Fifa_2017

April 16, 2017

CS2006 Python Practical 2

Dataset gathered from: https://www.kaggle.com/artimous/complete-fifa-2017-player-dataset-global The Dataset is the stats for each player and 50+ attricutes straight out of the video game FIFA 2017.

The first requirment is to refine the data and below the many libraries needed to refine and perform analysis on the data are imported.

```
In [5]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.patches as patch
        # from ipywidgets import *
        %matplotlib inline
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d.axes3d import Axes3D, get_test_data
        from mpl_toolkits.mplot3d.art3d import Poly3DCollection
        from matplotlib import cm
        from operator import itemgetter

        import pandas as pd
        import numpy as np

        from ipywidgets import widgets
        from IPython.display import display
```

Read in the csv file

```
In [7]: low_memory = False
        df = pd.read_csv("../Data/fifa_data.csv")
```

Check the types of each column in the CSV file

```
In [8]: df.dtypes
```

```
Out[8]: Name                  object
        Nationality           object
        National_Position     object
        National_Kit         float64
```

```
Club                  object
Club_Position         object
Club_Kit             float64
Club_Joining          object
Contract_Expiry      float64
Rating                 int64
Height                object
Weight                object
Preffered_Foot        object
Birth_Date            object
Age                    int64
Preffered_Position    object
Work_Rate             object
Weak_foot              int64
Skill_Moves            int64
Ball_Control           int64
Dribbling              int64
Marking                int64
Sliding_Tackle         int64
Standing_Tackle        int64
Aggression             int64
Reactions              int64
Attacking_Position     int64
Interceptions          int64
Vision                 int64
Composure              int64
Crossing               int64
Short_Pass             int64
Long_Pass              int64
Acceleration           int64
Speed                  int64
Stamina                int64
Strength               int64
Balance                int64
Agility                int64
Jumping                int64
Heading                int64
Shot_Power             int64
Finishing              int64
Long_Shots             int64
Curve                  int64
Freekick_Accuracy      int64
Penalties              int64
Volleys                int64
GK_Positioning         int64
GK_Diving              int64
GK_Kicking             int64
GK_Handling            int64
```

```
GK_Reflexes              int64
dtype: object
```

To begin really refining the data, the first thing done is remove any column with an empty value in it (NaN). This significanly reduces the size of the dataset in this particular case because the way data is provded by EA regarding players and their specefic stats.

```
In [9]: refinedData = df.copy()
        refinedData = refinedData.dropna()
        pd.options.display.float_format = '{:,.0f}'.format
```

Below duplicate rows are removed from the dataset.

```
In [10]: refinedData = refinedData.drop_duplicates()
```

Below the length of the new refined dataset is given below. As you can see it is a major pretty drop from over 17,000 to just over 1000.

```
In [11]: len(refinedData)

Out[11]: 1075

In [12]: len(df)

Out[12]: 17588
```

