Homework 9: SQL

> Adapted from cs61a of UC Berkeley.

Readings

You might find the following references useful:

- Section 4.3 - Declarative Programming

Starter Files

Get your starter file by cloning the repository: https://github.com/JacyCui/sicp-hw09.git

```
git clone https://github.com/JacyCui/sicp-hw09.git
```

`hw09.zip` is the starter file you need, you might need to unzip the file to get the skeleton code.

```
unzip hw09.zip
```

`README.md` is the handout for this homework. `solution` is a probrab solution of the homework. However, I might not give my solution exactly when the homework is posted. You need to finish the task on your own first. If any problems occurs, please make use of the comment section.

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 < hw09.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 hw09.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 --local
```

Questions

SQL

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`.

Q1: Size of Dogs

The Fédération Cynologique Internationale classifies a standard poodle as over 45 cm and up to 60 cm. The `sizes` table describes this and other such classifications, where a dog must be over the `min` and less than or equal to the `max` in `height` to qualify as a `size`.

Create a `size_of_dogs` table with two columns, one for each dog's `name` and another for its `size`.

```
-- The size of each dog
CREATE TABLE size_of_dogs AS
  SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

The output should look like the following:

```
sqlite> select * from size_of_dogs;
abraham|toy
barack|standard
clinton|standard
delano|standard
eisenhower|mini
fillmore|mini
grover|toy
herbert|mini
```

Use Ok to test your code:

```
python3 ok -q size_of_dogs --local
```

Q2: 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 "REPLACE THIS LINE WITH YOUR SOLUTION";
```

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 --local
```

Q3: 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 "REPLACE THIS LINE WITH YOUR SOLUTION";

-- Sentences about siblings that are the same size
CREATE TABLE sentences AS
  SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

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

```
sqlite> select * from sentences;
The two siblings, barack plus clinton have the same size: standard
The two siblings, abraham plus grover have the same size: toy
```

> **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 --local
```

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.

Another way to start using SQLite is to download a precompiled binary from the SQLite website. The latest version of SQLite at the time of writing is 3.28.0, but you can check for additional updates on the website.

However, before proceeding, please remove (or rename) any SQLite executables (`sqlite3`, `sqlite_shell.py`, and the like) from the current folder, or they may conflict with the official one you download below. Similarly, if you wish to switch back later, please remove or rename the one you downloaded and restore the files you removed.

Windows

1. Visit the download page linked above and navigate to the section Precompiled Binaries for Windows. Click on the link **sqlite-tools-win32-x86-*.zip** to download the binary.
2. Unzip the file. There should be a `sqlite3.exe` file in the directory after extraction.
3. Navigate to the folder containing the `sqlite3.exe` file and check that the version is at least 3.8.3:

```
$ cd path/to/sqlite
$ ./sqlite3 --version
3.12.1 2016-04-08 15:09:49 fe7d3b75fe1bde41511b323925af8ae1b910bc4d
```

macOS Yosemite (10.10) or newer

SQLite comes pre-installed. Check that you have a version that's greater than 3.8.3:

```
$ sqlite3
SQLite version 3.8.10.2
```

Mac OS X Mavericks (10.9) or older

SQLite comes pre-installed, but it is the wrong version.

1. Visit the download page linked above and navigate to the section **Precompiled Binaries for Mac OS X (x86)**. Click on the link **sqlite-tools-osx-x86-*.zip** to download the binary.
2. Unzip the file. There should be a `sqlite3` file in the directory after extraction.
3. Navigate to the folder containing the `sqlite3` file and check that the version is at least 3.8.3:

   ```
   $ cd path/to/sqlite
   $ ./sqlite3 --version
   3.12.1 2016-04-08 15:09:49 fe7d3b75fe1bde41511b323925af8ae1b910bc4d
   ```

Ubuntu

The easiest way to use SQLite on Ubuntu is to install it straight from the native repositories (the version will be slightly behind the most recent release):

```
$ sudo apt install sqlite3
$ sqlite3 --version
3.8.6 2014-08-15 11:46:33 9491ba7d738528f168657adb43a198238abde19e
```