**Module2-Spark Lab Assignment-1**

**Name – Aravind Sheri, Class ID – 18**

**Name – Durgam Dileep, Class ID – 03**

**Team – 06**

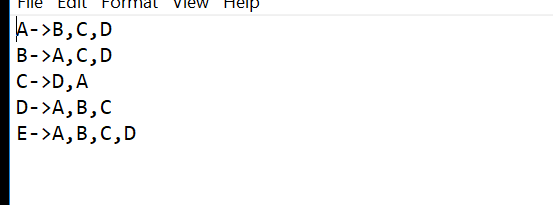
1. **Apache Spark MapReduce Algorithm for Finding mutual friends between users for the better search and save the time.**

**Approach:** Read the data from an input text file which contains user and their corresponding friends, and we need to read the data as lines and perform the mapper and reducer phases on the data. The output consists of a data which contains two users lists and the corresponding mutual friends.

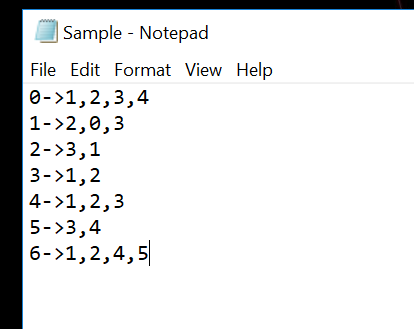
**Inputs:** Input is given as a text file which consists of the data of the users and their mutual friends.

**Screenshots:**

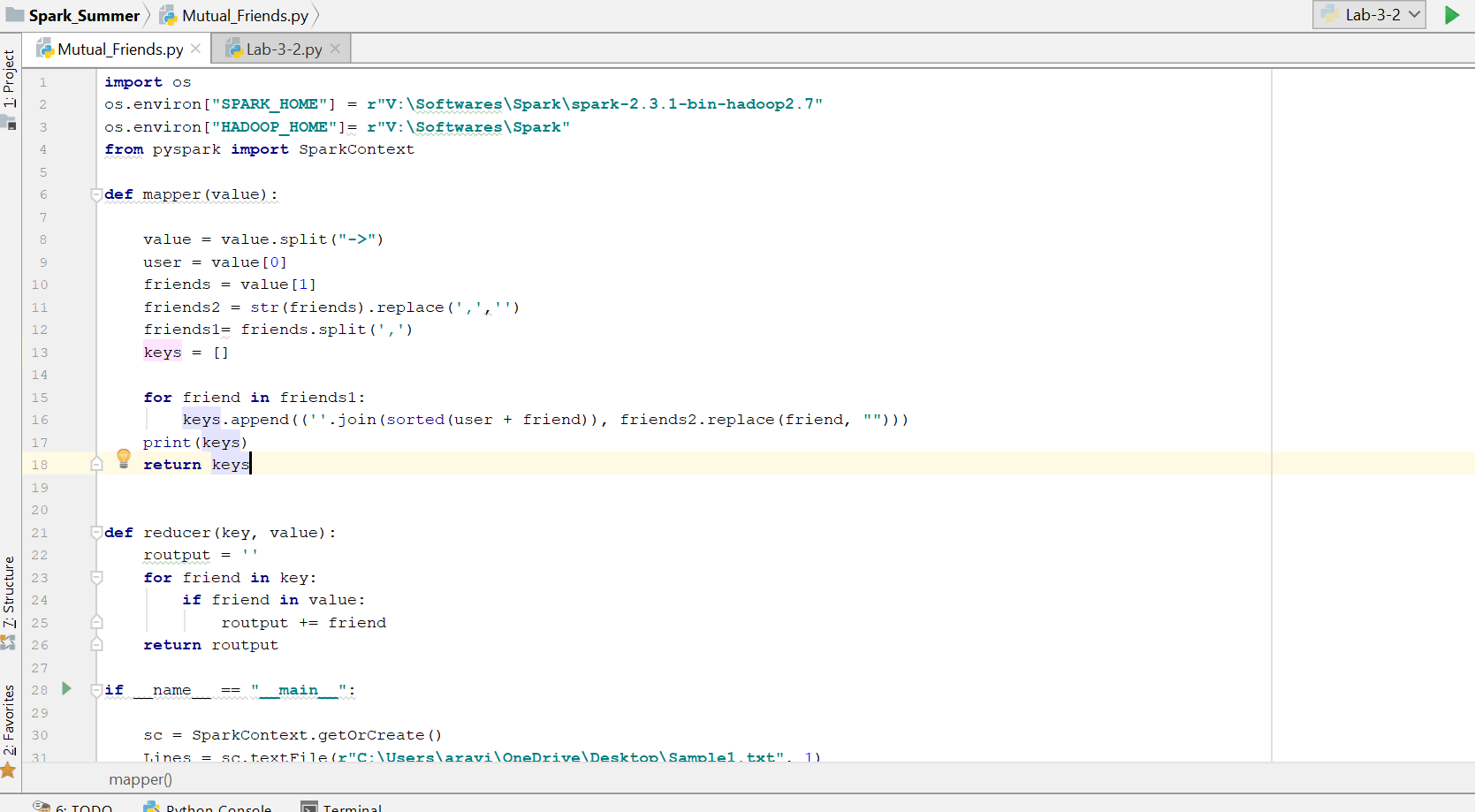
**Input 1:**

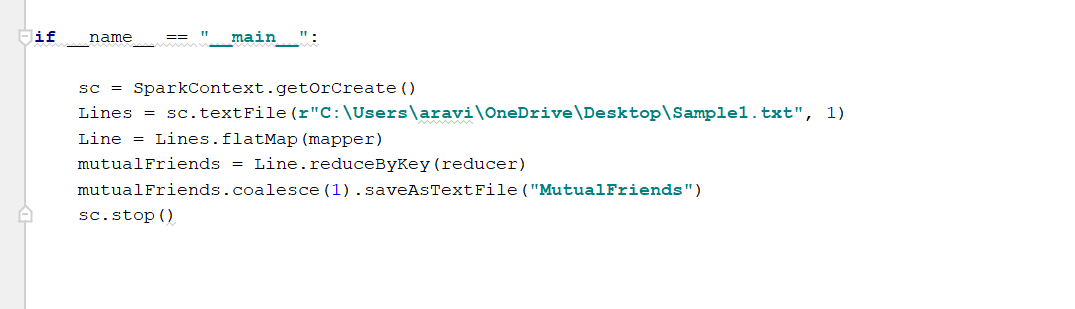


**Input 2:**



**Code:**



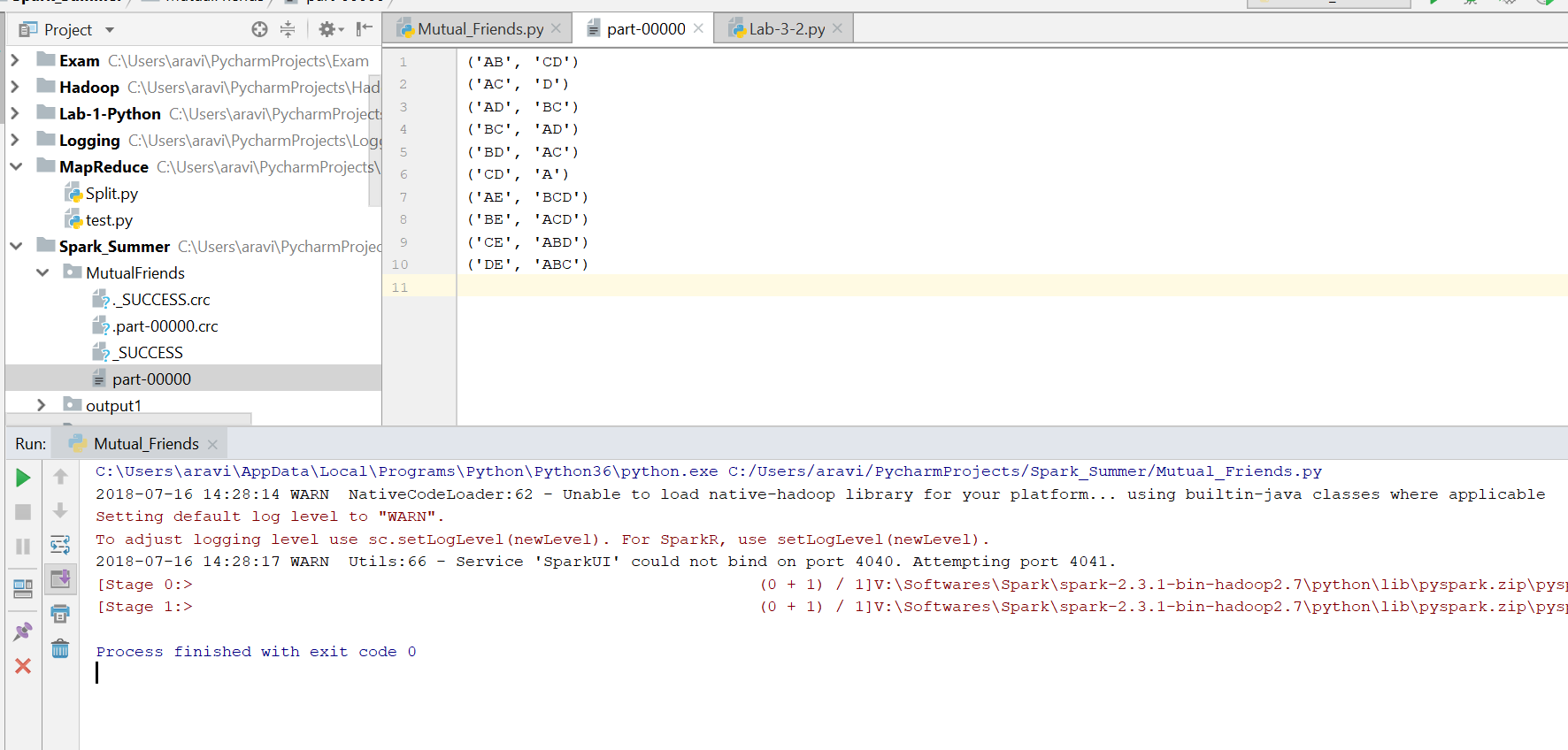


**Code Walk Through:**

* Input is given as text file and it consists of the user and the corresponding friends which are separated by ->.
* The line is split into two parts before and after ->. First part is the user and the second part is the friends for the user.
* The second part is then split into individual list of friends and then created another variable which consists of the friends without any separation characters between them.
* Iterating through the list of the friends and we will add the user and the friend as the key and the value is rest of the friends corresponding to the user.
* The list id then returned, and it will be used as the input to the mapper.
* The reducer will work on the key and the value that is passed from the mapper and for each friend in the key it is then checked in the value and if it is available it is added to the list.
* The final output consists of the data of the users and the mutual friends between them.

**Output Screenshots:**

**Output1:**



**Output2:**



**Conclusion:**

The output consists of the users and the mutual friends between them. The input is given in numbers and the letters.

1. **Creating Spark Data frames and RDD and store the data and perform the Queries on both the data and compare them.**

**Approach:** Chosen the data of FIFA World Cup and loaded the data using data frames and RDD and performed the SQL queries using spark. The outputs from the data frames and RDD are compared. Data is loaded, and the column names are declared through the header and the values are user to fetch the data.

**Inputs:**

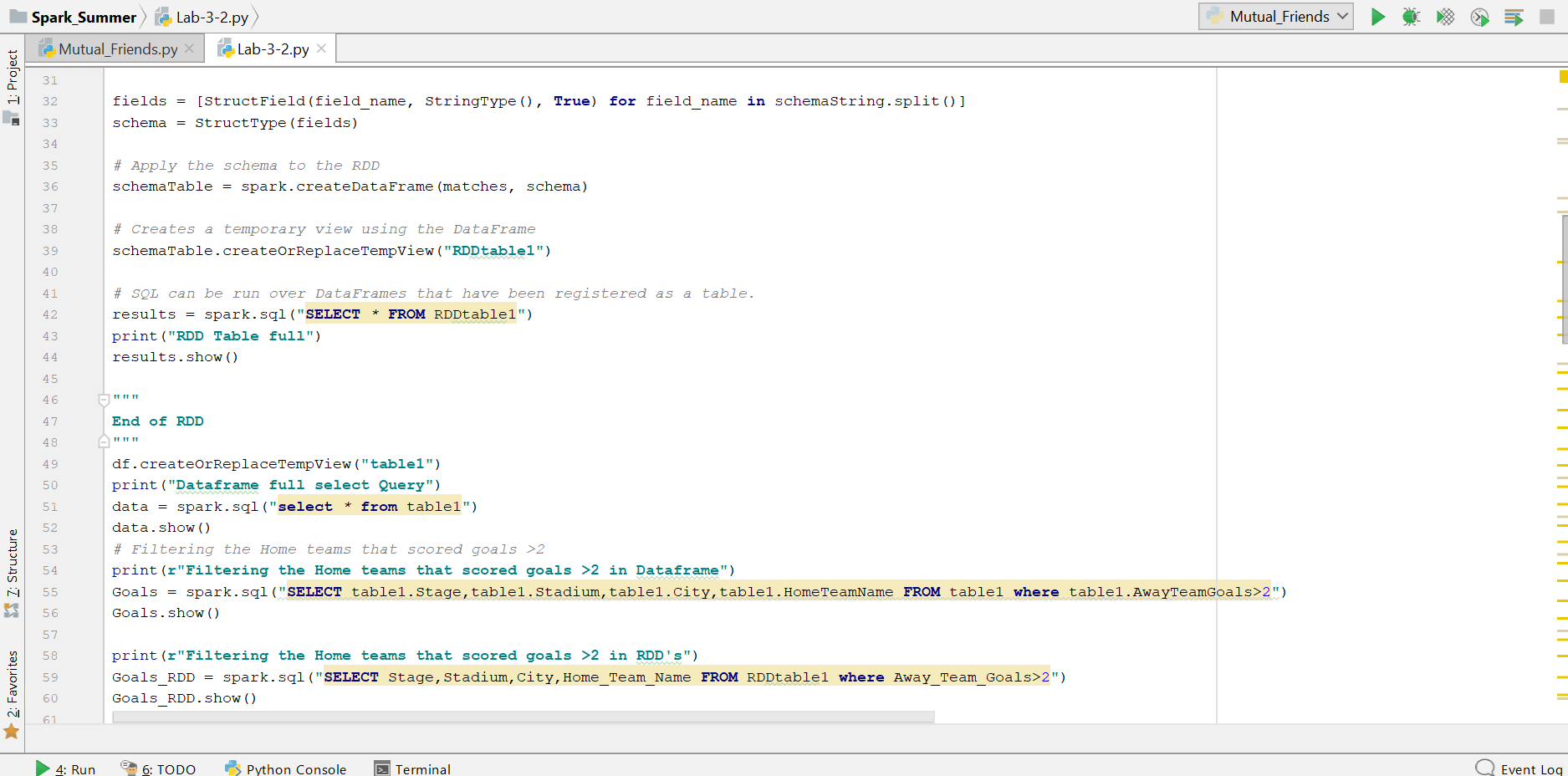
WorldCupMatches.csv – To load the data into data frames and perform the queries.

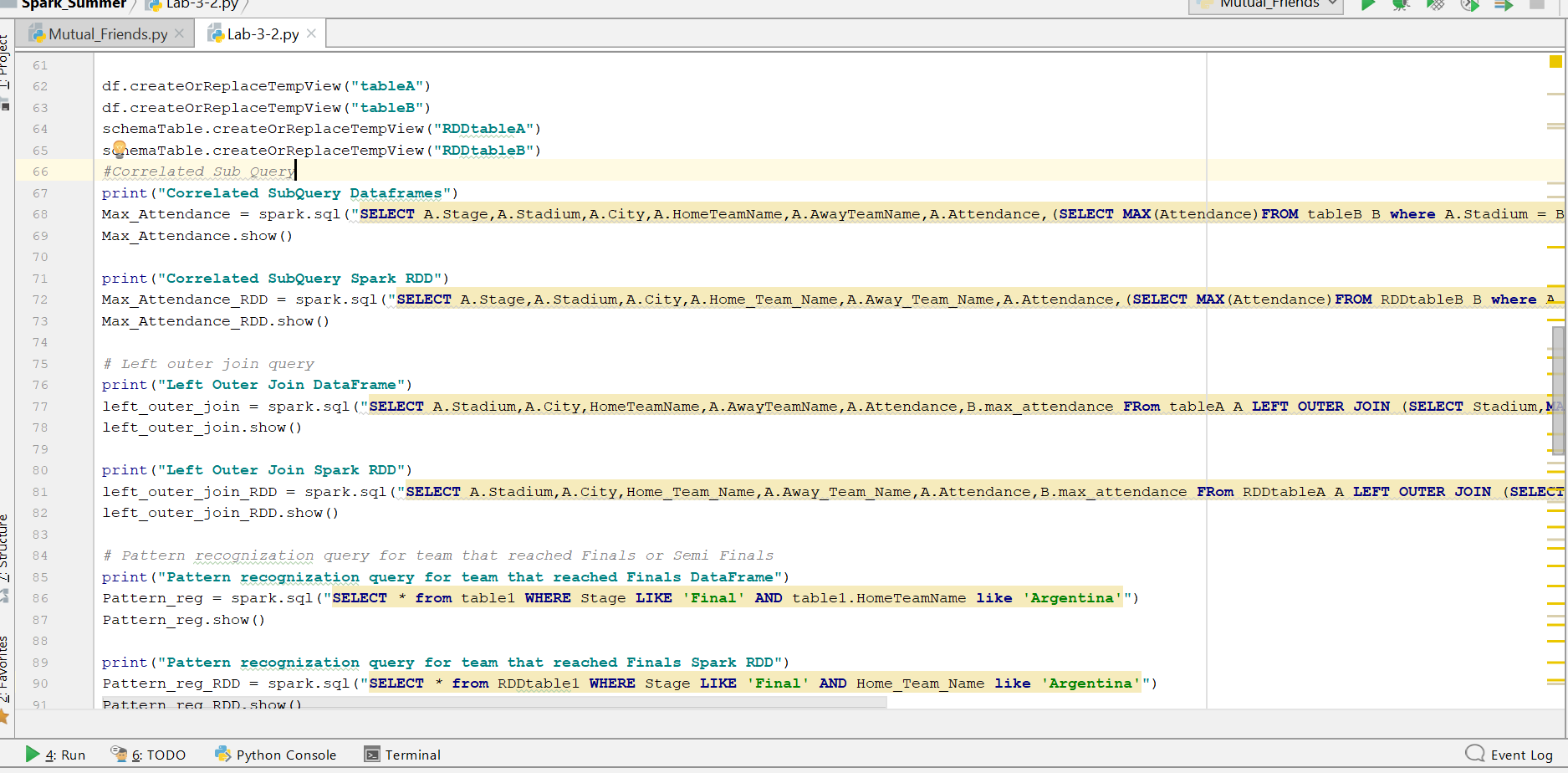
WorldCupMatches.txt - To load the data into Spark RDD and perform the queries.

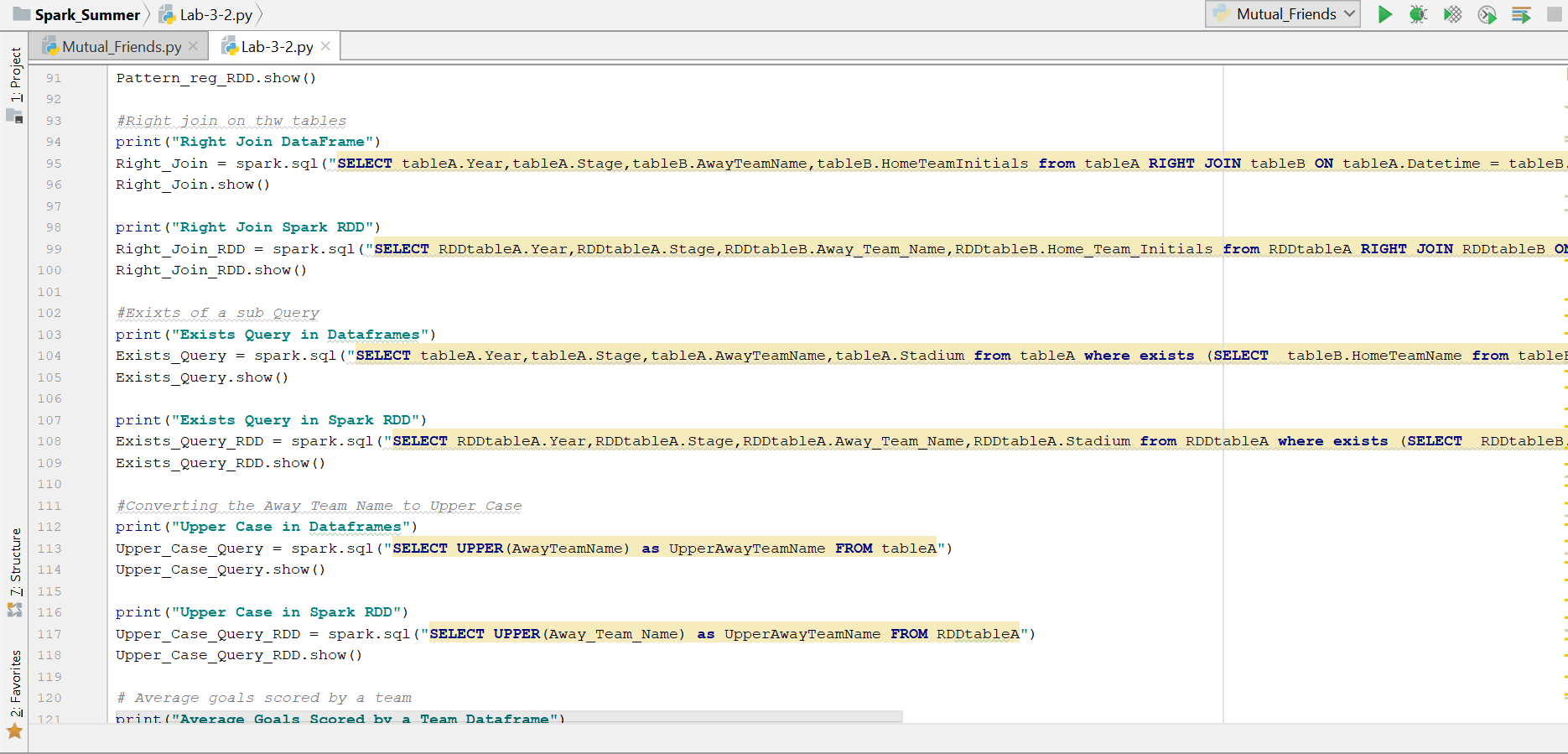
**Screenshots:**

**Code:**







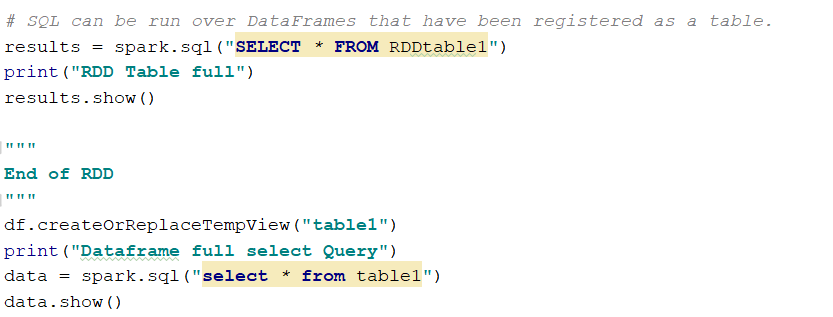


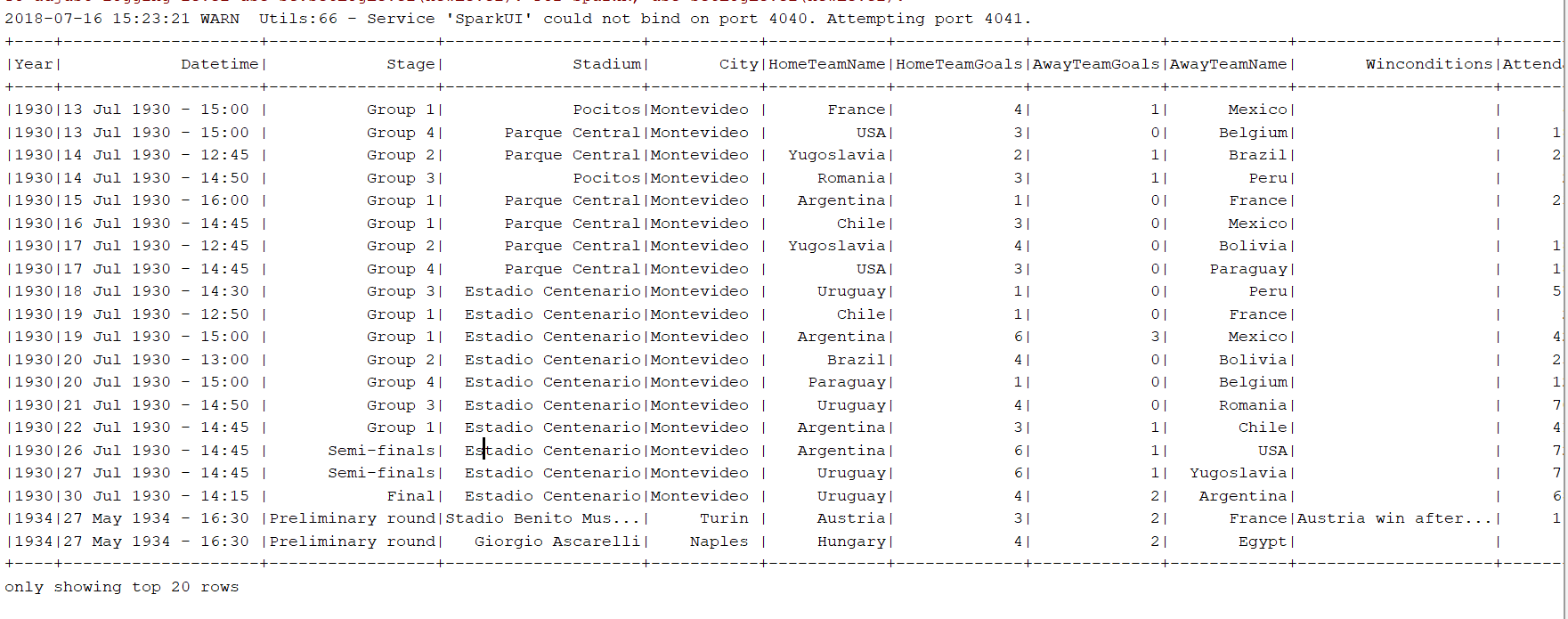


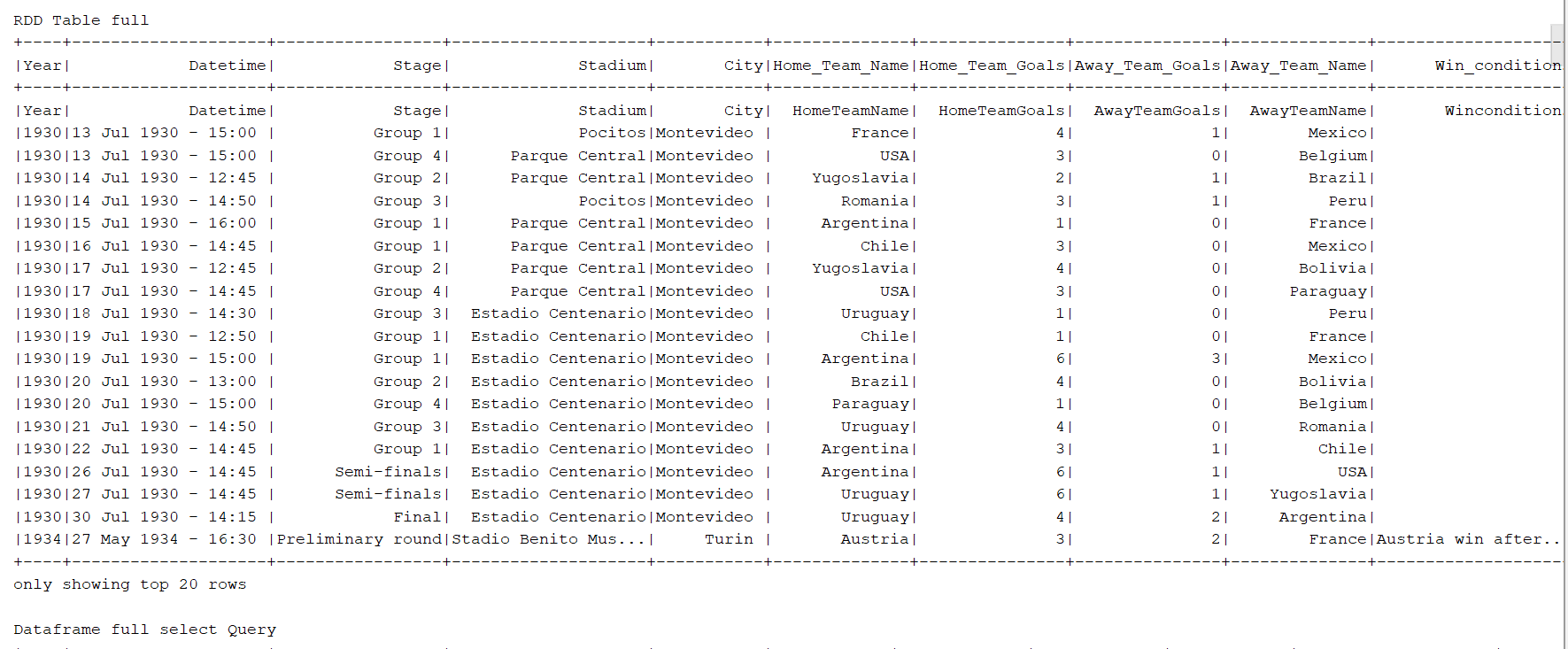
**Code Walkthrough:**

* CSV file is used to load the data into Spark Data Frames and Text file is used to load the data into Spark RDD.
* The schema for the data frames is created using the header of the CSV file and same are used as the column names and fetch the data.
* Schema is created for the Spark RDD as a string and split the string and assign the column names for the data.
* Queries are performed on both the data and compared.

**Load The data and Visualize the data loaded into the table for both Data Frames and RDD:**

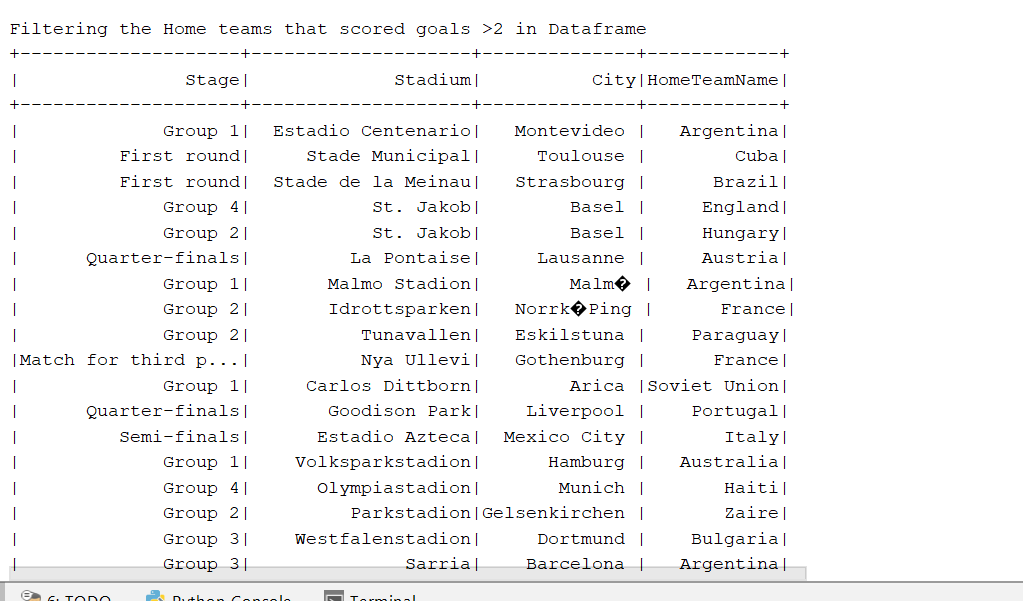






**Query – 1: Filtering the Home Teams that scored goals greater than 2.**

**Filtering in Data Frames:**

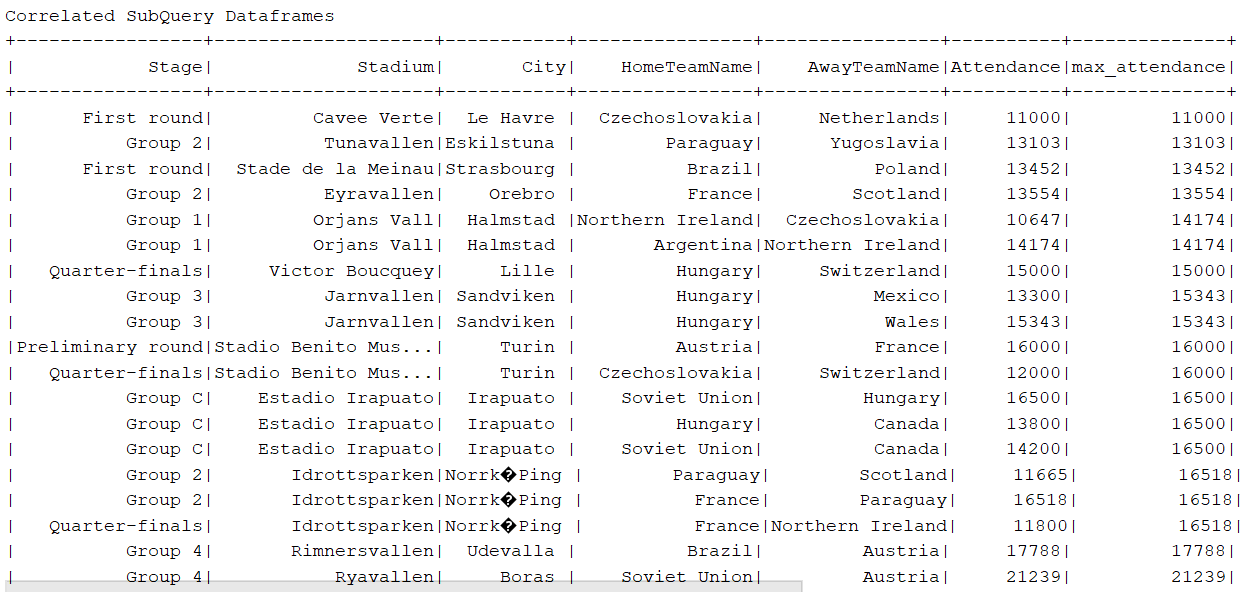


**Filtering in RDD:**

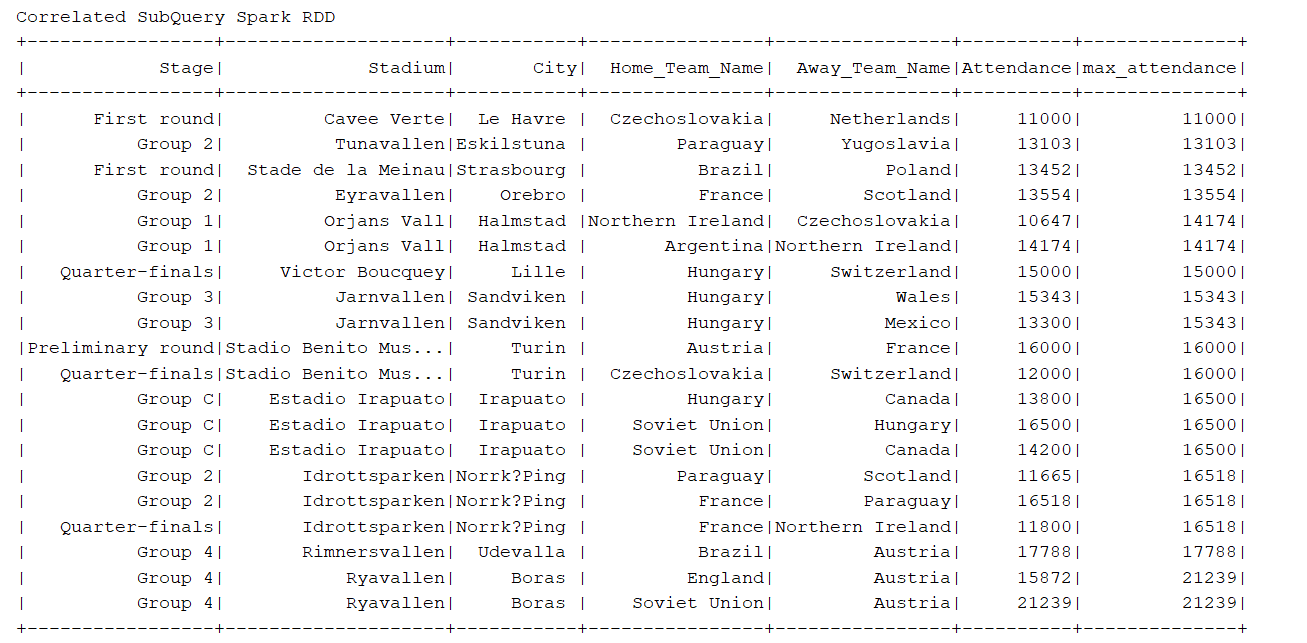


**Query - 2: Correlated Sub Query:**

**Correlated in Data Frames:**

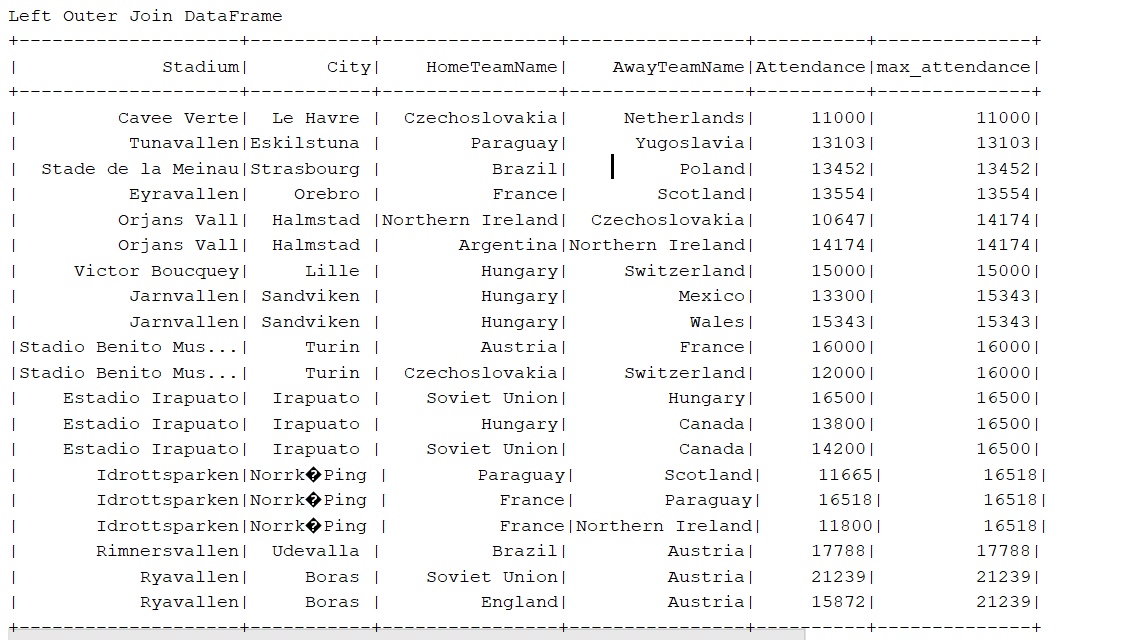


**Correlated in RDD:**

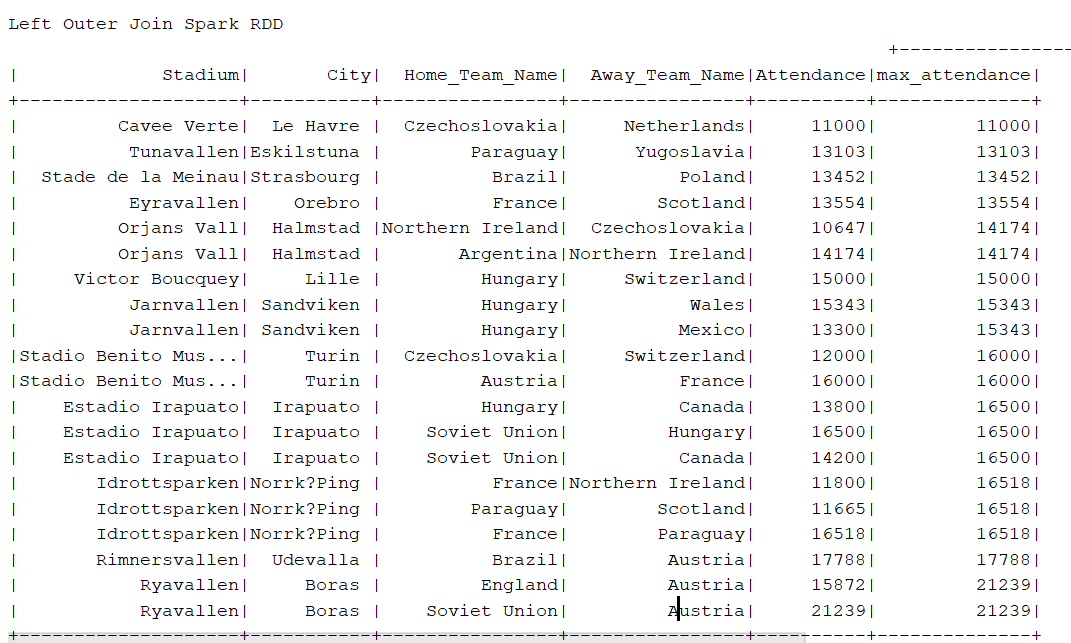


**Query – 3: Left Outer Join:**

**Left Outer Join in Data frames:**

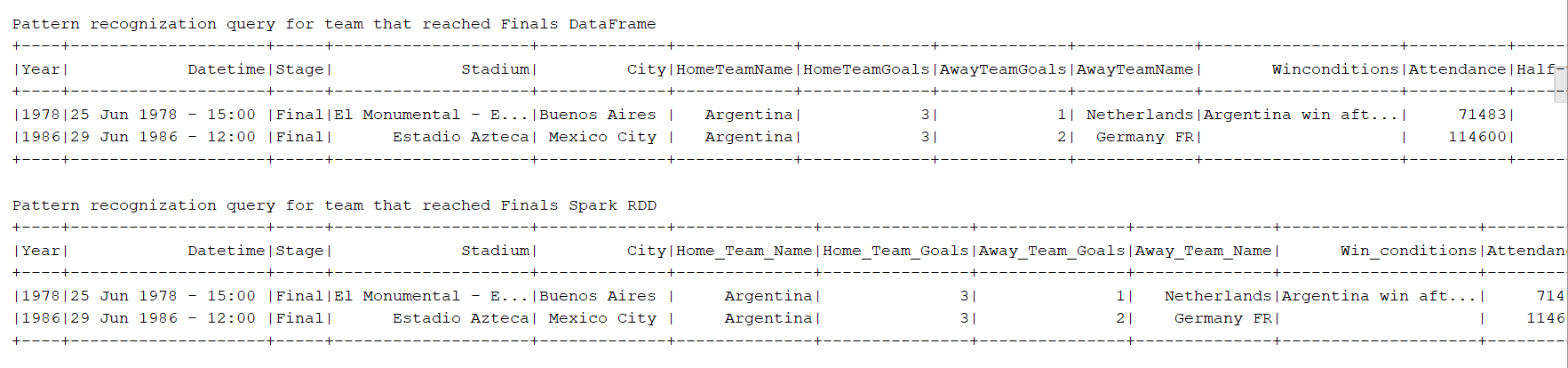


**Left Outer Join in RDD:**



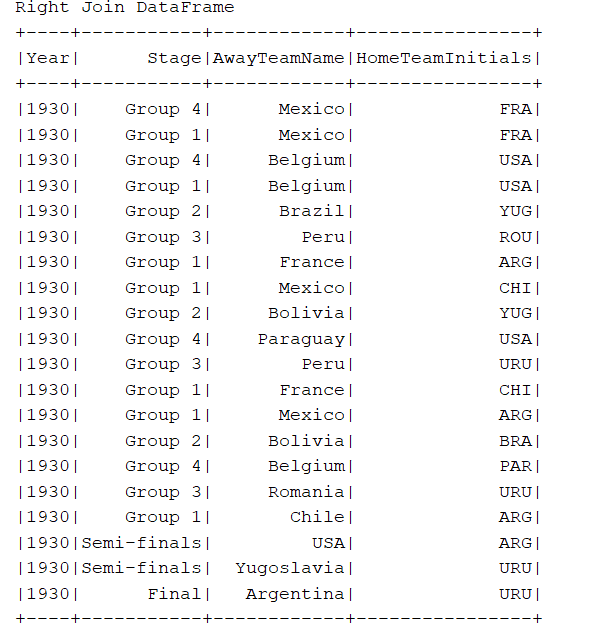
**Query -4: Pattern Recognition for the teams reached the stage final:**

**Pattern in both Data Frames and RDD:**

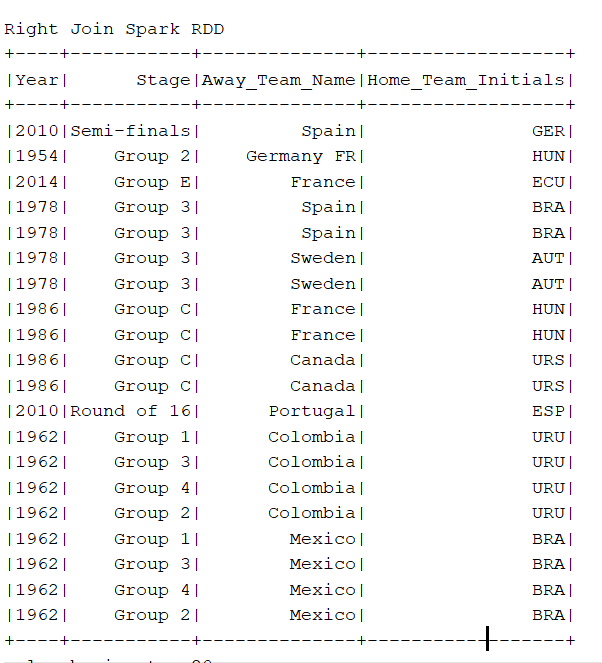


**Query – 5: Right Join on the tables.**

**Right Join on Data Frame:**

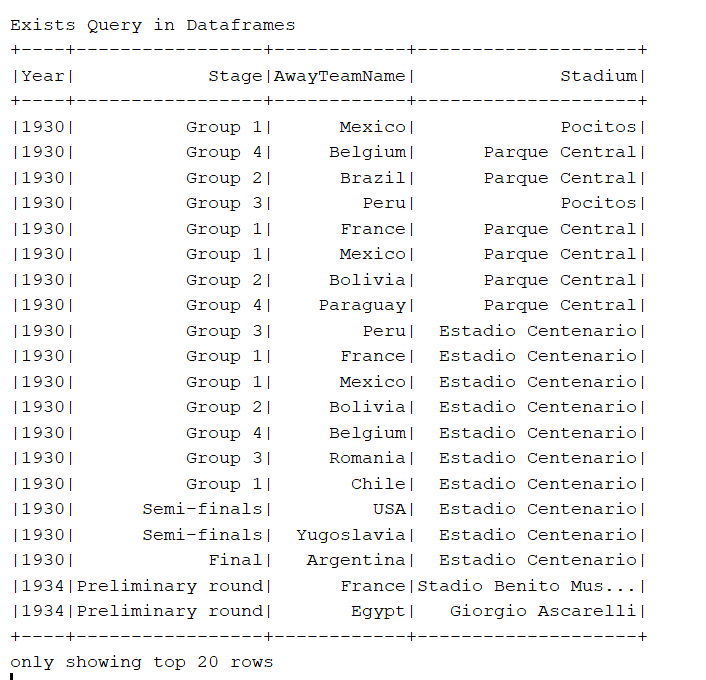


**Right Join on RDD:**

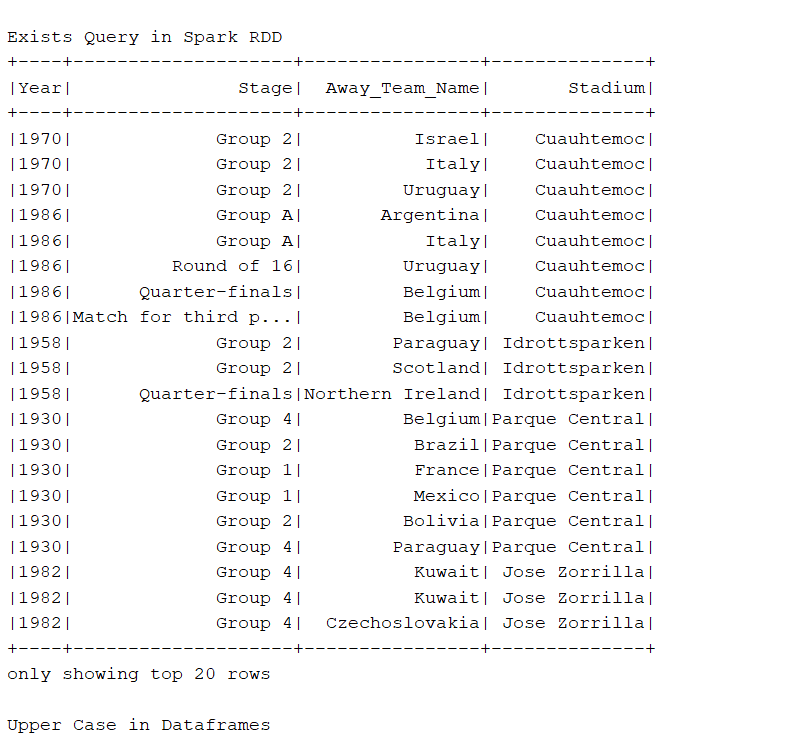


**Query -6: Exists of a Sub Query:**

**Exists in Data Frames:**

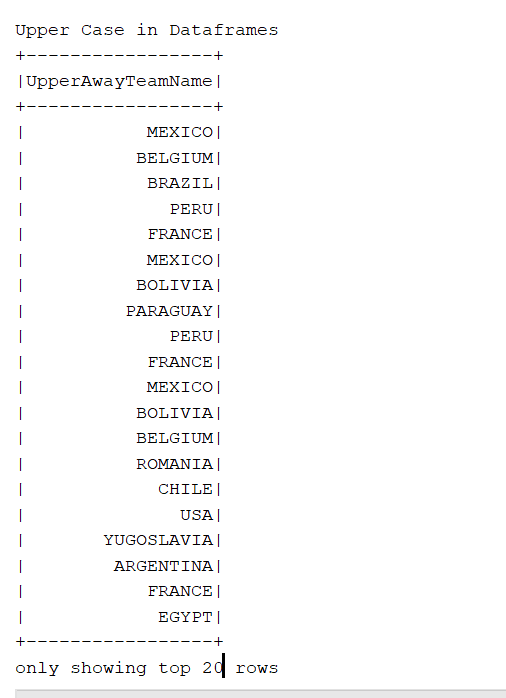


**Exists in RDD:**

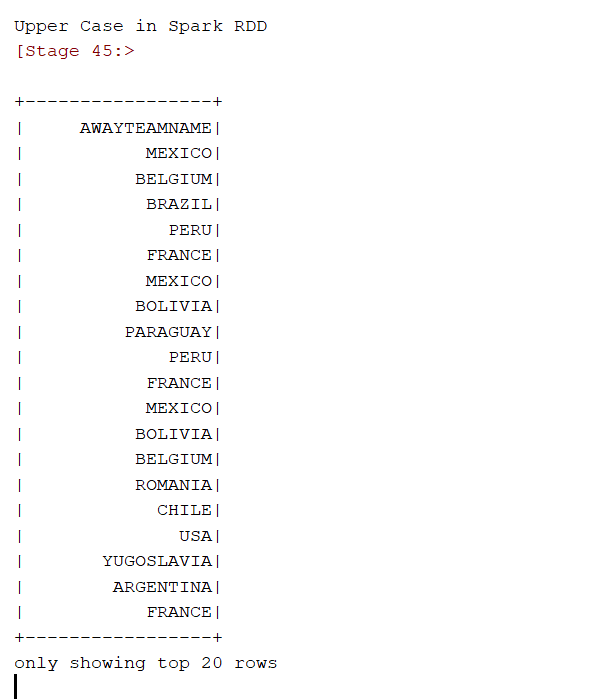


**Query -7: Upper Case of column Values:**

**Upper Case in Data Frames:**

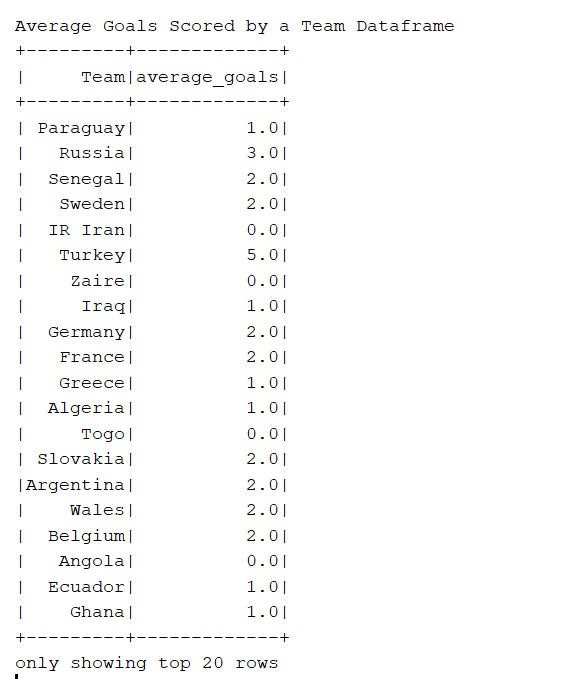


**Upper Case in RDD:**

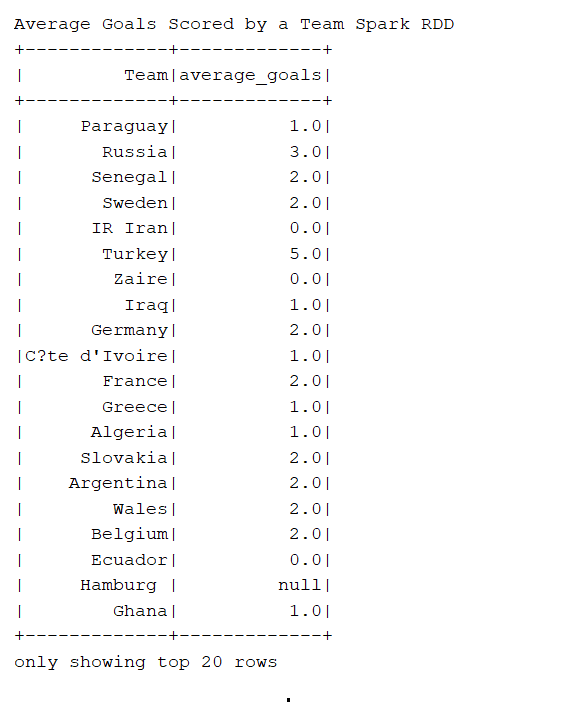


**Query-8: Average:**

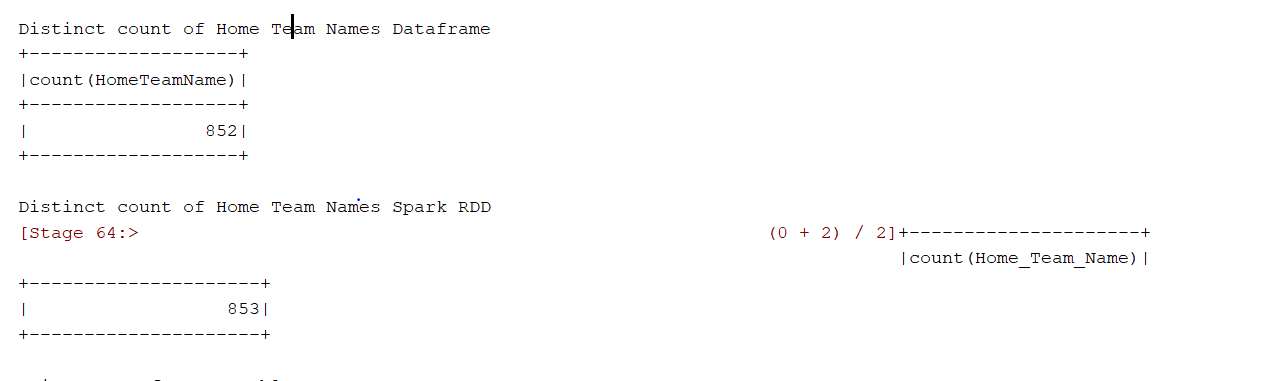
**Average – Data Frames**



**Average – RDD**

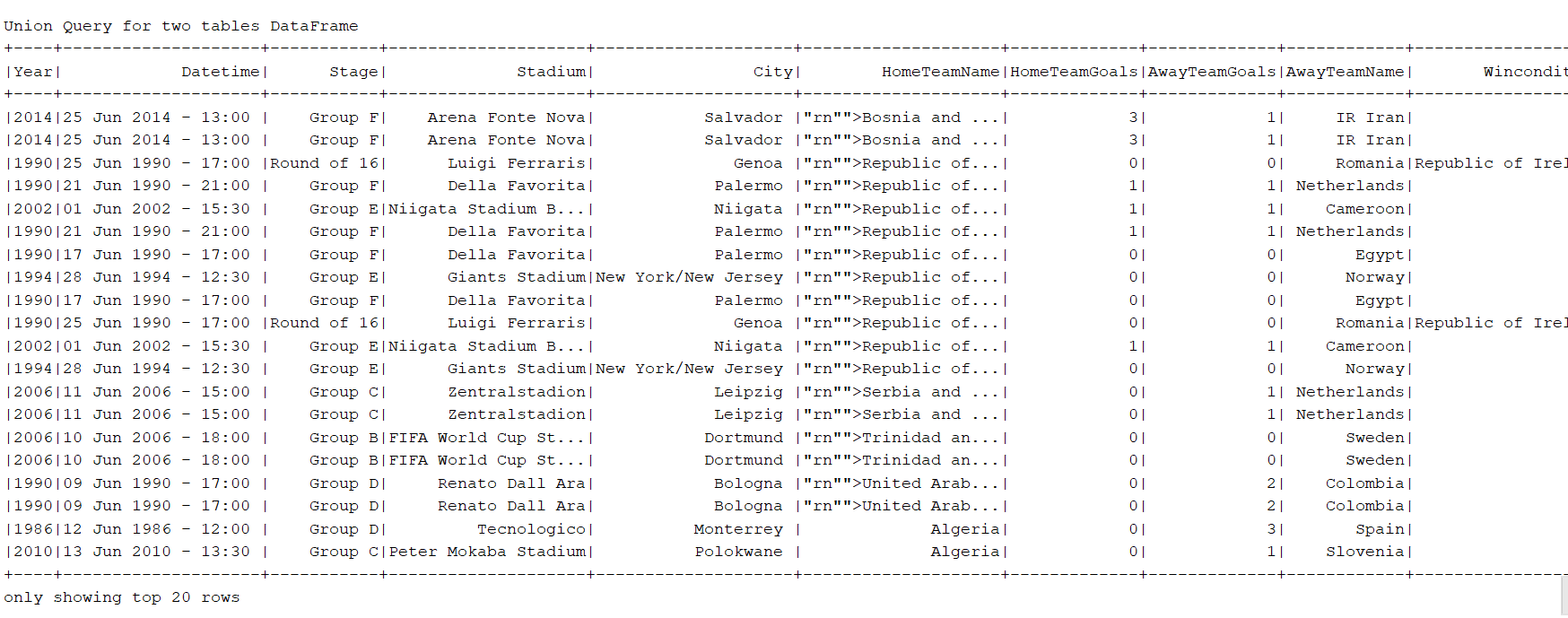


**Query – 9: Distinct Elements Count:**



**Query – 10: Union of two tables:**

**Union – Data Frames**



**Union RDD:**

