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**

Below data sets are used for this assignment

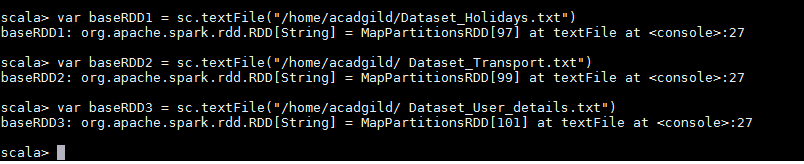


Below that, we are loading the dataset nto the spark context

var baseRDD1 = sc.textFile(“/home/acadgild/Dataset\_Holidays.txt”)

var baseRDD2 = sc.textFile(“/home/acadgild/Dataset\_Transport.txt”)

var baseRDD3 = sc.textFile(“/home/acadgild/Dataset\_User\_details.txt”)



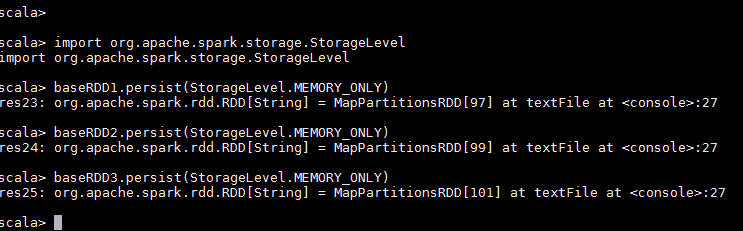
importing the singleton object for controlling the storage of RDD ,

import org.apache.spark.storage.StorageLevel

baseRDD1.persist(StorageLevel.MEMORY\_ONLY)

baseRDD2.persist(StorageLevel.MEMORY\_ONLY)

baseRDD3.persist(StorageLevel.MEMORY\_ONLY)

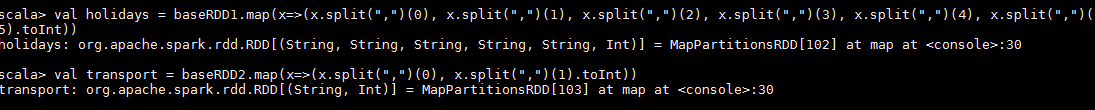


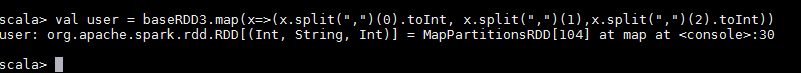
**Loading of datasets into name of holidays, transport and user RDD’s**

val holidays = 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))





**1) Considering age groups of < 20 , 20-35, 35 > ,Which age group spends the most**

**amount of money travelling.**

val agemap = user.map(x=>x.\_1->

{

If (x.\_3<20)

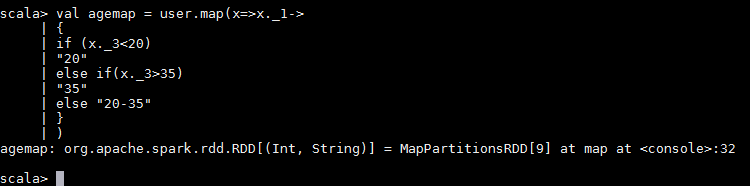
“20”

else if (x.\_3>35)

“35”

else “20-35”

})

****

val usermap =holidays.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)

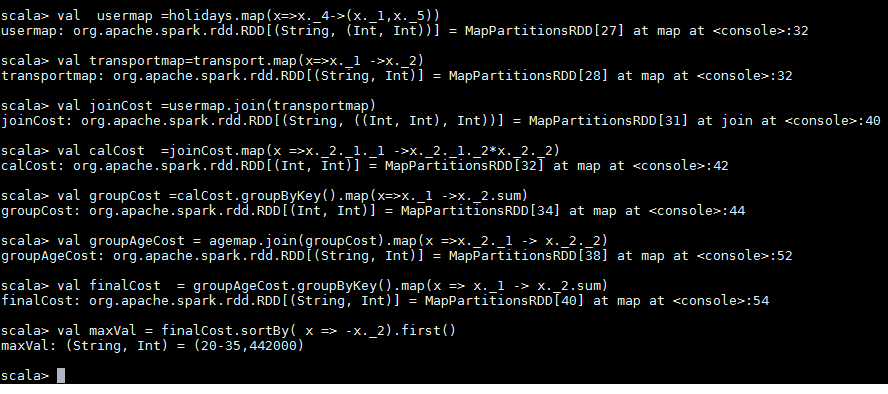
val groupCost =calCost.groupByKey().map(x=>x.\_1 ->x.\_2.sum)

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 screen**



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

val userholidays = holidays.map(x=>x.\_4 -> (x.\_1,x.\_5,x.\_6))

val usertransport = transport.map(x=>x.\_1->x.\_2)

val join1 = userholidays.join(usertransport)

val costdistamount = join1.map(x=>x.\_2.\_1.\_1 ->(x.\_2.\_1.\_3,x.\_2.\_1.\_2\*x.\_2.\_2))

val join2 = agemap.join(costdistamount).map(x=>(x.\_2.\_1,x.\_2.\_2.\_1) -> x.\_2.\_2.\_2)

val expeachagegroup = join2.groupByKey().map(x=>x.\_1->x.\_2.sum)

**output screen**

