Big Data Analysis Lab-03

Name: Gurvinder Kaur Matharu

PRN: 20190802077

```
pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.7/dist-packages (3
Requirement already satisfied: py4j==0.10.9.3 in /usr/local/lib/python3.7/dist-pack
```

```
from pyspark import SparkContext, SparkConf
sc = SparkContext()
```

```
data = sc.parallelize([1,2,3,4,5])
data.take(3)
```

```
[1, 2, 3]
```

```
data.reduce(lambda x, y : x + y)
```

```
15
```

```
save_rdd = sc.parallelize([1,2,3,4,5,6])
save_rdd.saveAsTextFile('file.txt')
```

```
filter_rdd = sc.parallelize([2, 3, 4, 5, 6, 7])
filter_rdd.filter(lambda x: x%2 == 0).collect()
```

```
[2, 4, 6]
```

```
data = sc.parallelize([(1,2),(3,4),(3,6),(3,4)])
data.collect()
```

```
[(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
type(data)
```

```
pyspark.rdd.RDD
```

```
data.count()
```

```
4
```

```
data.countByValue()
```

```
    defaultdict(int, {(1, 2): 1, (3, 4): 2, (3, 6): 1})
```

```
dataStr = sc.parallelize([(1,'mike'),(2,'john'),(3,'rambo'),(4,'bill')])
dataStr.collect()
```

```
    [(1, 'mike'), (2, 'john'), (3, 'rambo'), (4, 'bill')]
```

```
dataStr.count()
```

```
    4
```

```
dataStr.countByValue()
```

```
    defaultdict(int,
                {(1, 'mike'): 1, (2, 'john'): 1, (3, 'rambo'): 1, (4, 'bill'): 1})
```

```
data.top(2)
```

```
    [(3, 6), (3, 4)]
```

```
data.collect()
```

```
    [(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
data.sortByKey().collect()
```

```
    [(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
# lookup : Return all value associated with the given key.
data.lookup(3)
```

```
    [4, 6, 4]
```

```
data.keys().collect()
```

```
    [1, 3, 3, 3]
```

```
data.values().collect()
```

```
    [2, 4, 6, 4]
```

```
data.mapValues(lambda a : a*a).collect()
```

```
    [(1, 4), (3, 16), (3, 36), (3, 16)]
```

```
data.collect()
```

```
    [(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
data.reduceByKey(lambda x, y : x+y).collect()
```

```
    [(1, 2), (3, 14)]
```

```
data.reduceByKey(lambda x, y : x+y).collect()
```

```
    [(1, 2), (3, 14)]
```

```
# groupBy: This transformation groups all the rows with the same key into a single row.
result = data.groupByKey().collect()
```

```
result
```

```
    [(1, <pyspark.resultiterable.ResultIterable at 0x7fb984917c50>),
     (3, <pyspark.resultiterable.ResultIterable at 0x7fb984917ed0>)]
```

```
for (k,v) in result:
    print(k, list(v))
```

```
    1 [2]
    3 [4, 6, 4]
```

```
aa = data.groupByKey().mapValues(sum)
```

```
aa.collect()
```

```
    [(1, 2), (3, 14)]
```

```
bb = data.groupByKey().mapValues(max)
```

```
bb.collect()
```

```
    [(1, 2), (3, 6)]
```

```
data.collect()
```

```
    [(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
data.flatMapValues(lambda x: range(1, x)).collect()
```

```
    [(1, 1),
     (3, 1),
     (3, 2),
     (3, 3),
     (3, 1),
     (3, 2),
     (3, 3),
     (3, 4),
     (3, 5),
     (3, 1),
     (3, 2),
     (3, 3)]
```

```
data.collect()
```

```
    [(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
data2 = sc.parallelize([(3,9)])
data2.collect()
```

```
    [(3, 9)]
```

```
data.subtractByKey(data2).collect()
```

```
    [(1, 2)]
```

```
data2.subtractByKey(data).collect()
```

```
    []
```

```
data.collect()
```

```
    [(1, 2), (3, 4), (3, 6), (3, 4)]
```

```
data2 = sc.parallelize([(3,9),(4,15)])
data2.collect()
```

```
    [(3, 9), (4, 15)]
```

```
data.join(data2).collect()
```

```
    [(3, (4, 9)), (3, (6, 9)), (3, (4, 9))]
```

```
data2.join(data).collect()
```

```
    [(3, (9, 4)), (3, (9, 6)), (3, (9, 4))]
```

```
# rightOuterJoin: Perform a join between two RDDs where key must be present
# in the first RDD.
data.rightOuterJoin(data2).collect()
```

```
    [(4, (None, 15)), (3, (4, 9)), (3, (6, 9)), (3, (4, 9))]
```

```
data2.rightOuterJoin(data).collect()
```

```
    [(1, (None, 2)), (3, (9, 4)), (3, (9, 6)), (3, (9, 4))]
```

```
# leftOuterJoin: Perform a join between two RDDs where key must
# be present in the OTHER RDD.
data.leftOuterJoin(data2).collect()
```

```
[(1, (2, None)), (3, (4, 9)), (3, (6, 9)), (3, (4, 9))]
```

```
data2.leftOuterJoin(data).collect()
```

```
[(4, (15, None)), (3, (9, 4)), (3, (9, 6)), (3, (9, 4))]
```

```
sc.stop()
```

✓ 0s completed at 8:10 PM