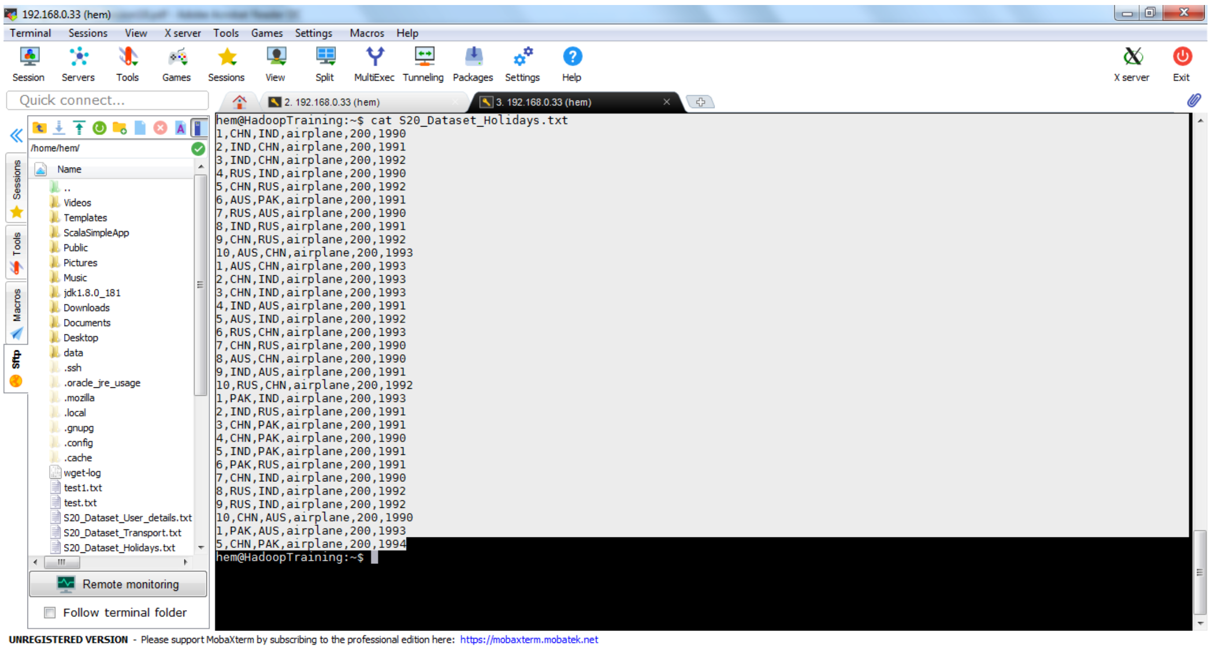
Dataset used,

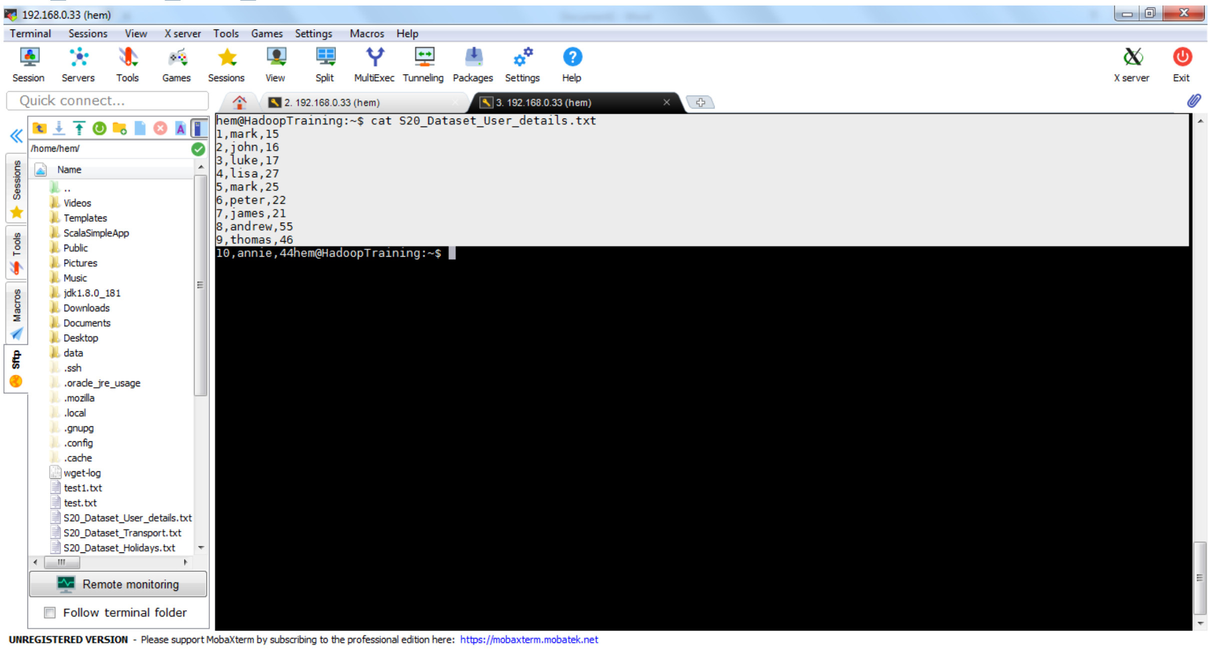
S20_Dataset_Holidays.txt,



```
hem@HadoopTraining:~$ cat S20_Dataset_Holidays.txt
1,CHN,IND,airplane,200,1990
2,IND,CHN,airplane,200,1991
3,IND,CHN,airplane,200,1992
4,RUS,IND,airplane,200,1990
5,CHN,RUS,airplane,200,1992
6,AUS,PAK,airplane,200,1991
7,RUS,AUS,airplane,200,1990
8,IND,RUS,airplane,200,1991
9,CHN,RUS,airplane,200,1992
10,AUS,CHN,airplane,200,1993
1,AUS,CHN,airplane,200,1993
2,CHN,IND,airplane,200,1993
3,CHN,IND,airplane,200,1993
4,IND,AUS,airplane,200,1991
5,AUS,IND,airplane,200,1992
6,RUS,CHN,airplane,200,1993
7,CHN,RUS,airplane,200,1990
8,AUS,CHN,airplane,200,1990
9,IND,AUS,airplane,200,1991
