kaggle data analysis

NAME : NEERAJ G PAREKH
CLASS: ET2
BATCH: ET2 22
PRN: 202401070163
ROLL NO: ET2-32

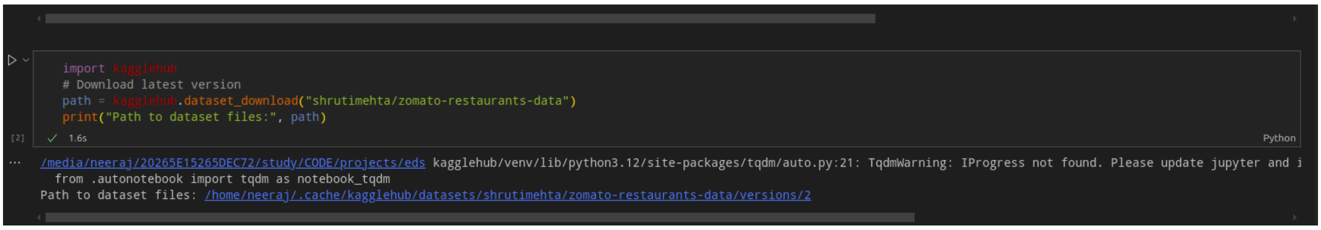
- **Link:** Zomato Restaurants Data on Kaggle

DOWNLOAD KAGGLE DATA VIA IMPORTING THROUGH KAGGLEHUB

How I Downloaded the Data

- The dataset was downloaded using the `kagglehub` library with the command:

```
path = kagglehub.dataset_download("shrutimehta/zomato-restaurants-data")
```

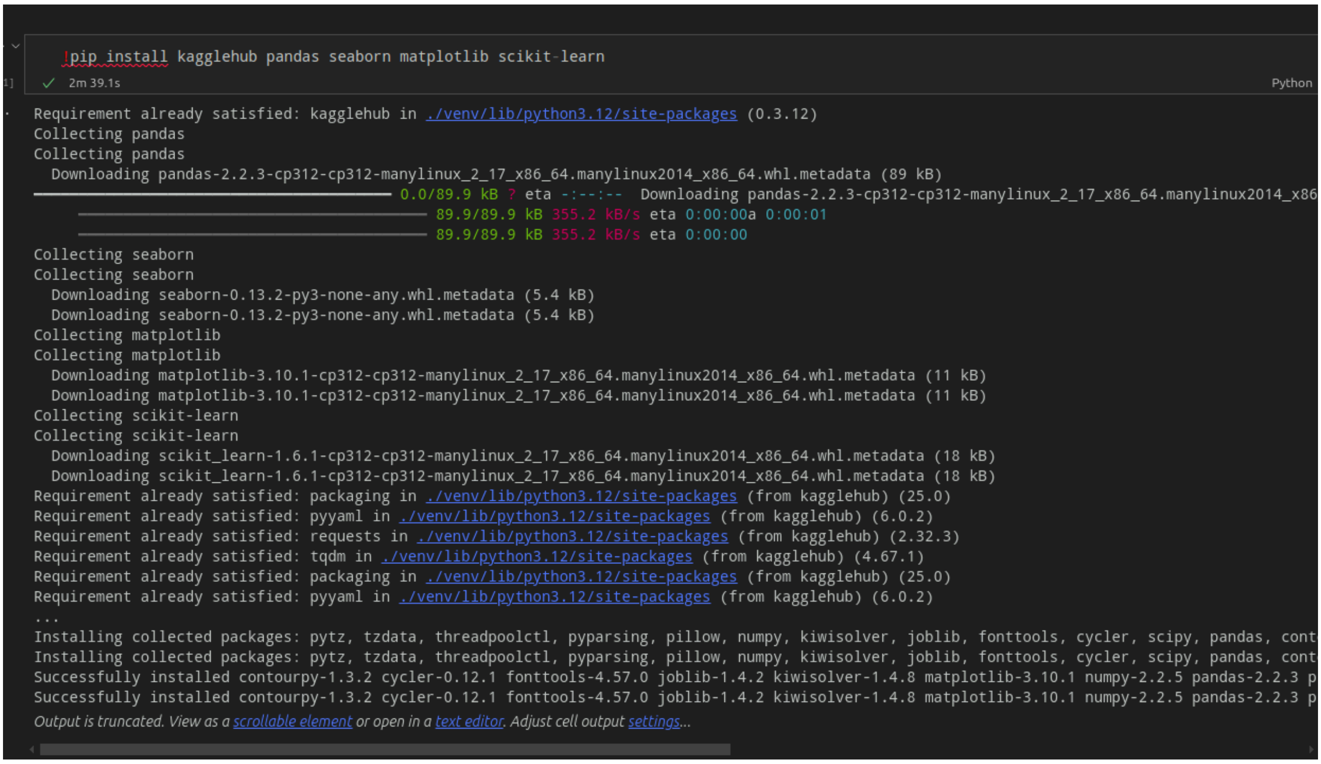


```
import kagglehub
# Download latest version
path = kagglehub.dataset_download("shrutimehta/zomato-restaurants-data")
print("Path to dataset files:", path)
```

[2] ✓ 1.6s Python

```
/media/neeraj/20265E15265DEC72/study/CODE/projects/eds kagglehub/venv/lib/python3.12/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and i
  from .autonotebook import tqdm as notebook_tqdm
Path to dataset files: /home/neeraj/.cache/kagglehub/datasets/shrutimehta/zomato-restaurants-data/versions/2
```

DOWNLOAD REQUIRED RESOURCE



```
!pip install kagglehub pandas seaborn matplotlib scikit-learn
```

```
Requirement already satisfied: kagglehub in ./venv/lib/python3.12/site-packages (0.3.12)
Collecting pandas
Collecting pandas
  Downloading pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (89 kB)
──────────────────────────────────────── 0.0/89.9 kB ? eta -:--:--  Downloading pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86
     ──────────────────────────────────────── 89.9/89.9 kB 355.2 kB/s eta 0:00:00a 0:00:01
     ──────────────────────────────────────── 89.9/89.9 kB 355.2 kB/s eta 0:00:00
Collecting seaborn
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Collecting matplotlib
Collecting matplotlib
  Downloading matplotlib-3.10.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
  Downloading matplotlib-3.10.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-1.6.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
  Downloading scikit_learn-1.6.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Requirement already satisfied: packaging in ./venv/lib/python3.12/site-packages (from kagglehub) (25.0)
Requirement already satisfied: pyyaml in ./venv/lib/python3.12/site-packages (from kagglehub) (6.0.2)
Requirement already satisfied: requests in ./venv/lib/python3.12/site-packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in ./venv/lib/python3.12/site-packages (from kagglehub) (4.67.1)
Requirement already satisfied: packaging in ./venv/lib/python3.12/site-packages (from kagglehub) (25.0)
Requirement already satisfied: pyyaml in ./venv/lib/python3.12/site-packages (from kagglehub) (6.0.2)
...
Installing collected packages: pytz, tzdata, threadpoolctl, pyparsing, pillow, numpy, kiwisolver, joblib, fonttools, cycler, scipy, pandas, cont
Installing collected packages: pytz, tzdata, threadpoolctl, pyparsing, pillow, numpy, kiwisolver, joblib, fonttools, cycler, scipy, pandas, cont
Successfully installed contourpy-1.3.2 cycler-0.12.1 fonttools-4.57.0 joblib-1.4.2 kiwisolver-1.4.8 matplotlib-3.10.1 numpy-2.2.5 pandas-2.2.3 p
Successfully installed contourpy-1.3.2 cycler-0.12.1 fonttools-4.57.0 joblib-1.4.2 kiwisolver-1.4.8 matplotlib-3.10.1 numpy-2.2.5 pandas-2.2.3 p
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

USE OF EACH LIBRARY :

Pandas

- **Data Loading and Inspection**
- **Data Cleaning**
- **Data Transformation** Convert data types and create new columns using operations such as `astype()` and `apply()`.
- **Aggregation and Grouping:** Calculate statistics such as mean, sum, and count for grouped data using `groupby()`.
- **Data Filtering and Sorting:** Filter rows based on conditions and sort data with `query()` and `sort_values()`.

NumPy

- **Numerical Operations:** Perform arithmetic and transformations on data arrays, such as normalizing data with min-max scaling.
- **Array Manipulation:** Use NumPy arrays for efficient storage and manipulation of numerical data, enhancing performance in large datasets.

Seaborn

- **Data Visualization:**

- **Correlation Heatmaps:** generate heatmaps for visualizing correlation matrices

Matplotlib

- **Plot Customization:** Customize plots and adjusting elements like titles, labels, and legends using Matplotlib's functions.
- **Data Plotting:** Generate basic plots, such as histograms and line charts, to explore data distributions and trends.

Scikit-learn

- **Data Preprocessing:** split data into training and test sets, preparing it for machine learning models.
- **Feature Engineering:** Use in label encoding to transform categorical variables into numerical formats suitable for model input.

Kagglehub

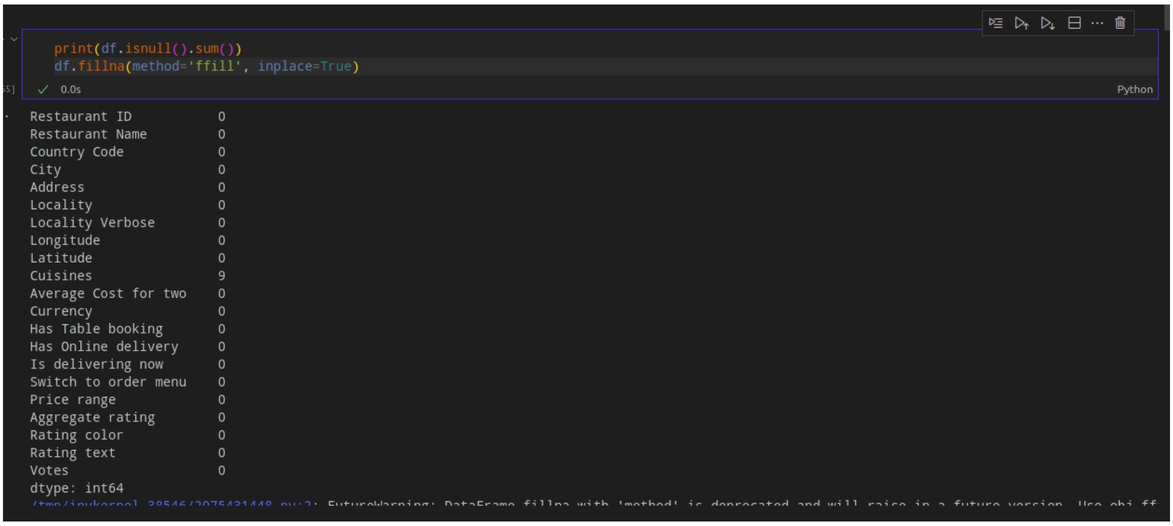
- **Data Access and Management:** While not directly used in data analysis, Kagglehub can be utilized to easily access and manage datasets from Kaggle within your environment, streamlining the workflow for data projects.

Openpyxl

- Used for reading Excel files (Here used for mapping country codes).

Questions/Problem Statements Explored

1. Identifying and filling missing values

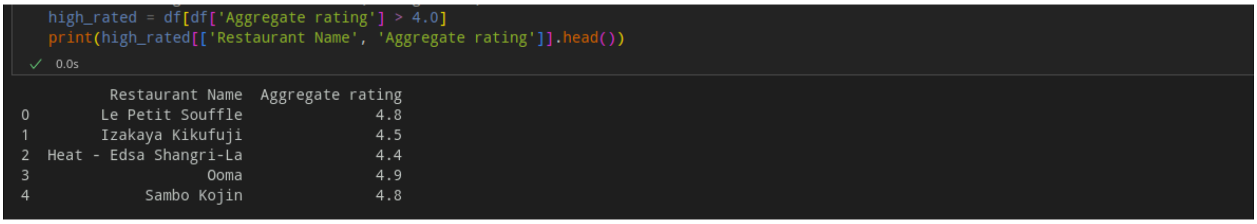
1. 

