2. 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.

Below data sets are used for this assignment

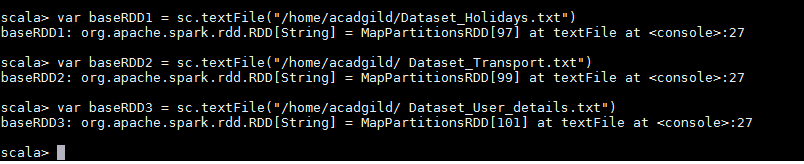


Below that, we are loading the dataset nto the spark context

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”)



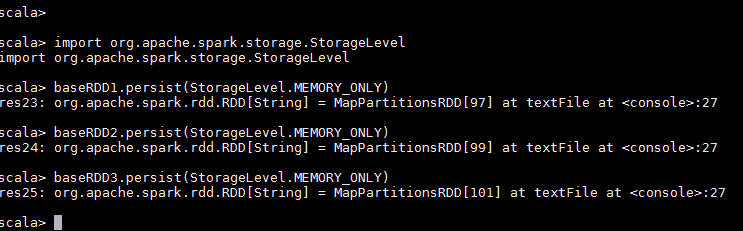
importing the singleton object for controlling the storage of RDD ,

import org.apache.spark.storage.StorageLevel

baseRDD1.persist(StorageLevel.MEMORY\_ONLY)

baseRDD2.persist(StorageLevel.MEMORY\_ONLY)

baseRDD3.persist(StorageLevel.MEMORY\_ONLY)

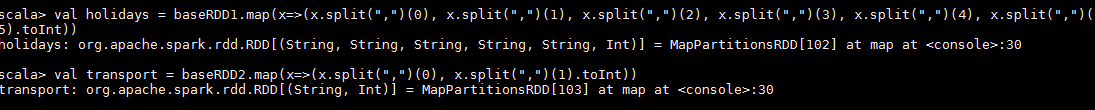


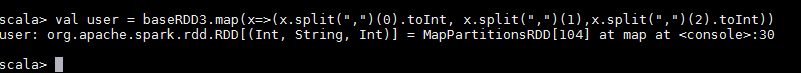
**Loading of datasets into name of holidays, transport and user RDD’s**

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

**Codes:**

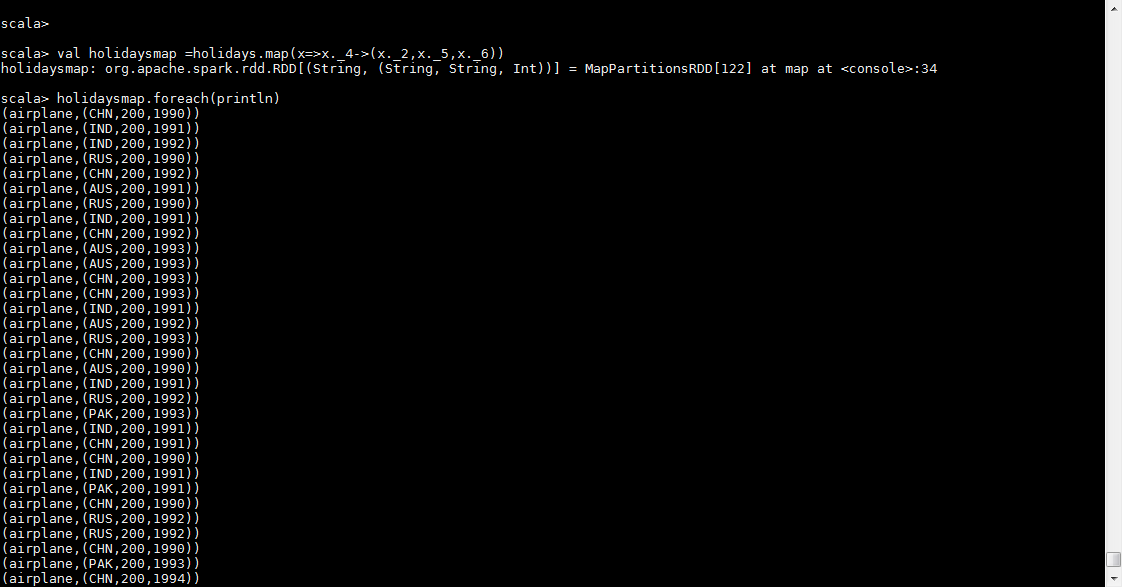
val holidaysmap =holidays.map(x=>x.\_4->(x.\_2,x.\_5,x.\_6))

