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

Create a case class Sports

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

Create RDD from the file

val sportsRDD = sc.textFile("/home/acadgild/inputdir/Sports_data.txt")

val firstline = sportsRDD.first

Filter out the first line from RDD as it contains headers

val dataRDD = sportsRDD.filter(line=>(line!=firstline)).map(line=>line.split(","))

map the RDD with case class Sports

val datamapRDD = dataRDD.map(s=>Sports(s(0),s(1),s(2),s(3),s(4).toInt,s(5).toInt,s(6)))

Convert to dataframe

val sportsDF = datamapRDD.toDF

Register Dataframe as Temp Table

sportsDF.registerTempTable("sports")

Create UDF for Concatenation:

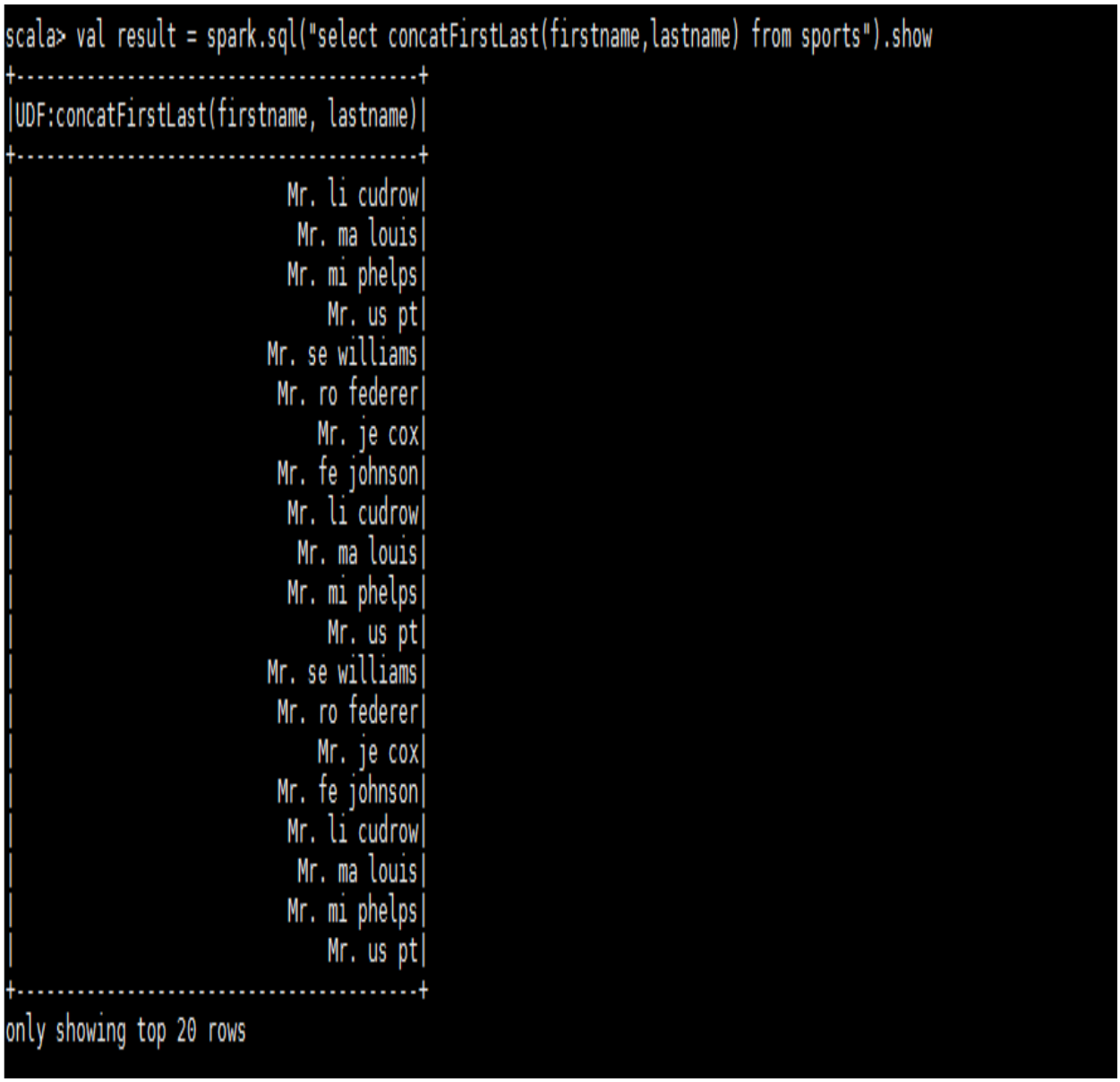
```
def concatFirstLast(firstname:String,lastname:String):String=
      {
            "Mr. "+firstname.substring(0,2)+" "+lastname
      }
```

Register the UDF:

spark.sqlContext.udf.register("concatFirstLast",concatFirstLast _)

Run the query:

val result = spark.sql("select concatFirstLast(firstname,lastname) from sports").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

Create UDF for Ranking:

```
def ranking(medaltype:String,age:Int):String=
	{
		if(medaltype=="gold" && age>=32)
		{
			"Pro"
		}
		else if(medaltype=="gold" && age<=31)
		{
			"amateur"
		}
		else if(medaltype=="silver" && age>=32)
		{
			"expert"
		}
		else if(medaltype=="silver" && age<=31)
		{
			"rookie"
		}
		else
		{
			"default"
		}
	}
```

Register the UDF:

```
spark.sqlContext.udf.register("ranking", ranking _)
```

Run the query:

val result = spark.sql("select sports.*,ranking(medal_type,age) rank from sports").show

```
scala> spark.sqlContext.udf.register("ranking", ranking _)
res2: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, IntegerType
)))

scala> val result = spark.sql("select sports.*,ranking(medal_type,age) rank  from sports").show
+---------+--------+--------+----------+---+----+-------+-------+
|firstname|lastname|  sports|medal_type|age|year|country|   rank|
+---------+--------+--------+----------+---+----+-------+-------+
|     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|
+---------+--------+--------+----------+---+----+-------+-------+
only showing top 20 rows

result: Unit = ()
```