Session 19: Assignment 19.1

Task 1:

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

<u>Solution:</u>

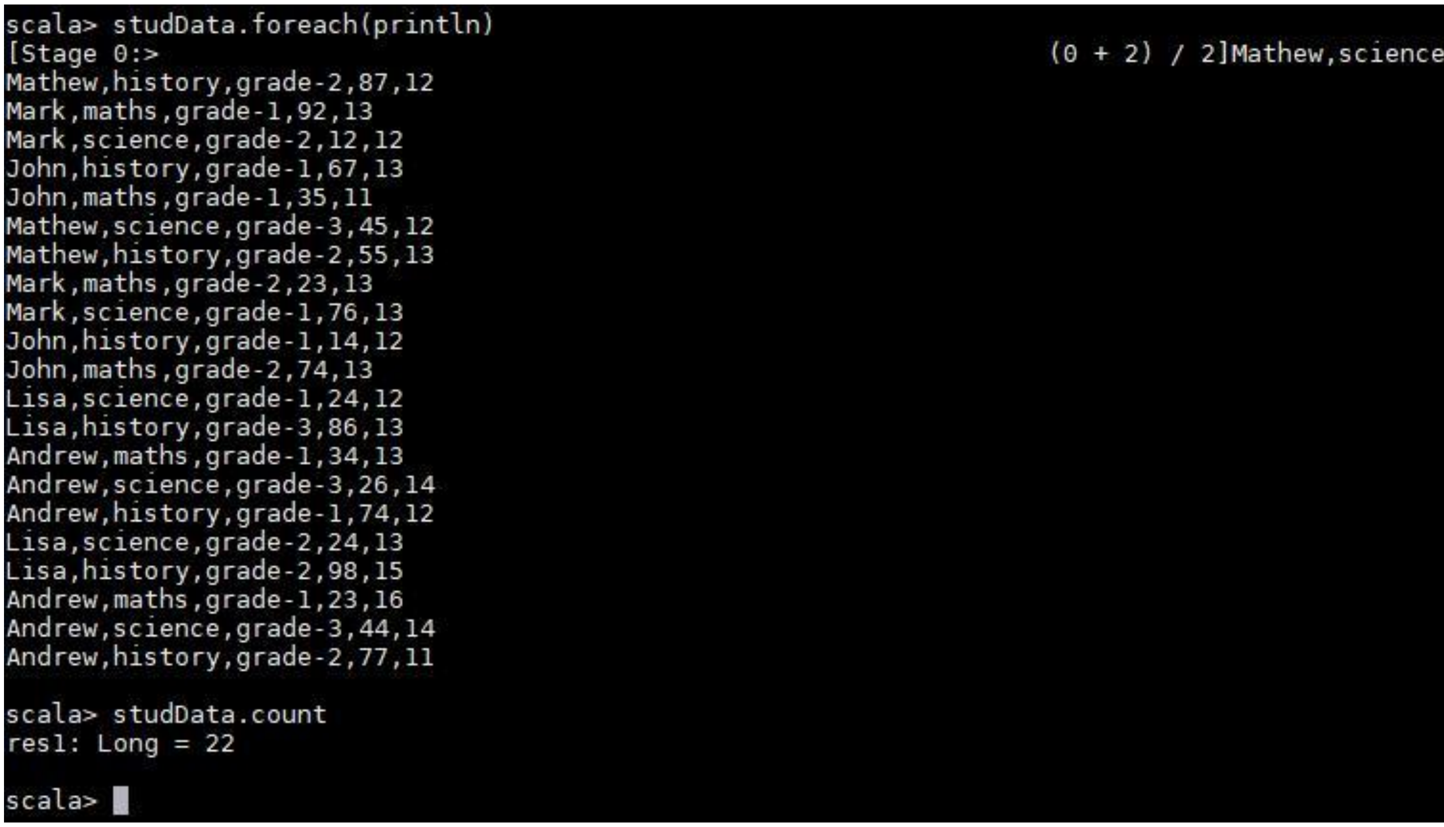
– We can extract the contents of text file stored in local file system, using the SparkContext

scala> val studData = sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")

Note: Since the question was ambiguous as number of rows means the count as well as the lines in file, providing solution for both.

– Now we will read the contents of the file using the below command *scala> studData.foreach(println)*

– We will count the number of rows in the file using the command below *scala> studData.count*

