Group 6 ETL Project

Max Izotov, Julissa Guzman, MaryLouise Pabello
November 4, 2020

Data Sources:

Our team obtained data from Kaggle to Analyze the Top Spotify Tracks of 2017 and 2018.

Data	Data Type	Data Source
Top Tracks of 2017 Spotify Data	csv	https://www.kaggle.com/nadintamer/top-tracks-of-2017
Top Tracks of 2018 Spotify Data	csv	https://www.kaggle.com/nadintamer/top-spotify-tracks-of-2018

The dataset can be used to analyze the song patterns from 2017-2018 and analyze what audio features make them popular.

Extraction:

We obtained the data from the Kaggle website as csv. These csv files are included in the repository, in the " Resources" data folder.

Transformation:

Our original data had the following columns: **id** (Spotify's unique URL for each song); **name** (a song's title); **artists**; **danceability** (a Spotify-determined numeric value for how suitable a song is for dancing); **energy** (a song's perceived intensity and activity); **key** (the song's musical pitch); **loudness** (measured in decibels); **mode** (a music term that identifies the song's musical scale as either major or minor); **speechiness** (how much spoken word is present in a song compared to instrumental); **acousticness** (a Spotify confidence value of a song's being acoustic [lacking any electronic amplification such as auto-tune]); **instrumentalness** (identifies if a song has no vocals present); **liveness** (identifies the presence of an audience in the song); **valence** (a Spotify-determined measure of a song's musical positiveness [will a person's emotions be happy when listening to a song]; **tempo** (a musical term that measures a song's speed and is measured in beats per minute); **duration_ms** (the length of a song in milliseconds); and **time_signature** (Spotify's estimate of a song's meter, a musical term that denotes how many beats are in each measure of a song).

Next, we loaded the following dependencies into Jupyter Notebook:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
import mysql.connector
```

then read the two csv files into Jupyter Notebook using Pandas and created the dataframes "data_2017" and "data_2018". We were only interested in keeping the columns name, artists, danceability, energy, loudness, speechiness, liveness, tempo, and duration. We used Pandas to create new data frames for the cleaned data, without the unwanted columns ("cleaner_2017" and "cleaner_2018").

We then renamed the columns "name" and "duration_ms" to "song_title" and "duration_min" to better reflect the information in those columns (a song's title and its duration to minutes instead of milliseconds). And then converted the remaining numerical columns (danceability, energy, loudness, speechiness, and liveness) into percentages so the value of each numerical measure was easier to understand. Our final dataframes are called "cleaner_2017" and "cleaner_2018".

See below for the code used to clean the 2017 data as described, which we then repeated to clean the 2018 data.

```
clean_2017 = data_2017[['name',
'artists','danceability','energy','speechiness','liveness','tempo','duration_ms']]
cleaner_2017 = clean_2017.rename(columns = {"name": "song_name", "duration_ms":
"duration_min"})
cleaner_2017['duration_min'] = cleaner_2017['duration_min'].div(60000).round(2)
cleaner_2017['tempo'] = cleaner_2017['tempo'].round(2)
cleaner_2017['danceability'] = cleaner_2017['danceability'].multiply(100)
cleaner_2017['energy'] = cleaner_2017['energy'].multiply(100)
cleaner_2017['speechiness'] = cleaner_2017['speechiness'].multiply(100)
cleaner_2017['liveness'] = cleaner_2017['liveness'].multiply(100)
cleaner_2017
```

Load

With our dataframes cleaned, the next step was to create two tables in the PostgreSQL database called "ETL_project" in order to load our data. The two tables are called "data_2017" and "data_2018". Refer to the file "postgrescode.txt" for the queries we ran in Postgres to create the tables.

We then went back to Juptyer Notebook and ran the following code to load our dataframes into the corresponding Postgres tables:

```
connection_string = "postgres:postgres@localhost:5432/etl_project"
engine = create_engine(f'postgresql://{connection_string}')
engine.table_names()
cleaner_2017.to_sql(name='data_2017', con=engine, if_exists='append', index=False)
cleaner_2018.to_sql(name='data_2018', con=engine, if_exists='append', index=False)
```

To confirm that the data loaded correctly, we ran the following queries in Postgres:

SELECT * FROM data_2017

SELECT * FROM data_2018

The tables returned matched the dataframes we created in Pandas.

We chose a relational database for our tables because the two tables have the “artists” column in common and that could serve as a primary key for both. Using “artists” as a key between the two tables, one could find out how many songs a particular artist had on Spotify’s top tracks for 2017 and 2018, for example.