Casey Kongpanickul

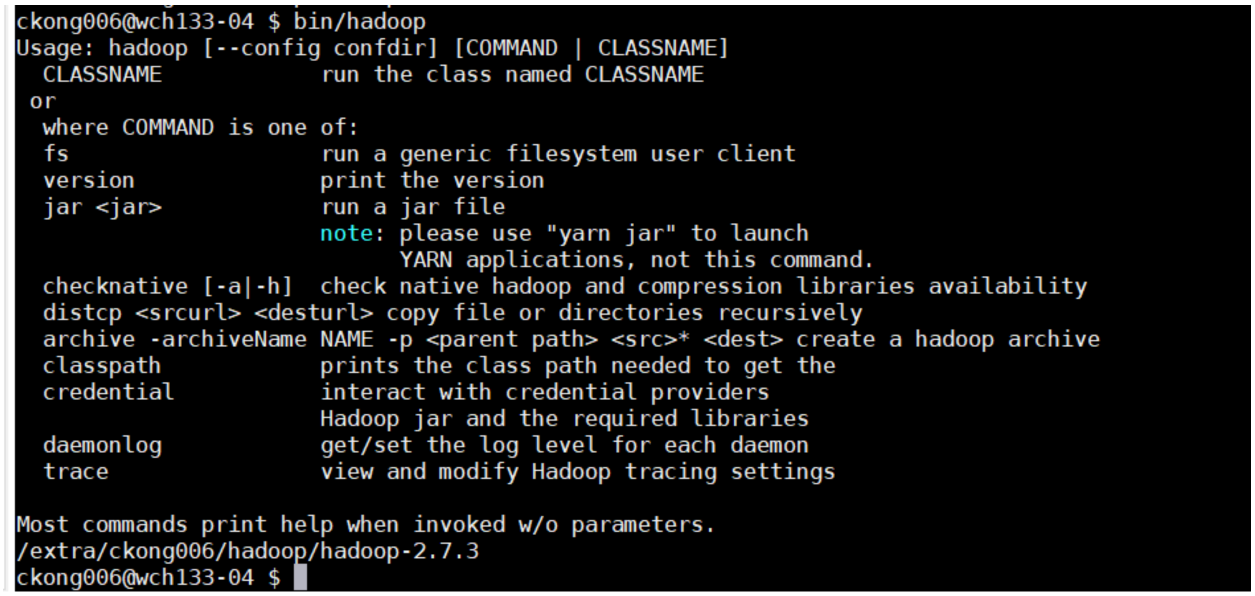
Assignment Phase 2 – Report

The strategy used to solve the 4 problems for phase two came in a couple steps. The first step was to learn how to use Python scripts as inputs for Hadoop instead of using Java. This made sense at the time because I have little to no experience in Java. I was able to find a free resource from Udacity videos loaded onto YouTube.

0:01 / 0:52
Intro to Hadoop and MapReduce
Udacity - 1 / 72
Introdução
Udacity
0:53
Fontes de dados
Udacity
1:16
Big Data - Intro to Hadoop and MapReduce
Udacity
0:38
Big Data - Intro to Hadoop and MapReduce
Udacity
0:30
Definição de Big Data
Udacity
0:21
Challenges - Intro to Hadoop and MapReduce
Udacity
0:20

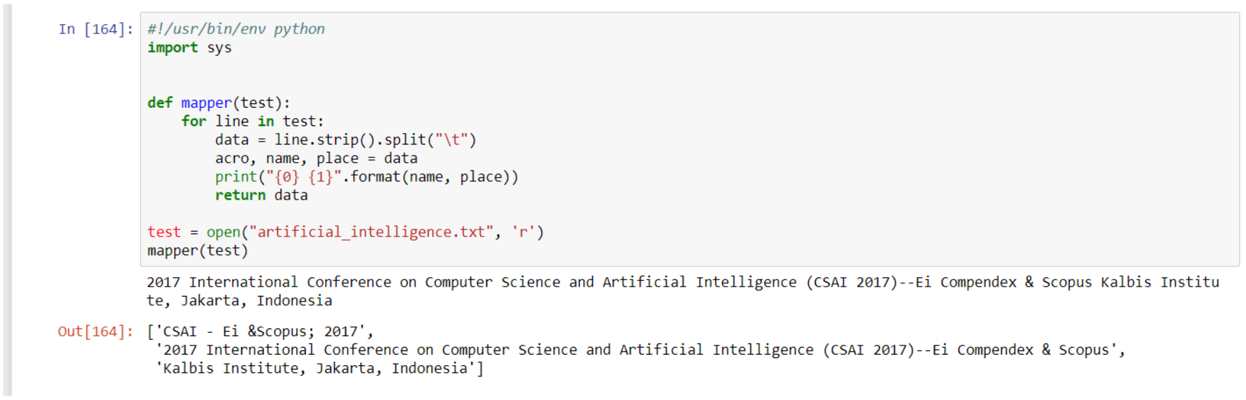
This is a class that explains Hadoop at a very high level as well as an introduction on how to write a mapper and reducer in python as well as run those scripts in Hadoop using the streaming jar.

