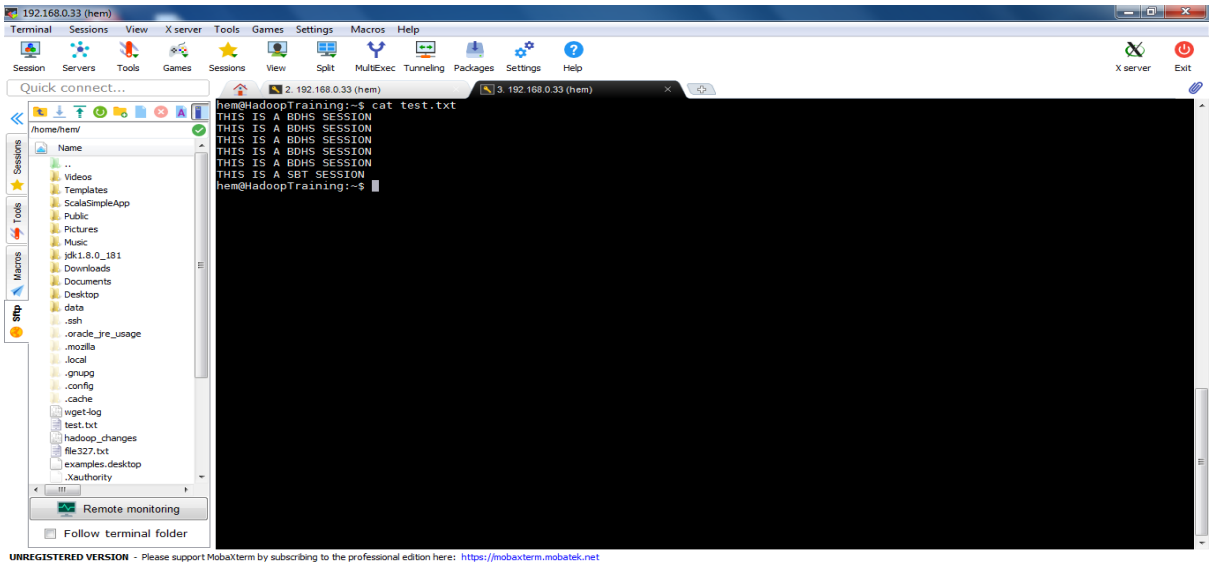
Task 1

Text File Used for Task 1



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

Spark Solution-:

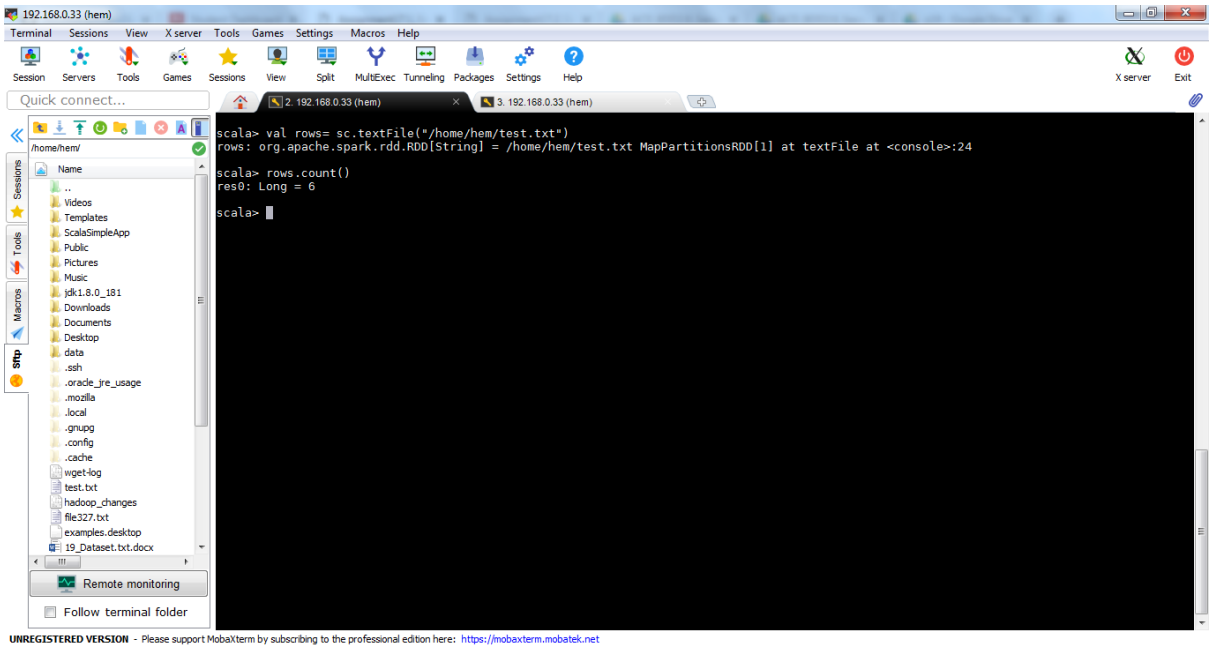
scala> val rows= sc.textFile("/home/hem/test.txt")

rows: org.apache.spark.rdd.RDD[String] = /home/hem/test.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> rows.count()

res0: Long = 6

Output-:



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

Spark Solution-:

scala> val rows= sc.textFile("/home/hem/test.txt")

rows: org.apache.spark.rdd.RDD[String] = /home/hem/test.txt MapPartitionsRDD[3] at textFile at <console>:24

