Session 19:

RDD DEEP DIVE

Assignment 1

Task 1

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

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

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.

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
```

using scpark context sc reading input file at location C:/Users/mypc/Desktop/input.txt & printing each line

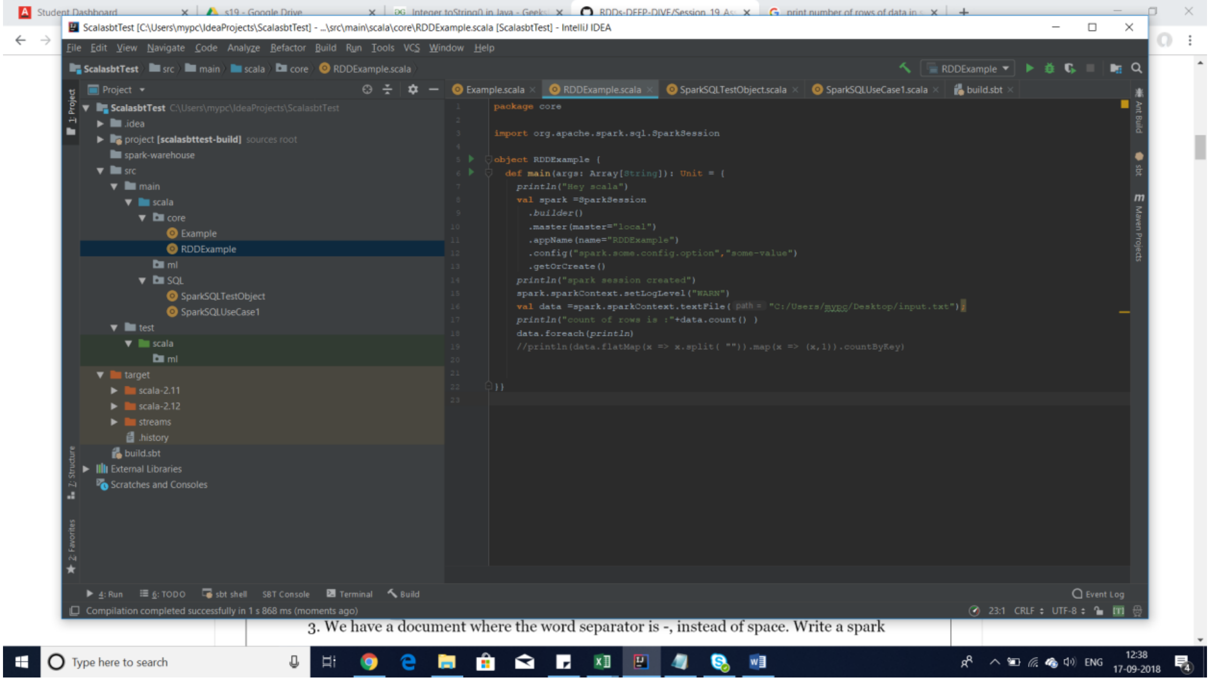
```
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/input.txt");
```

// **data.count**

(Counting the total number of lines within the file**)**

```
    println("count of rows is :"+data.count() )
    data.foreach(println)
```

```
//println(data.flatMap(x => x.split( "")).map(x => (x,1)).countByKey)



}}
```

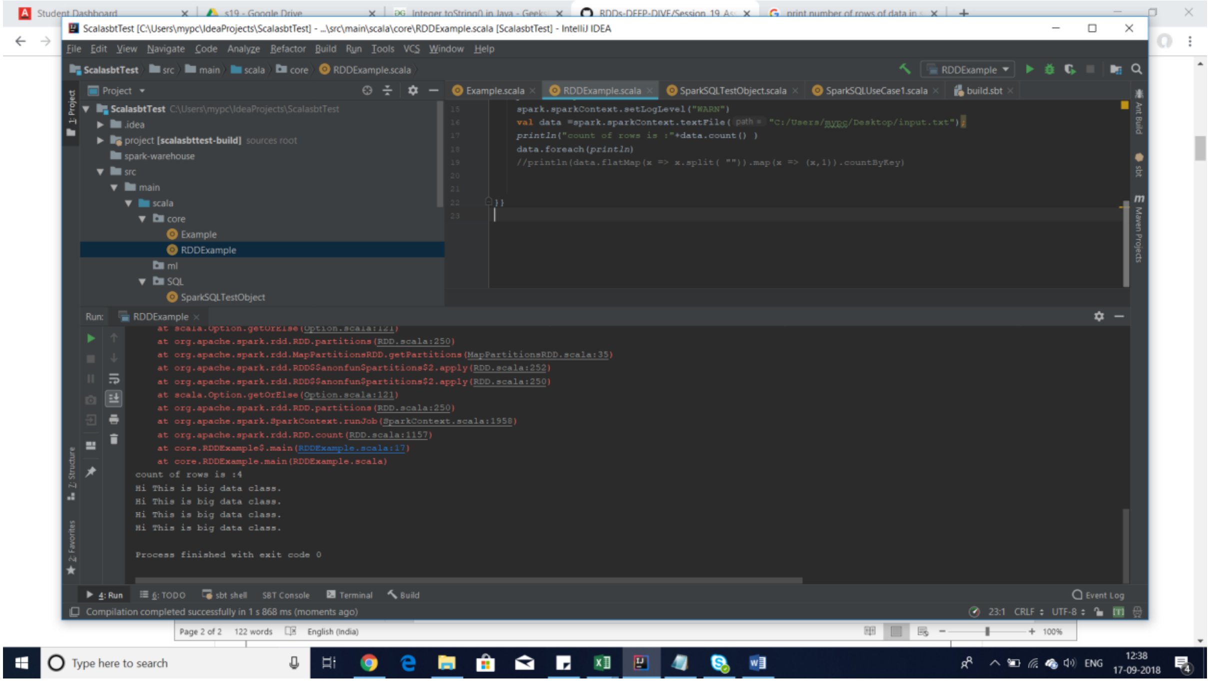


ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Example.scala
RDDExample.scala
SparkSQLTestObject.scala
SparkSQLUseCase1.scala
build.sbt

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/input.txt");
    println("count of rows is :"+data.count() )
    data.foreach(println)
    //println(data.flatMap(x => x.split( "")).map(x => (x,1)).countByKey)


}}
```

Compilation completed successfully in 1 s 868 ms (moments ago)

3. We have a document where the word separator is -, instead of space. Write a spark



ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA
Run: RDDExample

```
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/input.txt");
    println("count of rows is :"+data.count() )
    data.foreach(println)
    //println(data.flatMap(x => x.split( "")).map(x => (x,1)).countByKey)


}}
```

```
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
	at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
	at core.RDDExample$.main(RDDExample.scala:17)
	at core.RDDExample.main(RDDExample.scala)
count of rows is :4
Hi This is big data class.
Hi This is big data class.
Hi This is big data class.
Hi This is big data class.

Process finished with exit code 0
```

Compilation completed successfully in 1 s 868 ms (moments ago)

Page 2 of 2   122 words   English (India)

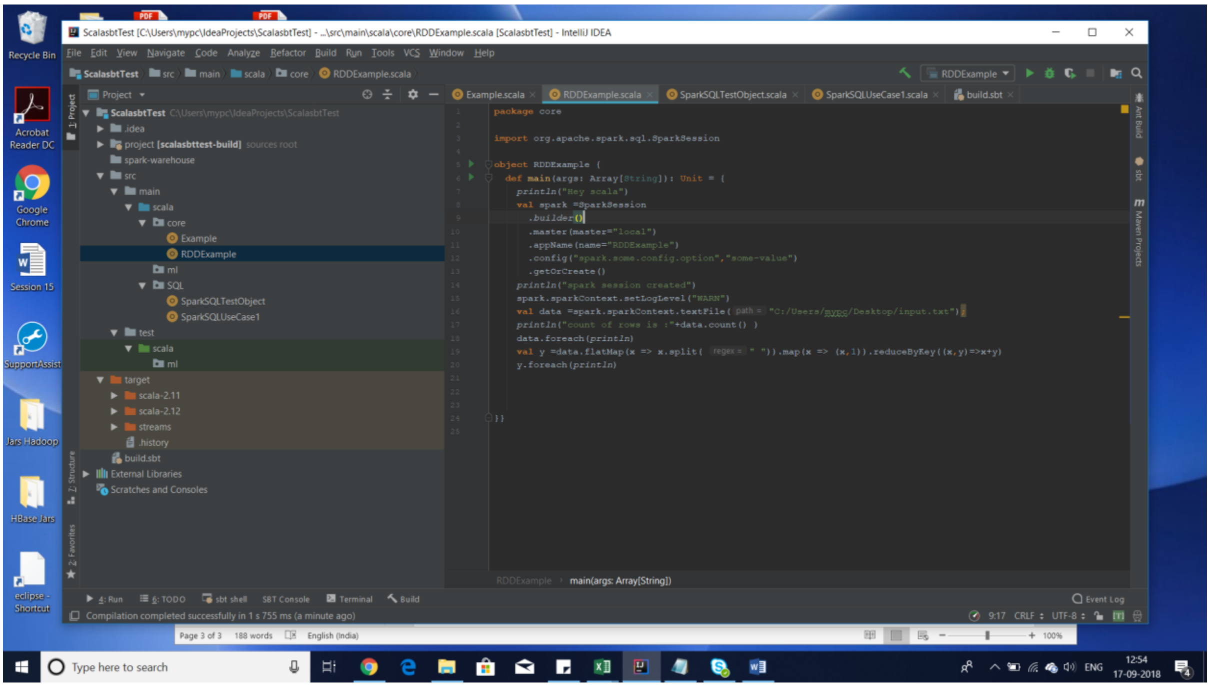
2. Write a program to read a text file and print the number of words in the document.

package core

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

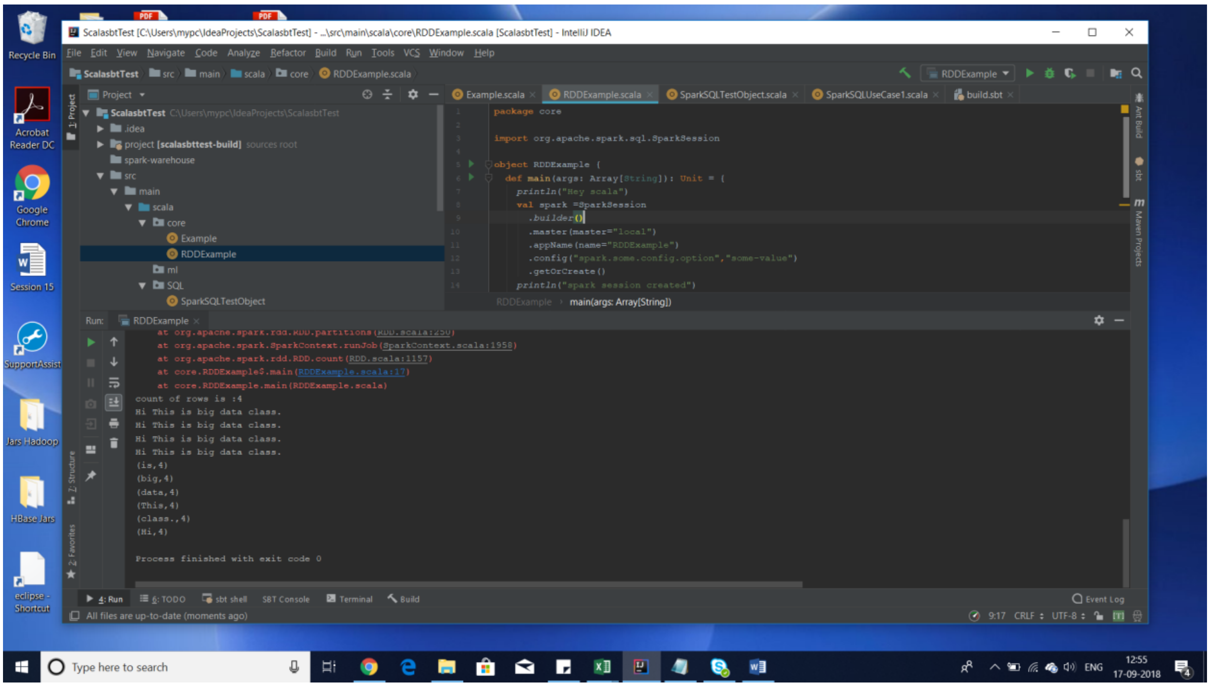
//creating RDD from a text file. Using split method on each line to split the line based on spaces. flatmap is used to map each line & then flatten those lines into isolated words. Using map again to associate an integer 1 with each word within file. reduceByKey treats all the words as key & adds up the ones for a particular word that is repeated

```
object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/input.txt");
    println("count of rows is :"+data.count() )
    data.foreach(println)
    val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    y.foreach(println)




}}
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/input.txt");
    println("count of rows is :"+data.count() )
    data.foreach(println)
    val y =data.flatMap(x => x.split( regex = " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    y.foreach(println)



}}
```

