Source of the data:

- We obtained this dataset from the website Kaggle.
- https://www.kaggle.com/karangadiya/fifa19

Gadiya, K. (2018, December 21). FIFA 19 complete player dataset. Kaggle. Retrieved December 11, 2021, from https://www.kaggle.com/karangadiya/fifa19

Observations:

- The observations are people, specifically FiFA 19 soccer players. We minimized the number of observations used to only 1000 players, in order to place the focus on the best players..

- We removed some variables and selected those that are usefeul when determining significant relationships and intrepretations and relevant to our goal, which is determining the best player card. Before we could acquire our statistical data and plot the graphs, we had to modify the variable categories. We changed the following variables to categories: Preferred Foot, International Reputation, Weak Foot, and Skill Moves.

In [3]:
```
import numpy as np
```

In [11]:
```
import pandas as pd
```

In [5]:
```
import statistics as stats
```

In [6]:
```
import matplotlib.pyplot as plt
```

In [7]:
```
import scipy as sp
```

In [8]:
```
import seaborn as sns
```

In [14]:
```
df=pd.read_csv("data.csv")
```

In [19]:
```
df['PreferredFoot'] = df.PreferredFoot.astype('category')
```

In [20]:
```
df['InternationalReputation'] = df.InternationalReputation.astype('category')
```

In [ ]:
```
df['WeakFoot'] = df.WeakFoot.astype('category')
```

In [ ]:
```
df['SkillMoves'] = df.SkillMoves.astype('category')
```

In [15]:
```
df
```

Out[15]:

	ID	Name	Age	Nationality	Overall	Potential	Club	ValueInMillions	WageInThousands

	ID	Name	Age	Nationality	Overall	Potential	Club	ValueInMillions	WageInThousands
0	158023	L. Messi	31	Argentina	94	94	FC Barcelona	110.5	565
1	20801	Cristiano Ronaldo	33	Portugal	94	94	Juventus	77.0	405
2	190871	Neymar Jr	26	Brazil	92	93	Paris Saint-Germain	118.5	290
3	193080	De Gea	27	Spain	91	93	Manchester United	72.0	260
4	192985	K. De Bruyne	27	Belgium	91	92	Manchester City	102.0	355
...	...	...	...	...	...	...	...	...	...
996	193361	R. Soriano	27	Italy	77	77	Torino	9.5	31
997	223058	D. Kuzyaev	25	Russia	77	80	NaN	0.0	0
998	152916	R. Sambueza	34	Argentina	77	77	Deportivo Toluca	4.5	26
999	194644	Montoya	27	Spain	77	78	Brighton & Hove Albion	8.0	43
1000	185174	A. Poli	28	Italy	77	77	Bologna	9.0	34

1001 rows × 29 columns

In [ ]:

```
df['WeightInPounds'].mean()
```

Out[ ]:

```
170.04695304695304
```

In [ ]:

```
df['WeightInPounds'].median()
```

Out[ ]:

```
170.0
```

In [ ]:

```
df['WeightInPounds'].quantile([0,.25,.5,.75,1])
```

Out[ ]:

```
0.00    130.0
0.25    159.0
0.50    170.0
0.75    181.0
1.00    220.0
Name: WeightInPounds, dtype: float64
```

In [ ]:

```
df['WeightInPounds'].var()
```

Out[ ]:

```
259.4867932067932
```

In [ ]:

```
df['WeightInPounds'].std()
```

Out[ ]:

```
16.108593768755647
```

In [ ]:

```
df['HeightInInches'].mean()
```

Out[ ]:
```
5.834625374625373
```

In [ ]:
```python
df['HeightInInches'].median()
```

Out[ ]:
```
6.0
```

In [ ]:
```python
df['HeightInInches'].quantile([0,.25,.5,.75,1])
```

Out[ ]:
```
0.00    5.1
0.25    5.6
0.50    6.0
0.75    6.2
1.00    6.7
Name: HeightInInches, dtype: float64
```

In [ ]:
```python
df['HeightInInches'].var()
```

Out[ ]:
```
0.2048174845154845
```

In [ ]:
```python
df['HeightInInches'].std()
```

Out[ ]:
```
0.45256765739001337
```

In [ ]:
```python
df['ReleaseClause'].mean()
```

Out[ ]:
```
36.424921135646684
```

In [ ]:
```python
df['ReleaseClause'].median()
```

Out[ ]:
```
28.3
```

In [ ]:
```python
df['ReleaseClause'].quantile([0,.25,.5,.75,1])
```

Out[ ]:
```
0.00      1.10
0.25     19.00
0.50     28.30
0.75     43.65
1.00    228.10
Name: ReleaseClause, dtype: float64
```

In [ ]:
```python
df['ReleaseClause'].var()
```

Out[ ]:
```
819.426251967458
```

In [ ]:
```python
df['ReleaseClause'].std()
```

Out[ ]:
```
28.625622298344155
```

In [ ]:
```python
df['WageInThousands'].mean()
```

Out[ ]:
```
65.4085914085914
```

In [ ]:
```
df['WageInThousands'].median()
```

Out[ ]:
```
46.0
```

In [ ]:
```
df['WageInThousands'].quantile([0,.25,.5,.75,1])
```

Out[ ]:
```
0.00      0.0
0.25     26.0
0.50     46.0
0.75     84.0
1.00    565.0
Name: WageInThousands, dtype: float64
```

