

```
In [ ]: # Libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import plotly.express as px
import ipywidgets as widgets
from IPython.display import display
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
import mplcursors
```

```
In [ ]: data = pd.read_csv(r"C:\Users\Praveen T\Desktop\ass\34_years_world_export_import_data
```

```
In [ ]: data.head(10)
```

```
Out[ ]:
```

	Partner Name	Year	Export (US\$ Thousand)	Import (US\$ Thousand)	Export Product Share (%)	Import Product Share (%)	Revealed comparative advantage	World Growth (%)	Country Growth (%)	Sir Ave
0	Aruba	1988	3498.10	328.49	100.0	100	NaN	NaN	NaN	NaN
1	Afghanistan	1988	213030.40	54459.52	100.0	100	NaN	NaN	NaN	NaN
2	Angola	1988	375527.89	370702.76	100.0	100	NaN	NaN	NaN	NaN
3	Anguila	1988	366.98	4.00	100.0	100	NaN	NaN	NaN	NaN
4	Albania	1988	30103.56	47709.30	100.0	100	NaN	NaN	NaN	NaN
5	Andorra	1988	67924.46	3284.01	100.0	100	NaN	NaN	NaN	NaN
6	Netherlands Antilles	1988	104759.21	24964.14	100.0	100	NaN	NaN	NaN	NaN
7	United Arab Emirates	1988	2945350.25	7091823.87	100.0	100	NaN	NaN	NaN	NaN
8	Argentina	1988	1136421.71	1928596.45	100.0	100	NaN	NaN	NaN	NaN
9	Antigua and Barbuda	1988	14406.52	2173.80	100.0	100	NaN	NaN	NaN	NaN

10 rows × 33 columns

```
In [ ]: # number of columns present in the dataset.
len(data.columns)
```

```
Out[ ]: 33
```

```
In [ ]: # number of rows present in the dataset.
len(data)
```

```
Out[ ]: 8096
```

```
In [ ]: # Descriptive statistics  
data.describe()
```

Out[]:

	Year	Export (US\$ Thousand)	Import (US\$ Thousand)	Export Product Share (%)	Import Product Share (%)	Revealed comparative advantage	World Growth (%)	Co Growth
count	8096.000000	8.096000e+03	8.096000e+03	8076.0	8096.0	4712.0	4410.000000	4410.0
mean	2004.908226	1.421192e+08	1.305216e+08	100.0	100.0	1.0	3.986016	3.9
std	9.707831	9.928417e+08	9.073802e+08	0.0	0.0	0.0	10.004221	10.0
min	1988.000000	0.000000e+00	3.000000e-02	100.0	100.0	1.0	-62.280000	-62.2
25%	1997.000000	4.274264e+05	1.601335e+05	100.0	100.0	1.0	-1.437500	-1.4
50%	2005.000000	3.719683e+06	2.053967e+06	100.0	100.0	1.0	3.830000	3.8
75%	2013.000000	2.585514e+07	2.102937e+07	100.0	100.0	1.0	9.407500	9.4
max	2021.000000	2.422743e+10	2.193121e+10	100.0	100.0	1.0	174.000000	174.0

8 rows × 32 columns

Data Preprocessing:

```
In [ ]: # finding number of missing values  
data.isna().sum()
```

```

Out[ ]: Partner Name          0
         Year              0
         Export (US$ Thousand) 0
         Import (US$ Thousand) 0
         Export Product Share (%) 20
         Import Product Share (%) 0
         Revealed comparative advantage 3384
         World Growth (%)      3686
         Country Growth (%)    3686
         AHS Simple Average (%) 16
         AHS Weighted Average (%) 16
         AHS Total Tariff Lines 16
         AHS Dutiable Tariff Lines Share (%) 16
         AHS Duty Free Tariff Lines Share (%) 16
         AHS Specific Tariff Lines Share (%) 16
         AHS AVE Tariff Lines Share (%) 16
         AHS MaxRate (%)        16
         AHS MinRate (%)        16
         AHS SpecificDuty Imports (US$ Thousand) 15
         AHS Dutiable Imports (US$ Thousand) 15
         AHS Duty Free Imports (US$ Thousand) 15
         MFN Simple Average (%) 15
         MFN Weighted Average (%) 15
         MFN Total Tariff Lines 15
         MFN Dutiable Tariff Lines Share (%) 15
         MFN Duty Free Tariff Lines Share (%) 15
         MFN Specific Tariff Lines Share (%) 16
         MFN AVE Tariff Lines Share (%) 16
         MFN MaxRate (%)        15
         MFN MinRate (%)        15
         MFN SpecificDuty Imports (US$ Thousand) 15
         MFN Dutiable Imports (US$ Thousand) 15
         MFN Duty Free Imports (US$ Thousand) 15
dtype: int64

```

```

In [ ]: import pandas as pd

# Step 1: Replace missing values with mean for constant values
constant_cols = ['AHS Simple Average (%)', 'AHS Weighted Average (%)', 'AHS Total Tariff Lines', 'AHS Dutiable Tariff Lines Share (%)', 'AHS Duty Free Tariff Lines Share (%)', 'AHS Specific Tariff Lines Share (%)', 'AHS AVE Tariff Lines Share (%)', 'AHS MaxRate (%)', 'AHS MinRate (%)', 'MFN Simple Average (%)', 'MFN Weighted Average (%)', 'MFN Total Tariff Lines', 'MFN Dutiable Tariff Lines Share (%)', 'MFN Duty Free Tariff Lines Share (%)', 'MFN Specific Tariff Lines Share (%)', 'MFN AVE Tariff Lines Share (%)', 'MFN MaxRate (%)', 'MFN MinRate (%)']

# Replace missing values in constant columns with mean
data[constant_cols] = data[constant_cols].fillna(data[constant_cols].mean())

# Step 2: Replace missing values with mean for each country's total
# For each country, find the mean and replace missing values with it
for country in data['Partner Name'].unique():
    country_data = data[data['Partner Name'] == country]
    country_mean = country_data.mean()
    data.loc[data['Partner Name'] == country] = country_data.fillna(country_mean)

# Step 3: Replace missing values in specific columns with the mean value for each country
specific_cols = ['Revealed comparative advantage', 'World Growth (%)', 'Country Growth (%)']
for col in specific_cols:
    country_means = data.groupby('Partner Name')[col].transform('mean')
    data[col].fillna(country_means, inplace=True)

```

```

# Step 4: Replace remaining missing values with 0 or overall average
remaining_null_cols = ['Revealed comparative advantage', 'World Growth (%)', 'Country'
for col in remaining_null_cols:
    # Replace missing values with 0 if possible, otherwise with overall average
    if data[col].isnull().sum() > 0:
        data[col].fillna(0, inplace=True)
    else:
        data[col].fillna(data[col].mean(), inplace=True)

# After performing these steps, any remaining missing values should be replaced with
# You can save the modified DataFrame to a new CSV file if needed
data.to_csv('filled_dataset.csv', index=False) # Replace 'filled_dataset.csv' with d

```

In []: `data.isna().sum()`

Out[]:

Partner Name	0
Year	0
Export (US\$ Thousand)	0
Import (US\$ Thousand)	0
Export Product Share (%)	0
Import Product Share (%)	0
Revealed comparative advantage	0
World Growth (%)	0
Country Growth (%)	0
AHS Simple Average (%)	0
AHS Weighted Average (%)	0
AHS Total Tariff Lines	0
AHS Dutiable Tariff Lines Share (%)	0
AHS Duty Free Tariff Lines Share (%)	0
AHS Specific Tariff Lines Share (%)	0
AHS AVE Tariff Lines Share (%)	0
AHS MaxRate (%)	0
AHS MinRate (%)	0
AHS SpecificDuty Imports (US\$ Thousand)	0
AHS Dutiable Imports (US\$ Thousand)	0
AHS Duty Free Imports (US\$ Thousand)	0
MFN Simple Average (%)	0
MFN Weighted Average (%)	0
MFN Total Tariff Lines	0
MFN Dutiable Tariff Lines Share (%)	0
MFN Duty Free Tariff Lines Share (%)	0
MFN Specific Tariff Lines Share (%)	0
MFN AVE Tariff Lines Share (%)	0
MFN MaxRate (%)	0
MFN MinRate (%)	0
MFN SpecificDuty Imports (US\$ Thousand)	0
MFN Dutiable Imports (US\$ Thousand)	0
MFN Duty Free Imports (US\$ Thousand)	0

dtype: int64

Outlier Detection

In []:

```

import numpy as np
from sklearn.neighbors import LocalOutlierFactor

numeric_columns = data.select_dtypes(include=np.number)

# Specify the number of neighbors
n_neighbors = 20

# Initialize the Local Outlier Factor (LOF) model

