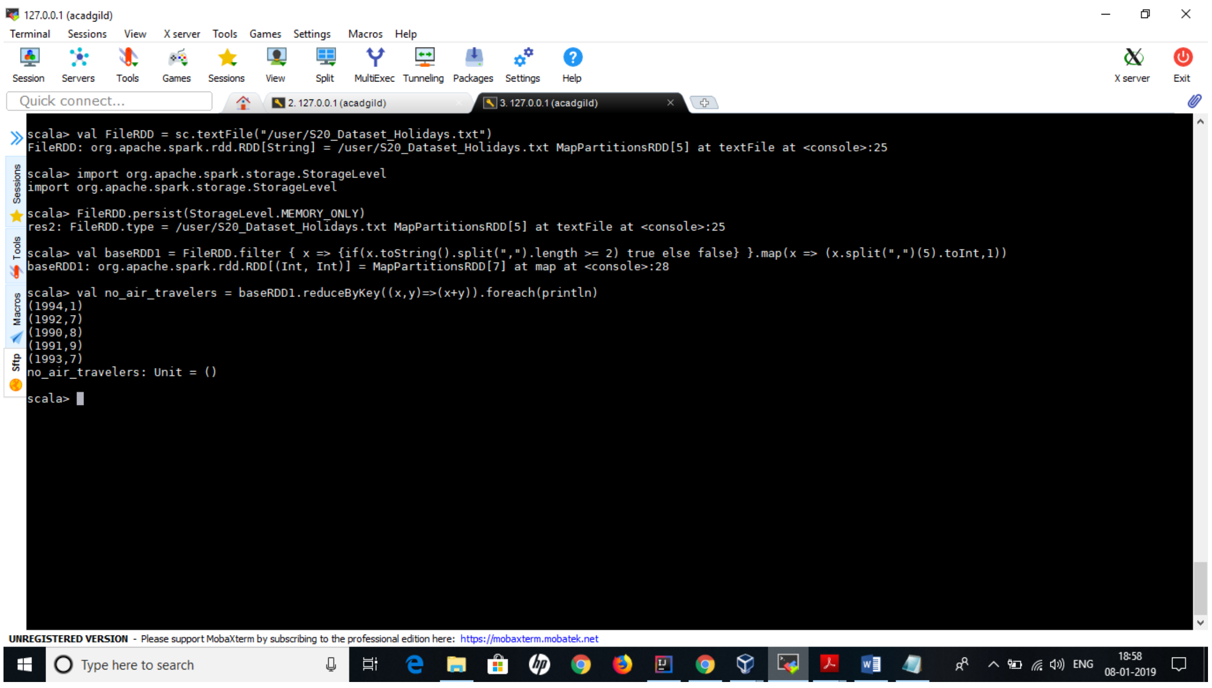
Assignment 20.1 Spark SQL 1

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

Solution:

> ***val FileRDD = sc.textFile("/user/S20_Dataset_Holidays.txt")***
>
> ***import org.apache.spark.storage.StorageLevel***
>
> ***FileRDD.persist(StorageLevel.MEMORY_ONLY)***
>
> ***val baseRDD1 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x => (x.split(",")(5).toInt,1))***
>
> ***val no_air_travelers = baseRDD1.reduceByKey((x,y)=>(x+y)).foreach(println)***

Output:


127.0.0.1 (acadgild)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 127.0.0.1 (acadgild)
3. 127.0.0.1 (acadgild)
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Type here to search
ENG 18:58 08-01-2019

