

Assignment 19.1 RDD Deep Dive

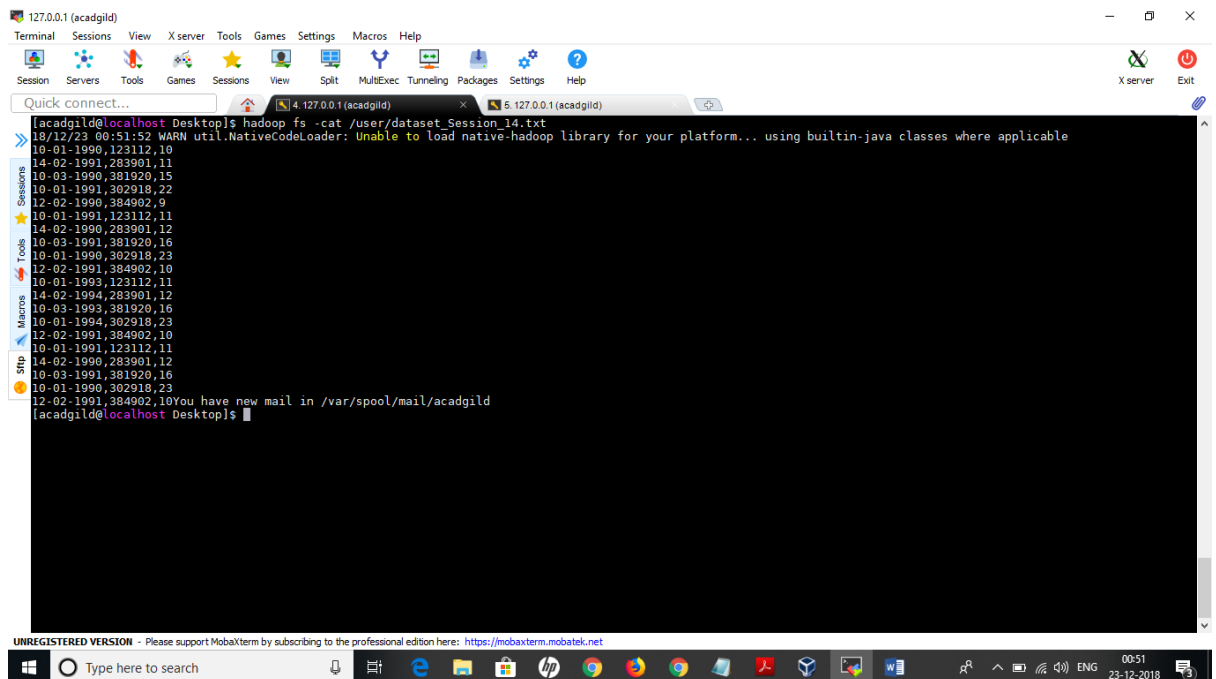
Task 1:

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

Solution:

```
val rows= sc.textFile("/user/dataset_Session_14.txt")  
  
rows.count()
```

Output:



The screenshot shows a terminal window titled "127.0.0.1 (acadgild)" with a menu bar (Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help) and a toolbar. The terminal displays the command `hadoop fs -cat /user/dataset_Session_14.txt` and its output, which is a list of 20 rows of data, each consisting of a date, a time, and a number. The output is as follows:

```
18/12/23 00:51:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
10-01-1990,123112,10  
14-02-1991,283901,11  
10-03-1990,381920,15  
10-01-1991,302918,22  
12-02-1990,384902,9  
10-01-1991,123112,11  
14-02-1990,283901,12  
10-03-1991,381920,16  
10-01-1990,302918,23  
12-02-1991,384902,10  
10-01-1993,123112,11  
14-02-1994,283901,12  
10-03-1993,381920,16  
10-01-1994,302918,23  
12-02-1991,384902,10  
10-01-1991,123112,11  
14-02-1990,283901,12  
10-03-1991,381920,16  
10-01-1990,302918,23  
12-02-1991,384902,10  
You have new mail in /var/spool/mail/acadgild  
[acadgild@localhost Desktop]$
```

The terminal window also shows a "Quick connect..." bar and a "UNREGISTERED VERSION" notice at the bottom.

The screenshot shows a MobaXterm terminal window titled '127.0.0.1 (acadgild)'. The terminal displays the following Scala code and its output:

```
scala> val rows= sc.textFile("/user/dataset_Session_14.txt")
rows: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_14.txt MapPartitionsRDD[14] at textFile at <console>:24

scala> rows.count()
res5: Long = 20

scala>
```

The terminal window has a menu bar (Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help) and a toolbar with icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help. The Windows taskbar at the bottom shows the search bar, task view, and various application icons. The system tray on the right indicates the time as 00:52 on 23-12-2018.

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

Solution:

```
cat Spark_word_file.txt
```

```
wc -w Spark_word_file.txt
```

```
val SparkFile = sc.textFile("/user/Spark_word_file.txt")
```

```
val words = SparkFile.flatMap(word=> word.split(" "))
```

```
words.count()
```

Output:

127.0.0.1 (acadgild)

Terminal Sessions View X server Tools Games Settings Macros Help

Quick connect...

4. 127.0.0.1 (acadgild) 5. 127.0.0.1 (acadgild)

```
[acadgild@localhost Desktop]$ hadoop fs -cat /user/Spark_word_file.txt
18/12/23 00:56:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
RDD stands for Resilient Distributed Datasets are Apache Spark's data abstraction, RDD is a logical reference of a dataset which is partitioned across many server machines in the cluster. RDDs are immutable and are self-recovered in case of failure. Dataset could be the data loaded externally by the user. RDDs can only be created by reading data from a stable storage such as HDFS or by transformations on existing RDDs.
[acadgild@localhost Desktop]$ wc -w Spark_word_file.txt
70 Spark_word_file.txt
[acadgild@localhost Desktop]$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Type here to search

127.0.0.1 (acadgild)

Terminal Sessions View X server Tools Games Settings Macros Help

Quick connect...

4. 127.0.0.1 (acadgild) 5. 127.0.0.1 (acadgild)

