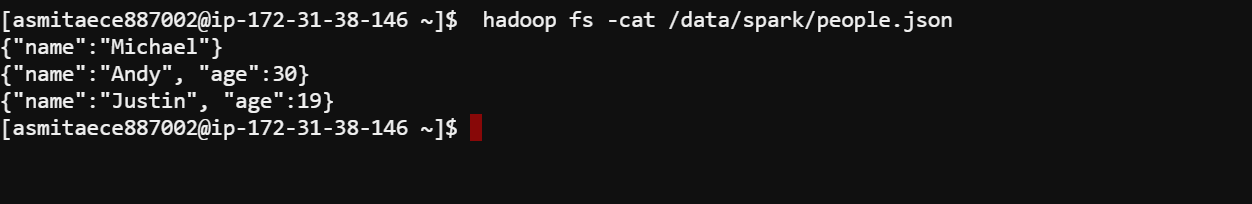
**Examples(marked by ##)**

**###:loading data from json file to spark dataframe**

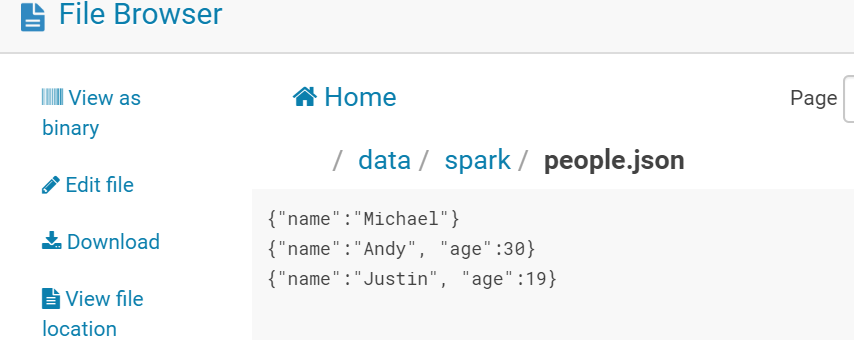
---first get the file from hadoop and copy it to home path

from web console----

hadoop fs -cat /data/spark/people.json



View of the file from Hue :



Next :

You can see that this file contains three JSON objects.

Each line is a valid JSON object. Each of these lines is separated by a newline.

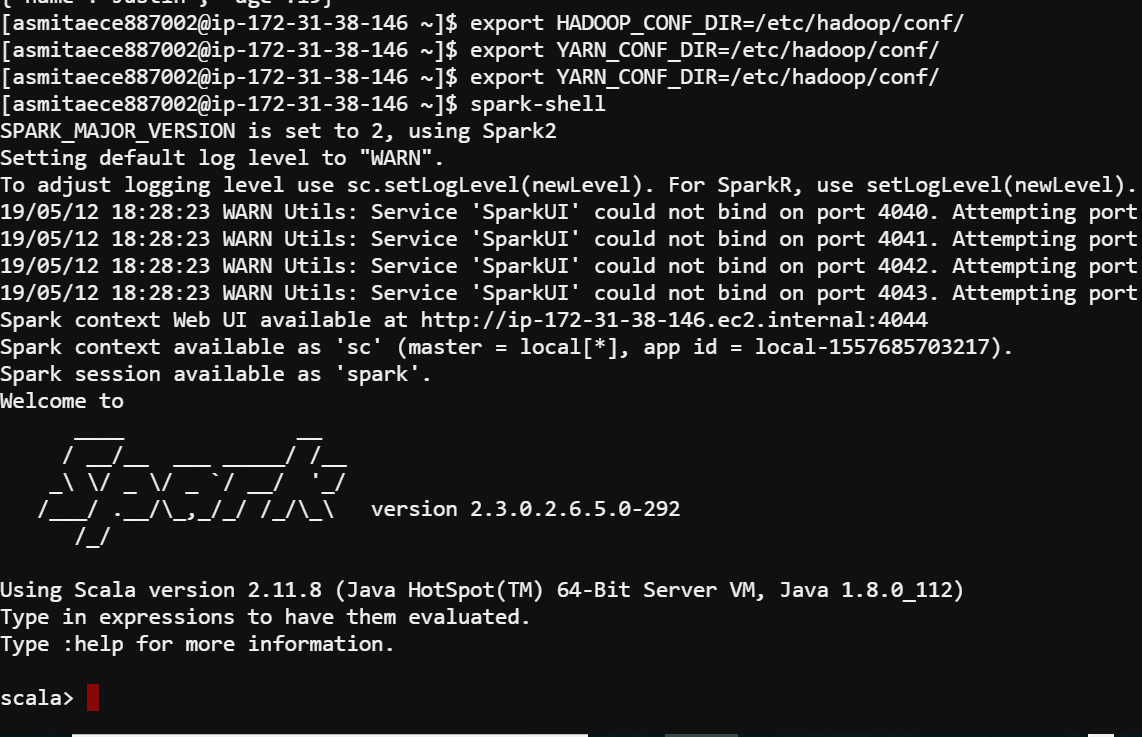
Please note the complete file does not represent a valid json object while each line is a valid json object.

