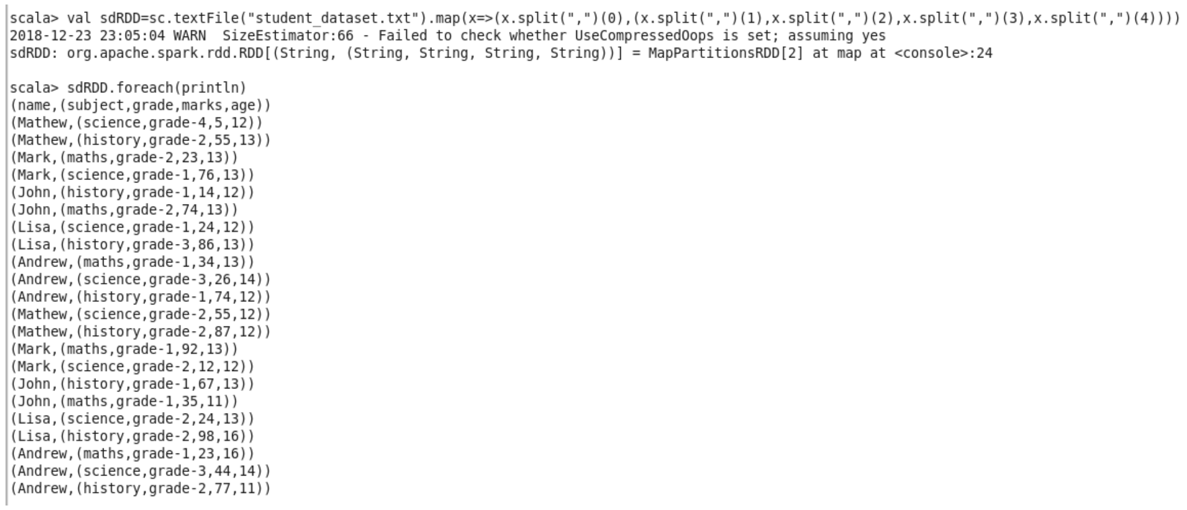
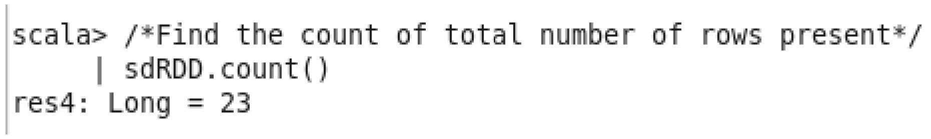
Task1:

1.Read the text file, and create a tupled rdd.



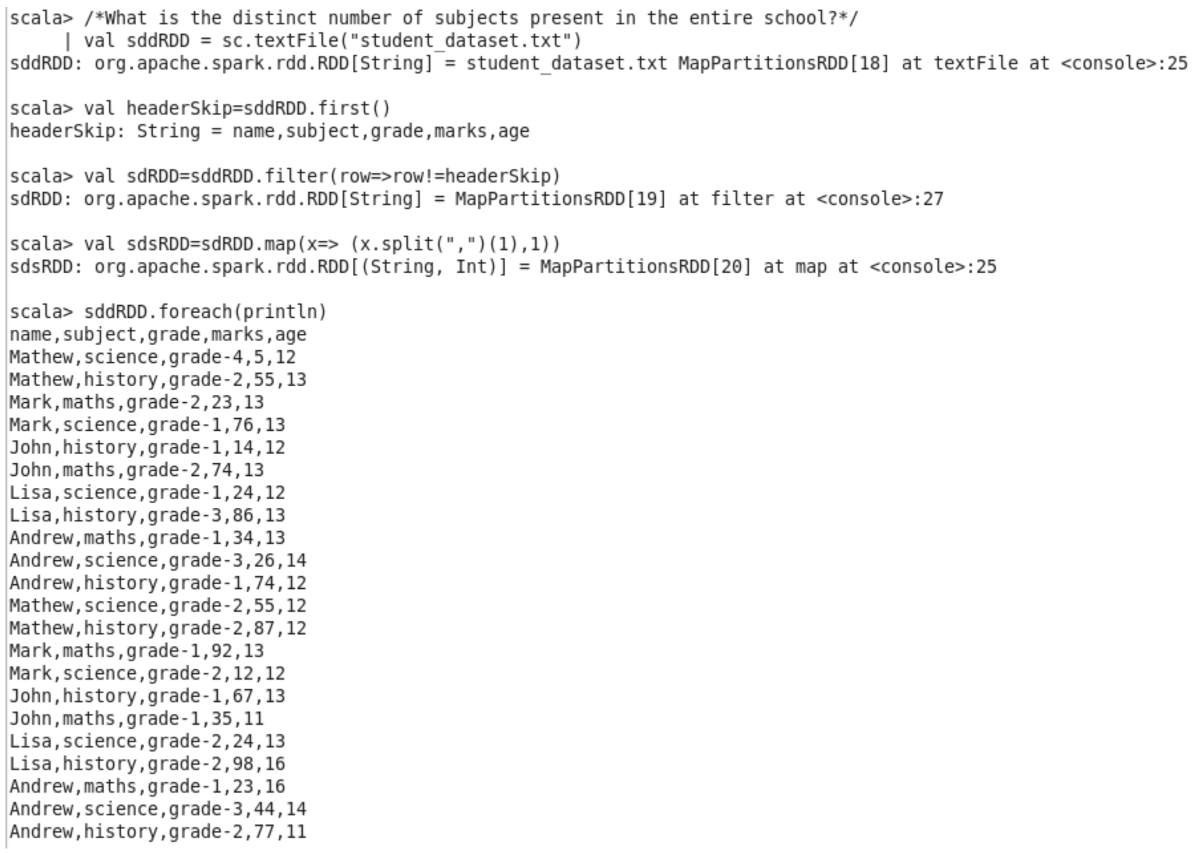
```
scala> val sdRDD=sc.textFile("student_dataset.txt").map(x=>(x.split(",")(0),(x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4))))
2018-12-23 23:05:04 WARN  SizeEstimator:66 - Failed to check whether UseCompressedOops is set; assuming yes
sdRDD: org.apache.spark.rdd.RDD[(String, (String, String, String, String))] = MapPartitionsRDD[2] at map at <console>:24

scala> sdRDD.foreach(println)
(name,(subject,grade,marks,age))
(Mathew,(science,grade-4,5,12))
(Mathew,(history,grade-2,55,13))
(Mark,(maths,grade-2,23,13))
(Mark,(science,grade-1,76,13))
(John,(history,grade-1,14,12))
(John,(maths,grade-2,74,13))
(Lisa,(science,grade-1,24,12))
(Lisa,(history,grade-3,86,13))
(Andrew,(maths,grade-1,34,13))
(Andrew,(science,grade-3,26,14))
(Andrew,(history,grade-1,74,12))
(Mathew,(science,grade-2,55,12))
(Mathew,(history,grade-2,87,12))
(Mark,(maths,grade-1,92,13))
(Mark,(science,grade-2,12,12))
(John,(history,grade-1,67,13))
(John,(maths,grade-1,35,11))
(Lisa,(science,grade-2,24,13))
(Lisa,(history,grade-2,98,16))
(Andrew,(maths,grade-1,23,16))
(Andrew,(science,grade-3,44,14))
(Andrew,(history,grade-2,77,11))
```

1. Find the count of total number of rows present.

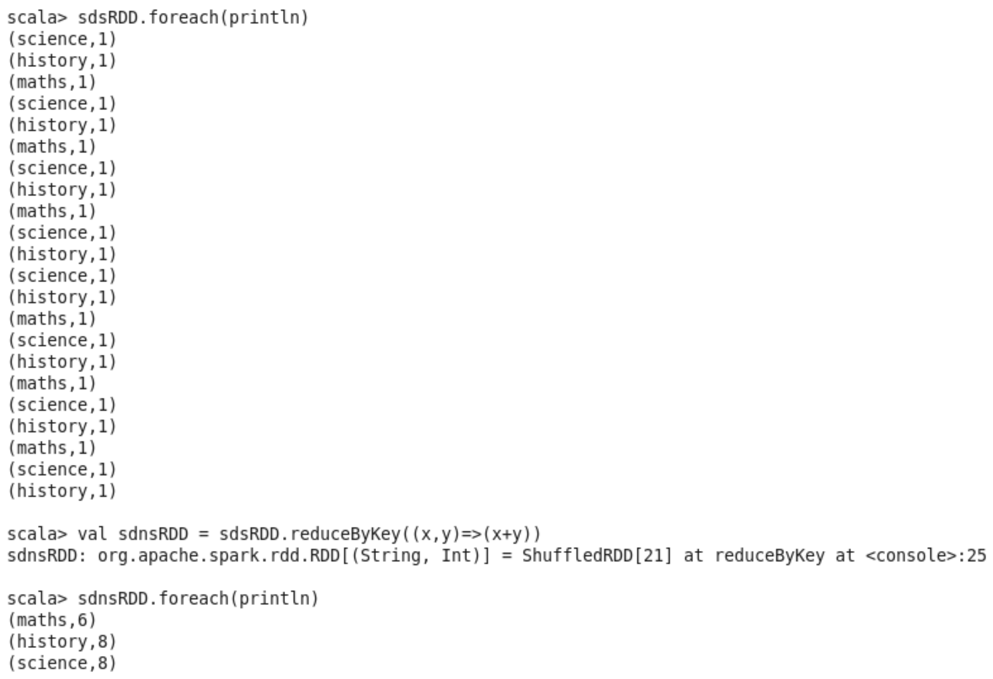


```
scala> /*Find the count of total number of rows present*/
     | sdRDD.count()
res4: Long = 23
```

