Laboratory Work 4 : DataFrames Merging, Data Aggregation, and Data Visualization

Course: Python Data Processing

Student: Bahar Berra Uyar / KH-222ia.e

Instructor: Svitlana Mykolaivna Kovalenko

Date: 03/11/2024

Topic and Goal of the Lab

The main goal of this laboratory work is to learn and apply various Pandas methods for data merging and aggregation using datasets related to energy supply, GDP, and journal contributions in the field of Energy Engineering and Power Technology.

Progress of the Work

Task 1: Load Energy Data

- Loaded energy data from the Excel file **"En_In.xls"** into a DataFrame, excluding footer and header information.
- Removed unnecessary columns and renamed the remaining columns to:

['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']

```
!pip install numpy openpyxl
```

```
Collecting numpy •••
```

```python
import pandas as pd
import numpy as np
pd.set_option('future.no_silent_downcasting', True)
# Assignment 1
df = pd.read_excel(
    io='En_In.xls',
    skiprows=17,
    usecols="C:F",
    names=['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable'],
    engine='xlrd'
)
df = df.drop(df.tail(38).index)
df.index = range(1, len(df) + 1)
print(df.head())
```

```
          Country Energy Supply Energy Supply per Capita  % Renewable
1     Afghanistan           321                       10    78.669280
2         Albania           102                       35   100.000000
3         Algeria          1959                       51     0.551010
4  American Samoa           ...                      ...     0.641026
5         Andorra             9                      121    88.695650
```

Task 2: Convert Energy Supply to Gigajoules

- Converted the **"Energy Supply"** values from petajoules to gigajoules, handling missing data by replacing them with np.NaN.

```python
# Assignment 2
df['Energy Supply'] = pd.to_numeric(df['Energy Supply'], errors='coerce')
df['Energy Supply'] *= 1_000_000
df.replace("...", np.nan, inplace=True)
print(df.head())
```

```
          Country  Energy Supply Energy Supply per Capita  % Renewable
1     Afghanistan   3.210000e+08                       10    78.669280
2         Albania   1.020000e+08                       35   100.000000
3         Algeria   1.959000e+09                       51     0.551010
4  American Samoa            NaN                      NaN     0.641026
5         Andorra   9.000000e+06                      121    88.695650
```

Task 3: Clean Country Names

- Cleaned country names by removing numbers and parentheses, ensuring consistency in the naming.

```
# Assignment 3
df['Country'] = df['Country'].str.replace(r'\s*\(.*?\)', '', regex=True)
df['Country'] = df['Country'].str.replace(r'\d+', '', regex=True)
df['Country'] = df['Country'].str.strip()
print(df.iloc[[24, 197, 98]])
```

```
         Country  Energy Supply Energy Supply per Capita  % Renewable
25       Bolivia   3.360000e+08                       32    31.477120
198  Switzerland   1.113000e+09                      136    57.745480
99          Iran   9.172000e+09                      119     5.707721
```

Task 4: Rename Specific Countries

- Renamed the following countries:
  - "Republic of Korea" to "South Korea"
  - "United States of America" to "United States"
  - "United Kingdom of Great Britain and Northern Ireland" to "United Kingdom"
  - "China, Hong Kong Special Administrative Region" to "Hong Kong"

```
# Assignment 4
country_replacements = {
    "Republic of Korea": "South Korea",
    "United States of America": "United States",
    "United Kingdom of Great Britain and Northern Ireland": "United Kingdom",
    "China, Hong Kong Special Administrative Region": "Hong Kong"
}
df['Country'] = df['Country'].replace(country_replacements)
print(df.loc[df['Country'].isin(['American Samoa', 'South Korea', 'Bolivia'])])
```

```
            Country  Energy Supply Energy Supply per Capita  % Renewable
4    American Samoa            NaN                      NaN     0.641026
25          Bolivia   3.360000e+08                       32    31.477120
165     South Korea   1.100700e+10                      221     2.279353
```

Task 5: Load GDP Data

- Loaded GDP data from **"gpd.csv"**, skipping the header.
- Renamed specific countries in the GDP dataset:
  - "Korea, Rep." to "South Korea"
  - "Iran, Islamic Rep." to "Iran"
  - "Hong Kong SAR, China" to "Hong Kong"

```
# Assignment 5
GDP = pd.read_csv("gpd.csv", skiprows=4)
country_gpd_replacements = {
    "Korea, Rep.": "South Korea",
    "Iran, Islamic Rep.": "Iran",
    "Hong Kong SAR, China": "Hong Kong"
}
GDP.iloc[:, 0] = GDP.iloc[:, 0].replace(country_gpd_replacements)
GDP.rename(columns={'Country Name': 'Country'}, inplace=True)
print(GDP.head(1))
```

```
  Country Country Code                                Indicator Name  \
0   Aruba          ABW  GDP at market prices (constant 2010 US$)

   Indicator Code  1960  1961  1962  1963  1964  1965  ...  2006  2007  2008  \
0  NY.GDP.MKTP.KD   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN   NaN   NaN

   2009          2010  2011  2012  2013  2014  2015
0   NaN  2.467704e+09   NaN   NaN   NaN   NaN   NaN

[1 rows x 60 columns]
```

