SPARK ASSIGNMENT 17.2

Given a dataset of college students as a text file (name, subject, grade, marks) :

Name = word(0)

Subject = word(1)

Grade = word(2)

Marks = word()

Problem Statement

//starting the hadoop 2.0 in cluster

[acadgild@localhost ~]$ start-dfs.sh

//creating and editing the text file

[acadgild@localhost ~]$ gedit collegestudent.txt

collegestudent.txt

//browsing through the text file

[acadgild@localhost ~]$ cat collegestudent.txt

//copying the file to hdfs from nfs

[acadgild@localhost ~]$ hadoop fs -put collegestudent.txt /user/acadgild/spark/

//creating a user defined HDFS directory

[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/spark/

//browsing through the hdfs file

[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/spark/collegestudent.txt

//displaying the hdfs file contents

Mathew,Science,grade-1,55

Mathew,Social Science,grade-1,55

Mathew,Medicine Science,grade-1,55

Pankti,Science,grade-1,65

Shanaya,Science,grade-1,75

Aarohi,Science,grade-1,85

Sayantan,Science,grade-1,81

Aditya,Science,grade-1,71

Mathew,Physics,grade-2,55

```
Pankti,Physics,grade-2,54
Shanaya,Physics,grade-2,98
Aarohi,Physics,grade-2,37
Sayantan,Physics,grade-2,40
Aditya,Physics,grade-2,50
Mathew,Chemistry,grade-3,80
Pankti,Chemistry,grade-3,35
Shanaya,Chemistry,grade-3,47
Aarohi,Chemistry,grade-3,57
Sayantan,Chemistry,grade-3,67
Aditya,Chemistry,grade-3,77
Roohi,Literature,grade-4,87
Rashi,Literature,grade-4,97
Bhavya,Literature,grade-4,93
Nitin,Literature,grade-4,83
Himaja,Literature,grade-4,73
Aditya,Quantum Physics,grade-3,77
Roohi,Interior Designing,grade-4,87
Rashi,Interior Designing,grade-4,97
Bhavya,Interior Designing,grade-4,93
Nitin,Interior Designing,grade-4,83
Himaja,Interior Designing,grade-4,73
//starting the spark at hdfs
[acadgild@localhost ~]$ spark-shell
```

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

```
//reading the text file in a tupled rdd
val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
//filtering empty lines from text file
val rdd2 = rdd1.filter(lines => !lines.equals(""))
//mapping name ,subject,grade,sum to key  pair and 1 to value
//conversion of sum to integer to key value pair
val rdd3 = rdd2.map(word =>
```

```
(
(
word.split(",")(0),
word.split(",")(1),
word.split(",")(2),
word.split(",")(3).toInt
),
1)
)
```

```
// Find the count of total number of rows present.
val rdd4 = rdd3.count()
// displaying the count of total number of rows present.
println(rdd4)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:29

scala> val rdd3 = rdd2.map(word =>
     | (
     | (
     | word.split(",")(0),
     | word.split(",")(1),
     | word.split(",")(2),
     | word.split(",")(3).toInt
     | ),
     | 1)
     | )
rdd3: org.apache.spark.rdd.RDD[((String, String, String, Int), Int)] = MapPartitionsRDD[3] at map at
 <console>:31

scala> val rdd4 = rdd3.count()
rdd4: Long = 31

scala> println(rdd4)
31

scala>
```

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

```
// reading the text file and conversion to rdd
val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
// filtering empty lines from rdd
val rdd2 = rdd1.filter(lines => !lines.equals(""))
```

```
// mapping the rdd with only subject as key pair and 1 as value rdd
val rdd3 = rdd2.map(word =>
(
(
word.split(",")(1)
),
1)
)
//reduce operation to grouping the key subject and summing the value
val rdd4 = rdd3.reduceByKey((x,y) => x+y)
//fetching the key values
val rdd5 = rdd4.keys
//counting the distinct subjects
val rdd6 = rdd5.count
//displaying the count of distinct subjects
println(rdd6)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/colleg
estudent.txt",1)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <con
sole>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <conso
le>:29

scala> val rdd3 = rdd2.map(word =>
     | (
     | (
     | word.split(",")(1)
     | ),
     | 1)
     | )
rdd3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <c
onsole>:31

scala> val rdd4 = rdd3.reduceByKey((x,y) => x+y)
rdd4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at
 <console>:33

scala> val rdd5 = rdd4.keys
rdd5: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[5] at keys at <console
>:35

scala> val rdd6 = rdd5.count
rdd6: Long = 8

scala> println(rdd6)
8

scala>
```