Here arrays are defined as the only possible options for specefic columns and then there are nested for loops to check each value in the dataset matches one of the values in the predefined array. If the value is not in the array then then that row gets marked for removal and at the end of this block of code that there is loop that removes all of the rows that were marked for removal because they had invalid input in them.

```
In [13]: nationalPositions = ["CAM", "CB", "CDM", "CM", "GK", "LAM", "LB", "LCB", "LCM", "LDM",
         clubPositions = ["CAM", "CB", "CDM", "CF", "CM", "GK", "LAM", "LB", "LCB", "LCM", "LDM"
         contractDate = [2017, 2018, 2019, 2020, 2021, 2022, 2023]
         prefferedFoot = ["Left", "Right"]
         array100 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
                     48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
                     90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]

         rowsToRemove = []
         rCounter = 0

         natPos = refinedData['National_Position']
         for natPoCounter in natPos:
             flag = 1
             for nationalPositionCounter in nationalPositions:
                         if nationalPositionCounter == natPoCounter:
                             flag = 0
             if flag == 1:
```

```
        rowsToRemove.append(rCounter)
    rCounter = rCounter + 1

rCounter = 0

clubPos = refinedData['Club_Position']
for clubPosCounter in clubPos:
    flag = 1
    for clubPositionCounter in clubPositions:
                if clubPositionCounter == clubPosCounter:
                    flag = 0
    if flag == 1:
        rowsToRemove.append(rCounter)
    rCounter = rCounter + 1

rCounter = 0

conDate = refinedData['Contract_Expiry']
for contractCounter in conDate:
    flag = 1
    for conDateCounter in contractDate:
                if conDateCounter == contractCounter:
                    flag = 0
    if flag == 1:
        rowsToRemove.append(rCounter)
    rCounter = rCounter + 1

rCounter = 0

prefFoot = refinedData['Preffered_Foot']
for prefFootCounter in prefFoot:
    flag = 1
    for footCounter in prefferedFoot:
                if footCounter == prefFootCounter:
                    flag = 0
    if flag == 1:
        rowsToRemove.append(rCounter)
    rCounter = rCounter + 1

rCounter = 0

# no need to check each indivudal rating because if there was an error it would make th
# and thus get removed by this check instead - no need to right 36 more for loops
overAllRating = refinedData['Preffered_Foot']
for rateCounter in overAllRating:
    flag = 1
    for rateCount in array100:
                if rateCount == rateCounter:
```

```
                    flag = 0
        if flag == 1:
            rowsToRemove.append(rCounter)
        rCounter = rCounter + 1

    # rCounter = 0

    # rowsToRemove.sort(reverse=True)
    # for i in rowsToRemove:
    #     refinedData = refinedData.drop(refinedData.index[[i]])

    refinedData.to_csv("refinedData.csv")
```

Here the national position data is shown with how many of each position is in the dataset. The same goes for the next few code blocks for Nationality, Club, Club Position, Contract Expiry, Rating, Height, Weight, Preffered Foot, Age, Work Rate, and Preffered Position. This is some of the data that goes on to be plotted using matplotlib.

```
In [14]: national_position = refinedData.groupby("National_Position")
         national_position.size()
```

```
Out[14]: National_Position
         CAM     19
         CB       9
         CDM      9
         CM       9
         GK      47
         LAM      4
         LB      39
         LCB     48
         LCM     25
         LDM     19
         LF       3
         LM      32
         LS      18
         LW       7
         LWB      4
         RAM      4
         RB      38
         RCB     46
         RCM     25
         RDM     18
         RF       3
         RM      34
         RS      18
         RW       7
         RWB      4
         ST      30
```

```
Sub    556
dtype: int64
```

```
In [15]: nationality = refinedData.groupby("Nationality")
         nationality.size()
```

```
Out[15]: Nationality
         Argentina              23
         Australia              23
         Austria                23
         Belgium                23
         Bolivia                22
         Brazil                 23
         Bulgaria               23
         Cameroon               23
         Canada                 23
         Chile                  22
         China PR               23
         Colombia               24
         Czech Republic         23
         Denmark                23
         Ecuador                22
         Egypt                  23
         England                23
         Finland                20
         France                 23
         Germany                23
         Greece                 24
         Hungary                23
         India                  24
         Italy                  23
         Ivory Coast            23
         Mexico                 23
         Netherlands            23
         Northern Ireland       23
         Norway                 23
         Paraguay               24
         Peru                   23
         Poland                 23
         Portugal               23
         Republic of Ireland    23
         Romania                21
         Russia                 23
         Scotland               23
         Slovenia               23
         South Africa           24
         Spain                  23
         Sweden                 23
```

```
Switzerland            23
Turkey                 23
United States          23
Uruguay                21
Venezuela              22
Wales                  23
dtype: int64
```

```
In [16]: club = refinedData.groupby("Club")
         club.size()
```

```
Out[16]: Club
         1. FC Köln              1
         1. FC Nürnberg          2
         1. FSV Mainz 05         2
         1860 München            1
         1899 Hoffenheim         4
         AC Ajaccio              1
         ADO Den Haag            1
         AIK                     3
         AS Monaco               8
         AS Nancy                1
         AS Saint-Étienne        2
         AZ                      1
         Aalborg BK              1
         Aarhus GF               1
         Aberdeen                1
         Adelaide United         1
         Ajax                    6
         Akhisarspor             1
         Al Ahli                 1
         Al Ittihad              1
         Al-Ettifaq              1
         América                 2
         Angers SCO              2
         Antalyaspor             1
         Arouca                  1
         Arsenal                11
         Arsenal Tula            1
         Aston Villa             5
         Atalanta                3
         Atl. Nacional           3
                                ..
         Terek Grozny            4
         Tigres                  7
         Toluca                  2
         Tondela                 1
         Torino                  4
```

```
Toronto FC              3
Toulouse FC             4
Trabzonspor             4
U.N.A.M.                3
Udinese                 4
Uni. Católica           2
Uni. de Chile           3
V. Guimarães            1
Valencia CF             4
Veracruz                2
VfB Stuttgart           2
VfL Wolfsburg           5
Viborg FF               1
Villarreal CF           3
Walsall                 1
Watford                 4
Werder Bremen           5
West Brom               8
West Ham                6
Whitecaps FC            2
Wigan Athletic          1
Wisa Kraków             1
Wolves                  1
Yokohama F. Marinos     1
Zenit                   3
dtype: int64
```

