# ASSIGNMENT 17.2

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

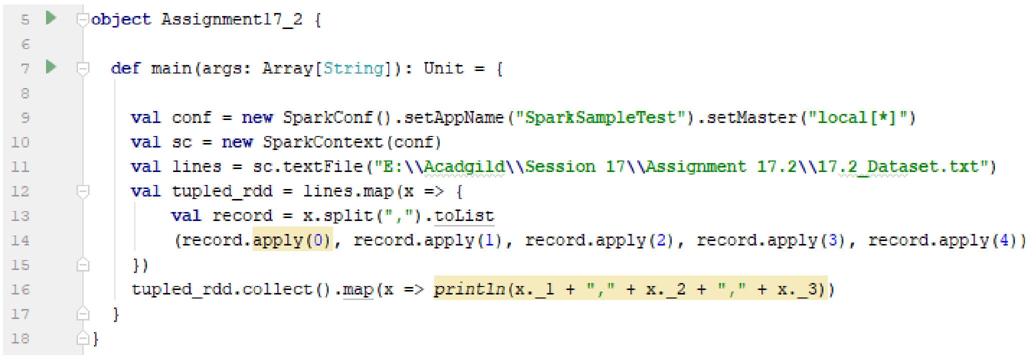
## Problem Statement 1:

1. Read the text file, and create a tupled rdd.
2. Find the count of total number of rows present.
3. What is the distinct number of subjects present in the entire school?
4. What is the count of students in the school, whose name is Mathew and marks are 55?

## Solution:

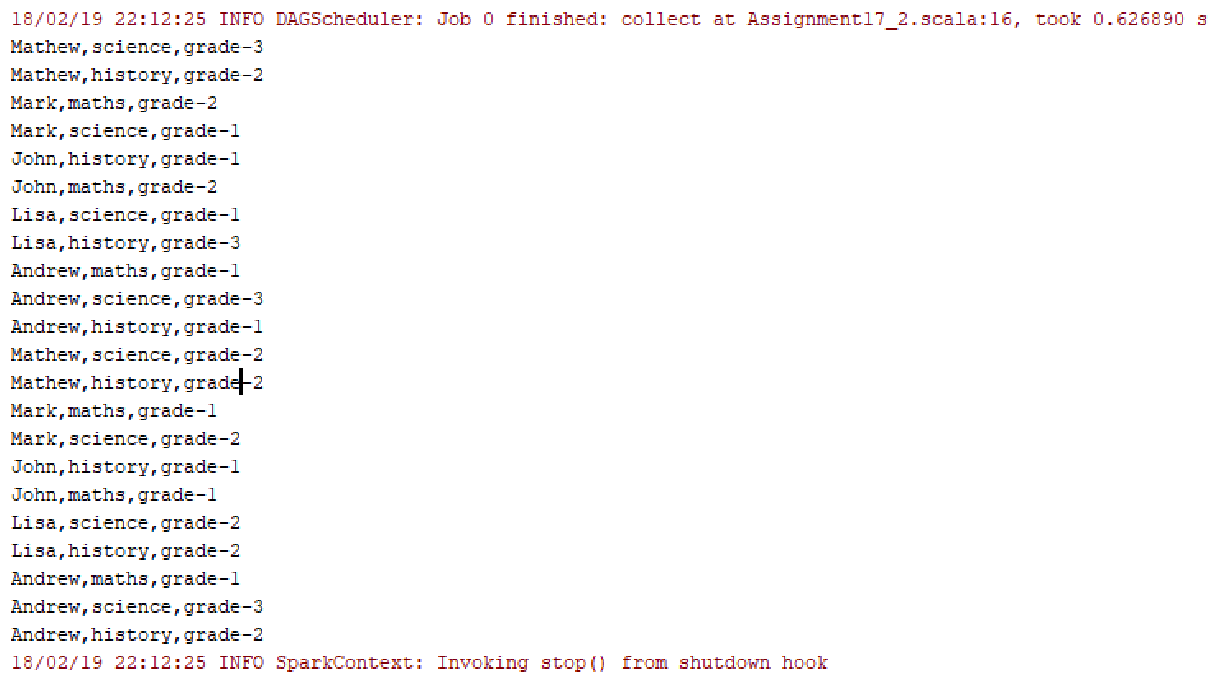
**1.** Here is the code snippet I have written in Scala to create a tupled RDD on given data:

```
object Assignment17_2 {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("SparkSampleTest").setMaster("local[*]")
    val sc = new SparkContext(conf)                // create spark context applying using spark config
    val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
    val tupled_rdd = lines.map(x => {              // apply map() on each line of text file
        val record = x.split(",").toList           //get column values splitting by comma, convert to list
        (record.apply(0), record.apply(1), record.apply(2), record.apply(3), record.apply(4))
    })                                             // access all columns by index in each row
    tupled_rdd.collect().map(x => println(x._1 + "," + x._2 + "," + x._3))     // print few columns
  }                                                                            // from tupled_rdd
}
```



```
object Assignment17_2 {

  def main(args: Array[String]): Unit = {

    val conf = new SparkConf().setAppName("SparkSampleTest").setMaster("local[*]")
    val sc = new SparkContext(conf)
    val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
    val tupled_rdd = lines.map(x => {
        val record = x.split(",").toList
        (record.apply(0), record.apply(1), record.apply(2), record.apply(3), record.apply(4))
    })
    tupled_rdd.collect().map(x => println(x._1 + "," + x._2 + "," + x._3))
  }
}
```