```

```
lof_model = LocalOutlierFactor(n_neighbors=n_neighbors, contamination='auto')

# Fit the model and predict outlier scores
outlier_scores = lof_model.fit_predict(numeric_columns)

# The 'outlier_scores' will contain 1 for inliers, -1 for outliers
# Convert -1 to True for easier filtering
outliers_mask = outlier_scores == -1

# Filter out the outliers
outliers = data[outliers_mask]

# Print or do further analysis with the outliers
print("Detected outliers:")
print(outliers)
```

Detected outliers:

	Partner Name	Year	Export (US\$ Thousand)	Import (US\$ Thousand)	\
0	Aruba	1988	3.498100e+03	3.284900e+02	
9	Antigua and Barbuda	1988	1.440652e+04	2.173800e+03	
18	Bahamas, The	1988	3.565688e+05	5.850492e+04	
25	Brunei	1988	1.984813e+05	1.523443e+06	
33	China	1988	1.357527e+07	1.425127e+07	
...	
8009	Nauru	2021	1.100976e+05	2.061846e+05	
8021	Korea, Dem. Rep.	2021	2.869447e+05	1.917634e+05	
8041	Special Categories	2021	8.549076e+07	8.548856e+07	
8055	Chad	2021	1.320752e+06	2.982370e+06	
8081	World	2021	2.422743e+10	2.193121e+10	

	Export Product Share (%)	Import Product Share (%)	\
0	100.0	100	
9	100.0	100	
18	100.0	100	
25	100.0	100	
33	100.0	100	
...	
8009	100.0	100	
8021	100.0	100	
8041	100.0	100	
8055	100.0	100	
8081	100.0	100	

	Revealed comparative advantage	World Growth (%)	Country Growth (%)	\
0	1.0	0.951429	0.951429	
9	1.0	-0.033077	-0.033077	
18	1.0	0.916522	0.916522	
25	1.0	4.939524	4.939524	
33	1.0	6.453448	6.453448	
...	
8009	0.0	0.000000	0.000000	
8021	0.0	0.000000	0.000000	
8041	0.0	0.000000	0.000000	
8055	1.0	0.000000	0.000000	
8081	1.0	12.590000	12.590000	

	AHS Simple Average (%)	...	MFN Total Tariff Lines	\
0	2.80	...	1152.0	
9	1.17	...	2274.0	
18	1.62	...	4526.0	
25	7.10	...	7597.0	
33	8.72	...	78322.0	
...	
8009	5.23	...	182968.0	
8021	8.07	...	431830.0	
8041	9.09	...	794548.0	
8055	4.55	...	220902.0	
8081	5.36	...	1535059.0	

	MFN Dutiable Tariff Lines Share (%)	\
0	63.54	
9	67.68	
18	69.24	
25	74.45	
33	67.57	
...	...	
8009	71.39	
8021	70.26	
8041	68.05	
8055	67.90	

8081	67.86
MFN Duty Free Tariff Lines Share (%) \	
0	22.74
9	20.80
18	18.34
25	17.22
33	19.67
...	...
8009	25.94
8021	26.37
8041	25.55
8055	25.25
8081	26.05
MFN Specific Tariff Lines Share (%) MFN AVE Tariff Lines Share (%) \	
0	70.32
9	56.32
18	59.62
25	39.70
33	19.62
...	...
8009	39.06
8021	32.74
8041	72.97
8055	28.04
8081	0.04
MFN MaxRate (%) MFN MinRate (%) \	
0	352.69
9	199.10
18	3000.00
25	3000.00
33	3000.00
...	...
8009	754.30
8021	3000.00
8041	3000.00
8055	3000.00
8081	3000.00
MFN SpecificDuty Imports (US\$ Thousand) \	
0	2.186000e+03
9	3.661140e+03
18	3.831487e+04
25	4.648980e+05
33	4.095565e+07
...	...
8009	4.990280e+03
8021	3.899550e+04
8041	2.239893e+06
8055	4.333552e+06
8081	5.557731e+09
MFN Dutiable Imports (US\$ Thousand) \	
0	3.128020e+03
9	2.178901e+04
18	3.449249e+05
25	1.548113e+06
33	7.176112e+07
...	...
8009	1.358092e+05
8021	2.016959e+05
8041	5.591468e+07

```
8055          4.381028e+06
8081          1.854956e+10

      MFN Duty Free Imports (US$ Thousand)
0                  0.0
9                  0.0
18                 0.0
25                 0.0
33                 0.0
...
8009                ...
8021                0.0
8041                0.0
8055                0.0
8081            31354.2
```

[223 rows x 33 columns]

```
In [ ]: cleaned_df = data.drop(outliers.index)

# Save the cleaned dataset to a new file
cleaned_df.to_csv('cleaned_dataset.csv', index=False)

# Print some information about the cleaning process
print(f"Original dataset shape: {data.shape}")
print(f"Number of outliers detected: {len(outliers)}")
print(f"Cleaned dataset shape: {cleaned_df.shape}")
print("Cleaned dataset saved as 'cleaned_dataset.csv'")
```

```
Original dataset shape: (8096, 33)
Number of outliers detected: 223
Cleaned dataset shape: (7873, 33)
Cleaned dataset saved as 'cleaned_dataset.csv'
```

```
In [ ]: #After cleaning data from the dataset count
len(cleaned_df)
```

```
Out[ ]: 7873
```

```
In [ ]: cleaned_df
```

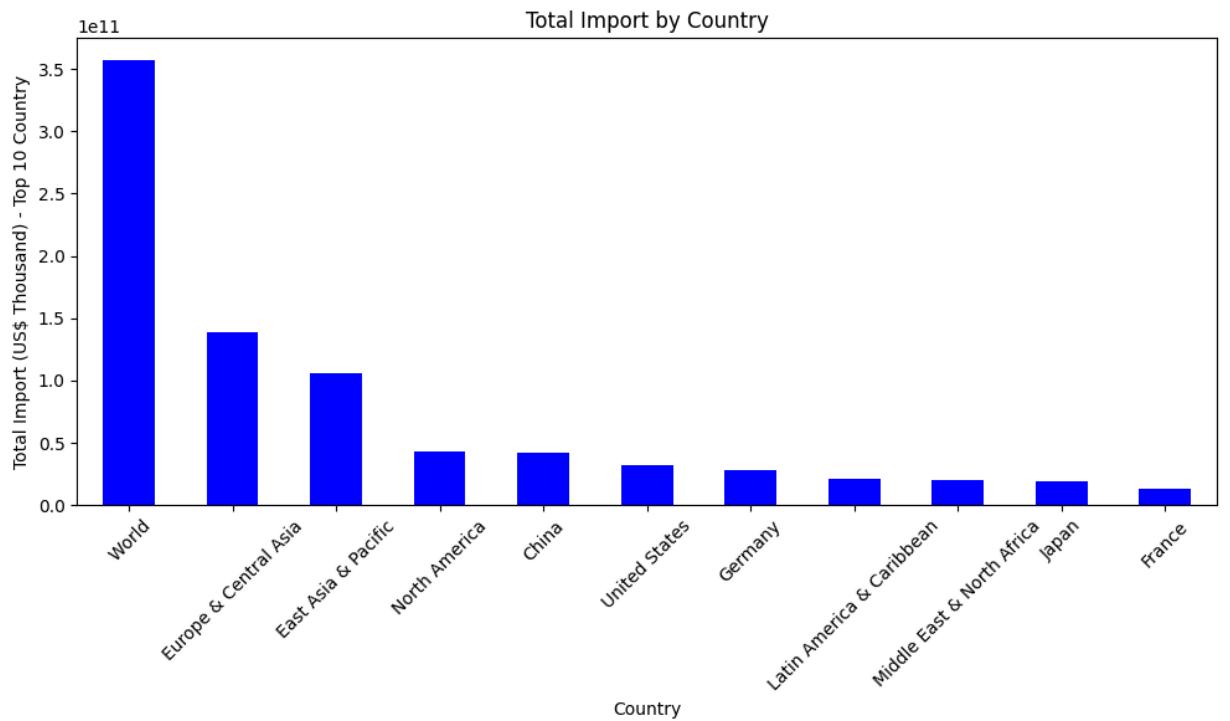
Out[]:

