Este é o CS50

Introdução à Ciência da Computação (CS50)

OpenCourseWare

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

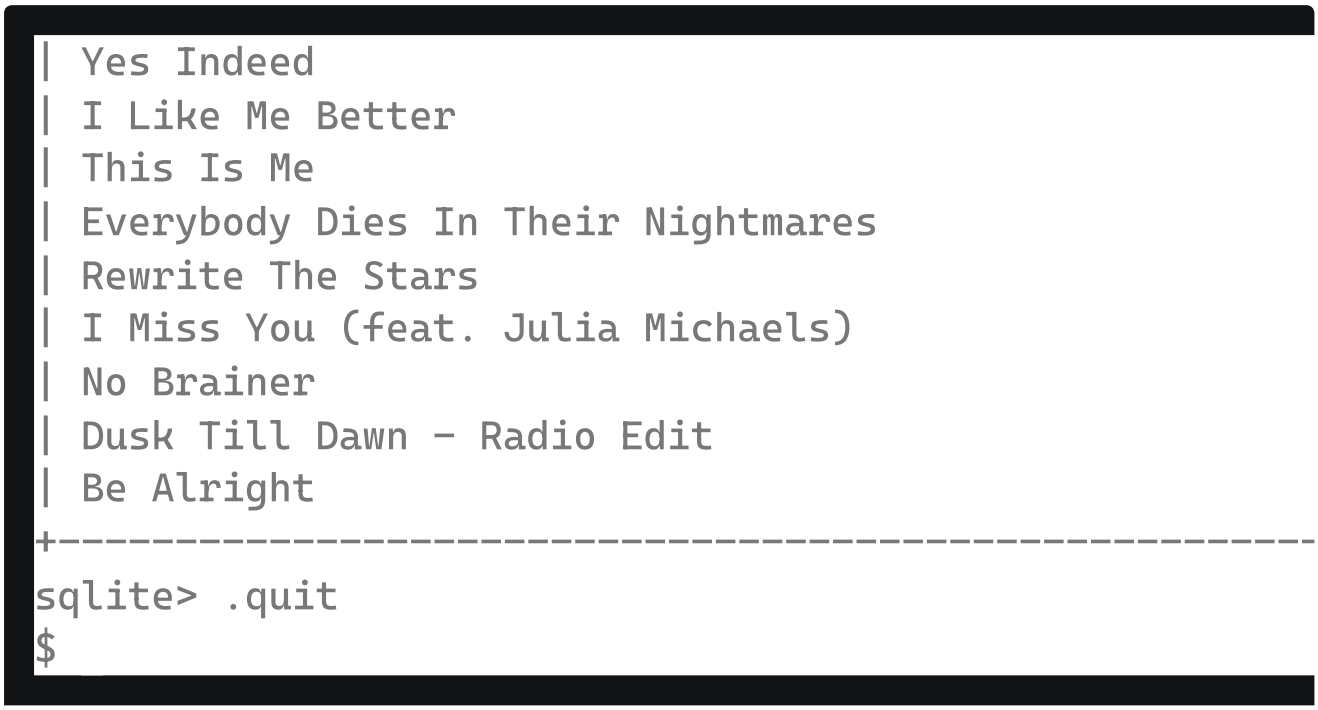
David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Músicas

Problema a resolver

Escreva consultas SQL para responder a perguntas sobre um banco de dados com as 100 músicas mais reproduzidas no Spotify (https://en.wikipedia.org/wiki/Spotify) em 2018.

Demonstração



Recorded with **asciinema**

Começando

Para resolver este problema, você usará um banco de dados fornecido pela equipe do CS50.

▼ **Baixe o código de distribuição.**

Abra o VS Code (https://cs50.dev/) .

Comece clicando dentro da janela do terminal e, em seguida, execute `cd` o comando. Você verá que o "prompt" será semelhante ao abaixo.

```
$
```

Clique dentro da janela do terminal e execute o seguinte comando.

```
wget https://cdn.cs50.net/2024/fall/psets/7/songs.zip
```

Em seguida, pressione Enter para baixar um arquivo ZIP chamado [nome do arquivo] `songs.zip` em seu espaço de código. Preste atenção ao espaço entre [nome do arquivo] `wget` e o URL a seguir, ou qualquer outro caractere!

