ASSIGNMENT 46.1 – CORE SPARK

Mar 2018 batch - Student: K. Anandaranga

---

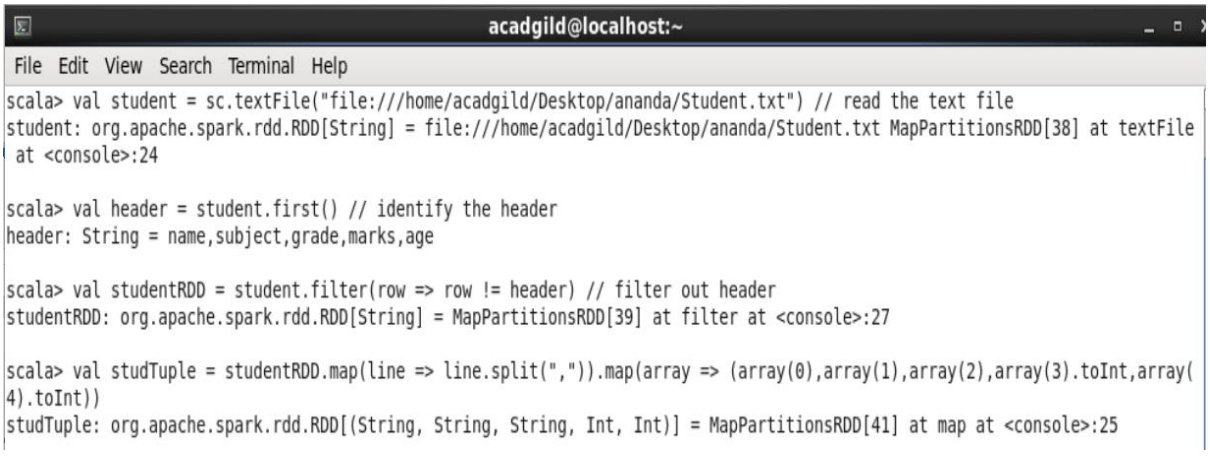
1. Task 1

Given a dataset of college students as a text file (name, subject, grade, marks):
Below is the actual dataset for reference (as downloaded from the url provided)

	name	subject	grade	marks	age
2	Mathew	science	grade-4	5	12
3	Mathew	history	grade-2	55	13
4	Mark	maths	grade-2	23	13
5	Mark	science	grade-1	76	13
6	John	history	grade-1	14	12
7	John	maths	grade-2	74	13
8	Lisa	science	grade-1	24	12
9	Lisa	history	grade-3	86	13
10	Andrew	maths	grade-1	34	13
11	Andrew	science	grade-3	26	14
12	Andrew	history	grade-1	74	12
13	Mathew	science	grade-2	55	12
14	Mathew	history	grade-2	87	12
15	Mark	maths	grade-1	92	13
16	Mark	science	grade-2	12	12
17	John	history	grade-1	67	13
18	John	maths	grade-1	35	11
19	Lisa	science	grade-2	24	13
20	Lisa	history	grade-2	98	16
21	Andrew	maths	grade-1	23	16
22	Andrew	science	grade-3	44	14
23	Andrew	history	grade-2	77	11

Problem Statement 1:

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



```
scala> studTuple.foreach(println)
(Mathew,science,grade-4,5,12)
(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-2,98,16)
(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.

```
scala> student.count
res8: Long = 23

scala> studentRDD.count
res9: Long = 22
```

The file had 23 rows, of which 22 rows pertained to the data (1 header row)

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

```
scala> val uniqSubject = studTuple.map{case(name, subject, grade, marks, age) => subject}.distinct
uniqSubject: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[49] at distinct at <console>:25

scala> uniqSubject.count
res13: Long = 3

scala> uniqSubject.foreach(println)
maths
history
science
```

There are 3 subjects – maths, history, science

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

```
scala> val data = studTuple.map{case (name, subject, grade, marks, age) => (name, marks)}
data: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[53] at map at <console>:25

scala> val filter = data.filter {case (name,marks) => (name == "Mathew" && marks == "55")}
filter: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[54] at filter at <console>:25

scala> val filter = data.filter {case (name,marks) => (name == "Mathew" && marks == "55")}.count
filter: Long = 0
```

There is no one that fits this condition

Problem Statement 2:

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

