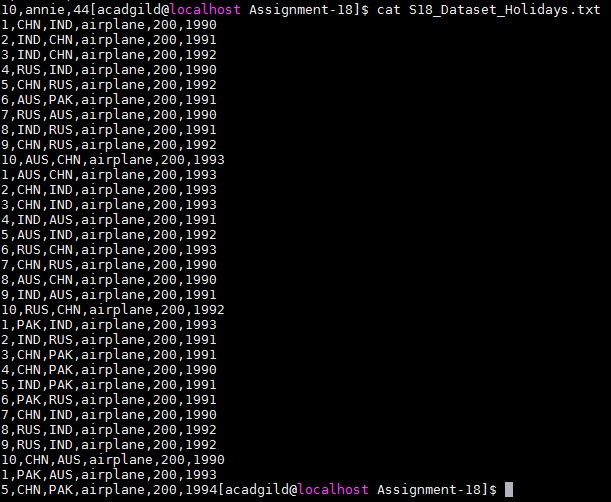
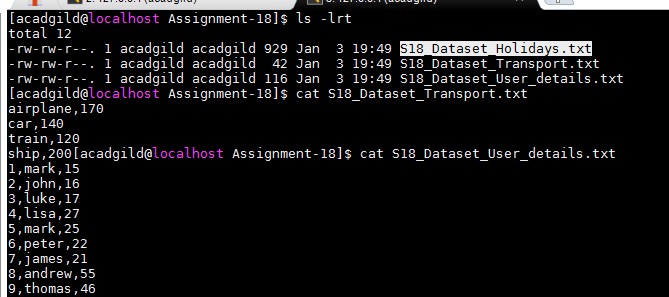
Assignment 18.2

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



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-

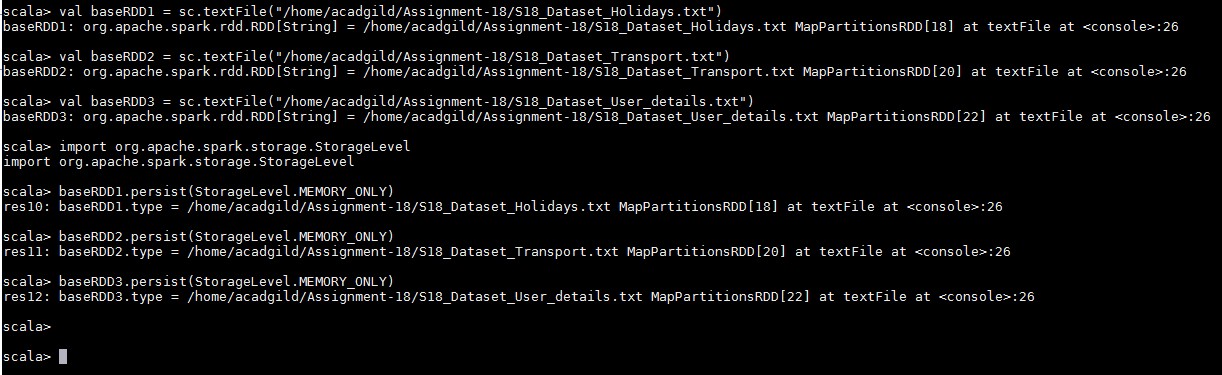
18/S18\_Dataset\_User\_details.txt")

import org.apache.spark.storage.StorageLevel

baseRDD1.persist(StorageLevel.MEMORY\_ONLY)

baseRDD2.persist(StorageLevel.MEMORY\_ONLY)

baseRDD3.persist(StorageLevel.MEMORY\_ONLY)



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.

Solution:-

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

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 travelmap = travel.map(x=> x.\_4 -> (x.\_2,x.\_5,x.\_6))

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

val join1 = travelmap.join(transportmap)