**Output:**



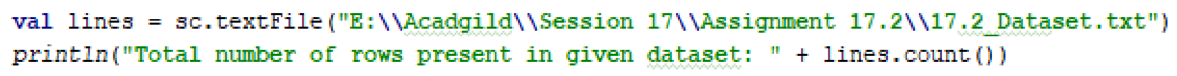
```
18/02/19 22:12:25 INFO DAGScheduler: Job 0 finished: collect at Assignment17_2.scala:16, took 0.626890 s
Mathew,science,grade-3
Mathew,history,grade-2
Mark,maths,grade-2
Mark,science,grade-1
John,history,grade-1
John,maths,grade-2
Lisa,science,grade-1
Lisa,history,grade-3
Andrew,maths,grade-1
Andrew,science,grade-3
Andrew,history,grade-1
Mathew,science,grade-2
Mathew,history,grade-2
Mark,maths,grade-1
Mark,science,grade-2
John,history,grade-1
John,maths,grade-1
Lisa,science,grade-2
Lisa,history,grade-2
Andrew,maths,grade-1
Andrew,science,grade-3
Andrew,history,grade-2
18/02/19 22:12:25 INFO SparkContext: Invoking stop() from shutdown hook
```

**2.** Spark code in Scala to find the count of total number of rows present:

val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

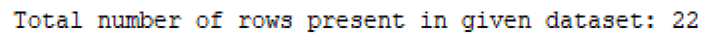
println("Total number of rows present in given dataset: " + lines.count())

// apply count() on lines RDD to get number of rows



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
println("Total number of rows present in given dataset: " + lines.count())
```

**Output:**



```
Total number of rows present in given dataset: 22
```

**3.** Spark code snippet in Scala to find the distinct number of subjects present in the entire school?

val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

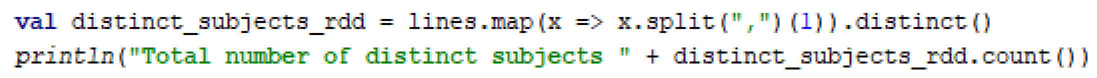
val distinct_subjects_rdd = lines.map(x => x.split(",") (1)).distinct() // split each line by comma

// get subject column by its index (1)

// apply distinct() on it to remove duplicate entries

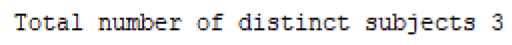
println("Total number of distinct subjects " + distinct_subjects_rdd.count())

// invoke count() on previous RDD to get the count of distinct subjects



