Project: Investigate a Dataset (Soccer Database)

Table of Contents

Introduction

> I investigate the Soccer dataset. Mainly, the dataset have 7 tables called 'Country', 'League', 'Match', 'Player', 'Player Attributes', 'Team' and 'Team Attributes'. the dataset contains useful data about 11 seasons between 2008 and 2016 in different leagues and a list of (players, teams) attributes
>
> ### During the analysis of the dataset I wanna focus answer these questions:

- How many matches are there in each league in the 2016 season?
- Which League had the most matches end as draw in the 2016 season?
- Which League had the most Wins or not Draw in the 2016 season?
- Which team had lost the fewest matches in the 2016 season?
- Which League had the most goals in the 2016 season?
- Which teams had the most wins of matches in the 2016 season?
- What teams improved the most over the time period?
- Which players had the most penalties?
- What team attributes lead to the most victories?
- How many Players have overall rating more than 90?
- What are the attributes that contribute to the players' overall rating?

In [1]:
```
# import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
% matplotlib inline
```

```
UsageError: Line magic function `%` not found.
```

Data Wrangling

- General Properties

In [2]:
```
# Load Data that I will investigate it
df_match = pd.read_csv('Match.csv')
df_player_attribute = pd.read_csv('Player_Attributes.csv')
```

```
df_team_attribute = pd.read_csv('Team_Attributes.csv')
df_team = pd.read_csv('Team.csv')
df_player = pd.read_csv('Player.csv')
df_country = pd.read_csv('Country.csv')
df_league = pd.read_csv('League.csv')
```

In [3]:
```
#Sample of Mactch Table
df_match.head()
```

Out[3]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	away_tea
0	1	1	1	2008/2009	1	2008-08-17 00:00:00	492473	9987	
1	2	1	1	2008/2009	1	2008-08-16 00:00:00	492474	10000	
2	3	1	1	2008/2009	1	2008-08-16 00:00:00	492475	9984	
3	4	1	1	2008/2009	1	2008-08-17 00:00:00	492476	9991	
4	5	1	1	2008/2009	1	2008-08-16 00:00:00	492477	7947	

5 rows × 115 columns

In [4]:
```
#Columns in the match table
df_match.columns
```

Out[4]:
```
Index(['id', 'country_id', 'league_id', 'season', 'stage', 'date',
       'match_api_id', 'home_team_api_id', 'away_team_api_id',
       'home_team_goal',
       ...
       'SJA', 'VCH', 'VCD', 'VCA', 'GBH', 'GBD', 'GBA', 'BSH', 'BSD', 'BSA'],
      dtype='object', length=115)
```

In [5]:
```
# Number of Rows, Columns
df_match.shape
```

Out[5]:
```
(25979, 115)
```

In [6]:
```
# information about Match table
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
```

In [7]:
```
# Number of Duplicated Records in Match table
df_match.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
# Number of NULL values in Match table
```

```
df_match.isna().sum().sum()
```

Out[8]: 407395

Match Table:-

```
Contain 25979 Records & 115 Columns.
No duplicate records
Has a lot of missing values (407395) but all null values in columns I
won't need in processes, So I'll drop it.
```

In [9]:
```
#Sample of Country Table
df_country
```

Out[9]:

	id	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

In [10]:
```
# Number of Rows, Columns
df_country.shape
```

Out[10]: (11, 2)

In [11]:
```
# information about country table
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
```

Country Table:-

```
Contain 11 Records & 2 Columns.
No duplicate records
No missing values.
```

In [12]:
```
#Sample of team Table
df_team.head()
```

Out[12]:

	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

In [13]:
```
# Number of Rows, Columns
df_team.shape
```

Out[13]:
```
(299, 5)
```

In [14]:
```
# information about team table
df_team.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
```

In [15]:
```
# Number of Duplicated Records team table
df_team.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
# Number of NULL values in each table in team table
df_team.isna().sum()
```

Out[16]:
```
id                   0
team_api_id          0
team_fifa_api_id    11
team_long_name       0
team_short_name      0
dtype: int64
```

In [17]:
```
# Number of NULL values in team table
df_team.isna().sum().sum()
```

Out[17]:
```
11
```

Team Table:-

```
Contain 299 Records & 5 Columns.
No duplicate records
Has missing values(11) but all null values in team_fifa_api_id column.
```

In [18]:
```
#Sample of team attribute Table
df_team_attribute.head()
```

Out[18]:

id	team_fifa_api_id	team_api_id	date	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpPla

	id	team_fifa_api_id	team_api_id	date	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpPla
0	1	434	9930	2010-02-22 00:00:00	60	Balanced	
1	2	434	9930	2014-09-19 00:00:00	52	Balanced	
2	3	434	9930	2015-09-10 00:00:00	47	Balanced	
3	4	77	8485	2010-02-22 00:00:00	70	Fast	
4	5	77	8485	2011-02-22 00:00:00	47	Balanced	

5 rows × 25 columns

In [19]:
```
# Number of Rows, Columns
df_team_attribute.shape
```

Out[19]:
```
(1458, 25)
```

In [20]:
```
# information about team attribute table
df_team_attribute.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 [21]:
```
# Number of Duplicated Records team attribute table
df_team_attribute.duplicated().sum()
```

Out[21]:
```
0
```

In [22]:
```
# Number of NULL values in each table in team attribute table
df_team_attribute.isna().sum()
```

Out[22]:
```
id                                  0
team_fifa_api_id                    0
team_api_id                         0
date                                0
buildUpPlaySpeed                    0
buildUpPlaySpeedClass               0
buildUpPlayDribbling              969
buildUpPlayDribblingClass           0
buildUpPlayPassing                  0
buildUpPlayPassingClass             0
buildUpPlayPositioningClass         0
chanceCreationPassing               0
chanceCreationPassingClass          0
chanceCreationCrossing              0
chanceCreationCrossingClass         0
chanceCreationShooting              0
chanceCreationShootingClass         0
chanceCreationPositioningClass      0
defencePressure                     0
defencePressureClass                0
defenceAggression                   0
defenceAggressionClass              0
defenceTeamWidth                    0
defenceTeamWidthClass               0
defenceDefenderLineClass            0
dtype: int64
```

In [23]:
```
# Number of NULL values in team attribute table
df_team_attribute.isna().sum().sum()
```

Out[23]:
```
969
```

Team Table:-

```
Contain 1458 Records & 25 Columns.
No duplicate records
Has a lot of missing values(969) but all null values in
buildUpPlayDribbling colums, So I'll drop it.
```

In [24]:
```
#Sample of league Table
df_league
```

Out[24]:

	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
5	13274	13274	Netherlands Eredivisie
6	15722	15722	Poland Ekstraklasa

	id	country_id	name
7	17642	17642	Portugal Liga ZON Sagres
8	19694	19694	Scotland Premier League
9	21518	21518	Spain LIGA BBVA
10	24558	24558	Switzerland Super League

In [25]:
```
# Number of Rows, Columns
df_league.shape
```

Out[25]:
```
(11, 3)
```

In [26]:
```
# information about league table
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
```

League Table:-

```
Contain 11 Records & 3 Columns.
No duplicate records
No missing values.
```

In [27]:
```
#Sample of player Table
df_player.head()
```

Out[27]:

	id	player_api_id	player_name	player_fifa_api_id	birthday	height	weight
0	1	505942	Aaron Appindangoye	218353	1992-02-29 00:00:00	182.88	187
1	2	155782	Aaron Cresswell	189615	1989-12-15 00:00:00	170.18	146
2	3	162549	Aaron Doran	186170	1991-05-13 00:00:00	170.18	163
3	4	30572	Aaron Galindo	140161	1982-05-08 00:00:00	182.88	198
4	5	23780	Aaron Hughes	17725	1979-11-08 00:00:00	182.88	154

In [28]:
```
# Number of Rows, Columns
df_player.shape
```

Out[28]:
```
(11060, 7)
```

In [29]:
```
# Number of Rows, Columns
df_team_attribute.duplicated().sum()
```

Out[29]:
```
0
```

In [30]:
```
# Number of NULL values in team table
df_team.isna().sum().sum()
```

Out[30]: 11

Player Table:-

```
Contain 11060 Records & 7 Columns.
No duplicate records
Has missing values(11).
```

In [31]:
```
#Sample of player attribute Table
df_player_attribute.head()
```

Out[31]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_v
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	

5 rows × 42 columns

In [32]:
```
# Number of Rows, Columns
df_player_attribute.shape
```

Out[32]: (183978, 42)

In [33]:
```
# information about player attribute table
df_player_attribute.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
```

In [34]:

```
# Number of Duplicated Records player attribute table
df_player_attribute.duplicated().sum()
```

Out[34]:

```
0
```

In [35]:

```
# Number of NULL values in each columns in _player attribute table
df_player_attribute.isna().sum()
```

Out[35]:

```
id                       0
player_fifa_api_id       0
player_api_id            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
```

In [36]:
```
# Number of NULL values in player attribute table
df_player_attribute.isna().sum().sum()
```

Out[36]: 47301

Player Attributes Table:-

```
Contain 183973 Records & 42 Columns.
No duplicate records
Has a lot of missing values(47301).
```

In [ ]:

Data Cleaning (Matches Table)

In [37]:
```
# Select columns which we need in analysis
df_match=df_match.loc[:,:'away_team_goal']
```

In [38]:
```
#Convert Date to DateTime type to gain availability to dedicate a year of each date
df_match['date'] = pd.to_datetime(df_match['date'])
#Show sample
df_match.head()
```

Out[38]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	away_team_
0	1	1	1	2008/2009	1	2008-08-17	492473	9987	
1	2	1	1	2008/2009	1	2008-08-16	492474	10000	
2	3	1	1	2008/2009	1	2008-08-16	492475	9984	
3	4	1	1	2008/2009	1	2008-08-17	492476	9991	
4	5	1	1	2008/2009	1	2008-08-16	492477	7947	

In [39]:
```
# Add new Columns to match table store years
df_match['season_year'] = df_match['date'].dt.year
df_match['season_year'].max()
```

Out[39]: 2016

In [40]:
```
# rename column: name to country_name
```

```
df_country.rename(columns={'name' : 'country_name', }, inplace=True)

# join df_match with country table by inner join type .
df_match = df_match.merge(df_country, how='inner', left_on= "country_id", right_on =

# drop column id_y
df_match.drop(columns=['id_y'], inplace=True)
# rename column: id_x to id
df_match.rename(columns={'id_x' : 'id'}, inplace=True)
#Show sample
df_match.head()
```

Out[40]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	away_team_
0	1	1	1	2008/2009	1	2008-08-17	492473	9987	
1	2	1	1	2008/2009	1	2008-08-16	492474	10000	
2	3	1	1	2008/2009	1	2008-08-16	492475	9984	
3	4	1	1	2008/2009	1	2008-08-17	492476	9991	
4	5	1	1	2008/2009	1	2008-08-16	492477	7947	

In [41]:

```
# join df_match with team table by inner join type for home team
df_match = df_match.merge(df_team, how='inner', left_on='home_team_api_id', right_on
# rename column: team_long_name to home_team_name
df_match.rename(columns={'team_long_name': 'home_team_name','country_name_x'  : 'cou

# drop column home_team_api_id and team_api_id
df_match.drop(columns=['home_team_api_id', 'team_api_id', 'id_y'], axis=1, inplace=T
#Show sample
df_match.head()
```

Out[41]:

	id	country_id	league_id	season	stage	date	match_api_id	away_team_api_id	home_team_
0	1	1	1	2008/2009	1	2008-08-17	492473	9993	
1	29	1	1	2008/2009	12	2008-11-15	492583	9999	
2	47	1	1	2008/2009	14	2008-11-29	492651	9984	
3	65	1	1	2008/2009	16	2008-12-13	492713	9986	
4	94	1	1	2008/2009	19	2009-01-24	492805	9998	

In [42]:

```
# join df_match with team table by left join type for away team
df_match = df_match.merge(df_team, how='left', left_on='away_team_api_id', right_on=
# rename column: team_long_name to away_team_name , id_x to id
df_match.rename(columns={'team_long_name': 'away_team_name', 'id_x' : 'id'}, inplace
# drop unnecessary columns
df_match.drop(columns=['team_api_id', 'away_team_api_id','team_fifa_api_id_x', 'team
```

```
#Show sample
df_match.head()
```

Out[42]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_goal	away_team_g
0	1	1	1	2008/2009	1	2008-08-17	492473	1	
1	29	1	1	2008/2009	12	2008-11-15	492583	1	
2	47	1	1	2008/2009	14	2008-11-29	492651	3	
3	65	1	1	2008/2009	16	2008-12-13	492713	1	
4	94	1	1	2008/2009	19	2009-01-24	492805	2	

In [43]:

```
# rename the two columns 'name' and 'id'
df_league.rename(columns={'name': 'league_name', 'id': 'league_id'}, inplace=True)

