Big data Hadoop – Assignment 7 –Spark SQL 2

Task 4

Dataset link

https://drive.google.com/open?id=1qIorA_mC6h4bruPtNOX_S44bPw4rb1Sa

Using spark-sql, Find:

1. **What are the total number of gold medal winners every year**

Step 1 : Added main function and created the spark object as below

```
Logger.getLogger("org").setLevel(Level.ERROR)
//Let us create a spark session object

//Let us create a spark session object

//Create a case class globally to be used inside the main method

val spark = SparkSession

  .builder()

  .master("local")

  .appName("Spark SQL Assignment 20")

  .config("spark.some.config.option", "some-value")

  .getOrCreate()


println("spark session object is created")
```

Step 2 : We will be using below dataset for this assignment

a. Sports_Data.txt
b. Columns are – firstname,lastname,sports,medal_type,age,year,country

File Edit Format View Help

```
firstname,lastname,sports,medal_type,age,year,cou
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
```

Step 3 : To Complete the assignment first we have to load the data from these local file to Dataframe in Spark SQL as below

a. Created the case class to map the details in Dataframe from text file

i. Case Class for Sports Data

```
//Case class to hold Sports Data

case class Sports_Data (firstname:String, lastname:String, sports:String, medal_type:String, age:Int, year:Int, cou
```

Step 4 : Load data from above created RDD in dataframe

a. Before doing that we have to remove the header present in the sports data RDD. This will be achieved by using first method and get all the rows not same as the header
b. Then DF is created from above dataset as below

Big data Hadoop – Assignment 7 –Spark SQL 2

```
//Read the Holiday Details from Local file

val data = spark.sparkContext.textFile("E:/medaldataset.txt")
import spark.implicits._

//Remove Header

val header = data.first()

//Create Holdiays DF

val SportsDF = data.filter(row => row != header).map( .split(","))

  .map(x => Sports_Data(firstname = x(0), lastname = x(1), sports = x(2), medal_type = x(3), age = x(4).toInt,
      year = x(5).toInt, country = x(6))).toDF()

//Printing each row of Sports DF

SportsDF.show()
```

OUTPUT :

```
        at com.rddtest7.spark5q12.main(spark5q12.scala)
spark session object is created
+---------+--------+--------+----------+---+----+-------+
|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|
+---------+--------+--------+----------+---+----+-------+
only showing top 20 rows
```

Task 1.1 What are the total number of gold medal winners every year

Solution Approach -

1. We have query the Sports Dataframe where medal_type is gold and group on year.
2. This will be achieved using filter , groupby and count operations of the Spark SQL.

Big data Hadoop – Assignment 7 –Spark SQL 2

Approach 1: Using SPARK SQL Operations -> Filter , GroupBy and Count

```
//Task 1.1 : What are the total number of gold medal winners every year

//Need to group on year where medal type if gold


//Approach 1: Using Spark SQL Operations

SportsDF.filter("medal_type='gold'").groupBy("year").count().orderBy("year").show()
```

OUTPUT :

```
+----+-----+
|year|count|
+----+-----+
|2014|    3|
|2015|    3|
|2016|    2|
|2017|    1|
+----+-----+
```

Approach 2: Using SQL Queries

```
//Approach 2: Using SQL Query

SportsDF.createOrReplaceTempView("Sports_Table")

spark.sql("Select year,count(year) as Winners from Sports_Table where medal_type='gold' group by year order by yea
```

OUTPUT :

```
+----+-------+
|year|Winners|
+----+-------+
|2014|      3|
|2015|      3|
|2016|      2|
|2017|      1|
+----+-------+
```

Task 1.2 How many silver medals have been won by USA in each sports ?

Solution Approach **-**

1. We have query the Sports Dataframe where country is USA , medal_type=’silver’ and group on sports.

Approach 1: Using SPARK SQL Operations ->Filter , GroupBy and Count

```
//Task 1.2 How many silver medals have been won by USA in each sport

//Need to group on sports where country is USA and medal_type is silver
//Approach 1 : Using Spark SQL operations

SportsDF.filter("country='USA' and medal_type='silver'").groupBy("sports").count().show()
```

OUTPUT :

```
+--------+-----+
|  sports|count|
+--------+-----+
|swimming|    3|
+--------+-----+
```

Approach 2: Using SQL Queries

```
//Approach 2: Using SQL Query
spark.sql("Select sports,count(sports) as Winners from Sports_Table where medal_type='silver' and country='USA' group by sports").show()
```

OUTPUT :

```
+--------+-------+
|  sports|Winners|
+--------+-------+
|swimming|      3|
+--------+-------+
```

Task 5 :

Task 5.1 : 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

UDFs in Spark SQL:
User-Defined Functions (aka UDF) is a feature of Spark SQL to define new Column-based functions that extend the vocabulary of Spark SQL's DSL for transforming Datasets.

