Assignment 17.2

Downloaded the dataset and uploaded the data set to the local file **/home/acadgild/Sumona**.

PROBLEM STATEMENT 1

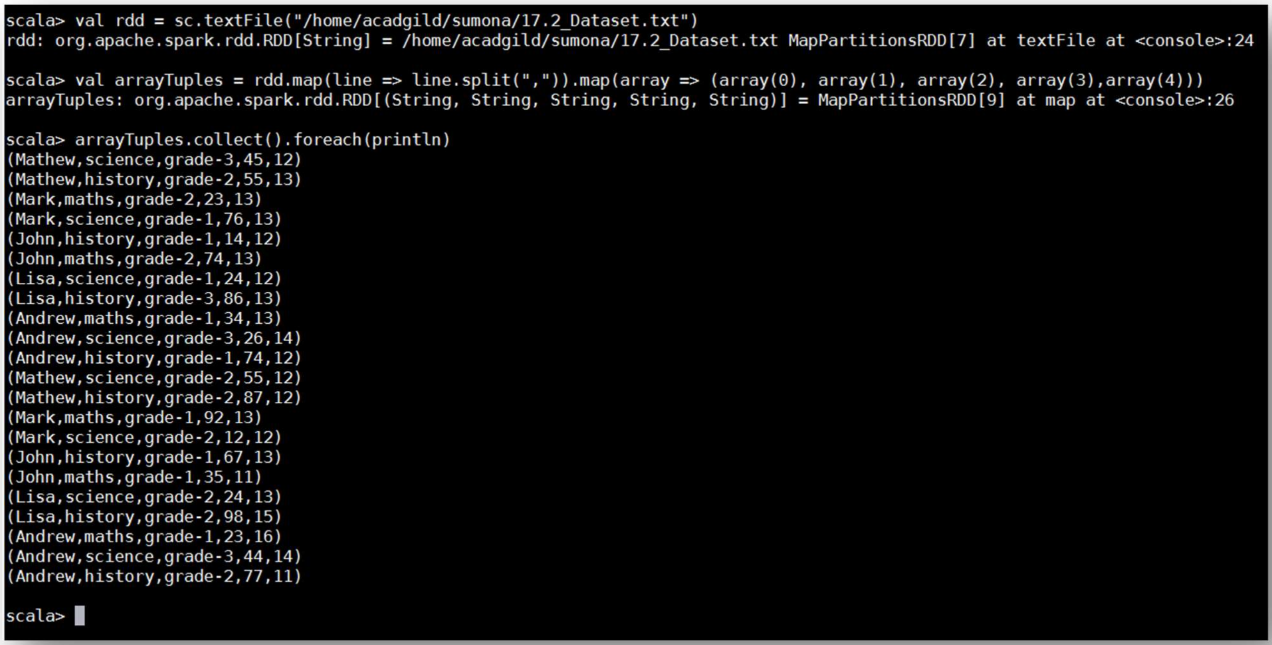
1. Read the text file, and create a tuple rdd.

Read the text file : **val rdd = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt")**

This command, will help to read the dataset from the local file system.

Create a tupled RDD : **val arrayTuples = rdd.map(line => line.split(",")).map(array => (array(0), array(1), array(2), array(3),array(4)))**

This command, will create a tuples of words (arrays) which are separated by a comma(,).

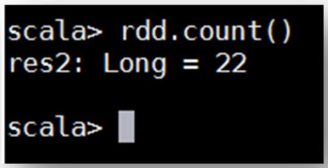


```
scala> val rdd = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt")
rdd: org.apache.spark.rdd.RDD[String] = /home/acadgild/sumona/17.2_Dataset.txt MapPartitionsRDD[7] at textFile at <console>:24

scala> val arrayTuples = rdd.map(line => line.split(",")).map(array => (array(0), array(1), array(2), array(3),array(4)))
arrayTuples: org.apache.spark.rdd.RDD[(String, String, String, String, String)] = MapPartitionsRDD[9] at map at <console>:26

scala> arrayTuples.collect().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)

scala> 
```

arrayTuples.collect().foreach(println) This command will print the output of the Tuples

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

rdd.count() This command helps to count the number of rows present in the dataset or a text file.