```
scala> val SparkFile = sc.textFile("/user/Spark_word_file.txt")
SparkFile: org.apache.spark.rdd.RDD[String] = /user/Spark_word_file.txt MapPartitionsRDD[16] at textFile at <console>:24

scala> val words = SparkFile.flatMap(word=> word.split(" "))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[17] at flatMap at <console>:26

scala> words.count()
res6: Long = 70

scala>
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Type here to search

- 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:

cat Spark_word_file1.txt

wc -w Spark_word_file1.txt

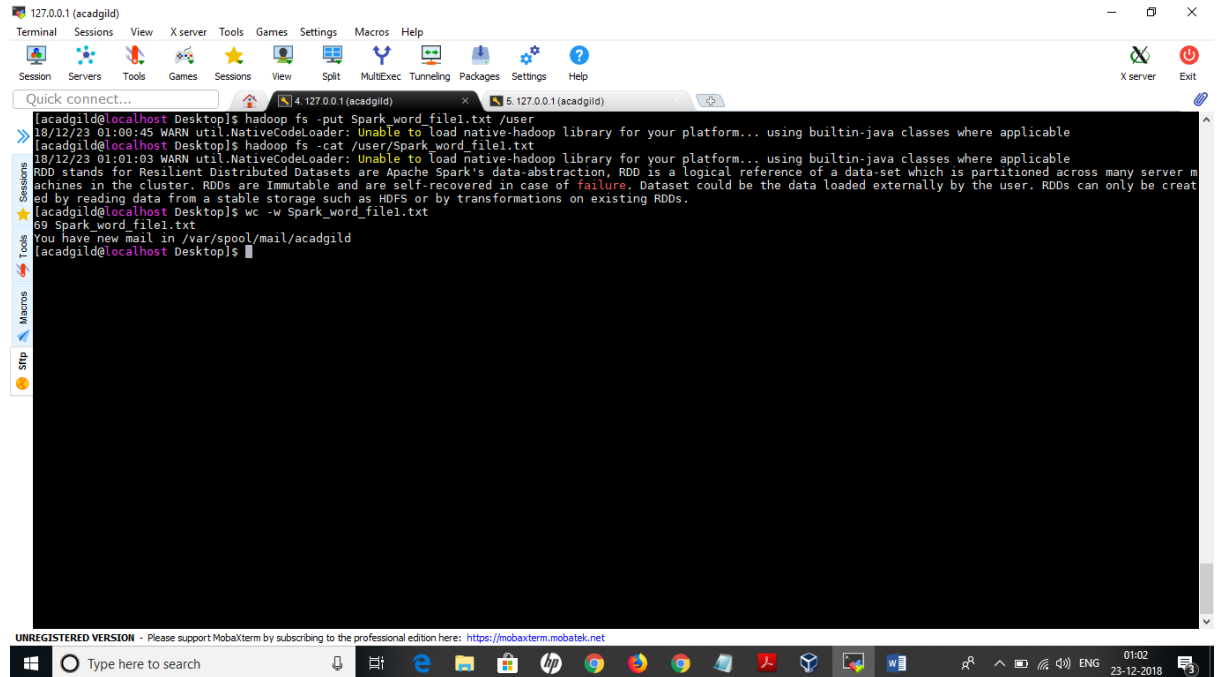
val SparkFile1 = sc.textFile("/user/Spark_word_file1.txt")

val words = SparkFile1.flatMap(word=> word.split(" "))

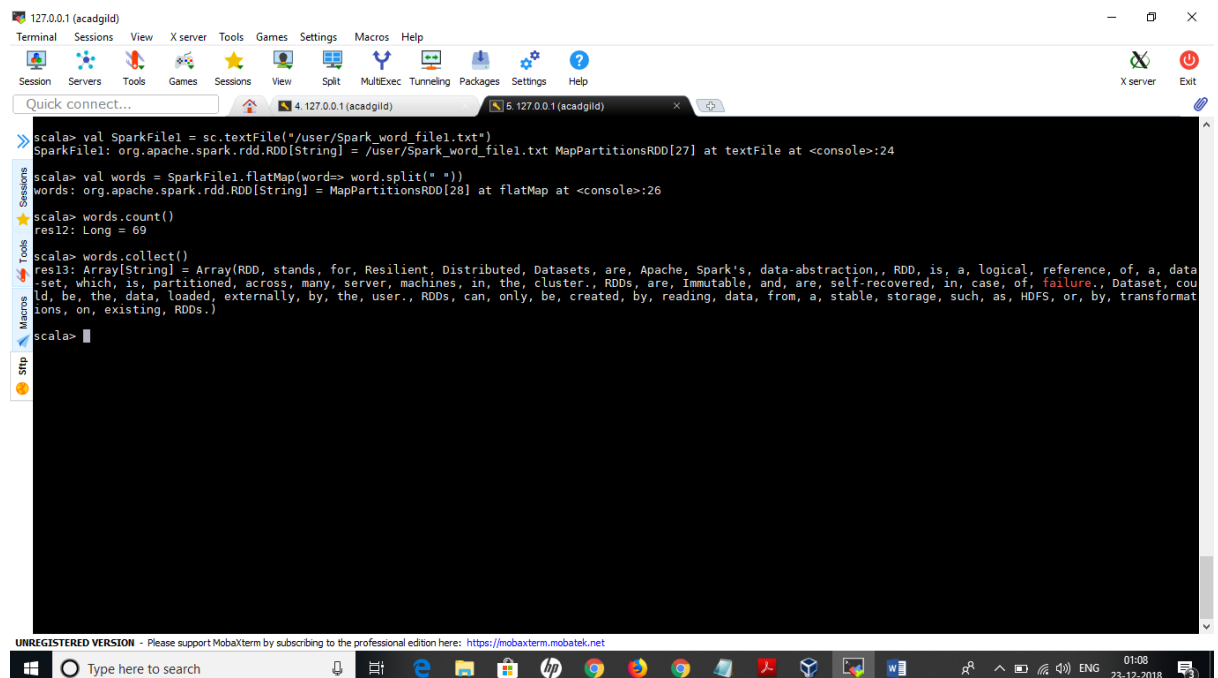
words.count()

words.collect()

Output:



```
127.0.0.1 (acadgild)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 4. 127.0.0.1 (acadgild) 5. 127.0.0.1 (acadgild)
[acadgild@localhost Desktop]$ hadoop fs -put Spark_word_file1.txt /user
18/12/23 01:00:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[acadgild@localhost Desktop]$ hadoop fs -cat /user/Spark_word_file1.txt
18/12/23 01:01:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
RDD stands for Resilient Distributed Datasets are Apache Spark's data-abstraction, RDD is a logical reference of a data-set which is partitioned across many server machines in the cluster. RDDs are Immutable and are self-recovered in case of failure. Dataset could be the data loaded externally by the user. RDDs can only be created by reading data from a stable storage such as HDFS or by transformations on existing RDDs.
[acadgild@localhost Desktop]$ wc -w Spark_word_file1.txt
69 Spark_word_file1.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Desktop]$
```



```
127.0.0.1 (acadgild)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 4. 127.0.0.1 (acadgild) 5. 127.0.0.1 (acadgild)
scala> val SparkFile1 = sc.textFile("/user/Spark_word_file1.txt")
SparkFile1: org.apache.spark.rdd.RDD[String] = /user/Spark_word_file1.txt MapPartitionsRDD[27] at textFile at <console>:24
scala> val words = SparkFile1.flatMap(word=> word.split(" "))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[28] at flatMap at <console>:26
scala> words.count()
res12: Long = 69
scala> words.collect()
res13: Array[String] = Array(RDD, stands, for, Resilient, Distributed, Datasets, are, Apache, Spark's, data-abstraction, RDD, is, a, logical, reference, of, a, data-set, which, is, partitioned, across, many, server, machines, in, the, cluster, RDDs, are, Immutable, and, are, self-recovered, in, case, of, failure, Dataset, could, be, the, data, loaded, externally, by, the, user, RDDs, can, only, be, created, by, reading, data, from, a, stable, storage, such, as, HDFS, or, by, transformations, on, existing, RDDs.)
scala>
```

