Assignment 21

Task 1

Using spark-sql, Find:

Data set used,

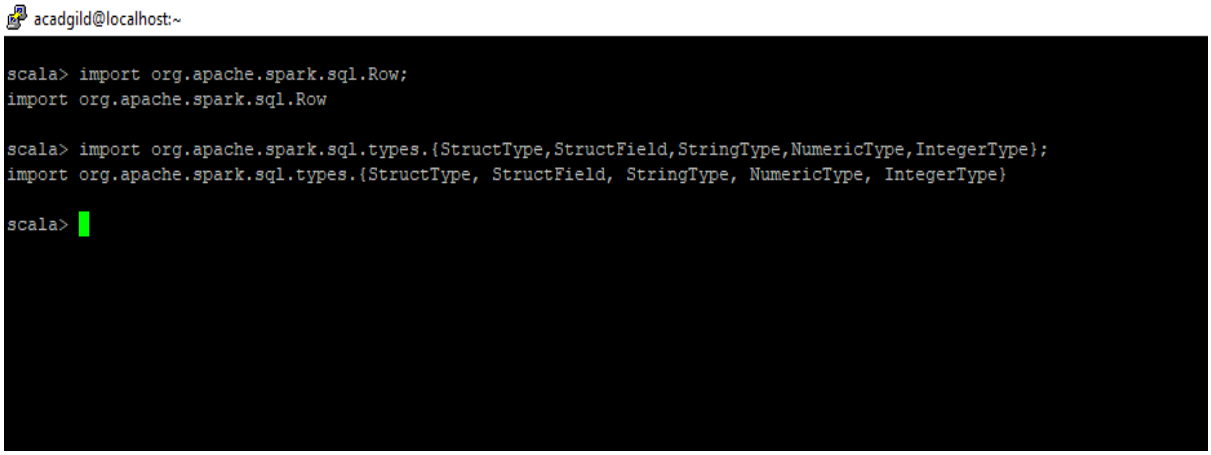
Sports_data.txt

acadgild@localhost:~

```
[acadgild@localhost ~]$ 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
^C
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -put Sports_data.txt
18/04/09 21:02:43 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
[acadgild@localhost ~]$
```

We will proceed with the tasks,
In order to proceed we need to import some dependencies as shown below,

import org.apache.spark.sql.Row;
import org.apache.spark.sql.types.{StructType,StructField,StringType,NumericType,IntegerType};



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

// Create an RDD

val sportsData = sc.textFile("/user/acadgild/Sports_data.txt")

// The schema is encoded in a string

val schemaString = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string"

// Generate the schema based on the string of schema

val schema = StructType(schemaString.split(",").map(x=>StructField(x.split(":")(0),if(x.split(":")(1).equals("string"))StringType else IntegerType,true)))

// Convert records of the RDD (sportsData) to Rows

val rowRDD = sportsData.map(_.split(",")).map(r=>Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))

// Apply the schema to the RDD

val sportsDataDF=spark.createDataFrame(rowRDD,schema)

// Creates a temporary view using the DataFrame

sportsDataDF.createOrReplaceTempView("sportsData")

