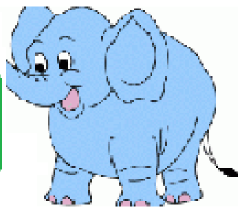
Data, BigData. Cloud. Anaytics & Virtualizaton HadoopExam.com | QuickTechie.com

SPARK SQL HANDSON LAB QUERYING CSV FILE USING DATAFRAME & SCALA CASE CLASSES AS SCHEMA

By www.HadoopExam.com

Note: These instructions should be used with the HadoopExam Apache Spark: Professional Trainings. Where it is executed and you can do hands on with trainer.



Data.csv
Auctioned,***bid,bidtime***,***bidder***,***bidderrate***,***openbid***,***price***,***item***,***daystolive***
8213034715,15,12.373,baman,3,12,20,book1,5
8213034725,65,21.33,thmpu,2,64,75,watch1,9
8213034735,85,23.3,lovekush,4,45,90,remote1,10
8213034745,115,44.44,jaipanee,3,111,130,s3phone,4
8213034755,950,12.1,john,2,700,999,iphone6,7
8213034705,95,2.92,alphi,0,95,117.5,xbox,4
8213034765,145,34.9,jukati,4,111,200,screwbox,9
8213034775,195,67.27373,lui fung,3,195,195,datapack1,12
8213034785,286,89.373,valleryka,3,200,300,wddisk,23
8213034795,766,90.33,jesika24,2,500,800, iphone6,63
8213034105,845,12.45,remisha,1,700,880, iphone6,20
8213034205,23,56.54,alokion,0,12,30,book23,44
8213034305,14,78.9,nimarana,4,10,25,babytoy,45
8213034405,09,45.12,great123,2,01,12,babytoy1,56
8213034505,11,32.73,zellonee,3,10,15,toy2,23
8213034605,22,21.2,papa1234,1,20,27,toy3,12
8213034805,76,29.1,monkey123,0,50,90,book1,15
8213034905,98,56.44,wowwow,4,50,100,AmzoneVoucher,16
8213034005,144,16.8,merahero,3,100,150,flipkartvoucher,19
8213035705,252,12.11,dabang34,2,200,260,snapdealvoucer,22

1. Required Import statement and initialize sqlContext

```
//    SQLContext entry point for working with structured data
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
// this is used to implicitly convert an RDD to a DataFrame.
import sqlContext.implicits._
// Import Spark SQL data types and Row.
import org.apache.spark.sql._
```

2. Load data file in hdfs using hue. (As shown in video)

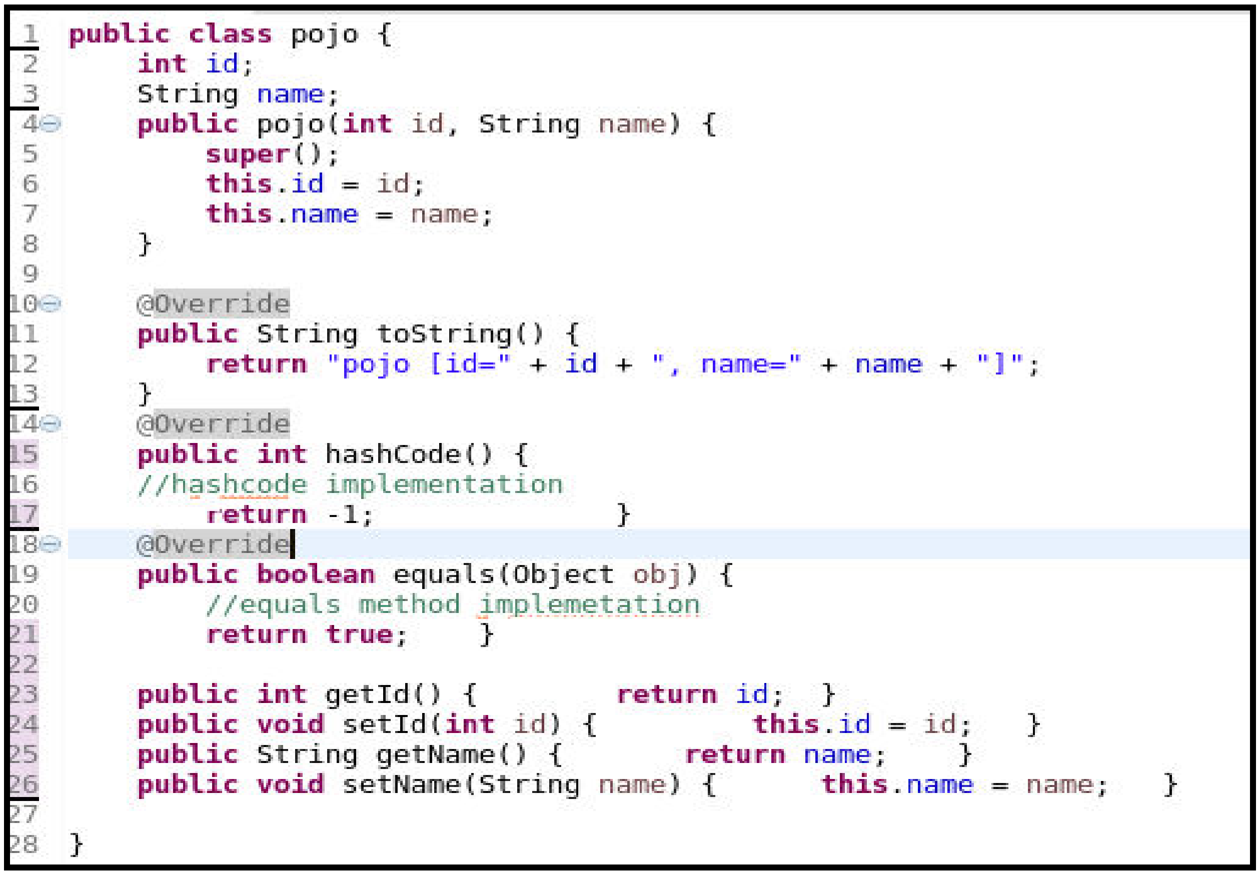
3. Load data in SparkRDD or create RDD using this data.

```
val datafile = sc.textFile("data.csv")
```

