**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Big-Data  Assignment-2  4/21/2015  Bhushan O Kothari |

**Step 1: Reading the text file.**

**>>> lines = sc.textFile("Documents/data")**

**>>> lines.collect()**

[u'doc6:of,crazy,fox,jumped,fox,ran', u'doc6:fox,ran,fast,over,fence,too,high', u'doc1:fox,jumped', u'doc1:fox,jumped,over,the,fence', u'doc1:of,crazy,fox,jumped,fox,ran', u'doc1:fox,ran,fast,over,fence,too,high', u'doc2:a,crazy,fox,jumped', u'doc2:a,crazy,fox,jumped,over,the,fence,again', u'doc3:fox,ran,fast,over,fence', u'doc1:fox,is,high,on,sugar', u'doc2:a,crazy,fox,ran,ran,ran,fast', u'doc3:a,crazy,fox,jumped,jumped,jumped,jumped', u'doc4:a,crazy,fox,ran,jumped,ran,jumped', u'doc4:a,crazy,fox,jumped,jumped', u'doc4:a,crazy,fox,jumped,jumped', u'doc5:a,crazy,fox,jumped,over,fence,very,high', u'doc5:a,crazy,fox,jumped,over,the,fence,again', u'doc6:book,reading,about,fox,and,fence', u'doc1:crazy,fox,jumped']

**Step 2: Splitting the input lines on ‘:’ and further splitting the comma separated string.**

**>>> words = lines.map(lambda x: (x.split(':')[0], x.split(':')[1].split(',')))**

**>>> words.collect()**

[(u'doc6', [u'of', u'crazy', u'fox', u'jumped', u'fox', u'ran']),

(u'doc6', [u'fox', u'ran', u'fast', u'over', u'fence', u'too', u'high']),

(u'doc1', [u'fox', u'jumped']), (u'doc1', [u'fox', u'jumped', u'over', u'the', u'fence']),

(u'doc1', [u'of', u'crazy', u'fox', u'jumped', u'fox', u'ran']), (u'doc1', [u'fox', u'ran', u'fast', u'over', u'fence', u'too', u'high']),

(u'doc2', [u'a', u'crazy', u'fox', u'jumped']),

(u'doc2', [u'a', u'crazy', u'fox', u'jumped', u'over', u'the', u'fence', u'again']),

(u'doc3', [u'fox', u'ran', u'fast', u'over', u'fence']), (u'doc1', [u'fox', u'is', u'high', u'on', u'sugar']),

(u'doc2', [u'a', u'crazy', u'fox', u'ran', u'ran', u'ran', u'fast']),

(u'doc3', [u'a', u'crazy', u'fox', u'jumped', u'jumped', u'jumped', u'jumped']), (u'doc4', [u'a', u'crazy', u'fox', u'ran', u'jumped', u'ran', u'jumped']),

(u'doc4', [u'a', u'crazy', u'fox', u'jumped', u'jumped']),

(u'doc4', [u'a', u'crazy', u'fox', u'jumped', u'jumped']),

(u'doc5', [u'a', u'crazy', u'fox', u'jumped', u'over', u'fence', u'very', u'high']),

(u'doc5', [u'a', u'crazy', u'fox', u'jumped', u'over', u'the', u'fence', u'again']),

(u'doc6', [u'book', u'reading', u'about', u'fox', u'and', u'fence']),

(u'doc1', [u'crazy', u'fox', u'jumped'])]

