Concise Manual for Data Processing & Excel Workbook Updation Script

Overview

This script is designed to read data from CSV and Excel files, process it, and update certain Excel workbooks with aggregated data. It employs Python with libraries like `pandas` and `openpyxl` to handle and manipulate large datasets.

Software Requirements

- **Python Installation**:
  - Ensure you have `Python` installed on your computer.
  - You can check this by typing `where python` in the Windows Command Prompt (`cmd.exe`).
  - Ensure your Python version is 3.7 or higher.
- **Required Packages**:
  - Install the necessary Python packages if you haven't: `pandas, numpy, ast, openpyxl, jupyter`.
  - Use the command `pip install packageName` in the command prompt to install each package. Replace `packageName` with the name of the package.
- **Choose an IDE**:
  - It's recommended to use an Integrated Development Environment (IDE) for coding.
  - Download and install Microsoft's Visual Studio Code (often called VS Code). Note: This is different from "Visual Studio".
- **VS Code Extensions**:
  - After setting up VS Code, add the 'Python' and 'Jupyter' extensions for a better coding experience with Python and Jupyter Notebooks.

How To Use Jupyter Notebooks

Jupyter Notebooks are interactive computing environments that enable users to create and share documents that contain live code, equations, visualizations, and narrative text. Here's a brief guide to use them:

1. **Launch Jupyter Notebook**: Open the command prompt or terminal and type 'jupyter notebook' and press Enter. This should open a new tab in your default web browser with the Jupyter interface.
2. **Navigate to the Desired Directory**: Use the Jupyter interface to navigate to the directory where your notebook (with an '.ipynb' extension) is located.
3. **Open an Existing Notebook or Create a New One**: Click on an existing notebook to open it or click on 'New' ¿ 'Python 3' to create a new notebook.

4. **Execute a Cell**: In the notebook, you can type Python code into a cell. To run the code in the current cell, press 'Shift + Enter'. This will execute the cell and move to the next one. To run a cell without moving to the next one, press 'Ctrl + Enter'.

5. **Save Your Work**: You can save your notebook by clicking on the floppy disk icon or by going to 'File' ¿ 'Save and Checkpoint'.

6. **Shutdown the Notebook**: Once done, close the browser tab containing the notebook. In the Jupyter interface, you can click on 'Running' to see all currently running notebooks and shut them down if necessary.

7. **Exit Jupyter**: To exit the Jupyter Notebook app, go back to your terminal and press 'Ctrl + C' twice.

Note: For more advanced operations and functionalities, you can explore the Jupyter Notebook documentation or various tutorials available online.

How To Use `template automator.ipynb`

1. **Setup Your Files**:

   a. Report Data: Have your `report.csv` in the `input/` directory.

   b. Mapping File: Place your `SetsAndMaps.xlsm` Excel file in the `input/` directory. Ensure it has sheets named "mapPRC" and "mapCOM".

   c. Filter File: Save your filter Excel files as `s_filter.xlsx` and `j_filter.xlsx` in the `input/` directory. These two will be concatenated to form one filter dataframe.

   d. Other Workbooks: Store the Excel workbooks you wish to update in the `input/` directory with names `WB1.xlsx`, `WB2.xlsx`, `WB3.xlsx`, etc.

2. **Run the Script**: Execute the script. Once it completes, your data will be processed, merged, filtered, aggregated, and relevant workbooks updated.

What Does This Script Do?

1. **Data Reading**:

   - It reads a report from a CSV file.
   - Reads two mapping tables from an Excel file.
   - Reads filter criteria from another Excel file or multiple concatenated Excel files.

2. **Data Processing**:

   - The special character `*` is treated as wildcard, meaning that any value is acceptable within this column.
   - Replaces 'Eps' values with 0 in a column of the report.
   - Merges the report with the mapping tables.
   - Automated error handling removes excess `'` and special characters that are not `*` within lists.
   - Makes sure `TargetUnit` column exists.

3. **Data Filtering**:

   - The script uses conditions defined in the filter Excel to select specific rows from the report. The conditions can involve checking if a value is within a list or equals a specific number/string.
   - Combines the filters and applies them to the report.

4. **Data Aggregation**:

   - The script aggregates (sums up) values based on unique identifiers and year.

- The summed values are adjusted based on the specified `TargetUnit` and rounded off.
- Values within the designated `cumulativeID` list are aggregated in ten year steps starting with the first available year.
- All other values are grouped by `UniqueID` and `Year` and then summed.

5. **Excel Workbook Updation**:
   - Based on the aggregations, specific cells in Excel workbooks (`WB1.xlsx`, `WB2.xlsx`, etc.) are updated with the new data.
   - **IMPORTANT:** Always wait for the `Done` message. Do not interrupt the Excel population operation once it is running since this will irreparably damage the Excel file.
6. **Output**:
   - Saves a merged version of the report as `merged_data.csv`.
   - Saves the results after applying filters as `results_filter.csv`.
   - Saves populated Excel workbooks as `WB1.xlsx`, etc.

Notes

- Ensure that your input files are correctly formatted and placed in the `input/` directory.
- If you want to add or change the filter or workbooks, adjust the file paths/names in the script.
- The script prints useful information while running, including which workbook, sheet, and row it's updating.
- Always keep backups of your files before running any script to prevent data loss.

Always verify the results using random selection and validation to ensure accuracy.