Task 2:

Problem Statement 1:

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

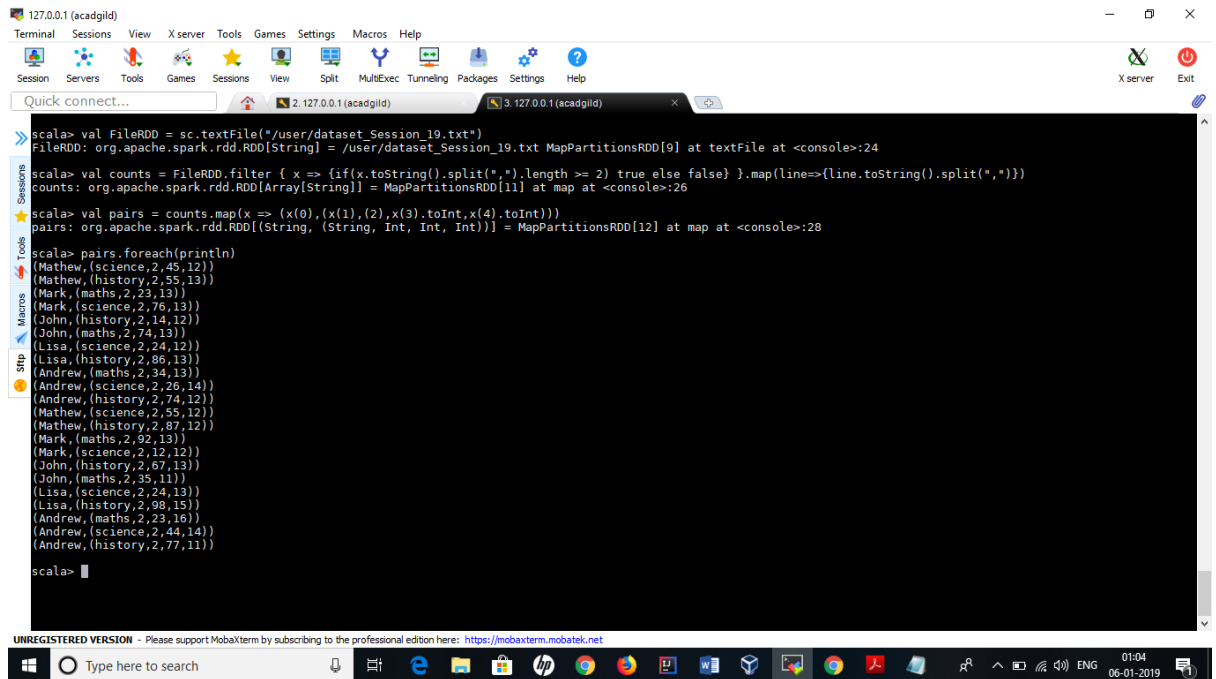
Solution:

```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
}.map(line=>{line.toString().split(",")})

val pairs = counts.map(x => (x(0),(x(1),(2),x(3).toInt,x(4).toInt)))

pairs.foreach(println)
```



```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[9] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(line=>{line.toString().split(",")})
counts: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[11] at map at <console>:26

scala> val pairs = counts.map(x => (x(0),(x(1),(2),x(3).toInt,x(4).toInt)))
pairs: org.apache.spark.rdd.RDD[(String, (String, Int, Int, Int))] = MapPartitionsRDD[12] at map at <console>:28

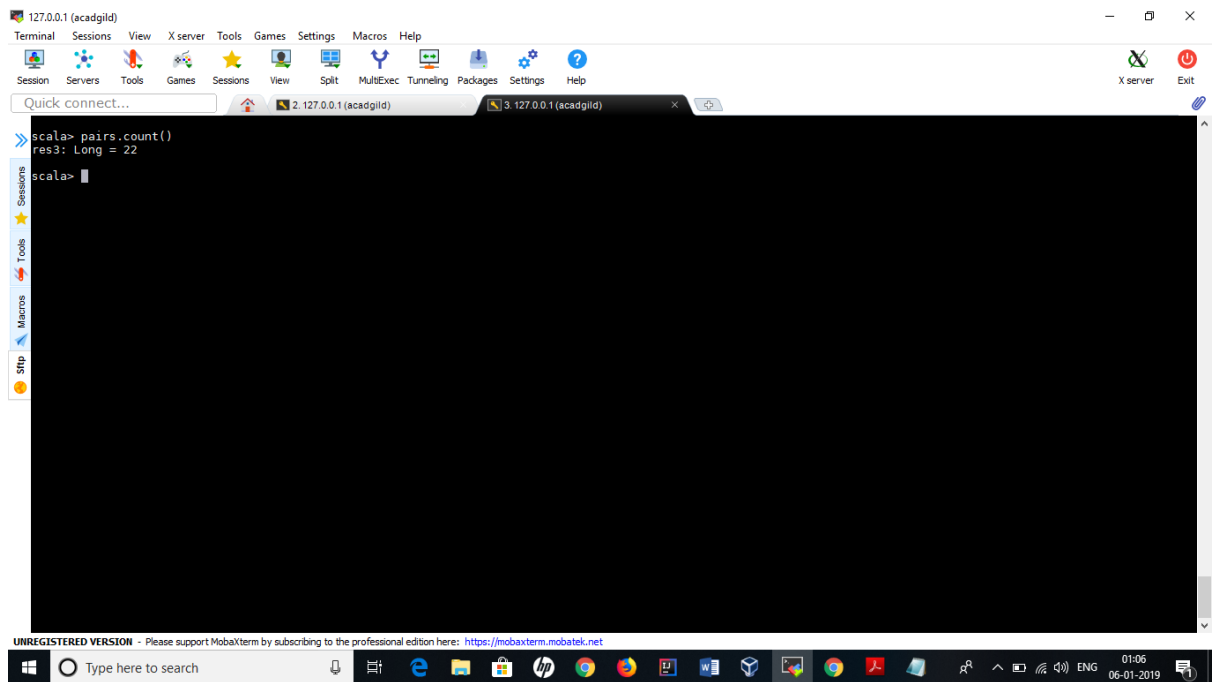
scala> pairs.foreach(println)
(Mathew,(science,2,45,12))
(Mathew,(history,2,55,13))
(Mark,(maths,2,23,13))
(Mark,(science,2,76,13))
(John,(history,2,14,12))
(John,(maths,2,74,13))
(Lisa,(science,2,24,12))
(Lisa,(history,2,88,13))
(Andrew,(maths,2,34,13))
(Andrew,(science,2,26,14))
(Andrew,(history,2,74,12))
(Mathew,(science,2,55,12))
(Mathew,(history,2,87,12))
(Mark,(maths,2,92,13))
(Mark,(science,2,12,12))
(John,(history,2,67,13))
(John,(maths,2,35,11))
(Lisa,(science,2,24,13))
(Lisa,(history,2,98,15))
(Andrew,(maths,2,23,16))
(Andrew,(science,2,44,14))
(Andrew,(history,2,77,11))

scala>
```

