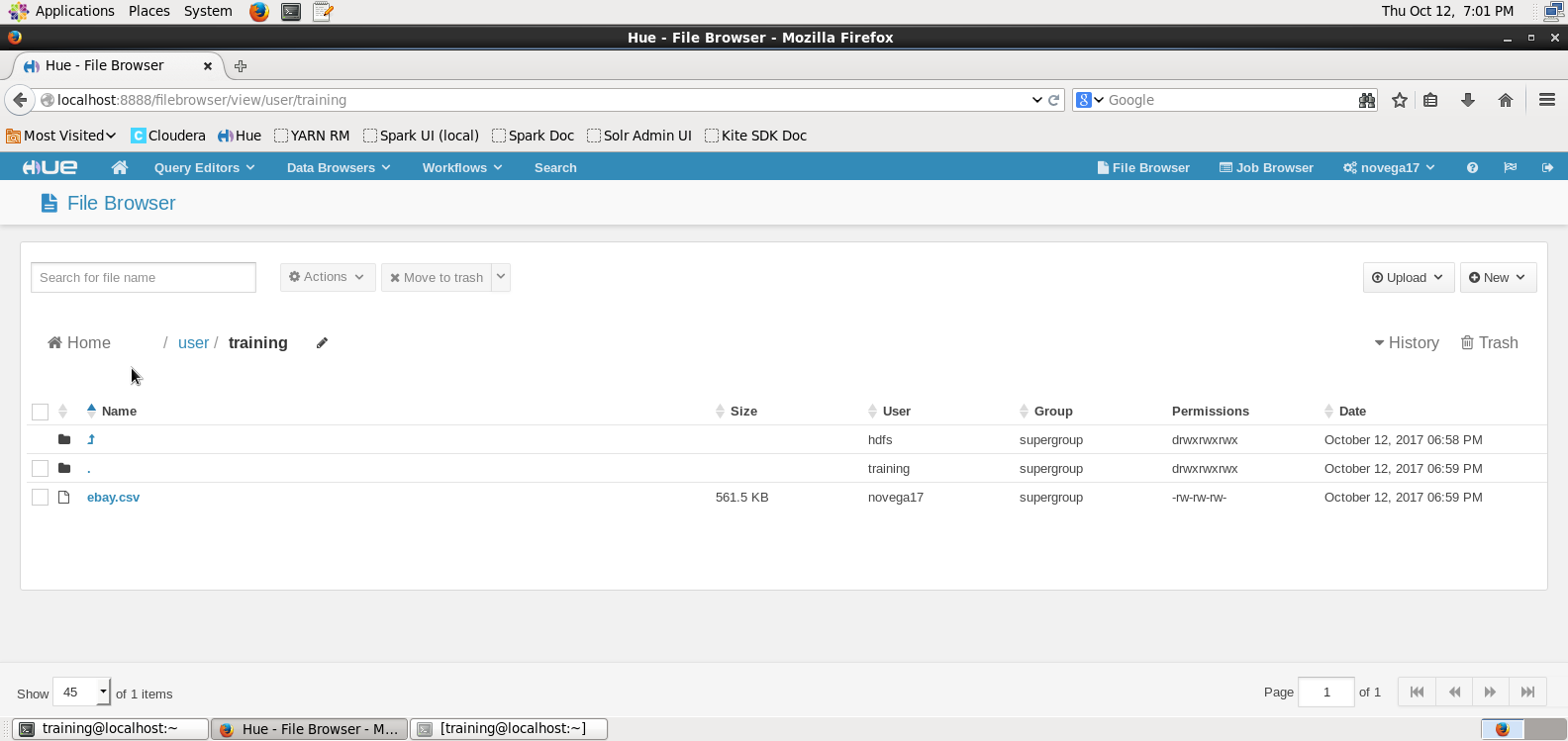
Assignment AS#2

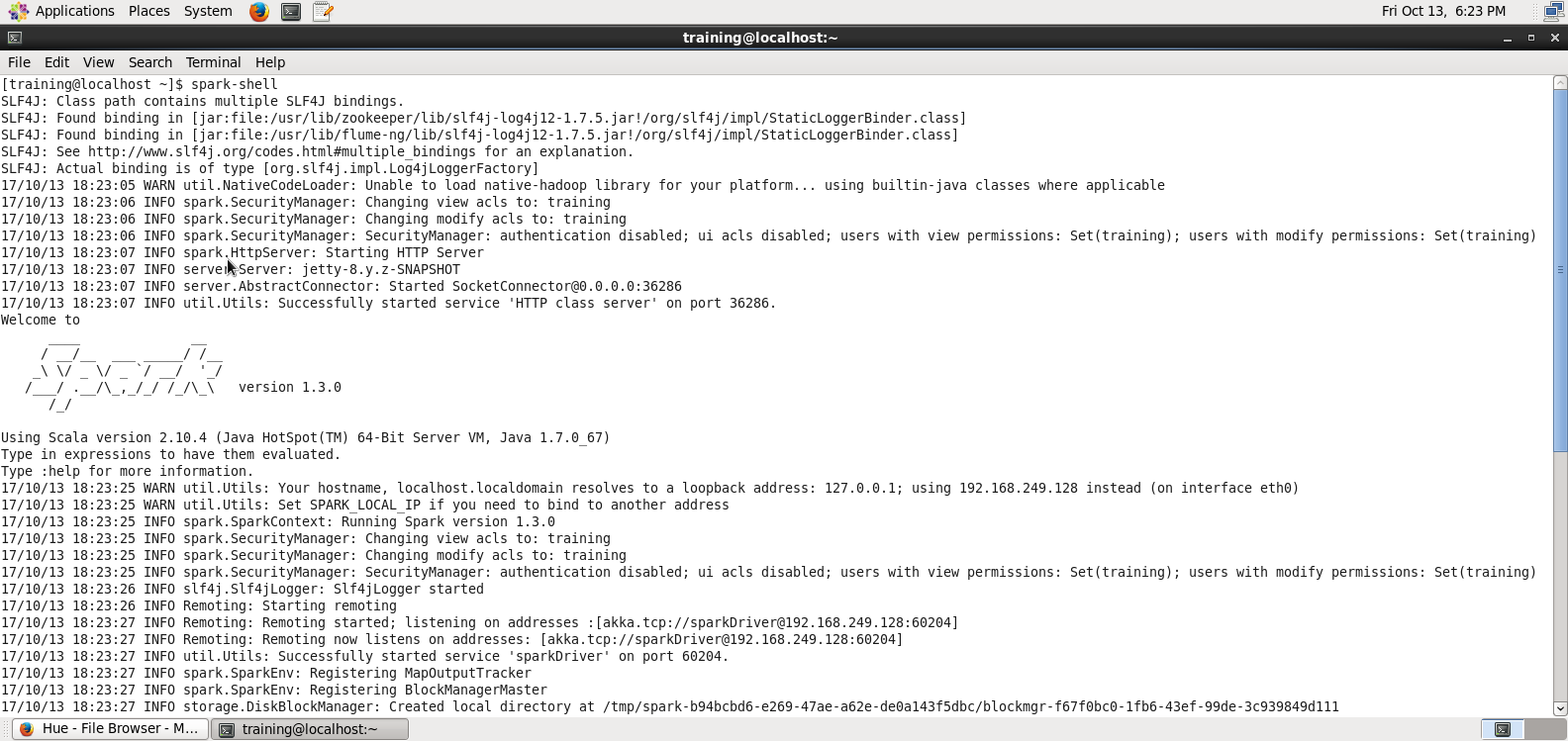
Part A: download and put into HDFS/user/training using HUE for steps 1 & 2



3

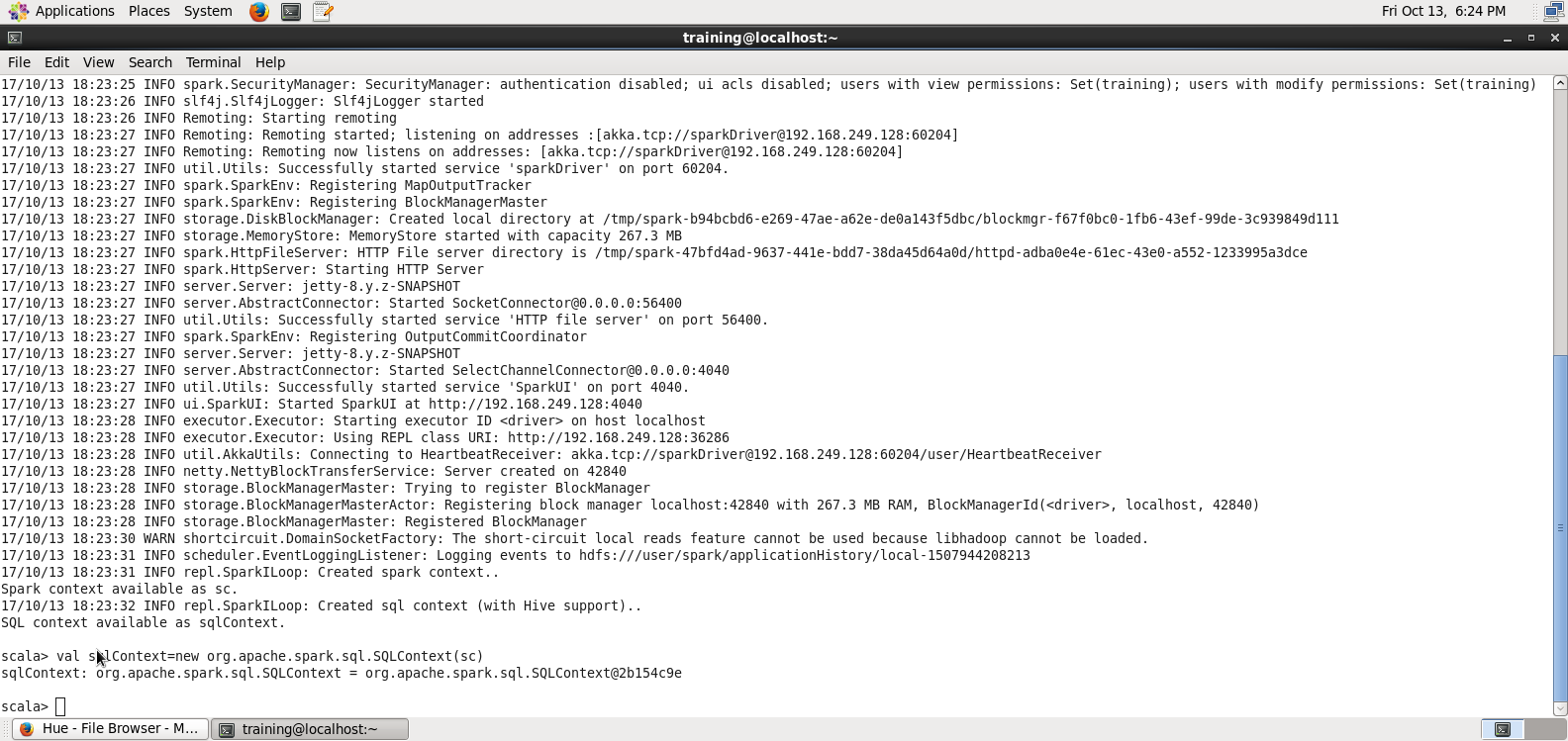
At the $:

