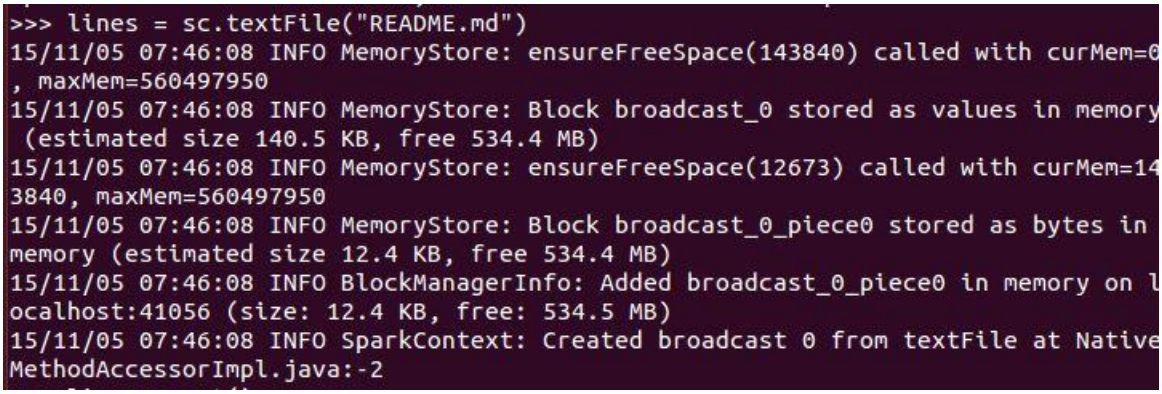
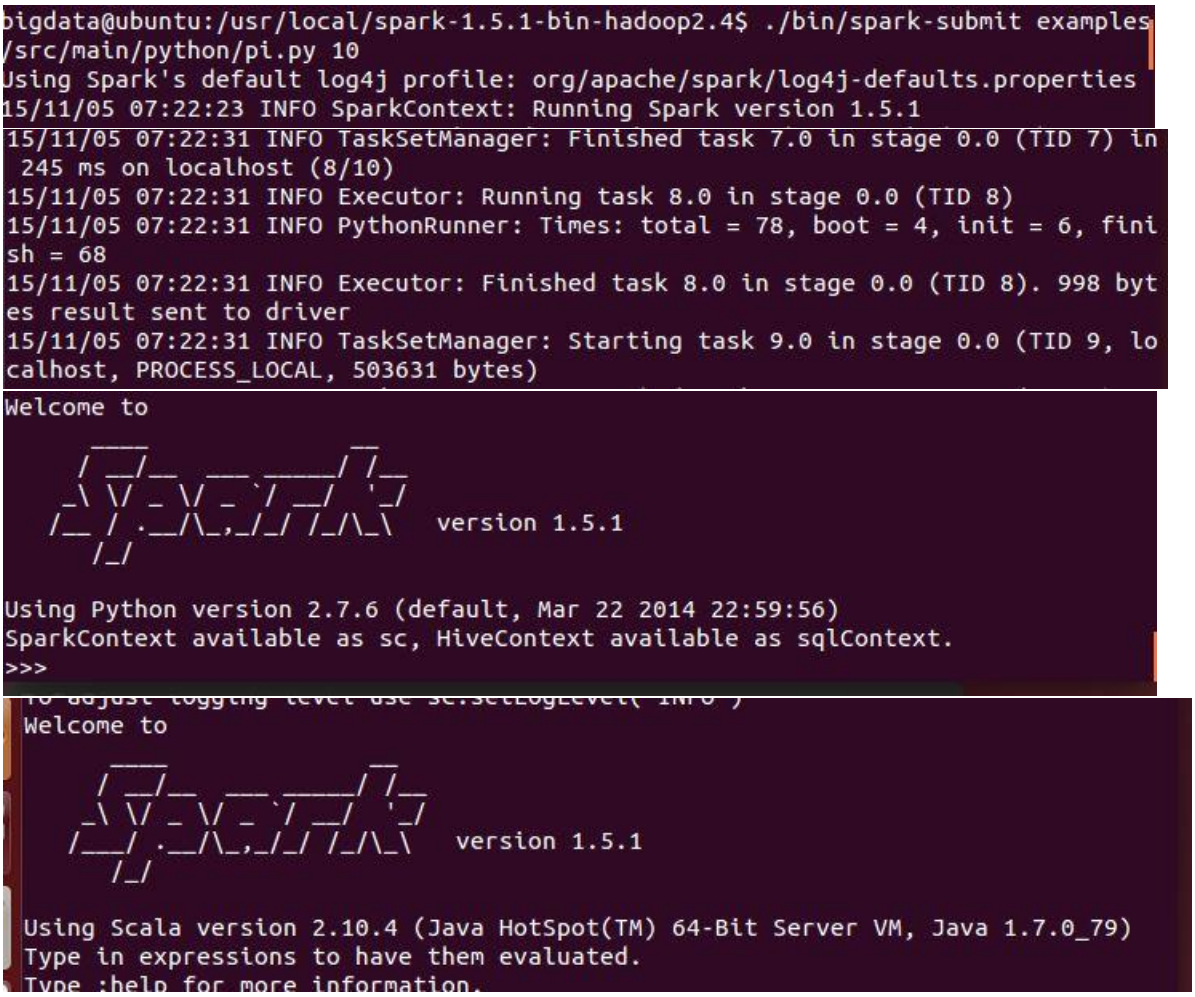
Anubha Bhargava – Homework 3

