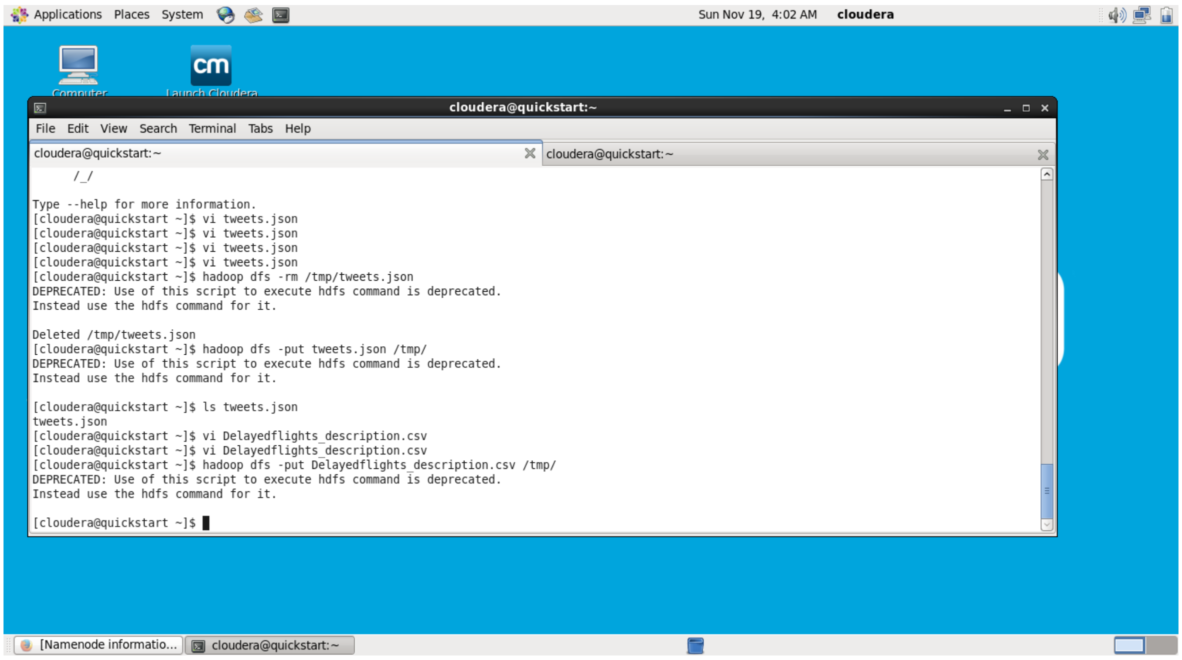
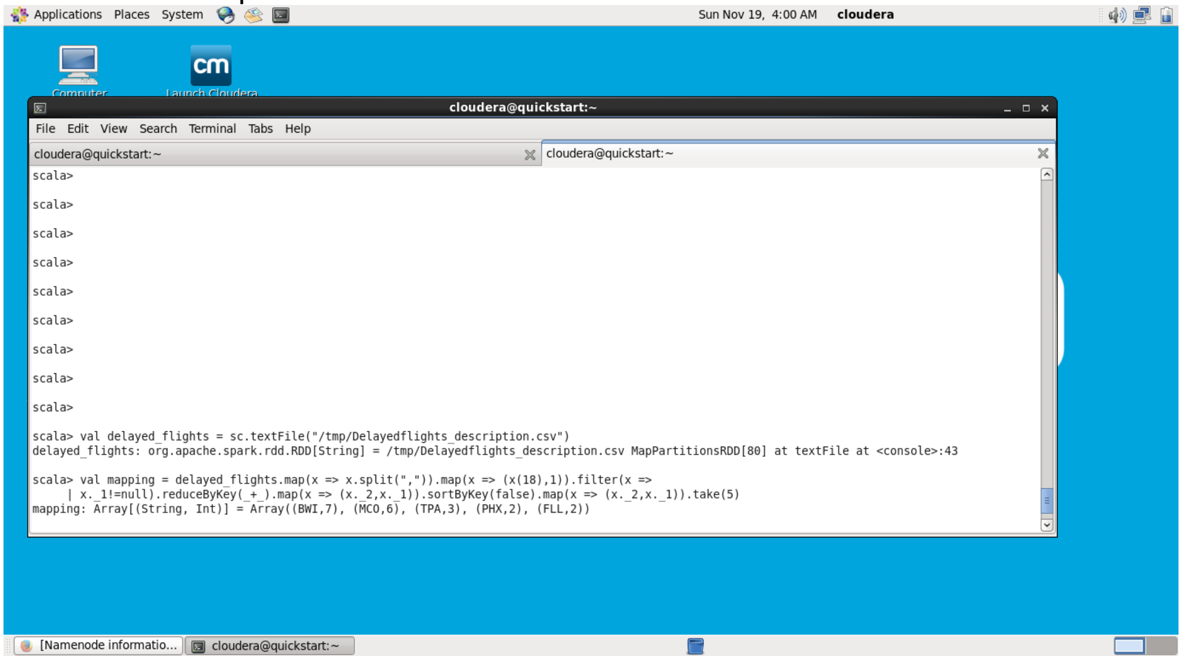
1. Copy delayed_flights.csv to HDFS using

