Assignment - 21.1

To complete this assignment the hands-on is performed in Idea IntelliJ, so inorder to proceed we have created Spark session object.

Once after that sports dataset is loaded and registered as a Temporary table. Please find the steps for the initial set up below.

SportsUDFs.scala

```scala
import org.apache.spark.sql.SparkSession
object SportsUDFs {

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

  def main(args: Array[String]): Unit = {
    println("Hello Sports Use Case!")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Spark SQL Use Case 1")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")

    //Set the log level as warning
    spark.sparkContext.setLogLevel("WARN")

    val data = spark.sparkContext
      .textFile( path = "C:\\Users\\Ankith M\\Desktop\\Hadoop\\Spark\\SparkSQL Assignment 21.1\\Sports_data.txt")

    val header = data.first()
    val sports_data = data.filter(x => x != header)


    sports_data.foreach(println(_))
```

```scala
    //For implicit conversions like converting RDDs and sequences  to DataFrames
    import spark.implicits._

    val sportsDF = sports_data.map(x=>x.split( regex = ",")).map(x => SportsData(x(0).trim,
      x(1).trim,x(2).trim,x(3).trim,x(4).trim.toInt,x(5).trim,x(6).trim)).toDF()

    sportsDF.printSchema()

    sportsDF.show()

    sportsDF.registerTempTable( tableName = "SPORTS_TAB")
    println("Sports table is registered!!")

    val sports_tabDF = spark.sql( sqlText = "select * from SPORTS_TAB").show()
```

Below is the dataset which will be used in the use case.

```
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
```

Dataset as DataFrame.

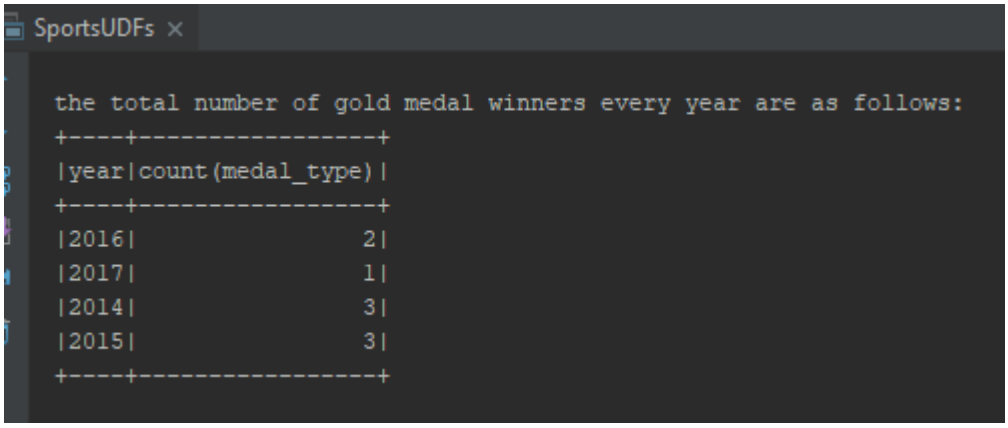
```
+---------+--------+--------+----------+---+----+-------+
|firstname|lastname|  sports|medal_type|age|year|country|
+---------+--------+--------+----------+---+----+-------+
|     lisa|  cudrow|javellin|      gold| 34|2015|    USA|
|   mathew|   louis|javellin|      gold| 34|2015|    RUS|
|  michael|  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|
+---------+--------+--------+----------+---+----+-------+
only showing top 20 rows
```

Task 1.1: What are the total number of gold medal winners every year?

Answer:

```
// Task 1.1 - What are the total number of gold medal winners every year?
println("the total number of gold medal winners every year are as follows: ")
val goldDF = spark.sql(
   sqlText = """SELECT year, count(medal_type)
    FROM SPORTS_TAB WHERE medal_type = "gold" group by year""").show()
```

Output:

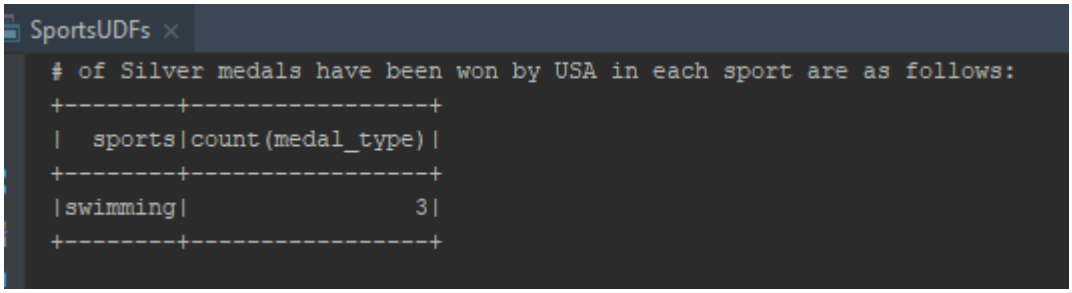


Task 1.2: How many silver medals have been won by USA in each sport?

