Don't Panic!

"A single desktop computer sitting on a desk in an empty office at night, in the style of a detective noir," generated by DALL·E 2

Problem to Solve

You're a trained "pentester." Companies often hire you to perform penetration tests and report vulnerabilities in their data systems. Not too long ago, you were hired by a small enterprise who needed you to run such a test on a SQLite database: one which powers their modest-traffic website.

To succeed in this covert operation, you'll need to…

- **Alter** the password of the website's administrative account.
- **Erase** any logs of the above password change recorded by the database.
- **Add** false data to throw the company off of your trail.

And now a golden opportunity has presented itself: you've maneuvered your way into the company premises, just in time to see a software engineer leave their desk. The engineer's connection to the database is still open. You estimate you have 5 minutes before they come back. Ready?

Demo

Recorded with **asciinema**

Distribution Code

▼ **Download the distribution code**

Log into 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/2024/x/psets/3/dont-panic.zip
```

in order to download a ZIP called `dont-panic.zip` into your codespace.

Then execute

```
unzip dont-panic.zip
```

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

```
rm dont-panic.zip
```

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

Now type

```
cd dont-panic
```

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

```
dont-panic/ $
```

If all was successful, you should execute

```
ls
```

and see a database named `dont-panic.db` alongside a `hack.sql` and `reset.sql` file. Executing `sqlite3 dont-panic.db` should open the database. If not, retrace your steps and see if you can determine where you went wrong!

Schema

Afraid there's not much time to explain the database's schema. Remember you can access a SQLite database's schema with `.schema`.

Specification

In `hack.sql`, write a sequence of SQL statements to achieve the following:

- **Alter** the password of the website's administrative account, `admin`, to instead be "oops!".
- **Erase** any logs of the above password change recorded by the database.
- **Add** false data to throw others off your trail. In particular, to frame `emily33`, make it only *appear*—in the `user_logs` table—as if the `admin` account has had its password changed to `emily33`'s password.

When your SQL statements in `hack.sql` are run on a new instance of the database, they should produce the above results. Just know the order in which these objectives are presented might not be the order in which they're best accomplished!

Also keep in mind that passwords are usually not stored "in the clear"—that is, as the plain characters that make up the password. Instead they're "hashed," or scrambled, to preserve privacy. Given this reality, you'll need to ensure the password to which you change the administrative password is also hashed. Thankfully, you know that the passwords in the `users` table are already stored as MD5 hashes. You can generate quickly generate such hashes from plaintext at md5hashgenerator.com.

Clock's ticking!

▼ **Hints**

- Recall you can `INSERT` into a table the rows returned by a `SELECT` statement, so long as the number of columns matches.
- You can create a subquery at any point in a SQL statement, not just as part of a `WHERE` clause. For instance, consider the following SQL query on a simplified `user_logs` table:

  ```
  INSERT INTO "user_logs" ("type", "password")
  SELECT 'update', (
      SELECT "password"
      FROM "users"
      WHERE "username" = 'carter'
  );
  ```

  The above query will insert a new row into the `user_logs` table. The column `type` will have the value "update" and the column `password` will have the current password of the user `carter`.

Usage

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

```
.read hack.sql
```

If you need to reset the database at any time, you can run

```
.read reset.sql
```

to return the database to its original form.

How to Test

Correctness

Execute the below to evaluate the correctness of your findings using `check50`

```
check50 cs50/problems/2024/sql/dont-panic
```

How to Submit

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

```
submit50 cs50/problems/2024/sql/dont-panic
```