1. Problem Statement

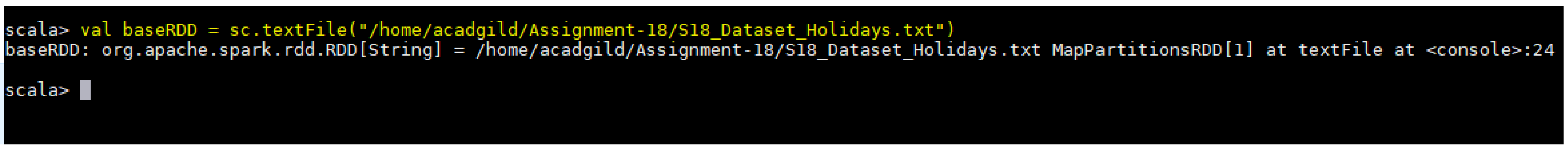
1) What is the distribution of the total number of air-travelers per year

Solution-

We will use Caching to store the files, since these files are being used again and again. Below is the command used to find the distribution-

- val baseRDD = sc.textFile("/home/acadgild/Assignment-8/S18_Dataset_Holidays.txt")
- import org.apache.spark.storage.StorageLevel
- baseRDD.persist(StorageLevel.MEMORY_ONLY)
- val splitRDD = baseRDD.map(x => (x.split(",")(5).toInt,1))
- val countSplit = splitRDD.reduceByKey((x,y) => (x + y))
- countSplit.foreach(println)

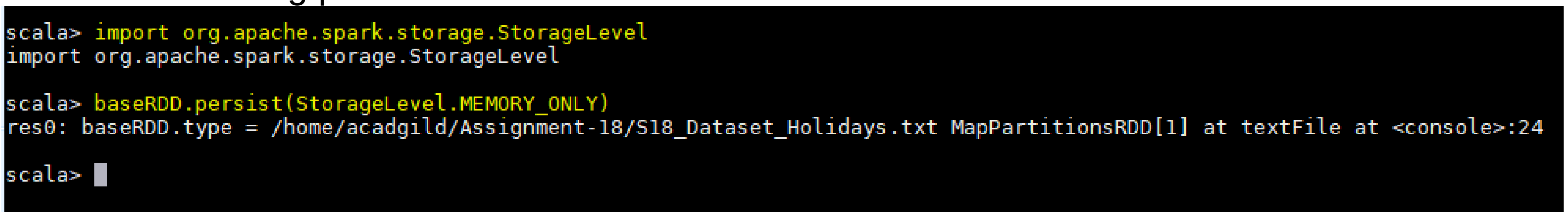
Now we will look into the code one by one. First we are reading a file from local using spark context-



```
scala> val baseRDD = sc.textFile("/home/acadgild/Assignment-18/S18_Dataset_Holidays.txt")
baseRDD: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-18/S18_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

scala>
```

