**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 name AS 'Name of Facilities'

FROM Facilities

WHERE Facilities.membercost <> 0

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

SELECT COUNT(name) AS 'Number of Facilities with no Charge'

FROM Facilities

WHERE Facilities.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 AS 'Facility Name', membercost, monthlymaintenance

FROM Facilities

WHERE 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. \*/

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 label

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 firstname, surname as lastname

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 DISTINCT CONCAT(Members.firstname, ' ', Members.surname) AS MemberName, Facilities.name AS CourtName

FROM Members

JOIN Bookings ON Members.memid = Bookings.memid

JOIN Facilities ON Bookings.facid = Facilities.facid

WHERE Facilities.name LIKE 'Tennis Court%'

ORDER BY MemberName

/\* 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 Facilities.name AS Facility, CONCAT(Members.firstname, ' ', Members.surname) AS MemberName,

CASE

WHEN Bookings.memid = 0 THEN slots \* Facilities.guestcost

ELSE slots \* Facilities.membercost

END AS Cost

FROM Bookings

JOIN Facilities ON Bookings.facid = Facilities.facid

JOIN Members ON Bookings.memid = Members.memid

WHERE Bookings.starttime LIKE '2012-09-14%'

AND ((Bookings.memid = 0 AND slots \* Facilities.guestcost > 30) OR (Bookings.memid != 0 AND slots \* Facilities.membercost > 30))

ORDER BY Cost DESC

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

SELECT Facility, MemberName, Cost

FROM

(

SELECT Facilities.name AS Facility, CONCAT(Members.firstname, ' ', Members.surname) AS MemberName,

CASE

WHEN Bookings.memid = 0 THEN slots \* Facilities.guestcost

ELSE slots \* Facilities.membercost

END AS Cost,

Bookings.starttime AS StartTime

FROM Bookings

JOIN Facilities ON Bookings.facid = Facilities.facid

JOIN Members ON Bookings.memid = Members.memid

) AS Subquery

WHERE StartTime LIKE '2012-09-14%'

AND Cost > 30

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! \*/

Using SQL:

SELECT Facilities.name AS Facility,

SUM(

CASE

WHEN Bookings.memid = 0 THEN Bookings.slots \* Facilities.guestcost

ELSE Bookings.slots \* Facilities.membercost

END

) AS TotalRevenue

FROM Bookings

JOIN Facilities ON Bookings.facid = Facilities.facid

GROUP BY Facility

HAVING TotalRevenue < 1000

ORDER BY TotalRevenue

Using Python:

from sqlalchemy import create\_engine

import pandas as pd

# Import database tables from the local 'sqlite\_db\_pythonsqlite.db' file

engine = create\_engine('sqlite:////Users/mj/Desktop/Springboard Bootcamp/Springboard Projects\_MJ/Unit 08\_Country Club/sqlite\_db\_pythonsqlite.db')

club\_facility = pd.read\_sql\_query('SELECT \* FROM Facilities', engine)

club\_booking = pd.read\_sql\_query('SELECT \* FROM Bookings', engine)

club\_member = pd.read\_sql\_query('SELECT \* FROM Members', engine)

# A list of facilities with a total revenue less than 1000

merged\_df = pd.merge(club\_booking, club\_facility, left\_on='facid', right\_on='facid', how='inner')

merged\_df['total\_revenue'] = merged\_df.apply(lambda row: row['slots'] \* row['guestcost'] if row['memid'] == 0 else row['slots'] \* row['membercost'], axis=1)

revenue\_df = merged\_df.groupby('name')['total\_revenue'].sum().reset\_index()

revenue\_df\_new = revenue\_df[revenue\_df['total\_revenue'] < 1000].copy()

revenue\_df\_new.sort\_values(by='total\_revenue', inplace=True)

revenue\_df\_new

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

Using SQL:

SELECT

CONCAT(m1.surname, ' ',m1.firstname) AS MemberName,

CONCAT(m2.surname, ' ',m2.firstname) AS Recommender

FROM Members m1

JOIN Members m2 ON m1.recommendedby = m2.memid

WHERE m1.recommendedby != ''

