ASSIGNMENT SPARK-1

PROBLEM STATEMENT:

A) Write a program to read a text file and print the number of rows of data in the document.

EXPLANATION: First we create a text file in which we save some data and save it in local directory. So the created text file is named as "Textfile.txt". Then we open the spark shell using below command.

CODE: spark-shell

EXPLANATION: Now we create a RDD named "test" which will read the text file from the **path:"file:///home/acadgild/Textfile.txt"** and then we use "flatMap" for splitting the data key and value pairs. Then we used "filter" function to get the values which are entered and will neglect the space characters after the last string. Both the result will be saved in variable "Row". We can see the content using command "collect()". Now we use "count()" function to get the number of rows.

CODE:

```
val test = sc.textFile("file:///home/acadgild/Textfile.txt")

val Rows = test.flatMap(x => x.split(",")).filter(x =>(x!=""))

Rows.collect()

Rows.count()
```

SOLUTION REPORT:

```
scala> val test = sc.textFile("file:///home/acadgild/Textfile.txt")
test: org.apache.spark.rdd.RDD[String] =
file:///home/acadgild/Textfile.txt MapPartitionsRDD[51] at textFile at
<console>:24

scala> val Rows = test.flatMap(x => x.split(",")).filter(x =>(x!=""))
Rows: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[53] at filter at
<console>:26

scala> Rows.collect()
res65: Array[String] = Array("how are you shravan ", "Jai Pubg ", "how are
you amit ", how are you yogi)

scala> Rows.count()
res64: Long = 4
```

```
scala> val test = sc.textFile("file:///home/acadgild/Textfile.txt")
test: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Textfile.txt MapP
artitionsRDD[51] at textFile at <console>:24

scala> val Rows = test.flatMap(x => x.split(",")).filter(x =>(x!=""))
Rows: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[53] at filter at <cons
ole>:26

scala> Rows.count()
res64: Long = 4

scala> Rows.collect()
res65: Array[String] = Array("how are you shravan ", "Jai Pubg ", "how are you a
mit ", how are you yogi)
```

B) Write a program to read a text file and print the number of words in the document.

EXPLANATION: Used the similar steps as used in above code. Then here used a additional transformation function i.e. "map" to separate each words mapped with value 1.

CODE:

```
val words = test.flatMap(x => x.split(" ")).filter(x =>(x!="")).map(y =>
(y,1))
```

SOLUTION REPORT:

```
scala> val words = test.flatMap(x => x.split(" ")).filter(x
=>(x!="")).map(y => (y,1))
words: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[59] at
map at <console>:26

scala> words.collect()
res67: Array[(String, Int)] = Array((how,1), (are,1), (you,1),
(shravan,1), (Jai,1), (Pubg,1), (how,1), (are,1), (you,1), (amit,1),
(how,1), (are,1), (you,1), (yogi,1))

scala> words.count()
res68: Long = 14
```

```
scala> val words = test.flatMap(x => x.split(" ")).filter(x =>(x!="")).map(y =>
(y,1))
words: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[59] at map at
<console>:26

scala> words.collect()
res67: Array[(String, Int)] = Array((how,1), (are,1), (you,1), (shravan,1), (Jai
,1), (Pubg,1), (how,1), (are,1), (you,1), (amit,1), (how,1), (are,1), (you,1), (
yogi,1))

scala> words.count()
res68: Long = 14
```

PROBLEM STATEMENT 1:

TASK 1.1 :1. Read the text file, and create a tupled rdd.

EXPLANATION: Below is the code we use to read the text file using spark context and creating a tuple RDD.
-> Created a tuple RDD with name as Key and the subject, grades and the marks as values.

CODE:

```
val testRDD = sc.textFile("/home/acadgild/hadoop/Dataset.txt").map(x =>
(x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt
,x.split(",")(4).toInt)))

testRDD.foreach(println)
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x =>
(x.split(",")(0),x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt,
x.split(",")(4).toInt))
testRDD: org.apache.spark.rdd.RDD[(String, String, String, Int, Int)] =
MapPartitionsRDD[124] at map at <console>:24

scala> testRDD.foreach(println)
(Mathew,science,grade-3,45,12)
(Mathew,history,grade-2,55,13)
(Mark,maths,grade-2,23,13)
(Mark,science,grade-1,76,13)
(John,history,grade-1,14,12)
(John,maths,grade-2,74,13)
(Lisa,science,grade-1,24,12)
(Lisa,history,grade-3,86,13)
(Andrew,maths,grade-1,34,13)
(Andrew,science,grade-3,26,14)
(Andrew,history,grade-1,74,12)
(Mathew,science,grade-2,55,12)
(Mathew,history,grade-2,87,12)
(Mark,maths,grade-1,92,13)
(Mark,science,grade-2,12,12)
(John,history,grade-1,67,13)
(John,maths,grade-1,35,11)
(Lisa,science,grade-2,24,13)
(Lisa,history,grade-2,98,15)
(Andrew,maths,grade-1,23,16)
(Andrew,science,grade-3,44,14)
(Andrew,history,grade-2,77,11)
```

Task1.2 - Find the count of total number of rows present

EXPLANATION: By using count() function we can see the number of lines present

in the text file.

