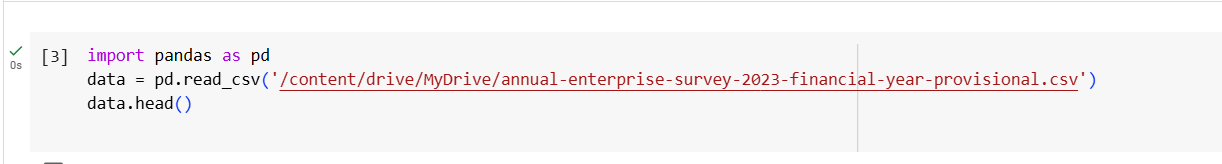
**Hexaware Technologies**

Python-Coding Challenge

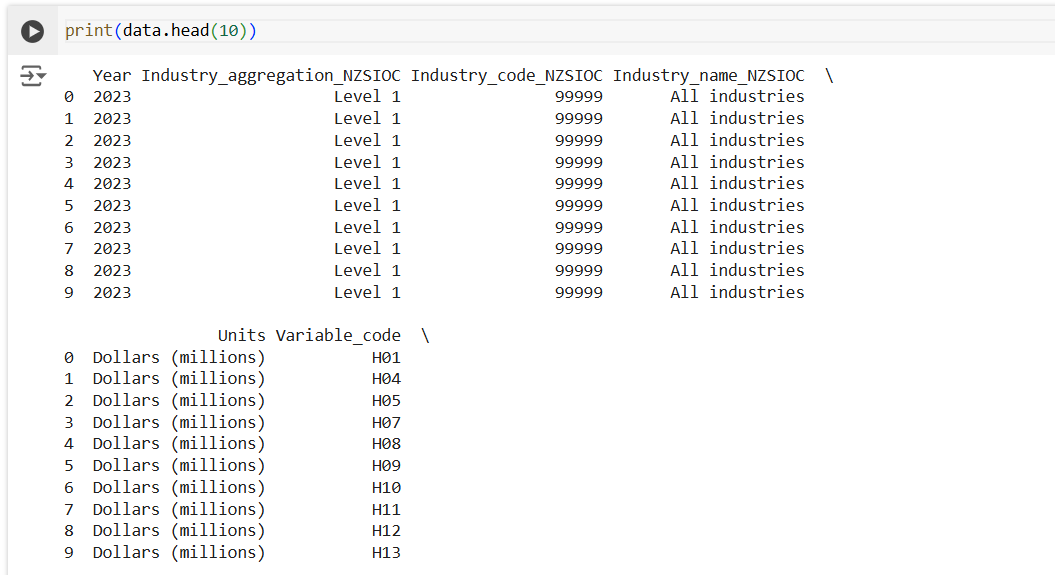
**Step 1: Importing Libraries and Files**

