Big data Hadoop - Assignment 7-RDD Deep Dive

Given a dataset of college students as a text file (name, subject, grade, marks) :

Dataset .txt

Task 1

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

```
// Create a SparkContext using every core of the local machine
val sc = new SparkContext("local[*]", "WordCount")

// Read each line of my book into an RDD
val newdataset = sc.textFile("E:/19_dataset.txt")

   //Task 1.1 Number of lines / Rows in the file

//Printing each line of data from document

newdataset.foreach(w=>println(w))

//Printing number of Rows in Dataset

val noOfLines = newdataset.count()

//printing total count of rows in document

println("no. Of lines present in the file: " + noOfLines)
```

OUTPUT :

```
        at com.rddtest7.assignment_7$.main(assignment_7.scala:16)
        at com.rddtest7.assignment_7.main(assignment_7.scala)
Mathew,science,grade-3,45,12
Mathew,science,grade-2,55,12
Mathew,history,grade-2,87,12
Mathew,history,grade-2,55,13
Mark,maths,grade-1,92,13
Mark,maths,grade-2,23,13
Mark,science,grade-2,12,12
Mark,science,grade-1,76,13
John,history,grade-1,67,13
John,history,grade-1,14,12
John,maths,grade-1,35,11
John,maths,grade-2,74,13
Lisa,science,grade-2,24,13
Lisa,science,grade-1,24,12
Lisa,history,grade-2,98,15
Lisa,history,grade-3,86,13
Andrew,maths,grade-1,23,16
Andrew,maths,grade-1,34,13
Andrew,science,grade-3,44,14
Andrew,science,grade-3,26,14
Andrew,history,grade-2,77,11
Andrew,history,grade-1,74,12
no. Of lines present in the file: 22
```

Big data Hadoop - Assignment 7-RDD Deep Dive

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

```
//task 1.2 Write a program to read a text file and print the number of words in the document.
//Create new RDD using flatmap method and split on ','
val splittedRDDs = newdataset.flatMap(f=>f.split(',')) //Here it will return RDD which has combination of wor
//function to verify if the passed string is numeric only or word
def isWord(a:String):Boolean={
  val numberPattern: Regex = "^[0-9]*$".r
  numberPattern.findFirstMatchIn(a) match {
    case  Some(_) =>
      return false
    case None =>
      return true
  }
}
//Looking at the dataset we must get first three elements which satisfies the definition of word here
val wordOnly = splittedRDDs.filter(f=>isWord(f))
wordOnly.foreach(x=>println(x))
//Now we can use the count function to get the number of words in the document
val countOfWords = wordOnly.count()
println("no. Of words present in the document: " + countOfWords)
```

OUTPUT :

Splitted words :

```
Mathew
science
Mathew
grade-3
science
grade-2
Mathew
Mathew
history
history
grade-2
grade-2
Mark
Mark
maths
maths
grade-2
grade-1
Mark
Mark
science
science
grade-1
grade-2
John
John
history
history
grade-1
grade-1
John
maths
grade-1
John
maths
grade-2
Lisa
```

Total numbers of words

```
grade-1
no. Of words present in the document: 66
```

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 Approach –

   1. Split the newdataset RDD using character '-'
   2. Here words will be having both only numeric and alpha numeric values
   3. Used flatmap operation to get RDD splitted by '-'
   4. Then mapped the newly created RDD with counter 1

```
//Task 1.3 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.

val splittedRDDs2 = newdataset.flatMap(f=>f.split('-'))

val AfterSplitting = splittedRDDs2.map(x=>(x,1)).reduceByKey(_+_)

AfterSplitting.foreach(x=>println(x))

val countAfterSplitting = AfterSplitting.count()

println("no. Of words present in the document after separating with -: " + countAfterSplitting)
```

OUTPUT :

```
(1,23,16,1)
(1,34,13,1)
(Mark,maths,grade,2)
(Lisa,history,grade,2)
(1,35,11,1)
(2,55,13,1)
(3,86,13,1)
(2,87,12,1)
(Mathew,history,grade,2)
(1,24,12,1)
(John,maths,grade,2)
(1,67,13,1)
(Andrew,science,grade,2)
(3,45,12,1)
(Lisa,science,grade,2)
(2,23,13,1)
(John,history,grade,2)
(2,98,15,1)
(3,44,14,1)
(1,74,12,1)
(2,77,11,1)
(2,74,13,1)
(Mark,science,grade,2)
(2,55,12,1)
(3,26,14,1)
(1,14,12,1)
(2,12,12,1)
(Mathew,science,grade,2)
(1,92,13,1)
(Andrew,maths,grade,2)
(1,76,13,1)
(2,24,13,1)
(Andrew,history,grade,2)
no. Of words present in the document after separating with -: 33
```