Below are steps to create udfs in the Spark SQL

Step 1 : First we have to import namespace 'org.apache.spark.sql.functions.udf' to extend the functionality / write the udfs.

```
import org.apache.spark._
import org.apache.spark.SparkContext._
import org.apache.spark.sql._
import org.apache.log4j._

import org.apache.spark.sql.functions.udf
```

Step 2 : Define a basic function scala which we would like perform the required functionality mentioned in task above. Here the function named as 'Name' is defined to accept two arguments first name and last name and returns the string output as asked.

```
 //Task 2.1 :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
//write a basic function in scala
def Name=(fname: String, lname: String)=>{
  var newName:String=null
  if (fname != null && lname != null) {
    newName="Mr.".concat(fname.substring(0, 2)).concat(" ")concat(lname)
  }
  newName
}
```

Step 3 : Once a basic function is created in scala we have can call this newly add method in Spark SQL as udf in two ways

Approach 1: Create udf for above function in scala and use it with SPARK SQL Operations

```
//first we have to create a UDF which returns the output as mentioned in above use case
//Writing the UDF
val Change_Name = udf(Name(_:String,_:String))

//Approach 1 : For calling the Custom user define function without registering
SportsDF.withColumn("Name", Change_Name($"firstname", $"lastname")).show()
```

OUTPUT :

```
+---------+--------+--------+-----------+---+----+-------+--------------+
|firstname|lastname|  sports|medal_type|age|year|country|          Name|
+---------+--------+--------+-----------+---+----+-------+--------------+
|     lisa|  cudrow|javellin|       gold| 34|2015|    USA|  Mr.li cudrow|
|   mathew|   louis|javellin|       gold| 34|2015|    RUS|   Mr.ma louis|
|  michael|  phelps|swimming|     silver| 32|2016|    USA|  Mr.mi phelps|
|     usha|      pt| running|     silver| 30|2016|    IND|      Mr.us pt|
|   serena|williams| running|       gold| 31|2014|    FRA|Mr.se williams|
|    roger| federer|  tennis|     silver| 32|2016|    CHN| Mr.ro federer|
|  jenifer|     cox|swimming|     silver| 32|2014|    IND|     Mr.je cox|
| fernando| johnson|swimming|     silver| 32|2016|    CHN| Mr.fe johnson|
|     lisa|  cudrow|javellin|       gold| 34|2017|    USA|  Mr.li cudrow|
|   mathew|   louis|javellin|       gold| 34|2015|    RUS|   Mr.ma louis|
|  michael|  phelps|swimming|     silver| 32|2017|    USA|  Mr.mi phelps|
|     usha|      pt| running|     silver| 30|2014|    IND|      Mr.us pt|
|   serena|williams| running|       gold| 31|2016|    FRA|Mr.se williams|
|    roger| federer|  tennis|     silver| 32|2017|    CHN| Mr.ro federer|
|  jenifer|     cox|swimming|     silver| 32|2014|    IND|     Mr.je cox|
| fernando| johnson|swimming|     silver| 32|2017|    CHN| Mr.fe johnson|
|     lisa|  cudrow|javellin|       gold| 34|2014|    USA|  Mr.li cudrow|
|   mathew|   louis|javellin|       gold| 34|2014|    RUS|   Mr.ma louis|
|  michael|  phelps|swimming|     silver| 32|2017|    USA|  Mr.mi phelps|
|     usha|      pt| running|     silver| 30|2014|    IND|      Mr.us pt|
+---------+--------+--------+-----------+---+----+-------+--------------+
only showing top 20 rows
```

Approach 2: By registering the udf so that it can be used wih sql queries

```
//Approach 2: By registering the function

spark.sqlContext.udf.register("Name", Name)


spark.sql("Select Name(firstname,lastname) as changed_Name, sports,medal_type,age,year,country from Sports_Table").show()
```

OUTPUT :

```
+--------------+--------+----------+---+----+-------+
|  changed_Name|  sports|medal_type|age|year|country|
+--------------+--------+----------+---+----+-------+
|  Mr.li cudrow|javellin|      gold| 34|2015|    USA|
|   Mr.ma louis|javellin|      gold| 34|2015|    RUS|
|  Mr.mi phelps|swimming|    silver| 32|2016|    USA|
|      Mr.us pt| running|    silver| 30|2016|    IND|
|Mr.se williams| running|      gold| 31|2014|    FRA|
| Mr.ro federer|  tennis|    silver| 32|2016|    CHN|
|     Mr.je cox|swimming|    silver| 32|2014|    IND|
| Mr.fe johnson|swimming|    silver| 32|2016|    CHN|
|  Mr.li cudrow|javellin|      gold| 34|2017|    USA|
|   Mr.ma louis|javellin|      gold| 34|2015|    RUS|
|  Mr.mi phelps|swimming|    silver| 32|2017|    USA|
|      Mr.us pt| running|    silver| 30|2014|    IND|
|Mr.se williams| running|      gold| 31|2016|    FRA|
| Mr.ro federer|  tennis|    silver| 32|2017|    CHN|
|     Mr.je cox|swimming|    silver| 32|2014|    IND|
| Mr.fe johnson|swimming|    silver| 32|2017|    CHN|
|  Mr.li cudrow|javellin|      gold| 34|2014|    USA|
|   Mr.ma louis|javellin|      gold| 34|2014|    RUS|
|  Mr.mi phelps|swimming|    silver| 32|2017|    USA|
|      Mr.us pt| running|    silver| 30|2014|    IND|
+--------------+--------+----------+---+----+-------+
only showing top 20 rows
```

