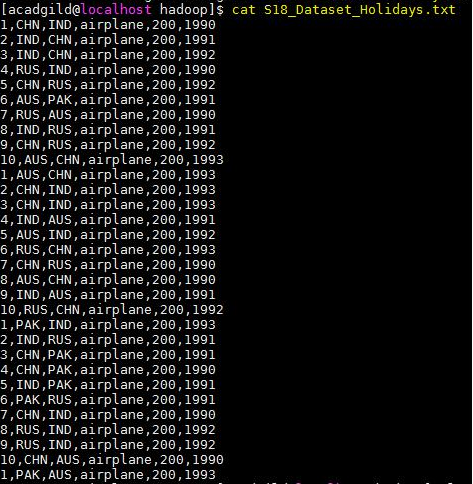
Dataset used,

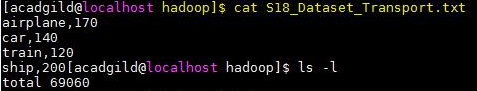
S18\_Dataset\_Holidays.txt,



S18\_Dataset\_User\_details.txt,



S18\_Dataset\_Transport.txt,



The above dataset has the data column wise, userID, Destination, arrival, travel mode, travel distance and the year,

We are loading the dataset into the spark context,

***val baseRDD1 = sc.textFile("/home/acadgild/hadoop/S18\_Dataset\_Holidays.txt")***

***val baseRDD2 = sc.textFile("/home/acadgild/hadoop/S18\_Dataset\_Transport.txt")***

***val baseRDD3 = sc.textFile("/home/acadgild/hadoop/S18\_Dataset\_User\_details.txt")***

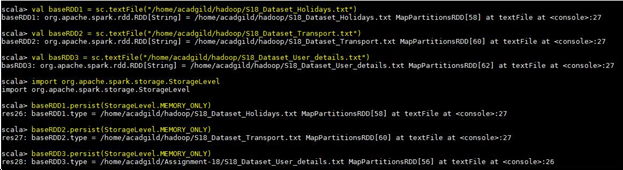
Importing the singleton object for controlling the storage of an RDD,

***import org.apache.spark.storage.StorageLevel***

***baseRDD1.persist(StorageLevel.MEMORY\_ONLY)***

***baseRDD2.persist(StorageLevel.MEMORY\_ONLY)***

***baseRDD3.persist(StorageLevel.MEMORY\_ONLY)***

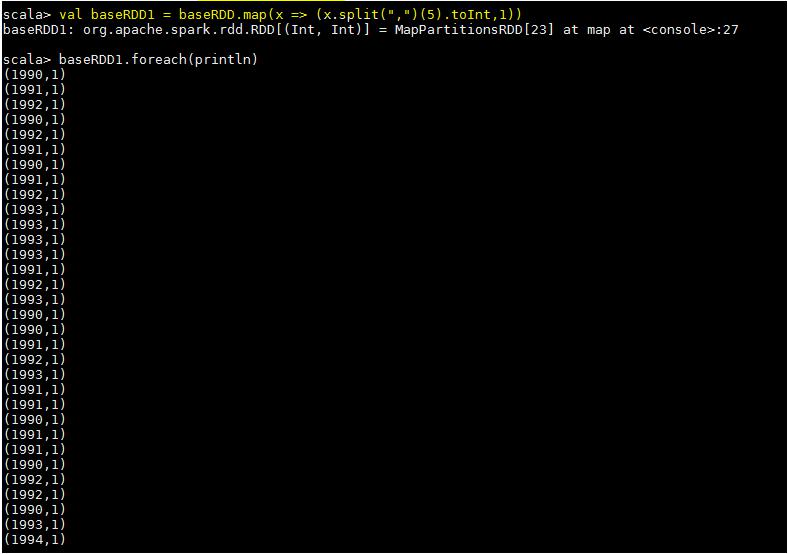
******

Task 1 - What is the distribution of the total number of air-travellers per year?