	Partner Name	Year	Export (US\$ Thousand)	Import (US\$ Thousand)	Export Product Share (%)	Import Product Share (%)	Revealed comparative advantage	World Growth (%)	Co G
1	Afghanistan	1988	2.130304e+05	5.445952e+04	100.0	100	1.0	5.270909	5.2
2	Angola	1988	3.755279e+05	3.707028e+05	100.0	100	1.0	-2.403333	-2.4
3	Anguila	1988	3.669800e+02	4.000000e+00	100.0	100	1.0	4.931667	4.9
4	Albania	1988	3.010356e+04	4.770930e+04	100.0	100	1.0	4.706000	4.7
5	Andorra	1988	6.792446e+04	3.284010e+03	100.0	100	1.0	0.965417	0.9
...
8091	Latin America & Caribbean	2021	1.330557e+09	1.310305e+09	100.0	100	0.0	0.000000	0.0
8092	Middle East & North Africa	2021	1.196712e+09	1.088471e+09	100.0	100	0.0	0.000000	0.0
8093	North America	2021	3.823319e+09	2.219849e+09	100.0	100	0.0	0.000000	0.0
8094	South Asia	2021	6.991380e+08	4.723832e+08	100.0	100	0.0	0.000000	0.0
8095	Sub-Saharan Africa	2021	4.951000e+08	4.350468e+08	100.0	100	0.0	0.000000	0.0

7873 rows × 33 columns

Exploratory Data Analysis (EDA)

```
In [ ]: # Import for Top 10 countries
import_by_country = data.groupby('Partner Name')['Import (US$ Thousand)'].sum().sort_top_n = 11
import_by_country = import_by_country.head(top_n)
# Create bar chart
plt.figure(figsize=(10, 6))
import_by_country.plot(kind='bar', color='blue')
plt.xlabel('Country')
plt.ylabel('Total Import (US$ Thousand) - Top 10 Country')
plt.title('Total Import by Country')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
In [ ]: # Export for Top 10 countries

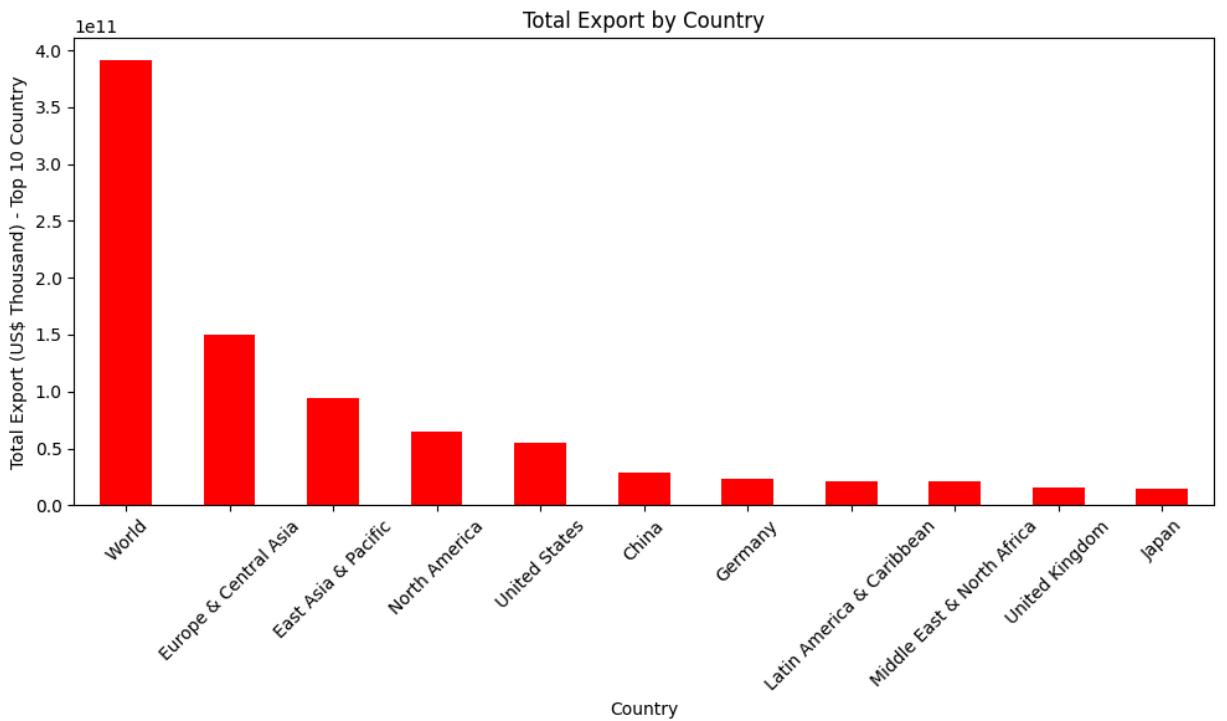
export_by_country = data.groupby('Partner Name')[ 'Export (US$ Thousand)'].sum().sort_ 

# Extract top N countries for visualization, if needed
top_n = 11
export_by_country = export_by_country.head(top_n)

# Create bar chart
plt.figure(figsize=(10, 6))
export_by_country.plot(kind='bar', color='red')

# Add Labels and title
plt.xlabel('Country')
plt.ylabel('Total Export (US$ Thousand) - Top 10 Country')
plt.title('Total Export by Country')

# Show plot
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
In [ ]: import matplotlib.pyplot as plt
import ipywidgets as widgets

def plot_export_import(country_name):
    country_data = data[data['Partner Name'] == country_name]
    country_data.set_index('Year', inplace=True)
    plt.figure(figsize=(10, 6))
    plt.bar(country_data.index, country_data['Export (US$ Thousand)'], color='skyblue')
    plt.bar(country_data.index, country_data['Import (US$ Thousand)'],
            bottom=country_data['Export (US$ Thousand)'], color='orange', label='Import')
    plt.xlabel('Year')
    plt.ylabel('Value (US$ Thousand)')
    plt.title(f'Export and Import Values for {country_name}')
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

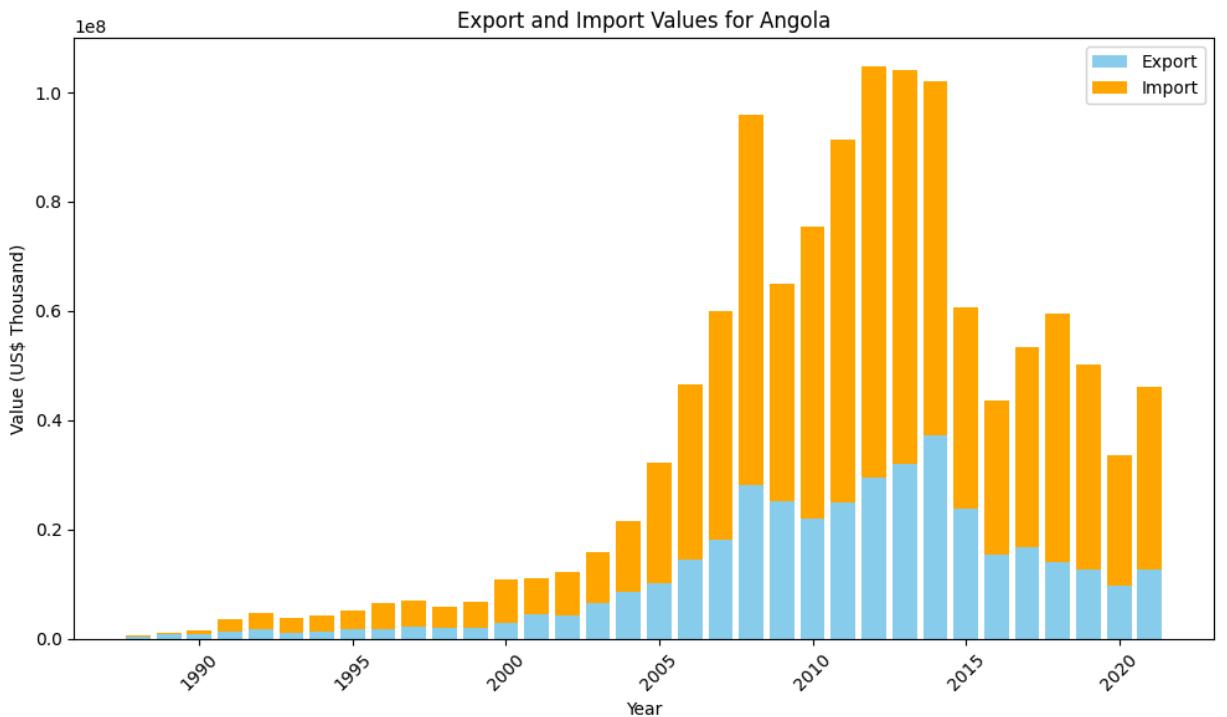
# Create dropdown menu for selecting country
country_dropdown = widgets.Dropdown(
    options=data['Partner Name'].unique(),
    description='Country:'
)

# Create function to update plot when dropdown value changes
def update_plot(change):
    country_name = change.new
    plot_export_import(country_name)

# Attach update_plot function to the dropdown's observe method
country_dropdown.observe(update_plot, names='value')

# Display the dropdown
display(country_dropdown)
```