Task 6: Load Sciamgo Journal Data

- Loaded the Sciamgo Journal and Country Rank data from "**scimagojr.xlsx**", focusing on Energy Engineering and Power Technology rankings.

```
# Assignment 6
sciamgo = pd.read_excel("scimagojr.xlsx", engine='openpyxl')
print(sciamgo.head(3))
```

```
   Rank        Country  Documents  Citable documents  Citations  \
0     1          China     127050             126767     597237
1     2  United States      96661              94747     792274
2     3          Japan      30504              30287     223024

   Self-citations  Citations per document  H index
0          411683                    4.70      138
1          265436                    8.20      230
2           61554                    7.31      134
```

Task 7: Join Datasets

- Merged the three datasets based on country names to create a new dataset containing only the last 10 years (2006-2015) of GDP data and the top 15 countries ranked by Scimagojr.

```python
# Assignment 7
# - Only keep the last 10 years (2006-2015) of GDP data
# - Only use the top 15 countries by Scimagojr 'Rank'

# Filter Scimagojr data to include only the top 15 countries
sciamgo_top15 = sciamgo[sciamgo['Rank'] <= 15]

# Select only the last 10 years of GDP data (2006-2015)
gdp_last_10_years = GDP[['Country'] + [str(year) for year in range(2006, 2016)]]

# Merge the top 15 countries with GDP data
merged = pd.merge(sciamgo_top15, gdp_last_10_years, on='Country', how='inner')

# Final merge with energy data
Result = pd.merge(merged, df, on='Country', how='inner')

# Set index to 'Country' for Result DataFrame
Result.set_index('Country', inplace=True)

# Select columns in the specified order
columns = ['Rank', 'Documents', 'Citable documents', 'Citations',
           'Self-citations', 'Citations per document', 'H index',
           'Energy Supply', 'Energy Supply per Capita', '% Renewable'] + \
          [str(year) for year in range(2006, 2016)]

Result = Result[columns]
print(Result)
```

```
                    Rank  Documents  Citable documents  Citations  \
Country
China                  1     127050             126767     597237
United States          2      96661              94747     792274
Japan                  3      30504              30287     223024
United Kingdom         4      20944              20357     206091
Russian Federation     5      18534              18301      34266
Canada                 6      17899              17620     215003
Germany                7      17027              16831     140566
India                  8      15005              14841     128763
France                 9      13153              12973     130632
South Korea           10      11983              11923     114675
Italy                 11      10964              10794     111850
Spain                 12       9428               9330     123336
Iran                  13       8896               8819      57470
Australia             14       8831               8725      90765
Brazil                15       8668               8596      60702

                    Self-citations  Citations per document  H index  \
Country
China                       411683                    4.70      138
United States               265436                    8.20      230
Japan                        61554                    7.31      134
United Kingdom               37874                    9.84      139
Russian Federation           12422                    1.85       57
Canada                       40930                   12.01      149
Germany                      27426                    8.26      126
India                        37209                    8.58      115
France                       28601                    9.93      114
South Korea                  22595                    9.57      104
Italy                        26661                   10.20      106
Spain                        23964                   13.08      115
Iran                         19125                    6.46       72
```

Task 8: Average GDP Function

- Created a function to determine the top 15 countries for average GDP over the last 10 years.

```
# Assignment 8
def top_15_gdp_average(df):
    average_gdp = df[[str(year) for year in range(2006, 2016)]].mean(axis=1)
    return average_gdp.nlargest(15)
top_15_gdp_average(Result)
```

```
Country
United States         1.536434e+13
China                 6.348609e+12
Japan                 5.542208e+12
Germany               3.493025e+12
France                2.681725e+12
United Kingdom        2.487907e+12
Brazil                2.189794e+12
Italy                 2.120175e+12
India                 1.769297e+12
Canada                1.660647e+12
Russian Federation    1.565459e+12
Spain                 1.418078e+12
Australia             1.164043e+12
South Korea           1.106715e+12
Iran                  4.441558e+11
dtype: float64
```

