PYTHON CODING CHALLENGE

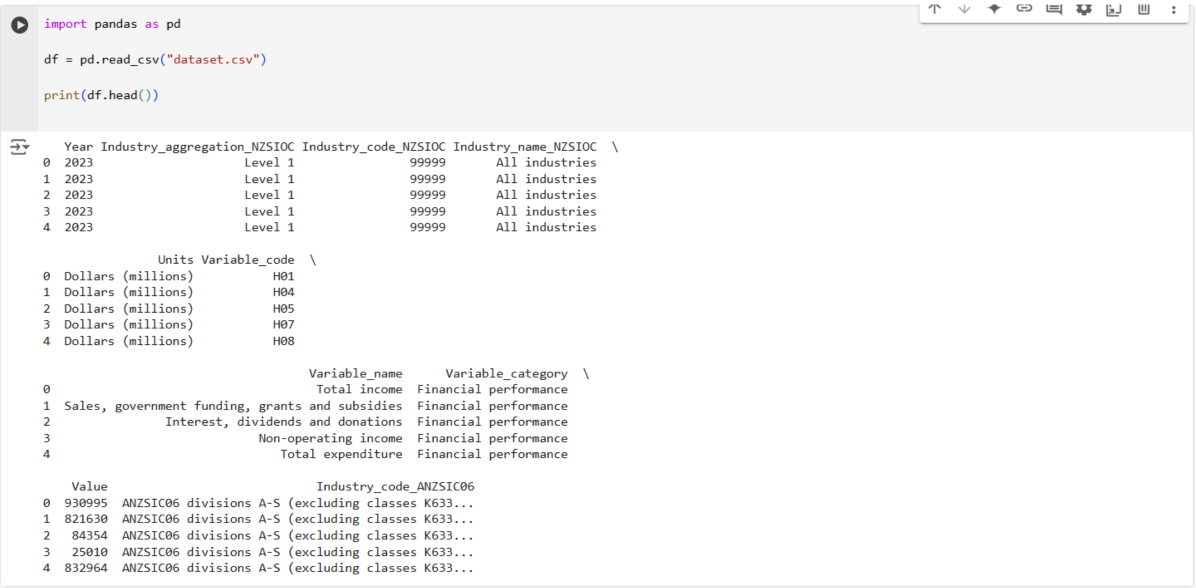
NAME : TEJASWINI GOKANAKONDA
ROLL No. : DE142
DATE : 15 - 11 -2024

Q1 Printing rows of the Data

Code

```python
import pandas as pd

df = pd.read_csv("dataset.csv")

print(df.head())
```

Output



```python
import pandas as pd

df = pd.read_csv("dataset.csv")

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...  
```

Explanation:

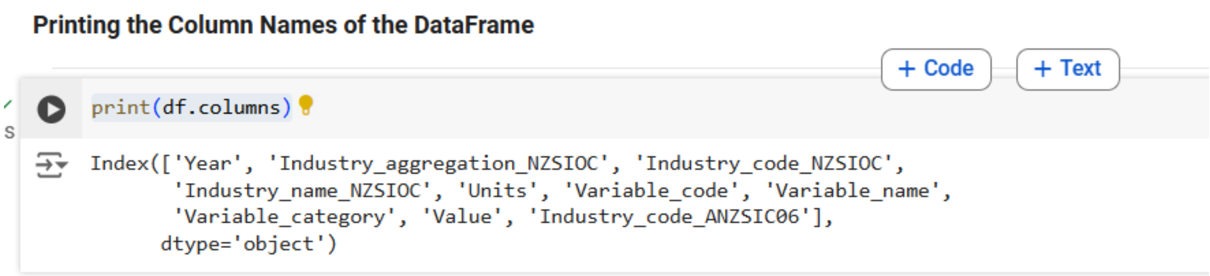
1. The **head()** function displays the first five rows by default.
2. This allows a quick look at the data structure and the contents of the dataset.
3. We can pass an integer n to view the first n rows as needed.

Q2 Printing the column names of the DataFrame

Code

```
print(df.columns)
```

Output



Printing the Column Names of the DataFrame

```
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')
```

Explanation:

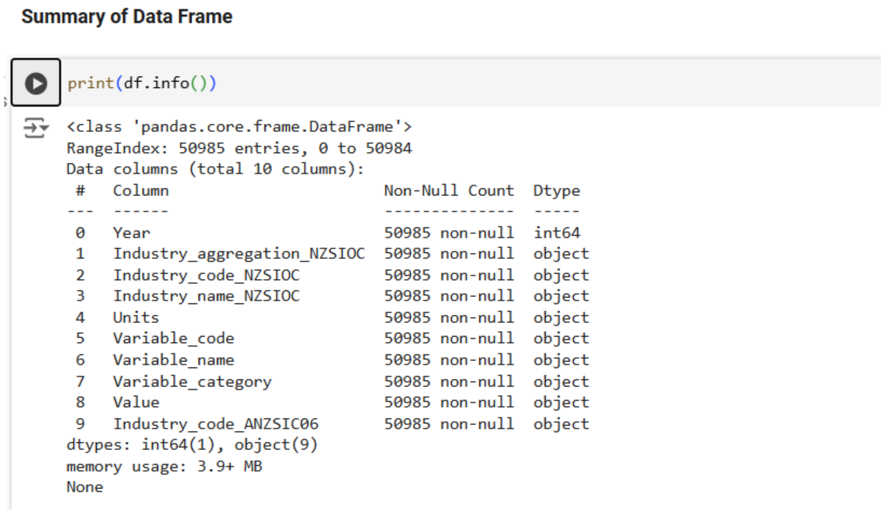
- **df.columns** returns an Index object containing column names.
- We can view and verify the correct loading of columns.
- This is especially useful for understanding the available fields.

Q3 Summary of Data Frame

Code

```
print(df.info())
```

Output



Summary of Data Frame

```
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
```

Explanation:

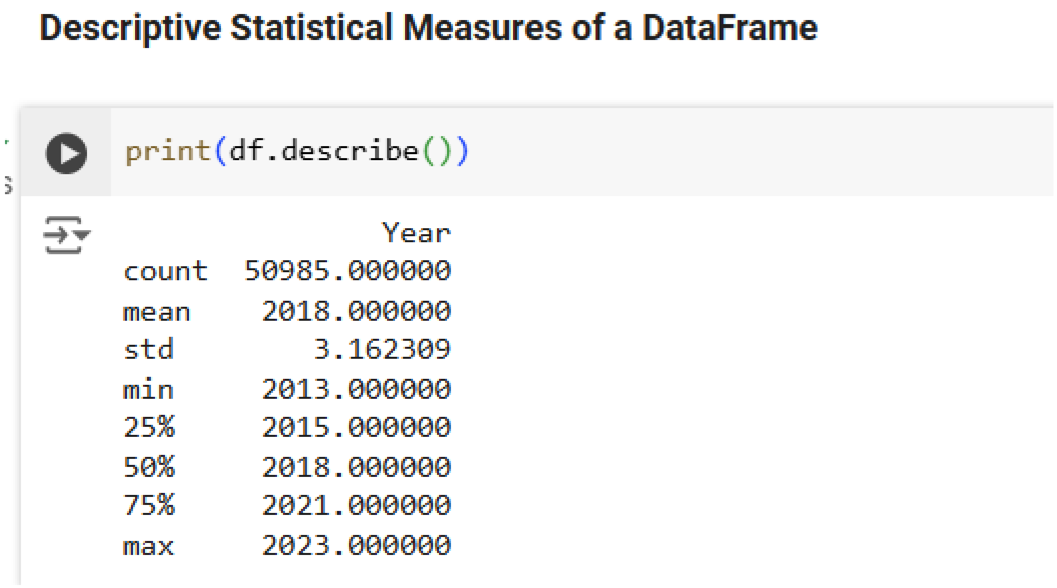
- `df.info()` provides information about data types and null values for each column.
- It's useful for assessing memory usage and data completeness.
- Essential for understanding the data format (numeric, object, etc.) and preparing for further analysis.

Q4 Descriptive Statistical Measures of a DataFrame

Code

```
print(df.describe())
```

Output



Descriptive Statistical Measures of a DataFrame

```
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
```

Explanation:

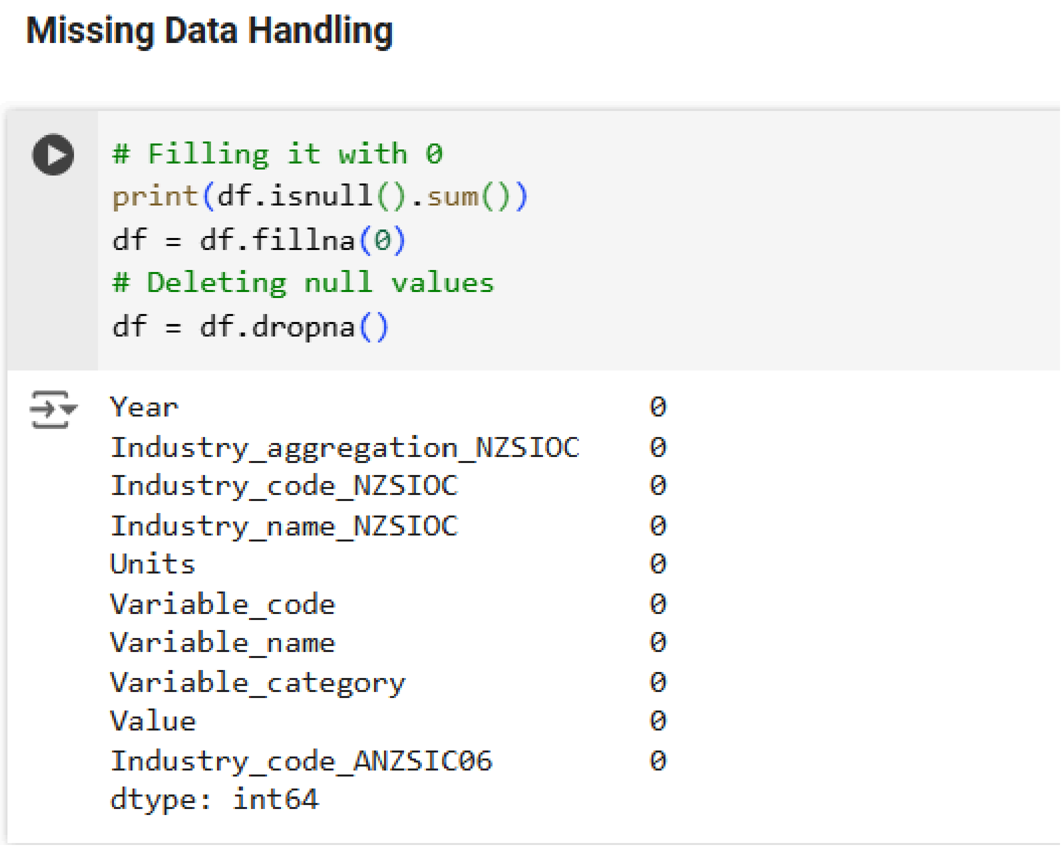
- **`describe()` calculates key statistical metrics (mean, std, min, etc.) for each numerical column.**
- **Helps in understanding data distribution, outliers, and central tendency.**
- **Useful for identifying any potential anomalies.**

Q5 Missing Data Handing

Code

```
# Filling it with 0

print(df.isnull().sum())

df = df.fillna(0)

# Deleting null values

df = df.dropna()
```

Output



Missing Data Handling

```
# Filling it with 0
print(df.isnull().sum())
df = df.fillna(0)
# Deleting null values
df = df.dropna()
```

```
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
```

Explanation:

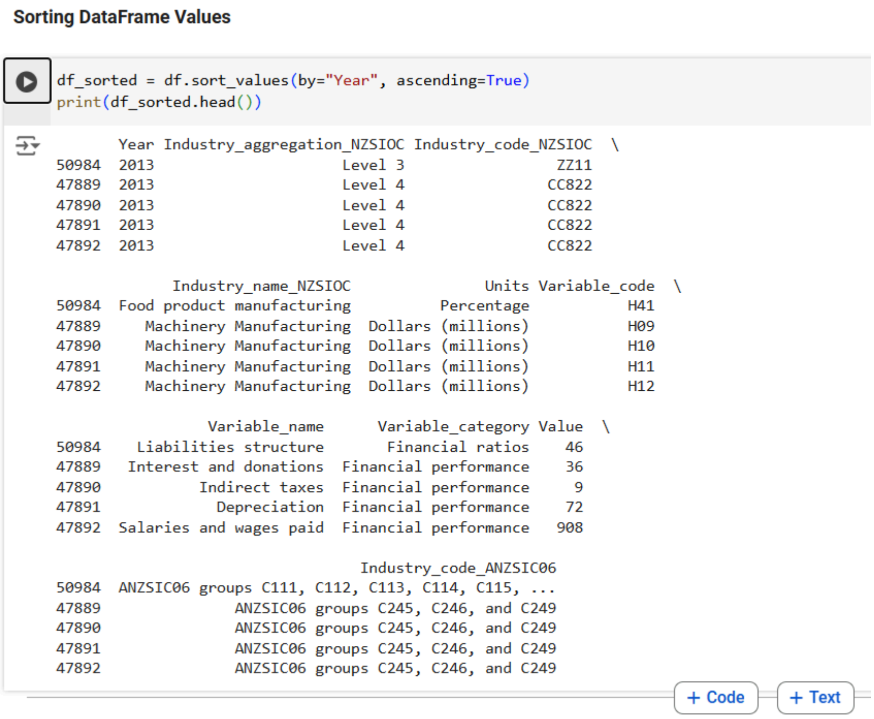
- **isnull().sum()** gives a count of missing values per column.
- We can handle missing data by using **fillna()** to replace NaNs or **dropna()** to remove rows with missing values.
- This ensures that the data is complete for analysis or avoids errors during computation.