Dropdown(description='Country:', options=('Aruba', 'Afghanistan', 'Angola', 'Anguilla', 'Albania', 'Andorra', '...'))



```
In [ ]: import matplotlib.pyplot as plt
import ipywidgets as widgets

def update_bar_chart(country_name, zoom_factor):
    country_data = data[data['Partner Name'] == country_name]
    if country_data.empty:
        print("No data found for the selected country.")
    else:
        # Plot bar chart for country growth over time
        plt.figure(figsize=(10, 6))
        plt.bar(country_data['Year'], country_data['Import (US$ Thousand)'], color='blue')
        plt.bar(country_data['Year'], country_data['Export (US$ Thousand)'], color='orange')

        # Add labels and title
        plt.xlabel('Year')
        plt.ylabel('Import (US$ Thousand)')
        plt.title(f'Import Values for {country_name}')

        # Apply zoom
        if zoom_factor > 0:
            min_import, max_import = min(country_data['Import (US$ Thousand)']), max(country_data['Import (US$ Thousand)'])
            plt.ylim(0, max_import * zoom_factor)

        # Show plot
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

# Create search bar for selecting country
country_dropdown = widgets.Dropdown(
    options=data['Partner Name'].unique(),
    description='Country:'
)

# Create zoom slider
zoom_slider = widgets.FloatSlider(
    value=1.0,
    min=0.1,
    max=2.0,
    step=0.1,
    description='Zoom Factor:'
)
```

```

)
# Display the dropdown, slider, and bar chart
widgets.interactive(update_bar_chart, country_name=country_dropdown, zoom_factor=zoom
Out[ ]: interactive(children=(Dropdown(description='Country:', options=('Aruba', 'Afghanista
n', 'Angola', 'Anguilla', '...
In [ ]: def update_bar_chart(country_name, zoom_factor):
    country_data = data[data['Partner Name'] == country_name]
    if country_data.empty:
        print("No data found for the selected country.")
    else:
        # Plot bar chart for country growth over time
        plt.figure(figsize=(10, 6))
        plt.bar(country_data['Year'], country_data['Export (US$ Thousand)'], color='r
        # Add labels and title
        plt.xlabel('Year')
        plt.ylabel('Export (US$ Thousand)')
        plt.title(f'Export Values for {country_name}')

        # Apply zoom
        if zoom_factor > 0:
            min_export, max_export = min(country_data['Export (US$ Thousand)']), max(
                plt.ylim(0, max_export * zoom_factor)

        # Show data point values on hover
        mplcursors.cursor(hover=True)

        # Show plot
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

# Create search bar for selecting country
country_dropdown = widgets.Dropdown(
    options=data['Partner Name'].unique(),
    description='Country:'
)

# Create zoom slider
zoom_slider = widgets.FloatSlider(
    value=1.0,
    min=0.1,
    max=2.0,
    step=0.1,
    description='Zoom Factor:'
)

# Display the dropdown, slider, and bar chart
widgets.interactive(update_bar_chart, country_name=country_dropdown, zoom_factor=zoom

```

```
Out[ ]: interactive(children=(Dropdown(description='Country:', options=('Aruba', 'Afghanista
n', 'Angola', 'Anguilla', '...

```

```

In [ ]: import matplotlib.pyplot as plt
import ipywidgets as widgets

def update_pie_chart(country_name):
    country_data = data[data['Partner Name'] == country_name]
    if country_data.empty:
        print("No data found for the selected country.")
    else:
        total_import = country_data['Import (US$ Thousand)'].sum()

```

```

total_export = country_data['Export (US$ Thousand)'].sum()

labels=['Import', 'Export']
values=[total_import, total_export]

plt.figure(figsize=(8, 8))
plt.pie(values, labels=labels, autopct='%.1f%%', startangle=140)
plt.title(f"Overall Import and Export Total for {country_name}")
plt.tight_layout()
plt.show()

# Create search bar for selecting country
country_dropdown = widgets.Dropdown(
    options=data['Partner Name'].unique(),
    description='Country:'
)

# Display the dropdown and pie chart
widgets.interactive(update_pie_chart, country_name=country_dropdown)

```

Out[]: interactive(children=(Dropdown(description='Country:', options=('Aruba', 'Afghanista
n', 'Angola', 'Anguila', '...')))

```

In [ ]: import matplotlib.pyplot as plt
import ipywidgets as widgets

def update_line_chart(country_name, zoom_factor):
    country_data = data[data['Partner Name'] == country_name]
    if country_data.empty:
        print("No data found for the selected country.")
    else:
        # Plot Line chart with markers for country growth over time
        plt.figure(figsize=(10, 6))
        plt.plot(country_data['Year'], country_data['Country Growth (%)'], marker='o'

        # Add Labels and title
        plt.xlabel('Year')
        plt.ylabel('Country Growth (%)')
        plt.title(f'Country Growth over Time for {country_name}')

        # Apply zoom
        if zoom_factor > 0:
            min_year, max_year = min(country_data['Year']), max(country_data['Year'])
            min_growth, max_growth = min(country_data['Country Growth (%)']), max(country_data['Country Growth (%)'])
            plt.xlim(min_year - 0.05 * zoom_factor * (max_year - min_year), max_year + 0.05 * zoom_factor * (max_year - min_year))
            plt.ylim(min_growth - 0.05 * zoom_factor * (max_growth - min_growth), max_growth + 0.05 * zoom_factor * (max_growth - min_growth))

        # Show plot
        plt.xticks(rotation=45)
        plt.grid(True) # Add grid lines for better readability
        plt.tight_layout()
        plt.show()

# Prompt the user to enter the country name and zoom factor
country_dropdown = widgets.Dropdown(
    options=data['Partner Name'].unique(),
    description='Country:'
)

zoom_slider = widgets.FloatSlider(
    value=1.0,
    min=0.1,
    max=2.0,
    step=0.1,
)

```

```
        description='Zoom Factor:'

    )

# Display the dropdown, slider, and Line chart
widgets.interactive(update_line_chart, country_name=country_dropdown, zoom_factor=zoo

Out[ ]: interactive(children=(Dropdown(description='Country:', options=('Aruba', 'Afghanista
n', 'Angola', 'Anguila', '...