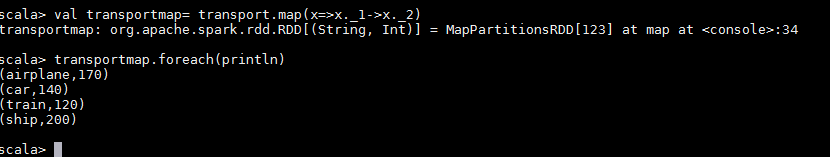
val transportmap= transport.map(x=>x.\_1->x.\_2)

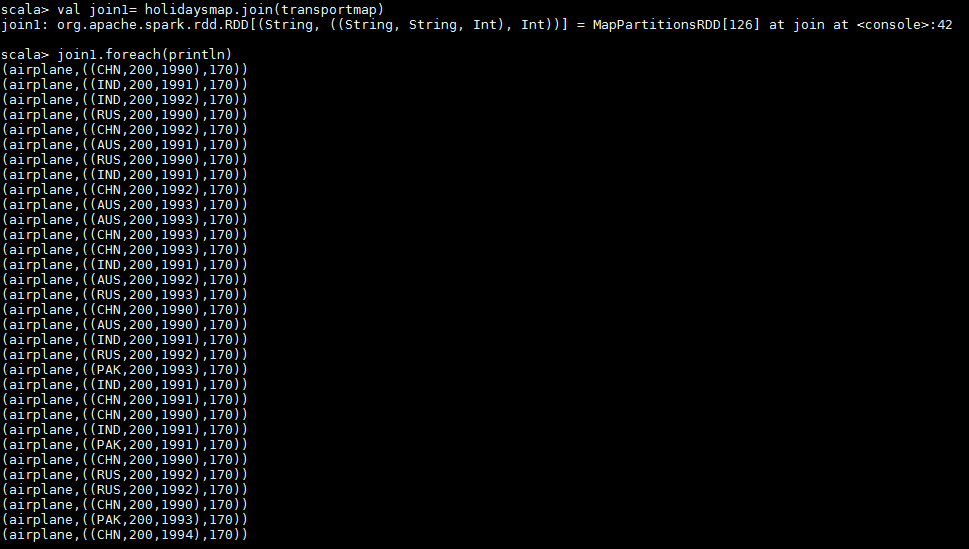
val join1= holidaysmap.join(transportmap)

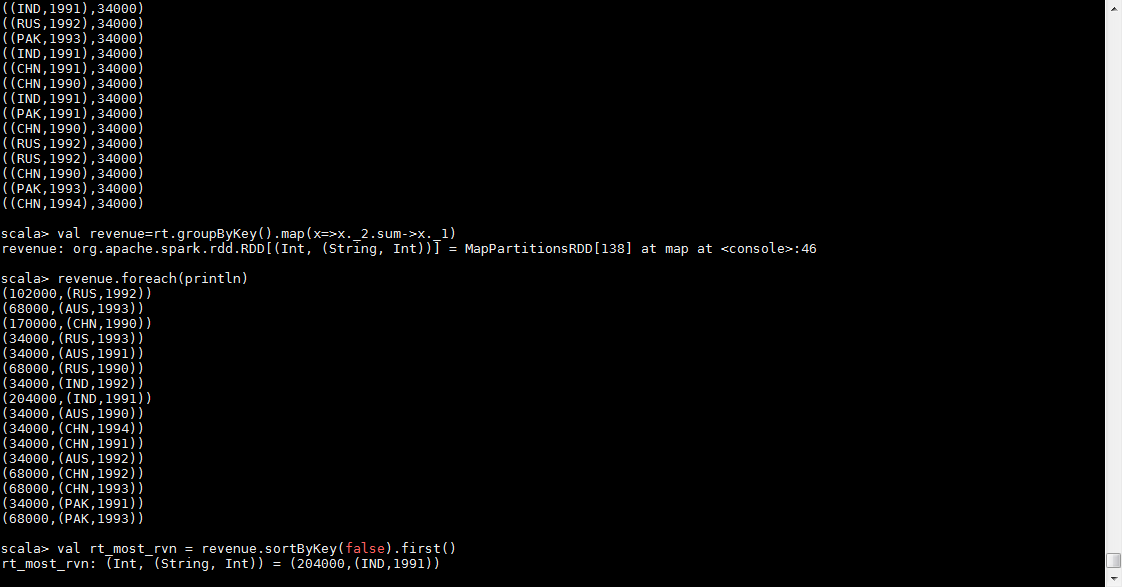
val rt =join1.map(x=>(x.\_2.\_1.\_1->x.\_2.\_1.\_3)->(x.\_2.\_1.\_2\*x.\_2.\_2))

