In [11]:
```
# Import necessary module
# from sqlalchemy import create_engine

import os
# import pymysql
import pandas as pd
import mysql.connector
from mysql.connector import errorcode
```

In [31]:
```
# from https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html
try:
    db = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="country_club"
        )
except mysql.connector.Error as err:
  if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
    print("Something is wrong with your user name or password")
  elif err.errno == errorcode.ER_BAD_DB_ERROR:
    print("Database does not exist")
  else:
    print(err)
# else:
#   cnx.close()
# db.close()   # at some point need to close the connection with this instruction

cursor = db.cursor(buffered=True)  # to avoid [error](https://stackoverflow.com/questions/29772337/python-mysql-connector-unread-result-found-when-using-fetchone)
```

In [30]:
```
cursor.close()
db.close()
```

In [34]:
```
# Build select statement for members table: stmt
query = 'SELECT * FROM members'

# Execute the statement and fetch the results: results
cursor.execute(query)

rows = cursor.fetchall()    # get all selected rows
for r in rows:
    print(r)
```

```
(0, 'GUEST', 'GUEST', 'GUEST', 0, '(000) 000-0000', '', '2012-07-01 00:00:0
0')
(1, 'Smith', 'Darren', '8 Bloomsbury Close, Boston', 4321, '555-555-5555',
'', '2012-07-02 12:02:05')
(2, 'Smith', 'Tracy', '8 Bloomsbury Close, New York', 4321, '555-555-5555',
'', '2012-07-02 12:08:23')
(3, 'Rownam', 'Tim', '23 Highway Way, Boston', 23423, '(844) 693-0723', '',
'2012-07-03 09:32:15')
(4, 'Joplette', 'Janice', '20 Crossing Road, New York', 234, '(833) 942-471
0', '1', '2012-07-03 10:25:05')
(5, 'Butters', 'Gerald', '1065 Huntingdon Avenue, Boston', 56754, '(844) 078-
4130', '1', '2012-07-09 10:44:09')
(6, 'Tracy', 'Burton', '3 Tunisia Drive, Boston', 45678, '(822) 354-9973',
'', '2012-07-15 08:52:55')
(7, 'Dare', 'Nancy', '6 Hunting Lodge Way, Boston', 10383, '(833) 776-4001',
'4', '2012-07-25 08:59:12')
(8, 'Boothe', 'Tim', '3 Bloomsbury Close, Reading, 00234', 234, '(811) 433-25
47', '3', '2012-07-25 16:02:35')
(9, 'Stibbons', 'Ponder', '5 Dragons Way, Winchester', 87630, '(833) 160-390
0', '6', '2012-07-25 17:09:05')
(10, 'Owen', 'Charles', '52 Cheshire Grove, Winchester, 28563', 28563, '(855)
542-5251', '1', '2012-08-03 19:42:37')
(11, 'Jones', 'David', '976 Gnats Close, Reading', 33862, '(844) 536-8036',
'4', '2012-08-06 16:32:55')
(12, 'Baker', 'Anne', '55 Powdery Street, Boston', 80743, '844-076-5141',
'9', '2012-08-10 14:23:22')
(13, 'Farrell', 'Jemima', '103 Firth Avenue, North Reading', 57392, '(855) 01
6-0163', '', '2012-08-10 14:28:01')
(14, 'Smith', 'Jack', '252 Binkington Way, Boston', 69302, '(822) 163-3254',
'1', '2012-08-10 16:22:05')
(15, 'Bader', 'Florence', '264 Ursula Drive, Westford', 84923, '(833) 499-352
7', '9', '2012-08-10 17:52:03')
(16, 'Baker', 'Timothy', '329 James Street, Reading', 58393, '833-941-0824',
'13', '2012-08-15 10:34:25')
(17, 'Pinker', 'David', '5 Impreza Road, Boston', 65332, '811 409-6734', '1
3', '2012-08-16 11:32:47')
(20, 'Genting', 'Matthew', '4 Nunnington Place, Wingfield, Boston', 52365,
'(811) 972-1377', '5', '2012-08-19 14:55:55')
(21, 'Mackenzie', 'Anna', '64 Perkington Lane, Reading', 64577, '(822) 661-28
98', '1', '2012-08-26 09:32:05')
(22, 'Coplin', 'Joan', '85 Bard Street, Bloomington, Boston', 43533, '(822) 4
99-2232', '16', '2012-08-29 08:32:41')
(24, 'Sarwin', 'Ramnaresh', '12 Bullington Lane, Boston', 65464, '(822) 413-1
470', '15', '2012-09-01 08:44:42')
(26, 'Jones', 'Douglas', '976 Gnats Close, Reading', 11986, '844 536-8036',
'11', '2012-09-02 18:43:05')
(27, 'Rumney', 'Henrietta', '3 Burkington Plaza, Boston', 78533, '(822) 989-8
876', '20', '2012-09-05 08:42:35')
(28, 'Farrell', 'David', '437 Granite Farm Road, Westford', 43532, '(855) 755
-9876', '', '2012-09-15 08:22:05')
(29, 'Worthington-Smyth', 'Henry', '55 Jagbi Way, North Reading', 97676, '(85
5) 894-3758', '2', '2012-09-17 12:27:15')
(30, 'Purview', 'Millicent', '641 Drudgery Close, Burnington, Boston', 34232,
'(855) 941-9786', '2', '2012-09-18 19:04:01')
(33, 'Tupperware', 'Hyacinth', '33 Cheerful Plaza, Drake Road, Westford', 686
66, '(822) 665-5327', '', '2012-09-18 19:32:05')
(35, 'Hunt', 'John', '5 Bullington Lane, Boston', 54333, '(899) 720-6978', '3
```

```
0', '2012-09-19 11:32:45')
(36, 'Crumpet', 'Erica', 'Crimson Road, North Reading', 75655, '(811) 732-481
6', '2', '2012-09-22 08:36:38')
(37, 'Smith', 'Darren', '3 Funktown, Denzington, Boston', 66796, '(822) 577-3
541', '', '2012-09-26 18:08:45')
```