1. I installed and tested Spark. I ran a few tests on Spark, as shown below to test the installation. I imported a textfile into Spark as shown below.



I ran spark-submit, Spark in Python and Spark and Scala successfully as shown:



Then, I used the Wikipedia data to calculate TF-IDF and performed clustering. First, I parsed the Wikipedia dataset using the Wikipedia-extractor.py python script.

wikipedia-extractor.py

```
# Codes starts here
import lxml.etree
import os
tree = lxml.etree.parse('/home/bigdata/Documents/Homework2/Wikipedia/
Wikipedia20150602/enwiki-latest-pages-articles1.xml-
p000000010p000010000')
namespaces = {'ns':'http://www.mediawiki.org/xml/export-0.10/'}
i = 0
el_list = tree.xpath('//ns:page', namespaces = namespaces)
for el in el_list:
    title = el.xpath('.//ns:title', namespaces = namespaces)[0].text
    print title
    f = file('parse/' + str(i), 'w')
    text = el.xpath('.//ns:revision/ns:text', namespaces = namespaces)
[0].text
    f.write(title.encode('utf-8') + '\n')
    f.write(text.encode('utf-8'))
    i = i + 1
    f.close()
# Codes ends here
```

```
bigdata@ubuntu:~/Documents/Homework3/parse$ ls
0     1432  1868  2301  2737  3171  3606  4040  4476  4910  5345  5780  6214
1     1433  1869  2302  2738  3172  3607  4041  4477  4911  5346  5781  6215
10    1434  187   2303  2739  3173  3608  4042  4478  4912  5347  5782  6216
100   1435  1870  2304  274   3174  3609  4043  4479  4913  5348  5783  6217
1000  1436  1871  2305  2740  3175  361   4044  448   4914  5349  5784  6218
1001  1437  1872  2306  2741  3176  3610  4045  4480  4915  535   5785  6219
1002  1438  1873  2307  2742  3177  3611  4046  4481  4916  5350  5786  622
1003  1439  1874  2308  2743  3178  3612  4047  4482  4917  5351  5787  6220
1004  144   1875  2309  2744  3179  3613  4048  4483  4918  5352  5788  6221
1005  1440  1876  231   2745  318   3614  4049  4484  4919  5353  5789  6222
```

Then, I created a script called cluster_tfidf.py which completed both clustering and TFIDF.

cluster_tfidf.py

```
from pyspark.mllib.feature import HashingTF, IDF
from pyspark.mllib.clustering import KMeans, KMeansModel
from numpy import array
from math import sqrt
from pyspark import SparkContext

sc=SparkContext()
rdd=sc.wholeTextFiles("/home/bigdata/Documents/Homework3/parse2/").map(lambda
(name,text): text.split())
tf=HashingTF()
tfVectors=tf.transform(rdd).cache()
idf=IDF()
idfModel=idf.fit(tfVectors)
tfIdfVectors=idfModel.transform(tfVectors)
model=KMeans.train(tfIdfVectors, 2, maxIterations=10, runs=10,
initializationMode="random")
clusters = model.predict(tfIdfVectors)
dataWithClusters = tfIdfVectors.zip(clusters)
dataWithClusters.saveAsTextFile("/home/bigdata/Documents/Homework3/
dataWithClusters/")
tfIdfVectors.saveAsTextFile("/home/bigdata/Documents/Homework3/tfIdfVectors/")
```