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

```
//reading the hdfs file in pair rdd

val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt")

//filtering empty lines from rdd

val rdd2 = rdd1.filter(lines => !lines.equals(""))

//reading the name,grade and marks in rdd map operation as key and 1 as value

val rdd3 = rdd2.map(word =>

(

(

word.split(",")(0),

word.split(",")(2),

word.split(",")(3).toInt

),1

)

)

//grouping the key values and summing up the corresponding values

val rdd4 = rdd3.reduceByKey((x,y)=> x+ y)

//filtering the data where student name is Mathew and marks are 55 and counting
all the occurences

val rdd5 = rdd4.filter(x=> (x._1._1 == "Mathew")).filter(x => (x._1._3 ==
55)).count

//display the total number of students whose name is Mathew and marks is 55.

println(rdd5)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt")
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:29

scala> val rdd3 = rdd2.map(word =>
     | (
     | (
     | word.split(",")(0),
     | word.split(",")(2),
     | word.split(",")(3).toInt
     | ),1
     | )
     | )
rdd3: org.apache.spark.rdd.RDD[((String, String, Int), Int)] = MapPartitionsRDD[3] at map at <consol
e>:31

scala> val rdd4 = rdd3.reduceByKey((x,y)=> x+ y)
rdd4: org.apache.spark.rdd.RDD[((String, String, Int), Int)] = ShuffledRDD[4] at reduceByKey at <con
sole>:33

scala> val rdd5 = rdd4.filter(x=> (x._1._1 == "Mathew")).filter(x => (x._1._3 == 55)).count
rdd5: Long = 2

scala> println(rdd5)
2
```

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

```
//reading the text file in rdd
val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
//filtering out the empty lines from rdd
val rdd2 = rdd1.filter(lines => !lines.equals(""))
//map operation to map grade as key and 1 as value
val rdd3 = rdd2.map(word =>
(
(
word.split(",")(2)
),
1)
)
//reduce operation to combine similar keys and summing up the respective values
val rdd31 = rdd3.reduceByKey((x,y)=> x+y)
```

```
//sorting the grades in ascending order

val rdd41 = rdd31.sortByKey()

//displaying the student count per grade

rdd41.foreach(println)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[8] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[9] at filter at <console>:29

scala> val rdd3 = rdd2.map(word =>
     | (
     | (
     | word.split(",")(2)
     | ),
     | 1)
     | )
rdd3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[10] at map at <console>:31

scala> val rdd31 = rdd3.reduceByKey((x,y)=> x+y)
rdd31: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at reduceByKey at <console>:33

scala> val rdd41 = rdd31.sortByKey()
rdd41: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[12] at sortByKey at <console>:35

scala> rdd41.foreach(println)
(grade-1,8)
(grade-2,6)
(grade-3,7)
(grade-4,10)
```