CODE: testRDD.count()

SOLUTION REPORT:

```
scala> testRDD.count()
res106: Long = 22
```

TASK 1.3: What is the distinct number of subjects present in the entire school

EXPLANATION: First creating a RDD to read the file and selecting only subject name and mapping them with value 1 and counting the values of occurrences using reduceByKey to get distinct number of subjects.

CODE:

```
val testRDD = sc.textFile("/home/acadgild/hadoop/dataset.txt").map(x=>
(x.split(",")(1),1))

val reduceRDD = testRDD.reduceByKey((x,y)=>(x+y))
```

SOLUTION REPORT:

```
scala> RDDreduce.foreach(println)
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x =>
(x.split(",")(1),1))
testRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[127]
at map at <console>:24

scala> val reduceRDD = testRDD.reduceByKey((x,y)=>(x+y))
reduceRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[128] at
reduceByKey at <console>:26

scala> reduceRDD.collect()
res107: Array[(String, Int)] = Array((maths,6), (history,8), (science,8))

scala> reduceRDD.foreach(println)
(maths,6)
(history,8)
(science,8)
```

```
scala> val reduceRDD = testRDD.reduceByKey((x,y)=>(x+y))
reduceRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[128] at reduceByKey at <c
onsole>:26

scala> reduceRDD.collect()
res107: Array[(String, Int)] = Array((maths,6), (history,8), (science,8))

scala> reduceRDD.foreach(println)
(maths,6)
(history,8)
(science,8)
```

TASK 1.4: What is the count of the number of students in the school, whose name is Mathew and Marks is 55

EXPLANATION:

In the first line code, we are reading the text file and creating a tuple RDD as "testRDD" with name & marks as key and mapping numerical 1 as value. Filter the tuple RDD by providing the condition Mather and mark as 55. Then counting each occurrences using the reduceByKey and the required output is below.

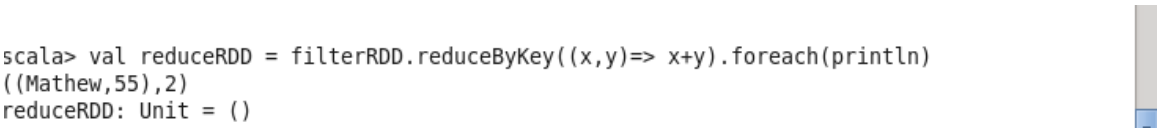
CODE:

```
val testRDD = sc.textFile("/home/acadgild/dataset.txt").map(x =>
((x.split(",")(0),x.split(",")(3).toInt),1))

val filterRDD = testRDD.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)

val reduceRDD = filterRDD.reduceByKey((x,y)=> x+y).foreach(println)
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(3).toInt),1))
testRDD: org.apache.spark.rdd.RDD[((String, Int), Int)] =
MapPartitionsRDD[131] at map at <console>:24

scala> val filterRDD = testRDD.filter(x => x._1._1 == "Mathew" && x._1._2
== 55)
filterRDD: org.apache.spark.rdd.RDD[((String, Int), Int)] =
MapPartitionsRDD[132] at filter at <console>:26

scala> testRDD.foreach(println)
((Mathew,45),1)
((Mathew,55),1)
((Mark,23),1)
((Mark,76),1)
((John,14),1)
((John,74),1)
((Lisa,24),1)
((Lisa,86),1)
((Andrew,34),1)
((Andrew,26),1)
((Andrew,74),1)
```

```
((Mathew,55),1)
((Mathew,87),1)
((Mark,92),1)
((Mark,12),1)
((John,67),1)
((John,35),1)
((Lisa,24),1)
((Lisa,98),1)
((Andrew,23),1)
((Andrew,44),1)
((Andrew,77),1)

scala> filterRDD.foreach(println)
((Mathew,55),1)
((Mathew,55),1)

scala> val reduceRDD = filterRDD.reduceByKey((x,y)=>
x+y).foreach(println)
((Mathew,55),2)
reduceRDD: Unit = ()
```



PROBLEM STATEMENT 2:

TASK 2.1: What is the count of students per grade in the school?

EXPLANATION: We are reading the text file by creating a tuple RDD with grade as key and mapping numerical 1 as values and reducing the number occurrences using reduceByKey and then by using foreach command we get the required output.

