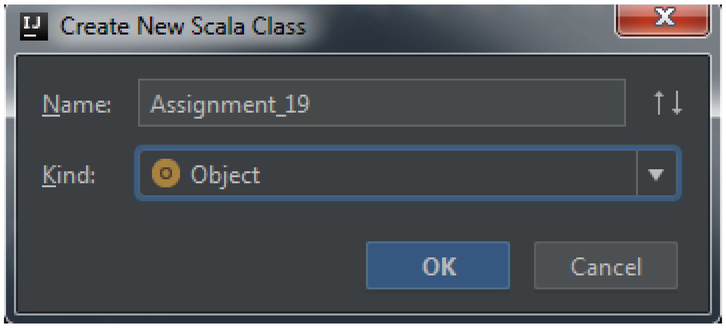
Session 19:
RDD DEEP DIVE
Assignment 1

- Prachi Mohite

Note : We will be using the IDEA IntelliJ to complete Assignment 19.

Created Project in IDEA and added new scala Objects as below



Name: Assignment_19

Kind: Object

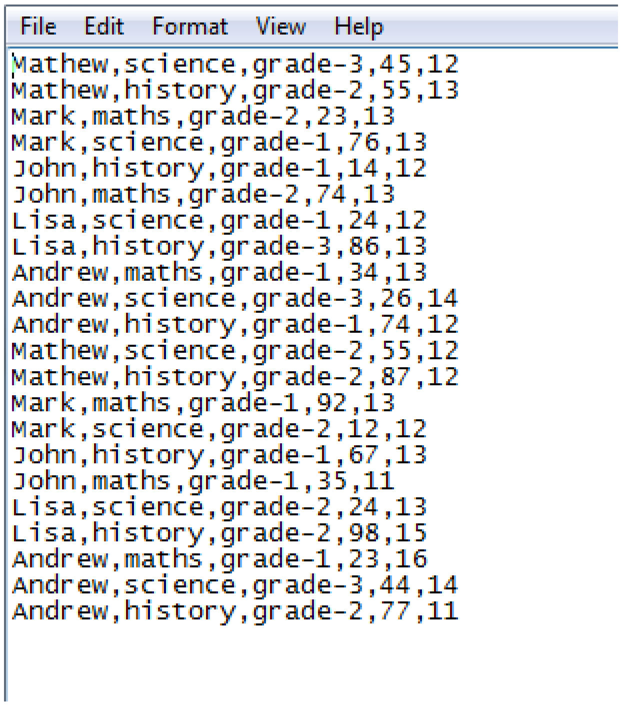
Below is the Dataset getting used for this assignment which has

Column 1 – Student Name

Column 2 – Subject Name

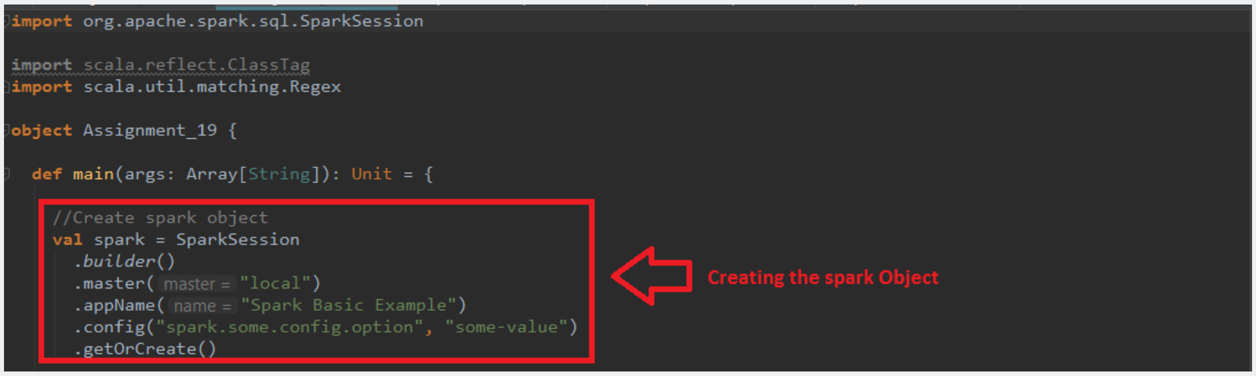
Column 3 – Grade Of a Student

Column 4 – Marks of Student



```
Mathew,science,grade-3,45,12
Mathew,history,grade-2,55,13
Mark,maths,grade-2,23,13
Mark,science,grade-1,76,13
John,history,grade-1,14,12
John,maths,grade-2,74,13
Lisa,science,grade-1,24,12
Lisa,history,grade-3,86,13
Andrew,maths,grade-1,34,13
Andrew,science,grade-3,26,14
Andrew,history,grade-1,74,12
Mathew,science,grade-2,55,12
Mathew,history,grade-2,87,12
Mark,maths,grade-1,92,13
Mark,science,grade-2,12,12
John,history,grade-1,67,13
John,maths,grade-1,35,11
Lisa,science,grade-2,24,13
Lisa,history,grade-2,98,15
Andrew,maths,grade-1,23,16
Andrew,science,grade-3,44,14
Andrew,history,grade-2,77,11
```

Added Below Code to create spark object



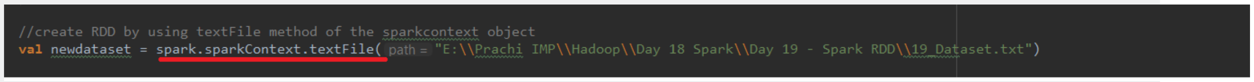
```
import org.apache.spark.sql.SparkSession

import scala.reflect.ClassTag
import scala.util.matching.Regex

object Assignment_19 {

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

    //Create spark object
    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Spark Basic Example")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
```

Task 1

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

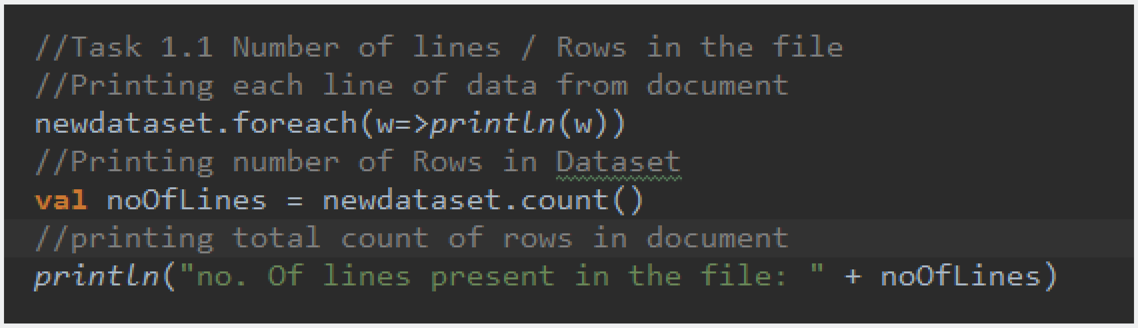
Solution Approach –

1. Read the text file and create the RDD . This can be done by using textFile(path='') method of spark context



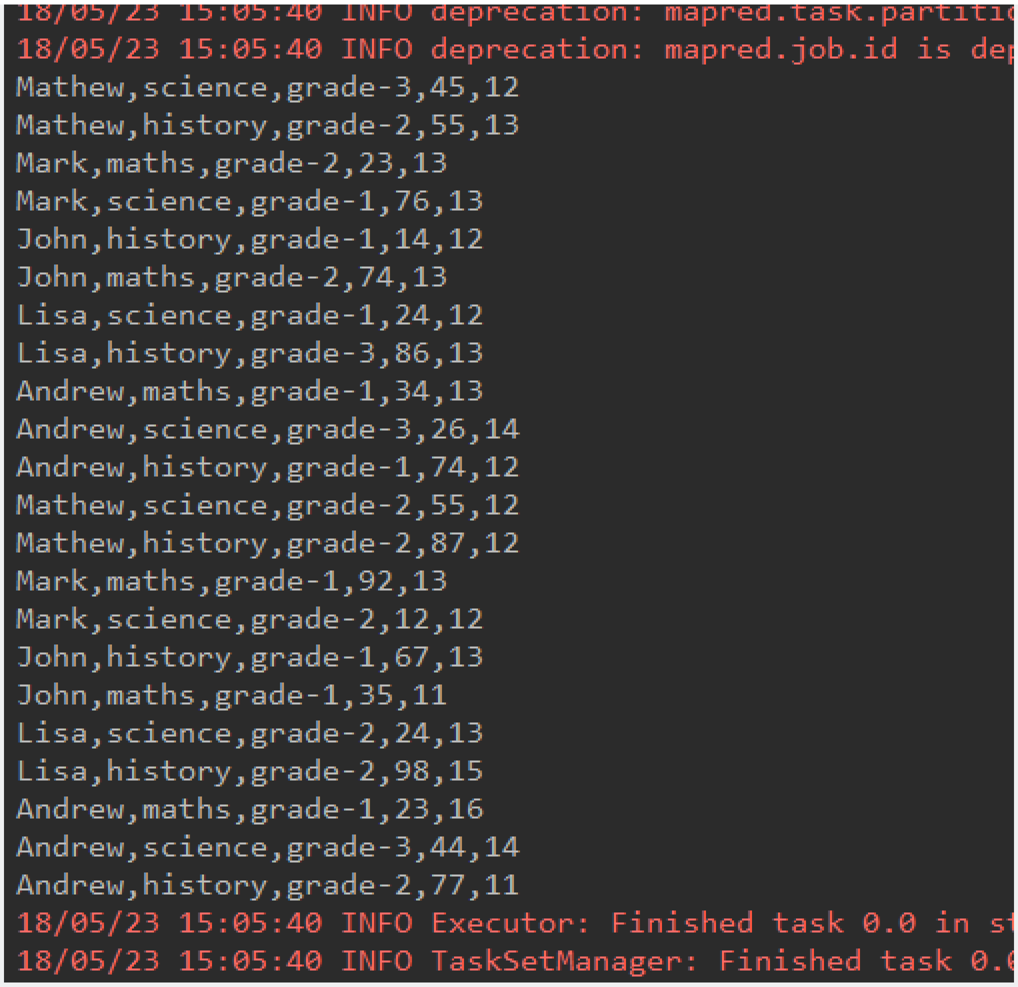
```
//create RDD by using textFile method of the sparkcontext object
val newdataset = spark.sparkContext.textFile( path = "E:\\Prachi IMP\\Hadoop\\Day 18 Spark\\Day 19 - Spark RDD\\19_Dataset.txt")
```

