1. Which route is generating the most revenue per year

*case class holidays(user\_id: Int, src: String, dest: String, travel\_mode: String, distance: Int, year\_of\_travel: Int)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_Holidays.txt")*

*val holidayRDD = rowsRDD.map{row => row.split(",")}.map{cols => holidays(cols(0).toInt, cols(1), cols(2), cols(3), cols(4).toInt, cols(5).toInt )}*

*val holidayDF = holidayRDD.toDF()*

*holidayDF.registerTempTable("holidays")*

*case class users(user\_id: Int, name: String, age: String)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_User\_details.txt")*

*val usersRDD = rowsRDD.map{row => row.split(",")}.map{cols => users(cols(0).toInt, cols(1), cols(2) )}*

*val usersDF = usersRDD.toDF()*

*usersDF.registerTempTable("users")*

*case class transport(travel\_mode: String, cost: Int)*

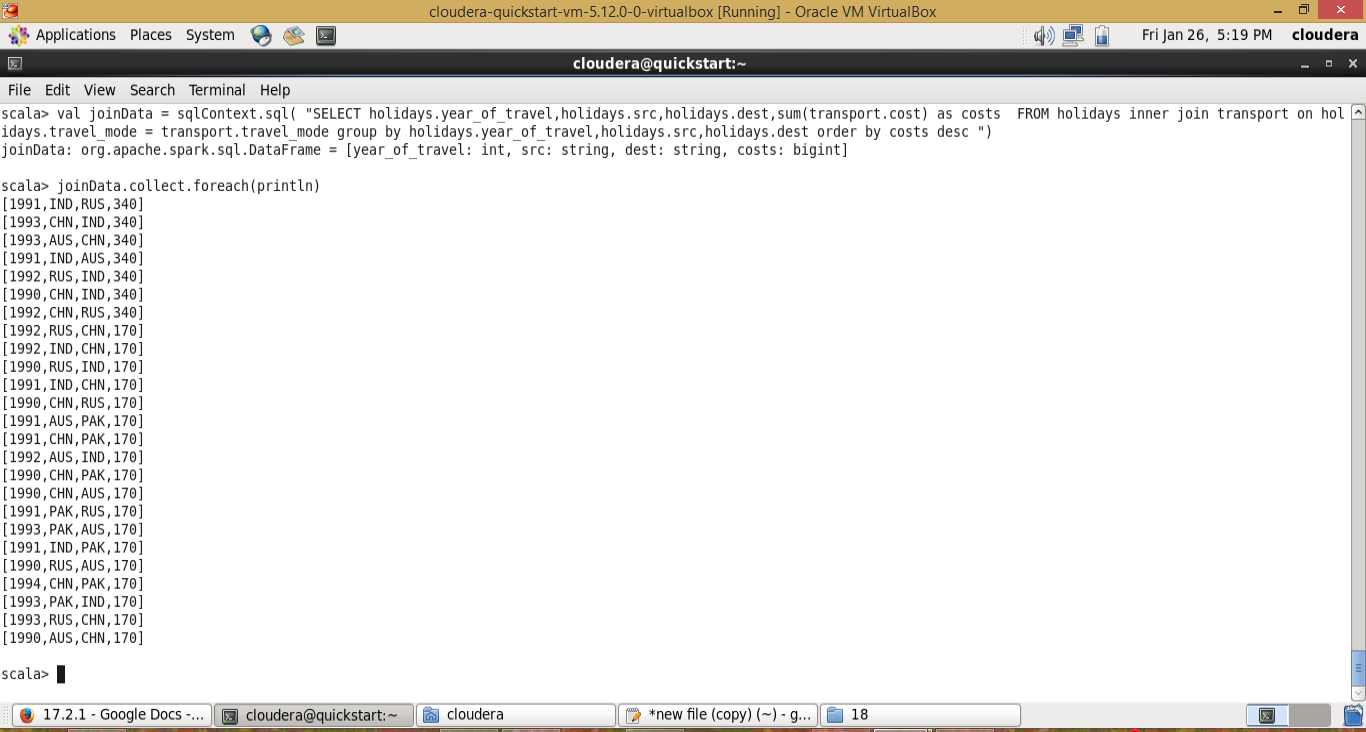
*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_Transport.txt")*

*val transportRDD = rowsRDD.map{row => row.split(",")}.map{cols => transport(cols(0), cols(1).toInt)}*

*val transportDF = transportRDD.toDF()*

*transportDF.registerTempTable("transport")*

*val joinData = sqlContext.sql( "SELECT holidays.year\_of\_travel,holidays.src,holidays.dest,sum(transport.cost) as costs  FROM holidays inner join transport on holidays.travel\_mode = transport.travel\_mode group by holidays.year\_of\_travel,holidays.src,holidays.dest order by costs desc ")*



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

*case class holidays(user\_id: Int, src: String, dest: String, travel\_mode: String, distance: Int, year\_of\_travel: Int)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_Holidays.txt")*

*val holidayRDD = rowsRDD.map{row => row.split(",")}.map{cols => holidays(cols(0).toInt, cols(1), cols(2), cols(3), cols(4).toInt, cols(5).toInt )}*

*val holidayDF = holidayRDD.toDF()*

*holidayDF.registerTempTable("holidays")*

*case class users(user\_id: Int, name: String, age: String)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_User\_details.txt")*

*val usersRDD = rowsRDD.map{row => row.split(",")}.map{cols => users(cols(0).toInt, cols(1), cols(2) )}*