CODE:

```
val testRDD = sc.textFile("/home/acadgild/hadoop/dataset.txt").map(x =>
(x.split(",")(2),1)).reduceByKey((x,y)=>x+y).foreach(println)
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>(x.split(",")(2),1)).reduceByKey((x,y) => x+y).foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)
testRDD: Unit = ()
```

```
scala> val testRDD = sc.textFile("file:///home/acadgild/dataset.txt").map(x =>(x.split(","
)(2),1)).reduceByKey((x,y) => x+y).foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)
testRDD: Unit = ()
```

TASK 2.2: Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)

EXPLANATION: First we are creating the testRDD to read the file and selecting name and grade as key and marks as value. We are using reduceByKey to add the occurrences of marks for each key which is student name and grade. Then calculating average by summing the marks and dividing by its count for each key. Below
screenshot shows the final result.

CODE:

```
val testRDD = sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val newRDD = testRDD.mapValues(x => (x,1))

val reduceRDD = newRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val AvgStudent = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
testRDD: org.apache.spark.rdd.RDD[((String, String), Int)] =
MapPartitionsRDD[149] at map at <console>:24

scala> val newRDD = testRDD.mapValues(x => (x,1))
newRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
MapPartitionsRDD[150] at mapValues at <console>:26

scala> newRDD.foreach(println)
((Mathew,grade-3),(45,1))
((Mathew,grade-2),(55,1))
((Mark,grade-2),(23,1))
((Mark,grade-1),(76,1))
((John,grade-1),(14,1))
((John,grade-2),(74,1))
((Lisa,grade-1),(24,1))
((Lisa,grade-3),(86,1))
((Andrew,grade-1),(34,1))
((Andrew,grade-3),(26,1))
((Andrew,grade-1),(74,1))
((Mathew,grade-2),(55,1))
((Mathew,grade-2),(87,1))
((Mark,grade-1),(92,1))
```

```
((Mark,grade-2),(12,1))
((John,grade-1),(67,1))
((John,grade-1),(35,1))
((Lisa,grade-2),(24,1))
((Lisa,grade-2),(98,1))
((Andrew,grade-1),(23,1))
((Andrew,grade-3),(44,1))
((Andrew,grade-2),(77,1))

scala> val reduceRDD = newRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 +
y._2))
reduceRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
ShuffledRDD[151] at reduceByKey at <console>:28

scala> reduceRDD.foreach(println)
((Lisa,grade-1),(24,1))
((Mark,grade-2),(35,2))
((Lisa,grade-2),(122,2))
((Mathew,grade-3),(45,1))
((Andrew,grade-2),(77,1))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((John,grade-1),(116,3))
((John,grade-2),(74,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
((Mathew,grade-2),(197,3))

scala> val AvgStudent =
reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgStudent: org.apache.spark.rdd.RDD[((String, String), Double)] =
MapPartitionsRDD[152] at mapValues at <console>:30

scala> AvgStudent.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
```

```
scala> val AvgStudent = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgStudent: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[152] a
t mapValues at <console>:30

scala> AvgStudent.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
```

TASK 2.3: What is the average score of students in each subject across all grades?

EXPLANATION: We are first creating testRDD to read the text file and we are extracting name and subject as key and marks as value. Now using mapValues we are mapping each value with 1. Then we are adding the marks and number of occurrences for each key using reducebyKey and calculating average by dividing the sum of marks and count of occurrences for each key.

