Python Coding Challenge 2

Dataset: **annual-enterprise-survey-2023-financial-year-provisional**

Name: **Harish Er**

[1]:
```
# Importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# Load the dataset
file_path = "C:/Users/91776/OneDrive/Desktop/Hexaware/Data Engineering/Coding Challenge 2/annual-enterprise-survey-
data = pd.read_csv(file_path)
```

1. Printing rows of the Data:

```
[3]: # 1: Printing rows of the data
     print("Question 1: First few rows of the data:")
     print(data.head())
```

Output:

```
Question 1: First few rows of the data:
   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: Preview the first few rows of the dataset using the .head() method.

2. **Printing the column names of the DataFrame:**

```
[4]: # 2: Printing column names of the DataFrame
print("\nQuestion 2: Column names of the DataFrame:")
print(data.columns.tolist())
```

Output:

```
Question 2: Column names of the DataFrame:
['Year', 'Industry_aggregation_NZSIOC', 'Industry_code_NZSIOC', 'Industry_name_NZSIOC', 'Units', 'Variable_code', 'Variable_name', 'Var
iable_category', 'Value', 'Industry_code_ANZSIC06']
```

Explanation: Retrieve the column names with the .columns attribute.

3. **Summary of Data Frame:**

```
[5]: # 3: Summary of DataFrame
print("\nQuestion 3: Summary of the DataFrame:")
print(data.info())
```

Output:

```
Question 3: Summary of the DataFrame:
<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: The .info() method gives an overview of the dataset, including data types and counts of non-null values.

4. **Descriptive Statistical Measures of a DataFrame:**

```
[6]: # 4: Descriptive Statistical Measures of a DataFrame
print("\nQuestion 4: Descriptive Statistics:")
print(data.describe())
```

Output:

```
Question 4: Descriptive Statistics:
                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: The .describe() method calculates statistics like mean, standard deviation, and percentiles for numerical columns.

5. Missing Data Handing:

```
[7]: # 5: Missing Data Handling
print("\nQuestion 5: Missing Data:")
missing_data = data.isnull().sum()
print(missing_data)
```

Output:

```
Question 5: Missing Data:
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
```

```
[8]: # Fill missing values with the mean for numerical columns
data_filled = data.fillna(data.mean(numeric_only=True))
print("\nMissing values filled with mean (numerical columns):")
print(data_filled.head())
```

```
Missing values filled with mean (numerical columns):
   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: Missing data can be handled by dropping rows or columns, or by filling missing values with default or computed values (like the mean).

6. Sorting DataFrame values:

[9]:
```
# 6: Sorting DataFrame values
# Sorting by the first numerical column
numerical_columns = data.select_dtypes(include='number').columns
if not numerical_columns.empty:
    numerical_column = numerical_columns[0]
    sorted_data = data.sort_values(by=numerical_column)
    print(f"\nStep 6: Data sorted by column '{numerical_column}':")
    print(sorted_data.head())
else:
    print("\nQuestion 6: No numerical columns to sort.")
```

Output:

```
Step 6: Data sorted by column 'Year':
       Year Industry_aggregation_NZSIOC Industry_code_NZSIOC  \
50867  2013                     Level 4                RS211
50868  2013                     Level 4                RS211
50869  2013                     Level 4                RS211
50948  2013                     Level 4                RS214
50949  2013                     Level 3                 ZZ11

                             Industry_name_NZSIOC              Units  \
50867                       Repair and Maintenance  Dollars (millions)
50868                       Repair and Maintenance  Dollars (millions)
50869                       Repair and Maintenance  Dollars (millions)
50948  Civil, Professional and Other Interest Groups          Percentage
50949                   Food product manufacturing  Dollars (millions)

      Variable_code              Variable_name      Variable_category   Value  \
50867           H23  Surplus before income tax  Financial performance     308
50868           H24               Total assets     Financial position   2,015
50869           H25             Current assets     Financial position   1,052
50948           H41      Liabilities structure       Financial ratios      88
50949           H01               Total income  Financial performance  36,411

                                Industry_code_ANZSIC06
50867               ANZSIC06 groups S941, S942, and S949
50868               ANZSIC06 groups S941, S942, and S949
50869               ANZSIC06 groups S941, S942, and S949
50948                                  ANZSIC06 group S955
50949  ANZSIC06 groups C111, C112, C113, C114, C115, ...
```

Explanation: Data can be sorted by specific columns using the .sort_values() method.