Codes used to achieve the above,

1. ***val baseRDD1 = baseRDD.map(x => (x.split(",")(5).toInt,1))***
2. ***val no\_air\_travelers = baseRDD1.reduceByKey((x,y)=>(x+y)).foreach(println)***

we are creating a tuple RDD baseRDD1 and mapping the key with numerical value 1.



We are reducing the number of occurrences using reduceByKey and printing the result. Therefore, Total no of air travelers per year is,

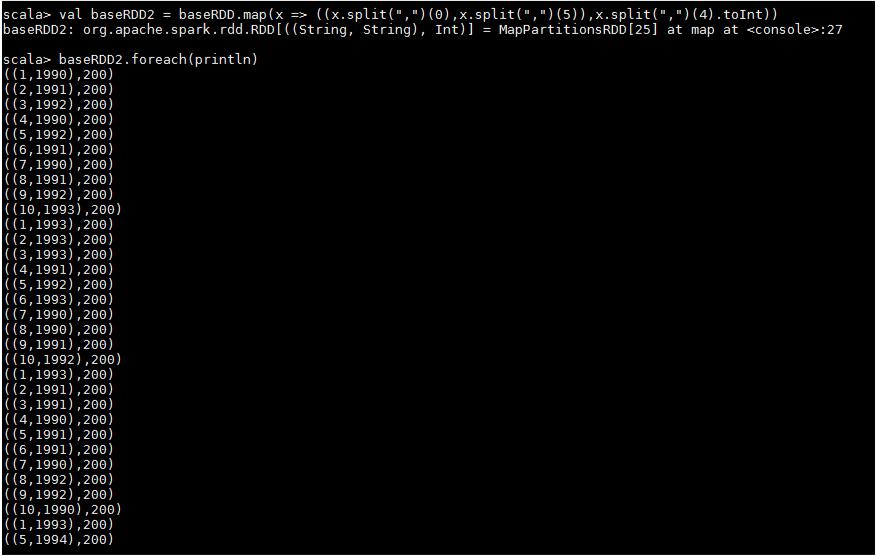


Task 2 - What is the total air distance covered by each user per year?

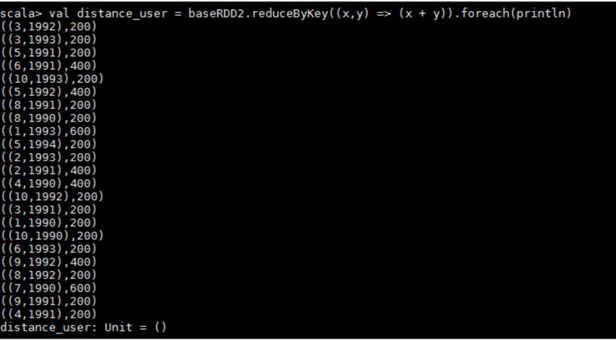
Codes used to achieve the above,

1. ***val baseRDD2 = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))***
2. ***val distance\_user = baseRDD2.reduceByKey((x,y) => (x + y)).foreach(println)***

We are creating a tuple rdd “baseRDD2” and mapping the key and value. Here the userID, year acts as key and the travel distance is value.



In the second step, we are reducing the number of occurrences using reduceByKey and printing the result, therefore the total air distance covered by each user per year is show below in the screenshot,



Task 3 - Which user has travelled the largest distance till date?

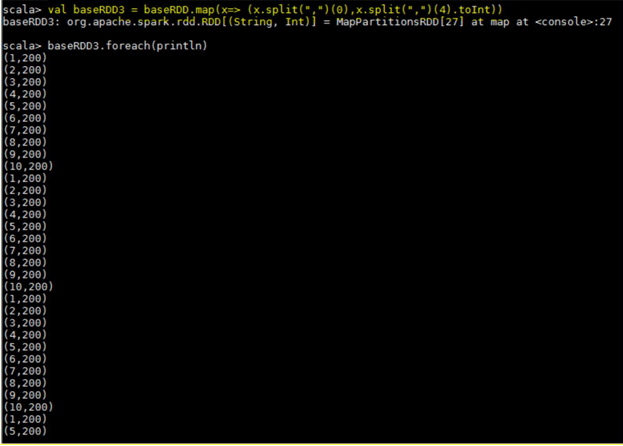
Codes used below,

1. ***val baseRDD3 = baseRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))***
2. ***val largest\_dist = baseRDD3.reduceByKey((x,y)=>(x+y)).takeOrdered(1)***

The tuple rdd “baseRDD3” is created to map the key and value from the baseRDD. Here the userID and is key and the travel distance is value,

In the 2nd step, we are reducing the number of occurrences using reduceByKey and using the takeOrdered function to get the result,

***largest\_dist: Array[(String, Int)] = Array((1,800))***



The required output,



Task 4 – What is the most preferred destination for all users?

