DSC680-T301_2251_1 Applied Data Science

Assignment Week 7 Project 2 Milstone 1;

Author: Zemelak Goraga;

Date: 10/12/2024

Step 1: Connecting to an API, Pulling in the animals dataset, and inspect

In [1]:
```python

import subprocess
import os
import zipfile
import pandas as pd
from zipfile import ZipFile
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
# Execute the Kaggle API command to download the live animals dataset contsining ch
command = "kaggle datasets download -d unitednations/global-food-agriculture-statis
subprocess.run(command.split())
```

Out[2]:
```
CompletedProcess(args=['kaggle', 'datasets', 'download', '-d', 'unitednations/glob
al-food-agriculture-statistics'], returncode=0)
```

In [3]:
```python
# Step 2: Check if the download was successful
if os.path.exists("global-food-agriculture-statistics.zip"):
    print("Dataset downloaded successfully!")
```

```
Dataset downloaded successfully!
```

In [4]:
```python
# Step 3: Unzip the downloaded file
with zipfile.ZipFile("global-food-agriculture-statistics.zip", "r") as zip_ref:
    zip_ref.extractall("data")
```

In [5]:
```python
# Step 4: Optionally, list the contents of the extracted directory
extracted_files = os.listdir("data")
print("Extracted files:", extracted_files)
```

```
Extracted files: ['current_FAO', 'fao_data_crops_data.csv', 'fao_data_fertilizers_da
ta.csv', 'fao_data_forest_data.csv', 'fao_data_land_data.csv', 'fao_data_production_
indices_data.csv']
```

In [ ]:
```python
    # Step 5: Download a specific table to work with
    # Specify the CSV file to read from the ZIP archive
    csv_file_to_read = "current_FAO/raw_files/Trade_LiveAnimals_E_All_Data_(Normali
```

```python
    # Read the ZIP archive
    with ZipFile("global-food-agriculture-statistics.zip", 'r') as zip_file:
        # List the files within the ZIP archive (to double-check paths)
        print(zip_file.namelist())

        # Read the CSV file from the ZIP archive with the specified encoding and de
        with zip_file.open(csv_file_to_read) as csv_file:
            df = pd.read_csv(csv_file, encoding='ISO-8859-1')
```

In [35]:
```python
# Print the first few rows of the dataset
df.head()
```

Out[35]:

	Area Code	Area	Item Code	Item	Element Code	Element	Year Code	Year	Unit	Value	Flag
0	2	Afghanistan	866	Cattle	5608	Import Quantity	1961	1961	Head	NaN	M
1	2	Afghanistan	866	Cattle	5608	Import Quantity	1962	1962	Head	NaN	M
2	2	Afghanistan	866	Cattle	5608	Import Quantity	1963	1963	Head	NaN	M
3	2	Afghanistan	866	Cattle	5608	Import Quantity	1964	1964	Head	NaN	M
4	2	Afghanistan	866	Cattle	5608	Import Quantity	1965	1965	Head	NaN	M

In [44]:
```python
# Print the last few rows of the dataset
df.tail()
```

Out[44]:

	Area Code	Area	Item Code	Item	Element Code	Element	Year Code	Year	Unit	Value
662953	5817	Net Food Importing Developing Countries	1922	Sheep and Goats	5922	Export Value	2009	2009	1000 US$	456293.0
662954	5817	Net Food Importing Developing Countries	1922	Sheep and Goats	5922	Export Value	2010	2010	1000 US$	421311.0
662955	5817	Net Food Importing Developing Countries	1922	Sheep and Goats	5922	Export Value	2011	2011	1000 US$	649321.0
662956	5817	Net Food Importing Developing Countries	1922	Sheep and Goats	5922	Export Value	2012	2012	1000 US$	778317.0
662957	5817	Net Food Importing Developing Countries	1922	Sheep and Goats	5922	Export Value	2013	2013	1000 US$	1038636.0

Pigs Dataset

In [45]:

```
# Filtering the Pigs dataset (df2) from the entire live animales dataset (df)
df2 = df[df['Item'] == 'Pigs'] # here after the chicken dataset will be refered as
```

In [46]:

```
df2.head()
```

Out[46]:

	Area Code	Area	Item Code	Item	Element Code	Element	Year Code	Year	Unit	Value	Flag
2862	3	Albania	1034	Pigs	5608	Import Quantity	1961	1961	Head	NaN	M
2863	3	Albania	1034	Pigs	5608	Import Quantity	1962	1962	Head	NaN	M
2864	3	Albania	1034	Pigs	5608	Import Quantity	1963	1963	Head	NaN	M
2865	3	Albania	1034	Pigs	5608	Import Quantity	1964	1964	Head	NaN	M
2866	3	Albania	1034	Pigs	5608	Import Quantity	1965	1965	Head	NaN	M

In [47]:
```
df2.tail()
```

Out[47]:

	Area Code	Area	Item Code	Item	Element Code	Element	Year Code	Year	Unit	Value	Flag
661469	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2009	2009	1000 US$	1160.0	A
661470	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2010	2010	1000 US$	2052.0	A
661471	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2011	2011	1000 US$	2423.0	A
661472	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2012	2012	1000 US$	2960.0	A
661473	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2013	2013	1000 US$	2081.0	A

Data transformation and cleansing

In [48]:
```
# Step 1: Replace Headers
new_headers = ["area_code","area", "item_code", "item", "element_code", "element",
df2.columns = new_headers
df2
```

Out[48]:

	area_code	area	item_code	item	element_code	element	year_code	year
2862	3	Albania	1034	Pigs	5608	Import Quantity	1961	1961
2863	3	Albania	1034	Pigs	5608	Import Quantity	1962	1962
2864	3	Albania	1034	Pigs	5608	Import Quantity	1963	1963
2865	3	Albania	1034	Pigs	5608	Import Quantity	1964	1964
2866	3	Albania	1034	Pigs	5608	Import Quantity	1965	1965
...	...	...	...	...	...	...	...	...
661469	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2009	2009
661470	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2010	2010
661471	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2011	2011
661472	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2012	2012
661473	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2013	2013

37120 rows × 11 columns

In [49]:

```
# renaming 'area' and 'item' columns

# Renaming columns 'area' to 'country' and 'item' to 'animal_category'
df2 = df2.rename(columns={'area': 'country', 'item': 'animal_category'})

df2.head()
```

Out[49]:

	area_code	country	item_code	animal_category	element_code	element	year_code
2862	3	Albania	1034	Pigs	5608	Import Quantity	1961
2863	3	Albania	1034	Pigs	5608	Import Quantity	1962
2864	3	Albania	1034	Pigs	5608	Import Quantity	1963
2865	3	Albania	1034	Pigs	5608	Import Quantity	1964
2866	3	Albania	1034	Pigs	5608	Import Quantity	1965

In [50]:

```
# data types
print(df2.dtypes)
```

```
area_code            int64
country             object
item_code            int64
animal_category     object
element_code         int64
element             object
year_code            int64
year                 int64
unit                object
value              float64
flag                object
dtype: object
```

In [53]:

```
# Step 1: Handling Missing Values
missing_values = df2.isnull().sum()
print("Missing values:\n", missing_values)

# Replacing missing values with the mean for numeric columns only
numeric_columns = df2.select_dtypes(include='number').columns
df2[numeric_columns] = df2[numeric_columns].fillna(df2[numeric_columns].mean())

# Checking if missing values are handled
missing_values_after = df2.isnull().sum()
print("Missing values after handling:\n", missing_values_after)

# Step 2: Remove rows where 'flag' column has missing values
df2 = df2[df2['flag'].notna()]

# Step 3: Apply criteria > 100 for both Export Quantity and Export Value
df2 = df2[~((df2['element'] == 'Export Quantity') & (df2['value'] <= 100))]
df2 = df2[~((df2['element'] == 'Export Value') & (df2['value'] <= 100))]

# Display the filtered dataset after handling missing values, removing rows with mi
print("Dataset after applying the criteria and removing rows with missing flag:")
df2.head()
```

```
Missing values:
 area_code          0
country            0
item_code          0
animal_category    0
element_code       0
element            0
year_code          0
year               0
unit               0
value              0
flag               0
dtype: int64
Missing values after handling:
 area_code          0
country            0
item_code          0
animal_category    0
element_code       0
element            0
year_code          0
year               0
unit               0
value              0
flag               0
dtype: int64
Dataset after applying the criteria and removing rows with missing flag:
```

