NZSE

New Zealand Skills and Education College

School of Tech

Bachelor of Business Information Management (Level 6)

Cover Sheet and Student Declaration

This sheet must be signed by the student and attached to the submitted assessment.

Course Title:	**Data Transformation and Management**	**Course code:**	**BBIM612**
Student Name:	Victor Viki	**Student ID:**	764706455
Assessment No & Type:	Assessment 1 - Project 1	**Cohort:**	BBIM7123C
Due Date:	20/12/24	**Date Submitted:**	20/12/24
Tutor's Name:	Giang Mai		
Assessment Weighting:	60%		
Total Marks:	100		

Student Declaration:

I declare that:

- I have read the New Zealand School of Education Ltd policies and regulations on assessments and understand what plagiarism is.
- I am aware of the penalties for cheating and plagiarism as laid down by the New Zealand School of Education Ltd.
- This is an original assessment and is entirely my own work.
- Where I have quoted or made use of the ideas of other writers, I have acknowledged the source.
- This assessment has been prepared exclusively for this course and has not been or will not be submitted as assessed work in any other course.
- It has been explained to me that this assessment may be used by NZSE Ltd, for internal and/or external moderation.

Student signature:

Date:

Tutor only to complete			
	Part A **(max. 20 marks)**	**Part B** **(max. 35 marks)**	**Part C** **(max. 45 marks)**
Assessment result:	**Marks: /100**	**Grade:**	

LO1 Requirements	□ **Met** □ **Not Met**	**Assessor signature:**
LO2 Requirements	□ **Met** □ **Not Met**	

BBIM612 DATA TRANSFORMATION AND MANAGEMENT

BBIM7123C

DECEMBER 20, 2024

VICTOR VIKI
764706455

Table of Contents

PART A - Data Collection

Business Objective

My business objective is to identify if there is a relationship between a country's population, GDP, and exports, as well as to see if the population and exports, have a significant impact on a country's GDP.

Task 1: Identify Data Sources

Website 1: List of Countries by Exports

This website includes the list of countries by exports. The data highlights the goods and services measured in US million, country names, exports in revenue, the year, and top export for 2021. This website is relevant to my scenario to inquire if the exports have an influence on the relationship between the population and GDP (Wikipedia, List of countries by exports, 2024).

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

Website 2: List of Countries and Dependencies by Population

This website contains data based on the list of countries by population within the United Nations. It is based on estimates published by the UN for 2024, highlighting the Population for 2022 and 2023, the change in percentage, and the region and subregion. It aligns with my scenario where the population for each country is to be compared with the other two sources based on exports and GDP produced (Wikipedia, List of countries by population (United Nations), 2024).

https://en.wikipedia.org/wiki/List_of_countries_by_population_(United_Nations)

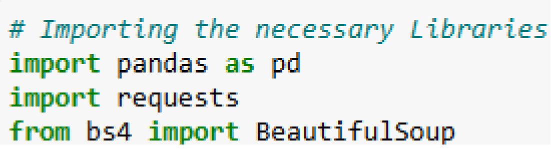
Website 3: List of Countries by GDP (nominal)

This website contains data in relation to the list of countries by GDP. The dataset contains 180 countries and their respective names, IMF's forecast and year, World Bank's estimate and year, and the United Nations's estimate and year. This website is crucial to my scenario as to research in whether the population and exports have an impact on a countries GDP (Wikipedia, List of countries by GDP (nominal), 2024).

https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)

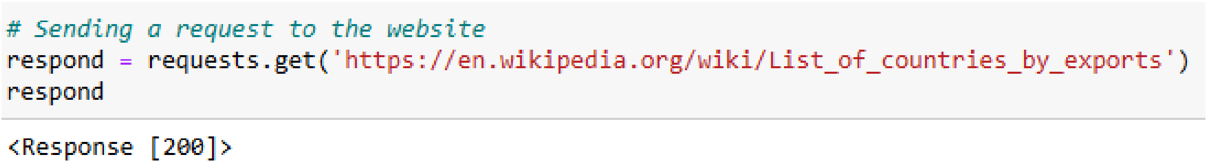
Task 2: Web Scraping

Website 1: List of Countries by Exports



```
# Importing the necessary Libraries
import pandas as pd
import requests
from bs4 import BeautifulSoup
```

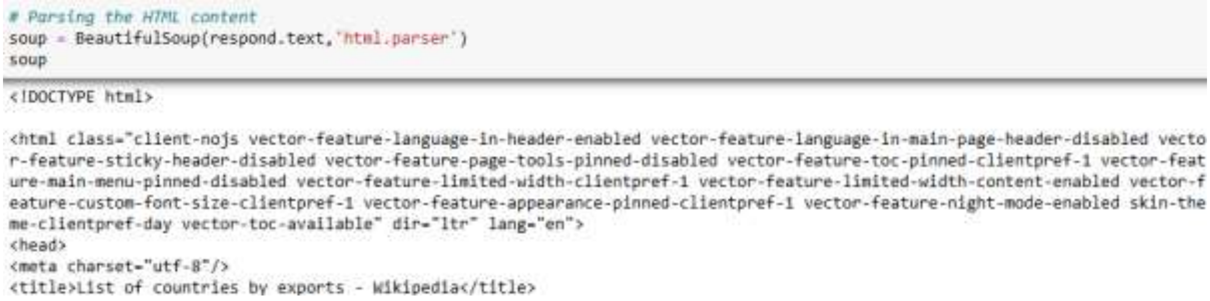
Here I am using Python to import the relevant libraries needed to perform the web scraping techniques. Pandas is used for working with dataset where it contains functions used for data cleaning, analysing, and manipulation.



```
# Sending a request to the website
respond = requests.get('https://en.wikipedia.org/wiki/List_of_countries_by_exports')
respond
```

```
<Response [200]>
```

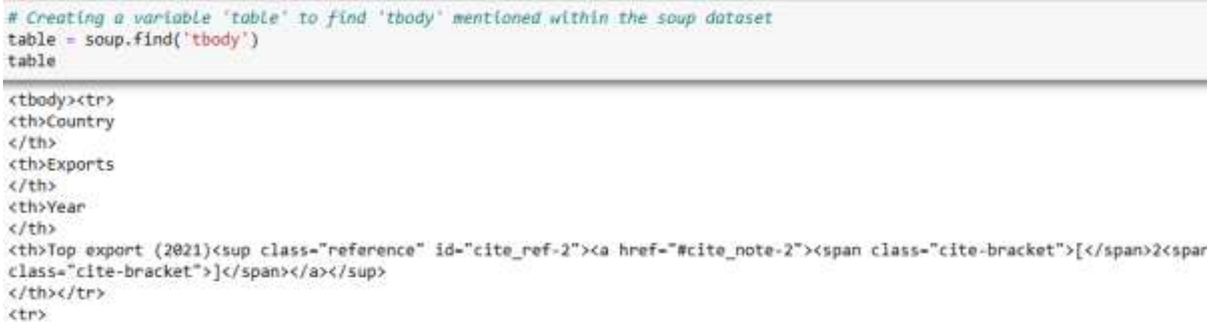
Here I am sending a request to the website to extract the dataset I am interested in. I have used this line of code to retrieve and print the results from the website to ensure if the retrieval was successful or not. The response of 200 indicates that it was successful.



```
# Parsing the HTML content
soup = BeautifulSoup(respond.text,'html.parser')
soup
```

```
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vecto
r-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feat
ure-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-limited-width-content-enabled vector-f
eature-custom-font-size-clientpref-1 vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-the
me-clientpref-day vector-toc-available" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of countries by exports - Wikipedia</title>
```

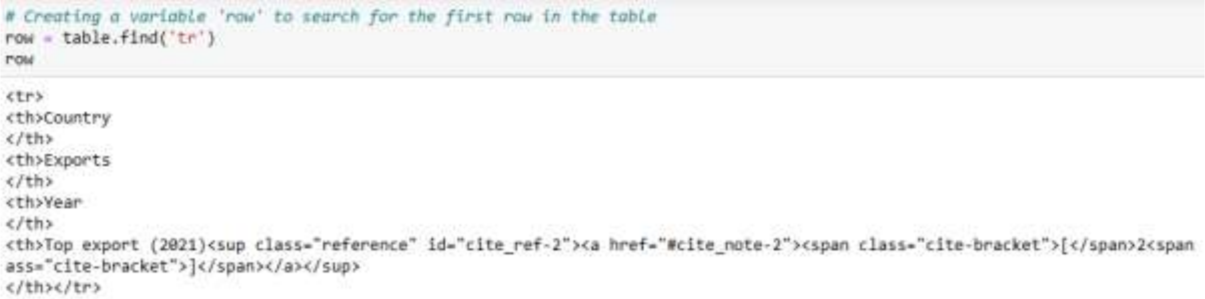
I am parsing the HTML, which is the process of examining and collecting the necessary data from unstructured, raw data that has been scraped from websites. The "Beautiful Soup" library, which is used to parse HTML and XML documents, is utilised for this. To extract data from HTML and XML in an organised and usable manner, it generates a parse tree for processed pages. The extracted data has been assigned under the variable ‘soup’.



```
# Creating a variable 'table' to find 'tbody' mentioned within the soup dataset
table = soup.find('tbody')
table
```

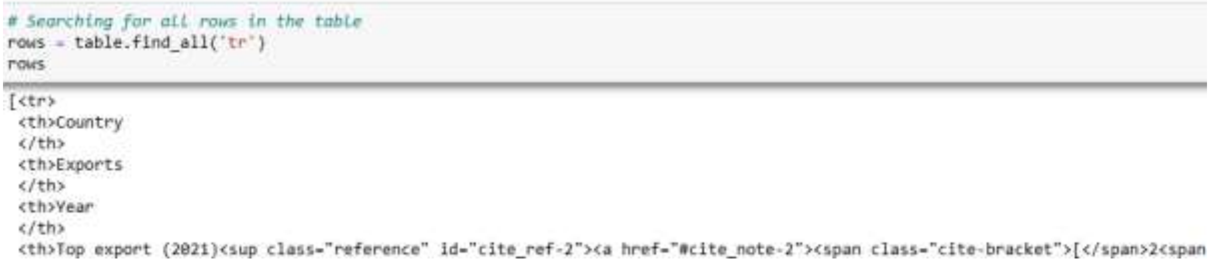
```
<tbody><tr>
<th>Country
</th>
<th>Exports
</th>
<th>Year
</th>
<th>Top export (2021)<sup class="reference" id="cite_ref-2"><a href="#cite_note-2"><span class="cite-bracket">[</span>2<span
class="cite-bracket">]</span></a></sup>
</th></tr>
<tr>
```

Here, I'm creating a variable named "table" to search for the first "tbody" that is mentioned in the soup data. The "find" method searches the parsed HTML content for the first matching element, stops when it finds one, and then returns it. The result has been shown.



```
# Creating a variable 'row' to search for the first row in the table
row = table.find('tr')
row
```

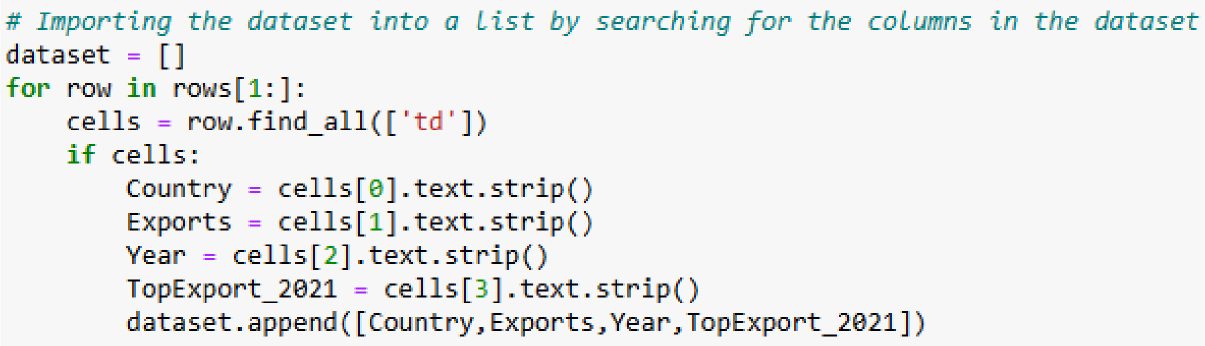
```
<tr>
<th>Country
</th>
<th>Exports
</th>
<th>Year
</th>
<th>Top export (2021)<sup class="reference" id="cite_ref-2"><a href="#cite_note-2"><span class="cite-bracket">[</span>2<span
ass="cite-bracket">]</span></a></sup>
</th></tr>
```



```
# Searching for all rows in the table
rows = table.find_all('tr')
rows
```

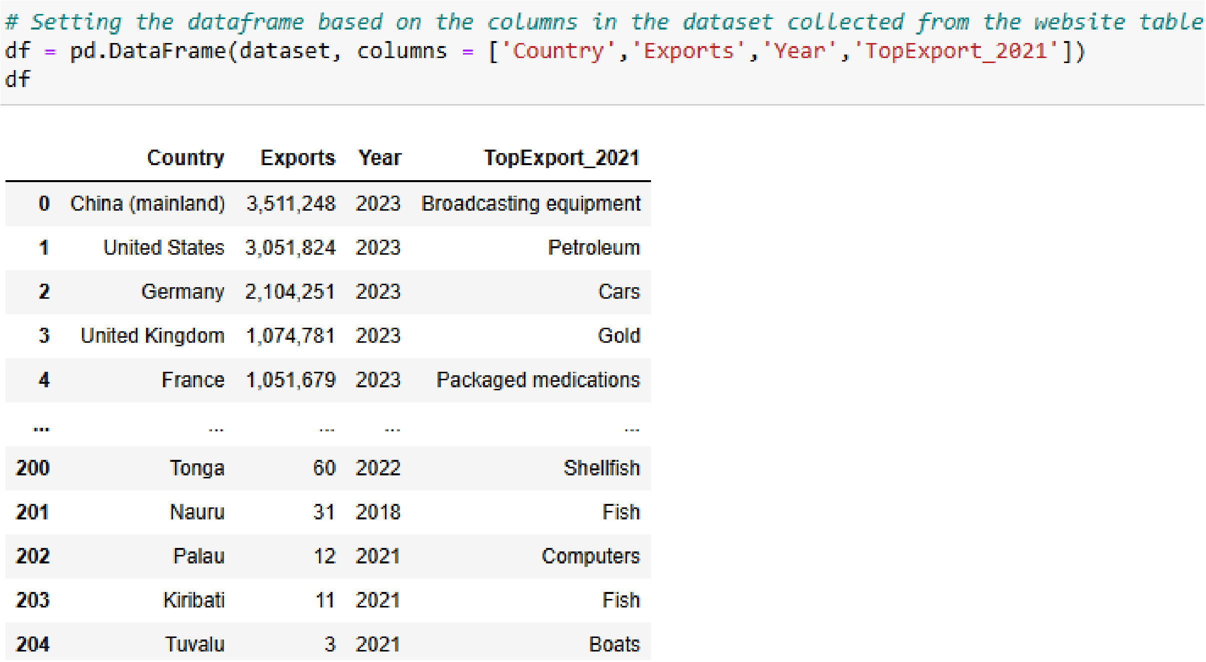
```
[<tr>
 <th>Country
 </th>
 <th>Exports
 </th>
 <th>Year
 </th>
 <th>Top export (2021)<sup class="reference" id="cite_ref-2"><a href="#cite_note-2"><span class="cite-bracket">[</span>2<span
```

These lines of code were applied to extract the data from the HTML and assign it in the variable names row and rows. By using the ‘find’ method in the first line of code, it will find the first row in the table, whereas the ‘find all’ method will search for all rows existent in a table.



```
# Importing the dataset into a list by searching for the columns in the dataset
dataset = []
for row in rows[1:]:
    cells = row.find_all(['td'])
    if cells:
        Country = cells[0].text.strip()
        Exports = cells[1].text.strip()
        Year = cells[2].text.strip()
        TopExport_2021 = cells[3].text.strip()
        dataset.append([Country,Exports,Year,TopExport_2021])
```

This block of code is used to import the dataset into a list, search for columns and rows, extract and clean the data. The data rows and column headers from the webpage are extracted into the specified variables using the "text strip" method, which also removes any white spaces and lines that may be present between them. The "append" technique is then used to combine the new dataset into a list.

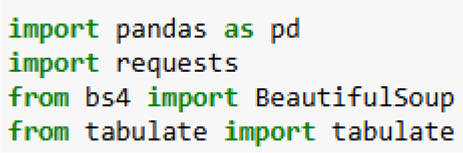


```
# Setting the dataframe based on the columns in the dataset collected from the website table
df = pd.DataFrame(dataset, columns = ['Country','Exports','Year','TopExport_2021'])
df
```

	Country	Exports	Year	TopExport_2021
0	China (mainland)	3,511,248	2023	Broadcasting equipment
1	United States	3,051,824	2023	Petroleum
2	Germany	2,104,251	2023	Cars
3	United Kingdom	1,074,781	2023	Gold
4	France	1,051,679	2023	Packaged medications
...	...	...	...	...
200	Tonga	60	2022	Shellfish
201	Nauru	31	2018	Fish
202	Palau	12	2021	Computers
203	Kiribati	11	2021	Fish
204	Tuvalu	3	2021	Boats

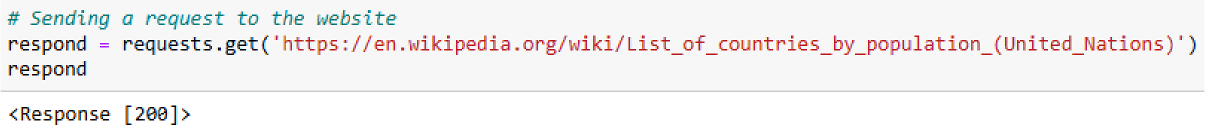
Here I am creating a DataFrame based on the list of columns in the dataset and displaying the output of the data in tabulation form.

