

TASK 1

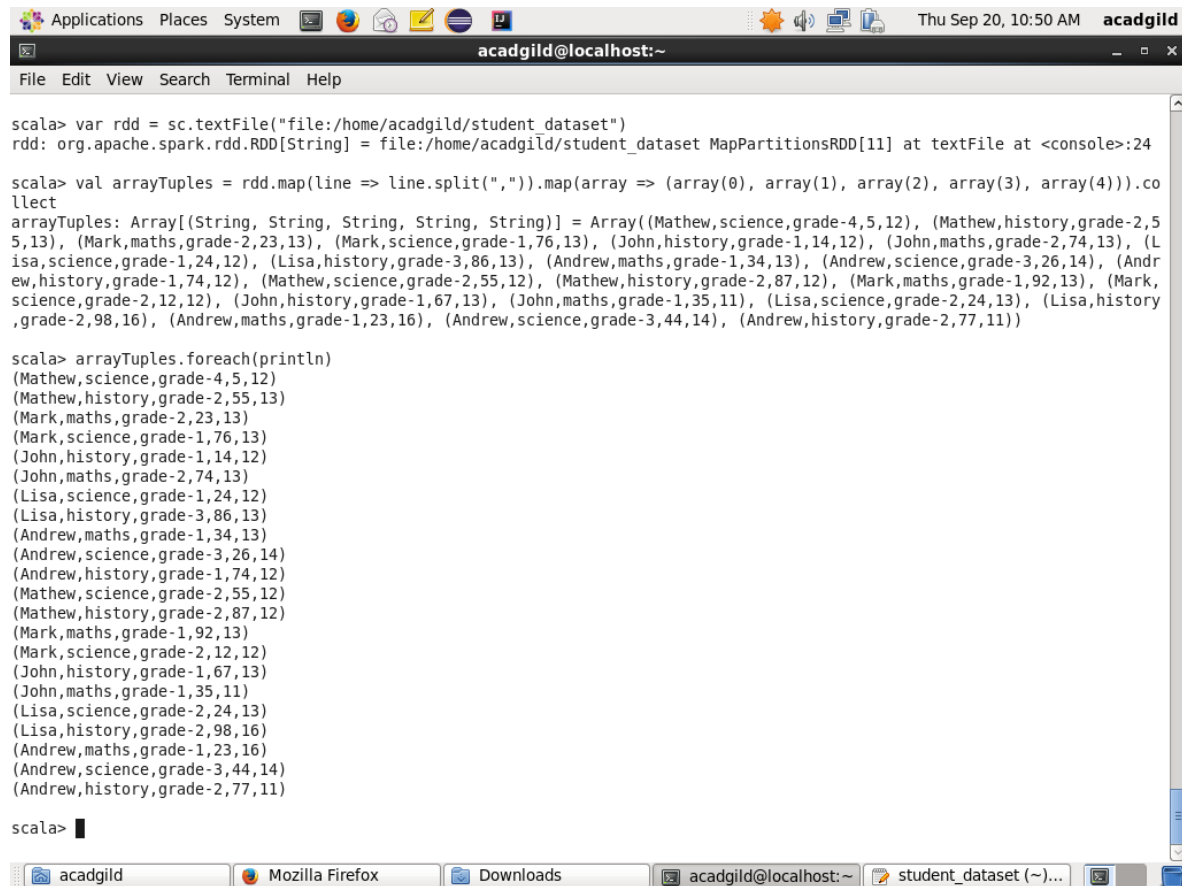
Given a dataset of college students as a text file (name, subject, grade, marks):

```
Mathew,science,grade-4,5,12
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-2,98,16
Andrew,maths,grade-1,23,16
Andrew,science,grade-3,44,14
Andrew,history,grade-2,77,11
```

Problem Statement 1:

1. Read the text file and create a tuple 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 the number of students in the school, whose name is Mathew and marks is 55?

read text file and create tuple RDD.