Compilation completed successfully in 1 s 755 ms (a minute ago)



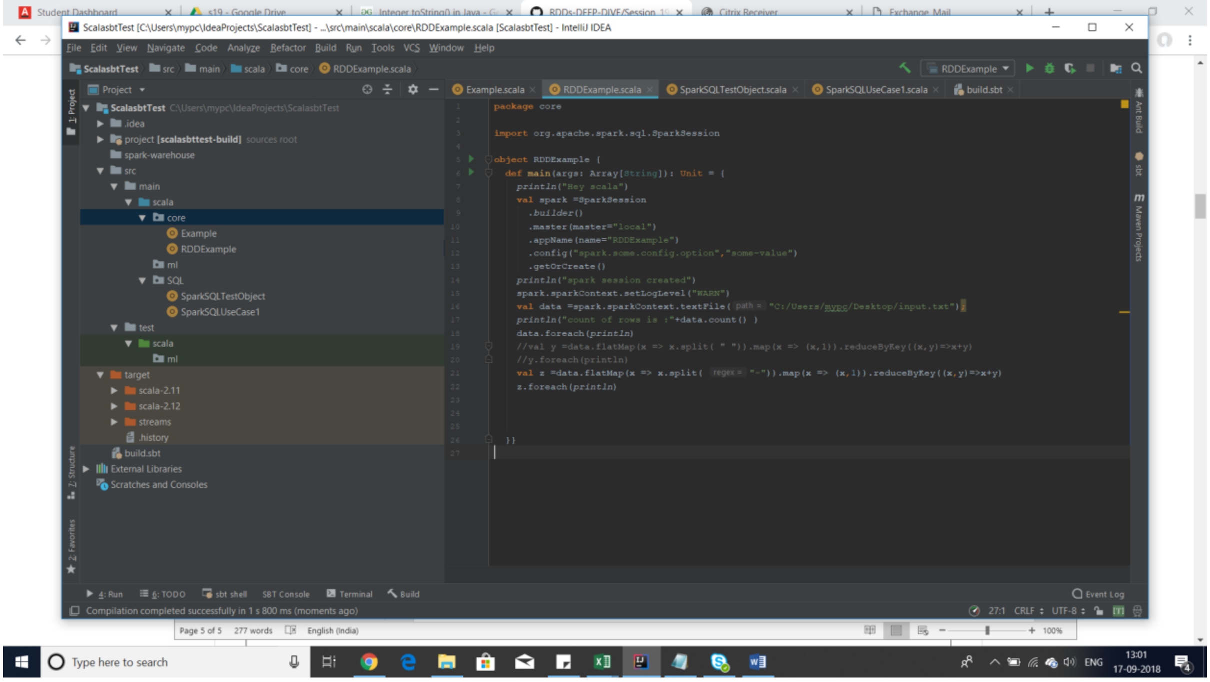
```
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
    at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
    at core.RDDExample$.main(RDDExample.scala:17)
    at core.RDDExample.main(RDDExample.scala)
count of rows is :4
Hi This is big data class.
Hi This is big data class.
Hi This is big data class.
Hi This is big data class.
(is,4)
(big,4)
(data,4)
(This,4)
(class.,4)
(Hi,4)

Process finished with exit code 0
```

All files are up-to-date (moments ago)

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.

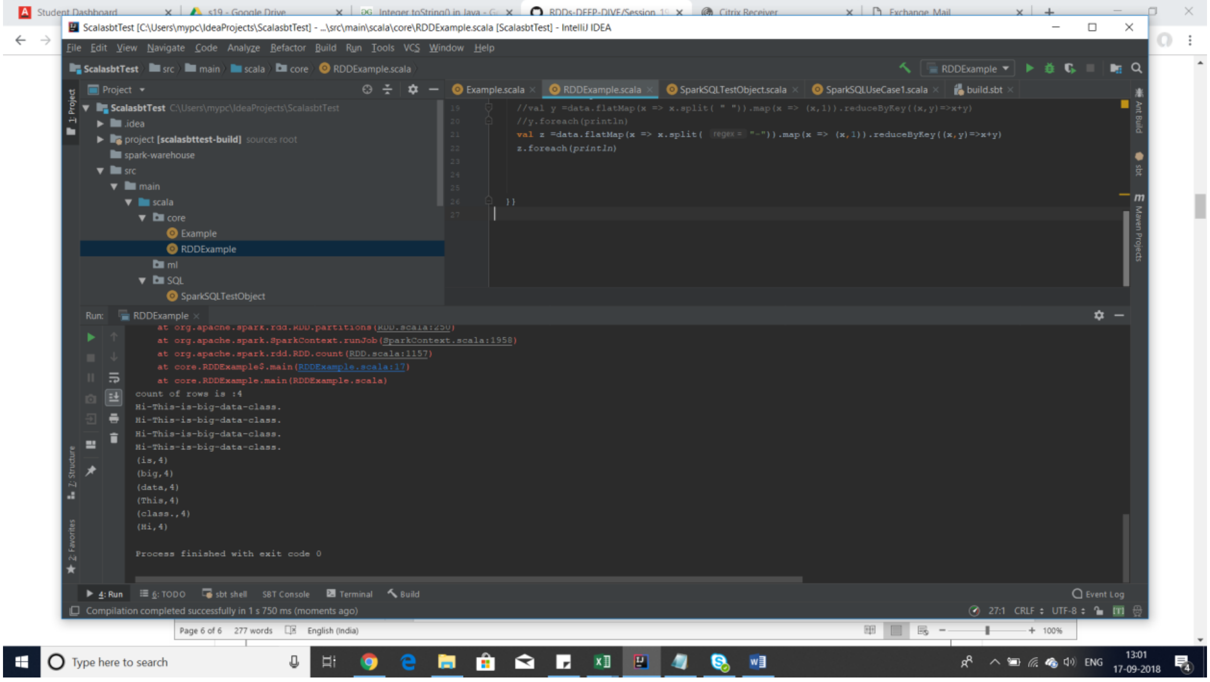
//Explanation: creating RDD from a text file. Using split method on each line to split the line based on hyphen(-). flatmap is used to map each line & then flatten those lines into isolated words. Using map again to associate an integer 1 with each word within file. reduceByKey treats all the words as key & adds up the ones for a word that is repeated

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/input.txt");
    println("count of rows is :"+data.count() )
    data.foreach(println)
    //val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    //y.foreach(println)
    val z =data.flatMap(x => x.split( "-")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    z.foreach(println)



}}
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/input.txt");
    println("count of rows is :"+data.count() )
    data.foreach(println)
    //val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    //y.foreach(println)
    val z =data.flatMap(x => x.split("-")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    z.foreach(println)



  }}
```

Compilation completed successfully in 1 s 800 ms (moments ago)



```
    //val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    //y.foreach(println)
    val z =data.flatMap(x => x.split("-")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    z.foreach(println)



  }}
```

```
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
	at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
	at core.RDDExample$.main(RDDExample.scala:17)
	at core.RDDExample.main(RDDExample.scala)
count of rows is :4
Hi-This-is-big-data-class.
Hi-This-is-big-data-class.
Hi-This-is-big-data-class.
Hi-This-is-big-data-class.
(is,4)
(big,4)
(data,4)
(This,4)
(class.,4)
(Hi,4)