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

Solution:

```
pairs.count()
```



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

Solution:

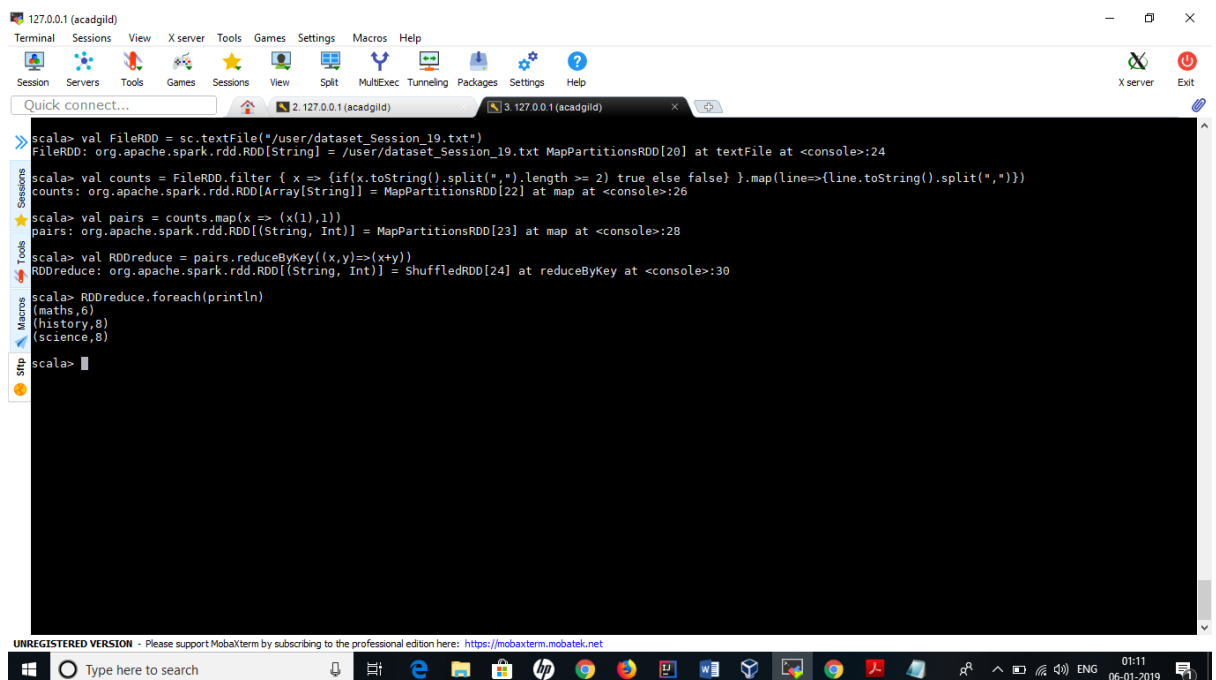
```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
}.map(line=>{line.toString().split(",")})

val pairs = counts.map(x => (x(1),1))

val RDDreduce = pairs.reduceByKey((x,y)=>(x+y))

RDDreduce.foreach(println)
```



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

Solution:

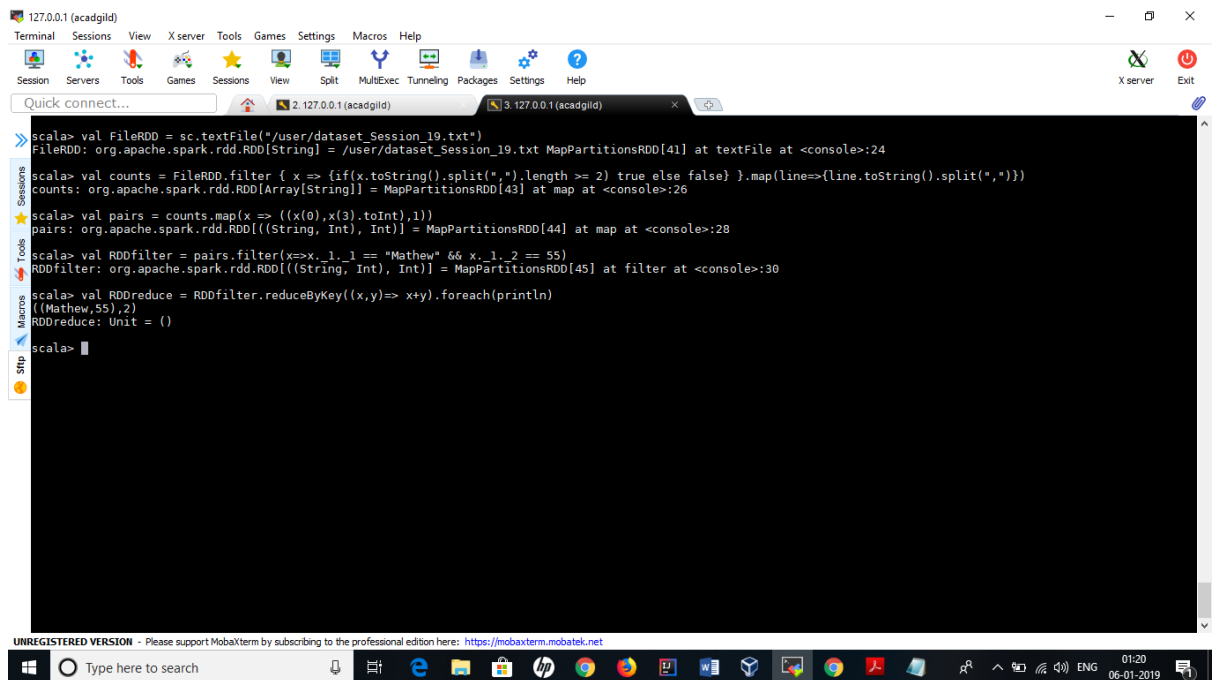
```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
}.map(line=>{line.toString().split(",")})

val pairs = counts.map(x => ((x(0),x(3).toInt),1))

val RDDfilter = pairs.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)

val RDDreduce = RDDfilter.reduceByKey((x,y)=> x+y).foreach(println)
```



```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[41] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(line=>{line.toString().split(",")})
counts: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[43] at map at <console>:26

scala> val pairs = counts.map(x => ((x(0),x(3).toInt),1))
pairs: org.apache.spark.rdd.RDD[(String, Int, Int)] = MapPartitionsRDD[44] at map at <console>:28

scala> val RDDfilter = pairs.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)
RDDfilter: org.apache.spark.rdd.RDD[(String, Int, Int)] = MapPartitionsRDD[45] at filter at <console>:30

scala> val RDDreduce = RDDfilter.reduceByKey((x,y)=> x+y).foreach(println)
((Mathew,55),2)
RDDreduce: Unit = ()

scala>
```

Problem Statement 2:

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

Solution:

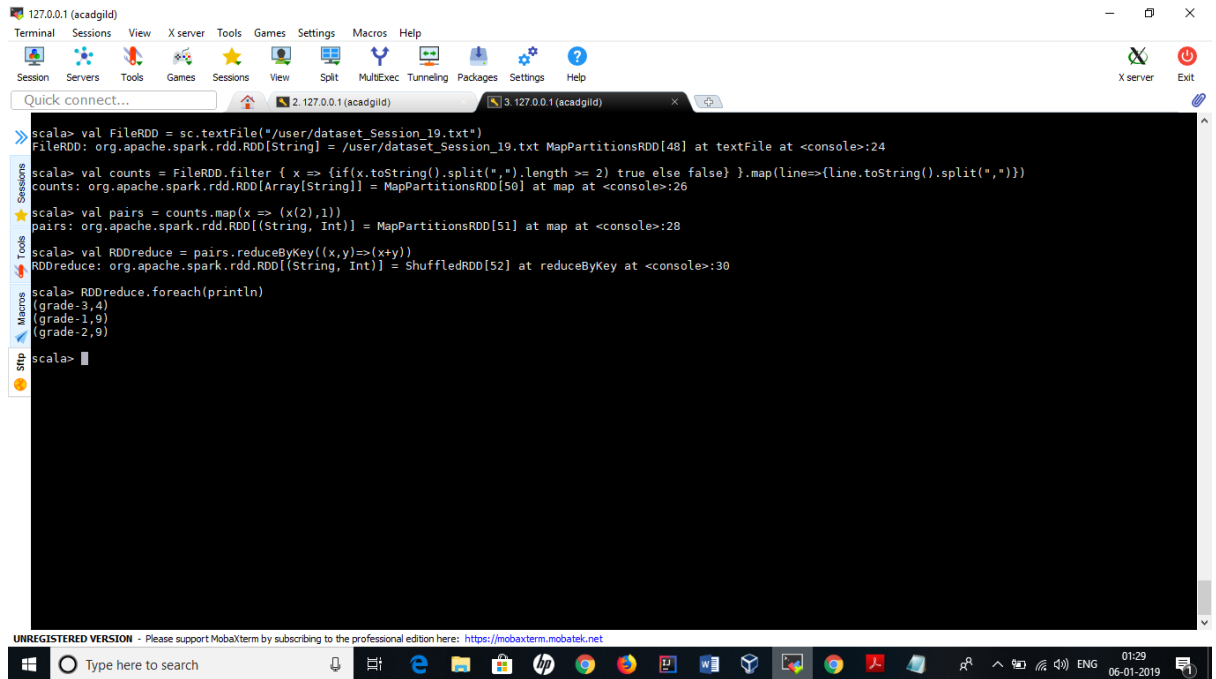
```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
}.map(line=>{line.toString().split(",")})

val pairs = counts.map(x => (x(2),1))

val RDDreduce = pairs.reduceByKey((x,y)=>(x+y))

RDDreduce.foreach(println)
```



```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[48] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(line=>{line.toString().split(",")})
counts: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[50] at map at <console>:26

scala> val pairs = counts.map(x => (x(2),1))
pairs: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[51] at map at <console>:28

scala> val RDDreduce = pairs.reduceByKey((x,y)=>(x+y))
RDDreduce: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[52] at reduceByKey at <console>:30

scala> RDDreduce.foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)

scala>
```

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

Solution:

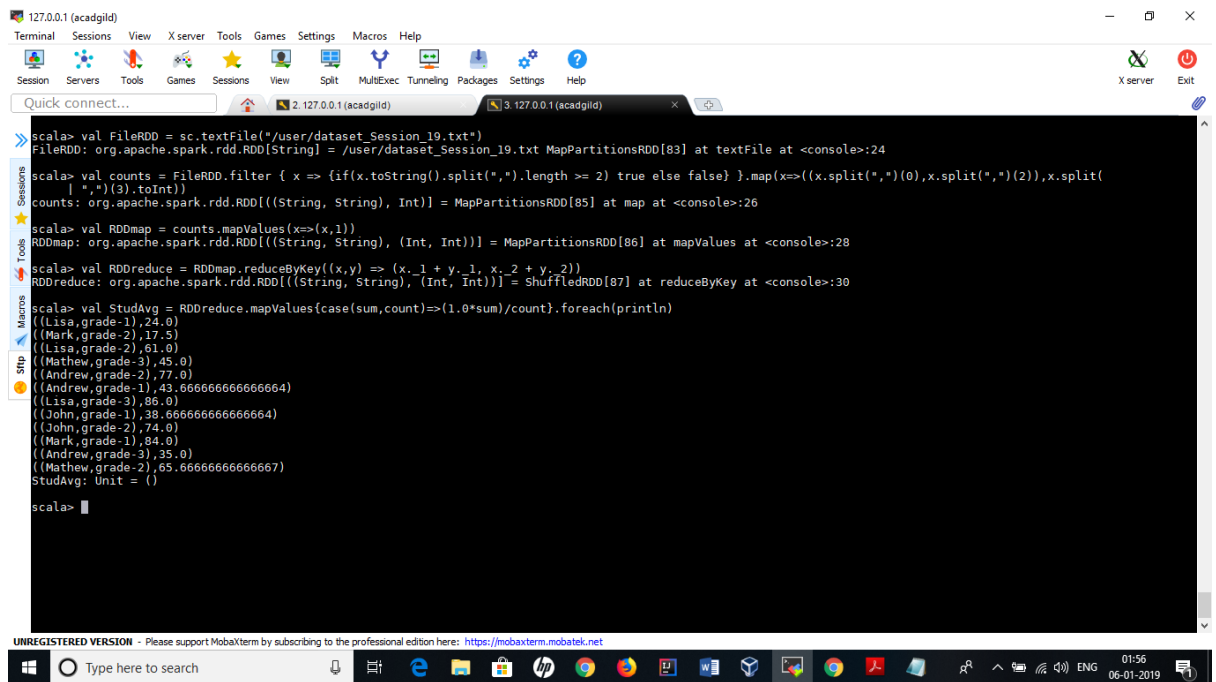
```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
.map(x=>({x.split(",")(0),x.split(",")(2)),x.split(
"")(3).toInt))

val RDDmap = counts.mapValues(x=>(x,1))

val RDDreduce = RDDmap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

val StudAvg = RDDreduce.mapValues{case(sum,count)=>{(1.0*sum)/count}}.foreach(println)
```

```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[83] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(
  ",")(3).toInt))
counts: org.apache.spark.rdd.RDD[(String, String), Int] = MapPartitionsRDD[85] at map at <console>:26

scala> val RDDmap = counts.mapValues(x=>(x,1))
RDDmap: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[86] at mapValues at <console>:28

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[87] at reduceByKey at <console>:30

scala> val StudAvg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),34.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
StudAvg: Unit = ()

scala>
```