val revenue=rt.groupByKey().map(x=>x.\_2.sum->x.\_1)

val rt\_most\_rvn = revenue.sortByKey(false).first()







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

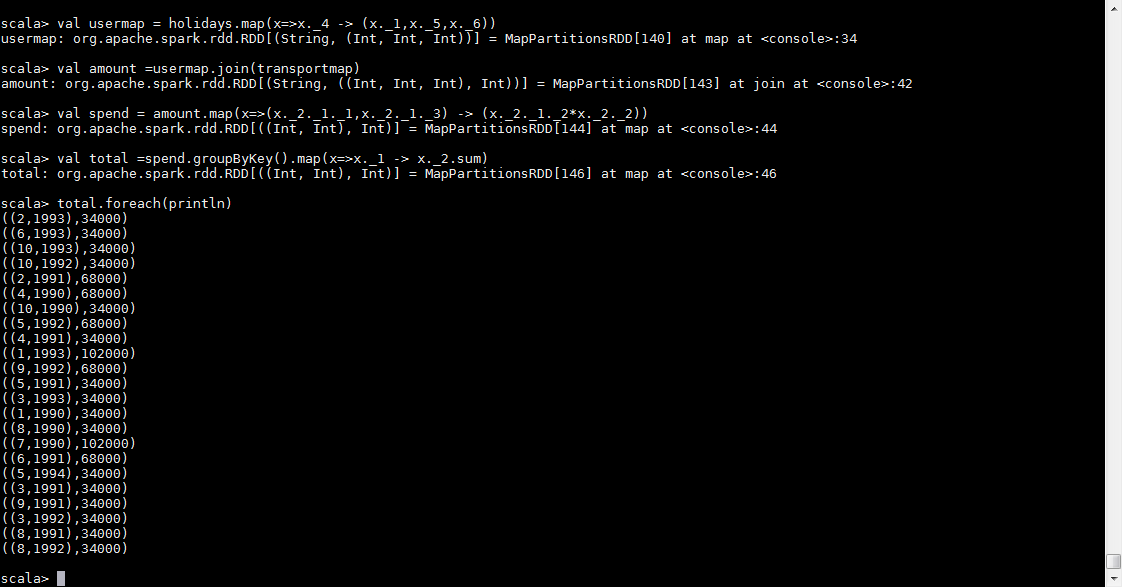
val usermap = holidays.map(x=>x.\_4 -> (x.\_1,x.\_5,x.\_6))

val amount =usermap.join(transportmap)

val spend = amount.map(x=>(x.\_2.\_1.\_1,x.\_2.\_1.\_3) -> (x.\_2.\_1.\_2\*x.\_2.\_2))

val total =spend.groupByKey().map(x=>x.\_1 -> x.\_2.sum)

**output screen**



**3)) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most**

**every year.**

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

{

If (x.\_3<20)

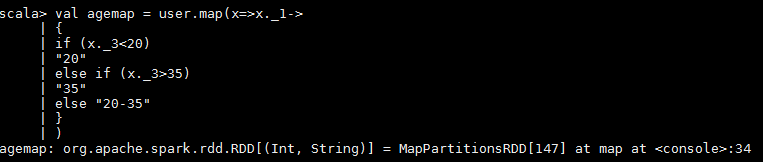
“20”

else if (x.\_3>35)

“35”

else “20-35”

})



val userid = holidays.map(x=>x.\_1 ->1)

val joinmap1 = agemap.join(userid)

val joinmap2 = joinmap1.map(x=>x.\_2.\_1 -> x.\_2.\_2)

val groupkey = joinmap2.groupByKey.map(x=>x.\_1 ->x.\_2.sum)

val mostGroup = groupkey.sortBy(x => -x.\_2).first()

**output screen**