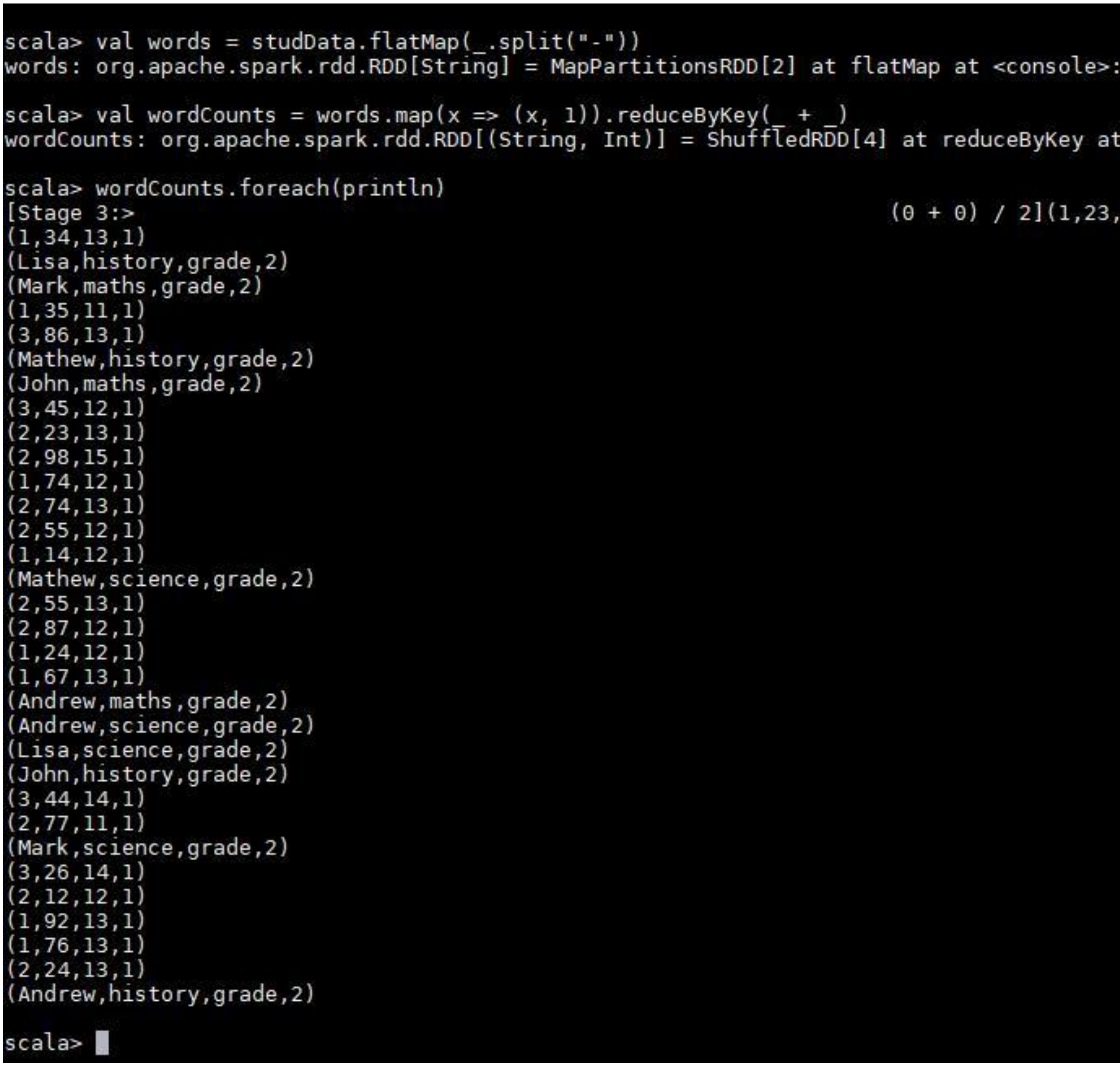


```
scala> studData.foreach(println)
[Stage 0:>                                          (0 + 2) / 2]Mathew,science
Mathew,history,grade-2,87,12
Mark,maths,grade-1,92,13
Mark,science,grade-2,12,12
John,history,grade-1,67,13
John,maths,grade-1,35,11
Mathew,science,grade-3,45,12
Mathew,history,grade-2,55,13
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
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

scala> studData.count
res1: Long = 22

scala> 
```

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

Solution:

– We can extract the contents of the text file stored in local file system, using the SparkContext
  scala> val studData = sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")
– Now we will read the contents of the file using the below command *scala> val words = studData.flatMap(_.split("-"))*
– This command will split the text file based on the (**-**) separator *scala> val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)*
– This command will sum up all the rows and sort them accordingly *scala> wordCounts.foreach(println)*
– This command will print the number of words in the text file



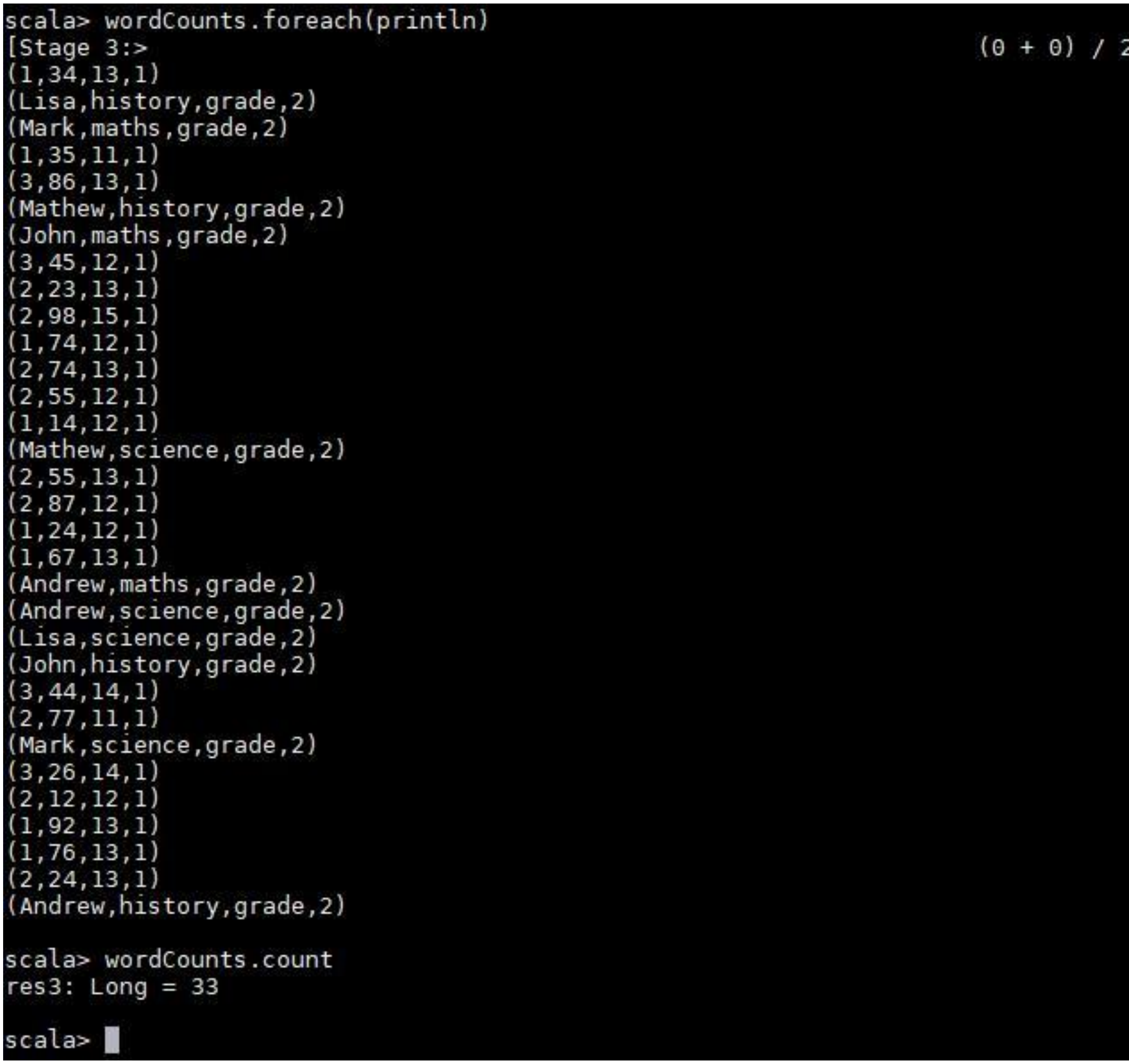
```
scala> val words = studData.flatMap(_.split("-"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:

scala> val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at

scala> wordCounts.foreach(println)
[Stage 3:>                                                          (0 + 0) / 2](1,23,
(1,34,13,1)
(Lisa,history,grade,2)
(Mark,maths,grade,2)
(1,35,11,1)
(3,86,13,1)
(Mathew,history,grade,2)
(John,maths,grade,2)
(3,45,12,1)
(2,23,13,1)
(2,98,15,1)
(1,74,12,1)
(2,74,13,1)
(2,55,12,1)
(1,14,12,1)
(Mathew,science,grade,2)
(2,55,13,1)
(2,87,12,1)
(1,24,12,1)
(1,67,13,1)
(Andrew,maths,grade,2)
(Andrew,science,grade,2)
(Lisa,science,grade,2)
(John,history,grade,2)
(3,44,14,1)
(2,77,11,1)
(Mark,science,grade,2)
(3,26,14,1)
(2,12,12,1)
(1,92,13,1)
(1,76,13,1)
(2,24,13,1)
(Andrew,history,grade,2)

scala> 
```

