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

Solution-

We have taken a dataset and kept in local file in below location-

```
[acadgild@localhost assignment-17.1]$ pwd
/home/acadgild/assignment-17.1
[acadgild@localhost assignment-17.1]$ ls -l
total 4
-rw-rw-r--. 1 acadgild acadgild 113 Dec 25 00:30 Sample-Data.txt
```

In below screenshot we are creating a RDD using spark context to read the file-

```
scala> val input = sc.textFile("/home/acadgild/assignment-17.1/Sample-Data.txt")
input: org.apache.spark.rdd.RDD[String] = /home/acadgild/assignment-17.1/Sample-Data.txt MapPartitionsRDD[1] at textFile at <console>:24

scala>
```

In below screenshot we are using function count() to count the number of rows in the file.

The result is 3-

```
scala> val rowCount = input.count()
rowCount: Long = 3

scala>
```

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

Solution-

```
scala> val input = sc.textFile("/home/acadgild/assignment-17.1/Sample-Data.txt")
input: org.apache.spark.rdd.RDD[String] = /home/acadgild/assignment-17.1/Sample-Data.txt MapPartitionsRDD[1] at textFile at <console>:24

scala>
```

We are using above file only for this problem also. In order to count the words we are first splitting it using the delimiter "-" and doing a flatmap-

```
scala> val words = input.flatMap(x => x.split("-"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:26
```

Now below screenshot shows the result of above RDD i.e. the words after splitting-

```
scala> words.collect()
res1: Array[String] = Array(This, is, my, first, assignment., It, will, count, the, number, of, lines, in, this, document., The, total, number, of, lines, is, 3)

scala>
```

Now we are using count() function to count the number of words present in the file-

```
scala> val count2 = words.count()
count2: Long = 22

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-

```
scala> val input = sc.textFile("/home/acadgild/assignment-17.1/Sample-Data.txt")
input: org.apache.spark.rdd.RDD[String] = /home/acadgild/assignment-17.1/Sample-Data.txt MapPartitionsRDD[1] at textFile at <console>:24

scala>
```

Here also we are again using above file only to do further operation and using split function to separate the words based on delimiter "-".

```
scala> val words = input.flatMap(x => x.split("-"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:26
```

Now we are mapping each word with the value 1 which will create a tuple. After that we are using reduceByKey to add those numbers of occurrences as shown below-

```
scala> val result = words.map(x => (x,1)).reduceByKey((x,y) => x+y)
result: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:28
```

Below screenshot shows the final result-

```
scala> result.foreach(println)
(this,1)
(is,2)
(will,1)
(This,1)
(first,1)
(total,1)
(my,1)
(lines,2)
(The,1)
(document.,1)
(assignment.,1)
(number,2)
(in,1)
(3,1)
(of,2)
(It,1)
(count,1)
(the,1)
```

Below is the dataset which we will be using for this Assignment in all problems. It has been kept in local file system-

```
[acadgild@localhost Assignment-17.2]$ pwd
/home/acadgild/Assignment-17.2
[acadgild@localhost Assignment-17.2]$ ls -l
total 4
-rw-rw-r--. 1 acadgild acadgild 622 Dec 25 01:45 17.2_Dataset.txt
[acadgild@localhost Assignment-17.2]$
```

Problem Statement 1:

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

Solution-

Below is the code used –

- val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => (x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt,x.split(",")(4).toInt)))
- val count1 = baseRDD.count()

Now first we are creating a RDD named as baseRDD using the spark context to read the input file and splitting the lines based on delimiter “,” to create a tuple.

```
scala> val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_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[7] at map at <console>:24
```

As shown below it as tupled RDD with name as key and subject,grade,marks and age as values-

```
scala> baseRDD.foreach(println)
(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))
(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))
```

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

Solution-

We are using count() function to find the number of rows present as shown below-

```
scala> val count1 = baseRDD.count()
count1: Long = 22
```

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

Solution-

Below is the code used-

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

Now we will try to understand each and every command one by one-

First we are creating a RDD to read the file and selecting only subject name and mapping them with value 1 -

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

Below shows the result of above RDD-

```
scala> baseRDD.foreach(println)
[Stage 1:>                                                          (0 + 0) / 2](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)
```

Now we are counting the values of occurrences using reduceByKey to get number of subjects

```
scala> val subCount = baseRDD.reduceByKey((x,y) => x + y)
subCount: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[6] at reduceByKey at <console>:26
```

Below screenshot shows the subject name and their count-

```
scala> subCount.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**

Solution-

Below is the code used-

- val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(3).toInt),1))
- val filterRDD = baseRDD.filter(x => x._1._1 == "Mathew" && x._1._2 == "55")
- val reduceVal = filterRDD.reduceByKey((x,y) => x + y)

First we are creating a RDD to read the file and selecting name and marks as key and mapping them with value 1 -

```
scala> val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(3).toInt),1))
baseRDD: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[19] at map at <console>:24
```

