Homework — Introduction to Data Mining (Jupyter Notebook)

Title: Countries by Population — From Web to DataFrame and Basic Analysis

Submission: Upload the completed notebook file YOURNAME_HW1_CountriesPopulation.ipynb to the course GitHub (rida87/DataMining: Data Mining UL).

Include any saved CSVs or images used.

Objectives

- Retrieve a table from a web page and load it into a Pandas DataFrame.
- Clean and convert textual numeric values to numeric types.
- Perform basic DataFrame queries and manipulations.
- Compute descriptive statistics (mean, median, mode, variance, std, quantiles).
- Create simple visualizations and perform one small Data Mining task (clustering).
- Practice working in Jupyter Notebook and documenting results with Markdown

Required setup (install the libarairies below if they are not installed)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import requests
from sklearn.cluster import KMeans
```

Dataset source

Use the Wikipedia page table:

https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population

(You will read the table from the page using requests and pandas.read_html().)

Step-by-step tasks (what to do — include code + short answer cells)

1) Fetch the table from the web (1 cell)

- Use requests.get() with a browser User-Agent header to fetch the page (avoids HTTP 403).
- Parse tables using pd.read_html(response.text).

Example code

##

import requests

url ="https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population"

headers = {"User-Agent":"Mozilla/5.0"}

resp = requests.get(url, headers=headers)

tables = pd.read_html(resp.text)

- Show how many tables were found and display the first rows of the first few tables to choose the correct one

##

2) Select & preview the correct table (1 cell)

- Inspect tables[i].head() to identify the table that contains country names and population.
- Assign it to df and display df.head().

3) Clean column names and extract relevant columns (1 cell)

- Standardize column names (strip spaces).
- Select a country/name column and a population column and rename them:

##

df = df[[country_col, pop_col]].copy()

df.columns = ['country','population_raw']

##

4) Clean numeric population values (1 cell)

- Remove footnote markers like [1], commas, and spaces.
- Convert values to integers and drop rows with missing population.

##

```
import re
def clean_num(x):
  if isinstance(x, str):
    x = re.sub(r"\[.*?\]", "", x)
    x = x.replace(",", "").strip()
  try:
    return int(x)
  except:
    return np.nan

df['population'] = df['population_raw'].apply(clean_num)
df = df.drop(columns='population_raw').dropna(subset=['population']).reset_index(drop=True)
df['population'] = df['population'].astype(int)
```

##

5) Basic info & sanity checks (1 cell)

- Print df.shape, df.info(), and df.head(10).
- Verify there are roughly ~200 countries and no missing population values.

6) Simple queries and manipulations (1–2 cells)

Perform and display results for:

- Top 10 most populous countries (sorted list).
- Number of countries with population < 1,000,000.

- Look up a country by name (e.g., France).
- Select only country and population columns and show examples.

Hints

##

top10 = df.sort_values('population', ascending=False).head(10)

small_countries = df[df['population'] < 1_000_000]

df[df['country'].str.contains("France", case=False, na=False)]

##

7) Descriptive statistics (1 cell)

Compute and display:

- Mean, median, mode, sample variance (ddof=1), sample standard deviation (ddof=1).
- Min, max, and the 25th/50th/75th percentiles.
- Also display df['population'].describe().

Example

##

pop = df['population']

print(pop.mean(), pop.median(), pop.mode().iloc[0])

print(pop.var(ddof=1), pop.std(ddof=1))

display(pop.describe())

##

8) Visualizations (1–2 cells)

- Histogram of population (log scale on x-axis recommended).
- Boxplot of log10(population).
- Make plots labeled and with titles.

Hints

##

```
plt.hist(pop, bins=40)

plt.xscale('log')

plt.title("Histogram of country populations (log-scale)")

plt.show()


plt.boxplot(np.log10(pop))

plt.title("Boxplot (log10 population)")

plt.show()
```

##

9) Outlier detection (1 cell)

- Use IQR on log10(population): compute Q1, Q3, IQR, then lower/upper cutoffs and list outliers.
- Display any outlier countries found and comment briefly (1–2 lines) on why they are outliers.

10) Data Mining mini-task — clustering (1 cell)

- Perform KMeans clustering with **3 clusters** on log10(population) to create groups: small/medium/large.
- Add a pop_cluster_name column with cluster labels ordered by centroid size.
- Display counts for each cluster and list top 5 countries in the 'large' cluster.

Hints

```
X = np.log10(df[['population']].values)

kmeans = KMeans(n_clusters=3, random_state=42).fit(X)

df['pop_cluster'] = kmeans.labels_

# Order clusters by centroid values to name them small/medium/large
```

11) Short interpretation questions (Markdown cell answers)

Add short written answers (2–5 lines each) to:

1. Which countries are in the 'large' cluster? List top 5.
2. Compare mean vs. median population and explain the difference.

3. Why use log-transform before visualization/clustering?
4. Name two problems when scraping web tables.
5. (Optional) Try KMeans with 4 clusters and comment on results.

Deliverables (what to submit)

1. YOURNAME_HW1_CountriesPopulation.ipynb — the completed Jupyter Notebook with code cells executed and Markdown answers.
2. (Optional) countries_population_clean.csv if you saved a local cleaned CSV.