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

```
//reading the text file into PAIR RDD

val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)

//filtering out the empty lines from rdd


val rdd2 = rdd1.filter(lines => !lines.equals(""))

//mapping name,grade as key pair and marks as value pair in rdd

val rdd3 = rdd2.map(word =>
((word.split(",")(0),word.split(",")(2)),word.split(",")(3).toInt))

//combining the key values and summing up the respective marks

val rdd4 = rdd3.reduceByKey((x,y)=> x+y).sortByKey()

//mapping the name and grade as key pair and 1 as value for count operation
```

```
val rdd31 = rdd2.map(word => ((word.split(",")(0),word.split(",")(2)),1))

//reducing the name and grade key pair by summing up the respective values and sorting keywise.

val rdd41 = rdd31.reduceByKey((a,b)=> a+b).sortByKey()

//joining two rdds for getting sum and count

val rdd5 = rdd4.join(rdd41)

//calculating average score per key pair

val rdd6 = rdd5.map{case(k,v)=> (k,v._1/v._2)}

//display the average score of each student per grade

rdd6.foreach(println)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[41] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[42] at filter at <console>:29

scala> val rdd3 = rdd2.map(word => ((word.split(",")(0),word.split(",")(2)),word.split(",")(3).toInt
))
rdd3: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[43] at map at <console>:3
1

scala> val rdd4 = rdd3.reduceByKey((x,y)=> x+y).sortByKey()
rdd4: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[45] at sortByKey at <console>:
33

scala> val rdd31 = rdd2.map(word => ((word.split(",")(0),word.split(",")(2)),1))
rdd31: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[46] at map at <console>:
31

scala> val rdd41 = rdd31.reduceByKey((a,b)=> a+b).sortByKey()
rdd41: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[48] at sortByKey at <console>
:33

scala> val rdd5 = rdd4.join(rdd41)
rdd5: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[51] at join at <co
nsole>:39

scala> val rdd6 = rdd5.map{case(k,v)=> (k,v._1/v._2)}
rdd6: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[52] at map at <console>:4
1
```

```
scala> rdd6.foreach(println)
((Aarohi,grade-2),37)
((Bhavya,grade-4),93)
((Pankti,grade-3),35)
((Shanaya,grade-1),75)
((Sayantan,grade-3),67)
((Sayantan,grade-1),81)
((Aarohi,grade-1),85)
((Sayantan,grade-2),40)
((Aditya,grade-3),77)
((Pankti,grade-1),65)
((Shanaya,grade-2),98)
((Nitin,grade-4),83)
((Mathew,grade-1),55)
((Rashi,grade-4),97)
((Mathew,grade-3),80)
((Shanaya,grade-3),47)
((Himaja,grade-4),73)
((Pankti,grade-2),54)
((Aditya,grade-2),50)
((Aarohi,grade-3),57)
((Aditya,grade-1),71)
((Roohi,grade-4),87)
((Mathew,grade-2),55)
```

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

```
//reading a text file into pair rdd

val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)

//filtering out the empty lines from rdd

val rdd2 = rdd1.filter(lines => !lines.equals(""))

//map operation to put subject as key pair and marks as value pair

val rdd3 = rdd2.map(word => (word.split(",")(1),word.split(",")(3).toInt))

//reduce operation to combine the keys and summing up the value pairs

val rdd4 = rdd3.reduceByKey((x,y)=> x+y).sortByKey()

//map operation to put subject as key pair and 1 as value pair

val rdd31 = rdd2.map(word => ((word.split(",")(1)),1))

//reduce operation to combine the keys and summing up the value pairs

val rdd41 = rdd31.reduceByKey((a,b)=> a+b).sortByKey()

//joining two rdds

val rdd5 = rdd4.join(rdd41)

//calculating average score per key pair

val rdd6 = rdd5.map{case(k,v)=> (k,v._1/v._2)}

//sorting the rdd in ascending order of keys

val rdd7 = rdd6.sortByKey()

//displaying the average score of each students across all grades

rdd7.foreach(println)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[14] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at filter at <console>:29

scala> val rdd3 = rdd2.map(word => (word.split(",")(1),word.split(",")(3).toInt))
rdd3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[16] at map at <console>:31

scala> val rdd4 = rdd3.reduceByKey((x,y)=> x+y).sortByKey()
rdd4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at sortByKey at <console>:33

scala> val rdd31 = rdd2.map(word => ((word.split(",")(1)),1))
rdd31: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[19] at map at <console>:31

scala> val rdd41 = rdd31.reduceByKey((a,b)=> a+b).sortByKey()
rdd41: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[21] at sortByKey at <console>:33

scala> val rdd5 = rdd4.join(rdd41)
rdd5: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[24] at join at <console>:39

scala> val rdd6 = rdd5.map{case(k,v)=> (k,v._1/v._2)}
rdd6: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[25] at map at <console>:41

scala> val rdd7 = rdd6.sortByKey()
rdd7: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[26] at sortByKey at <console>:43
```