2. Written the println method inside foreach operation of RDD , which make sure that every Row from document is getting print
3. Also used RDD's count() operation to get the number of records belonging to a File



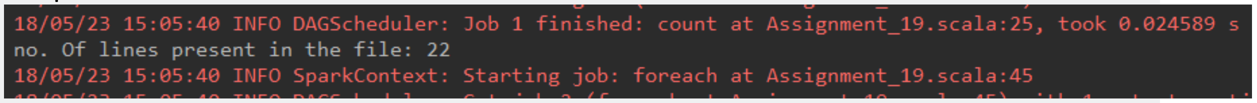
```
//Task 1.1 Number of lines / Rows in the file
//Printing each line of data from document
newdataset.foreach(w=>println(w))
//Printing number of Rows in Dataset
val noOfLines = newdataset.count()
//printing total count of rows in document
println("no. Of lines present in the file: " + noOfLines)
```

Output of Printing all Rows from document



```
18/05/23 15:05:40 INFO deprecation: mapred.task.partitic
18/05/23 15:05:40 INFO deprecation: mapred.job.id is dep
Mathew,science,grade-3,45,12
Mathew,history,grade-2,55,13
Mark,maths,grade-2,23,13
Mark,science,grade-1,76,13
John,history,grade-1,14,12
John,maths,grade-2,74,13
Lisa,science,grade-1,24,12
Lisa,history,grade-3,86,13
Andrew,maths,grade-1,34,13
Andrew,science,grade-3,26,14
Andrew,history,grade-1,74,12
Mathew,science,grade-2,55,12
Mathew,history,grade-2,87,12
Mark,maths,grade-1,92,13
Mark,science,grade-2,12,12
John,history,grade-1,67,13
John,maths,grade-1,35,11
Lisa,science,grade-2,24,13
Lisa,history,grade-2,98,15
Andrew,maths,grade-1,23,16
Andrew,science,grade-3,44,14
Andrew,history,grade-2,77,11
18/05/23 15:05:40 INFO Executor: Finished task 0.0 in st
18/05/23 15:05:40 INFO TaskSetManager: Finished task 0.0
```

Output of no. of Rows in document



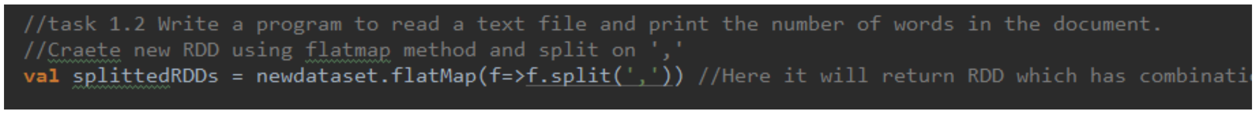
```
18/05/23 15:05:40 INFO DAGScheduler: Job 1 finished: count at Assignment_19.scala:25, took 0.024589 s
no. Of lines present in the file: 22
18/05/23 15:05:40 INFO SparkContext: Starting job: foreach at Assignment_19.scala:45
```

Task 1.2
Write a program to read a text file and print the number of words in the document.

Solution Approach -

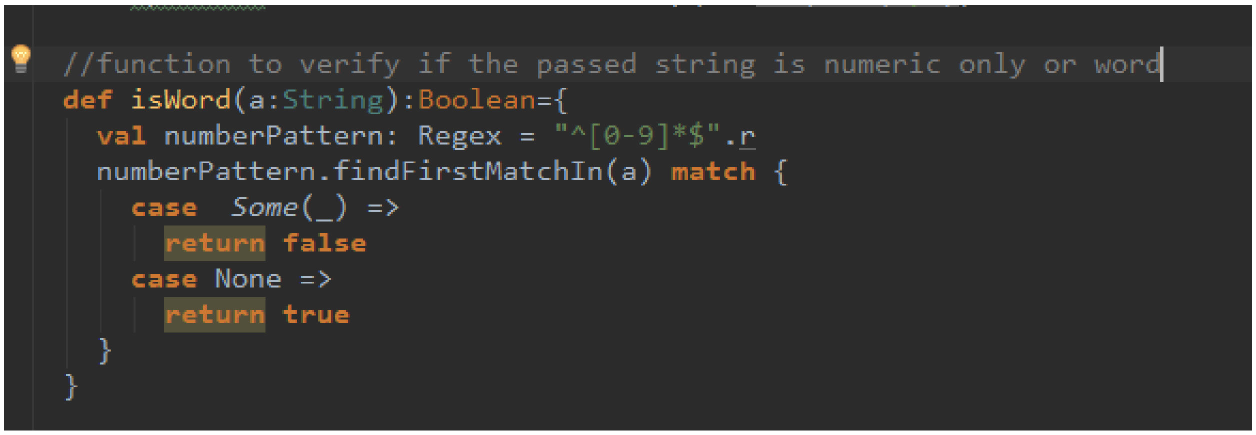
1. **We have made some assumption here for ‘words’ definition**. First we will consider **words is any value having alpha numeric contents and not only numeric values**. So in the given data set only below columns will be considered as words
   Column 1 – Student Name
   Column 2- Subject
   Column 3 – Grade
2. We have RDD created named as newdataset which is created in Task 1.1)

3. Created new RDD from *newdataset* using flatmap and split on ‘,’
    a. Flatmap will separate each row of the existing RDD in tow different elements by using regular expression given in split method which is used inside flatmap method.



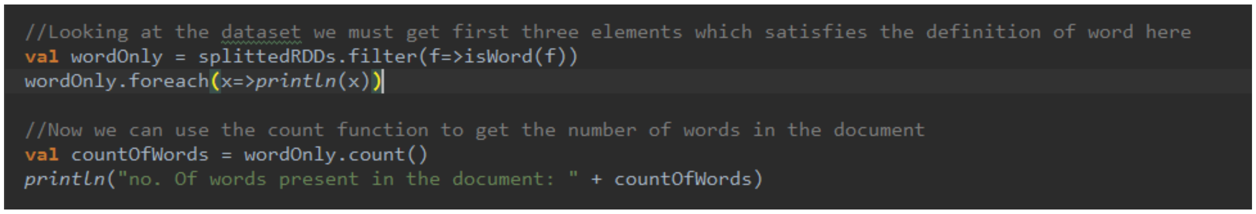
```
//task 1.2 Write a program to read a text file and print the number of words in the document.
//Craete new RDD using flatmap method and split on ','
val splittedRDDs = newdataset.flatMap(f=>f.split(',')) //Here it will return RDD which has combinati
```

4. Wrote new function which will return Boolean value based on evaluation of string passed to it
    a. If Word = true
    b. If only numeric then false



```
//function to verify if the passed string is numeric only or word
def isWord(a:String):Boolean={
  val numberPattern: Regex = "^[0-9]*$".r
  numberPattern.findFirstMatchIn(a) match {
    case  Some(_) =>
      return false
    case None =>
      return true
  }
}
```

5. Used filter operation on RDD to check if every element in RDD is word or numeric by calling above function



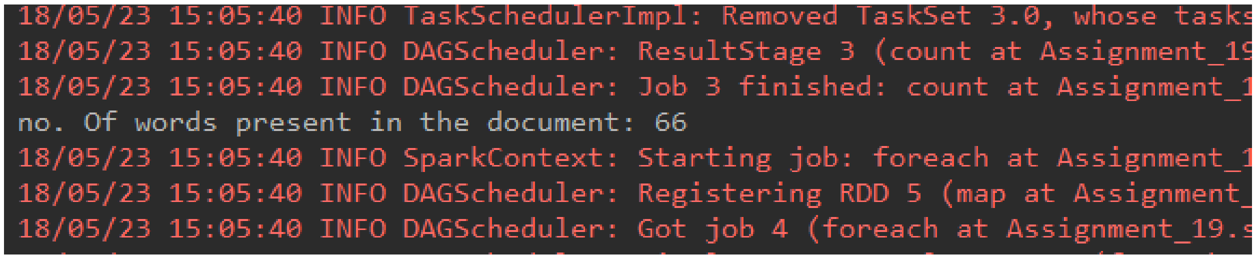
```
//Looking at the dataset we must get first three elements which satisfies the definition of word here
val wordOnly = splittedRDDs.filter(f=>isWord(f))
wordOnly.foreach(x=>println(x))

//Now we can use the count function to get the number of words in the document
val countOfWords = wordOnly.count()
println("no. Of words present in the document: " + countOfWords)
```

Entire Code for this Task

```
//task 1.2 Write a program to read a text file and print the number of
words in the document.
//Craete new RDD using flatmap method and split on ','
val splittedRDDs = newdataset.flatMap(f=>f.split(',')) //Here it will
return RDD which has combination of words (String + Aplhabates and
Only numbers will be excluded)

//function to verify if the passed string is numeric only or word
def isWord(a:String):Boolean={
  val numberPattern: Regex = "^[0-9]*$".r
  numberPattern.findFirstMatchIn(a) match {
    case  Some(_) =>
      return false
    case None =>
      return true
  }
}

//Looking at the dataset we must get first three elements which
satisfies the definition of word here
val wordOnly = splittedRDDs.filter(f=>isWord(f))
wordOnly.foreach(x=>println(x))

//Now we can use the count function to get the number of words in the
document
val countOfWords = wordOnly.count()
println("no. Of words present in the document: " + countOfWords)
```

