Session 19:
RDD DEEP DIVE
Assignment 1

Task 1

1.Write a program to read a text file and print the number of rows of data in the document.
2. Write a program to read a text file and print the number of words in the document.
3. We have a document where the word separator is -, instead of space. Write a spark code, to obtain the count of the total number of words present in the document.

Task 2

Problem Statement 1:

1. Read the text file, and create a tupled rdd.
2. Find the count of total number of rows present.
3. What is the distinct number of subjects present in the entire school
4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

Problem Statement 2:

1. What is the count of students per grade in the school?
2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)
3. What is the average score of students in each subject across all grades?
4. What is the average score of students in each subject per grade?
5. For all students in grade-2, how many have average score greater than 50?

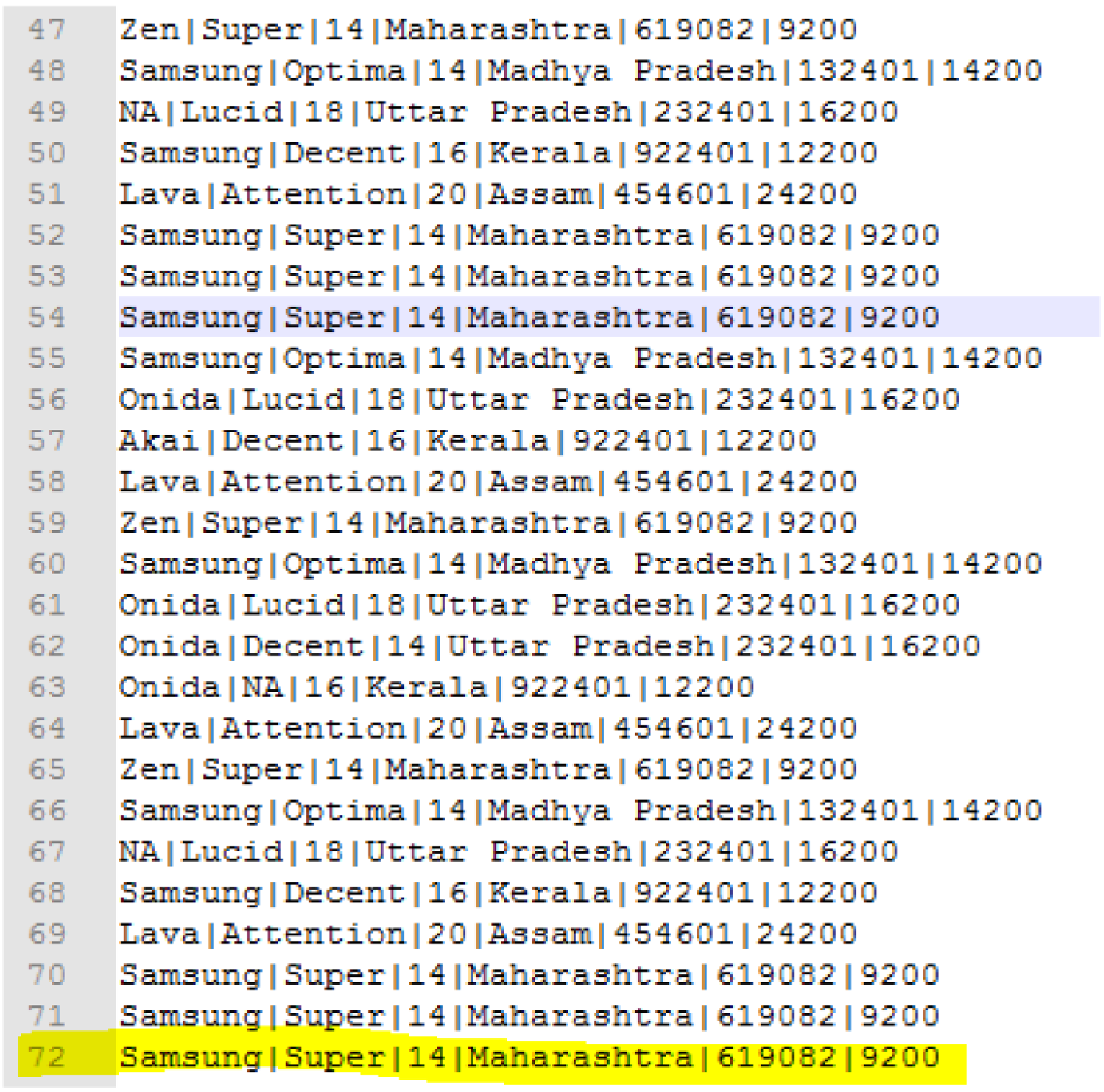
Problem Statement 3:

Are there any students in the college that satisfy the below criteria:
1. Average score per student_name across all grades is same as average score per student_name per grade
Hint - Use Intersection Property

Task 1

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

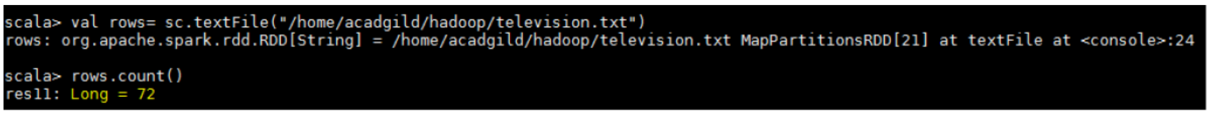
In this task, we are using a text file **“television.txt”** which has **72** rows as shown below,



```
47  Zen|Super|14|Maharashtra|619082|9200
48  Samsung|Optima|14|Madhya Pradesh|132401|14200
49  NA|Lucid|18|Uttar Pradesh|232401|16200
50  Samsung|Decent|16|Kerala|922401|12200
51  Lava|Attention|20|Assam|454601|24200
52  Samsung|Super|14|Maharashtra|619082|9200
53  Samsung|Super|14|Maharashtra|619082|9200
54  Samsung|Super|14|Maharashtra|619082|9200
55  Samsung|Optima|14|Madhya Pradesh|132401|14200
56  Onida|Lucid|18|Uttar Pradesh|232401|16200
57  Akai|Decent|16|Kerala|922401|12200
58  Lava|Attention|20|Assam|454601|24200
59  Zen|Super|14|Maharashtra|619082|9200
60  Samsung|Optima|14|Madhya Pradesh|132401|14200
61  Onida|Lucid|18|Uttar Pradesh|232401|16200
62  Onida|Decent|14|Uttar Pradesh|232401|16200
63  Onida|NA|16|Kerala|922401|12200
64  Lava|Attention|20|Assam|454601|24200
65  Zen|Super|14|Maharashtra|619082|9200
66  Samsung|Optima|14|Madhya Pradesh|132401|14200
67  NA|Lucid|18|Uttar Pradesh|232401|16200
68  Samsung|Decent|16|Kerala|922401|12200
69  Lava|Attention|20|Assam|454601|24200
70  Samsung|Super|14|Maharashtra|619082|9200
71  Samsung|Super|14|Maharashtra|619082|9200
72  Samsung|Super|14|Maharashtra|619082|9200
```

Read the text file,
scala> val rows= sc.textFile("/home/acadgild/hadoop/television.txt")

scala> rows.count()
res0: Long = 72

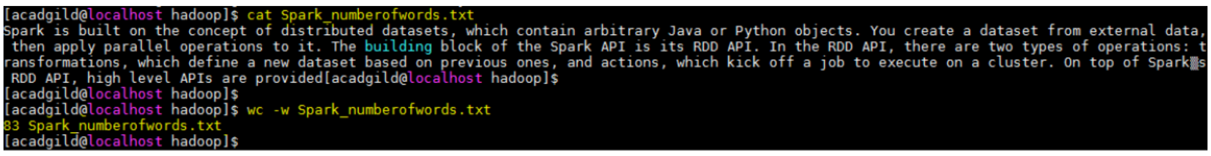


```
scala> val rows= sc.textFile("/home/acadgild/hadoop/television.txt")
rows: org.apache.spark.rdd.RDD[String] = /home/acadgild/hadoop/television.txt MapPartitionsRDD[21] at textFile at <console>:24

scala> rows.count()
res11: Long = 72
```

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

In this task, we are using a text file "***Spark_numberofwords.txt"*** which we created and it has number of words as 83, please see below,
cat Spark_numberofwords.txt

wc -w Spark_numberofwords.txt



```
[acadgild@localhost hadoop]$ cat Spark_numberofwords.txt
Spark is built on the concept of distributed datasets, which contain arbitrary Java or Python objects. You create a dataset from external data,
 then apply parallel operations to it. The building block of the Spark API is its RDD API. In the RDD API, there are two types of operations: t
ransformations, which define a new dataset based on previous ones, and actions, which kick off a job to execute on a cluster. On top of Spark▒s
 RDD API, high level APIs are provided[acadgild@localhost hadoop]$
[acadgild@localhost hadoop]$
[acadgild@localhost hadoop]$ wc -w Spark_numberofwords.txt
83 Spark_numberofwords.txt
[acadgild@localhost hadoop]$
```