In [17]:
```
club_position = refinedData.groupby("Club_Position")
club_position.size()
```

Out[17]:
```
Club_Position
CAM     31
CB       6
CDM     12
CF       1
CM       7
GK      80
LAM      3
LB      38
LCB     52
LCM     31
LDM     14
LF       2
LM      36
LS      12
LW      16
LWB      5
RAM      3
```

```
RB      44
RCB     49
RCM     23
RDM     16
RF       2
RM      27
RS      15
RW      18
RWB      4
Res    231
ST      41
Sub    256
dtype: int64
```

```
In [18]: contract_expiry = refinedData.groupby("Contract_Expiry")
         contract_expiry.size()
```

```
Out[18]: Contract_Expiry
         2,017    101
         2,018    224
         2,019    240
         2,020    242
         2,021    145
         2,022     59
         2,023     64
         dtype: int64
```

```
In [19]: overall_rating = refinedData.groupby("Rating")
         overall_rating.size()
```

```
Out[19]: Rating
         52     2
         53     2
         54     2
         55     3
         56     1
         57     1
         58     8
         59    10
         60    11
         61     8
         62    11
         63    18
         64    14
         65    23
         66    25
         67    37
         68    28
         69    41
```

```
70    47
71    56
72    57
73    71
74    71
75    60
76    65
77    57
78    50
79    47
80    45
81    37
82    38
83    49
84    17
85    13
86    16
87     7
88    11
89     8
90     3
92     3
93     1
94     1
dtype: int64
```

```
In [20]: height = refinedData.groupby("Height")
         height.size()
```

```
Out[20]: Height
         160 cm     1
         162 cm     1
         163 cm     1
         164 cm     2
         165 cm     4
         166 cm     2
         167 cm     3
         168 cm    12
         169 cm    13
         170 cm    32
         171 cm    16
         172 cm    16
         173 cm    32
         174 cm    29
         175 cm    53
         176 cm    36
         177 cm    24
         178 cm    61
```

```
179 cm    42
180 cm    77
181 cm    31
182 cm    45
183 cm    70
184 cm    56
185 cm    62
186 cm    39
187 cm    48
188 cm    67
189 cm    37
190 cm    47
191 cm    23
192 cm    24
193 cm    29
194 cm     7
195 cm     9
196 cm    14
197 cm     3
198 cm     3
199 cm     2
201 cm     1
203 cm     1
dtype: int64
```

```
In [21]: weight = refinedData.groupby("Weight")
         weight.size()
```

```
Out[21]: Weight
         55 kg     1
         58 kg     2
         59 kg     3
         60 kg     6
         61 kg     4
         62 kg     7
         63 kg     5
         64 kg    12
         65 kg    11
         66 kg    20
         67 kg    27
         68 kg    38
         69 kg    21
         70 kg    80
         71 kg    40
         72 kg    46
         73 kg    46
         74 kg    61
         75 kg    71
```

```
76 kg    56
77 kg    47
78 kg    58
79 kg    42
80 kg    44
81 kg    32
82 kg    44
83 kg    34
84 kg    35
85 kg    44
86 kg    30
87 kg    17
88 kg    23
89 kg    10
90 kg    15
91 kg    15
92 kg    11
93 kg     6
94 kg     1
95 kg     5
96 kg     2
97 kg     2
98 kg     1
dtype: int64
```

```
In [22]: foot = refinedData.groupby("Preffered_Foot")
         foot.size()
```

```
Out[22]: Preffered_Foot
         Left     235
         Right    840
         dtype: int64
```

```
In [23]: age = refinedData.groupby("Age")
         age.size()
```

```
Out[23]: Age
         17      1
         18      4
         19     17
         20     16
         21     32
         22     66
         23     77
         24     97
         25    103
         26     91
         27    111
         28     94
```

```
29     92
30     87
31     61
32     51
33     30
34     14
35     11
36      3
37     10
38      3
39      3
44      1
dtype: int64
```

```
In [24]: work_rate = refinedData.groupby("Work_Rate")
         work_rate.size()
```

```
Out[24]: Work_Rate
         High / High        121
         High / Low          52
         High / Medium      237
         Low / High          25
         Low / Low            1
         Low / Medium        21
         Medium / High      125
         Medium / Low        41
         Medium / Medium    452
         dtype: int64
```

```
In [25]: pref_position = refinedData.groupby("Preffered_Position")
         pref_position.size()
```

