Task 1:

Using count function to see the number of lines in the file. I have used 19_Dataset here as the file

```
scala> val myfile = sc.textFile("19_Dataset.txt")
myfile: org.apache.spark.rdd.RDD[String] = 19_Dataset.txt MapPartitionsRDD[1] at textFile at <con
sole>:24

scala> myfile.count
res0: Long = 23

scala>
```

acadgild@localhost:~ | acadgild

Here I have loaded a script with name wordcount.scala, As one can see it is showing the wordcount

```
scala> :load wordcount.scala
Loading wordcount.scala...
myfile: org.apache.spark.rdd.RDD[String] = test.txt MapPartitionsRDD[3] at textFile at <console>:
24
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[6] at reduceByKey at <console>:26
res2: Array[(String, Int)] = Array((are,2), (you,1), (how,2), (you.,1))

scala>
```

acadgild@localhost:~ | acadgild

To separate words by hyphen I am just changing the split condition with space by "-"

Applications Places System Tue Jun 12, 12:16 AM Acadgild

acadgild@localhost:~

File Edit View Search Terminal Tabs Help

acadgild@localhost:~ | acadgild@localhost:~

```
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/test.txt
18/06/12 00:16:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
how-are-you-how-are-you
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cat wordcount.scala
val myfile=sc.textFile("test.txt")

val counts = myfile.flatMap(_.split("-")).map(word => (word,1)).reduceByKey(_+_)
counts.collect
[acadgild@localhost ~]$
```

Task 2:

Output for problem1:

```
myfile: org.apache.spark.rdd.RDD[String] = test.txt MapPartitionsRDD[8] at textFile at <console>:
24
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at reduceByKey at <console>:26
res3: Array[(String, Int)] = Array((are,2), (you,2), (how,2))

scala> :load problem1.scala
Loading problem1.scala...
myfile: org.apache.spark.rdd.RDD[String] = 19_Dataset.txt MapPartitionsRDD[13] at textFile at <co
nsole>:24
No of rows = 22
d: org.apache.spark.rdd.RDD[(String, String, String, String, String)] = MapPartitionsRDD[15] at m
ap at <console>:26
array of tuples
res6: Array[(String, String, String, String, String)] = Array((Mathew,science,grade-3,45,12), (Ma
thew,history,grade-2,55,13), (Mark,maths,grade-2,23,13), (Mark,science,grade-1,76,13), (John,hist
ory,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,scien
ce,grade-3,44,14), (Andrew,history,grade-2,77,11))
b: Long = 3
no of unique subjects = 3
c: Long = 2
no of student with name mathew and marks 55 = 2

scala>
```

Code for problem 1:

First I am loading the text file, and then applying count for number of rows,.

For tupled rdd first the rows have to be split and then mapped to a tuple.

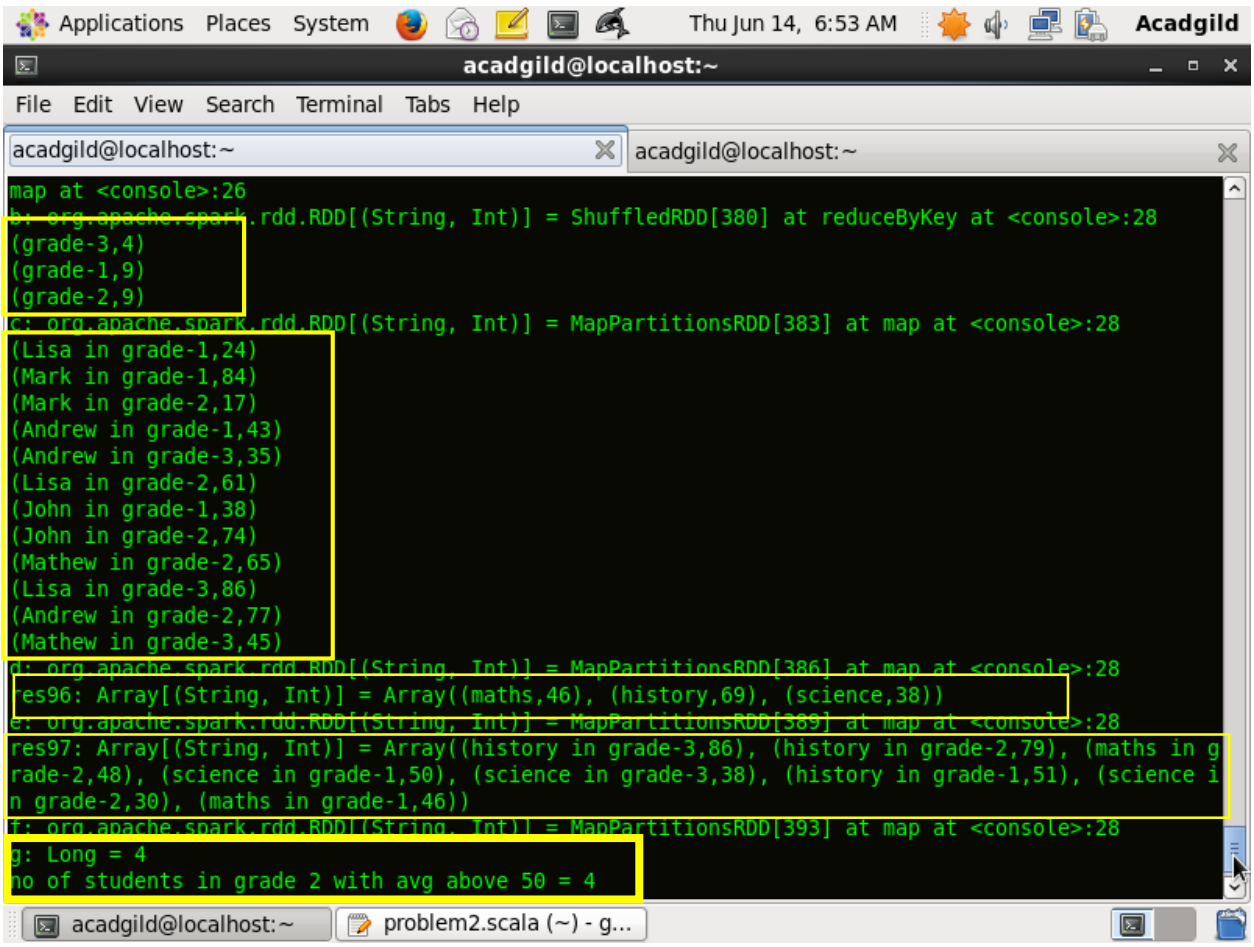
To find the number of unique subjects, first mapping only the subjects and then forming key value pair and then reduced by using reduceByKey and then getting its count