Website 2: List of Countries and Dependencies by Population



```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from tabulate import tabulate
```

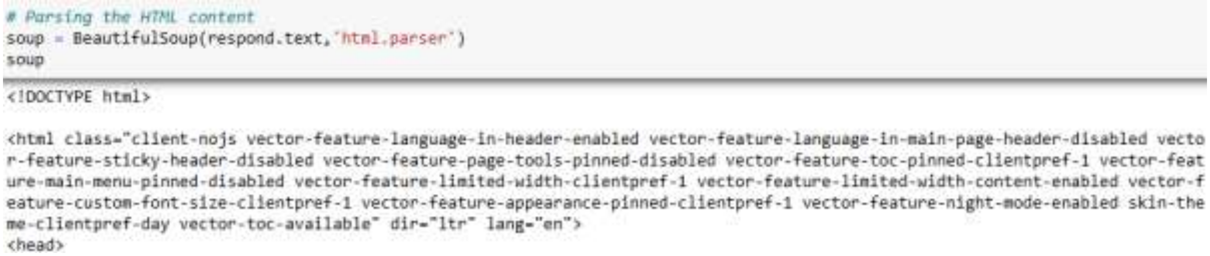
Here I am using Python to import the relevant libraries needed to perform the web scraping techniques for data cleaning, analysing, and manipulation.



```
# Sending a request to the website
respond = requests.get('https://en.wikipedia.org/wiki/List_of_countries_by_population_(United_Nations)')
respond
```

```
<Response [200]>
```

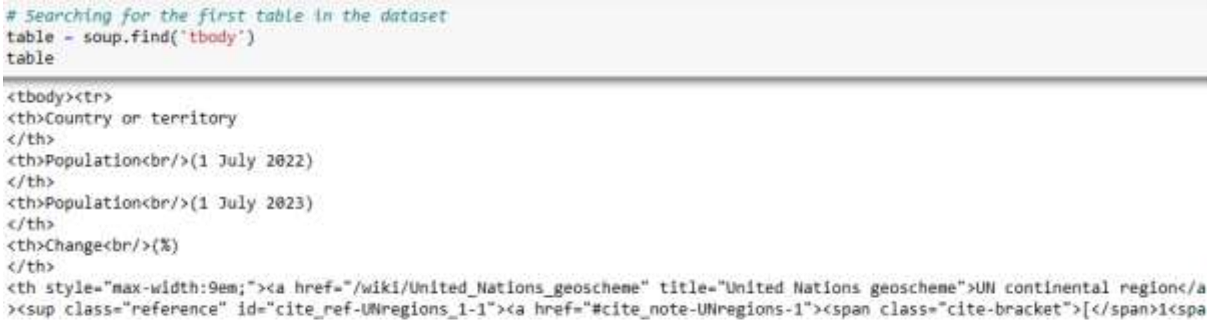
Here I am sending a request to the website to extract the dataset I am interested in I have received a response of 200 to ensure that the retrieval was successful.



```
# Parsing the HTML content
soup = BeautifulSoup(respond.text,'html.parser')
soup
```

```
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vecto
r-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feat
ure-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-limited-width-content-enabled vector-f
eature-custom-font-size-clientpref-1 vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-the
me-clientpref-day vector-toc-available" dir="ltr" lang="en">
<head>
```

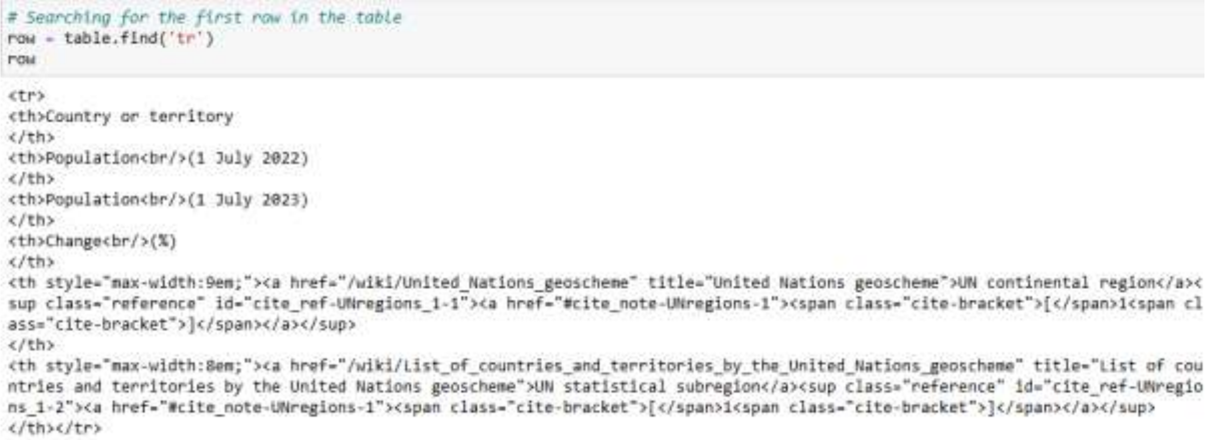
I am parsing the HTML, which is the process of examining and collecting the necessary data from unstructured, raw data that has been scraped from websites. The extracted data has been assigned under the variable ‘soup’.



```
# Searching for the first table in the dataset
table = soup.find('tbody')
table
```

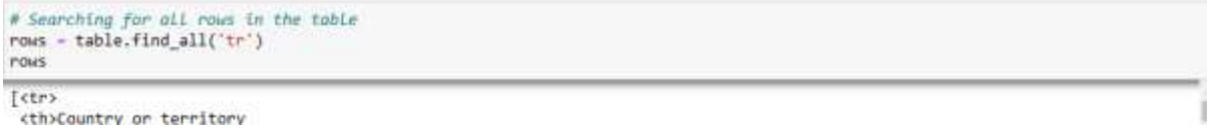
```
<tbody><tr>
<th>Country or territory
</th>
<th>Population<br/>(1 July 2022)
</th>
<th>Population<br/>(1 July 2023)
</th>
<th>Change<br/>(%)
</th>
<th style="max-width:9em;"><a href="/wiki/United_Nations_geoscheme" title="United Nations geoscheme">UN continental region</a
><sup class="reference" id="cite_ref-UNregions_1-1"><a href="#cite_note-UNregions-1"><span class="cite-bracket">[</span>1<spa
```

Here, I'm creating a variable named "table" to search for the first "tbody" that is mentioned in the soup data. The result has been shown.



```
# Searching for the first row in the table
row = table.find('tr')
row
```

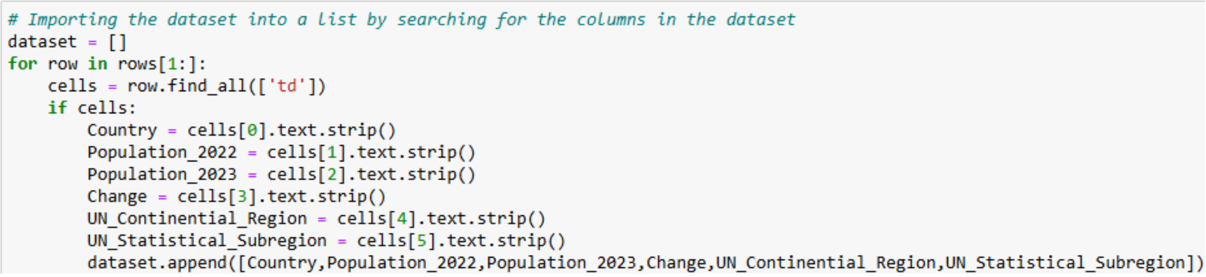
```
<tr>
<th>Country or territory
</th>
<th>Population<br/>(1 July 2022)
</th>
<th>Population<br/>(1 July 2023)
</th>
<th>Change<br/>(%)
</th>
<th style="max-width:9em;"><a href="/wiki/United_Nations_geoscheme" title="United Nations geoscheme">UN continental region</a><
sup class="reference" id="cite_ref-UNregions_1-1"><a href="#cite_note-UNregions-1"><span class="cite-bracket">[</span>1<span cl
ass="cite-bracket">]</span></a></sup>
</th>
<th style="max-width:8em;"><a href="/wiki/List_of_countries_and_territories_by_the_United_Nations_geoscheme" title="List of cou
ntries and territories by the United Nations geoscheme">UN statistical subregion</a><sup class="reference" id="cite_ref-UNregio
ns_1-2"><a href="#cite_note-UNregions-1"><span class="cite-bracket">[</span>1<span class="cite-bracket">]</span></a></sup>
</th></tr>
```



```
# Searching for all rows in the table
rows = table.find_all('tr')
rows
```

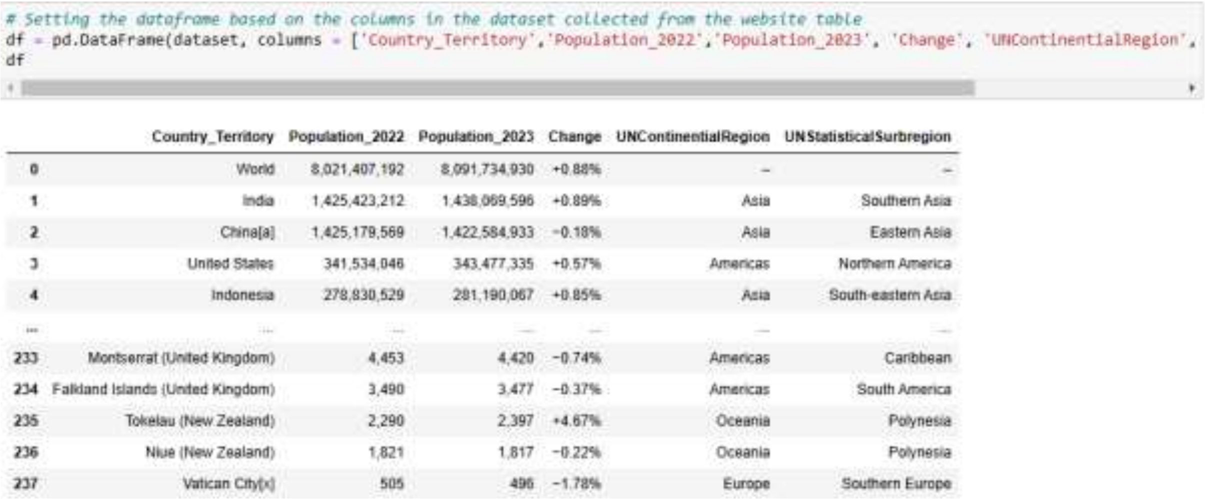
```
[<tr>
 <th>Country or territory
```

These lines of code were applied to extract the data from the HTML and assign it in the variable names row and rows to search for all the rows existent in a table.



```
# Importing the dataset into a list by searching for the columns in the dataset
dataset = []
for row in rows[1:]:
    cells = row.find_all(['td'])
    if cells:
        Country = cells[0].text.strip()
        Population_2022 = cells[1].text.strip()
        Population_2023 = cells[2].text.strip()
        Change = cells[3].text.strip()
        UN_Continential_Region = cells[4].text.strip()
        UN_Statistical_Subregion = cells[5].text.strip()
        dataset.append([Country,Population_2022,Population_2023,Change,UN_Continential_Region,UN_Statistical_Subregion])
```

This block of code is used to import the dataset into a list, search for columns and rows, extract and clean the data. The data rows and column headers from the webpage are extracted into the specified variables, and then combined as a new dataset into a list.

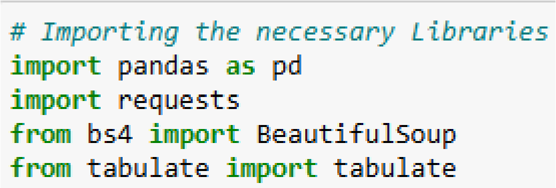


```
# Setting the dataframe based on the columns in the dataset collected from the website table
df = pd.DataFrame(dataset, columns = ['Country_Territory','Population_2022','Population_2023', 'Change', 'UNContinentialRegion',
df
```

	Country_Territory	Population_2022	Population_2023	Change	UNContinentialRegion	UNStatisticalSurbregion
0	World	8,021,407,192	8,091,734,930	+0.88%	–	–
1	India	1,425,423,212	1,438,069,596	+0.89%	Asia	Southern Asia
2	China[a]	1,425,179,569	1,422,584,933	−0.18%	Asia	Eastern Asia
3	United States	341,534,046	343,477,335	+0.57%	Americas	Northern America
4	Indonesia	278,830,529	281,190,067	+0.85%	Asia	South-eastern Asia
...	...	...	...	...	...	...
233	Montserrat (United Kingdom)	4,453	4,420	−0.74%	Americas	Caribbean
234	Falkland Islands (United Kingdom)	3,490	3,477	−0.37%	Americas	South America
235	Tokelau (New Zealand)	2,290	2,397	+4.67%	Oceania	Polynesia
236	Niue (New Zealand)	1,821	1,817	−0.22%	Oceania	Polynesia
237	Vatican City[x]	505	496	−1.78%	Europe	Southern Europe

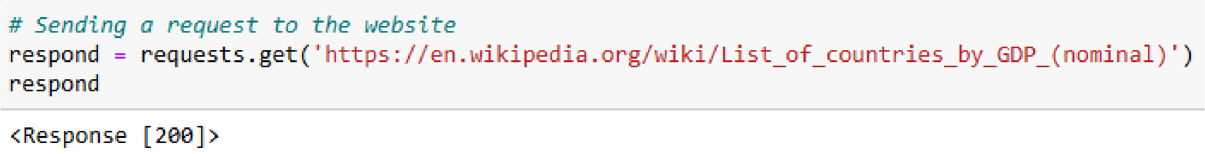
Here I am creating a DataFrame based on the list of columns in the dataset and displaying the output of the data in tabulation form.

Website 3: List of Countries by GDP (nominal)



```
# Importing the necessary Libraries
import pandas as pd
import requests
from bs4 import BeautifulSoup
from tabulate import tabulate
```

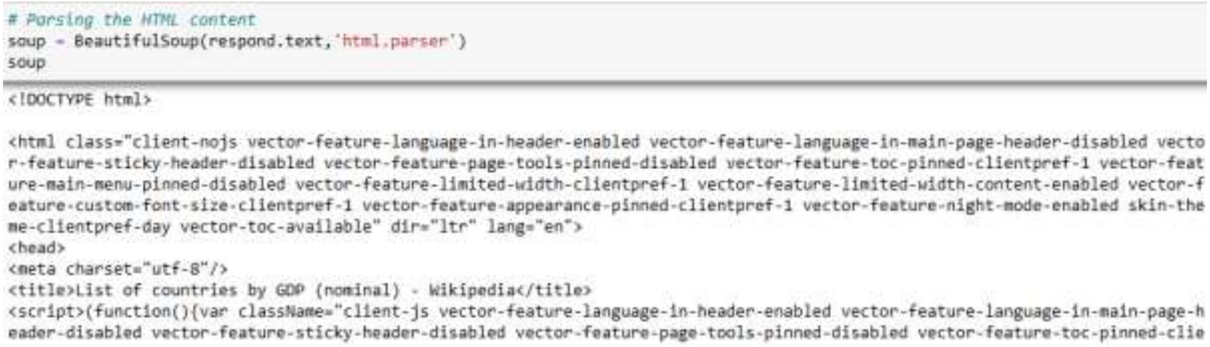
Here I am using Python to import the relevant libraries needed to perform the web scraping techniques for data cleaning, analysing, and manipulation.



```
# Sending a request to the website
respond = requests.get('https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)')
respond
```

```
<Response [200]>
```

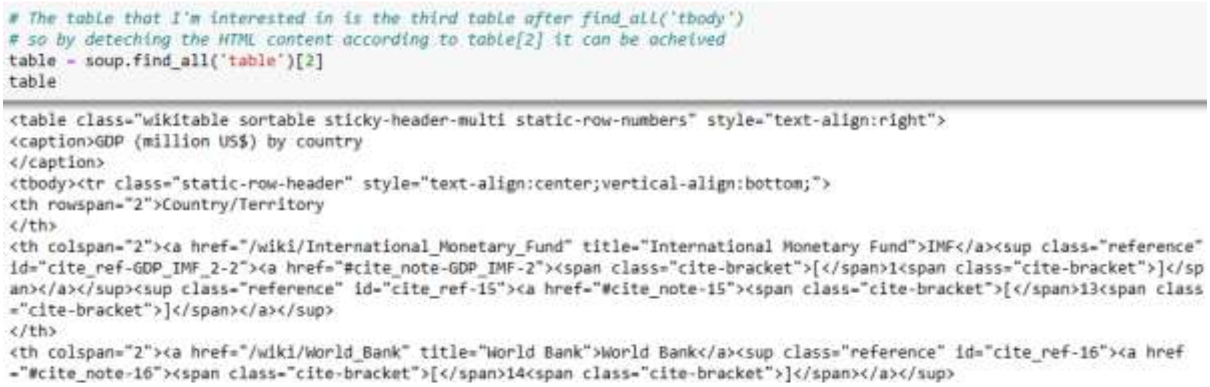
Here I am sending a request to the website to extract the dataset I am interested in I have received a response of 200 to ensure that the retrieval was successful.



```
# Parsing the HTML content
soup = BeautifulSoup(respond.text,'html.parser')
soup
```

```
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vecto
r-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feat
ure-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-limited-width-content-enabled vector-f
eature-custom-font-size-clientpref-1 vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-the
me-clientpref-day vector-toc-available" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of countries by GDP (nominal) - Wikipedia</title>
<script>(function(){var className="client-js vector-feature-language-in-header-enabled vector-feature-language-in-main-page-h
eader-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clie
```

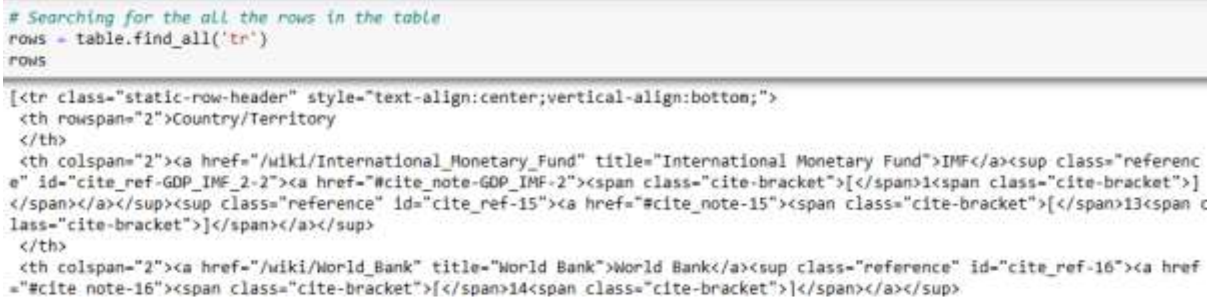
I am parsing the HTML, which is the process of examining and collecting the necessary data from unstructured, raw data that has been scraped from websites. The extracted data has been assigned under the variable 'soup'.



```
# The table that I'm interested in is the third table after find_all('tbody')
# so by deteching the HTML content according to table[2] it can be acheived
table = soup.find_all('table')[2]
table
```

```
<table class="wikitable sortable sticky-header-multi static-row-numbers" style="text-align:right">
<caption>GDP (million US$) by country
</caption>
<tbody><tr class="static-row-header" style="text-align:center;vertical-align:bottom;">
<th rowspan="2">Country/Territory
</th>
<th colspan="2"><a href="/wiki/International_Monetary_Fund" title="International Monetary Fund">IMF</a><sup class="reference"
id="cite_ref-GDP_IMF_2-2"><a href="#cite_note-GDP_IMF-2"><span class="cite-bracket">[</span>1<span class="cite-bracket">]</sp
an></a></sup><sup class="reference" id="cite_ref-15"><a href="#cite_note-15"><span class="cite-bracket">[</span>13<span class
="cite-bracket">]</span></a></sup>
</th>
<th colspan="2"><a href="/wiki/World_Bank" title="World Bank">World Bank</a><sup class="reference" id="cite_ref-16"><a href
="#cite_note-16"><span class="cite-bracket">[</span>14<span class="cite-bracket">]</span></a></sup>
```

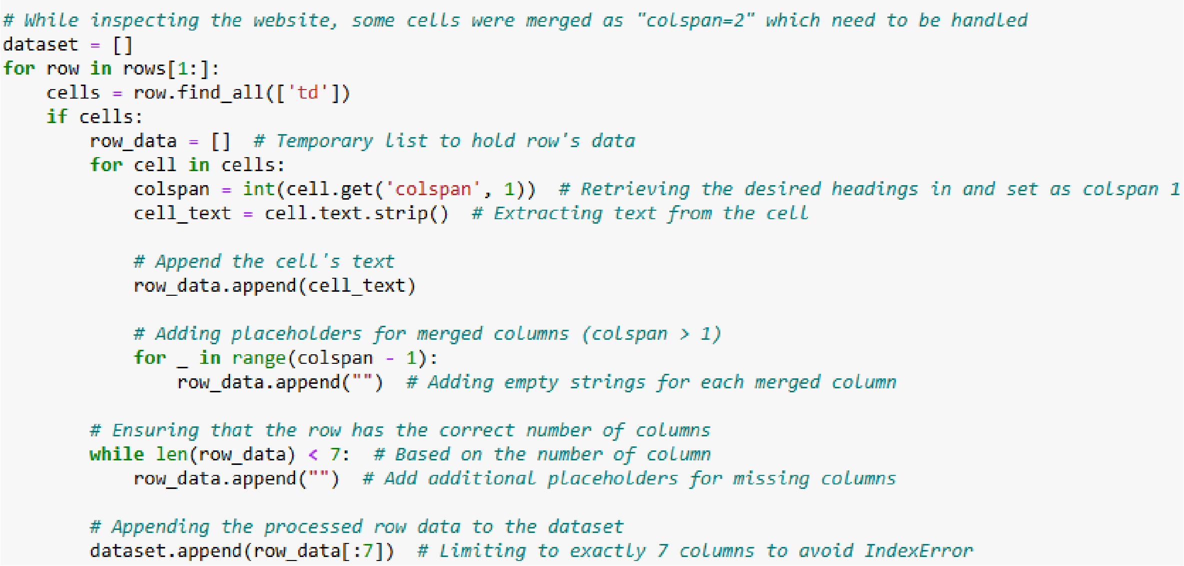
Here, I'm creating a variable named "table" to search for the first "tbody" that is mentioned in the soup data. The result has been shown.



```
# Searching for the all the rows in the table
rows = table.find_all('tr')
rows
```

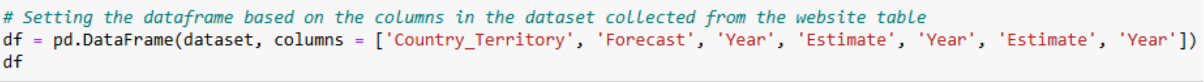
```
[<tr class="static-row-header" style="text-align:center;vertical-align:bottom;">
 <th rowspan="2">Country/Territory
 </th>
 <th colspan="2"><a href="/wiki/International_Monetary_Fund" title="International Monetary Fund">IMF</a><sup class="referenc
e" id="cite_ref-GDP_IMF_2-2"><a href="#cite_note-GDP_IMF-2"><span class="cite-bracket">[</span>1<span class="cite-bracket">]
</span></a></sup><sup class="reference" id="cite_ref-15"><a href="#cite_note-15"><span class="cite-bracket">[</span>13<span c
lass="cite-bracket">]</span></a></sup>
 </th>
 <th colspan="2"><a href="/wiki/World_Bank" title="World Bank">World Bank</a><sup class="reference" id="cite_ref-16"><a href
="#cite note-16"><span class="cite-bracket">[</span>14<span class="cite-bracket">]</span></a></sup>
```

This line of code was applied to extract the data from the HTML and assign it in the variable name rows to search for all the rows existent in a table. The result has been shown.