```
print(df.isnull().sum())
df.fillna(method='ffill', inplace=True)
```
✓ 0.0s

Python

```
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
/tmp/ipykernel_38546/2075431448.py:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ff
```

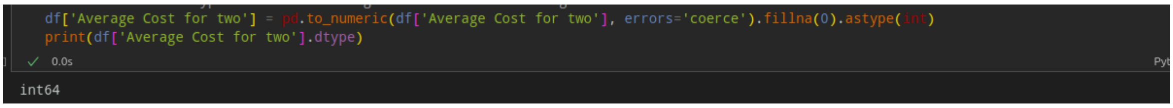
2. Filtering high-rated restaurants with rating above 4



```
high_rated = df[df['Aggregate rating'] > 4.0]
print(high_rated[['Restaurant Name', 'Aggregate rating']].head())
```
✓ 0.0s

```
         Restaurant Name  Aggregate rating
0        Le Petit Souffle               4.8
1        Izakaya Kikufuji               4.5
2  Heat - Edsa Shangri-La               4.4
3                    Ooma               4.9
4             Sambo Kojin               4.8
```

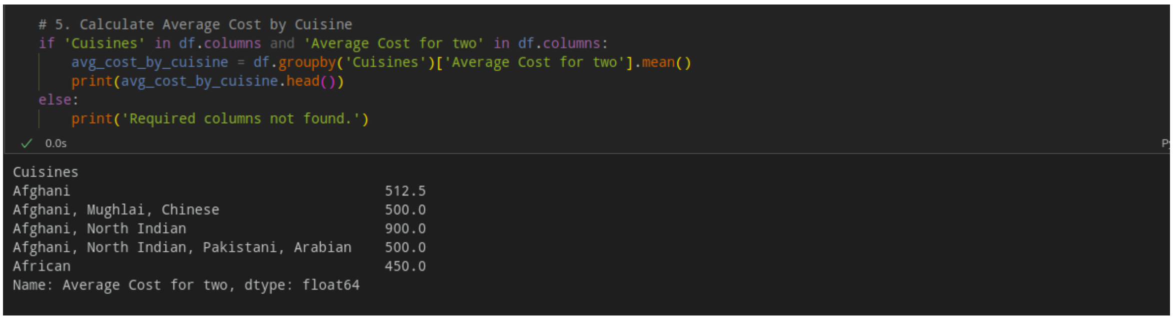
3. Data type conversions

1. 

```
df['Average Cost for two'] = pd.to_numeric(df['Average Cost for two'], errors='coerce').fillna(0).astype(int)
print(df['Average Cost for two'].dtype)
```
✓ 0.0s

```
int64
```

4. Calculating average cost by cuisine

1. 

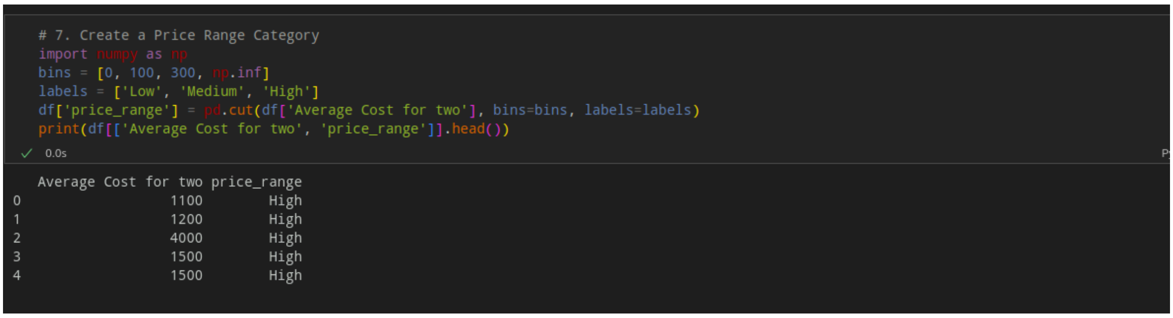
```
# 5. Calculate Average Cost by Cuisine
if 'Cuisines' in df.columns and 'Average Cost for two' in df.columns:
    avg_cost_by_cuisine = df.groupby('Cuisines')['Average Cost for two'].mean()
    print(avg_cost_by_cuisine.head())
else:
    print('Required columns not found.')
```
✓ 0.0s

```
Cuisines
Afghani                                      512.5
Afghani, Mughlai, Chinese                    500.0
Afghani, North Indian                        900.0
Afghani, North Indian, Pakistani, Arabian    500.0
African                                      450.0
Name: Average Cost for two, dtype: float64
```

5. Finding top expensive restaurants

1.

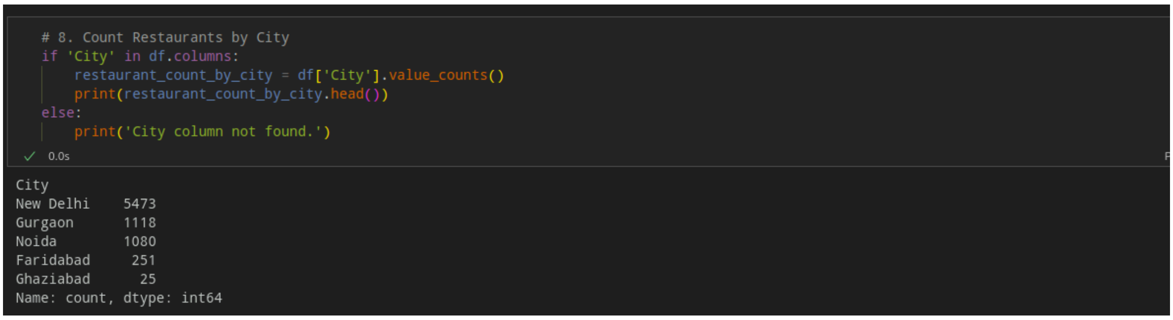
6. Creating price range categories

1. 

```
# 7. Create a Price Range Category
import numpy as np
bins = [0, 100, 300, np.inf]
labels = ['Low', 'Medium', 'High']
df['price_range'] = pd.cut(df['Average Cost for two'], bins=bins, labels=labels)
print(df[['Average Cost for two', 'price_range']].head())
```
✓ 0.0s