CODE:

```
val testRDD = sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))

val newRDD = testRDD.mapValues(x => (x,1))

val reduceRDD = newRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val AvgSubj = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
testRDD: org.apache.spark.rdd.RDD[((String, String), Int)] =
MapPartitionsRDD[155] at map at <console>:24

scala> val newRDD = testRDD.mapValues(x => (x,1))
newRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
MapPartitionsRDD[156] at mapValues at <console>:26

scala> newRDD.foreach(println)
((Mathew,science),(45,1))
((Mathew,history),(55,1))
((Mark,maths),(23,1))
((Mark,science),(76,1))
((John,history),(14,1))
((John,maths),(74,1))
```

```
((Lisa,science),(24,1))
((Lisa,history),(86,1))
((Andrew,maths),(34,1))
((Andrew,science),(26,1))
((Andrew,history),(74,1))
((Mathew,science),(55,1))
((Mathew,history),(87,1))
((Mark,maths),(92,1))
((Mark,science),(12,1))
((John,history),(67,1))
((John,maths),(35,1))
((Lisa,science),(24,1))
((Lisa,history),(98,1))
((Andrew,maths),(23,1))
((Andrew,science),(44,1))
((Andrew,history),(77,1))

scala> val reduceRDD = newRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 +
y._2))
reduceRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
ShuffledRDD[157] at reduceByKey at <console>:28

scala> reduceRDD.foreach(println)
((Lisa,history),(184,2))
((Mark,maths),(115,2))
((Andrew,science),(70,2))
((Mark,science),(88,2))
((Mathew,science),(100,2))
((Andrew,maths),(57,2))
((Mathew,history),(142,2))
((John,maths),(109,2))
((John,history),(81,2))
((Lisa,science),(48,2))
((Andrew,history),(151,2))

scala> val AvgSubj =
reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgSubj: org.apache.spark.rdd.RDD[((String, String), Double)] =
MapPartitionsRDD[158] at mapValues at <console>:30

scala> AvgSubj.foreach(println)
((Lisa,history),92.0)
((Mark,maths),57.5)
((Andrew,science),35.0)
((Mark,science),44.0)
((Mathew,science),50.0)
((Andrew,maths),28.5)
((Mathew,history),71.0)
((John,maths),54.5)
((John,history),40.5)
((Lisa,science),24.0)
((Andrew,history),75.5)
```

```
scala> val AvgSubj = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgSubj: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[158] at m
apValues at <console>:30

scala> AvgSubj.foreach(println)
((Lisa,history),92.0)
((Mark,maths),57.5)
((Andrew,science),35.0)
((Mark,science),44.0)
((Mathew,science),50.0)
((Andrew,maths),28.5)
((Mathew,history),71.0)
((John,maths),54.5)
((John,history),40.5)
((Lisa,science),24.0)
((Andrew,history),75.5)
```

TASK 2.4: What is the average score of students in each subject per grade?

EXPLANATION: In first step we are creating paired RDD named as testRDD to read the file and extracting subject and grade as key and marks as value. Then we are mapping the values of testRDD with numerical value 1 using function mapValue. We are adding marks and number of occurrences for each key using reduceByKey
and then calculating average by dividing the sum of marks with number of occurrences and the requested output is shown below.

CODE:

```
val testRDD = sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))

val newRDD = testRDD.mapValues(x => (x,1))

val reduceRDD = newRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val AvgGrade = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))
testRDD: org.apache.spark.rdd.RDD[((String, String), Int)] =
MapPartitionsRDD[161] at map at <console>:24

scala> val newRDD = testRDD.mapValues(x => (x,1))
newRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
MapPartitionsRDD[162] at mapValues at <console>:26

scala> newRDD.foreach(println)
((science,grade-3),(45,1))
((history,grade-2),(55,1))
((maths,grade-2),(23,1))
((science,grade-1),(76,1))
((history,grade-1),(14,1))
```

```
((maths,grade-2),(74,1))
((science,grade-1),(24,1))
((history,grade-3),(86,1))
((maths,grade-1),(34,1))
((science,grade-3),(26,1))
((history,grade-1),(74,1))
((science,grade-2),(55,1))
((history,grade-2),(87,1))
((maths,grade-1),(92,1))
((science,grade-2),(12,1))
((history,grade-1),(67,1))
((maths,grade-1),(35,1))
((science,grade-2),(24,1))
((history,grade-2),(98,1))
((maths,grade-1),(23,1))
((science,grade-3),(44,1))
((history,grade-2),(77,1))

scala> val reduceRDD = newRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 +
y._2))
reduceRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
ShuffledRDD[163] at reduceByKey at <console>:28

scala> reduceRDD.foreach(println)
((history,grade-2),(317,4))
((history,grade-3),(86,1))
((maths,grade-1),(184,4))
((science,grade-3),(115,3))
((science,grade-1),(100,2))
((science,grade-2),(91,3))
((history,grade-1),(155,3))
((maths,grade-2),(97,2))

scala> val AvgGrade =
reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgGrade: org.apache.spark.rdd.RDD[((String, String), Double)] =
MapPartitionsRDD[164] at mapValues at <console>:30

scala> AvgGrade.foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),38.333333333333336)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
```

```
scala> val AvgGrade = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgGrade: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[164] at
mapValues at <console>:30

scala> AvgGrade.foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),38.333333333333336)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
```

TASK 2.5: For all students in grade-2, how many have average score greater than 50?

EXPLANATIION: Creating a paired RDD named as baseRDD to read the file and extracting name and grade as key and marks as value. Now we are mapping each value of testRDD with 1. We are adding the marks of subject and number of occurrences per key using reduceByKey function. Then calculating the average of each student. Now in next step we are filtering the above result with student belonging to grade-2 and having marks greater than 50. Then using "count()" we get number of the counts and the data. Then using "foreach(println)" in place of "count()" in the filterMapRDD to get the output with student belonging to grade-2 and having marks greater than 50.