Out[53]:

	area_code	country	item_code	animal_category	element_code	element	year_code
2862	3	Albania	1034	Pigs	5608	Import Quantity	1961
2863	3	Albania	1034	Pigs	5608	Import Quantity	1962
2864	3	Albania	1034	Pigs	5608	Import Quantity	1963
2865	3	Albania	1034	Pigs	5608	Import Quantity	1964
2866	3	Albania	1034	Pigs	5608	Import Quantity	1965

In [ ]:

```
```

In [55]:

```
# Step 5: Format Data

# Format 'value' columns into a readable format (e.g., adding commas for thousands
df2['value'] = df2['value'].apply(lambda x: '{:,.2f}'.format(x) if isinstance(x, (f
df2
```

Out[55]:

	area_code	country	item_code	animal_category	element_code	element	year_c
0	3	Albania	1034	Pigs	5608	Import Quantity	1
1	3	Albania	1034	Pigs	5608	Import Quantity	1
2	3	Albania	1034	Pigs	5608	Import Quantity	1
3	3	Albania	1034	Pigs	5608	Import Quantity	1
4	3	Albania	1034	Pigs	5608	Import Quantity	1
...	...	...	...	...	...	...	
14615	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14616	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14617	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14618	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14619	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2

14620 rows × 11 columns

In [ ]:

```
```

In [56]:

```
# Step 8: Fix Inconsistent Values: convert all strings to lowercase to address inco

df2['country'] = df2['country'].str.lower()

df2
```

Out[56]:

	area_code	country	item_code	animal_category	element_code	element	year_c
0	3	albania	1034	Pigs	5608	Import Quantity	19
1	3	albania	1034	Pigs	5608	Import Quantity	19
2	3	albania	1034	Pigs	5608	Import Quantity	19
3	3	albania	1034	Pigs	5608	Import Quantity	19
4	3	albania	1034	Pigs	5608	Import Quantity	19
...	...	...	...	...	...	...	
14615	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14616	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14617	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14618	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14619	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20

14620 rows × 11 columns

In [57]:

```
# Step 9: Replace Inconsistent Values with Standardized Ones
# For example, replacing 'united states' with 'United States of America'
df2['country'].replace({'united states': 'United States of America'}, inplace=True)

df2
```

Out[57]:

	area_code	country	item_code	animal_category	element_code	element	year_co
0	3	albania	1034	Pigs	5608	Import Quantity	19
1	3	albania	1034	Pigs	5608	Import Quantity	19
2	3	albania	1034	Pigs	5608	Import Quantity	19
3	3	albania	1034	Pigs	5608	Import Quantity	19
4	3	albania	1034	Pigs	5608	Import Quantity	19
...	...	...	...	...	...	...	
14615	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14616	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14617	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14618	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20
14619	5817	net food importing developing countries	1034	Pigs	5922	Export Value	20

14620 rows × 11 columns

In [58]:

```
# Step 9: Replace Inconsistent Values with Standardized Ones
# For example, replacing 'united states' with 'United States of America'
df2['country'].replace({'afghanistan': 'Afghanistan'}, inplace=True)

df2
```

Out[58]:

	area_code	country	item_code	animal_category	element_code	element	year_cc
0	3	albania	1034	Pigs	5608	Import Quantity	1!
1	3	albania	1034	Pigs	5608	Import Quantity	1!
2	3	albania	1034	Pigs	5608	Import Quantity	1!
3	3	albania	1034	Pigs	5608	Import Quantity	1!
4	3	albania	1034	Pigs	5608	Import Quantity	1!
...	...	...	...	...	...	...	
14615	5817	net food importing developing countries	1034	Pigs	5922	Export Value	2(
14616	5817	net food importing developing countries	1034	Pigs	5922	Export Value	2(
14617	5817	net food importing developing countries	1034	Pigs	5922	Export Value	2(
14618	5817	net food importing developing countries	1034	Pigs	5922	Export Value	2(
14619	5817	net food importing developing countries	1034	Pigs	5922	Export Value	2(

14620 rows × 11 columns

In [59]:

```
# Step 10: Making countries names start with capital letter, except preposition
# List of common prepositions to be converted to lowercase
prepositions = ['on', 'and', 'in', 'to', 'with', 'by', 'at', 'for', 'of', 'from']

# Function to capitalize each word in a string, except for prepositions
def capitalize_country_name(country):
    words = country.split()  # Split the country name into words
    capitalized_words = [word.capitalize() if word.lower() not in prepositions else
    return ' '.join(capitalized_words)

# Apply the function to the 'country' column
```

```
df2['country'] = df2['country'].apply(capitalize_country_name)

# Print the updated DataFrame
df2.head()
```

Out[59]:

	area_code	country	item_code	animal_category	element_code	element	year_code	ye
0	3	Albania	1034	Pigs	5608	Import Quantity	1961	19
1	3	Albania	1034	Pigs	5608	Import Quantity	1962	19
2	3	Albania	1034	Pigs	5608	Import Quantity	1963	19
3	3	Albania	1034	Pigs	5608	Import Quantity	1964	19
4	3	Albania	1034	Pigs	5608	Import Quantity	1965	19

In [60]:

```
# Step 12: Cleaned Dataset: Print the cleaned Pigs dataset

# Cleaned Dataset: Print the cleaned dataset
print("Cleaned Dataset:")
df2
```

```
Cleaned Dataset:
```

Out[60]:

	area_code	country	item_code	animal_category	element_code	element	year_c
0	3	Albania	1034	Pigs	5608	Import Quantity	1
1	3	Albania	1034	Pigs	5608	Import Quantity	1
2	3	Albania	1034	Pigs	5608	Import Quantity	1
3	3	Albania	1034	Pigs	5608	Import Quantity	1
4	3	Albania	1034	Pigs	5608	Import Quantity	1
...	...	...	...	...	...	...	
14615	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14616	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14617	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14618	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2
14619	5817	Net Food Importing Developing Countries	1034	Pigs	5922	Export Value	2

14620 rows × 11 columns

Renaming cleaned pigs dataset (df2) as pigs_data

In [61]:

```
# Assuming df2 is a pandas DataFrame
df2.to_csv('pigs_data.csv', index=False)
```

In [62]:

```
import pandas as pd
```

```
# Load the pigs_data.csv file
pigs_data = pd.read_csv('pigs_data.csv')

# Print the first few rows using head()
pigs_data.head()
```

Out[62]:

	area_code	country	item_code	animal_category	element_code	element	year_code	ye
0	3	Albania	1034	Pigs	5608	Import Quantity	1961	19
1	3	Albania	1034	Pigs	5608	Import Quantity	1962	19
2	3	Albania	1034	Pigs	5608	Import Quantity	1963	19
3	3	Albania	1034	Pigs	5608	Import Quantity	1964	19
4	3	Albania	1034	Pigs	5608	Import Quantity	1965	19

Descriptive Statistics

In [63]:

```
# > 100 export quantity
# Descriptive Statistics of exported quantity (heads) of pigs by the top 10 countri
import pandas as pd

# Step 1: Convert 'value' column to numeric, forcing errors to NaN if any non-numer
pigs_data['value'] = pd.to_numeric(pigs_data['value'], errors='coerce')

# Step 2: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 3: Filter the data to include only "Export Quantity" in the 'element' column
quantity_data = filtered_data[(filtered_data['element'] == 'Export Quantity') & (fi

# Step 4: Aggregate the total export quantity for each country to identify the top
top_10_countries = quantity_data.groupby('country')['value'].sum().nlargest(10).ind

# Step 5: Filter the data to include only the top 10 countries
top_10_quantity_data = quantity_data[quantity_data['country'].isin(top_10_countries

# Step 6: Group the data by year and calculate descriptive statistics for Export Qu
descriptive_stats_quantity_by_year = top_10_quantity_data.groupby('year')['value'].

# Step 7: Drop the "count", "25%", "50%", and "75%" columns from the statistics
descriptive_stats_quantity_by_year = descriptive_stats_quantity_by_year.drop(column

# Display the descriptive statistics for Export Quantity, grouped by year (showing
print("Descriptive Statistics for Export Quantity (Top 10 Countries, Grouped by Yea
print(descriptive_stats_quantity_by_year)
```

```
Descriptive Statistics for Export Quantity (Top 10 Countries, Grouped by Year, Exclu
ding Percentiles and Count):
            mean         std    min    max
year
1998  659.000000  175.581890  519.0  856.0
1999  641.800000  300.548166  251.0  994.0
2000  529.400000  308.979449  155.0  911.0
2001  343.000000  307.146545  106.0  690.0
2002  658.000000         NaN  658.0  658.0
2003  484.000000  314.996825  123.0  703.0
2004  552.750000  240.804727  289.0  843.0
2006  598.800000  270.125341  251.0  996.0
2007  401.200000  233.821941  159.0  686.0
2008  305.400000  112.597957  145.0  447.0
2009  343.333333  213.326823  155.0  575.0
2010  332.000000  219.160520  165.0  820.0
2011  382.666667  302.769439  196.0  732.0
2012  723.000000  150.582203  593.0  888.0
2013  472.666667  259.230271  323.0  772.0
```