```
   Average Cost for two price_range
0                  1100        High
1                  1200        High
2                  4000        High
3                  1500        High
4                  1500        High
```

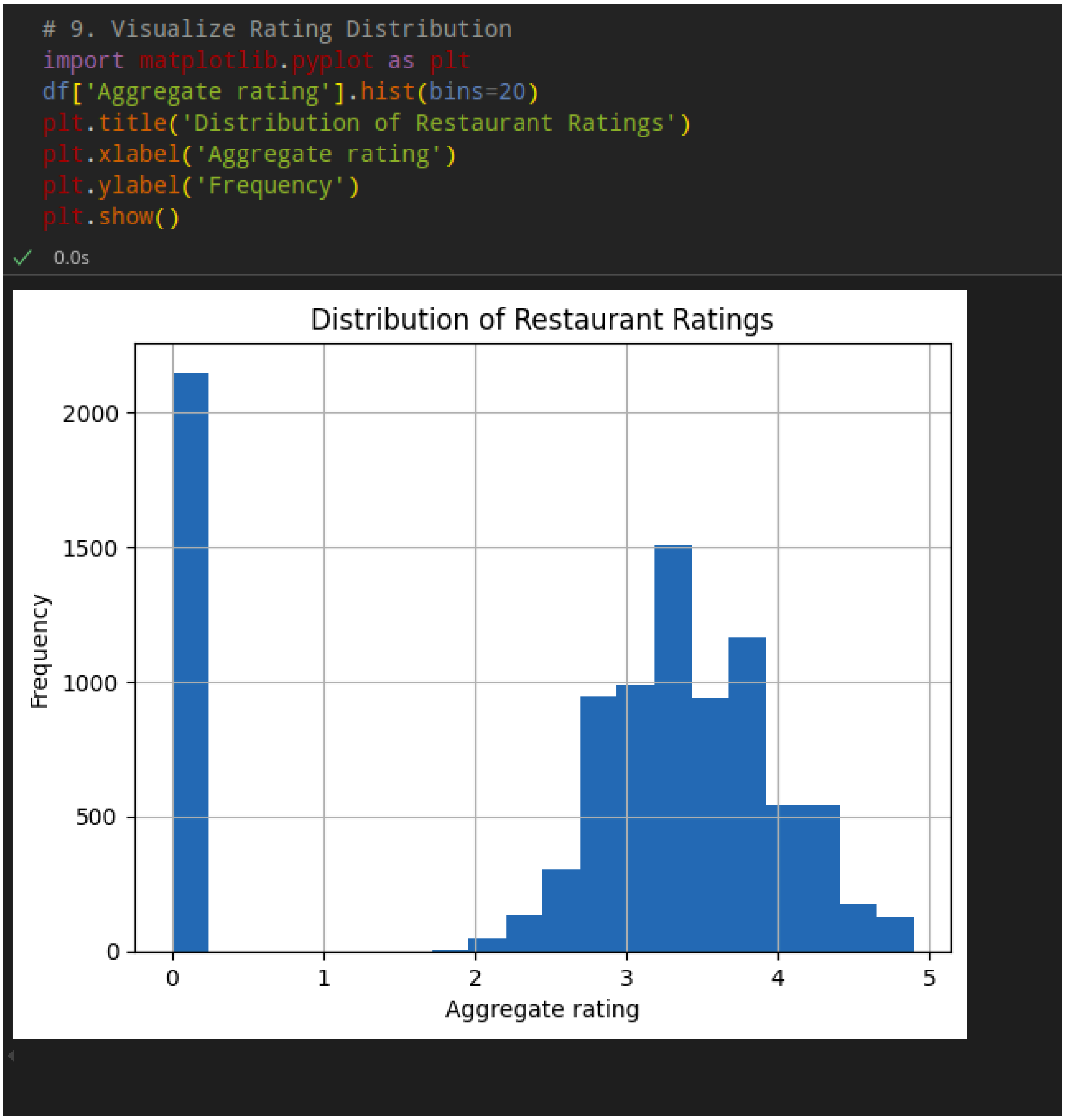
7. Counting restaurants by city

1. 

```
# 8. Count Restaurants by City
if 'City' in df.columns:
    restaurant_count_by_city = df['City'].value_counts()
    print(restaurant_count_by_city.head())
else:
    print('City column not found.')
```

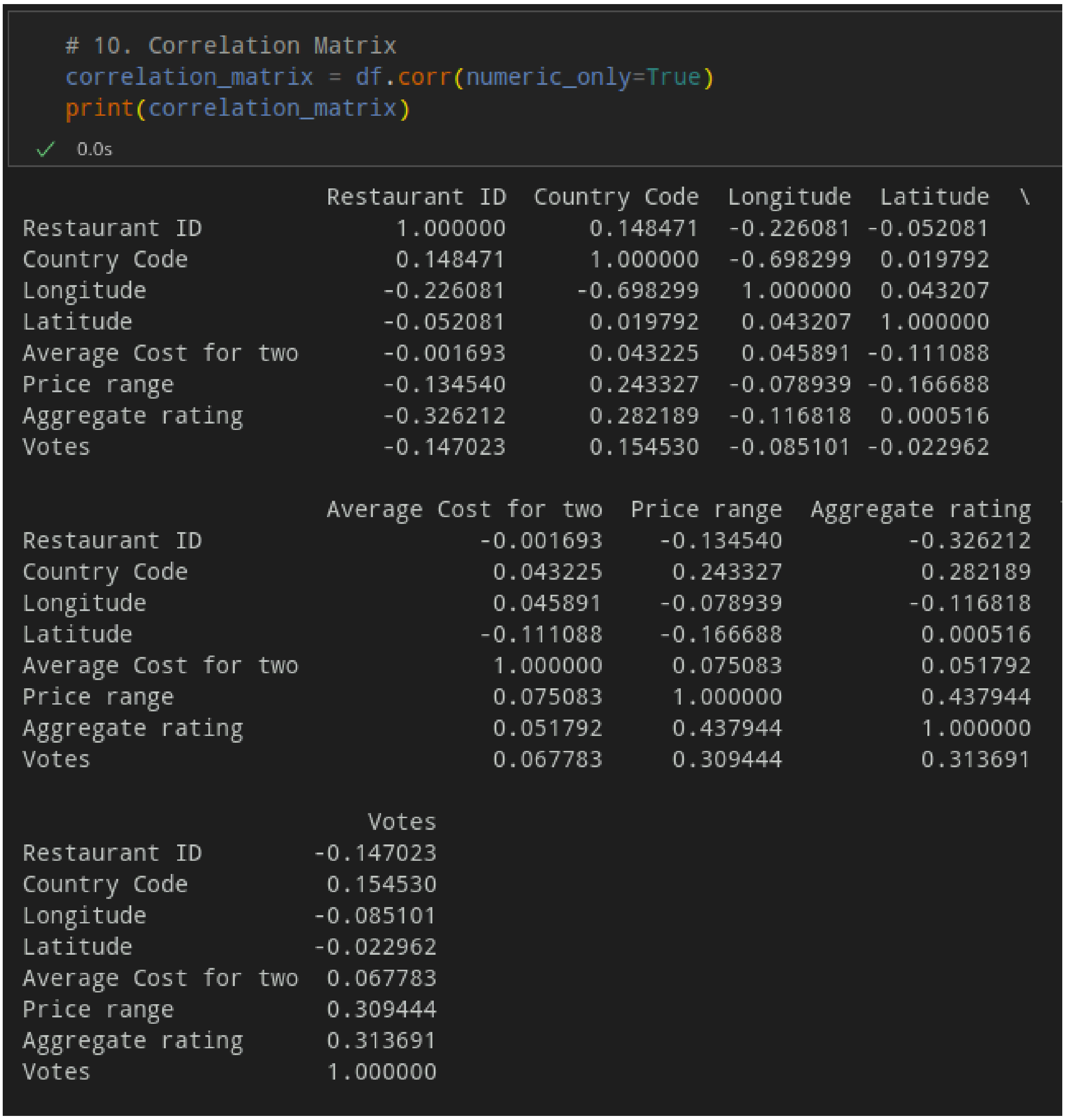
```
City
New Delhi     5473
Gurgaon       1118
Noida         1080
Faridabad      251
Ghaziabad       25
Name: count, dtype: int64
```

8. Visualizing rating distributions

1. 

```
# 9. Visualize Rating Distribution
import matplotlib.pyplot as plt
df['Aggregate rating'].hist(bins=20)
plt.title('Distribution of Restaurant Ratings')
plt.xlabel('Aggregate rating')
plt.ylabel('Frequency')
plt.show()
```

9. Correlation analysis

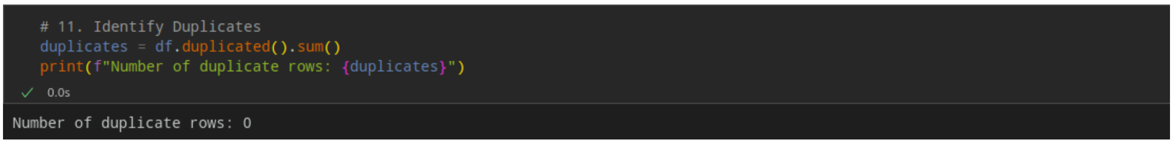
1. 

```
# 10. Correlation Matrix
correlation_matrix = df.corr(numeric_only=True)
print(correlation_matrix)
```

✓ 0.0s

```
                    Restaurant ID  Country Code  Longitude  Latitude  \
Restaurant ID            1.000000      0.148471  -0.226081 -0.052081   
Country Code             0.148471      1.000000  -0.698299  0.019792   
Longitude               -0.226081     -0.698299   1.000000  0.043207   
Latitude                -0.052081      0.019792   0.043207  1.000000   
Average Cost for two    -0.001693      0.043225   0.045891 -0.111088   
Price range             -0.134540      0.243327  -0.078939 -0.166688   
Aggregate rating        -0.326212      0.282189  -0.116818  0.000516   
Votes                   -0.147023      0.154530  -0.085101 -0.022962   

                    Average Cost for two  Price range  Aggregate rating  
Restaurant ID                  -0.001693    -0.134540         -0.326212   
Country Code                    0.043225     0.243327          0.282189   
Longitude                       0.045891    -0.078939         -0.116818   
Latitude                       -0.111088    -0.166688          0.000516   
Average Cost for two            1.000000     0.075083          0.051792   
Price range                     0.075083     1.000000          0.437944   
Aggregate rating                0.051792     0.437944          1.000000   
Votes                           0.067783     0.309444          0.313691   

                         Votes  
Restaurant ID        -0.147023  
Country Code          0.154530  
Longitude            -0.085101  
Latitude             -0.022962  
Average Cost for two  0.067783  
Price range           0.309444  
Aggregate rating      0.313691  
Votes                 1.000000  
```

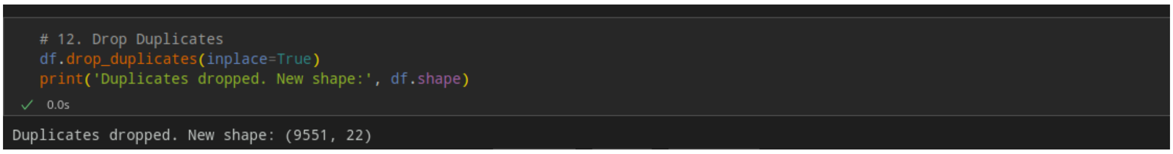
10. Identifying and removing duplicates

1. 

