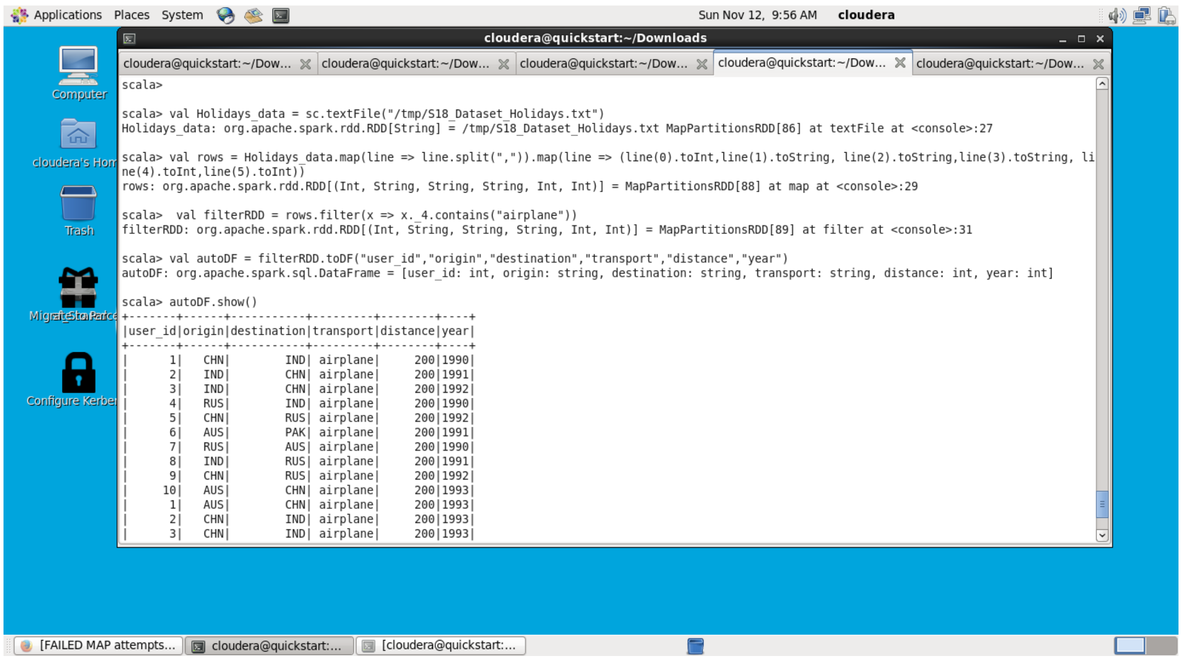
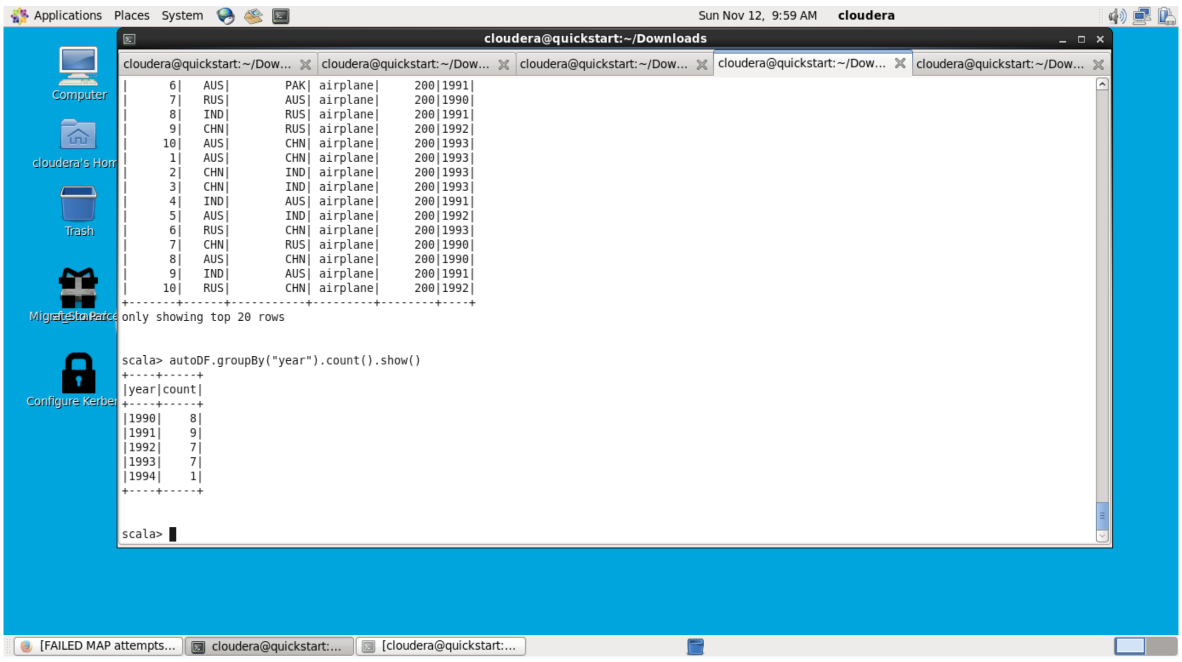
Convert text file to data frame to run queries:



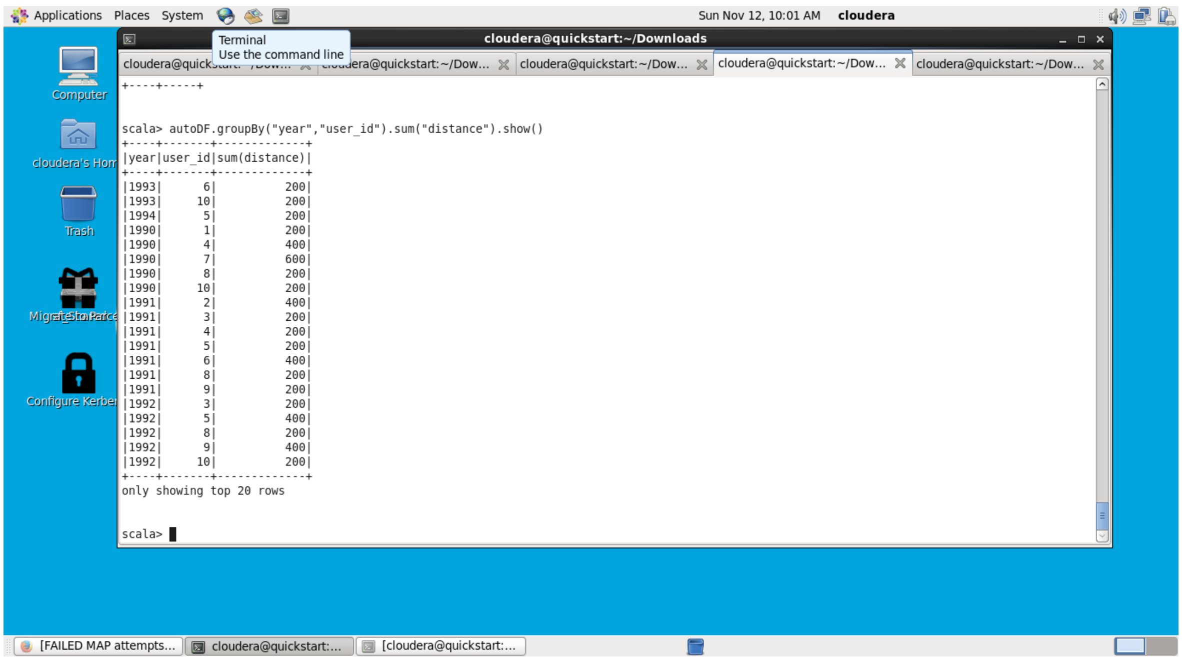
```
scala>

scala> val Holidays_data = sc.textFile("/tmp/S18_Dataset_Holidays.txt")
Holidays_data: org.apache.spark.rdd.RDD[String] = /tmp/S18_Dataset_Holidays.txt MapPartitionsRDD[86] at textFile at <console>:27

scala> val rows = Holidays_data.map(line => line.split(",")).map(line => (line(0).toInt,line(1).toString, line(2).toString,line(3).toString, li
ne(4).toInt,line(5).toInt))
rows: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[88] at map at <console>:29

scala>  val filterRDD = rows.filter(x => x._4.contains("airplane"))
filterRDD: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int)] = MapPartitionsRDD[89] at filter at <console>:31

scala> val autoDF = filterRDD.toDF("user_id","origin","destination","transport","distance","year")
autoDF: org.apache.spark.sql.DataFrame = [user_id: int, origin: string, destination: string, transport: string, distance: int, year: int]

scala> autoDF.show()
+-------+------+-----------+---------+--------+----+
|user_id|origin|destination|transport|distance|year|
+-------+------+-----------+---------+--------+----+
|      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|
```

