Assignment 17.2

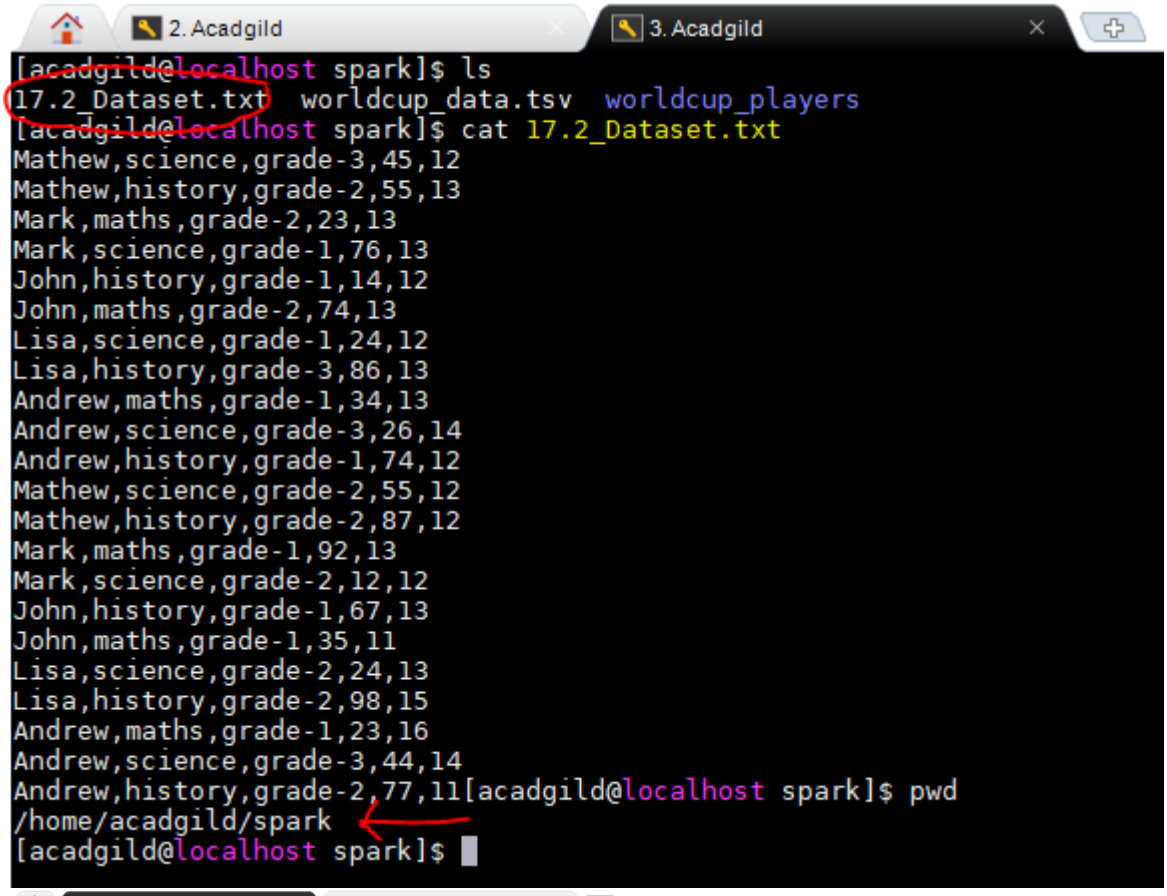
Problem Statement 1:

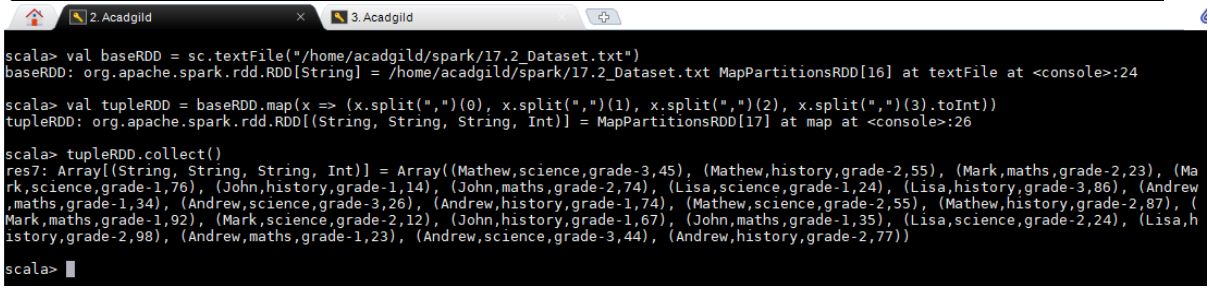
1. Read the text file, and create a tupled rdd.

