Project: Investigating European Soccer Database

Table of Contents

- Introduction
- Data Wrangling
- Exploratory Data Analysis
- Conclusion

Introduction

> The data to be investigated in this project is European Soccer Database
> This data is based on SQLite Database, it covesrs more than 25,000 matches in 11 European Countries
> To know more about European Soccer Database, check Dataset on Kaggle
>
> **Through investigating European Soccer Database, the following questions will be explored:**
>
> - Which player had the most penalties?
> - Leagues performing better goals than other leagues?

In [1]:
```
# Importing necessary libraries

%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sqlite3 as sq3
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Data Wrangling

> At this step, we are going to load our tables from the SQLite database
> All tables included in the database will be examined to know the relationship between them.

General Properties

In [2]:
```
# create a connection to our European Soccer database
con = sq3.connect('Desktop\database.sqlite')
```

In [3]:
```
# check some meta information about the multiple tables included in the database
con.execute('select * FROM sqlite_master').fetchall()
```

```
  'sqlite_sequence',
  'sqlite_sequence',
  4,
  'CREATE TABLE sqlite_sequence(name,seq)'),
 ('table',
  'Player_Attributes',
  'Player_Attributes',
  11,
  'CREATE TABLE "Player_Attributes" (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMEN
T,\n\t`player_fifa_api_id`\tINTEGER,\n\t`player_api_id`\tINTEGER,\n\t`date`\tTEX
T,\n\t`overall_rating`\tINTEGER,\n\t`potential`\tINTEGER,\n\t`preferred_foot`\tTEX
T,\n\t`attacking_work_rate`\tTEXT,\n\t`defensive_work_rate`\tTEXT,\n\t`crossing`\t
INTEGER,\n\t`finishing`\tINTEGER,\n\t`heading_accuracy`\tINTEGER,\n\t`short_passin
g`\tINTEGER,\n\t`volleys`\tINTEGER,\n\t`dribbling`\tINTEGER,\n\t`curve`\tINTEGE
R,\n\t`free_kick_accuracy`\tINTEGER,\n\t`long_passing`\tINTEGER,\n\t`ball_control`
\tINTEGER,\n\t`acceleration`\tINTEGER,\n\t`sprint_speed`\tINTEGER,\n\t`agility`\tI
NTEGER,\n\t`reactions`\tINTEGER,\n\t`balance`\tINTEGER,\n\t`shot_power`\tINTEGE
R,\n\t`jumping`\tINTEGER,\n\t`stamina`\tINTEGER,\n\t`strength`\tINTEGER,\n\t`long_
shots`\tINTEGER,\n\t`aggression`\tINTEGER,\n\t`interceptions`\tINTEGER,\n\t`positi
oning`\tINTEGER,\n\t`vision`\tINTEGER,\n\t`penalties`\tINTEGER,\n\t`marking`\tINTE
GER,\n\t`standing_tackle`\tINTEGER,\n\t`sliding_tackle`\tINTEGER,\n\t`gk_diving`\t
INTEGER,\n\t`gk_handling`\tINTEGER,\n\t`gk_kicking`\tINTEGER,\n\t`gk_positioning`
\tINTEGER,\n\t`gk_reflexes`\tINTEGER,\n\tFOREIGN KEY(`player_fifa_api_id`) REFEREN
CES `Player`(`player_fifa_api_id`),\n\tFOREIGN KEY(`player_api_id`) REFERENCES `Pl
ayer`(`player_api_id`)\n)'),
 ('table',
  'Player',
  'Player',
  14,
  'CREATE TABLE `Player` (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMENT,\n\t`player_
api_id`\tINTEGER UNIQUE,\n\t`player_name`\tTEXT,\n\t`player_fifa_api_id`\tINTEGER
UNIQUE,\n\t`birthday`\tTEXT,\n\t`height`\tINTEGER,\n\t`weight`\tINTEGER\n)'),
 ('index', 'sqlite_autoindex_Player_1', 'Player', 15, None),
 ('index', 'sqlite_autoindex_Player_2', 'Player', 17, None),
 ('table',
  'Match',
  'Match',
  18,
  'CREATE TABLE `Match` (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMENT,\n\t`country_
id`\tINTEGER,\n\t`league_id`\tINTEGER,\n\t`season`\tTEXT,\n\t`stage`\tINTEGER,\n\t
`date`\tTEXT,\n\t`match_api_id`\tINTEGER UNIQUE,\n\t`home_team_api_id`\tINTEGER,\n
\t`away_team_api_id`\tINTEGER,\n\t`home_team_goal`\tINTEGER,\n\t`away_team_goal`\t
INTEGER,\n\t`home_player_X1`\tINTEGER,\n\t`home_player_X2`\tINTEGER,\n\t`home_play
er_X3`\tINTEGER,\n\t`home_player_X4`\tINTEGER,\n\t`home_player_X5`\tINTEGER,\n\t`h
ome_player_X6`\tINTEGER,\n\t`home_player_X7`\tINTEGER,\n\t`home_player_X8`\tINTEGE
R,\n\t`home_player_X9`\tINTEGER,\n\t`home_player_X10`\tINTEGER,\n\t`home_player_X1
1`\tINTEGER,\n\t`away_player_X1`\tINTEGER,\n\t`away_player_X2`\tINTEGER,\n\t`away_
player_X3`\tINTEGER,\n\t`away_player_X4`\tINTEGER,\n\t`away_player_X5`\tINTEGER,\n
\t`away_player_X6`\tINTEGER,\n\t`away_player_X7`\tINTEGER,\n\t`away_player_X8`\tIN
TEGER,\n\t`away_player_X9`\tINTEGER,\n\t`away_player_X10`\tINTEGER,\n\t`away_playe
r_X11`\tINTEGER,\n\t`home_player_Y1`\tINTEGER,\n\t`home_player_Y2`\tINTEGER,\n\t`h
ome_player_Y3`\tINTEGER,\n\t`home_player_Y4`\tINTEGER,\n\t`home_player_Y5`\tINTEGE
R,\n\t`home_player_Y6`\tINTEGER,\n\t`home_player_Y7`\tINTEGER,\n\t`home_player_Y8`
\tINTEGER,\n\t`home_player_Y9`\tINTEGER,\n\t`home_player_Y10`\tINTEGER,\n\t`home_p
layer_Y11`\tINTEGER,\n\t`away_player_Y1`\tINTEGER,\n\t`away_player_Y2`\tINTEGER,\n
\t`away_player_Y3`\tINTEGER,\n\t`away_player_Y4`\tINTEGER,\n\t`away_player_Y5`\tIN
TEGER,\n\t`away_player_Y6`\tINTEGER,\n\t`away_player_Y7`\tINTEGER,\n\t`away_player
_Y8`\tINTEGER,\n\t`away_player_Y9`\tINTEGER,\n\t`away_player_Y10`\tINTEGER,\n\t`aw
ay_player_Y11`\tINTEGER,\n\t`home_player_1`\tINTEGER,\n\t`home_player_2`\tINTEGE
R,\n\t`home_player_3`\tINTEGER,\n\t`home_player_4`\tINTEGER,\n\t`home_player_5`\tI
NTEGER,\n\t`home_player_6`\tINTEGER,\n\t`home_player_7`\tINTEGER,\n\t`home_player_
8`\tINTEGER,\n\t`home_player_9`\tINTEGER,\n\t`home_player_10`\tINTEGER,\n\t`home_p
layer_11`\tINTEGER,\n\t`away_player_1`\tINTEGER,\n\t`away_player_2`\tINTEGER,\n\t`
```