```

```
In [ ]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import ipywidgets as widgets

# Your column names
column_names = ['Partner Name', 'Year', 'Export (US$ Thousand)', 'Import (US$ Thousan
    'Export Product Share (%)', 'Import Product Share (%)',
    'Revealed comparative advantage', 'World Growth (%)', 'Country Growth
    'AHS Simple Average (%)', 'AHS Weighted Average (%)', 'AHS Total Tari
    'AHS Dutiable Tariff Lines Share (%)', 'AHS Duty Free Tariff Lines Sh
    'AHS Specific Tariff Lines Share (%)', 'AHS AVE Tariff Lines Share (%
    'AHS MaxRate (%)', 'AHS MinRate (%)',
    'AHS SpecificDuty Imports (US$ Thousand)', 'AHS Dutiable Imports (US$%
    'AHS Duty Free Imports (US$ Thousand)', 'MFN Simple Average (%)', 'MF
    'MFN Total Tariff Lines', 'MFN Dutiable Tariff Lines Share (%)', 'MFN
    'MFN Specific Tariff Lines Share (%)', 'MFN AVE Tariff Lines Share (%
    'MFN MaxRate (%)', 'MFN MinRate (%)',
    'MFN SpecificDuty Imports (US$ Thousand)', 'MFN Dutiable Imports (US$%
    'MFN Duty Free Imports (US$ Thousand)']

# Create a search bar
search_bar = widgets.Text(
    placeholder='Type country name...', 
    description='Country:'
)

# Define a function to update the heatmap based on selected country
def update_heatmap(country_name):
    selected_data = data[data['Partner Name'] == country_name][column_names]
    if selected_data.empty:
        print("No data found for the selected country.")
    else:
        # Selecting only numeric columns for the heatmap
        numeric_data = selected_data.select_dtypes(include=['float64', 'int64'])

        # Calculating correlation matrix
        correlation_matrix = numeric_data.corr()

        # Plotting the heatmap
        plt.figure(figsize=(14, 10))
        sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu')
        plt.title(f'Correlation Heatmap for {country_name}')
        plt.show()

    # Link the search bar to the function
    widgets.interactive(update_heatmap, country_name=search_bar)
```

```
Out[ ]: interactive(children=(Text(value='', description='Country:', placeholder='Type countr
y name...'), Output()), ...
```

```
In [ ]: correlation_matrix = data.corr()

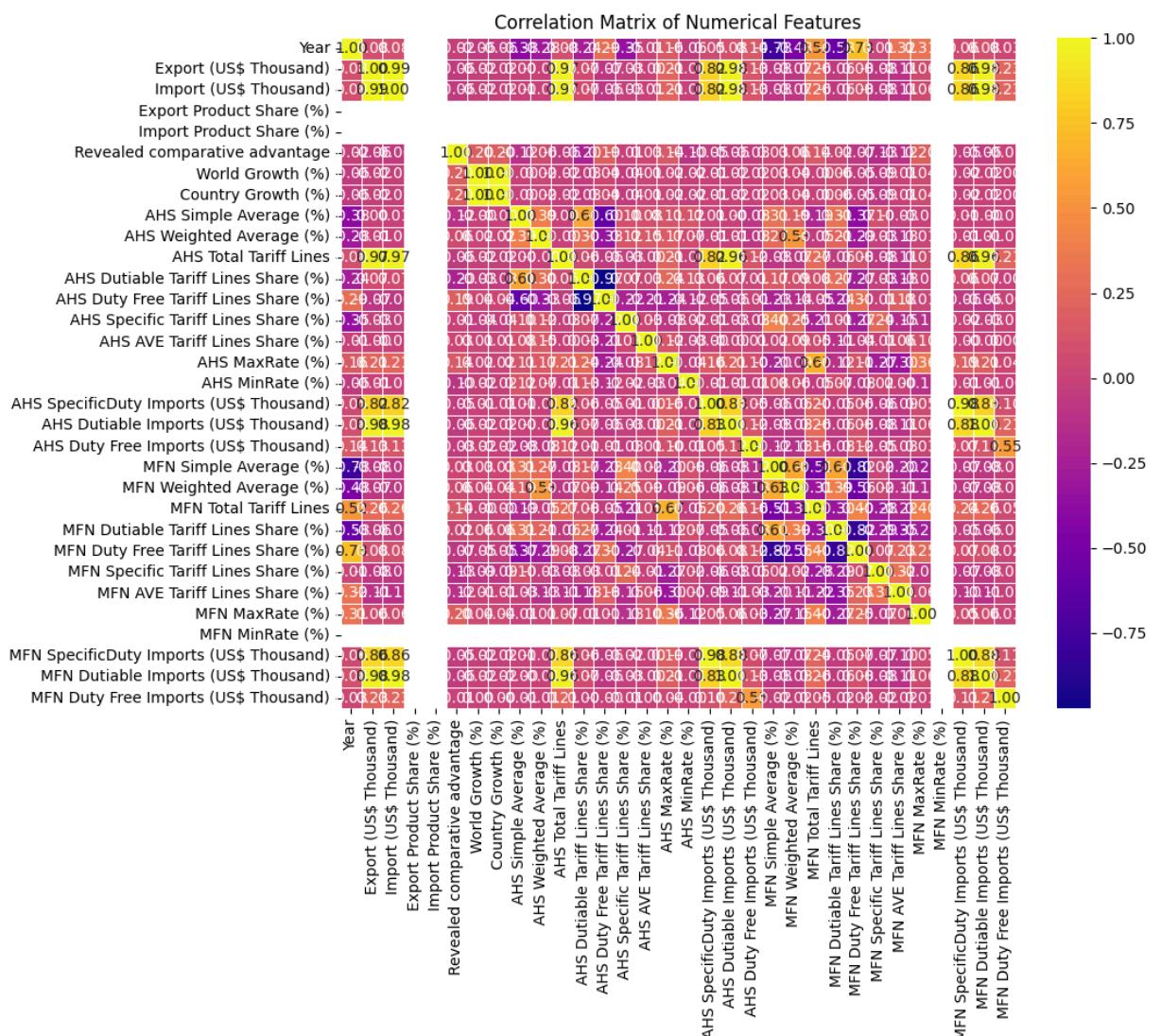
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='plasma', fmt=".2f", linewidths=0.5)
```

```
plt.title('Correlation Matrix of Numerical Features')
plt.show()

correlation_threshold = 0.8
highly_correlated_features = []

# Loop through columns of the correlation matrix
for i, column in enumerate(correlation_matrix.columns):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > correlation_threshold:
            # Add the pair of highly correlated features to the list
            highly_correlated_features.append((column, correlation_matrix.columns[j]))

print("Features with high correlation:")
for feature_pair in highly_correlated_features:
    print(f'{feature_pair[0]} and {feature_pair[1]}: {correlation_matrix.loc[feature_
```



Features with high correlation:

```

Import (US$ Thousand) and Export (US$ Thousand): 0.99
Country Growth (%) and World Growth (%): 1.00
AHS Total Tariff Lines and Export (US$ Thousand): 0.97
AHS Total Tariff Lines and Import (US$ Thousand): 0.97
AHS Duty Free Tariff Lines Share (%) and AHS Dutiable Tariff Lines Share (%): -0.97
AHS SpecificDuty Imports (US$ Thousand) and Export (US$ Thousand): 0.82
AHS SpecificDuty Imports (US$ Thousand) and Import (US$ Thousand): 0.82
AHS SpecificDuty Imports (US$ Thousand) and AHS Total Tariff Lines: 0.82
AHS Dutiable Imports (US$ Thousand) and Export (US$ Thousand): 0.98
AHS Dutiable Imports (US$ Thousand) and Import (US$ Thousand): 0.98
AHS Dutiable Imports (US$ Thousand) and AHS Total Tariff Lines: 0.96
AHS Dutiable Imports (US$ Thousand) and AHS SpecificDuty Imports (US$ Thousand): 0.83
MFN Duty Free Tariff Lines Share (%) and MFN Simple Average (%): -0.82
MFN Duty Free Tariff Lines Share (%) and MFN Dutiable Tariff Lines Share (%): -0.82
MFN SpecificDuty Imports (US$ Thousand) and Export (US$ Thousand): 0.86
MFN SpecificDuty Imports (US$ Thousand) and Import (US$ Thousand): 0.86
MFN SpecificDuty Imports (US$ Thousand) and AHS Total Tariff Lines: 0.86
MFN SpecificDuty Imports (US$ Thousand) and AHS SpecificDuty Imports (US$ Thousand): 0.98
MFN SpecificDuty Imports (US$ Thousand) and AHS Dutiable Imports (US$ Thousand): 0.88
MFN Dutiable Imports (US$ Thousand) and Export (US$ Thousand): 0.98
MFN Dutiable Imports (US$ Thousand) and Import (US$ Thousand): 0.98
MFN Dutiable Imports (US$ Thousand) and AHS Total Tariff Lines: 0.96
MFN Dutiable Imports (US$ Thousand) and AHS SpecificDuty Imports (US$ Thousand): 0.83
MFN Dutiable Imports (US$ Thousand) and AHS Dutiable Imports (US$ Thousand): 1.00
MFN Dutiable Imports (US$ Thousand) and MFN SpecificDuty Imports (US$ Thousand): 0.88