**Step 3: To associate Key, with string of values, we need to use flatMapValues() function**

**>>> def f(words):return words**

**...**

**>>> docWordRdd = words.flatMapValues(f)**

**>>> docWordRdd.collect()**

[(u'doc6', u'of'), (u'doc6', u'crazy'), (u'doc6', u'fox'), (u'doc6', u'jumped'), (u'doc6', u'fox'), (u'doc6', u'ran'), (u'doc6', u'fox'), (u'doc6', u'ran'), (u'doc6', u'fast'), (u'doc6', u'over'), (u'doc6', u'fence'), (u'doc6', u'too'), (u'doc6', u'high'), (u'doc1', u'fox'), (u'doc1', u'jumped'), (u'doc1', u'fox'), (u'doc1', u'jumped'), (u'doc1', u'over'), (u'doc1', u'the'), (u'doc1', u'fence'), (u'doc1', u'of'), (u'doc1', u'crazy'), (u'doc1', u'fox'), (u'doc1', u'jumped'), (u'doc1', u'fox'), (u'doc1', u'ran'), (u'doc1', u'fox'), (u'doc1', u'ran'), (u'doc1', u'fast'), (u'doc1', u'over'), (u'doc1', u'fence'), (u'doc1', u'too'), (u'doc1', u'high'), (u'doc2', u'a'), (u'doc2', u'crazy'), (u'doc2', u'fox'), (u'doc2', u'jumped'), (u'doc2', u'a'), (u'doc2', u'crazy'), (u'doc2', u'fox'), (u'doc2', u'jumped'), (u'doc2', u'over'), (u'doc2', u'the'), (u'doc2', u'fence'), (u'doc2', u'again'), (u'doc3', u'fox'), (u'doc3', u'ran'), (u'doc3', u'fast'), (u'doc3', u'over'), (u'doc3', u'fence'), (u'doc1', u'fox'), (u'doc1', u'is'), (u'doc1', u'high'), (u'doc1', u'on'), (u'doc1', u'sugar'), (u'doc2', u'a'), (u'doc2', u'crazy'), (u'doc2', u'fox'), (u'doc2', u'ran'), (u'doc2', u'ran'), (u'doc2', u'ran'), (u'doc2', u'fast'), (u'doc3', u'a'), (u'doc3', u'crazy'), (u'doc3', u'fox'), (u'doc3', u'jumped'), (u'doc3', u'jumped'), (u'doc3', u'jumped'), (u'doc3', u'jumped'), (u'doc4', u'a'), (u'doc4', u'crazy'), (u'doc4', u'fox'), (u'doc4', u'ran'), (u'doc4', u'jumped'), (u'doc4', u'ran'), (u'doc4', u'jumped'), (u'doc4', u'a'), (u'doc4', u'crazy'), (u'doc4', u'fox'), (u'doc4', u'jumped'), (u'doc4', u'jumped'), (u'doc4', u'a'), (u'doc4', u'crazy'), (u'doc4', u'fox'), (u'doc4', u'jumped'), (u'doc4', u'jumped'), (u'doc5', u'a'), (u'doc5', u'crazy'), (u'doc5', u'fox'), (u'doc5', u'jumped'), (u'doc5', u'over'), (u'doc5', u'fence'), (u'doc5', u'very'), (u'doc5', u'high'), (u'doc5', u'a'), (u'doc5', u'crazy'), (u'doc5', u'fox'), (u'doc5', u'jumped'), (u'doc5', u'over'), (u'doc5', u'the'), (u'doc5', u'fence'), (u'doc5', u'again'), (u'doc6', u'book'), (u'doc6', u'reading'), (u'doc6', u'about'), (u'doc6', u'fox'), (u'doc6', u'and'), (u'doc6', u'fence'), (u'doc1', u'crazy'), (u'doc1', u'fox'), (u'doc1', u'jumped')]

