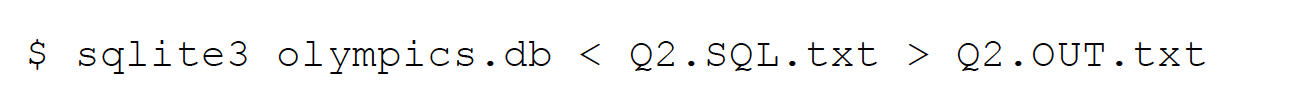
HW1:[30 pt] SQLite[[1]](#footnote-1)

The following questions will help refresh your memory about SQL and get you started with [SQLite](https://www.sqlite.org/index.html) ([basic commands](https://www.sitepoint.com/getting-started-sqlite3-basic-commands/)) --- a lightweight, serverless embedded database that can easily handle up to multiple GBs of data. As mentioned in class, SQLite is the world’s most popular embedded database. It is convenient to share data stored in an SQLite database --- just one cross-platform file, and no need to parse (unlike CSV

files).

You will modify the given Q2.SQL.txt file to add SQL statements and SQLite commands to it.

We will test your answers’ correctness by running your modified Q2.SQL.txt against olympics.db to generate Q2.OUT.txt (assuming the current directory contains the data files).



We will generate the Q2.OUT.txt using the above command. You may not receive any points (1)

if we are unable to generate the file, or (2) if you do not strictly follow the output formats specified in each question below.

We have added some lines of code in the Q2.SQL.txt file. Their purposes are:

* .headers off. : After each question, an output format has been given with a list of column names/headers. This command ensures that such headers are not displayed in the output.
* .separator ‘,’ : To specify that the input file and the output are comma-separated.
* select ‘ ‘ : This command prints a blank line. After each question’s query, this command ensures that there is a new line between each result in the output file.

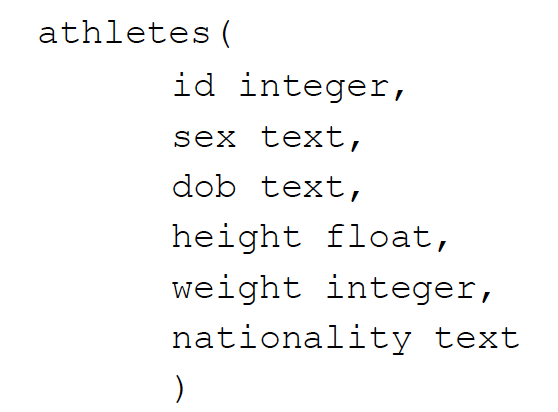
WARNING: Do not copy and paste any code/command from this PDF for use in the sqlite command prompt, because PDFs sometimes introduce hidden/special characters, causing SQL error. Manually type out the commands instead.

Note: For the questions in this section, you must use only [INNER JOIN](https://www.w3schools.com/sql/sql_join_inner.asp) when you perform a join between two tables. Other types of join may result in incorrect outputs.

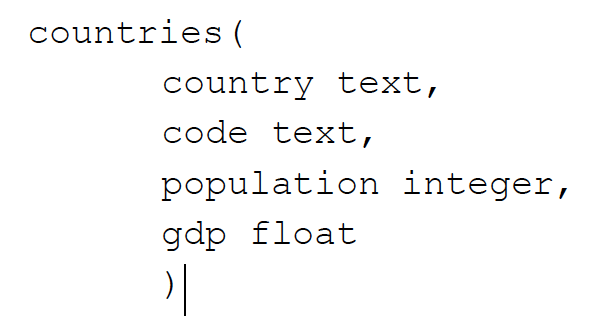
Note: Do not use .mode csv in your Q2.SQL.txt file. This will cause quotes to be printed in the output of each select ‘ ‘; statement.