spark-shell

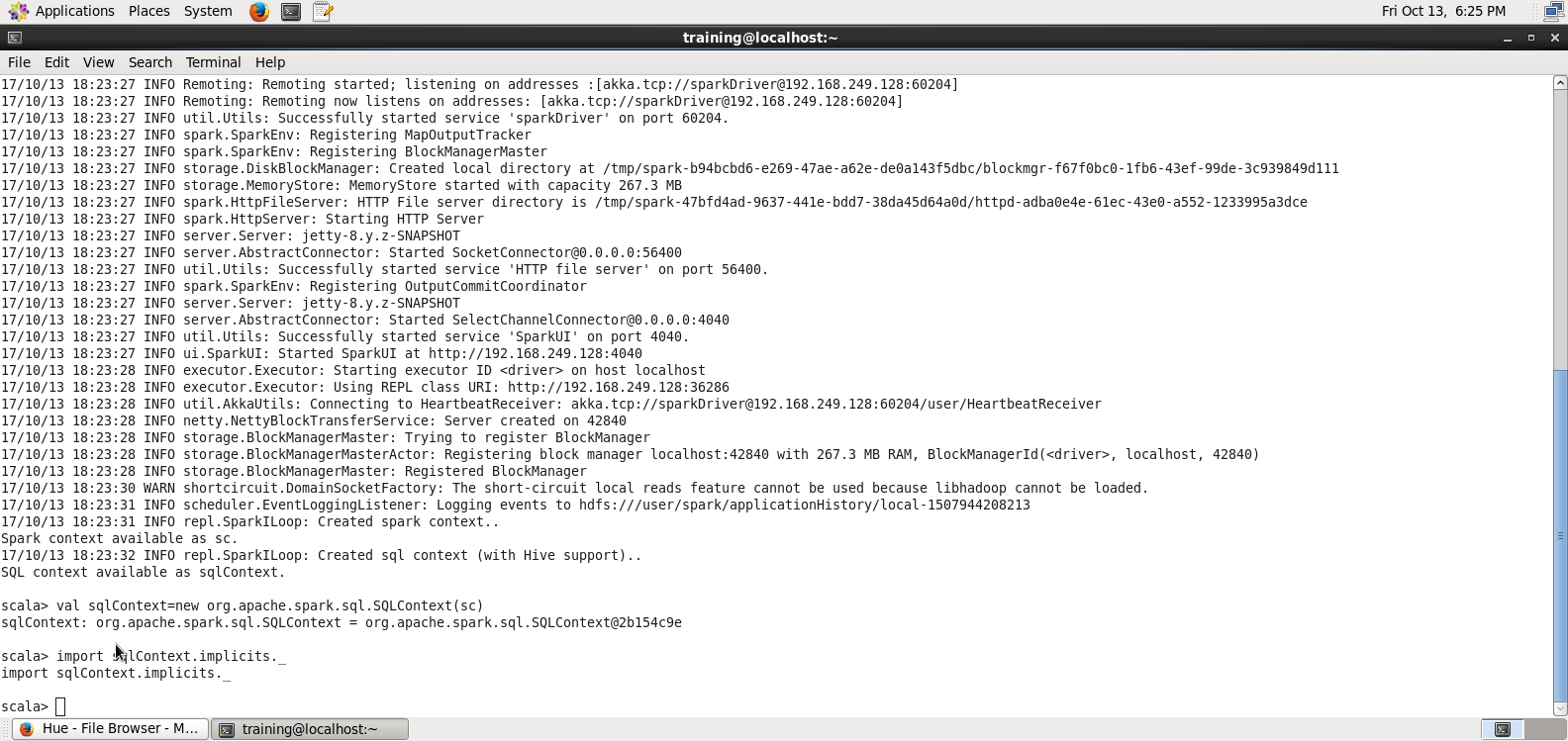


4

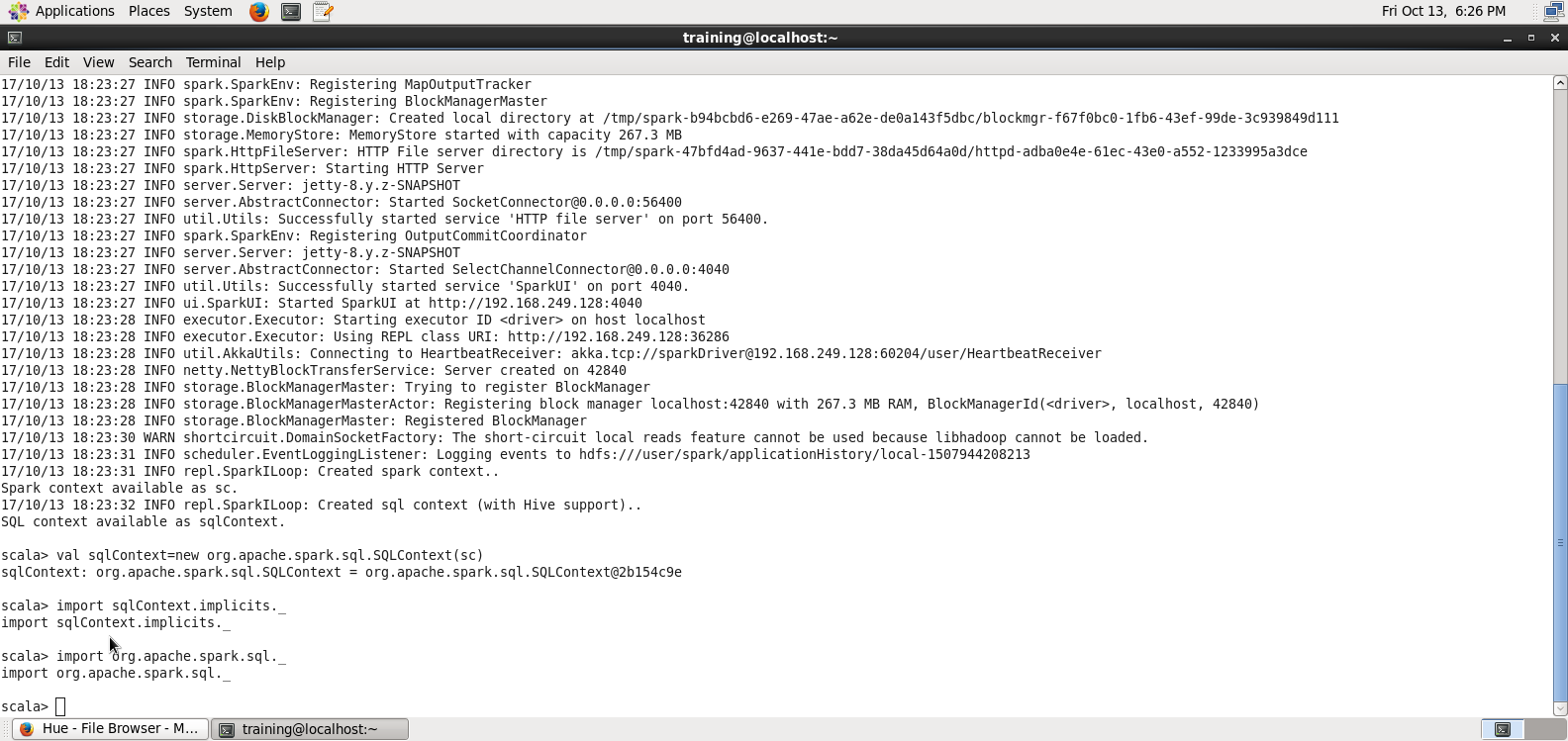
val sqlContext=new org.apache.spark.sql.SQLContext(sc)