Agora execute

```
unzip songs.zip
```

para criar uma pasta chamada `songs`. Você não precisa mais do arquivo ZIP, então pode executar

```
rm songs.zip
```

e responda com "y" seguido de Enter quando solicitado para remover o arquivo ZIP que você baixou.

Agora digite

```
cd songs
```

Em seguida, pressione Enter para entrar (ou seja, abrir) esse diretório. Seu prompt agora deve ser semelhante ao abaixo.

```
songs/ $
```

Se tudo correr bem, você deverá executar

```
ls
```

e você deverá ver 8 arquivos .sql, `songs.db`, e `answers.txt`.

Se você encontrar algum problema, siga esses mesmos passos novamente e veja se consegue determinar onde errou!

Entendimento

Você recebeu um arquivo chamado `songs.db`, um banco de dados SQLite que armazena dados do Spotify (https://developer.spotify.com/documentation/web-api/) sobre músicas e seus artistas. Este conjunto de dados contém as 100 músicas mais ouvidas no Spotify em 2018. Em uma janela de terminal, execute `sqlite3 songs.db` para que você possa começar a executar consultas no banco de dados.

Primeiro, quando `sqlite3` o sistema solicitar que você forneça uma consulta, digite-a `.schema` e pressione Enter. Isso exibirá as `CREATE TABLE` instruções usadas para gerar cada uma das tabelas no banco de dados. Ao examinar essas instruções, você poderá identificar as colunas presentes em cada tabela.

Observe que cada música `artist` tem um `topic` `id` e um `topic` `name`. Observe também que cada música tem um `topic` `name`, um `topic` `artist_id` (correspondente ao nome `id` do artista da música), bem como valores para dançabilidade, energia, tonalidade, volume, presença

de falas (presença de palavras faladas na faixa), valência, andamento e duração da música (medidos em milissegundos).

O desafio que você tem pela frente é escrever consultas SQL para responder a uma variedade de perguntas diferentes, selecionando dados de uma ou mais dessas tabelas. Depois disso, você refletirá sobre como o Spotify poderia usar esses mesmos dados em sua campanha anual Spotify Wrapped (https://en.wikipedia.org/wiki/Spotify_Wrapped) para caracterizar os hábitos dos ouvintes.

Detalhes da implementação

Para cada um dos problemas a seguir, você deve escrever uma única consulta SQL que retorne os resultados especificados. Sua resposta deve ser uma única consulta SQL, embora você possa aninhar outras consultas dentro dela. Você **não deve** presumir nada sobre os `id` tempos de músicas ou artistas específicos: suas consultas devem ser precisas mesmo que os tempos `id` de qualquer música ou pessoa sejam diferentes. Por fim, cada consulta deve retornar apenas os dados necessários para responder à pergunta: se o problema pedir apenas os nomes das músicas, por exemplo, sua consulta não deve retornar também o tempo de cada música.

1. Em `1.sql`, escreva uma consulta SQL para listar os nomes de todas as músicas no banco de dados.
    - Sua consulta deve retornar uma tabela com uma única coluna para o nome de cada música.
2. Em `2.sql`, escreva uma consulta SQL para listar os nomes de todas as músicas em ordem crescente de andamento (tempo).
    - Sua consulta deve retornar uma tabela com uma única coluna para o nome de cada música.
3. Em `3.sql`, escreva uma consulta SQL para listar os nomes das 5 músicas mais longas, em ordem decrescente de duração.
    - Sua consulta deve retornar uma tabela com uma única coluna para o nome de cada música.
4. Em `4.sql`, escreva uma consulta SQL que liste os nomes de todas as músicas que tenham dançabilidade, energia e valência maiores que 0,75.
    - Sua consulta deve retornar uma tabela com uma única coluna para o nome de cada música.
5. Em `5.sql`, escreva uma consulta SQL que retorne a energia média de todas as músicas.
    - Sua consulta deve retornar uma tabela com uma única coluna e uma única linha contendo a energia média.
6. Em `6.sql`, escreva uma consulta SQL que liste os nomes das músicas de Post Malone.
    - Sua consulta deve retornar uma tabela com uma única coluna para o nome de cada música.

