```
/* 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. */
SQL Query for Q1
SELECT name
FROM `Facilities
WHERE membercost = 0;


/* Q2: How many facilities do not charge a fee to members? */
SQL Query for Q2
SELECT COUNT (*) As num_free_facilities
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. */


SQL Query for Q3

SELECT  facid, name, membercost, monthlymaintenance
FROM `Facilities`
WHERE  membvercost > 0
 AND membercost < 0.2 * monthlymaintenance;

/* 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. */
SQL Query for Q4
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;


/* 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
FROM Members
WHERE joindate = (SELECT MAX(joindate) 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 * FROM Bookings;
SELECT facid FROM Bookings WHERE facid IS NOT NULL;


/* 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,
      CASE
            WHEN b.memid = 0 THEN 'Guest'
            ELSE CONCAT(m.firstname, ' ', m.surname)
        END AS member_name,
       (b.slots * CASE
            WHEN b.memid = 0 THEN f.guestcost
            ELSE F.MEMBERCOST
       END) AS cost
FROM Bookings b
JOIN Fcailities f ON b.facid = f.facid
LEFT JOIN Members m ON b.memid = m.memid
WHERE DATE (b.starttime)  '2012-09-14'
      AND (b.slots * CASE
          WHEN b.memid = 0 THEN f. guestcost
          ELSE f.membercost
        END) > 30
 ORDER BY cost DESC;


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


/* 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: SELECT
```

```
      f.name AS facility,
FROM (
    SELECT
      f.name AS facility,
      CASE
            WHEN b.memid = 0 THEN 'Guest'
            ELSE CONCAT(m.firstname, ' ', m.surname)
        END AS member_name,
       (b.slots * CASE
            WHEN b.memid = 0 THEN f.guestcost
            ELSE F.MEMBERCOST
       END) AS cost
FROM Bookings b
JOIN Fcailities f ON b.facid = f.facid
LEFT JOIN Members m ON b.memid = m.memid
WHERE DATE (b.starttime)  '2012-09-14'
) AS sub
WHERE cost > 30
ORDER BY cost DESC;
/* 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
      f.name AS facility,
      SUM(
        b.slots * CASE
            WHEN b.memid = 0 THEN f.guestcost
            ELSE f.membercost
        END
  ) AS revenue
FROM Bookings b
JOIN Facilities f ON b.facid = f.facid
GROUP BY f.name
HAVING revenue < 1000
ORDER BY revenue;

/* Q11: Produce a report of members and who recommended them in
alphabetic surname,firstname order */
SELECT
      CONCAT(m.firstname, ' ', m.surname) AS member_name,
      CONCAT(r.firstname, ' ', r.surnam) AS recommended_by
  FROM Members m
  LEFT JOIN M embers r ON m.recommendedby = r.memid
  ORDER BY m.surname, m.firstname;


/* Q12: Find the facilities with their usage by member, but not guests */
SELECT
      f.name AS facility,
      CONCAT(m.firstname, ' ', m.surname) AS member_name
FROM Bookings
JOIN Facilities f ON b.facid = f.facid
JOIN Members m ON b.memid = m.memid
```

```
WHERE b.memid != 0
ORDER BY f.name, member_name;


/* Q13: Find the facilities usage by month, but not guests */
SELECT
      f.name AS facility,
      DATE_FORMAT(b.starttime, '%y-%m') AS usage_month,
      COUNT(b.memid) AS usage_count
FROM Bookings b
JOIN Facilities f ON b.facid = f.facid
WHERE b.memid != 0
GROUP BY f.name, usage_month
ORDER BY f.name, usage_month;
```