|  |
| --- |
| Máster en Big Data |
| Tecnologías de Almacenamiento |
| 1. Hands-On: Desarrollo Apache Spark SQL |

Presentado por: José David Angulo y Albert Ripoll

Índice

[1. Introducción 3](#_Toc96244381)

[2. Entorno 3](#_Toc96244382)

[3. Creación del DataFrame 3](#_Toc96244383)

[4. Inspección de datos 7](#_Toc96244384)

# 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

# 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

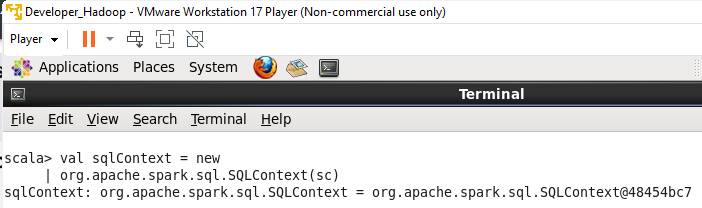
El dataset que utilizaremos se llama auctiondata.csv y está ubicado en */home/training/training\_materials/developer/data/auction.csv*

# Creación del DataFrame

1. Crear el SQL Context

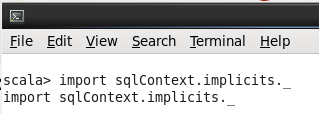
val sqlContext = new

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

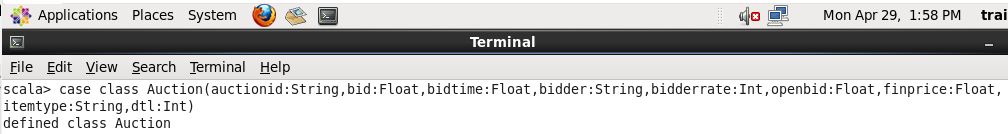


1. Realiza el import correspondiente para convertir un RDD en un DataFrame

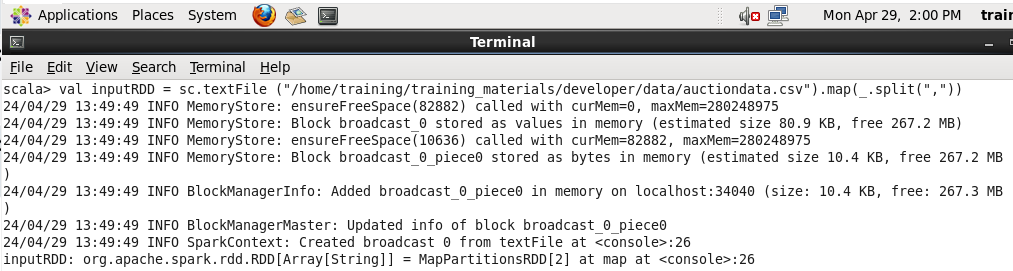
import sqlContext.implicits.\_



1. Definición del esquema usando una *case class*

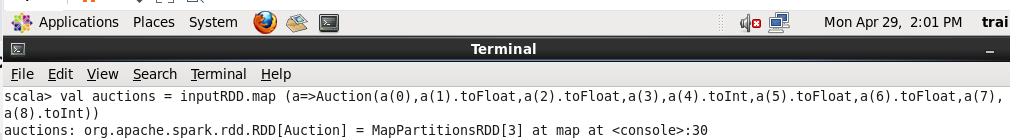
case class Auction(auctionid:String,bid:Float,bidtime:Float,bidder:String,bidderrate:Int,openbid:Float,finprice:Float,itemtype:String,dtl:Int)

1. 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(","))

