In [1]:
```
import sqlite3 as sql
```

In [5]:
```
db_path = r'C:\Users\Icy\Documents\School-GMU\SocialNetwork.db'
```

In [6]:
```
conn = sql.connect(db_path)
```

In [7]:
```
print( type(conn) )
```

```
<class 'sqlite3.Connection'>
```

In [92]:
```
create_table_query = 'CREATE TABLE event ( ' #here I am creating a table which is being references by another ta
create_table_query += ' e_ID int, '
create_table_query += ' u_ID int NOT NULL, '
create_table_query += ' topic varchar(15), '
create_table_query += ' day numeric(4,0), '
create_table_query += ' month numeric(4,0), '
create_table_query += ' primary key (e_ID) '
create_table_query += ');'

print(create_table_query) #here I am printing what I am trying to do with this query.
```

```
CREATE TABLE event (  e_ID int,  u_ID int NOT NULL,  topic varchar(15),  day numeric(4,0),  month numeric(4,
0),  primary key (e_ID) );
```

In [93]:
```
cursor = conn.execute( create_table_query ) #the change is only made after I use this. Conn stands for connect
```

In [94]:
```
conn.commit() #here I am commiting the changes to the disk. It's not neeeded to do this everysingle time
```

In [95]:
```
create_table_query = 'CREATE TABLE participates ( ' #here Im creating a table which references 2 different table
create_table_query += ' u_ID int NOT NULL, '
create_table_query += ' e_ID int NOT NULL, '
create_table_query += ' primary key (u_ID, e_ID), '
create_table_query += ' foreign key (u_ID) references user ON DELETE NO ACTION, '
create_table_query += ' foreign key (e_ID) references event ON DELETE CASCADE '
create_table_query += ');'

print(create_table_query)
```

```
CREATE TABLE participates (  u_ID int NOT NULL,  e_ID int NOT NULL,  primary key (u_ID, e_ID),  foreign key (u
_ID) references user ON DELETE NO ACTION,  foreign key (e_ID) references event ON DELETE CASCADE );
```

In [96]:
```
cursor = conn.execute( create_table_query )
```

...

In [97]:
```
conn.commit()
```

In [116]:
```
create_table_query = 'CREATE TABLE user ( '
create_table_query += ' u_ID int, '
create_table_query += ' follower_ID int, '
create_table_query += ' e_ID int, '
create_table_query += ' username varchar(15) NOT NULL, '
create_table_query += ' birthyear int, '
create_table_query += ' primary key (u_ID) '
create_table_query += ');'

print(create_table_query)
```

```
CREATE TABLE user (  u_ID int,  follower_ID int,  e_ID int,  username varchar(15) NOT NULL,  birthyear int,  p
rimary key (u_ID) );
```

In [ ]:
```
#the primary key from the schema "follows" is added to this because both tables share a many-to-many relationshi
#similarly, the primary key from "participates" is also added to this table
```

In [117]:
```
cursor = conn.execute( create_table_query )
```

In [118]:
```
create_table_query = 'CREATE TABLE follows ( '
create_table_query += ' u_ID int NOT NULL, '
create_table_query += ' follower_ID int NOT NULL, '
create_table_query += ' primary key (u_ID, follower_ID), '
create_table_query += ' foreign key (u_ID) references user ON DELETE NO ACTION, '
create_table_query += ' foreign key (follower_ID) references user ON DELETE CASCADE '
create_table_query += ');'

print(create_table_query)
```

```
CREATE TABLE follows (  u_ID int NOT NULL,  follower_ID int NOT NULL,  primary key (u_ID, follower_ID),  forei
gn key (u_ID) references user ON DELETE NO ACTION,  foreign key (follower_ID) references user ON DELETE CASCAD
E );
```

In [107]:
```
cursor = conn.execute( create_table_query )
```

...