10,RUS,CHN,airplane,200,1992
1,PAK,IND,airplane,200,1993
2,IND,RUS,airplane,200,1991
3,CHN,PAK,airplane,200,1991
4,CHN,PAK,airplane,200,1990
5,IND,PAK,airplane,200,1991
6,PAK,RUS,airplane,200,1991
7,CHN,IND,airplane,200,1990
8,RUS,IND,airplane,200,1992
9,RUS,IND,airplane,200,1992
10,CHN,AUS,airplane,200,1990
1,PAK,AUS,airplane,200,1993
5,CHN,PAK,airplane,200,1994
hem@HadoopTraining:~$
```

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

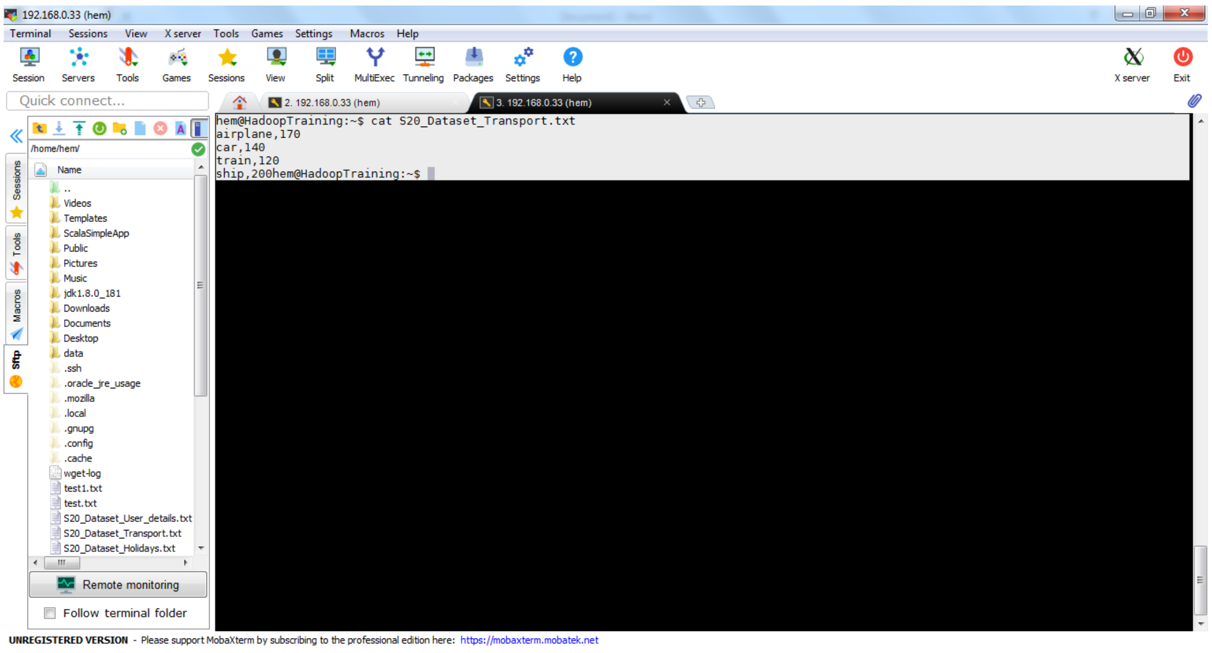
S20_Dataset_User_details.txt,



```
hem@HadoopTraining:~$ cat S20_Dataset_User_details.txt
1,mark,15
2,john,16
3,luke,17
4,lisa,27
5,mark,25
6,peter,22
7,james,21
8,andrew,55
9,thomas,46
10,annie,44hem@HadoopTraining:~$
```