2. What is the distinct number of subjects present in the entire school

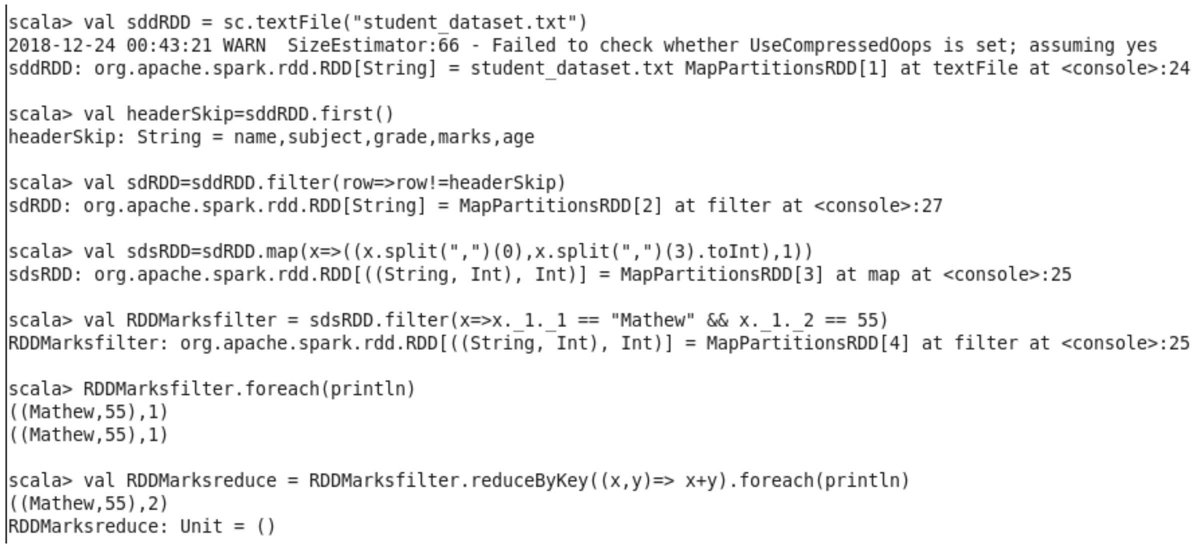


```
scala> /*What is the distinct number of subjects present in the entire school?*/
     | val sddRDD = sc.textFile("student_dataset.txt")
sddRDD: org.apache.spark.rdd.RDD[String] = student_dataset.txt MapPartitionsRDD[18] at textFile at <console>:25

scala> val headerSkip=sddRDD.first()
headerSkip: String = name,subject,grade,marks,age

scala> val sdRDD=sddRDD.filter(row=>row!=headerSkip)
sdRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[19] at filter at <console>:27

scala> val sdsRDD=sdRDD.map(x=> (x.split(",")(1),1))
sdsRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[20] at map at <console>:25

scala> sddRDD.foreach(println)
name,subject,grade,marks,age
Mathew,science,grade-4,5,12
Mathew,history,grade-2,55,13
Mark,maths,grade-2,23,13
Mark,science,grade-1,76,13
John,history,grade-1,14,12
John,maths,grade-2,74,13
Lisa,science,grade-1,24,12
Lisa,history,grade-3,86,13
Andrew,maths,grade-1,34,13
Andrew,science,grade-3,26,14
Andrew,history,grade-1,74,12
Mathew,science,grade-2,55,12
Mathew,history,grade-2,87,12
Mark,maths,grade-1,92,13
Mark,science,grade-2,12,12
John,history,grade-1,67,13
John,maths,grade-1,35,11
Lisa,science,grade-2,24,13
Lisa,history,grade-2,98,16
Andrew,maths,grade-1,23,16
Andrew,science,grade-3,44,14
Andrew,history,grade-2,77,11
```



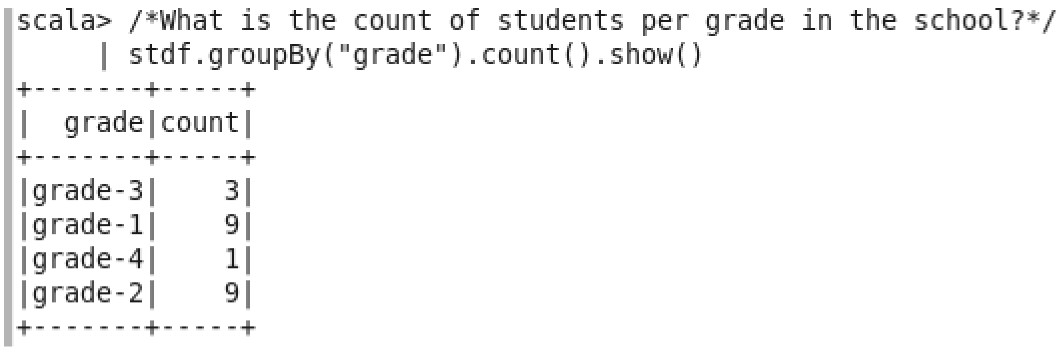
```
scala> sdsRDD.foreach(println)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)
(maths,1)
(science,1)
(history,1)

scala> val sdnsRDD = sdsRDD.reduceByKey((x,y)=>(x+y))
sdnsRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[21] at reduceByKey at <console>:25

scala> sdnsRDD.foreach(println)
(maths,6)
(history,8)
(science,8)
```

3. What is the count of the number of students in the school, whose name is Mathew and marks is 55

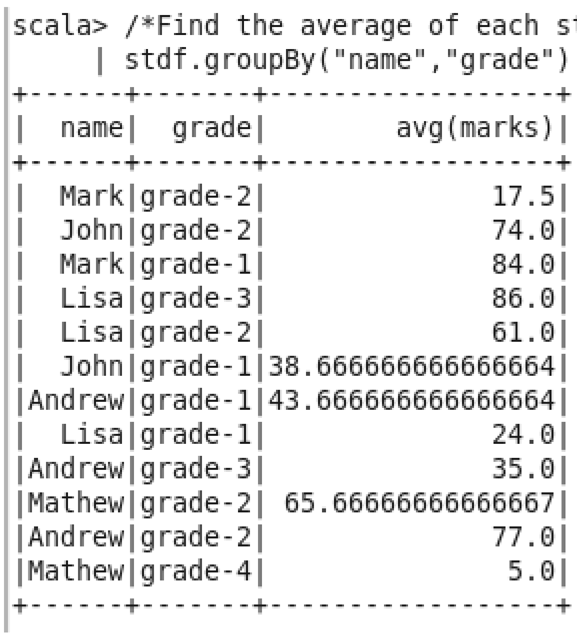

```
scala> val sddRDD = sc.textFile("student_dataset.txt")
2018-12-24 00:43:21 WARN  SizeEstimator:66 - Failed to check whether UseCompressedOops is set; assuming yes
sddRDD: org.apache.spark.rdd.RDD[String] = student_dataset.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val headerSkip=sddRDD.first()
headerSkip: String = name,subject,grade,marks,age

scala> val sdRDD=sddRDD.filter(row=>row!=headerSkip)
sdRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:27

scala> val sdsRDD=sdRDD.map(x=>((x.split(",")(0),x.split(",")(3).toInt),1))
sdsRDD: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[3] at map at <console>:25

scala> val RDDMarksfilter = sdsRDD.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)
RDDMarksfilter: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[4] at filter at <console>:25

scala> RDDMarksfilter.foreach(println)
((Mathew,55),1)
((Mathew,55),1)

scala> val RDDMarksreduce = RDDMarksfilter.reduceByKey((x,y)=> x+y).foreach(println)
((Mathew,55),2)
RDDMarksreduce: Unit = ()
```

Problem Statement 2:

1. What is the count of students per grade in the school?

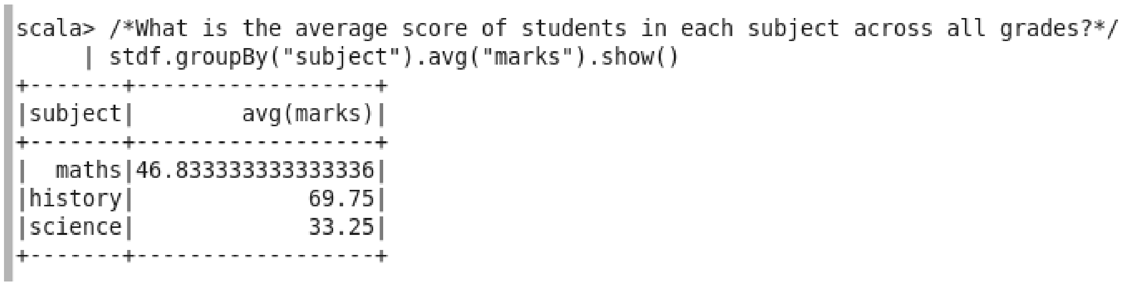

```
scala> /*What is the count of students per grade in the school?*/
     | stdf.groupBy("grade").count().show()
+-------+-----+
|  grade|count|
+-------+-----+
|grade-3|    3|
|grade-1|    9|
|grade-4|    1|
|grade-2|    9|
+-------+-----+
```