```
away_player_3`\tINTEGER,\n\t`away_player_4`\tINTEGER,\n\t`away_player_5`\tINTEGE
R,\n\t`away_player_6`\tINTEGER,\n\t`away_player_7`\tINTEGER,\n\t`away_player_8`\tI
NTEGER,\n\t`away_player_9`\tINTEGER,\n\t`away_player_10`\tINTEGER,\n\t`away_player
_11`\tINTEGER,\n\t`goal`\tTEXT,\n\t`shoton`\tTEXT,\n\t`shotoff`\tTEXT,\n\t`foulcom
mit`\tTEXT,\n\t`card`\tTEXT,\n\t`cross`\tTEXT,\n\t`corner`\tTEXT,\n\t`possession`
\tTEXT,\n\t`B365H`\tNUMERIC,\n\t`B365D`\tNUMERIC,\n\t`B365A`\tNUMERIC,\n\t`BWH`\tN
UMERIC,\n\t`BWD`\tNUMERIC,\n\t`BWA`\tNUMERIC,\n\t`IWH`\tNUMERIC,\n\t`IWD`\tNUMERI
C,\n\t`IWA`\tNUMERIC,\n\t`LBH`\tNUMERIC,\n\t`LBD`\tNUMERIC,\n\t`LBA`\tNUMERIC,\n\t
`PSH`\tNUMERIC,\n\t`PSD`\tNUMERIC,\n\t`PSA`\tNUMERIC,\n\t`WHH`\tNUMERIC,\n\t`WHD`
\tNUMERIC,\n\t`WHA`\tNUMERIC,\n\t`SJH`\tNUMERIC,\n\t`SJD`\tNUMERIC,\n\t`SJA`\tNUME
RIC,\n\t`VCH`\tNUMERIC,\n\t`VCD`\tNUMERIC,\n\t`VCA`\tNUMERIC,\n\t`GBH`\tNUMERIC,\n
\t`GBD`\tNUMERIC,\n\t`GBA`\tNUMERIC,\n\t`BSH`\tNUMERIC,\n\t`BSD`\tNUMERIC,\n\t`BSA
`\tNUMERIC,\n\tFOREIGN KEY(`country_id`) REFERENCES `country`(`id`),\n\tFOREIGN KE
Y(`league_id`) REFERENCES `League`(`id`),\n\tFOREIGN KEY(`home_team_api_id`) REFER
ENCES `Team`(`team_api_id`),\n\tFOREIGN KEY(`away_team_api_id`) REFERENCES `Team`
(`team_api_id`),\n\tFOREIGN KEY(`home_player_1`) REFERENCES `Player`(`player_api_i
d`),\n\tFOREIGN KEY(`home_player_2`) REFERENCES `Player`(`player_api_id`),\n\tFORE
IGN KEY(`home_player_3`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`hom
e_player_4`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`home_player_5`)
REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`home_player_6`) REFERENCES `
Player`(`player_api_id`),\n\tFOREIGN KEY(`home_player_7`) REFERENCES `Player`(`pla
yer_api_id`),\n\tFOREIGN KEY(`home_player_8`) REFERENCES `Player`(`player_api_id
`),\n\tFOREIGN KEY(`home_player_9`) REFERENCES `Player`(`player_api_id`),\n\tFOREI
GN KEY(`home_player_10`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`hom
e_player_11`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`away_player_1
`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`away_player_2`) REFERENCE
S `Player`(`player_api_id`),\n\tFOREIGN KEY(`away_player_3`) REFERENCES `Player`(`
player_api_id`),\n\tFOREIGN KEY(`away_player_4`) REFERENCES `Player`(`player_api_i
d`),\n\tFOREIGN KEY(`away_player_5`) REFERENCES `Player`(`player_api_id`),\n\tFORE
IGN KEY(`away_player_6`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`awa
y_player_7`) REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`away_player_8`)
REFERENCES `Player`(`player_api_id`),\n\tFOREIGN KEY(`away_player_9`) REFERENCES `
Player`(`player_api_id`),\n\tFOREIGN KEY(`away_player_10`) REFERENCES `Player`(`pl
ayer_api_id`),\n\tFOREIGN KEY(`away_player_11`) REFERENCES `Player`(`player_api_id
`)\n)'),
 ('index', 'sqlite_autoindex_Match_1', 'Match', 19, None),
 ('table',
  'League',
  'League',
  24,
  'CREATE TABLE `League` (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMENT,\n\t`country
_id`\tINTEGER,\n\t`name`\tTEXT UNIQUE,\n\tFOREIGN KEY(`country_id`) REFERENCES `co
untry`(`id`)\n)'),
 ('index', 'sqlite_autoindex_League_1', 'League', 25, None),
 ('table',
  'Country',
  'Country',
  26,
  'CREATE TABLE `Country` (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMENT,\n\t`name`
\tTEXT UNIQUE\n)'),
 ('index', 'sqlite_autoindex_Country_1', 'Country', 28, None),
 ('table',
  'Team',
  'Team',
  29,
  'CREATE TABLE "Team" (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMENT,\n\t`team_api_
id`\tINTEGER UNIQUE,\n\t`team_fifa_api_id`\tINTEGER,\n\t`team_long_name`\tTEXT,\n
\t`team_short_name`\tTEXT\n)'),
 ('index', 'sqlite_autoindex_Team_1', 'Team', 30, None),
 ('table',
  'Team_Attributes',
  'Team_Attributes',
  2,
  'CREATE TABLE `Team_Attributes` (\n\t`id`\tINTEGER PRIMARY KEY AUTOINCREMENT,\n
```