```
# 11. Identify Duplicates
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

✓ 0.0s

```
Number of duplicate rows: 0
```

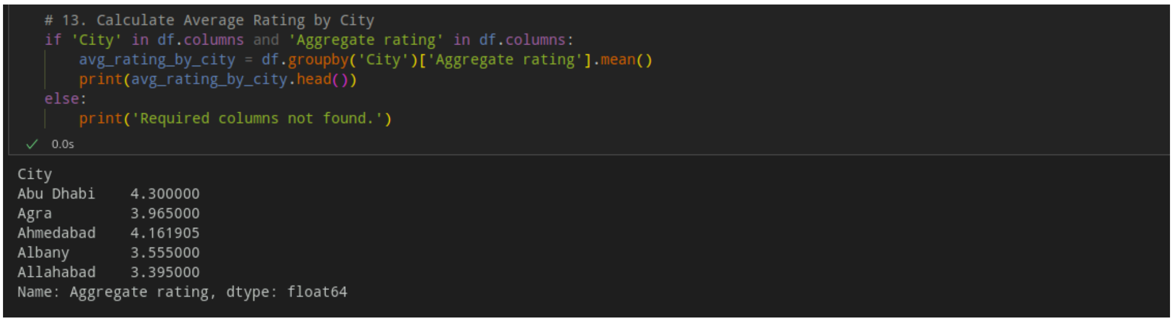
2. 

```
# 12. Drop Duplicates
df.drop_duplicates(inplace=True)
print('Duplicates dropped. New shape:', df.shape)
```

✓ 0.0s

```
Duplicates dropped. New shape: (9551, 22)
```

11. Calculating average rating by city

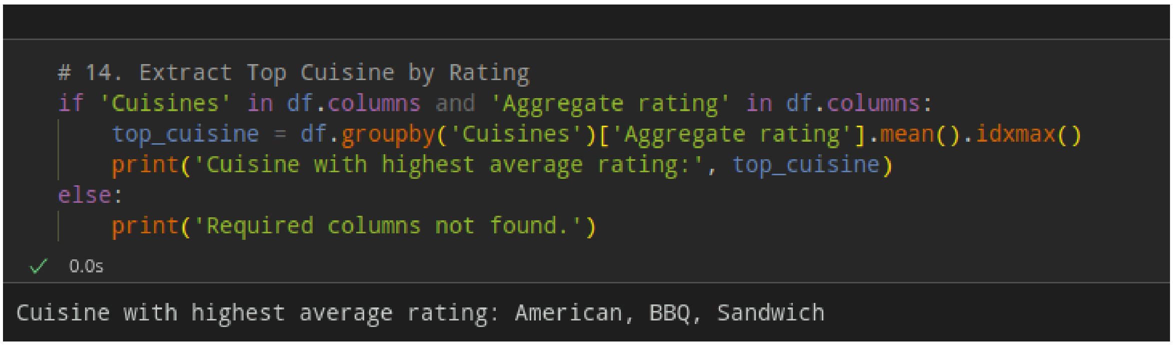
1. 

```
# 13. Calculate Average Rating by City
if 'City' in df.columns and 'Aggregate rating' in df.columns:
    avg_rating_by_city = df.groupby('City')['Aggregate rating'].mean()
    print(avg_rating_by_city.head())
else:
    print('Required columns not found.')
```

✓ 0.0s

```
City
Abu Dhabi    4.300000
Agra         3.965000
Ahmedabad    4.161905
Albany       3.555000
Allahabad    3.395000
Name: Aggregate rating, dtype: float64
```

12. Extracting top cuisine by rating

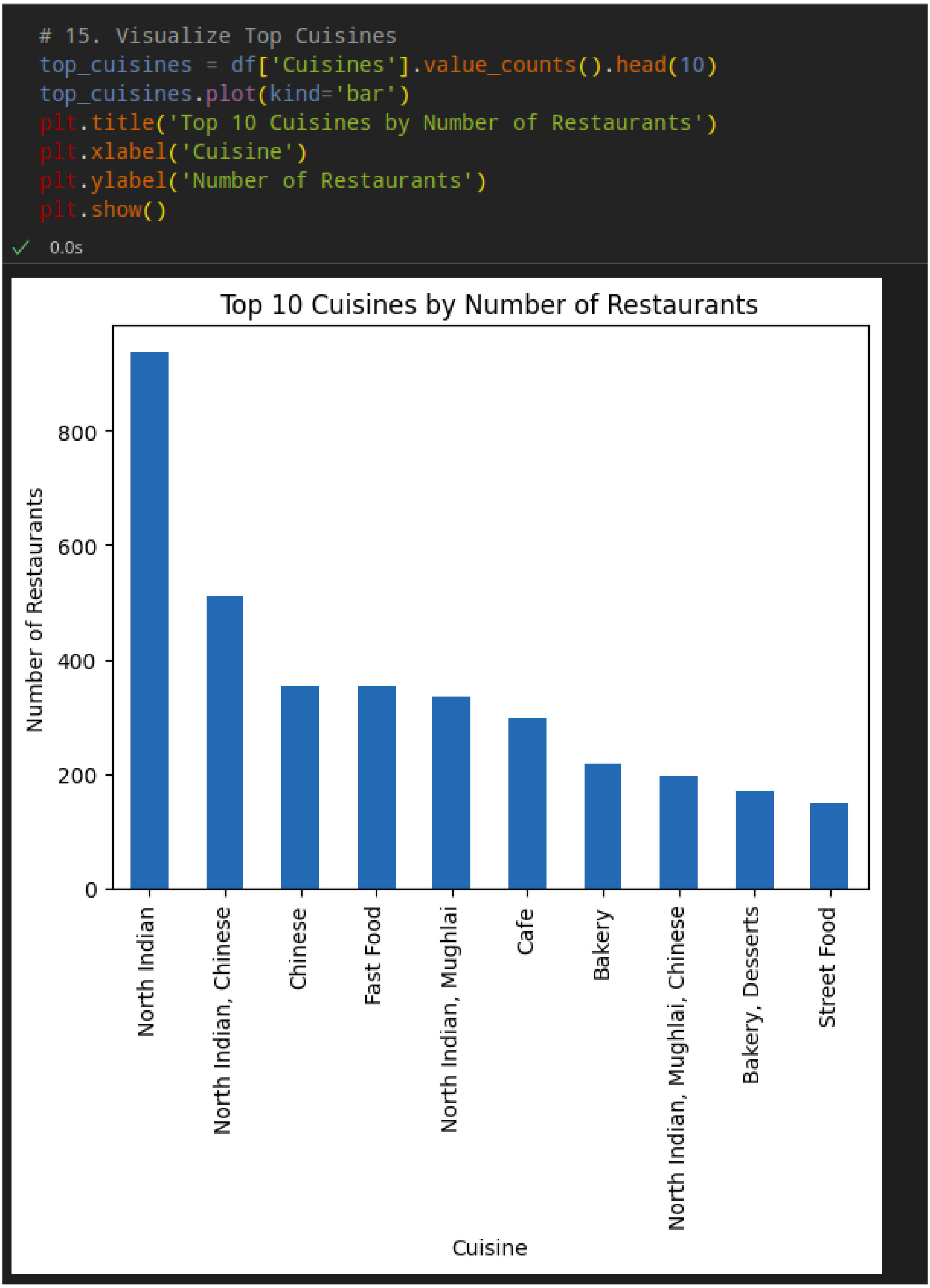
1. 

```
# 14. Extract Top Cuisine by Rating
if 'Cuisines' in df.columns and 'Aggregate rating' in df.columns:
    top_cuisine = df.groupby('Cuisines')['Aggregate rating'].mean().idxmax()
    print('Cuisine with highest average rating:', top_cuisine)
else:
    print('Required columns not found.')
```

✓ 0.0s

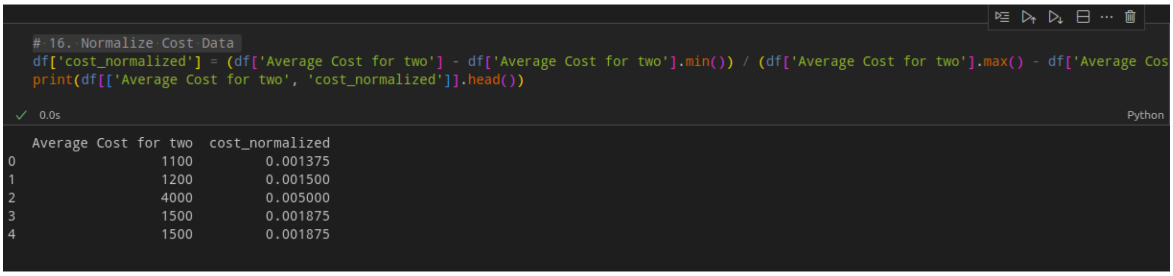
```
Cuisine with highest average rating: American, BBQ, Sandwich
```

13. Visualizing top cuisines

1. 

```
# 15. Visualize Top Cuisines
top_cuisines = df['Cuisines'].value_counts().head(10)
top_cuisines.plot(kind='bar')
plt.title('Top 10 Cuisines by Number of Restaurants')
plt.xlabel('Cuisine')
plt.ylabel('Number of Restaurants')
plt.show()
```