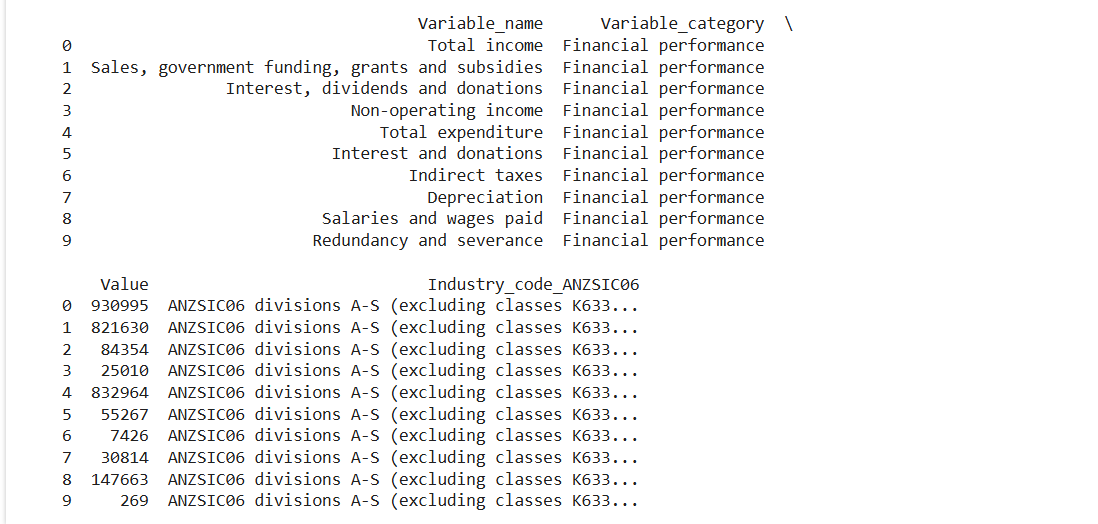
****

* **import pandas as pd:** Imports the Pandas library, which is essential for data manipulation and analysis in Python.
* **pd.read\_csv():** This function is used to load the CSV file into a Pandas DataFrame. The file path is passed as an argument, and the dataset is stored in the data variable for further processing.

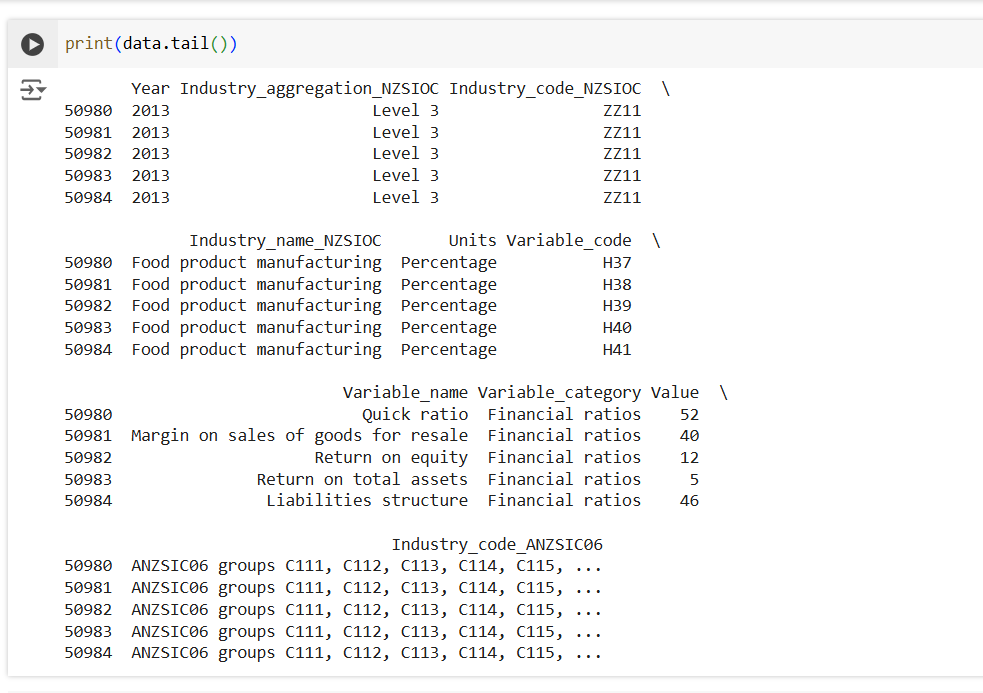
1. **Printing rows of the Data**

Using **head():** The function is used to display the first 10 rows of the DataFrame. By default, head() returns the first 5 rows, but you can specify the number of rows (10 in this case) to show more.