1. [2 pt] Import data. Create an SQLite database called olympics.db and provide the SQL code (and SQLite commands) used to create the following tables. Use SQLite’s [dot commands](https://www.sqlite.org/cli.html#special_commands_to_sqlite3_dot_commands_) (.separator STRING and .import FILE TABLE) to import data from files.

Import the data about the athletes from athletes.csv (**in the Q2 Folder, posted in CAVAS**) into a new table (in olympics.db) called athletes with the schema:



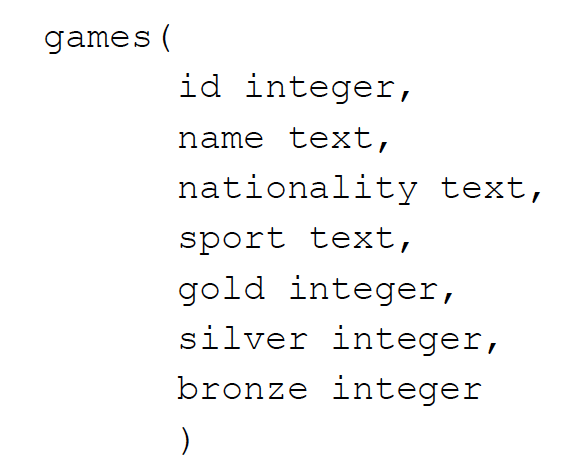
Import the data on the events from countries.csv (in the Q2 Folder) into a new table (in olympics.db) called countries with the schema:



Import the data on the games (such as athlete id, name, nationality information along with the

sport name and number of different medals won) from games.csv (in the Q2 Folder) into a new

table (in olympics.db) called games with the schema:



1. [1 pt] Build indices. Create the following four indexes that will speed up subsequent join

operations (speed improvement is negligible for this small database, but significant for larger

databases):

1. *id\_index* for athletes table’s id column

2. *nationality\_index* for athletes table’s nationality column

3. *id\_games\_index* for games table’s id column

4. *code\_index* for countries table’s code column

c. [1 pt] Quick computation. Find the total number of gold medals won by the athletes with

nationality ‘USA’.

Output format (a single line that contains one number):

***count\_usa\_gold***

d. [2 pt ] Distinct Sports: List all distinct sports where ‘USA’ won gold medals, in alphabetical

order. Use the [DISTINCT](https://sqlite.org/lang_select.html#distinct) keyword to avoid repetition of sports.

e. [2 pt] Handling Empty Cell: Notice that some of the cells in the population column of

countries are empty strings. Use the following command to explicitly set these empty cells to

NULL by updating the table as follows:

UPDATE countries SET population = NULL where population = ‘ ‘;

With the empty cells filled with null values, you can use the [IS NOT NULL](https://www.w3schools.com/sql/sql_null_values.asp) keyword to exclude such rows in any query.

Once you have updated the countries table to fill the empty cells with null values, list the top 5 most populated countries along with the sum of all medals (i.e., gold, silver and bronze combined) won by the country. Your result should be ordered by country population, from highest to lowest.

For example: if ‘United States’ has won 10 medals in total and *‘ United Kingdom’* has won 12, but USA is more populated than UK, then your result should order them as follows:

*United States, 10*

*United Kingdom, 12*

Format of each line in the output (5 lines total):

*full\_country\_name, sum\_of\_medals*

f. [2 pt] Female Athletes. Find the top 5 countries with the highest number of female athletes who have won at least one gold medal. List the names of these countries (full names and not the country code) along with their respective female athlete counts as output. Your output should be ordered by the count from highest to lowest.

Format of each line in the output (5 lines total):

*Full\_country\_name, count\_of\_gold\_winning\_female\_athletes*

g. [4 pt] Fit Athlete. The BMI of a person is calculated as the ratio of their weight (in kilograms) to their squared height (in meters). A person is considered ‘fit’ if his or her BMI is between 18.5 (inclusively) and 24.9 (inclusively).

