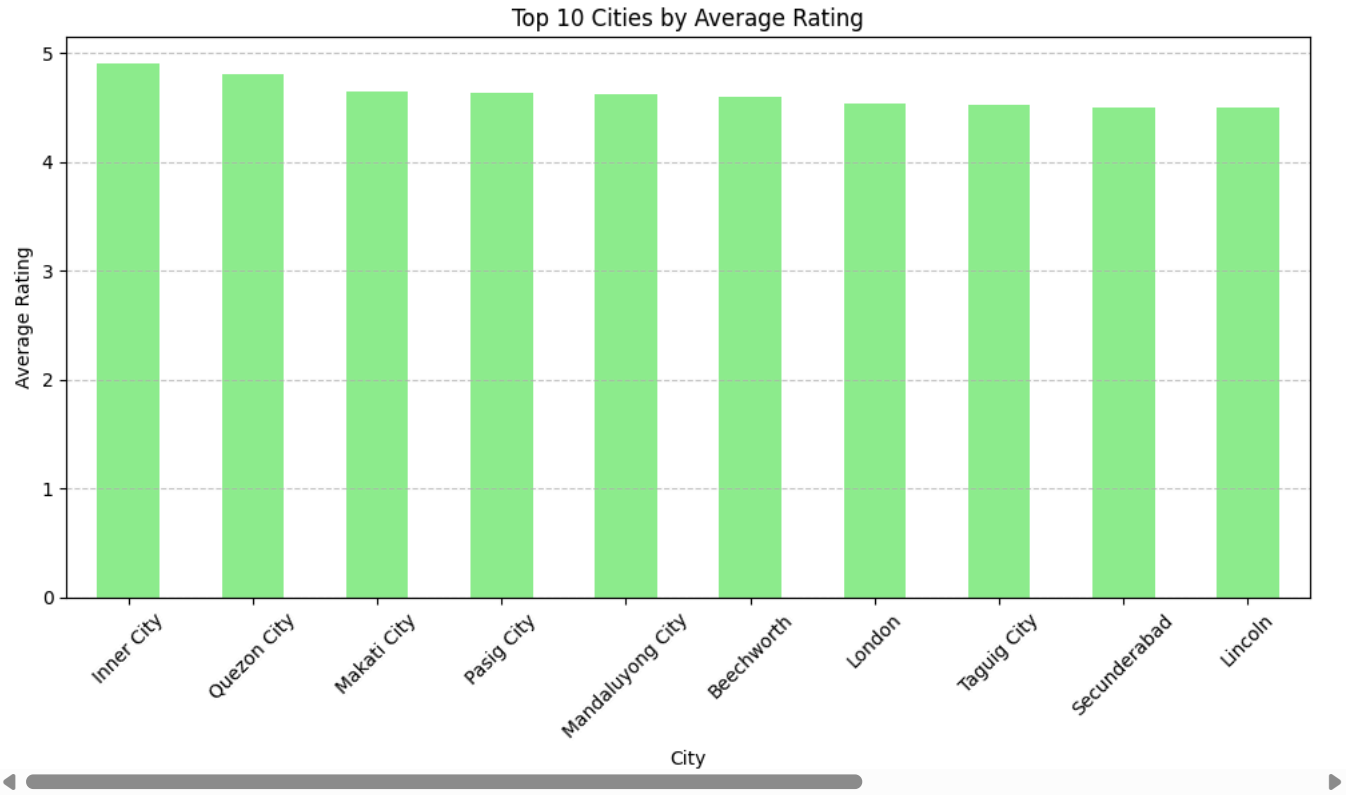
```
City with the highest average rating: Inner City (4.90)
```

```
# STEP 5: Visualization - Top 10 Cities by Number of Restaurants
import matplotlib.pyplot as plt
top_10_cities = city_counts.head(10)
plt.figure(figsize=(10, 6))
top_10_cities.plot(kind='bar', color='skyblue')
plt.title('Top 10 Cities by Number of Restaurants')
plt.xlabel('City')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```




```
# STEP 6: Visualization - Top 10 Cities by Average Rating
top_10_avg_rating = city_avg_rating.head(10)
plt.figure(figsize=(10, 6))
top_10_avg_rating.plot(kind='bar', color='lightgreen')
plt.title('Top 10 Cities by Average Rating')
```

```
plt.xlabel('City')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Task 3

```
# STEP 1: Check column and clean data
df['Price range'].isnull().sum()
df['Price range'] = pd.to_numeric(df['Price range'], errors='coerce')
df['Price range'].fillna(0, inplace=True)
```

```
/tmp/ipython-input-73-3638656352.py:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col

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

```
# STEP 2: Count restaurants by price range
price_counts = df['Price range'].value_counts().sort_index()
print("Number of restaurants per price range:")
print(price_counts)
```

```
Number of restaurants per price range:
Price range
1    4444
2    3113
3    1408
4     586
Name: count, dtype: int64
```

```
# STEP 3: Calculate percentage of restaurants in each price range
total_restaurants = len(df)
price_percentages = (price_counts / total_restaurants) * 100
price_percentages = price_percentages.round(2)
print("\nPercentage of restaurants in each price range:")
print(price_percentages)
```

```
Percentage of restaurants in each price range:
Price range
1    46.53
2    32.59
3    14.74
```

```
4     6.14
Name: count, dtype: float64
```

```
# STEP 4: Visualization - Bar Chart of Price Ranges
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 5))
price_counts.plot(kind='bar', color='coral')
plt.title("Distribution of Restaurants by Price Range")
plt.xlabel("Price Range")
plt.ylabel("Number of Restaurants")
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

Task 4

```
# STEP 1: Check values in 'Has Online delivery'
df['Has Online delivery'].value_counts(dropna=False)
```

Has Online delivery	count
No	7100
Yes	2451

dtype: int64

```
# STEP 2: Clean the data
# Convert all to consistent format (e.g., Yes/No)
df['Has Online delivery'] = df['Has Online delivery'].fillna('No')
df['Has Online delivery'] = df['Has Online delivery'].str.strip().str.capitalize()
```

