```
/* 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.

SELECT name
FROM `Facilities`
WHERE membercost = 0;

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(*)
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 < (monthlymaintenance * .2);

/* 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 name LIKE '%2';

/* 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 cheap_or_expensive
FROM Facilities
GROUP BY monthlymaintenance

/* 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, memid
FROM Members
WHERE memid = (SELECT MAX(memid) 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 CONCAT(surname, ", ", firstname) AS full_name,
        tennis.name
FROM (SELECT DISTINCT memid, name, facid
      FROM Bookings
           INNER JOIN Facilities
```

```
          USING (facid)
      WHERE name LIKE 'Tennis%'
      GROUP BY memid) AS tennis
INNER JOIN Members
     USING (memid)
WHERE surname NOT LIKE "GUEST"
GROUP BY full_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 surname AS Member, name AS Facility,
CASE WHEN Members.memid =0
       THEN Bookings.slots * Facilities.guestcost
     ELSE Bookings.slots * Facilities.membercost
       END AS cost
FROM Members
     JOIN Bookings ON Members.memid = Bookings.memid
     JOIN Facilities ON Bookings.facid
                        = Facilities.facid
WHERE Bookings.starttime >= '2012-09-14'
     AND Bookings.starttime < '2012-09-15'
     AND ((Members.memid =0
          AND Bookings.slots *
          Facilities.guestcost >30)
     OR (Members.memid !=0
          AND Bookings.slots *
          Facilities.membercost >30))
ORDER BY cost DESC
```

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

```
SELECT member, facility, cost
FROM (

     SELECT Members.surname AS member,
     Facilities.name AS facility,
     CASE WHEN Members.memid =0
           THEN Bookings.slots * Facilities.guestcost
           ELSE Bookings.slots * Facilities.membercost
        END AS cost
     FROM Members
     JOIN Bookings
     ON Members.memid = Bookings.memid
     JOIN Facilities
     ON Bookings.facid = Facilities.facid
```

```
      WHERE Bookings.starttime >= '2012-09-14'
            AND Bookings.starttime < '2012-09-15'
      ) AS bookings
WHERE 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 name, totalrevenue
FROM (
      SELECT Facilities.name, SUM(
      CASE WHEN memid =0
            THEN slots * Facilities.guestcost
          ELSE slots * membercost
            END ) AS totalrevenue
FROM Bookings
      INNER JOIN Facilities
      ON Bookings.facid = Facilities.facid
GROUP BY Facilities.name
) AS selected_facilities
WHERE totalrevenue <=1000
ORDER BY totalrevenue

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

SELECT r.memid AS 'Mem ID',
```

```
        CONCAT(r.surname, ", ", r.firstname) AS Member,
       m.memid as 'Rec By Mem ID',
        CONCAT(m.surname, ", ", m.firstname) AS 'Recommended By'
FROM Members AS m
      INNER JOIN Members AS r
      ON m.memid = r.recommendedby
WHERE r.recommendedby > 0
ORDER BY Member
```

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

```
SELECT f.facid,
        f.name,
        COUNT(b.bookid * b.slots) AS 'Usage'
FROM Bookings as b
     LEFT JOIN Facilities as f
ON f.facid = b.facid
WHERE b.memid > 0
GROUP BY b.facid
ORDER BY b.facid
```

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

```
SELECT f.facid, f.name, SUM(b.slots), EXTRACT(Month FROM b.starttime)
FROM Bookings as b
LEFT JOIN Facilities as f
     ON f.facid = b.facid
WHERE b.memid > 0
GROUP BY f.facid, EXTRACT(Month from b.starttime)
```