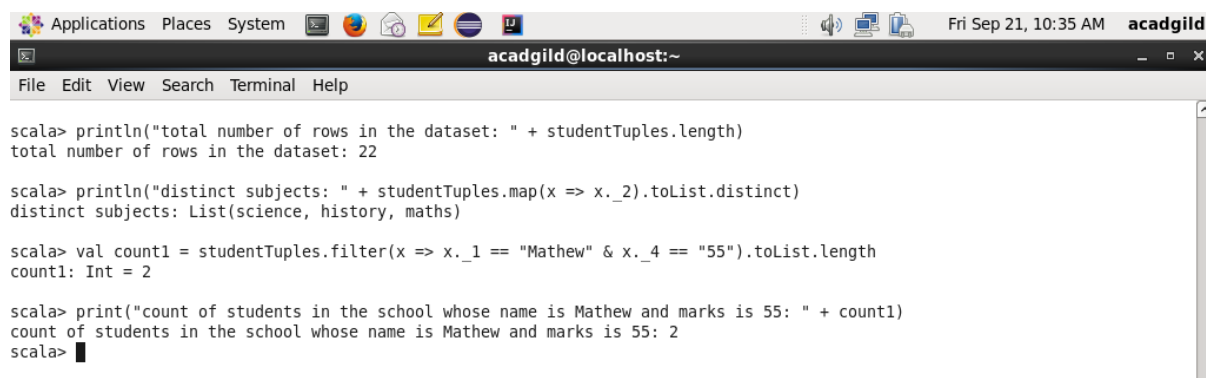
```
scala> var rdd = sc.textFile("file:/home/acadgild/student_dataset")
rdd: org.apache.spark.rdd.RDD[String] = file:/home/acadgild/student_dataset MapPartitionsRDD[11] at textFile at <console>:24

scala> val arrayTuples = rdd.map(line => line.split(",")).map(array => (array(0), array(1), array(2), array(3), array(4))).collect
arrayTuples: Array[(String, String, String, String, String)] = Array((Mathew,science,grade-4,5,12), (Mathew,history,grade-2,5,13), (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-2,98,16), (Andrew,maths,grade-1,23,16), (Andrew,science,grade-3,44,14), (Andrew,history,grade-2,77,11))

scala> arrayTuples.foreach(println)
(Mathew,science,grade-4,5,12)
(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-2,98,16)
(Andrew,maths,grade-1,23,16)
(Andrew,science,grade-3,44,14)
(Andrew,history,grade-2,77,11)

scala>
```

We can find the number of rows in the dataset, distinct number of subjects and count of number of students in school whose name is Mathew and total marks is 55 using simple Scala operations like length, toList and filter.



```
scala> println("total number of rows in the dataset: " + studentTuples.length)
total number of rows in the dataset: 22

scala> println("distinct subjects: " + studentTuples.map(x => x._2).toList.distinct)
distinct subjects: List(science, history, maths)

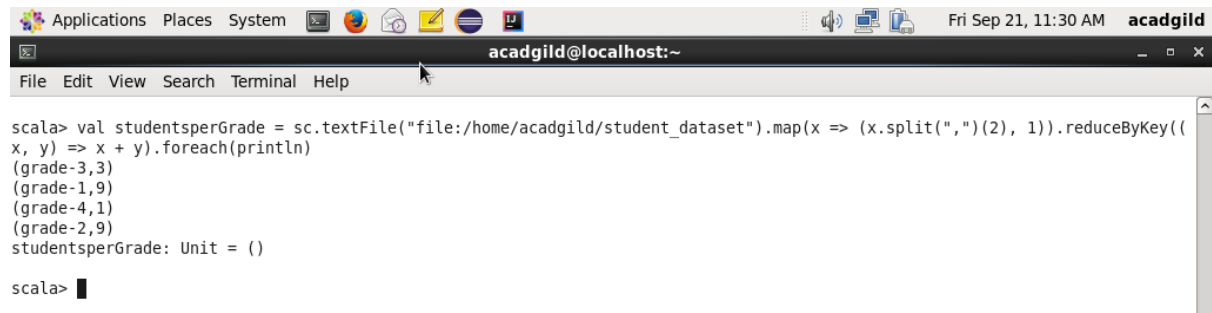
scala> val count1 = studentTuples.filter(x => x._1 == "Mathew" & x._4 == "55").toList.length
count1: Int = 2

scala> print("count of students in the school whose name is Mathew and marks is 55: " + count1)
count of students in the school whose name is Mathew and marks is 55: 2
scala>
```

