Workshop 2 – Introduction to SQL

So last week you learnt about what databases were, and began playing around with your own one through MySQL Workbench. This week we'll look at how you can communicate with your database, pulling out new information and new insights into your data.

In this tutorial you'll begin to use SQL (Structured Query Language) to manipulate your data. This is important stuff, and the few simple things you learn today will hold you in good stead for future database handling.

The tutorial will introduce you to five new skills.

1. Uploading data from an SQL dump
2. Running SELECT queries
3. Extracting summary statistics and ordered lists
4. Aggregating results by shared attributes
5. Amending table contents

Once again, we'll be working through all of this using MySQL Workbench. Just work your way through this worksheet; your tasks have been helpfully highlighted in **bold font**.

1 Loading Data

The first thing you'll need to do is load in the new datasets that we'll be working with this week. But, rather than getting you all to create the tables and run LOAD DATA INFILE again, we're going to be nice and give you an SQL dump of the data.

An SQL dump is a file created by a database engine to ease data transfer. It includes all of the table definitions (field names, data types, indexes etc.), and the data itself.

You can find the SQL dump file on Moodle or from here: http://dev.spatialdatacapture.org/data/workshop_2

Now run through the following steps to import the data:

1. Download the file (right click, Save Link As…).
2. Load up your database in MySQL Workbench 6.3 by opening the connection you setup last week
3. Open the database denoted with your username; right-click on it and select Set as Default Schema.
4. Go to File and Open SQL Script. Navigate to the downloaded file and open it.

The script will open up in a new tab in Workbench. Have a look through and see if you find any familiar parts of the script. You'll notice too that the data is included within the file.

Run the script by pressing the Lighting Key 

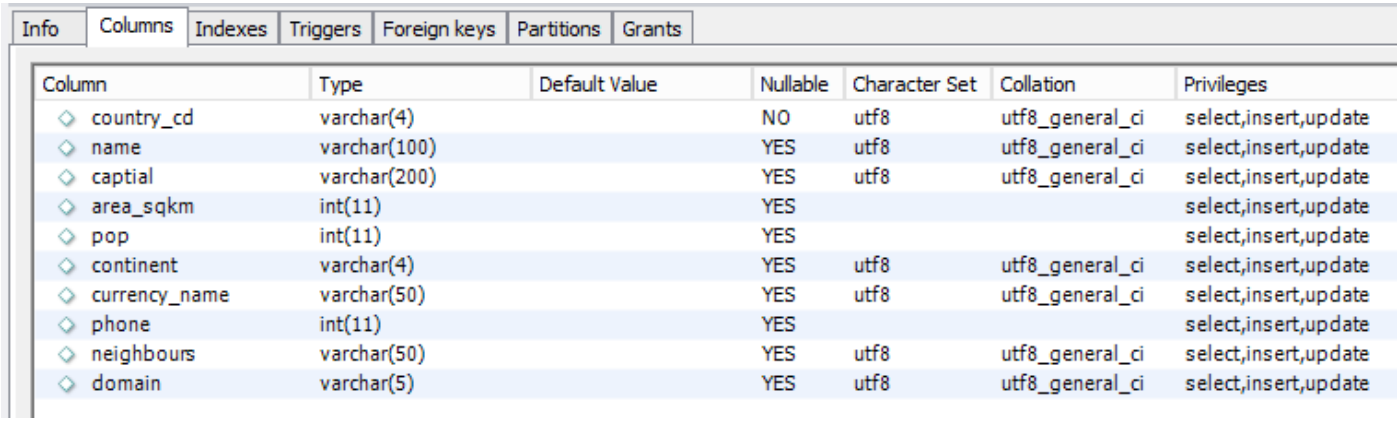
Then refresh your tables list (right click on Tables – Refresh) folder. You'll find two new tables – Cities and Countries.

2 SELECT Queries

During this tutorial, we'll be working with the **Countries** table. Open up the table data (Right Click – Select Rows) and have a look at the data. **What columns are within the dataset?**

You should also check out the table definitions and see how the columns are defined. These are found through Right Click – going to Table Inspector – and then going to the Columns tab. **What data types can be found within the table?**

SPOLIER ALERT – The Table Inspector window looks a bit like this. Column gives you the column name; Type refers to its data type (e.g. `varchar` = a string, `int` = an integer).



Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
country_cd	varchar(4)		NO	utf8	utf8_general_ci	select,insert,update
name	varchar(100)		YES	utf8	utf8_general_ci	select,insert,update
captial	varchar(200)		YES	utf8	utf8_general_ci	select,insert,update
area_sqkm	int(11)		YES			select,insert,update
pop	int(11)		YES			select,insert,update
continent	varchar(4)		YES	utf8	utf8_general_ci	select,insert,update
currency_name	varchar(50)		YES	utf8	utf8_general_ci	select,insert,update
phone	int(11)		YES			select,insert,update
neighbours	varchar(50)		YES	utf8	utf8_general_ci	select,insert,update
domain	varchar(5)		YES	utf8	utf8_general_ci	select,insert,update

Now, open a new SQL script by clicking on the  icon in the top left of the window. Let's start querying the data.