    - Você não deve presumir nada sobre o que `artist_id` é Post Malone.
7. Em `7.sql`, escreva uma consulta SQL que retorne a energia média das músicas de Drake.
    - Sua consulta deve retornar uma tabela com uma única coluna e uma única linha contendo a energia média.
    - Você não deve presumir nada sobre o que `artist_id` é o Drake's.
8. Em `8.sql`, escreva uma consulta SQL que liste os nomes das músicas que contam com a participação de outros artistas.
    - Músicas que contam com a participação de outros artistas incluirão a expressão "feat." no título da música.
    - Sua consulta deve retornar uma tabela com uma única coluna para o nome de cada música.

Dicas

Consulte esta referência de palavras-chave SQL (https://www.w3schools.com/sql/sql_ref_keywords.asp) para obter algumas dicas de sintaxe SQL que podem ser úteis!

Clique nos botões abaixo para ler algumas dicas!

▼ **Liste os nomes de todas as músicas no banco de dados.**

Lembre-se de que, para selecionar todos os valores em uma coluna de uma tabela, você pode usar `SELECT` a palavra-chave do SQL. `SELECT` Ela é seguida pela coluna (ou colunas) que você gostaria de selecionar, que por sua vez é seguida pelo `FROM table` nome `table` da tabela da qual você gostaria de selecionar.

Em `1.sql` seguida, tente escrever o seguinte:

```
-- All songs in the database.
SELECT name
FROM songs;
```

▼ **Liste os nomes de todas as músicas em ordem crescente de andamento (ou andamento).**

Lembre-se de que o SQL possui uma `ORDER BY` palavra-chave que permite ordenar os resultados da sua consulta pelo valor de uma determinada coluna. Por exemplo, ` `ORDER BY tempo` ORDER BY 1` ordenará os resultados pela coluna `1` `tempo`.

Em `2.sql` seguida, tente escrever o seguinte:

```
-- All songs in increasing order of tempo.
SELECT name
FROM songs
ORDER BY tempo;
```

▼ **Liste os nomes das 5 músicas mais longas, em ordem decrescente de duração.**

Lembre-se de que `ORDER BY` nem sempre é necessário ordenar em ordem crescente. Você pode especificar que seus resultados sejam classificados em ordem *decrescente* adicionando um ponto `DESC`. Por exemplo, ` `ORDER BY duration_ms DESC` sort("duração")` listará os resultados em ordem decrescente.

E lembre-se também que `LIMIT n` você pode especificar que deseja apenas o primeiro.`n`linhas que correspondem a uma consulta específica. Por exemplo, `LIMIT 5` retornará apenas os cinco primeiros resultados da consulta.

Em `3.sql` seguida, tente escrever o seguinte:

```
-- The names of the top 5 longest songs, in descending order of length.
SELECT name
FROM songs
ORDER BY duration_ms DESC
LIMIT 5;
```

▼ **Liste os nomes de quaisquer músicas que tenham dançabilidade, energia e valência superiores a 0,75.**

Lembre-se de que você pode filtrar resultados em SQL com `WHERE` cláusulas, que são seguidas por alguma condição que normalmente testa os valores nas colunas de uma linha.

Lembre-se também de que os operadores do SQL funcionam de maneira muito semelhante aos do C. Por exemplo, ` `>` ORDER BY` retorna "verdadeiro" quando o valor à esquerda é maior que o valor à direita. Você pode encadear essas expressões, usando `ORDER BY` `AND` ou ` `OR` ORDER BY`, para formar uma condição maior.

Em `4.sql` seguida, tente escrever o seguinte:

```
-- The names of any songs that have danceability, energy, and valence greater than 0
SELECT name
FROM songs
WHERE danceability > 0.75 AND energy > 0.75 AND valence > 0.75;
```

▼ **Calcule a energia média de todas as músicas.**

Lembre-se de que o SQL suporta palavras-chave não apenas para selecionar linhas específicas, mas também para *agregar* os dados nessas linhas. Em particular, você pode achar a `AVG` palavra-chave `AUM` (para calcular médias) útil. Para agregar os resultados de uma coluna, basta aplicar a função de agregação a essa coluna. Por exemplo, ` `SELECT AVG(energy)` AUM` encontrará a média dos valores na coluna de energia para a consulta fornecida.