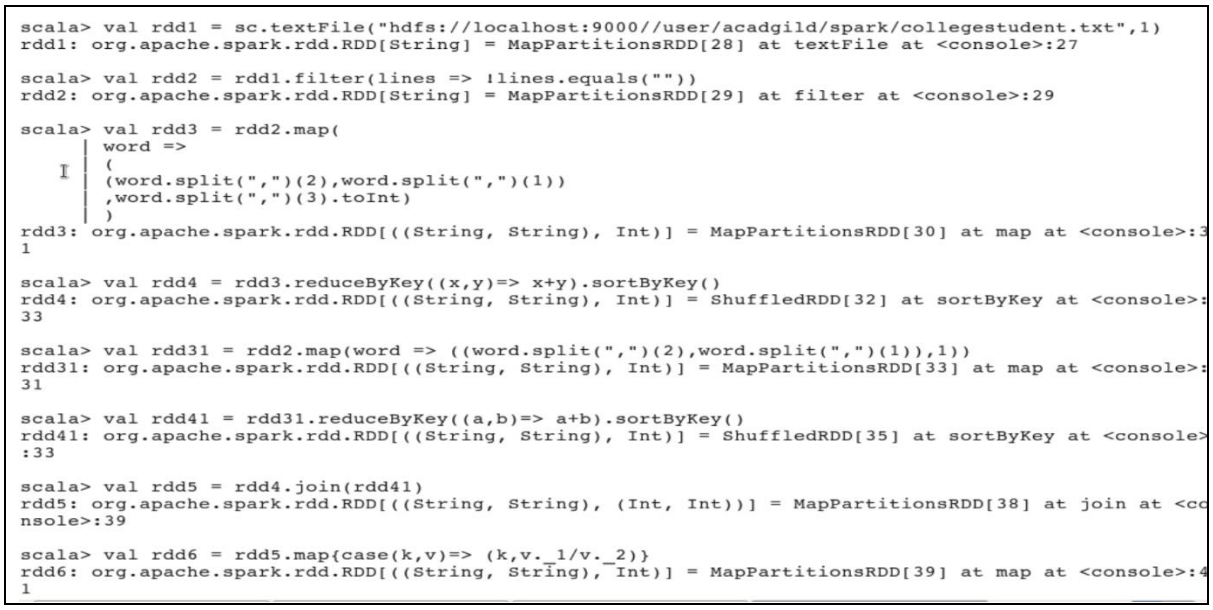
```
scala> rdd7.foreach(println)
(Chemistry,60)
(Interior Designing,86)
(Literature,86)
(Medicine Science,55)
(Physics,55)
(Quantum Physics,77)
(Science,72)
(Social Science,55)

scala> rdd7.collect()
res2: Array[(String, Int)] = Array((Chemistry,60), (Interior Designing,86), (Literature,86), (Medici
ne Science,55), (Physics,55), (Quantum Physics,77), (Science,72), (Social Science,55))
```

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

```
//reading the text file into rdd

val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)

//filtering the empty lines from rdd

val rdd2 = rdd1.filter(lines => !lines.equals(""))

//map operation to make grade and student as key pair and marks as value pair

val rdd3 = rdd2.map(

word =>

(

(word.split(",")(2),word.split(",")(1))

,word.split(",")(3).toInt)

)

//reduce operation to combine the keys and summing up the value pairs

val rdd4 = rdd3.reduceByKey((x,y)=> x+y).sortByKey()
```

```
//map operation to make grade and student as key pair and 1 as value pair

val rdd31 = rdd2.map(word => ((word.split(",")(2),word.split(",")(1)),1))

//reduce operation to combine the keys and summing up the value pairs

val rdd41 = rdd31.reduceByKey((a,b)=> a+b).sortByKey()

//joining two rdds

val rdd5 = rdd4.join(rdd41)

//calculating average score for each key pair

val rdd6 = rdd5.map{case(k,v)=> (k,v._1/v._2)}

//sorting the rdd in ascending order

val rdd7 = rdd6.sortByKey()

//displaying the average score for student per subject per grade

rdd7.foreach(println)

//displaying rdd in form of ARRAY(string)

rdd7.collect()
```

