ASSIGNMENT 19.1

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.

Solution#1

```
scala> val fileRDD = sc.textFile("file:///home/acadgild/story.txt")
fileRDD: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/story.txt MapPartitionsRDD[39] at textFile at <console>:24

# 1. To count number of lines in the file, we use count method of the RDD
scala> fileRDD.count
res19: Long = 22
scala> fileRDD.flatMap(x=>x.split(" "))
res21: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[40] at flatMap at <console>:27

# 2. To count number of words in the file, we can use flatMap to put all the words into a single Array and then use count to get the number of words in the file

scala> fileRDD.flatMap(x=>x.split(" ")).collect
res22: Array[String] = Array(Once, upon, a, time,, there, lived, a, very, cunning, and, mischievous, fox., He, used, to, speak, to, other, animals, sweetly, and, gain, their, trust,, before, playing, tricks, on, them., "", One, day, the, fox, met, a, stork., He, befriended, the, stork, and, acted, like, a, very, good, friend., Soon,, he, invited, the, stork, to, have, a, feast, with, him., The, stork, happily, accepted, the, invitation., "", The, day, of, the, feast, came,, and, the, stork, went, to, the, foxs, house., To, her, surprise, and, disappointment,, the, fox, said, that, he, could, not, make, a, big, feast, as, promised,, and, just, offered, some, soup., When, he, brought, the, soup, out, of, the, kitchen,, the, stork, saw, that, it, was, in, a, shallow, bowl!, "", The, poor,...
scala> fileRDD.flatMap(x=>x.split(" ")).count
res23: Long = 342


# 3. To count number of words in the file, we can use flatMap to put all the words into a single Array and then use count to get the number of words in the file. We use the "-" as the separator and use in string split method. To test we create a simple file like this and then use the below program to count the number of words

[acadgild.mmisra ~]$ cat story1.txt
This-a-test-program-to-check-use-of-a-different-separator
This-a-test-program-to-check-use-of-a-different-separator
This-a-test-program-to-check-use-of-a-different-separator
This-a-test-program-to-check-use-of-a-different-separator
This-a-test-program-to-check-use-of-a-different-separator
```

```
scala> val fileRDD = sc.textFile("file:///home/acadgild/story1.txt")
fileRDD: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/story1.txt MapPartitionsRDD[44] at
textFile at <console>:24

# Number of lines in the file
scala> fileRDD.count
res24: Long = 5

scala> fileRDD.flatMap(x=>x.split("-")).collect
res25: Array[String] = Array(This, a, test, program, to, check, use, of, a, different, separator, This, a, test,
program, to, check, use, of, a, different, separator, This, a, test, program, to, check, use, of, a, different,
separator, This, a, test, program, to, check, use, of, a, different, separator, This, a, test, program, to,
check, use, of, a, different, separator)

# Number of words in the file

scala> fileRDD.flatMap(x=>x.split("-")).count
res26: Long = 55
```

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

Reading the data set file which looks like this

```
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
…….

# We read the file into a the file RDD
scala> val fileRDD = sc.textFile("file:///home/acadgild/dataset-19.1.txt")
fileRDD: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/dataset-19.1.txt
MapPartitionsRDD[399] at textFile at <console>:24

# We then split each line based on comma and create tuple for each row. We also specify the data type
for each field as we map. We create a RDD of tuples
```

```
scala> val tRDD =
    | fileRDD.map(x=>x.split(",")).map(y=>(y(0),y(1),y(2),y(3).toInt,y(4).toInt))
tRDD: org.apache.spark.rdd.RDD[(String, String, String, Int, Int)] = MapPartitionsRDD[401] at map at
<console>:27

# Number of rows in the file

scala> tRDD.count
res184: Long = 22

#to find the distinct number of subjects in the school, we create the RDD of only subjects and then find
distinct
// x._2 is the subject name, we can find unique. Distinct number of subjects
// present in the school

scala> tRDD.map(x=>x._2).distinct.collect
res185: Array[String] = Array(history, science, maths)

scala> tRDD.map(x=>x._2).distinct.count
res186: Long = 3


#To find  the count of the number of students in the school, whose name is Mathew and
#marks is 55, we filter the RDD where name is Mathew and marks is 55.

scala> tRDD.filter(x=>(x._4==55) && (x._1=="Mathew") ).collect
res0: Array[(String, String, String, Int, Int)] = Array((Mathew,history,grade-2,55,13),
(Mathew,science,grade-2,55,12))

scala> tRDD.filter(x=>(x._4==55) && (x._1=="Mathew") ).count
res1: Long = 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?

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

#To fine the count of students per grade in the school map the RDD for to include only the grades and then do count by value

