/\* 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 < (.20\* monthlymaintenance) and 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 = 5 or facid =1

;

/\* 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,

case when (monthlymaintenance > 100) then 'Expensive'

else 'Cheap'

end

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 surname, firstname FROM Members

where CAST(joindate AS Date) = '2012-09-26'

/\* 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(firstname ,"" , surname) AS member, name AS facility

FROM country\_club.Members

INNER JOIN country\_club.Bookings

ON Members.memid = Bookings.memid

INNER JOIN country\_club.Facilities

ON Bookings.facid = Facilities.facid

WHERE name LIKE '%Tennis Court%'

ORDER BY member;

/\* 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(firstname, ' ' , surname) AS member,

name AS facility,

CASE WHEN firstname = 'GUEST' THEN guestcost\*slots

ELSE membercost\*slots END AS cost

FROM Members

INNER JOIN Bookings

ON Members.memid = Bookings.memid

INNER JOIN Facilities

ON Bookings.facid = Facilities.facid

WHERE starttime >= '2012-09-14' AND starttime < '2012-09-15'

AND CASE WHEN firstname = 'GUEST' THEN guestcost\*slots ELSE membercost\*slots END > 30

ORDER BY cost DESC;

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

SELECT

CONCAT(firstname , ' ' , surname) AS member,

name AS facility,

cost

FROM

(SELECT

firstname,

surname,

name,

CASE WHEN firstname = 'GUEST' THEN guestcost\*slots ELSE membercost\*slots END AS cost,

starttime

FROM Members

INNER JOIN Bookings

ON Members.memid = Bookings.memid

INNER JOIN Facilities

ON Bookings.facid = Facilities.facid) AS inner\_table

WHERE starttime >= '2012-09-14' AND starttime < '2012-09-15'

AND cost > 30

ORDER BY cost DESC;

SELECT \* FROM Bookings

INNER JOIN Members ON Members.memid = Bookings.memid

WHERE facid IN (SELECT facid

FROM Facilities

WHERE membercost > 30)

/\* 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 name, sum(slots) as sumslots, sum(guestcost) as totalrevenue from Facilities

inner join Bookings

on Bookings.facid=Facilities.facid

Group by name

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

select m.memid, concat(m.firstname,' ',m.surname) as Member, m.recommendedby,

concat(recom.firstname,' ',recom.surname) as Recommender

from Members m

left join Members recom on m.recommendedby = recom.memid

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

SELECT f.name,concat(m.firstname,' ',m.surname) as Member,

count(f.name) as bookings, slots

FROM Members m

inner join Bookings bk on bk.memid = m.memid

inner join Facilities f on f.facid = bk.facid

where m.memid>0

group by f.name,concat(m.firstname,' ',m.surname)

order by f.name,m.surname,m.firstname

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

SELECT memid, MONTHNAME(starttime) as month, name FROM Bookings

INNER JOIN Facilities

ON Bookings.facid = Facilities.facid

where memid >0