This is the output that was received from the script:

```
(SparseVector(1048576, {2550: 12.0817, 3932: 0.3463, 19267: 3.0204,
22582: 3.0204, 30674: 1.2287, 36754: 0.3463, 38034: 1.2287, 43344:
3.0204, 46446: 2.1041, 49181: 1.411, 50570: 1.0388, 50583: 0.6247,
72971: 3.0204, 89423: 3.0204, 93626: 3.0204, 95620: 2.615, 95965:
6.3124, 102217: 3.0204, 121051: 3.0204, 137800: 3.0204, 141964: 3.0204,
148055: 3.0204, 152402: 2.615, 159190: 3.0204, 160072: 3.0204, 167368:
3.0765, 168186: 15.1021, 176055: 3.0204, 182263: 6.0408, 189356:
0.6225, 195374: 3.0204, 196347: 11.6364, 200383: 1.3157, 211696:
3.0204, 213150: 3.0204, 218450: 3.0204, 225722: 3.0204, 234013: 0.8355,
235870: 3.0204, 236823: 12.0817, 251696: 3.0204, 261722: 1.9218,
263998: 3.0204, 267062: 5.2299, 280048: 3.0204, 284784: 2.615, 292923:
0.7178, 293156: 0.1582, 323305: 6.2475, 326588: 2.3123, 338406: 2.1041,
340523: 2.3273, 348047: 4.0221, 354917: 1.3381, 356253: 2.1041, 358193:
3.0204, 362969: 0.669, 382094: 3.0204, 385604: 3.0204, 393486: 3.0204,
413733: 36.6094, 417558: 1.6341, 420024: 3.0204, 420843: 0.669, 429729:
2.615, 442539: 6.0408, 443614: 3.0204, 448789: 1.2287, 448852: 3.0204,
452315: 8.4165, 457375: 3.0204, 470253: 2.615, 497087: 3.0204, 498760:
```

This algorithm is similar to that in Mahout, as it dumps out the data in clusters.

2. I downloaded IBM System G Graph Tools and the Basketball data. I used the following for the edges and nodes in the software:

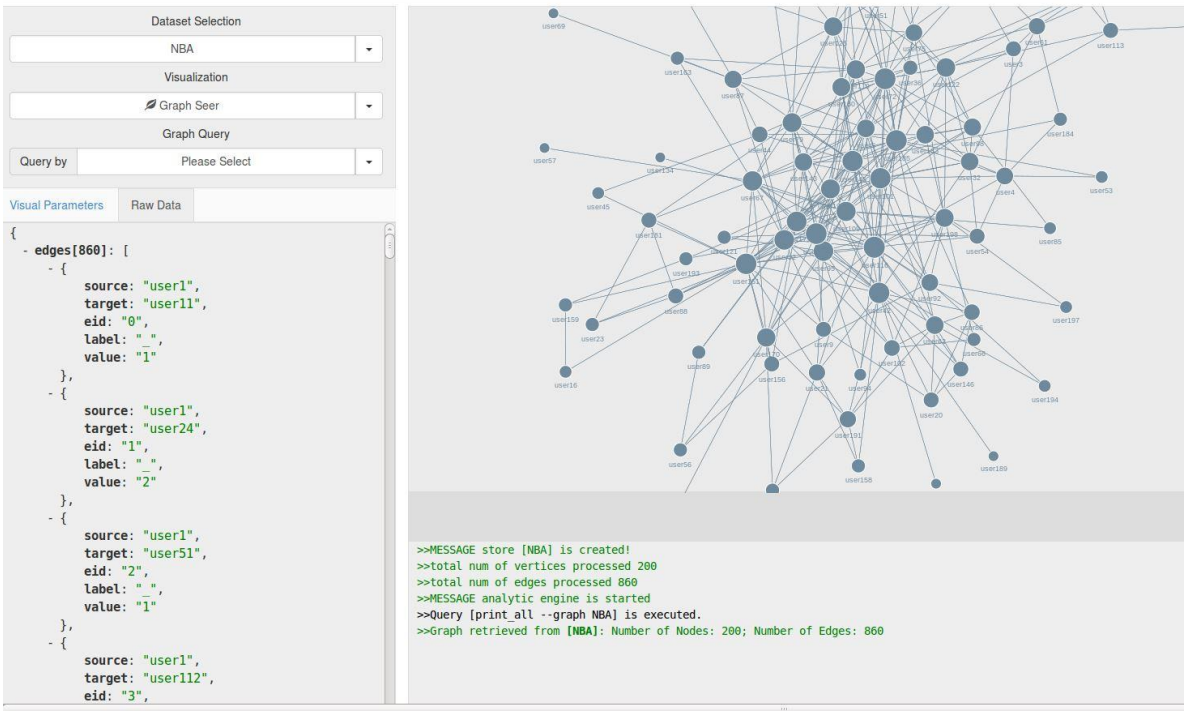
Basketball_Edge.csv

```
source,target,value
user11,user1,1
user24,user1,2
user51,user1,1
user112,user1,1
user125,user1,1
user135,user1,1
user180,user1,1
user185,user1,1
user6,user2,2
user41,user2,7
user111,user2,1
user195,user2,1
user4,user3,1
user5,user3,1
user17,user3,1
user61,user3,2
user122,user3,1
user3,user4,1
user32,user4,1
user53,user4,1
user54,user4,1
user101,user4,1
user116,user4,1
user184,user4,1
user185,user4,1
user3,user5,1
user67,user5,2
```