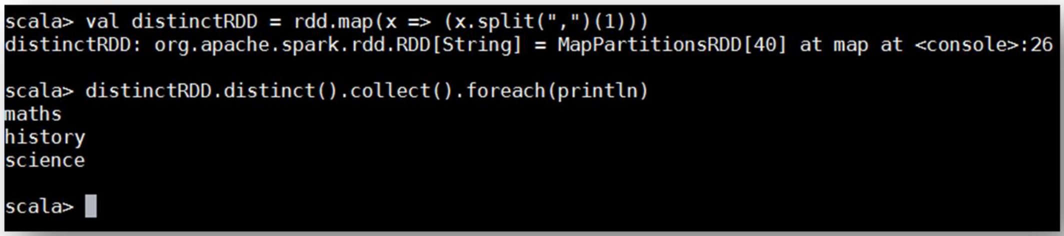


```
scala> rdd.count()
res2: Long = 22

scala>
```

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

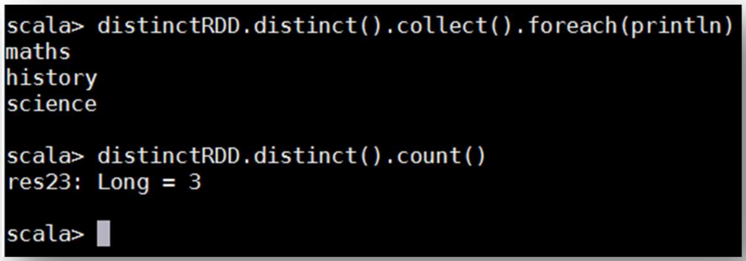
val distinctRDD = rdd.map(x => (x.split(",")(1))) This command will get the distinct values of the array 1 (which is assigned for subjects)

distinctRDD.distinct().collect().foreach(println) This command will collect the output and print in each line.



```
scala> val distinctRDD = rdd.map(x => (x.split(",")(1)))
distinctRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[40] at map at <console>:26

scala> distinctRDD.distinct().collect().foreach(println)
maths
history
science

scala>
```

Now, to get the count of number of distinct subjects, we use the command **distinctRDD.distinct().count()**



```
scala> distinctRDD.distinct().collect().foreach(println)
maths
history
science

scala> distinctRDD.distinct().count()
res23: Long = 3

scala>
```

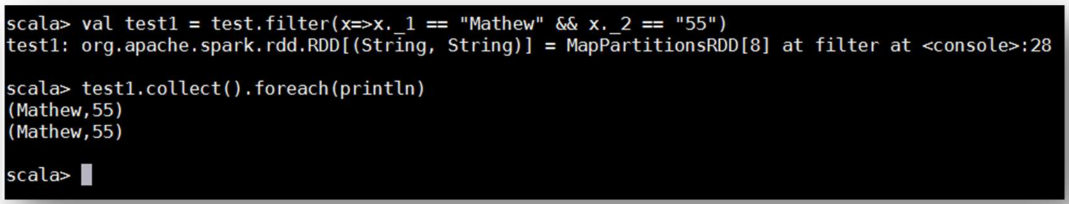
As you see, the count is 3. So we have 3 distinct subjects **Maths, History, Science**

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

First let us filter the columns from RDD with the command **val test = rdd.map(x => (x.split(",")(0), x.split(",")(3)))**

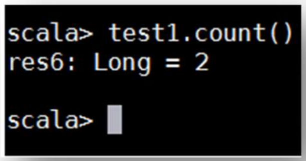
Now, to filter out the name to be **Mathew** and the marks to be **55**, we shall use the command

val test1 = test.filter(x=>x._1 == "Mathew" || x._2 == "55")



```
scala> val test1 = test.filter(x=>x._1 == "Mathew" && x._2 == "55")
test1: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[8] at filter at <console>:28

scala> test1.collect().foreach(println)
(Mathew,55)
(Mathew,55)

scala> 
```