Em `5.sql` seguida, tente escrever o seguinte:

```
-- The average energy of all the songs.
SELECT AVG(energy)
FROM songs;
```

▼ **Liste os nomes das músicas de Post Malone.**

Observe que, se você executar o comando `.schema songs` no prompt do SQLite, a `songs` tabela contém os nomes das músicas, mas não os nomes dos artistas! Em vez disso, `songs` há uma `artist_id` coluna. Para listar os nomes das músicas de Post Malone, você precisará primeiro identificar o ID do artista Post Malone.

```
-- Identify Post Malone's artist id
SELECT id
FROM artists
WHERE name = 'Post Malone';
```

Essa consulta retorna 54. Agora, você pode consultar a `songs` tabela para qualquer música com o ID do Post Malone.

```
SELECT name
FROM songs
WHERE artist_id = 54;
```

No entanto, de acordo com a especificação, você deve ter cuidado para não presumir o conhecimento de nenhum ID. Você poderia melhorar o design desta consulta aninhando *suas* duas consultas.

Em `6.sql` seguida, tente escrever o seguinte:

```
-- The names of songs that are by Post Malone.
SELECT name
FROM songs
WHERE artist_id =
(
    SELECT id
    FROM artists
    WHERE name = 'Post Malone'
);
```

▼ **Encontre a energia média das músicas do Drake.**

Observe que, assim como na consulta anterior, você precisará combinar várias tabelas para executar esta consulta com sucesso. Você pode usar subconsultas aninhadas novamente, mas considere também outra abordagem!

Lembre-se de que você pode usar a palavra-chave `UNION` do SQL `JOIN` para combinar várias tabelas em uma só, desde que especifique quais colunas dessas tabelas devem corresponder. Por exemplo, a seguinte consulta une as tabelas ` songs A` e ` artists B`, indicando que a

`artist_id` coluna `A` na `songs` tabela `B` e a `id` coluna `B` na `artists` tabela `C` devem corresponder:

```
SELECT *
FROM songs
JOIN artists ON songs.artist_id = artists.id
```

Com essas duas tabelas combinadas, basta filtrar sua seleção para encontrar a energia média das músicas do Drake.

Em `7.sql` seguida, tente escrever o seguinte:

```
-- The average energy of songs that are by Drake
SELECT AVG(energy)
FROM songs
JOIN artists ON songs.artist_id = artists.id
WHERE artists.name = 'Drake';
```