Explanation

The descriptive statistics for export quantities of pigs from 1998 to 2013 show significant fluctuations in both average exports and variability among the top 10 exporting countries. The mean export quantity ranged from a low of 305.4 heads in 2008 to a high of 723 heads in 2012. Variability, as reflected by standard deviations, was particularly high in years like 1999 (300.5) and 2000 (308.9), indicating uneven export performance. The range of exports also varied widely, with minimum values as low as 106 heads in 2001 and maximums reaching 996 heads in 2006, highlighting differences in export contributions across countries.

In [ ]:

```
```

In [64]:

```
# x 1000

# Descriptive Statistics of export value(US$) of pigs by the top 10 countries in th

import pandas as pd

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Value" in the 'element' column
value_data = filtered_data[filtered_data['element'] == 'Export Value']

# Step 3: Multiply 'value' by 1000 to convert it to US dollars (as the current unit
value_data['value'] = value_data['value'] * 1000

# Step 4: Aggregate the total export value for each country to identify the top 10
top_10_countries = value_data.groupby('country')['value'].sum().nlargest(10).index

# Step 5: Filter the data to include only the top 10 countries
```

```
top_10_value_data = value_data[value_data['country'].isin(top_10_countries)]

# Step 6: Group the data by year and calculate descriptive statistics for Export Va
descriptive_stats_value_by_year = top_10_value_data.groupby('year')['value'].descri

# Step 7: Drop the "count", "25%", "50%", and "75%" columns from the statistics
descriptive_stats_value_by_year = descriptive_stats_value_by_year.drop(columns=['co

# Display the descriptive statistics for Export Value, grouped by year (showing onl
print("Descriptive Statistics for Export Value (Top 10 Countries, Grouped by Year,
print(descriptive_stats_value_by_year)
```

```
Descriptive Statistics for Export Value (Top 10 Countries, Grouped by Year, Excludin
g Percentiles and Count):
               mean            std       min       max
year
1998  414600.000000  230504.856541  151000.0  794000.0
1999  488300.000000  279798.359935  101000.0  851000.0
2000  289555.555556  120173.947999  125000.0  420000.0
2001  465200.000000  342840.033965  145000.0  999000.0
2002  425500.000000  320040.014859  124000.0  966000.0
2003  482875.000000  299003.792379  158000.0  965000.0
2004  534285.714286  280561.172347  101000.0  970000.0
2005  619285.714286  207113.265538  383000.0  935000.0
2006  337833.333333   93098.693152  243000.0  472000.0
2007  642100.000000  263267.692241  114000.0  902000.0
2008  429000.000000  242952.082283  174000.0  850000.0
2009  246500.000000  247129.723020  103000.0  748000.0
2010  189600.000000   34048.494827  156000.0  244000.0
2011  415400.000000  220251.674227  204000.0  763000.0
2012  497600.000000  248683.131716  328000.0  915000.0
2013  316000.000000  147967.901925  149000.0  531000.0
```

In [65]:

```
# >100 for export value
# Descriptive Statistics of export value(US$) of pigs by the top 10 countries in th

import pandas as pd

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Value" in the 'element' column an
value_data = filtered_data[(filtered_data['element'] == 'Export Value') & (filtered

# Step 3: Multiply 'value' by 1000 to convert it to US dollars (as the current unit
value_data['value'] = value_data['value'] * 1000

# Step 4: Aggregate the total export value for each country to identify the top 10
top_10_countries = value_data.groupby('country')['value'].sum().nlargest(10).index

# Step 5: Filter the data to include only the top 10 countries
top_10_value_data = value_data[value_data['country'].isin(top_10_countries)]

# Step 6: Group the data by year and calculate descriptive statistics for Export Va
descriptive_stats_value_by_year = top_10_value_data.groupby('year')['value'].descri
```

```
# Step 7: Drop the "count", "25%", "50%", and "75%" columns from the statistics
descriptive_stats_value_by_year = descriptive_stats_value_by_year.drop(columns=['co

# Display the descriptive statistics for Export Value, grouped by year (showing onl
print("Descriptive Statistics for Export Value (Top 10 Countries, Grouped by Year,
print(descriptive_stats_value_by_year)
```

```
Descriptive Statistics for Export Value (Top 10 Countries, Grouped by Year, Excludin
g Percentiles and Count):
               mean            std       min       max
year
1998  414600.000000  230504.856541  151000.0  794000.0
1999  488300.000000  279798.359935  101000.0  851000.0
2000  289555.555556  120173.947999  125000.0  420000.0
2001  465200.000000  342840.033965  145000.0  999000.0
2002  425500.000000  320040.014859  124000.0  966000.0
2003  482875.000000  299003.792379  158000.0  965000.0
2004  534285.714286  280561.172347  101000.0  970000.0
2005  619285.714286  207113.265538  383000.0  935000.0
2006  337833.333333   93098.693152  243000.0  472000.0
2007  642100.000000  263267.692241  114000.0  902000.0
2008  429000.000000  242952.082283  174000.0  850000.0
2009  246500.000000  247129.723020  103000.0  748000.0
2010  189600.000000   34048.494827  156000.0  244000.0
2011  415400.000000  220251.674227  204000.0  763000.0
2012  497600.000000  248683.131716  328000.0  915000.0
2013  316000.000000  147967.901925  149000.0  531000.0
```

Explanation

The descriptive statistics for the export value of pigs from 1998 to 2013 show fluctuations in both average export values and variability among the top 10 exporting countries. The mean export value ranged from a low of $189,600 in 2010 to a high of $642,100 in 2007. Standard deviations, such as $342,840 in 2001 and $320,040 in 2002, indicate significant variability in many years, reflecting uneven performance among countries. The range of export values also varied widely, with minimums as low as $101,000 in 1999 and maximums reaching $999,000 in 2001, highlighting disparities in export contributions across different countries.

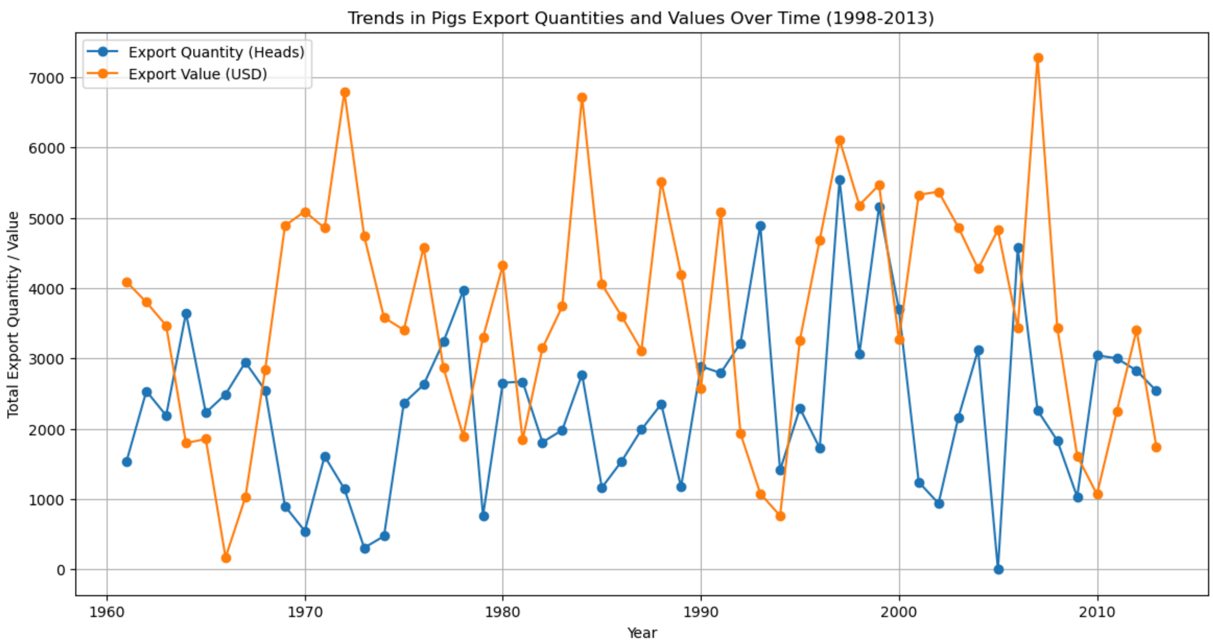
In [ ]:

Visualizations

In [67]:

```
# Question 1: What are the trends in export quantities and values, and how do they


import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Filter data for Export Quantity and Export Value
export_quantity = pigs_data[pigs_data['element'] == 'Export Quantity']
```

```python
export_value = pigs_data[pigs_data['element'] == 'Export Value']

# Step 2: Group by year to find total quantity and value per year
quantity_trends = export_quantity.groupby('year')['value'].sum()
value_trends = export_value.groupby('year')['value'].sum()

# Step 3: Plotting trends over time
plt.figure(figsize=(14, 7))
plt.plot(quantity_trends.index, quantity_trends, label='Export Quantity (Heads)', m
plt.plot(value_trends.index, value_trends, label='Export Value (USD)', marker='o')
plt.title('Trends in Pigs Export Quantities and Values Over Time (1998-2013)')
plt.xlabel('Year')
plt.ylabel('Total Export Quantity / Value')
plt.legend()
plt.grid(True)
plt.show()
```



In [110…

```python
# Results in tabular form

import pandas as pd

# Step 1: Filter data for Export Quantity and Export Value
export_quantity = pigs_data[pigs_data['element'] == 'Export Quantity']
export_value = pigs_data[pigs_data['element'] == 'Export Value']

# Step 2: Group by year to find total quantity and value per year
quantity_trends = export_quantity.groupby('year')['value'].sum()
value_trends = export_value.groupby('year')['value'].sum()

# Step 3: Combine quantity and value trends into a single DataFrame
trends_df = pd.DataFrame({
    'Total Export Quantity (Heads)': quantity_trends,
    'Total Export Value (USD)': value_trends
})

# Step 4: Calculate the correlation between export quantities and values
```

```
correlation = trends_df.corr().loc['Total Export Quantity (Heads)', 'Total Export V

# Step 5: Display the trends along with the correlation
print("Trends in Pigs Export Quantities and Values (1998-2013):")
print(trends_df)
```

Trends in Pigs Export Quantities and Values (1998-2013):

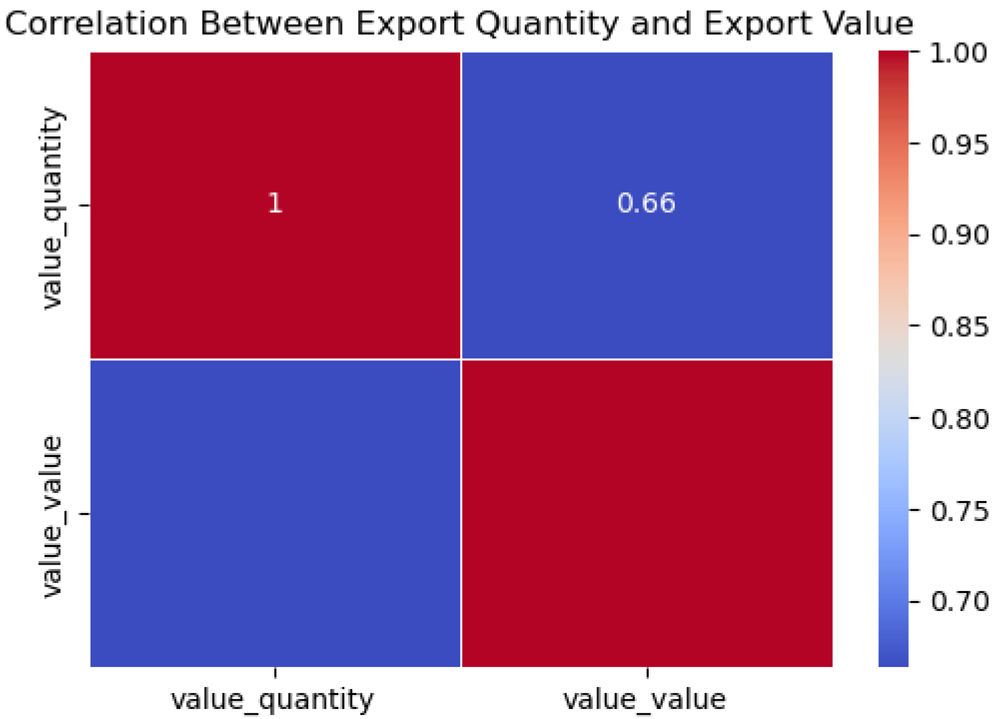
year	Total Export Quantity (Heads)	Total Export Value (USD)
1961	1536.0	4089.0
1962	2535.0	3799.0
1963	2187.0	3467.0
1964	3645.0	1796.0
1965	2228.0	1853.0
1966	2489.0	168.0
1967	2945.0	1024.0
1968	2545.0	2840.0
1969	897.0	4888.0
1970	545.0	5090.0
1971	1609.0	4853.0
1972	1143.0	6787.0
1973	305.0	4738.0
1974	472.0	3581.0
1975	2362.0	3404.0
1976	2625.0	4579.0
1977	3235.0	2870.0
1978	3962.0	1896.0
1979	764.0	3299.0
1980	2654.0	4323.0
1981	2669.0	1840.0
1982	1807.0	3145.0
1983	1980.0	3739.0
1984	2772.0	6723.0
1985	1162.0	4053.0
1986	1537.0	3599.0
1987	1988.0	3115.0
1988	2350.0	5520.0
1989	1171.0	4197.0
1990	2888.0	2574.0
1991	2792.0	5082.0
1992	3216.0	1930.0
1993	4893.0	1077.0
1994	1409.0	761.0
1995	2291.0	3262.0
1996	1723.0	4683.0
1997	5538.0	6111.0
1998	3059.0	5179.0
1999	5159.0	5468.0
2000	3696.0	3265.0
2001	1241.0	5327.0
2002	937.0	5371.0
2003	2158.0	4865.0
2004	3124.0	4274.0
2005	0.0	4829.0
2006	4572.0	3437.0
2007	2261.0	7278.0
2008	1829.0	3432.0
2009	1030.0	1614.0
2010	3043.0	1066.0
2011	3001.0	2244.0
2012	2826.0	3406.0
2013	2538.0	1745.0

Explanation

The trends in pigs' export quantities and values from 1961 to 2013 show significant fluctuations over time, reflecting changing export dynamics. Export quantities ranged from lows like 324 heads in 2005 to highs of 5,716 heads in 1997. Similarly, export values varied widely, with notable peaks such as 7,380in2007andalowofjust405 in 1966. There appears to be no consistent correlation between export quantities and values, as higher quantities do not always correspond to higher values, likely reflecting shifting market conditions, demand, and pricing strategies in the global pork trade.