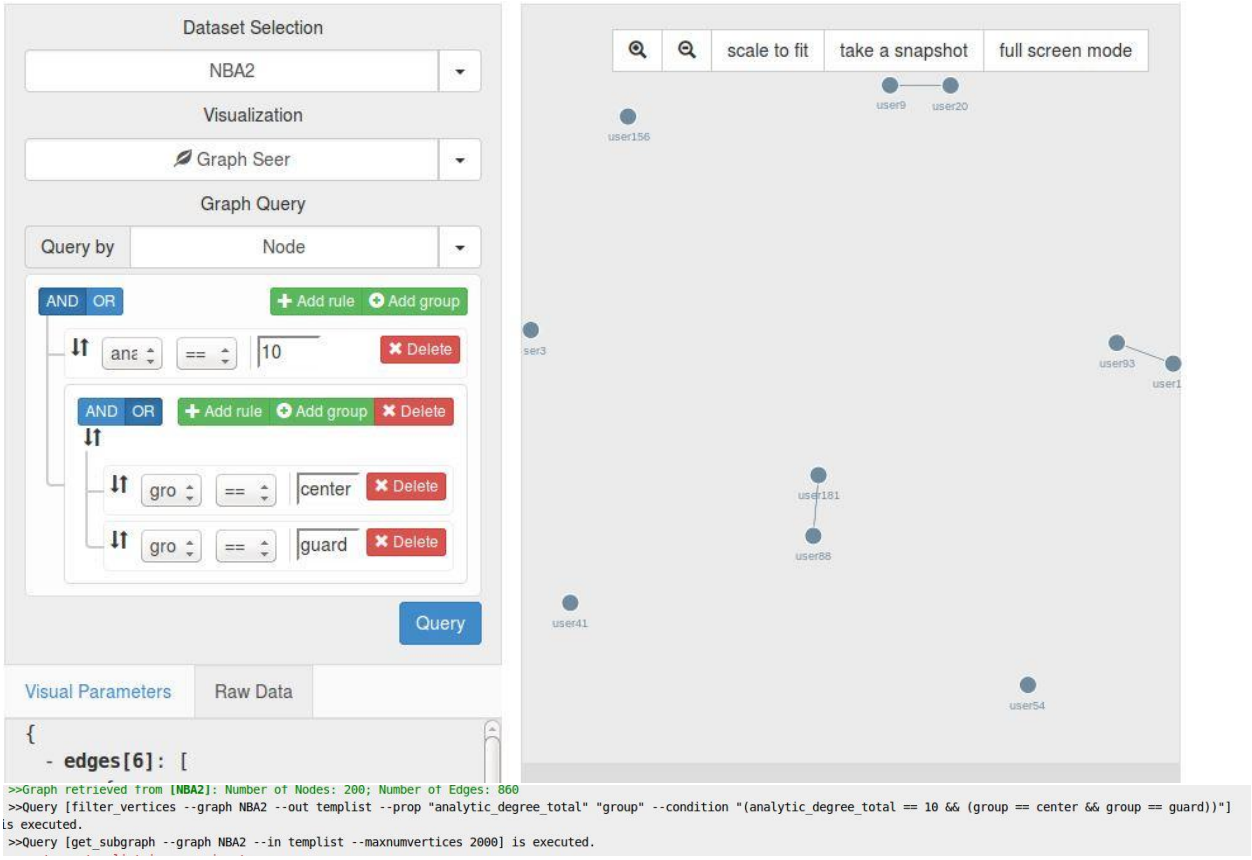
Basketball_Node.csv

```
id,group,"play name"
user1,forward,"Alaa Abdelnaby"
user2,center,"Kareem Abdul-jabbar"
user3,guard,"Mahmo Abdul-rauf"
user4,guard,"Tariq Abdul-wahad"
user5,forward,"Shareef Abdur-rahim"
user6,forward,"Tom Abernethy"
user7,guard,"Forest Able"
user8,forward,"John Abramovic"
user9,guard,"Alex Acker"
user10,guard,"Donald Ackerman"
user11,center,"Mark Acres"
user12,forward,"Charles Acton"
user13,center,"Alvan Adams"
user14,forward,"Don Adams"
user15,forward,"George Adams"
user16,guard,"Hassan Adams"
user17,guard,"Michael Adams"
user18,forward,"Rafael Addison"
user19,guard,"Rick Adelman"
user20,guard,"Arron Afflalo"
user21,guard,"Maurice Ager"
user22,forward,"Mark Aguirre"
user23,guard,"Blake Ahearn"
user24,guard,"Danny Ainge"
user25,forward,"Matt Aitch"
user26,center,"Alexis Ajinca"
user27,center,"Henry Akin"
```