7. Merge Data Frames:

```
[10]: # 7: Merging DataFrames
# Creating a second DataFrame for demonstration
df_example = pd.DataFrame({
    'Merge_Key': data.index[:5],  # Example keys from index
    'New_Column': ['A', 'B', 'C', 'D', 'E']
})
data_merged = data.merge(df_example, left_index=True, right_on='Merge_Key', how='left')
print("\nQuestion 7: Merged DataFrame:")
print(data_merged.head())
```

Output:

```
Question 7: Merged DataFrame:
     Year Industry_aggregation_NZSIOC Industry_code_NZSIOC  \
0.0  2023                     Level 1                99999
1.0  2023                     Level 1                99999
2.0  2023                     Level 1                99999
3.0  2023                     Level 1                99999
4.0  2023                     Level 1                99999

    Industry_name_NZSIOC              Units Variable_code  \
0.0       All industries  Dollars (millions)           H01
1.0       All industries  Dollars (millions)           H04
2.0       All industries  Dollars (millions)           H05
3.0       All industries  Dollars (millions)           H07
4.0       All industries  Dollars (millions)           H08

                                    Variable_name      Variable_category  \
0.0                                  Total income  Financial performance
1.0  Sales, government funding, grants and subsidies  Financial performance
2.0                Interest, dividends and donations  Financial performance
3.0                            Non-operating income  Financial performance
4.0                               Total expenditure  Financial performance

      Value                                Industry_code_ANZSIC06  Merge_Key  \
0.0  930995  ANZSIC06 divisions A-S (excluding classes K633...          0
1.0  821630  ANZSIC06 divisions A-S (excluding classes K633...          1
2.0   84354  ANZSIC06 divisions A-S (excluding classes K633...          2
3.0   25010  ANZSIC06 divisions A-S (excluding classes K633...          3
4.0  832964  ANZSIC06 divisions A-S (excluding classes K633...          4

    New_Column
0.0          A
1.0          B
2.0          C
3.0          D
4.0          E
```

Explanation: Created an example of merging two DataFrames based on a common column.

8. Apply Function:

[11]:
```
# 8: Apply Function
# Convert a column to uppercase for demonstration
string_columns = data.select_dtypes(include='object').columns
if not string_columns.empty:
    string_column = string_columns[0]
    data[string_column] = data[string_column].apply(lambda x: x.upper() if isinstance(x, str) else x)
    print(f"\nStep 8: Data after applying function on column '{string_column}':")
    print(data.head())
else:
    print("\nQuestion 8: No string columns for applying a function.")
```