Screenshots:



```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[28] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[29] at filter at <console>:29

scala> val rdd3 = rdd2.map(
     | word =>
     | (
     | (word.split(",")(2),word.split(",")(1))
     | ,word.split(",")(3).toInt)
     | )
rdd3: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[30] at map at <console>:3
1

scala> val rdd4 = rdd3.reduceByKey((x,y)=> x+y).sortByKey()
rdd4: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[32] at sortByKey at <console>:
33

scala> val rdd31 = rdd2.map(word => ((word.split(",")(2),word.split(",")(1)),1))
rdd31: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[33] at map at <console>:
31

scala> val rdd41 = rdd31.reduceByKey((a,b)=> a+b).sortByKey()
rdd41: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[35] at sortByKey at <console>
:33

scala> val rdd5 = rdd4.join(rdd41)
rdd5: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[38] at join at <co
nsole>:39

scala> val rdd6 = rdd5.map{case(k,v)=> (k,v._1/v._2)}
rdd6: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[39] at map at <console>:4
1
```

```
scala> val rdd7 = rdd6.sortByKey()
rdd7: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[40] at sortByKey at <console>
43

scala> rdd7.foreach(println)
((grade-1,Medicine Science),55)
((grade-1,Science),72)
((grade-1,Social Science),55)
((grade-2,Physics),55)
((grade-3,Chemistry),60)
((grade-3,Quantum Physics),77)
((grade-4,Interior Designing),86)
((grade-4,Literature),86)

scala> rdd7.collect()
res4: Array[((String, String), Int)] = Array(((grade-1,Medicine Science),55), ((grade-1,Science),72
, ((grade-1,Social Science),55), ((grade-2,Physics),55), ((grade-3,Chemistry),60), ((grade-3,Quantu
 Physics),77), ((grade-4,Interior Designing),86), ((grade-4,Literature),86))
```

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

```
//reading a text file into rdd
val rdd1 =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt")
//filtering out the empty lines from rdd
val rdd2 = rdd1.filter(lines => !lines.equals(""))
//mapping the student name and grade as key pair and 1 as value pair
val rdd3 = rdd2.map(word =>
(
(
word.split(",")(0),
word.split(",")(2)
),1
)
)
//filtering only those rdds which have grade-2
val rdd4 = rdd3.filter(x=> (x._1._2 == "grade-2"))
//reduce operation to combine the keys and summing up the value pairs
val rdd5 = rdd4.reduceByKey((x,y)=> x+ y)
//mapping the student name and grade as key pair and marks as value pair
```

```
val rdd31 = rdd2.map(word =>
(
(
word.split(",")(0),
word.split(",")(2)
),
word.split(",")(3).toInt
)
)
//filtering only those records in rdd which have grade-2
val rdd41 = rdd31.filter(x=> (x._1._2 == "grade-2"))
//reduce operation to combine the keys and summing up the value pairs
val rdd51 = rdd41.reduceByKey((x,y)=> x+ y)
//joining two rdds
val joinedrdd = rdd51.join(rdd5)
//calculating the average score key wise
val rdd6 = joinedrdd.map{case(k,v)=> (k,v._1/v._2)}
//filtering the average score greater than 50
val rdd7 = rdd6.filter(x=>(x._2 > 50))
//display the count of all students in grade-2,who have average score greater than
50
rdd7.foreach(println)
```