import sqlContext.implicits.\_



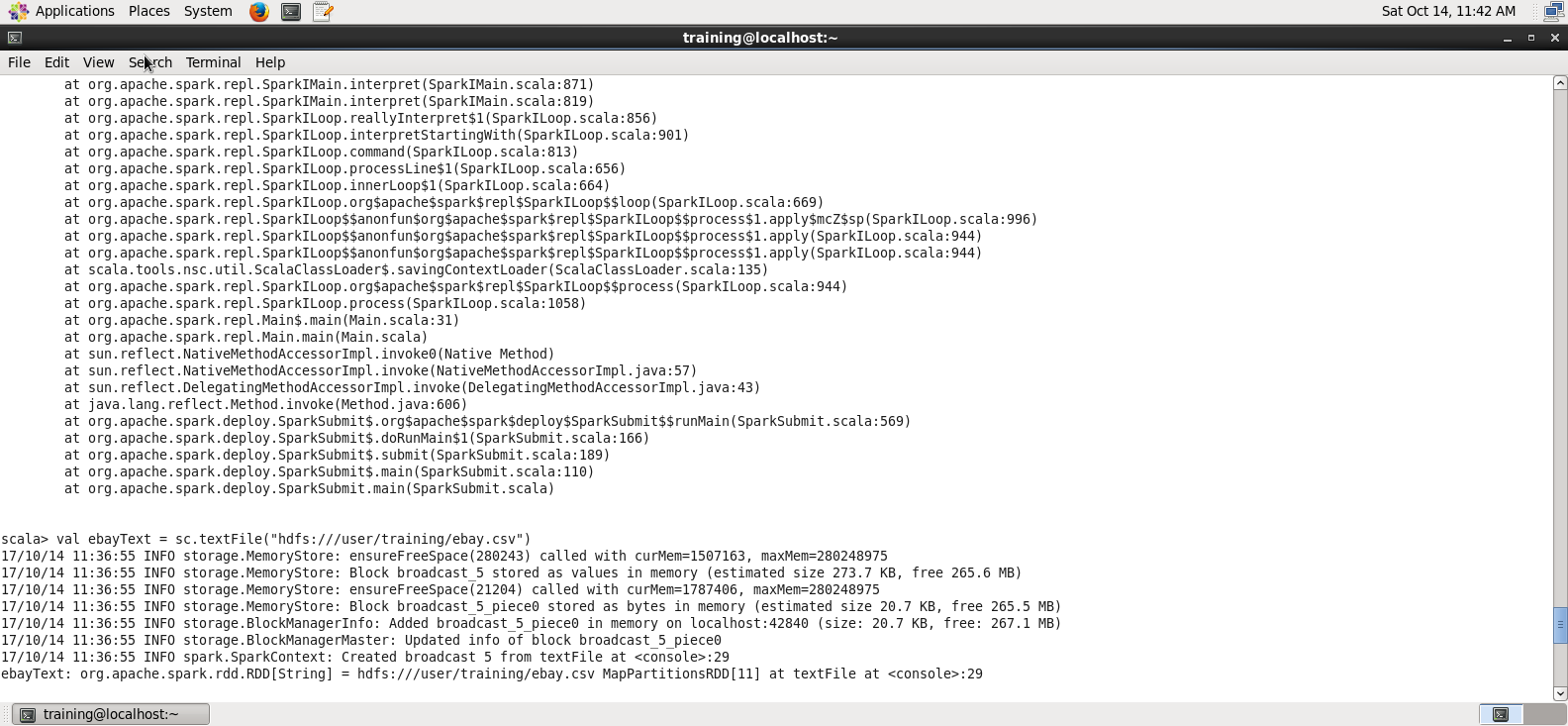
import org.apache.spark.sql.\_



5

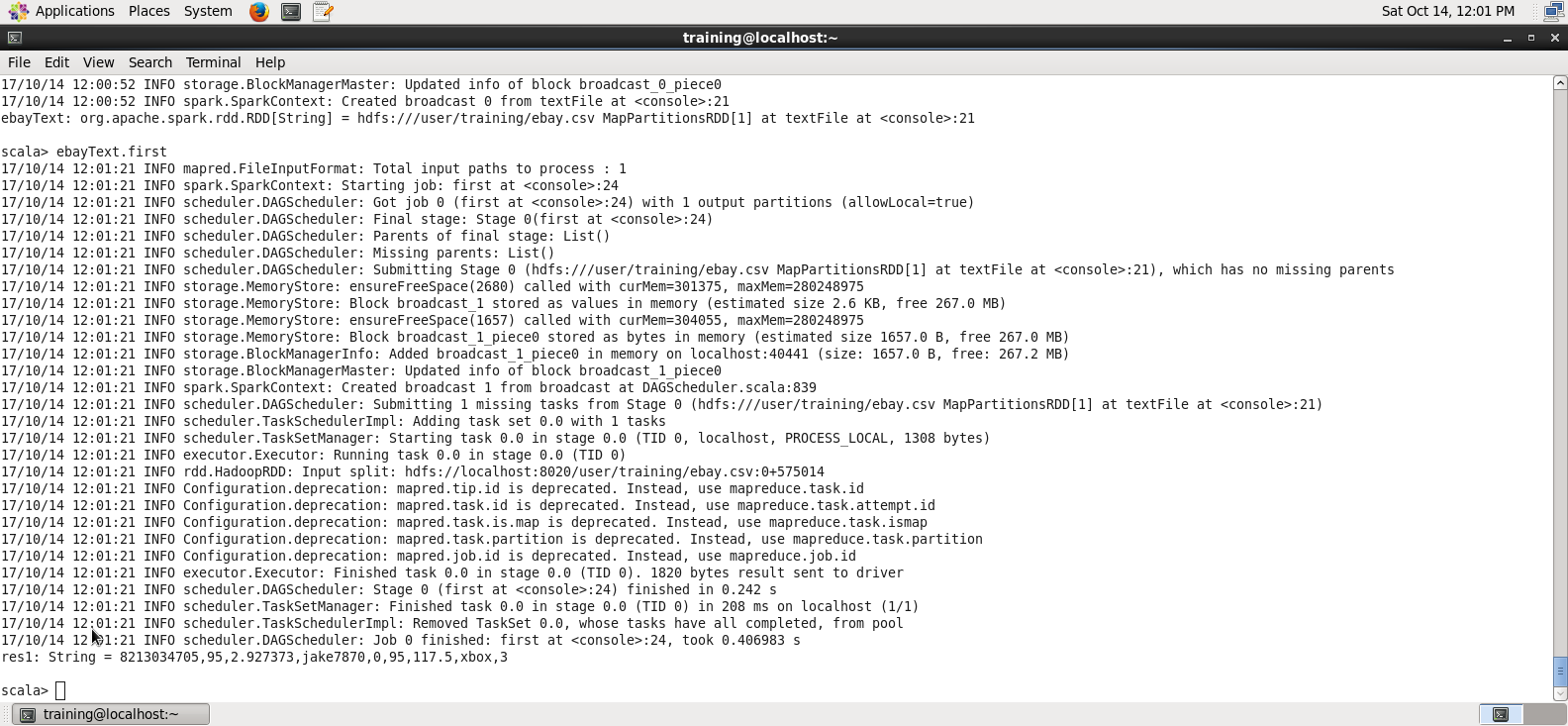
####wrong val mydata = sc.textFile("file:/user/training/ebay.csv")

val ebayText = sc.textFile("hdfs:///user/training/ebay.csv")



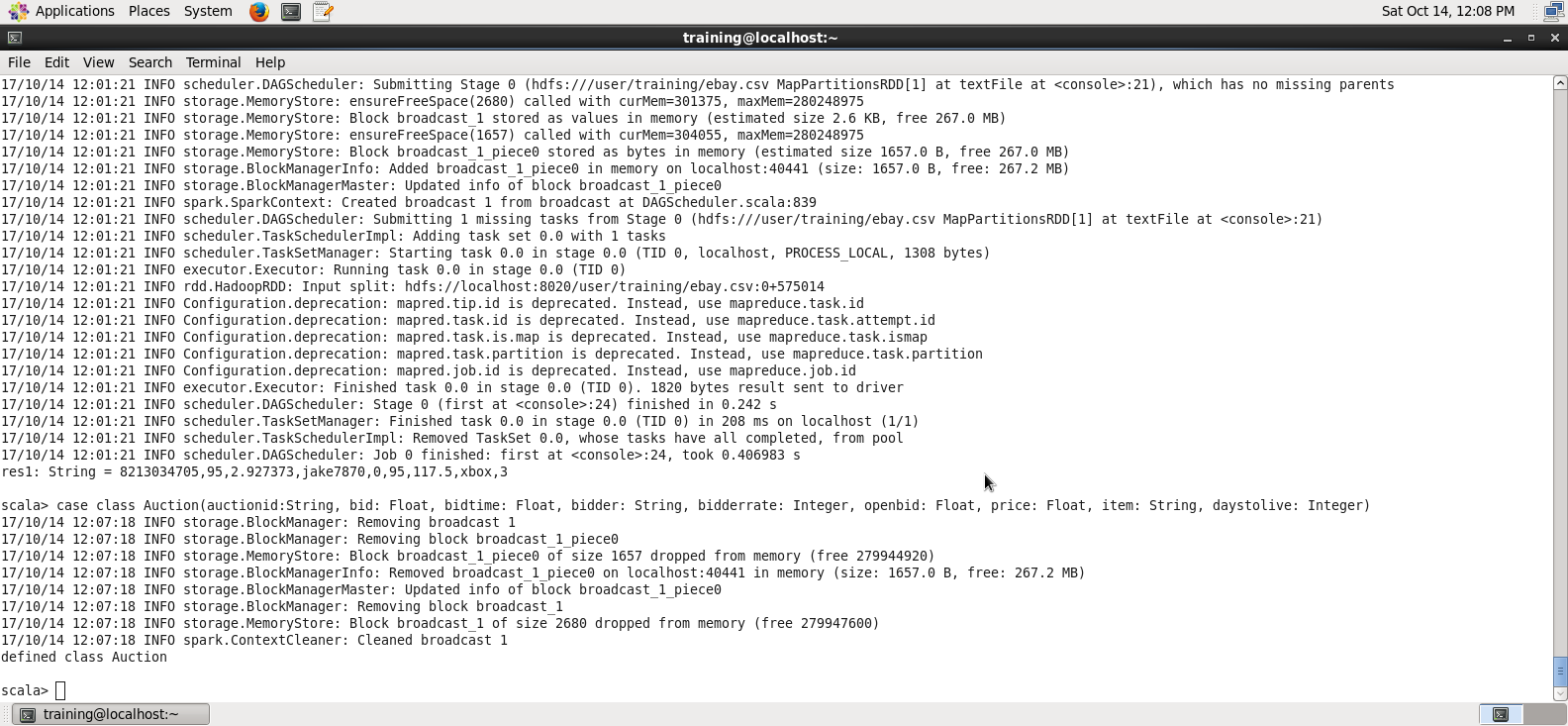
6

ebayText.first



7

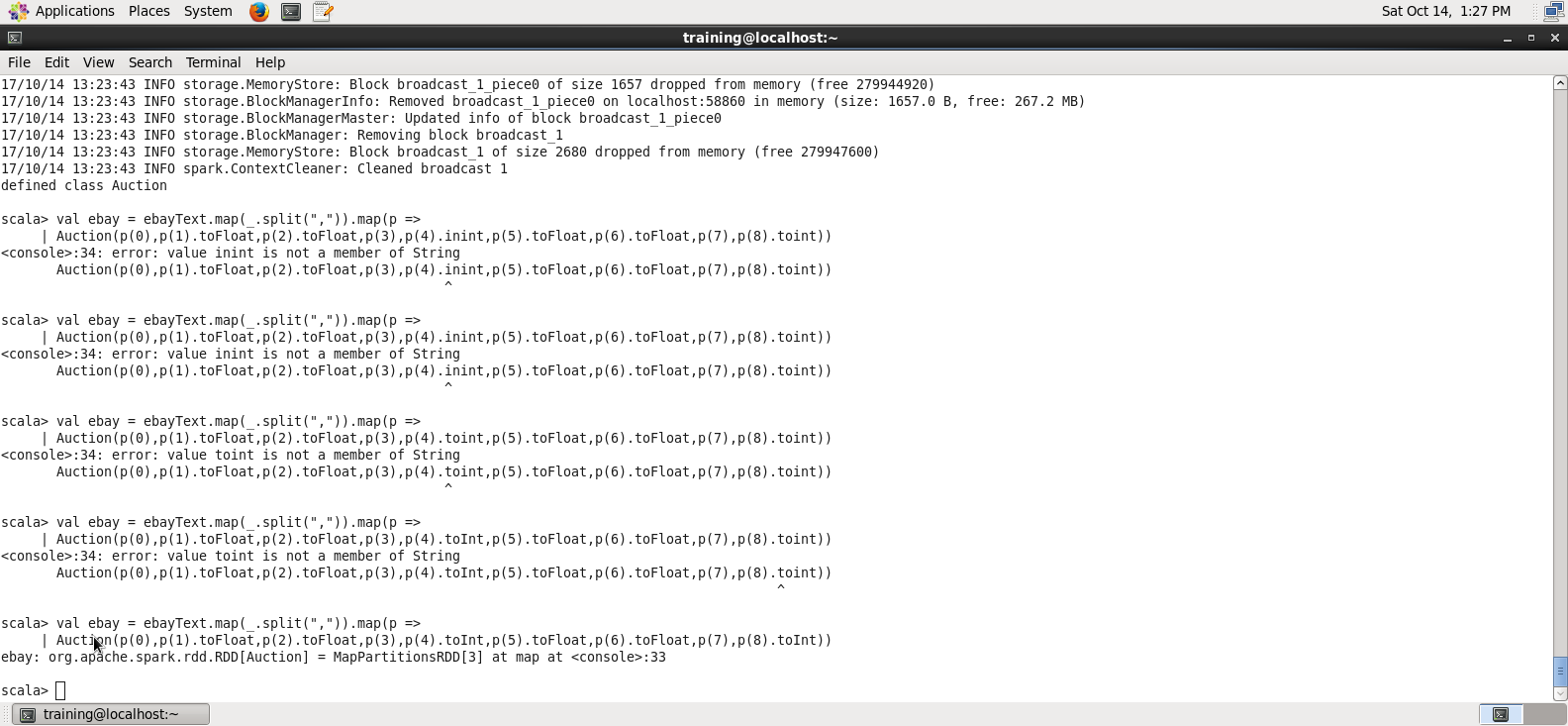
case class Auction(auctionid:String, bid: Float, bidtime: Float, bidder: String, bidderrate: Integer, openbid: Float, price: Float, item: String, daystolive: Integer)