```
scala> val student = sc.textFile("file:///home/acadgild/Desktop/ananda/Student.txt") // read the text file
student: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/Student.txt MapPartitionsRDD[68] at textFile
 at <console>:27

scala> val header = student.first() // identify the header
header: String = name,subject,grade,marks,age

scala> val studRdd = student.filter(row => row != header) // filter out header
studRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[69] at filter at <console>:30

scala> val baseRDD = studRdd.map(line => line.split(",")).map(array => (array(2),1))
baseRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[71] at map at <console>:28

scala> val RDDreduce = baseRDD.reduceByKey(_+_)
RDDreduce: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[72] at reduceByKey at <console>:28

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

The counts are shown above

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

```
scala> val student = sc.textFile("file:///home/acadgild/Desktop/ananda/Student.txt") // read the text file
student: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/Student.txt MapPartitionsRDD[82] at textFile
 at <console>:27

scala> val header = student.first() // identify the header
header: String = name,subject,grade,marks,age

scala> val studRdd = student.filter(row => row != header) // filter out header
studRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[83] at filter at <console>:30

scala> val baseRDD = studRdd.map(line => line.split(",")).map(array => ((array(0):String,array(1):String,array(2):String),arr
ay(3).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String, String), Int)] = MapPartitionsRDD[85] at map at <console>:28

scala> val RDDmap = baseRDD.mapValues(x => (x,1))
RDDmap: org.apache.spark.rdd.RDD[((String, String, String), (Int, Int))] = MapPartitionsRDD[86] at mapValues at <console>:28

scala> val RDDreduce = RDDmap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
RDDreduce: org.apache.spark.rdd.RDD[((String, String, String), (Int, Int))] = ShuffledRDD[87] at reduceByKey at <console>:28

scala> val SubAvg = RDDreduce.mapValues{case(sum,count) => (1.0*sum/count)}
SubAvg: org.apache.spark.rdd.RDD[((String, String, String), Double)] = MapPartitionsRDD[88] at mapValues at <console>:28
```

Result:

```
scala> SubAvg.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)
((Andrew,maths,grade-1),28.5)
((Mark,maths,grade-2),23.0)
((Mark,science,grade-2),12.0)
((Andrew,science,grade-3),35.0)
((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)
((Mathew,science,grade-4),5.0)
((Lisa,history,grade-2),98.0)
((Lisa,science,grade-1),24.0)
```

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

```
scala> val student = sc.textFile("file:///home/acadgild/Desktop/ananda/Student.txt") // read the text file
student: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/Student.txt MapPartitionsRDD[90] at textFile
 at <console>:27

scala> val header = student.first() // identify the header
header: String = name,subject,grade,marks,age

scala> val studRdd = student.filter(row => row != header) // filter out header
studRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[91] at filter at <console>:30

scala> val baseRDD2 = studRdd.map(line => line.split(",")).map(array => (array(1):String,array(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[93] at map at <console>:28

scala> val RDDmap = baseRDD2.mapValues(x => (x,1))
RDDmap: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[94] at mapValues at <console>:28

scala> val RDDreduce = RDDmap.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
RDDreduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[95] at reduceByKey at <console>:28

scala> val SubAvg = RDDreduce.mapValues{case(sum,count) => (1.0*sum/count)}
SubAvg: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[96] at mapValues at <console>:28

scala> SubAvg.foreach(println)
(maths,46.833333333333336)
(history,69.75)
(science,33.25)
```

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

```
scala> val student = sc.textFile("file:///home/acadgild/Desktop/ananda/Student.txt") // read the text file
student: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/Student.txt MapPartitionsRDD[98] at textFile
 at <console>:27

scala> val header = student.first() // identify the header
header: String = name,subject,grade,marks,age

scala> val studRdd = student.filter(row => row != header) // filter out header
studRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[99] at filter at <console>:30

scala> val baseRDD2 = studRdd.map(line => line.split(",")).map(array => ((array(2):String,array(1):String),array(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[101] at map at <console>:28

scala> val RDDmap = baseRDD2.mapValues(x => (x,1))
RDDmap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[102] at mapValues at <console>:28

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[103] at reduceByKey at <console>:28

scala> val SubAvg = RDDreduce.mapValues{case(sum,count) => (1.0*sum/count)}
SubAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[104] at mapValues at <console>:28

scala> SubAvg.foreach(println)
((grade-1,history),51.666666666666664)
((grade-2,history),79.25)
((grade-3,history),86.0)
((grade-3,science),35.0)
((grade-4,science),5.0)
((grade-1,maths),46.0)
((grade-1,science),50.0)
((grade-2,science),30.333333333333332)
((grade-2,maths),48.5)
```