Screenshots:

```
scala> val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt")
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[56] at textFile at <console>:27

scala> val rdd2 = rdd1.filter(lines => !lines.equals(""))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[57] at filter at <console>:29

scala> val rdd3 = rdd2.map(word =>
     | (
     | (
     | word.split(",")(0),
     | word.split(",")(2)
     | ),1
     | )
     | )
rdd3: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[58] at map at <console>:3
1

scala> val rdd4 = rdd3.filter(x=> (x._1._2 == "grade-2"))
rdd4: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[59] at filter at <console
>:33

scala> val rdd5 = rdd4.reduceByKey((x,y)=> x+ y)
rdd5: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[60] at reduceByKey at <console
>:35

scala> val rdd31 = rdd2.map(word =>
     | (
     | (
     | word.split(",")(0),
     | word.split(",")(2)
     | ),
     | word.split(",")(3).toInt
     | )
     | )
rdd31: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[61] at map at <console>:
31

scala> val rdd41 = rdd31.filter(x=> (x._1._2 == "grade-2"))
rdd41: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[62] at filter at <consol
e>:33

scala> val rdd51 = rdd41.reduceByKey((x,y)=> x+ y)
rdd51: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[63] at reduceByKey at <consol
e>:35

scala> val joinedrdd = rdd51.join(rdd5)
joinedrdd: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[66] at join a
t <console>:43

scala> val rdd6 = joinedrdd.map{case(k,v)=> (k,v._1/v._2)}
rdd6: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[67] at map at <console>:4
5

scala> val rdd7 = rdd6.filter(x=>(x._2>50))
rdd7: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[68] at filter at <console
>:47

scala> rdd7.foreach(println)
((Shanaya,grade-2),98)
((Pankti,grade-2),54)
((Mathew,grade-2),55)

scala> rdd7.collect()
res8: Array[((String, String), Int)] = Array(((Shanaya,grade-2),98), ((Pankti,grade-2),54), ((Mathew
,grade-2),55))
```

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

Average score per student_name across all grades is same as average score per student_name per grade