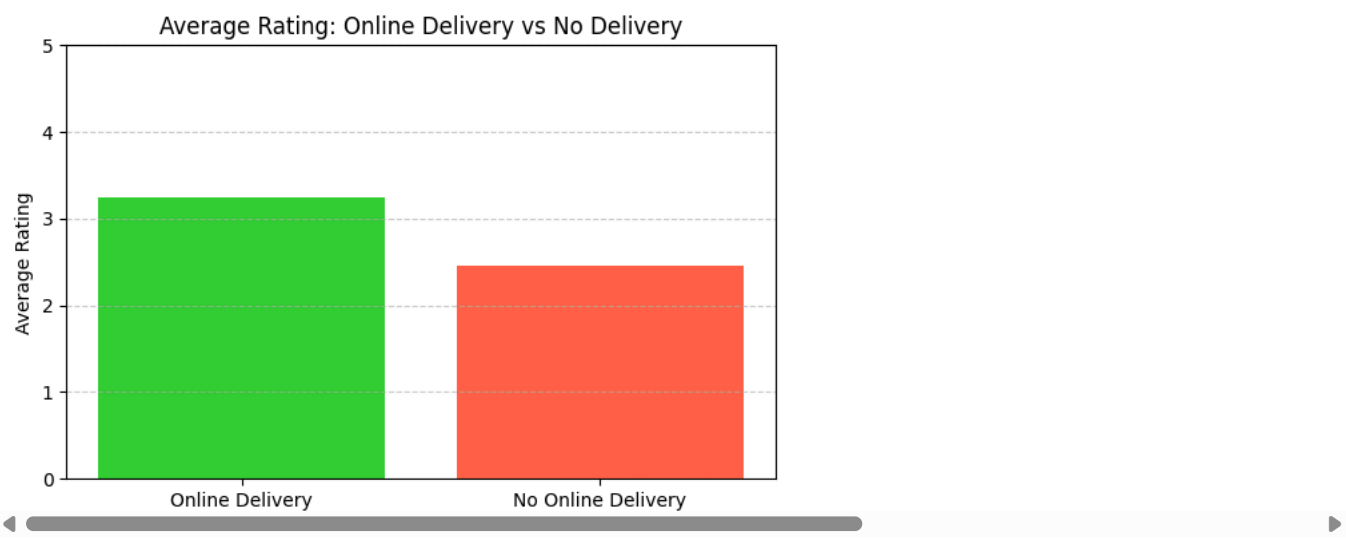
```
# STEP 3: Calculate percentage offering online delivery
total_restaurants = len(df)
offers_delivery = df[df['Has Online delivery'] == 'Yes']
delivery_count = len(offers_delivery)
percentage_delivery = (delivery_count / total_restaurants) * 100
print(f"Restaurants offering online delivery: {delivery_count} ({percentage_delivery:.2f}%)")
```

```
Restaurants offering online delivery: 2451 (25.66%)
```

```
# STEP 4: Compare average ratings (delivery vs non-delivery)
df['Aggregate rating'] = pd.to_numeric(df['Aggregate rating'], errors='coerce')
avg_rating_delivery = df[df['Has Online delivery'] == 'Yes']['Aggregate rating'].mean()
avg_rating_no_delivery = df[df['Has Online delivery'] == 'No']['Aggregate rating'].mean()
print(f"Average Rating (Online Delivery): {avg_rating_delivery:.2f}")
print(f"Average Rating (No Online Delivery): {avg_rating_no_delivery:.2f}")
```

```
Average Rating (Online Delivery): 3.25
Average Rating (No Online Delivery): 2.47
```

```
# STEP 5: Visualization - Average Rating Comparison
import matplotlib.pyplot as plt
ratings = [avg_rating_delivery, avg_rating_no_delivery]
labels = ['Online Delivery', 'No Online Delivery']
plt.figure(figsize=(6, 4))
plt.bar(labels, ratings, color=['limegreen', 'tomato'])
plt.title("Average Rating: Online Delivery vs No Delivery")
plt.ylabel("Average Rating")
plt.ylim(0, 5)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

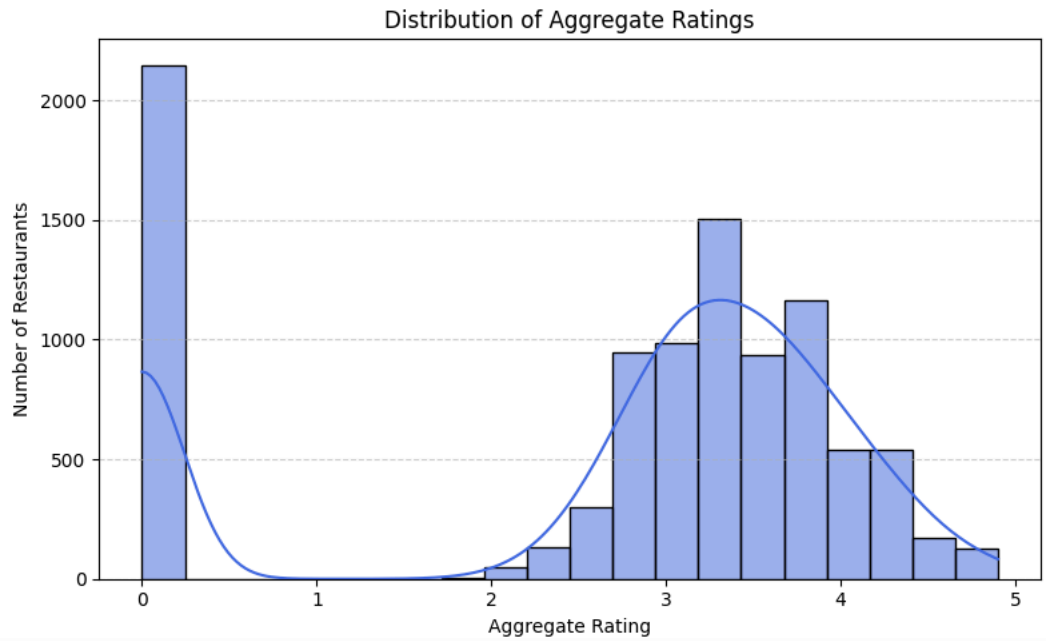


Level 2

Task 1

```
# STEP 1: Clean the necessary columns
df['Aggregate rating'] = pd.to_numeric(df['Aggregate rating'], errors='coerce')
df['Votes'] = pd.to_numeric(df['Votes'], errors='coerce').fillna(0)
```