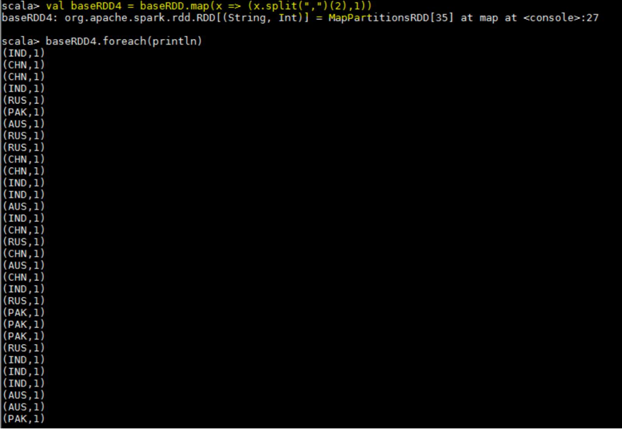
Codes used below,

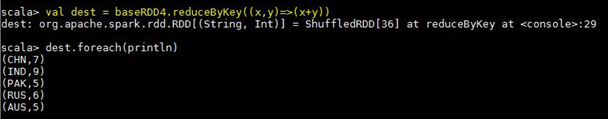
1. ***val baseRDD4 = baseRDD.map(x => (x.split(",")(2),1))***
2. ***val dest = baseRDD4.reduceByKey((x,y)=>(x+y))***
3. ***val dest =***

***baseRDD4.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(\_.\_2))***

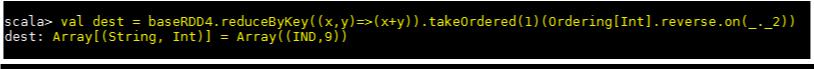
A tuple rdd created with the destination as key and numerical 1 as value, and we are reducing the number of occurrences using the reduceByKey. Now, the most preferred destination is taken by using the function takeOrdered and ordering the values descending so that we can get the required output.

