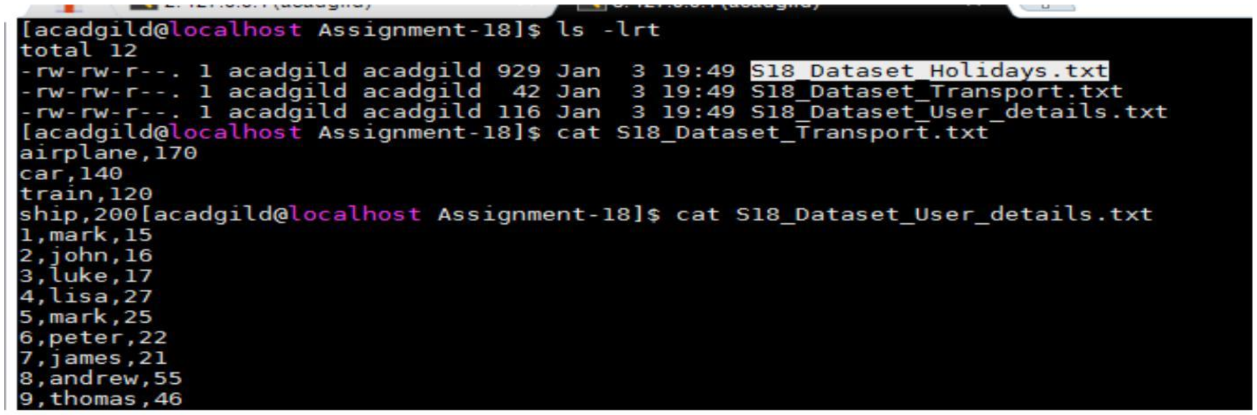
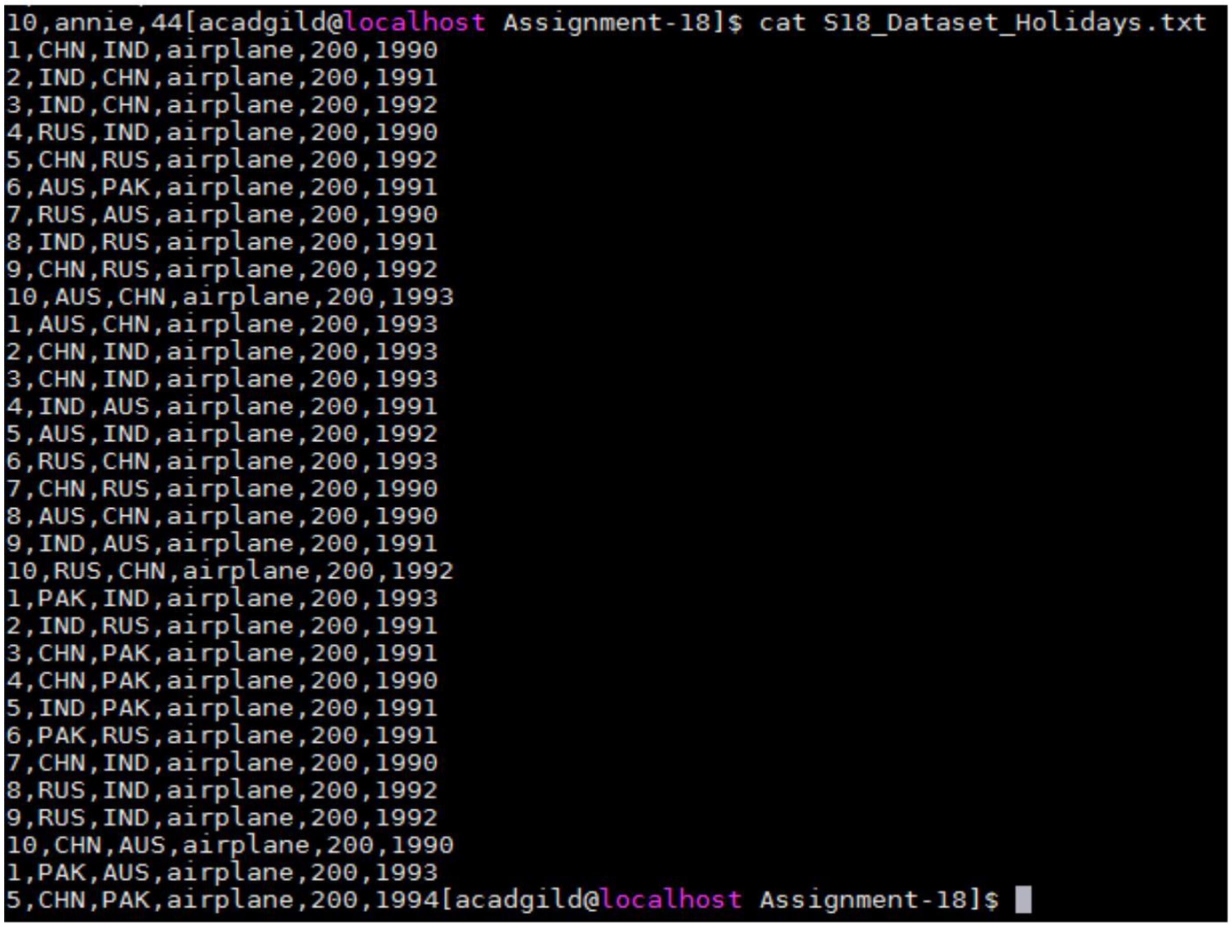
Assignment 18.3

