/\* Welcome to the SQL mini project. You will carry out this project partly in

the PHPMyAdmin interface, and partly in Jupyter via a Python connection.

This is Tier 1 of the case study, which means that there'll be more guidance for you about how to

setup your local SQLite connection in PART 2 of the case study.

The questions in the case study are exactly the same as with Tier 2.

PART 1: PHPMyAdmin

You will complete questions 1-9 below in the PHPMyAdmin interface.

Log in by pasting the following URL into your browser, and

using the following Username and Password:

URL: <https://sql.springboard.com/>

Username: student

Password: learn\_sql@springboard

The data you need is in the "country\_club" database. This database

contains 3 tables:

i) the "Bookings" table,

ii) the "Facilities" table, and

iii) the "Members" table.

In this case study, you'll be asked a series of questions. You can

solve them using the platform, but for the final deliverable,

paste the code for each solution into this script, and upload it

to your GitHub.

Before starting with the questions, feel free to take your time,

exploring the data, and getting acquainted with the 3 tables. \*/

/\* QUESTIONS

/\* Q1: Some of the facilities charge a fee to members, but some do not.

Write a SQL query to produce a list of the names of the facilities that do. \*/

**SELECT name**

**FROM Facilities**

**WHERE membercost > 0;**

/\* Q2: How many facilities do not charge a fee to members? \*/

**SELECT count(name)**

**FROM Facilities**

**WHERE membercost = 0;**

/\* Q3: Write an SQL query to show a list of facilities that charge a fee to members,

where the fee is less than 20% of the facility's monthly maintenance cost.

Return the facid, facility name, member cost, and monthly maintenance of the

facilities in question. \*/

**SELECT facid, name, membercost, monthlymaintenance**

**FROM Facilities**

**WHERE membercost > 0**

**AND membercost < 0.2\*monthlymaintenance;**

/\* Q4: Write an SQL query to retrieve the details of facilities with ID 1 and 5.

Try writing the query without using the OR operator. \*/

**SELECT \***

**FROM Facilities**

**WHERE facid IN (1,5);**

/\* Q5: Produce a list of facilities, with each labelled as

'cheap' or 'expensive', depending on if their monthly maintenance cost is

more than $100. Return the name and monthly maintenance of the facilities

in question. \*/

**SELECT name, monthlymaintenance, IF (monthlymaintenance < 100, 'Cheap', 'Expensive') AS label**

**FROM Facilities;**

/\* Q6: You'd like to get the first and last name of the last member(s)

who signed up. Try not to use the LIMIT clause for your solution. \*/

**SELECT firstname, surname**

**FROM Members**

**WHERE joindate = (SELECT MAX(joindate) FROM Members);**

/\* Q7: Produce a list of all members who have used a tennis court.

Include in your output the name of the court, and the name of the member

formatted as a single column. Ensure no duplicate data, and order by

the member name. \*/

**SELECT DISTINCT(concat(m.firstname, m.surname)) AS member\_name, f.name AS facility\_name**

**FROM Bookings AS b**

**JOIN Facilities AS f**

**ON b.facid = f.facid**

**JOIN Members AS m**

**ON b.memid = m.memid**

**WHERE b.facid IN (0,1)**

**ORDER BY member\_name, facility\_name;**

/\* Q8: Produce a list of bookings on the day of 2012-09-14 which

will cost the member (or guest) more than $30. Remember that guests have

different costs to members (the listed costs are per half-hour 'slot'), and

the guest user's ID is always 0. Include in your output the name of the

facility, the name of the member formatted as a single column, and the cost.

Order by descending cost, and do not use any subqueries. \*/

**SELECT concat(m.firstname, m.surname) AS member\_name, b.slots\*IF(b.memid = 0, f.guestcost, f.membercost) AS cost**

**FROM Bookings AS b**

**JOIN Members AS m**

**ON b.memid = m.memid**

**JOIN Facilities AS f**

**ON b.facid = f.facid**

**WHERE b.starttime LIKE ('2012-09-14%')**

**HAVING cost > 30**

**ORDER BY cost DESC;**

/\* Q9: This time, produce the same result as in Q8, but using a subquery. \*/

**SELECT concat(m.firstname, m.surname) AS member\_name, subq.slots\*IF(subq.memid = 0, f.guestcost, f.membercost) AS cost**

**FROM**

**(SELECT slots, memid, facid**

**FROM Bookings**

**WHERE starttime LIKE ('2012-09-14%') ) AS subq**

**JOIN Members AS m**

**ON subq.memid = m.memid**

**JOIN Facilities AS f**

**ON subq.facid = f.facid**

**HAVING cost > 30**

**ORDER BY cost DESC;**

/\* PART 2: SQLite

/\* We now want you to jump over to a local instance of the database on your machine.

Copy and paste the LocalSQLConnection.py script into an empty Jupyter notebook, and run it.

Make sure that the SQLFiles folder containing thes files is in your working directory, and

that you haven't changed the name of the .db file from 'sqlite\db\pythonsqlite'.

You should see the output from the initial query 'SELECT \* FROM FACILITIES'.

Complete the remaining tasks in the Jupyter interface. If you struggle, feel free to go back

to the PHPMyAdmin interface as and when you need to.

You'll need to paste your query into value of the 'query1' variable and run the code block again to get an output.

QUESTIONS:

/\* Q10: Produce a list of facilities with a total revenue less than 1000.

The output of facility name and total revenue, sorted by revenue. Remember

that there's a different cost for guests and members! \*/

**SELECT sub1.facility\_name, SUM(cost) AS revenue**

**FROM**

**(SELECT f.name AS facility\_name, b.slots\*IF(b.memid = 0, f.guestcost, f.membercost) AS cost**

**FROM Bookings AS b**

**JOIN Facilities AS f**

**ON b.facid = f.facid) AS sub1**

**GROUP BY sub1.facility\_name**

**HAVING revenue < 1000**

**ORDER BY revenue;**

/\* Q11: Produce a report of members and who recommended them in alphabetic surname,firstname order \*/

**SELECT concat(m1.firstname, m1.surname) AS member\_name, concat(m2.firstname, m2.surname) AS recommender\_name**

**FROM Members AS m1**

**JOIN Members AS m2**

**ON m1.recommendedby = m2.memid**

**WHERE m1.recommendedby > 0**

**ORDER BY member\_name;**

/\* Q12: Find the facilities with their usage by member, but not guests \*/

**SELECT f.name AS facility\_name, COUNT(b.memid) AS member\_usage**

**FROM Bookings AS b**

**JOIN Facilities as f**

**ON b.facid = f.facid**

**WHERE b.memid != 0**

**GROUP BY facility\_name**

**ORDER BY member\_usage;**

/\* Q13: Find the facilities usage by month, but not guests \*/

**SELECT f.name AS facility\_name, EXTRACT(MONTH FROM b.starttime) as month, COUNT(b.memid) AS member\_usage**

**FROM Bookings AS b**

**INNER JOIN Facilities AS f**

**ON b.facid = f.facid**

**WHERE memid != 0**

**GROUP BY facility\_name, month**

**ORDER BY facility\_name, month;**