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



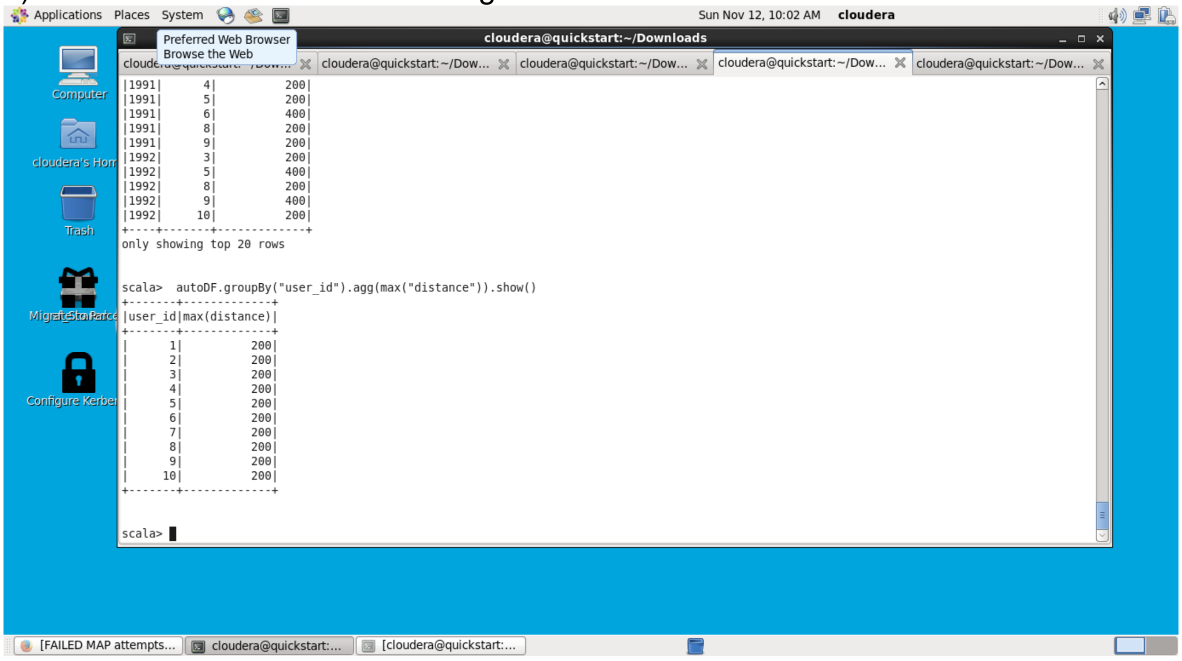
```
|      6|   AUS|        PAK| airplane|     200|1991|
|      7|   RUS|        AUS| airplane|     200|1990|
|      8|   IND|        RUS| airplane|     200|1991|
|      9|   CHN|        RUS| airplane|     200|1992|
|     10|   AUS|        CHN| airplane|     200|1993|
|      1|   AUS|        CHN| airplane|     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|
+-------+------+-----------+---------+--------+----+
only showing top 20 rows


scala> autoDF.groupBy("year").count().show()
+----+-----+
|year|count|
+----+-----+
|1990|    8|
|1991|    9|
|1992|    7|
|1993|    7|
|1994|    1|
+----+-----+


scala> 
```

