



In this lab, you will again use the PHPMyAdmin web interface to your database. Use the SQL tab for direct query entry. The lab covers the use of table aliases and SELECT DISTINCT.

Note again that these lab sheets are not assessed, but they may be discussed in tutorials and their content is examinable!

Estimated time to complete this lab: 30 minutes.

### A) Sister cities

Your city table should now have a number of cities in different countries in it, if possible several cities for most of the countries. If not, add (as a warm-up exercise) extra records with a SQL INSERT query. We'll also assume from here that the table is called **city** and the name and country of the respective city are in the **name** and **country** fields of the table.

Your next task is to find all possible pairings for sister cities, that is, all pairs of cities from different countries, listed together with their country. This is a little tricky because we only have one city table, but for each record of our dataset we need two cities and two countries. The solution is – of course – table aliasing. That is, we use one table alias for each of the two sister cities. Alias the city table for the first city as, e.g., c1, and that of the second sister city as, e.g., c2, and try this:

```
SELECT c1.name, c1.country, c2.name, c2.country
       FROM city c1, city c2
       WHERE c1.country != c2.country;
```

This should now give you a data set with all possible pairs of cities. Make note of the total number of rows (records) that the query says that it is returning. Now have a close look at the data set. All pairs are from different countries – that's what we wanted. But there is a problem: Each pair turns up twice, in different order. E.g., if you have a pair:

Auckland | NZ | Sydney | Australia

then there will also be a pair

Sydney | Australia | Auckland | NZ

Now of course that mightn't be what we want. Can you figure out a way of changing the WHERE clause such that this doesn't happen? Note: There is more than one possible solution here – you may want to think of several. If you're successful, the total number of rows returned will be half of what you got above.

### B) Country pairings

Now we want to know which possible country partnership might arise if any of the potential sister city partnerships above worked out. Simple, you might think, we just leave the city names out (and then do the little extra open-ended bit above):

```
SELECT c1.country, c2.country
       FROM city c1, city c2
       WHERE c1.country != c2.country;
```

Alas, now we get many identical country-country pairs in our data set. Even though we have biffed the city names as data to be returned, the records still relate to cities, not countries. Bummer! So how can we fix this? Try this:

```
SELECT DISTINCT c1.country, c2.country
       FROM city c1, city c2
       WHERE c1.country != c2.country;
```

DISTINCT ensures that the data set that is returned contains no duplicate records.

Of course, to make it truly work, you'll also have to apply the final fix for the swapped repeated pairs from A) here.