

Session 19:

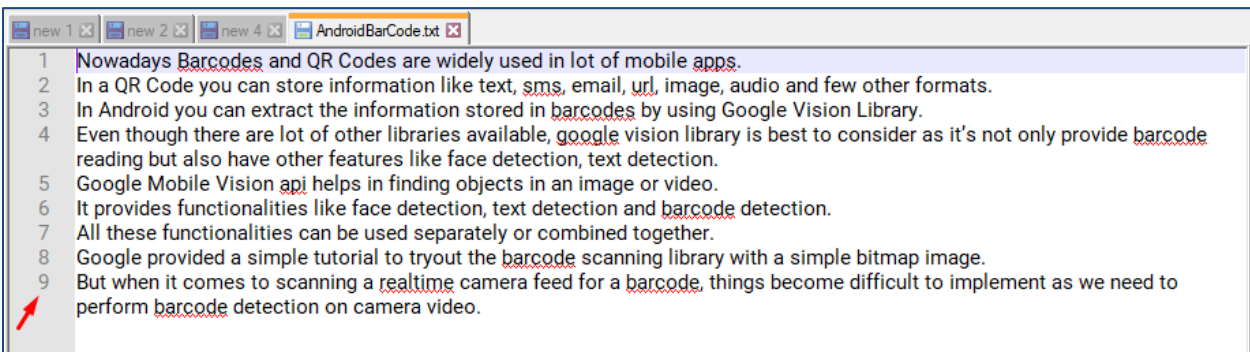
RDD DEEP DIVE

Assignment 1

TASK 1:

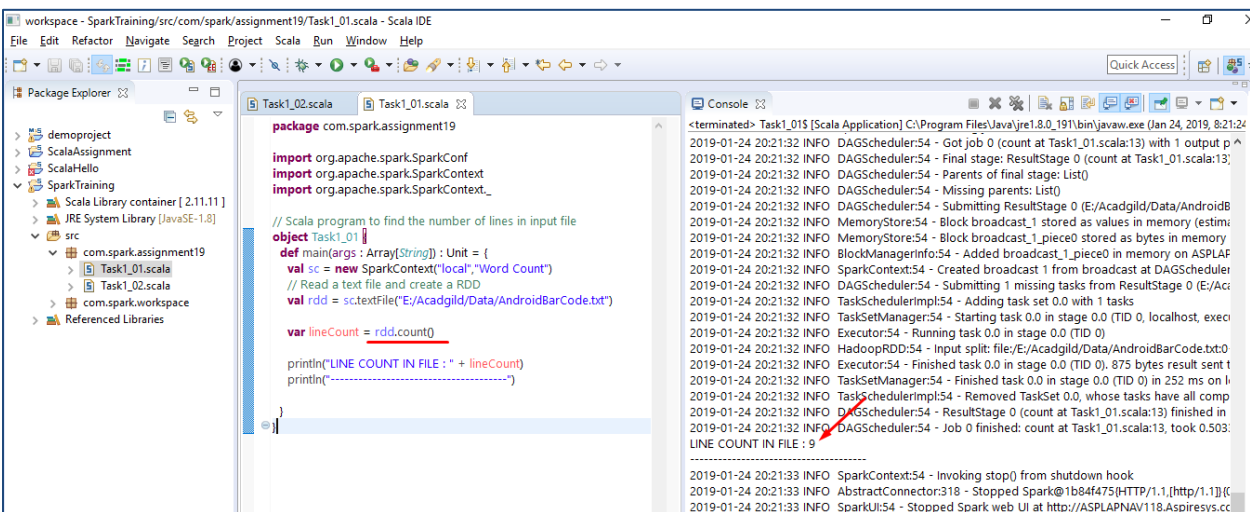
1. Write a program to read a text file and print the number of rows of data in the document.