To get the count for the query, we shall use the command



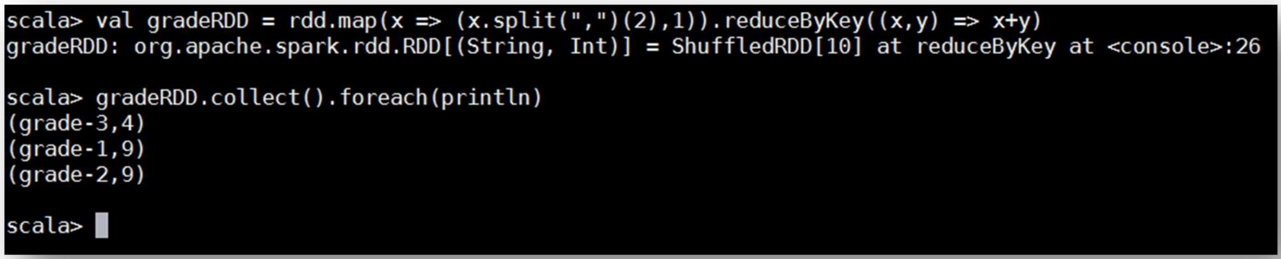
```
scala> test1.count()
res6: Long = 2

scala> 
```

PROBLEM STATEMENT 2

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

We will use Map command to filter out the column for Grade and the then also use **reduceByKey** to add the occurrence of the grades and then get the count of number of students for each grade

val gradeRDD = rdd.map(x => (x.split(",")(2),1)).reduceByKey((x,y) => x+y)


```
scala> val gradeRDD = rdd.map(x => (x.split(",")(2),1)).reduceByKey((x,y) => x+y)
gradeRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[10] at reduceByKey at <console>:26

scala> gradeRDD.collect().foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)

scala> 
```

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

First we shall filter out the columns from the RDD

val baseRDD = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

Then we map the values for the above RDD

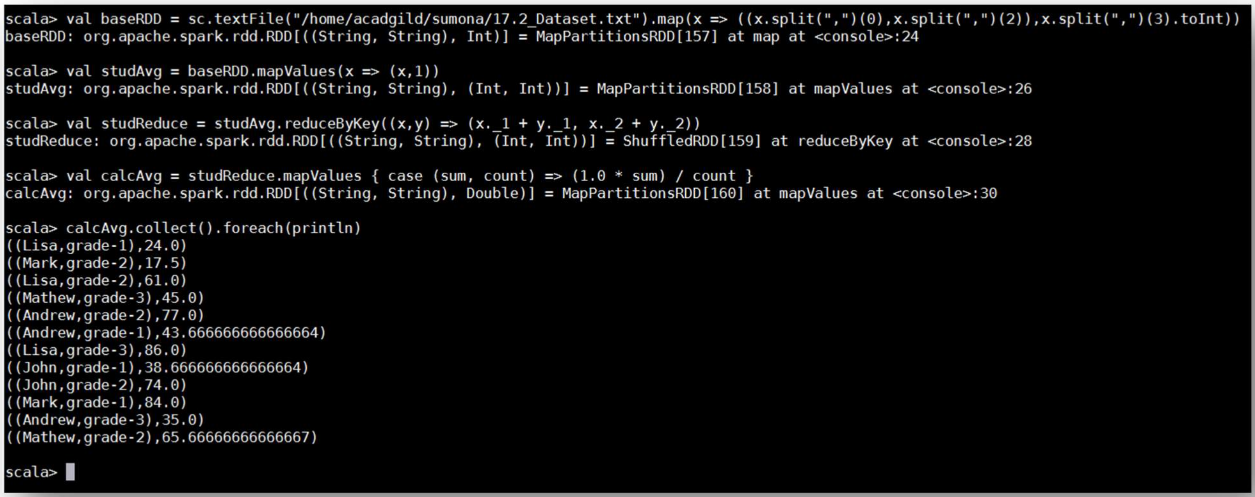
val studAvg = baseRDD.mapValues(x => (x,1))

Now, we are adding the occurrences of the marks using reduceByKey

val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

Now we are calculating the average of each students

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


```
scala> val baseRDD = sc.textFile("/home/acadgild/sumona/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[157] at map at <console>:24

scala> val studAvg = baseRDD.mapValues(x => (x,1))
studAvg: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[158] at mapValues at <console>:26

scala> val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
studReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[159] at reduceByKey at <console>:28

scala> val calcAvg = studReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
calcAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[160] at mapValues at <console>:30

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

scala> 
```

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

First we shall filter out the columns from the RDD

val baseRDD = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))