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.

Solution:

– To obtain the count of total number of words present in the document, we simply run the ".count" command i.e.

```
scala> wordCounts.count
```


```
scala> wordCounts.foreach(println)
[Stage 3:>                                                          (0 + 0) / 2
(1,34,13,1)
(Lisa,history,grade,2)
(Mark,maths,grade,2)
(1,35,11,1)
(3,86,13,1)
(Mathew,history,grade,2)
(John,maths,grade,2)
(3,45,12,1)
(2,23,13,1)
(2,98,15,1)
(1,74,12,1)
(2,74,13,1)
(2,55,12,1)
(1,14,12,1)
(Mathew,science,grade,2)
(2,55,13,1)
(2,87,12,1)
(1,24,12,1)
(1,67,13,1)
(Andrew,maths,grade,2)
(Andrew,science,grade,2)
(Lisa,science,grade,2)
(John,history,grade,2)
(3,44,14,1)
(2,77,11,1)
(Mark,science,grade,2)
(3,26,14,1)
(2,12,12,1)
(1,92,13,1)
(1,76,13,1)
(2,24,13,1)
(Andrew,history,grade,2)

scala> wordCounts.count
res3: Long = 33

scala> 
```

Task 2:

Problem Statement 1:

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

Solution:

- To create a tuple from the contents of the text file and display the contents of the tuple, we use the following commands

  scala> val file = studData.map(x=> x.split(",")) scala> file.collect

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

Solution:

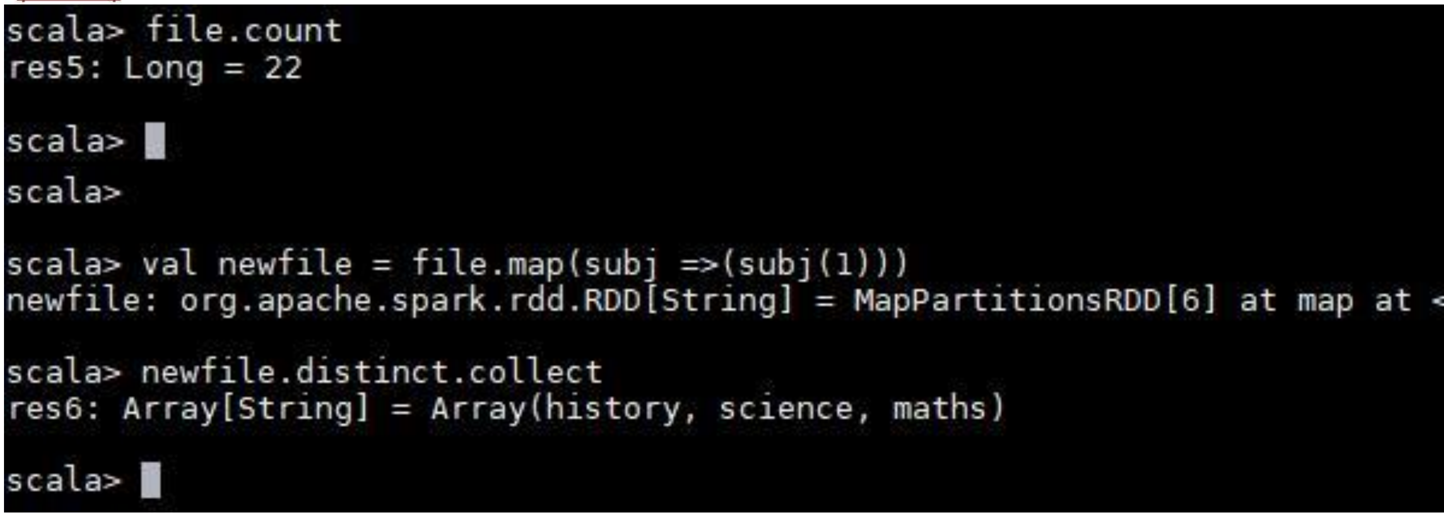
- To count total number of rows present we run the below command *scala> file.count*

