Session 19- RDD DEEP DIVE

Assignment 1

Dataset.txt

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

Task 1

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

```
scala> val input=sc.textFile("file:////home/acadgild/Downloads/DataSet.txt")
input: org.apache.spark.rdd.RDD[String] = file:////home/acadgild/Downloads/DataSet.txt MapPartitionsRDD[17] at textFile at <c
onsole>:24

scala> input.count
res10: Long = 23

scala>
```

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

```
scala> val countWords=input.flatMap(line =>line.split(",")).map(word => (word,1))
countWords: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[29] at map at <console>:26

scala> countWords.count
res16: Long = 111
```

3. We have a document where the number separator is -, instead of space. Write a spark code, to obtain the count of the total number of words present in the document.

```
scala> val countWords1=input.flatMap(line =>line.split("-")).map(word => (word,1))
countWords1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[31] at map at <console>:26

scala> countWords1.count
res17: Long = 45
```

Task 2

Problem Statement 1:

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

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt")
base: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet.txt MapPartitionsRDD[28] at textFile at <con
sole>:24

scala>

scala> val text=base.filter { x=> {if (x.split(",").length  >=4) true else false}}
text: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[29] at filter at <console>:26

scala> val text1=text.map(x => (x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split(",")(3).toInt,x.split(",")(4).toInt)
))
text1: org.apache.spark.rdd.RDD[(String, (String, String, Int, Int))] = MapPartitionsRDD[30] at map at <console>:28

scala>
```

```
scala> text1.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))

scala>
```

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

```
(Andrew,(science,grade-3,44,14))
(Andrew,(history,grade-2,77,11))

scala> text1.count
res16: Long = 22

scala>
```

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

```
scala> base.collect
res20: Array[String] = Array(maths, history, science)

scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x => (x.split(",")(1),1))
base: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[47] at map at <console>:33

scala> base.foreach(println)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)

scala> val rddReduce = base.reduceByKey((x,y)=>(x+y))
rddReduce: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[48] at reduceByKey at <console>:35

scala> rddReduce.foreach(println)
(maths,6)
(history,8)
(science,8)

scala>
```

- First we read the text file and created RDD by selecting on subject name and mapping them with value 1

- Second we are counting values of occurrence using reduceByKey to get distinct value of number of subjects.

Note:

Fetch only distinct subject in the school.

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x => (x.split(",")(1))).distinct
base: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[44] at distinct at <console>:33

scala> base.collect
res20: Array[String] = Array(maths, history, science)
```

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

- First we are reading the file and creating tuple RDD as “base” with name and marks as key and mapping the value 1.
- Secondly we filter with name “Mathew” and marks 55.
- Third, done by reduce by operation.

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x => ((x.split(",")(0),x.split(",")(3).toInt),
1))
base: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[58] at map at <console>:33

scala> base.collect
res25: Array[((String, Int), Int)] = Array(((Mathew,45),1), ((Mathew,55),1), ((Mark,23),1), ((Mark,76),1), ((John,14),1), ((J
ohn,74),1), ((Lisa,24),1), ((Lisa,86),1), ((Andrew,34),1), ((Andrew,26),1), ((Andrew,74),1), ((Mathew,55),1), ((Mathew,87),1)
, ((Mark,92),1), ((Mark,12),1), ((John,67),1), ((John,35),1), ((Lisa,24),1), ((Lisa,98),1), ((Andrew,23),1), ((Andrew,44),1),
 ((Andrew,77),1))

scala> val RddFilter= base.filter(x =>x._1._1=="Mathew" && x._1._2==55)
RddFilter: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[59] at filter at <console>:35

scala> RddFilter.collect
res26: Array[((String, Int), Int)] = Array(((Mathew,55),1), ((Mathew,55),1))

scala> val rddReduce=RddFilter.reduceByKey((x,y)=> x+y).foreach(println)
((Mathew,55),2)
rddReduce: Unit = ()

scala>
```

Problem Statement 2:

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

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x => ((x.split(",")(2),1)))
base: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[67] at map at <console>:33

scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x => ((x.split(",")(2),1))).reduceByKey((x,y)=
>x+y)
base: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[71] at reduceByKey at <console>:33

scala> base.foreach(println)
(grade-3,4)
(grade-1,9)
(grade-2,9)

scala>
```