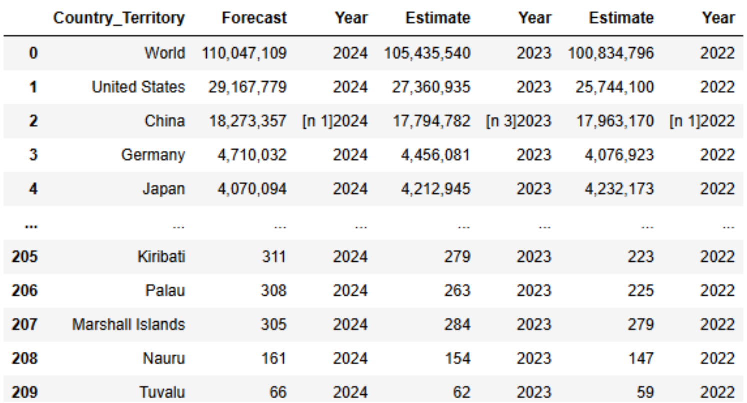


```
# While inspecting the website, some cells were merged as "colspan=2" which need to be handled
dataset = []
for row in rows[1:]:
    cells = row.find_all(['td'])
    if cells:
        row_data = []  # Temporary list to hold row's data
        for cell in cells:
            colspan = int(cell.get('colspan', 1))  # Retrieving the desired headings in and set as colspan 1
            cell_text = cell.text.strip()  # Extracting text from the cell

            # Append the cell's text
            row_data.append(cell_text)

            # Adding placeholders for merged columns (colspan > 1)
            for _ in range(colspan - 1):
                row_data.append("")  # Adding empty strings for each merged column

        # Ensuring that the row has the correct number of columns
        while len(row_data) < 7:  # Based on the number of column
            row_data.append("")  # Add additional placeholders for missing columns

        # Appending the processed row data to the dataset
        dataset.append(row_data[:7])  # Limiting to exactly 7 columns to avoid IndexError
```

This block of code is used to import the dataset into a list, search for columns and rows, extract and clean the data. The column headers had merged column names, so I needed to extract only the headers I was interested in. I have created a temporary list set as the variable ‘row data’ and set a for loop for cell in cells which is table data, to retrieve the desired headings and set them as ‘colspan 1’. The next line is used to extract the text from the cell and then append it to the row data. Another for loop has been set in range to add placeholders for the merged columns and adding empty strings for each merged column to fill in the gaps. A while loop has been set to check that the length of the row data has the correct number of columns which has been set to 7, while adding placeholders for any missing columns. Finally, the prepared data has been added and combined to the dataset list.



```
# Setting the dataframe based on the columns in the dataset collected from the website table
df = pd.DataFrame(dataset, columns = ['Country_Territory', 'Forecast', 'Year', 'Estimate', 'Year', 'Estimate', 'Year'])
df
```



	Country_Territory	Forecast	Year	Estimate	Year	Estimate	Year
0	World	110,047,109	2024	105,435,540	2023	100,834,796	2022
1	United States	29,167,779	2024	27,360,935	2023	25,744,100	2022
2	China	18,273,357	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
3	Germany	4,710,032	2024	4,456,081	2023	4,076,923	2022
4	Japan	4,070,094	2024	4,212,945	2023	4,232,173	2022
...	...	...	...	...	...	...	...
205	Kiribati	311	2024	279	2023	223	2022
206	Palau	308	2024	263	2023	225	2022
207	Marshall Islands	305	2024	284	2023	279	2022
208	Nauru	161	2024	154	2023	147	2022
209	Tuvalu	66	2024	62	2023	59	2022

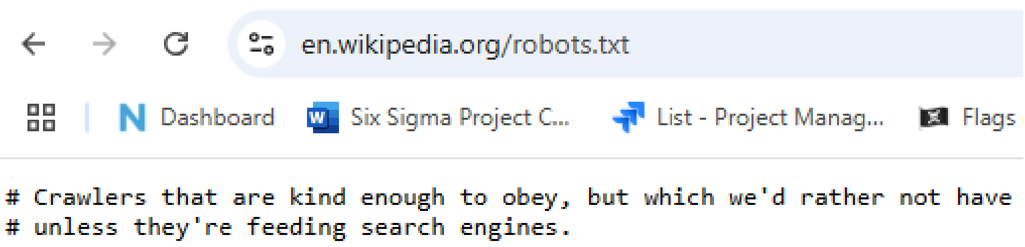
Here I am creating a DataFrame based on the list of columns in the dataset and displaying the output of the data in tabulation form.

Ethical Standards and Data Privacy Regulations

All the websites in which I have web scraped, is from Wikipedia which is a free internet-based encyclopaedia. My scraping process adheres to the ethical standards and complies with data privacy regulations by checking robots.txt, reading their Terms of Use, and checking for a websites API.

Robots.txt

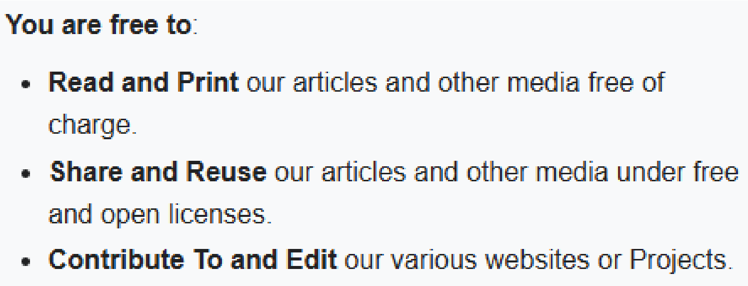
I have inserted 'robots.txt' into the URL and Wikipedia's robots.txt file states that it allows web crawlers to access its pages, which means that scraping is technically permitted. This is not enough to justify my reason for using it so I have found a more reliable evidence of web scraping ethically below.



```
# Crawlers that are kind enough to obey, but which we'd rather not have
# unless they're feeding search engines.
```

Wikipedia Terms of Use

In Wikipedia's Terms of Use document, it states we are free to read, print, share, reuse, contribute, and edit their websites. This ensures that I am allowed to web scrap from their websites, while adhering to their Terms of Use.



You are free to:

- **Read and Print** our articles and other media free of charge.
- **Share and Reuse** our articles and other media under free and open licenses.
- **Contribute To and Edit** our various websites or Projects.

Wikipedia's API

Additionally, Wikipedia provides a Media Wiki API, which is intended to access and retrieve content in an organised manner that is more effective and is less inclined to break any rules. This is a slightly safer and more dependable option for web scraping.

Complying with the data privacy regulations is important as to avoid legal consequences, protect user privacy, and maintain ethical standards which are important to be recognised as reliable and trustworthy scrapper.

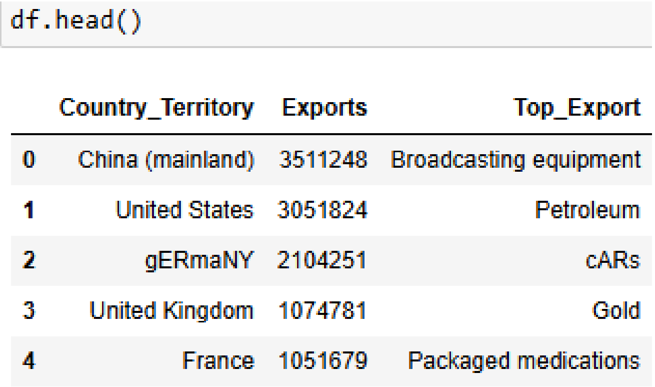
PART B - Data Preparation and Cleansing

Task 3 & 4: Data Preparation, Cleansing, and Documentation

Website 1: List of Countries by Exports

Typos

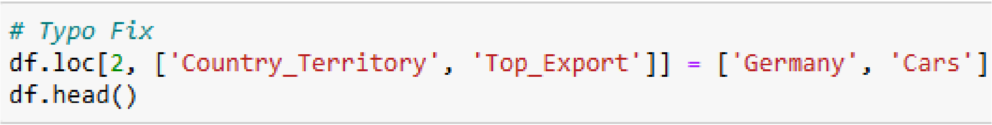
Here I have displayed the first few rows. There is a row that seems to have a typo in both the 'Country Territory' and 'Top Export' columns.



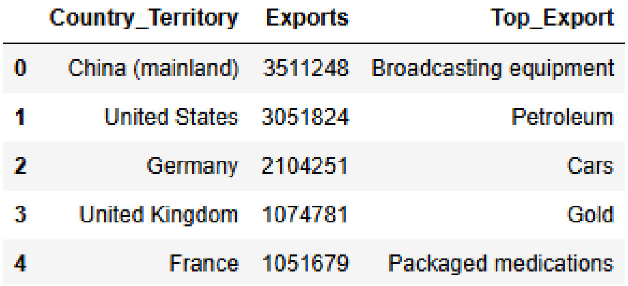
```
df.head()
```

	Country_Territory	Exports	Top_Export
0	China (mainland)	3511248	Broadcasting equipment
1	United States	3051824	Petroleum
2	gERmaNY	2104251	cARs
3	United Kingdom	1074781	Gold
4	France	1051679	Packaged medications

Here I have applied the 'loc' method to locate the specific row through the index and included the columns that contain the typos and then re-entered the correct spelling, displaying the results to view the change.



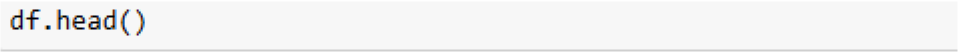
```
# Typo Fix
df.loc[2, ['Country_Territory', 'Top_Export']] = ['Germany', 'Cars']
df.head()
```



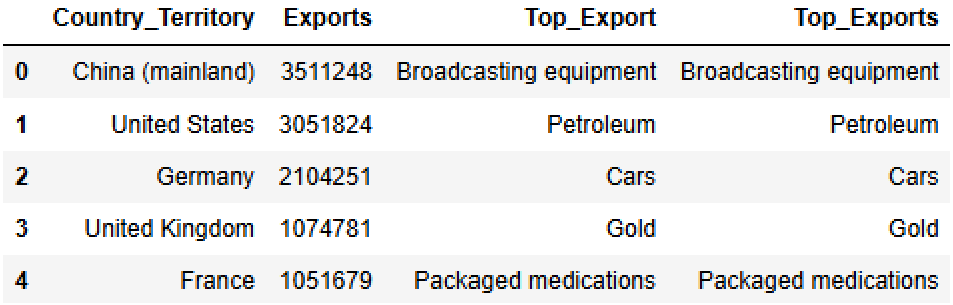
	Country_Territory	Exports	Top_Export
0	China (mainland)	3511248	Broadcasting equipment
1	United States	3051824	Petroleum
2	Germany	2104251	Cars
3	United Kingdom	1074781	Gold
4	France	1051679	Packaged medications

Column Duplicates

Here I have displayed the first few rows. There seems to be a duplicate column spotted in the dataset.

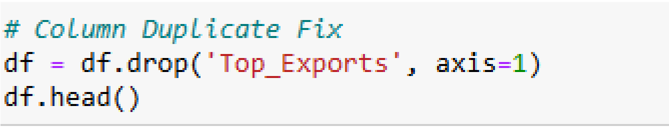


```
df.head()
```

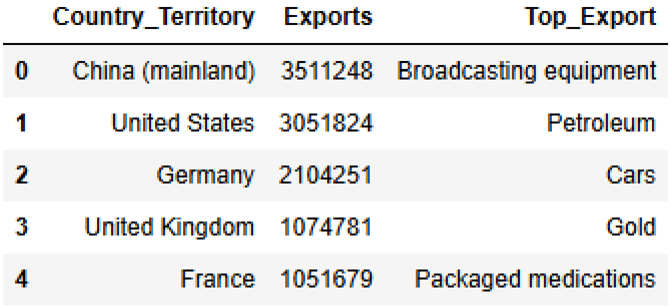


	Country_Territory	Exports	Top_Export	Top_Exports
0	China (mainland)	3511248	Broadcasting equipment	Broadcasting equipment
1	United States	3051824	Petroleum	Petroleum
2	Germany	2104251	Cars	Cars
3	United Kingdom	1074781	Gold	Gold
4	France	1051679	Packaged medications	Packaged medications

Here I am using the ‘drop’ method to drop the duplicate column mentioned above, specifying the exact column name and axis. The result has been shown.



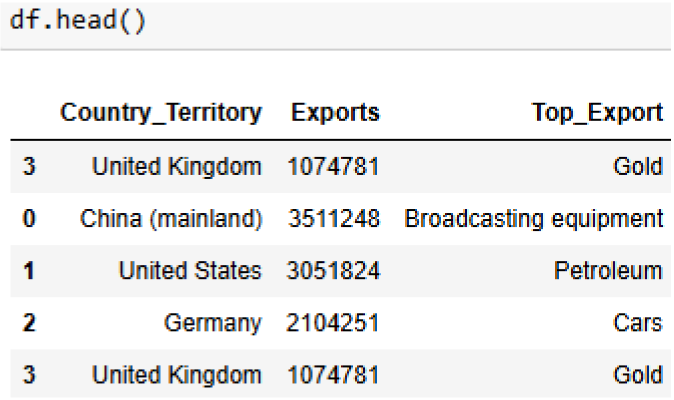
```
# Column Duplicate Fix
df = df.drop('Top_Exports', axis=1)
df.head()
```



	Country_Territory	Exports	Top_Export
0	China (mainland)	3511248	Broadcasting equipment
1	United States	3051824	Petroleum
2	Germany	2104251	Cars
3	United Kingdom	1074781	Gold
4	France	1051679	Packaged medications

Row Duplicates

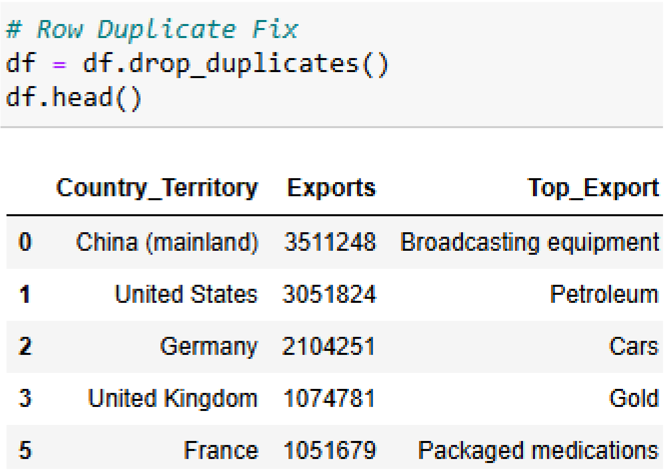
The first few rows have been displayed. There seems to be a duplicate row spotted in the dataset.



```
df.head()
```

	Country_Territory	Exports	Top_Export
3	United Kingdom	1074781	Gold
0	China (mainland)	3511248	Broadcasting equipment
1	United States	3051824	Petroleum
2	Germany	2104251	Cars
3	United Kingdom	1074781	Gold

Here I am using the ‘drop duplicates’ method to drop any duplicates found in the dataset and then displaying the results to ensure that it has been successful.

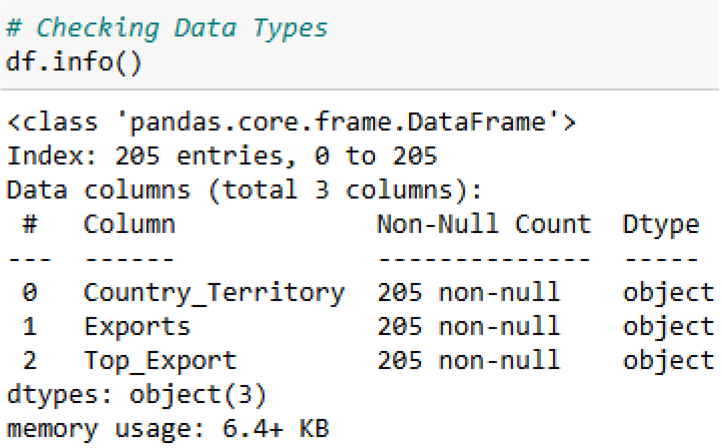


```
# Row Duplicate Fix
df = df.drop_duplicates()
df.head()
```

	Country_Territory	Exports	Top_Export
0	China (mainland)	3511248	Broadcasting equipment
1	United States	3051824	Petroleum
2	Germany	2104251	Cars
3	United Kingdom	1074781	Gold
5	France	1051679	Packaged medications

Incorrect Data Type

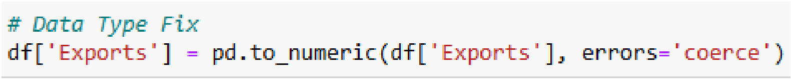
Here I am using the ‘info’ method to check the data type of each column. The ‘Exports’ column’s data type is set as object and should be changed to correct data type of integer.



