Isso é CS50

Introdução do CS50 à Ciência da Computação

OpenCourseWare

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

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.clubhouse.com/@davidjmalan) (https://www.facebook.com/dmalan)
(https://github.com/dmalan) (https://www.instagram.com/davidjmalan/)
(https://www.linkedin.com/in/malan/) (https://orcid.org/0000-0001-5338-2522)
(https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan)
(https://www.tiktok.com/@davidjmalan) (https://davidjmalan.t.me/)
(https://twitter.com/davidjmalan)

Movies

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

Getting Started

Log into code.cs50.io (https://code.cs50.io/), click on your terminal window, and execute `cd` by itself. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
wget https://cdn.cs50.net/2022/fall/psets/7/movies.zip
```

in order to download a ZIP called `movies.zip` into your codespace.

Then execute

```
unzip movies.zip
```

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

```
rm movies.zip
```

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

Now type

```
cd movies
```

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

```
movies/ $
```

Execute `ls` by itself, and you should see 13 .sql files, as well as `movies.db`.

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 `movies.db`, a SQLite database that stores data from IMDb (https://www.imdb.com/) about movies, the people who directed and starred in them, and their ratings. In a terminal window, run `sqlite3 movies.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 the `movies` table has an `id` column that uniquely identifies each movie, as well as columns for the `title` of a movie and the `year` in which the movie was released. The `people` table also has an `id` column, and also has columns for each person’s `name` and `birth` year.

Movie ratings, meanwhile, are stored in the `ratings` table. The first column in the table is `movie_id`: a foreign key that references the `id` of the `movies` table. The rest of the row contains data about the `rating` for each movie and the number of `votes` the movie has received on IMDb.

Finally, the `stars` and `directors` tables match people to the movies in which they acted or directed. (Only principal (https://www.imdb.com/interfaces/) stars and directors are included.) Each table has just two columns: `movie_id` and `person_id`, which reference a specific movie and person, respectively.

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.

Specification

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 movies or people: your queries should be accurate even if the `id` of any particular movie 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 movies, for example, then your query should not also output each movie’s release year.

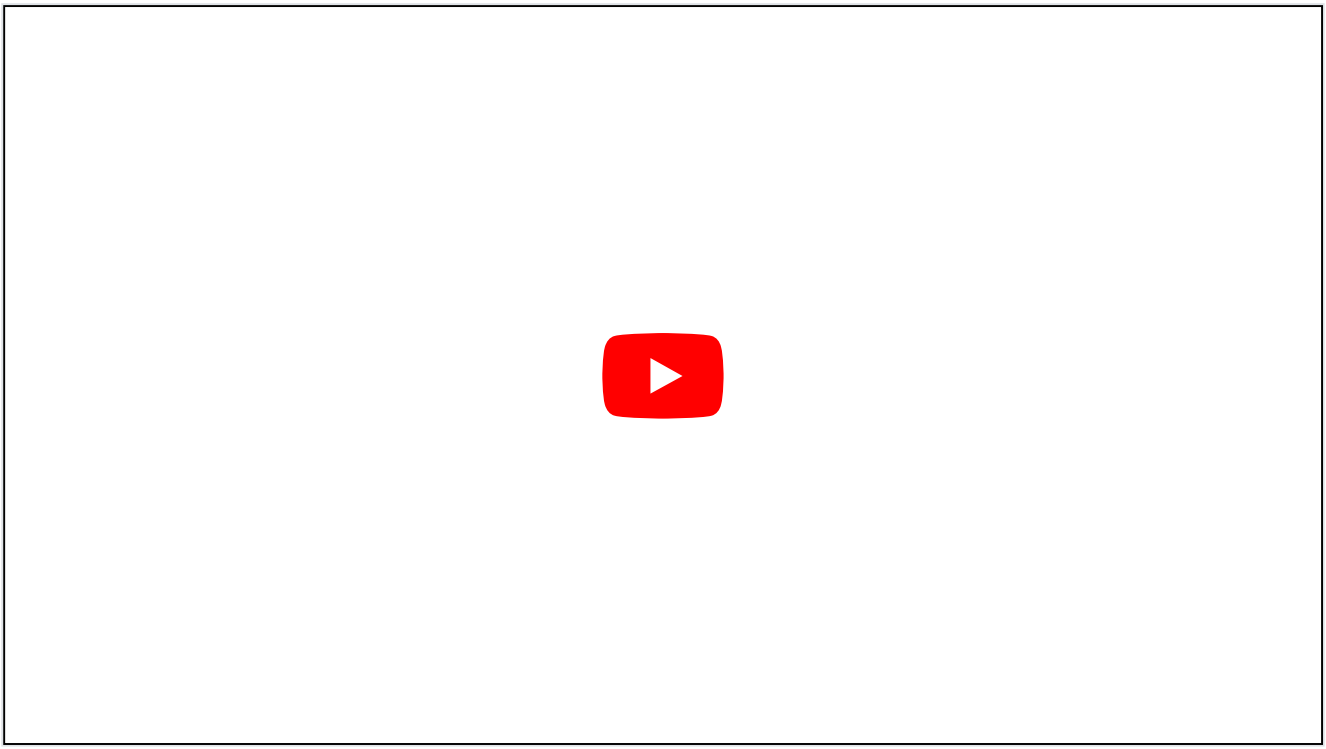
Você pode verificar os resultados de suas consultas com o próprio IMDb (https://www.imdb.com/) , mas saiba que as avaliações no site podem diferir daquelas em `movies.db`, pois mais votos podem ter sido dados desde que baixamos os dados!

1. Em `1.sql`, escreva uma consulta SQL para listar os títulos de todos os filmes lançados em 2008.
   - Sua consulta deve gerar uma tabela com uma única coluna para o título de cada filme.
2. Em `2.sql`, escreva uma consulta SQL para determinar o ano de nascimento de Emma Stone.
   - Sua consulta deve gerar uma tabela com uma única coluna e uma única linha (sem contar o cabeçalho) contendo o ano de nascimento de Emma Stone.
   - Você pode presumir que há apenas uma pessoa no banco de dados com o nome de Emma Stone.
3. No `3.sql`, escreva uma consulta SQL para listar os títulos de todos os filmes com data de lançamento igual ou posterior a 2018, em ordem alfabética.
   - Sua consulta deve gerar uma tabela com uma única coluna para o título de cada filme.
   - Os filmes lançados em 2018 devem ser incluídos, assim como os filmes com datas de lançamento no futuro.
4. Em `4.sql`, escreva uma consulta SQL para determinar o número de filmes com classificação IMDb de 10,0.