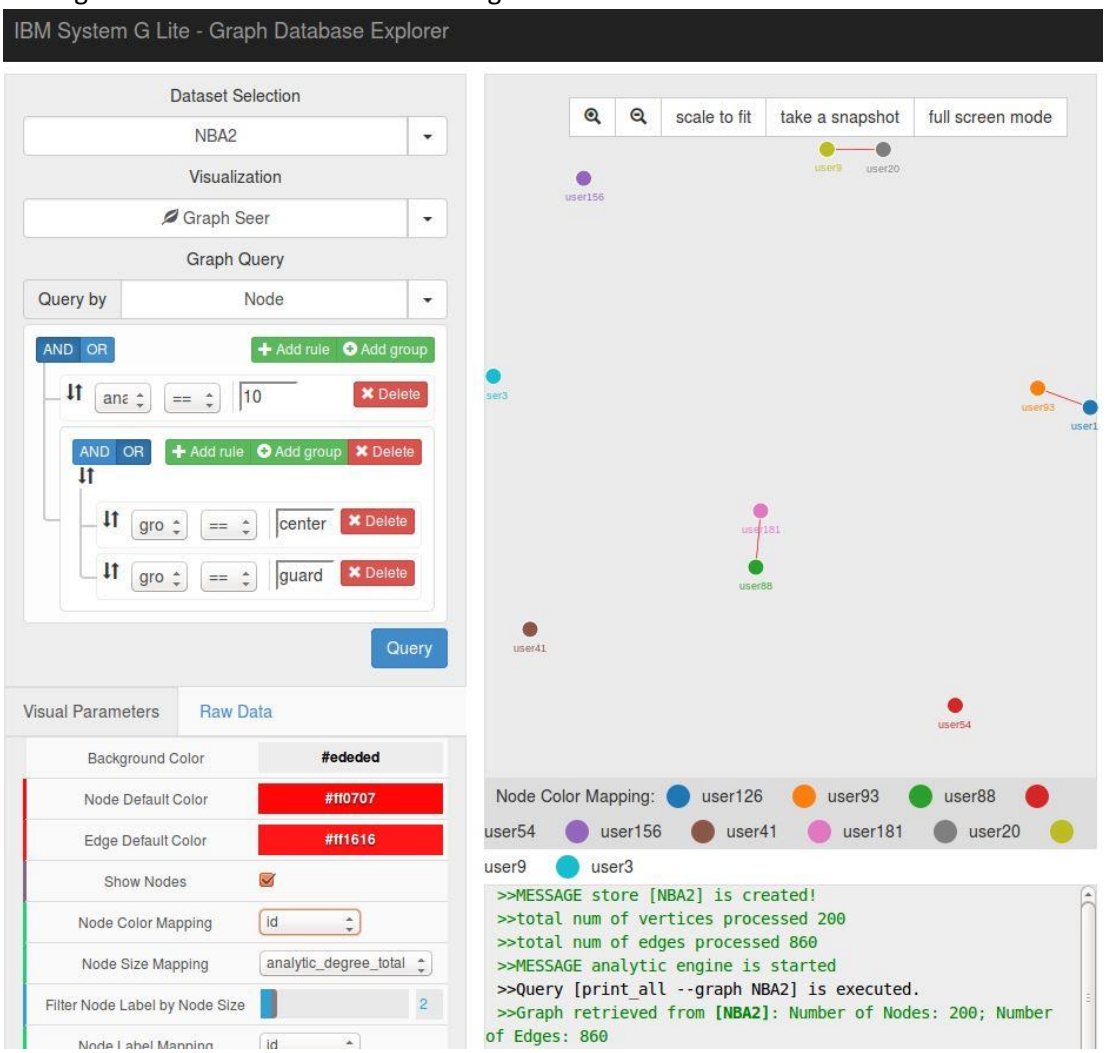
I created a knowledge graph using this data and injected it into a graph database.