1. Ahora Mapea el *inputRDD* a *case class*

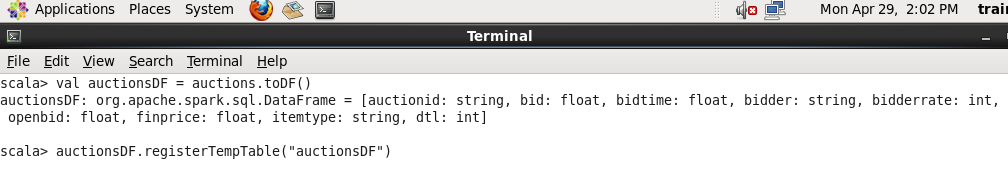
*val auctions = inputRDD.map (a=>Auction(a(0),a(1).toFloat,a(2).toFloat,a(3),a(4).toInt,a(5).toFloat,a(6).toFloat,a(7),a(8).toInt))*

**

1. 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)

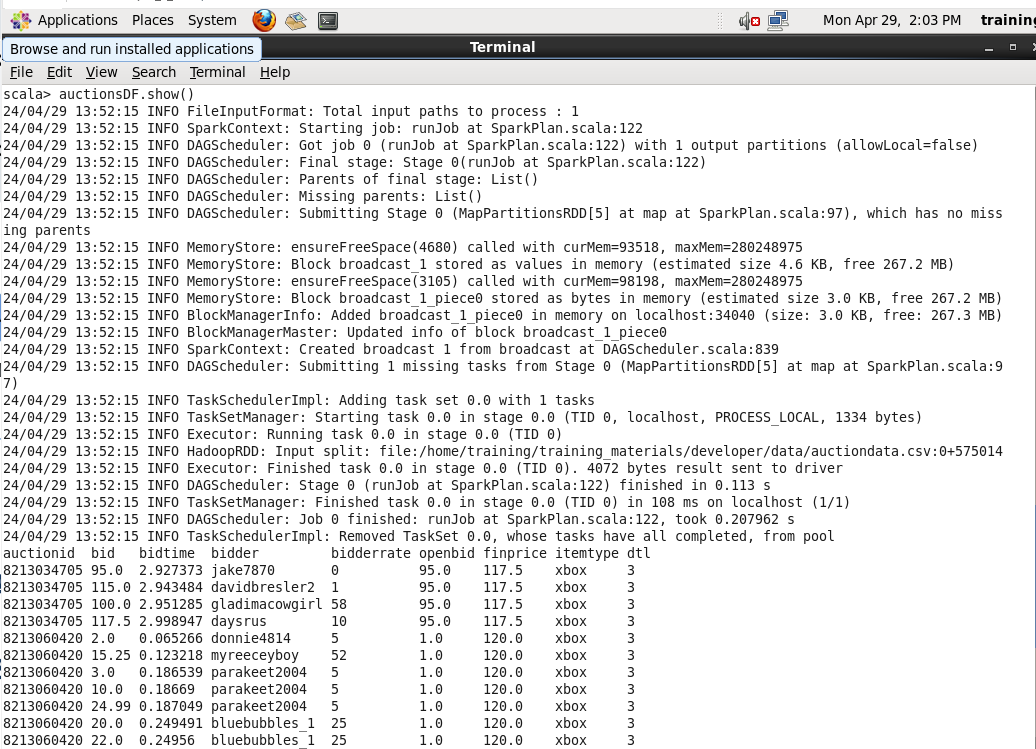
val auctionsDF = auctions.toDF()

auctionsDF.registerTempTable("auctionsDF")



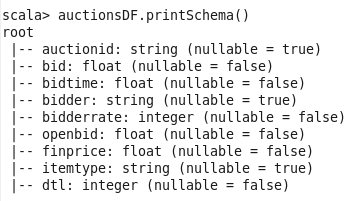
1. ¿Que acción puedes hacer para comprobar los datos en el DataFrame?

auctionsDF.show()



1. ¿Que función del DataFrame puedes utilizar para ver el esquema del mismo?

auctionsDF.printSchema()

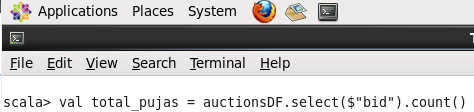


# Inspección de datos

1. ¿Cuál es el número total de pujas?

val total\_pujas = auctionsDF.select($”bid”).count()

Respuesta: 10654

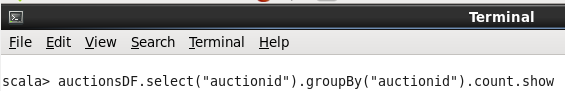


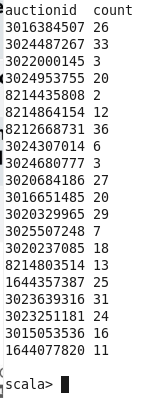
C:\Users\alber\Downloads\4a2.png

1. ¿Cuál es el número total de subastas distintas?