Below screenshot shows the result of the same-

```
scala> baseRDD.foreach(println)
((Mathew,45),12)
((Mathew,55),13)
((Mark,23),13)
((Mark,76),13)
((John,14),12)
((John,74),13)
((Lisa,24),12)
((Lisa,86),13)
((Andrew,34),13)
((Andrew,26),14)
((Andrew,74),12)
((Mathew,55),12)
((Mathew,87),12)
((Mark,92),13)
((Mark,12),12)
((John,67),13)
((John,35),11)
((Lisa,24),13)
((Lisa,98),15)
((Andrew,23),16)
((Andrew,44),14)
((Andrew,77),11)
```

Now we are filtering above result which contains name as Mathew and marks equal to 55

```
scala> val filterRDD = baseRDD.filter(x => x._1._1 == "Mathew" && x._1._2 == 55)
filterRDD: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[20] at filter at <console>:26

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

Now we are counting each occurrences -

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

scala>
```

Below screenshot shows the result for the same-

```
scala> reduceVal.foreach(println)
((Mathew,55),2)
```

Problem Statement 2:

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

Below is the code used for same-

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

First we are creating a RDD to read the file and selecting only grade and mapping 1 with it-

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

Below is the result of above RDD-

```
scala> baseRDD.foreach(println)
[Stage 14:>                                                 (0 + 0) / 2](grade-2,1)
(grade-2,1)
(grade-1,1)
(grade-2,1)
(grade-1,1)
(grade-1,1)
(grade-2,1)
(grade-2,1)
(grade-1,1)
(grade-3,1)
(grade-2,1)
(grade-3,1)
(grade-2,1)
(grade-2,1)
(grade-1,1)
(grade-1,1)
(grade-2,1)
(grade-1,1)
(grade-3,1)
(grade-1,1)
(grade-3,1)
(grade-1,1)
```

Now are using reduuceByKey to add the occurrences of the grades. Below screenshot shows same along with the result-

```
scala> val studCount = baseRDD.reduceByKey((x,y) => x + y )
studCount: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[25] at reduceByKey at <console>:26

scala> studCount.foreach(println)
(grade-2,9)
(grade-3,4)
(grade-1,9)
```

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

Solution-

Below is the code used to find the result-

- val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
- val studAvg = baseRDD.mapValues(x => (x,1))
- val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
- val calcAvg = studReduce.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 baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[2] at map at <console>:24
```

Below screenshot shows the result for above RDD-

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

Now we are mapping the values of above paired RDD with 1 using mapValues function-

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

scala> studAvg.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))
```

Here we are using reduceByKey to add the occurrences of marks for each key which is student name and grade-

```
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[4] at reduceByKey at <console>:28

scala> studReduce.foreach(println)
((Lisa,grade-1),(24,1))
((Mark,grade-2),(35,2))
((Andrew,grade-2),(77,1))
((Lisa,grade-2),(122,2))
((John,grade-1),(116,3))
((Mathew,grade-3),(45,1))
((John,grade-2),(74,1))
((Mathew,grade-2),(197,3))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
```

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

scala> calcAvg.foreach(println)
((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)
((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)
```

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

<u>Solution-</u>

Below is the code used to find the result-

- val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
- val subMap = baseRDD.mapValues(x => (x,1))
- val subReduce = subMap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
- val subAvg = subReduce.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/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[8] 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 subMap = baseRDD.mapValues(x => (x,1))
subMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[9] at mapValues at <console>:26

scala> subMap.foreach(println)
((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))
((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))
```

Now we are adding the marks and number of occurrences for each key using reducebyKey-

```
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[10] at reduceByKey at <console>:28

scala> subReduce.foreach(println)
((Lisa,history),(184,2))
((Mark,science),(88,2))
((John,history),(81,2))
((Lisa,science),(48,2))
((Andrew,history),(151,2))
((Mark,maths),(115,2))
((Andrew,science),(70,2))
((Mathew,science),(100,2))
((Andrew,maths),(57,2))
((Mathew,history),(142,2))
((John,maths),(109,2))
```

In below step we are calculating average by dividing the sum of marks and count of occurrences for each key

```
scala> val subAvg = subReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
subAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[11] at mapValues at <console>:30

scala> subAvg.foreach(println)
((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)
```

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

Solution- Below is the code used to find the result-

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

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/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[14] 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 1using function mapValues-

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

scala> gradeMap.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))
```

Now we are adding marks and number of occurrences for each key using reduceByKey-

```
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[16] at reduceByKey at <console>:28

scala> gradeReduce.foreach(println)
((history,grade-2),(317,4))
((science,grade-3),(115,3))
((science,grade-1),(100,2))
((science,grade-2),(91,3))
((history,grade-1),(155,3))
((history,grade-3),(86,1))
((maths,grade-1),(184,4))
((maths,grade-2),(97,2))
```

Then we are calculating average by dividing the sum of marks with number of occurrences-

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

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

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

Solution-

Below code has been used to find the required result-

- val baseRDD = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
- val studAvg = baseRDD.mapValues(x => (x,1))
- val studReduce = studAvg.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
- val calcAvg = studReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
- val filterGrade = calcAvg.filter(x => x._1._2 == "grade-2" && x._2 > 50)
- val countStud = filterGrade2.count()

First we are 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/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[2] at map at <console>:24
```

