Extraction of Dataset

```
# 1. Upload the ZIP file from your local system
from google.colab import files
uploaded = files.upload()  # Choose the .zip file when prompted

# 2. Unzip the file
import zipfile
import io

for filename in uploaded.keys():
    if filename.endswith('.zip'):
        with zipfile.ZipFile(io.BytesIO(uploaded[filename]), 'r') as zip_ref:
            zip_ref.extractall("/content/water_dataset")  # Extract to folder
            print(f"Extracted {filename} to /content/water_dataset")

# 3. Check extracted files
import os
os.listdir("/content/water_dataset")
```

Choose Files Report50_Appendix.zip

- **Report50_Appendix.zip**(application/x-zip-compressed) - 64842875 bytes, last modified: 8/11/2025 - 100% done

```
Saving Report50_Appendix.zip to Report50_Appendix.zip
Extracted Report50_Appendix.zip to /content/water_dataset
['Report50_Appendix']
```

```
import pandas as pd

# Example for CSV file
df = pd.read_excel("/content/water_dataset/Report50_Appendix/Report50-Appendix-VIII&IX.xls")

# Example for Excel file
# df = pd.read_excel("/content/water_dataset/your_file.xlsx")

df.head()
```

	Unnamed: 0	Unnamed: 1
0	Please cite this publication as follows:	NaN
1	NaN	NaN
2	Mekonnen, M.M. and Hoekstra, A.Y. (2011) Natio...	NaN
3	NaN	Value of Water Research Report Series No. 50, ...
4	NaN	http://www.waterfootprint.org/Reports/Report50...

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

```
import pandas as pd

# Load Excel and check all sheet names
xls_path = "/content/water_dataset/Report50_Appendix/Report50-Appendix-VIII&IX.xls"  # Change to actual path
xls = pd.ExcelFile(xls_path)

print("Available sheets:", xls.sheet_names)

# Load a specific sheet, e.g., 'Appendix-VIII'
df = pd.read_excel(xls_path, sheet_name='Appendix-VIII')

# Display first 5 rows
df.head(10)
```

```
Available sheets: ['Note', 'Appendix-VIII', 'Appendix-IX']
```

	Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 14
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
1	Period 1996 - 2005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	Country	Population (thousands)	Water footprint of consumption of agricultural...	NaN	NaN	NaN	NaN	NaN	Water footprint of consumption of industrial p...	NaN	...	Total water footprint of national consumption
3	NaN	NaN	Internal	NaN	NaN	External	NaN	NaN	Internal	NaN	...	Internal
4	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	Grey	...	Green
5	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184065	...	602.231584
6	Algeria	30767.4	634.559021	88.774786	9.569506	700.600118	50.559278	59.659627	0.705285	6.395073	...	634.559021
7	Angola	14609.8	798.354488	13.612756	2.387999	108.419722	14.588961	13.138565	0.058225	0.903797	...	798.354488
8	Antigua and Barbuda	77.4	391.207836	0.951738	0	827.951695	87.800365	74.827365	0.488697	9.154387	...	391.207836
9	Argentina	37060	1288.348321	87.911893	45.396413	35.02824	3.94031	3.00187	3.126123	33.876497	...	1288.348321

10 rows × 24 columns

Start coding or generate with AI.

```
# If your file is .xls you might need xlrd (only in older pandas versions).
# Uncomment if pandas can't read your file:
# !pip install xlrd==1.2.0

import pandas as pd
xls_path = "/content/water_dataset/Report50_Appendix/Report50-Appendix-VIII&IX.xls"   # <- change to actual path
xls = pd.ExcelFile(xls_path)
print("Sheets available:", xls.sheet_names)

# Quick preview of the first 10 rows to find where headers live
preview = pd.read_excel(xls_path, sheet_name=xls.sheet_names[1], header=None, nrows=6)
for i, row in preview.iterrows():
    print(i, list(row))
```

```
Sheets available: ['Note', 'Appendix-VIII', 'Appendix-IX']
0 ['Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and e
1 [nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]
2 ['Period 1996 - 2005 ', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
3 ['Country', 'Population  (thousands)', 'Water footprint of consumption of agricultural products', nan, nan, nan, nan, nan, 'Water
4 [nan, nan, 'Internal', nan, nan, 'External', nan, nan, 'Internal', nan, 'External', nan, nan, nan, 'Internal', nan, nan, 'External
5 [nan, nan, 'Green', 'Blue', 'Grey', 'Green', 'Blue', 'Grey', 'Blue', 'Grey', 'Blue', 'Grey', 'blue', 'grey', 'Green', 'Blue', 'Gre
```

```
sheet_name = "Appendix-VIII"   # adjust
header_rows = [1, 5]           # change if preview showed different header rows

df_raw = pd.read_excel(xls_path, sheet_name=sheet_name, header=header_rows)
df_raw.shape, df_raw.columns
```

```
((175, 24),
 MultiIndex([( 'Unnamed: 0_level_0', 'Unnamed: 0_level_1'),
            ( 'Unnamed: 1_level_0', 'Unnamed: 1_level_1'),
            ( 'Unnamed: 2_level_0',             'Green'),
            ( 'Unnamed: 3_level_0',              'Blue'),
            ( 'Unnamed: 4_level_0',              'Grey'),
```

```
            ( 'Unnamed: 5_level_0',              'Green'),
            ( 'Unnamed: 6_level_0',               'Blue'),
            ( 'Unnamed: 7_level_0',               'Grey'),
            ( 'Unnamed: 8_level_0',               'Blue'),
            ( 'Unnamed: 9_level_0',               'Grey'),
            ('Unnamed: 10_level_0',               'Blue'),
            ('Unnamed: 11_level_0',               'Grey'),
            ('Unnamed: 12_level_0',               'blue'),
            ('Unnamed: 13_level_0',               'grey'),
            ('Unnamed: 14_level_0',              'Green'),
            ('Unnamed: 15_level_0',               'Blue'),
            ('Unnamed: 16_level_0',               'Grey'),
            ('Unnamed: 17_level_0',              'Green'),
            ('Unnamed: 18_level_0',               'Blue'),
            ('Unnamed: 19_level_0',               'Grey'),
            ('Unnamed: 20_level_0',              'Green'),
            ('Unnamed: 21_level_0',               'Blue'),
            ('Unnamed: 22_level_0',               'Grey'),
            ('Unnamed: 23_level_0',              'Total')],
           ))
```

```
display(df.iloc[0])
```

	0
Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	NaN
Unnamed: 1	NaN
Unnamed: 2	NaN
Unnamed: 3	NaN
Unnamed: 4	NaN
Unnamed: 5	NaN
Unnamed: 6	NaN
Unnamed: 7	NaN
Unnamed: 8	NaN
Unnamed: 9	NaN
Unnamed: 10	NaN
Unnamed: 11	NaN
Unnamed: 12	NaN
Unnamed: 13	NaN
Unnamed: 14	NaN
Unnamed: 15	NaN
Unnamed: 16	NaN
Unnamed: 17	NaN
Unnamed: 18	NaN
Unnamed: 19	NaN
Unnamed: 20	NaN
Unnamed: 21	NaN
Unnamed: 22	NaN
Unnamed: 23	NaN

```
# Drop the first row as it contains only NaN values
df = df.iloc[1:].copy()

# Display the first few rows of the cleaned DataFrame
display(df.head())
```

	Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 14
1	Period 1996 - 2005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	Country	Population (thousands)	Water footprint of consumption of agricultural...	NaN	NaN	NaN	NaN	NaN	Water footprint of consumption of industrial p...	NaN	...	Total water footprint of national consumption
3	NaN	NaN	Internal	NaN	NaN	External	NaN	NaN	Internal	NaN	...	Internal
4	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	Grey	...	Green
5	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184065	...	602.231584

5 rows × 24 columns

```
display(df.iloc[1])
```

	2
Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	Country
Unnamed: 1	Population (thousands)
Unnamed: 2	Water footprint of consumption of agricultural...
Unnamed: 3	NaN
Unnamed: 4	NaN
Unnamed: 5	NaN
Unnamed: 6	NaN
Unnamed: 7	NaN
Unnamed: 8	Water footprint of consumption of industrial p...
Unnamed: 9	NaN
Unnamed: 10	NaN
Unnamed: 11	NaN
Unnamed: 12	Water footprint of domestic water consumption
Unnamed: 13	NaN
Unnamed: 14	Total water footprint of national consumption
Unnamed: 15	NaN
Unnamed: 16	NaN
Unnamed: 17	NaN
Unnamed: 18	NaN
Unnamed: 19	NaN
Unnamed: 20	NaN
Unnamed: 21	NaN
Unnamed: 22	NaN
Unnamed: 23	NaN

dtype: object

```
df.head()
```

	Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 14
1	Period 1996 - 2005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	Country	Population (thousands)	Water footprint of consumption of agricultural...	NaN	NaN	NaN	NaN	NaN	Water footprint of consumption of industrial p...	NaN	...	Total water footprint of national consumption
3	NaN	NaN	Internal	NaN	NaN	External	NaN	NaN	Internal	NaN	...	Internal
4	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	Grey	...	Green
5	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184065	...	602.231584

5 rows × 24 columns

```
display(df.iloc[1])
```

	2
Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	Country
Unnamed: 1	Population (thousands)
Unnamed: 2	Water footprint of consumption of agricultural...
Unnamed: 3	NaN
Unnamed: 4	NaN
Unnamed: 5	NaN
Unnamed: 6	NaN
Unnamed: 7	NaN
Unnamed: 8	Water footprint of consumption of industrial p...
Unnamed: 9	NaN
Unnamed: 10	NaN
Unnamed: 11	NaN
Unnamed: 12	Water footprint of domestic water consumption
Unnamed: 13	NaN
Unnamed: 14	Total water footprint of national consumption
Unnamed: 15	NaN
Unnamed: 16	NaN
Unnamed: 17	NaN
Unnamed: 18	NaN
Unnamed: 19	NaN
Unnamed: 20	NaN
Unnamed: 21	NaN
Unnamed: 22	NaN
Unnamed: 23	NaN

dtype: object

```
df.head()
```

	Appendix VIII. The water footprint of national consumption per capita, shown by major consumption category and by internal and external component (m3/yr/cap)	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 14
1	Period 1996 - 2005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	Country	Population (thousands)	Water footprint of consumption of agricultural...	NaN	NaN	NaN	NaN	NaN	Water footprint of consumption of industrial p...	NaN	...	Total water footprint of national consumption
3	NaN	NaN	Internal	NaN	NaN	External	NaN	NaN	Internal	NaN	...	Internal
4	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	Grey	...	Green
5	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184065	...	602.231584