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

Solution:

- To find the distinct number of subjects in the school we use RDD Action i.e. “distinct”, this removes all the duplicate records

  scala> val newfile = file.map(subj =>(subj(1)))
- This command will map the subject part of the tuple “file” *scala> newfile.distinct.collect*
- This will remove all the duplicates and return only unique values



```
scala> file.count
res5: Long = 22

scala> 
scala>

scala> val newfile = file.map(subj =>(subj(1)))
newfile: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[6] at map at <

scala> newfile.distinct.collect
res6: Array[String] = Array(history, science, maths)

scala> 
```

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

Solution:

– To accomplish this task, we split the text file by "," separator *sccala> val studFile = studData.map(x=> x.split(","))*

– Next, we filter the variable "studFile" where marks are equal to "55" and student name is "Mathew"

scala> val studFilter = studFile.filter(x => (x(3).toInt.equals(55)&& x(0).contains("Mathew"))).collect

Problem Statement 2:

1. What is the count of students per grade in the school?

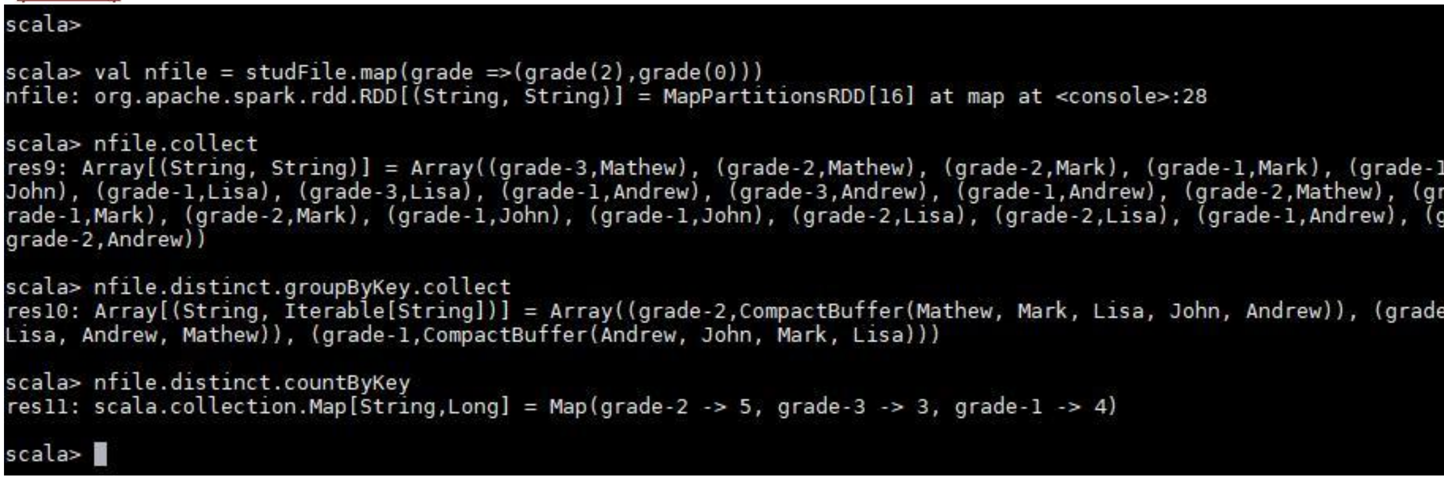
Solution:

To count the students per grade in the school, we have to perform the following tasks:

– We map the tuples with only two values, i.e. grade(2) which indicates the grades and grade(0) which indicates the name of the student *scala> val nfile = studFile.map(grade =>(grade(2),grade(0))) scala> nfile.collect*

- We perform groupByKey, to know how many students fall under each grade

  scala> nfile.distinct.groupByKey.collect
- We find the distinct values and calculate a count of it *scala> nfile.distinct.countByKey*



```
scala>

scala> val nfile = studFile.map(grade =>(grade(2),grade(0)))
nfile: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[16] at map at <console>:28

scala> nfile.collect
res9: Array[(String, String)] = Array((grade-3,Mathew), (grade-2,Mathew), (grade-2,Mark), (grade-1,Mark), (grade-1
John), (grade-1,Lisa), (grade-3,Lisa), (grade-1,Andrew), (grade-3,Andrew), (grade-1,Andrew), (grade-2,Mathew), (gr
rade-1,Mark), (grade-2,Mark), (grade-1,John), (grade-1,John), (grade-2,Lisa), (grade-2,Lisa), (grade-1,Andrew), (g
grade-2,Andrew))

scala> nfile.distinct.groupByKey.collect
res10: Array[(String, Iterable[String])] = Array((grade-2,CompactBuffer(Mathew, Mark, Lisa, John, Andrew)), (grade
Lisa, Andrew, Mathew)), (grade-1,CompactBuffer(Andrew, John, Mark, Lisa)))

scala> nfile.distinct.countByKey
res11: scala.collection.Map[String,Long] = Map(grade-2 -> 5, grade-3 -> 3, grade-1 -> 4)

scala> █
```

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

<u>Solution:</u>

To calculate average, we need to follow the steps below:

- We extract the contents of the text file to spark and then split the tuple based on "," separator and extract only 3 fields i.e. student_name, subject and marks

  scala> val avgEachStud = studData.map(x => (x.toString.split(","))).map(x => ((x(0),x(1)),x(3).toInt))
- We group them by adding 1 to each of the row, calculate sum of them and sort

them based on the key, this is to calculate the counts of number of records. In the same we calculate the total of all marks for each student and subject

scala> val totalLength = avgEachStud.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()

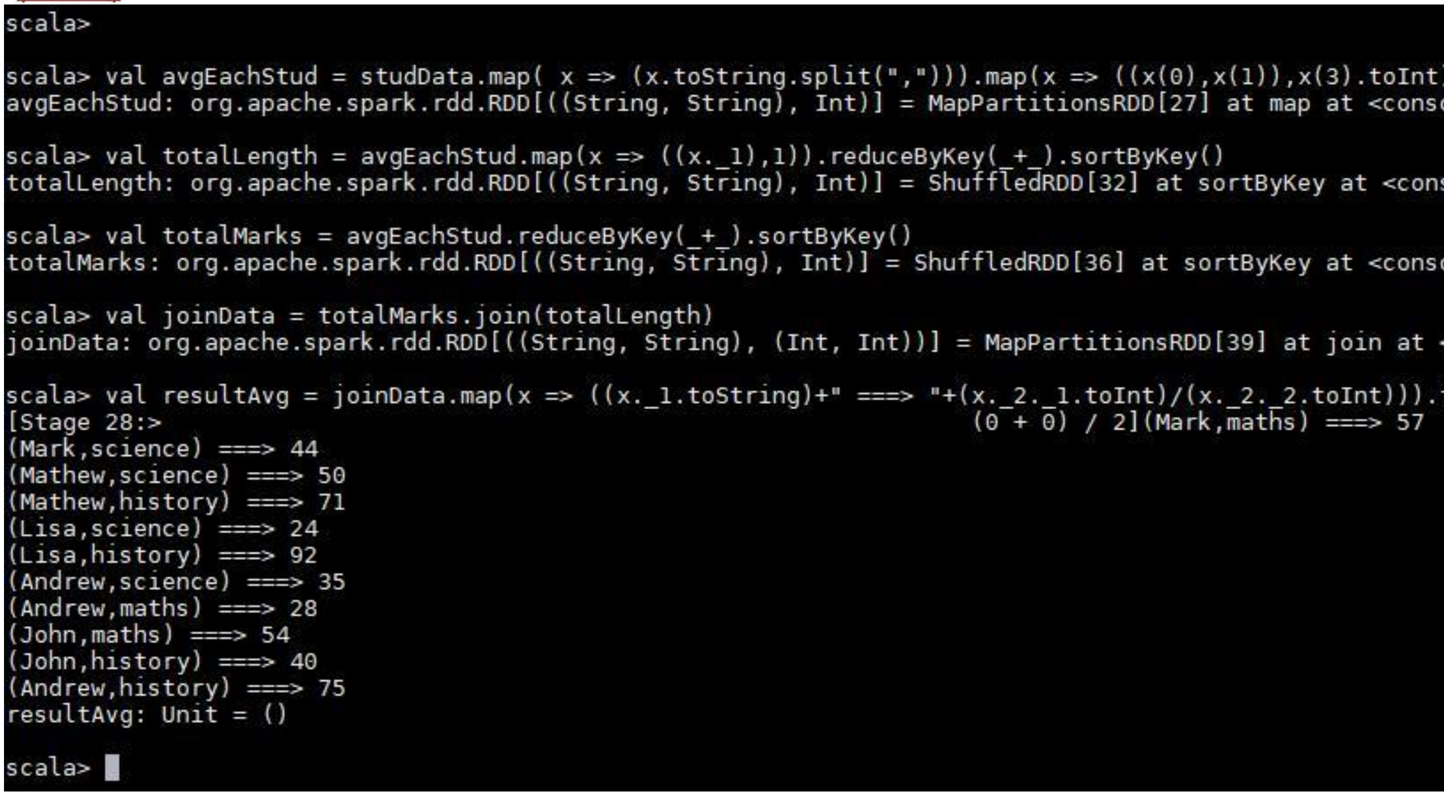
scala> val totalMarks = avgEachStud.reduceByKey(_+_).sortByKey()

– We now join both total marks and counts

scala> val joinData = totalMarks.join(totalLength)

– Finally, we calculate average, as we know average = sum/counts. We could compute the same

scala> val resultAvg = joinData.map(x => ((x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)



```
scala>

scala> val avgEachStud = studData.map( x => (x.toString.split(","))).map(x => ((x(0),x(1)),x(3).toInt
avgEachStud: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[27] at map at <cons

scala> val totalLength = avgEachStud.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
totalLength: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[32] at sortByKey at <cons

scala> val totalMarks = avgEachStud.reduceByKey(_+_).sortByKey()
totalMarks: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[36] at sortByKey at <cons

scala> val joinData = totalMarks.join(totalLength)
joinData: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[39] at join at

scala> val resultAvg = joinData.map(x => ((x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).
[Stage 28:>                                                   (0 + 0) / 2](Mark,maths) ===> 57
(Mark,science) ===> 44
(Mathew,science) ===> 50
(Mathew,history) ===> 71
(Lisa,science) ===> 24
(Lisa,history) ===> 92
(Andrew,science) ===> 35
(Andrew,maths) ===> 28
(John,maths) ===> 54
(John,history) ===> 40
(Andrew,history) ===> 75
resultAvg: Unit = ()

scala> █
```