Process finished with exit code 0
```

Compilation completed successfully in 1 s 750 ms (moments ago)

Task 2

Problem Statement 1:

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

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

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

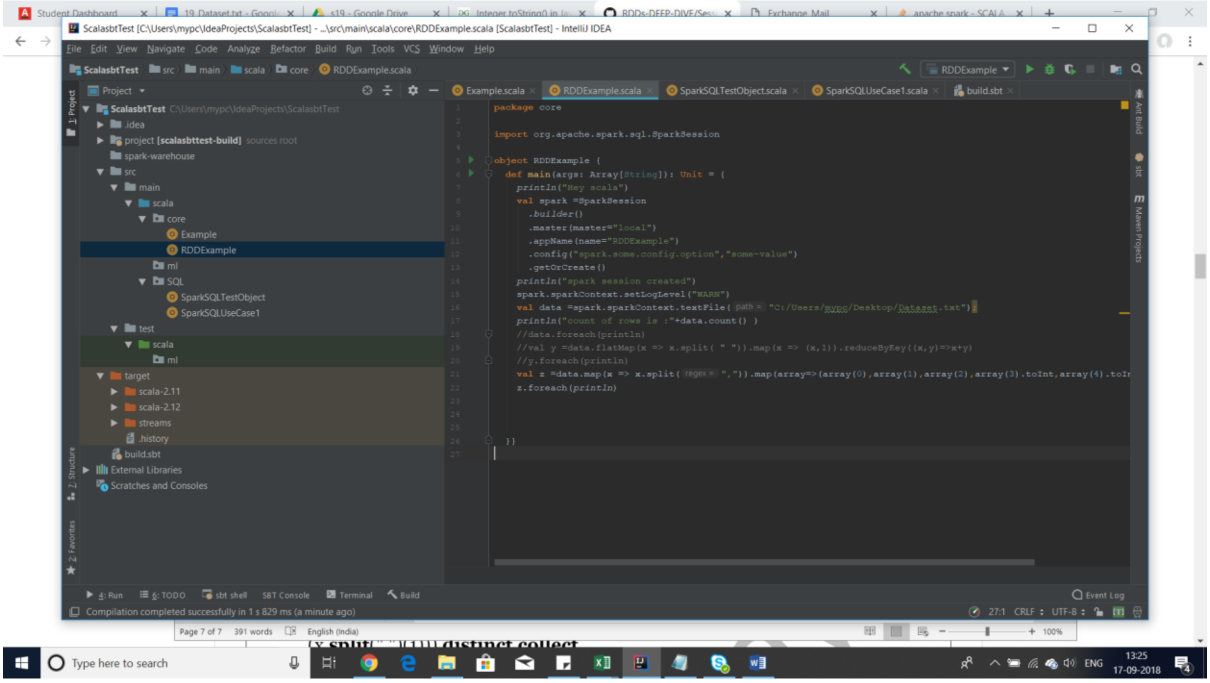
4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

Solution:

Explanation:

// creating tuple RDD from a text file. Using split method on each line to split the line based on “,”. Using map again to create a tuple of each line from the file. toInt is used to convert the numeric columns from String to Integer

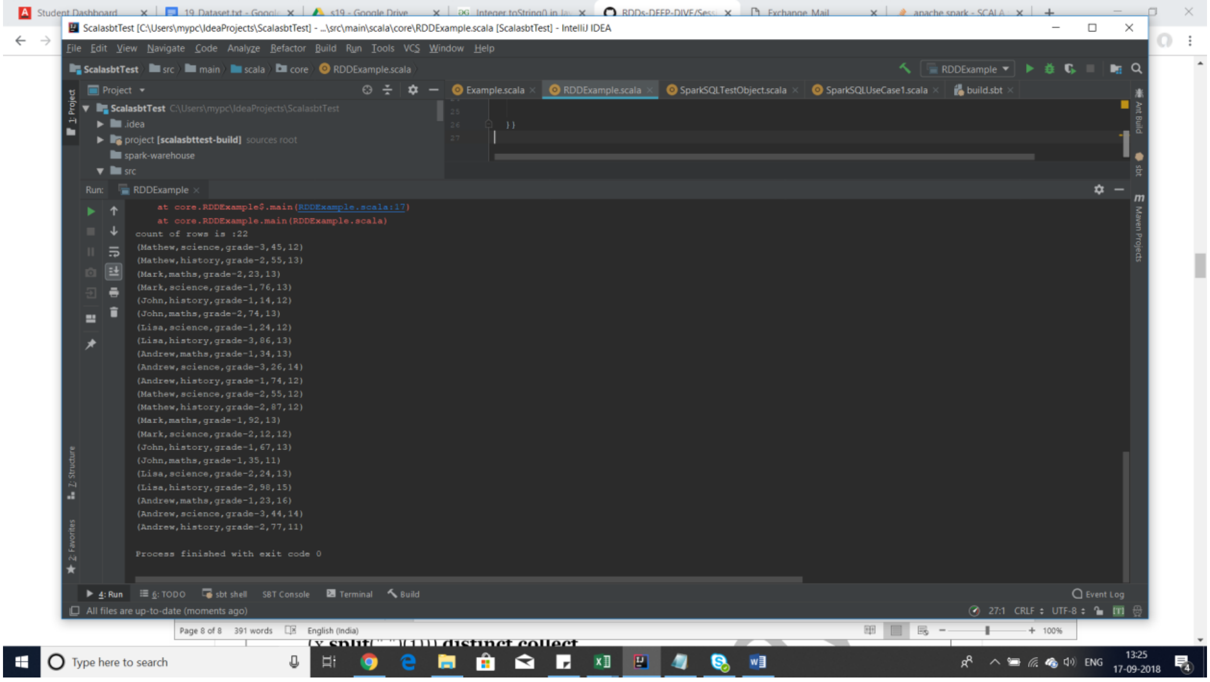
```
package core


import org.apache.spark.sql.SparkSession


object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )
    //data.foreach(println)
    //val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    //y.foreach(println)
    val z =data.map(x =>
x.split(",")).map(array=>(array(0),array(1),array(2),array(3).toInt,array(4).toInt)).collect
    z.foreach(println)



  }}
```



ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )
    //data.foreach(println)
    //val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    //y.foreach(println)
    val z =data.map(x => x.split(",")).map(array=>(array(0),array(1),array(2),array(3).toInt,array(4).toI
    z.foreach(println)



  }}
```

Compilation completed successfully in 1 s 829 ms (a minute ago)



```
    at core.RDDExample$.main(RDDExample.scala:17)
    at core.RDDExample.main(RDDExample.scala)
count of rows is :22
(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)

Process finished with exit code 0
```

All files are up-to-date (moments ago)

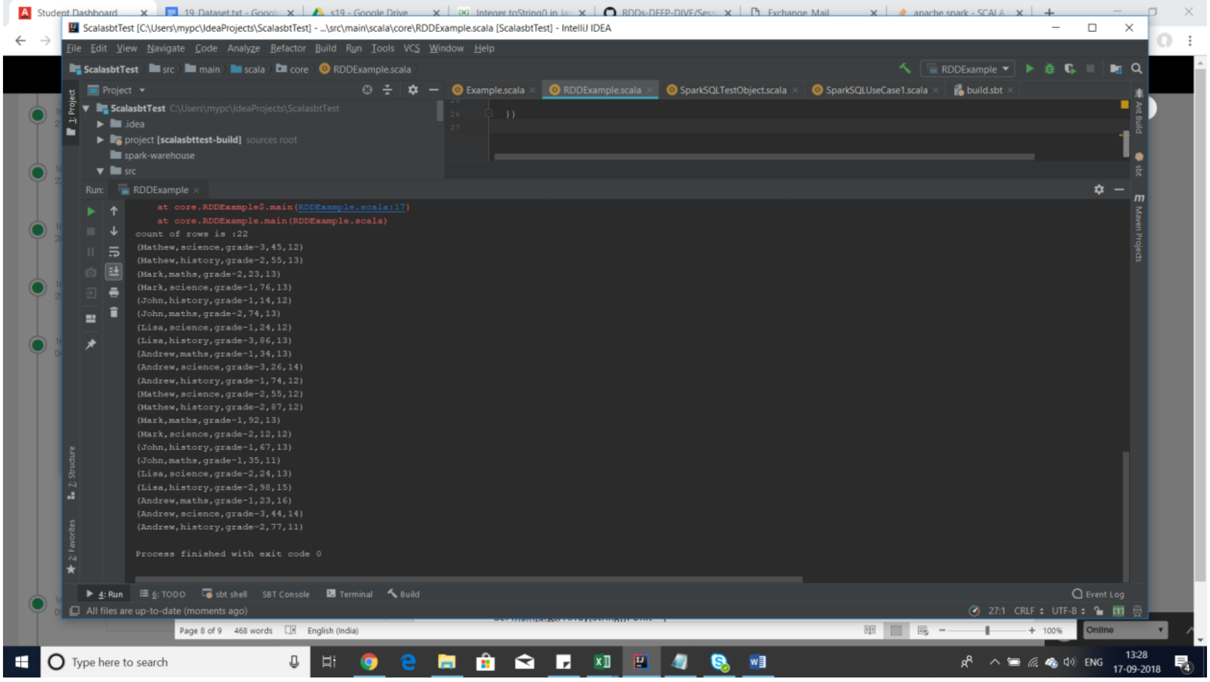
2. Find the count of total number of rows present.

Solution:We can find the count of rows using

“println("count of rows is :"+data.count())” in the program

object RDDExample {

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

  println("Hey scala")

  val spark =SparkSession

    .builder()

    .master(master="local")

    .appName(name="RDDExample")

    .config("spark.some.config.option","some-value")

    .getOrCreate()

  println("spark session created")

  spark.sparkContext.setLogLevel("WARN")

  val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");

  println("count of rows is :"+data.count() )

  val z =data.map(x =>
x.split(",")).map(array=>(array(0),array(1),array(2),array(3).toInt,array(4).toInt)).collect

  z.foreach(println)




 }}
```



```
      at core.RDDExample$.main(RDDExample.scala:17)
      at core.RDDExample.main(RDDExample.scala)
count of rows is :22
(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)

Process finished with exit code 0
```

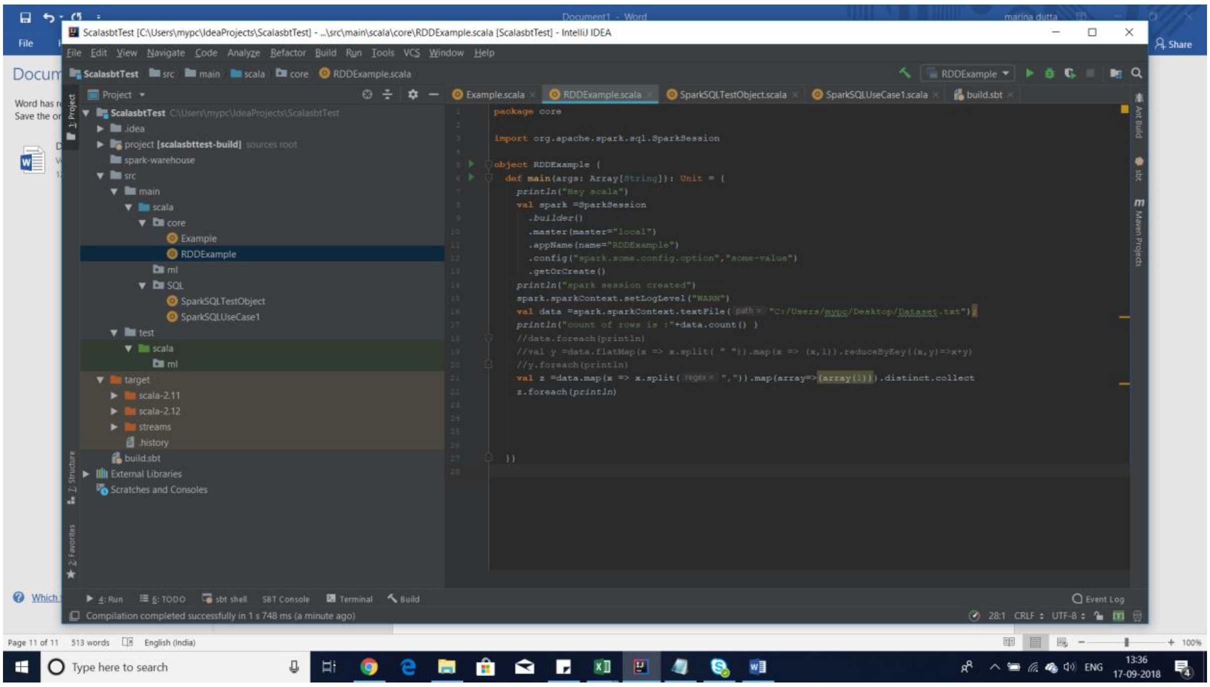
ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ScalasbtTest src main scala core RDDExample.scala
Project
ScalasbtTest C:\Users\mypc\IdeaProjects\ScalasbtTest
.idea
project [scalasbttest-build] sources root
spark-warehouse
src
Example.scala
RDDExample.scala
SparkSQLTestObject.scala
SparkSQLUseCase1.scala
build.sbt
Run: RDDExample
4: Run
6: TODO
sbt shell
SBT Console
Terminal
Build
Event Log
All files are up-to-date (moments ago)
Page 8 of 9 468 words English (India)
Type here to search
13:28
17-09-2018

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

Solution:

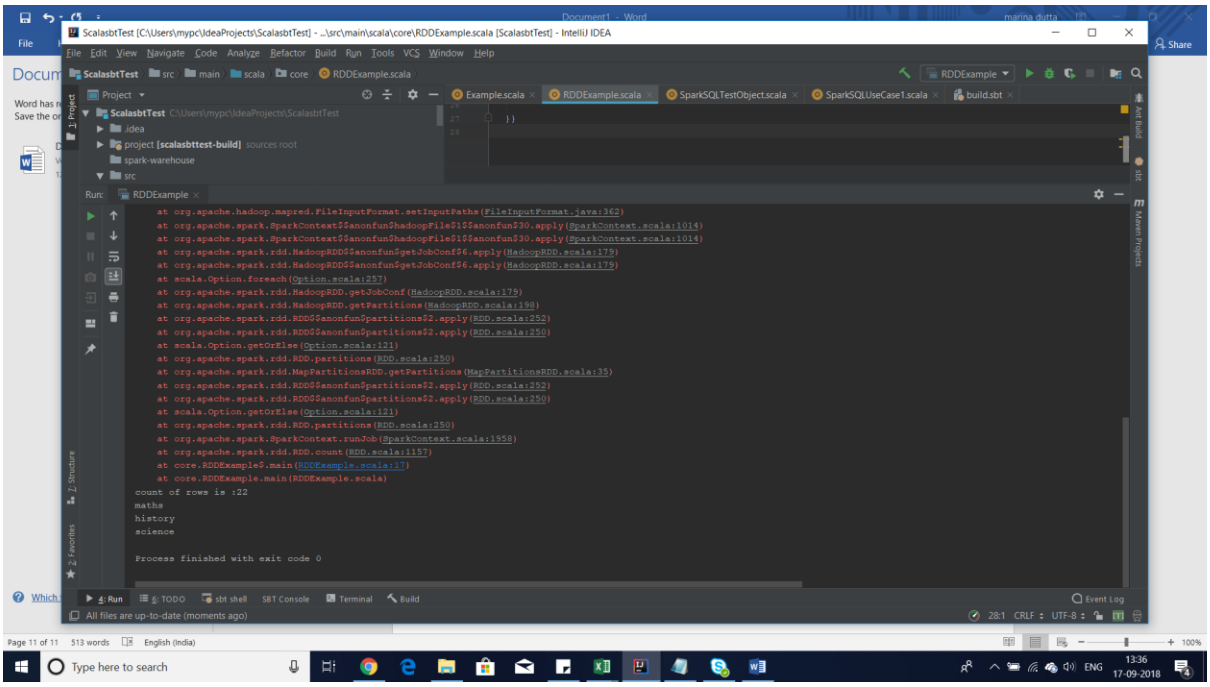
Explanation: Creating a n RDD by splitting an input dataset based on “,” & selecting the subject field at index position 1. using distinct method over field at index position 1 to find the unique values within that column. Displaying them by calling the action collect.