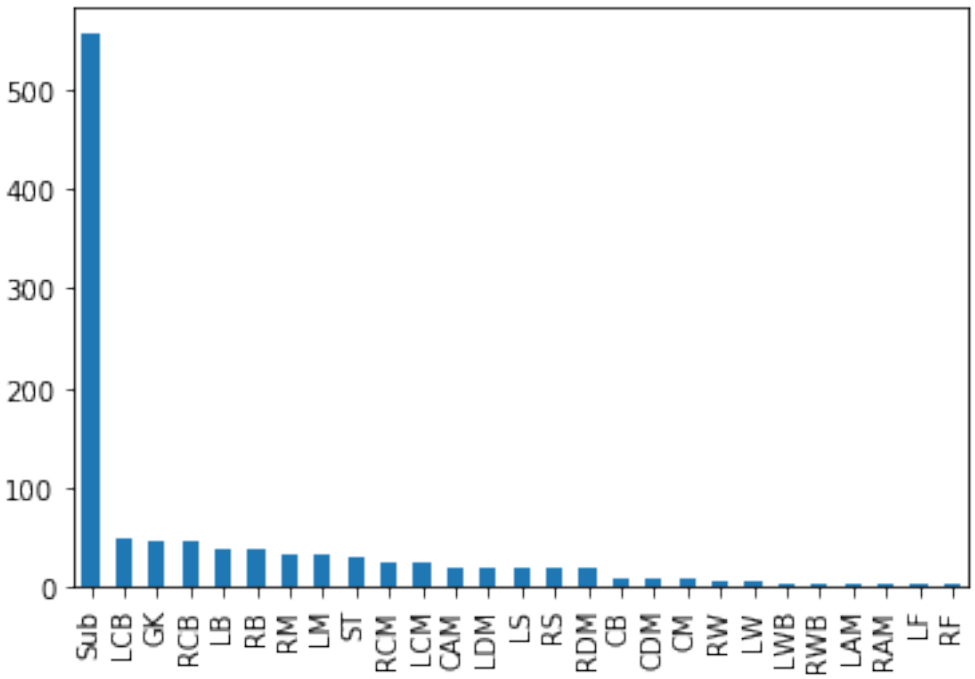
```
Out[25]: Preffered_Position
         CAM           12
         CAM/CF         4
         CAM/CM        12
         CAM/LM        12
         CAM/LM/CM      1
         CAM/LM/RM      2
         CAM/LW         4
         CAM/RM         7
         CAM/RM/LM      1
         CAM/RW         5
         CAM/ST         4
         CB           145
         CB/CDM        13
         CB/CM          2
         CB/LB         15
         CB/RB         18
```

```
CDM            27
CDM/CAM         3
CDM/CB          7
CDM/CM         46
CDM/LM          1
CDM/RB          3
CDM/RM          1
CF/CAM          1
CF/CAM/ST       1
CF/RM           1
CM             19
CM/CAM         16
CM/CB           1
CM/CDM         53
              ...
RM/CAM         10
RM/CF           1
RM/CM           2
RM/CM/CDM       1
RM/LM          18
RM/LW           1
RM/LW/LM        1
RM/RB           1
RM/RW           4
RM/RWB          1
RM/ST           6
RW              5
RW/CAM          3
RW/CF           1
RW/CM/LW        1
RW/LM           1
RW/LW           6
RW/RB           2
RW/RM           2
RW/ST           2
RWB/RM          1
ST            100
ST/CAM          8
ST/CF           4
ST/LM          16
ST/LW           9
ST/RM          11
ST/RM/RW        1
ST/RW           4
ST/RW/RM        1
dtype: int64
```

Below is a bar graph of the National Position data and as you can see the majority of the players

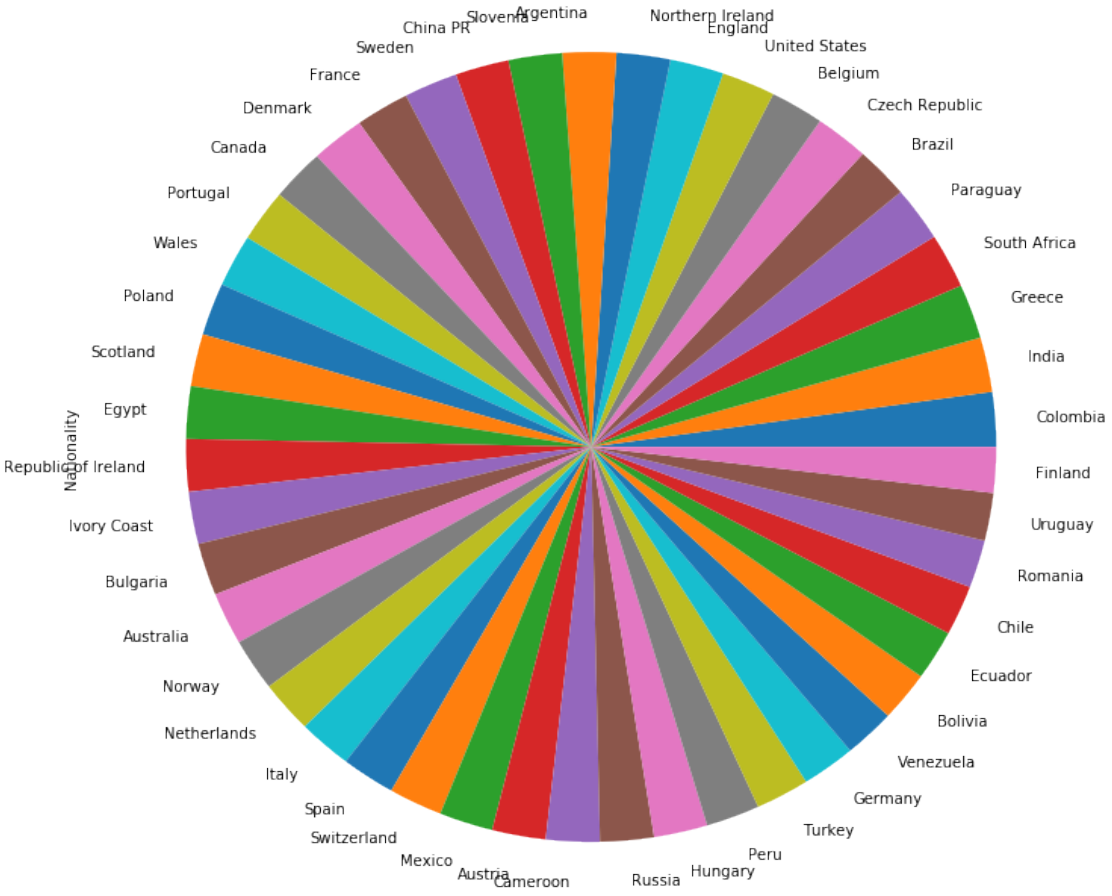
in the FIFA 2017 game are subs for their respective national teams.

```
In [26]: refinedData['National_Position'].value_counts().plot(kind="bar")
         plt.show()
```



