Project 2

CS 5/7394 - Applied Machine Learning

Matt Bartos and Jarod Cox

- **Due** - March 11 @ 11:59 pm pushed to Github repo
- **Teams** - You can do this project solo or in pairs. Not 3, not 4 not 5... Max of 2. If a 5394 student pairs with a 7394 student, the pair needs to do the 7394 work.

Below are 6 Kaggle Datasets. You will choose 1 to work with for this project.

- Airfare Prediction Dataset
- Chinese Rest Holiday Dataset
- Jigsaw Toxic Comment Classification Challenge
- Latest Covid 19 Dataset Worldwide
- Trains
- Football Data top 5 Leagues

Merging disparate datasets is a staple of the data exploration process. Therefore, for which ever data set above that you choose, you will need to independently find **an additional** dataset to merge with your selection. The only requirement is that it add to the richness of the original dataset. Students in the 7000-level version of the class need to find two additional data sets to merge with the original selection.

Note: If you want to start with a different data set, you need to get Fontenot's OK first.

Your Tasks

Below, there are cells that provide directions on what to do for the project.

You can insert as many cells between the ones below as you'd like, but please **Do NOT** change the cells already provided.

Part 1 - Getting Started

- Import libraries
- Load original Data (which ever one you chose from the provided list) into a data frame.
- Load your additional data set(s) into a data frame.
- In a markdown cell, provide a brief description of your the data sets you've chosen to work with.
- Develop a list of 3 - 4 questions that you hope to be able to answer after the exploration of the data and write them in this section.

In [1]:
```
# Import libraries
import pandas as pd
```

```
import numpy as np
```

In [2]:

```
# Load original Data
df = pd.read_csv('./datasets/combined_data.csv')
df.head(10)
```

Out[2]:

	Unnamed: 0	Home Team	Away Team	Score	Half Time Score	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Poss
0	0	MAN UTD	SWANSEA	1-2	0-1	5.9	5.6	7.6	60	
1	1	WEST BROM	SUNDERLAND	2-2	1-1	7.3	6.5	7.4	58	
2	2	LEICESTER CITY	EVERTON	2-2	1-2	7.0	6.5	6.3	37	
3	3	WEST HAM	TOTTENHAM	0-1	0-0	4.8	5.9	6.4	47	
4	4	QPR	HULL CITY	0-1	0-0	3.8	5.7	6.6	51	
5	5	STOKE	ASTON VILLA	0-1	0-0	2.8	6.5	7.0	63	
6	6	ARSENAL	CRYSTAL PALACE	2-1	1-1	5.8	7.7	5.6	76	
7	7	LIVERPOOL	SOUTHAMPTON	2-1	1-0	6.0	7.1	6.7	56	
8	8	NEWCASTLE	MAN CITY	0-2	0-1	4.6	5.5	8.1	44	
9	9	BURNLEY	CHELSEA	1-3	1-3	5.3	5.3	7.4	39	

10 rows × 42 columns

In [3]:

```
# Load additional data
# 1) Complete Player Database from 2019 (https://www.kaggle.com/karangadiya/fifa19)
df_player = pd.read_csv('./datasets/complete_player_dataset.csv')
df_player.drop(columns=['Photo', 'Flag', 'Club Logo', 'Unnamed: 0'], inplace=True)
df_player.head(10)
```

Out[3]:

	ID	Name	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	...	Con
0	158023	L. Messi	31	Argentina	94	94	FC Barcelona	€110.5M	€565K	2202	...	
1	20801	Cristiano Ronaldo	33	Portugal	94	94	Juventus	€77M	€405K	2228	...	
2	190871	Neymar Jr	26	Brazil	92	93	Paris Saint-Germain	€118.5M	€290K	2143	...	

	ID	Name	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	...	Con
3	193080	De Gea	27	Spain	91	93	Manchester United	€72M	€260K	1471	...	
4	192985	K. De Bruyne	27	Belgium	91	92	Manchester City	€102M	€355K	2281	...	
5	183277	E. Hazard	27	Belgium	91	91	Chelsea	€93M	€340K	2142	...	
6	177003	L. Modrić	32	Croatia	91	91	Real Madrid	€67M	€420K	2280	...	
7	176580	L. Suárez	31	Uruguay	91	91	FC Barcelona	€80M	€455K	2346	...	
8	155862	Sergio Ramos	32	Spain	91	91	Real Madrid	€51M	€380K	2201	...	
9	200389	J. Oblak	25	Slovenia	90	93	Atlético Madrid	€68M	€94K	1331	...	

10 rows × 85 columns

In [4]:
```
# Load additional data
# 2) International Match Datasets (https://www.kaggle.com/martj42/international-footbal
df_international = pd.read_csv('./datasets/international_data.csv')
df_international.tail(10)
```

Out[4]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	n
43178	2022-01-30	Senegal	Equatorial Guinea	3	1	African Cup of Nations	Yaoundé	Cameroon	
43179	2022-02-01	Bahrain	DR Congo	1	0	Friendly	Riffa	Bahrain	
43180	2022-02-01	Kuwait	Libya	0	2	Friendly	Kuwait City	Kuwait	
43181	2022-02-01	Nepal	Mauritius	1	0	Friendly	Kathmandu	Nepal	
43182	2022-02-01	Nicaragua	Belize	1	1	Friendly	Managua	Nicaragua	
43183	2022-02-01	Suriname	Guyana	2	1	Friendly	Paramaribo	Suriname	
43184	2022-02-02	Burkina Faso	Senegal	1	3	African Cup of Nations	Yaoundé	Cameroon	
43185	2022-02-03	Cameroon	Egypt	0	0	African Cup of Nations	Yaoundé	Cameroon	
43186	2022-02-05	Cameroon	Burkina Faso	3	3	African Cup of Nations	Yaoundé	Cameroon	

	date	home_team	away_team	home_score	away_score	tournament	city	country	n
43187	2022-02-06	Senegal	Egypt	0	0	African Cup of Nations	Yaoundé	Cameroon	

Dataset Descriptions:

Football Dataset: The dataset contains all information about club soccer matches from the top 5 football leagues ranging from 2014 to 2020. The dataset contains 42 columns each with information about the home team, away team, and the outcome of the match.

Player Dataset: The dataset contains information on all professional football players in the year 2019 sourced from FIFA. FIFA is the international governing body of football and EA sports' FIFA game ranks players based upon real life player performance and talent. The player dataset contains an overall rating for each player, the player's club, and the player's salary.

International Dataset: The dataset contains information about International football matches from the year 1872 to 2022. The football dataset focuses on the top 5 club leagues in the world. The international dataset focuses on a broader scope from international friendlies, regional tournaments, and the world cup.

List of questions to explore the data:

1. Which leagues are highest scoring? most fouls?
2. Does the home team win more games?
3. How does ball possesion relate to win ratio?
4. How do club games compare to international? (after mergining)

Part 2 - Data Inspection

Write some code to summarize the datasets. Think about the following questions:

- What type of data is each variable? (think like a data scientist here, not a computer scientist)
- What is the total size of the data sets?
- What time boundaries are there in the dataset? IOW, what time frame do they span?
- Are there any missing values in any of the variables?

Do this with Intentionality. Don't skimp.

Football Data Set Inspection

Type of each Variable

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12062 entries, 0 to 12061
Data columns (total 42 columns):
```

```
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Unnamed: 0                    12062 non-null  int64
 1   Home Team                     12062 non-null  object
 2   Away Team                     12062 non-null  object
 3   Score                         12062 non-null  object
 4   Half Time Score               12062 non-null  object
 5   Match Excitement              12062 non-null  float64
 6   Home Team Rating              12062 non-null  float64
 7   Away Team Rating              12062 non-null  float64
 8   Home Team Possession %        12062 non-null  int64
 9   Away Team Possession %        12062 non-null  int64
 10  Home Team Off Target Shots    12062 non-null  float64
 11  Home Team On Target Shots     12062 non-null  float64
 12  Home Team Total Shots         12062 non-null  float64
 13  Home Team Blocked Shots       12062 non-null  float64
 14  Home Team Corners             12062 non-null  float64
 15  Home Team Throw Ins           12062 non-null  float64
 16  Home Team Pass Success %      12062 non-null  float64
 17  Home Team Aerials Won         12062 non-null  float64
 18  Home Team Clearances          12062 non-null  float64
 19  Home Team Fouls               12062 non-null  float64
 20  Home Team Yellow Cards        12062 non-null  float64
 21  Home Team Second Yellow Cards 12062 non-null  float64
 22  Home Team Red Cards           12062 non-null  float64
 23  Away Team Off Target Shots    12062 non-null  float64
 24  Away Team On Target Shots     12062 non-null  float64
 25  Away Team Total Shots         12062 non-null  float64
 26  Away Team Blocked Shots       12062 non-null  float64
 27  Away Team Corners             12062 non-null  float64
 28  Away Team Throw Ins           12062 non-null  float64
 29  Away Team Pass Success %      12062 non-null  float64
 30  Away Team Aerials Won         12062 non-null  float64
 31  Away Team Clearances          12062 non-null  float64
 32  Away Team Fouls               12062 non-null  float64
 33  Away Team Yellow Cards        12062 non-null  float64
 34  Away Team Second Yellow Cards 12062 non-null  float64
 35  Away Team Red Cards           12062 non-null  float64
 36  Home Team Goals Scored        12062 non-null  int64
 37  Away Team Goals Scored        12062 non-null  int64
 38  Home Team Goals Conceeded     12062 non-null  int64
 39  Away Team Goals Conceeded     12062 non-null  int64
 40  year                          12062 non-null  int64
 41  league                        12062 non-null  object
dtypes: float64(29), int64(8), object(5)
memory usage: 3.9+ MB
```

All of the variables in the dataset and their types can be seen above. The teams, final score, half time score, and league are categorical variables and the only non numerical data. The rest of the columns are numerical data points.

Total Size of Data Set

In [6]:
```
print(df.shape)
```

