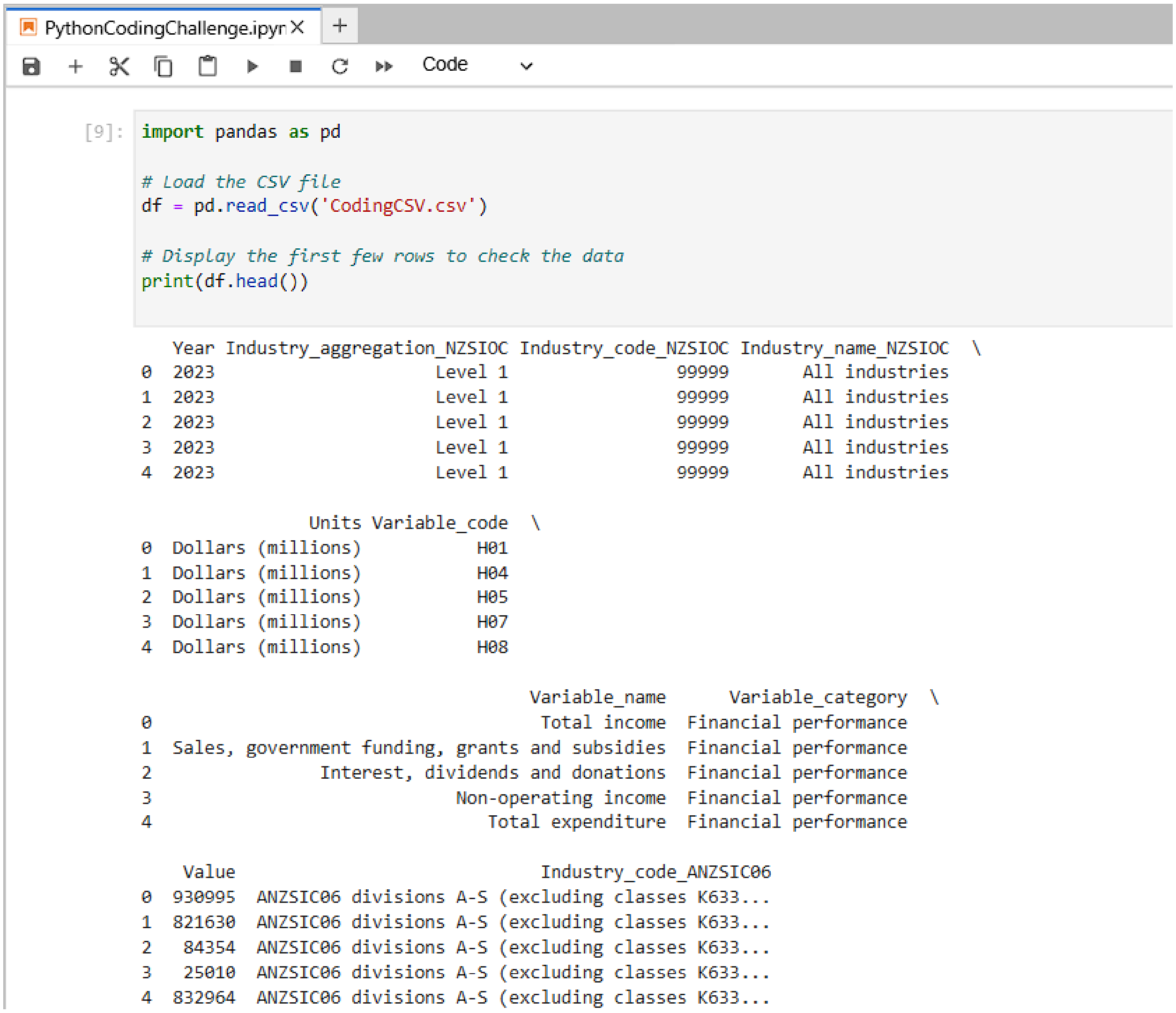
# Data Engineering Python Coding Challenge

## DE120 - J Jatin

## Load the CSV File:

- **Explanation:** Before analyzing or manipulating any data, you need to load it into a pandas DataFrame. The CSV file is read using pandas.read_csv() which converts the file into a tabular format.