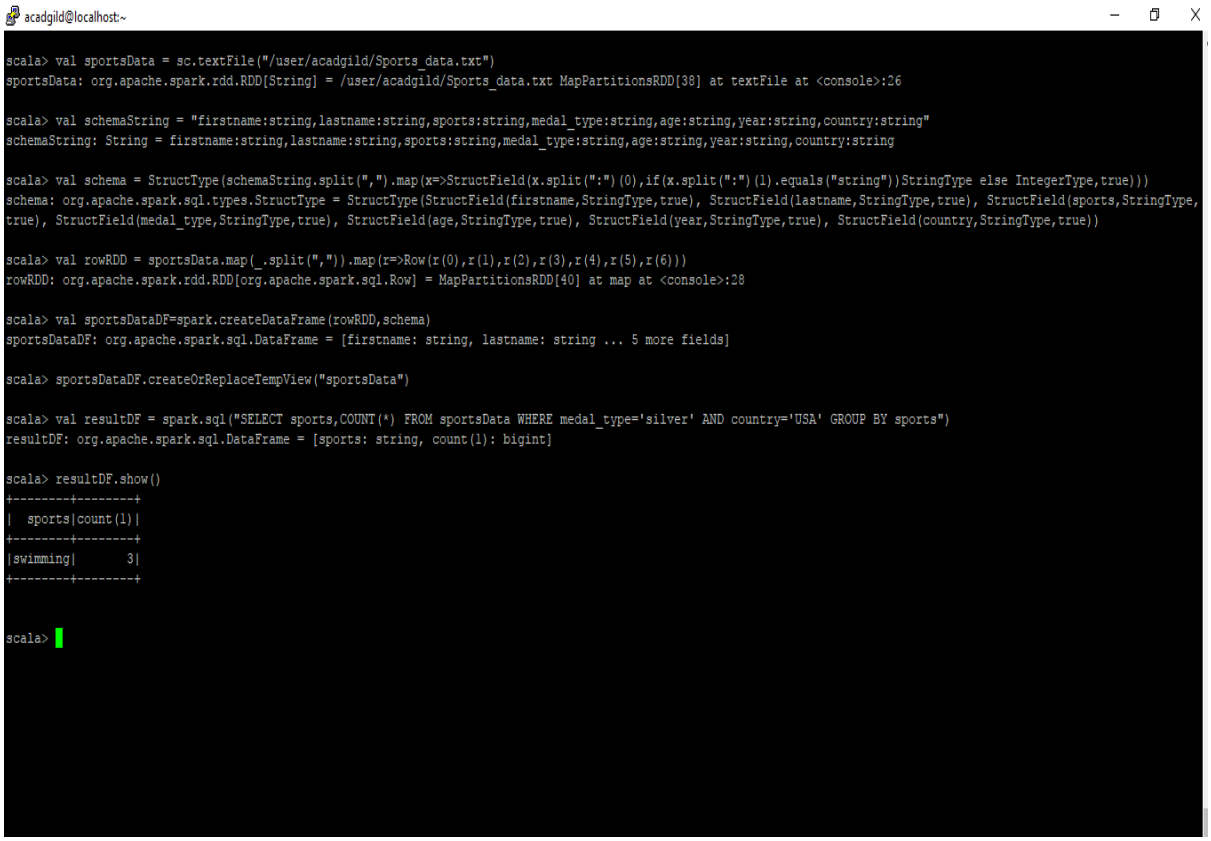
// SQL can be run over a temporary view created using DataFrames

val resultDF = spark.sql("SELECT year,COUNT(*) FROM sportsData WHERE medal_type='gold' GROUP BY year")

resultDF.show()

2. How many silver medals have been won by USA in each sport

```
// Create an RDD
val sportsData = sc.textFile("/user/acadgild/Sports_data.txt")

// The schema is encoded in a string
 val schemaString = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string"

// Generate the schema based on the string of schema
 val schema = StructType(schemaString.split(",").map(x=>StructField(x.split(":")(0),if(x.split(":")(1).equals("string"))StringType else IntegerType,true)))

// Convert records of the RDD (sportsData) to Rows
val rowRDD = sportsData.map(_.split(",")).map(r=>Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))

// Apply the schema to the RDD
val sportsDataDF=spark.createDataFrame(rowRDD,schema)

// Creates a temporary view using the DataFrame
sportsDataDF.createOrReplaceTempView("sportsData")

// SQL can be run over a temporary view created using DataFrames
val resultDF = spark.sql("SELECT sports,COUNT(*) FROM sportsData WHERE medal_type='silver' AND country='USA' GROUP BY sports")
resultDF.show()
```



Task 2

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**

val sportsData = sc.textFile("/user/acadgild/Sports_data.txt")

val schemaString = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,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),r(5),r(6)))

