Assignment 20: Spark SQL I Assignment Problems

Initial Terminal Execution:

```
[acadgild@localhost ~]$ jps
3383 Jps
[acadgild@localhost ~]$ sudo service sshd start
[sudo] password for acadgild:
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/09/04 00:30:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to
/home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.
out
localhost: starting datanode, logging to
/home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.o
ut
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to
/home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localhost.loc
aldomain.out
18/09/04 00:30:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to
/home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.localdom
ain.out
localhost: starting nodemanager, logging to
/home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localhost.localdomain.
out
[acadgild@localhost ~]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

scala> val initialHolidayRDD =
sc.textFile("file:///home/acadgild/Desktop/S20_Dataset_Holidays.txt")
18/09/04 02:48:11 WARN util.SizeEstimator: Failed to check whether UseCompressedOops is set;
assuming yes
initialHolidayRDD: org.apache.spark.rdd.RDD[String] =
file:///home/acadgild/Desktop/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at
<console>:24

scala> initialHolidayRDD.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

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

//Comment : To cache initialHolidayRDD using persist

scala> initialHolidayRDD.persist(StorageLevel.MEMORY_ONLY)
res1: initialHolidayRDD.type = file:///home/acadgild/Desktop/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val initialTransportRDD = sc.textFile("file:///home/acadgild/Desktop/S20_Dataset_Transport.txt")
initialTransportRDD: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/S20_Dataset_Transport.txt MapPartitionsRDD[3] at textFile at <console>:25

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

//Comment : To cache initialTransportRDD using persist

```
scala> initialTransportRDD.persist(StorageLevel.MEMORY_ONLY)
res3: initialTransportRDD.type = file:///home/acadgild/Desktop/S20_Dataset_Transport.txt
MapPartitionsRDD[3] at textFile at <console>:25

scala> val initialUserRDD =
sc.textFile("file:///home/acadgild/Desktop/S20_Dataset_User_details.txt")
initialUserRDD: org.apache.spark.rdd.RDD[String] =
file:///home/acadgild/Desktop/S20_Dataset_User_details.txt MapPartitionsRDD[5] at textFile at
<console>:25

scala> initialUserRDD.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
```

//Comment : To cache initialUserRDD using persist

```
scala> initialUserRDD.persist(StorageLevel.MEMORY_ONLY)
res5: initialUserRDD.type = file:///home/acadgild/Desktop/S20_Dataset_User_details.txt
MapPartitionsRDD[5] at textFile at <console>:25
```

Data Set Description :

User: id, name, age

Transport: transport_mode, cost_per_unit

Holidays: id, source, destination, transport_mode, distance, year

Data Sets Present:

1. S20_Dataset_Holidays.txt :

Terminal Execution :

[acadgild@localhost ~]$ cat /home/acadgild/Desktop/S20_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

2. S20_Dataset_Transport.txt :

Terminal Execution :

[acadgild@localhost ~]$ cat /home/acadgild/Desktop/S20_Dataset_Transport.txt
airplane,170
car,140
train,120
ship,200

3. S20_Dataset_User_details.txt:

Terminal Execution :

[acadgild@localhost ~]$ cat /home/acadgild/Desktop/S20_Dataset_User_details.txt1,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

Problem Statement

--==========

Task 1

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

Terminal Execution :

```
scala> val initialHolidayRDD1 = initialHolidayRDD.map(x => (x.split(",")(5).toInt,1))
initialHolidayRDD1: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[6] at map at
<console>:27

scala> initialHolidayRDD1.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)

scala> val totalAirPerYear = initialHolidayRDD1.reduceByKey((x,y)=>(x+y))
totalAirPerYear: org.apache.spark.rdd.RDD[(Int, Int)] = ShuffledRDD[7] at reduceByKey at <console>:29

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

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

Terminal Execution :

scala> val initialHolidayRDD2 = initialHolidayRDD.map(x =>((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
initialHolidayRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[8] at map at <console>:27

scala> val distancePerUserPerYear = initialHolidayRDD2.reduceByKey((x,y)=>(x+y))
distancePerUserPerYear: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[9] at reduceByKey at <console>:29

scala> distancePerUserPerYear.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)

3. Which user has travelled the largest distance till date

Terminal Execution :

```
scala> val initialHolidayRDD3 = initialHolidayRDD.map(x =>(x.split(",")(0),x.split(",")
(4).toInt))
initialHolidayRDD3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[10] at
map at <console>:27

scala> initialHolidayRDD3.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> val largestDistanceUser =
initialHolidayRDD3.reduceByKey((x,y)=>(x+y)).takeOrdered(1)
largestDistanceUser: Array[(String, Int)] = Array((1,800))

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

Terminal Execution :

scala> val initialHolidayRDD4 = initialHolidayRDD.map(x =>(x.split(",")(2),1))
initialHolidayRDD4: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[13] at map at
<console>:27

scala> initialHolidayRDD4.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)

```
scala> val destinations = initialHolidayRDD4.reduceByKey((x,y)=>(x+y))
destinations: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[14] at reduceByKey at
<console>:29

