/\* 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 2 of the case study, which means that there'll be less guidance for you about how to setup

your local SQLite connection in PART 2 of the case study. This will make the case study more challenging for you:

you might need to do some digging, aand revise the Working with Relational Databases in Python chapter in the previous resource.

Otherwise, the questions in the case study are exactly the same as with Tier 1.

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**

**or another way**

**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 f.name, f.facid, f.membercost, f.monthlymaintenance**

**FROM `Facilities` as f**

**WHERE f.membercost < 0.2\* f.monthlymaintenance and f.membercost> 0**

/\* 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 f.name, f.monthlymaintenance,**

**case when f.monthlymaintenance > 100 then 'expensive'**

**else 'cheap' end as type**

**FROM `Facilities` as f**

/\* 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 CONCAT( IFNULL( M.firstName, ' ' ) , ' ', IFNULL( M.surname, ' ' ) ) AS name**

**from Members as M**

**order by date(joindate) desc;**

/\* 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 f.name, CONCAT( IFNULL( M.firstName, ' ' ) , ' ', IFNULL( M.surname, ' ' ) ) AS name**

**from Facilities as f**

**inner join Bookings as B**

**on B.facid=f.facid**

**inner join Members as M**

**on B.memid=M.memid**

**where f.name LIKE "Tennis Court%"**

**order by 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 f.name AS Court, CONCAT( IFNULL( M.firstName, ' ' ) , ' ', IFNULL( M.surname, ' ' ) ) AS name,**

**case when B.memid=0 then B.slots\*f.guestcost**

**else B.slots\*f.membercost end as cost**

**FROM Facilities as f**

**join Bookings as B**

**on f.facid=B.facid**

**join Members as M**

**on B.memid=M.memid**

**WHERE (B.memid != 0 AND f.membercost \* B.slots >30)**

**OR (B.memid = 0 AND f.guestcost \* B.slots >30)**

**and**

**B.starttime LIKE "2012-09-14%"**

**order by cost desc;**

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

**SELECT sub.name AS facility, CONCAT( IFNULL( M.firstName, ' ' ) , ' ', IFNULL( M.surname, ' ' ) ) AS name, sub.cost**

**from (select f.name as name,**

**case when B.memid=0**

**then B.slots\*f.guestcost**

**else B.slots\*f.membercost**

**end as cost,**

**B.starttime AS bstarttime,**

**B.memid AS bmemid**

**FROM Facilities as f**

**join Bookings as B**

**on f.facid=B.facid**

**WHERE (B.memid != 0 AND f.membercost \* B.slots >30)**

**OR (B.memid = 0 AND f.guestcost \* B.slots >30)**

**order by cost desc) as sub**

**join Members as M**

**on sub.bmemid=M.memid**

**where sub.bstarttime LIKE "2012-09-14%"**

/\* PART 2: SQLite

Export the country club data from PHPMyAdmin, and connect to a local SQLite instance from Jupyter notebook

for the following questions.

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 sub.name AS facility, sum(sub.cost) as tot\_rev**

**from (select f.name as name,**

**case when B.memid=0**

**then B.slots\*f.guestcost**

**else B.slots\*f.membercost**

**end as cost,**

**B.memid AS bmemid**

**FROM Facilities as f**

**join Bookings as B**

**on f.facid=B.facid**

**WHERE (B.memid != 0 AND f.membercost \* B.slots)**

**OR (B.memid = 0 AND f.guestcost \* B.slots )**

**order by cost desc) as sub**

**join Members as M**

**on sub.bmemid=M.memid**

**group by sub.name**

**having tot\_rev < 1000**

**order by tot\_rev;**

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

**SELECT CONCAT( IFNULL( m.firstName, ' ' ) , ' ', IFNULL( m.surname, ' ' ) ) AS member\_name, CONCAT( IFNULL( r.firstName, ' ' ) , ' ', IFNULL( r.surname, ' ' ) ) AS recommender\_name**

**FROM Members AS m**

**inner JOIN Members AS r**

**ON (m.recommendedby = r.memid)**

**ORDER BY m.surname, m.firstname desc**

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

**SELECT f.name AS facility, CONCAT( IFNULL( M.firstName, ' ' ) , ' ', IFNULL( M.surname, ' ' ) ) AS name,**

**case when B.memid!=0 then count(B.facid) end as usagebymember**

**FROM Facilities as f**

**join Bookings as B**

**on f.facid=B.facid**

**join Members as M**

**on B.memid=M.memid**

**group by f.name, name**

**order by usagebymember desc;**

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

**SELECT f.name AS facility, CONCAT( IFNULL( M.firstName, ' ' ) , ' ', IFNULL( M.surname, ' ' ) ) AS name,**

**case when B.memid=0 then count(B.facid) end as usagebyguest**

**FROM Facilities as f**

**join Bookings as B**

**on f.facid=B.facid**

**join Members as M**

**on B.memid=M.memid**

**group by name**

**order by usagebyguest desc;**