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.

```
Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_151)
Type in expressions to have them evaluated.
Type :help for more information.

scala> var file = sc.textFile("/user/acadgild/hadoop/word-count.txt")
file: org.apache.spark.rdd.RDD[String] = /user/acadgild/hadoop/word-count.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val lines = file.count()
lines: Long = 2

scala>
```

Create an RDD from the a sample text file by using the

sc.textFile("path/on/hdfs/to/file.txt")

Count the number of lines in the file using

val lines = file.count()

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

```
scala> var file = sc.textFile("/user/acadgild/hadoop/word-count.txt")
file: org.apache.spark.rdd.RDD[String] = /user/acadgild/hadoop/word-count.txt MapPartitionsRDD[3] at textFile at <console>:24

scala> val words = file.flatMap(line => line.split(" "))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at flatMap at <console>:26

scala> val wordCount = words.count()
wordCount: Long = 44

scala>
```

Once the RDD is created from the text file,

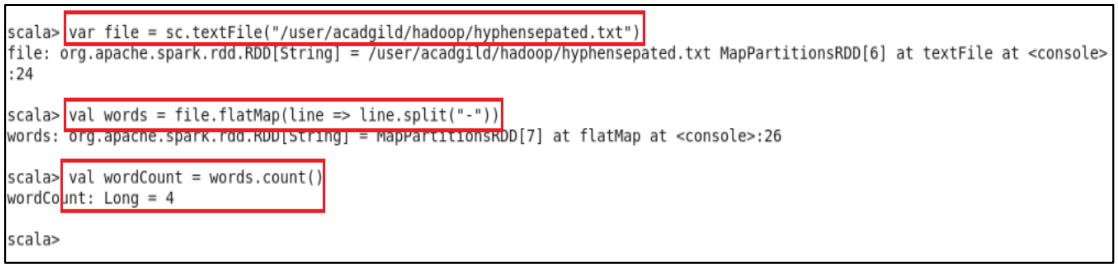
All the words from the text file are split based on the space character.

val words = file.flatMap(line => line.split(" "))

After splitting, the number of the words are counted.

val wordCount = words.count()

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.



For the purpose of this problem statement, we are creating a new RDD from a file in which words are separated by a hyphen.

var file = sc.textFile("/user/acadgild/hadoop/hyphensepated.txt")

Now, since the separator is a hyphen, the file is split based on the '-' character.

val words = file.flatMap(line => line.split("-"))

The number of words are then counted.

val wordCount = words.count()

Task 2:

Problem Statement 1:

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

```
scala>

scala> val rdd = sc.textFile("/user/acadgild/hadoop/19_Dataset.txt")
rdd: org.apache.spark.rdd.RDD[String] = /user/acadgild/hadoop/19_Dataset.txt MapPartitionsRDD[16] at textFile at <console>:24

scala> val a = rdd.map{ x =>
     | val splitted = x.split(",").toList
     | (splitted(0),splitted(1),splitted(2),splitted(3).toInt,splitted(4).toInt)
     | }
a: org.apache.spark.rdd.RDD[(String, String, String, Int, Int)] = MapPartitionsRDD[17] at map at <console>:26

scala> rdd.collect
res2: Array[String] = Array(Mathew,science,grade-3,45,12, Mathew,history,grade-2,55,13, Mark,maths,grade-2,23,13, Mark,scienc
e,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, Mathe
w,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, And
rew,history,grade-2,77,11)

scala> a.collect
res3: Array[(String, String, String, Int, Int)] = Array((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,g
rade-1,74,12), (Mathew,science,grade-2,55,12), (Mathew,history,grade-2,87,12), (Mark,maths,grade-1,92,13), (Mark,science,grad
e-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))

scala>
```

Creating an RDD from the provided dataset. The given file is loaded onto HDFS first and the HDFS path is given here.

val rdd = sc.textFile("/user/acadgild/hadoop/19_Dataset.txt")

Creating an tupled RDD by first splitting the file based on comma separator and then reading each record into a list.

