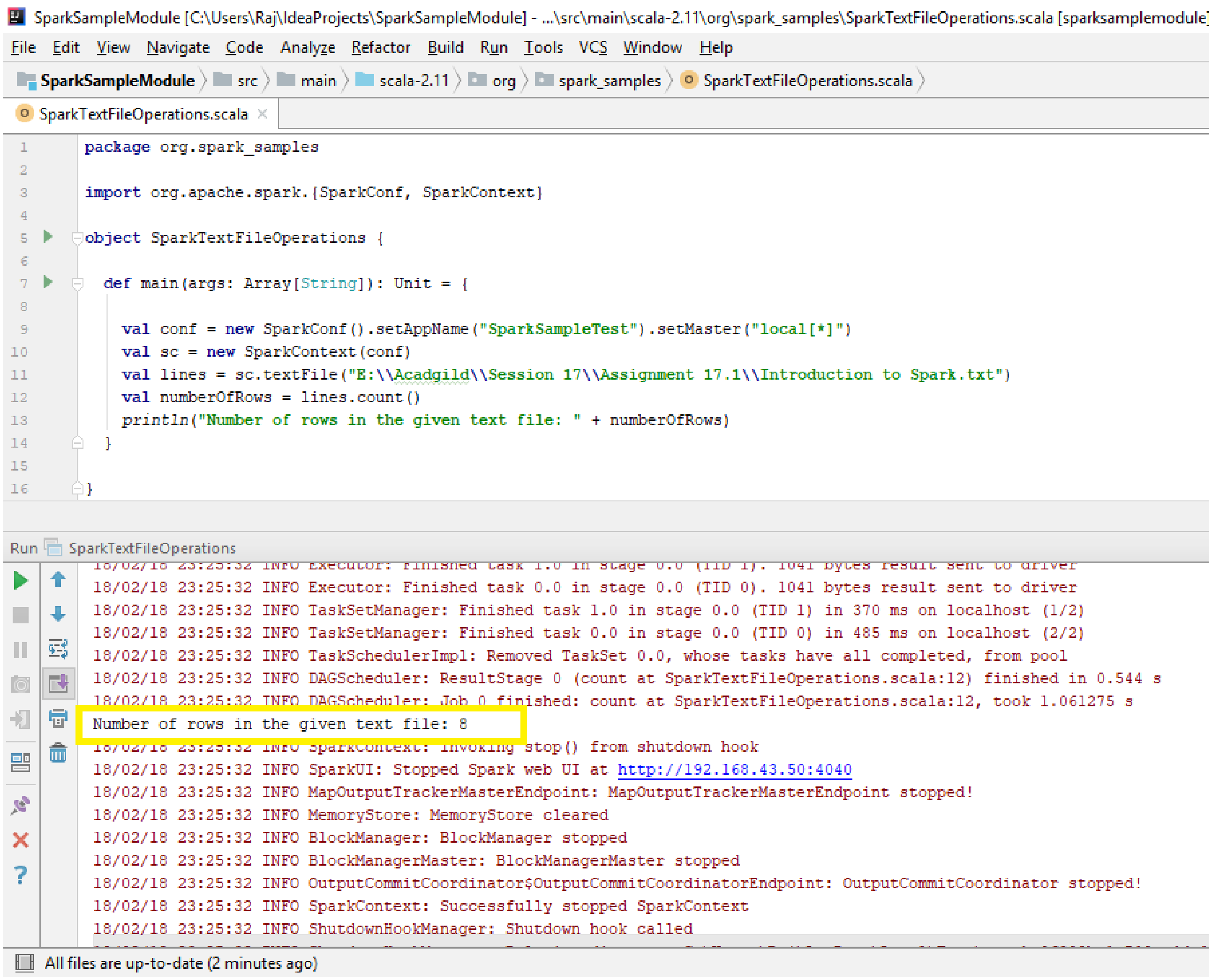
Task 1

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

```
object SparkTextFileOperations {
def main(args: Array[String]): Unit = {
val conf = new SparkConf().setAppName("SparkTest").setMaster("local[*]")
val sc = new SparkContext(conf)
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\Introduction to Spark.txt")
val numberOfRows = lines.count()
println("Number of rows in the given text file: " + numberOfRows)
}
}
```



