Spark Assignment 19.2

Given the below dataset . Solve the below mentioned problem statement in spark Sql.

19_Sports_Data.txt as a text file
(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
```

This assignment is done in the spark shell within the acadgildVM.

Steps Followed:

1) Copied the dataset file in the path /home/acadgild/spark/19_Sports_data.txt Then read the text file by using Sparksession(spark.read.textfile) as below and created a dataframe.

```
val df1= spark.read.textFile("/home/acadgild/spark/19_Sports_data.txt")
```

2) Created case class Sport_Data by specifying columns and datatypes, to describe the contents of the rows.

```
case class Sport_Data(firstname: String, lastname: String,sports: String,medal_type: String, age:Int,year:Int, country: String)
```

3) Then created drataframe df2, by using map function over the df1.

```
val df2 = df1.map(line => line.split(",")).map(rec=>Sport_Data(rec(0), rec(1), rec(2), rec(3), rec(4).toInt, rec(5).toInt, rec(6)))
```

4) Registered df2 as TempTable as below.

```
df2.registerTempTable("df2")
```

Spark-shell output

```
scala> val df1= spark.read.textFile("/home/acadgild/spark/19_Sports_data.txt")
df1: org.apache.spark.sql.Dataset[String] = [value: string]

scala> case class Sport_Data(firstname: String, lastname: String,sports: String,medal_type: String, age:Int,year:Int, country: String)
//columns and data types
defined class Sport_Data

scala> val df2 = df1.map(line => line.split(",")).map(rec=>Sport_Data(rec(0), rec(1), rec(2), rec(3), rec(4).toInt, rec(5).toInt, rec(6)
))
df2: org.apache.spark.sql.Dataset[Sport_Data] = [firstname: string, lastname: string ... 5 more fields]

scala> df2.registerTempTable("df2")
warning: there was one deprecation warning; re-run with -deprecation for details
```

5) The schema of df2 is as below

Spark-shell output

```
scala> df2.printSchema
root
 |-- firstname: string (nullable = true)
 |-- lastname: string (nullable = true)
 |-- sports: string (nullable = true)
 |-- medal_type: string (nullable = true)
 |-- age: integer (nullable = false)
 |-- year: integer (nullable = false)
 |-- country: string (nullable = true)
```

Problem Statement:

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

To change firstname and lastname columns into Mr.Firstname 2 letters space last name, imported sql.functions_, the created a UDF function as getConcatenated as

below.The function takes firstname and lastname as input and returns Mr.first_two_letters_of_firstname<space>lastname as output.

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

val getConcatenated = udf( (first: String, second: String) => {"Mr."+ first.substring(0,2) + " " + second } )
```

UDF

```
scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> val getConcatenated = udf( (first: String, second: String) => {"Mr."+ first.substring(0,2) + " " + second } )
getConcatenated: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType,
 StringType)))
```

Over df2, using withColumn function, created a new column FirstNameConcatlastName, then called the getConcatenatedfunction UDF function and passed firstname and lastname as arguments to find the value of FirstNameConcatlastName column. The output of getConcatenatedfunction will be the value of FirstNameConcatlastName column. Now using select function, displayed "FirstNameConcatlastName","sports","medal_type", "age","year".”country”,

```
val df7=df2.withColumn("FirstNameConcatlastName", getConcatenated($"firstname", $"lastname")).select("FirstNameConcatlastName","sports","medal_type", "age","year", "country").show()
```

Output

```
+----------------------+--------+----------+---+----+-------+
|FirstNameConcatlastName|  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

Spark-shell output

```
scala> val df7=df2.withColumn("FirstNameConcatlastName", getConcatenated($"firstname", $"lastname")).select("FirstNameConcatlastName","sports","medal_type", "age","year", "country").show()
+----------------------+--------+----------+---+----+-------+
|FirstNameConcatlastName|  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

df7: Unit = ()
```

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

To find the ranking for each record, imported sql.functions_, then created a UDF function as findRanking as below.Internally the UDF is calling the ranking function..The ranking function takes 2 argumets as input (age and medal_type), within which we are specifying the the problem statement condition

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

def ranking(age: Int,medal_type: String) :String={
 if (medal_type.equals("silver") && age >= 32 )
  "expert"
 else if (medal_type.equals("silver") && age <= 32 )
  "rookie"
```

```
 else if (medal_type.equals("gold") && age >= 32 )
 "pro"
  else if (medal_type.equals("gold") && age <= 32 )
 "amateur"
 else ""
}
```

```
val findRanking = udf(ranking(_:Int,_:String))
```

UDF

```
scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala>  def ranking(age: Int,medal_type: String) :String={
     |  if (medal_type.equals("silver") && age >= 32 )
     |   "expert"
     |  else if (medal_type.equals("silver") && age <= 32 )
     |   "rookie"
     |   else if (medal_type.equals("gold") && age >= 32 )
     |   "pro"
     |    else if (medal_type.equals("gold") && age <= 32 )
     |   "amateur"
     |   else ""
     |  }
ranking: (age: Int, medal_type: String)String

scala>  val findRanking = udf(ranking(_:Int,_:String))
findRanking: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(IntegerType
ringType)))
```

Over df2, using withColumn function, created a new column ranking, then called the findRanking UDF function and passed age and medal_type as arguments, to find the value of ranking column. The output of findRanking will be the value of ranking column. It is stored a dataframe in df8. Then printed the contents of df8.

```
val df8 = df2.withColumn("ranking", findRanking($"age",$"medal_type"))
```

```
df8.collect.foreach(println)
```

Output

```
scala> df8.collect.foreach(println)
[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]
[lisa,cudrow,javellin,gold,34,2014,USA,pro]
[mathew,louis,javellin,gold,34,2014,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,32,2014,CHN,expert]
[jenifer,cox,swimming,silver,32,2017,IND,expert]
[fernando,johnson,swimming,silver,32,2017,CHN,expert]

Spark-shell ouput

```
scala>  val df8 = df2.withColumn("ranking", findRanking($"age",$"medal_type"))
df8: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 6 more fields]

scala> df8.collect.foreach(println)
[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]
[lisa,cudrow,javellin,gold,34,2014,USA,pro]
[mathew,louis,javellin,gold,34,2014,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,32,2014,CHN,expert]
[jenifer,cox,swimming,silver,32,2017,IND,expert]
[fernando,johnson,swimming,silver,32,2017,CHN,expert]
```