CODE:

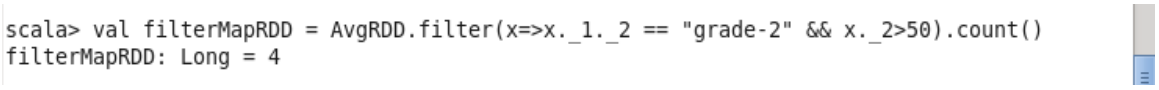
```
val testRDD = sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val mapRDD = testRDD.mapValues(x => (x,1))

val reduceRDD = mapRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val AvgRDD = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}

val filterMapRDD = AvgRDD.filter(x=>x._1._2 == "grade-2" &&
x._2>50).count()

val filterMapRDD = AvgRDD.filter(x=>x._1._2 == "grade-2" &&
x._2>50).foreach(println)
```

SOLUTION REPORT:

```
scala> val testRDD =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
testRDD: org.apache.spark.rdd.RDD[((String, String), Int)] =
MapPartitionsRDD[2] at map at <console>:24

scala> testRDD.foreach(println)
((Mathew,grade-3),45)
((Mathew,grade-2),55)
((Mark,grade-2),23)
```

```
((Mark,grade-1),76)
((John,grade-1),14)
((John,grade-2),74)
((Lisa,grade-1),24)
((Lisa,grade-3),86)
((Andrew,grade-1),34)
((Andrew,grade-3),26)
((Andrew,grade-1),74)
((Mathew,grade-2),55)
((Mathew,grade-2),87)
((Mark,grade-1),92)
((Mark,grade-2),12)
((John,grade-1),67)
((John,grade-1),35)
((Lisa,grade-2),24)
((Lisa,grade-2),98)
((Andrew,grade-1),23)
((Andrew,grade-3),44)
((Andrew,grade-2),77)

scala> val mapRDD = testRDD.mapValues(x=>(x,1))
mapRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
MapPartitionsRDD[3] at mapValues at <console>:26

scala> mapRDD.foreach(println)
((Mathew,grade-3),(45,1))
((Mathew,grade-2),(55,1))
((Mark,grade-2),(23,1))
((Mark,grade-1),(76,1))
((John,grade-1),(14,1))
((John,grade-2),(74,1))
((Lisa,grade-1),(24,1))
((Lisa,grade-3),(86,1))
((Andrew,grade-1),(34,1))
((Andrew,grade-3),(26,1))
((Andrew,grade-1),(74,1))
((Mathew,grade-2),(55,1))
((Mathew,grade-2),(87,1))
((Mark,grade-1),(92,1))
((Mark,grade-2),(12,1))
((John,grade-1),(67,1))
((John,grade-1),(35,1))
((Lisa,grade-2),(24,1))
((Lisa,grade-2),(98,1))
((Andrew,grade-1),(23,1))
((Andrew,grade-3),(44,1))
((Andrew,grade-2),(77,1))

scala> val reduceRDD = mapRDD.reduceByKey((x,y) =>(x._1 + y._1, x._2 +
y._2))
reduceRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
ShuffledRDD[4] at reduceByKey at <console>:28
```

```
scala> reduceRDD.foreach(println)
((Lisa,grade-1),(24,1))
((Mark,grade-2),(35,2))
((Lisa,grade-2),(122,2))
((Mathew,grade-3),(45,1))
((Andrew,grade-2),(77,1))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((John,grade-1),(116,3))
((John,grade-2),(74,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
((Mathew,grade-2),(197,3))

scala> val AvgRDD = reduceRDD.mapValues{case(sum,count)=>(1.0*sum)/count}
AvgRDD: org.apache.spark.rdd.RDD[((String, String), Double)] =
MapPartitionsRDD[5] at mapValues at <console>:30

scala> AvgRDD.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)

scala> val filterMapRDD = AvgRDD.filter(x=>x._1._2 == "grade-2" &&
x._2>50).count()
filterMapRDD: Long = 4

scala> val filterMapRDD = AvgRDD.filter(x=>x._1._2 == "grade-2" &&
x._2>50).foreach(println)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)
filterMapRDD: Unit = ()
```



```
scala> val filterMapRDD = AvgRDD.filter(x=>x._1._2 == "grade-2" && x._2>50).foreach(printl
n)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)
filterMapRDD: Unit = ()
```

PROBLEM STATEMENT 3:

Are there any students in the college that satisfy the below criteria:

TASK 3: Average score per student_name across all grades is same as average score per student_name per grade
Hint - Use Intersection Property.

=> To find the solution of above problem we will first calculate average of each student across all grades i.e. irrespective of grade.

EXPLANATION: We created a paired RDD named as testRDD1 by extracting only name and marks. Then we are mapping each value of above RDD with 1. Then we are adding the marks and number of occurrences for each student using reducebyKey. Then in next step we are calculating the average of each student.

