Spark Assignment 17.2

Given a dataset of college students as a text file (name, subject, grade, marks) : Solve the beloe mentioned problem statement in spark rdd.

This assignment is done in the spark shell within the acadgildVM.

Problem Statement 1:

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

Steps Followed:

1) Copied the dataset file in the path /home/acadgild/spark/17.2_Dataset.txt . Then read the text file by using sc.textfile as below.

   ```
   val student_file = sc.textFile("/home/acadgild/spark/17.2_Dataset.txt")
   ```

2) Then created a tuple rdd by using map function over the student file rdd.

   ```
   val student_tuple = student_file.map(line => line.split(",")).map(x => (x(0), x(1), x(2), x(3).toInt,x(4).toInt)).collect
   ```

3) Then created a parallelized collection by using parallelize method on an existing student_tuple. The elements of the collection are copied to form a distributed dataset that can be operated on in parallel, which will be used to solve other problems in the assignment.

   ```
   val tuplerdd= sc.parallelize(student_tuple)
   ```

Spark-shell output

```
scala> val student_file = sc.textFile("/home/acadgild/spark/17.2_Dataset.txt")
student_file: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/17.2_Dataset.txt MapPartitionsRDD[44] at textFile at
 <console>:24

scala> val student_tuple = student_file.map(line => line.split(",")).map(x => (x(0), x(1), x(2), x(3).toInt,x(4).toInt)).c
ollect
student_tuple: Array[(String, String, String, Int, Int)] = Array((Mathew,science,grade-3,45,12), (Mathew,history,grade-2,5
5,13), (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))

scala> val tuplerdd= sc.parallelize(student_tuple)
tuplerdd: org.apache.spark.rdd.RDD[(String, String, String, Int, Int)] = ParallelCollectionRDD[47] at parallelize at <cons
ole>:28
```

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

To find the total number of rows in student file, We are calling the count over tuplerdd as below

```
val row_count= tuplerdd.count
```

Spark-shell output

```
scala> val row_count= tuplerdd.count
row_count: Long = 22
```

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

To find the distinct number of subjects present in the entire school, we are calling the distinct function over the tuplerdd's 2nd element, which refers to the subject,.

The distinct function will returns the distinct subjects from the student file. Then finally calling the count function, to find the number of distinct subjects.

```
var distinct_subject = tuplerdd.map(x=> (x._2)).distinct().count()
```

Spark-shell output

```
scala> var distinct_subject = tuplerdd.map(x=> (x._2)).distinct().count()
distinct_subject: Long = 3
```

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

To find the count of students whose name is Mathew and marks is 55, we are calling the filter function over the tuplerdd's 1st element which refers to the name and 4th and 5th elements, which refers to the marks.,.

We are passing the filter condition as specified as name = Mathew marks 55 , then finally calling the count function, to find the number of items.

```
var mathew_55 = tuplerdd.filter(x=> (x._1).equals("Mathew") && ((x._4==55) || (x._5==55))).count()
```

Spark-shell ouput

```
scala> var mathew_55 = tuplerdd.filter(x=> (x._1).equals("Mathew") && ((x._4==55) || (x._5==55))).count()
mathew_55: Long = 2
```

Problem Statement 2:

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

To find the count of students per grade, we are calling the map function over the tuplerdd's 3rd element which refers to the grade as a key and setting 1 as value for each item.

Then calling the reduceBy key, where the values for each key are aggregated using the given reduce function. Here we are adding the values for each key and finally calling the collect action.

```
val stud_count_each_grade = tuplerdd.map(x=>(x._3,1)).reduceByKey((x,y) =>x+y).collect()
```

The output will have grade and number of of students in each grade.

Spark-shell output

```
scala> val stud_count_each_grade = tuplerdd.map(x=>(x._3,1)).reduceByKey((x,y) =>x+y).collect()
stud_count_each_grade: Array[(String, Int)] = Array((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!).***

To find the average of each students, we are calling the map function over the tuplerdd. And creating a key with tuplerdds 1st and 3rd element which refers to name and grade (Since mathew in grade 1 is different from mathew from some other grade) we are keeping the combination of name and grade as key and calulating the average of 4th and 5th element, which refers to the marks in each subject.

```
var dis1 = tuplerdd.map(x=> (((x._1),(x._3)),((x._4)+(x._5))/2))
```

Then calling the reduceBy key, where the values for each key are aggregated using the refuce function x+y/2. finally calling the collect action.

```
val avg_each_stud_per_grade = dis1.reduceByKey((x,y) => (x + y)/2).collect
```

The output will have name, grade and avarage marks of each student

Spark-shell output

```
scala> var dis1 = tuplerdd.map(x=> (((x._1),(x._3)),((x._4)+(x._5))/2))
dis1: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[10] at map at <console>:30

scala> val avg_each_stud_per_grade = dis1.reduceByKey((x,y) => (x + y)/2).collect
avg_each_stud_per_grade: Array[((String, String), Int)] = Array(((Lisa,grade-1),18), ((Mark,grade-2),15), ((Lisa,grade-2),
37), ((Mathew,grade-3),28), ((Andrew,grade-2),44), ((Andrew,grade-1),26), ((Lisa,grade-3),49), ((John,grade-1),24), ((John
,grade-2),43), ((Mark,grade-1),48), ((Andrew,grade-3),24), ((Mathew,grade-2),41))
```

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

To find the average score of students in each subject across all grades, we are calling the map function over the tuplerdd. And creating a key with tuplerdds 2nd element which refers to the subject and calulating the average of 4th and 5th element, which refers to the marks in each subject.

```
var dis2 = tuplerdd.map(x=> (((x._2)),((x._4)+(x._5))/2))
```

Then calling the reduceBy key, where the values for each key are aggregated using the refuce function x+y/2. finally calling the collect action.