Task 9: GDP Change Function

- Developed a function to calculate GDP change for the country with the 5th largest average GDP.

```
# Assignment 9
def gdp_change_5th_largest(df):
    average_gdp = df[[str(year) for year in range(2006, 2016)]].mean(axis=1)
    fifth_largest_country = average_gdp.nlargest(5).index[-1]
    gdp_change = df.loc[fifth_largest_country, '2015'] - df.loc[fifth_largest_country, '2006']
    return (fifth_largest_country, gdp_change)
gdp_change_5th_largest(Result)
```

```
('France', np.float64(153345695364.24023))
```

Task 10: Maximum Renewable Percentage Function

- Created a function to identify the country with the maximum percentage of renewable energy and its value.

```
# Assignment 10
def max_renewable(df):
    max_country = df['% Renewable'].idxmax()
    max_percentage = df['% Renewable'].max()
    return (max_country, max_percentage)
max_renewable(Result)
```

```
('Brazil', np.float64(69.64803))
```

Task 11: Estimate Population

- Estimated population based on Energy Supply and Energy Supply per Capita, determining the sixth most populous country.

```
# Assignment 11
def estimated_population(df):
    df['Energy Supply'] = pd.to_numeric(df['Energy Supply'], errors='coerce')
    df['Energy Supply per Capita'] = pd.to_numeric(df['Energy Supply per Capita'], errors='coerce')
    df['Estimated Population'] = df['Energy Supply'] / df['Energy Supply per Capita']
    df = df.dropna(subset=['Estimated Population'])
    sixth_most_populous = df['Estimated Population'].nlargest(6).idxmin()
    return (sixth_most_populous, df.loc[sixth_most_populous, 'Estimated Population'])
estimated_population(Result)
```

```
('Japan', np.float64(127409395.97315437))
```

Task 12: Correlation Calculation

- Estimated the number of citable documents per person and calculated the correlation between citable documents per capita and energy supply per capita.

```
# Assignment 12
def correlation_citable_energy(df):
    df['Citable per Capita'] = df['Citable documents'] / df['Estimated Population']
    return df['Citable per Capita'].corr(df['Energy Supply per Capita'])

# Call the function and display the result
correlation_citable_energy(Result)
```

```
np.float64(0.7940010435442946)
```

Task 13: Renewable Value Classification

- Created a binary column indicating whether a country's % Renewable is above or below the median.

```
# Assignment 13
def renewable_above_median(df):
    median_renewable = df['% Renewable'].median()
    return (df['% Renewable'] >= median_renewable).astype(int).sort_index()
renewable_above_median(Result)
```

```
Country
Australia             0
Brazil                1
Canada                1
China                 1
France                1
Germany               1
India                 0
Iran                  0
Italy                 1
Japan                 0
Russian Federation    1
South Korea           0
Spain                 1
United Kingdom        0
United States         0
Name: % Renewable, dtype: int64
```

Task 14: Group by Continent

- Utilized a dictionary to group countries by continent and generated a DataFrame showing the sample size, sum, mean, and standard deviation of estimated populations.

```python
# Assignment 14
import pandas as pd

# Define the continent dictionary
ContinentDict = {
    'China': 'Asia',
    'United States': 'North America',
    'Japan': 'Asia',
    'United Kingdom': 'Europe',
    'Russian Federation': 'Europe',
    'Canada': 'North America',
    'Germany': 'Europe',
    'India': 'Asia',
    'France': 'Europe',
    'South Korea': 'Asia',
    'Italy': 'Europe',
    'Spain': 'Europe',
    'Iran': 'Asia',
    'Australia': 'Australia',
    'Brazil': 'South America'
}

def continent_summary(df, continent_dict):
    df['Continent'] = df.index.map(continent_dict)
    summary = df.groupby('Continent')['Estimated Population'].agg(['count', 'sum', 'mean', 'std'])
    summary.rename(columns={'count': 'size'}, inplace=True)
    return summary

# Call the function and display the result
continent_summary(Result, ContinentDict)
```

Continent	size	sum	mean	std
Asia	5	2.898666e+09	5.797333e+08	6.790979e+08
Australia	1	2.331602e+07	2.331602e+07	NaN
Europe	6	4.579297e+08	7.632161e+07	3.464767e+07
North America	2	3.528552e+08	1.764276e+08	1.996696e+08
South America	1	2.059153e+08	2.059153e+08	NaN

Task 15: Bubble Chart

- Created a bubble chart visualizing % Renewable vs. Rank, with bubble size representing GDP in 2015 and color corresponding to continent.

```python
# Assignment 15
import matplotlib.pyplot as plt
ContinentDict = {
    'China': 'Asia',
    'United States': 'North America',
    'Japan': 'Asia',
    'United Kingdom': 'Europe',
    'Russian Federation': 'Europe',
    'Canada': 'North America',
    'Germany': 'Europe',
    'India': 'Asia',
    'France': 'Europe',
    'South Korea': 'Asia',
    'Italy': 'Europe',
    'Spain': 'Europe',
    'Iran': 'Asia',
    'Australia': 'Australia',
    'Brazil': 'South America'
}
Result['Continent'] = Result.index.map(ContinentDict)
x = Result['Rank']
y = Result['% Renewable']
size = Result['2015'] / 1e9
color_map = {
    'Asia': 'red',
    'North America': 'blue',
    'Europe': 'green',
    'Australia': 'yellow',
    'South America': 'orange'
}
```