   - Sua consulta deve gerar uma tabela com uma única coluna e uma única linha (sem contar o cabeçalho) contendo o número de filmes com classificação 10,0.
5. Em `5.sql`, escreva uma consulta SQL para listar os títulos e os anos de lançamento de todos os filmes de Harry Potter, em ordem cronológica.
   - Sua consulta deve gerar uma tabela com duas colunas, uma para o título de cada filme e outra para o ano de lançamento de cada filme.
   - Você pode presumir que o título de todos os filmes de Harry Potter começará com as palavras "Harry Potter" e que, se o título de um filme começar com as palavras "Harry Potter", é um filme de Harry Potter.
6. Em `6.sql`, escreva uma consulta SQL para determinar a classificação média de todos os filmes lançados em 2012.
   - Sua consulta deve gerar uma tabela com uma única coluna e uma única linha (sem contar o cabeçalho) contendo a classificação média.
7. Em `7.sql`, escreva uma consulta SQL para listar todos os filmes lançados em 2010 e suas classificações, em ordem decrescente de classificação. Para filmes com a mesma classificação, ordene-os alfabeticamente por título.
   - Sua consulta deve gerar uma tabela com duas colunas, uma para o título de cada filme e outra para a classificação de cada filme.
   - Filmes que não possuem classificações não devem ser incluídos no resultado.
8. Em `8.sql`, escreva uma consulta SQL para listar os nomes de todas as pessoas que estrelaram Toy Story.
   - Sua consulta deve gerar uma tabela com uma única coluna para o nome de cada pessoa.
   - Você pode presumir que há apenas um filme no banco de dados com o título Toy Story.
9. Em `9.sql`, escreva uma consulta SQL para listar os nomes de todas as pessoas que estrelaram um filme lançado em 2004, ordenados por ano de nascimento.
   - Sua consulta deve gerar uma tabela com uma única coluna para o nome de cada pessoa.
   - Pessoas com o mesmo ano de nascimento podem ser listadas em qualquer ordem.

- Não há necessidade de se preocupar com pessoas que não têm ano de nascimento listado, desde que aquelas que tenham um ano de nascimento estejam listadas em ordem.
- Se uma pessoa apareceu em mais de um filme em 2004, ela deve aparecer apenas uma vez em seus resultados.

10. Em `10.sql`, escreva uma consulta SQL para listar os nomes de todas as pessoas que dirigiram um filme que recebeu uma classificação de pelo menos 9,0.
    - Sua consulta deve gerar uma tabela com uma única coluna para o nome de cada pessoa.
    - Se uma pessoa dirigiu mais de um filme que recebeu uma classificação de pelo menos 9,0, ela deve aparecer apenas uma vez em seus resultados.
11. Em `11.sql`, escreva uma consulta SQL para listar os títulos dos cinco filmes com classificação mais alta (em ordem) estrelados por Chadwick Boseman, começando com o de classificação mais alta.
    - Sua consulta deve gerar uma tabela com uma única coluna para o título de cada filme.
    - Você pode presumir que há apenas uma pessoa no banco de dados com o nome de Chadwick Boseman.
12. Em `12.sql`, escreva uma consulta SQL para listar os títulos de todos os filmes estrelados por Johnny Depp e Helena Bonham Carter.
    - Sua consulta deve gerar uma tabela com uma única coluna para o título de cada filme.
    - Você pode presumir que há apenas uma pessoa no banco de dados com o nome de Johnny Depp.
    - Você pode presumir que há apenas uma pessoa no banco de dados com o nome de Helena Bonham Carter.
13. Em `13.sql`, escreva uma consulta SQL para listar os nomes de todas as pessoas que estrelaram um filme no qual Kevin Bacon também estrelou.
    - Sua consulta deve gerar uma tabela com uma única coluna para o nome de cada pessoa.
    - Pode haver várias pessoas chamadas Kevin Bacon no banco de dados. Certifique-se de selecionar apenas o Kevin Bacon nascido em 1958.
    - O próprio Kevin Bacon não deve ser incluído na lista resultante.

Passo a passo



Uso

Para testar suas consultas no VS Code, você pode consultar o banco de dados executando

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

onde `filename.sql` está o arquivo que contém sua consulta SQL.

Você também pode correr