**Step 4: Filter the words with length less than 3.**

**>>> docWordRddFiltered = docWordRdd.filter(lambda x : len(x[1])>2)**

**>>> docWordRddFiltered.collect()**

[(u'doc6', u'crazy'), (u'doc6', u'fox'), (u'doc6', u'jumped'), (u'doc6', u'fox'), (u'doc6', u'ran'), (u'doc6', u'fox'), (u'doc6', u'ran'), (u'doc6', u'fast'), (u'doc6', u'over'), (u'doc6', u'fence'), (u'doc6', u'too'), (u'doc6', u'high'), (u'doc1', u'fox'), (u'doc1', u'jumped'), (u'doc1', u'fox'), (u'doc1', u'jumped'), (u'doc1', u'over'), (u'doc1', u'the'), (u'doc1', u'fence'), (u'doc1', u'crazy'), (u'doc1', u'fox'), (u'doc1', u'jumped'), (u'doc1', u'fox'), (u'doc1', u'ran'), (u'doc1', u'fox'), (u'doc1', u'ran'), (u'doc1', u'fast'), (u'doc1', u'over'), (u'doc1', u'fence'), (u'doc1', u'too'), (u'doc1', u'high'), (u'doc2', u'crazy'), (u'doc2', u'fox'), (u'doc2', u'jumped'), (u'doc2', u'crazy'), (u'doc2', u'fox'), (u'doc2', u'jumped'), (u'doc2', u'over'), (u'doc2', u'the'), (u'doc2', u'fence'), (u'doc2', u'again'), (u'doc3', u'fox'), (u'doc3', u'ran'), (u'doc3', u'fast'), (u'doc3', u'over'), (u'doc3', u'fence'), (u'doc1', u'fox'), (u'doc1', u'high'), (u'doc1', u'sugar'), (u'doc2', u'crazy'), (u'doc2', u'fox'), (u'doc2', u'ran'), (u'doc2', u'ran'), (u'doc2', u'ran'), (u'doc2', u'fast'), (u'doc3', u'crazy'), (u'doc3', u'fox'), (u'doc3', u'jumped'), (u'doc3', u'jumped'), (u'doc3', u'jumped'), (u'doc3', u'jumped'), (u'doc4', u'crazy'), (u'doc4', u'fox'), (u'doc4', u'ran'), (u'doc4', u'jumped'), (u'doc4', u'ran'), (u'doc4', u'jumped'), (u'doc4', u'crazy'), (u'doc4', u'fox'), (u'doc4', u'jumped'), (u'doc4', u'jumped'), (u'doc4', u'crazy'), (u'doc4', u'fox'), (u'doc4', u'jumped'), (u'doc4', u'jumped'), (u'doc5', u'crazy'), (u'doc5', u'fox'), (u'doc5', u'jumped'), (u'doc5', u'over'), (u'doc5', u'fence'), (u'doc5', u'very'), (u'doc5', u'high'), (u'doc5', u'crazy'), (u'doc5', u'fox'), (u'doc5', u'jumped'), (u'doc5', u'over'), (u'doc5', u'the'), (u'doc5', u'fence'), (u'doc5', u'again'), (u'doc6', u'book'), (u'doc6', u'reading'), (u'doc6', u'about'), (u'doc6', u'fox'), (u'doc6', u'and'), (u'doc6', u'fence'), (u'doc1', u'crazy'), (u'doc1', u'fox'), (u'doc1', u'jumped')]