Task 2

Problem Statement 1:

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

Solution Approach –

1. We have already read file in RDD named as newdataset
2. Create new tupled RDD by using map operation where every element of row is separated by ‘,’

```
//Task 2

//Problem Statement 2.1.1 : Read the text file, and create a tupled rdd.

val tupledRDD = newdataset.map(x=>x.split(","))

//Printing tupled RDD

tupledRDD.collect().foreach(row => println(row.mkString(",")))
```

OUTPUT :

```
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.

```
//Problem Statement 2.1.2 : Find the count of total number of rows present.

val countResult = tupledRDD.count()

println("no. Of rows present in tupled RDD -: " + countResult)
```

OUTPUT:

```
no. Of rows present in tupled RDD -: 22
```

Big data Hadoop - Assignment 7-RDD Deep Dive

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

```
//Problem Statement 2.1.3 :What is the distinct number of subjects present in the entire school

//We have RDD created with Only words we can take

val subjectOnly = tupledRDD.map(item=>item(1))

//get distinct subjects

val distinctSubjects = subjectOnly.distinct()

distinctSubjects.foreach(x=>println(x))
```

OUTPUT :

```
maths
history
science
```

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

```
//Problem Statement 2.1.4 : What is the count of the number of students in the school, whose name is Mathew and

//marks is 55

def getStudent(name:String,marks:Int) : Boolean ={
  if (name.equalsIgnoreCase("Mathew") && marks==55)
  return true
  else
    return false
}
val students = tupledRDD.filter(x=>getStudent(x(0),x(3).toInt))

students.collect().foreach(row => println(row.mkString(",")))

val countStudents = students.count()

println("Number of students in the school, whose name is Mathew and marks is 55 -: " + countStudents)
```

OUTPUT:

```
Mathew,history,grade-2,55,13
Mathew,science,grade-2,55,12
Number of students in the school, whose name is Mathew and marks is 55 -: 2
```

Big data Hadoop - Assignment 7-RDD Deep Dive

Problem Statement 2:

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

   Solution Approach 1 –

   1. Create new RDD with key as Grade and value as 1
   2. Get distinct from above created RDD
   3. And get the count of students per grade by using the CountByKey operation

   Note: In this case we have considered that every student is distinct in every grade.

```
//Problem Statement 2.2.1:What is the count of students per grade in the school?

//first separate the grades and map it to count 1 this is considering every student is separate

val stdsWithGrades = tupledRDD.map(x=>(x(2),1))

val group = stdsWithGrades.countByKey()

group.foreach(x=>println("Students in Grade:"+x))
```

OUTPUT :

```
Students in Grade:(grade-2,9)
Students in Grade:(grade-3,4)
Students in Grade:(grade-1,9)
```

   Solution Approach 2 –

   Note: In this case we have to get the distinct students from per grade and then calculate the students.

   1. Create new RDD using map method where Key is Grade and Student Name
   2. Get distinct of the above RDD
   3. Then countbykey and get students belonging to every Grade

```
//Variation : Getting the distinct student counts per grade depending upon his name

val distinctStds = tupledRDD.map(x=>(x(2),x(0)))

val group1 = distinctStds.distinct.countByKey()
      group1.foreach(x=>println(x))
```

OUTPUT :

```
(grade-2,5)
(grade-3,3)
(grade-1,4)
```

Big data Hadoop - Assignment 7-RDD Deep Dive

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

   Solution Approach –

   1. Create a new RDD from tupled RDD using map method where key is Grade + Student Name and Value is marks of students.