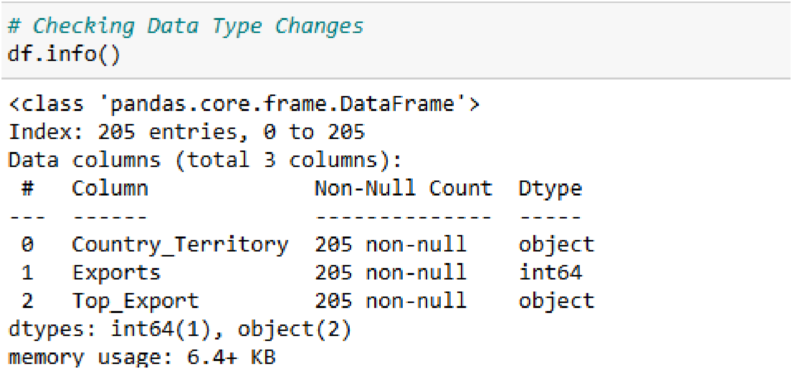
```
# Checking Data Types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 205 entries, 0 to 205
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Country_Territory  205 non-null    object
 1   Exports            205 non-null    object
 2   Top_Export         205 non-null    object
dtypes: object(3)
memory usage: 6.4+ KB
```

I have used the ‘to numeric’ method to change the specified column to a numerical data type and used the ‘errors=’coerce’’ line to handle invalid or non-convertible values during the type conversion.



```
# Data Type Fix
df['Exports'] = pd.to_numeric(df['Exports'], errors='coerce')
```

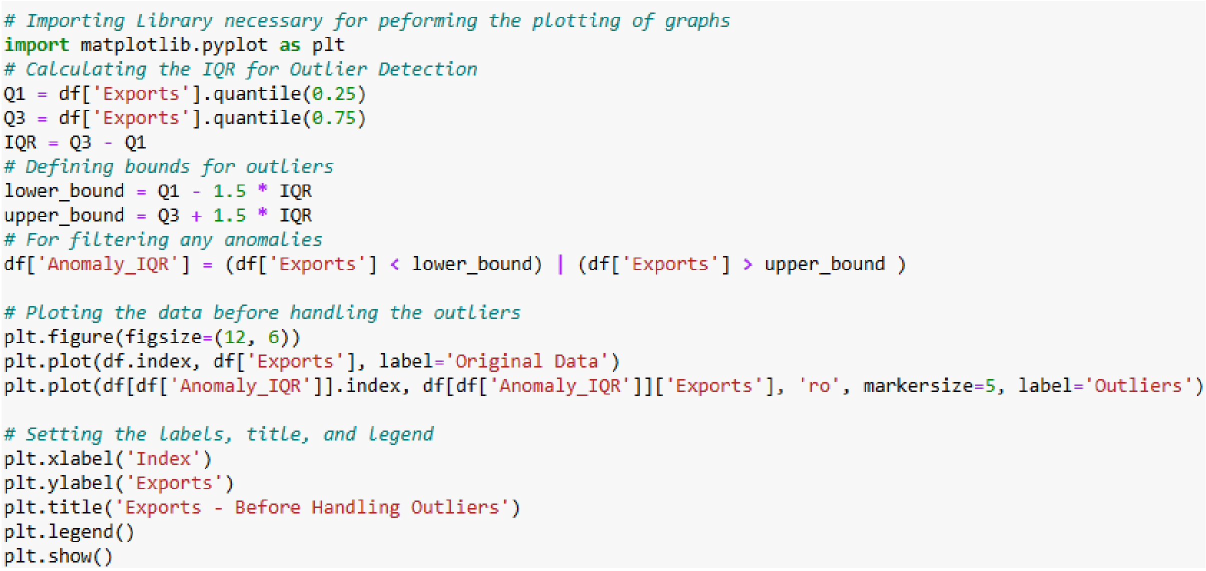


```
# Checking Data Type Changes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 205 entries, 0 to 205
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Country_Territory  205 non-null    object
 1   Exports            205 non-null    int64 
 2   Top_Export         205 non-null    object
dtypes: int64(1), object(2)
memory usage: 6.4+ KB
```

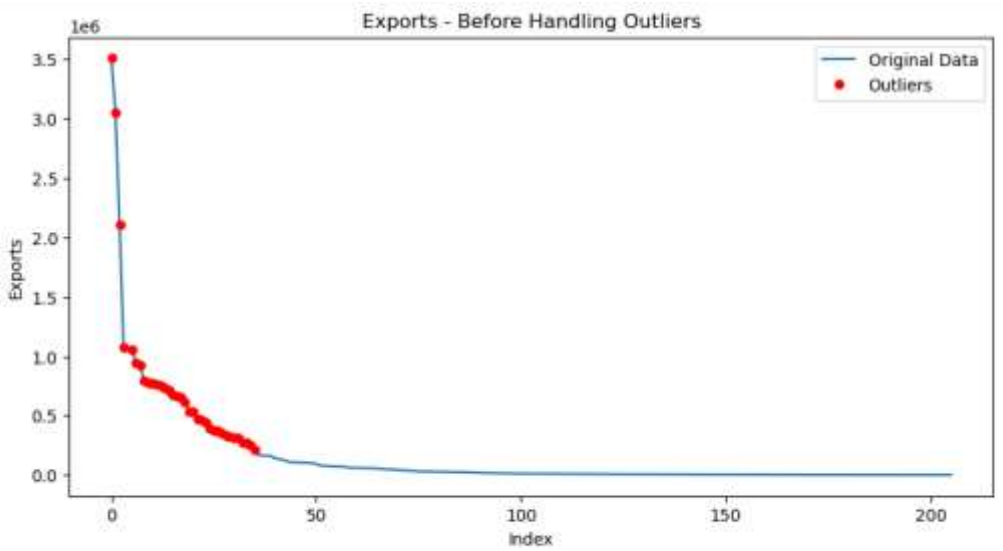
Outliers

Here I have imported the matplotlib library for plotting graphs. I have calculated the IQR for outlier detection using the 'quantile' method which calculates the values at specified quantiles of 0.25 and 0.75. I have defined the bounds for the lower and upper outliers potentially caught in the process. I have then used the 'filtering' method to define the 'Anomaly IQR' by filtering the Exports lesser than the lower bound or (represented by the |) the Exports greater than the upper bound. I have then plotted the data before handling the outliers by using various plotting methods and plotting the labels, title, and legend.

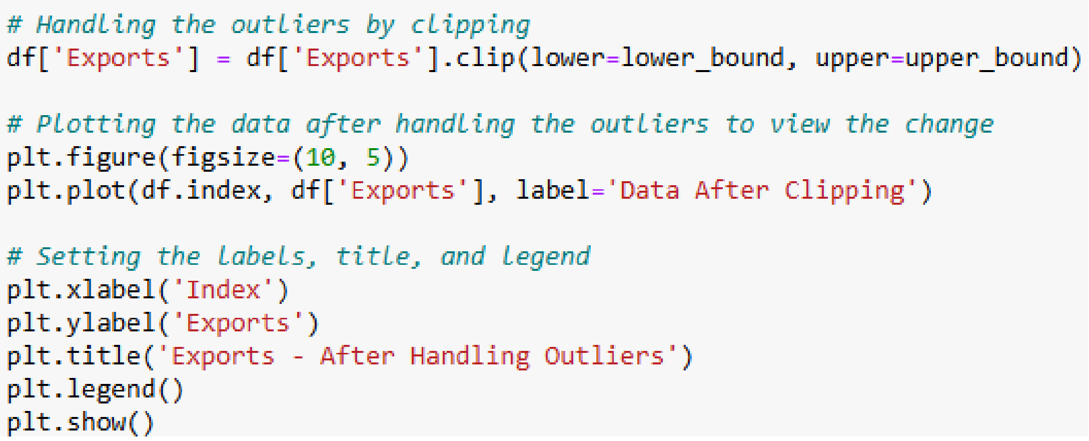


```
# Importing Library necessary for peforming the plotting of graphs
import matplotlib.pyplot as plt
# Calculating the IQR for Outlier Detection
Q1 = df['Exports'].quantile(0.25)
Q3 = df['Exports'].quantile(0.75)
IQR = Q3 - Q1
# Defining bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# For filtering any anomalies
df['Anomaly_IQR'] = (df['Exports'] < lower_bound) | (df['Exports'] > upper_bound )

# Ploting the data before handling the outliers
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Exports'], label='Original Data')
plt.plot(df[df['Anomaly_IQR']].index, df[df['Anomaly_IQR']]['Exports'], 'ro', markersize=5, label='Outliers')

# Setting the labels, title, and legend
plt.xlabel('Index')
plt.ylabel('Exports')
plt.title('Exports - Before Handling Outliers')
plt.legend()
plt.show()
```

Here is the output of the outliers detected before handling in the form of a graph. There are a handful of outliers that have been detected.

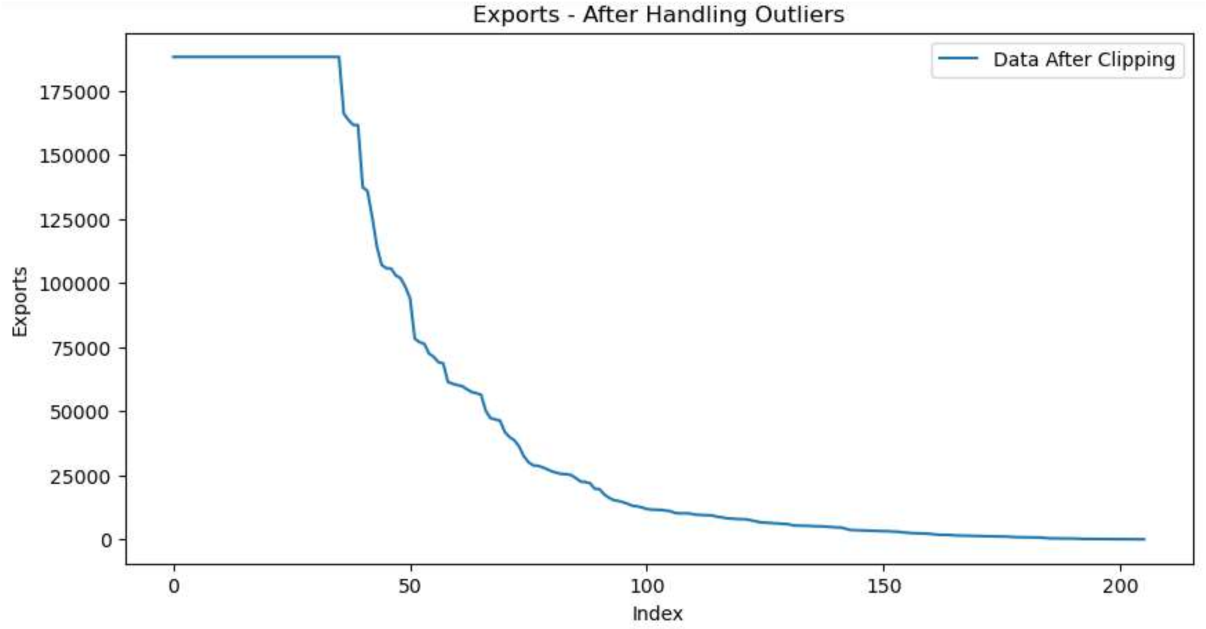


Here I am using the ‘clip’ method to cut any outliers found outside of the lower and upper bounds set previously. I have then plotted the data and set the labels, title, and legend.



```
# Handling the outliers by clipping
df['Exports'] = df['Exports'].clip(lower=lower_bound, upper=upper_bound)

# Plotting the data after handling the outliers to view the change
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Exports'], label='Data After Clipping')

# Setting the labels, title, and legend
plt.xlabel('Index')
plt.ylabel('Exports')
plt.title('Exports - After Handling Outliers')
plt.legend()
plt.show()
```

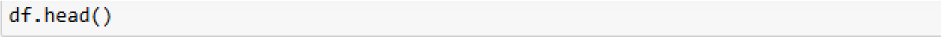
Here I have shown the results in a graph that shows the data after clipping the outliers. The dataset is now clean.



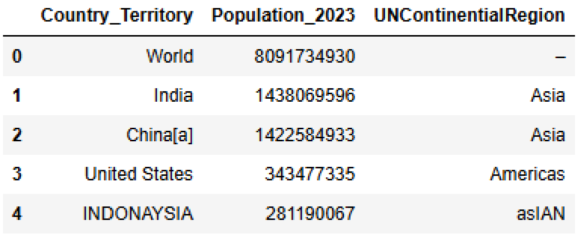
Website 2: List of Countries and Dependencies by Population

Typos

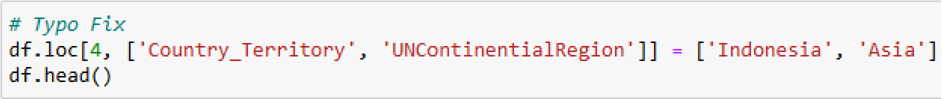
Here I have displayed the first few rows and noticed a row with typos found in two columns. I have then applied the 'loc' method to locate the specific row that contains the typos and then re-enter the correct spelling, displaying the results to view the change.



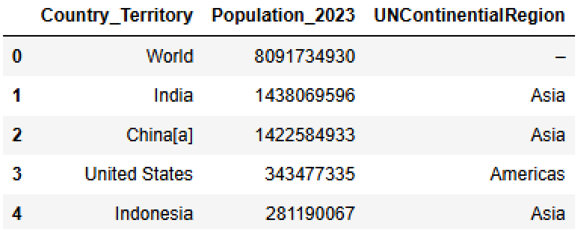
```
df.head()
```



	Country_Territory	Population_2023	UNContinentialRegion
0	World	8091734930	–
1	India	1438069596	Asia
2	China[a]	1422584933	Asia
3	United States	343477335	Americas
4	INDONAYSIA	281190067	asIAN



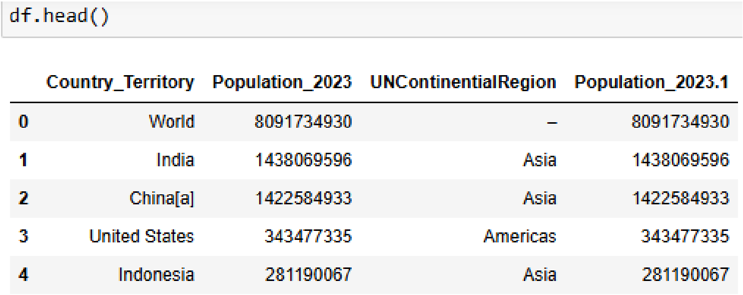
```
# Typo Fix
df.loc[4, ['Country_Territory', 'UNContinentialRegion']] = ['Indonesia', 'Asia']
df.head()
```



	Country_Territory	Population_2023	UNContinentialRegion
0	World	8091734930	–
1	India	1438069596	Asia
2	China[a]	1422584933	Asia
3	United States	343477335	Americas
4	Indonesia	281190067	Asia

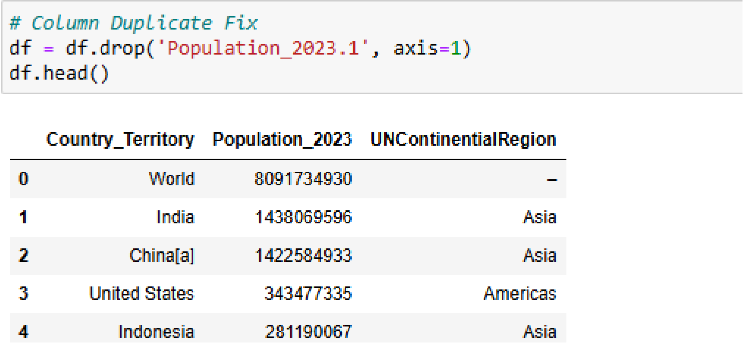
Column Duplicates

Here I have displayed the first few rows and noticed that there is a duplicated column that needs to be removed. I have then used the 'drop' method to drop the duplicate column, specifying the exact column name and axis. The result has been shown.



```
df.head()
```

	Country_Territory	Population_2023	UNContinentialRegion	Population_2023.1
0	World	8091734930	–	8091734930
1	India	1438069596	Asia	1438069596
2	China[a]	1422584933	Asia	1422584933
3	United States	343477335	Americas	343477335
4	Indonesia	281190067	Asia	281190067

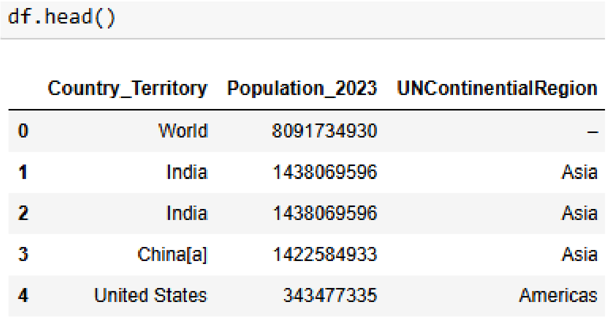


```
# Column Duplicate Fix
df = df.drop('Population_2023.1', axis=1)
df.head()
```

	Country_Territory	Population_2023	UNContinentialRegion
0	World	8091734930	–
1	India	1438069596	Asia
2	China[a]	1422584933	Asia
3	United States	343477335	Americas
4	Indonesia	281190067	Asia

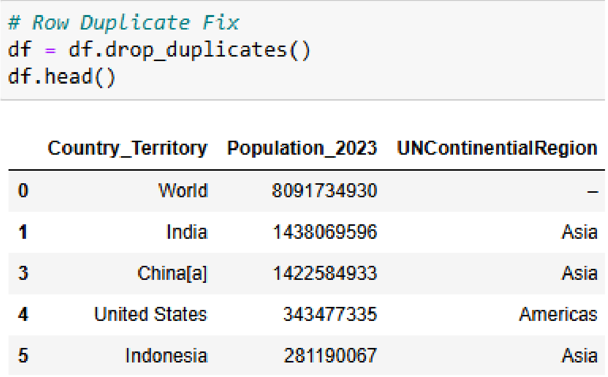
Row Duplicates

The first few rows have been displayed, and a duplicate row has been found in the dataset. I am using the ‘drop duplicates’ method to drop any duplicates found in the dataset and then displaying the results to ensure that it has been successful.



```
df.head()
```

	Country_Territory	Population_2023	UNContinentialRegion
0	World	8091734930	–
1	India	1438069596	Asia
2	India	1438069596	Asia
3	China[a]	1422584933	Asia
4	United States	343477335	Americas

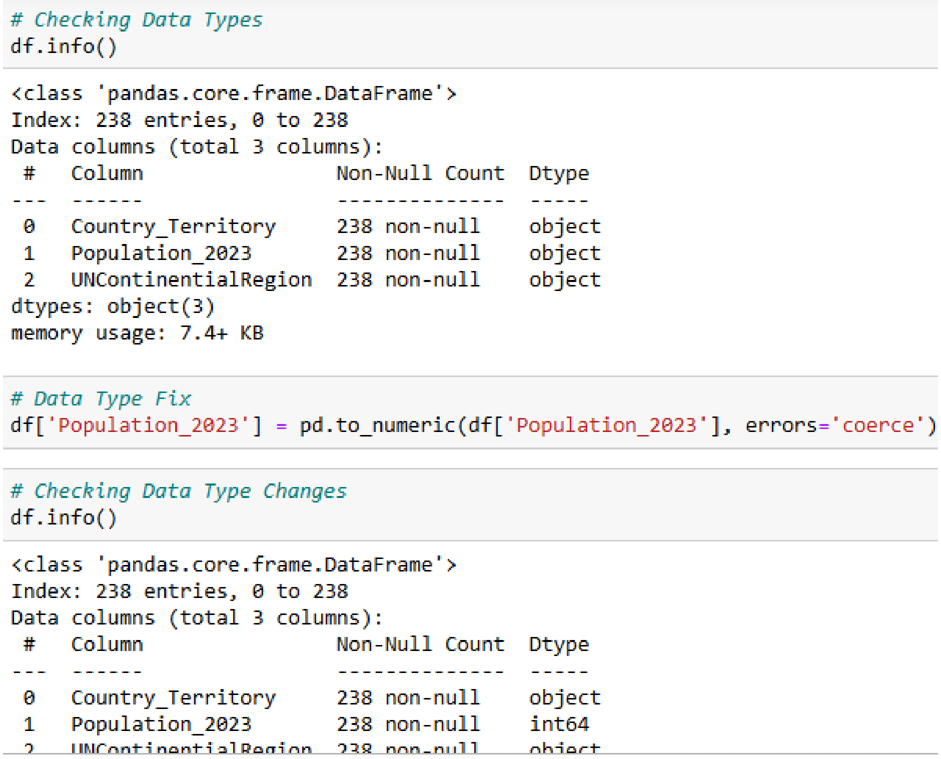


```
# Row Duplicate Fix
df = df.drop_duplicates()
df.head()
```

	Country_Territory	Population_2023	UNContinentialRegion
0	World	8091734930	–
1	India	1438069596	Asia
3	China[a]	1422584933	Asia
4	United States	343477335	Americas
5	Indonesia	281190067	Asia

Incorrect Data Type

I am using the ‘info’ method to check the data type of each column and have noticed that the ‘Population_2023’ column’s data type is set as object and should be changed to correct data type of integer. I have used the ‘to numeric’ method to change the specified column to a numerical data type and used the ‘errors=’coerce’’ line to handle invalid or non-convertible values during the type conversion. The results have been shown to acknowledge the changes.



```
# Checking Data Types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 238 entries, 0 to 238
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Country_Territory     238 non-null    object
 1   Population_2023       238 non-null    object
 2   UNContinentialRegion  238 non-null    object
dtypes: object(3)
memory usage: 7.4+ KB
```

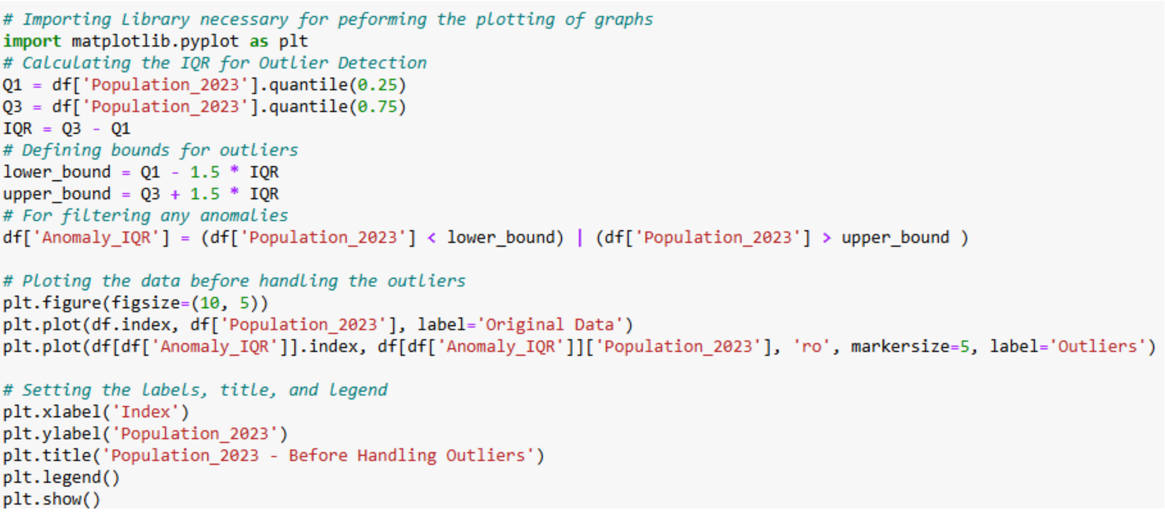
```
# Data Type Fix
df['Population_2023'] = pd.to_numeric(df['Population_2023'], errors='coerce')
```

```
# Checking Data Type Changes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 238 entries, 0 to 238
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Country_Territory     238 non-null    object
 1   Population_2023       238 non-null    int64 
 2   UNContinentialRegion  238 non-null    object
```

