The code below is the completed "Tier 2" of Springboard's "SQL Case Study - Country Club" for the Data Science Career Track. The first 9 questions were completed using PostgreSQL in PHPMyAdmin, before completing the remaining questions in this Jupyter Notebook using python and the sqlite3 package.

In [1]:

```
"""/* 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
FROM Facilities
WHERE membercost >0



/* Q2: How many facilities do not charge a fee to members? */
SELECT COUNT(name)
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*0.2);

/* 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 facid > 0
AND facid NOT BETWEEN 2 AND 4
AND facid < 6;

/* 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 label
FROM Facilities
ORDER BY label DESC;



/* 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, MAX(joindate) as joindate
FROM Members
WHERE firstname <> "GUEST";


/* 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 f.name AS 'facility',
        CONCAT_WS(" ", m.firstname, m.surname) AS 'member'
FROM Bookings as b
```

```
LEFT JOIN Facilities as f
        ON b.facid = f.facid
LEFT JOIN Members as m
        ON b.memid = m.memid
WHERE f.name LIKE 'Tennis Court %'
GROUP BY member
ORDER BY member;


/* 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', CONCAT_WS(" ", m.firstname, m.surname) AS 'member',
    CASE
        WHEN
            m.memid > 0 THEN slots*membercost
        WHEN
            m.memid = 0 THEN slots*guestcost
        END AS 'cost'
FROM Bookings as b
LEFT JOIN Facilities as f
    ON b.facid = f.facid
LEFT JOIN Members as m
    ON b.memid = m.memid
WHERE date(b.starttime) = '12-09-14' HAVING cost > 30
ORDER BY cost DESC;

/* Q9: This time, produce the same result as in Q8, but using a subquery. */
SELECT
 name as 'facility', CONCAT_WS(" ", s.firstname, s.surname) AS 'member', cost
FROM
  (
  SELECT
    firstname,
    surname,
    name,
    CASE
        WHEN firstname = 'GUEST'
            THEN guestcost * slots
        ELSE membercost * slots
    END AS cost,
    starttime
  FROM
    Members
  INNER JOIN Bookings
  ON Members.memid = Bookings.memid
  INNER JOIN Facilities
  ON Bookings.facid = Facilities.facid
  ) AS s
WHERE
  starttime = '2012-09-14'
 AND cost > 30
ORDER BY cost DESC;""";
```

Below are the questions completed inside this notebook using python/sqlite3:

In [2]:

```
import sqlite3
import pandas as pd # Using pandas to clean up appearances of query results
def sql(query): # defined function because I'm lazy and didn't want keep typing methods
    connection = sqlite3.connect('sqlite_db_pythonsqlite.db')
    cur = connection.cursor().execute
```

```
    return pd.DataFrame(cur(query).fetchall())

sql('Select * from Facilities') #Function Test
```

Out[2]:

	0	1	2	3	4	5
0	0	Tennis Court 1	5.0	25.0	10000	200
1	1	Tennis Court 2	5.0	25.0	8000	200
2	2	Badminton Court	0.0	15.5	4000	50
3	3	Table Tennis	0.0	5.0	320	10
4	4	Massage Room 1	9.9	80.0	4000	3000
5	5	Massage Room 2	9.9	80.0	4000	3000
6	6	Squash Court	3.5	17.5	5000	80
7	7	Snooker Table	0.0	5.0	450	15
8	8	Pool Table	0.0	5.0	400	15

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!

In [3]:

```
db=sql("""SELECT f.name,
    SUM(CASE
        WHEN b.memid <>0
            THEN f.membercost*b.slots
        ELSE f.guestcost*b.slots
        END) AS revenue
    FROM Bookings as b
    INNER JOIN Facilities as f
        ON b.facid = f.facid
    GROUP BY f.name
        HAVING revenue > 1000
    ORDER BY revenue DESC;
        """)
db.rename(columns={0:'Facility', 1:'Total Revenue'})
```

Out[3]:

	Facility	Total Revenue
0	Massage Room 1	50351.6
1	Massage Room 2	14454.6
2	Tennis Court 2	14310.0
3	Tennis Court 1	13860.0
4	Squash Court	13468.0
5	Badminton Court	1906.5

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

In [4]:

```
db=sql("""
    SELECT m.surname,
    m.firstname,
    m.recommendedby AS recomender_id,
    r.surname || " " ||r.firstname  AS 'Recommended By'
    FROM Members AS m
```

```
    LEFT JOIN Members AS r
        ON m.recommendedby = r.memid
    WHERE m.recommendedby != 0
    ORDER BY r.surname;""")
db.rename(columns={0:'Last Name', 1:'First Name', 2:'Recommending ID', 3:'Recommended By'}
```

Out[4]:

	Last Name	First Name	Recommending ID	Recommended By
0	GUEST	GUEST		None
1	Smith	Darren		None
2	Smith	Tracy		None
3	Rownam	Tim		None
4	Tracy	Burton		None
5	Farrell	Jemima		None
6	Farrell	David		None
7	Tupperware	Hyacinth		None
8	Smith	Darren		None
9	Sarwin	Ramnaresh	15	Bader Florence
10	Coplin	Joan	16	Baker Timothy
11	Genting	Matthew	5	Butters Gerald
12	Baker	Timothy	13	Farrell Jemima
13	Pinker	David	13	Farrell Jemima
14	Rumney	Henrietta	20	Genting Matthew
15	Jones	Douglas	11	Jones David
16	Dare	Nancy	4	Joplette Janice
17	Jones	David	4	Joplette Janice
18	Hunt	John	30	Purview Millicent
19	Boothe	Tim	3	Rownam Tim
20	Joplette	Janice	1	Smith Darren
21	Butters	Gerald	1	Smith Darren
22	Owen	Charles	1	Smith Darren
23	Smith	Jack	1	Smith Darren
24	Mackenzie	Anna	1	Smith Darren
25	Worthington-Smyth	Henry	2	Smith Tracy
26	Purview	Millicent	2	Smith Tracy
27	Crumpet	Erica	2	Smith Tracy
28	Baker	Anne	9	Stibbons Ponder
29	Bader	Florence	9	Stibbons Ponder
30	Stibbons	Ponder	6	Tracy Burton

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

In [5]:

```
db=sql("""
```

```
SELECT b.facid,
    COUNT(b.memid) AS 'usage',
    f.name
FROM
    (SELECT facid, memid FROM Bookings WHERE memid <>0) AS b
LEFT JOIN Facilities AS f
    ON b.facid = f.facid
GROUP BY b.facid;
""")
db.rename(columns={0:"Facility ID", 1:'Usage', 2:'Facility'})
```

Out[5]:

	Facility ID	Usage	Facility
0	0	308	Tennis Court 1
1	1	276	Tennis Court 2
2	2	344	Badminton Court
3	3	385	Table Tennis
4	4	421	Massage Room 1
5	5	27	Massage Room 2
6	6	195	Squash Court
7	7	421	Snooker Table
8	8	783	Pool Table

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

In [6]:

```
db=sql("""
    SELECT b.months,
    COUNT (b.memid) AS 'usage'
    FROM(
        SELECT strftime('%m', starttime) AS "months", memid
        FROM Bookings
        WHERE memid <>0) AS b
    GROUP BY b.months;
""")
db.rename(columns ={0:'Month', 1:'Usage'})
```

Out[6]:

	Month	Usage
0	07	480
1	08	1168
2	09	1512