```
(12062, 42)
```

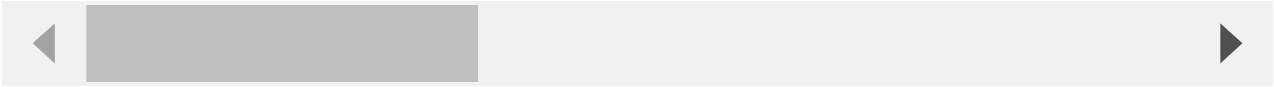
There are 42 columns and 12062 rows in the dataset which leads to a total of 506,604 data points because there is no missing data.

In [7]:
```
df.describe()
```

Out[7]:

	Unnamed: 0	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Away Team Possession %	Home Tear Off Targe Shot
count	12062.000000	12062.000000	12062.000000	12062.000000	12062.000000	12062.000000	12062.00000
mean	6030.500000	5.391494	6.399702	6.142539	51.199469	48.800531	5.64433
std	3482.143808	1.719655	0.988037	0.992074	11.275257	11.275257	2.78152
min	0.000000	1.400000	2.100000	1.300000	17.000000	16.000000	0.00000
25%	3015.250000	4.100000	5.700000	5.500000	43.000000	41.000000	4.00000
50%	6030.500000	5.300000	6.400000	6.100000	51.000000	49.000000	5.00000
75%	9045.750000	6.400000	7.000000	6.800000	59.000000	57.000000	7.00000
max	12061.000000	10.000000	10.000000	10.000000	84.000000	83.000000	20.00000

8 rows × 37 columns



In [8]:

```
df.describe(include=object)
```

Out[8]:

	Home Team	Away Team	Score	Half Time Score	league
count	12062	12062	12062	12062	12062
unique	146	146	91	38	5
top	EIBAR	EIBAR	1-0	0-0	spanish
freq	133	133	1129	3637	2660

The above two tables give an idea of the totals, unique values, and other details of the dataset.

TimeFrame

In [9]:

```
print('Data set begins at year: ' + str(df['year'].min()))
print('Data set ends at year: ' + str(df['year'].max()))
```

```
Data set begins at year: 2014
Data set ends at year: 2020
```

The year column is used to determine the time frame of the dataset. The first entries are in 2014 and the last entries are in 2020, showing a span of 7 years.

Missing Data

Looking at the info output, it can be confirmed that each column has 12062 data points and confirms that there is no missing data.

In [10]:

```
df.isnull().sum()
```

Out[10]:
```
Unnamed: 0                      0
Home Team                       0
Away Team                       0
Score                           0
Half Time Score                 0
Match Excitement                0
Home Team Rating                0
Away Team Rating                0
Home Team Possession %          0
Away Team Possession %          0
Home Team Off Target Shots      0
Home Team On Target Shots       0
Home Team Total Shots           0
Home Team Blocked Shots         0
Home Team Corners               0
Home Team Throw Ins             0
Home Team Pass Success %        0
Home Team Aerials Won           0
Home Team Clearances            0
Home Team Fouls                 0
Home Team Yellow Cards          0
Home Team Second Yellow Cards   0
Home Team Red Cards             0
Away Team Off Target Shots      0
Away Team On Target Shots       0
Away Team Total Shots           0
Away Team Blocked Shots         0
Away Team Corners               0
Away Team Throw Ins             0
Away Team Pass Success %        0
Away Team Aerials Won           0
Away Team Clearances            0
Away Team Fouls                 0
Away Team Yellow Cards          0
Away Team Second Yellow Cards   0
Away Team Red Cards             0
Home Team Goals Scored          0
Away Team Goals Scored          0
Home Team Goals Conceeded       0
Away Team Goals Conceeded       0
year                            0
league                          0
dtype: int64
```

Player Data Set Inspection

Type of each Variable

In [11]:
```
df_player.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 85 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ID                        18207 non-null  int64
 1   Name                      18207 non-null  object
 2   Age                       18207 non-null  int64
 3   Nationality               18207 non-null  object
 4   Overall                   18207 non-null  int64
 5   Potential                 18207 non-null  int64
 6   Club                      17966 non-null  object
 7   Value                     18207 non-null  object
```

```
8   Wage                      18207 non-null  object
9   Special                   18207 non-null  int64
10  Preferred Foot            18159 non-null  object
11  International Reputation  18159 non-null  float64
12  Weak Foot                 18159 non-null  float64
13  Skill Moves               18159 non-null  float64
14  Work Rate                 18159 non-null  object
15  Body Type                 18159 non-null  object
16  Real Face                 18159 non-null  object
17  Position                  18147 non-null  object
18  Jersey Number             18147 non-null  float64
19  Joined                    16654 non-null  object
20  Loaned From               1264 non-null   object
21  Contract Valid Until      17918 non-null  object
22  Height                    18159 non-null  object
23  Weight                    18159 non-null  object
24  LS                        16122 non-null  object
25  ST                        16122 non-null  object
26  RS                        16122 non-null  object
27  LW                        16122 non-null  object
28  LF                        16122 non-null  object
29  CF                        16122 non-null  object
30  RF                        16122 non-null  object
31  RW                        16122 non-null  object
32  LAM                       16122 non-null  object
33  CAM                       16122 non-null  object
34  RAM                       16122 non-null  object
35  LM                        16122 non-null  object
36  LCM                       16122 non-null  object
37  CM                        16122 non-null  object
38  RCM                       16122 non-null  object
39  RM                        16122 non-null  object
40  LWB                       16122 non-null  object
41  LDM                       16122 non-null  object
42  CDM                       16122 non-null  object
43  RDM                       16122 non-null  object
44  RWB                       16122 non-null  object
45  LB                        16122 non-null  object
46  LCB                       16122 non-null  object
47  CB                        16122 non-null  object
48  RCB                       16122 non-null  object
49  RB                        16122 non-null  object
50  Crossing                  18159 non-null  float64
51  Finishing                 18159 non-null  float64
52  HeadingAccuracy           18159 non-null  float64
53  ShortPassing              18159 non-null  float64
54  Volleys                   18159 non-null  float64
55  Dribbling                 18159 non-null  float64
56  Curve                     18159 non-null  float64
57  FKAccuracy                18159 non-null  float64
58  LongPassing               18159 non-null  float64
59  BallControl               18159 non-null  float64
60  Acceleration              18159 non-null  float64
61  SprintSpeed               18159 non-null  float64
62  Agility                   18159 non-null  float64
63  Reactions                 18159 non-null  float64
64  Balance                   18159 non-null  float64
65  ShotPower                 18159 non-null  float64
66  Jumping                   18159 non-null  float64
67  Stamina                   18159 non-null  float64
68  Strength                  18159 non-null  float64
69  LongShots                 18159 non-null  float64
70  Aggression                18159 non-null  float64
71  Interceptions             18159 non-null  float64
72  Positioning               18159 non-null  float64
```

```
 73  Vision                  18159 non-null  float64
 74  Penalties               18159 non-null  float64
 75  Composure               18159 non-null  float64
 76  Marking                 18159 non-null  float64
 77  StandingTackle          18159 non-null  float64
 78  SlidingTackle           18159 non-null  float64
 79  GKDiving                18159 non-null  float64
 80  GKHandling              18159 non-null  float64
 81  GKKicking               18159 non-null  float64
 82  GKPositioning           18159 non-null  float64
 83  GKReflexes              18159 non-null  float64
 84  Release Clause          16643 non-null  object
dtypes: float64(38), int64(5), object(42)
memory usage: 11.8+ MB
```

This data set has a good mix of categorical and numerical data. Many of the positions and naming variables are categorial while the stats and skill variables are numerical.

Total Size of Data Set

In [12]:
```
df_player.shape
```

Out[12]: (18207, 85)

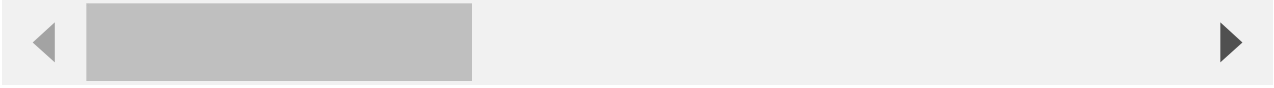
The data set has 85 columns and 18207 rows.

In [13]:
```
df_player.describe()
```

Out[13]:

	ID	Age	Overall	Potential	Special	International Reputation	Weak Fo
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18159.000000	18159.0000
mean	214298.338606	25.122206	66.238699	71.307299	1597.809908	1.113222	2.9472
std	29965.244204	4.669943	6.908930	6.136496	272.586016	0.394031	0.6604
min	16.000000	16.000000	46.000000	48.000000	731.000000	1.000000	1.0000
25%	200315.500000	21.000000	62.000000	67.000000	1457.000000	1.000000	3.0000
50%	221759.000000	25.000000	66.000000	71.000000	1635.000000	1.000000	3.0000
75%	236529.500000	28.000000	71.000000	75.000000	1787.000000	1.000000	3.0000
max	246620.000000	45.000000	94.000000	95.000000	2346.000000	5.000000	5.0000

8 rows × 43 columns



In [14]:
```
df_player.describe(include=object)
```

Out[14]:

	Name	Nationality	Club	Value	Wage	Preferred Foot	Work Rate	Body Type	Real Face	Position
count	18207	18207	17966	18207	18207	18159	18159	18159	18159	18147

	Name	Nationality	Club	Value	Wage	Preferred Foot	Work Rate	Body Type	Real Face	Position
unique	17194	164	651	217	144	2	9	10	2	27
top	J. Rodríguez	England	Frosinone	€1.1M	€1K	Right	Medium/ Medium	Normal	No	ST
freq	11	1662	33	431	4900	13948	9810	10595	16505	2152

4 rows × 42 columns

The above two tables give an idea of the totals, unique values, and other details of the dataset.

TimeFrame

Entire data set was from 2019, single year of data.

Missing Data

In [15]:
```
print(df_player.isnull().sum())
```

```
ID                   0
Name                 0
Age                  0
Nationality          0
Overall              0
                   ...
GKHandling          48
GKKicking           48
GKPositioning       48
GKReflexes          48
Release Clause    1564
Length: 85, dtype: int64
```

In [16]:
```
df_player.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 85 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ID                        18207 non-null  int64
 1   Name                      18207 non-null  object
 2   Age                       18207 non-null  int64
 3   Nationality               18207 non-null  object
 4   Overall                   18207 non-null  int64
 5   Potential                 18207 non-null  int64
 6   Club                      17966 non-null  object
 7   Value                     18207 non-null  object
 8   Wage                      18207 non-null  object
 9   Special                   18207 non-null  int64
 10  Preferred Foot            18159 non-null  object
 11  International Reputation  18159 non-null  float64
 12  Weak Foot                 18159 non-null  float64
 13  Skill Moves               18159 non-null  float64
 14  Work Rate                 18159 non-null  object
 15  Body Type                 18159 non-null  object
```