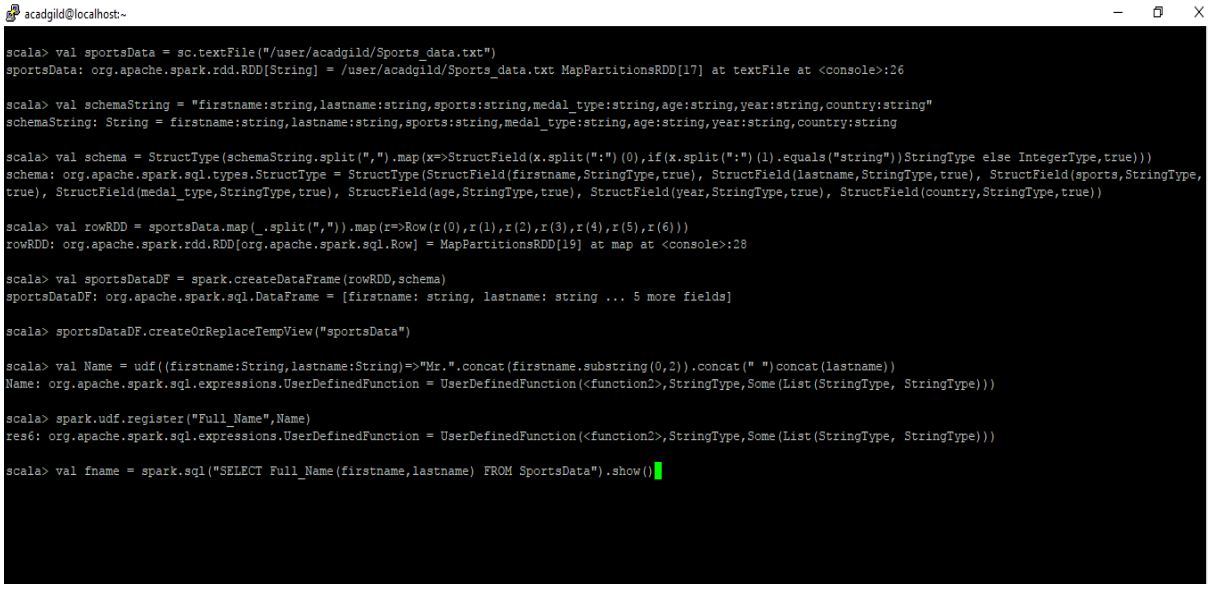
val sportsDataDF = spark.createDataFrame(rowRDD,schema)

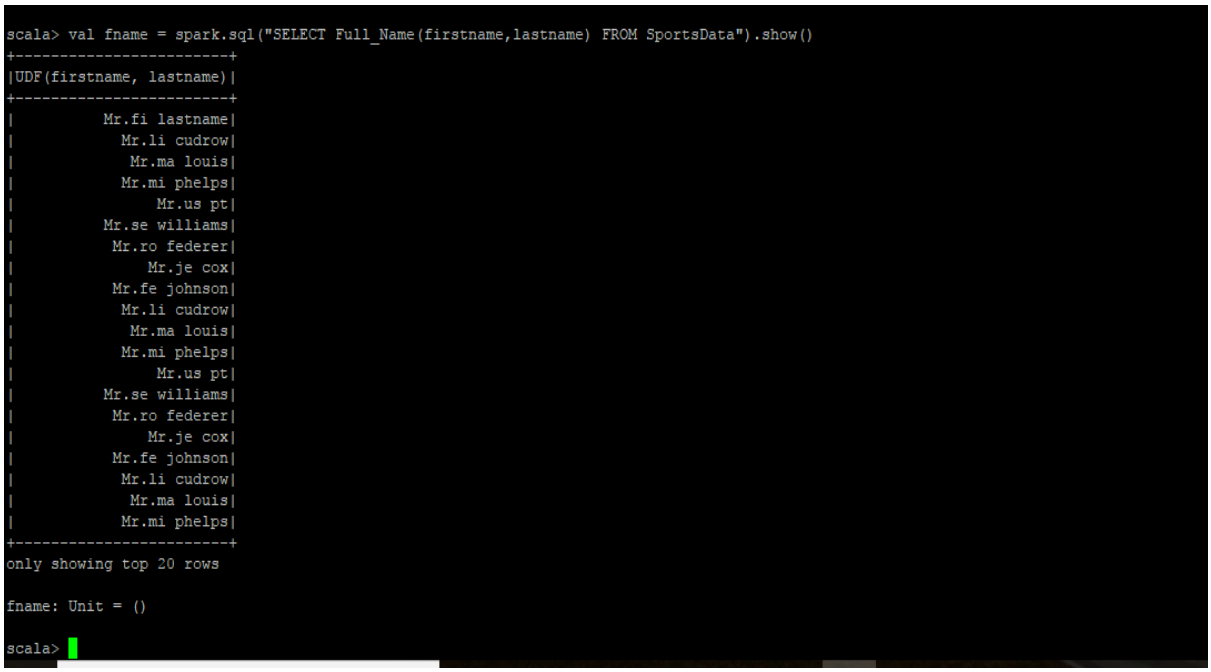
sportsDataDF.createOrReplaceTempView("sportsData")