```
//Problem Statement 2.2.2 :  Find the average of each student (Note - Mathew is grade-1, is different from Mathew
  //some other grade!
   //We have to create a RDD which is combination of 1. Grade 2. Student Name 3. Marks

   val students1 = tupledRDD.map(x=>((x(2),x(0)),x(3).toInt))
```

2. Get the distinct of above RDD on key Grade + Student_name
3. Group on above distinct RDD on key Grade+ Student_name
4. Use mapValues operations to get sum of marks per grade per student

```
//Get Distinct of the above RDD based on key (Grade + Student Name)
//Group these students based on key (Grade + Student Name)
//Use Map Values to sum the values based on Key
val sum1 = students1.distinct.groupByKey().mapValues(x=>x.sum)
sum1.foreach(x=>println(x))
```

5. After this Create a RDD which has key as Grade + Student_name and value as counter (1)
6. Group above RDD on Key and get the sum of counter by using mapvalues method

```
//Now create second RDD to calculate average of each student i.e. we must have count of subjects of every stude
//We have to create a RDD which is combination of 1. Grade 2. Student Name 3. Counter
val students2 = tupledRDD.map(x=>((x(2),x(0)),1))
//Sum up the counter to get the subjects count
val count1 = students2.groupByKey().mapValues(x=>x.sum)
count1.foreach(x=>println(x))
```

7. Once these two RDD created

1. One for Sum of marks
2. For total Subjects for a Student, Join these two RDD and create new RDD

8. On above created RDD , used map method to get the Key as Grade + Student Name and value as Sum Of marks / no. Of Subjects scored by student

```
//we have to join these two RDDs to get the Sum and count in one RDD so that we can get the Average of each stu
val joinedRDD = sum1.join(count1)
joinedRDD.foreach(x=>println(x))
//calculate average
val averageOfEachStd = joinedRDD.map(x=>( (x._1),(x._2._1/x._2._2)))
averageOfEachStd.foreach(x=>println("Average Of Student: " + x))
```

Entire code :

Big data Hadoop - Assignment 7-RDD Deep Dive

```
//Problem Statement 2.2.2 :  Find the average of each student (Note - Mathew is grade-1, is different from Mathew
   //some other grade!
    //We have to create a RDD which is combination of 1. Grade 2. Student Name 3. Marks
    val students1 = tupledRDD.map(x=>((x(2),x(0)),x(3).toInt))
    //Get Distinct of the above RDD based on key (Grade + Student Name)
    //Group these students based on key (Grade + Student Name)
    //Use Map Values to sum the values based on Key
    val sum1 = students1.distinct.groupByKey().mapValues(x=>x.sum)
    sum1.foreach(x=>println(x))
    //Now create second RDD to calculate average of each student i.e. we must have count of subjects of every stu
    //We have to create a RDD which is combination of 1. Grade 2. Student Name 3. Counter
    val students2 = tupledRDD.map(x=>((x(2),x(0)),1))
    //Sum up the counter to get the subjects count
    val count1 = students2.groupByKey().mapValues(x=>x.sum)
    count1.foreach(x=>println(x))
    //we have to join these two RDDs to get the Sum and count in one RDD so that we can get the Average of each st
    val joinedRDD = sum1.join(count1)
    joinedRDD.foreach(x=>println(x))
    //calculate average
    val averageOfEachStd = joinedRDD.map(x=>( (x._1),(x._2._1/x._2._2)))
    averageOfEachStd.foreach(x=>println("Average Of Student: " + x))
}
```

OUTPUT :

```
Average Of Student: ((grade-1,John),38)
Average Of Student: ((grade-1,Mark),84)
Average Of Student: ((grade-1,Lisa),24)
Average Of Student: ((grade-2,John),74)
Average Of Student: ((grade-2,Andrew),77)
Average Of Student: ((grade-3,Mathew),45)
Average Of Student: ((grade-1,Andrew),43)
Average Of Student: ((grade-2,Mathew),47)
Average Of Student: ((grade-2,Mark),17)
Average Of Student: ((grade-3,Lisa),86)
Average Of Student: ((grade-3,Andrew),35)
Average Of Student: ((grade-2,Lisa),61)
```

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

```
//Problem statement 2.2.3 : What is the average score of students in each subject across all grades?

//Calculate the total of each subject for all grades

//Calculate number of students , those got marks for a subject

val subjectsMarks = tupledRDD.map(x=>(x(1),x(3).toInt))

val sum3 = subjectsMarks.groupByKey().mapValues(x=>x.sum)

sum3.foreach(x=>println(x))

val noOfStudents = tupledRDD.map(x=>(x(1),1)).groupByKey().mapValues(x=>x.sum)

noOfStudents.foreach(x=>println(x))

//join these RDD and calculate the average

val avg2 = sum3.join(noOfStudents).map(x=>( x._1 ,(x._2._1/x._2._2)))

avg2.foreach(x=>println("Average for Subject:"+ x._1+"==>"+x._2))
```

OUTPUT :

```
(maths,281)
(history,558)
(science,306)
(history,8)
(science,8)
(maths,6)
Average for Subject:maths==>46
Average for Subject:history==>69
Average for Subject:science==>38
```

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

```
//Problem statement 2.2.4 : What is the average score of students in each subject per grade?

