Assignment Day 19

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.

Ans:

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

Note:(Since the question was ambiguous as number of rows means the count as well as the lines in file both. so providing both the solutions**)**

sc.textFile("/dataset/input.txt**").foreach(println)**

(using scpark context sc reading input file at **HDFS** location /dataset/input.txt & printing each line **)**

sc.textFile("/dataset/input.txt**").count**

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

ScreenShot:

Input File:

```
[acadgild@localhost spark]$ hadoop fs -cat /dataset/input.txt
18/08/30 11:28:21 WARN util.NativeCodeLoader: Unable to load n
e applicable
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost spark]$
```

Sahil Sahay

```
scala> sc.textFile("/dataset/input.txt").foreach(println)
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line
Hi this is a line

scala> sc.textFile("/dataset/input.txt").count
res225: Long = 7
```

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

Ans:

sc.textFile("/dataset/input.txt**").flatMap(x => x.split(" ")).map(**x => (x,1)**).reduceByKey(**(x,y) => x+y**).foreach(println)**

Explanation:

creating RDD from a text file over **HDFS.** 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.

ScreenShot:

```
scala> sc.textFile("/dataset/input.txt").flatMap(x => x.split(" ")).map(x => (x,1)).reduceByKey((x,y) => x+y).foreach(println)
(this,7)
(is,7)
(a,7)
(line,7)
(Hi,7)
```

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.

Ans:

sc.textFile("/dataset/input.txt").**flatMap**(x => x.**split**("-")).**map**(x => (x,1)).**reduceByKey**((x,y) => x+y).foreach(**println**)

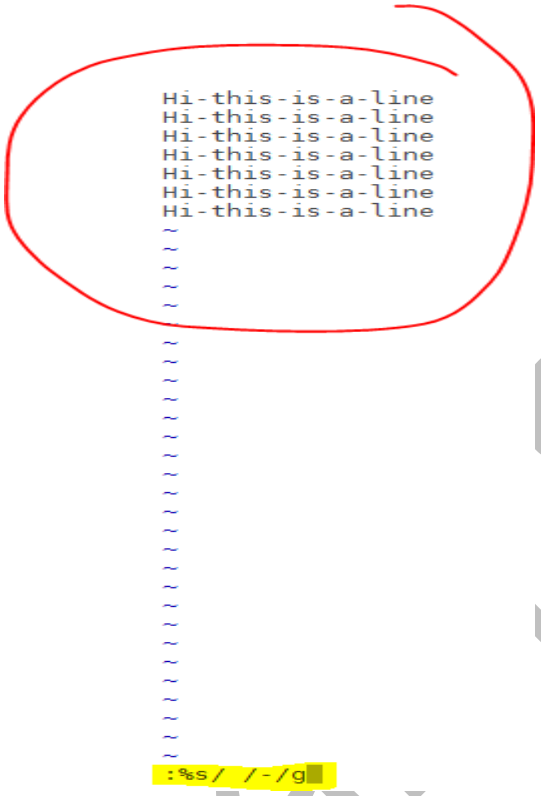
Sahil Sahay

Explanation:

creating RDD from a text file over **HDFS.** 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.

ScreenShot:

Input Data




```
[acadgild@localhost spark]$ hadoop fs -cat /dataset/input.txt
18/08/30 11:49:58 WARN util.NativeCodeLoader: Unable to load na
e applicable
Hi-this-is-a-line
Hi-this-is-a-line
Hi-this-is-a-line
Hi-this-is-a-line
Hi-this-is-a-line
Hi-this-is-a-line
Hi-this-is-a-line
You have new mail in /var/spool/mail/acadgild
```

```
scala> sc.textFile("/dataset/input.txt").flatMap(x => x.split("-")).map(x => (x,1)).reduceByKey((x,y) => x+y).foreach(println)
(this,7)
(is,7)
(a,7)
(line,7)
(Hi,7)
```

Sahay

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

Ans:

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

sc.**textFile**("/spark/19_Dataset.txt").**map**(x => (x.**split**(",")(**0**),x.**split**(",")(**1**),x.**split**(",")(**2**),x.**split**(",")(**3**).toInt,x.**split**(",")(**4**).**toInt**)). **foreach**(println)

Explanation:

creating tuple RDD from a text file over **HDFS.** 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

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").map(x => (x.split(",")(0),x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt,x.split(","
)(4).toInt)).collect.foreach(println)
(Mathew,science,grade-3,45,12)
(Mathew,history,grade-2,55,13)
(Mark,maths,grade-2,23,13)
(Mark,science,grade-1,76,13)
(John,history,grade-1,14,12)
(John,maths,grade-2,74,13)
(Lisa,science,grade-1,24,12)
(Lisa,history,grade-3,86,13)
(Andrew,maths,grade-1,34,13)
(Andrew,science,grade-3,26,14)
(Andrew,history,grade-1,74,12)
(Mathew,science,grade-2,55,12)
(Mathew,history,grade-2,87,12)
(Mark,maths,grade-1,92,13)
(Mark,science,grade-2,12,12)
(John,history,grade-1,67,13)
(John,maths,grade-1,35,11)
(Lisa,science,grade-2,24,13)
(Lisa,history,grade-2,98,15)
(Andrew,maths,grade-1,23,16)
(Andrew,science,grade-3,44,14)
(Andrew,history,grade-2,77,11)
```

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**count**

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

Sahay

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").count
res234: Long = 22

scala>
```

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**map**(x => (x.**split**(",")(1))).**distinct.collect**

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.