```
16  Real Face                 18159 non-null  object
17  Position                  18147 non-null  object
18  Jersey Number             18147 non-null  float64
19  Joined                    16654 non-null  object
20  Loaned From               1264 non-null   object
21  Contract Valid Until      17918 non-null  object
22  Height                    18159 non-null  object
23  Weight                    18159 non-null  object
24  LS                        16122 non-null  object
25  ST                        16122 non-null  object
26  RS                        16122 non-null  object
27  LW                        16122 non-null  object
28  LF                        16122 non-null  object
29  CF                        16122 non-null  object
30  RF                        16122 non-null  object
31  RW                        16122 non-null  object
32  LAM                       16122 non-null  object
33  CAM                       16122 non-null  object
34  RAM                       16122 non-null  object
35  LM                        16122 non-null  object
36  LCM                       16122 non-null  object
37  CM                        16122 non-null  object
38  RCM                       16122 non-null  object
39  RM                        16122 non-null  object
40  LWB                       16122 non-null  object
41  LDM                       16122 non-null  object
42  CDM                       16122 non-null  object
43  RDM                       16122 non-null  object
44  RWB                       16122 non-null  object
45  LB                        16122 non-null  object
46  LCB                       16122 non-null  object
47  CB                        16122 non-null  object
48  RCB                       16122 non-null  object
49  RB                        16122 non-null  object
50  Crossing                  18159 non-null  float64
51  Finishing                 18159 non-null  float64
52  HeadingAccuracy           18159 non-null  float64
53  ShortPassing              18159 non-null  float64
54  Volleys                   18159 non-null  float64
55  Dribbling                 18159 non-null  float64
56  Curve                     18159 non-null  float64
57  FKAccuracy                18159 non-null  float64
58  LongPassing               18159 non-null  float64
59  BallControl               18159 non-null  float64
60  Acceleration              18159 non-null  float64
61  SprintSpeed               18159 non-null  float64
62  Agility                   18159 non-null  float64
63  Reactions                 18159 non-null  float64
64  Balance                   18159 non-null  float64
65  ShotPower                 18159 non-null  float64
66  Jumping                   18159 non-null  float64
67  Stamina                   18159 non-null  float64
68  Strength                  18159 non-null  float64
69  LongShots                 18159 non-null  float64
70  Aggression                18159 non-null  float64
71  Interceptions             18159 non-null  float64
72  Positioning               18159 non-null  float64
73  Vision                    18159 non-null  float64
74  Penalties                 18159 non-null  float64
75  Composure                 18159 non-null  float64
76  Marking                   18159 non-null  float64
77  StandingTackle            18159 non-null  float64
78  SlidingTackle             18159 non-null  float64
79  GKDiving                  18159 non-null  float64
80  GKHandling                18159 non-null  float64
```

```
 81  GKKicking                 18159 non-null  float64
 82  GKPositioning             18159 non-null  float64
 83  GKReflexes                18159 non-null  float64
 84  Release Clause            16643 non-null  object
dtypes: float64(38), int64(5), object(42)
memory usage: 11.8+ MB
```

Using the .info() table, looking at the count of each column there are many with missing data. Many of the positions are missing over 2000 data points. Lots of the other columns have about 50 missing data points.

International Data Set Inspection

Type of each Variable

In [17]:
```
df_international.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43188 entries, 0 to 43187
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date        43188 non-null  object
 1   home_team   43188 non-null  object
 2   away_team   43188 non-null  object
 3   home_score  43188 non-null  int64 
 4   away_score  43188 non-null  int64 
 5   tournament  43188 non-null  object
 6   city        43188 non-null  object
 7   country     43188 non-null  object
 8   neutral     43188 non-null  bool  
dtypes: bool(1), int64(2), object(6)
memory usage: 2.7+ MB
```

The data set has two numerical variables, the home and away scores of a game, with the rest of the variables being categorical.

Total Size of Data Set

In [18]:
```
df_international.shape
```

Out[18]:
```
(43188, 9)
```

The data set has 9 columns and 43188 rows. There is no missing data so there is a total of 388,692 data points.

In [19]:
```
df_international.describe()
```

Out[19]:

	home_score	away_score
count	43188.000000	43188.000000
mean	1.742544	1.182666
std	1.753055	1.399664
min	0.000000	0.000000

	home_score	away_score
25%	1.000000	0.000000
50%	1.000000	1.000000
75%	2.000000	2.000000
max	31.000000	21.000000

In [20]:
```
df_international.describe(include=object)
```

Out[20]:

	date	home_team	away_team	tournament	city	country
count	43188	43188	43188	43188	43188	43188
unique	15421	306	303	126	1976	267
top	2012-02-29	Brazil	Uruguay	Friendly	Kuala Lumpur	United States
freq	66	586	551	17276	667	1237

The above two tables give an idea of the totals, unique values, and other details of the dataset.

TimeFrame

In [21]:
```
print('Data set begins at year: ' + str(df_international['date'].min()))
print('Data set ends at year: ' + str(df_international['date'].max()))
```

```
Data set begins at year: 1872-11-30
Data set ends at year: 2022-02-06
```

This data set spans from November 30, 1872 to February 2, 2022, a span of 150 years.

Missing Data

Looking at the info output, it can be confirmed that each column has 43188 data points and confirms that there is no missing data.

In [22]:
```
print(df_international.isnull().sum())
```

```
date          0
home_team     0
away_team     0
home_score    0
away_score    0
tournament    0
city          0
country       0
neutral       0
dtype: int64
```

Part 3 - Data Description

- Create a data description (data dictionary) for your data sets.
  - Describe each variable (4,5,6 word explanation)

- If categorical, what levels are present? If the levels are encoded, what do the codes mean? (ex: 0 = male, 1 = female)
- If numeric, provide min, max, median and any other univariate stats you'd like to add in.

- Where appropriate, provide histograms or other visualizations to characterize each variable.

Football Data Set Dictionary

Home Team

-name of the home team in the match
-nominal categorical variable, no encoding

Away Team

-name of the away team in the match
-nominal categorical variable, no encoding

Score

-score of the match
-nominal categorical variable, no encoding

Half Time Score

-half time score of the match
-nominal categorical variable, no encoding

Match Excitement

-a score of how exciting the match was on 0 to 10 scale

In [23]:
```
print("Min:",df['Match Excitement'].min())
print("Max:",df['Match Excitement'].max())
print("Median:",df['Match Excitement'].median())
```

```
Min: 1.4
Max: 10.0
Median: 5.3
```

Home Team Rating

-a score of how well the home team performed on 0 to 10 scale

In [24]:
```
print("Min:",df['Home Team Rating'].min())
print("Max:",df['Home Team Rating'].max())
print("Median:",df['Home Team Rating'].median())
```

```
Min: 2.1
Max: 10.0
Median: 6.4
```

Away Team Rating

-a score of well the away team performed on a 0 to 10 scale

In [25]:
```
print("Min:",df['Away Team Rating'].min())
print("Max:",df['Away Team Rating'].max())
print("Median:",df['Away Team Rating'].median())
```

```
Min: 1.3
Max: 10.0
Median: 6.1
```

Home Team Possession %

-% of how much home team had the ball

In [26]:
```
print("Min:",df['Home Team Possession %'].min())
print("Max:",df['Home Team Possession %'].max())
print("Median:",df['Home Team Possession %'].median())
```

```
Min: 17
Max: 84
Median: 51.0
```

Away Team Possession %

-% of how much away team had the ball

In [27]:
```
print("Min:",df['Away Team Possession %'].min())
print("Max:",df['Away Team Possession %'].max())
print("Median:",df['Away Team Possession %'].median())
```

```
Min: 16
Max: 83
Median: 49.0
```

Home Team Off Target Shots

-how many shots were off target by home team

In [28]:
```
print("Min:",df['Home Team Off Target Shots'].min())
print("Max:",df['Home Team Off Target Shots'].max())
print("Median:",df['Home Team Off Target Shots'].median())
```

```
Min: 0.0
Max: 20.0
Median: 5.0
```

Home Team On Target Shots

-how many shots were on target by home team

In [29]:
```
print("Min:",df['Home Team On Target Shots'].min())
print("Max:",df['Home Team On Target Shots'].max())
print("Median:",df['Home Team On Target Shots'].median())
```

```
Min: 0.0
```

```
Max: 18.0
Median: 4.0
```

Home Team Total Shots

-total number of shots by home team

In [30]:
```
print("Min:",df['Home Team Total Shots'].min())
print("Max:",df['Home Team Total Shots'].max())
print("Median:",df['Home Team Total Shots'].median())
```

```
Min: 0.0
Max: 47.0
Median: 13.0
```

Away Team Off Target Shots

-how many shots were off target by away team

In [31]:
```
print("Min:",df['Away Team Off Target Shots'].min())
print("Max:",df['Away Team Off Target Shots'].max())
print("Median:",df['Away Team Off Target Shots'].median())
```

```
Min: 0.0
Max: 17.0
Median: 4.0
```

Away Team On Target Shots

-how many shots were on target by away team

In [32]:
```
print("Min:",df['Away Team On Target Shots'].min())
print("Max:",df['Away Team On Target Shots'].max())
print("Median:",df['Away Team On Target Shots'].median())
```

```
Min: 0.0
Max: 15.0
Median: 4.0
```

Away Team Total Shots

-total number of shots by away team

In [33]:
```
print("Min:",df['Away Team Total Shots'].min())
print("Max:",df['Away Team Total Shots'].max())
print("Median:",df['Away Team Total Shots'].median())
```

```
Min: 0.0
Max: 39.0
Median: 11.0
```

Home Team Goals Scored

-goals scored by home team

In [34]:
```
print("Min:",df['Home Team Goals Scored'].min())
```

```
print("Max:",df['Home Team Goals Scored'].max())
print("Median:",df['Home Team Goals Scored'].median())
```

```
Min: 0
Max: 10
Median: 1.0
```

Away Team Goals Scored

-goals scored by away team

In [35]:
```
print("Min:",df['Away Team Goals Scored'].min())
print("Max:",df['Away Team Goals Scored'].max())
print("Median:",df['Away Team Goals Scored'].median())
```

```
Min: 0
Max: 9
Median: 1.0
```

Home Team Goals Conceeded

-goals conceeded by home team

In [36]:
```
print("Min:",df['Home Team Goals Conceeded'].min())
print("Max:",df['Home Team Goals Conceeded'].max())
print("Median:",df['Home Team Goals Conceeded'].median())
```

```
Min: 0
Max: 9
Median: 1.0
```

Year

-year of match

In [37]:
```
print("Min:",df['year'].min())
print("Max:",df['year'].max())
print("Median:",df['year'].median())
```

