Assignment 18.1

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

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

First we shall read and load the text file in to an RDD. Following is the command:

val holidayRDD = sc.textFile("/home/acadgild/sumona/S18_Dataset_Holidays.txt")

Then, from the above RDD we are going to select the columns i.e. the year column from the RDD.

val splitRDD = holidayRDD.map(x => (x.split(",")(5).toInt,1))

```
scala> val holidayRDD = sc.textFile("/home/acadgild/sumona/S18_Dataset_Holidays.txt")
holidayRDD: org.apache.spark.rdd.RDD[String] = /home/acadgild/sumona/S18_Dataset_Holidays.txt MapPartitionsRDD[7] at textFile at <console>:24

scala> import org.apache.spark.storage.StorageLevel
import org.apache.spark.storage.StorageLevel

scala> holidayRDD.persist(StorageLevel.MEMORY_ONLY)
res1: holidayRDD.type = /home/acadgild/sumona/S18_Dataset_Holidays.txt MapPartitionsRDD[7] at textFile at <console>:24

scala> val splitRDD = holidayRDD.map(x => (x.split(",")(5).toInt,1))
splitRDD: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[8] at map at <console>:27

scala> splitRDD.collect().foreach(println)
(1990,1)
(1991,1)
(1992,1)
(1990,1)
(1992,1)
(1991,1)
(1990,1)
(1991,1)
(1992,1)
(1993,1)
(1993,1)
(1993,1)
(1993,1)
(1991,1)
(1992,1)
(1993,1)
(1990,1)
(1990,1)
(1991,1)
(1992,1)
(1993,1)
(1991,1)
(1991,1)
(1990,1)
(1991,1)
(1991,1)
(1990,1)
```

And now by using the **reduceByKey** command we are going to combine the occurrences of the year and add the value

val countSplit = splitRDD.reduceByKey((x,y) => (x + y))

And then get the count of the distribution of the total number of air-travelers per year

countSplit.collect().foreach(println)

```
scala> val countSplit = splitRDD.reduceByKey((x,y) => (x + y))
countSplit: org.apache.spark.rdd.RDD[(Int, Int)] = ShuffledRDD[9] at reduceByKey at <console>:29

scala> countSplit.collect().foreach(println)
(1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)

scala>
```

As you can see the output shows the no. of air-traveler for each year

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

From the RDD we will select the columns required i.e. user id, year and distance. Following is the command:

val splitRDD = holidayRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))

```
scala> val splitRDD = holidayRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
splitRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[10] at map at <console>:27

scala> splitRDD.collect().foreach(println)
((1,1990),200)
((2,1991),200)
((3,1992),200)
((4,1990),200)
((5,1992),200)
((6,1991),200)
((7,1990),200)
((8,1991),200)
((9,1992),200)
((10,1993),200)
((1,1993),200)
((2,1993),200)
((3,1993),200)
((4,1991),200)
((5,1992),200)
((6,1993),200)
((7,1990),200)
((8,1990),200)
((9,1991),200)
((10,1992),200)
((1,1993),200)
((2,1991),200)
((3,1991),200)
((4,1990),200)
((5,1991),200)
((6,1991),200)
((7,1990),200)
((8,1992),200)
((9,1992),200)
((10,1990),200)
((1,1993),200)
((5,1994),200)

scala>
```

Now we will use the **reduceByKey** and combine the occurrence of user id and get the output

val distRDD = splitRDD.reduceByKey((x,y) => (x + y))

distRDD.collect().foreach(println)

```
scala> val distRDD = splitRDD.reduceByKey((x,y) => (x + y))
distRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[11] at reduceByKey at <console>:29

scala> distRDD.collect().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)

scala>
```

As you can see the output gives the total air distance covered by each user per year

3. Which user has travelled the largest distance till date

We will first select the columns from RDD i.e. the user id and the year column

val userRDD = holidayRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))

```
scala> val userRDD = holidayRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))
userRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at map at <console>:27

scala> userRDD.collect().foreach(println)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(5,200)

scala>
```