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



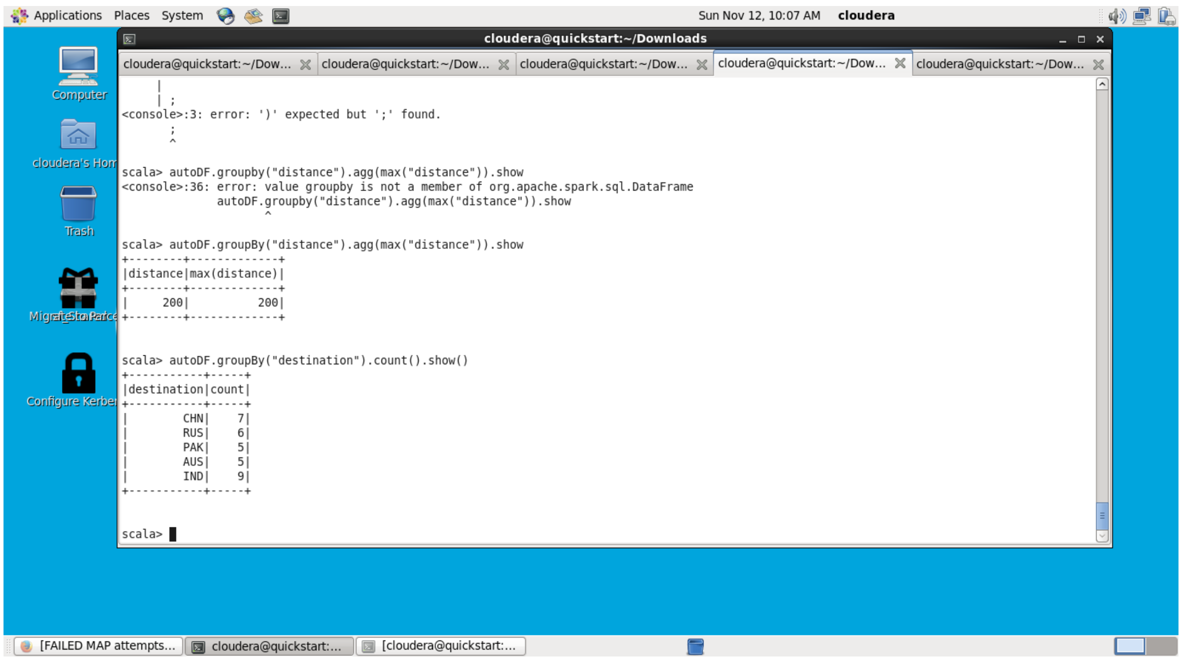
```
+----+-----+


scala> autoDF.groupBy("year","user_id").sum("distance").show()
+----+-------+-------------+
|year|user_id|sum(distance)|
+----+-------+-------------+
|1993|      6|          200|
|1993|     10|          200|
|1994|      5|          200|
|1990|      1|          200|
|1990|      4|          400|
|1990|      7|          600|
|1990|      8|          200|
|1990|     10|          200|
|1991|      2|          400|
|1991|      3|          200|
|1991|      4|          200|
|1991|      5|          200|
|1991|      6|          400|
|1991|      8|          200|
|1991|      9|          200|
|1992|      3|          200|
|1992|      5|          400|
|1992|      8|          200|
|1992|      9|          400|
|1992|     10|          200|
+----+-------+-------------+
only showing top 20 rows


scala> 
```

3) Which user has travelled the largest distance till date



```
|1991|      4|          200|
|1991|      5|          200|
|1991|      6|          400|
|1991|      8|          200|
|1991|      9|          200|
|1992|      3|          200|
|1992|      5|          400|
|1992|      8|          200|
|1992|      9|          400|
|1992|     10|          200|
+----+-------+-------------+
only showing top 20 rows


scala>  autoDF.groupBy("user_id").agg(max("distance")).show()
+-------+-------------+
|user_id|max(distance)|
+-------+-------------+
|      1|          200|
|      2|          200|
|      3|          200|
|      4|          200|
|      5|          200|
|      6|          200|
|      7|          200|
|      8|          200|
|      9|          200|
|     10|          200|
+-------+-------------+


scala> 
```

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


Applications
Places
System
Sun Nov 12, 10:07 AM
cloudera
cloudera@quickstart:~/Downloads
cloudera@quickstart:~/Dow...
cloudera@quickstart:~/Dow...
cloudera@quickstart:~/Dow...
cloudera@quickstart:~/Dow...
cloudera@quickstart:~/Dow...
Computer
cloudera's Hom
Trash
Configure Kerber
[FAILED MAP attempts...
cloudera@quickstart:...
[cloudera@quickstart:...

```
      |
      | ;
<console>:3: error: ')' expected but ';' found.
       ;
       ^


scala> autoDF.groupby("distance").agg(max("distance")).show
<console>:36: error: value groupby is not a member of org.apache.spark.sql.DataFrame
              autoDF.groupby("distance").agg(max("distance")).show
                     ^


scala> autoDF.groupBy("distance").agg(max("distance")).show
+--------+-------------+
|distance|max(distance)|
+--------+-------------+
|     200|          200|
+--------+-------------+



scala> autoDF.groupBy("destination").count().show()
+-----------+-----+
|destination|count|
+-----------+-----+
|        CHN|    7|
|        RUS|    6|
|        PAK|    5|
|        AUS|    5|
|        IND|    9|
+-----------+-----+


scala> 
```