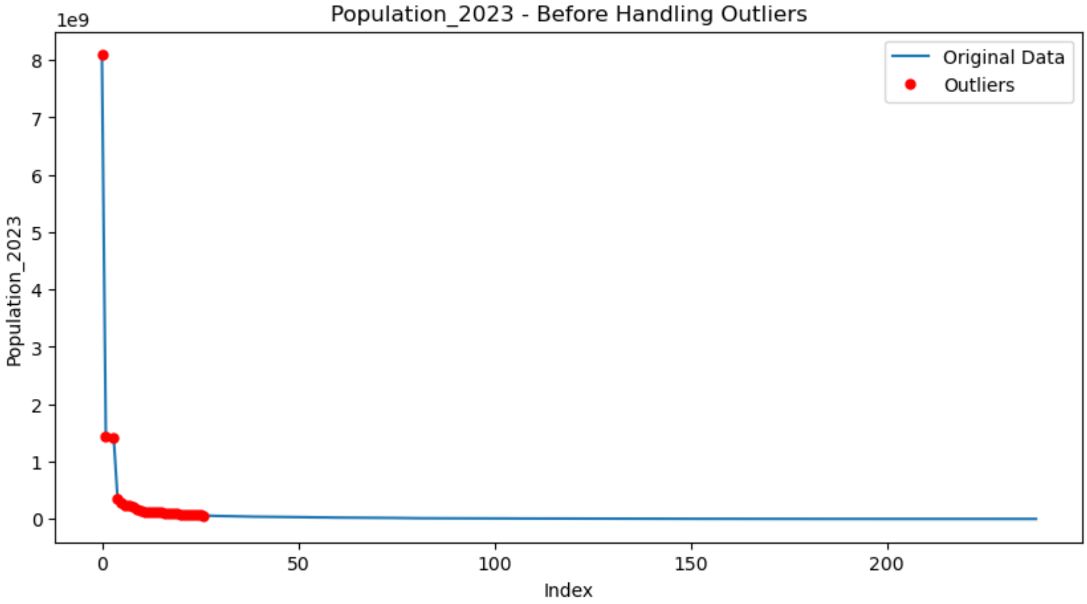
Outliers

I have imported the matplotlib library for plotting graphs. I have calculated the IQR for outlier detection and defined the bounds for the lower and upper outliers potentially caught in the process. I have then used the ‘filtering’ method to define the ‘Anomaly IQR’ and then plotted the data before handling the outliers by using various plotting methods and plotting the labels, title, and legend.

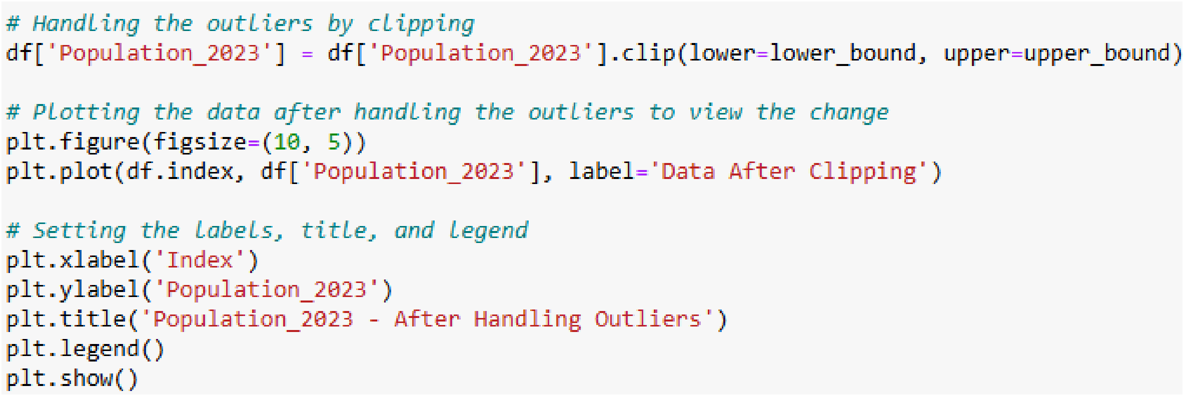


```
# Importing Library necessary for peforming the plotting of graphs
import matplotlib.pyplot as plt
# Calculating the IQR for Outlier Detection
Q1 = df['Population_2023'].quantile(0.25)
Q3 = df['Population_2023'].quantile(0.75)
IQR = Q3 - Q1
# Defining bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# For filtering any anomalies
df['Anomaly_IQR'] = (df['Population_2023'] < lower_bound) | (df['Population_2023'] > upper_bound )

# Ploting the data before handling the outliers
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Population_2023'], label='Original Data')
plt.plot(df[df['Anomaly_IQR']].index, df[df['Anomaly_IQR']]['Population_2023'], 'ro', markersize=5, label='Outliers')

# Setting the labels, title, and legend
plt.xlabel('Index')
plt.ylabel('Population_2023')
plt.title('Population_2023 - Before Handling Outliers')
plt.legend()
plt.show()
```

Here is the output of the outliers detected before handling in the form of a graph. There are a handful of outliers that have been detected.

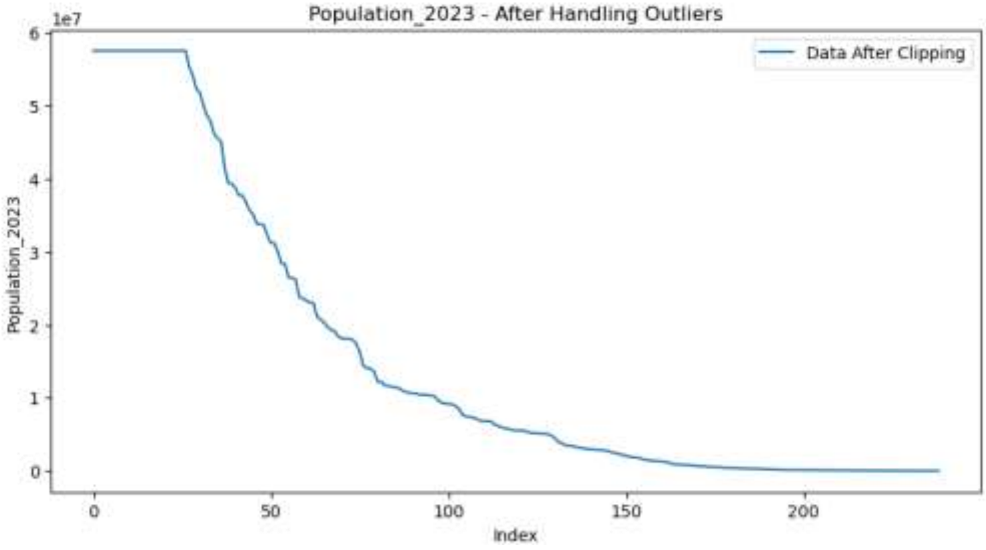


Here I am using the ‘clip’ method to cut any outliers found outside of the lower and upper bounds set previously. I have then plotted the data and set the labels, title, and legend.



```
# Handling the outliers by clipping
df['Population_2023'] = df['Population_2023'].clip(lower=lower_bound, upper=upper_bound)

# Plotting the data after handling the outliers to view the change
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Population_2023'], label='Data After Clipping')

# Setting the labels, title, and legend
plt.xlabel('Index')
plt.ylabel('Population_2023')
plt.title('Population_2023 - After Handling Outliers')
plt.legend()
plt.show()
```

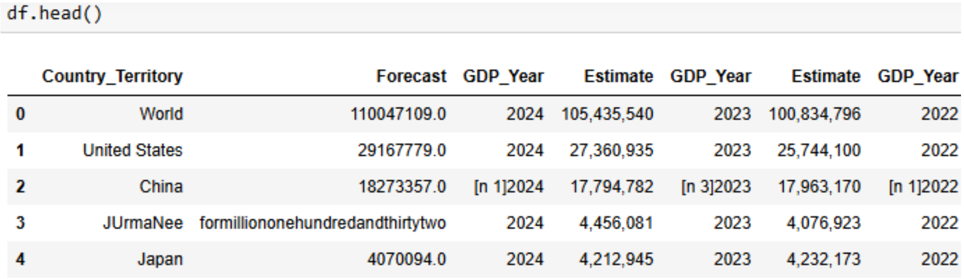
Here I have shown the results in a graph that shows the data after clipping the outliers. The dataset is now clean.



Website 3: List of Countries by GDP (nominal)

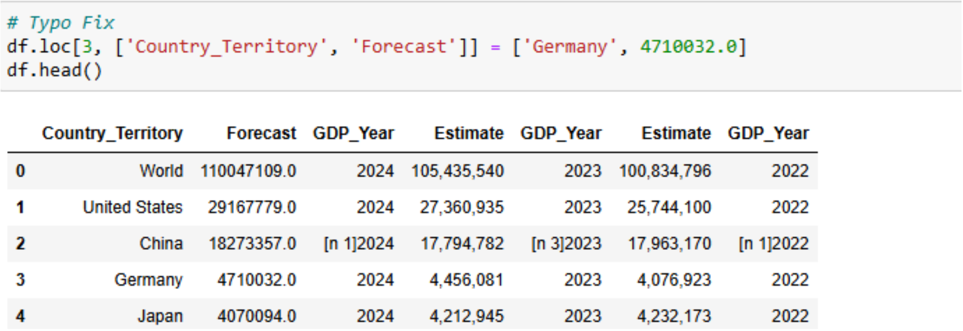
Typos

Here I have displayed the first few rows and noticed a row with typos found in two columns. I have then applied the 'loc' method to locate the specific row that contains the typos and then re-enter the correct spelling, displaying the results to view the change.



```
df.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year	Estimate	GDP_Year
0	World	110047109.0	2024	105,435,540	2023	100,834,796	2022
1	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022
2	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
3	JUrmaNee	formilliononehundredandthirtytwo	2024	4,456,081	2023	4,076,923	2022
4	Japan	4070094.0	2024	4,212,945	2023	4,232,173	2022

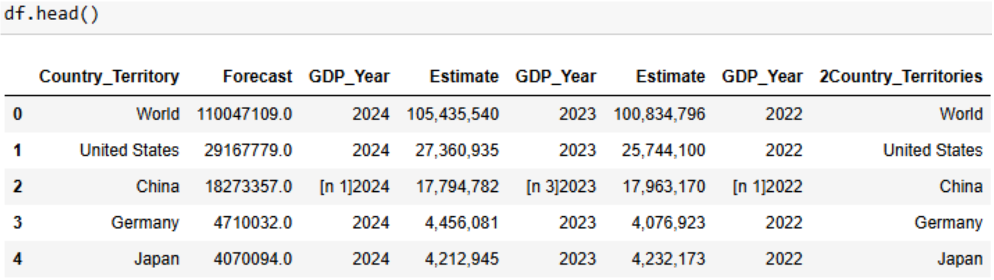


```
# Typo Fix
df.loc[3, ['Country_Territory', 'Forecast']] = ['Germany', 4710032.0]
df.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year	Estimate	GDP_Year
0	World	110047109.0	2024	105,435,540	2023	100,834,796	2022
1	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022
2	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
3	Germany	4710032.0	2024	4,456,081	2023	4,076,923	2022
4	Japan	4070094.0	2024	4,212,945	2023	4,232,173	2022

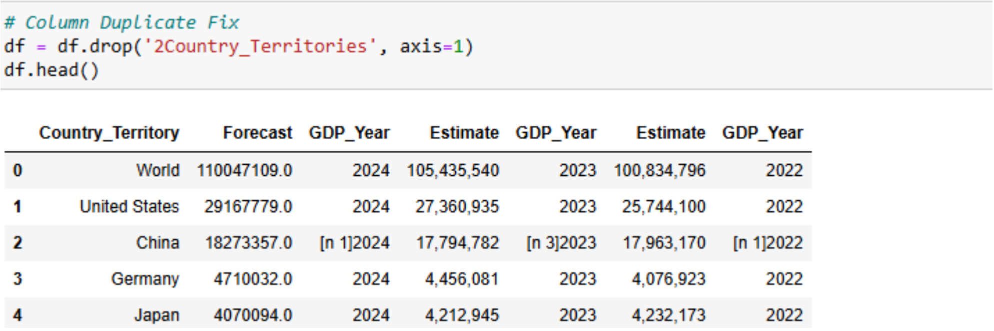
Column Duplicates

Here I have displayed the first few rows and noticed that there is a duplicated column that needs to be removed. I have then used the 'drop' method to drop the duplicate column, specifying the exact column name and axis. The result has been shown.



```
df.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year	Estimate	GDP_Year	2Country_Territories
0	World	110047109.0	2024	105,435,540	2023	100,834,796	2022	World
1	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022	United States
2	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022	China
3	Germany	4710032.0	2024	4,456,081	2023	4,076,923	2022	Germany
4	Japan	4070094.0	2024	4,212,945	2023	4,232,173	2022	Japan

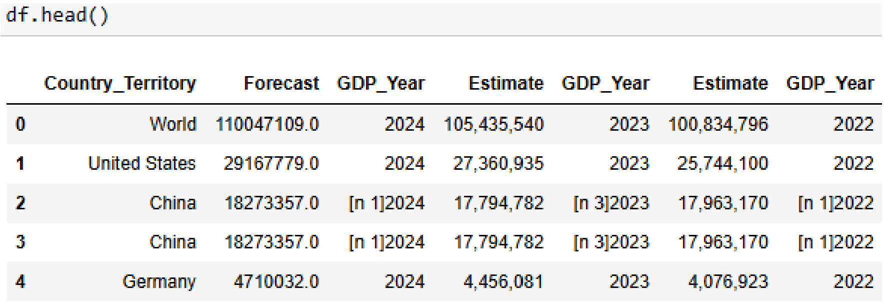


```
# Column Duplicate Fix
df = df.drop('2Country_Territories', axis=1)
df.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year	Estimate	GDP_Year
0	World	110047109.0	2024	105,435,540	2023	100,834,796	2022
1	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022
2	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
3	Germany	4710032.0	2024	4,456,081	2023	4,076,923	2022
4	Japan	4070094.0	2024	4,212,945	2023	4,232,173	2022

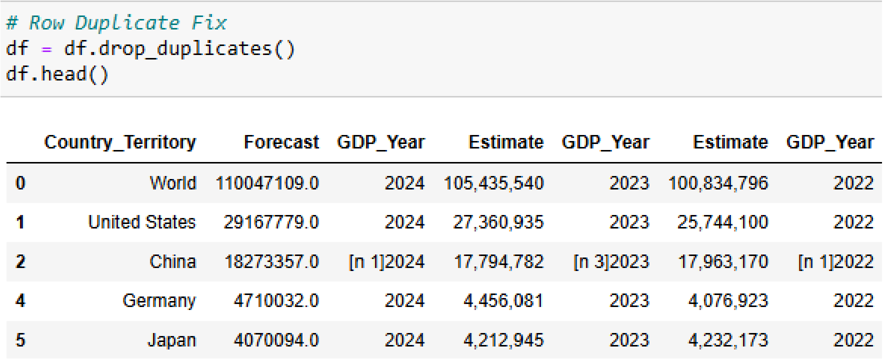
Row Duplicates

The first few rows have been displayed, and a duplicate row has been found in the dataset. I am using the ‘drop duplicates’ method to drop any duplicates found in the dataset and then displaying the results to ensure that it has been successful.



```
df.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year	Estimate	GDP_Year
0	World	110047109.0	2024	105,435,540	2023	100,834,796	2022
1	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022
2	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
3	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
4	Germany	4710032.0	2024	4,456,081	2023	4,076,923	2022

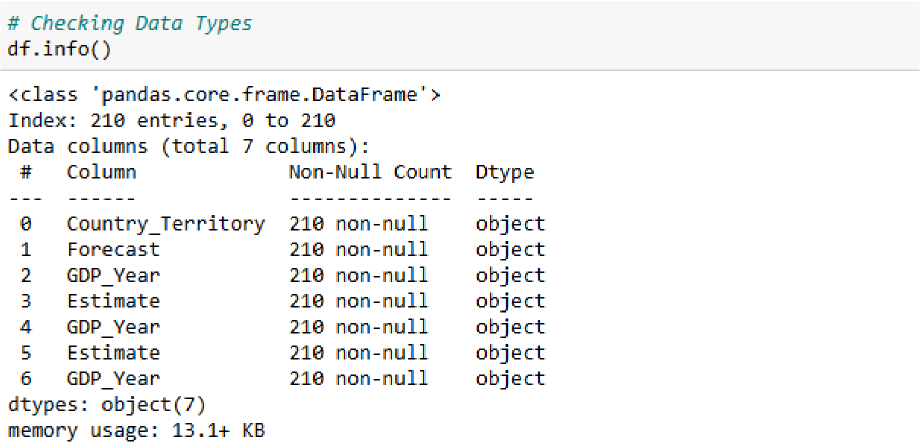


```
# Row Duplicate Fix
df = df.drop_duplicates()
df.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year	Estimate	GDP_Year
0	World	110047109.0	2024	105,435,540	2023	100,834,796	2022
1	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022
2	China	18273357.0	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
4	Germany	4710032.0	2024	4,456,081	2023	4,076,923	2022
5	Japan	4070094.0	2024	4,212,945	2023	4,232,173	2022

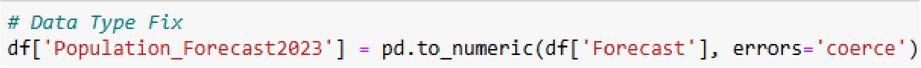
Incorrect Data Type

I am using the ‘info’ method to check the data type of each column and have noticed that the ‘Forecast’ column’s data type is set as object and should be changed to correct data type of integer. I have used the ‘to numeric’ method to change the specified column to a numerical data type and used the ‘errors=’coerce’’ line to handle invalid or non-convertible values during the type conversion.



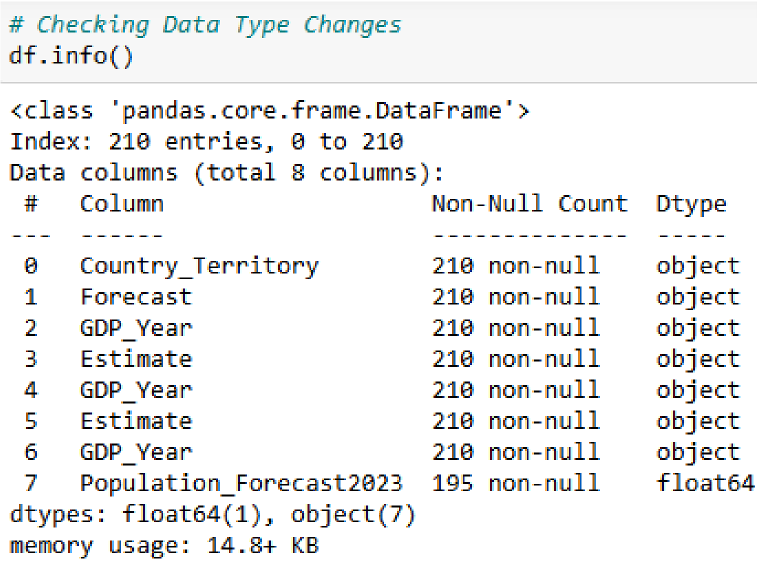
```
# Checking Data Types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 210 entries, 0 to 210
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Country_Territory  210 non-null    object
 1   Forecast           210 non-null    object
 2   GDP_Year           210 non-null    object
 3   Estimate           210 non-null    object
 4   GDP_Year           210 non-null    object
 5   Estimate           210 non-null    object
 6   GDP_Year           210 non-null    object
dtypes: object(7)
memory usage: 13.1+ KB
```



```
# Data Type Fix
df['Population_Forecast2023'] = pd.to_numeric(df['Forecast'], errors='coerce')
```

The results have been shown to acknowledge the changes.