SQL mini project question #1

In [52]:
```
query = 'SELECT name FROM facilities WHERE membercost > 0'

cursor.execute(query)

q1result = cursor.fetchall()

print('These are the facilites that charge members:')

for r in q1result:
    facility = r[0]
    print(facility)
```

```
These are the facilites that charge members:
Tennis Court 1
Tennis Court 2
Massage Room 1
Massage Room 2
Squash Court
```

SQL mini project question #2

In [51]:
```
query = 'SELECT COUNT( name )FROM Facilities WHERE membercost = 0'

cursor.execute(query)

q2result = cursor.fetchall()

for r in q2result:
    num1 = r[0] # convert the tuple to an integer
    print('There are', num1, 'free facilities.')
```

```
There are 4 free facilities.
```

SQL mini project question #3

In [59]:
```
query = 'SELECT * FROM facilities WHERE membercost < ( monthlymaintenance * 0.
2 )'

cursor.execute(query)

q3result = cursor.fetchall()

print('Facilites where membercost < 20% of montlymaintenance:\n')
print('ID','Name')
for r in q3result:
    facid = r[0] # convert the tuple to an integer
    facility = r[1]
    print(facid, facility)
```

```
Facilites where membercost < 20% of montlymaintenance:

ID Name
0 Tennis Court 1
1 Tennis Court 2
2 Badminton Court
3 Table Tennis
4 Massage Room 1
5 Massage Room 2
6 Squash Court
7 Snooker Table
8 Pool Table
```

SQL mini project question #4

In [61]:
```
query = 'SELECT * FROM facilities WHERE facid BETWEEN 1 AND 5 '

cursor.execute(query)

q4result = cursor.fetchall()

print('Facilites where facid in 1 to 5:\n')
print('ID','Name')
for r in q4result:
    facid = r[0] # convert the tuple to an integer
    facility = r[1]
    print(facid, facility)
```

```
Facilites where facid in 1 to 5:

ID Name
1 Tennis Court 2
2 Badminton Court
3 Table Tennis
4 Massage Room 1
5 Massage Room 2
```

SQL mini project question #5

In [100]:
```
query = "SELECT name, monthlymaintenance, CASE WHEN monthlymaintenance > 100 T
HEN 'expensive' ELSE 'cheap' END AS 'cheapexpensive' FROM facilities;"

cursor.execute(query)

q5result = cursor.fetchall()

data = {}
df5 = pd.DataFrame(data)

print('Facilites where expensive is > $100 per month:\n')

# Both print result to screen and append to dataframe
for r in q5result:
    name = r[0] # convert the tuple to an integer
    monthlymaint = r[1]
    cheapexpensive = r[2]
    data = {
        'name': r[0],
        'monthlymaint': r[1],
        'cheapexpensive': r[2]
    }
    df5 = df5.append(data, ignore_index=True)
#     print(name, monthlymaint, cheapexpensive)

df5 = df5[['name','monthlymaint','cheapexpensive']]
df5
```

```
Facilites where expensive is > $100 per month:
```

Out[100]:

	name	monthlymaint	cheapexpensive
0	Tennis Court 1	200.0	expensive
1	Tennis Court 2	200.0	expensive
2	Badminton Court	50.0	cheap
3	Table Tennis	10.0	cheap
4	Massage Room 1	3000.0	expensive
5	Massage Room 2	3000.0	expensive
6	Squash Court	80.0	cheap
7	Snooker Table	15.0	cheap
8	Pool Table	15.0	cheap