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



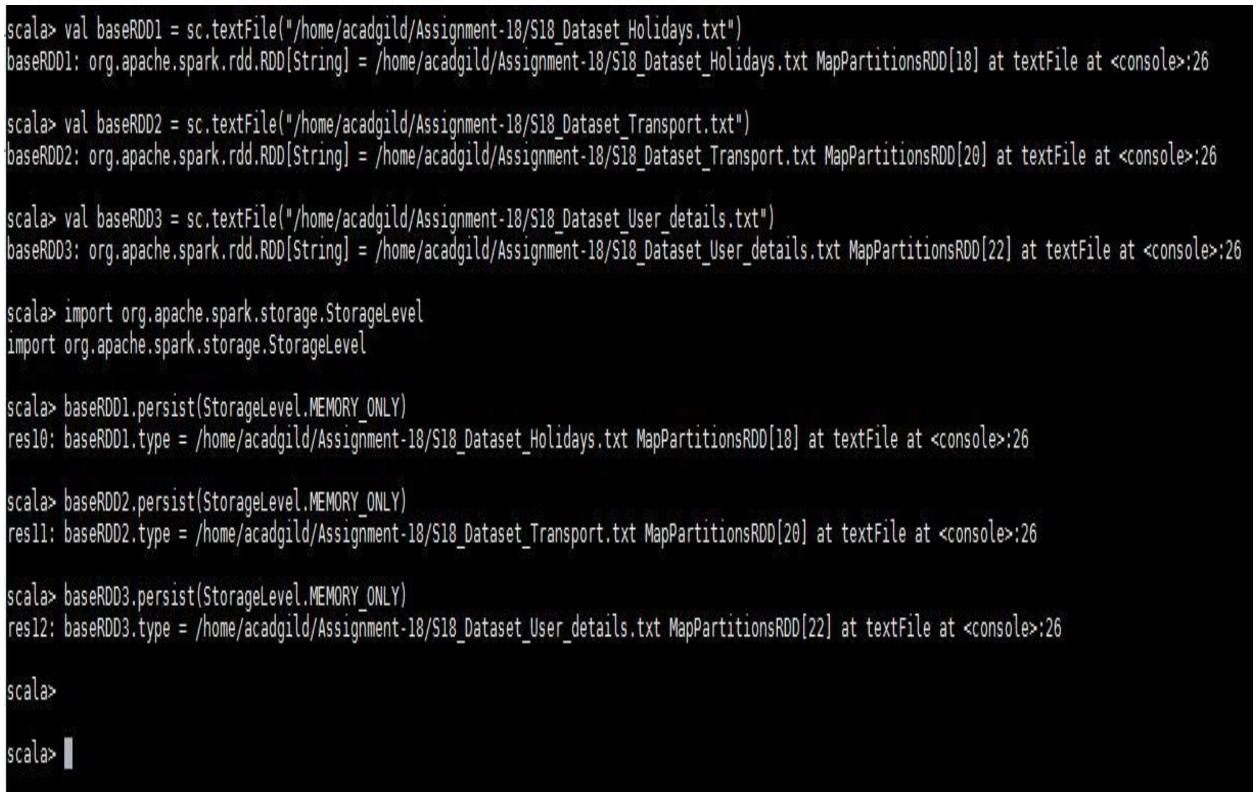
```
[acadgild@localhost Assignment-18]$ ls -lrt
total 12
-rw-rw-r--. 1 acadgild acadgild 929 Jan  3 19:49 S18_Dataset_Holidays.txt
-rw-rw-r--. 1 acadgild acadgild  42 Jan  3 19:49 S18_Dataset_Transport.txt
-rw-rw-r--. 1 acadgild acadgild 116 Jan  3 19:49 S18_Dataset_User_details.txt
[acadgild@localhost 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. Considering age groups of < 20 , 20-35, 35 > ,Which age group spends the most amount of money travelling.

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

travelling? Solution:-

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

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 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)
  ```