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

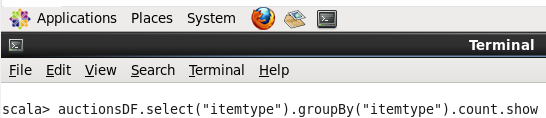
auctionsDF.select(“auctionid”).groupBy(“auctionid”).count.show

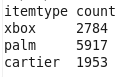




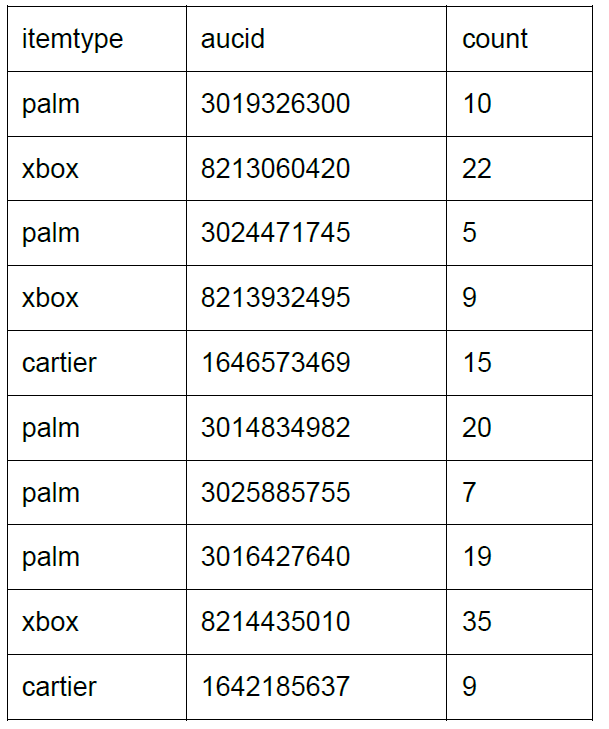
1. ¿Cuál es el número de objetos distintos?

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

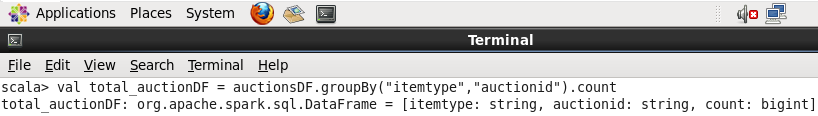


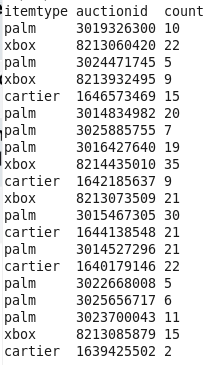


1. Queremos contar el número de pujas por subasta y tipo de objeto (como se muestra a continuación). ¿Como se podría hacer? *(pista: usar groupBy)*

****

val total\_auctionDF = auctionsDF.groupBy("itemtype","auctionid").count

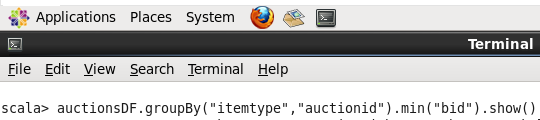


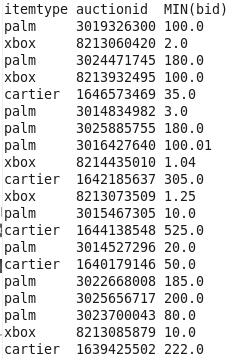


e) y f) Por cada elemento subastado y tipo, calcular el máximo, mínimo y la media de pujas. (pista: utilizar *groupBy* y *agg*)