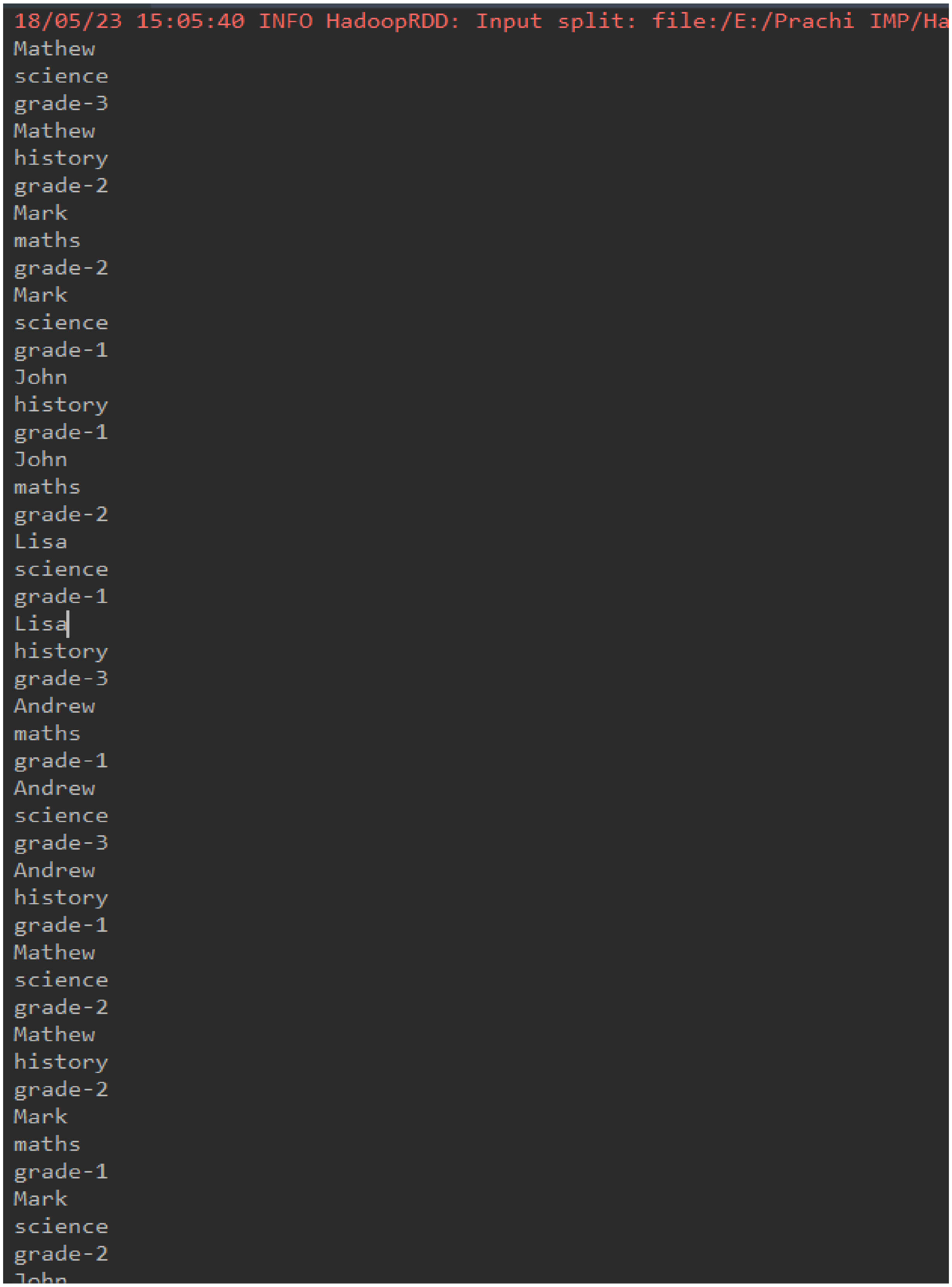
Output

Displying coun of only words



```
18/05/23 15:05:40 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks
18/05/23 15:05:40 INFO DAGScheduler: ResultStage 3 (count at Assignment_19
18/05/23 15:05:40 INFO DAGScheduler: Job 3 finished: count at Assignment_1
no. Of words present in the document: 66
18/05/23 15:05:40 INFO SparkContext: Starting job: foreach at Assignment_1
18/05/23 15:05:40 INFO DAGScheduler: Registering RDD 5 (map at Assignment_
18/05/23 15:05:40 INFO DAGScheduler: Got job 4 (foreach at Assignment_19.s
```

Displaying only words



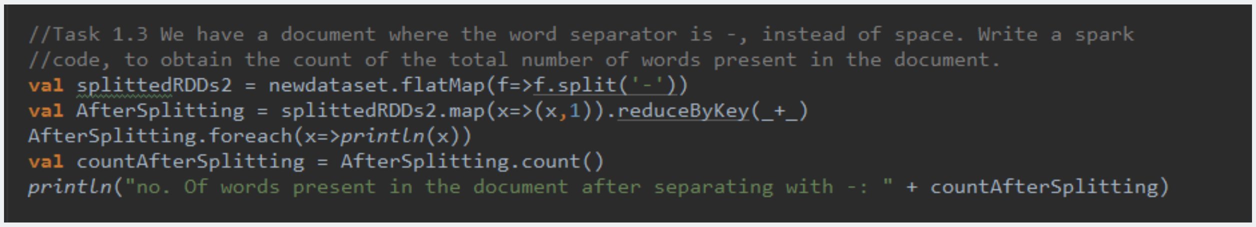
```
18/05/23 15:05:40 INFO HadoopRDD: Input split: file:/E:/Prachi IMP/Ha
Mathew
science
grade-3
Mathew
history
grade-2
Mark
maths
grade-2
Mark
science
grade-1
John
history
grade-1
John
maths
grade-2
Lisa
science
grade-1
Lisa
history
grade-3
Andrew
maths
grade-1
Andrew
science
grade-3
Andrew
history
grade-1
Mathew
science
grade-2
Mathew
history
grade-2
Mark
maths
grade-1
Mark
science
grade-2
John
```

Task 1.3
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 Approach –

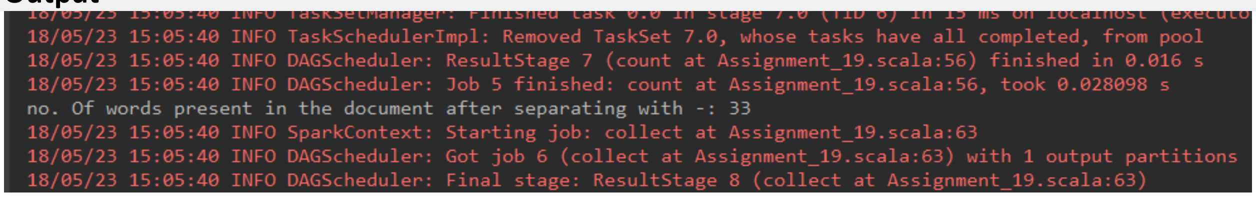
1. Split the newdataset RDD using character '-'
2. Here words will be having both only numeric and alpha numeric values
3. Used flatmap operation to get RDD splitted by '-'
4. Then mapped the newly created RDD with counter 1

Code



```
//Task 1.3 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.
val splittedRDDs2 = newdataset.flatMap(f=>f.split('-'))
val AfterSplitting = splittedRDDs2.map(x=>(x,1)).reduceByKey(_+_)
AfterSplitting.foreach(x=>println(x))
val countAfterSplitting = AfterSplitting.count()
println("no. Of words present in the document after separating with -: " + countAfterSplitting)
```

Output



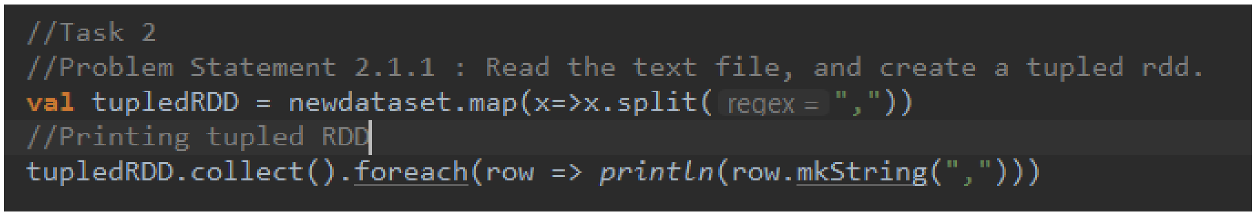
```
18/05/23 15:05:40 INFO TaskSetManager: Finished task 0.0 in stage 7.0 (TID 6) in 15 ms on localhost (executo
18/05/23 15:05:40 INFO TaskSchedulerImpl: Removed TaskSet 7.0, whose tasks have all completed, from pool
18/05/23 15:05:40 INFO DAGScheduler: ResultStage 7 (count at Assignment_19.scala:56) finished in 0.016 s
18/05/23 15:05:40 INFO DAGScheduler: Job 5 finished: count at Assignment_19.scala:56, took 0.028098 s
no. Of words present in the document after separating with -: 33
18/05/23 15:05:40 INFO SparkContext: Starting job: collect at Assignment_19.scala:63
18/05/23 15:05:40 INFO DAGScheduler: Got job 6 (collect at Assignment_19.scala:63) with 1 output partitions
18/05/23 15:05:40 INFO DAGScheduler: Final stage: ResultStage 8 (collect at Assignment_19.scala:63)
```

Task 2
Problem Statement 1:
2.1.1
Read the text file, and create a tupled rdd.

