

# Máster en Big Data

---

## Tecnologías de Almacenamiento

### 8. Hands-On: Desarrollo Apache Spark SQL

Presentado por: José David Angulo y Albert Ripoll

## Índice

1. Introducción .....	3
2. Entorno .....	3
3. Creación del DataFrame .....	3
4. Inspección de datos .....	7

# 1. Introducción

El objetivo de este Hands-On es el de familiarizarse con una de las librerías más populares del framework de Spark como es SparkSQL

## 2. Entorno

Para la realización de los ejercicios se va a utilizar *spark-shell* en scala ya que nos proporciona un entorno muy dinámico para la introducción de funciones y nos permite recibir una respuesta inmediata.

Para ello, utilizaremos la máquina virtual desplegada en Hands-On anteriores llamada Developer\_Hadoop y ejecutaremos el Spark Shell ubicado en `/home/training/spark-1.3.1/bin`

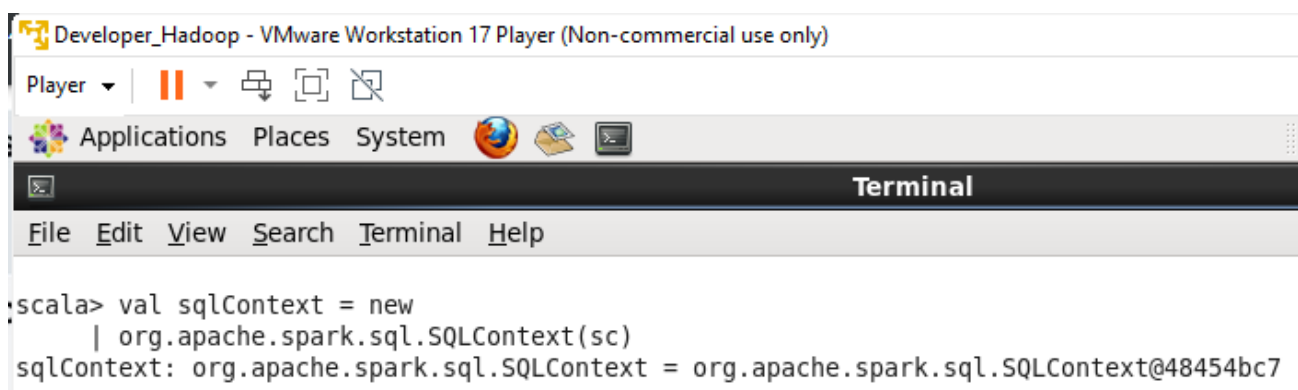
El dataset que utilizaremos se llama `auctiondata.csv` y está ubicado en `/home/training/training_materials/developer/data/auction.csv`

## 3. Creación del DataFrame

a) Crear el SQL Context

```
val sqlContext = new
```

```
org.apache.spark.sql.SQLContext(sc)
```



The screenshot shows a terminal window titled "Developer\_Hadoop - VMware Workstation 17 Player (Non-commercial use only)". The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The command prompt is "scala>". The user enters the following code:

```
scala> val sqlContext = new  
      | org.apache.spark.sql.SQLContext(sc)  
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@48454bc7
```

b) Realiza el import correspondiente para convertir un RDD en un DataFrame

```
import sqlContext.implicits._
```

```
scala> import sqlContext.implicits._
import sqlContext.implicits._
```

c) Definición del esquema usando una *case class*

**case class**

**Auction(auctionid:String,bid:Float,bidtime:Float,bidder:String,bidderrate:Int,openbid:Float,finprice:Float,itemtype:String,dtl:Int)**

```
scala> case class Auction(auctionid:String,bid:Float,bidtime:Float,bidder:String,bidderrate:Int,openbid:Float,finprice:Float,
itemtype:String,dtl:Int)
defined class Auction
```

d) Crea un RDD llamado *inputRDD* para cargar el *auctiondata.csv*. Asegurate de dividir el archivo de entrada con el separador “,”

**val inputRDD = sc.textFile**

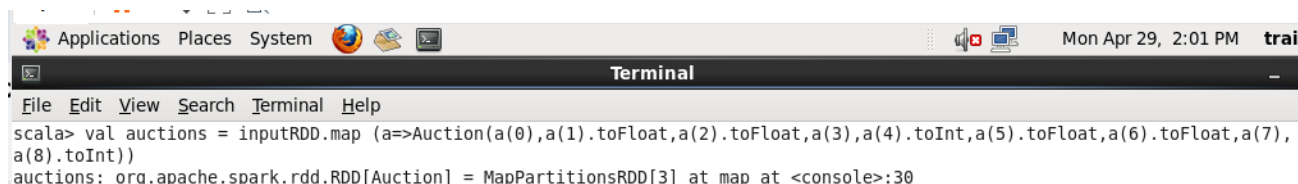
**(""/home/training/training\_materials/developer/data/auctiondata.csv").map(\_.\_split(","))**

```
scala> val inputRDD = sc.textFile ("/home/training/training_materials/developer/data/auctiondata.csv").map(_._split(","))
24/04/29 13:49:49 INFO MemoryStore: ensureFreeSpace(82882) called with curMem=0, maxMem=280248975
24/04/29 13:49:49 INFO MemoryStore: Block broadcast_0 stored as values in memory (estimated size 80.9 KB, free 267.2 MB)
24/04/29 13:49:49 INFO MemoryStore: ensureFreeSpace(10636) called with curMem=82882, maxMem=280248975
24/04/29 13:49:49 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 10.4 KB, free 267.2 MB)
24/04/29 13:49:49 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on localhost:34040 (size: 10.4 KB, free: 267.3 MB)
24/04/29 13:49:49 INFO BlockManagerMaster: Updated info of block broadcast_0_piece0
24/04/29 13:49:49 INFO SparkContext: Created broadcast 0 from textFile at <console>:26
inputRDD: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[2] at map at <console>:26
```

e) Ahora Mapea el *inputRDD* a *case class*

**val auctions = inputRDD.map**

**(a=>Auction(a(0),a(1).toFloat,a(2).toFloat,a(3),a(4).toInt,a(5).toFloat,a(6).toFloat,a(7),a(8).toInt))**

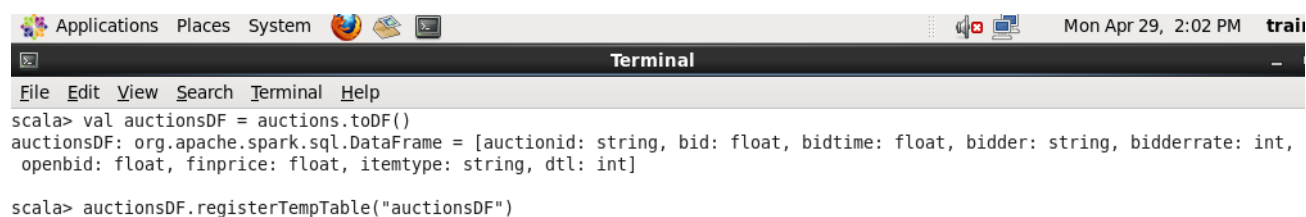


```
scala> val auctions = inputRDD.map (a=>Auction(a(0),a(1).toFloat,a(2).toFloat,a(3),a(4).toInt,a(5).toFloat,a(6).toFloat,a(7),a(8).toInt))
auctions: org.apache.spark.rdd.RDD[Auction] = MapPartitionsRDD[3] at map at <console>:30
```