* 1. Minima puja

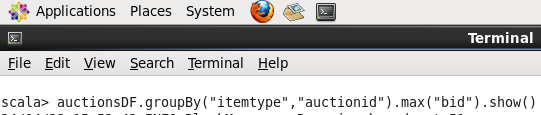
auctionsDF.groupBy("itemtype","auctionid").min("bid").show()

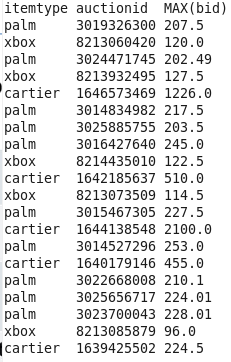




* 1. Maxima puja

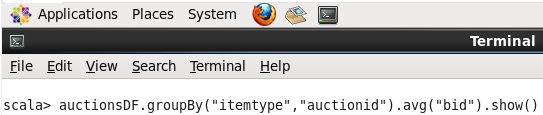
auctionsDF.groupBy("itemtype","auctionid").max("bid").show()

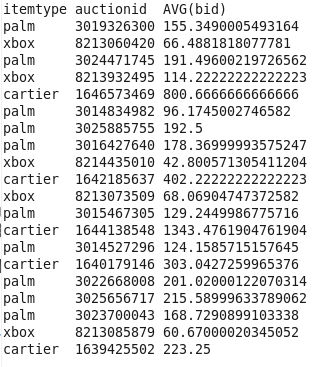




* 1. Puja media

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

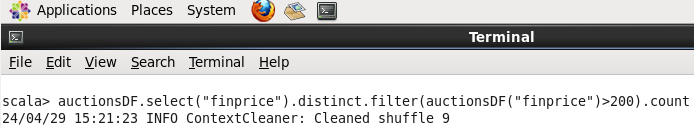




g) ¿Cuál es el número de subastas cuyo precio es superior a 200?

Hay 212 subastas cuyo precio es mayor a 200.

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



C:\Users\alber\Downloads\4g2.png

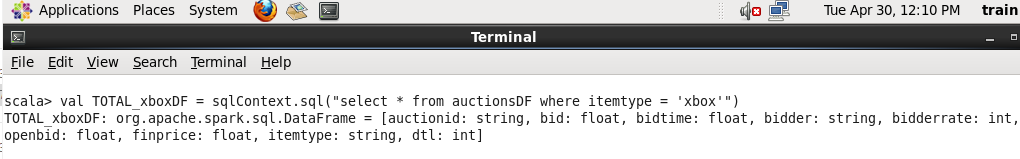
h) Queremos ejecutar algunos comandos básicos sobre todas las subastas que tienen un elemento del tipo “xbox”. Es decir, queremos asilar en un nuevo dataframe con datos de los artículos “xbox” ¿Qué manera tenemos de hacer esto?

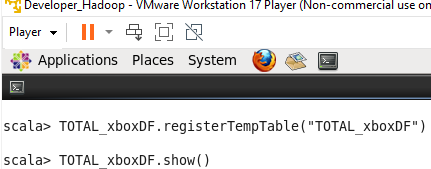
*(pista: al tener registrado el DataFrame como una tabla podemos usar sentencias sql, el resultado será un nuevo DataFrame donde podremos aplicar acciones sobre el)*