8

val ebay = ebayText.map(\_.split(",")).map(p =>

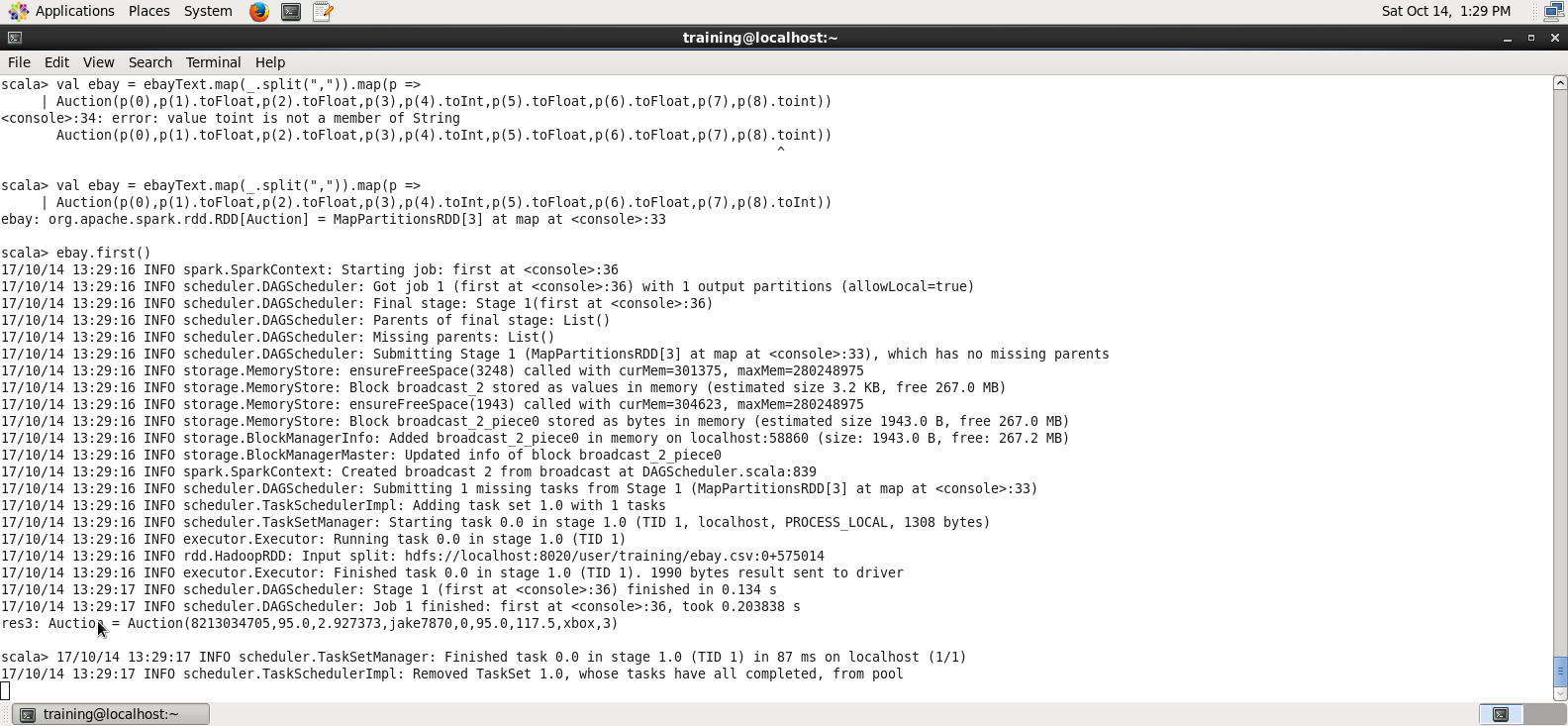
Auction(p(0),p(1).toFloat,p(2).toFloat,p(3),p(4).toInt,p(5).toFloat,p(6).toFloat,p(7),p(8).toInt))



9

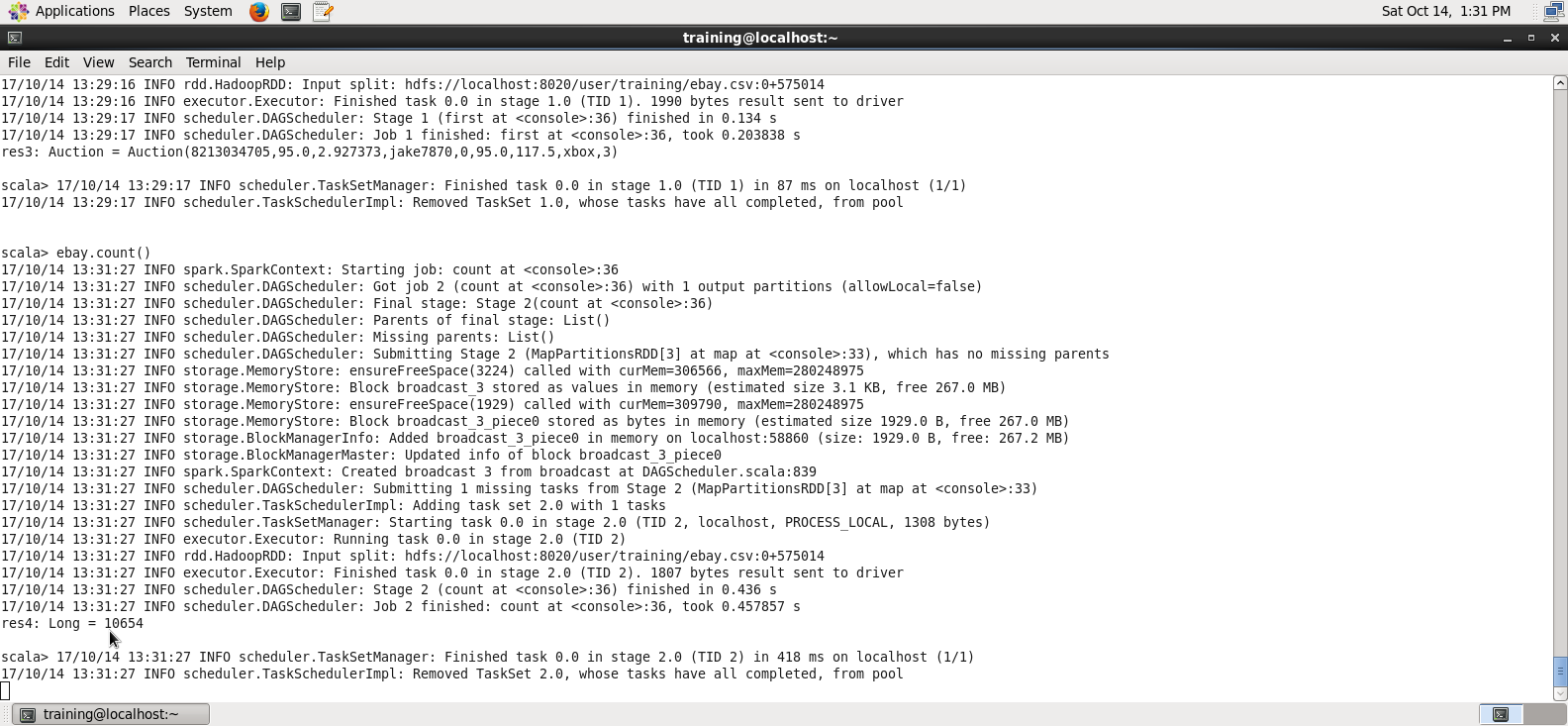
###ebayText.first

ebay.first()



10

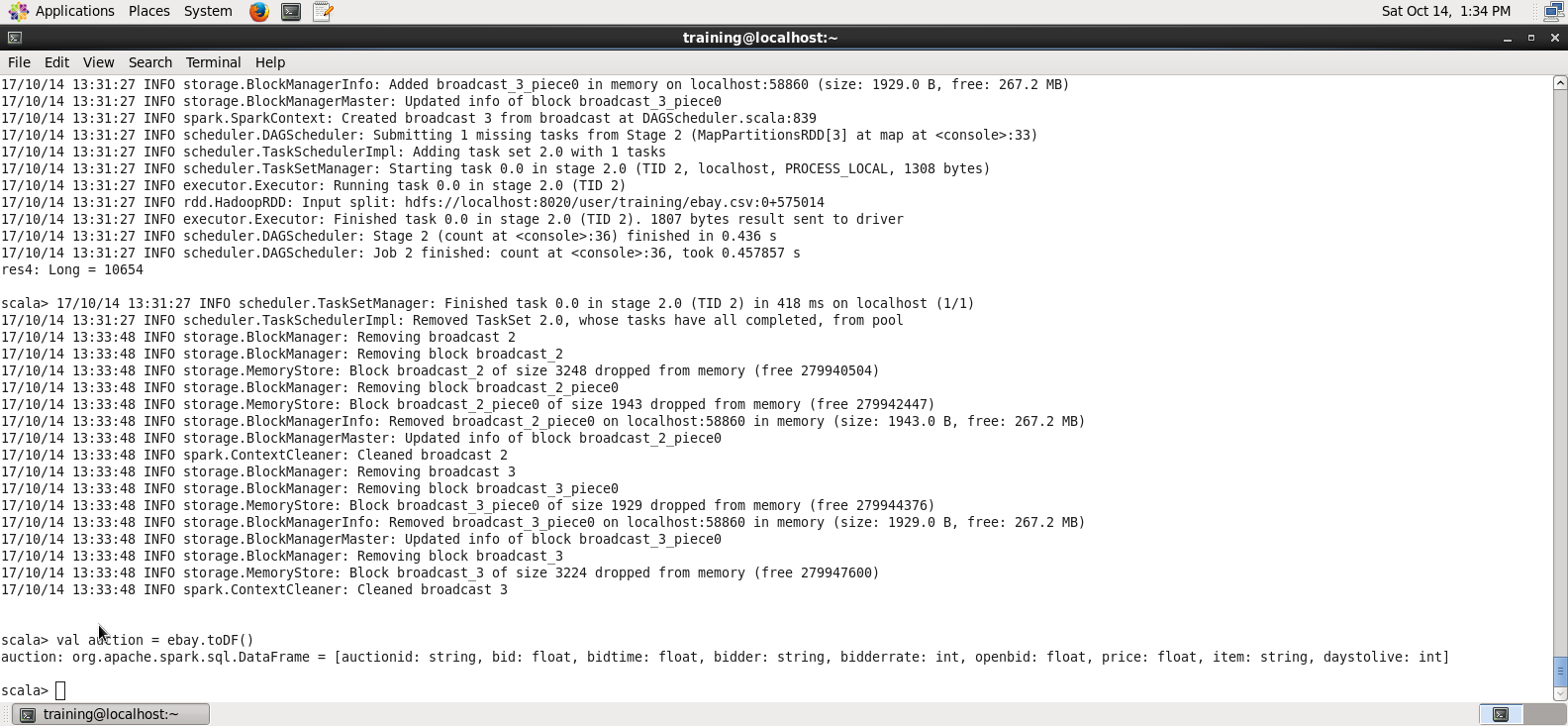
ebay.count()