Q6 Sorting DataFrame values

Code

```
df_sorted = df.sort_values(by="Year", ascending=True)

print(df_sorted.head())
```

Output



Sorting DataFrame Values

```
df_sorted = df.sort_values(by="Year", ascending=True)
print(df_sorted.head())
```

```
       Year Industry_aggregation_NZSIOC Industry_code_NZSIOC  \
50984  2013                     Level 3                 ZZ11
47889  2013                     Level 4                CC822
47890  2013                     Level 4                CC822
47891  2013                     Level 4                CC822
47892  2013                     Level 4                CC822

             Industry_name_NZSIOC               Units Variable_code  \
50984  Food product manufacturing          Percentage           H41
47889     Machinery Manufacturing  Dollars (millions)           H09
47890     Machinery Manufacturing  Dollars (millions)           H10
47891     Machinery Manufacturing  Dollars (millions)           H11
47892     Machinery Manufacturing  Dollars (millions)           H12

                   Variable_name      Variable_category Value  \
50984      Liabilities structure       Financial ratios    46
47889     Interest and donations  Financial performance    36
47890             Indirect taxes  Financial performance     9
47891               Depreciation  Financial performance    72
47892    Salaries and wages paid  Financial performance   908

                               Industry_code_ANZSIC06
50984  ANZSIC06 groups C111, C112, C113, C114, C115, ...
47889               ANZSIC06 groups C245, C246, and C249
47890               ANZSIC06 groups C245, C246, and C249
47891               ANZSIC06 groups C245, C246, and C249
47892               ANZSIC06 groups C245, C246, and C249
```

Explanation:

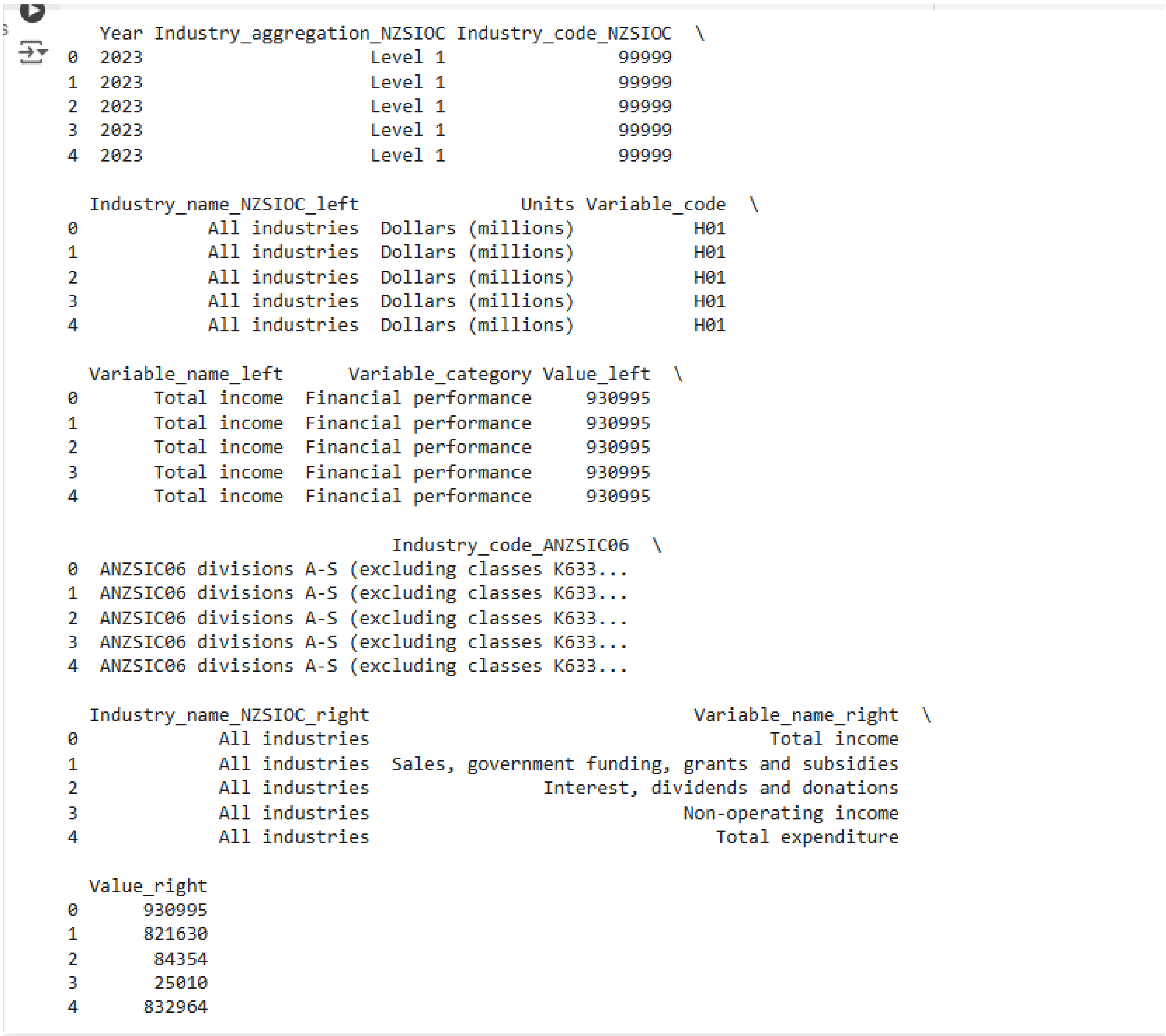
- `sort_values()` sorts the DataFrame by a specified column (e.g., Year).
- Sorting data helps in organizing it for easier visualization and analysis.
- `ascending=True` sorts in ascending order, but we can also set it to `False` for descending.