NOTE: Copied only half the file as not able to copy full file from local system to cloudera, so some results are empty as there is no data.



```
     /_/

Type --help for more information.
[cloudera@quickstart ~]$ vi tweets.json
[cloudera@quickstart ~]$ vi tweets.json
[cloudera@quickstart ~]$ vi tweets.json
[cloudera@quickstart ~]$ vi tweets.json
[cloudera@quickstart ~]$ hadoop dfs -rm /tmp/tweets.json
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Deleted /tmp/tweets.json
[cloudera@quickstart ~]$ hadoop dfs -put tweets.json /tmp/
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

[cloudera@quickstart ~]$ ls tweets.json
tweets.json
[cloudera@quickstart ~]$ vi Delayedflights_description.csv
[cloudera@quickstart ~]$ vi Delayedflights_description.csv
[cloudera@quickstart ~]$ hadoop dfs -put Delayedflights_description.csv /tmp/
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

[cloudera@quickstart ~]$ 
```

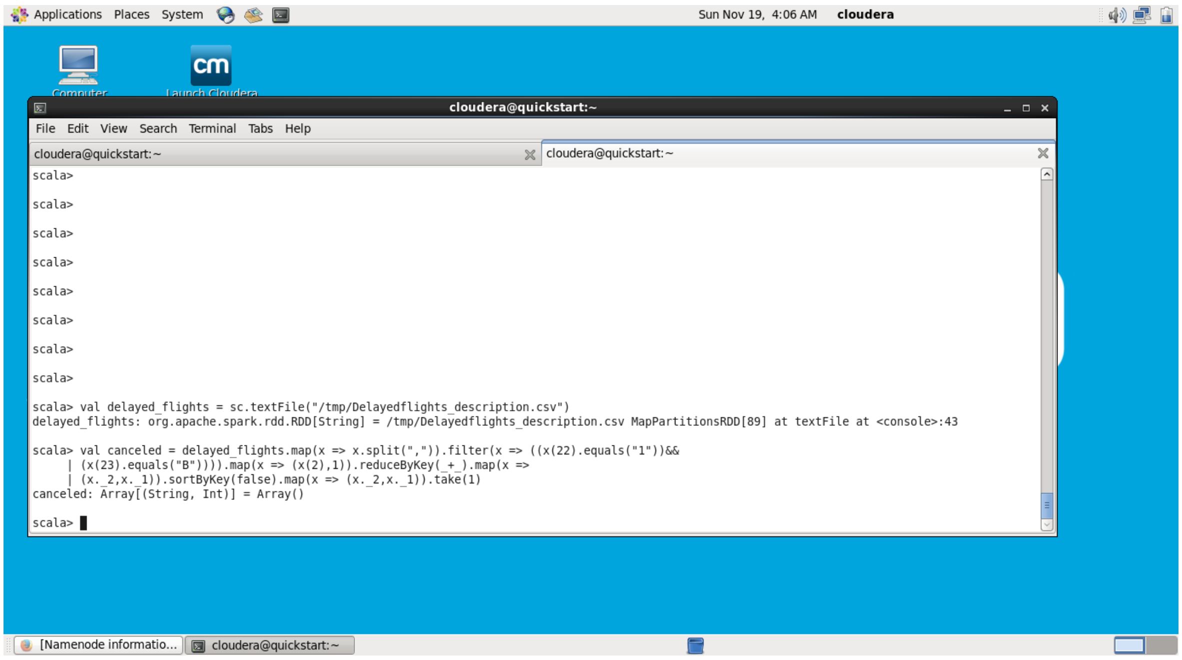
Problem statement 1:
1. Find out the top 5 most visited destinations.



```
scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala> val delayed_flights = sc.textFile("/tmp/Delayedflights_description.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /tmp/Delayedflights_description.csv MapPartitionsRDD[80] at textFile at <console>:43

scala> val mapping = delayed_flights.map(x => x.split(",")).map(x => (x(18),1)).filter(x =>
     | x._1!=null).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(5)
mapping: Array[(String, Int)] = Array((BWI,7), (MCO,6), (TPA,3), (PHX,2), (FLL,2))
```