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

Solution:

```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

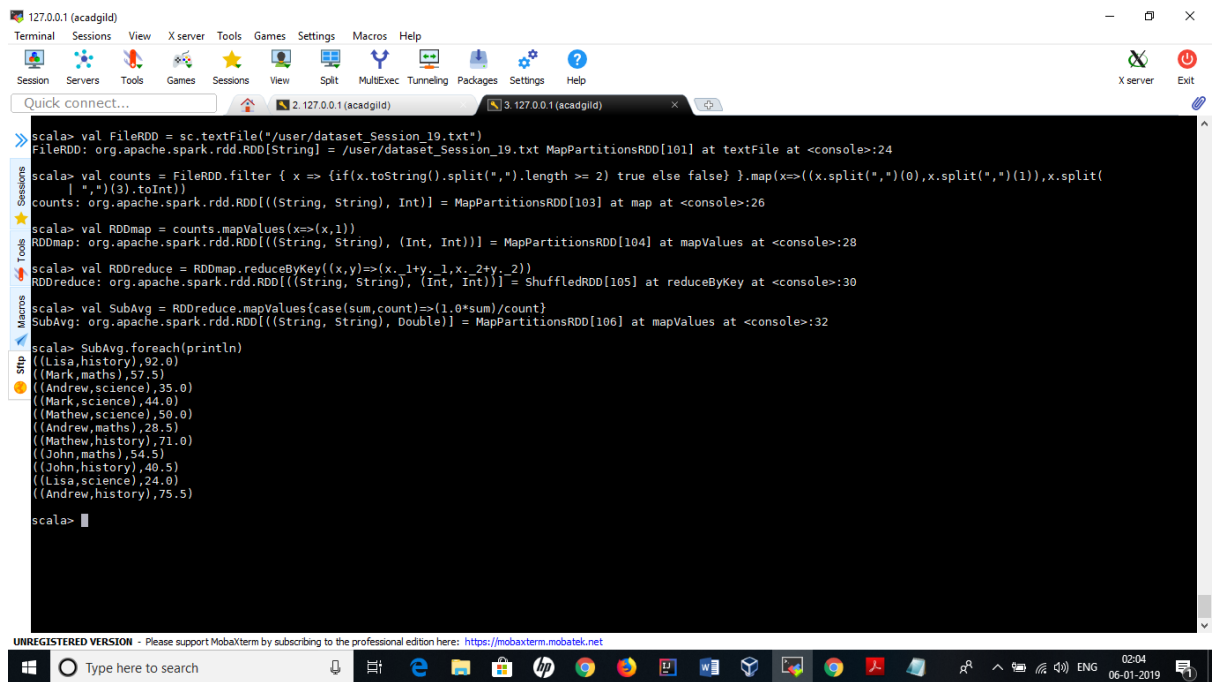
val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
}.map(x=>((x.split(",")(0),x.split(",")(1)),x.split(
","")(3).toInt))

val RDDmap = counts.mapValues(x=>(x,1))

val RDDreduce = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))

val SubAvg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}

SubAvg.foreach(println)
```



```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[101] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
counts: org.apache.spark.rdd.RDD[(String, String), Int] = MapPartitionsRDD[103] at map at <console>:26

scala> val RDDmap = counts.mapValues(x=>(x,1))
RDDmap: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[104] at mapValues at <console>:28

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[105] at reduceByKey at <console>:30

scala> val SubAvg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}
SubAvg: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[106] at mapValues at <console>:32

scala> SubAvg.foreach(println)
((Lisa,history),92.0)
((Mark,maths),57.5)
((Andrew,science),35.0)
((Mark,science),44.0)
((Mathew,science),50.0)
((Andrew,maths),28.5)
((Mathew,history),71.0)
((John,maths),54.5)
((John,history),40.5)
((Lisa,science),24.0)
((Andrew,history),75.5)

scala>
```

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

Solution:

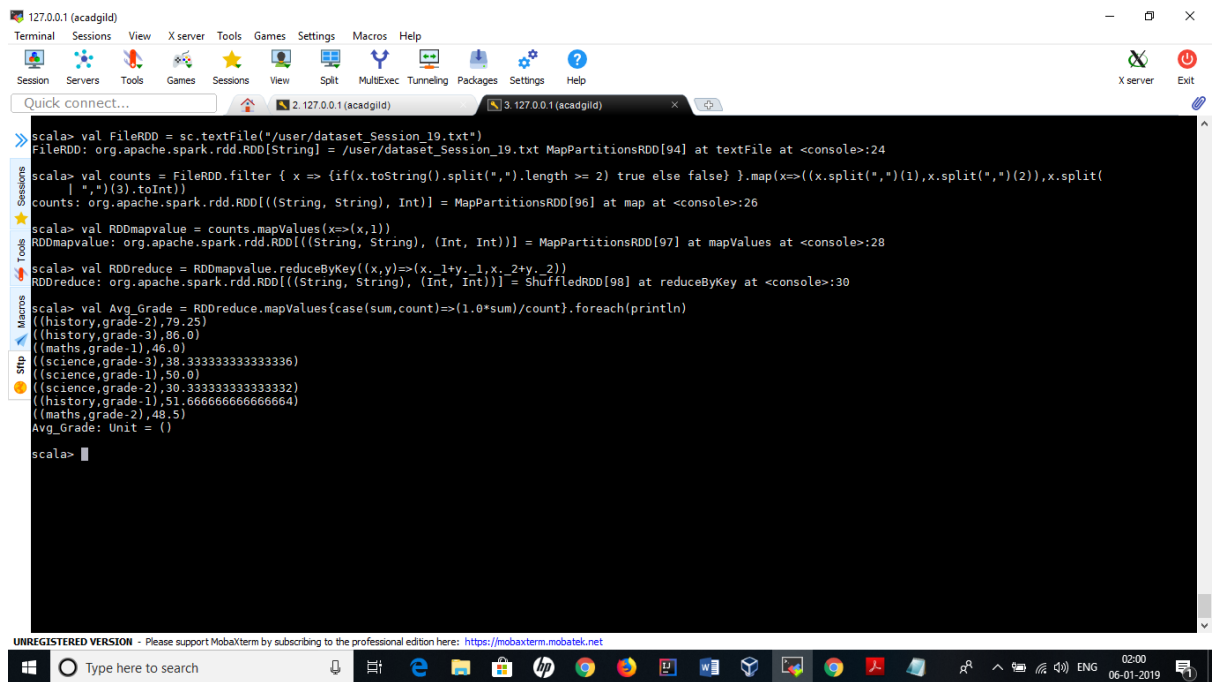
```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
.map(x=>((x.split(",")(1),x.split(",")(2)),x.split(
"")(3).toInt))

val RDDmapvalue = counts.mapValues(x=>(x,1))

val RDDreduce = RDDmapvalue.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))

val Avg_Grade =
RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}.foreach(println)
```



```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[94] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))
counts: org.apache.spark.rdd.RDD[(String, String), Int]] = MapPartitionsRDD[96] at map at <console>:26

scala> val RDDmapvalue = counts.mapValues(x=>(x,1))
RDDmapvalue: org.apache.spark.rdd.RDD[(String, String), (Int, Int))] = MapPartitionsRDD[97] at mapValues at <console>:28

scala> val RDDreduce = RDDmapvalue.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
RDDreduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[98] at reduceByKey at <console>:30

scala> val Avg_Grade = RDDreduce.mapValues(case(sum,count)=>(1.0*sum)/count).foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),30.333333333333336)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
Avg_Grade: Unit = ()

scala>
```