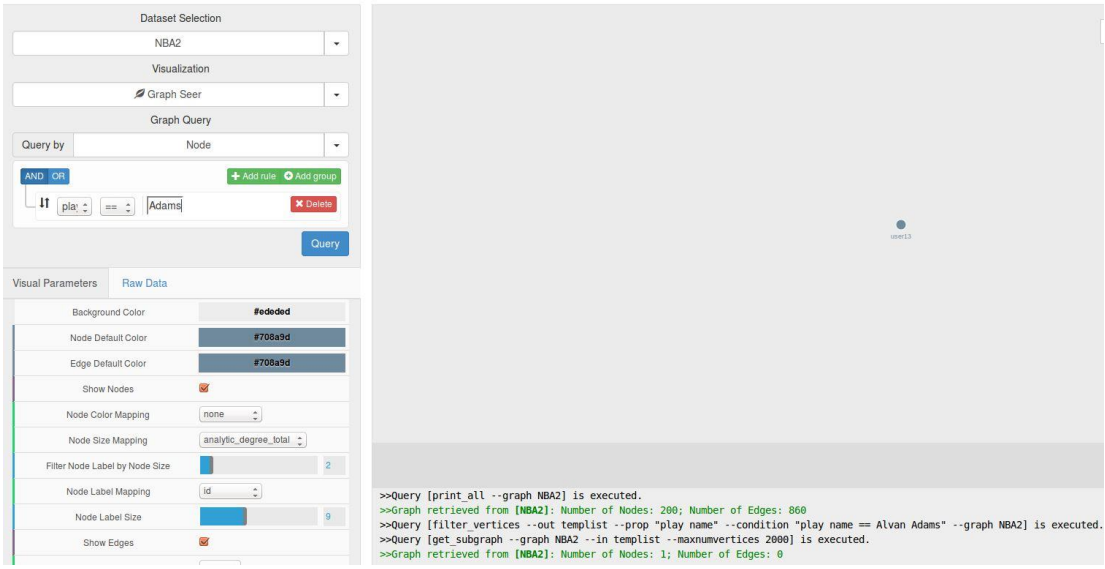
I then tried some visualizations with my queries, as shown below. I set the analytic degree total to 10, and group to center and guard and received the following results.



I changed the colors of the nodes and edges.



I looked up the player Alvan Adams and received one player, as displayed.



I was unable to use the Wikipedia dataset, as I couldn’t convert it from a .sql file. I successfully analyzed the Basketball data, however.