Output:

```
Step 8: Data after applying function on column 'Industry_aggregation_NZSIOC':
   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: The .apply() method allows us to apply a function across rows or columns of the DataFrame.

9. By using the lambda operator:

[12]:
```
# 9: By using the lambda operator
# Adding 10 to the first numerical column
if not numerical_columns.empty:
    data[f"{numerical_column}_plus_10"] = data[numerical_column].apply(lambda x: x + 10)
    print(f"\nStep 9: New column with values increased by 10 from column '{numerical_column}':")
    print(data[[numerical_column, f"{numerical_column}_plus_10"]].head())
else:
    print("\nQuestion 9: No numerical columns to apply lambda.")
```

Output:

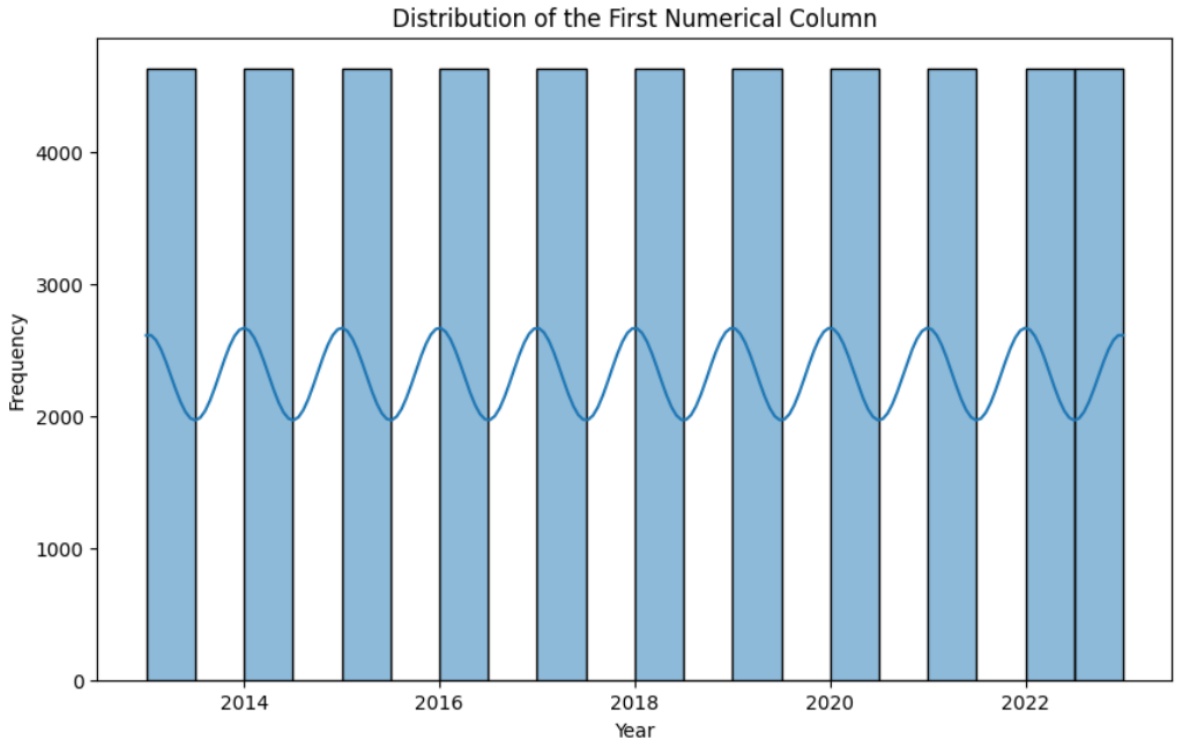
```
Step 9: New column with values increased by 10 from column 'Year':
   Year  Year_plus_10
0  2023          2033
1  2023          2033
2  2023          2033
3  2023          2033
4  2023          2033
```

Explanation: The lambda operator creates short, anonymous functions that can be applied to DataFrame columns.

10.Visualizing DataFrame:

```
[13]: # 10: Visualizing DataFrame
# Bar plot for the first numerical column
if not numerical_columns.empty:
    plt.figure(figsize=(10, 6))
    sns.histplot(data[numerical_column], kde=True, bins=20)
    plt.title('Distribution of the First Numerical Column')
    plt.xlabel(numerical_column)
    plt.ylabel('Frequency')
    plt.show()
else:
    print("\nQuestion 10: No numerical columns for visualization.")
```

Output:



Explanation: Visualization helps us understand trends, distributions, and relationships in the data. We'll use libraries like matplotlib or seaborn.

11.What is the number of columns in the dataset?:

```
[14]: # 11: Find the number of columns in the dataset
num_columns = len(data.columns)
print(f"Question 11: The number of columns in the dataset is: {num_columns}")
```

Output:

```
Question 11: The number of columns in the dataset is: 11
```

Explanation: To determine the number of columns in the dataset, we can use the len() function on the .columns attribute of the DataFrame.