# join df_match with league table by inner join type for away team
df_match = df_match.merge(df_league, how='inner', on='league_id')

# drop now country_id and league_id
df_match.drop(columns=["league_id","country_id_y" , "country_id_x"], inplace=True)

#Show sample
df_match.head()
```

Out[43]:

	id	season	stage	date	match_api_id	home_team_goal	away_team_goal	season_year	coun
0	1	2008/2009	1	2008-08-17	492473	1	1	2008	
1	29	2008/2009	12	2008-11-15	492583	1	1	2008	
2	47	2008/2009	14	2008-11-29	492651	3	2	2008	
3	65	2008/2009	16	2008-12-13	492713	1	0	2008	
4	94	2008/2009	19	2009-01-24	492805	2	0	2009	

Data Cleaning (Players)

In [44]:

```
# join df_player with df_player_attribute table by inner join type
df_player = df_player.merge(df_player_attribute, on=['player_api_id','player_fifa_ap
# rename column: id_x to id
df_player.rename(columns={'id_x' : 'id'}, inplace=True)
# drop column id_y,player_api_id, player_fifa_api_id
df_player.drop(columns=["id_y","player_api_id" , "player_fifa_api_id"], inplace=True
```

```
#Show sample
df_player.head()
```

Out[44]:

	id	player_name	birthday	height	weight	date	overall_rating	potential	preferred_foot	a
0	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2016-02-18 00:00:00	67.0	71.0	right	
1	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2015-11-19 00:00:00	67.0	71.0	right	
2	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2015-09-21 00:00:00	62.0	66.0	right	
3	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2015-03-20 00:00:00	61.0	65.0	right	
4	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2007-02-22 00:00:00	61.0	65.0	right	

5 rows × 44 columns

In [45]:

```
## Drop duplicated records
df_player.drop_duplicates(inplace = True)
## Drop records have missed value
df_player.dropna(inplace=True)
```

Data Cleaning (Teams Table)

In [46]:

```
# join df_team with df_team_attribute table by inner join type
df_team = df_team.merge(df_team_attribute , on = ['team_api_id','team_fifa_api_id'],
# drop column id_y,team_api_id, team_fifa_api_id, team_short_name
df_team.drop(columns=["id_y","team_api_id" , "team_fifa_api_id","team_short_name"],
# rename column: id_x to id
df_team.rename(columns={'id_x': 'id'}, inplace=True)

#Show sample
df_player.head()
```

Out[46]:

	id	player_name	birthday	height	weight	date	overall_rating	potential	preferred_foot	a
0	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2016-02-18 00:00:00	67.0	71.0	right	
1	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2015-11-19 00:00:00	67.0	71.0	right	
2	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2015-09-21 00:00:00	62.0	66.0	right	
3	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2015-03-20 00:00:00	61.0	65.0	right	

	id	player_name	birthday	height	weight	date	overall_rating	potential	preferred_foot	a
4	1	Aaron Appindangoye	1992-02-29 00:00:00	182.88	187	2007-02-22 00:00:00	61.0	65.0	right	

5 rows × 44 columns

In [47]:
```
# Add new Columns to team table store years
df_team['date'] = pd.to_datetime(df_team['date'])
df_team['year'] = df_team['date'].dt.year
```

In [48]:
```
#Show sample
df_team.head()
```

Out[48]:

	id	team_long_name	date	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpPlayDribbling	bui
0	1	KRC Genk	2010-02-22	45	Balanced	NaN	
1	1	KRC Genk	2011-02-22	66	Balanced	NaN	
2	1	KRC Genk	2012-02-22	53	Balanced	NaN	
3	1	KRC Genk	2013-09-20	58	Balanced	NaN	
4	1	KRC Genk	2014-09-19	58	Balanced	52.0	

5 rows × 25 columns

In [49]:
```
## Drop records have missed value
df_team.dropna(inplace=True)
## Drop duplicated records
df_team.drop_duplicates(inplace = True)
```

Exploratory Data Analysis

How many matches are there in the 2016 season?

In [50]:
```
# Create a new fuction which return name of winner team foorm every match
def win(df_match):
    home_score = df_match[0]
    away_score = df_match[1]
    home_team_name = df_match[2]
    away_team_name = df_match[3]

    if home_score > away_score:
        return home_team_name
    elif home_score < away_score:
        return away_team_name
    else:
        return 'DRAW'
```

```
# Add new column which store winners team within Win Function
df_match['winner']  = df_match[['home_team_goal', 'away_team_goal', 'home_team_name'
```

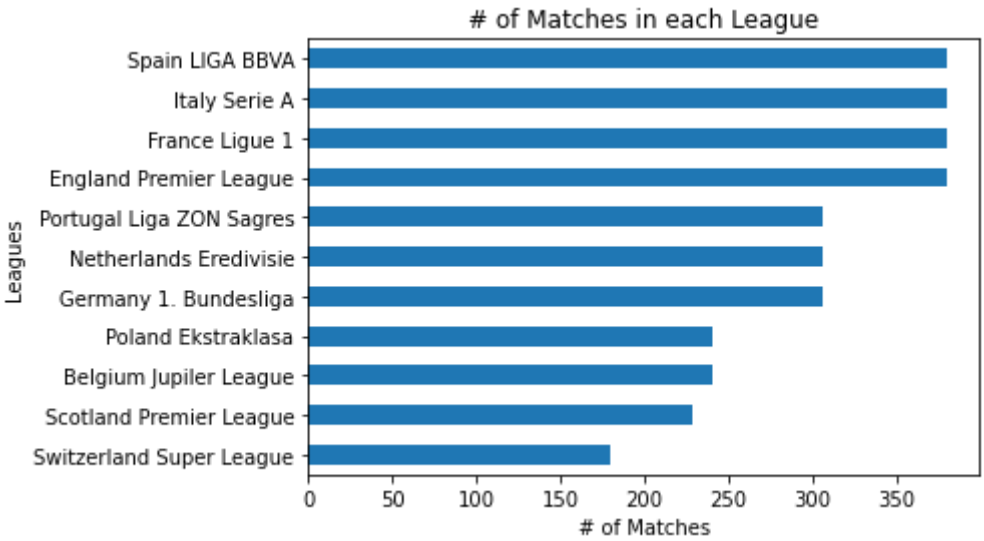
In [51]:
```
# Filter matches that only played in the 2015/2016 season
match_2016 = df_match[df_match['season'] == '2015/2016']
```

In [52]:
```
# Cuunt every match Played in each league in the 2015/2016 season
match_Played = match_2016.groupby('league_name')['home_team_name'].count().sort_valu
match_Played
```

Out[52]:
```
league_name
Switzerland Super League    180
Scotland Premier League     228
Belgium Jupiler League      240
Poland Ekstraklasa          240
Germany 1. Bundesliga       306
Netherlands Eredivisie      306
Portugal Liga ZON Sagres    306
England Premier League      380
France Ligue 1              380
Italy Serie A               380
Spain LIGA BBVA             380
Name: home_team_name, dtype: int64
```

In [53]:
```
# Figure the result
match_Played.plot(kind='barh', title='# of Matches in each League');
plt.xlabel('# of Matches')
plt.ylabel('Leagues')
```

Out[53]:
```
Text(0, 0.5, 'Leagues')
```



Which League had the most matches end as draw in the 2016 season?

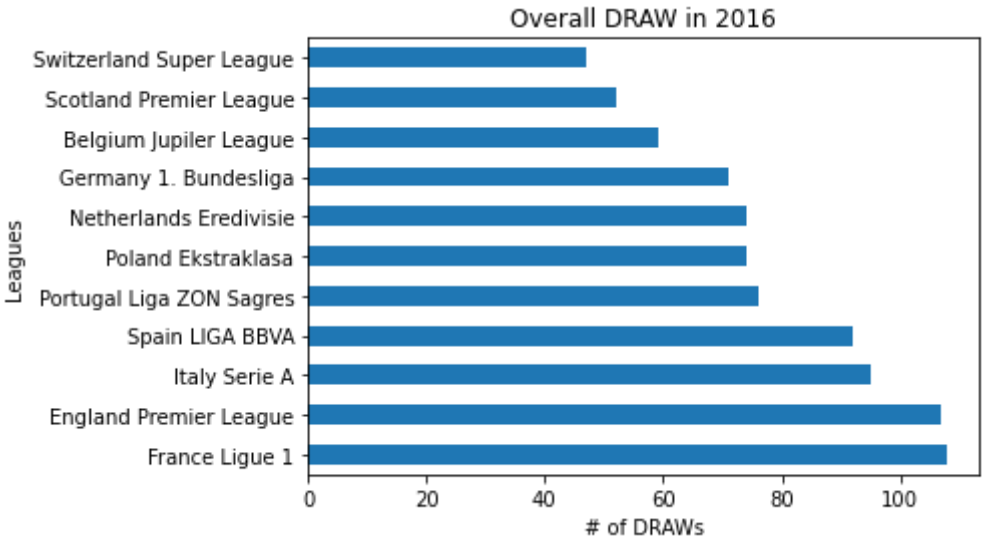
In [54]:
```
match_2016[['winner',"league_name"]].value_counts().loc["DRAW"]
```

Out[54]:
```
league_name
France Ligue 1              108
England Premier League      107
Italy Serie A                95
Spain LIGA BBVA              92
Portugal Liga ZON Sagres     76
Poland Ekstraklasa           74
Netherlands Eredivisie       74
Germany 1. Bundesliga        71
Belgium Jupiler League       59
Scotland Premier League      52
```