Then we are using persist to cache the file-

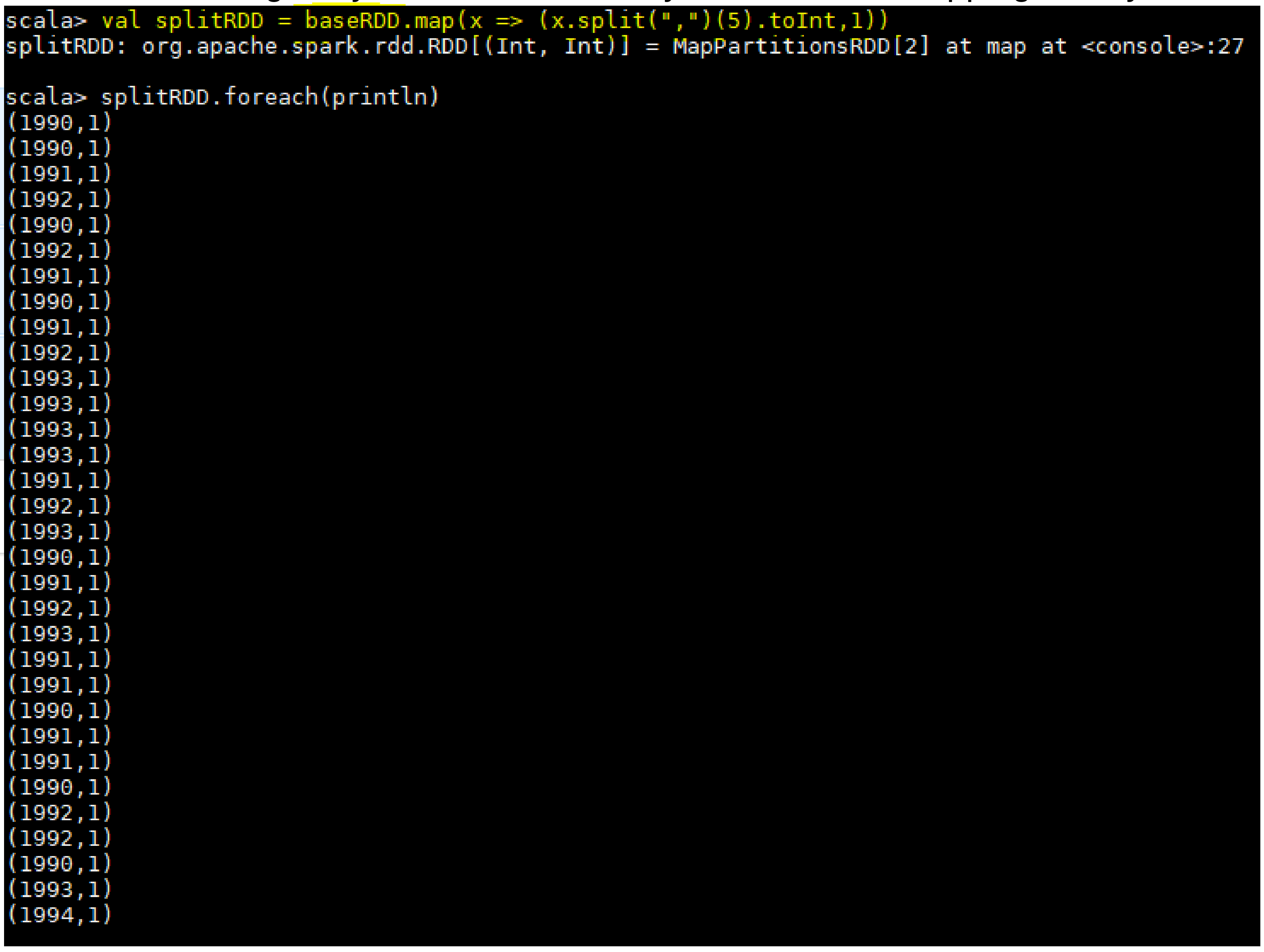


```
scala> import org.apache.spark.storage.StorageLevel
import org.apache.spark.storage.StorageLevel

scala> baseRDD.persist(StorageLevel.MEMORY_ONLY)
res0: baseRDD.type = /home/acadgild/Assignment-18/S18_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

scala>
```

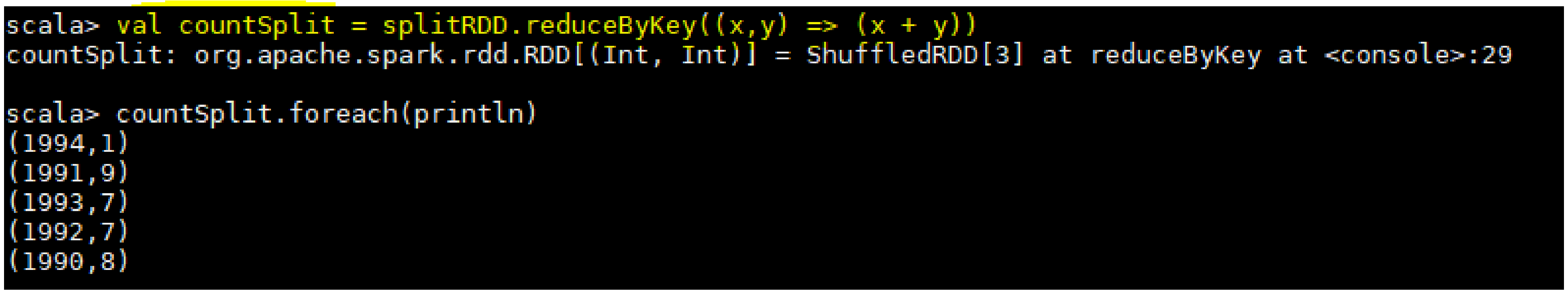
Now we are taking the year from the Holidays dataset and mapping each year with 1-



```
scala> val splitRDD = baseRDD.map(x => (x.split(",")(5).toInt,1))
splitRDD: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[2] at map at <console>:27

scala> splitRDD.foreach(println)
(1990,1)
(1990,1)
(1991,1)
(1992,1)
(1990,1)
(1992,1)
(1991,1)
(1990,1)
(1991,1)
(1992,1)
(1993,1)
(1993,1)
(1993,1)
(1993,1)
(1991,1)
(1992,1)
(1993,1)
(1990,1)
(1991,1)
(1992,1)
(1993,1)
(1991,1)
(1991,1)
(1990,1)
(1991,1)
(1991,1)
(1990,1)
(1992,1)
(1992,1)
(1990,1)
(1993,1)
(1994,1)
```