SparkSampleModule [C:\Users\Raj\IdeaProjects\SparkSampleModule] - ...\src\main\scala-2.11\org\spark_samples\SparkTextFileOperations.scala [sparksamplemodule]
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SparkSampleModule src main scala-2.11 org spark_samples SparkTextFileOperations.scala
SparkTextFileOperations.scala
Run SparkTextFileOperations

```
package org.spark_samples

import org.apache.spark.{SparkConf, SparkContext}

object SparkTextFileOperations {

  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 17.1\\Introduction to Spark.txt")
    val numberOfRows = lines.count()
    println("Number of rows in the given text file: " + numberOfRows)
  }

}
```

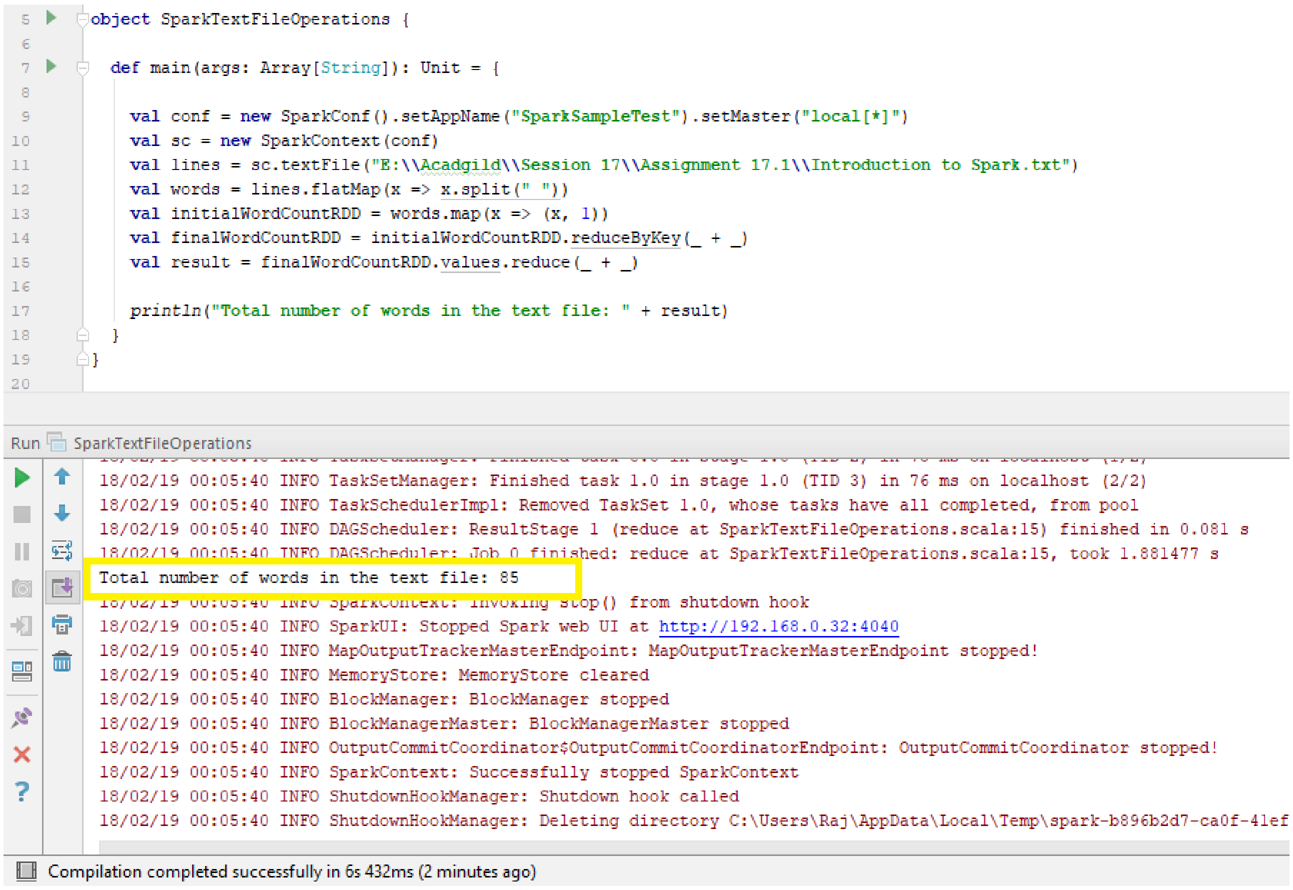
```
18/02/18 23:25:32 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 1041 bytes result sent to driver
18/02/18 23:25:32 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1041 bytes result sent to driver
18/02/18 23:25:32 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 370 ms on localhost (1/2)
18/02/18 23:25:32 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 485 ms on localhost (2/2)
18/02/18 23:25:32 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/02/18 23:25:32 INFO DAGScheduler: ResultStage 0 (count at SparkTextFileOperations.scala:12) finished in 0.544 s
18/02/18 23:25:32 INFO DAGScheduler: Job 0 finished: count at SparkTextFileOperations.scala:12, took 1.061275 s
Number of rows in the given text file: 8
18/02/18 23:25:32 INFO SparkContext: Invoking stop() from shutdown hook
18/02/18 23:25:32 INFO SparkUI: Stopped Spark web UI at http://192.168.43.50:4040
18/02/18 23:25:32 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/02/18 23:25:32 INFO MemoryStore: MemoryStore cleared
18/02/18 23:25:32 INFO BlockManager: BlockManager stopped
18/02/18 23:25:32 INFO BlockManagerMaster: BlockManagerMaster stopped
18/02/18 23:25:32 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/02/18 23:25:32 INFO SparkContext: Successfully stopped SparkContext
18/02/18 23:25:32 INFO ShutdownHookManager: Shutdown hook called
```