```
Switzerland Super League     47
dtype: int64
```

In [55]:
```
# Figure the result
match_2016[['winner',"league_name"]].value_counts().loc["DRAW"].plot(kind='barh', ti
plt.xlabel('# of DRAWs')
plt.ylabel('Leagues')
```

Out[55]:
```
Text(0, 0.5, 'Leagues')
```



In [56]:
```
match_2016[['winner',"league_name"]].value_counts().loc["DRAW"]
```

Out[56]:
```
league_name
France Ligue 1               108
England Premier League       107
Italy Serie A                 95
Spain LIGA BBVA               92
Portugal Liga ZON Sagres      76
Poland Ekstraklasa            74
Netherlands Eredivisie        74
Germany 1. Bundesliga         71
Belgium Jupiler League        59
Scotland Premier League       52
Switzerland Super League      47
dtype: int64
```

Which team had the most Wins or not Draw in the 2016 season?

In [57]:
```
wins16 = match_2016.groupby(["league_name"]).apply(lambda x: (x["winner"]!= 'DRAW').
wins16
```

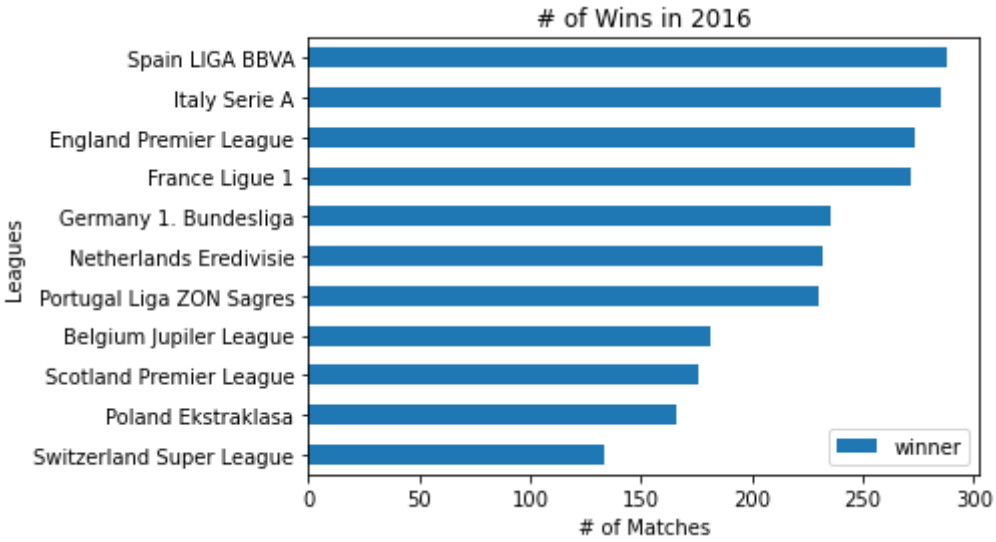
Out[57]:

	league_name	winner
10	Switzerland Super League	133
6	Poland Ekstraklasa	166
8	Scotland Premier League	176
0	Belgium Jupiler League	181
7	Portugal Liga ZON Sagres	230
5	Netherlands Eredivisie	232
3	Germany 1. Bundesliga	235
2	France Ligue 1	272

	league_name	winner
1	England Premier League	273
4	Italy Serie A	285
9	Spain LIGA BBVA	288

In [58]:
```
# Figure the result
wins16.plot(x= 'league_name' ,kind='barh', title='# of Wins in 2016')
plt.xlabel('# of Matches')
plt.ylabel('Leagues')
```

Out[58]:
```
Text(0, 0.5, 'Leagues')
```



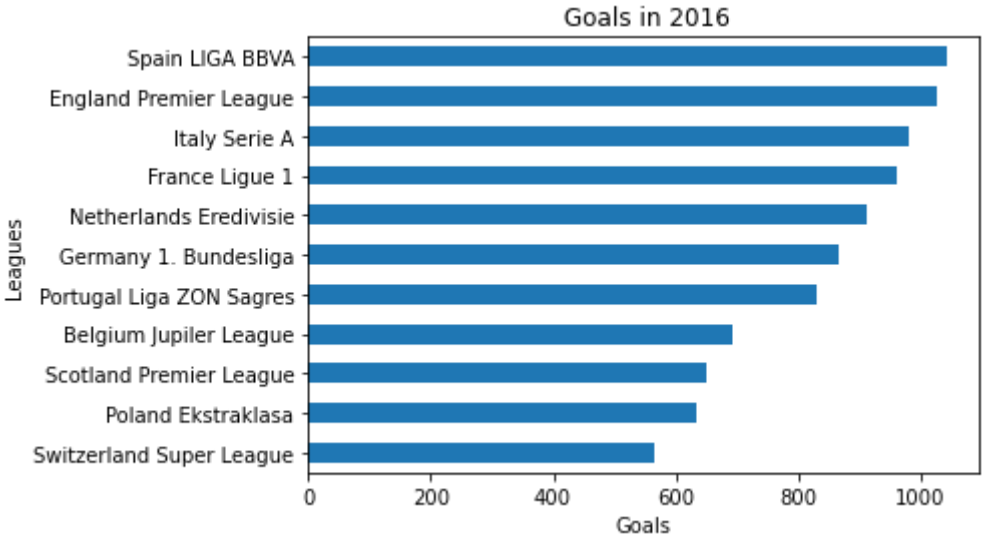
How many goals in each League are there in the 2016 season?

In [59]:
```
goals_2016 = match_2016.groupby('league_name')['home_team_goal'].sum().sort_values(a
goals_2016.sort_values(ascending = True, inplace =True)
goals_2016
```

Out[59]:
```
league_name
Switzerland Super League     566
Poland Ekstraklasa           635
Scotland Premier League      650
Belgium Jupiler League       694
Portugal Liga ZON Sagres     831
Germany 1. Bundesliga        866
Netherlands Eredivisie       912
France Ligue 1               960
Italy Serie A                979
England Premier League      1026
Spain LIGA BBVA             1043
dtype: int64
```

In [60]:
```
# Figure the result
goals_2016.plot(kind='barh', title=' Goals in 2016')
plt.xlabel('Goals')
plt.ylabel('Leagues')
```

Out[60]:
```
Text(0, 0.5, 'Leagues')
```

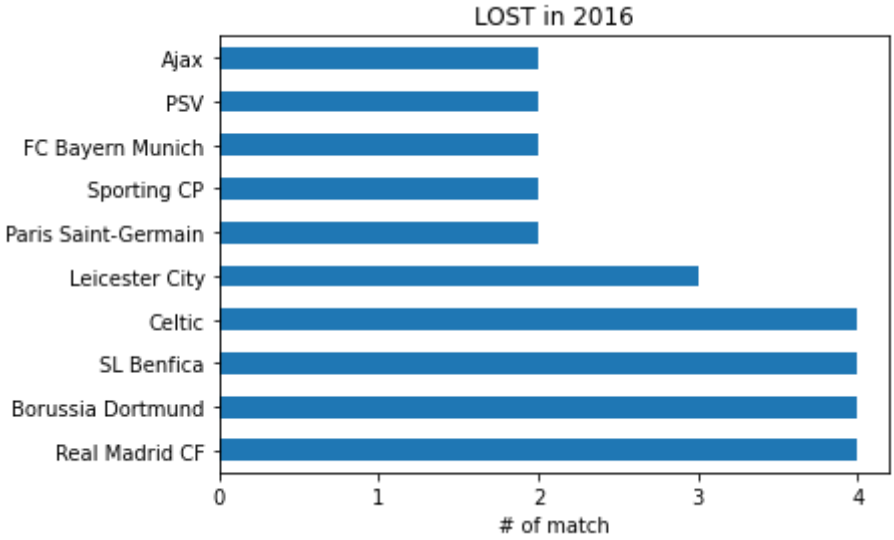


Which team had lost the fewest matches in the 2016 season?

In [61]:
```
# Add New Column f9r loser teams within lose function
def lose(df_match):
    home_score = df_match[0]
    away_score = df_match[1]
    home_team_name = df_match[2]
    away_team_name = df_match[3]

    if home_score < away_score:
        return home_team_name
    elif home_score > away_score:
        return away_team_name
    else:
        return 'DRAW'

df_match['loser']  = df_match[['home_team_goal', 'away_team_goal', 'home_team_name',
```

In [62]:
```
# Figure the result
match_2016 = df_match[df_match['season'] == '2015/2016']

match_2016.loser.value_counts().tail(10).plot(kind='barh', title='LOST in 2016',xtic
plt.xlabel('# of match')
```

Out[62]:
```
Text(0.5, 0, '# of match')
```



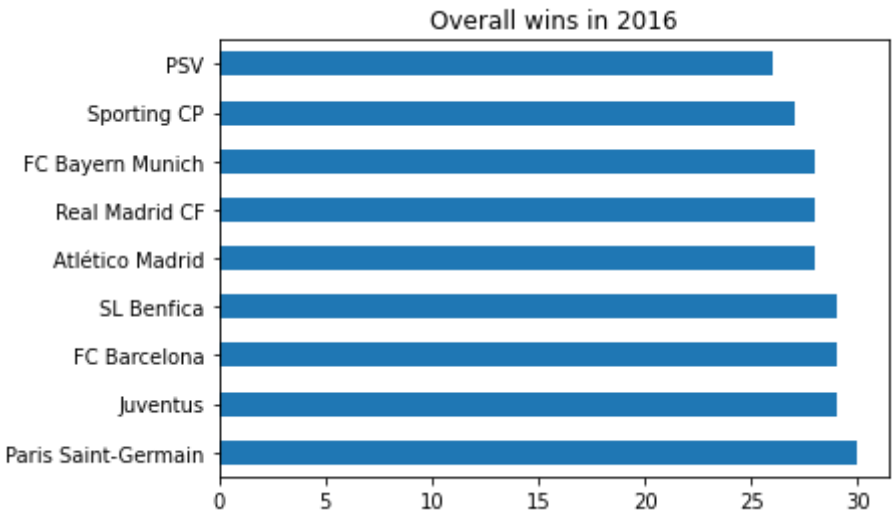
Which teams had the most wins of matches in the 2016 season?

In [63]:
```
wins_16 = match_2016.winner.value_counts().head(10).iloc[1:]
wins_16
```

Out[63]:
```
Paris Saint-Germain    30
Juventus               29
FC Barcelona           29
SL Benfica             29
Atlético Madrid        28
Real Madrid CF         28
FC Bayern Munich       28
Sporting CP            27
PSV                    26
Name: winner, dtype: int64
```

In [64]:
```
# Figure the result
wins_16.plot(kind='barh', title='Overall wins in 2016')
```

Out[64]:
```
<AxesSubplot:title={'center':'Overall wins in 2016'}>
```



What teams improved the most over the time period?

In [65]:
```
# Select wins games in 2010
W2010 = df_match[(df_match['season_year'] == 2010) & (df_match['winner'] != 'DRAW')]
# Select lose games in 2010
L2010 = df_match[(df_match['season_year'] == 2010) & (df_match['loser'] != 'DRAW')]
# count wins games in 2010
countW2010 = W2010['winner'].count()
# count lose games in 2010
countL2010 = L2010['loser'].count()
# Sustract between winner and loser in 2010
R2010 = W2010['winner'].value_counts()/countW2010 - L2010['loser'].value_counts()/co

# Select wins games in 2010
W2016 = df_match[(df_match['season_year'] == 2016) & (df_match['winner'] != 'DRAW')]
# Select lose games in 2010
L2016 = df_match[(df_match['season_year'] == 2016) & (df_match['loser'] != 'DRAW')]
# count wins games in 2010
countW2016 = W2016['winner'].count()
# count lose games in 2010
countL2016 = L2016['loser'].count()
# Sustract between winner and loser in 2010
R2016 = W2016['winner'].value_counts()/countW2010 - L2016['loser'].value_counts()/co
```

In [66]:
```
# Select wins match in 2010
match_2010 = df_match[df_match['season_year'] == 2010]
# Select wins match in 2016
```

```
match_2016 = df_match[df_match['season_year'] == 2016]
#select average of away team goal in 2010
df_match_2010_away = match_2010.groupby(['away_team_name'])['away_team_goal'].mean()
#select average of home team goal in 2010
df_match_2010_home = match_2010.groupby(['home_team_name'])['home_team_goal'].mean()
#select average of away team goal in 2016
df_match_2016_away = match_2016.groupby(['away_team_name'])['away_team_goal'].mean()
#select average of home team goal in 2010
df_match_2016_home = match_2016.groupby(['home_team_name'])['home_team_goal'].mean()
#select average of all team goal in 2010
df_match_total_2010 = (df_match_2010_away + df_match_2010_home) / 2
#select average of all team goal in 2016
df_match_total_2016 = (df_match_2016_away + df_match_2016_home) / 2
#select rate of change subtract average of all team goal in 2016 and 2010
#then add result to subtract wins and loses
diff_match_2016_2010 = ((df_match_total_2016 - df_match_total_2010) + ( R2016 - R201
```

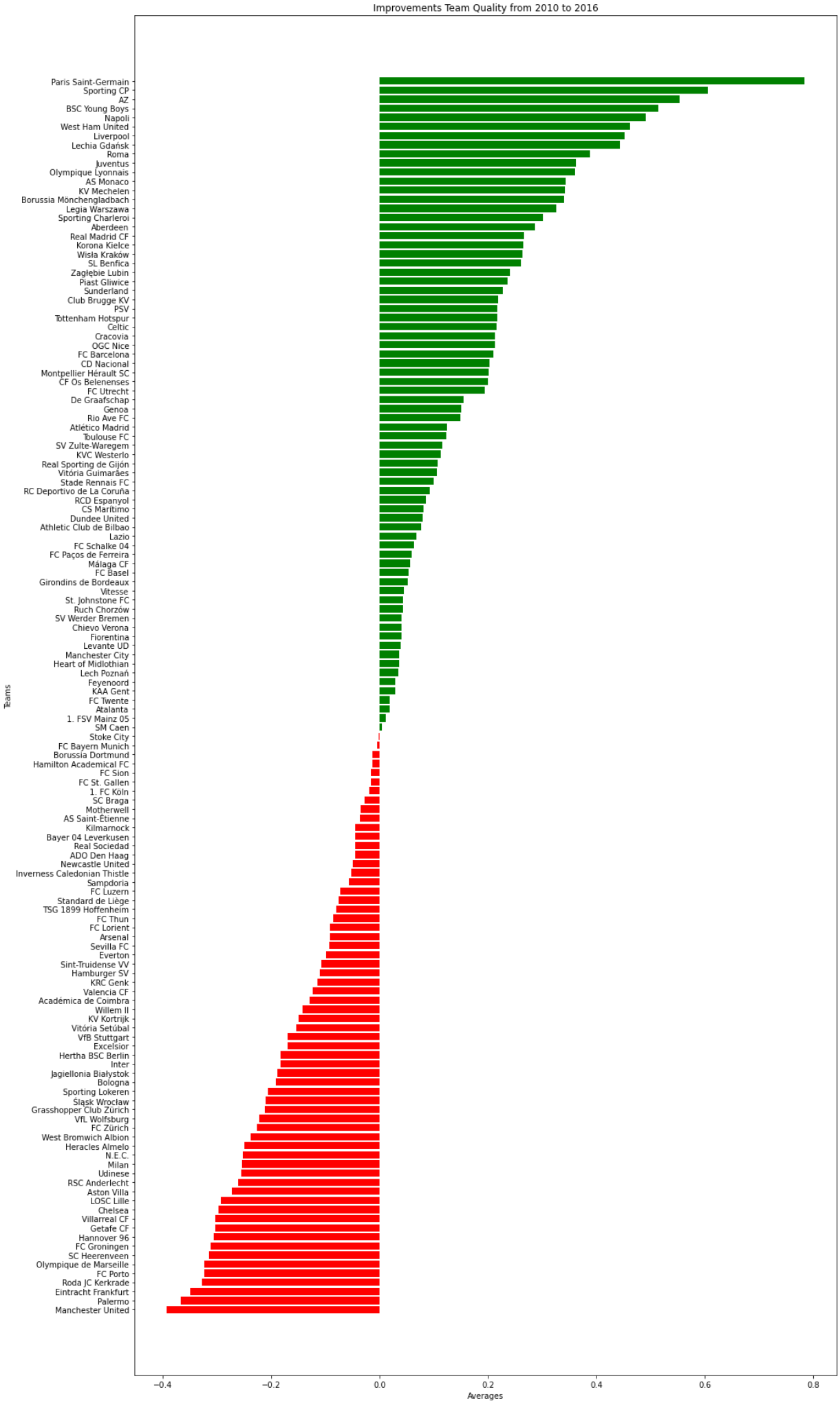
In [67]:

```
#set color list with green color for positive values and red color for negative valu
color = []
def coloring(df):
        if df < 0:
            return 'red' #red color for negative values
        else:
            return 'green'  #green color for positive values
#call function coloring and store the result in coloor list
color= diff_match_2016_2010.sort_values().apply(coloring)
```

In [68]:

```
diff_match_2016_2010.dropna(inplace=True)
print('Improved teams :' + str(diff_match_2016_2010[diff_match_2016_2010.sort_values

# sort the values
sorted_index = diff_match_2016_2010.sort_values().index

fig, ax = plt.subplots(figsize=(15, 25))

# plot a horizontal bar
plt.barh(range(0,len(sorted_index)), diff_match_2016_2010.sort_values(),color = colo

# Set the position of the y ticks
ax.set_yticks(range(0,len(sorted_index)))

# Set the position of the y ticks labels
ax.set_yticklabels(sorted_index)

# Set the y axis label
ax.set_ylabel('Teams')

# Set the chart's title
ax.set_title('Improvements Team Quality from 2010 to 2016')

# Set the y axis label
plt.xlabel("Averages")
#show Chart
plt.tight_layout();
```

```
Improved teams :72
```


Improvements Team Quality from 2010 to 2016
Teams
Averages
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Paris Saint-Germain
Sporting CP
AZ
BSC Young Boys
Napoli
West Ham United
Liverpool
Lechia Gdańsk
Roma
Juventus
Olympique Lyonnais
AS Monaco
KV Mechelen
Borussia Mönchengladbach
Legia Warszawa
Sporting Charleroi
Aberdeen
Real Madrid CF
Korona Kielce
Wisła Kraków
SL Benfica
Zagłębie Lubin
Piast Gliwice
Sunderland
Club Brugge KV
PSV
Tottenham Hotspur
Celtic
Cracovia
OGC Nice
FC Barcelona
CD Nacional
Montpellier Hérault SC
CF Os Belenenses
FC Utrecht
De Graafschap
Genoa
Rio Ave FC
Atlético Madrid
Toulouse FC
SV Zulte-Waregem
KVC Westerlo
Real Sporting de Gijón
Vitória Guimarães
Stade Rennais FC
RC Deportivo de La Coruña
RCD Espanyol
CS Marítimo
Dundee United
Athletic Club de Bilbao
Lazio
FC Schalke 04
FC Paços de Ferreira
Málaga CF
FC Basel
Girondins de Bordeaux
Vitesse
St. Johnstone FC
Ruch Chorzów
SV Werder Bremen
Chievo Verona
Fiorentina
Levante UD
Manchester City
Heart of Midlothian
Lech Poznań
Feyenoord
KAA Gent
FC Twente
Atalanta
1. FSV Mainz 05
SM Caen
Stoke City
FC Bayern Munich
Borussia Dortmund
Hamilton Academical FC
FC Sion
FC St. Gallen
1. FC Köln
SC Braga
Motherwell
AS Saint-Étienne
Kilmarnock
Bayer 04 Leverkusen
Real Sociedad
ADO Den Haag
Newcastle United
Inverness Caledonian Thistle
Sampdoria
FC Luzern
Standard de Liège
TSG 1899 Hoffenheim
FC Thun
FC Lorient
Arsenal
Sevilla FC
Everton
Sint-Truidense VV
Hamburger SV
KRC Genk
Valencia CF
Académica de Coimbra
Willem II
KV Kortrijk
Vitória Setúbal
VfB Stuttgart
Excelsior
Hertha BSC Berlin
Inter
Jagiellonia Białystok
Bologna
Sporting Lokeren
Śląsk Wrocław
Grasshopper Club Zürich
VfL Wolfsburg
FC Zürich
West Bromwich Albion
Heracles Almelo
N.E.C.
Milan
Udinese
RSC Anderlecht
Aston Villa
LOSC Lille
Chelsea
Villarreal CF
Getafe CF
Hannover 96
FC Groningen
SC Heerenveen
Olympique de Marseille
FC Porto
Roda JC Kerkrade
Eintracht Frankfurt
Palermo
Manchester United

Which players had the most penalties?

In [69]:
```
# Select top player name and their penalties in descending order
df_player.groupby(['player_name'])['penalties'].max().sort_values(ascending=False)[:
```

Out[69]:
```
player_name
Rickie Lambert     96.0
Mario Balotelli    95.0
Xavi Hernandez     95.0
Andrea Pirlo       95.0
Paul Scholes       95.0
David Trezeguet    94.0
Cesc Fabregas      94.0
Adrian Mutu        94.0
Iker Casillas      94.0
Hernan Crespo      93.0
Name: penalties, dtype: float64
```

In [70]:
```
# Select a top team  that most winner in 2015/2016
top_teams = df_team[(df_team["team_long_name"].isin(wins_16.index)) & (df_team['year
top_teams.shape
```

Out[70]:
```
(9, 25)
```

In [71]:
```
# Select teams which have most wins in 2016
wins_16.index
```

Out[71]:
```
Index(['Paris Saint-Germain', 'Juventus', 'FC Barcelona', 'SL Benfica',
       'Atlético Madrid', 'Real Madrid CF', 'FC Bayern Munich', 'Sporting CP',
       'PSV'],
      dtype='object')
```

In [ ]:

In [72]:
```
digital_attributes = ['team_long_name', 'buildUpPlaySpeed', 'buildUpPlayDribbling',
```

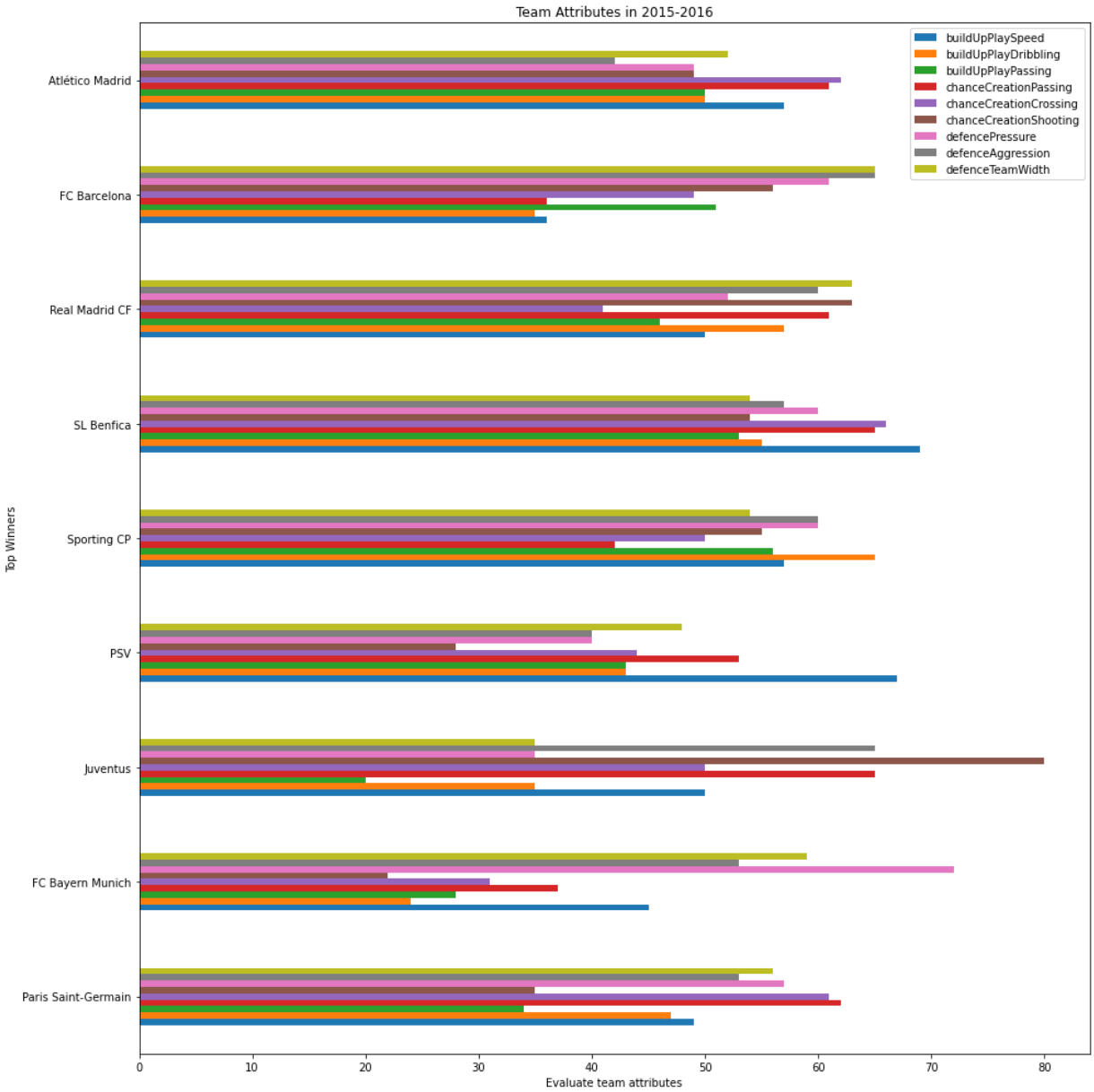
In [73]:
```
#Sample of top_teams with only digital columns
top_teams[digital_attributes].head()
```

Out[73]:

	team_long_name	buildUpPlaySpeed	buildUpPlayDribbling	buildUpPlayPassing	chanceCreatior
370	Paris Saint-Germain	49	47.0	34	
497	FC Bayern Munich	45	24.0	28	
716	Juventus	50	35.0	20	
885	PSV	67	43.0	43	
1052	Sporting CP	57	65.0	56	

In [74]:
```
# Figure the result
fig, ax = plt.subplots(figsize=(14, 14))

# create a new bar char
ax = top_teams[digital_attributes].plot.barh(ax=ax);

ax.set_yticklabels(top_teams['team_long_name'])

# Set the y axis label
ax.set_ylabel('Top Winners')
```

```
# Set the chart's title
ax.set_title('Team Attributes in 2015-2016')