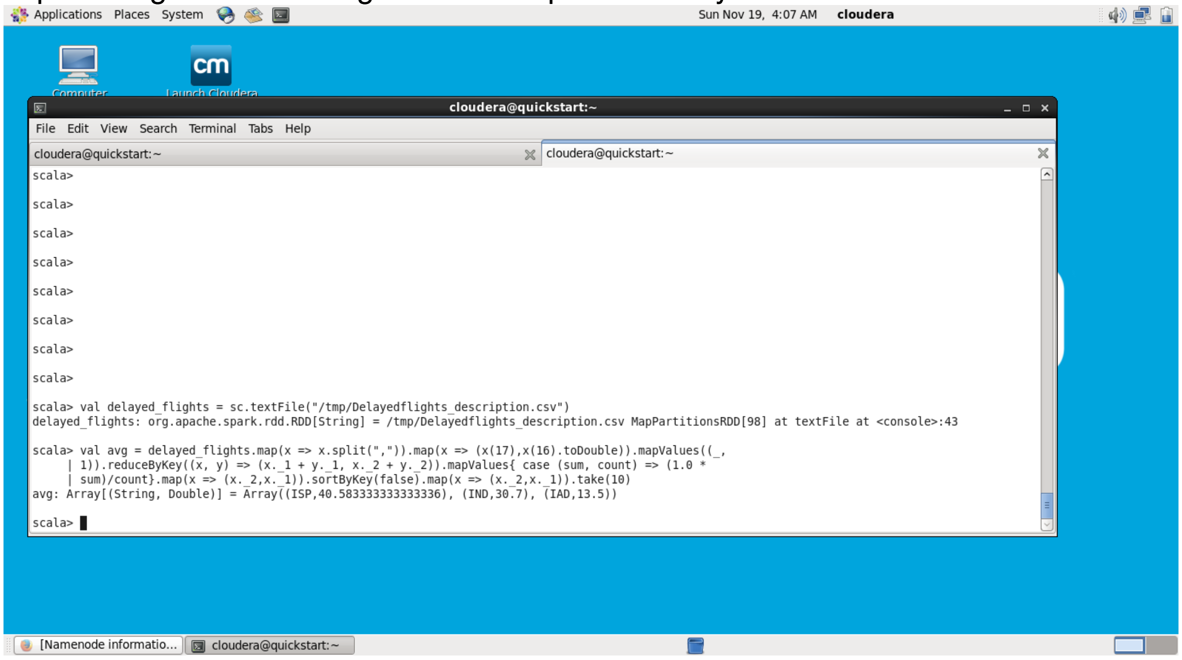
Problem statement 2

Which month has seen the most number of cancellations due to bad weather?