For finding count of student with name Mathew and marks 55 first mapping only student names and marks followed by filter to remove records which dosent match our condition and finally applying count

```
[acadgild@localhost ~]$ cat problem1.scala
val myfile = sc.textFile("19_Dataset.txt")
println("No of rows = " + myfile.count)

val a = myfile.map(x=>x.split(",")).map(s => (s(0),s(1),s(2),s(3),s(4)))
println("array of tuples")
a.collect

val b = a.map(s=>s._2).map(s=>(s,1)).reduceByKey(_+_).count
println("no of unique subjects = " + b)

val c = a.map(s=>(s._1,s._4)).filter(s=>(s._1=="Mathew" && s._2=="55")).count
println("no of student with name mathew and marks 55 = " + c)
```

Output for problem2:

First one can see the output for count of students per grade then average of student scores.
Followed by avg score of students in each subject then avg of subject but specifc to each grade and finally the number of students in grade 2 with avg of more than 50



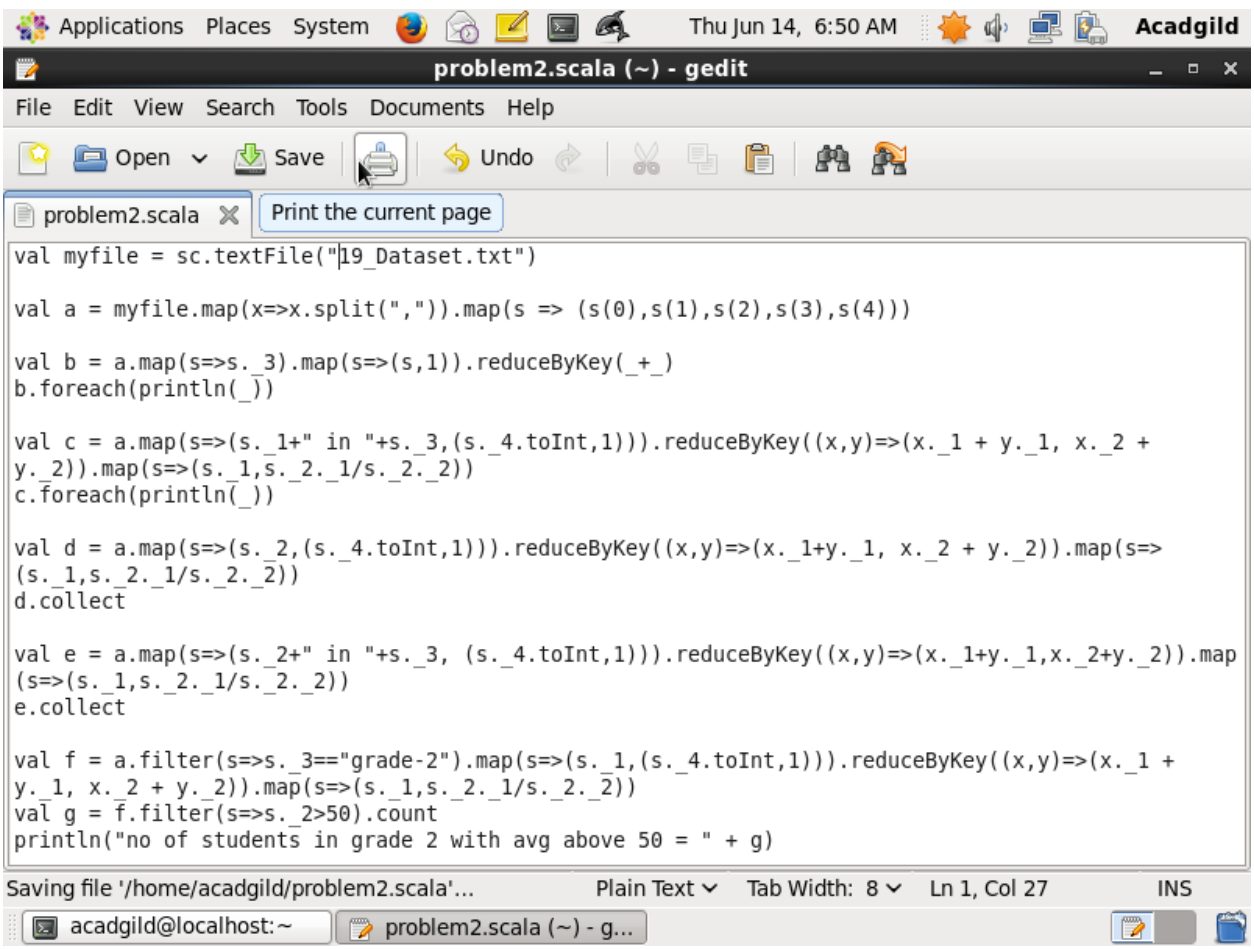
Code for problem2:

Val a contains the tupled rdd. B contains the contains the count of student In each grade

Val c contains avg score of students specdific to name and grade

Val d contains avg score of student in each subject

Val f contains the avg score of only grade 2 which is then filtered to find student in garde 2 with avg above 50




```
val myfile = sc.textFile("19_Dataset.txt")

val a = myfile.map(x=>x.split(",")).map(s => (s(0),s(1),s(2),s(3),s(4)))

val b = a.map(s=>s._3).map(s=>(s,1)).reduceByKey(_+_)
b.foreach(println(_))

val c = a.map(s=>(s._1+" in "+s._3,(s._4.toInt,1))).reduceByKey((x,y)=>(x._1 + y._1, x._2 +
y._2)).map(s=>(s._1,s._2._1/s._2._2))
c.foreach(println(_))

val d = a.map(s=>(s._2,(s._4.toInt,1))).reduceByKey((x,y)=>(x._1+y._1, x._2 + y._2)).map(s=>
(s._1,s._2._1/s._2._2))
d.collect

val e = a.map(s=>(s._2+" in "+s._3, (s._4.toInt,1))).reduceByKey((x,y)=>(x._1+y._1,x._2+y._2)).map
(s=>(s._1,s._2._1/s._2._2))
e.collect

val f = a.filter(s=>s._3=="grade-2").map(s=>(s._1,(s._4.toInt,1))).reduceByKey((x,y)=>(x._1 +
y._1, x._2 + y._2)).map(s=>(s._1,s._2._1/s._2._2))
val g = f.filter(s=>s._2>50).count
println("no of students in grade 2 with avg above 50 = " + g)
```

Code for problem 3:

```
val myfile = sc.textFile("19_Dataset.txt")

val a = myfile.map(x=>x.split(",")).map(s => (s(0),s(1),s(2),s(3),s(4)))

val b = a.map(s=>(s._1,(s._4.toInt,1))).reduceByKey((x,y)=>(x._1 + y._1, x._2 + y._2)).map(s=>
(s._1,s._2._1/s._2._2))
b.foreach(println(_))

val c = a.map(s=>(s._1+" in "+s._3,(s._4.toInt,1))).reduceByKey((x,y)=>(x._1 + y._1, x._2 +
y._2)).map(s=>(s._1.split(" ")(0),s._2._1/s._2._2))
c.foreach(println(_))

val newvalue = c.intersection(b).count

println("no of students with mentioned criteria = " + newvalue)
```

Output for problem3:

```
myfile: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/19_Dataset.txt MapPartitionsRDD[
514] at textFile at <console>:24
a: org.apache.spark.rdd.RDD[(String, String, String, String, String)] = MapPartitionsRDD[516] at
map at <console>:26
b: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[519] at map at <console>:28
(Mark,50)
(Andrew,46)
(Mathew,60)
(John,47)
(Lisa,58)
c: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[522] at map at <console>:28
(Lisa,24)
(Mark,84)
(Mark,17)
(Andrew,43)
(Andrew,35)
(Lisa,61)
(John,38)
(John,74)
(Mathew,65)
(Lisa,86)
(Andrew,77)
(Mathew,45)
newvalue: Long = 0
no of students with mentioned criteria = 0

scala>
```