

Session 21:

SPARK SQL 2

Assignment 1

Task 1

Using spark-sql, Find:

1. What are the total number of gold medal winners every year
2. How many silver medals have been won by USA in each sport

Input: sports_data.txt

1	firstname,lastname,sports,medal_type,age,year,country
2	lisa,cudrow,javellin,gold,34,2015,USA
3	matthew,louis,javellin,gold,34,2015,RUS
4	michael,phelps,swimming,silver,32,2016,USA
5	usha,pt,running,silver,30,2016,IND
6	serena,williams,running,gold,31,2014,FRA
7	roger,federer,tennis,silver,32,2016,CHN
8	jenifer,cox,swimming,silver,32,2014,IND
9	fernando,johnson,swimming,silver,32,2016,CHN
10	lisa,cudrow,javellin,gold,34,2017,USA
11	matthew,louis,javellin,gold,34,2015,RUS
12	michael,phelps,swimming,silver,32,2017,USA
13	usha,pt,running,silver,30,2014,IND
14	serena,williams,running,gold,31,2016,FRA
15	roger,federer,tennis,silver,32,2017,CHN
16	jenifer,cox,swimming,silver,32,2014,IND
17	fernando,johnson,swimming,silver,32,2017,CHN
18	lisa,cudrow,javellin,gold,34,2014,USA
19	matthew,louis,javellin,gold,34,2014,RUS
20	michael,phelps,swimming,silver,32,2017,USA
21	usha,pt,running,silver,30,2014,IND
22	serena,williams,running,gold,31,2016,FRA
23	roger,federer,tennis,silver,32,2014,CHN
24	jenifer,cox,swimming,silver,32,2017,IND
25	fernando,johnson,swimming,silver,32,2017,CHN
26	

Program:

```
Task1.scala Task2.scala
import org.apache.log4j._
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.types.{StructType, StructField, StringType}
import org.apache.spark.sql.Row

object Task1 {
  def main(args : Array[String]) : Unit = {
    Logger.getLogger("org").setLevel(Level.ERROR)

    val spark = SparkSession.builder.appName("Students").master("local").getOrCreate()
    import spark.implicits._

    val rddSports = spark.sparkContext.textFile("E:/Acadgild/Data/Sports_data.txt")
    val sportsHeader = rddSports.first()
    val filteredRddSports = rddSports.filter(row => row != sportsHeader)

    val schema = StructType(sportsHeader.split(",").map(fieldName => StructField(fieldName,StringType, true)))
    val rowDD = filteredRddSports.map(_._split(",")).map(x => Row(x(0), x(1), x(2), x(3), x(4),x(5),x(6)))
    val sportsDF = spark.createDataFrame(rowDD, schema)
    sportsDF.createOrReplaceTempView("sports_data")

    println("1 --> What are the total number of gold medal winners every year")
    val goldWinners = spark.sql("SELECT year, COUNT(*) AS medal_count FROM sports_data WHERE trim(medal_type)=\"gold\" GROUP BY year")
    goldWinners.show()

    println("2 --> How many silver medals have been won by USA in each sport")
    val medalCount = spark.sql("SELECT sports, COUNT(*) FROM sports_data WHERE country=\"USA\" AND medal_type=\"silver\" GROUP BY sports")
    medalCount.show()

    spark.stop()
  }
}
```