2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)



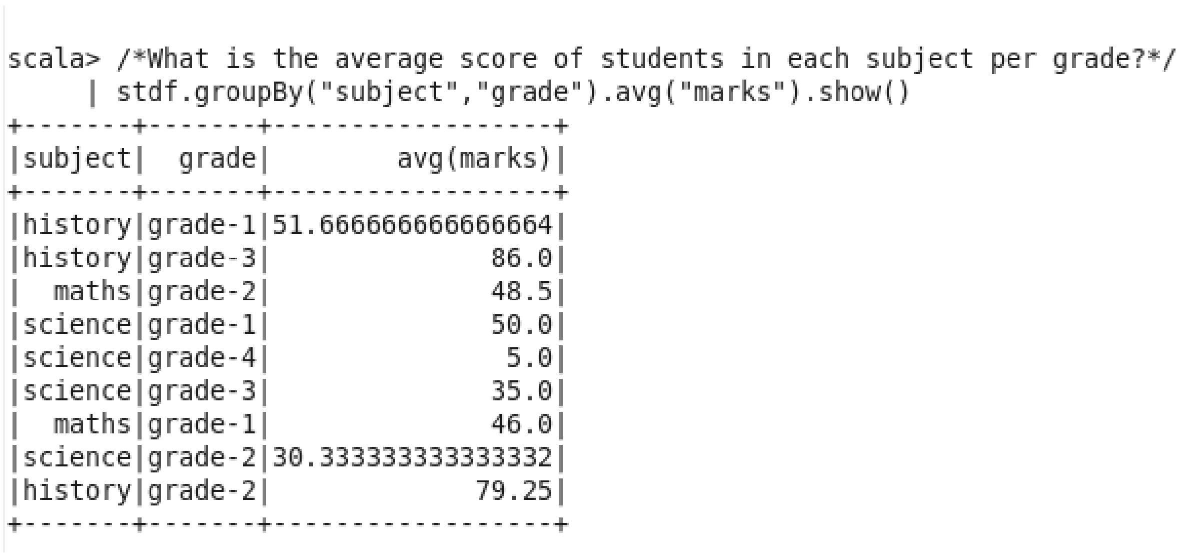
```
scala> /*Find the average of each student*/
     | stdf.groupBy("name","grade").avg("marks").show()
+------+-------+-----------------+
|  name|  grade|       avg(marks)|
+------+-------+-----------------+
|  Mark|grade-2|             17.5|
|  John|grade-2|             74.0|
|  Mark|grade-1|             84.0|
|  Lisa|grade-3|             86.0|
|  Lisa|grade-2|             61.0|
|  John|grade-1|38.666666666666664|
|Andrew|grade-1|43.666666666666664|
|  Lisa|grade-1|             24.0|
|Andrew|grade-3|             35.0|
|Mathew|grade-2| 65.66666666666667|
|Andrew|grade-2|             77.0|
|Mathew|grade-4|              5.0|
+------+-------+-----------------+
```

2. What is the average score of students in each subject across all grades?



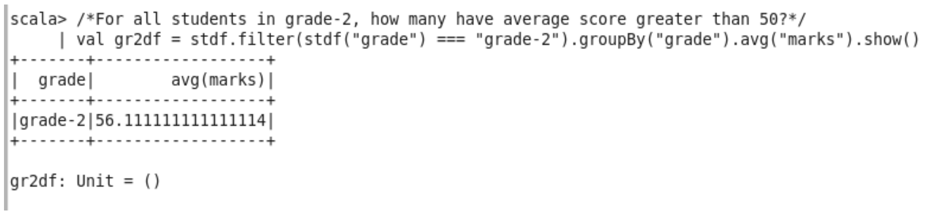
```
scala> /*What is the average score of students in each subject across all grades?*/
     | stdf.groupBy("subject").avg("marks").show()
+-------+-----------------+
|subject|       avg(marks)|
+-------+-----------------+
|  maths|46.833333333333336|
|history|            69.75|
|science|            33.25|
+-------+-----------------+
```

3. What is the average score of students in each subject per grade?



```
scala> /*What is the average score of students in each subject per grade?*/
     | stdf.groupBy("subject","grade").avg("marks").show()
+-------+-------+-----------------+
|subject|  grade|       avg(marks)|
+-------+-------+-----------------+
|history|grade-1|51.666666666666664|
|history|grade-3|             86.0|
|  maths|grade-2|             48.5|
|science|grade-1|             50.0|
|science|grade-4|              5.0|
|science|grade-3|             35.0|
|  maths|grade-1|             46.0|
|science|grade-2|30.333333333333332|
|history|grade-2|            79.25|
+-------+-------+-----------------+
```

4. For all students in grade-2, how many have average score greater than 50?



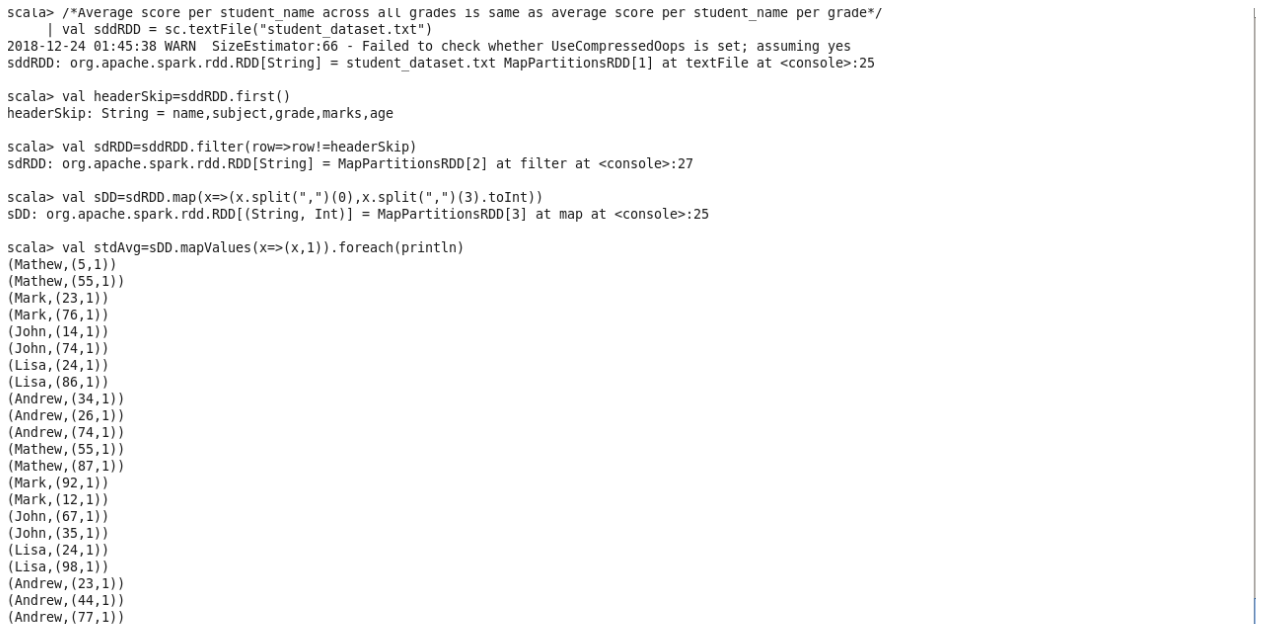
```
scala> /*For all students in grade-2, how many have average score greater than 50?*/
     | val gr2df = stdf.filter(stdf("grade") === "grade-2").groupBy("grade").avg("marks").show()
+-------+-----------------+
|  grade|       avg(marks)|
+-------+-----------------+
|grade-2|56.111111111111114|
+-------+-----------------+

gr2df: Unit = ()
```

Problem Statement 3:

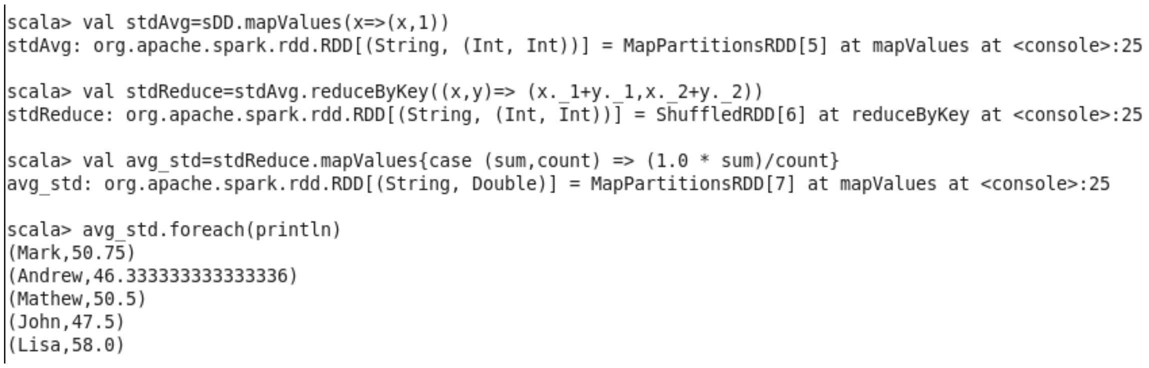
Are there any students in the college that satisfy the below criteria :

