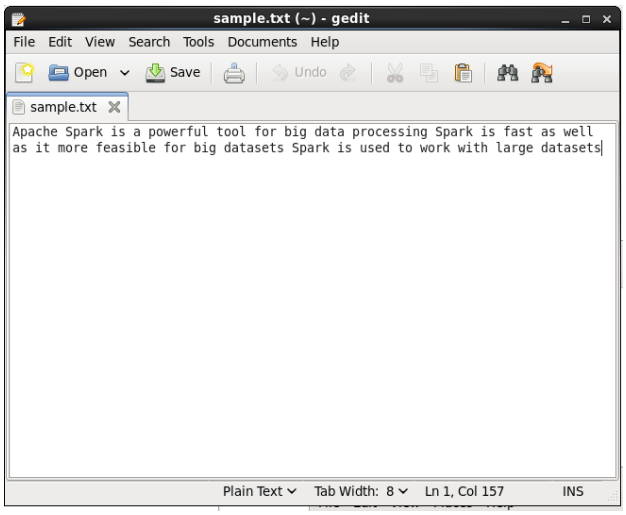
Assignment 6

Name: Aditya Wandhekar
Srn no: 202201449
Roll no: 34
Div: A(A2)

PS: Perform Word Count with Spark Python

CODE & OUTPUT:

INPUT:




```
[cloudera@quickstart ~]$ pyspark
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)
SparkContext available as sc, HiveContext available as sqlContext.
```

```
>>> words = sc.textFile("file:///home/cloudera/sample.txt").flatMap(lambda line:
 line.split(" "))
>>> a=words.map(lambda word: (word, 1))
>>> b=a.reduceByKey(lambda a,b:a+b)
>>> b.collect()
[(u'is', 3), (u'powerful', 1), (u'it', 1), (u'as', 2), (u'for', 2), (u'data', 1)
, (u'fast', 1), (u'to', 1), (u'Apache', 1), (u'Spark', 3), (u'more', 1), (u'used
', 1), (u'big', 2), (u'tool', 1), (u'processing', 1), (u'with', 1), (u'a', 1), (
u'datasets', 2), (u'work', 1), (u'well', 1), (u'feasible', 1), (u'large', 1)]
```

Explaining the steps(code):

1. Loading Data:

Assume you have a text file and you want to count the occurrences of each word in that file.

Here, sc.textFile() creates an RDD from a text file. Each element in the RDD corresponds to a line in the text file.

Mapping to Word Pairs (map operation):

You need to split each line into words and then assign a count of 1 for each word.

- flatMap: This operation is used to split each line into individual words. It is similar to a map but can produce multiple output values per input value.
- map: This operation takes each word and creates a pair (word, 1). This is the intermediate data structure for counting occurrences.

2. Reducing by Key (reduceByKey operation):

Now, you need to aggregate the counts by word. This is where the reduceByKey operation comes into play.

- reduceByKey: This operation takes the pairs and applies a function (lambda x, y: x + y) to combine the values of each word. Here, x and y represent the counts of a word, and they are summed together. If the word appears multiple times, their counts will be added.

3. Collecting the Results (collect operation):

Once the counting is done, you can collect the results back to the driver (your local machine or where the Spark session is running).

- collect(): This brings the result (which is distributed across the Spark cluster) back to your local machine, allowing you to view or process it.

4. Displaying the Results:

Now that you have the result, you can print it or process it as needed.

```
>>> b.collect()
[(u'is', 3), (u'powerful', 1), (u'it', 1), (u'as', 2), (u'for', 2), (u'data', 1)
, (u'fast', 1), (u'to', 1), (u'Apache', 1), (u'Spark', 3), (u'more', 1), (u'used
', 1), (u'big', 2), (u'tool', 1), (u'processing', 1), (u'with', 1), (u'a', 1), (
u'datasets', 2), (u'work', 1), (u'well', 1), (u'feasible', 1), (u'large', 1)]
```