Problem Statement 2:

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

Create a tuple RDD with grade as the key and assign the numerical value 1 for each using map. Calculate the count per grade using reduceByKey and the operation sum.

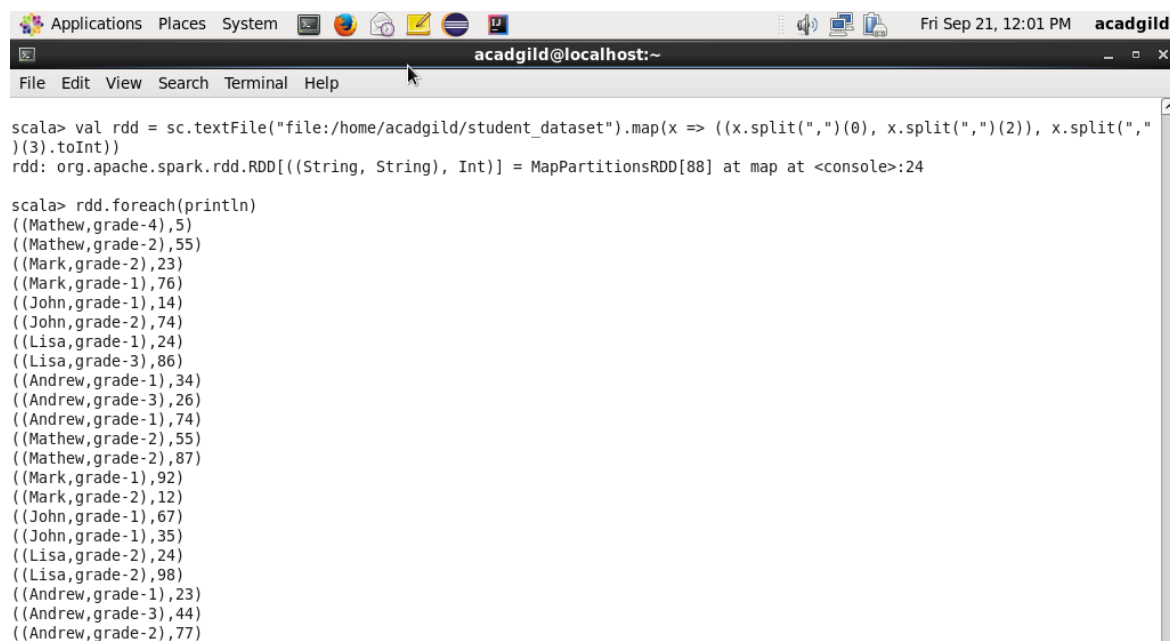
A screenshot of a terminal window titled 'acadgild@localhost:~'. The terminal shows the following Scala code and its output:

```
scala> val studentsperGrade = sc.textFile("file:/home/acadgild/student_dataset").map(x => (x.split(",")(2), 1)).reduceByKey((x, y) => x + y).foreach(println)
(grade-3,3)
(grade-1,9)
(grade-4,1)
(grade-2,9)
studentsperGrade: Unit = ()

scala>
```

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

Create an RDD with name and grade as key and marks as value.