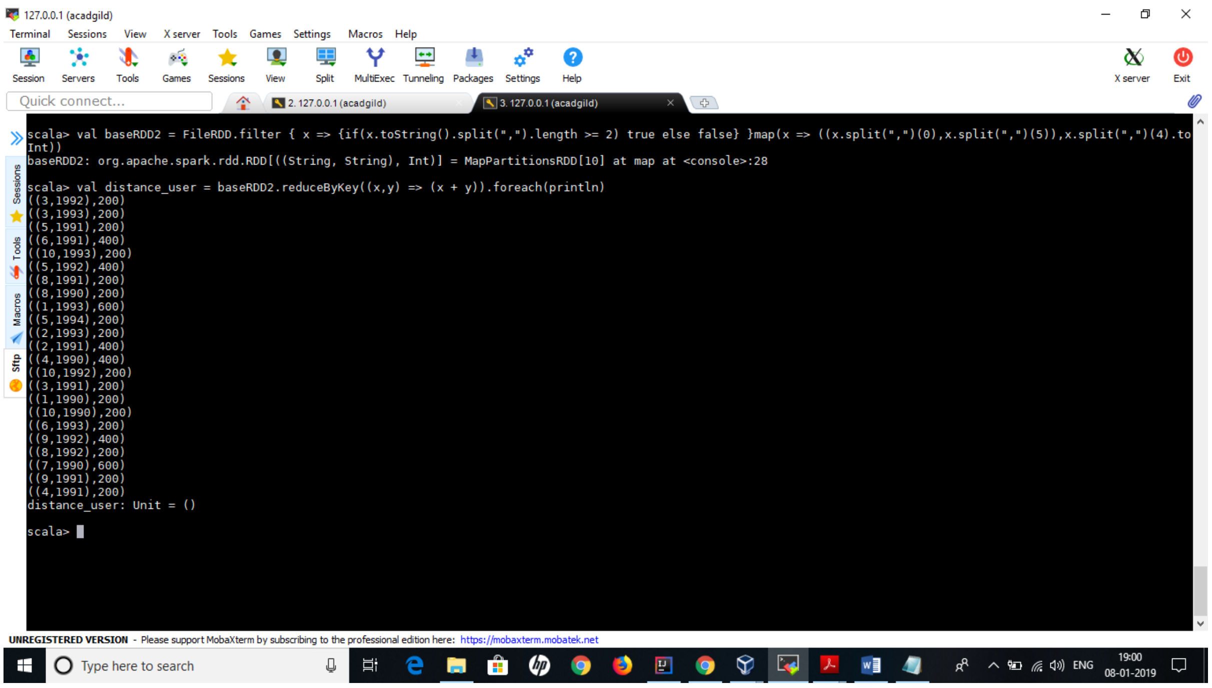
```
scala> val FileRDD = sc.textFile("/user/S20_Dataset_Holidays.txt")
FileRDD: org.apache.spark.rdd.RDD[String] = /user/S20_Dataset_Holidays.txt MapPartitionsRDD[5] at textFile at <console>:25

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

scala> FileRDD.persist(StorageLevel.MEMORY_ONLY)
res2: FileRDD.type = /user/S20_Dataset_Holidays.txt MapPartitionsRDD[5] at textFile at <console>:25

scala> val baseRDD1 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x => (x.split(",")(5).toInt,1))
baseRDD1: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[7] at map at <console>:28

scala> val no_air_travelers = baseRDD1.reduceByKey((x,y)=>(x+y)).foreach(println)
(1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)
no_air_travelers: Unit = ()

scala> 
```

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

Solution:

> ***val baseRDD2 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))***
>
> ***val distance_user = baseRDD2.reduceByKey((x,y) => (x + y)).foreach(println)***

Output:



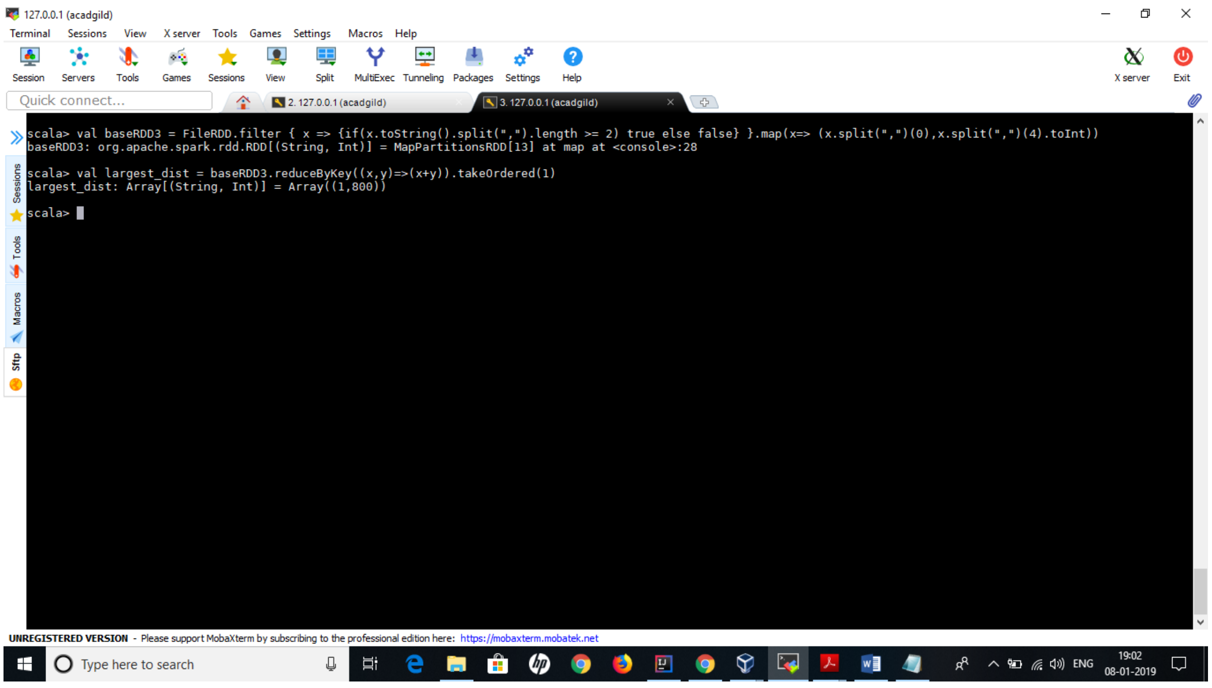
```
scala> val baseRDD2 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).to
Int))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[10] at map at <console>:28

scala> val distance_user = baseRDD2.reduceByKey((x,y) => (x + y)).foreach(println)
((3,1992),200)
((3,1993),200)
((5,1991),200)
((6,1991),400)
((10,1993),200)
((5,1992),400)
((8,1991),200)
((8,1990),200)
((1,1993),600)
((5,1994),200)
((2,1993),200)
((2,1991),400)
((4,1990),400)
((10,1992),200)
((3,1991),200)
((1,1990),200)
((10,1990),200)
((6,1993),200)
((9,1992),400)
((8,1992),200)
((7,1990),600)
((9,1991),200)
((4,1991),200)
distance_user: Unit = ()

scala> 
```

3. Which user has travelled the largest distance till date.

Solution:

> ***val baseRDD3 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=> (x.split(",")(0),x.split(",")(4).toInt))***
>
> ***val largest_dist = baseRDD3.reduceByKey((x,y)=>(x+y)).takeOrdered(1)***

Output:



```
scala> val baseRDD3 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x=> (x.split(",")(0),x.split(",")(4).toInt))
baseRDD3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[13] at map at <console>:28

scala> val largest_dist = baseRDD3.reduceByKey((x,y)=>(x+y)).takeOrdered(1)
largest_dist: Array[(String, Int)] = Array((1,800))

scala> 
```