```
val distinct_subjects_rdd = lines.map(x => x.split(",")(1)).distinct()
println("Total number of distinct subjects " + distinct_subjects_rdd.count())
```

**Output:**



```
Total number of distinct subjects 3
```

**4.** Spark code snippet to get count of students in school whose name is Mathew and marks are 55:

val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

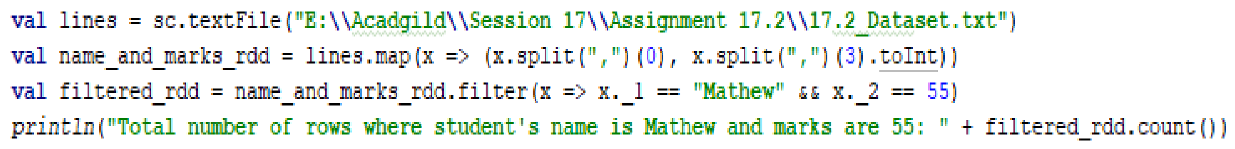
val name_and_marks_rdd = lines.map(x => (x.split(",")(0), x.split(",")(3).toInt))

// split each line by comma, get student's name and marks by their index

val filtered_rdd = name_and_marks_rdd.filter(x => x._1 == "Mathew" && x._2 == 55)

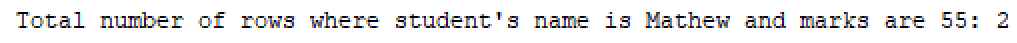
// apply filter() to get only those records with name 'Mathew' and marks '55'

println("Total number of rows where student's name is Mathew and marks are 55: " + filtered_rdd.count()) // invoke **action,** count() to get number of entries with given filter conditions



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_and_marks_rdd = lines.map(x => (x.split(",")(0), x.split(",")(3).toInt))
val filtered_rdd = name_and_marks_rdd.filter(x => x._1 == "Mathew" && x._2 == 55)
println("Total number of rows where student's name is Mathew and marks are 55: " + filtered_rdd.count())
```

**Output:**



```
Total number of rows where student's name is Mathew and marks are 55: 2
```

**Problem Statement 2:**

1. What is the count of students per grade in the school?
2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)
3. What is the average score of students in each subject across all grades?
4. What is the average score of students in each subject per grade?
5. For all students in grade-2, how many have average score greater than 50?

## Solution:

**1.** Spark code snippet in Scala to get the count of students per grade in the school:

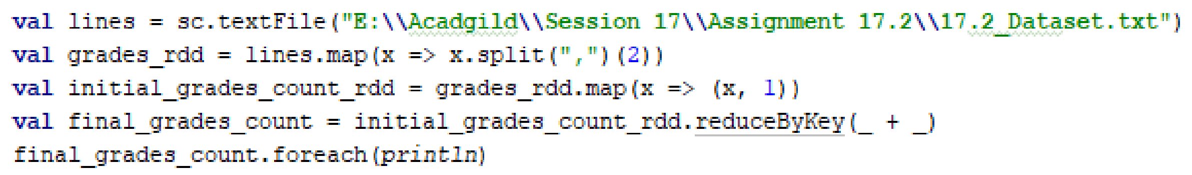
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

val grades_rdd = lines.map(x => x.split(",") (2))  // split each line by comma, get student’s
// grade by its index position

val initial_grades_count_rdd = grades_rdd.map(x => (x, 1))  // initialize count for each grade

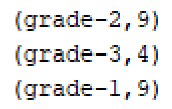
val final_grades_count = initial_grades_count_rdd.reduceByKey(_ + _) // sum up count for each grade

final_grades_count.foreach(println)  // print count of students for each grade



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val grades_rdd = lines.map(x => x.split(",")(2))
val initial_grades_count_rdd = grades_rdd.map(x => (x, 1))
val final_grades_count = initial_grades_count_rdd.reduceByKey(_ + _)
final_grades_count.foreach(println)
```

**Output:**