Solution Approach –

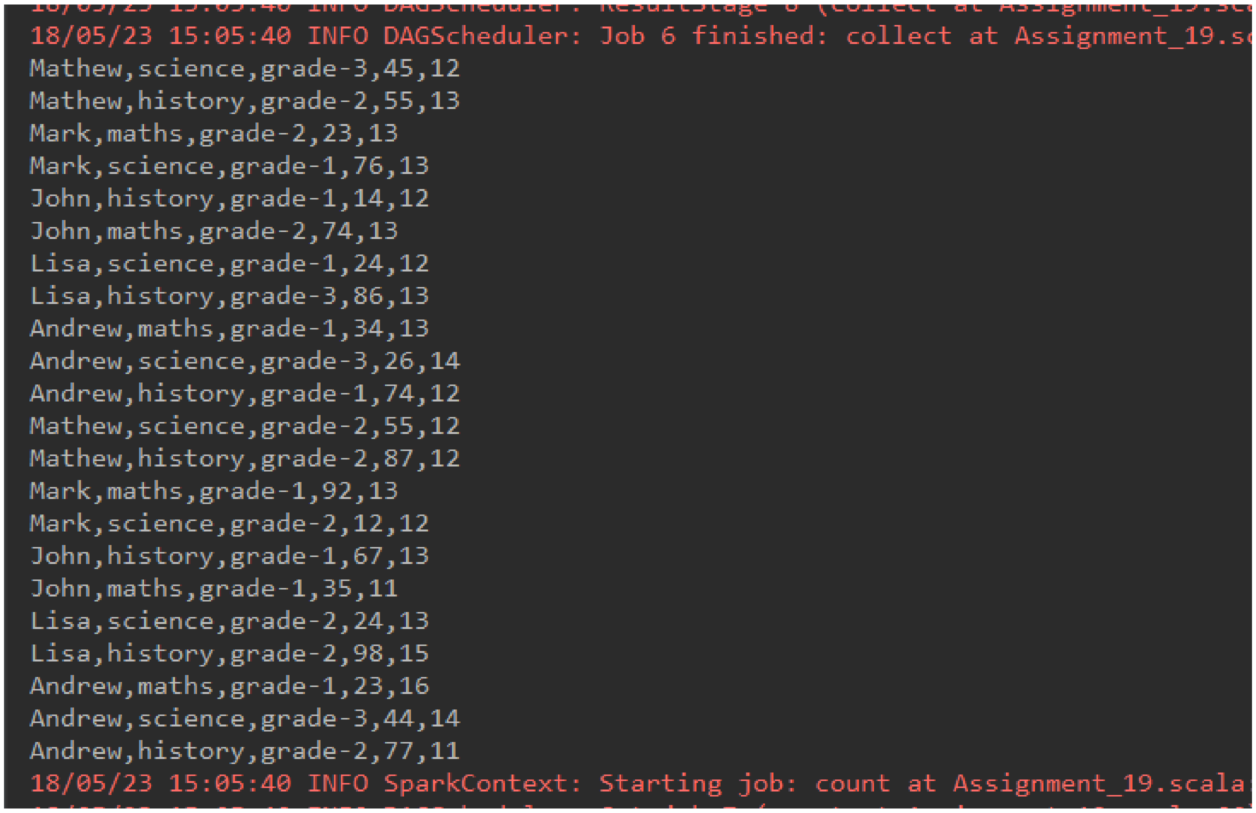
1. We have already read file in RDD named as *newdataset*
2. Create new tupled RDD by using map operation where every element of row is separated by ','

Code



```
//Task 2
//Problem Statement 2.1.1 : Read the text file, and create a tupled rdd.
val tupledRDD = newdataset.map(x=>x.split( regex = ","))
//Printing tupled RDD
tupledRDD.collect().foreach(row => println(row.mkString(",")))
```

Output



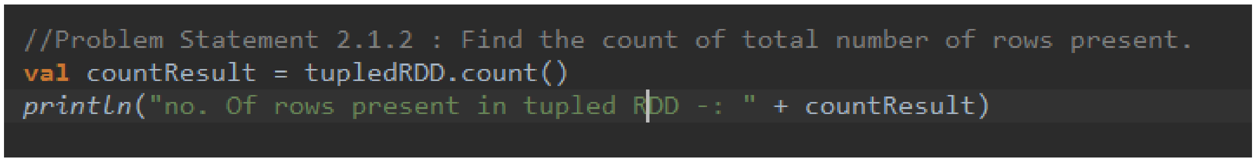
```
18/05/23 15:05:40 INFO DAGScheduler: Job 6 finished: collect at Assignment_19.sc
Mathew,science,grade-3,45,12
Mathew,history,grade-2,55,13
Mark,maths,grade-2,23,13
Mark,science,grade-1,76,13
John,history,grade-1,14,12
John,maths,grade-2,74,13
Lisa,science,grade-1,24,12
Lisa,history,grade-3,86,13
Andrew,maths,grade-1,34,13
Andrew,science,grade-3,26,14
Andrew,history,grade-1,74,12
Mathew,science,grade-2,55,12
Mathew,history,grade-2,87,12
Mark,maths,grade-1,92,13
Mark,science,grade-2,12,12
John,history,grade-1,67,13
John,maths,grade-1,35,11
Lisa,science,grade-2,24,13
Lisa,history,grade-2,98,15
Andrew,maths,grade-1,23,16
Andrew,science,grade-3,44,14
Andrew,history,grade-2,77,11
18/05/23 15:05:40 INFO SparkContext: Starting job: count at Assignment_19.scala
```

Task 2
Problem Statement 1:
2.1.2
Find the count of total number of rows present.

Solution Approach

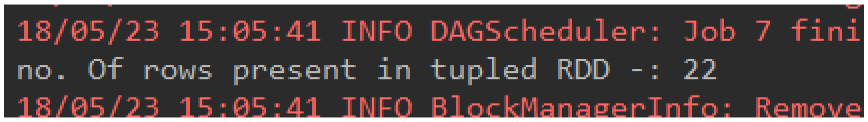
1. Used the count operation to get total number of rows present in tupled RDD.

Code



```scala
//Problem Statement 2.1.2 : Find the count of total number of rows present.
val countResult = tupledRDD.count()
println("no. Of rows present in tupled RDD -: " + countResult)
```

Output



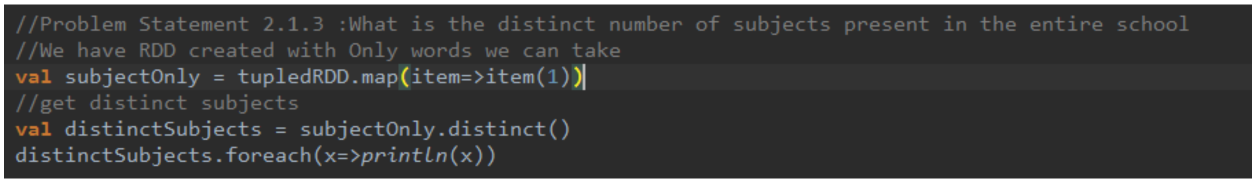
```
18/05/23 15:05:41 INFO DAGScheduler: Job 7 fini
no. Of rows present in tupled RDD -: 22
18/05/23 15:05:41 INFO BlockManagerInfo: Remove
```

Task 2
Problem Statement 1:
2.1.3
What is the distinct number of subjects present in the entire school

Solution Approach -

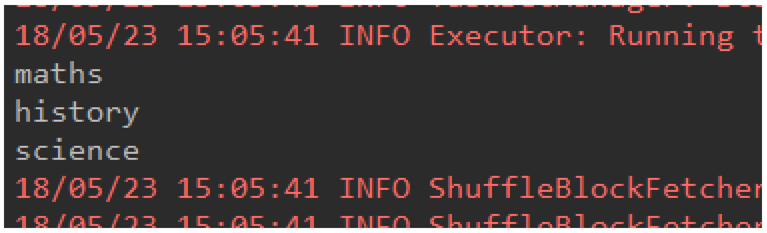
1. From the tupled RDD extracted only subjects column by using the map method
2. After getting only subjects used distinct method to get distinct subjects in school

Code



```
//Problem Statement 2.1.3 :What is the distinct number of subjects present in the entire school
//We have RDD created with Only words we can take
val subjectOnly = tupledRDD.map(item=>item(1))
//get distinct subjects
val distinctSubjects = subjectOnly.distinct()
distinctSubjects.foreach(x=>println(x))
```

Output



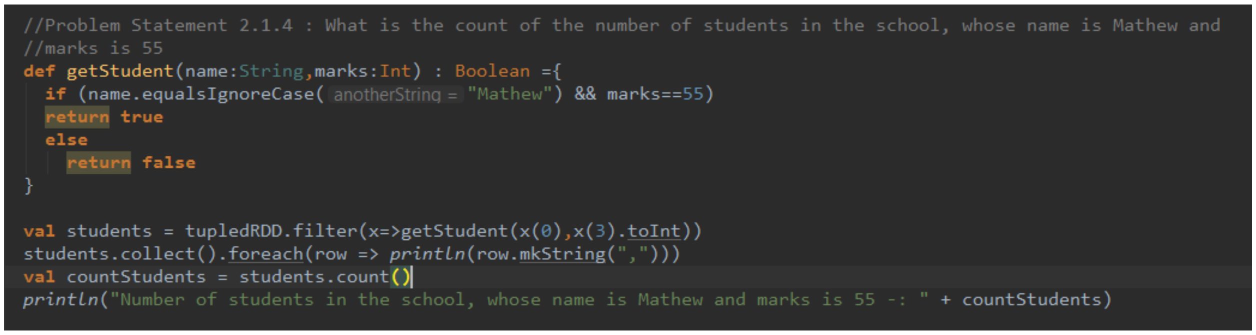
```
18/05/23 15:05:41 INFO Executor: Running t
maths
history
science
18/05/23 15:05:41 INFO ShuffleBlockFetcher
18/05/23 15:05:41 INFO ShuffleBlockFetcher
```

Task 2
Problem Statement 1:
2.1.3
What is the count of the number of students in the school, whose name is Mathew and marks is 55

Solution Approach -

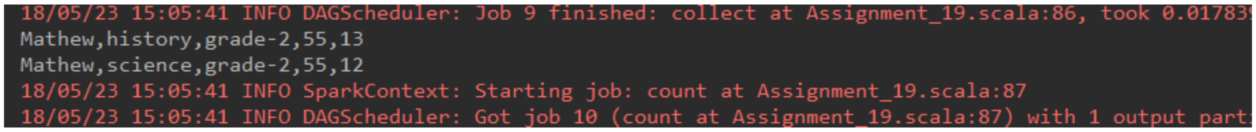
1. From the tupled RDD extracted only student name and marks column by using the map method
2. After getting new RDD used filter operation where condition is written to get student whose name = 'Mathew' and marks are 55. Instead of writing these conditions in single line created the function and called the same in filter operation.