Now we are using reduceByKey to above RDD to add the count of each year to find the number of passengers travelled in that particular year-



```
scala> val countSplit = splitRDD.reduceByKey((x,y) => (x + y))
countSplit: org.apache.spark.rdd.RDD[(Int, Int)] = ShuffledRDD[3] at reduceByKey at <console>:29

scala> countSplit.foreach(println)
(1994,1)
(1991,9)
(1993,7)
(1992,7)
(1990,8)
```

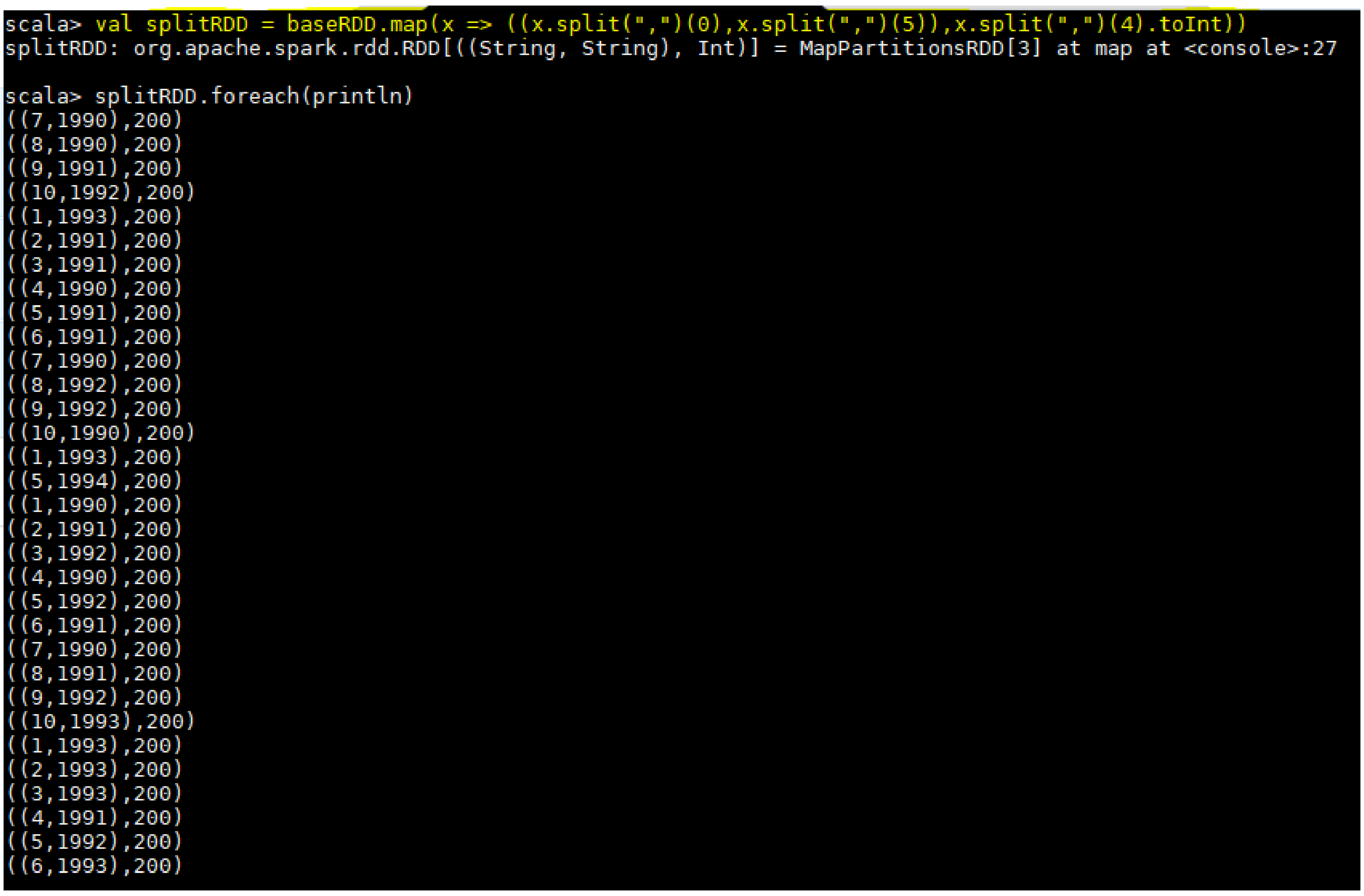
2) What is the total air distance covered by each user per year