Spark Operation

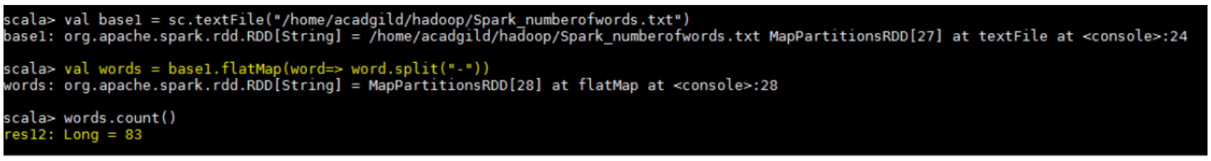
Read the text file,

scala> val base = sc.textFile("/home/acadgild/hadoop/Spark_numberofwords.txt")

scala> val words = base.flatMap(word=> word.split(" "))

scala> words.count()

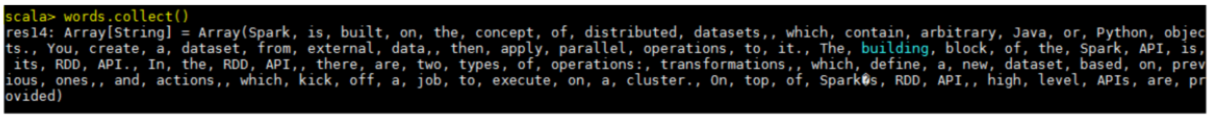
res12: Long = 83



```
scala> val base1 = sc.textFile("/home/acadgild/hadoop/Spark_numberofwords.txt")
base1: org.apache.spark.rdd.RDD[String] = /home/acadgild/hadoop/Spark_numberofwords.txt MapPartitionsRDD[27] at textFile at <console>:24

scala> val words = base1.flatMap(word=> word.split("-"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[28] at flatMap at <console>:28

scala> words.count()
res12: Long = 83
```

3. We have a document where the word separator is -, instead of space. Write a spark code, to obtain the count of the total number of words present in the document.

words.collect()



```
scala> words.collect()
res14: Array[String] = Array(Spark, is, built, on, the, concept, of, distributed, datasets,, which, contain, arbitrary, Java, or, Python, objec
ts., You, create, a, dataset, from, external, data,, then, apply, parallel, operations, to, it., The, building, block, of, the, Spark, API, is,
 its, RDD, API., In, the, RDD, API,, there, are, two, types, of, operations:, transformations,, which, define, a, new, dataset, based, on, prev
ious, ones,, and, actions,, which, kick, off, a, job, to, execute, on, a, cluster., On, top, of, Spark�s, RDD, API,, high, level, APIs, are, pr
ovided)
```

Task 2

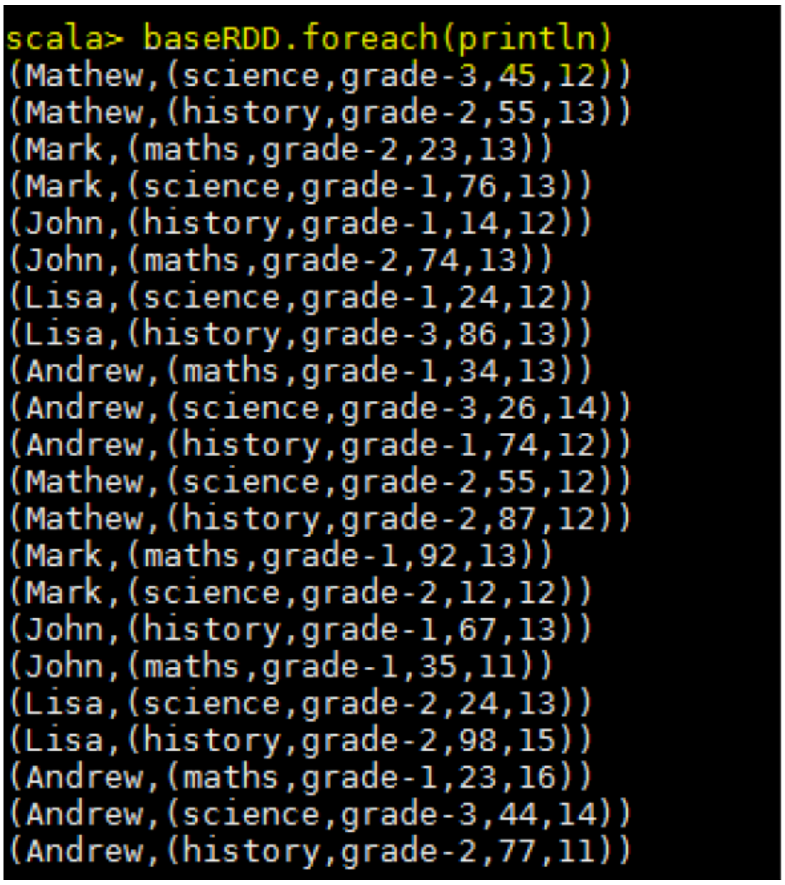
Problem Statement 1:

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

Below is the code we use to read the text file using spark context and creating a tuple RDD,

scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x => (x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt,x.split(",")(4).toInt)))
scala> baseRDD.foreach(println)

We have create a tuple RDD with name as Key and the subject, grades and the marks as values.

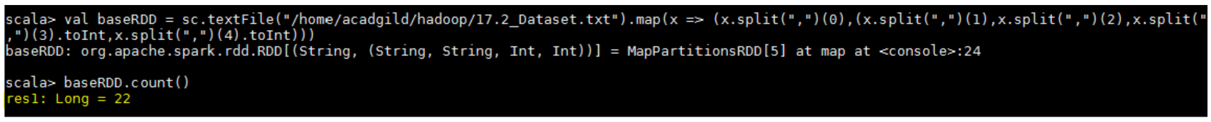


```
scala> baseRDD.foreach(println)
(Mathew,(science,grade-3,45,12))
(Mathew,(history,grade-2,55,13))
(Mark,(maths,grade-2,23,13))
(Mark,(science,grade-1,76,13))
(John,(history,grade-1,14,12))
(John,(maths,grade-2,74,13))
(Lisa,(science,grade-1,24,12))
(Lisa,(history,grade-3,86,13))
(Andrew,(maths,grade-1,34,13))
(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))
```

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

scala> baseRDD.count()
res1: Long = 22

By using count() function we can see the number of lines present in the text file.



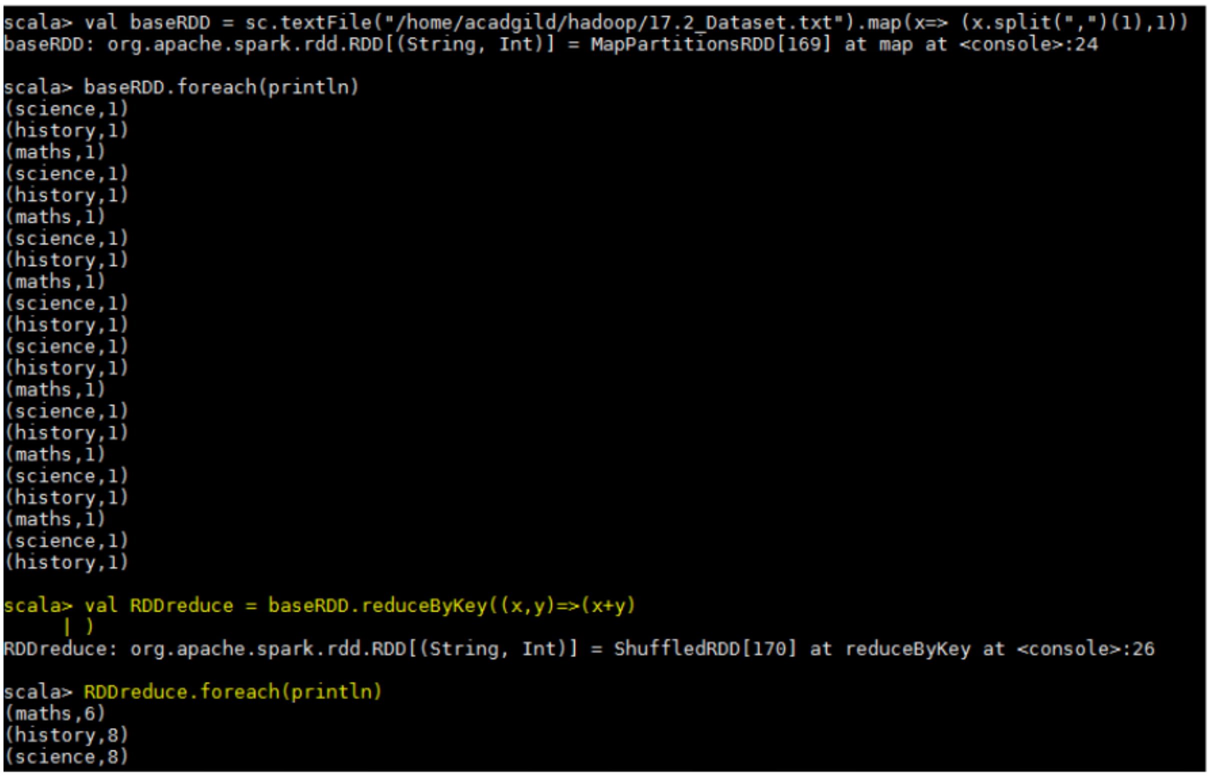
```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x => (x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split("
,")(3).toInt,x.split(",")(4).toInt)))
baseRDD: org.apache.spark.rdd.RDD[(String, (String, String, Int, Int))] = MapPartitionsRDD[5] at map at <console>:24

scala> baseRDD.count()
res1: Long = 22
```

3. What is the distinct number of subjects present in the entire school

Please see the below codes used for this task,
val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=> (x.split(",")(1),1))
val RDDreduce = baseRDD.reduceByKey((x,y)=>(x+y))

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

First we are 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.