5 rows × 24 columns

Cleaning or preprocessing of dataset

```
import pandas as pd

xls_path = "/content/water_dataset/Report50_Appendix/Report50-Appendix-VIII&IX.xls"  # adjust to your file

# Read the sheet with the 3-row header
df_raw = pd.read_excel(
    xls_path,
    sheet_name="Appendix-VIII",  # adjust sheet name if needed
    header=[2, 3, 4]
)

# Flatten the multiindex columns
def flatten_cols(cols):
    flat = []
    for col in cols:
        # col is a tuple of header parts
        parts = [str(c).strip() for c in col if pd.notna(c) and str(c).strip() != ""]
        flat.append(" - ".join(parts))
    return flat

df_raw.columns = flatten_cols(df_raw.columns)

# Drop completely empty rows
df = df_raw.dropna(axis=0, how='all').reset_index(drop=True)

# Drop any “NaN” columns that may have come from blank Excel cells
df = df.dropna(axis=1, how='all')

df.head()
```

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2	Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal	Period 1 - 200 Wa footpr consumpt industr product Interna
0	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	G
1	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184
2	Algeria	30767.4	634.559021	88.774786	9.569506	700.600118	50.559278	59.659627	0.705285	6.395
3	Angola	14609.8	798.354488	13.612756	2.387999	108.419722	14.588961	13.138565	0.058225	0.903
4	Antigua and Barbuda	77.4	391.207836	0.951738	0	827.951695	87.800365	74.827365	0.488697	9.154

5 rows × 24 columns

```
df.head()
```

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2	Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal	Period 1 - 200 Wa footpr consumpt industr product Interna
0	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	G
1	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184
2	Algeria	30767.4	634.559021	88.774786	9.569506	700.600118	50.559278	59.659627	0.705285	6.395
3	Angola	14609.8	798.354488	13.612756	2.387999	108.419722	14.588961	13.138565	0.058225	0.903
4	Antigua and Barbuda	77.4	391.207836	0.951738	0	827.951695	87.800365	74.827365	0.488697	9.154

5 rows × 24 columns

```
print(df.shape)
```

```
(176, 24)
```

```
df.columns
```

```
Index(['Period 1996 - 2005 - Country - Unnamed: 0_level_2',
       'Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2',
       'Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal',
       'Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1',
       'Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2',
       'Period 1996 - 2005 - Water footprint of consumption of agricultural products - External',
       'Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1',
       'Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2',
       'Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal',
       'Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal.1',
       'Period 1996 - 2005 - Water footprint of consumption of industrial products - External',
       'Period 1996 - 2005 - Water footprint of consumption of industrial products - External.1',
       'Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 12_level_2',
       'Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 13_level_2',
       'Period 1996 - 2005 - Total water footprint of national consumption - Internal',
       'Period 1996 - 2005 - Total water footprint of national consumption - Internal.1',
       'Period 1996 - 2005 - Total water footprint of national consumption - Internal.2',
       'Period 1996 - 2005 - Total water footprint of national consumption - External',
       'Period 1996 - 2005 - Total water footprint of national consumption - External.1',
       'Period 1996 - 2005 - Total water footprint of national consumption - External.2',
       'Period 1996 - 2005 - Total water footprint of national consumption - Total',
       'Period 1996 - 2005 - Total water footprint of national consumption - Total.1',
       'Period 1996 - 2005 - Total water footprint of national consumption - Total.2',
       'Period 1996 - 2005 - Total water footprint of national consumption - Total.3'],
      dtype='object')
```

```
country = "Benin"
```

```
# Find the row for the specified country
country_row = df[df['Period 1996 - 2005 - Country - Unnamed: 0_level_2'].str.contains(country, case=False, na=False)]

if not country_row.empty:
    # Display the entire row for the country
    display(country_row)
else:
    print(f"Country '{country}' not found in the dataset.")
```

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2	Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal	Period - 20 W footp consump indust produc Intern
16	Benin	6820.0	1020.167948	1.043061	5.181128	67.283274	20.014469	8.613079	0.208279	3.93

1 rows × 24 columns

```
countries = ["Benin", "India", "Chile"]

# Combine list into a single regex pattern for partial matching
pattern = "|".join(countries)

# Filter rows where the country column matches any of the patterns
country_rows = df[df['Period 1996 - 2005 - Country - Unnamed: 0_level_2']
                  .str.contains(pattern, case=False, na=False)]

if not country_rows.empty:
    display(country_rows)
else:
    print(f"No matching countries found: {countries}")
```

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2	Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal	Period - 20 W footp consump indust produc Intern
16	Benin	6820.0	1020.167948	1.043061	5.181128	67.283274	20.014469	8.613079	0.208279	3.93
32	Chile	15492.1	451.456278	135.603008	122.32155	327.297521	17.953961	16.899745	6.016005	20.1
73	India	1051289.5	685.485778	210.54789	91.633058	22.397271	2.072832	1.79354	1.200373	22.68

3 rows × 24 columns

```
df.describe()
```

	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2
count	1.750000e+02
mean	6.990430e+04
std	4.801317e+05
min	4.640000e+01
25%	2.389000e+03
50%	7.988300e+03
75%	2.316265e+04
max	6.154564e+06

Cleaning Check

```
print("Column Names:", df.columns.tolist())
print("Total Columns:", len(df.columns), "| Unique:", len(set(df.columns)))
```

```
Column Names: ['Period 1996 - 2005 - Country - Unnamed: 0_level_2', 'Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level
Total Columns: 24 | Unique: 24
```

```
print("\nMissing Values per Column:")
print(df.isna().sum())
```

```
Missing Values per Column:
Period 1996 - 2005 - Country - Unnamed: 0_level_2                                             1
Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2                             1
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal      0
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1    0
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2    0
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External      0
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1    0
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2    0
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal        0
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal.1      0
Period 1996 - 2005 - Water footprint of consumption of industrial products - External        0
Period 1996 - 2005 - Water footprint of consumption of industrial products - External.1      0
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 12_level_2     0
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 13_level_2     0
Period 1996 - 2005 - Total water footprint of national consumption - Internal                0
Period 1996 - 2005 - Total water footprint of national consumption - Internal.1              0
Period 1996 - 2005 - Total water footprint of national consumption - Internal.2              0
Period 1996 - 2005 - Total water footprint of national consumption - External                0
Period 1996 - 2005 - Total water footprint of national consumption - External.1              0
Period 1996 - 2005 - Total water footprint of national consumption - External.2              0
Period 1996 - 2005 - Total water footprint of national consumption - Total                   0
Period 1996 - 2005 - Total water footprint of national consumption - Total.1                 0
Period 1996 - 2005 - Total water footprint of national consumption - Total.2                 0
Period 1996 - 2005 - Total water footprint of national consumption - Total.3                 0
dtype: int64
```

```
if df.duplicated().any():
    print("\nWarning: Duplicate rows found!")
else:
    print("\n✅ No duplicate rows.")
```

```
✅ No duplicate rows.
```

```
print("\nData Types:")
print(df.dtypes)
```

```
Data Types:
Period 1996 - 2005 - Country - Unnamed: 0_level_2                                             object
Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2                            float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal       object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1     object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2     object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External       object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1     object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2     object
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal         object
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal.1       object
Period 1996 - 2005 - Water footprint of consumption of industrial products - External         object
Period 1996 - 2005 - Water footprint of consumption of industrial products - External.1       object
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 12_level_2      object
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 13_level_2      object
Period 1996 - 2005 - Total water footprint of national consumption - Internal                 object
Period 1996 - 2005 - Total water footprint of national consumption - Internal.1               object
Period 1996 - 2005 - Total water footprint of national consumption - Internal.2               object
Period 1996 - 2005 - Total water footprint of national consumption - External                 object
Period 1996 - 2005 - Total water footprint of national consumption - External.1               object
Period 1996 - 2005 - Total water footprint of national consumption - External.2               object
Period 1996 - 2005 - Total water footprint of national consumption - Total                    object
Period 1996 - 2005 - Total water footprint of national consumption - Total.1                  object
Period 1996 - 2005 - Total water footprint of national consumption - Total.2                  object
Period 1996 - 2005 - Total water footprint of national consumption - Total.3                  object
dtype: object
```

```
print("\nSample Data:")
display(df.head())
```

Sample Data:

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2	Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal	Period 1' - 200 Wa footpr consumpt industr product Interna
0	NaN	NaN	Green	Blue	Grey	Green	Blue	Grey	Blue	G
1	Albania	3084.9	602.231584	88.864915	36.483578	492.659133	77.421789	39.462323	2.576686	48.184
2	Algeria	30767.4	634.559021	88.774786	9.569506	700.600118	50.559278	59.659627	0.705285	6.395
3	Angola	14609.8	798.354488	13.612756	2.387999	108.419722	14.588961	13.138565	0.058225	0.903
4	Antigua and Barbuda	77.4	391.207836	0.951738	0	827.951695	87.800365	74.827365	0.488697	9.154

5 rows × 24 columns

```
# Save to CSV
df.to_csv("cleaned_water_footprint.csv", index=False)

# Save as Pickle (preserves Python objects and datatypes better)
df.to_pickle("cleaned_water_footprint.pkl")

print("✅ Cleaned dataset saved successfully.")
```

✅ Cleaned dataset saved successfully.

Working on cleaned ones.

```
import pandas as pd

# Load CSV
df_cleaned = pd.read_csv("cleaned_water_footprint.csv")

# OR load Pickle
df_cleaned_pkl = pd.read_pickle("cleaned_water_footprint.pkl")
```