Solution-

Here also we are using same baseRDD to find the result. Below is the code used-

- val splitRDD = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
- val distRDD = splitRDD.reduceByKey((x,y) => (x + y))
- distRDD.foreach(println)

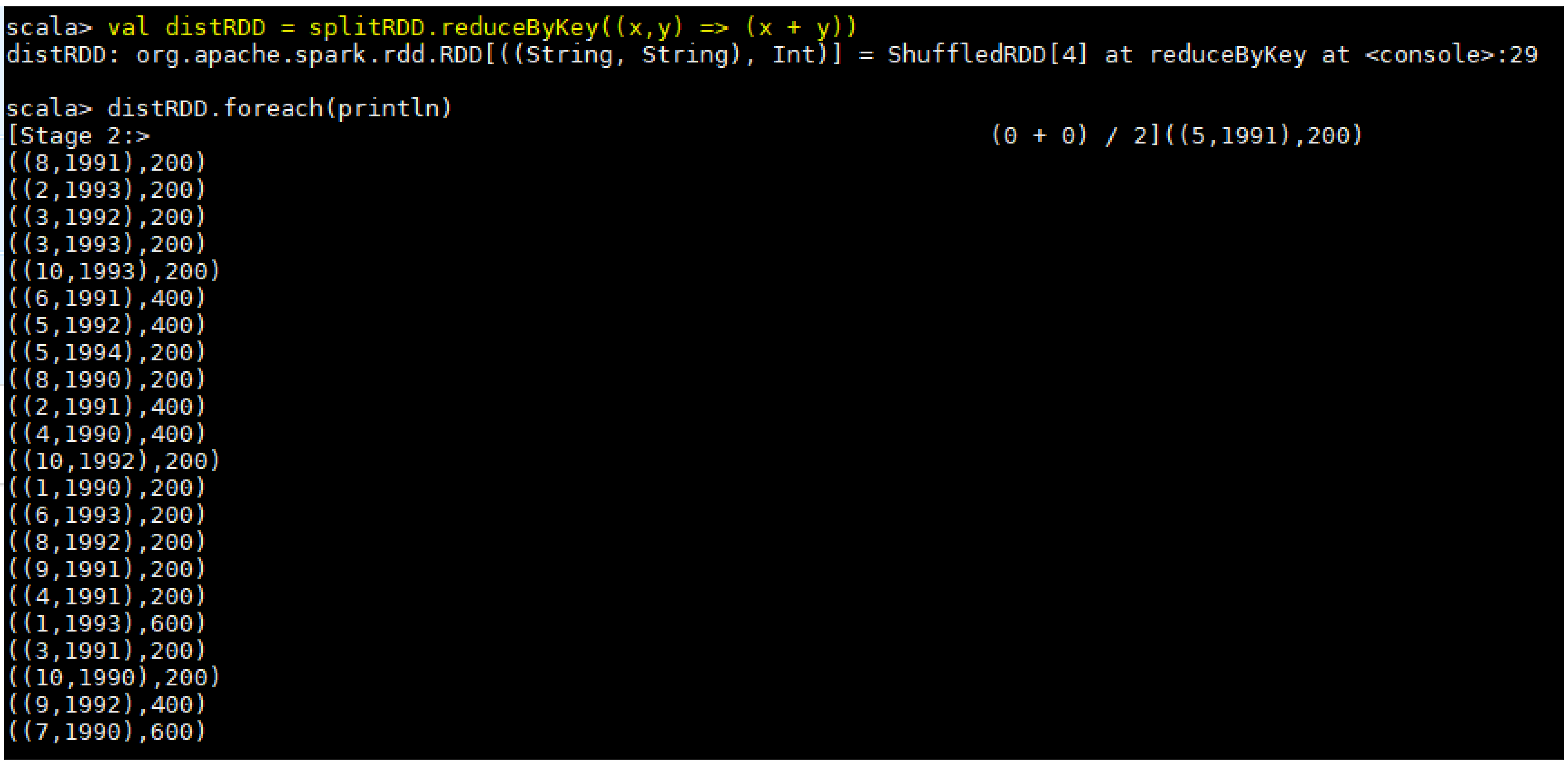
First we are extracting User ID and year as Key and distance as value in the form of Paired RDD-



```
scala> val splitRDD = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
splitRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[3] at map at <console>:27

scala> splitRDD.foreach(println)
((7,1990),200)
((8,1990),200)
((9,1991),200)
((10,1992),200)
((1,1993),200)
((2,1991),200)
((3,1991),200)
((4,1990),200)
((5,1991),200)
((6,1991),200)
((7,1990),200)
((8,1992),200)
((9,1992),200)
((10,1990),200)
((1,1993),200)
((5,1994),200)
((1,1990),200)
((2,1991),200)
((3,1992),200)
((4,1990),200)
((5,1992),200)
((6,1991),200)
((7,1990),200)
((8,1991),200)
((9,1992),200)
((10,1993),200)
((1,1993),200)
((2,1993),200)
((3,1993),200)
((4,1991),200)
((5,1992),200)
((6,1993),200)
```

