Session: SPARK SQL

Assignment

Student Name: Subham Vishal

Course: Big Data Hadoop & Spark Training

Assignment – Introduction to Spark SQL.

Contents

Introduction

In this assignment, we are going to perform SPARK SQL concepts.

Dataset

```
[acadgild@localhost hadoop]$ cat Sports_data.txt
firstname,lastname,sports,medal_type,age,year,country
lisa,cudrow,javellin,gold,34,2015,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2016,USA
usha,pt,running,silver,30,2016,IND
serena,williams,running,gold,31,2014,FRA
roger,federer,tennis,silver,32,2016,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2016,CHN
lisa,cudrow,javellin,gold,34,2017,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2017,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
fernando,johnson,swimming,silver,32,2017,CHN[acadgild@lo
```

Problem Statement

Using spark-sql, Find:

1. What are the total number of gold medal winners every year
2. How many silver medals have been won by USA in each sport

xSpark SQL is a Spark module for structured data processing. A **DataFrame** is a **Dataset** organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python. **DataFrames** can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs.

A row in **DataFrame** is represented by Row object. Row can be used to create a row object by using named arguments, the fields will be sorted by names.

The fields in it can be accessed like attributes.

In 2.0, SparkSession is the entry point for creation of DataFrames.

Task – 1 - What are the total number of gold medal winners every year

Please see the codes used below,

1. ***val SportsData = sc.textFile("/home/acadgild/hadoop/Sports_data.txt")***
2. ***val schemaString =***

   "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string"
3. ***val schema = StructType(schemaString.split(",").map(x => StructField(x.split(":")(0),if(x.split(":")(1).equals("string"))StringType else IntegerType, true)))***
4. ***val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0), r(1), r(2), r(3), r(4), r(5), r(6)))***
5. ***val SportsDataDF = spark.createDataFrame(rowRDD, schema)***
6. ***SportsDataDF.createOrReplaceTempView("SportsData")***
7. ***val resultDF = spark.sql("SELECT year,COUNT (*) FROM SportsData WHERE medal_type = 'gold' GROUP BY year")***
8. ***resultDF.show()***

We will proceed with the tasks,

In order to proceed we need to import some dependencies as shown below,

- ***import org.apache.spark.sql.Row;***
- ***import org.apache.spark.sql.types.{StructType,StructField,StringType,NumericType,IntegerType};***

```
scala> import org.apache.spark.sql.Row;
import org.apache.spark.sql.Row

scala> import org.apache.spark.sql.types.{StructType,StructField,StringType,NumericType,IntegerType};
import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType}
```

Step -1 – we are creating a RDD from Input DataSet,

```
scala> val SportsData = sc.textFile("/home/acadgild/hadoop/Sports_data.txt")
18/01/11 16:52:56 WARN SizeEstimator: Failed to check whether UseCompressedOops is set; assuming yes
SportsData: org.apache.spark.rdd.RDD[String] = /home/acadgild/hadoop/Sports_data.txt MapPartitionsRDD[1] at textFile at <console>:26

scala> SportsData.foreach(println)
firstname,lastname,sports,medal_type,age,year,country
lisa,cudrow,javellin,gold,34,2015,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2016,USA
usha,pt,running,silver,30,2016,IND
serena,williams,running,gold,31,2014,FRA
roger,federer,tennis,silver,32,2016,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2016,CHN
lisa,cudrow,javellin,gold,34,2017,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2017,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
fernando,johnson,swimming,silver,32,2017,CHN
```

