**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)

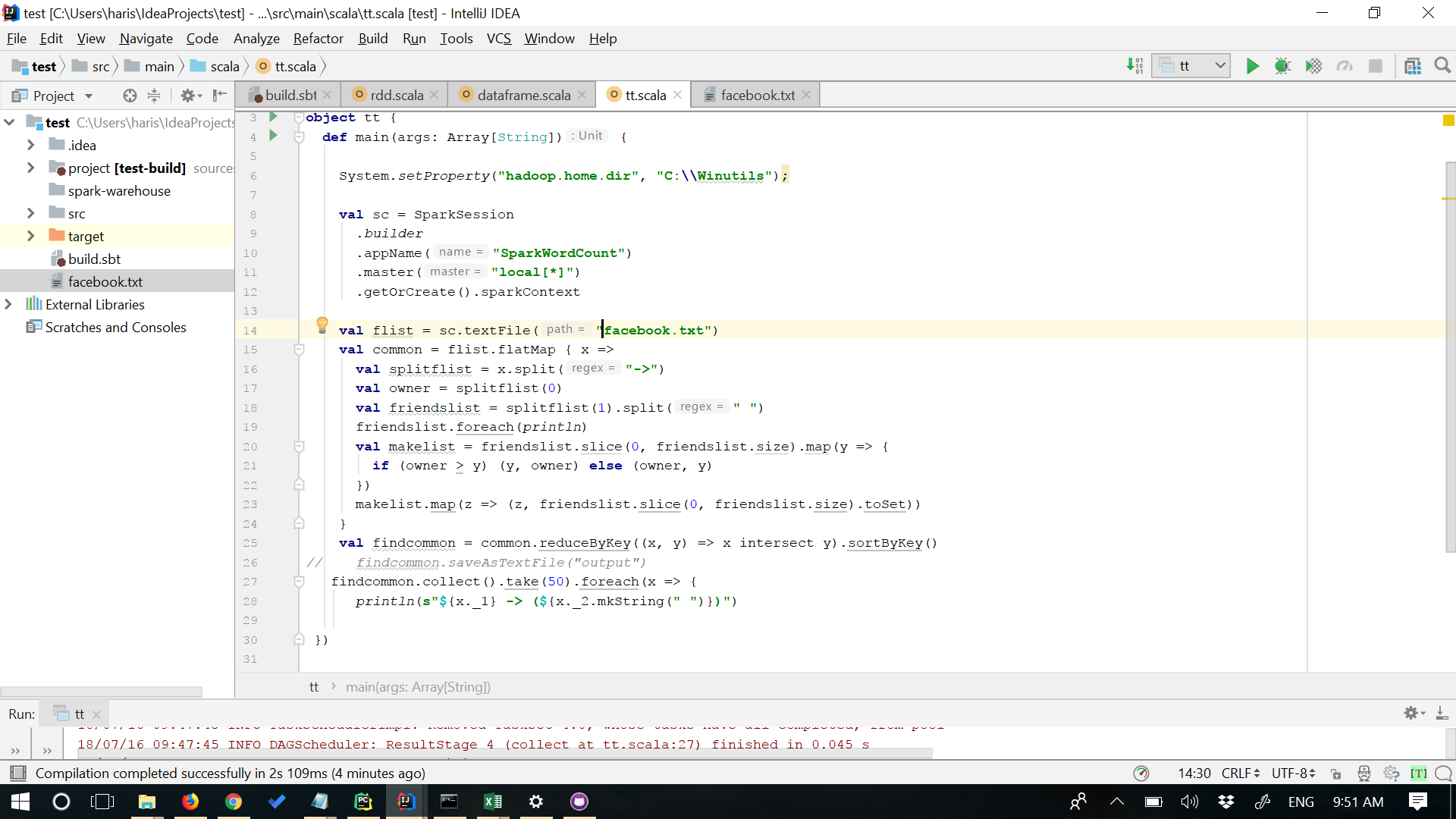
**Video link: https://youtu.be/J8h5q\_x6trc**