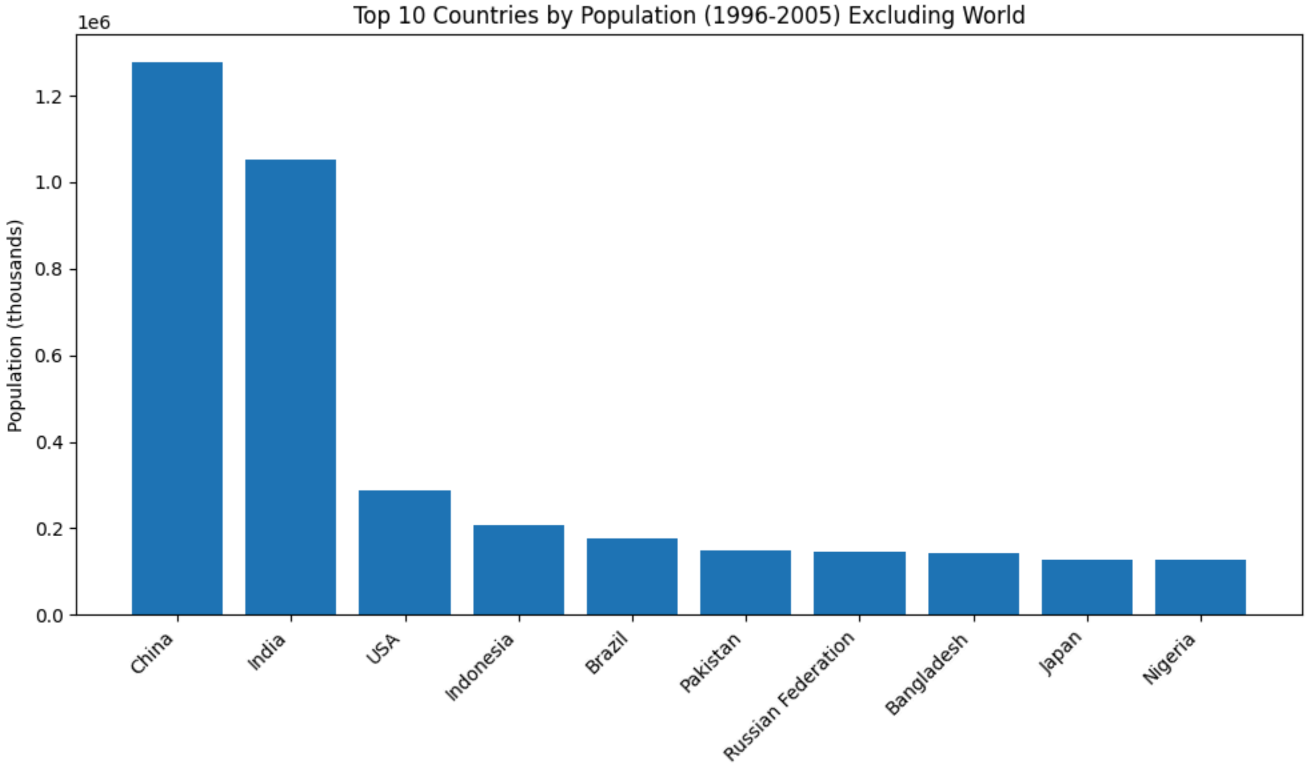
```
df_cleaned.head()
```

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Peri 20(foc consu agr p E
0	NaN	NaN	Green	Blue	Grey	Green	Blue	
1	Albania	3084.9	602.2315836586439	88.86491480425744	36.48357818121366	492.65913305173393	77.4217894564738	39.46232
2	Algeria	30767.4	634.5590207541915	88.77478580796868	9.569505932031948	700.6001182035875	50.559277610878894	59.6596
3	Angola	14609.8	798.3544881648705	13.612755716098459	2.387998998481127	108.41972227414324	14.588961160295314	13.138564
4	Antigua and Barbuda	77.4	391.2078359882946	0.9517381369414658	0	827.9516946250168	87.80036476099782	74.82736

5 rows × 24 columns

```
import matplotlib.pyplot as plt

# Pick the right column (renaming for convenience)
pop_col = 'Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2'
country_col = 'Period 1996 - 2005 - Country - Unnamed: 0_level_2' # Corrected country column name

# Filter out the 'World' row
```

```
df_filtered = df_cleaned[df_cleaned[country_col] != 'World'].copy()


# Get top 10 countries by population from the filtered DataFrame
top10_population = df_filtered.nlargest(10, pop_col)

# Plot
plt.figure(figsize=(10, 6))
plt.bar(top10_population[country_col], top10_population[pop_col]) # Use the corrected country column name
plt.xticks(rotation=45, ha="right")
plt.ylabel("Population (thousands)")
plt.title("Top 10 Countries by Population (1996-2005) Excluding World")
plt.tight_layout()
plt.show()
```



Start coding or generate with AI.

```
print(df_cleaned.columns.tolist())
```

```
['Period 1996 - 2005 - Country - Unnamed: 0_level_2', 'Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2', 'Period :
```

```
print("\nData Types:")
print(df_cleaned.dtypes)
```

```
Data Types:
Period 1996 - 2005 - Country - Unnamed: 0_level_2                                  object
Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2                 float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal       object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1     object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2     object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External       object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1     object
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2     object
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal         object
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal.1       object
Period 1996 - 2005 - Water footprint of consumption of industrial products - External         object
Period 1996 - 2005 - Water footprint of consumption of industrial products - External.1       object
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 12_level_2     object
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 13_level_2     object
Period 1996 - 2005 - Total water footprint of national consumption - Internal               object
Period 1996 - 2005 - Total water footprint of national consumption - Internal.1             object
Period 1996 - 2005 - Total water footprint of national consumption - Internal.2             object
Period 1996 - 2005 - Total water footprint of national consumption - External               object
Period 1996 - 2005 - Total water footprint of national consumption - External.1             object
Period 1996 - 2005 - Total water footprint of national consumption - External.2             object
Period 1996 - 2005 - Total water footprint of national consumption - Total                  object
Period 1996 - 2005 - Total water footprint of national consumption - Total.1                object
Period 1996 - 2005 - Total water footprint of national consumption - Total.2                object
Period 1996 - 2005 - Total water footprint of national consumption - Total.3                object
```

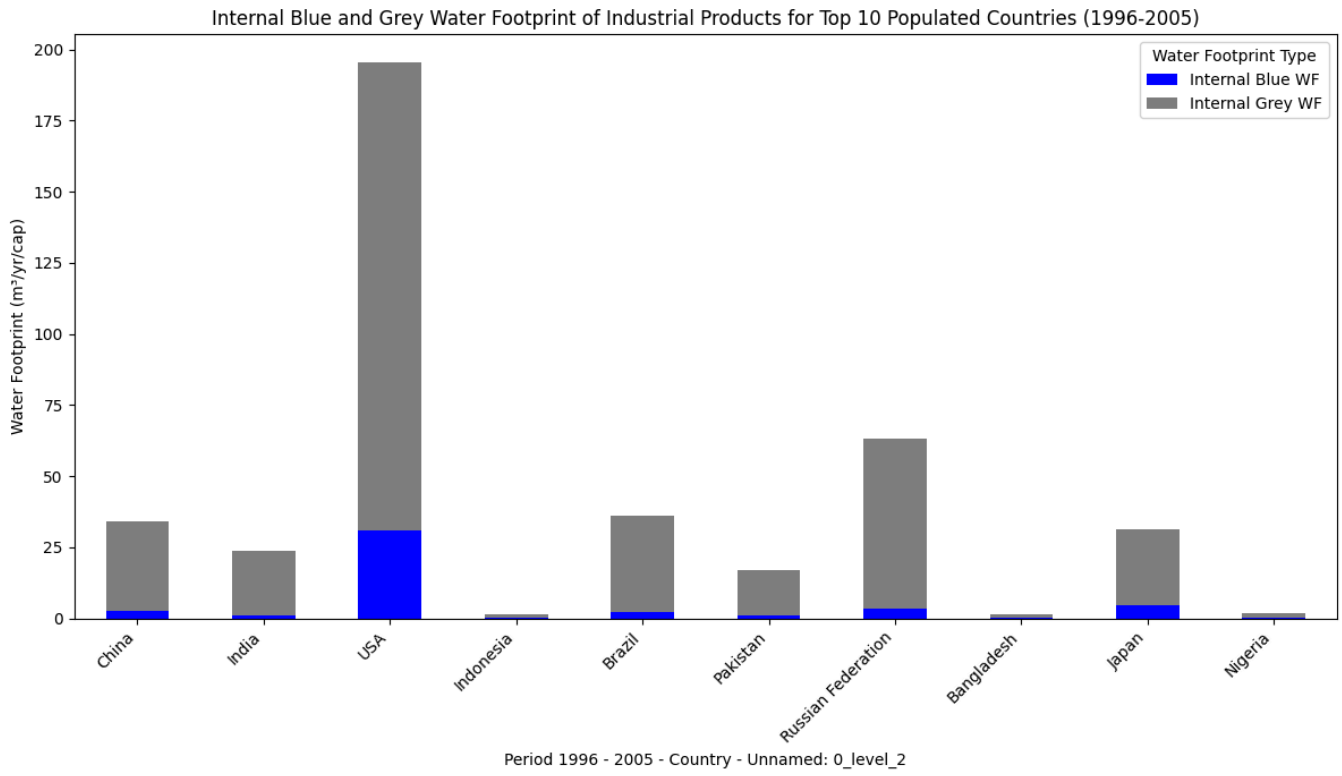
```
dtype: object
```

```python
# Identify the columns that should be numeric (excluding the country column)
numeric_cols = df_cleaned.columns.drop('Period 1996 - 2005 - Country - Unnamed: 0_level_2')

# Convert these columns to numeric, coercing errors
for col in numeric_cols:
    df_cleaned[col] = pd.to_numeric(df_cleaned[col], errors='coerce')

# Display the updated data types
print("\nUpdated Data Types:")
print(df_cleaned.dtypes)
```

```
Updated Data Types:
Period 1996 - 2005 - Country - Unnamed: 0_level_2                                          object
Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2                        float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal      float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1    float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2    float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External      float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1    float64
Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2    float64
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal        float64
Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal.1      float64
Period 1996 - 2005 - Water footprint of consumption of industrial products - External        float64
Period 1996 - 2005 - Water footprint of consumption of industrial products - External.1      float64
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 12_level_2     float64
Period 1996 - 2005 - Water footprint of domestic water consumption - Unnamed: 13_level_2     float64
Period 1996 - 2005 - Total water footprint of national consumption - Internal                float64
Period 1996 - 2005 - Total water footprint of national consumption - Internal.1              float64
Period 1996 - 2005 - Total water footprint of national consumption - Internal.2              float64
Period 1996 - 2005 - Total water footprint of national consumption - External                float64
Period 1996 - 2005 - Total water footprint of national consumption - External.1              float64
Period 1996 - 2005 - Total water footprint of national consumption - External.2              float64
Period 1996 - 2005 - Total water footprint of national consumption - Total                   float64
Period 1996 - 2005 - Total water footprint of national consumption - Total.1                 float64
Period 1996 - 2005 - Total water footprint of national consumption - Total.2                 float64
Period 1996 - 2005 - Total water footprint of national consumption - Total.3                 float64
dtype: object
```

Start coding or generate with AI.

```python
import matplotlib.pyplot as plt
import pandas as pd # Ensure pandas is imported

# Define column names for easier access
country_col = 'Period 1996 - 2005 - Country - Unnamed: 0_level_2'
pop_col = 'Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2'
industrial_internal_blue_col = 'Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal'
industrial_internal_grey_col = 'Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal.1'


# Filter out the 'World' row and get the top 10 by population
df_filtered = df_cleaned[df_cleaned[country_col] != 'World'].copy()
top10_countries_data = df_filtered.nlargest(10, pop_col)

# Select only the relevant columns for plotting
plot_data = top10_countries_data[[country_col, industrial_internal_blue_col, industrial_internal_grey_col]].copy()

# Set the country name as the index for easier plotting
plot_data = plot_data.set_index(country_col)

# Plotting
plt.figure(figsize=(12, 7))

# Create a stacked bar plot
plot_data.plot(kind='bar', stacked=True, color=['blue', 'grey'], ax=plt.gca())

plt.title('Internal Blue and Grey Water Footprint of Industrial Products for Top 10 Populated Countries (1996-2005)')
plt.ylabel('Water Footprint (m³/yr/cap)')
plt.xticks(rotation=45, ha="right")
plt.legend(['Internal Blue WF', 'Internal Grey WF'], title='Water Footprint Type')
plt.tight_layout()
plt.show()
```



Inference 1

So, here's the inference. See the before graph where I have added top 10 populated Countries graph. Where China and India are on a lead to population.

But here you see. This one is the Water Footprint graph with respect those top 10 populated countries. Even though it's a old one but just see the graph. USA at the first and then Russian Fedartion which is at the 7th position. But in the or regarding the footprint it's still at the second position.

