Dentsu Data Engineering Challenge

This take-home challenge involves pulling data from an API, creating a database, and running an analysis. We should be able to replicate all your work - please provide any code, notes, and/or documents you used while analyzing the data in a zip file package to help us follow your thought process. This take home assignment is property of Dentsu and is not to be shared without permission.

Intro

For this challenge, we'll be working with the Spotify API. We recommend the `spotipy` library to interact with the API in Python. We ask that you munge the data into a SQLite database. `sqlite3` is part of the Python standard library. Please attach the SQLite db file as part of your submission. Finally, answer some questions using SQL queries and Pandas. Feel free to add visualizations using the tool of your choice.

Spotipy API

To use the Spotipy Python API, you need to use a Spotify "app." If you have a Spotify account you can set one up on their website. Otherwise feel free to use ours:

```
os.environ['SPOTIPY_CLIENT_ID'] = '00b7317977ad4c0d971af8274f1aa790'
os.environ['SPOTIPY_CLIENT_SECRET'] = '6efbf45fe72d435f9739d0c0f4c26db5'
os.environ['SPOTIPY_REDIRECT_URI'] = 'https://360i.com/'
base_url = 'https://api.spotify.com'
scope = 'playlist-read-private'

# spotify:playlist:
rap_caviar = '5yolys8XG4q7YfjYGl5Lff'
token = util.prompt_for_user_token('Puffer Fish', scope=scope)
spotify = Spotify(auth=token)
```

Documentation for the python API can be found here.

ETL

Please build a database as described below using the Spotify "Rap Caviar" playlist.

1. Create a table `tracks` and fill it with songs from the Spotify "Rap Caviar" playlist. For songs with multiple artists, you can just use the first one listed.

   `tracks` table minimum fields:

   - track_id
   - track_name
   - artist_id
   - artist_name
   - album_id
   - album_name

    - album_release_date
    - album_type
    - track_popularity
    - explicit
2. Create a table `artists` and fill it with all of the artists in the `tracks` table.

   `artists` table minimum fields:

    - id
    - name
    - popularity
    - followers

Analysis

Please answer the following questions using SQL queries with the `sqlite3` package. Your result should use a single select statement (without any WITH clauses) and return a single table:

1. On what day of the week are most albums released?
2. Which artists have 3+ tracks in `tracks`? Sort your results by the number of tracks from highest to lowest.
3. Which tracks have both a track and artist popularity of at least 90?

Please answer the following questions using Pandas:

4. What are the top 3 months for album releases?
5. Are clean or explicit songs more popular on average?
6. Is there a relationship between track and artist popularity? If so, what is it?

If you find that you have some time and want to investigate ideas outside of the questions above, we'd be delighted, but it's not expected. We understand that you've got a busy schedule. Below are some possible directions to go in.

7. Add additional fields to the tables of your choosing.
8. Pick another playlist and add to the track and artist tables.
9. Compare features across playlists.
10. Normalize your database.