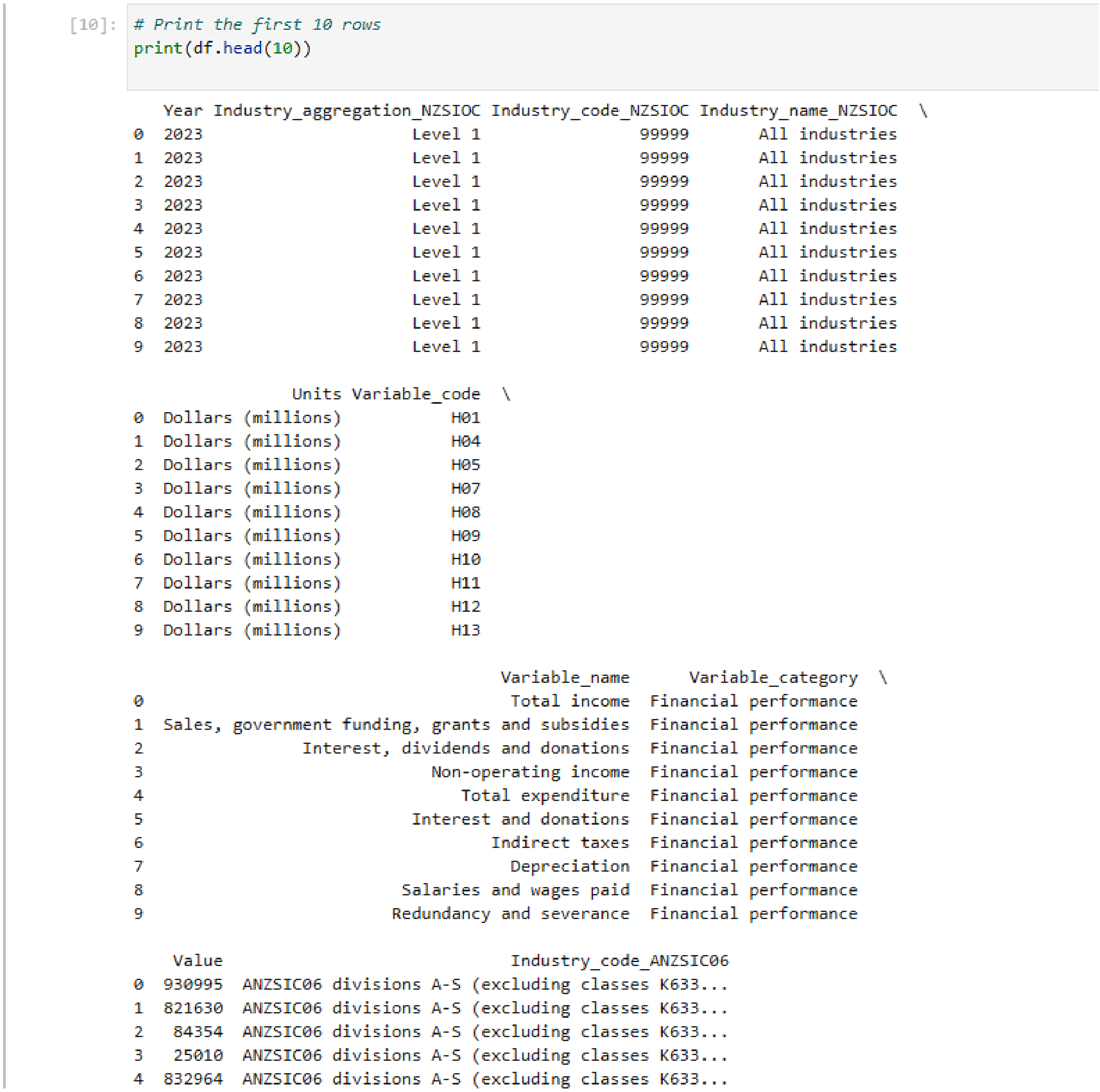


```python
import pandas as pd

# Load the CSV file
df = pd.read_csv('CodingCSV.csv')

# Display the first few rows to check the data
print(df.head())
```

```
   Year Industry_aggregation_NZSIOC Industry_code_NZSIOC Industry_name_NZSIOC  \
0  2023                     Level 1                99999       All industries
1  2023                     Level 1                99999       All industries
2  2023                     Level 1                99999       All industries
3  2023                     Level 1                99999       All industries
4  2023                     Level 1                99999       All industries

                Units Variable_code  \
0  Dollars (millions)           H01
1  Dollars (millions)           H04
2  Dollars (millions)           H05
3  Dollars (millions)           H07
4  Dollars (millions)           H08

                                     Variable_name      Variable_category  \
0                                     Total income  Financial performance
1  Sales, government funding, grants and subsidies  Financial performance
2                Interest, dividends and donations  Financial performance
3                             Non-operating income  Financial performance
4                                Total expenditure  Financial performance

    Value                             Industry_code_ANZSIC06
0  930995  ANZSIC06 divisions A-S (excluding classes K633...
1  821630  ANZSIC06 divisions A-S (excluding classes K633...
2   84354  ANZSIC06 divisions A-S (excluding classes K633...
3   25010  ANZSIC06 divisions A-S (excluding classes K633...
4  832964  ANZSIC06 divisions A-S (excluding classes K633...
```

## 1. Print Rows of the Data:

- **Explanation:** To get an idea of what your data looks like, you can print the first or last few rows of the DataFrame using .head() or .tail().