The output of the each step is shown below,





The required output,

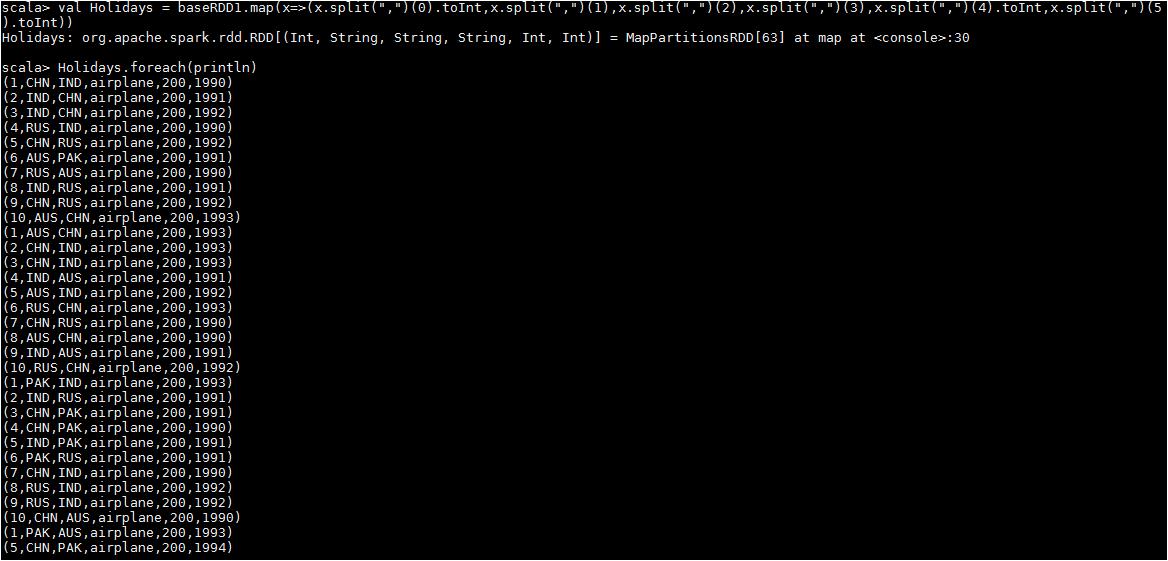


<Extra steps before performing Task5 onwards>

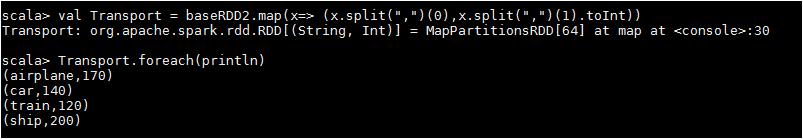
We are loading the dataset’s in the name of **holidays**, **transport** and **user** RDD’s.

***val holidays =***

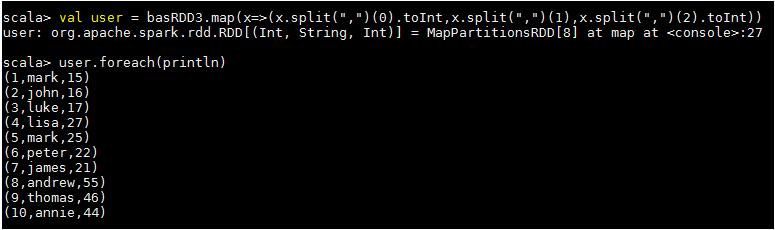
***baseRDD1.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.s plit(",")(5).toInt))***

******

***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))***

******

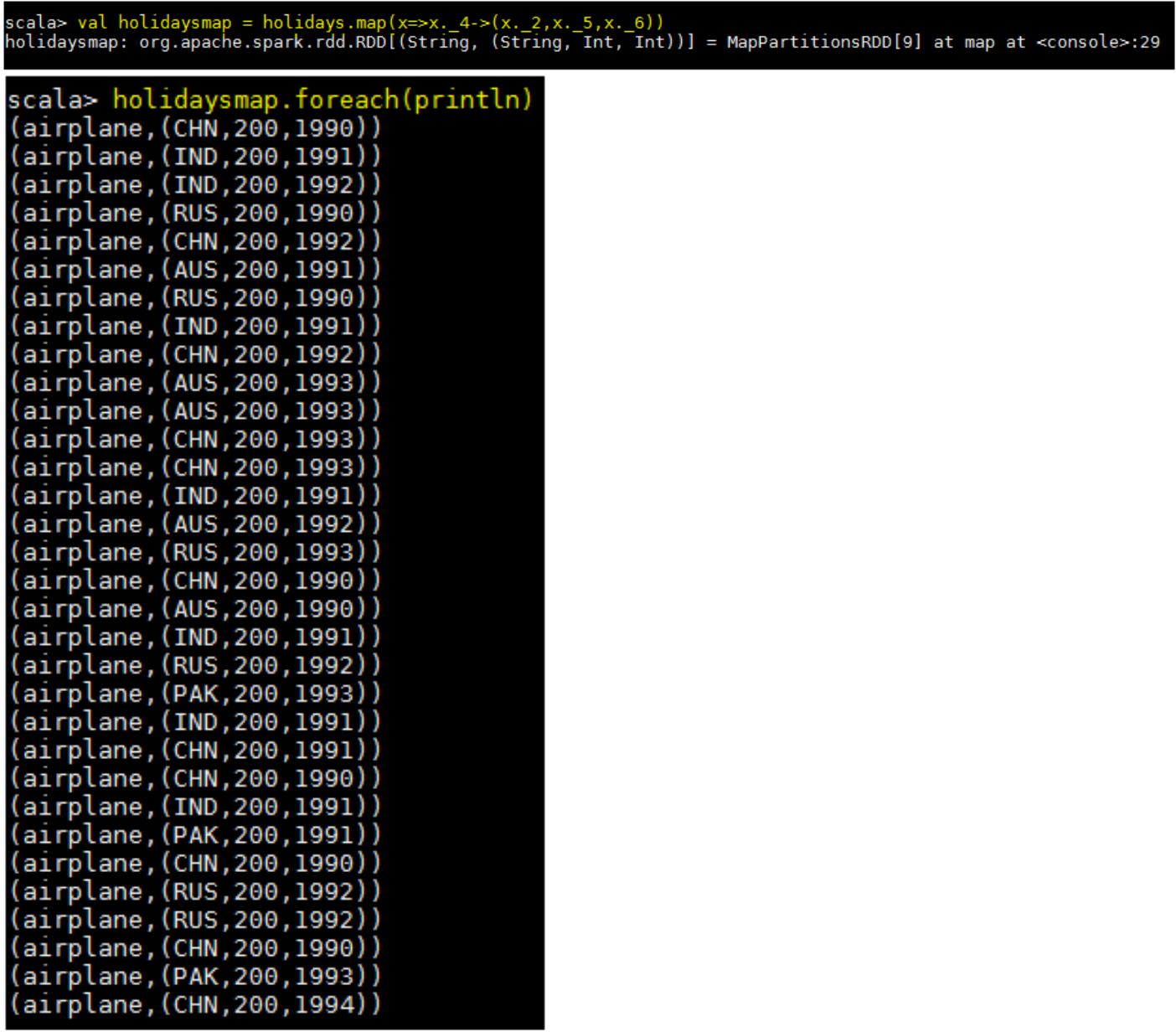
Task 5 - Which route is generating the most revenue per year?