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

   - Reading file and fetching name, grade and marks in “base1”

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt")
base: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet.txt MapPartitionsRDD[84] at textFile at <con
sole>:33

scala> val base1 =base.map(x =>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
base1: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[85] at map at <console>:35

scala> base1.foreach(println)
((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)
```

   - Mapping “base1” values and counting it with numeric 1

```
sole>:33

scala> val rddMap=base1.mapValues(x=>(x,1))
rddMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[80] at mapValues at <console>:37

scala> rddMap.foreach(println)
((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))
```

- Using ReduceByKey operation to add occurrences of marks for each key and marks value

```
scala> val rddReduce=rddMap.reduceByKey((x,y) =>(x._1+y._1,x._2+y._2))
rddReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[81] at reduceByKey at <console>:39

scala> rddReduce.foreach(println)
((Lisa,grade-1),(24,1))
((Mark,grade-2),(35,2))
((Lisa,grade-2),(122,2))
((Mathew,grade-3),(45,1))
((Andrew,grade-2),(77,1))
((Andrew,grade-1),(131,3))
((Lisa,grade-3),(86,1))
((John,grade-1),(116,3))
((John,grade-2),(74,1))
((Mark,grade-1),(168,2))
((Andrew,grade-3),(70,2))
((Mathew,grade-2),(197,3))
```

- Calculating the average by summing marks and dividing it with by its count of each key.

```
scala> val studAvg=rddReduce.mapValues{ case(sum,count) => (1.0*sum)/count}
studAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[82] at mapValues at <console>:45

scala> studAvg.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)
```

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

- Reading text file and creating RDD, extracting name and subject as key and marks as value.

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt")
base: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet.txt MapPartitionsRDD[90] at textFile at <con
sole>:33

scala> val baseRdd=base.map(x=>((x.split(",")(0),x.split(",")(1)),x.split(",")(3).toInt))
baseRdd: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[91] at map at <console>:35

scala> baseRdd.count
res39: Long = 22

scala> baseRdd.collect
res40: Array[((String, String), Int)] = Array(((Mathew,science),45), ((Mathew,history),55), ((Mark,maths),23), ((Mark,science
),76), ((John,history),14), ((John,maths),74), ((Lisa,science),24), ((Lisa,history),86), ((Andrew,maths),34), ((Andrew,scienc
e),26), ((Andrew,history),74), ((Mathew,science),55), ((Mathew,history),87), ((Mark,maths),92), ((Mark,science),12), ((John,h
istory),67), ((John,maths),35), ((Lisa,science),24), ((Lisa,history),98), ((Andrew,maths),23), ((Andrew,science),44), ((Andre
w,history),77))
```

- Using mapValues mapping each value with 1

```
scala> val Rddmap=baseRdd.mapValues(x=>(x,1))
Rddmap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[92] at mapValues at <console>:37

scala> Rddmap.foreach(println)
((Mathew,science),(45,1))
((Mathew,history),(55,1))
((Mark,maths),(23,1))
((Mark,science),(76,1))
((John,history),(14,1))
((John,maths),(74,1))
((Lisa,science),(24,1))
((Lisa,history),(86,1))
((Andrew,maths),(34,1))
((Andrew,science),(26,1))
((Andrew,history),(74,1))
((Mathew,science),(55,1))
((Mathew,history),(87,1))
((Mark,maths),(92,1))
((Mark,science),(12,1))
((John,history),(67,1))
((John,maths),(35,1))
((Lisa,science),(24,1))
((Lisa,history),(98,1))
((Andrew,maths),(23,1))
((Andrew,science),(44,1))
((Andrew,history),(77,1))
```

