Problem Statement

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**

In order to proceed we need to define the dataframe first for the text input file we have-
Below is code which is used to find the result-

- import org.apache.spark.sql.functions.udf
- val SportsData = sc.textFile("/home/acadgild/Assignment-19/Sports_data.txt")
- val schemaString = "firstname:string,lastname:string,sports:string,medal:string,age:integer,year:integer,country:string"
- val schema = StructType(schemaString.split(",").map(x => StructField(x.split(":")(0), if (x.split(":")(1).equals("string")) StringType else IntegerType, true)))
- val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0), r(1), r(2), r(3), r(4).toInt, r(5).toInt, r(6)))
- val SportsDataDF = spark.createDataFrame(rowRDD, schema)
- SportsDataDF.createOrReplaceTempView("Sports_Data")
- val Name = udf((firstname: String, lastname: String) => "Mr. ".concat(firstname.substring(0,2)).concat(" ")concat(lastname))
- spark.udf.register("Full_Name", Name)
- val fname = spark.sql("SELECT Full_Name(firstname, lastname) FROM Sports_Data")
- fname.show()

Now lets see each and every line one by one.
In order to proceed we need to import some dependencies as shown below-

```
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}

scala>
```

Now we are creating a RDD which reads from the input file-

```
scala> val SportsData = sc.textFile("/home/acadgild/Assignment-19/Sports_data.txt")
SportsData: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-19/Sports_data.txt MapPartitionsRDD[3] at textFile at <console>:26

scala> SportsData.foreach(println)
[Stage 0:>                                                          (0 + 0) / 2]roger,federer,tennis,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2015,USA
jenifer,cox,swimming,silver,32,2014,IND
mathew,louis,javellin,gold,34,2015,RUS
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
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
```

Since it is a text file we need to define schema too. Below screenshot shows the same-

```
scala> val schemaString = "firstname:string,lastname:string,sports:string,medal:string,age:integer,year:integer,country:string"
schemaString: String = firstname:string,lastname:string,sports:string,medal:string,age:integer,year:integer,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), StructField(sports,StringTy
pe,true), StructField(medal,StringType,true), StructField(age,IntegerType,true), StructField(year,IntegerType,true), StructField(country,StringType,true))
```

Now we are splitting the input file and extracting the rows from it-

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

scala> rowRDD.foreach(println)
[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]
```

Now we are creating the dataframe by passing the RDD which reads the file and schema to spark session object-

```
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: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- year: integer (nullable = true)
 |-- country: string (nullable = true)
```

Here we are defining the UDF which will take 2 strings (columns) as input and will concatenate them with Mr. appended in it-

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

scala> val Name = udf((firstname: String, lastname: String) => "Mr. ".concat(firstname.substring(0,2)).concat(" ")concat(lastname))
Name: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, StringType)))
```

Now we need to register the UDF. Here we doing the same and giving it an alias as Full_Name.

Finally we can apply this UDF on the columns to give the required result-

```
scala> spark.udf.register("Full_Name", Name)
res15: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, StringType)))

scala> val fname = spark.sql("SELECT Full_Name(firstname, lastname) FROM Sports_Data")
fname: org.apache.spark.sql.DataFrame = [UDF(firstname, lastname): string]

scala> fname.show()
+-----------------------+
|UDF(firstname, lastname)|
+-----------------------+
|           Mr. li cudrow|
|            Mr. ma louis|
|           Mr. mi phelps|
|               Mr. us pt|
|         Mr. se williams|
|           Mr. ro federer|
|              Mr. je cox|
|           Mr. fe johnson|
|           Mr. li cudrow|
|            Mr. ma louis|
|           Mr. mi phelps|
|               Mr. us pt|
|         Mr. se williams|
|           Mr. ro federer|
|              Mr. je cox|
|           Mr. fe johnson|
|           Mr. li cudrow|
|            Mr. ma louis|
|           Mr. mi phelps|
|               Mr. us pt|
+-----------------------+
```

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

Here we will work with the dataframe created in above problem. We just need to write an UDF. Below is the UDF that we have used to define the new column-

- val Rank = udf((medal: String, age: Int) => (medal, age) match {
  case (medal,age) if medal == "gold" && age >= 32 => "Pro"
  case (medal,age) if medal == "gold" && age <= 31 => "Amateur"
  case (medal,age) if medal == "silver" && age >= 32 => "Expert"
  case (medal,age) if medal == "silver" && age <= 31 => "Rookie"
- })

Here we are classifying each player based on age and the medal he has got-

```
scala> val Rank = udf((medal: String, age: Int) => (medal, age) match {
     | case (medal,age) if medal == "gold" && age >= 32 => "Pro"
     | case (medal,age) if medal == "gold" && age <= 31 => "Amateur"
     | case (medal,age) if medal == "silver" && age >= 32 => "Expert"
     | case (medal,age) if medal == "silver" && age <= 31 => "Rookie"
     | })
Rank: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, IntegerType)))

scala> spark.udf.register("Ranking", Rank)
res18: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, IntegerType)))

scala>
```

Below code shows the registering of UDF and command to add a new column-

- spark.udf.register("Ranking", Rank)
- val RankRDD = SportsDataDF.withColumn("Ranking",Rank(SportsDataDF.col("medal"),SportsDataDF.col("age")))

Below shows the final result for same-

```
scala> val RankRDD = SportsDataDF.withColumn("Ranks", Rank(SportsDataDF.col("medal"),SportsDataDF.col("age")))
RankRDD: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 6 more fields]

scala> RankRDD.show()
+---------+--------+--------+------+---+----+-------+-------+
|firstname|lastname|  sports| medal|age|year|country|  Ranks|
+---------+--------+--------+------+---+----+-------+-------+
|     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|
+---------+--------+--------+------+---+----+-------+-------+
```