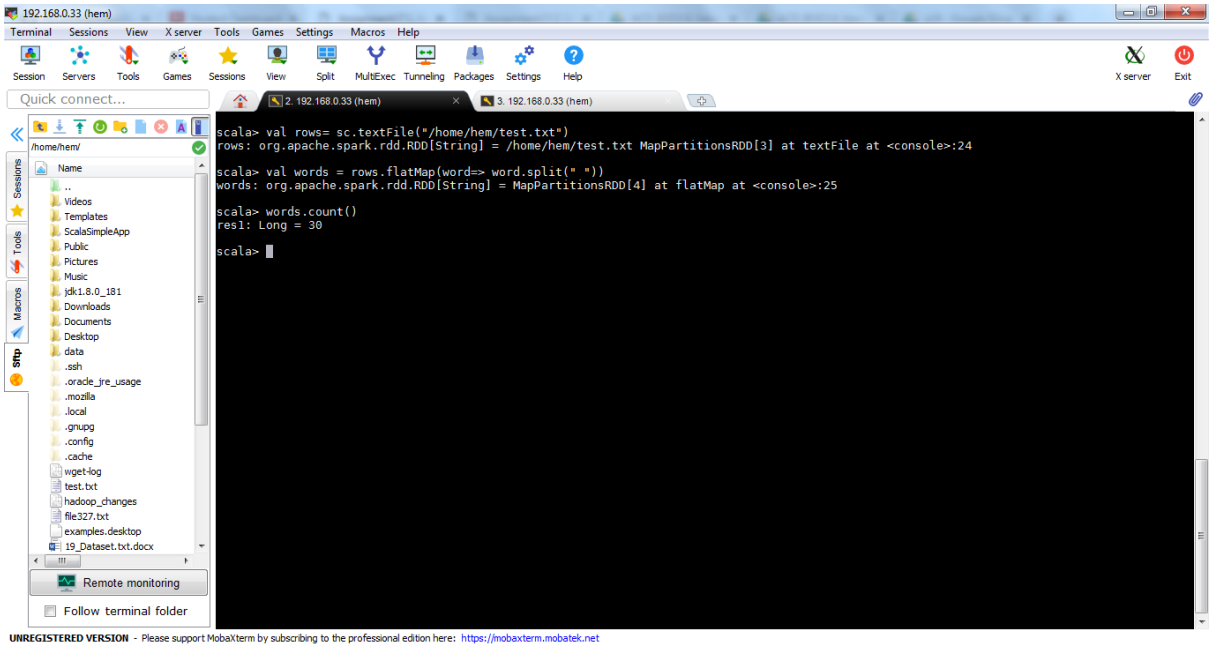
scala> val words = rows.flatMap(word=> word.split(" "))

words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at flatMap at <console>:25

scala> words.count()

res1: Long = 30

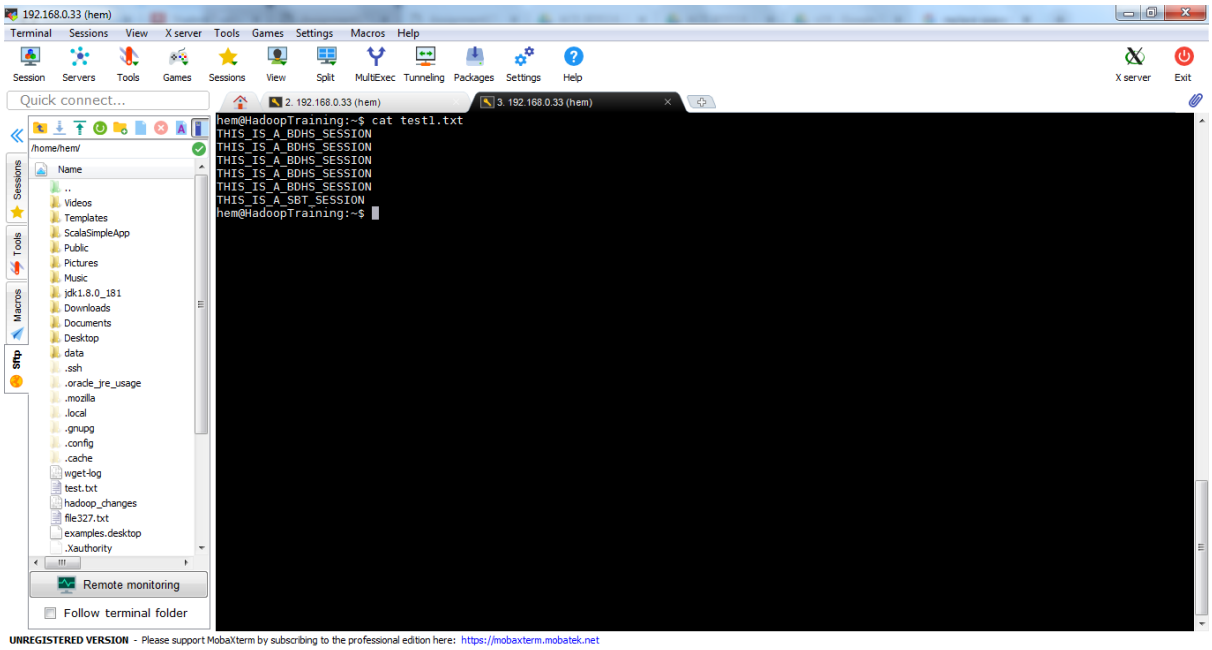
Output-:



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.

Task 2

Text Document-:



Spark Solution-:

scala> val base1 = sc.textFile("/home/hem/test1.txt")

base1: org.apache.spark.rdd.RDD[String] = /home/hem/test1.txt MapPartitionsRDD[14] at textFile at <console>:24

scala> val words = base1.flatMap(word=> word.split("-"))