- f) Convierte el RDD resultante del punto anterior en un *DataFrame* llamado *auctionsDF* y regístralo como una tabla (Registrandolo de esta manera, podremos ejecutar consultas SQL usando los métodos proporcionados por *sqlContext*)

```
val auctionsDF = auctions.toDF()
```

```
auctionsDF.registerTempTable("auctionsDF")
```



```
scala> val auctionsDF = auctions.toDF()
auctionsDF: org.apache.spark.sql.DataFrame = [auctionid: string, bid: float, bidtime: float, bidder: string, bidderrate: int, openbid: float, finprice: float, itemtype: string, dtl: int]

scala> auctionsDF.registerTempTable("auctionsDF")
```

- g) ¿Que acción puedes hacer para comprobar los datos en el *DataFrame*?

```
auctionsDF.show()
```

```

Applications Places System Mon Apr 29, 2:03 PM training
Browse and run installed applications Terminal
File Edit View Search Terminal Help
scala> auctionsDF.show()
24/04/29 13:52:15 INFO FileInputFormat: Total input paths to process : 1
24/04/29 13:52:15 INFO SparkContext: Starting job: runJob at SparkPlan.scala:122
24/04/29 13:52:15 INFO DAGScheduler: Got job 0 (runJob at SparkPlan.scala:122) with 1 output partitions (allowLocal=false)
24/04/29 13:52:15 INFO DAGScheduler: Final stage: Stage 0(runJob at SparkPlan.scala:122)
24/04/29 13:52:15 INFO DAGScheduler: Parents of final stage: List()
24/04/29 13:52:15 INFO DAGScheduler: Missing parents: List()
24/04/29 13:52:15 INFO DAGScheduler: Submitting Stage 0 (MapPartitionsRDD[5] at map at SparkPlan.scala:97), which has no missing parents
24/04/29 13:52:15 INFO MemoryStore: ensureFreeSpace(4680) called with curMem=93518, maxMem=280248975
24/04/29 13:52:15 INFO MemoryStore: Block broadcast 1 stored as values in memory (estimated size 4.6 KB, free 267.2 MB)
24/04/29 13:52:15 INFO MemoryStore: ensureFreeSpace(3105) called with curMem=98198, maxMem=280248975
24/04/29 13:52:15 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 3.0 KB, free 267.2 MB)
24/04/29 13:52:15 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:34040 (size: 3.0 KB, free: 267.3 MB)
24/04/29 13:52:15 INFO BlockManagerMaster: Updated info of block broadcast_1_piece0
24/04/29 13:52:15 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:839
24/04/29 13:52:15 INFO DAGScheduler: Submitting 1 missing tasks from Stage 0 (MapPartitionsRDD[5] at map at SparkPlan.scala:97)
24/04/29 13:52:15 INFO TaskSchedulerImpl: Adding task set 0.0 with 1 tasks
24/04/29 13:52:15 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, PROCESS_LOCAL, 1334 bytes)
24/04/29 13:52:15 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
24/04/29 13:52:15 INFO HadoopRDD: Input split: file:/home/training/training materials/developer/data/auctiondata.csv:0+575014
24/04/29 13:52:15 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 4072 bytes result sent to driver
24/04/29 13:52:15 INFO DAGScheduler: Stage 0 (runJob at SparkPlan.scala:122) finished in 0.113 s
24/04/29 13:52:15 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 108 ms on localhost (1/1)
24/04/29 13:52:15 INFO DAGScheduler: Job 0 finished: runJob at SparkPlan.scala:122, took 0.207962 s
24/04/29 13:52:15 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
auctionid bid bidtime bidder bidderrate openbid finprice itemtype dtl
8213034705 95.0 2.927373 jake7870 0 95.0 117.5 xbox 3
8213034705 115.0 2.943484 davidbresler2 1 95.0 117.5 xbox 3
8213034705 100.0 2.951285 gladimacowgirl 58 95.0 117.5 xbox 3
8213034705 117.5 2.998947 daysrus 10 95.0 117.5 xbox 3
8213060420 2.0 0.065266 donnie4814 5 1.0 120.0 xbox 3
8213060420 15.25 0.123218 myreeceyboy 52 1.0 120.0 xbox 3
8213060420 3.0 0.186539 parakeet2004 5 1.0 120.0 xbox 3
8213060420 10.0 0.18669 parakeet2004 5 1.0 120.0 xbox 3
8213060420 24.99 0.187049 parakeet2004 5 1.0 120.0 xbox 3
8213060420 20.0 0.249491 bluebubbles_1 25 1.0 120.0 xbox 3
8213060420 22.0 0.24956 bluebubbles_1 25 1.0 120.0 xbox 3

```

h) ¿Que función del DataFrame puedes utilizar para ver el esquema del mismo?

**auctionsDF.printSchema()**

```

scala> auctionsDF.printSchema()
root
|-- auctionid: string (nullable = true)
|-- bid: float (nullable = false)
|-- bidtime: float (nullable = false)
|-- bidder: string (nullable = true)
|-- bidderrate: integer (nullable = false)
|-- openbid: float (nullable = false)
|-- finprice: float (nullable = false)
|-- itemtype: string (nullable = true)
|-- dtl: integer (nullable = false)

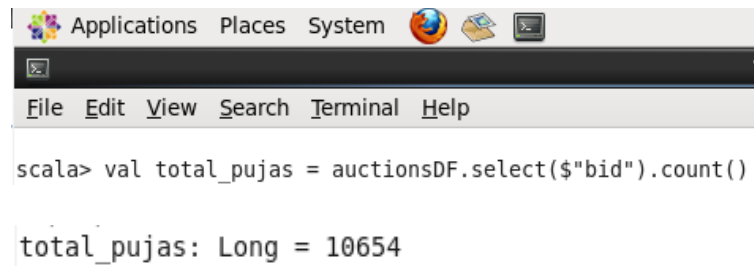
```