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

S20_Dataset_Transport.txt



```
hem@HadoopTraining:~$ cat S20_Dataset_Transport.txt
airplane,170
car,140
train,120
ship,200hem@HadoopTraining:~$
```

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

Task 1

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

Spark Solution-:

val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")

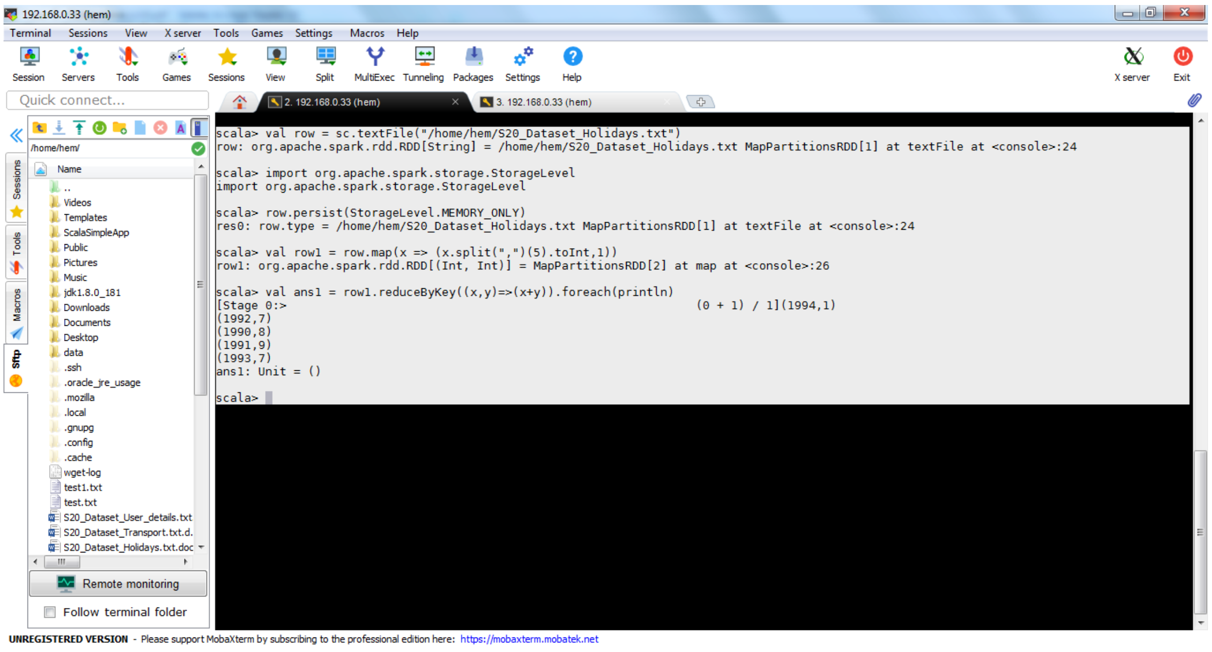
import org.apache.spark.storage.StorageLevel

row.persist(StorageLevel.MEMORY_ONLY)

val row1 = row.map(x => (x.split(",")(5).toInt,1))

val ans1 = row1.reduceByKey((x,y)=>(x+y)).foreach(println)

Output-:



```
scala> val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")
row: org.apache.spark.rdd.RDD[String] = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

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

scala> row.persist(StorageLevel.MEMORY_ONLY)
res0: row.type = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val row1 = row.map(x => (x.split(",")(5).toInt,1))
row1: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[2] at map at <console>:26

scala> val ans1 = row1.reduceByKey((x,y)=>(x+y)).foreach(println)
[Stage 0:>                                                          (0 + 1) / 1](1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)
ans1: Unit = ()

scala>
```

