

INTRODUCTION TO BIG DATA

PROJECT REPORT - EUROPEAN SOCCER ANALYSIS - GROUP 10

SAHAJ GANDHI | ANIKET GIRIYALKAR | ADITYA KALYAN JAYANTI  
 CSCI 620 | 04/18/2018

Table of Contents

[INTRODUCTION 4](#_Toc511942113)

[Data Migration 5](#_Toc511942114)

[Transferring SQLite file to Server SQL 5](#_Toc511942115)

[Opening our dataset in R and Rattle 9](#_Toc511942116)

[SCHEMA DIAGRAM 14](#_Toc511942117)

[DATA CLEANING 15](#_Toc511942118)

[DATA EXPLORATION 17](#_Toc511942119)

[DATA TRANSFORMATION 19](#_Toc511942120)

[DATA VISUALIZATION 20](#_Toc511942121)

[PREDICTIVE ANALYSIS 38](#_Toc511942122)

[Top 20 Goalkeepers 38](#_Toc511942123)

[Top 10 Player statistics 39](#_Toc511942124)

[Top 20 strikers 40](#_Toc511942125)

[Top 20 defenders 44](#_Toc511942126)

[Top 10 and bottom 10 lifetime average ratings 47](#_Toc511942127)

[Team matches with league details and match results 49](#_Toc511942128)

[Retired footballers with ages - top and bottom 52](#_Toc511942129)

[Potential vs height 54](#_Toc511942130)

[Player lifetime rating average 55](#_Toc511942131)

[Player in terms of playing position predicted 56](#_Toc511942132)

[Height and header accuracy correlation 60](#_Toc511942133)

[Goal summary by season 61](#_Toc511942134)

[Fitness 62](#_Toc511942135)

[Currently playing footballers 63](#_Toc511942136)

[Goals and percentage by foot 65](#_Toc511942137)

[DATA MODELS 66](#_Toc511942138)

[Look alike model 66](#_Toc511942139)

[Percentile overall ratings 70](#_Toc511942140)

[Percentile strikers 72](#_Toc511942141)

[Percentile goalkeeper 73](#_Toc511942142)

[Percentile defenders 74](#_Toc511942143)

[Naive Bayes 76](#_Toc511942144)

[CONCLUSION 81](#_Toc511942145)

[REFERENCES 81](#_Toc511942146)

# INTRODUCTION

Soccer is a global sport. It is played in almost every country in the world, and soccer competitions have global viewing numbers that blow all other sports (and even the Olympics) out of the water. The 2014 FIFA World Cup Final has been by far the most viewed sports event in recent times, with over 700 million viewers. It makes sense that its top professional leagues are different and superior in structure than those of any other sport in the world.

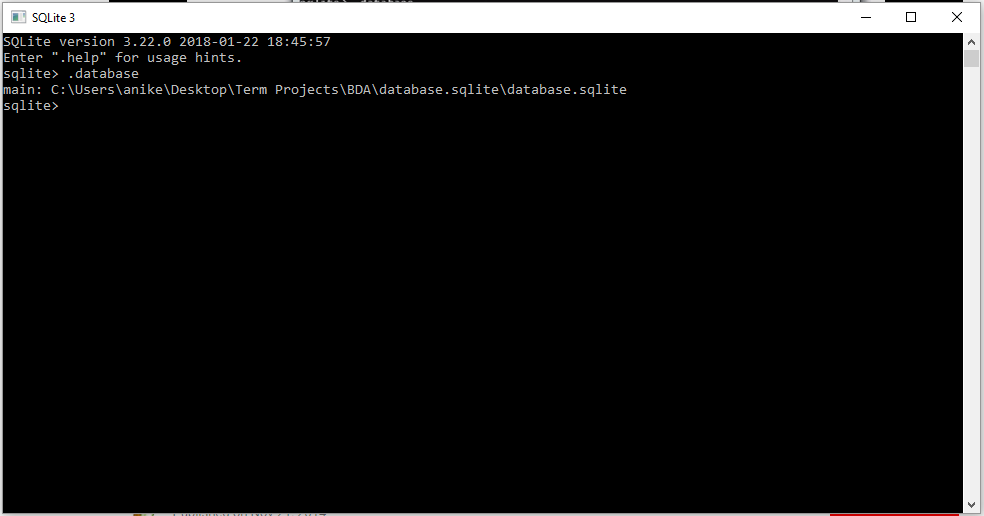
European Soccer is more competitive than any other sports leagues because of the format of the leagues, the multiple competitions, and the intense rivalry between not only each team, but in each league.

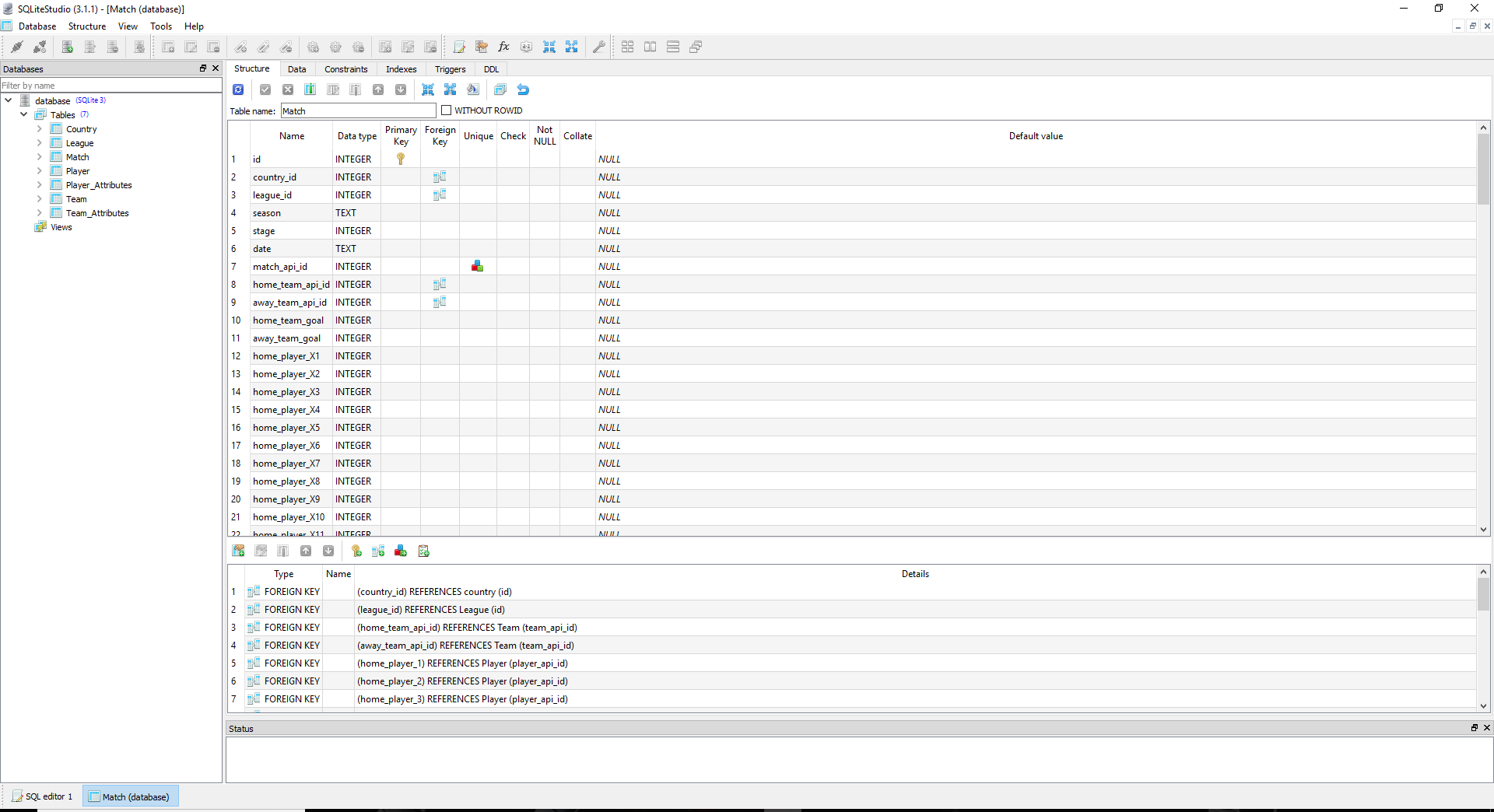
To understand what makes European Soccer the most competitive league in the world, we have chosen to examine this database, which contains information about over 2500 matches, about 10000 players representing over 11 European Countries playing in the top tiers of their championships.

In this project we attempt to understand the factors of the teams and the players who play in Europe and analyze the player\_attributes and team\_attributes and generate insights from the data.

# Data Migration

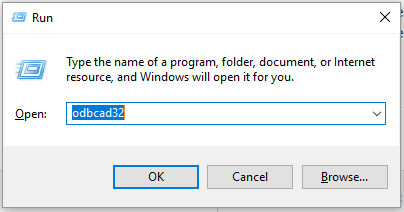
## Transferring SQLite file to Server SQL

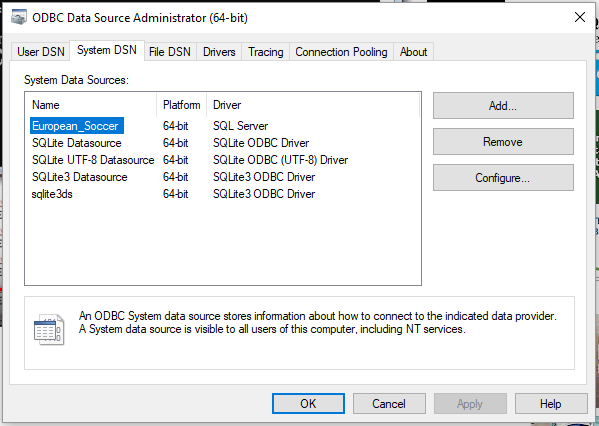


Firstly, we opened the dataset in Sqlite command line.

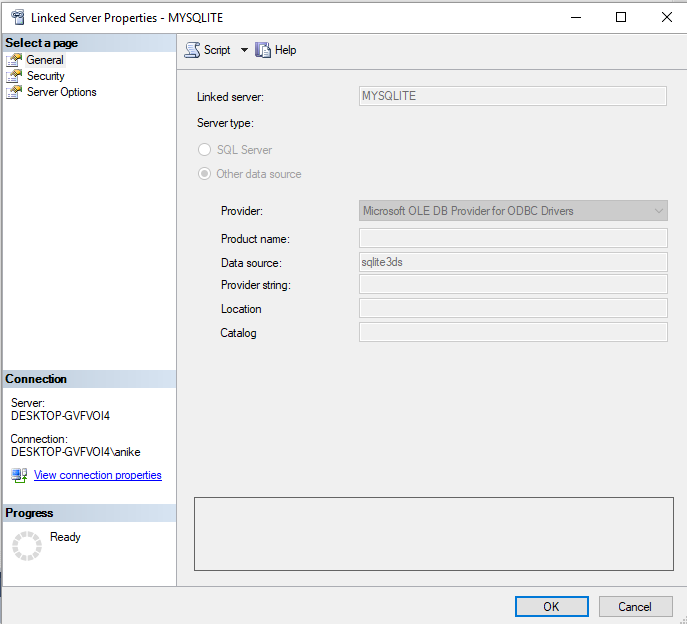
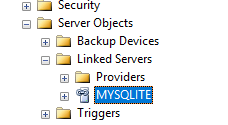
Then, we observed our dataset entities and all the table structures in the SQLite GUI.

After this step we created the DSN for connecting the SQLite server to SSMS server. For this we type odbacd32 in the run command and create a DSN.





Now, in SSMS we create a new linked server and hence connect it with the DSN we have created in the previous step.



The data source has the same name as the name of the DSN we created for SQLite.

Now to get our data from the dataset, we create tables in SSMS having the same structure as the source Dataset and insert data from the tables of SQLite into the tables that we have created in SSMS.  



The above queries explain how the data is selected from tables of SQLite and inserted into SSMS

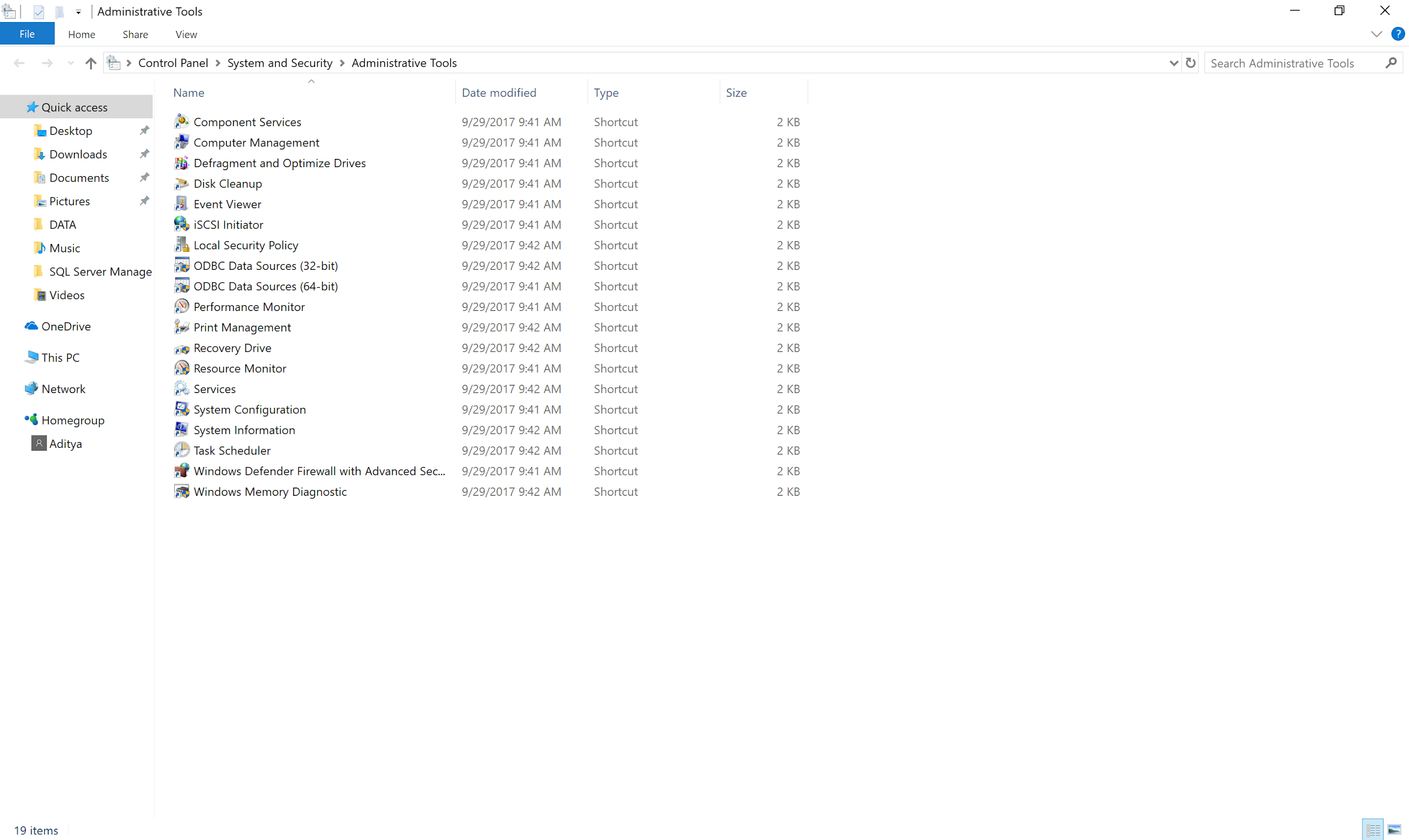
## Opening our dataset in R and Rattle

ODBC aims to provide common API access to SQL based data management systems.

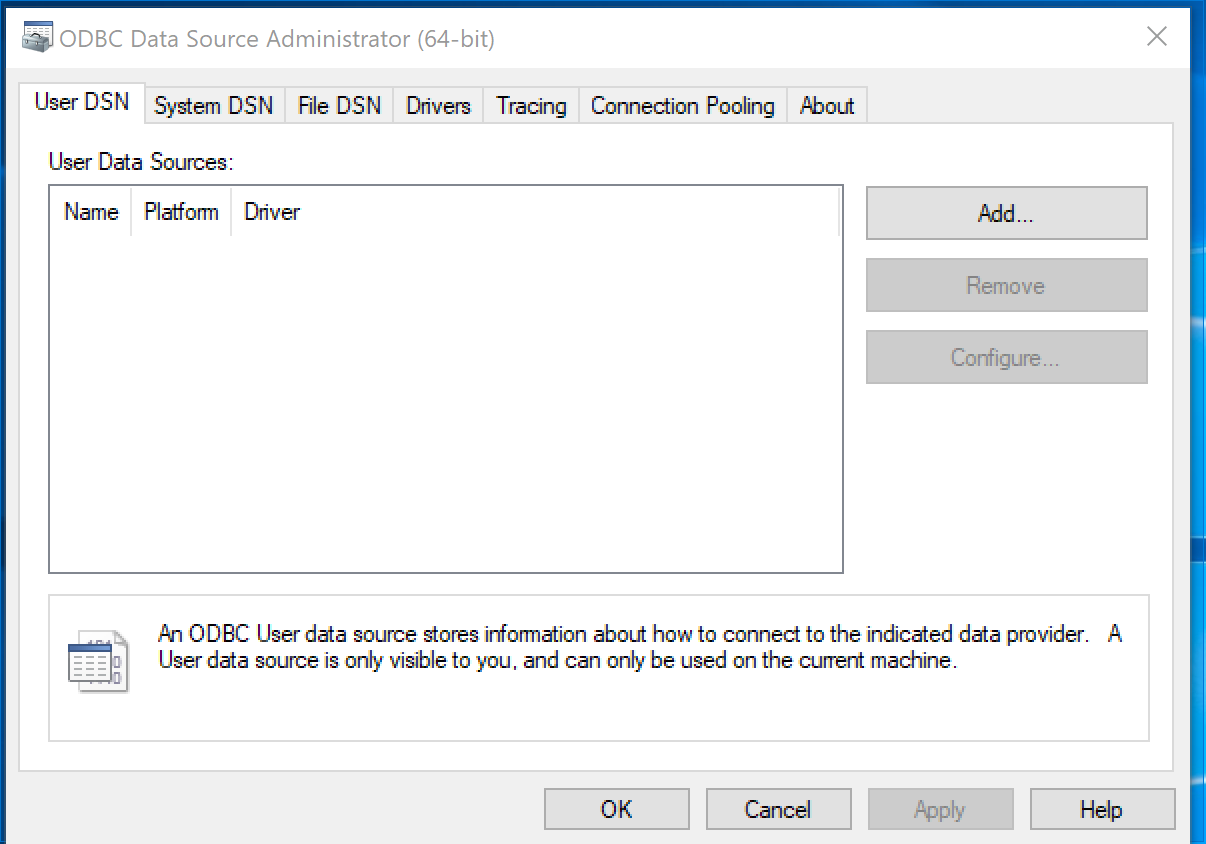
A connection to a specific database is called a Data Source Name or DSN.

