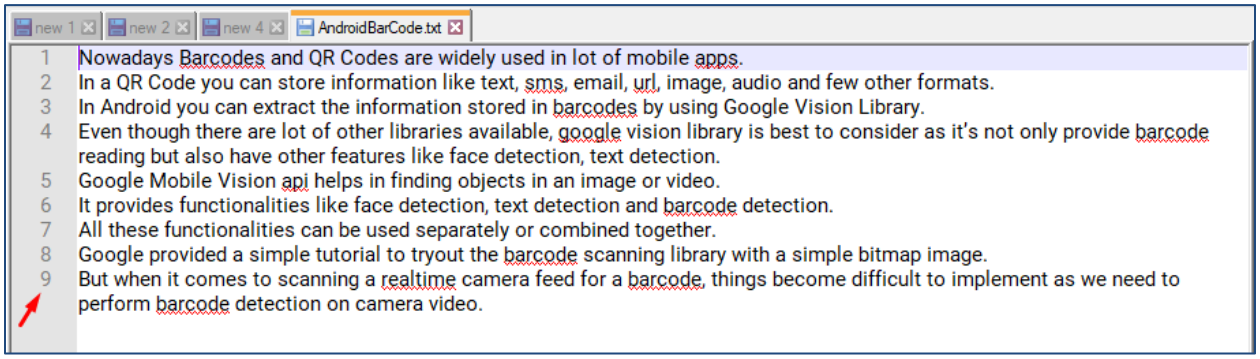
Session 19:
RDD DEEP DIVE
Assignment 1

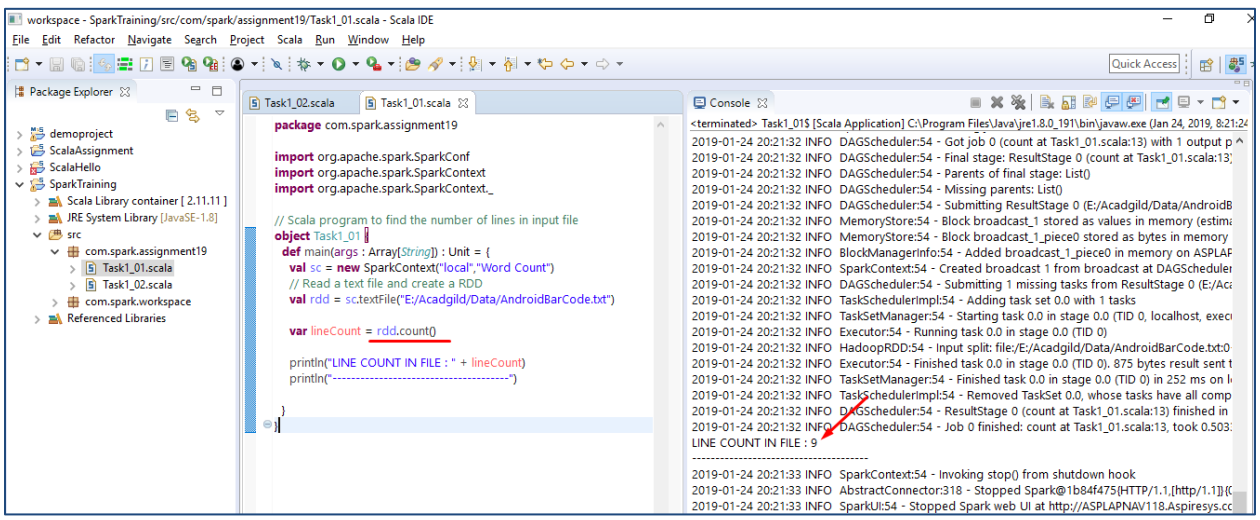
TASK 1:

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

   Input File - *AndroidBarCode.txt*

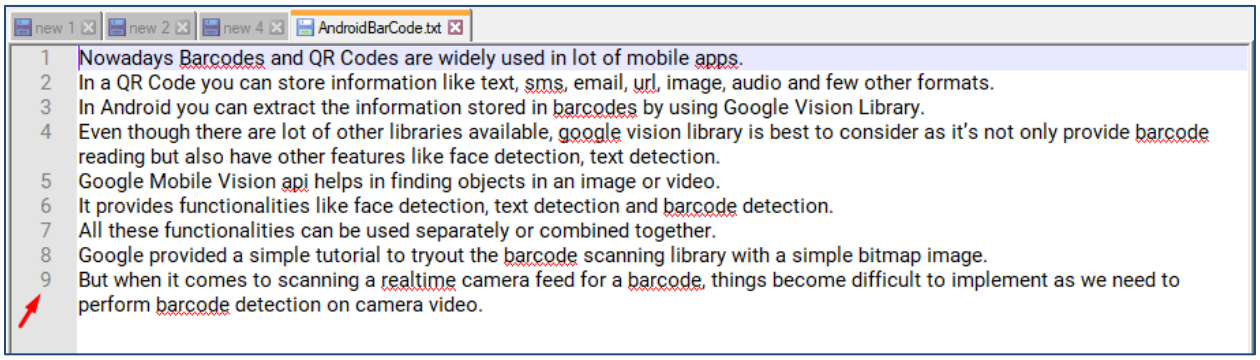


   Program & Execution:

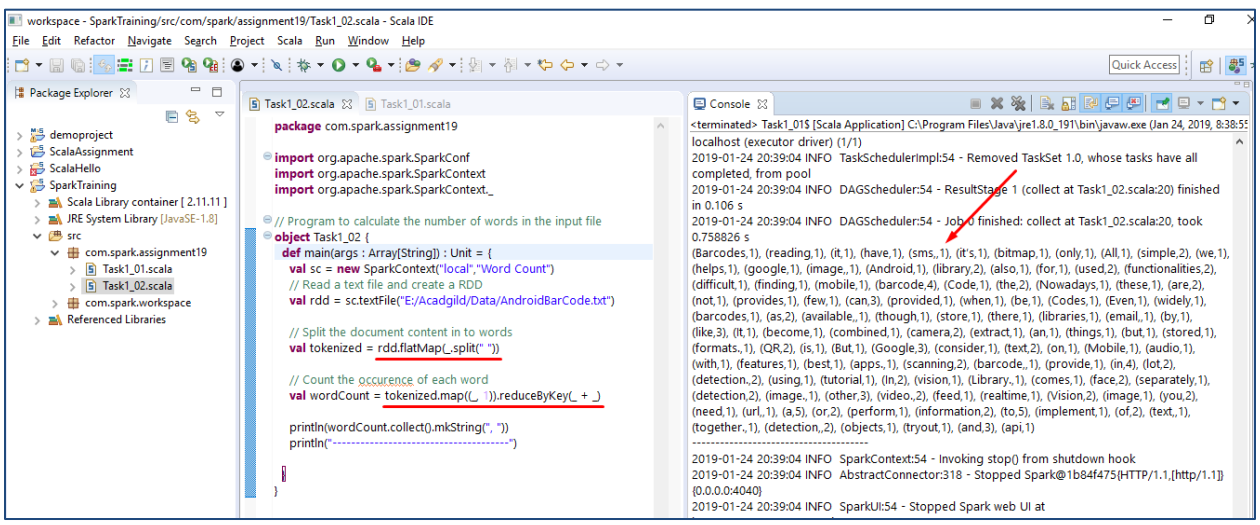


2. Write a program to read a text file and print the number of words in the document.

Input File - *AndroidBarCode.txt*

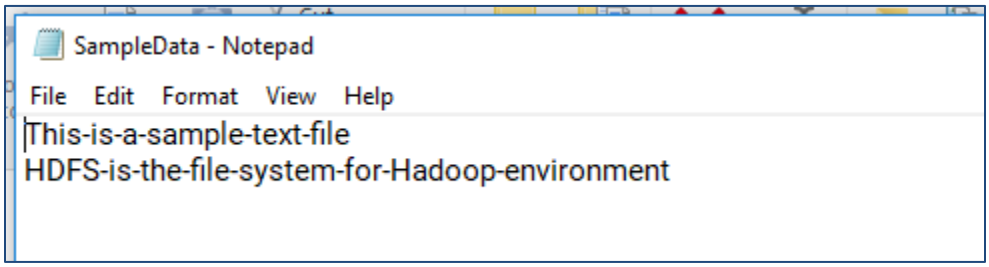


Program & Execution:

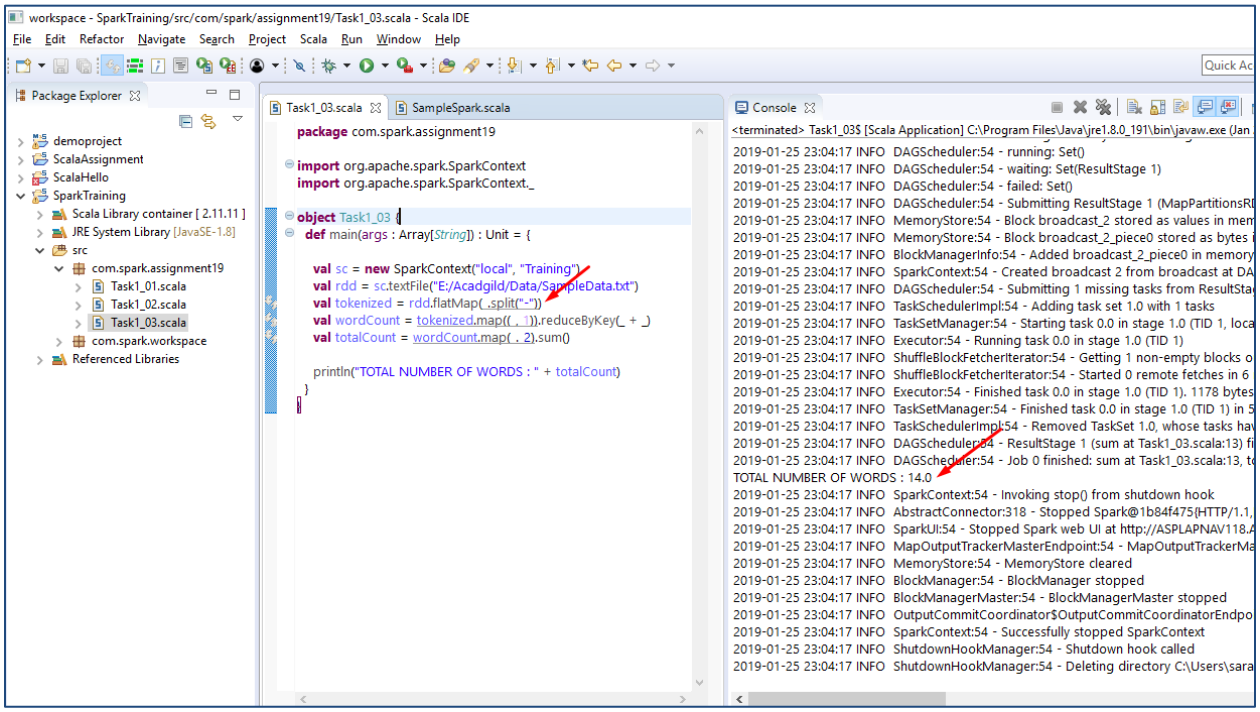


3. We have a document where the word separator is -, instead of space. Write a spark code, to obtain the count of the total number of words present in the document.

Input Data: *SampleData.txt*



Program & Execution:



TASK 2:

Problem Statement – 1:

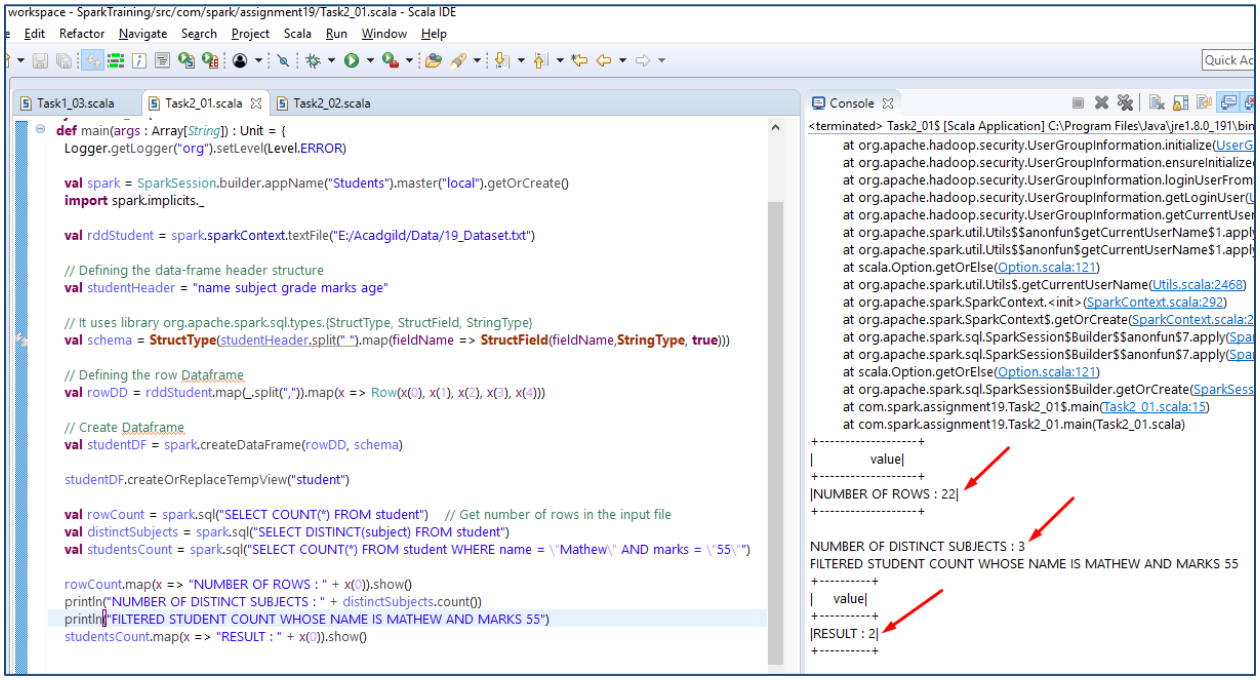
1. Read the text file, and create a tupled rdd.
2. Find the count of total number of rows present.
3. What is the distinct number of subjects present in the entire school
4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

