# Lab 7: Songs

> You are welcome to collaborate with one or two classmates on this lab, though it is expected that every student in any such group contribute equally to the lab.

Write SQL queries to answer questions about a database of songs.

## Getting Started

Here's how to download this lab into your own CS50 IDE. Log into CS50 IDE (https://ide.cs50.io/) and then, in a terminal window, execute each of the below.

- Execute `cd` to ensure that you're in `~/` (i.e., your home directory, aka `~`).
- Execute `wget https://cdn.cs50.net/2020/fall/labs/7/lab7.zip` to download a (compressed) ZIP file with this problem's distribution.
- Execute `unzip lab7.zip` to uncompress that file.
- Execute `rm lab7.zip` followed by `yes` or `y` to delete that ZIP file.
- Execute `ls`. You should see a directory called `lab7`, which was inside of that ZIP file.
- Execute `cd lab7` to change into that directory.
- Execute `ls`. You should see a `songs.db` file, and some empty `.sql` files as well.

## Understanding

Provided to you is a file called `songs.db`, a SQLite database that stores data from Spotify (https://developer.spotify.com/documentation/web-api/) about songs and their artists. This dataset contains the top 100 streamed songs on Spotify in 2018. In a terminal window, run `sqlite3 songs.db` so that you can begin executing queries on the database.

First, when `sqlite3` prompts you to provide a query, type `.schema` and press enter. This will output the `CREATE TABLE` statements that were used to generate each of the tables in the database. By examining those statements, you can identify the columns present in each table.

Notice that every `artist` has an `id` and a `name`. Notice, too, that every song has a `name`, an `artist_id` (corresponding to the `id` of the artist of the song), as well as values for the danceability, energy, key, loudness, speechiness (presence of spoken words in a track), valence, tempo, and duration of the song (measured in milliseconds).

The challenge ahead of you is to write SQL queries to answer a variety of different questions by selecting data from one or more of these tables.

## Implementation Details

For each of the following problems, you should write a single SQL query that outputs the results specified by each problem. Your response must take the form of a single SQL query, though you may nest other queries inside of your query. You **should not** assume anything about the `id`s of any particular songs or artists: your queries should be accurate even if the `id` of any particular song or person were different. Finally, each query should return only the data necessary to answer the question: if the problem only asks you to output the names of songs, for example, then your query should not also output each song's tempo.

1. In `1.sql`, write a SQL query to list the names of all songs in the database.
   - Your query should output a table with a single column for the name of each song.
2. In `2.sql`, write a SQL query to list the names of all songs in increasing order of tempo.
   - Your query should output a table with a single column for the name of each song.
3. In `3.sql`, write a SQL query to list the names of the top 5 longest songs, in descending order of length.
   - Your query should output a table with a single column for the name of each song.
4. In `4.sql`, write a SQL query that lists the names of any songs that have danceability, energy, and valence greater than 0.75.
   - Your query should output a table with a single column for the name of each song.
5. In `5.sql`, write a SQL query that returns the average energy of all the songs.

- Your query should output a table with a single column and a single row containing the average energy.

6. In `6.sql`, write a SQL query that lists the names of songs that are by Post Malone.
   - Your query should output a table with a single column for the name of each song.
   - You should not make any assumptions about what Post Malone's `artist_id` is.
7. In `7.sql`, write a SQL query that returns the average energy of songs that are by Drake.
   - Your query should output a table with a single column and a single row containing the average energy.
   - You should not make any assumptions about what Drake's `artist_id` is.
8. In `8.sql`, write a SQL query that lists the names of the songs that feature other artists.
   - Songs that feature other artists will include "feat." in the name of the song.
   - Your query should output a table with a single column for the name of each song.

### Hints

- See this SQL keywords reference (https://www.w3schools.com/sql/sql_ref_keywords.asp) for some SQL syntax that may be helpful!

### Testing

Execute the below to evaluate the correctness of your code using `check50`.

```
check50 cs50/labs/2020/fall/songs
```

## How to Submit

1. Download each of your 8 `.sql` files (named `1.sql`, `2.sql`, ..., `8.sql`) by control-clicking or right-clicking on the files in CS50 IDE's file browser and choosing **Download**.
2. Go to CS50's Gradescope page (https://www.gradescope.com/courses/157004).
3. Click "Lab 7: Songs".
4. Drag and drop your `.sql` files to the area that says "Drag & Drop". Be sure that each file is correctly named!
5. Click "Upload".

You should see a message that says "Lab 7: Songs submitted successfully!"

## Acknowledgements

Dataset from Kaggle (https://www.kaggle.com/nadintamer/top-spotify-tracks-of-2018).