In [ ]:
```
df['WageInThousands'].var()
```

Out[ ]:
```
3992.0118861138862
```

In [ ]:
```
df['WageInThousands'].std()
```

Out[ ]:
```
63.18237005774543
```

In [ ]:
```
df['ValueInMillions'].mean()
```

Out[ ]:
```
19.35764235764236
```

In [ ]:
```
df['ValueInMillions'].median()
```

Out[ ]:
```
14.5
```

In [ ]:
```
df['ValueInMillions'].var()
```

Out[ ]:
```
548.2524840359641
```

In [ ]:
```
df['ValueInMillions'].std()
```

Out[ ]:
```
23.41479199215667
```

In [ ]:
```
df['ValueInMillions'].quantile([0,.25,.5,.75,1])
```

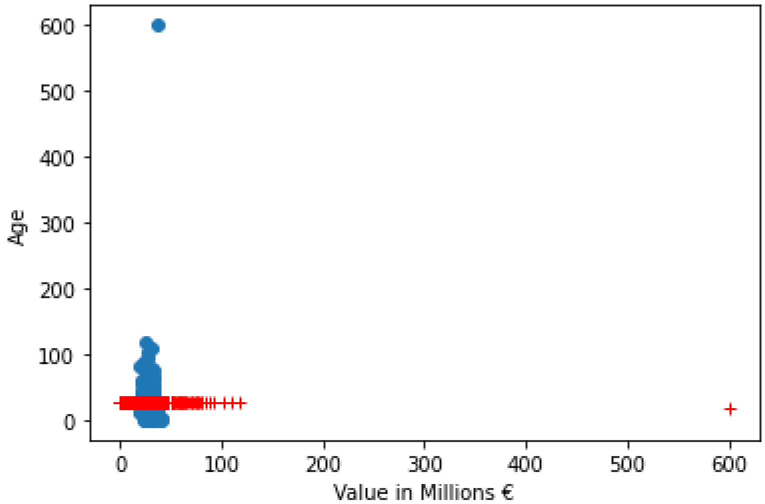
Out[ ]:
```
0.00      0.0
0.25     10.0
0.50     14.5
0.75     22.0
1.00    600.0
Name: ValueInMillions, dtype: float64
```

In [ ]:
```
df['Age'].corr(df['ValueInMillions'])
```

Out[ ]:
```
-0.0880316465244258
```

In [ ]:

```
x = df['ValueInMillions']
y = df['Age']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b,'r+')
plt.scatter(df['Age'],df['ValueInMillions'])
plt.ylabel("Age")
plt.xlabel("Value in Millions €")
plt.show()
```



In [ ]:

```
100*df['Dribbling'].value_counts(normalize=True)
```

Out[ ]:

```
78     5.794206
80     5.094905
76     4.795205
84     4.395604
82     4.395604
         ...
94     0.099900
30     0.099900
95     0.099900
96     0.099900
34     0.099900
Name: Dribbling, Length: 80, dtype: float64
```

In [ ]:

```
100*df['Jumping'].value_counts(normalize=True)
```

Out[ ]:

```
73     4.795205
76     4.695305
74     4.495504
75     3.896104
69     3.696304
         ...
40     0.099900
43     0.099900
95     0.099900
31     0.099900
42     0.099900
Name: Jumping, Length: 66, dtype: float64
```

In [ ]:

```
100*df['Strength'].value_counts(normalize=True)
```

Out[ ]:

```
78     4.695305
68     4.595405
72     4.195804
79     4.195804
76     4.095904
```

```
       ...
41    0.099900
35    0.099900
32    0.099900
44    0.099900
30    0.099900
Name: Strength, Length: 63, dtype: float64
```

In [ ]:

```
100*df['SprintSpeed'].value_counts(normalize=True)
```

Out[ ]:

```
76    4.895105
69    4.695305
77    4.295704
79    4.295704
75    3.696304
       ...
95    0.199800
33    0.199800
39    0.199800
36    0.099900
31    0.099900
Name: SprintSpeed, Length: 66, dtype: float64
```

In [ ]:

```
100*df['BallControl'].value_counts(normalize=True)
```

Out[ ]:

```
80    7.592408
79    6.693307
83    6.493506
78    6.493506
81    6.293706
       ...
36    0.099900
38    0.099900
37    0.099900
42    0.099900
55    0.099900
Name: BallControl, Length: 75, dtype: float64
```

Notice that the athlete has a higher probability of possessing a higher value if he is under the age of 30.

In [ ]:

```
df['Jumping'].corr(df['Strength'])
```

Out[ ]:

```
0.3361784412480953
```

There is a direct association between the players' strength and their jumping abilities, as can be seen in this graph. The graph implies that a player's strength correlates with their ability to jump higher, which is logical.