```
Min: 2014
Max: 2020
Median: 2017.0
```

League

-name of the league match takes place in

-nominal categorical variable, no encoding

Player Data Set Dictionary

ID

-player ID

In [38]:
```
print("Min:",df_player['ID'].min())
```

```
print("Max:",df_player['ID'].max())
print("Median:",df_player['ID'].median())
```

```
Min: 16
Max: 246620
Median: 221759.0
```

Name

-name of the player

-nominal categorical variable, no encoding

Age

-age of player

In [39]:

```
print("Min:",df_player['Age'].min())
print("Max:",df_player['Age'].max())
print("Median:",df_player['Age'].median())
```

```
Min: 16
Max: 45
Median: 25.0
```

Nationality

-nationality of the player

-nominal categorical variable, no encoding

Overall

-skill rating of the player

In [40]:

```
print("Min:",df_player['Overall'].min())
print("Max:",df_player['Overall'].max())
print("Median:",df_player['Overall'].median())
```

```
Min: 46
Max: 94
Median: 66.0
```

Potential

-potential rating of the player

In [41]:

```
print("Min:",df_player['Potential'].min())
print("Max:",df_player['Potential'].max())
print("Median:",df_player['Potential'].median())
```

```
Min: 48
Max: 95
Median: 71.0
```

Club

-the club the player is on
-nominal categorical variable, no encoding

Value

-the money value of a player
-nominal categorical variable, no encoding

Wage

-salary of the player
-nominal categorical variable, no encoding

Preferred Foot

-which foot the player uses more
-nominal categorical variable, no encoding

International Reputation

-how well known the player is

In [42]:
```
print("Min:",df_player['International Reputation'].min())
print("Max:",df_player['International Reputation'].max())
print("Median:",df_player['International Reputation'].median())
```

```
Min: 1.0
Max: 5.0
Median: 1.0
```

Weak Foot

-skill of player's weak foot

In [43]:
```
print("Min:",df_player['Weak Foot'].min())
print("Max:",df_player['Weak Foot'].max())
print("Median:",df_player['Weak Foot'].median())
```

```
Min: 1.0
Max: 5.0
Median: 3.0
```

Skill Moves

-rating of a player's skill moves

In [44]:
```
print("Min:",df_player['Skill Moves'].min())
print("Max:",df_player['Skill Moves'].max())
print("Median:",df_player['Skill Moves'].median())
```

```
Min: 1.0
Max: 5.0
Median: 2.0
```

Work Rate

-how hard the player works

-nominal categorical variable, no encoding

Body Type

-what type of body the player has

-nominal categorical variable, no encoding

Real Face

-if the player's face scan is contained in the fifa database

-nominal categorical variable, dichotomous variable, no encoding, yes or no

Jersey Number

-jersey number of player

In [45]:
```
print("Min:",df_player['Jersey Number'].min())
print("Max:",df_player['Jersey Number'].max())
print("Median:",df_player['Jersey Number'].median())
```

```
Min: 1.0
Max: 99.0
Median: 17.0
```

Joined

-when the player joined his club

-nominal categorical variable, no encoding

International Data Set Dictionary

In [46]:
```
df_international.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43188 entries, 0 to 43187
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date        43188 non-null  object
 1   home_team   43188 non-null  object
 2   away_team   43188 non-null  object
 3   home_score  43188 non-null  int64 
 4   away_score  43188 non-null  int64 
 5   tournament  43188 non-null  object
 6   city        43188 non-null  object
 7   country     43188 non-null  object
 8   neutral     43188 non-null  bool  
dtypes: bool(1), int64(2), object(6)
memory usage: 2.7+ MB
```

date

-when the game took place
-nominal categorical variable, no encoding

home_team

-name of country playing labeled as home team
-nominal categorical variable, no encoding

away_team

-name of country playing labeled as away team
-nominal categorical variable, no encoding

home_score

-score of the home team

In [47]:
```
print("Min:",df_international['home_score'].min())
print("Max:",df_international['home_score'].max())
print("Median:",df_international['home_score'].median())
```

```
Min: 0
Max: 31
Median: 1.0
```

away_score

-score of the away team

In [48]:
```
print("Min:",df_international['away_score'].min())
print("Max:",df_international['away_score'].max())
print("Median:",df_international['away_score'].median())
```

```
Min: 0
Max: 21
Median: 1.0
```

tournament

-what tournament was being played
-nominal categorical variable, no encoding

city

-what city tournmanet was in
-nominal categorical variable, no encoding

country

-what country tournmanet was in
-nominal categorical variable, no encoding

neutral

-if match was a neutral site or not
-nominal categorical variable, no encoding, true or false

- No encoding was present in the datasets. Any one hot encoding necessary will be done when merging the data (see merge#2). Upon merging a dichotomous "victor" variable will be created.

Part 4 - Merge the data

Now that you have a better feel for each of your two (or three, for the 7394 students) data sets, it is time to merge them. Describe your strategy for merging the data sets and then actually perform the merge.

Develop a strategy for verifying that the data is properly merged (hoping and finger-crossing are not valid strategies).

Strategy for Merging Dataset #1:
The complete playerbase dataset has all the top players, their rating, their salary, and the club they play for. Using the playerbase dataset, each club's average player rating and average salary cap will be calculated. Using the calculated values from the player database, the club's respective ratinbg values will be merged into the dataframe "on" the "Home Team" feature creating home rating/salary features. The merge then will be performed again on the "Away Team" creating the away rating/salary features. Since the club names are different between datasets, we will manually need to create a dictionary to translate between the two club name styles.

In [49]:
```
df_temp = df[df["year"] == 2019]
df_matches = df_temp[df_temp["league"] == 'spanish']
df_matches = df_matches.drop(columns='Unnamed: 0').copy()
df_matches #currently contains all the matches from 2019-2021 spanish league
```

Out[49]:

	Home Team	Away Team	Score	Half Time Score	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Away Team Possession %	Hor Tea (Targ Sh
11302	ATHLETIC	BARCELONA	1-0	0-0	4.7	7.0	5.7	28	72	(
11303	CELTA	REAL MADRID	1-3	0-1	7.5	4.9	7.3	56	44	7
11304	VALENCIA	REAL SOCIEDAD	1-1	0-0	5.6	5.6	6.0	37	63	4
11305	MALLORCA	EIBAR	2-1	1-0	5.8	6.2	5.4	36	64	1(
11306	LEGANÉS	OSASUNA	0-1	0-0	3.3	5.6	6.3	51	49	(
...	...	...	...	...	...	...	...	...	...	
11677	OSASUNA	MALLORCA	2-2	1-1	6.9	6.4	6.5	44	56	(
11678	SEVILLA FC	VALENCIA	1-0	0-0	3.7	6.8	5.7	62	38	1(
11679	ESPANYOL	CELTA	0-0	0-0	2.0	6.0	6.7	50	50	7

	Home Team	Away Team	Score	Half Time Score	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Away Team Possession %	Hor Tea (Targ Sho
11680	GRANADA	ATHLETIC	4-0	1-0	5.5	7.7	4.6	36	64	
11681	LEGANÉS	REAL MADRID	2-2	1-1	7.5	6.4	6.5	35	65	

380 rows × 41 columns

Method to Validate the Merged Dataset#1:

Seperately calculate the average salary and player rating for the team 'FC Barcelona.' Verify the merged dataframe values for FC Barcelona equals the control value calculated here. Also we will utilize counts to make sure the number of rows and number of features are accurate.

In [50]:
```
df_Barcelona = df_player[df_player["Club"] == 'FC Barcelona']
df_Barcelona = df_Barcelona.head(14)
# calculate the average rating of starting 11 + subs
avgRating = df_Barcelona['Overall'].sum() / 14.0
print('expected average rating: ' + str(avgRating))
# calculate the average team salary
salaryList = df_Barcelona['Value'] # get list of salaries
doubleList = [float(x[1:-1]) for x in salaryList] #convert salary to a double
avgSalary = sum(doubleList) / len(doubleList)
print('expected average salary: ' + str(avgSalary))
```

```
expected average rating: 86.71428571428571
expected average salary: 50.107142857142854
```

Calculate Average Rating and Salary:

Creating a lookup datastructure will help us to resolve the difference in 'club names' between the two datasets. The club information function will return the average salary and average player rating of the top 14 players of each club. The top 14 players were selected because that is how many are allowed total in an offical football match (starting 11 players + 3 subsitutions).

In [51]:
```
lookup = {'ATHLETIC':'Athletic Club de Bilbao', 'CELTA':'RC Celta', 'VALENCIA':'Valenci
       'VILLARREAL':'Villarreal CF', 'ALAVÉS':'Deportivo Alavés', 'ESPANYOL':'RCD Espan
       'ATLETICO MADRID':'Atlético Madrid', 'GRANADA':'Granada CF', 'LEVANTE':'Levante
       'GETAFE':'Getafe CF', 'BARCELONA':'FC Barcelona', 'SEVILLA FC':'Sevilla FC', 'RE
       'VALLADOLID':'Real Valladolid CF'}

# get information on a professional football club that has a translation in the lookup
def getClubInformation(target):
    df_target = df_player[df_player["Club"] == lookup[target]]
    df_target = df_target.head(14)
    # calculate the average rating of starting 11 + subs
    avgRating = df_target['Overall'].sum() / 14.0
    # calculate the average team salary
    salaryList = df_target['Value'] # get list of salaries
```

```
    doubleList = [float(x[1:-1]) if x[-1] != 'K' else float(x[1:-1]) / 100 for x in sal
    avgSalary = sum(doubleList) / len(doubleList)
    return avgRating, avgSalary
```

Prepare Club Information for Merging :

In [52]:

```
teams = df_matches['Home Team'].unique()
teamsList = []
ratingsList = []
salaryList = []
for team in teams:
    averageRating, averageSalary = getClubInformation(team)
    teamsList.append(team)
    ratingsList.append(averageRating)
    salaryList.append(averageSalary)

df_SpanishClubs = pd.DataFrame({'Team': teamsList,
                   'Team Rating': ratingsList,
                   'Team Salary (millions)': salaryList})

df_SpanishClubs.head()
```

Out[52]:

	Team	Team Rating	Team Salary (millions)
0	ATHLETIC	79.214286	12.678571
1	CELTA	77.571429	11.850000
2	VALENCIA	81.428571	21.750000
3	MALLORCA	67.714286	4.207143
4	LEGANÉS	75.357143	6.978571

Perform Merge#1:

In [53]:

```
df_SpanishClubs.columns = ['Home Team','Home Rating','Home Salary']
df_matches = df_matches.merge(df_SpanishClubs, how='inner', on='Home Team')
df_SpanishClubs.columns = ['Away Team','Away Rating','Away Salary']
df_matches = df_matches.merge(df_SpanishClubs, how='inner', on='Away Team')
df_matches.head()
```

Out[53]:

	Home Team	Away Team	Score	Half Time Score	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Away Team Possession %	Home Team Off Target Shots
0	ATHLETIC	BARCELONA	1-0	0-0	4.7	7.0	5.7	28	72	6.0
1	CELTA	BARCELONA	2-2	0-1	8.4	7.0	5.9	32	68	1.0
2	VALENCIA	BARCELONA	2-0	0-0	5.3	7.1	5.1	26	74	2.0
3	MALLORCA	BARCELONA	0-4	0-2	6.2	5.0	8.4	35	65	7.0
4	LEGANÉS	BARCELONA	1-2	1-0	5.6	5.5	6.9	23	77	5.0

5 rows × 45 columns

Validating Merge#1: As seen above Barcelona is the away team at index at 0. We will compare the merged values with our control values calculated above as described in our strategy.

In [54]:
```
print('Merged Away Rating of Barcelona: ' + str(df_matches.iloc[0]['Away Rating']))
print('Merged Away Salary of Barcelona: ' + str(df_matches.iloc[0]['Away Salary']))
print('Column Count: ' + str(len(df_matches.columns)))
print('---------------------')
print(df.isnull().sum())
```

```
Merged Away Rating of Barcelona: 86.71428571428571
Merged Away Salary of Barcelona: 50.107142857142854
Column Count: 45
---------------------
Unnamed: 0                      0
Home Team                       0
Away Team                       0
Score                           0
Half Time Score                 0
Match Excitement                0
Home Team Rating                0
Away Team Rating                0
Home Team Possession %          0
Away Team Possession %          0
Home Team Off Target Shots      0
Home Team On Target Shots       0
Home Team Total Shots           0
Home Team Blocked Shots         0
Home Team Corners               0
Home Team Throw Ins             0
Home Team Pass Success %        0
Home Team Aerials Won           0
Home Team Clearances            0
Home Team Fouls                 0
Home Team Yellow Cards          0
Home Team Second Yellow Cards   0
Home Team Red Cards             0
Away Team Off Target Shots      0
Away Team On Target Shots       0
Away Team Total Shots           0
Away Team Blocked Shots         0
Away Team Corners               0
Away Team Throw Ins             0
Away Team Pass Success %        0
Away Team Aerials Won           0
Away Team Clearances            0
Away Team Fouls                 0
Away Team Yellow Cards          0
Away Team Second Yellow Cards   0
Away Team Red Cards             0
Home Team Goals Scored          0
Away Team Goals Scored          0
Home Team Goals Conceeded       0
Away Team Goals Conceeded       0
year                            0
league                          0
dtype: int64
```

The data was merged successfully. The Away values for Barcelona rating/salary are correct in the dataframe when it is correspondingly the home or away team. The calculations match the control value. The number of columns also increases by 4 as predicted (home rating, home salary, away rating, away salary). Also there are no NaN values so the lookup datastructure/merge was successful.

Strategy for Merging & Validating Dataset #2:
After data cleaning, the International Dataset will be appended to the end of the parent dataframe to add more matches using df.append(). Some columns will be NaN because data such as ball possesion will be unvailable. Using the dataset#1 the player rating and player salary can be calculated for the international teams.

To validate the merge we will look at the number of rows after the merge before dropping the NaN values, validating they add up properly. After dropping the NaN values, we will observe the NaN count in the player rating/salary column to ensure the player dataset was merged in properly. At that point there should be no NaN values in the dataframe and the number of rows/matches in the dataframe will be verified.

Preparing International Dataset
First filter by year and get the matches from 2019. Predicting based on 2019 will provide the most accurate results because that is what our player salary/rating data is predicated upon. We will also need to clean the dataframe and make the columns match that of the parent dataframe.

In [55]:
```
# CLEANING THE international dataframe (df_national_2019 is our target dataframe)
# make the 'year column'
yearList = [x[0:4] for x in df_international['date']]
df_international['year'] = yearList
#get only matches in 2019
df_national_2019_slice = df_international[df_international["year"] >= '2011'] #CHANGED
df_national_2019 = df_national_2019_slice.copy() # create a copy of the slice, slice sh
# create league column:
leagueList = ['international' for x in range (0, len(df_national_2019.index))]
df_national_2019['league'] = leagueList
# create score columns:
home_score = df_national_2019['home_score'].values
away_score = df_national_2019['away_score'].values
scoreList = [str(home_score[i]) + '-' + str(away_score[i]) for i in range(0,len(home_sc
df_national_2019['Score'] = scoreList
df_national_2019['Home Team Goals Conceeded'] = away_score
df_national_2019['Away Team Goals Conceeded'] = home_score

#drop undersired columns
df_national_2019.drop(columns=['date', 'country', 'tournament', 'city', 'neutral'], inp
#rename the columns
df_national_2019.columns = ['Home Team','Away Team', 'Home Team Goals Scored', 'Away Te
df_national_2019
```

Out[55]:

	Home Team	Away Team	Home Team Goals Scored	Away Team Goals Scored	year	league	Score	Home Team Goals Conceeded	Away Team Goals Conceeded

	Home Team	Away Team	Home Team Goals Scored	Away Team Goals Scored	year	league	Score	Home Team Goals Conceeded	Away Team Goals Conceeded
32851	Iran	Angola	1	0	2011	international	1-0	0	1
32852	Iraq	China PR	2	3	2011	international	2-3	3	2
32853	Jordan	Uzbekistan	2	2	2011	international	2-2	2	2
32854	United Arab Emirates	Syria	2	0	2011	international	2-0	0	2
32855	Bahrain	North Korea	0	1	2011	international	0-1	1	0
...	...	...	...	...	...	...	...	...	...
43183	Suriname	Guyana	2	1	2022	international	2-1	1	2
43184	Burkina Faso	Senegal	1	3	2022	international	1-3	3	1
43185	Cameroon	Egypt	0	0	2022	international	0-0	0	0
43186	Cameroon	Burkina Faso	3	3	2022	international	3-3	3	3
43187	Senegal	Egypt	0	0	2022	international	0-0	0	0

10337 rows × 9 columns

In [56]:

```
# Calculate the average player ratiang and salary
def getCountryInformation(country):
    # refactored function to work with internatonal matches to get average rating and s
    lookup2 = {'South Korea':'Korea Republic'}
    df_target = None
    if country in lookup2:
        df_target = df_player[df_player["Nationality"] == lookup2[country]]
    else:
        df_target = df_player[df_player["Nationality"] == country]
    n = len(df_target)
    if n >= 14:
        n = 14
    elif n == 0:
        return None, 0
    df_target = df_target.head(n)
    # calculate the average rating of starting 11 + subs
    avgRating = df_target['Overall'].sum() / n
    # calculate the average team salary
    salaryList = list(df_target['Value']) # get list of salaries
    if '€0' in salaryList:
        return None, 0
    doubleList = [float(x[1:-1]) if x[-1] != 'K' else float(x[1:-1]) / 100 for x in sal
    avgSalary = sum(doubleList) / len(doubleList)
    #avgSalary = 0
    return avgRating, avgSalary
```

```
# call the function to get international data
#teams = df_national_2019['Home Team'].unique() + list(set(df_national_2019['Away Team'
list1 = df_national_2019['Home Team'].unique()
list2 = df_national_2019['Away Team'].unique()
teams = list(set(list1) | set(list2))
teamsList = []
ratingsList = []
salaryList = []
for team in teams:
    averageRating, averageSalary = getCountryInformation(team)
    teamsList.append(team)
    ratingsList.append(averageRating)
    salaryList.append(averageSalary)

df_intRatings = pd.DataFrame({'Team': teamsList,
                   'Team Rating': ratingsList,
                   'Team Salary (millions)': salaryList})

df_intRatings
```

Out[56]:

	Team	Team Rating	Team Salary (millions)
0	Andorra	62.000000	2.900000
1	Zanzibar	NaN	0.000000
2	Dominican Republic	72.000000	14.000000
3	Jersey	NaN	0.000000
4	Saudi Arabia	71.785714	3.450000
...	...	...	...
282	New Caledonia	71.000000	1.800000
283	Brunei	NaN	0.000000
284	Chile	78.642857	11.850000
285	England	83.000000	30.321429
286	Djibouti	NaN	0.000000

287 rows × 3 columns

Perform Merge#2:

In [57]:

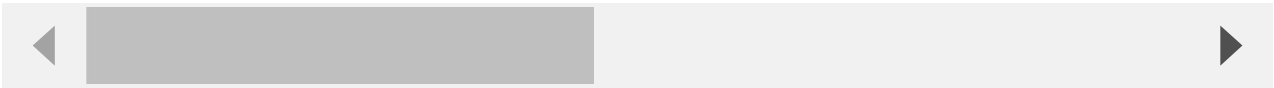
```
# (1) merge the international ratings / internatonal salary in
df_intRatings.columns = ['Home Team','Home Rating','Home Salary']
df_national_2019 = df_national_2019.merge(df_intRatings, how='inner', on='Home Team')
df_intRatings.columns = ['Away Team','Away Rating','Away Salary']
df_national_2019 = df_national_2019.merge(df_intRatings, how='inner', on='Away Team')
# (2) append the new rows into the dataframe
df_club_international_matches = df_matches.append(df_national_2019).copy()
# (3) clean up dataframe for analysis
victor = ['Home' if df_club_international_matches.iloc[x]['Home Team Goals Scored'] > d
victor = ['Draw' if df_club_international_matches.iloc[x]['Home Team Goals Scored'] ==
```