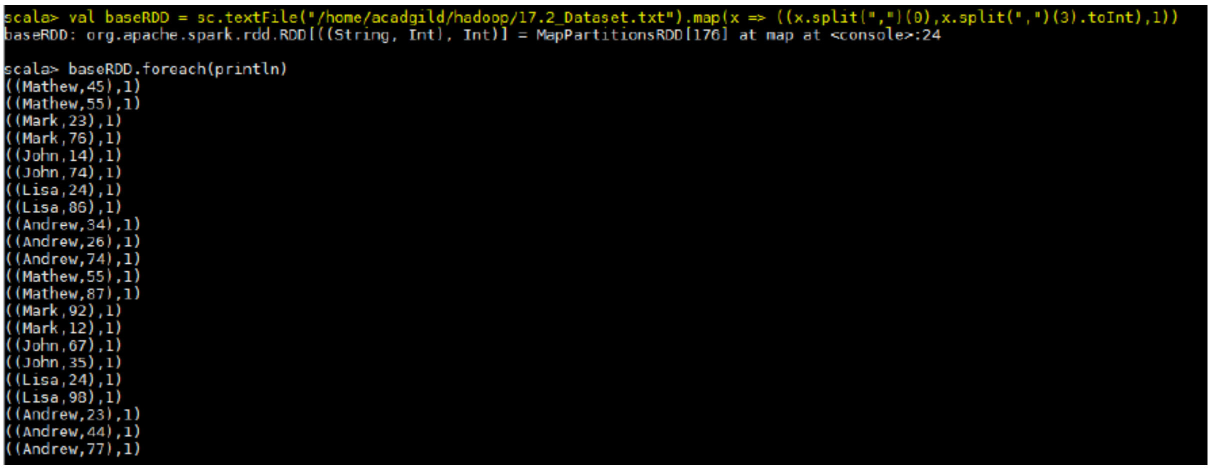


```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=> (x.split(",")(1),1))
baseRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[169] at map at <console>:24

scala> baseRDD.foreach(println)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)

scala> val RDDreduce = baseRDD.reduceByKey((x,y)=>(x+y)
     | )
RDDreduce: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[170] at reduceByKey at <console>:26

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

4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

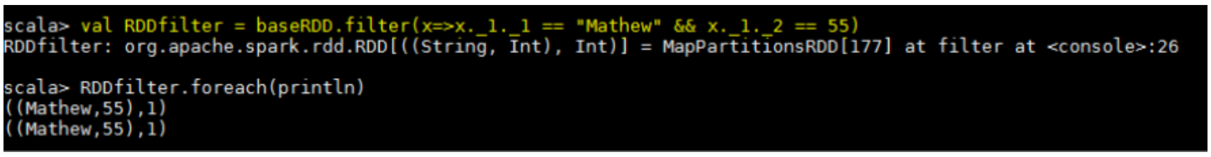
val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(3).toInt),1))
val RDDfilter = baseRDD.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)
val RDDreduce = RDDfilter.reduceByKey((x,y)=> x+y).foreach(println)

In the first line code, we are reading the text file and creating a tuple RDD as "baseRDD" with name & marks as key and mapping numerical 1 as value.



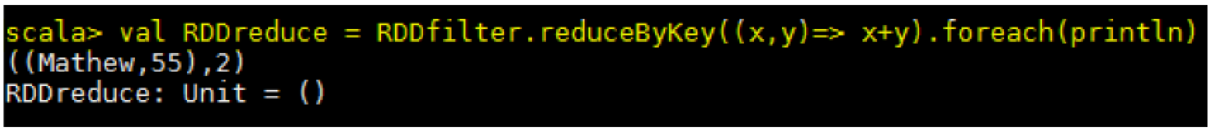
```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(3).toInt),1))
baseRDD: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[176] at map at <console>:24

scala> baseRDD.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)
```

Filter the tuple RDD by providing the condition Mather and mark as 55,



```
scala> val RDDfilter = baseRDD.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)
RDDfilter: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[177] at filter at <console>:26

scala> RDDfilter.foreach(println)
((Mathew,55),1)
((Mathew,55),1)
```

Now we are counting each occurrences using the **reduceByKey** and the required output is below,

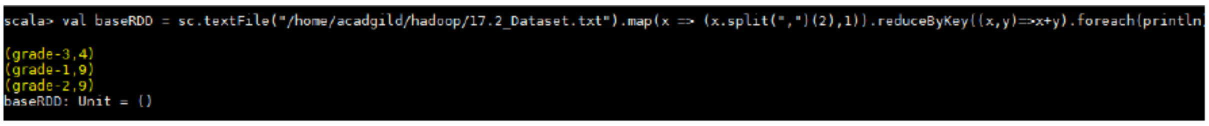


```
scala> val RDDreduce = RDDfilter.reduceByKey((x,y)=> x+y).foreach(println)
((Mathew,55),2)
RDDreduce: Unit = ()
```

Problem Statement 2:
1. What is the count of students per grade in the school?

val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x => (x.split(",")(2),1)).reduceByKey((x,y)=>x+y).foreach(println)

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, please see the required output below,

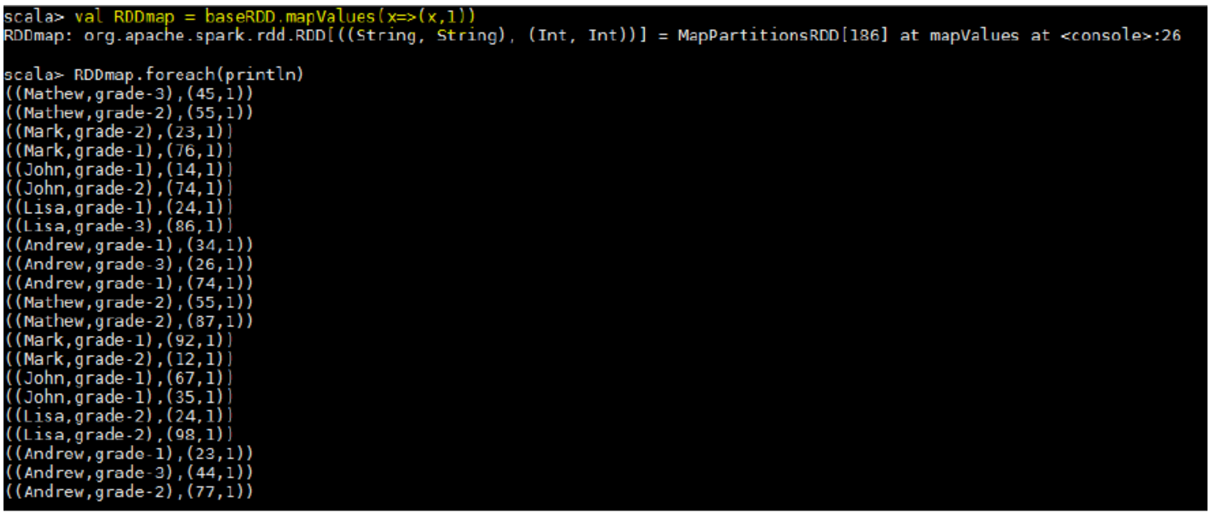


```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x => (x.split(",")(2),1)).reduceByKey((x,y)=>x+y).foreach(println

