Solirius data engineering coding test

After receiving this document, you will be given two days to complete the following tasks.

You will be provided with a code repo as a zip file , which will contain the files you need to complete the tests. Once the test is complete, add your code to a zip file and email it to natalya.short@solirius.com. If you finish the test early, you can email to signal this, so that we can get back to you sooner. After the test, one of our engineers will discuss your code with you in the next interview stage.

Test requirements

- *Python 3*
- *Either Pyspark (recommended) or Pandas*
- *Luigi (recommended) or other pipeline API*
- *Pytest (recommended) or other testing framework*
- *Python Great expectations (recommended)*

Task one

Given a dataset of films, the client would like you to perform some transformations and cleaning of the dataset so that their recommendation service can access particular films faster. The output of each stage must be within a folder called output .

- *Stage 1: Import the input dataset, convert the data types, then save as a parquet file.*
  - *schema.json file should be used to load the dataset*
  - *The output parquet file must be called: films*
- *Stage 2: Taking the output from stage 1, create a folder called genres, and inside this folder produce parquet files for each subgenre from the dataset. So for example, Action.parquet will contain all the die hard/mission impossible films, etc.*
  - *Example valid names:*
    - *Adventure*
    - *genre=Action*
    - *Faith_and_Spirituality*
    - *genre=Faith_and_Spirituality*
  - *The schema of each genre parquet must match the schema from stage1*
  - *A genre name like “Action, adventure` should be broken down into separate genres [“Action”, “Adventure”]*

Once the stages have been implemented, they must be automated using a pipeline API of your choice.

Task two

The client requires a function to detect similarity between films. The function will take in a film's id, and a threshold percentage as input, and will return a dataframe that contains all films with a similarity percentage above the threshold. The way similarity is calculated is up to you, but the output should be sensible. (For example, any star wars film should be similar to all other star wars films, or films by the same director have a similar style etc.)

Task three

The client requires functionality to allow users to query their dataset. The user should be able to use any number of columns, alongside a range of values for each column to filter films. For example, the user may want to find all films by Quentin Tarantino or George Lucas that were released between 1979 and 2000. The user should be able to use a variety of querying methods, such as between a range of values, or from a set of specific values, where appropriate. The dataset that should be used is your output from Task one, and the system you build should be flexible and reusable. You must build just the backend for this functionality. You should create some suitable test cases to demonstrate this functionality is working as intended.