Homework 9 Solutions **hw09.zip (hw09.zip)**

Solution Files

You can find the solutions in the hw09.sql (hw09.sql) file.

Usage

First, check that a file named `sqlite_shell.py` exists alongside the assignment files. If you don't see it, or if you encounter problems with it, scroll down to the Troubleshooting section to see how to download an official precompiled SQLite binary before proceeding.

You can start an interactive SQLite session in your Terminal or Git Bash with the following command:

```
python3 sqlite_shell.py
```

While the interpreter is running, you can type `.help` to see some of the commands you can run.

To exit out of the SQLite interpreter, type `.exit` or `.quit` or press `Ctrl-C`. Remember that if you see `...>` after pressing enter, you probably forgot a `;`.

You can also run all the statements in a `.sql` file by doing the following:

1. Runs your code and then exits SQLite immediately afterwards.

   ```
   python3 sqlite_shell.py < lab13.sql
   ```

2. Runs your code and then opens an interactive SQLite session, which is similar to running Python code with the interactive `-i` flag.

   ```
   python3 sqlite_shell.py --init lab13.sql
   ```

To complete this homework assignment, you will need to use SQLite version 3.8.3 or greater.

To check your progress, you can run `sqlite3` directly by running:

```
python3 sqlite_shell.py --init hw09.sql
```

You should also check your work using `ok`:

```
python3 ok
```

Questions (Required)

Dog Data

In each question below, you will define a new table based on the following tables.

```
CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"          , "clinton"         UNION
  SELECT "delano"           , "herbert"         UNION
  SELECT "fillmore"         , "abraham"         UNION
  SELECT "fillmore"         , "delano"          UNION
  SELECT "fillmore"         , "grover"          UNION
  SELECT "eisenhower"       , "fillmore";

CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur, 26 AS height UNION
  SELECT "barack"         , "short"      , 52           UNION
  SELECT "clinton"        , "long"       , 47           UNION
  SELECT "delano"         , "long"       , 46           UNION
  SELECT "eisenhower"     , "short"      , 35           UNION
  SELECT "fillmore"       , "curly"      , 32           UNION
  SELECT "grover"         , "short"      , 28           UNION
  SELECT "herbert"        , "curly"      , 31;

CREATE TABLE sizes AS
  SELECT "toy" AS size, 24 AS min, 28 AS max UNION
  SELECT "mini"       , 28       , 35        UNION
  SELECT "medium"     , 35       , 45        UNION
  SELECT "standard"   , 45       , 60;
```

Your tables should still perform correctly even if the values in these tables change. For example, if you are asked to list all dogs with a name that starts with h, you should write:

```
SELECT name FROM dogs WHERE "h" <= name AND name < "i";
```

Instead of assuming that the `dogs` table has only the data above and writing

```
SELECT "herbert";
```

The former query would still be correct if the name `grover` were changed to `hoover` or a row was added with the name `harry`.

Copy

Copy your code from hw08 (/hw/hw08) for `size_of_dogs` to be able to more easily to do then next problems!

Q1: By Parent Height

Create a table `by_parent_height` that has a column of the names of all dogs that have a `parent`, ordered by the height of the parent from tallest parent to shortest parent.

```
-- All dogs with parents ordered by decreasing height of their parent
CREATE TABLE by_parent_height AS
  SELECT child FROM parents, dogs WHERE name = parent ORDER BY -height;
```

For example, `fillmore` has a parent (`eisenhower`) with height 35, and so should appear before `grover` who has a parent (`fillmore`) with height 32. The names of dogs with parents of the same height should appear together in any order. For example, `barack` and `clinton` should both appear at the end, but either one can come before the other.

```
sqlite> select * from by_parent_height;
herbert
fillmore
abraham
delano
grover
barack
clinton
```

Use Ok to test your code:

```
python3 ok -q by_parent_height
```

We need information from both the `parents` and the `dogs` table. This time, the only rows that make sense are the ones where a child is matched up with their parent. Finally, we order the result by descending height.

Q2: Sentences

There are two pairs of siblings that have the same size. Create a table that contains a row with a string for each of these pairs. Each string should be a sentence describing the siblings by their size.

```
-- Filling out this helper table is optional
CREATE TABLE siblings AS
  SELECT a.child AS first, b.child AS second FROM parents AS a, parents AS b
    WHERE a.parent = b.parent AND a.child < b.child;

-- Sentences about siblings that are the same size
CREATE TABLE sentences AS
  SELECT first || " and " || second || " are " || a.size || " siblings"
    FROM siblings, size_of_dogs AS a, size_of_dogs AS b
    WHERE a.size = b.size AND a.name = first AND b.name = second;
```

Each sibling pair should appear only once in the output, and siblings should be listed in alphabetical order (e.g. `"barack and clinton..."` instead of `"clinton and barack..."`), as follows:

```
sqlite> select * from sentences;
abraham and grover are toy siblings
barack and clinton are standard siblings
```

Hint: First, create a helper table containing each pair of siblings. This will make comparing the sizes of siblings when constructing the main table easier.

Hint: If you join a table with itself, use `AS` within the `FROM` clause to give each table an alias.

Hint: In order to concatenate two strings into one, use the `||` operator.

Use Ok to test your code:

```
python3 ok -q sentences
```

Roughly speaking, there are two tasks we need to solve here:

Figure out which dogs are siblings

A sibling is someone you share a parent with. This will probably involve the `parents` table.

It might be tempting to join this with `dogs`, but there isn't any extra information provided by a dogs table that we need at this time. Furthermore, we still need information on sibling for a given dog, since the `parents` table just associates each dog to a parent.

The next step, therefore, is to match all children to all other children by joining the parents table to itself. The only rows here that make sense are the rows that represent sibling relationships since they share the same parent.

Remember that we want to avoid duplicates! If dog A and B are siblings, we don't want both A/B and B/A to appear in the final result. We also definitely don't want A/A to be a sibling pair. Enforcing ordering on the sibling names ensures that we don't have either issue.

Construct sentences based on sibling information

After determining the siblings, constructing the sentences just requires us to get the size of each sibling. We could join on the `dogs` and `sizes` tables as we did in an earlier problem, but there's no need to redo that work. Instead, we'll reuse our `size_of_dogs` table to figure out the size of each sibling in each pair.

Submit

Make sure to submit this assignment by running:

```
python3 ok --submit
```

Troubleshooting/Advanced SQLite

Troubleshooting

Python already comes with a built-in SQLite database engine to process SQL. However, it doesn't come with a "shell" to let you interact with it from the terminal. Because of this, until now, you have been using a simplified SQLite shell written by us. However, you may find the shell is old, buggy, or lacking in features. In that case, you may want to download and use the official SQLite executable.

If running `python3 sqlite_shell.py` didn't work, you can download a precompiled sqlite directly by following the following instructions and then use `sqlite3` and `./sqlite3` instead of `python3 sqlite_shell.py` based on which is specified for your platform.