By using the **reduceByKey** we are going to combine the occurrence

val totaldistRDD = userRDD.reduceByKey((x,y) => (x+y))

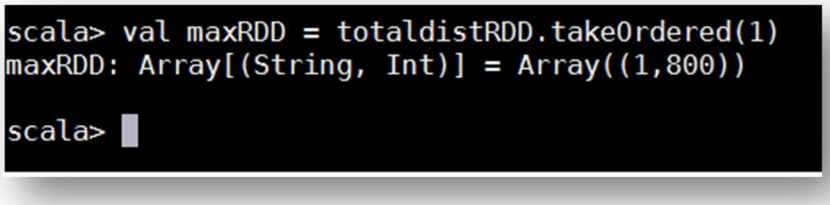
```
scala> val totaldistRDD = userRDD.reduceByKey((x,y) => (x+y))
totaldistRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[13] at reduceByKey at <console>:29

scala> totaldistRDD.collect().foreach(println)
(4,600)
(8,600)
(7,600)
(5,800)
(6,600)
(2,600)
(9,600)
(3,600)
(1,800)
(10,600)

scala>
```

Now to get the user id who travelled the largest distance till date, we will use the following command.

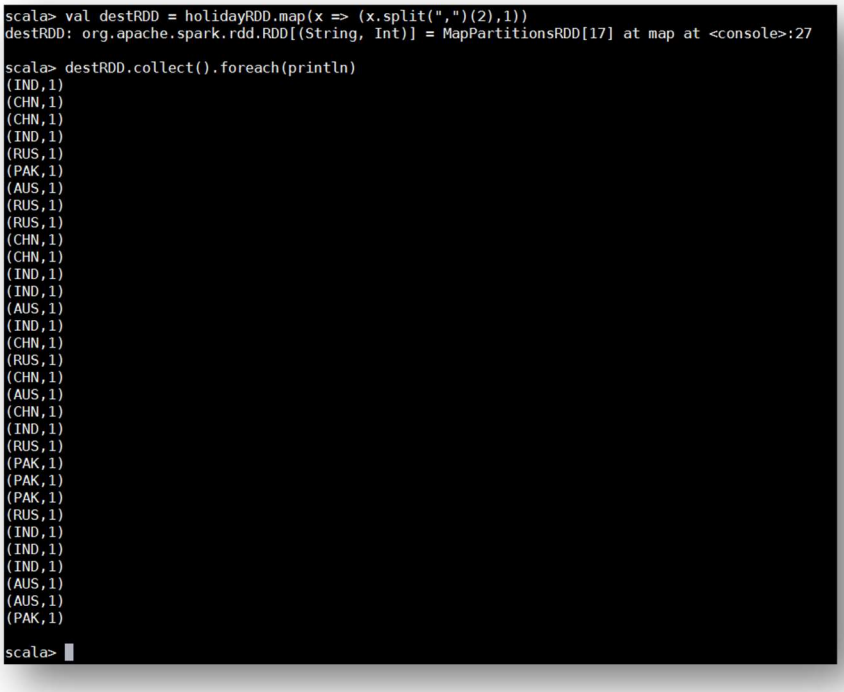
val maxRDD = totaldistRDD.takeOrdered(1)



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

We will first select the columns from the RDD i.e the destination column

val destRDD = holidayRDD.map(x => (x.split(",")(2),1))



By using the **reduceByKey** we will combine the occurrence and add the values

val destreduceRDD = destRDD.reduceByKey((x,y) => (x + y))

```
scala> val destreduceRDD = destRDD.reduceByKey((x,y) => (x + y))
destreduceRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at reduceByKey at <console>:29

scala> destreduceRDD.collect().foreach(println)
(CHN,7)
(IND,9)
(PAK,5)
(RUS,6)
(AUS,5)

scala>
```

Now to get most preferred destination we will be using the following command

val maxRDD = destreduceRDD.takeOrdered(1)(Ordering[Int].reverse.on(_._2))

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

scala>
```

As you can see IND is the most preferred destination