## 4. Inspección de datos

a) ¿Cuál es el número total de pujas?

```
val total_pujas = auctionsDF.select($"bid").count()
```

Respuesta: 10654

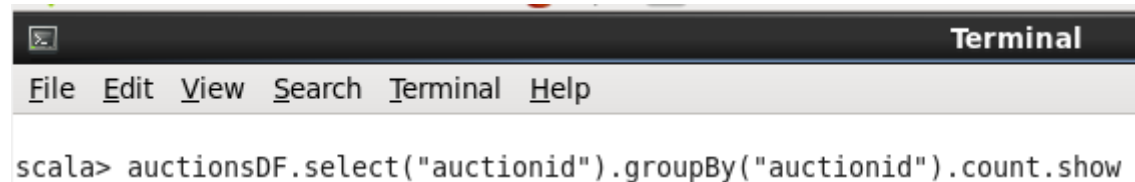


```
Applications Places System [Globe] [Folder] [Terminal Icon]  
File Edit View Search Terminal Help  
scala> val total_pujas = auctionsDF.select($"bid").count()  
  
total_pujas: Long = 10654
```

b) ¿Cuál es el número total de subastas distintas?

Hay 20 subastas distintas. El siguiente código nos muestra cuantas pujas hay en cada subasta y se pueden contar 20 líneas.

```
auctionsDF.select("auctionid").groupBy("auctionid").count.show
```



```
Terminal  
File Edit View Search Terminal Help  
scala> auctionsDF.select("auctionid").groupBy("auctionid").count.show
```

```

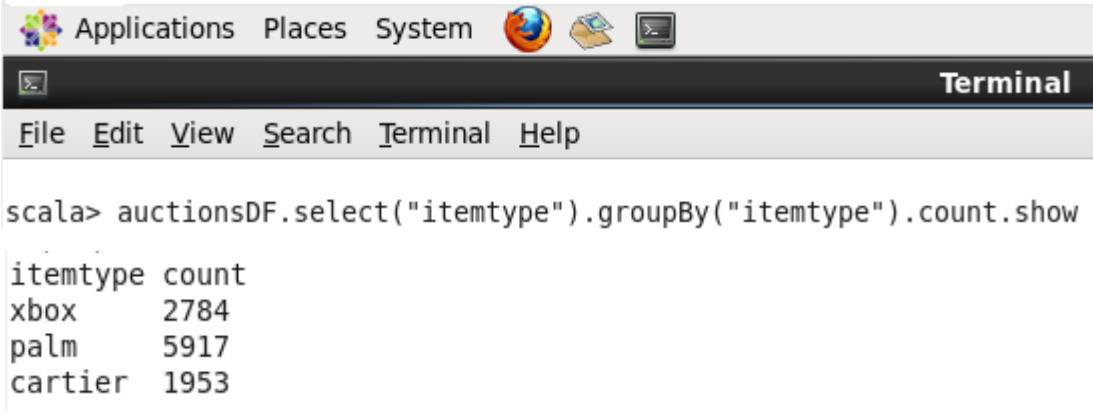
auctionid  count
3016384507 26
3024487267 33
3022000145 3
3024953755 20
8214435808 2
8214864154 12
8212668731 36
3024307014 6
3024680777 3
3020684186 27
3016651485 20
3020329965 29
3025507248 7
3020237085 18
8214803514 13
1644357387 25
3023639316 31
3023251181 24
3015053536 16
1644077820 11

```

```
scala> █
```

c) ¿Cuál es el número de objetos distintos?

Hay 3 objetos distintos. Xbox, Palm, Cartier. El siguiente código nos muestra cuantas pujas hay en cada objeto y se pueden contar 3 líneas.



```

Applications  Places  System  [Globe Icon] [Folder Icon] [Terminal Icon]
Terminal
File Edit View Search Terminal Help

scala> auctionsDF.select("itemtype").groupBy("itemtype").count.show

itemtype count
xbox       2784
palm       5917
cartier    1953

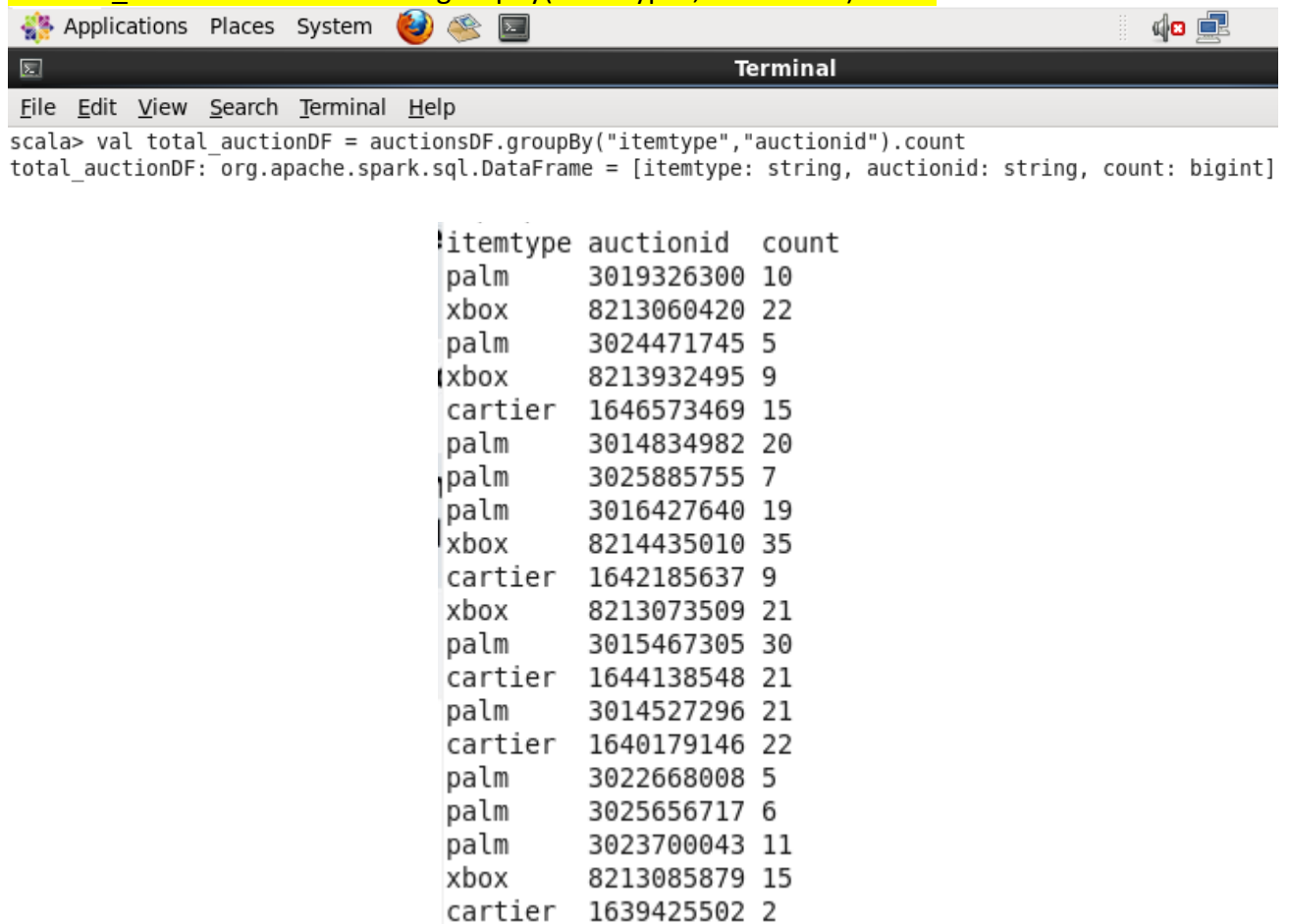
```

