Project: Investigating Soccer Database for Seasons 2008/2009 to 2015/2016

Table of Contents

Introduction

This soccer database comes from Kaggle. It contains data for soccer matches, players, and teams from 11 European countries from 2008 to 2016. There are 7 datasets: country, league, Match, Player, Player Attributes, Team, Team Attributes. Questions for Analysis:

- How many matches did each season and each league have?
- Who are the top 3 winning teams for home or away matches? Compare results?
- What does match results convey?
- Which Season had the highest number of winnings by home and away teams?
- Did Celtic team improve its performance throughout the 8 seasons?
- What teams improved the most over the time period?
- What team attributes lead to the most victories?
- Who is the oldest and the youngest players?
- How many players use either preferred right or left foot?
- Who made the most penalties?
- What are the attributes of the best players based on their average overall ratings?

```
# import the modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# color set used for visualization.
plt.style.use('seaborn')
sns_colors = sns.color_palette("Set3", 12)
```

Data Wrangling

The data for the indicators were downloaded separately, therefore I had to first clean each dataset and then merge them together.

```
# Load your data and print out a few lines. Perform operations to
inspect data.
# use shape and info()
df_country= pd.read_csv('Country.csv')
df_country.head()
```

```
       id     name
0       1  Belgium
1    1729  England
2    4769   France
3    7809  Germany
4   10257    Italy
```

```
df_country.shape
```

```
(11, 2)
```

Country dataset has 11 entries representing 11 countries with their id number and name.

```
df_country.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      11 non-null     int64 
 1   name    11 non-null     object
dtypes: int64(1), object(1)
memory usage: 304.0+ bytes
```

There are no null values, but will have to rename columns to "country_id" and "country_name" for consistency and it will be easy to merge with other datasets.

```
df_league= pd.read_csv('League.csv')
df_league.head()
```

```
       id  country_id                      name
0       1           1    Belgium Jupiler League
1    1729        1729    England Premier League
2    4769        4769            France Ligue 1
3    7809        7809     Germany 1. Bundesliga
4   10257       10257             Italy Serie A
```

```
df_league.shape
```

```
(11, 3)
```

There are 11 entries representing 11 leagues with their respective id number, league full name and country.

```
df_league.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          11 non-null     int64 
```

```
 1   country_id  11 non-null     int64
 2   name        11 non-null     object
dtypes: int64(2), object(1)
memory usage: 392.0+ bytes
```

Fortunately, no null values, but will have to rename columns to "league_id", 'country_id' and "league_name" for consistency and later for merging. Although the values in both league id and country id are the same, I will not merge them.

```
df_match= pd.read_csv('Match.csv')
df_match.head()
```

```
   id  country_id  league_id     season  stage                 date  \
0   1           1          1  2008/2009      1  2008-08-17 00:00:00
1   2           1          1  2008/2009      1  2008-08-16 00:00:00
2   3           1          1  2008/2009      1  2008-08-16 00:00:00
3   4           1          1  2008/2009      1  2008-08-17 00:00:00
4   5           1          1  2008/2009      1  2008-08-16 00:00:00

   match_api_id  home_team_api_id  away_team_api_id
home_team_goal  ...  \
0        492473              9987              9993
1  ...
1        492474             10000              9994
0  ...
2        492475              9984              8635
0  ...
3        492476              9991              9998
5  ...
4        492477              7947              9985
1  ...

    SJA   VCH   VCD   VCA   GBH   GBD   GBA   BSH   BSD   BSA
0  4.00  1.65  3.40  4.50  1.78  3.25  4.00  1.73  3.40  4.20
1  3.80  2.00  3.25  3.25  1.85  3.25  3.75  1.91  3.25  3.60
2  2.50  2.35  3.25  2.65  2.50  3.20  2.50  2.30  3.20  2.75
3  7.50  1.45  3.75  6.50  1.50  3.75  5.50  1.44  3.75  6.50
4  1.73  4.50  3.40  1.65  4.50  3.50  1.65  4.75  3.30  1.67

[5 rows x 115 columns]
```

```
df_match.shape
```

```
(25979, 115)
```

Their are 25979 entries representing number of matches across all 11 leagues and 8 seasons from 2008/2009 till 2015/2016.

```
df_match.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25979 entries, 0 to 25978
Columns: 115 entries, id to BSA
dtypes: float64(96), int64(9), object(10)
memory usage: 22.8+ MB
```

```
df_match.isnull()
```

```
          id  country_id  league_id  season  stage   date
match_api_id  \
0      False       False      False   False  False  False
False
1      False       False      False   False  False  False
False
2      False       False      False   False  False  False
False
3      False       False      False   False  False  False
False
4      False       False      False   False  False  False
False
...      ...         ...        ...     ...    ...    ...            ..
.
25974  False       False      False   False  False  False
False
25975  False       False      False   False  False  False
False
25976  False       False      False   False  False  False
False
25977  False       False      False   False  False  False
False
25978  False       False      False   False  False  False
False

       home_team_api_id  away_team_api_id  home_team_goal  ...    SJA
VCH  \
0                 False             False           False  ...  False
False
1                 False             False           False  ...  False
False
2                 False             False           False  ...  False
False
3                 False             False           False  ...  False
False
4                 False             False           False  ...  False
False
...                 ...               ...             ...  ...    ...
...
25974             False             False           False  ...   True
True
25975             False             False           False  ...   True
```

```
True
25976             False             False           False  ...   True
True
25977             False             False           False  ...   True
True
25978             False             False           False  ...   True
True

         VCD    VCA    GBH    GBD    GBA    BSH    BSD    BSA
0      False  False  False  False  False  False  False  False
1      False  False  False  False  False  False  False  False
2      False  False  False  False  False  False  False  False
3      False  False  False  False  False  False  False  False
4      False  False  False  False  False  False  False  False
...      ...    ...    ...    ...    ...    ...    ...    ...
25974   True   True   True   True   True   True   True   True
25975   True   True   True   True   True   True   True   True
25976   True   True   True   True   True   True   True   True
25977   True   True   True   True   True   True   True   True
25978   True   True   True   True   True   True   True   True

[25979 rows x 115 columns]
```

```
df_match.isnull().sum()
```

```
id                0
country_id        0
league_id         0
season            0
stage             0
              ...
GBD           11817
GBA           11817
BSH           11818
BSD           11818
BSA           11818
Length: 115, dtype: int64
```

Only the first 11 columns don't have any null values. there is no use for the other columns, since there is no clear explanation of what they are or entail. I will drop them.

```
df_teams= pd.read_csv('Team.csv')
df_teams.head()
```

```
   id  team_api_id  team_fifa_api_id     team_long_name
team_short_name
0   1         9987             673.0           KRC Genk
GEN
1   2         9993             675.0       Beerschot AC
BAC
2   3        10000           15005.0   SV Zulte-Waregem
```

```
ZUL
3   4         9994            2007.0   Sporting Lokeren
LOK
4   5         9984            1750.0  KSV Cercle Brugge
CEB
```

```
df_teams.count()
```

```
id                  299
team_api_id         299
team_fifa_api_id    288
team_long_name      299
team_short_name     299
dtype: int64
```

```
df_teams.shape
```

```
(299, 5)
```

In the team dataset, there are 299 entries representing 299 teams from 11 countries.

```
df_teams.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                299 non-null    int64
 1   team_api_id       299 non-null    int64
 2   team_fifa_api_id  288 non-null    float64
 3   team_long_name    299 non-null    object
 4   team_short_name   299 non-null    object
dtypes: float64(1), int64(2), object(2)
memory usage: 11.8+ KB
```

Only 'team_fifa_api_id' column has null values. I will drop it, since `team_api_id` column is only common with match dataset.

```
df_teams_attri= pd.read_csv('Team_Attributes.csv')
df_teams_attri.head()
```

```
   id  team_fifa_api_id  team_api_id                 date
buildUpPlaySpeed  \
0   1               434         9930  2010-02-22 00:00:00
60
1   2               434         9930  2014-09-19 00:00:00
52
2   3               434         9930  2015-09-10 00:00:00
47
3   4                77         8485  2010-02-22 00:00:00
70
```

```
4   5                 77         8485  2011-02-22 00:00:00
47

  buildUpPlaySpeedClass  buildUpPlayDribbling
buildUpPlayDribblingClass  \
0              Balanced                   NaN
Little
1              Balanced                  48.0
Normal
2              Balanced                  41.0
Normal
3                  Fast                   NaN
Little
4              Balanced                   NaN
Little

   buildUpPlayPassing buildUpPlayPassingClass  ...
chanceCreationShooting  \
0                  50                   Mixed  ...
55
1                  56                   Mixed  ...
64
2                  54                   Mixed  ...
64
3                  70                    Long  ...
70
4                  52                   Mixed  ...
52

   chanceCreationShootingClass chanceCreationPositioningClass  \
0                       Normal                      Organised
1                       Normal                      Organised
2                       Normal                      Organised
3                         Lots                      Organised
4                       Normal                      Organised

   defencePressure defencePressureClass  defenceAggression  \
0               50               Medium                 55
1               47               Medium                 44
2               47               Medium                 44
3               60               Medium                 70
4               47               Medium                 47

  defenceAggressionClass defenceTeamWidth  defenceTeamWidthClass  \
0                  Press               45                 Normal
1                  Press               54                 Normal
2                  Press               54                 Normal
3                 Double               70                   Wide
4                  Press               52                 Normal
```

```
  defenceDefenderLineClass
0                    Cover
1                    Cover
2                    Cover
3                    Cover
4                    Cover

[5 rows x 25 columns]
```

```
df_teams_attri.shape
```

```
(1458, 25)
```

```
df_teams_attri.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458 entries, 0 to 1457
Data columns (total 25 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              1458 non-null   int64  
 1   team_fifa_api_id                1458 non-null   int64  
 2   team_api_id                     1458 non-null   int64  
 3   date                            1458 non-null   object 
 4   buildUpPlaySpeed                1458 non-null   int64  
 5   buildUpPlaySpeedClass           1458 non-null   object 
 6   buildUpPlayDribbling            489 non-null    float64
 7   buildUpPlayDribblingClass       1458 non-null   object 
 8   buildUpPlayPassing              1458 non-null   int64  
 9   buildUpPlayPassingClass         1458 non-null   object 
 10  buildUpPlayPositioningClass     1458 non-null   object 
 11  chanceCreationPassing           1458 non-null   int64  
 12  chanceCreationPassingClass      1458 non-null   object 
 13  chanceCreationCrossing          1458 non-null   int64  
 14  chanceCreationCrossingClass     1458 non-null   object 
 15  chanceCreationShooting          1458 non-null   int64  
 16  chanceCreationShootingClass     1458 non-null   object 
 17  chanceCreationPositioningClass  1458 non-null   object 
 18  defencePressure                 1458 non-null   int64  
 19  defencePressureClass            1458 non-null   object 
 20  defenceAggression               1458 non-null   int64  
 21  defenceAggressionClass          1458 non-null   object 
 22  defenceTeamWidth                1458 non-null   int64  
 23  defenceTeamWidthClass           1458 non-null   object 
 24  defenceDefenderLineClass        1458 non-null   object 
dtypes: float64(1), int64(11), object(13)
memory usage: 284.9+ KB
```

In the teams attributes dataset, there are 1458 entries, only `buildUpPlayDribbling` has null values, so it can be droped. `date` column needs to be changed to datetime.

```
df_players= pd.read_csv('Player.csv')
df_players.head()
```

```
   id  player_api_id         player_name  player_fifa_api_id  \
0   1         505942  Aaron Appindangoye              218353   
1   2         155782     Aaron Cresswell              189615   
2   3         162549         Aaron Doran              186170   
3   4          30572       Aaron Galindo              140161   
4   5          23780        Aaron Hughes               17725   

              birthday  height  weight  
0  1992-02-29 00:00:00  182.88     187  
1  1989-12-15 00:00:00  170.18     146  
2  1991-05-13 00:00:00  170.18     163  
3  1982-05-08 00:00:00  182.88     198  
4  1979-11-08 00:00:00  182.88     154  
```

```
df_players.shape
```

```
(11060, 7)
```

```
df_players.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11060 entries, 0 to 11059
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   id                  11060 non-null  int64  
 1   player_api_id       11060 non-null  int64  
 2   player_name         11060 non-null  object 
 3   player_fifa_api_id  11060 non-null  int64  
 4   birthday            11060 non-null  object 
 5   height              11060 non-null  float64
 6   weight              11060 non-null  int64  
dtypes: float64(1), int64(4), object(2)
memory usage: 605.0+ KB
```