(grade-3,4)
(grade-1,9)
(grade-2,9)
baseRDD: Unit = ()
```

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

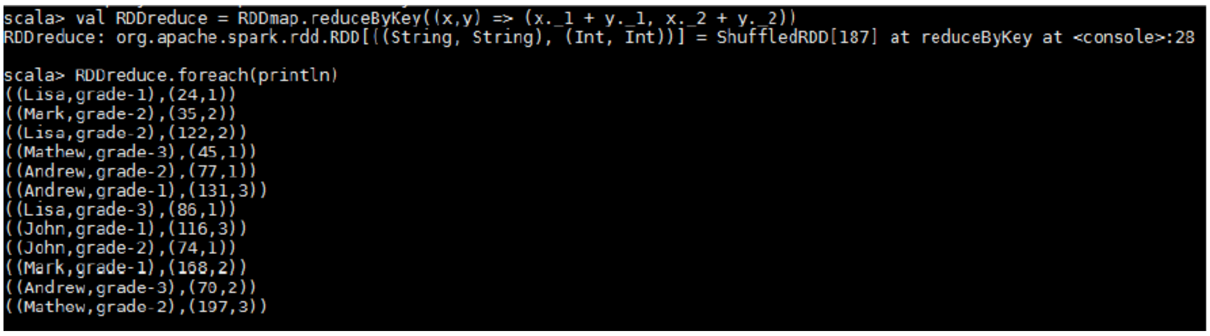
***val baseRDD =
sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(
",")(3).toInt))
val RDDmap = baseRDD.mapValues(x=>(x,1))
val RDDreduce = RDDmap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
val StudAvg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}***

First we are creating the **baseRDD** to read the file and selecting name and grade as key and marks as value,



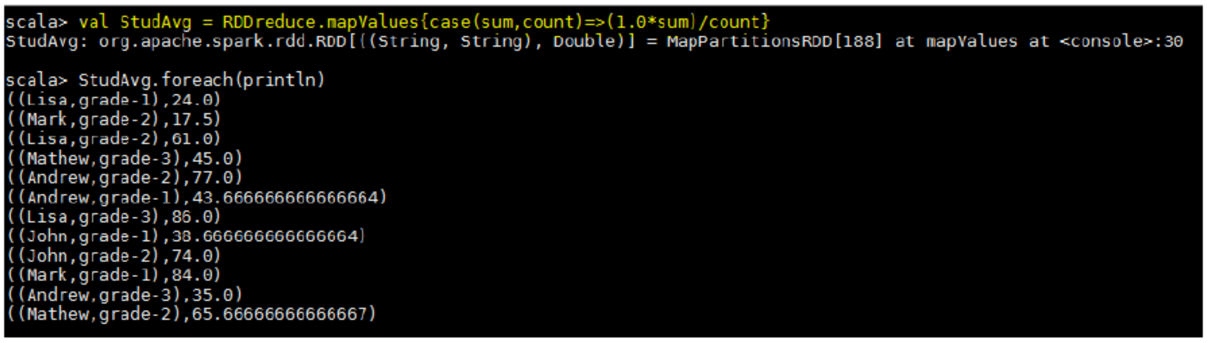
```
scala> val RDDmap = baseRDD.mapValues(x=>(x,1))
RDDmap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[186] at mapValues at <console>:26

scala> RDDmap.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))
```

We are using reduceByKey to add the occurrences of marks for each key which is student name and grade,



```
scala> val RDDreduce = RDDmap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
RDDreduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[187] at reduceByKey at <console>:28

scala> RDDreduce.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))
```

Now we are calculating average by summing the marks and dividing by its count for each key. Below screenshot shows the final result,

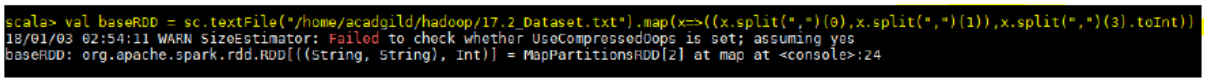


```
scala> val StudAvg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}
StudAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[188] at mapValues at <console>:30

scala> StudAvg.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)
```

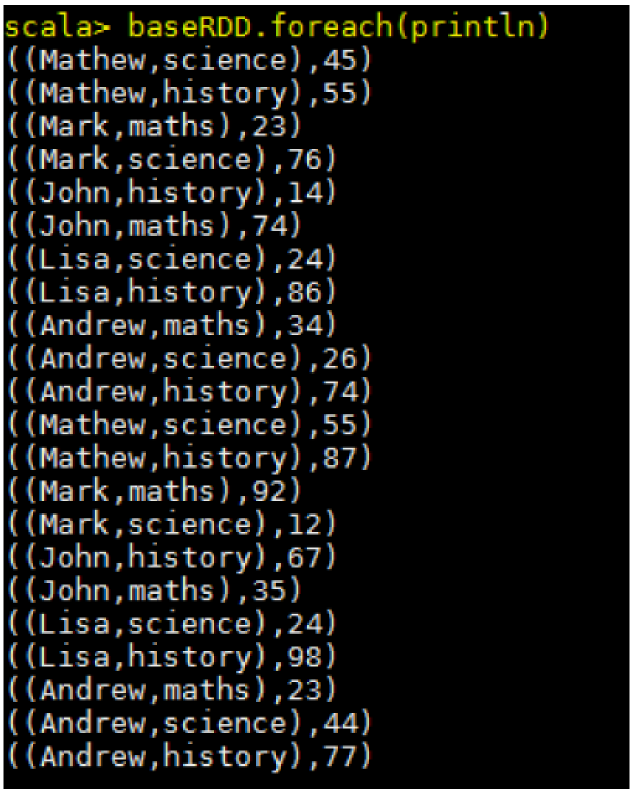
3. What is the average score of students in each subject across all grades?

val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
val RDDmap = baseRDD.mapValues(x=>(x,1))
val RDDreduce = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
val SubAvg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}

We are first creating baseRDD to read the text file and we are extracting name and subject as key and marks as value,

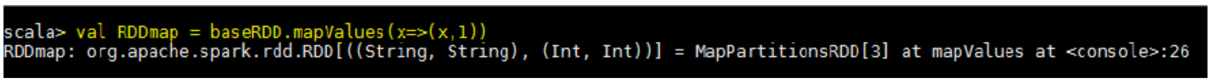


```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
18/01/03 02:54:11 WARN SizeEstimator: Failed to check whether UseCompressedOops is set; assuming yes
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[2] at map at <console>:24
```

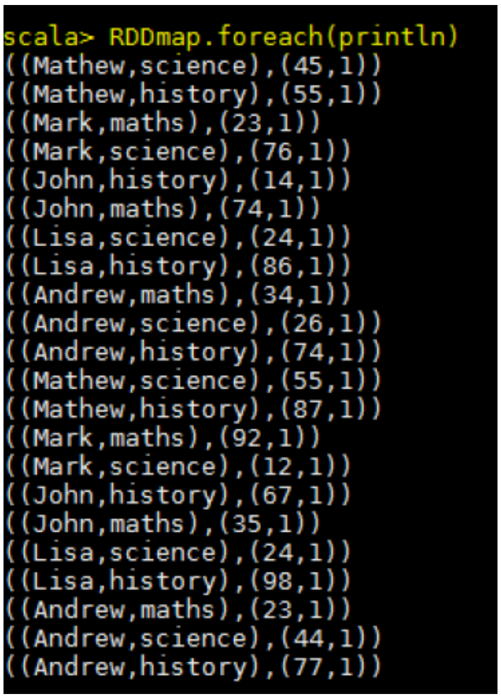


```
scala> baseRDD.foreach(println)
((Mathew,science),45)
((Mathew,history),55)
((Mark,maths),23)
((Mark,science),76)
((John,history),14)
((John,maths),74)
((Lisa,science),24)
((Lisa,history),86)
((Andrew,maths),34)
((Andrew,science),26)
((Andrew,history),74)
((Mathew,science),55)
((Mathew,history),87)
((Mark,maths),92)
((Mark,science),12)
((John,history),67)
((John,maths),35)
((Lisa,science),24)
((Lisa,history),98)
((Andrew,maths),23)
((Andrew,science),44)
((Andrew,history),77)
```

Now using **mapValues** we are mapping each value with 1-

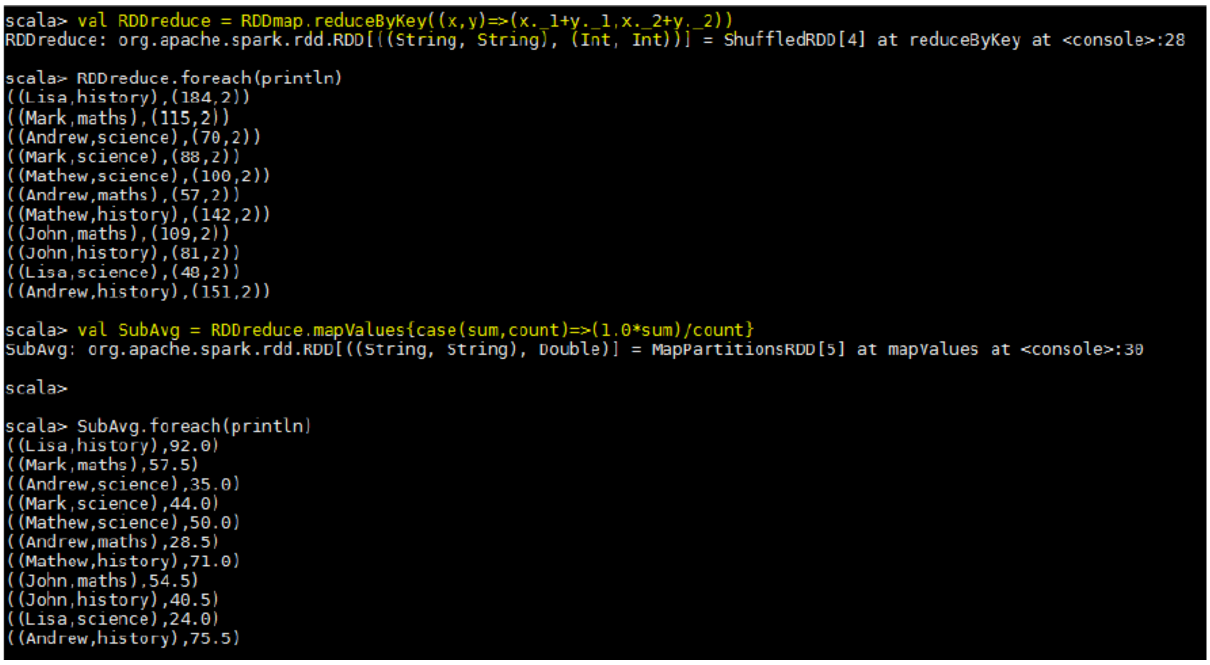