CODE:

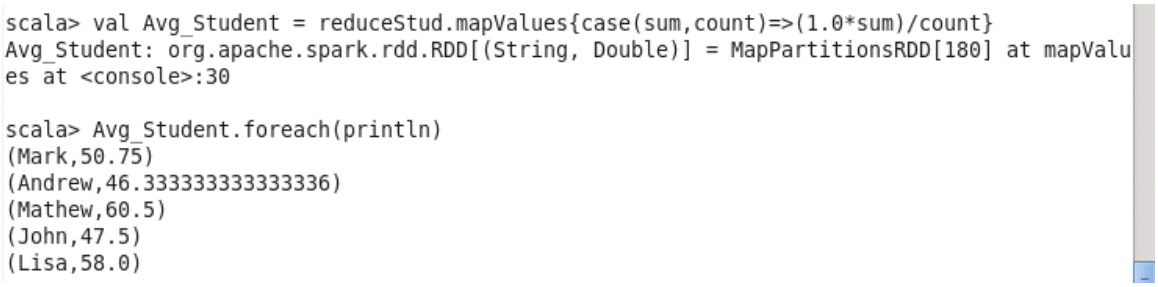
```
val testRDD1 = sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>(x.split(",")(0),x.split(",")(3).toInt))

val AvgStudent = testRDD1.mapValues(x=>(x,1))

val reduceStud= AvgStudent.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val Avg_Student = reduceStud.mapValues{case(sum,count)=>(1.0*sum)/count}
```

SOLUTION REPORT:

```
scala> val testRDD1 =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>(x.split(",")(0),x.split(",")(3).toInt))
testRDD1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[176]
at map at <console>:24

scala> testRDD1.foreach(println)
(Mathew,45)
(Mathew,55)
(Mark,23)
(Mark,76)
(John,14)
(John,74)
(Lisa,24)
(Lisa,86)
```

```
(Andrew,34)
(Andrew,26)
(Andrew,74)
(Mathew,55)
(Mathew,87)
(Mark,92)
(Mark,12)
(John,67)
(John,35)
(Lisa,24)
(Lisa,98)
(Andrew,23)
(Andrew,44)
(Andrew,77)

scala> val AvgStudent = testRDD1.mapValues(x=>(x,1)).foreach(println)
(Mathew,(45,1))
(Mathew,(55,1))
(Mark,(23,1))
(Mark,(76,1))
(John,(14,1))
(John,(74,1))
(Lisa,(24,1))
(Lisa,(86,1))
(Andrew,(34,1))
(Andrew,(26,1))
(Andrew,(74,1))
(Mathew,(55,1))
(Mathew,(87,1))
(Mark,(92,1))
(Mark,(12,1))
(John,(67,1))
(John,(35,1))
(Lisa,(24,1))
(Lisa,(98,1))
(Andrew,(23,1))
(Andrew,(44,1))
(Andrew,(77,1))
AvgStudent: Unit = ()

scala> val reduceStud= AvgStudent.reduceByKey((x,y) => (x._1 + y._1, x._2
+ y._2))
reduceStud: org.apache.spark.rdd.RDD[(String, (Int, Int))] =
ShuffledRDD[179] at reduceByKey at <console>:28

scala> reduceStud.foreach(println)
(Mark,(203,4))
(Andrew,(278,6))
(Mathew,(242,4))
(John,(190,4))
(Lisa,(232,4))

scala> val Avg_Student =
```

```
reduceStud.mapValues{case(sum,count)=>(1.0*sum)/count}
Avg_Student: org.apache.spark.rdd.RDD[(String, Double)] =
MapPartitionsRDD[180] at mapValues at <console>:30

scala> Avg_Student.foreach(println)
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,60.5)
(John,47.5)
(Lisa,58.0)
```



```
scala> val Avg_Student = reduceStud.mapValues{case(sum,count)=>(1.0*sum)/count}
Avg_Student: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[180] at mapValu
es at <console>:30

scala> Avg_Student.foreach(println)
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,60.5)
(John,47.5)
(Lisa,58.0)
```

=> Now the second step of this problem is to find the average of each student per grade.So used below code to:

CODE:

```
val testRDD2 = sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val grade = testRDD2.mapValues(x=>(x,1))

val reduce_grade = grade.reduceByKey((x,y) =>(x._1 + y._1, x._2 + y._2))

val Avg_grade = reduce_grade.mapValues{case(sum,count)=>(1.0*sum)/count}
```

EXPLANATION: So first we are creating another paired RDD named as testRDD2 by extracting name and grade as key and marks as value from the input file. Then we are mapping each value of testRDD2 with 1 using mapValues function. Then we are adding the marks and number of occurrences of 1 for each key using reduceByKey() function. Then we are calculating average of each key by dividing the sum of marks with the count.

