Weather & Streaming Data Engineering Project

PROJECT WRITEUP

Prepared by: Michelle Gordon

Introduction

Music and technology are deeply entrenched in the daily lives of most human beings. Music is more accessible than ever thanks to technology that allows us to have unimaginably large music libraries available at our fingertips. Gone are the days of being hindered by the price of physical media or listening to a radio station all day long to finally hear your favorite song. Something else that is deeply entrenched in the daily life of humankind is weather. Every single day, we make decisions about what to wear, how early to leave the house, and when and where to hold celebrations based on weather conditions. Moods and physical pain are affected by weather conditions such as the barometric pressure, or cloud coverage so it is no stretch of the imagination to think weather patterns may affect the music we choose to listen to.

The high-level goals of this project were to create a data pipeline that would:

- extract historical music streaming data from the Spotify API
- extract historical weather data from the OpenWeather API
- Load the transformed data in a data warehouse so that an analytics team can extract insights about the effects of weather on music streaming choices.

In the future, this project could be expanded to see weather effects on audio book and podcast streaming choices as well.

The following sections will introduce the team that took on this project as well as explain the technical components, methodologies used, the overall architecture and retrospective thoughts on the completed project.

The Team

The entire project was completed by a one-person team consisting of University of Utah MSIS student, Michelle Gordon. Michelle has played with tech and music since the age of six. When her father wasn't teaching her to write DOS commands on her family's Radio Shack computer, she was performing paid concerts (She charged five cents per ticket) in her apartment complex's playground with the help of her mini Casio keyboard and Fisher Price karaoke

machine. Presently, she is about to graduate with her co-terminal BS/MSIS degree from the Eccles School of Business with plans to enter the field in a coding-forward position.

Technical Components

The project was completed using a combination of the Python programming language, Jupyter Notebooks, RESTful API's, JSON files, and MySQL. Visual Studio Code (VSCode) was used to write and execute the Python files and Jupyter Notebooks. MySQL Workbench was used to setup the destination database. All the project files were managed via Github's desktop application and their VSCode extension.

Extraction

Spotify's account services, Spotify's API, and OpenWeather's API were used as the data sources. While the intention was to have all the data programmatically downloaded, Spotify only provides historical streaming data for users via their account services. They require that you request the data in your account setting, wait for them to gather all the data, and then manually download it from the account settings page. The audio features for each song in the streaming history were extracted programmatically from Spotify's API using Python scripts. The process involved the following steps (all automatically run by Python unless otherwise indicated:

1. Manually download the streaming history JSON file which contains 6416 records.
2. Load the JSON file into a Pandas dataframe. Each row is a track streamed, so there may be multiple tracks per day and the tracks will likely appear more than once in the dataset.
3. One row at a time, send a get request with the track name and artist name to the Spotify API's track search endpoint to then receive the track's id number.
4. Append the id numbers to the dataframe and save it as an updated JSON file.
5. Break up the streaming history into batches of 100 records (as it is the maximum number of id numbers Spotify accepts at once) and send one batch at a time in the get request for audio track features.
6. Save all the features in a dataframe and then serialize the data into a JSON and CSV file.

    a. Even though only one file format was needed, both file formats were used as a safety net in case one of them caused issues further down the pipeline.
7. At this point there is a streaming history file that now contains id numbers, and an audio features file. Both files have 6416 records.

The historical weather data was also extracted programmatically using Python scripts to interact with the API endpoint. This data includes information such as precipitation, wind speed, wind direction, barometric pressure, temperature, and cloud coverage. The process to extract the weather data was as follows:

1. Send a get request to OpenWeather's API with the zip code and country code for Taylorsville, Utah (where Michelle lives since her streaming history is being used for the project) to receive the city's latitude and longitude coordinates.
2. Create a list with each date that is part of the streaming history (367 days total).
3. Send one date at a time, along with the latitude and longitude coordinates, to OpenWeather's API to receive that day's weather data. Then append that data to a JSON file.
4. At this point, there will be a JSON file containing 367 records.