In [86]:
```
conn.commit()
```

In [87]:
```
query = "SELECT * FROM follows;"
df = pd.read_sql_query(query, conn) #here I am just visualizing how my table looks and if they worked fine.
```

In [88]:
```
df
```

Out[88]:

u_ID	follower_ID

In [89]:
```
query = "SELECT * FROM participates;"
df = pd.read_sql_query(query, conn)
```

In [90]:
```
df #The tables are empty because I haven't added anything yet
```

Out[90]:

u_ID	e_ID

In [119]:
```
query = "SELECT * FROM event;"
df = pd.read_sql_query(query, conn)
```

In [120]:
```
df
```

Out[120]:

e_ID	u_ID	topic	day	month

In [121]:
```
query = "SELECT * FROM user;"
df = pd.read_sql_query(query, conn)
```

In [122]:
```
df
```

Out[122]:

u_ID	follower_ID	e_ID	username	birthyear

In [ ]:
```
# Below I am starting to add tuples into my table.
```

In [153]:
```
insert_data_query = "INSERT INTO event VALUES ("
insert_data_query += "16, "
insert_data_query += "16, "
insert_data_query += "'Birthdays', "
insert_data_query += "15,"
insert_data_query += "06"
insert_data_query += ");"




print( insert_data_query )
```

```
INSERT INTO event VALUES (16, 16, 'Birthdays', 15,06);
```

In [154]:
```
cursor = conn.execute( insert_data_query ) #the changes are not execute until I use this
```

...

In [275]:
```
insert_data_query = "INSERT INTO participates VALUES ("
insert_data_query += "16, "
insert_data_query += "16 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO participates VALUES (16, 16 );
```

In [276]:
```
cursor = conn.execute( insert_data_query )
```

In [157]:
```
insert_data_query = "INSERT INTO user VALUES ("
insert_data_query += "16, "
insert_data_query += "16, "
insert_data_query += "16, "
insert_data_query += "'tammybee', "
insert_data_query += "1988"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO user VALUES (16, 16, 16, 'tammybee', 1988);
```

In [158]:
```
cursor = conn.execute( insert_data_query )
```

In [159]:
```
insert_data_query = "INSERT INTO follows VALUES ("
insert_data_query += "16, "
insert_data_query += "160 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO follows VALUES (16, 160 );
```

In [160]:
```
cursor = conn.execute( insert_data_query )
```

...

In [427]:
```
insert_data_query = "INSERT INTO event VALUES ("
insert_data_query += "18, "
insert_data_query += "18, "
insert_data_query += "'Beer', "
insert_data_query += "11, "
insert_data_query += "02 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO event VALUES (18, 18, 'Birthdays', 11, 02 );
```

In [428]:
```
cursor = conn.execute( insert_data_query )
```

...

In [300]:
```
insert_data_query = "INSERT INTO event VALUES ("
insert_data_query += "20, "
insert_data_query += "20, "
insert_data_query += "'Wine', "
insert_data_query += "08, "
insert_data_query += "06"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO event VALUES (20, 20, 'Wine', 08, 06);
```

In [301]:
```
cursor = conn.execute( insert_data_query )
```

...

In [175]:
```
insert_data_query = "INSERT INTO event VALUES ("
insert_data_query += "22, "
insert_data_query += "22, "
insert_data_query += "'Parties', "
insert_data_query += "10, "
insert_data_query += "12"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO event VALUES (22, 22, 'Parties', 10, 12);
```

In [176]:
```
cursor = conn.execute( insert_data_query )
```

In [177]:
```
insert_data_query = "INSERT INTO event VALUES ("
insert_data_query += "24, "
insert_data_query += "24, "
insert_data_query += "'Houses', "
insert_data_query += "14, "
insert_data_query += "17"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO event VALUES (24, 24, 'Houses', 14, 17);
```

In [178]:
```
cursor = conn.execute( insert_data_query )
```

In [302]:
```
query = "SELECT * FROM event;"
df = pd.read_sql_query(query, conn) #after inserting 5 new tuples into the table, checking to see everything in
```

In [303]:
```
df #df stands for data frame. Here the results are shown more like a table
```