After this I went ahead and set up the Hadoop environment through the VM.



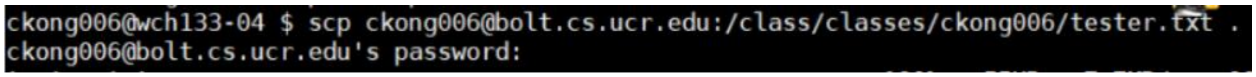
```
ckong006@wch133-04 $ bin/hadoop
Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]
  CLASSNAME            run the class named CLASSNAME
 or
  where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
                       note: please use "yarn jar" to launch
                             YARN applications, not this command.
  checknative [-a|-h]  check native hadoop and compression libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
  classpath            prints the class path needed to get the
  credential           interact with credential providers
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  trace                view and modify Hadoop tracing settings

Most commands print help when invoked w/o parameters.
/extra/ckong006/hadoop/hadoop-2.7.3
ckong006@wch133-04 $
```

From this point I was able to use Jupyter to go ahead and write and test my code before bringing it into the VM for Hadoop to run.



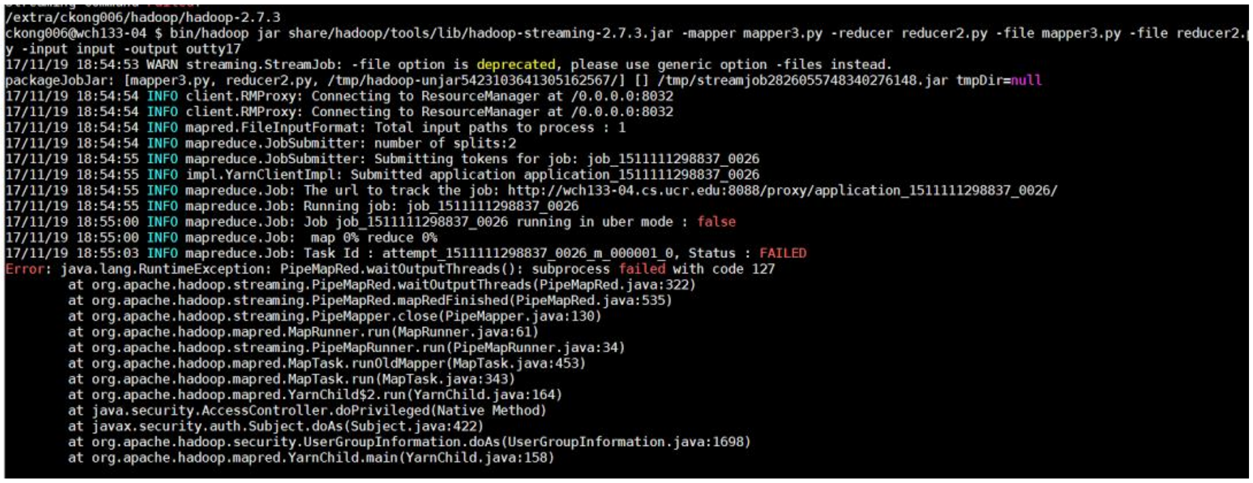
```
In [164]: #!/usr/bin/env python
          import sys


          def mapper(test):
              for line in test:
                  data = line.strip().split("\t")
                  acro, name, place = data
                  print("{0} {1}".format(name, place))
                  return data

          test = open("artificial_intelligence.txt", 'r')
          mapper(test)

          2017 International Conference on Computer Science and Artificial Intelligence (CSAI 2017)--Ei Compendex & Scopus Kalbis Institu
          te, Jakarta, Indonesia

Out[164]: ['CSAI - Ei &Scopus; 2017',
           '2017 International Conference on Computer Science and Artificial Intelligence (CSAI 2017)--Ei Compendex & Scopus',
           'Kalbis Institute, Jakarta, Indonesia']
```