val a = rdd.map{ x =>
val splitted = x.split(",").toList
(splitted(0),splitted(1),splitted(2),splitted(3).toInt,splitted(4).toInt)
}

When both the original and the tupled rdd are displayed, we can see the difference. The RDD ‘a’ consists of Tuples, while the original one consits on a list of comma separated values.

rdd.collect

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

```
scala> a.collect
res3: Array[(String, String, String, Int, Int)] = Array((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,g
rade-1,74,12), (Mathew,science,grade-2,55,12), (Mathew,history,grade-2,87,12), (Mark,maths,grade-1,92,13), (Mark,science,grad
e-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))

scala>

scala> a.count
res4: Long = 22
```

Command used foe counting the number of rows in an RDD is count.

a.count

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

```
scala>

scala> val distsubjects = rdd.map( .split(",")).map(c => c(1)).distinct().count
distsubjects: Long = 3

scala>
```

The rdd created is split based on the comma separator and then 2nd column I.e the 'subject' column is extracted in order to map into a new RDD.

The distinct operator is applied in the 'subject' column and the count of the distinct subjects is captured.

val distcubjects = tupledrdd.map(_.split(",")).map(c => c(1)).distinct().count

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

```
scala>

scala> val ps1t4 = a.filter(p => p._1 == "Mathew" && p._4 == 55).count
ps1t4: Long = 2

scala>

scala>
```

The RDD created earlier is used. The column number 1 and 4 pertaining to name and marks respectively are compared to the required values and the same is filtered out. The count of the records matching this filter is then captured.

val ps1t4 = a.filter(p => p._1 == "Mathew" && p._4 == 55).count

Problem Statement 2:

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

```
scala> val x = rdd.map(_.split(",")).map(c => c(2))
x: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[39] at map at <console>:26

scala> val num = x.map((_, 1)).reduceByKey(_ + _)
num: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[41] at reduceByKey at <console>:28

scala> num.collect
res6: Array[(String, Int)] = Array((grade-3,4), (grade-1,9), (grade-2,9))

scala>

scala>
```

A new RDD is containing the 3rd column, 'grade' is created.

val x = rdd.map(_.split(",")).map(c => c(2))

Then the number of times each grade is repeated is counted using the map and reduce method.

val num = x.map((_, 1)).reduceByKey(_ + _)

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

```
scala> val avg_each_stu = student.map( x => (x.toString.split(","))).map(x => ((x(0),x(2)),x(3).toInt))
avg_each_stu: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[49] at map at <console>:26

scala> val total_len = avg_each_stu.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
total_len: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[52] at sortByKey at <console>:28

scala> total_len.collect
res33: Array[((String, String), Int)] = Array(((Andrew,grade-1),3), ((Andrew,grade-2),1), ((Andrew,grade-3),2), ((John,grade-
1),3), ((John,grade-2),1), ((Lisa,grade-1),1), ((Lisa,grade-2),2), ((Lisa,grade-3),1), ((Mark,grade-1),2), ((Mark,grade-2),2)
, ((Mathew,grade-2),3), ((Mathew,grade-3),1))

scala> val total_marks = avg_each_stu.reduceByKey(_+_).sortByKey()
total_marks: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[54] at sortByKey at <console>:28

scala> total_marks.collect
res34: Array[((String, String), Int)] = Array(((Andrew,grade-1),131), ((Andrew,grade-2),77), ((Andrew,grade-3),70), ((John,gr
ade-1),116), ((John,grade-2),74), ((Lisa,grade-1),24), ((Lisa,grade-2),122), ((Lisa,grade-3),86), ((Mark,grade-1),168), ((Mar
k,grade-2),35), ((Mathew,grade-2),197), ((Mathew,grade-3),45))

scala> val join_data = total_marks.join(total_len)
join_data: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[57] at join at <console>:32

scala> val result_avg = join_data.map(x => ( ( x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)
(Lisa,grade-1) ===> 24
(Mark,grade-2) ===> 17
(Lisa,grade-2) ===> 61
(Mathew,grade-3) ===> 45
(Andrew,grade-2) ===> 77
(Andrew,grade-1) ===> 43
(Lisa,grade-3) ===> 86
(John,grade-1) ===> 38
(John,grade-2) ===> 74
(Mark,grade-1) ===> 84
(Andrew,grade-3) ===> 35
(Mathew,grade-2) ===> 65
result_avg: Unit = ()

scala>
```

Multiple RDDs are created by performing transformations.