d) Queremos contar el número de pujas por subasta y tipo de objeto (como se muestra a continuación). ¿Como se podría hacer? (pista: usar *groupBy*)



itemtype	aucid	count
palm	3019326300	10
xbox	8213060420	22
palm	3024471745	5
xbox	8213932495	9
cartier	1646573469	15
palm	3014834982	20
palm	3025885755	7
palm	3016427640	19
xbox	8214435010	35
cartier	1642185637	9

```
val total_auctionDF = auctionsDF.groupBy("itemtype","auctionid").count
```



```

Applications Places System
Terminal
File Edit View Search Terminal Help
scala> val total_auctionDF = auctionsDF.groupBy("itemtype","auctionid").count
total_auctionDF: org.apache.spark.sql.DataFrame = [itemtype: string, auctionid: string, count: bigint]

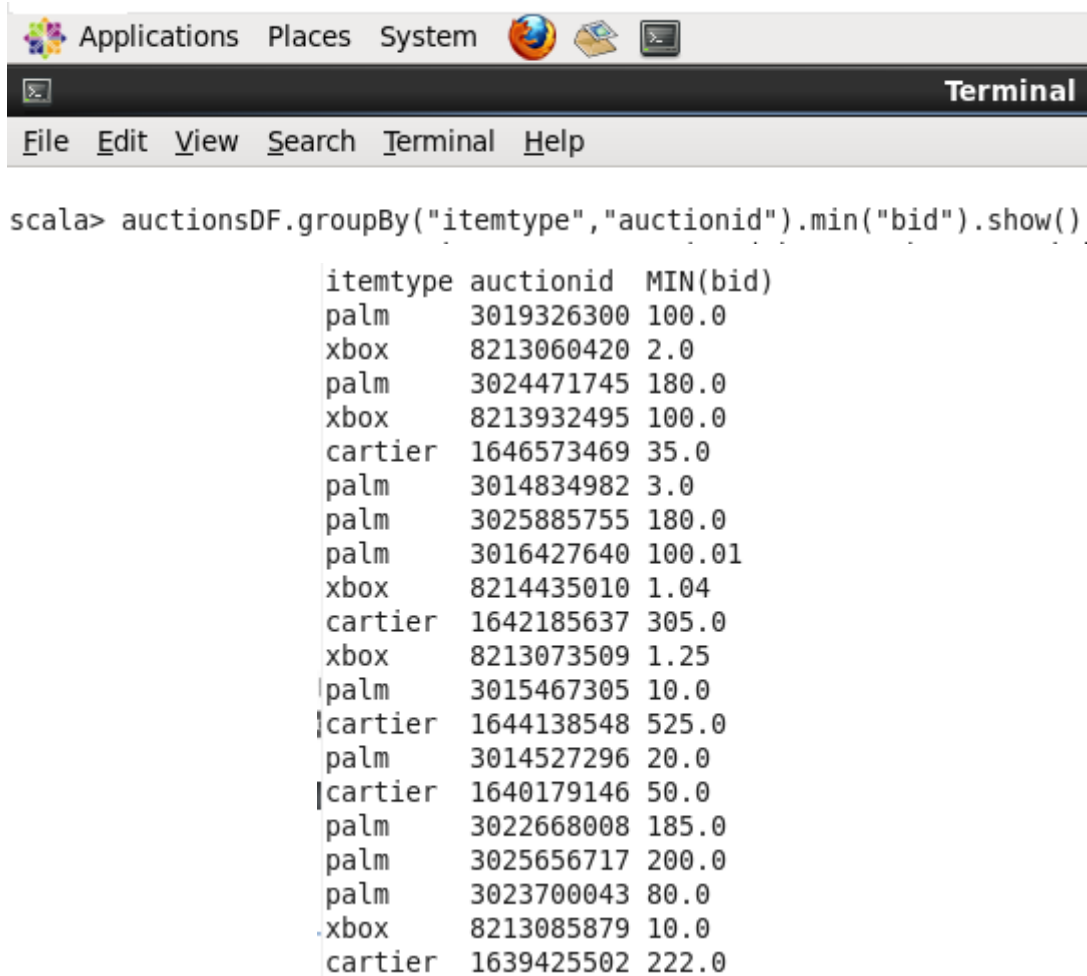
itemtype auctionid count
palm      3019326300 10
xbox      8213060420 22
palm      3024471745 5
xbox      8213932495 9
cartier   1646573469 15
palm      3014834982 20
palm      3025885755 7
palm      3016427640 19
xbox      8214435010 35
cartier   1642185637 9
xbox      8213073509 21
palm      3015467305 30
cartier   1644138548 21
palm      3014527296 21
cartier   1640179146 22
palm      3022668008 5
palm      3025656717 6
palm      3023700043 11
xbox      8213085879 15
cartier   1639425502 2

```

e) y f) Por cada elemento subastado y tipo, calcular el máximo, mínimo y la media de pujas. (pista: utilizar *groupBy* y *agg*)

a. Minima puja

```
auctionsDF.groupBy("itemtype","auctionid").min("bid").show()
```