Now based on User ID and year we are calculating the total distance per user per year. Below screenshot shows the final result-



```
scala> val distRDD = splitRDD.reduceByKey((x,y) => (x + y))
distRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[4] at reduceByKey at <console>:29

scala> distRDD.foreach(println)
[Stage 2:>                                                    (0 + 0) / 2]((5,1991),200)
((8,1991),200)
((2,1993),200)
((3,1992),200)
((3,1993),200)
((10,1993),200)
((6,1991),400)
((5,1992),400)
((5,1994),200)
((8,1990),200)
((2,1991),400)
((4,1990),400)
((10,1992),200)
((1,1990),200)
((6,1993),200)
((8,1992),200)
((9,1991),200)
((4,1991),200)
((1,1993),600)
((3,1991),200)
((10,1990),200)
((9,1992),400)
((7,1990),600)
```

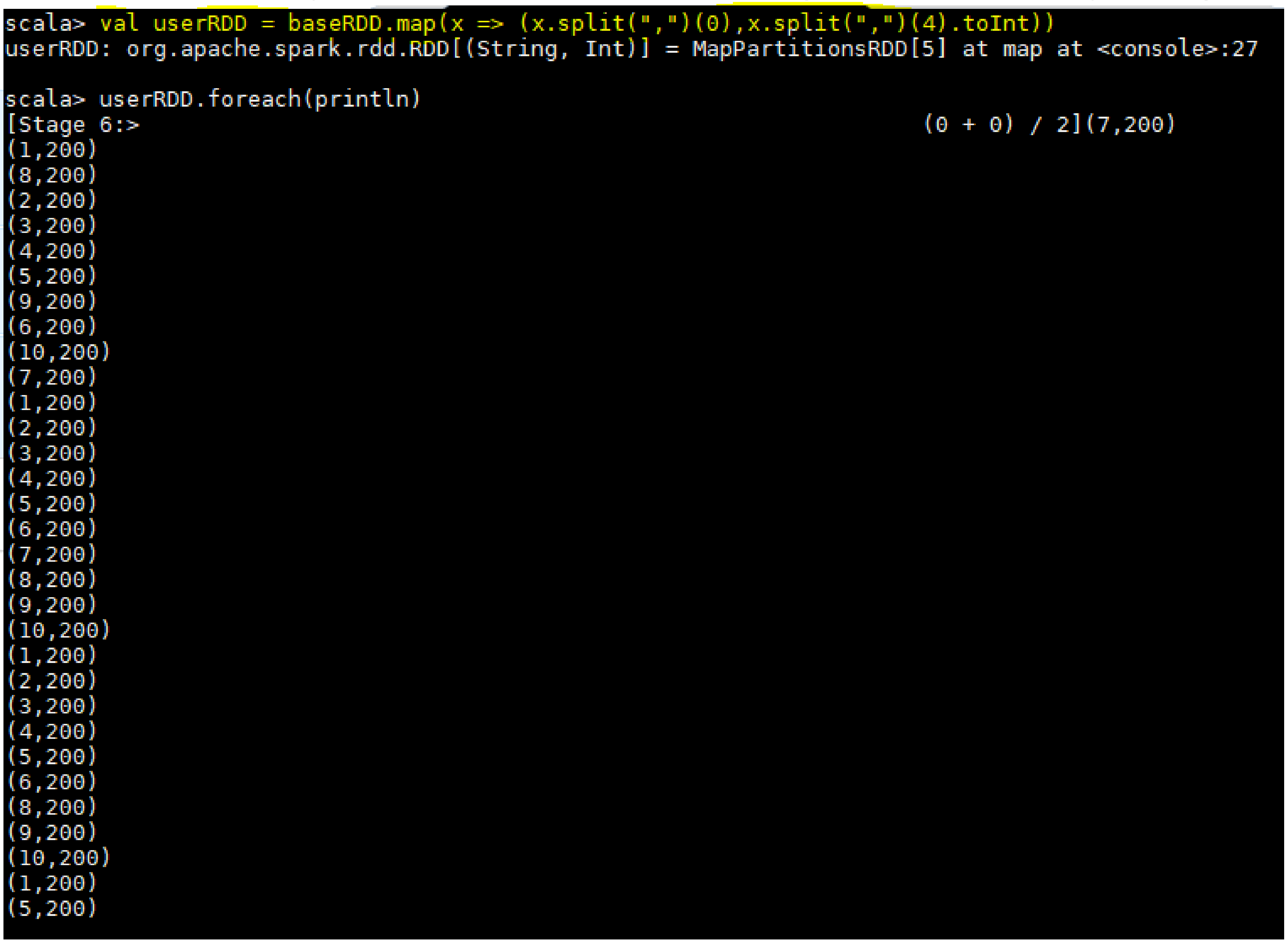
3) Which user has travelled the largest distance till date