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

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

5. Which route is generating the most revenue per year

Terminal Execution :

```
scala> val holiday = initialHolidayRDD.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")
(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5).toInt))
holiday: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[16] at
map at <console>:27

scala> val transport = initialTransportRDD.map(x => (x.split(",")(0),x.split(",")(1).toInt))
transport: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[17] at map at <console>:27
```

```
scala> val user = initialUserRDD.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))
user: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[18] at map at <console>:27


scala> holiday.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)

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

```
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)


scala> val holidayRelation = holiday.map(x =>x._4->(x._2,x._5,x._6))
holidayRelation: org.apache.spark.rdd.RDD[(String, (String, Int, Int))] = MapPartitionsRDD[19] at
map at <console>:29

scala> val transportRelation = transport.map(x =>x._1->x._2)
transportRelation: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[20] at map at
<console>:29

scala> holidayRelation.foreach(println)
(airplane,(CHN,200,1990))
(airplane,(IND,200,1991))
(airplane,(IND,200,1992))
(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))
(airplane,(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,200,1992))
(airplane,(RUS,200,1992))
(airplane,(CHN,200,1990))
(airplane,(PAK,200,1993))
(airplane,(CHN,200,1994))

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

scala> val joinRelation = holidayRelation.join(transportRelation)
joinRelation: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[23] at join at <console>:37

scala> joinRelation.foreach(println)
(airplane,((CHN,200,1990),170))
(airplane,((IND,200,1991),170))
(airplane,((IND,200,1992),170))
(airplane,((RUS,200,1990),170))
(airplane,((CHN,200,1992),170))
(airplane,((AUS,200,1991),170))
(airplane,((RUS,200,1990),170))
(airplane,((IND,200,1991),170))
(airplane,((CHN,200,1992),170))
(airplane,((AUS,200,1993),170))
(airplane,((AUS,200,1993),170))
(airplane,((CHN,200,1993),170))
(airplane,((CHN,200,1993),170))
(airplane,((IND,200,1991),170))
(airplane,((AUS,200,1992),170))
(airplane,((RUS,200,1993),170))
(airplane,((CHN,200,1990),170))
(airplane,((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,((CHN,200,1991),170))
(airplane,((CHN,200,1990),170))
(airplane,((IND,200,1991),170))
(airplane,((PAK,200,1991),170))
(airplane,((CHN,200,1990),170))
(airplane,((RUS,200,1992),170))
(airplane,((RUS,200,1992),170))

(airplane,((CHN,200,1990),170))
(airplane,((PAK,200,1993),170))
(airplane,((CHN,200,1994),170))

scala> val routeMap = joinRelation.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[24] at map at <console>:39

scala> routeMap.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)

scala> val revenueMap = routeMap.groupByKey().map(x=>x._2.sum->x._1)
revenueMap: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[26] at map at <console>:41

scala> revenueMap.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 mostRevenueRoutePerYear = revenueMap.sortByKey(false).first()
mostRevenueRoutePerYear: (Int, (String, Int)) = (204000,(IND,1991))

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

Terminal Execution :

scala> val userRelation = holiday.map(x=>x._4 ->(x._1,x._5,x._6))
userRelation: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[28] at map at <console>:29

scala> userRelation.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))
(airplane,(10,200,1990))
(airplane,(1,200,1993))
(airplane,(5,200,1994))

scala> val amountJoin = userRelation.join(transportRelation)
amountJoin: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[31] at join at <console>:37

scala> amountJoin.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))
(airplane,((6,200,1991),170))
(airplane,((7,200,1990),170))
(airplane,((8,200,1992),170))
(airplane,((9,200,1992),170))

(airplane,((10,200,1990),170))
(airplane,((1,200,1993),170))
(airplane,((5,200,1994),170))

scala> val spendMap = amountJoin.map(x=>(x._2._1._1,x._2._1._3)->(x._2._1._2*x._2._2))
spendMap: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[32] at map at <console>:39

scala> spendMap.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)
((5,1994),34000)

scala> val totalPerUserPerYear = spendMap.groupByKey().map(x=>x._1->x._2.sum)
totalPerUserPerYear: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[34] at map at <console>:41

scala> totalPerUserPerYear.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)

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

Terminal Execution :

```
scala> val AgeGroup = user.map(x=>x._1->{
     | if(x._3<20) "20"
     | else if(x._3>35) "35"
     | else "20-35"
     | })
AgeGroup: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[35] at map at
<console>:29

scala> AgeGroup.foreach(println)
(1,20)
(2,20)
(3,20)
(4,20-35)
(5,20-35)
(6,20-35)
(7,20-35)
```

```
(8,35)
(9,35)
(10,35)

scala> val id = holiday.map(x=>x._1->1)
id: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[36] at map at <console>:29

scala> id.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)

scala> val join1 = AgeGroup.join(id)
join1: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[39] at join at
<console>:37
```

```
scala> join1.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))

scala> val join1Map = join1.map(x=>x._2._1 -> x._2._2)
join1Map: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[40] at map at
<console>:39

scala> join1Map.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)

scala> val group = join1Map.groupByKey.map(x=>x._1->x._2.sum)
group: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[42] at map at <console>:41

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

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

Thus, (20-35) age group is the most travelling group.