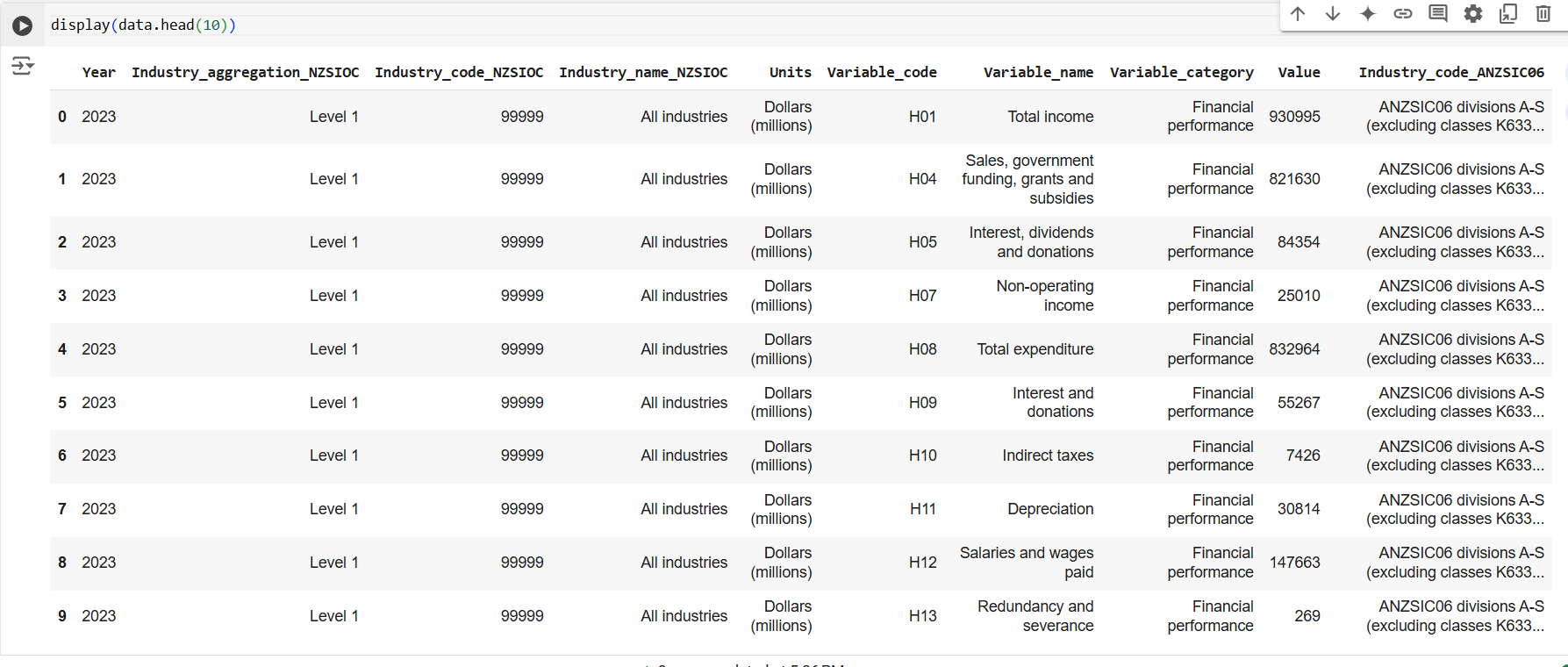
****

****

Using **tail():** The tail() function displays the last 5 rows of the DataFrame**.**

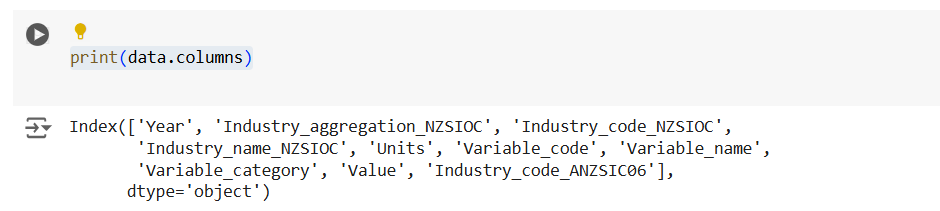
****

Using **display():** The display() function is specific to environments like Google Colab and Jupyter Notebooks.