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

Solution:

```
val FileRDD = sc.textFile("/user/dataset_Session_19.txt")

val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false}
}.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(
"",")(3).toInt))

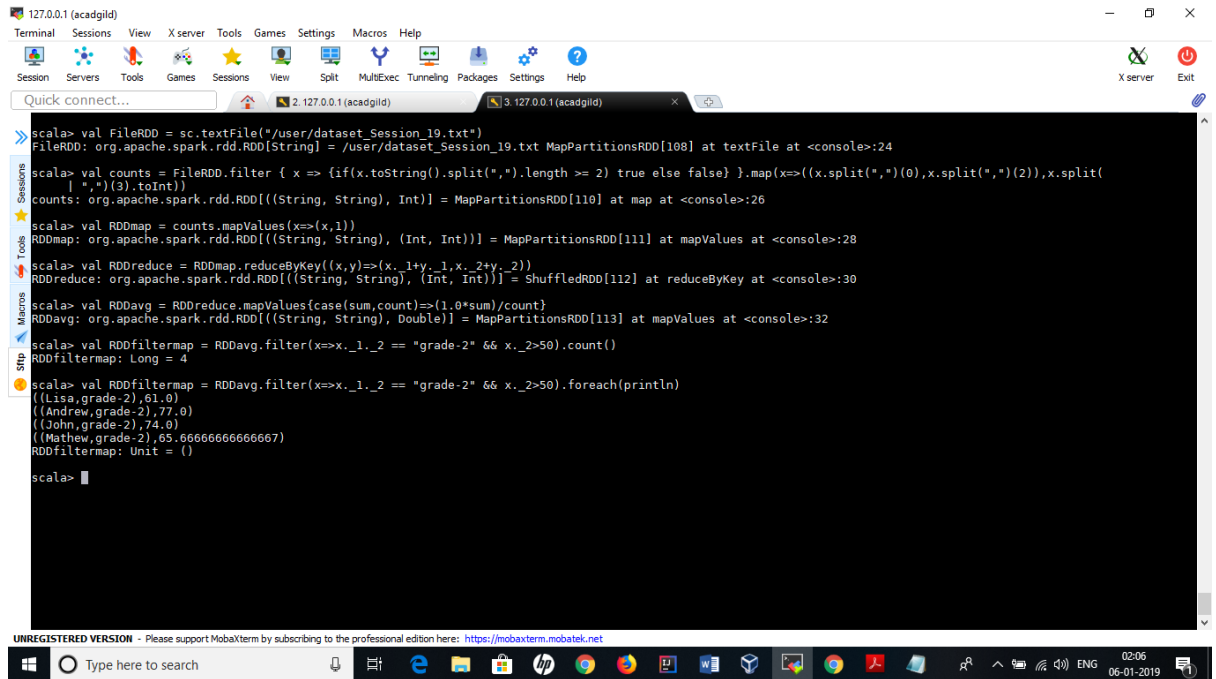
val RDDmap = counts.mapValues(x=>(x,1))

val RDDreduce = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))

val RDDavg = RDDreduce.mapValues{case(sum,count)=>(1.0*sum)/count}

val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).count()

val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).foreach(println)
```



```
scala> val FileRDD = sc.textFile("/user/dataset_Session_19.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[108] at textFile at <console>:24

scala> val counts = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(
  ",")(3).toInt))
counts: org.apache.spark.rdd.RDD[(String, String), Int] = MapPartitionsRDD[110] at map at <console>:26

scala> val RDDmap = counts.mapValues(x=>(x,1))
RDDmap: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[111] at mapValues at <console>:28

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[112] at reduceByKey at <console>:30

scala> val RDDavg = RDDreduce.mapValues{case (sum,count)=>(1.0*sum)/count}
RDDavg: org.apache.spark.rdd.RDD[(String, String), Double] = MapPartitionsRDD[113] at mapValues at <console>:32

scala> val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).count()
RDDfiltermap: Long = 4

scala> val RDDfiltermap = RDDavg.filter(x=>x._1._2 == "grade-2" && x._2>50).foreach(println)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)
RDDfiltermap: Unit = ()

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

Solution:

```
val FileRDD1 = sc.textFile("/user/dataset_Session_19.txt")

val counts1 = FileRDD1.filter { x => {if(x.toString().split(",").length >= 2) true else false} }
.map(x=>(x.split(",")(0),x.split(",")(3).toInt))

val studAvg = counts1.mapValues(x=>(x,1))

val studReduce = studAvg.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))

val Avg_Stud = studReduce.mapValues{case (sum,count) => (1.0 * sum)/count}

val FileRDD2 = sc.textFile("/user/dataset_Session_19.txt")

val counts2 = FileRDD2.filter { x => {if(x.toString().split(",").length >= 2) true else false} }
.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(
","),3).toInt))

val grade = counts2.mapValues(x=>(x,1))

val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))

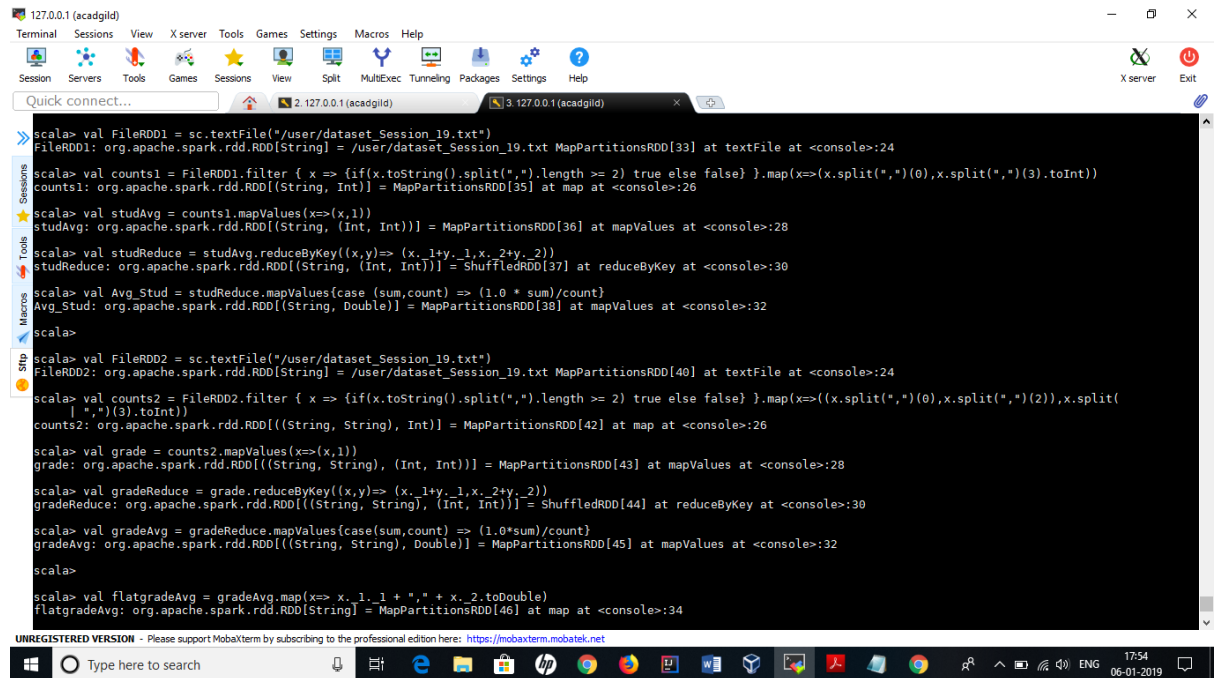
val gradeAvg = gradeReduce.mapValues{case (sum,count) => (1.0*sum)/count}
```