```
\t`team_fifa_api_id`\tINTEGER,\n\t`team_api_id`\tINTEGER,\n\t`date`\tTEXT,\n\t`buildUpPlaySpeed`\tINTEGER,\n\t`buildUpPlaySpeedClass`\tTEXT,\n\t`buildUpPlayDribbling`\tINTEGER,\n\t`buildUpPlayDribblingClass`\tTEXT,\n\t`buildUpPlayPassing`\tINTEGER,\n\t`buildUpPlayPassingClass`\tTEXT,\n\t`buildUpPlayPositioningClass`\tTEXT,\n\t`chanceCreationPassing`\tINTEGER,\n\t`chanceCreationPassingClass`\tTEXT,\n\t`chanceCreationCrossing`\tINTEGER,\n\t`chanceCreationCrossingClass`\tTEXT,\n\t`chanceCreationShooting`\tINTEGER,\n\t`chanceCreationShootingClass`\tTEXT,\n\t`chanceCreationPositioningClass`\tTEXT,\n\t`defencePressure`\tINTEGER,\n\t`defencePressureClass`\tTEXT,\n\t`defenceAggression`\tINTEGER,\n\t`defenceAggressionClass`\tTEXT,\n\t`defenceTeamWidth`\tINTEGER,\n\t`defenceTeamWidthClass`\tTEXT,\n\t`defenceDefenderLineClass`\tTEXT,\n\tFOREIGN KEY(`team_fifa_api_id`) REFERENCES `Team`(`team_fifa_api_id`),\n\tFOREIGN KEY(`team_api_id`) REFERENCES `Team`(`team_api_id`)\n)')]
```

In [4]:
```
# Exploring Player_Attributes Table
Player_Attributes = pd.read_sql('select * FROM Player_Attributes', con)
```

In [5]:
```
Player_Attributes.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 [6]:
```
# Selecting first 5 rows in Player_Attributes Table
Player_Attributes.head(5)
```

Out[6]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking
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 [7]:
```
# Exploring Player Table
Player =  pd.read_sql('select * FROM Player', con)
```

In [8]:
```
# Selecting the first 5 rows in Player Table
Player.head(5)
```

Out[8]:

	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 [9]:
```
# Exploring Team Table
Team = pd.read_sql('select * FROM Team', con)
```

In [10]:
```
# selecting first 5 rows in Team Table
Team.head(5)
```

Out[10]:

	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 [11]:
```
# Exploring Match Table
Match = pd.read_sql('select * FROM Match', con)
```

In [12]:
```
# Displaying all Columns of Match Table
for col in Match.columns:
    print(col)
```

id
country_id
league_id
season
stage
date
match_api_id
home_team_api_id
away_team_api_id
home_team_goal
away_team_goal
home_player_X1
home_player_X2
home_player_X3
home_player_X4
home_player_X5
home_player_X6
home_player_X7
home_player_X8
home_player_X9
home_player_X10
home_player_X11
away_player_X1
away_player_X2
away_player_X3
away_player_X4
away_player_X5
away_player_X6
away_player_X7
away_player_X8
away_player_X9
away_player_X10
away_player_X11
home_player_Y1
home_player_Y2
home_player_Y3
home_player_Y4
home_player_Y5
home_player_Y6
home_player_Y7
home_player_Y8
home_player_Y9
home_player_Y10
home_player_Y11
away_player_Y1
away_player_Y2
away_player_Y3
away_player_Y4
away_player_Y5
away_player_Y6
away_player_Y7
away_player_Y8
away_player_Y9
away_player_Y10
away_player_Y11
home_player_1
home_player_2
home_player_3
home_player_4
home_player_5
home_player_6
home_player_7
home_player_8
home_player_9

```
home_player_10
home_player_11
away_player_1
away_player_2
away_player_3
away_player_4
away_player_5
away_player_6
away_player_7
away_player_8
away_player_9
away_player_10
away_player_11
goal
shoton
shotoff
foulcommit
card
cross
corner
possession
B365H
B365D
B365A
BWH
BWD
BWA
IWH
IWD
IWA
LBH
LBD
LBA
PSH
PSD
PSA
WHH
WHD
WHA
SJH
SJD
SJA
VCH
VCD
VCA
GBH
GBD
GBA
BSH
BSD
BSA
```

In [13]:

```
# Selecting the first 5 rows of Match Table
Match.head(5)
```

Out[13]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	away_te
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 [14]:
```
# Exploring League Table
League = pd.read_sql('select * FROM League', con)
```

In [15]:
```
# displaying table information
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
```

In [16]:
```
# displaying all the rows from League table since it is only 11 enteries
League
```