```
package core


import org.apache.spark.sql.SparkSession


object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )
    val z =data.map(x => x.split(",")).map(array=>(array(1))).distinct.collect
    z.foreach(println)



  }}
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :"+data.count() )
    //data.foreach(println)
    //val y =data.flatMap(x => x.split( " ")).map(x => (x,1)).reduceByKey((x,y)=>x+y)
    //y.foreach(println)
    val z =data.map(x => x.split(",")).map(array=>(array(1))).distinct.collect
    z.foreach(println)




  }}
```

Compilation completed successfully in 1 s 748 ms (a minute ago)



```
	at org.apache.hadoop.mapred.FileInputFormat.setInputPaths(FileInputFormat.java:362)
	at org.apache.spark.SparkContext$$anonfun$hadoopFile$1$$anonfun$30.apply(SparkContext.scala:1014)
	at org.apache.spark.SparkContext$$anonfun$hadoopFile$1$$anonfun$30.apply(SparkContext.scala:1014)
	at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
	at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
	at scala.Option.foreach(Option.scala:257)
	at org.apache.spark.rdd.HadoopRDD.getJobConf(HadoopRDD.scala:179)
	at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:198)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
	at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
	at core.RDDExample$.main(RDDExample.scala:17)
	at core.RDDExample.main(RDDExample.scala)
count of rows is :22
maths
history
science

Process finished with exit code 0
```

All files are up-to-date (moments ago)

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

Solution:

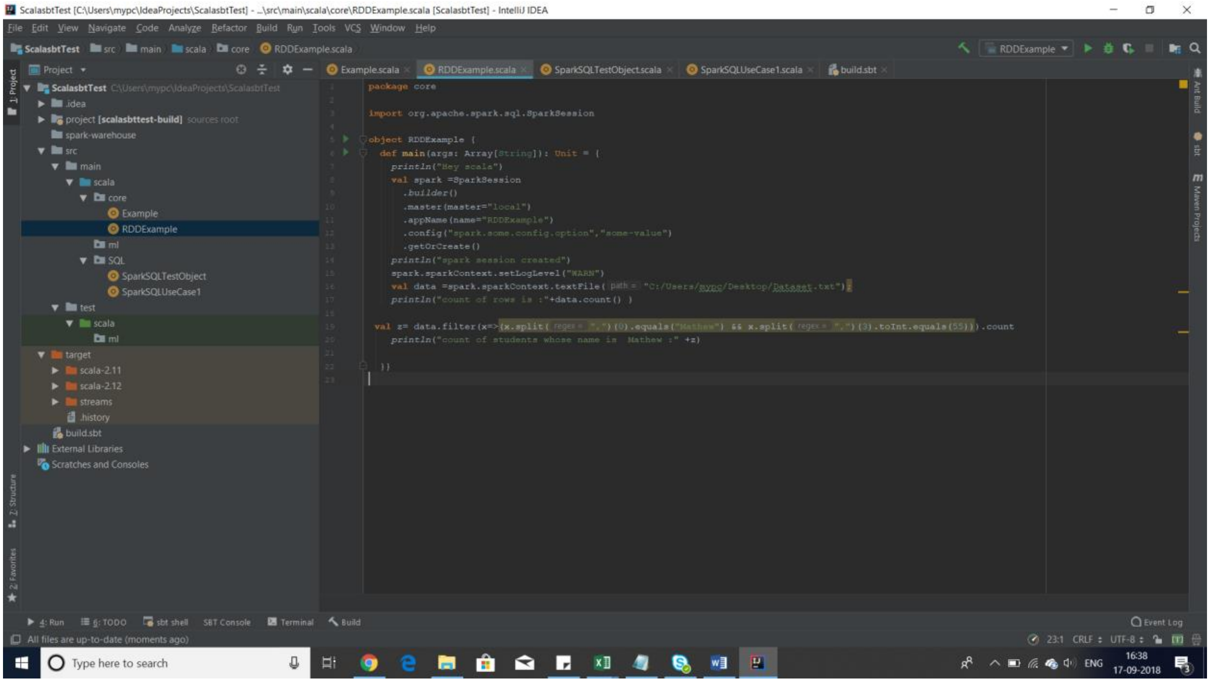
Explanation:

// Using the filter function over the RDD to select all those first fields whose name is “Mathew” using the .equals() function to match the given

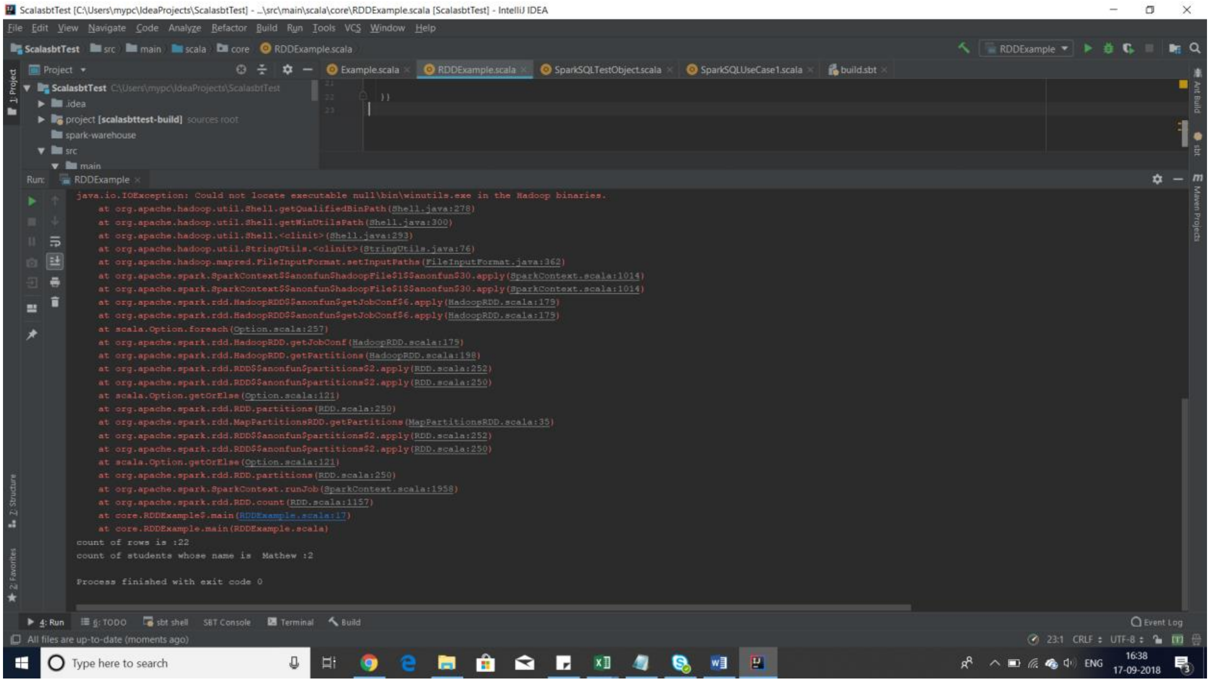
condition. filter returns all the matching columns when condition becomes True. Using logical && operator to

find all such fields where marks = 55. Using count method to to count all such lines that holds true for both the conditions

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )

 val z= data.filter(x=>(x.split(",")(0).equals("Mathew") && x.split(",")(3).toInt.equals(55))).count
    println("count of students whose name is  Mathew :" +z)

  }}
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )

  val z= data.filter(x=>(x.split( regex = ",")(0).equals("Mathew") && x.split( regex = ",")(3).toInt.equals(55))).count
    println("count of students whose name is  Mathew :" +z)

  }}
```



```
java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.
	at org.apache.hadoop.util.Shell.getQualifiedBinPath(Shell.java:278)
	at org.apache.hadoop.util.Shell.getWinUtilsPath(Shell.java:300)
	at org.apache.hadoop.util.Shell.<clinit>(Shell.java:293)
	at org.apache.hadoop.util.StringUtils.<clinit>(StringUtils.java:76)
	at org.apache.hadoop.mapred.FileInputFormat.setInputPaths(FileInputFormat.java:362)
	at org.apache.spark.SparkContext$$anonfun$hadoopFile$1$$anonfun$30.apply(SparkContext.scala:1014)
	at org.apache.spark.SparkContext$$anonfun$hadoopFile$1$$anonfun$30.apply(SparkContext.scala:1014)
	at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
	at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
	at scala.Option.foreach(Option.scala:257)
	at org.apache.spark.rdd.HadoopRDD.getJobConf(HadoopRDD.scala:179)
	at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:198)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
	at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
	at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
	at core.RDDExample$.main(RDDExample.scala:17)
	at core.RDDExample.main(RDDExample.scala)
count of rows is :22
count of students whose name is  Mathew :2

Process finished with exit code 0
```