Transformation

Once all of the data was captured and serialized into JSON files, the data was loaded into Pandas dataframes. Transformations were applied using Jupyter notebooks to see the changes more easily as they were occurring. Some of the transformations included:

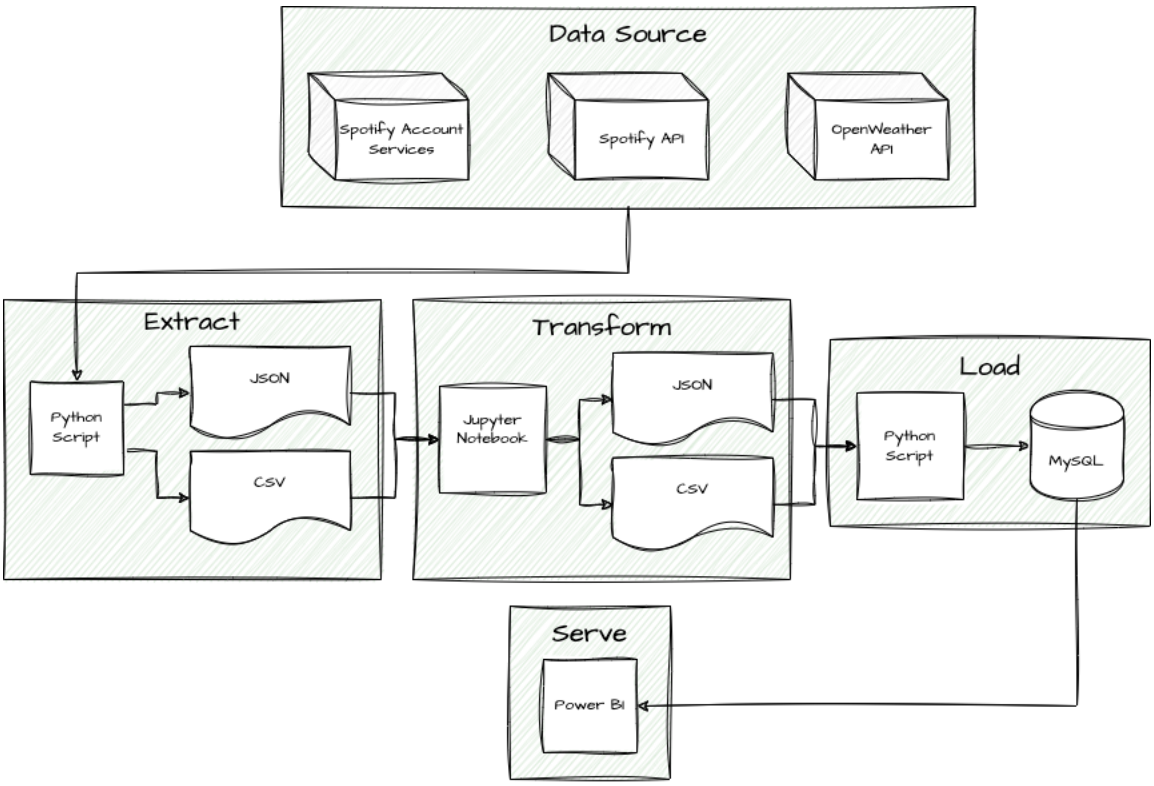
- changing datetime stamps to dates
- removing unnecessary columns
- converting song lengths from milliseconds to minutes
- changing numerical categorical data to more descriptive information
- Removing rows with zeroes for track ids, as these were tracks Spotify could not find id info on.
    - Some of these circumstances seemed to come from changed track names or artist names. As these records made up less than 5% of the data, they were removed.

Once these transformations were complete, all three dataframes (streaming history, track audio features, and weather data) were serialized into JSON and CSV files. Both file formats were used as a safety measure in case one of the files caused issues during the loading phase.

Loading & Serving

MySQL was used to host the data warehouse that would serve data to analytics teams. To keep data integrity, the three datasets were inserted into three respective tables in the database as is. The goal was to create views for analytics purposes instead of requiring analysts to join the tables themselves. Giving analysts access to the views would give them access to the data formatted the way they need it. One view was created that joined all three datasets together, and three other niche views were created as well.