# Set the y axis label
plt.xlabel("Evaluate team attributes")

plt.tight_layout();
```

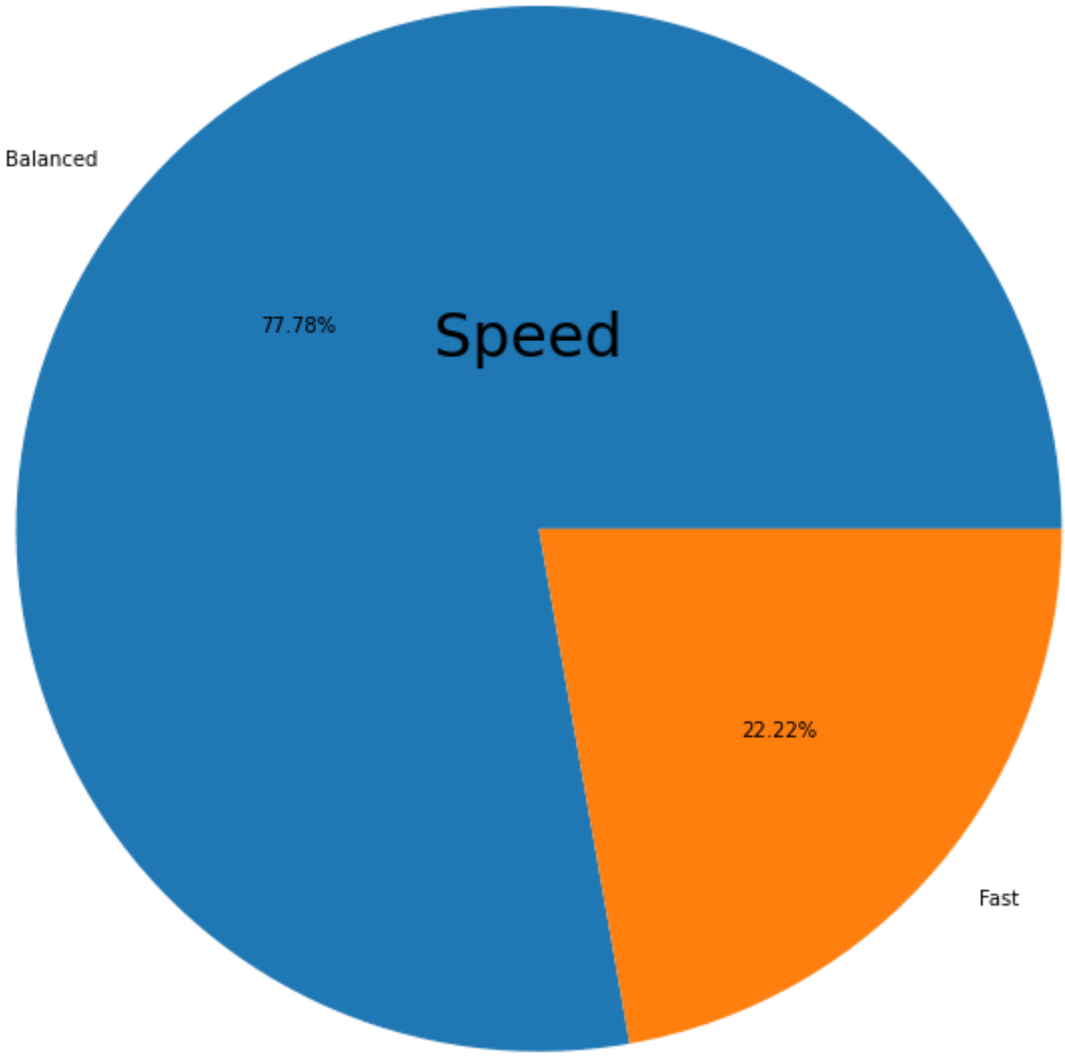


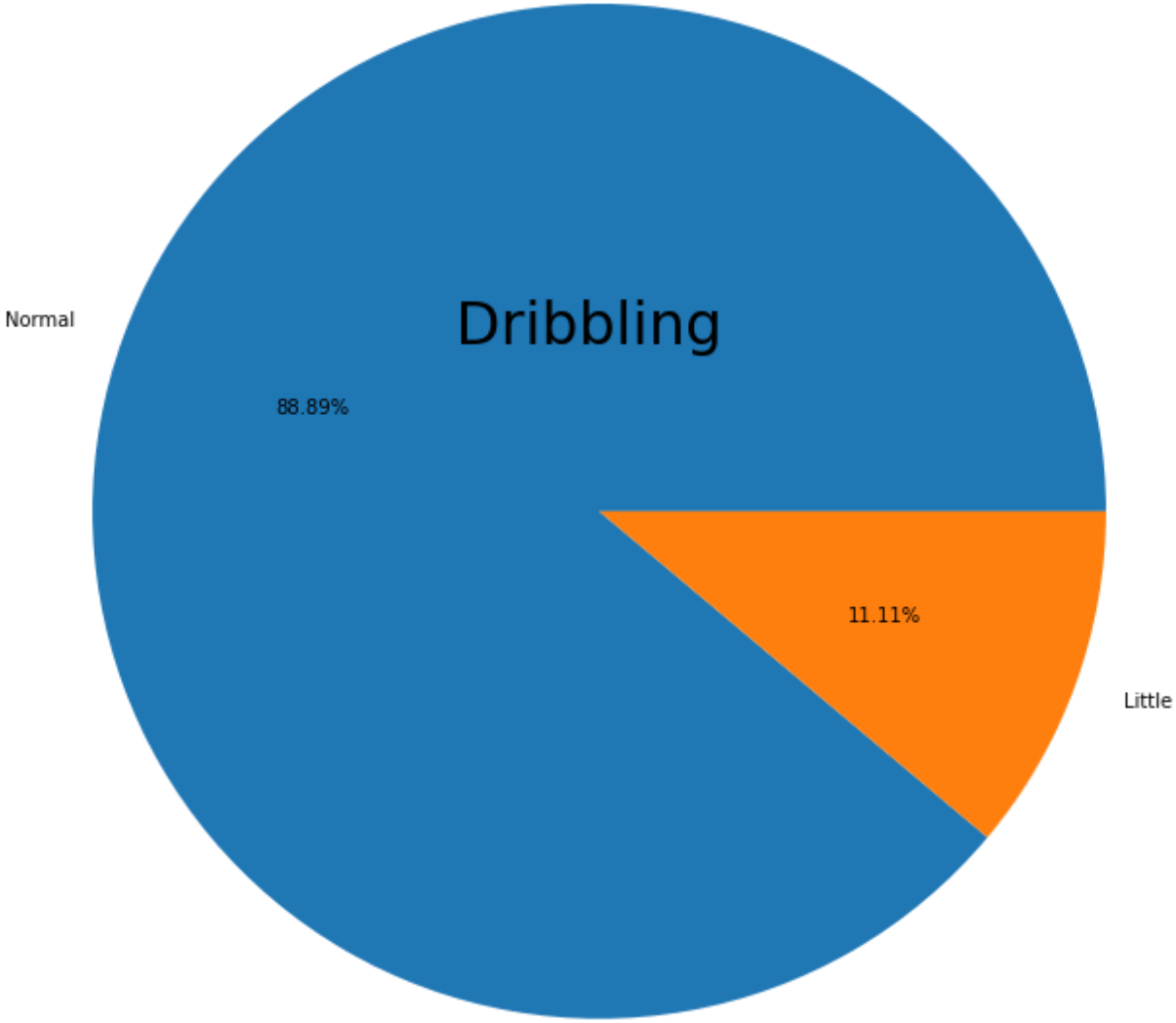
In [75]:

```
# list all columns name store descriptive values
descriptive_attributes = ['buildUpPlaySpeedClass','buildUpPlayDribblingClass','build
# assign only columns in descriptive_attributes
attr = top_teams[descriptive_attributes]
```

In [76]:

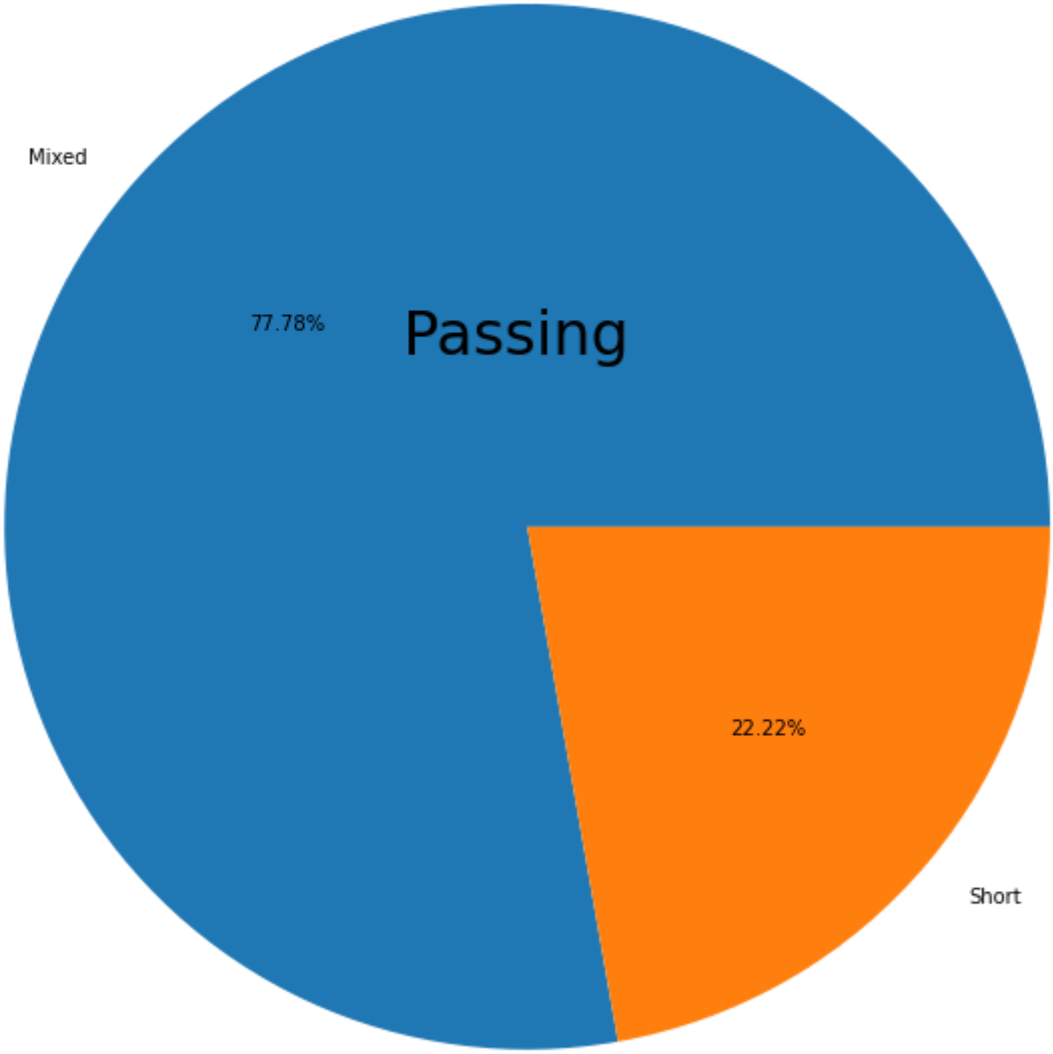
```
plt.figure(0)
# Create 1st chart here.
plt.pie(attr['buildUpPlaySpeedClass'].value_counts(), labels =  attr['buildUpPlaySpe
plt.figtext(.5,.8,'Speed',fontsize=30,ha='center')

plt.figure(1)
# Create 2nd chart here.
plt.pie(attr['buildUpPlayDribblingClass'].value_counts(), labels =  attr['buildUpPla
plt.figtext(.5,.8,'Dribbling',fontsize=30,ha='center')