11060 players and no null values.

```
df_players_attri= pd.read_csv('Player_Attributes.csv')
df_players_attri.head(7)
```

```
   id  player_fifa_api_id  player_api_id                 date
overall_rating  \
0   1              218353         505942  2016-02-18 00:00:00
67.0   
1   2              218353         505942  2015-11-19 00:00:00
67.0   
2   3              218353         505942  2015-09-21 00:00:00
62.0   
3   4              218353         505942  2015-03-20 00:00:00
```

```
61.0
4   5              218353         505942  2007-02-22 00:00:00
61.0
5   6              189615         155782  2016-04-21 00:00:00
74.0
6   7              189615         155782  2016-04-07 00:00:00
74.0

   potential preferred_foot attacking_work_rate defensive_work_rate
crossing  \
0       71.0          right              medium              medium
49.0
1       71.0          right              medium              medium
49.0
2       66.0          right              medium              medium
49.0
3       65.0          right              medium              medium
48.0
4       65.0          right              medium              medium
48.0
5       76.0           left                high              medium
80.0
6       76.0           left                high              medium
80.0

   ...  vision  penalties  marking  standing_tackle  sliding_tackle  \
0  ...    54.0       48.0     65.0             69.0            69.0
1  ...    54.0       48.0     65.0             69.0            69.0
2  ...    54.0       48.0     65.0             66.0            69.0
3  ...    53.0       47.0     62.0             63.0            66.0
4  ...    53.0       47.0     62.0             63.0            66.0
5  ...    66.0       59.0     76.0             75.0            78.0
6  ...    66.0       59.0     76.0             75.0            78.0

   gk_diving  gk_handling  gk_kicking  gk_positioning  gk_reflexes
0        6.0         11.0        10.0             8.0          8.0
1        6.0         11.0        10.0             8.0          8.0
2        6.0         11.0        10.0             8.0          8.0
3        5.0         10.0         9.0             7.0          7.0
4        5.0         10.0         9.0             7.0          7.0
5       14.0          7.0         9.0             9.0         12.0
6       14.0          7.0         9.0             9.0         12.0

[7 rows x 42 columns]
```

```
df_players_attri.shape
```

```
(183978, 42)
```

```
df_players_attri.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183978 entries, 0 to 183977
Data columns (total 42 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   id                   183978 non-null  int64  
 1   player_fifa_api_id   183978 non-null  int64  
 2   player_api_id        183978 non-null  int64  
 3   date                 183978 non-null  object 
 4   overall_rating       183142 non-null  float64
 5   potential            183142 non-null  float64
 6   preferred_foot       183142 non-null  object 
 7   attacking_work_rate  180748 non-null  object 
 8   defensive_work_rate  183142 non-null  object 
 9   crossing             183142 non-null  float64
 10  finishing            183142 non-null  float64
 11  heading_accuracy     183142 non-null  float64
 12  short_passing        183142 non-null  float64
 13  volleys              181265 non-null  float64
 14  dribbling            183142 non-null  float64
 15  curve                181265 non-null  float64
 16  free_kick_accuracy   183142 non-null  float64
 17  long_passing         183142 non-null  float64
 18  ball_control         183142 non-null  float64
 19  acceleration         183142 non-null  float64
 20  sprint_speed         183142 non-null  float64
 21  agility              181265 non-null  float64
 22  reactions            183142 non-null  float64
 23  balance              181265 non-null  float64
 24  shot_power           183142 non-null  float64
 25  jumping              181265 non-null  float64
 26  stamina              183142 non-null  float64
 27  strength             183142 non-null  float64
 28  long_shots           183142 non-null  float64
 29  aggression           183142 non-null  float64
 30  interceptions        183142 non-null  float64
 31  positioning          183142 non-null  float64
 32  vision               181265 non-null  float64
 33  penalties            183142 non-null  float64
 34  marking              183142 non-null  float64
 35  standing_tackle      183142 non-null  float64
 36  sliding_tackle       181265 non-null  float64
 37  gk_diving            183142 non-null  float64
 38  gk_handling          183142 non-null  float64
 39  gk_kicking           183142 non-null  float64
 40  gk_positioning       183142 non-null  float64
 41  gk_reflexes          183142 non-null  float64
dtypes: float64(35), int64(3), object(4)
memory usage: 59.0+ MB
```

With the exception of the first 4 columns, the rest have null values. I noticed that there is multiple entries for each single player in different dates form 2007 till 2016. This explains the 183978 entries.

General Properties Concluded:

11 countries 11 league champions 25979 matches 299 teams 21 team sttributes 11060 players 38 players attributes

Data Cleaning

Changes in `df_country`

```
# Rename columns in ```df_country``` for consistency.
df_country.rename(columns={'id':'country_id',
'name':'country_name'},inplace=True)
df_country
```

```
    country_id country_name
0            1      Belgium
1         1729      England
2         4769       France
3         7809      Germany
4        10257        Italy
5        13274  Netherlands
6        15722       Poland
7        17642     Portugal
8        19694     Scotland
9        21518        Spain
10       24558  Switzerland
```

```
# Drop country_id column and rename the rest of the columns in
df_league.
df_league.rename(columns={'id':'league_id',
'name':'league_name'},inplace=True)
df_league.drop('country_id',axis= 1, inplace= True)
df_league
```

```
    league_id                league_name
0           1     Belgium Jupiler League
1        1729     England Premier League
2        4769             France Ligue 1
3        7809      Germany 1. Bundesliga
4       10257              Italy Serie A
5       13274     Netherlands Eredivisie
6       15722         Poland Ekstraklasa
7       17642   Portugal Liga ZON Sagres
8       19694    Scotland Premier League
9       21518            Spain LIGA BBVA
10      24558   Switzerland Super League
```

Changes in `df_teams`

```
# Check for any duplicated team name
duplicated= df_teams[df_teams.duplicated('team_long_name')]
duplicated
```

```
        id  team_api_id  team_fifa_api_id        team_long_name  \
24    2510       274581          111560.0  Royal Excel Mouscron
183  31445         8020          111429.0         Polonia Bytom
199  32409         8024             301.0           Widzew Łódź

    team_short_name
24              MOP
183             GOR
199             WID
```

```
# Drop the duplicated names.
df_teams.drop([df_teams.index[24], df_teams.index[183],
df_teams.index[199]], axis=0, inplace=True)
df_teams.count()
```

```
id                  296
team_api_id         296
team_fifa_api_id    285
team_long_name      296
team_short_name     296
dtype: int64
```

```
# Drop unneeded columns.
df_teams.drop(['id', 'team_fifa_api_id', 'team_short_name'],axis= 1,
inplace= True)
df_teams.head()
```

```
   team_api_id     team_long_name
0         9987           KRC Genk
1         9993       Beerschot AC
2        10000   SV Zulte-Waregem
3         9994   Sporting Lokeren
4         9984  KSV Cercle Brugge
```

```
df_teams.rename(columns={'team_api_id':'team_id'},inplace=True)

# Save changes.
df_teams.to_csv('teams_clean.csv', index=False)
```

Changes in `df_teams_attri`

```
# Add team names column.
team_attribute= df_teams_attri.merge(df_teams, left_on='team_api_id',
right_on='team_id', how='inner')

# Drop null values.
team_attribute.dropna(axis=1, inplace=True)
```

```
# Drop unnecessary columns
team_attribute.drop(['id','team_fifa_api_id', 'team_api_id'], axis=1,
inplace=True)

# Rearrange columns.
team_attribute= team_attribute[['team_id','team_long_name','date',
'buildUpPlaySpeed', 'buildUpPlaySpeedClass',
       'buildUpPlayDribblingClass', 'buildUpPlayPassing',
       'buildUpPlayPassingClass', 'buildUpPlayPositioningClass',
       'chanceCreationPassing', 'chanceCreationPassingClass',
       'chanceCreationCrossing', 'chanceCreationCrossingClass',
       'chanceCreationShooting', 'chanceCreationShootingClass',
       'chanceCreationPositioningClass', 'defencePressure',
       'defencePressureClass', 'defenceAggression',
'defenceAggressionClass',
       'defenceTeamWidth', 'defenceTeamWidthClass',
'defenceDefenderLineClass']]

# Change date column to datetime.
team_attribute['date']= pd.to_datetime(team_attribute['date'])

#drop non nonnumeric columns and team id.
team_attribute.drop(['team_id','buildUpPlaySpeedClass','buildUpPlayPas
singClass','buildUpPlayDribblingClass','buildUpPlayPositioningClass',

'chanceCreationPassingClass','chanceCreationCrossingClass','chanceCrea
tionShootingClass',

'chanceCreationPositioningClass','defencePressureClass','defenceAggres
sionClass',

'defenceTeamWidthClass','defenceDefenderLineClass'],axis= 1, inplace=
True)

# Save changes
team_attribute.to_csv('team_attribute_clean.csv', index=False)
```

Changes in `df_match`

```
# I am only interested in the first 11 columns and the reset is
dropped
df_match.drop(df_match.loc[:, 'home_player_X1':'BSA'], axis=1,
inplace=True)

# Add country_name from df_country by using merge() and create new
dataframe match.
match= pd.merge(df_match, df_country, on='country_id', how='inner')

# Add league_name from df_league by using merge().
match= match.merge(df_league, on='league_id', how='inner')

# Add team names as home_team and away_team using merge with df_teams.
```

```
# Create and add home_team.
match= match.merge(df_teams, left_on='home_team_api_id',
right_on='team_id',how='inner')
match.rename(columns={'team_long_name':'home_team'},inplace=True)

# Create and add away_team.
match= match.merge(df_teams, left_on='away_team_api_id',
right_on='team_id',how='left')
match.rename(columns={'team_long_name':'away_team'},inplace=True)

match.head()
```

```
   id  country_id  league_id     season  stage                 date  \
0   1           1          1  2008/2009      1  2008-08-17 00:00:00
1  29           1          1  2008/2009     12  2008-11-15 00:00:00
2  47           1          1  2008/2009     14  2008-11-29 00:00:00
3  65           1          1  2008/2009     16  2008-12-13 00:00:00
4  94           1          1  2008/2009     19  2009-01-24 00:00:00

   match_api_id  home_team_api_id  away_team_api_id  home_team_goal  \
0        492473              9987              9993               1
1        492583              9987              9999               1
2        492651              9987              9984               3
3        492713              9987              9986               1
4        492805              9987              9998               2

   away_team_goal country_name             league_name  team_id_x
home_team  \
0               1      Belgium  Belgium Jupiler League       9987  KRC
Genk
1               1      Belgium  Belgium Jupiler League       9987  KRC
Genk
2               2      Belgium  Belgium Jupiler League       9987  KRC
Genk
3               0      Belgium  Belgium Jupiler League       9987  KRC
Genk
4               0      Belgium  Belgium Jupiler League       9987  KRC
Genk

   team_id_y           away_team
0     9993.0        Beerschot AC
1     9999.0       KSV Roeselare
2     9984.0   KSV Cercle Brugge
3     9986.0  Sporting Charleroi
4     9998.0           RAEC Mons
```

```
# Drop unnecessary columns in match
match.drop(['id','country_id', 'league_id',
            'home_team_api_id','away_team_api_id',
            'team_id_x','team_id_y'], axis=1, inplace=True)
```

```python
# Rearrange the remaining columns.
match= match[['country_name','league_name','season', 'stage', 'date',
'match_api_id',

'home_team','away_team','home_team_goal','away_team_goal']]

match.head()
```

```
  country_name             league_name     season  stage
date  \
0      Belgium  Belgium Jupiler League  2008/2009      1  2008-08-17
00:00:00
1      Belgium  Belgium Jupiler League  2008/2009     12  2008-11-15
00:00:00
2      Belgium  Belgium Jupiler League  2008/2009     14  2008-11-29
00:00:00
3      Belgium  Belgium Jupiler League  2008/2009     16  2008-12-13
00:00:00
4      Belgium  Belgium Jupiler League  2008/2009     19  2009-01-24
00:00:00

   match_api_id home_team           away_team  home_team_goal
away_team_goal
0        492473  KRC Genk        Beerschot AC               1
1
1        492583  KRC Genk       KSV Roeselare               1
1
2        492651  KRC Genk   KSV Cercle Brugge               3
2
3        492713  KRC Genk  Sporting Charleroi               1
0
4        492805  KRC Genk           RAEC Mons               2
0
```

```python
# Creat column for match result and add it to match dataframe.
def result(match):
    if match['home_team_goal'] > match['away_team_goal']:
        return match['home_team']
    elif match['home_team_goal'] < match['away_team_goal']:
        return match['away_team']
    elif match['home_team_goal'] == match['away_team_goal']:
        return 'Tie'

match['match_result']= match.apply(result, axis=1)

match.head()
```

```
  country_name             league_name     season  stage
date  \
0      Belgium  Belgium Jupiler League  2008/2009      1  2008-08-17
00:00:00
```

```
1      Belgium  Belgium Jupiler League  2008/2009     12  2008-11-15
00:00:00
2      Belgium  Belgium Jupiler League  2008/2009     14  2008-11-29
00:00:00
3      Belgium  Belgium Jupiler League  2008/2009     16  2008-12-13
00:00:00
4      Belgium  Belgium Jupiler League  2008/2009     19  2009-01-24
00:00:00

   match_api_id home_team           away_team  home_team_goal
away_team_goal  \
0        492473  KRC Genk        Beerschot AC               1
1
1        492583  KRC Genk       KSV Roeselare               1
1
2        492651  KRC Genk   KSV Cercle Brugge               3
2
3        492713  KRC Genk  Sporting Charleroi               1
0
4        492805  KRC Genk           RAEC Mons               2
0

  match_result
0          Tie
1          Tie
2     KRC Genk
3     KRC Genk
4     KRC Genk
```

```
#check for null values.
match.isnull().sum()
```

```
country_name        0
league_name         0
season              0
stage               0
date                0
match_api_id        0
home_team           0
away_team         187
home_team_goal      0
away_team_goal      0
match_result       38
dtype: int64
```

```
match.dropna(axis=0, inplace=True)
```

```
match.isnull().sum()
```

```
country_name      0
league_name       0
```

```
season            0
stage             0
date              0
match_api_id      0
home_team         0
away_team         0
home_team_goal    0
away_team_goal    0
match_result      0
dtype: int64
```

```
#check for duplicates.
match.duplicated().sum()
```

```
0
```

```
match.to_csv('match_clean.csv', index=False)
```