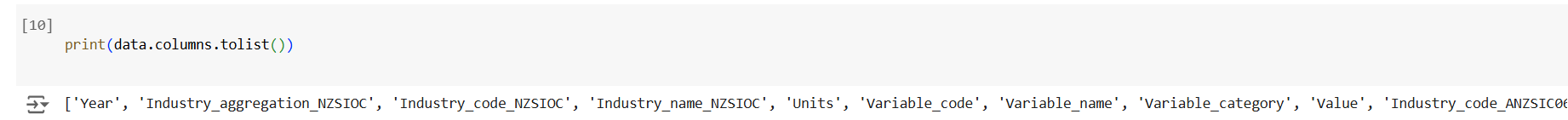
****

1. **Printing the column names of the DataFrame**

Using **data.columns:** The data.columns attribute returns the column names of the DataFrame as an Index object. This method is quick and efficient for listing the column names directly.

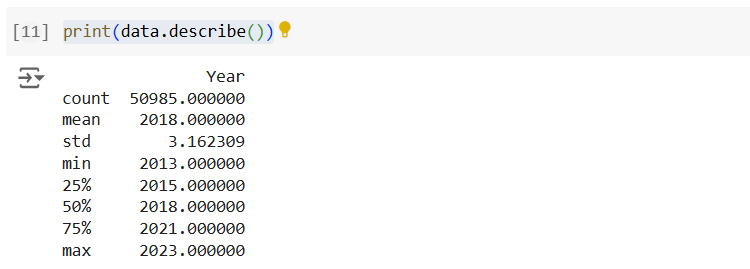


Using **data.columns.tolist():** The tolist() function is used to convert the column names from the Index object into a Python list. This is useful when you want to work with the column names as a list for further operations.

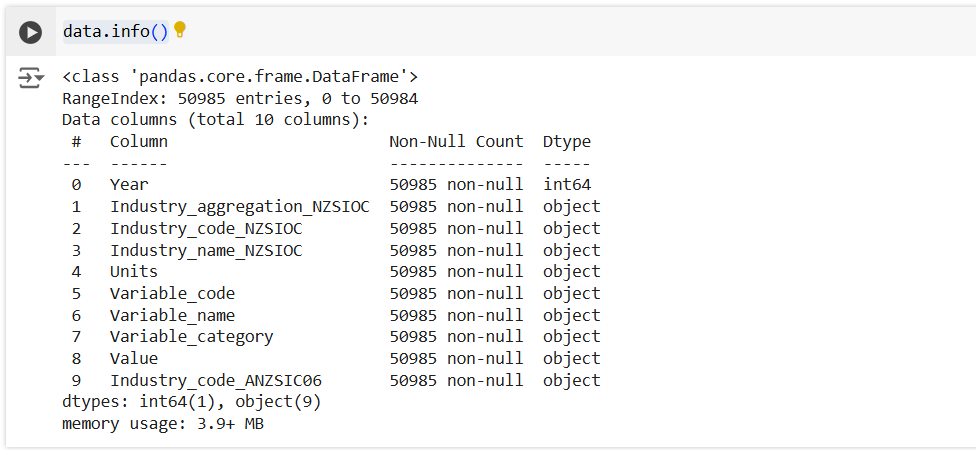


1. **Summary of Data Frame**

Using **data.describe():** The describe() function provides a summary of the DataFrame, including statistical measures such as the mean, standard deviation, minimum, maximum, and quartiles for numeric columns.

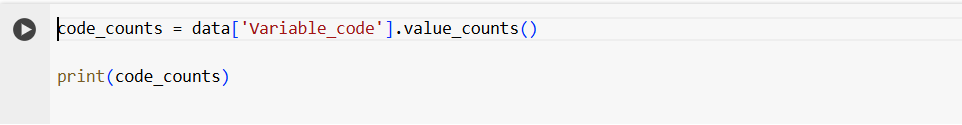
****

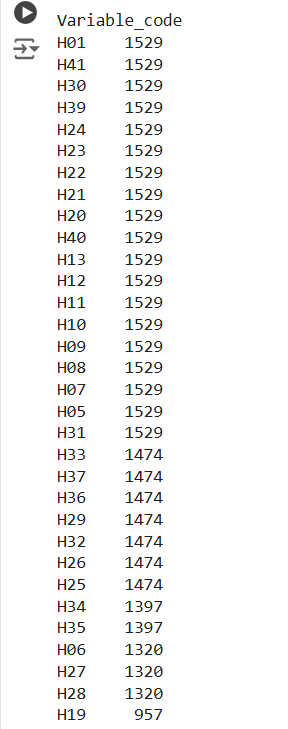
Using **data.info() :** The info() method provides a concise summary of the DataFrame, including the number of entries (rows), the column names, the data types of each column, and the number of non-null values.