Input File - *AndroidBarCode.txt*



```
1 Nowadays Barcodes and QR Codes are widely used in lot of mobile apps.
2 In a QR Code you can store information like text, sms, email, url, image, audio and few other formats.
3 In Android you can extract the information stored in barcodes by using Google Vision Library.
4 Even though there are lot of other libraries available, google vision library is best to consider as it's not only provide barcode
  reading but also have other features like face detection, text detection.
5 Google Mobile Vision api helps in finding objects in an image or video.
6 It provides functionalities like face detection, text detection and barcode detection.
7 All these functionalities can be used separately or combined together.
8 Google provided a simple tutorial to tryout the barcode scanning library with a simple bitmap image.
9 But when it comes to scanning a realtime camera feed for a barcode, things become difficult to implement as we need to
  perform barcode detection on camera video.
```

Program & Execution:



```
package com.spark.assignment19

import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

// Scala program to find the number of lines in input file
object Task1_01 {
  def main(args: Array[String]): Unit = {
    val sc = new SparkContext("local", "Word Count")
    // Read a text file and create a RDD
    val rdd = sc.textFile("E:/Acadgild/Data/AndroidBarCode.txt")

    var lineCount = rdd.count()

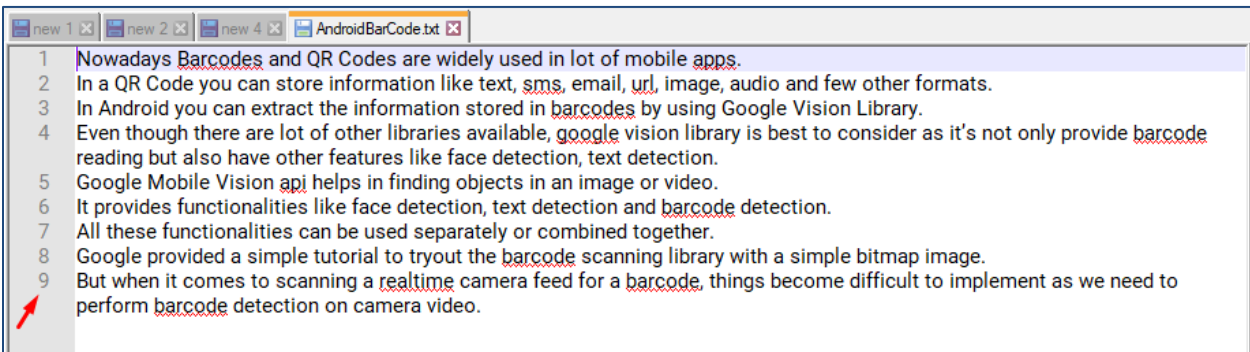
    println("LINE COUNT IN FILE : " + lineCount)
    println("-----")
  }
}
```

Console Output:

```
<terminated> Task1_01$ [Scala Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Jan 24, 2019, 8:21:24)
2019-01-24 20:21:32 INFO DAGScheduler:54 - Got job 0 (count at Task1_01.scala:13) with 1 output p ^
2019-01-24 20:21:32 INFO DAGScheduler:54 - Final stage: ResultStage 0 (count at Task1_01.scala:13)
2019-01-24 20:21:32 INFO DAGScheduler:54 - Parents of final stage: List()
2019-01-24 20:21:32 INFO DAGScheduler:54 - Missing parents: List()
2019-01-24 20:21:32 INFO DAGScheduler:54 - Submitting ResultStage 0 (E:/Acadgild/Data/AndroidB
2019-01-24 20:21:32 INFO MemoryStore:54 - Block broadcast_1 stored as values in memory (estim
2019-01-24 20:21:32 INFO MemoryStore:54 - Block broadcast_1_piece0 stored as bytes in memory
2019-01-24 20:21:32 INFO BlockManagerInfo:54 - Added broadcast_1_piece0 in memory on ASPLAF
2019-01-24 20:21:32 INFO SparkContext:54 - Created broadcast 1 from broadcast at DAGScheduler
2019-01-24 20:21:32 INFO DAGScheduler:54 - Submitting 1 missing tasks from ResultStage 0 (E:/Ac
2019-01-24 20:21:32 INFO TaskSchedulerImpl:54 - Adding task set 0.0 with 1 tasks
2019-01-24 20:21:32 INFO TaskSetManager:54 - Starting task 0.0 in stage 0.0 (TID 0, localhost, exec
2019-01-24 20:21:32 INFO Executor:54 - Running task 0.0 in stage 0.0 (TID 0)
2019-01-24 20:21:32 INFO HadoopRDD:54 - Input split: file:/E:/Acadgild/Data/AndroidBarCode.txt:0
2019-01-24 20:21:32 INFO Executor:54 - Finished task 0.0 in stage 0.0 (TID 0), 875 bytes result sent t
2019-01-24 20:21:32 INFO TaskSetManager:54 - Finished task 0.0 in stage 0.0 (TID 0) in 252 ms on l
2019-01-24 20:21:32 INFO TaskSchedulerImpl:54 - Removed TaskSet 0.0, whose tasks have all comp
2019-01-24 20:21:32 INFO DAGScheduler:54 - ResultStage 0 (count at Task1_01.scala:13) finished in
2019-01-24 20:21:32 INFO DAGScheduler:54 - Job 0 finished: count at Task1_01.scala:13, took 0.503
  LINE COUNT IN FILE : 9
-----
2019-01-24 20:21:33 INFO SparkContext:54 - Invoking stop() from shutdown hook
2019-01-24 20:21:33 INFO AbstractConnector:318 - Stopped Spark@1b84f475@http://1.1.1.1:11111
2019-01-24 20:21:33 INFO SparkUI:54 - Stopped Spark web UI at http://ASPLAPNAV118.Aspiresys.cc
```