1. Average score per student_name across all grades is same as average score per student_name per grade

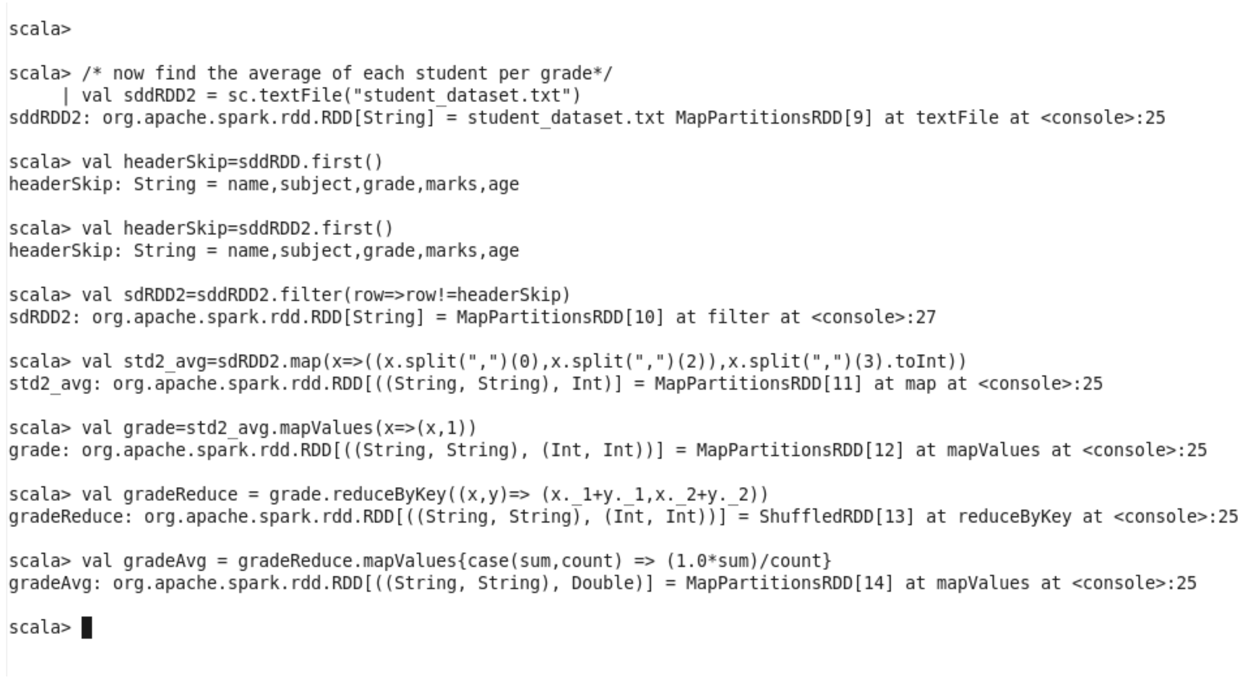
Hint - Use Intersection Property.



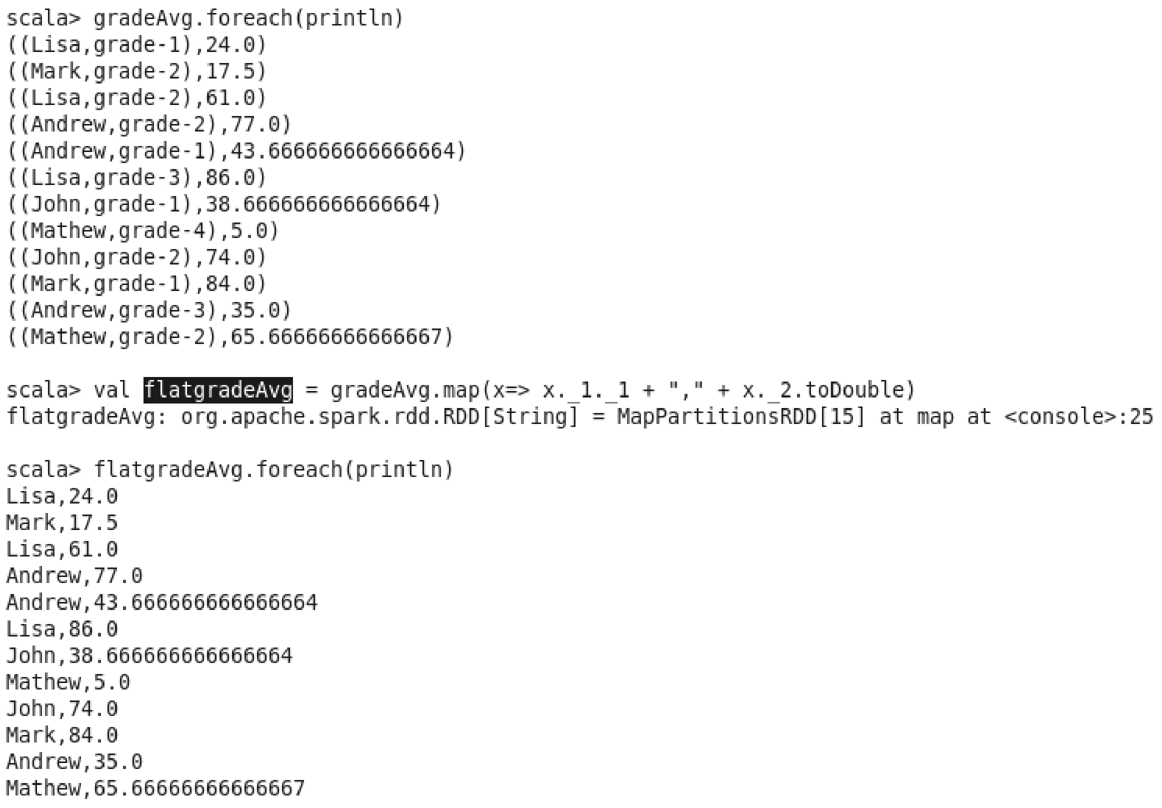
```
scala> /*Average score per student_name across all grades is same as average score per student_name per grade*/
     | val sddRDD = sc.textFile("student_dataset.txt")
2018-12-24 01:45:38 WARN  SizeEstimator:66 - Failed to check whether UseCompressedOops is set; assuming yes
sddRDD: org.apache.spark.rdd.RDD[String] = student_dataset.txt MapPartitionsRDD[1] at textFile at <console>:25

scala> val headerSkip=sddRDD.first()
headerSkip: String = name,subject,grade,marks,age

scala> val sdRDD=sddRDD.filter(row=>row!=headerSkip)
sdRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:27

scala> val sDD=sdRDD.map(x=>(x.split(",")(0),x.split(",")(3).toInt))
sDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:25

scala> val stdAvg=sDD.mapValues(x=>(x,1)).foreach(println)
(Mathew,(5,1))
(Mathew,(55,1))
(Mark,(23,1))
(Mark,(76,1))
(John,(14,1))
(John,(74,1))
(Lisa,(24,1))
(Lisa,(86,1))
(Andrew,(34,1))
(Andrew,(26,1))
(Andrew,(74,1))
(Mathew,(55,1))
(Mathew,(87,1))
(Mark,(92,1))
(Mark,(12,1))
(John,(67,1))
(John,(35,1))
(Lisa,(24,1))
(Lisa,(98,1))
(Andrew,(23,1))
(Andrew,(44,1))
(Andrew,(77,1))
```



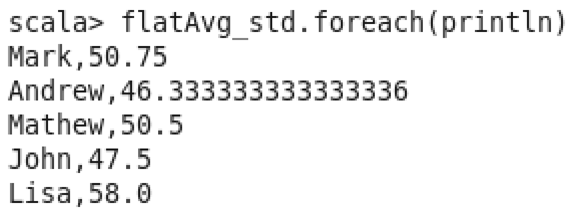
```
scala> val stdAvg=sDD.mapValues(x=>(x,1))
stdAvg: org.apache.spark.rdd.RDD[(String, (Int, Int))] = MapPartitionsRDD[5] at mapValues at <console>:25

scala> val stdReduce=stdAvg.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
stdReduce: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ShuffledRDD[6] at reduceByKey at <console>:25

scala> val avg_std=stdReduce.mapValues{case (sum,count) => (1.0 * sum)/count}
avg_std: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[7] at mapValues at <console>:25

scala> avg_std.foreach(println)
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,50.5)
(John,47.5)
(Lisa,58.0)
```