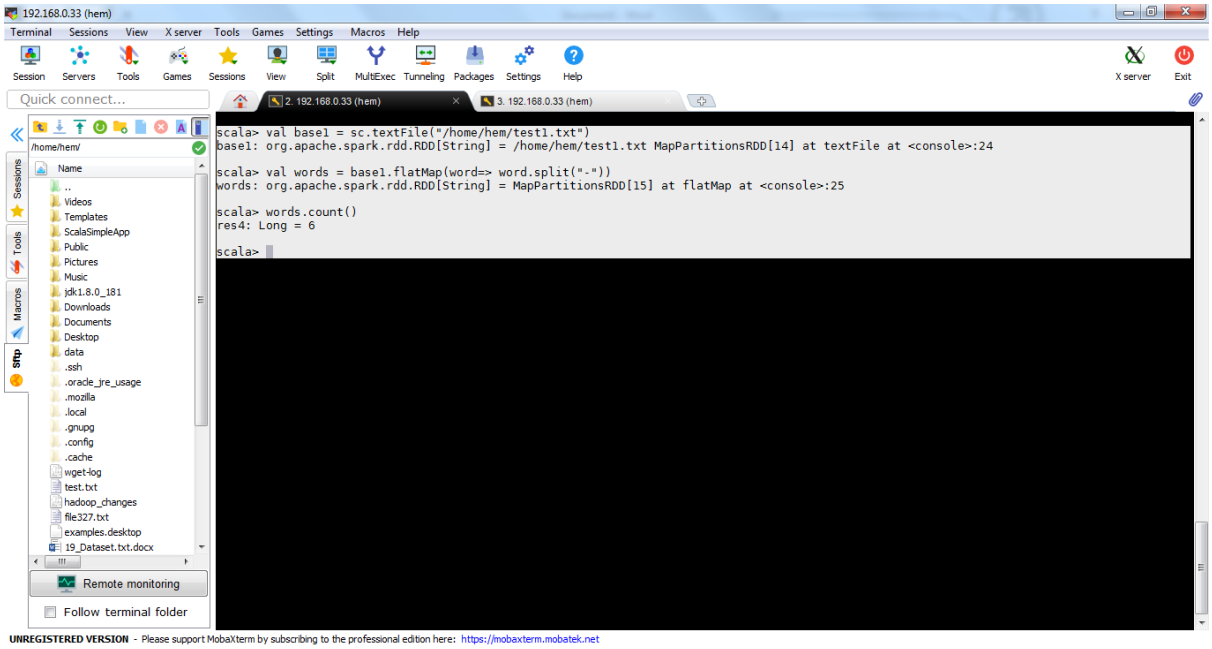
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at flatMap at <console>:25

scala> words.count()

res4: Long = 6

scala>

Output-:



Problem Statement 1:

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

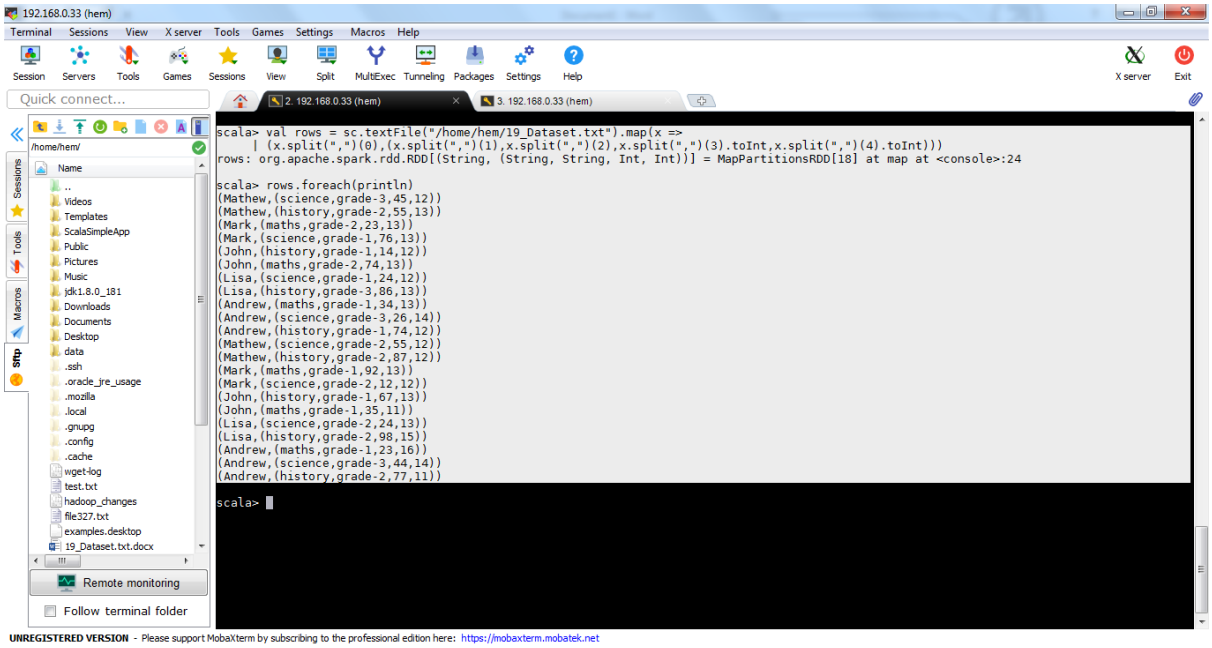
Spark Solution-:

```
scala> val rows = sc.textFile("/home/hem/19_Dataset.txt").map(x =>
    | (x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt,x.split(",")(4).toInt)))
rows: org.apache.spark.rdd.RDD[(String, (String, String, Int, Int))] = MapPartitionsRDD[18] at map at <console>:24
scala> rows.foreach(println)
(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))

Output-:



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

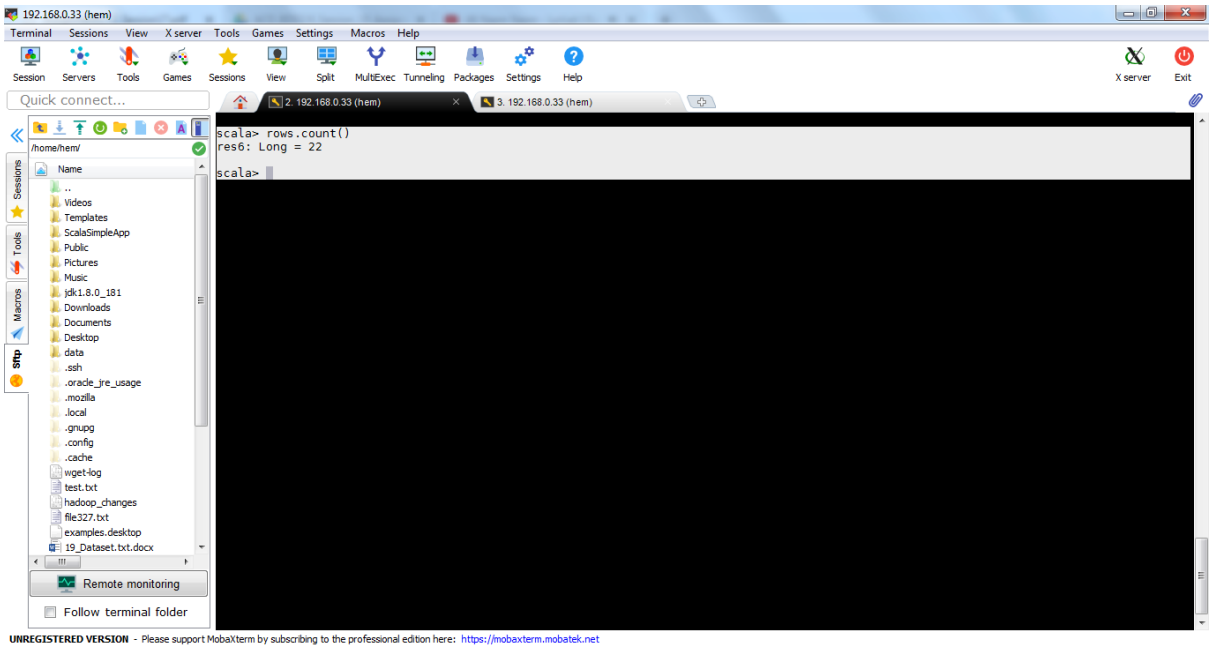
Spark Solution-:

scala> rows.count()

res6: Long = 22

scala>

Output-:

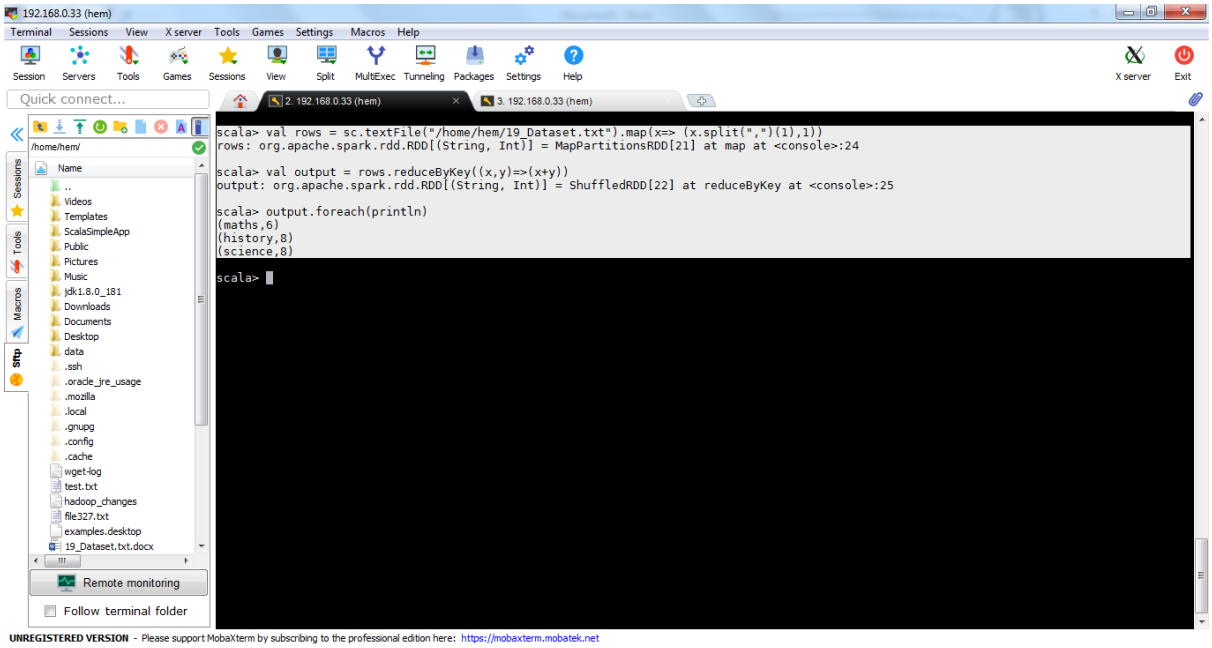


3. What is the distinct number of subjects present in the entire school

Spark Solution-:

```
scala> val rows = sc.textFile("/home/hem/19_Dataset.txt").map(x=> (x.split(",")(1),1))

rows: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[21] at map at <console>:24


scala> val output = rows.reduceByKey((x,y)=>(x+y))

output: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[22] at reduceByKey at <console>:25


scala> output.foreach(println)

(maths,6)

(history,8)

(science,8)
```

Output-:

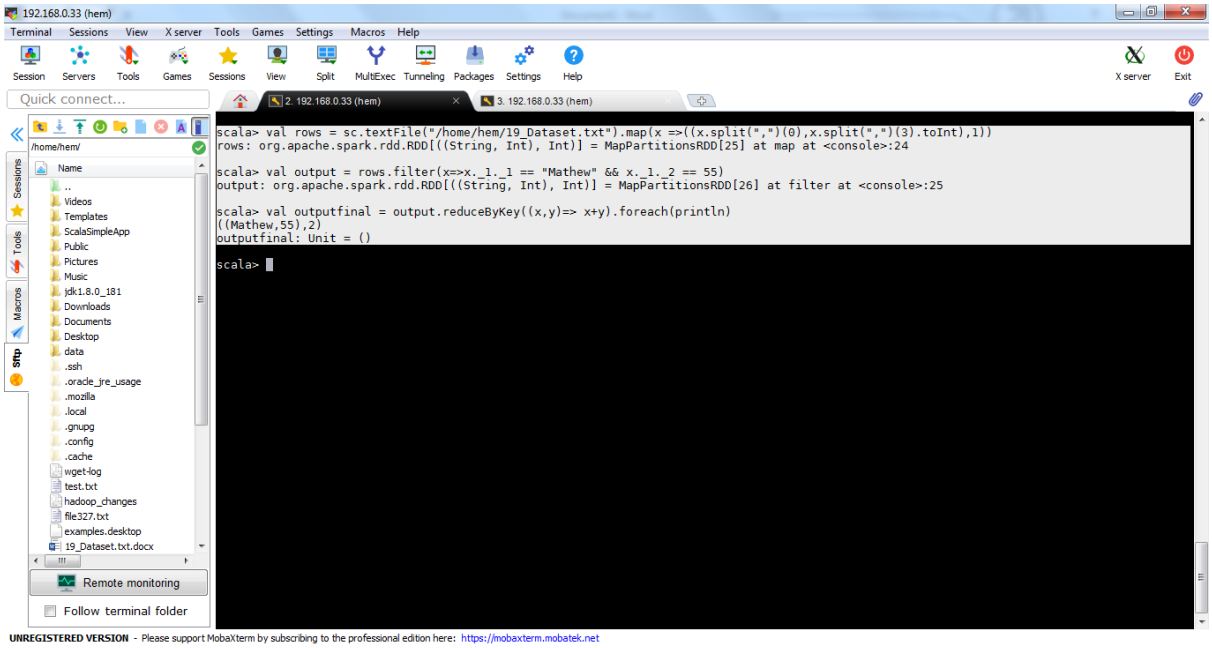


4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

Spark Solution-:

```
val rows = sc.textFile("/home/hem/19_Dataset.txt").map(x =>((x.split(",")(0),x.split(",")(3).toInt),1))

val output = rows.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)

val outputfinal = output.reduceByKey((x,y)=> x+y).foreach(println)
```

Output-:



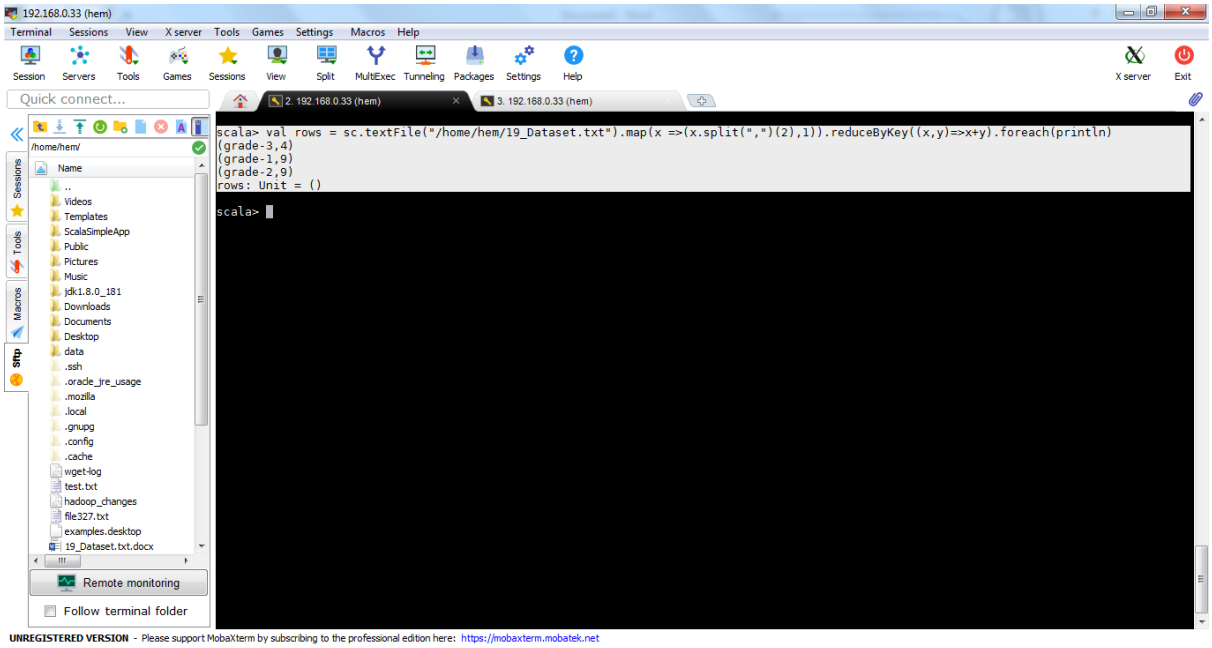
Problem Statement 2:

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

Spark Solution-:

val rows = sc.textFile("/home/hem/19_Dataset.txt").map(x =>(x.split(",")(2),1)).reduceByKey((x,y)=>x+y).foreach(println)

Output-:

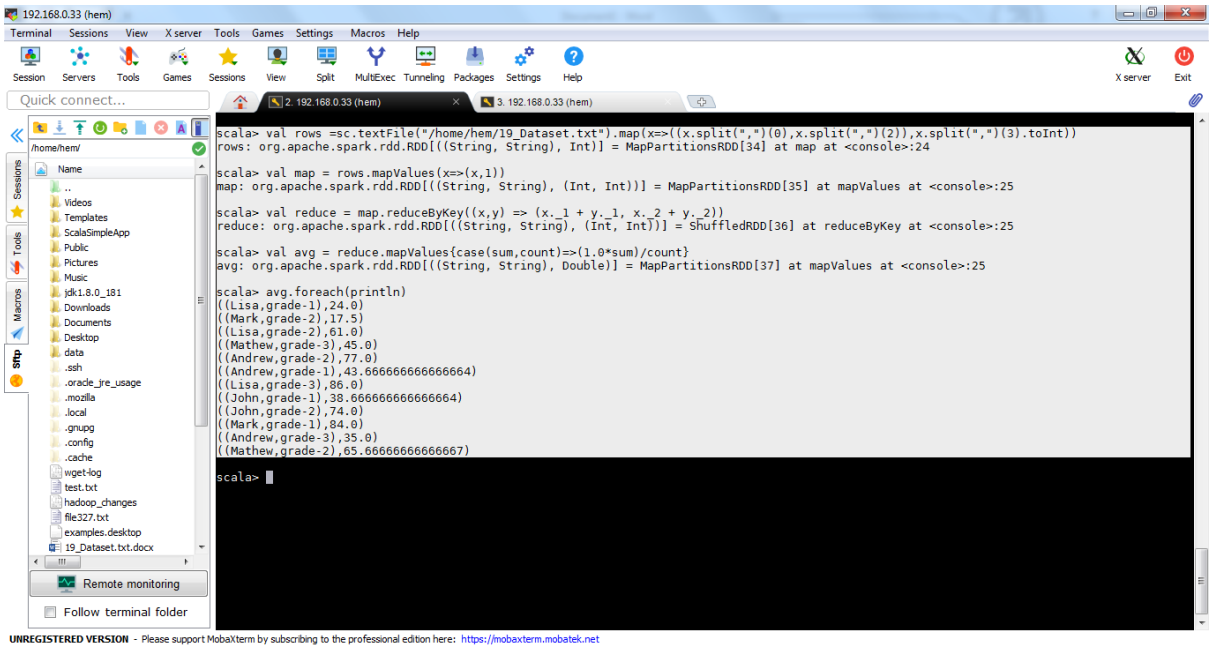


2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)

Spark Solution-:

```
val rows
=sc.textFile("/home/hem/19_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val map = rows.mapValues(x=>(x,1))

val reduce = map.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))

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

avg.foreach(println)
```

Output-:

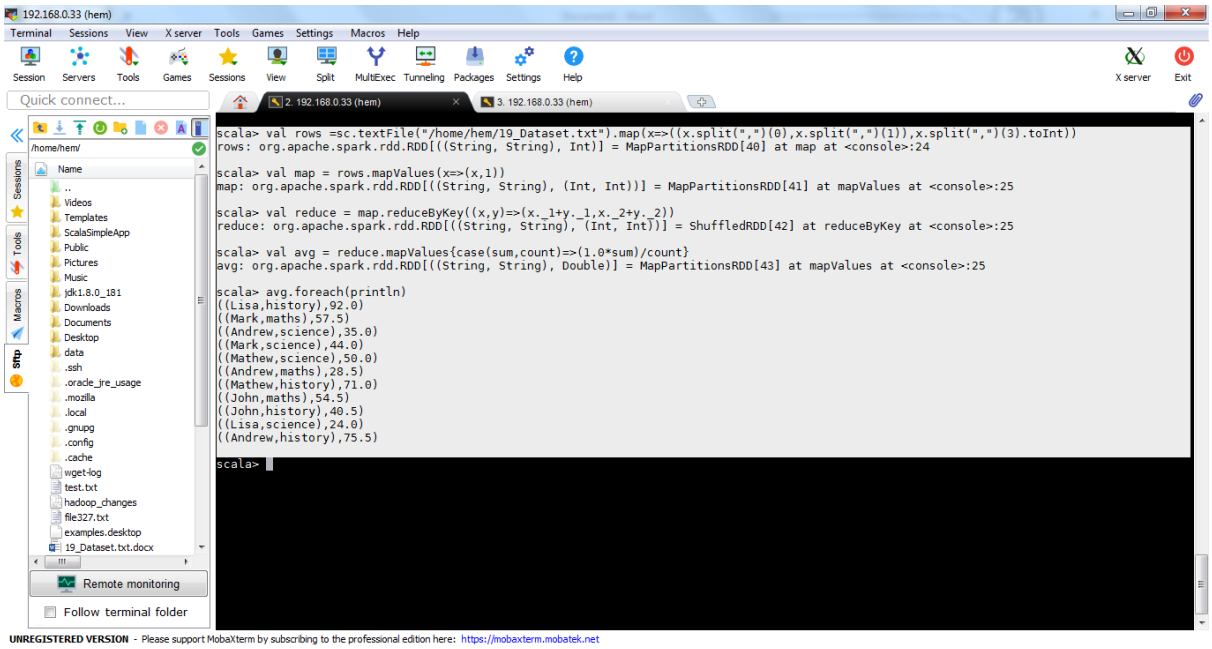


3. What is the average score of students in each subject across all grades?

Spark Solution-:

```
val rows
=sc.textFile("/home/hem/19_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))

val map = rows.mapValues(x=>(x,1))

val reduce = map.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))

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

avg.foreach(println)
```

Output-:

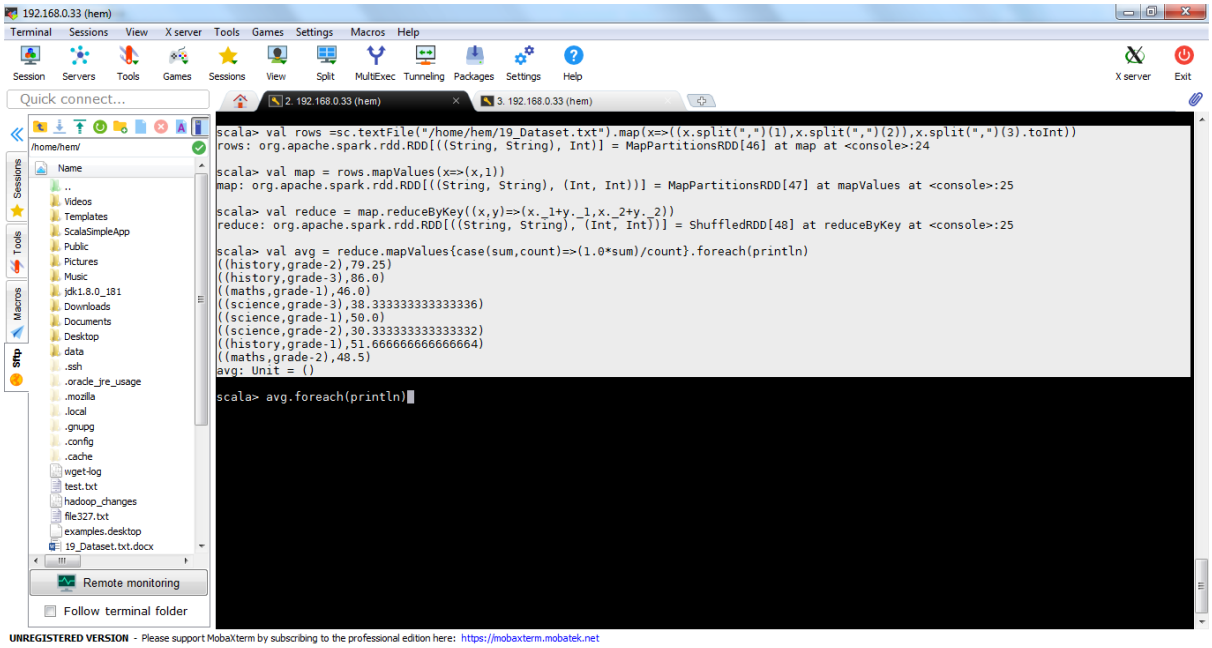


4. What is the average score of students in each subject per grade?

Spark Solution-:

```
val rows
=sc.textFile("/home/hem/19_Dataset.txt").map(x=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))

val map = rows.mapValues(x=>(x,1))

val reduce = map.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))

val avg = reduce.mapValues{case(sum,count)=>(1.0*sum)/count}.foreach(println)

avg.foreach(println)
```

Output-:



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

Spark Solution-:

val rows =sc.textFile("/home/hem/19_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val map = rows.mapValues(x=>(x,1))

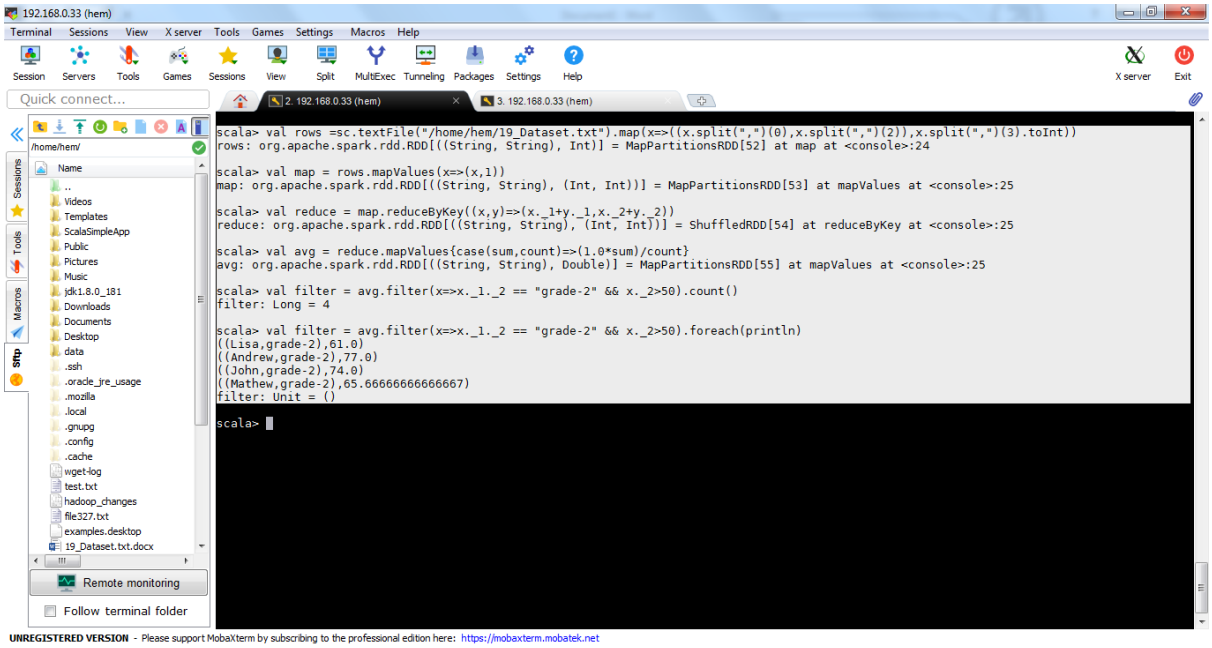
val reduce = map.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))

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

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

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

Output-:



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

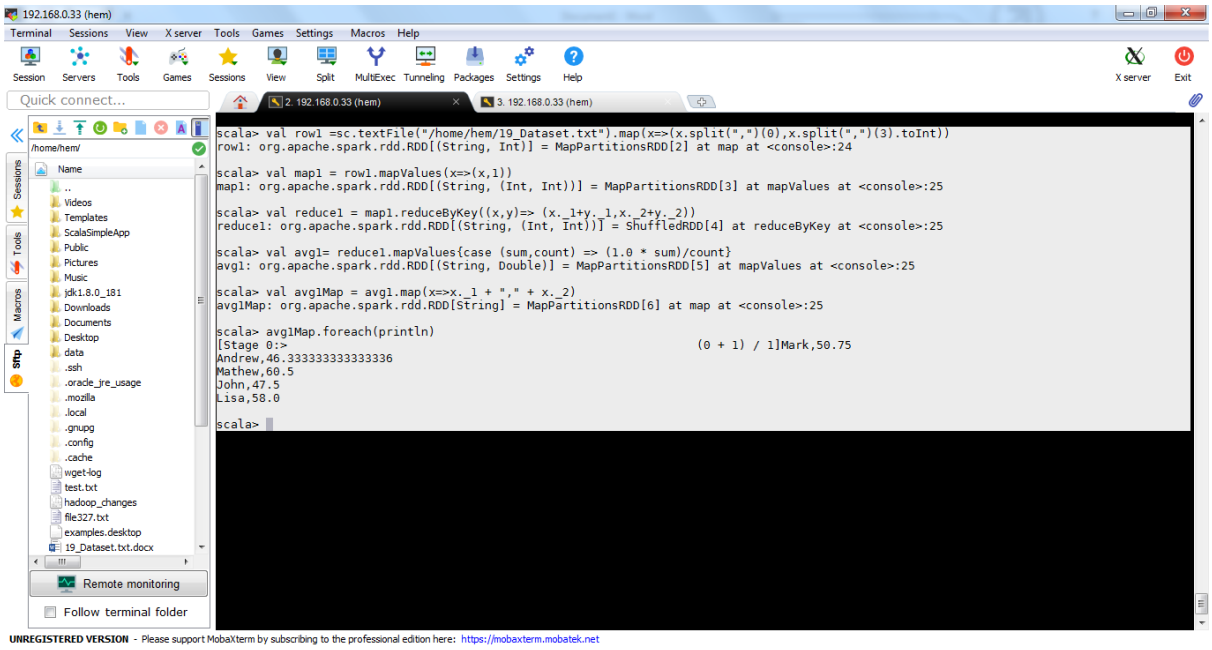
Hint - Use Intersection Property

Spark Solution-:

Avrage Score Per Student Name Across All Grades