Out[303]:

	e_ID	u_ID	topic	day	month
0	16	16	Birthdays	15	6
1	20	20	Wine	8	6
2	18	18	Beer	11	2
3	22	22	Parties	10	12
4	24	24	Houses	14	17

In [277]:
```
insert_data_query = "INSERT INTO participates VALUES ("
insert_data_query += "20, "
insert_data_query += "20 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO participates VALUES (20, 20 );
```

In [278]:
```
cursor = conn.execute( insert_data_query )
```

In [279]:
```
insert_data_query = "INSERT INTO participates VALUES ("
insert_data_query += "18, "
insert_data_query += "18 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO participates VALUES (18, 18 );
```

In [280]:
```
cursor = conn.execute( insert_data_query )
```

In [281]:
```
insert_data_query = "INSERT INTO participates VALUES ("
insert_data_query += "22, "
insert_data_query += "22 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO participates VALUES (22, 22 );
```

In [282]:
```
cursor = conn.execute( insert_data_query )
```

In [283]:
```
insert_data_query = "INSERT INTO participates VALUES ("
insert_data_query += "24, "
insert_data_query += "24 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO participates VALUES (24, 24 );
```

In [284]:
```
cursor = conn.execute( insert_data_query )
```

In [285]:
```
query = "SELECT * FROM participates;"
df = pd.read_sql_query(query, conn) #5 new tuples inserted into the table. Checking to see if everything worked
```

In [286]:
```
df
```

Out[286]:

	u_ID	e_ID
0	16	16
1	20	20
2	18	18
3	22	22
4	24	24

In [193]:
```
insert_data_query = "INSERT INTO user VALUES ("
insert_data_query += "18, "
insert_data_query += "18, "
insert_data_query += "18, "
insert_data_query += "'crex', "
insert_data_query += "1988"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO user VALUES (18, 18, 18, 'crex', 1988);
```

In [194]:
```
cursor = conn.execute( insert_data_query ) #This is used so the query can be executed
```

In [197]:
```
insert_data_query = "INSERT INTO user VALUES ("
insert_data_query += "20, "
insert_data_query += "20, "
insert_data_query += "20, "
insert_data_query += "'ncarmona', "
insert_data_query += "1957"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO user VALUES (20, 20, 20, 'ncarmona', 1957);
```

In [198]:
```
cursor = conn.execute( insert_data_query )
```

In [201]:
```
insert_data_query = "INSERT INTO user VALUES ("
insert_data_query += "22, "
insert_data_query += "22, "
insert_data_query += "22, "
insert_data_query += "'Jbarbosa', "
insert_data_query += "1961"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO user VALUES (22, 22, 22, 'Jbarbosa', 1961);
```

In [202]:
```
cursor = conn.execute( insert_data_query )
```

In [203]:
```
insert_data_query = "INSERT INTO user VALUES ("
insert_data_query += "24, "
insert_data_query += "24, "
insert_data_query += "24, "
insert_data_query += "'Jrex', "
insert_data_query += "1956"
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO user VALUES (24, 24, 24, 'Jrex', 1956);
```

In [204]:
```
cursor = conn.execute( insert_data_query )
```

In [205]:
```
query = "SELECT * FROM user;"
df = pd.read_sql_query(query, conn)
```

In [206]:
```
df #df corresponds to the location on the disk. I could have used df1 or df2 ...
```

Out[206]:

	u_ID	follower_ID	e_ID	username	birthyear
0	16	16	16	tammybee	1988
1	18	18	18	crex	1988
2	20	20	20	ncarmona	1957
3	22	22	22	Jbarbosa	1961
4	24	24	24	Jrex	1956

In [207]:
```
insert_data_query = "INSERT INTO follows VALUES ("
insert_data_query += "18, "
insert_data_query += "180 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO follows VALUES (18, 180 );
```

In [208]:
```
cursor = conn.execute( insert_data_query )
```

In [209]:
```
insert_data_query = "INSERT INTO follows VALUES ("
insert_data_query += "20, "
insert_data_query += "200 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO follows VALUES (20, 200 );
```