All files are up-to-date (2 minutes ago)

Output:
Number of rows in the given text file: 8

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

```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\Introduction to Spark.txt")
val words = lines.flatMap(x => x.split(" "))
val initialWordCountRDD = words.map(x => (x, 1))
val finalWordCountRDD = initialWordCountRDD.reduceByKey(_ + _)
val result = finalWordCountRDD.values.reduce(_ + _)
println("Total number of words in the text file: " + result)
```



```
object SparkTextFileOperations {

  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 17.1\\Introduction to Spark.txt")
    val words = lines.flatMap(x => x.split(" "))
    val initialWordCountRDD = words.map(x => (x, 1))
    val finalWordCountRDD = initialWordCountRDD.reduceByKey(_ + _)
    val result = finalWordCountRDD.values.reduce(_ + _)

    println("Total number of words in the text file: " + result)
  }
}
```

Run SparkTextFileOperations

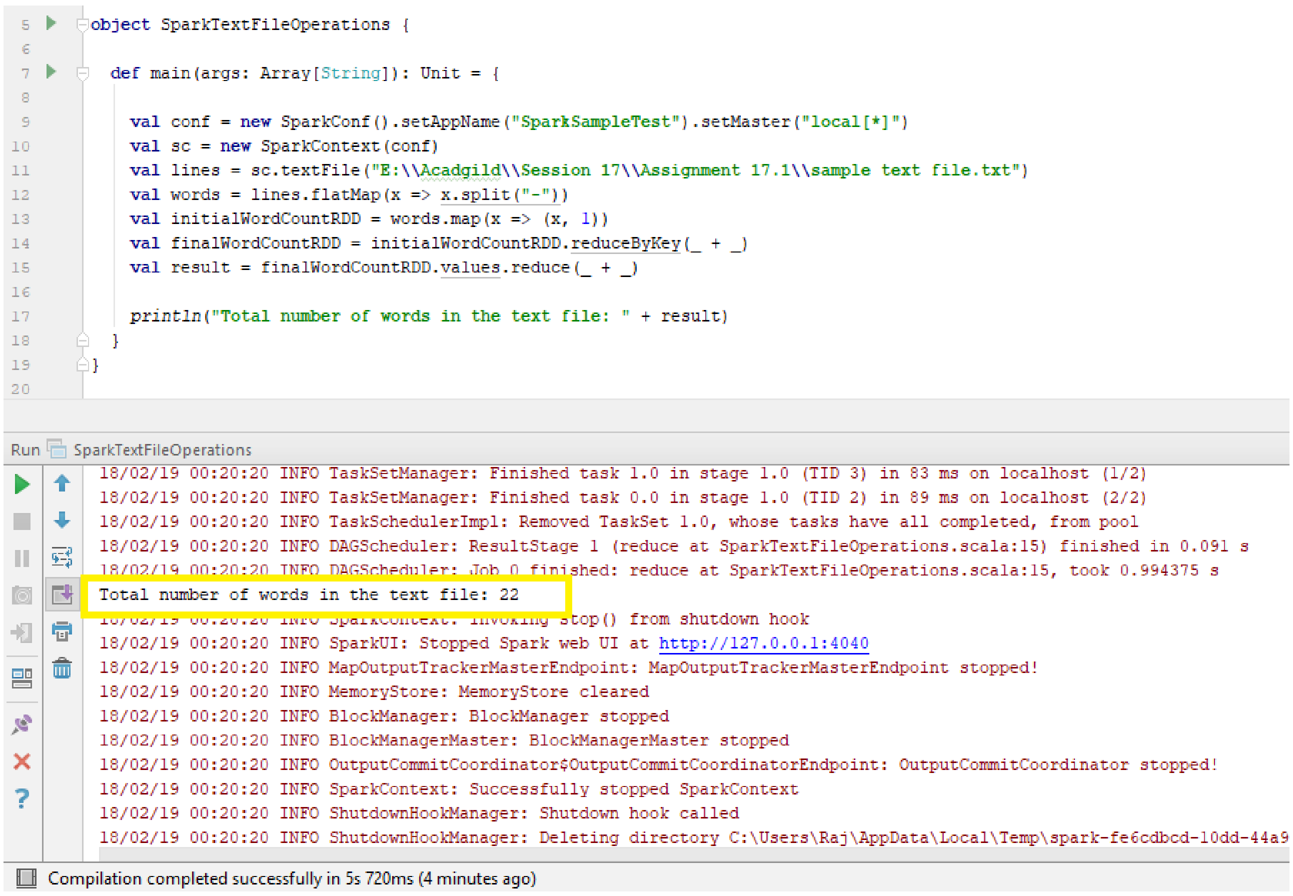
```
18/02/19 00:05:40 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 3) in 76 ms on localhost (2/2)
18/02/19 00:05:40 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
18/02/19 00:05:40 INFO DAGScheduler: ResultStage 1 (reduce at SparkTextFileOperations.scala:15) finished in 0.081 s
18/02/19 00:05:40 INFO DAGScheduler: Job 0 finished: reduce at SparkTextFileOperations.scala:15, took 1.881477 s
Total number of words in the text file: 85
18/02/19 00:05:40 INFO SparkContext: Invoking stop() from shutdown hook