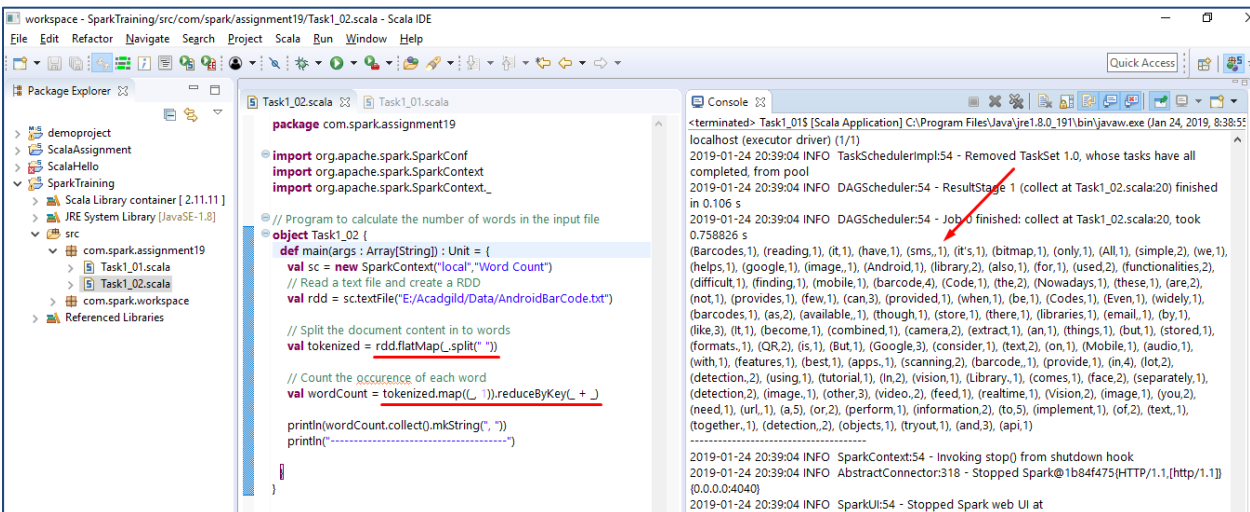
2. Write a program to read a text file and print the number of words in the document.