//Calculate the total of each subject for per grades

//Calculate number of students , those got marks for a subject per grade

//Create a RDD with Key Grade + Subject And Value as Marks

val subMarksPerGrade =  tupledRDD.map(x=>((x(1),x(2)),x(3).toInt))

val sum4 = subMarksPerGrade.groupByKey().mapValues(x=>x.sum)

sum4.foreach(x=>println(x))
//Calculate count of students

val noOfStudents1 = tupledRDD.map(x=>((x(1),x(2)),1)).groupByKey().mapValues(x=>x.sum)

noOfStudents1.foreach(x=>println(x))
/join these RDD and calculate the average

val avg3 = sum4.join(noOfStudents1).map(x=>(x._1 ,(x._2._1/x._2._2)))

avg3.foreach(x=>println("Average for :"+ x._1+"==>"+x._2))
```

OUTPUT :

```
Average for :(history,grade-3)==>86
Average for :(maths,grade-1)==>46
Average for :(maths,grade-2)==>48
Average for :(history,grade-2)==>79
Average for :(science,grade-3)==>38
Average for :(science,grade-1)==>50
Average for :(science,grade-2)==>30
Average for :(history,grade-1)==>51
```

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

```
//Problem statement 2.2.5 :For all students in grade-2, how many have average score greater than 50?

//We have already calculated the average of every student of every grade in Problem statement 2.2.2

//Need to filter the RDD to get students of grade 2 and average is above 50

val grade2Students = averageOfEachStd.filter(x=>x._1._1.equalsIgnoreCase("grade-2") && x._2>=50)

grade2Students.foreach(x=>println(x))
```

OUTPUT :

```
((grade-2,Lisa),61)
((grade-2,John),74)
((grade-2,Andrew),77)
```

Big data Hadoop - Assignment 7-RDD Deep Dive

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

```
//Problem Statement 2.3.1: Are there any students in the college that satisfy the below criteria
// Average score per student_name across all grades is same as average score per student_name per grade
//We have already calculated average per student_name per grade in problem statement 2.2.2
//Calculate average per student_name
//Create above tow RDD with key Student_Name & Value as average
//Intersect these two RDDs and get the student_name having same average in both the above mentioned cases
val sumPerStdName = tupledRDD.map(x=>(x(0),x(3).toInt)).groupByKey().mapValues(y=>y.sum)
sumPerStdName.foreach(x=>println(x))
val countOfStdName = tupledRDD.map(x=>(x(0),1)).groupByKey().mapValues(y=>y.sum)
countOfStdName.foreach(x=>println(x))

//join these two RDDs to get average per student_name
 val avgStdName = sumPerStdName.join(countOfStdName).map(x=>(x._1,x._2._1/x._2._2))

//The RDD containing average for student_name per grade is averageOfEachStd now intersect this RDD with avgStdN
val avgPerGrade  = averageOfEachStd.map(x=>(x._1._2,x._2))
avgStdName.foreach( x=>println("Average per student_Name"+x))

avgPerGrade.foreach( x=>println("Average per student_Name & Grade"+x))

val commonStds = avgStdName.intersection(avgPerGrade)

commonStds.foreach(x=>println("Students having same average per name and per grade"+x))
```

OUTPUT :

```
(Mark,203)
(Andrew,278)
(John,190)
(Lisa,232)
(Mathew,242)
(Mark,4)
(Andrew,6)
(John,4)
(Lisa,4)
(Mathew,4)
Average per student_Name(Mark,50)
Average per student_Name(Andrew,46)
Average per student_Name(John,47)
Average per student_Name(Lisa,58)
Average per student_Name(Mathew,60)
Average per student_Name & Grade(Andrew,43)
Average per student_Name & Grade(Mathew,47)
Average per student_Name & Grade(Mark,17)
Average per student_Name & Grade(Lisa,86)
Average per student_Name & Grade(Andrew,35)
Average per student_Name & Grade(Lisa,61)
Average per student_Name & Grade(John,38)
Average per student_Name & Grade(Mark,84)
Average per student_Name & Grade(Lisa,24)
Average per student_Name & Grade(John,74)
Average per student_Name & Grade(Andrew,77)
Average per student_Name & Grade(Mathew,45)
```

Students having same average per name and per grade : returns null value.