A screenshot of a terminal window titled 'acadgild@localhost:~'. The terminal shows the following Scala code and its output:

```
scala> val rdd = sc.textFile("file:/home/acadgild/student_dataset").map(x => ((x.split(",")(0), x.split(",")(2)), x.split(",")(3).toInt))
rdd: org.apache.spark.rdd.RDD[(String, String), Int] = MapPartitionsRDD[88] at map at <console>:24

scala> rdd.foreach(println)
((Mathew,grade-4),5)
((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)
((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)
((Andrew,grade-1),23)
((Andrew,grade-3),44)
((Andrew,grade-2),77)
```

Map each value with 1.

Use reduceByKey to add the occurrences of marks for each key which is student name and grade.



```
Applications  Places  System  acadgild@localhost:~
File Edit View Search Terminal Help

scala> rddMap.foreach(println)
((Mathew,grade-4),(5,1))
((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))

scala> val rddreduce = rddMap.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rddreduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[90] at reduceByKey at <console>:25
```

Calculate the average by adding marks and dividing by its count for each key.

```
scala> val studentAvg = rddreduce.mapValues{case(sum ,count) => (1.0 * sum) / count}
studentAvg: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[91] at mapValues at <console>:25

scala> studentAvg.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((Mathew,grade-4),5.0)
((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?

Create an RDD with name and subject as key and marks as value.

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
  
scala> val rdd = sc.textFile("file:/home/acadgild/student_dataset").map(x => ((x.split(",")(0), x.split(",")(1)), x.split(",")  
(3).toInt))  
rdd: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[100] at map at <console>:24  
  
scala> rdd.foreach(println)  
((Mathew,science),5)  
((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)
```

Map each value with 1.

```
Applications Places System Fri Sep 21, 6:36 PM  
acadgild@localhost:~  
File Edit View Search Terminal Help  
  
scala> val rddMap = rdd.mapValues(x => (x,1))  
rddMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[102] at mapValues at <console>:25  
  
scala> rddMap.foreach(println)  
((Mathew,science),(5,1))  
((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))  
((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))
```

Use reduceByKey to add the occurrences of marks for each key which is student name and subject. Calculate the average by adding marks and dividing by its count for each key.

```
scala> val rddreduce = rddMap.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rddreduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[104] at reduceByKey at <console>:25

scala> rddreduce.foreach(println)
((Lisa,history),(184,2))
((Mark,maths),(115,2))
((Andrew,science),(70,2))
((Mark,science),(88,2))
((Mathew,science),(60,2))
((Andrew,maths),(57,2))
((Mathew,history),(142,2))
((John,maths),(109,2))
((John,history),(81,2))
((Lisa,science),(48,2))
((Andrew,history),(151,2))

scala> val studAvgAllSubjects = rddreduce.mapValues{case(sum ,count) => (1.0 * sum) / count}
studAvgAllSubjects: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[105] at mapValues at <console>:25

scala> studAvgAllSubjects.foreach(println)
((Lisa,history),92.0)
((Mark,maths),57.5)
((Andrew,science),35.0)
((Mark,science),44.0)
((Mathew,science),30.0)
((Andrew,maths),28.5)
((Mathew,history),71.0)
((John,maths),54.5)
((John,history),40.5)
((Lisa,science),24.0)
((Andrew,history),75.5)
```

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

Create an RDD with subject and grade as key and marks as value.

```
scala> val rdd = sc.textFile("file:/home/acadgild/student_dataset").map(x => ((x.split(",")(1), x.split(",")(2)), x.split(",")
)(3).toInt))
rdd: org.apache.spark.rdd.RDD[(String, String), Int] = MapPartitionsRDD[108] at map at <console>:24

scala> rdd.foreach(println)
((science,grade-4),5)
((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)
```

Map each value with 1.

Use reduceByKey to add the occurrences of marks for each key which is subject and grade.