To make an ODBC connection:

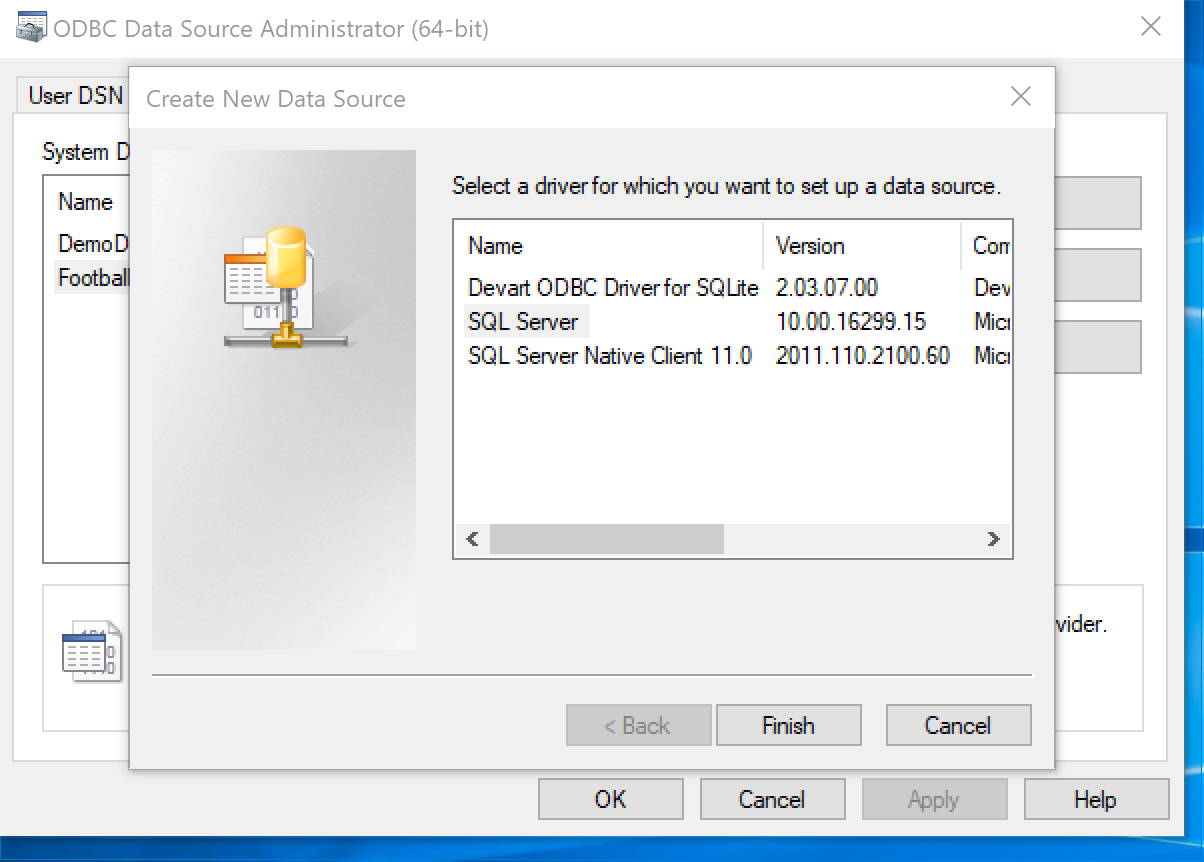
* *library(RODBC)*
* *Channel odbcConnect(“FootballDSN”)*

****

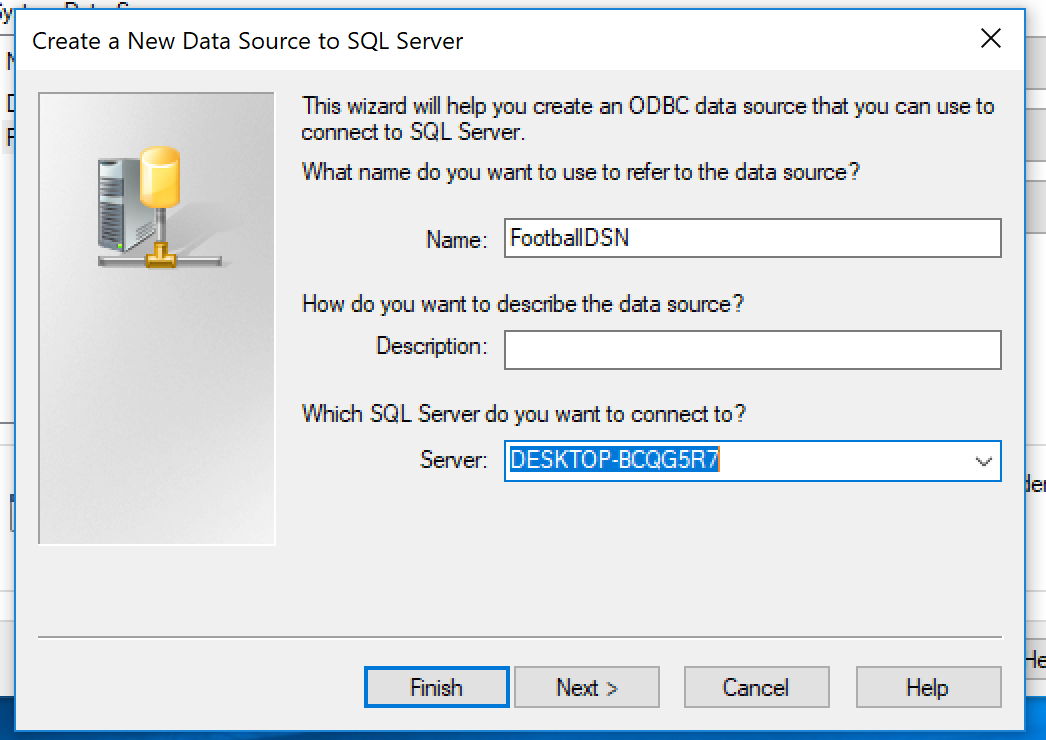
Select, ODBC data sources(64-bit) under administrative tools.

****

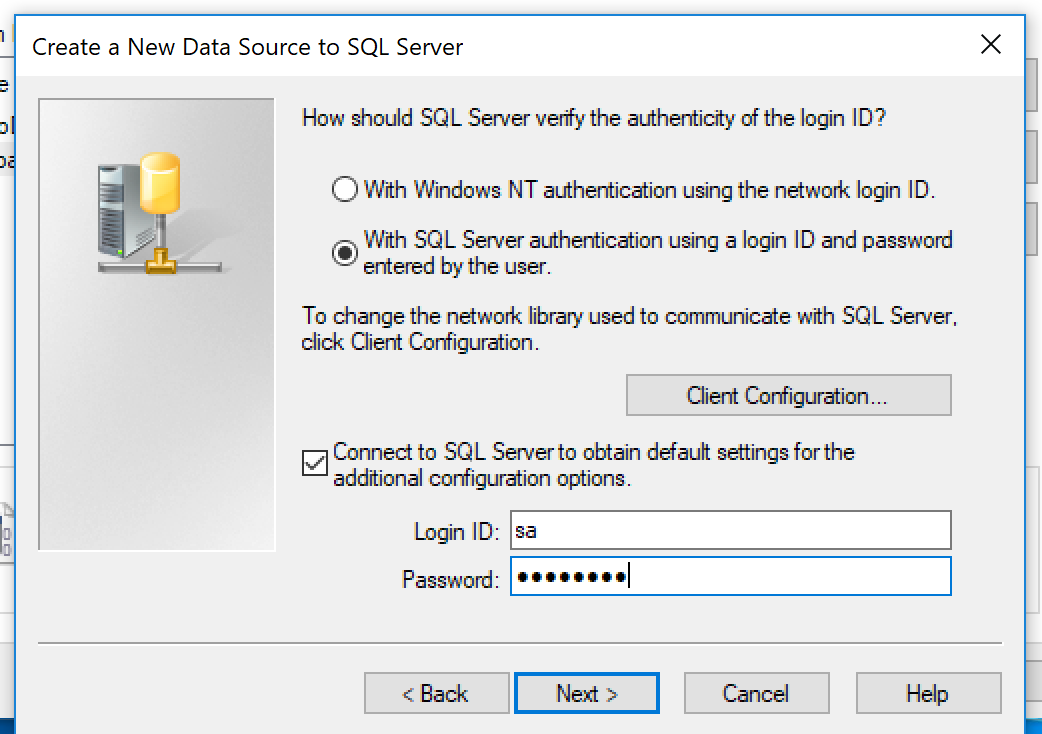
Adding a new User DSN in ODBC data sources.

****

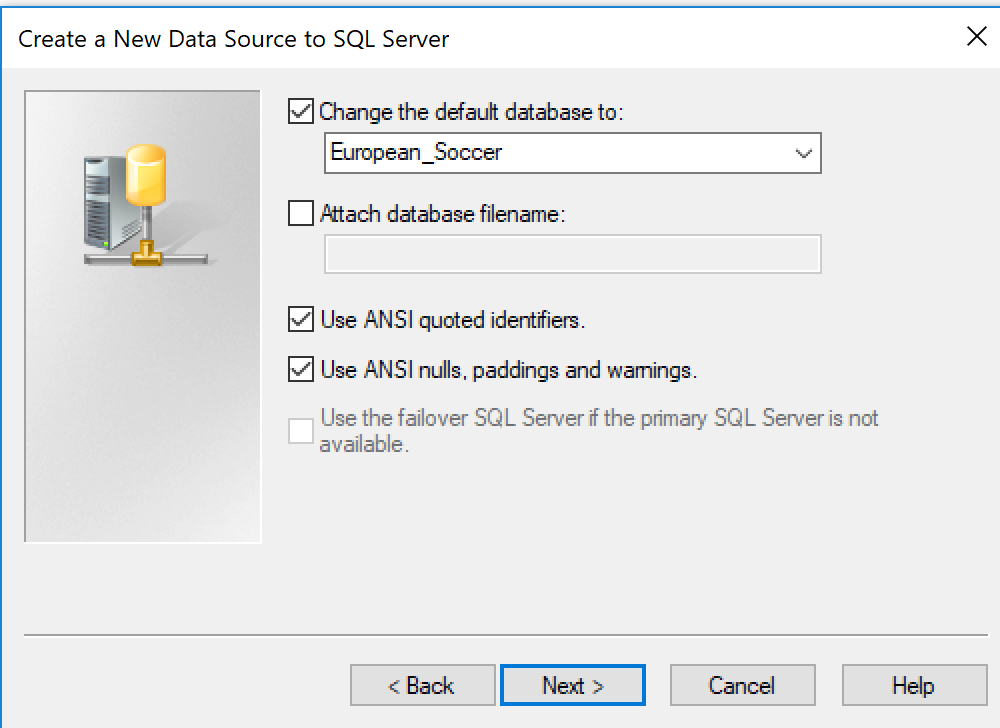
SQL Server driver is selected to set up the data source.

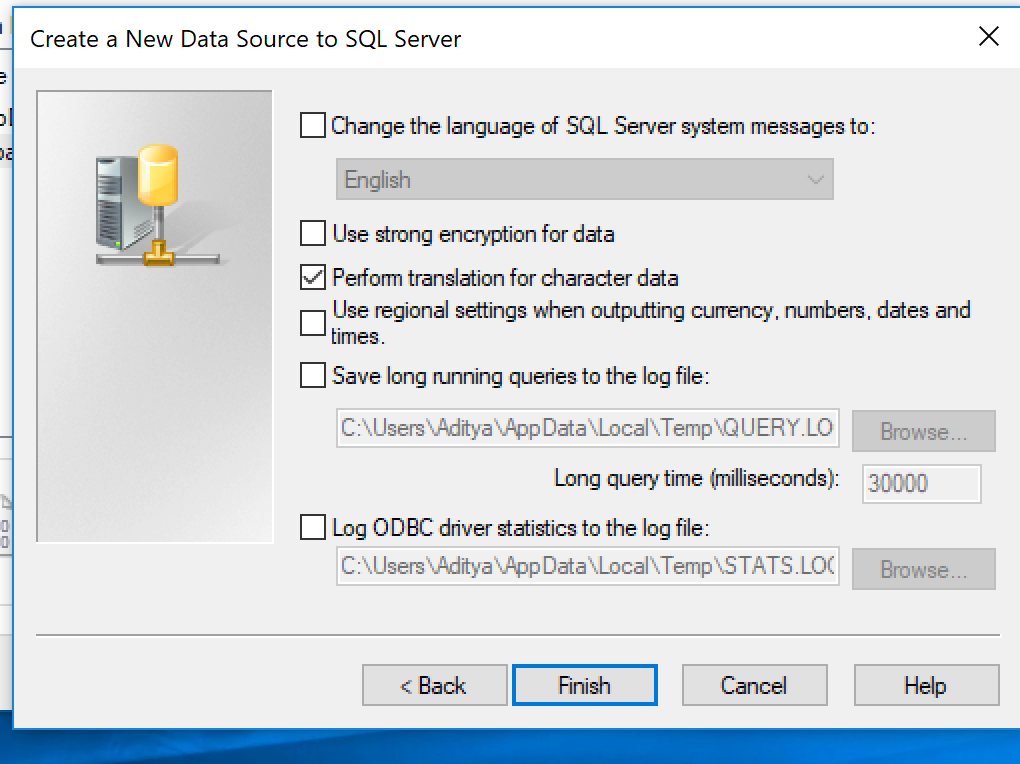
****

“FootballDSN” is a name given to the data source. The default SQL Server is selected.

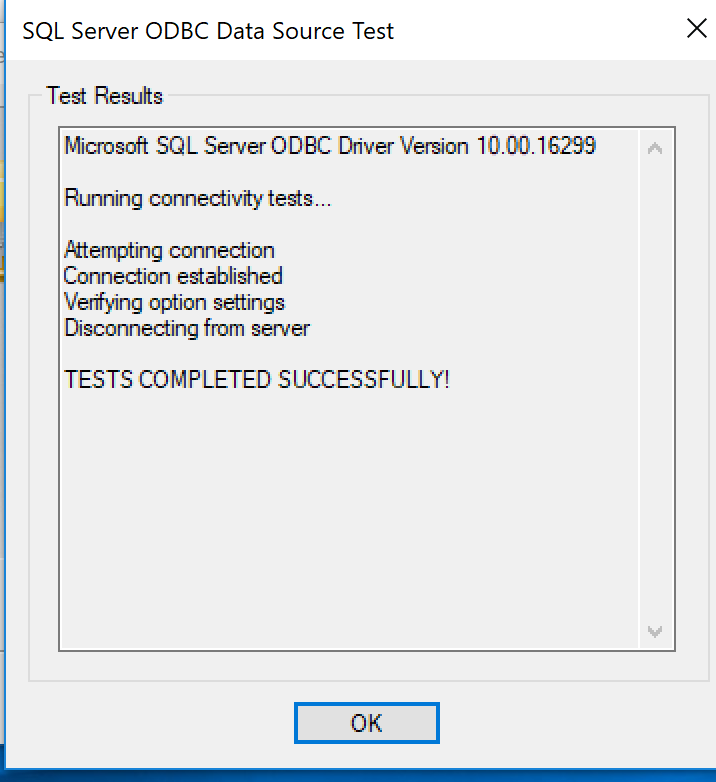
****

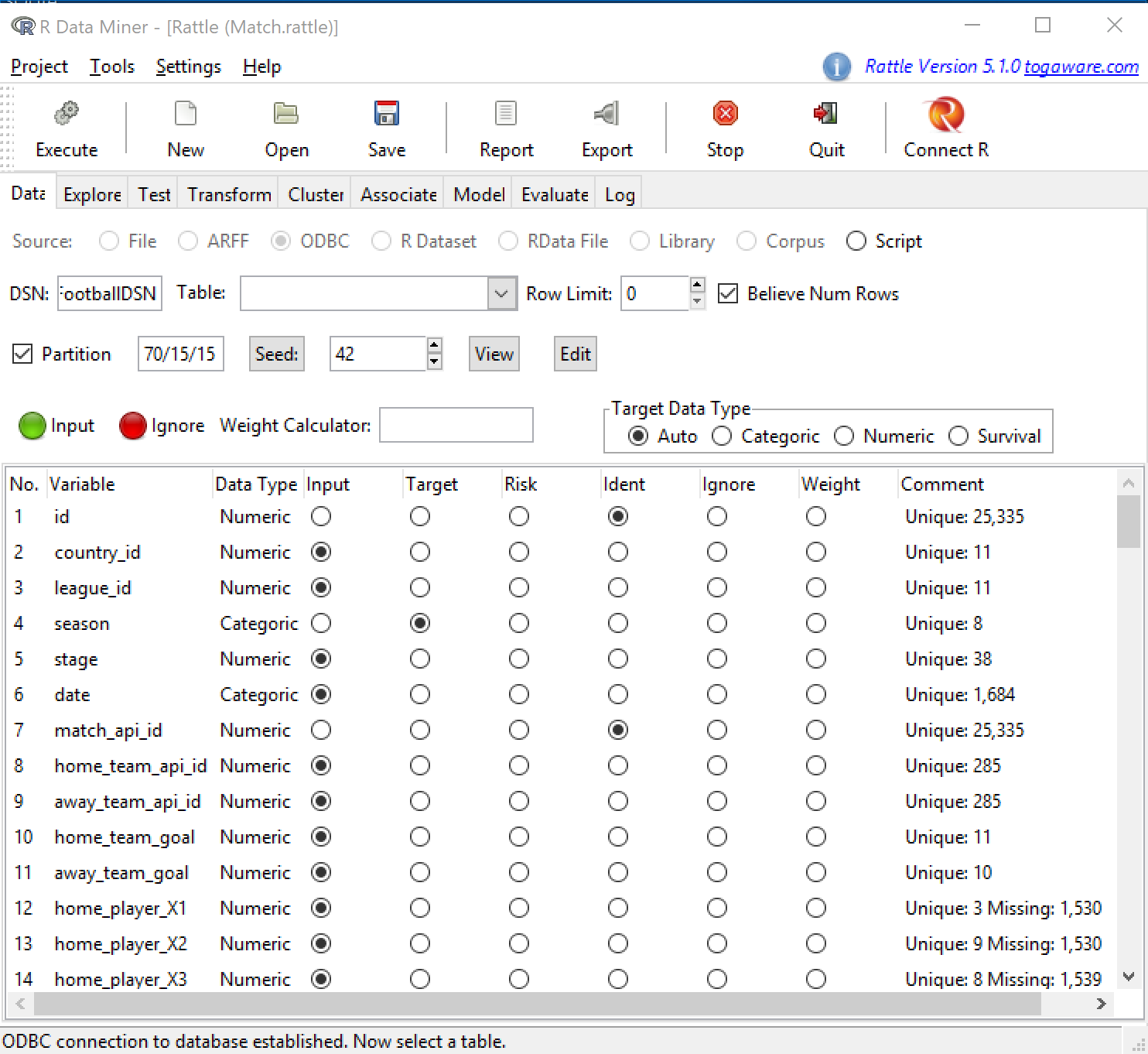
SQL Server is authenticated with log in ID and password. The default settings are used to obtain additional configuration options. With log in ID as “sa”. This authentication allows the user to access the current databases present in SQL Server Management System (SSMS).

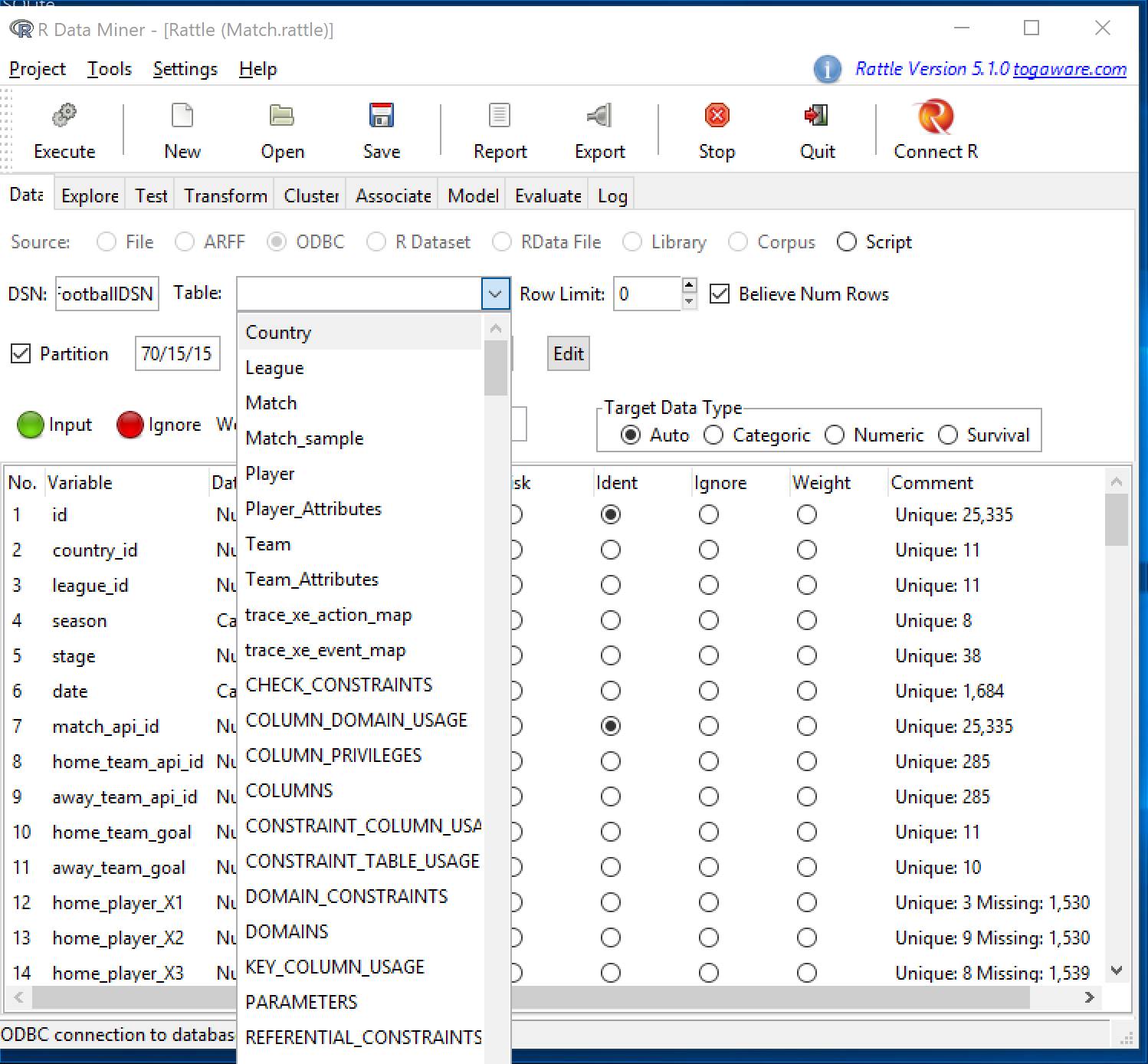
****The default database is selected from the list of databases available in SSMS.

****

A new data source is created for the default database. Select “Finish” to test the data source.

****

****

****

We select the table we want from the dropdown.

