Session 20: SPARK SQL 1 Assignment 1

Associated Data Files

https://drive.google.com/open?id=1oWb_IxIzb5PkFgf6P6lwbHXubAMI-HvK

Task 1:

Data set used: DataSet_Holidays.txt

- Load DataSet File in baseRDD

```
scala> val baseRDD=sc.textFile("file:///home/acadgild/Downloads/DataSet_Holidays.txt")
baseRDD: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet_Holidays.txt MapPartitionsRDD[20] at text
File at <console>:26

scala> baseRDD.foreach(println)
```

- printing dataset

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

- To avoid ArrayIndexOutofBound error we have filtered data accordingly in “text”

```
scala> val text=baseRDD.filter{ x=>{if (x.split(",").length >=5) true else false}}
text: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[16] at filter at <console>:28
```

1. What is the distribution of the total number of air-travelers per year.
   - Fetching records with year only and map key with numeric count 1

```
scala> val baseRDD1=text.map(x =>(x.split(",")(5).toInt,1))
baseRDD1: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[17] at map at <console>:30

scala> baseRDD1.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)
(1992,1)
(1992,1)
(1990,1)
(1993,1)
(1994,1)
```

   - Reducing no. of occurrences using reduceByKey in air_travel.

```
scala> val air_travel=baseRDD1.reduceByKey((x,y) =>(x+y)).foreach(println)
(1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)
air_travel: Unit = ()
```

2. What is the total air distance covered by each user per year
   - Created tuple “baseRDD2” and mapping key and value. Here userID, year acts as key and travel distance as value.

```
scala> val baseRDD2=text.map(x =>((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[23] at map at <console>:30

scala> val distance_user=baseRDD2.reduceByKey((x,y)=>(x+y)).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)
distance_user: Unit = ()

scala>
```

3. Which user has travelled the largest distance till date.
   - Created baseRDD3 to map values, and userID as key and travel distance is value.

```
scala> val baseRDD3=text.map(x =>(x.split(",")(0),x.split(",")(4).toInt))
baseRDD3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[25] at map at <console>:30
```

   - Reducing number of occurrences using reduceByKey

```
scala> val large_dist=baseRDD3.reduceByKey((x,y)=>(x+y))
large_dist: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[26] at reduceByKey at <console>:32

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

scala> large_dist.takeOrdered(1)
```

   - Used takeOrdered function to get result

```
scala> large_dist.takeOrdered(1)
res13: Array[(String, Int)] = Array((1,800))

scala>
```

4. What is the most preferred destination for all the users.
   - Created tuple RDD destination as key and numeric 1 as value.

```
scala> val baseRDD4=text.map(x =>(x.split(",")(2),1))
baseRDD4: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[28] at map at <console>:30
```

```
scala> baseRDD4.foreach(println)
(IND,1)
(CHN,1)
(CHN,1)
(IND,1)
(RUS,1)
(PAK,1)
(AUS,1)
(RUS,1)
(RUS,1)
(CHN,1)
(CHN,1)
(IND,1)
(IND,1)
(AUS,1)
(IND,1)
(CHN,1)
(RUS,1)
(CHN,1)
(AUS,1)
(CHN,1)
(IND,1)
(RUS,1)
(PAK,1)
(PAK,1)
(PAK,1)
(RUS,1)
(IND,1)
(IND,1)
(IND,1)
(AUS,1)
(AUS,1)
(PAK,1)
```

   - Reducing the number of occurrences using reduceByKey. Now, we want most destination is taken by using function takeOrdered and ordering in descending manner.

```
scala> val dest=baseRDD4.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(_._2))
dest: Array[(String, Int)] = Array((IND,9))

scala>
```

Note: Creating new tuple rdd's for below query

Loading dataset in spark context baseRDD1, baseRDD2, baseRDD3 and to avoid ArrayIndexOutofBound we are filtering record in text1, text2, text3.