Out[16]:

	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
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 [17]:
```
# Exploring Country Table
Country = pd.read_sql('select * FROM Country', con)
```

In [18]:
```
# displaying table information
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
```

In [19]:
```
# displaying all the rows since the table is only 11 enteries
Country
```

Out[19]:

	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 [20]:
```
# Exploring Team Table
Team = pd.read_sql('select * FROM Team', con)
```

In [21]:
```
# selecting the first 5 rows in Team table
Team.head(5)
```

Out[21]:

	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 [22]:
```
# Exploring Team_Attributes Table
Team_Attributes = pd.read_sql('select * FROM Team_Attributes', con)
```

In [23]:
```
# Selecting the first 5 rows in Team_Attributes table
Team_Attributes.head(5)
```

Out[23]:

	id	team_fifa_api_id	team_api_id	date	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpP
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

Data Cleaning (checking for duplicates or null Values)

In [24]:
```
# Tables at use will be checked for any duplicates or Null Values
```

In [25]:
```
# Checking for duplicates at player table
Player.duplicated().sum()
```

Out[25]:
```
0
```

In [26]:
```
# Checking for nullvalues at player table
Player.isnull().sum().sum()
```

Out[26]:
```
0
```

In [27]:
```
# Checking for duplicates at Player_Attributes table
Player_Attributes.duplicated().sum()
```

Out[27]:
```
0
```

In [28]:
```
# Checking for nullvalues at Player_Attributes table
Player_Attributes.isnull().sum().sum()
```

Out[28]:
```
47301
```

In [29]:
```
# dropping the null values found at Player_Attributes table
Player_Attributes.dropna(inplace=True)

# checking if the null values are removed from Player_Attributes
Player_Attributes.isnull().sum().sum()
```

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

In [30]:
```
# Checking for duplicates at Team table
Team.duplicated().sum()
```

Out[30]:
```
0
```

In [31]:
```
# Checking for null values at Team table
Team.isnull().sum().sum()
```

Out[31]:
```
11
```

In [32]:
```
# dropping the null values found at Team table
Team.dropna(inplace=True)

# checking if the null values are removed from Team table
Team.isnull().sum().sum()
```

Out[32]:
```
0
```

In [33]:
```
# Checking for duplicates at Match table
Match.duplicated().sum()
```

Out[33]:
```
0
```

In [34]:
```
# Checking for null values at Match table
Match.isnull().sum().sum()
```

Out[34]:
```
407395
```

In [35]:
```
# dropping null values from Match table
Match.dropna(inplace=True)

#checking if the values are removed from Match table
Match.isnull().sum().sum()
```

Out[35]:
```
0
```

In [36]:
```
# Checking for null values at League table
League.isnull().sum().sum()
```

Out[36]:
```
0
```

In [37]:
```
# Checking for duplicates at League table
League.duplicated().sum()
```

Out[37]:
```
0
```

In [38]:
```
# Checking for null values at Country table
Country.isnull().sum().sum()
```

Out[38]:
```
0
```

In [39]:
```
# Checking for duplicates at Country table
Country.duplicated().sum()
```

Out[39]:
```
0
```

In [40]:
```
# Checking for duplicates at Team table
Team.duplicated().sum()
```

Out[40]:
```
0
```

In [41]:
```
# Checking for duplicates at Team_Attributes table
Team_Attributes.duplicated().sum()
```

Out[41]:
```
0
```

In [42]:
```
# Checking for null values at Team_Attributes table
Team_Attributes.isnull().sum().sum()
```

Out[42]:
```
969
```

In [43]:
```
# dropping null values from Team_Attributes table
Team_Attributes.dropna(inplace=True)

# confirming removing the null values from Team_Attributes
Team_Attributes.isnull().sum().sum()
```

Out[43]:
```
0
```

Exploratory Data Analysis

> Now that the data is trimmed and cleaned, it's time for data exploration. At this step, Questions posed at the beginning will be addressed here and have in depth insight about the exploration process and how we get to the conclusions presented.

Research Question 1 (Which player had the most penalties?)

> To answer this question we need to get the name of the players from **Player** table and their penalites from **Player_Attributes** table.
> And Since these data are given in separate columns from different tables , they need merging for a complete form.

> SQL query will be used to do the selection of the needed columns from the different tables and join them.

In [44]:
```
# sql query for selecting the player names and the penalites columns using joins
players_penalties = pd.read_sql('select Player.player_name, Player_Attributes.penal
                        FROM Player_Attributes\
                        JOIN Player\
                        ON Player_Attributes.player_api_id = Player.player_api_id\
                        ORDER BY  Player_Attributes.date', con )
```

In [45]:
```
# displaying first 5 rows from the quey output
players_penalties.head(5)
```

Out[45]:

	player_name	penalties	date
0	Aaron Appindangoye	47.0	2007-02-22 00:00:00
1	Aaron Cresswell	29.0	2007-02-22 00:00:00
2	Aaron Doran	36.0	2007-02-22 00:00:00
3	Aaron Galindo	60.0	2007-02-22 00:00:00
4	Aaron Hughes	81.0	2007-02-22 00:00:00

In [46]:
```
# selecting the player of the highest penalities through idxmax method
players_penalties.loc[players_penalties.penalties.idxmax()]
```

Out[46]:
```
player_name          Rickie Lambert
penalties                      96.0
date           2015-09-21 00:00:00
Name: 159711, dtype: object
```

In [47]:
```
# Listing the 10 largest penalities and the highlighting the largest number
players_penalties.nlargest(10, 'penalties')\
                 .style.highlight_max(color = 'lightgreen', subset = ['penalties']
```