```
(grade-2,9)
(grade-3,4)
(grade-1,9)
```

**2.** Spark code in Scala to find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)

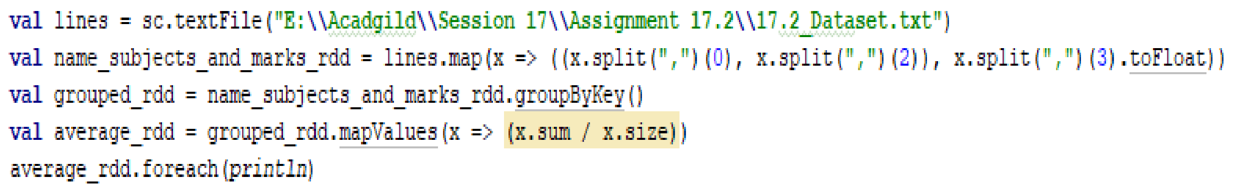
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

val name_grades_and_marks_rdd = lines.map(x => ((x.split(",") (0), x.split(",") (2)), x.split(",") (3).toFloat))  // split each line by comma, get student’s name,
// grade and marks by their index in text file

val grouped_rdd = name_grades_and_marks_rdd.groupByKey()
// group marks based on the (name, grade) pair

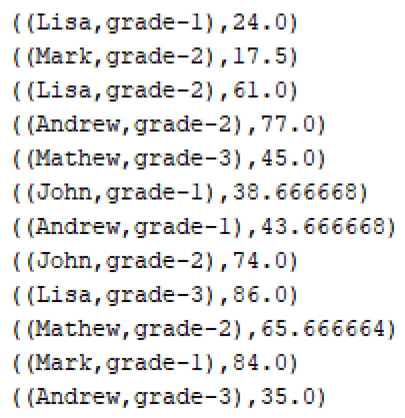
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
// compute average for each grade of every student

average_marks_rdd.foreach(println)  // print average of marks scored by each student



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_subjects_and_marks_rdd = lines.map(x => ((x.split(",")(0), x.split(",")(2)), x.split(",")(3).toFloat))
val grouped_rdd = name_subjects_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_rdd.foreach(println)
```

**Output:**



```
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Mathew,grade-3),45.0)
((John,grade-1),38.666668)
((Andrew,grade-1),43.666668)
((John,grade-2),74.0)
((Lisa,grade-3),86.0)
((Mathew,grade-2),65.666664)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
```

**3.** Spark code in Scala to get the average score of students in each subject across all grades:

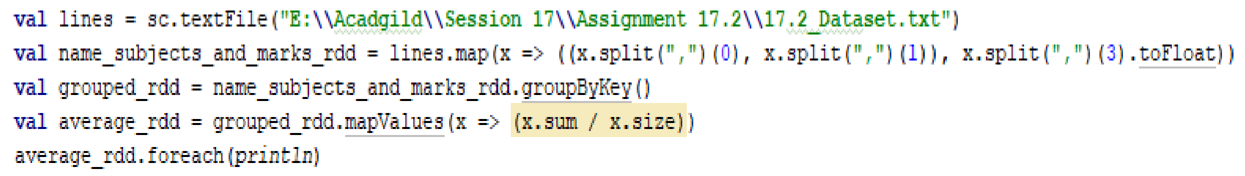
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

val name_subject_and_marks_rdd = lines.map(x => ((x.split(",") (0), x.split(",") (1)), x.split(",") (3).toFloat)) // split each line by comma, get student's name, // subject and marks by their index in text file

val grouped_rdd = name_subject_and_marks_rdd.groupByKey() // group marks based on the (name, subject) pair

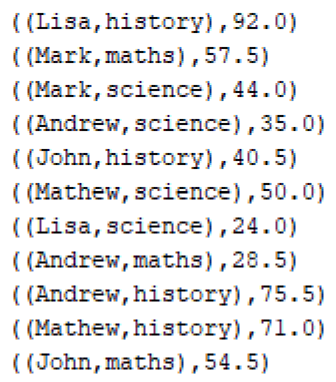
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size)) // compute average for each subject of every student

average_marks_rdd.foreach(println) // print average of marks scored by each student