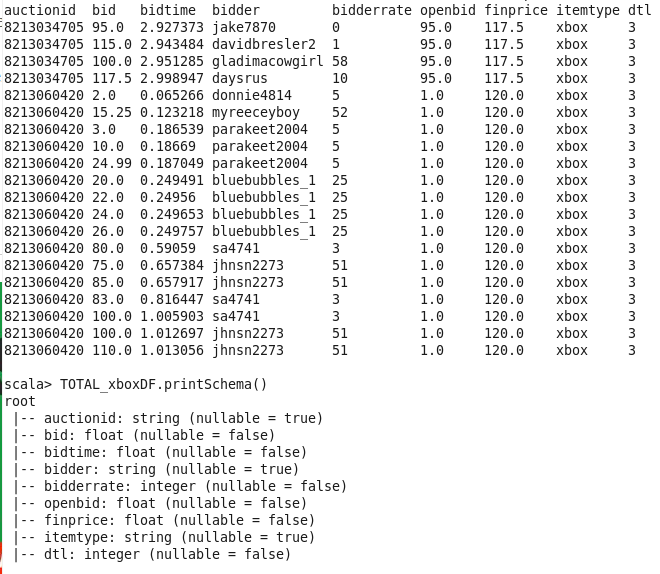
val TOTAL\_xboxDF = sqlContext.sql("select \* from auctionsDF where itemtype = 'xbox'")

TOTAL\_xboxDF.registerTempTable(TOTAL\_xboxDF”)

TOTAL\_xboxDF.show()

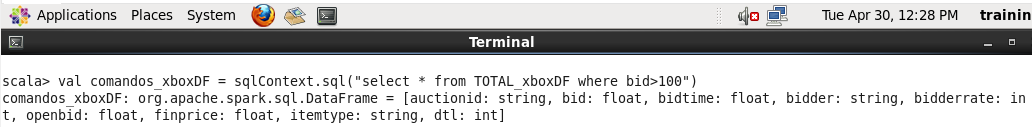
**

**

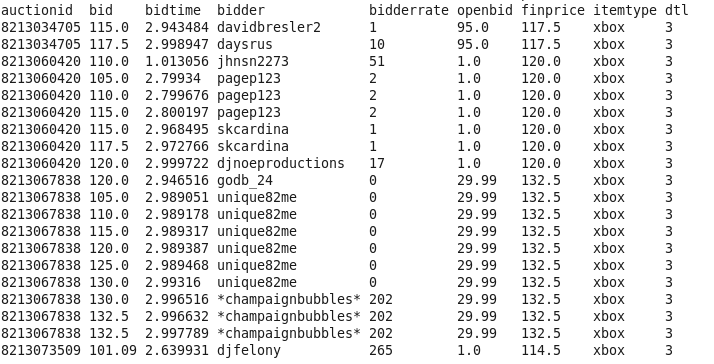
**

val comandos\_xboxDF = sqlContext.sql(“select \* from TOTAL\_xboxDF where bid>100”)

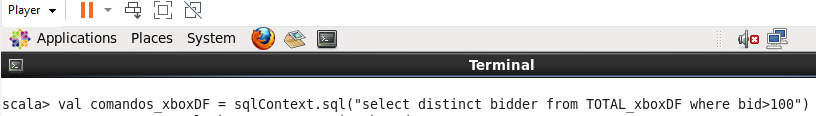
comandos\_xboxDF.show()

**

*C:\Users\alber\Downloads\h5.png*

**

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

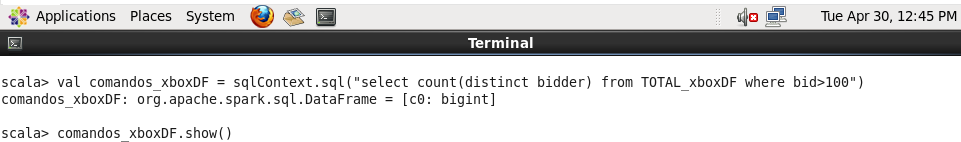
**

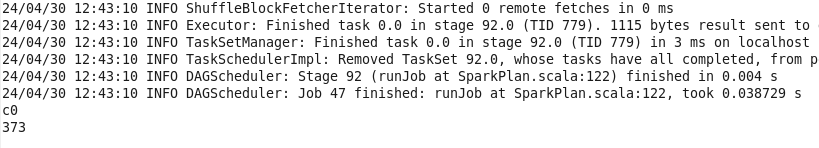
**

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

**

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

val comandos\_xboxDF = sqlContext.sql(“select auctionid,bid,bidder,itemtype from TOTAL\_xboxDF”)

comandos\_xboxDF.show()