Input File - *AndroidBarCode.txt*



```
1 Nowadays Barcodes and QR Codes are widely used in lot of mobile apps.
2 In a QR Code you can store information like text, sms, email, url, image, audio and few other formats.
3 In Android you can extract the information stored in barcodes by using Google Vision Library.
4 Even though there are lot of other libraries available, google vision library is best to consider as it's not only provide barcode
5 reading but also have other features like face detection, text detection.
6 Google Mobile Vision api helps in finding objects in an image or video.
7 It provides functionalities like face detection, text detection and barcode detection.
8 All these functionalities can be used separately or combined together.
9 Google provided a simple tutorial to tryout the barcode scanning library with a simple bitmap image.
10 But when it comes to scanning a realtime camera feed for a barcode, things become difficult to implement as we need to
11 perform barcode detection on camera video.
```

Program & Execution:



```
package com.spark.assignment19
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._

// Program to calculate the number of words in the input file
object Task1_02 {
  def main(args: Array[String]) : Unit = {
    val sc = new SparkContext("local", "Word Count")
    // Read a text file and create a RDD
    val rdd = sc.textFile("E:/Acadgild/Data/AndroidBarCode.txt")

    // Split the document content in to words
    val tokenized = rdd.flatMap(_.split(" "))

    // Count the occurrence of each word
    val wordCount = tokenized.map(_._1).reduceByKey(_ + _)

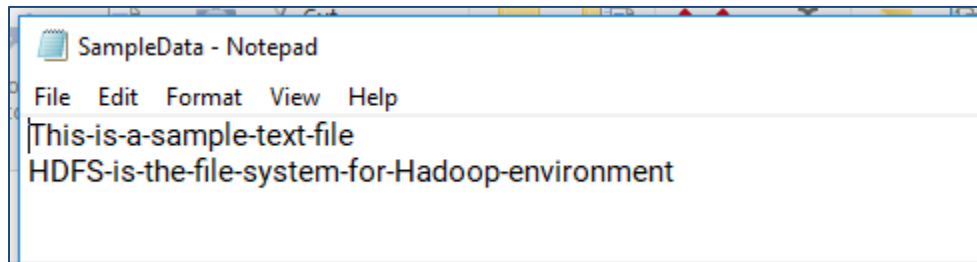
    println(wordCount.collect().mkString(", "))
    println("-----")
  }
}
```

Console Output:

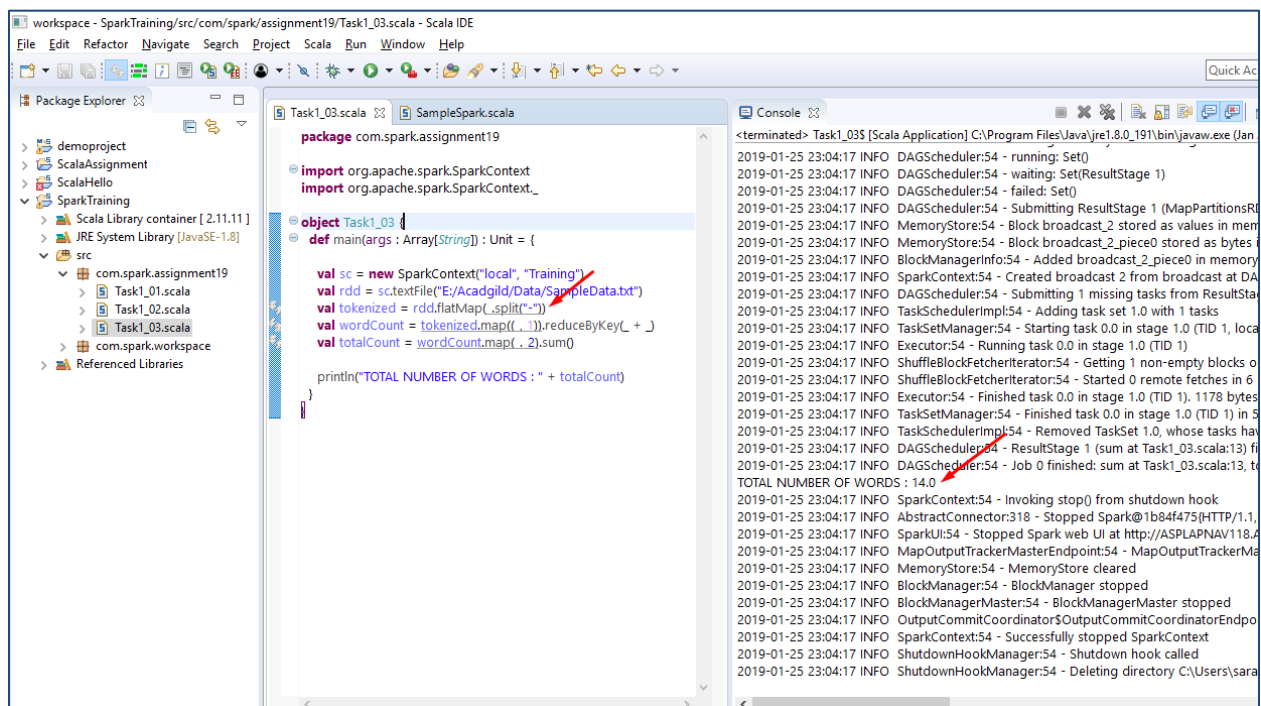
```
<terminated> Task1_01$ [Scala Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Jan 24, 2019, 8:38:55)
localhost (executor driver) (1/1)
2019-01-24 20:39:04 INFO TaskSchedulerImpl:54 - Removed TaskSet 1.0, whose tasks have all
completed, from pool.
2019-01-24 20:39:04 INFO DAGScheduler:54 - ResultStage 1 (collect at Task1_02.scala:20) finished
in 0.106 s
2019-01-24 20:39:04 INFO DAGScheduler:54 - Job 0 finished: collect at Task1_02.scala:20, took
0.758826 s
(Barcodes,1), (reading,1), (it,1), (have,1), (sms,1), (it's,1), (bitmap,1), (only,1), (All,1), (simple,2), (we,1),
(helps,1), (google,1), (image,1), (Android,1), (library,2), (also,1), (for,1), (used,2), (functionalities,2),
(difficult,1), (finding,1), (mobile,1), (barcode,4), (Code,1), (the,2), (Nowadays,1), (these,1), (are,2),
(not,1), (provides,1), (few,1), (can,3), (provided,1), (when,1), (be,1), (Codes,1), (Even,1), (widely,1),
(barcodes,1), (as,2), (available,1), (though,1), (store,1), (there,1), (libraries,1), (email,1), (by,1),
(like,3), (it,1), (become,1), (combined,1), (camera,2), (extract,1), (an,1), (things,1), (but,1), (stored,1),
(formats,1), (QR,2), (is,1), (But,1), (Google,3), (consider,1), (text,2), (on,1), (Mobile,1), (audio,1),
(with,1), (features,1), (best,1), (apps,1), (scanning,2), (barcode,1), (provide,1), (in,4), (lot,2),
(detection,2), (using,1), (tutorial,1), (in,2), (vision,1), (Library,1), (comes,1), (face,2), (separately,1),
(detection,2), (image,1), (other,3), (video,2), (feed,1), (realtime,1), (Vision,2), (image,1), (you,2),
(need,1), (url,1), (a,5), (or,2), (perform,1), (information,2), (to,5), (implement,1), (of,2), (text,1),
(together,1), (detection,2), (objects,1), (tryout,1), (and,3), (api,1)
-----
2019-01-24 20:39:04 INFO SparkContext:54 - Invoking stop() from shutdown hook
2019-01-24 20:39:04 INFO AbstractConnector:318 - Stopped Spark@1b84f475[http://1.1.[http://1.1]]
[0.0.0.0:4040]
2019-01-24 20:39:04 INFO SparkUI:54 - Stopped Spark web UI at
```

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:

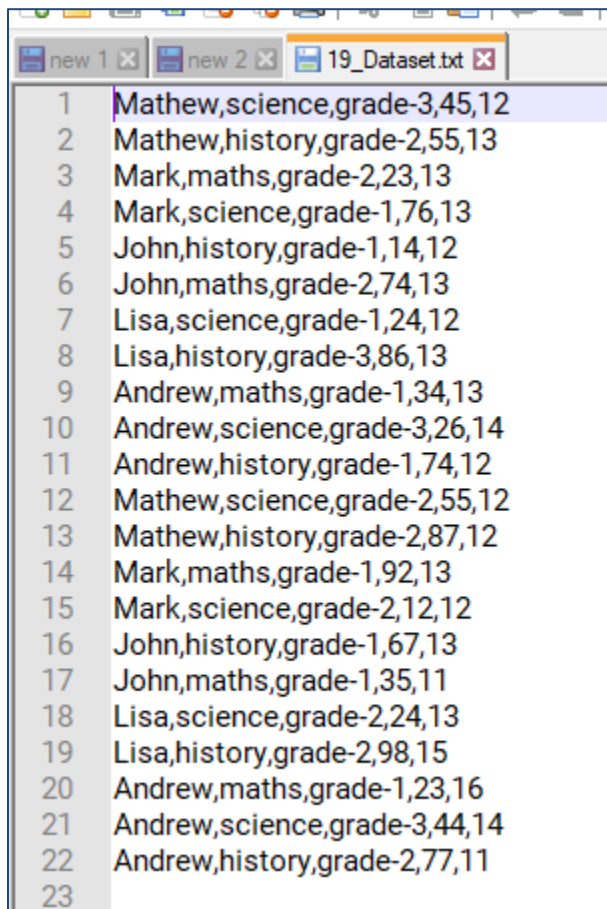


TASK 2:

Problem Statement – 1:

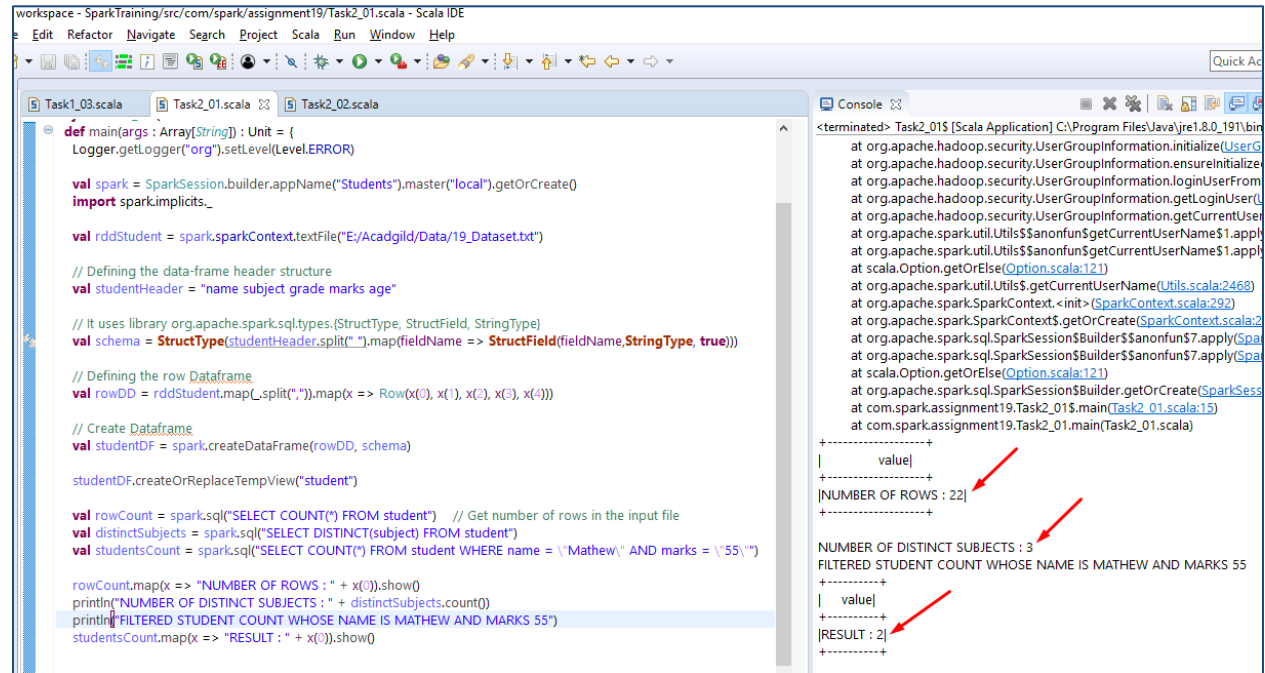
1. Read the text file, and create a tupled rdd.
2. Find the count of total number of rows present.
3. What is the distinct number of subjects present in the entire school
4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