val baseRDD = sc.textFile("/home/acadgild/spark/17.2_Dataset.txt")

val tupleRDD = baseRDD.map(x => (x.split(",")(0), x.split(",")(1), x.split(",")(2), x.split(",")(3).toInt))

tupleRDD.collect()

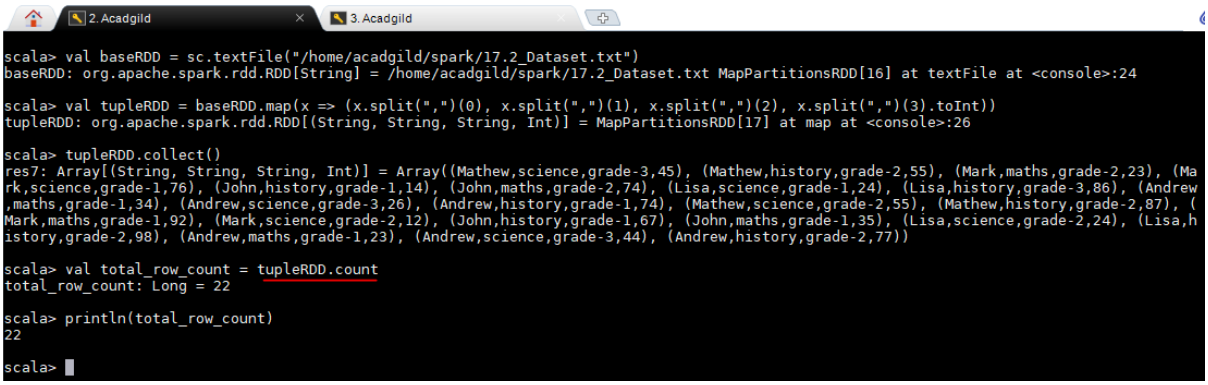




2. Find the count of total number of rows present.

val total_row_count = tupleRDD.count

println(total_row_count)



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

val distinctSubjectRDD = tupleRDD.map(t=> t._2).distinct

distinctSubjectRDD.collect

val distinct_subject_count = distinctSubjectRDD.count

println(distinct_subject_count)

```
scala> val distinctSubjectRDD = tupleRDD.map(t=> t._2).distinct
distinctSubjectRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[25] at distinct at <console>:28

scala> distinctSubjectRDD.collect
res15: Array[String] = Array(maths, history, science)

scala> val distinct_subject_count = distinctSubjectRDD.count
distinct_subject_count: Long = 3

scala> println(distinct_subject_count)
3

scala>
```

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

val studentRDD = tupleRDD.filter(t=> t._1 == "Mathew" && t._4 == 55)

studentRDD.collect

val student_count = studentRDD.count

println(student_count)

```
scala> val studentRDD = tupleRDD.filter(t=> t._1 == "Mathew" && t._4 == 55)
studentRDD: org.apache.spark.rdd.RDD[(String, String, String, Int)] = MapPartitionsRDD[26] at filter at <console>:28

scala> studentRDD.collect
res17: Array[(String, String, String, Int)] = Array((Mathew,history,grade-2,55), (Mathew,science,grade-2,55))

scala> val student_count = studentRDD.count
student_count: Long = 2

scala> println(student_count)
2

scala>
```

Problem Statement 2:

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

val gradeStudentRDD = tupleRDD.map(t => (t._3, t._1))

val distinctGradeStudentRDD = gradeStudentRDD.distinct

val distinctGradeStudentMapCountRDD = distinctGradeStudentRDD.map(t => (t._1, 1))

val gradeStudentCountRDD = distinctGradeStudentMapCountRDD.reduceByKey((x, y) => x+y)

val sortedGradeStudentCountRDD = gradeStudentCountRDD.sortByKey()

sortedGradeStudentCountRDD.collect

sortedGradeStudentCountRDD.foreach(println)

```
scala> val gradeStudentRDD = tupleRDD.map(t => (t._3, t._1))
gradeStudentRDD: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[28] at map at <console>:28

scala> val distinctGradeStudentRDD = gradeStudentRDD.distinct
distinctGradeStudentRDD: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[31] at distinct at <console>:30

scala> val distinctGradeStudentMapCountRDD = distinctGradeStudentRDD.map(t => (t._1, 1))
distinctGradeStudentMapCountRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[32] at map at <console>:32

scala> val gradeStudentCountRDD = distinctGradeStudentMapCountRDD.reduceByKey((x, y) => x+y)
gradeStudentCountRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[33] at reduceByKey at <console>:34

scala> val sortedGradeStudentCountRDD = gradeStudentCountRDD.sortByKey()
sortedGradeStudentCountRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[34] at sortByKey at <console>:36

scala> sortedGradeStudentCountRDD.collect
res19: Array[(String, Int)] = Array((grade-1,4), (grade-2,5), (grade-3,3))

scala> sortedGradeStudentCountRDD.foreach(println)
(grade-1,4)
(grade-2,5)
(grade-3,3)

scala>
```

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

val studentGradeMarksRDD = tupleRDD.map(t=> ((t._1, t._3), (t._4, 1)))

val totalMarksCountByStudentRDD = studentGradeMarksRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val averageByStudentRDD = totalMarksCountByStudentRDD.map(t=> (t._1._1, t._2._1.toFloat / t._2._2.toFloat))

averageByStudentRDD.collect

averageByStudentRDD.foreach(println)

```
scala> val studentGradeMarksRDD = tupleRDD.map(t=> ((t._1, t._3), (t._4, 1)))
studentGradeMarksRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[6] at map at <console>:28

scala> val totalMarksCountByStudentRDD = studentGradeMarksRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
totalMarksCountByStudentRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[7] at reduceByKey at <console>:30

scala> val averageByStudentRDD = totalMarksCountByStudentRDD.map(t=> (t._1._1, t._2._1.toFloat / t._2._2.toFloat))
averageByStudentRDD: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[8] at map at <console>:32

scala> averageByStudentRDD.collect
res5: Array[(String, Float)] = Array((Lisa,24.0), (Mark,17.5), (Lisa,61.0), (Mathew,45.0), (Andrew,77.0), (Andrew,43.666668), (Lisa,86.0)
, (John,38.666668), (John,74.0), (Mark,84.0), (Andrew,35.0), (Mathew,65.666664))

scala> averageByStudentRDD.foreach(println)
(Lisa,24.0)
(Mark,17.5)
(Lisa,61.0)
(Mathew,45.0)
(Andrew,77.0)
(Andrew,43.666668)
(Lisa,86.0)
(John,38.666668)
(John,74.0)
(Mark,84.0)
(Andrew,35.0)
(Mathew,65.666664)

scala>
```

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

val studentMarksCountBySubjectRDD = tupleRDD.map(t=> (t._2, (t._4, 1)))

val totalMarksCountBySubjectRDD = studentMarksCountBySubjectRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val averageBySubjectRDD = totalMarksCountBySubjectRDD.map(t=> (t._1, t._2._1.toFloat / t._2._2.toFloat))

averageBySubjectRDD.collect

averageBySubjectRDD.foreach(println)

```
scala> val studentMarksCountBySubjectRDD = tupleRDD.map(t=> (t._2, (t._4, 1)))
studentMarksCountBySubjectRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[9] at map at <console>:28

scala> val totalMarksCountBySubjectRDD = studentMarksCountBySubjectRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
totalMarksCountBySubjectRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[10] at reduceByKey at <console>:30

scala> val averageBySubjectRDD = totalMarksCountBySubjectRDD.map(t=> (t._1, t._2._1.toFloat / t._2._2.toFloat))
averageBySubjectRDD: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[11] at map at <console>:32

scala> averageBySubjectRDD.collect
res10: Array[(String, Float)] = Array((maths,46.833332), (history,69.75), (science,38.25))

scala> averageBySubjectRDD.foreach(println)
(maths,46.833332)
(history,69.75)
(science,38.25)

scala>
```