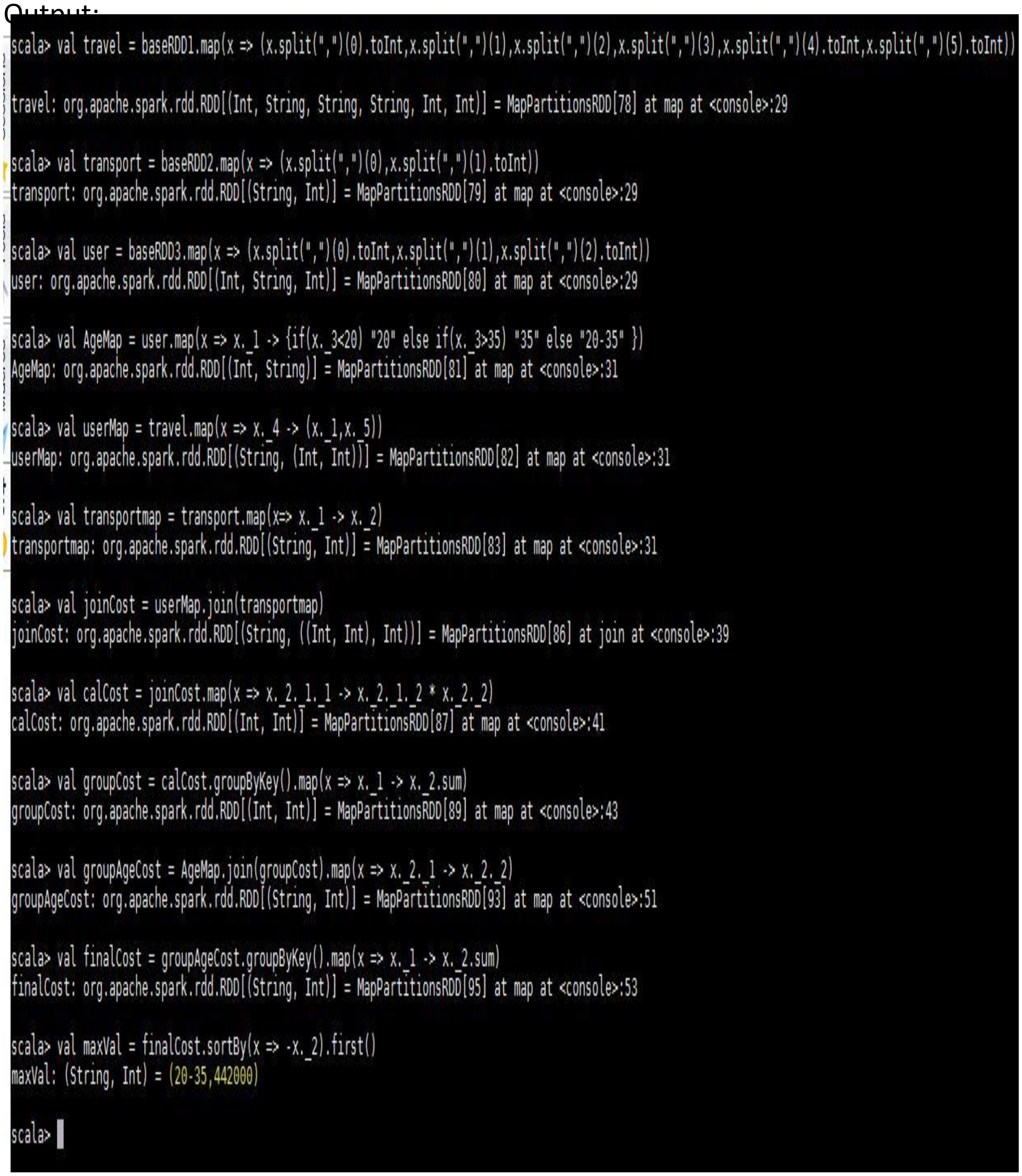
-
  ```
  val groupCost = calCost.groupByKey().map(x => x._1 -> x._2.sum)
  ```

-
  ```
  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()

Output:



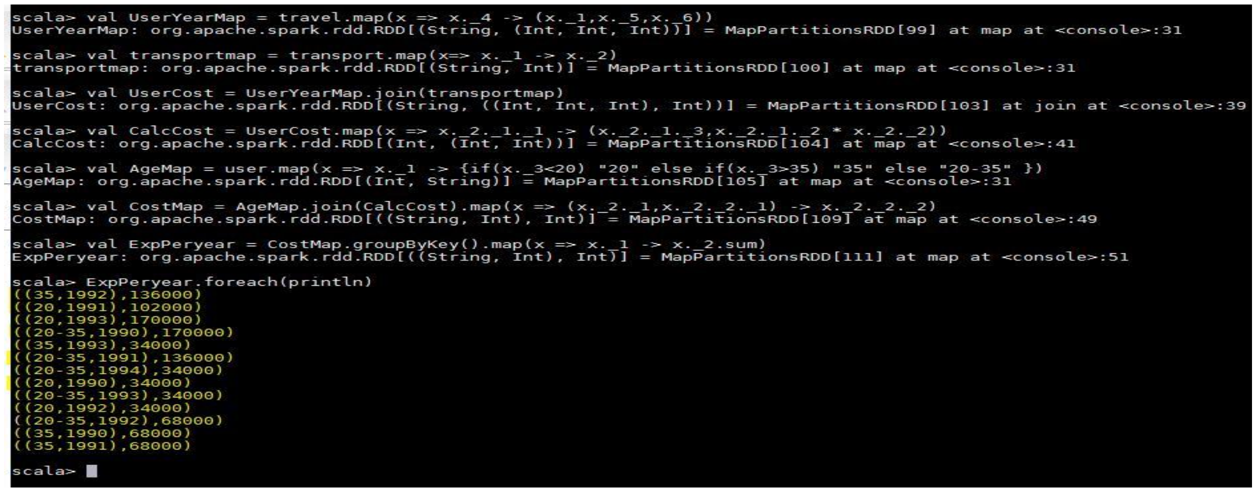
```
scala> val travel = baseRDD1.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5).toInt))

travel: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[78] 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[79] 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[80] at map at <console>:29

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[81] at map at <console>:31

scala> val userMap = travel.map(x => x._4 -> (x._1,x._5))
userMap: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[82] at map at <console>:31

scala> val transportmap = transport.map(x=> x._1 -> x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[83] at map at <console>:31

scala> val joinCost = userMap.join(transportmap)
joinCost: org.apache.spark.rdd.RDD[(String, ((Int, Int), Int))] = MapPartitionsRDD[86] 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[87] 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[89] 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[93] 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[95] at map at <console>:53

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

scala> 
```

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

Below is the code used:-

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

Output:-



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

scala> val transportmap = transport.map(x=> x._1 -> x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[100] at map at <console>:31

scala> val UserCost = UserYearMap.join(transportmap)
UserCost: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[103] 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[104] 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[105] 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[109] 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[111] at map at <console>:51

scala> ExpPeryear.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> 
```