**TASK 1:Hadoop MapReduce Algorithm**

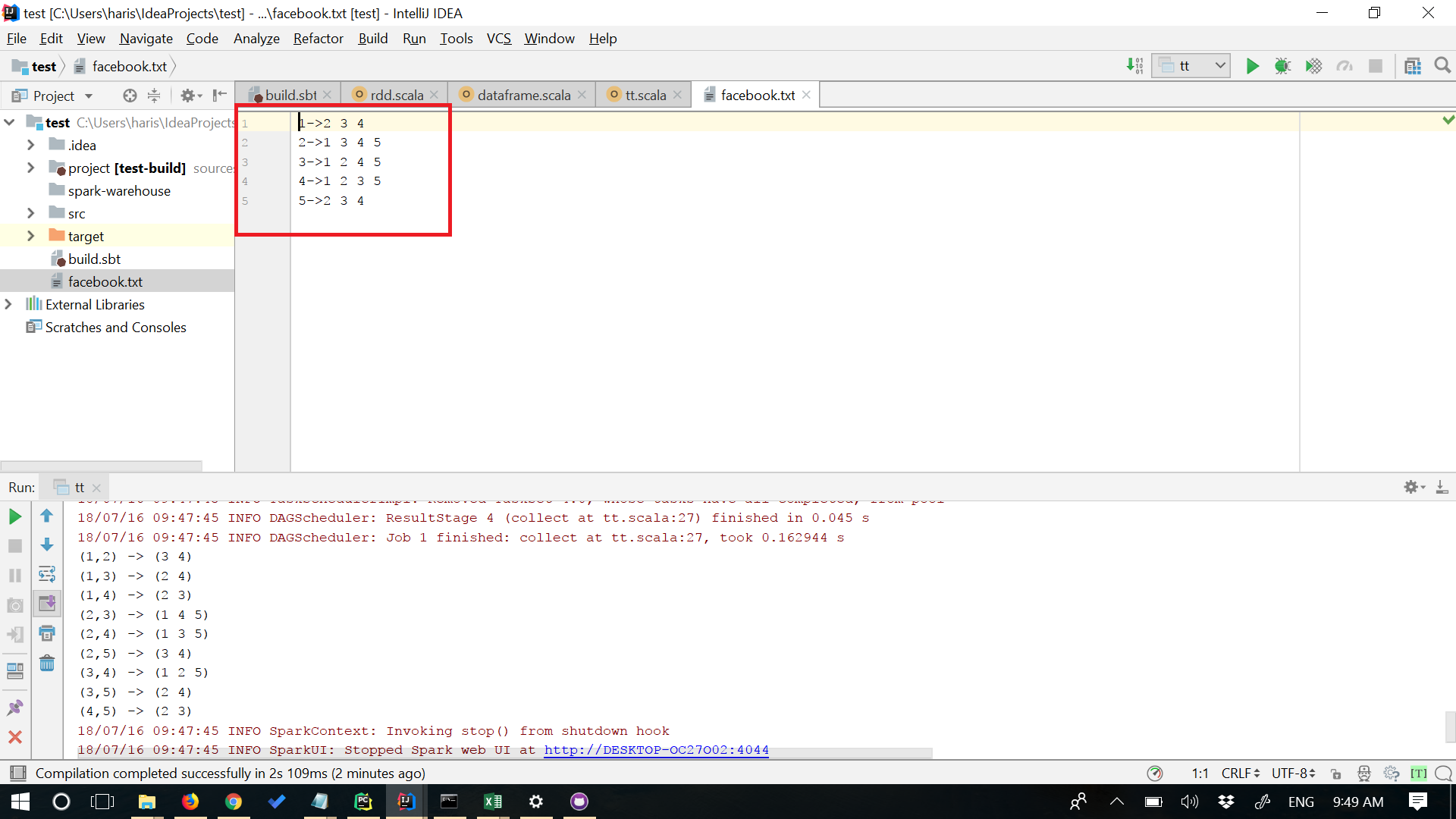
**Objective:**

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

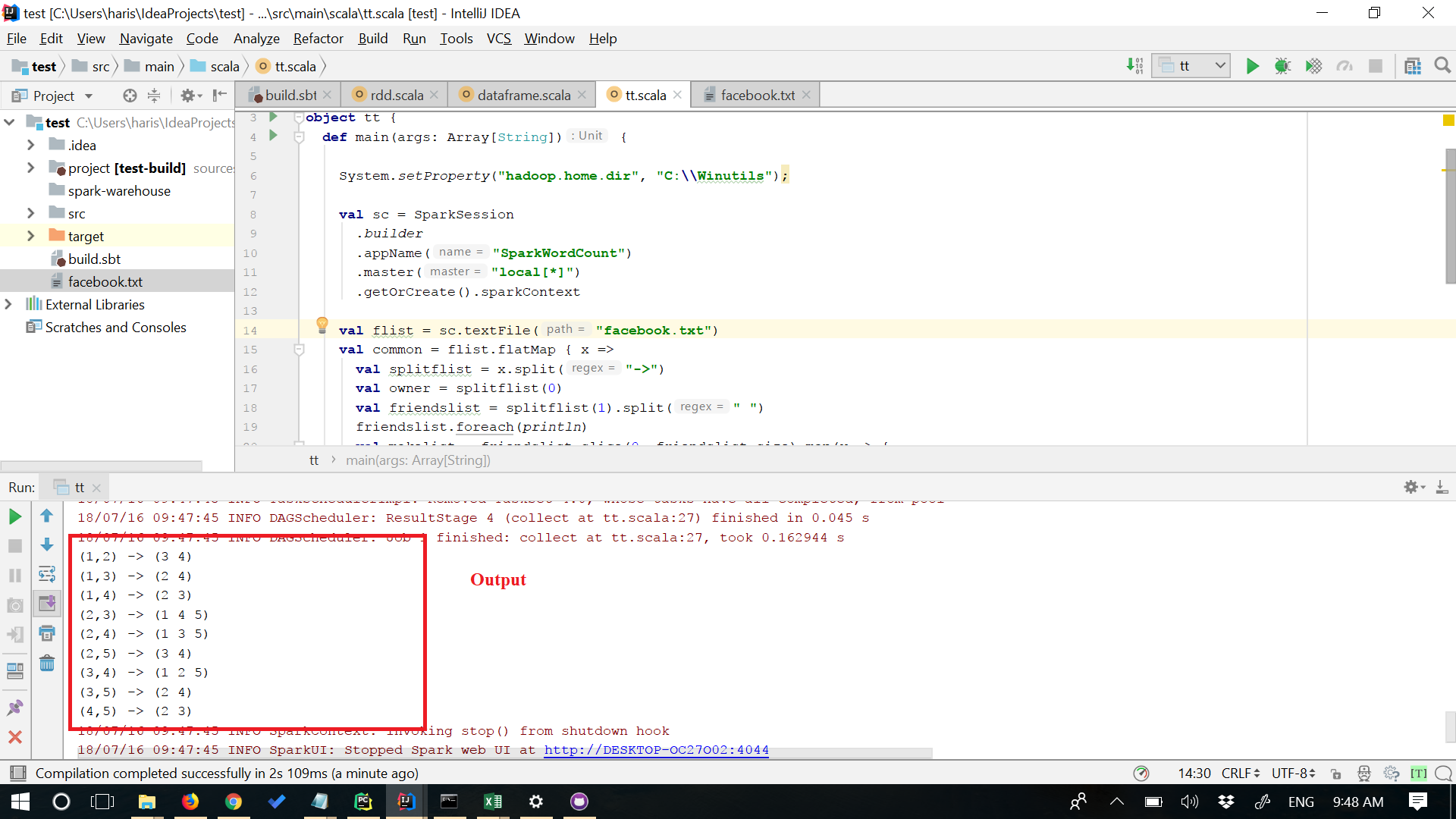
**Execution of Mapreduce:**



**Input:**



**Output:**



**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:**

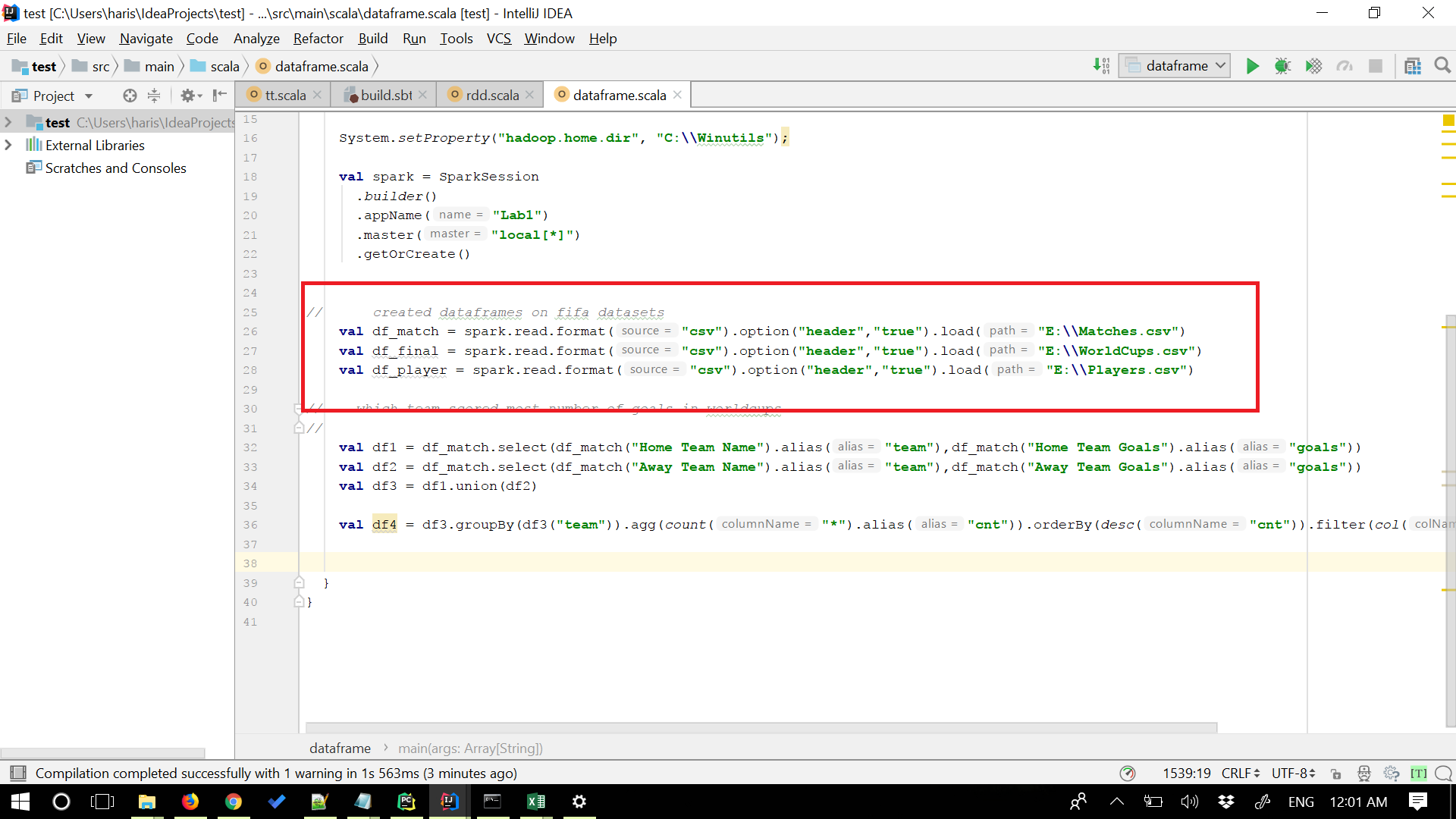
a) Creating a DataFrame from the datasets

b) Performing 10 Intutive questions

c) Queries in spark RDD's

Datasets Used:

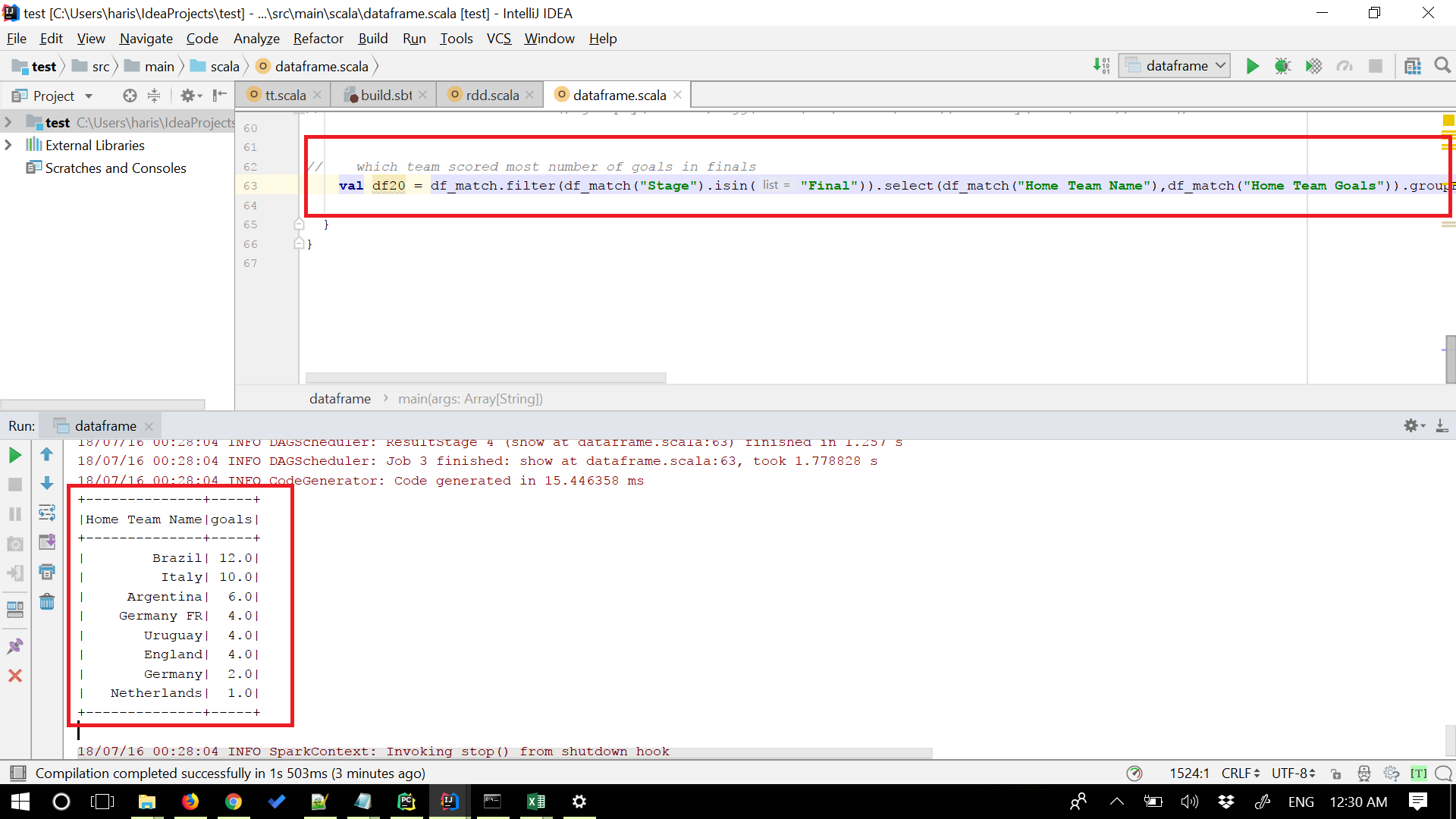
**a) Creating a DataFrame from the datasets**



