

### Lab Assignment\_3

**NAME: HARISH CHANDRA JYOSHI**

**CLASS ID: 06**

**TEAM ID: 04**

**UMKC EMAIL ID: [hjddh@mail.umkc.edu](mailto:hjddh@mail.umkc.edu)**

**NAME: ATLURI VENKATA AKHILA KRISHNA**

**CLASS ID: 01**

**TEAM ID: 04**

**UMKC EMAIL ID: [vagq2@mail.umkc.edu](mailto:vagq2@mail.umkc.edu)**

**Video link: [https://youtu.be/J8h5q\\_x6trc](https://youtu.be/J8h5q_x6trc)**

### **TASK 1:Hadoop MapReduce Algorithm**

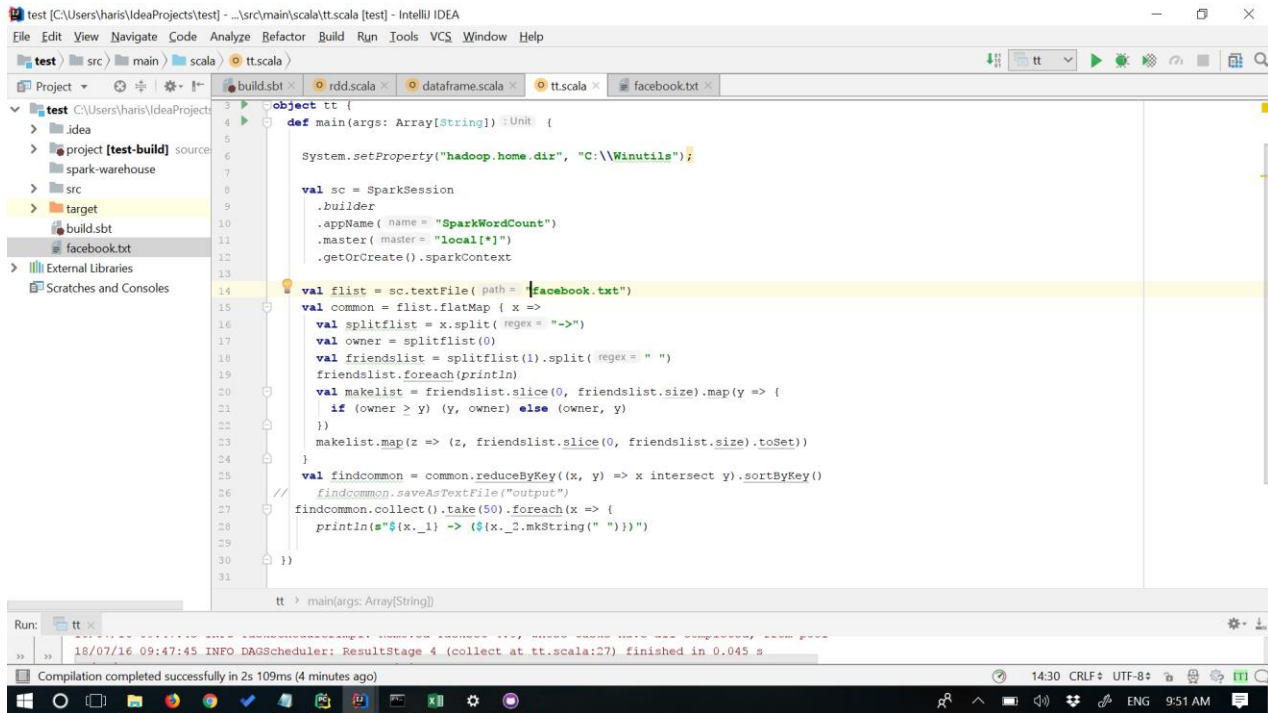
---

**Objective:**

Implementing MapReduce algorithm for finding Facebook common friends and running the MapReduce job on Apache Spark

## Lab Assignment\_3

### Execution of Mapreduce:



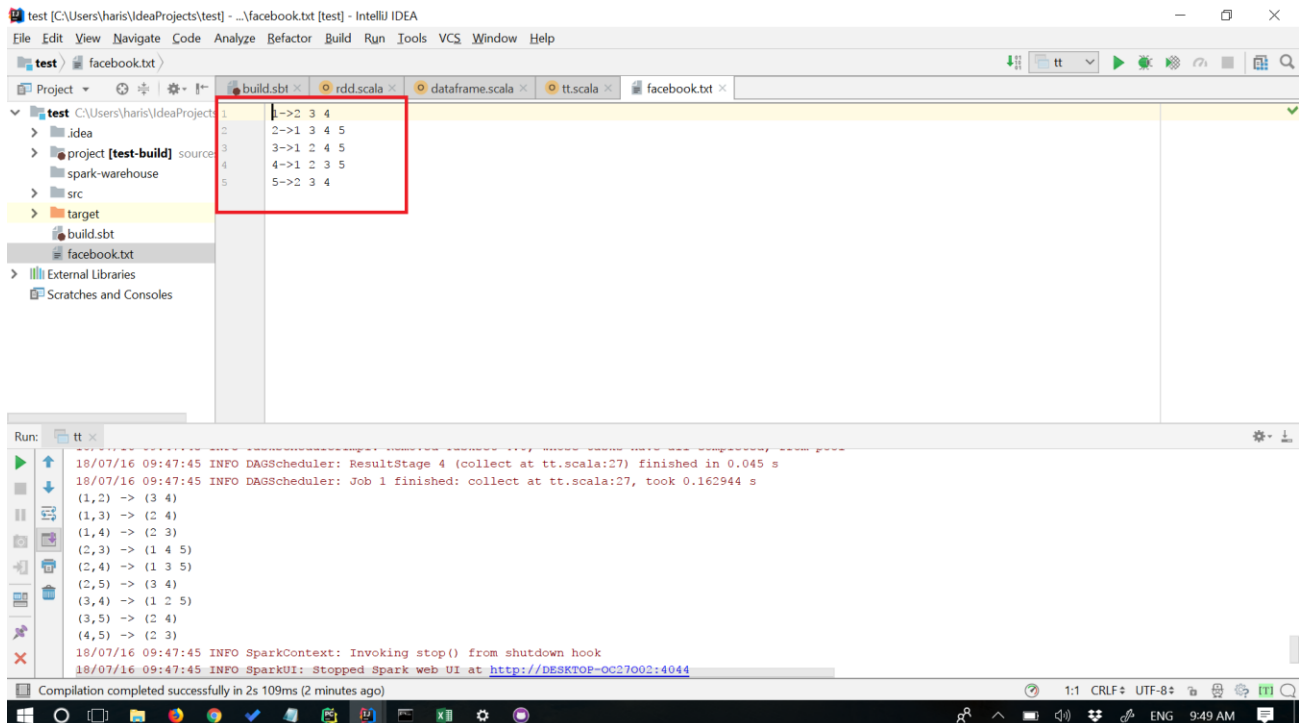
```
object tt {  
  def main(args: Array[String]): Unit {  
    System.setProperty("hadoop.home.dir", "C:\\\\Winutils");  
  
    val sc = SparkSession  
      .builder  
      .appName("SparkWordCount")  
      .master("local[*]")  
      .getOrCreate().sparkContext  
  
    val flist = sc.textFile(path = "facebook.txt")  
    val common = flist.flatMap { x =>  
      val splitlist = x.split(regex = ">")  
      val owner = splitlist(0)  
      val friendslist = splitlist(1).split(regex = " ")  
      friendslist.foreach(println)  
      val makelist = friendslist.slice(0, friendslist.size).map(y => {  
        if (owner > y) (y, owner) else (owner, y)  
      })  
      makelist.map(z => (z, friendslist.slice(0, friendslist.size).toSet))  
    }  
    val findcommon = common.reduceByKey((x, y) => x intersect y).sortByKey()  
    findcommon.saveAsTextFile("output")  
    findcommon.collect().take(50).foreach(x => {  
      println(s"${x._1} -> ${x._2.mkString(" ")})  
    })  
  }  
}
```

Run: tt

18/07/16 09:47:45 INFO DAGScheduler: ResultStage 4 (collect at tt.scala:27) finished in 0.045 s

Compilation completed successfully in 2s 109ms (4 minutes ago)

### Input:



```
1 1->2 3 4  
2 2->1 3 4 5  
3 3->1 2 4 5  
4 4->1 2 3 5  
5 5->2 3 4
```

Run: tt

18/07/16 09:47:45 INFO DAGScheduler: ResultStage 4 (collect at tt.scala:27) finished in 0.045 s

18/07/16 09:47:45 INFO DAGScheduler: Job 1 finished: collect at tt.scala:27, took 0.162944 s

(1,2) -> (3 4)  
(1,3) -> (2 4)  
(1,4) -> (2 3)  
(2,3) -> (1 4 5)  
(2,4) -> (1 3 5)  
(2,5) -> (3 4)  
(3,4) -> (1 2 5)  
(3,5) -> (2 4)  
(4,5) -> (2 3)

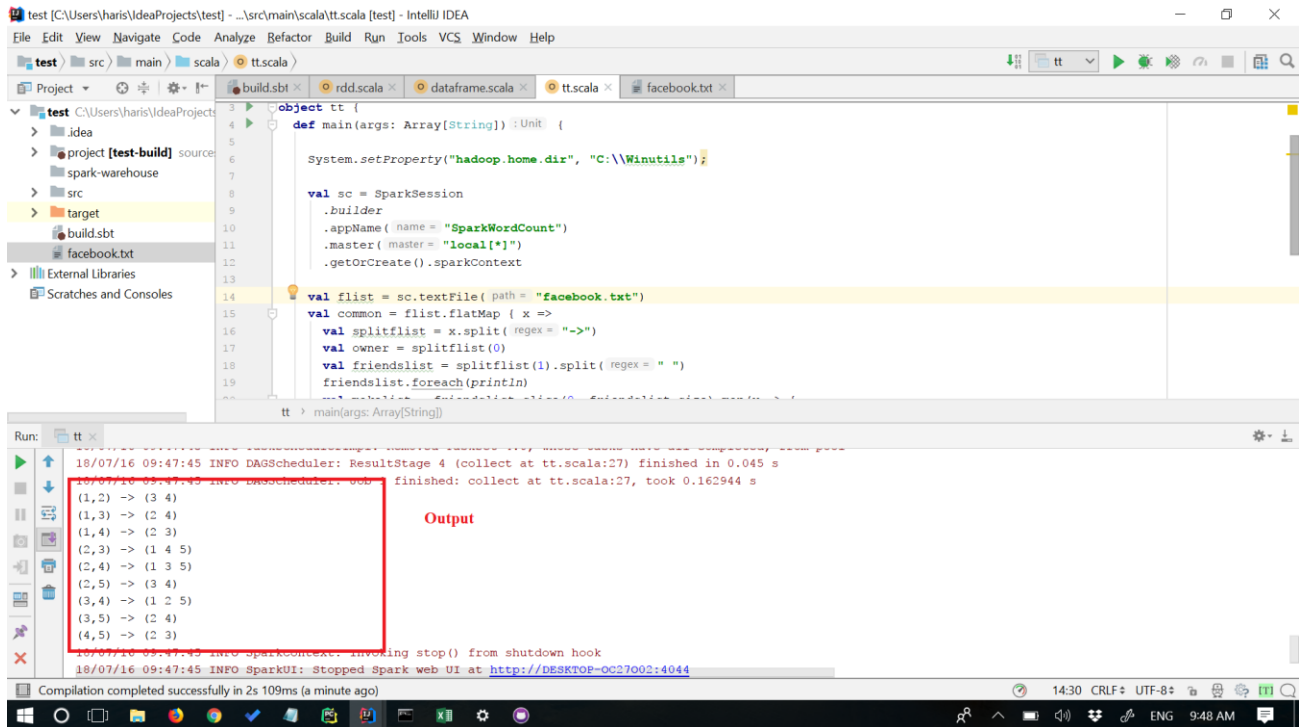
18/07/16 09:47:45 INFO SparkContext: Invoking stop() from shutdown hook

18/07/16 09:47:45 INFO SparkUI: Stopped Spark web UI at <http://DESKTOP-0027002:4044>

Compilation completed successfully in 2s 109ms (2 minutes ago)

## Lab Assignment\_3

### Output:



The screenshot shows the IntelliJ IDEA interface. The main editor displays a Scala file named `tt.scala` with the following code:

```
object tt {  
  def main(args: Array[String]): Unit {  
    System.setProperty("hadoop.home.dir", "C:\\Winutils");  
  
    val sc = SparkSession  
      .builder  
      .appName("SparkWordCount")  
      .master("local[*]")  
      .getOrCreate().sparkContext  
  
    val flist = sc.textFile(path = "facebook.txt")  
    val common = flist.flatMap { x =>  
      val splitlist = x.split(regex = "->")  
      val owner = splitlist(0)  
      val friendslist = splitlist(1).split(regex = " ")  
      friendslist.foreach(println)  
    }  
  }  
}
```

The Run console at the bottom shows the output of the program:

```
18/07/16 09:47:45 INFO DAGScheduler: ResultStage 4 (collect at tt.scala:27) finished in 0.045 s  
18/07/16 09:47:45 INFO DAGScheduler: Job 0 finished: collect at tt.scala:27, took 0.162944 s  
(1,2) -> (3 4)  
(1,3) -> (2 4)  
(1,4) -> (2 3)  
(2,3) -> (1 4 5)  
(2,4) -> (1 3 5)  
(2,5) -> (3 4)  
(3,4) -> (1 2 5)  
(3,5) -> (2 4)  
(4,5) -> (2 3)  
18/07/16 09:47:45 INFO SparkContext: Invoking stop() from shutdown hook  
18/07/16 09:47:45 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-0G27002:4044  
Compilation completed successfully in 2s 109ms (a minute ago)
```

The output is highlighted with a red box, and the word "Output" is written in red text to the right of the box.

### Algorithm: The map function

map(owner,friends list): emit(owner,friend)-> friends list return set of (key,value) pairs that each key,(owner,friend),has values(friends list)

### Algorithm: The Reduce function

reduce(owner, friend)->friends list emit (owner, friend)->friends list

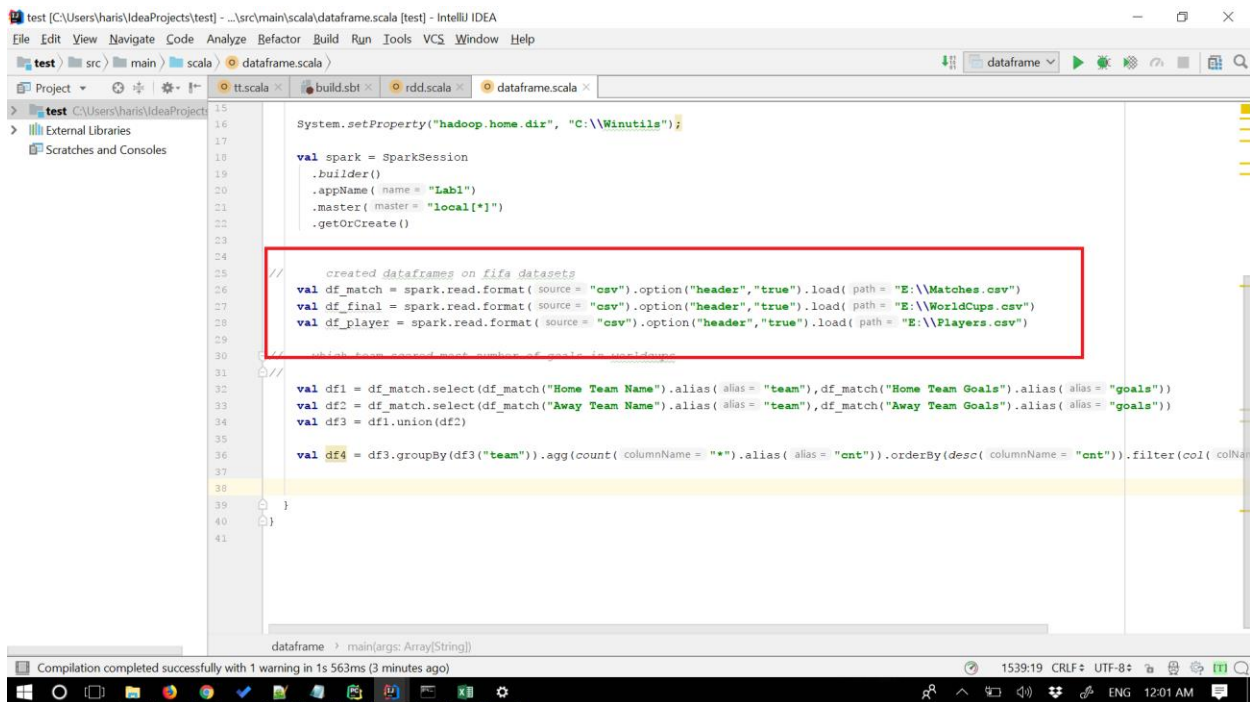
# TASK 2: Spark Data Frames

### Objective:

- a) Creating a DataFrame from the datasets
- b) Performing 10 Intuitive questions
- c) Queries in spark RDD's

Datasets Used:

## a) Creating a DataFrame from the datasets



```
test [C:\Users\haris\IdeaProjects\test] - ...src\main\scala\dataframe.scala [test] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
test src main scala dataframe.scala
Project test
External Libraries
Scratches and Consoles

15 System.setProperty("hadoop.home.dir", "C:\\Winutils");
16
17
18
19 val spark = SparkSession
20   .builder()
21   .appName( name = "Lab1")
22   .master( master = "local[*]")
23   .getOrCreate()
24
25 // created dataframes on fifa datasets
26 val df_match = spark.read.format( source = "csv").option("header","true").load( path = "E:\\Matches.csv")
27 val df_final = spark.read.format( source = "csv").option("header","true").load( path = "E:\\WorldCups.csv")
28 val df_player = spark.read.format( source = "csv").option("header","true").load( path = "E:\\Players.csv")
29
30 // which team scored most number of goals in world cups
31
32 val df1 = df_match.select(df_match("Home Team Name").alias( alias = "team"),df_match("Home Team Goals").alias( alias = "goals"))
33 val df2 = df_match.select(df_match("Away Team Name").alias( alias = "team"),df_match("Away Team Goals").alias( alias = "goals"))
34 val df3 = df1.union(df2)
35
36 val df4 = df3.groupBy(df3("team")).agg(count( columnName = "*" ).alias( alias = "cnt")).orderBy(desc( columnName = "cnt")).filter(col( colName
37
38
39 }
40
41

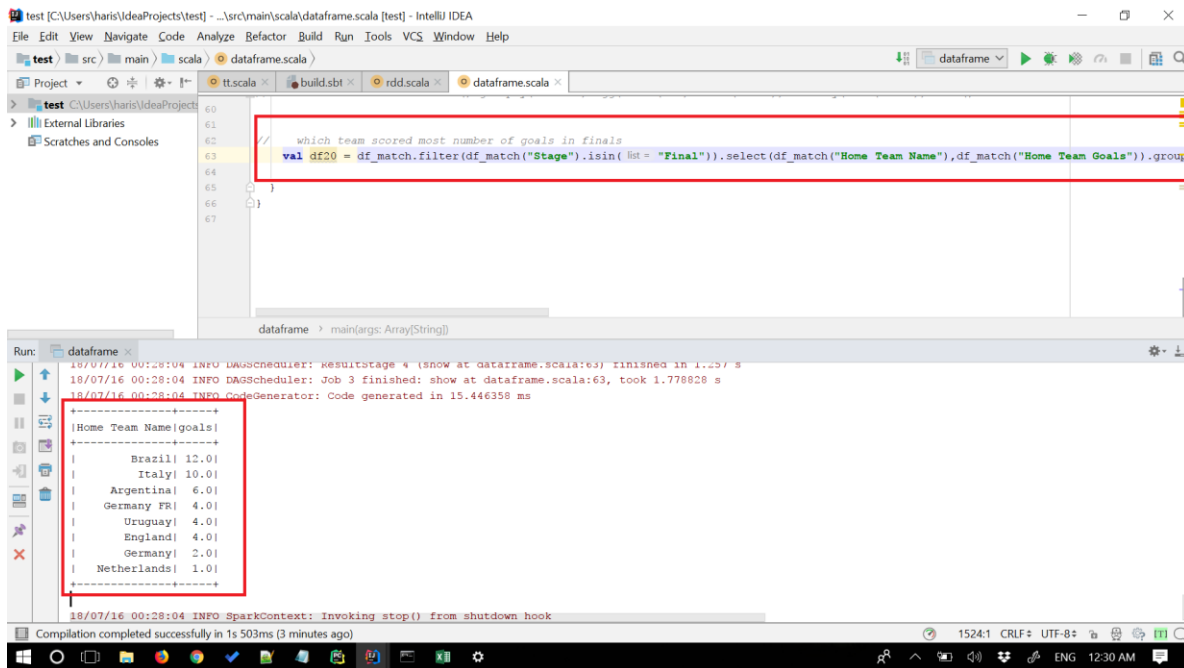
dataframe main(args: Array[String])
Compilation completed successfully with 1 warning in 1s 563ms (3 minutes ago) 1539:19 CRLF UTF-8 ENG 12:01 AM
```

## Lab Assignment\_3

### b) Performing 10 Intutive questions

#### 1) Teams that scored most number of goals in finals

Query: `val df20 = df_match.filter(df_match("Stage").isin("Final")).select(df_match("Home Team Name"),df_match("Home Team Goals")).groupBy(df_match("Home Team Name")).agg(sum(df_match("Home Team Goals")).alias("goals")).orderBy(desc("goals")).show()`



The screenshot shows the IntelliJ IDEA interface. The top pane displays a Scala file named `dataframe.scala` with the following code:

```
60  
61  
62 // which team scored most number of goals in finals  
63 val df20 = df_match.filter(df_match("Stage").isin(list = "Final")).select(df_match("Home Team Name"),df_match("Home Team Goals")).groupBy(df_match("Home Team Name")).agg(sum(df_match("Home Team Goals")).alias("goals")).orderBy(desc("goals")).show()  
64  
65  
66  
67
```

The bottom pane shows the output of the `show()` method, which is a table with two columns: `Home Team Name` and `goals`. The output is as follows:

Home Team Name	goals
Brazil	12.0
Italy	10.0
Argentina	6.0
Germany FR	4.0
Uruguay	4.0
England	4.0
Germany	2.0
Netherlands	1.0

The output is displayed in the Run console, which also shows logs from the DAGScheduler and CodeGenerator. The bottom status bar indicates that the compilation completed successfully in 1s 503ms (3 minutes ago).

## Lab Assignment\_3

### 2) Average number of goals scored by a team in worldcups

Query: `val Average_goals = spark.sql("SELECT Country AS Teams, ROUND(AVG(GoalsScored),0) AS average_goals FROM table2 GROUP BY Country")`  
`Average_goals.show()`

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `dataframe.scala`. The code defines a Spark SQL query to calculate the average goals scored by teams in World Cups. The query is: `val Average_goals = spark.sql("SELECT Country AS Teams, ROUND(AVG(GoalsScored),0) AS average_goals FROM table2 GROUP BY Country")`. The output of the `show()` method is displayed in the Run console, showing a table with two columns: `Teams` and `average_goals`. The data is as follows:

Teams	average_goals
Sweden	126.0
Germany	122.0
France	128.0
Argentina	102.0
Korea/Japan	161.0
Chile	89.0
Italy	93.0
Spain	146.0
USA	141.0
Uruguay	70.0
Mexico	114.0
Switzerland	140.0
Brazil	130.0

## Lab Assignment\_3

### 3) Most number of time a country has hosted Worldcups

Query: `val df21 = df_final.groupBy("Country").count().orderBy(desc("count")).show()`

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `dataframe.scala`. The code defines `df21` as a DataFrame representing the number of times each country has hosted the World Cup, ordered by count in descending order. The output of the `show()` method is displayed in the Run console.

```
59 // val df17 = df16.distinct().groupBy("coach").agg(count("")).alias("cnt").orderBy(desc("cnt")).show()
60
61
62 // which team scored most number of goals in finals
63 val df20 = df_match.filter(df_match("Stage").isin("Final")).select(df_match("Home Team Name"), df_match("Home Team Goals")).groupBy("Home Team Name").count().orderBy(desc("count")).show()
64
65 // which countries has hosted more number of times
66 val df21 = df_final.groupBy(col = "Country").count().orderBy(desc(columnName = "count")).show()
67
68
69
70
```

The Run console output shows the following DataFrame:

Country	count
Germany	2
Italy	2
Mexico	2
France	2
Brazil	2
Korea/Japan	1
Sweden	1
Spain	1
Switzerland	1
Argentina	1
Chile	1
South Africa	1
England	1

Compilation completed successfully in 1s 345ms (a minute ago)

## Lab Assignment\_3

### 4) Player who has won most matches in Worldcup

Query: val df14 =

```
df_match.filter(df_match("Stage").isin("Final")).select(df_match("Year"),when(df_match("Home Team Goals") < df_match("Away Team Goals"),df_match("Away Team Name")).otherwise(df_match("Home Team Name")).alias("team"),when(df_match("Home Team Goals") < df_match("Away Team Goals"),df_match("Away Team Initials")).otherwise(df_match("Home Team Initials")).alias("initials"),df_match("RoundID"),df_match("MatchID")) val df15 = df_player.select(df_player("RoundID"),df_player("MatchID"),df_player("Coach Name"),df_player("Team Initials")) val df16 = df14.join(df15,(df14("RoundID") <=> df15("RoundID") && df14("MatchID") <=> df15("MatchID") && df14("initials") <=> df15("Team Initials"))).select(df15("Coach Name").alias("coach"),df14("Year")) val df17 = df16.distinct().groupBy("coach").agg(count("*").alias("cnt")).orderBy(desc("cnt")).show()
```

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `dataframe.scala`. The code defines several DataFrames and performs a join operation to find the player who has won the most matches in the World Cup. The query is highlighted with a red box. Below the code, the execution results are shown in a console window, also highlighted with a red box. The results are a table with two columns: `coach` and `cnt`.

```
val df14 = df_match.filter(df_match("Stage").isin("Final")).select(df_match("Year"),when(df_match("Home Team Goals") < df_match("Away Team Goals"),df_match("Away Team Name")).otherwise(df_match("Home Team Name")).alias("team"),when(df_match("Home Team Goals") < df_match("Away Team Goals"),df_match("Away Team Initials")).otherwise(df_match("Home Team Initials")).alias("initials"),df_match("RoundID"),df_match("MatchID")) val df15 = df_player.select(df_player("RoundID"),df_player("MatchID"),df_player("Coach Name"),df_player("Team Initials")) val df16 = df14.join(df15,(df14("RoundID") <=> df15("RoundID") && df14("MatchID") <=> df15("MatchID") && df14("initials") <=> df15("Team Initials"))).select(df15("Coach Name").alias("coach"),df14("Year")) val df17 = df16.distinct().groupBy("coach").agg(count("*").alias("cnt")).orderBy(desc("cnt")).show()
```

```
coach|cnt|
-----+---+
|POZZO Vittorio (ITA)| 2|
|BILARDO Carlos (ARG)| 1|
|RAMSEY Alf (ENG)| 1|
|JACQUET Aime (FRA)| 1|
|ZAGALLO Mario (BRA)| 1|
|MENOTTI Cesar Lui...| 1|
|LOEW Joachim (GER)| 1|
|HERBERGER Sepp (FRG)| 1|
|SUPPICI Alberto (...)| 1|
|DEL BOSQUE Vicent...| 1|
|PARREIRA Carlos A...| 1|
|SCHOEN Helmut (FRG)| 1|
```



## Lab Assignment\_3

### 5) Players who won most number of Worldcups

Query: val df10 =

```
df_match.filter(df_match("Stage").isin("Final")).select(df_match("Year"),when(df_match("Home Team Goals") < df_match("Away Team Goals"),df_match("Away Team Name")).otherwise(df_match("Home Team Name")).alias("team"),when(df_match("Home Team Goals") < df_match("Away Team Goals"),df_match("Away Team Initials")).otherwise(df_match("Home Team Initials")).alias("initials"),df_match("RoundID"),df_match("MatchID")) val df11 = df_player.select(df_player("RoundID"),df_player("MatchID"),df_player("Player Name"),df_player("Team Initials")) val df12 = df10.join(df11,(df10("RoundID") <=> df11("RoundID") && df10("MatchID") <=> df11("MatchID") && df10("initials") <=> df11("Team Initials"))).select(df11("Player Name").alias("player"),df10("Year").alias("year")) val df13 = df12.distinct().groupBy("player").agg(count("*").alias("cnt")).orderBy(desc("cnt")).show()
```

The screenshot shows the IntelliJ IDEA IDE with a Scala project. The code in the main editor defines several DataFrames and performs a series of joins and aggregations to find the player who has won the most World Cup matches. The code is as follows:

```
val df1 = df_match.join(df_player, (df10("RoundID") <=> df11("RoundID") && df10("MatchID") <=> df11("MatchID")))
val df10 = df1.filter(df10("Stage").isin("Final")).select(df10("Year"),when(df10("Home Team Goals") < df10("Away Team Goals"),df10("Away Team Name")).otherwise(df10("Home Team Name")).alias("team"),when(df10("Home Team Goals") < df10("Away Team Goals"),df10("Away Team Initials")).otherwise(df10("Home Team Initials")).alias("initials"),df10("RoundID"),df10("MatchID"))
val df11 = df_player.select(df_player("RoundID"),df_player("MatchID"),df_player("Player Name"),df_player("Team Initials"))
val df12 = df10.join(df11,(df10("RoundID") <=> df11("RoundID") && df10("MatchID") <=> df11("MatchID") && df10("initials") <=> df11("Team Initials"))).select(df11("Player Name").alias("player"),df10("Year").alias("year"))
val df13 = df12.distinct().groupBy("player").agg(count("*").alias("cnt")).orderBy(desc("cnt")).show()
```

The Run console at the bottom shows the output of the query, which is a list of players and the number of matches they have won (cnt):

player	cnt
Edson Arant...	3
DIDI	2
MAURO RAMOS	2
Eraldo MONZEGU	2
NILTON SANTOS	2
RONALDO	2
BELLINI	2
Mario ZAGALLO	2
ZITO	2
DIDA	2
CAFU	2
GARRINCHA	2
VAVA	2
Guido MASETTI	2

## Lab Assignment\_3

### 6) Home team who scored goals greater than 3

Query: `val Goals = spark.sql("SELECT Stage,Stadium,City,Home_Team_Name FROM table4 WHERE Home_Team_Goals >= 3 AND Home_Team_Goals <= 10") Goals.show()`

The screenshot shows the IntelliJ IDEA interface. The main editor displays Scala code for a Spark application. A red box highlights the SQL query used to filter home teams based on goals scored. Below the code, the 'Run' console shows the execution output, including a log message and a table of results. A red box highlights the table output.

```
71 //
72 val df22=df_final.select("Winner").distinct().count()
73 println("Number of distinct countries who have won the worldcup:" +df22);
74
75 df_match.createOrReplaceTempView( viewName = "table1")
76 df_final.createOrReplaceTempView( viewName = "table2")
77 df_player.createOrReplaceTempView( viewName = "table3")
78 df_match1.createOrReplaceTempView( viewName = "table4")
79
80
81 // Filtering the Home teams that scored goals >=3 and <=10
82 val Goals = spark.sql( sqlText = "SELECT Stage,Stadium,City,Home_Team_Name FROM table4 WHERE Home_Team_Goals >= 3 AND Home_Team_Goals <= 10")
83 Goals.show()
84
```

Run: dataframe x

```
18/07/16 02:03:53 INFO DAGScheduler: Job 4 finished: show at dataframe.scala:81, took 0.179537 s
```

Stage	Stadium	City	Home_Team_Name
Group 1	Pocitos	Montevideo	France
Group 4	Parque Central	Montevideo	USA
Group 3	Pocitos	Montevideo	Romania
Group 1	Parque Central	Montevideo	Chile
Group 2	Parque Central	Montevideo	Yugoslavia
Group 4	Parque Central	Montevideo	USA
Group 1	Estadio Centenario	Montevideo	Argentina
Group 2	Estadio Centenario	Montevideo	Brazil
Group 3	Estadio Centenario	Montevideo	Uruguay
Group 1	Estadio Centenario	Montevideo	Argentina

Compilation completed successfully in 1s 400ms (4 minutes ago)

## Lab Assignment\_3

### 7) Teams scored most number of goals in Worldcups

Query: val df1 = df\_match.select(df\_match("Home Team Name").alias("team"),df\_match("Home Team Goals").alias("goals")) val df2 = df\_match.select(df\_match("Away Team Name").alias("team"),df\_match("Away Team Goals").alias("goals")) val df3 = df1.union(df2) val df4=df3.groupBy(df3("team")).agg(count("\*").alias("cnt")).orderBy(desc("cnt")).filter(col("team")isNotNull).show()

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `dataframe.scala`. The code defines a Spark DataFrame `df_match` and performs a series of operations to find the team with the most goals. The code is as follows:

```
val df_final = spark.read.format("csv").option("header", "true").load(path = "E:\\WorldCups.csv")
val df_player = spark.read.format("csv").option("header", "true").load(path = "E:\\Players.csv")

// which team scored most number of goals in worldcups
val df1 = df_match.select(df_match("Home Team Name").alias("team"), df_match("Home Team Goals").alias("goals"))
val df2 = df_match.select(df_match("Away Team Name").alias("team"), df_match("Away Team Goals").alias("goals"))
val df3 = df1.union(df2)
val df4 = df3.groupBy(df3("team")).agg(count(columnName = "*").alias("cnt")).orderBy(desc(columnName = "cnt")).filter(col(columnName = "team") isNotNull).show()
```

The output of the `show()` method is displayed in the Run console, showing a table with two columns: `team` and `cnt`. The data is as follows:

team	cnt
Brazil	100
Italy	83
Argentina	81
Germany FR	62
England	62
France	61
Spain	59
Mexico	54
Netherlands	54
Uruguay	52
Germany	48

The bottom status bar indicates that the compilation completed successfully with 1 warning in 1s 563ms (a minute ago).

## Lab Assignment\_3

### 8) Players part of most number of Worldcups

Query: val df5 =

df\_player.select(df\_player("RoundID"),df\_player("MatchID"),df\_player("Player Name")) val

df6 = df\_match.select(df\_match("RoundID"),df\_match("MatchID"),df\_match("Year")) val

df7 = df5.join(df6,(df5("RoundID") <=> df6("RoundID") && df5("MatchID") <=>

df6("MatchID"))).select(df5("Player Name"),df6("Year")) val df8 =

df7.distinct().groupBy(df7("Player

Name")).agg(count("\*").alias("cnt")).orderBy(desc("cnt")).show()

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `dataframe.scala`. The code defines several DataFrames and performs a join, followed by a distinct operation and a group-by aggregation to find the number of World Cup appearances for each player. The output is displayed in the Run console.

```
31 //
32 val df1 = df_match.select(df_match("Home Team Name").alias("team"),df_match("Home Team Goals").alias("goals"))
33 val df2 = df_match.select(df_match("Away Team Name").alias("team"),df_match("Away Team Goals").alias("goals"))
34 val df3 = df1.union(df2)
35 //
36 val df4 = df3.groupBy(df3("team")).agg(count("*").alias("cnt")).orderBy(desc("cnt")).filter(col("team").isNotNull).show()
37 //
38 //
39 val df5 = df_player.select(df_player("RoundID"),df_player("MatchID"),df_player("Player Name"))
40 val df6 = df_match.select(df_match("RoundID"),df_match("MatchID"),df_match("Year"))
41 val df7 = df5.join(df6,(df5("RoundID") <=> df6("RoundID") && df5("MatchID") <=> df6("MatchID"))).select(df5("Player Name"),df6("Year"))
42 val df8 = df7.distinct().groupBy(df7("Player Name")).agg(count(columnName = "*").alias("cnt")).orderBy(desc(columnName = "cnt"))
```

The Run console output shows the following table:

Player Name	cnt
RONALDO	6
Antonio CARRASQAL	5
OSCAR	4
SONG	4
PAREDES	4
JONES	4
Sepp MAIER	4
DIDA	4
NILTON SANTOS	4
XAVI	4
CASTILHO	4
GAMARRA	4
Diego MARADONA	4
Lev YASHIN	4

## Lab Assignment\_3

### 9) pattern Recognition Germany

Query: `val Patternreg = spark.sql("SELECT * from table2 WHERE Third LIKE 'Germany'")`  
`Patternreg.show()`

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `dataframe.scala`. The code defines a Spark SQL query to filter rows where the 'Third' column contains 'Germany'. The output of the `show()` method is displayed in the Run console, showing a table of football match data.

```
83 // Goals.show()
84
85
86 // pattern recognition where third position is germany
87 val Patternreg = spark.sql( sqlText = "SELECT * from table2 WHERE Third LIKE 'Germany'")
88 Patternreg.show()
89
90
91
92
```

Run: dataframe x

```
18/07/16 02:11:40 INFO DAGScheduler: ResultStage 4 (show at dataframe.scala:86) finished in 0.102 s
18/07/16 02:11:40 INFO DAGScheduler: Job 4 finished: show at dataframe.scala:86, took 0.106125 s
```

Year	Country	Winner	Runners-Up	Third	Fourth	GoalsScored	QualifiedTeams	MatchesPlayed	Attendance
1934	Italy	Italy	Czechoslovakia	Germany	Austria	70	16	17	363.000
2006	Germany	Italy	France	Germany	Portugal	147	32	64	3.359.439
2010	South Africa	Spain	Netherlands	Germany	Uruguay	145	32	64	3.178.856

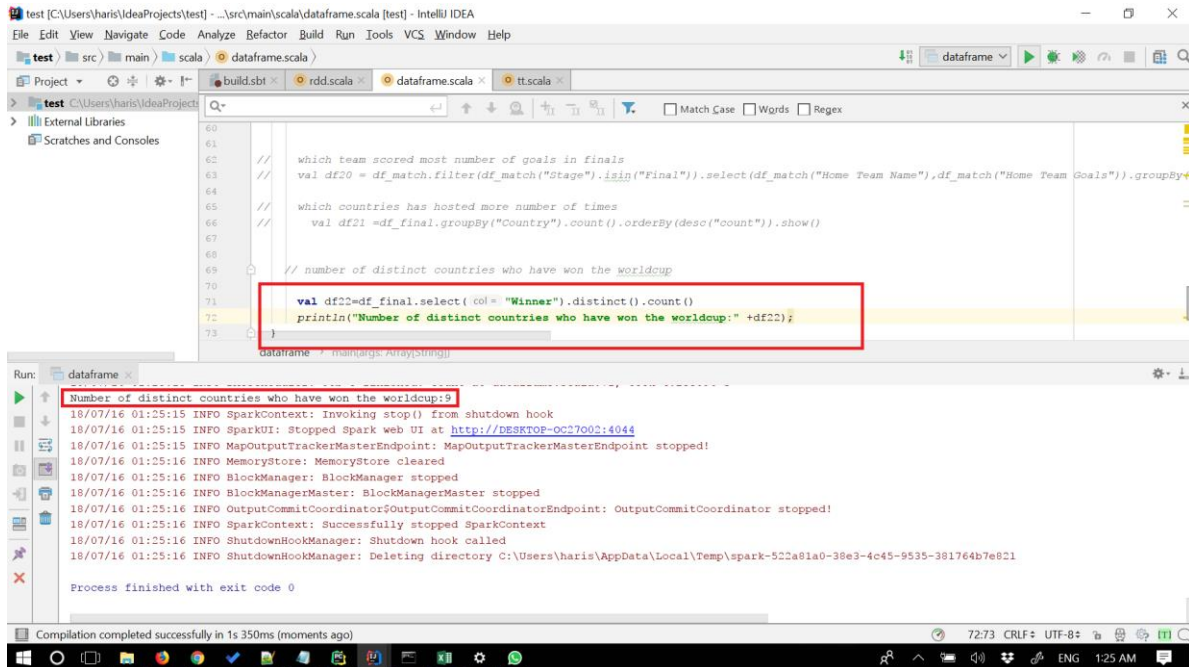
```
18/07/16 02:11:40 INFO SparkContext: Invoking stop() from shutdown hook
18/07/16 02:11:40 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-OC27002:4044
18/07/16 02:11:40 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/07/16 02:11:40 INFO MemoryStore: MemoryStore cleared
18/07/16 02:11:40 INFO BlockManager: BlockManager stopped
```

Compilation completed successfully in 1s 251ms (a minute ago)

## Lab Assignment\_3

### 10) Number of distinct countries Who won the Worldcup

Query: `val df22=df_final.select("Winner").distinct().count() println("Number of distinct countries who have won the worldcup:" +df22);`



The screenshot shows the IntelliJ IDEA interface. The main editor displays a Scala file named `dataframe.scala` with the following code:

```
60
61
62 // which team scored most number of goals in finals
63 // val df20 = df_match.filter(df_match("Stage").isin("Final")).select(df_match("Home Team Name"),df_match("Home Team Goals")).groupBy
64
65 // which countries has hosted more number of times
66 // val df21 =df_final.groupBy("Country").count().orderBy(desc("count")).show()
67
68
69 // number of distinct countries who have won the worldcup
70
71 val df22=df_final.select(col="Winner").distinct().count()
72 println("Number of distinct countries who have won the worldcup:" +df22);
73 }
```

The code is highlighted with a red box. The `Run` tab at the bottom shows the execution output:

```
Number of distinct countries who have won the worldcup:9
18/07/16 01:25:15 INFO SparkContext: Invoking stop() from shutdown hook
18/07/16 01:25:15 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-OC27002:4044
18/07/16 01:25:15 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/07/16 01:25:16 INFO MemoryStore: MemoryStore cleared
18/07/16 01:25:16 INFO BlockManager: BlockManager stopped
18/07/16 01:25:16 INFO BlockManagerMaster: BlockManagerMaster stopped
18/07/16 01:25:16 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/07/16 01:25:16 INFO SparkContext: Successfully stopped SparkContext
18/07/16 01:25:16 INFO ShutdownHookManager: Shutdown hook called
18/07/16 01:25:16 INFO ShutdownHookManager: Deleting directory C:\Users\harris\AppData\Local\Temp\spark-522a81a0-38e3-4c45-9535-381764b7e021

Process finished with exit code 0
```

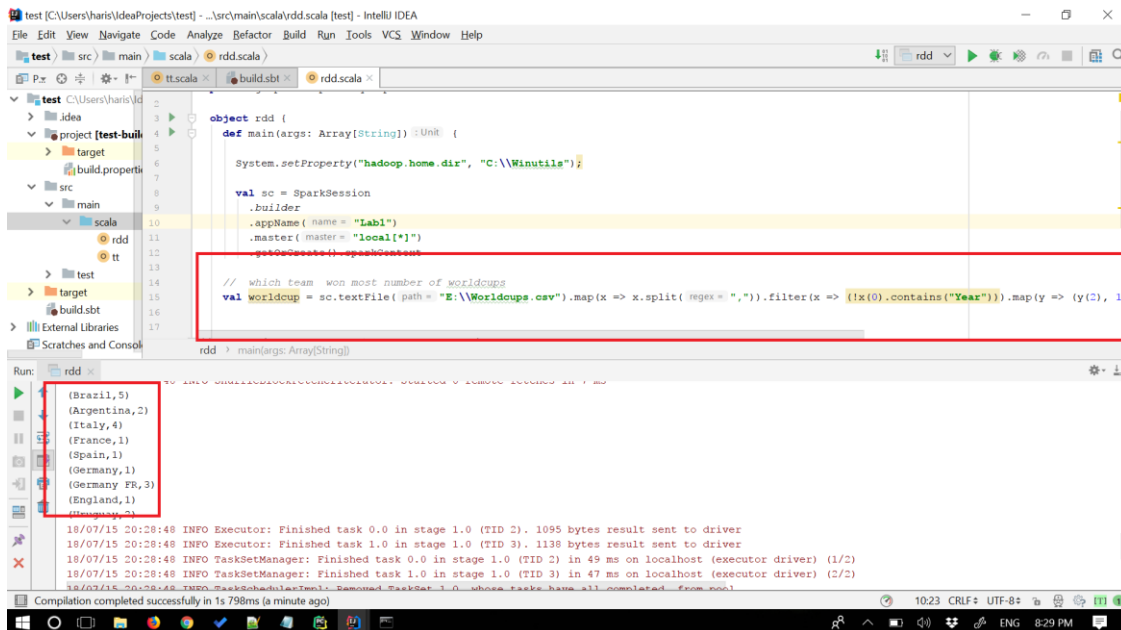
The output line `Number of distinct countries who have won the worldcup:9` is highlighted with a red box. The status bar at the bottom indicates "Compilation completed successfully in 1s 350ms (moments ago)".

## Lab Assignment\_3

### c) Queries in spark RDD's

#### 1) Team won most number of Worldcups

Query: `val worldcup = sc.textFile("E:\Worldcups.csv").map(x => x.split(",")).filter(x => (!x(0).contains("Year"))).map(y => (y(2), 1)).re`



```
object rdd {  
  def main(args: Array[String]) : Unit {  
    System.setProperty("hadoop.home.dir", "C:\\Winutils")  
  
    val sc = SparkSession  
      .builder  
      .appName("Lab1")  
      .master("local[*]")  
      .getOrCreate() // SparkContext  
  
    // which team won most number of worldcups  
    val worldcup = sc.textFile(path = "E:\\Worldcups.csv").map(x => x.split(regex = ",")).filter(x => !x(0).contains("Year")).map(y => (y(2), 1)).reduceByKey(_+_).collect() // RDD of (Team, Wins)  
  
    rdd > main(args: Array[String])  
  }  
}
```

Run: rdd

```
(Brazil,5)  
(Argentina,2)  
(Italy,4)  
(France,1)  
(Spain,1)  
(Germany,1)  
(Germany FR,3)  
(England,1)  
(Uruguay,2)
```

18/07/15 20:28:48 INFO Executor: Finished task 0.0 in stage 1.0 (TID 2). 1095 bytes result sent to driver  
18/07/15 20:28:48 INFO Executor: Finished task 1.0 in stage 1.0 (TID 3). 1138 bytes result sent to driver  
18/07/15 20:28:48 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 49 ms on localhost (executor driver) (1/2)  
18/07/15 20:28:48 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 3) in 47 ms on localhost (executor driver) (2/2)  
18/07/15 20:28:48 INFO TaskScheduler: Removed TaskSet 1.0 whose tasks have all completed from pool  
Compilation completed successfully in 1s 798ms (a minute ago)

## Lab Assignment\_3

### 2) Team that reached most number of finals.

Query: val worldcup =

```
sc.textFile("E:\\Worldcups.csv").map(x=>x.split(",")).filter(x=>(!x(0).contains("Year"))) val  
winner = worldcup.map(y=>(y(2),1)) val runner = worldcup.map(y=>(y(3),1)) val finalists  
= winner.union(runner).reduceByKey((x,y)=>(x+y)).sortBy(_._2,false).foreach(println)
```

```
test [C:\Users\harris\IdeaProjects\test] - ...src\main\scala\rdd.scala [test] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

test > src > main > scala > rdd.scala
val sc = SparkSession
  .builder
  .appName("Lab1")
  .master("local[*]")
  .getOrCreate().sparkContext

// which team won most number of worldcups
val worldcup = sc.textFile("E:\\Worldcups.csv").map(x => x.split(",")).filter(x => (!x(0).contains("Year")))

// which team reached most number of finals
val worldcup = sc.textFile(path = "E:\\Worldcups.csv").map(x=>x.split(regex = ",")).filter(x=>(!x(0).contains("Year")))
val winner = worldcup.map(y=>(y(2),1))
val runner = worldcup.map(y=>(y(3),1))
val finalists = winner.union(runner).reduceByKey((x,y)=>(x+y)).sortBy(_._2, ascending = false).foreach(println)

Run: rdd
18/07/15 20:30:47 INFO ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 4 blocks
18/07/15 20:30:47 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/07/15 20:30:47 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/07/15 20:30:47 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
(Brazil, 7)
(Argentina, 5)
(Netherlands, 3)
(Germany FR, 6)
(Hungary, 2)
(Italy, 6)
(Uruguay, 2)
(Czechoslovakia, 2)
(France, 2)
(Germany, 2)
(Sweden, 1)

Compilation completed successfully in 813ms (moments ago)
16:3 CRLF UTF-8 ENG 8:31 PM
```



## Lab Assignment\_3

### 3) Players with most number of matches

Query: val worldcup =

sc.textFile("E:\Players.csv").map(x=>x.split(",")).filter(x=>(!x(0).contains("RoundID"))) val

views = worldcup.map(x=>(x(6),1)).reduceByKey((x,y)=>

(x+y)).coalesce(1).sortBy(\_.\_2,false).take(20).foreach(println)

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `rdd.scala` open. The code defines a `worldcup` RDD from a CSV file, filters out rows containing "RoundID", and then maps each row to a tuple of a player name and a value of 1. These are then reduced by key to get the total number of matches for each player. The results are sorted by the number of matches in descending order and the top 20 are printed. A red box highlights the code for creating `worldcup` and `views`.

```
val winner = worldcup.map(y=>(y(2),1))
val runner = worldcup.map(y=>(y(3),1))
val finalists = winner.union(runner).reduceByKey((x,y)=>(x+y)).sortBy(_._2,false).foreach(println)

// Players with most number of matches
val worldcup = sc.textFile(path = "E:\Players.csv").map(x=>x.split(regex = ",")).filter(x=>(!x(0).contains("RoundID")))
val views = worldcup.map(x=>(x(6),1)).reduceByKey((x,y)=>(x+y)).coalesce(numPartitions = 1).sortBy(_._2, ascending = false).take(num = 20).foreach(println)
```

The Run console shows the output of the program, listing the top 20 players and their match counts. A red box highlights the output list.

```
(RONALDO, 33)
(RLOSE, 32)
(OSCAR, 28)
(MOLLER, 28)
(CAFU, 26)
(JULIO CESAR, 26)
(LAHM, 25)
(SCHWEINSTEIGER, 25)
(MERTESACKER, 25)
(DIDA, 25)
(Sepp MAIER, 25)
(PODOLSKI, 25)
(SILVA, 25)
(LEAO, 25)
```

Compilation completed successfully in 1s 821ms (a minute ago)

## Lab Assignment\_3

### 4) Countries which won Worldcup as hosting countries

Query: val worldcup =

```
sc.textFile("E:\\Worldcups.csv").map(x=>x.split(",")).filter(x=>(!x(0).contains("Year"))) val  
hostwinners = worldcup.filter(x=>(x(1)==x(2))).map(x=>(x(0),x(1))).foreach(println)
```

The screenshot shows the IntelliJ IDEA IDE with a Scala file named `rdd.scala` open. The code defines a SparkContext `sc` and reads a CSV file `E:\\Worldcups.csv`. It filters out rows containing "Year" and then filters for rows where the first column (Year) equals the second column (Host). The results are printed as tuples.

```
18/07/15 20:55:07 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
18/07/15 20:55:08 INFO HadoopRDD: Input split: file:/E:/Worldcups.csv:0+706
18/07/15 20:55:08 INFO HadoopRDD: Input split: file:/E:/Worldcups.csv:706+706
(1930,Uruguay)
(1978,Argentina)
(1934,Italy)
(1998,France)
(1966,England)
18/07/15 20:55:08 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 794 bytes result sent to driver
18/07/15 20:55:08 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 794 bytes result sent to driver
18/07/15 20:55:08 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 137 ms on localhost (executor driver) (1/2)
18/07/15 20:55:08 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 126 ms on localhost (executor driver) (2/2)
18/07/15 20:55:08 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/07/15 20:55:08 INFO DAGScheduler: ResultStage 0 (foreach at rdd.scala:32) finished in 0.230 s
```

Compilation completed successfully in 1s 786ms (a minute ago)

## Lab Assignment\_3

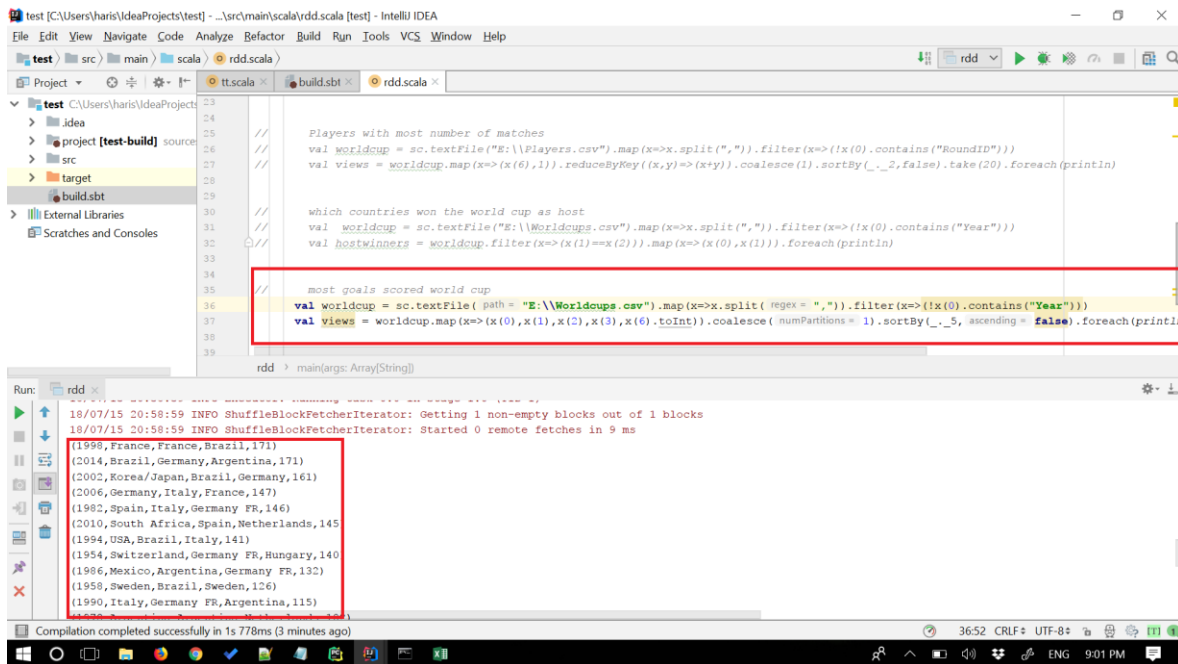
### 5) Most goals scored in a worldcup

Query: val worldcup =

sc.textFile("E:\\Worldcups.csv").map(x=>x.split(",")).filter(x=>(!x(0).contains("Year"))) val

views =

worldcup.map(x=>(x(0),x(1),x(2),x(3),x(6).toInt)).coalesce(1).sortBy(\_.\_5,false).foreach(print  
ln)



The screenshot shows the IntelliJ IDEA IDE with a Scala file named `rdd.scala` open. The code defines a `worldcup` RDD and a `views` RDD. The `views` RDD is created by mapping the `worldcup` RDD to a tuple of (country, opponent, year, goals, total goals) and then coalescing the results. The code is executed, and the output is shown in the Run console. The output lists the top 10 teams with the most goals scored in a World Cup.

```
val worldcup = sc.textFile("E:\\Worldcups.csv").map(x=>x.split(",")).filter(x=>(!x(0).contains("Year")))
val views = worldcup.map(x=>(x(0),x(1),x(2),x(3),x(6).toInt)).coalesce(1).sortBy(_._5,false).foreach(println)
```

```
(1998, France, France, Brazil, 171)
(2014, Brazil, Germany, Argentina, 171)
(2002, Korea/Japan, Brazil, Germany, 161)
(2006, Germany, Italy, France, 147)
(1982, Spain, Italy, Germany FR, 146)
(2010, South Africa, Spain, Netherlands, 145)
(1994, USA, Brazil, Italy, 141)
(1954, Switzerland, Germany FR, Hungary, 140)
(1986, Mexico, Argentina, Germany FR, 132)
(1958, Sweden, Brazil, Sweden, 126)
(1990, Italy, Germany FR, Argentina, 115)
```

### References:

1. <https://www.kaggle.com/abecklas/fifa-world-cup/version/5>
2. <https://snap.stanford.edu/data/egonets-Facebook.html>
3. [https://www.tutorialspoint.com/spark\\_sql/spark\\_sql\\_dataframes.htm](https://www.tutorialspoint.com/spark_sql/spark_sql_dataframes.htm)
4. <http://spark.apache.org/docs/latest/sql-programming-guide.html>