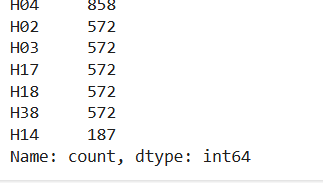


1. **Descriptive Statistical Measures of a DataFrame**

Using **value\_counts() :** The value\_counts() function counts the occurrences of each unique value in a specific column—in this case, the 'Variable\_code' column.

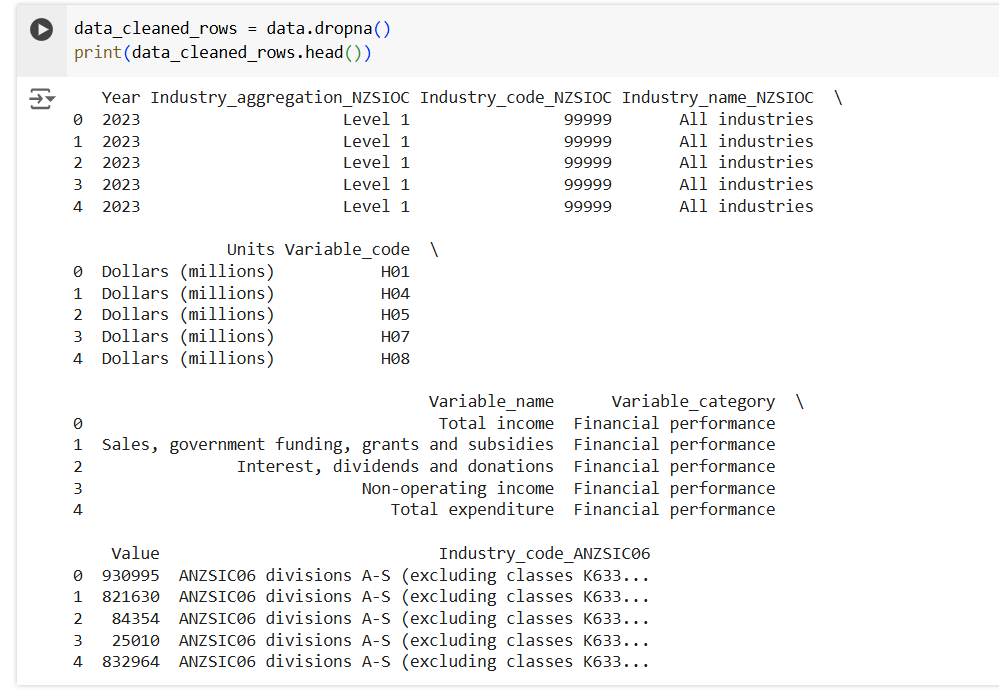
****

****

****

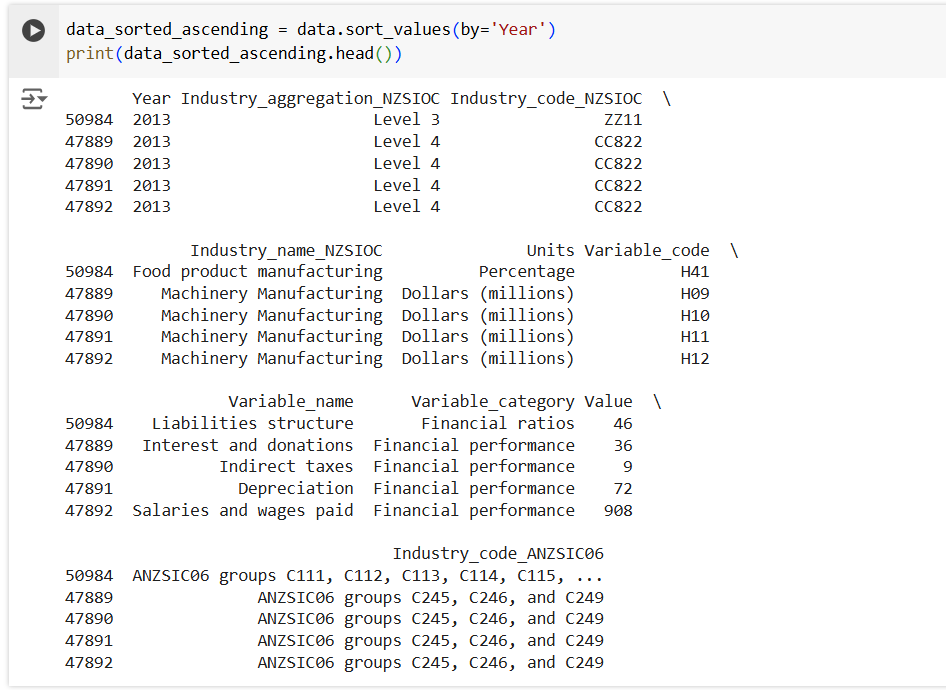