First the data from csv file is read into an RDD as :

val student = sc.textFile("/user/acadgild/hadoop/19_Dataset.txt")

Since the average of each individual student with respect to grade has to be found out, we can extract the required columns into an RDD.

val avg_each_stu = student.map(x => (x.toString.split(","))).map(x => ((x(0),x(2)),x(3).toInt))

To calculate average, there is no direct function, so we create two RDDs one for the number of student has occured in the list and the other for the sum of marks for each student.

val total_len = avg_each_stu.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
val total_marks = avg_each_stu.reduceByKey(_+_).sortByKey()

Both these RDDs are joined to put together the count and sum of each student.

val join_data = total_marks.join(total_len)

From the joined RDD, we can perform the average and display the results.

val result_avg = join_data.map(x => ((x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)

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

```
scala>

scala> val avg_each_sub = student.map( x => (x.toString.split(","))).map(x => (x(1),x(3).toInt))
avg_each_sub: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[82] at map at <console>:26

scala> avg_each_sub.collect
res45: Array[(String, Int)] = Array((science,45), (history,55), (maths,23), (science,76), (history,14), (maths,74), (science,
24), (history,86), (maths,34), (science,26), (history,74), (science,55), (history,87), (maths,92), (science,12), (history,67)
, (maths,35), (science,24), (history,98), (maths,23), (science,44), (history,77))

scala> val total_len = avg_each_sub.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
total_len: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[83] at sortByKey at <console>:28

scala> total_len.collect
res46: Array[(String, Int)] = Array((history,8), (maths,6), (science,8))

scala> val total_marks = avg_each_sub.reduceByKey(_+_).sortByKey()
total_marks: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[87] at sortByKey at <console>:28

scala> total_marks.collect
res47: Array[(String, Int)] = Array((history,558), (maths,281), (science,306))

scala> val join_data = total_marks.join(total_len)
join_data: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[90] at join at <console>:32

scala> join_data.collect
res48: Array[(String, (Int, Int))] = Array((maths,(281,6)), (history,(558,8)), (science,(306,8)))

scala> val result_avg = join_data.map(x => ( ( x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)
maths ===> 46
history ===> 69
science ===> 38
result_avg: Unit = ()
```

The same ‘student’ RDD is reused here.

Since the average of students in each subject across all grades, the subject and the corresponding marks from the Student RDD.

val avg_each_sub = student.map(x => (x.toString.split(","))).map(x => (x(1),x(3).toInt))

The number of times each subject is repeated and sum of marks for each subject are put into 2 RDDs.

val total_len = avg_each_sub.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
val total_marks = avg_each_sub.reduceByKey(_+_).sortByKey()

Both these RDDs are joined to put together the count and sum of each student.

val join_data = total_marks.join(total_len)

From the joined RDD, we can perform the average and display the results.

val result_avg = join_data.map(x => ((x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)

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

```
scala> val avg_sub_grd = student.map( x => (x.toString.split(","))).map(x => ((x(1),x(2)),x(3).toInt))
avg_sub_grd: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[106] at map at <console>:26

scala> avg_sub_grd.collect
res55: Array[((String, String), Int)] = Array(((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))

scala> val total_len = avg_sub_grd.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
total_len: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[109] at sortByKey at <console>:28

scala> val total_marks = avg_sub_grd.reduceByKey(_+_).sortByKey()
total_marks: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[111] at sortByKey at <console>:28

scala> total_marks.collect
res56: Array[((String, String), Int)] = Array(((history,grade-1),155), ((history,grade-2),317), ((history,grade-3),86), ((mat
hs,grade-1),184), ((maths,grade-2),97), ((science,grade-1),100), ((science,grade-2),91), ((science,grade-3),115))

scala> val join_data = total_marks.join(total_len)
join_data: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[114] at join at <console>:32

scala> join_data.collect
res57: Array[((String, String), (Int, Int))] = Array(((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 result_avg = join_data.map(x => ( ( x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)
(history,grade-2) ===> 79
(history,grade-3) ===> 86
(maths,grade-1) ===> 46
(science,grade-3) ===> 38
(science,grade-1) ===> 50
(science,grade-2) ===> 30
(history,grade-1) ===> 51
(maths,grade-2) ===> 48
result_avg: Unit = ()
```