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

val studentMarksCountBySubjectAndGradeRDD = tupleRDD.map(t=> ((t._2, t._3), (t._4, 1)))

val totalMarksCountBySubjectAndGradeRDD = studentMarksCountBySubjectAndGradeRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val averageBySubjectAndGradeRDD = totalMarksCountBySubjectAndGradeRDD.map(t=> (t._1, t._2._1.toFloat / t._2._2.toFloat))

averageBySubjectAndGradeRDD.collect

averageBySubjectAndGradeRDD.foreach(println)

```
scala> val studentMarksCountBySubjectAndGradeRDD = tupleRDD.map(t=> ((t._2, t._3), (t._4, 1)))
studentMarksCountBySubjectAndGradeRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[20] at map at <console
>:28

scala> val totalMarksCountBySubjectAndGradeRDD = studentMarksCountBySubjectAndGradeRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
totalMarksCountBySubjectAndGradeRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[21] at reduceByKey at <consol
e>:30

scala> val averageBySubjectAndGradeRDD = totalMarksCountBySubjectAndGradeRDD.map(t=> (t._1, t._2._1.toFloat / t._2._2.toFloat))
averageBySubjectAndGradeRDD: org.apache.spark.rdd.RDD[((String, String), Float)] = MapPartitionsRDD[22] at map at <console>:32

scala> averageBySubjectAndGradeRDD.collect
res17: Array[((String, String), Float)] = Array(((history,grade-2),79.25), ((history,grade-3),86.0), ((maths,grade-1),46.0), ((science,gr
ade-3),38.333332), ((science,grade-1),50.0), ((science,grade-2),30.333334), ((history,grade-1),51.666668), ((maths,grade-2),48.5))

scala> averageBySubjectAndGradeRDD.foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),38.333332)
((science,grade-1),50.0)
((science,grade-2),30.333334)
((history,grade-1),51.666668)
((maths,grade-2),48.5)

scala>
```

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

val grade2StudentRDD = tupleRDD.filter(t=> t._3 == "grade-2")

val grade2StudentMarksCountByNameRDD = grade2StudentRDD.map(t=> (t._1, (t._4, 1)))

val totalGrade2StudentMarksCountByNameRDD = grade2StudentMarksCountByNameRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val averageGrade2StudentMarksCountByNameRDD = totalGrade2StudentMarksCountByNameRDD.map(t=> (t._1, t._2._1.toFloat / t._2._2.toFloat))

var filterAverageGrade2StudentMarksCountByNameRDD = averageGrade2StudentMarksCountByNameRDD.filter(t=> t._2 > 50)

filterAverageGrade2StudentMarksCountByNameRDD.collect

println(filterAverageGrade2StudentMarksCountByNameRDD.count)

```
scala> val grade2StudentRDD = tupleRDD.filter(t=> t._3 == "grade-2")
grade2StudentRDD: org.apache.spark.rdd.RDD[(String, String, String, Int)] = MapPartitionsRDD[23] at filter at <console>:28

scala> val grade2StudentMarksCountByNameRDD = grade2StudentRDD.map(t=> (t._1, (t._4, 1)))
grade2StudentMarksCountByNameRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[24] at map at <console>:30

scala> val totalGrade2StudentMarksCountByNameRDD = grade2StudentMarksCountByNameRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
totalGrade2StudentMarksCountByNameRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[25] at reduceByKey at <console>:32

scala> val averageGrade2StudentMarksCountByNameRDD = totalGrade2StudentMarksCountByNameRDD.map(t=> (t._1, t._2._1.toFloat / t._2._2.toFlo
at))
averageGrade2StudentMarksCountByNameRDD: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[26] at map at <console>:34

scala> var filterAverageGrade2StudentMarksCountByNameRDD =  averageGrade2StudentMarksCountByNameRDD.filter(t=> t._2 > 50)
filterAverageGrade2StudentMarksCountByNameRDD: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[27] at filter at <console>:36

scala> filterAverageGrade2StudentMarksCountByNameRDD.collect
res19: Array[(String, Float)] = Array((Andrew,77.0), (Mathew,65.666664), (John,74.0), (Lisa,61.0))

scala> println(filterAverageGrade2StudentMarksCountByNameRDD.count)
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.**

Step1: Calculate the average of marks of each student name across grades

Use the code below to calculate the average marks per student across grades. In this final result, map is used to create a key by concatenating name, ---, average marks with value being the average marks