In [210]:
```
cursor = conn.execute( insert_data_query )
```

In [211]:
```
insert_data_query = "INSERT INTO follows VALUES ("
insert_data_query += "22, "
insert_data_query += "220 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO follows VALUES (22, 220 );
```

In [212]:
```
cursor = conn.execute( insert_data_query )
```

In [213]:
```
insert_data_query = "INSERT INTO follows VALUES ("
insert_data_query += "24, "
insert_data_query += "240 "
insert_data_query += ");"

print( insert_data_query )
```

```
INSERT INTO follows VALUES (24, 240 );
```

In [214]:
```
cursor = conn.execute( insert_data_query )
```

In [215]:
```
query = "SELECT * FROM follows;"
df = pd.read_sql_query(query, conn) #checking to see everything on the table
```

In [216]:
```
df
```

Out[216]:

	u_ID	follower_ID
0	16	16
1	16	160
2	18	180
3	20	200
4	22	220
5	24	240

Insert 15 random pairs of (user id, event id) in the Table Participates.

In [219]:
```
users_ID = [34, 35, 46, 52, 72, 73, 97, 76, 34, 65, 43, 41, 38, 98, 56]
```

In [292]:
```
import random

#here I am adding random values to the table participates. This is known as a for loop where data is added in a

for i in range(15):


   u_id = str(i)

   random_user= random.randint(16, 24)
   u_ID = str(random_user)

   random_e_ID = random.randint(16,24)
   e_ID = str (random_e_ID)

   insert_data_query = "INSERT INTO participates VALUES ( "
   insert_data_query += u_ID +", "
   insert_data_query += e_ID
   insert_data_query += ");"


   try: cursor = conn.execute(insert_data_query)
   except: print ("Already exists",u_ID,e_ID)
```

```
Already exists 18 18
Already exists 20 21
Already exists 18 18
```

In [272]:
```
cursor = conn.execute( 'DELETE FROM participates;' ) #if I execute this, everything from the table is deleted bu
```

In [294]:
```
query = "SELECT * FROM participates;"
df = pd.read_sql_query(query, conn) #here I am checking if everything was added successfully
```

In [295]:
```
df
```

Out[295]:

	u_ID	e_ID
0	16	16
1	20	20
2	18	18
3	22	22
4	24	24
5	20	16
6	23	17
7	19	20
8	16	17
9	18	20
10	21	17
11	17	16
12	20	21
13	18	23
14	21	22
15	23	19
16	21	16

Submit a query to your database that returns the username and birthyear of a user, as well as the event topic, day and month, for all the users that participated in events. (If a user participated in multiple events, she will have multiple records in the result with her user information, but different event information.)

In [306]:
```
query = 'SELECT username, birthyear, topic, day, month '
query +='FROM user NATURAL JOIN participates NATURAL JOIN event;'
result = conn.execute( query )

print( result.fetchall() ) #here I am just printing the query but not commiting it to the disk
```

```
[('tammybee', 1988, 'Birthdays', 15, 6), ('crex', 1988, 'Beer', 11, 2), ('ncarmona', 1957, 'Wine', 8, 6), ('Jb
arbosa', 1961, 'Parties', 10, 12), ('Jrex', 1956, 'Houses', 14, 17)]
```

In [307]:
```
# adding a new colunm to the table user called lat
alter_query = 'ALTER TABLE user ADD COLUMN lat REAL;'
cursor = conn.execute( alter_query )
```

In [308]:
```
# adding a new colunm to the table user called lon
alter_query = 'ALTER TABLE user ADD COLUMN lon REAL;'
cursor = conn.execute( alter_query )
```

In [309]:
```
df = pd.read_sql_query('SELECT * FROM user;', conn)
df.head()

#here the new colunms were created but since there are no values, it shows as none for each tuple.
```

Out[309]:

	u_ID	follower_ID	e_ID	username	birthyear	lat	lon
0	16	16	16	tammybee	1988	None	None
1	18	18	18	crex	1988	None	None
2	20	20	20	ncarmona	1957	None	None
3	22	22	22	Jbarbosa	1961	None	None
4	24	24	24	Jrex	1956	None	None