```
scala> val RDDmap = baseRDD.mapValues(x=>(x,1))
RDDmap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[3] at mapValues at <console>:26
```



```
scala> RDDmap.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))
```

Now 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.

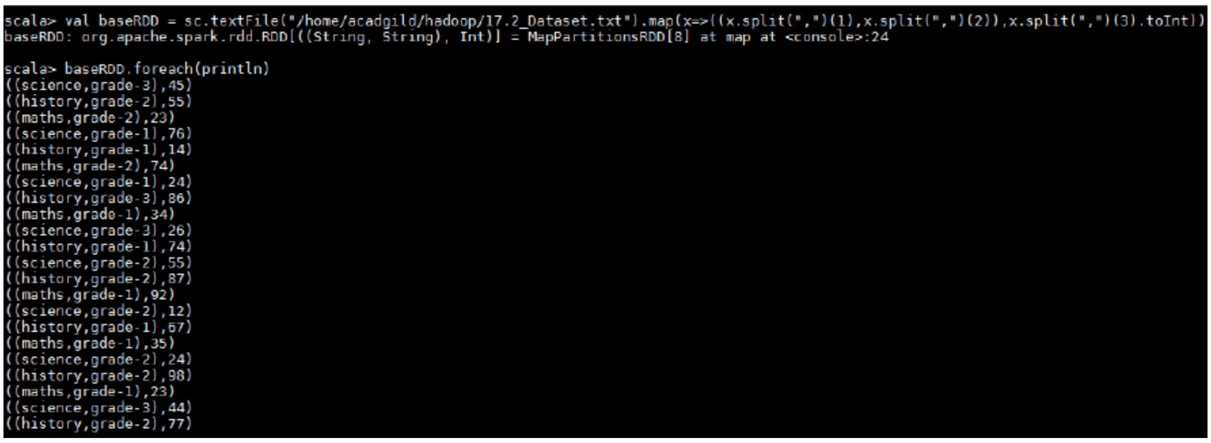


```
scala> val RDDreduce = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
RDDreduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[4] at reduceByKey at <console>:28

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

scala>

scala> SubAvg.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)
```

4. What is the average score of students in each subject per grade?

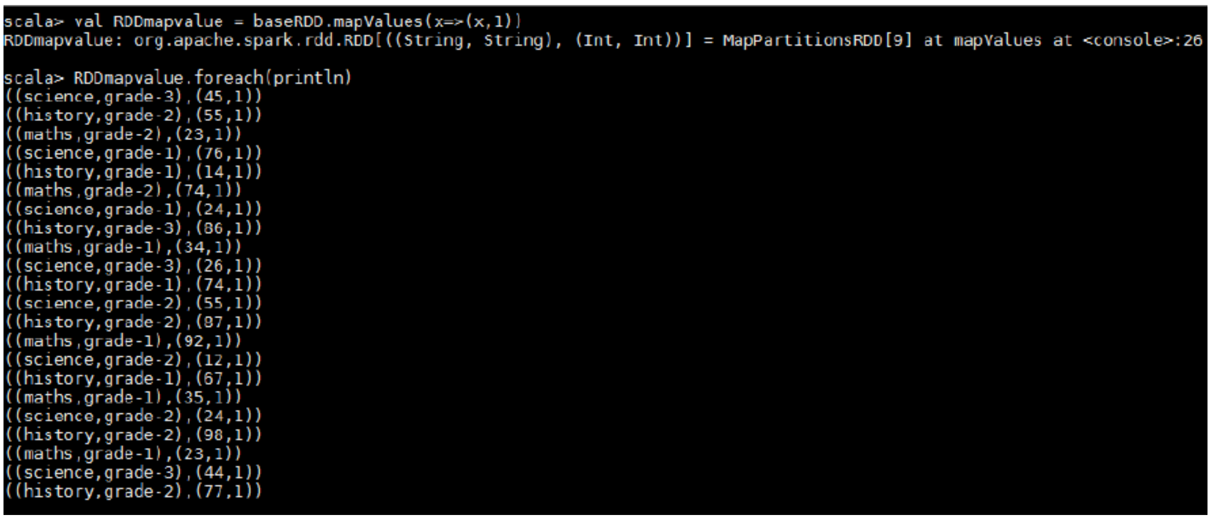
val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))
val RDDmapvalue = baseRDD.mapValues(x=>(x,1))
val RDDreduce = RDDmapvalue.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
val Avg_Grade = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}.foreach(println)

In first step we are creating paired RDD named as **baseRDD** to read the file and extracting subject and grade as key and marks as value



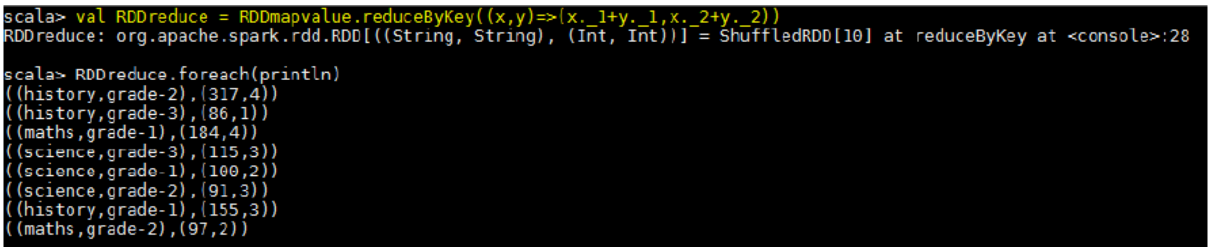
```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[8] at map at <console>:24

scala> baseRDD.foreach(println)
((science,grade-3),45)
((history,grade-2),55)
((maths,grade-2),23)
((science,grade-1),76)
((history,grade-1),14)
((maths,grade-2),74)
((science,grade-1),24)
((history,grade-3),86)
((maths,grade-1),34)
((science,grade-3),26)
((history,grade-1),74)
((science,grade-2),55)
((history,grade-2),87)
((maths,grade-1),92)
((science,grade-2),12)
((history,grade-1),67)
((maths,grade-1),35)
((science,grade-2),24)
((history,grade-2),98)
((maths,grade-1),23)
((science,grade-3),44)
((history,grade-2),77)
```

Then we are mapping the values of **baseRDD** with numerical value 1 using function **mapValues**



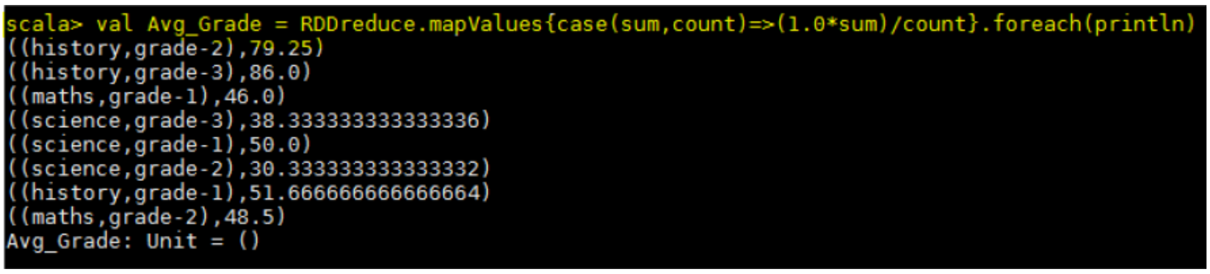
```
scala> val RDDmapvalue = baseRDD.mapValues(x=>(x,1))
RDDmapvalue: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[9] at mapValues at <console>:26

scala> RDDmapvalue.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))
```

We are adding marks and number of occurrences for each key using **reduceByKey**



```
scala> val RDDreduce = RDDmapvalue.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
RDDreduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[10] at reduceByKey at <console>:28

scala> RDDreduce.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))
```

We are calculating average by dividing the sum of marks with number of occurrences and the requested output is shown below,

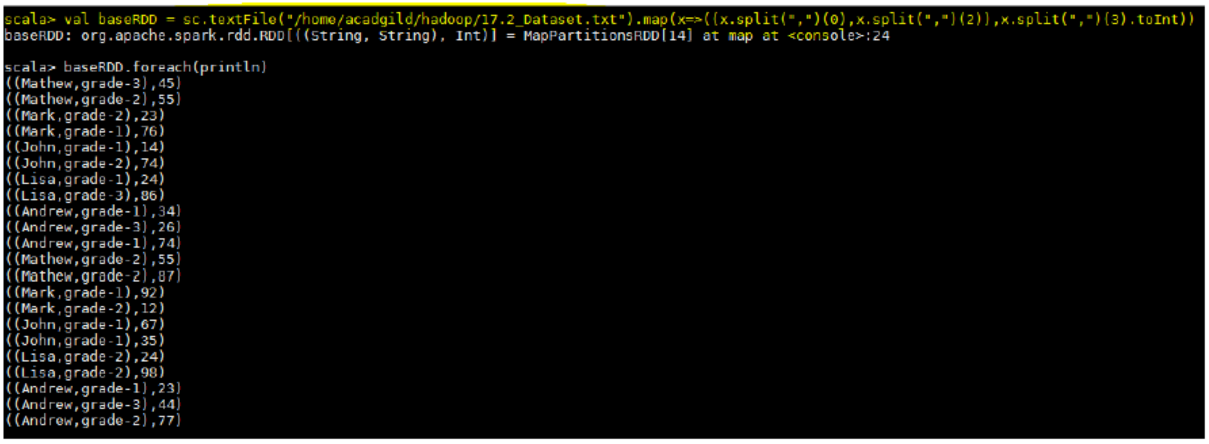