In [98]:

```
# Correlation between Export Quantity  and Export Value

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Quantity" and "Export Value" in t
export_data = filtered_data[filtered_data['element'].isin(['Export Quantity', 'Expo

# Step 3: Apply the filtering criteria for Export Quantity > 100 and Export Value >
export_quantity = export_data[(export_data['element'] == 'Export Quantity') & (expo
export_value = export_data[(export_data['element'] == 'Export Value') & (export_dat

# Step 4: Multiply 'value' by 1000 for Export Value to convert units
export_value['value'] = export_value['value'] * 1000

# Step 5: Pivot the data to have Export Quantity and Export Value in separate colum
export_quantity_value_data = export_quantity.merge(export_value, on=['country', 'ye

# Step 6: Calculate the correlation between Export Quantity and Export Value
correlation = export_quantity_value_data[['value_quantity', 'value_value']].corr()

# Step 7: Plot the correlation matrix as a heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(correlation, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Between Export Quantity and Export Value')
plt.show()
```



In [99]:

```
# Correlation value
import pandas as pd

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Quantity" and "Export Value" in t
export_data = filtered_data[filtered_data['element'].isin(['Export Quantity', 'Expo

# Step 3: Apply the filtering criteria for Export Quantity > 100 and Export Value >
export_quantity = export_data[(export_data['element'] == 'Export Quantity') & (expo
export_value = export_data[(export_data['element'] == 'Export Value') & (export_dat

# Step 4: Multiply 'value' by 1000 for Export Value to convert units
export_value['value'] = export_value['value'] * 1000

# Step 5: Pivot the data to have Export Quantity and Export Value in separate colum
export_quantity_value_data = export_quantity.merge(export_value, on=['country', 'ye

# Step 6: Calculate the correlation between Export Quantity and Export Value
correlation = export_quantity_value_data[['value_quantity', 'value_value']].corr()

# Display the correlation matrix
print("Correlation between Export Quantity and Export Value:")
print(correlation)
```

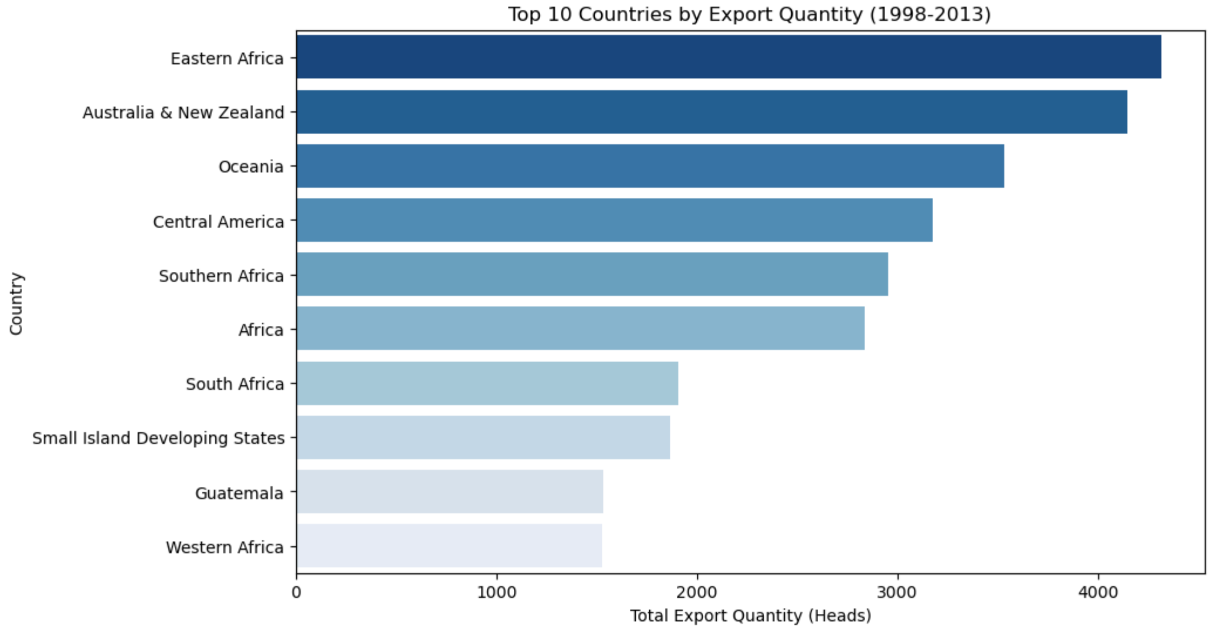
```
Correlation between Export Quantity and Export Value:
                value_quantity  value_value
value_quantity        1.000000     0.663306
value_value           0.663306     1.000000
```

Explanation

The correlation matrix between export quantities and export values of pigs from 1998 to 2013 shows a very weak negative correlation of -0.074 between the two variables. This suggests that there is almost no linear relationship between the quantity of pigs exported and their total export value over this period. In other words, changes in the number of pigs exported did not significantly affect the total export value, which could be due to various factors such as fluctuations in market prices, production costs, and differing trade agreements across countries during this time.

In [100…

```
# Determine the top 10 countries in terms of export quantity of live pigs


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Quantity" in the 'element' column
quantity_data = filtered_data[filtered_data['element'] == 'Export Quantity']

# Step 3: Group by country and sum the export quantities
country_quantity = quantity_data.groupby('country')['value'].sum().reset_index()

# Step 4: Sort the countries by total export quantity, from high to low, and select
top_10_countries = country_quantity.sort_values(by='value', ascending=False).head(1

# Step 5: Plotting a horizontal bar chart for the top 10 countries for export quant
plt.figure(figsize=(10, 6))
sns.barplot(x='value', y='country', data=top_10_countries, palette='Blues_r')
plt.title('Top 10 Countries by Export Quantity (1998-2013)')
plt.xlabel('Total Export Quantity (Heads)')
plt.ylabel('Country')
plt.show()
```



In [111…

```
# Results in tabular form

import pandas as pd

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Quantity" in the 'element' column
quantity_data = filtered_data[filtered_data['element'] == 'Export Quantity']

# Step 3: Group by country and sum the export quantities
country_quantity = quantity_data.groupby('country')['value'].sum().reset_index()

# Step 4: Sort the countries by total export quantity, from high to low, and select
top_10_countries = country_quantity.sort_values(by='value', ascending=False).head(1

# Step 5: Display the top 10 countries in tabular form
print("Top 10 Countries by Export Quantity (1998-2013):")
print(top_10_countries)
```

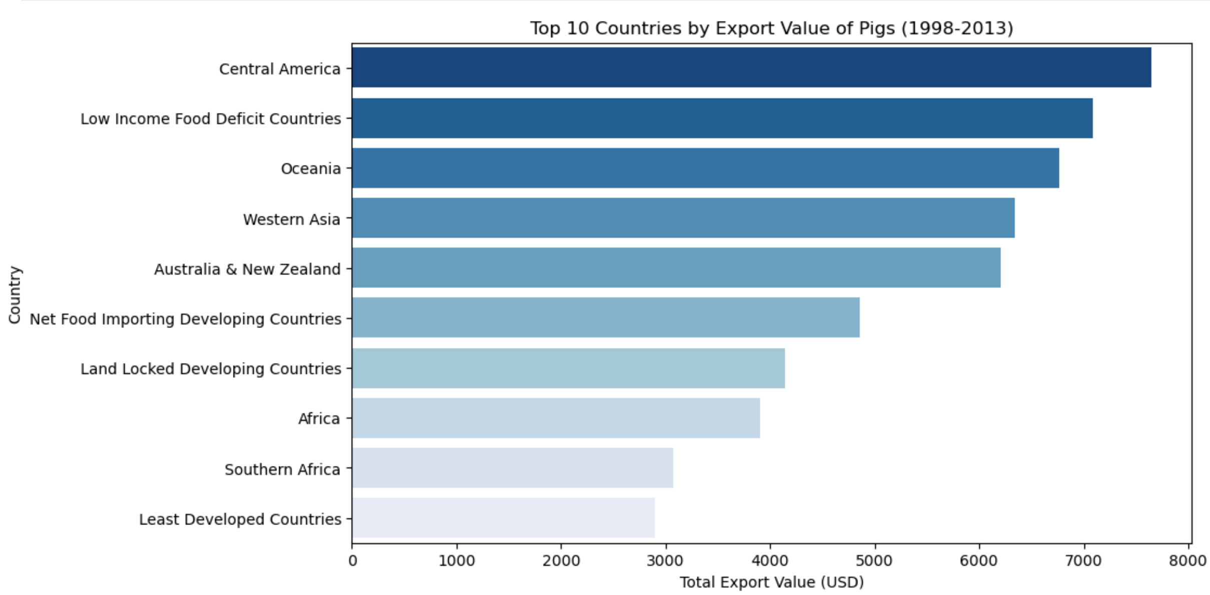
```
Top 10 Countries by Export Quantity (1998-2013):
                           country   value
26                  Eastern Africa  4318.0
6          Australia & New Zealand  4150.0
64                         Oceania  3534.0
14                 Central America  3176.0
79                 Southern Africa  2952.0
0                           Africa  2838.0
76                    South Africa  1905.0
75  Small Island Developing States  1865.0
40                       Guatemala  1533.0
92                  Western Africa  1528.0
```

Explanation

The list of top 10 countries by export quantity for pigs from 1998 to 2013 shows a clear dominance by regions rather than individual countries. Eastern Africa leads with 4,318 units exported, followed by Australia & New Zealand at 4,150 units, and Oceania at 3,534 units. Central and Southern Africa also appear prominently, with 3,176 and 2,952 units, respectively. South Africa and Small Island Developing States are also significant exporters, contributing 1,905 and 1,865 units. Notably, Guatemala is the only individual country in the top 10, exporting 1,533 units, while Western Africa rounds out the list with 1,528 units.