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

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

Spark Solution-:

val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")

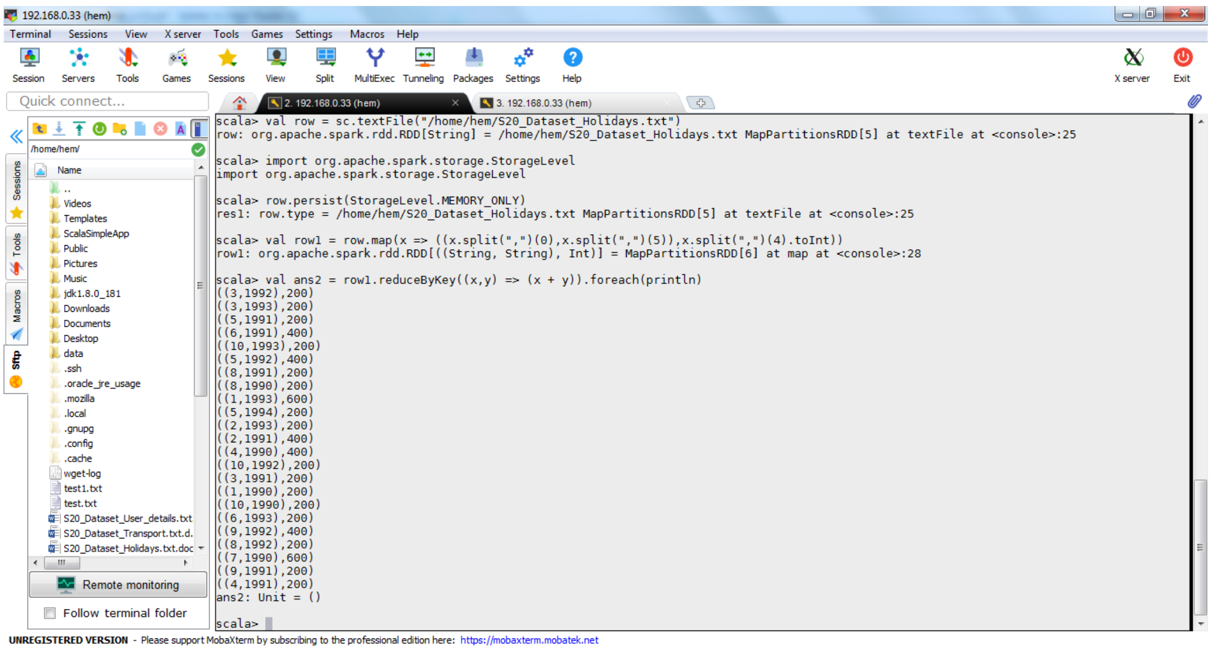
import org.apache.spark.storage.StorageLevel

row.persist(StorageLevel.MEMORY_ONLY)

val row1 = row.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))

val ans2 = row1.reduceByKey((x,y) => (x + y)).foreach(println)

Output-:



```
scala> val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")
row: org.apache.spark.rdd.RDD[String] = /home/hem/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> row.persist(StorageLevel.MEMORY_ONLY)
res1: row.type = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[5] at textFile at <console>:25

scala> val row1 = row.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
row1: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[6] at map at <console>:28

scala> val ans2 = row1.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)
ans2: Unit = ()

scala> 
```

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

3) Which user has travelled the largest distance till date

Spark Solution-:

val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")