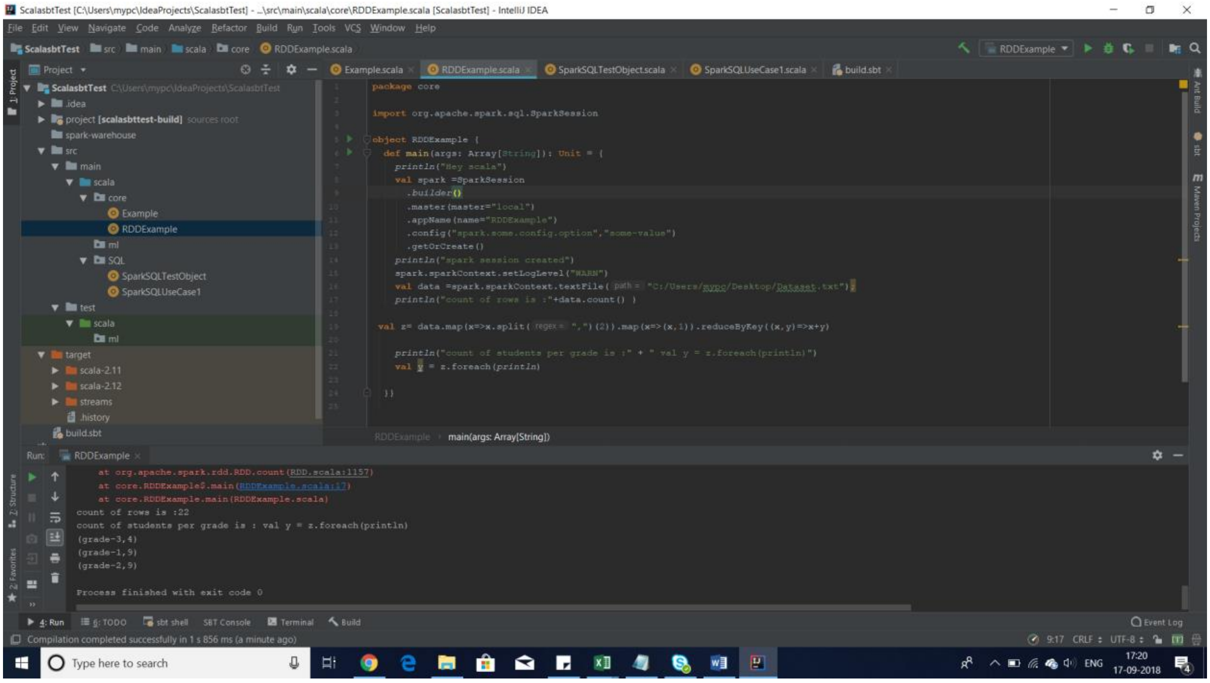
Problem Statement 2:

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

    import org.apache.spark.sql.SparkSession

// **Explanation:** Creating an RDD & selecting the field grade at the 2nd index i.e. 3rd position. Assigning 1 to each grade & thereby performing reduceByKey to count the value of each grade.

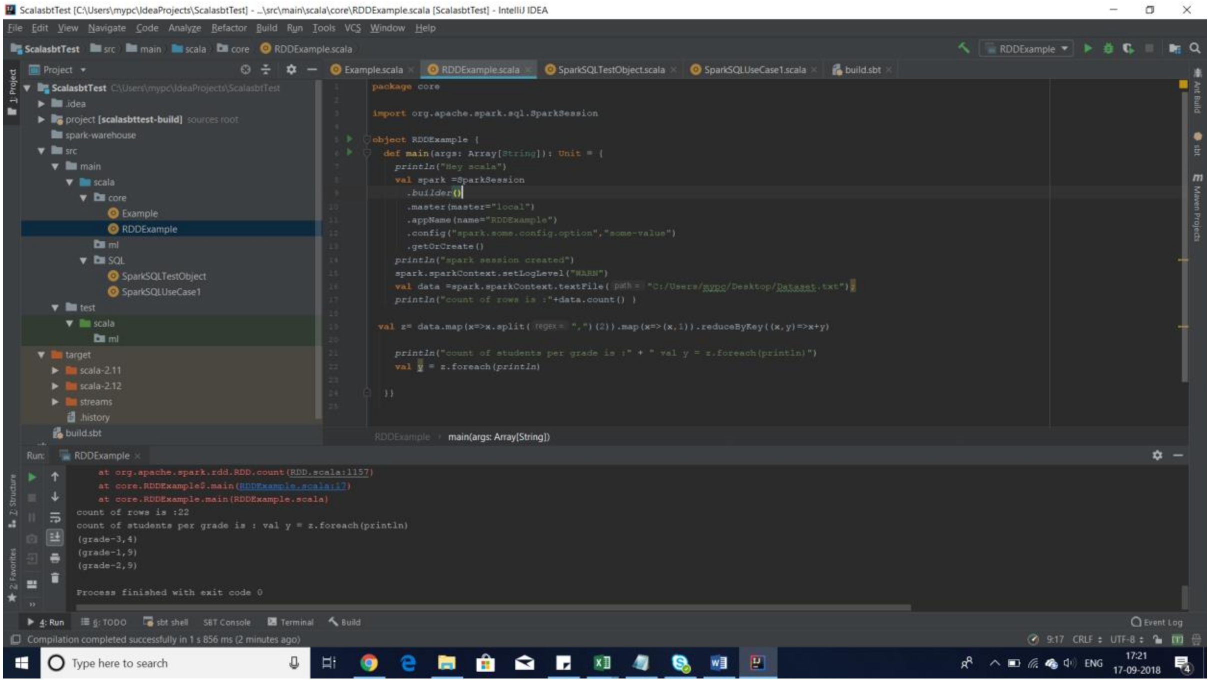
```
object RDDExample {
 def main(args: Array[String]): Unit = {
  println("Hey scala")
  val spark =SparkSession
   .builder()
   .master(master="local")
   .appName(name="RDDExample")
   .config("spark.some.config.option","some-value")
   .getOrCreate()
  println("spark session created")
  spark.sparkContext.setLogLevel("WARN")
  val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");
  println("count of rows is :"+data.count() )

 val z= data.map(x=>x.split(",")(2)).map(x=>(x,1)).reduceByKey((x,y)=>x+y)

  println("count of students per grade is :" + " val y = z.foreach(println)")
  val y = z.foreach(println)

 }}
```



ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ScalasbtTest src main scala core RDDExample.scala
Project
Example.scala RDDExample.scala SparkSQLTestObject.scala SparkSQLUseCase1.scala build.sbt
ScalasbtTest C:\Users\mypc\IdeaProjects\ScalasbtTest
.idea
project [scalasbttest-build] sources root
spark-warehouse
src
main
scala
core
Example
RDDExample
ml
SQL
SparkSQLTestObject
SparkSQLUseCase1
test
scala
ml
target
scala-2.11
scala-2.12
streams
.history
build.sbt
RDDExample main(args: Array[String])
Run: RDDExample
4: Run 6: TODO sbt shell SBT Console Terminal Build
Event Log
Compilation completed successfully in 1 s 856 ms (a minute ago)
Type here to search

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )

  val z= data.map(x=>x.split( regex = ",")(2)).map(x=>(x,1)).reduceByKey((x,y)=>x+y)

    println("count of students per grade is :" + " val y = z.foreach(println)")
    val y = z.foreach(println)

  }}
```

```
    at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
    at core.RDDExample$.main(RDDExample.scala:17)
    at core.RDDExample.main(RDDExample.scala)
count of rows is :22
count of students per grade is : val y = z.foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)

Process finished with exit code 0
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark =SparkSession
      .builder()
      .master(master="local")
      .appName(name="RDDExample")
      .config("spark.some.config.option","some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data =spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt");
    println("count of rows is :"+data.count() )

  val z= data.map(x=>x.split(",")(2)).map(x=>(x,1)).reduceByKey((x,y)=>x+y)

    println("count of students per grade is :" + " val y = z.foreach(println)")
    val y = z.foreach(println)

  }}
```

```
	at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
	at core.RDDExample$.main(RDDExample.scala:17)
	at core.RDDExample.main(RDDExample.scala)
count of rows is :22
count of students per grade is : val y = z.foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)

Process finished with exit code 0
```

Compilation completed successfully in 1 s 856 ms (2 minutes ago)

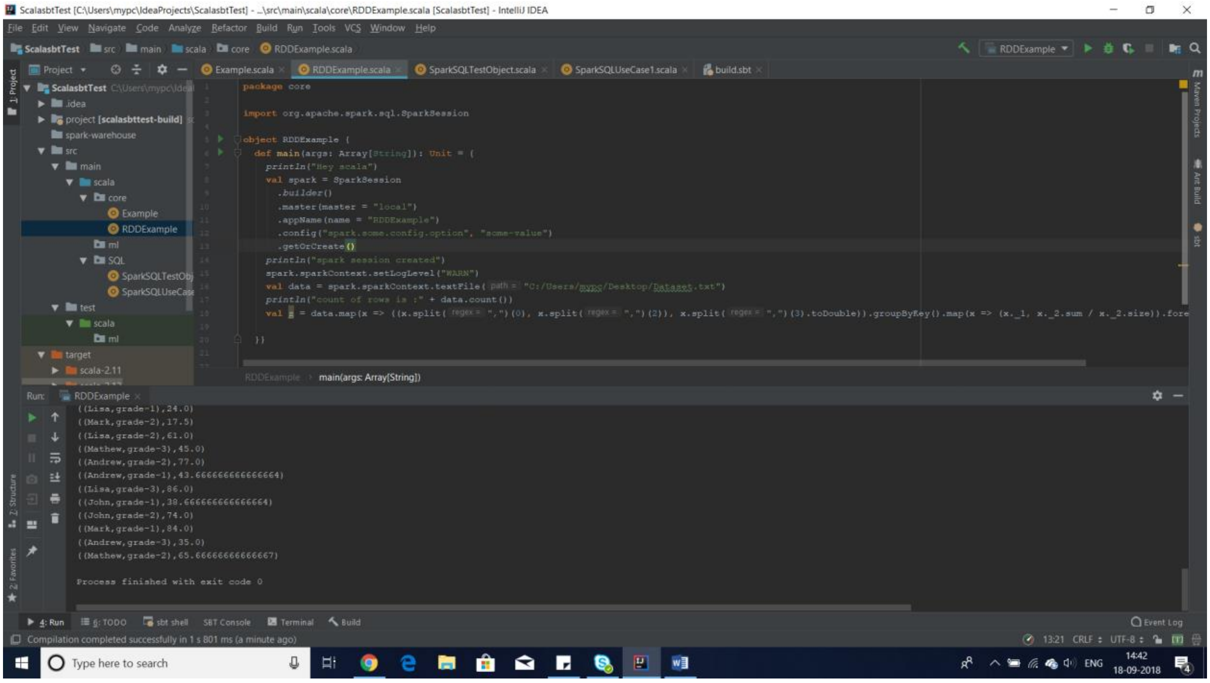
2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)

Explanation:

Creating an RDD from textFile. Selecting name & grade as key, marks as value. Converting marks into double datatype just to avoid any truncation after decimal. groupByKey returns a tuple of ((name, grade), marks). Applying map function & selecting name, grade as first value of outer tuple & marks as second value. Adding the marks for respective grade & for each distinct grade fetching the count of marks as x._2.size to divide the sum & find the average on it. Here grouping key is student & grade.

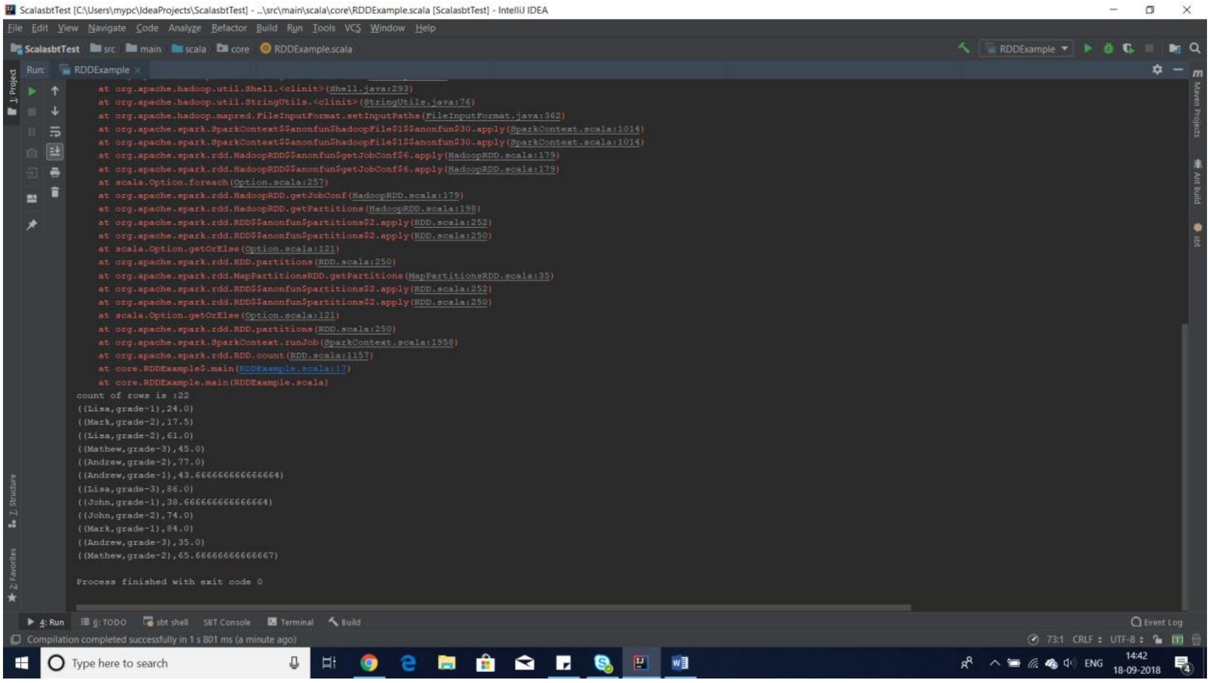
```
import org.apache.spark.sql.SparkSession