```
# (4) add needed columns for data analysis
df_club_international_matches['Winner'] = victor
df_club_international_matches['Total Goals'] = [df_club_international_matches.iloc[x]['
df_club_international_matches['Majority Possesion'] = ['Home' if df_club_international_
# drop unneeded columns
col =  ['Home Team Blocked Shots', 'Home Team Corners',
        'Home Team Throw Ins', 'Home Team Pass Success %',
        'Home Team Aerials Won', 'Home Team Clearances', 'Home Team Fouls',
        'Home Team Yellow Cards', 'Home Team Second Yellow Cards',
        'Home Team Red Cards', 'Away Team Off Target Shots',
        'Away Team On Target Shots', 'Away Team Total Shots',
        'Away Team Blocked Shots', 'Away Team Corners', 'Away Team Throw Ins',
        'Away Team Pass Success %', 'Away Team Aerials Won',
        'Away Team Clearances', 'Away Team Fouls', 'Away Team Yellow Cards',
        'Away Team Second Yellow Cards', 'Away Team Red Cards'
         ]
df_club_international_matches = df_club_international_matches.drop(col, axis=1)
# (5) One hot encode the Winner column
df_club_international_matches = pd.concat([df_club_international_matches,pd.get_dummies
# output results
df_club_international_matches
```

Out[57]:

	Home Team	Away Team	Score	Half Time Score	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Away Team Possession %	Hor Tea (Targ Sho
0	ATHLETIC	BARCELONA	1-0	0-0	4.7	7.0	5.7	28.0	72.0	
1	CELTA	BARCELONA	2-2	0-1	8.4	7.0	5.9	32.0	68.0	
2	VALENCIA	BARCELONA	2-0	0-0	5.3	7.1	5.1	26.0	74.0	
3	MALLORCA	BARCELONA	0-4	0-2	6.2	5.0	8.4	35.0	65.0	
4	LEGANÉS	BARCELONA	1-2	1-0	5.6	5.5	6.9	23.0	77.0	
...	...	...	...	...	...	...	...	...	...	
10332	Barawa	Yorkshire	2-7	NaN	NaN	NaN	NaN	NaN	NaN	N
10333	Parishes of Jersey	Yorkshire	2-1	NaN	NaN	NaN	NaN	NaN	NaN	N
10334	Barawa	Surrey	1-3	NaN	NaN	NaN	NaN	NaN	NaN	N
10335	Yorkshire	Parishes of Jersey	1-0	NaN	NaN	NaN	NaN	NaN	NaN	N
10336	Kernow	Barawa	5-0	NaN	NaN	NaN	NaN	NaN	NaN	N

10717 rows × 28 columns



In [58]:

```
# drop NaN rating columns
df_club_international_matches.dropna(subset = ["Home Rating"], inplace=True)
```

```
df_club_international_matches.dropna(subset = ["Away Rating"], inplace=True)
df_club_international_matches
```

Out[58]:

	Home Team	Away Team	Score	Half Time Score	Match Excitement	Home Team Rating	Away Team Rating	Home Team Possession %	Away Team Possession %	Hom Tear Of Targe Shot
0	ATHLETIC	BARCELONA	1-0	0-0	4.7	7.0	5.7	28.0	72.0	6.
1	CELTA	BARCELONA	2-2	0-1	8.4	7.0	5.9	32.0	68.0	1.
2	VALENCIA	BARCELONA	2-0	0-0	5.3	7.1	5.1	26.0	74.0	2.
3	MALLORCA	BARCELONA	0-4	0-2	6.2	5.0	8.4	35.0	65.0	7.
4	LEGANÉS	BARCELONA	1-2	1-0	5.6	5.5	6.9	23.0	77.0	5.
...	...	...	...	...	...	...	...	...	...	
9891	Poland	Italy	0-1	NaN	NaN	NaN	NaN	NaN	NaN	Nal
9892	Poland	Italy	0-0	NaN	NaN	NaN	NaN	NaN	NaN	Nal
9893	Uruguay	Italy	2-2	NaN	NaN	NaN	NaN	NaN	NaN	Nal
9894	Finland	Italy	1-2	NaN	NaN	NaN	NaN	NaN	NaN	Nal
9895	Switzerland	Italy	0-0	NaN	NaN	NaN	NaN	NaN	NaN	Nal

5753 rows × 28 columns

Validating Merge#2:

The output below shows there were no null values for home/away rating/salary, meaning the player dataframe merge was sucessful. The null columns where no data could be found is shown aswell. Looking at the row count, the merge was successful because it matched the expected number of games. After the df merges there is a correct row count of 10717 (this is before NaN values are dropped).

- 10337 international games + 380 club games = 10717 total games/rows

In [59]:

```
print(df_club_international_matches.isnull().sum())
```

```
Home Team                     0
Away Team                     0
Score                         0
Half Time Score            5373
Match Excitement           5373
Home Team Rating           5373
Away Team Rating           5373
Home Team Possession %     5373
Away Team Possession %     5373
Home Team Off Target Shots 5373
Home Team On Target Shots  5373
Home Team Total Shots      5373
Home Team Goals Scored        0
```

```
Away Team Goals Scored          0
Home Team Goals Conceeded       0
Away Team Goals Conceeded       0
year                            0
league                          0
Home Rating                     0
Home Salary                     0
Away Rating                     0
Away Salary                     0
Winner                          0
Total Goals                     0
Majority Possesion              0
Winner_Away                     0
Winner_Draw                     0
Winner_Home                     0
dtype: int64
```

Part 5 - Explore Bivariate relationships

- Choose a reasoned set of variables to explore further. You don't have to explore all possible pairs of variables, nor do we want to grade that much. Choose 7 - 9 variables. One should be a variable that you'd like to predict (target variable) using the others (predictor variables).
- List your predictor variables
- List your target variable
- Briefly describe why you have chosen these.

Use any of the available visualizations from Seaborn to explore the relationships between the variables. Explore the relationships among the predictor variables as well as the relationship between each predictor variable and the target variable. Which of the predictor variables are most strongly related? Are there any interesting relationships between categorical predictors and numeric predictors? If there are any dichotomous variables, does that influence any of the relationships? Are the relationships positive or negative?

Below each plot, you should provide a description and interpretation of the plot. Make sure to include why the variables in that plot were chosen and what you hope the reader would gain from it as well.

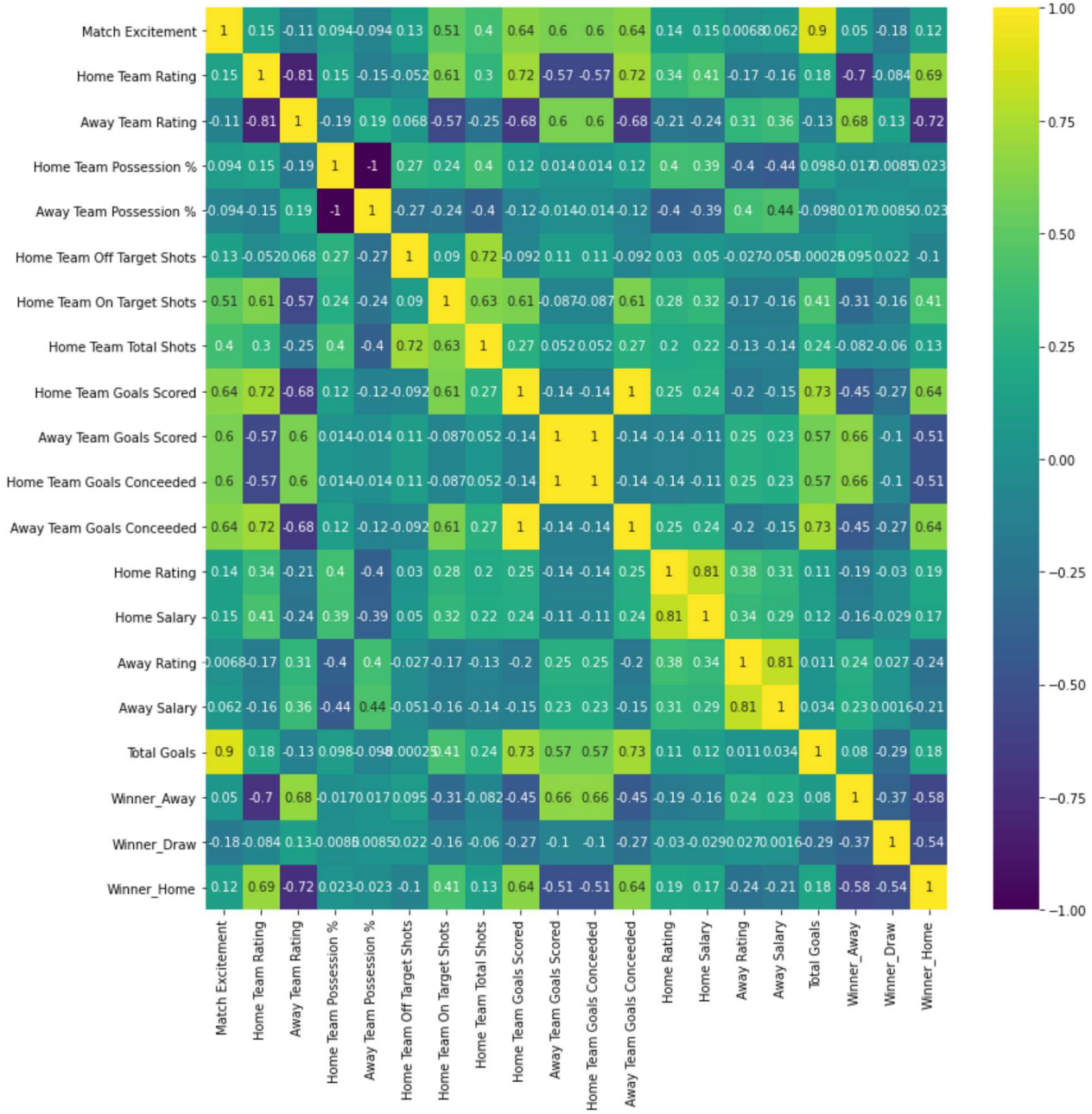
1. Home Team Possession vs Game Outcome (Winner)
2. Away Team Possession vs Game Outcome (Winner)
3. Categorical Variables: international vs spain goals scored

Examining Correlation Heatmap:

In [60]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(13,13))         # Sample figsize in inches
sns.heatmap(df_club_international_matches.corr(), annot = True, cmap = 'viridis')
```

Out[60]:
```
<AxesSubplot:>
```



The heatmap shows the correlation matrix for the dataframe. This plot helps us to understand which variables are related and maybe helps to identify bivariate hidden relationships. For example, the match excitement variable is very closely correlated with the total goals of the match, a relationship that might be worth exploring. Looking at other relationships, the team rating and team salary variables are closely related. This relationships makes sense as better teams tend to have higher salary caps.