11

import sqlContext.implicits.\_

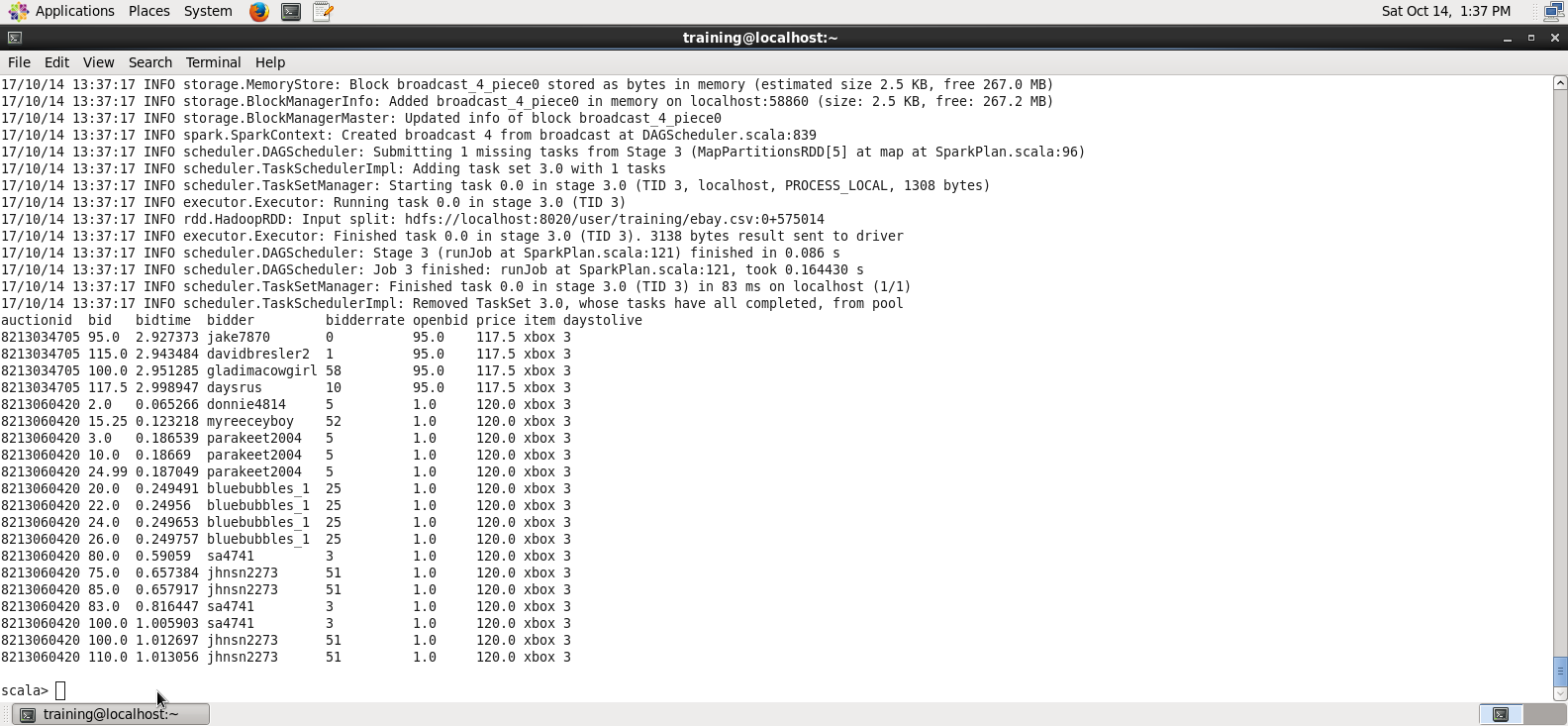
val auction = ebay.toDF()



**PartB AS2**

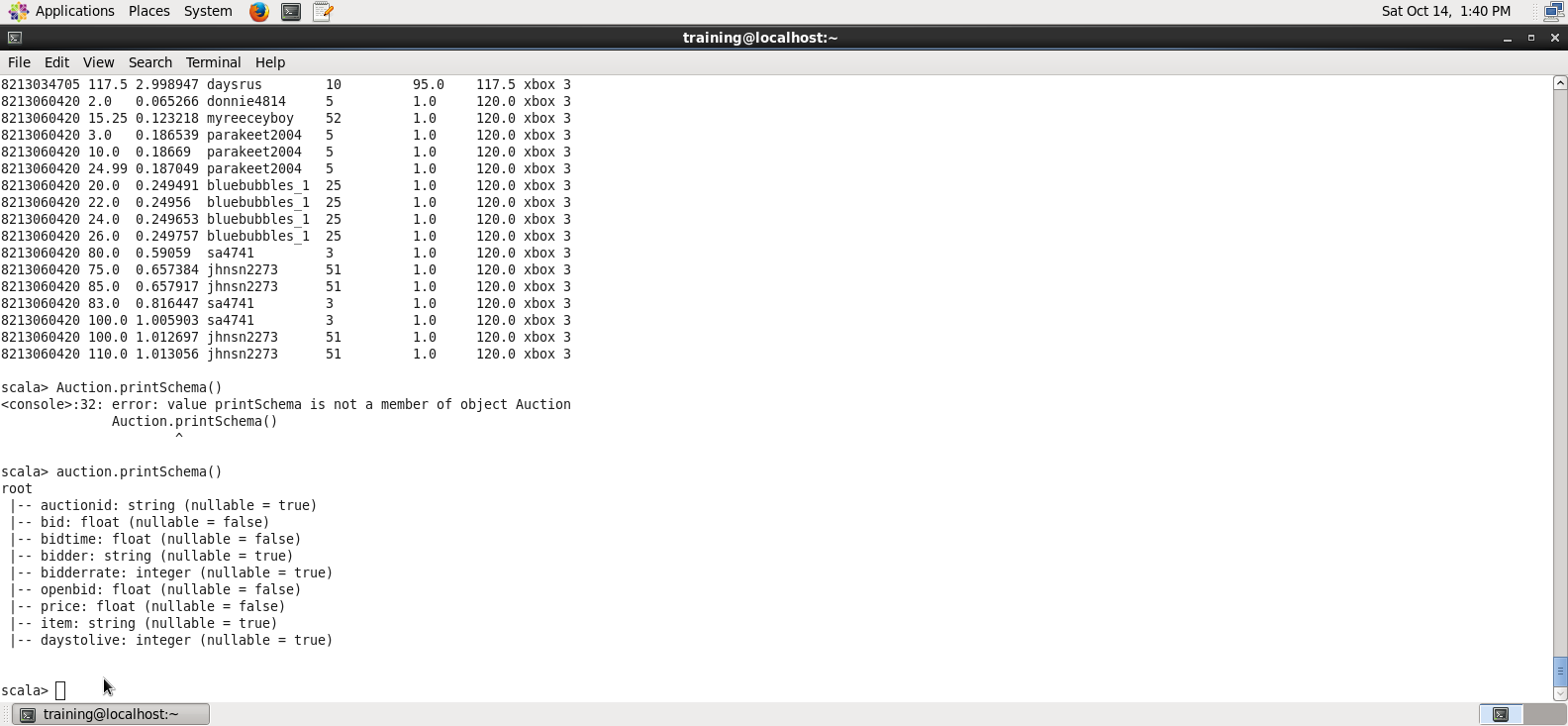
**12**

auction.show()



**13**

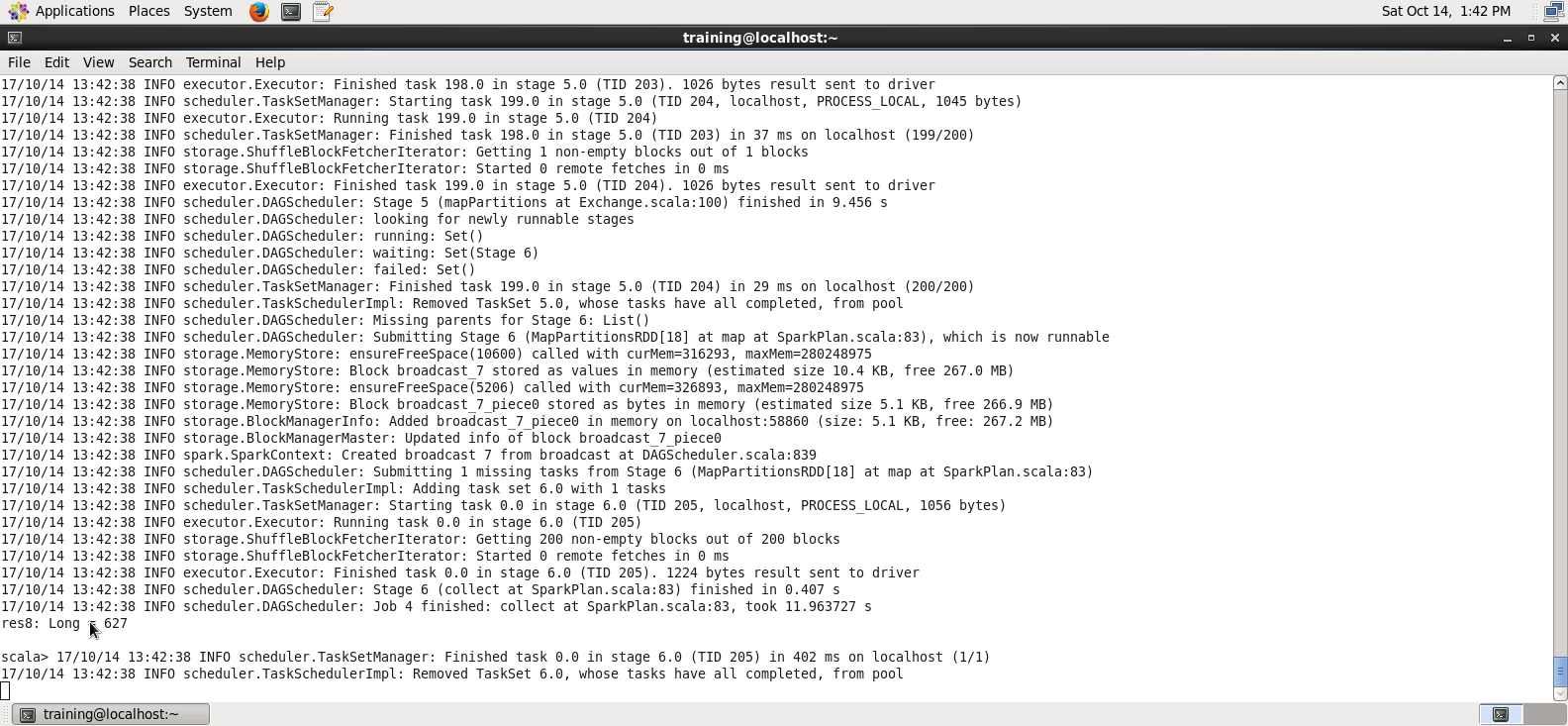
auction.printSchema()



