COMP283 Practical 2, Week 3. 15th February 2018
MySQL Workbench vs Command-Line MySQL Database Manipulation

Last week we used MySQL Workbench to import and manipulate a mysql database from a mysql database export file (aka “dump” file). This week we’ll do something similar from the command line and compare the two.

Log in and startup MySQL Workbench.
Click the tab for your database and then once the window opens up, in the Navigation panel on the left hand side, click the little icon of two arrows pointing away from each other. This should shrink the management sectjon of the navigation panel and make it less cluttered for what we want to do.

Only do the following if you completed last weeks exercise and saved your database with the modified salaries table. If you cannot remember whether you did, then export it again now. Use a name that you can remember - you’ll need this later.

In the schemas section, open up the view of the tables of your database using the small triangle.

Click the first table in your database and then hold down the shift key on the keyboard and click the last table in your database. All the tables should now be selected.

What we want to do is clear out the database so that we can re-import everything. If you had the appropriate privileges to create a database on our system, you could just drop the database and re-create it, but on our system if you try this, you’ll remove your database, but not be able to recreate it… SO DON’T DO THIS!!

If you right click the the tables you selected, and choose “Drop all” and then apply this, you'll find you get an error. What is the problem?

We can delete the remaining tables individually. If you right click your list of tables and choose “refresh all” you should see the two tables that refused to be dropped. Click one of them and choose “Drop table”. Do the same with the other one.
Your database should now contain no tables. Congratulations.

So now, we’re going to use the mysql command line interface - you have probably used this before in other Computer Science database modules.

From the Start Menu, search for MobaXTerm. Click “Session” and then “SSH” (left most icon).
Once it starts, select an SSH host - we’re going to be using one of the linux farm computers to run our mysql commands.

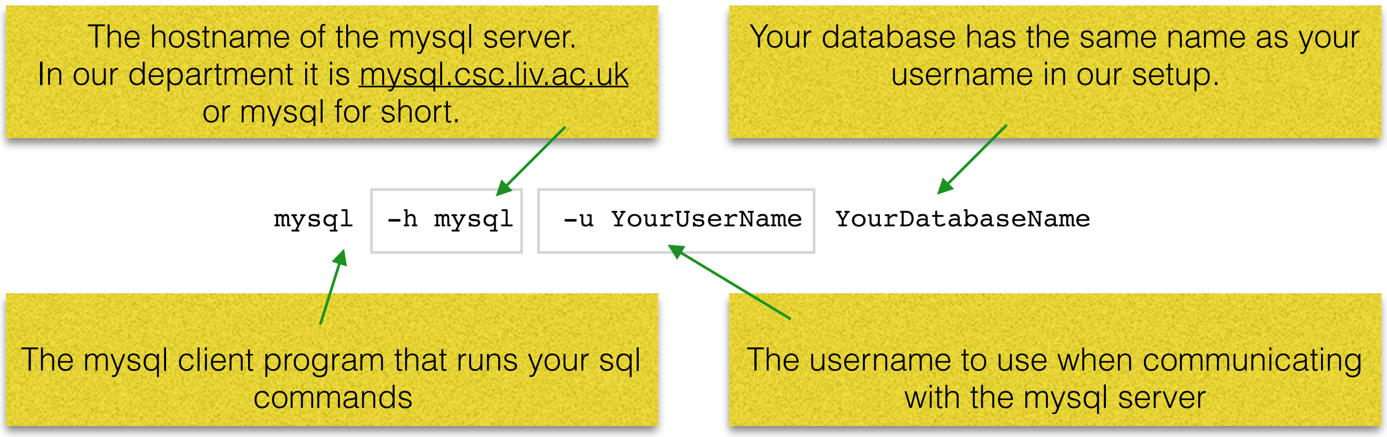
We have to tell the SSH program which linux server we want to connect to. Rather than everyone connect to the same server, use the table below to find the one for you to use:

Birth Month	Use this Linux Server
January, February	linux01.csc.liv.ac.uk
March, April	linux02.csc.liv.ac.uk

Birth Month	Use this Linux Server
May, June	linux03.csc.liv.ac.uk
July, August	linux04.csc.liv.ac.uk
September, October	linux08.csc.liv.ac.uk
November, December	linux06.csc.liv.ac.uk
any problems use:	linux08.csc.liv.ac.uk

Enter your username and password when prompted. Don't choose to save your password for the future.

To start up an interactive (aka command line) mysql session enter the following command:


The hostname of the mysql server.
In our department it is mysql.csc.liv.ac.uk
or mysql for short.
Your database has the same name as your username in our setup.
The mysql client program that runs your sql commands
The username to use when communicating with the mysql server

```
mysql  -h mysql  -u YourUserName  YourDatabaseName
```

e.g. if you are x7ab1

```
mysql -h mysql -u  x7ab1  x7ab1
```

Once the interactive session is running, we can issue mysql commands e.g.:

```
show tables;
```

Since we dropped all the tables from our database, it should be empty.