Step -2 – we are defining a schema since it is a text file and splitting the input file using the delimiters and extracting the rows from it.

```
scala> val schemaString = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string"
schemaString: String = firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string

scala> val schema = StructType(schemaString.split(",").map(x => StructField(x.split(":")(0),if(x.split(":")(1).equals("string"))StringType else
 IntegerType, true)))
schema: org.apache.spark.sql.types.StructType = StructType(StructField(firstname,StringType,true), StructField(lastname,StringType,true), Struc
tField(sports,StringType,true), StructField(medal_type,StringType,true), StructField(age,StringType,true), StructField(year,StringType,true), S
tructField(country,StringType,true))
```

```
scala> val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0), r(1), r(2), r(3), r(4), r(5), r(6)))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[3] at map at <console>:28

scala> rowRDD.foreach(println)
[firstname,lastname,sports,medal_type,age,year,country]
[lisa,cudrow,javellin,gold,34,2015,USA]
[mathew,louis,javellin,gold,34,2015,RUS]
[michael,phelps,swimming,silver,32,2016,USA]
[usha,pt,running,silver,30,2016,IND]
[serena,williams,running,gold,31,2014,FRA]
[roger,federer,tennis,silver,32,2016,CHN]
[jenifer,cox,swimming,silver,32,2014,IND]
[fernando,johnson,swimming,silver,32,2016,CHN]
[lisa,cudrow,javellin,gold,34,2017,USA]
[mathew,louis,javellin,gold,34,2015,RUS]
[michael,phelps,swimming,silver,32,2017,USA]
[usha,pt,running,silver,30,2014,IND]
[serena,williams,running,gold,31,2016,FRA]
[roger,federer,tennis,silver,32,2017,CHN]
[jenifer,cox,swimming,silver,32,2014,IND]
[fernando,johnson,swimming,silver,32,2017,CHN]
[lisa,cudrow,javellin,gold,34,2014,USA]
[mathew,louis,javellin,gold,34,2014,RUS]
[michael,phelps,swimming,silver,32,2017,USA]
[usha,pt,running,silver,30,2014,IND]
[serena,williams,running,gold,31,2016,FRA]
[roger,federer,tennis,silver,32,2014,CHN]
[jenifer,cox,swimming,silver,32,2017,IND]
[fernando,johnson,swimming,silver,32,2017,CHN]
```

We have created the **dataframe** by passing the RDD which reads the file and schema to spark session object-

The schema of the created **Dataframe** can be seen below.

```
scala> val SportsDataDF = spark.createDataFrame(rowRDD, schema)
SportsDataDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more fields]

scala> SportsDataDF.printSchema()
root
 |-- firstname: string (nullable = true)
 |-- lastname: string (nullable = true)
 |-- sports: string (nullable = true)
 |-- medal_type: string (nullable = true)
 |-- age: string (nullable = true)
 |-- year: string (nullable = true)
 |-- country: string (nullable = true)
```

Expected Result

Now, we are using the simple SQL query so that we can execute our query by applying it on the temporary table created,

```
scala> SportsDataDF.createOrReplaceTempView("SportsData")

scala> val resultDF = spark.sql("SELECT year,COUNT (*) FROM SportsData WHERE medal_type = 'gold' GROUP BY year")
resultDF: org.apache.spark.sql.DataFrame = [year: string, count(1): bigint]

scala> resultDF.show()
+----+--------+
|year|count(1)|
+----+--------+
|2016|       2|
|2017|       1|
|2014|       3|
|2015|       3|
+----+--------+
```

Task – 2 - How many silver medals have been won by USA in each sport

Here we use the same **DataFrame** to get the desired result,

We are using the simple SQL query so that we can execute our query by applying it on the temporary table created,

The code used is,

1. ***val result2DF = spark.sql("SELECT sports, COUNT (*) FROM SportsData WHERE medal_type = 'silver' and country ='USA' GROUP BY sports")***
2. ***result2DF.show()***

Expected Output

```
scala> val result2DF = spark.sql("SELECT sports, COUNT (*) FROM SportsData WHERE medal_type = 'silver' and country ='USA' GROUP BY sports")
result2DF: org.apache.spark.sql.DataFrame = [sports: string, count(1): bigint]

scala> result2DF.show()
+--------+--------+
|  sports|count(1)|
+--------+--------+
|swimming|       3|
+--------+--------+
```