Input File: *19_Dataset.txt (Columns: Name, Subject, Grade, Marks, Age)*

new 1 | new 2 | 19_Dataset.txt

```
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
```

Program & Execution:



Problem Statement – 2:

1. What is the count of students per grade in the school?
2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)
3. What is the average score of students in each subject across all grades?
4. What is the average score of students in each subject per grade?
5. For all students in grade-2, how many have average score greater than 50?

Program & Execution:

Task1_03.scala | Task2_01.scala | Task2_02.scala

```scala
package com.spark.assignment19

import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.log4j._
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.types.{StructType, StructField, StringType}
import org.apache.spark.sql.Row

object Task2_02 {
 def main(args : Array[String]) : Unit = {
  Logger.getLogger("org").setLevel(Level.ERROR)

  val spark = SparkSession.builder.appName("Students").master("local").getOrCreate()
  import spark.implicits._

  val rddStudent = spark.sparkContext.textFile("E:/Acadgild/Data/19_Dataset.txt")

  // Defining the data-frame header structure
  val studentHeader = "name subject grade marks age"

  // It uses library org.apache.spark.sql.types.{StructType, StructField, StringType}
  val schema = StructType(studentHeader.split(" ").map(fieldName => StructField(fieldName,StringType, true)))

  // Defining the row Dataframe
  val rowDD = rddStudent.map(_.split(",")).map(x => Row(x(0), x(1), x(2), x(3), x(4)))

  // Create Dataframe
  val studentDF = spark.createDataFrame(rowDD, schema)

  studentDF.createOrReplaceTempView("student")
```

```scala
  val groupStudents = spark.sql("SELECT grade, COUNT(*) FROM student GROUP BY grade")
  println("1 --> NUMBER OF STUDENTS BY GRADE")
  groupStudents.show()

  val avgOfStudents = spark.sql("SELECT name, grade, AVG(marks) FROM student GROUP BY name, grade")
  println("2 -->  AVERAGE OF EACH STUDENT")
  avgOfStudents.show()

  val avgScoreGrades = spark.sql("SELECT subject, AVG(marks) FROM student GROUP BY subject")
  println("3 --> AVERAGE SCORE OF STUDENTS IN EACH SUBJECT ACROSS ALL GRADES")
  avgScoreGrades.show()

  val avgScoreSubject = spark.sql("SELECT subject, grade, AVG(marks) FROM student GROUP BY subject, grade")
  println("4 --> AVERAGE SCORE OF STUDENTS IN EACH SUBJECT PER GRADE")
  avgScoreSubject.show()

  val avgScore = spark.sql("SELECT name, AVG(marks) FROM student WHERE grade = \"grade-2\" GROUP BY name HAVING AVG(marks) > 50")
  println("5 --> ALL STUDENTS IN GRADE-2, NUMBER OF THOSE HAVING AVERAGE SCORE GREATER THAN 50")
  avgScore.show()
 }
}
```

```
1 --> NUMBER OF STUDENTS BY GRADE
+-------+--------+
|  grade|count(1)|
+-------+--------+
|grade-3|       4|
|grade-1|       9|
|grade-2|       9|
+-------+--------+

2 -->  AVERAGE OF EACH STUDENT
+------+-------+-------------------------+
|  name|  grade|avg(CAST(marks AS DOUBLE))|
+------+-------+-------------------------+
|  Mark|grade-2|                     17.5|
|  John|grade-2|                     74.0|
|  Mark|grade-1|                     84.0|
|  Lisa|grade-3|                     86.0|
|  Lisa|grade-2|                     61.0|
|  John|grade-1|        38.666666666666664|
|Mathew|grade-3|                     45.0|
|Andrew|grade-1|        43.666666666666664|
|  Lisa|grade-1|                     24.0|
|Andrew|grade-3|                     35.0|
|Mathew|grade-2|        65.66666666666667|
|Andrew|grade-2|                     77.0|
+------+-------+-------------------------+
```

```
3 --> AVERAGE SCORE OF STUDENTS IN EACH SUBJECT ACROSS ALL GRADES
+-------+-------------------------+
|subject|avg(CAST(marks AS DOUBLE))|
+-------+-------------------------+
|  maths|        46.833333333333336|
|history|                    69.75|
|science|                    38.25|
+-------+-------------------------+

4 --> AVERAGE SCORE OF STUDENTS IN EACH SUBJECT PER GRADE
+-------+-------+-------------------------+
|subject|  grade|avg(CAST(marks AS DOUBLE))|
+-------+-------+-------------------------+
|history|grade-1|        51.666666666666664|
|history|grade-3|                     86.0|
|  maths|grade-2|                     48.5|
|science|grade-1|                     50.0|
|science|grade-3|        38.333333333333336|
|  maths|grade-1|                     46.0|
|science|grade-2|        30.333333333333332|
|history|grade-2|                    79.25|
+-------+-------+-------------------------+

5 --> ALL STUDENTS IN GRADE-2, NUMBER OF THOSE HAVING AVERAGE SCORE GREATER THAN 50
+------+-------------------------+
|  name|avg(CAST(marks AS DOUBLE))|
+------+-------------------------+
|Mathew|        65.66666666666667|
|  John|                    74.0|
|Andrew|                    77.0|
|  Lisa|                    61.0|
+------+-------------------------+
```