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

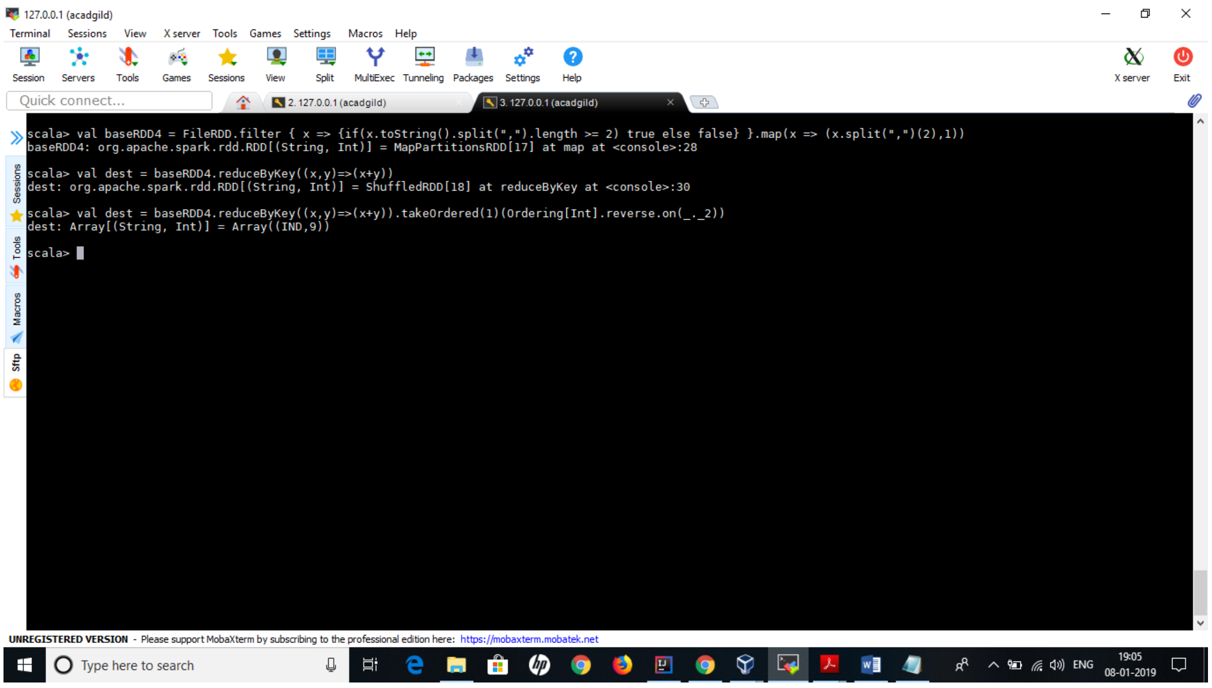
Solution:

val baseRDD4 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x => (x.split(",")(2),1))

val dest = baseRDD4.reduceByKey((x,y)=>(x+y))

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

Output:



```
scala> val baseRDD4 = FileRDD.filter { x => {if(x.toString().split(",").length >= 2) true else false} }.map(x => (x.split(",")(2),1))
baseRDD4: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[17] at map at <console>:28

scala> val dest = baseRDD4.reduceByKey((x,y)=>(x+y))
dest: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at reduceByKey at <console>:30

scala> val dest = baseRDD4.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(_._2))
dest: Array[(String, Int)] = Array((IND,9))

scala> 
```

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

Solution:

val FileRDD1 = sc.textFile("/user/S20_Dataset_Transport.txt")

val FileRDD2 = sc.textFile("/user/S20_Dataset_User_details.txt")

import org.apache.spark.storage.StorageLevel

FileRDD1.persist(StorageLevel.MEMORY_ONLY)

FileRDD2.persist(StorageLevel.MEMORY_ONLY)

val holidays = FileRDD.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 = FileRDD1.map(x=> (x.split(",")(0),x.split(",")(1).toInt))

val user = FileRDD2.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))

