**Task 1**

**Using spark-sql, Find:**

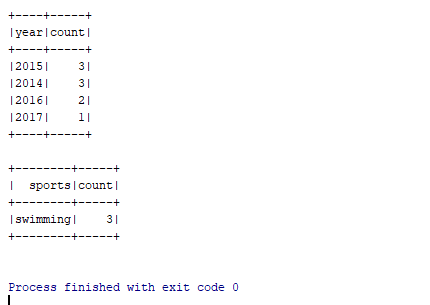
**1. What are the total number of gold medal winners every year**

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

**Code logic:**

**package** SQL  
  
**import** SQL.SparkSQLUseCase1.building  
**import** org.apache.spark.sql.SparkSession  
  
**object** SparkSQLAssignment {  
  
 **case class** sportsdata(fname: String, lname: String, sports: String, medalType: String, age: Int, year: Int, country: String)  
  
 **def** main(args: Array[String]): Unit = {  
  
 *println*(**"hey scala"**)  
  
 **val** spark = SparkSession  
 .*builder*()  
 .master(**"local"**)  
 .appName(**"Spark SQL 2 Assignment"**)  
 .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(**"G:\\Bigdata\\Spark\\sports\_data1.txt"**);  
  
 *println*(**"Sports Data->>"** + data.count())  
  
 **val** header = data.first()  
  
  
 **val** data1 = data.filter(row => row != header)  
  
 *println*(**"Header removed from the data !"**)  
  
 *//For implicit conversions like converting RDDs and sequences to DataFramee* **import** spark.implicits.\_  
  
 **val** build = data1.map(x=> x.split(**","**)).map(x => *sportsdata*(x(0),x(1),x(2),x(3),x(4).toInt,x(5).toInt,x(6))).toDF  
  
 build.show()  
  
 build.createOrReplaceTempView(**"sportsTable"**)  
 **val** build1 = spark.sql(**"select year,count(\*) as count from sportsTable where medalType = 'gold' GROUP BY year"**)  
  
 build1.show()  
  
 **val** build2 = spark.sql(**"select sports,count(\*) as count from sportsTable where medalType = 'silver' and country = 'USA' GROUP BY sports"**)  
  
 build2.show()  
 }  
  
}

**Output**



**Task 2**

**Using udfs on dataframe**

**1. Change firstname, lastname columns into**

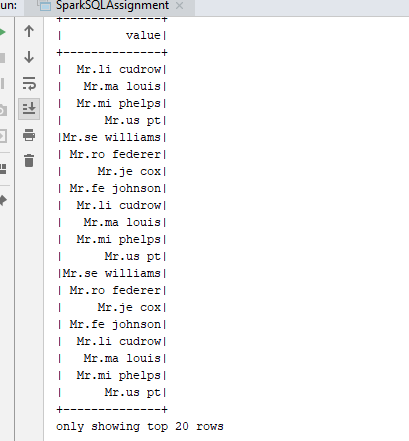
**Mr.first\_two\_letters\_of\_firstname<space>lastname**

**for example - michael, phelps becomes Mr.mi phelps**

**Code logic**

**package** SQL  
  
**import** SQL.SparkSQLUseCase1.building  
**import** org.apache.spark.sql.SparkSession  
  
**object** SparkSQLAssignment {  
  
 **case class** sportsdata(fname: String, lname: String, sports: String, medalType: String, age: Int, year: Int, country: String)  
  
 **def** main(args: Array[String]): Unit = {  
  
 *println*(**"hey scala"**)  
  
 **val** spark = SparkSession  
 .*builder*()  
 .master(**"local"**)  
 .appName(**"Spark SQL 2 Assignment"**)  
 .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(**"G:\\Bigdata\\Spark\\sports\_data1.txt"**);  
  
 *println*(**"Sports Data->>"** + data.count())  
  
 **val** header = data.first()  
  
  
 **val** data1 = data.filter(row => row != header)  
  
 *println*(**"Header removed from the data !"**)  
  
 *//For implicit conversions like converting RDDs and sequences to DataFramee* **import** spark.implicits.\_  
  
 **val** build = data1.map(x=> x.split(**","**)).map(x => *sportsdata*(x(0),x(1),x(2),x(3),x(4).toInt,x(5).toInt,x(6))).toDF  
  
 build.show()  
  
 build.createOrReplaceTempView(**"sportsTable"**)  
**val** build3 = spark.sql(**"select fname,lname from sportsTable"**)  
  
 **val** udf = build3.map(x => (**"Mr."**.concat(x(0).toString.substring(0,2)).concat(**" "**).concat(x(1).toString)))  
  
 udf.show()  
  
  
 }  
  
}

**Output**



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

**Code logic**

**package** SQL  
  
**import** SQL.SparkSQLUseCase1.building  
**import** org.apache.spark.sql.SparkSession  
**import** org.apache.spark.sql.functions.udf  
**object** SparkSQLAssignment {  
  
 **case class** sportsdata(fname: String, lname: String, sports: String, medalType: String, age: Int, year: Int, country: String)  
  
 **def** main(args: Array[String]): Unit = {  
  
 *println*(**"hey scala"**)  
  
 **val** spark = SparkSession  
 .*builder*()  
 .master(**"local"**)  
 .appName(**"Spark SQL 2 Assignment"**)  
 .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(**"G:\\Bigdata\\Spark\\sports\_data1.txt"**);  
  
 *println*(**"Sports Data->>"** + data.count())  
  
 **val** header = data.first()  
  
  
 **val** data1 = data.filter(row => row != header)  
  
 *println*(**"Header removed from the data !"**)  
  
 *//For implicit conversions like converting RDDs and sequences to DataFramee* **import** spark.implicits.\_  
  
 **val** build = data1.map(x=> x.split(**","**)).map(x => *sportsdata*(x(0),x(1),x(2),x(3),x(4).toInt,x(5).toInt,x(6))).toDF  
  
 build.show()  
  
 **def** ranking =(age:Int,medalType:String) => (age,medalType) **match** {  
  
 **case**(age,medalType) **if** medalType == **"gold"** && age >=32 => **"pro"  
  
 case**(age,medalType) **if** medalType == **"gold"** && age <=32 => **"amateur"  
  
 case**(age,medalType) **if** medalType == **"silver"** && age >=32 => **"expert"  
  
 case**(age,medalType) **if** medalType == **"silver"** && age <=32 => **"rookie"** }  
   
 **val** rank = *udf*(ranking)  
  
 **val** output = build.withColumn(**"ranking"**,rank(build.col(**"age"**),build.col(**"medalType"**)))  
  
 output.show()  
  
  
 }  
  
}

**Output**