Fetching records and printing in holidays, trans and user.

```
scala> val baseRDD1=sc.textFile("file:///home/acadgild/Downloads/DataSet_Holidays.txt")
baseRDD1: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet_Holidays.txt MapPartitionsRDD[43] at tex
tFile at <console>:27

scala> val text1=baseRDD1.filter{ x=>{if (x.split(",").length >=5) true else false}}
text1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[44] at filter at <console>:29
```

File Edit View Search Terminal Help

```
text1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[44] at filter at <console>:29

scala> val holidays=text1.map(x =>(x.split(",")(0).toInt,x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5
).toInt))
holidays: org.apache.spark.rdd.RDD[(Int, String, String, Int, Int)] = MapPartitionsRDD[45] at map at <console>:31

scala> holidays.foreach(println)
(1,IND,airplane,200,1990)
(2,CHN,airplane,200,1991)
(3,CHN,airplane,200,1992)
(4,IND,airplane,200,1990)
(5,RUS,airplane,200,1992)
(6,PAK,airplane,200,1991)
(7,AUS,airplane,200,1990)
(8,RUS,airplane,200,1991)
(9,RUS,airplane,200,1992)
(10,CHN,airplane,200,1993)
(1,CHN,airplane,200,1993)
(2,IND,airplane,200,1993)
(3,IND,airplane,200,1993)
(4,AUS,airplane,200,1991)
(5,IND,airplane,200,1992)
(6,CHN,airplane,200,1993)
(7,RUS,airplane,200,1990)
(8,CHN,airplane,200,1990)
(9,AUS,airplane,200,1991)
(10,CHN,airplane,200,1992)
(1,IND,airplane,200,1993)
(2,RUS,airplane,200,1991)
(3,PAK,airplane,200,1991)
(4,PAK,airplane,200,1990)
(5,PAK,airplane,200,1991)
(6,RUS,airplane,200,1991)
(7,IND,airplane,200,1990)
(8,IND,airplane,200,1992)
(9,IND,airplane,200,1992)
(10,AUS,airplane,200,1990)
```

```
scala> val text2=baseRDD2.filter{ x=>{if (x.split(",").length >=2) true else false}}
text2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[48] at filter at <console>:29

scala> val trans=text2.map(x =>(x.split(",")(0),x.split(",")(1).toInt))
trans: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[49] at map at <console>:31

scala> trans.foreach(println)
(airplane,170)
(car,140)
(train,120)
(ship,200)

scala> val baseRDD3=sc.textFile("file:///home/acadgild/Downloads/DataSet_User_details.txt")
baseRDD3: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/DataSet_User_details.txt MapPartitionsRDD[51] at
 textFile at <console>:27

scala> val text3=baseRDD3.filter{ x=>{if (x.split(",").length >=3) true else false}}
text3: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[52] at filter at <console>:29

scala> text3.foreach(println)
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


scala> val user =text3.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))
user: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[53] at map at <console>:31

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

5. Which route is generating the most revenue per year?
   - In “holidaysmap” RDD mapping travel mode as key and destination, distance and year as value.
   - In “transportmap” creating rdd as travel mode as key and rate as values
   - Then joining the two RDD using JSON in “join1”
   - In route RDD, destination & year as key and values as multiplication of cost and distance.

```
scala> val holidaysmap=holidays.map(x=>x._4->(x._2,x._5,x._6))
holidaysmap: org.apache.spark.rdd.RDD[(String, (String, Int, Int))] = MapPartitionsRDD[14] at map at <console>:30

scala> val transportmap=transport.map(x=>x._1->x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[15] at map at <console>:30

scala> val join1=holidaysmap.join(transportmap)
join1: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[18] at join at <console>:40

scala> val route=join1.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[19] at map at <console>:42
```

- Collect above query data

```
scala> holidaysmap.collect
res7: Array[(String, (String, Int, Int))] = Array((airplane,(CHN,200,1990)), (airplane,(IND,200,1991)), (airplane,(IND,200,19
92)), (airplane,(RUS,200,1990)), (airplane,(CHN,200,1992)), (airplane,(AUS,200,1991)), (airplane,(RUS,200,1990)), (airplane,(
IND,200,1991)), (airplane,(CHN,200,1992)), (airplane,(AUS,200,1993)), (airplane,(AUS,200,1993)), (airplane,(CHN,200,1993)), (
airplane,(CHN,200,1993)), (airplane,(IND,200,1991)), (airplane,(AUS,200,1992)), (airplane,(RUS,200,1993)), (airplane,(CHN,200
,1990)), (airplane,(AUS,200,1990)), (airplane,(IND,200,1991)), (airplane,(RUS,200,1992)), (airplane,(PAK,200,1993)), (airplan
e,(IND,200,1991)), (airplane,(CHN,200,1991)), (airplane,(CHN,200,1990)), (airplane,(IND,200,1991)), (airplane,(PAK,200,1991))
, (airplane,(CHN,200,1990)), (airplane,(RUS,200...
scala> transportmap.collect
res8: Array[(String, Int)] = Array((airplane,170), (car,140), (train,120), (ship,200))