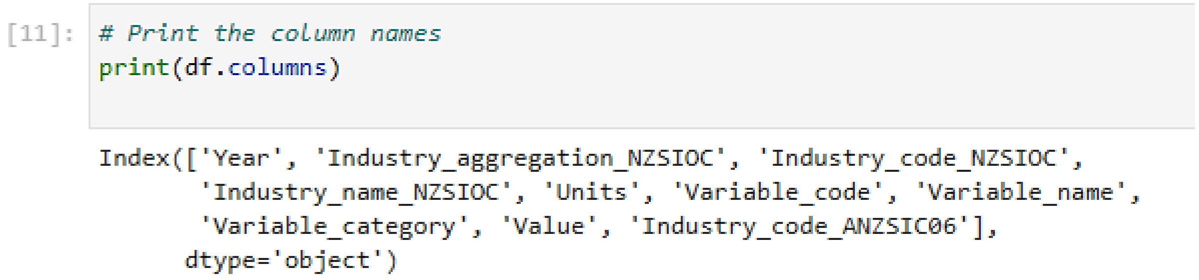


```
[10]: # Print the first 10 rows
      print(df.head(10))
```

```
   Year Industry_aggregation_NZSIOC Industry_code_NZSIOC Industry_name_NZSIOC  \
0  2023                     Level 1                99999       All industries   
1  2023                     Level 1                99999       All industries   
2  2023                     Level 1                99999       All industries   
3  2023                     Level 1                99999       All industries   
4  2023                     Level 1                99999       All industries   
5  2023                     Level 1                99999       All industries   
6  2023                     Level 1                99999       All industries   
7  2023                     Level 1                99999       All industries   
8  2023                     Level 1                99999       All industries   
9  2023                     Level 1                99999       All industries   

                Units Variable_code  \
0  Dollars (millions)           H01   
1  Dollars (millions)           H04   
2  Dollars (millions)           H05   
3  Dollars (millions)           H07   
4  Dollars (millions)           H08   
5  Dollars (millions)           H09   
6  Dollars (millions)           H10   
7  Dollars (millions)           H11   
8  Dollars (millions)           H12   
9  Dollars (millions)           H13   

                                     Variable_name      Variable_category  \
0                                     Total income  Financial performance   
1  Sales, government funding, grants and subsidies  Financial performance   
2                Interest, dividends and donations  Financial performance   
3                             Non-operating income  Financial performance   
4                                Total expenditure  Financial performance   
5                           Interest and donations  Financial performance   
6                                   Indirect taxes  Financial performance   
7                                     Depreciation  Financial performance   
8                           Salaries and wages paid  Financial performance   
9                          Redundancy and severance  Financial performance   

    Value                             Industry_code_ANZSIC06  
0  930995  ANZSIC06 divisions A-S (excluding classes K633...  
1  821630  ANZSIC06 divisions A-S (excluding classes K633...  
2   84354  ANZSIC06 divisions A-S (excluding classes K633...  
3   25010  ANZSIC06 divisions A-S (excluding classes K633...  
4  832964  ANZSIC06 divisions A-S (excluding classes K633...  
```

## 2. Print the Column Names:

- **Explanation:** DataFrames consist of columns with names. Printing the column names helps you know what data is available and how to reference it.