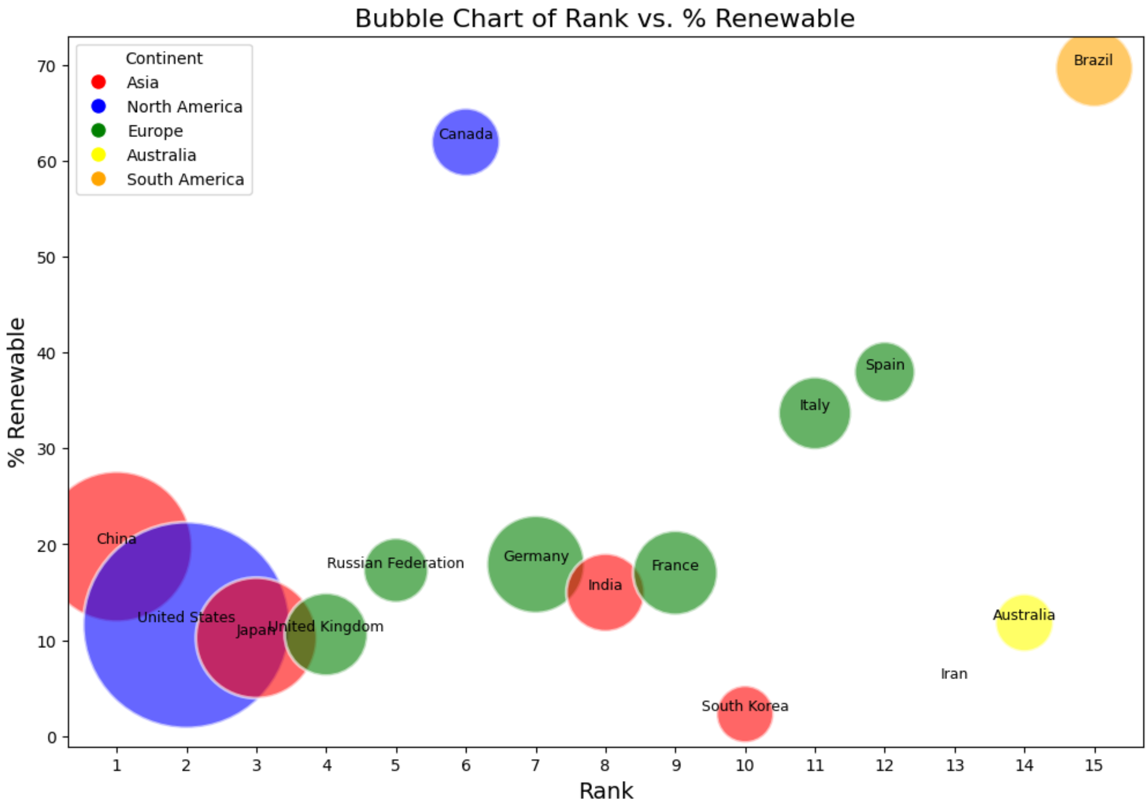
```python
plt.figure(figsize=(12, 8))
bubble = plt.scatter(x, y, s=size, c=Result['Continent'].map(color_map), alpha=0.6, edgecolors="w", linewidth=2)
for i in range(len(Result)):
    country = Result.index[i]
    plt.text(x.iloc[i], y.iloc[i], country, fontsize=9, ha='center', va='bottom')

plt.title('Bubble Chart of Rank vs. % Renewable', fontsize=16)
plt.xlabel('Rank', fontsize=14)
plt.ylabel('% Renewable', fontsize=14)


plt.xticks(ticks=range(1, 16))


plt.grid(False)

handles = []
for continent, color in color_map.items():
    handles.append(plt.Line2D([0], [0], marker='o', color='w', label=continent,
                              markerfacecolor=color, markersize=10))
plt.legend(handles=handles, title='Continent')
plt.show()
```



Link to Jupyter Notebook

- https://github.com/BaharBerra/PythonLab4.git

Conclusions

This laboratory work successfully demonstrated how to manipulate and analyze various datasets using Pandas. Through merging, cleaning, and visualizing data, insights regarding energy supply and economic indicators were derived. This process not only enhances understanding of data handling in Python but also emphasizes the importance of data quality and structure in analysis.