```
val studentMarksCountByNameAcrossGradesRDD = tupleRDD.map(t=> (t._1, (t._4, 1)))
val totalStudentMarksCountByNameAcrossGradesRDD =
studentMarksCountByNameAcrossGradesRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 +
y._2))
val averageStudentMarksCountByNameAcrossGradesRDD =
totalStudentMarksCountByNameAcrossGradesRDD.map(t=> (t._1, t._2._1.toFloat /
t._2._2.toFloat))
val studentAverageMarksNameByKeyAcrossGradesRDD =
averageStudentMarksCountByNameAcrossGradesRDD.map(t=> (t._1 + "---" + t._2, t))
studentAverageMarksNameByKeyAcrossGradesRDD.collect
```

```
scala> val studentMarksCountByNameAcrossGradesRDD = tupleRDD.map(t=> (t._1, (t._4, 1)))
studentMarksCountByNameAcrossGradesRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[28] at map at <console>:28

scala> val totalStudentMarksCountByNameAcrossGradesRDD = studentMarksCountByNameAcrossGradesRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 +
 y._2))
totalStudentMarksCountByNameAcrossGradesRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[29] at reduceByKey at <console>
:30

scala> val averageStudentMarksCountByNameAcrossGradesRDD = totalStudentMarksCountByNameAcrossGradesRDD.map(t=> (t._1, t._2._1.toFloat / t
._2._2.toFloat))
averageStudentMarksCountByNameAcrossGradesRDD: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[30] at map at <console>:32

scala> val studentAverageMarksNameByKeyAcrossGradesRDD = averageStudentMarksCountByNameAcrossGradesRDD.map(t=> (t._1 + "---" + t._2, t))
studentAverageMarksNameByKeyAcrossGradesRDD: org.apache.spark.rdd.RDD[(String, (String, Float))] = MapPartitionsRDD[31] at map at <consol
e>:34

scala> studentAverageMarksNameByKeyAcrossGradesRDD.collect
res21: Array[(String, (String, Float))] = Array((Mark---50.75,(Mark,50.75)), (Andrew---46.333332,(Andrew,46.333332)), (Mathew---60.5,(Mat
hew,60.5)), (John---47.5,(John,47.5)), (Lisa---58.0,(Lisa,58.0)))

scala>
```

Step2: Calculate average marks each student name per grade

Use the code below to calculate the average marks per student across grades. In this final result, map is used to create a key by concatenating name, ---, average marks with value being the average marks

```
val studentMarksCountByNameAndGradeRDD = tupleRDD.map(t=> ((t._1, t._3), (t._4, 1)))

val totalStudentMarksCountByNameAndGradesRDD =
studentMarksCountByNameAndGradeRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val averageStudentMarksCountByNameAndGradeRDD =
totalStudentMarksCountByNameAndGradesRDD.map(t=> (t._1._1, t._2._1.toFloat /
t._2._2.toFloat))

val studentAverageMarksNameByKeyForAGradeRDD =
averageStudentMarksCountByNameAndGradeRDD.map(t=> (t._1 + "---" + t._2, t))

studentAverageMarksNameByKeyForAGradeRDD.collect
```

```
scala> val studentMarksCountByNameAndGradeRDD = tupleRDD.map(t=> ((t._1, t._3), (t._4, 1)))
studentMarksCountByNameAndGradeRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[32] at map at <console>:2
8

scala> val totalStudentMarksCountByNameAndGradesRDD = studentMarksCountByNameAndGradeRDD.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

totalStudentMarksCountByNameAndGradesRDD: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[33] at reduceByKey at <c
onsole>:30

scala> val averageStudentMarksCountByNameAndGradeRDD = totalStudentMarksCountByNameAndGradesRDD.map(t=> (t._1._1, t._2._1.toFloat / t._2.
_2.toFloat))
averageStudentMarksCountByNameAndGradeRDD: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[34] at map at <console>:32