Then we map the values for the above RDD

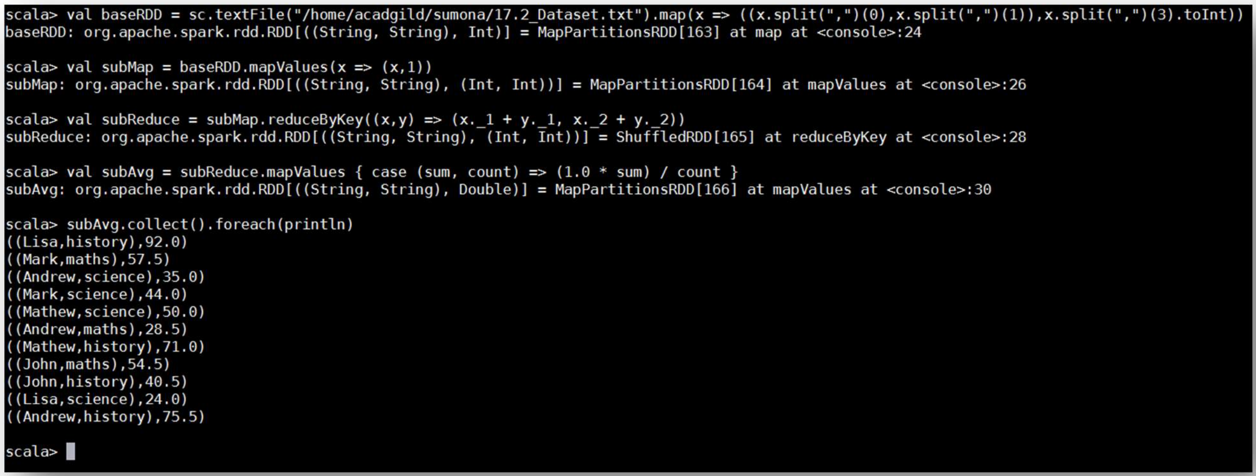
val subMap = baseRDD.mapValues(x => (x,1))

Now, we are adding the occurrences of the marks using reduceByKey

val subReduce = subMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

Now we are calculating the average of students in each subject

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



```
scala> val baseRDD = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[163] at map at <console>:24

scala> val subMap = baseRDD.mapValues(x => (x,1))
subMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[164] at mapValues at <console>:26

scala> val subReduce = subMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
subReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[165] at reduceByKey at <console>:28

scala> val subAvg = subReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
subAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[166] at mapValues at <console>:30

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

scala> 
```

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

First we shall filter out the columns from the RDD

val baseRDD = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))

Then we map the values for the above RDD

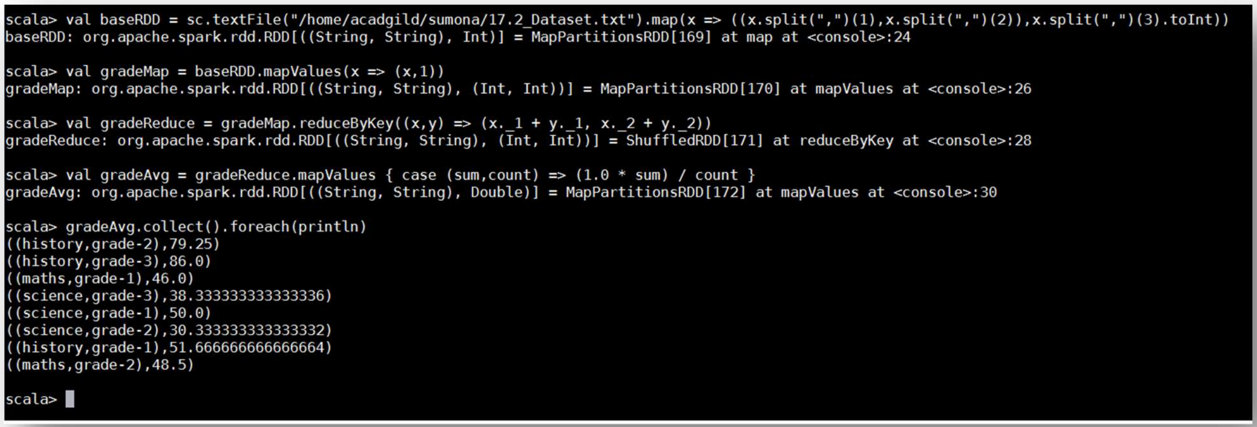
val gradeMap = baseRDD.mapValues(x => (x,1))