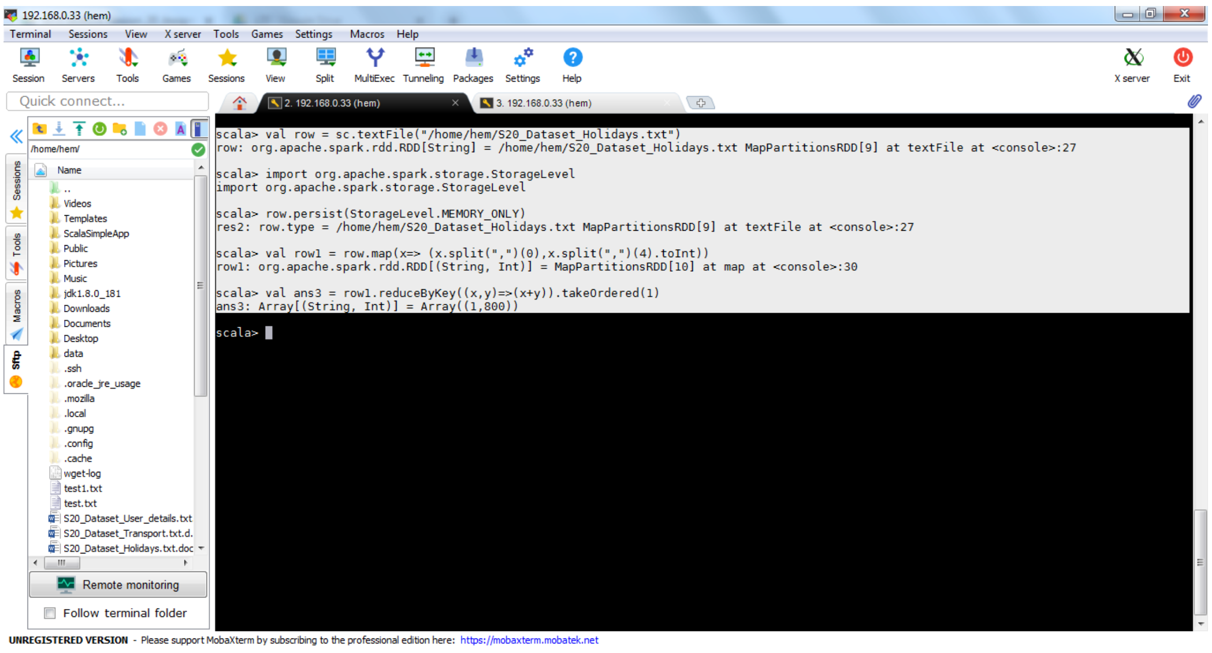
import org.apache.spark.storage.StorageLevel

row.persist(StorageLevel.MEMORY_ONLY)

val row1 = row.map(x=> (x.split(",")(0),x.split(",")(4).toInt))

val ans3 = row1.reduceByKey((x,y)=>(x+y)).takeOrdered(1)

Output-:



```
scala> val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")
row: org.apache.spark.rdd.RDD[String] = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[9] at textFile at <console>:27

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

scala> row.persist(StorageLevel.MEMORY_ONLY)
res2: row.type = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[9] at textFile at <console>:27

scala> val row1 = row.map(x=> (x.split(",")(0),x.split(",")(4).toInt))
row1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[10] at map at <console>:30

scala> val ans3 = row1.reduceByKey((x,y)=>(x+y)).takeOrdered(1)
ans3: Array[(String, Int)] = Array((1,800))

scala> 
```

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

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

Spark Solution-:

val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")

import org.apache.spark.storage.StorageLevel

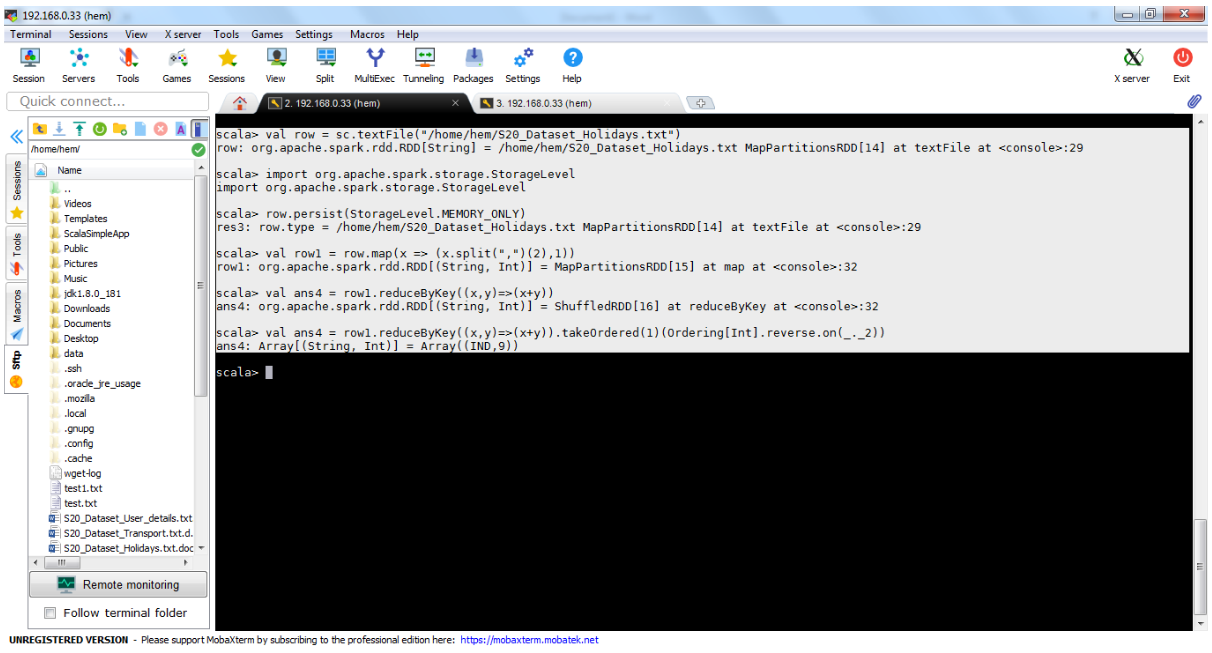
row.persist(StorageLevel.MEMORY_ONLY)