```
# STEP 2: Rating Distribution
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 5))
sns.histplot(df['Aggregate rating'].dropna(), bins=20, kde=True, color='royalblue')
plt.title('Distribution of Aggregate Ratings')
plt.xlabel('Aggregate Rating')
plt.ylabel('Number of Restaurants')
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



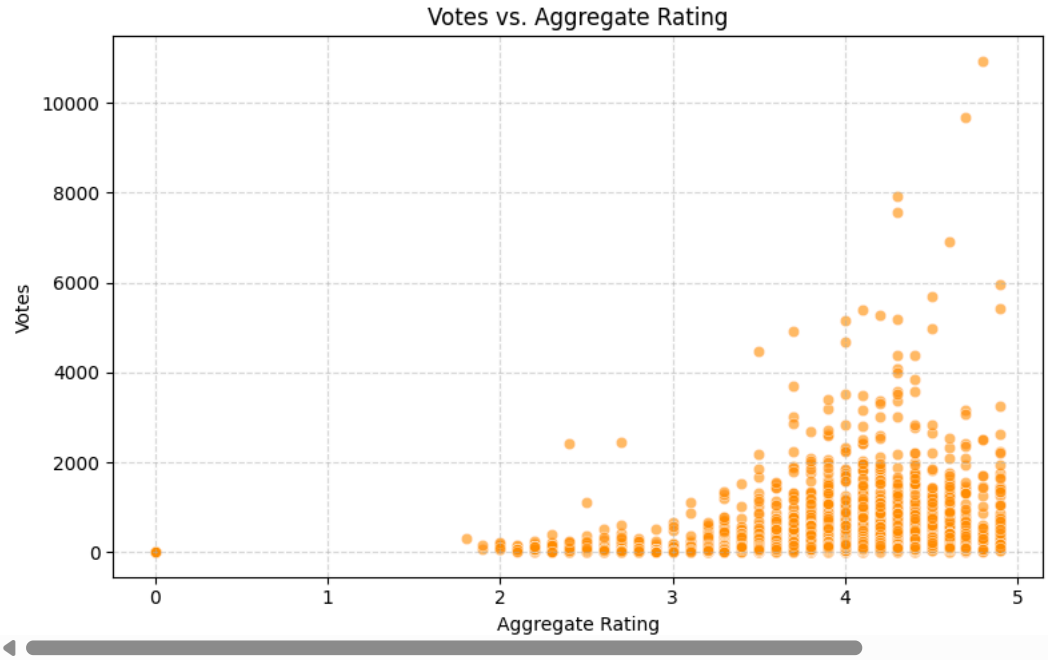

```
# STEP 3: Most Common Rating Range
rating_counts = df['Aggregate rating'].value_counts().sort_values(ascending=False)
most_common_rating = rating_counts.idxmax()
most_common_count = rating_counts.max()
print(f"Most common rating: {most_common_rating} ({most_common_count} restaurants)")
```

```
Most common rating: 0.0 (2148 restaurants)
```

```
# STEP 4: Average number of votes
avg_votes = df['Votes'].mean()
print(f"Average number of votes: {avg_votes:.2f}")
```

```
Average number of votes: 156.91
```

```
# Scatter plot of Rating vs Votes
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Aggregate rating', y='Votes', data=df, color='darkorange', alpha=0.6)
plt.title('Votes vs. Aggregate Rating')
plt.xlabel('Aggregate Rating')
plt.ylabel('Votes')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

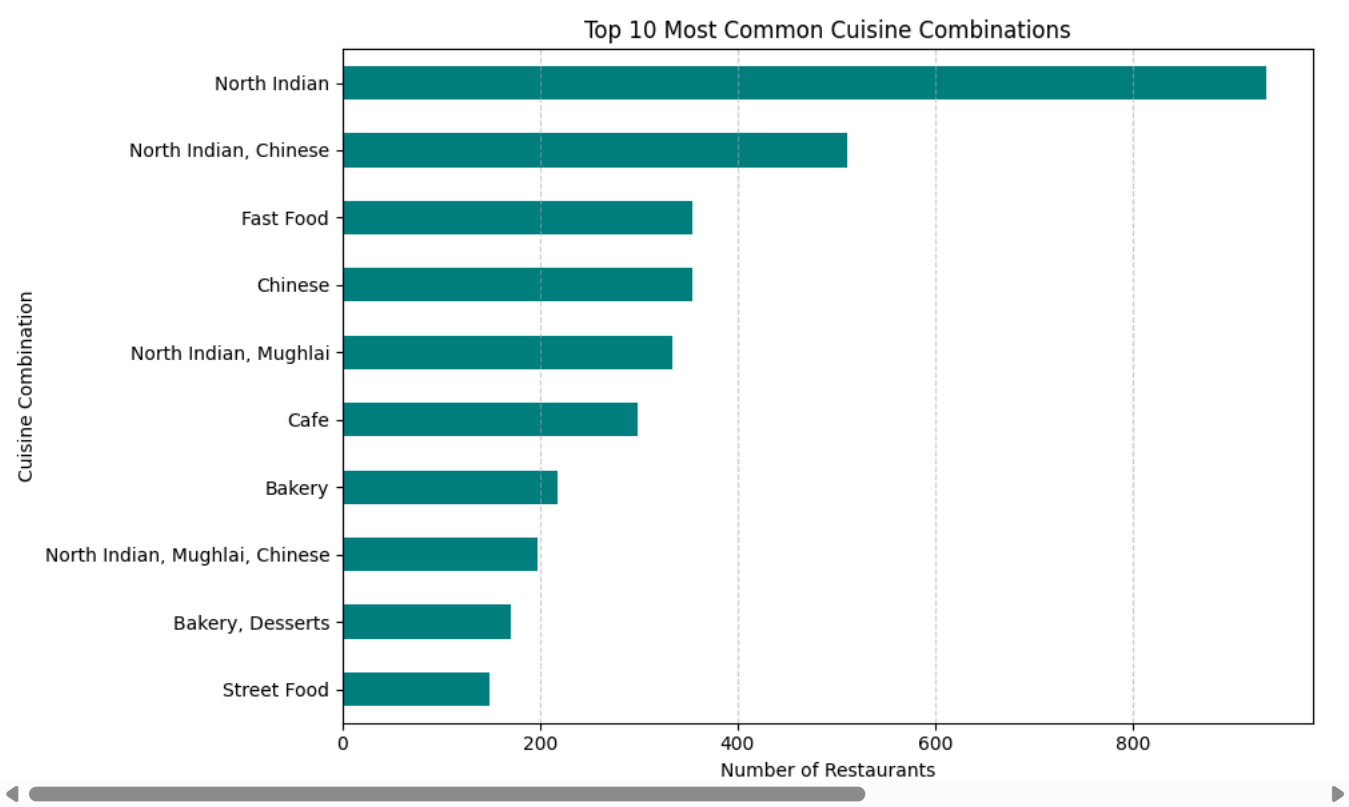


Task 2

