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?

There are 4 facilities that are free to members.

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 / monthlymaintenance < .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 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 value
    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 * 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
  CONCAT_WS(" ", m.firstname, m.surname) AS member,
  f.name AS facility
  FROM Facilities AS f
```

```
JOIN Bookings AS b
ON b.facid = f.facid
JOIN Members AS m
ON m.memid = b.memid
WHERE f.name LIKE 'Tennis%'
AND m.firstname != 'GUEST';
```

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
   CONCAT_WS(' ', m.firstname, m.surname) AS member,
   f.name AS facility,
   CASE WHEN m.memid = 0 THEN f.guestcost * slots
   ELSE f.membercost * slots END AS cost
   FROM Bookings AS b
   INNER JOIN Members AS m ON m.memid = b.memid
   INNER JOIN Facilities AS f ON f.facid = b.facid
   WHERE DATE(starttime) = '2012-09-14'
   AND m.memid = 0
   AND f.guestcost * slots > 30
   OR DATE(starttime) = '2012-09-14'
   AND f.membercost * slots > 30;
```

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

```
SELECT
   (SELECT CONCAT_WS(' ', firstname, surname) AS name FROM Members AS m WHERE
b.memid = m.memid) AS fullname,
   (SELECT slots FROM Facilities WHERE b.facid = facid) *
   (SELECT CASE WHEN b.memid = 0 THEN guestcost ELSE membercost END AS cost
FROM Facilities WHERE b.facid = facid) AS totalcost
   FROM Bookings AS b
   WHERE (SELECT slots FROM Facilities WHERE b.facid = facid)*
   (SELECT CASE WHEN b.memid = 0 THEN guestcost ELSE membercost END AS cost
FROM Facilities WHERE b.facid = facid) >
   30 AND DATE(starttime) = '2012-09-14';
```

PART 2: SQLite

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!

```
import pandas as pd
import sqlite3

conn = sqlite3.connect('./sqlite_db_pythonsqlite.db')

conn.execute("SELECT * FROM Facilities;”)

cursor = conn.cursor()

cursor.fetchall()

facdf = pd.read_sql_query("SELECT * from Facilities;", conn)

bookdf = pd.read_sql_query("SELECT * from Bookings;", conn)

guest_visits_dict = bookdf[bookdf['memid'] == 0].groupby(‘facid')['slots'].sum().to_dict()

member_visits_dict = bookdf[bookdf['memid'] != 0].groupby('facid')['slots'].sum().to_dict()

revenue_dict = {}

for k,v in guest_visits_dict.items():
    revenue_dict[facdf.iloc[k]['name']] = \
        v * facdf.iloc[k]['guestcost'] + member_visits_dict[k] * facdf.iloc[k]['membercost']

revenue_series = pd.Series(revenue_dict)

revenue_series.sort_values(ascending = False, inplace=True)

revenue_series[revenue_series < 1000]
```

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

```
memdf = pd.read_sql_query("SELECT m1.surname || ', ' || m1.firstname
        AS member, m2.surname || ', ' || m2.firstname AS recommender FROM Members as m1
        INNER JOIN Members AS m2 ON m2.memid = m1.recommendedby;", conn)

memdf.sort_values(by='member')
```

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

```
facility_use_by_members = bookdf[bookdf['memid'] != 0]

zipped_fac_use_mem = \
        zip(facdf[‘name’],facility_use_by_members.groupby('facid')['memid'].count())

fac_use_mem_df = pd.DataFrame(zipped_fac_use_mem, columns=['facility',
‘count']).set_index('facility')
```

```
fac_use_mem_df.sort_values('count', ascending=False)
```

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

```
from datetime import datetime

bookdf['starttime'] = pd.to_datetime(bookdf['starttime']).dt.to_period('M')

fac_use_month_join = bookdf.join(facdf, on=['facid'], lsuffix='_')

fac_use_month_df = pd.DataFrame(fac_use_month_join.groupby(['name','starttime'])
['facid'].count())
```