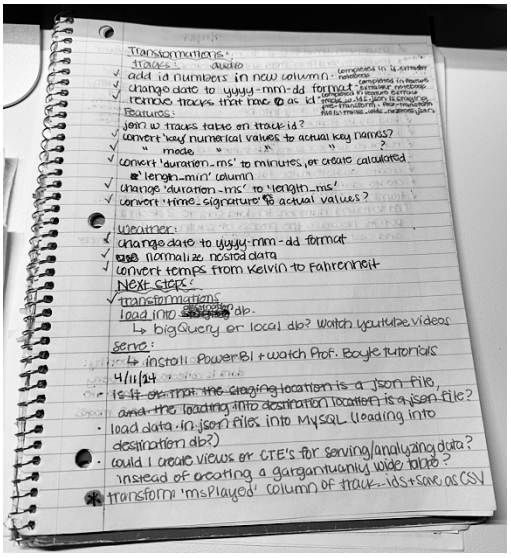
Architecture



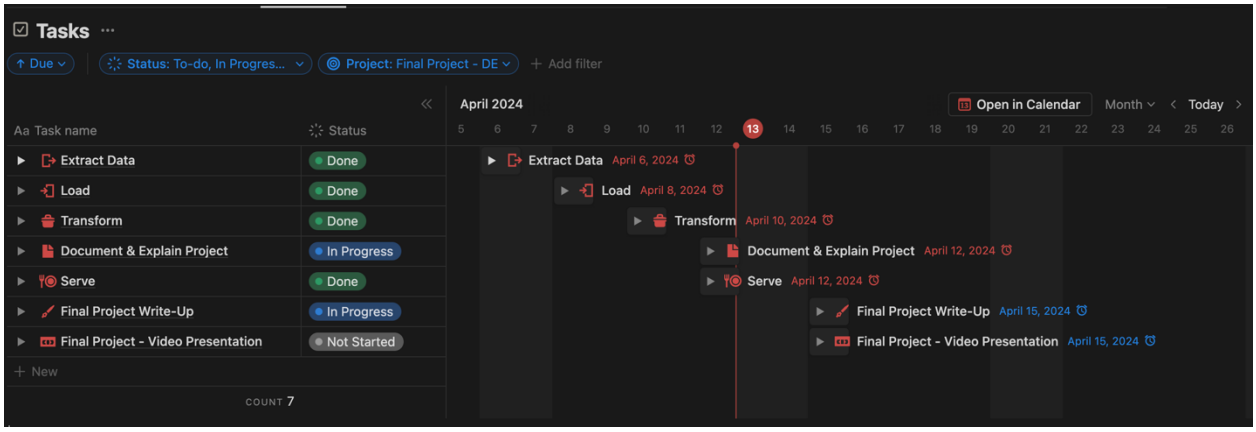
Python scripts and Jupyter Notebooks were used because taking a software engineering approach was more desired than choosing a low-code route. These technologies also aligned with the technologies used in the Data Engineering course. Additionally, MySQL and Power BI were chosen for the loading and serving steps because they were the tools the I was most familiar with and therefore, would be most successful with. Additionally, as previously noted, CSV and JSON file formats were both used as a safeguard in case one of

the formats became problematic at any step of the process. As noted in the following section, that proved to be a good choice because of some problems that popped up in the ETL process.

Project Work Review



Research and planning were the first steps taken in beginning this project. The APIs' documentation along with YouTube tutorial videos were used to understand how to correctly extract the necessary data from the APIs. Preliminary working files were created to test the API calls using very small data sets. This was to make sure the API calls were successfully authenticated and returning the correct information. From this point, the planning phase began. On paper, the different stages of the project were written out, as well as detailed steps for each stage to help make a clear pathway. Notion, a productivity service, was used to schedule due dates for each step of the project and easily track the progress being made.



One of the themes that popped up while working on this project is the fact that I kept underestimating the difficulty of each step of the process. For example, extracting data seemed like a simple step, and proved to be more difficult than expected once the real data was flowing through the pipelines. At that point, I thought that extraction was the most difficult step, and that the next stages would go a lot more quickly and smoothly. That was certainly not the case. Each step had its difficulties and frustrating moments.

One of the biggest surprises came from getting data from Spotify (the streaming history) with what seemed to be orphaned records. When the songs from the history were sent into the API to get the track id number, many songs came back without ids. It was also surprising that even the data coming back from these songs without ids wasn't consistent. Some of them had the entire key-value structure setup, but with empty values. Some only came back with the first key, but empty. Other songs returned an empty list. It was unexpected to see all these inconsistencies with raw data that was being fed back to its own source.