1. **Missing Data Handing**

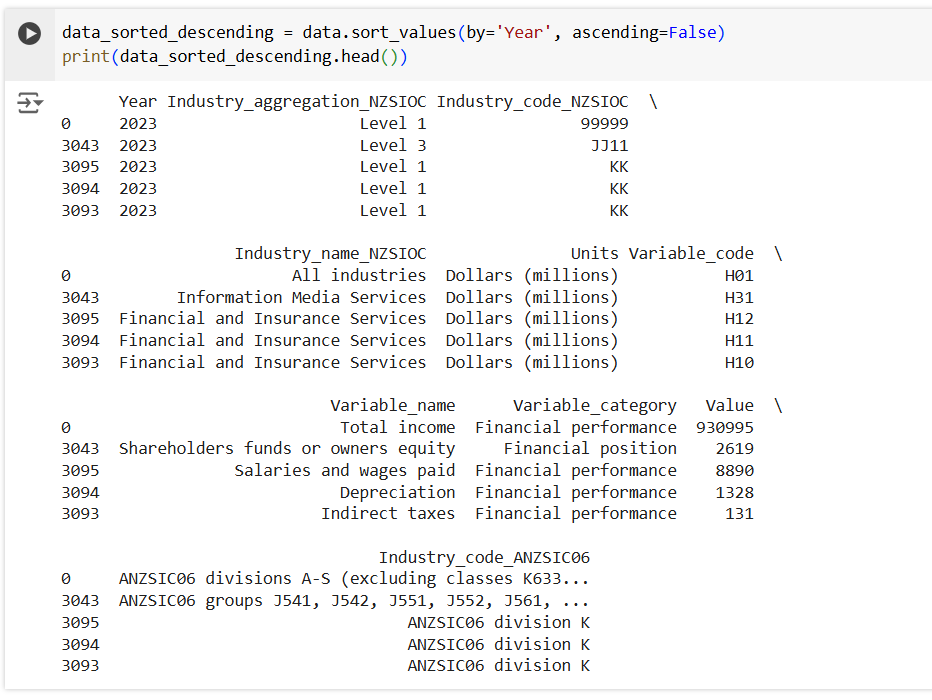
**Using dropna() :** The dropna() function removes any rows with missing values (NaNs) from the DataFrame. This method is useful when you want to eliminate incomplete records and ensure that your dataset only contains rows with complete data.