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

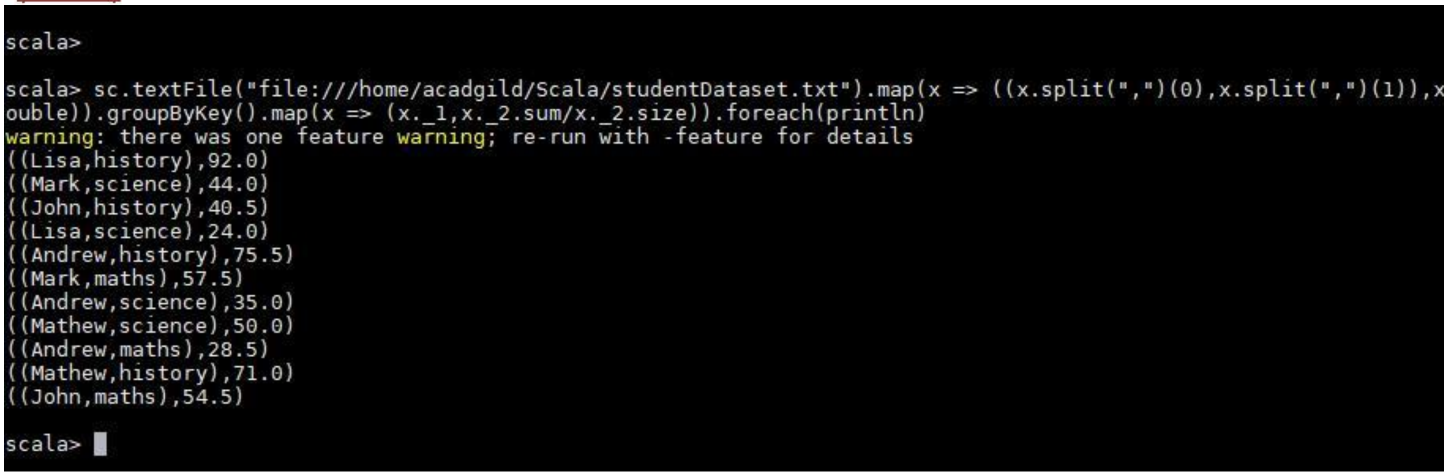
Solution:

To find the average score of students in each subject for all grades, the following steps have been used:

- Creating an RDD from textFile
- Select name & subject as key, marks as value
- Convert marks into double datatype just to avoid any truncation after decimal
- groupByKey returns a tuple of ((name, subject), marks)
- Apply map function & select name, subject as first value of outer tuple & marks as second value
- Add the marks for respective subject & for each distinct subject fetch the count of marks as x._2.size to divide the sum & find the average on it.
- Here group key is student & subject *scala>*

  sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x.split(",")(3)toDouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).sortByKey().foreach(println)

  This gives use the average of each student in each subject for all grades.



```
scala>

scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x
ouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).foreach(println)
warning: there was one feature warning; re-run with -feature for details
((Lisa,history),92.0)
((Mark,science),44.0)
((John,history),40.5)
((Lisa,science),24.0)
((Andrew,history),75.5)
((Mark,maths),57.5)
((Andrew,science),35.0)
((Mathew,science),50.0)
((Andrew,maths),28.5)
((Mathew,history),71.0)
((John,maths),54.5)

scala> 
```

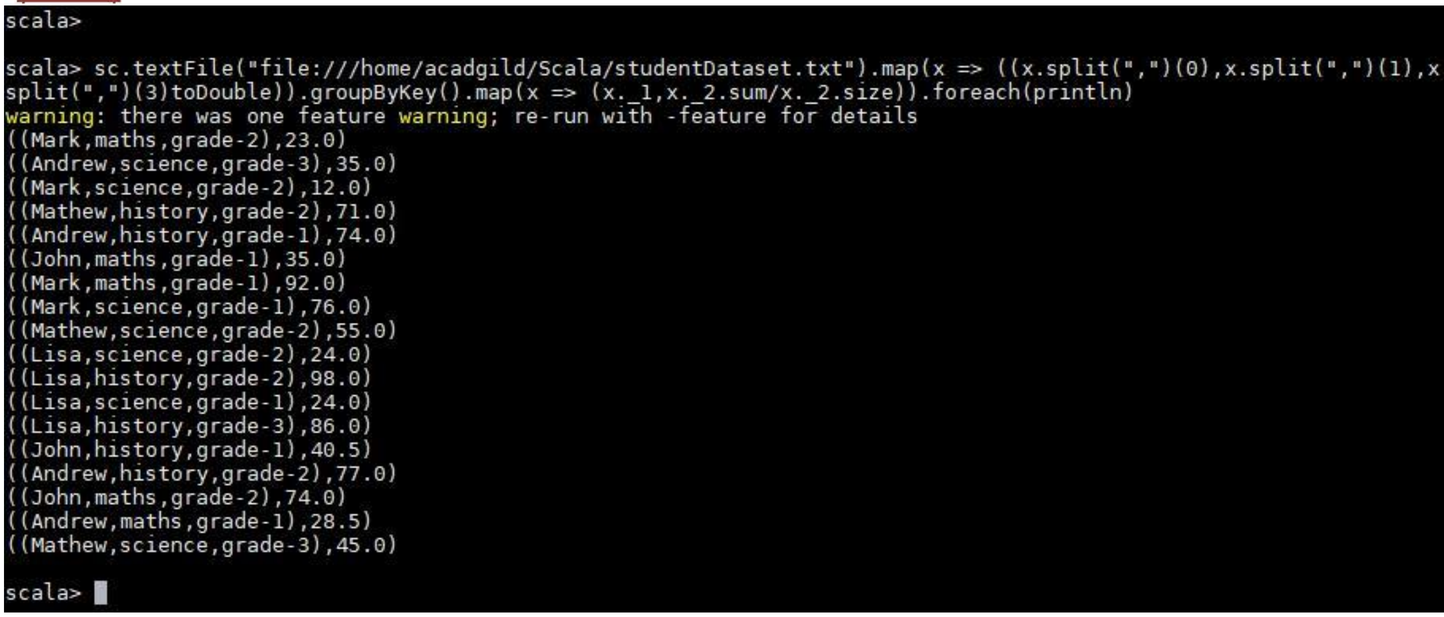
4. What is the average score of students in each subject per grade?
<u>Solution:</u>

To find the average score of students in each subject for each grade, the following steps have been used:

- Creating an RDD from textFile.
- Select name, subject & grade as key, marks as value.
- Convert marks into double datatype just to avoid any truncation after decimal.
- groupByKey returns a tuple of ((name, subject, grade), marks).
- Apply map function & select name, subject & grade as first value of outer tuple & marks as second value.
- Add the marks for respective subject, grade & for each distinct subject, grade fetch the count of marks as x._2.size to divide the sum & find the average on it.
- Here group key is name, subject & grade.