```
scala> //count of students per grade, x._3 is grade

scala> tRDD.map(x=>x._3).countByValue
res2: scala.collection.Map[String,Long] = Map(grade-2 -> 9, grade-3 -> 4, grade-1 -> 9)
```

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

To distinguish Mathew in one grade from another and to find average for each student, we transform the RDD and make a Paired RDD where first name(key) is name+grade and the value is marks obtained.

```
scala> val pRDD = tRDD.map(x=>((x._1+"-"+x._3)->x._4)).collect
pRDD: Array[(String, Int)] = Array((Mathew-grade-3,45), (Mathew-grade-2,55), (Mark-grade-2,23), (Mark-grade-1,76), (John-grade-1,14), (John-grade-2,74), (Lisa-grade-1,24), (Lisa-grade-3,86), (Andrew-grade-1,34), (Andrew-grade-3,26), (Andrew-grade-1,74), (Mathew-grade-2,55), (Mathew-grade-2,87), (Mark-grade-1,92), (Mark-grade-2,12), (John-grade-1,67), (John-grade-1,35), (Lisa-grade-2,24), (Lisa-grade-2,98), (Andrew-grade-1,23), (Andrew-grade-3,44), (Andrew-grade-2,77))

scala> val pRDD = tRDD.map(x=>((x._1+"-"+x._3)->x._4))
pRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at map at <console>:28
```

this will calculate sum and count for each student

```
scala> val aRDD = pRDD.mapValues(a=>(a,1))
aRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[13] at mapValues at <console>:30

scala> aRDD.collect
res10: Array[(String, (Int, Int))] = Array((Mathew-grade-3,(45,1)), (Mathew-grade-2,(55,1)), (Mark-grade-2,(23,1)), (Mark-grade-1,(76,1)), (John-grade-1,(14,1)), (John-grade-2,(74,1)), (Lisa-grade-1,(24,1)), (Lisa-grade-3,(86,1)), (Andrew-grade-1,(34,1)), (Andrew-grade-3,(26,1)), (Andrew-grade-1,(74,1)), (Mathew-grade-2,(55,1)), (Mathew-grade-2,(87,1)), (Mark-grade-1,(92,1)), (Mark-grade-2,(12,1)), (John-grade-
```

1,(67,1)), (John-grade-1,(35,1)), (Lisa-grade-2,(24,1)), (Lisa-grade-2,(98,1)), (Andrew-grade-1,(23,1)), (Andrew-grade-3,(44,1)), (Andrew-grade-2,(77,1)))

scala>

scala> val cRDD=aRDD.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
cRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[14] at reduceByKey at <console>:32

scala> cRDD.collect
res11: Array[(String, (Int, Int))] = Array((Mathew-grade-3,(45,1)), (Mark-grade-2,(35,2)), (Lisa-grade-2,(122,2)), (Andrew-grade-2,(77,1)), (John-grade-2,(74,1)), (Mark-grade-1,(168,2)), (Mathew-grade-2,(197,3)), (John-grade-1,(116,3)), (Lisa-grade-3,(86,1)), (Andrew-grade-3,(70,2)), (Andrew-grade-1,(131,3)), (Lisa-grade-1,(24,1)))

scala>

To calculate average, we use the mapValues where we divide the sum of the marks with the count of subjects

scala> val avgRDD= cRDD.mapValues(a=>(a._1).toDouble/a._2)
avgRDD: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[15] at mapValues at <console>:34

scala> avgRDD.collect
res12: Array[(String, Double)] = Array((Mathew-grade-3,45.0), (Mark-grade-2,17.5), (Lisa-grade-2,61.0), (Andrew-grade-2,77.0), (John-grade-2,74.0), (Mark-grade-1,84.0), (Mathew-grade-2,65.66666666666667), (John-grade-1,38.666666666666664), (Lisa-grade-3,86.0), (Andrew-grade-3,35.0), (Andrew-grade-1,43.666666666666664), (Lisa-grade-1,24.0))

scala>

Following is the average of each sudent

scala> avgRDD.map(a=> println("name"+a._1 + " avg marks=" + a._2)).collect
nameMathew-grade-3 avg marks=45.0
nameMark-grade-2 avg marks=17.5
nameLisa-grade-2 avg marks=61.0
nameAndrew-grade-2 avg marks=77.0
nameJohn-grade-2 avg marks=74.0
nameMark-grade-1 avg marks=84.0
nameMathew-grade-2 avg marks=65.66666666666667
nameJohn-grade-1 avg marks=38.666666666666664
nameLisa-grade-3 avg marks=86.0
nameAndrew-grade-3 avg marks=35.0
nameAndrew-grade-1 avg marks=43.666666666666664
nameLisa-grade-1 avg marks=24.0
res13: Array[Unit] = Array((), (), (), (), (), (), (), (), (), (), (), ())

#We can do the above calculations in a single Scala statement as well as shown below.