val Name = udf((firstname:String,lastname:String)=>"Mr.".concat(firstname.substring(0,2)).concat(" ")concat(lastname))

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

val RankingRDD = spark.sql("SELECT Full_Name(firstname,lastname) FROM SportsData").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

```
val sportsData = sc.textFile("/user/acadgild/Sports_data.txt")

val schemaString =
"firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,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),r(5),r(6)))

val sportsDataDF = spark.createDataFrame(rowRDD,schema)

sportsDataDF.createOrReplaceTempView("sportsData")

val Ranking = udf((medal: String, age: Int) => (medal,age) match

{

case (medal,age) if medal == "gold" && age >= 32 => "Pro"

case (medal,age) if medal == "gold" && age <= 32 => "amateur"

case (medal,age) if medal == "silver" && age >= 32 => "expert"

case (medal,age) if medal == "silver" && age <= 32 => "rookie"
})

spark.udf.register("Ranks", Ranking)

val RankingRDD =
sportsDataDF.withColumn("Ranks",Ranking(sportsDataDF.col("medal_type"),sportsDataDF.col("age")))

RankingRDD.show()
```

acadgild@localhost:~

```
scala> val sportsData = sc.textFile("/user/acadgild/Sports_data.txt")
sportsData: org.apache.spark.rdd.RDD[String] = /user/acadgild/Sports_data.txt MapPartitionsRDD[3] at textFile at <console>:27

scala> val schemaString = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string"
schemaString: String = firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,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,StringType,
true), StructField(medal_type,StringType,true), StructField(age,StringType,true), StructField(year,StringType,true), StructField(country,StringType,true))

scala> val rowRDD = sportsData.map(_.split(",")).map(r=>Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[5] at map at <console>:29

scala> val sportsDataDF = spark.createDataFrame(rowRDD,schema)
sportsDataDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more fields]

scala> sportsDataDF.createOrReplaceTempView("sportsData")

scala> val Ranking = udf((medal: String, age: Int) => (medal,age) match
     | {
     | case (medal,age) if medal == "gold" && age >= 32 => "Pro"
     | case (medal,age) if medal == "gold" && age <= 32 => "amateur"
     | case (medal,age) if medal == "silver" && age >= 32 => "expert"
     | case (medal,age) if medal == "silver" && age <= 32 => "rookie"
     |
     | })
Ranking: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, IntegerType)))

scala> spark.udf.register("Ranks", Ranking)
res4: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType, IntegerType)))

scala> val RankingRDD = sportsDataDF.withColumn("Ranks",Ranking(sportsDataDF.col("medal_type"),sportsDataDF.col("age")))
```

acadgild@localhost:~

```
scala> val RankingRDD = sportsDataDF.withColumn("Ranks",Ranking(sportsDataDF.col("medal_type"),sportsDataDF.col("age")))
RankingRDD: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 6 more fields]

scala> RankingRDD.show()
+---------+--------+--------+----------+---+----+-------+-------+
|firstname|lastname|  sports|medal_type|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|
+---------+--------+--------+----------+---+----+-------+-------+
only showing top 20 rows


scala>
```