# Web Technologies project support worksheet 3

1. Creating CSV files from your Excel document

You should have created data for all 3 of your tables using Excel. Make sure that you have created at least about 20 songs, 5 artists and 5 customers.

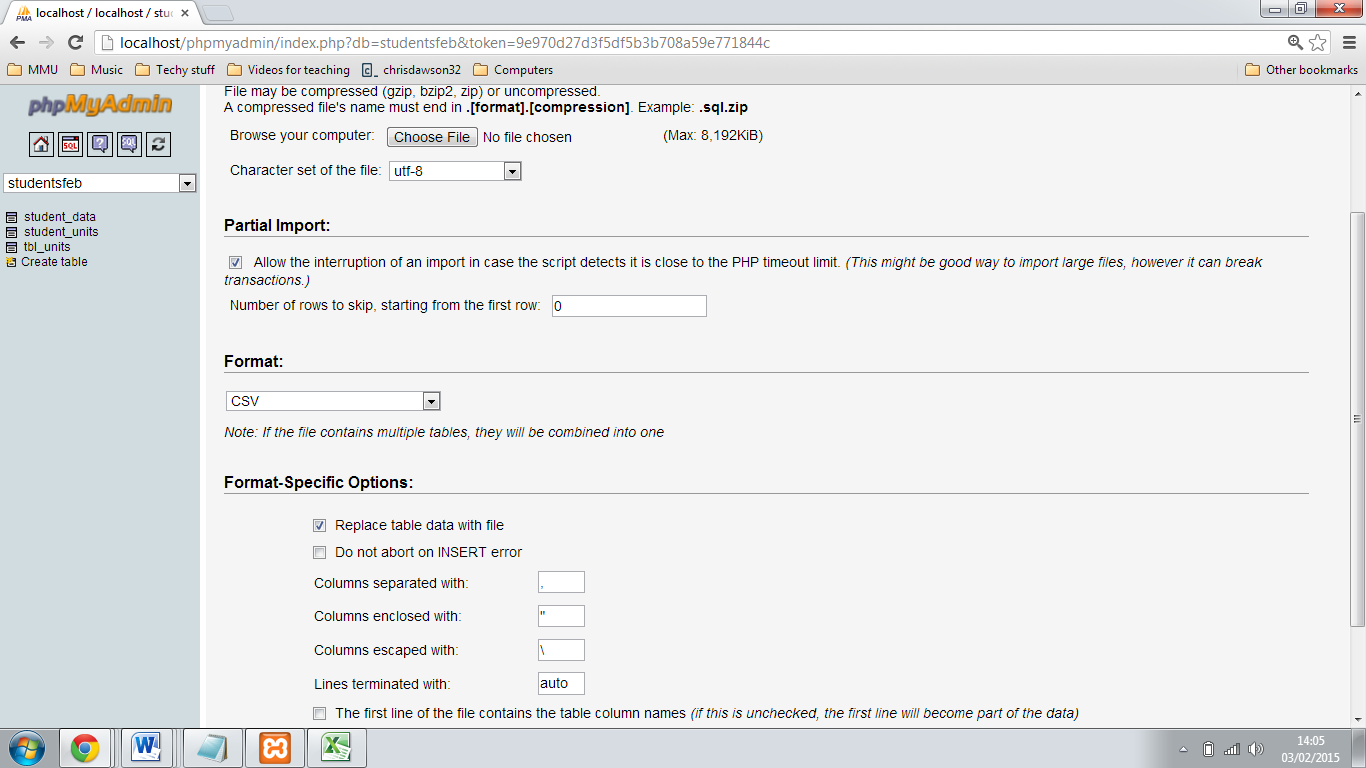
NB: it is important that every ID assigned in the songs table as a foreign key has an entry in its original artists table.

Once done, save the 3 worksheets as CSVs as demonstrated in the week 4 workbook.

1. Creating CSV files from your Excel document

Now, import them into the relevant table.

NB, as there may be data already in the table, you can choose to make PHPMyAdmin replace your table data with the file data. Then upload the data using the Import function in PhPMyAdmin.



1. Running queries from the songs and artists tables

In order to create the songs\_artists.php file for your project you need to create a JOIN query pulling up Song, Artist, Release date, Genre, Price, Nationality, Label from the songs and artists tables. Use the following query structure as a template to create this query:

SELECT **table1.field**, **table2.field**, etc FROM **table1** LEFT JOIN **table2** ON **table1**.**ID\_field** = **table2**.**ID\_field;**

Note that you do not have to include the table name if there is no ambiguity, i.e. if the field name is not repeated in the other table.

NB: as mentioned above, it is important to copy your SQL temporarily every time you run it so that it is not lost once you click on Go. It is a very good idea to save your queries in a Notepad file.

Once you have the query working, adapt them so that the songs are in alphabetical order.

Next you can start to add more songs and artists so that your tables are complete. You can do this from your Excel file, and then add them using the import function in PHPMyAdmin.

1. Applying foreign key constraints

If you have followed the table set-up instructions in past worksheets and added data correctly, you should now already have gone through the process of **applying foreign keys** to your tables. This process is as follows:

* adding an ID field that is the primary key of one table (e.g. artist ID in artists table) into another (e.g. artist ID in songs table).
* Adding ID values in the foreign key table (i.e. an artistID **5** in the **songs** table against the song **Skyfall**) that exists in the original table (artistID **5 = Adele** in the **artists** table).

If you have assigned your foreign key IDs correctly (i.e you have not assigned IDs that do not exist) your queries should work correctly.

Once you are happy that your data is robust and properly related to that in other tables, you can **apply foreign key constraints**. As opposed to simply **applying foreign keys** (which is a structural process) **applying foreign key restraints** is a data validation process which prevents database users from uploading content that is not valid. For example, this might include:

* adding an artistID to a song where the artistID does not exist in the **songs** table
* deleting an artistID from the **artists** table where this artistID is still assigned to a song in the **songs** table.

Although it is not absolutely necessary to do this for the database to function correctly, doing so is good industry practice as it reduces the possibility of data entry mistakes.

To do this:

1. using the GUI, create an index for every field that will be a foreign key.
2. create foreign key constaints using the **Relation view** function accessed via the structure tab (see lab session notes week 4).