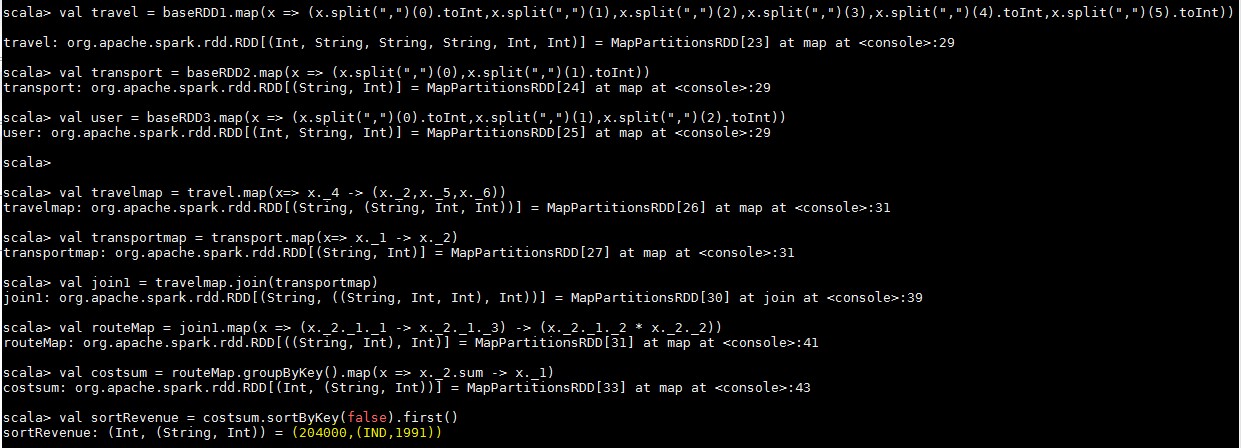
val routeMap = join1.map(x => (x.\_2.\_1.\_1 -> x.\_2.\_1.\_3) -> (x.\_2.\_1.\_2 \*

x.\_2.\_2))

val costsum = routeMap.groupByKey().map(x => x.\_2.sum -> x.\_1)

val sortRevenue = costsum.sortByKey(false).first()

Output:-



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

Below is the code used:-

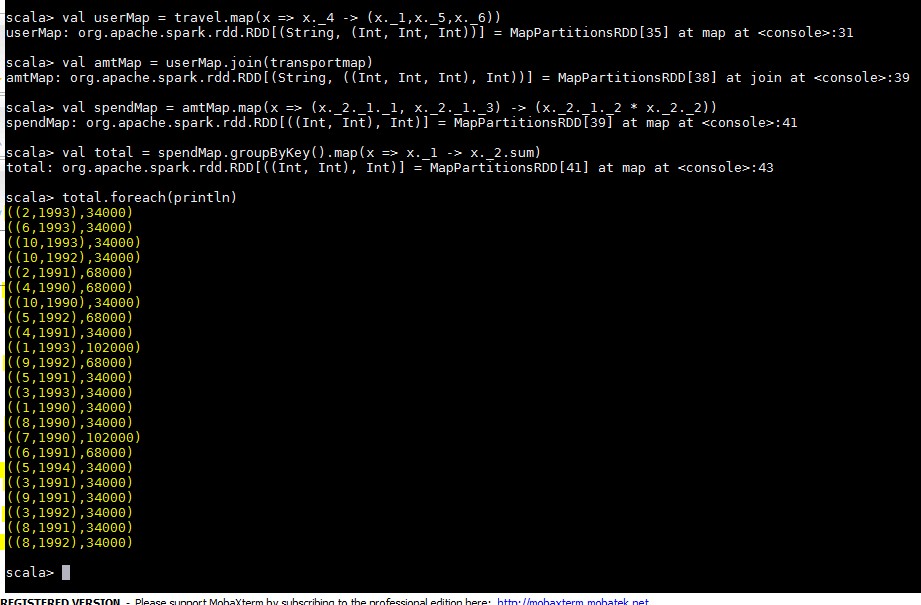
val userMap = travel.map(x => x.\_4 -> (x.\_1,x.\_5,x.\_6))

val amtMap = userMap.join(transportmap)

val spendMap = amtMap.map(x => (x.\_2.\_1.\_1, x.\_2.\_1.\_3) -> (x.\_2.\_1.\_2 \* x.\_2.\_2))

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

total.foreach(println) Output:-



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

Below is the code used:-

val AgeMap = user.map(x => x.\_1 -> {if(x.\_3<20) "20" else if(x.\_3>35) "35" else "20-35" })

val UIDMap = travel.map(x => x.\_1 -> 1)

val joinMap = AgeMap.join(UIDMap)

val joinMap2 = joinMap.map(x => x.\_2.\_1 -> x.\_2.\_2)

val groupKey = joinMap2.groupByKey.map(x => x.\_1 -> x.\_2.sum)

val maxVal = groupKey.sortBy(x => -x.\_2).first() Output:-

