Session 21 – Spark SQL - II

Assignment 1

The associated data file is copied to HDFS and loaded as RDD. A data frame is created from the RDD.

Copying onto HDFS:

hadoop fs -put Sports_data.txt '/user/acadgild/hadoop/'

An rdd is created from the text file from HDFS.

```
scala> val sportsrdd = sc.textFile("/user/acadgild/hadoop/Sports_data.txt")
sportsrdd: org.apache.spark.rdd.RDD[String] = /user/acadgild/hadoop/Sports_data.txt MapPartitionsRDD[7] at textFile at <conso
le>:27

scala> val headers = sportsrdd.first
headers: String = firstname,lastname,sports,medal_type,age,year,country

scala> import org.apache.spark.sql.types.StructType
import org.apache.spark.sql.types.StructType

scala> import org.apache.spark.sql.types.{StructField,StringType}
import org.apache.spark.sql.types.{StructField, StringType}

scala> val fs = headers.split(",").map(f => StructField(f,StringType))
fs: Array[org.apache.spark.sql.types.StructField] = Array(StructField(firstname,StringType,true), StructField(lastname,String
Type,true), StructField(sports,StringType,true), StructField(medal_type,StringType,true), StructField(age,StringType,true), S
tructField(year,StringType,true), StructField(country,StringType,true))

scala> val schema = StructType(fs)
schema: org.apache.spark.sql.types.StructType = StructType(StructField(firstname,StringType,true), StructField(lastname,Strin
gType,true), StructField(sports,StringType,true), StructField(medal_type,StringType,true), StructField(age,StringType,true),
StructField(year,StringType,true), StructField(country,StringType,true))

scala> val noheaders = sportsrdd.filter(_!=headers)
noheaders: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[8] at filter at <console>:33

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

scala> val rows = noheaders.map( .split(",")).map(a => Row.fromSeq(a))
rows: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[10] at map at <console>:36

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

scala> sportsdf.registerTempTable("Sports")
warning: there was one deprecation warning; re-run with -deprecation for details

scala>
```

Creation of RDD from text file.

val sportsrdd = sc.textFile("/user/acadgild/hadoop/Sports_data.txt")

Extract the first record from the RDD as they constitute the headers.

val headers = sportsrdd.first

For creating the schema, 2 imports are to be done.

import org.apache.spark.sql.types.StructType
import org.apache.spark.sql.types.{StructField,StringType}

The header is split and a Schema is created out of it.

val fs = headers.split(",").map(f => StructField(f,StringType))
val schema = StructType(fs)

Apart from the headers, all the other rows in the rdd are taken as data for the dataframe.

val noheaders = sportsrdd.filter(_ !=headers)
import org.apache.spark.sql.Row
val rows = noheaders.map(_.split(",")).map(a => Row.fromSeq(a))

Creation of dataframe from the rdd.

val sportsdf = spark.createDataFrame(rows,schema)

Register the dataframe as a temporary table, for ease of querying.

sportsdf.registerTempTable("Sports")

Also, initialize the sqlContext prior to registering spark functions as Spark UDFs.

```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@31cf5483

scala>
```

Task 1:

Using spark-sql, Find:

1) What are the total number of gold medal winners every year?

```
scala> val q11 = spark.sql("select year,count(medal_type) from Sports where medal_type = 'gold' group by year")
q11: org.apache.spark.sql.DataFrame = [year: string, count(medal_type): bigint]

scala> q11.show
+----+-----------------+
|year|count(medal_type)|
+----+-----------------+
|2016|                2|
|2017|                1|
|2014|                3|
|2015|                3|
+----+-----------------+
```

The temporary table ‘Sports’ is used for querying.

val q11 = spark.sql("select year,count(medal_type) from Sports where medal_type = 'gold' group by year")

To display the data from the resultant dataframe.

q11.show

The number of gold medals per year are displayed.

2) How many silver medals have been won by USA in each sport?

```
scala> val q12 = spark.sql("select country,sports,count(medal_type) from Sports where country like 'USA' and medal_type = 'si
lver' group by sports,country")
q12: org.apache.spark.sql.DataFrame = [country: string, sports: string ... 1 more field]

scala> q12.show
+-------+--------+-----------------+
|country|  sports|count(medal_type)|
+-------+--------+-----------------+
|    USA|swimming|                3|
+-------+--------+-----------------+
```

The temporary table 'Sports' is used for querying.

val q12 = spark.sql("select country,sports,count(medal_type) from Sports where country like 'USA' and medal_type = 'silver' group by sports,country")

The resultant dataframe is displayed as below.