Now, we are adding the occurrences of the marks using reduceByKey

val gradeReduce = gradeMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

Now we are calculating the average of students in each subject per grade

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



```
scala> val baseRDD = sc.textFile("/home/acadgild/sumona/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[169] at map at <console>:24

scala> val gradeMap = baseRDD.mapValues(x => (x,1))
gradeMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[170] at mapValues at <console>:26

scala> val gradeReduce = gradeMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
gradeReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[171] at reduceByKey at <console>:28

scala> val gradeAvg = gradeReduce.mapValues { case (sum,count) => (1.0 * sum) / count }
gradeAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[172] at mapValues at <console>:30

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

scala> 
```

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

First we shall filter out the columns from the RDD

val baseRDD = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

Then we map the values for the above RDD

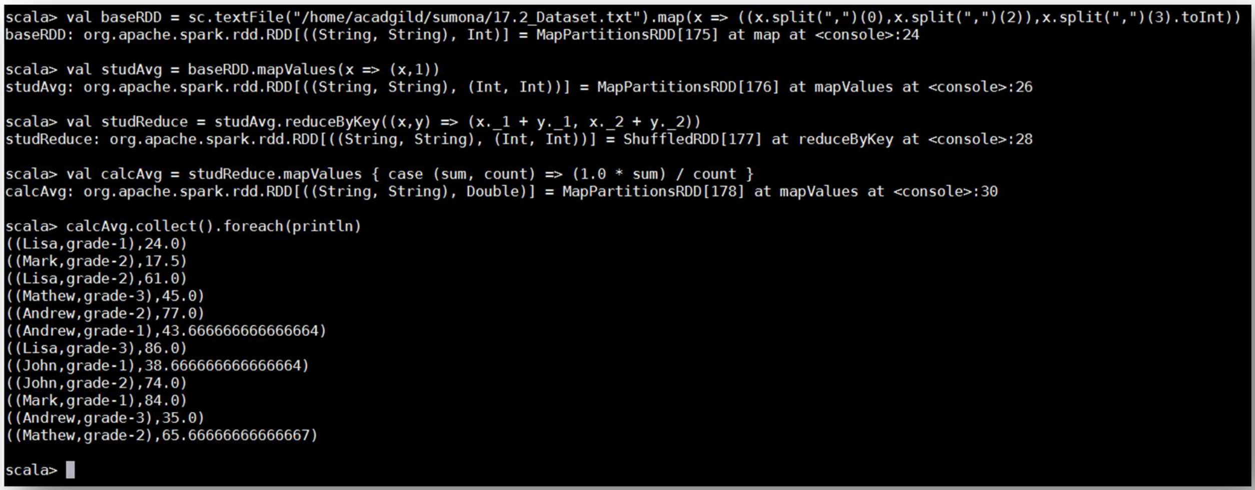
val studAvg = baseRDD.mapValues(x => (x,1))

Now, we are adding the occurrences of the marks using reduceByKey

val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

Then calculate the average marks as per grades

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

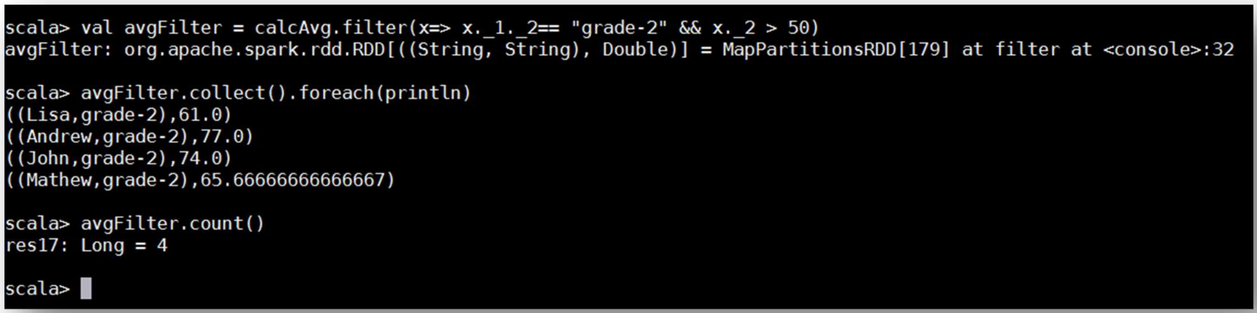


```
scala> val baseRDD = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[175] at map at <console>:24

scala> val studAvg = baseRDD.mapValues(x => (x,1))
studAvg: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[176] at mapValues at <console>:26

scala> val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
studReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[177] at reduceByKey at <console>:28

scala> val calcAvg = studReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
calcAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[178] at mapValues at <console>:30

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

scala> 
```