```
# STEP 1: Handle missing values in Cuisines and convert to consistent format
df['Cuisines'] = df['Cuisines'].fillna('Unknown')
df['Cuisines'] = df['Cuisines'].str.strip()


# STEP 2: Count the most frequent combinations (as-is)
cuisine_combos = df['Cuisines'].value_counts().head(10)
print("Top 10 cuisine combinations:")
print(cuisine_combos)
```

```
Top 10 cuisine combinations:
Cuisines
North Indian                      936
North Indian, Chinese             511
Fast Food                         354
Chinese                           354
North Indian, Mughlai             334
Cafe                              299
Bakery                            218
North Indian, Mughlai, Chinese    197
Bakery, Desserts                  170
Street Food                       149
Name: count, dtype: int64
```

```
# STEP 3: Visualize the Top 10 combinations
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
cuisine_combos.plot(kind='barh', color='teal')
plt.gca().invert_yaxis()
plt.title('Top 10 Most Common Cuisine Combinations')
plt.xlabel('Number of Restaurants')
plt.ylabel('Cuisine Combination')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

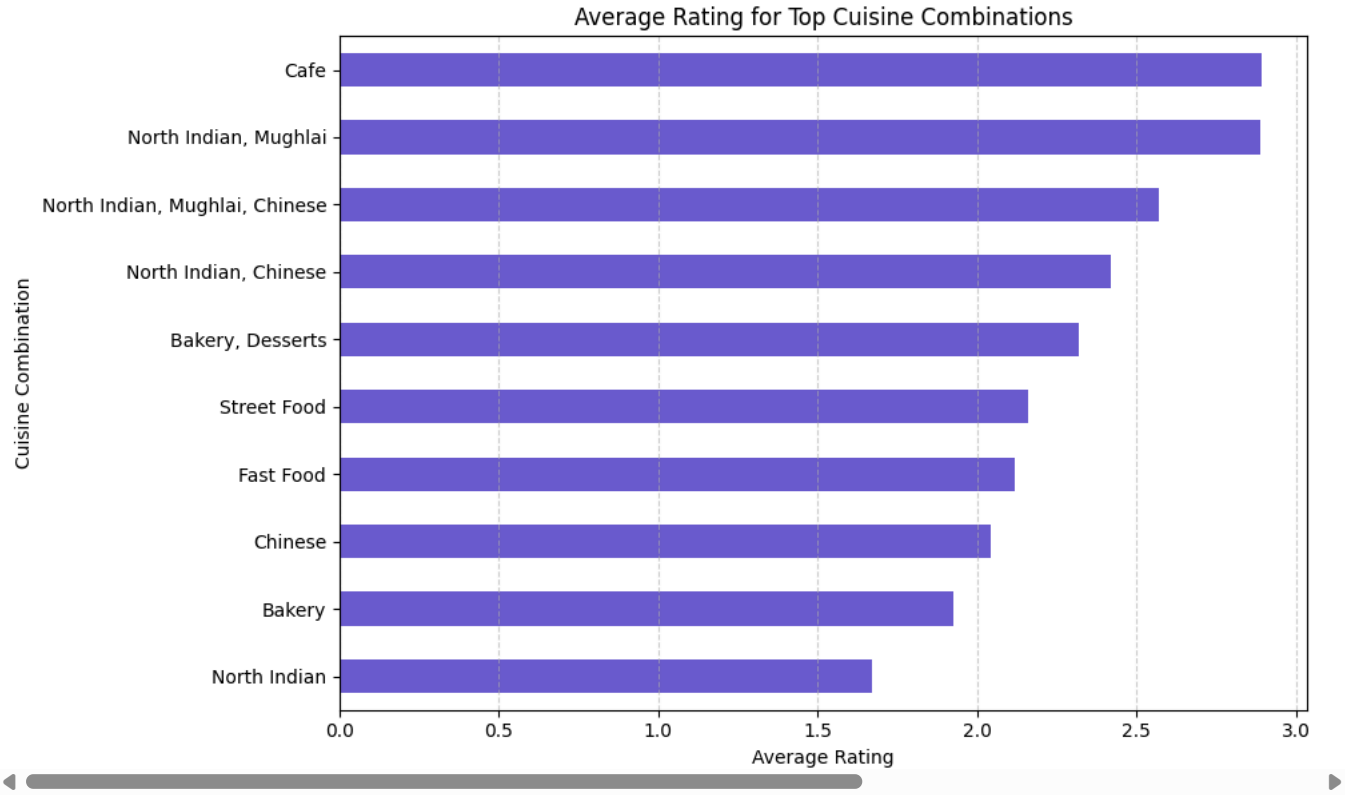



```
# STEP 4: Average rating by top cuisine combinations
top_combos = cuisine_combos.index.tolist()
combo_ratings = df[df['Cuisines'].isin(top_combos)].groupby('Cuisines')['Aggregate rating'].mean().sort_values(ascending=False)
print("\nAverage rating for top cuisine combinations:")
print(combo_ratings)
```

```
Average rating for top cuisine combinations:
Cuisines
Cafe                              2.890970
North Indian, Mughlai             2.888623
North Indian, Mughlai, Chinese    2.568528
North Indian, Chinese             2.421722
Bakery, Desserts                  2.317647
```

```
Street Food                    2.161745
Fast Food                      2.118362
Chinese                        2.042090
Bakery                         1.924312
North Indian                   1.672329
Name: Aggregate rating, dtype: float64
```

```
# STEP 5: Visualize rating comparison for top combos
plt.figure(figsize=(10, 6))
combo_ratings.plot(kind='barh', color='slateblue')
plt.gca().invert_yaxis()
plt.title('Average Rating for Top Cuisine Combinations')
plt.xlabel('Average Rating')
plt.ylabel('Cuisine Combination')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Task 3