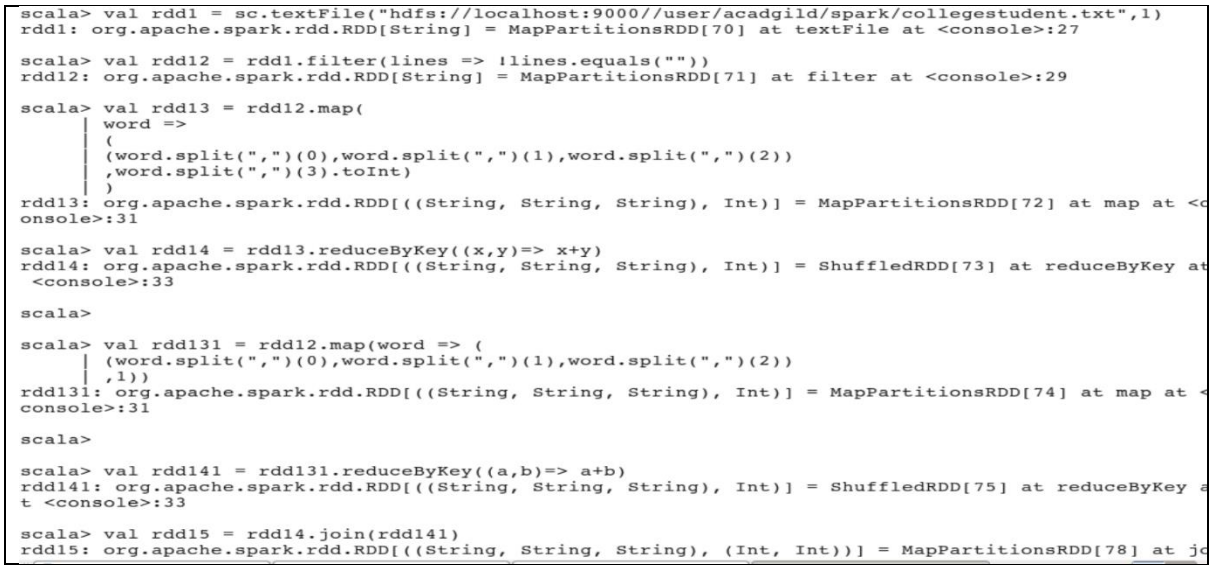
Hint - Use Intersection Property.

```
//across all  grades

//reading the text file into pair rdd

val rdd1 = sc.textFile("hdfs://localhost:9000//user/acadgild/spark/collegestudent.txt",1)

//filtering out the empty lines from rdd

val rdd12 = rdd1.filter(lines => !lines.equals(""))

//map operation to put student name,subject and grade as key pair and marks as value pair

val rdd13 = rdd12.map(

word =>

(

(word.split(",")(0),word.split(",")(1),word.split(",")(2))

,word.split(",")(3).toInt)

)

//reduce operation to group keys and sum up the value pairs

val rdd14 = rdd13.reduceByKey((x,y)=> x+y)

//map operation to put student name,subject and grade as key pair and 1 as value pair

val rdd131 = rdd12.map(word => (

(word.split(",")(0),word.split(",")(1),word.split(",")(2))

,1))

//reduce operation to group keys and sum up the value pairs

val rdd141 = rdd131.reduceByKey((a,b)=> a+b)

//joining two rdds

val rdd15 = rdd14.join(rdd141)

//calculating the average score

0val rdd16 = rdd15.map{case(k,v)=> (k,v._1/v._2)}

//sorting operaton to sort rdd in ascending order of key pairs

val rdd17 = rdd16.sortByKey()

//display the final result
```

```
rdd17.foreach(println)
```

```
//per student name per grade logic

val rdd23 = rdd12.map(
word =>
(
(word.split(",")(0),word.split(",")(1),word.split(",")(2))
,word.split(",")(3).toInt)
)
val rdd24 = rdd23.reduceByKey((x,y)=> x+y)
val rdd231 = rdd12.map(word => (
(word.split(",")(0),word.split(",")(1),word.split(",")(2))
,1))
val rdd241 = rdd231.reduceByKey((a,b)=> a+b)
val rdd25 = rdd24.join(rdd241)
val intersect = rdd15.intersection(rdd25)
val avg = intersect.map{case(k,v)=> (k,v._1/v._2)}
val finalavg = avg.sortByKey()
finalavg.foreach(println)
```