Changes in `df_players`

```
#Remove id and player_fifa_api_id.
df_players.drop(['id','player_fifa_api_id'], axis=1, inplace=True)

#Convert birthday column to datetime.
df_players['birthday']= pd.to_datetime(df_players['birthday'])

# Rename column player_api_id to player_id
df_players.rename(columns={'player_api_id':'player_id'},inplace=True)

df_players.head()
```

```
   player_id         player_name   birthday  height  weight
0     505942  Aaron Appindangoye 1992-02-29  182.88     187
1     155782     Aaron Cresswell 1989-12-15  170.18     146
2     162549         Aaron Doran 1991-05-13  170.18     163
3      30572       Aaron Galindo 1982-05-08  182.88     198
4      23780        Aaron Hughes 1979-11-08  182.88     154
```

```
df_players.dtypes
```

```
player_id                int64
player_name             object
birthday        datetime64[ns]
height                 float64
weight                   int64
dtype: object
```

```
df_players.isnull().sum()
```

```
player_id      0
player_name    0
birthday       0
height         0
```

```
weight         0
dtype: int64
```

```
df_players.duplicated().sum()
```

```
0
```

Changes in `df_players_attri`

```
# Drop all null values.
df_players_attri.dropna(inplace=True)

# Drop id and player_fifa_api_id
df_players_attri.drop(['id','player_fifa_api_id'], axis=1,
inplace=True)

# Rename column player_api_id to player_id
df_players_attri.rename(columns={'player_api_id':'player_id'},inplace=
True)

#Convert date column to datetime.
df_players_attri['date']= pd.to_datetime(df_players_attri['date'])

# Merge df_players and df_players_attri
players= df_players_attri.merge(df_players, left_on='player_id',
right_on='player_id',how='inner')

# Rearrange columns.
players= players[['player_id','player_name','birthday', 'height',
'weight','date','overall_rating','potential',

'preferred_foot','attacking_work_rate','defensive_work_rate','crossing
','finishing','free_kick_accuracy',

'long_passing','ball_control','acceleration','sprint_speed','agility',
'reactions','balance','shot_power',

'jumping','stamina','strength','long_shots','aggression','interception
s','positioning','vision','penalties',

'marking','standing_tackle','sliding_tackle','gk_diving','gk_handling'
,'gk_kicking','gk_positioning','gk_reflexes']]
```

```
players.head()
```

```
   player_id         player_name   birthday  height  weight       date
\
0     505942  Aaron Appindangoye 1992-02-29  182.88     187 2016-02-18

1     505942  Aaron Appindangoye 1992-02-29  182.88     187 2015-11-19

2     505942  Aaron Appindangoye 1992-02-29  182.88     187 2015-09-21
```

```
3     505942  Aaron Appindangoye 1992-02-29  182.88     187 2015-03-20

4     505942  Aaron Appindangoye 1992-02-29  182.88     187 2007-02-22


   overall_rating  potential preferred_foot attacking_work_rate  ...
vision  \
0            67.0       71.0          right              medium  ...
54.0
1            67.0       71.0          right              medium  ...
54.0
2            62.0       66.0          right              medium  ...
54.0
3            61.0       65.0          right              medium  ...
53.0
4            61.0       65.0          right              medium  ...
53.0

   penalties  marking  standing_tackle  sliding_tackle  gk_diving  \
0       48.0     65.0             69.0            69.0        6.0
1       48.0     65.0             69.0            69.0        6.0
2       48.0     65.0             66.0            69.0        6.0
3       47.0     62.0             63.0            66.0        5.0
4       47.0     62.0             63.0            66.0        5.0

   gk_handling  gk_kicking  gk_positioning  gk_reflexes
0         11.0        10.0             8.0          8.0
1         11.0        10.0             8.0          8.0
2         11.0        10.0             8.0          8.0
3         10.0         9.0             7.0          7.0
4         10.0         9.0             7.0          7.0

[5 rows x 39 columns]
```

```
# Save changes.
players.to_csv('players_clean.csv', index=False)
```

Exploratory Data Analysis

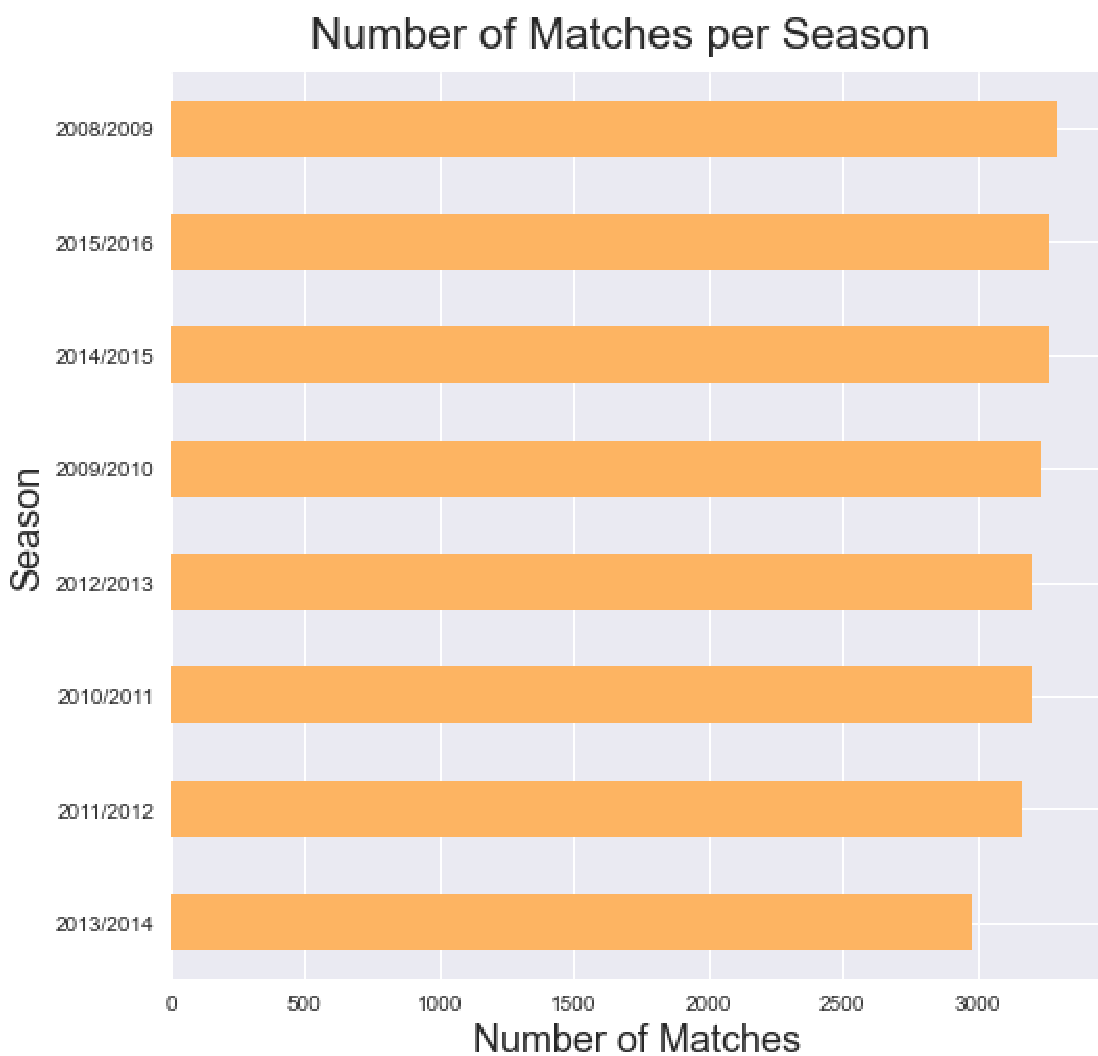
How many matches did each season have?

```
df=match['season'].value_counts().sort_values(ascending=True)
df
```

```
2013/2014    2974
2011/2012    3162
2010/2011    3202
2012/2013    3202
```

```
2009/2010    3230
2014/2015    3265
2015/2016    3266
2008/2009    3296
Name: season, dtype: int64
```

```
#Plot results.
colors= sns_colors[5]
df.plot(kind='barh', color=colors, figsize=(8,8));
plt.figtext(.5,.9,'Number of Matches per Season', fontsize=21,
ha='center');
plt.xlabel('Number of Matches', fontsize=18);
plt.ylabel('Season', fontsize=18);
```



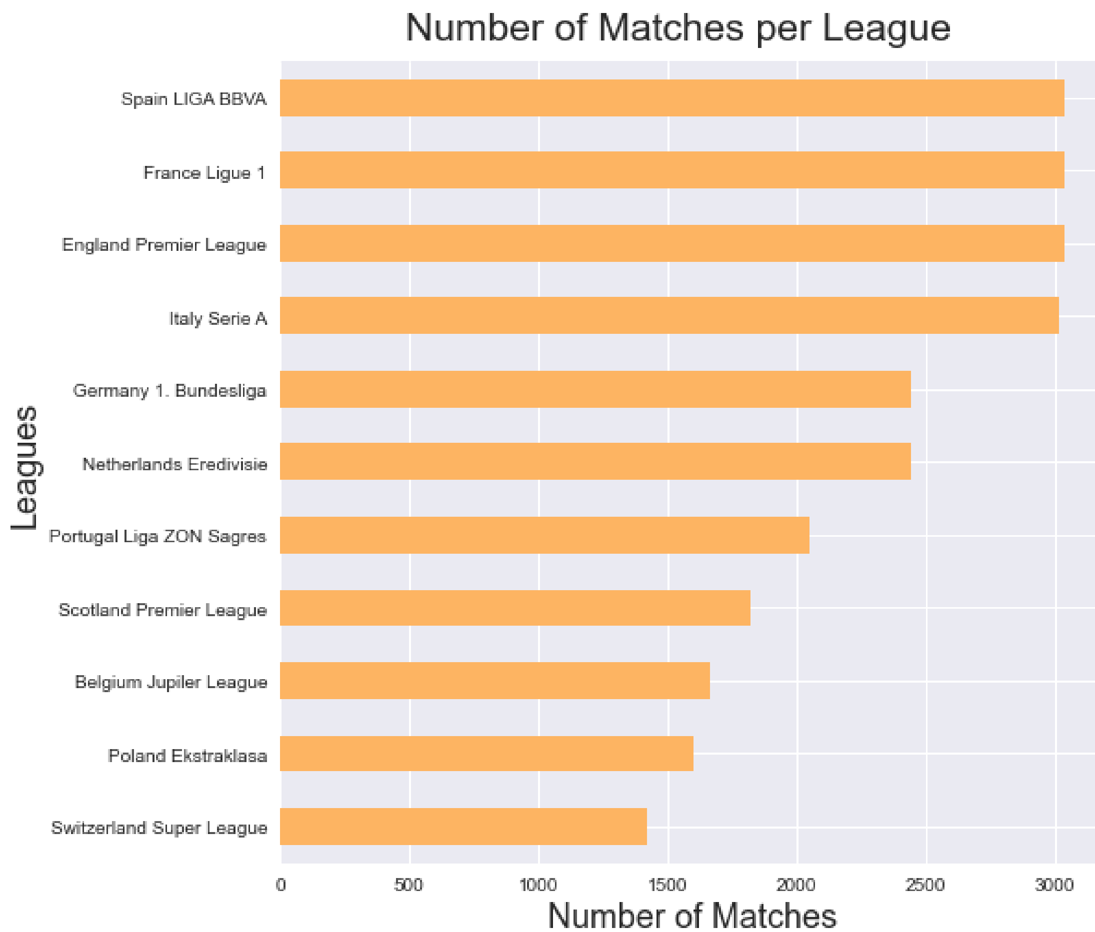
Throughout the 8 seasons, 2008/2009 had the most number of matches while 2013/2014 had the lowest number of matches.

How many matches did each league have?

```
df1= match['league_name'].value_counts().sort_values(ascending=True)
df1
```

```
Switzerland Super League    1422
Poland Ekstraklasa          1598
Belgium Jupiler League      1668
Scotland Premier League     1824
Portugal Liga ZON Sagres    2052
Netherlands Eredivisie      2448
Germany 1. Bundesliga       2448
Italy Serie A               3017
England Premier League      3040
France Ligue 1              3040
Spain LIGA BBVA             3040
Name: league_name, dtype: int64
```

```
#Plot results.
colors= sns_colors[5]
df1.plot(kind='barh', color=colors, figsize=(8,8))
plt.figtext(.5,.9,'Number of Matches per League', fontsize=21,
ha='center');
plt.xlabel('Number of Matches', fontsize=18);
plt.ylabel('Leagues', fontsize=18);
```



England Premier League, Spain LIGA BBVA, France Ligue 1 have the highest number of matches throughout the 8 seasons.

Who are the top 3 winning teams for home or away matches? Compare results?

```
# Top 3 winning home teams.
win_home= match.query('home_team == match_result')
home=
win_home['match_result'].value_counts().sort_values(ascending=False)
[:3]
print(home)
```

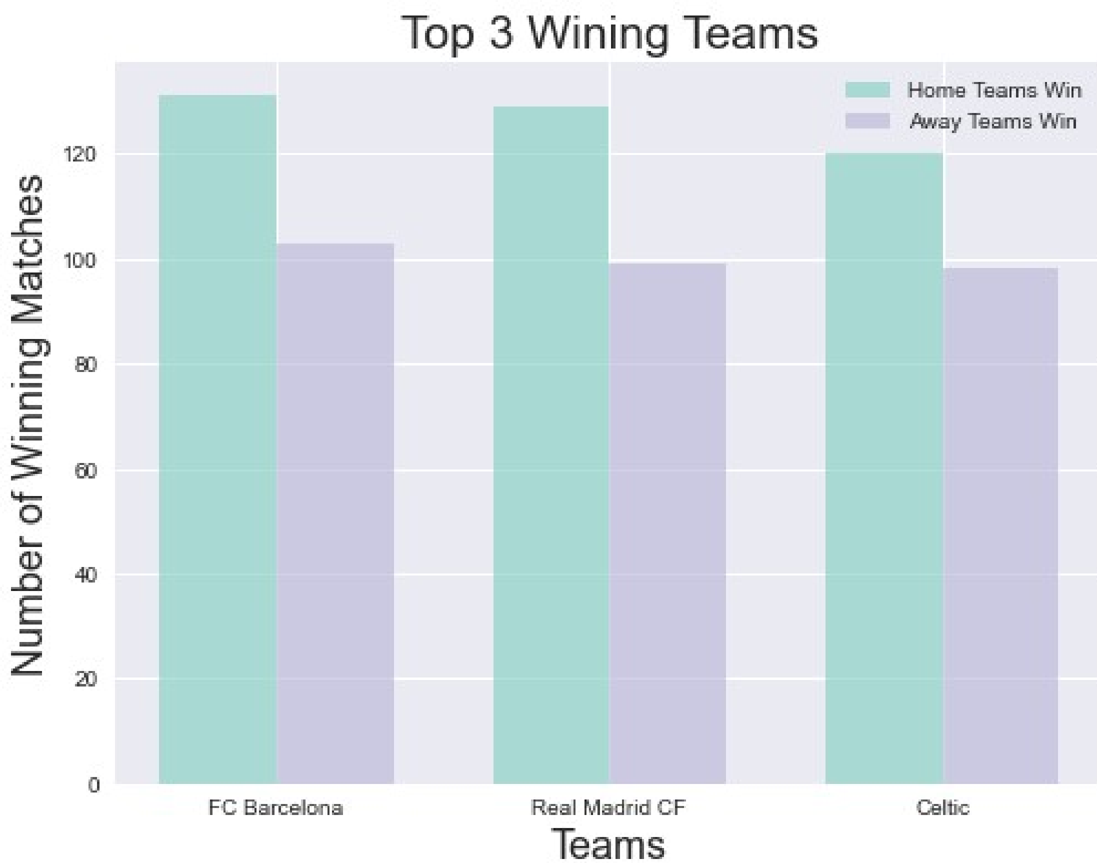
```
FC Barcelona      131
Real Madrid CF    129
Celtic            120
Name: match_result, dtype: int64
```

```
# Top 3 winning away teams.
win_away= match.query('away_team == match_result')
away=
win_away['match_result'].value_counts().sort_values(ascending=False)
[:3]
print(away)
```

```
FC Barcelona      103
Real Madrid CF     99
Celtic             98
Name: match_result, dtype: int64
```

`FC Barcelona`, `Real Madrid CF`,and `Celtic` are top 3 winning teams both as a home team and away team with a difference in winning results.So, lets compare the results.

```
# Compare the results.
ind = np.arange(len(home))
width = 0.35
fig = plt.figure(figsize = (8, 6));

home_bars = plt.bar(ind, home, width, color=sns_colors[0], alpha=.7,
label='Home Teams Win');
away_bars = plt.bar(ind+width, away, width, color=sns_colors[2],
alpha=.7, label='Away Teams Win');

locations = ind + width / 2
labels = ['FC Barcelona', 'Real Madrid CF', 'Celtic']
plt.xticks(locations, labels);

plt.ylabel('Number of Winning Matches', fontsize=18);
plt.xlabel('Teams', fontsize=18);
plt.title('Top 3 Wining Teams', fontsize=21);

plt.legend();
```



Top teams who play in their homeland have more winnings than playing away from their homeland.

What does match results convey?

To answer this, we will have to calculate total wins by home teams and away teams, and also total number of ties. Then, we'll get their percentages and plot them for visual comparison.

```
# Total wins by home teams.
total_home_wins= win_home['match_result'].count()

# Total wins by away teams.
total_away_wins= win_away['match_result'].count()

# Total ties.
tie= match.query('match_result == "Tie"')
total_tie= tie['match_result'].count()

# Total matches
total_matches= match['match_result'].count()
```

```python
# Percentage of home teams winnings.
home_proportion= (total_home_wins/total_matches)*100
home_proportion
```

```
45.872563190998946
```

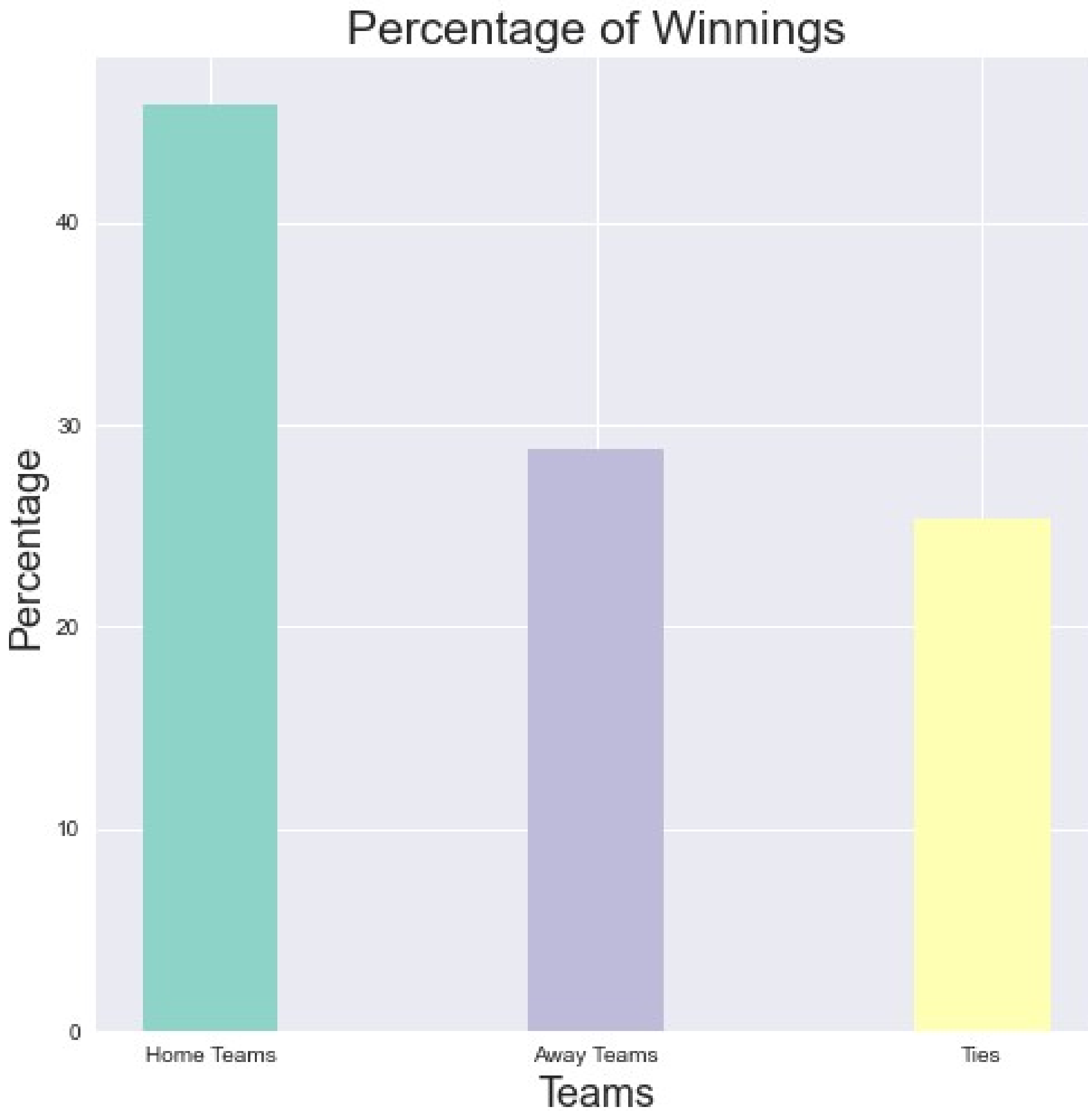
```python
# Percentage of away teams winnings.
away_proportion= (total_away_wins/total_matches)*100
away_proportion
```

```
28.800250029300305
```

```python
# Percentage of total ties.
tie_proportion= (total_tie/total_matches)*100
tie_proportion
```

```
25.327186779700746
```

```python
# Plot results using bar chart.
data = {'Home Teams':home_proportion, 'Away Teams':away_proportion,
'Ties':tie_proportion}
keys = data.keys()
values = data.values()

fig = plt.figure(figsize = (8, 8));

plt.bar(keys, values, color =[sns_colors[0], sns_colors[2],
sns_colors[1]],
        width = 0.35);

plt.xlabel("Teams", fontsize=18);
plt.ylabel("Percentage", fontsize=18);
plt.title("Percentage of Winnings", fontsize=21);
```



Home teams have 45.87% chance of winning a match compared to away teams of 28.74% chance. There are 25.38% chance for ties.

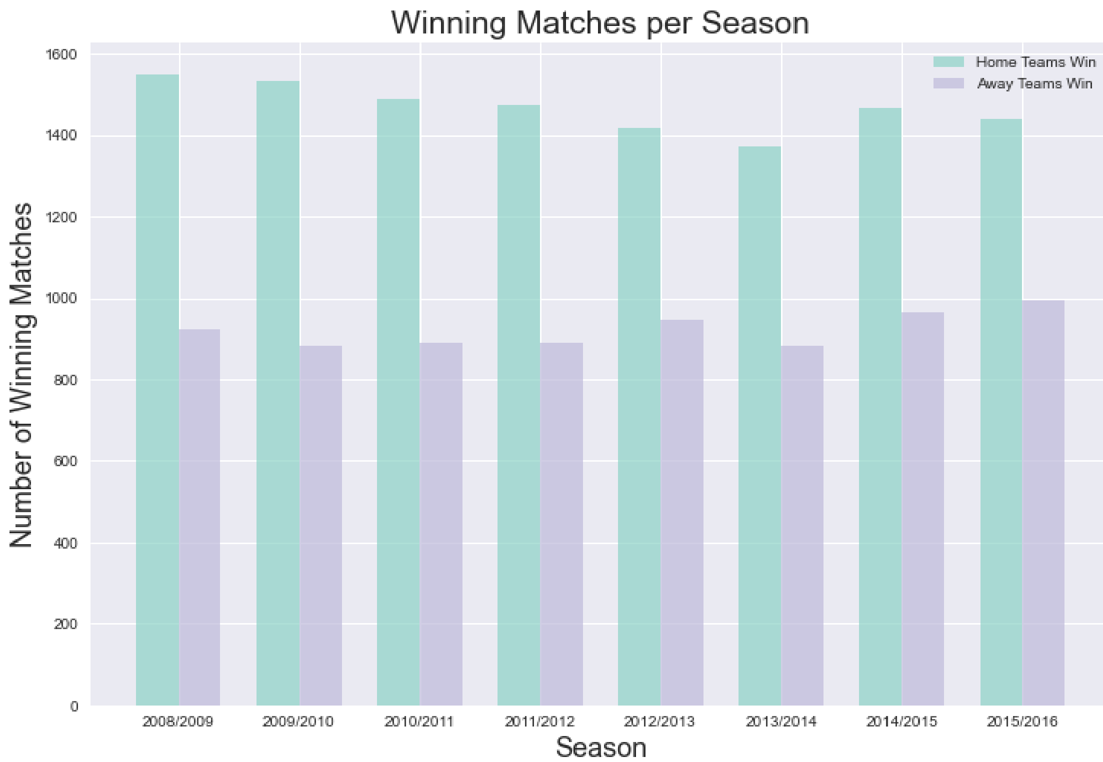
Which Season had the highest number of winnings by home and away teams?

```
# The season with highest number of home teams winning
home_season = win_home.groupby(['season']).count()
print('The season with highest number of home teams winning:\
n{}'.format(home_season['match_result'].nlargest(1)))

The season with highest number of home teams winning:
season
2008/2009    1550
Name: match_result, dtype: int64

# The season with highest number of away teams winning.
away_season = win_away.groupby(['season']).count()
```

```
print('The season with highest number of away teams winning:\
n{}'.format(away_season['match_result'].nlargest(1)))
```