SOLUTION REPORT:

```
scala> val testRDD2 =
sc.textFile("file:///home/acadgild/dataset.txt").map(x
=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
testRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] =
MapPartitionsRDD[183] at map at <console>:24

scala> testRDD2.foreach(println)
((Mathew,grade-3),45)
((Mathew,grade-2),55)
((Mark,grade-2),23)
((Mark,grade-1),76)
```

```
((John,grade-1),14)
((John,grade-2),74)
((Lisa,grade-1),24)
((Lisa,grade-3),86)
((Andrew,grade-1),34)
((Andrew,grade-3),26)
((Andrew,grade-1),74)
((Mathew,grade-2),55)
((Mathew,grade-2),87)
((Mark,grade-1),92)
((Mark,grade-2),12)
((John,grade-1),67)
((John,grade-1),35)
((Lisa,grade-2),24)
((Lisa,grade-2),98)
((Andrew,grade-1),23)
((Andrew,grade-3),44)
((Andrew,grade-2),77)

scala> val grade = testRDD2.mapValues(x=>(x,1))
grade: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
MapPartitionsRDD[184] at mapValues at <console>:26

scala> grade.foreach(println)
((Mathew,grade-3),(45,1))
((Mathew,grade-2),(55,1))
((Mark,grade-2),(23,1))
((Mark,grade-1),(76,1))
((John,grade-1),(14,1))
((John,grade-2),(74,1))
((Lisa,grade-1),(24,1))
((Lisa,grade-3),(86,1))
((Andrew,grade-1),(34,1))
((Andrew,grade-3),(26,1))
((Andrew,grade-1),(74,1))
((Mathew,grade-2),(55,1))
((Mathew,grade-2),(87,1))
((Mark,grade-1),(92,1))
((Mark,grade-2),(12,1))
((John,grade-1),(67,1))
((John,grade-1),(35,1))
((Lisa,grade-2),(24,1))
((Lisa,grade-2),(98,1))
((Andrew,grade-1),(23,1))
((Andrew,grade-3),(44,1))
((Andrew,grade-2),(77,1))

scala> val reduce_grade = grade.reduceByKey((x,y) =>(x._1 + y._1, x._2 +
y._2))
reduce_grade: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] =
ShuffledRDD[185] at reduceByKey at <console>:28

scala> reduce_grade.foreach(println)
```

```
((Lisa,grade-1),(24,1))
((Mark,grade-2),(35,2))
((Lisa,grade-2),(122,2))
((Mathew,grade-3),(45,1))
((Andrew,grade-2),(77,1))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((John,grade-1),(116,3))
((John,grade-2),(74,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
((Mathew,grade-2),(197,3))

scala> val Avg_grade =
reduce_grade.mapValues{case(sum,count)=>(1.0*sum)/count}
Avg_grade: org.apache.spark.rdd.RDD[((String, String), Double)] =
MapPartitionsRDD[186] at mapValues at <console>:30

scala> Avg_grade.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
```

=> Now to proceed further we are extracting name and marks from above RDD.

EXPLANATION: Then next we are using intersection function between flatAvg_grade and flatAvg_Student RDD's to find whether any common student is there.

Using command "CommonValue.foreach(println)" we check whether there is any common students that are having average score per student_name across all grades is same as average score per student_name per grade.

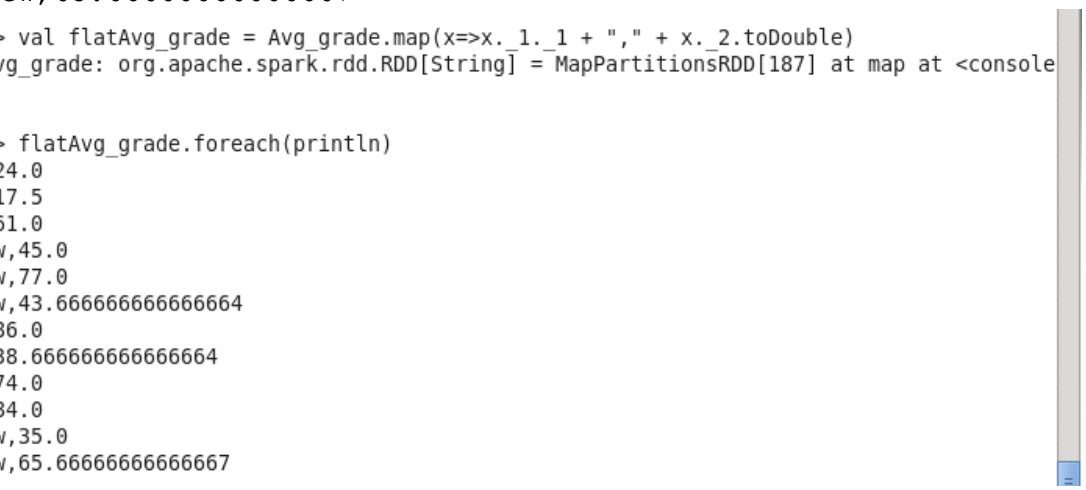
CODE:

```
val flatAvg_grade = Avg_grade.map(x=>x._1._1 + "," + x._2.toDouble)

val flatAvg_Student = Avg_Student.map(x=>x._1 + "," + x._2)

val CommonValue = flatAvg_grade.intersection(flatAvg_Student)
```