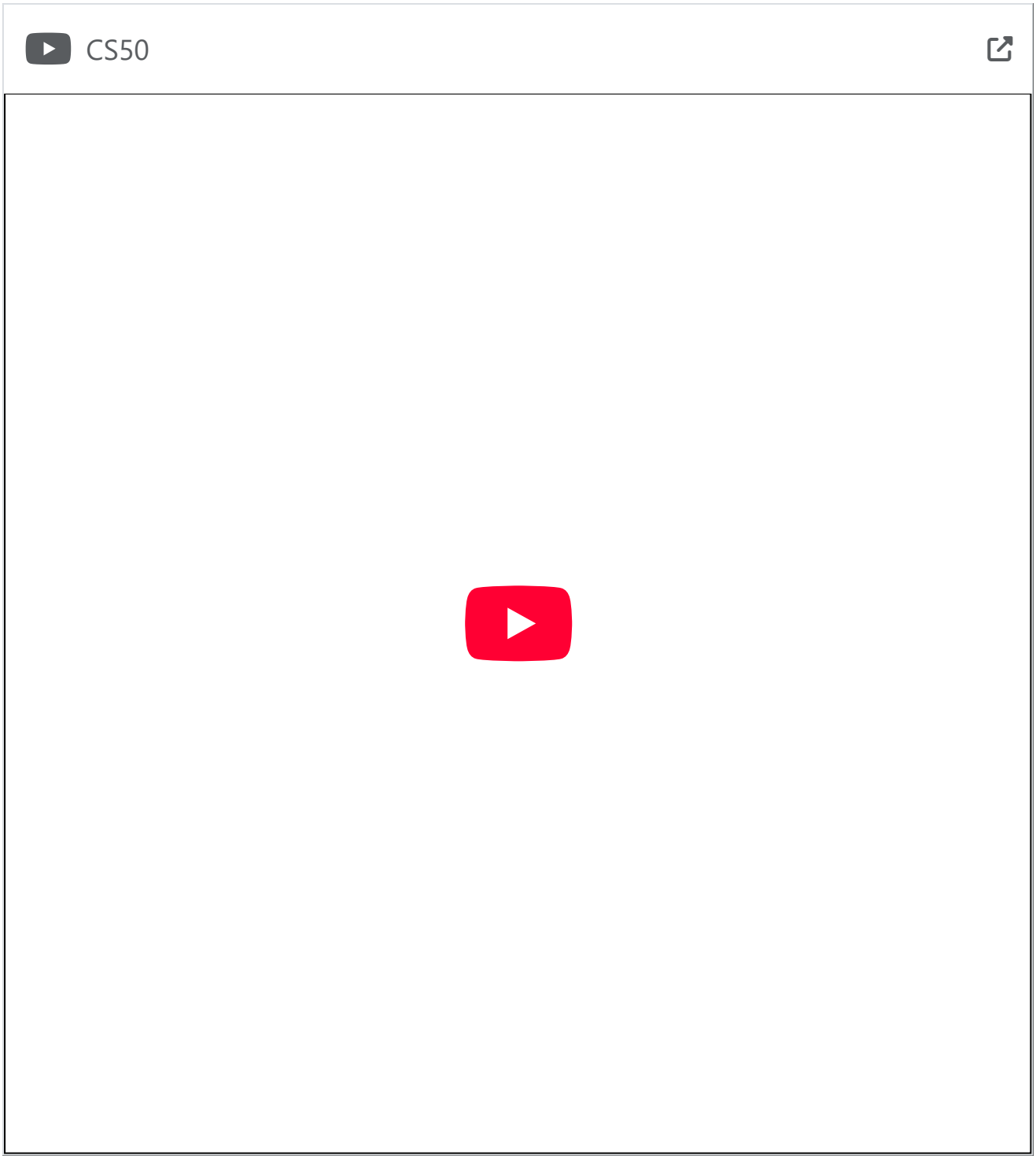
▼ **Liste os nomes das músicas que contam com a participação de outros artistas.**

Para esta consulta, observe que músicas com participação de outros artistas geralmente contêm a palavra "feat." no título. Lembre-se de que `LIKE` a palavra-chave do SQL pode ser usada para encontrar correspondências com determinadas frases (como "feat."!). Para isso, você pode usar `%` um caractere curinga que corresponde a qualquer sequência de caracteres.

Em `8.sql` seguida, tente escrever o seguinte:

```
-- The names of songs that feature other artists.
SELECT name
FROM songs
WHERE name LIKE '%feat.%';
```

Passo a passo



▶ **Não sabe como resolver?**

Spotify Wrapped

O Spotify Wrapped (https://en.wikipedia.org/wiki/Spotify_Wrapped) é um recurso que apresenta as 100 músicas mais tocadas pelos usuários do Spotify no último ano. Em 2021, o Spotify Wrapped calculou uma "Aura Sonora" (https://newsroom.spotify.com/2021-12-01/learn-more-about-the-audio-aura-in-your-spotify-2021-wrapped-with-aura-reader-mystic-michaela/) para cada usuário, uma "leitura de seus dois humores mais proeminentes, ditados por suas músicas e artistas mais ouvidos no ano". Suponha que o Spotify determine uma aura sonora analisando a

energia, a valência e a dançabilidade médias das 100 músicas mais ouvidas por uma pessoa no último ano. Em 2021 `answers.txt`, reflita sobre as seguintes questões:

- Se `songs.db` a lista contivesse as 100 músicas mais ouvidas de um ouvinte em 2018, como você caracterizaria a aura sonora dessa pessoa?
- Elabore uma hipótese sobre por que a maneira como você calculou essa aura pode *não* ser muito representativa do ouvinte. Que maneiras melhores você proporia para calcular essa aura?

Não se esqueça de enviar `answers.txt` junto com todos os seus `.sql` arquivos!

Como testar

Correção

```
check50 cs50/problems/2025/x/songs
```

Como enviar

```
submit50 cs50/problems/2025/x/songs
```

Agradecimentos

Conjunto de dados do Kaggle (https://www.kaggle.com/nadintamer/top-spotify-tracks-of-2018)
.