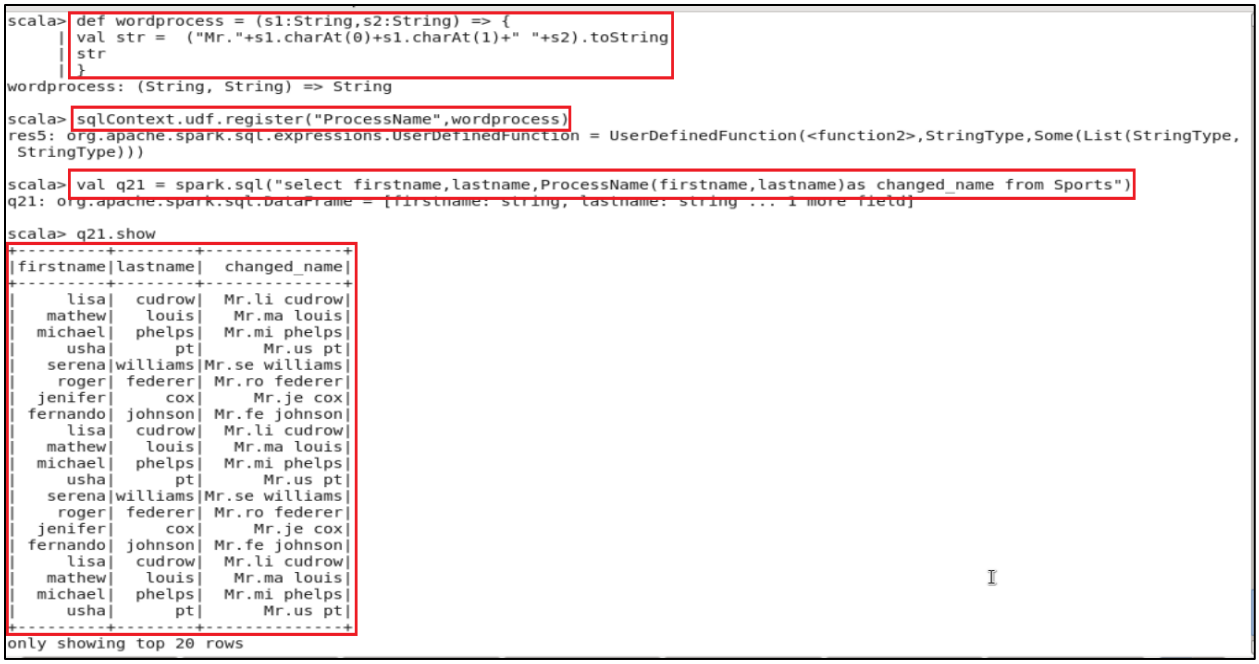
q12.show

USA has won 3 silver medals in the sport of swimming.

Task 2:

Using udfs on dataframe

1) Change firstname, lastname columns into Mr.first_two_letters_of_firstname <space> lastname
   for example - michael, phelps becomes Mr.mi phelps



Creating a function Scala for processing the first name and last name.

def wordprocess = (s1:String,s2:String) => {
val str = ("Mr."+s1.charAt(0)+s1.charAt(1)+" "+s2).toString
str
}

Registering this scala function as a spark UDF.

sqlContext.udf.register("ProcessName",wordprocess)

Now using this UDF as a part of query on the Sports temporary table.

val q21 = spark.sql("select ProcessName(firstname,lastname) from Sports")

The first name, last name and the processed name is displayed in the results.

q21.show

2) Add a new column called ranking using udfs on dataframe, where :
gold medalist, with age >= 32 are ranked as pro
gold medalists, with age <= 31 are ranked amateur
silver medalist, with age >= 32 are ranked as expert
silver medalists, with age <= 31 are ranked rookie

```
scala> def ranking = (medal:String,age:Int) => {
     | var rank =""
     | if(medal.equals("gold")) {
     | if(age>=32) rank="pro" else rank="amateur"
     | }
     | if(medal.equals("silver")){
     | if(age>=32) rank="expert" else rank="rookie"
     | }
     | rank
     | }
ranking: (String, Int) => String

scala> sqlContext.udf.register("Ranking",ranking)
res7: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType,
 IntegerType)))

scala> val q22 = spark.sql("select *,Ranking(medal_type,age) as Ranking from Sports")
q22: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 6 more fields]

scala> q22.show
+---------+--------+--------+----------+---+----+-------+-------+
|firstname|lastname|  sports|medal_type|age|year|country|Ranking|
+---------+--------+--------+----------+---+----+-------+-------+
|     lisa|  cudrow|javellin|      gold| 34|2015|    USA|    pro|
|   mathew|   louis|javellin|      gold| 34|2015|    RUS|    pro|
|  michael|  phelps|swimming|    silver| 32|2016|    USA| expert|
|     usha|      pt| running|    silver| 30|2016|    IND| rookie|
|   serena|williams| running|      gold| 31|2014|    FRA|amateur|
|    roger| federer|  tennis|    silver| 32|2016|    CHN| expert|
|  jenifer|     cox|swimming|    silver| 32|2014|    IND| expert|
| fernando| johnson|swimming|    silver| 32|2016|    CHN| expert|
|     lisa|  cudrow|javellin|      gold| 34|2017|    USA|    pro|
|   mathew|   louis|javellin|      gold| 34|2015|    RUS|    pro|
|  michael|  phelps|swimming|    silver| 32|2017|    USA| expert|
|     usha|      pt| running|    silver| 30|2014|    IND| rookie|
|   serena|williams| running|      gold| 31|2016|    FRA|amateur|
|    roger| federer|  tennis|    silver| 32|2017|    CHN| expert|
|  jenifer|     cox|swimming|    silver| 32|2014|    IND| expert|
| fernando| johnson|swimming|    silver| 32|2017|    CHN| expert|
```

Creating a scala UDF for giving the ranking based on age and the medal achieved.

```
def ranking = (medal:String,age:Int) => {
    var rank =""
    if(medal.equals("gold")) {
    if(age>=32) rank="pro" else rank="amateur"
    }
    if(medal.equals("silver")){
    if(age>=32) rank="expert" else rank="rookie"
    }
    rank
    }
```

Registering this scala function as a spark UDF.

```
sqlContext.udf.register("Ranking",ranking)
```

Now using this UDF as a part of query on the Sports temporary table.

```
val q22 = spark.sql("select *,Ranking(medal_type,age) as Ranking from Sports")
```

In the resultant dataframe, a new column ‘Ranking’ is added based on the registered UDF.