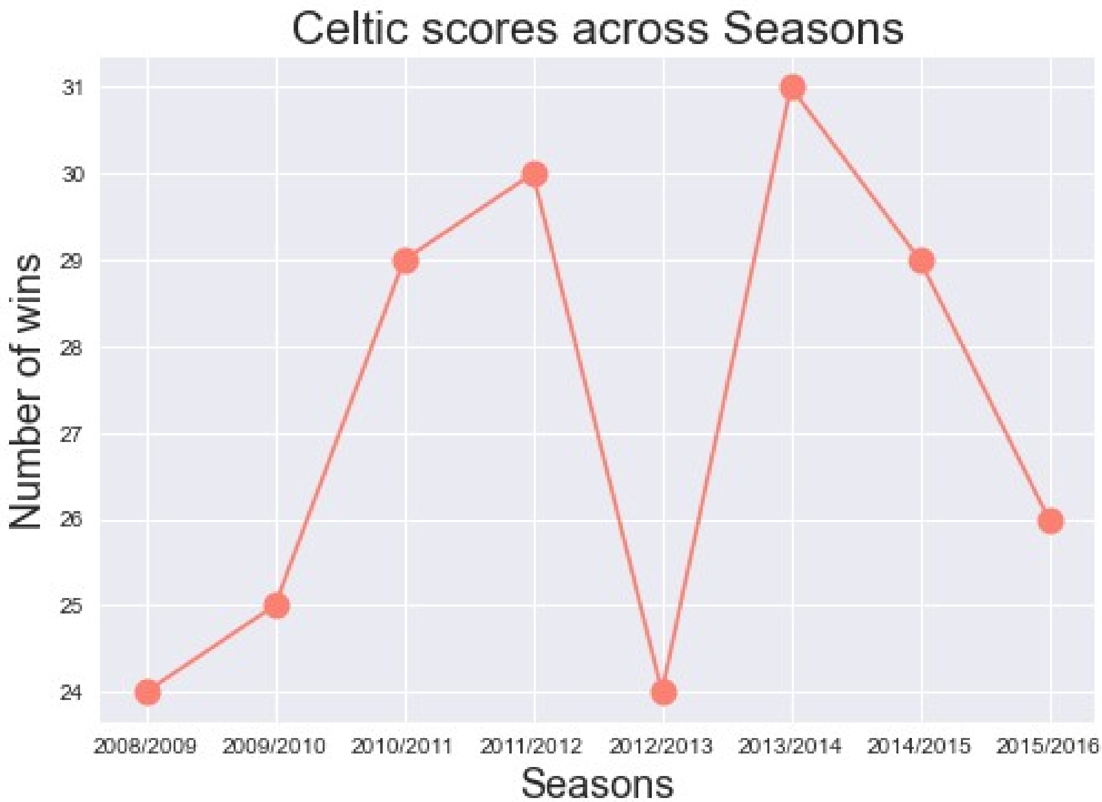
```
The season with highest number of away teams winning:
season
2015/2016    994
Name: match_result, dtype: int64
```

```
# Plot the results.

ind = np.arange(len(away_season['match_result']))
width = 0.35
fig = plt.subplots(figsize =(12, 8))

home_bars = plt.bar(ind, home_season['match_result'], width,
color=sns_colors[0], alpha=.7, label='Home Teams Win');
away_bars = plt.bar(ind+width, away_season['match_result'], width,
color=sns_colors[2], alpha=.7, label='Away Teams Win');

locations = ind + width / 2
labels = ['2008/2009', '2009/2010',
'2010/2011','2011/2012','2012/2013','2013/2014','2014/2015','2015/2016
']
plt.xticks(locations, labels);

plt.ylabel('Number of Winning Matches', fontsize=18);
plt.xlabel('Season', fontsize=18);
plt.title('Winning Matches per Season', fontsize=21);

plt.legend();
```



Did Celtic team improve its performance throughout the 8 seasons?

```
# Create a dataframe with match results end with Celtic winning.
df4= match.loc[match['match_result']== 'Celtic']
# Get number of winning matches per season for Celtic
df5= df4['season'].value_counts()

# Plot results
season=
['2008/2009','2009/2010','2010/2011','2011/2012','2012/2013','2013/201
4','2014/2015','2015/2016']
heights = df5.reindex(season)
labels = season

plt.plot(labels, heights, color=sns_colors[3], marker='o',
markerfacecolor=sns_colors[3], markersize=12);
plt.title('Celtic scores across Seasons', fontsize=21)
plt.xlabel('Seasons', fontsize=18);
plt.ylabel('Number of wins', fontsize=18);
```



Celtic team's performance fluctuated across the 8 seasons. They started with 24 wins in 2008/2009 and gradually increased till 2011/2012 achieving 30 wins. They had a fallout in 2012/2013 with 24 wins. In 2013/2014, they improved significantly achieveing 31 wins. Their performance started to drop gradually ending with 26 wins in 2015/2016.

What are the teams that improved the most over the time period?

To do this, I will have to calculate mean difference of total goal scores between two seasons which will be 2008/2009 and 2015/2016. First,I will have to get the mean of total goal scores seperately. Second, get their difference.

```
#2008/2009.
match_2008= match.query('season == "2008/2009"')
match_2008.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3296 entries, 0 to 25337
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   country_name    3296 non-null   object
 1   league_name     3296 non-null   object
 2   season          3296 non-null   object
 3   stage           3296 non-null   int64
 4   date            3296 non-null   object
```

```
 5   match_api_id    3296 non-null   int64
 6   home_team       3296 non-null   object
 7   away_team       3296 non-null   object
 8   home_team_goal  3296 non-null   int64
 9   away_team_goal  3296 non-null   int64
 10  match_result    3296 non-null   object
dtypes: int64(4), object(7)
memory usage: 309.0+ KB
```

```
# Calculate home teams mean goal scores.
home_2008= match_2008.groupby(['home_team'])['home_team_goal'].mean()
# Calculate away teams mean goal scores.
away_2008= match_2008.groupby(['away_team'])['away_team_goal'].mean()
# Add them to get the total goal scores for the whole season.
total_2008= (home_2008 + away_2008)/2
total_2008.head()
```

```
home_team
1. FC Köln       1.029412
AC Bellinzona    1.222222
ADO Den Haag     1.205882
AJ Auxerre       0.921053
AS Monaco        1.078947
dtype: float64
```

```
#2015/2016.
match_2015= match.query('season == "2015/2016"')
match_2015.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3266 entries, 91 to 25783
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   country_name    3266 non-null   object
 1   league_name     3266 non-null   object
 2   season          3266 non-null   object
 3   stage           3266 non-null   int64
 4   date            3266 non-null   object
 5   match_api_id    3266 non-null   int64
 6   home_team       3266 non-null   object
 7   away_team       3266 non-null   object
 8   home_team_goal  3266 non-null   int64
 9   away_team_goal  3266 non-null   int64
 10  match_result    3266 non-null   object
dtypes: int64(4), object(7)
memory usage: 306.2+ KB
```

```
# Calculate home teams mean goal scores.
home_2015= match_2015.groupby(['home_team'])['home_team_goal'].mean()
# Calculate away teams mean goal scores.
```

```
away_2015= match_2015.groupby(['away_team'])['away_team_goal'].mean()
# Add them to get the total goal scores for the whole season.
total_2015= (home_2015 + away_2015)/2
total_2015.head()
```

```
home_team
1. FC Köln           1.117647
1. FSV Mainz 05      1.352941
ADO Den Haag         1.411765
AS Monaco            1.500000
AS Saint-Étienne     1.105263
dtype: float64
```

```
# Get the difference between 2008 & 2015.
difference= total_2015 - total_2008
difference.head()
```

```
home_team
1. FC Köln          0.088235
1. FSV Mainz 05          NaN
AC Bellinzona            NaN
ADO Den Haag        0.205882
AJ Auxerre               NaN
dtype: float64
```

Since there is difference in number of matches between 2008/2009 and 2015/2016, there will be NaN values will conducting the necessary calculations. Thus, we will have to drop them.

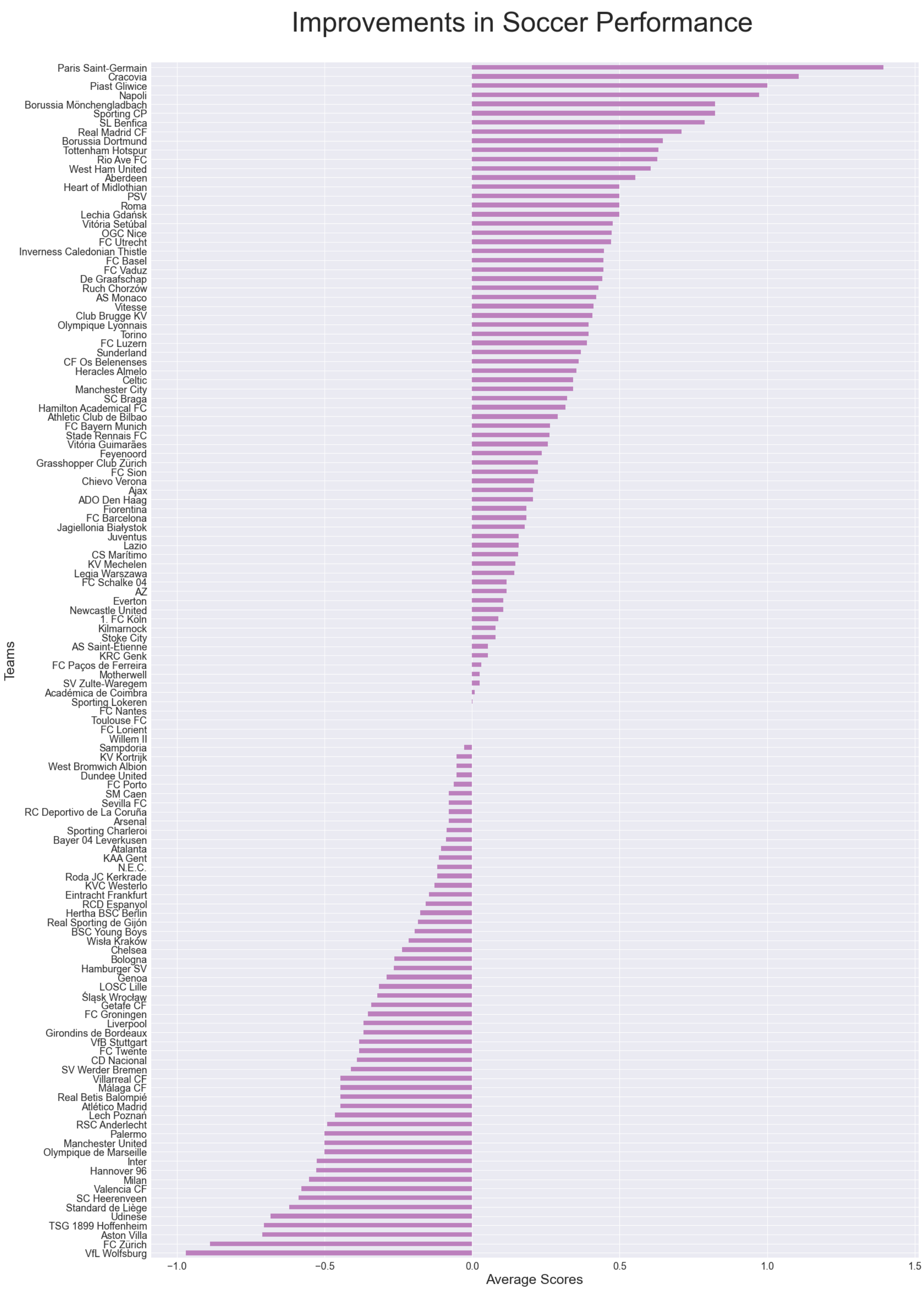
```
difference.dropna(inplace=True)
```

```
# Top 10 teams who achieved improvements throughout the 8 seasons
difference.sort_values(ascending=False)[:10]
```

```
home_team
Paris Saint-Germain          1.394737
Cracovia                     1.107143
Piast Gliwice                1.000000
Napoli                       0.973684
Borussia Mönchengladbach     0.823529
Sporting CP                  0.823529
SL Benfica                   0.788235
Real Madrid CF               0.710526
Borussia Dortmund            0.647059
Tottenham Hotspur            0.631579
dtype: float64
```

```
# Plot results
colors= sns_colors[9]
difference.sort_values(ascending=True).plot(kind='barh',fontsize=18,co
lor=colors, figsize=(25,40));
plt.figtext(.5,.9,'Improvements in Soccer Performance', fontsize=50,
```

```python
ha='center')
plt.xlabel('Average Scores', fontsize=25);
plt.ylabel('Teams', fontsize=25);
```



What team attributes lead to the most victories?

```
# top three teams throughout the 8 seasons.
top= match['match_result'].value_counts().sort_values(ascending=False)
[1:4]
top
```

```
FC Barcelona      234
Real Madrid CF    228
Celtic            218
Name: match_result, dtype: int64
```

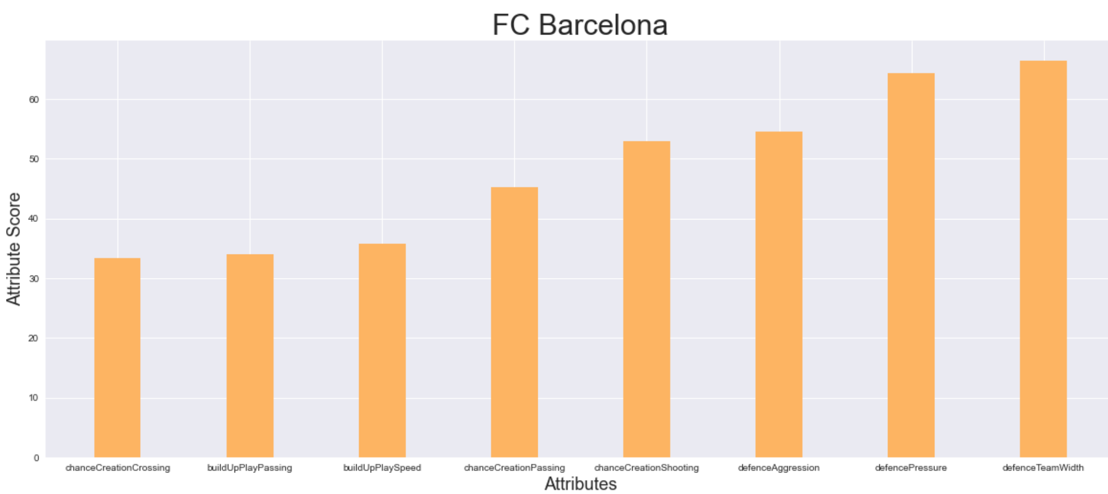
```
team_attribute.describe()
```

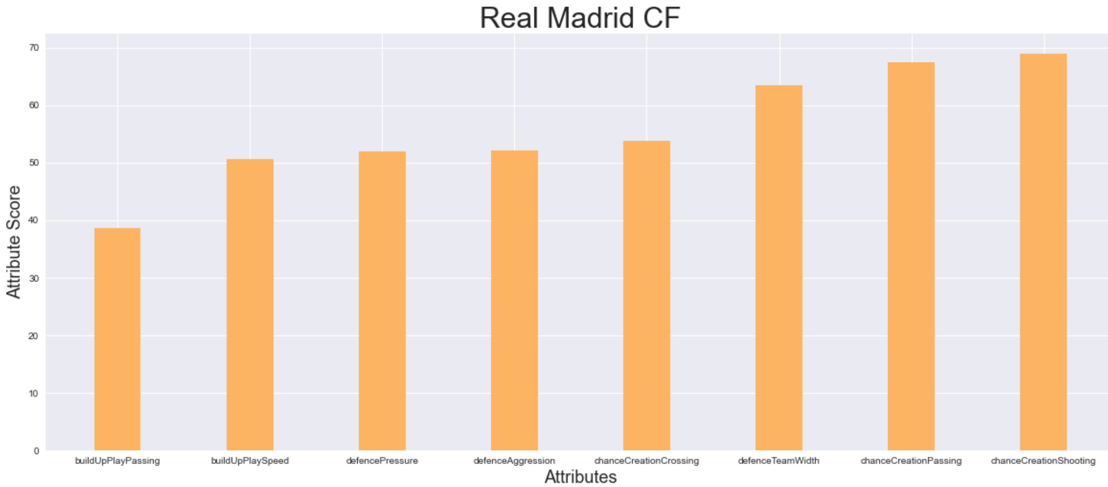
```
       buildUpPlaySpeed  buildUpPlayPassing  chanceCreationPassing  \
count       1451.000000         1451.000000            1451.000000
mean          52.448656           48.456237              52.170227
std           11.537493           10.880225              10.354907
min           20.000000           20.000000              21.000000
25%           45.000000           40.000000              46.000000
50%           52.000000           50.000000              52.000000
75%           62.000000           55.000000              59.000000
max           80.000000           80.000000              80.000000

       chanceCreationCrossing  chanceCreationShooting  defencePressure
\
count             1451.000000             1451.000000      1451.000000

mean                53.732598               53.964163        46.035837

std                 11.073742               10.343018        10.224249

min                 20.000000               22.000000        23.000000

25%                 47.000000               48.000000        39.000000

50%                 53.000000               53.000000        45.000000

75%                 62.000000               61.000000        51.000000

max                 80.000000               80.000000        72.000000


       defenceAggression  defenceTeamWidth
count        1451.000000       1451.000000
mean           49.254307         52.164714
std             9.725727          9.582082
min            24.000000         29.000000
25%            44.000000         47.000000
50%            48.000000         52.000000
```

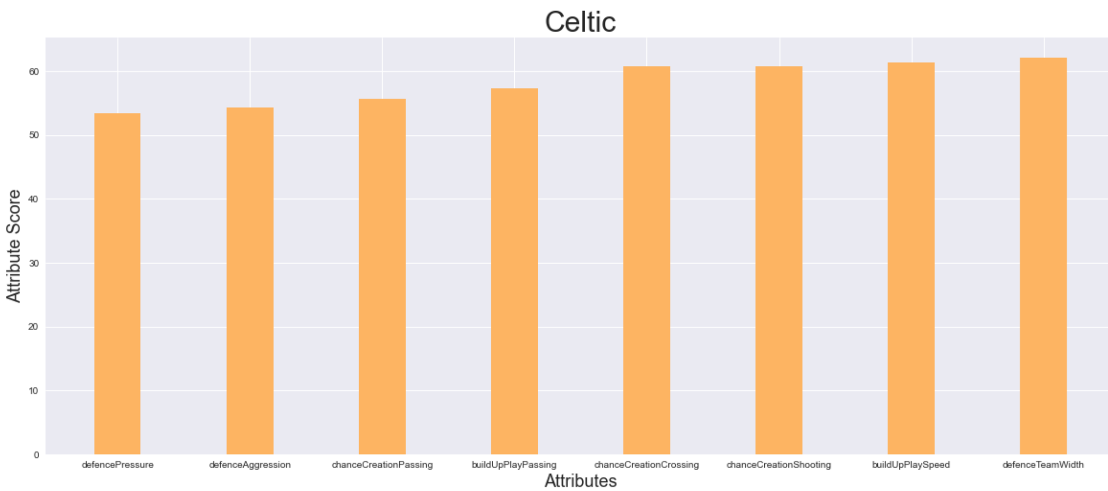
```
75%              55.000000          58.000000
max              72.000000          73.000000
```

On a whole scale, there are total 1451 entries for each attribute. Overall mean scores is between 46 and 53.9. Maximum score recorded is 80 and minimum score recorded is 20 across all attributes.

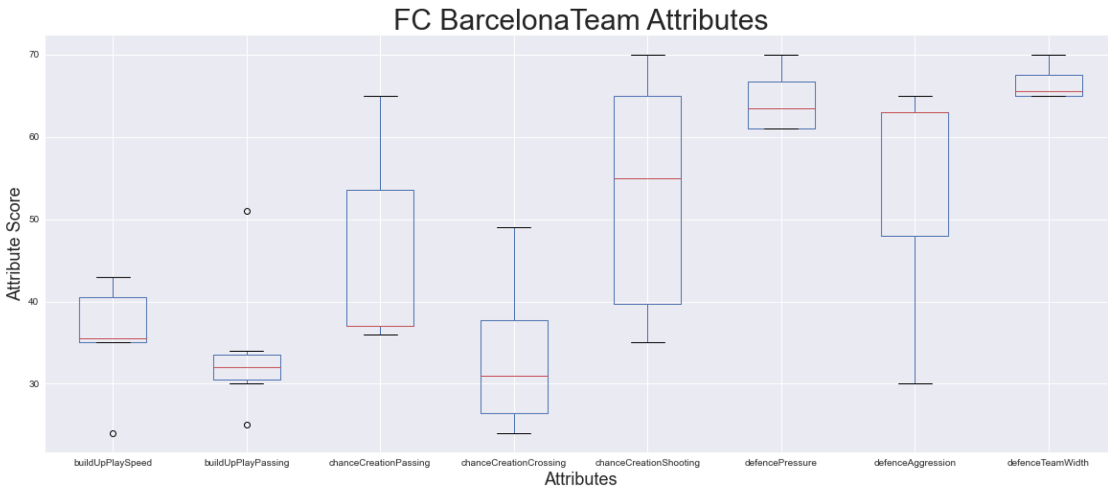
```
# Using for loop, get mean values of team attributes for each top three teams.
list= top.to_dict().keys()
for team in list:
    x= team_attribute.loc[(team_attribute.team_long_name == team)]
    y= x.mean(numeric_only=True).sort_values().to_dict()

    keys = y.keys()
    values = y.values()

    fig = plt.figure(figsize=(20,8));

    plt.bar(keys, values, color =sns_colors[5] ,width = 0.35);
    plt.title(team, fontsize=30);
    plt.ylabel('Attribute Score', fontsize=18);
    plt.xlabel('Attributes', fontsize=18);
```

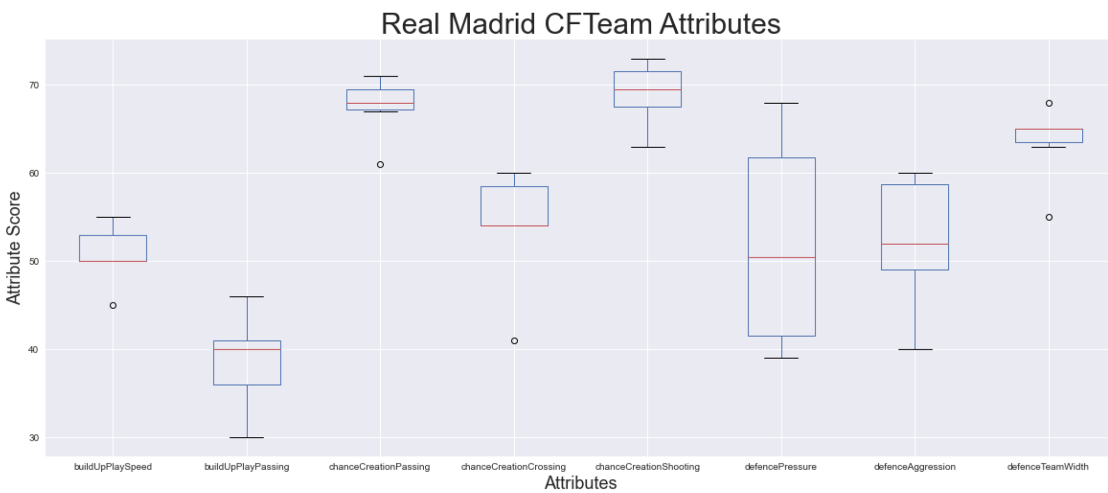


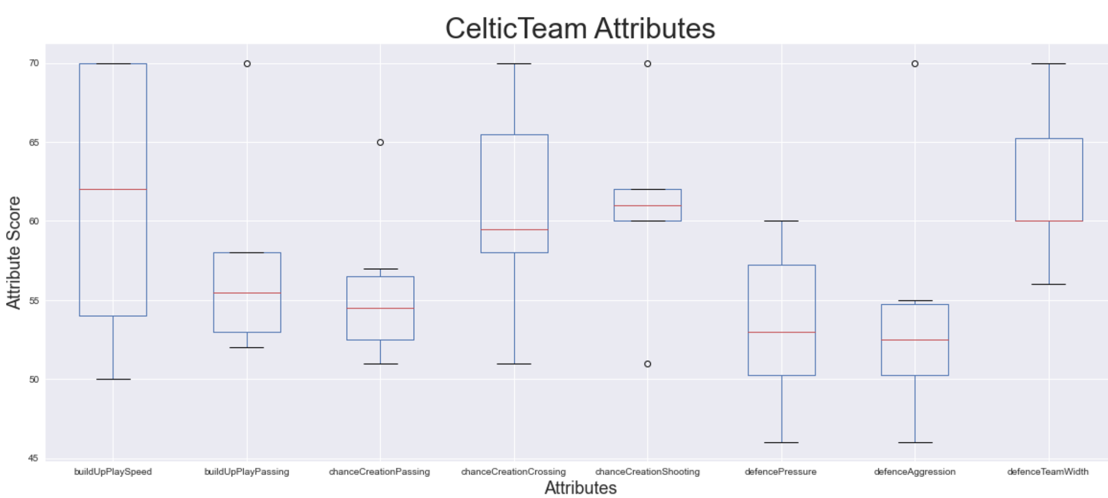





```python
# Use box plot to show the five-number summary for each attribute for each top team.
list= ['FC Barcelona','Real Madrid CF','Celtic']
for item in list:
    x= team_attribute.loc[(team_attribute.team_long_name == item)]
    x.groupby('team_long_name').plot(kind='box', figsize=(20,8));
    plt.title(item + 'Team Attributes', fontsize=30);
    plt.ylabel('Attribute Score', fontsize=18);
    plt.xlabel('Attributes', fontsize=18);
```







Who is the oldest and the youngest player?

```
#Youngest player.
df_players['birthday'].max()

Timestamp('1999-04-24 00:00:00')
```

```
young_player= df_players.query('birthday == birthday.max()')
print('Youngest Player is:')
young_player
```

```
Youngest Player is:
```

	player_id	player_name	birthday	height	weight
5176	682552	Jonathan Leko	1999-04-24	182.88	141

```
# Oldest player.
df_players['birthday'].min()
```

```
Timestamp('1967-01-23 00:00:00')
```

```
old_player= df_players.query('birthday == birthday.min()')
print('Oldest Player is:')
old_player
```

```
Oldest Player is:
```

	player_id	player_name	birthday	height	weight
289	39425	Alberto Fontana	1967-01-23	185.42	161

Who is the tallest players?

```
df_players['height'].max()
```

```
208.28
```

```
tallest_player= df_players.query('height == height.max()')
print('Tallest Player is:')
tallest_player
```

```
Tallest Player is:
```

	player_id	player_name	birthday	height	weight
5901	148325	Kristof van Hout	1987-02-09	208.28	243

who has the highest and the lowest average of overall rating?

```
# Highest average of overall rating.
highest_rating= players.groupby('player_name')
highest_rating.overall_rating.mean().sort_values(ascending=False)[:1]
```

```
player_name
Lionel Messi    92.192308
Name: overall_rating, dtype: float64
```

```
# Lowest average of overall rating.
lowest_rating= players.groupby('player_name')
lowest_rating.overall_rating.mean().sort_values(ascending=False)[-1:]
```