This gives use the average of each student in each subject for each grade.

*scala>
sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0),x.split(",")(1),x.split(",")(2)),x.split(",")(3)toDouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).foreach(println)*



```
scala>

scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0),x.split(",")(1),x
split(",")(3)toDouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).foreach(println)
warning: there was one feature warning; re-run with -feature for details
((Mark,maths,grade-2),23.0)
((Andrew,science,grade-3),35.0)
((Mark,science,grade-2),12.0)
((Mathew,history,grade-2),71.0)
((Andrew,history,grade-1),74.0)
((John,maths,grade-1),35.0)
((Mark,maths,grade-1),92.0)
((Mark,science,grade-1),76.0)
((Mathew,science,grade-2),55.0)
((Lisa,science,grade-2),24.0)
((Lisa,history,grade-2),98.0)
((Lisa,science,grade-1),24.0)
((Lisa,history,grade-3),86.0)
((John,history,grade-1),40.5)
((Andrew,history,grade-2),77.0)
((John,maths,grade-2),74.0)
((Andrew,maths,grade-1),28.5)
((Mathew,science,grade-3),45.0)

scala> 
```

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

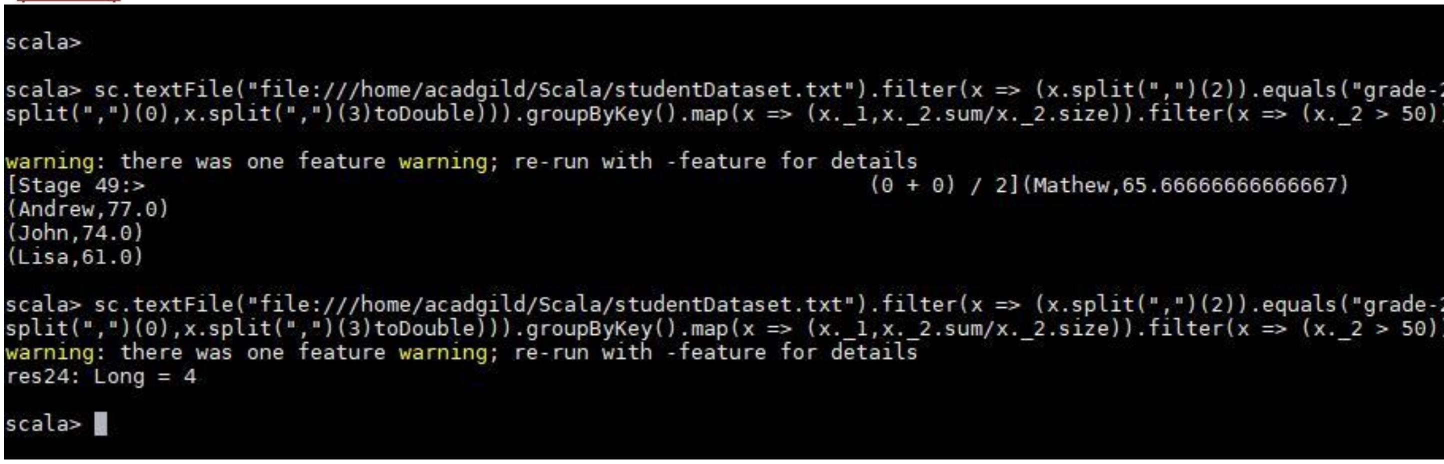
Solution:

To find the students in grade 2 who have average score greater than 50, the following steps have been used:

- Filter the records where grade equals grade-2 & then split the record where key is name value is marks
- Next, group by Name to find out average of marks by calculating total marks for each name & dividing by count of times marks were awarded to each student
- Finally, apply filter & count whoever has average marks greater than 50

```
scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")
.filter(x => (x.split(",")(2)).equals("grade-2"))
.map(x => ((x.split(",")(0),x.split(",")(3)toDouble))).groupByKey()
.map(x => (x._1,x._2.sum/x._2.size)).filter(x => (x._2 > 50))
.foreach(println)
```

```
scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")
.filter(x => (x.split(",")(2)).equals("grade-2"))
.map(x => ((x.split(",")(0),x.split(",")(3)toDouble))).groupByKey()
.map(x => (x._1,x._2.sum/x._2.size)).filter(x => (x._2 > 50)).count()
```



```
scala>

scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").filter(x => (x.split(",")(2)).equals("grade-
split(",")(0),x.split(",")(3)toDouble))).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).filter(x => (x._2 > 50)

warning: there was one feature warning; re-run with -feature for details
[Stage 49:>                                                    (0 + 0) / 2](Mathew,65.66666666666667)
(Andrew,77.0)
(John,74.0)
(Lisa,61.0)

scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").filter(x => (x.split(",")(2)).equals("grade-
split(",")(0),x.split(",")(3)toDouble))).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).filter(x => (x._2 > 50)
warning: there was one feature warning; re-run with -feature for details
res24: Long = 4

scala> █
```

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

Solution:

Average score per student_name across all grades:

- Map the 1st column (student_name) and the addition of both the marks as one column
- Using the result from above, group the data by the key (student_name)
- Using the result from above, map the 1st column and the result of the following as the 2nd column:
  - Sum of the key-grouped marks divided by the count of the marks columns (2 columns)

Average score per student_name per grade

- Map the 1st and 3rd column (student_name and grade) and the addition of both the marks as one column
- The rest is same as above.

