/\* 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 set up 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 AS facility\_name

FROM Facilities

WHERE membercost > 0.0;

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

SELECT COUNT(name) AS number\_facilities\_without\_member\_fee

FROM Facilities

WHERE membercost = 0.0;

/\* 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 AS facility\_name, membercost AS member\_cost, monthlymaintenance AS monthly\_maintenance\_cost

FROM Facilities

WHERE membercost > 0.0

AND membercost < (monthlymaintenance \* .20);

/\* 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,

CASE WHEN monthlymaintenance > 100 THEN ‘expensive’

ELSE ‘cheap’ END

AS cost\_label

FROM Facilities

ORDER BY cost\_label;

/\* 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 AS lastname, MAX(joindate) AS last\_members\_signed\_up

FROM Members

WHERE firstname NOT LIKE 'GUEST'

ORDER BY last\_members\_signed\_up 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 memid, name AS tennis\_court\_name, CONCAT(firstname, " ", surname) AS member\_name

FROM Facilities, Members

WHERE name LIKE 'Tennis%' AND CONCAT(surname, " ", firstname) NOT LIKE 'GUEST%'

ORDER BY member\_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 DISTINCT b.bookid, CONCAT(m.firstname, " ", m.surname) AS member\_guest, f.name AS facility\_name, b.slots, (b.slots \* f.guestcost) AS total\_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 starttime LIKE '2012-09-14%'

AND b.memid = 0

AND (b.slots \* f.guestcost) > 30

UNION

SELECT DISTINCT b.bookid, CONCAT(m.firstname, " ", m.surname) AS member\_guest, f.name AS facility\_name, b.slots, (b.slots \* f.membercost) AS total\_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 starttime LIKE '2012-09-14%'

AND b.memid != 0

AND (b.slots \* f.membercost) > 30

ORDER BY total\_cost DESC;

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

SELECT DISTINCT b.bookid, CONCAT(m.firstname, " ", m.surname) AS member\_guest, f.name AS facility\_name, b.slots, b.starttime,

CASE WHEN b.memid = 0 THEN (f.guestcost\*b.slots)

ELSE (f.membercost\*b.slots)

END

AS total\_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.bookid IN

(SELECT DISTINCT b.bookid

FROM Bookings as b, Facilities as f

WHERE b.facid = f.facid

AND b.memid != 0

AND (f.membercost\*b.slots) > 30

AND b.starttime LIKE '2012-09-14%')

OR b.bookid IN

(SELECT DISTINCT b.bookid

FROM Bookings as b, Facilities as f

WHERE b.facid = f.facid

AND b.memid = 0

AND (f.guestcost\*b.slots)>30

AND b.starttime LIKE '2012-09-14%')

ORDER BY total\_cost DESC;

A more elegent solutions is:

SELECT b.bookid, CONCAT(m.firstname, " ", m.surname) AS member\_guest, f.name AS facility\_name, b.slots,

CASE WHEN b.memid = 0 THEN (f.guestcost\*b.slots)

ELSE (f.membercost\*b.slots)

END

AS total\_cost

FROM Bookings AS b, Facilities AS f, Members AS m

WHERE b.facid = f.facid

AND b.memid = m.memid

AND b.starttime LIKE '2012-09-14%'

AND (CASE WHEN b.memid = 0 THEN (f.guestcost\*b.slots)

ELSE (f.membercost\*b.slots)

END) > 30

ORDER BY total\_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 guestquery.facility\_name AS facility\_name, (guestquery.guest\_booking\_revenue + memberquery.member\_booking\_revenue) AS revenue

FROM (

SELECT f.name AS facility\_name, SUM(f.guestcost \* b.slots) AS guest\_booking\_revenue

FROM Facilities AS f, Bookings AS b

WHERE b.facid = f.facid

AND b.memid=0

GROUP BY facility\_name

ORDER BY guest\_booking\_revenue) guestquery

INNER JOIN (

SELECT f.name AS facility\_name, SUM(f.membercost \* b.slots) AS member\_booking\_revenue

FROM Facilities AS f, Bookings AS b

WHERE b.facid = f.facid

AND b.memid != 0

GROUP BY facility\_name

ORDER BY member\_booking\_revenue) memberquery

ON guestquery.facility\_name = memberquery.facility\_name

WHERE (guestquery.guest\_booking\_revenue + memberquery.member\_booking\_revenue) < 3000

ORDER BY revenue DESC;

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

THIS CODE IS INCOMPLETE AND NEEDS ADDITIONAL WORK TO FULLY ANSWER THE QUESTION.

SELECT CONCAT(surname, ", ", firstname) AS member\_name, memid, recommendedby

FROM Members

WHERE recommendedby LIKE ‘\_%’

ORDER BY surname, firstname;

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

SELECT f.name AS facility\_name, SUM(b.slots) AS member\_use

FROM Facilities as f, Bookings as b

WHERE f.facid = b.facid

AND b.memid != 0

GROUP BY facility\_name

ORDER by member\_use DESC

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

THIS CODE IS INCOMPLETE AND NEEDS ADDITIONAL WORK TO FULLY ANSWER THE QUESTION.

SELECT f.name AS facility\_name, SUM(b.slots) AS member\_use,

CASE WHEN b.starttime LIKE '2012-01%' THEN 'Jan'

WHEN b.starttime LIKE '2012-02%' THEN 'Feb'

WHEN b.starttime LIKE '2012-03%' THEN 'Mar'

WHEN b.starttime LIKE '2012-04%' THEN 'Apr'

WHEN b.starttime LIKE '2012-05%' THEN 'May'

WHEN b.starttime LIKE '2012-06%' THEN 'Jun'

WHEN b.starttime LIKE '2012-07%' THEN 'Jul'

WHEN b.starttime LIKE '2012-08%' THEN 'Aug'

WHEN b.starttime LIKE '2012-09%' THEN 'Sep'

WHEN b.starttime LIKE '2012-10%' THEN 'Oct'

WHEN b.starttime LIKE '2012-11%' THEN 'Nov'

ELSE 'Dec' END

AS month

FROM Facilities as f, Bookings as b

WHERE f.facid = b.facid

AND b.memid != 0

AND b.starttime LIKE '2012-07%'

GROUP BY facility\_name

ORDER BY facility\_name;