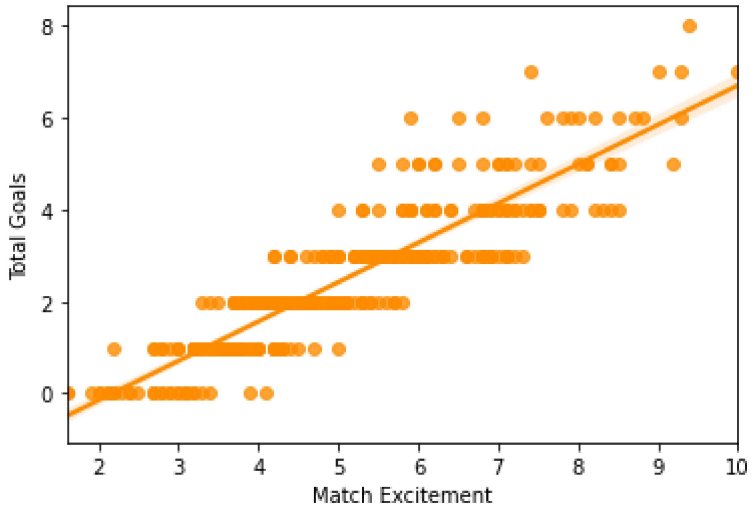
For the one hot encoded variables, the win variables, the correlation matrix often does not produce accurate values due to the computational method used. It works best for numerical variables. (https://stats.stackexchange.com/questions/256344/why-is-correlation-not-very-useful-when-one-of-the-variables-is-categorical)

Relationship #1 Match Excitement and Total Goals: Bivariate analysis of two numerical variables The heatmap shows the variables 'Match Excitement' and 'Total Goals' have a correlation of 0.90. Both variables are numerical. The Match Excitement will be the predictor variable (x-axis) and the

Total Goals will be the target variable. A scatterplot can be used to perform bivariate analysis on two numerical variables (https://shecancode.io/blog/univariate-and-bivariate-analysis-usingseaborn). A regplot was used to add a regression line

In [61]:
```
df_spain = df_club_international_matches[df_club_international_matches["league"] == 'sp
df_spain = df_spain.copy() # create a copy of the slice, slice should be read only
# only spain games have match excitement, so only use those
sns.regplot(df_club_international_matches["Match Excitement"],
                df_club_international_matches['Total Goals'], color='darkorange')
```

Out[61]:
```
<AxesSubplot:xlabel='Match Excitement', ylabel='Total Goals'>
```



The purpose of this plot was is to show their is a relationship between match match excitement and total goals. The higher the match excitement the higher the total number of goals on average. The regression line through the data has a nearly linear slope, meaning there is a positive linear relationship between the variables. For further exploration, how the dataset calculated the "match excitement" data field should be looked into.

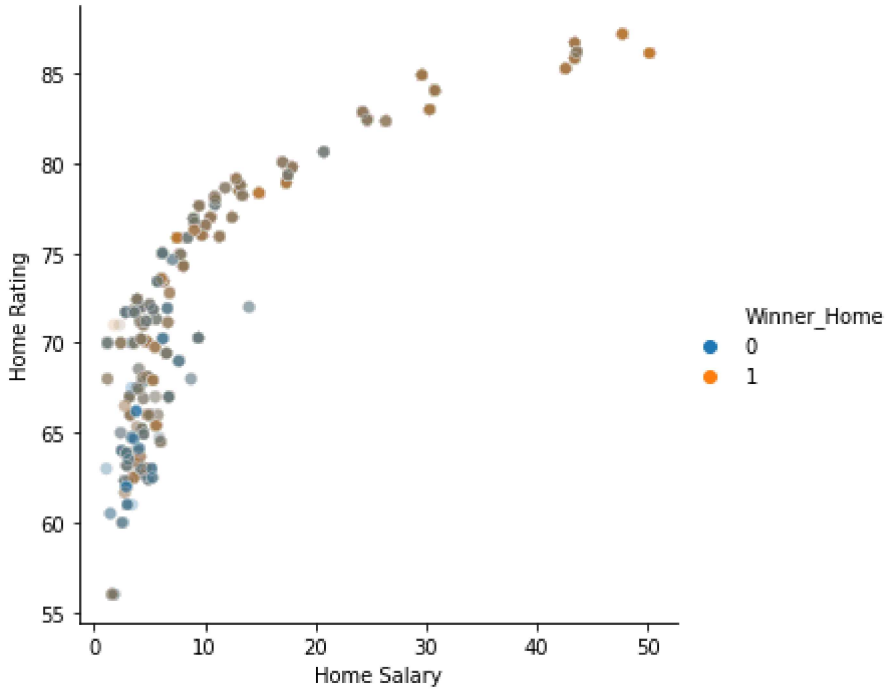
Relationship #2 Team Rating/Salay predicting Game Results: A seaborn relplot will be used to plot these two numeric variables examining the relationship between the target categorical variable (Victor).

- After mergining the datasets, their is a team rating column and team salary column. Based off the heat matrix, we can see the team rating and team salary are closely correlated. The home team rating and home team salary will be used as predictors for victory. The rating and salary will be used to predict the one hot encoded target column 'Winner Home'.
- The parent dataframe will sliced into two dataframe based whether club or international games. This was due to the disparity of ratings/salary in club versus international countries

In [62]:
```
df_international = df_club_international_matches[df_club_international_matches["league"
df_international = df_international.copy() # create a copy of the slice, slice should b

sns.relplot(data=df_international, x="Home Salary", y="Home Rating", hue="Winner_Home",
```

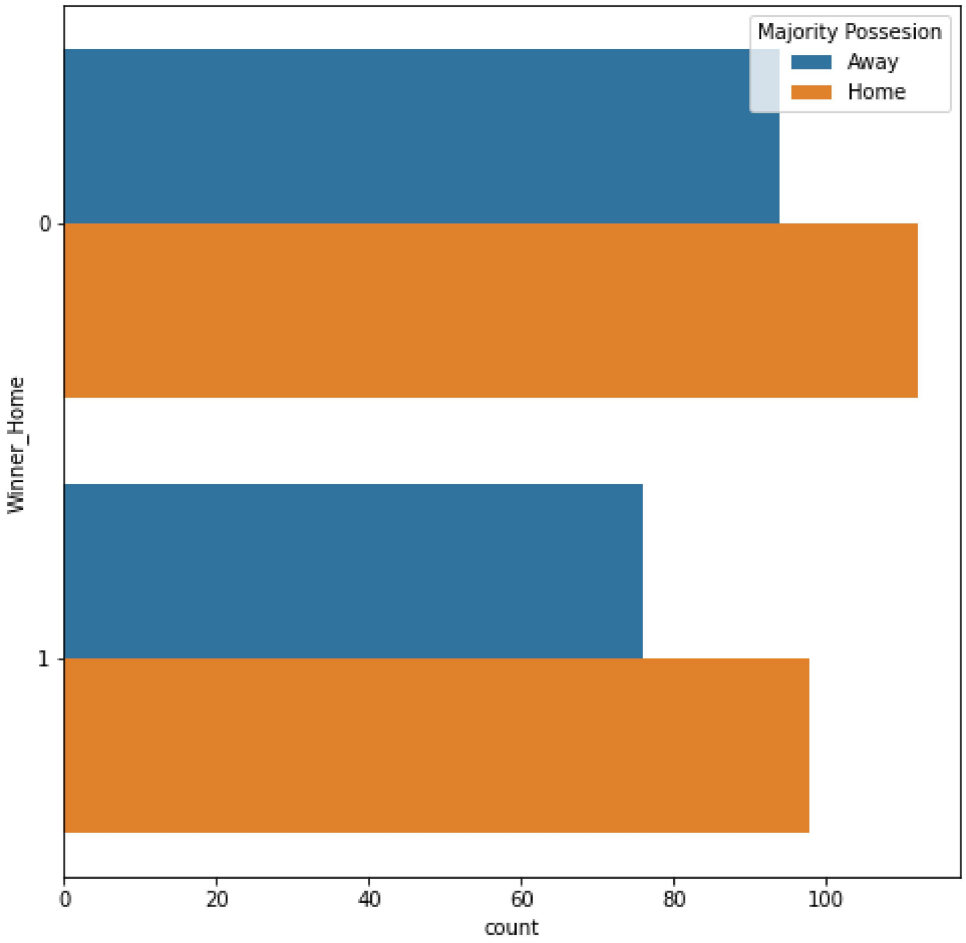
Out[62]:
```
<seaborn.axisgrid.FacetGrid at 0x1cff5042970>
```



The relationship between rating/salary and victor is not as strong as expected, but setting the alpha value to a low value helps make the relationship more clear. When the home team has a lower salary/rating, the data points tend to be darker/oranger meaning a loss is more likely (home team lost more on average). When the home salary and home rating is higher, the data points become more blue meaning a win is more likely on average for a higher rated/salary team.

Relationship #3 Possesion predicting Game Results: Categorical variables
For the club games, the ball possesion of each team is listed. Since these are categorical variables, the correlation/relationship between the variables is unknown. This graph affects to explore if more ball possesion directly means a higher chance of victory. Possesion is the predictor variable and winner_home (the victor) is the target variable.

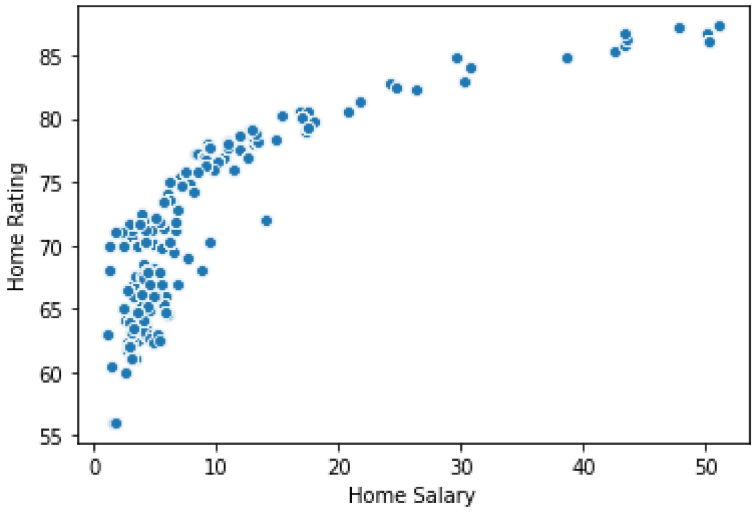
In [63]:
```
fig, ax = plt.subplots(figsize=(8,8))         # Sample figsize in inches
ax = sns.countplot(y="Winner_Home", hue="Majority Possesion", data=df_spain)
```



Regardless of if the home team won and lost, the home team possessed the ball more than the away team. This means that ball possesion in of itself is not an accurate predictor of the winner of the game. This graph also revealed that the home team lost more than it won. Meaning, a home field advantage relationship does not really need to be explored, at least in the Spanish league.