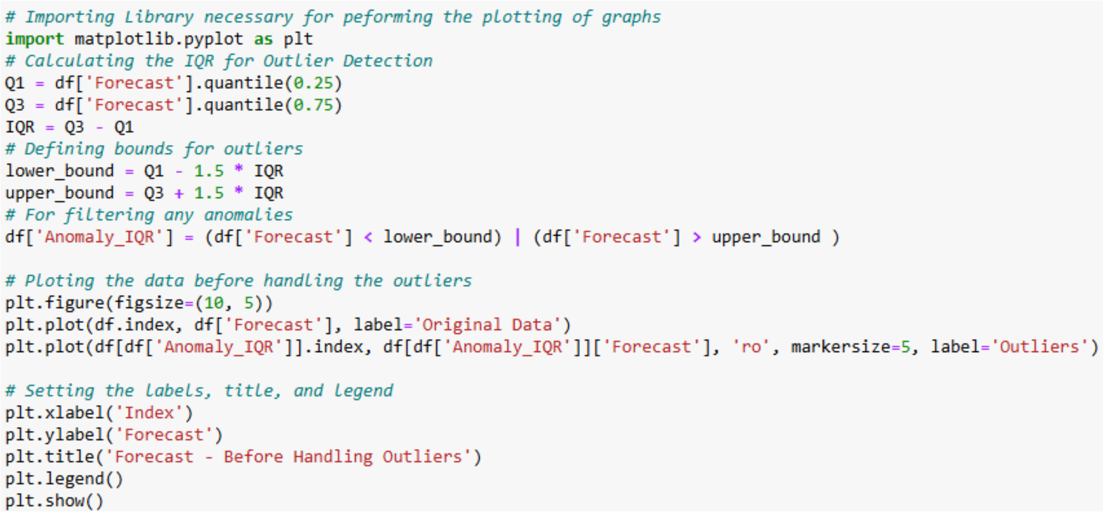


```
# Checking Data Type Changes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 210 entries, 0 to 210
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Country_Territory       210 non-null    object 
 1   Forecast                210 non-null    object 
 2   GDP_Year                210 non-null    object 
 3   Estimate                210 non-null    object 
 4   GDP_Year                210 non-null    object 
 5   Estimate                210 non-null    object 
 6   GDP_Year                210 non-null    object 
 7   Population_Forecast2023  195 non-null    float64
dtypes: float64(1), object(7)
memory usage: 14.8+ KB
```

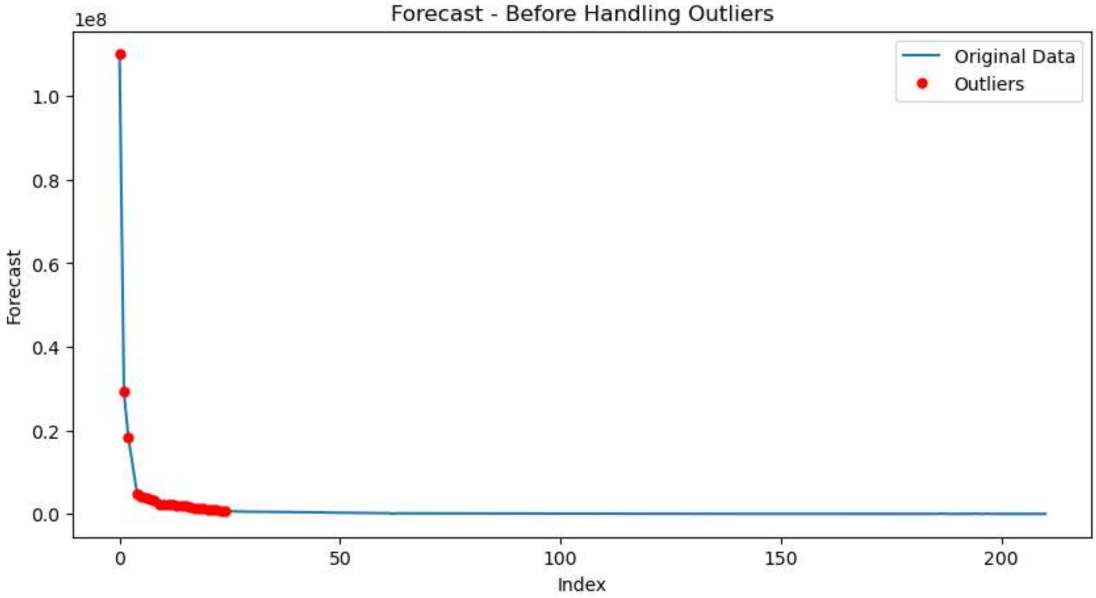
Outliers

I have imported the matplotlib library for plotting graphs. I have calculated the IQR for outlier detection and defined the bounds for the lower and upper outliers potentially caught in the process. I have then used the 'filtering' method to define the 'Anomaly IQR' and then plotted the data before handling the outliers by using various plotting methods and plotting the labels, title, and legend.

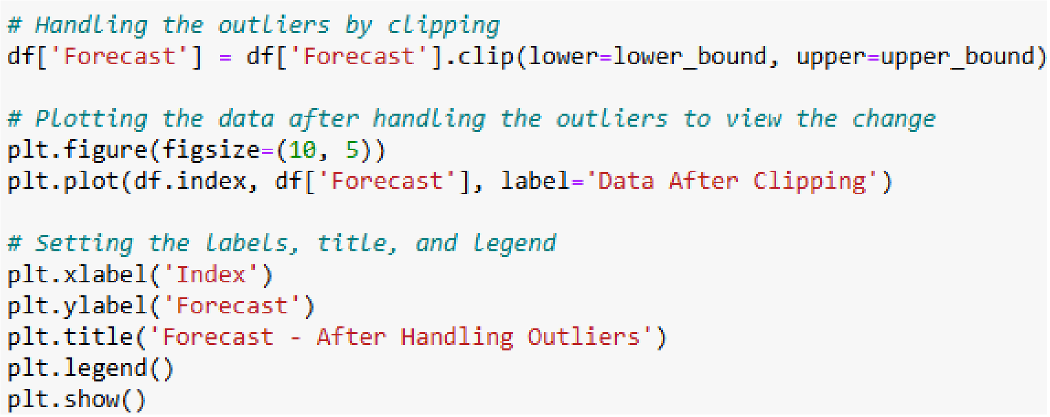


```
# Importing Library necessary for peforming the plotting of graphs
import matplotlib.pyplot as plt
# Calculating the IQR for Outlier Detection
Q1 = df['Forecast'].quantile(0.25)
Q3 = df['Forecast'].quantile(0.75)
IQR = Q3 - Q1
# Defining bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# For filtering any anomalies
df['Anomaly_IQR'] = (df['Forecast'] < lower_bound) | (df['Forecast'] > upper_bound )

# Ploting the data before handling the outliers
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Forecast'], label='Original Data')
plt.plot(df[df['Anomaly_IQR']].index, df[df['Anomaly_IQR']]['Forecast'], 'ro', markersize=5, label='Outliers')

# Setting the labels, title, and legend
plt.xlabel('Index')
plt.ylabel('Forecast')
plt.title('Forecast - Before Handling Outliers')
plt.legend()
plt.show()
```

Here is the output of the outliers detected before handling in the form of a graph. There are a handful of outliers that have been detected.

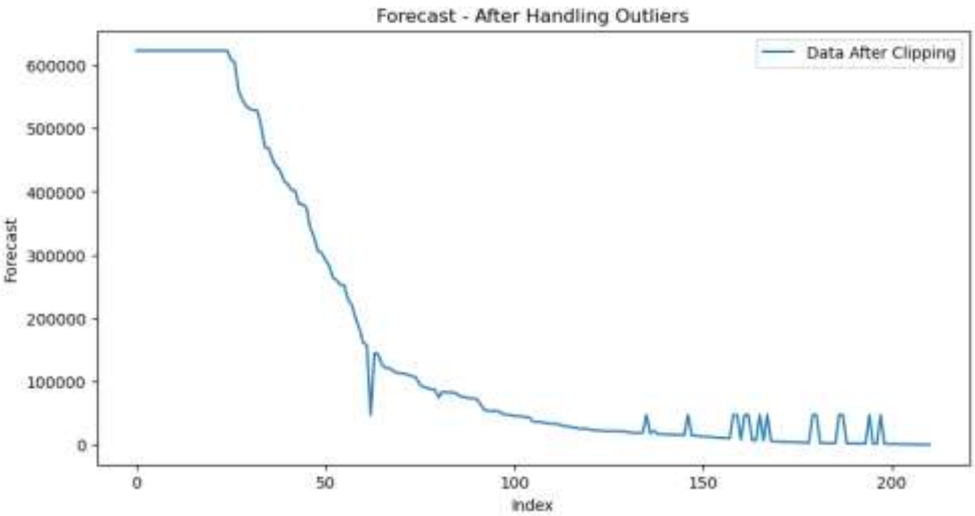


Here I am using the ‘clip’ method to cut any outliers found outside of the lower and upper bounds set previously. I have then plotted the data and set the labels, title, and legend.



```
# Handling the outliers by clipping
df['Forecast'] = df['Forecast'].clip(lower=lower_bound, upper=upper_bound)

# Plotting the data after handling the outliers to view the change
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Forecast'], label='Data After Clipping')

# Setting the labels, title, and legend
plt.xlabel('Index')
plt.ylabel('Forecast')
plt.title('Forecast - After Handling Outliers')
plt.legend()
plt.show()
```

Here I have shown the results in a graph that shows the data after clipping the outliers. The dataset is now clean.



Challenged Faced

Typos: Typo errors have been found in every data collection. It is not a good practice to possess inconsistent and inaccurate data, so the "loc" method has been used to find the row or rows that require correction and clean up mistakes.

Column duplicates: Multiple columns have been duplicated which is a problem to have as an extra column filled with copied data doesn’t help and isn’t useful for any operations or analysis. The duplicate columns can be eliminated by using the ‘drop’ method and specifying the exact column name and axis.

Row duplicates: There have been instances of row duplicates which can lead to inaccurate information if not handled properly. The rows have been dropped using the ‘drop duplicates’ method that searches for duplicate values and gets rid of them.

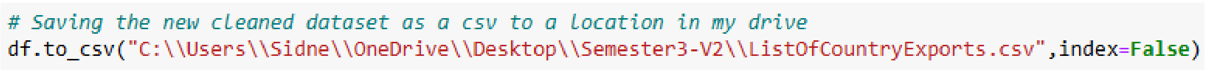
Incorrect data type: I have identified and corrected the incorrect data type columns, restoring them to their original data type. Converting columns into the proper data types is crucial because it guarantees that data is gathered and analysed accurately. When functions are applied, incorrect data types can cause them to malfunction. For instance, a column with an object data type won't function when summary statistics techniques like mean, median, mode, etc. are used.

Outliers: Every dataset contains outliers, which are eliminated to cut down on noise because they have the potential to distort the findings of data analysis. Any outliers discovered outside of the lower and upper bounds set were removed using the "clip" procedure. This will guarantee the trustworthiness of the results and the validity of statistical tests by preventing inaccurate approximations.

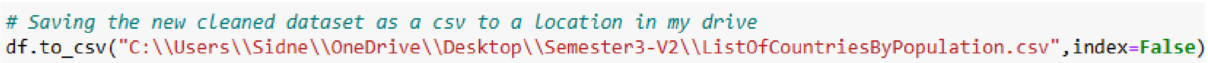
PART C - Data Importation

Task 5: Store Datasets

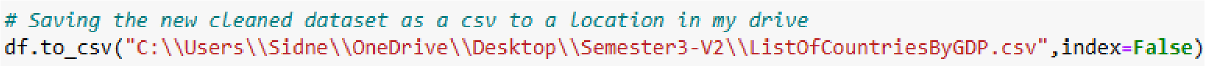
Saving Cleaned Data



```
# Saving the new cleaned dataset as a csv to a location in my drive
df.to_csv("C:\\Users\\Sidne\\OneDrive\\Desktop\\Semester3-V2\\ListOfCountryExports.csv",index=False)
```



```
# Saving the new cleaned dataset as a csv to a location in my drive
df.to_csv("C:\\Users\\Sidne\\OneDrive\\Desktop\\Semester3-V2\\ListOfCountriesByPopulation.csv",index=False)
```

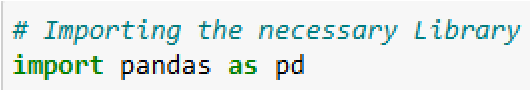


```
# Saving the new cleaned dataset as a csv to a location in my drive
df.to_csv("C:\\Users\\Sidne\\OneDrive\\Desktop\\Semester3-V2\\ListOfCountriesByGDP.csv",index=False)
```

Above I have used the lines of code to save the cleaned and transformed dataset as a CSV file and chosen the location in which to save it to via the file path.

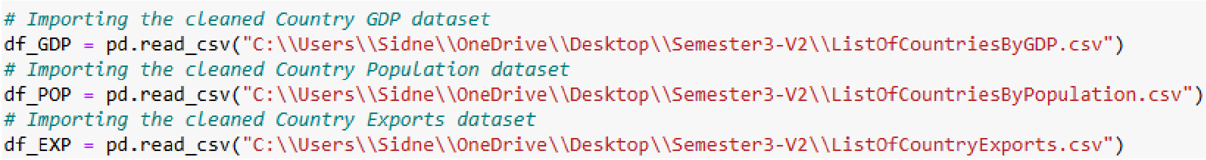
Task 6: Merge Data

Inner merging of the Three Datasets



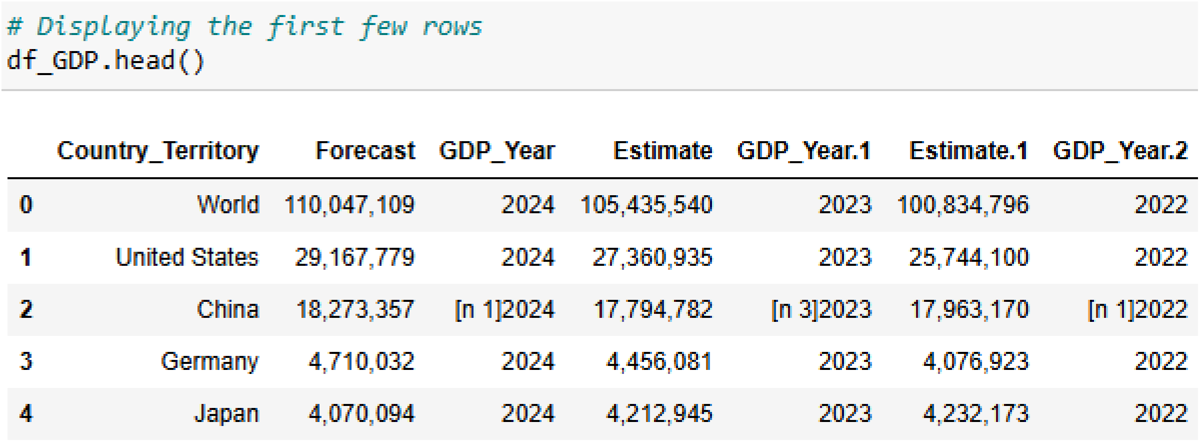
```python
# Importing the necessary Library
import pandas as pd
```

I am importing the pandas library to perform the merging operations.



```python
# Importing the cleaned Country GDP dataset
df_GDP = pd.read_csv("C:\\Users\\Sidne\\OneDrive\\Desktop\\Semester3-V2\\ListOfCountriesByGDP.csv")
# Importing the cleaned Country Population dataset
df_POP = pd.read_csv("C:\\Users\\Sidne\\OneDrive\\Desktop\\Semester3-V2\\ListOfCountriesByPopulation.csv")
# Importing the cleaned Country Exports dataset
df_EXP = pd.read_csv("C:\\Users\\Sidne\\OneDrive\\Desktop\\Semester3-V2\\ListOfCountryExports.csv")
```

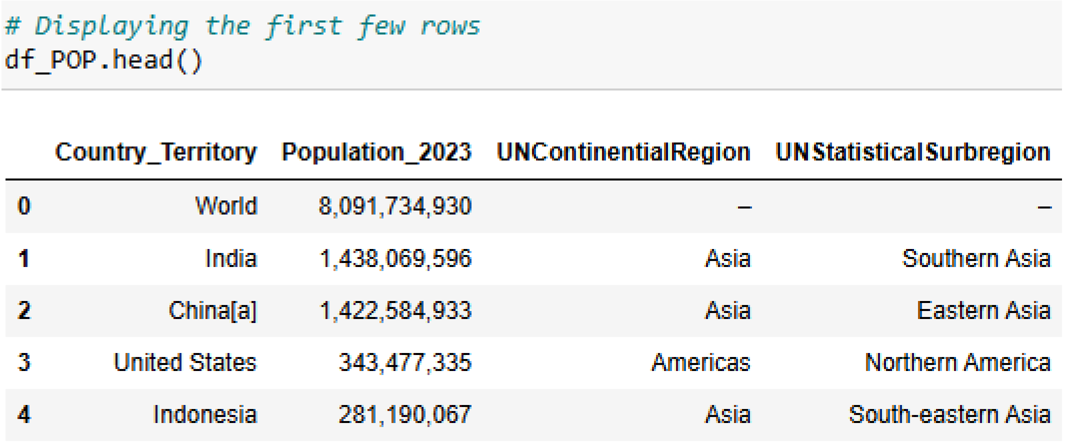
Here I am importing the cleaned datasets and assigning a different variable to each of them.



```python
# Displaying the first few rows
df_GDP.head()
```

	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year.1	Estimate.1	GDP_Year.2
0	World	110,047,109	2024	105,435,540	2023	100,834,796	2022
1	United States	29,167,779	2024	27,360,935	2023	25,744,100	2022
2	China	18,273,357	[n 1]2024	17,794,782	[n 3]2023	17,963,170	[n 1]2022
3	Germany	4,710,032	2024	4,456,081	2023	4,076,923	2022
4	Japan	4,070,094	2024	4,212,945	2023	4,232,173	2022

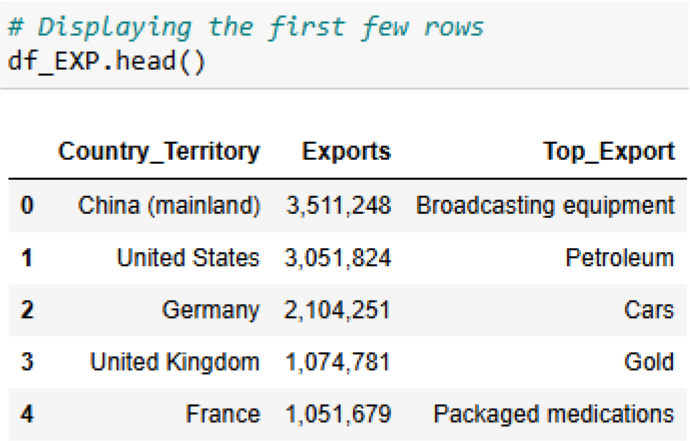
Displaying the first few rows for the country GDP datasets.



```python
# Displaying the first few rows
df_POP.head()
```

	Country_Territory	Population_2023	UNContinentialRegion	UNStatisticalSurbregion
0	World	8,091,734,930	–	–
1	India	1,438,069,596	Asia	Southern Asia
2	China[a]	1,422,584,933	Asia	Eastern Asia
3	United States	343,477,335	Americas	Northern America
4	Indonesia	281,190,067	Asia	South-eastern Asia

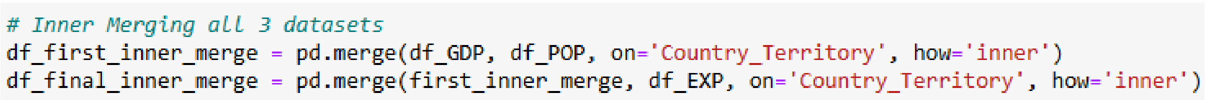
Displaying the first few rows for the country POP datasets.



```
# Displaying the first few rows
df_EXP.head()
```