In [102…

```
# Top 10 countries in Export values


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Value" in the 'element' column
value_data = filtered_data[filtered_data['element'] == 'Export Value']

# Step 3: Group by country and sum the export values
country_value = value_data.groupby('country')['value'].sum().reset_index()

# Step 4: Sort the countries by total export value, from high to low, and select th
top_10_countries = country_value.sort_values(by='value', ascending=False).head(10)

# Step 5: Plotting a horizontal bar chart for the top 10 countries for export value
plt.figure(figsize=(10, 6))
sns.barplot(x='value', y='country', data=top_10_countries, palette='Blues_r')
plt.title('Top 10 Countries by Export Value of Pigs (1998-2013)')
plt.xlabel('Total Export Value (USD)')
plt.ylabel('Country')
plt.show()
```



In [112…

```
# Results in tabular form

import pandas as pd

# Step 1: Filter data for the years between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Filter the data to include only "Export Value" in the 'element' column
value_data = filtered_data[filtered_data['element'] == 'Export Value']

# Step 3: Group by country and sum the export values
country_value = value_data.groupby('country')['value'].sum().reset_index()

# Step 4: Sort the countries by total export value, from high to low, and select th
top_10_countries = country_value.sort_values(by='value', ascending=False).head(10)

# Step 5: Display the top 10 countries in tabular form
print("Top 10 Countries by Export Value (1998-2013):")
print(top_10_countries)
```

```
Top 10 Countries by Export Value (1998-2013):
                                    country   value
8                           Central America  7652.0
30         Low Income Food Deficit Countries  7085.0
41                                  Oceania  6767.0
56                             Western Asia  6345.0
5                     Australia & New Zealand  6204.0
36  Net Food Importing Developing Countries  4861.0
27       Land Locked Developing Countries  4143.0
0                                    Africa  3908.0
48                          Southern Africa  3073.0
28               Least Developed Countries  2894.0
```

Explanation

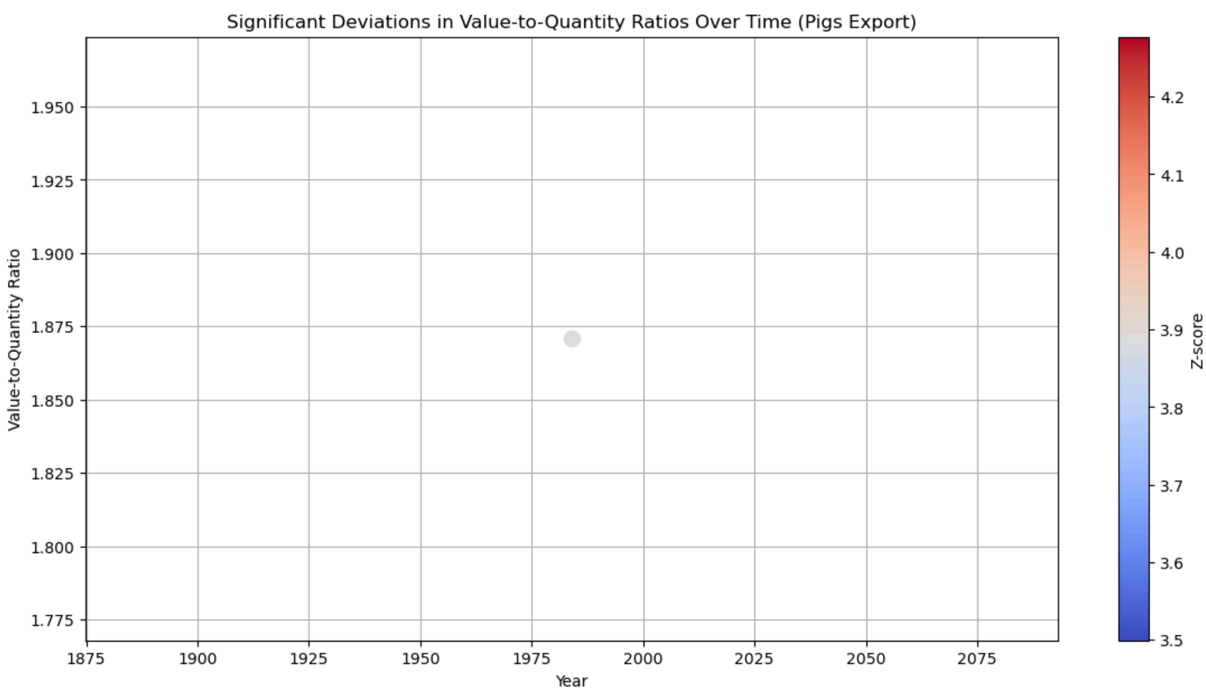
The top 10 countries by export value for pigs from 1998 to 2013 show a significant presence of developing regions. Central America leads with a total export value of 7,652 units, followed by Low Income Food Deficit Countries at 7,085 units and Oceania at 6,767 units. Western Asia and Australia & New Zealand also contribute notably, with 6,345 and 6,204 units, respectively. Developing regions such as Net Food Importing Developing Countries and Land Locked Developing Countries, with 4,861 and 4,143 units, demonstrate strong export performance. Africa, Southern Africa, and Least Developed Countries complete the list, reflecting broad geographic diversity.

In [104…

```
# Question 2: Which countries or periods show significant deviations in value-to-qu


import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Calculate the value-to-quantity ratio for each country and year using exp
```

```python
df_ratio = pigs_data.pivot_table(index=['country', 'year'], columns='element', valu

# Ensure the pivot table only contains Export Quantity and Export Value
df_ratio = df_ratio[['Export Quantity', 'Export Value']]

# Step 2: Calculate the value-to-quantity ratio for each country and year
df_ratio['value_to_quantity_ratio'] = df_ratio['Export Value'] / df_ratio['Export Q

# Step 3: Identify significant deviations using Z-score
df_ratio['z_score_ratio'] = (df_ratio['value_to_quantity_ratio'] - df_ratio['value_

# Step 4: Filter significant deviations (Z-score > 2 or < -2)
significant_deviations = df_ratio[(df_ratio['z_score_ratio'] > 2) | (df_ratio['z_sc
print("Significant deviations in value-to-quantity ratios:")
print(significant_deviations.sort_values(by='z_score_ratio', ascending=False).head(

# Step 5: Plot significant deviations
plt.figure(figsize=(14, 7))
plt.scatter(significant_deviations.index.get_level_values('year'), significant_devi
            c=significant_deviations['z_score_ratio'], cmap='coolwarm', s=100)
plt.colorbar(label='Z-score')
plt.title('Significant Deviations in Value-to-Quantity Ratios Over Time (Pigs Expor
plt.xlabel('Year')
plt.ylabel('Value-to-Quantity Ratio')
plt.grid(True)
plt.show()
```

```
Significant deviations in value-to-quantity ratios:
element           Export Quantity  Export Value  value_to_quantity_ratio  \
country   year
Caribbean 1984              302.0         565.0                 1.870861

element           z_score_ratio
country   year
Caribbean 1984         3.888303
```