18/02/19 00:05:40 INFO SparkUI: Stopped Spark web UI at http://192.168.0.32:4040
18/02/19 00:05:40 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/02/19 00:05:40 INFO MemoryStore: MemoryStore cleared
18/02/19 00:05:40 INFO BlockManager: BlockManager stopped
18/02/19 00:05:40 INFO BlockManagerMaster: BlockManagerMaster stopped
18/02/19 00:05:40 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/02/19 00:05:40 INFO SparkContext: Successfully stopped SparkContext
18/02/19 00:05:40 INFO ShutdownHookManager: Shutdown hook called
18/02/19 00:05:40 INFO ShutdownHookManager: Deleting directory C:\Users\Raj\AppData\Local\Temp\spark-b896b2d7-ca0f-41ef
```

Compilation completed successfully in 6s 432ms (2 minutes ago)

Output:
Total number of words in the text file: 85

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 lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\sample text file.txt")
val words = lines.flatMap(x => x.split("–"))
val initialWordCountRDD = words.map(x => (x, 1))
val finalWordCountRDD = initialWordCountRDD.reduceByKey(_ + _)
val result = finalWordCountRDD.values.reduce(_ + _)
println("Total number of words in the text file: " + result)
```



```
object SparkTextFileOperations {

  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 17.1\\sample text file.txt")
    val words = lines.flatMap(x => x.split("-"))
    val initialWordCountRDD = words.map(x => (x, 1))
    val finalWordCountRDD = initialWordCountRDD.reduceByKey(_ + _)
    val result = finalWordCountRDD.values.reduce(_ + _)