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

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

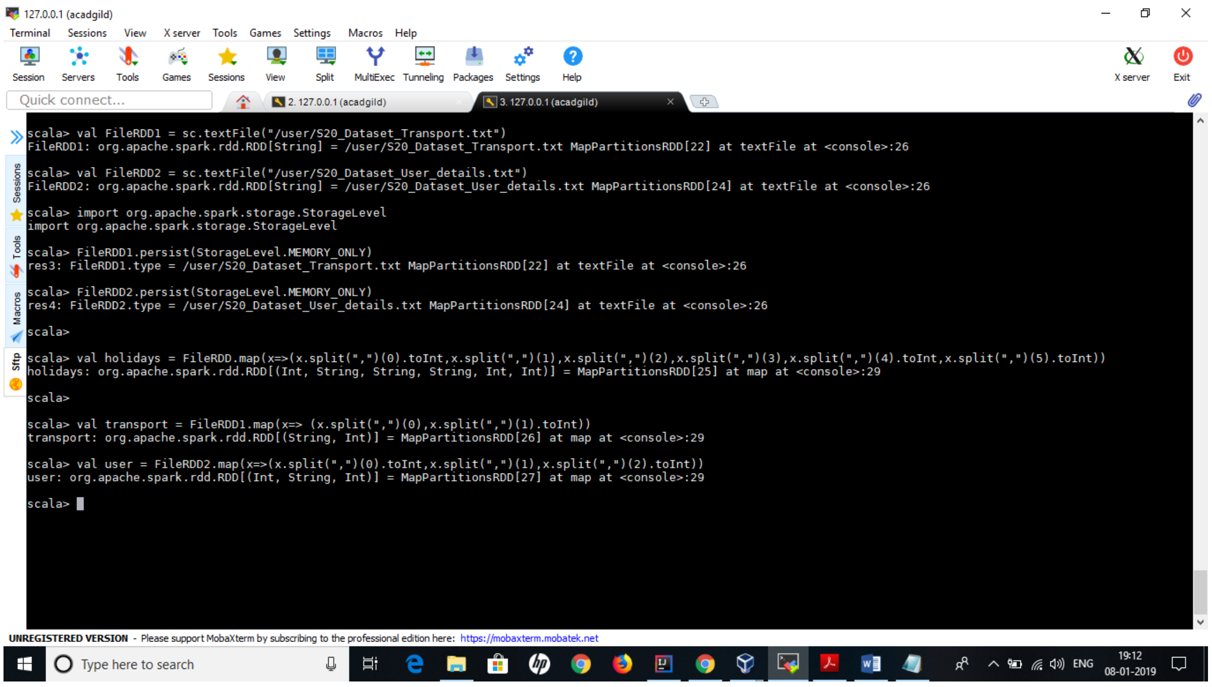
val join1 = holidaysmap.join(transportmap)