**14**

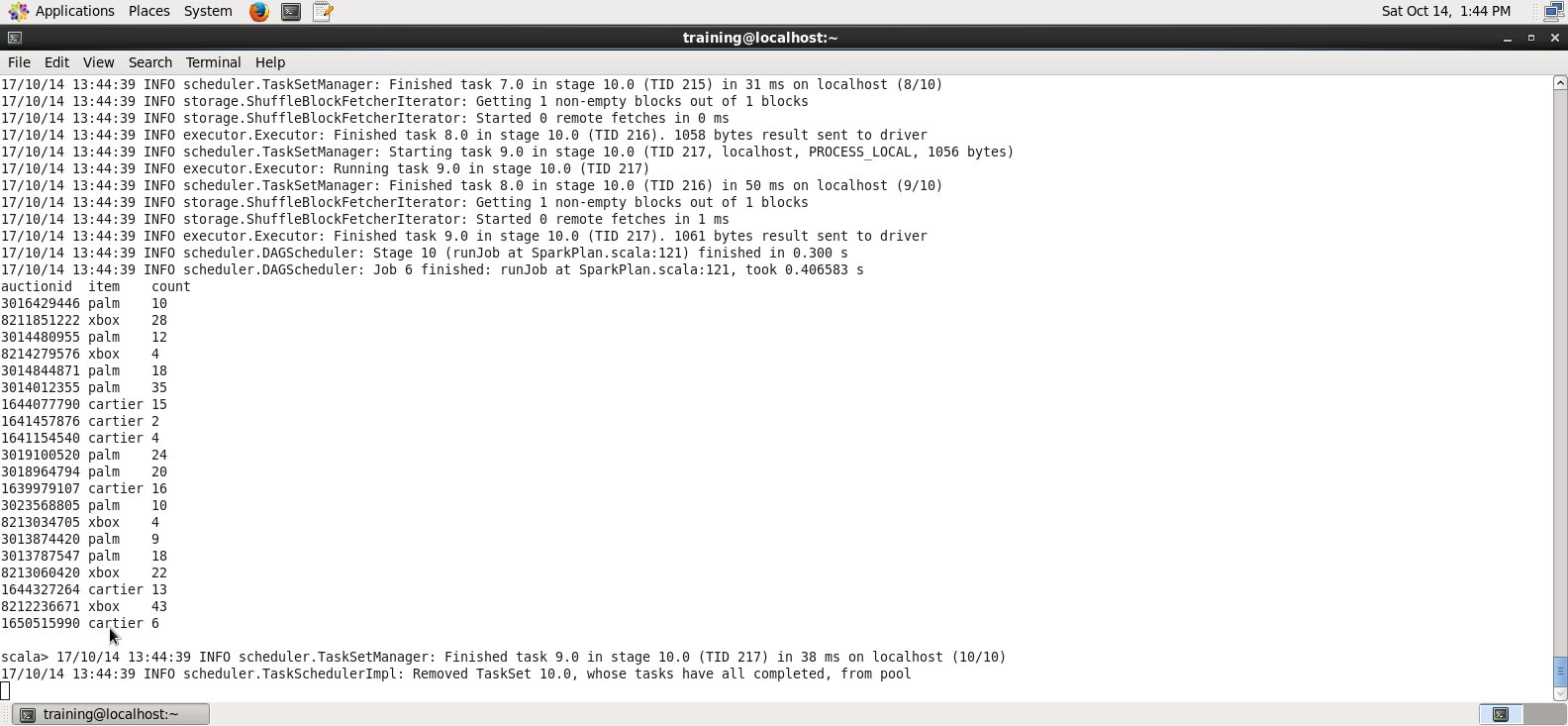
auction.select("auctionid").distinct.count

answer 627



**15**

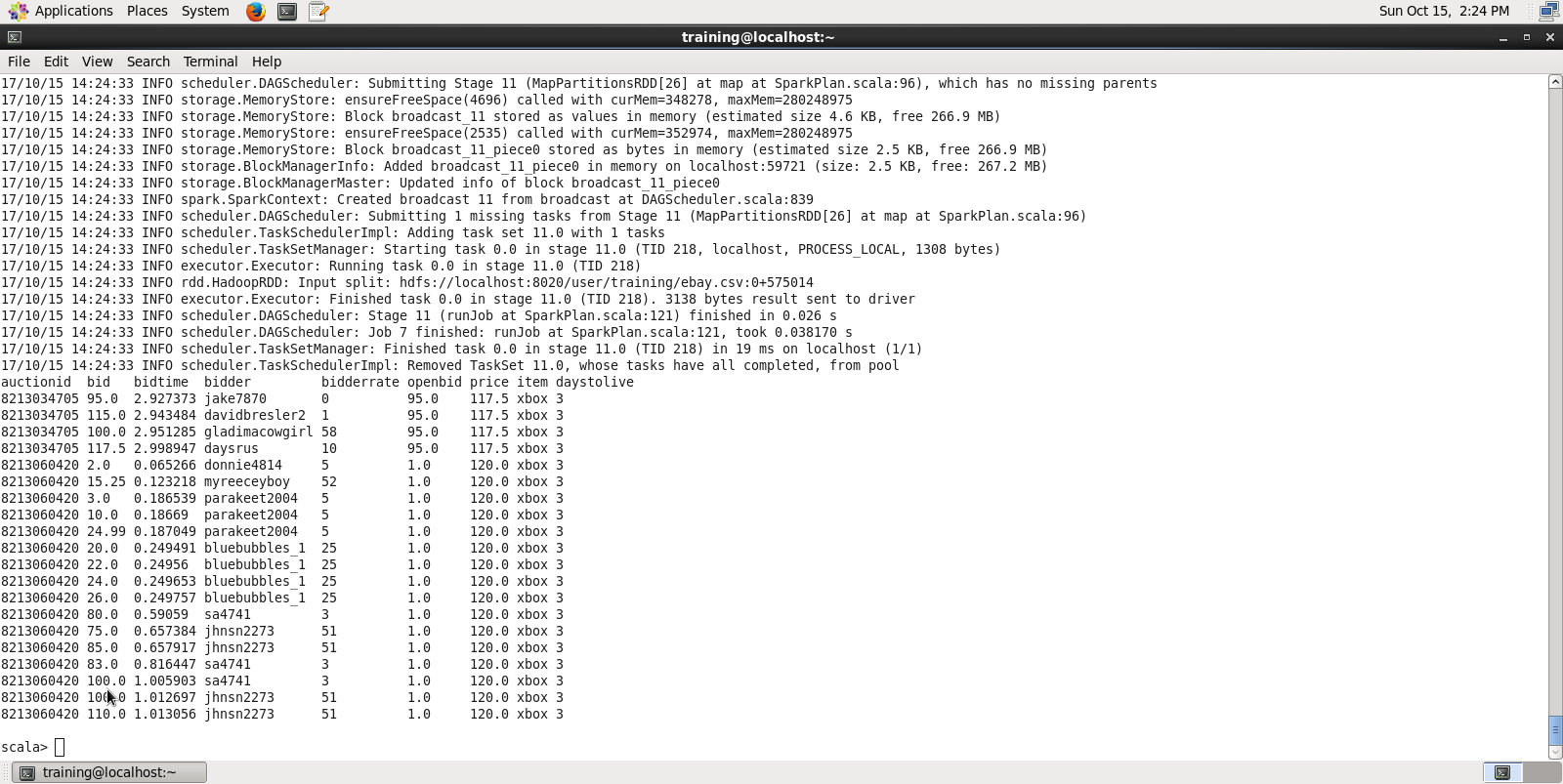
auction.groupBy("auctionid", "item").count.show