Solution-

Here also we have used Caching. Below is the code used to find the result-

- val userRDD = baseRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))
- val totaldistRDD = userRDD.reduceByKey((x,y) => (x+y))
- val maxRDD = totaldistRDD.takeOrdered(1)

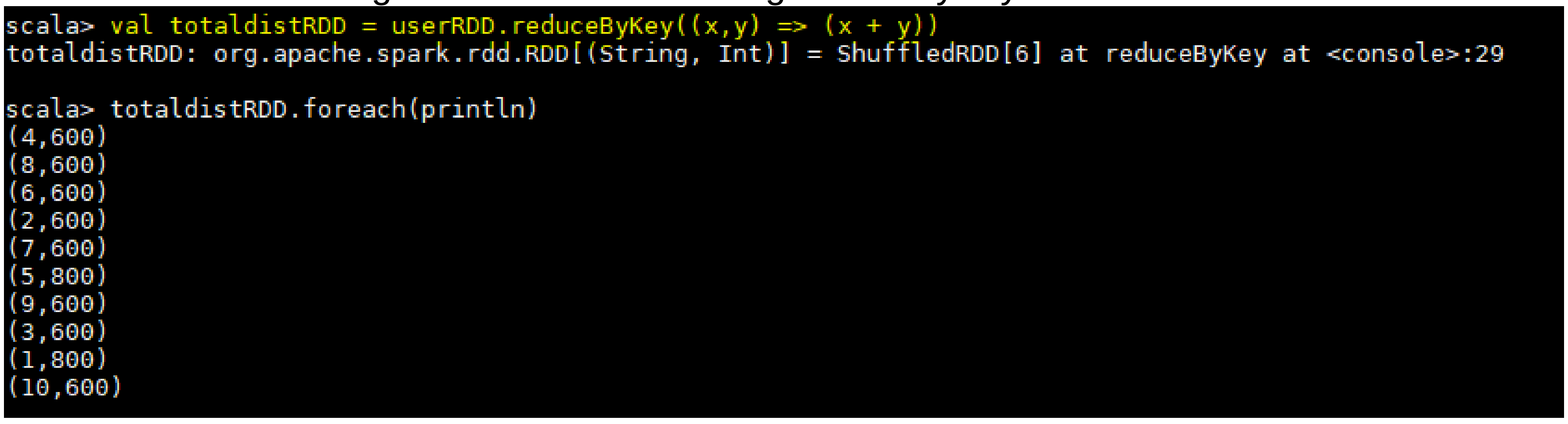
First we are extracting User ID and distance from the input file in form of key value pairs-



```
scala> val userRDD = baseRDD.map(x => (x.split(",")(0),x.split(",")(4).toInt))
userRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[5] at map at <console>:27

scala> userRDD.foreach(println)
[Stage 6:>                                                    (0 + 0) / 2](7,200)
(1,200)
(8,200)
(2,200)
(3,200)
(4,200)
(5,200)
(9,200)
(6,200)
(10,200)
(7,200)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(8,200)
(9,200)
(10,200)
(1,200)
(5,200)
```

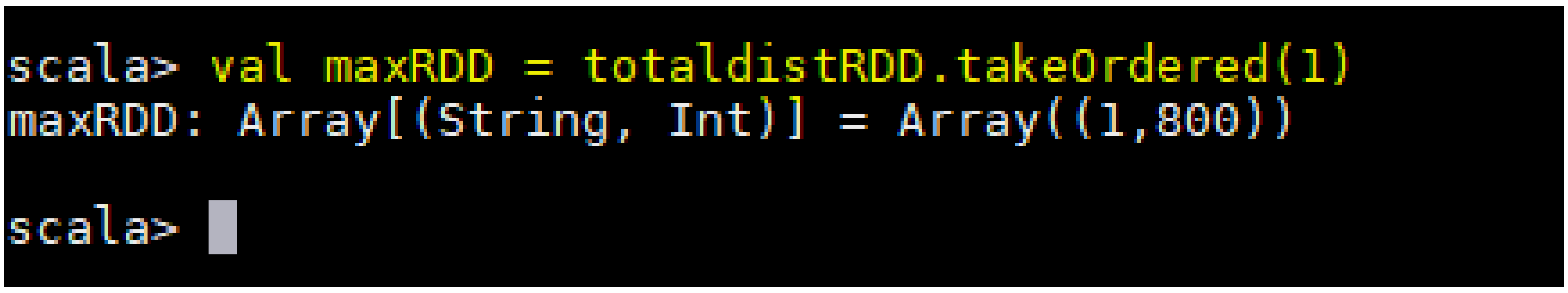
Then we are calculating the total distance using reduceByKey for each User



```
scala> val totaldistRDD = userRDD.reduceByKey((x,y) => (x + y))
totaldistRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[6] at reduceByKey at <console>:29

scala> totaldistRDD.foreach(println)
(4,600)
(8,600)
(6,600)
(2,600)
(7,600)
(5,800)
(9,600)
(3,600)
(1,800)
(10,600)
```