```
# STEP 1: Install Folium
!pip install folium
```

```
Requirement already satisfied: folium in /usr/local/lib/python3.11/dist-packages (0.19.7)
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.11/dist-packages (from folium) (3.1.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from folium) (2.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.11/dist-packages (from folium) (2025.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2>=2.9->folium) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->folium) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->folium) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->folium) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->folium) (2025.6.15)
```

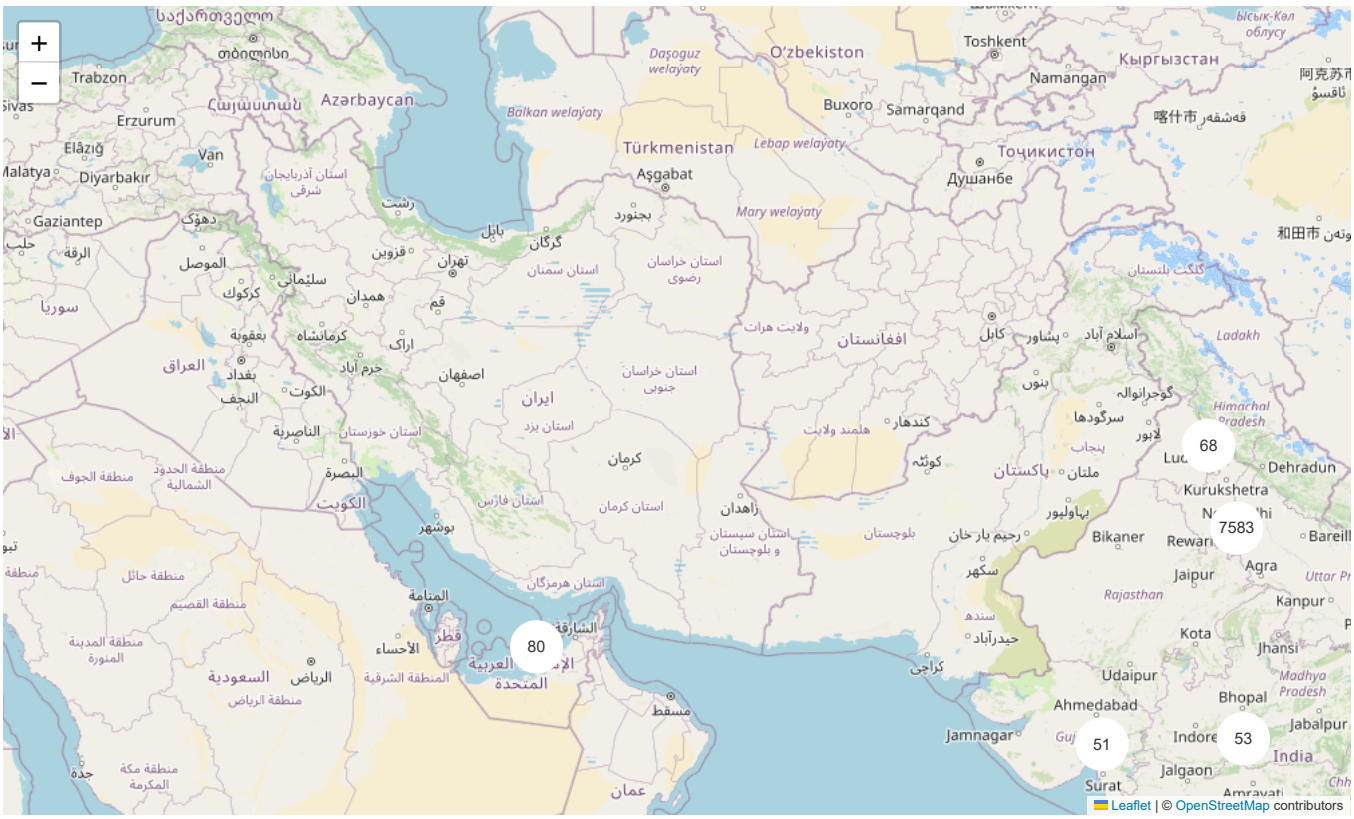
```
# STEP 2: Import libraries
import pandas as pd
import folium
from folium.plugins import MarkerCluster
# Load data (if not already loaded)
# df = pd.read_csv("Dataset .csv")
```