In [ ]:

```
df['SprintSpeed'].corr(df['Agility'])
```

Out[ ]:

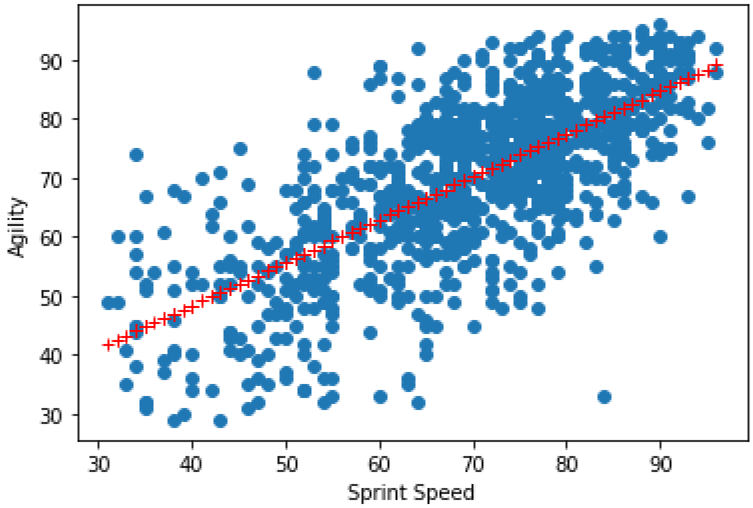
```
0.705399575116226
```

In [ ]:

```
x = df['SprintSpeed']
y = df['Agility']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b,'r+')
plt.scatter(df['SprintSpeed'],df['Agility'])
plt.xlabel("Sprint Speed")
plt.ylabel("Agility")
```

Out[ ]:

```
Text(0, 0.5, 'Agility')
```



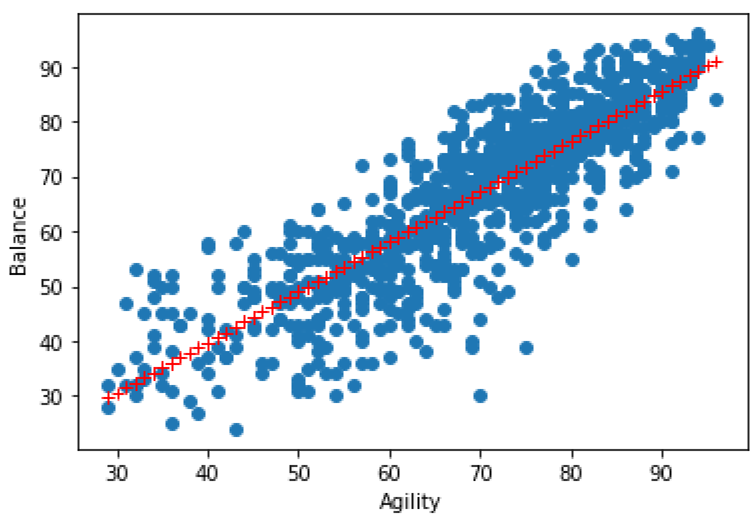
Agility enhances a player's sprint speed by helping the body maintain good alignment and posture throughout movement, which explains the positive correlation between these two variables.

In [ ]:
```
df['Agility'].corr(df['Balance'])
```

Out[ ]:
```
0.8588518827099209
```

In [ ]:
```
x = df['Agility']
y = df['Balance']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, 'r+')
plt.scatter(df['Agility'],df['Balance'])
plt.xlabel("Agility")
plt.ylabel("Balance")
```

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



There is a clear link between agility and balance, demonstrating their interdependence. Balance programmes for soccer players may be an effective way to improve agility, which in turn may help prevent injuries.

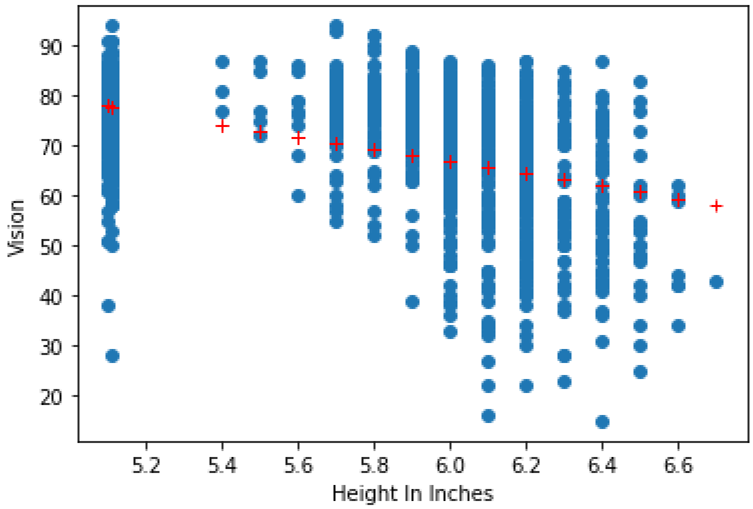
In [ ]:
```
df['HeightInInches'].corr(df['Vision'])
```

Out[ ]:
```
-0.40968454146379213
```

In [ ]:
```
x = df['HeightInInches']
```

```
y = df['Vision']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, 'r+')
plt.scatter(df['HeightInInches'],df['Vision'])
plt.xlabel("Height In Inches")
plt.ylabel("Vision")
```

Out[ ]: Text(0, 0.5, 'Vision')



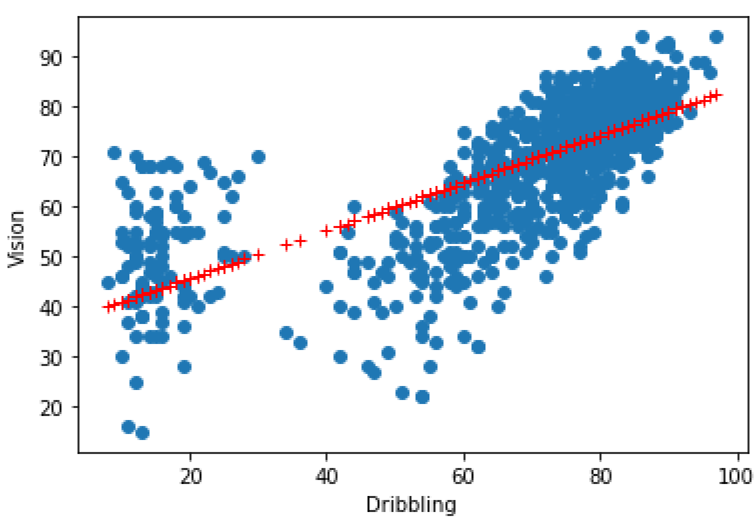
There is a strong correlation between height and vision in this graph. This could be explained with the fact that taller people tend to have better eyesight.