Below screenshot shows the final result-



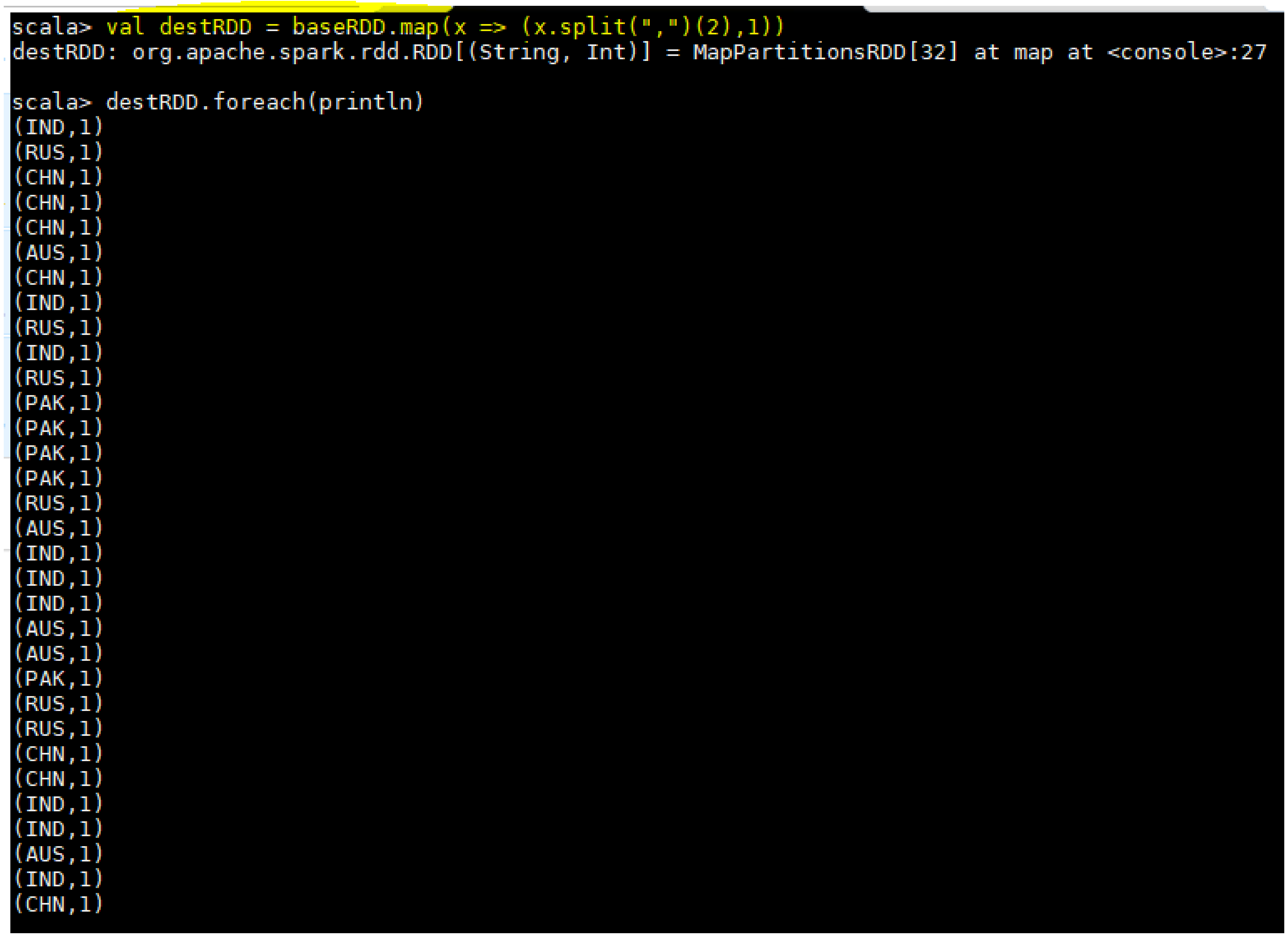
```
scala> val maxRDD = totaldistRDD.takeOrdered(1)
maxRDD: Array[(String, Int)] = Array((1,800))

scala> 
```

4) What is the most preferred destination for all users.