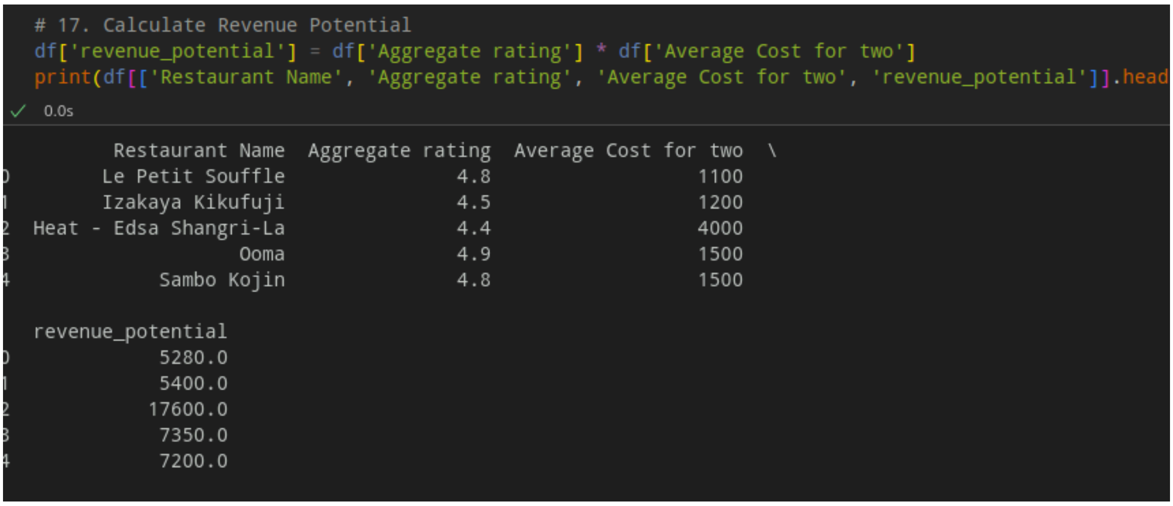
✓ 0.0s

14. Normalizing cost data by scale the "Average Cost for two" column to a range between 0 and 1.
    1. so takes the data easier to compare with other features
    2. 

```
# 16. Normalize Cost Data
df['cost_normalized'] = (df['Average Cost for two'] - df['Average Cost for two'].min()) / (df['Average Cost for two'].max() - df['Average Cos
print(df[['Average Cost for two', 'cost_normalized']].head())
```

✓ 0.0s

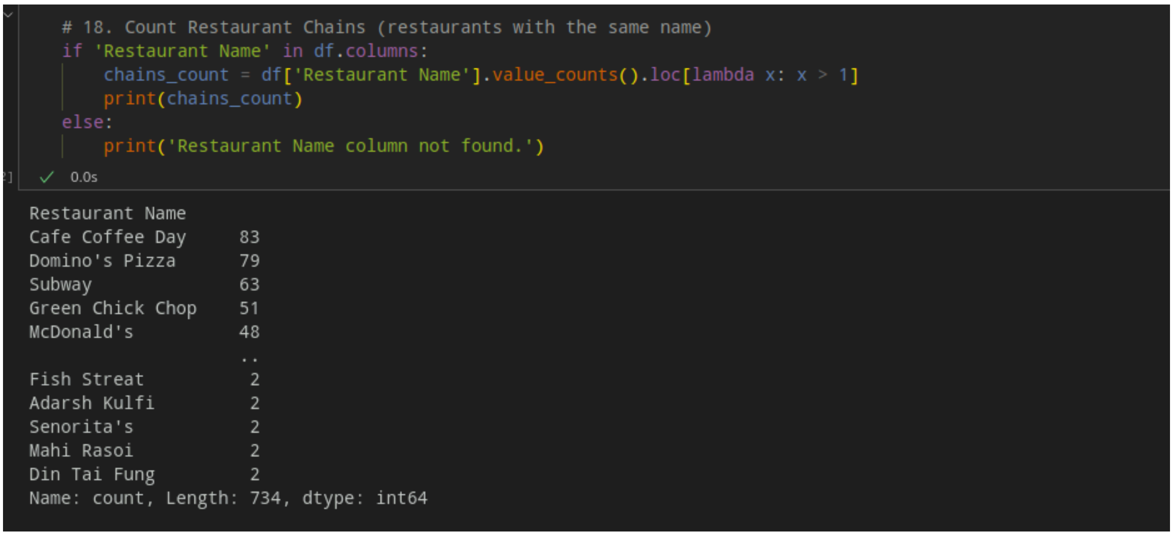
```
   Average Cost for two  cost_normalized
0                  1100         0.001375
1                  1200         0.001500
2                  4000         0.005000
3                  1500         0.001875
4                  1500         0.001875
```

15. Calculating revenue potential
    1. 

```
# 17. Calculate Revenue Potential
df['revenue_potential'] = df['Aggregate rating'] * df['Average Cost for two']
print(df[['Restaurant Name', 'Aggregate rating', 'Average Cost for two', 'revenue_potential']].head
```

✓ 0.0s

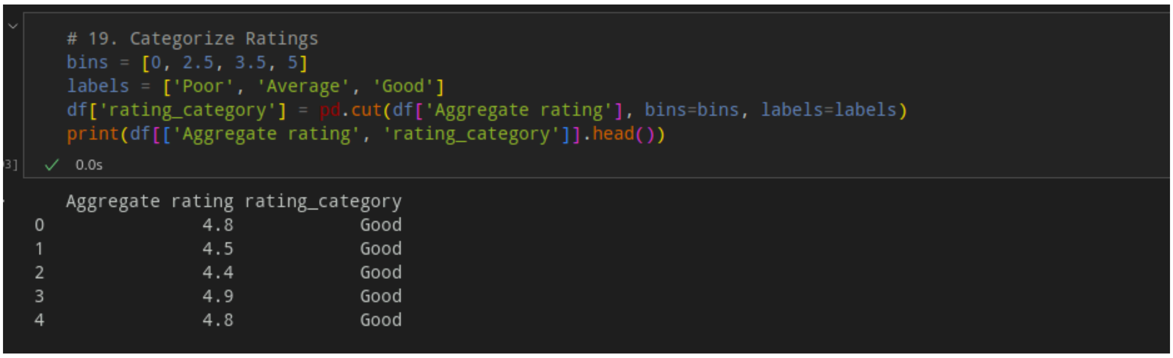
```
          Restaurant Name  Aggregate rating  Average Cost for two  \
0        Le Petit Souffle               4.8                  1100
1        Izakaya Kikufuji               4.5                  1200
2  Heat - Edsa Shangri-La               4.4                  4000
3                    Ooma               4.9                  1500
4             Sambo Kojin               4.8                  1500

   revenue_potential
0             5280.0
1             5400.0
2            17600.0
3             7350.0
4             7200.0
```

16. Counting restaurant chains
    1. 

```
# 18. Count Restaurant Chains (restaurants with the same name)
if 'Restaurant Name' in df.columns:
    chains_count = df['Restaurant Name'].value_counts().loc[lambda x: x > 1]
    print(chains_count)
else:
    print('Restaurant Name column not found.')
```

✓ 0.0s

```
Restaurant Name
Cafe Coffee Day     83
Domino's Pizza      79
Subway              63
Green Chick Chop    51
McDonald's          48
                    ..
Fish Streat          2
Adarsh Kulfi         2
Senorita's           2
Mahi Rasoi           2
Din Tai Fung         2
Name: count, Length: 734, dtype: int64
```

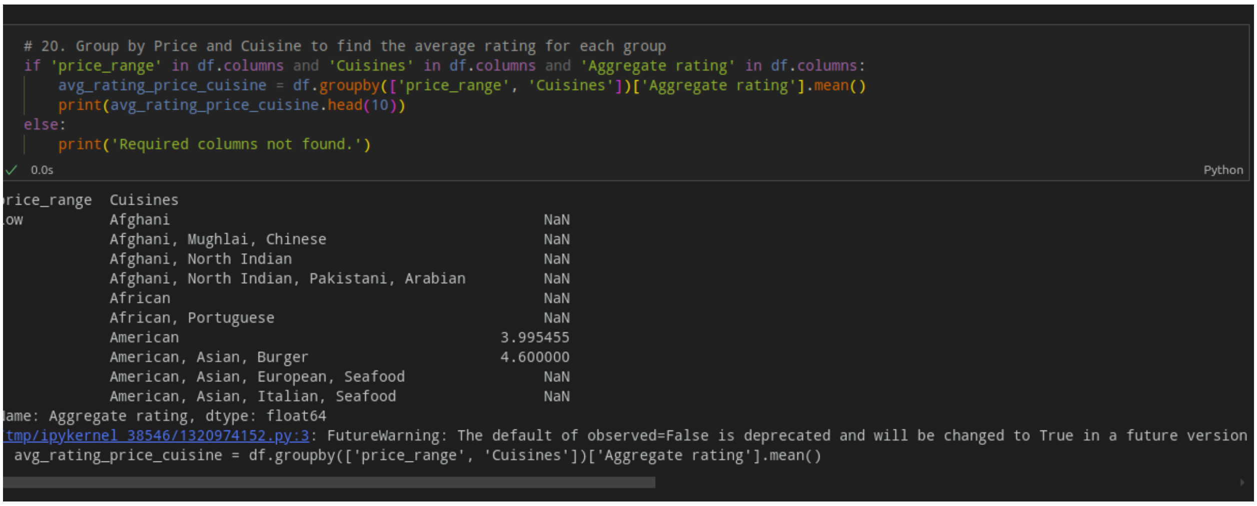
17. Categorizing ratings
    1. 

```
# 19. Categorize Ratings
bins = [0, 2.5, 3.5, 5]
labels = ['Poor', 'Average', 'Good']
df['rating_category'] = pd.cut(df['Aggregate rating'], bins=bins, labels=labels)
print(df[['Aggregate rating', 'rating_category']].head())
```

✓ 0.0s

```
   Aggregate rating rating_category
0               4.8            Good
1               4.5            Good
2               4.4            Good
3               4.9            Good
4               4.8            Good
```