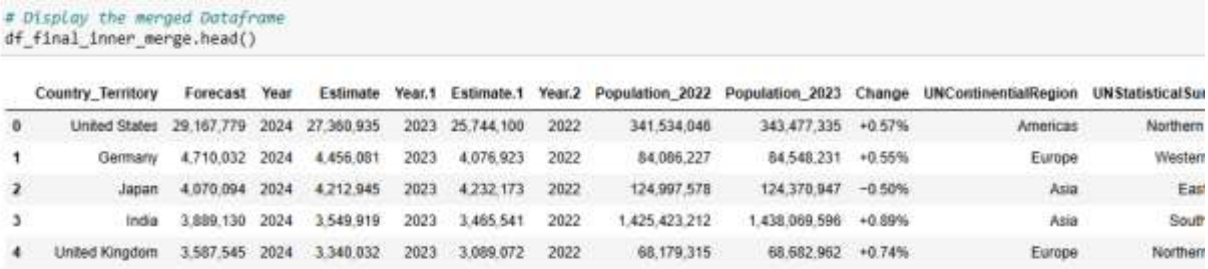
	Country_Territory	Exports	Top_Export
0	China (mainland)	3,511,248	Broadcasting equipment
1	United States	3,051,824	Petroleum
2	Germany	2,104,251	Cars
3	United Kingdom	1,074,781	Gold
4	France	1,051,679	Packaged medications

Displaying the first few rows for the country EXP datasets.



```
# Inner Merging all 3 datasets
df_first_inner_merge = pd.merge(df_GDP, df_POP, on='Country_Territory', how='inner')
df_final_inner_merge = pd.merge(first_inner_merge, df_EXP, on='Country_Territory', how='inner')
```

Here I am using the ‘pd.merge’ method to merge the chosen Data Frames of ‘Country GDP’ and ‘Country Population’ based on the ‘Country Territory’ and using an inner merge. I am then merging the merged Data Frame with the last Data Frame based on the same conditions.



```
# Display the merged Dataframe
df_final_inner_merge.head()
```

	Country_Territory	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNContinentialRegion	UNStatisticalSu
0	United States	29,167,779	2024	27,360,935	2023	25,744,100	2022	341,534,046	343,477,335	+0.57%	Americas	Northern
1	Germany	4,710,032	2024	4,456,081	2023	4,076,923	2022	84,086,227	84,548,231	+0.55%	Europe	Wester
2	Japan	4,070,094	2024	4,212,945	2023	4,232,173	2022	124,997,578	124,370,947	−0.50%	Asia	Eas
3	India	3,889,130	2024	3,549,919	2023	3,465,541	2022	1,425,423,212	1,438,069,596	+0.89%	Asia	Sout
4	United Kingdom	3,587,545	2024	3,340,032	2023	3,089,072	2022	68,179,315	68,682,962	+0.74%	Europe	Northern

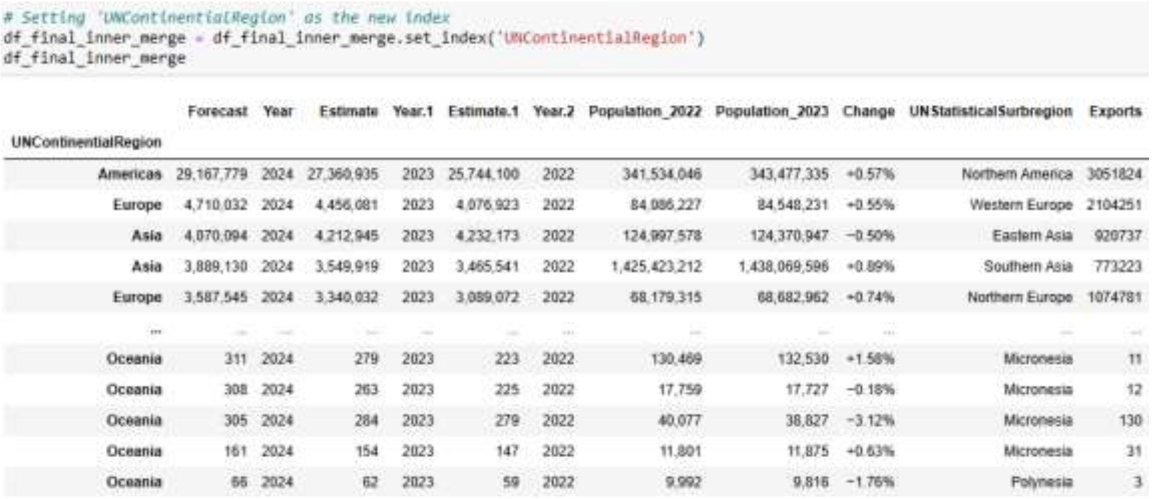
Here I am displaying the final inner merged Data Frame.

I have chosen the inner merge for this operation based on the column ‘Country Territory’ to create a new DataFrame that includes only those rows that have key and common values present in each Data Frame.

Task 7: Indexing

Filtering the Dataset

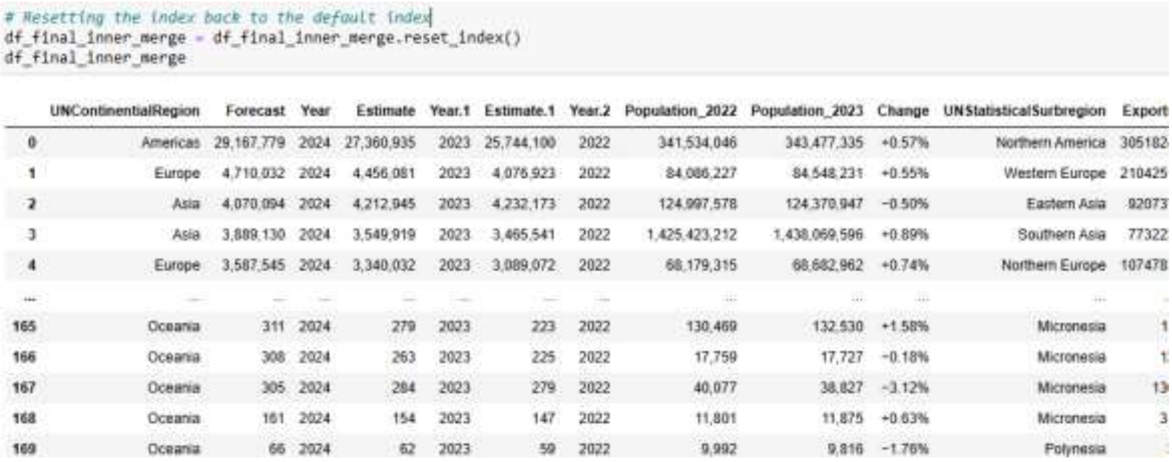
Setting index by ‘UNContinentalRegion’.



```
# Setting 'UNContinentialRegion' as the new index
df_final_inner_merge = df_final_inner_merge.set_index('UNContinentialRegion')
df_final_inner_merge
```

	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNStatisticalSurbregion	Exports
UNContinentialRegion											
Americas	29,167,779	2024	27,360,935	2023	25,744,100	2022	341,534,046	343,477,335	+0.57%	Northern America	3051824
Europe	4,710,032	2024	4,456,081	2023	4,076,923	2022	84,086,227	84,548,231	+0.55%	Western Europe	2104251
Asia	4,070,094	2024	4,212,945	2023	4,232,173	2022	124,997,578	124,370,947	−0.50%	Eastern Asia	920737
Asia	3,889,130	2024	3,549,919	2023	3,465,541	2022	1,425,423,212	1,438,069,596	+0.89%	Southern Asia	773223
Europe	3,587,545	2024	3,340,032	2023	3,089,072	2022	68,179,315	68,682,962	+0.74%	Northern Europe	1074781
...	...	...	...	...	...	...	...	...	...	...	...
Oceania	311	2024	279	2023	223	2022	130,469	132,530	+1.58%	Micronesia	11
Oceania	308	2024	263	2023	225	2022	17,759	17,727	−0.18%	Micronesia	12
Oceania	305	2024	284	2023	279	2022	40,077	38,827	−3.12%	Micronesia	130
Oceania	161	2024	154	2023	147	2022	11,801	11,875	+0.63%	Micronesia	31
Oceania	66	2024	62	2023	59	2022	9,992	9,816	−1.76%	Polynesia	3

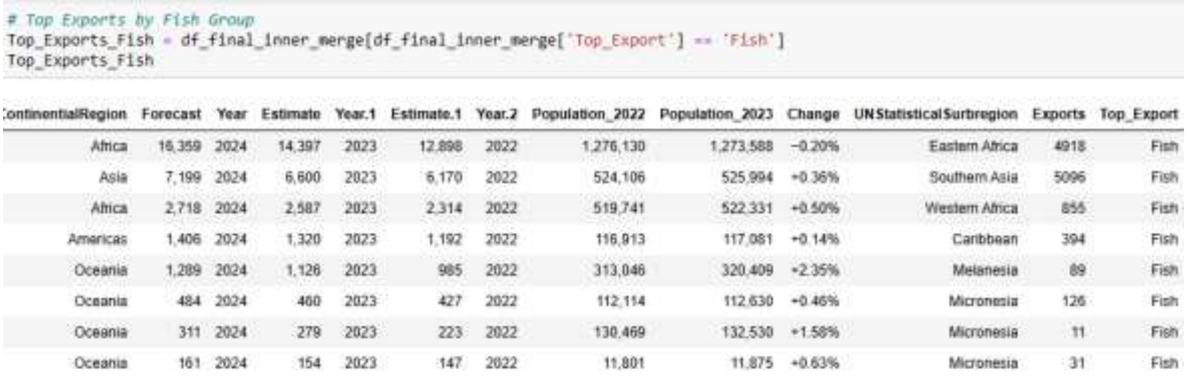
Resetting the index back to the default index.



```
# Resetting the index back to the default index
df_final_inner_merge = df_final_inner_merge.reset_index()
df_final_inner_merge
```

	UNContinentialRegion	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNStatisticalSurbregion	Export
0	Americas	29,167,779	2024	27,360,935	2023	25,744,100	2022	341,534,046	343,477,335	+0.57%	Northern America	305182
1	Europe	4,710,032	2024	4,456,081	2023	4,076,923	2022	84,086,227	84,548,231	+0.55%	Western Europe	210425
2	Asia	4,070,094	2024	4,212,945	2023	4,232,173	2022	124,997,578	124,370,947	−0.50%	Eastern Asia	92073
3	Asia	3,889,130	2024	3,549,919	2023	3,465,541	2022	1,425,423,212	1,438,069,596	+0.89%	Southern Asia	77322
4	Europe	3,587,545	2024	3,340,032	2023	3,089,072	2022	68,179,315	68,682,962	+0.74%	Northern Europe	107478
...	...	...	...	...	...	...	...	...	...	...	...	
165	Oceania	311	2024	279	2023	223	2022	130,469	132,530	+1.58%	Micronesia	1
166	Oceania	308	2024	263	2023	225	2022	17,759	17,727	−0.18%	Micronesia	1
167	Oceania	305	2024	284	2023	279	2022	40,077	38,827	−3.12%	Micronesia	13
168	Oceania	161	2024	154	2023	147	2022	11,801	11,875	+0.63%	Micronesia	3
169	Oceania	66	2024	62	2023	59	2022	9,992	9,816	−1.76%	Polynesia	

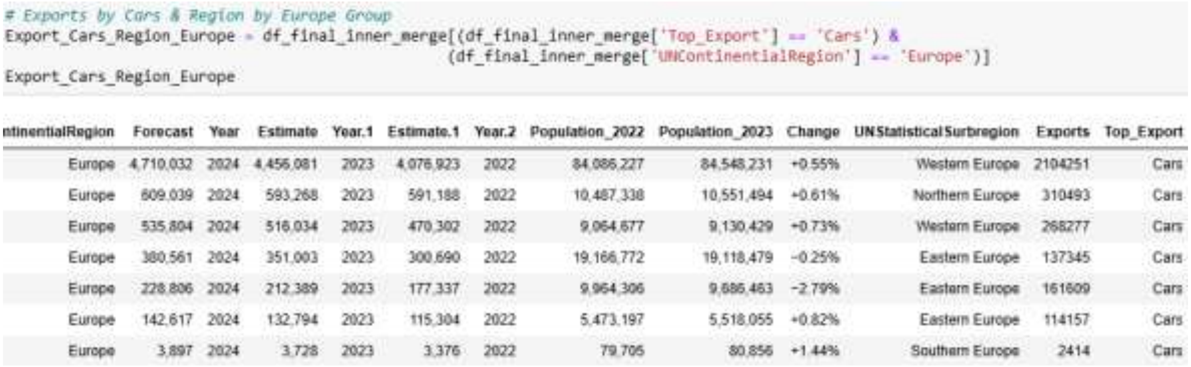
Filtering the Top Exports by Fish Group



```
# Top Exports by Fish Group
Top_Exports_Fish = df_final_inner_merge[df_final_inner_merge['Top_Export'] == 'Fish']
Top_Exports_Fish
```

ontinentialRegion	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNStatisticalSurbregion	Exports	Top_Export
Africa	16,359	2024	14,397	2023	12,898	2022	1,276,130	1,273,588	−0.20%	Eastern Africa	4918	Fish
Asia	7,199	2024	6,600	2023	6,170	2022	524,106	525,994	+0.36%	Southern Asia	5096	Fish
Africa	2,718	2024	2,587	2023	2,314	2022	519,741	522,331	+0.50%	Western Africa	855	Fish
Americas	1,406	2024	1,320	2023	1,192	2022	116,913	117,081	+0.14%	Caribbean	394	Fish
Oceania	1,289	2024	1,126	2023	985	2022	313,046	320,409	+2.35%	Melanesia	89	Fish
Oceania	484	2024	460	2023	427	2022	112,114	112,630	+0.46%	Micronesia	126	Fish
Oceania	311	2024	279	2023	223	2022	130,469	132,530	+1.58%	Micronesia	11	Fish
Oceania	161	2024	154	2023	147	2022	11,801	11,875	+0.63%	Micronesia	31	Fish

Filtering the Exports by Cars & Region by Europe Group



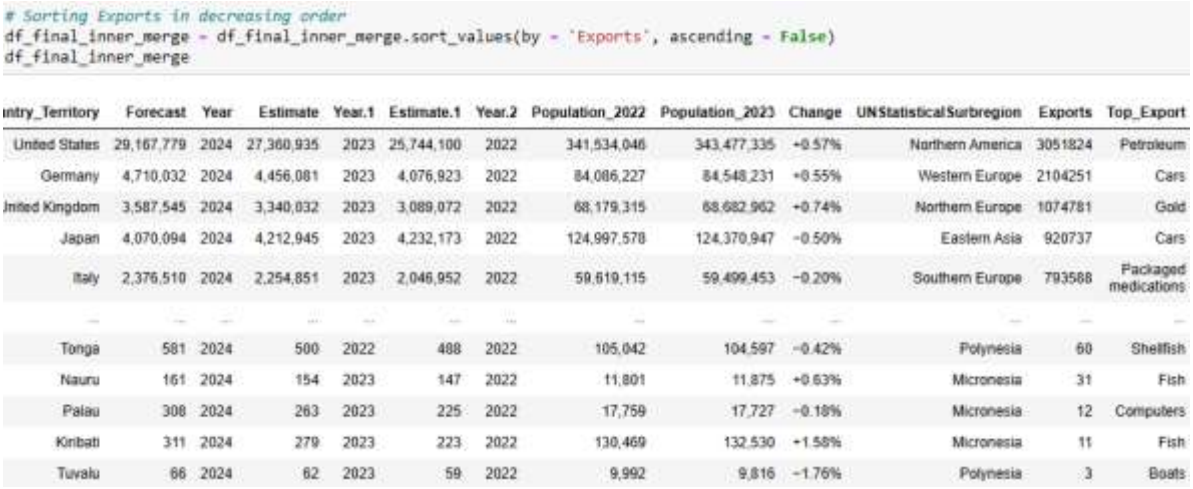
```python
# Exports by Cars & Region by Europe Group
Export_Cars_Region_Europe = df_final_inner_merge[(df_final_inner_merge['Top_Export'] == 'Cars') &
                                                 (df_final_inner_merge['UNContinentialRegion'] == 'Europe')]
Export_Cars_Region_Europe
```

ntinentialRegion	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNStatisticalSurbregion	Exports	Top_Export
Europe	4,710,032	2024	4,456,081	2023	4,076,923	2022	84,086,227	84,548,231	+0.55%	Western Europe	2104251	Cars
Europe	609,039	2024	593,268	2023	591,188	2022	10,487,338	10,551,494	+0.61%	Northern Europe	310493	Cars
Europe	535,804	2024	516,034	2023	470,302	2022	9,064,677	9,130,429	+0.73%	Western Europe	268277	Cars
Europe	380,561	2024	351,003	2023	300,690	2022	19,166,772	19,118,479	-0.25%	Eastern Europe	137345	Cars
Europe	228,806	2024	212,389	2023	177,337	2022	9,964,306	9,686,463	-2.79%	Eastern Europe	161609	Cars
Europe	142,617	2024	132,794	2023	115,304	2022	5,473,197	5,518,055	+0.82%	Eastern Europe	114157	Cars
Europe	3,897	2024	3,728	2023	3,376	2022	79,705	80,856	+1.44%	Southern Europe	2414	Cars

Task 8: Sorting

Sorting in Ascending and Descending Order

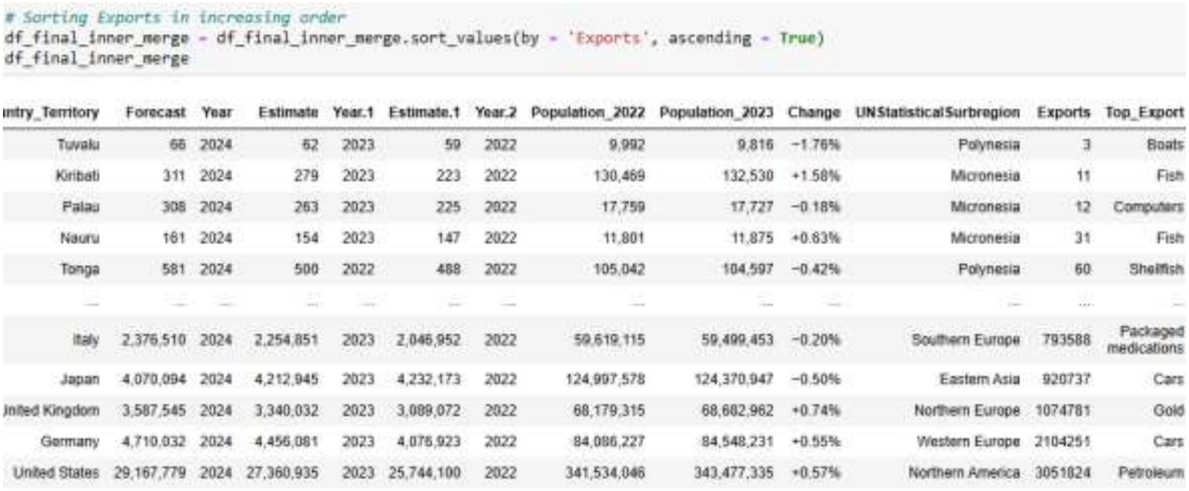
Here I have sorted the merged data by using the 'sort values' method to sort the dataset by the 'Exports' column in ascending order. My approach to this is that it can clearly show the highest amount of costs while in line with the other rows following behind. This is useful to check the top exports, country territory, GDP forecast and estimates, and population at the time from highest exports lowering in order, checking the top performing countries.



```python
# Sorting Exports in decreasing order
df_final_inner_merge = df_final_inner_merge.sort_values(by = 'Exports', ascending = False)
df_final_inner_merge
```

intry_Territory	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNStatisticalSurbregion	Exports	Top_Export
United States	29,167,779	2024	27,360,935	2023	25,744,100	2022	341,534,046	343,477,335	+0.57%	Northern America	3051824	Petroleum
Germany	4,710,032	2024	4,456,081	2023	4,076,923	2022	84,086,227	84,548,231	+0.55%	Western Europe	2104251	Cars
United Kingdom	3,587,545	2024	3,340,032	2023	3,089,072	2022	68,179,315	68,682,962	+0.74%	Northern Europe	1074781	Gold
Japan	4,070,094	2024	4,212,945	2023	4,232,173	2022	124,997,578	124,370,947	-0.50%	Eastern Asia	920737	Cars
Italy	2,376,510	2024	2,254,851	2023	2,046,952	2022	59,619,115	59,499,453	-0.20%	Southern Europe	793588	Packaged medications
...	...	...	...	...	...	...	...	...	...	...	...	...
Tonga	581	2024	500	2022	488	2022	105,042	104,597	-0.42%	Polynesia	60	Shellfish
Nauru	161	2024	154	2023	147	2022	11,801	11,875	+0.63%	Micronesia	31	Fish
Palau	308	2024	263	2023	225	2022	17,759	17,727	-0.18%	Micronesia	12	Computers
Kiribati	311	2024	279	2023	223	2022	130,469	132,530	+1.58%	Micronesia	11	Fish
Tuvalu	66	2024	62	2023	59	2022	9,992	9,816	-1.76%	Polynesia	3	Boats