**16**

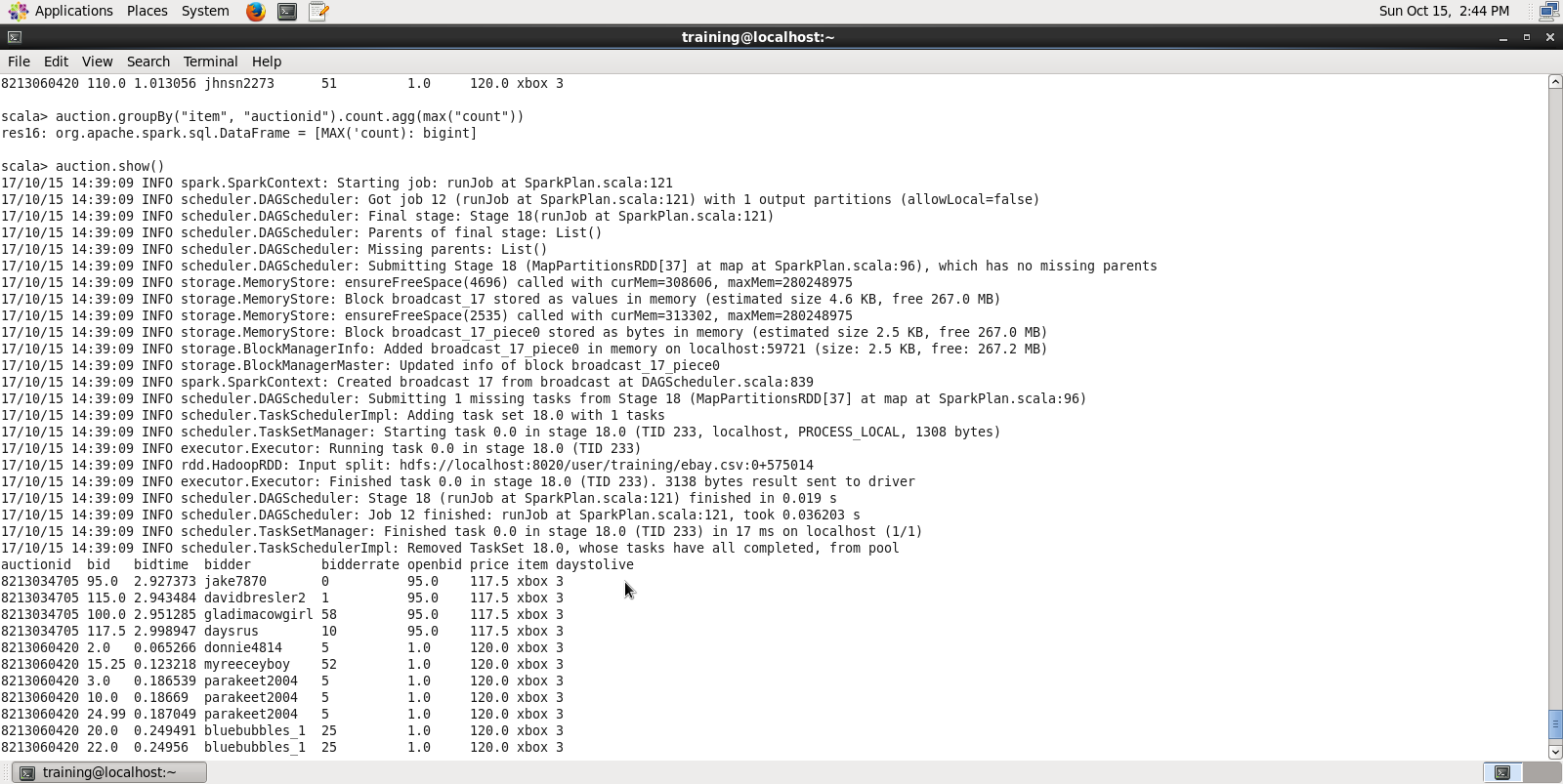
auctionb.groupBy("item", "auctionid").count.agg(avg("count"))

auction.show()



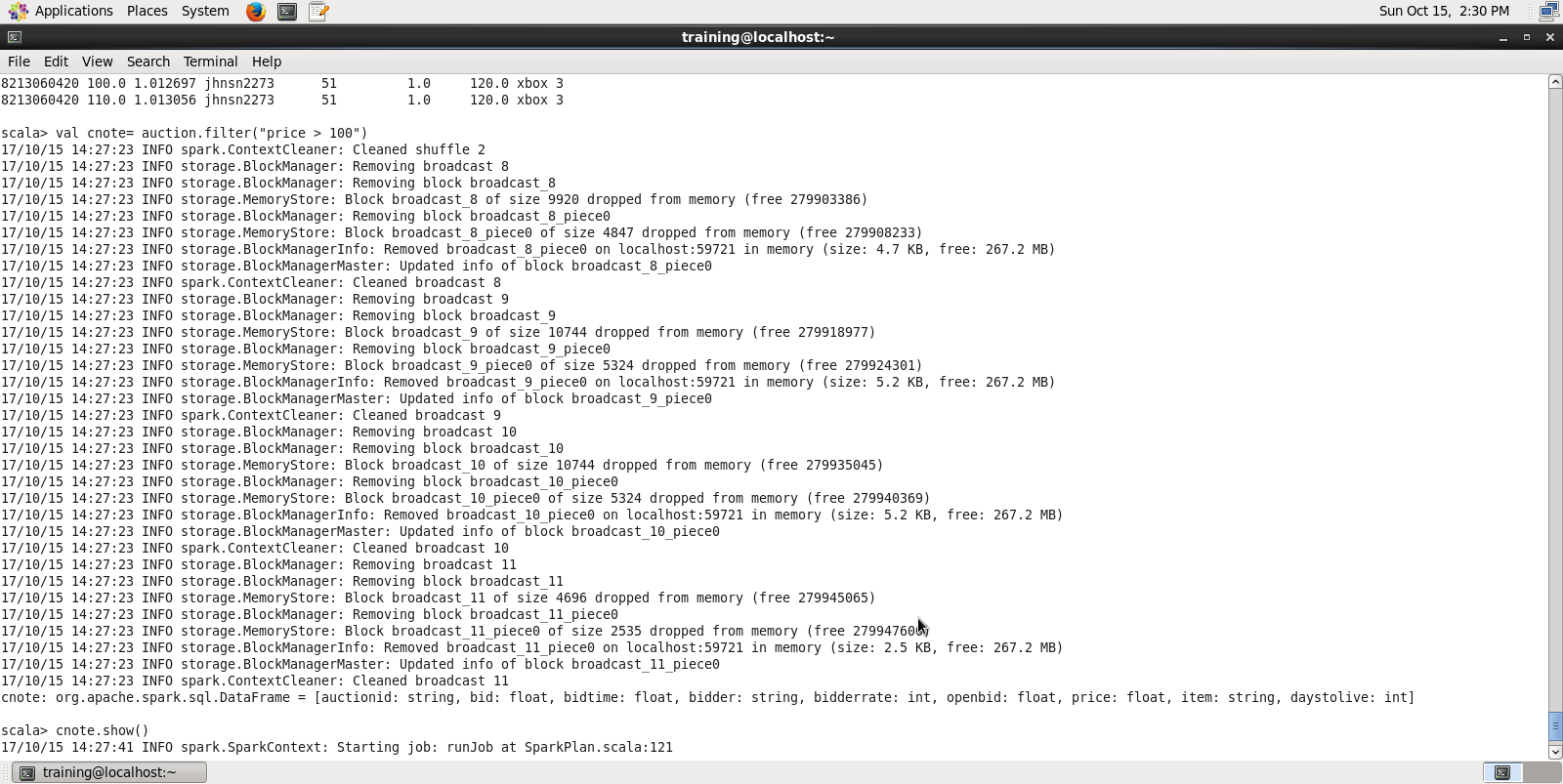
auction.groupBy("item", "auctionid").count.agg(max("count"))

auction.show()



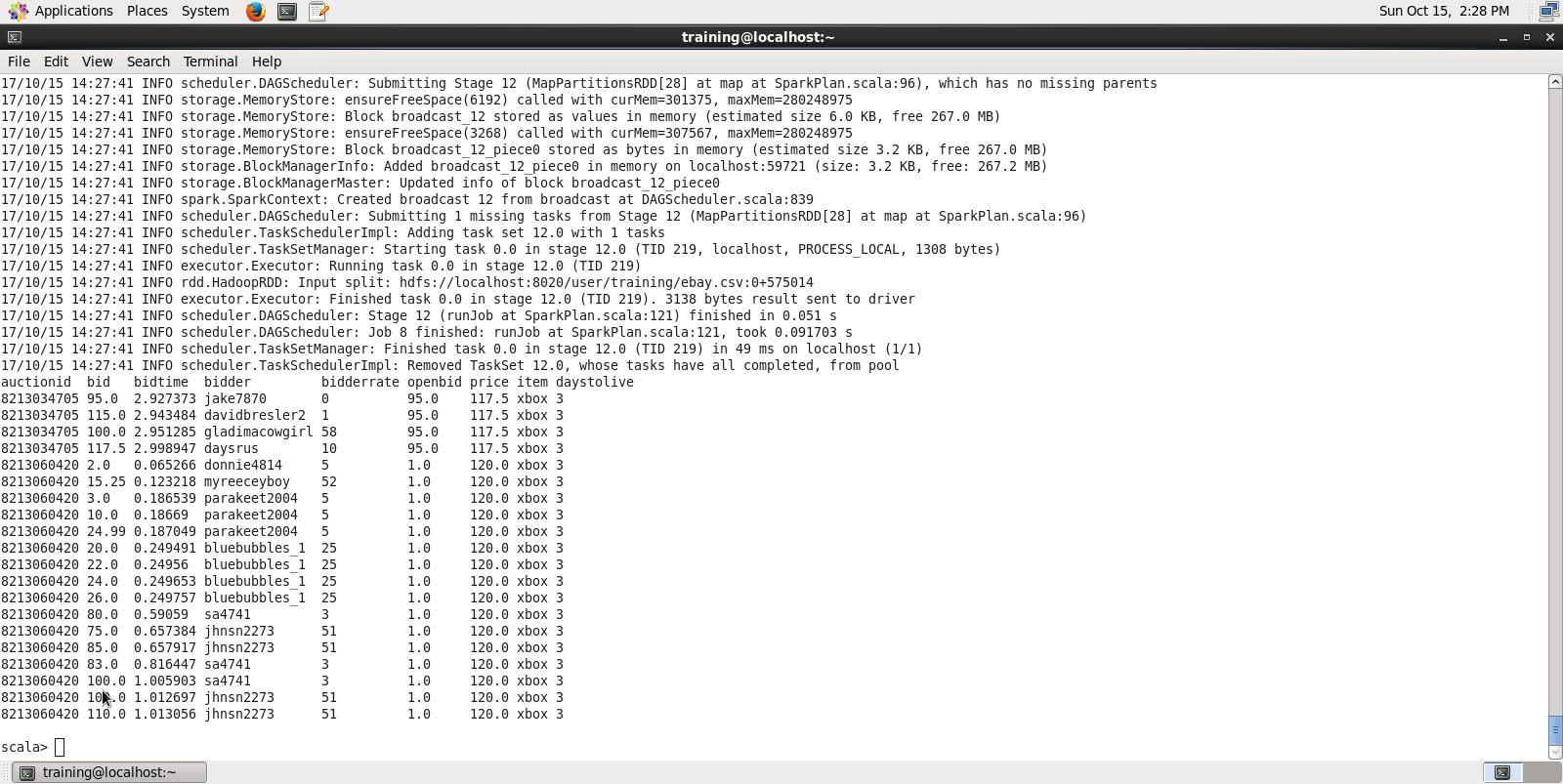
**17**

val cnote= auction.filter("price > 100")