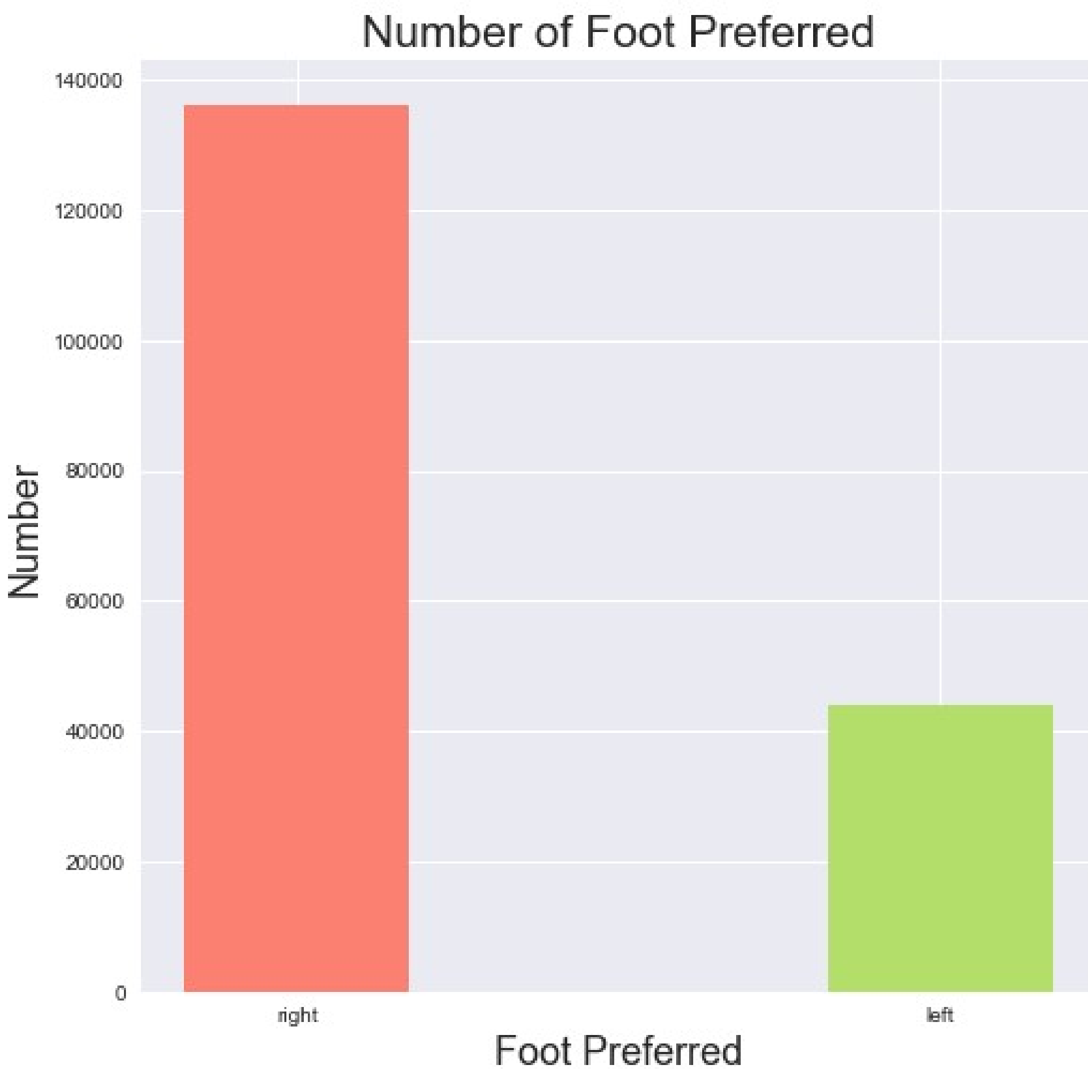
```
player_name
Gianluca D'Angelo    43.75
Name: overall_rating, dtype: float64
```

How many players use either preferred right or left foot?

```
players['preferred_foot'].value_counts()
```

```
right    136247
left      44107
Name: preferred_foot, dtype: int64
```

```
# Plot results using bar chart.
foot= players['preferred_foot'].value_counts().to_dict()
keys = foot.keys()
values = foot.values()

fig = plt.figure(figsize = (8, 8));

plt.bar(keys, values, color =[sns_colors[3], sns_colors[6]], width =
0.35);
plt.xlabel("Foot Preferred", fontsize=18);
plt.ylabel("Number", fontsize=18);
plt.title("Number of Foot Preferred", fontsize=21);
```



Who made the most penalties?

```
# Use average.
penalty= players.groupby('player_name')
print('The player who made the most penalties:')
penalty.penalties.mean().sort_values(ascending=False)[:1]
```

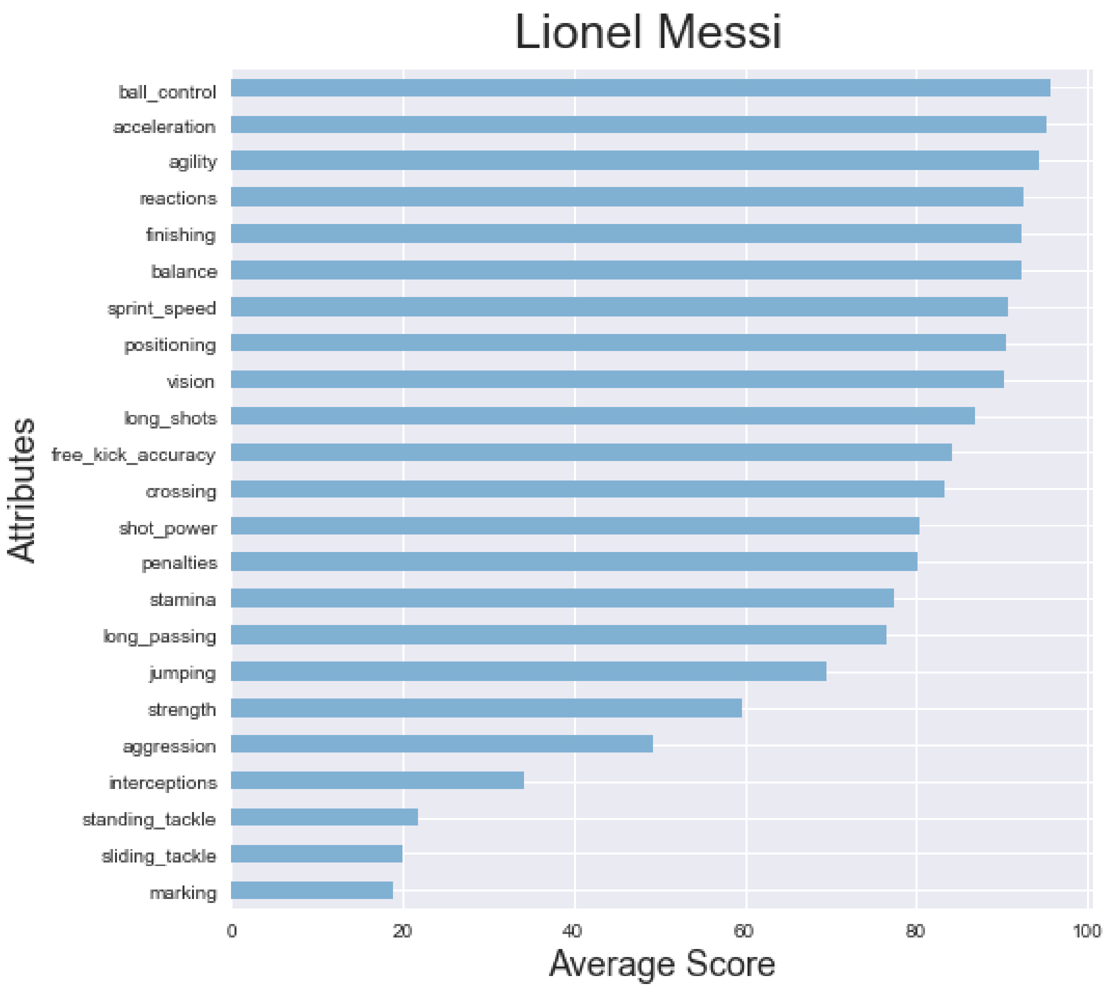
```
The player who made the most penalties:

player_name
Mario Balotelli    89.565217
Name: penalties, dtype: float64
```

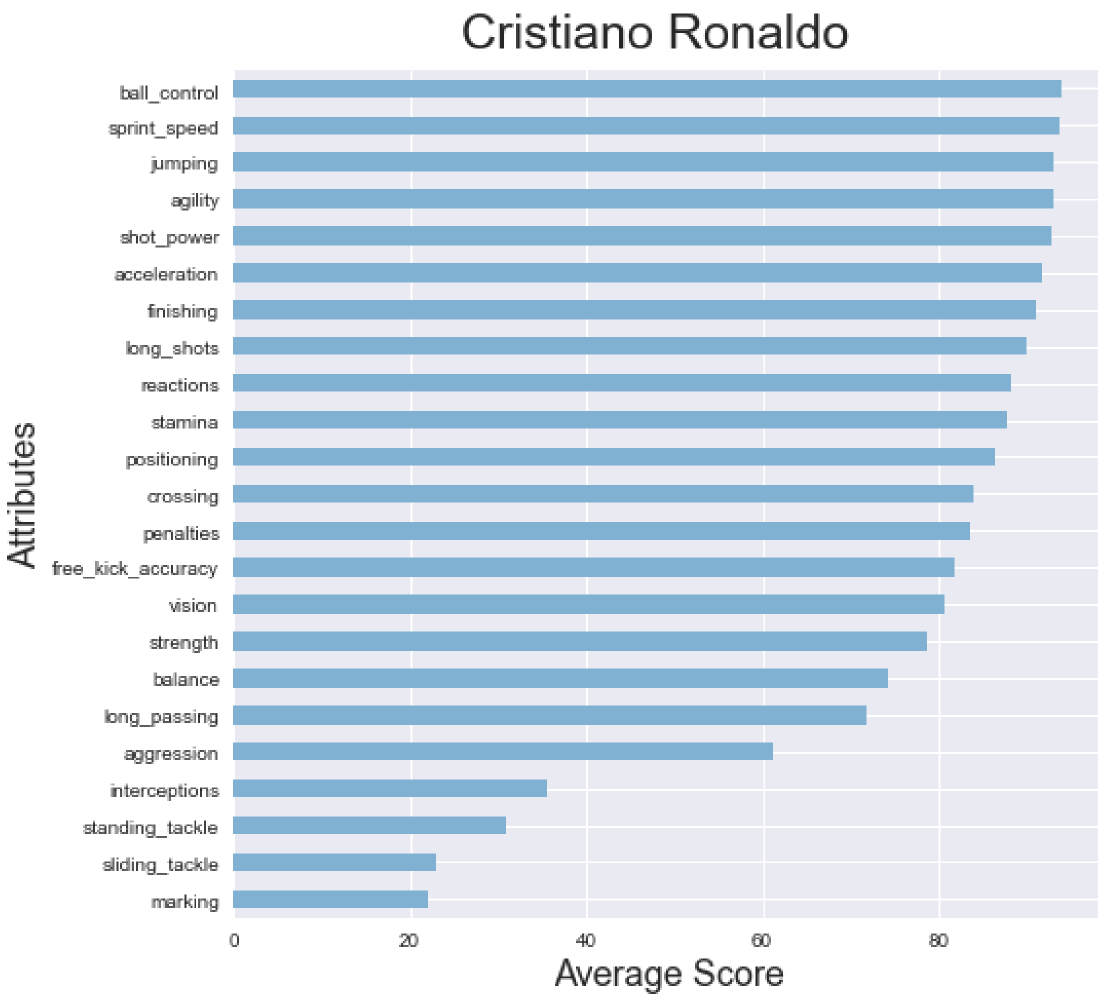
What are the attributes of the 5 best players based on their average overall ratings?