18. Grouping by price and cuisine

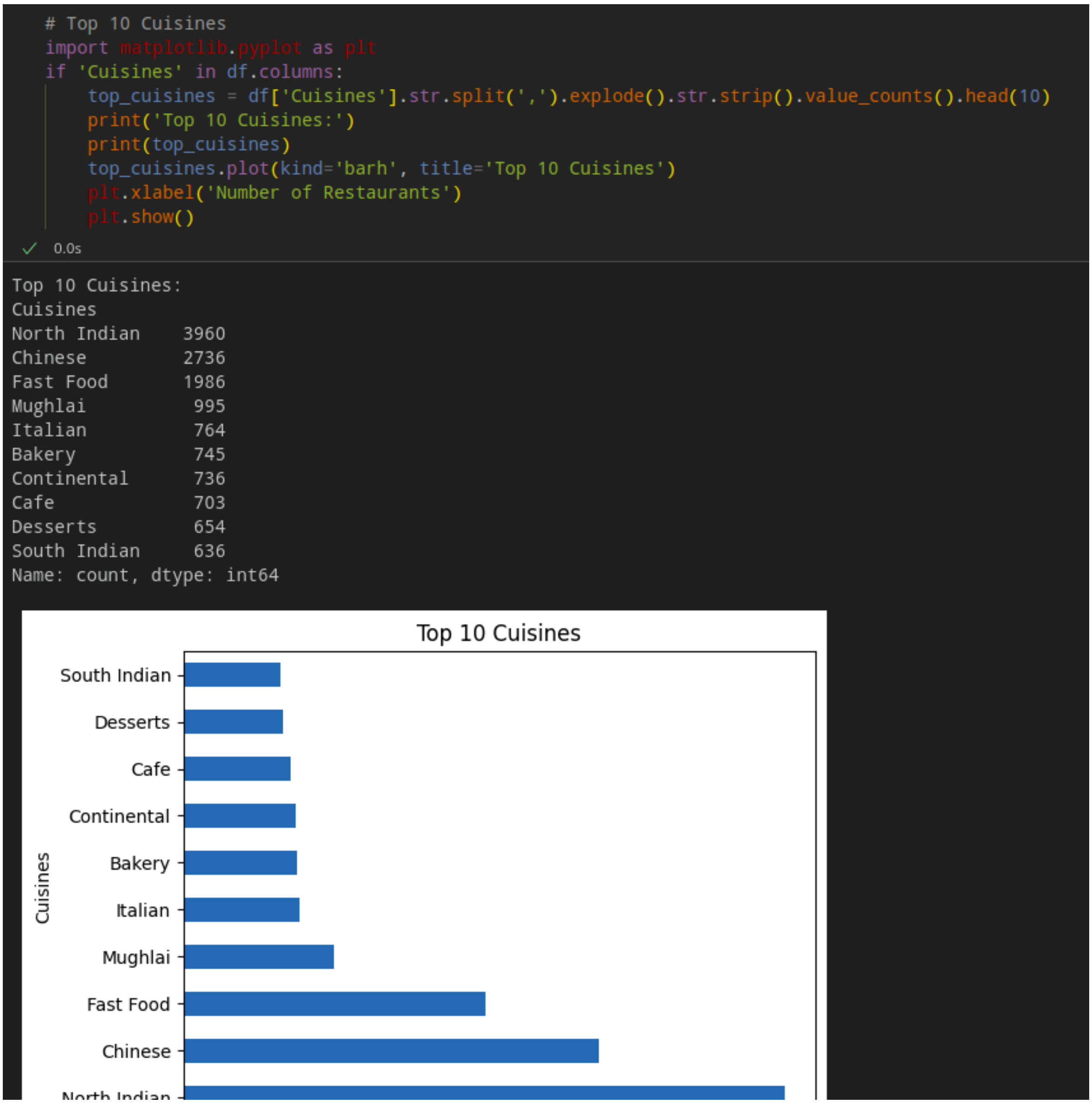


```
# 20. Group by Price and Cuisine to find the average rating for each group
if 'price_range' in df.columns and 'Cuisines' in df.columns and 'Aggregate rating' in df.columns:
    avg_rating_price_cuisine = df.groupby(['price_range', 'Cuisines'])['Aggregate rating'].mean()
    print(avg_rating_price_cuisine.head(10))
else:
    print('Required columns not found.')
```

✓ 0.0s Python

```
rice_range  Cuisines
ow          Afghani                                        NaN
            Afghani, Mughlai, Chinese                      NaN
            Afghani, North Indian                          NaN
            Afghani, North Indian, Pakistani, Arabian      NaN
            African                                        NaN
            African, Portuguese                            NaN
            American                                  3.995455
            American, Asian, Burger                   4.600000
            American, Asian, European, Seafood             NaN
            American, Asian, Italian, Seafood              NaN
ame: Aggregate rating, dtype: float64
tmp/ipykernel_38546/1320974152.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version
 avg_rating_price_cuisine = df.groupby(['price_range', 'Cuisines'])['Aggregate rating'].mean()
```

19. top 10 cuisines

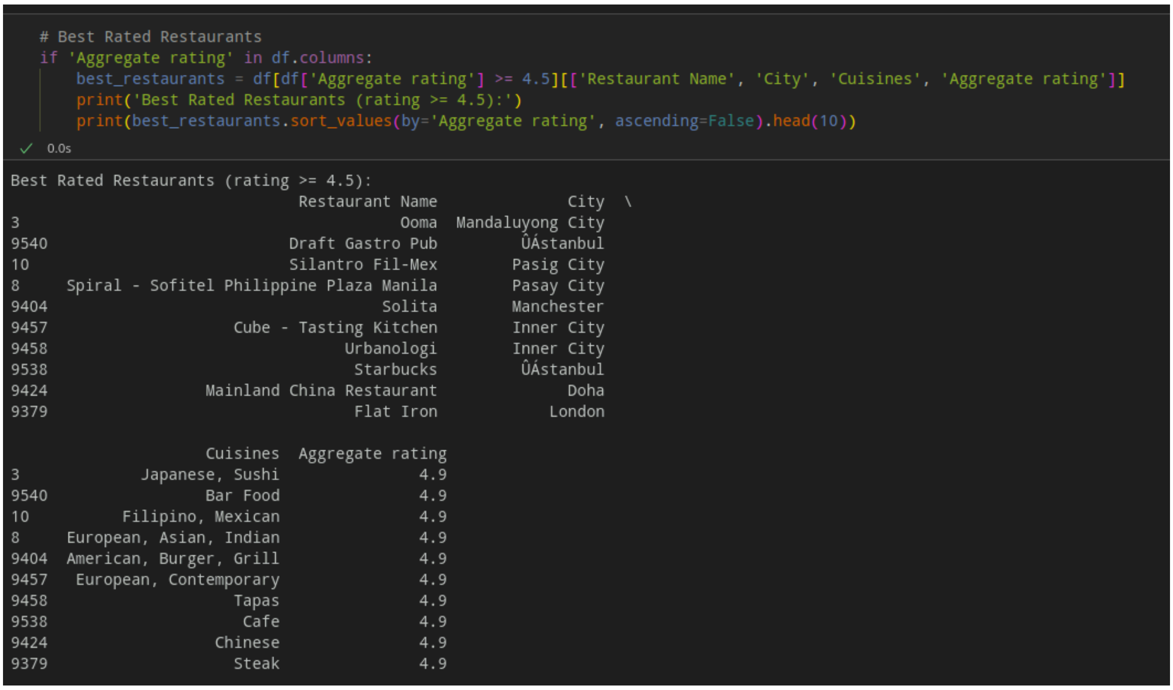
1. 

```
# Top 10 Cuisines
import matplotlib.pyplot as plt
if 'Cuisines' in df.columns:
    top_cuisines = df['Cuisines'].str.split(',').explode().str.strip().value_counts().head(10)
    print('Top 10 Cuisines:')
    print(top_cuisines)
    top_cuisines.plot(kind='barh', title='Top 10 Cuisines')
    plt.xlabel('Number of Restaurants')
    plt.show()
```

✓ 0.0s

```
Top 10 Cuisines:
Cuisines
North Indian    3960
Chinese         2736
Fast Food       1986
Mughlai          995
Italian          764
Bakery           745
Continental      736
Cafe             703
Desserts         654
South Indian     636
Name: count, dtype: int64
```

20. Best Rated restaurants

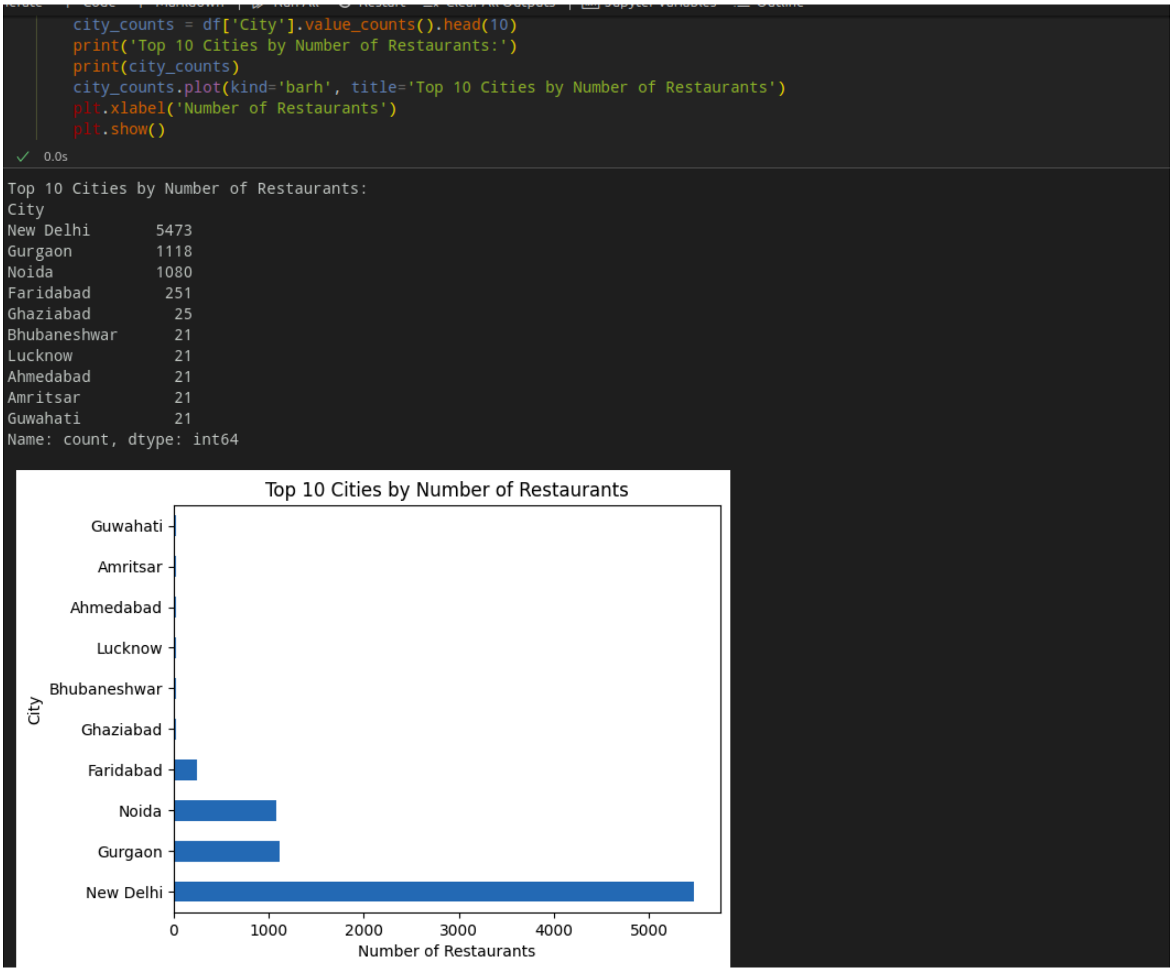
1. 

```
# Best Rated Restaurants
if 'Aggregate rating' in df.columns:
    best_restaurants = df[df['Aggregate rating'] >= 4.5][['Restaurant Name', 'City', 'Cuisines', 'Aggregate rating']]
    print('Best Rated Restaurants (rating >= 4.5):')
    print(best_restaurants.sort_values(by='Aggregate rating', ascending=False).head(10))
```

