Takehome: Data Normalisation

Introduction

Please approach this takehome assessment the way you would for a real world problem. The overall assessment should take you 2-4 hours, with a hard limit on 4 hours of work. The purpose of this task is to identify how you approach and prioritise a problem. We would rather have working but incomplete solutions, than you spend more than 4 hours on it or a non-working solution.

Some other things to note about this assessment:

- Ask us questions. We will try our best to answer your questions quickly via the email below.
- Use a language you are comfortable in, but ensure that instructions to build (where applicable) and run the code are provided in a `README.md`.
- Use existing libraries and internet references where they make sense. Similar to "real-world" development there is no need to re-invest the wheel for everything.
- Focus on an MVP that works and can be shared. Once that is running add additional features.
- The solution should run on either Windows or Linux. If this cannot be ensured natively, please make use of Docker.

If you have any questions through this takehome assessment, please email us at: oliver.fritz@covarius.com. Once complete please publish the solution via a publicly accessible git repo (via GitHub, GitLab etc.) and send us a link to it.

We will be excited to review and discuss your code over video chat no matter the level of polish of your submission.

Storyboard

Covarius is developing an application that takes bank account data from one financial system and converts it into an acceptable format for another. The tool should run in the command line and read a JSON file and convert the contents to a CSV file. If the application was named `convert` it should work similar to the following:

```
$ ./convert input.json output.csv
```

where `input.json` contains the input data and `output.csv` is the output file. Any errors that occur due to the data being incomplete should be logged to the console via `stdout` or `stderr`, but the tool should continue to the next record.

Below are the details of the individual systems and their data formats.

> The format is as below:
>
> - `code` represent field names or code
> - *literal* represent literal values

Problem Details

Input Format

File format: JSON
Fields:

- `ibanNumber`: IBAN account number. Optional.
- `sortCode`: UK Sort code number. Optional.
- `accountNumber`: UK account number. Optional.
- `unstructuredAccountNumber`: Free format account field. Optional.
- `bankName`: Name of a bank and branch in a format of `<bank_name> - <country_code>`. Required.
- `name1` and `name2`: Name of the account. Limited to 35 characters each.
- `notes`: Free-format field. Optional.

Comments:

- To be valid the account needs at least one of the following combinations:
  - `ibanNumber` OR
  - `sortcode` and `accountNumber` OR
  - `unstructuredAccountNumber`
  - If none are found, the error should be output.
- The whitespaces within `bankName` should be removed.

Output Format

File format: CSV
Fields:

- `accountNumber`: Valid account number. Required.
- `accountNumberType`: The type of the account number, can either be: *iban*, *gbDomestic* or *unstructured*. Required.
- `bankName`: Name of the bank. Required.

- `branchCountry`: Country code of the bank branch. Required.
- `name1` and `name2`: Name of the account. `name1` is limited to 30 characters, `name2` is limited to 20 characters. Required.
- `userComments`: Free-format field, max 30 characters. Optional.

Mappings:

- The `accountNumber` and `accountNumberType` should map the fields from the input file's with the following priority:
    1. `ibanNumber` value to `accountNumber` and *iban* to `accountNumberType`
    2. `sortcode` and `accountNumber` values concatonated to `accountNumber` and *gbDomestic* to `accountNumberType`
    3. `unstructuredAccountNumber` value to `accountNumber` and *unstructured* to `accountNumberType`
- `bankName` should be the bank name portion of the input files's `bankName`
- `branchCountry` should be the 2-character ISO country code
- `name1` and `name2` should be mapped from the input files's concatonated `name1` and `name2` fields, respecting the field length.
- `userComments` should contain the sanitised version of `notes`, only accepting ASCII characters. Excess should be removed.

Safe Assumptions:

- The input document is in UTF-8, although some characters may be incorrectly encoded.
- Any type of line endings are permissible in the output file.