In [ ]:
```
df['Dribbling'].corr(df['Vision'])
```

Out[ ]: 0.7349019081632446

In [ ]:
```
x = df['Dribbling']
y = df['Vision']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, 'r+')
plt.scatter(df['Dribbling'],df['Vision'])
plt.xlabel("Dribbling")
plt.ylabel("Vision")
```

Out[ ]: Text(0, 0.5, 'Vision')



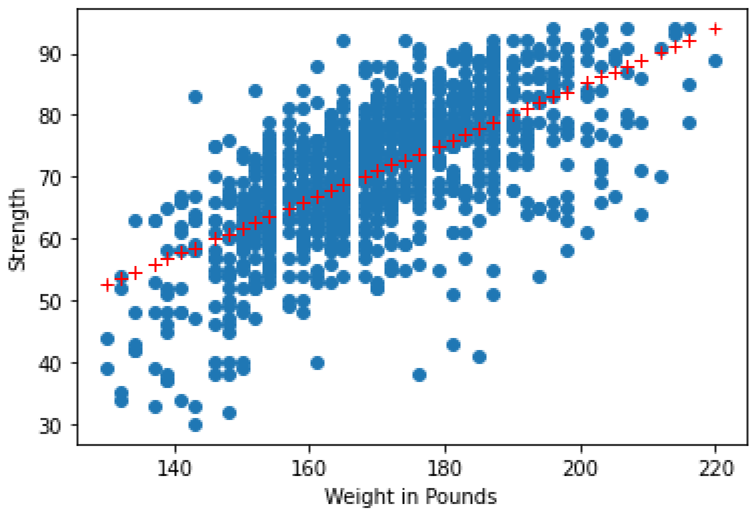
There is a significant positive correlation here. The fact that higher dribbling skills necessitate high levels of vision could explain this association. It is possible to deduce that athletes who excel at dribbling have great eyesight. The stronger a player's eyesight, the better his dribbling skills are likely to be.

In [ ]:
```
df['WeightInPounds'].corr(df['Strength'])
```

Out[ ]:
```
0.6540476289586655
```

In [ ]:
```
x = df['WeightInPounds']
y = df['Strength']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, 'r+')
plt.scatter(df['WeightInPounds'],df['Strength'])
plt.xlabel("Weight in Pounds")
plt.ylabel("Strength")
```

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



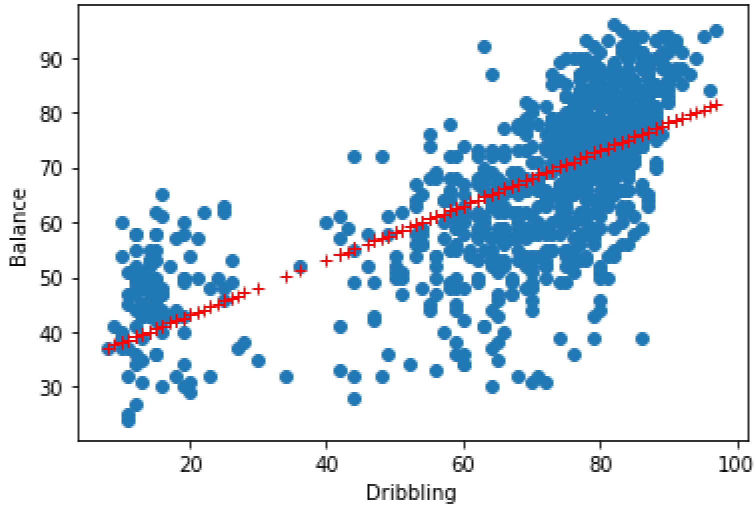
Here, there is a substantial positive link between weight and strength. This could be explained by the fact that these players all have fit bodies, implying that weight here means muscle mass and a larger physique. Greater muscular mass equates to greater strength.

In [ ]:
```
df['Dribbling'].corr(df['Balance'])
```

Out[ ]:
```
0.7041649248407201
```

In [ ]:
```
x = df['Dribbling']
y = df['Balance']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, 'r+')
plt.scatter(df['Dribbling'],df['Balance'])
plt.xlabel("Dribbling")
plt.ylabel("Balance")
```

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



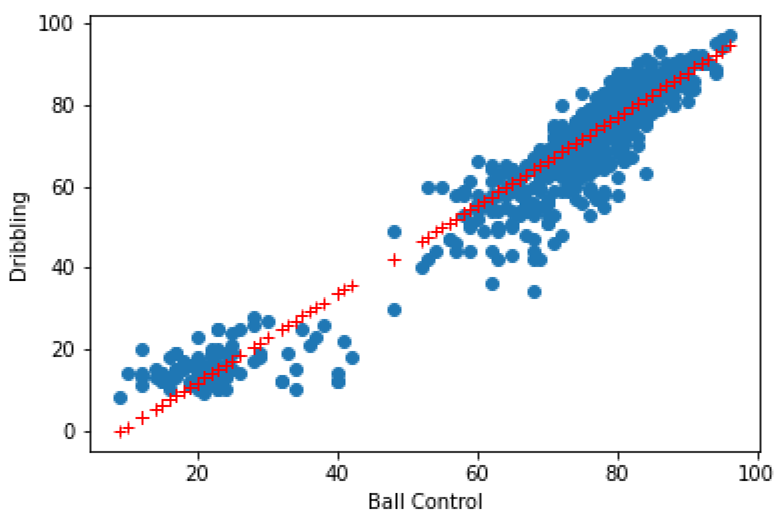
There is a strong link between dribbling and balance in this situation. Dribbling necessitates a high level of stability and balance, which explains the connection.

