


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>)

Players



The 1901 Boston Americans (now Red Sox) Team

Problem to Solve

If you're not familiar, baseball is a popular sport in which two teams of 9 players take turns batting (hitting a ball) and fielding (catching and throwing hit balls). Points ("runs") are scored when a hitting team's player hits a ball and eventually touches all bases before the fielding team's players have the chance to get them "out." Baseball is arguably most popular in the United States and Canada, where the [MLB \(Major League Baseball\)](https://en.wikipedia.org/wiki/Major_League_Baseball) (https://en.wikipedia.org/wiki/Major_League_Baseball) has served as the professional association for players since 1876.

In a database called `players.db`, using a table called `players`, answer questions about MLB players who've played from 1871 to 2023.

Demo

```
$ sqlite3 players.db
sqlite> .tables
players
sqlite> SELECT "first_name", "last_name", "debut"
...>
```

Recorded with [asciinema](#)

Distribution Code

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

▼ Download the distribution code

Log into cs50.dev (<https://cs50.dev/>), 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/sql/2023/x/psets/0/players.zip
```

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

Then execute

```
unzip players.zip
```

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

```
rm players.zip
```

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

Now type

```
cd players
```

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

```
players/ $
```

If all was successful, you should execute

```
ls
```

and see a database named `players.db` alongside several `.sql` files. Executing `sqlite3 players.db` should open the database in `sqlite3`, via which you'll execute SQL queries. If not, retrace your steps and see if you can determine where you went wrong!

Schema

In `players.db` you'll find a single table, `players`. In the `players` table, you'll find the following columns:

- `id`, which uniquely identifies each row (player) in the table
- `first_name`, which is the first name of the player
- `last_name`, which is the last name of the player
- `bats`, which is the side (`R` for right or `L` for left) the player bats on
- `throws`, which is the hand (`R` for right or `L` for left) the player throws with
- `weight`, which is the player's weight in pounds
- `height`, which is the player's height in inches
- `debut`, which is the date (expressed as `YYYY-MM-DD`) the player began their career in the MLB
- `final_game`, which is the date (expressed as `YYYY-MM-DD`) the player played their last game in the MLB
- `birth_year`, which is the year the player was born
- `birth_month`, which is the month (expressed as an integer) the player was born
- `birth_day`, which is the day the player was born
- `birth_city`, which is the city in which the player was born
- `birth_state`, which is the state in which the player was born

- `birth_country`, which is the country in which the player was born

Specification

1. In `1.sql`, write a SQL query to find the hometown (including city, state, and country) of Jackie Robinson.
2. In `2.sql`, write a SQL query to find the side (e.g., right or left) Babe Ruth hit.
3. In `3.sql`, write a SQL query to find the ids of rows for which a value in the column `debut` is missing.
4. In `4.sql`, write a SQL query to find the first and last names of players who were *not* born in the United States. Sort the results alphabetically by first name, then by last name.
5. In `5.sql`, write a SQL query to return the first and last names of all right-handed batters. Sort the results alphabetically by first name, then by last name.
6. In `6.sql`, write a SQL query to return the first name, last name, and debut date of players born in Pittsburgh, Pennsylvania (PA). Sort the results first by debut date—from most recent to oldest—then alphabetically by first name, followed by last name.
7. In `7.sql`, write a SQL query to count the number of players who bat (or batted) right-handed and throw (or threw) left-handed, *or vice versa*.
8. In `8.sql`, write a SQL query to find the average height and weight, rounded to two decimal places, of baseball players who debuted on or after January 1st, 2000. Return the columns with the name “Average Height” and “Average Weight”, respectively.
9. In `9.sql`, write a SQL query to find the players who played their final game in the MLB in 2022. Sort the results alphabetically by first name, then by last name.
10. In `10.sql`, write SQL query to answer a question of your choice. This query should:
 - Make use of `AS` to rename a column
 - Involve at least condition, using `WHERE`
 - Sort by at least one column using `ORDER BY`

► 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

```
$ cat FILENAME | sqlite3 players.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, you're encouraged to instead test your code on your own for each of the following. If you're using the `players.db` database provided in this problem's distribution, you should find that...

- Executing `1.sql` results in a table with 3 columns and 1 row.
- Executing `2.sql` results in a table with 1 column and 1 row.
- Executing `3.sql` results in a table with 1 column and 213 rows.
- Executing `4.sql` results in a table with 2 columns and 2814 rows.
- Executing `5.sql` results in a table with 2 columns and 12878 row.
- Executing `6.sql` results in a table with 3 columns and 134 rows.
- Executing `7.sql` results in a table with 1 columns and 1 row.
- Executing `8.sql` results in a table with 2 columns and 1 row.
- Executing `9.sql` results in a table with 2 columns and 516 rows.

`10.sql` is up to you!

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

Correctness

```
check50 cs50/problems/2023/sql/players
```

How to Submit

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

```
submit50 cs50/problems/2023/sql/players
```

Acknowledgements

Data retrieved and modified from github.com/chadwickbureau/baseballdatabank/tree/master (<https://github.com/chadwickbureau/baseballdatabank/tree/master>), licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License (<http://creativecommons.org/licenses/by-sa/3.0/>).