Now, we shall filter out the students in **grade-2** having an average more than **50**

val avgFilter = calcAvg.filter(x=> x._1._2 == "grade-2" && x._2 > 50)

To get the count we shall run the command **avgFilter.count()**



```
scala> val avgFilter = calcAvg.filter(x=> x._1._2== "grade-2" && x._2 > 50)
avgFilter: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[179] at filter at <console>:32

scala> avgFilter.collect().foreach(println)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)

scala> avgFilter.count()
res17: Long = 4

scala> 
```

PROBLEM STATEMENT 3

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

We will first get the list of average score of each students for grade

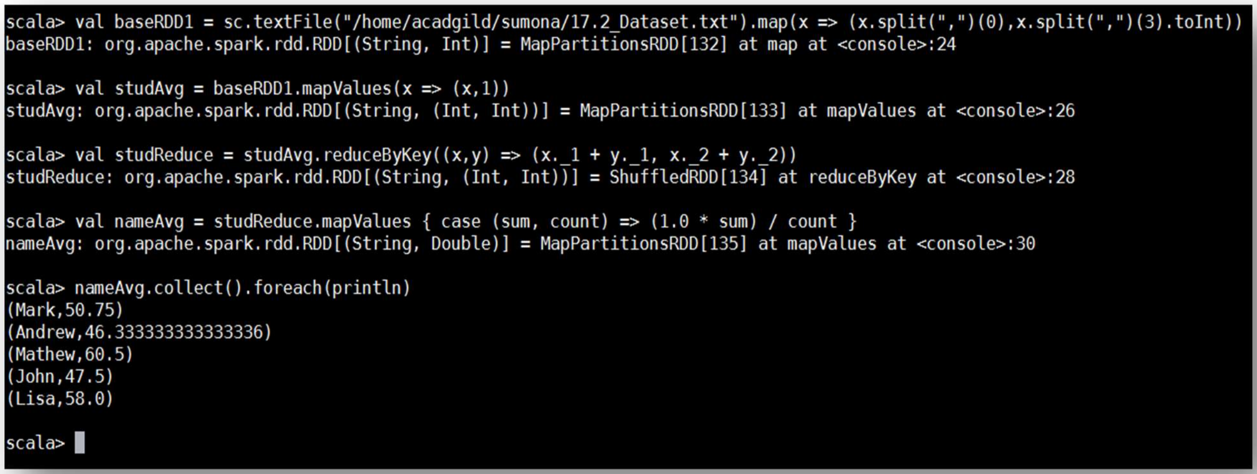
The following commands are used to Filter the columns, map each value of the filtered RDD, adding the marks of students and the number of occurrence and calculate the average of the students respectively

val baseRDD1 = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => (x.split(",")(0),x.split(",")(3).toInt))

val studAvg = baseRDD1.mapValues(x => (x,1))

val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

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



```
scala> val baseRDD1 = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => (x.split(",")(0),x.split(",")(3).toInt))
baseRDD1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[132] at map at <console>:24

scala> val studAvg = baseRDD1.mapValues(x => (x,1))
studAvg: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[133] at mapValues at <console>:26

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[134] at reduceByKey at <console>:28

scala> val nameAvg = studReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
nameAvg: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[135] at mapValues at <console>:30

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

scala> 
```