In [310]:
```
# adding a new colunm to the table event called elat
alter_query = 'ALTER TABLE event ADD COLUMN elat REAL;'
cursor = conn.execute( alter_query )
```

In [311]:
```
# adding a new colunm to the table user called lon
alter_query = 'ALTER TABLE event ADD COLUMN elon REAL;'
cursor = conn.execute( alter_query )
```

In [312]:
```
df = pd.read_sql_query('SELECT * FROM event;', conn)
df.head()

#here the new colunms were created as well but since there are no values, it shows as none for each tuple.
```

Out[312]:

	e_ID	u_ID	topic	day	month	elat	elon
0	16	16	Birthdays	15	6	None	None
1	20	20	Wine	8	6	None	None
2	18	18	Beer	11	2	None	None
3	22	22	Parties	10	12	None	None
4	24	24	Houses	14	17	None	None

Insert random values for the latitude and longitude of each user.

In [459]:
```
# There are 5 people in the "user" table
for i in range(6):
  lat = random.uniform(-115, -73)
  lon = random.uniform(32, 40)

  # The numbers are rounded to a precision of 3 decimal digits

  lat = round(lat, 3)
  lon = round(lon, 3)

  update_query = 'UPDATE user SET '
  update_query += 'lat = ' + str(lat) + ', '
  update_query += 'lon = ' + str(lon) + ' '
  update_query += 'WHERE u_id = ' + str(16 + i ) + ';' #this allows diff random values to be added to each tuple



 #This will update random values for latitudes and longitudes in the table.

  print(update_query)

  cursor = conn.execute( update_query )
```

```
UPDATE user SET lat = -106.191, lon = 38.125 WHERE u_id = 16;
UPDATE user SET lat = -86.879, lon = 32.072 WHERE u_id = 17;
UPDATE user SET lat = -81.655, lon = 34.735 WHERE u_id = 18;
UPDATE user SET lat = -113.351, lon = 33.319 WHERE u_id = 19;
UPDATE user SET lat = -81.484, lon = 39.377 WHERE u_id = 20;
UPDATE user SET lat = -77.986, lon = 38.322 WHERE u_id = 21;
```

In [460]:
```
df = pd.read_sql_query('SELECT * FROM user;', conn)
df.head()

#now I can confirm that each tuple was updated and it now has a value for the new lat and lon columns.
```

Out[460]:

	u_ID	follower_ID	e_ID	username	birthyear	lat	lon
0	16	16	16	tammybee	1988	-106.191	38.125
1	18	18	18	crex	1988	-81.655	34.735
2	20	20	20	ncarmona	1957	-81.484	39.377
3	22	22	22	Jbarbosa	1961	-110.083	33.493
4	24	24	24	Jrex	1956	-110.083	33.493

Insert random values for the latitude and longitude of each event.

In [461]:
```
# There are 5 people in the use table
for i in range(6):
 elat = random.uniform(-117, -75)
 elon = random.uniform(32, 40)

 # The numbers are rounded to a precision of 3 decimal digits
 elat = round(elat, 3)
 elon = round(elon, 3)

 update_query = 'UPDATE event SET '
 update_query += 'elat = ' + str(elat) + ', '
 update_query += 'elon = ' + str(elon) + ' '

 update_query += 'WHERE e_id = ' + str(16 + i) + ';'

 print(update_query)
 cursor = conn.execute( update_query )

#the same thing is being done here and random values of lat and lon are being added to the "event" table
```

```
UPDATE event SET elat = -110.472, elon = 37.143 WHERE e_id = 16;
UPDATE event SET elat = -93.846, elon = 37.68 WHERE e_id = 17;
UPDATE event SET elat = -98.97, elon = 33.534 WHERE e_id = 18;
UPDATE event SET elat = -100.421, elon = 35.056 WHERE e_id = 19;
UPDATE event SET elat = -76.095, elon = 34.374 WHERE e_id = 20;
UPDATE event SET elat = -88.843, elon = 39.505 WHERE e_id = 21;
```

