CS50's Introduction to Databases with SQL

OpenCourseWare

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

Carter Zenke (https://carterzenke.me)
carter@cs50.harvard.edu
(https://github.com/carterzenke) (https://www.linkedin.com/in/carterzenke/)

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)

Cyberchase

Problem to Solve

Welcome to Cyberspace! *Cyberchase* (https://pbskids.org/cyberchase/) is an animated, educational kid's television series, aired by the United States' Public Broadcasting Service (PBS) (https://www.pbs.org/) since 2002. Originally designed to "show kids that math is everywhere and everyone can be good at it," (https://www.pbs.org/parents/shows/cyberchase/about) the world of *Cyberchase* centers on Jackie, Matt, and Inez as they team up with Digit—a "cybird"—to stop Hacker from taking over Cyberspace and infecting Motherboard. Along the way, the quartet learn math, science, and problem-solving skills to thwart Hacker in his attempts.

In a database called `cyberchase.db`, using a table called `episodes`, chase answers to PBS's questions about *Cyberchase*'s episodes thus far.

Demo

Recorded with asciinema

Distribution Code

For this problem, you'll need to download `cyberchase.db`, along with several `.sql` files in which you'll write your queries.

▶ **Download the distribution code**

Schema

Each database has some "schema"—the tables and columns into which the data is organized. In `cyberchase.db` you'll find a single table, `episodes`. In the `episodes` table, you'll find the following columns:

- `id`, which uniquely identifies each row (episode) in the table
- `season`, which is the season number in which the episode aired
- `episode_in_season`, which is the episode's number within its given season
- `title`, which is the episode's title
- `topic`, which identifies the ideas the episode aimed to teach
- `air_date`, which is the date (expressed as `YYYY-MM-DD`) on which the episode "aired" (i.e., was published)
- `production_code`, which is the unique ID used by PBS to refer to each episode internally

Specification

For each of the following questions, 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. You **should not** assume anything about the `id`s of any particular episodes: your queries should be accurate even if the `id` of any particular episode 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 episodes, for example, then your query should not also output each episodes's air date.

1. In `1.sql`, write a SQL query to list the titles of all episodes in *Cyberchase*'s original season, Season 1.
2. In `2.sql`, list the season number of, and title of, the first episode of every season.
3. In `3.sql`, find the production code for the episode "Hackerized!".
4. In `4.sql`, write a query to find the titles of episodes that do not yet have a listed topic.
5. In `5.sql`, find the title of the holiday episode that aired on December 31st, 2004.
6. In `6.sql`, list the titles of episodes from season 6 (2008) that were released early, in 2007.
7. In `7.sql`, write a SQL query to list the titles and topics of all episodes teaching fractions.
8. In `8.sql`, write a query that counts the number of episodes released in the last 6 years, from 2018 to 2023, inclusive.
    - You might find it helpful to know you can use `BETWEEN` with dates, such as `BETWEEN '2000-01-01' AND '2000-12-31'`.
9. In `9.sql`, write a query that counts the number of episodes released in *Cyberchase*'s first 6 years, from 2002 to 2007, inclusive.
10. In `10.sql`, write a SQL query to list the ids, titles, and production codes of all episodes. Order the results by *production code*, from earliest to latest.
11. In `11.sql`, list the titles of episodes from season 5, in reverse alphabetical order.
12. In `12.sql`, count the number of unique episode titles.
13. In `13.sql`, write a SQL query to explore a question of your choice. This query should:
    - Involve at least one condition, using `WHERE` with `AND` or `OR`

▶ **Feeling more comfortable?**

Usage

To test your queries as you write them in your `.sql` files, you can query the database by running

```
.read FILENAME
```

where `FILENAME` is the name of the file containing your SQL query. For example,

```
.read 1.sql
```

You can also run (in your terminal, rather than inside `sqlite3`)

```
$ cat FILENAME | sqlite3 cyberchase.db > output.txt
```

to redirect the output of the query to a text file called `output.txt`. (This can be useful for checking how many rows are returned by your query!)

How to Test

While `check50` is available for this problem (see below), you're encouraged to instead test your code on your own for each of the following. If you're using the `cyberchase.db` database provided in this problem's distribution, you should find that…

- Executing `1.sql` results in a table with 1 column and 26 rows.
- Executing `2.sql` results in a table with 2 columns and 14 rows.
- Executing `3.sql` results in a table with 1 column and 1 row.
- Executing `4.sql` results in a table with 1 column and 26 rows.
- Executing `5.sql` results in a table with 1 column and 1 row.
- Executing `6.sql` results in a table with 1 column and 2 rows.
- Executing `7.sql` results in a table with 2 columns and 6 rows.
- Executing `8.sql` results in a table with 1 column and 1 row.
- Executing `9.sql` results in a table with 1 column and 1 row.
- Executing `10.sql` results in a table with 3 columns and 140 rows.
- Executing `11.sql` results in a table with 1 column and 10 rows.
- Executing `12.sql` results in a table with 1 column and 1 row.

`13.sql` is up to you!

Note that row counts do not include header rows that only show column names.

Correctness

You can also check your code using `check50`, a program that CS50 will use to test your code when you submit. Type the following at your terminal $ prompt.

```
check50 cs50/problems/2023/sql/cyberchase
```

Green smilies mean your program has passed a test! Red frownies will indicate your program output something unexpected. Visit the URL that `check50` outputs to see the input `check50` handed to your program, what output it expected, and what output your program actually gave.

How to Submit

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

```
submit50 cs50/problems/2023/sql/cyberchase
```

Acknowledgements

Data retrieved from en.wikipedia.org/wiki/List_of_Cyberchase_episodes (https://en.wikipedia.org/wiki/List_of_Cyberchase_episodes).