Calculate the average by adding marks and dividing by its count for each key.

```
scala> val rddMap = rdd.mapValues(x => (x,1))
rddMap: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[109] at mapValues at <console>:25

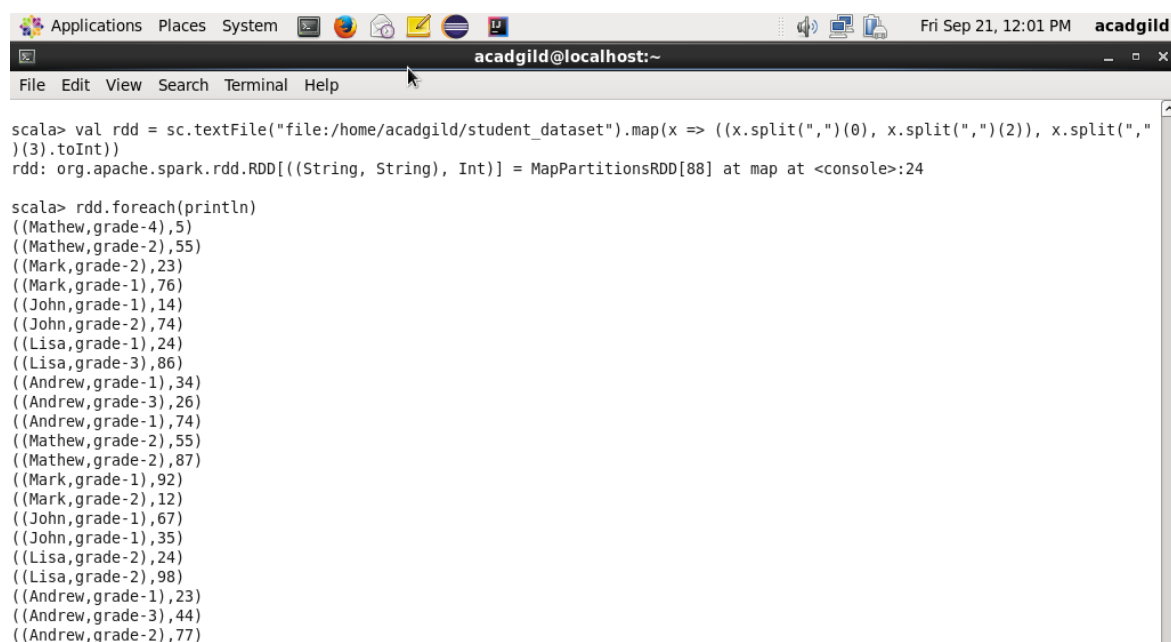
scala> val rddreduce = rddMap.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rddreduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[110] at reduceByKey at <console>:25

scala> val AvgSubjectsPerGrade = rddreduce.mapValues{case(sum ,count) => (1.0 * sum) / count}
AvgSubjectsPerGrade: org.apache.spark.rdd.RDD[(String, String), (Double)] = MapPartitionsRDD[111] at mapValues at <console>:25

scala> AvgSubjectsPerGrade.foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((science,grade-4),5.0)
((maths,grade-1),46.0)
((science,grade-3),35.0)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
```

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

Create an RDD with name and grade as key and marks as value.



```
scala> val rdd = sc.textFile("file:/home/acadgild/student_dataset").map(x => ((x.split(",")(0), x.split(",")(2)), x.split(",")
)(3).toInt))
rdd: org.apache.spark.rdd.RDD[(String, String), (Int)] = MapPartitionsRDD[88] at map at <console>:24

scala> rdd.foreach(println)
((Mathew,grade-4),5)
((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)
((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)
((Andrew,grade-1),23)
((Andrew,grade-3),44)
((Andrew,grade-2),77)
```

Map each value with 1.

Use reduceByKey to add the occurrences of marks for each key which is name and grade.

Calculate the average by adding marks and dividing by its count for each key.

