WebAutomato test task

In the financial sector, one of the routine tasks is mapping data from various sources in Excel tables. For example, a company may have a target format for employee health insurance tables (Template) and may receive tables from other departments with essentially the same information, but with different column names, different value formats, and duplicate or irrelevant columns.

Your task is to devise an approach using LLM for mapping source tables (in our case table A and table B) to the template by transferring values and transforming values into the target format of the template table (Template table for our task).

Important notes:

- The approach must be general - i.e. it should not rely on specific column names, formats and values.
- The format of source data in a certain column is guaranteed to be consistent across all the rows. All you need is to automatically infer the target format from Template table and create conversion functions (if needed) for each column.
- Table A and table B can contain millions of rows, hence your solution should not consider fitting the whole table content into the prompt, nor should it process each row through the LLM (it would be too costly)

Possible solution approach:

- Extract information about the columns of the Template table and tables A and B in the format of a text description.
- For each of the candidate tables (A and B), ask the LLM to find similar columns in the Template.
- In case of ambiguous mapping, choose the first option
- Automatically generate data mapping code for each column display in the final Template format. For example, for date columns, they may be in different formats, and it is necessary to change the order from dd.mm.yyyy to mm.dd.yyyy.
- Check that all data has been transferred correctly; if any errors occur, issue an alert to the person.

Expected result:

- Code on GitHub using OpenAI API, Langchain, and other LLMOps tools of your choice.
- A command line interface in which you can perform such an operation.

The script should take a template table and a source data table (A or B in our case) and a target table name.

It can be something like this:

```
convert_table.py --source <source CSV> --template <template CSV> --target <target CSV>
```

As the output, the user receives the target table in the Template format (columns, value formats) but with values from the source table.

- Your thoughts on edge cases and how they can be overcome