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

Program & Execution:

Task2_01.scala | Task2_02.scala | Task2_03.scala

```scala
package com.spark.assignment19

import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.log4j._
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.types.{StructType, StructField, StringType}
import org.apache.spark.sql.Row

object Task2_03 {
  def main(args : Array[String]) : Unit = {
    Logger.getLogger("org").setLevel(Level.ERROR)

    val spark = SparkSession.builder.appName("Students").master("local").getOrCreate()
    import spark.implicits._

    val rddStudent = spark.sparkContext.textFile("E:/Acadgild/Data/19_Dataset.txt")

    // Defining the data-frame header structure
    val studentHeader = "name subject grade marks age"

    // It uses library org.apache.spark.sql.types.{StructType, StructField, StringType}
    val schema = StructType(studentHeader.split(" ").map(fieldName => StructField(fieldName,StringType, true)))

    // Defining the row Dataframe
    val rowDD = rddStudent.map(_.split(",")).map(x => Row(x(0), x(1), x(2), x(3), x(4)))

    // Create Dataframe
    val studentDF = spark.createDataFrame(rowDD, schema)

    studentDF.createOrReplaceTempView("student")
```

```scala
    println("Average score per student_name across all grades is same as average score per student_name per grad
    val avgPerStudAllGrades = spark.sql("SELECT name, AVG(marks) FROM student GROUP BY name")
    val avgPerStudPerGrade = spark.sql("SELECT name, AVG(marks) FROM student GROUP BY name, grade")

    avgPerStudAllGrades.show()
    avgPerStudPerGrade.show()

    println("INTERSECTION OF DATAFRAMES")
    val resultIntersect = avgPerStudAllGrades.intersect(avgPerStudPerGrade)
    if (resultIntersect.count() == 0) {
      println("There is no common average score in two data frames")
    }
    else {
      resultIntersect.show()
    }

  }
}
```

```
Console
<terminated> Task2_03$ [Scala Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Jan 2
        at com.spark.assignment19.Task2_03.main(Task2_03.scala)
Average score per student_name across all grades is same as average score per student
+------+-------------------------+
|  name|avg(CAST(marks AS DOUBLE))|
+------+-------------------------+
|Mathew|                     60.5|
|  Mark|                    50.75|
|  John|                     47.5|
|Andrew|       46.333333333333336|
|  Lisa|                     58.0|
+------+-------------------------+

+------+-------------------------+
|  name|avg(CAST(marks AS DOUBLE))|
+------+-------------------------+
|  Mark|                     17.5|
|  John|                     74.0|
|  Mark|                     84.0|
|  Lisa|                     86.0|
|  Lisa|                     61.0|
|  John|       38.666666666666664|
|Mathew|                     45.0|
|Andrew|       43.666666666666664|
|  Lisa|                     24.0|
|Andrew|                     35.0|
|Mathew|       65.66666666666667|
|Andrew|                     77.0|
+------+-------------------------+

INTERSECTION OF DATAFRAMES
There is no common average score in two data frames
```