My lack of experience with JSON files and exporting them from Pandas dataframes became a frequent source of problems. Although I thought I was familiar with JSON due to the course assignments, problems kept popping up when it came to the interaction between dataframes, JSON files, and JSON data coming in from API's. There were times it seemed like the dataframe had been pivoted when saved to a JSON file and I could not figure out why. Therefore, the decision was made to save files in both JSON and CSV format so that I revert to the CSV files if there were problems with the JSON data.

Ultimately, the extract and transform stages of the project proved to be more complex and time-consuming than expected. I now understand why it's so easy for people to underestimate the work a data engineer does when in reality, it is exceedingly complex. The biggest culprit is that data engineers are at the mercy of the data sources – especially if it is a 3rd party data source.

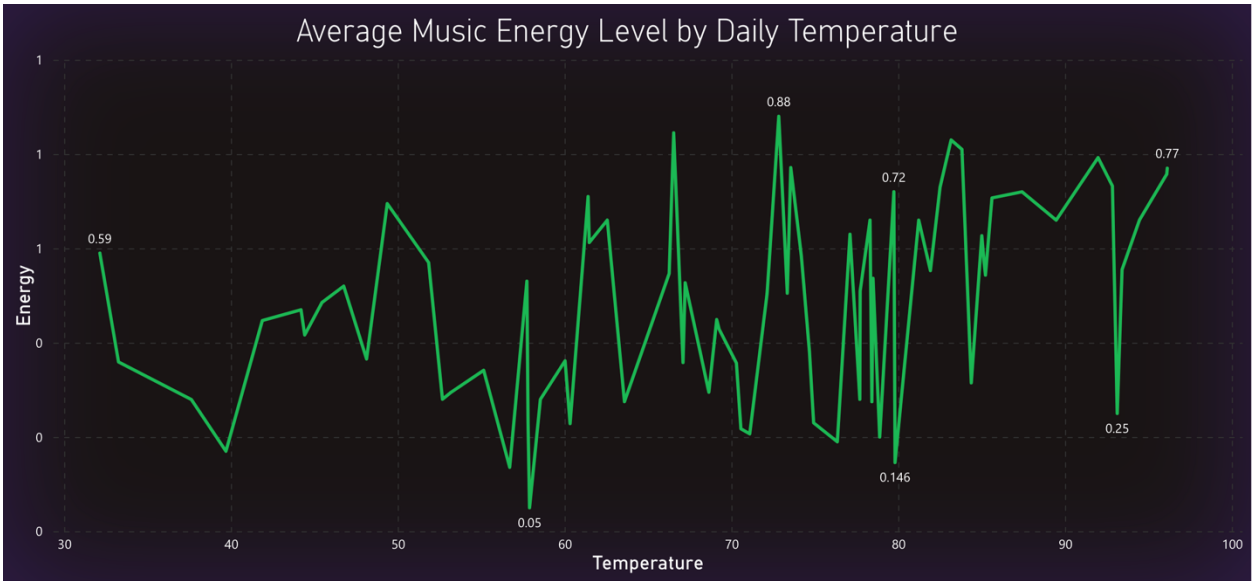
Retrospective

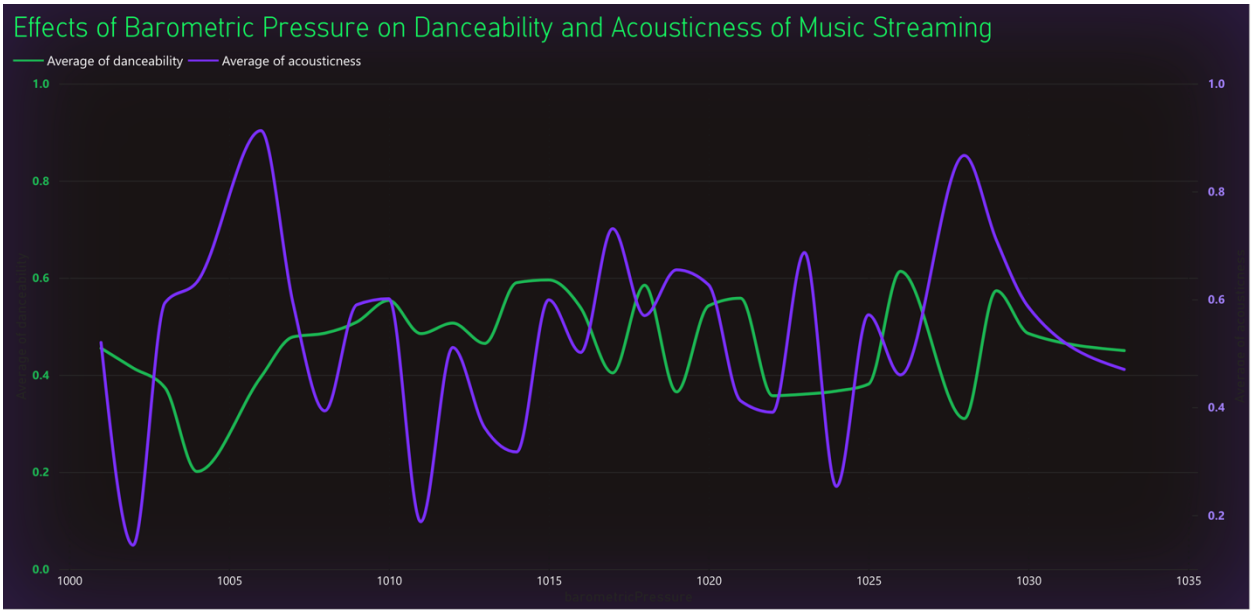
Despite the difficulties, the project still proved to be successful. In the first few minutes of analyzing and visualizing the data in the database, all of the work became well worth the effort. It has been rewarding to see some of the insights gained from pairing up the weather data with the streaming history.

