A Simple Task for Pandas-DataFrame:

Dataset under discussion - Sample URL:

https://github.com/ShahzadSarwar10/FULLSTACK-AI-BOOTCAMP-B2-MonTOFri-7TO9-PM-Explorer/blob/main/DataSetForPractice/macro_monthly.csv

It is macro factors – US data.

TASK:

1. Load above CVS file above, into DataFrame variable , with Pandas, following columns
   With auto Index column.
   Print DataFrame.
2. Call following method/properties of DataFrame, print output and analyze the output.
   .info()
   .dtypes
   .describe()
   .shape
   .
3. Explore
   https://www.geeksforgeeks.org/python-pandas-dataframe-to_string/
   Use , DataFrame method - .to_string()
   Use, debug, trace and play with following parameters.

Parameters:

- ✓ buf: Buffer to write the output string to (e.g., a file). Defaults to None, which means the output is returned as a string.
- ✓ columns: Specifies a subset of columns to include in the output. If None, all columns are printed.
- ✓ col_space: Defines the minimum width of each column.
- ✓ header: Whether to print column names. Can also accept a list of column name aliases.
- ✓ index: Whether to include index labels. Default is True.
- ✓ na_rep: String representation for missing values (NaN). Default is 'NaN'.
- ✓ formatters: Dictionary or list of functions to apply to columns for formatting their output.
- ✓ float_format: Formatter function to apply specifically to floating-point numbers.
- ✓ sparsify: Controls hierarchical index formatting. If False, prints every multi-index key at each row.
- ✓ index_names: Whether to print index names. Default is True.
- ✓ justify: Alignment of column headers ('left', 'right', 'center', 'justify' or 'justify-all').
- ✓ max_rows: Maximum number of rows to display. If exceeded, truncates output.
- ✓ max_cols: Maximum number of columns to display. If exceeded, truncates output.
- ✓ show_dimensions: If True, displays the shape (rows x columns) of the DataFrame.
- ✓ decimal: Specifies the character for decimal separation (e.g., ',' for European formatting).
- ✓ line_width: Defines the maximum character width of a row before wrapping text."""

4. On given DataFrame – select top 4 rows, and print – verify, debug, analyze

5. On given DataFrame – select bottom 4 rows, and print – verify, debug, anzlyze
6. On Given DataFrame – access the Name column for “Industrial_Production” and print whole column– verify, debug, analyze

   Then next, access the name column for “Manufacturers_New_Orders: Durable Goods” and print whole column
7. On Given DataFrame – access access multiple columns like “Industrial_Production” and “Manufacturers_New_Orders: Durable Goods”
   Print it- – verify, debug, analyze

8. Selecting a single row using .loc
   With auto index value 3 , print the returned row and analyze results- – verify, debug, analyze

9. Selecting multiple rows using .loc
   With auto index value 3 , 5, 7 , print the returned rows and analyze results.

10. Selecting a slice of rows using .loc
    With auto index value range of 5 to 15 , print the returned row and analyze results.

11. Conditional selection of rows using .loc
    “Year” equal “1993” or “1994” or “1997” And “Unemployment_Rate” not less than 1
    , print the returned row and analyze results.

12. Selecting a single column using .loc– auto index column value 9 , only select following columns “Industrial_Production”,”Retail_Sales” , ” Manufacturers_New_Orders: Durable Goods” , ”Personal_Consumption_Expenditures”
    , print the returned row and analyze results.

13. Selecting a slice of columns using .loc
    Form “**Industrial_Production**” <= 0.5
    , print the returned row and analyze results.

14. Combined row and column selection using .loc
    “**Industrial_Production**” <= 0.5
    and Columns “Consumer_Price Index” > 0.2
    , print the returned row and analyze results.

15. Selecting a single row using .iloc
    Select 4th row
    , print the returned row and analyze results.

16. Selecting multiple rows using .iloc
    Select – 2th row, 7th row, 8th row, 36th row, and 9th row

, print the returned row and analyze results.

17. Selecting a slice of rows using .iloc
    Select from 10th to 23th row
    , print the returned row and analyze results.

18. Selecting a single column using .iloc
    Select 5th column
    , print the returned row and analyze results.

19. Selecting multiple columns using .iloc
    Select 2nd column, 3th column, 8th columns
    , print the returned row and analyze results.

20. Selecting a slice of columns using .iloc
    Range: Select from 2nd column to 8th columns
    , print the returned row and analyze results.

21. Combined row and column selection using .iloc
    Select – 4th row, 5th row. 7th row, and 25th row
    Select 3rd column, 5th column , 7th column
    , print the returned row and analyze results.

22. Combined row and column selection using .iloc
    Select range : 3nd row, 34th row
    Select range : 3rd column to 6th column
    , print the returned row and analyze results.

23. Add a New Row to a Pandas DataFrame

    print the returned dataFrame and analyze results.

24. delete row with index 4
    print the returned dataFrame and analyze results.

25. delete row with index from 5 to 9th row
    print the returned dataFrame and analyze results.

26. Delete “All_Employees” column

    print the returned dataFrame and analyze results.

27. Delete “Personal_Consumption_Expenditures” and “National_Home_Price_Index” columns
    print the returned dataFrame and analyze results.

28. Rename column “Personal_Consumption_Expenditures” to “Personal_Consumption_Expenditures _Changed”

    Print the returned dataFrame and analyze results.

29. Rename label from 5 to 8
    Print the returned dataFrame and analyze results.

30. query() to Select Data
    where: “**Industrial_Production**” <= 0.5
    and Columns “Consumer_Price Index” > 0.2
    and “Year” = “1992”
    Print the returned dataFrame and analyze results.

31. sort DataFrame by “Consumer_Price Index” in ascending order

32. “group the DataFrame by the “Year” column and calculate the sum of “National_Home_Price_Index” for each category
    Print the returned dataFrame and analyze results.
33. use dropna() to remove rows with any missing values
    Print the returned dataFrame and analyze results.

34. filling NaN values with 0

Reference code: https://github.com/ShahzadSarwar10/FULLSTACK-AI-BOOTCAMP-B2-MonTOFri-7TO9-PM-Explorer/blob/main/Week2/Case2-2-Pandas-Zameencom-property-data-By-Kaggle.py

Ask questions, if you have confusions. ASK me, Call me on whatsapp.

Let’s put best efforts.

Thanks