Now export the environment variables :

**export HADOOP\_CONF\_DIR=/etc/hadoop/conf/**

**export YARN\_CONF\_DIR=/etc/hadoop/conf/**

call spark-shell.



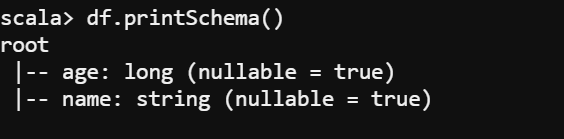
Next read the file :

import spark.implicits.\_

var df = spark.read.json("/data/spark/people.json")

df.show()

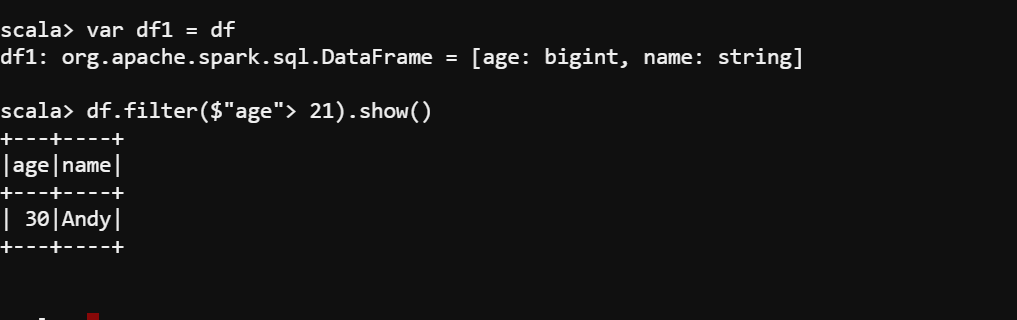




**##Now check the records having age >21**

var df1 = df

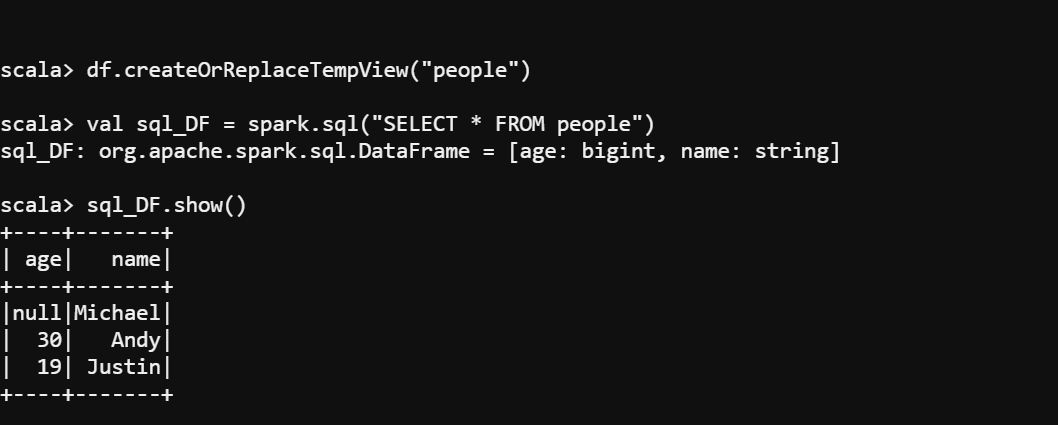
df.filter($"age"> 21).show()