Codes used below,

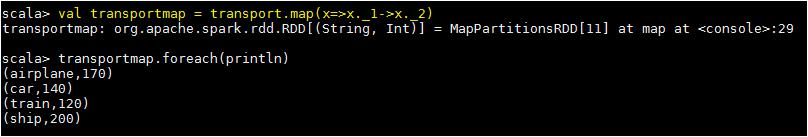
1. ***val holidaysmap = holidays.map(x=>x.\_4->(x.\_2,x.\_5,x.\_6))***
2. ***val transportmap = transport.map(x=>x.\_1->x.\_2)***
3. ***val join1 = holidaysmap.join(transportmap)***
4. ***val route = join1.map(x=>(x.\_2.\_1.\_1->x.\_2.\_1.\_3)->(x.\_2.\_1.\_2\*x.\_2.\_2))***
5. ***val revenue = route.groupByKey().map(x=>x.\_2.sum->x.\_1)***
6. ***val routemostrevenue = revenue.sortByKey(false).first()***

**result*: routemostrevenue: (Int, (String, Int)) = (204000,(IND,1991))***

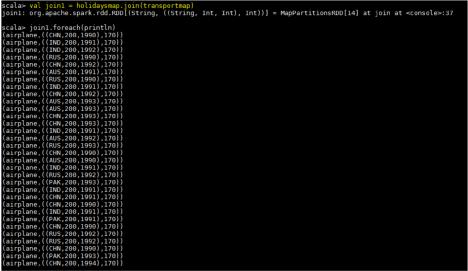
**Step 1** –we are mapping the key and value from the base RDD holidays as travel mode as key and thedestination, distance and the year as value.



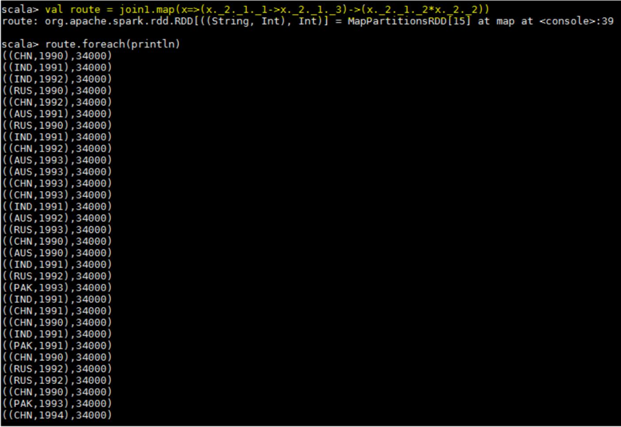
**Step -2** –same as, we are creating a tuple RDD as travel mode as key and the rate as values, shown below.