**Step 5: Append 1, to every pair.**

**>>> docWordAppenOne = docWordRddFiltered.map(lambda x:(x,1))**

**>>> docWordAppenOne.collect()**

[((u'doc6', u'crazy'), 1), ((u'doc6', u'fox'), 1), ((u'doc6', u'jumped'), 1), ((u'doc6', u'fox'), 1), ((u'doc6', u'ran'), 1), ((u'doc6', u'fox'), 1), ((u'doc6', u'ran'), 1), ((u'doc6', u'fast'), 1), ((u'doc6', u'over'), 1), ((u'doc6', u'fence'), 1), ((u'doc6', u'too'), 1), ((u'doc6', u'high'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'jumped'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'jumped'), 1), ((u'doc1', u'over'), 1), ((u'doc1', u'the'), 1), ((u'doc1', u'fence'), 1), ((u'doc1', u'crazy'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'jumped'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'ran'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'ran'), 1), ((u'doc1', u'fast'), 1), ((u'doc1', u'over'), 1), ((u'doc1', u'fence'), 1), ((u'doc1', u'too'), 1), ((u'doc1', u'high'), 1), ((u'doc2', u'crazy'), 1), ((u'doc2', u'fox'), 1), ((u'doc2', u'jumped'), 1), ((u'doc2', u'crazy'), 1), ((u'doc2', u'fox'), 1), ((u'doc2', u'jumped'), 1), ((u'doc2', u'over'), 1), ((u'doc2', u'the'), 1), ((u'doc2', u'fence'), 1), ((u'doc2', u'again'), 1), ((u'doc3', u'fox'), 1), ((u'doc3', u'ran'), 1), ((u'doc3', u'fast'), 1), ((u'doc3', u'over'), 1), ((u'doc3', u'fence'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'high'), 1), ((u'doc1', u'sugar'), 1), ((u'doc2', u'crazy'), 1), ((u'doc2', u'fox'), 1), ((u'doc2', u'ran'), 1), ((u'doc2', u'ran'), 1), ((u'doc2', u'ran'), 1), ((u'doc2', u'fast'), 1), ((u'doc3', u'crazy'), 1), ((u'doc3', u'fox'), 1), ((u'doc3', u'jumped'), 1), ((u'doc3', u'jumped'), 1), ((u'doc3', u'jumped'), 1), ((u'doc3', u'jumped'), 1), ((u'doc4', u'crazy'), 1), ((u'doc4', u'fox'), 1), ((u'doc4', u'ran'), 1), ((u'doc4', u'jumped'), 1), ((u'doc4', u'ran'), 1), ((u'doc4', u'jumped'), 1), ((u'doc4', u'crazy'), 1), ((u'doc4', u'fox'), 1), ((u'doc4', u'jumped'), 1), ((u'doc4', u'jumped'), 1), ((u'doc4', u'crazy'), 1), ((u'doc4', u'fox'), 1), ((u'doc4', u'jumped'), 1), ((u'doc4', u'jumped'), 1), ((u'doc5', u'crazy'), 1), ((u'doc5', u'fox'), 1), ((u'doc5', u'jumped'), 1), ((u'doc5', u'over'), 1), ((u'doc5', u'fence'), 1), ((u'doc5', u'very'), 1), ((u'doc5', u'high'), 1), ((u'doc5', u'crazy'), 1), ((u'doc5', u'fox'), 1), ((u'doc5', u'jumped'), 1), ((u'doc5', u'over'), 1), ((u'doc5', u'the'), 1), ((u'doc5', u'fence'), 1), ((u'doc5', u'again'), 1), ((u'doc6', u'book'), 1), ((u'doc6', u'reading'), 1), ((u'doc6', u'about'), 1), ((u'doc6', u'fox'), 1), ((u'doc6', u'and'), 1), ((u'doc6', u'fence'), 1), ((u'doc1', u'crazy'), 1), ((u'doc1', u'fox'), 1), ((u'doc1', u'jumped'), 1)]

**Step 6: Reduce stage (Reduce all key value pairs)**

**>>> reducedRdd = docWordAppenOne.reduceByKey(lambda x,y : x+y)**

**>>> reducedRdd.collect()**

[((u'doc1', u'high'), 2), ((u'doc5', u'fox'), 2), ((u'doc2', u'fox'), 3), ((u'doc1', u'crazy'), 2), ((u'doc3', u'jumped'), 4), ((u'doc6', u'book'), 1), ((u'doc6', u'fence'), 2), ((u'doc6', u'about'), 1), ((u'doc6', u'high'), 1), ((u'doc1', u'sugar'), 1), ((u'doc3', u'fox'), 2), ((u'doc6', u'crazy'), 1), ((u'doc1', u'over'), 2), ((u'doc6', u'reading'), 1), ((u'doc6', u'fox'), 4), ((u'doc2', u'over'), 1), ((u'doc2', u'jumped'), 2), ((u'doc1', u'the'), 1), ((u'doc1', u'fast'), 1), ((u'doc6', u'over'), 1), ((u'doc6', u'jumped'), 1), ((u'doc2', u'the'), 1), ((u'doc5', u'the'), 1), ((u'doc2', u'fast'), 1), ((u'doc2', u'crazy'), 3), ((u'doc2', u'again'), 1), ((u'doc5', u'over'), 2), ((u'doc3', u'over'), 1), ((u'doc4', u'fox'), 3), ((u'doc5', u'very'), 1), ((u'doc5', u'jumped'), 2), ((u'doc4', u'crazy'), 3), ((u'doc3', u'fence'), 1), ((u'doc2', u'fence'), 1), ((u'doc6', u'too'), 1), ((u'doc3', u'fast'), 1), ((u'doc6', u'fast'), 1), ((u'doc5', u'crazy'), 2), ((u'doc6', u'and'), 1), ((u'doc1', u'ran'), 2), ((u'doc1', u'fox'), 7), ((u'doc1', u'jumped'), 4), ((u'doc5', u'again'), 1), ((u'doc3', u'crazy'), 1), ((u'doc5', u'fence'), 2), ((u'doc2', u'ran'), 3), ((u'doc1', u'fence'), 2), ((u'doc1', u'too'), 1), ((u'doc4', u'jumped'), 6), ((u'doc5', u'high'), 1), ((u'doc4', u'ran'), 2), ((u'doc3', u'ran'), 1), ((u'doc6', u'ran'), 2)]