SOLUTION REPORT:

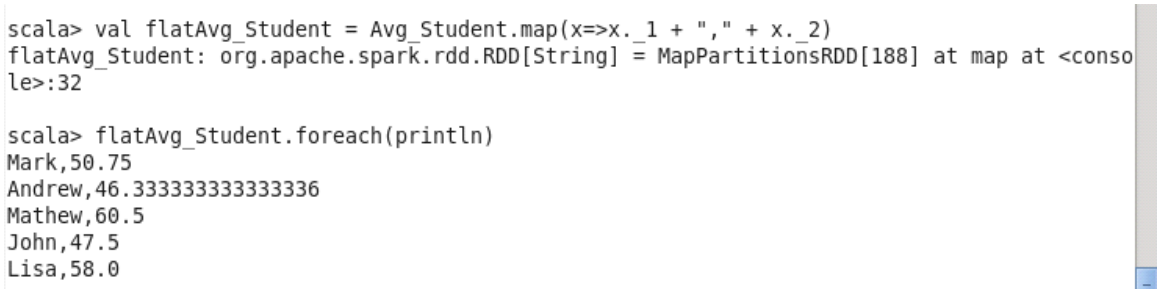
```
scala> val flatAvg_grade = Avg_grade.map(x=>x._1._1 + "," + x._2.toDouble)
flatAvg_grade: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[187]
```

```
at map at <console>:32

scala> flatAvg_grade.foreach(println)
Lisa,24.0
Mark,17.5
Lisa,61.0
Mathew,45.0
Andrew,77.0
Andrew,43.666666666666664
Lisa,86.0
John,38.666666666666664
John,74.0
Mark,84.0
Andrew,35.0
Mathew,65.66666666666667
```



```
scala> val flatAvg_grade = Avg_grade.map(x=>x._1._1 + "," + x._2.toDouble)
flatAvg_grade: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[187] at map at <console
>:32

scala> flatAvg_grade.foreach(println)
Lisa,24.0
Mark,17.5
Lisa,61.0
Mathew,45.0
Andrew,77.0
Andrew,43.666666666666664
Lisa,86.0
John,38.666666666666664
John,74.0
Mark,84.0
Andrew,35.0
Mathew,65.66666666666667
```

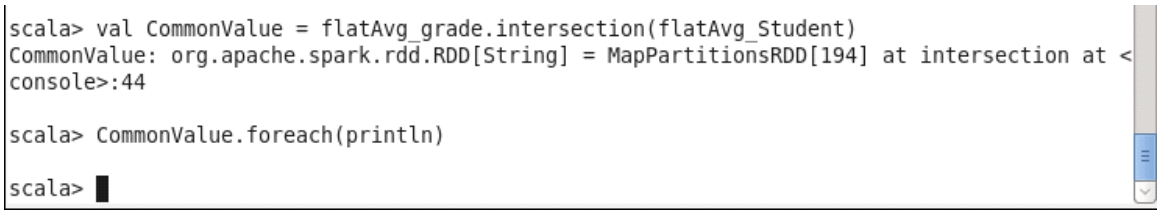
```
scala> val flatAvg_Student = Avg_Student.map(x=>x._1 + "," + x._2)
flatAvg_Student: org.apache.spark.rdd.RDD[String] =
MapPartitionsRDD[188] at map at <console>:32

scala> flatAvg_Student.foreach(println)
Mark,50.75
Andrew,46.333333333333336
Mathew,60.5
John,47.5
Lisa,58.0
```



```
scala> val flatAvg_Student = Avg_Student.map(x=>x._1 + "," + x._2)
flatAvg_Student: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[188] at map at <conso
le>:32

scala> flatAvg_Student.foreach(println)
Mark,50.75
Andrew,46.333333333333336
Mathew,60.5
John,47.5
Lisa,58.0
```

```
scala> val CommonValue = flatAvg_grade.intersection(flatAvg_Student)
CommonValue: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[194] at
intersection at <console>:44
```

```
scala> CommonValue.foreach(println)

scala>
```



=> So here the command CommonValue.foreach(println) shows that no common students are there having average score per student_name across all grades is same as average score per student_name per grade.