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_subjects_and_marks_rdd = lines.map(x => ((x.split(",")(0), x.split(",")(1)), x.split(",")(3).toFloat))
val grouped_rdd = name_subjects_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_rdd.foreach(println)
```

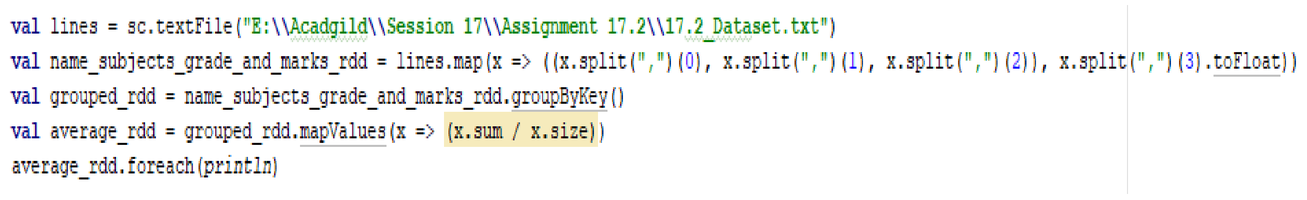
**Output:**



```
((Lisa,history),92.0)
((Mark,maths),57.5)
((Mark,science),44.0)
((Andrew,science),35.0)
((John,history),40.5)
((Mathew,science),50.0)
((Lisa,science),24.0)
((Andrew,maths),28.5)
((Andrew,history),75.5)
((Mathew,history),71.0)
((John,maths),54.5)
```

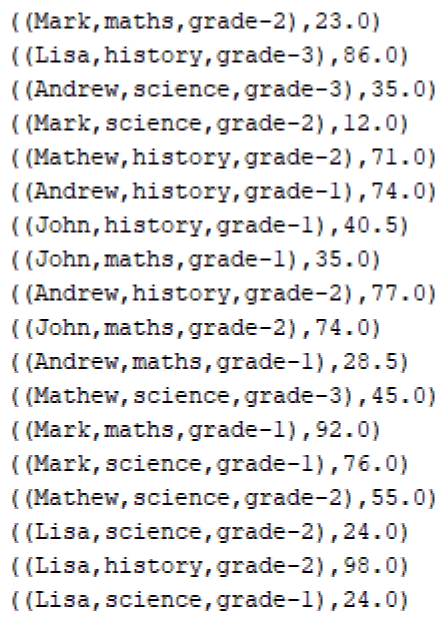
**4.** Spark code in Scala to get the average score of students in each subject per grade:

```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_subject_grade_and_marks_rdd = lines.map(x => ((x.split(",") (0), x.split(",") (1),
x.split(",") (2)), x.split(",") (3).toFloat))        // split each line by comma, get student's name,
                                                      // subject, grade and marks by their index in text file
val grouped_rdd = name_subject_grade_and_marks_rdd.groupByKey()
                                                    // group marks based on the (name, subject, grade) pair
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
                                                      // compute average for each grade of every subject
                                                      // for every student
average_marks_rdd.foreach(println)                    // print average of marks scored by each student
```



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_subjects_grade_and_marks_rdd = lines.map(x => ((x.split(",")(0), x.split(",")(1), x.split(",")(2)), x.split(",")(3).toFloat))
val grouped_rdd = name_subjects_grade_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_rdd.foreach(println)
```

**Output:**



```
((Mark,maths,grade-2),23.0)
((Lisa,history,grade-3),86.0)
((Andrew,science,grade-3),35.0)
((Mark,science,grade-2),12.0)
((Mathew,history,grade-2),71.0)
((Andrew,history,grade-1),74.0)
((John,history,grade-1),40.5)
((John,maths,grade-1),35.0)
((Andrew,history,grade-2),77.0)
((John,maths,grade-2),74.0)
((Andrew,maths,grade-1),28.5)
((Mathew,science,grade-3),45.0)
((Mark,maths,grade-1),92.0)
((Mark,science,grade-1),76.0)
((Mathew,science,grade-2),55.0)
((Lisa,science,grade-2),24.0)
((Lisa,history,grade-2),98.0)
((Lisa,science,grade-1),24.0)
```