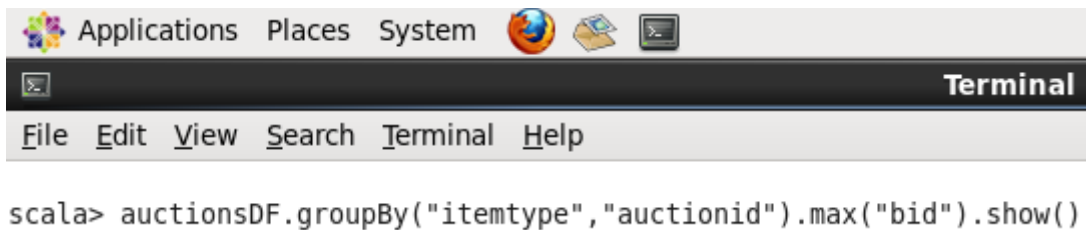


```
scala> auctionsDF.groupBy("itemtype","auctionid").min("bid").show()
```

itemtype	auctionid	MIN(bid)
palm	3019326300	100.0
xbox	8213060420	2.0
palm	3024471745	180.0
xbox	8213932495	100.0
cartier	1646573469	35.0
palm	3014834982	3.0
palm	3025885755	180.0
palm	3016427640	100.01
xbox	8214435010	1.04
cartier	1642185637	305.0
xbox	8213073509	1.25
palm	3015467305	10.0
cartier	1644138548	525.0
palm	3014527296	20.0
cartier	1640179146	50.0
palm	3022668008	185.0
palm	3025656717	200.0
palm	3023700043	80.0
xbox	8213085879	10.0
cartier	1639425502	222.0

b. Maxima puja

```
auctionsDF.groupBy("itemtype","auctionid").max("bid").show()
```

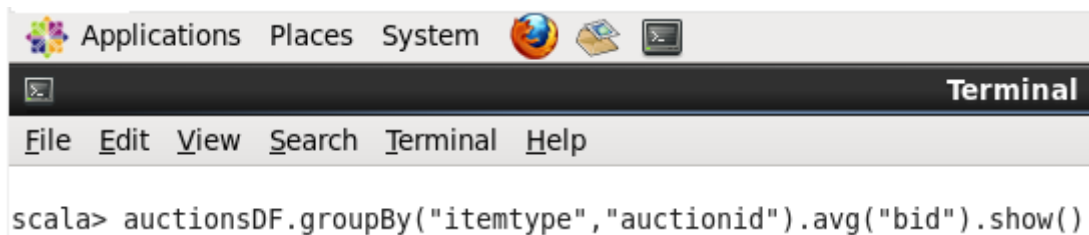


```
scala> auctionsDF.groupBy("itemtype","auctionid").max("bid").show()
```

itemtype	auctionid	MAX(bid)
palm	3019326300	207.5
xbox	8213060420	120.0
palm	3024471745	202.49
xbox	8213932495	127.5
cartier	1646573469	1226.0
palm	3014834982	217.5
palm	3025885755	203.5
palm	3016427640	245.0
xbox	8214435010	122.5
cartier	1642185637	510.0
xbox	8213073509	114.5
palm	3015467305	227.5
cartier	1644138548	2100.0
palm	3014527296	253.0
cartier	1640179146	455.0
palm	3022668008	210.1
palm	3025656717	224.01
palm	3023700043	228.01
xbox	8213085879	96.0
cartier	1639425502	224.5

c. Puja media

```
auctionsDF.groupBy("itemtype","auctionid").avg("bid").show()
```



The screenshot shows a terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal prompt is "scala>". The command entered is "auctionsDF.groupBy("itemtype","auctionid").avg("bid").show()".

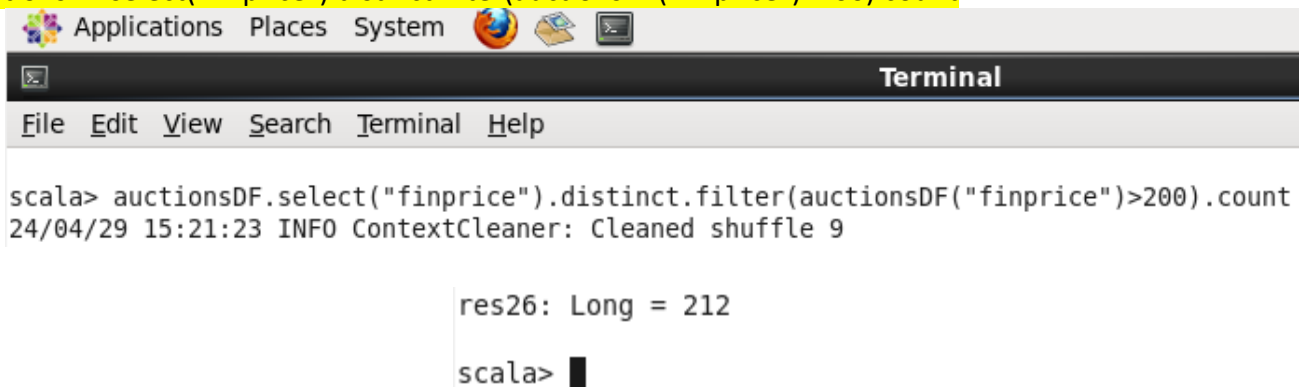
```
scala> auctionsDF.groupBy("itemtype","auctionid").avg("bid").show()
```

itemtype	auctionid	AVG(bid)
palm	3019326300	155.3490005493164
xbox	8213060420	66.4881818077781
palm	3024471745	191.49600219726562
xbox	8213932495	114.22222222222223
cartier	1646573469	800.6666666666666
palm	3014834982	96.1745002746582
palm	3025885755	192.5
palm	3016427640	178.36999993575247
xbox	8214435010	42.800571305411204
cartier	1642185637	402.22222222222223
xbox	8213073509	68.06904747372582
palm	3015467305	129.2449986775716
cartier	1644138548	1343.4761904761904
palm	3014527296	124.1585715157645
cartier	1640179146	303.0427259965376
palm	3022668008	201.02000122070314
palm	3025656717	215.58999633789062
palm	3023700043	168.7290899103338
xbox	8213085879	60.67000020345052
cartier	1639425502	223.25

g) ¿Cuál es el número de subastas cuyo precio es superior a 200?