Find the athletes who are fit (according to the above definition) and list their names along with their BMI values. Sort the list first alphabetically by their names and then by their BMI values from lowest to highest, and limit the result to 10 rows.

Note: The athletes table lists the height in meters and weight in kilograms.

Format of each line in the output:

*athlete\_name, bmi\_value*

Hint: Consider using [BETWEEN](https://www.w3schools.com/sql/sql_between.asp) .

h. [5 pt] Percentages of Gold & Silver Medals: Calculate the percentage of gold medals and silver medals won by each country in INTEGER FORMAT only (if you use float in your intermediate calculations, then round down the final value using *‘CAST (value as int)’* ). Order the countries first by their gold medal percentage (highest to lowest) and then by their silver medal percentage (highest to lowest). List the full names of the countries along with their gold and silver medal percentages. Limit the result to the top 20 countries.

Format of each line in the output (20 lines total):

*full\_country\_name, gold\_medal\_percentage, silver\_medal\_percentage*

Hint: Consider using SELECT statements to find the denominator in the percentage calculation.

Note: Percentage of gold medals won by a country = Number of gold medals won by that country / Total number of gold medals in the olympics.

i. [4 pt] Creating a view: [Create a view ( virtual table )](https://sqlite.org/lang_createview.html) called *gdp\_metrics*, where each row

consists of distinct pairs of countries such that country1’s GDP always exceeds country2’s

GDP by no more than 100 (note that the difference in the GDPs of the two countries should not

be equal to 0). Please use the following method to calculate the difference:

*(country1\_gdp - country2\_gdp)*

In each pair *<full\_country1\_name, full\_country2\_name>* you form, country1\_gdp should be

greater than country2\_gdp.

Each row of the view should be of the form:

*full\_country1\_name, full\_country2\_name, difference\_gdp*

The format of the view is:

*gdp(full\_country1\_name, full\_country2\_name, diff)*

Using this view, write a query to display the first 5 rows in the view after sorting the gdp differences in descending order.

Format of each line in the output (5 lines total):

*country1\_full\_name, country2\_fullﬂname, diff*

Note: Remember that creating a view will produce no output.

Optional reading: [Why create views?](https://stackoverflow.com/questions/1278521/why-do-you-create-a-view-in-a-database)

j. [1 pt] Calculate the total number of such pairs created from the view made in part i.

Output format:

*count\_total\_pairs*

k. [6 pt] SQLite supports the simple but powerful Full Text Search (FTS) for fast text-based querying ([FTS documentation](https://www.sqlite.org/fts3.html) ).

Import the movie overview data from movie-overview.txt (in the Q2 folder) into a new FTS table (in olympics.db) called *movie\_overview* with the schema:

*movie\_overview (*

*id integer,*

*name text,*

*year integer,*

*overview text,*

*popularity decimal*

*)*

Note: Create the table using fts3 or fts4 only. Also note that keywords like NEAR, AND, OR and NOT are case-sensitive in FTS queries.

1. [2pt] Count the number of movies whose overview field starts with the word *“in”*.

Output format:

*count\_overview\_in*

2. [2pt] List, in ascending order, the id†s of the movies that contain the terms “love” and “city” in their overview†fields with no more than 7 intervening terms in between.

Output format (one or more lines):

*id*

3. [2pt] List the names of the movies that contain partially/fully the word “geo” in their overview field. Look for words in the overview field that begin with the word “geo”. For example, “geography”, “Georgia”, and “geo” all partially/fully match and begin with the word “geo”.

Output format (one or more lines):

*movie\_name*

**Deliverables:** Place all the files listed below in the Q2 folder

● *Q2.SQL.txt* : Modified file additionally containing all the SQL statements and SQLite commands you have used to answer questions a - i in the appropriate sequence.

● *Q2.OUT.txt* : Output of the queries in Q2.SQL.txt . Check above for how to generate this file.

1. This homework is created by Prof Chau at Georgia Tech [↑](#footnote-ref-1)