**## register a view and check the query records**

df.createOrReplaceTempView("people")

val sql\_DF = spark.sql("SELECT \* FROM people")



**##using Encoders**

---Define a case class

**case class Employee (name:String , age : Long )**

**---using case classes as encoders**

**val caseclassDS = Seq(Employee("Andy",32)).toDS()**

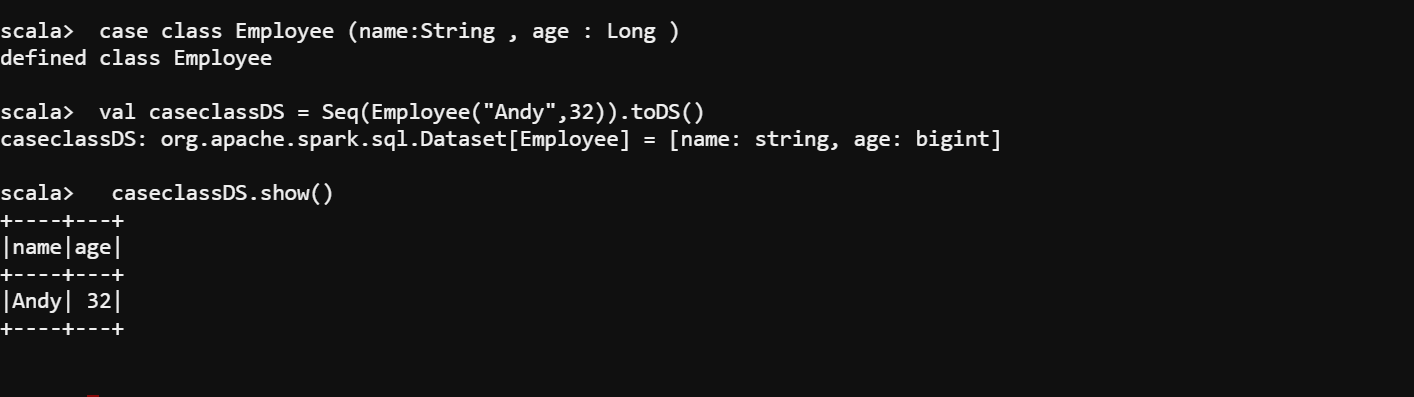
**caseclassDS.show()**

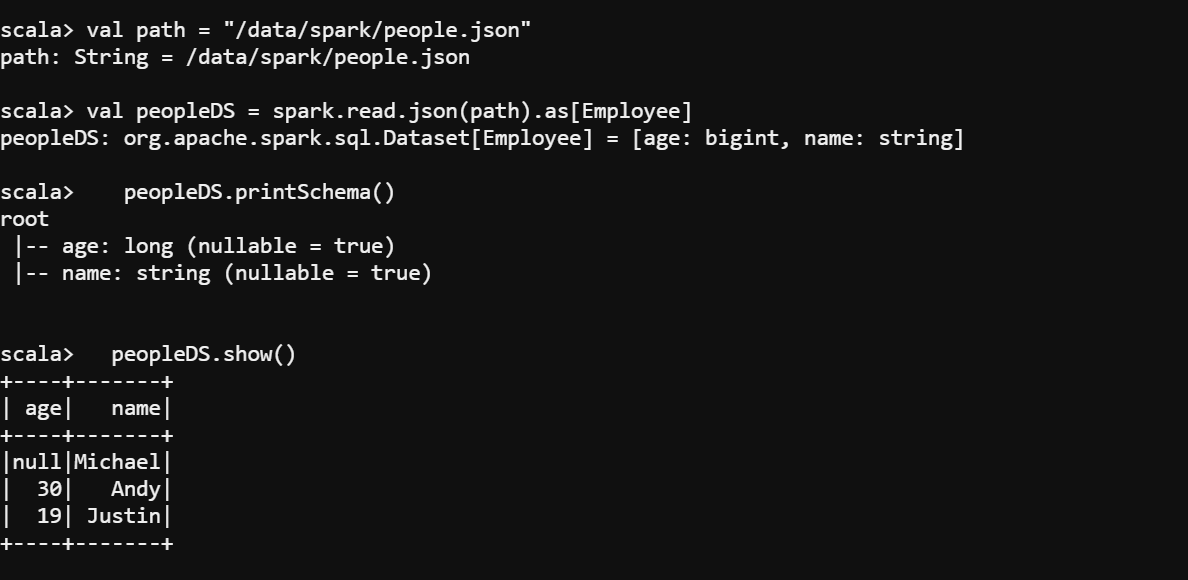
**val path = "/data/spark/people.json"**

