/\* 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, membercost

FROM Facilities

WHERE membership cost>0;

A screenshot of a table

Description automatically generated

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

SELECT name, membercost

FROM Facilities

WHERE membercost=0;

A table with text and numbers

Description automatically generated

/\* 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 name, membercost, monthlymaintenance

FROM Facilities

WHERE membercost>0 AND membercost<monthlymaintenance\*0.2;

A table with numbers and text

Description automatically generated

/\* 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);

A screenshot of a computer

Description automatically generated

/\* 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 AS expensetype

FROM Facilities;

A screenshot of a computer

Description automatically generated

/\* 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, joindate

FROM Members

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

ISSUES

SELECT firstname, surname, MAX(joindate)

FROM Members;

A close up of a number

Description automatically generated

/\* 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 AS facility\_name, CONCAT\_WS(' ', m.firstname,m.surname) AS member\_name

FROM Bookings AS b

INNER JOIN Members AS m

ON b.memid=m.memid

INNER JOIN Facilities AS f

ON b.facid=f.facid

WHERE f.name LIKE '%Tennis Court%'

ORDER BY member\_name;

A screenshot of a tennis match

Description automatically generated

/\* 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 facility\_name,

CONCAT\_WS(' ', m.firstname, m.surname) AS member\_name,

CASE

WHEN b.memid = 0

THEN guestcost \* slots

ELSE membercost \* slots

END AS cost

FROM Bookings AS b

JOIN Facilities AS f

ON b.facid = f.facid

JOIN Members AS m

ON b.memid = m.memid

WHERE b.starttime >= '2012-09-14 00:00:00'

AND b.starttime < '2012-09-15 00:00:00'

AND ((b.memid = 0 AND guestcost \* slots > 30)OR (b.memid<>0 AND membercost \* slots>30))

ORDER BY cost DESC;

A screenshot of a group of rooms

Description automatically generated

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

SELECT facility\_name, member\_name, cost

FROM (

SELECT f.name AS facility\_name,

CONCAT\_WS(' ', m.firstname, m.surname) AS member\_name,

CASE

WHEN b.memid = 0

THEN b.slots \* f.guestcost

ELSE b.slots \* f.membercost

END AS cost

FROM Bookings AS b

INNER JOIN Facilities AS f

ON b.facid = f.facid

INNER JOIN Members AS m

ON b.memid = m.memid

WHERE b.starttime >= '2012-09-14 00:00:00'

AND b.starttime < '2012-09-15 00:00:00') AS sub

WHERE sub.cost > 30

ORDER BY sub.cost DESC;

A screenshot of a table

Description automatically generated

/\* 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

SELECT f.name, SUM(CASE WHEN memid=0

THEN guestcost \* slots

ELSE membercost \* slots

END) AS total\_revenue

FROM Bookings AS b

INNER JOIN Facilities AS f

ON b.facid=f.facid

GROUP BY f.name

HAVING total\_revenue<1000

ORDER BY total\_revenue;

A screenshot of a graph

Description automatically generated

SELECT m.firstname,m.surname, r.surname AS recommeder\_surname, r.firstname AS recommender\_firstname

FROM Members as m

JOIN Members as r

ON m.memid=r.recommendedby

ORDER BY r.surname,r.firstname;

A screenshot of a computer

Description automatically generated

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

SELECT f.facid, f.name AS facility\_name,SUM(b.slots) AS total\_usage

FROM Bookings as b

INNER JOIN Facilities as f

ON b.facid=f.facid

WHERE b.memid != 0

GROUP BY facility\_name

ORDER BY total\_usage DESC;

A screenshot of a table

Description automatically generated

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

SELECT MONTH(b.starttime) AS month, f.name AS facility\_name, SUM(b.slots) AS total\_monthlyusage

FROM Bookings AS b

INNER JOIN Facilities AS f

ON b.facid=f.facid

WHERE b.memid !=0

GROUP BY facility\_name

ORDER BY total\_monthlyusage DESC;