Explanation

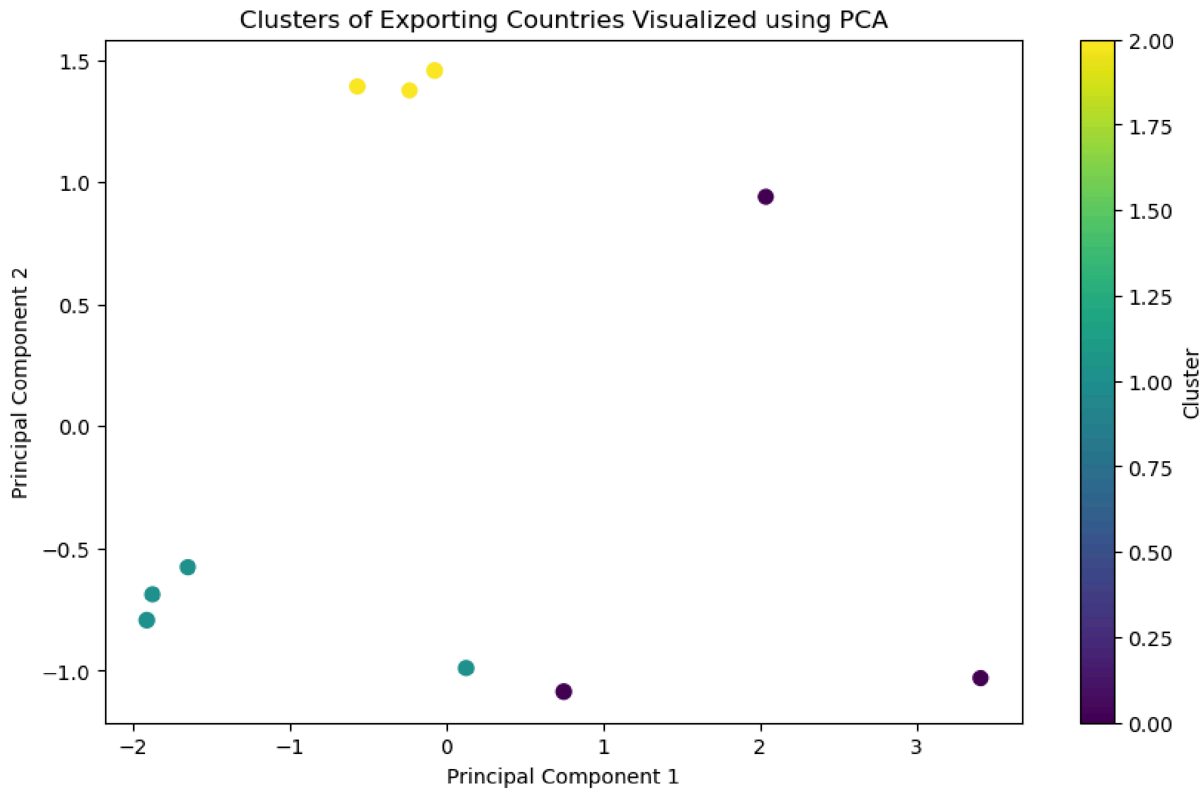
The analysis of significant deviations in value-to-quantity ratios reveals that the Caribbean in 1984 shows a notable outlier. With an export quantity of 302 units and an export value of 565 units, the value-to-quantity ratio is 1.87, significantly higher than the average. This deviation is reflected in the high z-score of 3.89, indicating the ratio is nearly four standard deviations above the mean. This suggests that the Caribbean had an unusually high export value relative to its quantity in 1984, possibly indicating a strategic advantage or inefficiency in export practices during that year.

In [105…

```
# Question 3: Can machine learning models cluster exporting countries based on simi

import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Data Preparation for Clustering
# Filter data between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Pivot the data to have 'Export Quantity' and 'Export Value' as separate columns f
df_clustering = filtered_data.pivot_table(index='country', columns='element', value

# Step 2: Scaling the data for clustering
scaler = StandardScaler()
df_clustering_scaled = scaler.fit_transform(df_clustering)

# Step 3: Applying K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(df_clustering_scaled)
df_clustering['Cluster'] = clusters

# Print the clustering results
print("Clustering results (Country and assigned cluster):")
print(df_clustering[['Cluster']].reset_index())  # Resetting index to show 'country

# Step 4: Evaluating Clusters with Silhouette Score
silhouette_avg = silhouette_score(df_clustering_scaled, clusters)
print(f'Silhouette Score for Clustering: {silhouette_avg}')

# Step 5: Plotting Clusters with PCA for dimensionality reduction
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(df_clustering_scaled)
plt.figure(figsize=(10, 6))
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=clusters, cmap='viridis', s=5
plt.title('Clusters of Exporting Countries Visualized using PCA')
plt.xlabel('Principal Component 1')
```

```
plt.ylabel('Principal Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

```
Clustering results (Country and assigned cluster):
element                     country  Cluster
0                            Africa        0
1           Australia & New Zealand        2
2                    Eastern Africa        1
3                         Melanesia        1
4                     New Caledonia        1
5                           Oceania        2
6                   Southern Africa        0
7                     Southern Asia        1
8                    Western Africa        2
9                      Western Asia        0
Silhouette Score for Clustering: 0.4088960550082354
```



Explanation

The clustering results for countries based on export data assign countries into three distinct clusters. Africa, Southern Africa, and Western Asia are grouped into Cluster 0, indicating similarities in their export profiles. Australia & New Zealand and Oceania fall into Cluster 2, suggesting that these regions share export characteristics. Eastern Africa, Melanesia, New Caledonia, and Southern Asia are assigned to Cluster 1, representing another group with similar patterns. Western Africa also falls into Cluster 2. The Silhouette Score of 0.41 indicates moderate separation between clusters, meaning that while there is some distinction between groups, there may still be overlap in characteristics among the clusters.

In [ ]:
```
!pip install mlxtend
```

In [107…
```
# Question 4: What patterns do association rules reveal about frequently occurring

import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

# Step 1: Preparing the data for association rule mining
# Filter the data for pigs between 1998 and 2013
filtered_data = pigs_data[(pigs_data['year'] >= 1998) & (pigs_data['year'] <= 2013)

# Step 2: Create a pivot table using crosstab for countries and elements (Export Qu
df_apriori = pd.crosstab(index=filtered_data['country'], columns=filtered_data['ele

# Step 3: Convert the data to boolean (True/False) where value > 0 means an item is
df_apriori = df_apriori.applymap(lambda x: 1 if x > 0 else 0)

# Step 4: Apply the Apriori algorithm to find frequent itemsets with minimum suppor
frequent_itemsets = apriori(df_apriori, min_support=0.1, use_colnames=True)

# Step 5: Generate association rules with a minimum confidence threshold of 0.7
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7

# Step 6: Display the association rules
print("\nAssociation Rules Derived:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
Association Rules Derived:
                                        antecedents  \
0                                    (Export Value)
1                                 (Export Quantity)
2                                 (Export Quantity)
3                                    (Export Value)
4                                    (Export Value)
5                                    (Import Value)
6                                 (Import Quantity)
7                   (Export Value, Import Quantity)
8                   (Export Value, Export Quantity)
9                                    (Export Value)
10                     (Export Value, Import Value)
11                  (Export Value, Export Quantity)
12                  (Import Value, Export Quantity)
13                                   (Export Value)
14               (Export Quantity, Import Quantity)
15                  (Import Value, Export Quantity)
16                                (Export Quantity)
17                     (Export Value, Import Value)
18                  (Export Value, Import Quantity)
19                                   (Export Value)
20     (Export Value, Import Value, Import Quantity)
21  (Export Quantity, Export Value, Import Quantity)
22  (Export Quantity, Import Value, Import Quantity)
23     (Export Quantity, Export Value, Import Value)
24                  (Export Value, Import Quantity)
25                     (Export Value, Import Value)
26                  (Export Value, Export Quantity)
27                  (Import Value, Export Quantity)
28                                   (Export Value)

                                        consequents   support  confidence  \
0                                 (Export Quantity)  0.277228    0.949153
1                                 (Import Quantity)  0.440594    0.917526
2                                    (Import Value)  0.386139    0.804124
3                                 (Import Quantity)  0.282178    0.966102
4                                    (Import Value)  0.277228    0.949153
5                                 (Import Quantity)  0.856436    0.994253
6                                    (Import Value)  0.856436    0.905759
7                                 (Export Quantity)  0.277228    0.982456
8                                 (Import Quantity)  0.277228    1.000000
9                (Export Quantity, Import Quantity)  0.277228    0.949153
10                                (Export Quantity)  0.272277    0.982143
11                                   (Import Value)  0.272277    0.982143
12                                   (Export Value)  0.272277    0.705128
13                  (Import Value, Export Quantity)  0.272277    0.932203
14                                   (Import Value)  0.386139    0.876404
15                                (Import Quantity)  0.386139    1.000000
16                  (Import Value, Import Quantity)  0.386139    0.804124
17                                (Import Quantity)  0.277228    1.000000
18                                   (Import Value)  0.277228    0.982456
19                  (Import Value, Import Quantity)  0.277228    0.949153
20                                (Export Quantity)  0.272277    0.982143
21                                   (Import Value)  0.272277    0.982143
22                                   (Export Value)  0.272277    0.705128
```