Hay 212 subastas cuyo precio es mayor a 200.

```
auctionsDF.select("finprice").distinct.filter(auctionsDF("finprice")>200).count
```



```

Applications Places System
Terminal
File Edit View Search Terminal Help

scala> auctionsDF.select("finprice").distinct.filter(auctionsDF("finprice")>200).count
24/04/29 15:21:23 INFO ContextCleaner: Cleaned shuffle 9

res26: Long = 212

scala>

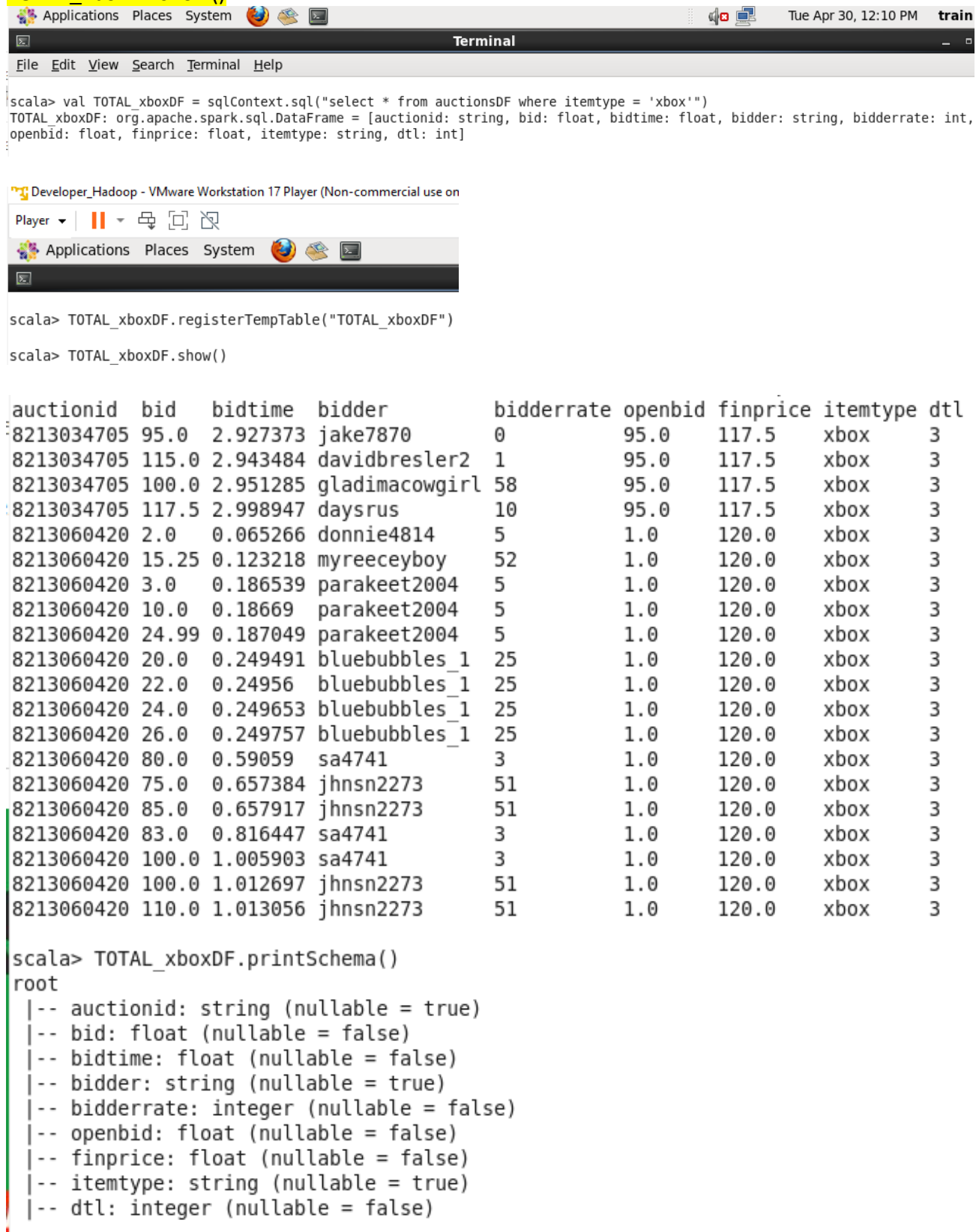
```

h) Queremos ejecutar algunos comandos básicos sobre todas las subastas que tienen un elemento del tipo “xbox”. Es decir, queremos asilar en un nuevo dataframe con datos de los artículos “xbox” ¿Qué manera tenemos de hacer esto?

*(pista: al tener registrado el DataFrame como una tabla podemos usar sentencias sql, el resultado será un nuevo DataFrame donde podremos aplicar acciones sobre el)*

```
val TOTAL_xboxDF = sqlContext.sql("select * from auctionsDF where itemtype = 'xbox'")
```

```
TOTAL_xboxDF.registerTempTable(TOTAL_xboxDF")
TOTAL_xboxDF.show()
```



```
scala> val TOTAL_xboxDF = sqlContext.sql("select * from auctionsDF where itemtype = 'xbox'")
TOTAL_xboxDF: org.apache.spark.sql.DataFrame = [auctionid: string, bid: float, bidtime: float, bidder: string, bidderrate: int, openbid: float, finprice: float, itemtype: string, dtl: int]
```

```
scala> TOTAL_xboxDF.registerTempTable("TOTAL_xboxDF")
scala> TOTAL_xboxDF.show()
```

auctionid	bid	bidtime	bidder	bidderrate	openbid	finprice	itemtype	dtl
8213034705	95.0	2.927373	jake7870	0	95.0	117.5	xbox	3
8213034705	115.0	2.943484	davidbresler2	1	95.0	117.5	xbox	3
8213034705	100.0	2.951285	gladimacowgirl	58	95.0	117.5	xbox	3
8213034705	117.5	2.998947	daysrus	10	95.0	117.5	xbox	3
8213060420	2.0	0.065266	donnies4814	5	1.0	120.0	xbox	3
8213060420	15.25	0.123218	myreeceyboy	52	1.0	120.0	xbox	3
8213060420	3.0	0.186539	parakeet2004	5	1.0	120.0	xbox	3
8213060420	10.0	0.18669	parakeet2004	5	1.0	120.0	xbox	3
8213060420	24.99	0.187049	parakeet2004	5	1.0	120.0	xbox	3
8213060420	20.0	0.249491	bluebubbles_1	25	1.0	120.0	xbox	3
8213060420	22.0	0.24956	bluebubbles_1	25	1.0	120.0	xbox	3
8213060420	24.0	0.249653	bluebubbles_1	25	1.0	120.0	xbox	3
8213060420	26.0	0.249757	bluebubbles_1	25	1.0	120.0	xbox	3
8213060420	80.0	0.59059	sa4741	3	1.0	120.0	xbox	3
8213060420	75.0	0.657384	jhnsn2273	51	1.0	120.0	xbox	3
8213060420	85.0	0.657917	jhnsn2273	51	1.0	120.0	xbox	3
8213060420	83.0	0.816447	sa4741	3	1.0	120.0	xbox	3
8213060420	100.0	1.005903	sa4741	3	1.0	120.0	xbox	3
8213060420	100.0	1.012697	jhnsn2273	51	1.0	120.0	xbox	3
8213060420	110.0	1.013056	jhnsn2273	51	1.0	120.0	xbox	3

```
scala> TOTAL_xboxDF.printSchema()
root
|-- auctionid: string (nullable = true)
|-- bid: float (nullable = false)
|-- bidtime: float (nullable = false)
|-- bidder: string (nullable = true)
|-- bidderrate: integer (nullable = false)
|-- openbid: float (nullable = false)
|-- finprice: float (nullable = false)
|-- itemtype: string (nullable = true)
|-- dtl: integer (nullable = false)
```

```
val comandos_xboxDF = sqlContext.sql("select * from TOTAL_xboxDF where bid>100")
```

