Assignment 18.3

Downloaded the 3 datasets **S18_DatasetHolidays.txt**, **S18_Dataset_Transport.txt** and **S18_Dataset_user_details.txt**

And loaded the datasets into the location /home/acadgild/sumona

```
scala> val baseRDD1 = sc.textFile("/home/acadgild/sumona/S18_Dataset_Holidays.txt")
baseRDD1: org.apache.spark.rdd.RDD[String] = /home/acadgild/sumona/S18_Dataset_Holidays.txt MapPartitionsRDD[27] at textFile at <console>:25

scala> val baseRDD2 = sc.textFile("/home/acadgild/sumona/S18_Dataset_Transport.txt")
baseRDD2: org.apache.spark.rdd.RDD[String] = /home/acadgild/sumona/S18_Dataset_Transport.txt MapPartitionsRDD[29] at textFile at <console>:25

scala> val baseRDD3 = sc.textFile("/home/acadgild/sumona/S18_Dataset_User_details.txt")
baseRDD3: org.apache.spark.rdd.RDD[String] = /home/acadgild/sumona/S18_Dataset_User_details.txt MapPartitionsRDD[31] at textFile at <console>:25

scala>
```

Then we perform a map of the above RDD's

```
scala> val travel = baseRDD1.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5).toInt))
travel: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[32] at map at <console>:28

scala> val transport = baseRDD2.map(x => (x.split(",")(0),x.split(",")(1).toInt))
transport: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[33] at map at <console>:28

scala> val user = baseRDD3.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))
user: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[34] at map at <console>:28

scala>
```

1. Considering age groups of < 20 , 20-35, 35 > ,Which age group spends the most amount of money travelling.

First, we shall set a condition for the Age as the following

val AgeMap = user.map(x => x._1 ->

| {

| if(x._3<20)

| "20"

| else if(x._3>35)

| "35"

| else "20-35"

| })

```
scala> val AgeMap = user.map(x => x._1 ->
     | {
     | if(x._3<20)
     | "20"
     | else if(x._3>35)
     | "35"
     |  else "20-35"
     | })
```

val userMap = travel.map(x => x._4 -> (x._1,x._5))

val transportmap = transport.map(x=> x._1 -> x._2)

val joinCost = userMap.join(transportmap)

val calCost = joinCost.map(x => x._2._1._1 -> x._2._1._2 * x._2._2)

val groupCost = calCost.groupByKey().map(x => x._1 -> x._2.sum)

val groupAgeCost = AgeMap.join(groupCost).map(x => x._2._1 -> x._2._2)

val finalCost = groupAgeCost.groupByKey().map(x => x._1 -> x._2.sum)

val maxVal = finalCost.sortBy(x => -x._2).first()

```
scala> val userMap = travel.map(x => x._4 -> (x._1,x._5))
userMap: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[87] at map at <console>:31

scala> val transportmap = transport.map(x=> x._1 -> x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[88] at map at <console>:31

scala> val joinCost = userMap.join(transportmap)
joinCost: org.apache.spark.rdd.RDD[(String, ((Int, Int), Int))] = MapPartitionsRDD[91] at join at <console>:39

scala> val calCost = joinCost.map(x => x._2._1._1 -> x._2._1._2 * x._2._2)
calCost: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[92] at map at <console>:41

scala> val groupCost = calCost.groupByKey().map(x => x._1 -> x._2.sum)
groupCost: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[94] at map at <console>:43

scala> val groupAgeCost = AgeMap.join(groupCost).map(x => x._2._1 -> x._2._2)
groupAgeCost: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[98] at map at <console>:51

scala> val finalCost = groupAgeCost.groupByKey().map(x => x._1 -> x._2.sum)
finalCost: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[100] at map at <console>:53

scala> val maxVal = finalCost.sortBy(x => -x._2).first()
maxVal: (String, Int) = (20-35,442000)

scala>
```

As we see the output, the age between 20-35 has been spending much of the amount on travelling

2. What is the amount spent by each age-group, every year in travelling?

We first perform a Map on the RDD's and then perform a join.

val UserYearMap = travel.map(x => x._4 -> (x._1,x._5,x._6))

val transportmap = transport.map(x=> x._1 -> x._2)

val UserCost = UserYearMap.join(transportmap)

val CalcCost = UserCost.map(x => x._2._1._1 -> (x._2._1._3,x._2._1._2 * x._2._2))

Then we set a condition for the Age group

```
val AgeMap = user.map(x => x._1 ->
    | {
    | if(x._3<20)
    | "20"
    | else if(x._3>35)
    | "35"
    | else "20-35"
    | })
val CostMap = AgeMap.join(CalcCost).map(x => (x._2._1,x._2._2._1) -> x._2._2._2)
```

Then, using the **groupByKey** we are combining the occurrence of the age and the value

```
val ExpPeryear = CostMap.groupByKey().map(x => x._1 -> x._2.sum)
```

```
scala> val UserYearMap = travel.map(x => x._4 -> (x._1,x._5,x._6))
UserYearMap: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[110] at map at <console>:31

scala> val transportmap = transport.map(x=> x._1 -> x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[111] at map at <console>:31

scala> val UserCost = UserYearMap.join(transportmap)
UserCost: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[114] at join at <console>:39

scala> val CalcCost = UserCost.map(x => x._2._1._1 -> (x._2._1._3,x._2._1._2 * x._2._2))
CalcCost: org.apache.spark.rdd.RDD[(Int, (Int, Int))] = MapPartitionsRDD[115] at map at <console>:41

scala> val AgeMap = user.map(x => x._1 ->
     | {
     | if(x._3<20)
     | "20"
     | else if(x._3>35)
     | "35"
     | else "20-35"
     | })
AgeMap: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[116] at map at
 <console>:31

scala> val CostMap = AgeMap.join(CalcCost).map(x => (x._2._1,x._2._2._1) -> x._2._2._2)
CostMap: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[120] at map at <console>:49

scala> val ExpPeryear = CostMap.groupByKey().map(x => x._1 -> x._2.sum)
ExpPeryear: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[122] at map at <console>:51

scala>
```

Output:

```
scala> ExpPeryear.collect().foreach(println)
((35,1992),136000)
((20,1991),102000)
((20,1993),170000)
((20-35,1990),170000)
((35,1993),34000)
((20-35,1991),136000)
((20-35,1994),34000)
((20,1990),34000)
((20-35,1993),34000)
((20,1992),34000)
((20-35,1992),68000)
((35,1990),68000)
((35,1991),68000)

scala>
```