**b) Performing 10 Intutive questions**

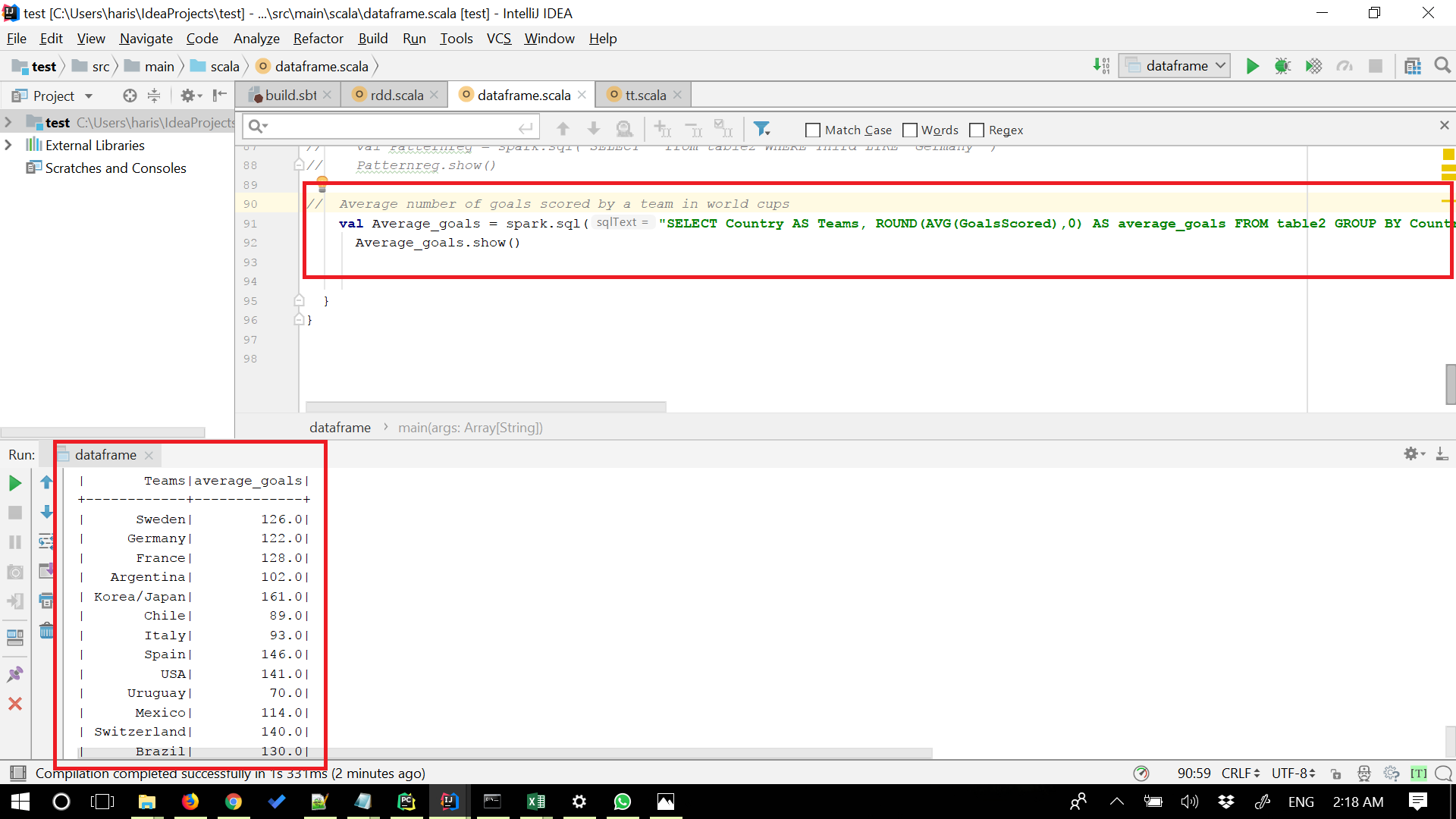
**1) Teams that scored most number of goals in finals**

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()



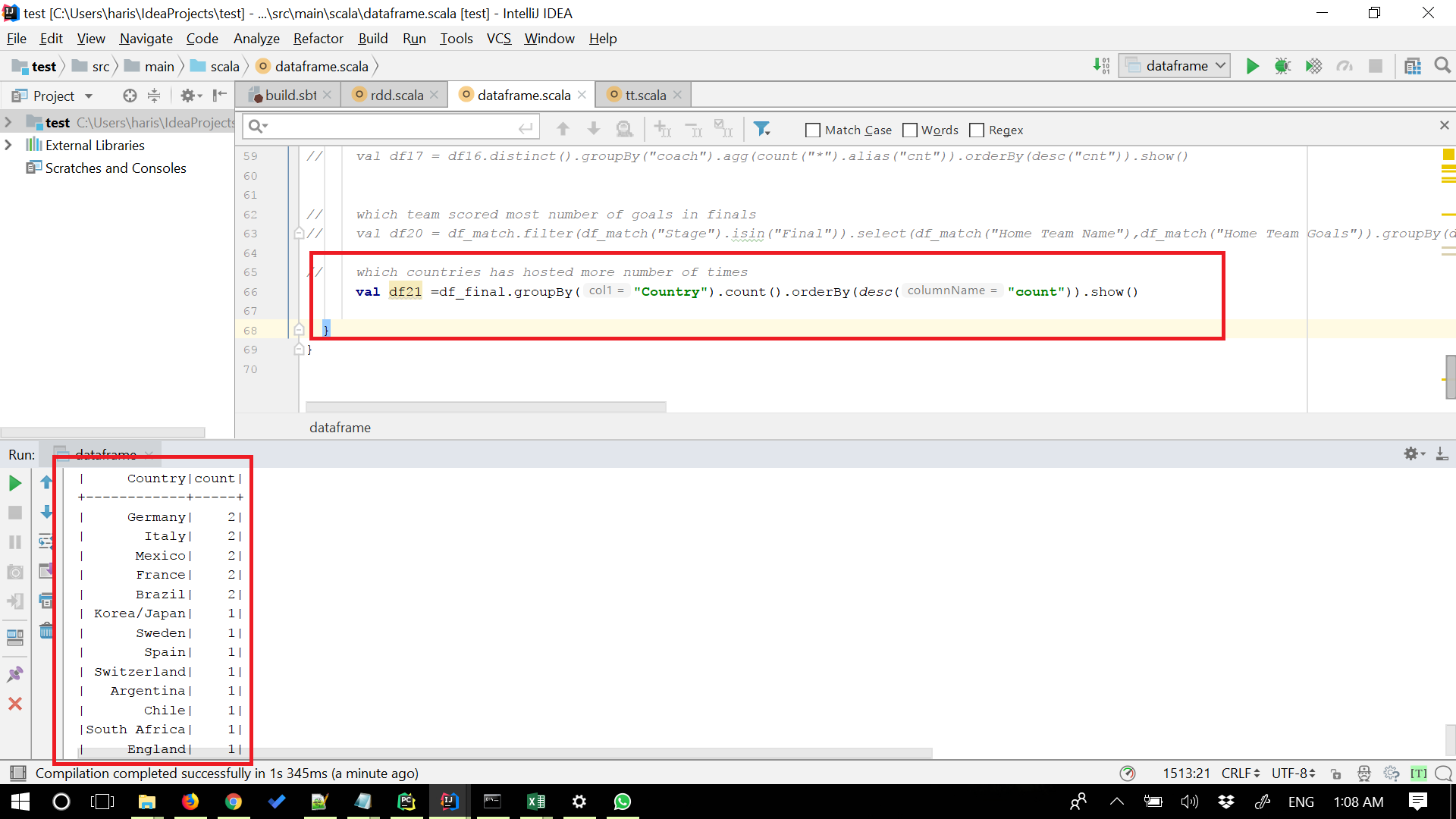
**2) Average number of goals scored by a team in worldcups**

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()