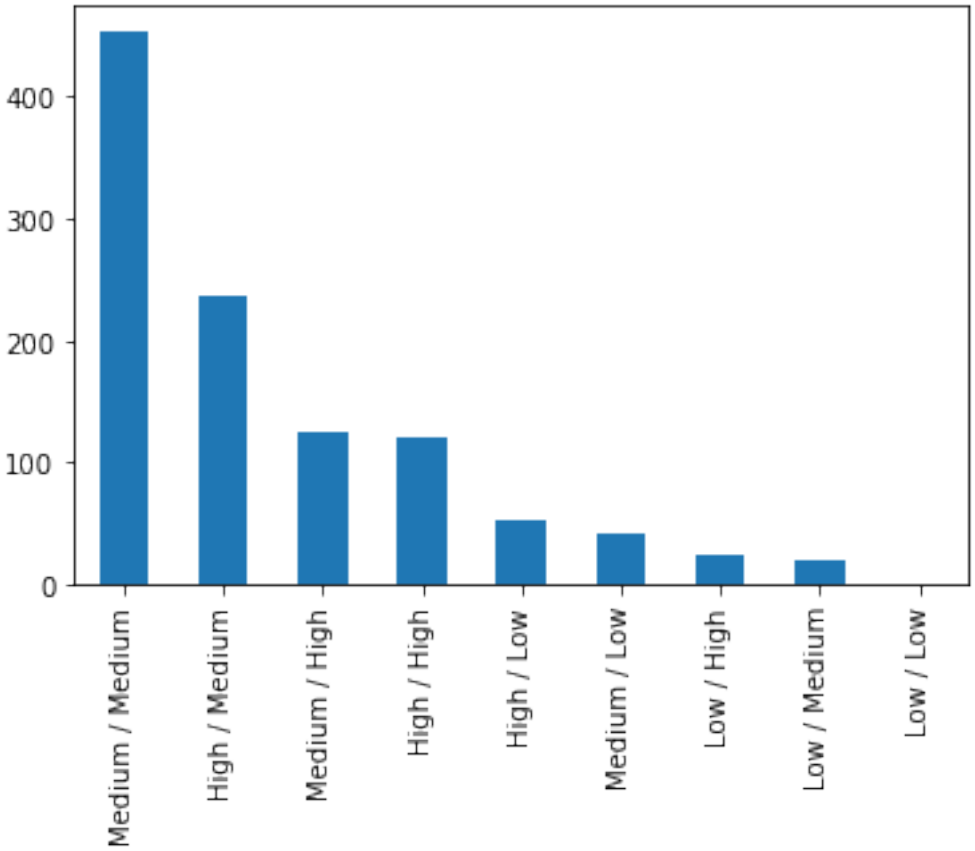
Below is a pie chart of the nationalities represented in the game and as you can see it is a pretty even distrbution of players from each country.

```
In [27]: refinedData['Nationality'].value_counts().plot(kind="pie", figsize = (12,12))
         plt.show()
```