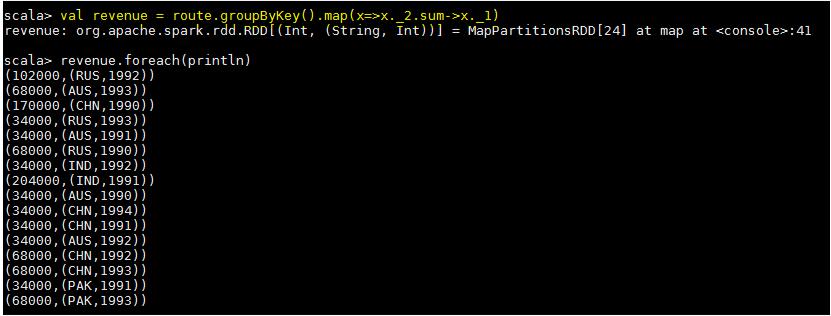
**Step 3** – we are joining the 2 RDD’s holidaysmap and the transport using join function,



**Step 4** –we are mapping the new RDD join1 as below, destination & year as key and the values asmultiplication of the cost and the distance.



Step 5 – using groupByKey function, we are grouping the destination & year with the sum of the costs.



Expected output,

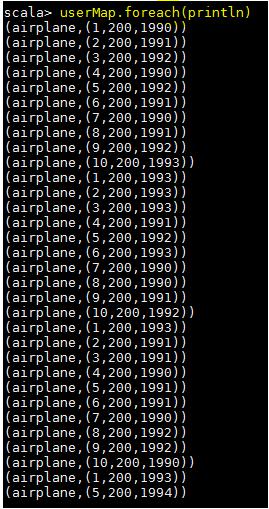


Task 6 - What is the total amount spent by every user on air-travel per year?

The codes used for this task below,

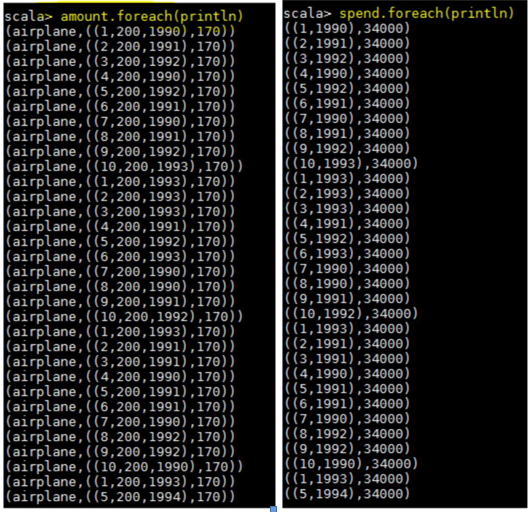
1. ***val userMap = holidays.map(x => x.\_4 -> (x.\_1,x.\_5,x.\_6))***
2. ***val amount = userMap.join(transportmap)***
3. ***val spend = amount.map(x => (x.\_2.\_1.\_1, x.\_2.\_1.\_3) -> (x.\_2.\_1.\_2 \* x.\_2.\_2))***
4. ***val total = spend.groupByKey().map(x => x.\_1 -> x.\_2.sum)***

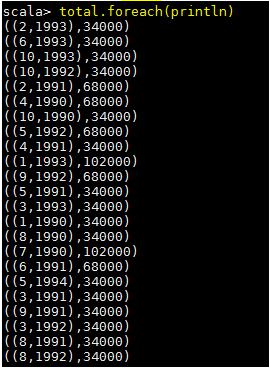
Step -1 – we are creating a tuple rdd from a baseRDD ***holidays*** making the travel mode as Key and the userID, distance & year as values.