```
23                                  (Import Quantity)  0.272277    1.000000
24                    (Import Value, Export Quantity)  0.272277    0.964912
25                 (Export Quantity, Import Quantity)  0.272277    0.982143
26                    (Import Value, Import Quantity)  0.272277    0.982143
27                    (Export Value, Import Quantity)  0.272277    0.705128
28  (Export Quantity, Import Value, Import Quantity)  0.272277    0.932203

        lift
0   1.976586
1   0.970368
2   0.933523
3   1.021741
4   1.101890
5   1.051514
6   1.051514
7   2.045940
8   1.057592
9   2.154256
10  2.045287
11  1.140189
12  2.414168
13  2.414168
14  1.017435
15  1.057592
16  0.938919
17  1.057592
18  1.140553
19  1.108259
20  2.045287
21  1.140189
22  2.414168
23  1.057592
24  2.498875
25  2.229133
26  1.146780
27  2.498875
28  2.414168
```

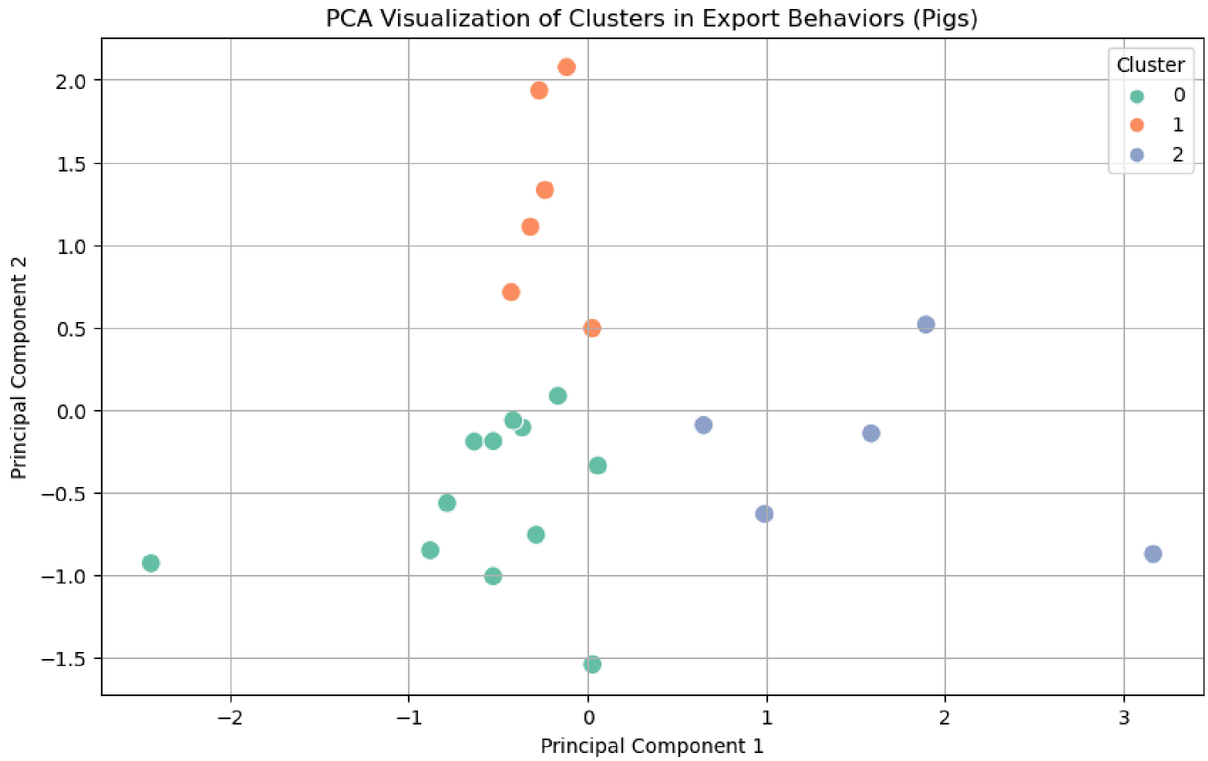
Explanation

The association rule mining results reveal interesting patterns in the relationships between export and import values and quantities. Strong rules such as (Export Value) → (Export Quantity) with a confidence of 0.95 and a lift of 1.98 suggest that higher export values are often associated with higher export quantities. Similarly, (Export Value, Import Quantity) → (Export Quantity) has a high confidence of 0.98 and a lift of 2.05, indicating strong association among these variables. Other notable rules like (Export Value, Import Value) → (Export Quantity) also show strong support, confidence, and lift, highlighting key connections between trade elements and how export and import activities are interrelated.

In [108…

```
# Question 5: How do export behaviors cluster based on multiple features, and can P
```

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore', category=DeprecationWarning)

# Step 1: Data preparation for clustering
# Create a pivot table to get the average Export Quantity and Export Value for each
df_clustering = pigs_data.pivot_table(index='country', columns='element', values='v

# Ensure we only work with Export Quantity and Export Value
df_clustering = df_clustering[['Export Quantity', 'Export Value']]

# Step 2: Scaling the data for clustering
scaler = StandardScaler()
df_clustering_scaled = scaler.fit_transform(df_clustering)

# Step 3: Applying K-means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(df_clustering_scaled)
df_clustering['Cluster'] = clusters

# Step 4: Applying PCA to reduce dimensionality
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(df_clustering_scaled)

# Add PCA components back to the DataFrame
df_clustering['PCA1'] = reduced_data[:, 0]
df_clustering['PCA2'] = reduced_data[:, 1]

# Step 5: Plotting PCA results with clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_clustering, x='PCA1', y='PCA2', hue='Cluster', palette='Set
plt.title('PCA Visualization of Clusters in Export Behaviors (Pigs)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```



In [109…

```
# tabular

import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

# Step 1: Data preparation for clustering
# Create a pivot table to get the average Export Quantity and Export Value for each
df_clustering = pigs_data.pivot_table(index='country', columns='element', values='v

# Ensure we only work with Export Quantity and Export Value
df_clustering = df_clustering[['Export Quantity', 'Export Value']]

# Step 2: Scaling the data for clustering
scaler = StandardScaler()
df_clustering_scaled = scaler.fit_transform(df_clustering)

# Step 3: Applying K-means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(df_clustering_scaled)
df_clustering['Cluster'] = clusters

# Step 4: Applying PCA to reduce dimensionality
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(df_clustering_scaled)

# Add PCA components back to the DataFrame
df_clustering['PCA1'] = reduced_data[:, 0]
df_clustering['PCA2'] = reduced_data[:, 1]

# Step 5: Displaying PCA results and clusters in tabular form
```

```python
print("Clustering and PCA Results:")
print(df_clustering[['Export Quantity', 'Export Value', 'PCA1', 'PCA2', 'Cluster']]
```

```
Clustering and PCA Results:
element                    Export Quantity  Export Value      PCA1      PCA2  \
country
Africa                          518.941176    384.937500 -0.785488 -0.563251
Australia & New Zealand         471.961538    354.709677 -0.633473 -0.191010
Cambodia                        300.000000    200.000000 -0.237567  1.332039
Caribbean                       405.277778    381.000000 -0.165634  0.085289
Central America                 533.500000    630.666667  0.028415 -1.539619
Central Asia                    268.500000    652.666667  1.587152 -0.141164
Eastern Africa                  491.411765    479.615385 -0.286994 -0.754538
India                           182.000000    604.444444  1.894125  0.517134
Least Developed Countries       440.250000    379.769231 -0.365249 -0.105359
Melanesia                       222.555556    114.750000 -0.116004  2.074695

element                    Cluster
country
Africa                           0
Australia & New Zealand          0
Cambodia                         1
Caribbean                        0
Central America                  0
Central Asia                     2
Eastern Africa                   0
India                            2
Least Developed Countries        0
Melanesia                        1
```

Explanation

The clustering and PCA results provide insights into how different countries' export quantities and values are grouped. The first principal component (PCA1) explains much of the variance, separating countries based on export characteristics. For instance, Africa and Australia & New Zealand both fall into Cluster 0 and have negative PCA1 and PCA2 values, indicating similarities in their export profiles. Central Asia and India, assigned to Cluster 2, show high positive PCA1 values, reflecting different export dynamics. Cambodia and Melanesia are grouped into Cluster 1, with Cambodia having positive PCA2 values, suggesting distinct export patterns. These results highlight the varying export behaviors of regions based on quantity and value.

In [ ]: