

Assignment 21.1 Spark SQL 2

Task 1

Using spark-sql, Find:

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

Solution:

```
package SQL
import org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}
import org.apache.spark.sql.{Row, SparkSession}

object SparkSQLAssignment {

  def main(args: Array[String]): Unit = {

    println("hey scala")

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

    println("Spark Session Object Created")

    spark.sparkContext.setLogLevel("WARN")

    val SportsData = spark.sparkContext.textFile("C:/Users/Gaurav/Desktop/Sports_data.txt")
    val schemaString =
      "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,coun
try: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 resultDF = spark.sql("SELECT year,COUNT (*) FROM SportsData WHERE medal_type =
'gold' GROUP BY year")
    resultDF.show()
```

Output:

```

val schemaString =
  "first_name:string,last_name:string,sports:string,medal_type:string,age:string,year:string,country:string"
val schema = StructType(schemaString.split(" ").map(x =>
  StructField(x.split(" ").split(":")(0),if(x.split(" ").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 resultDF = spark.sql("SELECT year,COUNT (*) FROM SportsData WHERE medal_type = 'gold' GROUP BY year")
resultDF.show()

```

```

19/01/08 19:55:16 INFO SparkContext: Running Spark version 2.1.0
19/01/08 19:55:18 INFO SecurityManager: Changing view acls to: Gaurav
19/01/08 19:55:18 INFO SecurityManager: Changing modify acls to: Gaurav
19/01/08 19:55:18 INFO SecurityManager: Changing view acls groups to:
19/01/08 19:55:18 INFO SecurityManager: Changing modify acls groups to:
19/01/08 19:55:18 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(Gaurav); groups with view permissions: Set();
19/01/08 19:55:19 INFO SparkEnv: Registering MapOutputTracker
19/01/08 19:55:19 INFO SparkEnv: Registering BlockManagerMaster
19/01/08 19:55:19 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
19/01/08 19:55:19 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
19/01/08 19:55:19 INFO DiskBlockManager: Created local directory at C:\Users\Gaurav\AppData\Local\Temp\blockmgr-90ad7368-8ea0-4ac8-bf4e-da8a72b45230
19/01/08 19:55:19 INFO MemoryStore: MemoryStore started with capacity 1170.6 MB
19/01/08 19:55:19 INFO SparkEnv: Registering OutputCommitCoordinator
19/01/08 19:55:20 INFO Utils: Successfully started service 'SparkUI' on port 4040.
19/01/08 19:55:20 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.213.1:4040
19/01/08 19:55:20 INFO Executor: Starting executor ID driver on host localhost
19/01/08 19:55:20 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 52242.
19/01/08 19:55:20 INFO NettyBlockTransferService: Server created on 192.168.213.1:52242
19/01/08 19:55:20 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy

```

```

19/01/08 19:55:20 INFO Executor: Starting executor ID driver on host localhost
19/01/08 19:55:20 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 52242.
19/01/08 19:55:20 INFO NettyBlockTransferService: Server created on 192.168.213.1:52242
19/01/08 19:55:20 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
19/01/08 19:55:20 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.213.1, 52242, None)
19/01/08 19:55:20 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.213.1:52242 with 1170.6 MB RAM, BlockManagerId(driver, 192.168.213.1, 52242, None)
19/01/08 19:55:20 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 192.168.213.1, 52242, None)
19/01/08 19:55:20 INFO BlockManager: Initialised BlockManager: BlockManagerId(driver, 192.168.213.1, 52242, None)
19/01/08 19:55:20 INFO SharedState: Warehouse path is 'file:/C:/Users/Gaurav/IdeaProjects/ScalaSparkProject/spark-warehouse/'.
Spark Session Object Created
=====
|year|count(1)|
=====
|2016|        2|
|2017|        1|
|2014|        3|
|2015|        3|
=====
Process finished with exit code 0

```

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

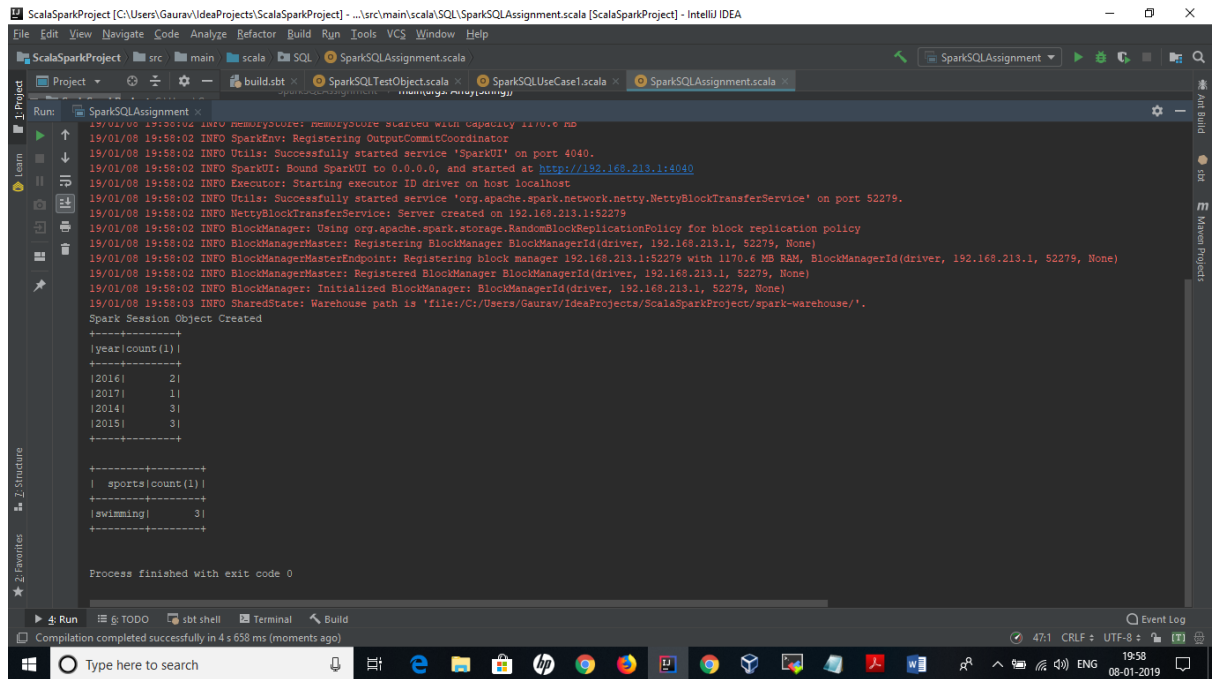
Solution:

```

val result2DF = spark.sql("SELECT sports, COUNT (*) FROM SportsData WHERE medal_type = 'silver' and country = 'USA' GROUP BY sports")
result2DF.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:

```

package SQL

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.udf

object Assignment21_Task2 {

  //declare a case class SPORTS holding the dataset description of the sports_data.
  case class SPORTS(firstname: String, lastname: String, sports: String, medal_type: String,
age: String, year: String, country: String)

  def main(args: Array[String]): Unit = {
    println("Assignment Number 21 Task 2 !!!")

    // Use new SparkSession interface in Spark
    val spark = SparkSession
      .builder()
      .master("local[*]")
      .appName("Assignment21")
      .config("spark.some.config.option", "some-value")

```

```

    .getOrCreate()

    // load the dataset using the textFile method.
    val sports_data_with_header =
spark.sparkContext.textFile("C:/Users/Gaurav/Desktop/Sports_data.txt")
    //creating a variable header, which holds the first line of the dataset, in our data set
Sports_data.txt the first line is a header line.
    val header = sports_data_with_header.first()

    //filter the header line from the dataset using the filter RDD
    val sports_data = sports_data_with_header.filter(row => row != header)

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

    //preparing a structure for the data, mapping it to the case class structure, and finally
converting it to a data frame.
    val sports_data_df = sports_data.map(_._split(",")).map(x => SPORTS(x(0), x(1), x(2), x(3),
x(4), x(5), x(6))).toDF

    sports_data_df.show()

    //Use udf method to define first_and_last_name_concat udf to concat Mr.first 2 letter of
firstname and lastname
    val first_and_last_name_concat = udf((first_name: String, last_name: String) =>
"Mr.".concat(first_name.substring(0, 2)).concat(" ").concat(last_name))

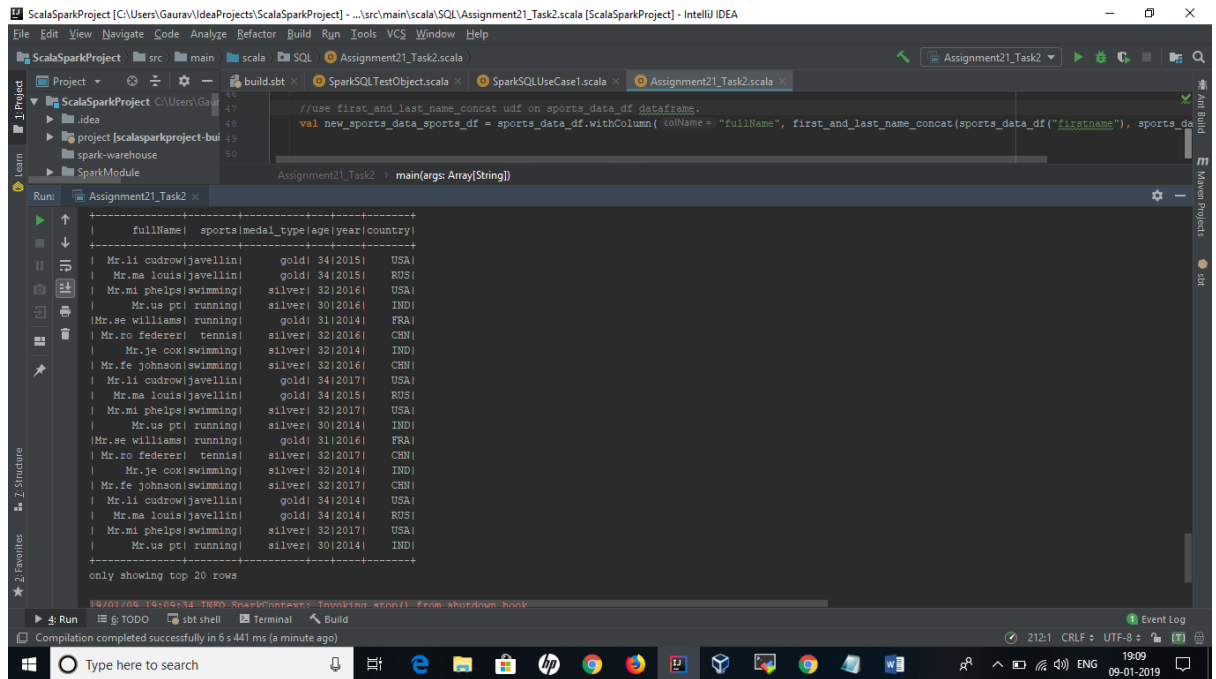
    //use first_and_last_name_concat udf on sports_data_df dataframe.
    val new_sports_data_sports_df = sports_data_df.withColumn("fullName",
first_and_last_name_concat(sports_data_df("firstname"), sports_data_df("lastname")))

new_sports_data_sports_df.select("fullName","sports","medal_type","age","year","country
").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:

```
val Rank = udf((medal_type: String, age: Int) => {
  if (medal_type == "gold" && age >= 32) "pro"
  else if (medal_type == "gold" && age <= 31) "amateur"
  else if (medal_type == "silver" && age >= 32) "expert"
  else if (medal_type == "silver" && age <= 31) "rookie"
  else "no-level"
})

//apply RANK udf on sports_data_df dataframe.
sports_data_df.withColumn("ranking", Rank(sports_data_df("medal_type"),
sports_data_df("age"))).show()
```

Output:

ScalaSparkProject [C:\Users\Gaurav\IdeaProjects\ScalaSparkProject] - ...src\main\scala\SQL\Assignment21_Task2.scala [ScalaSparkProject] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

ScalaSparkProject src main scala SQL Assignment21_Task2.scala

Project build.sbt SparkSQLTestObject.scala SparkSQLUseCase1.scala Assignment21_Task2.scala

ScalaSparkProject C:\Users\Gaurav\IdeaProjects\ScalaSparkProject

Run Assignment21_Task2

```
19/01/09 19:12:30 INFO CodeGenerator: Code generated in 27.80073 ms
+-----+
|firstname|lastname| sports|medal_type|age|year|country|ranking|
+-----+
| lisa| cudrow|javelin| gold| 34|2015| USA| pro|
| mathew| louis|javelin| 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|2016| 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|javelin| gold| 34|2017| USA| pro|
| mathew| louis|javelin| 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|javelin| gold| 34|2014| USA| pro|
| mathew| louis|javelin| 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

19/01/09 19:12:30 INFO SparkContext: Invoking stop() from shutdown hook
19/01/09 19:12:30 INFO SparkUI: Stopped Spark web UI at http://192.168.81.3:18040
```

Run TODO sbt shell Terminal Build

Compilation completed successfully in 5 s 529 ms (a minute ago)

280.1 CRLF UTF-8 19:12 09-01-2019

Type here to search