```
val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
```

```
val flatAvg_Stud = Avg_Stud.map(x=> x._1 + "," + x._2)
```

```
val commanval = flatgradeAvg.intersection(flatAvg_Stud)
```

```
commanval.foreach(println)
```



The screenshot shows a Scala REPL window with the following code and output:

```
scala> val FileRDD1 = sc.textFile("/user/dataset_Session_19.txt")
FileRDD1: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[33] at textFile at <console>:24

scala> val counts1 = FileRDD1.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>(x.split(",")(0),x.split(",")(3).toInt))
counts1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[35] at map at <console>:26

scala> val studAvg = counts1.mapValues(x=>(x._1))
studAvg: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[36] at mapValues at <console>:28

scala> val studReduce = studAvg.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
studReduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[37] at reduceByKey at <console>:30

scala> val Avg_Stud = studReduce.mapValues(case (sum,count) => (1.0 * sum)/count)
Avg_Stud: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[38] at mapValues at <console>:32

scala>

scala> val FileRDD2 = sc.textFile("/user/dataset_Session_19.txt")
FileRDD2: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[40] at textFile at <console>:24

scala> val counts2 = FileRDD2.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
counts2: org.apache.spark.rdd.RDD[(String, String), (Int)] = MapPartitionsRDD[42] at map at <console>:26

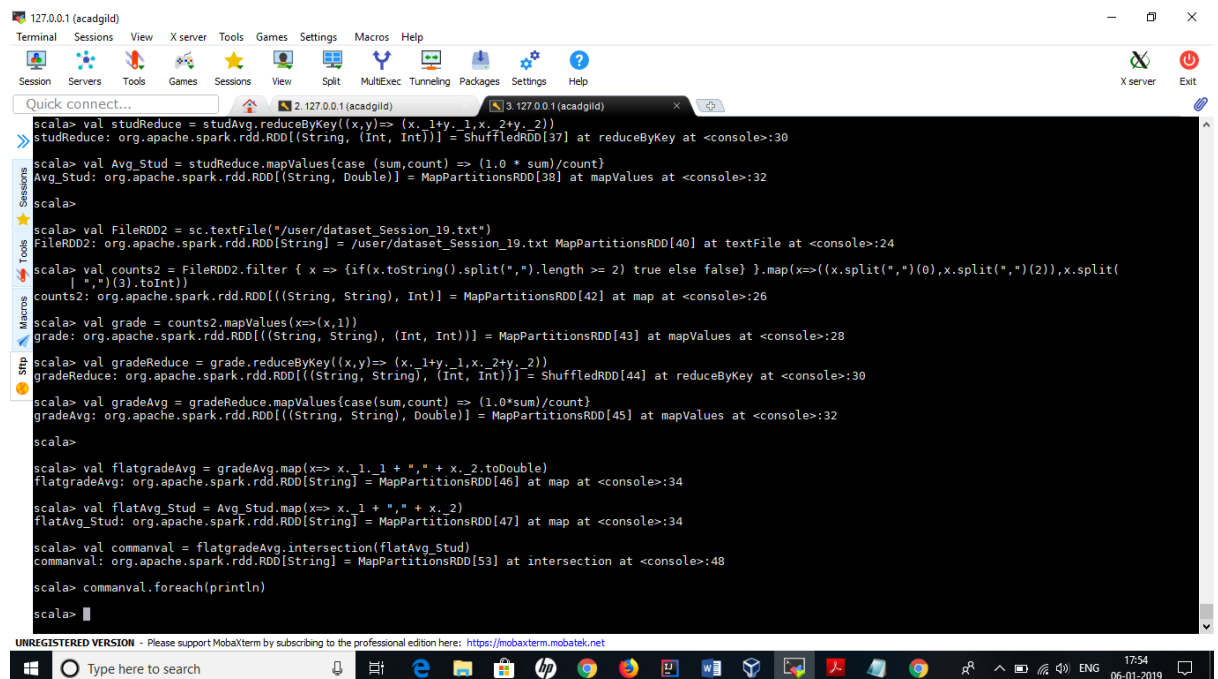
scala> val grade = counts2.mapValues(x=>(x._1))
grade: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[43] at mapValues at <console>:28

scala> val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
gradeReduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[44] at reduceByKey at <console>:30

scala> val gradeAvg = gradeReduce.mapValues(case (sum,count) => (1.0*sum)/count)
gradeAvg: org.apache.spark.rdd.RDD[(String, String), (Double)] = MapPartitionsRDD[45] at mapValues at <console>:32

scala>

scala> val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[46] at map at <console>:34
```



The screenshot shows a Scala REPL window with the following code and output:

```
scala> val studReduce = studAvg.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
studReduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[37] at reduceByKey at <console>:30

scala> val Avg_Stud = studReduce.mapValues(case (sum,count) => (1.0 * sum)/count)
Avg_Stud: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[38] at mapValues at <console>:32

scala>

scala> val FileRDD2 = sc.textFile("/user/dataset_Session_19.txt")
FileRDD2: org.apache.spark.rdd.RDD[String] = /user/dataset_Session_19.txt MapPartitionsRDD[40] at textFile at <console>:24

scala> val counts2 = FileRDD2.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
counts2: org.apache.spark.rdd.RDD[(String, String), (Int)] = MapPartitionsRDD[42] at map at <console>:26

scala> val grade = counts2.mapValues(x=>(x._1))
grade: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = MapPartitionsRDD[43] at mapValues at <console>:28

scala> val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
gradeReduce: org.apache.spark.rdd.RDD[(String, String), (Int, Int)] = ShuffledRDD[44] at reduceByKey at <console>:30

scala> val gradeAvg = gradeReduce.mapValues(case (sum,count) => (1.0*sum)/count)
gradeAvg: org.apache.spark.rdd.RDD[(String, String), (Double)] = MapPartitionsRDD[45] at mapValues at <console>:32

scala>

scala> val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[46] at map at <console>:34

scala> val flatAvg_Stud = Avg_Stud.map(x=> x._1 + "," + x._2)
flatAvg_Stud: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[47] at map at <console>:34

scala> val commanval = flatgradeAvg.intersection(flatAvg_Stud)
commanval: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[53] at intersection at <console>:48

scala> commanval.foreach(println)

scala>
```