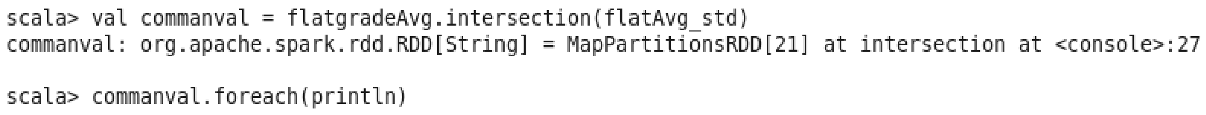
```
scala>

scala> /* now find the average of each student per grade*/
     | val sddRDD2 = sc.textFile("student_dataset.txt")
sddRDD2: org.apache.spark.rdd.RDD[String] = student_dataset.txt MapPartitionsRDD[9] at textFile at <console>:25

scala> val headerSkip=sddRDD.first()
headerSkip: String = name,subject,grade,marks,age

scala> val headerSkip=sddRDD2.first()
headerSkip: String = name,subject,grade,marks,age

scala> val sdRDD2=sddRDD2.filter(row=>row!=headerSkip)
sdRDD2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[10] at filter at <console>:27

scala> val std2_avg=sdRDD2.map(x=>((x.split(",")(0),x.split(",")(2)),x.split(",")(3).toInt))
std2_avg: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[11] at map at <console>:25

scala> val grade=std2_avg.mapValues(x=>(x,1))
grade: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = MapPartitionsRDD[12] at mapValues at <console>:25

scala> val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
gradeReduce: org.apache.spark.rdd.RDD[((String, String), (Int, Int))] = ShuffledRDD[13] at reduceByKey at <console>:25

scala> val gradeAvg = gradeReduce.mapValues{case(sum,count) => (1.0*sum)/count}
gradeAvg: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[14] at mapValues at <console>:25

scala> 
```



```
scala> gradeAvg.foreach(println)
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((Mathew,grade-4),5.0)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)

scala> val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
flatgradeAvg: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at map at <console>:25

scala> flatgradeAvg.foreach(println)
Lisa,24.0
Mark,17.5
Lisa,61.0
Andrew,77.0
Andrew,43.666666666666664
Lisa,86.0
John,38.666666666666664
Mathew,5.0
John,74.0
Mark,84.0
Andrew,35.0
Mathew,65.66666666666667
```



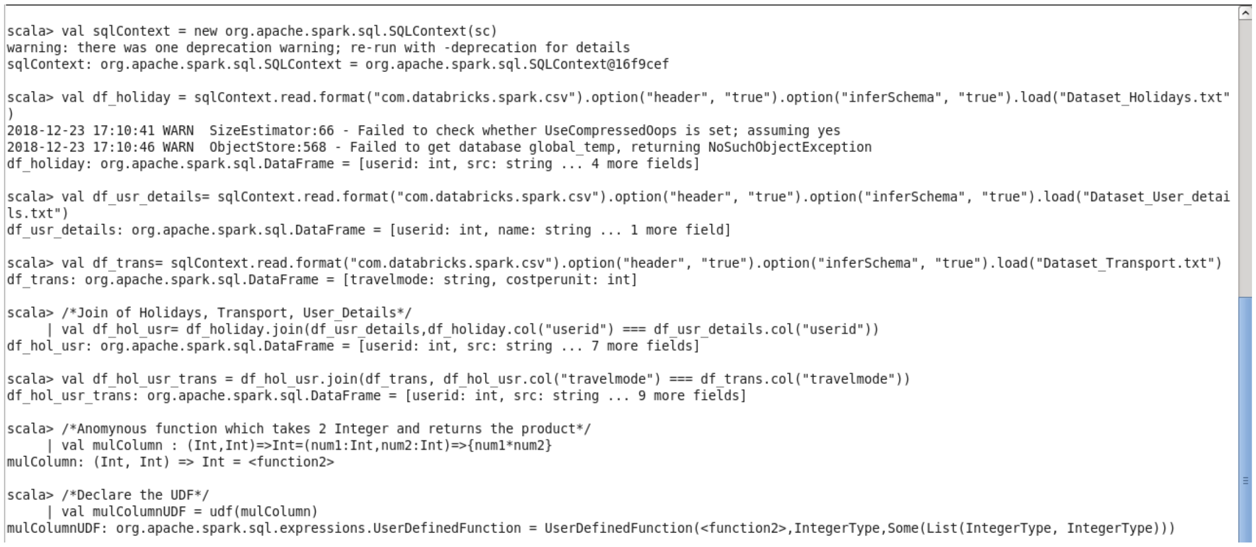
```
scala> flatAvg_std.foreach(println)
Mark,50.75
Andrew,46.333333333333336
Mathew,50.5
John,47.5
Lisa,58.0
```



```
scala> val commanval = flatgradeAvg.intersection(flatAvg_std)
commanval: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[21] at intersection at <console>:27

scala> commanval.foreach(println)
```

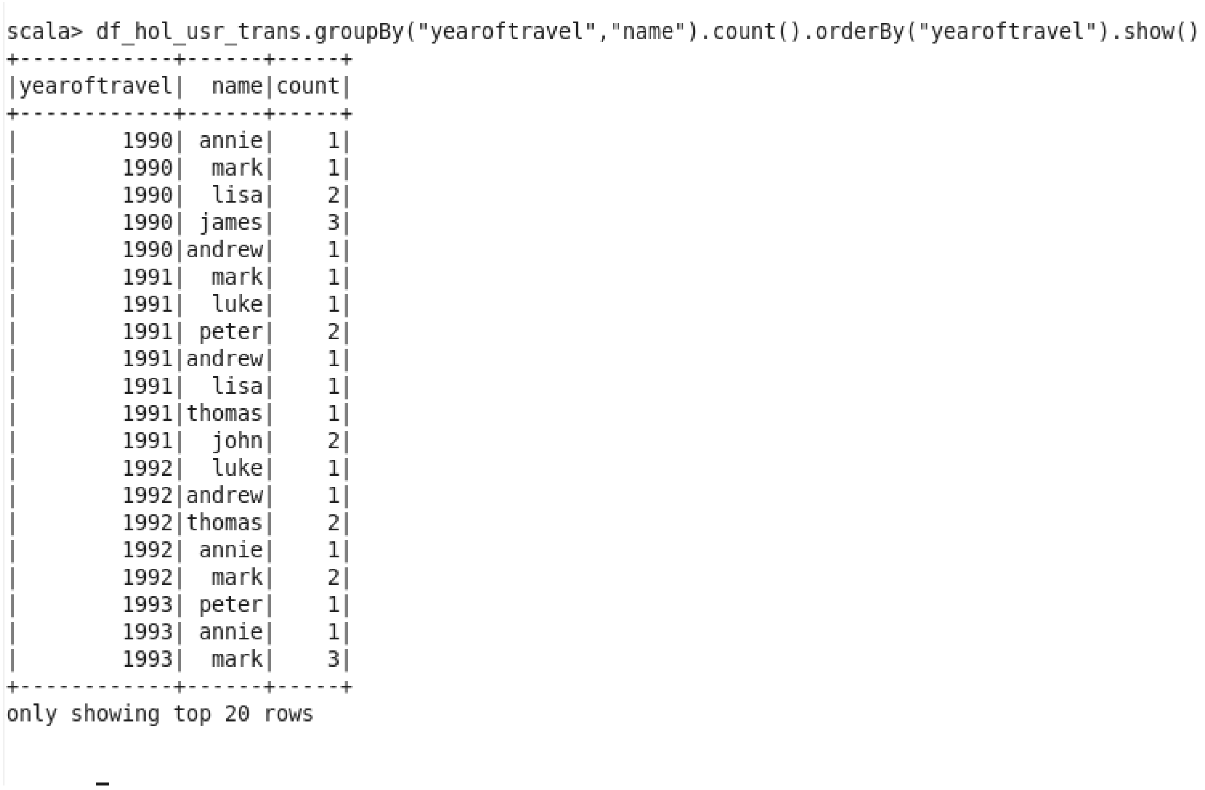
As per the above output no common students are there having average
score per student_name across all grades is same as average score per **student_name** per grade

