Project 2: Investigate the European Soccer Database

Table of Contents

Introduction

For this project, I will investigate the European Soccer Database from Kaggle which contains seven tables: "Country", "League", "Match", "Player", "Player_Attributes", "Team", and "Team_Attributes". This dataset covers 11 of the top European leagues from 2008 to 2016. As a lifelong football tragic, it was an easy choice to select this dataset for investigation.

The key questions that I would like to answer include:

1. **Home fortress?**: Does a home advantage exist, and if so, which league has the strongest such advantage?
2. **Who are the goal machines and goal leakers?**: Who are the best and worst performing home and away teams?
3. **Who runs North London?**: Which team in North London has achieved the most wins?
4. **Speedy gonzales**: Is pace a factor in a player's overall rating?

Import Packages

In [1]:

```
# Import packages

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

Data Wrangling

Data Gathering

To gather data, I downloaded the dataset as 'database.sqllite' from Kaggle. I also downloaded and installed 'TablePlus' (a SQL client) which I used to select and join datasets.

Firstly, I brainstormed what tables and fields I may require to answer my research questions.

I noted that I would primarily need:

1. the 'Match' table (joined with 'Country' and 'League' to distinguish between countries and leagues; and also joined with 'Team' to bring in team names), and
2. the 'Player' table (joined with 'Player_Attributes' to bring in the attribute-level data for players).

Then I used my knowledge of SQL to perform the required joins, and exported the data to csv format (see SQL scripts below).

SQL script for (1): Exported as match.csv

```
SELECT Match.id,
Country.name AS country_name,
League.name AS league_name,
season,
stage,
date,
H.team_long_name AS home_team,
A.team_long_name AS away_team,
home_team_goal,
away_team_goal

FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS H ON H.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS A ON A.team_api_id = Match.away_team_api_id;
```

SQL script for (2): Exported as player.csv

```
SELECT * FROM Player
INNER JOIN Player_Attributes ON Player.player_api_id = Player_Attributes.player_api_id;
```

Import CSVs

In [2]:

```
# Import CSVs

match = pd.read_csv('match.csv')
player = pd.read_csv('player.csv')
```

In [3]:

```
# Explore 'match' data

match.head()
```

Out[3]:

	id	country_name	league_name	season	stage	date	home_team	away_team	hom
0	1	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17 00:00:00	KRC Genk	Beerschot AC	
1	2	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16 00:00:00	SV Zulte-Waregem	Sporting Lokeren	
2	3	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16 00:00:00	KSV Cercle Brugge	RSC Anderlecht	
3	4	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17 00:00:00	KAA Gent	RAEC Mons	
4	5	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16 00:00:00	FCV Dender EH	Standard de Liège	

Ideas for data wrangling / feature engineering:

- Add a column indicating who the winner of the match was (i.e. home team, away team, or drawn).

In [4]:

```
match.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25979 entries, 0 to 25978
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   id              25979 non-null  int64 
 1   country_name    25979 non-null  object
 2   league_name     25979 non-null  object
 3   season          25979 non-null  object
 4   stage           25979 non-null  int64 
 5   date            25979 non-null  object
 6   home_team       25979 non-null  object
 7   away_team       25979 non-null  object
 8   home_team_goal  25979 non-null  int64 
 9   away_team_goal  25979 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 2.0+ MB
```

There do not seem to be any null values (25979 observations across all columns). Note, the need to drop null values has been avoided by selecting columns which do not have nulls during the data gathering phase (there are columns in the original SQL dataset which contain null values, and if these were carried into this dataset then I would have needed to drop the null values at that point).

Ideas for data wrangling / feature engineering:

- Date should be changed to date-time format.

In [4]:

```
match.shape
```

Out[4]:

```
(25979, 10)
```

In [5]:

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

Out[5]:

```
id                  0
country_name        0
league_name         0
season              0
stage               0
date                0
home_team           0
away_team           0
home_team_goal      0
away_team_goal      0
dtype: int64
```

In [6]:

```
match.duplicated().sum()
```

Out[6]:

```
0
```

There do not seem to be any duplicated values.

In [7]:

```
# Explore 'player' data

player.head()
```

Out[7]:

	id	player_api_id	player_name	player_fifa_api_id	birthday	height	weight	id.1	player_fifa
0	1	505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187	1	
1	1	505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187	2	
2	1	505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187	3	
3	1	505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187	4	
4	1	505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187	5	

5 rows × 49 columns

In [8]:

```
player.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183978 entries, 0 to 183977
Data columns (total 49 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   id                   183978 non-null  int64  
 1   player_api_id        183978 non-null  int64  
 2   player_name          183978 non-null  object 
 3   player_fifa_api_id   183978 non-null  int64  
 4   birthday             183978 non-null  object 
 5   height               183978 non-null  float64
 6   weight               183978 non-null  int64  
 7   id.1                 183978 non-null  int64  
 8   player_fifa_api_id.1 183978 non-null  int64  
 9   player_api_id.1      183978 non-null  int64  
 10  date                 183978 non-null  object 
 11  overall_rating       183142 non-null  float64
 12  potential            183142 non-null  float64
 13  preferred_foot       183142 non-null  object 
 14  attacking_work_rate  180748 non-null  object 
 15  defensive_work_rate  183142 non-null  object 
 16  crossing             183142 non-null  float64
 17  finishing            183142 non-null  float64
 18  heading_accuracy     183142 non-null  float64
 19  short_passing        183142 non-null  float64
 20  volleys              181265 non-null  float64
 21  dribbling            183142 non-null  float64
 22  curve                181265 non-null  float64
 23  free_kick_accuracy   183142 non-null  float64
 24  long_passing         183142 non-null  float64
 25  ball_control         183142 non-null  float64
 26  acceleration         183142 non-null  float64
 27  sprint_speed         183142 non-null  float64
 28  agility              181265 non-null  float64
 29  reactions            183142 non-null  float64
 30  balance              181265 non-null  float64
 31  shot_power           183142 non-null  float64
 32  jumping              181265 non-null  float64
 33  stamina              183142 non-null  float64
 34  strength             183142 non-null  float64
 35  long_shots           183142 non-null  float64
 36  aggression           183142 non-null  float64
 37  interceptions        183142 non-null  float64
 38  positioning          183142 non-null  float64
 39  vision               181265 non-null  float64
 40  penalties            183142 non-null  float64
 41  marking              183142 non-null  float64
 42  standing_tackle      183142 non-null  float64
 43  sliding_tackle       181265 non-null  float64
 44  gk_diving            183142 non-null  float64
 45  gk_handling          183142 non-null  float64
 46  gk_kicking           183142 non-null  float64
 47  gk_positioning       183142 non-null  float64
 48  gk_reflexes          183142 non-null  float64
dtypes: float64(36), int64(7), object(6)
memory usage: 68.8+ MB
```

I do not intend to use 'date', so I will not clean this. There are evidently null values throughout the dataset which I will dig into further.

In [9]:

```
player.shape
```

Out[9]:

```
(183978, 49)
```

In [10]:

```
player.isnull().sum()
```

Out[10]:

```
id                        0
player_api_id             0
player_name               0
player_fifa_api_id        0
birthday                  0
height                    0
weight                    0
id.1                      0
player_fifa_api_id.1      0
player_api_id.1           0
date                      0
overall_rating          836
potential               836
preferred_foot          836
attacking_work_rate    3230
defensive_work_rate     836
crossing                836
finishing               836
heading_accuracy        836
short_passing           836
volleys                2713
dribbling               836
curve                  2713
free_kick_accuracy      836
long_passing            836
ball_control            836
acceleration            836
sprint_speed            836
agility                2713
reactions               836
balance                2713
shot_power              836
jumping                2713
stamina                 836
strength                836
long_shots              836
aggression              836
interceptions           836
positioning             836
vision                 2713
penalties               836
marking                 836
standing_tackle         836
sliding_tackle         2713
gk_diving               836
gk_handling             836
gk_kicking              836
gk_positioning          836
gk_reflexes             836
dtype: int64
```

From a total of 183978 rows, there are a number of columns which contain between 836 and 3230 rows of null values. Given the research question which relies on the 'player' dataset is more concerned at a broader / macro level trend, I have taken the decision to drop all null values.

In [11]:

```
player.duplicated().sum()
```

Out[11]:

```
0
```

Data Cleaning

In [12]:

```
# Clean 'match' data

match_clean = match.copy()
```

In [13]:

```
# Convert 'date' to datetime format

match_clean['date'] = match_clean['date'].astype('datetime64[ns]')
```

In [14]:

```
match_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25979 entries, 0 to 25978
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              25979 non-null  int64
 1   country_name    25979 non-null  object
 2   league_name     25979 non-null  object
 3   season          25979 non-null  object
 4   stage           25979 non-null  int64
 5   date            25979 non-null  datetime64[ns]
 6   home_team       25979 non-null  object
 7   away_team       25979 non-null  object
 8   home_team_goal  25979 non-null  int64
 9   away_team_goal  25979 non-null  int64
dtypes: datetime64[ns](1), int64(4), object(5)
memory usage: 2.0+ MB
```

In [15]:

```
# Clean 'player' data

player_clean = player.copy()
```

In [16]:

```
# Drop null values

player_clean.dropna(inplace=True)
```

In [17]:

```
player_clean.isnull().sum()
```

Out[17]:

```
id                     0
player_api_id          0
player_name            0
player_fifa_api_id     0
birthday               0
height                 0
weight                 0
id.1                   0
player_fifa_api_id.1   0
player_api_id.1        0
date                   0
overall_rating         0
potential              0
preferred_foot         0
attacking_work_rate    0
defensive_work_rate    0
crossing               0
finishing              0
heading_accuracy       0
short_passing          0
volleys                0
dribbling              0
curve                  0
free_kick_accuracy     0
long_passing           0
ball_control           0
acceleration           0
sprint_speed           0
agility                0
reactions              0
balance                0
shot_power             0
jumping                0
stamina                0
strength               0
long_shots             0
aggression             0
interceptions          0
positioning            0
vision                 0
penalties              0
marking                0
standing_tackle        0
sliding_tackle         0
gk_diving              0
gk_handling            0
gk_kicking             0
gk_positioning         0
gk_reflexes            0
dtype: int64
```

Feature Engineering

In [18]:

```
match_clean.head()
```

Out[18]:

	id	country_name	league_name	season	stage	date	home_team	away_team	home_t
0	1	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17	KRC Genk	Beerschot AC	
1	2	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	SV Zulte-Waregem	Sporting Lokeren	
2	3	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	KSV Cercle Brugge	RSC Anderlecht	
3	4	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17	KAA Gent	RAEC Mons	
4	5	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	FCV Dender EH	Standard de Liège	

In [19]:

```
# Create function to determine outcome of the match, and then create a column th
at contains this information

def outcome(a):
    home_score = a[0]
    away_score = a[1]
    home_team = a[2]
    away_team = a[3]

    if home_score > away_score:
        return home_team
    elif home_score < away_score:
        return away_team
    else:
        return 'Drawn'

match_clean['outcome']  = match_clean[['home_team_goal', 'away_team_goal', 'home
_team', 'away_team']].apply(outcome, axis=1)
```

Note, I used the following GitHub repo as a guide for the above function: https://github.com/ozlerhakan/soccer/blob/master/investigate-a-dataset-template-main.ipynb (https://github.com/ozlerhakan/soccer/blob/master/investigate-a-dataset-template-main.ipynb)

In [20]:

```
match_clean.head()
```

Out[20]:

	id	country_name	league_name	season	stage	date	home_team	away_team	home_t
0	1	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17	KRC Genk	Beerschot AC	
1	2	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	SV Zulte-Waregem	Sporting Lokeren	
2	3	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	KSV Cercle Brugge	RSC Anderlecht	
3	4	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17	KAA Gent	RAEC Mons	
4	5	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	FCV Dender EH	Standard de Liège	

Exploratory Data Analysis

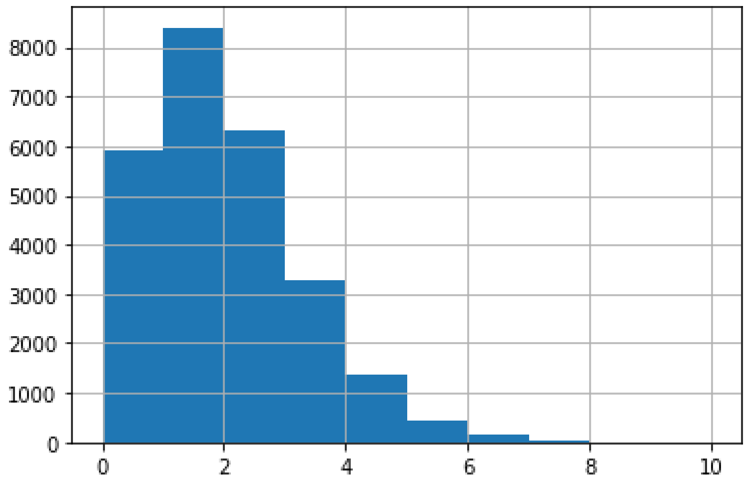
Research Question 1)

Home fortress?: Does a home advantage exist, and if so, which league has the strongest such advantage?

In [21]:

```
# Distribution of home team goals across all leagues

match_clean.home_team_goal.hist();
```



In [22]:

```
match_clean.home_team_goal.describe()
```

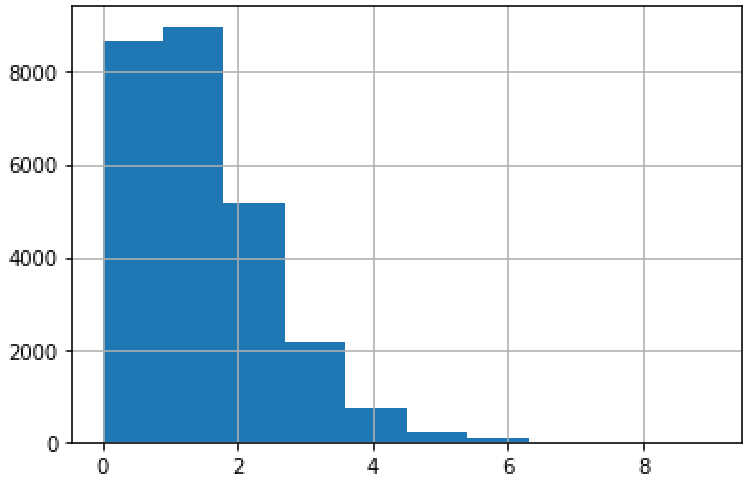
Out[22]:

```
count    25979.000000
mean         1.544594
std          1.297158
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max         10.000000
Name: home_team_goal, dtype: float64
```

In [23]:

```
# Distribution of away team goals across all leagues

match_clean.away_team_goal.hist();
```



In [24]:

```
match_clean.away_team_goal.describe()
```

Out[24]:

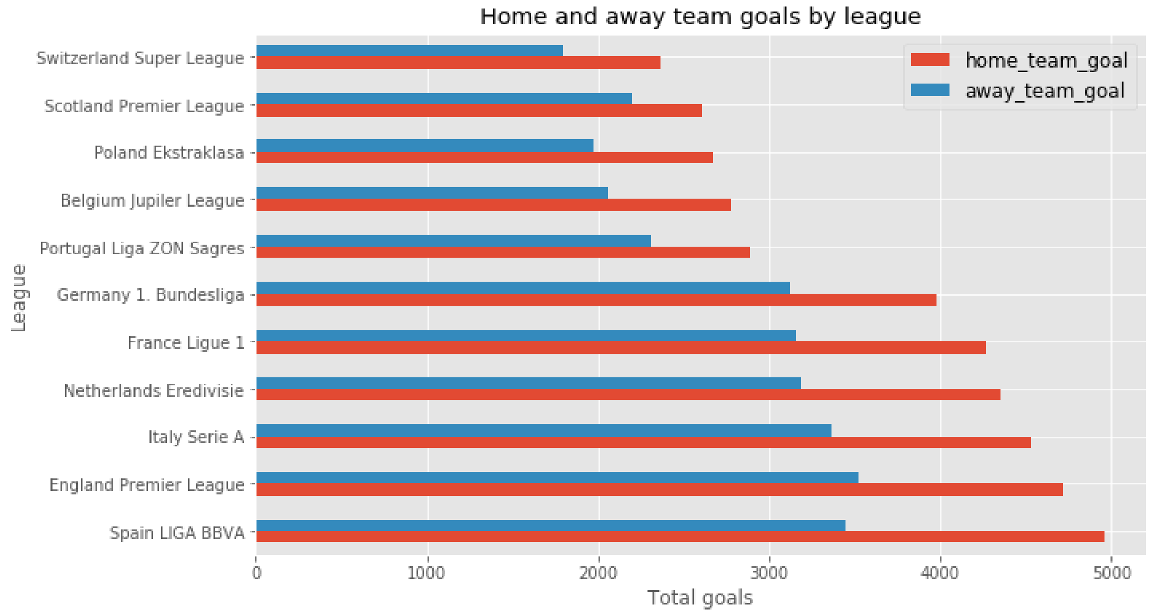
```
count    25979.000000
mean         1.160938
std          1.142110
min          0.000000
25%          0.000000
50%          1.000000
75%          2.000000
max          9.000000
Name: away_team_goal, dtype: float64
```

The initial histograms and summary statistics indicate that, on average, the home team scores 1.54 goals per game whilst the away team scores 1.16 goals per game, across the major European leagues. This indicates that there is some kind of home advantage.

In [26]:

```
# Plot of home and away team goals by league

goals = match_clean.groupby('league_name').agg({"home_team_goal":"sum","away_tea
m_goal":"sum"}).sort_values(['home_team_goal'],ascending=False)
goals.plot(kind="barh", figsize = (10,6))
plt.title("Home and away team goals by league")
plt.style.use('ggplot') # playing around with styles
plt.legend(loc = "best" , prop = {"size" : 12})
plt.xlabel("Total goals")
plt.ylabel("League")
plt.show();
```



This plot shows that the home advantage exists across each league (to varying degrees).

The plot also shows that the top three leagues with highest home (and away goals) scored are:

- Spain LIGA BBVA
- England Premier League
- Italy Serie A

In [27]:

```
# Plot home goals only

home_league_goals = match_clean.groupby('league_name')[['home_team_goal']].sum()
.sum(axis=1).sort_values(ascending=False)
home_league_goals
```

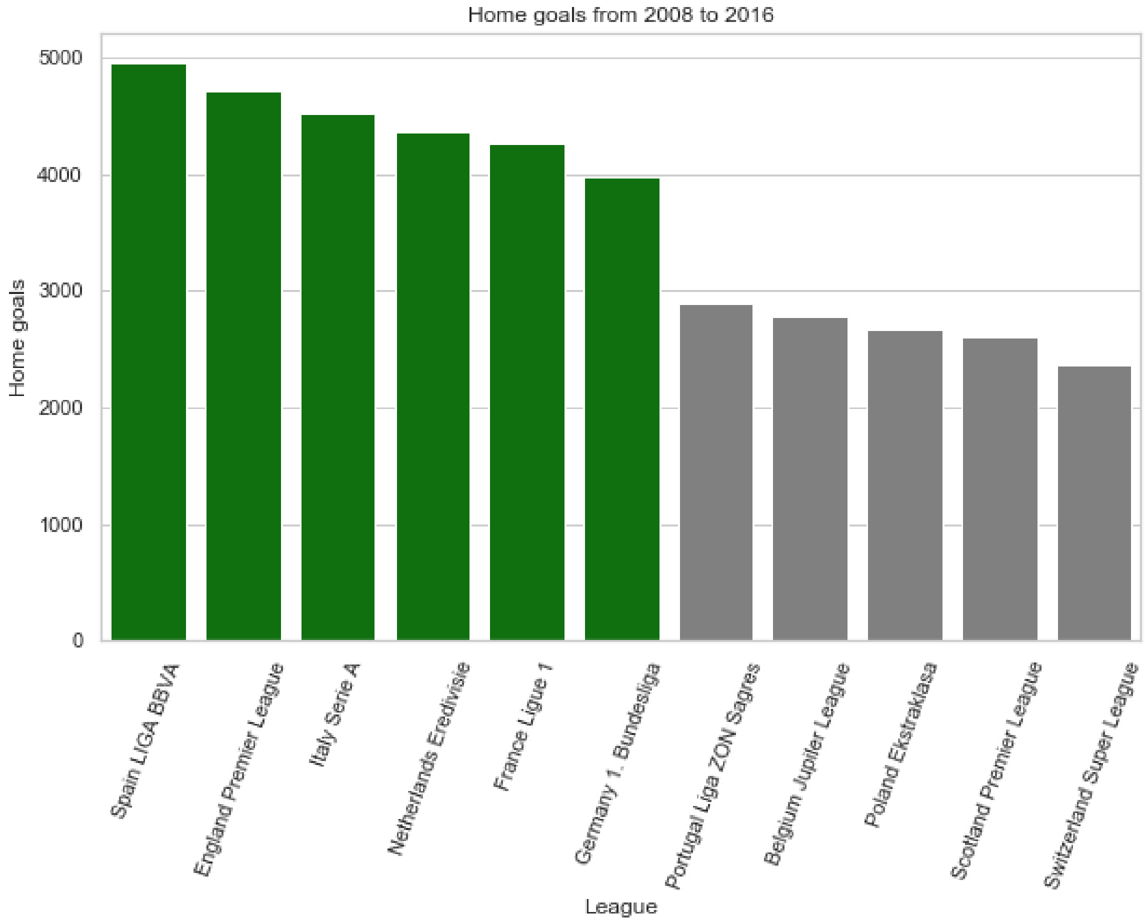
Out[27]:

```
league_name
Spain LIGA BBVA             4959
England Premier League      4715
Italy Serie A               4528
Netherlands Eredivisie      4357
France Ligue 1              4265
Germany 1. Bundesliga       3982
Portugal Liga ZON Sagres    2890
Belgium Jupiler League      2781
Poland Ekstraklasa          2678
Scotland Premier League     2607
Switzerland Super League    2365
dtype: int64
```

In [33]:

```
# Using Seaborn to plot the results

sns.set(style="whitegrid")
h_labels = home_league_goals.index
h_values = home_league_goals
h_mean = h_values.mean()
colours = ['grey' if (x < h_mean) else 'green' for x in h_values] # Green shadin
g for above average results
plt.figure(figsize=(10,6))
sns.barplot(x=h_labels,y=h_values, palette=colours)
plt.title('Home goals from 2008 to 2016')
plt.xticks(rotation=70)
plt.xlabel('League')
plt.ylabel('Home goals');
```



Here we can see a stark difference between two subgroups across the 11 leagues (green shade are above average, and grey shade are below average). This makes me wonder whether there are an even amount of matches played across all leagues.

In [34]:

```
# Plot away goals only

away_league_goals = match_clean.groupby('league_name')[['away_team_goal']].sum()
.sum(axis=1).sort_values(ascending=False)
away_league_goals
```

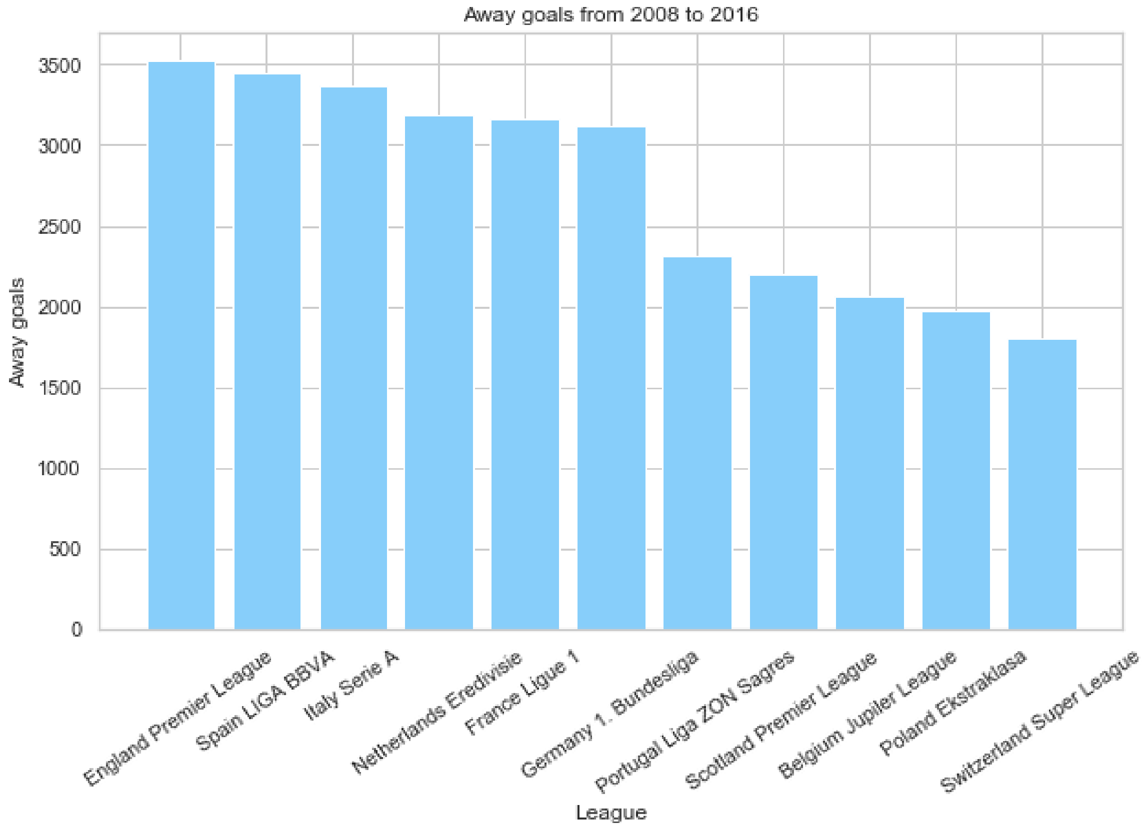
Out[34]:

```
league_name
England Premier League      3525
Spain LIGA BBVA             3453
Italy Serie A               3367
Netherlands Eredivisie      3185
France Ligue 1              3162
Germany 1. Bundesliga       3121
Portugal Liga ZON Sagres    2311
Scotland Premier League     2197
Belgium Jupiler League      2060
Poland Ekstraklasa          1978
Switzerland Super League    1801
dtype: int64
```

In [35]:

```
# Using matplotlib to plot the results

sns.set(style="whitegrid")
a_labels = away_league_goals.index
a_values = away_league_goals
a_mean = a_values.mean()
plt.figure(figsize=(10,6))
plt.bar(x=a_labels,height=a_values, color='lightskyblue')
plt.title('Away goals from 2008 to 2016')
plt.xticks(rotation=35)
plt.xlabel('League')
plt.ylabel('Away goals');
```



We can see a similar pattern emerging for away goals. Here I decided to use a plt bar plot.

In [36]:

```
# Find the number of matches played per league

matches_played = match_clean.groupby('league_name')
matches_played.id.count().sort_values(ascending=False)
```

Out[36]:

```
league_name
Spain LIGA BBVA             3040
France Ligue 1              3040
England Premier League      3040
Italy Serie A               3017
Netherlands Eredivisie      2448
Germany 1. Bundesliga       2448
Portugal Liga ZON Sagres    2052
Poland Ekstraklasa          1920
Scotland Premier League     1824
Belgium Jupiler League      1728
Switzerland Super League    1422
Name: id, dtype: int64
```

To determine whether the previous plot provides a true indication of which leagues have the highest goals scored, I decided to check whether the number of matches played was the same across the leagues.

As per the above output, there is a significant difference across the leagues in terms of total matches played. This would make the previous plot of just relying on total goals misleading if seeking to compare goals scored across leagues.

In [37]:

```
# Calculate mean goals scored per match per league

mean_goals = match_clean[['league_name','home_team_goal','away_team_goal']].grou
pby('league_name').mean().sort_values('home_team_goal', ascending=False)
```

In [38]:

```
mean_goals
```

Out[38]:

league_name	home_team_goal	away_team_goal
Netherlands Eredivisie	1.779820	1.301062
Switzerland Super League	1.663150	1.266526
Spain LIGA BBVA	1.631250	1.135855
Germany 1. Bundesliga	1.626634	1.274918
Belgium Jupiler League	1.609375	1.192130
England Premier League	1.550987	1.159539
Italy Serie A	1.500829	1.116009
Scotland Premier League	1.429276	1.204496
Portugal Liga ZON Sagres	1.408382	1.126218
France Ligue 1	1.402961	1.040132
Poland Ekstraklasa	1.394792	1.030208

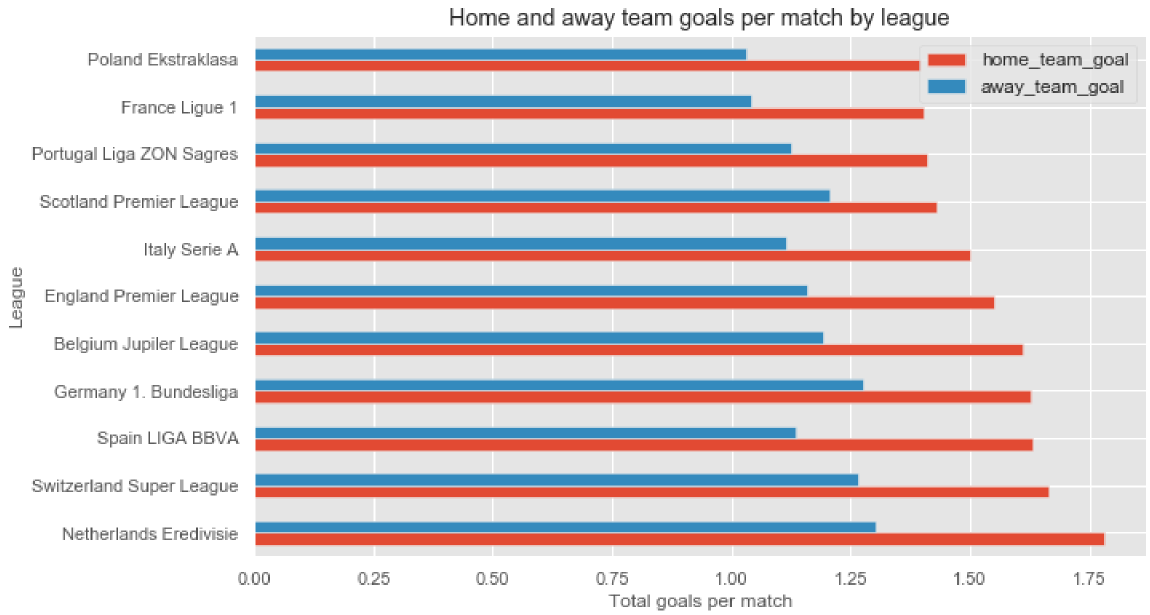
By calculating mean goals scored per game, we see a different story emerging. If we focus on home team goals, the top three leagues are now:

- Netherlands Eredivisie
- Switzerland Super League
- Spain LIGA BBVA

In [40]:

```
# Plot of home and away team goals by league

mean_goals.plot(kind="barh", figsize = (10,6))
plt.title("Home and away team goals per match by league")
plt.style.use('ggplot')
plt.legend(loc = "best" , prop = {"size" : 12})
plt.xlabel("Total goals per match")
plt.ylabel("League");
```



In [41]:

```
# Calculate ratio of home to away goals per league to get a % breakdown

mean_total_goals = mean_goals['home_team_goal'] + mean_goals['away_team_goal']

mean_goals['home_team_goal_percent'] = 100 * mean_goals['home_team_goal'] / mean
_total_goals
mean_goals['away_team_goal_percent'] = 100 * mean_goals['away_team_goal'] / mean
_total_goals

mean_goals_percent = mean_goals[['home_team_goal_percent','away_team_goal_percen
t']].sort_values('home_team_goal_percent', ascending=False)
```

In [42]:

```
mean_goals_percent
```

Out[42]:

league_name	home_team_goal_percent	away_team_goal_percent
Spain LIGA BBVA	58.951498	41.048502
Netherlands Eredivisie	57.769822	42.230178
Poland Ekstraklasa	57.517182	42.482818
Belgium Jupiler League	57.446809	42.553191
France Ligue 1	57.425609	42.574391
Italy Serie A	57.352755	42.647245
England Premier League	57.220874	42.779126
Switzerland Super League	56.769083	43.230917
Germany 1. Bundesliga	56.060819	43.939181
Portugal Liga ZON Sagres	55.566237	44.433763
Scotland Premier League	54.267277	45.732723

This view is interesting as it shows the extent of home advantage across the leagues on a % basis. The strongest home advantage is in Spain LIGA BBVA (58.95% of goals are home goals); whilst the weakest home advantage is in Portugal (55.57% of goals are home goals).

Research Question 2)

Who are the goal machines and goal leakers?: Who are the best and worst performing home and away teams?

Home team analysis

In [43]:

```
# Find bottom five home teams by home goals scored

low_home_scorers = match_clean.groupby('home_team').home_team_goal.sum().sort_va
lues(ascending=True).head(5)
low_home_scorers
```

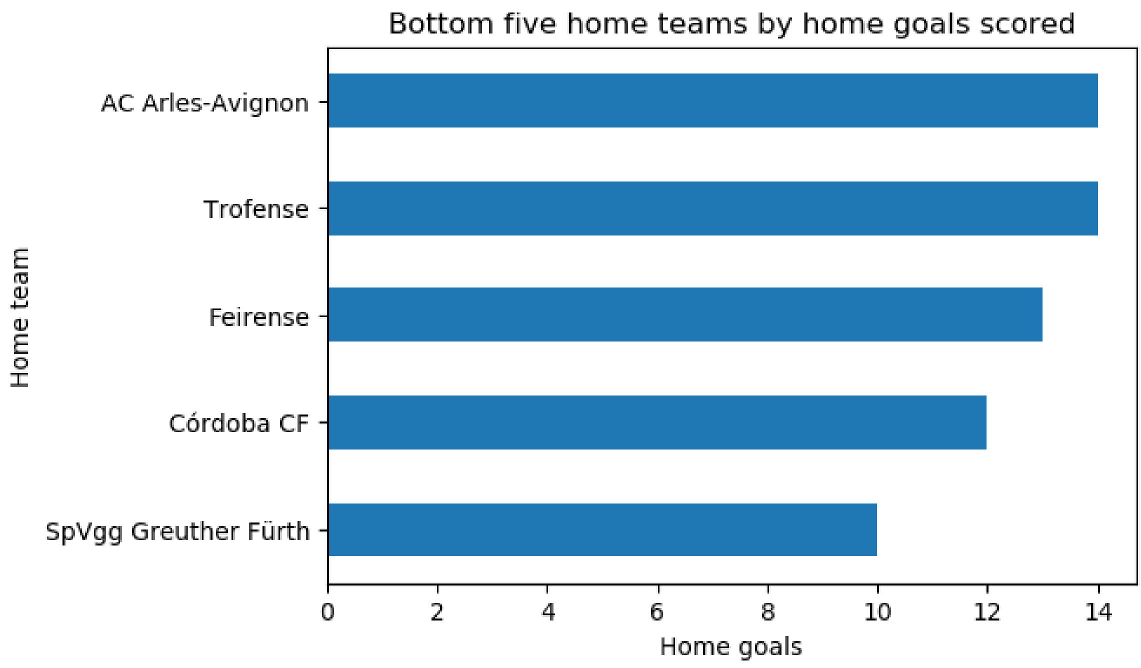
Out[43]:

```
home_team
SpVgg Greuther Fürth     10
Córdoba CF               12
Feirense                 13
Trofense                 14
AC Arles-Avignon         14
Name: home_team_goal, dtype: int64
```

In [49]:

```
# Plot results

low_home_scorers.plot(kind='barh', figsize=(6,4))
plt.style.use('default')
plt.xlabel('Home goals')
plt.ylabel('Home team')
plt.title('Bottom five home teams by home goals scored');
```



In [50]:

```
# Find top five home teams by home goals scored

high_home_scorers = match_clean.groupby('home_team').home_team_goal.sum().sort_v
alues(ascending=False).head(5)
high_home_scorers
```

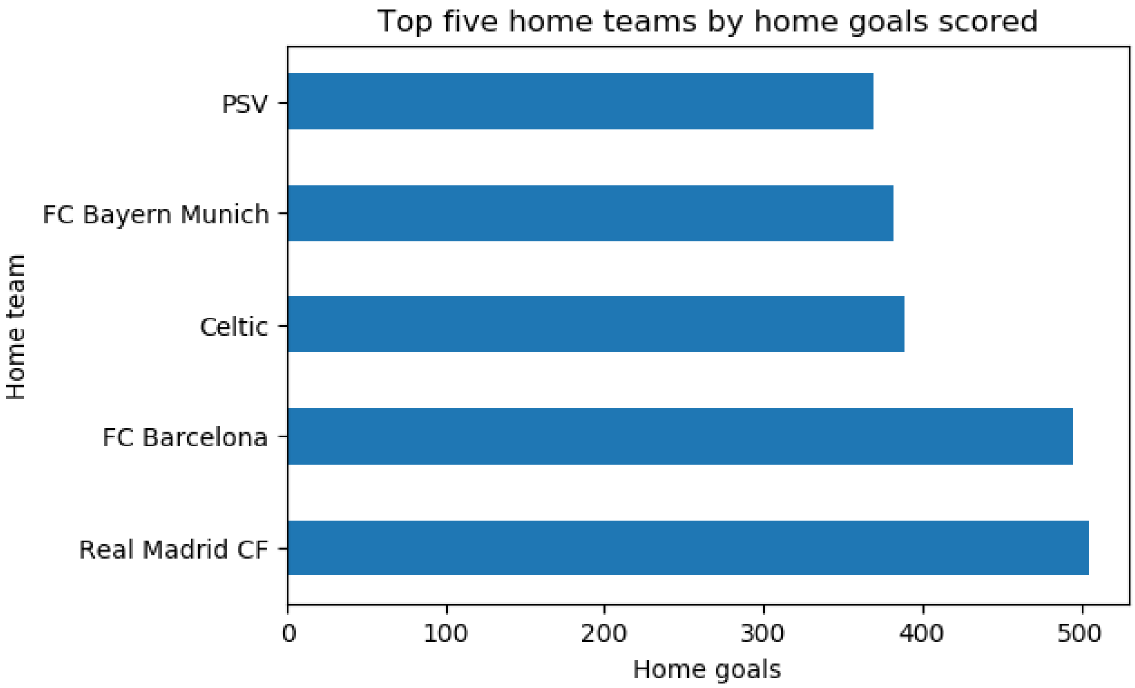
Out[50]:

```
home_team
Real Madrid CF      505
FC Barcelona        495
Celtic              389
FC Bayern Munich    382
PSV                 370
Name: home_team_goal, dtype: int64
```

In [51]:

```
# Plot results

high_home_scorers.plot(kind='barh', figsize=(6,4))
plt.style.use('default')
plt.xlabel('Home goals')
plt.ylabel('Home team')
plt.title('Top five home teams by home goals scored');
```



Away team analysis

In [52]:

```
# Find bottom five away teams by away goals scored

low_away_scorers = match_clean.groupby('away_team').away_team_goal.sum().sort_va
lues(ascending=True).head(5)
low_away_scorers
```

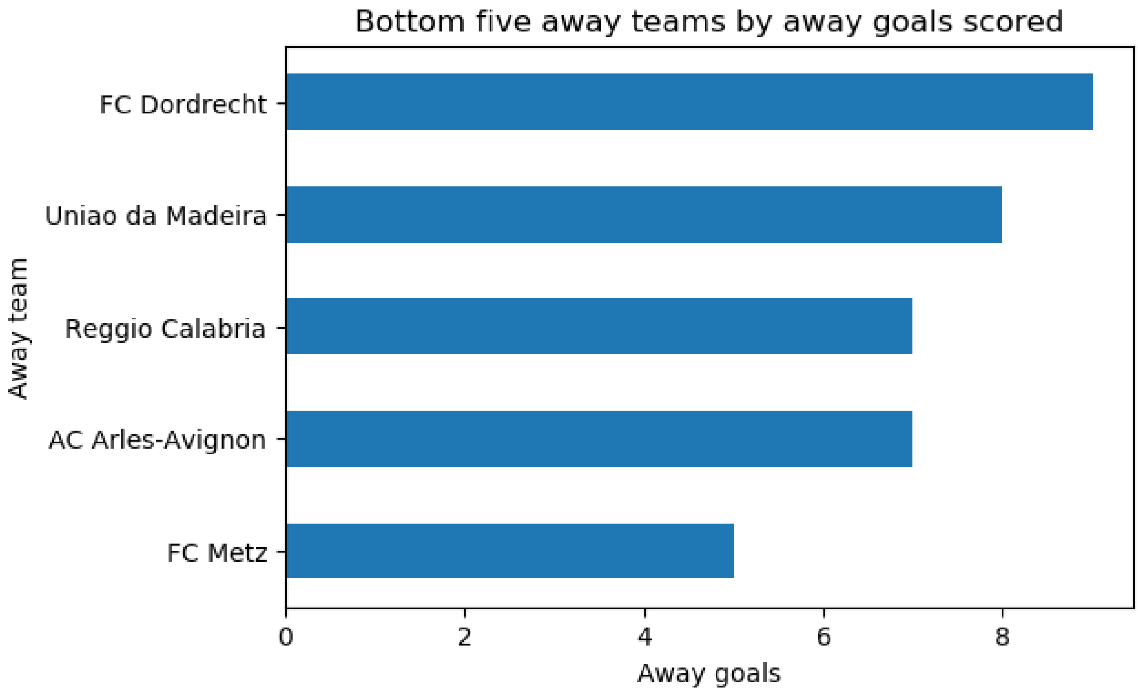
Out[52]:

```
away_team
FC Metz               5
AC Arles-Avignon      7
Reggio Calabria       7
Uniao da Madeira      8
FC Dordrecht          9
Name: away_team_goal, dtype: int64
```

In [53]:

```
# Plot results

low_away_scorers.plot(kind='barh', figsize=(6,4))
plt.style.use('default')
plt.xlabel('Away goals')
plt.ylabel('Away team')
plt.title('Bottom five away teams by away goals scored');
```



In [56]:

```
# Find top five away teams by away goals scored

high_away_scorers = match_clean.groupby('away_team').away_team_goal.sum().sort_v
alues(ascending=False).head(5)
high_away_scorers
```

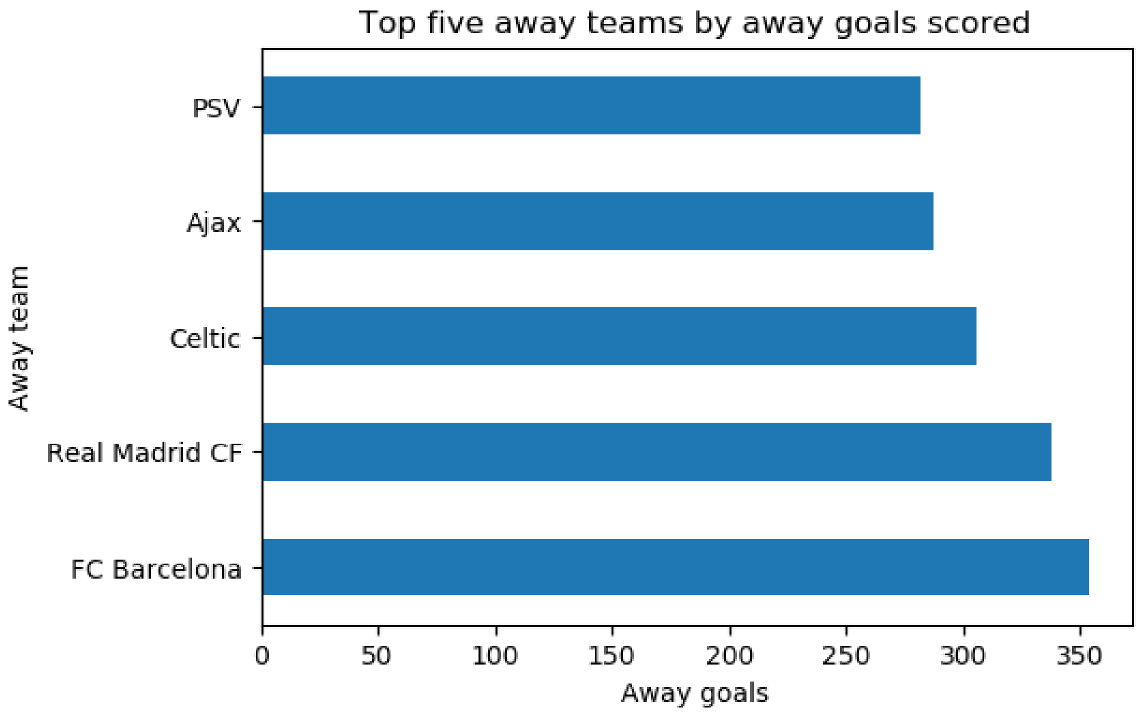
Out[56]:

```
away_team
FC Barcelona      354
Real Madrid CF    338
Celtic            306
Ajax              287
PSV               282
Name: away_team_goal, dtype: int64
```

In [57]:

```
# Plot results

high_home_scorers.plot(kind='barh', figsize=(6,4))
plt.style.use('default')
plt.xlabel('Away goals')
plt.ylabel('Away team')
plt.title('Top five away teams by away goals scored');
```



Research Question 3)

Who runs North London?: Which team in North London has achieved the most wins?

In [66]:

```
# Create a subset of EPL teams only, count the wins over the time period and pro
duce a top 6 ranking
# I expect Arsenal and Tottenham Hotspur to be in the top 6 teams over the refer
ence period

epl = match_clean.query('league_name == "England Premier League"')
top_6 = epl.groupby('outcome')['outcome'].count().sort_values(ascending=False)[1
:7]
```

In [67]:

```
top_6
```

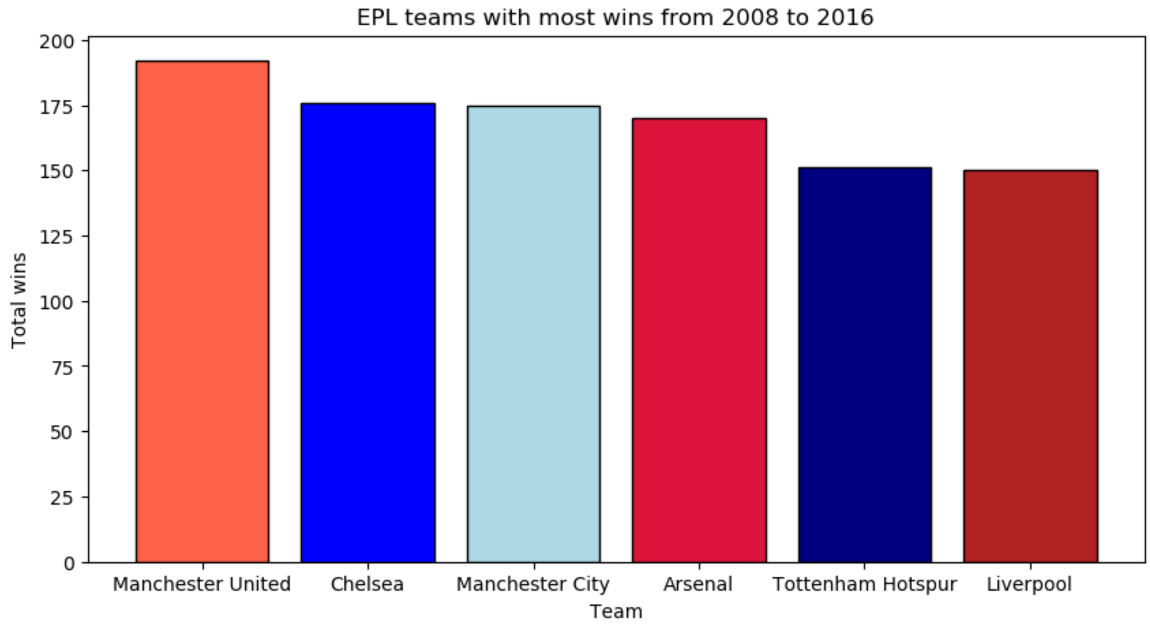
Out[67]:

```
outcome
Manchester United    192
Chelsea              176
Manchester City      175
Arsenal              170
Tottenham Hotspur    151
Liverpool            150
Name: outcome, dtype: int64
```

In [68]:

```
# Plot the results, and add the jersey colour to the graph :D

plt.figure(figsize=(10,5))
plt.bar(top_6.index,top_6,color=['tomato', 'blue', 'lightblue', 'crimson', 'nav
y', 'firebrick'],edgecolor='black')
plt.title('EPL teams with most wins from 2008 to 2016')
plt.xlabel('Team')
plt.ylabel('Total wins');
```



Here, we can see that Arsenal (170) has more wins over Tottenham Hotspur (151) across the reference period, indicating that Arsenal are the more dominant team in North London.

It is interesting to note that Arsenal have a similar number of wins as Chelsea and Manchester City who both have high-profile wealthy ownership which invest significantly in the purchase of big-name players. Arsenal have notoriously been low "spenders" during the reference period, so the fact that Arsenal have maintained a consistent number of wins is admirable (note, this is from an Arsenal fan!).

To extend this study further, it would be interesting to link in transfer spend data and to look at trends across time of both transfer expenditure and team performance.

Research Question 4)

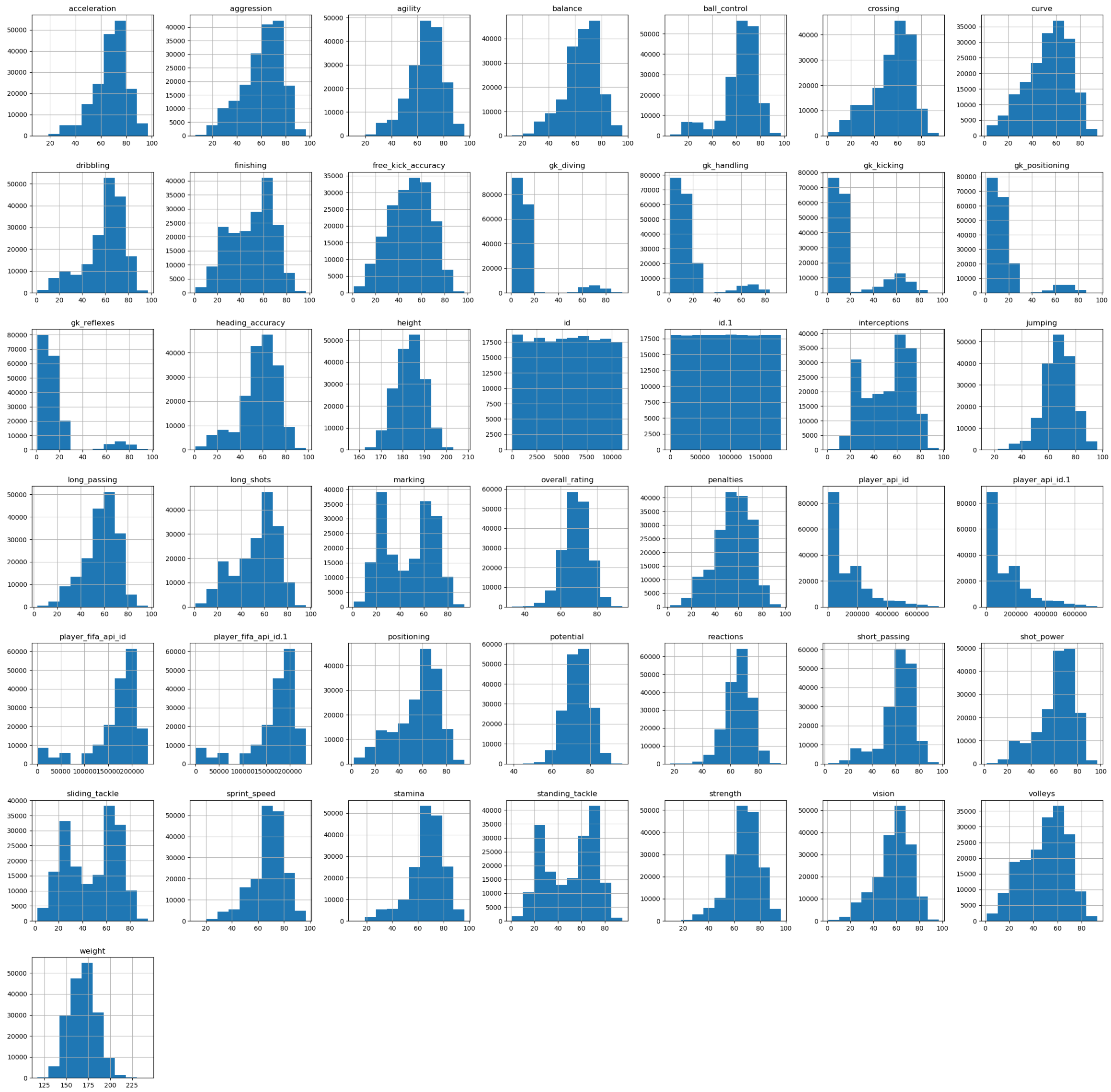
Speedy gonzales: Is pace a factor in a player's overall rating?

High level analysis

In [69]:

```
# Produce histograms for player attributes

player_clean.hist(figsize=(30,30));
```


acceleration
aggression
agility
balance
ball_control
crossing
curve
dribbling
finishing
free_kick_accuracy
gk_diving
gk_handling
gk_kicking
gk_positioning
gk_reflexes
heading_accuracy
height
id
id.1
interceptions
jumping
long_passing
long_shots
marking
overall_rating
penalties
player_api_id
player_api_id.1
player_fifa_api_id
player_fifa_api_id.1
positioning
potential
reactions
short_passing
shot_power
sliding_tackle
sprint_speed
stamina
standing_tackle
strength
vision
volleys
weight

In [70]:

```
# Produce a correlation matrix for player attributes

corr = player_clean.corr()
corr
```

Out[70]:

	id	player_api_id	player_fifa_api_id	height	weight	id.1
id	1.000000	0.002181	0.003607	0.009247	0.012342	0.999989
player_api_id	0.002181	1.000000	0.556736	-0.065485	-0.161037	0.002048
player_fifa_api_id	0.003607	0.556736	1.000000	-0.040756	-0.121283	0.003499
height	0.009247	-0.065485	-0.040756	1.000000	0.762176	0.009334
weight	0.012342	-0.161037	-0.121283	0.762176	1.000000	0.012384
id.1	0.999989	0.002048	0.003499	0.009334	0.012384	1.000000
player_fifa_api_id.1	0.003853	0.556557	0.999856	-0.040624	-0.121140	0.003744
player_api_id.1	0.002181	1.000000	0.556736	-0.065485	-0.161037	0.002048
overall_rating	-0.003743	-0.328315	-0.278784	-0.004651	0.063945	-0.003738
potential	0.000878	0.010588	-0.021330	-0.035375	-0.009193	0.000837
crossing	-0.020163	-0.113365	-0.065588	-0.473698	-0.411654	-0.020231
finishing	-0.008150	-0.062312	-0.029890	-0.320104	-0.254560	-0.008171
heading_accuracy	-0.011851	-0.130282	-0.103220	0.111904	0.097846	-0.011781
short_passing	-0.006653	-0.090237	-0.065161	-0.364106	-0.326018	-0.006701
volleys	-0.006932	-0.131262	-0.088748	-0.335513	-0.261252	-0.006916
dribbling	-0.014719	0.015616	0.047592	-0.490154	-0.432850	-0.014784
curve	-0.019478	-0.099430	-0.052564	-0.452054	-0.387934	-0.019523
free_kick_accuracy	-0.008371	-0.152683	-0.108860	-0.383592	-0.314920	-0.008396
long_passing	-0.008114	-0.139584	-0.111083	-0.297312	-0.269579	-0.008137
ball_control	-0.013937	-0.053940	-0.024812	-0.419595	-0.370126	-0.013976
acceleration	-0.008123	0.101536	0.178265	-0.522063	-0.462094	-0.008212
sprint_speed	-0.011803	0.094236	0.178306	-0.430250	-0.385437	-0.011897
agility	-0.000849	0.026467	0.116158	-0.618819	-0.556005	-0.000947
reactions	-0.005750	-0.312538	-0.233568	-0.082576	-0.019009	-0.005740
balance	-0.009859	0.021300	0.008198	-0.672173	-0.555836	-0.009909
shot_power	-0.010374	-0.126514	-0.079996	-0.250186	-0.170661	-0.010371
jumping	-0.004316	-0.141646	-0.073014	-0.003158	0.042218	-0.004279
stamina	-0.010458	-0.109958	0.015467	-0.236377	-0.212975	-0.010506
strength	-0.009025	-0.234866	-0.178140	0.519012	0.563952	-0.008954
long_shots	-0.010358	-0.119638	-0.068653	-0.361112	-0.288412	-0.010382
aggression	-0.018067	-0.212509	-0.169974	0.028850	0.067334	-0.018034
interceptions	-0.008485	-0.185482	-0.168939	0.001858	-0.007965	-0.008480
positioning	-0.015610	-0.105157	-0.078818	-0.374485	-0.305190	-0.015643
vision	-0.007866	-0.188087	-0.163124	-0.383292	-0.323229	-0.007928
penalties	-0.011783	-0.162481	-0.175265	-0.270864	-0.194014	-0.011751
marking	-0.010330	-0.089772	-0.075197	0.045017	0.016453	-0.010329

	id	player_api_id	player_fifa_api_id	height	weight	id.1
standing_tackle	-0.012523	-0.086706	-0.070792	0.025966	0.001163	-0.012515
sliding_tackle	-0.011132	-0.073595	-0.054939	0.011284	-0.014010	-0.011101
gk_diving	0.014246	-0.071825	-0.093191	0.315629	0.311371	0.014251
gk_handling	0.010887	-0.125345	-0.139063	0.309854	0.311752	0.010911
gk_kicking	0.008686	-0.229704	-0.248324	0.191084	0.211216	0.008758
gk_positioning	0.013986	-0.125525	-0.141165	0.309086	0.310953	0.014015
gk_reflexes	0.014652	-0.121947	-0.131792	0.313144	0.312210	0.014671

43 rows × 43 columns

In [71]:

```
# Produce descriptive statistics for player attributes

player_clean.describe().transpose()
```

Out[71]:

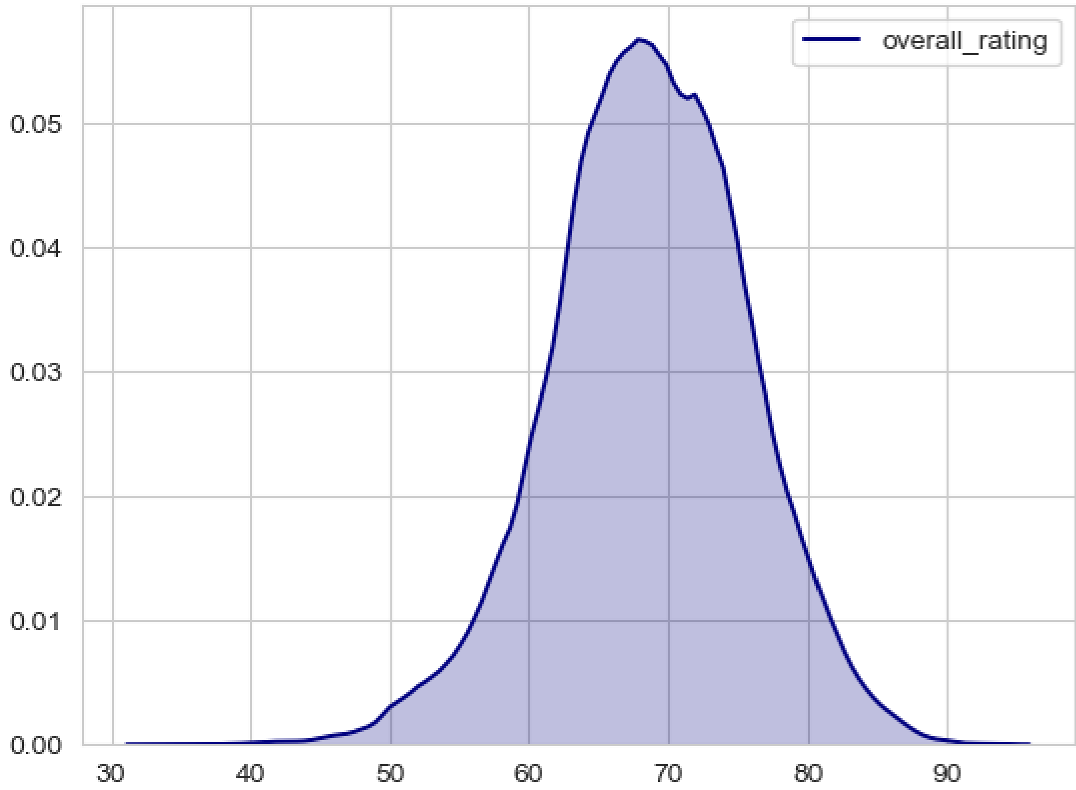
	count	mean	std	min	25%	50%	
id	180354.0	5520.609207	3190.378836	1.00	2758.00	5533.00	
player_api_id	180354.0	137653.145514	137599.735284	2625.00	35451.00	80291.00	1
player_fifa_api_id	180354.0	166805.312530	52825.971635	2.00	156593.00	183774.00	2
height	180354.0	181.877895	6.398588	157.48	177.80	182.88	
weight	180354.0	168.774593	15.098388	117.00	159.00	168.00	
id.1	180354.0	91995.886274	53092.657914	1.00	46074.25	92003.50	1
player_fifa_api_id.1	180354.0	166822.125803	52821.443279	2.00	156616.00	183792.00	2
player_api_id.1	180354.0	137653.145514	137599.735284	2625.00	35451.00	80291.00	1
overall_rating	180354.0	68.635317	7.027950	33.00	64.00	69.00	
potential	180354.0	73.479457	6.581963	39.00	69.00	74.00	
crossing	180354.0	55.142071	17.247231	1.00	45.00	59.00	
finishing	180354.0	49.962136	19.041760	1.00	34.00	53.00	
heading_accuracy	180354.0	57.263476	16.478716	1.00	49.00	60.00	
short_passing	180354.0	62.486726	14.172493	3.00	57.00	65.00	
volleys	180354.0	49.488927	18.252319	1.00	35.00	52.00	
dribbling	180354.0	59.265755	17.741351	1.00	52.00	64.00	
curve	180354.0	53.001408	18.245476	2.00	41.00	56.00	
free_kick_accuracy	180354.0	49.392783	17.820262	1.00	36.00	50.00	
long_passing	180354.0	57.084578	14.412035	3.00	49.00	59.00	
ball_control	180354.0	63.453846	15.187692	5.00	59.00	67.00	
acceleration	180354.0	67.709405	13.011580	10.00	61.00	69.00	
sprint_speed	180354.0	68.101628	12.585984	12.00	62.00	69.00	
agility	180354.0	65.995082	12.963670	11.00	58.00	68.00	
reactions	180354.0	66.148297	9.145011	17.00	61.00	67.00	
balance	180354.0	65.190082	13.076192	12.00	58.00	67.00	
shot_power	180354.0	61.866474	16.129537	2.00	54.00	66.00	
jumping	180354.0	66.977333	11.017828	14.00	60.00	68.00	
stamina	180354.0	67.053401	13.200669	10.00	61.00	69.00	
strength	180354.0	67.432477	12.085131	10.00	60.00	69.00	
long_shots	180354.0	53.387560	18.370204	1.00	41.00	58.00	
aggression	180354.0	60.946217	16.101618	6.00	51.00	64.00	
interceptions	180354.0	51.897374	19.483338	1.00	34.00	56.00	
positioning	180354.0	55.730730	18.458218	2.00	45.00	60.00	
vision	180354.0	57.868176	15.152408	1.00	49.00	60.00	
penalties	180354.0	54.933448	15.556645	2.00	45.00	57.00	
marking	180354.0	46.757433	21.226730	1.00	25.00	50.00	

	count	mean	std	min	25%	50%
standing_tackle	180354.0	50.354065	21.496289	1.00	29.00	56.00
sliding_tackle	180354.0	48.029342	21.592830	2.00	25.00	53.00
gk_diving	180354.0	14.696685	16.841454	1.00	7.00	10.00
gk_handling	180354.0	15.947786	15.841297	1.00	8.00	11.00
gk_kicking	180354.0	20.526304	21.143898	1.00	8.00	12.00
gk_positioning	180354.0	16.015043	16.070772	1.00	8.00	11.00
gk_reflexes	180354.0	16.325310	17.185450	1.00	8.00	11.00

In [72]:

```
# Plot the distribution of overall player rating

sns.set_style("whitegrid")
sns.kdeplot(player.overall_rating, shade=True, color="navy");
```



In [73]:

```
# Produce descriptive statistics for overall rating only

player_clean.overall_rating.describe()
```

Out[73]:

```
count    180354.000000
mean         68.635317
std           7.027950
min          33.000000
25%          64.000000
50%          69.000000
75%          73.000000
max          94.000000
Name: overall_rating, dtype: float64
```

Analysis of pace and overall rating

There are two key metrics that could be used as a proxy for pace. These are 'acceleration' and 'sprint_speed'. First I would like to check whether they are highly correlated (as would be my expectation), and then I will use one to compare with 'overall_rating'.

In [76]:

```
# Correlation between acceleration and sprint_speed

player_clean['acceleration'].corr(player['sprint_speed'])
```

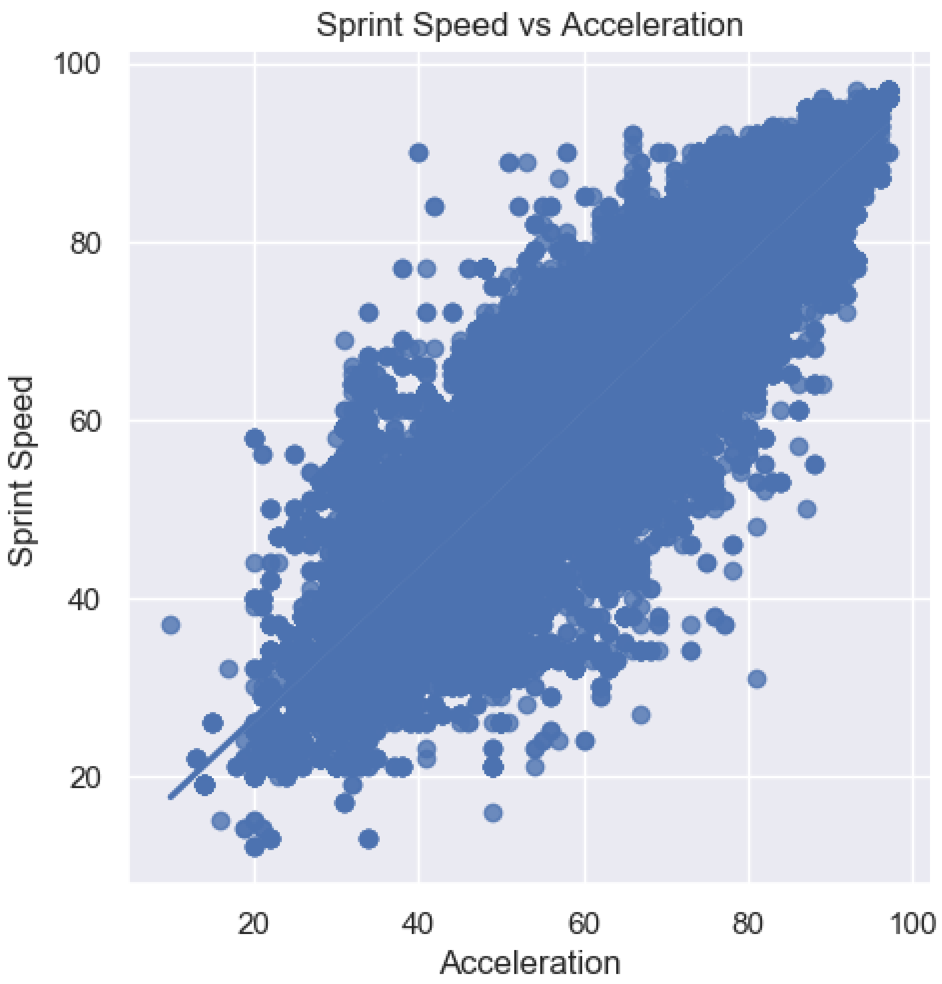
Out[76]:

```
0.9050435815597864
```

In [77]:

```
# Plot scatter plot results

sns.set()
sns.lmplot(x='acceleration',y='sprint_speed',data=player_clean)
plt.title('Sprint Speed vs Acceleration')
plt.xlabel('Acceleration')
plt.ylabel('Sprint Speed');
```



There is a high correlation between 'sprint_speed' and 'acceleration'. I will choose to use 'sprint_speed' for the analysis with 'overall_rating'.

In [78]:

```
# Correlation between overall_rating and sprint_speed

player_clean['overall_rating'].corr(player['sprint_speed'])
```

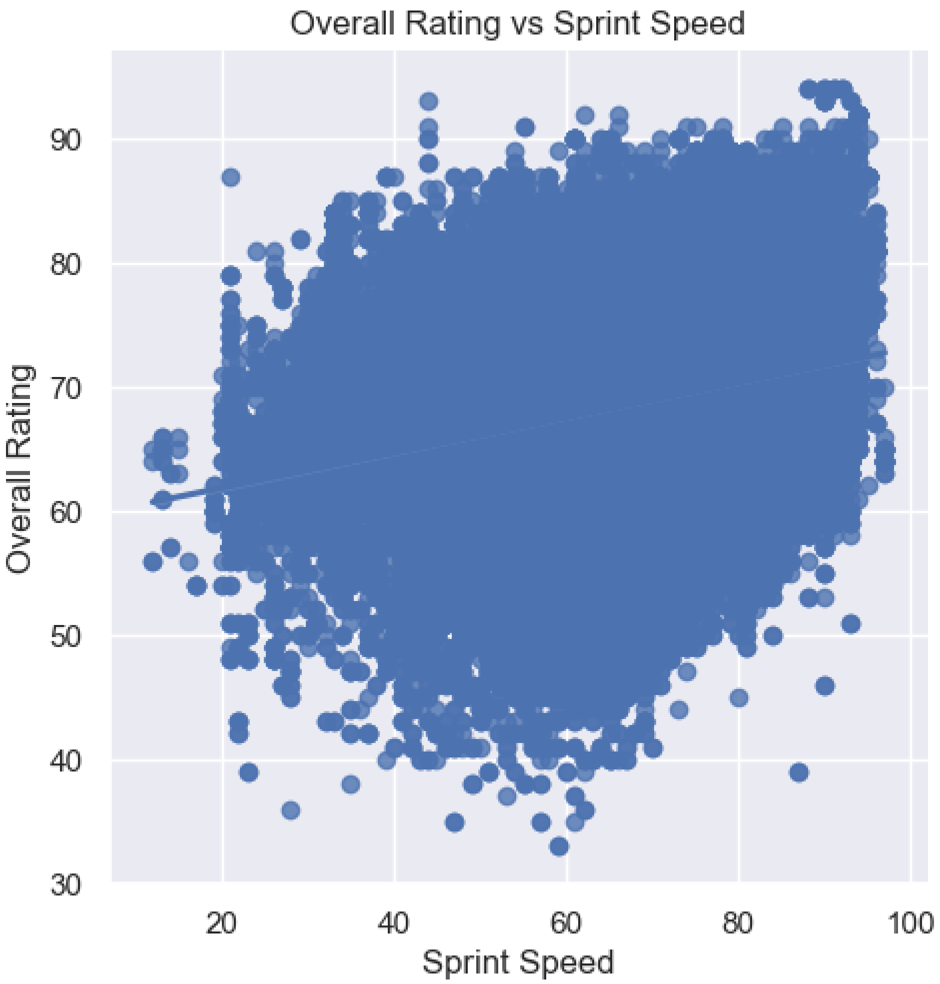
Out[78]:

```
0.25304806290254583
```

In [79]:

```
# Plot scatter plot results

sns.set()
sns.lmplot(x='sprint_speed',y='overall_rating',data=player_clean)
plt.title('Overall Rating vs Sprint Speed')
plt.xlabel('Sprint Speed')
plt.ylabel('Overall Rating');
```



There appears to be a weak positive correlation between 'sprint_speed' and 'overall_rating'. I suspect that there may be some value in extending this study to differentiate between defensive and offensive players, as my hypothesis would be such that there would be a stronger correlation between 'sprint_speed' and 'overall_rating' for offensive players where running at speed is a valuable attribute, than for defensive players.

Conclusions

1. **Home fortress?**: Does a home advantage exist, and if so, which league has the strongest such advantage?

> A home advantage does exist across all leagues. On a goals per match basis, the advantage is strongest in Spain LIGA BBVA (58.95% of goals are scored by the home team).

1. **Who are the goal machines and goal leakers?**: What are the best and worst performing home and away teams?

> - Lowest home scorer: SpVgg Greuther Fürth (10 home goals)
> - Highest home scorer: Real Madrid (505 home goals)
> - Lowest away scorer: FC Metz (5 away goals)
> - Highest away scorer: FC Barcelona (354 away goals)
>
> Note, the lowest scoring teams should be taken with a grain of salt as these may be newly promoted teams who have not been in the top league for an extended period of time.

1. **Who runs North London?**: Which team in North London has achieved the most wins?

> As an Arsenal fan, it gives me great pleasure to report that Arsenal is the most dominant team in North London with 170 wins to Tottenham Hotspur's 151 wins.

1. **Speedy gonzales**: Is pace a factor in a player's overall rating?

> Whilst there is a slightly positive correlation between sprint speed and overall rating, it does not appear to be very strong.
>
> To extend this study further, it would be useful to segment between defensive and offensive players, as my hypothesis would be that offensive players with pace may have a higher overall rating, whilst sprint speed may not be as relevant to defensive players.