4. Check RDD using first line output, whether RDD is correctly loaded or not.

```
datafile.first()
```

5. Define Scala case class :


```
public class pojo {
    int id;
    String name;
    public pojo(int id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    @Override
    public String toString() {
        return "pojo [id=" + id + ", name=" + name + "]";
    }
    @Override
    public int hashCode() {
    //hashcode implementation
        return -1;              }
    @Override
    public boolean equals(Object obj) {
        //equals method implemetation
        return true;    }

    public int getId() {        return id;  }
    public void setId(int id) {         this.id = id;    }
    public String getName() {       return name;    }
    public void setName(String name) {      this.name = name;    }

}
```

```
//define the schema using a case class
case class Data(auctionid: String, bid: Float, bidtime: Float, bidder: String, bidderrate: Integer, openbid: Float, price: Float, item: String, daystolive: Integer)
```

6. Create an RDD from data using Scala case class as schema.

```
// create an RDD of Data objects
val allDataRDD = datafile.map(_.split(",")).map(p =>
Data(p(0),p(1).toFloat,p(2).toFloat,p(3),p(4).toInt,p(5).toFloat,p(6).toFloat,p(7),p(8).toInt ))
```

7. Check the first element in RDD and count number of elements.

```
 allDataRDD.first()
allDataRDD.count()
```

8. **RDD to DataFrme**: A DataFrame is a distributed collection of data organized into named columns. Spark SQL supports automatically converting an RDD containing case classes to a DataFrame with the method toDF():

```
val rddDF = allDataRDD.toDF()
```

9. Explore dataframes using scala API.

```
// Display the top 20 rows of DataFrame
rddDF.show()

//PrintSchema
rddDF.printSchema()

// How many auctions were held?
rddDF.select("auctionid").distinct.count

// How many bids per item?
rddDF.groupBy("auctionid", "item").count.show

// What's the min number of bids per item? what's the average? what's the max?
rddDF.groupBy("item", "auctionid").count.agg(min("count"), avg("count"),max("count")).show

// Get the auctions with closing price > 100
val highprice= rddDF.filter("price > 100")

// display dataframe in a tabular format
highprice.show()
```

10. **Register dataframe as a Temporary table:** We can register a DataFrame as a temporary table using a any name, and then run SQL statements using the sql methods provided by sqlContext. Here are some example queries using sqlContext:

```
// register the DataFrame as a temp table
rddDF.registerTempTable("datatable")

// How many bids per auction?
val results =sqlContext.sql("SELECT auctionid, item,    count(bid) FROM datatable GROUP BY auctionid,
item")

// display dataframe in a tabular format
results.show()

// display max price for an auctionId
val results =sqlContext.sql("SELECT auctionid, MAX(price) FROM datatable    GROUP BY item,auctionid")
results.show()

//Getting the Physical plan for a querry.
val results =sqlContext.sql("SELECT auctionid, item,    count(bid) FROM datatable GROUP BY auctionid,
item").explain
```