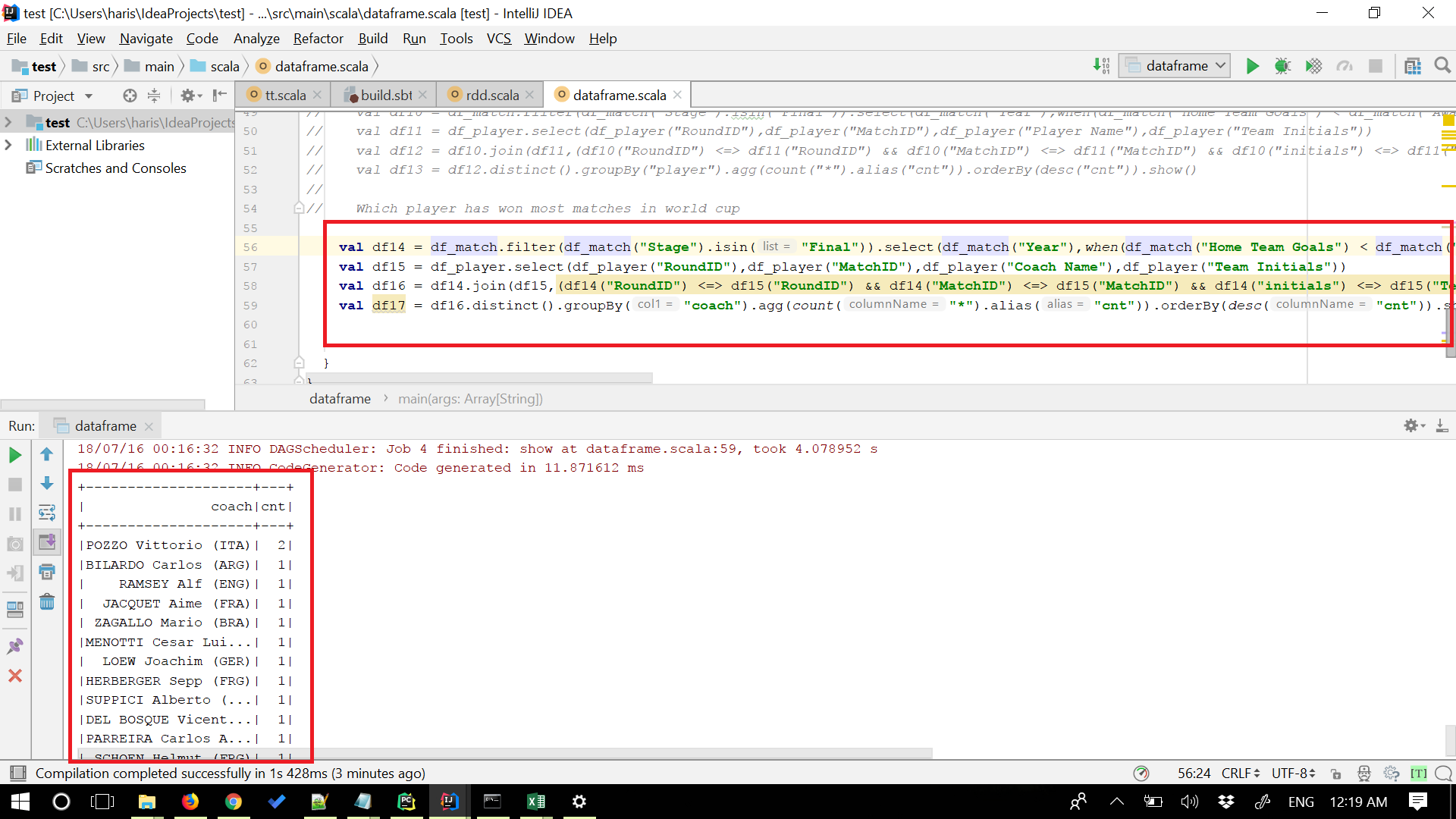
**3) Most number of time a country has hosted Worldcups**

Query: val df21 =df\_final.groupBy("Country").count().orderBy(desc("count")).show()



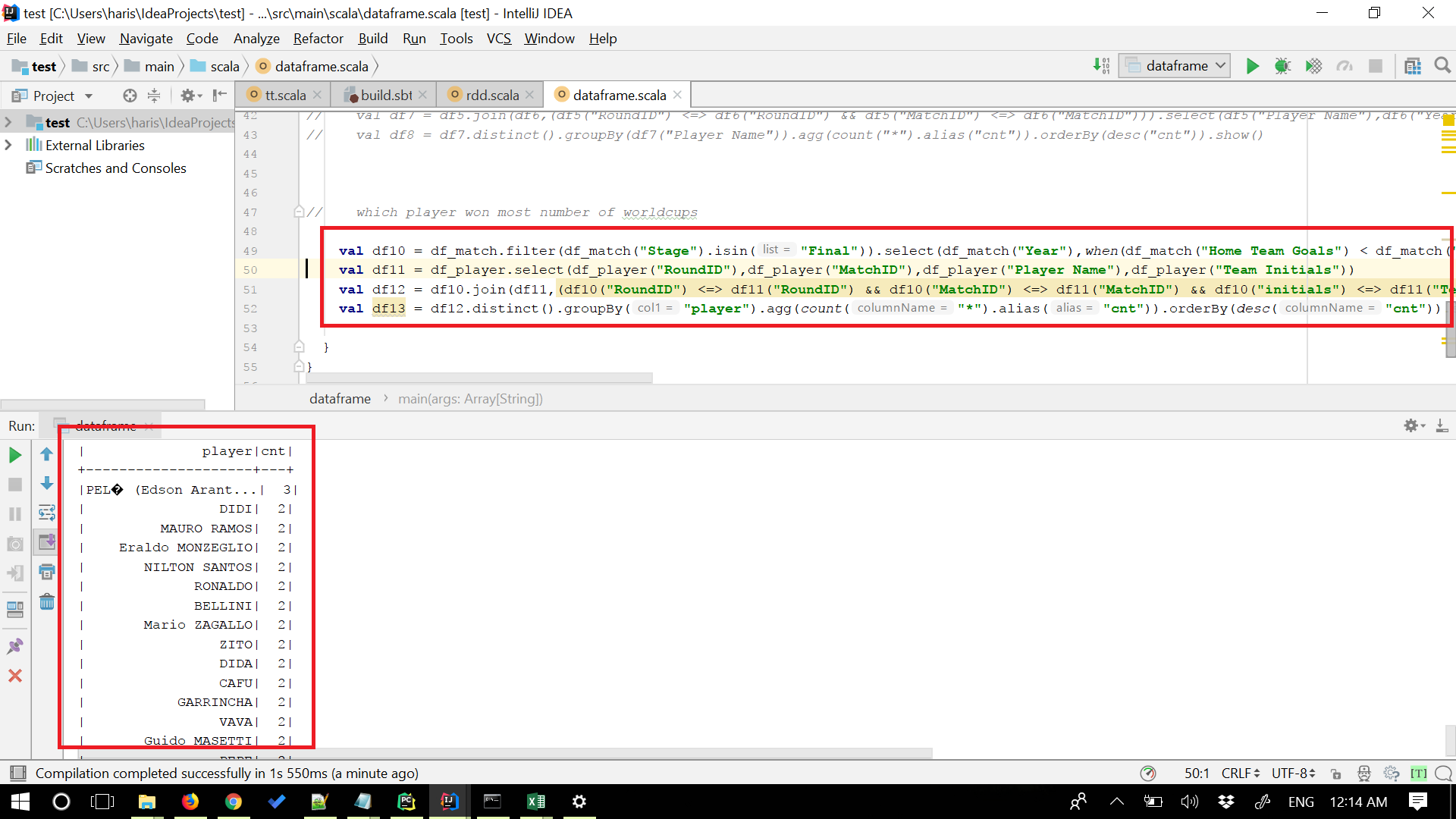
**4) Player who has won most matches in Worldcup**

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()



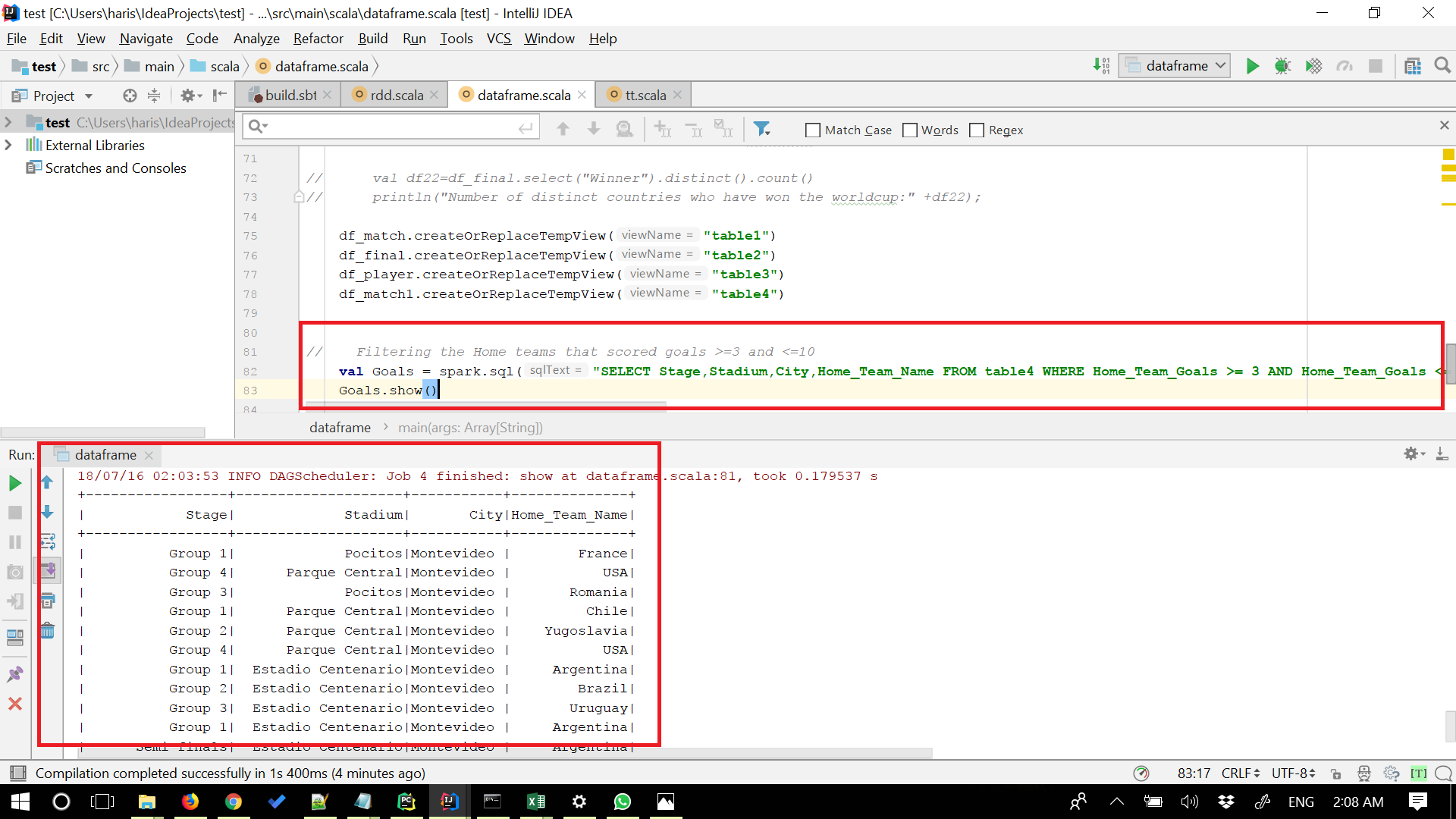
**5) Players who won most number of Worldcups**

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()



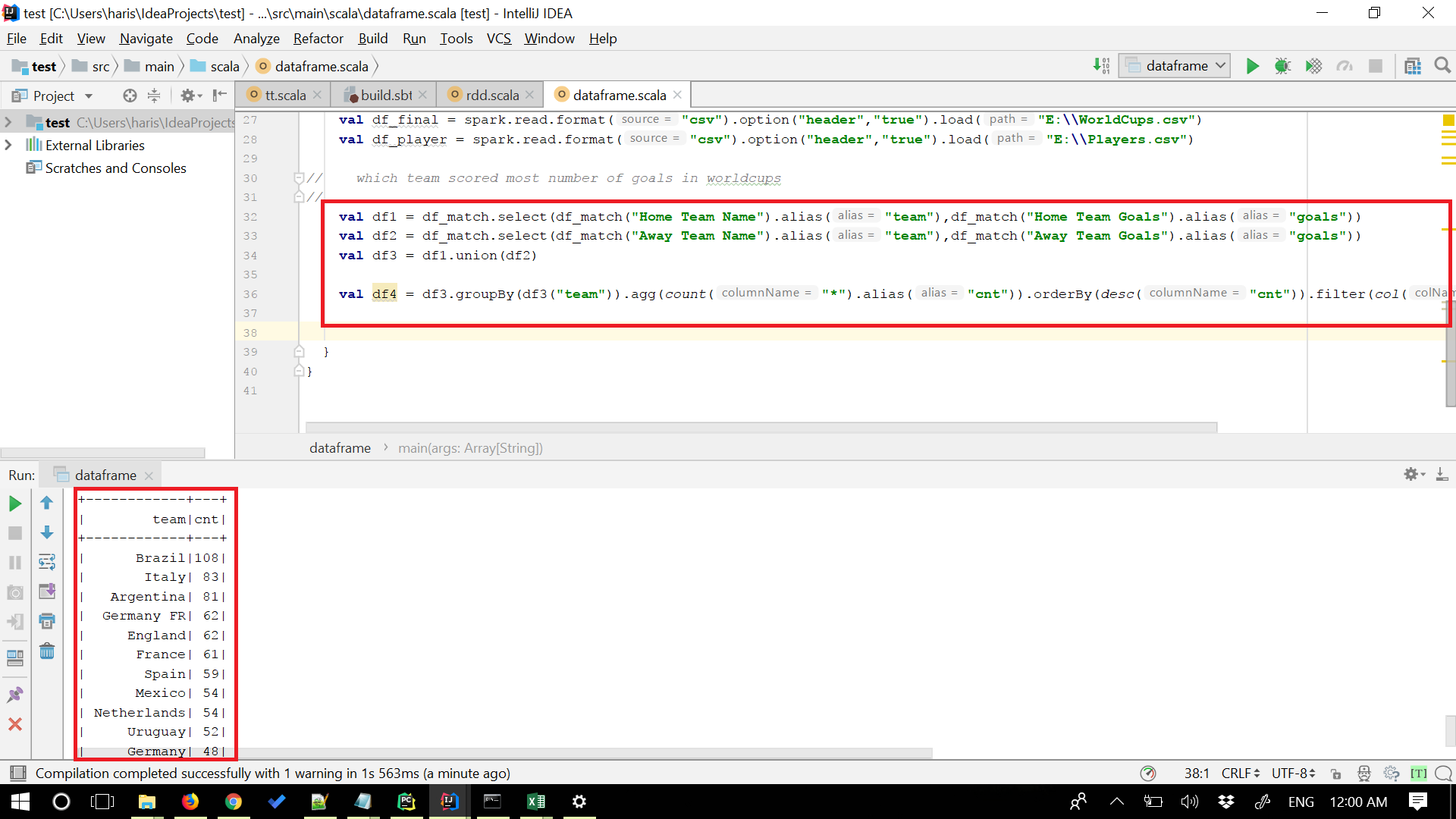
**6) Home team who scored goals greater than 3**

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()



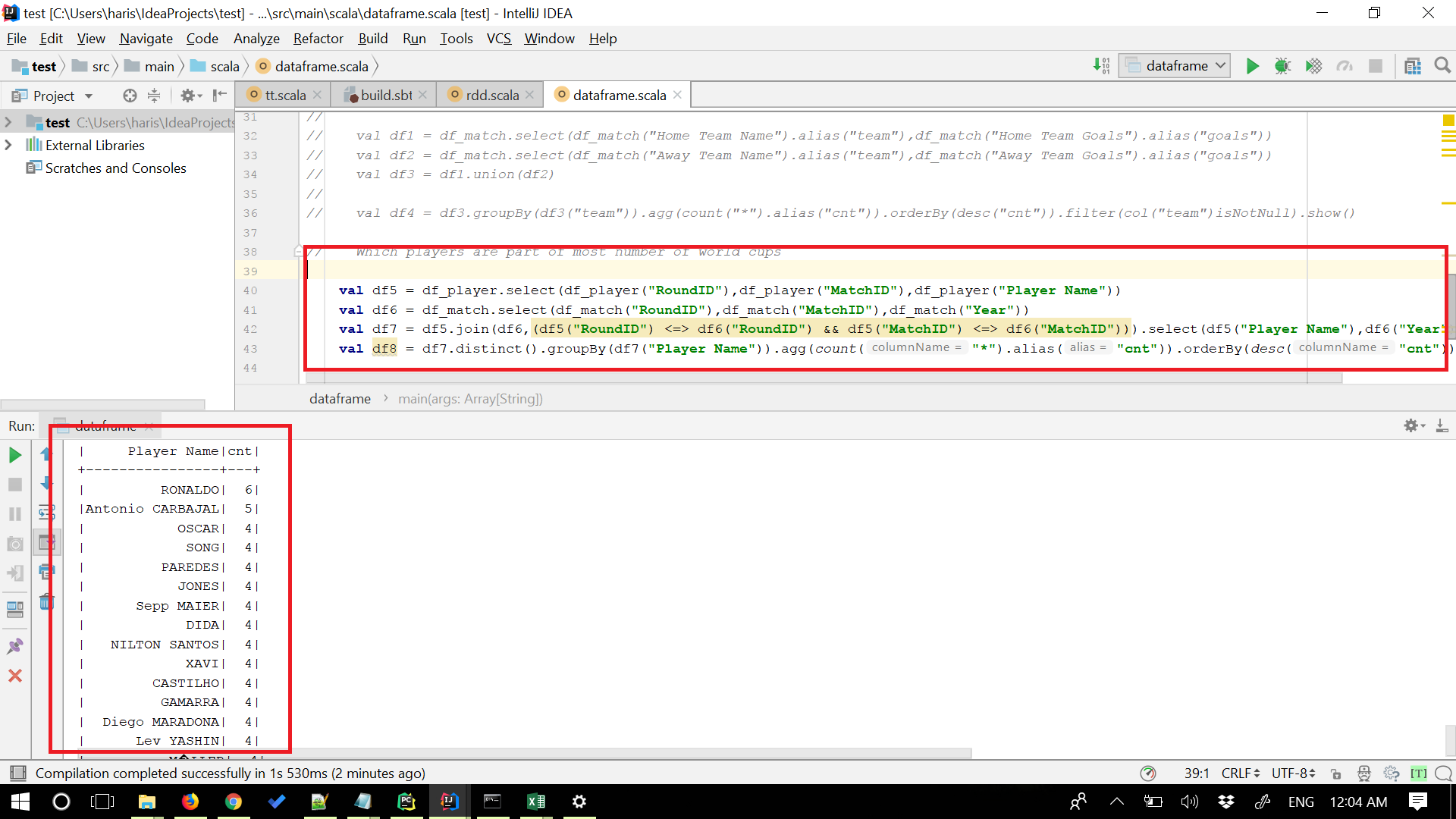
**7) Teams scored most number of goals in Worldcups**

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()



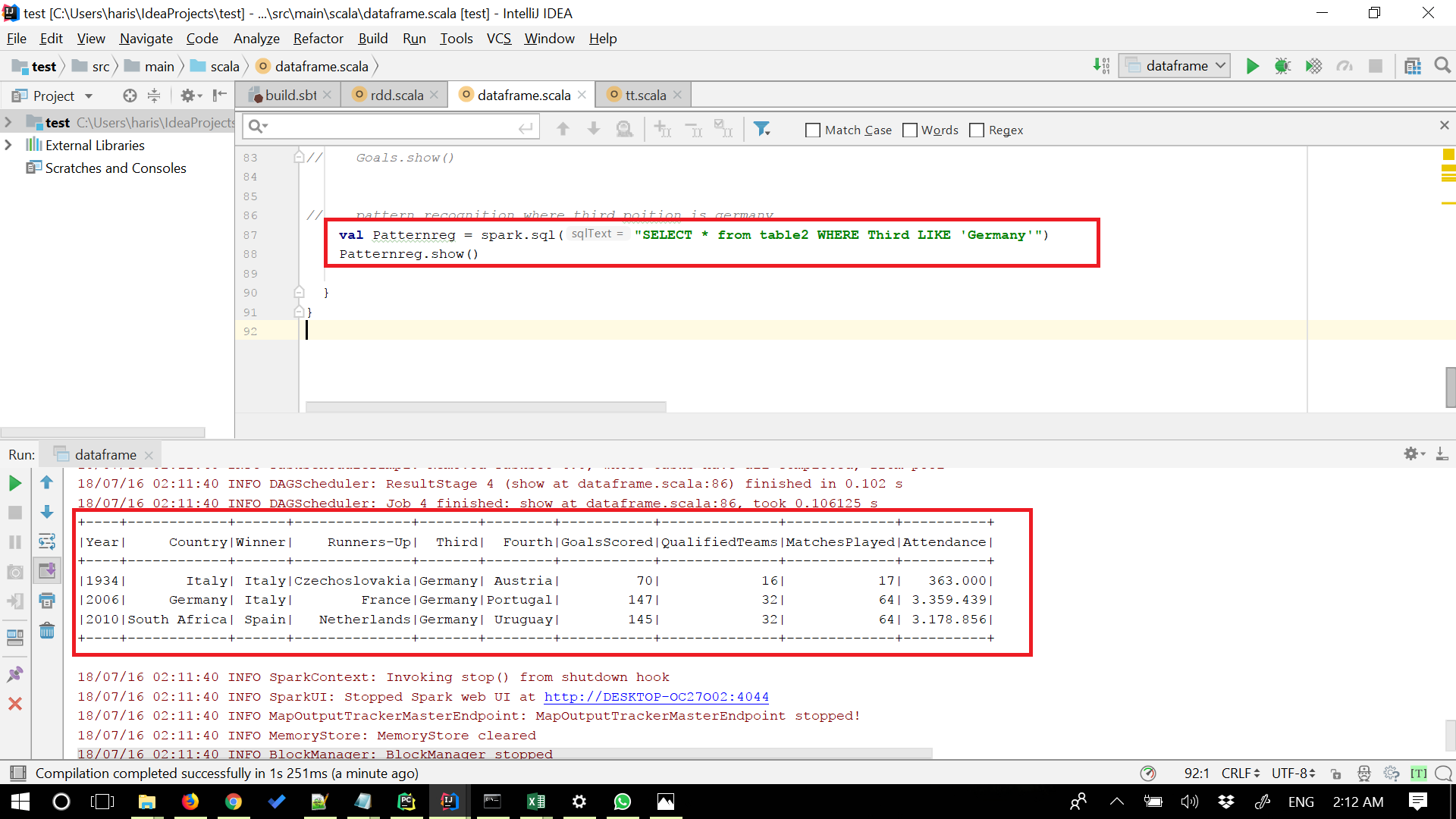
**8) Players part of most number of Worldcups**

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()