Task 5.2 Using udfs on dataframe 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

Basic scala function to perform the required above task

Big data Hadoop – Assignment 7 –Spark SQL 2

```
//Task 2.2 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

//Write basic scala function for the required use case

def ranking_recived =(medal_type:String,age:Int)=> {

  if(medal_type.equalsIgnoreCase("gold") && age>=32) "pro"

  else if(medal_type.equalsIgnoreCase("gold") && age <=31) "amateur"

  else if(medal_type.equalsIgnoreCase("silver") && age >= 32) "amateur"

  else if(medal_type.equalsIgnoreCase("silver") && age <= 31) "amateur"

  else ""
```

Approach 1: Create udf for above function in scala and use it with SPARK SQL Operations

```
val Rankings = udf(ranking_recived(_:String,_:Int))


//Approach 1: Without Registering the UDF and calling with Spark SQL Operatios

SportsDF.withColumn("Ranking",Rankings($"medal type",$"age")).show()
```

OUTPUT :

```
+---------+--------+--------+----------+---+----+-------+-------+
|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|amateur|
|     usha|      pt| running|    silver| 30|2016|    IND|amateur|
|   serena|williams| running|      gold| 31|2014|    FRA|amateur|
|    roger| federer|  tennis|    silver| 32|2016|    CHN|amateur|
|  jenifer|     cox|swimming|    silver| 32|2014|    IND|amateur|
| fernando| johnson|swimming|    silver| 32|2016|    CHN|amateur|
|     lisa|  cudrow|javellin|      gold| 34|2017|    USA|    pro|
|   mathew|   louis|javellin|      gold| 34|2015|    RUS|    pro|
|  michael|  phelps|swimming|    silver| 32|2017|    USA|amateur|
|     usha|      pt| running|    silver| 30|2014|    IND|amateur|
|   serena|williams| running|      gold| 31|2016|    FRA|amateur|
|    roger| federer|  tennis|    silver| 32|2017|    CHN|amateur|
|  jenifer|     cox|swimming|    silver| 32|2014|    IND|amateur|
| fernando| johnson|swimming|    silver| 32|2017|    CHN|amateur|
|     lisa|  cudrow|javellin|      gold| 34|2014|    USA|    pro|
|   mathew|   louis|javellin|      gold| 34|2014|    RUS|    pro|
|  michael|  phelps|swimming|    silver| 32|2017|    USA|amateur|
|     usha|      pt| running|    silver| 30|2014|    IND|amateur|
+---------+--------+--------+----------+---+----+-------+-------+
only showing top 20 rows
```

Big data Hadoop – Assignment 7 –Spark SQL 2

Approach 2: By registering the udf so that it can be used wih sql queries

```
//Approach 2:By Registering the function

spark.sqlContext.udf.register("Rankings",ranking_recived)

spark.sql("Select Rankings(medal_type,age) as changed_Name, sports,medal_type,age,year,country from Sports_Table").show()
```

OUTPUT :

```
+------------+--------+----------+---+----+-------+
|changed_Name|  sports|medal_type|age|year|country|
+------------+--------+----------+---+----+-------+
|         pro|javellin|      gold| 34|2015|    USA|
|         pro|javellin|      gold| 34|2015|    RUS|
|     amateur|swimming|    silver| 32|2016|    USA|
|     amateur| running|    silver| 30|2016|    IND|
|     amateur| running|      gold| 31|2014|    FRA|
|     amateur|  tennis|    silver| 32|2016|    CHN|
|     amateur|swimming|    silver| 32|2014|    IND|
|     amateur|swimming|    silver| 32|2016|    CHN|
|         pro|javellin|      gold| 34|2017|    USA|
|         pro|javellin|      gold| 34|2015|    RUS|
|     amateur|swimming|    silver| 32|2017|    USA|
|     amateur| running|    silver| 30|2014|    IND|
|     amateur| running|      gold| 31|2016|    FRA|
|     amateur|  tennis|    silver| 32|2017|    CHN|
|     amateur|swimming|    silver| 32|2014|    IND|
|     amateur|swimming|    silver| 32|2017|    CHN|
|         pro|javellin|      gold| 34|2014|    USA|
|         pro|javellin|      gold| 34|2014|    RUS|
|     amateur|swimming|    silver| 32|2017|    USA|
|     amateur| running|    silver| 30|2014|    IND|
+------------+--------+----------+---+----+-------+
only showing top 20 rows
```