In [ ]:
```
df['BallControl'].corr(df['Dribbling'])
```

Out[ ]:
```
0.9647121301117824
```

In [ ]:
```
x = df['BallControl']
y = df['Dribbling']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, 'r+')
plt.scatter(df['BallControl'],df['Dribbling'],)
plt.xlabel("Ball Control")
plt.ylabel("Dribbling")
```

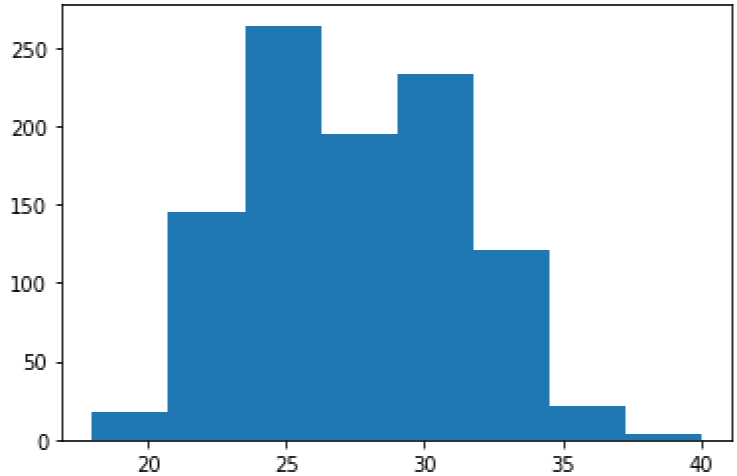
Out[ ]:
```
Text(0, 0.5, 'Dribbling')
```



Ball control and dribbling have a high positive association. Because dribbling necessitates complete control of the ball, this relationship makes sense because they are mutually reliant.

In [ ]:
```
a=df['Age']
```

In [ ]:
```
plt.hist(a, bins = 8)
plt.show()
```



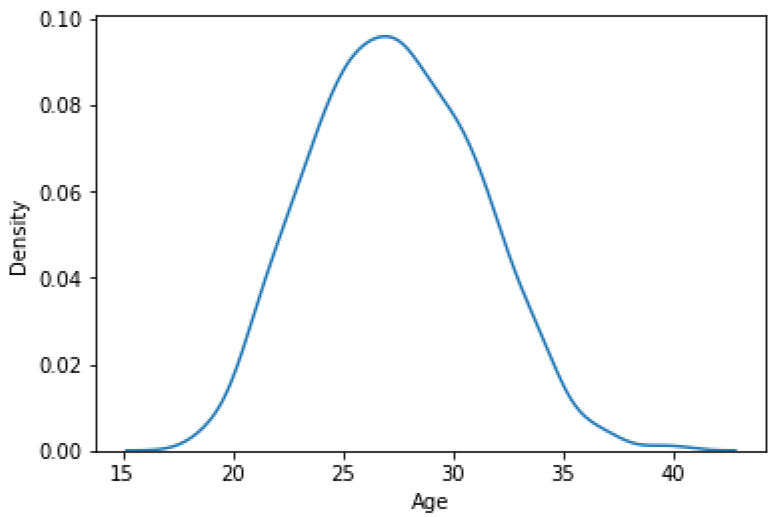
Notice that the top 1000 players aren't bound by age, meaning there's a broad range of players from 19 years old (K. Mbapp) to 40 years old (G. Buffon). Also, either by examining the graph or by constructing a frequency table, we can see that most players lie within the interval of age 24 to age 32.

In [ ]:

```
sns.kdeplot(df['Age'])
```

Out[ ]:

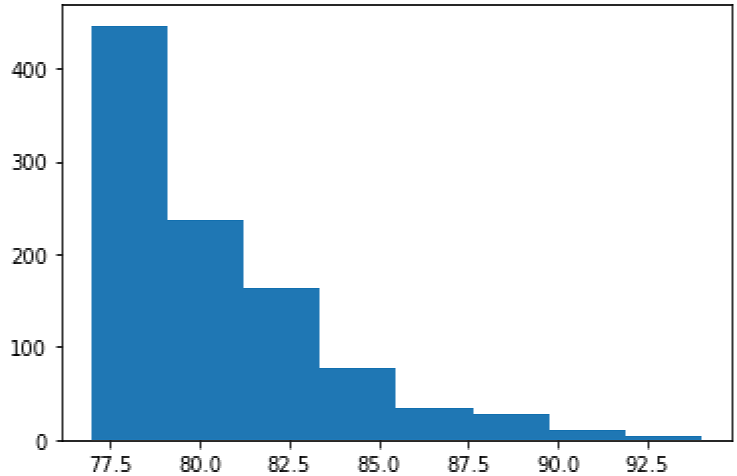
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



The density plot above displays the age of the players recorded in this data set. It is clear that most players are aged between 20 and 35, and only a minimal number of players are aged 15 or 40.

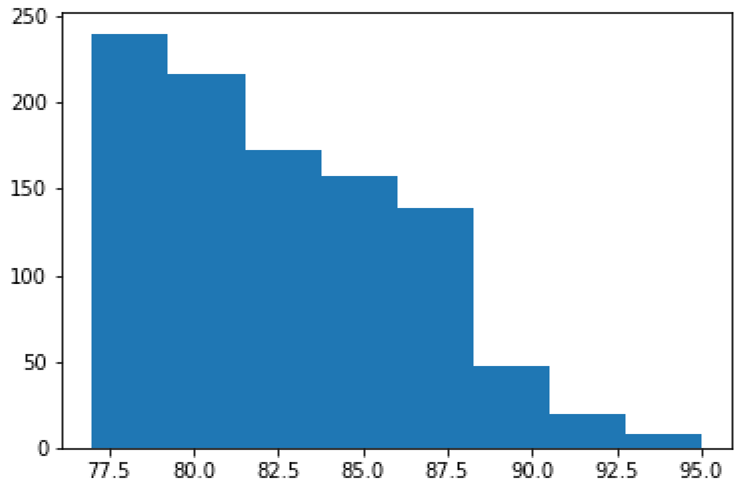
In [ ]:

```
o=df['Overall']
plt.hist(o, bins = 8)
plt.show()
```



In [ ]:

```
p=df['Potential']
plt.hist(p, bins = 8)
plt.show()
```



In [ ]:

```
df['Overall'].corr(df['Potential'])
```

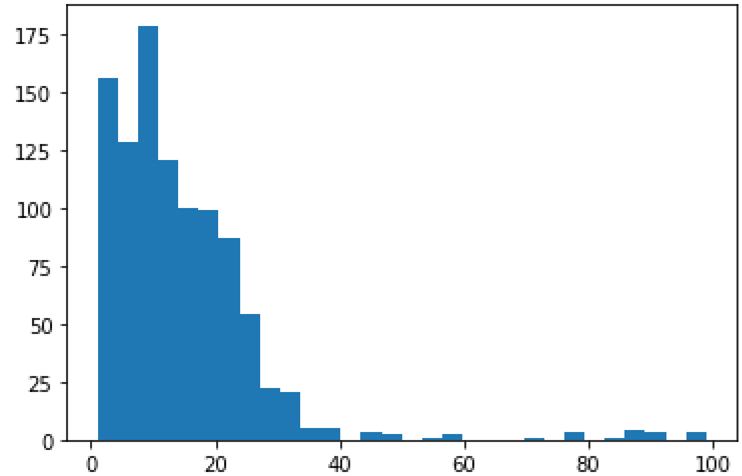
Out[ ]:

```
0.7489315036169497
```

As shown in the graphs and by the correlation, an increase in the players' overall abilities is evident, meaning that they are assigned higher potential abilities and are expected to show improvements throughout the season.

In [ ]:

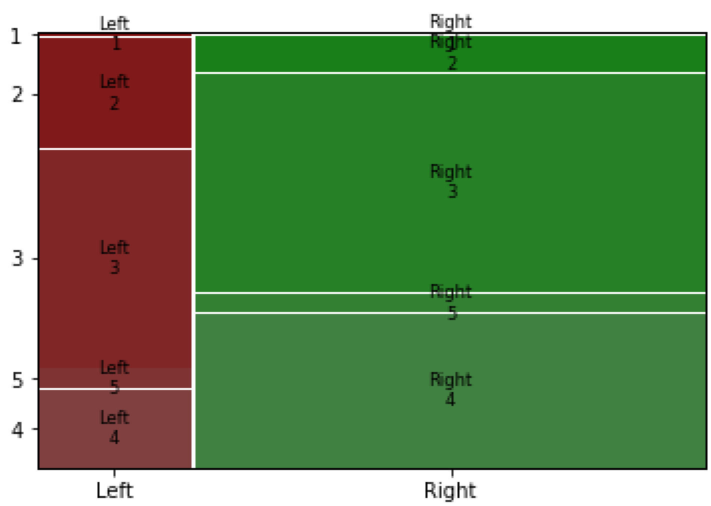
```
j=df['JerseyNumber']
plt.hist(j, bins = 30)
plt.show()
```



The graph shows that the number 10 has the highest count, indicating that it is the most worn number. Most of the best players to ever play the game wore this number, primarily for historical reasons. Pele and Maradona, two of the greatest players of all time, both utilized it, which is why the number 10 is associated with brilliance, therefore making it every player's desire. It is sometimes granted to players based on their position rather than their abilities.

