* 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. */

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.

1.
   ```
   SELECT name
   FROM `Facilities`
   WHERE membercost !=0
   ```

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

2.
   ```
   SELECT COUNT( name ) AS noCostFacilties
   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. */

```
3. SELECT facid, name, membercost, monthlymaintenance
   FROM Facilities
   WHERE membercost !=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.

```
4. SELECT *
   FROM Facilities
   HAVING MOD(facid, 4 ) =1
```

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.

```
5. SELECT name, monthlymaintenance,
           CASE WHEN monthlymaintenance >100 THEN 'expensive'
           ELSE 'cheap' END AS affordability
   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.

```
6. 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.

```
7. SELECT DISTINCT CONCAT( m.firstname, ', ', m.surname ) AS fullname,
   f.name
   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.facid =0 OR b.facid =1
ORDER BY fullname
```

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.

```
8.  SELECT CONCAT( m.firstname, ', ', m.surname ) AS fullname, f.name,
    CASE WHEN b.memid =0 THEN f.guestcost * b.slots
            ELSE f.membercost * b.slots END AS cost
    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 b.starttime LIKE "2012-09-14%"
    HAVING cost >30
    ORDER BY fullname
```

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

```
9.  SELECT name, fullname, cost
    FROM (
            SELECT CONCAT( m.firstname, ', ', m.surname ) AS fullname, f.name,
                    CASE WHEN b.memid =0 THEN f.guestcost * b.slots
                    ELSE f.membercost * b.slots END AS cost
            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 b.starttime LIKE "2012-09-14%"
            ORDER BY fullname
    ) AS subquery
    HAVING cost >30
```

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!

```
10.SELECT f.name, SUM(
          CASE WHEN b.memid =0 THEN f.guestcost * b.slots
                    ELSE f.membercost * b.slots END ) AS total_revenue
     FROM Bookings AS b
     INNER JOIN Members AS m ON b.memid = m.memid
     INNER JOIN Facilities AS f ON b.facid = f.facid
     GROUP BY f.name
     HAVING total_revenue < 1000
     ORDER BY total_revenue
```

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

```
11.SELECT CONCAT( Members.firstname, ', ', surname ) AS member,
     Recommender.fullname AS recommender
     FROM Members
     INNER JOIN (
          SELECT memid, CONCAT( firstname, ', ', surname ) AS fullname
          FROM Members) AS Recommender
     ON Members.recommendedby = Recommender.memid
     WHERE recommendedby BETWEEN 1 AND 30
     ORDER BY member
```

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

```
12.SELECT f.name AS facility, SUM( b.slots ) AS totalMemberUsage
     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 b.memid !=0
     GROUP BY f.name
```

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

```
13.SELECT f.name AS facility,
          MONTH( b.starttime ) AS monthOfUsage,
          SUM( b.slots ) AS totalMemberUsage
     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 b.memid !=0
     GROUP BY f.name, monthOfUsage
     ORDER BY monthOfUsage
```