Code



```
//Problem Statement 2.1.4 : What is the count of the number of students in the school, whose name is Mathew and
//marks is 55
def getStudent(name:String,marks:Int) : Boolean ={
  if (name.equalsIgnoreCase( anotherString = "Mathew") && marks==55)
  return true
  else
    return false
}

val students = tupledRDD.filter(x=>getStudent(x(0),x(3).toInt))
students.collect().foreach(row => println(row.mkString(",")))
val countStudents = students.count()
println("Number of students in the school, whose name is Mathew and marks is 55 -: " + countStudents)
```

Output



```
18/05/23 15:05:41 INFO DAGScheduler: Job 9 finished: collect at Assignment_19.scala:86, took 0.01783
Mathew,history,grade-2,55,13
Mathew,science,grade-2,55,12
18/05/23 15:05:41 INFO SparkContext: Starting job: count at Assignment_19.scala:87
18/05/23 15:05:41 INFO DAGScheduler: Got job 10 (count at Assignment_19.scala:87) with 1 output part
```

Task 2

Problem Statement 2:

2.2.1

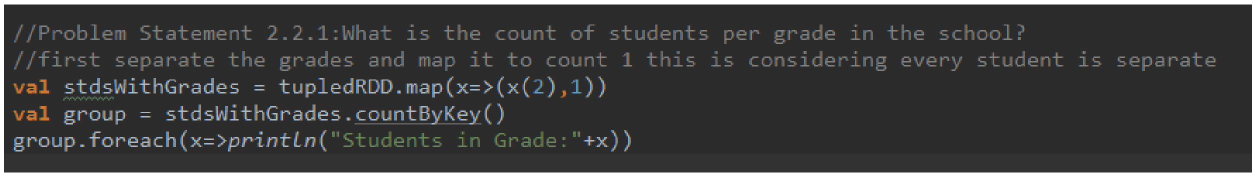
What is the count of students per grade in the school?

Solution Approach 1 –

1. Create new RDD with key as Grade and value as 1
2. Get distinct from above created RDD
3. And get the count of students per grade by using the CountByKey operation

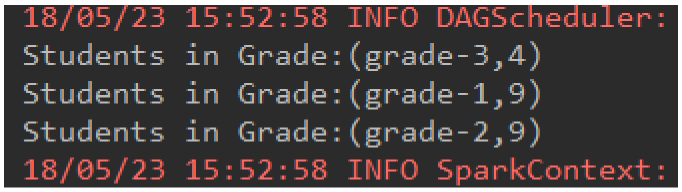
Note: In this case we have considered that every student is distinct in every grade.

Code



```
//Problem Statement 2.2.1:What is the count of students per grade in the school?
//first separate the grades and map it to count 1 this is considering every student is separate
val stdsWithGrades = tupledRDD.map(x=>(x(2),1))
val group = stdsWithGrades.countByKey()
group.foreach(x=>println("Students in Grade:"+x))
```

Output



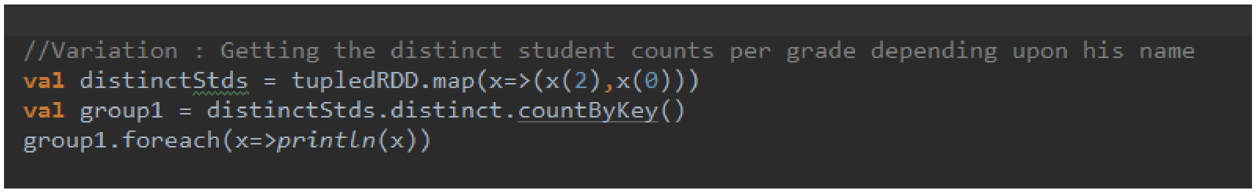
```
18/05/23 15:52:58 INFO DAGScheduler:
Students in Grade:(grade-3,4)
Students in Grade:(grade-1,9)
Students in Grade:(grade-2,9)
18/05/23 15:52:58 INFO SparkContext:
```

Solution Approach 2 –

Note: In this case we have to get the distinct students from per grade and then calculate the students.

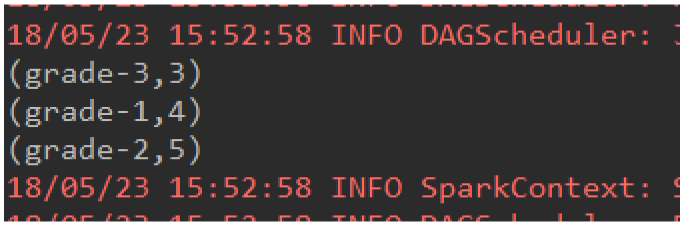
1. Create new RDD using map method where Key is Grade and Student Name
2. Get distinct of the above RDD
3. Then countbykey and get students belonging to every Grade

Code



```
//Variation : Getting the distinct student counts per grade depending upon his name
val distinctStds = tupledRDD.map(x=>(x(2),x(0)))
val group1 = distinctStds.distinct.countByKey()
group1.foreach(x=>println(x))
```

Output

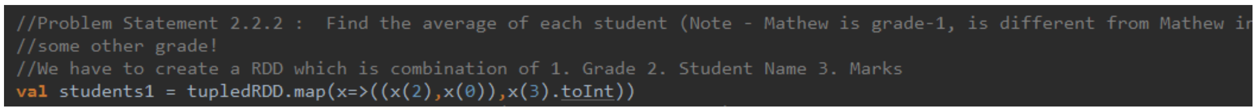


```
18/05/23 15:52:58 INFO DAGScheduler: J
(grade-3,3)
(grade-1,4)
(grade-2,5)
18/05/23 15:52:58 INFO SparkContext: S
```

Task 2
Problem Statement 2:
2.2.2
Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)

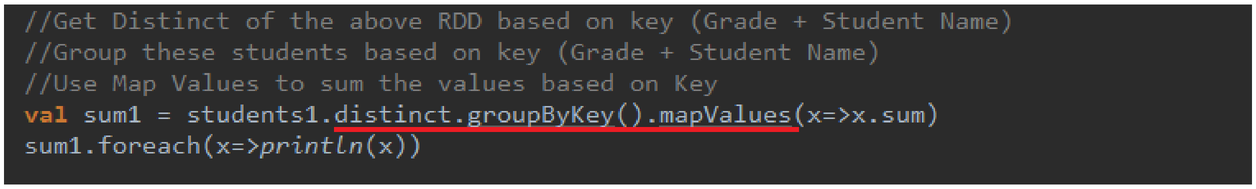
Solution Approach –

1. Create a new RDD from tupled RDD using map method where key is Grade + Student Name and Value is marks of students.



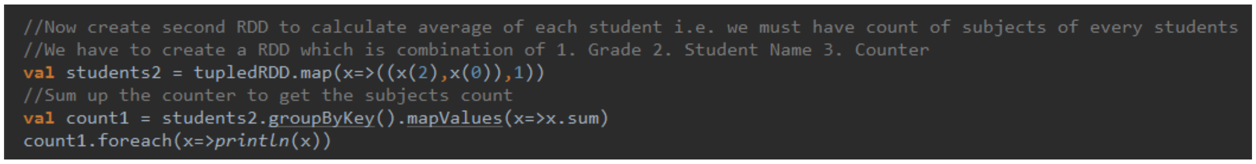
```
//Problem Statement 2.2.2 :  Find the average of each student (Note - Mathew is grade-1, is different from Mathew ir
//some other grade!
//We have to create a RDD which is combination of 1. Grade 2. Student Name 3. Marks
val students1 = tupledRDD.map(x=>((x(2),x(0)),x(3).toInt))
```

2. Get the distinct of above RDD on key ➔ Grade + Student_name
3. Group on above distinct RDD on key ➔ Grade+ Student_name
4. Use mapValues operations to get sum of marks per grade per student



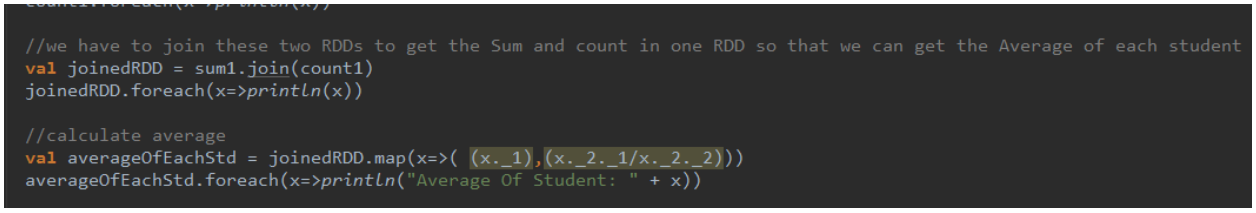
```
//Get Distinct of the above RDD based on key (Grade + Student Name)
//Group these students based on key (Grade + Student Name)
//Use Map Values to sum the values based on Key
val sum1 = students1.distinct.groupByKey().mapValues(x=>x.sum)
sum1.foreach(x=>println(x))
```

5. After this Create a RDD which has key as Grade + Student_name and value as counter (1)
6. Group above RDD on Key and get the sum of counter by using mapvalues method



```
//Now create second RDD to calculate average of each student i.e. we must have count of subjects of every students
//We have to create a RDD which is combination of 1. Grade 2. Student Name 3. Counter
val students2 = tupledRDD.map(x=>((x(2),x(0)),1))
//Sum up the counter to get the subjects count
val count1 = students2.groupByKey().mapValues(x=>x.sum)
count1.foreach(x=>println(x))
```