In [462]:
```
df = pd.read_sql_query('SELECT * FROM event;', conn) #select * shows me everything in the table
df.head()
```

Out[462]:

	e_ID	u_ID	topic	day	month	elat	elon
0	16	16	Birthdays	15	6	-110.472	37.143
1	20	20	Wine	8	6	-76.095	34.374
2	18	18	Beer	11	2	-98.970	33.534
3	22	22	Parties	10	12	-92.081	34.645
4	24	24	Houses	14	17	-92.081	34.645

In [463]:
```
conn.commit()  #here I am commiting all changes to the disk
```

In [464]:
```
query = ' SELECT username, lat, lon, topic, elat, elon '
query +=' FROM user NATURAL JOIN participates NATURAL JOIN event; '
result = conn.execute( query )

print( result.fetchall() )
```

```
[('tammybee', -106.191, 38.125, 'Birthdays', -110.472, 37.143), ('crex', -81.655, 34.735, 'Beer', -98.97, 33.5
34), ('ncarmona', -81.484, 39.377, 'Wine', -76.095, 34.374), ('Jbarbosa', -110.083, 33.493, 'Parties', -92.08
1, 34.645), ('Jrex', -110.083, 33.493, 'Houses', -92.081, 34.645)]
```

Write a join query that finds all the user named and locations of users, who attended an event, and the id and location of the event. Maintain this result in a dataframe df1. Show df1.

In [465]:
```
df1 = pd.read_sql_query('SELECT username, lat, lon, topic, elat, elon FROM user NATURAL JOIN participates NATURA
df1 #natural join query combines multiple table that are related to each other
```

Out[465]:

	username	lat	lon	topic	elat	elon
0	tammybee	-106.191	38.125	Birthdays	-110.472	37.143
1	crex	-81.655	34.735	Beer	-98.970	33.534
2	ncarmona	-81.484	39.377	Wine	-76.095	34.374
3	Jbarbosa	-110.083	33.493	Parties	-92.081	34.645
4	Jrex	-110.083	33.493	Houses	-92.081	34.645

Write a query that counts the number of users who participate in each event and shows the event id and user count. Keep the result in another dataframe df2. Show df2.

In [466]:
```
df2 = pd.read_sql_query('SELECT count (u_ID), e_ID , topic FROM user NATURAL JOIN participates NATURAL JOIN even
df2
```

Out[466]:

	count (u_ID)	e_ID	topic
0	1	16	Birthdays
1	1	18	Beer
2	1	20	Wine
3	1	22	Parties
4	1	24	Houses

In [386]:
```
query = ' SELECT count (u_ID), e_ID , topic FROM user NATURAL JOIN participates NATURAL JOIN event GROUP BY e_ID
df2 = pd.read_sql_query(query, conn)
df2.head()
```

Out[386]:

	count (u_ID)	e_ID	topic
0	1	16	Birthdays
1	1	18	Beer
2	1	20	Wine
3	1	22	Parties
4	1	24	Houses

• Calculate the harvesine distance between each user to the event(s) that they participate in, using the coordinates from your query result in df1. Insert those distance values in a new ‘dist’ column in df1.

In [467]:
```
from math import sin, cos, sqrt, atan2, radians
def harvesine_dist(lat1, lon1, lat2, lon2):
 R = 6373.0 # approximate radius of earth in km
 dlon = lon2 - lon1
 dlat = lat2 - lat1
 a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
 c = 2 * atan2(sqrt(a), sqrt(1 - a))
 distance = R * c
 return distance
#in order to get the havesine, the math packet needs to be imported
```

In [468]:
```
import numpy as np
eLat = np.array(df['elat'])
eLon = np.array(df['elon'])
Lat = np.array(df['lat'])
Lon = np.array(df['lon'])
```

...

In [349]:
```
elat
```

Out[349]:
```
-86.679
```

In [355]:
```
query = 'SELECT COUNT(*) FROM participates;'
result = conn.execute( query )
```

In [388]:
```
df1 = pd.read_sql_query(query, conn)
df1.head()
```

...