Answer:

```
// Task 1.2 - How many silver medals have been won by USA in each sport?
println("# of Silver medals have been won by USA in each sport are as follows: ")
val silverDF = spark.sql(
   sqlText = """select sports, count(medal_type) from SPORTS_TAB
    where country = "USA" and medal_type = "silver"
    group by sports""").show()
```

Output:



Task 2.1: Using udfs on dataframe – Change firstname, lastname columns into ➔ Mr.first_two_letters_of_firstname<space>lastname for example - michael, phelps becomes Mr.mi phelps

Answer:

```
// Task 2.1 - 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 Name = (firstname:String, lastname:String)=>"Mr. "
  .concat(firstname.substring(0,2))
  .concat( str = " ")concat(lastname)

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

val fullName = spark.sql( sqlText = """select Full_Name(firstname, lastname)
  as Full_Name from SPORTS_TAB""").show()
```

Output:

```
+---------------+
|      Full_Name|
+---------------+
|  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|
+---------------+
only showing top 20 rows
```

Task 2.2: 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

Answer:

SportsUDFs.scala

```
// Task 2.2 - 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 Ranking = (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( name = "Ranks", Ranking)

val RankStatus = spark.sql( sqlText = """select *, Ranks(medal_type, age)
  as Rank from SPORTS_TAB""").show()

}
}
```

Output:

```
+---------+--------+--------+----------+---+----+-------+-------+
|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
```

The whole code base for this use case is as follows:

```
import org.apache.spark.sql.SparkSession

object SportsUDFs {

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

  def main(args: Array[String]): Unit = {
    println("Hello Sports Use Case!")

    val spark = SparkSession
      .builder()
      .master("local")
      .appName("Spark SQL Use Case 1")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")

    //Set the log level as warning
    spark.sparkContext.setLogLevel("WARN")

    val data = spark.sparkContext
      .textFile("C:\\Users\\Ankith M\\Desktop\\Hadoop\\Spark\\SparkSQL Assignment
21.1\\Sports_data.txt")

    val header = data.first()
    val sports_data = data.filter(x => x != header)


    sports_data.foreach(println(_))

    //For implicit conversions like converting RDDs and sequences  to DataFrames
    import spark.implicits._

    val sportsDF = sports_data.map(x=>x.split(",")).map(x => SportsData(x(0).trim,
      x(1).trim,x(2).trim,x(3).trim,x(4).trim.toInt,x(5).trim,x(6).trim)).toDF()

    sportsDF.printSchema()

    sportsDF.show()

    sportsDF.registerTempTable("SPORTS_TAB")
    println("Sports table is registered!!")

    val sports_tabDF = spark.sql("select * from SPORTS_TAB").show()

    // Task 1.1 - What are the total number of gold medal winners every year?
    println("the total number of gold medal winners every year are as follows: ")
    val goldDF = spark.sql(
      """SELECT year, count(medal_type)
        FROM SPORTS_TAB WHERE medal_type = "gold" group by year""").show()

    // Task 1.2 - How many silver medals have been won by USA in each sport?
    println("# of Silver medals have been won by USA in each sport are as follows:
")
    val silverDF = spark.sql(
      """select sports, count(medal_type) from SPORTS_TAB
        where country = "USA" and medal_type = "silver"
        group by sports""").show()

    // Task 2.1 - 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 Name = (firstname:String, lastname:String)=>"Mr. "
      .concat(firstname.substring(0,2))
      .concat(" ")concat(lastname)

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

    val fullName = spark.sql("""select Full_Name(firstname, lastname)
      as Full_Name from SPORTS_TAB""").show()


    // Task 2.2 - 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 Ranking = (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 RankStatus = spark.sql("""select *, Ranks(medal_type, age)
      as Rank from SPORTS_TAB""").show()

  }
}
```