comandos\_xboxDF.show()

```
Applications Places System [Icons] Tue Apr 30, 12:28 PM trainin
Terminal

scala> val comandos_xboxDF = sqlContext.sql("select * from TOTAL_xboxDF where bid>100")
comandos_xboxDF: org.apache.spark.sql.DataFrame = [auctionid: string, bid: float, bidtime: float, bidder: string, bidderrate: in
t, openbid: float, finprice: float, itemtype: string, dtl: int]

scala> comandos_xboxDF.show()

auctionid bid    bidtime bidder      bidderrate openbid finprice itemtype dtl
8213034705 115.0  2.943484 davidbresler2 1      95.0    117.5  xbox    3
8213034705 117.5  2.998947 daysrus      10     95.0    117.5  xbox    3
8213060420 110.0  1.013056 jhnsn2273    51     1.0     120.0  xbox    3
8213060420 105.0  2.79934  pagep123     2     1.0     120.0  xbox    3
8213060420 110.0  2.799676 pagep123     2     1.0     120.0  xbox    3
8213060420 115.0  2.800197 pagep123     2     1.0     120.0  xbox    3
8213060420 115.0  2.968495 skcardina    1     1.0     120.0  xbox    3
8213060420 117.5  2.972766 skcardina    1     1.0     120.0  xbox    3
8213060420 120.0  2.999722 djnoeproductions 17    1.0     120.0  xbox    3
8213067838 120.0  2.946516 godb_24      0     29.99   132.5  xbox    3
8213067838 105.0  2.989051 unique82me   0     29.99   132.5  xbox    3
8213067838 110.0  2.989178 unique82me   0     29.99   132.5  xbox    3
8213067838 115.0  2.989317 unique82me   0     29.99   132.5  xbox    3
8213067838 120.0  2.989387 unique82me   0     29.99   132.5  xbox    3
8213067838 125.0  2.989468 unique82me   0     29.99   132.5  xbox    3
8213067838 130.0  2.99316  unique82me   0     29.99   132.5  xbox    3
8213067838 130.0  2.996516 *champaignbubbles* 202    29.99   132.5  xbox    3
8213067838 132.5  2.996632 *champaignbubbles* 202    29.99   132.5  xbox    3
8213067838 132.5  2.997789 *champaignbubbles* 202    29.99   132.5  xbox    3
8213073509 101.09 2.639931 djfelony     265    1.0     114.5  xbox    3
```

Val comandos\_xboxDF = sqlContext.sql("select distinct bidder from TOTAL\_xboxDF where bid>100")

```
Player [Icons] Applications Places System [Icons] [Icons]
Terminal

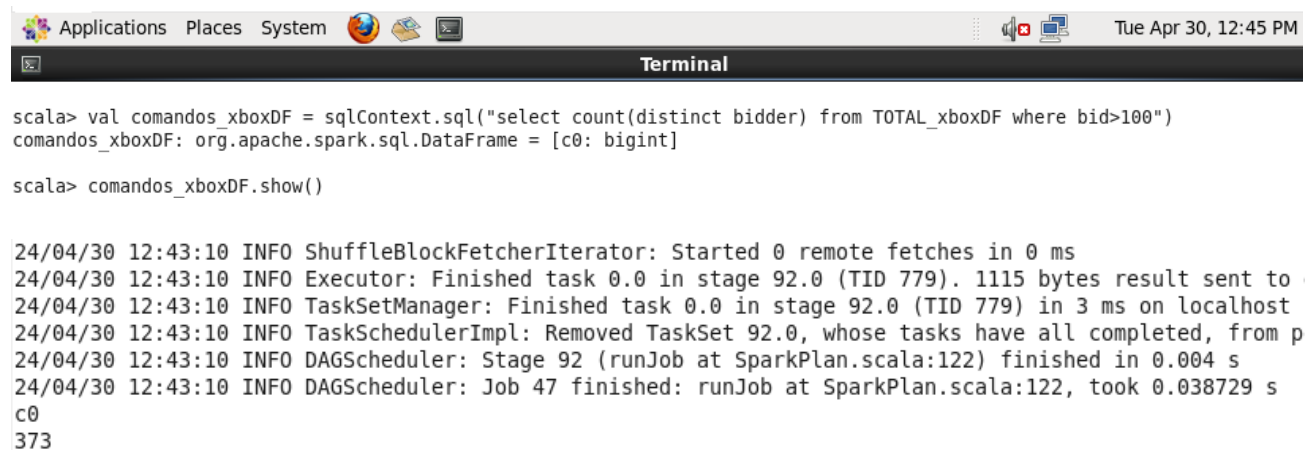
scala> val comandos_xboxDF = sqlContext.sql("select distinct bidder from TOTAL_xboxDF where bid>100")
```

```
bidder
tchick4270
loraron513
darkaglmax84
tanktabooda
daysrus
205_dmsdud
kermitthefrock
kiflayghiorghis
rr6kids
jazzjake2005
pjamkeat
skcardina
biznezb4plezure
42179odie
nalinprabhu
sno_oman
daveonbay
damon8452
sweetie6684
dbecerraii
```

```
Val comandos_xboxDF = sqlContext.sql("select count(distinct bidder) from TOTAL_xboxDF where
bid>100")
```

```
comandos_xboxDF.show()
```