```

Dimensionality Reduction Using PCA

```
In [ ]: import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Separate features and target variable
X = data.drop(columns=['Country Growth (%)'])
y = data['Country Growth (%)']

# Identify numeric and categorical columns
numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = X.select_dtypes(include=['object']).columns

# Scale the numeric features
scaler = StandardScaler()
X_scaled = X.copy()
X_scaled[numeric_cols] = scaler.fit_transform(X_scaled[numeric_cols])

# Perform dimensionality reduction using PCA
pca = PCA(n_components=5) # Adjust the number of components as needed
X_pca = pca.fit_transform(X_scaled[numeric_cols])

# Convert X_pca to a DataFrame
X_pca_df = pd.DataFrame(X_pca, columns=[f"PC_{i+1}" for i in range(X_pca.shape[1])])

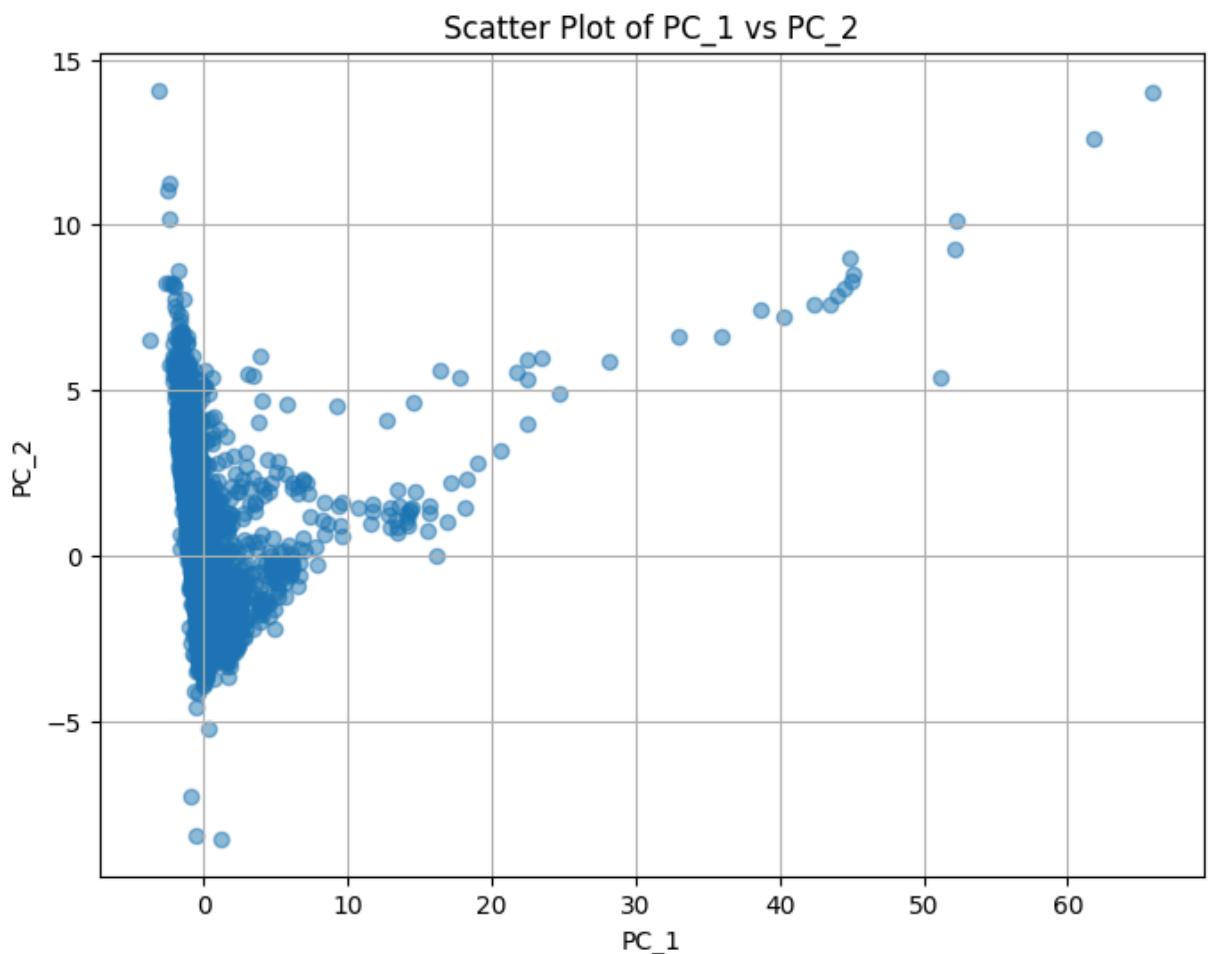
# Return the dimensionality-reduced data
X_pca_df.head()
```

```
Out[ ]:      PC_1    PC_2    PC_3    PC_4    PC_5
0 -1.040762  0.718222  2.988260  0.567414  0.029431
1 -1.148696  0.737958  3.257855 -1.344785  0.022144
2 -1.075279  1.021369  2.718948 -0.767445  0.875148
3 -1.311203  1.272154  3.869812 -0.046377  0.023758
4 -1.070487  1.130308  2.423634 -0.448297 -0.368462
```

```
In [ ]: import matplotlib.pyplot as plt
```

```
# Extract PC_1 and PC_2 values
PC_1 = X_pca_df['PC_1']
PC_2 = X_pca_df['PC_2']

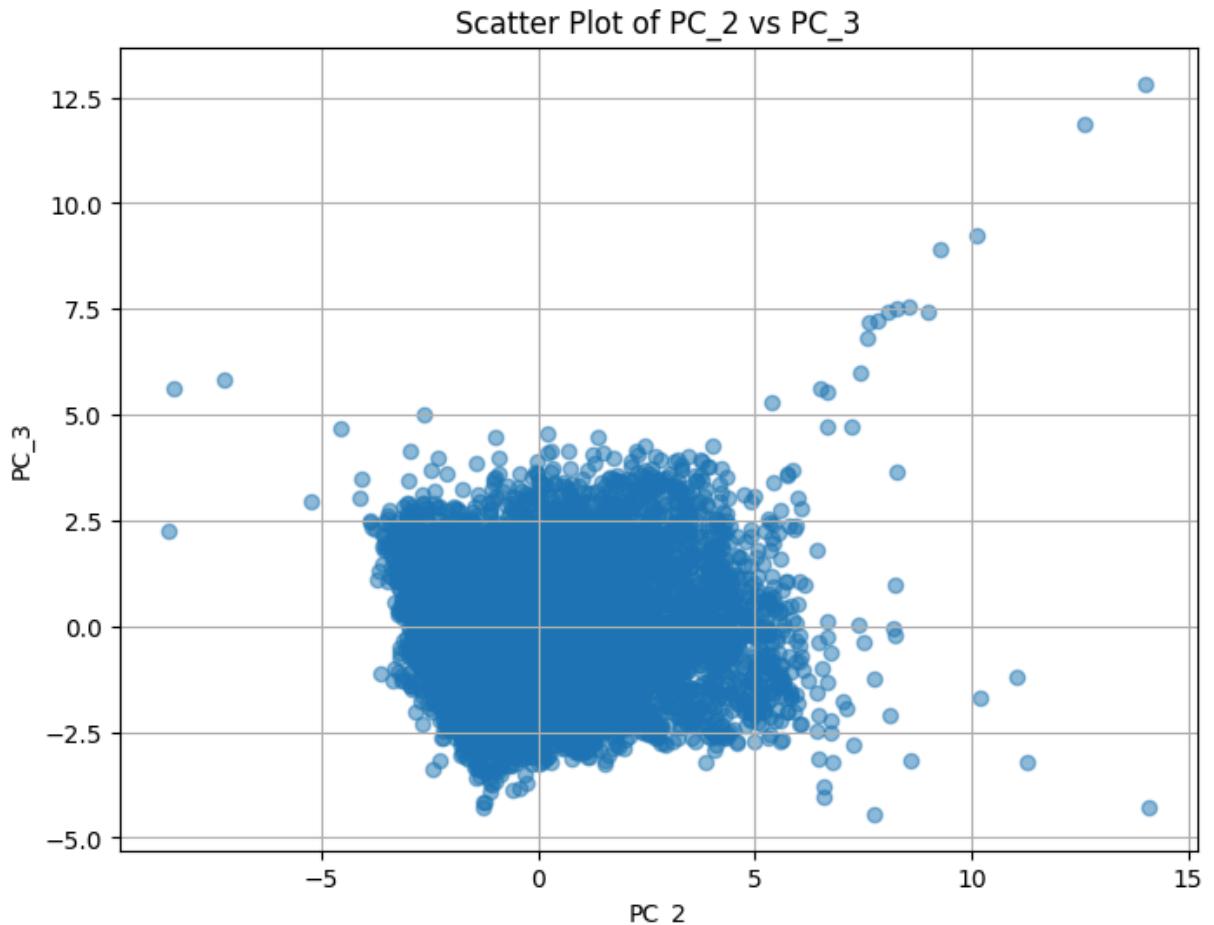
# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(PC_1, PC_2, alpha=0.5)
plt.title('Scatter Plot of PC_1 vs PC_2')
plt.xlabel('PC_1')
plt.ylabel('PC_2')
plt.grid(True)
plt.show()
```



```
In [ ]: import matplotlib.pyplot as plt
```

```
# Extract PC_1 and PC_2 values
PC_2 = X_pca_df['PC_2']
PC_3 = X_pca_df['PC_3']
```

```
# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(PC_2, PC_3, alpha=0.5)
plt.title('Scatter Plot of PC_2 vs PC_3')
plt.xlabel('PC_2')
plt.ylabel('PC_3')
plt.grid(True)
plt.show()
```