****

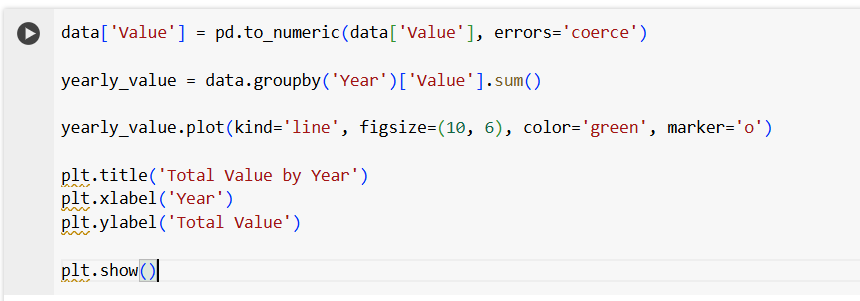
1. **Sorting DataFrame values**

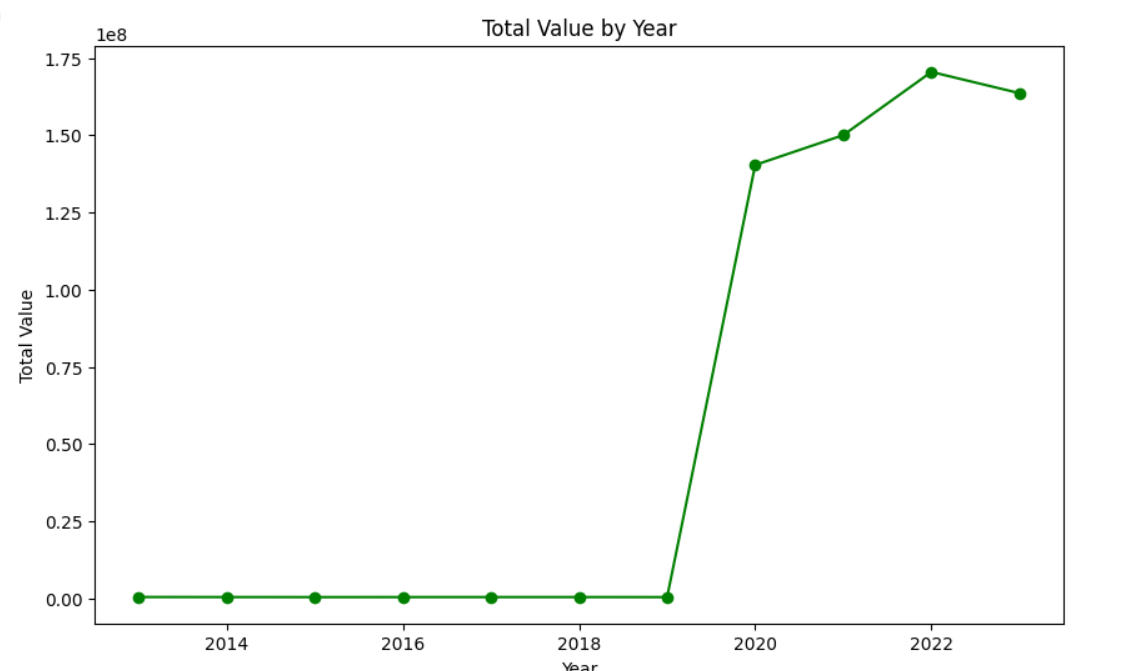
Using **sort\_values():** The sort\_values() function sorts the DataFrame based on the values of one or more columns. In this case, we sorted the data by the 'Year' column in ascending order.