✓ 0.0s

```
Best Rated Restaurants (rating >= 4.5):
                              Restaurant Name             City  \
3                                        Ooma  Mandaluyong City
9540                        Draft Gastro Pub         ÛÁstanbul
10                          Silantro Fil-Mex        Pasig City
8      Spiral - Sofitel Philippine Plaza Manila        Pasay City
9404                                   Solita        Manchester
9457                   Cube - Tasting Kitchen        Inner City
9458                               Urbanologi        Inner City
9538                                Starbucks         ÛÁstanbul
9424                  Mainland China Restaurant              Doha
9379                                Flat Iron            London

                      Cuisines  Aggregate rating
3              Japanese, Sushi               4.9
9540                  Bar Food               4.9
10            Filipino, Mexican              4.9
8      European, Asian, Indian               4.9
9404   American, Burger, Grill               4.9
9457    European, Contemporary               4.9
9458                     Tapas               4.9
9538                      Cafe               4.9
9424                   Chinese               4.9
9379                     Steak               4.9
```

21. City wise number of restaurants

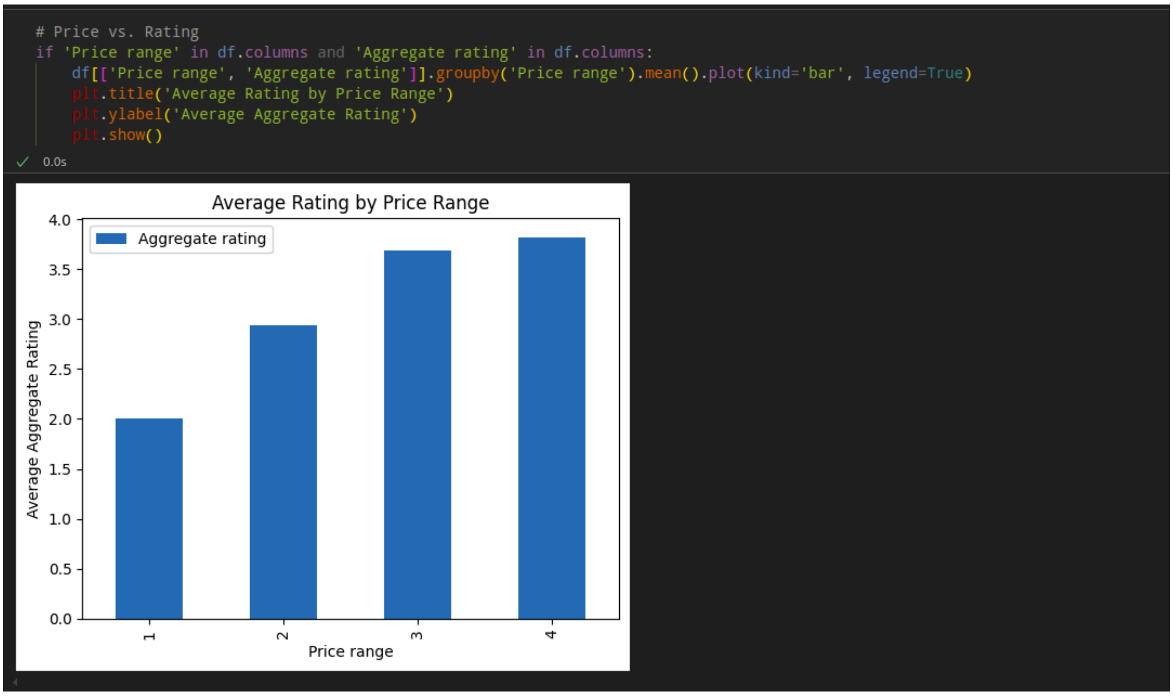
1. 

```
city_counts = df['City'].value_counts().head(10)
print('Top 10 Cities by Number of Restaurants:')
print(city_counts)
city_counts.plot(kind='barh', title='Top 10 Cities by Number of Restaurants')
plt.xlabel('Number of Restaurants')
plt.show()
```

✓ 0.0s

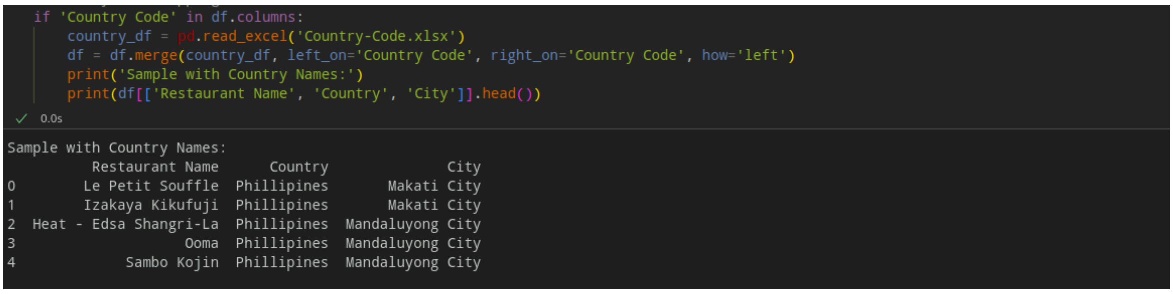
```
Top 10 Cities by Number of Restaurants:
City
New Delhi       5473
Gurgaon         1118
Noida           1080
Faridabad        251
Ghaziabad         25
Bhubaneshwar      21
Lucknow           21
Ahmedabad         21
Amritsar          21
Guwahati          21
Name: count, dtype: int64
```

22. Price vs Rating in avg

1. 

```
# Price vs. Rating
if 'Price range' in df.columns and 'Aggregate rating' in df.columns:
    df[['Price range', 'Aggregate rating']].groupby('Price range').mean().plot(kind='bar', legend=True)
    plt.title('Average Rating by Price Range')
    plt.ylabel('Average Aggregate Rating')
    plt.show()
```

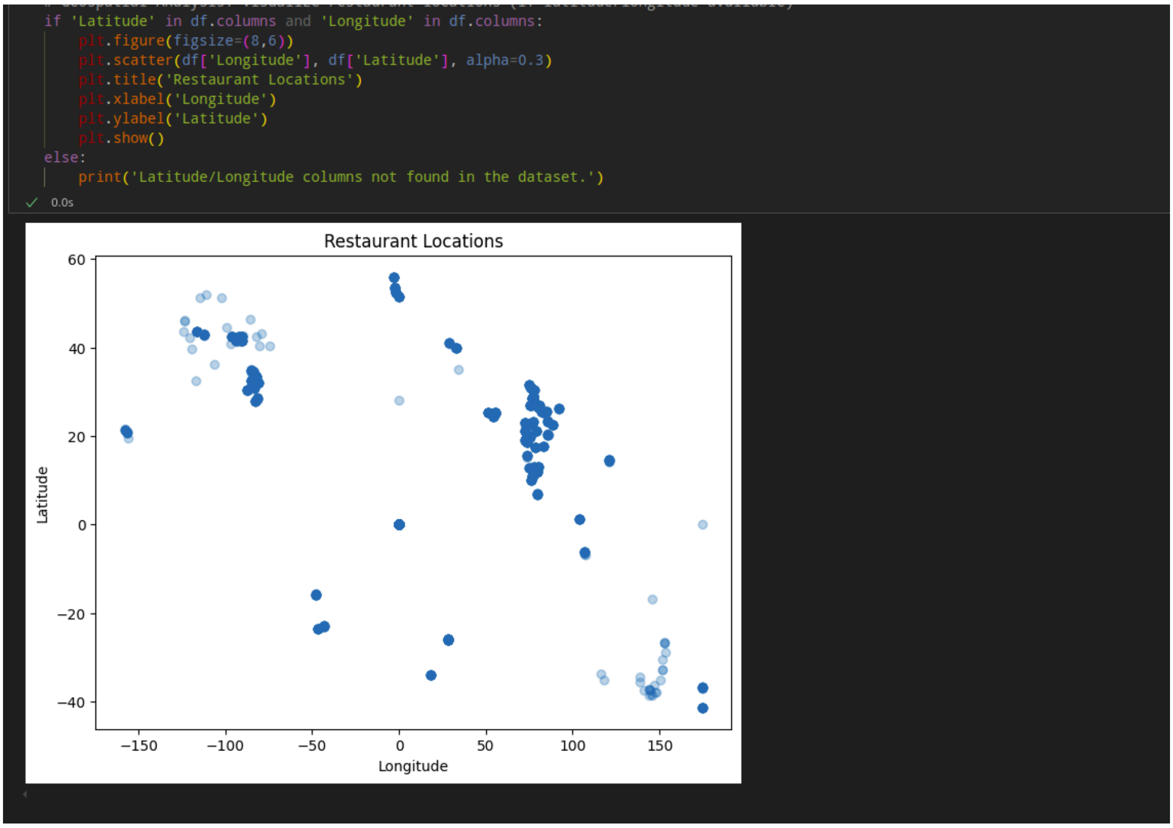
23. Country wise data

1. 

```
if 'Country Code' in df.columns:
    country_df = pd.read_excel('Country-Code.xlsx')
    df = df.merge(country_df, left_on='Country Code', right_on='Country Code', how='left')
    print('Sample with Country Names:')
    print(df[['Restaurant Name', 'Country', 'City']].head())
```

```
Sample with Country Names:
          Restaurant Name      Country             City
0        Le Petit Souffle  Phillipines       Makati City
1        Izakaya Kikufuji  Phillipines       Makati City
2  Heat - Edsa Shangri-La  Phillipines  Mandaluyong City
3                    Ooma  Phillipines  Mandaluyong City
4             Sambo Kojin  Phillipines  Mandaluyong City
```