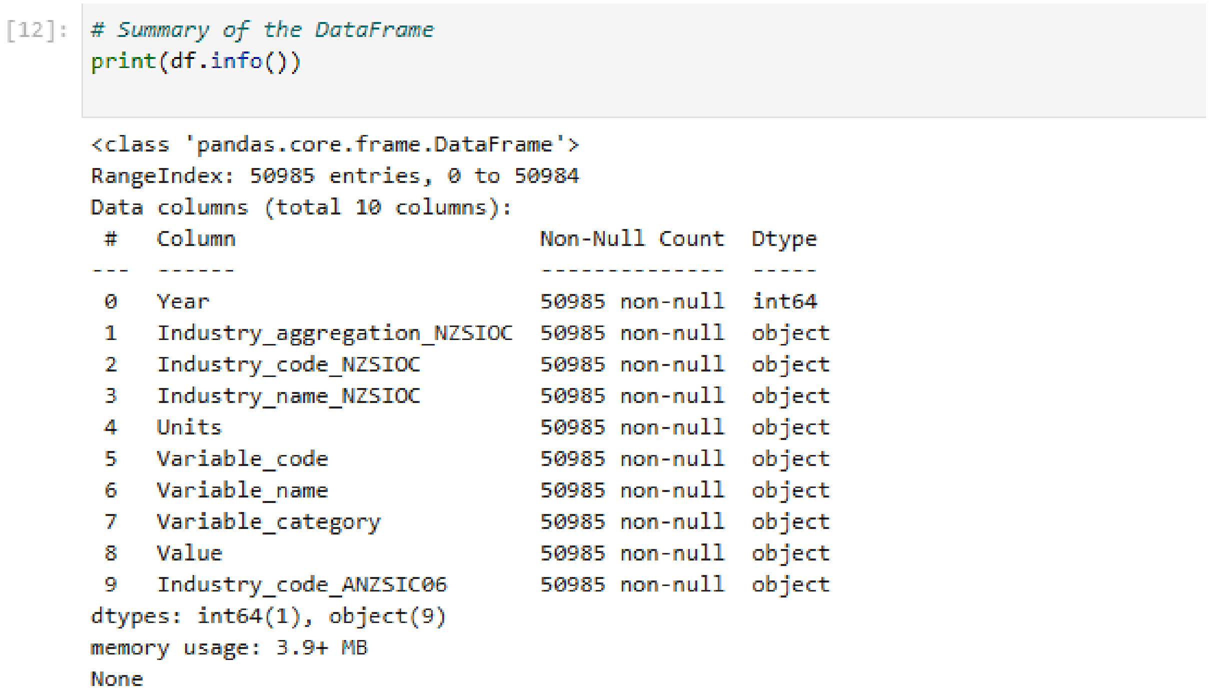


```
[11]: # Print the column names
      print(df.columns)
```

```
Index(['Year', 'Industry_aggregation_NZSIOC', 'Industry_code_NZSIOC',
       'Industry_name_NZSIOC', 'Units', 'Variable_code', 'Variable_name',
       'Variable_category', 'Value', 'Industry_code_ANZSIC06'],
      dtype='object')
```

## 3. Summary of the DataFrame:

- **Explanation:** The .info() function provides a concise summary, including the data types of columns, non-null counts, and memory usage.

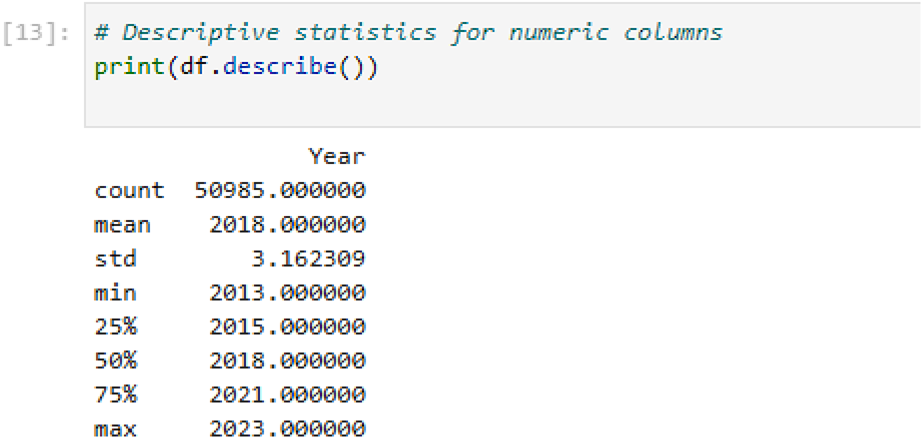


```
[12]: # Summary of the DataFrame
      print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50985 entries, 0 to 50984
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   Year                         50985 non-null  int64 
 1   Industry_aggregation_NZSIOC  50985 non-null  object
 2   Industry_code_NZSIOC         50985 non-null  object
 3   Industry_name_NZSIOC         50985 non-null  object
 4   Units                        50985 non-null  object
 5   Variable_code                50985 non-null  object
 6   Variable_name                50985 non-null  object
 7   Variable_category            50985 non-null  object
 8   Value                        50985 non-null  object
 9   Industry_code_ANZSIC06       50985 non-null  object
dtypes: int64(1), object(9)
memory usage: 3.9+ MB
None
```

## 4. Descriptive Statistical Measures:

- **Explanation:** The .describe() method provides basic statistical summaries (e.g., mean, standard deviation, minimum, and maximum) for numerical columns.