Execution:

```
at com.spark.assignment21.Task1.main(Task1.scala)
1 --> What are the total number of gold medal winners every year
+----+-----+
|year|medal_count|
+----+-----+
|2014|      3|
|2015|      3|
|2016|      2|
|2017|      1|
+----+-----+

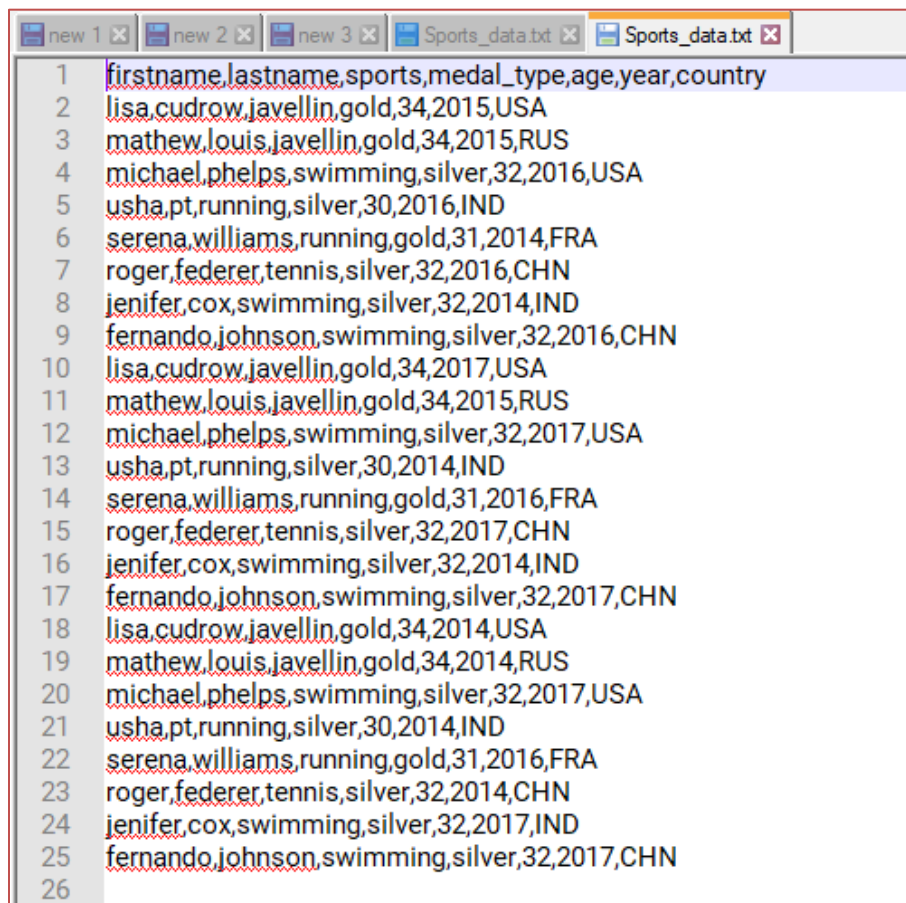
2 --> How many silver medals have been won by USA in each sport
+-----+-----+
| sports|count(1)|
+-----+-----+
|swimming|      3|
+-----+-----+
```

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
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

Input - Sports_data.txt



1	2	3	4	5	6	7	8
1	firstname	lastname	sports	medal_type	age	year	country
2	lisa	cudrow	javellin	gold	34	2015	USA
3	mathew	louis	javellin	gold	34	2015	RUS
4	michael	phelps	swimming	silver	32	2016	USA
5	usha	pt	running	silver	30	2016	IND
6	serena	williams	running	gold	31	2014	FRA
7	roger	federer	tennis	silver	32	2016	CHN
8	jenifer	cox	swimming	silver	32	2014	IND
9	fernando	johnson	swimming	silver	32	2016	CHN
10	lisa	cudrow	javellin	gold	34	2017	USA
11	mathew	louis	javellin	gold	34	2015	RUS
12	michael	phelps	swimming	silver	32	2017	USA
13	usha	pt	running	silver	30	2014	IND
14	serena	williams	running	gold	31	2016	FRA
15	roger	federer	tennis	silver	32	2017	CHN
16	jenifer	cox	swimming	silver	32	2014	IND
17	fernando	johnson	swimming	silver	32	2017	CHN
18	lisa	cudrow	javellin	gold	34	2014	USA
19	mathew	louis	javellin	gold	34	2014	RUS
20	michael	phelps	swimming	silver	32	2017	USA
21	usha	pt	running	silver	30	2014	IND
22	serena	williams	running	gold	31	2016	FRA
23	roger	federer	tennis	silver	32	2014	CHN
24	jenifer	cox	swimming	silver	32	2017	IND
25	fernando	johnson	swimming	silver	32	2017	CHN
26							