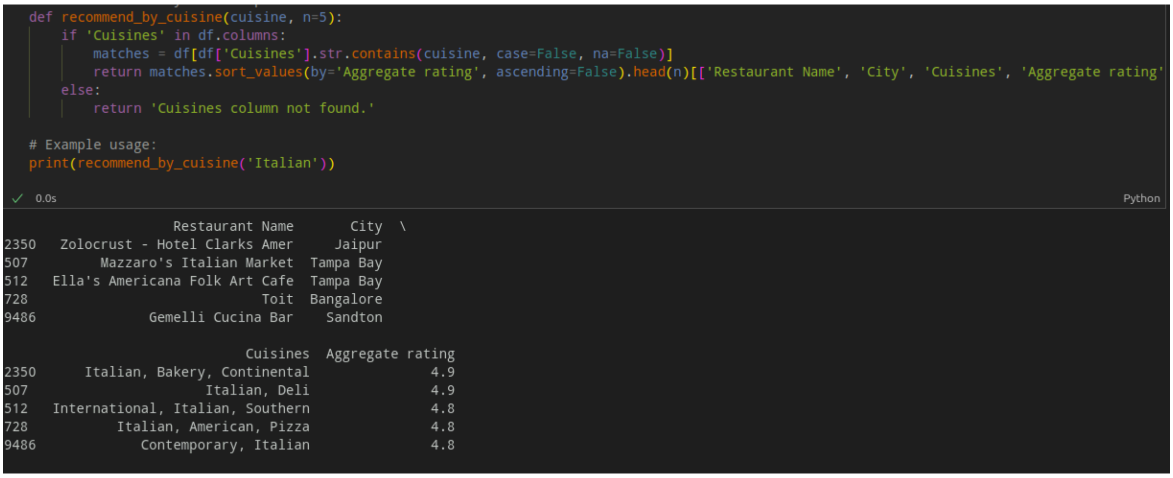
24. Geo-spatially plot graph for each country as per their longitude and latitude

1. 

```
if 'Latitude' in df.columns and 'Longitude' in df.columns:
    plt.figure(figsize=(8,6))
    plt.scatter(df['Longitude'], df['Latitude'], alpha=0.3)
    plt.title('Restaurant Locations')
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')
    plt.show()
else:
    print('Latitude/Longitude columns not found in the dataset.')
```

25. Cuisine recommendation based on ratings

1.

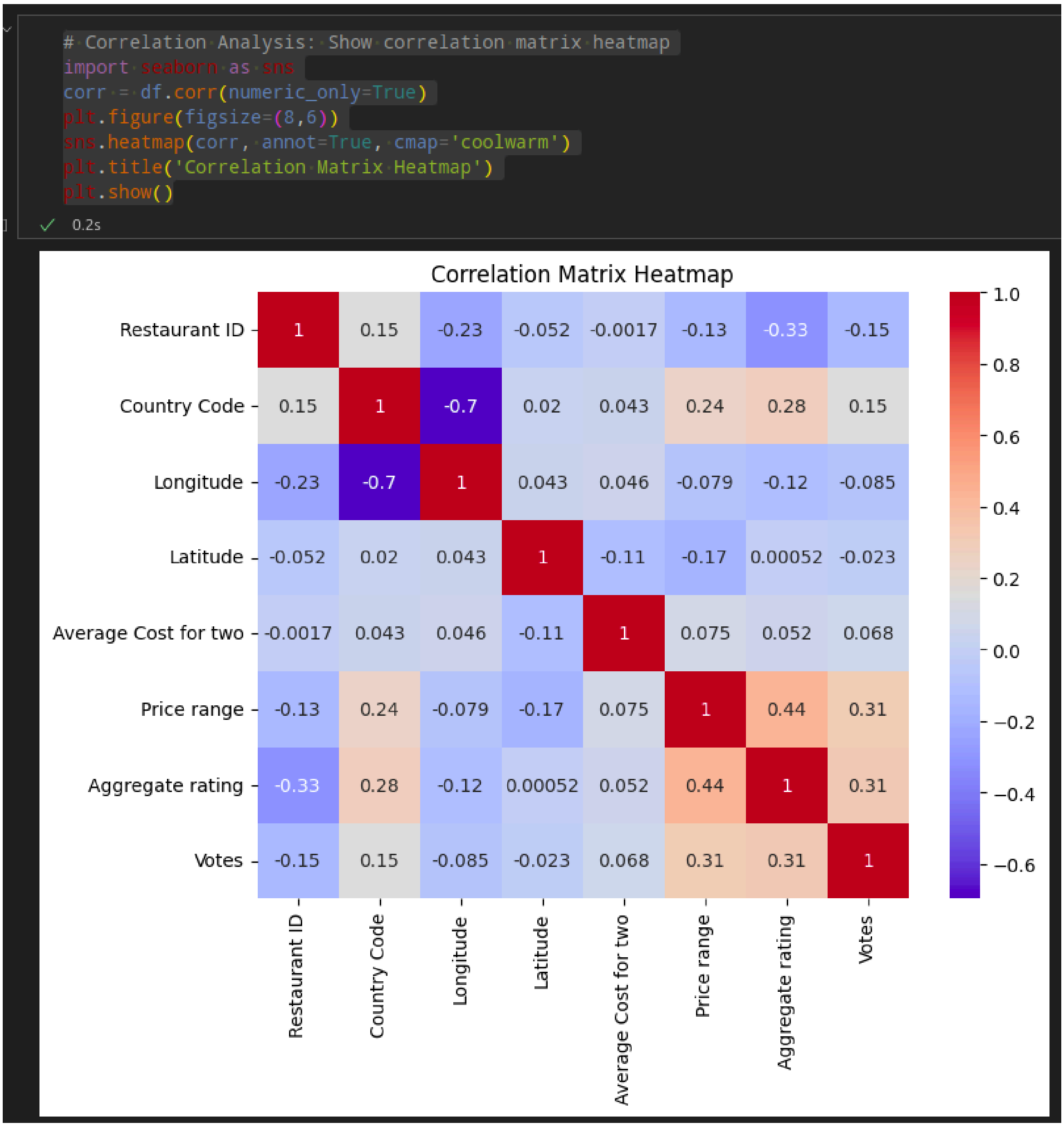


```
def recommend_by_cuisine(cuisine, n=5):
    if 'Cuisines' in df.columns:
        matches = df[df['Cuisines'].str.contains(cuisine, case=False, na=False)]
        return matches.sort_values(by='Aggregate rating', ascending=False).head(n)[['Restaurant Name', 'City', 'Cuisines', 'Aggregate rating'
    else:
        return 'Cuisines column not found.'

# Example usage:
print(recommend_by_cuisine('Italian'))
```

```
                  Restaurant Name       City  \
2350   Zolocrust - Hotel Clarks Amer     Jaipur
507         Mazzaro's Italian Market  Tampa Bay
512   Ella's Americana Folk Art Cafe  Tampa Bay
728                             Toit  Bangalore
9486               Gemelli Cucina Bar    Sandton

                          Cuisines  Aggregate rating
2350     Italian, Bakery, Continental               4.9
507                    Italian, Deli               4.9
512   International, Italian, Southern              4.8
728          Italian, American, Pizza               4.8
9486            Contemporary, Italian               4.8
```

26. Correlation matrix

1.



```
# Correlation Analysis: Show correlation matrix heatmap
import seaborn as sns
corr = df.corr(numeric_only=True)
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```

Correlation Matrix Heatmap

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
Restaurant ID	1	0.15	-0.23	-0.052	-0.0017	-0.13	-0.33	-0.15
Country Code	0.15	1	-0.7	0.02	0.043	0.24	0.28	0.15
Longitude	-0.23	-0.7	1	0.043	0.046	-0.079	-0.12	-0.085
Latitude	-0.052	0.02	0.043	1	-0.11	-0.17	0.00052	-0.023
Average Cost for two	-0.0017	0.043	0.046	-0.11	1	0.075	0.052	0.068
Price range	-0.13	0.24	-0.079	-0.17	0.075	1	0.44	0.31
Aggregate rating	-0.33	0.28	-0.12	0.00052	0.052	0.44	1	0.31
Votes	-0.15	0.15	-0.085	-0.023	0.068	0.31	0.31	1

2. by normalising data between range of -1 to 1 , we can get correlation heatmap that gives us to -spot such issues visually ,

   1. variables that move together (in positive) , move opposite(negative correlation) ,, unrelated(close to 0) ,
   2. How strongly different numeric features (such as rating , avg costs , country code etc) are related to each other.
      1. For example, to know if restaurants with higher costs tend to have higher ratings. we can use this heatmap.

3. we use seaborn library for visualisation , plt figure for size of each plot

---