```
tRDD.map(x=>((x._1+"-"+x._3)->x._4))
.mapValues(a=>(a,1))
.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
.mapValues(a=>(a._1).toDouble/a._2).collect
```

```
scala> tRDD.map(x=>((x._1+"-"+x._3)->x._4))
res6: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[8] at map at <console>:29

scala> .mapValues(a=>(a,1))
res7: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[9] at mapValues at <console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
res8: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[10] at reduceByKey at <console>:33

scala> .mapValues(a=>(a._1).toDouble/a._2).collect
res9: Array[(String, Double)] = Array((Mathew-grade-3,45.0), (Mark-grade-2,17.5), (Lisa-grade-2,61.0),
(Andrew-grade-2,77.0), (John-grade-2,74.0), (Mark-grade-1,84.0), (Mathew-grade-
2,65.66666666666667), (John-grade-1,38.666666666666664), (Lisa-grade-3,86.0), (Andrew-grade-
3,35.0), (Andrew-grade-1,43.666666666666664), (Lisa-grade-1,24.0))
```

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

#To find average marks for each subject across all grades, we use the same scheme as above. The only difference is that we create paired RDD with subject as the key and the marks as the value

```
tRDD.map(x=>(x._2->x._4))
.mapValues(x=>(x,1))
.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
.mapValues(a=>(a._1).toDouble/a._2).collect
```

```
scala> tRDD.map(x=>(x._2->x._4))
res10: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at map at <console>:29

scala> .mapValues(x=>(x,1))
res11: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[13] at mapValues at
<console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
res12: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[14] at reduceByKey at <console>:33

scala> .mapValues(a=>(a._1).toDouble/a._2).collect
res13: Array[(String, Double)] = Array((history,69.75), (science,38.25), (maths,46.833333333333336))
```

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

#To find average marks for each subject per grade we make the pair RDD where the key=subject+grade and the value is marks. We then find the average in the same way as above

```
scala> tRDD.map(x=>(x._3+"-"+x._2->x._4))
res15: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[16] at map at <console>:29

scala> .mapValues(x=>(x,1))
res16: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[17] at mapValues at
<console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
res17: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[18] at reduceByKey at <console>:33

scala> .mapValues(a=>(a._1).toDouble/a._2).collect
res18: Array[(String, Double)] = Array((grade-3-history,86.0), (grade-3-science,38.333333333333336),
(grade-1-science,50.0), (grade-1-history,51.666666666666664), (grade-2-maths,48.5), (grade-2-
science,30.333333333333332), (grade-1-maths,46.0), (grade-2-history,79.25))
```

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

#To find this we filter the original RDD only for grade-2 and then find the student average as in above and then filter again for the records having average greater than 50