**Step 7: sortBy( function sorts the RDD by given keyFunction. In this case keyFunction is lambda x: x[1]. And ‘False’ is the input to a function to sort it descending order.)**

**>>> sortedRdd = reducedRdd.sortBy((lambda x:x[1]),False)**

**>>> sortedRdd.collect()**

[((u'doc1', u'fox'), 7), ((u'doc4', u'jumped'), 6), ((u'doc3', u'jumped'), 4), ((u'doc6', u'fox'), 4), ((u'doc1', u'jumped'), 4), ((u'doc2', u'fox'), 3), ((u'doc2', u'crazy'), 3), ((u'doc4', u'fox'), 3), ((u'doc4', u'crazy'), 3), ((u'doc2', u'ran'), 3), ((u'doc1', u'high'), 2), ((u'doc5', u'fox'), 2), ((u'doc1', u'crazy'), 2), ((u'doc6', u'fence'), 2), ((u'doc3', u'fox'), 2), ((u'doc1', u'over'), 2), ((u'doc2', u'jumped'), 2), ((u'doc5', u'over'), 2), ((u'doc5', u'jumped'), 2), ((u'doc5', u'crazy'), 2), ((u'doc1', u'ran'), 2), ((u'doc5', u'fence'), 2), ((u'doc1', u'fence'), 2), ((u'doc4', u'ran'), 2), ((u'doc6', u'ran'), 2), ((u'doc6', u'book'), 1), ((u'doc6', u'about'), 1), ((u'doc6', u'high'), 1), ((u'doc1', u'sugar'), 1), ((u'doc6', u'crazy'), 1), ((u'doc6', u'reading'), 1), ((u'doc2', u'over'), 1), ((u'doc1', u'the'), 1), ((u'doc1', u'fast'), 1), ((u'doc6', u'over'), 1), ((u'doc6', u'jumped'), 1), ((u'doc2', u'the'), 1), ((u'doc5', u'the'), 1), ((u'doc2', u'fast'), 1), ((u'doc2', u'again'), 1), ((u'doc3', u'over'), 1), ((u'doc5', u'very'), 1), ((u'doc3', u'fence'), 1), ((u'doc2', u'fence'), 1), ((u'doc6', u'too'), 1), ((u'doc3', u'fast'), 1), ((u'doc6', u'fast'), 1), ((u'doc6', u'and'), 1), ((u'doc5', u'again'), 1), ((u'doc3', u'crazy'), 1), ((u'doc1', u'too'), 1), ((u'doc5', u'high'), 1), ((u'doc3', u'ran'), 1)]