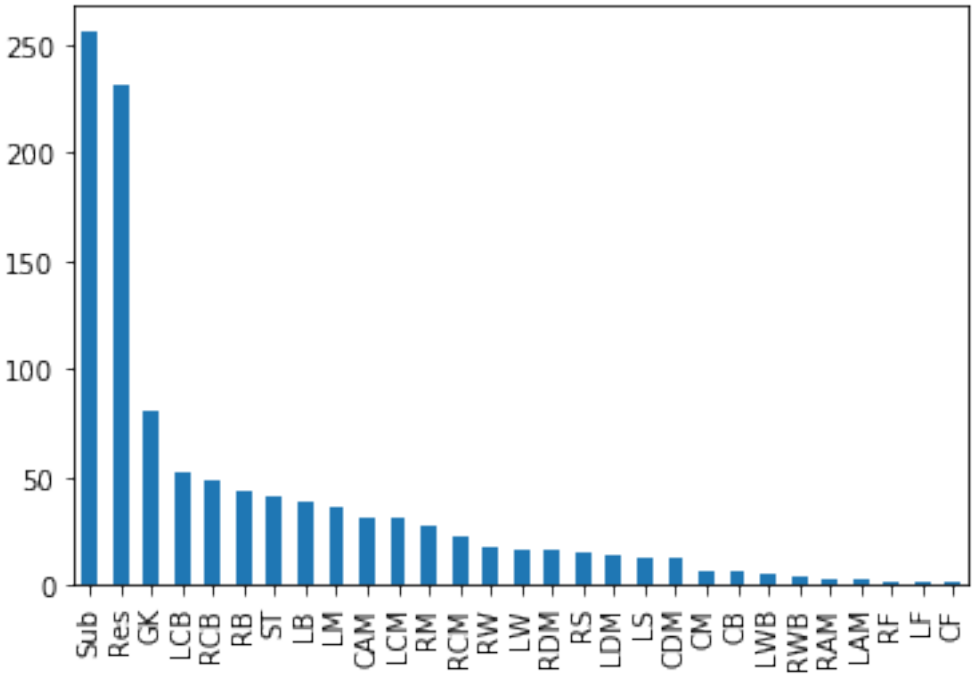
Below is a bar char showing the distribution of work rates among players. The work rate stat in FIFA 17 describes a players leaning towards offense of defense and as such high/medium means high offense skills/medium deffense skills.

```
In [28]: refinedData['Work_Rate'].value_counts().plot(kind="bar")
         plt.show()
```



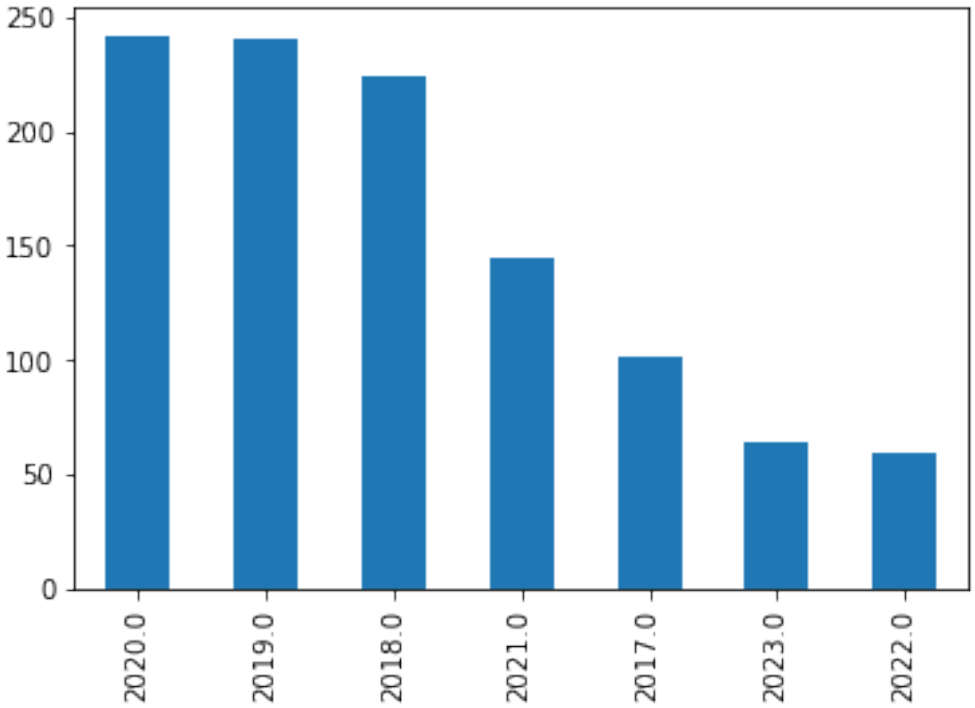
Below is the distribution of club positions among players and as is simmilar to the national position data the majority of players in FIFA are substitutes.

```
In [29]: refinedData['Club_Position'].value_counts().plot(kind="bar")
         plt.show()
```