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

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

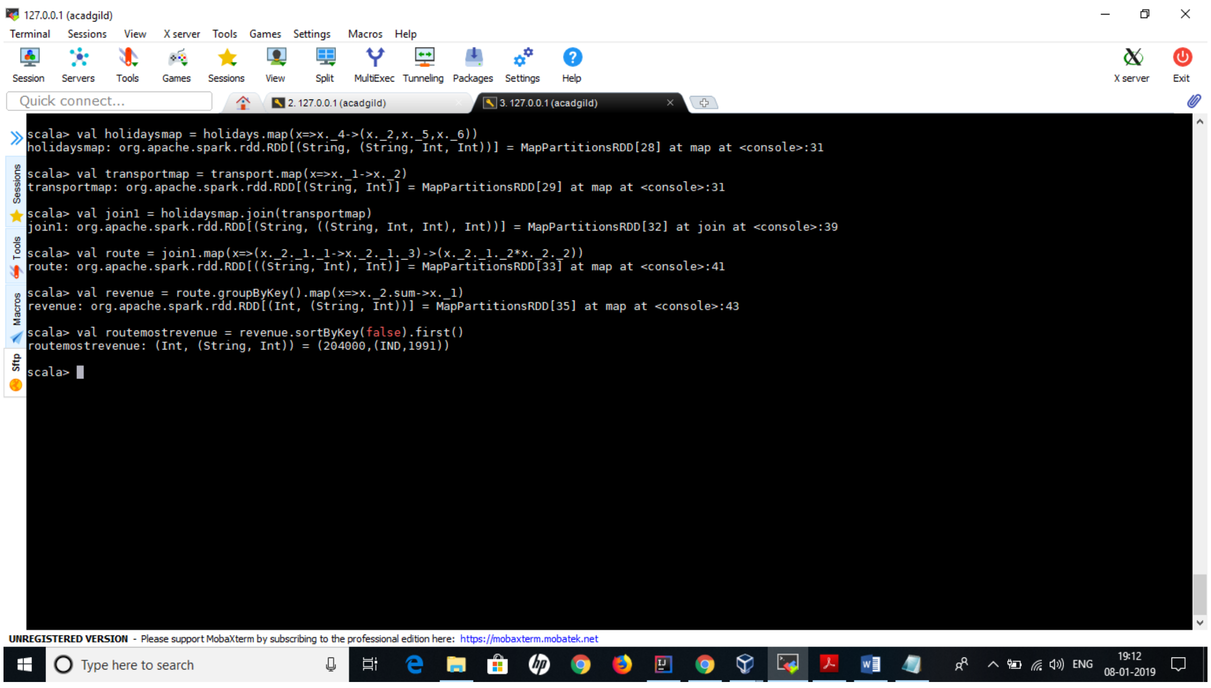
val routemostrevenue = revenue.sortByKey(false).first()

Output:


127.0.0.1 (acadgild)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. 127.0.0.1 (acadgild)
3. 127.0.0.1 (acadgild)
Sessions Tools Macros Sftp

```
scala> val FileRDD1 = sc.textFile("/user/S20_Dataset_Transport.txt")
FileRDD1: org.apache.spark.rdd.RDD[String] = /user/S20_Dataset_Transport.txt MapPartitionsRDD[22] at textFile at <console>:26

scala> val FileRDD2 = sc.textFile("/user/S20_Dataset_User_details.txt")
FileRDD2: org.apache.spark.rdd.RDD[String] = /user/S20_Dataset_User_details.txt MapPartitionsRDD[24] at textFile at <console>:26

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

scala> FileRDD1.persist(StorageLevel.MEMORY_ONLY)
res3: FileRDD1.type = /user/S20_Dataset_Transport.txt MapPartitionsRDD[22] at textFile at <console>:26

scala> FileRDD2.persist(StorageLevel.MEMORY_ONLY)
res4: FileRDD2.type = /user/S20_Dataset_User_details.txt MapPartitionsRDD[24] at textFile at <console>:26

scala>

scala> val holidays = FileRDD.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5).toInt))
holidays: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[25] at map at <console>:29

scala>

scala> val transport = FileRDD1.map(x=> (x.split(",")(0),x.split(",")(1).toInt))
transport: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[26] at map at <console>:29

scala> val user = FileRDD2.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))
user: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[27] at map at <console>:29

scala>
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



```
scala> val holidaysmap = holidays.map(x=>x._4->(x._2,x._5,x._6))
holidaysmap: org.apache.spark.rdd.RDD[(String, (String, Int, Int))] = MapPartitionsRDD[28] at map at <console>:31

scala> val transportmap = transport.map(x=>x._1->x._2)
transportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[29] at map at <console>:31

scala> val join1 = holidaysmap.join(transportmap)
join1: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[32] at join at <console>:39

scala> val route = join1.map(x=>(x._2._1._1->x._2._1._3)->(x._2._1._2*x._2._2))
route: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[33] at map at <console>:41

scala> val revenue = route.groupByKey().map(x=>x._2.sum->x._1)
revenue: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[35] at map at <console>:43

scala> val routemostrevenue = revenue.sortByKey(false).first()
routemostrevenue: (Int, (String, Int)) = (204000,(IND,1991))

scala> 
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

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

Solution:

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

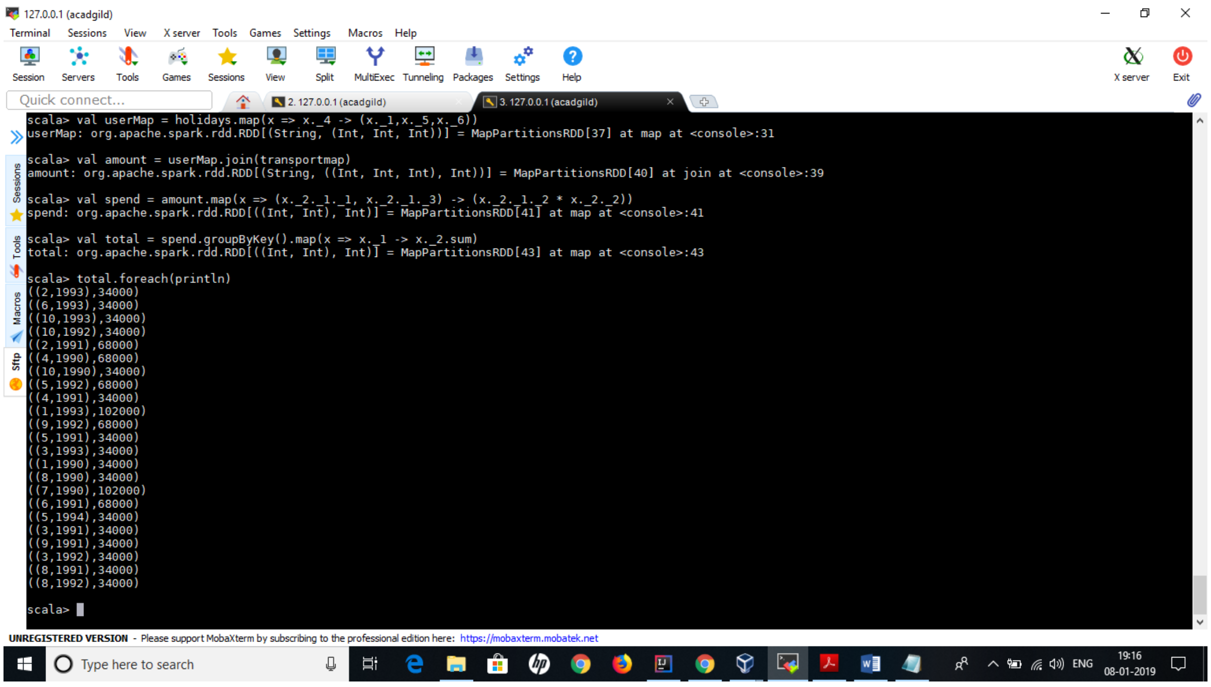
val amount = userMap.join(transportmap)

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)

total.foreach(println)

Output:



```
scala> val userMap = holidays.map(x => x._4 -> (x._1,x._5,x._6))
userMap: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[37] at map at <console>:31

scala> val amount = userMap.join(transportmap)
amount: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[40] at join at <console>:39

scala> val spend = amount.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))
spend: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[41] at map at <console>:41

scala> val total = spend.groupByKey().map(x => x._1 -> x._2.sum)
total: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[43] at map at <console>:43

scala> total.foreach(println)
((2,1993),34000)
((6,1993),34000)
((10,1993),34000)
((10,1992),34000)
((2,1991),68000)
((4,1990),68000)
((10,1990),34000)
((5,1992),68000)
((4,1991),34000)
((1,1993),102000)
((9,1992),68000)
((5,1991),34000)
((3,1993),34000)
((1,1990),34000)
((8,1990),34000)
((7,1990),102000)
((6,1991),68000)
((5,1994),34000)
((3,1991),34000)
((9,1991),34000)
((3,1992),34000)
((8,1991),34000)
((8,1992),34000)

scala> 
```