plt.show() #show all figures
```

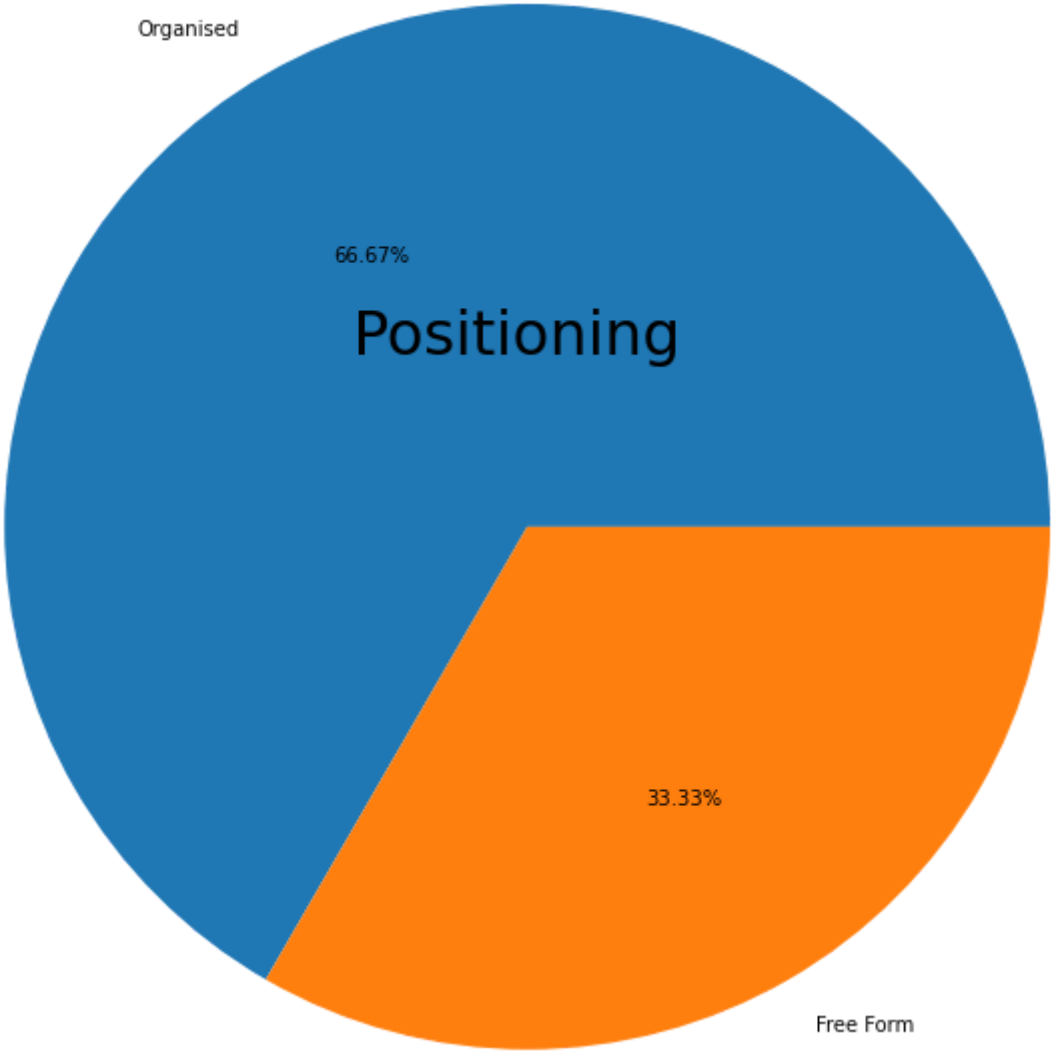

Speed
Balanced
77.78%
22.22%
Fast



In [77]:

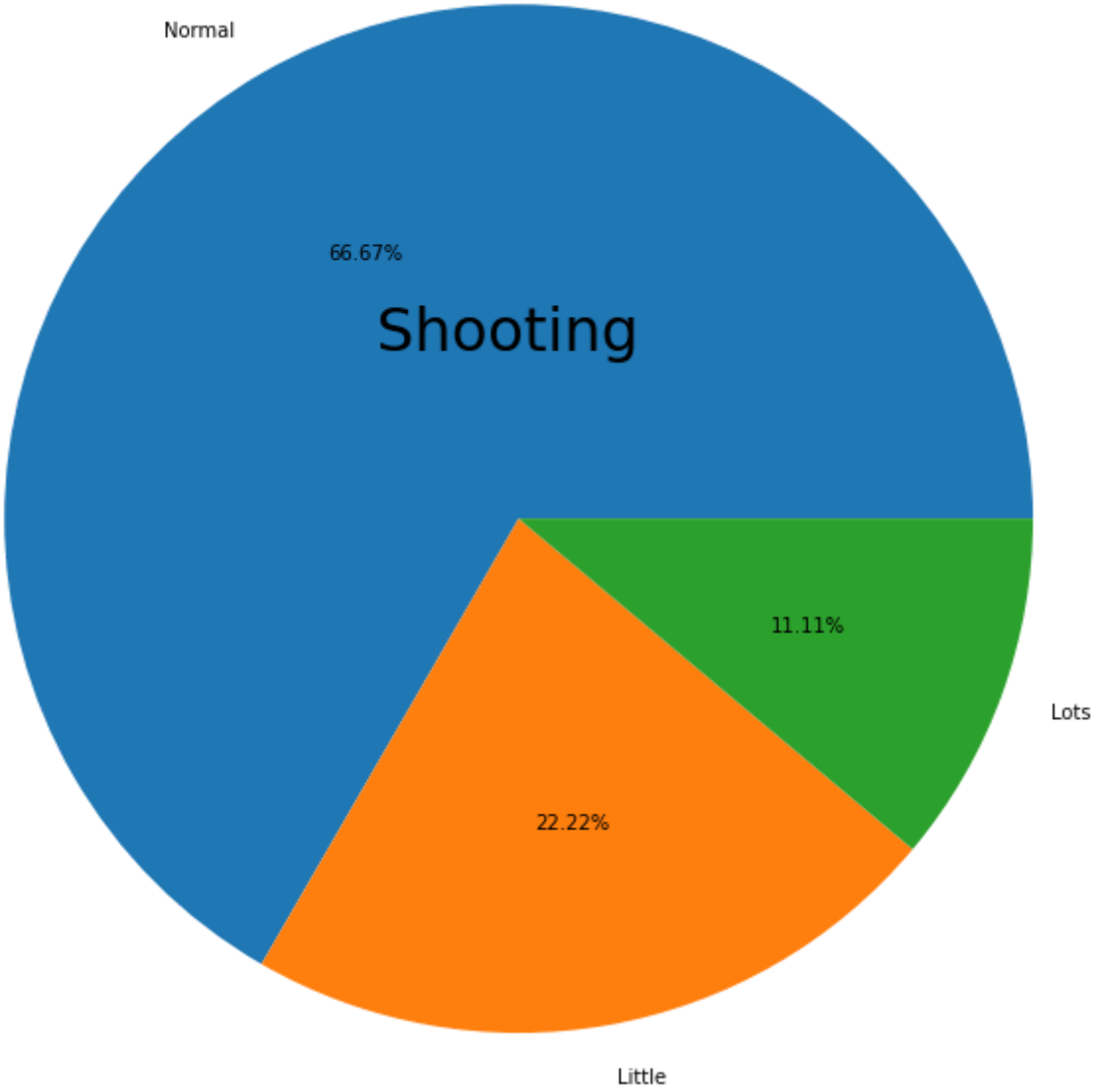
```
plt.figure(2)
# Create 3ird chart here.
plt.pie(attr['buildUpPlayPassingClass'].value_counts(), labels =  attr['buildUpPlayP
plt.figtext(.5,.8,'Passing',fontsize=30,ha='center')

plt.figure(3)
# Create 4th chart here.
plt.pie(attr['buildUpPlayPositioningClass'].value_counts(), labels =  attr['buildUpP
plt.figtext(.5,.8,'Positioning',fontsize=30,ha='center')


plt.show() #show all figures
```

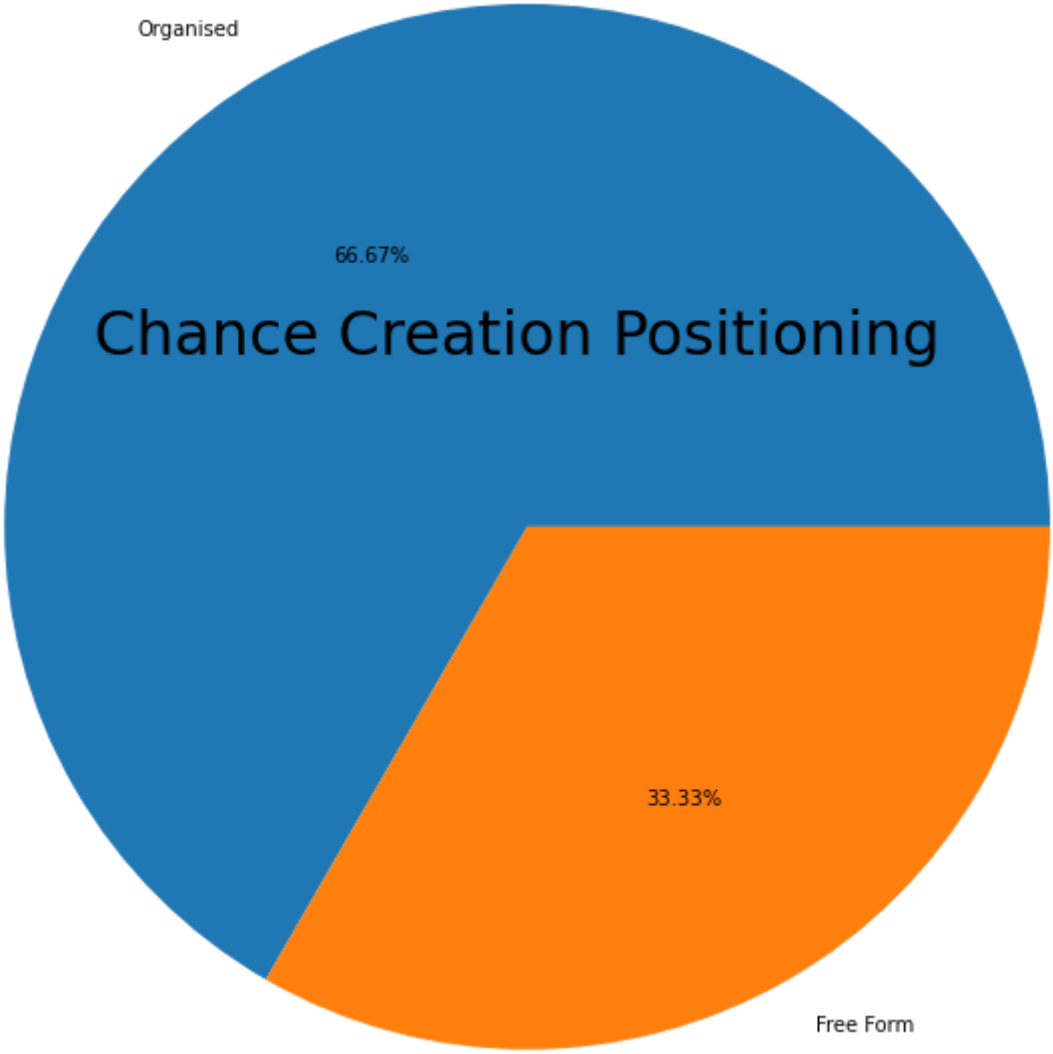

Passing
Mixed
77.78%
22.22%
Short



In [78]:

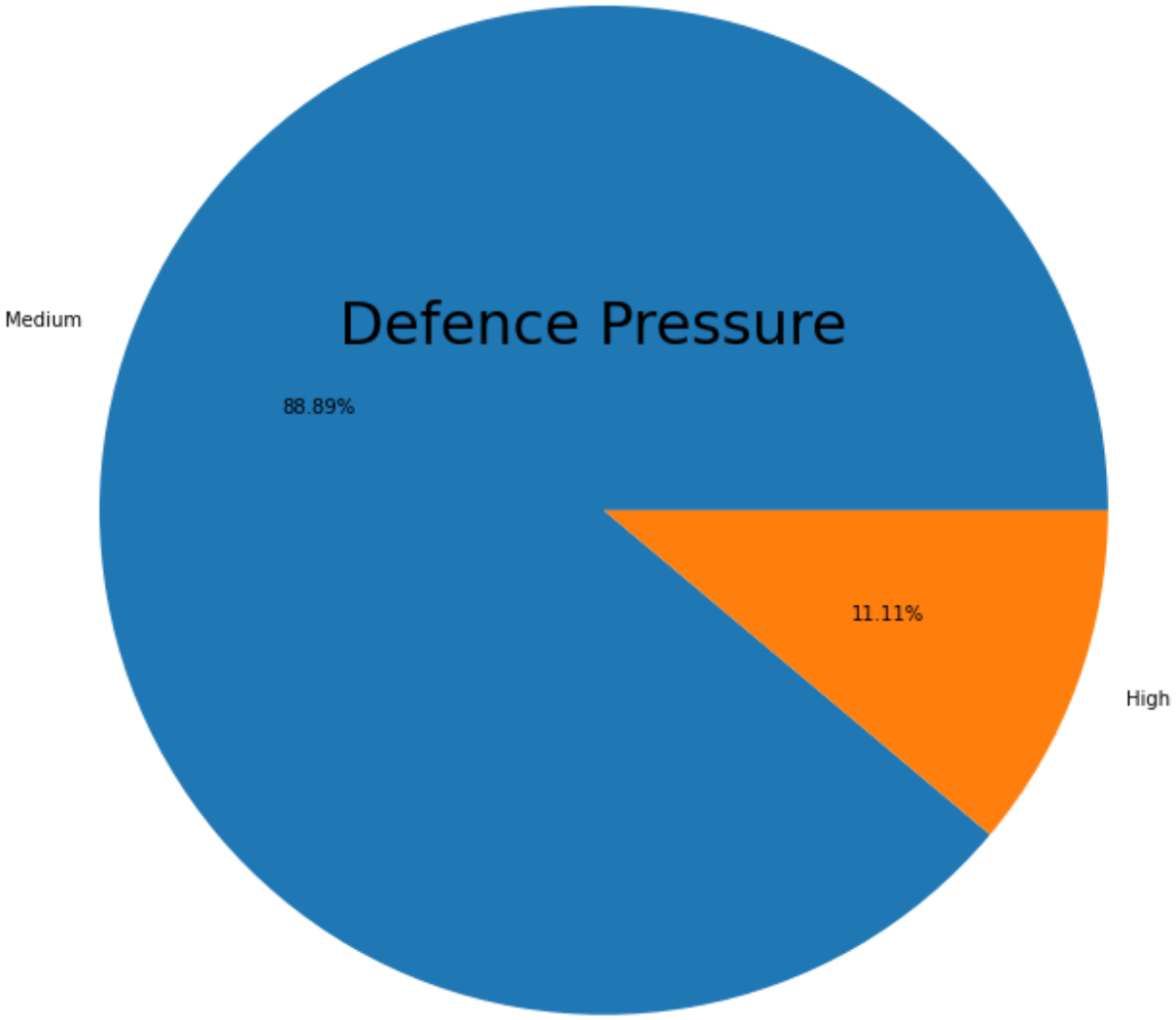
```
plt.figure(4)
# Create 5th chart here.
plt.pie(attr['chanceCreationShootingClass'].value_counts(), labels =  attr['chanceCr
plt.figtext(.5,.8,'Shooting',fontsize=30,ha='center')

plt.figure(5)
# Create 6th chart here.
plt.pie(attr['chanceCreationPositioningClass'].value_counts(), labels =  attr['chanc
plt.figtext(.5,.8,'Chance Creation Positioning',fontsize=30,ha='center')