Solution-

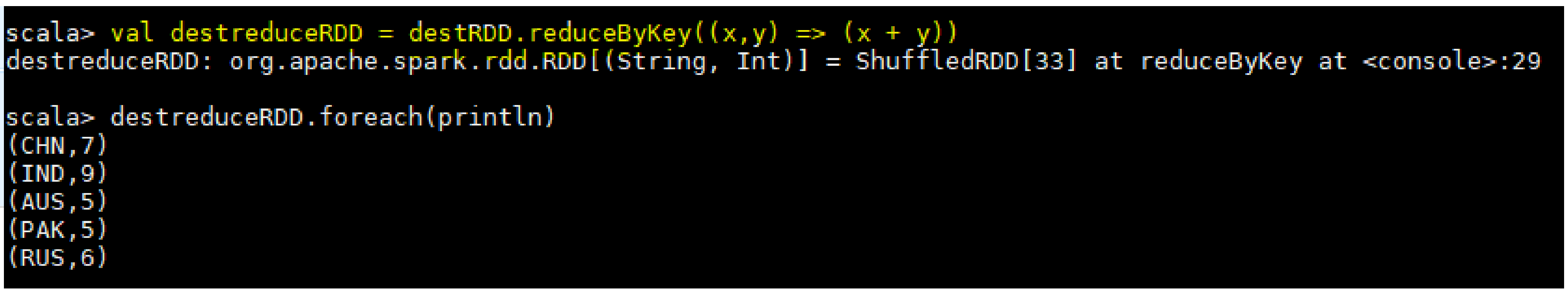
Here we are extracting destination from input file and mapping each destination with 1-



```
scala> val destRDD = baseRDD.map(x => (x.split(",")(2),1))
destRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[32] at map at <console>:27

scala> destRDD.foreach(println)
(IND,1)
(RUS,1)
(CHN,1)
(CHN,1)
(CHN,1)
(AUS,1)
(CHN,1)
(IND,1)
(RUS,1)
(IND,1)
(RUS,1)
(PAK,1)
(PAK,1)
(PAK,1)
(PAK,1)
(RUS,1)
(AUS,1)
(IND,1)
(IND,1)
(IND,1)
(AUS,1)
(AUS,1)
(PAK,1)
(RUS,1)
(RUS,1)
(CHN,1)
(CHN,1)
(IND,1)
(IND,1)
(AUS,1)
(IND,1)
(CHN,1)
```

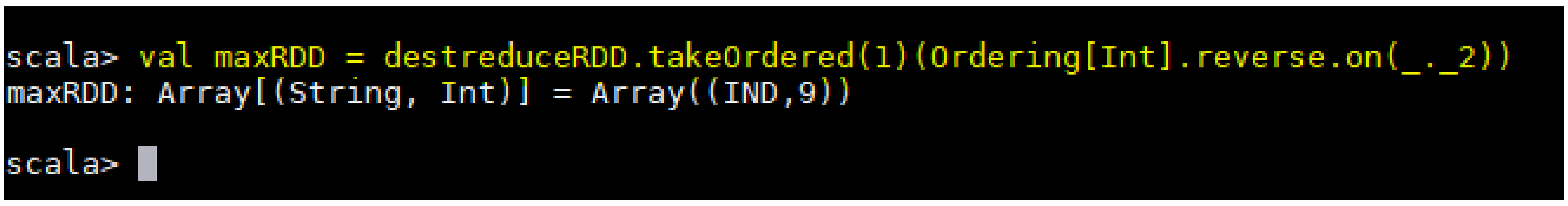
Now we adding each count for each destination-



```
scala> val destreduceRDD = destRDD.reduceByKey((x,y) => (x + y))
destreduceRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[33] at reduceByKey at <console>:29

scala> destreduceRDD.foreach(println)
(CHN,7)
(IND,9)
(AUS,5)
(PAK,5)
(RUS,6)
```

Below screenshot shows result for the most preferred destination which is India with highest frequency-



```
scala> val maxRDD = destreduceRDD.takeOrdered(1)(Ordering[Int].reverse.on(_._2))
maxRDD: Array[(String, Int)] = Array((IND,9))

scala> 
```