So let's import the database from the mysql dump that we used before. The easiest way to do this is to use linux's input redirect feature to provide all the appropriate table create and data insert commands from the database dump that we used last week. Exit from the interactive mysql session for now. To do this, type

`quit;`   *<press the return key>*

You're now back in the linux command line environment.
Locate your copy of the "prac1.sql" file that you downloaded at the start of last week's practical. If you can't find it, you can download it from here:

```
https://intranet.csc.liv.ac.uk/~phil/Teaching/COMP283/prac1/
```

Have a look at the contents of the file:

```
less  prac1.sql
```

It's the mysql database dump file. You can actually see in the first few comment lines that I created it using software called "Sequel Pro" (which is available for Mac). You can also see the create table and insert commands - they look like they would if you were to type them yourself. (To exit from the *less* program, press a "q")
We can feed all of the sql commands into the mysql program using linux's input redirection. Type the following (use your own username and database name in the appropriate places).

```
mysql  -h mysql  -u username databasename   <   prac1.sql
```

This will start up the mysql program and send it commands from the prac1.sql file. If all works well, you'll see nothing until it has finished processing. The result will be that the database tables will be created and populated with the saved data. **You should find yourself back in the linux command line environment.** The command ran and once all the lines of the prac1.sql file were used up as input the program ended.
From the linux prompt, go back into the mysql interactive environment (you can use the up arrow keys to retrieve earlier commands. Find the one that invoked the mysql environment just after you logged in to linux). Once you've got mysql running, do a "show tables" - you should see the following:

```
+-----------------+
| Tables_in_u6pj1 |
+-----------------+
| departments     |
| dept_emp        |
| dept_manager    |
| employees       |
| salaries        |
| titles          |
+-----------------+
6 rows in set (0.00 sec)
```

Let's look at the salaries table.

```
describe salaries;
```

```
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| emp_no    | int(11) | NO   | PRI | NULL    |       |
| salary    | int(11) | NO   |     | NULL    |       |
| from_date | date    | NO   | PRI | NULL    |       |
| to_date   | date    | NO   |     | NULL    |       |
+-----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

We need to modify the salary field to be decimal(10,2) rather than int(11). It was easy in MySQL Workbench, but we need to figure out what the mysql commands are to enable us to do that. In fact, MySQL Workbench generates the required statements when you manipulate the graphical UI and so I copied them so that you don't need to figure out which ones they are. To alter the salary field type the following:

```
ALTER TABLE `salaries`
CHANGE COLUMN `salary` `salary` DECIMAL(10,2) NOT NULL ;
```

(the name of the column you're modifying appears twice, because this command can be used to change the name of the column if we wish, and the second time it appears in the command is it's new name - which we are keeping the same as it's old name). Note that the generated mysql statements use the optional single back quote character around table and field names. It's a good idea to get used to doing this because it means that the interpreter can tell the difference between them and commands or other text. If at some point in the future "salary" became a command in mysql, if you did not put single back quotes around it when used as the name of a table, the interpreter would complain because it would see it as a command, rather than a table name. Note that these are not the usual ' character (as used in words like " isn't "). The character we want is on the top left key on a PC keyboard - it is a backwards leaning single quote character `

OK, run the command and describe the table again. It should now have the appropriate type for the salary field. Have a look at the data in the salaries table:

```
select * from `salaries` limit 10;
```

The salary field should display with the appropriate decimal places.

The final change is to make the salary an hourly rate, rather than a yearly salary. As last week we are going to divide every salary by 1,680.

```
UPDATE `salaries` SET `salary` = `salary` / 1680;
```

(Remember that you don't need to enter the ` characters round the names of fields and tables, but it can be a good habit to get into.)

OK, now have another look at some data from the salaries table using the select statement. The salary bit should be a smaller number, with two places of decimals. Note that last week MySQL Workbench gave us lots of warnings when modifying the whole column of data. The interactive mysql session on linux just lets you get on with it (making it easier to do, and also making it easier to utterly destroy the data in your database. That's why its good to make backups before you make major changes).

If all that is ok, we now need to save a copy of our database. The easiest way when using linux is to exit the mysql interactive session and use the mysqldump program. We use linux's output redirect feature to save the output from the mysqldump program in to a file.

```
mysqldump -h mysql -u YourUserName YourDBName > linuxExample.sql
```

(There are lots of options for the mysql dump program, which control features of the dumped database. For now we will go with the default options by not specifying any ourselves).

You should now have a file containing a copy of your modified mysql database.

Again you can use *less* to have a look at its contents.

If you compare this new dump of your database to the one you took when you used MySQL Workbench to make one, there would be many differences in the contents of the file, although the data and database structure are almost certainly exactly the same and you'd get the same database created from both.

Exit from the SSH application, re-open MySQL Workbench and your database and check the tables from there. They should all appear fine and contain the modifications that we want.