This is CS50

CS50's Introduction to Computer Science

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/) (https://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

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

Open VS Code (https://cs50.dev/).

Start by clicking inside your terminal window, then execute `cd` by itself. You should find that its "prompt" resembles the below.

```
$
```

Click inside of that terminal window and then execute

```
wget https://cdn.cs50.net/2022/fall/labs/7/songs.zip
```

followed by Enter in order to download a ZIP called `songs.zip` in your codespace. Take care not to overlook the space between `wget` and the following URL, or any other character for that matter!

Now execute

```
unzip songs.zip
```

to create a folder called `songs`. You no longer need the ZIP file, so you can execute

```
rm songs.zip
```

and respond with "y" followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd songs
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
songs/ $
```

If all was successful, you should execute

```
ls
```

and you should see 8 .sql files, `songs.db`, and `answers.txt`.

If you run into any trouble, follow these same steps again and see if you can determine where you went wrong!

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. After you do so, you'll reflect on the ways Spotify might use this same data in their annual Spotify Wrapped (https://en.wikipedia.org/wiki/Spotify_Wrapped) campaign to characterize listeners' habits.

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.

Walkthrough

This video was recorded when the course was still using CS50 IDE for writing code. Though the interface may look different from your codespace, the behavior of the two environments should be largely similar!

CS50 Video Player

Usage

As well as running your queries in `sqlite3`, you can test your queries in the VS Code terminal by running

```
$ cat filename.sql | sqlite3 songs.db
```

where `filename.sql` is the file containing your SQL query.

Hints

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

▶ **Not sure how to solve?**

Spotify Wrapped

Spotify Wrapped (https://en.wikipedia.org/wiki/Spotify_Wrapped) is a feature presenting Spotify users' 100 most played songs from the past year. In 2021, Spotify Wrapped calculated an "Audio Aura" (https://newsroom.spotify.com/2021-12-01/learn-more-about-the-audio-aura-in-your-spotify-2021-wrapped-with-aura-reader-mystic-michaela/) for each user, a "reading of [their] two most prominent moods as dictated by [their] top songs and artists of the year." Suppose Spotify determines an audio aura by looking at the average energy, valence, and danceability of a person's top 100 songs from the past year. In `answers.txt`, reflect on the following questions:

- If `songs.db` contains the top 100 songs of one listener from 2018, how would you characterize their audio aura?
- Hypothesize about why the way you've calculated this aura might *not* be very representative of the listener. What better ways of calculating this aura would you propose?

Be sure to submit `answers.txt` along with each of your `.sql` files!

Testing

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

```
check50 cs50/labs/2023/x/songs
```

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/labs/2023/x/songs
```

Acknowledgements

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