Summary

describe your solution as an overview in a short paragraph:

My solution contains two parts: data transformation and Flask API. Data transformation performs the data load, data normalization, data analysis, report generation, and table combination on key.

The result from data transformation has been kept as CSV and JSON files in the output folder which are later served as a database/data source for Flask API

The flask API provides three endpoints to query different information from the existing dataset.

Route	Verb	info
max_value	GET	return the value set in variable NORMALIZATION_MAX
playlist/<playlist_id>	GET	display all elements of the playlist specified by playlist_id, if playlist_id doesn't exist, return 404
tracklist/<playlist_id>	GET	return a list of track_ids associated with the playlist, if playlist_id doesn't exist, return 404

describe and motivate your choice of libraries:

I used pandas, NumPy, and Flask. Because pandas can load CSV to the data frame and provides complex operations on the data frame. I was thinking the data scale is not extremely large, and the procedure was not slow, so that’s why I chose pandas

a short mention of implementation design considerations / code organization

I have considered splitting the task into two projects - 1. Data transformation and 2. Flask API

```
.
├── data_transformation                    # Task I & II
│   ├── src
│   │   ├── __init.py__
│   │   ├── data_toolkit.py              # Apply data transformation
│   │   └── transformation.py             # Data transformation entry
│   ├── test
│   │   └── test_transfromation.py             # Unit test
│   └── data_exploration.ipynb               # Data exploration with Jupyter Notebook
├── flask_api
│   ├── app
│   │   ├── app.py             # api entry point
│   │   └── resrouces
│   │       ├── __init__.py
│   │       └── resrouce.py          # api endpoints
│   └── test
├── output      # output folder to save the result
├── log      # output folder to save the log
├── .gitignore
├── requirements.txt
└── readme.md
```

Specifically, in data transformation, there is a “src” and “test” folder to separate the functionality. For Flask API there is an “app” and “test” folder.

The output folder is both worked as the output for the data operation result and the input for the flask API. The config file is also shared across the code.

mention briefly alternative choices of libraries / implementation choices that you may consider if you had more time

About libraries I would have considered using **Pyspark**, Pysparks compute in a parallel way and provide lazy execution which makes it faster than pandas.

In terms of implementation, I would use **Docker**, since it makes the deployment safer and faster in a new environment. **Airflow** to automate the job and have separate **development** and **production** environments. **Swagger** to document the API and implement **Pytest** for API.

mention any obstacles or difficulties you had/have regarding this task

I did a simple data exploration before I started coding. I noticed the data was quite clean, but there were a few missing data and data type issues. When I performed the data normalization. I wondered if I should check if are there any outliers, and I actually noticed the data range varies huge (std for some columns are extremely high). So I kept the 2%-98%

of the dataset as it is. I am not sure about that, maybe I didn't need to remove outlier at all…

Also, I am a little bit confused about the question. Since I am not sure whether the project should be one Flask API (Questions 1&2 should serve the API inside the API) or the project should be split into different sub-tasks. So that also made me hard to design the code organization.

I am not sure if my solution is what the question wants.

mention the time needed to complete the task

I think the task took me around 8-10 hours to complete designing, coding, and documentation