- Adding marks and number of occurrences for each key using reduceByKey and calculating average by dividing the sum of marks and count of its occurrences for each key

```
scala> val rddReduce=Rddmap.reduceByKey((x,y) =>(x._1+y._1,x._2+y._2))
rddReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[93] at reduceByKey at <console>:39

scala> val SubjectAvg=rddReduce.mapValues{ case(sum,count) => (1.0*sum)/count}
SubjectAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[94] at mapValues at <console>:45

scala> SubjectAvg.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?
   - Read textfile and extract subject, grade, key and marks as value.

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt")
base: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet.txt MapPartitionsRDD[96] at textFile at <con
sole>:33

scala> val baseRdd=base.map(x=>((x.split(",")(1),x.split(",")(2)),x.split(",")(3).toInt))
baseRdd: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[97] at map at <console>:35

scala> baseRdd.foreach(println)
((science,grade-3),45)
((history,grade-2),55)
((maths,grade-2),23)
((science,grade-1),76)
((history,grade-1),14)
((maths,grade-2),74)
((science,grade-1),24)
((history,grade-3),86)
((maths,grade-1),34)
((science,grade-3),26)
((history,grade-1),74)
((science,grade-2),55)
((history,grade-2),87)
((maths,grade-1),92)
((science,grade-2),12)
((history,grade-1),67)
((maths,grade-1),35)
((science,grade-2),24)
((history,grade-2),98)
((maths,grade-1),23)
((science,grade-3),44)
((history,grade-2),77)
```

   - Mapping base RDD with value 1 using mapValue function in Rddmapvalue

```
scala> val Rddmapvalue=baseRdd.mapValues(x=>(x,1))
Rddmapvalue: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[98] at mapValues at <console>:37
```

   - Adding marks and number of occurrences for each key using reduceByKey

```
scala> val rddReduce=Rddmapvalue.reduceByKey((x,y) =>(x._1+y._1,x._2+y._2))
rddReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[99] at reduceByKey at <console>:39
```

- Calculating average by dividing sum of marks with number occurrence.

```
scala> val avg_grade=rddReduce.mapValues{ case(sum,count) => (1.0*sum)/count}.foreach(println)
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),38.333333333333336)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
avg_grade: Unit = ()

scala>
```

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

- Read file and create RDD by extracting values name and grade as key and marks as values and mapping each key with value 1.

```
scala> val base=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x =>((x.split(",")(0),x.split(",")(2)),x.split
(",")(3).toInt))
base: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[108] at map at <console>:33

scala> val RddMap=base.mapValues(x =>(x,1))
RddMap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[109] at mapValues at <console>:35

scala> RddMap.collect
res45: Array[((String, String), (Int, Int))] = Array(((Mathew,grade-3),(45,1)), ((Mathew,grade-2),(55,1)), ((Mark,grade-2),(2
3,1)), ((Mark,grade-1),(76,1)), ((John,grade-1),(14,1)), ((John,grade-2),(74,1)), ((Lisa,grade-1),(24,1)), ((Lisa,grade-3),(8
6,1)), ((Andrew,grade-1),(34,1)), ((Andrew,grade-3),(26,1)), ((Andrew,grade-1),(74,1)), ((Mathew,grade-2),(55,1)), ((Mathew,g
rade-2),(87,1)), ((Mark,grade-1),(92,1)), ((Mark,grade-2),(12,1)), ((John,grade-1),(67,1)), ((John,grade-1),(35,1)), ((Lisa,g
rade-2),(24,1)), ((Lisa,grade-2),(98,1)), ((Andrew,grade-1),(23,1)), ((Andrew,grade-3),(44,1)), ((Andrew,grade-2),(77,1)))
```

- Used reduceByKey marks and number of occurrences per key and calculating average of each student.

```
scala> val rddReduce=RddMap.reduceByKey((x,y) =>(x._1+y._1,x._2+y._2))
rddReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[110] at reduceByKey at <console>:37

scala> val Rddavg=rddReduce.mapValues{case(sum,count)=>(1.0*sum)/count}
Rddavg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[111] at mapValues at <console>:43

scala> Rddavg.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)

scala> val rddfilter=Rddavg.filter(x =>x._2._2=="grade-2" && x._2>50)
```