    println("Total number of words in the text file: " + result)
  }
}
```

```
18/02/19 00:20:20 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 3) in 83 ms on localhost (1/2)
18/02/19 00:20:20 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 89 ms on localhost (2/2)
18/02/19 00:20:20 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
18/02/19 00:20:20 INFO DAGScheduler: ResultStage 1 (reduce at SparkTextFileOperations.scala:15) finished in 0.091 s
18/02/19 00:20:20 INFO DAGScheduler: Job 0 finished: reduce at SparkTextFileOperations.scala:15, took 0.994375 s
Total number of words in the text file: 22
18/02/19 00:20:20 INFO SparkContext: Invoking stop() from shutdown hook
18/02/19 00:20:20 INFO SparkUI: Stopped Spark web UI at http://127.0.0.1:4040
18/02/19 00:20:20 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/02/19 00:20:20 INFO MemoryStore: MemoryStore cleared
18/02/19 00:20:20 INFO BlockManager: BlockManager stopped
18/02/19 00:20:20 INFO BlockManagerMaster: BlockManagerMaster stopped
18/02/19 00:20:20 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/02/19 00:20:20 INFO SparkContext: Successfully stopped SparkContext
18/02/19 00:20:20 INFO ShutdownHookManager: Shutdown hook called
18/02/19 00:20:20 INFO ShutdownHookManager: Deleting directory C:\Users\Raj\AppData\Local\Temp\spark-fe6cdbcd-10dd-44a9
```

Compilation completed successfully in 5s 720ms (4 minutes ago)

Output:
Total number of words in the text file: 22

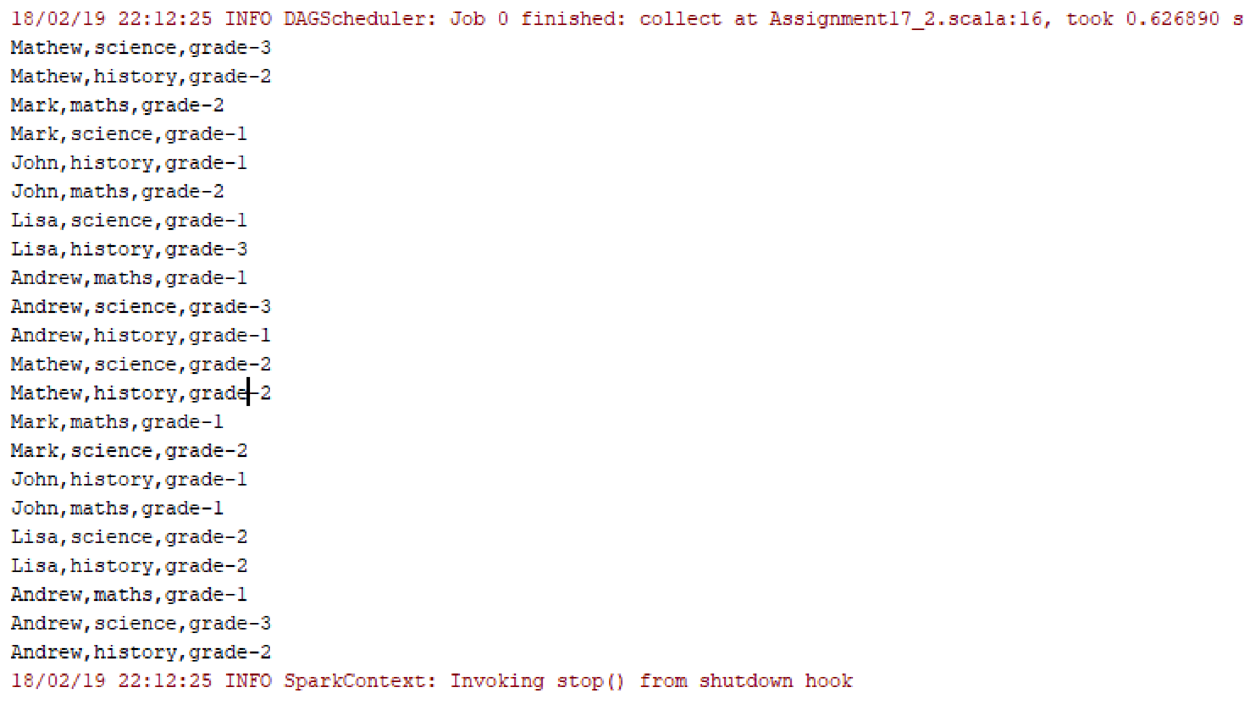
Task 2

Problem Statement 1:

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

```
object Assignment19 {
def main(args: Array[String]): Unit = {
val conf = new SparkConf().setAppName("SparkSampleTest").setMaster("local[*]")
val sc = new SparkContext(conf)
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_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:

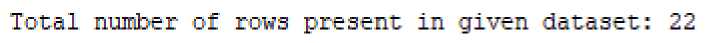


```
18/02/19 22:12:25 INFO DAGScheduler: Job 0 finished: collect at Assignment17_2.scala:16, took 0.626890 s
Mathew,science,grade-3
Mathew,history,grade-2
Mark,maths,grade-2
Mark,science,grade-1
John,history,grade-1
John,maths,grade-2
Lisa,science,grade-1
Lisa,history,grade-3
Andrew,maths,grade-1
Andrew,science,grade-3
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("D:\\Acadgild\\Session 19\\Assignment 19\\19_Dataset.txt")
println("Total number of rows present in given dataset: " + lines.count())
```