Out[47]:

	player_name	penalties	date
159711	Rickie Lambert	96.000000	2015-09-21 00:00:00
165205	Rickie Lambert	96.000000	2015-10-23 00:00:00
169788	Rickie Lambert	96.000000	2015-12-24 00:00:00
177120	Rickie Lambert	96.000000	2016-03-10 00:00:00
16854	Andrea Pirlo	95.000000	2008-08-30 00:00:00
22113	Andrea Pirlo	95.000000	2009-02-22 00:00:00
34358	Paul Scholes	95.000000	2010-02-22 00:00:00
35317	Xavi Hernandez	95.000000	2010-02-22 00:00:00
82770	Mario Balotelli	95.000000	2013-04-26 00:00:00
84102	Mario Balotelli	95.000000	2013-05-10 00:00:00

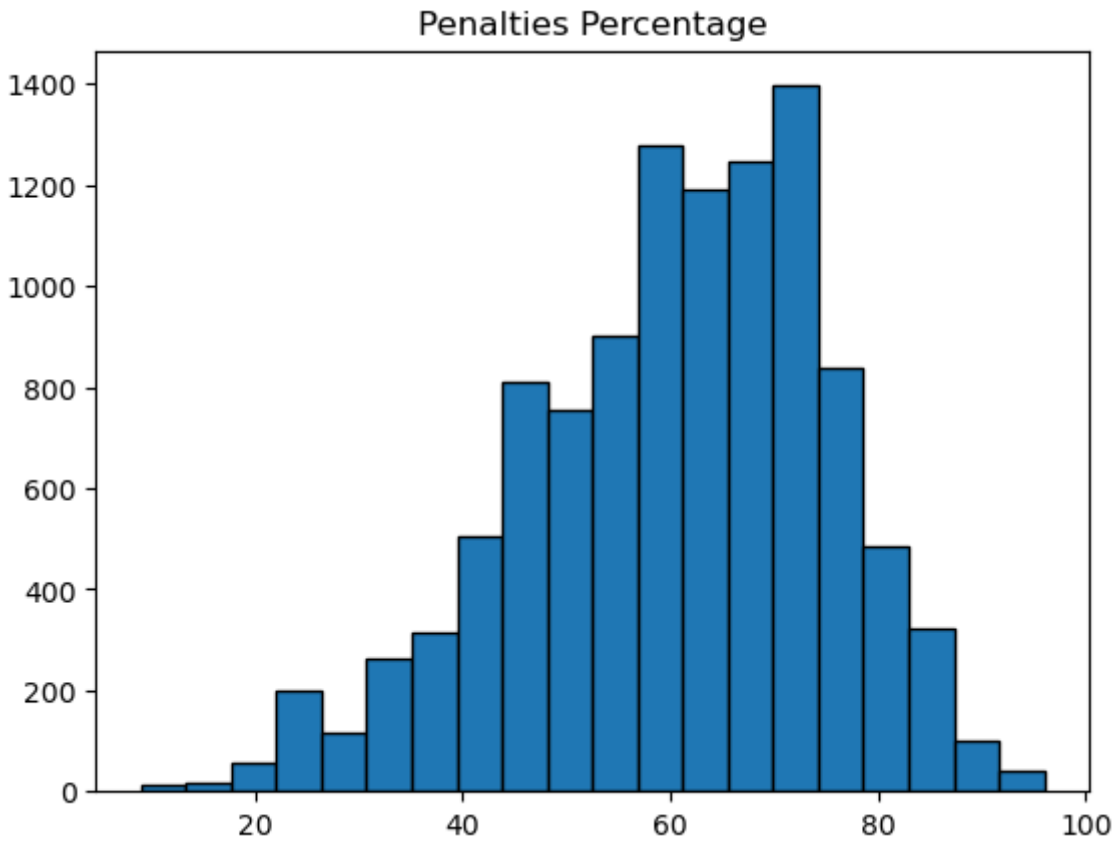
In [48]:
```
# Displaying Players with largest penalties
max_penalties = players_penalties.groupby(['player_name', 'date'])['penalties'].ma
                                 .groupby(level = 'player_name') \
                                 .nlargest(1).reset_index(level=1, drop=True) \
                                 .sort_values(ascending = False)
```

In [49]:
```
# printing out the largest scored penalties
max_penalties
```

Out[49]:
```
player_name             date
Rickie Lambert          2015-09-21 00:00:00    96.0
Andrea Pirlo            2008-08-30 00:00:00    95.0
Xavi Hernandez          2010-02-22 00:00:00    95.0
Mario Balotelli         2013-04-26 00:00:00    95.0
Paul Scholes            2010-02-22 00:00:00    95.0
                                               ...
Igor Stefanovic         2007-02-22 00:00:00    11.0
Giedrius Arlauskis      2013-09-20 00:00:00    11.0
Jakub Szumski           2013-09-20 00:00:00    11.0
Timothy van der Meulen  2007-02-22 00:00:00    10.0
Jakub Divis             2007-02-22 00:00:00     9.0
Name: penalties, Length: 10848, dtype: float64
```

In [50]:
```
# plotting the largest scored penalties
max_penalties.hist(bins=20, edgecolor = 'black', grid=False)
plt.title('Penalties Percentage')
```

Out[50]:
```
Text(0.5, 1.0, 'Penalties Percentage')
```



> **Conclusion: We conclude that Rickie Lambert has the largest penalties amongest other players scoring 96**

Research Question 2 (Which League Outperformed the other leagues and scored more goals through seasons??)

> To get insights about that question, we need to get the names of the leagues and the goals scored whether at home team or away team.
> Total goals (home team goals plus away team goals) can also be calculated to be used as an indicator of the performance of a league.
> For example, if the number of total goals of a league is very large compared to that of other leagues, this league has a better performance or good offensive ability.