- Filter student with grade-2 and marks with >50 in RDD filter and taking rddfilter.count and print values.

```
scala> val rddfilter=Rddavg.filter(x =>x._1._2 =="grade-2" && x._2>50)
rddfilter: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[112] at filter at <console>:45

scala> rdd.filter.count
<console>:33: error: not found: value rdd
       rdd.filter.count
       ^

scala> rddfilter.count
res48: Long = 4

scala> rddfilter.foreach(println)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)

scala>
```

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

   - Extracting only name and marks and mapping values each values and rdd as 1.

```
scala> val baseRDD1=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x =>(x.split(",")(0),x.split(",")(3).toInt
))
baseRDD1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[2] at map at <console>:24

scala> val studAvg=baseRDD1.mapValues(x =>(x,1))
studAvg: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[3] at mapValues at <console>:26

scala> studAvg.foreach(println)
(Mathew,(45,1))
(Mathew,(55,1))
(Mark,(23,1))
(Mark,(76,1))
(John,(14,1))
(John,(74,1))
(Lisa,(24,1))
(Lisa,(86,1))
(Andrew,(34,1))
(Andrew,(26,1))
(Andrew,(74,1))
(Mathew,(55,1))
(Mathew,(87,1))
(Mark,(92,1))
(Mark,(12,1))
(John,(67,1))
(John,(35,1))
(Lisa,(24,1))
(Lisa,(98,1))
(Andrew,(23,1))
(Andrew,(44,1))
(Andrew,(77,1))
```

- Adding marks and number of occurrence for each student using reduceByKey and calculating the average of each student.

```
scala> val studReduce=studAvg.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
studReduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[9] at reduceByKey at <console>:37

scala> val Avg_Stud=studReduce.mapValues{case (sum,count) =>(1.0*sum)/count}
Avg_Stud: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[10] at mapValues at <console>:39

scala> Avg_Stud.foreach(println)
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,60.5)
(John,47.5)
(Lisa,58.0)
```

Now, we need to find average of each student per grade

- Extracting name and grade as key and marks as values and mapped values with numeric 1.

```
scala> val baseRDD2=sc.textFile("file:///home/acadgild/Downloads/DataSet.txt").map(x =>((x.split(",")(0),x.split(",")(2)),x.s
plit(",")(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[13] at map at <console>:33

scala> val grade=baseRDD2.mapValues(x =>(x,1))
grade: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[14] at mapValues at <console>:35

scala> grade.foreach(println)
((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))
```

- Using reduceByKey operation adding marks and number of occurrences of 1 to each key and calculating average of each key by dividing the sum of marks with count.

```
scala> val gradeReduce=grade.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
gradeReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[15] at reduceByKey at <console>:3

scala> val gradeAvg=gradeReduce.mapValues{case(sum,count)=>(1.0*sum)/count}
gradeAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[16] at mapValues at <console>:39

scala> gradeAvg.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)

scala> val flatgradeAvg=gradeAvg.map(x =>x._1._1 + "," +x._2.toDouble)
```

- Using intersection we are finding common name and “NO COMMON Names FOUND”

```
scala> val flatgradeAvg=gradeAvg.map(x =>x._1._1 + "," +x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[18] at map at <console>:41

scala> flatgradeAvg.foreach(println)
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> val flatAvg_Stud=Avg_Stud.map(x=>x._1 + "," + x._2)
flatAvg_Stud: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[19] at map at <console>:41

scala> flatAvg_Stud.foreach(println)
Mark,50.75
Andrew,46.333333333333336
Mathew,60.5
John,47.5
Lisa,58.0

scala> val commonval=flatgradeAvg.intersection(flatAvg_Stud)
commonval: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[25] at intersection at <console>:53

scala> commonval.foreach(println)

scala>
```