Output:

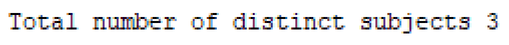


```
Total number of rows present in given dataset: 22
```

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

```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_Dataset.txt")
val distinct_subjects_rdd = lines.map(x => x.split(",") (1)).distinct()
println("Total number of distinct subjects " + distinct_subjects_rdd.count())
```

Output:

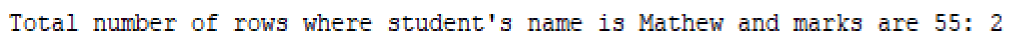


```
Total number of distinct subjects 3
```

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

```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_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:



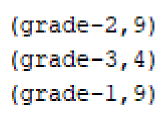
```
Total number of rows where student's name is Mathew and marks are 55: 2
```

Problem Statement 2:

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

```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_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:

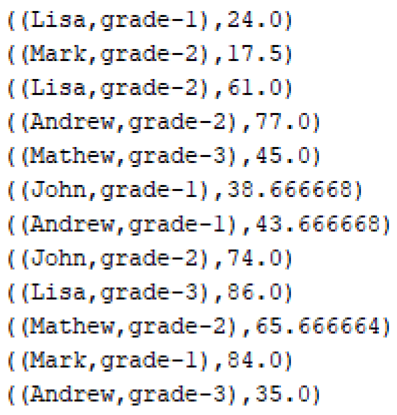


```
(grade-2,9)
(grade-3,4)
(grade-1,9)
```

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("D:\\Acadgild\\Session 19\\Assignment 19\\19_Dataset.txt")
val name_grades_and_marks_rdd = lines.map(x => ((x.split(",") (0), x.split(",") (2)), x.split(",")
(3).toFloat))
val grouped_rdd = name_grades_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_marks_rdd.foreach(println)
```

Output:

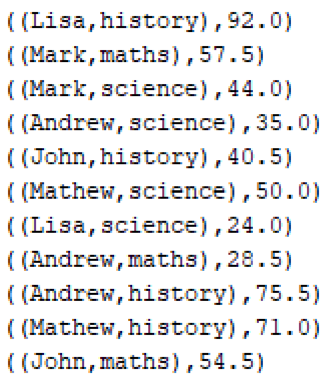


```
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Mathew,grade-3),45.0)
((John,grade-1),38.666668)
((Andrew,grade-1),43.666668)
((John,grade-2),74.0)
((Lisa,grade-3),86.0)
((Mathew,grade-2),65.666664)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
```

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

```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_Dataset.txt")
val name_subject_and_marks_rdd = lines.map(x => ((x.split(",") (0), x.split(",") (1)), x.split(",")
(3).toFloat))
val grouped_rdd = name_subject_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_marks_rdd.foreach(println)
```

Output:

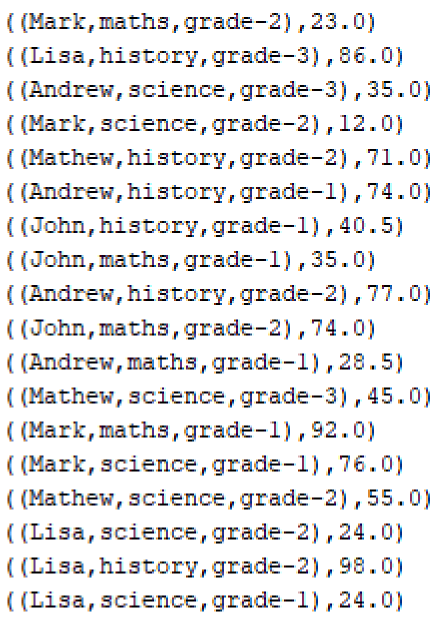


```
((Lisa,history),92.0)
((Mark,maths),57.5)
((Mark,science),44.0)
((Andrew,science),35.0)
((John,history),40.5)
((Mathew,science),50.0)
((Lisa,science),24.0)
((Andrew,maths),28.5)
((Andrew,history),75.5)
((Mathew,history),71.0)
((John,maths),54.5)
```

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