val row1 = row.map(x => (x.split(",")(2),1))

val ans4 = row1.reduceByKey((x,y)=>(x+y))

val ans4 = row1.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(_._2))

Output-:



```
scala> val row = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")
row: org.apache.spark.rdd.RDD[String] = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[14] at textFile at <console>:29

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

scala> row.persist(StorageLevel.MEMORY_ONLY)
res3: row.type = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[14] at textFile at <console>:29

scala> val row1 = row.map(x => (x.split(",")(2),1))
row1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[15] at map at <console>:32

scala> val ans4 = row1.reduceByKey((x,y)=>(x+y))
ans4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[16] at reduceByKey at <console>:32

scala> val ans4 = row1.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(_._2))
ans4: Array[(String, Int)] = Array((IND,9))

scala>
```

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

5)Which route is generating the most revenue per year

Spark Solution-:

val row1 = sc.textFile("/home/hem/hadoop/S20_Dataset_Holidays.txt")

val row2 = sc.textFile("/home/hem/hadoop/S20_Dataset_Transport.txt")

val row3 = sc.textFile("/home/hem/hadoop/S20

import org.apache.spark.storage.StorageLevel

row1.persist(StorageLevel.MEMORY_ONLY)_Dataset_User_details.txt")

row2.persist(StorageLevel.MEMORY_ONLY)

row3.persist(StorageLevel.MEMORY_ONLY)

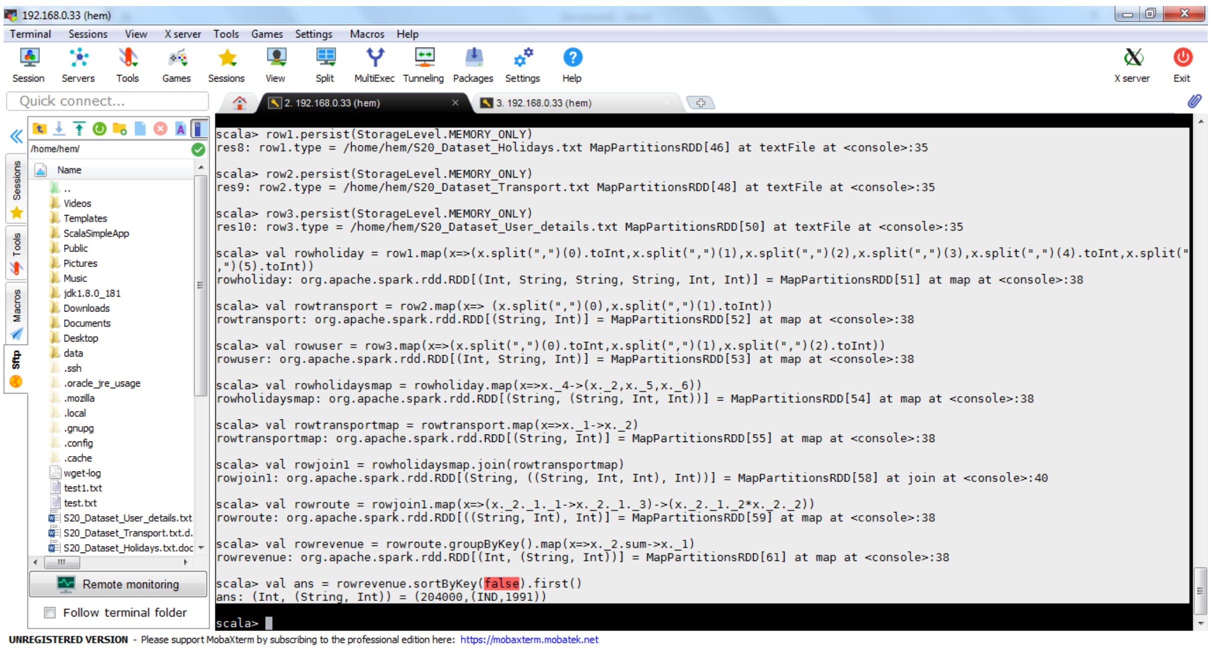
Output-:

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

Spark Solution-:

```
val row1 = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")

val row2 = sc.textFile("/home/hem/S20_Dataset_Transport.txt")

val row3 = sc.textFile("/home/hem/S20_Dataset_User_details.txt")

import org.apache.spark.storage.StorageLevel

row1.persist(StorageLevel.MEMORY_ONLY)

row2.persist(StorageLevel.MEMORY_ONLY)

row3.persist(StorageLevel.MEMORY_ONLY)

val rowholiday =
row1.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split("
,")(5).toInt))

val rowtransport = row2.map(x=> (x.split(",")(0),x.split(",")(1).toInt))

val rowuser = row3.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))

val rowholidaysmap = rowholiday.map(x=>x._4->(x._2,x._5,x._6))

val rowtransportmap = rowtransport.map(x=>x._1->x._2)

val rowjoin1 = rowholidaysmap.join(rowtransportmap)

val rowroute = rowjoin1.map(x=>(x._2._1._1->x._2._1._3)->(x._2._1._2*x._2._2))

val rowrevenue = rowroute.groupByKey().map(x=>x._2.sum->x._1)

val ans = rowrevenue.sortByKey(false).first()
```

Output-:



192.168.0.33 (hem)
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. 192.168.0.33 (hem)
3. 192.168.0.33 (hem)
/home/hem/
Name
..
Videos
Templates
ScalaSimpleApp
Public
Pictures
Music
jdk1.8.0_181
Downloads
Documents
Desktop
data
.ssh
.oracle_jre_usage
.mozilla
.local
.gnupg
.config
.cache
wget-log
test1.txt
test.txt
S20_Dataset_User_details.txt
S20_Dataset_Transport.txt.d.
S20_Dataset_Holidays.txt.doc
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

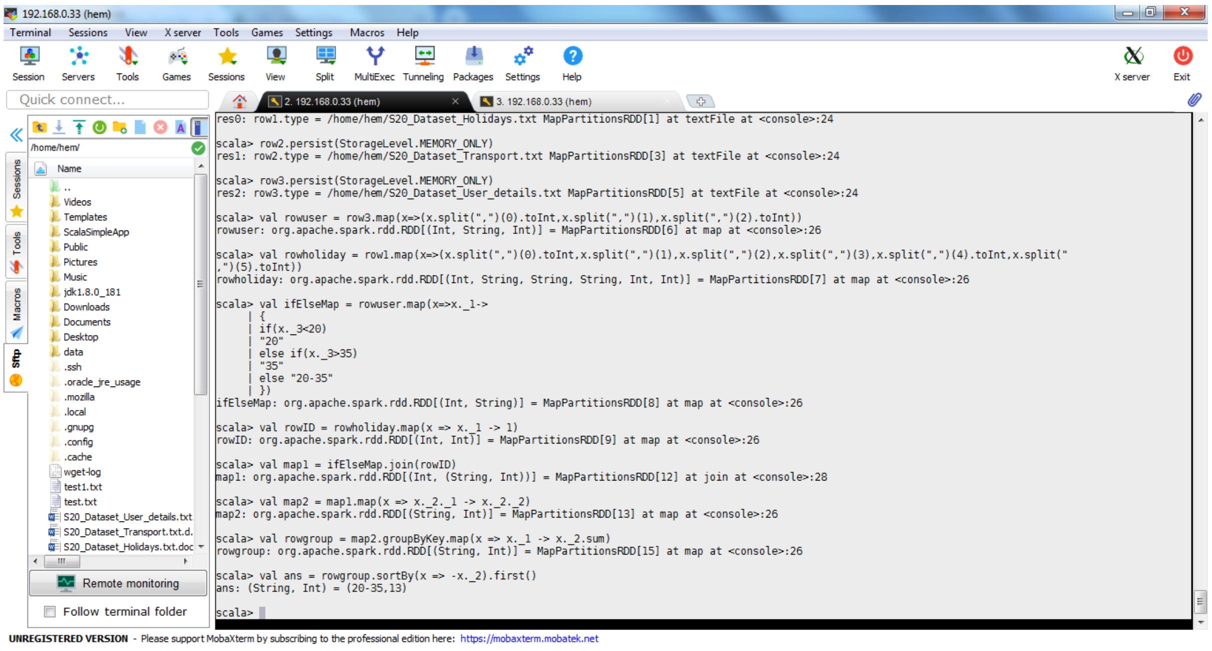
```
scala> row1.persist(StorageLevel.MEMORY_ONLY)
res8: row1.type = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[46] at textFile at <console>:35

scala> row2.persist(StorageLevel.MEMORY_ONLY)
res9: row2.type = /home/hem/S20_Dataset_Transport.txt MapPartitionsRDD[48] at textFile at <console>:35

scala> row3.persist(StorageLevel.MEMORY_ONLY)
res10: row3.type = /home/hem/S20_Dataset_User_details.txt MapPartitionsRDD[50] at textFile at <console>:35

scala> val rowholiday = row1.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split("
,")(5).toInt))
rowholiday: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[51] at map at <console>:38

scala> val rowtransport = row2.map(x=> (x.split(",")(0),x.split(",")(1).toInt))
rowtransport: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[52] at map at <console>:38

scala> val rowuser = row3.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))
rowuser: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[53] at map at <console>:38

scala> val rowholidaysmap = rowholiday.map(x=>x._4->(x._2,x._5,x._6))
rowholidaysmap: org.apache.spark.rdd.RDD[(String, (String, Int, Int))] = MapPartitionsRDD[54] at map at <console>:38

scala> val rowtransportmap = rowtransport.map(x=>x._1->x._2)
rowtransportmap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[55] at map at <console>:38

scala> val rowjoin1 = rowholidaysmap.join(rowtransportmap)
rowjoin1: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[58] at join at <console>:40

scala> val rowroute = rowjoin1.map(x=>(x._2._1._1->x._2._1._3)->(x._2._1._2*x._2._2))
rowroute: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[59] at map at <console>:38

scala> val rowrevenue = rowroute.groupByKey().map(x=>x._2.sum->x._1)
rowrevenue: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[61] at map at <console>:38

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

scala> 
```