However, when it came time to put everything together in Hadoop I ran into several issues mostly stemming from inexperience with the system itself. First it took me a while to get acquainted with using WinSCP to bring files in and retrieve files from the VM. I finally tackled this problem with the below code:



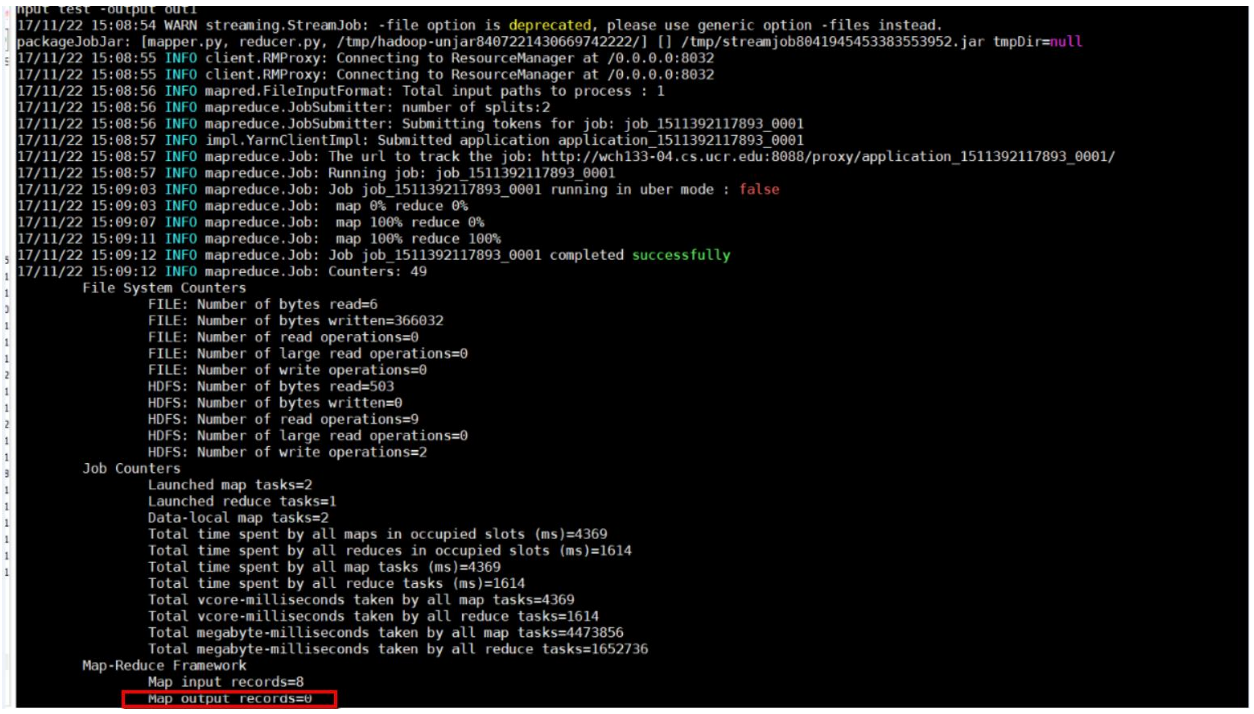
```
ckong006@wch133-04 $ scp ckong006@bolt.cs.ucr.edu:/class/classes/ckong006/tester.txt .
ckong006@bolt.cs.ucr.edu's password:
```