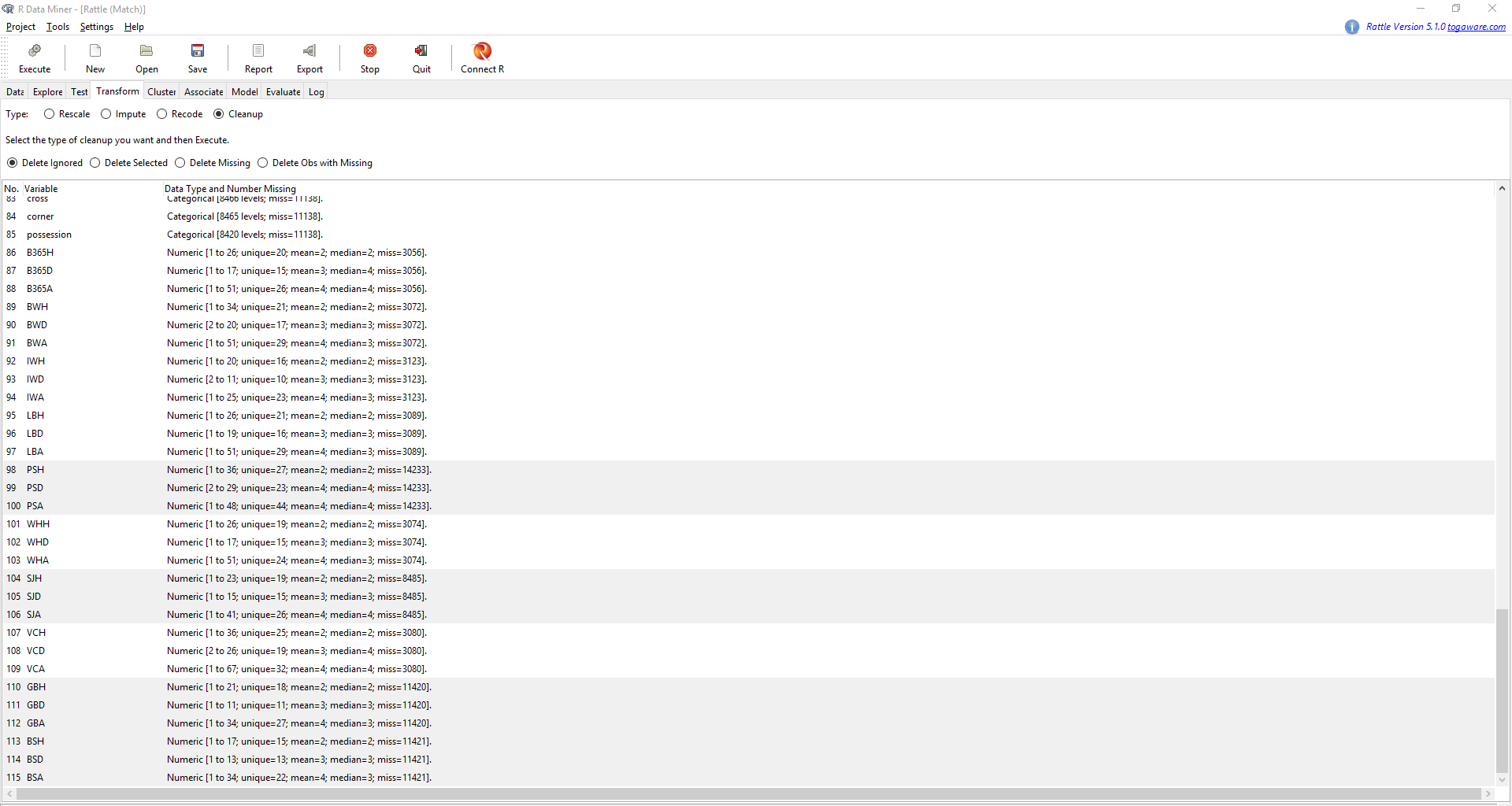
# SCHEMA DIAGRAM



# DATA CLEANING

After we have opened the dataset on R and Rattle we browse through each table to check if it needs to be cleaned. On careful observations we find that the tables Player, Team, League, Country, Player\_Attributes, Team\_Attributes are well organized and have very less missing values. But in the table match we observe that almost more than 30% of the data is missing in the following attributes:

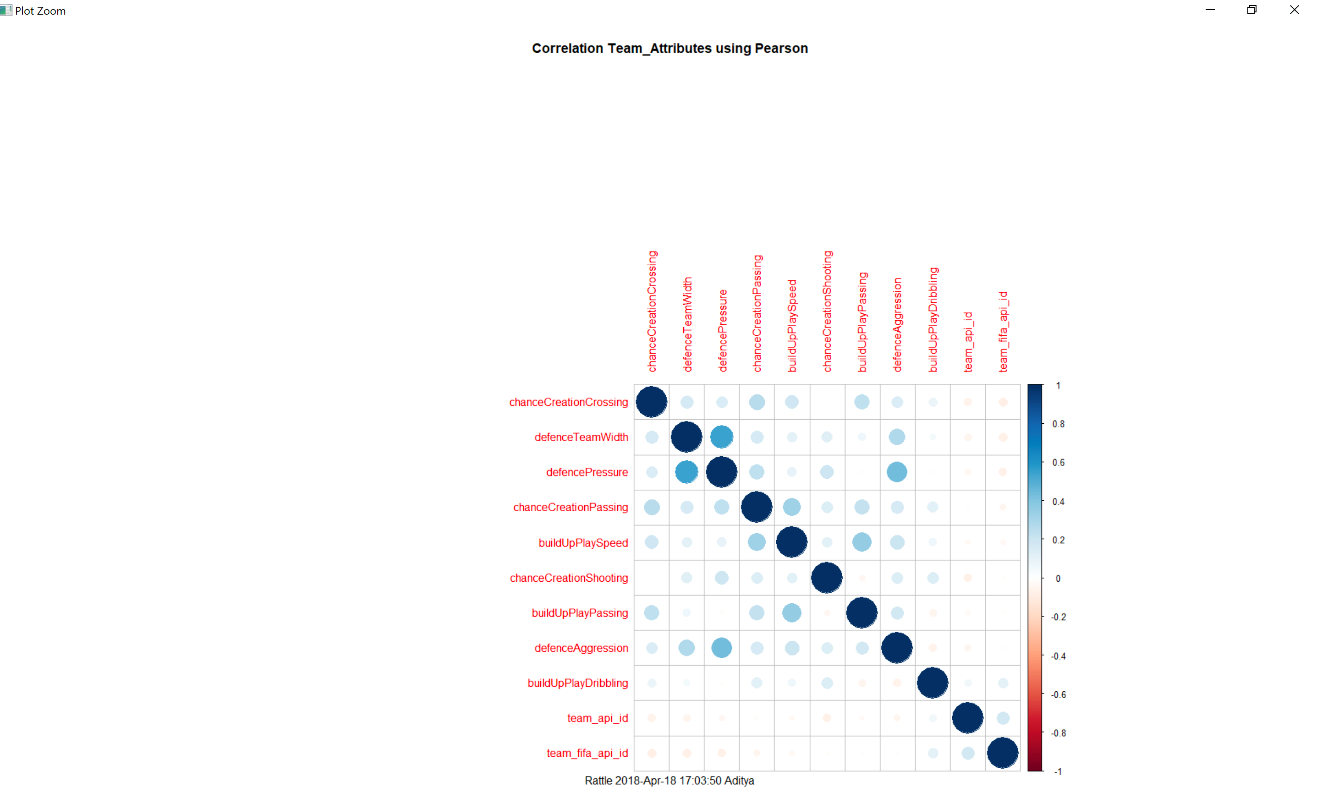
* PSH and PH - Pinnacle home win odds
* PSD and PD - Pinnacle draw odds
* PSA and PA - Pinnacle away win odds
* SJH - Stan James home win odds
* SJD - Stan James draw odds
* SJA - Stan James away win odds
* GBH - Gamebookers home win odds
* GBD - Gamebookers draw odds
* GBA - Gamebookers away win odds
* BSH - Blue Square home win odds
* BSD - Blue Square draw odds
* BSA - Blue Square away win odds
* GOAL
* SHOTON
* SHOTOFF
* FOULCOMMIT
* CARD
* CROSS
* CORNER
* POSSESSION

Since 30% is huge amount of data compared to the total number of observations, we can’t even replace the missing values with either mean, median, or any custom value and hence delete these columns.

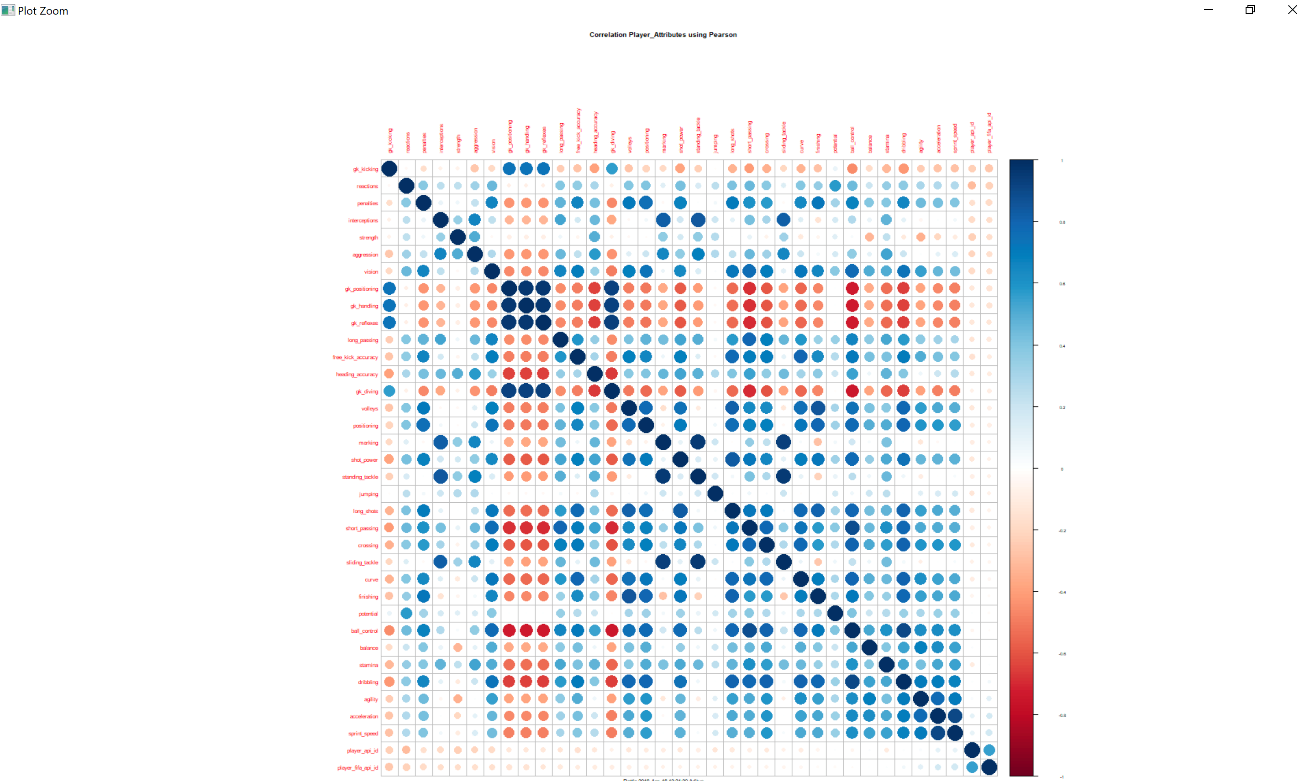
As we see in the figure above that the missing values are far too much and hence it justifies the deleting of the above-mentioned attributes.

# DATA EXPLORATION

In data exploration, we explore all our tables to see if we can find any correlation between the attributes. Using the below two plots we observe the correlations between the numeric  
attributes of the tables Team\_Attributes and Player\_Attributes.



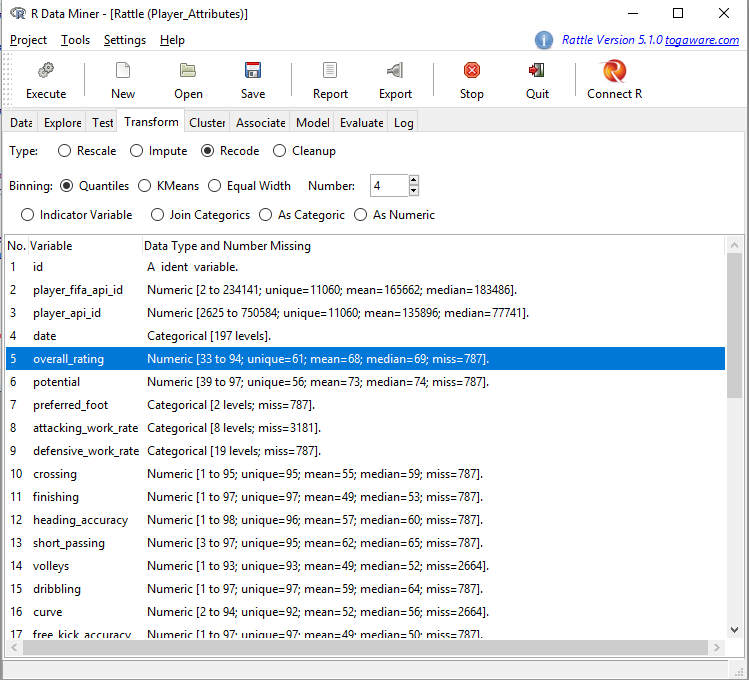
The blue circles denote the positive correlations between the attributes, and lighter the color less the correlation. Red circles indicate the negative correlation. We observe the attributes showing positive correlations more closely and visualize these to obtain further insights on these attributes and what they represent.



We see that in tables Team, League, Country, and Player we don’t have any interesting correlations and hence we do not study the attributes of these tables as it doesn’t make any sense to read for the relations that do not exist.

# DATA TRANSFORMATION

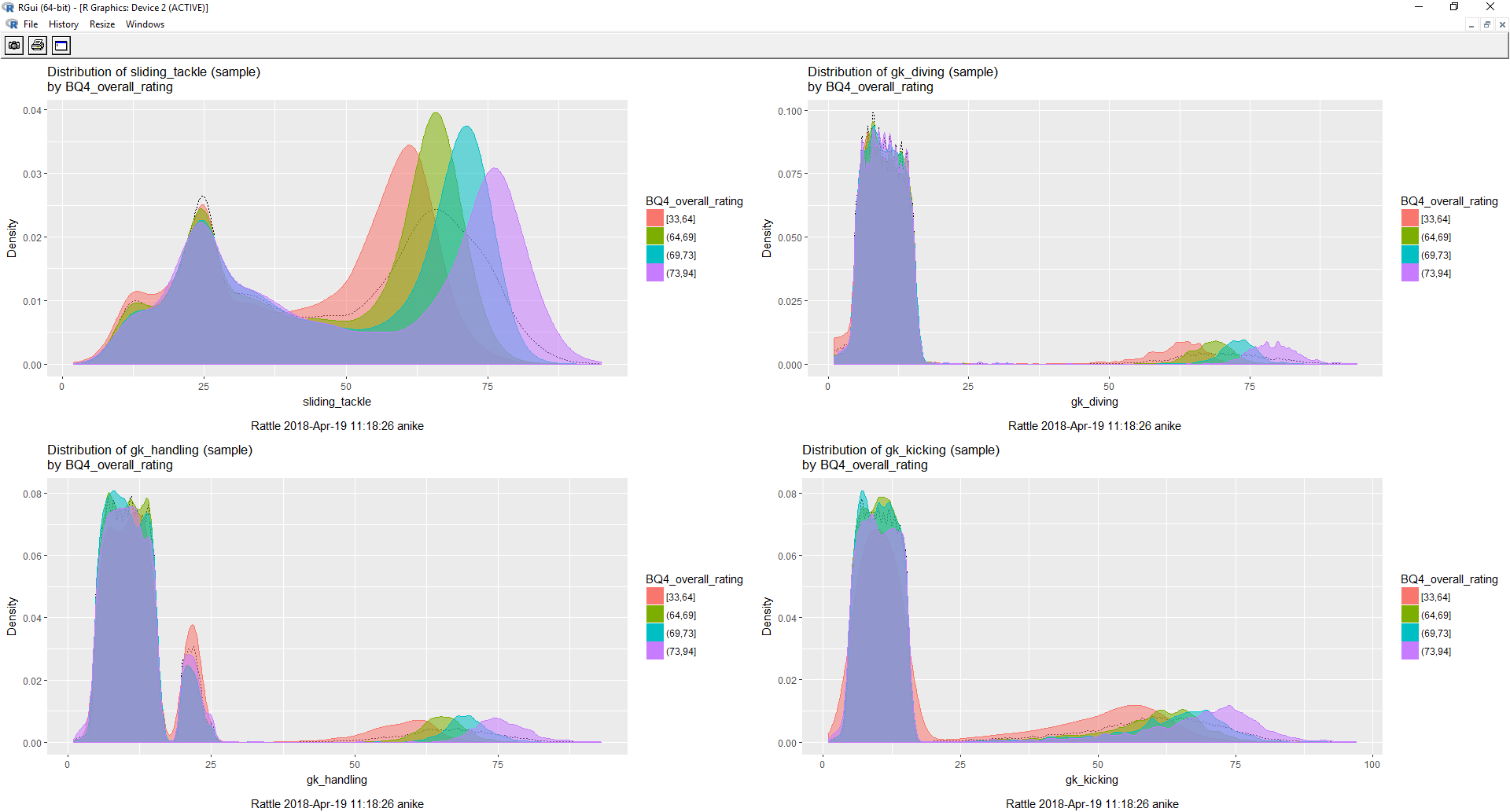
We find in the table player\_attributes most of the values have very good correlation with overall\_rating, but the original type of this target variable is numeric and has more than 60 unique values which makes it very hard for us to draw any meaningful analysis from this attribute. Hence, we use binning(recode) on the attribute overall\_rating and make it our target variable. This target variable is of categoric type and has 4 unique values, each of this value is the range in which the original distribution is adjusted.



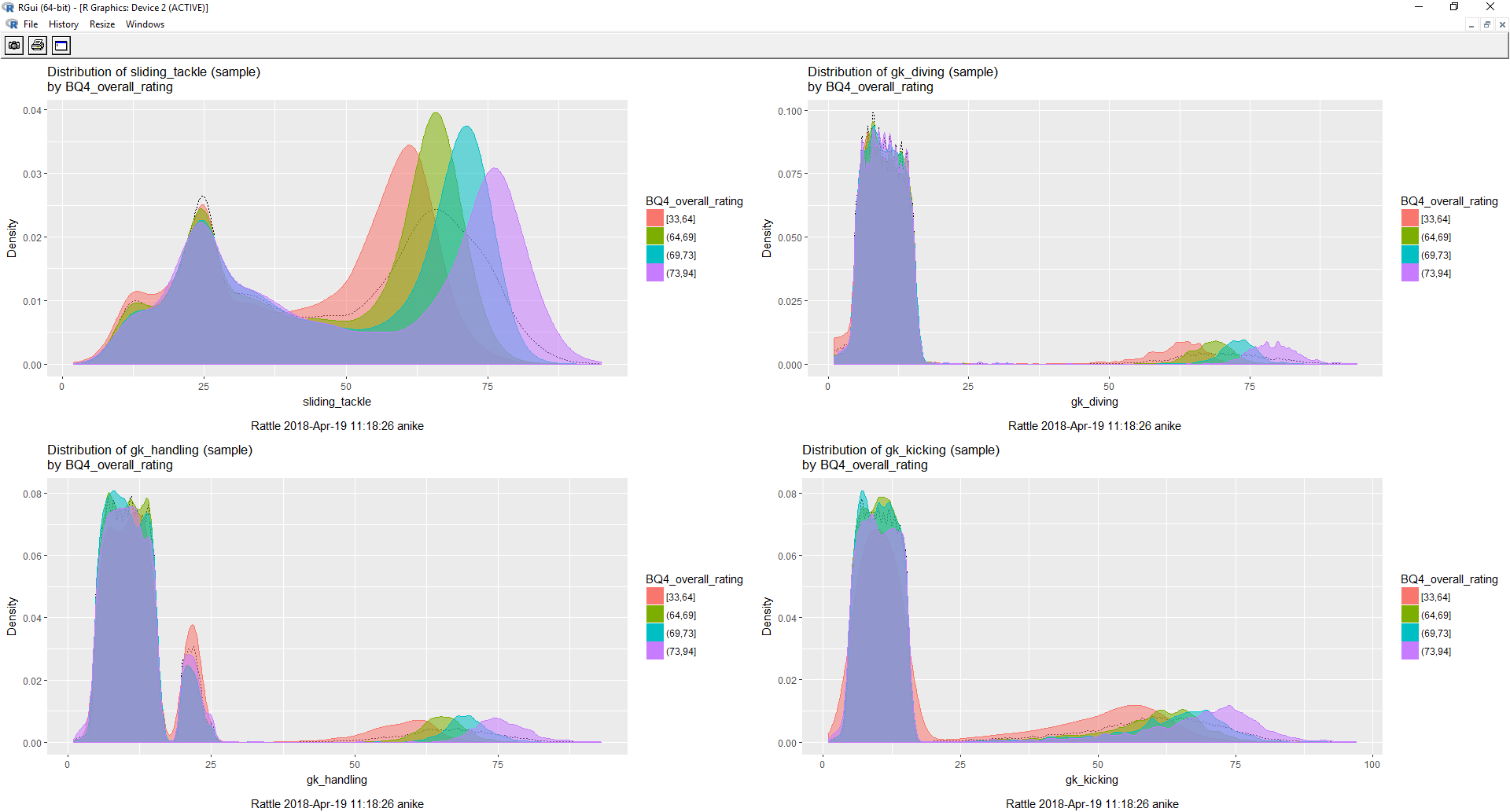
This attribute will help us to visualize our data in the player\_attributes in a much better way. Also, applying transformations on attributes of other tables will not yield any interesting observations different from already existing attributes. Hence, this is the only attribute we will perform transformation on.

# DATA VISUALIZATION

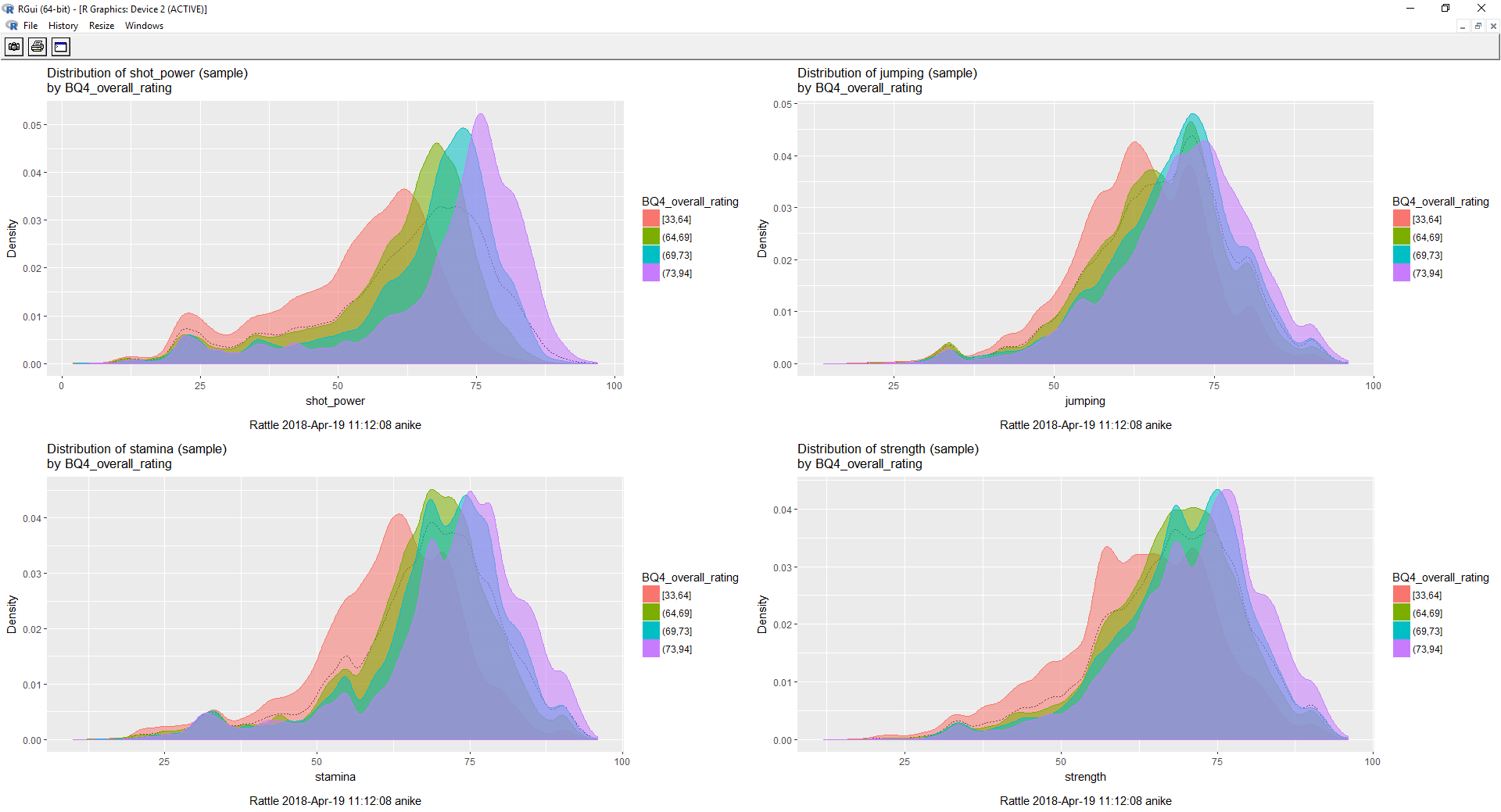
On observing relations between all the attributes and various plots from all the tables we separate out only the one’s who provide information in some or the other. We find out that apart from tables Player\_Attributes and Team\_Attributes there is no other table whose visualization gives any useful result. The selected plots that we found interesting are given below with the description of what we observed for each.



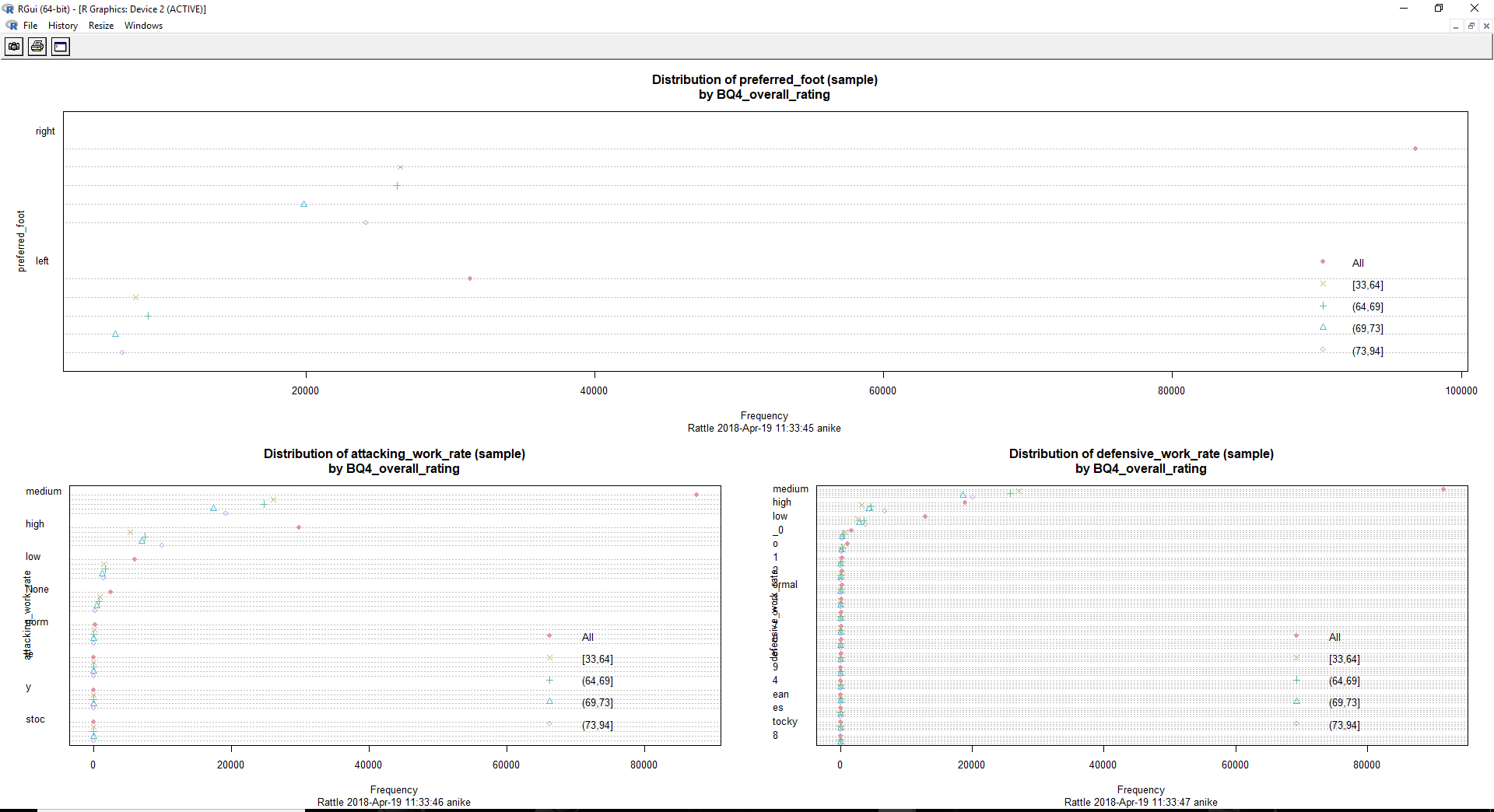
The above histogram plot is distribution of attributes short\_passing, volleys, dribbling, and curve plotted against the transformed target variable overall\_rating. All these groups are rightly skewed, indicating that most of the ratings of each are greater than the central value and each of these attributes have larger number of users having values more than 50 ratings. Also, for each attribute we observe that overall rating range is more than half the value and have more users with ratings in the 3rd and 4th ranges.



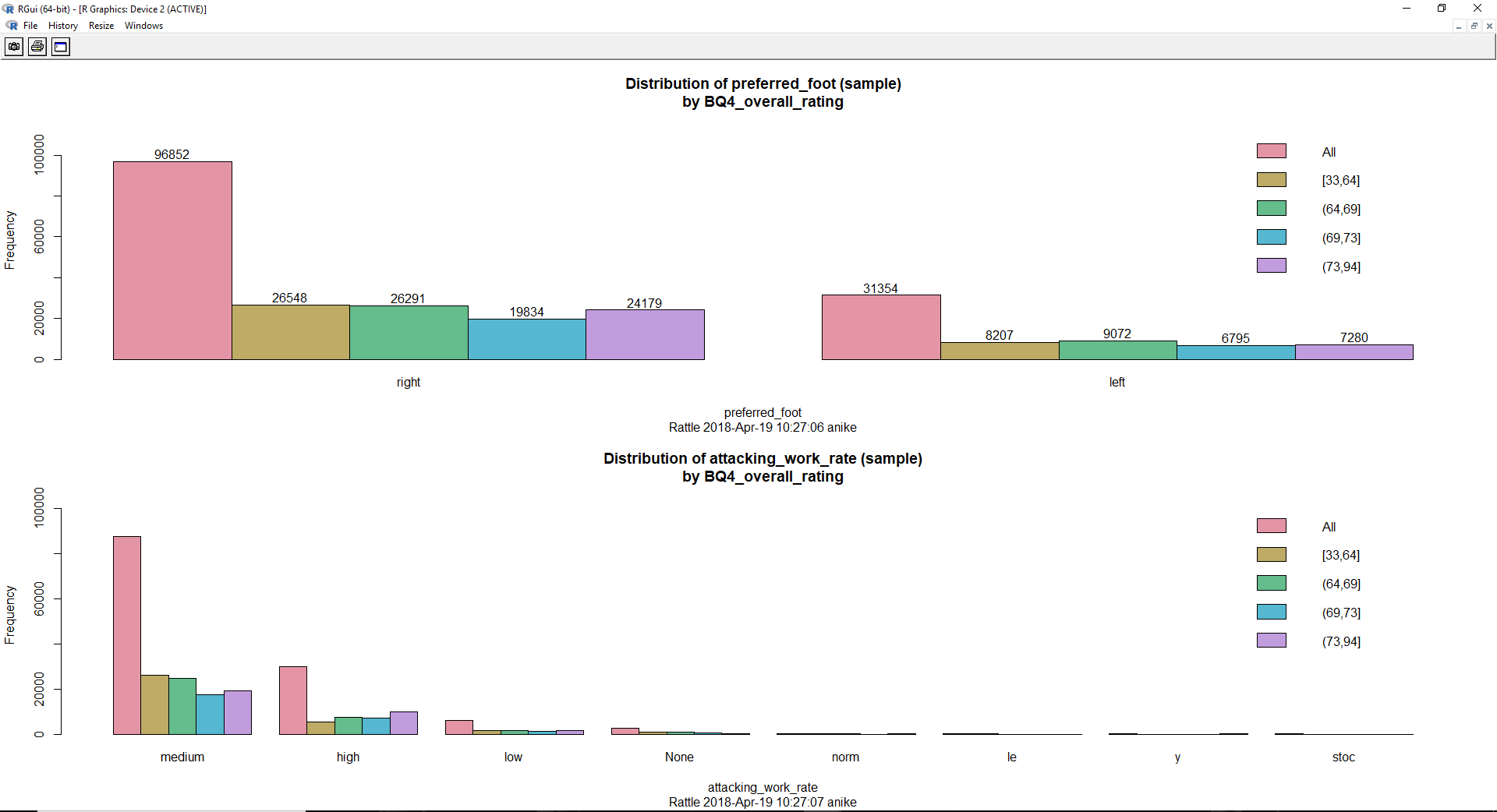
In the above plots we observe the distributions of goalkeepers and plot them against overall ratings. The data is skewed towards the left indicating most of the players are not goalkeepers. However, in the distribution of sliding\_tackle vs overall\_rating we observe that it has values on both left and right sides, indicating that these attributes might be present in defensive type of players like goalkeepers and defenders, from the left distributed values we can infer that certain position players like strikers might not posses this property.



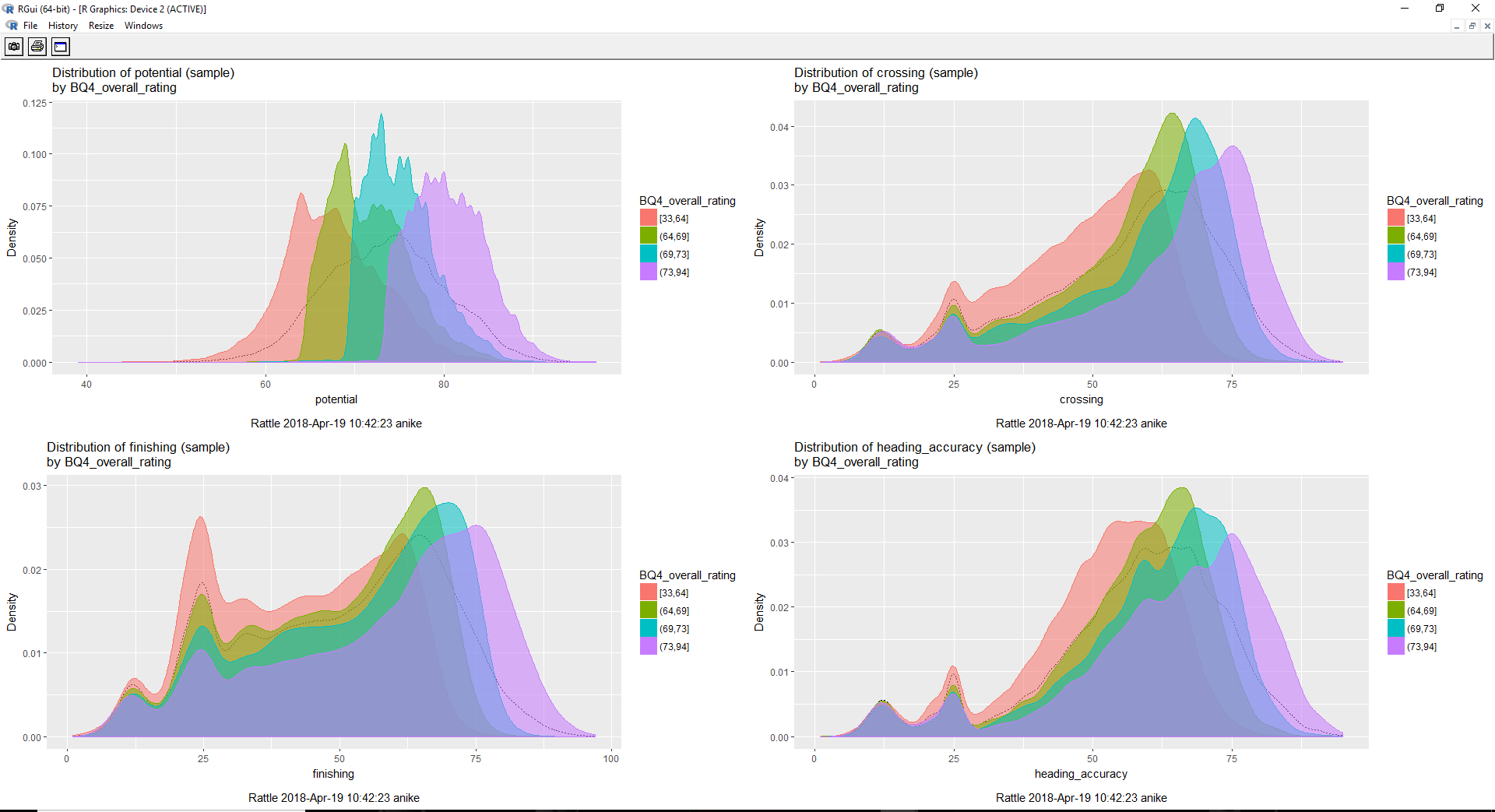
In this distribution we observe that all the plots are rightly skewed, indicating that these attributes are present on almost all the players and it can be inferred that this has close relation with fitness and it is expected by each professional player to have good fitness or at least have a threshold amount of fitness required by the game to survive the game.



In the above plot, we observe the dot plots of preferred\_foot, attacking\_work\_rate, defensive\_work\_rate with the overall rating. In the plots of attacking\_work\_rate and defensive\_work\_rate we observe that the values at zero do not yield in any useful results and other values can be used to get any meaningful results. However, in the plot of preferred\_foot we see that the dot plot can yield meaningful results throughout the distribution.



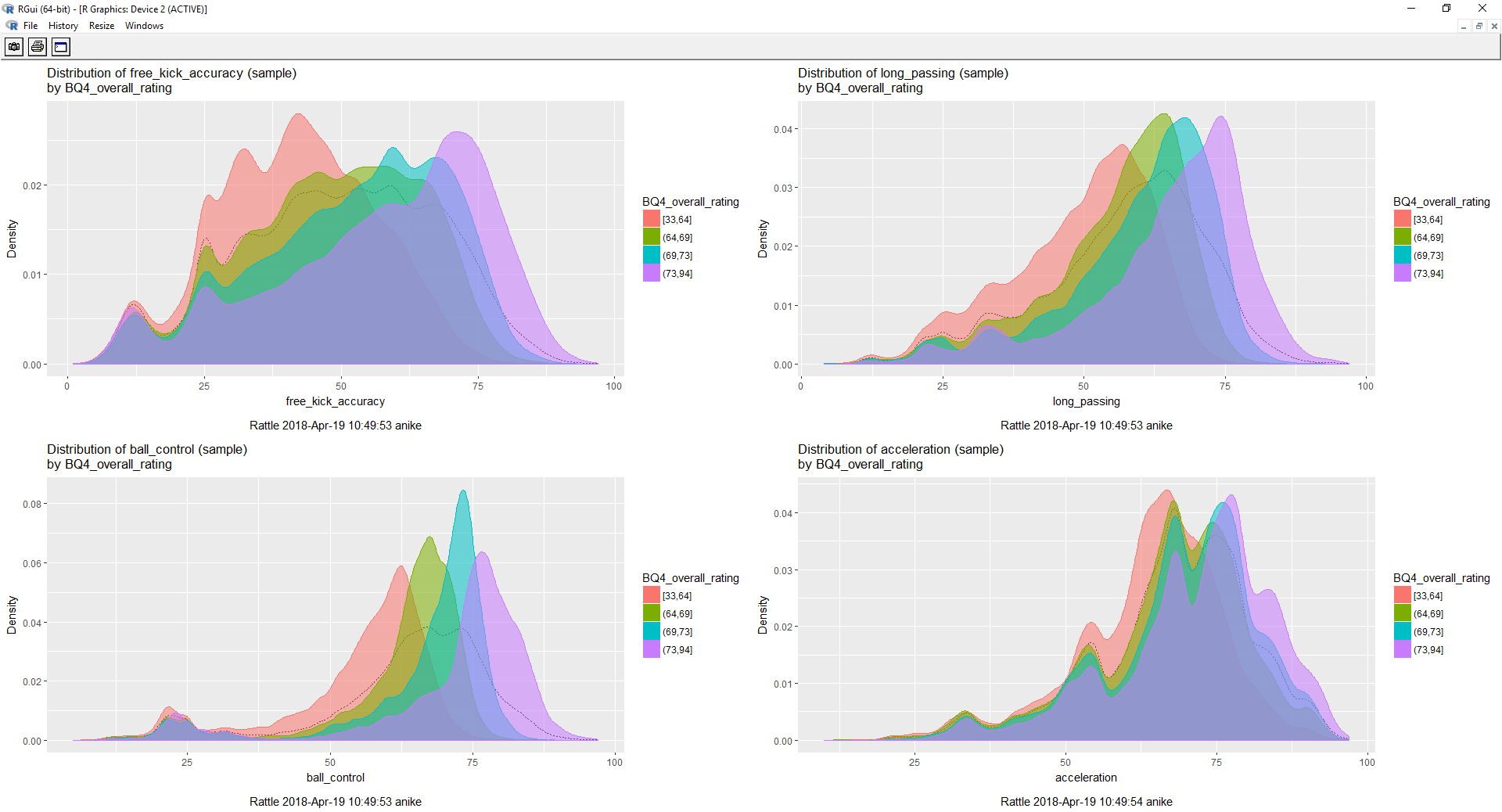
Above bar plot is a distribution of preferred\_foot and attacking\_work\_rate against overall\_rating. From this plot we can say that the overall\_ratings for left and right footers is almost evenly distributed (with difference being in the range of 100-1000 observations). However, when the attacking\_work\_rate is high overall rating is in the range (73,94). From rest of the plot we cannot extract anything useful.



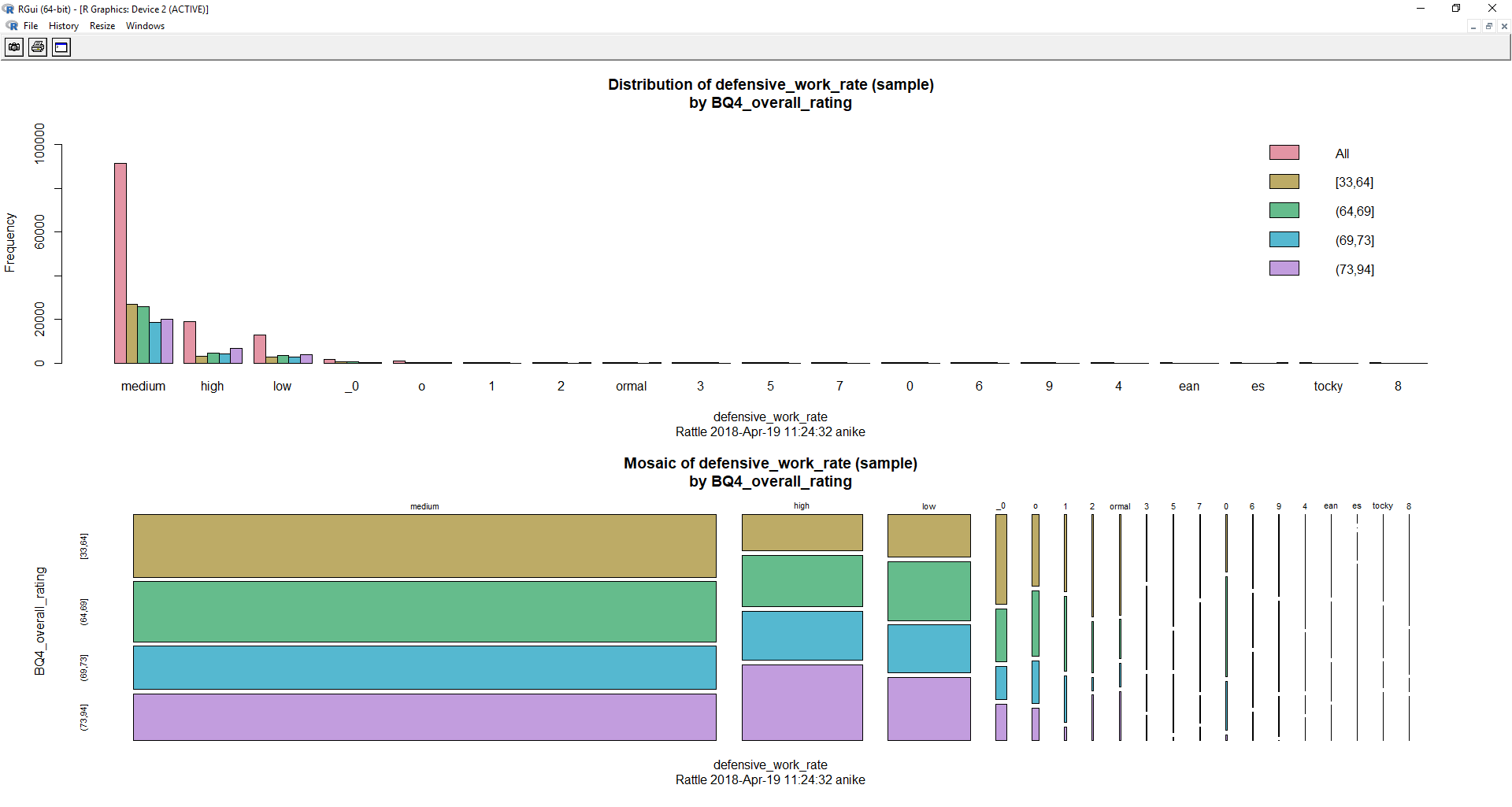
The above histogram plots are of potential, crossing, finishing, and heading\_accuracy vs overall\_rating. In the potential plot, we see that the attribute potential is almost directly proportional to the overall\_rating, i.e. their ranges match. The plot of crossing and heading accuracy are rightly skewed indicating that these are qualities present in most of the players. Finishing has a distribution that is not particularly skewed to any one side, indicating that it is a quality present in many players but more in certain kind of players, like the attacking midfielders or strikers.

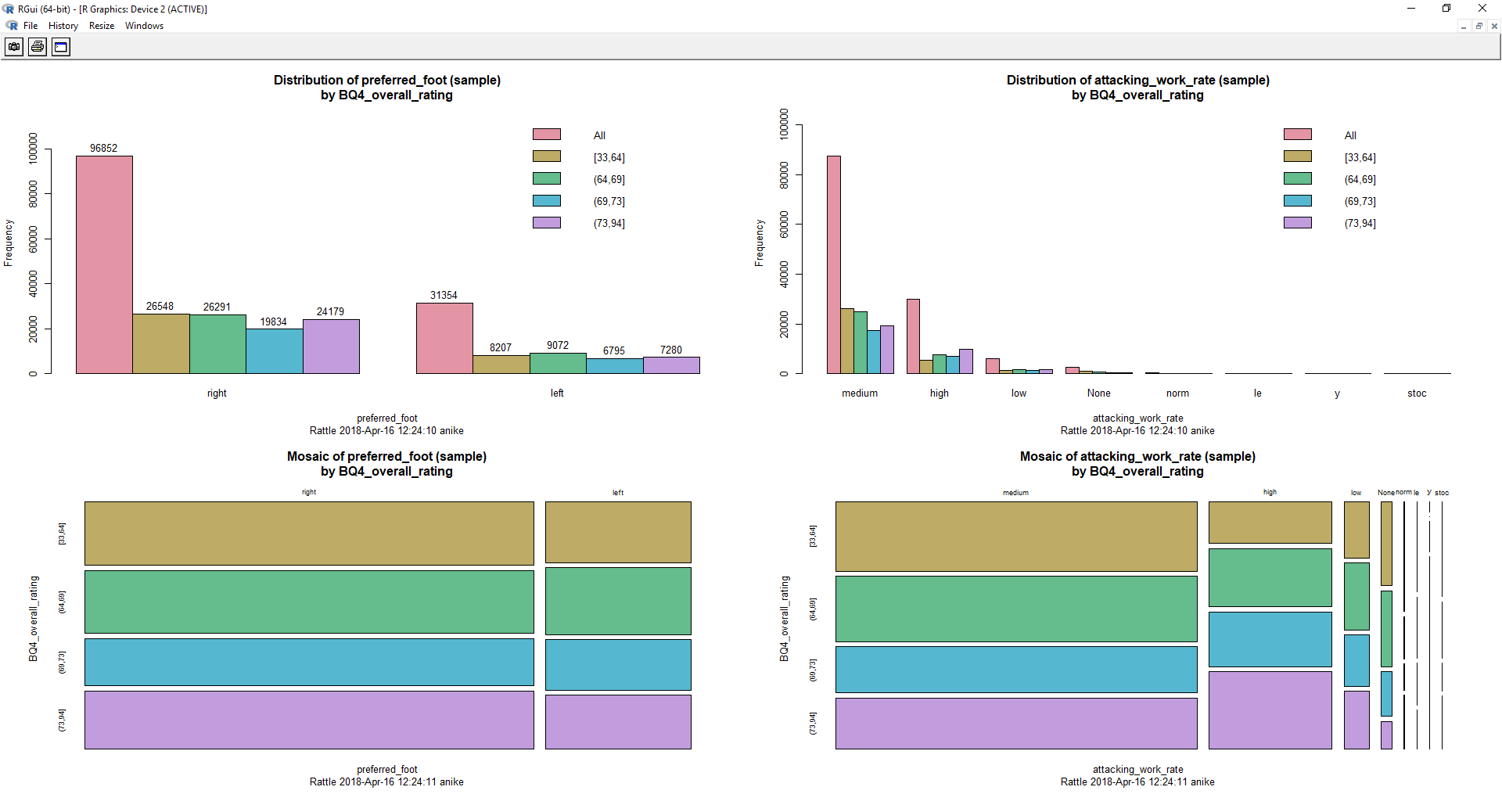


In the above plots we see that the long\_shots, aggression, positioning, and interceptions vs overall\_rating are all mostly skewed towards the right indicating that a certain type of players in each plot need not have higher value for that attribute. Positioning and Aggression is expected from all the players and hence we see that it has very less values on the left side. However, interceptions are a quality which may be less preferred for strikers than other players which is a reason why it has some values on the left side.

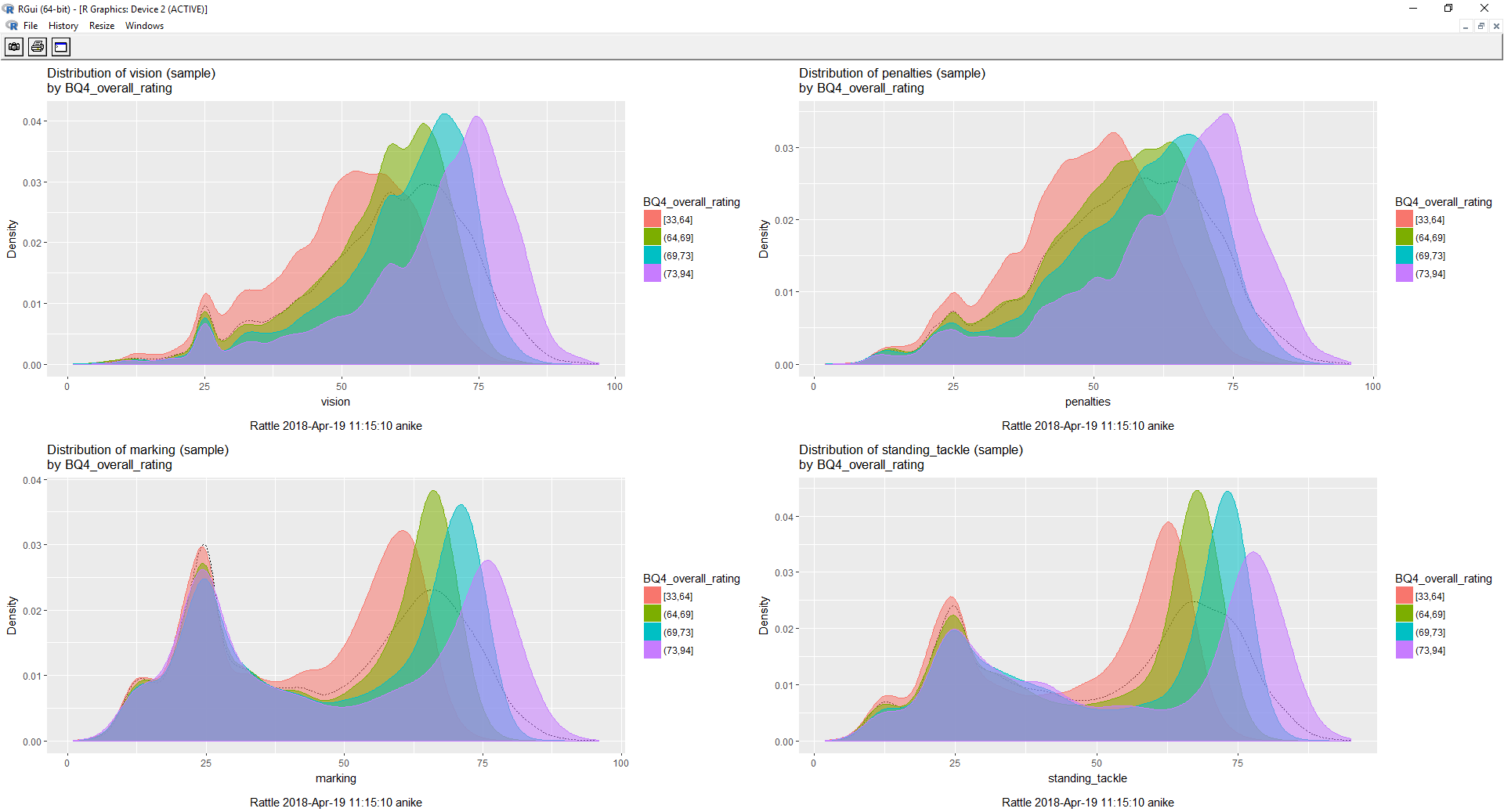


The above plots of free\_kick\_accuracy, long\_passing, ball\_control, and acceleration vs overall\_ratings are all skewed towards the right (except for the attribute free\_kick\_accuracy which is almost evenly distributed) indicating that these are the qualities expected by each player. The free\_kick\_accuracy is expected more out of attacking players (attacking midfielders and strikers).

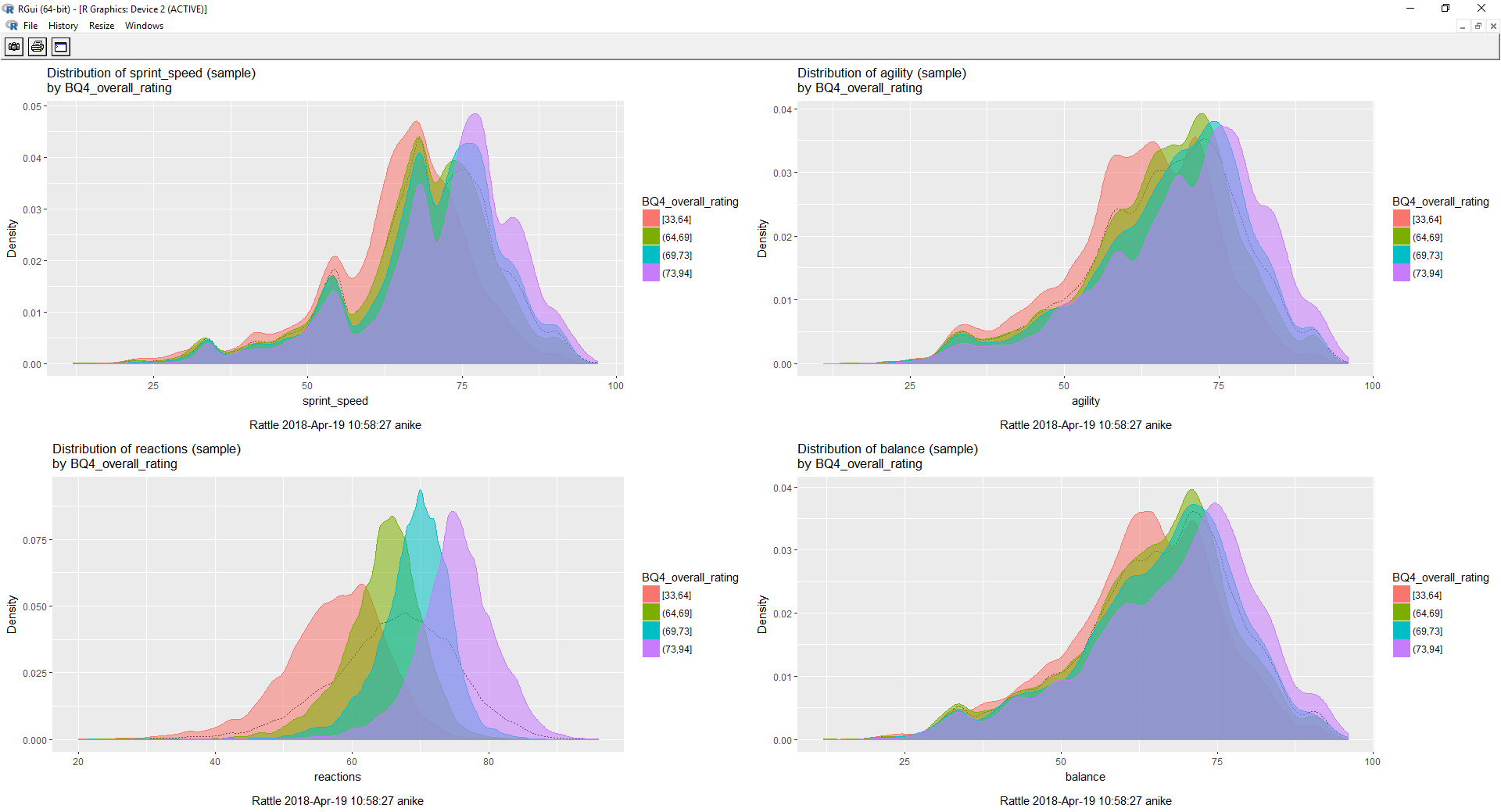
We observe the bar and mosaic plots of defensive\_work\_rate vs overall\_rating. We see in the bar plot that when the defensive\_work\_rate is medium the overall\_ratings are in the range 33,64 and 64,69. From the mosaic plot we observe that the frequency of medium defensive\_work\_rate is the maximum followed by high defensive\_work\_rate and then by the low defensive\_work\_rate. Thus, we can infer from this that most of the players have a medium defensive work rate.



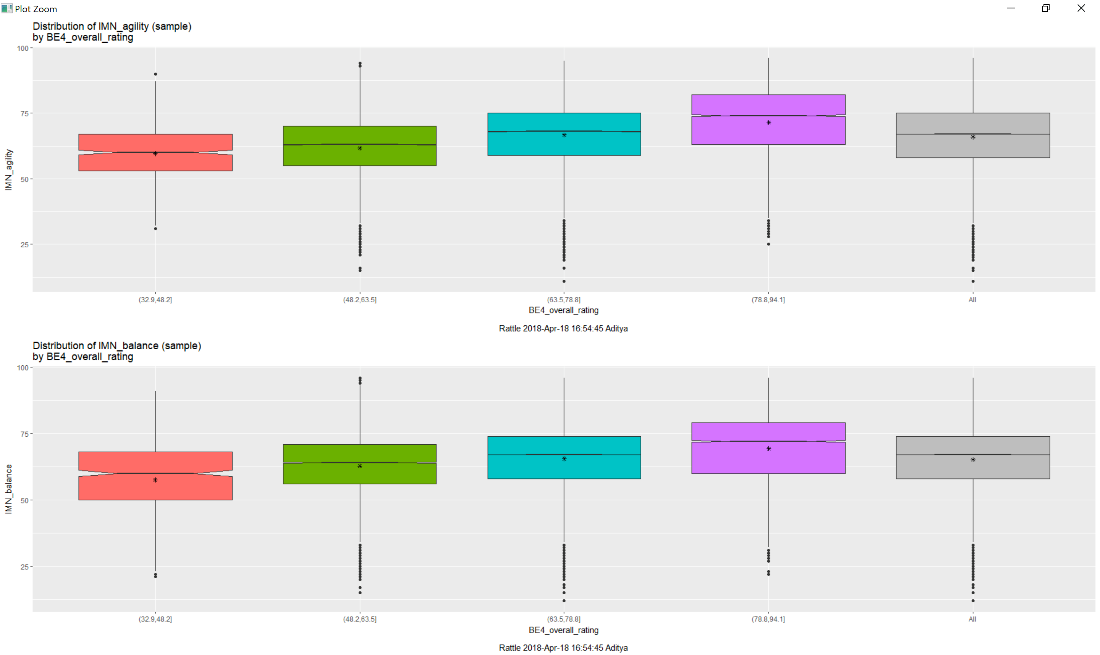
In the above mosaic and bar plots we plot preferred\_foot and attacking\_work\_rate against overall rating. From the graphs we observe that there are more right footers than the left ones. Left footers have the maximum players in the range 64,69. When attacking\_work\_rate is the medium, we observe that most of the players have ratings in the range 33,64 and 64,69. We also observe that most of the players have medium attacking\_work\_rate, then followed by high attacking\_work\_rate and then by low attacking\_work\_rate.



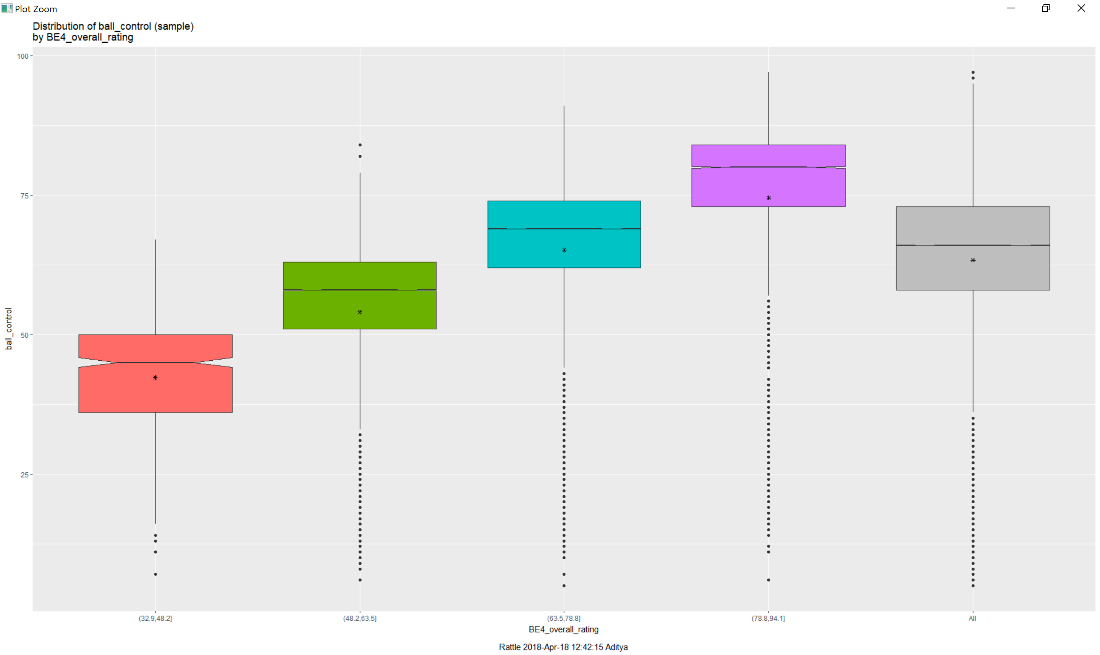
In the above observations we observe that the penalties and vision are skewed towards the right indicating that all the players possess this quality, the values on the left side can be the players who are goalkeeper as it is not essential for them to have higher values for these attributes. Marking and standing\_tackle are qualities not expected from attacking midfielders or strikers who might represent the values on the left side.



In these plots we see that balance, sprint and agility are skewed towards the right indicating these are the attributes expected from all the players and higher these values better are the overall\_ratings. However, in the reactions plot, even though its skewed on the right, it can be observed that there are ranges of reactions in which each reaction occupies a slot in the overall\_rating range.



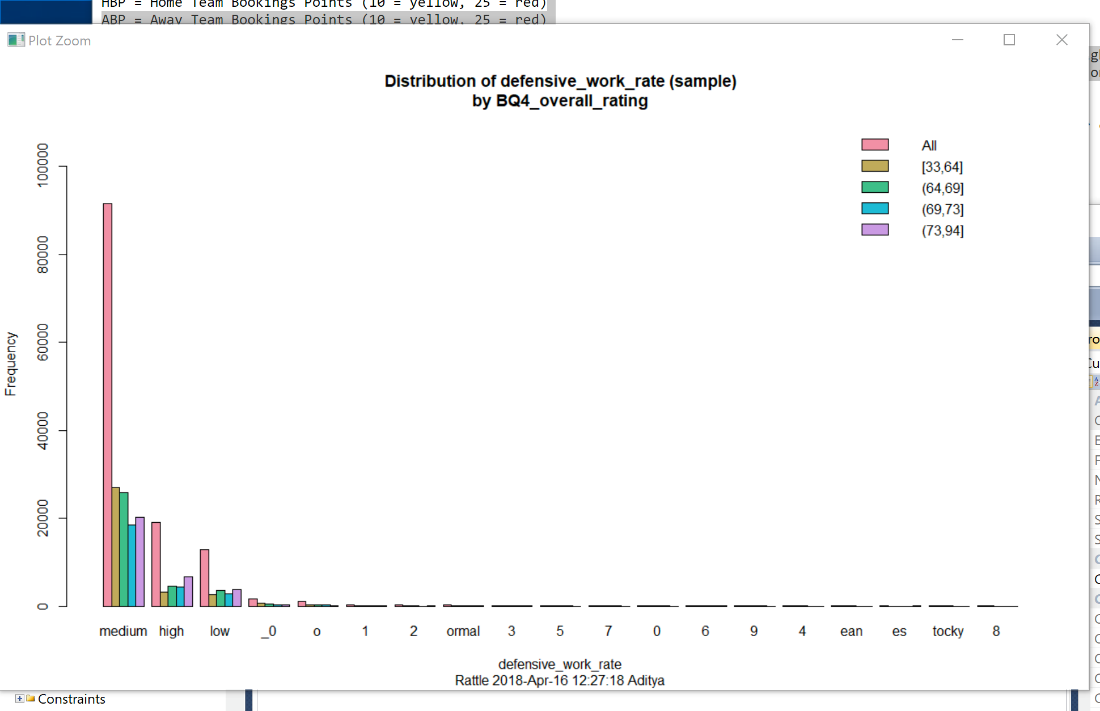
The above box plot shows distribution of player agility and balance with respect to overall rating. It can be observed that, when the overall rating is between (78.8 & 94.1) agility and balance has the greatest value which lies between 62.5 and 75.



The box plot shows distribution of ball control with respect to overall rating. It can be observed that the value of ball control is the highest when the overall rating is between (78.8 & 94.1). The value of the mean is closer to 75. However, the median is greater 75.

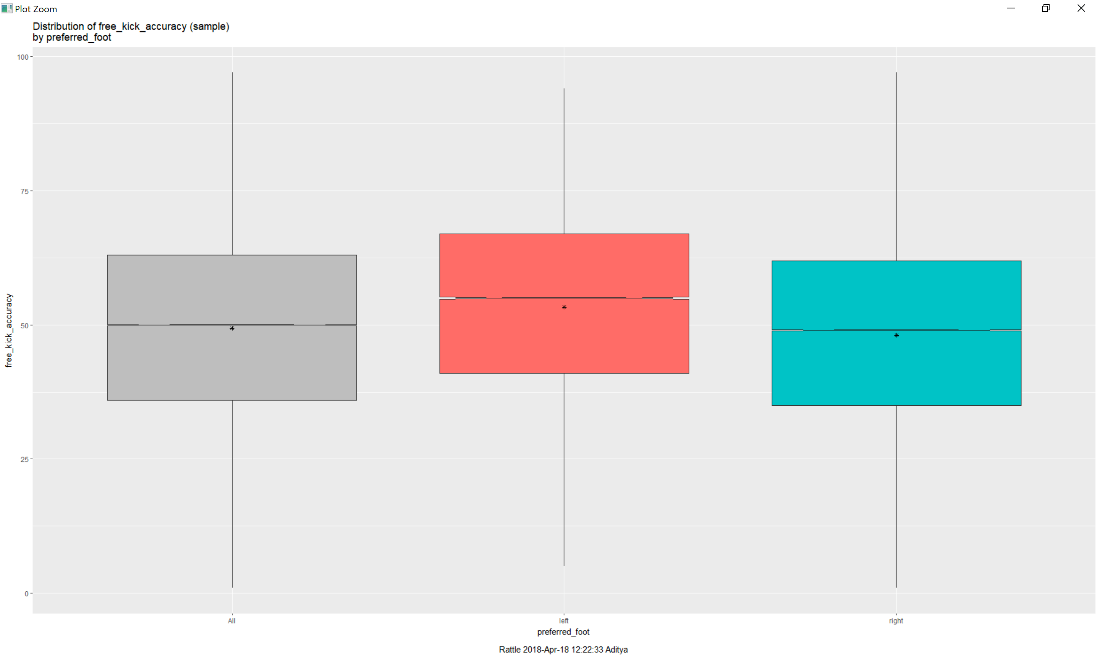


Visualizing distribution of crossing with respect to the preferred foot, With the mean/median of left greater than that of right. It can be inferred that most players prefer left foot over right while crossing.

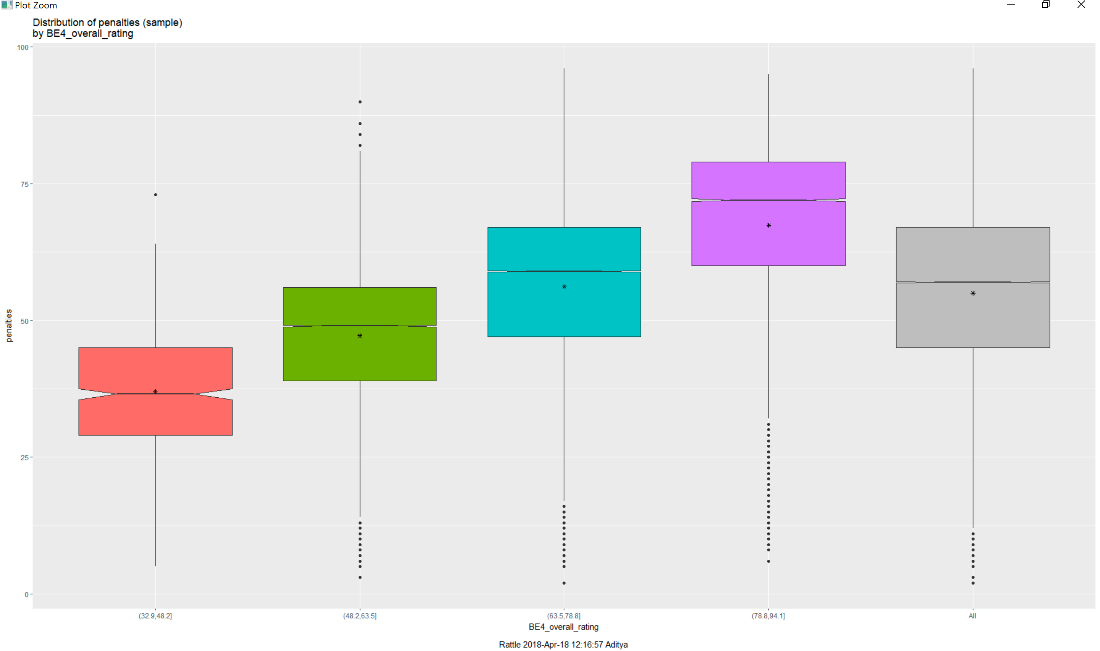


From the above plot it can be concluded that, more number of players with rating (33,64)

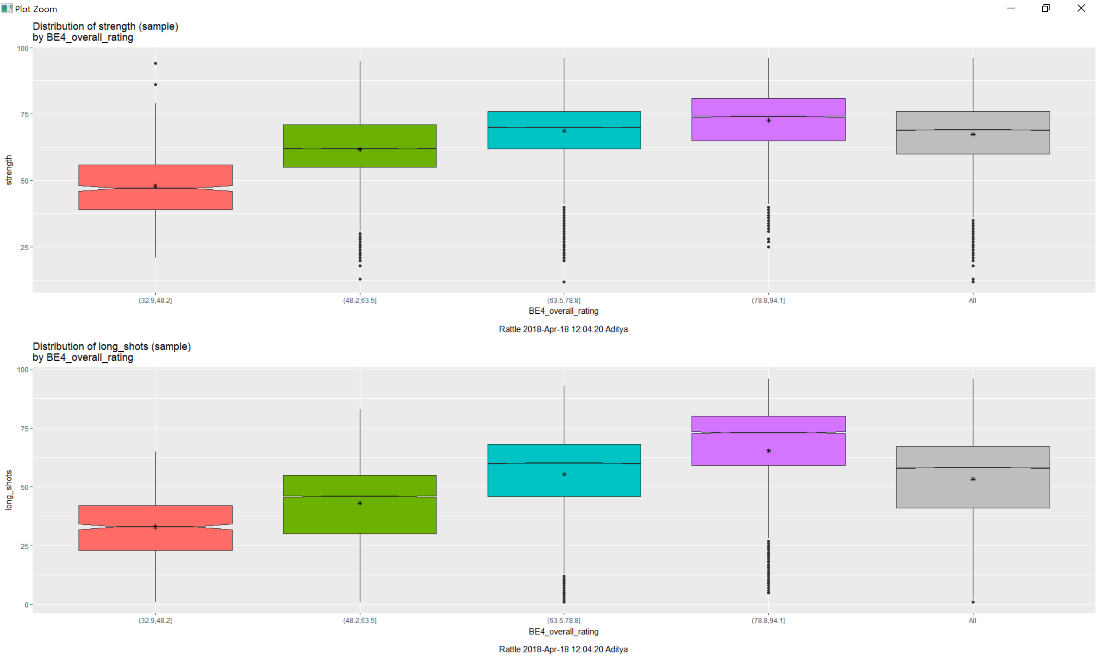
prefer defensive work rate over those with rating (69,73) and (73,94).



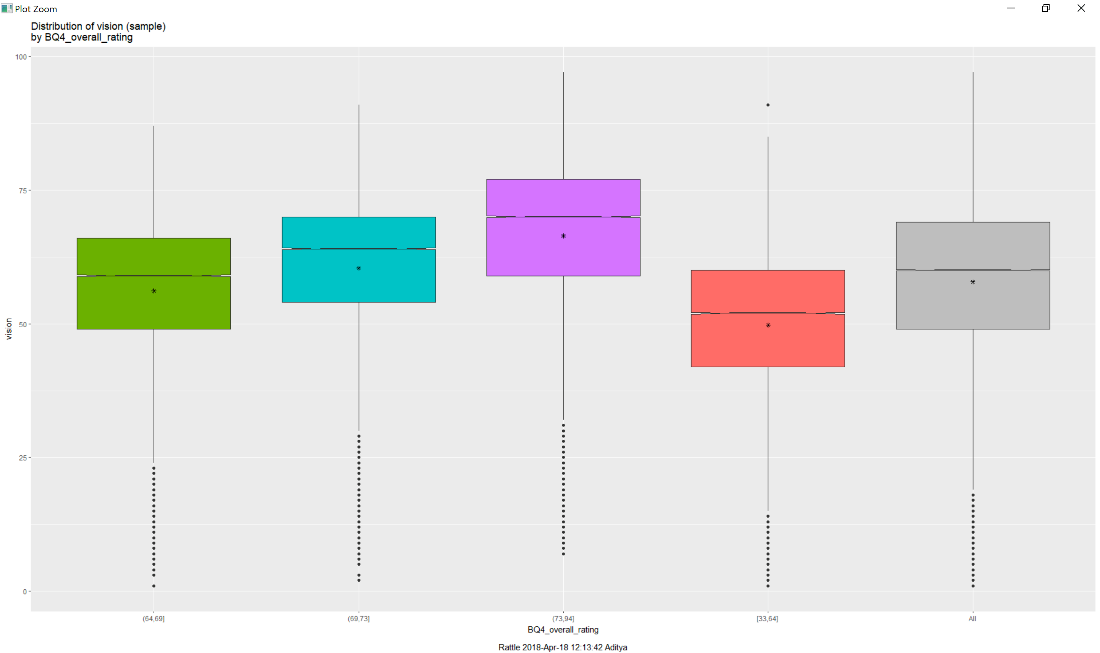
The inference of the above box plot is; players who prefer left foot over right have a greater accuracy when it comes to a free kick.



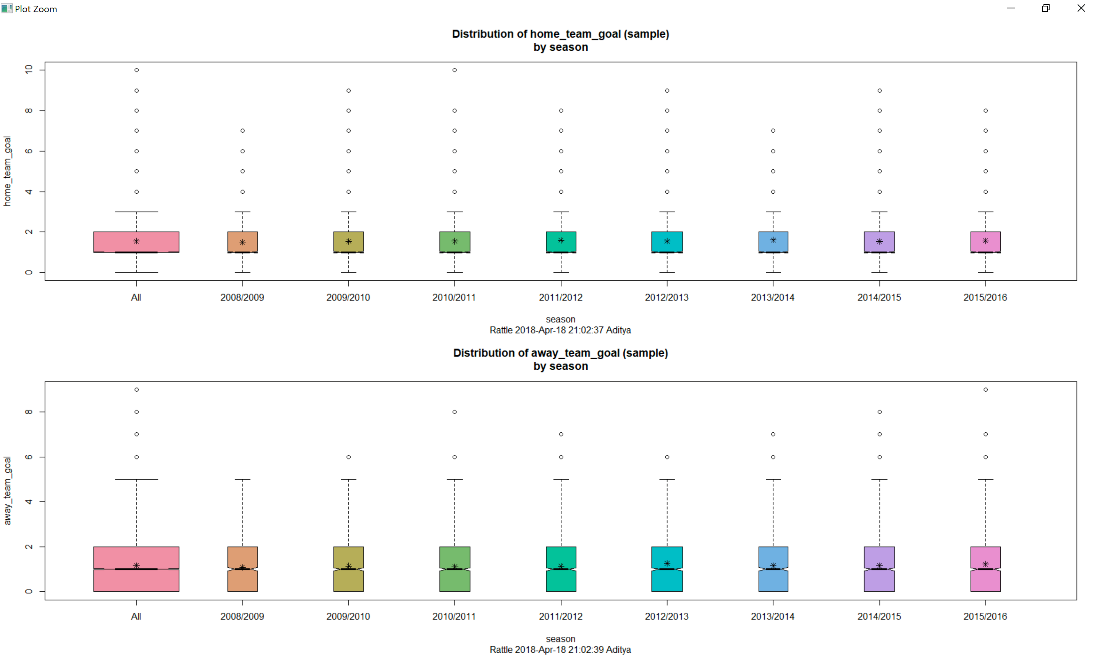
From the previous box plots, it was inferred that players with a high rating have a higher agility, balance and ball control. But, On the contrary there is a higher probability of a penalty among highly rated players.



The strength and ability to make a long shot is common among players with high rating and less common among players with low rating (<48.2).



All players have a vision rating between (50 & 75). But, the only exception is nearly half the players having a rating (<64) have a vision rating less than 50. From all the previous plots, it can be observed that the player with rating greater than 73 but less than 94 have greater value for all the attributes and hence are more likely to be expensive to buy. However, players with high rating are more susceptible to penalties.



In the above box plot, It can be inferred that there is no major difference among home team goals and away team goals from season 2009 to 2016.

# PREDICTIVE ANALYSIS

## Top 20 Goalkeepers

select top 20 Player.player\_name as Name, avg(Player\_Attributes.gk\_diving) as Diving, avg(Player\_Attributes.gk\_handling) as Handling,

avg(Player\_Attributes.gk\_kicking) as Kicking, avg(Player\_Attributes.gk\_positioning) as Positioning, avg(Player\_Attributes.gk\_reflexes) as Reflexes,

(2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case  
 when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 40  
 then 'Retired'  
 else  
  case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 40

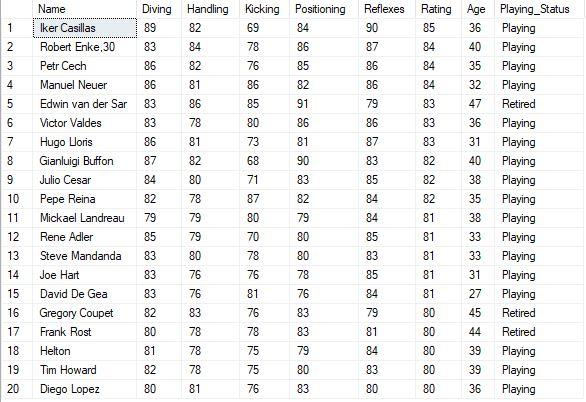
   then 'Playing'  
  end

end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

group by Player.player\_name, Player.birthday

order by Rating desc



## Top 10 Player statistics

select top 10 Player.player\_name as Name, avg(overall\_rating) as Lifetime\_Average,

avg(Player\_Attributes.acceleration) as Acceleration, avg(Player\_Attributes.aggression) as Aggresion,

avg(Player\_Attributes.agility) as Agility, avg(Player\_Attributes.balance) as Balance,

avg(Player\_Attributes.ball\_control) as Ball\_control, avg(Player\_Attributes.crossing) as Crossing,

avg(Player\_Attributes.curve) as Curve, avg(Player\_Attributes.dribbling) as Dribbling,

avg(Player\_Attributes.finishing) as Finishing, avg(Player\_Attributes.free\_kick\_accuracy) as Free\_Kick\_Accuracy,

avg(Player\_Attributes.heading\_accuracy) as Header\_Accuracy, avg(Player\_Attributes.interceptions) as Interceptions,

avg(Player\_Attributes.jumping) as Jumping, avg(Player\_Attributes.long\_passing) as Long\_Passing,

avg(Player\_Attributes.long\_shots) as Long\_Shots, avg(Player\_Attributes.marking) as Marking,

avg(Player\_Attributes.penalties) as Penalties,avg(Player\_Attributes.positioning) as Positioning,

avg(Player\_Attributes.potential) as Potential, avg(Player\_Attributes.reactions) as Reactions,

avg(Player\_Attributes.short\_passing) as Short\_Passing, avg(Player\_Attributes.shot\_power) as Shot\_Power,

avg(Player\_Attributes.sliding\_tackle) as Sliding\_Tackle, avg(Player\_Attributes.sprint\_speed) as Sprint\_Speed,

avg(Player\_Attributes.stamina) as Stamina, avg(Player\_Attributes.standing\_tackle) as Standing\_Tackle,

avg(Player\_Attributes.strength) as Strength, avg(Player\_Attributes.vision) as Vision,

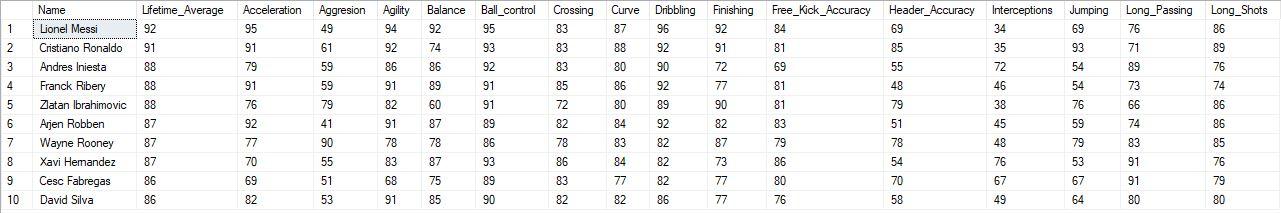
avg(Player\_Attributes.volleys) as Volleys

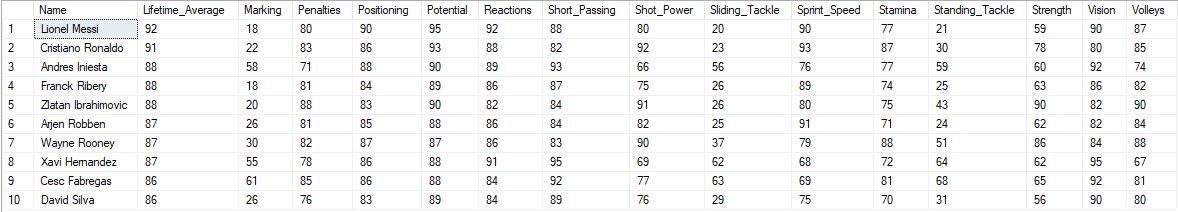
from Player\_Attributes FULL OUTER JOIN Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where overall\_rating > 0

group by Player\_Attributes.player\_api\_id, Player.player\_name

order by lifetime\_average DESC;





## Top 20 strikers

select top 20 Player.player\_name as Name, avg(Player\_Attributes.crossing) as Crossing, avg(Player\_Attributes.short\_passing) as Short\_Passing,

avg(Player\_Attributes.vision) as Vision, avg(Player\_Attributes.ball\_control) as Ball\_Control,

avg(Player\_Attributes.penalties) as Penalties, avg(Player\_Attributes.potential) as Potential,

avg(Player\_Attributes.volleys) as Volleys, avg(Player\_Attributes.finishing) as Finishing, avg(Player\_Attributes.dribbling) as Dribbling,

(avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

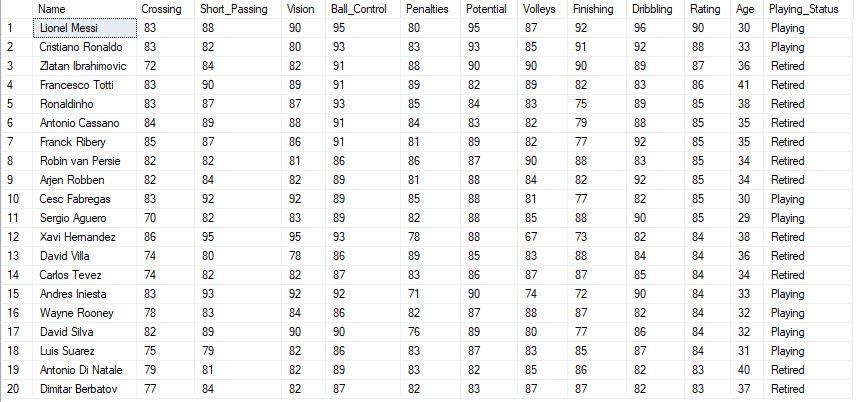
 end

end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

group by Player.player\_name, Player.birthday

order by Rating desc



select top 20 Player.player\_name as Name, avg(Player\_Attributes.crossing) as Crossing, avg(Player\_Attributes.short\_passing) as Short\_Passing,

avg(Player\_Attributes.vision) as Vision, avg(Player\_Attributes.ball\_control) as Ball\_Control,

avg(Player\_Attributes.penalties) as Penalties, avg(Player\_Attributes.potential) as Potential,

avg(Player\_Attributes.volleys) as Volleys, avg(Player\_Attributes.finishing) as Finishing, avg(Player\_Attributes.dribbling) as Dribbling,

(avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

 end

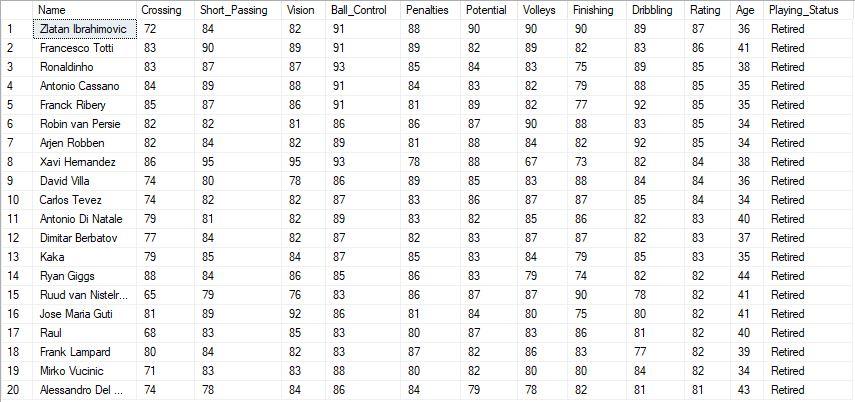
end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

group by Player.player\_name, Player.birthday

order by Rating desc



select top 20 Player.player\_name as Name, avg(Player\_Attributes.crossing) as Crossing, avg(Player\_Attributes.short\_passing) as Short\_Passing,

avg(Player\_Attributes.vision) as Vision, avg(Player\_Attributes.ball\_control) as Ball\_Control,

avg(Player\_Attributes.penalties) as Penalties, avg(Player\_Attributes.potential) as Potential,

avg(Player\_Attributes.volleys) as Volleys, avg(Player\_Attributes.finishing) as Finishing, avg(Player\_Attributes.dribbling) as Dribbling,

(avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

 end

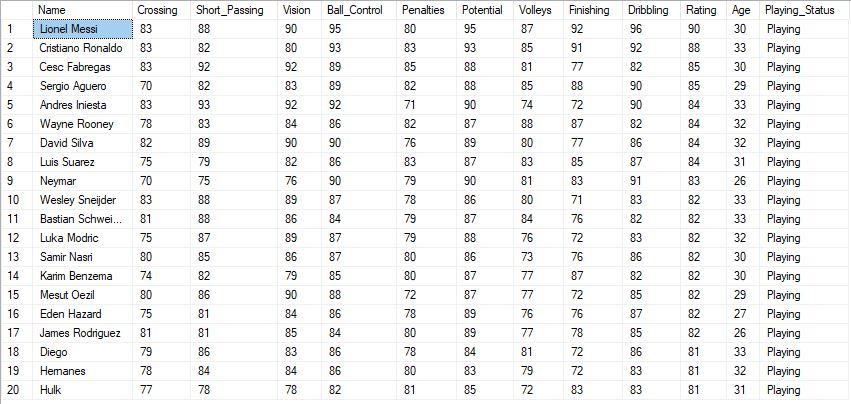
end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

group by Player.player\_name, Player.birthday

order by Rating desc



## Top 20 defenders

select top 20 Player.player\_name as Name, avg(Player\_Attributes.heading\_accuracy) as Heading\_Accuracy, avg(Player\_Attributes.interceptions) as Interceptions,

avg(Player\_Attributes.sliding\_tackle) as Sliding\_Tackles, avg(Player\_Attributes.standing\_tackle) as Standing\_Tackles,

avg(Player\_Attributes.aggression) as Aggression, avg(Player\_Attributes.strength) as Strength,

(avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

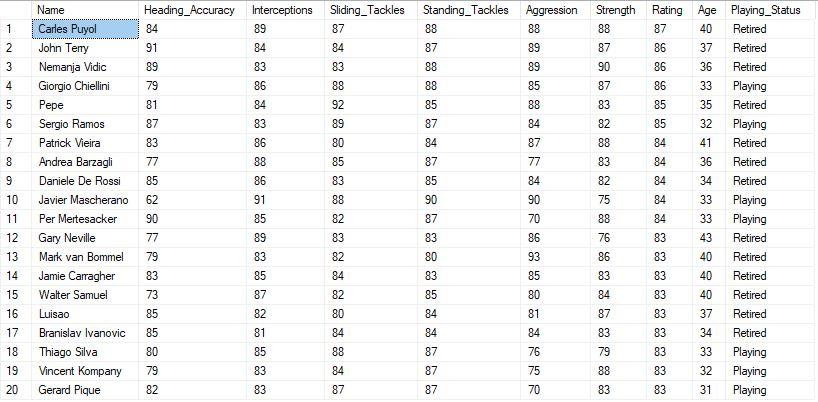
 end

end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

group by Player.player\_name, Player.birthday

order by Rating desc



select top 20 Player.player\_name as Name, avg(Player\_Attributes.heading\_accuracy) as Heading\_Accuracy, avg(Player\_Attributes.interceptions) as Interceptions,

avg(Player\_Attributes.sliding\_tackle) as Sliding\_Tackles, avg(Player\_Attributes.standing\_tackle) as Standing\_Tackles,

avg(Player\_Attributes.aggression) as Aggression, avg(Player\_Attributes.strength) as Strength,

(avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

 end

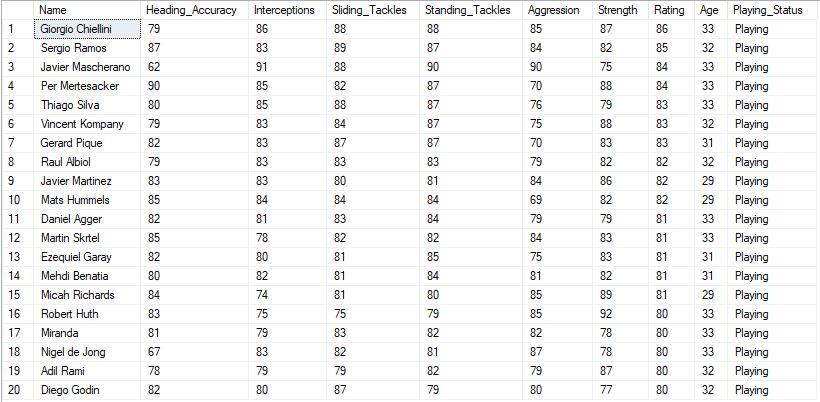
end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

group by Player.player\_name, Player.birthday

order by Rating desc



select top 20 Player.player\_name as Name, avg(Player\_Attributes.heading\_accuracy) as Heading\_Accuracy, avg(Player\_Attributes.interceptions) as Interceptions,

avg(Player\_Attributes.sliding\_tackle) as Sliding\_Tackles, avg(Player\_Attributes.standing\_tackle) as Standing\_Tackles,

avg(Player\_Attributes.aggression) as Aggression, avg(Player\_Attributes.strength) as Strength,

(avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

 end

end) as Playing\_Status

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

group by Player.player\_name, Player.birthday

order by Rating desc



## Top 10 and bottom 10 lifetime average ratings

select top 10 Player\_Attributes.player\_api\_id, Player.player\_name, avg(overall\_rating) as lifetime\_average

from Player\_Attributes FULL OUTER JOIN Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where overall\_rating > 0

group by Player\_Attributes.player\_api\_id, Player.player\_name

order by lifetime\_average DESC;



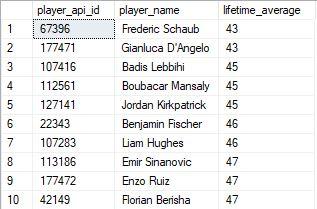
select top 10 Player\_Attributes.player\_api\_id, Player.player\_name, avg(overall\_rating) as lifetime\_average

from Player\_Attributes FULL OUTER JOIN Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where overall\_rating > 0

group by Player\_Attributes.player\_api\_id, Player.player\_name

order by lifetime\_average ASC;



## Team matches with league details and match results

select top 25 M.id, Country.name as Country, League.name as 'League Name', m.season as Season,

substring(m.date,1,10) AS Date, HomeTeam.team\_long\_name as 'Home Team', AwayTeam.team\_long\_name as 'Away Team',

M.home\_team\_goal as 'Home Team Goals', M.away\_team\_goal as 'Away Team Goals',

(case

when M.home\_team\_goal > M.away\_team\_goal

then 'Home team won'

else

 case

  when M.home\_team\_goal < M.away\_team\_goal

  then 'Away team won'

  else

   case

    when M.home\_team\_goal = M.away\_team\_goal

    then 'Match Drawn'

   end

 end

end) as 'Win/Loss/Draw Class'

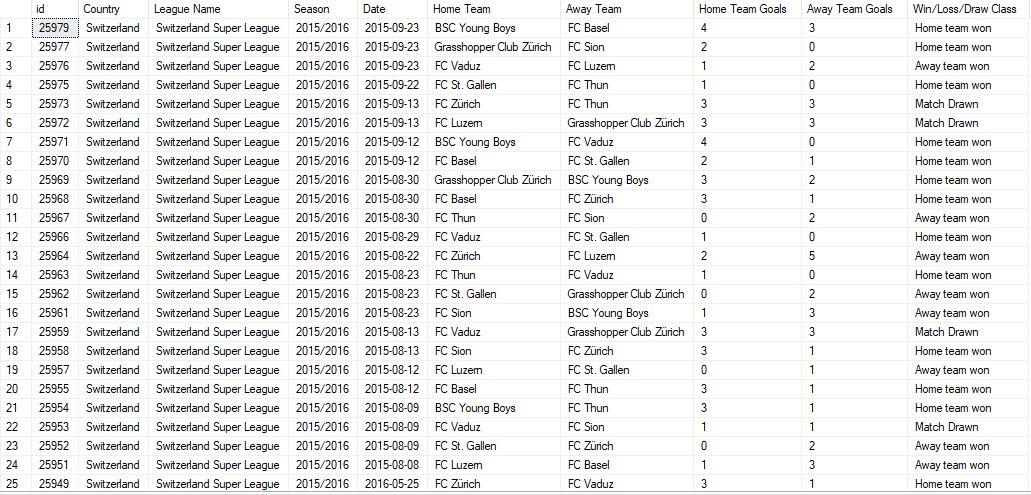
from match as M

inner join team as HomeTeam on M.home\_team\_api\_id = HomeTeam.team\_api\_id

inner join team as AwayTeam on M.away\_team\_api\_id = AwayTeam.team\_api\_id

inner join Country on M.country\_id = Country.id

inner join League on M.country\_id = League.id



select top 25 M.id, Country.name as Country, League.name as 'League Name', m.season as Season,

substring(m.date,1,10) AS Date, HomeTeam.team\_long\_name as 'Home Team', AwayTeam.team\_long\_name as 'Away Team',

M.home\_team\_goal as 'Home Team Goals', M.away\_team\_goal as 'Away Team Goals',

(case

when M.home\_team\_goal > M.away\_team\_goal

then 'Home team won'

else

 case

  when M.home\_team\_goal < M.away\_team\_goal

  then 'Away team won'

  else

   case

    when M.home\_team\_goal = M.away\_team\_goal

    then 'Match Drawn'

   end

 end

end) as 'Win/Loss/Draw Class'

from match as M

inner join team as HomeTeam on M.home\_team\_api\_id = HomeTeam.team\_api\_id

inner join team as AwayTeam on M.away\_team\_api\_id = AwayTeam.team\_api\_id

inner join Country on M.country\_id = Country.id

inner join League on M.country\_id = League.id

order by id DESC



## Retired footballers with ages - top and bottom

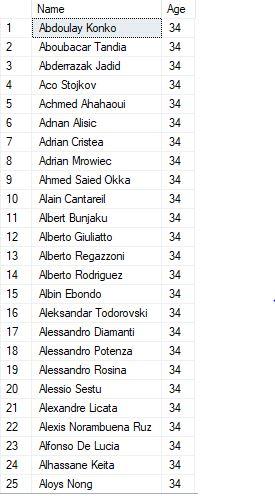
use European\_Soccer

select top 25 player\_name AS Name,Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age

from Player

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) >= 34

order by Age asc



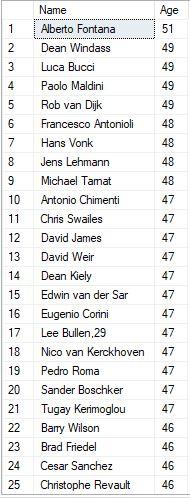
use European\_Soccer

select top 25 player\_name AS Name,Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age

from Player

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) >= 34

order by Age desc



## Potential vs height

use European\_Soccer

SELECT

CASE

WHEN ROUND(height,0)<165

then 165

WHEN ROUND(height,0)>195

then 195

ELSE ROUND(height,0)

END AS calc\_height,

COUNT(height) AS distribution, (avg(PA\_Grouped.avg\_overall\_rating)) AS avg\_overall\_rating, (avg(PA\_Grouped.avg\_potential)) AS avg\_potential,

AVG(weight) AS avg\_weight

FROM PLAYER

LEFT JOIN (SELECT Player\_Attributes.player\_api\_id,

avg(Player\_Attributes.overall\_rating) AS avg\_overall\_rating,

avg(Player\_Attributes.potential) AS avg\_potential

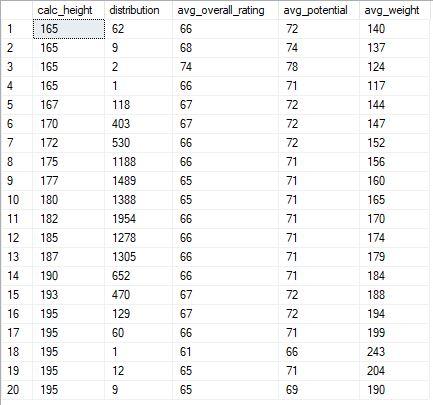
FROM Player\_Attributes

GROUP BY Player\_Attributes.player\_api\_id)

AS PA\_Grouped ON PLAYER.player\_api\_id = PA\_Grouped.player\_api\_id

GROUP BY player.height

ORDER BY calc\_height



## Player lifetime rating average

use European\_Soccer

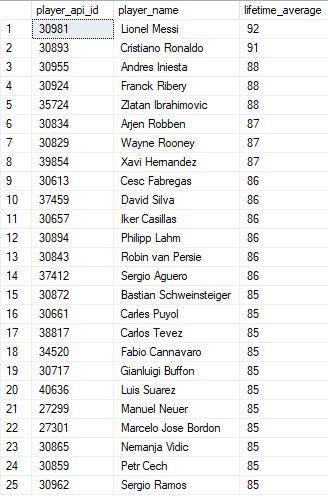
select top 25 Player\_Attributes.player\_api\_id, Player.player\_name, avg(overall\_rating) as lifetime\_average

from Player\_Attributes FULL OUTER JOIN Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where overall\_rating > 0

group by Player\_Attributes.player\_api\_id, Player.player\_name

order by lifetime\_average DESC;



## Player in terms of playing position predicted

use European\_Soccer

select Player.player\_name as Name, avg(overall\_rating) as 'Lifetime Rating',

(avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11 as 'Defender Rating',

(avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18 as 'Striker Rating',

(2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8 as 'Goalkeeper Rating',

(case

when ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8) > ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) and ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8) > ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11)

then 'Goalkeeper'

else

 case

  when ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) >= ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) and ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) > ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8)

   then case

    when ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) - ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) <= 20 and ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) - ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) >= 0

     then 'Defensive Midfielder'

    else 'Defender'

   end

 else

  case

   when ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) > ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) and ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) > ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8)

   then case

     when ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) - ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) <= 25 and ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) - ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) >= 0

      then 'Attacking Midfielder'

     else 'Striker'

     end

  end

 end

end) as 'Ideal Player Position'

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where Player\_Attributes.heading\_accuracy >= 0 and Player\_Attributes.interceptions >= 0 and Player\_Attributes.sliding\_tackle >= 0 and

Player\_Attributes.standing\_tackle >= 0 and Player\_Attributes.aggression >= 0 and Player\_Attributes.strength >= 0 and

Player\_Attributes.crossing >= 0 and Player\_Attributes.short\_passing >= 0 and Player\_Attributes.vision >= 0 and

Player\_Attributes.ball\_control >= 0 and Player\_Attributes.penalties >= 0 and Player\_Attributes.potential >= 0 and

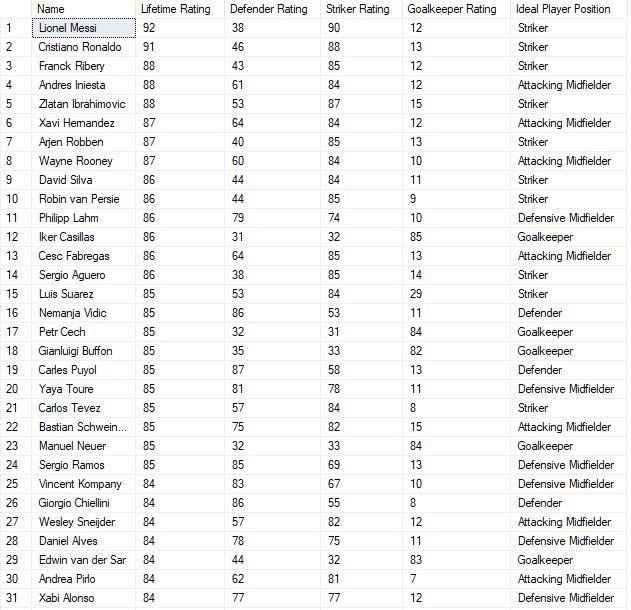
Player\_Attributes.volleys >= 0 and Player\_Attributes.finishing >= 0 and Player\_Attributes.dribbling >= 0 and

Player\_Attributes.gk\_diving >= 0 and Player\_Attributes.gk\_handling >= 0 and Player\_Attributes.gk\_kicking >= 0 and

Player\_Attributes.gk\_positioning >= 0 and Player\_Attributes.gk\_reflexes >= 0

group by Player.player\_name, Player.birthday

order by 'Lifetime Rating' DESC



## Height and header accuracy correlation

use European\_Soccer

select Player.player\_name as Name, avg(Player.height) as Height,avg(Player\_Attributes.heading\_accuracy) AS Header\_Accuracy

from Player\_Attributes FULL OUTER JOIN Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where height > 180

group by Player.player\_name

order by Height ASC, Header\_Accuracy ASC



## Goal summary by season

use European\_Soccer

SELECT Country.name AS country\_name, League.name AS league\_name, season,count(distinct stage) AS number\_of\_stages,

count(distinct HT.team\_long\_name) AS number\_of\_teams, avg(home\_team\_goal) AS avg\_home\_team\_goals,

avg(away\_team\_goal) AS avg\_away\_team\_goals, avg(home\_team\_goal-away\_team\_goal) AS avg\_goal\_dif,

avg(home\_team\_goal+away\_team\_goal) AS avg\_goals, sum(home\_team\_goal+away\_team\_goal) AS total\_goals

from Match

JOIN Country on Country.id = Match.country\_id

JOIN League on League.id = Match.league\_id

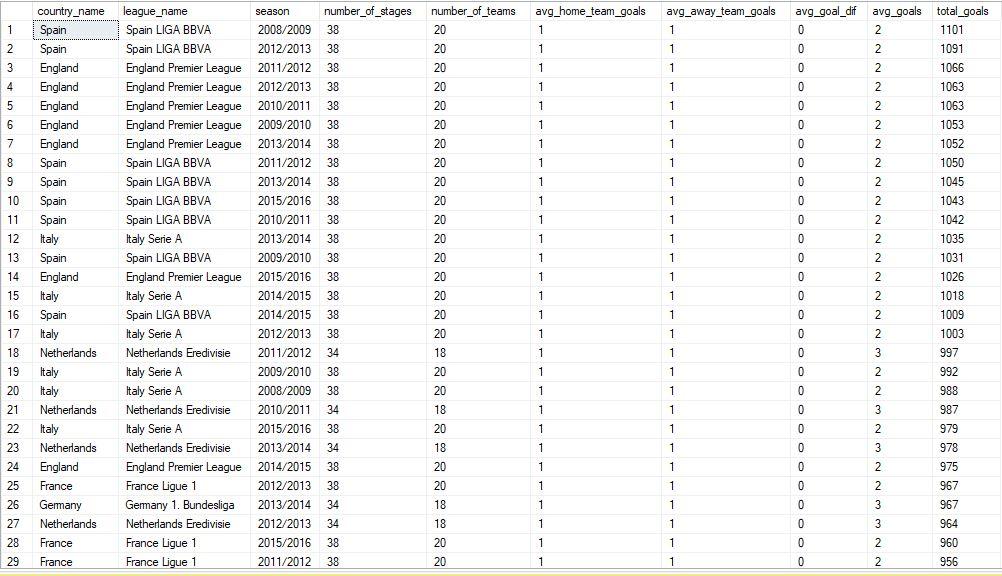
LEFT JOIN Team AS HT on HT.team\_api\_id = Match.home\_team\_api\_id

LEFT JOIN Team AS AT on AT.team\_api\_id = Match.away\_team\_api\_id

GROUP BY Country.name, League.name, season

HAVING count(distinct stage) > 10

ORDER BY total\_goals DESC, Country.name, League.name, season DESC



## Fitness

use European\_Soccer

select top 25 player.player\_name as Name, cast(round(((weight\*703\*2.54\*2.54)/(height\*height)),0) as int) as bmi,

avg(Player\_Attributes.acceleration) as acceleration, avg(Player\_Attributes.sprint\_speed) as sprint\_speed,

avg(Player\_Attributes.agility) as agility, avg(Player\_Attributes.jumping) as jumping , avg(Player\_Attributes.stamina) as stamina,

((avg(Player\_Attributes.acceleration) + avg(Player\_Attributes.sprint\_speed) + avg(Player\_Attributes.agility) +

avg(Player\_Attributes.jumping) + avg(Player\_Attributes.stamina))\*cast(round(((weight\*703\*2.54\*2.54)/(height\*height)),0) as int))/(5\*25) as fitness

from player full outer join Player\_Attributes on Player.player\_api\_id = Player\_Attributes.player\_api\_id

group by player.player\_name, height, weight

order by fitness DESC



## Currently playing footballers

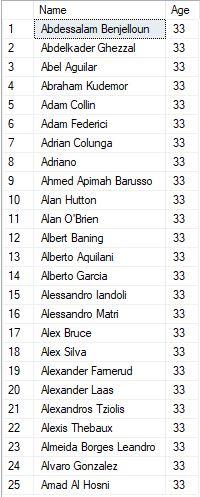
use European\_Soccer

select top 25 player\_name AS Name,Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age

from Player

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) < 34

order by Age desc



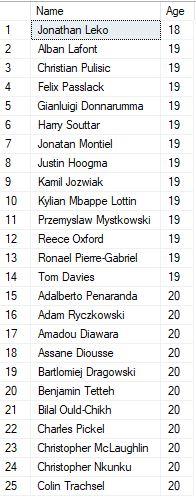
use European\_Soccer

select top 25 player\_name AS Name,Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age

from Player

where Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) < 34

order by Age asc



## Goals and percentage by foot

use European\_Soccer

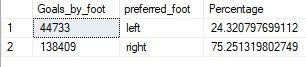
select count(preferred\_foot) as Goals\_by\_foot, preferred\_foot, (count(preferred\_foot)\*100.0/(select count(\*) from Player\_Attributes)) as Percentage

from Player\_Attributes

where preferred\_foot != 'NULL'

group by preferred\_foot

order by preferred\_foot



# DATA MODELS

## Look alike model

use European\_Soccer

select cols.Name, cols.[Lifetime Rating], cols.[Ideal Player Position],

(case

when cols.[Ideal Player Position] = 'Goalkeeper'

then 100 - cols.[Goalkeeper Rating]

else

 case

  when cols.[Ideal Player Position] = 'Defender'

  then 100 - cols.[Defender Rating]

  else

   case

    when cols.[Ideal Player Position] = 'Defensive Midfielder'

    then 100 - ((cols.[Defender Rating] + cols.[Striker Rating])/2)

    else

     case

      when cols.[Ideal Player Position] = 'Striker'

      then 100 - cols.[Striker Rating]

      else

       case

        when cols.[Ideal Player Position] = 'Attacking Midfielder'

        then 100 - ((cols.[Defender Rating] + cols.[Striker Rating])/2)

       end

     end

   end

 end

end) as 'Position Rating'

from (

select Player.player\_name as Name, avg(overall\_rating) as 'Lifetime Rating',

(avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11 as 'Defender Rating',

(avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18 as 'Striker Rating',

(2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8 as 'Goalkeeper Rating',

(case

 when ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8) > ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) and ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8) > ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11)

 then 'Goalkeeper'

 else

  case

   when ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) >= ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) and ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) > ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8)

    then case

     when ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) - ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) <= 20 and ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) - ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) >= 0

      then 'Defensive Midfielder'

     else 'Defender'

    end

  else

   case

    when ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) > ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) and ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) > ((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8)

    then case

      when ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) - ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) <= 25 and ((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) + 2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) + 3\*avg(Player\_Attributes.dribbling) )/18) - ((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) + 2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11) >= 0

       then 'Attacking Midfielder'

      else 'Striker'

      end

   end

  end

end) as 'Ideal Player Position'

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

where Player\_Attributes.heading\_accuracy >= 0 and Player\_Attributes.interceptions >= 0 and Player\_Attributes.sliding\_tackle >= 0 and

Player\_Attributes.standing\_tackle >= 0 and Player\_Attributes.aggression >= 0 and Player\_Attributes.strength >= 0 and

Player\_Attributes.crossing >= 0 and Player\_Attributes.short\_passing >= 0 and Player\_Attributes.vision >= 0 and

Player\_Attributes.ball\_control >= 0 and Player\_Attributes.penalties >= 0 and Player\_Attributes.potential >= 0 and

Player\_Attributes.volleys >= 0 and Player\_Attributes.finishing >= 0 and Player\_Attributes.dribbling >= 0 and

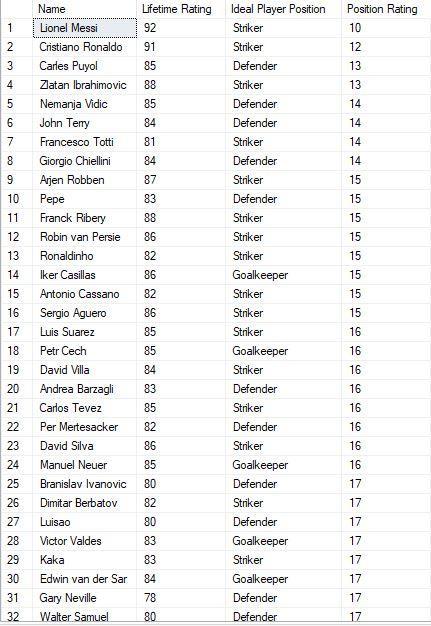
Player\_Attributes.gk\_diving >= 0 and Player\_Attributes.gk\_handling >= 0 and Player\_Attributes.gk\_kicking >= 0 and

Player\_Attributes.gk\_positioning >= 0 and Player\_Attributes.gk\_reflexes >= 0

group by Player.player\_name, Player.birthday

) cols

order by 'Position Rating' ASC



## Percentile overall ratings

Use European\_Soccer

Select \*

from (SELECT DISTINCT

           player\_name,

           /\*STDEV(overall\_rating) OVER (PARTITION BY [player\_name]) as Standard\_Deviation,\*/

           PERCENTILE\_CONT(0.9) WITHIN GROUP (ORDER BY overall\_rating) OVER (PARTITION BY player\_name) as 'Percentile'

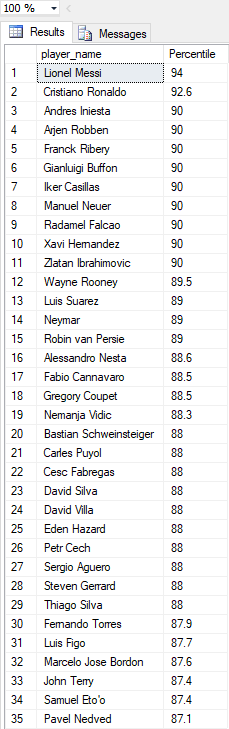
FROM        Player\_Attributes,Player

where Player.player\_api\_id = Player\_Attributes.player\_api\_id

) as col

where col.Percentile > 87

order by col.Percentile Desc



## Percentile strikers

select top 20 Player.player\_name as Name, avg(Player\_Attributes.crossing) as Crossing, avg(Player\_Attributes.short\_passing) as Short\_Passing,

avg(Player\_Attributes.vision) as Vision, avg(Player\_Attributes.ball\_control) as Ball\_Control,

avg(Player\_Attributes.penalties) as Penalties, avg(Player\_Attributes.potential) as Potential,

avg(Player\_Attributes.volleys) as Volleys, avg(Player\_Attributes.finishing) as Finishing, avg(Player\_Attributes.dribbling) as Dribbling,

(avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

 end

end) as Playing\_Status,

PERCENTILE\_CONT(0.9) within group(order by((avg(Player\_Attributes.crossing) + avg(Player\_Attributes.short\_passing) + 2\*avg(Player\_Attributes.vision) + 3\*avg(Player\_Attributes.ball\_control) +

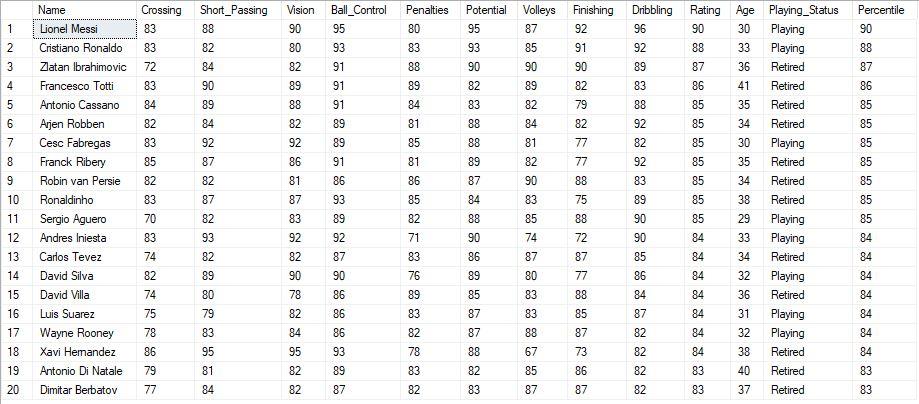
2\*avg(Player\_Attributes.penalties) + 2\*avg(Player\_Attributes.potential) + avg(Player\_Attributes.volleys) + 3\*avg(Player\_Attributes.finishing) +

3\*avg(Player\_Attributes.dribbling) )/18)) OVER(partition by Player.player\_name) as 'Percentile'

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

group by Player.player\_name, Player.birthday

order by Percentile desc



## Percentile goalkeeper

select top 20 Player.player\_name as Name, avg(Player\_Attributes.gk\_diving) as Diving, avg(Player\_Attributes.gk\_handling) as Handling,

avg(Player\_Attributes.gk\_kicking) as Kicking, avg(Player\_Attributes.gk\_positioning) as Positioning, avg(Player\_Attributes.gk\_reflexes) as Reflexes,

(2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 40

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 40

  then 'Playing'

 end

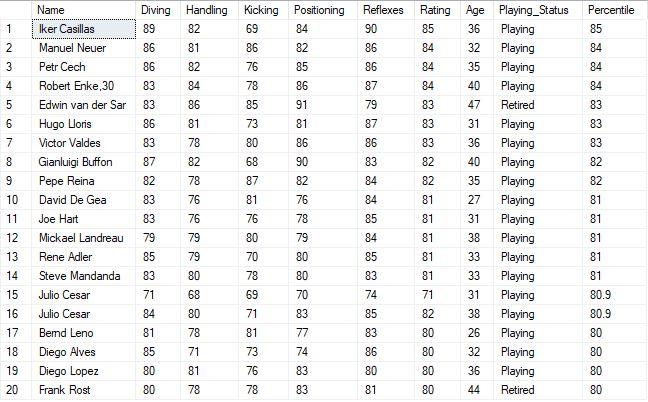
end) as Playing\_Status,

PERCENTILE\_CONT(0.9) within group(order by((2\*avg(Player\_Attributes.gk\_diving) + avg(Player\_Attributes.gk\_handling) + avg(Player\_Attributes.gk\_kicking) + avg(Player\_Attributes.gk\_positioning) + 3\*avg(Player\_Attributes.gk\_reflexes) )/8)) OVER(partition by Player.player\_name) as 'Percentile'

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

group by Player.player\_name, Player.birthday

order by Percentile DESC



## Percentile defenders

select top 20 Player.player\_name as Name, avg(Player\_Attributes.heading\_accuracy) as Heading\_Accuracy, avg(Player\_Attributes.interceptions) as Interceptions,

avg(Player\_Attributes.sliding\_tackle) as Sliding\_Tackles, avg(Player\_Attributes.standing\_tackle) as Standing\_Tackles,

avg(Player\_Attributes.aggression) as Aggression, avg(Player\_Attributes.strength) as Strength,

(avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11 as Rating,

Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) as Age,

(case

when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) > 33

then'Retired'

else

 case

  when Floor(datediff(day,substring(birthday,1,10), GETDATE())/365.25) <= 33

  then 'Playing'

 end

end) as Playing\_Status,

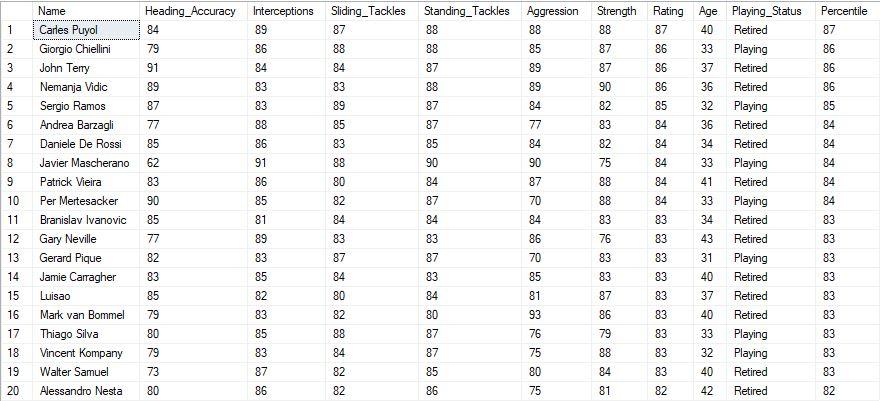
PERCENTILE\_CONT(0.9) within group(order by((avg(Player\_Attributes.heading\_accuracy) + 3\*avg(Player\_Attributes.interceptions) + 2\*avg(Player\_Attributes.sliding\_tackle) +

2\*avg(Player\_Attributes.standing\_tackle) + avg(Player\_Attributes.aggression) + 2\*avg(Player\_Attributes.strength) )/11)) OVER(partition by Player.player\_name) as 'Percentile'

from Player\_Attributes full outer join Player on Player\_Attributes.player\_api\_id = Player.player\_api\_id

group by Player.player\_name, Player.birthday

order by Percentile desc



## Naive Bayes

Use European\_Soccer

Select col.Country\_of\_Match,col.Win\_Loss\_Draw\_Class as 'Home\_win/Away\_win/Draw', count(col.Country\_of\_Match) as 'Counter'

from(select HomeTeam.team\_long\_name as 'Home Team', AwayTeam.team\_long\_name as 'Away Team',

M.home\_team\_goal as 'Home Team Goals', M.away\_team\_goal as 'Away Team Goals',

(case

when M.home\_team\_goal > M.away\_team\_goal

then '1'

else

case

when M.home\_team\_goal < M.away\_team\_goal

then '2'

else

case

when M.home\_team\_goal = M.away\_team\_goal

then '3'

end

end

end) as Win\_Loss\_Draw\_Class,

Country.name as Country\_of\_Match

from (match as M

inner join team as HomeTeam on M.home\_team\_api\_id = HomeTeam.team\_api\_id

inner join team as AwayTeam on M.away\_team\_api\_id = AwayTeam.team\_api\_id

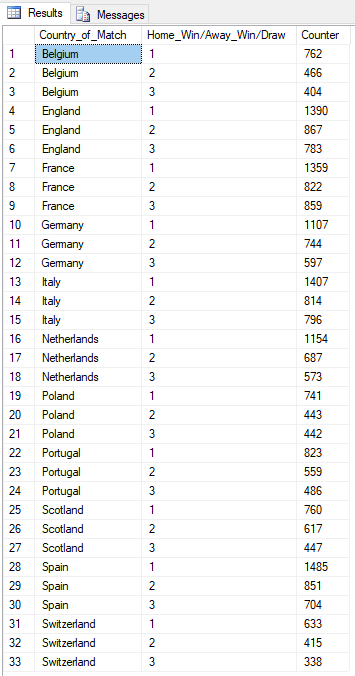
inner join Country on M.country\_id = Country.id

inner join League on M.country\_id = League.id)

)col

group by col.Country\_of\_Match, col.Win\_Loss\_Draw\_Class

order by col.Country\_of\_Match asc, col.Win\_Loss\_Draw\_Class asc



|  |  |
| --- | --- |
| **P(country/Home\_win/Away\_win/Draw)** | **Naive Bayes Values** |
| p(Belgium/home\_win) | 11.028221827511 |
| p(Belgium/away\_win) | 6.744293138609 |
| p(Belgium/draw) | 5.846983751069 |
| p(England/home\_win) | 20.117097559371 |
| p(England/away\_win) | 12.547858693507 |
| p(England/draw) | 11.332149200710 |
| p(France/home\_win) | 19.668442865600 |
| p(France/away\_win) | 11.896585750937 |
| p(France/draw) | 12.432076837050 |
| p(Germany/home\_win) | 16.021314387211 |
| p(Germany/away\_win) | 10.767712650483 |
| p(Germany/draw) | 8.640221038089 |
| p(Italy/home\_win) | 20.363134004341 |
| p(Italy/away\_win) | 11.780803894480 |
| p(Italy/draw) | 11.520294717452 |
| p(Netherlands/home\_win) | 16.701532793895 |
| p(Netherlands /away\_win) | 9.942766923228 |
| p(Netherlands/draw) | 8.292875468719 |
| p(Poland/home\_win) | 10.724294454312 |
| p(Poland/away\_win) | 6.411420301295 |
| p(Poland/draw) | 6.396947569238 |
| p(Portugal/home\_win) | 11.911058482994 |
| p(Portugal/away\_win) | 8.090257219919 |
| p(Portugal/draw) | 7.033747779751 |
| p(Scotland/home\_win) | 10.999276363397 |
| p(Scotland/away\_win) | 8.929675679231 |
| p( Scotland/draw) | 6.469311229524 |
| p(Spain/home\_win) | 21.492007104795 |
| p(Spain/away\_win) | 12.316294980593 |
| p(Spain/draw) | 10.188803368199 |
| p(Switzerland/home\_win) | 9.161239392145 |
| p(Switzerland/away\_win) | 6.006183803697 |
| p(Switzerland/draw) | 4.891783435300 |

# 

# CONCLUSION

The soccer database in SQLite format was migrated to SSMS with an ODBC (64 bit) interface. A 'System DSN' with a linked server created was created in SSMS and connected to the database in SQLite. This data obtained from the dataset was recorded in tables created in SSMS. Further, a 'User DSN' was created to connect the soccer database in SSMS to R/Rattle to perform data cleaning, transformation, and visualization. Moreover, Fascinating outcomes were gathered through predictive analysis by SQL queries in SQL Server.

# REFERENCES

* <https://www.kaggle.com/hugomathien/soccer> - Dataset Source
* <https://www.mssqltips.com/sqlservertip/3087/creating-a-sql-server-linked-server-to-sqlite-to-import-data/>
* <http://www.football-data.co.uk/notes.txt> **-** MetaData about
* <https://stackoverflow.com/questions/40453995/split-one-large-denormalized-table-into-a-normalized-database>
* [https://www.youtube.com/watch?v-y-lbCiAtHsk](https://www.youtube.com/watch?v=y-lbCiAtHsk) (DSN for database on SQL Server).
* <https://en.wikipedia.org/wiki/Data_source_name>
* <https://cran.r-project.org/web/packages/RODBC/vignettes/RODBC.pdf>