Filter the result with students who are in grade-2 and whose marks are greater than 50.

```
scala> val rddMap = rdd.mapValues(x => (x,1))
rddMap: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[115] at mapValues at <console>:25

scala> val rddreduce = rddMap.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rddreduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[116] at reduceByKey at <console>:25

scala> val rddAverageScore = rddreduce.mapValues{case(sum ,count) => (1.0 * sum) / count}
rddAverageScore: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[117] at mapValues at <console>:25

scala> rddAverageScore.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((Mathew,grade-4),5.0)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.666666666666667)

scala> val rddAverageScoreFilter = rddAverageScore.filter(x => x._1._2 == "grade-2" && x._2 > 50)
rddAverageScoreFilter: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[118] at filter at <console>:25

scala> rddAverageScoreFilter.foreach(println)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.666666666666667)
```

Problem Statement 3:

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.**

Create an RDD with name as key and marks as value

```
scala> val rdd = sc.textFile("file:/home/acadgild/student_dataset").map(x => (x.split(",")(0), x.split(",")(3).toInt))
rdd: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[121] at map at <console>:24

scala> rdd.foreach(println)
(Mathew,5)
(Mathew,55)
(Mark,23)
(Mark,76)
(John,14)
(John,74)
```

Calculate average score per student across all grades as below

```
scala> val rddMap = rdd.mapValues(x => (x,1))
rddMap: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[125] at mapValues at <console>:25

scala> val rddreduce = rddMap.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rddreduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[126] at reduceByKey at <console>:25

scala> val rddStudentAverage = rddreduce.mapValues{case(sum ,count) => (1.0 * sum) / count}
rddStudentAverage: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[127] at mapValues at <console>:25

scala> rddStudentAverage.foreach(println)
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,50.5)
(John,47.5)
(Lisa,58.0)
```

Calculate average score per student per grade as below


```
scala> val rdd = sc.textFile("file:/home/acadgild/student_dataset").map(x => ((x.split(",")(0), x.split(",")(2)), x.split(",")
(3).toInt))
rdd: org.apache.spark.rdd.RDD[(String, String), Int] = MapPartitionsRDD[130] at map at <console>:24

scala> val rddMap = rdd.mapValues(x => (x,1))
rddMap: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[131] at mapValues at <console>:25

scala> val rddreduce = rddMap.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rddreduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[132] at reduceByKey at <console>:25

scala> val rddStudentAveragePerGrade = rddreduce.mapValues{case(sum ,count) => (1.0 * sum) / count}
rddStudentAveragePerGrade: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[133] at mapValues at <console>:25

scala> rddStudentAveragePerGrade.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((Mathew,grade-4),5.0)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.666666666666667)
```

Flatten both the RDDs and use intersection to find the common val.

```
scala> val studentAvgPerGrade_flat = rddStudentAveragePerGrade.map(x => x._1._1 + "," + x._2.toDouble)
studentAvgPerGrade_flat: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[134] at map at <console>:25

scala> val studentAvg_flat = rddStudentAverage.map(x => x._1 + "," + x._2.toDouble)
studentAvg_flat: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[135] at map at <console>:25

scala> val commonval = studentAvgPerGrade_flat.intersection(studentAvg_flat)
commonval: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[141] at intersection at <console>:27

scala> commonval.foreach(println)

scala> █
```

There are no common values.

Task 2

Dataset of travellers: https://drive.google.com/drive/folders/0B_P3pWagdlrrVThBaUdVSUtzbms

1) What is the distribution of the total number of air-travellers per year?

Create an RDD with year of travel as the key and assign a value of 1 to each.

Use reduceByKey to add the values for each key which is travel year.

```
scala> sc.textFile("file:/home/acadgild/Dataset_Holidays.txt").map(x => (x.split(",")(5), 1)).reduceByKey((x, y) => x + y).foreach(println)
(1994,1)
(1990,8)
(1991,9)
(1992,7)
(1993,7)
```

2) What is the total air distance covered by each user per year?

Create an RDD with the user_id and year of travel as key and distance as the value.