Finally I had all the pieces and was ready to run the code in Hadoop. The first obstacle at this point was a subroutine run error.



```
/extra/ckong006/hadoop/hadoop-2.7.3
ckong006@wch133-04 $ bin/hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar -mapper mapper3.py -reducer reducer2.py -file mapper3.py -file reducer2.
y -input input -output outty17
17/11/19 18:54:53 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper3.py, reducer2.py, /tmp/hadoop-unjar5423103641305162567/] [] /tmp/streamjob2826055748340276148.jar tmpDir=null
17/11/19 18:54:54 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/19 18:54:54 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/19 18:54:54 INFO mapred.FileInputFormat: Total input paths to process : 1
17/11/19 18:54:54 INFO mapreduce.JobSubmitter: number of splits:2
17/11/19 18:54:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1511111298837_0026
17/11/19 18:54:55 INFO impl.YarnClientImpl: Submitted application application_1511111298837_0026
17/11/19 18:54:55 INFO mapreduce.Job: The url to track the job: http://wch133-04.cs.ucr.edu:8088/proxy/application_1511111298837_0026/
17/11/19 18:54:55 INFO mapreduce.Job: Running job: job_1511111298837_0026
17/11/19 18:55:00 INFO mapreduce.Job: Job job_1511111298837_0026 running in uber mode : false
17/11/19 18:55:00 INFO mapreduce.Job:  map 0% reduce 0%
17/11/19 18:55:03 INFO mapreduce.Job: Task Id : attempt_1511111298837_0026_m_000001_0, Status : FAILED
Error: java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 127
        at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:322)
        at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:535)
        at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:130)
        at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:61)
        at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:34)
        at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:453)
        at org.apache.hadoop.mapred.MapTask.run(MapTask.java:343)
        at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1698)
        at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)
```

After some research and some help from Ekta it turns out that Hadoop streaming jar did not seem to like code in Python 3, it took a second but I re-worked my code into Python 2.7. My final obstacle was that Hadoop would run but nothing was being output from the mapper. I re-worked my code multiple times but ultimately could not figure out what was going on.

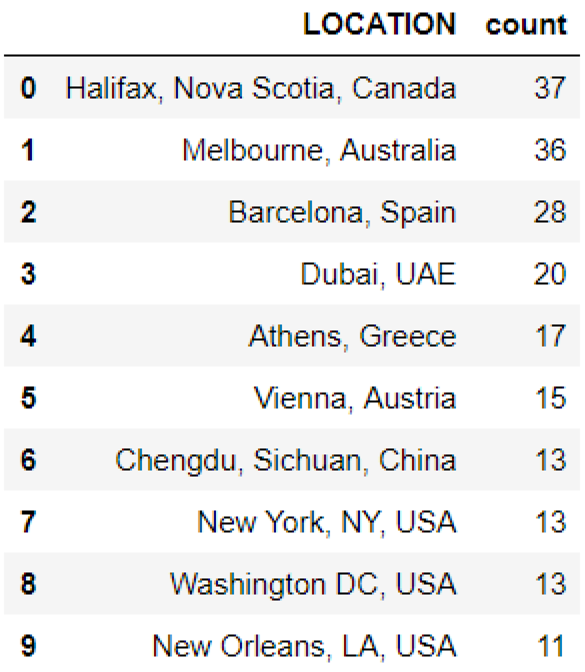


```
nput test -output out1
17/11/22 15:08:54 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-unjar8407221430669742222/] [] /tmp/streamjob8041945453383553952.jar tmpDir=null
17/11/22 15:08:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/22 15:08:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/22 15:08:56 INFO mapred.FileInputFormat: Total input paths to process : 1
17/11/22 15:08:56 INFO mapreduce.JobSubmitter: number of splits:2
17/11/22 15:08:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1511392117893_0001
17/11/22 15:08:57 INFO impl.YarnClientImpl: Submitted application application_1511392117893_0001
17/11/22 15:08:57 INFO mapreduce.Job: The url to track the job: http://wch133-04.cs.ucr.edu:8088/proxy/application_1511392117893_0001/
17/11/22 15:08:57 INFO mapreduce.Job: Running job: job_1511392117893_0001
17/11/22 15:09:03 INFO mapreduce.Job: Job job_1511392117893_0001 running in uber mode : false
17/11/22 15:09:03 INFO mapreduce.Job:  map 0% reduce 0%
17/11/22 15:09:07 INFO mapreduce.Job:  map 100% reduce 0%
17/11/22 15:09:11 INFO mapreduce.Job:  map 100% reduce 100%
17/11/22 15:09:12 INFO mapreduce.Job: Job job_1511392117893_0001 completed successfully
17/11/22 15:09:12 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=6
                FILE: Number of bytes written=366032
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=503
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=4369
                Total time spent by all reduces in occupied slots (ms)=1614
                Total time spent by all map tasks (ms)=4369
                Total time spent by all reduce tasks (ms)=1614
                Total vcore-milliseconds taken by all map tasks=4369
                Total vcore-milliseconds taken by all reduce tasks=1614
                Total megabyte-milliseconds taken by all map tasks=4473856
                Total megabyte-milliseconds taken by all reduce tasks=1652736
        Map-Reduce Framework
                Map input records=8
                Map output records=0
```

At this point I was pretty defeated and running out of time. I decided as a backup plan I would just solve the problem using Python Pandas.