object RDDExample {

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

  println("Hey scala")

  val spark = SparkSession

   .builder()
```

```
      .master(master = "local")

      .appName(name = "RDDExample")

      .config("spark.some.config.option", "some-value")

      .getOrCreate()

    println("spark session created")

    spark.sparkContext.setLogLevel("WARN")

    val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")

    println("count of rows is :" + data.count())

    val z = data.map(x => ((x.split(",")(0), x.split(",")(2)), x.split(",")(3).toDouble)).groupByKey().map(x =>
(x._1, x._2.sum / x._2.size)).foreach(println)


  }}
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z = data.map(x => ((x.split( regex = ",")(0), x.split( regex = ",")(2)), x.split( regex = ",")(3).toDouble)).groupByKey().map(x => (x._1, x._2.sum / x._2.size)).fore

  }}
```

```
((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),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)

Process finished with exit code 0
```



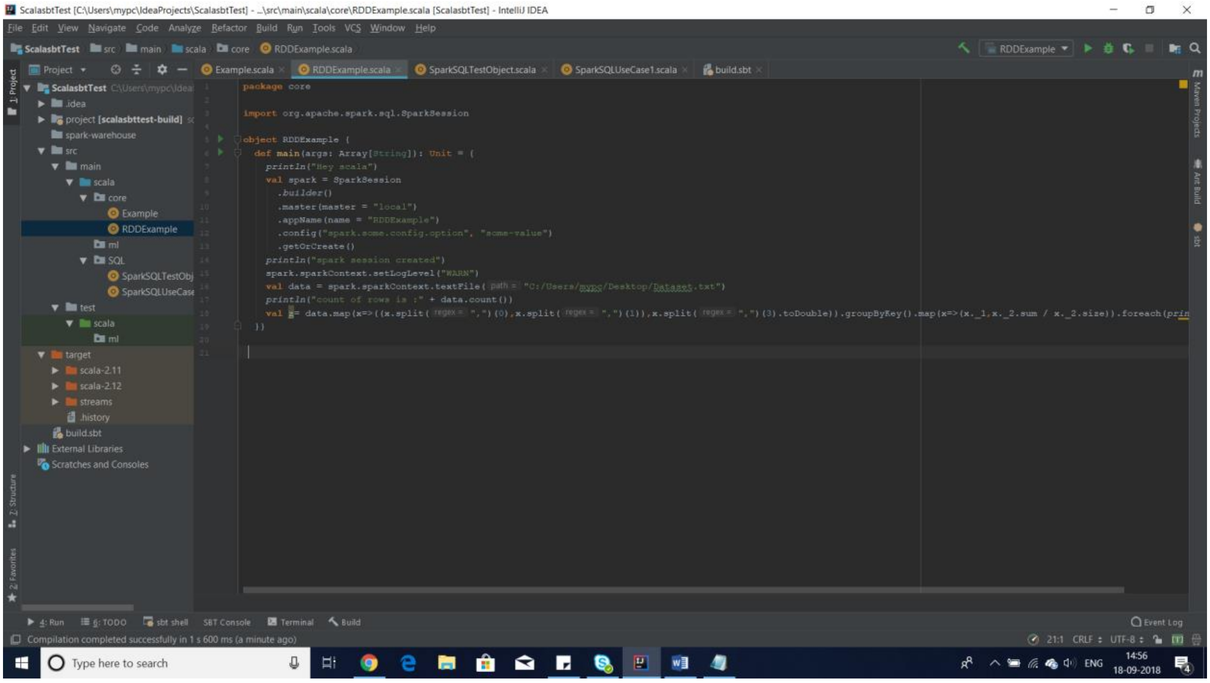
ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ScalasbtTest src main scala core RDDExample.scala
Run: RDDExample

```
    at org.apache.hadoop.util.Shell.<clinit>(Shell.java:293)
    at org.apache.hadoop.util.StringUtils.<clinit>(StringUtils.java:76)
    at org.apache.hadoop.mapred.FileInputFormat.setInputPaths(FileInputFormat.java:362)
    at org.apache.spark.SparkContext$$anonfun$hadoopFile$1$$anonfun$30.apply(SparkContext.scala:1014)
    at org.apache.spark.SparkContext$$anonfun$hadoopFile$1$$anonfun$30.apply(SparkContext.scala:1014)
    at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
    at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
    at scala.Option.foreach(Option.scala:257)
    at org.apache.spark.rdd.HadoopRDD.getJobConf(HadoopRDD.scala:179)
    at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:198)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
    at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
    at core.RDDExample$.main(RDDExample.scala:17)
    at core.RDDExample.main(RDDExample.scala)
count of rows is :22
((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),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)

Process finished with exit code 0
```

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

Explanation: Creating an RDD from textFile. Selecting name & subject as key, marks as value. Converting marks into double datatype just to avoid any truncation after decimal. groupByKey returns a tuple of ((name, subject), marks). Applying map function & selecting name, subject as first value of outer tuple & marks as second value. Adding the marks for respective subject & for each distinct subject fetching the count of marks as x._2.size to divide the sum & find the average on it. Here grouping key is student & subject.

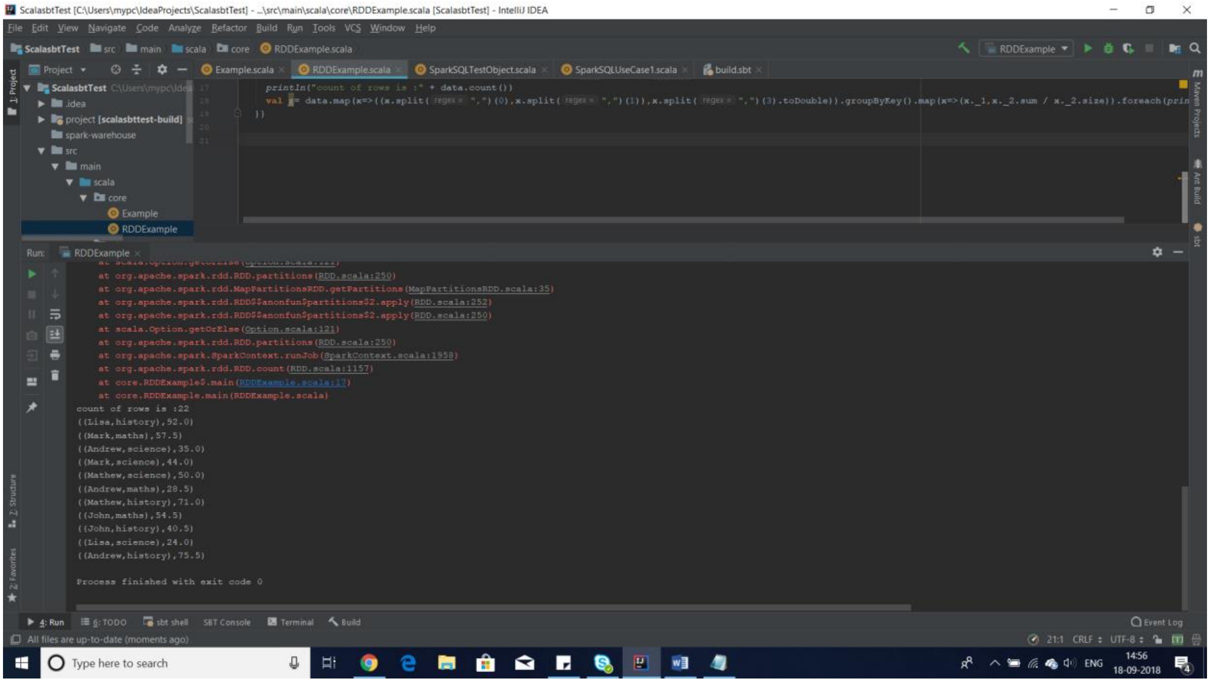
```
import org.apache.spark.sql.SparkSession


object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")

    spark.sparkContext.setLogLevel("WARN")

    val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")

    println("count of rows is :" + data.count())

    val z=
data.map(x=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x._2.size)).foreach(println)

  }}
```



ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA

```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z= data.map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(1)),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x._2.size)).foreach(prin
  }}
```

Compilation completed successfully in 1 s 600 ms (a minute ago)



ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA

```
    println("count of rows is :" + data.count())
    val z= data.map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(1)),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x._2.size)).foreach(prin
  }}
```

```
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
    at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
    at core.RDDExample$.main(RDDExample.scala:17)
    at core.RDDExample.main(RDDExample.scala)
count of rows is :22
((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)

