

BigData Assignment 7.6

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

2. Add a new column called ranking using udfs on dataframe, where :

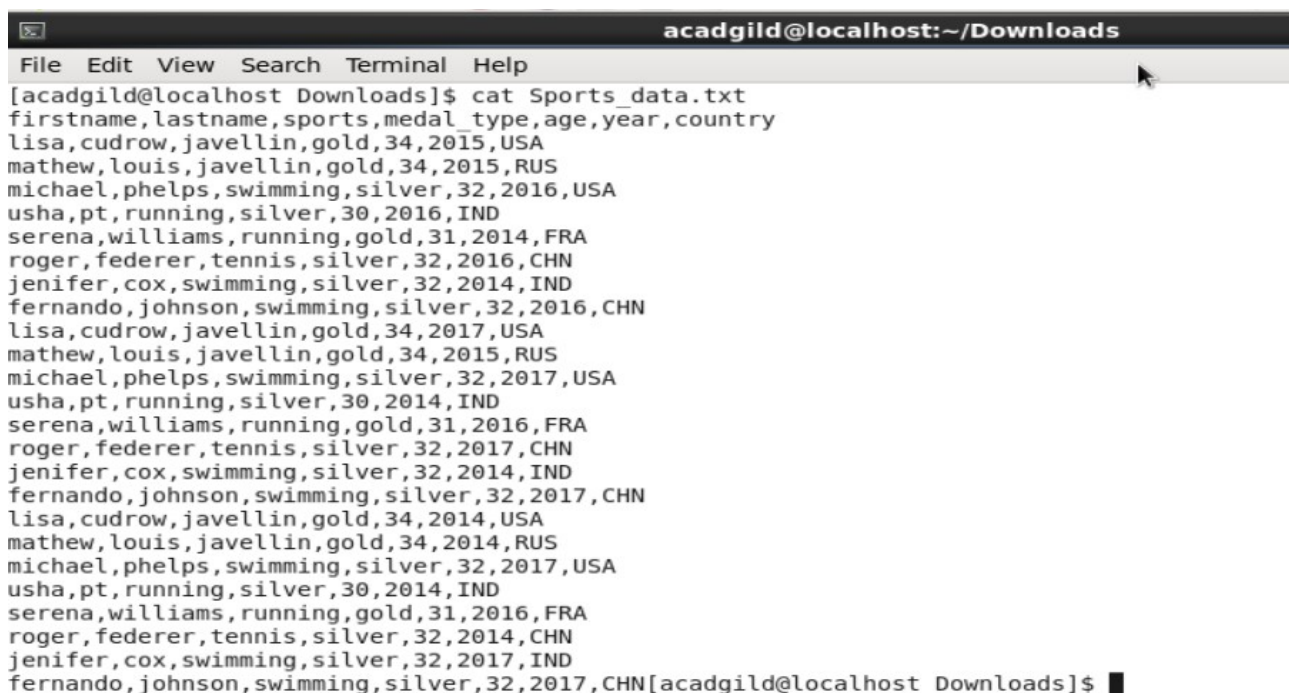
gold medalist, with age ≥ 32 are ranked as pro

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

The content of the dataset

A terminal window titled 'acadgild@localhost:~/Downloads' showing the output of the command 'cat Sports_data.txt'. The output is a list of athlete names, sports, medal types, ages, years, and countries, separated by commas. The data is repeated for two different years (2014 and 2016/2017) for each athlete.

```
acadgild@localhost:~/Downloads
File Edit View Search Terminal Help
[acadgild@localhost Downloads]$ 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@localhost Downloads]$
```

Solution -

//Read the dataset.

var sports_data =

sc.textFile("/home/acadgild/Downloads/Sports_data.txt")