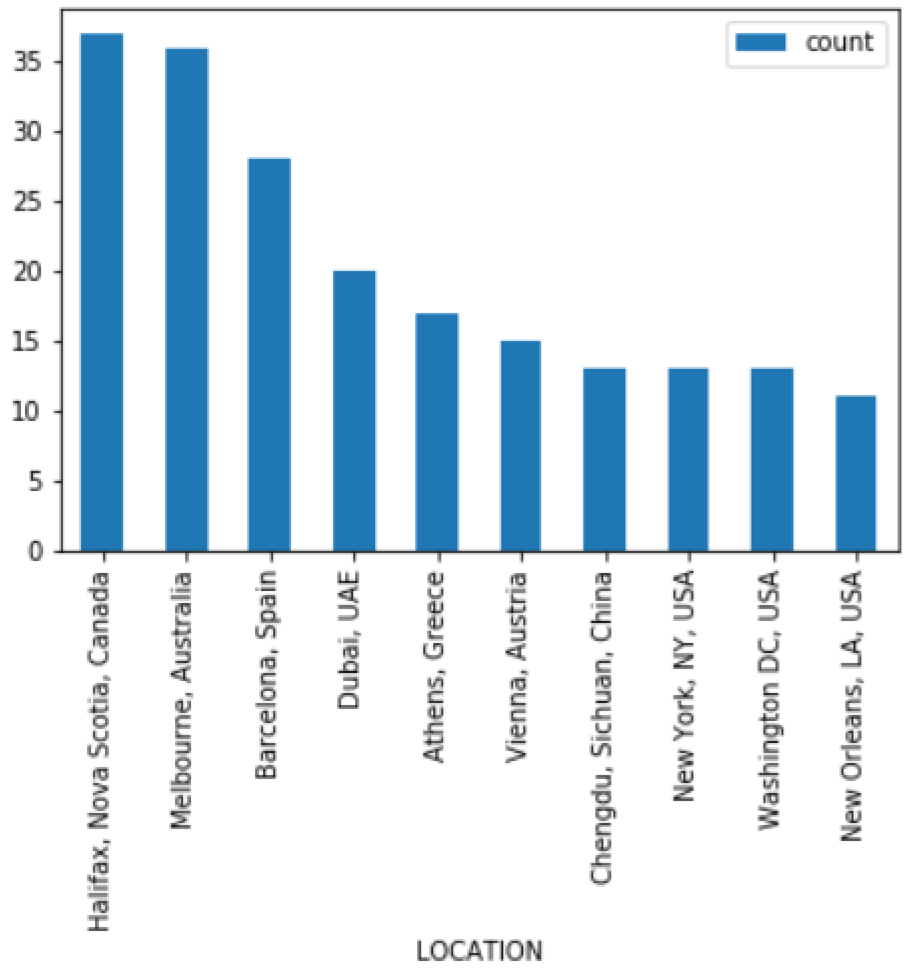
I will attach the jupyter notebook in my submission for my assignment. Here are the outputs from the problems