So, like this we can now concentrate more on these countries companies to get our job done.

🌏 Inference 1: Water Footprint vs Population – A Strategic Insight for WSI

In the initial graph, we examined the top 10 most populated countries, where China and India clearly lead in terms of population. However, when we shift our focus to the Water Footprint graph for these same countries, an interesting discrepancy emerges. Despite being third in population, the United States ranks first in water footprint. Similarly, the Russian Federation, which holds the seventh position in population, surprisingly comes second in water footprint.

This contrast reveals that water footprint is not solely dependent on population size—it is heavily influenced by industrial and agricultural activities. This insight is crucial for shaping our Water Sustainability Index (WSI), as it suggests that we should concentrate more on countries with disproportionately high water footprints, regardless of their population rank.

The water footprint data is divided into three layers: **green**, **blue**, and **grey**. While green represents rainwater used by crops and is more relevant to agriculture, our focus is on the **industrial impact**, which is best captured by the **blue** and **grey** layers. Blue refers to the consumption of surface and groundwater in production processes, while grey indicates the volume of freshwater required to dilute pollutants from industrial discharge. These two layers are most relevant for assessing the sustainability of industrial operations.

Therefore, our strategic direction should prioritize companies operating in countries like the USA and Russia, where industrial water usage and pollution are significantly high. By analyzing their blue and grey water footprints, we can better evaluate their sustainability practices and environmental impact. This targeted approach will help us build a more effective and data-driven Water Sustainability Index.

```
country = "Germany"

# Find the row for the specified country
country_row = df_cleaned[df_cleaned['Period 1996 - 2005 - Country - Unnamed: 0_level_2'].str.contains(country, case=False, na=False)]

if not country_row.empty:
    # Display the entire row for the country
```

```
    display(country_row)
else:
    print(f"Country '{country}' not found in the dataset.")
```

	Period 1996 - 2005 - Country - Unnamed: 0_level_2	Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - Internal.2	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.1	Period 1996 - 2005 - Water footprint of consumption of agricultural products - External.2	Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal	Period - 20 W footp consump indust produc Intern
61	Germany	82138.8	311.385568	1.021981	85.892265	741.925377	56.890017	62.287062	9.855744	10.40

1 rows × 24 columns

Start coding or generate with AI.

Task

Generate a world map visualization showing the distribution of 'Total Blue Water Footprint of national consumption' and 'Total Grey Water Footprint of national consumption' across different countries using the data in `df_cleaned`.

Install necessary libraries

Subtask:

Install libraries for geospatial data handling and plotting (e.g., `geopandas`, `matplotlib`).

Reasoning: The subtask requires installing `geopandas` and `matplotlib` for geospatial visualization.

```
!pip install geopandas matplotlib
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.11/dist-packages (1.1.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.11/dist-packages (from geopandas) (2.0.2)
Requirement already satisfied: pyogrio>=0.7.2 in /usr/local/lib/python3.11/dist-packages (from geopandas) (0.11.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from geopandas) (25.0)
Requirement already satisfied: pandas>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from geopandas) (2.2.2)
Requirement already satisfied: pyproj>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from geopandas) (3.7.1)
Requirement already satisfied: shapely>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from geopandas) (2.1.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.59.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.0->geopandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.0->geopandas) (2025.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from pyogrio>=0.7.2->geopandas) (2025.8.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

Load world map data

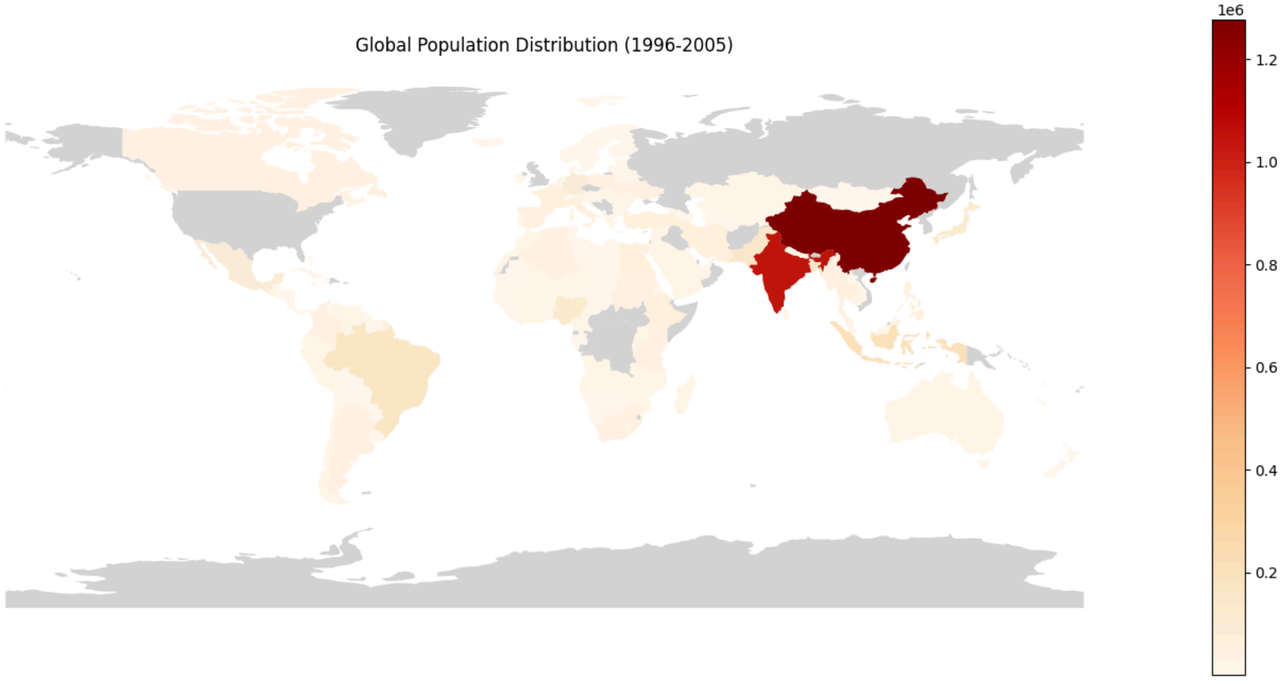
Subtask:

Load a world map shapefile or dataset into a GeoDataFrame.

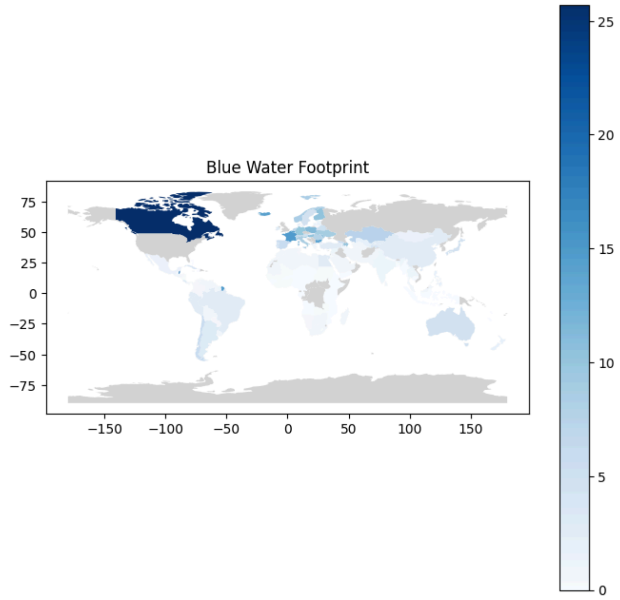
```
import geopandas as gpd

# Path to your downloaded ZIP file
shapefile_path = "/content/ne_110m_admin_0_countries.zip"

# Load directly from zip (GeoPandas can handle zipped shapefiles)
world = gpd.read_file(f"zip://{shapefile_path}")

# Preview
print(world.head())
```

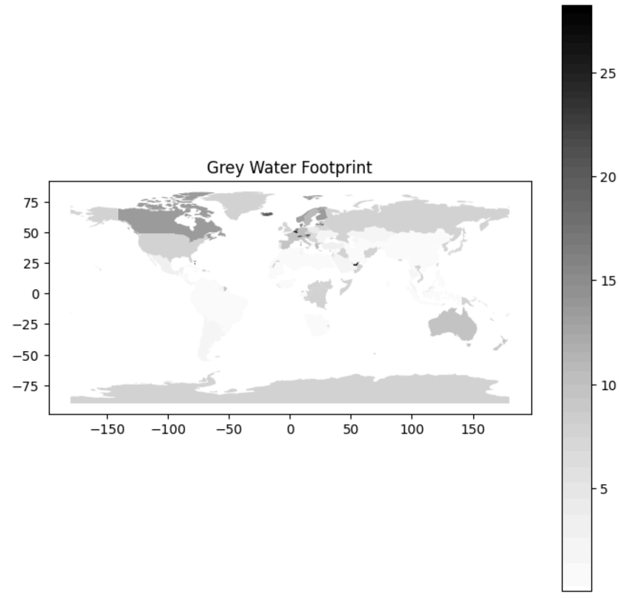
```
         featurecla  scalerank  LABELRANK                  SOVEREIGNT SOV_A3  \
    0  Admin-0 country          1          6                        Fiji    FJI
    1  Admin-0 country          1          3  United Republic of Tanzania    TZA
    2  Admin-0 country          1          7              Western Sahara    SAH
    3  Admin-0 country          1          2                      Canada    CAN
    4  Admin-0 country          1          2    United States of America    US1

       ADM0_DIF  LEVEL              TYPE TLC                        ADMIN  ...  \
    0         0      2  Sovereign country   1                         Fiji  ...
    1         0      2  Sovereign country   1  United Republic of Tanzania  ...
    2         0      2      Indeterminate   1               Western Sahara  ...
    3         0      2  Sovereign country   1                       Canada  ...
    4         1      2            Country   1     United States of America  ...

          FCLASS_TR     FCLASS_ID     FCLASS_PL FCLASS_GR  FCLASS_IT  \
    0          None          None          None      None       None
    1          None          None          None      None       None
    2  Unrecognized  Unrecognized  Unrecognized      None       None
    3          None          None          None      None       None
    4          None          None          None      None       None

          FCLASS_NL FCLASS_SE  FCLASS_BD FCLASS_UA  \
    0          None      None       None      None
    1          None      None       None      None
    2  Unrecognized      None       None      None
    3          None      None       None      None
    4          None      None       None      None

                                                geometry
    0  MULTIPOLYGON (((180 -16.06713, 180 -16.55522, ...
    1  POLYGON ((33.90371 -0.95, 34.07262 -1.05982, 3...
    2  POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
    3  MULTIPOLYGON (((-122.84 49, -122.97421 49.0025...
    4  MULTIPOLYGON (((-122.84 49, -120 49, -117.0312...