//Stored the header

```
val header = sports_data.first()
```

```
//Filtered the data without header
```

```
val actual_data = sports_data.filter(row => row != header)
```

```
//Created the dataframe where the rows are splitted and converted to its appropriate type and namme is assigned to the columns.
```

```
val sportsDF = actual_data.map(x => x.split(",")).map(x => x(0),x(1),x(2),x(3),x(4).toInt,x(5).toInt,x(6))).toDF("firstname","lastname","sports","medal_type","age","year","country")
```

```
//Created a table of the dataframe
```

```
sportsDF.createOrReplaceTempView("SportsData")
```

A screenshot of a Scala REPL window titled 'acadgild@localhost:~/RITESH/7.1'. The window shows the following code being executed:

```
scala> val sports_data = sc.textFile("file:///home/acadgild/Downloads/Sports_data.txt")
sports_data: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Downloads/Sports_data.txt MapPartitionsRDD[45] at textFile at <console>:24

scala> val header = sports_data.first()
header: String = firstname,lastname,sports,medal_type,age,year,country

scala> val actual_data = sports_data.filter(row=> row!=header)
actual_data: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[46] at filter at <console>:28

scala> val sportsDF = actual_data.map(x=>x.split(",")).map(x=>(x(0),x(1),x(2),x(3),x(4).toInt,x(5).toInt,x(6))).toDF("firstname","lastname","sports","medal_type","age","year","country")
sportsDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more fields]

scala> sportsDF.createOrReplaceTempView("SportsData")

scala> █
```

1.

created udf 'name' to combine firstname and lastname. Then created a column Fullname using withColumn function to store it. Then dropped the two columns.

```
val name =
```

```
udf((fname:String,lname:String)=>{"Mr."+fname.slice(0,2)+" "+lname+""})
```

```
val result = sportsDF.withColumn("Fullname",name($"firstname", $"lastname"))
```

```
val finalDF = result.drop("firstname").drop("lastname")
```

```
finalDF.show
```

Output -

```
acadgild@localhost:~/RITESH/7.1
scala> val name = udf((fname:String, lname:String)=>{"Mr."+fname.slice(0,2)+" "+lname})
name: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>, StringType, Some(List(StringType, StringType)))

scala> val result = sportsDF.withColumn("Fullname", name($"firstname", $"lastname"))
result: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 6 more fields]

scala> val finalDF = result.drop("firstname").drop("lastname")
finalDF: org.apache.spark.sql.DataFrame = [sports: string, medal_type: string ... 4 more fields]

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

2.

Created an udf function to assign ranking as per the question . Then added the column using withColumn function.

```
val rank =
udf((medal:String,age:String)=>{if(age<=31&&medal=="silver")
{"rookie";}else{if(age.>=32&&medal=="silver")
{"expert";}else{if(age<=31&&medal=="gold")
{"amateur";}else{if(age.>=32&&medal=="gold")
{"pro";}else{"beginer";}}}}})
```

```
val finalDF=sports.withColumn("ranking",player($"medal_type",
$"age"))
```

```
finalDF.show
```

Output -

```
acadgild@localhost:~/RITESH/7.1
File Edit View Search Terminal Help

scala> val rank = udf((medal:String,age:Int)=>{if(age<=31&medal=="silver"){ "rookie";} else {if(age>=32&medal=="silver"){ "expert";}else{if(age<=31&medal=="gold"){ "amateur";}else{if(age>=32&medal=="gold"){ "pro";}else{ "beginer";}}}}})
rank: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType,IntegerType)))

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

scala> finalDF.show
+-----+-----+-----+-----+-----+-----+-----+
|firstname|lastname| sports|medal_type|age|year|country|ranking|
+-----+-----+-----+-----+-----+-----+-----+
|lisa|cudrow|javelin|gold|34|2015|USA|pro|
|matthew|louis|javelin|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|javelin|gold|34|2017|USA|pro|
|matthew|louis|javelin|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|javelin|gold|34|2014|USA|pro|
|matthew|louis|javelin|gold|34|2014|RUS|pro|
|michael|phelps|swimming|silver|32|2017|USA|expert|
|usha|pt|running|silver|30|2014|IND|rookie|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows
```