Relationship #4 Relationship between salary and player rating: scatterplot to compare two numerical continous bivariate variables
The correlation matrix shows that the salary has a high correlation with player rating and vice vesra. It was important to ensure these variables are closely related because they are used to support other claims. The predictor variable was salary and the target variable was rating.

In [64]:
```
x = sns.scatterplot(x = df_club_international_matches['Home Salary'],
                    y = df_club_international_matches['Home Rating'])
```

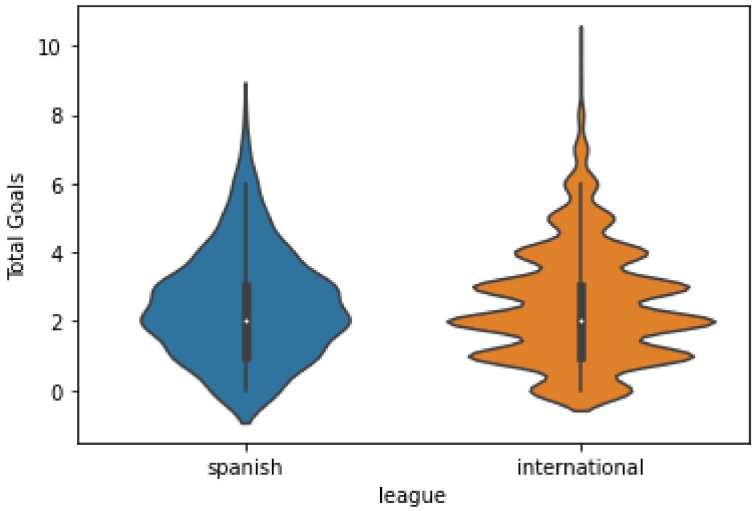


This graph shows that as the home salary increases the home rating appears to increase. The graph appears to grow similarly to the function f(x) = log(n). The relationship between the home salary and home rating are tightly corelated, they both tend to grow at a rate of log(n).

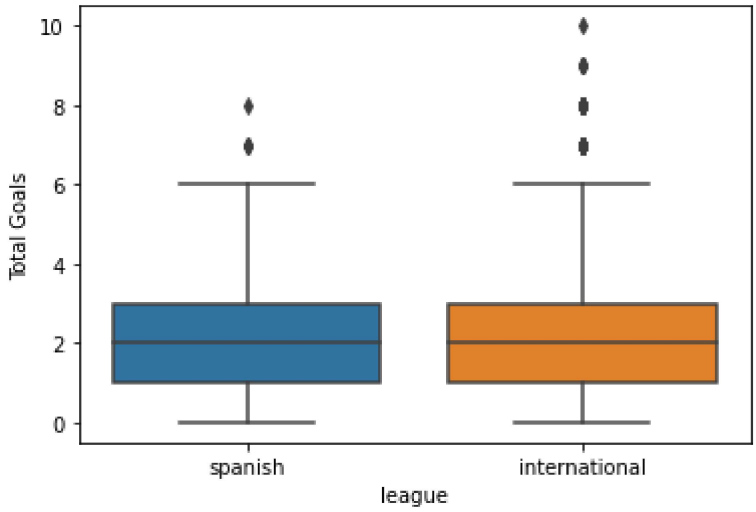
Relationship #5 Club Goal scoring versus International Goal scoring :

Next, I wanted to examine the categorical variable "league" and its affect on the outcome of the game. Meaning, the predictor variable would be league and the target variable would be the number of goals. The hypothesis is that international games may be higher scoring games than club games due to the potential for greater skill imbalence. For example a country like Brazil has multiple world cup victories, players on many top level professional teams, and greater resources. When Brazil goes up against a smaller country without as many experienced players the game may be higher scoring (Brazil would score many goals). In contrast, club teams tend to have more balanced leagues and matchups.

In [65]:
```
x = sns.violinplot(y = df_club_international_matches['Total Goals'], x = df_club_intern
```



In [66]:
```
x = sns.boxplot(y = df_club_international_matches['Total Goals'], x = df_club_internati
```

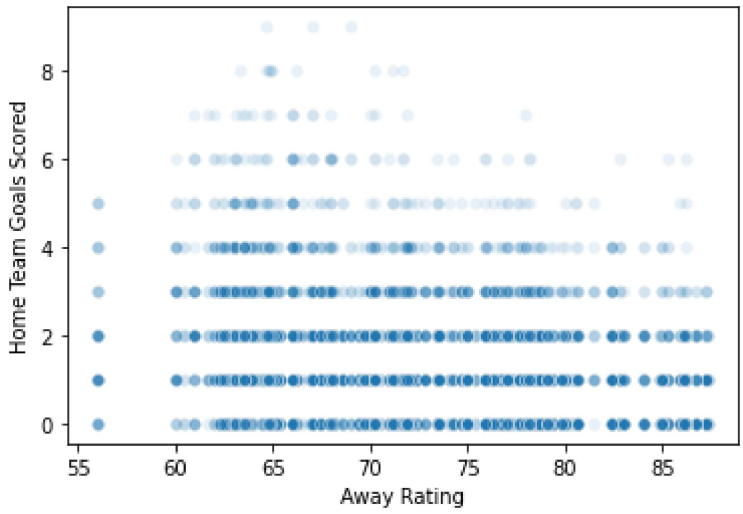


After examining the box plot and violion plot the prediction was incorrect. There is no considerable difference in the distribution of total goals in club versus international professional soccer matches. However, there were more considerable outliers or "blowout" games in the international category.

Relationship #6 Goals Scored versus Team Rating :

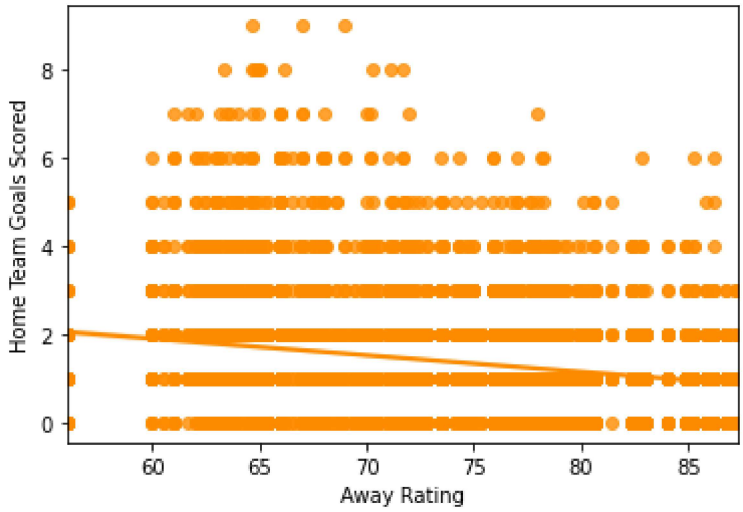
When examining the heatmap, first strong positive correlation was valued. However, the team rating and goals scored variables appear to be somewhat negatively correlated.

In [67]:
```
x = sns.scatterplot(x = df_club_international_matches['Away Rating'], y = df_club_inter
```



In [68]:
```
sns.regplot(df_club_international_matches["Away Rating"],
                    df_club_international_matches['Home Team Goals Scored'], color='dar
```

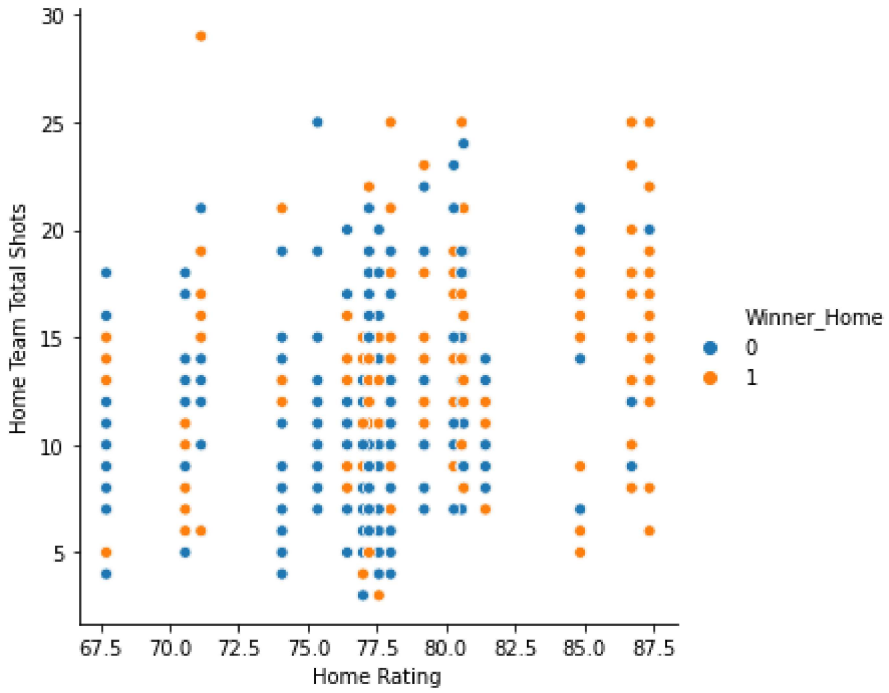
Out[68]:
```
<AxesSubplot:xlabel='Away Rating', ylabel='Home Team Goals Scored'>
```



Both the regression line and scatterplot show that the variables of opposing team rating and current team goals scored are somewhat negatively correlated. By setting the alpha to a low value on the scatterplot, we can see where the kernel density of goals are concerntated. For lower rated teams, the number of goals scored distribution is much wider with more goals being scored against them. For higher rated teams, the total number of goals scored against them is much lower. The regression plot shows a negative correlation.

Relationship #7 Total Shots predicting winner of the game:
The logic behind this was if more total shots are taken, that team is more likely to win the game. The Home Team total shots variable and Home rating were utilized to predict the target variable of victor (winner*home*

In [69]:
```
x = sns.relplot(data=df_spain, x="Home Rating", y="Home Team Total Shots", hue="Winner_
```



The graph distribution shows that more losses appear when less total shots are taken, and more wins appear for the home team when more total shots are taken. The home rating was graphed on

the x-axis, so the upper right hand corner contains the highest concentration of wins (highest shots and highest team rating).