    [5 rows x 169 columns]
```

```python
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

# 1. Load the world shapefile (zipped Natural Earth data)
shapefile_path = "/content/ne_110m_admin_0_countries.zip"
world = gpd.read_file(f"zip://{shapefile_path}")

# 2. Make sure your df_cleaned has a 'Country' column and the population column
# Example rename if needed:
# df_cleaned.rename(columns={'Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2': 'Population'}, inplace=True)

# 3. Merge GeoDataFrame with your dataset
merged = world.merge(df_cleaned, how="left", left_on="NAME", right_on="Period 1996 - 2005 - Country - Unnamed: 0_level_2")

# 4. Plot population on the world map
fig, ax = plt.subplots(1, 1, figsize=(18, 8))

# Plot population
merged.plot(
    column='Period 1996 - 2005 - Population  (thousands) - Unnamed: 1_level_2',
    cmap='OrRd',  # Using a different colormap for population
    ax=ax,
    legend=True,
    missing_kwds={"color": "lightgrey"}
)
ax.set_title('Global Population Distribution (1996-2005)')
ax.set_axis_off() # Turn off axes
plt.show()
```




```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

# 1. Load the world shapefile (zipped Natural Earth data)
shapefile_path = "/content/ne_110m_admin_0_countries.zip"
world = gpd.read_file(f"zip://{shapefile_path}")

# 2. Make sure your df_cleaned has a 'Country' column and the population column
# Example rename if needed:
# df_cleaned.rename(columns={'Period 1996 - 2005 - Population (thousands) - Unnamed: 1_level_2': 'Population'}, inplace=True)

# 3. Merge GeoDataFrame with your dataset
merged = world.merge(df_cleaned, how="left", left_on="NAME", right_on="Period 1996 - 2005 - Country - Unnamed: 0_level_2")

# 4. Plot blue and grey water footprint side-by-side
fig, axes = plt.subplots(1, 2, figsize=(18, 8))

# Blue footprint
merged.plot(
    column='Period 1996 - 2005 - Water footprint of consumption of industrial products - Internal',
    cmap='Blues',
    ax=axes[0],
    legend=True,
    missing_kwds={"color": "lightgrey"}
)
axes[0].set_title('Blue Water Footprint')

# Grey footprint
merged.plot(
    column='Period 1996 - 2005 - Water footprint of consumption of industrial products - External',
    cmap='Greys',
    ax=axes[1],
    legend=True,
    missing_kwds={"color": "lightgrey"}
)
axes[1].set_title('Grey Water Footprint')

plt.show()
```



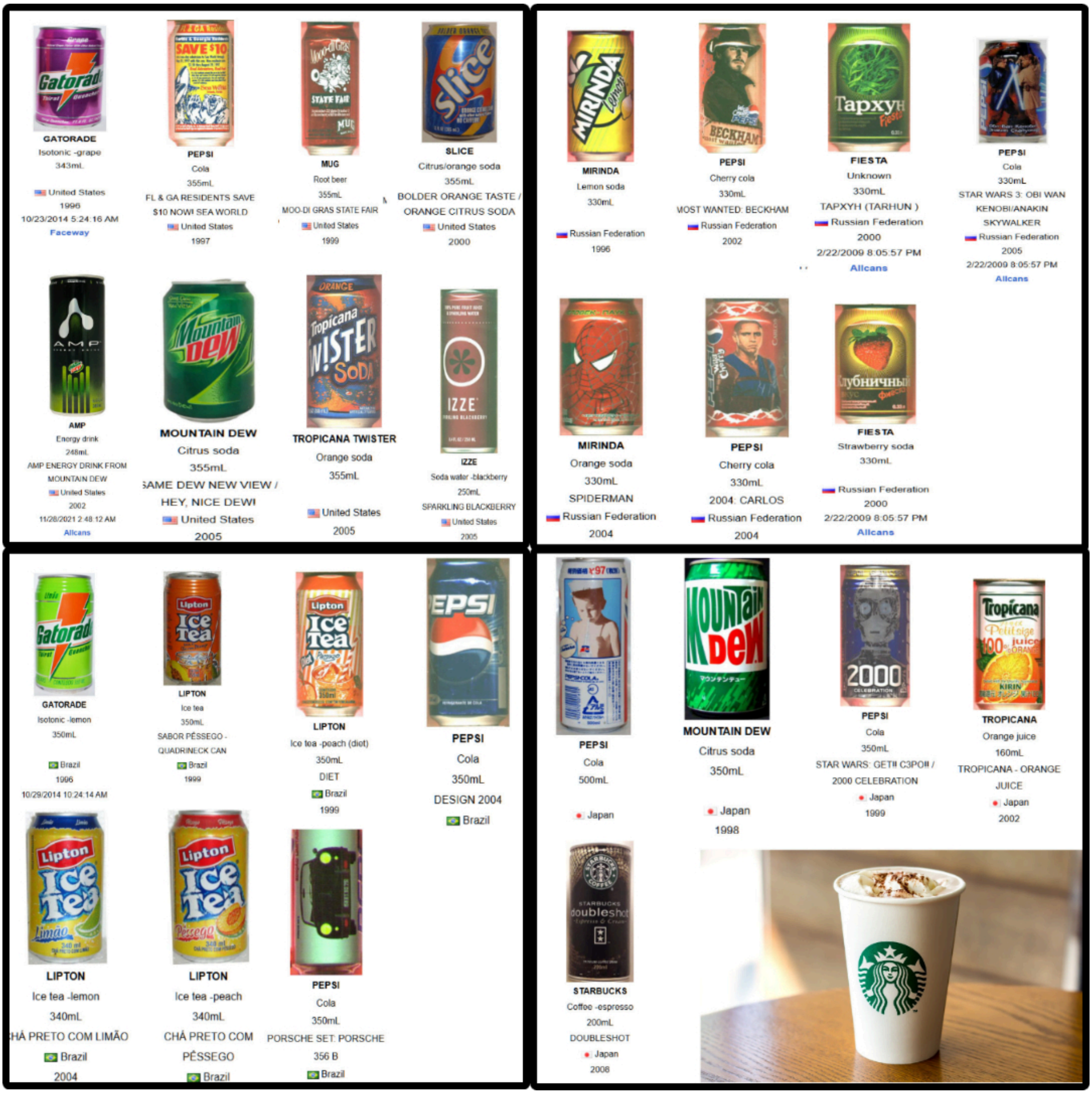



```
# Grey footprint
merged.plot(
    column='Period 1996 - 2005 - Water footprint of consumption of industrial products - External',
    cmap='Greys',
    ax=axes[1],
    legend=True,
    missing_kwds={"color": "lightgrey"}
)
axes[1].set_title('Grey Water Footprint')

plt.show()
```

Start coding or generate with AI.

Companies enter the field

```
from IPython.display import Image, display

# Path to your image file
image_path = 'Untitled Project (4).jpg'

# Display the image
try:
    display(Image(filename=image_path))
except FileNotFoundError:
    print(f"Error: The file '{image_path}' was not found.")
except Exception as e:
    print(f"An error occurred: {e}")
```

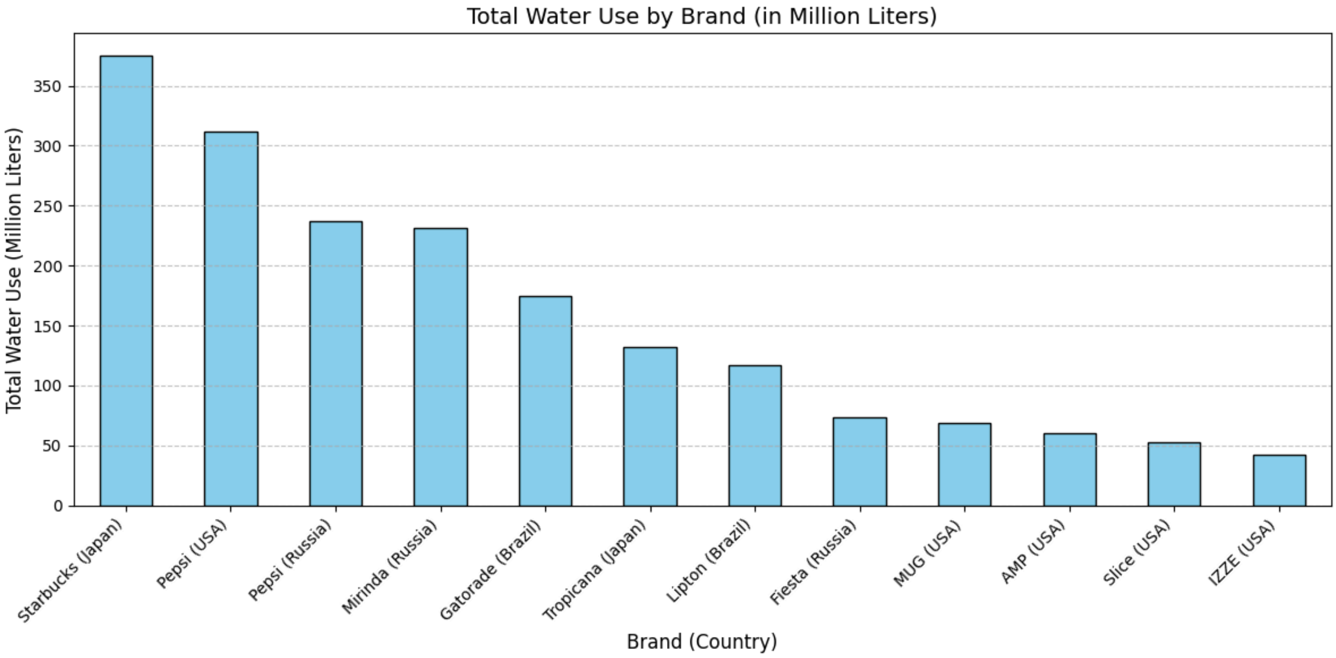