**val peopleDS = spark.read.json(path).as[Employee]**

**peopleDS.printSchema()**

**peopleDS.show()**





**###Inferring the Schema Using Reflection**

**1st check the text file:**



**Next check the code in spark-shell**

import spark.implicits.\_

///define a case class

case class Person(name: String, age: Long)

///read the file

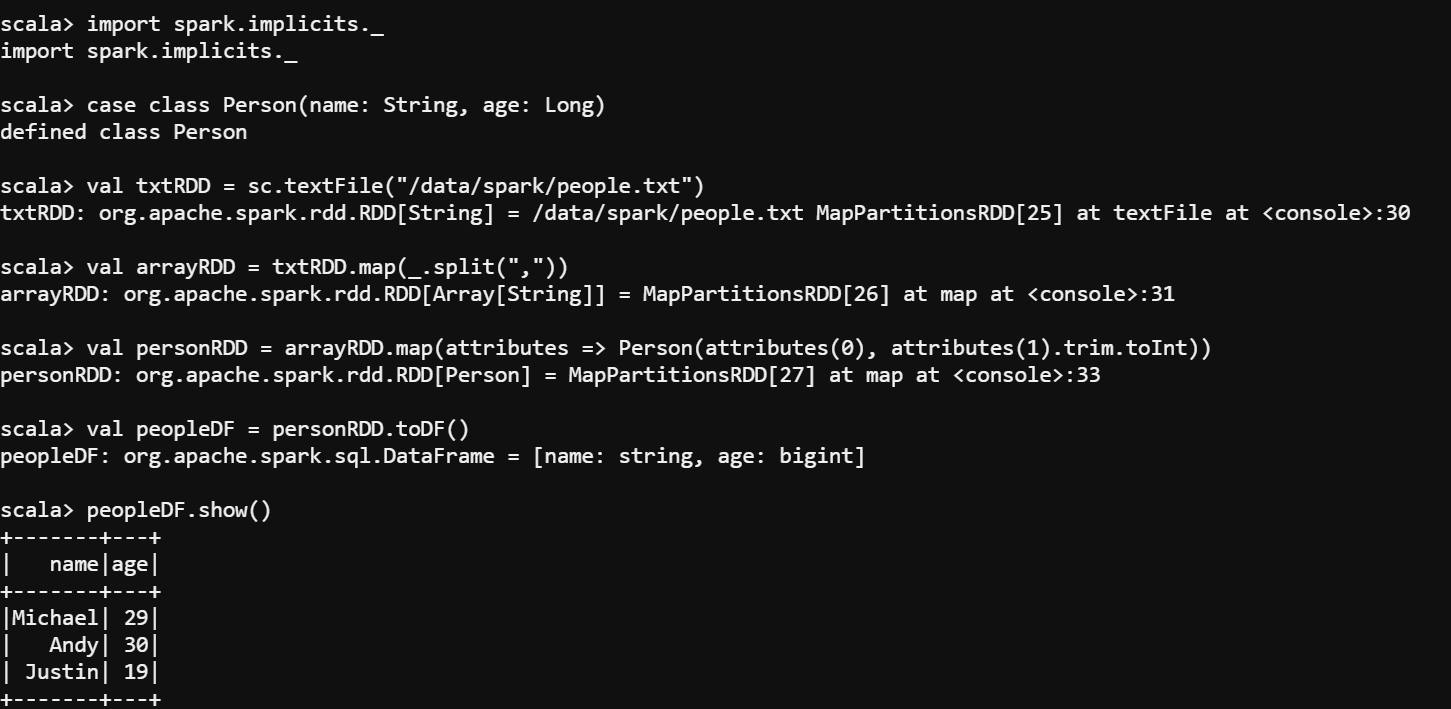
val txtRDD = sc.textFile("/data/spark/people.txt")

val arrayRDD = txtRDD.map(\_.split(","))

val personRDD = arrayRDD.map(attributes => Person(attributes(0), attributes(1).trim.toInt))

val peopleDF = personRDD.toDF()

peopleDF.show()