Screenshots:



```
scala> val rdd16 = rdd15.map{case(k,v)=> (k,v._1/v._2)}
rdd16: org.apache.spark.rdd.RDD[((String, String, String), Int)] = MapPartitionsRDD[79] at map at <c
onsole>:41

scala> val rdd17 = rdd16.sortByKey()
rdd17: org.apache.spark.rdd.RDD[((String, String, String), Int)] = ShuffledRDD[80] at sortByKey at <
console>:43
```

```
scala> rdd17.foreach(println)
((Aarohi,Chemistry,grade-3),57)
((Aarohi,Physics,grade-2),37)
((Aarohi,Science,grade-1),85)
((Aditya,Chemistry,grade-3),77)
((Aditya,Physics,grade-2),50)
((Aditya,Quantum Physics,grade-3),77)
((Aditya,Science,grade-1),71)
((Bhavya,Interior Designing,grade-4),93)
((Bhavya,Literature,grade-4),93)
((Himaja,Interior Designing,grade-4),73)
((Himaja,Literature,grade-4),73)
((Mathew,Chemistry,grade-3),80)
((Mathew,Medicine Science,grade-1),55)
((Mathew,Physics,grade-2),55)
((Mathew,Science,grade-1),55)
((Mathew,Social Science,grade-1),55)
((Nitin,Interior Designing,grade-4),83)
((Nitin,Literature,grade-4),83)
((Pankti,Chemistry,grade-3),35)
((Pankti,Physics,grade-2),54)
((Pankti,Science,grade-1),65)
((Rashi,Interior Designing,grade-4),97)
((Rashi,Literature,grade-4),97)
((Roohi,Interior Designing,grade-4),87)
((Roohi,Literature,grade-4),87)
((Sayantan,Chemistry,grade-3),67)
((Sayantan,Physics,grade-2),40)
((Sayantan,Science,grade-1),81)
((Shanaya,Chemistry,grade-3),47)
((Shanaya,Physics,grade-2),98)
((Shanaya,Science,grade-1),75)
```

```
scala> val rdd23 = rdd12.map(
     | word =>
     | (
     | (word.split(",")(0),word.split(",")(1),word.split(",")(2))
     | ,word.split(",")(3).toInt)
     | )
rdd23: org.apache.spark.rdd.RDD[((String, String, String), Int)] = MapPartitionsRDD[81] at map at <c
onsole>:31

scala> val rdd24 = rdd23.reduceByKey((x,y)=> x+y)
rdd24: org.apache.spark.rdd.RDD[((String, String, String), Int)] = ShuffledRDD[82] at reduceByKey at
 <console>:33

scala> val rdd231 = rdd12.map(word => (
     | (word.split(",")(0),word.split(",")(1),word.split(",")(2))
     | ,1))
rdd231: org.apache.spark.rdd.RDD[((String, String, String), Int)] = MapPartitionsRDD[83] at map at
console>:31

scala> val rdd241 = rdd231.reduceByKey((a,b)=> a+b)
rdd241: org.apache.spark.rdd.RDD[((String, String, String), Int)] = ShuffledRDD[84] at reduceByKey a
t <console>:33

scala> val rdd25 = rdd24.join(rdd241)
rdd25: org.apache.spark.rdd.RDD[((String, String, String), (Int, Int))] = MapPartitionsRDD[87] at jo
in at <console>:39

scala> val intersect = rdd15.intersection(rdd25)
intersect: org.apache.spark.rdd.RDD[((String, String, String), (Int, Int))] = MapPartitionsRDD[93] a
t intersection at <console>:51

scala> val avg = intersect.map{case(k,v)=> (k,v._1/v._2)}
avg: org.apache.spark.rdd.RDD[((String, String, String), Int)] = MapPartitionsRDD[94] at map at <con
sole>:53
```

```
scala> val avg = intersect.map{case(k,v)=> (k,v._1/v._2)}
avg: org.apache.spark.rdd.RDD[((String, String, String), Int)] = MapPartitionsRDD[94] at map at <con
sole>:53


scala> val finalavg = avg.sortByKey()
finalavg: org.apache.spark.rdd.RDD[((String, String, String), Int)] = ShuffledRDD[95] at sortByKey a
t <console>:55
```

```
scala> finalavg.foreach(println)
((Aarohi,Chemistry,grade-3),57)
((Aarohi,Physics,grade-2),37)
((Aarohi,Science,grade-1),85)
((Aditya,Chemistry,grade-3),77)
((Aditya,Physics,grade-2),50)
((Aditya,Quantum Physics,grade-3),77)
((Aditya,Science,grade-1),71)
((Bhavya,Interior Designing,grade-4),93)
((Bhavya,Literature,grade-4),93)
((Himaja,Interior Designing,grade-4),73)
((Himaja,Literature,grade-4),73)
((Mathew,Chemistry,grade-3),80)
((Mathew,Medicine Science,grade-1),55)
((Mathew,Physics,grade-2),55)
((Mathew,Science,grade-1),55)
((Mathew,Social Science,grade-1),55)
((Nitin,Interior Designing,grade-4),83)
((Nitin,Literature,grade-4),83)
((Pankti,Chemistry,grade-3),35)
((Pankti,Physics,grade-2),54)
((Pankti,Science,grade-1),65)
((Rashi,Interior Designing,grade-4),97)
((Rashi,Literature,grade-4),97)
((Roohi,Interior Designing,grade-4),87)
((Roohi,Literature,grade-4),87)
((Sayantan,Chemistry,grade-3),67)
((Sayantan,Physics,grade-2),40)
((Sayantan,Science,grade-1),81)
((Shanaya,Chemistry,grade-3),47)
((Shanaya,Physics,grade-2),98)
((Shanaya,Science,grade-1),75)
```