SQL mini project question #6

In [99]:
```
query = 'SELECT firstname, surname, joindate FROM members ORDER BY joindate DE
SC'

cursor.execute(query)

q6result = cursor.fetchall()

data = {}
df6 = pd.DataFrame(data)

print('Members and their Join Date :\n')

# Both print result to screen and append to dataframe
for r in q6result:
    surname = r[1] # convert the tuple to an integer
    firstname = r[0]
    joindate = r[2]
    data = {
        'surname': surname,
        'firstname': firstname,
        'joindate': joindate
    }
    df6 = df6.append(data, ignore_index=True)
#     print(surname,',', firstname, joindate)

df6 = df6[['surname','firstname','joindate']]
df6
```

```
Members and their Join Date :
```

Out[99]:

	surname	firstname	joindate
0	Smith	Darren	2012-09-26 18:08:45
1	Crumpet	Erica	2012-09-22 08:36:38
2	Hunt	John	2012-09-19 11:32:45
3	Tupperware	Hyacinth	2012-09-18 19:32:05
4	Purview	Millicent	2012-09-18 19:04:01
5	Worthington-Smyth	Henry	2012-09-17 12:27:15
6	Farrell	David	2012-09-15 08:22:05
7	Rumney	Henrietta	2012-09-05 08:42:35
8	Jones	Douglas	2012-09-02 18:43:05
9	Sarwin	Ramnaresh	2012-09-01 08:44:42
10	Coplin	Joan	2012-08-29 08:32:41
11	Mackenzie	Anna	2012-08-26 09:32:05
12	Genting	Matthew	2012-08-19 14:55:55
13	Pinker	David	2012-08-16 11:32:47
14	Baker	Timothy	2012-08-15 10:34:25
15	Bader	Florence	2012-08-10 17:52:03
16	Smith	Jack	2012-08-10 16:22:05
17	Farrell	Jemima	2012-08-10 14:28:01
18	Baker	Anne	2012-08-10 14:23:22
19	Jones	David	2012-08-06 16:32:55
20	Owen	Charles	2012-08-03 19:42:37
21	Stibbons	Ponder	2012-07-25 17:09:05
22	Boothe	Tim	2012-07-25 16:02:35
23	Dare	Nancy	2012-07-25 08:59:12
24	Tracy	Burton	2012-07-15 08:52:55
25	Butters	Gerald	2012-07-09 10:44:09
26	Joplette	Janice	2012-07-03 10:25:05
27	Rownam	Tim	2012-07-03 09:32:15
28	Smith	Tracy	2012-07-02 12:08:23
29	Smith	Darren	2012-07-02 12:02:05
30	GUEST	GUEST	2012-07-01 00:00:00

/ *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 LIMIT 0 , 30

/ *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 *FROM* `Facilities` *WHERE membercost < (monthlymaintenance* 0.2) LIMIT 0 , 30

/ *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) LIMIT 0 , 30

/ *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 cheapexpensive FROM `Facilities` LIMIT 0 , 30

/ *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, joindate FROM `Members` ORDER BY joindate DESC LIMIT 0 , 30

/ *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 b.facid, f.name, concat(m.firstname,' ',m.surname) as membername, b.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 b.date = '2012-09-14' and b.cost > 30 order by b.cost desc

/ *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 b.facid, b.starttime, f.name as facility, concat(m.firstname,' ',m.surname) as membername, case when (m.firstname = 'GUEST' and f.guestcost *b.slots > 30) then f.guestcost* b.slots when (m.firstname != 'GUEST' and f.membercost *b.slots > 30) then f.membercost* b.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 having b.starttime like '2012-09-14%' and cost is not null order by cost desc;

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

select b.facid, b.starttime, f.name as facility, concat(m.firstname,' ',m.surname) as membername, (select distinct f.guestcost *b.slots from Facilities where f.guestcost* b.slots > 30 and m.firstname = 'GUEST') + (select distinct f.membercost *b.slots from Facilities where f.membercost* b.slots > 30 and m.firstname != 'GUEST') 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 having b.starttime like '2012-09-14%' and cost is not null order by cost 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!* /

select f.name as facility,
sum(case when m.firstname = 'GUEST' then f.guestcost *b.slots when m.firstname != 'GUEST' then f.membercost* b.slots end) as revenue from bookings as b inner join members as m on m.memid = b.memid inner join facilities as f on f.facid = b.facid group by f.name having revenue > 1000 order by revenue desc;

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

select a.memid as memberID, concat(a.surname,', ',a.firstname) as member, b.memid as recommenderID, concat(b.surname,', ',b.firstname) as recommender from members a, members b where a.recommendedby = b.memid and a.memid != 0 and b.memid != 0 order by a.surname;

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

SELECT b.facid as facilityID, sum(b.slots) as memberusage FROM bookings as b WHERE b.memid != 0 group by b.facid order by b.facid

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

SELECT month(b.starttime) as month, sum(b.slots) as facilityusage FROM bookings as b WHERE b.memid != 0 group by month(b.starttime) order by month(b.starttime)