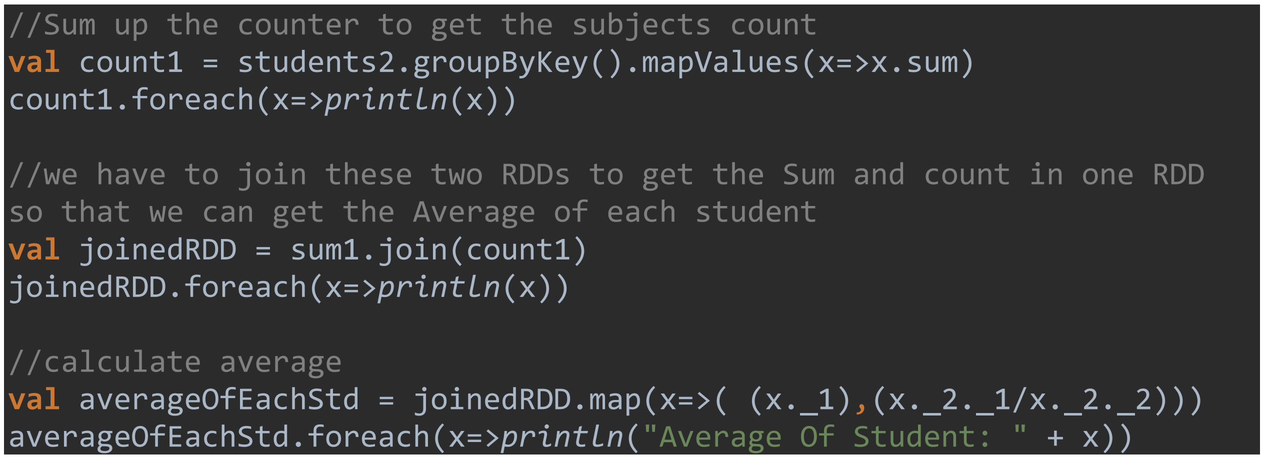
7. Once these two RDD created 1. One for Sum of marks 2. For total Subjects for a Student, Join these two RDD and create new RDD
8. On above created RDD , used map method to get the Key as Grade + Student Name and value as ➔ Sum Of marks / no. Of Subjects scored by student



```
count1.foreach(x=>println(x))

//we have to join these two RDDs to get the Sum and count in one RDD so that we can get the Average of each student
val joinedRDD = sum1.join(count1)
joinedRDD.foreach(x=>println(x))

//calculate average
val averageOfEachStd = joinedRDD.map(x=>( (x._1),(x._2._1/x._2._2)))
averageOfEachStd.foreach(x=>println("Average Of Student: " + x))
```

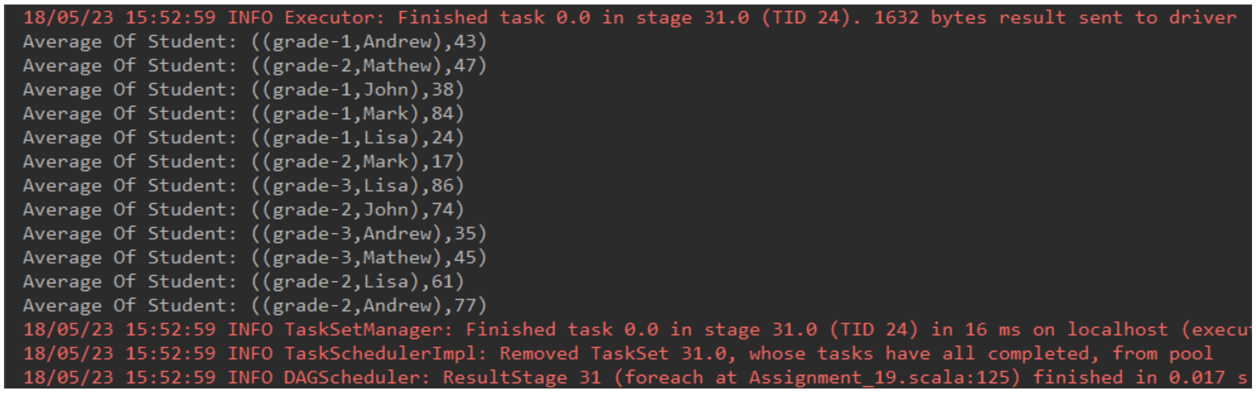
Entrie Code for above Problem Statement

```
//Problem Statement 2.2.2 :  Find the average of each student (Note -
Mathew is grade-1, is different from Mathew in
//some other grade!
//We have to create a RDD which is combination of 1. Grade 2. Student
Name 3. Marks
val students1 = tupledRDD.map(x=>((x(2),x(0)),x(3).toInt))
//Get Distinct of the above RDD based on key (Grade + Student Name)
//Group these students based on key (Grade + Student Name)
//Use Map Values to sum the values based on Key
val sum1 = students1.distinct.groupByKey().mapValues(x=>x.sum)
sum1.foreach(x=>println(x))

//Now create second RDD to calculate average of each student i.e. we
must have count of subjects of every students
//We have to create a RDD which is combination of 1. Grade 2. Student
Name 3. Counter
val students2 = tupledRDD.map(x=>((x(2),x(0)),1))
```



```scala
//Sum up the counter to get the subjects count
val count1 = students2.groupByKey().mapValues(x=>x.sum)
count1.foreach(x=>println(x))

//we have to join these two RDDs to get the Sum and count in one RDD
so that we can get the Average of each student
val joinedRDD = sum1.join(count1)
joinedRDD.foreach(x=>println(x))

//calculate average
val averageOfEachStd = joinedRDD.map(x=>( (x._1),(x._2._1/x._2._2)))
averageOfEachStd.foreach(x=>println("Average Of Student: " + x))
```

Output

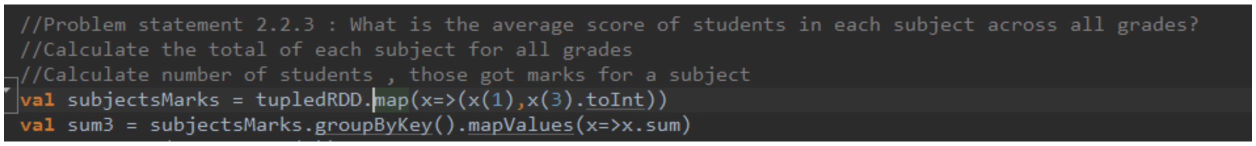


```
18/05/23 15:52:59 INFO Executor: Finished task 0.0 in stage 31.0 (TID 24). 1632 bytes result sent to driver
Average Of Student: ((grade-1,Andrew),43)
Average Of Student: ((grade-2,Mathew),47)
Average Of Student: ((grade-1,John),38)
Average Of Student: ((grade-1,Mark),84)
Average Of Student: ((grade-1,Lisa),24)
Average Of Student: ((grade-2,Mark),17)
Average Of Student: ((grade-3,Lisa),86)
Average Of Student: ((grade-2,John),74)
Average Of Student: ((grade-3,Andrew),35)
Average Of Student: ((grade-3,Mathew),45)
Average Of Student: ((grade-2,Lisa),61)
Average Of Student: ((grade-2,Andrew),77)
18/05/23 15:52:59 INFO TaskSetManager: Finished task 0.0 in stage 31.0 (TID 24) in 16 ms on localhost (execu
18/05/23 15:52:59 INFO TaskSchedulerImpl: Removed TaskSet 31.0, whose tasks have all completed, from pool
18/05/23 15:52:59 INFO DAGScheduler: ResultStage 31 (foreach at Assignment_19.scala:125) finished in 0.017 s
```

Task 2
Problem Statement 2:
2.2.3
What is the average score of students in each subject across all grades?

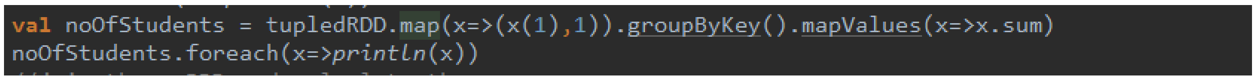
Solution Approach –

1. Create a new RDD from tupled RDD using map method where key is subject and Value is marks of students.
2. Get the sum of all the marks per subject



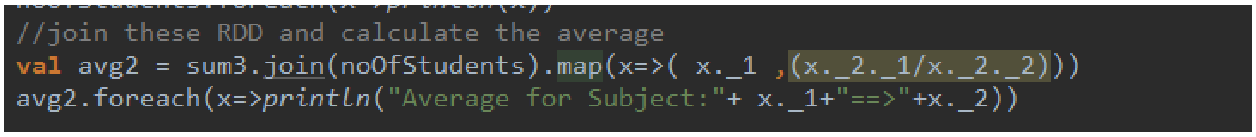
```scala
//Problem statement 2.2.3 : What is the average score of students in each subject across all grades?
//Calculate the total of each subject for all grades
//Calculate number of students , those got marks for a subject
val subjectsMarks = tupledRDD.map(x=>(x(1),x(3).toInt))
val sum3 = subjectsMarks.groupByKey().mapValues(x=>x.sum)
```

3. Create a new RDD from tupled RDD to get the no. of students scored in every subject , so key is subject and value is counter (1) and get the sum of counter



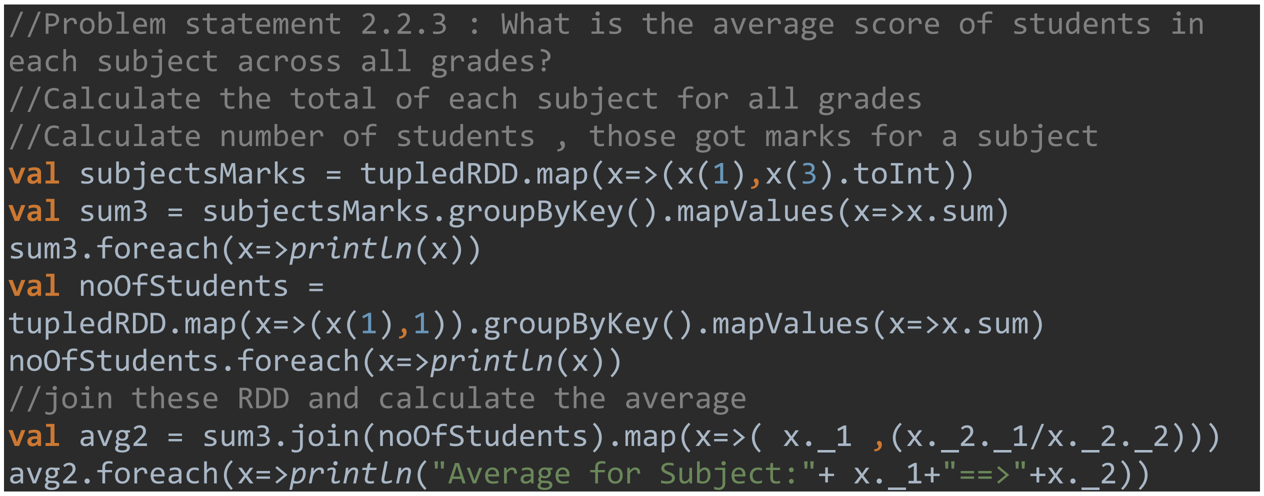
```
val noOfStudents = tupledRDD.map(x=>(x(1),1)).groupByKey().mapValues(x=>x.sum)
noOfStudents.foreach(x=>println(x))
```