Use reduceByKey to add the distance for each key which is (user_id, year of travel).

```
scala> sc.textFile("file:/home/acadgild/Dataset_Holidays.txt").map(x => ((x.split(",")(0).toInt, x.split(",")(5)), x.split(",")(4).toInt)).reduceByKey((x, y) => x + y).sortByKey().foreach(println)
((1,1990),200)
((1,1993),600)
((2,1991),400)
((2,1993),200)
((3,1991),200)
((3,1992),200)
((3,1993),200)
((4,1990),400)
((4,1991),200)
((5,1991),200)
((5,1992),400)
((5,1994),200)
((6,1991),400)
((6,1993),200)
((7,1990),600)
((8,1990),200)
((8,1991),200)
((8,1992),200)
((9,1991),200)
((9,1992),400)
((10,1990),200)
((10,1992),200)
((10,1993),200)
```

3) Which user has travelled the largest distance till date?

Create an RDD with user_id as the key and distance as the value.

Use reduceByKey to add the distance for each key which is user_id.

Sort the RDD by value, by swapping keys and values and using sortByKey(false) for descending sort order.

```
scala> sc.textFile("file:/home/acadgild/Dataset_Holidays.txt").map(x => ((x.split(",")(0).toInt, x.split(",")(4).toInt)).reduceByKey((x, y) => x + y).map(x => x.swap).sortByKey(false).map(x => x.swap).foreach(println)
(1,800)
(5,800)
(4,600)
(6,600)
(3,600)
(7,600)
(9,600)
(8,600)
(10,600)
(2,600)
```

The result shows us that User_id = 1 has travelled the largest distance till date.

4) What is the most preferred destination for all users?

Create an RDD with destination as the key and add 1 as the value for each of them.

Use reduceByKey to add the occurrence for each key which is destination.

Sort the RDD by value, by swapping keys and values and using sortByKey(false) for descending sort order.

```
scala> sc.textFile("file:/home/acadgild/Dataset_Holidays.txt").map(x => (x.split(",")(2), 1)).reduceByKey((x, y) => x + y).map(x => x.swap).sortByKey(false).map(x => x.swap).foreach(println)
(IND,9)
(CHN,7)
(RUS,6)
(PAK,5)
(AUS,5)
```

The result shows us that IND is the preferred destination for all users.

5) Which route is generating the most revenue per year?

To find the most travelled route, create an RDD with source and destination as key and add 1 as the value for each pair.

Use reduceByKey to add the occurrence for each key which is source and destination.

Sort the RDD by value, by swapping keys and values and using sortByKey(false) for descending sort order.

```
scala> sc.textFile("file:/home/acadgild/Dataset_Holidays.txt").map(x => ((x.split(",")(1), x.split(",")(2)), 1)).reduceByKey((x, y) => x + y).map(x => x.swap).sortByKey(false).map(x => x.swap).foreach(println)
((CHN,IND),4)
((CHN,RUS),3)
((RUS,IND),3)
((AUS,CHN),3)
((CHN,PAK),3)
((IND,CHN),2)
((IND,AUS),2)
((IND,RUS),2)
((RUS,CHN),2)
((PAK,IND),1)
((IND,PAK),1)
((CHN,AUS),1)
((PAK,RUS),1)
((AUS,PAK),1)
((PAK,AUS),1)
((RUS,AUS),1)
((AUS,IND),1)
```

6) What is the total amount spent by every user on air-travel per year?

```
scala> sc.textFile("file:/home/acadgild/Dataset_Holidays.txt").map(x => ((x.split(",")(0).toInt, x.split(",")(5)), x.split(",")(4).toInt)).reduceByKey((x, y) => x + y).sortByKey().mapValues(x => x * 170).foreach(println)
((1,1990),34000)
((1,1993),102000)
((2,1991),68000)
((2,1993),34000)
((3,1991),34000)
((3,1992),34000)
((3,1993),34000)
((4,1990),68000)
((4,1991),34000)
((5,1991),34000)
((5,1992),68000)
((5,1994),34000)
((6,1991),68000)
((6,1993),34000)
((7,1990),102000)
((8,1990),34000)
((8,1991),34000)
((8,1992),34000)
((9,1991),34000)
((9,1992),68000)
((10,1990),34000)
((10,1992),34000)
((10,1993),34000)
```