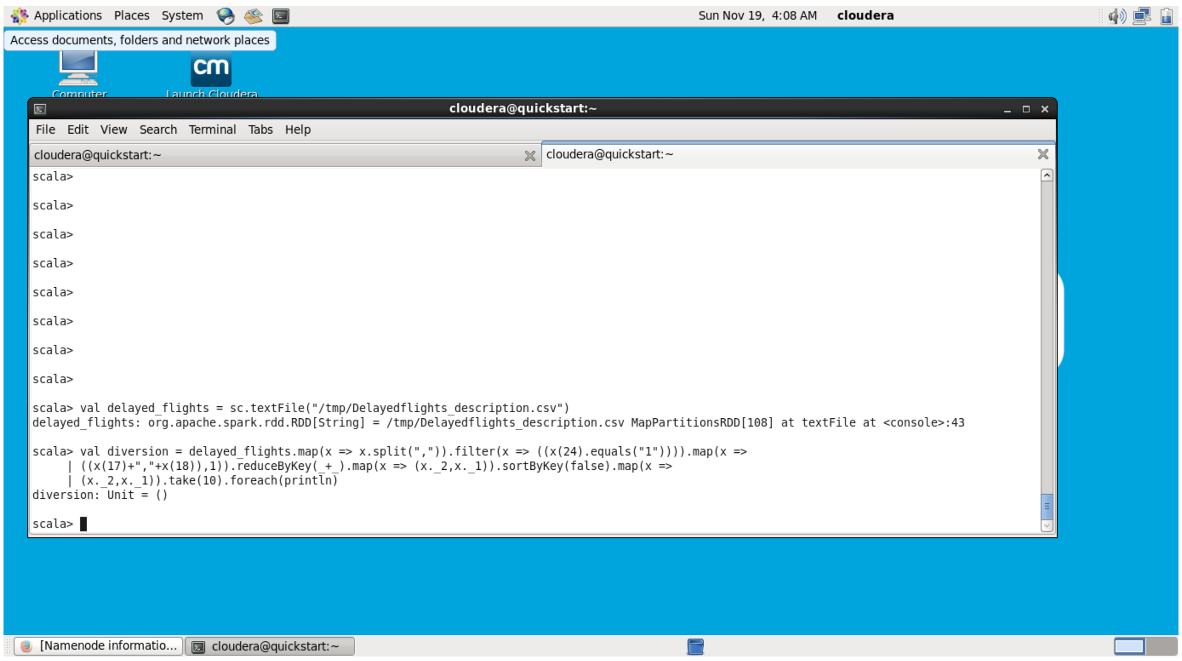
```
scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala> val delayed_flights = sc.textFile("/tmp/Delayedflights_description.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /tmp/Delayedflights_description.csv MapPartitionsRDD[89] at textFile at <console>:43

scala> val canceled = delayed_flights.map(x => x.split(",")).filter(x => ((x(22).equals("1"))&&
     | (x(23).equals("B")))).map(x => (x(2),1)).reduceByKey(_+_).map(x =>
     | (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(1)
canceled: Array[(String, Int)] = Array()

scala> 
```

Problem statement 3:
Top ten origins with the highest AVG departure delay



```
scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala> val delayed_flights = sc.textFile("/tmp/Delayedflights_description.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /tmp/Delayedflights_description.csv MapPartitionsRDD[98] at textFile at <console>:43

scala> val avg = delayed_flights.map(x => x.split(",")).map(x => (x(17),x(16).toDouble)).mapValues((_,
     | 1)).reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2)).mapValues{ case (sum, count) => (1.0 *
     | sum)/count}.map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(10)
avg: Array[(String, Double)] = Array((ISP,40.583333333333336), (IND,30.7), (IAD,13.5))

scala> 
```

Problem statement 4
Which route (origin & destination) has seen the maximum diversion?


Applications Places System
Sun Nov 19, 4:08 AM cloudera
Access documents, folders and network places
Computer
Launch Cloudera
cloudera@quickstart:~
File Edit View Search Terminal Tabs Help
cloudera@quickstart:~
cloudera@quickstart:~
[Namenode informatio...
cloudera@quickstart:~

```
scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala> val delayed_flights = sc.textFile("/tmp/Delayedflights_description.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /tmp/Delayedflights_description.csv MapPartitionsRDD[108] at textFile at <console>:43

scala> val diversion = delayed_flights.map(x => x.split(",")).filter(x => ((x(24).equals("1")))).map(x =>
     | ((x(17)+","+x(18)),1)).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x =>
     | (x._2,x._1)).take(10).foreach(println)
diversion: Unit = ()

scala> 
```