*val usersDF = usersRDD.toDF()*

*usersDF.registerTempTable("users")*

*case class transport(travel\_mode: String, cost: Int)*

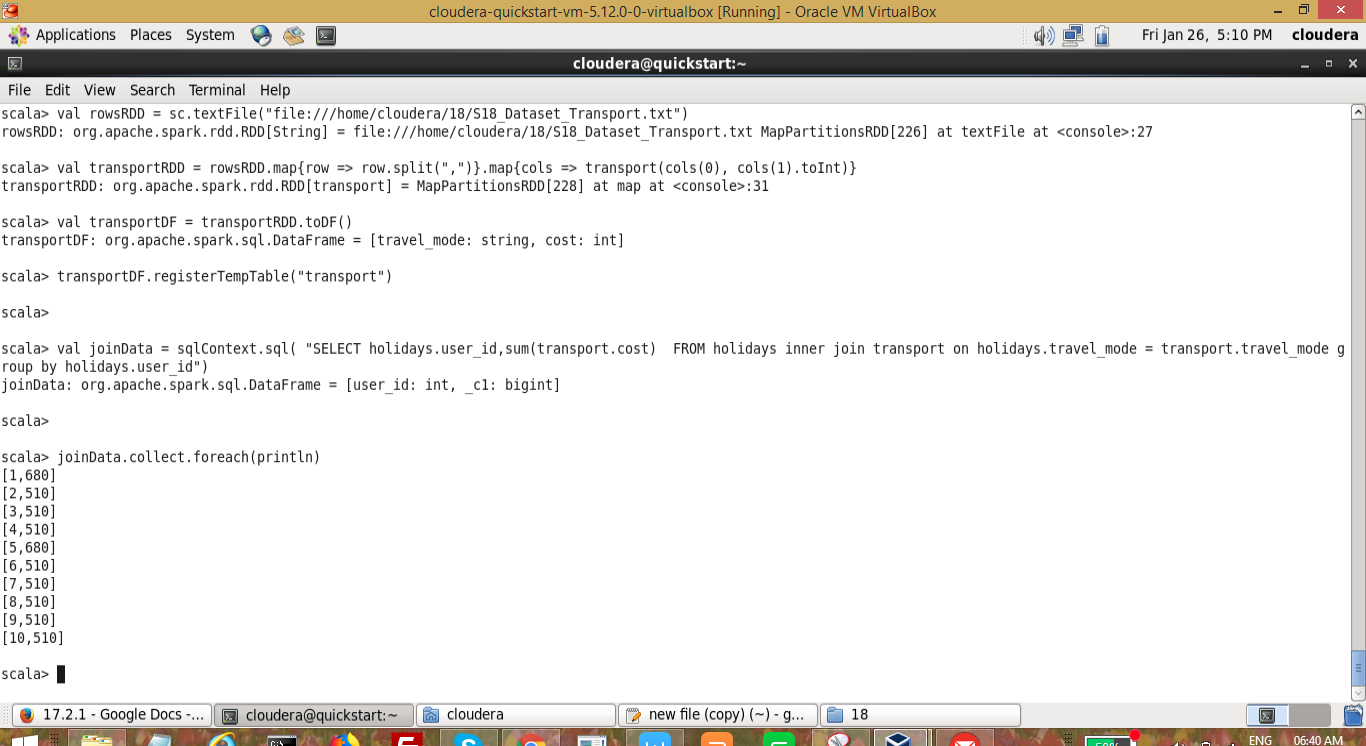
*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_Transport.txt")*

*val transportRDD = rowsRDD.map{row => row.split(",")}.map{cols => transport(cols(0), cols(1).toInt)}*

*val transportDF = transportRDD.toDF()*

*transportDF.registerTempTable("transport")*

*val joinData = sqlContext.sql( "SELECT holidays.user\_id,sum(transport.cost)  FROM holidays inner join transport on holidays.travel\_mode = transport.travel\_mode group by holidays.user\_id")*



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

every year.

Use the dataset given below:

*case class holidays(user\_id: Int, src: String, dest: String, travel\_mode: String, distance: Int, year\_of\_travel: Int)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_Holidays.txt")*

*val holidayRDD = rowsRDD.map{row => row.split(",")}.map{cols => holidays(cols(0).toInt, cols(1), cols(2), cols(3), cols(4).toInt, cols(5).toInt )}*

*val holidayDF = holidayRDD.toDF()*

*holidayDF.registerTempTable("holidays")*

*case class users(user\_id: Int, name: String, age: String)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_User\_details.txt")*

*val usersRDD = rowsRDD.map{row => row.split(",")}.map{cols => users(cols(0).toInt, cols(1), cols(2) )}*

*val usersDF = usersRDD.toDF()*

*usersDF.registerTempTable("users")*

*case class transport(travel\_mode: String, cost: Int)*

*val rowsRDD = sc.textFile("file:///home/cloudera/18/S18\_Dataset\_Transport.txt")*

*val transportRDD = rowsRDD.map{row => row.split(",")}.map{cols => transport(cols(0), cols(1).toInt)}*

*val transportDF = transportRDD.toDF()*

*transportDF.registerTempTable("transport")*

*def ageGroup = (age:Int) => {*

*var subrank = "";*

*if(age >= 32){subrank = "pro";  }*

*else if(age <20 ){ subrank = "small";  }*

*else if(age >= 20 && age <=35 ){ subrank = "medium";  }*

*else if(age > 35){ subrank = "high";  }*

*subrank*

*}*

*sqlContext.udf.register("agegroup",ageGroup)*

*val joinData = sqlContext.sql( "SELECT agegroup(users.age) as agegroup,holidays.year\_of\_travel,count(holidays.user\_id)  FROM holidays inner join users on holidays.user\_id = users.user\_id  group by agegroup(users.age),holidays.year\_of\_travel order by count(holidays.user\_id) desc limit 1")*