ORDER BY MemberName

Using Python:

from sqlalchemy import create\_engine

import pandas as pd

# Import database tables from the local 'sqlite\_db\_pythonsqlite.db' file

engine = create\_engine('sqlite:////Users/mj/Desktop/Springboard Bootcamp/Springboard Projects\_MJ/Unit 08\_Country Club/sqlite\_db\_pythonsqlite.db')

club\_facility = pd.read\_sql\_query('SELECT \* FROM Facilities', engine)

club\_booking = pd.read\_sql\_query('SELECT \* FROM Bookings', engine)

club\_member = pd.read\_sql\_query('SELECT \* FROM Members', engine)

# Produce a report of members and who recommended them

club\_member['recommendedby'].replace('', np.nan, inplace=True) # Missing values treatment

club\_member['recommendedby'].fillna(-1, inplace=True)

club\_member['recommendedby'] = club\_member['recommendedby'].astype('int64')

report\_member = pd.merge(club\_member, club\_member, left\_on='recommendedby', right\_on='memid', suffixes=('\_member', '\_recommender'))

report\_member['MemberName'] = report\_member['surname\_member'] + ' ' + report\_member['firstname\_member']

report\_member['Recommender'] = report\_member['surname\_recommender'] + ' ' + report\_member['firstname\_recommender']

report\_df = report\_member[['MemberName', 'Recommender']].sort\_values('MemberName')

report\_df

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

Using SQL:

SELECT Facilities.name AS Facility, COUNT(\*) AS MemberUsage

FROM Bookings

JOIN Facilities ON Bookings.facid = Facilities.facid

WHERE Bookings.memid != 0

GROUP BY Facilities.name

Using Python:

from sqlalchemy import create\_engine

import pandas as pd

# Import database tables from the local 'sqlite\_db\_pythonsqlite.db' file

engine = create\_engine('sqlite:////Users/mj/Desktop/Springboard Bootcamp/Springboard Projects\_MJ/Unit 08\_Country Club/sqlite\_db\_pythonsqlite.db')

club\_facility = pd.read\_sql\_query('SELECT \* FROM Facilities', engine)

club\_booking = pd.read\_sql\_query('SELECT \* FROM Bookings', engine)

club\_member = pd.read\_sql\_query('SELECT \* FROM Members', engine)

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

usage\_member = pd.merge(club\_booking, club\_facility, left\_on='facid', right\_on='facid')

usage\_member = usage\_member[usage\_member['memid'] != 0]

grouped\_usage\_member = usage\_member.groupby('name').size().reset\_index(name='MemberUsage')

grouped\_usage\_member.rename(columns={'name': 'Facility'}, inplace=True)

grouped\_usage\_member

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

Using SQL:

SELECT

Facilities.name AS Facility,

MONTH(Bookings.starttime) AS Month,

COUNT(\*) AS member\_usage

FROM Bookings

JOIN Facilities ON Bookings.facid = Facilities.facid

WHERE Bookings.memid != 0

GROUP BY Facilities.name, Month

ORDER BY Facility, Month

Using Python:

from sqlalchemy import create\_engine

import pandas as pd

# Import database tables from the local 'sqlite\_db\_pythonsqlite.db' file

engine = create\_engine('sqlite:////Users/mj/Desktop/Springboard Bootcamp/Springboard Projects\_MJ/Unit 08\_Country Club/sqlite\_db\_pythonsqlite.db')

club\_facility = pd.read\_sql\_query('SELECT \* FROM Facilities', engine)

club\_booking = pd.read\_sql\_query('SELECT \* FROM Bookings', engine)

club\_member = pd.read\_sql\_query('SELECT \* FROM Members', engine)

# Find the facilities usage by month, but not guests

usage\_month = pd.merge(club\_booking, club\_facility, left\_on='facid', right\_on='facid')

usage\_month = usage\_month[usage\_month['memid'] != 0]

usage\_month['Month'] = pd.to\_datetime(usage\_month['starttime']).dt.month

grouped\_usage\_month = usage\_month.groupby(['name', 'Month']).size().reset\_index(name='member\_usage')

grouped\_usage\_month.rename(columns={'name': 'Facility'}, inplace=True)

grouped\_usage\_month.sort\_values(by=['Facility', 'Month']).reset\_index(drop=True, inplace=True)

grouped\_usage\_month