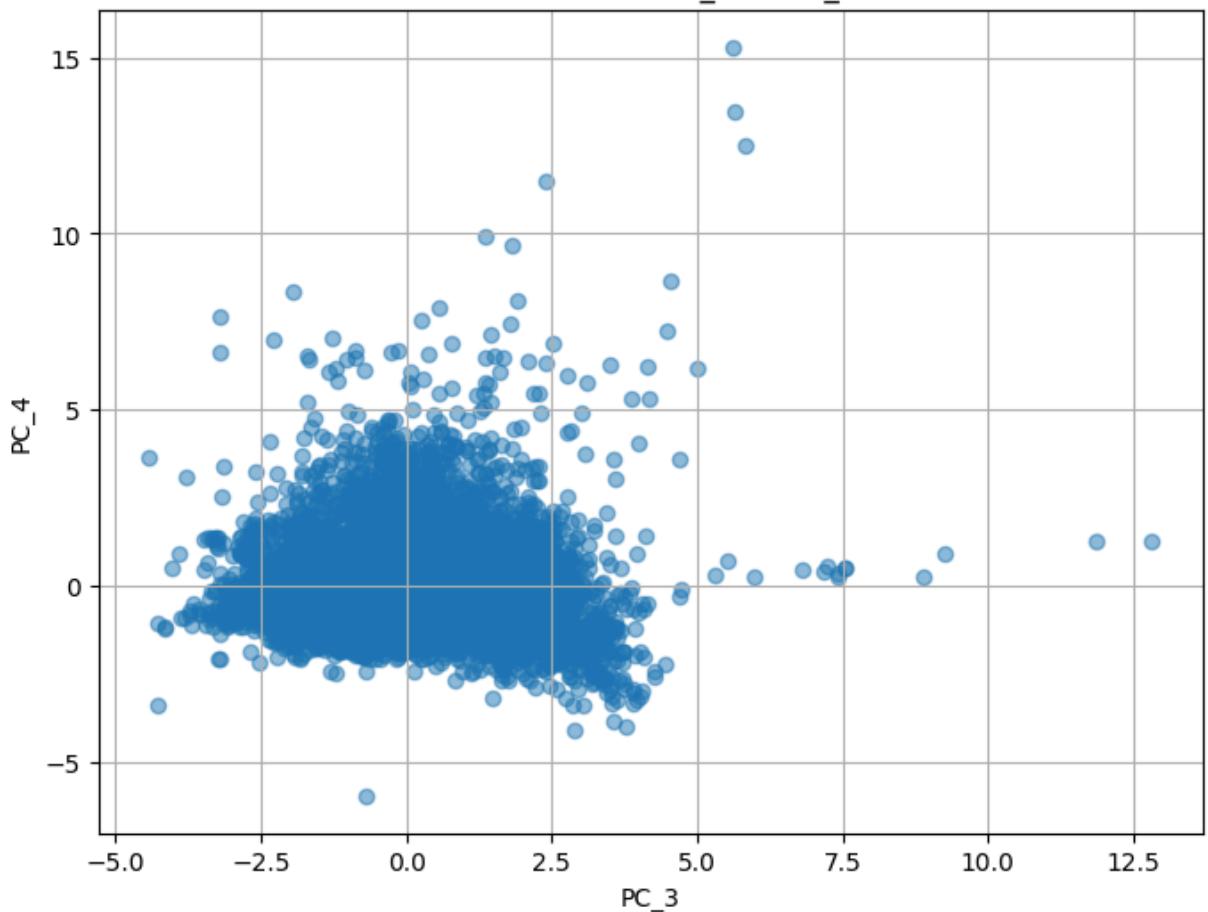
In []:

```
import matplotlib.pyplot as plt

# Extract PC_1 and PC_2 values
PC_3 = X_pca_df['PC_3']
PC_4 = X_pca_df['PC_4']

# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(PC_3, PC_4, alpha=0.5)
plt.title('Scatter Plot of PC_3 vs PC_4')
plt.xlabel('PC_3')
plt.ylabel('PC_4')
plt.grid(True)
plt.show()
```