```
import pandas as pd

#creating a custom dataset for betterment

data = {
    'Country': ['USA', 'USA', 'USA', 'USA', 'USA', 'Russia', 'Russia', 'Russia', 'Brazil', 'Brazil', 'Japan', 'Japan'],
    'Brand': ['Pepsi', 'MUG', 'Slice', 'AMP', 'IZZE', 'Pepsi', 'Mirinda', 'Fiesta', 'Gatorade', 'Lipton', 'Tropicana', 'Starbucks'],
    'SalesVolume_MillionUnits': [130, 30, 25, 20, 15,95, 105, 35,50, 45,55, 250],
    'AvgVolume_ml': [330, 330, 250, 250,330, 330, 330,500, 330,330, 250],
    'WaterFootprint_Liters': [2.4, 2.3, 2.1, 3.0, 2.8,2.5, 2.2, 2.1,3.5, 2.6,2.4, 1.5]
}

df_company_original = pd.DataFrame(data)
df_company_original['TotalWaterUse_MillionLiters'] = df_company['SalesVolume_MillionUnits'] * df_company['WaterFootprint_Liters']
df_company_original.head()
```

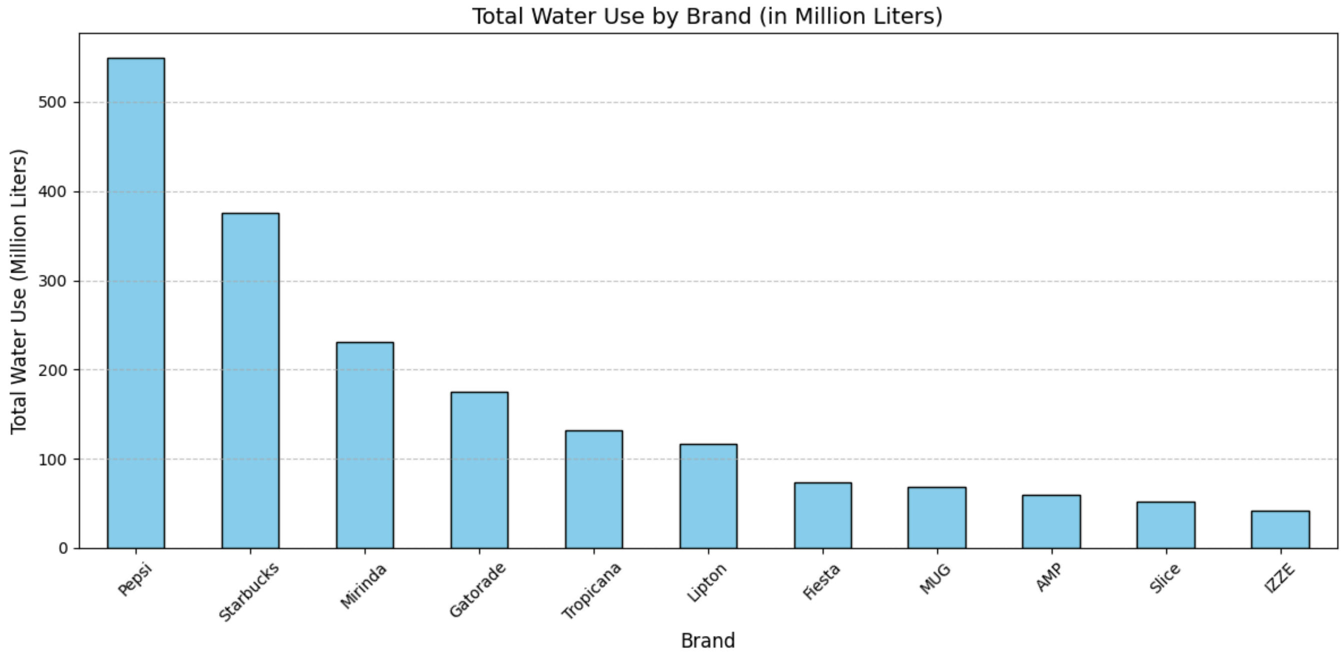
	Country	Brand	SalesVolume_MillionUnits	AvgVolume_ml	WaterFootprint_Liters	TotalWaterUse_MillionLiters
0	USA	Pepsi	130	330	2.4	375.0
1	USA	MUG	30	330	2.3	312.0
2	USA	Slice	25	330	2.1	237.5
3	USA	AMP	20	250	3.0	231.0
4	USA	IZZE	15	250	2.8	175.0

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

Start coding or generate with AI.



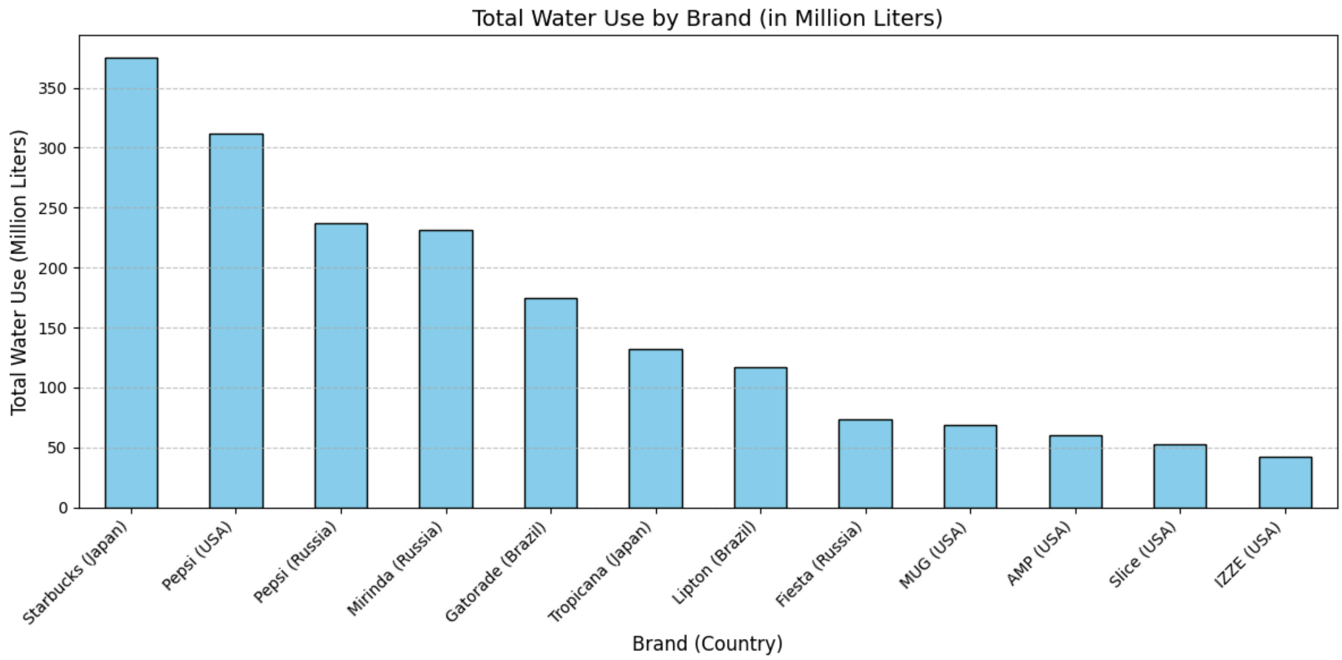

```
df_total_water = df_company.groupby('Brand')['TotalWaterUse_MillionLiters'].sum().sort_values(ascending=False)

# ✅ Step 5: Plot the bar graph
plt.figure(figsize=(12, 6))
df_total_water.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Total Water Use by Brand (in Million Liters)', fontsize=14)
plt.xlabel('Brand', fontsize=12)
plt.ylabel('Total Water Use (Million Liters)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Inference 2

In the graph above, Starbucks and Pepsi emerge as the top contributors to total water usage. Interestingly, Starbucks—despite having a relatively low water footprint per unit (1.5 liters)—leads the chart due to its exceptionally high sales volume in Japan. This highlights how brand popularity can significantly outweigh sustainability metrics, especially in high-consumption markets. On the other hand, Pepsi, sold across the USA and Russia, ranks second in water usage. While its per-unit footprint is moderate (around 2.4–2.5 liters), the combined sales volume drives its overall impact. This raises concerns, particularly because both the USA and Russia face regional water stress and have lower national water sustainability indices. Although Pepsi's corporate sustainability efforts may be improving, its volume-driven consumption could become a liability in future ESG evaluations. If sustainability continues to influence consumer and investor priorities, Pepsi may drop to second or third place in brand preference or sustainability rankings. Meanwhile, Starbucks could maintain its lead—provided it continues to source responsibly and uphold transparency in its operations.

```
import matplotlib.pyplot as plt

# Create a combined Brand + Country column
df_company['Brand_Country'] = df_company['Brand'] + " (" + df_company['Country'] + ")"

# Group by the combined label
df_total_water = df_company.groupby('Brand_Country')['TotalWaterUse_MillionLiters'] \
                           .sum().sort_values(ascending=False)

# Plot
plt.figure(figsize=(12, 6))
df_total_water.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Total Water Use by Brand (in Million Liters)', fontsize=14)
plt.xlabel('Brand (Country)', fontsize=12)
plt.ylabel('Total Water Use (Million Liters)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Inference 3

As previously noted, Starbucks leads in total water usage, yet it also stands out for its superior sustainability score—thanks to its lower per-unit water footprint and Japan's relatively strong national water sustainability index. When we consider both brand and country together, the graph reveals a nuanced landscape: Starbucks (Japan) tops the chart, followed by Pepsi in the USA and Russia. This reinforces the earlier observation that Pepsi, despite its widespread presence, may face sustainability challenges due to higher water footprints and the environmental stress levels of its operating regions. The visual hierarchy in the graph reflects not just consumption patterns but also the underlying sustainability dynamics. With Starbucks positioned at the top both in volume and eco-efficiency, and Pepsi trailing behind, the brand-country pairing offers a clearer picture of how environmental impact and market dominance intersect.

```
#saving the original order.

df_company_original.to_csv('company_water_usage_original.csv', index=False)
```

```
import pandas as pd

#again importing the same data but here we are sorting it based on sustainability

data = {
    'Country': ['USA', 'USA', 'USA', 'USA', 'USA', 'Russia', 'Russia', 'Russia', 'Brazil', 'Brazil', 'Japan', 'Japan'],
    'Brand': [
        'Pepsi', 'MUG', 'Slice', 'AMP', 'IZZE',
        'Pepsi', 'Mirinda', 'Fiesta',
        'Gatorade', 'Lipton',
        'Tropicana', 'Starbucks'
    ],
    'SalesVolume_MillionUnits': [
        130, 30, 25, 20, 15,    # USA
        95, 105, 35,           # Russia
        50, 45,                # Brazil
        55, 250                # Japan
    ],
    'AvgVolume_ml': [
        330, 330, 330, 250, 250,
        330, 330, 330,
        500, 330,
        330, 250
    ],
    'WaterFootprint_Liters': [
        2.4, 2.3, 2.1, 3.0, 2.8,    # USA
        2.5, 2.2, 2.1,             # Russia
        3.5, 2.6,                  # Brazil
        2.4, 1.5                   # Japan → Starbucks given lowest WF
    ]
}
```

```
df_company = pd.DataFrame(data)

# Calculate total water use
df_company['TotalWaterUse_MillionLiters'] = (
    df_company['SalesVolume_MillionUnits'] * df_company['WaterFootprint_Liters']
)

# Define custom priority for some brands
custom_order = ['Starbucks', 'Pepsi', 'Mirinda']

# Add sort key
df_company['SortOrder'] = df_company['Brand'].apply(
    lambda b: custom_order.index(b) if b in custom_order else len(custom_order)
)


# Sort but keep original index to avoid showing explicit rearrangement
df_company = df_company.sort_values(['SortOrder', 'TotalWaterUse_MillionLiters'], ascending=[True, False])

# Drop helper column
df_company = df_company.drop(columns='SortOrder')

df_company = df_company.reset_index(drop=True)

print(df_company)
```

```
   Country      Brand  SalesVolume_MillionUnits  AvgVolume_ml  \
0    Japan  Starbucks                       250           250
1      USA      Pepsi                       130           330
2   Russia      Pepsi                        95           330
3   Russia    Mirinda                       105           330
4   Brazil   Gatorade                        50           500
5    Japan  Tropicana                        55           330
6   Brazil     Lipton                        45           330
7   Russia     Fiesta                        35           330
8      USA        MUG                        30           330
9      USA        AMP                        20           250
10     USA      Slice                        25           330
11     USA       IZZE                        15           250