In [ ]:

```
from statsmodels.graphics.mosaicplot import mosaic
mosaic(data=df, index=['PreferredFoot', 'WeakFoot'])
plt.show()
```



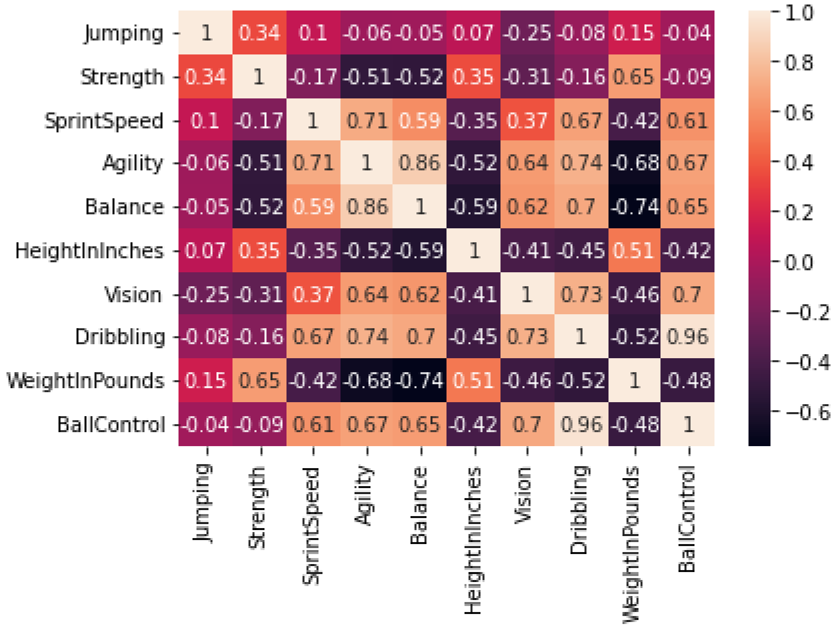
The mosaic plot displays the relationship between the Preferred Foot and Weak Foot variables. Each block displays the number of players that use either their right or left foot.

In [ ]:

```
cormat=df[['Jumping','Strength','SprintSpeed','Agility','Balance','HeightInInches','Visio
cormat

sns.heatmap(cormat, annot=True)
```

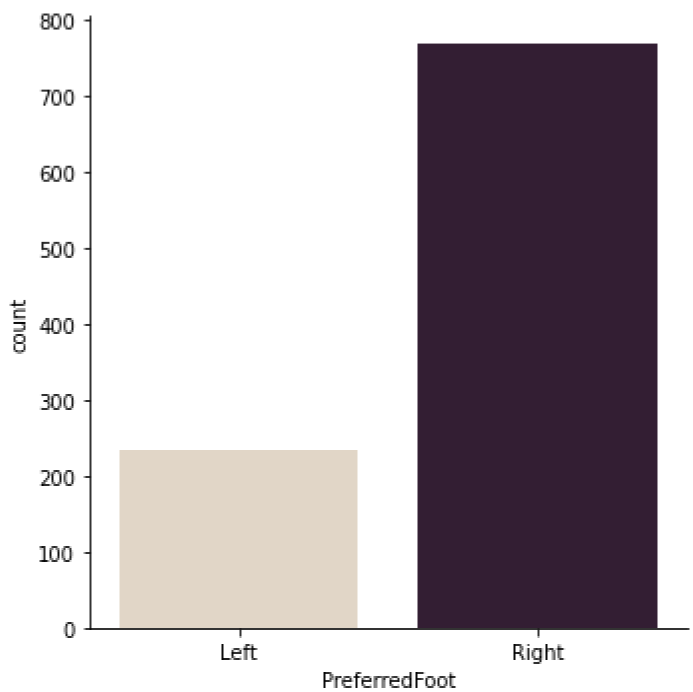
Out[ ]: <AxesSubplot:>



The heat map displays the correlation between all variables

In [ ]:
```
sns.catplot(x="PreferredFoot", kind="count", palette="ch:.25", data=df)
```

Out[ ]: `<seaborn.axisgrid.FacetGrid at 0x7ff5822416d0>`



In [ ]:
```
df['PreferredFoot'].value_counts()
```

Out[ ]:
```
Right     768
Left      233
Name: PreferredFoot, dtype: int64
```

In [ ]:
```
tb=pd.crosstab(index=df['PreferredFoot'], columns='count')
tb
```

Out[ ]:

col_0	count
PreferredFoot	

col_0	count
PreferredFoot	
Left	233
Right	768

In [ ]:

```
pd.crosstab(index=df['WeakFoot'], columns='count')
```

Out[ ]:

col_0	count
WeakFoot	
1	4
2	125
3	511
4	317
5	44

In [ ]:

```
pd.crosstab(index=df['InternationalReputation'], columns='count')
```

Out[ ]:

col_0	count
InternationalReputation	
1	245
2	431
3	268
4	51
5	6

In [ ]:

```
pd.crosstab(index=df['SkillMoves'], columns='count')
```

Out[ ]:

col_0	count
SkillMoves	
1	109
2	179
3	346
4	335
5	32

In [ ]:

```
pd.crosstab(index=df['Overall'], columns='count')
```

Out[ ]:

col_0	count
Overall	
77	107

col_0	count
Overall	
78	168
79	171
80	143
81	93
82	94
83	70
84	45
85	33
86	22
87	13
88	17
89	11
90	5
91	6
92	1
94	2

In [ ]:

```
pd.crosstab(index=df['Potential'], columns='count')
```

Out[ ]:

col_0	count
Potential	
77	71
78	72
79	97
80	129
81	88
82	84
83	89
84	85
85	72
86	56
87	46
88	37
89	26
90	21
91	12
92	8

col_0	count
Potential	
93	4
94	3
95	1

In [ ]:

```
pd.crosstab(index=df['Nationality'], columns='count')
```

Out[ ]:

col_0	count
Nationality	
Albania	2
Algeria	7
Angola	1
Argentina	72
Armenia	1
...	...
United States	4
Uruguay	19
Venezuela	3
Wales	3
Zimbabwe	1