```
scala> tRDD.filter(x=>(x._3=="grade-2")).map(x=>(x._1->x._4))
res19: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[21] at map at <console>:29

scala> .mapValues(x=>(x,1))
res20: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[22] at mapValues at
<console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
res21: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[23] at reduceByKey at <console>:33

scala> .mapValues(a=>(a._1).toDouble/a._2)
res22: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[24] at mapValues at
<console>:35

scala> .filter(_._2>50).count
res23: Long = 4
```

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

Solution

##We first calculate the average score per student name across all grades and then find average marks for each student and each grade and then find the intersection of these RDDs

```
scala> // average marks for each student across all grades

scala> val fRDD =

    | fileRDD.map(x=>x.split(",")).map(y=>(y(0),y(1),y(2),y(3).toDouble,y(4).toInt))

fRDD: org.apache.spark.rdd.RDD[(String, String, String, Double, Int)] = MapPartitionsRDD[313] at map at <console>:27

scala> val tRDD = fRDD.map(x=>(x._1->x._4))

tRDD: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[314] at map at <console>:28

scala> .mapValues(x=>(x,1.toDouble))

res146: org.apache.spark.rdd.RDD[(String, (Double, Double))] = MapPartitionsRDD[315] at mapValues at <console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))

res147: org.apache.spark.rdd.RDD[(String, (Double, Double))] = ShuffledRDD[316] at reduceByKey at <console>:33

scala> .mapValues(a=>(a._1/a._2).toDouble)

res148: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[317] at mapValues at <console>:35

scala> // for grade-1

scala> val tRDD1 =

    | fRDD.filter(x=>(x._3=="grade-1")).map(x=>(x._1->x._4))

tRDD1: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[319] at map at <console>:29

scala> .mapValues(x=>(x,1.toDouble))

res149: org.apache.spark.rdd.RDD[(String, (Double, Double))] = MapPartitionsRDD[320] at mapValues at <console>:31
```

```
scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))

res150: org.apache.spark.rdd.RDD[(String, (Double, Double))] = ShuffledRDD[321] at reduceByKey at <console>:33

scala> .mapValues(a=>a._1/a._2)

res151: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[322] at mapValues at <console>:35


scala> // for grade-2

scala> val tRDD2 =

     | fRDD.filter(x=>(x._3=="grade-2")).map(x=>(x._1->x._4))

tRDD2: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[324] at map at <console>:29

scala> .mapValues(x=>(x,1.toDouble))

res152: org.apache.spark.rdd.RDD[(String, (Double, Double))] = MapPartitionsRDD[325] at mapValues at <console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))

res153: org.apache.spark.rdd.RDD[(String, (Double, Double))] = ShuffledRDD[326] at reduceByKey at <console>:33

scala> .mapValues(a=>a._1/a._2)

res154: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[327] at mapValues at <console>:35

scala> // for grade-3

scala> val tRDD3 =

     | fRDD.filter(x=>(x._3=="grade-3")).map(x=>(x._1->x._4))

tRDD3: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[329] at map at <console>:29

scala> .mapValues(x=>(x,1.toDouble))

res155: org.apache.spark.rdd.RDD[(String, (Double, Double))] = MapPartitionsRDD[330] at mapValues at <console>:31

scala> .reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))

res156: org.apache.spark.rdd.RDD[(String, (Double, Double))] = ShuffledRDD[331] at reduceByKey at <console>:33

scala> .mapValues(a=>a._1/a._2)
```

res157: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[332] at mapValues at <console>:35

scala>

now find the intersection. We see that the resultant set is empty

scala> tRDD.intersection(tRDD1).intersection(tRDD2).intersection(tRDD3).collect

res158: Array[(String, Double)] = Array()