```
# STEP 3: Clean & prepare coordinates
df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce')
df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')
# Drop rows with missing or zero coordinates
geo_df = df.dropna(subset=['Longitude', 'Latitude'])
```

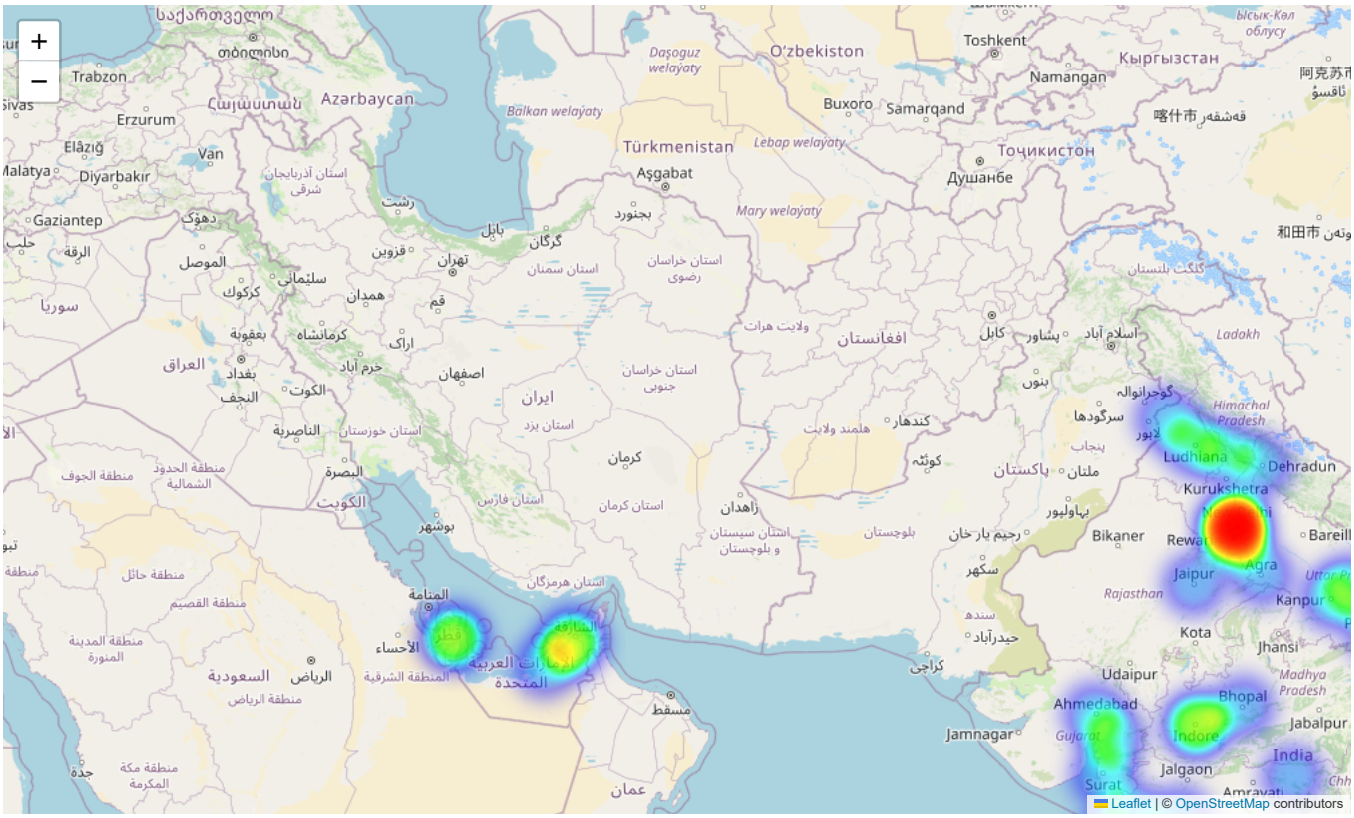
```
geo_df = geo_df[(geo_df['Longitude'] != 0) & (geo_df['Latitude'] != 0)]
print(f"Total valid restaurant coordinates: {len(geo_df)}")
```

```
Total valid restaurant coordinates: 9052
```

```
# STEP 4: Create an interactive map using Folium
# Center of map based on mean coordinates
map_center = [geo_df['Latitude'].mean(), geo_df['Longitude'].mean()]
restaurant_map = folium.Map(location=map_center, zoom_start=5)
# Add markers using clustering
marker_cluster = MarkerCluster().add_to(restaurant_map)
for idx, row in geo_df.iterrows():
    popup_text = f"{row['Restaurant Name']} ({row['City']})<br>Rating: {row['Aggregate rating']}"
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=popup_text
    ).add_to(marker_cluster)
restaurant_map
```




```
from folium.plugins import HeatMap
heatmap_data = geo_df[['Latitude', 'Longitude']].dropna().values.tolist()
restaurant_heatmap = folium.Map(location=map_center, zoom_start=5)
HeatMap(heatmap_data).add_to(restaurant_heatmap)
restaurant_heatmap
```



Task 4

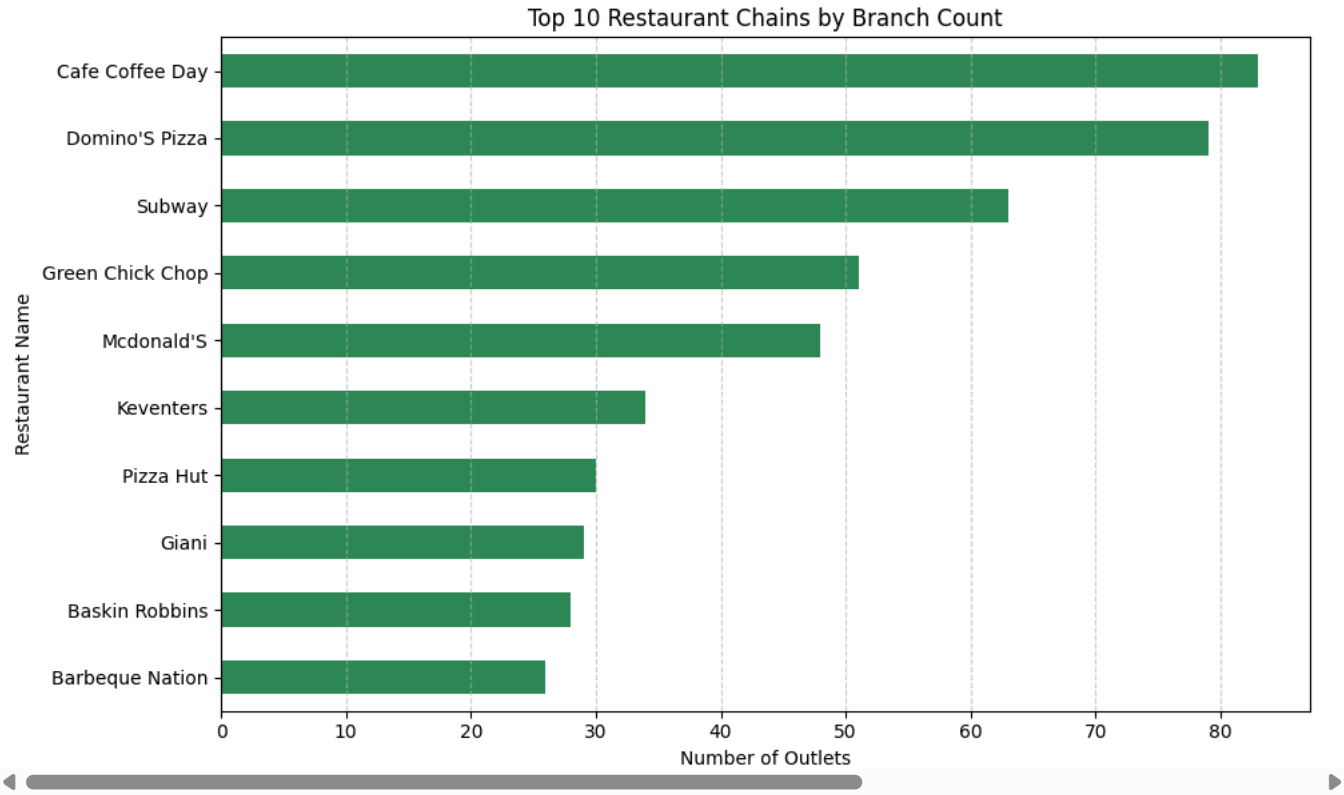
```
# STEP 1: Clean restaurant names
df['Restaurant Name'] = df['Restaurant Name'].str.strip().str.title()
# Count frequency of restaurant names
name_counts = df['Restaurant Name'].value_counts()
# Filter to show only chains (appearing more than once)
chains = name_counts[name_counts > 1]
print(f"Total unique restaurant chains found: {len(chains)}")
chains.head(10)
```