```
scala> val Avg_Grade = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}.foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),38.333333333333336)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
Avg_Grade: Unit = ()
```

5. For all students in grade-2, how many have average score greater than 50?

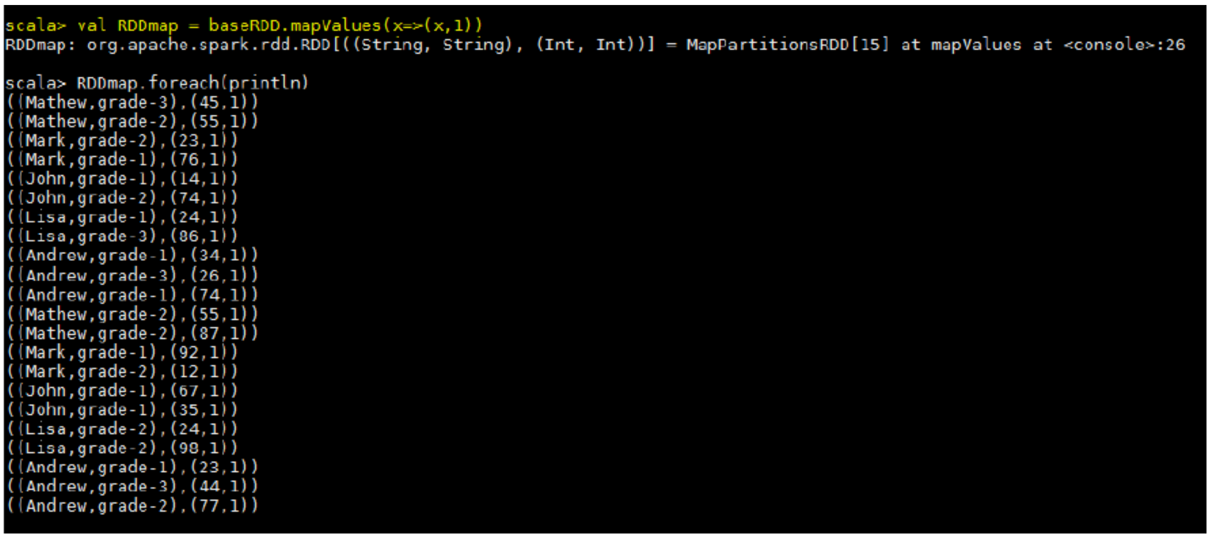
val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
val RDDmap = baseRDD.mapValues(x=>(x,1))
val RDDreduce = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
val RDDavg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}
val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).count()
val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).foreach(println)

Creating a paired RDD named as baseRDD to read the file and extracting name and grade as key and marks as value,



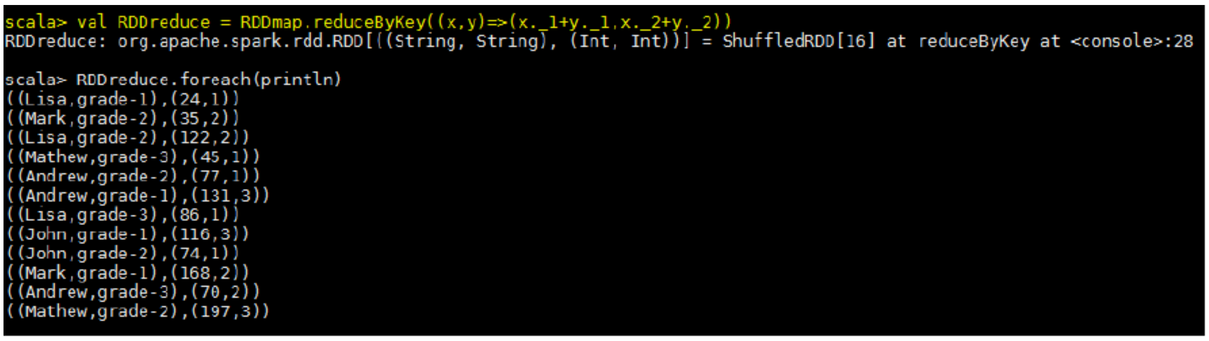
```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[14] at map at <console>:24

scala> baseRDD.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)
```

Now we are mapping each value of **baseRDD** with 1 as shown below,



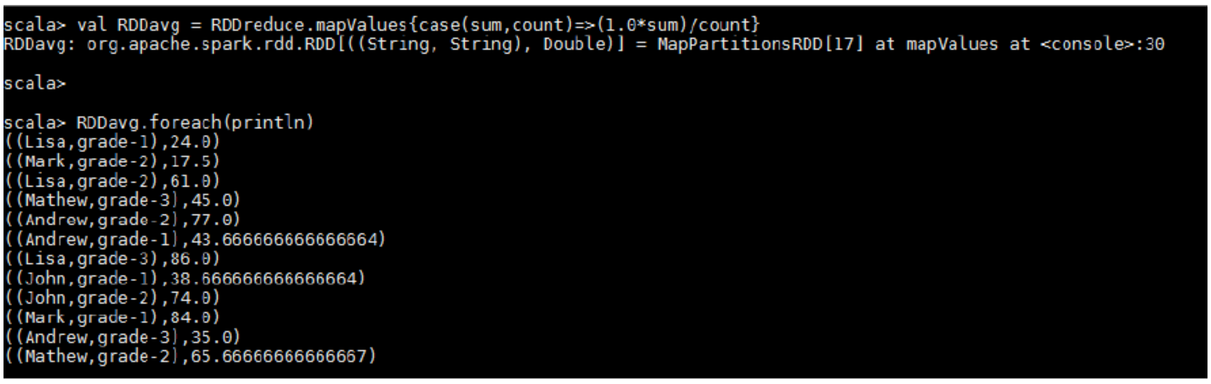
```
scala> val RDDmap = baseRDD.mapValues(x=>(x,1))
RDDmap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[15] at mapValues at <console>:26

scala> RDDmap.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))
```

We are adding the marks of subject and number of occurrences per key using **reduceByKey** function-



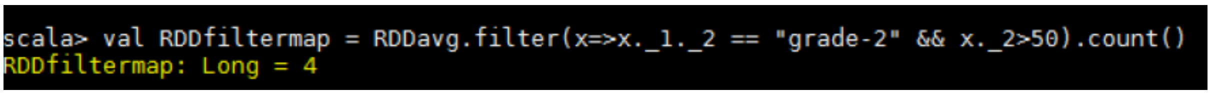
```
scala> val RDDreduce = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
RDDreduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[16] at reduceByKey at <console>:28

scala> RDDreduce.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))
```

Here we are calculating the average of each student,

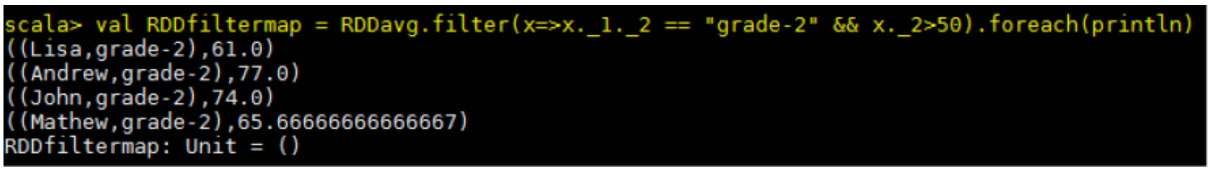


```
scala> val RDDavg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}
RDDavg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[17] at mapValues at <console>:30

scala>

scala> RDDavg.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 in below step we are filtering the above result with student belonging to **grade-2** and having marks **greater than 50,** the number of the counts and the data is shown in the below.