```
var avg_all_grades = dis2.reduceByKey((x,y) => (x + y)/2).collect
```

The output will have subject and avarage score of students in each subject

Spark-shell ouput

```
scala> var dis2 = tuplerdd.map(x=> (((x._2)),((x._4)+(x._5))/2))
dis2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at map at <console>:30

scala> var avg_all_grades = dis2.reduceByKey((x,y) => (x + y)/2).collect
avg_all_grades: Array[(String, Int)] = Array((maths,25), (history,46), (science,24))
```

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

To find the average score of students in each subject pergrades, we are calling the map function over the tuplerdd. And creating a key with tuplerdds 2nd which refers to the subject and 3rd element which refers to the grade (Since we need to find the average score per subject for each grade) we are keeping the combination of grade and subject as key and calulating the average of 4th and 5th element, which refers to the marks in each subject.

```
var dis3 = tuplerdd.map(x=> (((x._2),(x._3)),((x._4)+(x._5))/2))
```

Then calling the reduceBy key, where the values for each key are aggregated using the refuce function x+y/2. finally calling the collect action

```
var avg_each_grade = dis3.reduceByKey((x,y) => (x + y)/2).collect
```

. The output will have subject ,grade and avarage score of students in each subject per grade

Spark-shell ouput

```
scala> var dis3 = tuplerdd.map(x=> (((x._2),(x._3)),((x._4)+(x._5))/2))
dis3: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[14] at map at <console>:30

scala> var avg_each_grade = dis3.reduceByKey((x,y) => (x + y)/2).collect
avg_each_grade: Array[((String, String), Int)] = Array(((history,grade-2),46), ((history,grade-3),49), ((maths,grade-1),24
), ((science,grade-3),26), ((science,grade-1),31), ((science,grade-2),20), ((history,grade-1),34), ((maths,grade-2),30))
```

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

To find the count of students who are in grade 2 with an average score of over 50 , we are parallelizing the val avg_each_stud_per_grade (as we cant filter and call count over an array we are parallelizing) and creating dis4 rdd, then calling the calling the filter function over the dis4rdd's 2nd subset of the 1st element key, which refers to the grade and 2nd elements, which refers to the average marks.,

var dis4 = sc.parallelize(avg_each_stud_per_grade)

We are passing the filter condition as specified as grade = grade-2 marks 50..There are no students from grade 2 has average of over 50.

```
var avg_grd2_over_50 = dis4.filter(x => x._1._2.equals("grade-2") && (x._2)>50).count()
```

The output will give the number of students in grade-2 who has an average of over 50

Spark-shell ouput

```
scala> var dis4 = sc.parallelize(avg_each_stud_per_grade)
dis4: org.apache.spark.rdd.RDD[((String, String), Int)] = ParallelCollectionRDD[33] at parallelize at <console>:34

scala> var avg_grd2_over_50 = dis4.filter(x => x._1._2.equals("grade-2") && (x._2)>50).count()
avg_grd2_over_50: Long = 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?***

Steps Followed:

1) To find the the average score per student name across all grades,we are calling the map function over the tuplerdd. And creating a key with tuplerdds 1st element which refers to the name and calulating the average of 4th and 5th element, which refers to the marks in each subject.

```
var dis5 = tuplerdd.map(x=> (((x._1)),((x._4)+(x._5))/2))
```

Then calling the reduceBy key, where the values for each key are aggregated using the refuce function x+y/2. finally calling the collect action.

```
val avg_each_stud_across_grades = dis5.reduceByKey((x,y) => (x + y)/2).collect
```

The output will have name and avarage marks of each student.

Spark-shell output

```
scala> var dis5 = tuplerdd.map(x=> (((x._1)),((x._4)+(x._5))/2))
dis5: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[35] at map at <console>:30

scala> val avg_each_stud_across_grades = dis5.reduceByKey((x,y) => (x + y)/2).collect
avg_each_stud_across_grades: Array[(String, Int)] = Array((Mark,26), (Andrew,35), (Mathew,40), (John,28), (Lisa,40))
```

2) We have already calculated the average per student name per grade and stored in the rdd, avg_each_stud_per_grade . This rdd contains name, grade and avarage marks of each student. But we only need name and avegrage marks since we need to use interset() function. So we are using map function over the rdd, and flatting it to return an array of string as below.

```
val flat_avg_each_stud_per_grade= avg_each_stud_per_grade.map(x => x._1._1 + ',' + x._2)
```

3) We are also flattening using map function over the rdd, avg_each_stud_across_grades, to return an array of string.

```
val flat_avg_each_stud_across_grades= avg_each_stud_across_grades.map(x=>x._1+ ','+x._2)
```

4) Finally calling intersect function over, the flattened rdd's as below

flat_avg_each_stud_across_grades.intersect(flat_avg_each_stud_per_grade)

Spark-shell output

```
scala> val flat_avg_each_stud_across_grades= avg_each_stud_across_grades.map(x=>x._1+ ','+x._2)
flat_avg_each_stud_across_grades: Array[String] = Array(Mark,26, Andrew,35, Mathew,40, John,28, Lisa,40)

scala> val flat_avg_each_stud_per_grade= avg_each_stud_per_grade.map(x => x._1._1 + ',' + x._2)
flat_avg_each_stud_per_grade: Array[String] = Array(Lisa,18, Mark,15, Lisa,37, Mathew,28, Andrew,44, Andrew,26, Lisa,49, J
ohn,24, John,43, Mark,48, Andrew,24, Mathew,41)

scala> flat_avg_each_stud_across_grades.intersect(flat_avg_each_stud_per_grade)
res1: Array[String] = Array()
```