The first query is simple. All you're going to do is select all of the data from the countries table. To do this you just **enter the following command and run it:**

```
SELECT * FROM countries;
```

REMEMBER to add the semicolon, to make sure it remains separate from all other commands you write.

Helpful Tip #35251 MySQL Workbench has an annoying habit of limiting the number of rows it returns for each SQL statement (the default is 1000). If you want to switch this off, navigate to Edit – Preferences – SQL Queries – And untick the Limit Rows box.

You now have all of the data from the countries table. But maybe you only want certain columns from that table. To limit this, replace the * with the column names that you want. **Add something like the following to the script, and run it again by highlighting it and pressing the Lightning Key**.

```
SELECT name, area_sqkm FROM countries;
```

But still, this is not particularly interesting. Let's start putting SQL to the test by limiting what we get back. We do this using by adding a WHERE command, after which we can add some conditions. Here is an example:

```
SELECT name, pop FROM countries WHERE currency_name = 'Pound';
```

Instead, I want you to create a statement that returns the country name and population, ONLY when the country is found within Asia. Asian countries are denoted with the 'AS' continent code, which must be contained within single speech marks.

Try writing this query, in the same way as the one above, to return all countries in Asia. How many rows are returned?

Q1

If you want to check your answer then ask the Slack AnswerBot! When you see the blue circle, use the question number to ask the AnswerBot for help or a hint.

Now you're getting the hand of this, we will start adding additional conditions on the data, getting a bit more specific.

Next, alter your last statement so you only select countries that are within Asia AND with a population of above 1 million people. Remember to use the correct column names. How many countries does this query return?

Now alter the statement again to limit your Asian countries to just those with a population of more than 1 million and less than 5 million people. Sounds a bit complicated, but remember this is just the same structure; you're simply adding extra conditions.

You should now have list of countries looking something like this ->

name	pop
United Arab Emirates	4975593
Armenia	2968000
Georgia	4630000
Kuwait	2789132
Lebanon	4125247
Mongolia	3086918
Oman	2967717
Palestinian Territory	3800000
Singapore	4701069
Turkmenistan	4940916
NULL	NULL

Finally, let's say we want to add countries from South America to this list. We do this by specifying that we want countries from Asia OR South America. **So first, let's run this statement that brings all of our requests together.**

```
SELECT * FROM countries
WHERE continent = 'AS' AND pop > 1000000 AND pop < 5000000 OR continent = 'SA';
```

Notice anything wrong with the results?

You should find that while the Asian countries are limited to just those with a population between 1m and 5m people, ALL of the South American countries are returned! The lesson is to be wary of the `OR`! Adding a statement using `OR` can potentially conflict your `AND` statements.

To get around this, we protect the other statements from the `OR` statement by containing it within brackets. The continent conditions are grouped together and executed separately from the population statements, like so:

Helpful Tip #151577 You'll probably need to look up column names quite often. In Workbench, you can find a list of these either in Table Inspector, or by expanding your Tables list (in the Schemas panel), and then expanding the table name.

```
SELECT * FROM countries
WHERE (continent = 'AS' OR continent = 'SA') AND pop > 1000000 AND pop < 5000000;
```

Run this and see what you get. The results should be more in line with what we wanted – giving us just countries with limited populations, from Asia or South America.

Next, run a query that extracts the country codes and country names for all rows that have an area greater than 5000000 sqkm^2. How many countries do you get?

Q2

The queries so far have dealt mainly with simple condition statements. We also need to know how to query text data. The SQL structure is the same, although the syntax is a bit more complicated.

Let's first try searching for a country by specific name. We did this earlier when we searched for countries from specific continents. Write a new query to extract all columns for Brazil.

However, we're often not interested in all of the text within a specific field, just some of it. SQL provides a function for searching for parts of strings within a text field. To do so we combine SQL's `LIKE` function with the `%` placeholder, to search for a part of a string. The search string can include spaces, the `%` takes the place of any additional text. **Here is an example, try running it.**

```
SELECT * FROM countries WHERE name LIKE '%land%';
```

This extracts data for all countries whose name includes the string 'land'.

Using this same method, we can turn out attention to the `neighbours` column. This field contains all of the country codes (`country_cd`) of all neighbouring countries as a string.

Your final task for this section is to first, find the country code for Russia; and then, write a query to produce all of the rows for only the countries neighbouring Russia. Hint: You'll use Russia's country code as the partial search string like above.

Helpful Tip #894521
Clicking the Execute button every time gets pretty boring. Run the query you've just written by pressing CTRL and Enter on the keyboard instead.

3 Summary Statistics and Ordering Results

Using built-in functions, you can quickly and easily generate summary statistics using SQL. These are useful for getting a grip on the data you have, which will often run into thousands or millions of rows.