```
val row1 =sc.textFile("/home/hem/19_Dataset.txt").map(x=>(x.split(",")(0),x.split(",")(3).toInt))

val map1 = row1.mapValues(x=>(x,1))

val reduce1 = map1.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))

val avg1= reduce1.mapValues{case (sum,count) => (1.0 * sum)/count}

val avg1Map = avg1.map(x=>x._1 + "," + x._2)

avg1Map.foreach(println)
```

Output-:

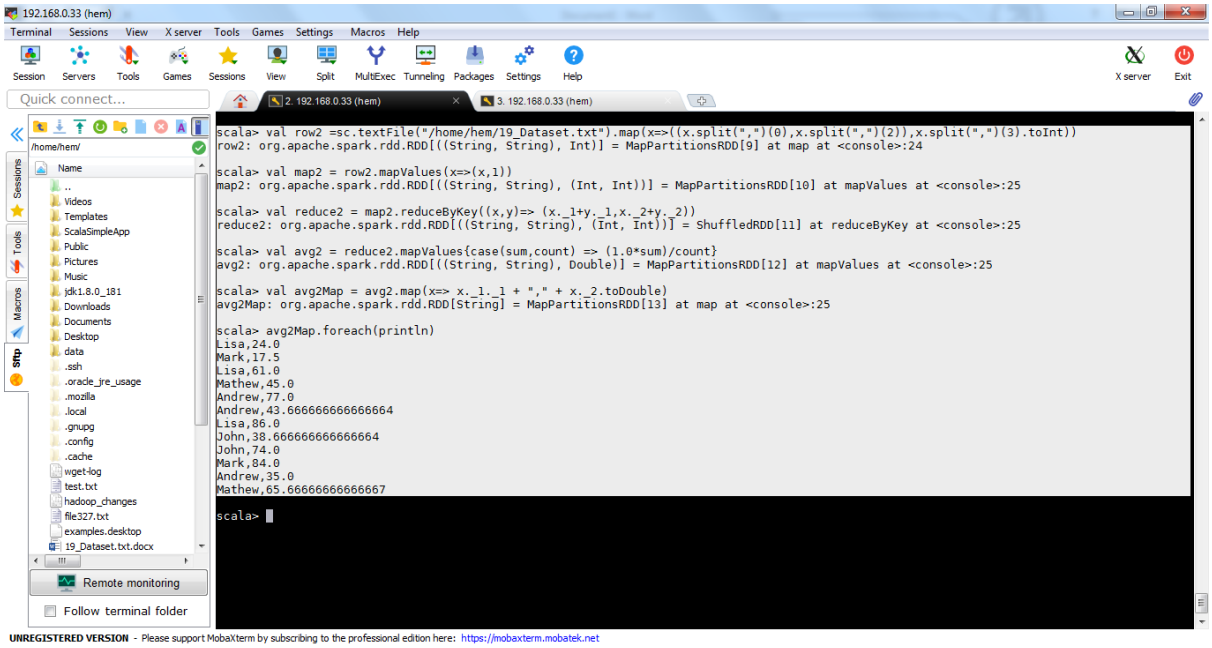


Average Score Per Student Name Per Grade

Spark Solution-:

```
val row2 =sc.textFile("/home/hem/19_Dataset.txt").map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))

val map2 = row2.mapValues(x=>(x,1))

val reduce2 = map2.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))

val avg2 = reduce2.mapValues{case(sum,count) => (1.0*sum)/count}

val avg2Map = avg2.map(x=> x._1._1 + "," + x._2.toDouble)

avg2Map.foreach(println)
```

Output-:



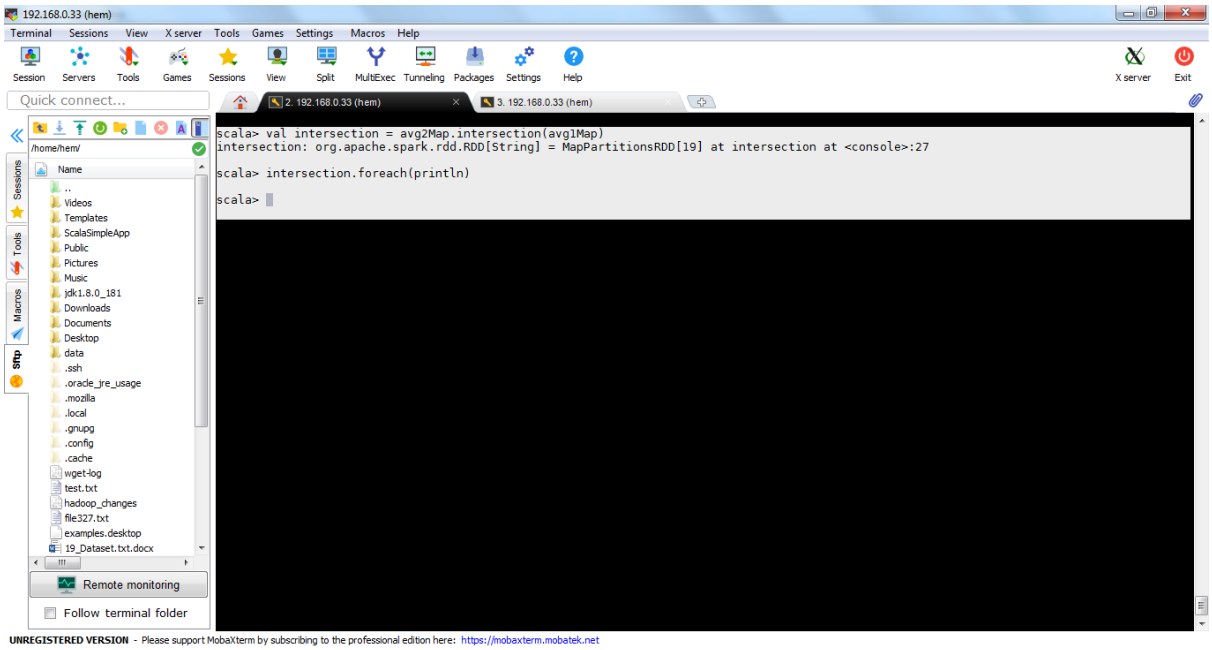
InterSection-:

Spark Solution-:

val intersection = avg2Map.intersection(avg1Map)

intersection.foreach(println)

Output-:



There is no output as nothing is common