Task 2



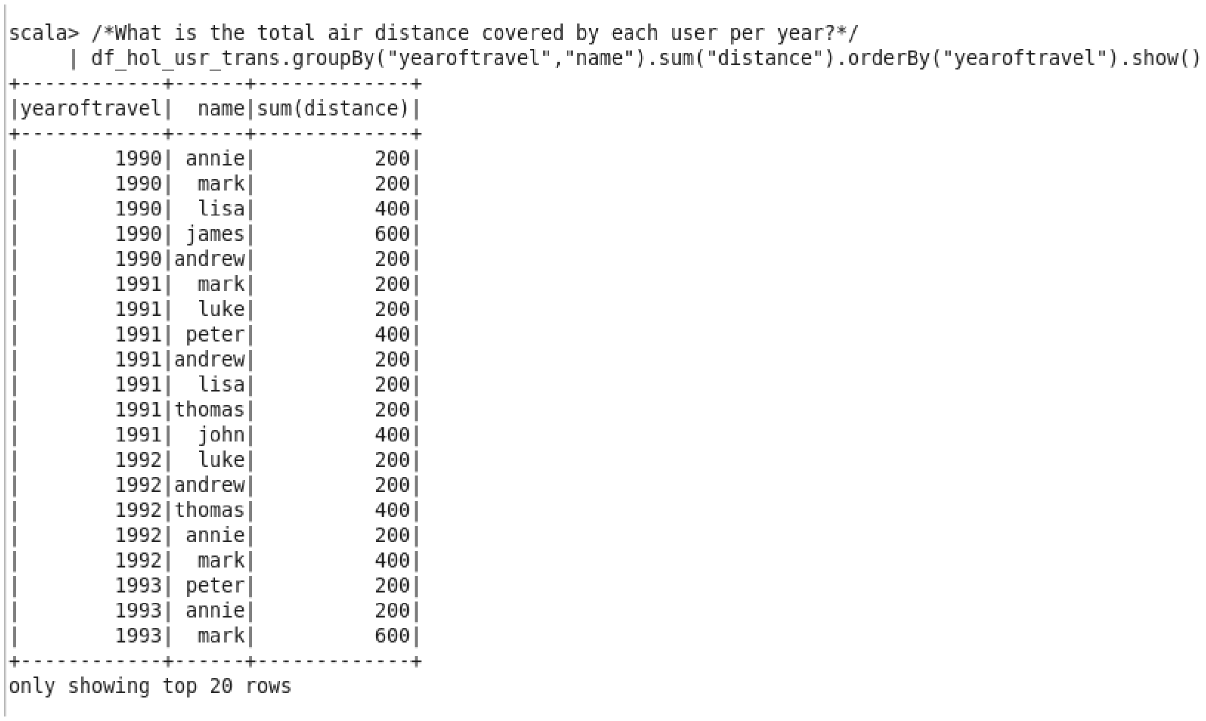
```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@16f9cef

scala> val df_holiday = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("Dataset_Holidays.txt"
)
2018-12-23 17:10:41 WARN  SizeEstimator:66 - Failed to check whether UseCompressedOops is set; assuming yes
2018-12-23 17:10:46 WARN  ObjectStore:568 - Failed to get database global_temp, returning NoSuchObjectException
df_holiday: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 4 more fields]

scala> val df_usr_details= sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("Dataset_User_detai
ls.txt")
df_usr_details: org.apache.spark.sql.DataFrame = [userid: int, name: string ... 1 more field]

scala> val df_trans= sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("Dataset_Transport.txt")
df_trans: org.apache.spark.sql.DataFrame = [travelmode: string, costperunit: int]

scala> /*Join of Holidays, Transport, User_Details*/
     | val df_hol_usr= df_holiday.join(df_usr_details,df_holiday.col("userid") === df_usr_details.col("userid"))
df_hol_usr: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 7 more fields]

scala> val df_hol_usr_trans = df_hol_usr.join(df_trans, df_hol_usr.col("travelmode") === df_trans.col("travelmode"))
df_hol_usr_trans: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 9 more fields]

scala> /*Anomynous function which takes 2 Integer and returns the product*/
     | val mulColumn : (Int,Int)=>Int=(num1:Int,num2:Int)=>{num1*num2}
mulColumn: (Int, Int) => Int = <function2>

scala> /*Declare the UDF*/
     | val mulColumnUDF = udf(mulColumn)
mulColumnUDF: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,IntegerType,Some(List(IntegerType, IntegerType)))
```

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



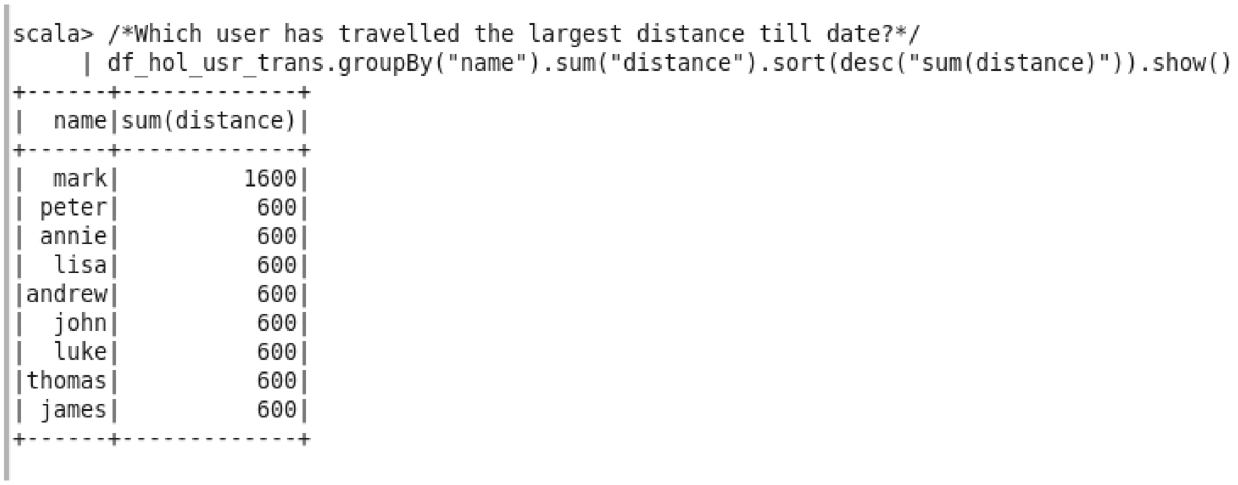
```
scala> df_hol_usr_trans.groupBy("yearoftravel","name").count().orderBy("yearoftravel").show()
+------------+------+-----+
|yearoftravel|  name|count|
+------------+------+-----+
|        1990| annie|    1|
|        1990|  mark|    1|
|        1990|  lisa|    2|
|        1990| james|    3|
|        1990|andrew|    1|
|        1991|  mark|    1|
|        1991|  luke|    1|
|        1991| peter|    2|
|        1991|andrew|    1|
|        1991|  lisa|    1|
|        1991|thomas|    1|
|        1991|  john|    2|
|        1992|  luke|    1|
|        1992|andrew|    1|
|        1992|thomas|    2|
|        1992| annie|    1|
|        1992|  mark|    2|
|        1993| peter|    1|
|        1993| annie|    1|
|        1993|  mark|    3|
+------------+------+-----+
only showing top 20 rows
```

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



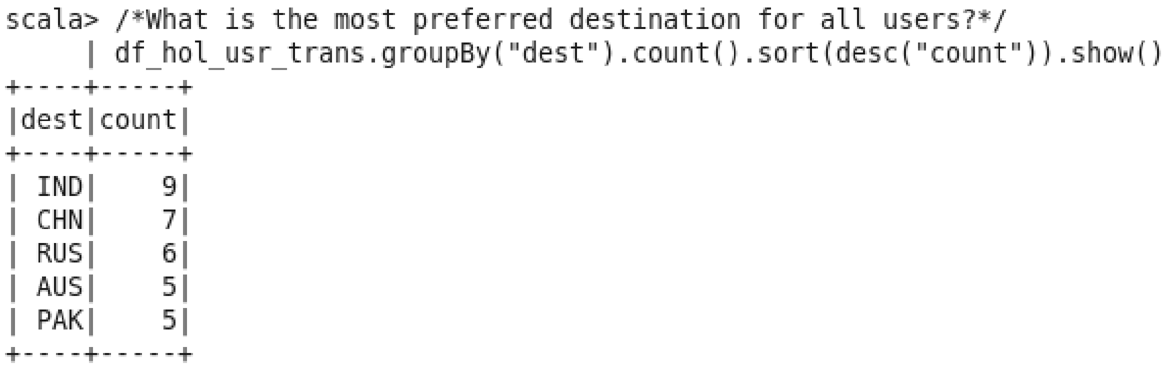
```
scala> /*What is the total air distance covered by each user per year?*/
     | df_hol_usr_trans.groupBy("yearoftravel","name").sum("distance").orderBy("yearoftravel").show()
+------------+------+-------------+
|yearoftravel|  name|sum(distance)|
+------------+------+-------------+
|        1990| annie|          200|
|        1990|  mark|          200|
|        1990|  lisa|          400|
|        1990| james|          600|
|        1990|andrew|          200|
|        1991|  mark|          200|
|        1991|  luke|          200|
|        1991| peter|          400|
|        1991|andrew|          200|
|        1991|  lisa|          200|
|        1991|thomas|          200|
|        1991|  john|          400|
|        1992|  luke|          200|
|        1992|andrew|          200|
|        1992|thomas|          400|
|        1992| annie|          200|
|        1992|  mark|          400|
|        1993| peter|          200|
|        1993| annie|          200|
|        1993|  mark|          600|
+------------+------+-------------+
only showing top 20 rows
```