Q7 Merge Data Frames

Code

```
df1 = df[['Industry_code_ANZSIC06', 'Industry_name_NZSIOC',
'Variable_name', 'Value']]

merged_df = pd.merge(df, df1, on="Industry_code_ANZSIC06",
suffixes=('_left', '_right'))

print(merged_df.head())
```

Output



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

  Industry_name_NZSIOC_left               Units Variable_code  \
0            All industries  Dollars (millions)           H01   
1            All industries  Dollars (millions)           H01   
2            All industries  Dollars (millions)           H01   
3            All industries  Dollars (millions)           H01   
4            All industries  Dollars (millions)           H01   

  Variable_name_left      Variable_category Value_left  \
0       Total income  Financial performance     930995   
1       Total income  Financial performance     930995   
2       Total income  Financial performance     930995   
3       Total income  Financial performance     930995   
4       Total income  Financial performance     930995   

                              Industry_code_ANZSIC06  \
0  ANZSIC06 divisions A-S (excluding classes K633...   
1  ANZSIC06 divisions A-S (excluding classes K633...   
2  ANZSIC06 divisions A-S (excluding classes K633...   
3  ANZSIC06 divisions A-S (excluding classes K633...   
4  ANZSIC06 divisions A-S (excluding classes K633...   

  Industry_name_NZSIOC_right                              Variable_name_right  \
0             All industries                                     Total income   
1             All industries  Sales, government funding, grants and subsidies   
2             All industries                 Interest, dividends and donations   
3             All industries                             Non-operating income   
4             All industries                                Total expenditure   

  Value_right  
0      930995  
1      821630  
2       84354  
3       25010  
4      832964  
```

Explanation:

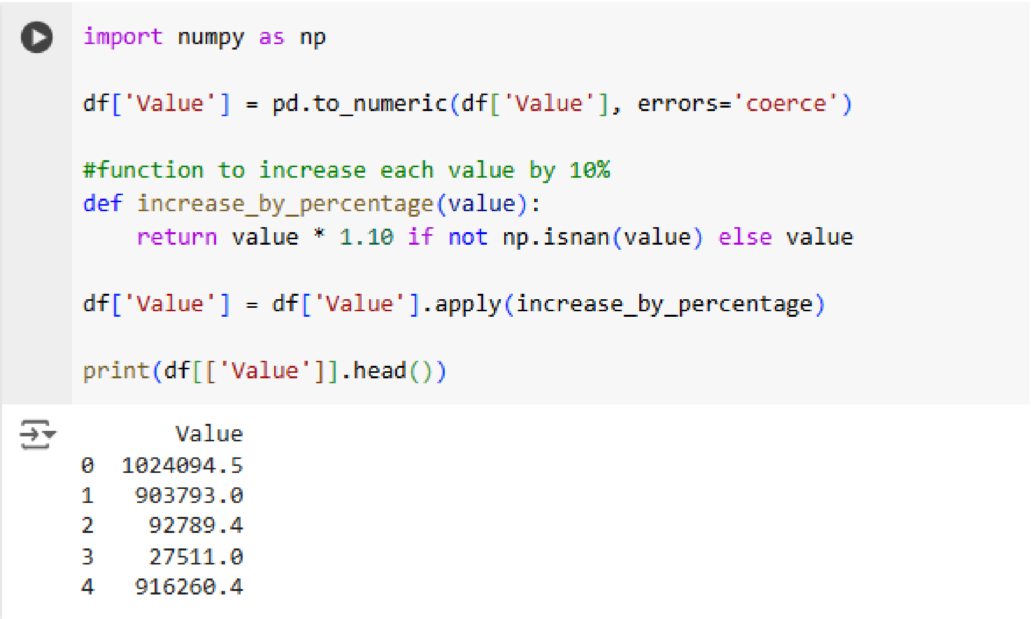
1. Selecting Subset Columns: `df1` is created by selecting columns that could be relevant to merge with the main DataFrame. Here, columns like `Industry_code_ANZSIC06` and `Value` are selected, with the goal of adding them back in a merged DataFrame to check for consistency or changes.
2. Merge Operation: We use `pd.merge()` on the column `Industry_code_ANZSIC06` to combine `df` with `df1`. The `suffixes` parameter helps differentiate similarly named columns.
3. Display Merged Data: `merged_df.head()` will show the first few rows of the merged DataFrame for verification.

Q8 Apply Function

Code