The same ‘student’ RDD is reused here.

Since the average of students in each subject per grade, the subject, grade and the corresponding marks from the Student RDD.

val avg_sub_grd = student.map(x => (x.toString.split(","))).map(x => ((x(1),x(2)),x(3).toInt))

The number of times each subject,grade is repeated and sum of marks for each subject are put into 2 RDDs.

val total_len = avg_sub_grd.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
val total_marks = avg_sub_grd.reduceByKey(_+_).sortByKey()

Both these RDDs are joined to put together the count and sum of each student.

val join_data = total_marks.join(total_len)

From the joined RDD, we can perform the average and display the results.

val result_avg = join_data.map(x => ((x._1.toString)+" ===> "+(x._2._1.toInt)/(x._2._2.toInt))).foreach(println)

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

```
scala>

scala> val avg_abv50 = student.map( x => (x.toString.split(","))).filter(x => x(2) == "grade-2").map(x => ((x(0),x(2)),x(3).t
oInt))
avg_abv50: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[39] at map at <console>:26

scala> avg_abv50.collect
res6: Array[((String, String), Int)] = Array(((Mathew,grade-2),55), ((Mark,grade-2),23), ((John,grade-2),74), ((Mathew,grade-
2),55), ((Mathew,grade-2),87), ((Mark,grade-2),12), ((Lisa,grade-2),24), ((Lisa,grade-2),98), ((Andrew,grade-2),77))

scala> val total_len = avg_abv50.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
total_len: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[42] at sortByKey at <console>:28

scala> val total_marks = avg_abv50.reduceByKey(_+_).sortByKey()
total_marks: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[44] at sortByKey at <console>:28

scala> val join_data = total_marks.join(total_len)
join_data: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[47] at join at <console>:32

scala> join_data.collect
res7: Array[((String, String), (Int, Int))] = Array(((Mark,grade-2),(35,2)), ((Lisa,grade-2),(122,2)), ((Andrew,grade-2),(77,
1)), ((John,grade-2),(74,1)), ((Mathew,grade-2),(197,3)))

scala> val result_avg = join_data.map(x => ( ( x._1.toString),(x._2._1.toInt)/(x._2._2.toInt))).filter(x => x._2 > 50).foreac
h(println)
((Lisa,grade-2),61)
((Andrew,grade-2),77)
((John,grade-2),74)
((Mathew,grade-2),65)
result_avg: Unit = ()

scala>
```

The same ‘student’ RDD is reused here.

Since the average of students in the 2nd grade are required, we filter only grade students from the Students RDD.

val avg_abv50 = student.map(x => (x.toString.split(","))).filter(x => x(2) == "grade-2").map(x => ((x(0),x(2)),x(3).toInt))

The number of times each name,grade is repeated and sum of marks for each subject are put into 2 RDDs.

val total_len = avg_abv50.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
val total_marks = avg_abv50.reduceByKey(_+_).sortByKey()

Both these RDDs are joined to put together the count and sum of each student.

val join_data = total_marks.join(total_len)

From the joined RDD, we can perform the average. The average greater than 50 is filtered and the results are displayed.

val result_avg = join_data.map(x =>
((x._1.toString),(x._2._1.toInt)/(x._2._2.toInt))).filter(x => x._2 > 50).foreach(println)