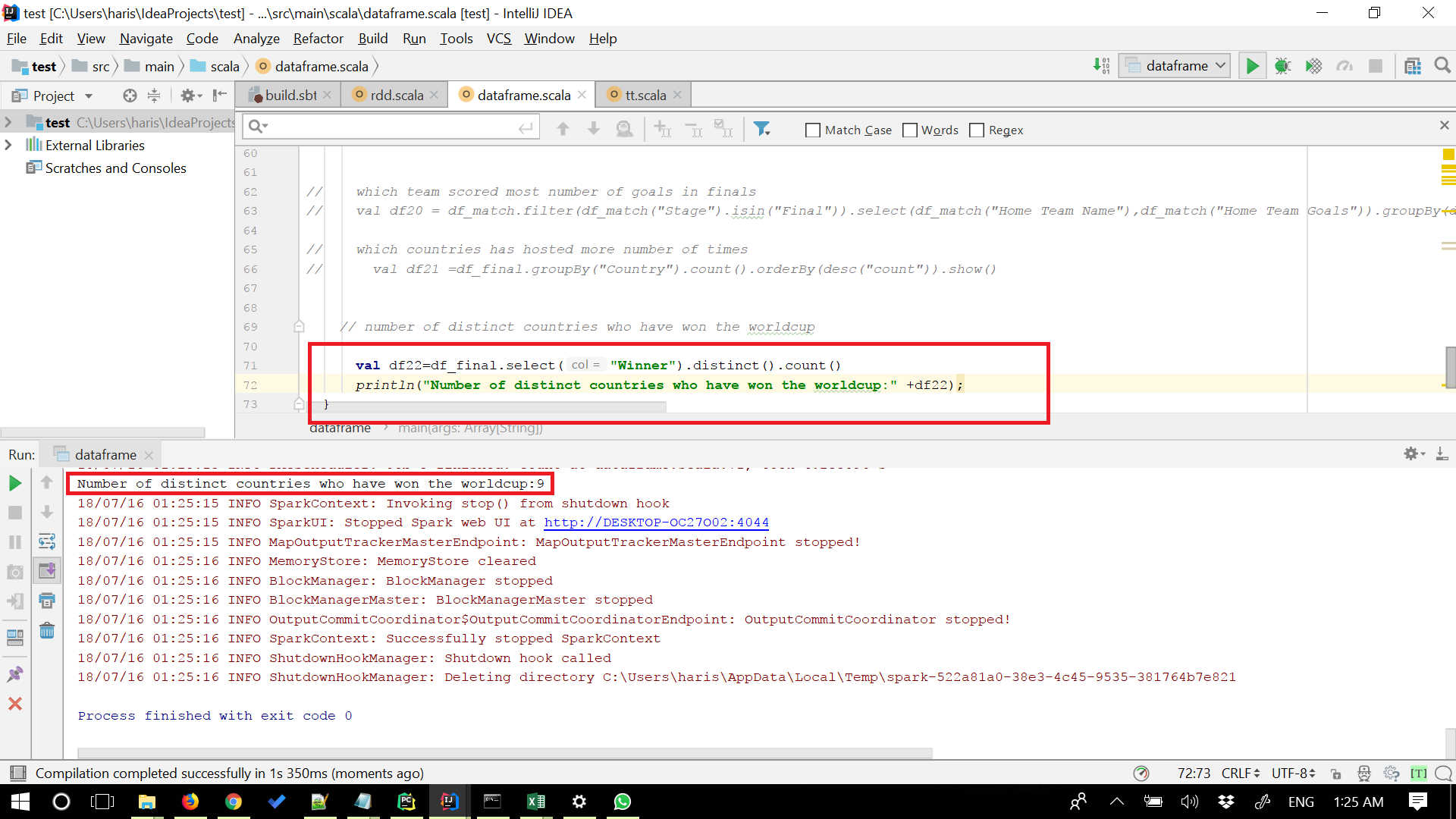
**9) pattern Recognition Germany**

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



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

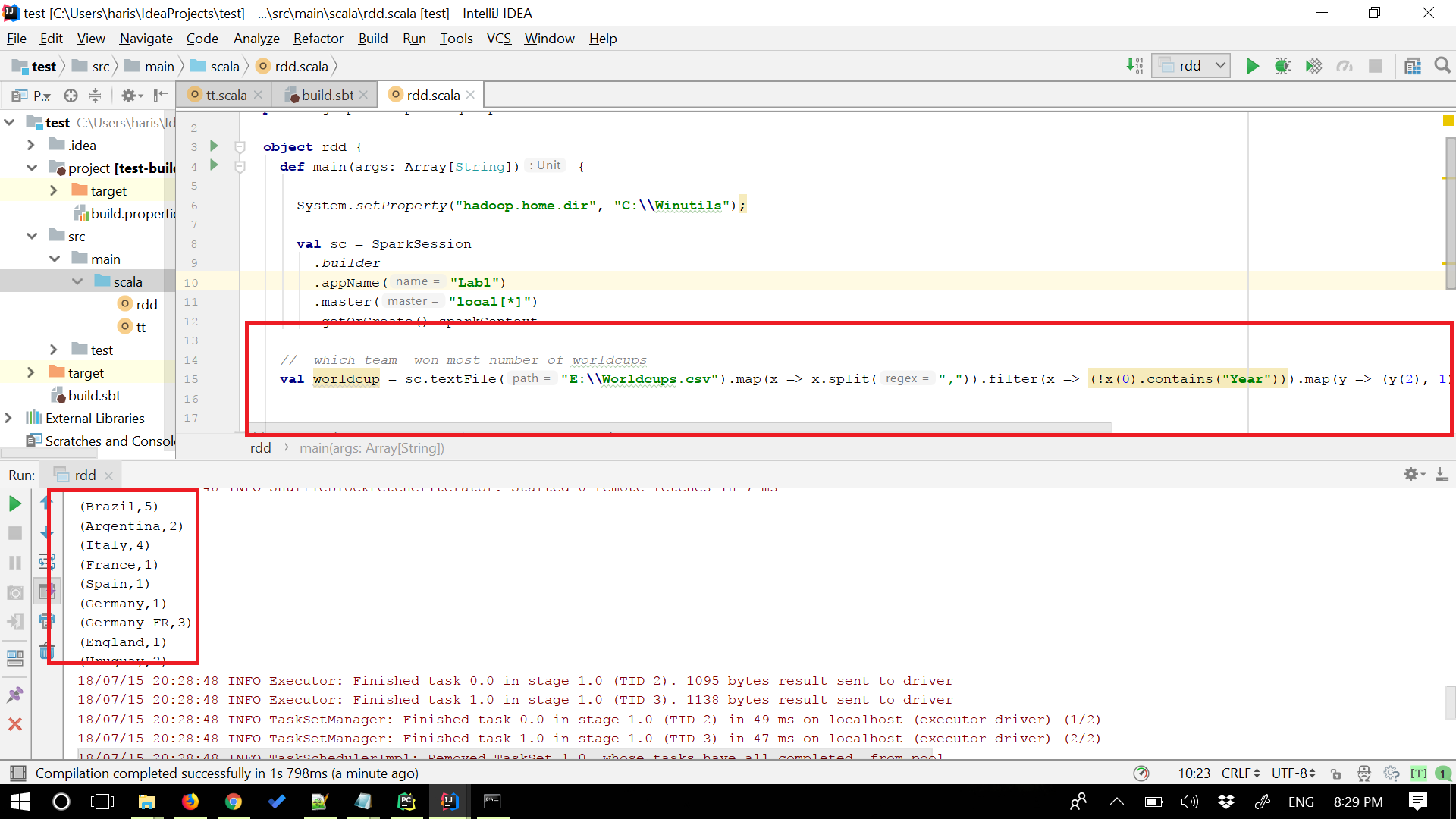
Query: val df22=df\_final.select("Winner").distinct().count() println("Number of distinct countries who have won the worldcup:" +df22);



**c) Queries in spark RDD's**

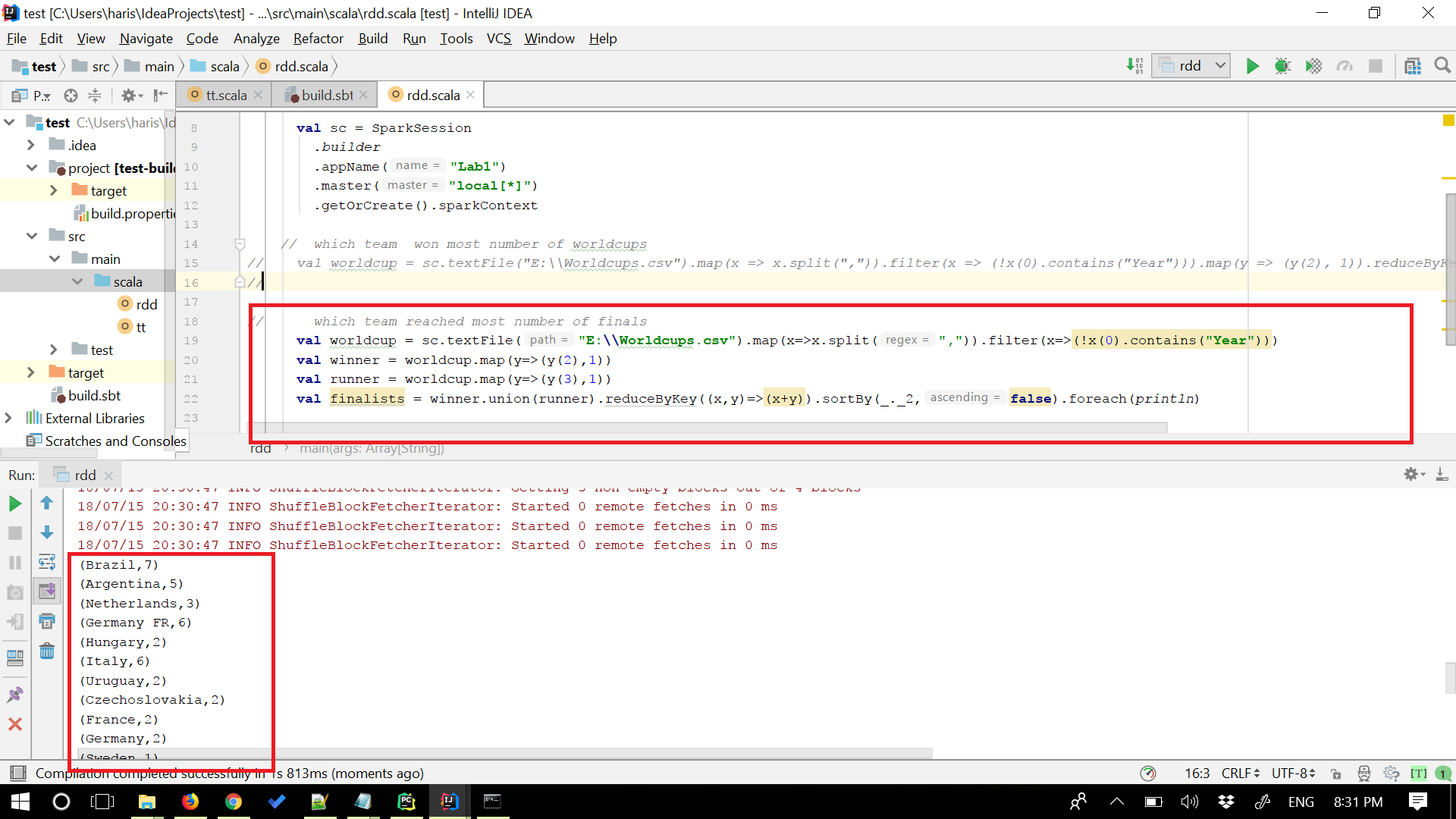
**1) Team won most number of Worldcups**

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



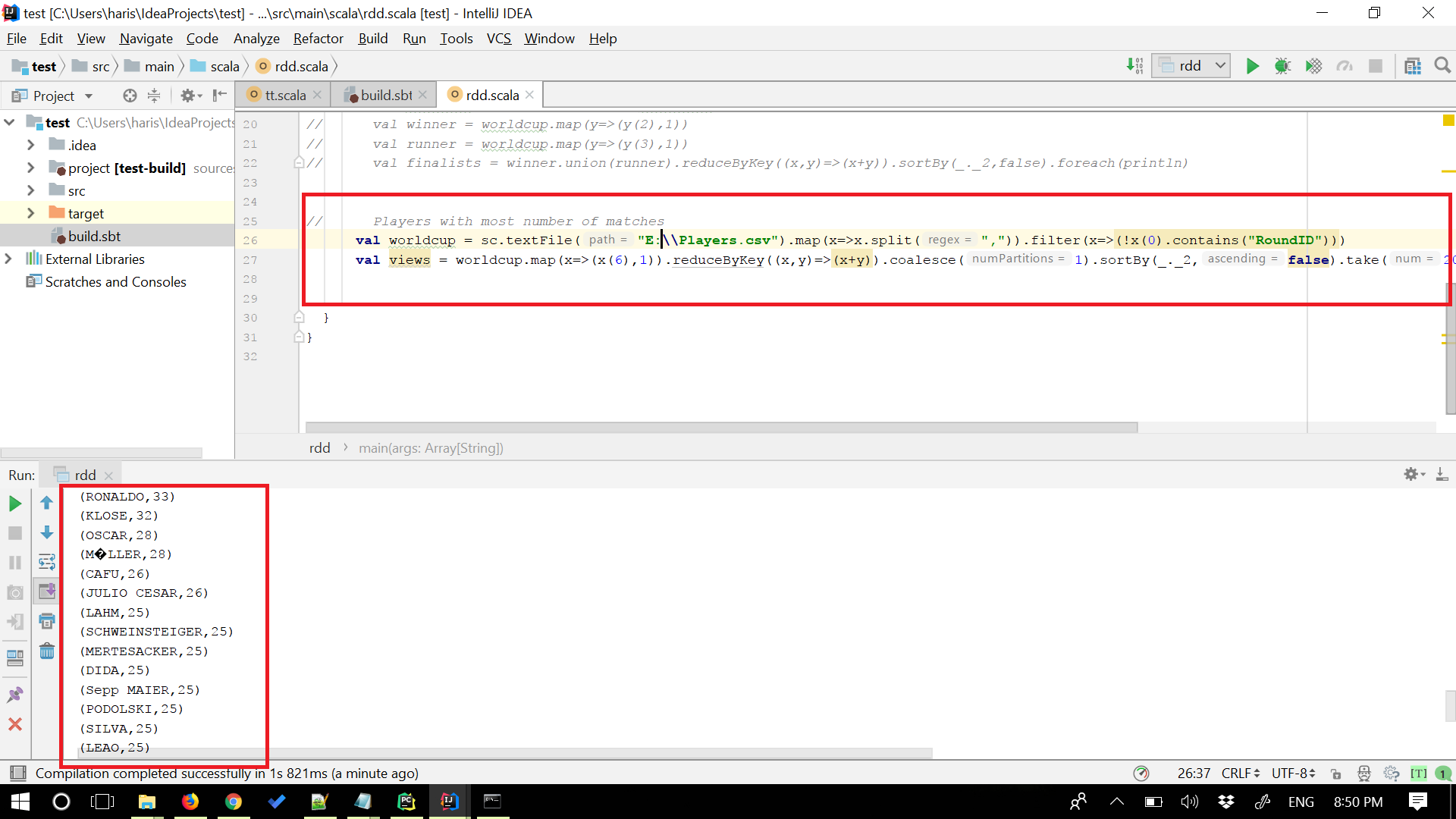
**2) Team that reached most number of finals.**