```
$ cat filename.sql | sqlite3 movies.db > output.txt
```

para redirecionar a saída da consulta para um arquivo de texto chamado `output.txt`. (Isso pode ser útil para verificar quantas linhas são retornadas por sua consulta!)

dicas

- Consulte esta referência de palavras-chave SQL (https://www.w3schools.com/sql/sql_ref_keywords.asp) para obter alguma sintaxe SQL que pode ser útil!
- Consulte sqlstyle.guide (https://www.sqlstyle.guide/) para obter indicações sobre um bom estilo em SQL, especialmente à medida que suas consultas se tornam mais complexas!

teste

Embora `check50` esteja disponível para esse problema, você é encorajado a testar seu código por conta própria para cada um dos itens a seguir. Você pode executar `sqlite3 movies.db` consultas adicionais no banco de dados para garantir que seu resultado esteja correto.

Se estiver usando o `movies.db` banco de dados fornecido na distribuição deste conjunto de problemas, você deve descobrir que

- A execução `1.sql` resulta em uma tabela com 1 coluna e 10.050 linhas.
- A execução `2.sql` resulta em uma tabela com 1 coluna e 1 linha.
- A execução `3.sql` resulta em uma tabela com 1 coluna e 88.918 linhas.
- A execução `4.sql` resulta em uma tabela com 1 coluna e 1 linha.
- A execução `5.sql` resulta em uma tabela com 2 colunas e 12 linhas.
- A execução `6.sql` resulta em uma tabela com 1 coluna e 1 linha.
- A execução `7.sql` resulta em uma tabela com 2 colunas e 7.085 linhas.
- A execução `8.sql` resulta em uma tabela com 1 coluna e 4 linhas.
- A execução `9.sql` resulta em uma tabela com 1 coluna e 18.946 linhas.
- A execução `10.sql` resulta em uma tabela com 1 coluna e 3.392 linhas.
- A execução `11.sql` resulta em uma tabela com 1 coluna e 5 linhas.
- A execução `12.sql` resulta em uma tabela com 1 coluna e 7 linhas.
- A execução `13.sql` resulta em uma tabela com 1 coluna e 182 linhas.

Observe que as contagens de linhas não incluem linhas de cabeçalho que mostram apenas nomes de colunas.

Se sua consulta retornar um número de linhas ligeiramente diferente da saída esperada, verifique se você está lidando corretamente com duplicatas! Para consultas que solicitam uma lista de nomes, nenhuma pessoa deve ser listada duas vezes, mas duas pessoas diferentes com o mesmo nome devem ser listadas.

Execute o abaixo para avaliar a exatidão do seu código usando `check50`.

```
check50 cs50/problems/2023/x/movies
```

Como enviar

Em seu terminal, execute o abaixo para enviar seu trabalho.

```
submit50 cs50/problems/2023/x/movies
```

Reconhecimentos

Informação cortesia do IMDb (imdb.com (https://www.imdb.com)). Usado com permissão.