Below screenshot shows the result for above RDD-

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

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

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

scala> studAvg.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))
```

In below step we are adding the marks of subject and number of occurrences per key using reduceByKey function-

```
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[4] at reduceByKey at <console>:28

scala> studReduce.foreach(println)
((Lisa,grade-1),(24,1))
((Mark,grade-2),(35,2))
((Andrew,grade-2),(77,1))
((Lisa,grade-2),(122,2))
((John,grade-1),(116,3))
((Mathew,grade-3),(45,1))
((John,grade-2),(74,1))
((Mathew,grade-2),(197,3))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
```

Here we are calculating the average of each student -

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

scala> calcAvg.foreach(println)
((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)
((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)
```

Now in below step we are filtering the above result with student belonging to grade-2 and having marks greater than 50-

```
scala> val filterGrade2 = calcAvg.filter(x => x._1._2 == "grade-2" && x._2 > 50)
filterGrade2: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[6] at filter at <console>:32

scala> filterGrade2.foreach(println)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)
((Lisa,grade-2),61.0)
```

Below screenshot shows the final result which is the count which is 4

```
scala> val countStud = filterGrade2.count()
countStud: 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

Solution- To find the solution of above problem we will 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/Assignment-17.2/17.2_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 }

First we create a paired RDD named as baseRDD1 by extracting only name and marks-

```
scala> val baseRDD1 = sc.textFile("/home/acadgild/Assignment-17.2/17.2_Dataset.txt").map(x => (x.split(",")(0),x.split(",")(3).toInt))
baseRDD1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[9] 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))
studAvg: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[10] at mapValues at <console>:26

scala> studAvg.foreach(println)
(Mathew,(45,1))
(Mathew,(55,1))
(Mark,(23,1))
(Mathew,(55,1))
(Mathew,(87,1))
(Mark,(76,1))
(Mark,(92,1))
(John,(14,1))
(Mark,(12,1))
(John,(74,1))
(John,(67,1))
(John,(35,1))
(Lisa,(24,1))
(Lisa,(98,1))
(Andrew,(23,1))
(Lisa,(24,1))
(Andrew,(44,1))
(Lisa,(86,1))
(Andrew,(34,1))
(Andrew,(77,1))
(Andrew,(26,1))
(Andrew,(74,1))
```

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[11] at reduceByKey at <console>:28

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

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

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

scala> nameAvg.foreach(println)
[Stage 0:>                                                          (0 + 2) / 2](Mathew,60.5)
(Mark,50.75)
(Andrew,46.333333333333336)
(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 the same-

- val baseRDD2 = sc.textFile("/home/acadgild/Assignment-17.2/17.2_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 }

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/Assignment-17.2/17.2_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[8] at map at <console>:24

scala> baseRDD2.foreach(println)
((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)
((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)
((Andrew,grade-1),23)
((Andrew,grade-3),44)
((Andrew,grade-2),77)
```

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

```
scala> val gradeMap = baseRDD2.mapValues(x => (x,1))
gradeMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[9] at mapValues at <console>:26

scala> gradeMap.foreach(println)
((Mathew,grade-2),(55,1))
((Mathew,grade-3),(45,1))
((Mathew,grade-2),(87,1))
((Mathew,grade-2),(55,1))
((Mark,grade-2),(23,1))
((Mark,grade-1),(76,1))
((Mark,grade-1),(92,1))
((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))
((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-

```
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[10] at reduceByKey at <console>:28

scala> gradeReduce.foreach(println)
((Mark,grade-2),(35,2))
((Lisa,grade-2),(122,2))
((Mathew,grade-3),(45,1))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
((Lisa,grade-1),(24,1))
((Andrew,grade-2),(77,1))
((John,grade-1),(116,3))
((John,grade-2),(74,1))
((Mathew,grade-2),(197,3))
```

In below step we are 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[11] at mapValues at <console>:30

scala> gradeAvg.foreach(println)
((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)
((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)
```

Now to proceed further we are extracting name and marks from above RDD

```
scala> val flatgradeAvg = gradeAvg.map(x => x._1._1 + "," + x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[14] at map at <console>:32
```

In below step we are using intersection function between flatgradeAvg and flatnameAvg rdd's to find whether any common student is there.
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 flatnameAvg = nameAvg.map(x => x._1 + "," + x._2)
flatnameAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at map at <console>:32

scala> flatnameAvg
res10: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at map at <console>:32

scala> val comman = flatgradeAvg.intersection(flatnameAvg)
comman: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[21] at intersection at <console>:44

scala> comman.foreach(println)

scala>
```