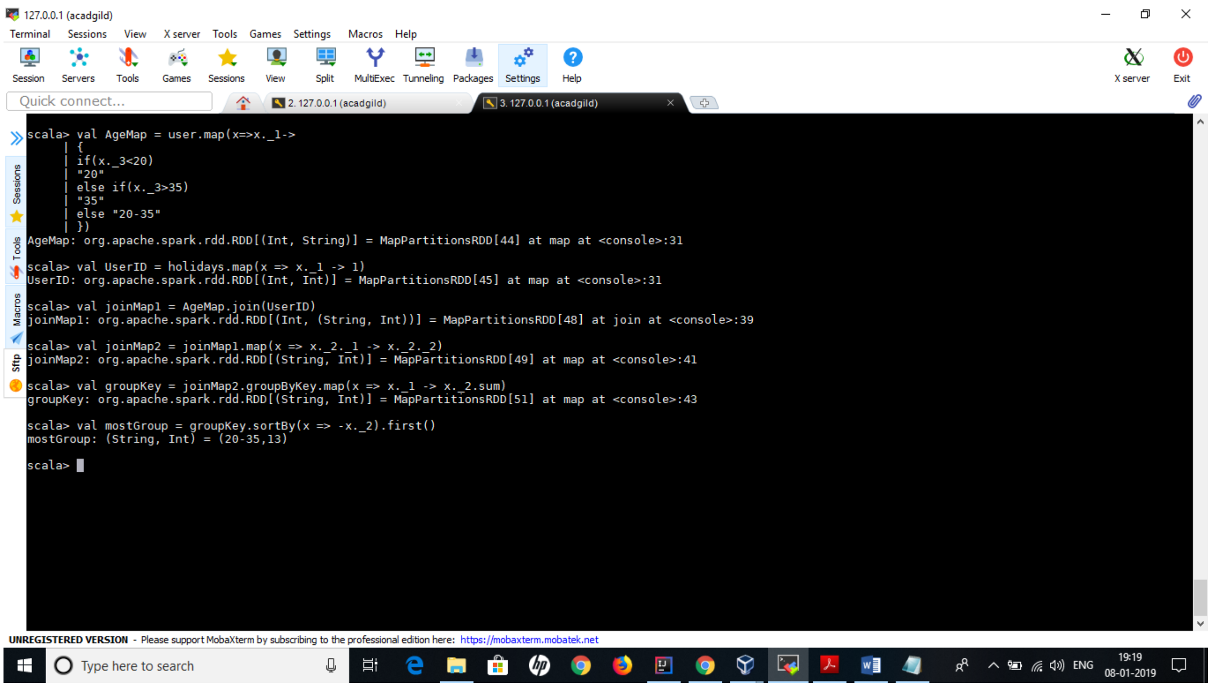
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

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

Solution:

```
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:


127.0.0.1 (acadgild)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. 127.0.0.1 (acadgild)
3. 127.0.0.1 (acadgild)
Sessions Tools Macros Sftp

```
scala> val AgeMap = user.map(x=>x._1->
     | {
     | if(x._3<20)
     | "20"
     | else if(x._3>35)
     | "35"
     | else "20-35"
     | })
AgeMap: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[44] at map at <console>:31

scala> val UserID = holidays.map(x => x._1 -> 1)
UserID: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[45] at map at <console>:31

scala> val joinMap1 = AgeMap.join(UserID)
joinMap1: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[48] at join at <console>:39

scala> val joinMap2 = joinMap1.map(x => x._2._1 -> x._2._2)
joinMap2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[49] at map at <console>:41

scala> val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)
groupKey: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[51] at map at <console>:43

scala> val mostGroup = groupKey.sortBy(x => -x._2).first()
mostGroup: (String, Int) = (20-35,13)

scala> 
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net