CITS1402 Week 3 Lab

Gordon Royle

2022 Semester Two

Learning Aims

This lab is concerned with combining the `SELECT` and `JOIN` commands to perform more advanced queries, along with additional practice on `ORDER BY` and `LIMIT`.

We start with a worked example from the `sqliteWorld.db` database.

This is a very old database that was originally distributed by MySQL as a sample database. I have changed it to SQLite, but otherwise left it unchanged. The data is hugely out of date, but this does not affect its utility as a teaching database.

Worked Example

Suppose that you must list each city and the country in which it lies.The names of the cities are in the table `City` while the names of the countries are in the table `Country` and so we need to join those two tables. The `JOIN` condition will need to use the *three-letter country code* in order to ensure that the two halves of each row of the joined table refer to the same country.

Unfortunately, the column that contains the country code has a *different name* in the two tables, namely `countryCode` in the table `City`, but `code` in the table `Country`. This means that we cannot use the `USING` keyword to perform the join, but must use another way.

```
SELECT <stuff>
FROM City, Country
WHERE countryCode = code;
```

Next we have to decide what columns we want to extract from this table, so let's just look at a few lines of the table. I have changed the output mode to "comma-separated values" which separates each column with a comma. (I find this just slightly easier to read than the default mode using the vertical bar.)

```
sqlite> .headers on
sqlite> .mode csv
sqlite> SELECT *
   ...> FROM City, Country
   ...> WHERE countryCode = code LIMIT 4;
ID,Name,CountryCode,pop,Code,Name,Capital,continent
1,Kabul,AFG,1780000,AFG,Afghanistan,1,Asia
2,Qandahar,AFG,237500,AFG,Afghanistan,1,Asia
3,Herat,AFG,186800,AFG,Afghanistan,1,Asia
4,Mazar-e-Sharif,AFG,127800,AFG,Afghanistan,1,Asia
```

The question asks us to return the *name* of the `City`, which is stored in the column `Name`, and the name of the `Country`, which is stored . . . in a *different column* also called `Name`. So we have two columns called `Name`, which is obviously a potential source of error.

In particular, if we try

```
SELECT Name, Name
FROM City, Country
WHERE countryCode = code;
```

then SQLite will complain `ambiguous column name: Name` meaning that it can't tell which column is being referred to.

However, SQLite knows that the first four columns came from the table `City` and the final four columns came from the table `City`, and we can *disambiguate* the same-named columns by using the "full name" that specifies both the table- and column-name.

```
SELECT City.Name, Country.Name
FROM City, Country
WHERE countryCode = code;
```

```
sqlite> SELECT City.Name, Country.Name
   ...> FROM City, Country
   ...> WHERE countryCode = code
   ...> LIMIT 4;
Name,Name
Kabul,Afghanistan
Qandahar,Afghanistan
Herat,Afghanistan
Mazar-e-Sharif,Afghanistan
```

In the output table the two columns still have the same name, namely `Name`, and so it would be probably be better to *assign new names* to the output columns.

```
SELECT City.Name AS cityName, Country.Name AS countryName
FROM City, Country
WHERE countryCode = code;
```

```
sqlite> SELECT City.Name AS cityName, Country.Name AS countryName
   ...> FROM City, Country
   ...> WHERE countryCode = code
   ...> LIMIT 4;
cityName,countryName
Kabul,Afghanistan
Qandahar,Afghanistan
Herat,Afghanistan
Mazar-e-Sharif,Afghanistan
```

It is important to realise that this *does not alter* the structure of the database itself — the newly-named columns are columns of the table that is *returned by the query*, and which is simply printed to the screen, and which vanishes as soon as the user quits `sqlite3`.

To make the results of a query *persistent*—in other words, to *save* the results—it is necessary to insert them into a table; we will eventually learn how to do this.

Lab Questions

This lab should be completed by the end of Teaching Week 3.

Note that the Week 4 Lab submission may ask for questions from this lab as well as the next one, so it is advisable not to treat this as a week off.

Your commands will be tested against a database with the same schema as `sqliteWorld` but not necessarily the same data.

Question 1. Write a SQL query that will output the full details (i.e., just use `SELECT *`) for each city with population strictly greater than 2500000. Note that the city Abidjan in Côte D'Ivoire has population exactly 2500000 and therefore should *not* appear in the output.

Question 2. Write a SQL query that will output the *city name* and *country name* for each European city.

The first few rows of the output table should be:

```
Amsterdam|Netherlands
Rotterdam|Netherlands
Haag|Netherlands
```

QUESTION 3. Alter the worked-example query at the top of this lab sheet to print out all the Australian cities. (Use `AND` to add a further condition to the `WHERE` clause.)

The first few rows of the output table should be:

```
Sydney|Australia
Melbourne|Australia
Brisbane|Australia
```

QUESTION 4. List the 3-letter country codes of all of the countries where English is spoken. By "English is spoken" we mean that English appears as *any one* of the rows of the table `CountryLanguage` for that country.

(Hint: You only need to use *one table* for this question, so no joins are needed.)

The first few rows of the output table should be:

```
ABW
AIA
ANT
```

QUESTION 5. Write a SQL query to list the rows of the table `CountryLanguage` relating to Afghanistan, which has country code AFG.

Just by looking at the output (not writing another query) can you tell how many languages are spoken in Afghanistan and how many of them are *official languages*?

QUESTION 6. Write a single query to construct a 2-column table where each row contains a *country name* and a *language* spoken in that country. If there are multiple languages spoken in a country, then the output should contain multiple rows, one for each language.

If your query is correct, then the first few rows of the output table should be:

```
Aruba|Dutch
Aruba|English
Aruba|Papiamento
Aruba|Spanish
Afghanistan|Balochi
Afghanistan|Dari
```

QUESTION 7. Write a single query to construct a 2-column table where each row contains a *country name* and an *official language* spoken in that country. If there are multiple official languages for a country, then the output should contain multiple rows, one for each official language.

Note: The `isOfficial` column uses the text values `"T"` and `"F"` to indicate true and false respectively.

If your query is correct, the first few rows of the output table should be:

```
Aruba|Dutch
Afghanistan|Dari
Afghanistan|Pashto
Anguilla|English
Albania|Albaniana
```

QUESTION 8. Write a SQL query to list *just the names* of all the countries where French is an official language

If your query is correct, the first few rows of the output table should be:

```
Burundi
Belgium
Canada
```

QUESTION 9. Write a SQL query to list *just the names* of all the countries where French is an official language even though French is spoken by fewer than 25% of the inhabitants. For example, Vanuatu has three official languages, including French, although French is actually spoken by only 14% of the inhabitants.

If your query is correct, then it should return 13 countries from Burundi to Vanuatu.

QUESTION 10. Write a SQL query to list the ID numbers of each *capital city*. (Your list may have some blank lines in it, can you figure out why?)

QUESTION 11. Write a single SQL query to list the *name* of each country along with the *name* of its capital city. (You may ignore "countries" with no capital city.)

If your query is correct, the first few rows of the output table should be:

```
Aruba|Oranjestad
Afghanistan|Kabul
```

```
Angola|Luanda
```

Question 12. CHALLENGE List the country codes of all countries that have both English and French as an official language. (There are only three such countries.)

Hint: You only need to use the table `CountryLanguage` but because SQL is a "row-processing machine", you cannot *directly* refer to two different rows of the same table. If only you could work with *two copies* of the same table at the same time ...