List of average of each students for each grade

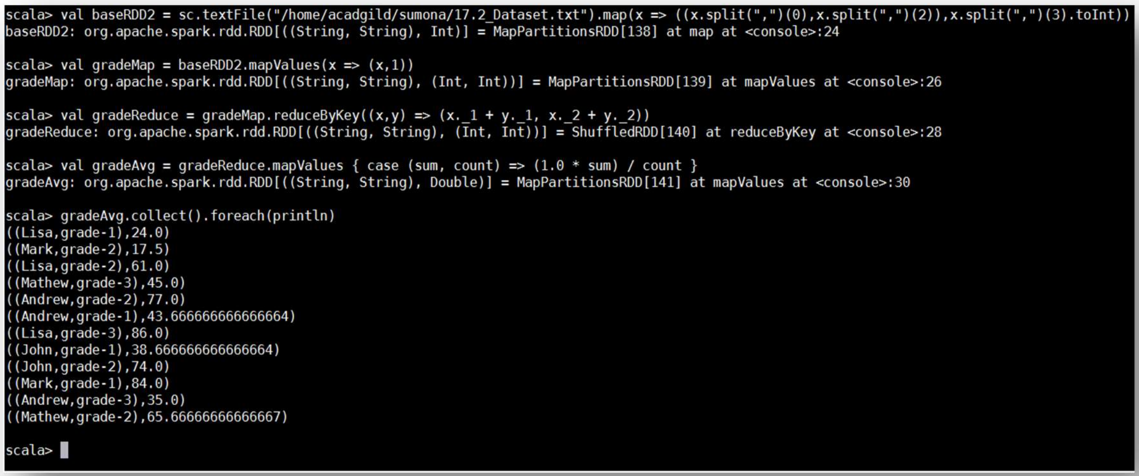
The following commands are used to Filter the columns, map each value of the filtered RDD, adding the marks of students and the number of occurrence and calculate the average of the students respectively

val baseRDD2 = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val gradeMap = baseRDD2.mapValues(x => (x,1))

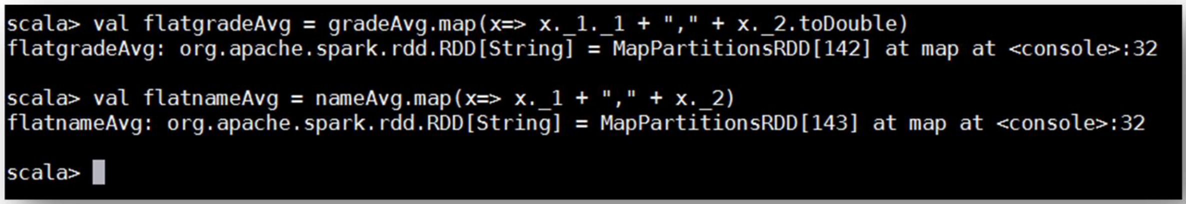
val gradeReduce = gradeMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

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



```
scala> val baseRDD2 = sc.textFile("/home/acadgild/sumona/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[138] at map at <console>:24

scala> val gradeMap = baseRDD2.mapValues(x => (x,1))
gradeMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[139] at mapValues at <console>:26

scala> val gradeReduce = gradeMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
gradeReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[140] at reduceByKey at <console>:28

scala> val gradeAvg = gradeReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
gradeAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[141] at mapValues at <console>:30

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

scala> 
```

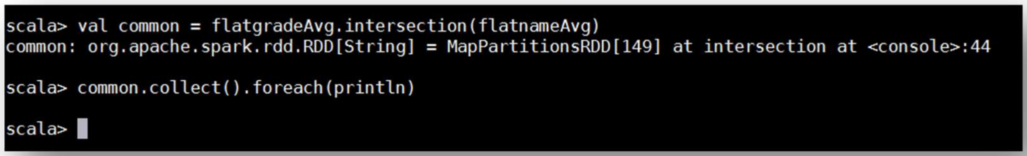
Now we shall extract the name and marks from the RDD’s



```
scala> val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[142] at map at <console>:32

scala> val flatnameAvg = nameAvg.map(x=> x._1 + "," + x._2)
flatnameAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[143] at map at <console>:32

scala> 
```

Now lets us use the **intersection** function and check if there are any Students who have the same average for marks for all grades and average

val common = flatgradeAvg.intersection(flatnameAvg)



```
scala> val common = flatgradeAvg.intersection(flatnameAvg)
common: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[149] at intersection at <console>:44

scala> common.collect().foreach(println)

scala> 
```

There are no students which have similar average marks for all grade as that of average marks for each grade.
]]>