Process finished with exit code 0
```

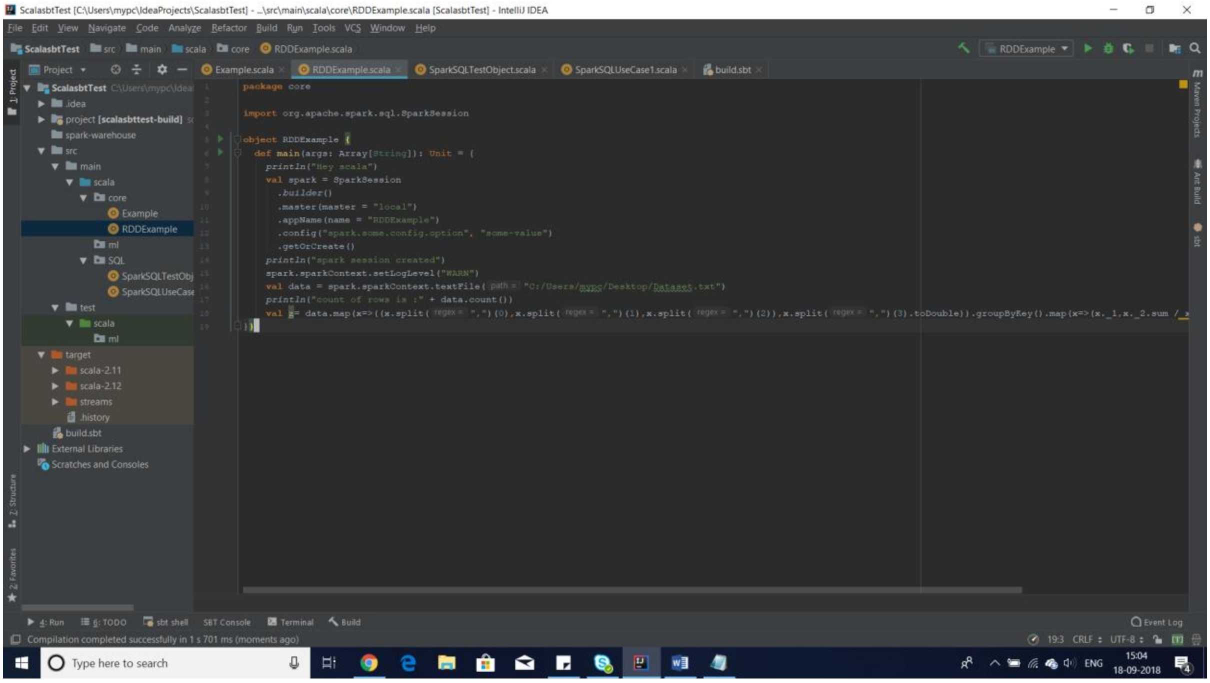
All files are up-to-date (moments ago)

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

Explanation: Creating an RDD from textFile. Selecting name, subject & grade as key, marks as value. Converting marks into double datatype just to avoid any truncation after decimal. groupByKey returns a tuple of ((name, subject, grade), marks). Applying map function

& selecting name, subject & grade as first value of outer tuple & marks as second value. Adding the marks for respective subject, grade & for each distinct subject, grade

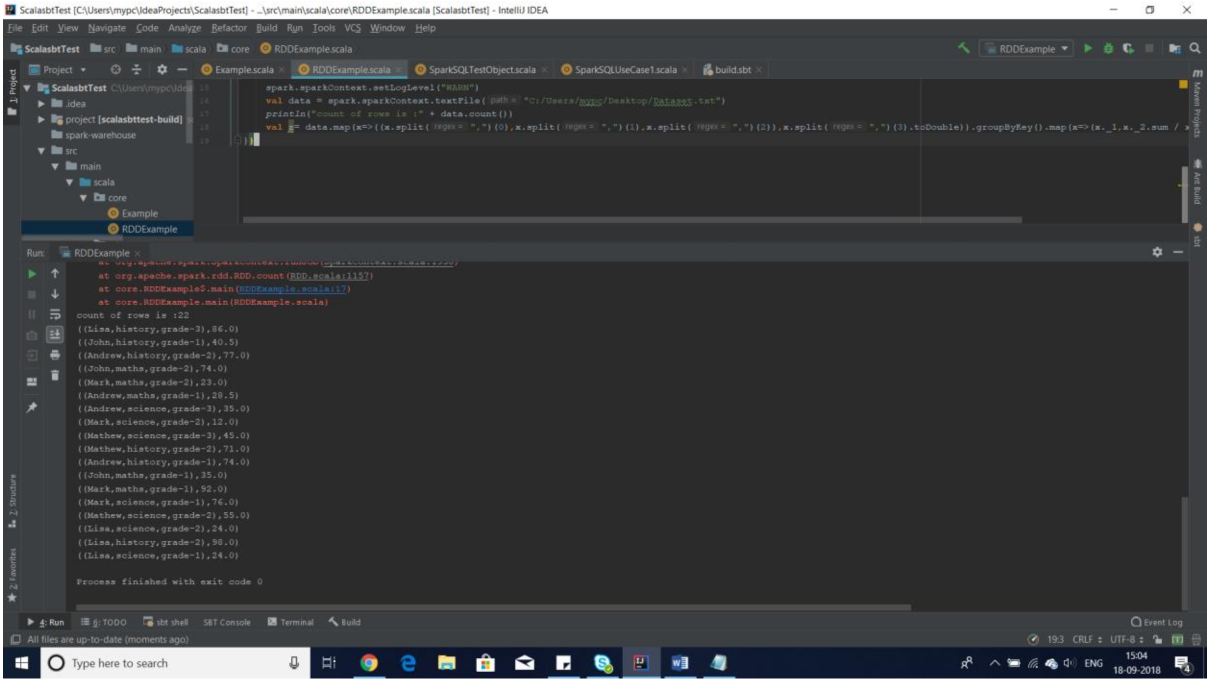
```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z=
data.map(x=>((x.split(",")(0),x.split(",")(1),x.split(",")(2)),x.split(",")(3).toDouble)).groupByKey().map(
x=>(x._1,x._2.sum / x._2.size)).foreach(println)
}}
```



ScalasbtTest [C:\Users\mypc\IdeaProjects\ScalasbtTest] - ...\src\main\scala\core\RDDExample.scala [ScalasbtTest] - IntelliJ IDEA

```scala
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z= data.map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(1),x.split( regex = ",")(2)),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum /_x
}}
```

Compilation completed successfully in 1 s 701 ms (moments ago)



```scala
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z= data.map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(1),x.split( regex = ",")(2)),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x
}}
```

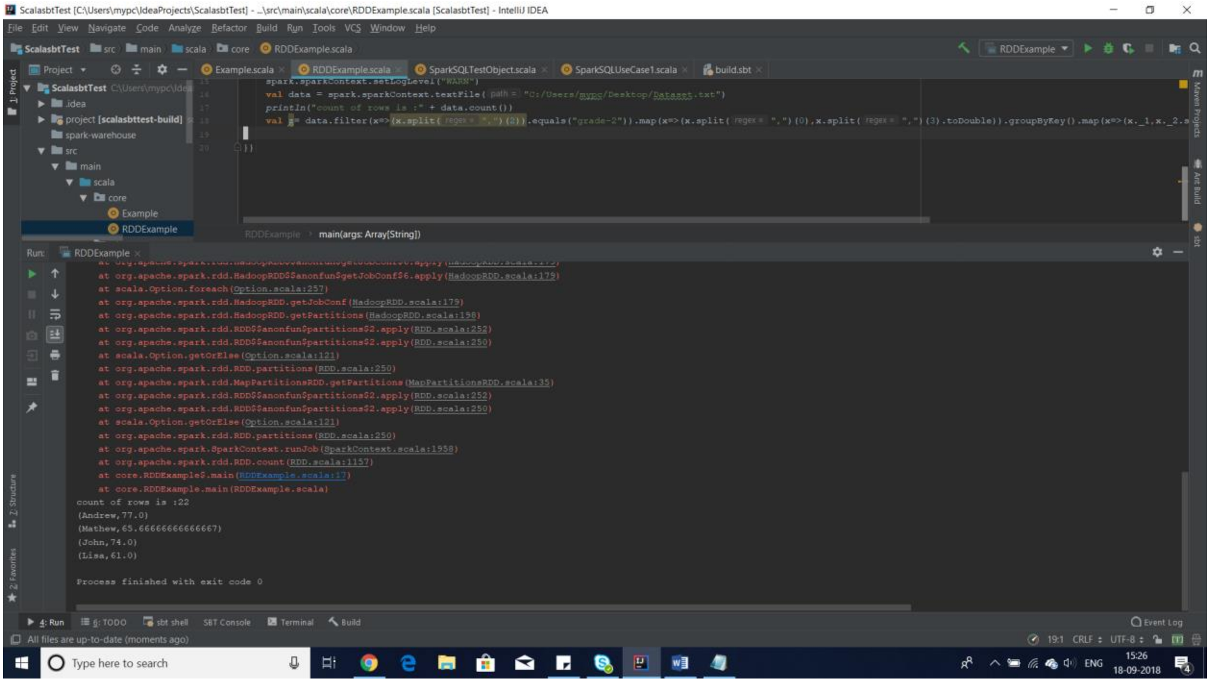
```
	at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
	at core.RDDExample$.main(RDDExample.scala:17)
	at core.RDDExample.main(RDDExample.scala)
count of rows is :22
((Lisa,history,grade-3),86.0)
((John,history,grade-1),40.5)
((Andrew,history,grade-2),77.0)
((John,maths,grade-2),74.0)
((Mark,maths,grade-2),23.0)
((Andrew,maths,grade-1),28.5)
((Andrew,science,grade-3),35.0)
((Mark,science,grade-2),12.0)
((Mathew,science,grade-3),45.0)
((Mathew,history,grade-2),71.0)
((Andrew,history,grade-1),74.0)
((John,maths,grade-1),35.0)
((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)

Process finished with exit code 0
```

All files are up-to-date (moments ago)

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

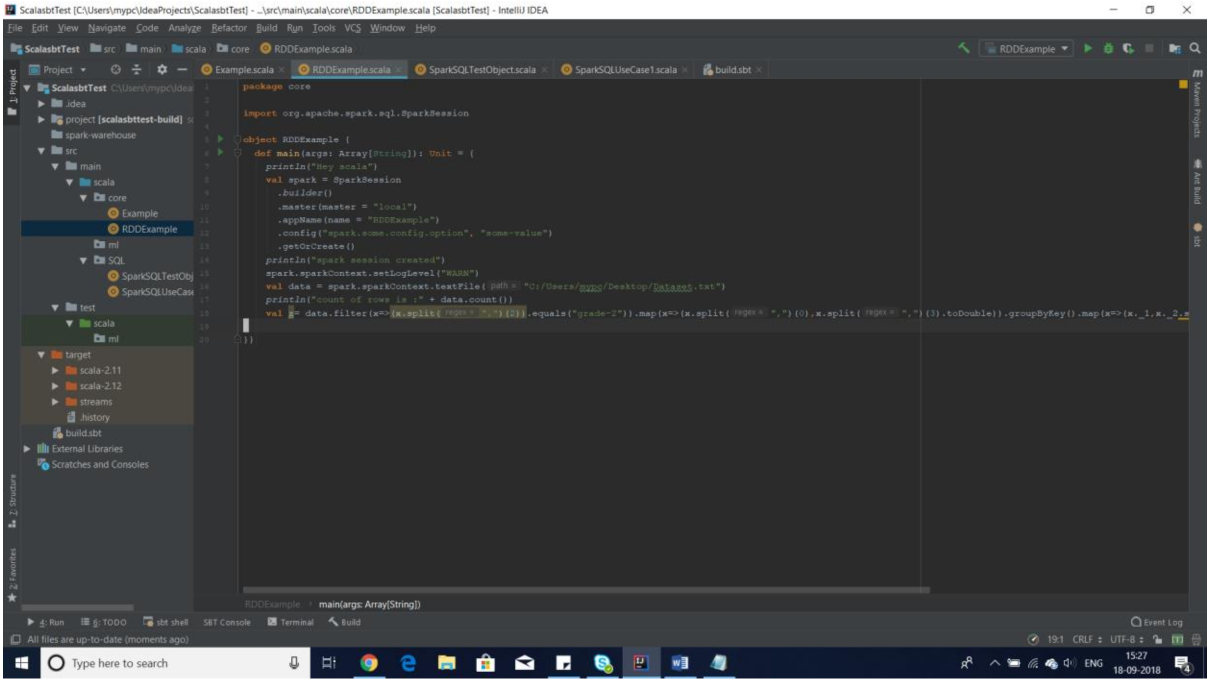
Explanation: After the RDD creation. Filtering the records where grade equals grade-2 & then splitting the record where key is name value is marks. Thereby grouping by Name to calculate average of marks by calculating total marks for each name & dividing by count of times marks were awarded to each student. At last, applying filter & counting whoever has average marks greater than 50.