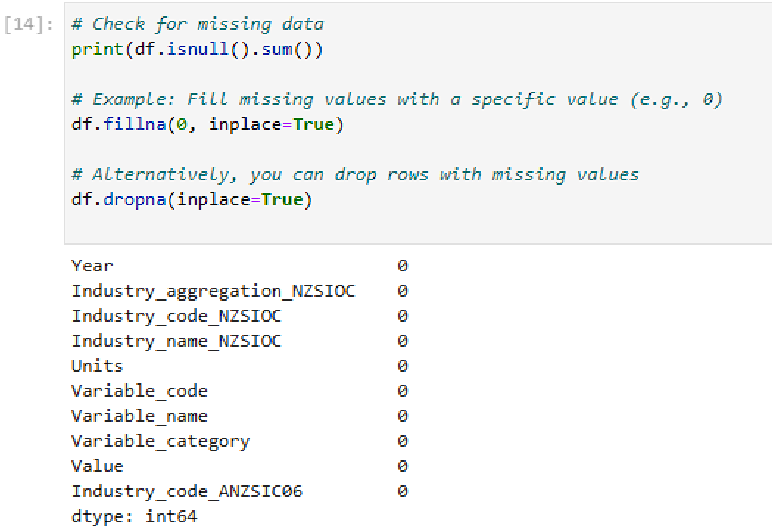


```
[13]: # Descriptive statistics for numeric columns
      print(df.describe())
```

```
               Year
count  50985.000000
mean    2018.000000
std        3.162309
min     2013.000000
25%     2015.000000
50%     2018.000000
75%     2021.000000
max     2023.000000
```

## 5. Handling Missing Data:

- **Explanation:** Missing data is common in datasets. You can check for missing values and decide whether to fill them (e.g., with 0 or a mean value) or drop rows with missing data.



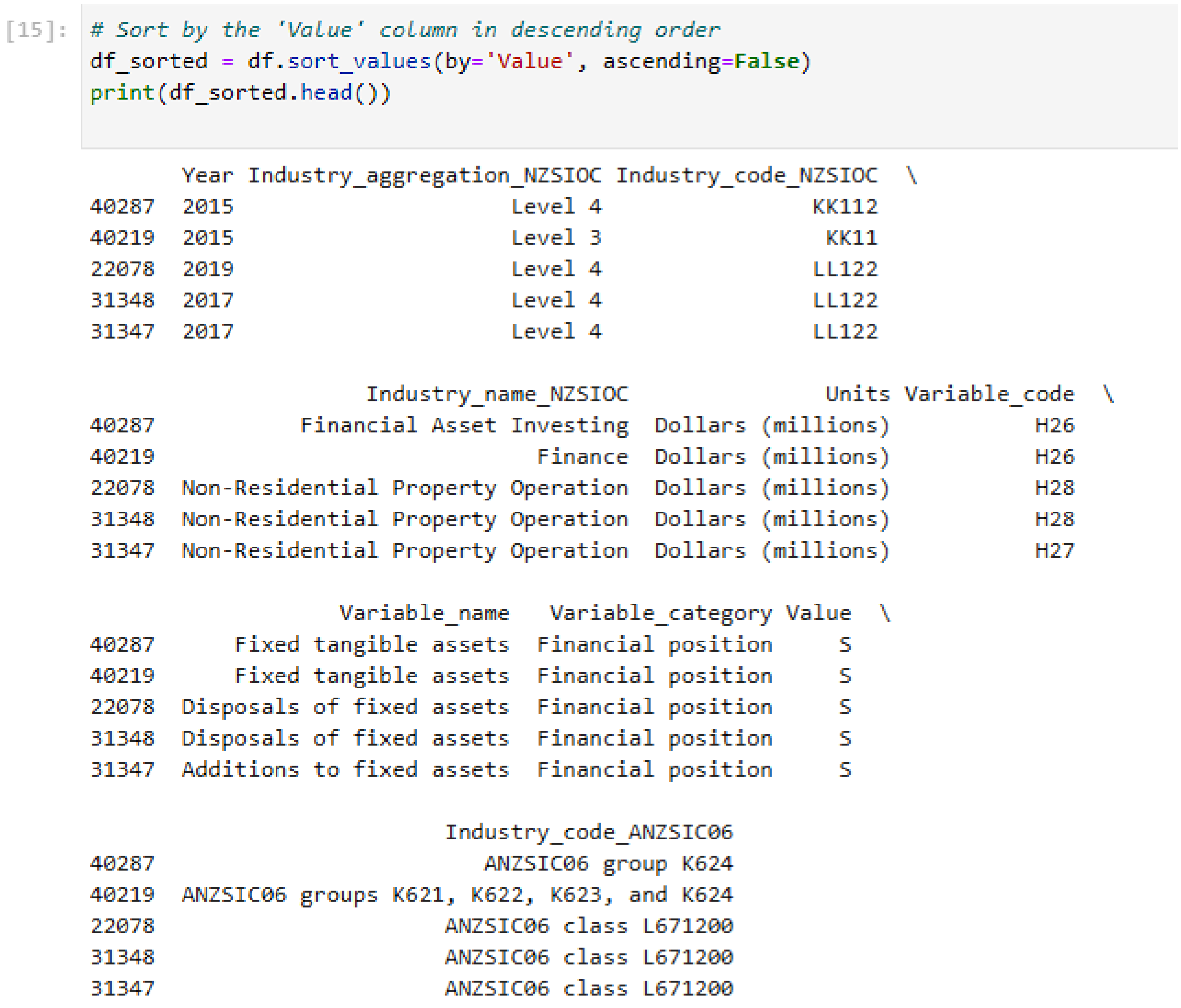
```
[14]: # Check for missing data
      print(df.isnull().sum())

      # Example: Fill missing values with a specific value (e.g., 0)
      df.fillna(0, inplace=True)

      # Alternatively, you can drop rows with missing values
      df.dropna(inplace=True)
```

```
Year                        0
Industry_aggregation_NZSIOC 0
Industry_code_NZSIOC        0
Industry_name_NZSIOC        0
Units                       0
Variable_code               0
Variable_name               0
Variable_category           0
Value                       0
Industry_code_ANZSIC06      0
dtype: int64
```