Comparing the results: (Using the Hint given)

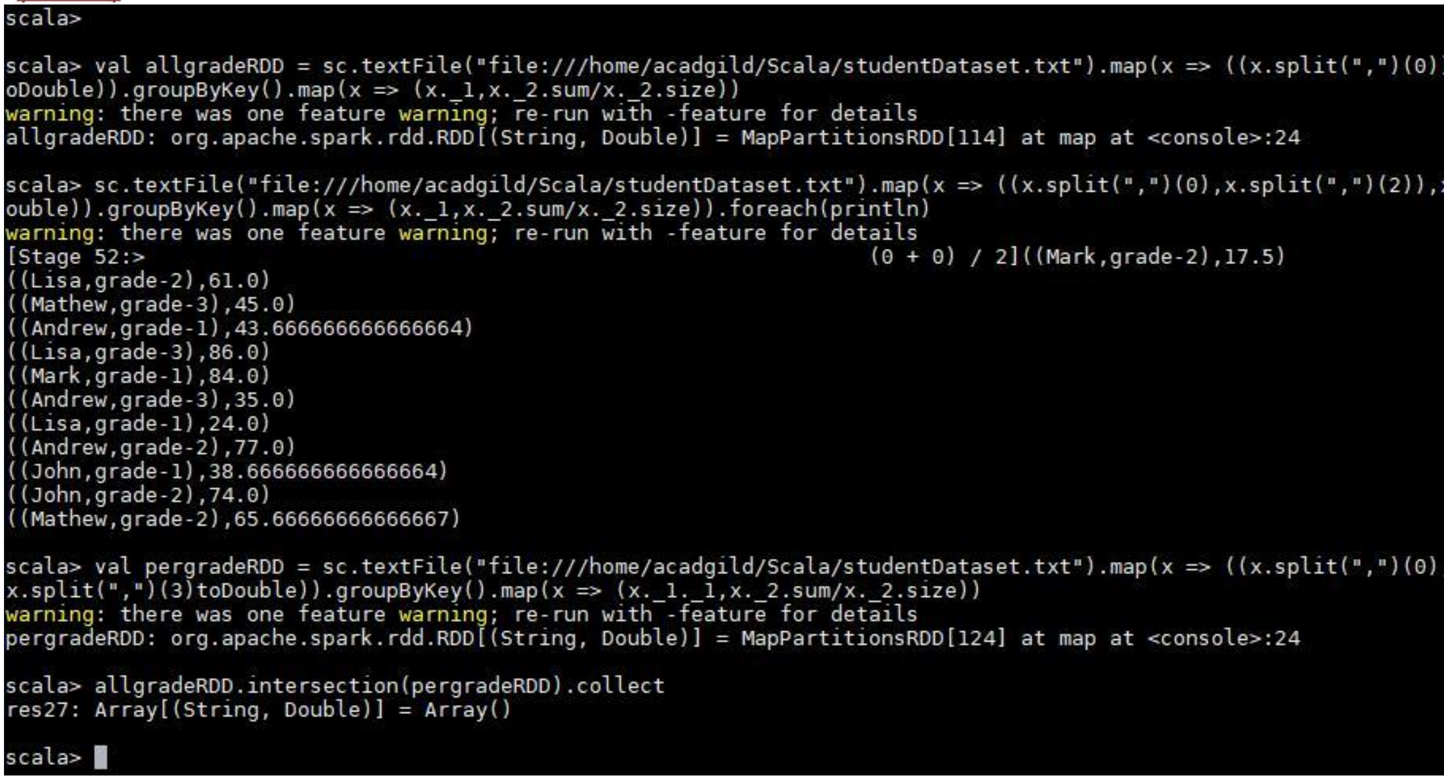
- Finding the common data in both the lists by using the intersection property

Result shows that there is no student that fulfils the criteria given above.

```
scala> val allgradeRDD =
        sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")
        .map(x => (x.split(",")(0)),x.split(",")(3)toDouble)).groupByKey()
        .map(x => (x._1,x._2.sum/x._2.size))

scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")
        .map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3)toDouble))
        .groupByKey().map(x => (x._1,x._2.sum/x._2.size)).foreach(println)

scala> val pergradeRDD =
        sc.textFile("file:///home/acadgild/Scala/studentDataset.txt")
        .map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3)toDouble))
```

.groupByKey().map(x => (x._1._1,x._2.sum/x._2.size))

scala> allgradeRDD.intersection(pergradeRDD).collect



```
scala>

scala> val allgradeRDD = sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0)
oDouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size))
warning: there was one feature warning; re-run with -feature for details
allgradeRDD: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[114] at map at <console>:24

scala> sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),
ouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size)).foreach(println)
warning: there was one feature warning; re-run with -feature for details
[Stage 52:>                                                  (0 + 0) / 2]((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Lisa,grade-1),24.0)
((Andrew,grade-2),77.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)

scala> val pergradeRDD = sc.textFile("file:///home/acadgild/Scala/studentDataset.txt").map(x => ((x.split(",")(0)
x.split(",")(3)toDouble)).groupByKey().map(x => (x._1._1,x._2.sum/x._2.size))
warning: there was one feature warning; re-run with -feature for details
pergradeRDD: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[124] at map at <console>:24

scala> allgradeRDD.intersection(pergradeRDD).collect
res27: Array[(String, Double)] = Array()

scala> █
```