**SKIP CODE SEGMENT FOR PARAGRAPHED EXPLANATION**

import mysql.connector **import of sql python library**

**Creating of MySQL Connection**

cnx = mysql.connector.connect(user='root', host = '127.0.0.1', password = 'VFR4cde3'

, database = 'test')

**Declaration of SQL cursor. We cannot execute SQL statements through connection. We need to do it through cursors.**

cur = cnx.cursor()

**Preparation of SQL statements. Note: See how %(start)s named placeholders are placed.**

sqlStep1 = "SELECT S\_ID, LOCATION, UNIX\_TIMESTAMP('2015-08-20 %(start)d:00:00') FROM starhub WHERE UNIX\_TIME BETWEEN UNIX\_TIMESTAMP('2015-08-20 %(start)d:00:00') AND UNIX\_TIMESTAMP('2015-08-20 %(start)d:59:59') GROUP BY S\_ID;"

sqlStep2 = "SELECT LOCATION, UNIX\_TIMESTAMP('2015-08-20 %(endtime)d:00:00') FROM starhub WHERE UNIX\_TIME BETWEEN UNIX\_TIMESTAMP('2015-08-20 %(endtime)d:00:00') AND UNIX\_TIMESTAMP('2015-08-20 %(endtime)d:59:59') AND S\_ID = '%(s\_id)s' AND LOCATION != '%(location)s';"

sqlInsertMovement = "INSERT INTO `test`.`movement`(`movement\_id`,`UNIX\_START`,`UNIX\_END`,`START\_LOCATION`,`DEST\_LOCATION`,`COUNT`) VALUES(%(movement\_id)d,%(unixstart)d,%(unixend)d,'%(start)s','%(dest)s',1);"

sqlUpdateCount = "UPDATE `test`.`movement` SET `COUNT` = `COUNT` + 1 WHERE `UNIX\_START` = %(unixstart)d AND `UNIX\_END` = %(unixend)d AND `START\_LOCATION` = '%(start)s' AND `DEST\_LOCATION` = '%(dest)s';"

sqlSelectMovement = "SELECT COUNT(\*) AS c FROM movement WHERE UNIX\_START = %(unixstart)d AND UNIX\_END = %(unixend)d AND `START\_LOCATION` = '%(start)s' AND `DEST\_LOCATION` = '%(dest)s';"

**The (starting time) hour which we are generating movement for. Ie 8, when we are doing 8 to 9am.**

runninghourInt = 0

**Function to prepare SQL statement (Placing value into the string placeholders)**

def prep(sql, parameter):

p = sql%parameter

return p

**To escape single quotes on location names (People’s Park) which will cause SQL syntax error.**

def escape(string):

return string.replace("'","\\'")

totalMovement = 0

numMovement = 0

**Start of logic. See paragraphed explanation.**

while(runninghourInt <= 22):

runninghourEndInt=runninghourInt + 1

print("Started Step 1: Selecting " + str(runninghourInt))

dataSql1={

'start':runninghourInt

}

cur.execute(sqlStep1%dataSql1)

step1data = cur.fetchall()

print("data obtained, adding movements")

numSID = 0

numUpdate = 0

for(S\_ID, LOCATION, UNIX\_TIME) in step1data:

numSID = numSID + 1

unixstart = int(UNIX\_TIME)

startlocation = LOCATION

for i in range(runninghourEndInt, 24):

dataSql2 = {

'endtime': i,

's\_id': S\_ID,

'location': escape(startlocation),

}

cur.execute(prep(sqlStep2,dataSql2))

step2data = cur.fetchall()

step2location = ""

unixend = 0

thereOrNot = False

for(LOCATION, UNIX\_TIME) in step2data:

thereOrNot = True

unixend = int(UNIX\_TIME)

step2location = LOCATION

break

if(thereOrNot == False):

continue

dataSelectMovement = {

'movement\_id': numMovement,

'unixstart': unixstart,

'unixend': unixend,

'start': escape(startlocation),

'dest': escape(step2location)

}

cur.execute(prep(sqlSelectMovement,dataSelectMovement))

step3data = cur.fetchall()

alrHaveMovement = False

for(c) in step3data:

if(c[0] > 0):

alrHaveMovement = True

break

if(alrHaveMovement):

cur.execute(prep(sqlUpdateCount, dataSelectMovement))

cnx.commit()

numUpdate = numUpdate + 1

else:

cur.execute(prep(sqlInsertMovement, dataSelectMovement))

cnx.commit()

numMovement = numMovement + 1

runninghourInt = runninghourInt + 1

cur.close()

cnx.close()

print("COMPLETE. Total movement is " + str(totalMovement))

**Paragraph explanation**

In summary, the code has three steps:

1. Obtain list of S\_IDs and their location from **starhub table** where time falls in between current running hour and 59 min 59 sec of the hour.
   1. If running hour is 8, we will select all unique S\_IDs and their locations from 0800 to 0859:59.
2. For each S\_IDs returned from step 1, we now have to go through **starhub table** and look for them hourly from running hour + 1 to 23:59:59.
   1. Why we do hour groupings, refer to Database Dictionary -> About raw data (Limit amount of usable data pairs)..
   2. If running hour is 8, we will look for each S\_ID for their presence in 0900-0959:59 group, then again in 1000-1059:59, until 2300-2359:59.
   3. If each S\_ID is found once in each hours groups, we will grab the new location he/she is in. This is recorded as an instance pattern of movement.
      1. point a and point b, occurring at time a and time b.
3. If the exact movement pattern instance is already recorded, we will not add another row. Instead, we increment the count of instance, under column ‘count’.
   1. This can possibly translate to multiple starhub users travelling together.
      1. If these two do not know each other, may be can introduce a function to let them know each other.