scala> join1.collect
res9: Array[(String, ((String, Int, Int), Int))] = Array((airplane,((CHN,200,1990),170)), (airplane,((IND,200,1991),170)), (a
irplane,((IND,200,1992),170)), (airplane,((RUS,200,1990),170)), (airplane,((CHN,200,1992),170)), (airplane,((AUS,200,1991),17
0)), (airplane,((RUS,200,1990),170)), (airplane,((IND,200,1991),170)), (airplane,((CHN,200,1992),170)), (airplane,((AUS,200,1
993),170)), (airplane,((AUS,200,1993),170)), (airplane,((CHN,200,1993),170)), (airplane,((CHN,200,1993),170)), (airplane,((IN
D,200,1991),170)), (airplane,((AUS,200,1992),170)), (airplane,((RUS,200,1993),170)), (airplane,((CHN,200,1990),170)), (airpla
ne,((AUS,200,1990),170)), (airplane,((IND,200,1991),170)), (airplane,((RUS,200,1992),170)), (airplane,((PAK,200,1993),170)),
(airplane,((IND,200,1991),170)), (airplane,((CH...
```

```
scala> route.foreach(println)
((CHN,1990),34000)
((IND,1991),34000)
((IND,1992),34000)
((RUS,1990),34000)
((CHN,1992),34000)
((AUS,1991),34000)
((RUS,1990),34000)
((IND,1991),34000)
((CHN,1992),34000)
((AUS,1993),34000)
((AUS,1993),34000)
((CHN,1993),34000)
((CHN,1993),34000)
((IND,1991),34000)
((AUS,1992),34000)
((RUS,1993),34000)
((CHN,1990),34000)
((AUS,1990),34000)
((IND,1991),34000)
((RUS,1992),34000)
((PAK,1993),34000)
((IND,1991),34000)
((CHN,1991),34000)
((CHN,1990),34000)
((IND,1991),34000)
((PAK,1991),34000)
((CHN,1990),34000)
((RUS,1992),34000)
((RUS,1992),34000)
((CHN,1990),34000)
((PAK,1993),34000)
((CHN,1994),34000)
```

- Now using groupByKey, we are grouping destination & year with sum of costs.

```
scala> val revenue=route.groupByKey.map(x=>x._2.sum->x._1)
revenue: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[21] at map at <console>:44

scala> revenue.foreach(println)
(102000,(RUS,1992))
(68000,(AUS,1993))
(170000,(CHN,1990))
(34000,(RUS,1993))
(34000,(AUS,1991))
(68000,(RUS,1990))
(34000,(IND,1992))
(204000,(IND,1991))
(34000,(AUS,1990))
(34000,(CHN,1994))
(34000,(CHN,1991))
(34000,(AUS,1992))
(68000,(CHN,1992))
(68000,(CHN,1993))
(34000,(PAK,1991))
(68000,(PAK,1993))

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

6. What is the total amount spent by every user on air-travel per year
   - Creating userMap tuple RDD from holidays and rdd making travel mode as key and userID, distance & year as values and printing results

```
scala> val userMap=holidays.map(x=>x._4->(x._1,x._5,x._6))
userMap: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[23] at map at <console>:30

scala> userMap.foreach(println)
(airplane,(1,200,1990))
(airplane,(2,200,1991))
(airplane,(3,200,1992))
(airplane,(4,200,1990))
(airplane,(5,200,1992))
(airplane,(6,200,1991))
(airplane,(7,200,1990))
(airplane,(8,200,1991))
(airplane,(9,200,1992))
(airplane,(10,200,1993))
(airplane,(1,200,1993))
(airplane,(2,200,1993))
(airplane,(3,200,1993))
(airplane,(4,200,1991))
(airplane,(5,200,1992))
(airplane,(6,200,1993))
(airplane,(7,200,1990))
(airplane,(8,200,1990))
(airplane,(9,200,1991))
(airplane,(10,200,1992))
(airplane,(1,200,1993))
(airplane,(2,200,1991))
(airplane,(3,200,1991))
(airplane,(4,200,1990))
(airplane,(5,200,1991))
(airplane,(6,200,1991))
(airplane,(7,200,1990))
(airplane,(8,200,1992))
(airplane,(9,200,1992))
```