Program:

```
Task1.scala Task2.scala
package com.spark.assignment20

import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.log4j._
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.types.{StructType, StructField, StringType}
import org.apache.spark.sql.Row
//import org.apache.spark.sql.functions.udf
import org.apache.spark.sql.functions.{col, udf}

object Task2 {
  def main(args : Array[String]) : Unit = {
    Logger.getLogger("org").setLevel(Level.ERROR)

    val spark = SparkSession.builder.appName("Players").master("local").getOrCreate()
    import spark.implicits._

    val rddSports = spark.sparkContext.textFile("E:/Acadgild/Data/Sports_data.txt")
    val sportsHeader = rddSports.first()
    val filteredRddSports = rddSports.filter(row => row != sportsHeader)

    val schema = StructType(sportsHeader.split(",").map(fieldName => StructField(fieldName,StringType, true)))
    val rowDD = filteredRddSports.map(_._split(",")).map(x => Row(x(0), x(1), x(2), x(3), x(4),x(5),x(6)))
    val sportsDF = spark.createDataFrame(rowDD, schema)
    sportsDF.createOrReplaceTempView("sports_data")

    // Define UDF to form full name. Function takes input parameters as firstname and lastname
    val getFullName = (fname : String, lname : String) => {
      "Mr." + fname.take(2) + " " + lname
    }

    // Define UDF to get player proficiency. Input parameters as medal type and age
    val getPlayerProficiency = (medal_type : String, age : String) => {
      if (medal_type == "gold" && age.toInt >= 32)
        "Professional"
      else if (medal_type == "gold" && age.toInt <= 31)
        "Amateur"
      else if (medal_type == "silver" && age.toInt >= 32)
        "Expert"
      else if (medal_type == "silver" && age.toInt <= 31)
        "Rookie"
      else
        "Novice"
    }

    // Register UDF
    spark.udf.register("formFullName", getFullName)
    spark.udf.register("playerProficiency", getPlayerProficiency)

    val playerData = spark.sql("SELECT firstname, lastname, formFullName(firstname,lastname) AS fullname, playerProficiency(playerProficiency,age) AS playerProficiency FROM sports_data")
    playerData.show()
  }
}
```

Execution:

```
Console [Task2$ (2) [Scala Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe]
<terminated> Task2$ (2) [Scala Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe
  at org.apache.spark.sql.Session$Builder$.anonfun$7$.apply(Session.scala:121)
  at scala.Option.getOrElse(Option.scala:121)
  at org.apache.spark.sql.Session$Builder$.getOrElse(Session.scala:121)
  at com.spark.assignment20.Task2$.main(Task2.scala:17)
  at com.spark.assignment20.Task2.main(Task2.scala)

+-----+-----+-----+
|firstname|lastname|  fullname| proficiency|
+-----+-----+-----+
| lisa| cudrow| Mr.li cudrow|Professional|
| mathew| louis| Mr.ma louis|Professional|
| michael| phelps| Mr.mi phelps| Expert|
| usha| pt| Mr.us pt| Rookie|
| serena|williams|Mr.se williams| Amateur|
| roger| federer| Mr.ro federer| Expert|
| jenifer| cox| Mr.je cox| Expert|
| fernando| johnson| Mr.fe johnson| Expert|
| lisa| cudrow| Mr.li cudrow|Professional|
| mathew| louis| Mr.ma louis|Professional|
| michael| phelps| Mr.mi phelps| Expert|
| usha| pt| Mr.us pt| Rookie|
| serena|williams|Mr.se williams| Amateur|
| roger| federer| Mr.ro federer| Expert|
| jenifer| cox| Mr.je cox| Expert|
| fernando| johnson| Mr.fe johnson| Expert|
| lisa| cudrow| Mr.li cudrow|Professional|
| mathew| louis| Mr.ma louis|Professional|
| michael| phelps| Mr.mi phelps| Expert|
| usha| pt| Mr.us pt| Rookie|
+-----+-----+-----+
only showing top 20 rows
```