(No Missing Data in this Dataset)

### 6. Sorting the DataFrame:

- **Explanation:** Sorting helps you reorganize the dataset based on one or more columns, such as sorting by 'Value' in ascending or descending order.



[15]:

```
# Sort by the 'Value' column in descending order
df_sorted = df.sort_values(by='Value', ascending=False)
print(df_sorted.head())
```

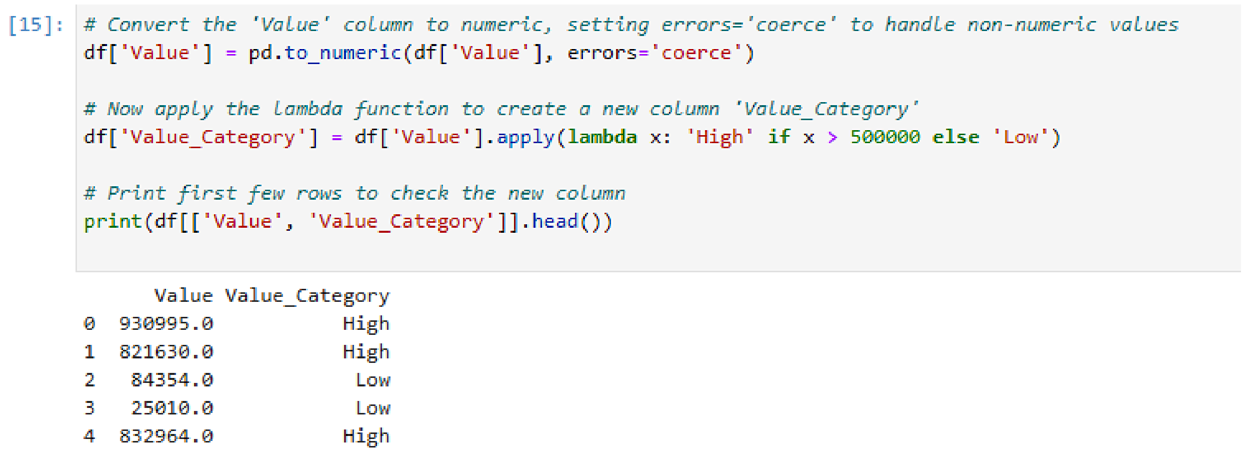
```
       Year Industry_aggregation_NZSIOC Industry_code_NZSIOC  \
40287  2015                     Level 4                KK112   
40219  2015                     Level 3                 KK11   
22078  2019                     Level 4                LL122   
31348  2017                     Level 4                LL122   
31347  2017                     Level 4                LL122   

                     Industry_name_NZSIOC               Units Variable_code  \
40287           Financial Asset Investing  Dollars (millions)           H26   
40219                             Finance  Dollars (millions)           H26   
22078  Non-Residential Property Operation  Dollars (millions)           H28   
31348  Non-Residential Property Operation  Dollars (millions)           H28   
31347  Non-Residential Property Operation  Dollars (millions)           H27   

                   Variable_name   Variable_category Value  \
40287      Fixed tangible assets  Financial position     S   
40219      Fixed tangible assets  Financial position     S   
22078  Disposals of fixed assets  Financial position     S   
31348  Disposals of fixed assets  Financial position     S   
31347  Additions to fixed assets  Financial position     S   

                           Industry_code_ANZSIC06  
40287                         ANZSIC06 group K624  
40219  ANZSIC06 groups K621, K622, K623, and K624  
22078                      ANZSIC06 class L671200  
31348                      ANZSIC06 class L671200  
31347                      ANZSIC06 class L671200  
```

### 9. Using Lambda Functions:

- **Explanation:** Lambda functions allow you to apply quick, anonymous functions without explicitly defining them. Useful for simple operations.