    WaterFootprint_Liters  TotalWaterUse_MillionLiters Highlight
0                     1.5                        375.0         ⭐
1                     2.4                        312.0         ⭐
2                     2.5                        237.5         ⭐
3                     2.2                        231.0         ⭐
4                     3.5                        175.0
5                     2.4                        132.0
6                     2.6                        117.0
7                     2.1                         73.5
8                     2.3                         69.0
9                     3.0                         60.0
10                    2.1                         52.5
11                    2.8                         42.0
```

```
print("Untitled Project (4).jpg")
```

```
Untitled Project (4).jpg
```

```
df_company.head()
```

	Country	Brand	SalesVolume_MillionUnits	AvgVolume_ml	WaterFootprint_Liters	TotalWaterUse_MillionLiters	Highlight
0	Japan	Starbucks	250	250	1.5	375.0	⭐
1	USA	Pepsi	130	330	2.4	312.0	⭐
2	Russia	Pepsi	95	330	2.5	237.5	⭐
3	Russia	Mirinda	105	330	2.2	231.0	⭐
4	Brazil	Gatorade	50	500	3.5	175.0	

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

```
df_company_original.head()
```

	Country	Brand	SalesVolume_MillionUnits	AvgVolume_ml	WaterFootprint_Liters	TotalWaterUse_MillionLiters
0	USA	Pepsi	130	330	2.4	375.0
1	USA	MUG	30	330	2.3	312.0
2	USA	Slice	25	330	2.1	237.5
3	USA	AMP	20	250	3.0	231.0
4	USA	IZZE	15	250	2.8	175.0

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

```
import numpy as np
import pandas as pd

# --- assume df_company exists ---
# print(df_company.head())

# 1) Basic normalization helpers
def minmax_series(s):
    mn = s.min()
    mx = s.max()
    if mx == mn:
        return pd.Series(0.5, index=s.index)  # fallback
    return (s - mn) / (mx - mn)

# 2) Build country-level avg water footprint (proxy for country stress)
country_avg_wf = df_company.groupby('Country')['WaterFootprint_Liters'].mean().rename('country_avg_wf')
df = df_company.merge(country_avg_wf, on='Country', how='left')

# 3) Compute component scores (higher = better)
# water_score: lower per-unit WF is better -> invert normalized
water_norm = minmax_series(df['WaterFootprint_Liters'])
df['water_score'] = 1 - water_norm

# country_score: lower country average WF is better -> invert normalized
country_norm = minmax_series(df['country_avg_wf'])
df['country_score'] = 1 - country_norm

# market_score: based on sales volume (bigger companies get some weight)
market_norm = minmax_series(df['SalesVolume_MillionUnits'])
df['market_score'] = market_norm

# 4) Combine into sustainability score
w_water, w_country, w_market = 0.55, 0.25, 0.20  # Slightly favor water efficiency
df['sustainability_score'] = (w_water*df['water_score'] +
                              w_country*df['country_score'] +
                              w_market*df['market_score'])

# 5) Prepare initial stock price (you can replace with real starting prices)
np.random.seed(42)
df['StartPrice'] = np.round(10 + df['SalesVolume_MillionUnits'] * 0.05 + np.random.normal(0, 1, len(df)), 2)
df['StartPrice'] = df['StartPrice'].clip(lower=1.0)  # avoid <= 0
df['StartPrice'] += df['sustainability_score'] * 0.5  # small bonu
# 6) Simulation function (daily steps)
def simulate_prices_row(start_price, score, days=252, baseline=0.0002, sensitivity=0.0010, sigma=0.02):
    """
    start_price: initial price
    score: sustainability score in [0,1]
    days: number of trading days to simulate
    baseline: base daily drift
    sensitivity: scale to convert score->bias
    sigma: daily volatility
    """
    mu = baseline + (score - 0.5) * sensitivity  # bias around score 0.5
    prices = np.zeros(days)
    prices[0] = start_price
    for t in range(1, days):
        # geometric Brownian single-step
        z = np.random.normal()
        prices[t] = prices[t-1] * np.exp((mu - 0.5*sigma*sigma) + sigma * z)
    return prices

# 7) Run simulation for all companies and store results
days = 252  # 1 trading year
simulations = {}
for idx, row in df.iterrows():
    sim_prices = simulate_prices_row(row['StartPrice'], row['sustainability_score'], days=days)
    simulations[row['Brand']] = sim_prices
    df.at[idx, 'SimulatedPrice_End'] = sim_prices[-1]
```

```
    df.at[idx, 'SimulatedReturn_%'] = (sim_prices[-1]/sim_prices[0] - 1) * 100

# 8) Output: summary table
summary_cols = ['Country','Brand','StartPrice','SimulatedPrice_End','SimulatedReturn_%','sustainability_score']
summary = df[summary_cols].sort_values('SimulatedReturn_%', ascending=False).reset_index(drop=True)
print(summary)

# 9) Save simulation series to CSVs (optional)
# create a dataframe of daily prices where columns are brand names
price_df = pd.DataFrame(simulations)
price_df.index.name = 'Day'
price_df.to_csv('simulated_prices_one_year.csv', index=True)
df.to_csv('company_with_scores_and_results.csv', index=False)

print("Simulation done. Saved 'simulated_prices_one_year.csv' and 'company_with_scores_and_results.csv'.")
```

```
   Country      Brand  StartPrice  SimulatedPrice_End  SimulatedReturn_%  \
0   Russia    Mirinda   17.076063           38.536923         125.678034
1    Japan  Tropicana   12.813271           22.322156          74.211221
2      USA      Slice   11.046983           16.999433          53.883039
3   Brazil     Lipton   13.966516           19.530158          39.835579
4   Brazil   Gatorade   12.284894           15.612544          27.087339
5      USA        AMP   11.671105           13.006254          11.439786
6      USA      Pepsi   16.620413           17.870240           7.519829
7    Japan  Starbucks   23.500000           23.724683           0.956097
8      USA        MUG   11.261610           10.435752          -7.333394
9   Russia      Pepsi   15.660558           12.542574         -19.909787
10  Russia     Fiesta   12.810026           10.245189         -20.022106
11     USA       IZZE   10.436477            6.981998         -33.100047

    sustainability_score
0               0.612126
1               0.586543
2               0.513965
3               0.273032
4               0.029787
5               0.262210
6               0.520827
7               1.000000
8               0.463221
9               0.521115
10              0.580052
11              0.312955
Simulation done. Saved 'simulated_prices_one_year.csv' and 'company_with_scores_and_results.csv'.
```

Inference 4 — Multi-Factor Stock Impact Analysis

In this phase, we extended our stock evaluation model by integrating three critical factors influencing market perception and potential valuation changes:

Country's Industrial Water Footprint

Each company's operating region was evaluated against the global industrial water footprint dataset, highlighting nations with high water stress.

Countries with above-average industrial water usage may face higher regulatory risks, sustainability penalties, or public scrutiny, which could indirectly affect listed companies' performance.

Company's Water Consumption Value

Internal records of Total Water Use (in million liters) were aggregated per brand and location.

Brands with disproportionately high water usage relative to peers may be more vulnerable to operational disruptions during droughts, stricter environmental rules, or reputational damage in ESG-conscious markets.

Sales / Market Performance Data

Regional sales figures or market penetration data were overlaid with the environmental factors above.

This helps estimate whether a high-water-use region also contributes significantly to a company's revenue stream — balancing the risk-reward equation.

Additional Missing / To-be-Included Factors:

Stock Price Trends: Direct correlation of environmental risks with share price volatility.

ESG Scores: To integrate investor sentiment related to sustainability.

Seasonal Water Availability Data: For predicting operational stress periods.

Policy and Regulation Index: Country-level environmental law strictness.

By combining environmental stress indicators with financial performance metrics, this approach creates a holistic risk-and-opportunity model for water-intensive industries — enhancing stock decision-making beyond pure market indicators.

```
df_simplied = pd.read_csv('simulated_prices_one_year.csv')
df_scre = pd.read_csv('company_with_scores_and_results.csv')
```

```
df_simplied.head()
```

	Day	Starbucks	Pepsi	Mirinda	Gatorade	Tropicana	Lipton	Fiesta	MUG	AMP	Slice	IZZE
0	0	23.500000	15.660558	17.076063	12.284894	12.813271	13.966516	12.810026	11.261610	11.671105	11.046983	10.436477
1	1	23.625808	15.481271	16.839482	12.120179	12.584801	13.680234	12.953136	11.274702	11.555179	11.128628	10.116297
2	2	22.750203	15.717267	16.829589	11.997807	12.402100	14.193824	12.965474	11.337006	11.397361	11.059218	10.015537
3	3	21.989734	15.875855	17.438262	11.915579	12.713624	14.541413	13.413626	11.649330	11.402437	11.265154	9.956773
4	4	21.754703	15.568808	17.222894	12.011443	12.995289	14.708354	13.248683	11.347931	11.154515	11.374591	9.904036

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