In [360]:
```
dist_list = []
for i in range(len(eLat)):
 dist = harvesine_dist(eLat[i], eLon[i], Lat[i], Lon[i])
 dist_list.append(dist)

print( dist_list )
#this query gives the harvesine  distance which is the angular distance between two point on earth
```

```
[12311.91614196848, 12311.91614196848, 12311.91614196848, 12311.91614196848, 12311.91614196848]
```

In [469]:
```
query = 'SELECT * FROM event natural join user natural join participates; '
df1 = pd.read_sql_query(query, conn)
df1.head()
```

Out[469]:

	e_ID	u_ID	topic	day	month	elat	elon	follower_ID	username	birthyear	lat	lon
0	16	16	Birthdays	15	6	-110.472	37.143	16	tammybee	1988	-106.191	38.125
1	20	20	Wine	8	6	-76.095	34.374	20	ncarmona	1957	-81.484	39.377
2	18	18	Beer	11	2	-98.970	33.534	18	crex	1988	-81.655	34.735
3	22	22	Parties	10	12	-92.081	34.645	22	Jbarbosa	1961	-110.083	33.493
4	24	24	Houses	14	17	-92.081	34.645	24	Jrex	1956	-110.083	33.493

In [470]:
```
df1['distance'] = dist_list
df1.head()
```

Out[470]:

	e_ID	u_ID	topic	day	month	elat	elon	follower_ID	username	birthyear	lat	lon	distance
0	16	16	Birthdays	15	6	-110.472	37.143	16	tammybee	1988	-106.191	38.125	12311.916142
1	20	20	Wine	8	6	-76.095	34.374	20	ncarmona	1957	-81.484	39.377	12311.916142
2	18	18	Beer	11	2	-98.970	33.534	18	crex	1988	-81.655	34.735	12311.916142
3	22	22	Parties	10	12	-92.081	34.645	22	Jbarbosa	1961	-110.083	33.493	12311.916142
4	24	24	Houses	14	17	-92.081	34.645	24	Jrex	1956	-110.083	33.493	12311.916142

In [ ]:
```
#The above distance corresponds to the distance between the each user and the corresponding event they participa
#The above information about the distance is store in the main memory and not in the disk. So it;s a short-term
#Another query needs to be submitted if the this information is to be trasnfered to the disk
#Since the info is in the memory, this information can be lost if the power is lost or the computer is restartea
```

Plot all the events on a map, as red circles. The size of the circle is proportional to the number of users who participate in the event. Show the map.

In [367]:
```
import folium
from folium import plugins
#this is the packet that allow maps to be added to python
```

In [481]:
```
query = 'SELECT * '
query += 'FROM user NATURAL JOIN participates NATURAL JOIN event '
df = pd.read_sql_query(query, conn)
df1.head()
```

Out[481]:

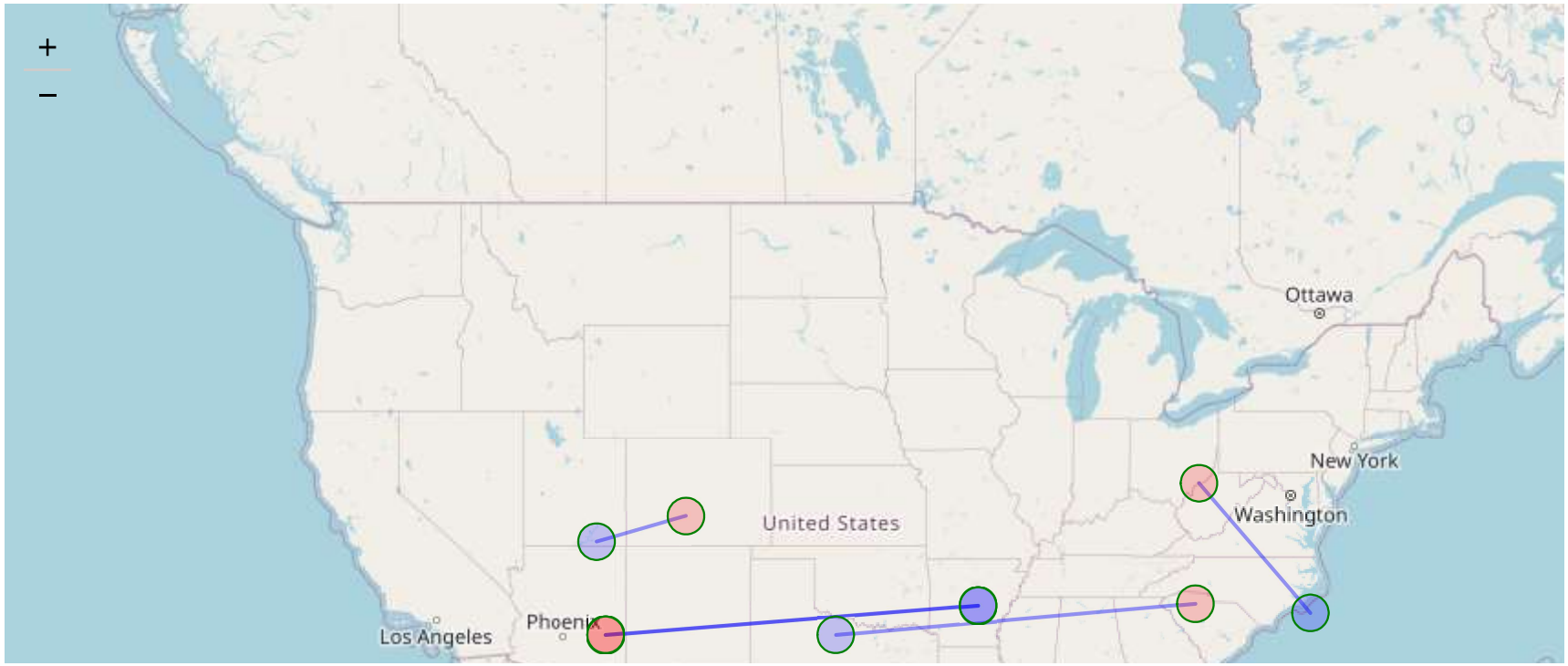
	e_ID	u_ID	topic	day	month	elat	elon	follower_ID	username	birthyear	lat	lon	distance
0	16	16	Birthdays	15	6	-110.472	37.143	16	tammybee	1988	-106.191	38.125	12311.916142
1	20	20	Wine	8	6	-76.095	34.374	20	ncarmona	1957	-81.484	39.377	12311.916142
2	18	18	Beer	11	2	-98.970	33.534	18	crex	1988	-81.655	34.735	12311.916142
3	22	22	Parties	10	12	-92.081	34.645	22	Jbarbosa	1961	-110.083	33.493	12311.916142
4	24	24	Houses	14	17	-92.081	34.645	24	Jrex	1956	-110.083	33.493	12311.916142

In [499]:
```
folium_map = folium.Map(location=[37.0902, -95.7129], # USA coordinates
 zoom_start=6,
tiles="openstreetmap")
folium_map
```

Out[499]:

In [501]:
```
for index, row in df1.iterrows ():

   #size

   #radius= (row['topic'])/10
   zoom_start=6,

   #color
   color1= 'red'
   color2= 'blue'

   # add marker to the map
   folium.CircleMarker(location= (row['lon'], row['lat']), weight=1, radius= 10,
             color=color, fill=True, fill_color= color1).add_to (folium_map)

   folium.CircleMarker(location= (row['elon'], row['elat']), weight=1, radius= 10,
             color=color, fill=True, fill_color= color2).add_to (folium_map)

folium_map
```

Out[501]:



In [500]:
```
for index, row in df1.iterrows():
  p1 = [row['lon'], row['lat']]
  p2 = [row['elon'], row['elat']]
  zoom_start=8,
  folium.PolyLine(locations=[p1, p2], opacity=0.4, weight = 2, color='blue').add_to (folium_map)
folium_map
```

...

In [ ]:
```
```