75 rows × 1 columns

In [ ]:

```
pd.crosstab(index=df['WorkRate'], columns='count')
```

Out[ ]:

col_0	count
WorkRate	
High/ High	150
High/ Low	53
High/ Medium	252
Low/ High	16
Low/ Medium	8
Medium/ High	143
Medium/ Low	40
Medium/ Medium	339

In [16]:

```
pd.crosstab(df['Age'], df['Overall'],margins=True)
```

Out[16]:

Overall	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	94	All

Overall	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	94	All
Age																		
18	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
19	0	3	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	7
20	0	4	3	1	2	0	0	0	0	0	0	0	0	0	0	0	0	10
21	6	9	8	6	2	3	2	0	0	0	0	0	0	0	0	0	0	36
22	1	8	8	8	6	4	6	4	1	1	0	0	0	0	0	0	0	47
23	8	17	10	5	3	6	5	2	4	2	0	0	0	0	0	0	0	62
24	5	14	12	15	11	4	7	4	1	1	1	0	2	0	0	0	0	77
25	5	20	16	11	15	10	4	4	1	2	3	0	0	1	0	0	0	92
26	4	19	16	14	5	9	9	5	2	3	0	6	2	0	0	1	0	95
27	13	16	16	12	6	13	7	2	3	3	1	1	2	0	3	0	0	98
28	15	8	24	18	8	6	7	3	2	3	1	1	0	1	0	0	0	97
29	7	14	14	6	9	7	7	4	6	3	1	2	1	1	0	0	0	82
30	14	14	14	6	6	7	6	6	3	0	2	2	1	0	0	0	0	81
31	3	8	10	12	7	9	4	4	3	2	4	1	1	0	1	0	1	70
32	7	6	9	10	5	7	0	3	3	0	0	0	1	2	2	0	0	55
33	6	3	3	7	5	1	3	0	1	1	0	2	1	0	0	0	1	34
34	9	3	5	8	1	1	1	3	0	1	0	0	0	0	0	0	0	32
35	2	1	1	0	1	1	2	0	2	0	0	0	0	0	0	0	0	10
36	0	0	0	2	1	2	0	0	1	0	0	0	0	0	0	0	0	6
37	2	0	1	0	0	2	0	1	0	0	0	0	0	0	0	0	0	6
39	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
40	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2
All	107	168	171	143	93	94	70	45	33	22	13	17	11	5	6	1	2	1001

We decided to inspect the relationship between the age and the overall to see if there was any meaningful interpretation of the results; however, we found that there wasn't any relationship whatsoever.

In [18]:

```
pd.crosstab(df['Age'], df['Potential'],margins=True)
```

Out[18]:

Potential	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	All
Age																				
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0	1	7
20	0	0	0	0	0	0	0	0	0	5	2	0	2	1	0	0	0	0	0	10
21	0	0	0	0	0	0	1	6	7	8	4	4	3	1	0	2	0	0	0	36
22	0	0	0	0	0	1	3	7	8	6	7	7	3	3	0	2	0	0	0	47
23	0	0	0	1	1	4	11	11	9	6	5	7	2	2	2	1	0	0	0	62

Potential	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	All
Age																				
24	0	0	0	2	6	10	10	11	8	9	6	6	4	2	1	1	0	1	0	77
25	0	1	5	14	14	8	13	12	11	1	4	1	3	2	2	0	1	0	0	92
26	0	5	5	14	18	9	12	6	5	5	6	1	3	2	2	1	1	0	0	95
27	7	12	9	22	7	9	12	4	2	6	1	1	0	3	1	1	1	0	0	98
28	14	6	22	22	6	8	5	5	2	3	2	1	0	1	0	0	0	0	0	97
29	7	12	13	8	10	5	6	6	7	2	2	2	1	1	0	0	0	0	0	82
30	14	14	14	6	6	7	6	6	3	0	2	2	1	0	0	0	0	0	0	81
31	3	8	10	12	7	9	4	4	3	2	4	1	1	0	1	0	0	1	0	70
32	7	6	9	10	5	7	0	3	3	0	0	0	1	2	2	0	0	0	0	55
33	6	3	3	7	5	1	3	0	1	1	0	2	1	0	0	0	0	1	0	34
34	9	3	5	8	1	1	1	3	0	1	0	0	0	0	0	0	0	0	0	32
35	2	1	1	0	1	1	2	0	2	0	0	0	0	0	0	0	0	0	0	10
36	0	0	0	2	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	6
37	2	0	1	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	6
39	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
40	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
All	71	72	97	129	88	84	89	85	72	56	46	37	26	21	12	8	4	3	1	1001

We decided to inspect the relationship between the age and the potential to see if there was any meaningful interpretation of the results. It seems that the younger the player is the more potential he has.

In [ ]:

```
df.dtypes
```

Out[ ]:

```
ID                            int64
Name                         object
Age                           int64
Nationality                  object
Overall                       int64
Potential                     int64
Club                         object
ValueInMillions             float64
WageInThousands               int64
Special                       int64
PreferredFoot              category
InternationalReputation    category
WeakFoot                   category
SkillMoves                 category
WorkRate                     object
JerseyNumber                  int64
Joined                       object
ContractValidUntil           object
HeightInInches              float64
WeightInPounds                int64
ReleaseClause               float64
Jumping                       int64
Strength                      int64
SprintSpeed                   int64
Agility                       int64
Balance                       int64
```

```
Vision                         int64
Dribbling                      int64
BallControl                    int64
dtype: object
```