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

```
scala> val student = sc.textFile("file:///home/acadgild/Desktop/ananda/Student.txt") // read the text file
student: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/Student.txt MapPartitionsRDD[106] at textFil
e at <console>:27

scala> val header = student.first() // identify the header
header: String = name,subject,grade,marks,age

scala> val studRdd = student.filter(row => row != header) // filter out header
studRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[107] at filter at <console>:30

scala> val baseRDD2 = studRdd.map(line => line.split(",")).map(array => ((array(0):String,array(2):String),array(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[109] at map at <console>:28

scala> val RDDmap = baseRDD2.mapValues(x => (x,1))
RDDmap: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[110] at mapValues at <console>:28

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[111] at reduceByKey at <console>:28

scala> val StudAvgMark = RDDreduce.mapValues{case(sum,count) => (1.0*sum/count)}
StudAvgMark: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[112] at mapValues at <console>:28

scala> val StudFilter = StudAvgMark.filter(x => x._1._2 == "grade-2"&& x._2>50)
StudFilter: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[113] at filter at <console>:28

scala> StudFilter.count
res24: Long = 4

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

There are 4 rows that meet the criteria.

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

```
scala> val student = sc.textFile("file:///home/acadgild/Desktop/ananda/Student.txt") // read the text file
student: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/Student.txt MapPartitionsRDD[130] at textFil
e at <console>:27

scala> val header = student.first() // identify the header
header: String = name,subject,grade,marks,age

scala> val studRdd = student.filter(row => row != header) // filter out header
studRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[131] at filter at <console>:30

scala> val baseRDD1 = studRdd.map(line => line.split(",")).map(array => (array(0):String,array(3).toInt))
baseRDD1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[133] at map at <console>:28

scala> val RDDmap1 = baseRDD1.mapValues(x => (x,1))
RDDmap1: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[134] at mapValues at <console>:28

scala> val RDDreduce1 = RDDmap1.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
RDDreduce1: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[135] at reduceByKey at <console>:28

scala> val StudAvgMark = RDDreduce1.mapValues{case(sum,count) => (1.0*sum/count)}
StudAvgMark: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[136] at mapValues at <console>:28

scala> StudAvgMark.foreach(println)
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,50.5)
(John,47.5)
(Lisa,58.0)
```

```
scala> val baseRDD2 = studRdd.map(line => line.split(",")).map(array => ((array(0):String,array(2):String),array(3).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[138] at map at <console>:28

scala> val RDDmap2 = baseRDD2.mapValues(x => (x,1))
RDDmap2: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[139] at mapValues at <console>:28

scala> val RDDreduce2 = RDDmap2.reduceByKey((x,y) => (x._1 + y._1, x._2 + y._2))
RDDreduce2: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[140] at reduceByKey at <console>:28

scala> val StudAvgMarkByGrade = RDDreduce2.mapValues{case(sum,count) => (1.0*sum/count)}
StudAvgMarkByGrade: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[141] at mapValues at <console>:28