La respuesta es que hay 373 distintos



```
Applications Places System [Icons] Tue Apr 30, 12:45 PM
Terminal

scala> val comandos_xboxDF = sqlContext.sql("select count(distinct bidder) from TOTAL_xboxDF where bid>100")
comandos_xboxDF: org.apache.spark.sql.DataFrame = [c0: bigint]

scala> comandos_xboxDF.show()

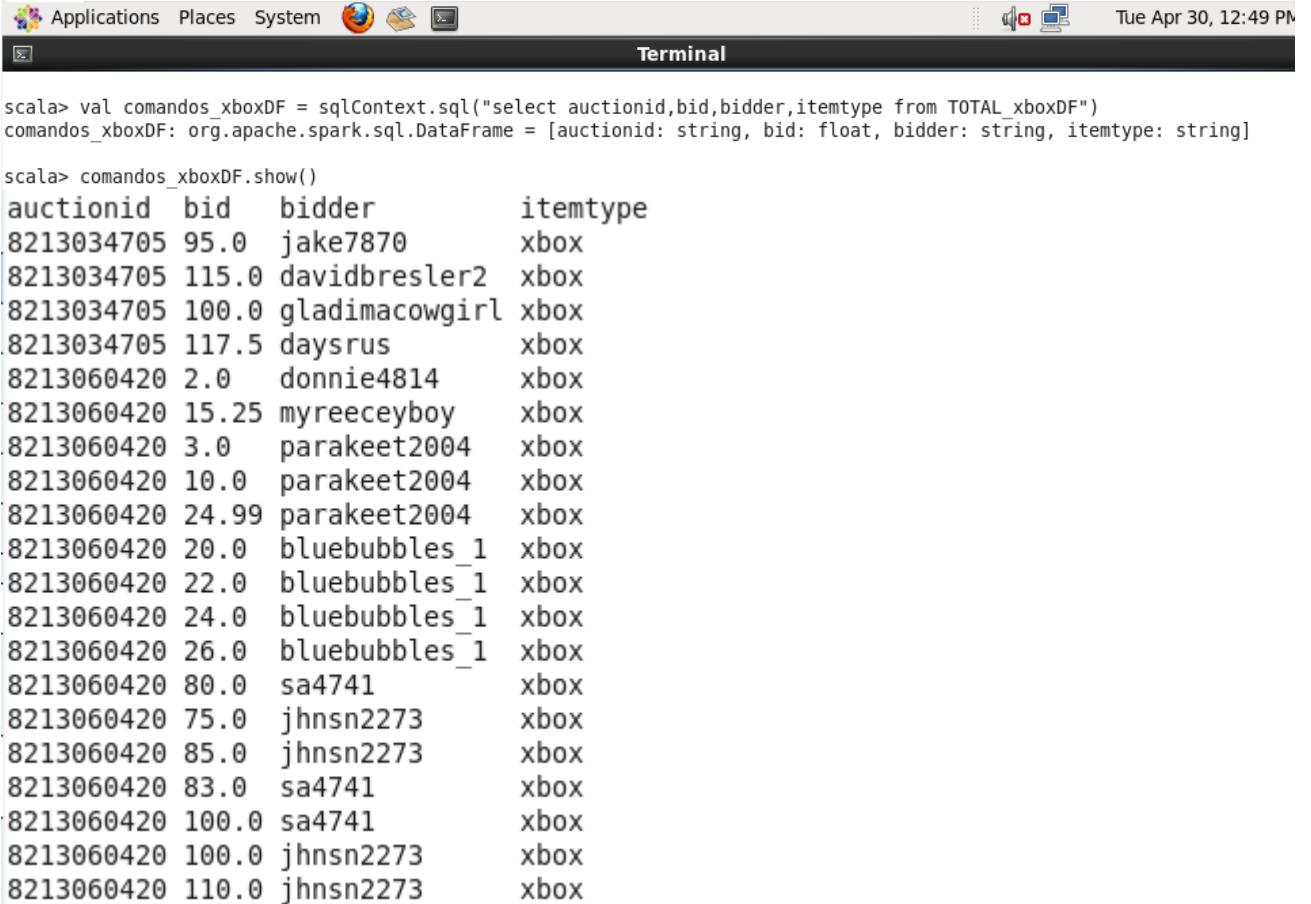
24/04/30 12:43:10 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
24/04/30 12:43:10 INFO Executor: Finished task 0.0 in stage 92.0 (TID 779). 1115 bytes result sent to
24/04/30 12:43:10 INFO TaskSetManager: Finished task 0.0 in stage 92.0 (TID 779) in 3 ms on localhost
24/04/30 12:43:10 INFO TaskSchedulerImpl: Removed TaskSet 92.0, whose tasks have all completed, from p
24/04/30 12:43:10 INFO DAGScheduler: Stage 92 (runJob at SparkPlan.scala:122) finished in 0.004 s
24/04/30 12:43:10 INFO DAGScheduler: Job 47 finished: runJob at SparkPlan.scala:122, took 0.038729 s
c0
373
```

i) Mostrar el precio de todas las subastas con el elemento Xbox implicado (puedes utilizar el dataframe anterior)



```
val comandos_xboxDF = sqlContext.sql("select auctionid,bid,bidder,itemtype from  
TOTAL_xboxDF")
```

```
comandos_xboxDF.show()
```



The image shows a terminal window titled "Terminal" with a macOS-style title bar. The terminal displays the execution of a Spark SQL query and its results. The query selects auctionid, bid, bidder, and itemtype from a table named TOTAL\_xboxDF. The results are printed as a table with 4 columns: auctionid, bid, bidder, and itemtype. There are 20 rows of data, all with itemtype 'xbox'.

```
scala> val comandos_xboxDF = sqlContext.sql("select auctionid,bid,bidder,itemtype from TOTAL_xboxDF")  
comandos_xboxDF: org.apache.spark.sql.DataFrame = [auctionid: string, bid: float, bidder: string, itemtype: string]  
  
scala> comandos_xboxDF.show()  
auctionid  bid    bidder      itemtype  
8213034705 95.0   jake7870     xbox  
8213034705 115.0  davidbresler2 xbox  
8213034705 100.0  gladimacowgirl xbox  
8213034705 117.5  daysrus      xbox  
8213060420 2.0    donnie4814   xbox  
8213060420 15.25  myreeceyboy  xbox  
8213060420 3.0    parakeet2004 xbox  
8213060420 10.0   parakeet2004 xbox  
8213060420 24.99  parakeet2004 xbox  
8213060420 20.0   bluebubbles_1 xbox  
8213060420 22.0   bluebubbles_1 xbox  
8213060420 24.0   bluebubbles_1 xbox  
8213060420 26.0   bluebubbles_1 xbox  
8213060420 80.0   sa4741       xbox  
8213060420 75.0   jhnsn2273    xbox  
8213060420 85.0   jhnsn2273    xbox  
8213060420 83.0   sa4741       xbox  
8213060420 100.0  sa4741       xbox  
8213060420 100.0  jhnsn2273    xbox  
8213060420 110.0  jhnsn2273    xbox
```