- Joining userMap with already created RDD transportMap using Join and printing results in amount.

```
scala> val amount=userMap.join(transportmap)
amount: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[26] at join at <console>:40

scala> amount.foreach(println)
(airplane,((1,200,1990),170))
(airplane,((2,200,1991),170))
(airplane,((3,200,1992),170))
(airplane,((4,200,1990),170))
(airplane,((5,200,1992),170))
(airplane,((6,200,1991),170))
(airplane,((7,200,1990),170))
(airplane,((8,200,1991),170))
(airplane,((9,200,1992),170))
(airplane,((10,200,1993),170))
(airplane,((1,200,1993),170))
(airplane,((2,200,1993),170))
(airplane,((3,200,1993),170))
(airplane,((4,200,1991),170))
(airplane,((5,200,1992),170))
(airplane,((6,200,1993),170))
(airplane,((7,200,1990),170))
(airplane,((8,200,1990),170))
(airplane,((9,200,1991),170))
(airplane,((10,200,1992),170))
(airplane,((1,200,1993),170))
(airplane,((2,200,1991),170))
(airplane,((3,200,1991),170))
(airplane,((4,200,1990),170))
(airplane,((5,200,1991),170))
```

- Now calculating the expenses of each year by multiplying distance and amount sent for travel mode.

```
scala> val spend=amount.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[27] at map at <console>:42

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

- Finally, we are summing total value for each user year wise.

```
scala> val total=spend.groupByKey.map(x=>x._1->x._2.sum)
total: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[29] at map at <console>:44

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)

scala>
```

7. Considering age groups of <20, 20-35, 35>, which group is travelling the most every year.
   - In order to take particular group, we need to create RDD with that logic which gives set of groups

```
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[30] at map at <console>:30
```

   - Mapping userID from holidays with count 1

```
scala> val userID=holidays.map(x=>x._1->1)
userID: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[31] at map at <console>:30

scala> userID.foreach(println)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(6,1)
(7,1)
(8,1)
(9,1)
(10,1)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(6,1)
(7,1)
(8,1)
(9,1)
(10,1)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(6,1)
(7,1)
(8,1)
(9,1)
(10,1)
(1,1)
(5,1)
```

- Joining two rdd userID and AgeMap in joinMap1 and then filtering userID

```
scala> val joinMap1=AgeMap.join(userID)
joinMap1: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[34] at join at <console>:40

scala> joinMap1.foreach(println)
(4,(20-35,1))
(4,(20-35,1))
(4,(20-35,1))
(1,(20,1))
(1,(20,1))
(1,(20,1))
(1,(20,1))
(6,(20-35,1))
(6,(20-35,1))
(6,(20-35,1))
(3,(20,1))
(3,(20,1))
(3,(20,1))
(7,(20-35,1))
(7,(20-35,1))
(7,(20-35,1))
(9,(35,1))
(9,(35,1))
(9,(35,1))
(8,(35,1))
(8,(35,1))
(8,(35,1))
(10,(35,1))
(10,(35,1))
(10,(35,1))
(5,(20-35,1))
(5,(20-35,1))
(5,(20-35,1))
(5,(20-35,1))
(2,(20,1))
(2,(20,1))
(2,(20,1))
```

- Eliminating userID in joinMap2

```
scala> val joinMap2=joinMap1.map(x=>x._2._1->x._2._2)
joinMap2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[38] at map at <console>:42

scala> joinMap2.foreach(println)
(20-35,1)
(20-35,1)
(20-35,1)
(20,1)
(20,1)
(20,1)
(20,1)
(20-35,1)
(20-35,1)
(20-35,1)
(20,1)
(20,1)
(20,1)
(20-35,1)
(20-35,1)
(20-35,1)
(35,1)
(35,1)
(35,1)
(35,1)
(35,1)
(35,1)
(35,1)
(35,1)
(35,1)
(20-35,1)
(20-35,1)
(20-35,1)
(20-35,1)
(20,1)
(20,1)
(20,1)
```

- In grouping rdd, summing up the total value by grouping by age group

```
scala> val group=joinMap2.groupByKey.map(x=>x._1->x._2.sum)
group: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[40] at map at <console>:44

scala> group.foreach(println)
(20,10)
(20-35,13)
(35,9)
```

- Using first() function found which age group has travelled the most every year.

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

scala>
```