A Simple Task for Pandas-DataFrame:

Dataset under discussion - Sample URL:

https://github.com/ShahzadSarwar10/AI-ML-Explorer/blob/main/USOpen-DataSet/Real_Estate_Sales_2001-2022_GL-Short.csv

It is REAL ESTATE – US data.

TASK:

1. Load above CVS file above, into DataFrame variable , with Pandas, following columns
   With "Serial Number" as 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 paramters.

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 7 rows, and print

5. On given DataFrame – select bottom 9 rows, and print
6. On Given DataFrame – access the Name column for “Town” and print whole column
   Then next, access the name column for “Residential Type” and print whole column
7. On Given DataFrame – access access multiple columns like “Town” and “Date Recorded”
   Print it.

   Reference:

   https://www.datacamp.com/tutorial/loc-vs-iloc

8. Selecting a single row using .loc
   With index – “Serial Number” value “200008” , print the returned row and analyze results.

9. Selecting multiple rows using .loc
   With index – “Serial Number” value “200305” and “200207”,”20000048” , print the returned rows and analyze results.

10. Selecting a slice of rows using .loc
    With index – “Serial Number” value range of “2020090” and “200121” , print the returned row and analyze results.

11. Conditional selection of rows using .loc
    “Sale Amount” greater then “202500”
    , print the returned row and analyze results.

12. Conditional selection of rows using .loc
    “Town” equal to “Ansonia”
    , print the returned row and analyze results.

13. Conditional selection of rows using .loc
    “Residential Type” equal to “Condo” and “Assessed Value” is equal to - less then 180500
    , print the returned row and analyze results.

14. Selecting a single column using .loc
    With index – “Serial Number” value “201354” , only select following columns
    “Address”,” Assessed Value” , ”Sale Amount” , ”Sales Ratio” , ”Property Type”
    , print the returned row and analyze results.

15. Selecting a slice of columns using .loc
    Form “Date Recorded” to “Sale Amount”
    , print the returned row and analyze results.

16. Combined row and column selection using .loc
    “Residential Type” equal to “Condo” and Columns th “Date Recorded” to “Assessed Value”

, print the returned row and analyze results.

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

18. Selecting multiple rows using .iloc
    Select – 7th row, 9th row and 15th row
    , print the returned row and analyze results.

19. Selecting a slice of rows using .iloc
    Select from 5th to 13th row
    , print the returned row and analyze results.

20. Selecting a single column using .iloc
    Select 3rd column
    , print the returned row and analyze results.

21. Selecting multiple columns using .iloc
    Select 2nd column, 4th column, 7th columns
    , print the returned row and analyze results.

22. Selecting a slice of columns using .iloc
    Range: Select from 2nd column to 5th columns
    , print the returned row and analyze results.

23. Combined row and column selection using .iloc
    Select – 7th row, 9th row and 15th row
    Select 2nd column, 4th column
    , print the returned row and analyze results.

24. Combined row and column selection using .iloc
    Select range : 2nd row, 6th row
    Select range : 2nd column to 4th column
    , print the returned row and analyze results.

25. Add a New Row to a Pandas DataFrame

    print the returned dataFrame and analyze results.

26. delete row with index 2
    print the returned dataFrame and analyze results.

27. delete row with index from 4 to 7th row

print the returned dataFrame and analyze results.

28. Delete “Residential Type” column

    print the returned dataFrame and analyze results.

29. Delete “Assessor Remarks” and “Location” columns
    print the returned dataFrame and analyze results.

30. Rename column “List Year: to “List_Year_Changed”

    Print the returned dataFrame and analyze results.

31. Rename label from “200400” to “20040333”
    Print the returned dataFrame and analyze results.

32. query() to Select Data
    where: "Assessed Value” < 127400
    “Property Type” = “Commercial”

    “Residential Type” not equal to “Single Family”

    Print the returned dataFrame and analyze results.

33. sort DataFrame by price in ascending order column “Assessed Value”

34. “group the DataFrame by the “Property Type” column and calculate the sum of “Sale Amount” for each category
    Print the returned dataFrame and analyze results.
35. use dropna() to remove rows with any missing values
    Print the returned dataFrame and analyze results.

36. filling NaN values with 0

Reference code: https://github.com/ShahzadSarwar10/AI-ML-Explorer/blob/main/Week4/Case4-17-zameencom-property-data-By-Kaggle.py

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

Let’s put best efforts.

Thanks