/* 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 2 of the case study, which means that there'll be less guidance for you about how to setup
your local SQLite connection in PART 2 of the case study. This will make the case study more challenging for you:
you might need to do some digging, aand revise the Working with Relational Databases in Python chapter in the previous resource.

Otherwise, the questions in the case study are exactly the same as with Tier 1.

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 *
FROM Facilities
WHERE membercost > 0
```

or for just the names:

```
SELECT name
FROM Facilities
WHERE membercost > 0
```

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

```
SELECT *
FROM Facilities
WHERE membercost = 0
```

or for the number of facilites:

```
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 < 0.2 * 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. */

```
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 'cheap'
ELSE 'expensive'
END AS IsCheap
FROM Facilities
```

/* Q6: You'd like to get the first and last name of the last member(s) who signed up.*/

```
SELECT firstname, surname
FROM Members
ORDER BY joindate DESC
LIMIT 1
```

**/* 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
CASE WHEN f.facid = 0 THEN CONCAT_WS(m.firstname, m.surname, ': Tennis Court 1')
WHEN f.facid = 1 THEN CONCAT_WS (m.firstname, m.surname, ': Tennis Court 2')
END AS courtrecords
FROM Bookings AS b
LEFT JOIN Members AS m
ON b.memid = m.memid
LEFT JOIN Facilities AS f
ON b.facid = f.facid
WHERE f.facid IN (0,1)
GROUP BY m.firstname, m.surname, f.facid
ORDER BY m.firstname, m.surname
```

**/* 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,
CASE WHEN m.memid = 0 THEN m.firstname
ELSE CONCAT (m.firstname, ' ', m.surname)
END AS name,
CASE WHEN m.memid = 0 THEN b.slots * f.guestcost
ELSE b.slots * f.membercost
END AS dayscost
FROM Bookings AS b
LEFT JOIN Members AS m
ON b.memid = m.memid
LEFT JOIN Facilities AS f
ON b.facid = f.facid
WHERE starttime BETWEEN '2012-09-14 00:00:00' AND '2012-09-14 23:59:59'
```

```
AND 30 < CASE WHEN m.memid = 0 THEN b.slots * f.guestcost
ELSE b.slots * f.membercost
END
ORDER BY
CASE WHEN m.memid = 0 THEN b.slots * f.guestcost
ELSE b.slots * f.membercost
END DESC
```

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

```
SELECT subq.facility, subq.name, subq.dayscost
FROM
      (SELECT f.name as facility, b.starttime AS starttime,
      CASE
            WHEN m.memid = 0 THEN m.firstname
            ELSE CONCAT (m.firstname, ' ', m.surname)
            END AS name,
      CASE
            WHEN m.memid = 0 THEN b.slots * f.guestcost
            ELSE b.slots * f.membercost
            END AS dayscost

    FROM Bookings AS b
            LEFT JOIN Members AS m
            ON b.memid = m.memid
      LEFT JOIN Facilities AS f
                  ON b.facid = f.facid
) AS subq
WHERE
      subq.starttime BETWEEN '2012-09-14 00:00:00' AND '2012-09-14
      23:59:59'
      AND subq.dayscost > 30
ORDER BY subq.dayscost DESC
```

/* PART 2: SQLite

Export the country club data from PHPMyAdmin, and connect to a local SQLite instance from Jupyter notebook
for the following questions.

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! */

```
run = con.execute('''SELECT subq.name, SUM(subq.tot_rev) AS TotalRev
FROM (SELECT f.name AS name, f.facid AS facid,
CASE WHEN b.memid = 0 THEN b.slots * f.guestcost
```

```
ELSE b.slots * f.membercost
END AS tot_rev
FROM Bookings AS b
LEFT JOIN Facilities AS f
ON b.facid = f.facid
) AS subq
GROUP BY subq.facid
HAVING TotalRev < 1000''')
df = pd.DataFrame(run.fetchall())
df
```

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

```
run = con.execute('''SELECT fm.surname, fm.firstname, sm.surname,
sm.firstname
FROM Members as fm
LEFT JOIN Members as sm
ON fm.memid = sm.recommendedby
ORDER BY fm.surname, fm.firstname, sm.surname, sm.firstname
''')
df = pd.DataFrame(run.fetchall())
df
```

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

```
run = con.execute('''SELECT subq.fname AS facilityname, SUM(subq.memu)
AS memusage
FROM
(SELECT f.name AS fname, f.facid AS facid,
CASE WHEN b.memid = 0 THEN 0
ELSE b.slots * 1
END AS memu
FROM Bookings as b
LEFT JOIN Facilities AS f
ON b.facid = f.facid
) AS subq
GROUP BY subq.facid
''')
df = pd.DataFrame(run.fetchall())
df
```

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

```
run = con.execute('''SELECT subq.month AS month, subq.fname AS
facilityname, SUM(subq.memu) AS memusage
FROM
(SELECT f.name AS fname, f.facid AS facid, strftime('%m', b.starttime)
AS Month,
CASE WHEN b.memid = 0 THEN 0
ELSE b.slots * 1
```

```
END AS memu
FROM Bookings as b
LEFT JOIN Facilities AS f
ON b.facid = f.facid
) AS subq
GROUP BY subq.month, subq.facid''')
df = pd.DataFrame(run.fetchall())
df
```