**pd.to_numeric(df['Value'], errors='coerce'):** This function converts the 'Value' column to numeric data. Any invalid string values (like letters or symbols) will be converted to NaN (Not a Number).

**Lambda function:** The lambda function will now correctly compare numeric values in the 'Value' column to 500,000 and categorize them as "High" or "Low."

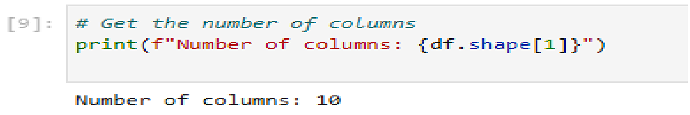


```
[15]: # Convert the 'Value' column to numeric, setting errors='coerce' to handle non-numeric values
      df['Value'] = pd.to_numeric(df['Value'], errors='coerce')

      # Now apply the lambda function to create a new column 'Value_Category'
      df['Value_Category'] = df['Value'].apply(lambda x: 'High' if x > 500000 else 'Low')

      # Print first few rows to check the new column
      print(df[['Value', 'Value_Category']].head())
```

```
      Value Value_Category
0  930995.0           High
1  821630.0           High
2   84354.0            Low
3   25010.0            Low
4  832964.0           High
```

## 11. Number of Columns in the Dataset:

- **Explanation:** The shape of the DataFrame gives you the number of rows and columns. shape[1] returns the number of columns.

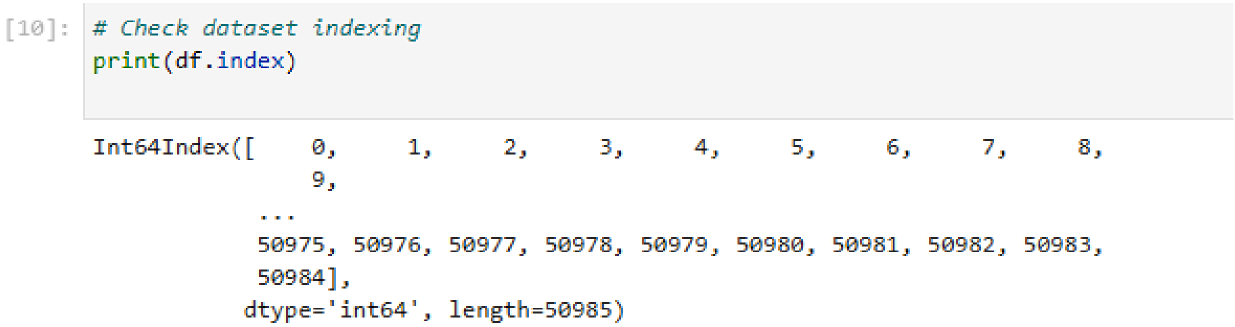


```
[9]: # Get the number of columns
     print(f"Number of columns: {df.shape[1]}")
```

```
Number of columns: 10
```

## 13. How the Dataset is Indexed:

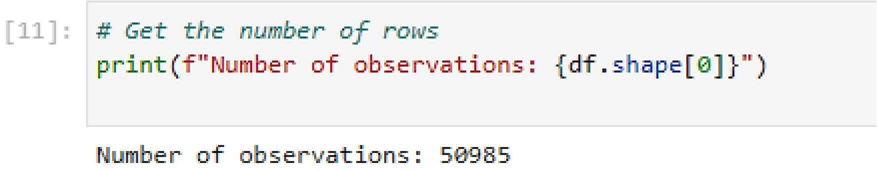
- **Explanation:** DataFrames are indexed (usually by row numbers), but the index can also be custom (e.g., dates). Checking the index tells you how data is organized.



```
[10]: # Check dataset indexing
      print(df.index)

      Int64Index([    0,     1,     2,     3,     4,     5,     6,     7,     8,
                  9,
                  ...
                 50975, 50976, 50977, 50978, 50979, 50980, 50981, 50982, 50983,
                 50984],
                dtype='int64', length=50985)
```

## 14. Number of Observations in the Dataset:

- **Explanation:** The number of rows or "observations" gives you an idea of the dataset's size. This can be checked using shape[0].