```
import numpy as np

df['Value'] = pd.to_numeric(df['Value'], errors='coerce')

#function to increase each value by 10%

def increase_by_percentage(value):

    return value * 1.10 if not np.isnan(value) else value

df['Value'] = df['Value'].apply(increase_by_percentage)

print(df[['Value']].head())
```

Output



```
import numpy as np

df['Value'] = pd.to_numeric(df['Value'], errors='coerce')

#function to increase each value by 10%
def increase_by_percentage(value):
    return value * 1.10 if not np.isnan(value) else value

df['Value'] = df['Value'].apply(increase_by_percentage)

print(df[['Value']].head())
```

```
       Value
0  1024094.5
1   903793.0
2    92789.4
3    27511.0
4   916260.4
```

Explanation:

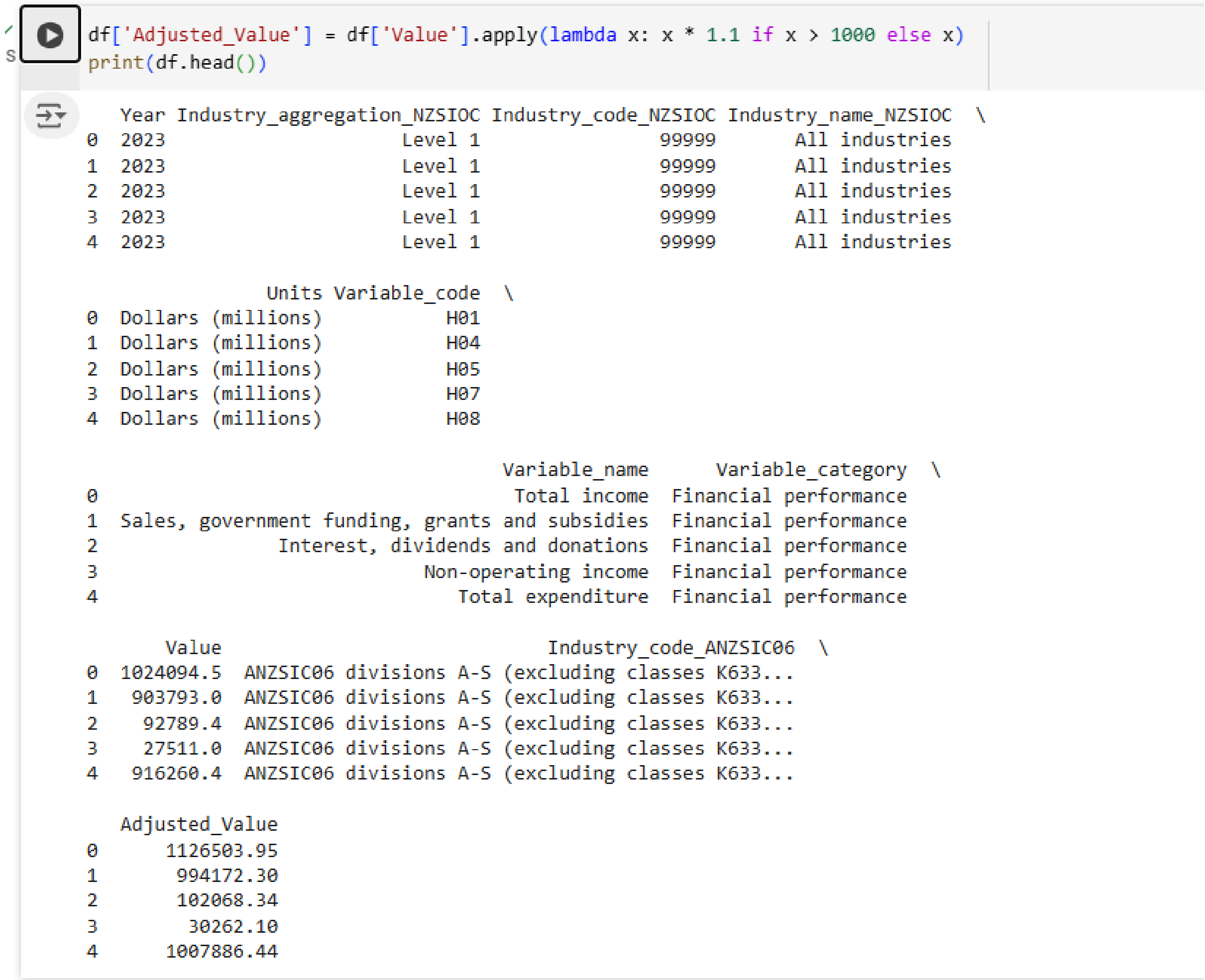
1. We used **NumPy** (`numpy`) to help handle non-numeric values more safely when applying transformations to a DataFrame column in pandas.
2. Converting to Numeric: `pd.to_numeric(df['Value'], errors='coerce')` converts the `Value` column to a numeric type. Non-numeric entries are set to `NaN`.
3. Handling NaN Values: `increase_by_percentage` applies the 10% increase only if the value is not `NaN`.
4. Error Prevention: This approach avoids errors by ensuring the function only attempts multiplication on numeric values.

Q9 By using the lambda operator

Code