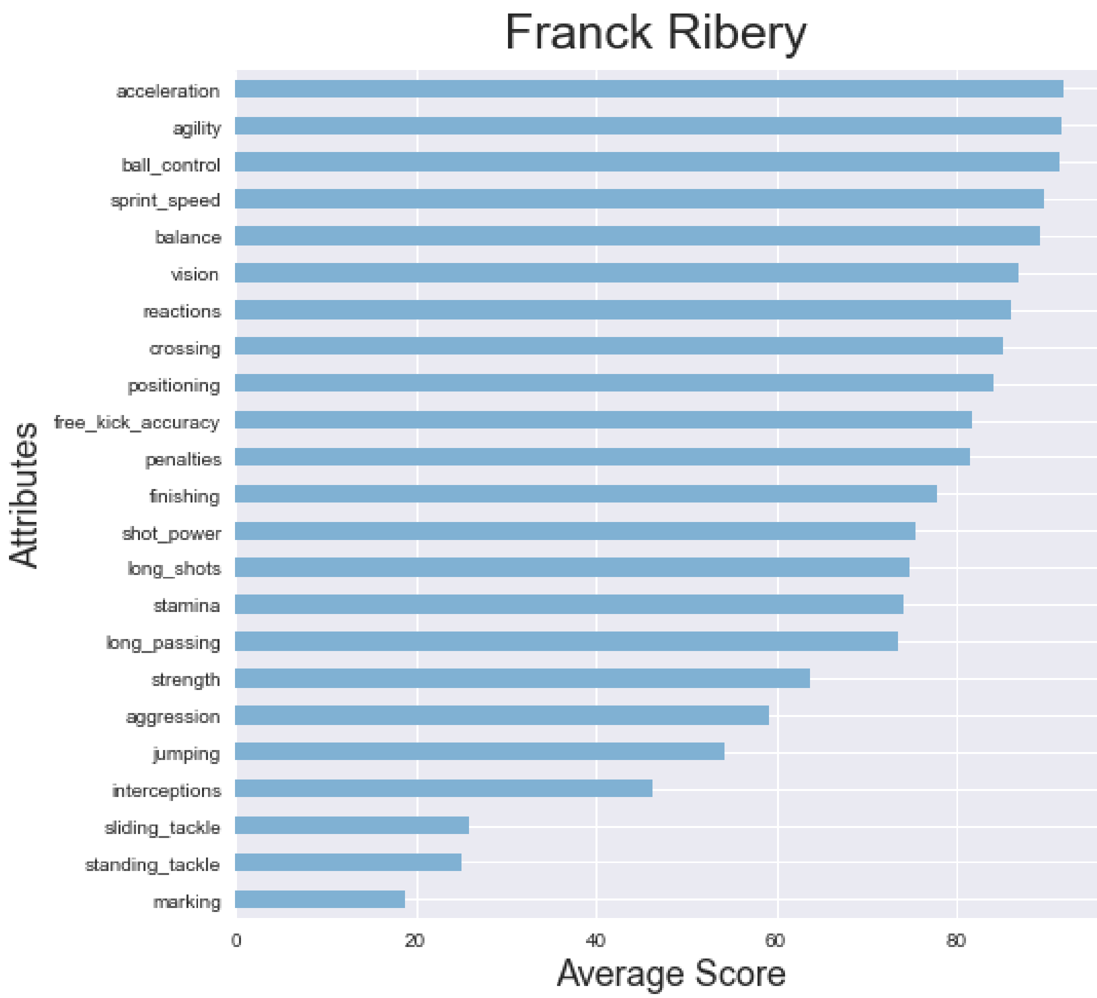
```
best_players=
players.groupby('player_name').overall_rating.mean().sort_values(ascen
ding=False)[:5]
best_players.to_dict().keys()
```

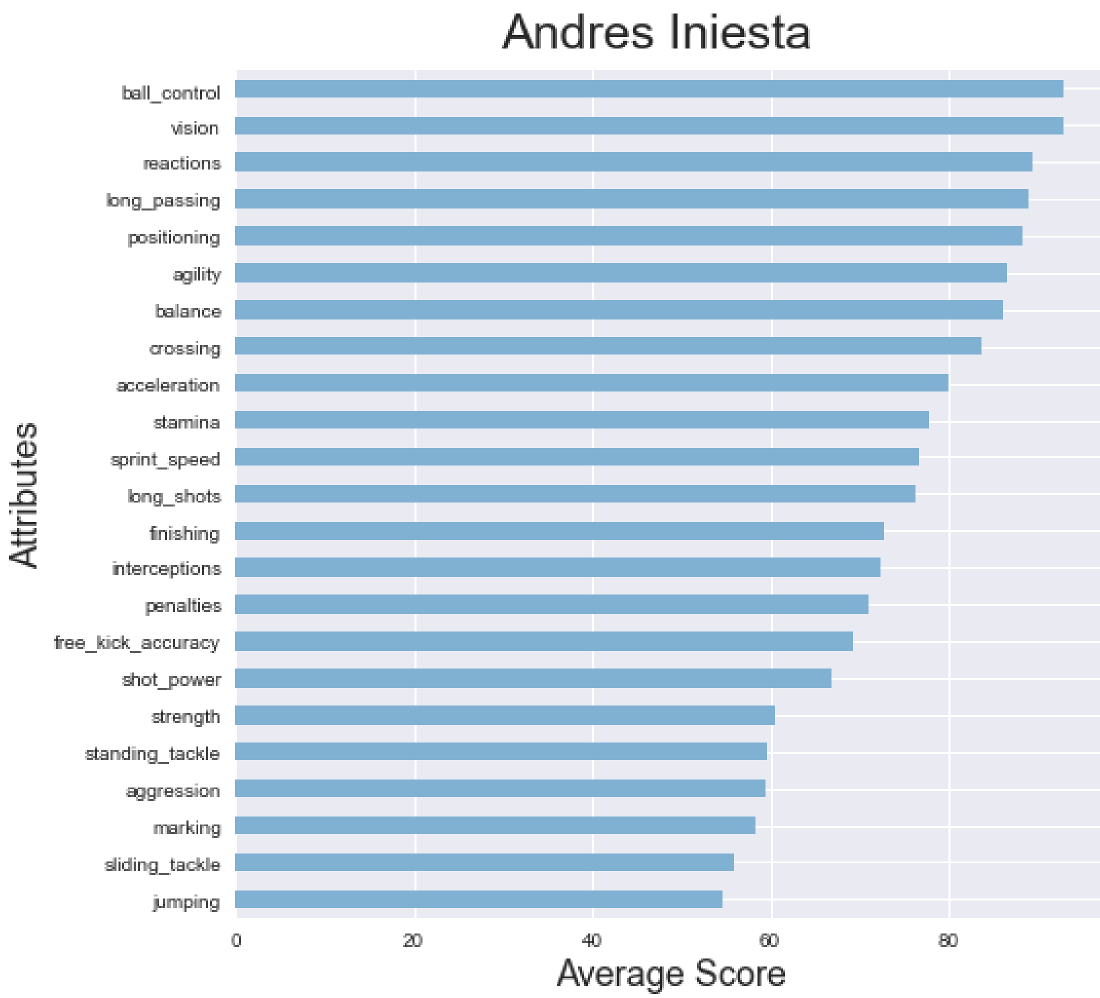
```
dict_keys(['Lionel Messi', 'Cristiano Ronaldo', 'Franck Ribery',
'Andres Iniesta', 'Zlatan Ibrahimovic'])
```

```
# Drop few not needed columns.
players.drop(['player_id','birthday',
'height','overall_rating','potential', 'weight','gk_diving',
'gk_handling',
       'gk_kicking', 'gk_positioning', 'gk_reflexes'], axis=1,
inplace=True)

# Using for loop, get mean values of player attributes for each 5 best
players.
list= best_players.to_dict()
for player in list:
    x= players.loc[(players.player_name == player)]
    y= x.mean(numeric_only=True).sort_values()
    colors= sns_colors[4]
    y.plot(kind='barh', color=colors, figsize=(8,8));
    plt.figtext(.5,.9,player, fontsize=25, ha='center');
    plt.xlabel('Average Score', fontsize=18);
    plt.ylabel('Attributes', fontsize=18);
    plt.show();
```

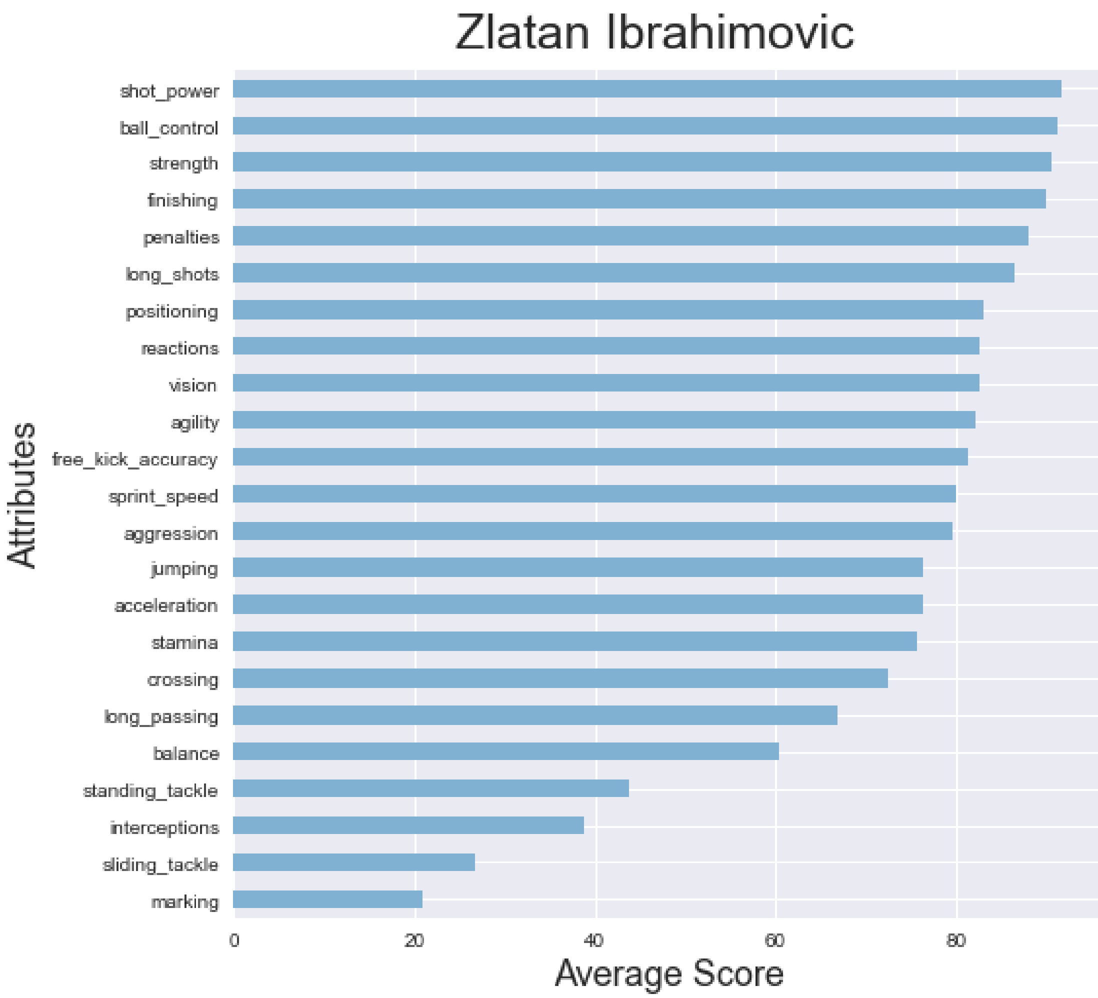

Lionel Messi
Attributes
Average Score
ball_control
acceleration
agility
reactions
finishing
balance
sprint_speed
positioning
vision
long_shots
free_kick_accuracy
crossing
shot_power
penalties
stamina
long_passing
jumping
strength
aggression
interceptions
standing_tackle
sliding_tackle
marking
0
20
40
60
80
100


Cristiano Ronaldo
ball_control
sprint_speed
jumping
agility
shot_power
acceleration
finishing
long_shots
reactions
stamina
positioning
crossing
penalties
free_kick_accuracy
vision
strength
balance
long_passing
aggression
interceptions
standing_tackle
sliding_tackle
marking
Attributes
0
20
40
60
80
Average Score


Franck Ribery
Attributes
acceleration
agility
ball_control
sprint_speed
balance
vision
reactions
crossing
positioning
free_kick_accuracy
penalties
finishing
shot_power
long_shots
stamina
long_passing
strength
aggression
jumping
interceptions
sliding_tackle
standing_tackle
marking
0
20
40
60
80
Average Score



Andres Iniesta
ball_control
vision
reactions
long_passing
positioning
agility
balance
crossing
acceleration
stamina
sprint_speed
long_shots
finishing
interceptions
penalties
free_kick_accuracy
shot_power
strength
standing_tackle
aggression
marking
sliding_tackle
jumping
Attributes
0
20
40
60
80
Average Score



Conclusions

From the soccer datasets provided, we can conclude the following:

1. Season 2008/2009 had the highest number of matches.
2. England, France, and Spain with their respective leagues hosted the highest number of matches throughout the 8 seasons.
3. 'FC Barcelona','Real Madrid CF','Celtic' are the most victorious teams throughout the 8 seasons.
4. Teams who play on their homeland are more likely to win by 45.87%.
5. Paris Saint-German has made the highest performance progress throughout the 8 seasons.
6. Features that mostly lead teams to victory differs among diferent teams. But for the top three teams these are their highest attribute mean score:
   - FC Barcelona: defence pressure and defence team width.
   - Real Madrid CF: chance creation passing and chance creation shooting.

- Celtic: build up play speed and defence team width.

7. Jonathan Leko is the youngest player, while Alberto Fontana is the oldest player.
8. Kristof van Hout is the tallest player.
9. Lionel Messi had the highest average overall rating, while Gianluca D'Angelo has the lowest average overall rating.
10. Number of players who prefere to use their right foot is greater than those who use their left foot.
11. Mario Balotelli made the most penalties on average.
12. Combined top attributes differs for each player.
    - Lionel Messi: ball control, acceleration, and agility.
    - Cristiano Ronaldo: ball control, sprint speed, and jumping.
    - Frank Ribery: acceleration, agility, and ball control.
    - Andres Iniesta: ball control, vision, and reactions.
    - Zlatan Ibrahimovic: shot power, ball control, and strength.

Limitation:

- There is no explanation for attributes in both teams and players.
- Duplicated team names with different id numbers.
- Duplicated players name but with different set of data. Also, in player_name column there were some players with only first name or surname.
- The player dataset doesn't include what team each player belonged to.