```
package core


import org.apache.spark.sql.SparkSession


object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z= data.filter(x=>(x.split(",")(2)).equals("grade-2")).map(x=>(x.split(",")(0),x.split(",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x._2.size)).filter(x=>(x._2>50)).foreach(println)


}}
```



```
spark.sparkContext.setLogLevel("WARN")
val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")
println("count of rows is :" + data.count())
val z= data.filter(x=>(x.split(",")(2)).equals("grade-2")).map(x=>(x.split(",")(0),x.split(",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.s
}}
```

```
    at org.apache.spark.rdd.HadoopRDD$$anonfun$getJobConf$6.apply(HadoopRDD.scala:179)
    at scala.Option.foreach(Option.scala:257)
    at org.apache.spark.rdd.HadoopRDD.getJobConf(HadoopRDD.scala:179)
    at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:198)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
    at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
    at org.apache.spark.SparkContext.runJob(SparkContext.scala:1958)
    at org.apache.spark.rdd.RDD.count(RDD.scala:1157)
    at core.RDDExample$.main(RDDExample.scala:17)
    at core.RDDExample.main(RDDExample.scala)
count of rows is :22
(Andrew,77.0)
(Mathew,65.66666666666667)
(John,74.0)
(Lisa,61.0)

Process finished with exit code 0
```



```
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
    val z= data.filter(x=>(x.split(",")(2)).equals("grade-2")).map(x=>(x.split(",")(0),x.split(",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.s

}}
```

Problem Statement 3:

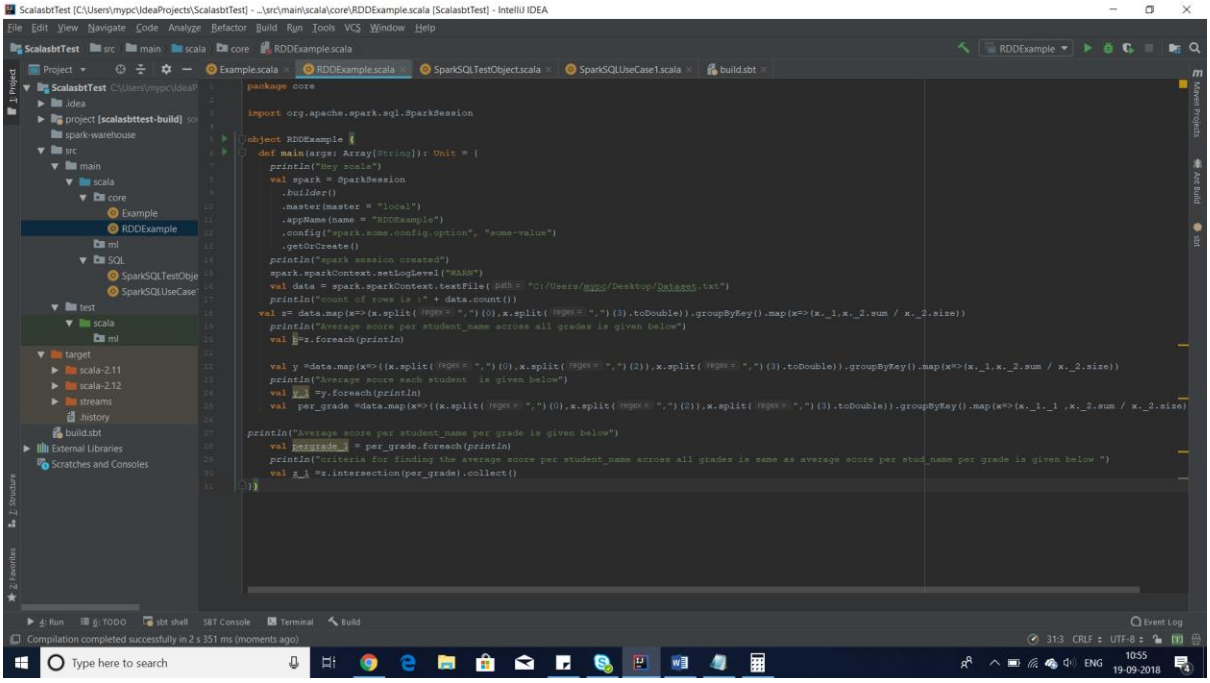
Are there any students in the college that satisfy the below criteria:

1. Average score per student_name across all grades is same as average score per stud_name per grade

package core

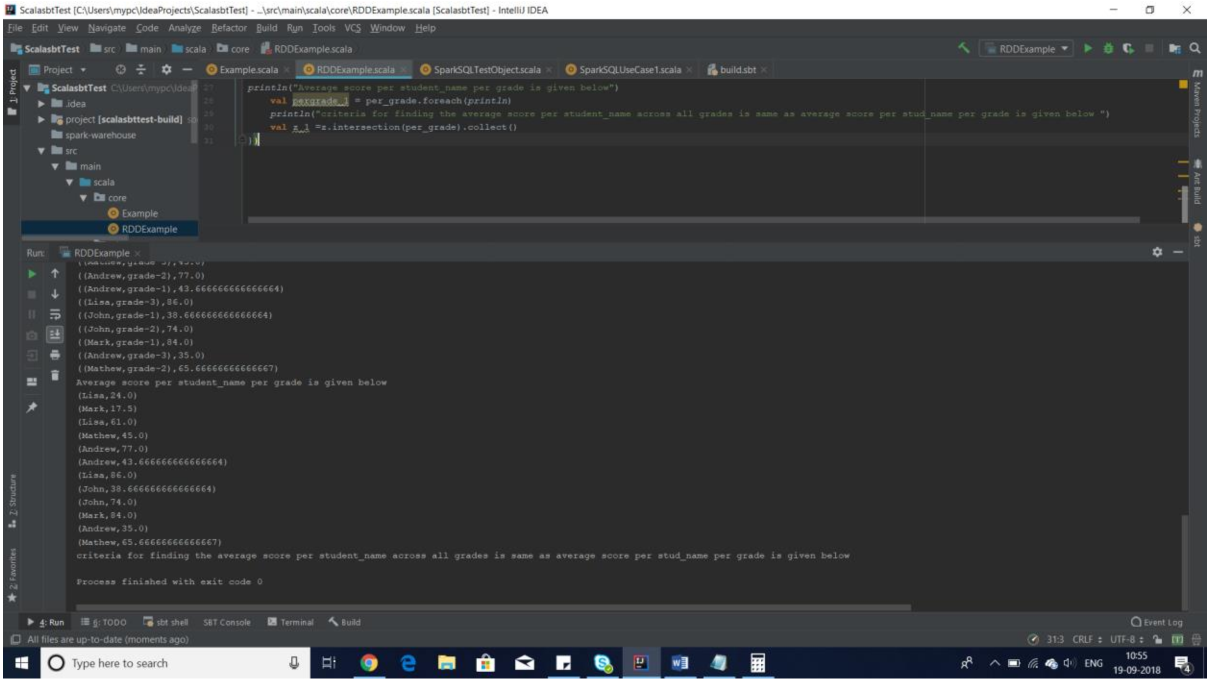
Solution: Here we are using the intersection property to find the above criteria. Finding the common data in both the lists by using the intersection property.

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

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile("C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
  //Finding the average score per student_name across all grades


 val z = data.map(x=>(x.split(",")(0),x.split(",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum /
x._2.size))

    println("Average score per student_name across all grades is given below")
    val b=z.foreach(println)
  //average score per student per grade(

    val y
  =data.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toDouble)).groupByKey().map(x=>(
  x._1,x._2.sum / x._2.size))
    println("Average score each student  is given below")
    val y_1 =y.foreach(println)
  //average score per student per grade

 val  per_grade
=data.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toDouble)).groupByKey().map(x>(x._1._1
,x._2.sum / x._2.size))


  println("Average score per student_name per grade is given below")
    val pergrade_1 = per_grade.foreach(println)
    println("criteria for finding the average score per student_name across all grades is same
  as average score per stud_name per grade is given below ")
    val z_1 =z.intersection(per_grade).collect()
  }}
```



```scala
package core

import org.apache.spark.sql.SparkSession

object RDDExample {
  def main(args: Array[String]): Unit = {
    println("Hey scala")
    val spark = SparkSession
      .builder()
      .master(master = "local")
      .appName(name = "RDDExample")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("spark session created")
    spark.sparkContext.setLogLevel("WARN")
    val data = spark.sparkContext.textFile( path = "C:/Users/mypc/Desktop/Dataset.txt")
    println("count of rows is :" + data.count())
  val z= data.map(x=>(x.split( regex = ",")(0),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x._2.size))
    println("Average score per student_name across all grades is given below")
    val b=z.foreach(println)

    val y =data.map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(2)),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1,x._2.sum / x._2.size))
    println("Average score each student  is given below")
    val y_1 =y.foreach(println)
    val  per_grade =data.map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(2)),x.split( regex = ",")(3).toDouble)).groupByKey().map(x=>(x._1._1 ,x._2.sum / x._2.size)

println("Average score per student_name per grade is given below")
    val pergrade_1 = per_grade.foreach(println)
    println("criteria for finding the average score per student_name across all grades is same as average score per stud_name per grade is given below ")
    val z_1 =z.intersection(per_grade).collect()
}}
```

Compilation completed successfully in 2 s 351 ms (moments ago)



```scala
println("Average score per student_name per grade is given below")
    val pergrade_1 = per_grade.foreach(println)
    println("criteria for finding the average score per student_name across all grades is same as average score per stud_name per grade is given below ")
    val z_1 =z.intersection(per_grade).collect()
}}
```

```
((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),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
Average score per student_name per grade is given below
(Lisa,24.0)
(Mark,17.5)
(Lisa,61.0)
(Mathew,45.0)
(Andrew,77.0)
(Andrew,43.666666666666664)
(Lisa,86.0)
(John,38.666666666666664)
(John,74.0)
(Mark,84.0)
(Andrew,35.0)
(Mathew,65.66666666666667)
criteria for finding the average score per student_name across all grades is same as average score per stud_name per grade is given below

Process finished with exit code 0
```

All files are up-to-date (moments ago)

Note: Comparing the results using the intersection property we found that there was no common data in both the lists. There is no record that matches the criteria. After using the intersection property we did not get any output result hence no common data that matches the criteria.