#1



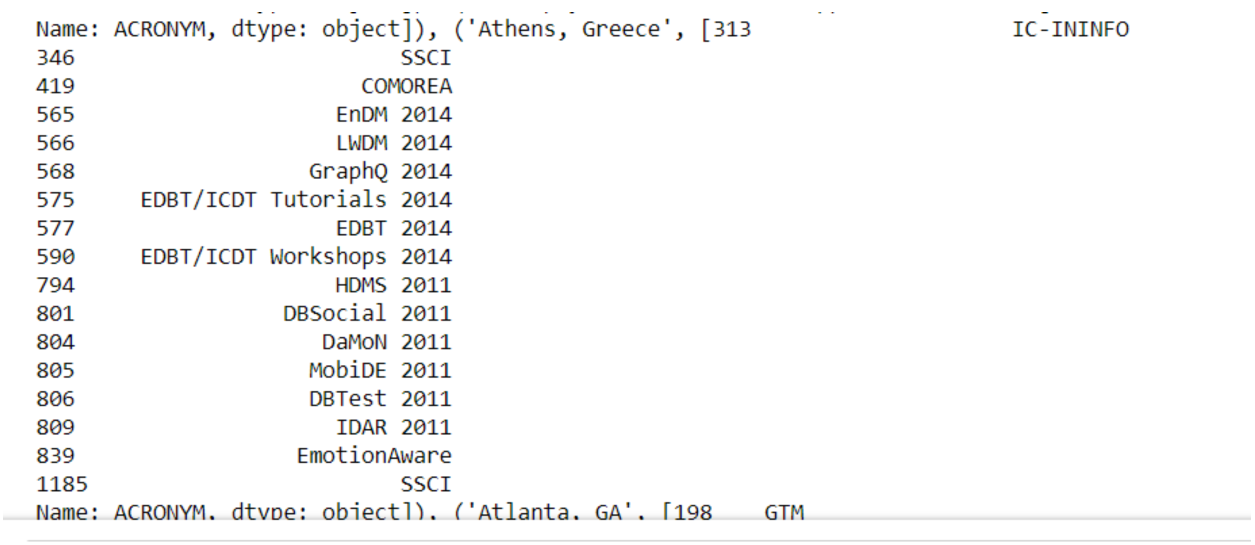
	LOCATION	count
0	Halifax, Nova Scotia, Canada	37
1	Melbourne, Australia	36
2	Barcelona, Spain	28
3	Dubai, UAE	20
4	Athens, Greece	17
5	Vienna, Austria	15
6	Chengdu, Sichuan, China	13
7	New York, NY, USA	13
8	Washington DC, USA	13
9	New Orleans, LA, USA	11

Visualization for #1



These are the top 10 locations for sheer number of conferences.

#2

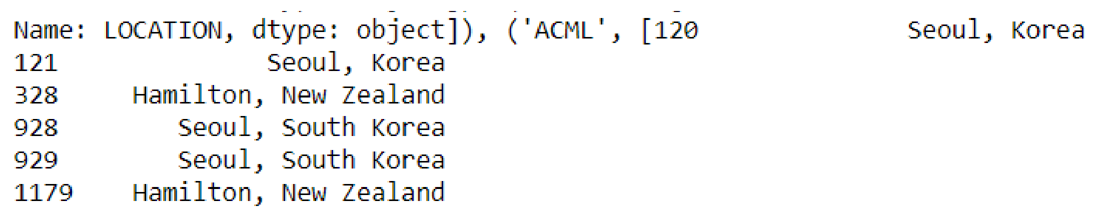


```
Name: ACRONYM, dtype: object]), ('Athens, Greece', [313                    IC-ININFO
346                    SSCI
419                 COMOREA
565                EnDM 2014
566                LWDM 2014
568              GraphQ 2014
575     EDBT/ICDT Tutorials 2014
577                EDBT 2014
590     EDBT/ICDT Workshops 2014
794                HDMS 2011
801            DBSocial 2011
804               DaMoN 2011
805              MobiDE 2011
806              DBTest 2011
809                IDAR 2011
839             EmotionAware
1185                   SSCI
Name: ACRONYM, dtype: object]), ('Atlanta, GA', [198    GTM
```

Conferences grouped by place. For this problem I used a loop to iterate through the lines and create a dictionary with the place being the key and adding values for each conference.

#3

This problem was very similar to #2 except I reversed the order in which the dictionary was built to view conferences that had been done in multiple locations



```
Name: LOCATION, dtype: object]), ('ACML', [120              Seoul, Korea
121              Seoul, Korea
328     Hamilton, New Zealand
928         Seoul, South Korea
929         Seoul, South Korea
1179    Hamilton, New Zealand
```

In conclusion, I did actually learn a lot about Hadoop (and pandas / matplotlib after stuggling with Hadoop). It's unfortunate that I couldn't get Hadoop to do what I want in the end. But it was not for lack of trying. I will continue to try this problem on my own time until I master the tool.