Here, I've used the "sort values" method to arrange the merged data in descending order by the "Exports" column. My approach to this, it that it can clearly display the lowest costs while positioned alongside the column rows that follow. To determine which countries are performing the worst, it is helpful to look at the lowest exports, national territory, GDP prediction and estimations, and population during that time.



```
# Sorting Exports in increasing order
df_final_inner_merge = df_final_inner_merge.sort_values(by = 'Exports', ascending = True)
df_final_inner_merge
```

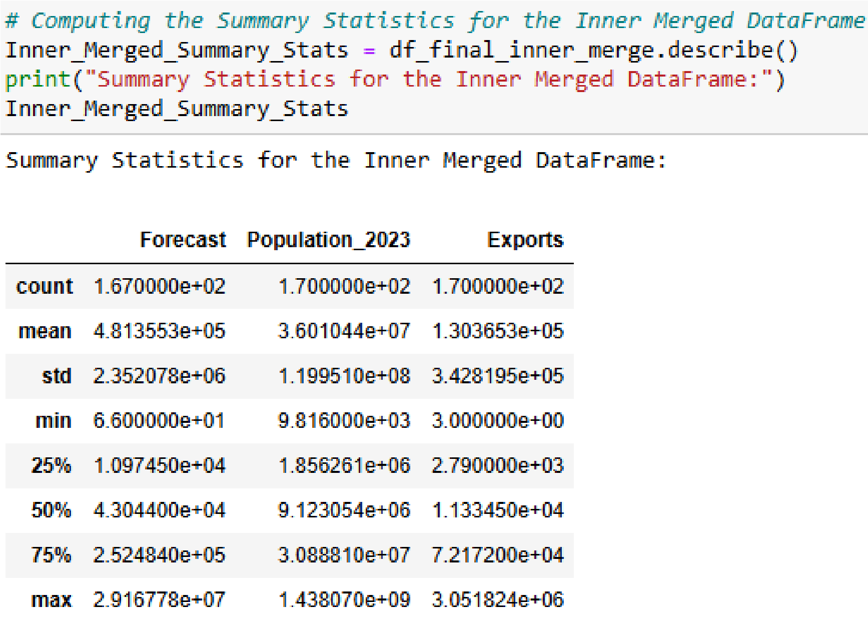
intry_Territory	Forecast	Year	Estimate	Year.1	Estimate.1	Year.2	Population_2022	Population_2023	Change	UNStatisticalSurbregion	Exports	Top_Export
Tuvalu	66	2024	62	2023	59	2022	9,992	9,816	−1.76%	Polynesia	3	Boats
Kiribati	311	2024	279	2023	223	2022	130,469	132,530	+1.58%	Micronesia	11	Fish
Palau	308	2024	263	2023	225	2022	17,759	17,727	−0.18%	Micronesia	12	Computers
Nauru	161	2024	154	2023	147	2022	11,801	11,875	+0.63%	Micronesia	31	Fish
Tonga	581	2024	500	2022	488	2022	105,042	104,597	−0.42%	Polynesia	60	Shellfish
...	...	...	...	...	...	...	...	...	...	...	...	...
Italy	2,376,510	2024	2,254,851	2023	2,046,952	2022	59,619,115	59,499,453	−0.20%	Southern Europe	793588	Packaged medications
Japan	4,070,094	2024	4,212,945	2023	4,232,173	2022	124,997,578	124,370,947	−0.50%	Eastern Asia	920737	Cars
Jnited Kingdom	3,587,545	2024	3,340,032	2023	3,089,072	2022	68,179,315	68,682,962	+0.74%	Northern Europe	1074781	Gold
Germany	4,710,032	2024	4,456,081	2023	4,076,923	2022	84,086,227	84,548,231	+0.55%	Western Europe	2104251	Cars
United States	29,167,779	2024	27,360,935	2023	25,744,100	2022	341,534,046	343,477,335	+0.57%	Northern America	3051824	Petroleum

Task 9: Summary Statistics

Interpretation and Analysis

I have computed the summary statistics for the inner merged Data Frame by using the 'describe' method. It provides a numerical description of the dataset's key features. It shows the count, average, standard deviation, minimum, inter quartile ranges, and the maximum for each numerical column. From this, we can see that for the 'Forecast' column, the minimum amount is $660.00 in GDP, the maximum amount is $29,167,780.00 in GDP, and the average was at $481,355.30. From the 'Population_2023' column, we can see that the minimum population count was at 9,816, the maximum population count was at 1,438,070,000, and the average was at 36,010,440. For the 'Exports' column the minimum amount of exports was $3.00, the maximum amount of exports was $3,051,824.00, and the average amount was $130,365.30. From this impossible data, I can conclude that an average country may have an average population 36,010,440, an average export amount of

$130,365.30, and an average GDP of $481,355.30.



```python
# Computing the Summary Statistics for the Inner Merged DataFrame
Inner_Merged_Summary_Stats = df_final_inner_merge.describe()
print("Summary Statistics for the Inner Merged DataFrame:")
Inner_Merged_Summary_Stats
```

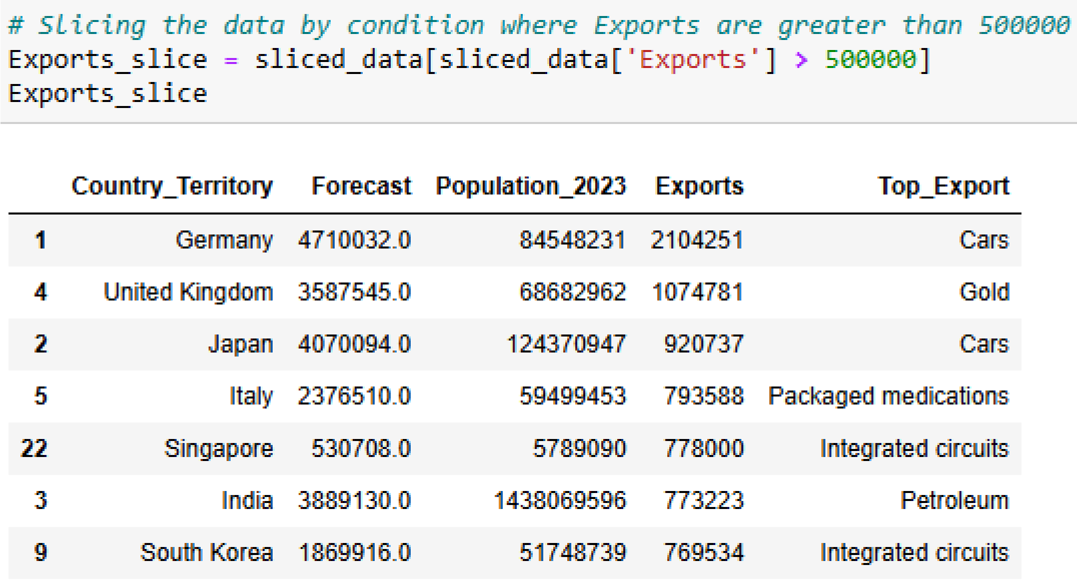
```
Summary Statistics for the Inner Merged DataFrame:
```

	Forecast	Population_2023	Exports
count	1.670000e+02	1.700000e+02	1.700000e+02
mean	4.813553e+05	3.601044e+07	1.303653e+05
std	2.352078e+06	1.199510e+08	3.428195e+05
min	6.600000e+01	9.816000e+03	3.000000e+00
25%	1.097450e+04	1.856261e+06	2.790000e+03
50%	4.304400e+04	9.123054e+06	1.133450e+04
75%	2.524840e+05	3.088810e+07	7.217200e+04
max	2.916778e+07	1.438070e+09	3.051824e+06

Task 10: Slicing

Exports Greater then 500,000

Here I am slicing the data by condition where the values in the ‘Exports’ column is greater than $500,000. I have only included the columns that I am interested in and have performed the slice operations as displayed below. My approach to this is to view what are the top performing exports. Here we see that there are 7 results which adhere to the condition applied with their respective country name, GDP forecast amount, population, export total, and type of export.



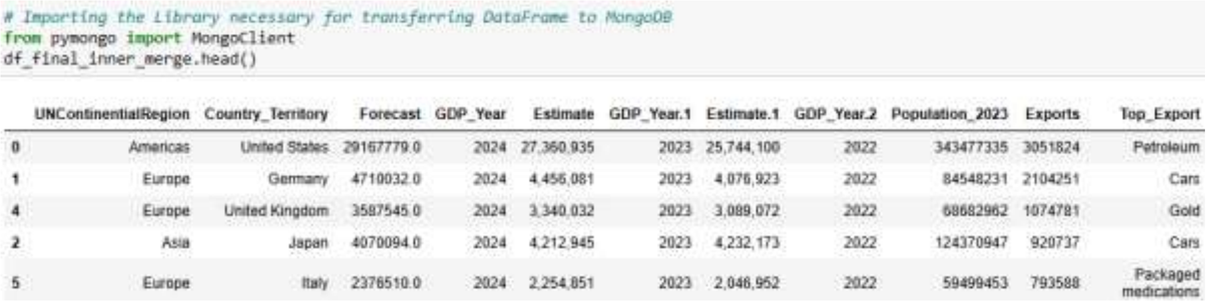
```python
# Slicing the data by condition where Exports are greater than 500000
Exports_slice = sliced_data[sliced_data['Exports'] > 500000]
Exports_slice
```

	Country_Territory	Forecast	Population_2023	Exports	Top_Export
1	Germany	4710032.0	84548231	2104251	Cars
4	United Kingdom	3587545.0	68682962	1074781	Gold
2	Japan	4070094.0	124370947	920737	Cars
5	Italy	2376510.0	59499453	793588	Packaged medications
22	Singapore	530708.0	5789090	778000	Integrated circuits
3	India	3889130.0	1438069596	773223	Petroleum
9	South Korea	1869916.0	51748739	769534	Integrated circuits

Task 11: Data Import

Importing into MongoDB

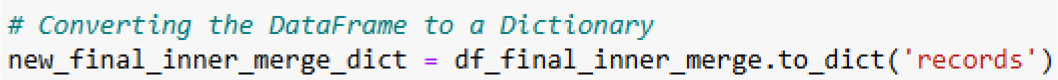
Here I am importing the Library necessary for transferring the DataFrame to MongoDB.



```
# Importing the Library necessary for transferring DataFrame to MongoDB
from pymongo import MongoClient
df_final_inner_merge.head()
```

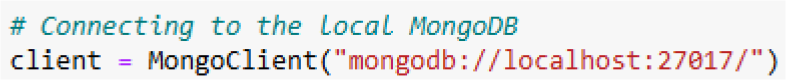
	UNContinentialRegion	Country_Territory	Forecast	GDP_Year	Estimate	GDP_Year.1	Estimate.1	GDP_Year.2	Population_2023	Exports	Top_Export
0	Americas	United States	29167779.0	2024	27,360,935	2023	25,744,100	2022	343477335	3051824	Petroleum
1	Europe	Germany	4710032.0	2024	4,456,081	2023	4,076,923	2022	84548231	2104251	Cars
4	Europe	United Kingdom	3587545.0	2024	3,340,032	2023	3,089,072	2022	68682962	1074781	Gold
2	Asia	Japan	4070094.0	2024	4,212,945	2023	4,232,173	2022	124370947	920737	Cars
5	Europe	Italy	2376510.0	2024	2,254,851	2023	2,046,952	2022	59499453	793588	Packaged medications

Here I am converting the Data Frame into a dictionary using the ‘to dict’ method under records and assigning a new variable name to it. This is due to MongoDB requiring JSON-like documents for conversion.



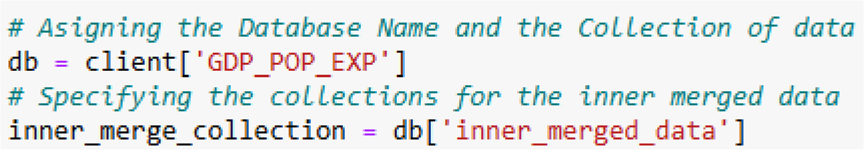
```
# Converting the DataFrame to a Dictionary
new_final_inner_merge_dict = df_final_inner_merge.to_dict('records')
```

Here I am connecting to my local MongoDB server using the ‘pymongo’ library.



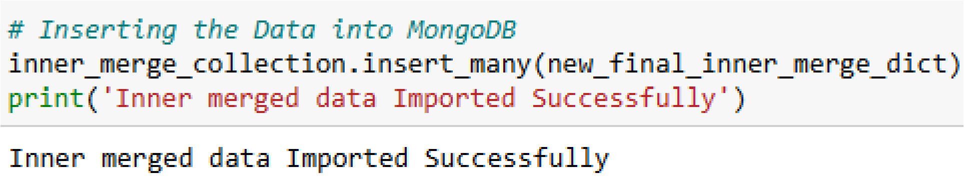
```
# Connecting to the local MongoDB
client = MongoClient("mongodb://localhost:27017/")
```

Here I am naming the database for MongoDB and creating a collection in the database for storing the inner merged dataset.



```
# Asigning the Database Name and the Collection of data
db = client['GDP_POP_EXP']
# Specifying the collections for the inner merged data
inner_merge_collection = db['inner_merged_data']
```

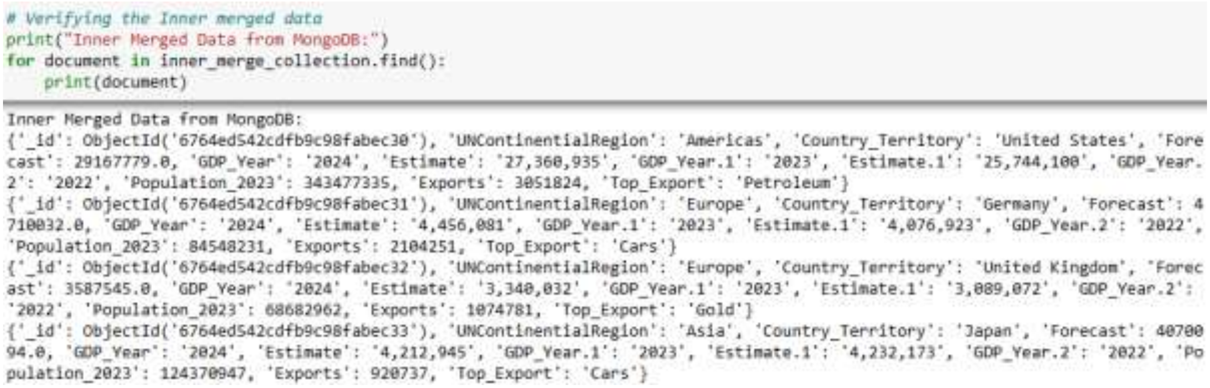
I am Inserting the converted dictionary of the inner merged data into the respective collections using the ‘insert many’ method. I have displayed the text to show that the import has been successful.



```
# Inserting the Data into MongoDB
inner_merge_collection.insert_many(new_final_inner_merge_dict)
print('Inner merged data Imported Successfully')
```

```
Inner merged data Imported Successfully
```

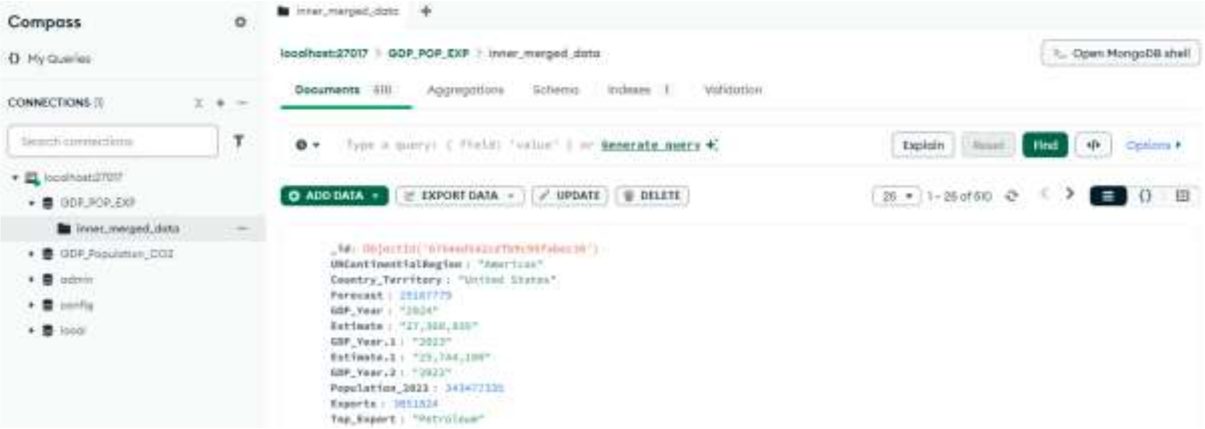
Here I am verifying the data import of the inner merged collection by querying the MongoDB collections with the ‘for loop’ and ‘find’ methods to find and print the document of the final dataset.



```
# Verifying the Inner merged data
print("Inner Merged Data from MongoDB:")
for document in inner_merge_collection.find():
    print(document)
```

```
Inner Merged Data from MongoDB:
{'_id': ObjectId('6764ed542cdfb9c98fabec30'), 'UNContinentialRegion': 'Americas', 'Country_Territory': 'United States', 'Fore
cast': 29167779.0, 'GDP_Year': '2024', 'Estimate': '27,360,935', 'GDP_Year.1': '2023', 'Estimate.1': '25,744,100', 'GDP_Year.
2': '2022', 'Population_2023': 343477335, 'Exports': 3051824, 'Top_Export': 'Petroleum'}
{'_id': ObjectId('6764ed542cdfb9c98fabec31'), 'UNContinentialRegion': 'Europe', 'Country_Territory': 'Germany', 'Forecast': 4
710032.0, 'GDP_Year': '2024', 'Estimate': '4,456,081', 'GDP_Year.1': '2023', 'Estimate.1': '4,076,923', 'GDP_Year.2': '2022',
'Population_2023': 84548231, 'Exports': 2104251, 'Top_Export': 'Cars'}
{'_id': ObjectId('6764ed542cdfb9c98fabec32'), 'UNContinentialRegion': 'Europe', 'Country_Territory': 'United Kingdom', 'Forec
ast': 3587545.0, 'GDP_Year': '2024', 'Estimate': '3,340,032', 'GDP_Year.1': '2023', 'Estimate.1': '3,089,072', 'GDP_Year.2':
'2022', 'Population_2023': 68682962, 'Exports': 1074781, 'Top_Export': 'Gold'}
{'_id': ObjectId('6764ed542cdfb9c98fabec33'), 'UNContinentialRegion': 'Asia', 'Country_Territory': 'Japan', 'Forecast': 40700
94.0, 'GDP_Year': '2024', 'Estimate': '4,212,945', 'GDP_Year.1': '2023', 'Estimate.1': '4,232,173', 'GDP_Year.2': '2022', 'Po
pulation_2023': 124370947, 'Exports': 920737, 'Top_Export': 'Cars'}
```

The data base has successfully been transferred to the MongoDB.



```
_id: ObjectId('6764ed542cdfb9c98fabec30')
UNContinentialRegion : "Americas"
Country_Territory : "United States"
Forecast : 29167779
GDP_Year : "2024"
Estimate : "27,360,935"
GDP_Year.1 : "2023"
Estimate.1 : "25,744,100"
GDP_Year.2 : "2022"
Population_2023 : 343477335
Exports : 3051824
Top_Export : "Petroleum"
```

References

Wikipedia. (2024, November 15). *List of countries by exports*. Wikipedia: https://en.wikipedia.org/wiki/List_of_countries_by_exports

Wikipedia. (2024, December 18). *List of countries by GDP (nominal)*. Wikipedia: https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)

Wikipedia. (2024, December 12). *List of countries by population (United Nations)*. Wikipedia: https://en.wikipedia.org/wiki/List_of_countries_by_population_(United_Nations)