4. Now join above tow created RDD to get new RDD where we will map it on key Subject and value is average = total marks / count of students scoered in that subject



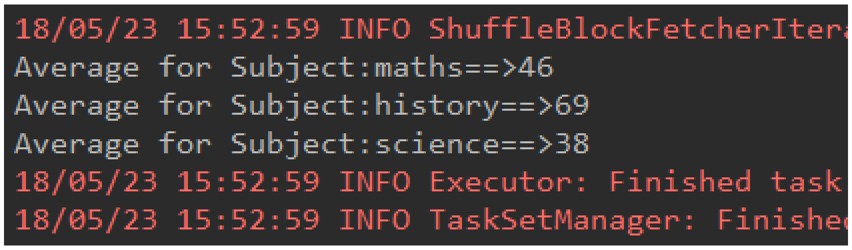
```
//join these RDD and calculate the average
val avg2 = sum3.join(noOfStudents).map(x=>( x._1 ,(x._2._1/x._2._2)))
avg2.foreach(x=>println("Average for Subject:"+ x._1+"==>"+x._2))
```

Entire Code



```
//Problem statement 2.2.3 : What is the average score of students in
each subject across all grades?
//Calculate the total of each subject for all grades
//Calculate number of students , those got marks for a subject
val subjectsMarks = tupledRDD.map(x=>(x(1),x(3).toInt))
val sum3 = subjectsMarks.groupByKey().mapValues(x=>x.sum)
sum3.foreach(x=>println(x))
val noOfStudents =
tupledRDD.map(x=>(x(1),1)).groupByKey().mapValues(x=>x.sum)
noOfStudents.foreach(x=>println(x))
//join these RDD and calculate the average
val avg2 = sum3.join(noOfStudents).map(x=>( x._1 ,(x._2._1/x._2._2)))
avg2.foreach(x=>println("Average for Subject:"+ x._1+"==>"+x._2))
```

Output

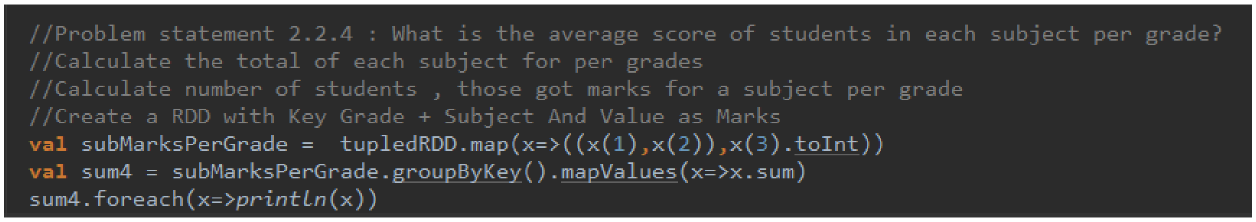


```
18/05/23 15:52:59 INFO ShuffleBlockFetcherItera
Average for Subject:maths==>46
Average for Subject:history==>69
Average for Subject:science==>38
18/05/23 15:52:59 INFO Executor: Finished task
18/05/23 15:52:59 INFO TaskSetManager: Finishe
```

Task 2
Problem Statement 2:
2.2.4
What is the average score of students in each subject per grade?

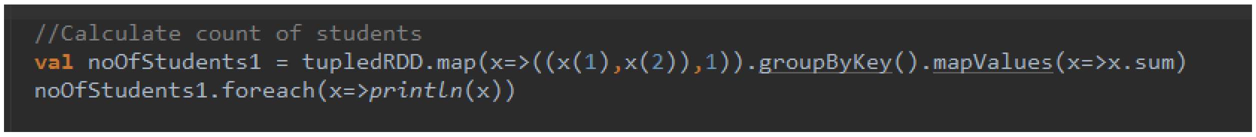
Solution Approach –

1. Create a new RDD from tupled RDD using map method where key is subject + grade and Value is marks of students.
2. Get distinct above RDD and sum on the values by using GroupByKey and mapValues method



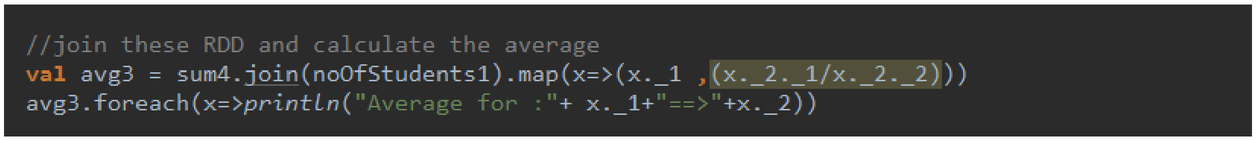
```
//Problem statement 2.2.4 : What is the average score of students in each subject per grade?
//Calculate the total of each subject for per grades
//Calculate number of students , those got marks for a subject per grade
//Create a RDD with Key Grade + Subject And Value as Marks
val subMarksPerGrade =  tupledRDD.map(x=>((x(1),x(2)),x(3).toInt))
val sum4 = subMarksPerGrade.groupByKey().mapValues(x=>x.sum)
sum4.foreach(x=>println(x))
```

3. Create a RDD for getting total count of stduents appeared for a subject per grade where Key is ➔ Grade + Subject and value is ➔ counter i.e. 1



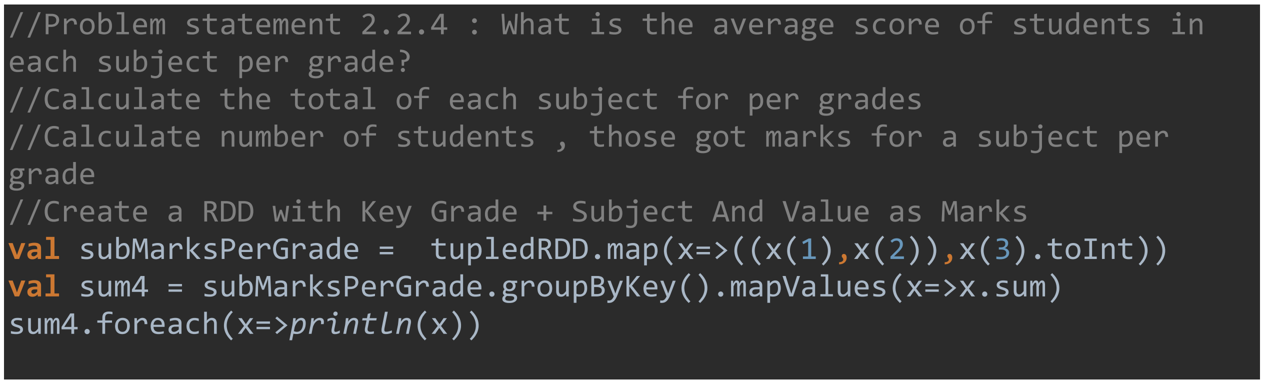
```
//Calculate count of students
val noOfStudents1 = tupledRDD.map(x=>((x(1),x(2)),1)).groupByKey().mapValues(x=>x.sum)
noOfStudents1.foreach(x=>println(x))
```

4. Join above to RDD to get combined RDD from which we can get required RDD where key is ➔ Grade + Subject and value is ➔ Average = Sum / total



```
//join these RDD and calculate the average
val avg3 = sum4.join(noOfStudents1).map(x=>(x._1 ,(x._2._1/x._2._2)))
avg3.foreach(x=>println("Average for :"+ x._1+"==>"+x._2))
```

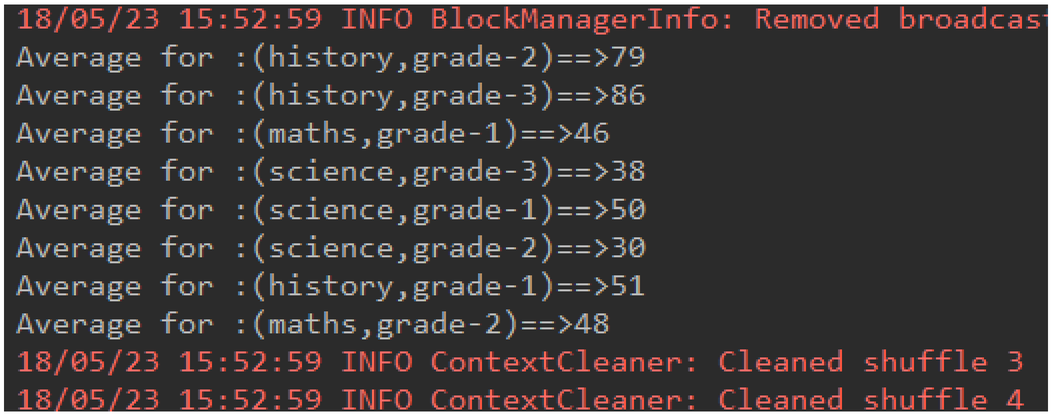
Entire Code



```
//Problem statement 2.2.4 : What is the average score of students in
each subject per grade?
//Calculate the total of each subject for per grades
//Calculate number of students , those got marks for a subject per
grade
//Create a RDD with Key Grade + Subject And Value as Marks
val subMarksPerGrade =  tupledRDD.map(x=>((x(1),x(2)),x(3).toInt))
val sum4 = subMarksPerGrade.groupByKey().mapValues(x=>x.sum)
sum4.foreach(x=>println(x))
```

```
//Calculate count of students
val noOfStudents1 =
tupledRDD.map(x=>((x(1),x(2)),1)).groupByKey().mapValues(x=>x.sum)
noOfStudents1.foreach(x=>println(x))

//join these RDD and calculate the average
val avg3 = sum4.join(noOfStudents1).map(x=>(x._1 ,(x._2._1/x._2._2)))
avg3.foreach(x=>println("Average for :"+ x._1+"==>"+x._2))
```