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

2 RDDs are created first and then an intersection opoeration is performed on both the RDDs.

```
scala>

scala> val avg_stu = student.map( x => (x.toString.split(","))).map(x => (x(0),x(3).toInt))
avg_stu: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[88] at map at <console>:26

scala> avg_stu.collect
res15: Array[(String, Int)] = Array((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), (L
isa,98), (Andrew,23), (Andrew,44), (Andrew,77))

scala> val total_len1 = avg_stu.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
total_len1: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[91] at sortByKey at <console>:28

scala> val total_marks1 = avg_stu.reduceByKey(_+_).sortByKey()
total_marks1: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[93] at sortByKey at <console>:28

scala> val join_data1 = total_marks1.join(total_len1)
join_data1: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[96] at join at <console>:32

scala> val result_avg1 = join_data1.map(x => ( ( x._1.toString),(x._2._1.toInt)/(x._2._2.toInt)))
result_avg1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[97] at map at <console>:34

scala> result_avg1.collect
res16: Array[(String, Int)] = Array((Mark,50), (Andrew,46), (Mathew,60), (John,47), (Lisa,58))
```

RDD1 (result_avg1): Average score per student_name across all grades.

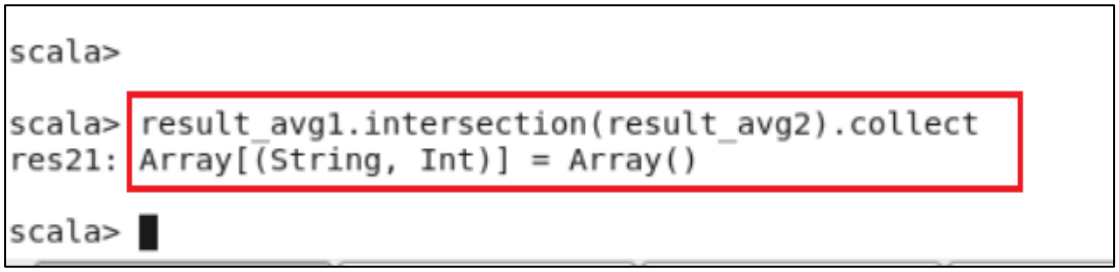
In a same way as the above Problem statement, the student name and the corresponding marks are extracted, and average is calculated for each student name.

```
scala> val avg_each_stu = student.map( x => (x.toString.split(","))).map(x => ((x(0),x(2)),x(3).toInt))
avg_each_stu: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[116] at map at <console>:26

scala> avg_each_stu.collect
res19: Array[((String, String), Int)] = Array(((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 total_len2 = avg_each_stu.map(x => ((x._1),1)).reduceByKey(_+_).sortByKey()
total_len2: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[119] at sortByKey at <console>:28

scala> val total_marks2 = avg_each_stu.reduceByKey(_+_).sortByKey()
total_marks2: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[121] at sortByKey at <console>:28

scala> val join_data2 = total_marks2.join(total_len2)
join_data2: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[124] at join at <console>:32

scala> val result_avg2 = join_data2.map(x => ( ( x._1.toString),(x._2._1.toInt)/(x._2._2.toInt)))
result_avg2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[125] at map at <console>:34

scala> result_avg2.collect
res20: Array[(String, Int)] = Array(((Lisa,grade-1),24), ((Mark,grade-2),17), ((Lisa,grade-2),61), ((Mathew,grade-3),45), ((A
ndrew,grade-2),77), ((Andrew,grade-1),43), ((Lisa,grade-3),86), ((John,grade-1),38), ((John,grade-2),74), ((Mark,grade-1),84)
, ((Andrew,grade-3),35), ((Mathew,grade-2),65))

scala>
```

RDD 2(result_avg1): Average score per student_name per grade.

Here, both the student name and the grade are extracted for calculating average in a similar fashion.

Instead of converting the type to “Unit” and displaying the results, the ‘result_avg’ is maintained as an RDD type.



Intersection operation returns elements that are common b/w both RDDs. On performing an intersection on both the resultant RDDs, we can see that an empty Array is returned meaning that there is no student that satisfies the specified criteria.