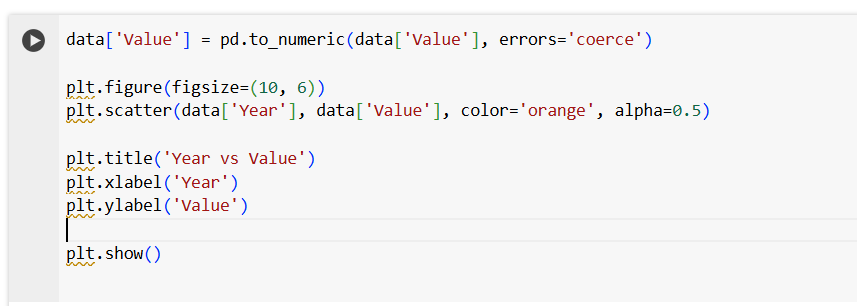


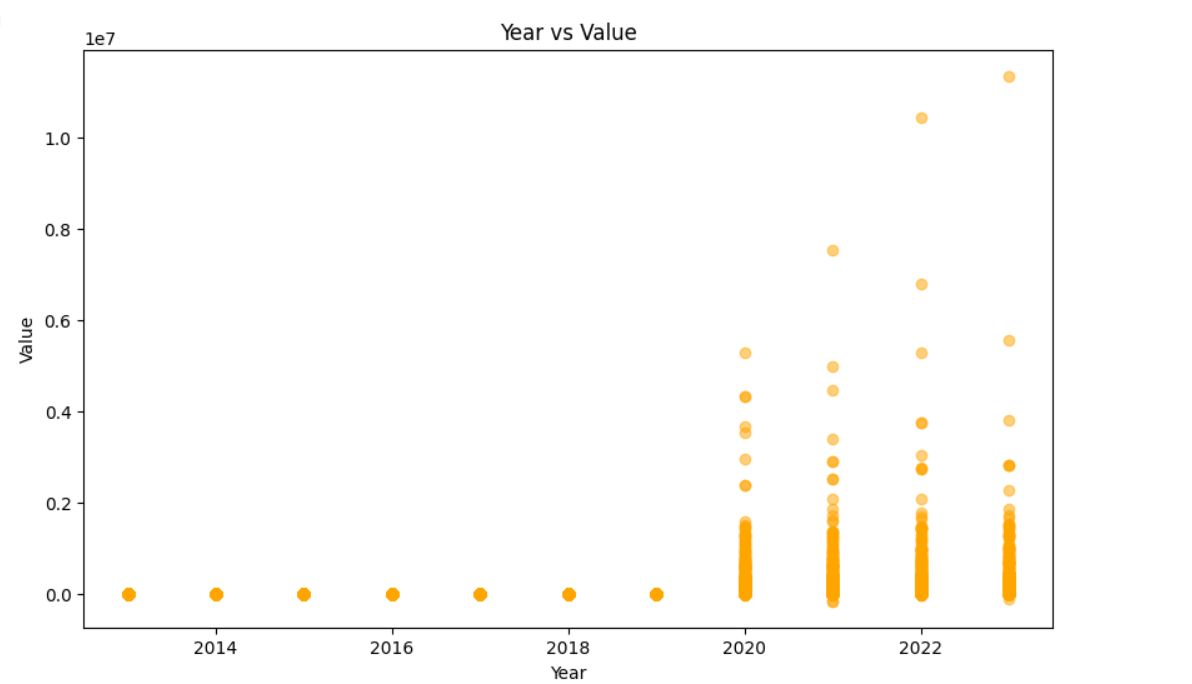
1. **Visualizing DataFrame  
    Line Plot:** In this method, we first convert the 'Value' column to numeric using **pd.to\_numeric()** to handle any non-numeric values (using errors='coerce' to turn invalid entries into NaN). Then, we group the data by 'Year' and calculate the sum of 'Value' for each year using groupby() and sum(). The resulting yearly\_value is plotted as a line plot to visualize the total value over the years. The marker='o' adds markers to each data point.

****

****

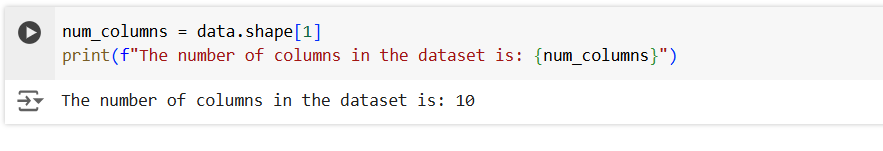
**Scatter Plot :** In this approach, we again convert the 'Value' column to numeric. Then, we create a scatter plot to visualize the relationship between 'Year' and 'Value'. Scatter plots are useful for showing how one variable correlates with another, and in this case, it helps in examining the distribution of values over different years. The alpha=0.5 adds transparency to the points, making the plot more readable when points overlap.

****

****

1. **What is the number of columns in the dataset?**

**Using data.shape:** The data.shape attribute returns the dimensions of the DataFrame as a tuple: (number of rows, number of columns). By accessing the second element with [1], we retrieve the number of columns in the dataset.

****