Output



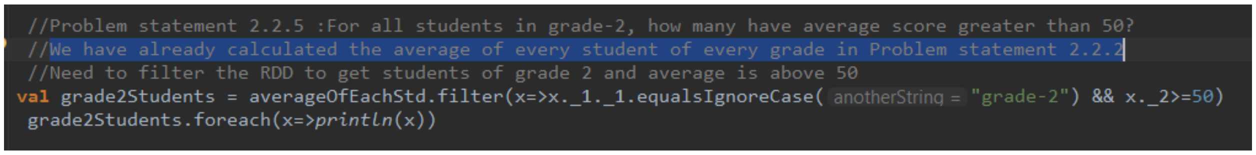
```
18/05/23 15:52:59 INFO BlockManagerInfo: Removed broadcas
Average for :(history,grade-2)==>79
Average for :(history,grade-3)==>86
Average for :(maths,grade-1)==>46
Average for :(science,grade-3)==>38
Average for :(science,grade-1)==>50
Average for :(science,grade-2)==>30
Average for :(history,grade-1)==>51
Average for :(maths,grade-2)==>48
18/05/23 15:52:59 INFO ContextCleaner: Cleaned shuffle 3
18/05/23 15:52:59 INFO ContextCleaner: Cleaned shuffle 4
```

Task 2
Problem Statement 2:
2.2.5
For all students in grade-2, how many have average score greater than 50?

Solution Approach –

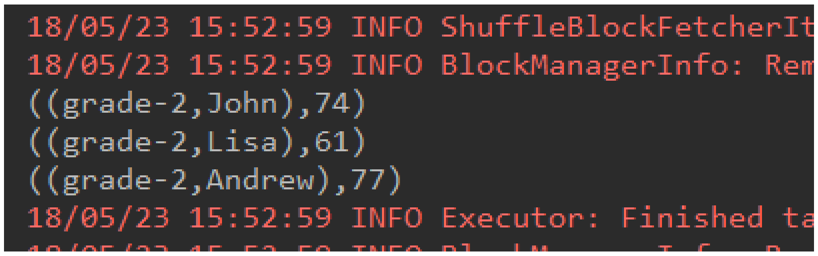
1. We have already calculated the average of every student of every grade in Problem statement 2.2.2
2. Filtered the RDD to get students belonging to grade-2 and average score greater than 50

Code



```
 //Problem statement 2.2.5 :For all students in grade-2, how many have average score greater than 50?
 //We have already calculated the average of every student of every grade in Problem statement 2.2.2
 //Need to filter the RDD to get students of grade 2 and average is above 50
val grade2Students = averageOfEachStd.filter(x=>x._1._1.equalsIgnoreCase( anotherString = "grade-2") && x._2>=50)
 grade2Students.foreach(x=>println(x))
```

Output

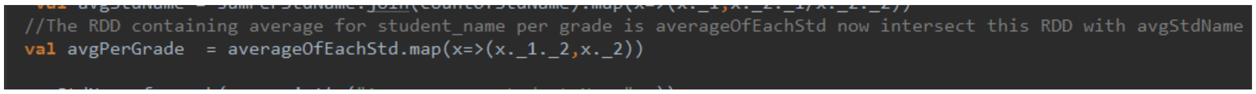


```
18/05/23 15:52:59 INFO ShuffleBlockFetcherIt
18/05/23 15:52:59 INFO BlockManagerInfo: Rem
((grade-2,John),74)
((grade-2,Lisa),61)
((grade-2,Andrew),77)
18/05/23 15:52:59 INFO Executor: Finished ta
```

Task 3
Problem Statement 1:
3.1.1
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

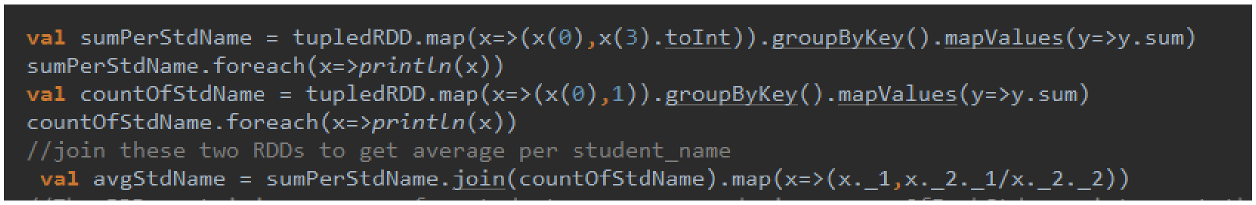
Solution Approach –

1. We have already calculated the average of every student of every grade in Problem statement 2.2.2
    a. Create a new RDD from above one using map method where key ➔ Student_Name and value is Average



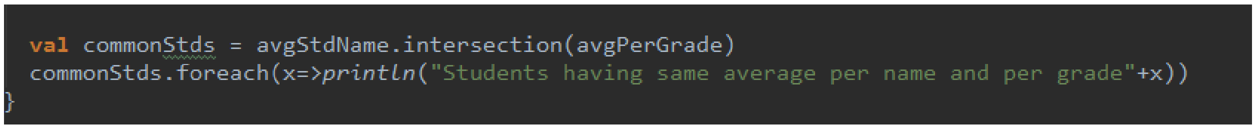
```
//The RDD containing average for student_name per grade is averageOfEachStd now intersect this RDD with avgStdName
val avgPerGrade  = averageOfEachStd.map(x=>(x._1._2,x._2))
```

2. We have to create RDD which has average of students across all grades
    a. Create new RDD from tupled RDD where key ➔ Student_Name and value ➔ Marks
    b. Get sum of values using mapValues method.
    c. Create another RDD where key ➔ Student_Name and value ➔ Counter (1)
    d. Get sum of Value (counter) using mapValues method
    e. Join above to RDDs get a RDD where key ➔ Student_Name and Average = total / Count



```
val sumPerStdName = tupledRDD.map(x=>(x(0),x(3).toInt)).groupByKey().mapValues(y=>y.sum)
sumPerStdName.foreach(x=>println(x))
val countOfStdName = tupledRDD.map(x=>(x(0),1)).groupByKey().mapValues(y=>y.sum)
countOfStdName.foreach(x=>println(x))
//join these two RDDs to get average per student_name
 val avgStdName = sumPerStdName.join(countOfStdName).map(x=>(x._1,x._2._1/x._2._2))
```

3. Intersect above two RDDs (from 1.a and from 2.e)



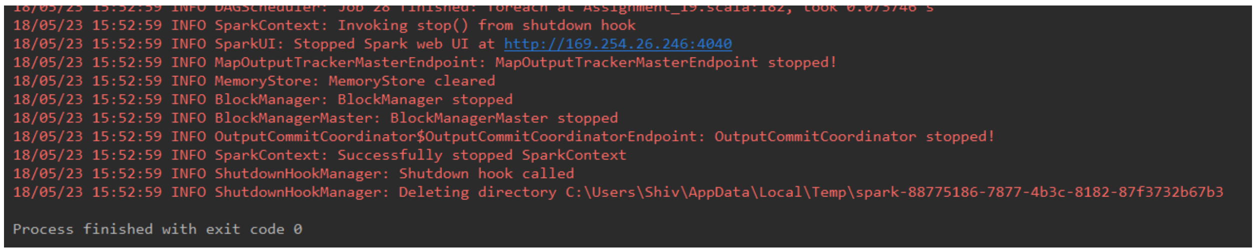
```
  val commonStds = avgStdName.intersection(avgPerGrade)
  commonStds.foreach(x=>println("Students having same average per name and per grade"+x))
}
```

Entire Code

```
//Problem Statement 2.3.1: 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
//We have already calculated average per student_name per grade in
problem statement 2.2.2
//Calculate average per student_name
//Create above tow RDD with key Student_Name & Value as average
//Intersect these two RDDs and get the student_name having same
average in both the above mentioned cases

val sumPerStdName =
tupledRDD.map(x=>(x(0),x(3).toInt)).groupByKey().mapValues(y=>y.sum)
sumPerStdName.foreach(x=>println(x))
val countOfStdName =
tupledRDD.map(x=>(x(0),1)).groupByKey().mapValues(y=>y.sum)
countOfStdName.foreach(x=>println(x))
//join these two RDDs to get average per student_name
 val avgStdName =
sumPerStdName.join(countOfStdName).map(x=>(x._1,x._2._1/x._2._2))
//The RDD containing average for student_name per grade is
averageOfEachStd now intersect this RDD with avgStdName
val avgPerGrade  = averageOfEachStd.map(x=>(x._1._2,x._2))

avgStdName.foreach( x=>println("Average per student_Name"+x))
avgPerGrade.foreach( x=>println("Average per student_Name & Grade"+x))

val commonStds = avgStdName.intersection(avgPerGrade)
commonStds.foreach(x=>println("Students having same average per name
and per grade"+x))
```

Output

It returns null value



```
18/05/23 15:52:59 INFO DAGScheduler: Job 28 finished: foreach at Assignment_19.scala:182, took 0.073746 s
18/05/23 15:52:59 INFO SparkContext: Invoking stop() from shutdown hook
18/05/23 15:52:59 INFO SparkUI: Stopped Spark web UI at http://169.254.26.246:4040
18/05/23 15:52:59 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/05/23 15:52:59 INFO MemoryStore: MemoryStore cleared
18/05/23 15:52:59 INFO BlockManager: BlockManager stopped
18/05/23 15:52:59 INFO BlockManagerMaster: BlockManagerMaster stopped
18/05/23 15:52:59 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/05/23 15:52:59 INFO SparkContext: Successfully stopped SparkContext
18/05/23 15:52:59 INFO ShutdownHookManager: Shutdown hook called
18/05/23 15:52:59 INFO ShutdownHookManager: Deleting directory C:\Users\Shiv\AppData\Local\Temp\spark-88775186-7877-4b3c-8182-87f3732b67b3

Process finished with exit code 0
```