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

val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

val grade_2_rdd = lines.filter(x => (x.split(",")(2) == "grade-2"))
// Fetch only those records which have grade-2

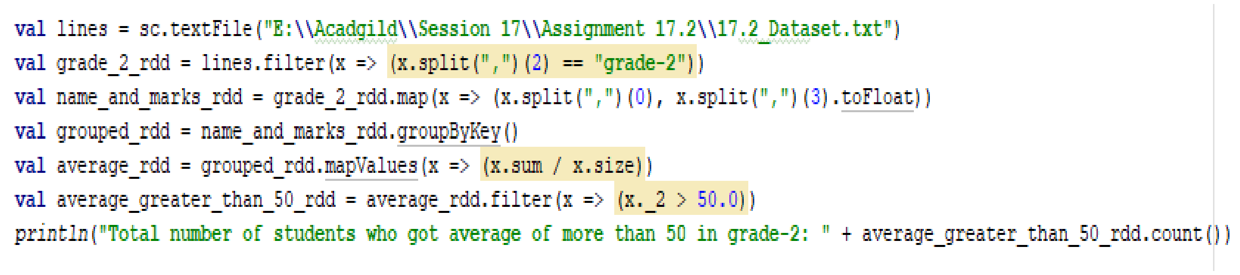
val name_and_marks_rdd = grade_2_rdd.map(x => (x.split(",")(0), x.split(",")(3).toFloat))
// split each line by comma, get student's name,
// and marks by their index in text file

val grouped_rdd = name_and_marks_rdd.groupByKey() // group marks based on the name

val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size)) // compute average of marks

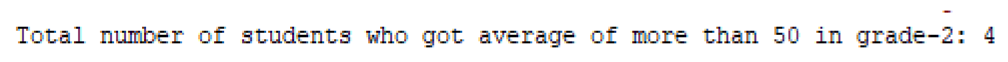
val average_greater_than_50_rdd = average_rdd.filter(x => (x._2 > 50.0))
// Fetch only those records having average greater than 50

println("Total number of students who got average of more than 50 in grade-2: " + average_greater_than_50_rdd.count()) // print the count of students having average greater
// than 50 in grade-2



```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val grade_2_rdd = lines.filter(x => (x.split(",")(2) == "grade-2"))
val name_and_marks_rdd = grade_2_rdd.map(x => (x.split(",")(0), x.split(",")(3).toFloat))
val grouped_rdd = name_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
val average_greater_than_50_rdd = average_rdd.filter(x => (x._2 > 50.0))
println("Total number of students who got average of more than 50 in grade-2: " + average_greater_than_50_rdd.count())
```

**Output:**



```
Total number of students who got average of more than 50 in grade-2: 4
```

**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.

Here is the Spark code snippet in Scala to perform required operations:

val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")

val name_and_marks_rdd = lines.map(x => (x.split(",") (0), x.split(",") (3).toFloat))

// split each line by comma, get student's name, and marks by their index in text file

val grouped_rdd = name_and_marks_rdd.groupByKey()  // group marks based on the name

val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))  // compute average of marks

average_rdd.foreach(println)  // print average score per student_name across all grades

val name_grade_and_marks_rdd = lines.map(x => ((x.split(",") (0), x.split(",") (2)), x.split(",") (3). toFloat))  // split each line by comma, get student's name, grade and marks by their index

val grouped_rdd2 = name_grade_and_marks_rdd.groupByKey()  // group marks based on the name

val average_rdd2 = grouped_rdd2.mapValues(x => (x.sum / x.size))  // compute average of marks

val simplified_average_rdd2 = average_rdd2.map(x => (x._1._1, x._2))  // get rid of grade

