Assignment 18.2

Below is the dataset which we will be using for this Assignment in all problems. It has been kept in local file system:-

```
[acadgild@localhost Assignment-18]$ ls -lrt
total 12
-rw-rw-r--. 1 acadgild acadgild 929 Jan  3 19:49 S18_Dataset_Holidays.txt
-rw-rw-r--. 1 acadgild acadgild  42 Jan  3 19:49 S18_Dataset_Transport.txt
-rw-rw-r--. 1 acadgild acadgild 116 Jan  3 19:49 S18_Dataset_User_details.txt
[acadgild@localhost Assignment-18]$ cat S18_Dataset_Transport.txt
airplane,170
car,140
train,120
ship,200[acadgild@localhost Assignment-18]$ cat S18_Dataset_User_details.txt
1,mark,15
2,john,16
3,luke,17
4,lisa,27
5,mark,25
6,peter,22
7,james,21
8,andrew,55
9,thomas,46
```

```
10,annie,44[acadgild@localhost Assignment-18]$ cat S18_Dataset_Holidays.txt
1,CHN,IND,airplane,200,1990
2,IND,CHN,airplane,200,1991
3,IND,CHN,airplane,200,1992
4,RUS,IND,airplane,200,1990
5,CHN,RUS,airplane,200,1992
6,AUS,PAK,airplane,200,1991
7,RUS,AUS,airplane,200,1990
8,IND,RUS,airplane,200,1991
9,CHN,RUS,airplane,200,1992
10,AUS,CHN,airplane,200,1993
1,AUS,CHN,airplane,200,1993
2,CHN,IND,airplane,200,1993
3,CHN,IND,airplane,200,1993
4,IND,AUS,airplane,200,1991
5,AUS,IND,airplane,200,1992
6,RUS,CHN,airplane,200,1993
7,CHN,RUS,airplane,200,1990
8,AUS,CHN,airplane,200,1990
9,IND,AUS,airplane,200,1991
10,RUS,CHN,airplane,200,1992
1,PAK,IND,airplane,200,1993
2,IND,RUS,airplane,200,1991
3,CHN,PAK,airplane,200,1991
4,CHN,PAK,airplane,200,1990
5,IND,PAK,airplane,200,1991
6,PAK,RUS,airplane,200,1991
7,CHN,IND,airplane,200,1990
8,RUS,IND,airplane,200,1992
9,RUS,IND,airplane,200,1992
10,CHN,AUS,airplane,200,1990
1,PAK,AUS,airplane,200,1993
5,CHN,PAK,airplane,200,1994[acadgild@localhost Assignment-18]$
```

DataSet is uploaded in as follows:-

- val baseRDD1 = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_Holidays.txt")
- val baseRDD2 = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_Transport.txt")
- val baseRDD3 = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_User_details.txt")
- import org.apache.spark.storage.StorageLevel
- baseRDD1.persist(StorageLevel.MEMORY_ONLY)
- baseRDD2.persist(StorageLevel.MEMORY_ONLY)
- baseRDD3.persist(StorageLevel.MEMORY_ONLY)

```
scala> val baseRDD1 = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_Holidays.txt")
baseRDD1: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-18/S18_Dataset_Holidays.txt MapPartitionsRDD[18] at textFile at <console>:26

scala> val baseRDD2 = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_Transport.txt")
baseRDD2: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-18/S18_Dataset_Transport.txt MapPartitionsRDD[20] at textFile at <console>:26

scala> val baseRDD3 = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_User_details.txt")
baseRDD3: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-18/S18_Dataset_User_details.txt MapPartitionsRDD[22] at textFile at <console>:26

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

scala> baseRDD1.persist(StorageLevel.MEMORY_ONLY)
res10: baseRDD1.type = /home/acadgild/Assignment-18/S18_Dataset_Holidays.txt MapPartitionsRDD[18] at textFile at <console>:26

scala> baseRDD2.persist(StorageLevel.MEMORY_ONLY)
res11: baseRDD2.type = /home/acadgild/Assignment-18/S18_Dataset_Transport.txt MapPartitionsRDD[20] at textFile at <console>:26

scala> baseRDD3.persist(StorageLevel.MEMORY_ONLY)
res12: baseRDD3.type = /home/acadgild/Assignment-18/S18_Dataset_User_details.txt MapPartitionsRDD[22] at textFile at <console>:26

scala>

scala>
```

Problem Statement:-

1. Which route is generating the most revenue per year
2. What is the total amount spent by every user on air-travel per year
3. Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

Solution:-

- **Which route is generating the most revenue per year**

Below is the code used:-

- 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))
- val transport = baseRDD2.map(x => (x.split(",")(0),x.split(",")(1).toInt))
- val user = baseRDD3.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))

- val travelmap = travel.map(x=> x._4 -> (x._2,x._5,x._6))
- val transportmap = transport.map(x=> x._1 -> x._2)
- val join1 = travelmap.join(transportmap)
- val routeMap = join1.map(x => (x._2._1._1 -> x._2._1._3) -> (x._2._1._2 * x._2._2))
- val costsum = routeMap.groupByKey().map(x => x._2.sum -> x._1)
- val sortRevenue = costsum.sortByKey(false).first()

Output:-

```
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[23] at map at <console>:29

scala> val transport = baseRDD2.map(x => (x.split(",")(0),x.split(",")(1).toInt))
transport: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[24] at map at <console>:29

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[25] at map at <console>:29

scala>

scala> val travelmap = travel.map(x=> x._4 -> (x._2,x._5,x._6))
travelmap: org.apache.spark.rdd.RDD[(String, (String, Int, Int))] = MapPartitionsRDD[26] at map at <console>:31

scala> val transportmap = transport.map(x=> x._1 -> x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[27] at map at <console>:31

scala> val join1 = travelmap.join(transportmap)
join1: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[30] at join at <console>:39

scala> val routeMap = join1.map(x => (x._2._1._1 -> x._2._1._3) -> (x._2._1._2 * x._2._2))
routeMap: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[31] at map at <console>:41

scala> val costsum = routeMap.groupByKey().map(x => x._2.sum -> x._1)
costsum: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[33] at map at <console>:43

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

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

Below is the code used:-

- val userMap = travel.map(x => x._4 -> (x._1,x._5,x._6))
- val amtMap = userMap.join(transportmap)
- val spendMap = amtMap.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))
- val total = spendMap.groupByKey().map(x => x._1 -> x._2.sum)
- total.foreach(println)

Output:-

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

Below is the code used:-

- val AgeMap = user.map(x => x._1 -> {if(x._3<20) "20" else if(x._3>35) "35" else "20-35" })
- val UIDMap = travel.map(x => x._1 -> 1)
- val joinMap = AgeMap.join(UIDMap)
- val joinMap2 = joinMap.map(x => x._2._1 -> x._2._2)
- val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)
- val maxVal = groupKey.sortBy(x => -x._2).first()

Output:-

```
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[46] at map at <console>:31

scala> val UIDMap = travel.map(x => x._1 -> 1)
UIDMap: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[47] at map at <console>:31

scala> val joinMap = AgeMap.join(UIDMap)
joinMap: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[50] at join at <console>:39

scala> val joinMap2 = joinMap.map(x => x._2._1 -> x._2._2)
joinMap2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[51] at map at <console>:41

scala> val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)
groupKey: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[53] at map at <console>:43

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

scala>
```