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



1	Mathew,science,grade-3,45,12
2	Mathew,history,grade-2,55,13
3	Mark,maths,grade-2,23,13
4	Mark,science,grade-1,76,13
5	John,history,grade-1,14,12
6	John,maths,grade-2,74,13
7	Lisa,science,grade-1,24,12
8	Lisa,history,grade-3,86,13
9	Andrew,maths,grade-1,34,13
10	Andrew,science,grade-3,26,14
11	Andrew,history,grade-1,74,12
12	Mathew,science,grade-2,55,12
13	Mathew,history,grade-2,87,12
14	Mark,maths,grade-1,92,13
15	Mark,science,grade-2,12,12
16	John,history,grade-1,67,13
17	John,maths,grade-1,35,11
18	Lisa,science,grade-2,24,13
19	Lisa,history,grade-2,98,15
20	Andrew,maths,grade-1,23,16
21	Andrew,science,grade-3,44,14
22	Andrew,history,grade-2,77,11
23	

Program & Execution:



The screenshot displays the Scala IDE interface with a workspace containing three files: Task1_03.scala, Task2_01.scala, and Task2_02.scala. The main editor shows the code for Task2_01.scala, which is a Spark application. The code defines a main function that sets up a Spark session, reads a text file, defines a schema, creates a DataFrame, and executes SQL queries to count rows and distinct subjects, with a filter for a specific student and mark.

```
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")

  val rowCount = spark.sql("SELECT COUNT(*) FROM student") // Get number of rows in the input file
  val distinctSubjects = spark.sql("SELECT DISTINCT(subject) FROM student")
  val studentsCount = spark.sql("SELECT COUNT(*) FROM student WHERE name = 'Mathew' AND marks = '55'")

  rowCount.map(x => "NUMBER OF ROWS : " + x(0)).show()
  println("NUMBER OF DISTINCT SUBJECTS : " + distinctSubjects.count())
  println("FILTERED STUDENT COUNT WHOSE NAME IS MATHEW AND MARKS 55")
  studentsCount.map(x => "RESULT : " + x(0)).show()
}
```

The console output shows the execution of the program, including the Spark application's startup logs and the results of the SQL queries. Red arrows point from the console output to the corresponding lines in the code. The output shows the number of rows (22), the number of distinct subjects (3), and the filtered student count (2).

```
<terminated> Task2_01$ [Scala Application] C:\Program Files\Java\jre1.8.0_191\bin
at org.apache.hadoop.security.UserGroupInformation.initialize(UserG
at org.apache.hadoop.security.UserGroupInformation.ensureInitialize
at org.apache.hadoop.security.UserGroupInformation.loginUserFrom
at org.apache.hadoop.security.UserGroupInformation.getCurrentUser
at org.apache.spark.util.Utils$$anonfun$getCurrentUserName$1.appl
at org.apache.spark.util.Utils$$anonfun$getCurrentUserName$1.appl
at scala.Option.getOrElse(Option.scala:121)
at org.apache.spark.util.Utils$.getCurrentUserName(Utils.scala:2468)
at org.apache.spark.SparkContext.<init> (SparkContext.scala:292)
at org.apache.spark.SparkContext$.getOrCreate(SparkContext.scala:2
at org.apache.spark.sql.SparkSession$Builder$$anonfun$7.apply(Spa
at org.apache.spark.sql.SparkSession$Builder$$anonfun$7.apply(Spa
at scala.Option.getOrElse(Option.scala:121)
at org.apache.spark.sql.SparkSession$Builder.getOrCreate(SparkSess
at com.spark.assignment19.Task2_01$.main(Task2_01.scala:15)
at com.spark.assignment19.Task2_01.main(Task2_01.scala)

+-----+
| value|
+-----+
|NUMBER OF ROWS : 22|
+-----+

NUMBER OF DISTINCT SUBJECTS : 3
FILTERED STUDENT COUNT WHOSE NAME IS MATHEW AND MARKS 55
+-----+
| value|
+-----+
|RESULT : 2|
+-----+
```

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
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")

    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| 65.666666666666667|
| 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| 65.666666666666667|
| 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 ✕

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")

    println("Average score per student_name across all grades is same as average score per student_name per grade")
    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 [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
```