plt.show() #show all figures
```

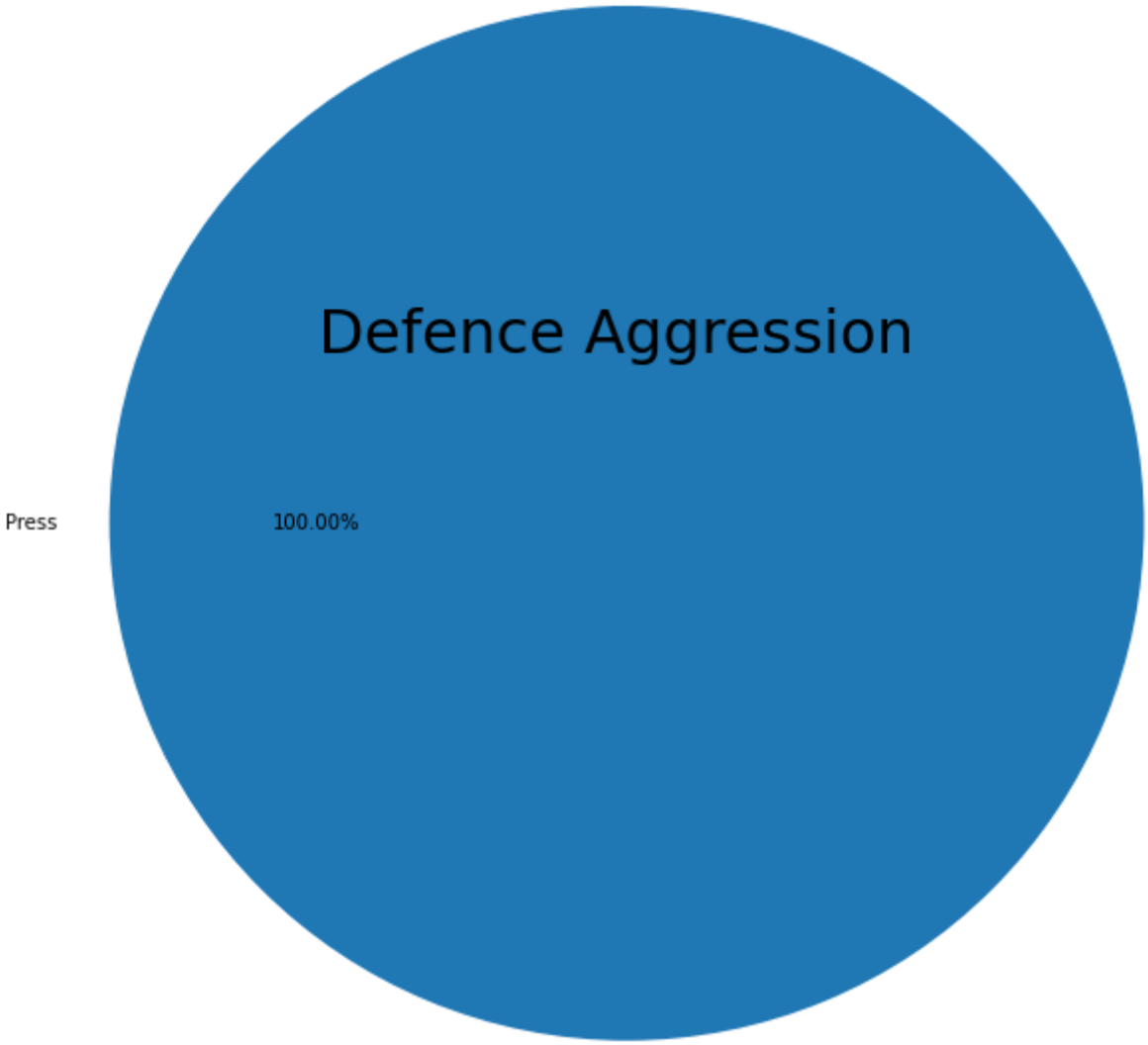

Normal
66.67%
Shooting
11.11%
Lots
22.22%
Little



In [79]:

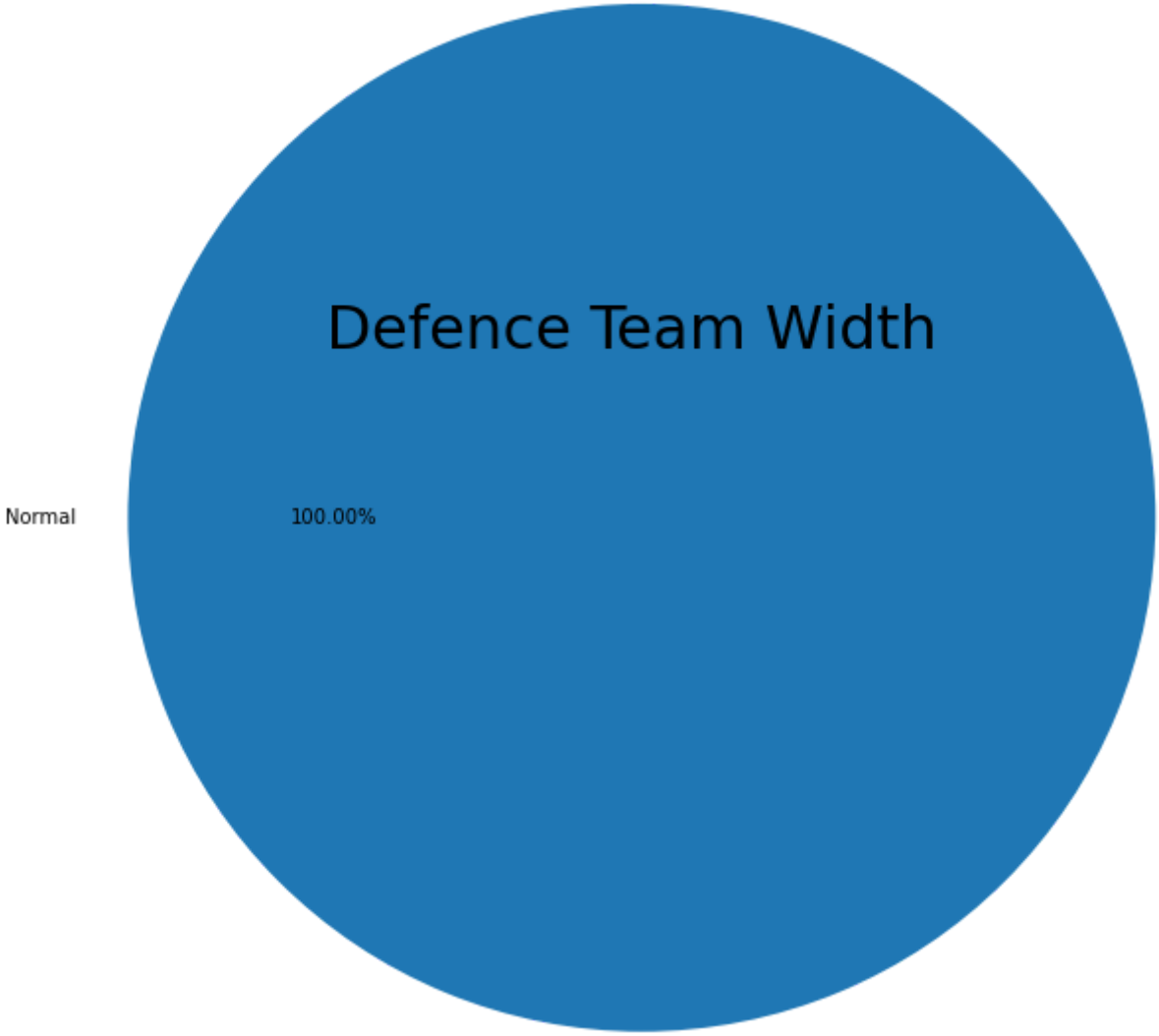
```
plt.figure(6)
# Create 7th chart here.
plt.pie(attr['defencePressureClass'].value_counts(), labels =  attr['defencePressure
plt.figtext(.5,.8,'Defence Pressure',fontsize=30,ha='center')

plt.figure(7)
# Create 8th chart here.
plt.pie(attr['defenceAggressionClass'].value_counts(), labels =  attr['defenceAggres
plt.figtext(.5,.8,'Defence Aggression',fontsize=30,ha='center')


plt.show() #show all figures
```

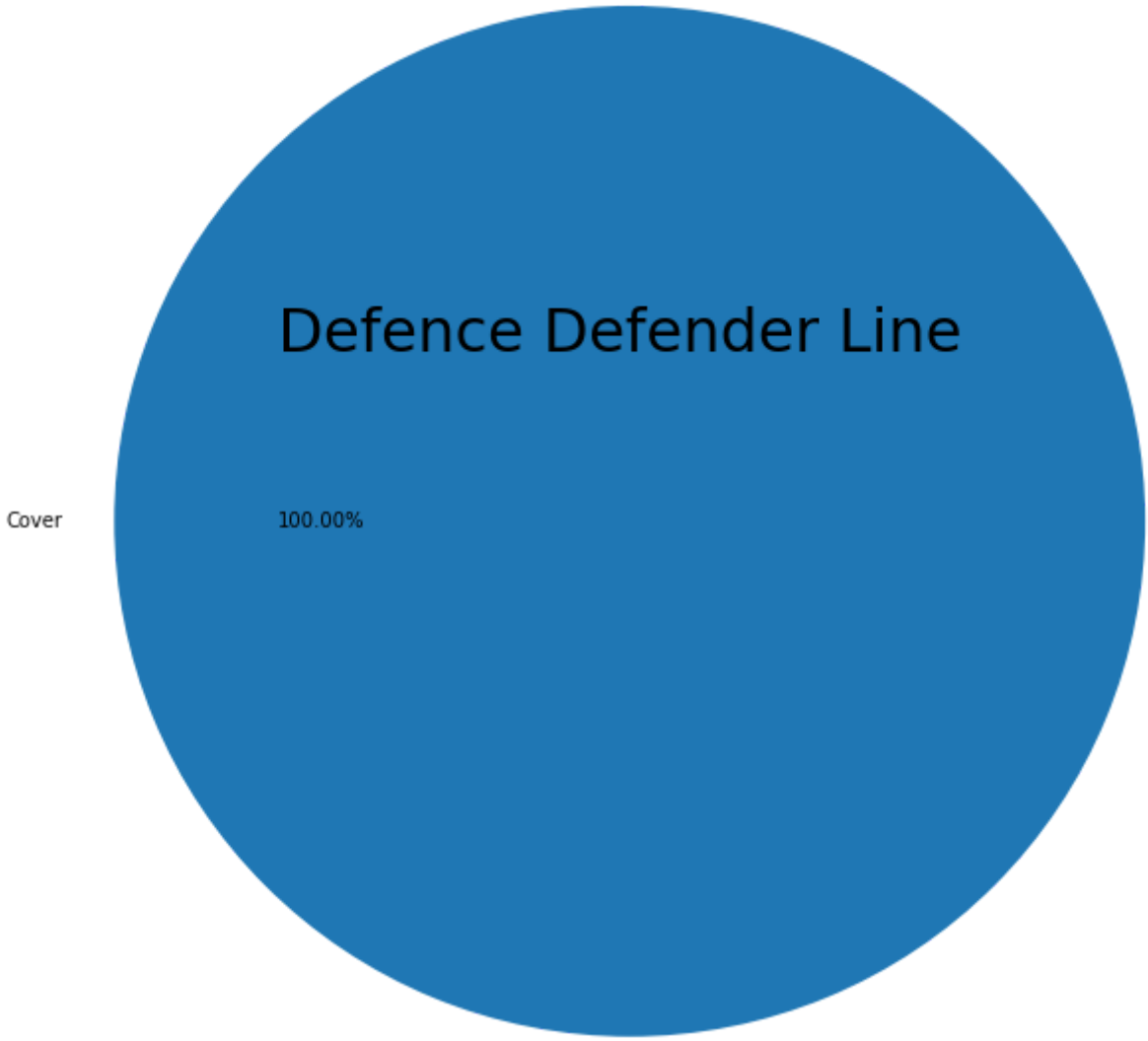

Defence Pressure
Medium
88.89%
11.11%
High



In [80]:

```
plt.figure(8)
# Create 9th chart here.
plt.pie(attr['defenceTeamWidthClass'].value_counts(), labels =  attr['defenceTeamWid
plt.figtext(.5,.8,'Defence Team Width',fontsize=30,ha='center')

plt.figure(9)
# Create 10th chart here.
plt.pie(attr['defenceDefenderLineClass'].value_counts(), labels =  attr['defenceDefe
plt.figtext(.5,.8,'Defence Defender Line',fontsize=30,ha='center')


plt.show() #show all figures
```


Defence Team Width
Normal
100.00%



How many Players have overall rating more than 90 ?

In [81]:
```
# Select Maximum Rate
df_player['overall_rating'].max()
```

Out[81]: 94.0

In [82]:
```
# average of players' overall rating
average_rate = df_player['overall_rating'].mean()
average_rate
```

Out[82]: 68.63280955234481

In [83]:
```
#Select players have above average rating Then Count it
above_rating = df_player[df_player['overall_rating'] > average_rate].player_name.nun
above_rating
```

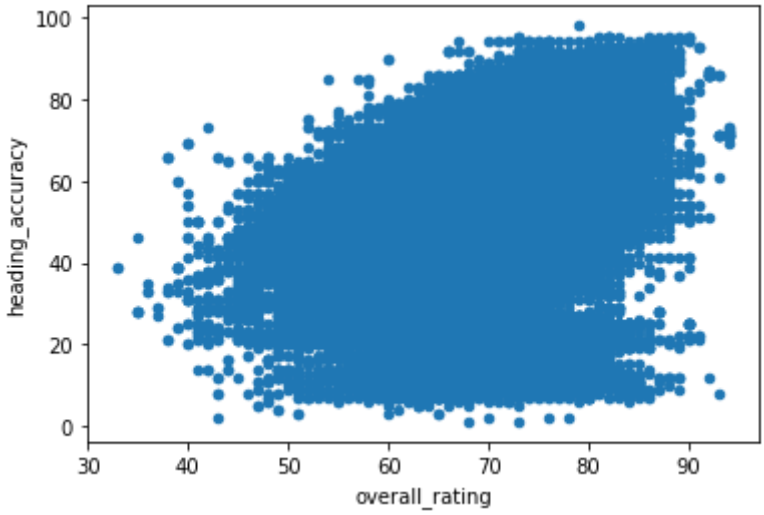
Out[83]: 6467

In [84]:
```
#Count players have overall rating more than 90
df_player[(df_player['overall_rating'] > 90)].player_name.nunique()
```

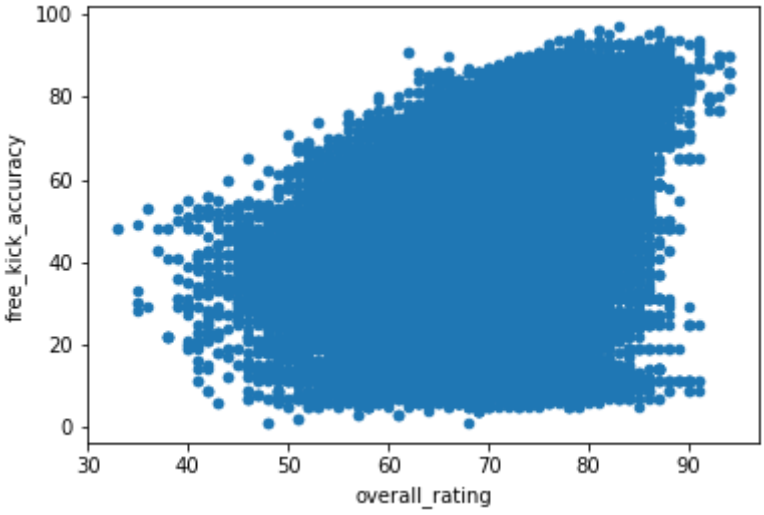
Out[84]: 12

What are the attributes that contribute to the players' overall rating?

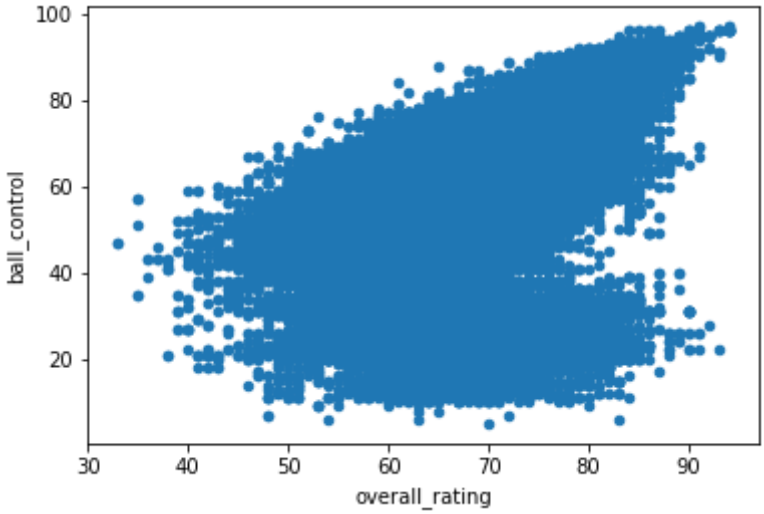
In [85]:
```
#correlation between heading accuracy and rating
df_player.plot(x='overall_rating', y='heading_accuracy', kind='scatter');
```



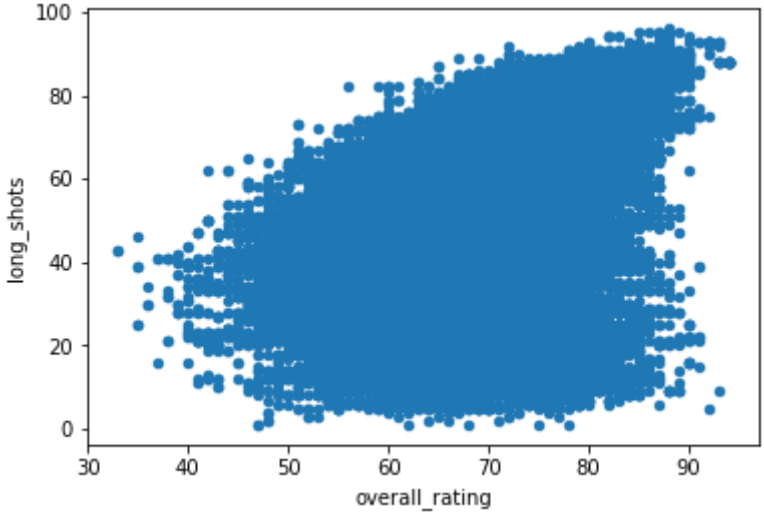
In [86]:
```
#correlation between rating and free kick accuracy and rating
df_player.plot(x='overall_rating', y='free_kick_accuracy', kind='scatter');
```



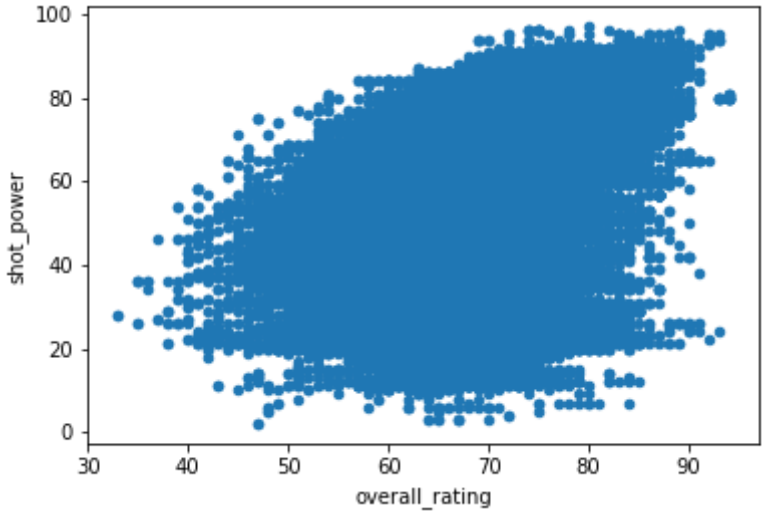
In [87]:
```
#correlation between rating and ball control and rating
df_player.plot(x='overall_rating', y='ball_control', kind='scatter');
```



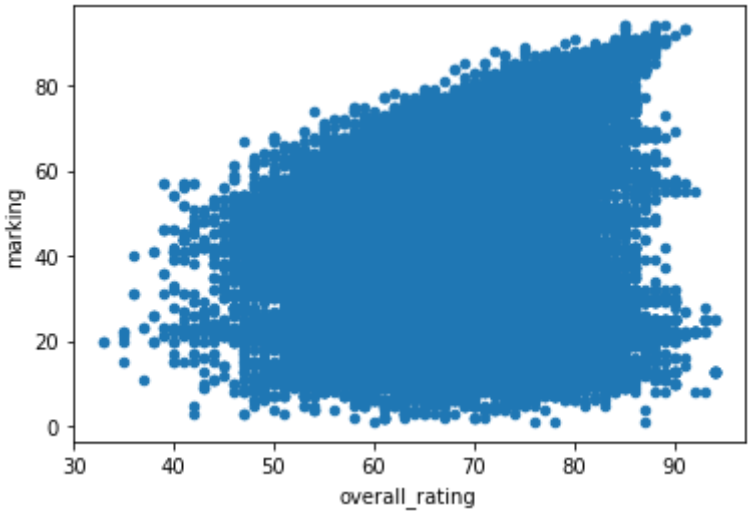
In [88]:
```
#correlation between long shots and rating
df_player.plot(x='overall_rating', y='long_shots', kind='scatter');
```



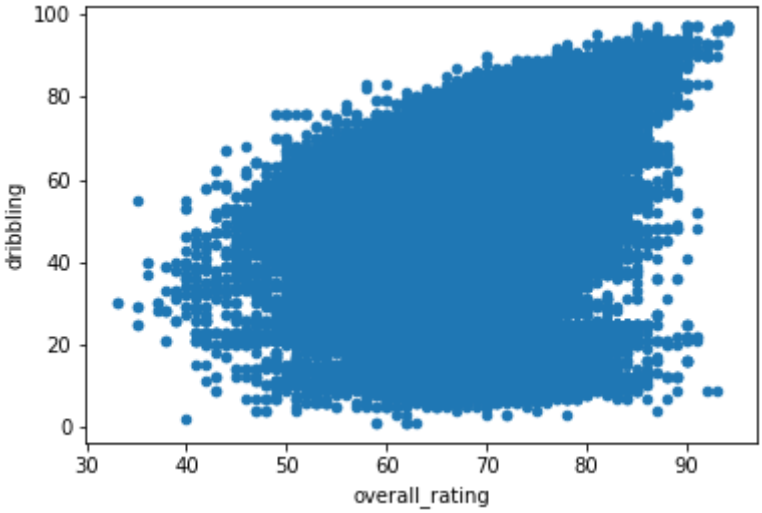
In [89]:
```
#correlation between shot power and rating
df_player.plot(x='overall_rating', y='shot_power', kind='scatter');
```



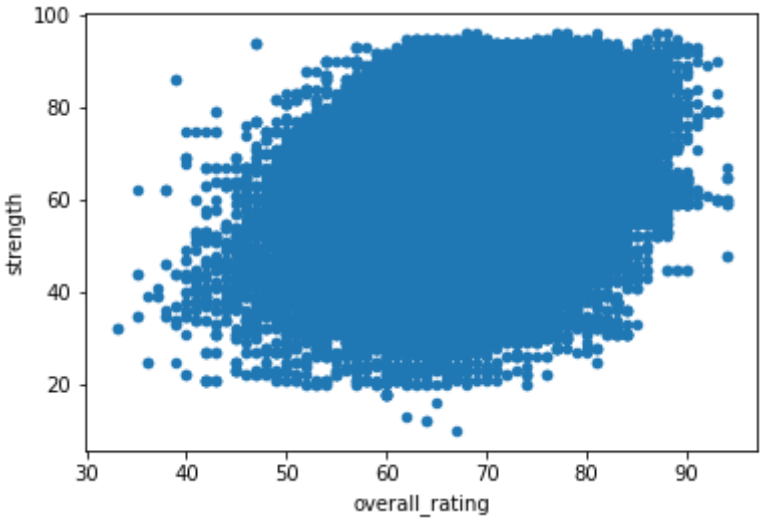
In [90]:
```
#correlation marking and rating
df_player.plot(x='overall_rating', y='marking', kind='scatter');
```



In [91]:
```
#correlation between dribbling and rating
df_player.plot(x='overall_rating', y='dribbling', kind='scatter');
```



In [92]:
```
#correlation between strength and rating
df_player.plot(x='overall_rating', y='strength', kind='scatter');
```



Conclusions

> I found that all leagues have not same number of matches in one season, So each league have diffrent number of teams England Premier League & France Ligue 1 & Italy Serie A & Spain LIGA BBVA have the most games: 380 Matches.
>
> The most league had Draw games is France Ligue 1 108 games than England Premier League 107 games.
>
> The most league had Win or lose games is Spain LIGA BBVA 288 games then Italy Serie A 285 games.
>
> the fewest team had losing matches in the 2016 season is Paris Saint-Germain which only lose 2 game.
>
> The league that had the most score a Goals in 2015/2016 season is England Premier League 1026 goals then Spain LIGA BBVA 1043 goals.
>
> The most team had won in 2015/16 is Paris Saint-Germain which win 30 game.

> From 2010 to 2016, the most improved teams by looking at the average Win times and goals are 'Paris Saint-Germain', 'Sporting CP', 'AZ', 'BSC Young Boys' and 'Napoli'.
>
> Rickie Lambert, Mario Balotelli, Xavi Hernandez, and Andrea Pirlo are the most penalty scorer in total.
>
> Most team attributes that lead the teams to win depend on the Change Creation Passing, defense pressure, Defense Aggression, build-up speed, and build dribbling column. Knowing that these results are according to the analysis of the top 10 winning teams
>
> The count of Players who have an overall rating of more than 90 is 12 players
>
> Most Top Player attributes depend on a balanced play speed, shot power, dribbling, strength.