8.3.3 SQL Case Study - Country Club

**/* 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;
```

/* Q2: How many facilities do not charge a fee to members? */
ANSWER = 4

```
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 < (.20 * 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 'expensive'
        ELSE 'cheap'
END AS sub
FROM Facilities;
```

Prepared by: **Prem Ananda**, Springboard Data Science
07.15.2020

/* 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 last
FROM `Members`
WHERE firstname != 'GUEST'
ORDER BY `Members`.`joindate`  DESC;
```

**/* 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(m.surname, ',  ', m.firstname) AS member_name, f.name AS court_name
FROM Members AS m
LEFT JOIN Bookings as b
        ON b.memid = m.memid
LEFT JOIN Facilities as f
        ON f.facid = b.facid
WHERE f.name LIKE '%enni%' AND f.name NOT LIKE '%abl%'
HAVING member_name NOT LIKE '%UES%'
ORDER BY member_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 f.name AS facility,
        CONCAT(m.surname, ',  ', m.firstname) AS member_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 Facilities AS f
        ON f.facid = b.facid
        AND b.starttime LIKE '2012-09-14%'
        AND (((b.memid =0) AND (f.guestcost * b.slots >30))
        OR ((b.memid !=0)
        AND (f.membercost * b.slots > 30)))
INNER JOIN Members AS m
        ON m.memid = b.memid
ORDER BY cost DESC;
```

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

```
SELECT *
FROM (SELECT f.name AS facility, CONCAT(m.firstname, ',  ', m.surname ) AS member_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 Facilities AS f
        ON b.facid = f.facid
AND b.starttime LIKE '2012-09-14%'
INNER JOIN Members AS m
        ON b.memid = m.memid) AS sub
WHERE sub.cost  > 30
ORDER BY sub.cost DESC;
```

/* PART 2: SQLite

**/* 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,
      total_revenue
FROM
        (SELECT name,
        SUM(CASE WHEN b.memid = 0 THEN guestcost * slots
                ELSE membercost * slots END) AS total_revenue
        FROM Bookings AS b
        JOIN Facilities AS f
        ON b.facid = f.facid
        GROUP BY name) AS subquery
WHERE total_revenue < 1000
ORDER BY total_revenue;
```

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

```
SELECT DISTINCT(CONCAT(m1.surname, ', ', m1.firstname)) AS member_name,
        m2.recommendedby AS recommender
FROM Members as m1
JOIN Members as m2
        ON  m2.recommendedby = m1.memid
WHERE m1.memid != 0
ORDER BY member_name;
```

Prepared by: **Prem Ananda**, Springboard Data Science
07.15.2020

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

```
SELECT b.memid, f.name, b.slots
FROM Facilities AS f
LEFT JOIN Bookings AS b
        ON f.facid = b.facid
WHERE b.memid != 0
GROUP BY b.memid;
```

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

```
SELECT MONTH(b.starttime) AS month, f.name, b.slots, b.memid
FROM Facilities AS f
LEFT JOIN Bookings AS b
ON f.facid = b.facid
WHERE b.memid != 0
GROUP BY month;
```