Scatter Plot of PC_3 vs PC_4

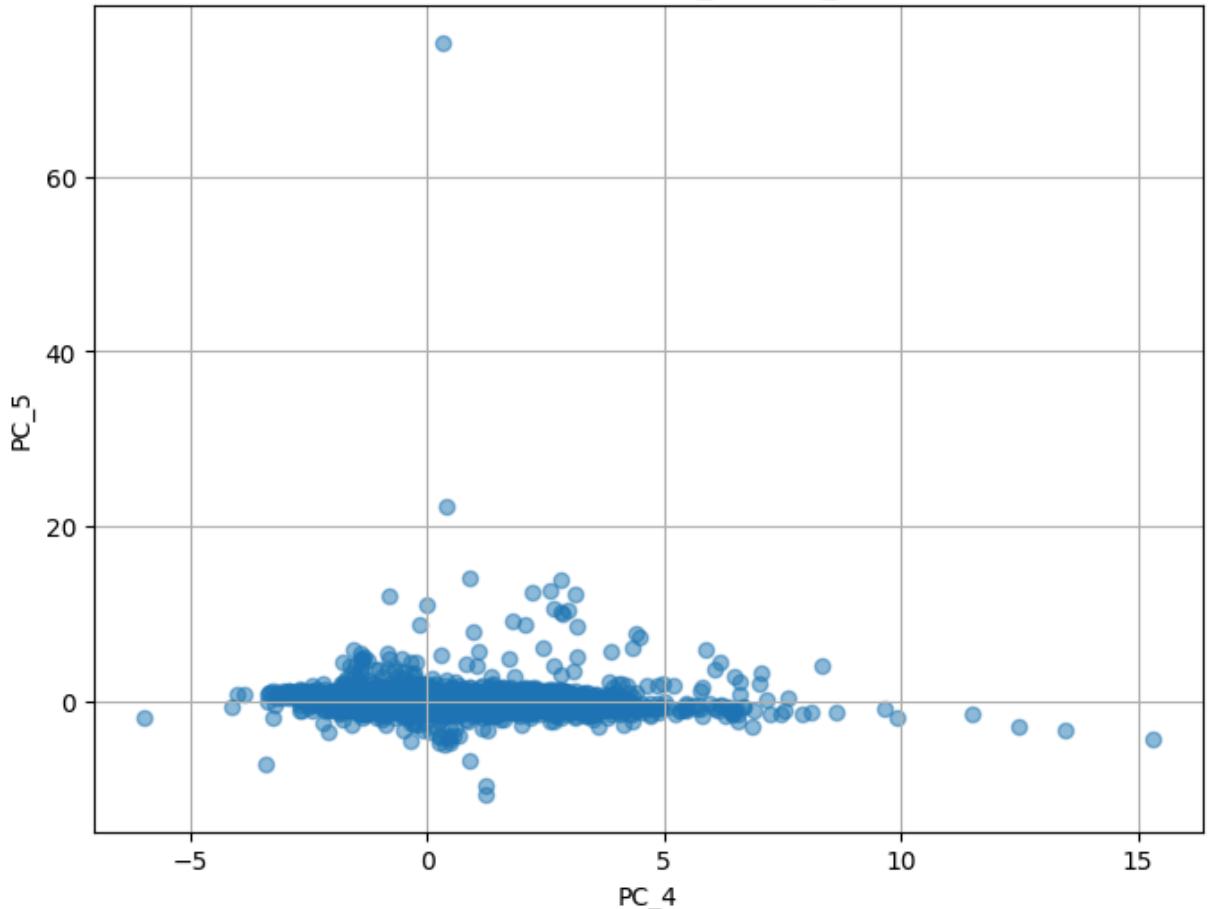


```
In [ ]: import matplotlib.pyplot as plt

# Extract PC_1 and PC_2 values
PC_4 = X_pca_df['PC_4']
PC_5 = X_pca_df['PC_5']

# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(PC_4, PC_5, alpha=0.5)
plt.title('Scatter Plot of PC_4 vs PC_5')
plt.xlabel('PC_4')
plt.ylabel('PC_5')
plt.grid(True)
plt.show()
```

Scatter Plot of PC_4 vs PC_5



Feature Selection:

```
In [ ]: import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

numeric_df = data.select_dtypes(include=['float64', 'int64'])
X = numeric_df.drop(columns=['Country Growth (%)']) # Drop the target variable if available
y = numeric_df['Country Growth (%)'] if 'Country Growth (%)' in numeric_df.columns else None

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 1. Correlation Analysis
# Calculate the correlation matrix
correlation_matrix = X_train.corr()

# Select features with high absolute correlation coefficients
high_corr_features = correlation_matrix[abs(correlation_matrix) > 0.5].stack().reset_index()
high_corr_features = high_corr_features[high_corr_features['level_0'] != high_corr_features['level_1']]
high_corr_features = high_corr_features.drop_duplicates(subset='correlation')

# Print selected features
print("Features selected based on correlation analysis:")
print(high_corr_features)
```

Features selected based on correlation analysis:

	level_0 \
1	Year
2	Year
3	Year
4	Year
6	Export (US\$ Thousand)
7	Export (US\$ Thousand)
8	Export (US\$ Thousand)
9	Export (US\$ Thousand)
10	Export (US\$ Thousand)
11	Export (US\$ Thousand)
14	Import (US\$ Thousand)
15	Import (US\$ Thousand)
16	Import (US\$ Thousand)
17	Import (US\$ Thousand)
18	Import (US\$ Thousand)
22	AHS Simple Average (%)
23	AHS Simple Average (%)
25	AHS Weighted Average (%)
29	AHS Total Tariff Lines
30	AHS Total Tariff Lines
31	AHS Total Tariff Lines
32	AHS Total Tariff Lines
35	AHS Dutiable Tariff Lines Share (%)
42	AHS MaxRate (%)
48	AHS SpecificDuty Imports (US\$ Thousand)
49	AHS SpecificDuty Imports (US\$ Thousand)
50	AHS SpecificDuty Imports (US\$ Thousand)
56	AHS Dutiable Imports (US\$ Thousand)
57	AHS Dutiable Imports (US\$ Thousand)
59	AHS Duty Free Imports (US\$ Thousand)
62	MFN Simple Average (%)
63	MFN Simple Average (%)
64	MFN Simple Average (%)
65	MFN Simple Average (%)
69	MFN Weighted Average (%)
77	MFN Dutiable Tariff Lines Share (%)
92	MFN SpecificDuty Imports (US\$ Thousand)

	level_1	correlation
1	MFN Simple Average (%)	-0.781912
2	MFN Total Tariff Lines	0.529837
3	MFN Dutiable Tariff Lines Share (%)	-0.588655
4	MFN Duty Free Tariff Lines Share (%)	0.732181
6	Import (US\$ Thousand)	0.994391
7	AHS Total Tariff Lines	0.974314
8	AHS SpecificDuty Imports (US\$ Thousand)	0.866187
9	AHS Dutiable Imports (US\$ Thousand)	0.975085
10	MFN SpecificDuty Imports (US\$ Thousand)	0.876842
11	MFN Dutiable Imports (US\$ Thousand)	0.975233
14	AHS Total Tariff Lines	0.975288
15	AHS SpecificDuty Imports (US\$ Thousand)	0.872060
16	AHS Dutiable Imports (US\$ Thousand)	0.982335
17	MFN SpecificDuty Imports (US\$ Thousand)	0.882099
18	MFN Dutiable Imports (US\$ Thousand)	0.982400
22	AHS Dutiable Tariff Lines Share (%)	0.601368
23	AHS Duty Free Tariff Lines Share (%)	-0.606018
25	MFN Weighted Average (%)	0.501949
29	AHS SpecificDuty Imports (US\$ Thousand)	0.886332
30	AHS Dutiable Imports (US\$ Thousand)	0.958610
31	MFN SpecificDuty Imports (US\$ Thousand)	0.892302
32	MFN Dutiable Imports (US\$ Thousand)	0.959199
35	AHS Duty Free Tariff Lines Share (%)	-0.971186

```

42          MFN Total Tariff Lines      0.668215
48          AHS Dutiable Imports (US$ Thousand) 0.878564
49  MFN SpecificDuty Imports (US$ Thousand) 0.980972
50          MFN Dutiable Imports (US$ Thousand) 0.878620
56  MFN SpecificDuty Imports (US$ Thousand) 0.896671
57          MFN Dutiable Imports (US$ Thousand) 0.999943
59          MFN Duty Free Imports (US$ Thousand) 0.574226
62          MFN Weighted Average (%) 0.631528
63          MFN Total Tariff Lines -0.510274
64          MFN Dutiable Tariff Lines Share (%) 0.611282
65  MFN Duty Free Tariff Lines Share (%) -0.817359
69  MFN Duty Free Tariff Lines Share (%) -0.559221
77  MFN Duty Free Tariff Lines Share (%) -0.825792
92  MFN Dutiable Imports (US$ Thousand) 0.896709

```

```

In [ ]: rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Get feature importances
feature_importances = pd.DataFrame({'Feature': X_train.columns, 'Importance': rf.feature_importances_})
feature_importances = feature_importances.sort_values(by='Importance', ascending=False)

# Select top features based on importance scores
top_features = feature_importances.head(10) # Select top 10 features (you can adjust this number)

# Print selected features
print("\nTop features selected based on feature importance:")
top_features

```

Top features selected based on feature importance:

	Feature	Importance
6	World Growth (%)	0.991193
14	AHS MaxRate (%)	0.002595
30	MFN Duty Free Imports (US\$ Thousand)	0.001587
12	AHS Specific Tariff Lines Share (%)	0.001248
20	MFN Weighted Average (%)	0.000729
0	Year	0.000587
24	MFN Specific Tariff Lines Share (%)	0.000375
23	MFN Duty Free Tariff Lines Share (%)	0.000326
21	MFN Total Tariff Lines	0.000210
7	AHS Simple Average (%)	0.000199

In []: