Data Engineer Assignment
Summer 2022

Task Summary

This assignment involves data collected for playlists on Spotify. Each playlist contains a range of numeric attributes, which describe specific parameters of the playlist. We also provide a file with a list of tracks per playlist. The task is to build a data pipeline with slight transformation of the data, linking the two sources, setting up a database for the data and a REST service that can serve the data. Find more details on the task below.

Data

Please download the following 2 CSV files contained in this zip file:

- playlists+tracks.zip

(Note: Both CSV files use ; as the field separator)

playlists.csv:
This file contains information about 12,000 playlists from Spotify with the following columns:

- playlist id
- playlist name
- followed_by (# of followers of the playlist)
- num_unique_artist_first_page (number of unique artists of the first 50 tracks in the playlist)
- artist_followers_avg, artist_followers_stdev (average and standard deviation of # of followers of the artists of the top 50 tracks in each playlist)
- average and standard deviation of a range of numeric features from the tracks on those playlists, computed from the top 50 tracks in each playlist:
    - acousticness, duration, danceability, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time_signature, valence

playlist_tracks.csv

This file contains slightly more than 1 million rows, each of them with playlist_id;track_id.
The playlist_id is repeated, so in the end you'll have a list of tracks that are part of the same playlist_id.
Note that the same track_id is allowed to appear on multiple playlists.

Task

Build a data pipeline with a Web service:

1) **Analyze and transform playlists data:**
   - Read **playlists.csv**

   Normalize:
   - **Define** an environment variable called NORMALIZATION_MAX or **a config file with same variable** and set its value to 100
   - Read the config variable or environment variable in your code
   - **Normalize** all numeric columns in the playlist data to the range 0-NORMALIZATION_MAX (using 100 here as per default env var setting, but possibly another value if the env var is changed)
   - Output a CSV file "playlists_normalized.csv" containing all the normalized values

   Average:
   - Remove the "name" column
   - Compute the **average** of all normalized numeric values across the entire dataset
   - Output a CSV file "playlists_average.csv" containing 2 rows: the header row (minus "name" column) and a row with the average values.

2) **Load and match tracks to playlists:**
   - Read **playlist_tracks.csv**
   - Parse it to make a list of tracks belonging to a playlist_id
   - Output a JSON file "playlist_tracks_stats.json" with the following (as key/value pairs):
     - Total number of playlists
     - Total number of unique tracks
     - Minimum number of tracks of all playlists
     - Average number of tracks per playlist
     - Maximum number of tracks of all playlists
   - Perform a matching of the playlist_ids in this data to the playlist_ids in the data of step 1 (playlists.csv), so that the name of the playlist can be added to the list of tracks (used in 3rd endpoint below)

3) **Build a REST service:**
   - Build a simple web service API using Python Flask

- At initialization, the web service should perform the steps 1 and 2 above
- The service should have the following endpoints:
    - GET /max_value
    - GET /playlist/<playlist_id>
    - GET /tracklist/<playlist_id>
- **The response of all endpoints shall be in JSON format**
- GET /max_value should return “max_value”: and the value set in variable NORMALIZATION_MAX
- GET /playlist/<playlist_id> should return all elements of the playlist specified by playlist_id, including name and the **normalized** values
- The response format should be a JSON dictionary with “key”: “value” elements where “key” corresponds to each field name in the header of the original CSV file (/ the field name in DB)
- GET /tracklist/<playlist_id> should return a dictionary containing playlist_id, playlist name and a list of track_ids associated with the playlist as per data loaded in step 2 above
- On both endpoints, if a playlist_id is provided that is not in the data, respond with 404 “Playlist_id not found”

- Add unit tests for the data reading and transformation steps (steps 1 and 2)
- If time permits, add tests for the endpoints
- **Add a README.md file that mentions how to start the service**

Implementation details

- Use Python 3 (preferably 3.8 or 3.9)
- You may use any publicly available Python libraries (if you use other/private ones, please motivate in the summary report below)
- Write clean code and document the code well
- Wherever possible use Object Oriented Programming
- Add a **requirements.txt** listing the libraries you used and are needed to run the code, with exact version number (==)
- All **output files** generated should go into a **folder ./output** and should be **included** in the submitted repository

Summary Report

Please write a summary report of approximately 2 pages length which includes the following:

- describe your solution as an overview in a short paragraph
- describe and motivate your choice of libraries
- short mention of implementation design considerations / code organization

- mention briefly alternative choices of libraries / implementation choices that you may consider if you had more time
- mention any obstacles or difficulties you had/have regarding this task
- mention the time needed to complete the task

Include the report in the root of the submitted repository as either .txt, .md, or .pdf file named “summary_report.*”.

Submission

- store code **and summary report** on a **private** GitHub repository
- **share the Github repo** with Github users: audiofeature, philtgun-utopia, jackjc9, oscaratutopia
- **notify** the person who gave you the assignment that you finished the task

If you can’t use Github for whatever reason, send a zip file with the content **including the summary** to the person who gave you the assignment.

Deadline

- 1 week after receiving the assignment (please let us know if this is not feasible for you)

Contact

- The hiring manager who contacted you
- Thomas Lidy <thomas.lidy@utopiamusic.com>
- Victor Marklund <victor.marklund@utopiamusic.com>