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").map(x => (x.split(",")(1))).distinct.collect
res235: Array[String] = Array(maths, history, science)
```

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**filter**(x => (x.**split**(",")(0).**equals**("Mathew") && x.**split**(",")(3).**toInt.equals**(55))).**count**

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

Sahil Sahay

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

ScreenShot:

```
[acadgild@localhost ~]$ hadoop fs -cat /spark/19_Dataset.txt | sort -t"," -k1
18/08/30 09:56:26 WARN util.NativeCodeLoader: Unable to load native-hadoop lil
e applicable
Andrew,history,grade-1,74,12
Andrew,history,grade-2,77,11
Andrew,maths,grade-1,23,16
Andrew,maths,grade-1,34,13
Andrew,science,grade-3,26,14
Andrew,science,grade-3,44,14
John,history,grade-1,14,12
John,history,grade-1,67,13
John,maths,grade-1,35,11
John,maths,grade-2,74,13
Lisa,history,grade-2,98,15
Lisa,history,grade-3,86,13
Lisa,science,grade-1,24,12
Lisa,science,grade-2,24,13
Mark,maths,grade-1,92,13
Mark,maths,grade-2,23,13
Mark,science,grade-1,76,13
Mark,science,grade-2,12,12
Mathew,history,grade-2,55,13
Mathew,history,grade-2,87,12
Mathew,science,grade-2,55,12
Mathew,science,grade-3,45,12
[acadgild@localhost ~]$
```

```
scala> sc.textFile("/spark/19_Dataset.txt").filter(x => (x.split(",")(0).equals("Mathew") && x.split(",")(3).toInt.equals(55))).count
res236: Long = 2
```

Task 2:

Problem Statement 2:

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**map**(x => ((x.**split**(",")(2)),1)).**reduceByKey**((x,y) => x+y).foreach(**println**)

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

or(The other method to perform this task**)**

Sahi

sc.textFile("/spark/19_Dataset.txt").**map**(x => (x.**split**(",")(2))).**countByValue**().foreach(**println**)

Explanation: Performing similar operation as above but in this case using builtin method countByValue, which counts distinct value within the RDD without the need of mapping the value one to one.

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(2)),1)).reduceByKey((x,y) => x+y).foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)

scala> sc.textFile("/spark/19_Dataset.txt").map(x => (x.split(",")(2))).countByValue().foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)
```

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**map**(x => ((x.**split**(",")(0),x.**split**(",")(2)),x.**split**(",")(3)**toDouble**)).**groupByKey**()

.**map**(x => (x._1,x._2.**sum**/x._2.**size**)).**collect**

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.

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3)toDouble)).groupByKey()
warning: there was one feature warning; re-run with -feature for details
res241: org.apache.spark.rdd.RDD[((String, String), Iterable[Double])] = ShuffledRDD[485] at groupByKey at <console>:25

scala> .map(x => (x._1,x._2.sum/x._2.size)).foreach(println)
((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)
```

Sahil Saha

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**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**)

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.

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(1)),x.split(",")(3)toDouble)).groupByKey()
warning: there was one feature warning; re-run with -feature for details
res243: org.apache.spark.rdd.RDD[((String, String), Iterable[Double])] = ShuffledRDD[490] at groupByKey at <console>:25

scala> .map(x => (x._1,x._2.sum/x._2.size)).foreach(println)
((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)
```

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**map**(x => ((x.**split**(",")(**0**),x.**split**(",")(**1**),x.**split**(",")(**2**)),x.**split**(",")(**3**)**toDouble**)).**groupBy Key**().map(x => (x._1,x._2.**sum**/x._2.**size**)).**foreach**(println)

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

Sa... ahay

fetching the count of marks as x._2.size to divide the sum & find the average on it. Here grouping key is name, subject & grade.