In [51]:
```
# SQL Query to select home_team_goal, away_team_goal and seasons from match and leg
league_goals = pd.read_sql('select Match.home_team_goal ,Match.away_team_goal , Lea
                           FROM Match\
                           JOIN League\
                           ON Match.country_id = League.country_id', con)
```

In [52]:
```
# displaying the first 24 data from the prev query grouping results by League and s
league_goals.groupby(['league', 'season'])['home_team_goal','away_team_goal'].sum(
```

Out[52]:

league	season	home_team_goal	away_team_goal
Belgium Jupiler League	2008/2009	499	356
	2009/2010	308	257
	2010/2011	382	253
	2011/2012	421	270
	2012/2013	375	328
	2013/2014	18	12
	2014/2015	376	292
	2015/2016	402	292
England Premier League	2008/2009	532	410
	2009/2010	645	408
	2010/2011	617	446
	2011/2012	604	462
	2012/2013	592	471
	2013/2014	598	454
	2014/2015	560	415
	2015/2016	567	459
France Ligue 1	2008/2009	489	369
	2009/2010	528	388
	2010/2011	510	380
	2011/2012	560	396
	2012/2013	558	409
	2013/2014	538	395
	2014/2015	536	411
	2015/2016	546	414

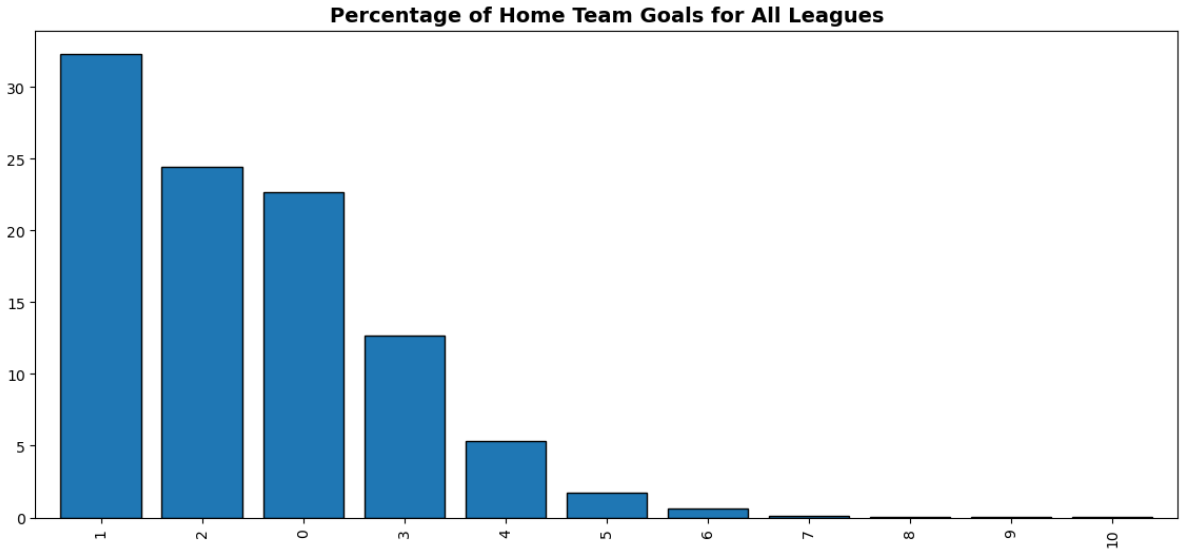
In [53]:
```
# Applying aggr function to get the total sum of home team goals and away team goal
league_goals.groupby('league')['home_team_goal','away_team_goal'].sum()
```

Out[53]:

league	home_team_goal	away_team_goal
Belgium Jupiler League	2781	2060
England Premier League	4715	3525
France Ligue 1	4265	3162
Germany 1. Bundesliga	3982	3121
Italy Serie A	4528	3367
Netherlands Eredivisie	4357	3185
Poland Ekstraklasa	2678	1978
Portugal Liga ZON Sagres	2890	2311
Scotland Premier League	2607	2197
Spain LIGA BBVA	4959	3453
Switzerland Super League	2365	1801

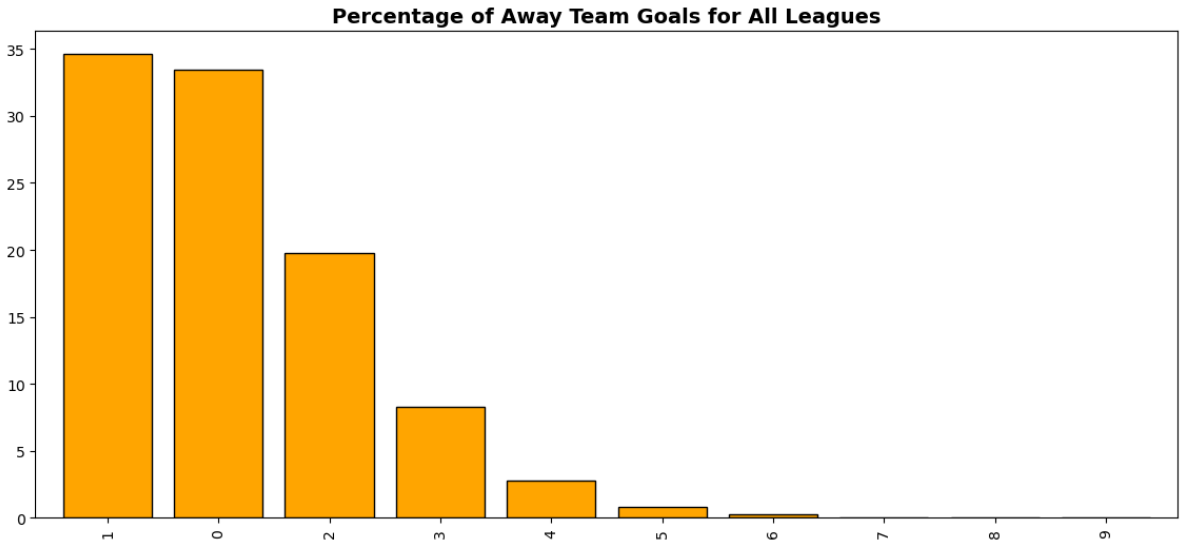
In [54]:
```
# Plotting the home team goal scored in all leagues
league_goals['home_team_goal'].value_counts(normalize=True)\
                              .mul(100).plot.bar(edgecolor='black', figsize=[14,6]
plt.title('Percentage of Home Team Goals for All Leagues', fontsize = 14, weight =
```