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



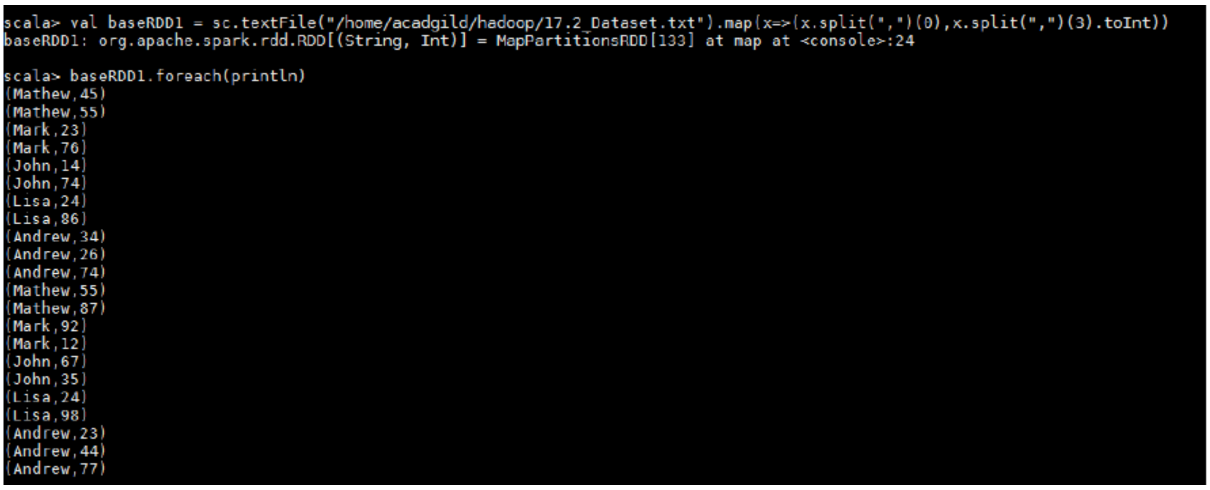
```
scala> val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).foreach(println)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)
RDDfiltermap: Unit = ()
```

Problem Statement 3:
Are there any students in the college that satisfy the below criteria:
1. Average score per student_name across all grades is same as average score per student_name per grade

First calculate average of each student across all grades i.e. irrespective of grade. Below is the code used to find the same

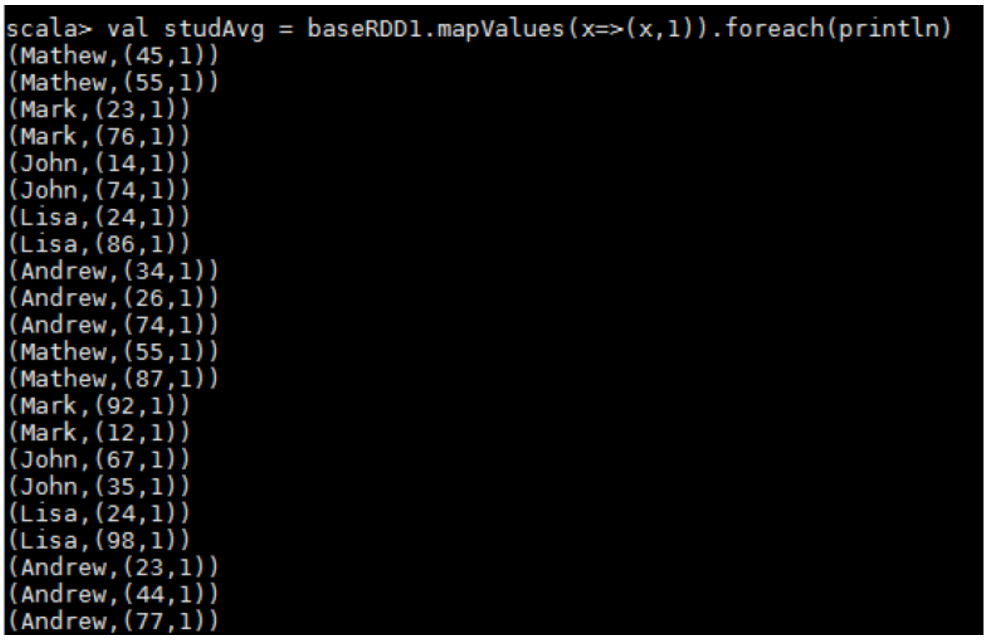
val baseRDD1 = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>(x.split(",")(0),x.split(",")(3).toInt))
val studAvg = baseRDD1.mapValues(x=>(x,1)).foreach(println)
scala> val studReduce = studAvg.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
scala> val Avg_Stud = studReduce.mapValues{case (sum,count) => (1.0 * sum)/count}

We created a paired RDD named as baseRDD1 by extracting only name and marks,

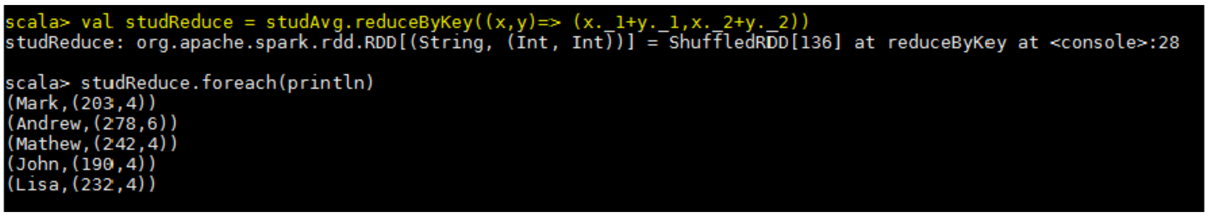


```
scala> val baseRDD1 = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>(x.split(",")(0),x.split(",")(3).toInt))
baseRDD1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[133] at map at <console>:24

scala> baseRDD1.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)
```

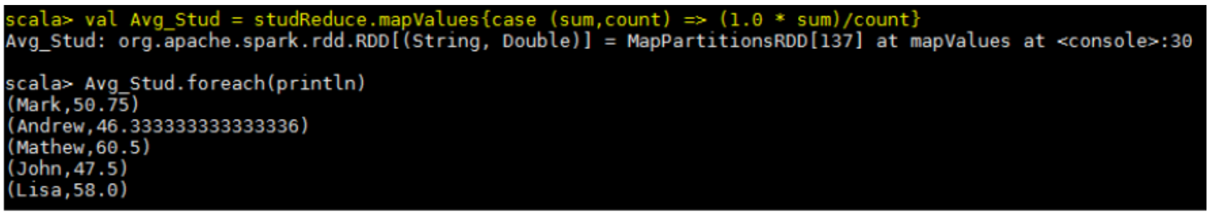
Then we are mapping each value of above RDD with 1,


```
scala> val studAvg = baseRDD1.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))
```

Then we are adding the marks and number of occurrences for each student using **redducebyKey** as shown below,


```
scala> val studReduce = studAvg.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
studReduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[136] at reduceByKey at <console>:28

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

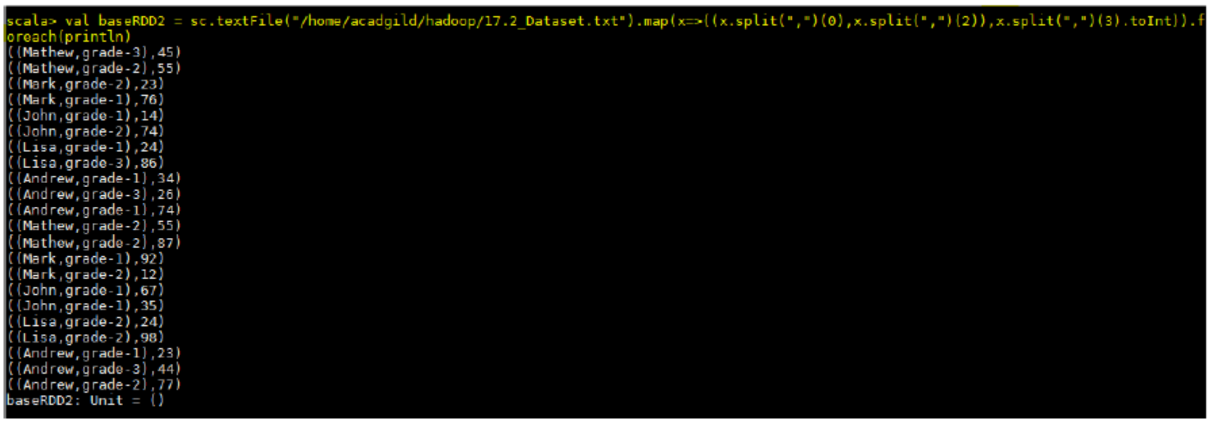
In below step we are calculating the average of each student,


```
scala> val Avg_Stud = studReduce.mapValues{case (sum,count) => (1.0 * sum)/count}
Avg_Stud: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[137] at mapValues at <console>:30

scala> Avg_Stud.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. We have used below code to find,

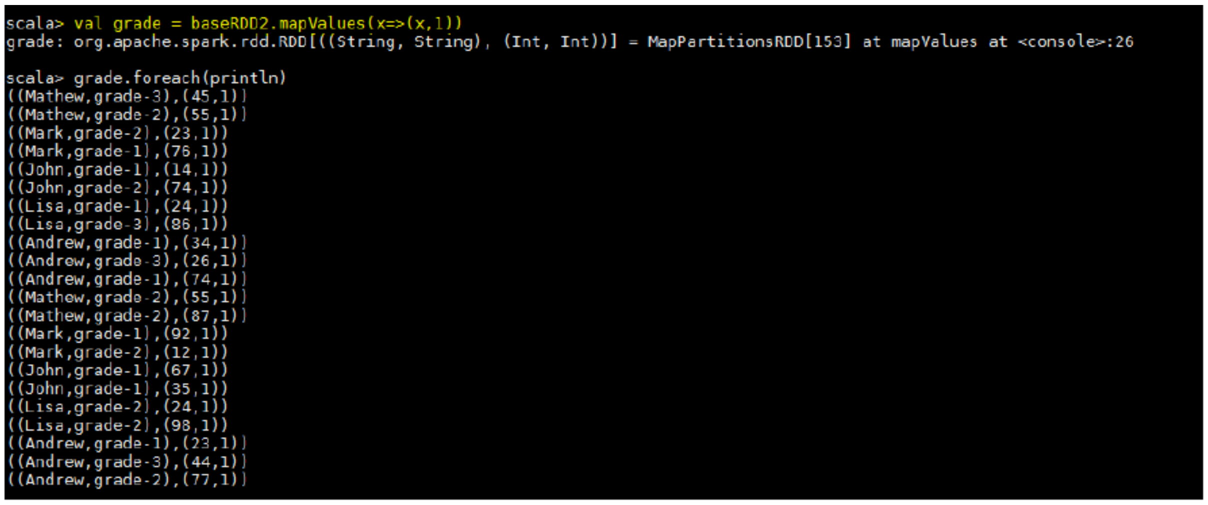
val baseRDD2 = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt)).foreach(println)
val grade = baseRDD2.mapValues(x=>(x,1))
val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
val gradeAvg = gradeReduce.mapValues{case(sum,count) => (1.0*sum)/count}