3) Which user has travelled the largest distance till date



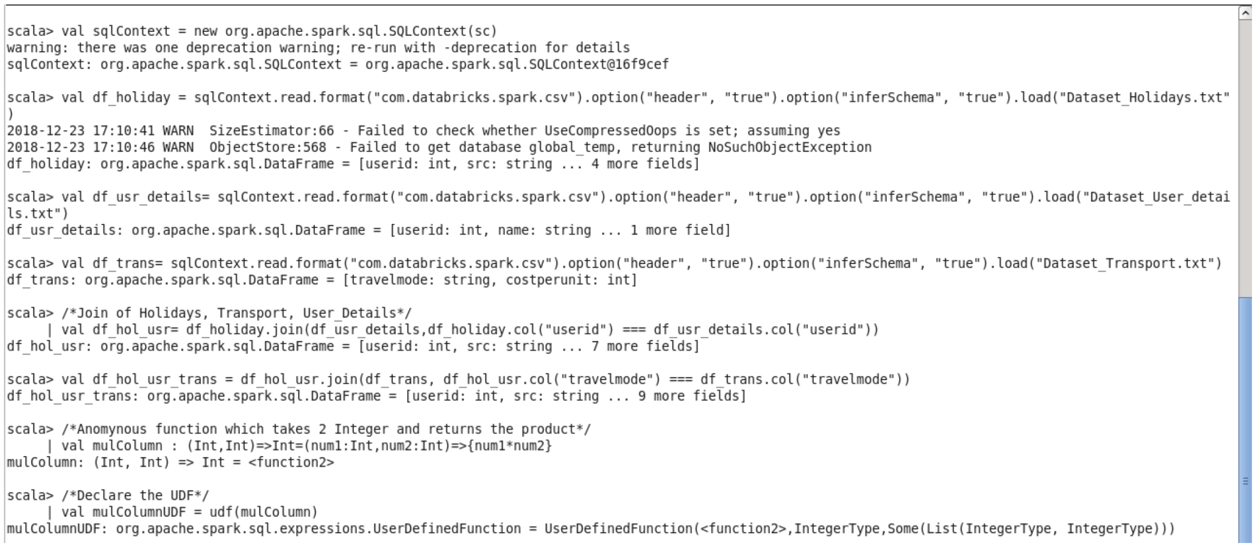
```
scala> /*Which user has travelled the largest distance till date?*/
     | df_hol_usr_trans.groupBy("name").sum("distance").sort(desc("sum(distance)")).show()
+------+-------------+
|  name|sum(distance)|
+------+-------------+
|  mark|         1600|
| peter|          600|
| annie|          600|
|  lisa|          600|
|andrew|          600|
|  john|          600|
|  luke|          600|
|thomas|          600|
| james|          600|
+------+-------------+
```

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

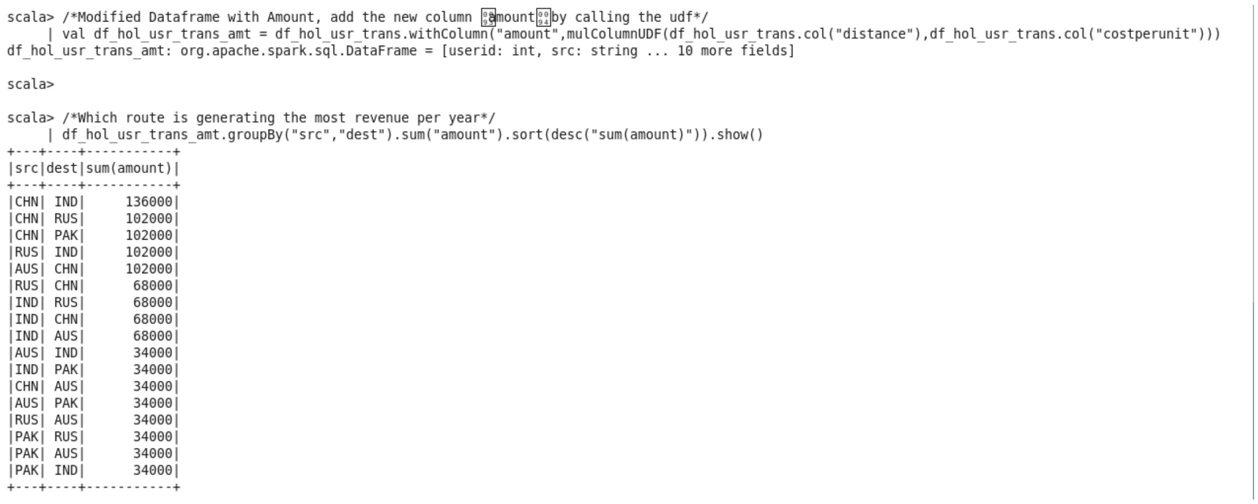


```
scala> /*What is the most preferred destination for all users?*/
     | df_hol_usr_trans.groupBy("dest").count().sort(desc("count")).show()
+----+-----+
|dest|count|
+----+-----+
| IND|    9|
| CHN|    7|
| RUS|    6|
| AUS|    5|
| PAK|    5|
+----+-----+
```

1) Which route is generating the most revenue per year

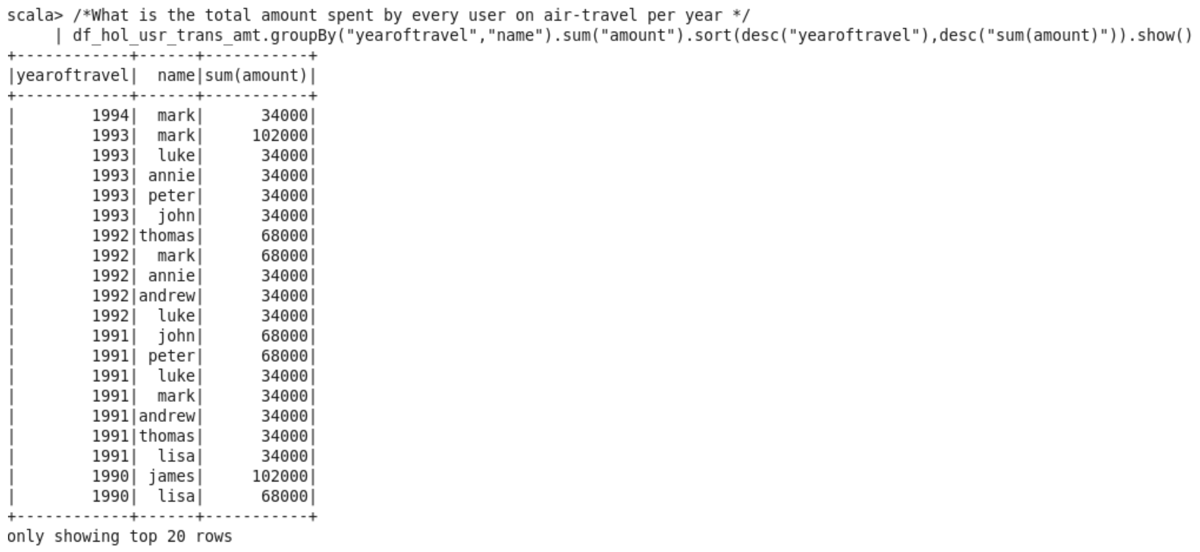


```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@16f9cef

scala> val df_holiday = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("Dataset_Holidays.txt"
)
2018-12-23 17:10:41 WARN  SizeEstimator:66 - Failed to check whether UseCompressedOops is set; assuming yes
2018-12-23 17:10:46 WARN  ObjectStore:568 - Failed to get database global_temp, returning NoSuchObjectException
df_holiday: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 4 more fields]

scala> val df_usr_details= sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("Dataset_User_detai
ls.txt")
df_usr_details: org.apache.spark.sql.DataFrame = [userid: int, name: string ... 1 more field]

scala> val df_trans= sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("Dataset_Transport.txt")
df_trans: org.apache.spark.sql.DataFrame = [travelmode: string, costperunit: int]

scala> /*Join of Holidays, Transport, User_Details*/
     | val df_hol_usr= df_holiday.join(df_usr_details,df_holiday.col("userid") === df_usr_details.col("userid"))
df_hol_usr: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 7 more fields]

scala> val df_hol_usr_trans = df_hol_usr.join(df_trans, df_hol_usr.col("travelmode") === df_trans.col("travelmode"))
df_hol_usr_trans: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 9 more fields]

scala> /*Anomynous function which takes 2 Integer and returns the product*/
     | val mulColumn : (Int,Int)=>Int=(num1:Int,num2:Int)=>{num1*num2}
mulColumn: (Int, Int) => Int = <function2>

scala> /*Declare the UDF*/
     | val mulColumnUDF = udf(mulColumn)
mulColumnUDF: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,IntegerType,Some(List(IntegerType, IntegerType)))
```