```
Total unique restaurant chains found: 742
```

Restaurant Name	count
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

dtype: int64

```
# STEP 2: Plot the Top 10 most common chains
import matplotlib.pyplot as plt
top_chains = chains.head(10)
plt.figure(figsize=(10, 6))
top_chains.plot(kind='barh', color='seagreen')
plt.gca().invert_yaxis()
plt.title('Top 10 Restaurant Chains by Branch Count')
plt.xlabel('Number of Outlets')
plt.ylabel('Restaurant Name')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

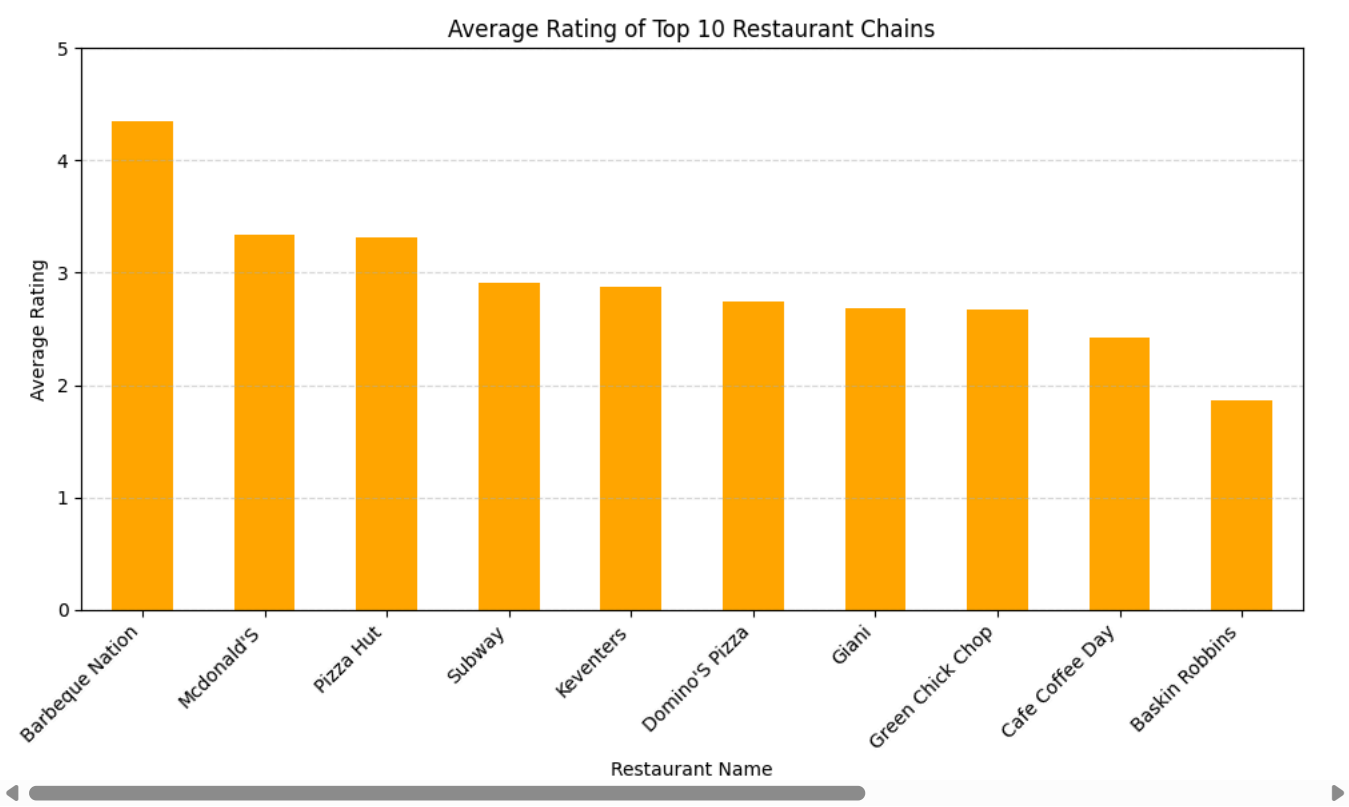



```
# STEP 3: Analyze ratings and votes for these chains
# Filter the main dataframe to include only chain restaurants
chain_df = df[df['Restaurant Name'].isin(chains.index)]
# Group by Restaurant Name
chain_stats = chain_df.groupby('Restaurant Name').agg({
    'Aggregate rating': 'mean',
    'Votes': 'mean',
    'Restaurant ID': 'count'
}).rename(columns={
    'Aggregate rating': 'Avg Rating',
    'Votes': 'Avg Votes',
    'Restaurant ID': 'Branch Count'
}).sort_values(by='Branch Count', ascending=False)
# Display top 10 chains with stats
chain_stats.head(10)
```

Restaurant Name	Avg Rating	Avg Votes	Branch Count
Cafe Coffee Day	2.419277	29.253012	83
Domino'S Pizza	2.740506	84.088608	79
Subway	2.907937	97.206349	63
Green Chick Chop	2.672549	18.901961	51
Mcdonald'S	3.339583	110.229167	48
Keventers	2.870588	37.147059	34
Pizza Hut	3.320000	165.366667	30
Giani	2.689655	29.448276	29
Baskin Robbins	1.860714	15.285714	28
Barbeque Nation	4.353846	1082.384615	26

Next steps: Generate code with chain_stats | View recommended plots | New interactive sheet

```
# STEP 4: Optional - Visualize Average Ratings for Top 10 Chains
top_rating_chains = chain_stats.head(10).sort_values(by='Avg Rating', ascending=False)
plt.figure(figsize=(10, 6))
top_rating_chains['Avg Rating'].plot(kind='bar', color='orange')
plt.title('Average Rating of Top 10 Restaurant Chains')
plt.ylabel('Average Rating')
plt.xticks(rotation=45, ha='right')
plt.ylim(0, 5)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



Level 3

Task 2