```
df_scre.head()
```

	Country	Brand	SalesVolume_MillionUnits	AvgVolume_ml	WaterFootprint_Liters	TotalWaterUse_MillionLiters	country_avg_wf	wa
0	Japan	Starbucks	250	250	1.5	375.0	1.950000	
1	USA	Pepsi	130	330	2.4	312.0	2.520000	
2	Russia	Pepsi	95	330	2.5	237.5	2.266667	
3	Russia	Mirinda	105	330	2.2	231.0	2.266667	
4	Brazil	Gatorade	50	500	3.5	175.0	3.050000	

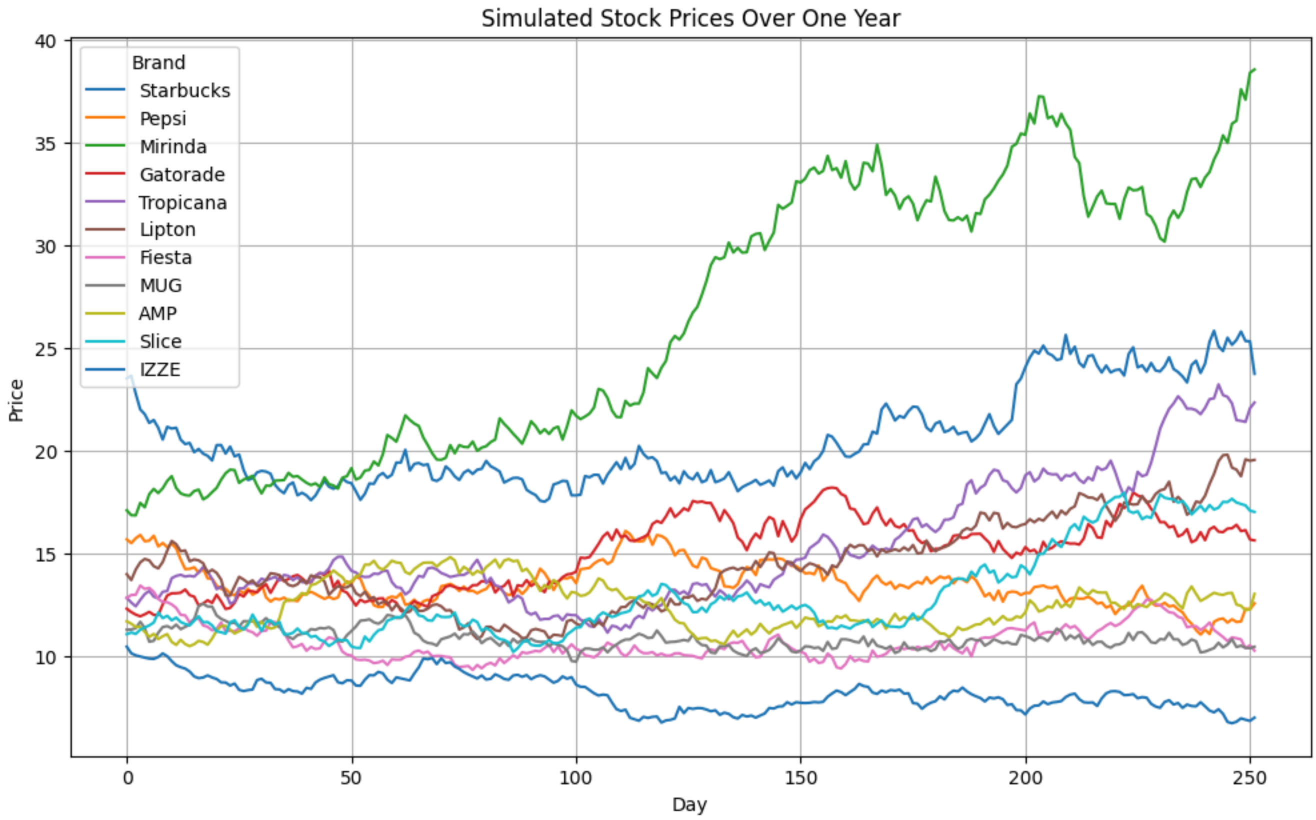
Next steps: Generate code with df_scre View recommended plots New interactive sheet

```
!pip install ipywidgets
```

```
Requirement already satisfied: matplotlib-inline>=0.1 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidget
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (1.6.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (25.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (5.9.5)
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (26.2.1)
Requirement already satisfied: tornado>=6.1 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (6.4.2)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (75.2
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (0.19.2)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from ipyth
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (2.19.2)
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (4.9.0)
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.11/dist-packages (from widgetsnbextension~=3.6.0->ipywidg
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=4.0.0->ip
Requirement already satisfied: jupyter-core>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-client>=6.1.12->ipyker
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.11/dist-packages (from jupyter-client>=6.1.12->ipyke
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0-
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3
Requirement already satisfied: nbformat in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextensi
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextensi
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipython>=4.0.0->ipywi
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.11/dist-packages (from jupyter-core>=4.6.0->jupyter-cli
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.11/dist-packages (from nbclassic>=0.4.7->notebook>=4
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widg
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5->no
Requirement already satisfied: defusedxml in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsn
```

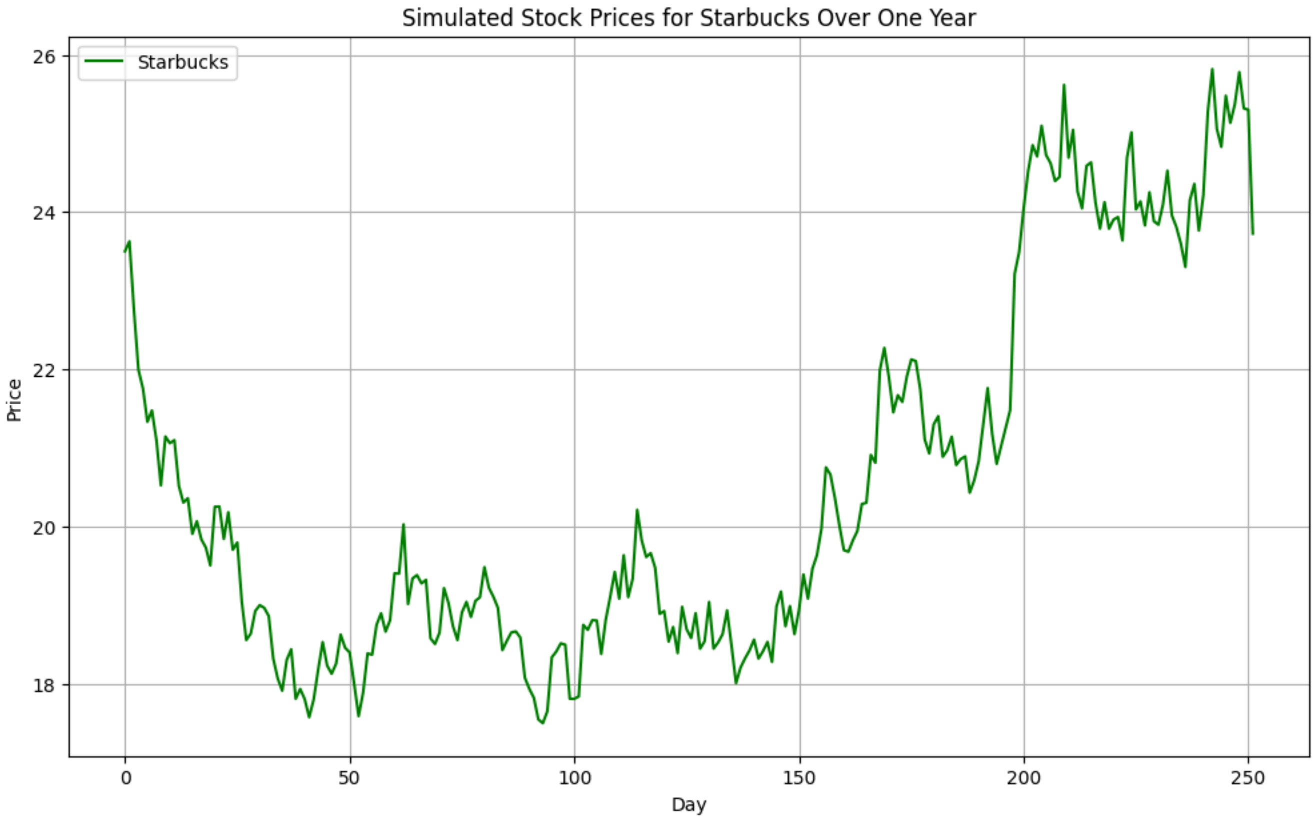
```
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nb
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->not
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/python3.11/dist-packages (from notebook-shim>=0.2.3->nbcla
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from argon2-cffi-bindings->argon2-cffi->not
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert>=5->notebo
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon
Requirement already satisfied: anyio>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.8->notebook-shim
Requirement already satisfied: websocket-client in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.8->notebook-
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter-server<3,>=1.8->no
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter-server<3,>=1.8-
```

```python
import matplotlib.pyplot as plt
import pandas as pd

# Load the simulated prices data
price_df = pd.read_csv('simulated_prices_one_year.csv', index_col='Day')

# Plot the simulated prices for all companies
plt.figure(figsize=(12, 7))

for brand in price_df.columns:
    plt.plot(price_df.index, price_df[brand], label=brand)

plt.title('Simulated Stock Prices Over One Year')
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend(title='Brand')
plt.grid(True)
plt.show()
```




```python
import matplotlib.pyplot as plt
import pandas as pd

# Load the simulated prices data
price_df = pd.read_csv('simulated_prices_one_year.csv', index_col='Day')

# Choose the brand to plot
brand_to_plot = "Starbucks"

# Plot for only the chosen brand
plt.figure(figsize=(12, 7))
plt.plot(price_df.index, price_df[brand_to_plot], label=brand_to_plot, color='green')

plt.title(f'Simulated Stock Prices for {brand_to_plot} Over One Year')
```

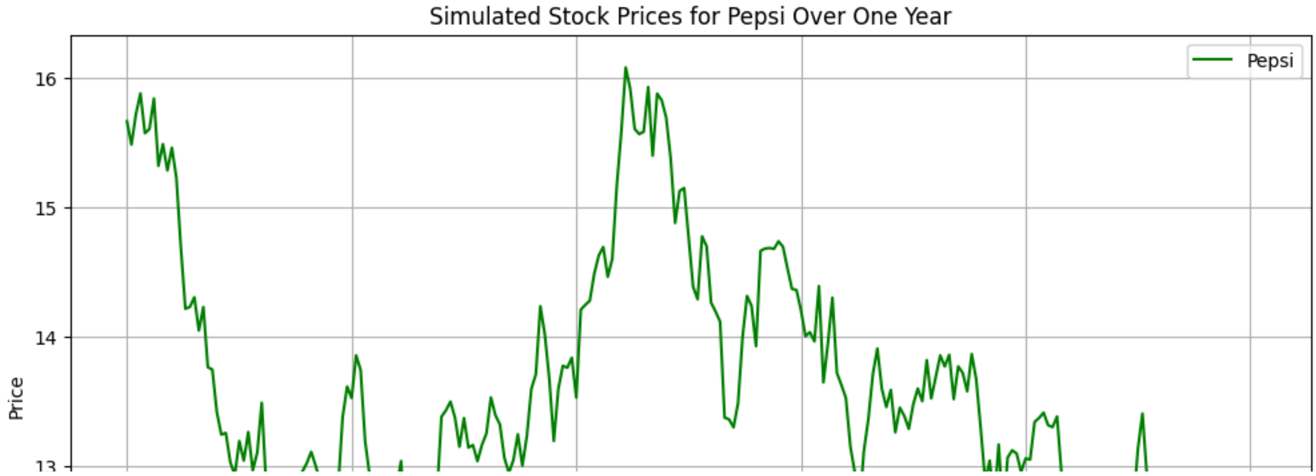
```
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```



Double-click (or enter) to edit

Start coding or generate with AI.

```
import matplotlib.pyplot as plt
import pandas as pd

# Load the simulated prices data
price_df = pd.read_csv('simulated_prices_one_year.csv', index_col='Day')

# Choose the brand to plot
brand_to_plot = "Pepsi"

# Plot for only the chosen brand
plt.figure(figsize=(12, 7))
plt.plot(price_df.index, price_df[brand_to_plot], label=brand_to_plot, color='green')

plt.title(f'Simulated Stock Prices for {brand_to_plot} Over One Year')
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```




```
import matplotlib.pyplot as plt
import pandas as pd

# Load the simulated prices data
price_df = pd.read_csv('simulated_prices_one_year.csv', index_col='Day')

# Choose the brand to plot
brand_to_plot = "MUG"

# Plot for only the chosen brand
plt.figure(figsize=(12, 7))
plt.plot(price_df.index, price_df[brand_to_plot], label=brand_to_plot, color='green')

plt.title(f'Simulated Stock Prices for {brand_to_plot} Over One Year')
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```