```
df['Adjusted_Value'] = df['Value'].apply(lambda x: x * 1.1 if
x > 1000 else x)

print(df.head())
```

Output



```
df['Adjusted_Value'] = df['Value'].apply(lambda x: x * 1.1 if x > 1000 else x)
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  1024094.5  ANZSIC06 divisions A-S (excluding classes K633...
1   903793.0  ANZSIC06 divisions A-S (excluding classes K633...
2    92789.4  ANZSIC06 divisions A-S (excluding classes K633...
3    27511.0  ANZSIC06 divisions A-S (excluding classes K633...
4   916260.4  ANZSIC06 divisions A-S (excluding classes K633...

   Adjusted_Value
0      1126503.95
1       994172.30
2       102068.34
3        30262.10
4      1007886.44
```

Explanation:

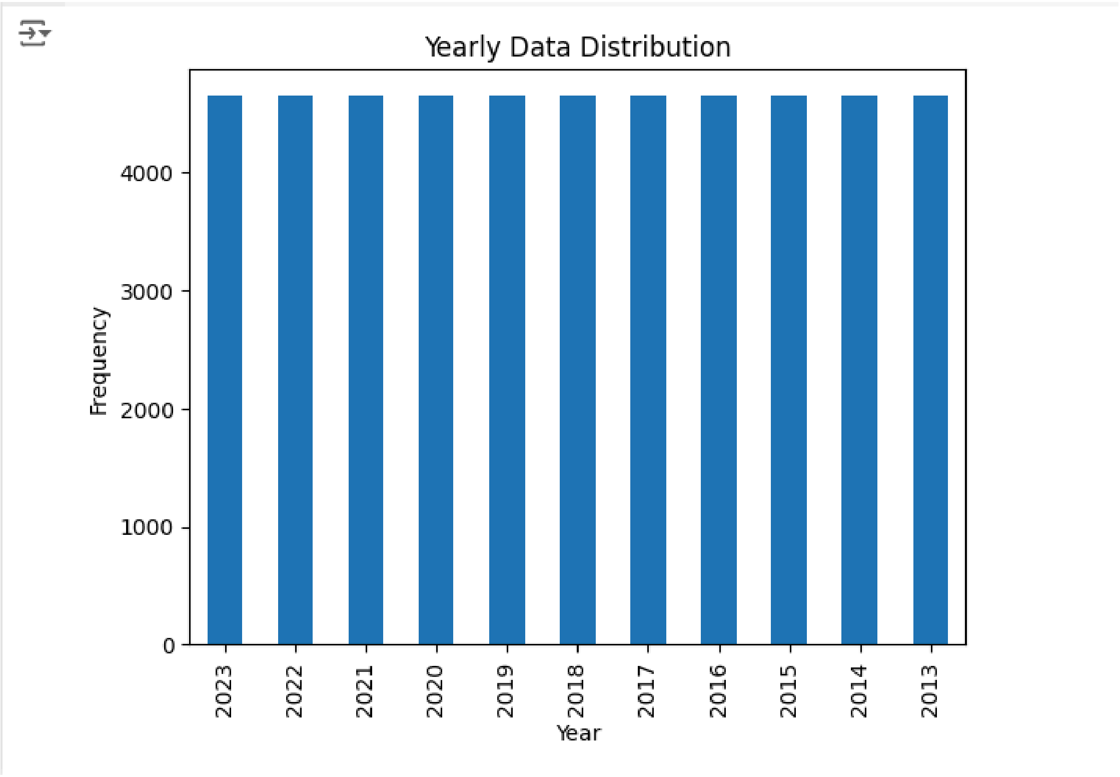
- Lambda functions provide a concise way to apply custom operations.
- Here, a lambda is used to apply a conditional transformation.
- Ideal for quick modifications without defining a separate function.

Q10 Visualizing DataFrame

Code