Out[54]:
```
Text(0.5, 1.0, 'Percentage of Home Team Goals for All Leagues')
```



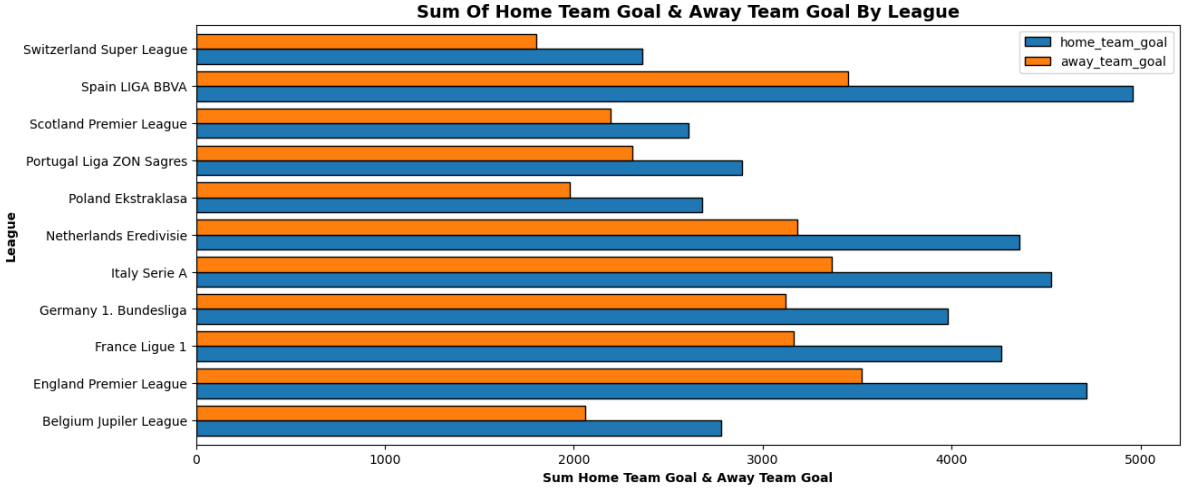
In [55]:
```
# Plotting the away team goal scored in all leagues
league_goals['away_team_goal'].value_counts(normalize=True)\
                              .mul(100).plot.bar(edgecolor='black', color='orange'
plt.title('Percentage of Away Team Goals for All Leagues', fontsize = 14, weight =
```

Out[55]:
```
Text(0.5, 1.0, 'Percentage of Away Team Goals for All Leagues')
```

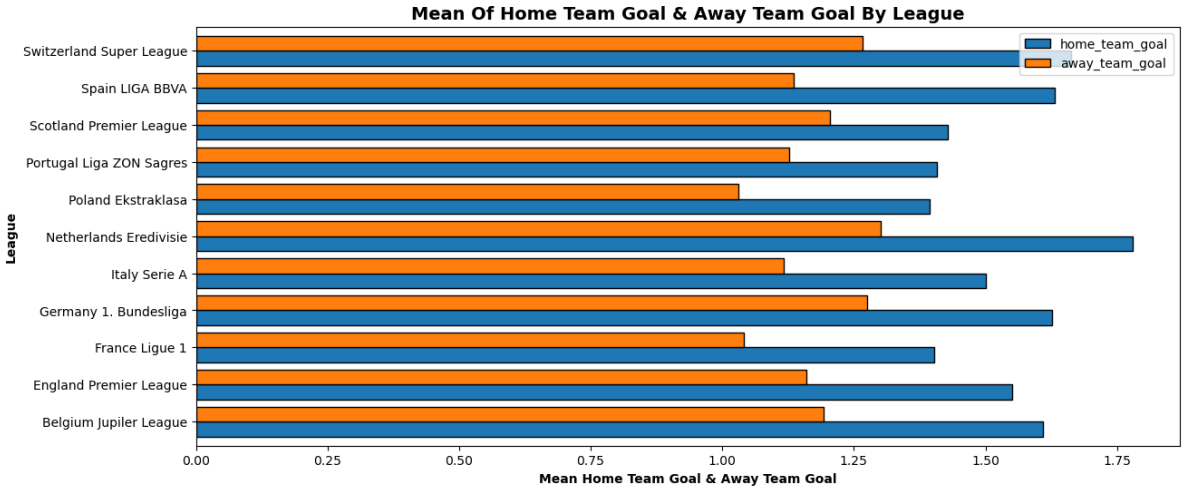


In [56]:
```
# creating a plotting function (LeaguePlot) for Group horizontal bar plottng
def LeaguePlot(df, grbVar, agrVar, stat= 'sum'):
    '''
    This function is created to easily plot groups data in a horizontal bar
    inputs: df(DataFrame)
            Grouping Columns (grbVar),
            Aggregating Columns (agrVar)
            aggregating functions (stat) => Optional Argument
    Output: Horizontal Bar plottng

    '''
    df.groupby([grbVar])[agrVar].agg(stat).plot.barh(edgecolor='black', figsize=[1
    # replace _ with a space for leagues names
    grbVar=grbVar.replace('_', ' ')
    # # replace _ with a space for Aggregating Column(s)
    if isinstance(agrVar, list):
        agrVar=' & '.join([x.replace("_", " ") for x in agrVar])
    else:
        agrVar=agrVar.replace('_', ' ')

    # Add  title format it
    plt.title(f'{stat} of {agrVar} by {grbVar}'.title(), fontsize = 14, weight = '
    # Add y labale and format it
    plt.ylabel(grbVar.title(), fontsize = 10, weight = 'bold')
    # Add x labale and format it
    plt.xlabel(f'{stat} {agrVar}'.title(), fontsize = 10, weight = 'bold')
```

In [57]:
```
# calling LeaguePlot passing home_team_goal and away_team_goal arguments to visual
LeaguePlot(league_goals, 'league', ['home_team_goal','away_team_goal'])
```