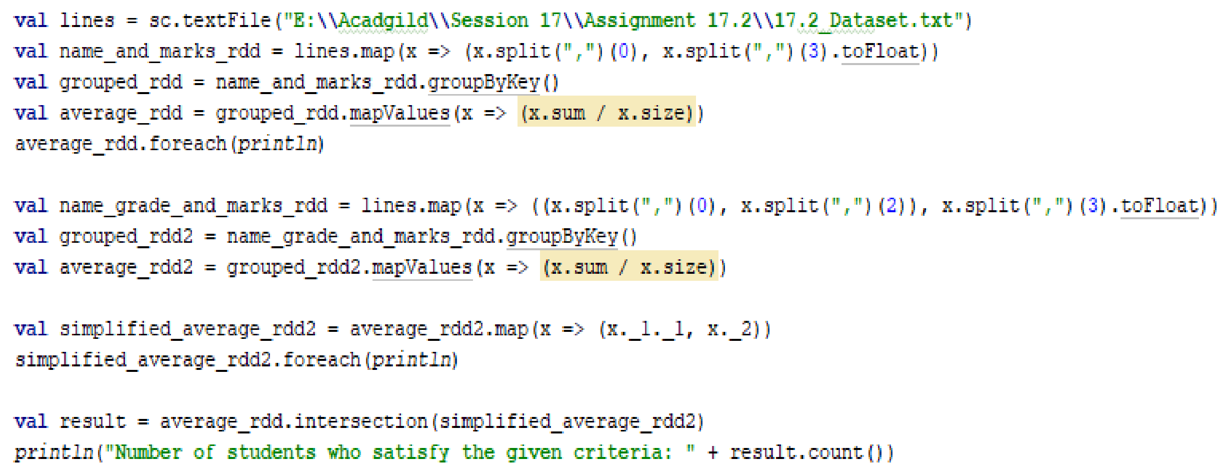
// values in the average_rdd2 to match the format of other RDD consists of average across all grades

simplified_average_rdd2.foreach(println)  // print average score per student_name per grade

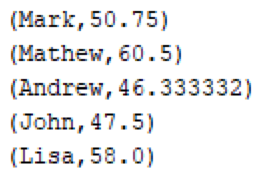
val res = average_rdd.intersection(simplified_average_rdd2)

// apply intersection() to check results for given criteria

println("Number of students who satisfy the given criteria: " + result.count())

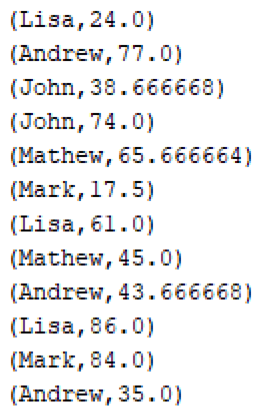

```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_and_marks_rdd = lines.map(x => (x.split(",")(0), x.split(",")(3).toFloat))
val grouped_rdd = name_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_rdd.foreach(println)

val name_grade_and_marks_rdd = lines.map(x => ((x.split(",")(0), x.split(",")(2)), x.split(",")(3).toFloat))
val grouped_rdd2 = name_grade_and_marks_rdd.groupByKey()
val average_rdd2 = grouped_rdd2.mapValues(x => (x.sum / x.size))

val simplified_average_rdd2 = average_rdd2.map(x => (x._1._1, x._2))
simplified_average_rdd2.foreach(println)

val result = average_rdd.intersection(simplified_average_rdd2)
println("Number of students who satisfy the given criteria: " + result.count())
```

**Output for average score per student_name across all grades:**



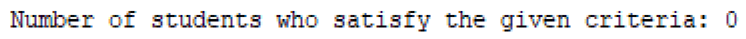
```
(Mark,50.75)
(Mathew,60.5)
(Andrew,46.333332)
(John,47.5)
(Lisa,58.0)
```

**Output for average score per student_name per grade:**



```
(Lisa,24.0)
(Andrew,77.0)
(John,38.666668)
(John,74.0)
(Mathew,65.666664)
(Mark,17.5)
(Lisa,61.0)
(Mathew,45.0)
(Andrew,43.666668)
(Lisa,86.0)
(Mark,84.0)
(Andrew,35.0)
```

**Final Result:**



```
Number of students who satisfy the given criteria: 0
```

Hence, we can conclude that **there are no students in the college who satisfy the given criteria** of falling under both of these results on average scores.