scala> StudAvgMarkByGrade.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((Mathew,grade-4),5.0)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
```

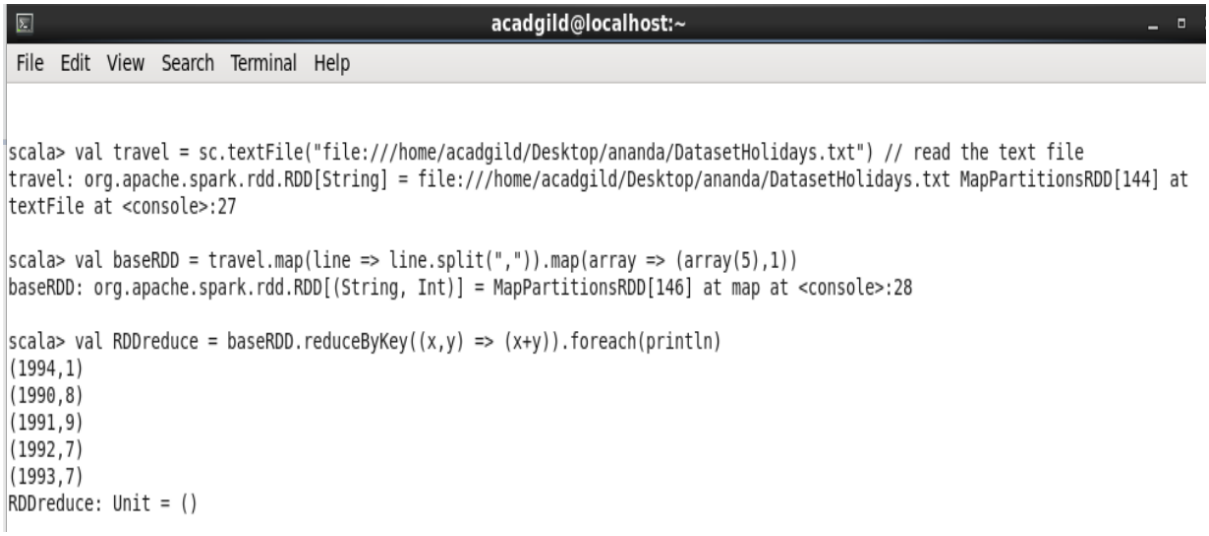
```
scala> val Chk2GradeAvg = StudAvgMarkByGrade.map(x => (x._1._1 + ","+ x._2.toDouble))
Chk2GradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[142] at map at <console>:28

scala> Chk2GradeAvg
res34: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[142] at map at <console>:28

scala> Chk2GradeAvg.foreach(println)
Lisa,24.0
Mark,17.5
Lisa,61.0
Andrew,77.0
Andrew,43.666666666666664
Lisa,86.0
John,38.666666666666664
Mathew,5.0
John,74.0
Mark,84.0
Andrew,35.0
Mathew,65.66666666666667
```

Results show that there is no intersection between the 2 datasets.

5. Task 2

Use below link to download the dataset:
https://drive.google.com/drive/folders/0B_P3pWagdIrrVThBaUdVSUtzbms

1) What is the distribution of the total number of air-travelers per year



2) What is the total air distance covered by each user per year

```
scala> val travel = sc.textFile("file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt") // read the text file
travel: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt MapPartitionsRDD[149] at
textFile at <console>:27

scala> val baseRDD = travel.map(line => line.split(",")).map(array => ((array(0),array(5)),array(4).toInt))
baseRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[151] at map at <console>:28

scala> val distRDD = baseRDD.reduceByKey(_+_).foreach(println)
((3,1992),200)
((3,1993),200)
((5,1991),200)
((6,1991),400)
((10,1993),200)
((5,1992),400)
((8,1991),200)
((8,1990),200)
((1,1993),600)
((5,1994),200)
((2,1993),200)
((2,1991),400)
((4,1990),400)
((10,1992),200)
((3,1991),200)
((1,1990),200)
((10,1990),200)
((6,1993),200)
((9,1992),400)
((8,1992),200)
((7,1990),600)
((9,1991),200)
((4,1991),200)
distRDD: Unit = ()
```

3) Which user has travelled the largest distance till date

```
scala> val travel = sc.textFile("file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt") // read the text file
travel: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt MapPartitionsRDD[154] at
textFile at <console>:27

scala> val baseRDD = travel.map(line => line.split(",")).map(array => ((array(0), array(4).toInt)))
baseRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[156] at map at <console>:28

scala> val largeDist = baseRDD.reduceByKey(_+_).takeOrdered(1)
largeDist: Array[(String, Int)] = Array((1,800))
```

User “1” has travelled the largest distance of 800

4) What is the most preferred destination for all users.