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(1),x.split(",")(2)),x.split(",")(3)toDouble)).group
ByKey()
warning: there was one feature warning; re-run with -feature for details
res245: org.apache.spark.rdd.RDD[((String, String, String), Iterable[Double])] = ShuffledRDD[495] at groupByKey at <console>:25

scala> .map(x => (x._1,x._2.sum/x._2.size)).foreach(println)
((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)
```

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

Ans:

sc.textFile("/spark/19_Dataset.txt").**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)).**count**()

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.

ScreenShot:

```
scala> sc.textFile("/spark/19_Dataset.txt").filter(x => (x.split(",")(2)).equals("grade-2")).map(x => ((x.split(",")(0),x.split(",")(3)
toDouble)))
warning: there was one feature warning; re-run with -feature for details
res18: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[69] at map at <console>:25

scala> .groupByKey().map(x => (x._1,x._2.sum/x._2.size)).filter(x => (x._2 > 50)).foreach(println)
(Andrew,77.0)
(Mathew,65.66666666666667)
(John,74.0)
(Lisa,61.0)

scala> sc.textFile("/spark/19_Dataset.txt").filter(x => (x.split(",")(2)).equals("grade-2")).map(x => ((x.split(",")(0),x.split(",")(3)
toDouble)))
warning: there was one feature warning; re-run with -feature for details
res20: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[76] at map at <console>:25

scala> .groupByKey().map(x => (x._1,x._2.sum/x._2.size)).filter(x => (x._2 > 50)).count()
res21: Long = 4
```

Sahil Sar

Task 2

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

Ans:

```
val rdd1 = sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(0)),x.split(",")(3)toDouble)).groupByKey().map(x => (x._1,x._2.sum/x._2.size))
```

```
sc.textFile("/spark/19_Dataset.txt").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)
```

```
val rdd2 = sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3)toDouble)).groupByKey().map(x => (x._1._1,x._2.sum/x._2.size))
```

```
rdd1.intersection(rdd2).collect
```

Explanation:

Creating two RDD's from textFile. Selecting name as key in **rdd1** & (name,grade) as key in **rdd2**, marks as value. Converting marks into double datatype just to avoid any truncation after decimal. groupByKey returns a tuple of ((name), marks) in **rdd1** & a tuple of ((name, grade), marks) in **rdd2**. Applying map function & selecting ((name), marks) in **rdd1** & a tuple of ((name, grade), marks) in **rdd2** 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 name in **rdd1** & a tuple of (name, grade) in **rdd2**.

Note : There is no student whose average score across grades is same as average score per grade. Manually verified, Since was not getting blank result

Sahil Sahay

Mathew(Across Grades) - (45 + 55 + 55 + 87)/4 = **60.5**

Mathew (grade_2) = **65.66**

Methew (grade_3) = **45**

Mark (Across Grades) - (23+76+ 92 +12) / 4 = **50.75**

Mark (grade_1) = (76+92)/2 = **84**

Mark (grade_2) = (23+12)/2 = **17.50**

John (Across Grades) - (14+74+67+35) / 4 = **47.59**

John (grade_1)= (14+67+35)/3 = **38.66**

John (grade_2) = **74**

Lisa (Across Grades) - (24+86+24+98)/4 = **58**

Lisa(grade_1) = **24**

Lisa(grade_2) = (24+98)/2 = **61**

Lisa (grade_3) = **86**

Andrew (Across Grades) - (34+26+74 +23 +44 +77)/6 = **46.33**

Andrew (grade_1) - (34+74+23)/3 = **43.66**

Andrew (grade_2) - **77**

Andrew (grade_3) - (26+44)/2 = **35**.

Mathew(Across Grades) - (45 + 55 + 55 + 87)/4 = **60.5**

Mathew (grade_2) = **65.66**

Methew (grade_3) = **45**

Mark (Across Grades) - (23+76+ 92 +12) / 4 = **50.75**

Mark (grade_1) = (76+92)/2 = **84**

Mark (grade_2) = (23+12)/2 = **17.50**

Sahil Sahay

John (Across Grades) - (14+74+67+35) / 4 = **47.59**

John (grade_1) - (14+67+35)/3 = **38.66**

John (grade_2) - **74**

Lisa (Across Grades) - (24+86+24+98)/4 = **58**

Lisa(grade_1) - **24**

Lisa(grade_2) - (24+98)/2 = **61**

Lisa (grade_3) - **86**

Andrew (Across Grades) - (34+26+74 +23 +44 +77)/6 = **46.33**

Andrew (grade_1) - (34+74+23)/3 = **43.66**

Andrew (grade_2) - **77**

Andrew (grade_3) - (26+44)/2 = **35**

Sahay

ScreenShot:

```
scala> rdd1.collect
res253: Array[(String, Double)] = Array((Mark,50.75), (Andrew,46.333333333333336), (Mathew,60.5), (John,47.5), (Lisa,58.0))

scala>

scala> sc.textFile("/spark/19_Dataset.txt").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)
warning: there was one feature warning; re-run with -feature for details
((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)

scala>

scala> val rdd2 = sc.textFile("/spark/19_Dataset.txt").map(x => ((x.split(",")(0),x.split(",")(2)),x.split(",")(3)toDouble)).groupByKey
().map(x => (x._1,x._2.sum/x._2.size))
warning: there was one feature warning; re-run with -feature for details
rdd2: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[560] at map at <console>:24

scala> rdd2.collect
res255: Array[(String, Double)] = Array((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))

scala>

scala> rdd1.intersection(rdd2).collect
res256: Array[(String, Double)] = Array()
```

********** **End** **********