```
df.columns
```

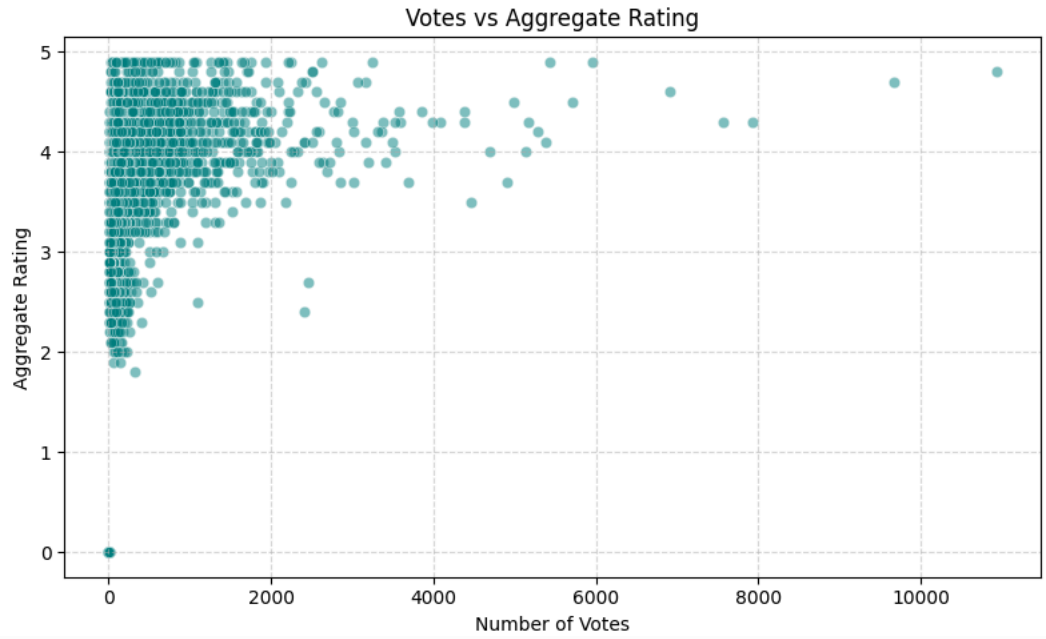
```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
   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

```
# STEP 1: Convert Votes and Ratings to numeric
df['Votes'] = pd.to_numeric(df['Votes'], errors='coerce').fillna(0)
df['Aggregate rating'] = pd.to_numeric(df['Aggregate rating'], errors='coerce').fillna(0)


# STEP 2: Identify highest and lowest vote counts
max_votes = df['Votes'].max()
min_votes = df['Votes'].min()
top_voted = df[df['Votes'] == max_votes]
least_voted = df[df['Votes'] == min_votes]
print("⏳ Restaurant(s) with the highest votes:")
print(top_voted[['Restaurant Name', 'City', 'Votes', 'Aggregate rating']])
print("\n🕳️ Restaurant(s) with the lowest votes:")
print(least_voted[['Restaurant Name', 'City', 'Votes', 'Aggregate rating']])
```

```
⏳ Restaurant(s) with the highest votes:
     Restaurant Name       City  Votes  Aggregate rating
728            Toit  Bangalore  10934               4.8
```

```
Restaurant(s) with the lowest votes:
                Restaurant Name        City  Votes  Aggregate rating
69              Cantinho Da Gula  S��o Paulo      0               0.0
874                The Chaiwalas   Faridabad      0               0.0
879           Fusion Food Corner   Faridabad      0               0.0
880                Punjabi Rasoi   Faridabad      0               0.0
887                Baskin Robbin   Faridabad      0               0.0
...                          ...         ...    ...               ...
9044               6 Packs Momos       Noida      0               0.0
9098                   Cafe' Wow       Noida      0               0.0
9099  Chef'S Basket Pop Up Caf��       Noida      0               0.0
9103           The Hangout-Deli       Noida      0               0.0
9111                   Platters       Noida      0               0.0

[1094 rows x 4 columns]
```

```
# STEP 3: Visualize Correlation - Votes vs Rating
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Votes', y='Aggregate rating', data=df, alpha=0.5, color='teal')
plt.title('Votes vs Aggregate Rating')
plt.xlabel('Number of Votes')
plt.ylabel('Aggregate Rating')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```




```
# STEP 4: Calculate Correlation Coefficient
correlation = df['Votes'].corr(df['Aggregate rating'])
print(f"📈 Correlation between Votes and Rating: {correlation:.3f}")
```

```
📈 Correlation between Votes and Rating: 0.314
```

Task 3

```
# STEP 1: Clean necessary columns
df['Price range'] = pd.to_numeric(df['Price range'], errors='coerce')
df['Has Online delivery'] = df['Has Online delivery'].fillna('No').str.strip().str.capitalize()
df['Has Table booking'] = df['Has Table booking'].fillna('No').str.strip().str.capitalize()
```

```
# STEP 2: Group by Price Range and calculate service availability percentages
grouped = df.groupby('Price range').agg({
    'Has Online delivery': lambda x: (x == 'Yes').sum(),
    'Has Table booking': lambda x: (x == 'Yes').sum(),
    'Restaurant ID': 'count'
}).rename(columns={
    'Has Online delivery': 'Online Delivery Count',
    'Has Table booking': 'Table Booking Count',
    'Restaurant ID': 'Total Restaurants'
})
grouped['% Online Delivery'] = (grouped['Online Delivery Count'] / grouped['Total Restaurants']) * 10
grouped['% Table Booking'] = (grouped['Table Booking Count'] / grouped['Total Restaurants']) * 100
```

```
grouped = grouped[['% Online Delivery', '% Table Booking']]
```

Price range	% Online Delivery	% Table Booking
1	15.774077	0.022502
2	41.310633	7.677482
3	29.190341	45.738636
4	9.044369	46.757679

Next steps: Generate code with grouped | View recommended plots | New interactive sheet

```
# STEP 3: Visualize the results
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
grouped.plot(kind='bar', figsize=(10, 6), color=['darkcyan', 'gold'])
plt.title('Service Availability by Price Range')
plt.xlabel('Price Range')
plt.ylabel('Percentage of Restaurants Offering Service')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
```