```
scala> val travel = sc.textFile("file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt") // read the text file
travel: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt MapPartitionsRDD[160] at
textFile at <console>:27

scala> val baseRDD = travel.map(line => line.split(",")).map(array => ((array(2), 1)))
baseRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[162] at map at <console>:28

scala> val destRDD = baseRDD.reduceByKey(_+_).takeOrdered(1)(Ordering[Int].reverse.on(_._2))
destRDD: Array[(String, Int)] = Array((IND,9))
```

Most preferred destination is India

6. Task 3

Use the same dataset from Task 2

1) Which route is generating the most revenue per year

```
scala> val holiday = sc.textFile("file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt") // read the text file
holiday: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/DatasetHolidays.txt MapPartitionsRDD[187] at
 textFile at <console>:27

scala> val transport = sc.textFile("file:///home/acadgild/Desktop/ananda/DatasetTransport.txt") // read the text file
transport: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/DatasetTransport.txt MapPartitionsRDD[189]
 at textFile at <console>:27

scala> val user = sc.textFile("file:///home/acadgild/Desktop/ananda/DatasetUserDetails.txt") // read the text file
user: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/ananda/DatasetUserDetails.txt MapPartitionsRDD[191] at
 textFile at <console>:27

scala> val holidayRDD = holiday.map(line => line.split(",")).map(array => ((array(0).toInt,array(1),array(2),array(3),array(4
).toInt,array(5).toInt)))
holidayRDD: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[193] at map at <console>:28

scala> val transportRDD = transport.map(line => line.split(",")).map(array => ((array(0),array(1).toInt)))
transportRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[195] at map at <console>:28

scala> val userRDD = user.map(line => line.split(",")).map(array => ((array(0).toInt,array(1),array(2).toInt)))
userRDD: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[197] at map at <console>:28

scala> val holidaymap = holidayRDD.map(x => x._4 ->(x._2,x._5, x._6))
holidaymap: org.apache.spark.rdd.RDD[(String, (String, Int, Int))] = MapPartitionsRDD[198] at map at <console>:28

scala> val transportmap = transportRDD.map(x => x._1 -> x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[199] at map at <console>:28


scala> val join = holidaymap.join(transportmap)
join: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[202] at join at <console>:30

scala> val route = join.map(x => (x._2._1._1 ->x._2._1._3) -> (x._2._1._2 * x._2._2))
route: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[203] at map at <console>:28

scala> val revenue = route.groupByKey().map(x => x._2.sum -> x._1)
revenue: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[205] at map at <console>:28

scala> val routeHighRevenue = revenue.sortByKey(false).first()
routeHighRevenue: (Int, (String, Int)) = (204000,(IND,1991))
```

2) What is the total amount spent by every user on air-travel per year

```
scala> val usermap = holidayRDD.map(x => x._4 -> (x._1, x._5, x._6))
usermap: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[207] at map at <console>:28

scala> val amt = usermap.join(transportmap)
amt: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[210] at join at <console>:30

scala> val spend = amt.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))
spend: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[211] at map at <console>:28

scala> val total = spend.groupByKey().map(x => x._1 -> x._2.sum)
total: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[213] at map at <console>:28

scala> total.foreach(println)
((2,1993),34000)
((6,1993),34000)
((10,1993),34000)
((10,1992),34000)
((2,1991),68000)
((4,1990),68000)
((10,1990),34000)
((5,1992),68000)
((4,1991),34000)
((1,1993),102000)
((9,1992),68000)
((5,1991),34000)
((3,1993),34000)
((1,1990),34000)
((8,1990),34000)
((7,1990),102000)
((6,1991),68000)
((5,1994),34000)
((3,1991),34000)
((9,1991),34000)
((3,1992),34000)
((8,1991),34000)
((8,1992),34000)
```

3) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

```
scala> val MapAge = userRDD.map(x => x._1 ->
     | {
     | if (x._3 < 20) "20"
     | else if (x._3 > 35) "35"
     | else "20-35"
     | })
MapAge: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[214] at map at <console>:28

scala> val userID = holidayRDD.map(x => x._1 -> 1)
userID: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[215] at map at <console>:28

scala> val joinMap1 = MapAge.join(userID)
joinMap1: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[218] at join at <console>:30

scala> val joinMap2 = joinMap1.map(x => x._2._1 -> x._2._2)
joinMap2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[219] at map at <console>:28

scala> val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)
groupKey: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[221] at map at <console>:28

scala> val mostGroup = groupKey.sortBy(x => -x._2).first()
mostGroup: (String, Int) = (20-35,13)
```

It shows that age group “20-35” is travelling the most every year.