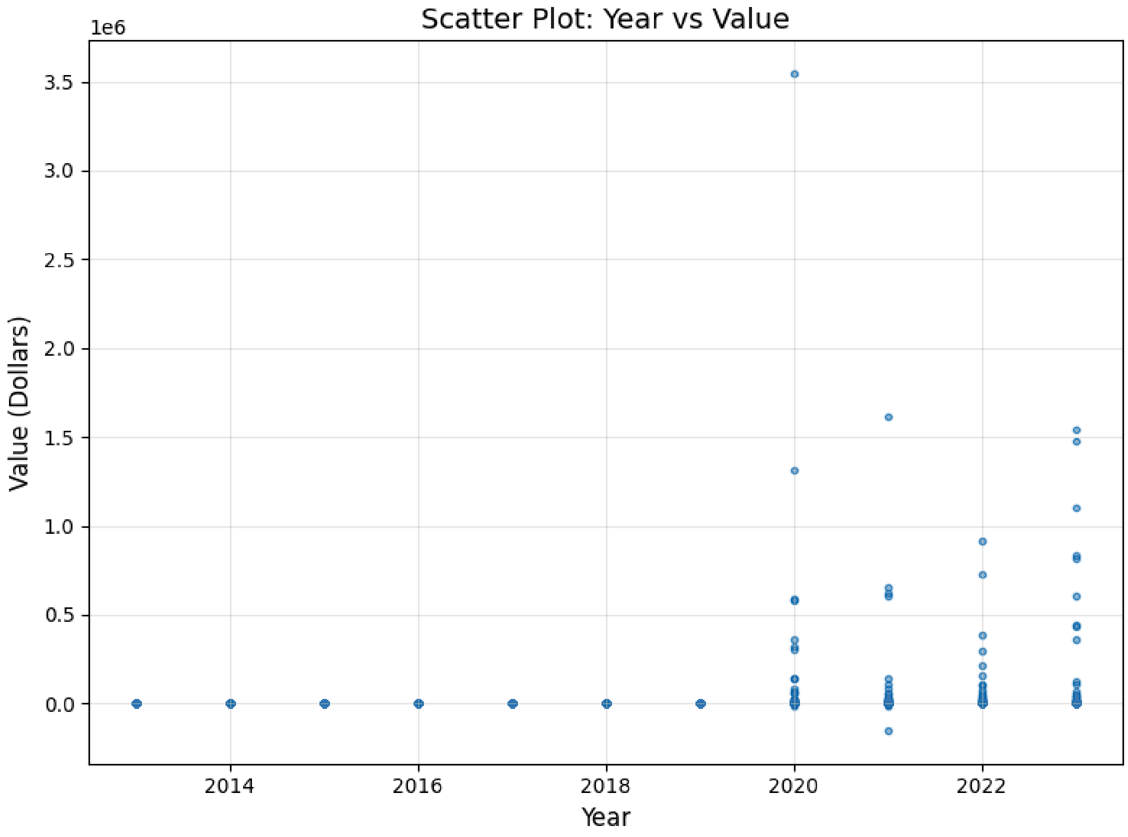
```
[11]: # Get the number of rows
      print(f"Number of observations: {df.shape[0]}")

      Number of observations: 50985
```

## 13. Visualizing DataFrame:

**Explanation:** The code processes a CSV file using Pandas to prepare data for a scatter plot. It reads the file, converts the Value column to numeric (replacing invalid entries with NaN), and drops rows with missing values. It samples 1000 rows randomly. Using Matplotlib, it creates a scatter plot of Year (x-axis) versus Value (y-axis), with transparency and grid lines to improve readability. This approach ensures efficient and clear visualization of large datasets.

[17]:

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

# Load the CSV file
file_path = 'CodingCSV.csv'
df = pd.read_csv(file_path)

# Convert the 'Value' column to numeric, replacing non-numeric entries with NaN
df['Value'] = pd.to_numeric(df['Value'], errors='coerce')

# Drop rows with NaN in 'Value' or other relevant columns
df_clean = df.dropna(subset=['Value'])

# Sample a subset of the data to reduce size for plotting
sampled_df = df_clean.sample(n=1000, random_state=42)

# Scatter plot: Year vs. Value
plt.figure(figsize=(8, 6))
plt.scatter(sampled_df['Year'], sampled_df['Value'], alpha=0.6, s=10)
plt.title('Scatter Plot: Year vs Value', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Value (Dollars)', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```


Scatter Plot: Year vs Value
1e6
Value (Dollars)
Year