// Register the DataFrame as a temporary view

**peopleDF.createOrReplaceTempView("people")**

// SQL statements can be run by using the sql methods provided by Spark

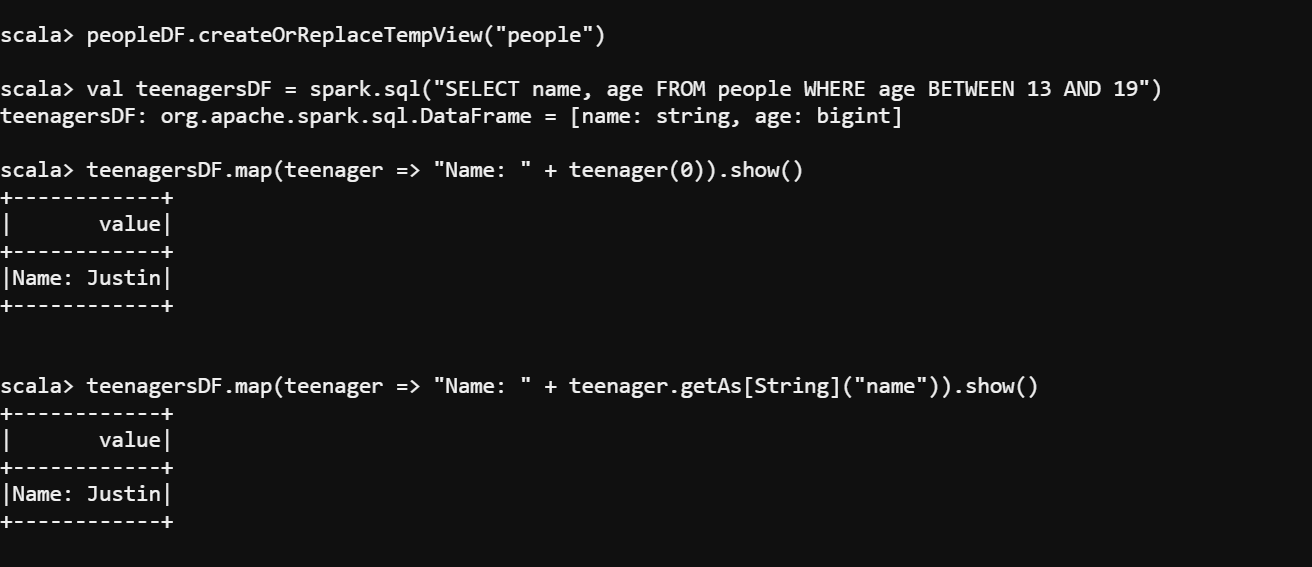
**val teenagersDF = spark.sql("SELECT name, age FROM people WHERE age BETWEEN 13 AND 19")**

// The columns of a row in the result can be accessed by field index

**teenagersDF.map(teenager => "Name: " + teenager(0)).show()**

// or by field name

**teenagersDF.map(teenager => "Name: " + teenager.getAs[String]("name")).show()**



**############When dataframes cannot be created using schema by reflection , we need to do programatically**

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.\_

//The schema is encoded in a string

//user provided variable

val schemaString = "name age"///idealy this string should be coming from a json file

val filename = "/data/spark/people.txt"

val fieldsArray = schemaString.split(" ")

val fields = fieldsArray.map(f=> StructField(f,StringType ,nullable = true))

val schema = StructType(fields)

val peopleRDD = spark.sparkContext.textFile(filename)

val rowRDD = peopleRDD.map(\_.split(",")).map(attributes => Row.fromSeq(attributes))

val peopleDF = spark.createDataFrame(rowRDD, schema)

peopleDF.show()