Step -2 – we are joining the created tuple RDD **userMap** with the already created tuple RDD **transportMap** using the join function.

Step – 3 – now, we are calculating the expenditure for each user by multiplying the distance and the amount spent for the travel mode airplane,



In the final step, we are summing the total value for each user yearly wise, please see the expected result in the below screen shot.

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

The codes used for this task are below,

In Order to considering particular age groups, we are using a below if, else logic to define a RDD **AgeMap** which gives you a set of age groups,

***val AgeMap = user.map(x=>x.\_1->***

***| {***

***| if(x.\_3<20)***

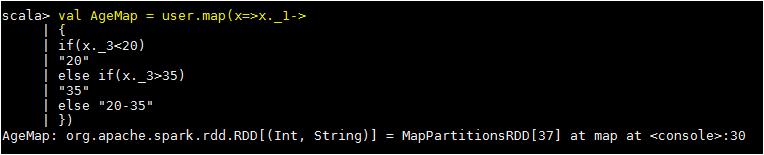
***| "20"***

***| else if(x.\_3>35)***

***| "35"***

***| else "20-35"***

***| })***

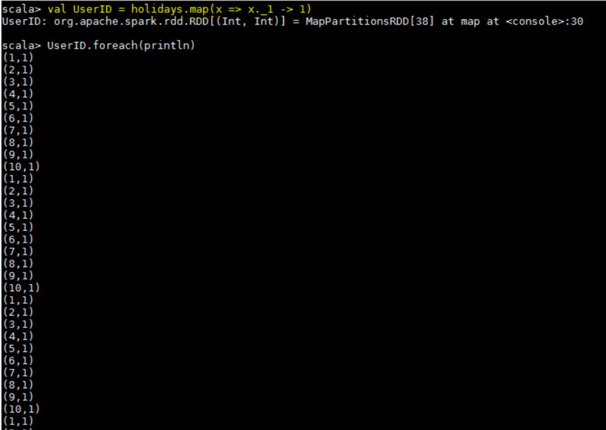
******

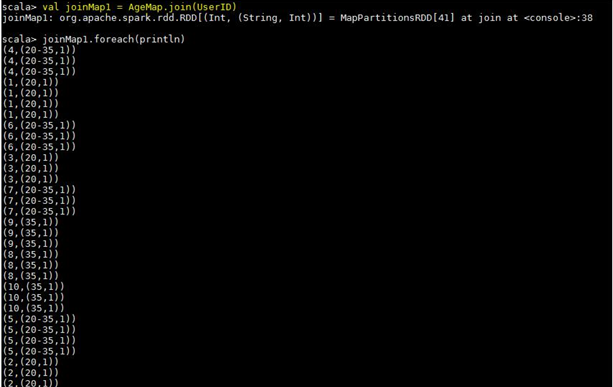
Actual codes,

1. ***val UserID = holidays.map(x => x.\_1 -> 1)***
2. ***val joinMap1 = AgeMap.join(UserID)***
3. ***val joinMap2 = joinMap1.map(x => x.\_2.\_1 -> x.\_2.\_2)***
4. ***val groupKey = joinMap2.groupByKey.map(x => x.\_1 -> x.\_2.sum)***
5. ***val mostGroup = groupKey.sortBy(x => -x.\_2).first()***

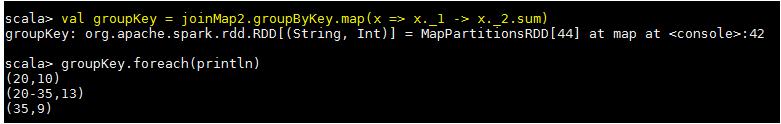
Step – 1- we are just mapping the user ID from the RDD **holidays** with the numerical 1.

Step -2 – In this step, we are joining the 2 RDD’s **UserID** and the **AgeMap,** So we are getting the below tuple RDD





Step – 4 – we just summed the total value by grouping the Age Group,



Step -5 - We use the function **first()** to find the age group who is travelling the most every year from the given dataset. The expected output shown below,