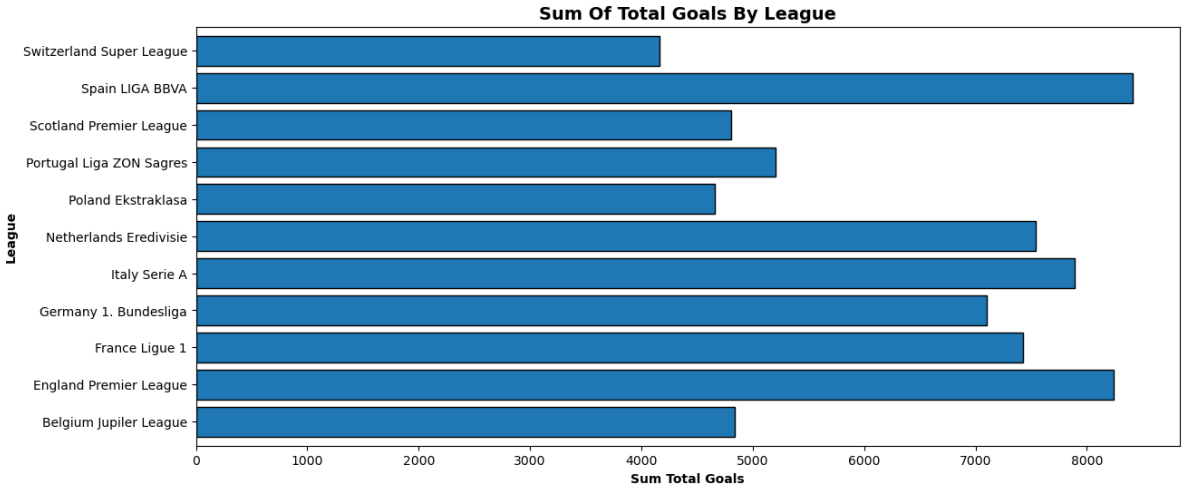


In [58]:
```
# calling LeaguePlot function to visualize average of home_team_goal and away_team_
LeaguePlot(league_goals, 'league', ['home_team_goal','away_team_goal'], stat= 'mea
```

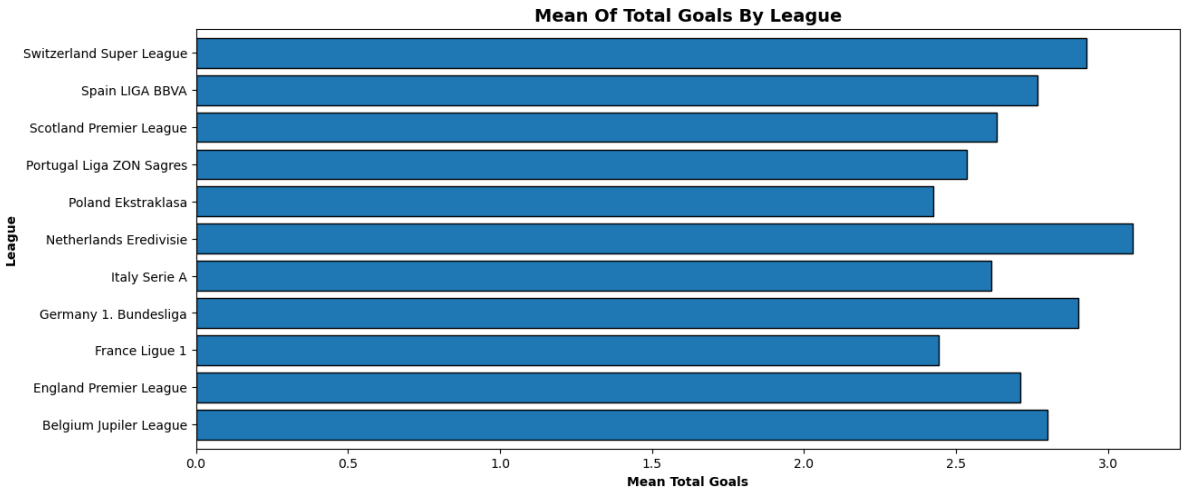


In [59]:
```
# displaying a column for the total goals (Home team goals + away team goals)
league_goals['total_goals'] = league_goals['home_team_goal'] + league_goals['away_
```

In [60]:
```
# calling LeaguePlot function to visualize sum of leagues total goals
LeaguePlot(league_goals, 'league', 'total_goals')
```



In [61]:
```
# calling LeaguePlot function to visualize average total goals of leagues
LeaguePlot(league_goals, 'league', 'total_goals', stat= 'mean'  )
```



Conclusion: We conclude that Spain LIGA BBVA outperformed other leaugues considering the Goals through (2008 Till 2015) Seasons

Conclusion

In this project we investigated European Soccer Database, containing data about 25.000+ matches from 2008 to 2016.
The database is complex as it has a lot of separate tables, which could be joined to model a relational data structure.
SQL Queries were used to extract data from tables into dataframed to be easily accessed, data has a lot of missing data, so through data cleaning duplicates and null values were addressed.

In this project we analyzed:

> - Which player had the most penalties?
> - Leagues performing better goals than other leagues?

After Applying the needed statistical analysis, we can find out that that **Rickie Lambert** has the "largest penalties" amongest other players scoring 96.

We could also visualize that the “league outperformed others ” by scoring most of the goals is **Spain LIGA BBVA**.

In overview, this project focused on the big picture analysis and hadn’t an intention to imply statistical inference performed in the analysis to test the significance of the results found.

But for a deeper research a suggestion to the area covered in this project can be found at:

> - Rein R, Memmert D. Big data and tactical analysis in elite soccer: Future challenges and opportunities for sports science. Springerplus 2016
> - (Handbook of Statistical Methods and Analyses in Sports) Book
> - Zhang S. Home advantage in soccer. PIT Journal 2015