So first we are creating another paired RDD named as baseRDD2 by extracting name and grade as key and marks as value from the input file,



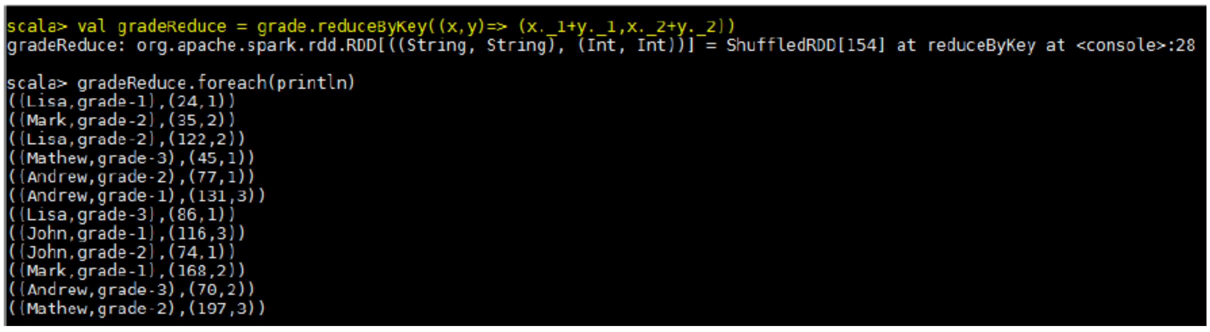
```
scala> val baseRDD2 = sc.textFile("/home/acadgild/hadoop/17.2_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt)).f
oreach(println)
((Mathew,grade-3),45)
((Mathew,grade-2),55)
((Mark,grade-2),23)
((Mark,grade-1),76)
((John,grade-1),14)
((John,grade-2),74)
((Lisa,grade-1),24)
((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)
baseRDD2: Unit = ()
```

Then we are mapping each value of baseRDD2 with 1 using **mapValues** function



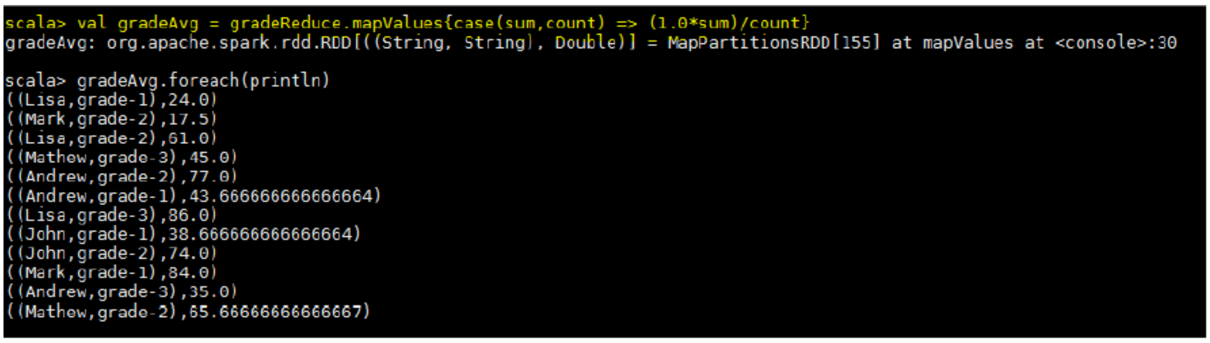
```
scala> val grade = baseRDD2.mapValues(x=>(x,1))
grade: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[153] at mapValues at <console>:26

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))
```

Then we are adding the marks and number of occurrences of 1 for each key using reduceByKey() function in below



```
scala> val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
gradeReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[154] at reduceByKey at <console>:28

scala> gradeReduce.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))
```

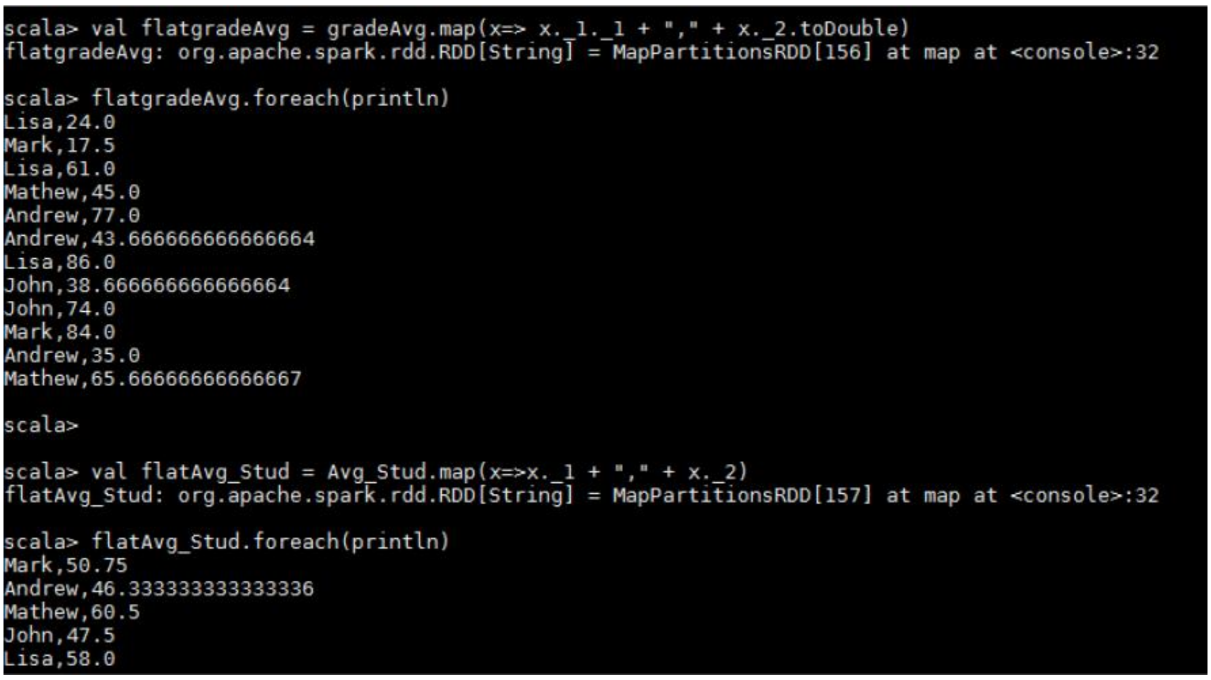
Calculating average of each key by dividing the sum of marks with the count



```
scala> val gradeAvg = gradeReduce.mapValues{case(sum,count) => (1.0*sum)/count}
gradeAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[155] at mapValues at <console>:30

scala> gradeAvg.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)
```

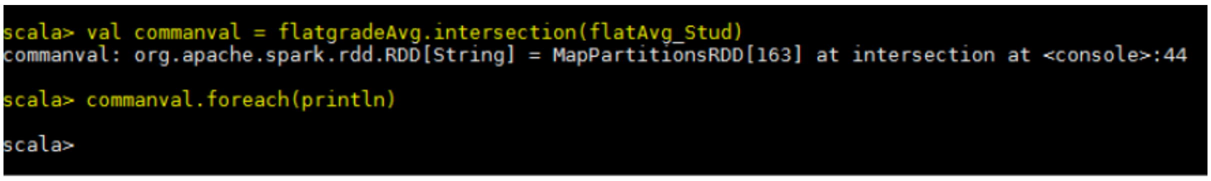
We are extracting name and marks from above RDD,
In below step we are using intersection function between flatgradeAvg and flatnameAvg rdd's to find whether any common student is there.

scala> val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
scala> val commanval = flatgradeAvg.intersection(flatAvg_Stud)



```
scala> val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[156] at map at <console>:32

scala> flatgradeAvg.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>

scala> val flatAvg_Stud = Avg_Stud.map(x=>x._1 + "," + x._2)
flatAvg_Stud: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[157] at map at <console>:32

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

So the command **comman.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



```
scala> val commanval = flatgradeAvg.intersection(flatAvg_Stud)
commanval: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[163] at intersection at <console>:44

scala> commanval.foreach(println)

scala>
```