Data Management tool WS

Skills

💡 The entire code for the solution should be pushed to the private GitHub repository, and the people CC'd on this email should be invited to the repository. Please share the repository at least 12 hours before the interview. Our expectation is that the candidate spends not more than 4 hours on the task, even if the solution is not fully completed. It should be clearly indicated which parts are not solved. Demonstrate your ability to work with version control systems by splitting the solution into smaller steps and having separate commits/pull requests for each part.

1. Task Overview (For the Candidate)

Task Title: Develop a Basic Python Data Analysis Assistant

Objective: Create an interactive, command-line Python tool that enables users to manage datasets, perform data exploration and analysis, and generate visualizations and reports. This tool will support data scientists and AI practitioners in organizing and easing their data workflows.

---

2. Features and Functionality

Dataset Management:

- **Load Datasets:**
  - Load datasets from CSV files.
  - Assign a user-defined name to each dataset for easy reference.
  - Store datasets in organized folders on the disk.
- **View Datasets:**
  - List all loaded datasets with basic information (e.g., number of rows and columns).

    - Display the first N rows of a selected dataset.
- **Remove Datasets:**
    - Remove datasets from the inventory when they are no longer needed.

Data Exploration and Analysis:

- **Summary Statistics:**
    - Calculate and display basic statistics such as mean, median, mode, and standard deviation for numerical columns.
- **Missing Data Handling:**
    - Identify missing values in the dataset.
- **Data Transformation:**
    - **Filtering:**
        - Allow users to filter data based on conditions.
    - **Data Cleaning:**
        - Provide basic data cleaning functions such as removing duplicates.

Visualization and Reporting:

- **Generate Visualizations and Reports:**
    - Create simple plots (e.g., histograms, bar charts) for numerical or categorical data.
    - Save plots as image files.
    - Compile summaries of analyses performed, including statistics and references to generated plots.
    - Save reports to text or CSV files.

Interactive Command-Line Interface:

- **User Commands:**
    - Implement commands like `load`, `list`, `view`, `analyze`, `visualize`, `clean`, `report`, `help`, and `exit`.
    - Provide clear prompts and instructions for each action.

Workflow Example:

1. **Startup:**
   - Display a welcome message.
   - Show available commands or prompt the user for input.
2. **Dataset Management:**
   - **Load:** User loads a dataset using the `load` command, specifying the file path and a dataset name.
   - **List:** User lists all loaded datasets with the `list` command.
   - **View:** User views dataset contents with the `view` command.
3. **Data Exploration and Analysis:**
   - **Analyze:** User performs summary statistics with the `analyze` command.
   - **Handle Missing Data:** User manages missing values using the `analyze missing` subcommand.
   - **Transform:** User filters or sorts data with the `filter`
   - **Clean:** User removes duplicates with the `clean` command.
4. **Visualization and Reporting:**
   - **Visualize and Report:** User generates and saves plots and compiles reports with the `visualize` command
5. **Exiting the Tool:**
   - User exits the tool with the `exit` command, optionally saving any progress or reports.

---

3. Requirements and Constraints

To ensure the task remains manageable and focused, please adhere to the following constraints:

- **Programming Language:** Python.
- **Object-Oriented Programming:**
  - Utilize OOP principles.
- **Functional Requirements:**

    - Handle at least two different datasets.
    - Implement at least three main functionalities: Dataset Management, Data Exploration & Analysis, and Visualization & Reporting.
- **Data Storage:**
    - **Datastore Structure:**
        - Organize datasets in designated folders on the disk.
        - Each dataset should be a CSV file within its folder.
    - **Metadata Storage:**
        - Store any necessary metadata between runs using local JSON or CSV files.
- **User Interface:**
    - Text-based command-line interface.
    - Provide clear instructions and feedback to the user.
- **Libraries:**
    - Use pure Python for the core functionality.
    - You may incorporate additional libraries such as Pandas for data handling and Matplotlib or Seaborn for visualization.

---

4. Bonus Features (Optional)

These features are not required but can showcase your creativity and additional skills:

- **Advanced Visualizations:**
    - Implement more complex plots like scatter plots, box plots, or heatmaps.
- **Statistical Modeling:**
    - Allow users to perform simple linear regression or clustering.
- **Session Management:**
    - Implement functionality to save and load user sessions.
- **Configuration Files:**

    - Allow the tool to read configurations from a file (e.g., default dataset paths).
- **Automated Testing:**
    - Include a suite of unit tests for your code.
- **Undo operation for missing value handler**

---

5. Sample Input/Output

Here are Example of how the tool might appear when run to help you understand the expected structure.

```
python dataset_manager.py

Welcome to the Python Data Analysis Assistant!
Type 'help' to see available commands.

> help

Available commands:
- load [file_path] [dataset_name]: Load a dataset from a CSV file.
- list: List all loaded datasets.
- view [dataset_name]: View the first few rows of a dataset.
- analyze [dataset_name]: Perform data analysis on a dataset.
- visualize [dataset_name]: Generate visualizations for a dataset.
- clean [dataset_name]: Perform data cleaning operations.
- report [dataset_name]: Generate a report for a dataset.
- help: Show this help message.
- exit: Exit the program.

> load data/sales.csv sales_data

Dataset 'sales_data' loaded successfully.

> list

Loaded datasets:
1. sales_data (Rows: 1000, Columns: 5)
```

```
> view sales_data

Displaying the first 5 rows of 'sales_data':
| Date       | Product | Units Sold | Revenue | Region |
|------------|---------|------------|---------|--------|
| 2021-01-01 | Widget  | 100        | 1000    | North  |
| 2021-01-02 | Gadget  | 150        | 2250    | South  |
| 2021-01-03 | Widget  | 200        | 2000    | East   |
| 2021-01-04 | Gizmo   | 50         | 750     | West   |
| 2021-01-05 | Gadget  | 125        | 1875    | North  |

> analyze sales_data

Select analysis type:
1. Summary statistics
2. Missing data report
3. Frequency counts

Enter your choice (1-3): 1

Summary statistics for 'sales_data':
- Units Sold:
  - Mean: 125
  - Median: 125
  - Std Dev: 50
- Revenue:
  - Mean: $1500
  - Median: $1500
  - Std Dev: $500

> visualize sales_data

Select plot type:
1. Histogram
2. Bar Chart

Enter your choice (1-2): 1
Enter the column name for the histogram: Revenue

Histogram for 'Revenue' generated and saved as 'sales_data_Revenue_hist
ogram.png'.

> clean sales_data

Select cleaning operation:
1. Remove duplicates
2. Handle missing values

Enter your choice (1-2): 2

Missing data handling options:
1. Remove rows with missing values
2. Fill missing values with mean (numerical columns)
3. Fill missing values with mode (categorical columns)

Enter your choice (1-3): 1

Rows with missing values removed. Updated dataset has 990 rows.

> report sales_data

Generating report for 'sales_data'...
Report saved as 'sales_data_report.txt'.

> exit

Thank you for using the Python Data Analysis Assistant. Goodbye!
```