```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_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))
val grouped_rdd = name_subject_grade_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_marks_rdd.foreach(println)
```

Output:

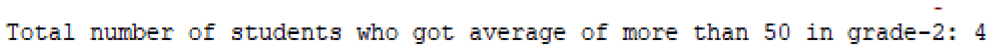


```
((Mark,maths,grade-2),23.0)
((Lisa,history,grade-3),86.0)
((Andrew,science,grade-3),35.0)
((Mark,science,grade-2),12.0)
((Mathew,history,grade-2),71.0)
((Andrew,history,grade-1),74.0)
((John,history,grade-1),40.5)
((John,maths,grade-1),35.0)
((Andrew,history,grade-2),77.0)
((John,maths,grade-2),74.0)
((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("D:\\Acadgild\\Session 19\\Assignment 19\\19_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:



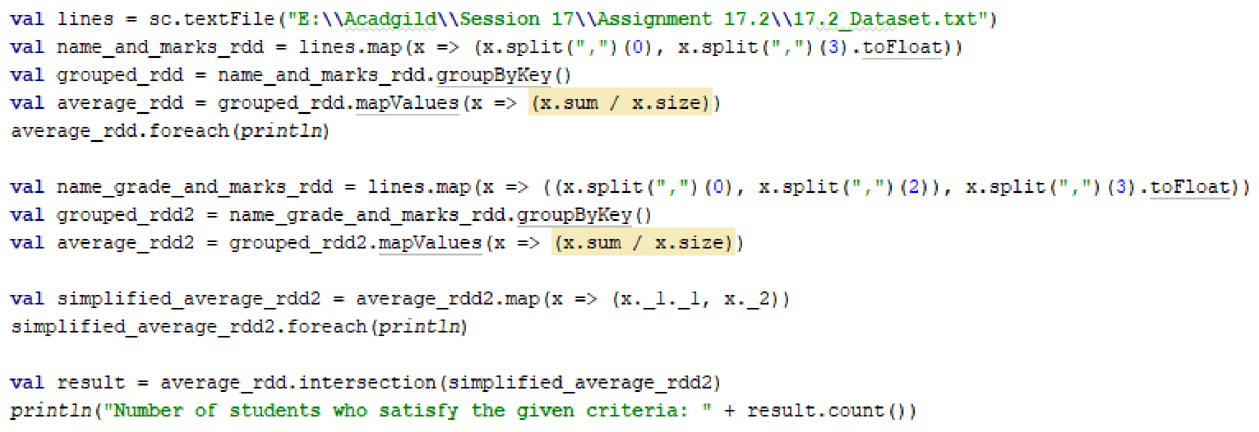
```
Total number of students who got average of more than 50 in grade-2: 4
```

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.

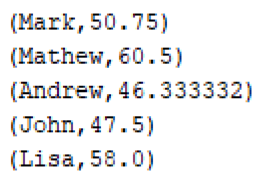
```
val lines = sc.textFile("D:\\Acadgild\\Session 19\\Assignment 19\\19_Dataset.txt")
val name_and_marks_rdd = lines.map(x => (x.split(",") (0), x.split(",") (3).toFloat))
val grouped_rdd = name_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_rdd.foreach(println)
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 res = average_rdd.intersection(simplified_average_rdd2)
println("Number of students who satisfy the given criteria: " + result.count())



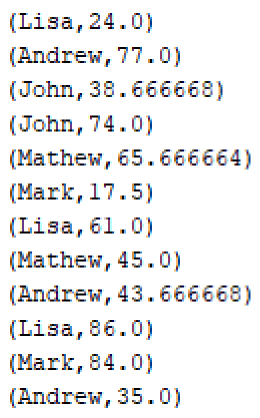
```
val lines = sc.textFile("E:\\Acadgild\\Session 17\\Assignment 17.2\\17.2_Dataset.txt")
val name_and_marks_rdd = lines.map(x => (x.split(",")(0), x.split(",")(3).toFloat))
val grouped_rdd = name_and_marks_rdd.groupByKey()
val average_rdd = grouped_rdd.mapValues(x => (x.sum / x.size))
average_rdd.foreach(println)

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:



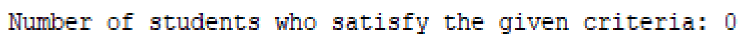
```
(Mark,50.75)
(Mathew,60.5)
(Andrew,46.333332)
(John,47.5)
(Lisa,58.0)
```

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.