scala> val studentAverageMarksNameByKeyForAGradeRDD = averageStudentMarksCountByNameAndGradeRDD.map(t=> (t._1 + "---" + t._2, t))
studentAverageMarksNameByKeyForAGradeRDD: org.apache.spark.rdd.RDD[(String, (String, Float))] = MapPartitionsRDD[35] at map at <console>:
34

scala> studentAverageMarksNameByKeyForAGradeRDD.collect
res22: Array[(String, (String, Float))] = Array((Lisa---24.0,(Lisa,24.0)), (Mark---17.5,(Mark,17.5)), (Lisa---61.0,(Lisa,61.0)), (Mathew-
--45.0,(Mathew,45.0)), (Andrew---77.0,(Andrew,77.0)), (Andrew---43.666668,(Andrew,43.666668)), (Lisa---86.0,(Lisa,86.0)), (John---38.6666
68,(John,38.666668)), (John---74.0,(John,74.0)), (Mark---84.0,(Mark,84.0)), (Andrew---35.0,(Andrew,35.0)), (Mathew---65.666664,(Mathew,65
.666664)))

scala>
```

Step3: Use set intersection to find common student obtained from step1 and step2

Her use set intersection on studentAverageMarksNameByKeyAcrossGradesRDD, studentAverageMarksNameByKeyForAGradeRDD. Print the common students

val commonStudentsRDD = studentAverageMarksNameByKeyAcrossGradesRDD.intersection(studentAverageMarksNameByKeyForAGradeRDD)

commonStudentsRDD.collect

commonStudentsRDD.foreach(println)

```
scala> val commonStudentsRDD = studentAverageMarksNameByKeyAcrossGradesRDD.intersection(studentAverageMarksNameByKeyForAGradeRDD)
commonStudentsRDD: org.apache.spark.rdd.RDD[(String, (String, Float))] = MapPartitionsRDD[41] at intersection at <console>:44

scala> commonStudentsRDD.collect
res23: Array[(String, (String, Float))] = Array()

scala> commonStudentsRDD.foreach(println)

scala>
```

Note that there is no student whose average score across grades is same as average score per grade

Mathew(Across Grades) - (45 + 55 + 55 + 87)/4 = 60.5

Mathew (grade_2) = 65.66

Methew (grade_3) = 45

Mark (Across Grades) - (23+76+ 92 +12) / 4 = 50.75

Mark (grade_1) = (76+92)/2 = 84

Mark (grade_2) = (23+12)/2 = 17.50

John (Across Grades) - (14+74+67+35) / 4 = 47.59

John (grade_1) - (14+67+35)/3 = 38.66

John (grade_2) - 74

Lisa (Across Grades) - (24+86+24+98)/4 = 58

Lisa(grade_1) - 24

Lisa(grade_2) - (24+98)/2 = 61

Lisa (grade_3) - 86

Andrew (Across Grades) - (34+26+74 +23 +44 +77)/6 = 46.33

Andrew (grade_1) - (34+74+23)/3 = 43.66

Andrew (grade_2) - 77

Andrew (grade_3) - (26+44)/2 = 35. I have manually verified

Mathew(Across Grades) - (45 + 55 + 55 + 87)/4 = 60.5

Mathew (grade_2) = 65.66

Methew (grade_3) = 45

Mark (Across Grades) - (23+76+ 92 +12) / 4 = 50.75

Mark (grade_1) = (76+92)/2 = 84

Mark (grade_2) = (23+12)/2 = 17.50

John (Across Grades) - (14+74+67+35) / 4 = 47.59

John (grade_1) - (14+67+35)/3 = 38.66

John (grade_2) - 74

Lisa (Across Grades) - (24+86+24+98)/4 = 58

Lisa(grade_1) - 24

Lisa(grade_2) - (24+98)/2 = 61

Lisa (grade_3) - 86

Andrew (Across Grades) - (34+26+74 +23 +44 +77)/6 = 46.33

Andrew (grade_1) - (34+74+23)/3 = 43.66

Andrew (grade_2) - 77

Andrew (grade_3) - (26+44)/2 = 35