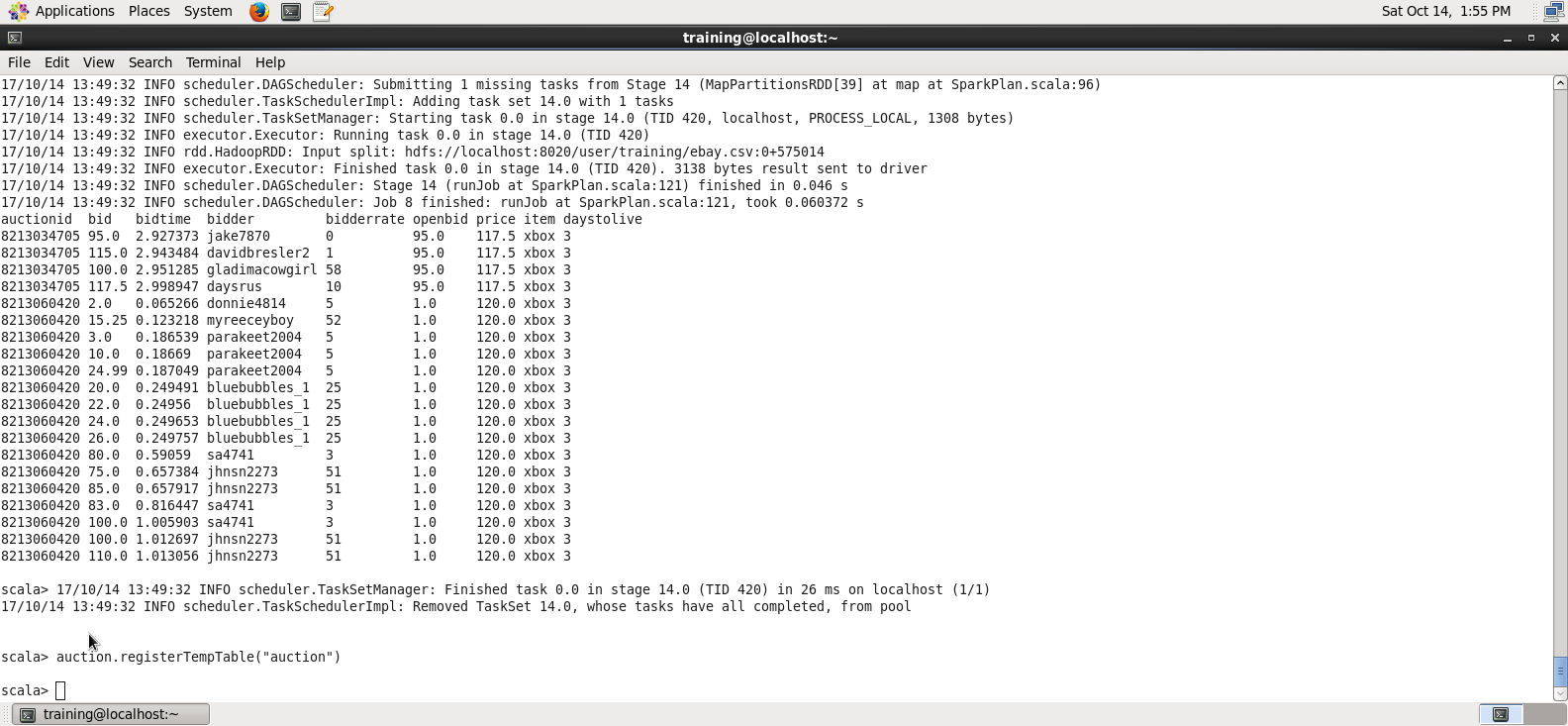
**18**

cnote.show()



**19**

auction.registerTempTable("auction")

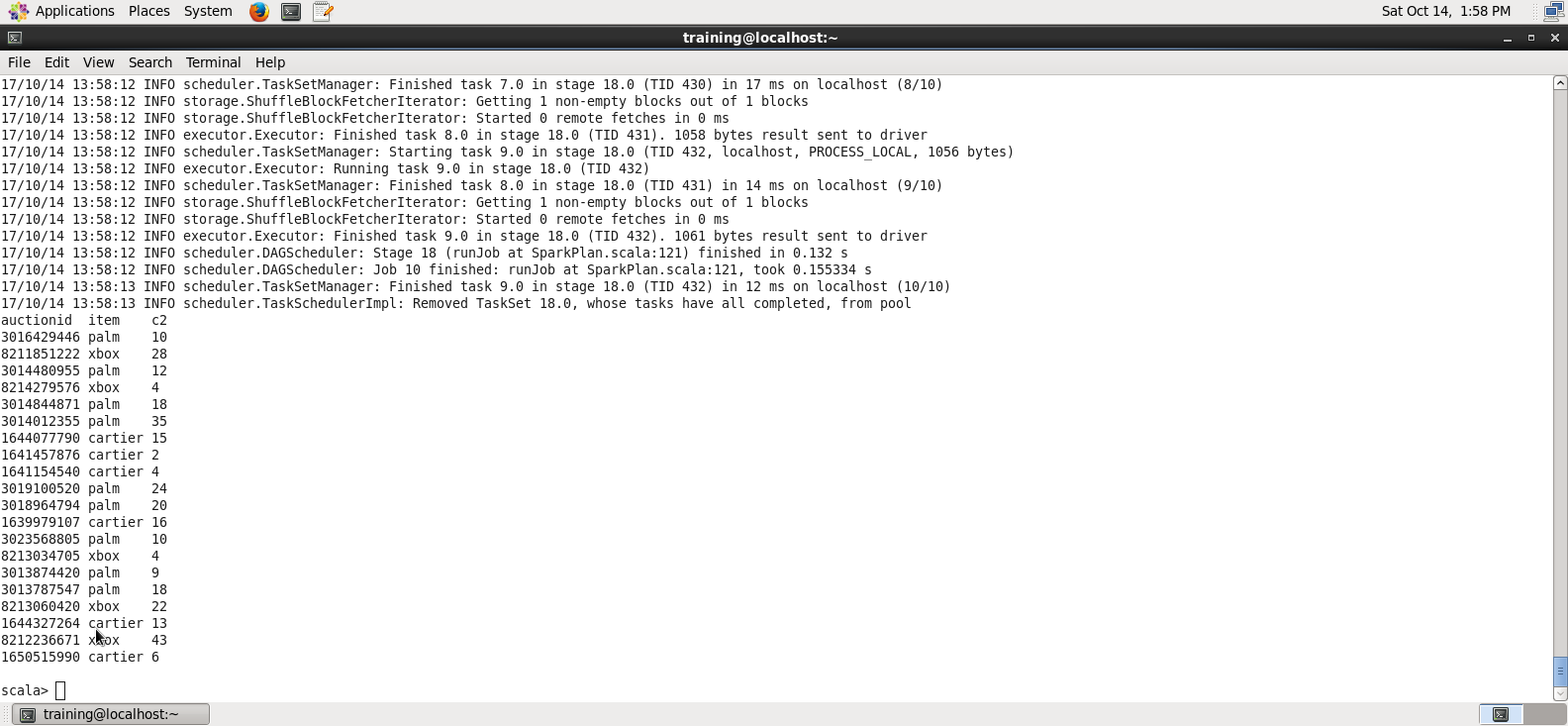


**20**

val answer =sqlContext.sql("SELECT auctionid, item, count(bid) FROM auction GROUP BY auctionid, item")

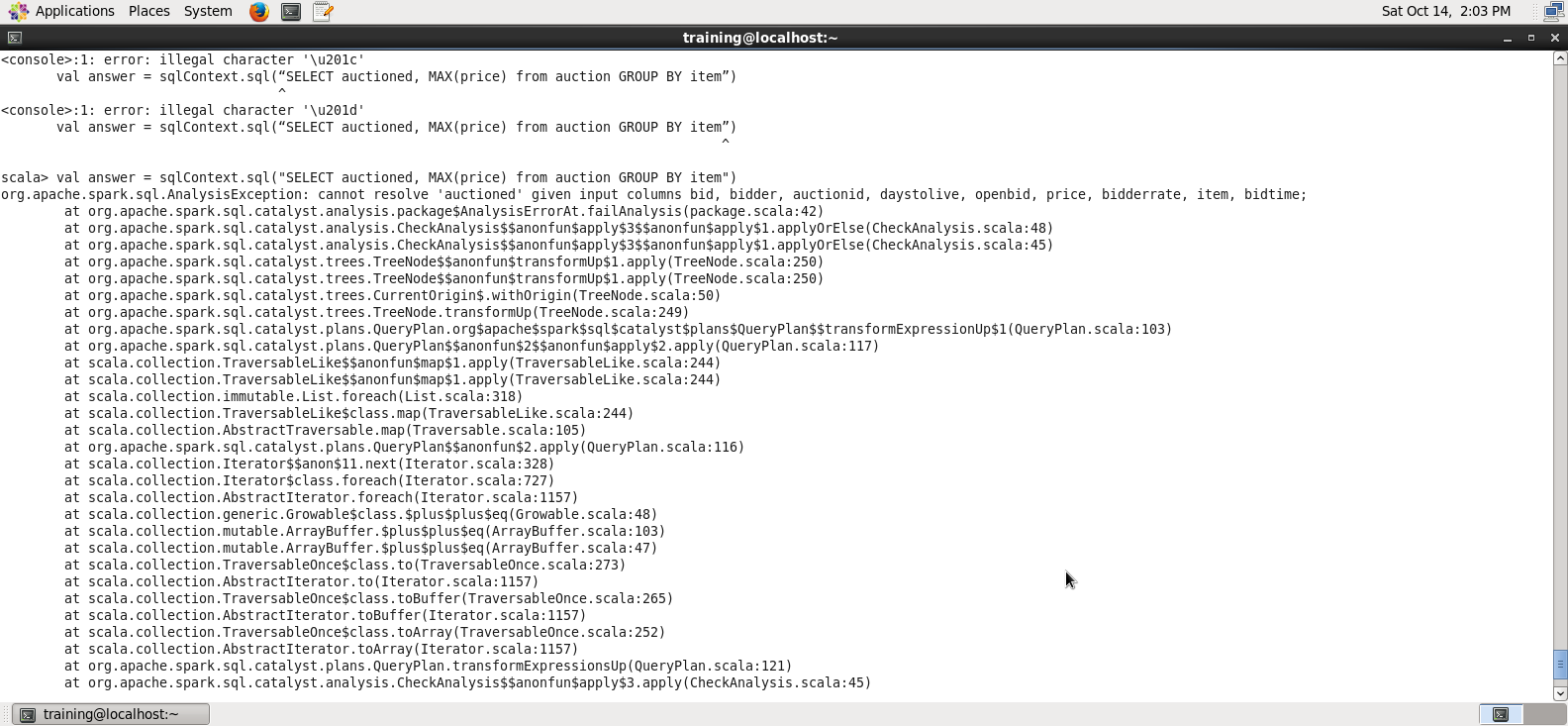
**21**

answer.show()



**22**

**val answer = sqlContext.sql(**"**SELECT auctioned, MAX(price) from auction GROUP BY item**"**)**



**answer.show()**