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

Spark Solution-:

```
val row1 = sc.textFile("/home/hem/S20_Dataset_Holidays.txt")

val row2 = sc.textFile("/home/hem/S20_Dataset_Transport.txt")

val row3 = sc.textFile("/home/hem/S20_Dataset_User_details.txt")

import org.apache.spark.storage.StorageLevel

row1.persist(StorageLevel.MEMORY_ONLY)

row2.persist(StorageLevel.MEMORY_ONLY)

row3.persist(StorageLevel.MEMORY_ONLY)

val rowuser = row3.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))

val rowholiday =
row1.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split("
,")(5).toInt))

val ifElseMap = rowuser.map(x=>x._1->| {

| if(x._3<20)

| "20"

| else if(x._3>35)

| "35"
| else "20-35"

| })

val rowID = rowholiday.map(x => x._1 -> 1)

val map1 = ifElseMap.join(rowID)

val map2 = map1.map(x => x._2._1 -> x._2._2)

val rowgroup = map2.groupByKey.map(x => x._1 -> x._2.sum)

val ans = rowgroup.sortBy(x => -x._2).first()
```

Output-:



```
res0: row1.type = /home/hem/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> row2.persist(StorageLevel.MEMORY_ONLY)
res1: row2.type = /home/hem/S20_Dataset_Transport.txt MapPartitionsRDD[3] at textFile at <console>:24

scala> row3.persist(StorageLevel.MEMORY_ONLY)
res2: row3.type = /home/hem/S20_Dataset_User_details.txt MapPartitionsRDD[5] at textFile at <console>:24

scala> val rowuser = row3.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))
rowuser: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[6] at map at <console>:26

scala> val rowholiday = row1.map(x=>(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split("
,")(5).toInt))
rowholiday: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[7] at map at <console>:26

scala> val ifElseMap = rowuser.map(x=>x._1->
     | {
     | if(x._3<20)
     | "20"
     | else if(x._3>35)
     | "35"
     | else "20-35"
     | })
ifElseMap: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[8] at map at <console>:26

scala> val rowID = rowholiday.map(x => x._1 -> 1)
rowID: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[9] at map at <console>:26

scala> val map1 = ifElseMap.join(rowID)
map1: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[12] at join at <console>:28

scala> val map2 = map1.map(x => x._2._1 -> x._2._2)
map2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[13] at map at <console>:26

scala> val rowgroup = map2.groupByKey.map(x => x._1 -> x._2.sum)
rowgroup: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[15] at map at <console>:26

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

scala>
```

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