```
import matplotlib.pyplot as plt

df['Year'].value_counts().plot(kind='bar')

plt.xlabel("Year")

plt.ylabel("Frequency")

plt.title("Yearly Data Distribution")

plt.show()
```

Output



Explanation :

- Data Aggregation: `df['Year'].value_counts()` counts occurrences of each year in the `Year` column, grouping data by year.
- Bar Plot: `.plot(kind='bar')` generates a bar chart to visually represent these frequencies.

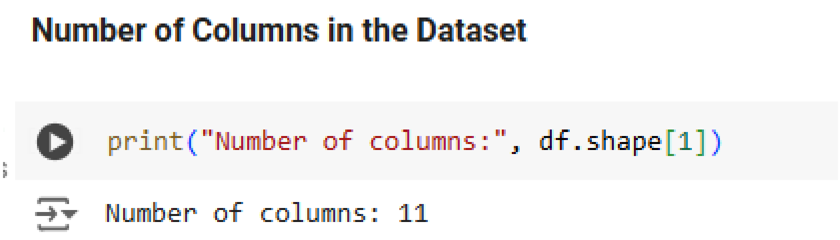
- Labels and Title: **plt.xlabel**, **plt.ylabel**, and **plt.title** add descriptive labels to the x-axis, y-axis, and chart title to clarify the data being shown.
- Display: **plt.show()** renders the chart.

Q11 What is the number of columns in the dataset?

Code

```
print("Number of columns:", df.shape[1])
```

Output



Number of Columns in the Dataset

```
print("Number of columns:", df.shape[1])
```

```
Number of columns: 11
```

Explanation:

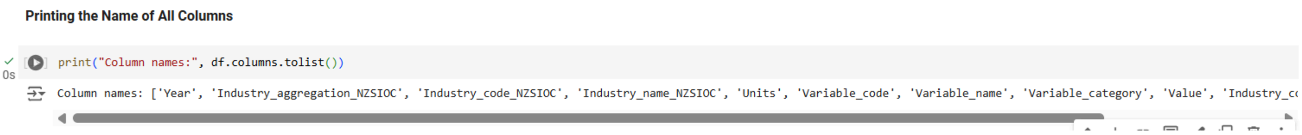
- **df.shape[1]** provides the count of columns in the DataFrame.
- It helps in quickly assessing the DataFrame's dimensionality.
- Useful for verifying dataset structure against expectations.

Q12 print the name of all the columns.

Code

```
print("Column names:", df.columns.tolist())
```

Output



Printing the Name of All Columns

```
print("Column names:", df.columns.tolist())
```

```
Column names: ['Year', 'Industry_aggregation_NZSIOC', 'Industry_code_NZSIOC', 'Industry_name_NZSIOC', 'Units', 'Variable_code', 'Variable_name', 'Variable_category', 'Value', 'Industry_c
```

Explanation:

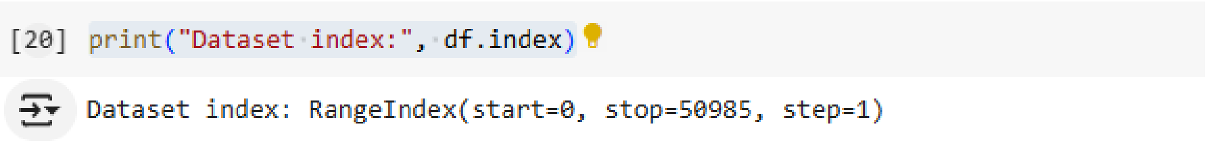
- **df.columns.tolist()** lists all column names in the DataFrame.
- Ensures all expected columns are present and correctly named.
- Useful for further analysis or feature selection.

Q13 How is the dataset indexed?

Code

```
print("Dataset index:", df.index)
```

Output



```
[20] print("Dataset index:", df.index)
Dataset index: RangeIndex(start=0, stop=50985, step=1)
```

Explanation:

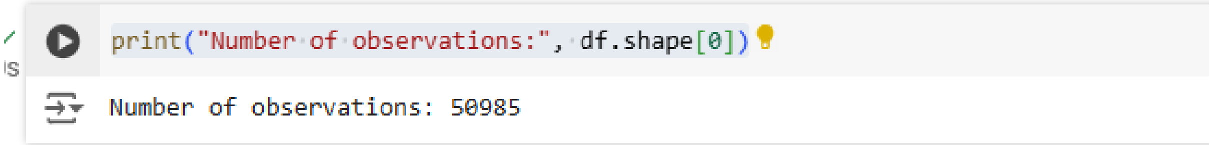
- **df.index** provides information on the DataFrame index.
- Knowing the index type (RangeIndex, etc.) aids in understanding data access patterns.
- Useful for aligning data or troubleshooting mismatches.

Q14 What is the number of observations in the dataset?

Code

```
print("Number of observations:", df.shape[0])
```

Output



```
print("Number of observations:", df.shape[0])
Number of observations: 50985
```

Explanation:

- **df.shape[0]** gives the row count or number of observations.
- This count is essential for sample size assessment and analysis.
- Ensures data adequacy for statistical and machine learning purposes.