One of the first things you'll need to know about a new dataset is its row count. You can achieve this by using the `COUNT(*)` function. You place this where you previously requested column names, like so:

```
SELECT COUNT(*) FROM countries;
```

More importantly, it allows you to pull out row counts for queried data. **Use the code above, and what you've learnt already, to find out how many countries there are in Europe only.**

Other functions focus solely on extracting statistics relating to specific columns. Examples include `AVG(`*`column`*`)`, `SUM(`*`column`*`)`, `MAX(`*`column`*`)` and `MIN(`*`column`*`)`. In each case you replace *`column`* with the column name you are interested in. So to find out the average country population you'd run the following:

```
SELECT AVG(pop) FROM countries;
```

Likewise, `SUM(pop)` would give you the sum population, `MAX(pop)` would yield the maximum population size, and `MIN(pop)` would give you the minimum country population.

Try finding out the mean population size for countries in Europe alone.

Q3

Column arithmetic is another useful feature of SQL. This allows you to add, divide or multiply columns together to derive new data.

For this example, we'll calculate country population density from the fields given. So we divide our population field by our country area field. We provide the result with an alias using the `AS` function. And we add in the country name field to make sure we know which density result relates to which country.

Helpful Tip #451349 Add comments to your SQL code by using `--` and then your comment text. It'll be ignored when you execute your queries.

The resulting command looks like this. Run it.

```
SELECT name, pop/area_sqkm AS population_density FROM countries;
```

At this point, if I were to ask you which country has the highest population density would you be able tell me? Yes, well maybe you already knew it was Monaco, or maybe you searched through already. But don't you want to make your life easier? Of course you do.

So this takes us nicely onto `ORDER BY`, a very simple but very useful command. You add `ORDER BY` to the end of any SQL query to order the way in which the results are returned. Again, this is a very useful function when dealing with thousands of rows of data.

We can order according to any column within the given SQL command. So to order countries by area, we'd use our `ORDER BY` command followed by the area column name, like so. **Give it a try.**

```
SELECT * FROM countries ORDER BY area_sqkm;
```

Now the results are all nicely ordered. But wait, I hear you think, what if I want them in the results ordered from top to bottom? Well, I have just the command you. Just add `ASC` or `DESC` after the column name, to sort in ascending or descending order.

So to reverse the order of our area results we just need to add `DESC` after the ordered column name to find the world's largest country. Run the command and find out what you already knew.

`ORDER BY` works on strings (A to Z, or Z to A) and will sort columns by their alias name too.

4 Aggregating Data

Often your data will be provided in a lower granularity than what you need. Aggregating data on shared attributes helps you generate summary statistics relating to grouped data. For this SQL provides the `GROUP BY` function.

Let's say we want to know the total populations using each type of currency. Within the current dataset this would be hard to get at, requiring us to add up the populations of each country using each type of currency. If we `GROUP BY` using `currency_name`, we can shortcut to an answer quickly.

`GROUP BY` allows us to aggregate rows based on shared attributes, and then generate summary statistics relating to each group. In the same way as above we select our summary functions, but then use `GROUP BY` to choose the column to group rows by. The `GROUP BY` function comes at the end of our statement, just like the `ORDER BY` function.

For identifying sum populations for each currency we'd run the following. **Give this a try yourself.**

```
SELECT currency_name, SUM(pop) FROM countries GROUP BY currency_name;
```

How many unique currencies are there? And which one is used by the most people?

Now, a task for you – use this same structure to group by continent, find the average population for each continent, and then, finally, order by average population in descending order. Which continent has the highest average population?

5 Amending Table Contents

At some point, you'll want to amend your table contents, either to add, remove or change some of the data within your database. Workbench provides some functionality for this, but generally writing the code itself will give you much more control over what is happening.

Before we start this, we may need to correct another one of MySQL Workbench's annoying default settings, specifically a 'safe mode' option that prevents you from editing your database. To turn this off, go to Preferences – the SQL Editor tab – and untick the 'Safe Updates' option at the bottom of that window. If it's unticked already, then you're good to go.

To add a new row of data, you use the `INSERT INTO` function. When you run this command you must name the table you are inserting into, and then provide the values being added within brackets. Values have to be provided in the right format (e.g. text must be enclosed within speech marks).

Look back to the lecture and find out how we added a new city to that database. Adjust this query to fit with the country table and add a made up country. Hint: Remember to enter the right number of field values, and in the correct format. AnswerBot might be able to give you a hint here.

Another important function is to be able to update table contents. You can click and edit individual rows using Workbench, but the scripting allows you correct multiple rows at once.

You do this by using the `UPDATE` and `SET` functions combined with a `WHERE` command. An example can again be found in the lecture notes. Be careful with using this one, as if you forget to include a `WHERE` statement all of your records will be amended!

How can you alter this command to suit the country table? Change the population of your made up country to 500, and change the capital city to 'CASAville'.

The final function you need to know is how to delete records based on certain attributes. This works in a similar way to `UPDATE`, except we use the terms `DELETE FROM` to specify the table we wish to delete from. A `WHERE` statement is used to specify which rows to delete, without a `WHERE` statement all records will be deleted – so be careful, there is no undo!

Once more, look at the examples given in the lecture. Create a new DELETE FROM statement that deletes the row containing the made up country you added to the countries table earlier.

Homework – Complete the SQL Quiz on Moodle
Don't worry, it's not assessed, it's really easy and very short.