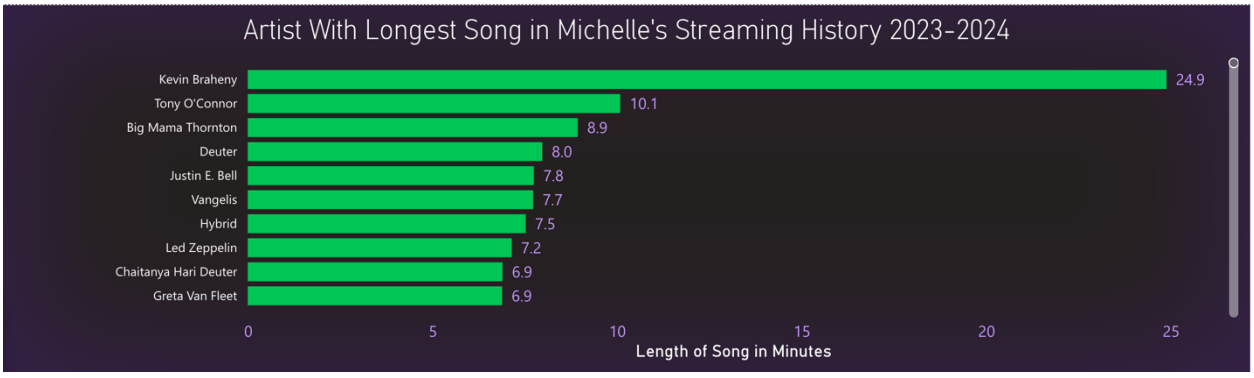
There are two things that would greatly enhance future projects. First, it would be wise for me to learn more about the interaction between JSON and the Pandas library in Python. A lot of time and frustration would likely be saved by understanding the complexities that exist under the hood. Second, using Apache Airflow or a cloud-based orchestration service would boost the efficiency and reliability of the whole project. Furthermore, this project would benefit from having team members that are versed in cloud services for data engineering as they would be able to leverage all of the benefits that come from hosting projects like these in the cloud.

Lastly, this is a project that could provide fascinating insights if executed at a much larger scale than what I did. The streaming history for all Spotify users in a city could be collected along with the respective weather information to create a much larger data set and more conclusive insights.

Included on the following page are visualizations built from the data served from the project.


Average Music Energy Level by Daily Temperature
Energy
Temperature
0.59
0.05
0.88
0.72
0.146
0.25
0.77


Effects of Barometric Pressure on Danceability and Acousticness of Music Streaming
Average of danceability
Average of acousticness


Artist With Longest Song in Michelle's Streaming History 2023-2024
Kevin Braheny 24.9
Tony O'Connor 10.1
Big Mama Thornton 8.9
Deuter 8.0
Justin E. Bell 7.8
Vangelis 7.7
Hybrid 7.5
Led Zeppelin 7.2
Chaitanya Hari Deuter 6.9
Greta Van Fleet 6.9
Length of Song in Minutes