**Step 8: Swap the Key Value pairs**

**>>> swappedRdd = sortedRdd.map(lambda (x,y): ( (x[1]), (y,x[0]) ) )**

**>>> swappedRdd.collect()**

[(u'fox', (7, u'doc1')), (u'jumped', (6, u'doc4')), (u'jumped', (4, u'doc3')), (u'fox', (4, u'doc6')), (u'jumped', (4, u'doc1')), (u'fox', (3, u'doc2')), (u'crazy', (3, u'doc2')), (u'fox', (3, u'doc4')), (u'crazy', (3, u'doc4')), (u'ran', (3, u'doc2')), (u'high', (2, u'doc1')), (u'fox', (2, u'doc5')), (u'crazy', (2, u'doc1')), (u'fence', (2, u'doc6')), (u'fox', (2, u'doc3')), (u'over', (2, u'doc1')), (u'jumped', (2, u'doc2')), (u'over', (2, u'doc5')), (u'jumped', (2, u'doc5')), (u'crazy', (2, u'doc5')), (u'ran', (2, u'doc1')), (u'fence', (2, u'doc5')), (u'fence', (2, u'doc1')), (u'ran', (2, u'doc4')), (u'ran', (2, u'doc6')), (u'book', (1, u'doc6')), (u'about', (1, u'doc6')), (u'high', (1, u'doc6')), (u'sugar', (1, u'doc1')), (u'crazy', (1, u'doc6')), (u'reading', (1, u'doc6')), (u'over', (1, u'doc2')), (u'the', (1, u'doc1')), (u'fast', (1, u'doc1')), (u'over', (1, u'doc6')), (u'jumped', (1, u'doc6')), (u'the', (1, u'doc2')), (u'the', (1, u'doc5')), (u'fast', (1, u'doc2')), (u'again', (1, u'doc2')), (u'over', (1, u'doc3')), (u'very', (1, u'doc5')), (u'fence', (1, u'doc3')), (u'fence', (1, u'doc2')), (u'too', (1, u'doc6')), (u'fast', (1, u'doc3')), (u'fast', (1, u'doc6')), (u'and', (1, u'doc6')), (u'again', (1, u'doc5')), (u'crazy', (1, u'doc3')), (u'too', (1, u'doc1')), (u'high', (1, u'doc5')), (u'ran', (1, u'doc3'))]

>>>

**Step 9: groupByKey() groups the records and creates an iterable of values**

**>>> groupedRdd = swappedRdd.groupByKey()**

**>>> groupedRdd.collect()**

[(u'and', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77810>), (u'again', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77510>), (u'crazy', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77710>), (u'fence', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77690>), (u'ran', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77c10>), (u'jumped', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77790>), (u'over', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77610>), (u'fox', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b776d0>), (u'about', <pyspark.resultiterable.ResultIterable object at 0x7f9ff3b77650>), (u'fast', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d2610>), (u'sugar', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d20d0>), (u'high', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d22d0>), (u'very', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d2110>), (u'book', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d23d0>), (u'too', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d2190>), (u'the', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d2250>), (u'reading', <pyspark.resultiterable.ResultIterable object at 0x7f9ff40d2350>)]

**Step 10: x[1].data displays the iterable list.**

**>>> finalRdd = groupedRdd.map(lambda x:(x[0],x[1].data))**

**>>> finalRdd.collect()**

[(u'and', [(1, u'doc6')]),

(u'again', [(1, u'doc2'), (1, u'doc5')]),

(u'crazy', [(3, u'doc2'), (3, u'doc4'), (2, u'doc1'), (2, u'doc5'), (1, u'doc6'), (1, u'doc3')]),

(u'fence', [(2, u'doc6'), (2, u'doc5'), (2, u'doc1'), (1, u'doc3'), (1, u'doc2')]),

(u'ran', [(3, u'doc2'), (2, u'doc1'), (2, u'doc4'), (2, u'doc6'), (1, u'doc3')]),

(u'jumped', [(6, u'doc4'), (4, u'doc3'), (4, u'doc1'), (2, u'doc2'), (2, u'doc5'), (1, u'doc6')]),

(u'over', [(2, u'doc1'), (2, u'doc5'), (1, u'doc2'), (1, u'doc6'), (1, u'doc3')]),

(u'fox', [(7, u'doc1'), (4, u'doc6'), (3, u'doc2'), (3, u'doc4'), (2, u'doc5'), (2, u'doc3')]),

(u'about', [(1, u'doc6')]), (u'fast', [(1, u'doc1'), (1, u'doc2'), (1, u'doc3'), (1, u'doc6')]),

(u'sugar', [(1, u'doc1')]), (u'high', [(2, u'doc1'), (1, u'doc6'), (1, u'doc5')]), (u'very', [(1, u'doc5')]),

(u'book', [(1, u'doc6')]), (u'too', [(1, u'doc6'), (1, u'doc1')]),

(u'the', [(1, u'doc1'), (1, u'doc2'), (1, u'doc5')]),

(u'reading', [(1, u'doc6')])]

>>>