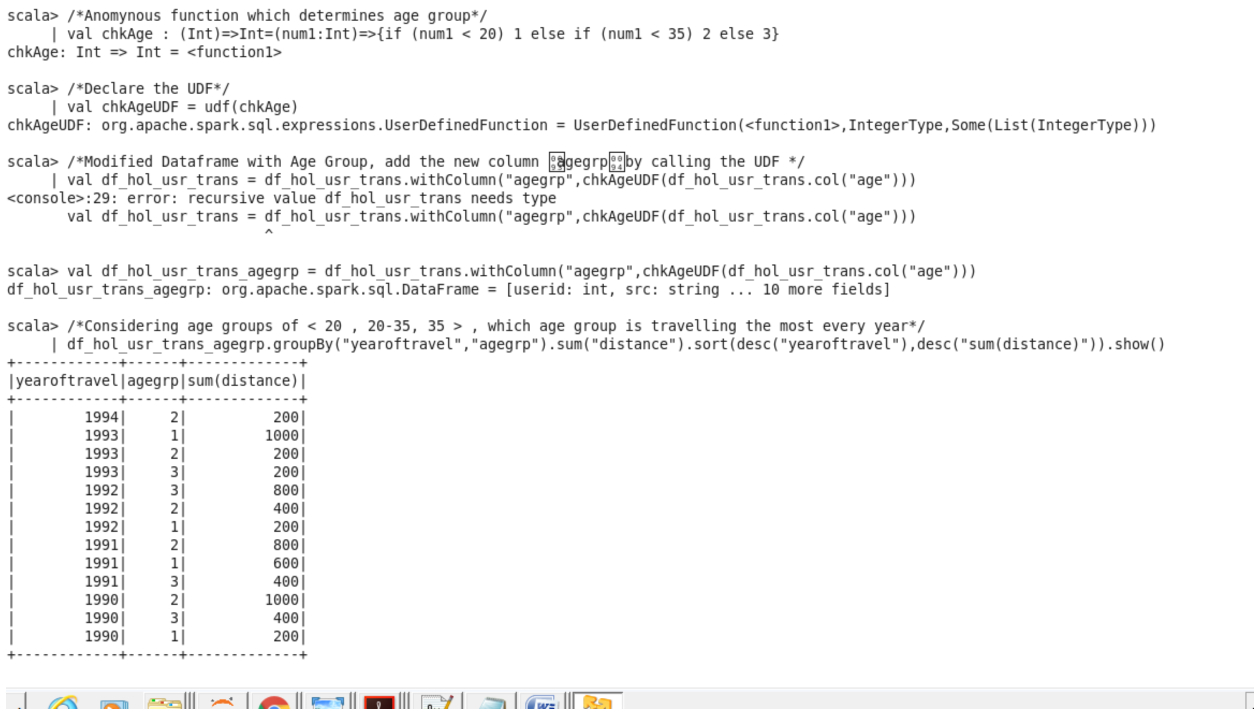
```
scala> /*Modified Dataframe with Amount, add the new column amount by calling the udf*/
     | val df_hol_usr_trans_amt = df_hol_usr_trans.withColumn("amount",mulColumnUDF(df_hol_usr_trans.col("distance"),df_hol_usr_trans.col("costperunit")))
df_hol_usr_trans_amt: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 10 more fields]

scala>

scala> /*Which route is generating the most revenue per year*/
     | df_hol_usr_trans_amt.groupBy("src","dest").sum("amount").sort(desc("sum(amount)")).show()
+---+----+-----------+
|src|dest|sum(amount)|
+---+----+-----------+
|CHN| IND|     136000|
|CHN| RUS|     102000|
|CHN| PAK|     102000|
|RUS| IND|     102000|
|AUS| CHN|     102000|
|RUS| CHN|      68000|
|IND| RUS|      68000|
|IND| CHN|      68000|
|IND| AUS|      68000|
|AUS| IND|      34000|
|IND| PAK|      34000|
|CHN| AUS|      34000|
|AUS| PAK|      34000|
|RUS| AUS|      34000|
|PAK| RUS|      34000|
|PAK| AUS|      34000|
|PAK| IND|      34000|
+---+----+-----------+
```

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



```
scala> /*What is the total amount spent by every user on air-travel per year */
     | df_hol_usr_trans_amt.groupBy("yearoftravel","name").sum("amount").sort(desc("yearoftravel"),desc("sum(amount)")).show()
+------------+------+-----------+
|yearoftravel|  name|sum(amount)|
+------------+------+-----------+
|        1994|  mark|      34000|
|        1993|  mark|     102000|
|        1993|  luke|      34000|
|        1993| annie|      34000|
|        1993| peter|      34000|
|        1993|  john|      34000|
|        1992|thomas|      68000|
|        1992|  mark|      68000|
|        1992| annie|      34000|
|        1992|andrew|      34000|
|        1992|  luke|      34000|
|        1991|  john|      68000|
|        1991| peter|      68000|
|        1991|  luke|      34000|
|        1991|  mark|      34000|
|        1991|andrew|      34000|
|        1991|thomas|      34000|
|        1991|  lisa|      34000|
|        1990| james|     102000|
|        1990|  lisa|      68000|
+------------+------+-----------+
only showing top 20 rows
```

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



```
scala> /*Anomynous function which determines age group*/
     | val chkAge : (Int)=>Int=(num1:Int)=>{if (num1 < 20) 1 else if (num1 < 35) 2 else 3}
chkAge: Int => Int = <function1>

scala> /*Declare the UDF*/
     | val chkAgeUDF = udf(chkAge)
chkAgeUDF: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,IntegerType,Some(List(IntegerType)))

scala> /*Modified Dataframe with Age Group, add the new column �agegrp�by calling the UDF */
     | val df_hol_usr_trans = df_hol_usr_trans.withColumn("agegrp",chkAgeUDF(df_hol_usr_trans.col("age")))
<console>:29: error: recursive value df_hol_usr_trans needs type
       val df_hol_usr_trans = df_hol_usr_trans.withColumn("agegrp",chkAgeUDF(df_hol_usr_trans.col("age")))
                              ^

scala> val df_hol_usr_trans_agegrp = df_hol_usr_trans.withColumn("agegrp",chkAgeUDF(df_hol_usr_trans.col("age")))
df_hol_usr_trans_agegrp: org.apache.spark.sql.DataFrame = [userid: int, src: string ... 10 more fields]

scala> /*Considering age groups of < 20 , 20-35, 35 > , which age group is travelling the most every year*/
     | df_hol_usr_trans_agegrp.groupBy("yearoftravel","agegrp").sum("distance").sort(desc("yearoftravel"),desc("sum(distance)")).show()
+------------+------+-------------+
|yearoftravel|agegrp|sum(distance)|
+------------+------+-------------+
|        1994|     2|          200|
|        1993|     1|         1000|
|        1993|     2|          200|
|        1993|     3|          200|
|        1992|     3|          800|
|        1992|     2|          400|
|        1992|     1|          200|
|        1991|     2|          800|
|        1991|     1|          600|
|        1991|     3|          400|
|        1990|     2|         1000|
|        1990|     3|          400|
|        1990|     1|          200|
+------------+------+-------------+
scala> df_hol_usr_trans_agegrp.show()
+------+---+----+----------+--------+------------+------+------+---+----------+-----------+------+
|userid|src|dest|travelmode|distance|yearoftravel|userid|  name|age|travelmode|costperunit|agegrp|
+------+---+----+----------+--------+------------+------+------+---+----------+-----------+------+
|     1|CHN| IND|  airplane|     200|        1990|     1|  mark| 15|  airplane|        170|     1|
|     2|IND| CHN|  airplane|     200|        1991|     2|  john| 16|  airplane|        170|     1|
|     3|IND| CHN|  airplane|     200|        1992|     3|  luke| 17|  airplane|        170|     1|
|     4|RUS| IND|  airplane|     200|        1990|     4|  lisa| 27|  airplane|        170|     2|
|     5|CHN| RUS|  airplane|     200|        1992|     5|  mark| 25|  airplane|        170|     2|
|     6|AUS| PAK|  airplane|     200|        1991|     6| peter| 22|  airplane|        170|     2|
|     7|RUS| AUS|  airplane|     200|        1990|     7| james| 21|  airplane|        170|     2|
|     8|IND| RUS|  airplane|     200|        1991|     8|andrew| 55|  airplane|        170|     3|
|     9|CHN| RUS|  airplane|     200|        1992|     9|thomas| 46|  airplane|        170|     3|
|    10|AUS| CHN|  airplane|     200|        1993|    10| annie| 44|  airplane|        170|     3|
|     1|AUS| CHN|  airplane|     200|        1993|     1|  mark| 15|  airplane|        170|     1|
|     2|CHN| IND|  airplane|     200|        1993|     2|  john| 16|  airplane|        170|     1|
|     3|CHN| IND|  airplane|     200|        1993|     3|  luke| 17|  airplane|        170|     1|
|     4|IND| AUS|  airplane|     200|        1991|     4|  lisa| 27|  airplane|        170|     2|
|     5|AUS| IND|  airplane|     200|        1992|     5|  mark| 25|  airplane|        170|     2|
|     6|RUS| CHN|  airplane|     200|        1993|     6| peter| 22|  airplane|        170|     2|
|     7|CHN| RUS|  airplane|     200|        1990|     7| james| 21|  airplane|        170|     2|
|     8|AUS| CHN|  airplane|     200|        1990|     8|andrew| 55|  airplane|        170|     3|
|     9|IND| AUS|  airplane|     200|        1991|     9|thomas| 46|  airplane|        170|     3|
|    10|RUS| CHN|  airplane|     200|        1992|    10| annie| 44|  airplane|        170|     3|
+------+---+----+----------+--------+------------+------+------+---+----------+-----------+------+
only showing top 20 rows
```