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)



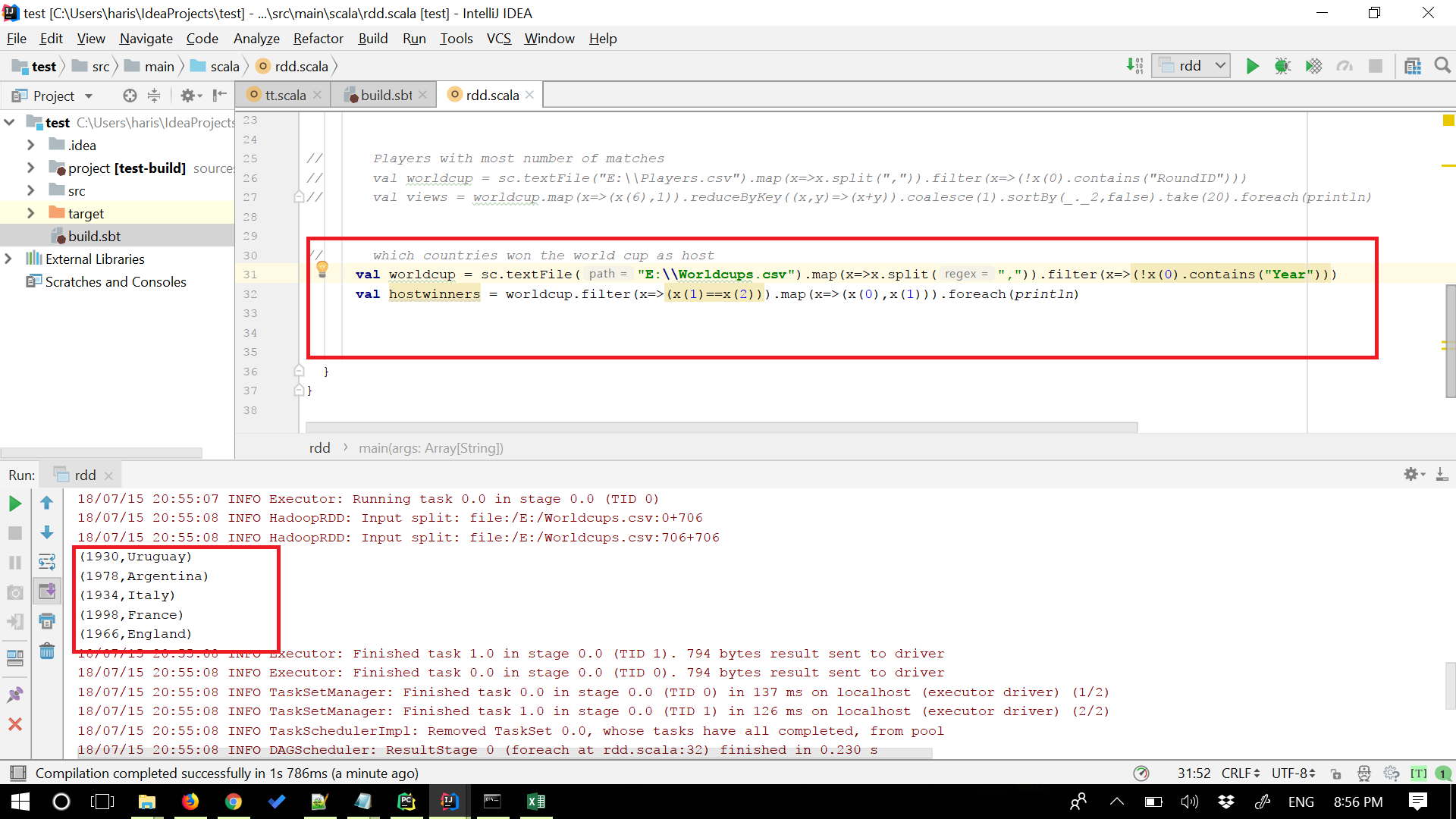
**3)Players with most number of matches**

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)



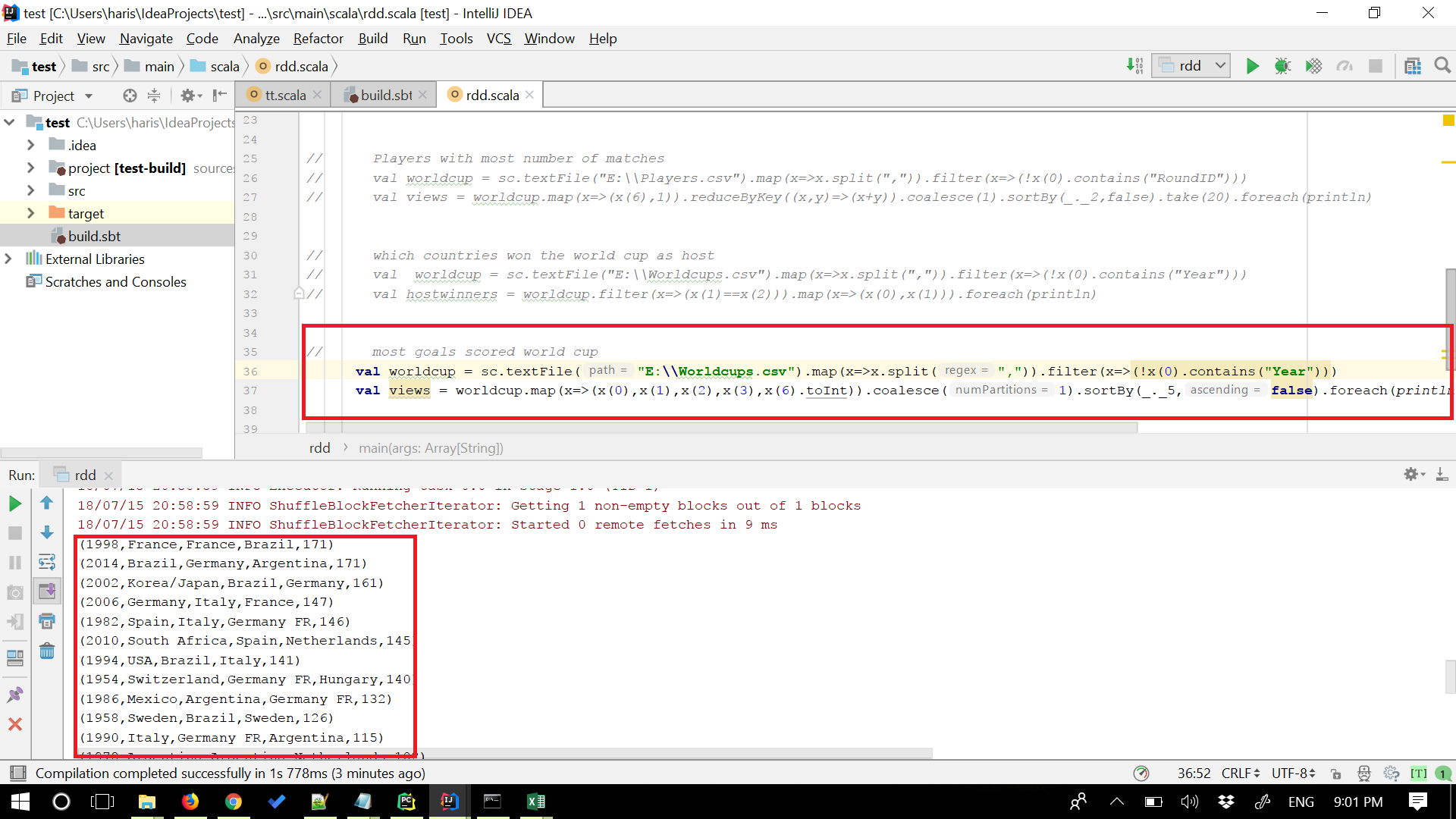
**4) Countries which won Worldcup as hosting countries**

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)



**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(println)



**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>
4. <http://spark.apache.org/docs/latest/sql-programming-guide.html>