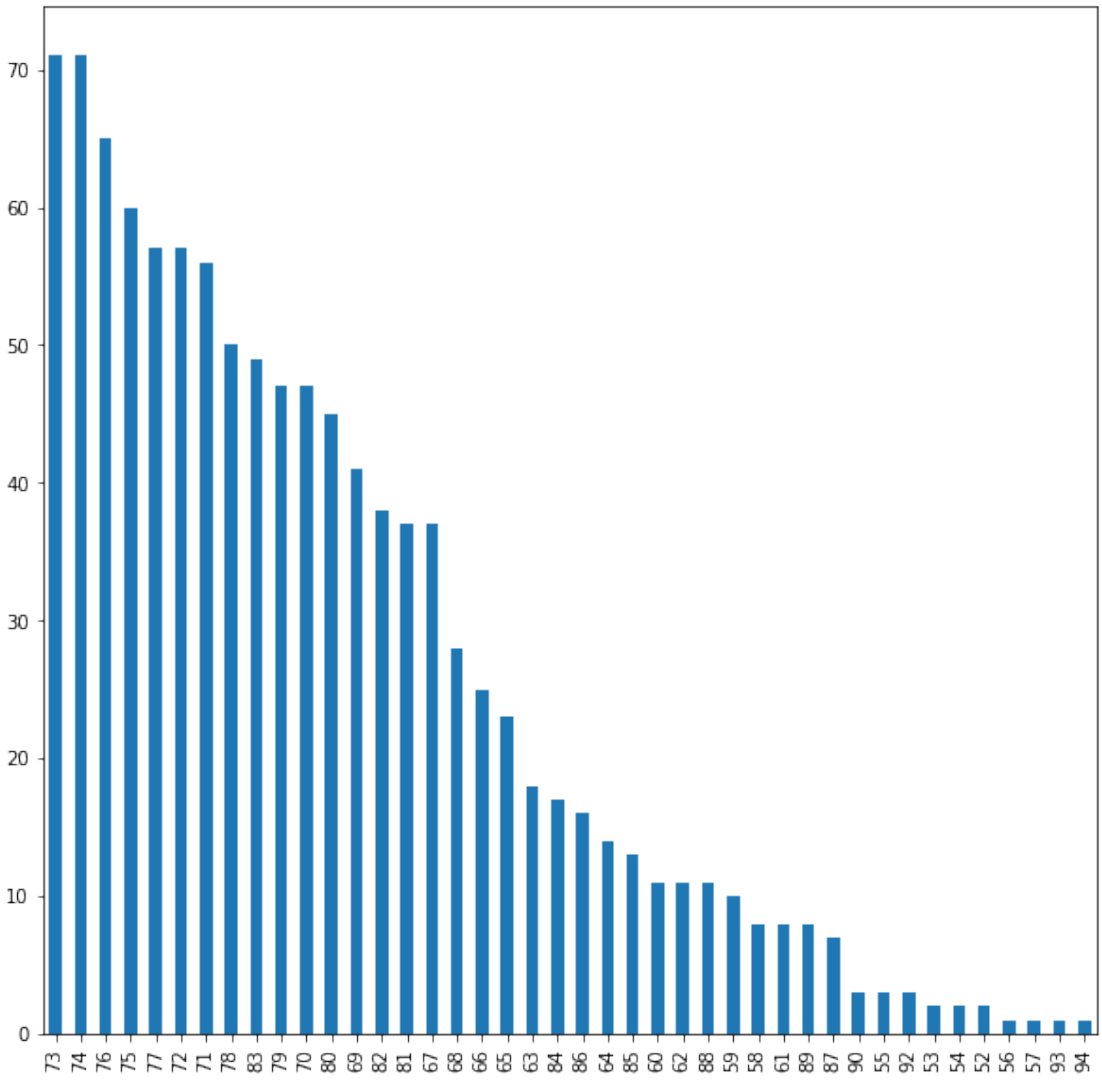
Below is the data that shows how many players have their contract epxiery date in each each in the future and with the players in FIFA17 the majoirty of them have contracts that expire in 2020 but almost the same amount have contracts that expire in 2019 or 2018.

```
In [30]: refinedData['Contract_Expiry'].value_counts().plot(kind="bar")
         plt.show()
```



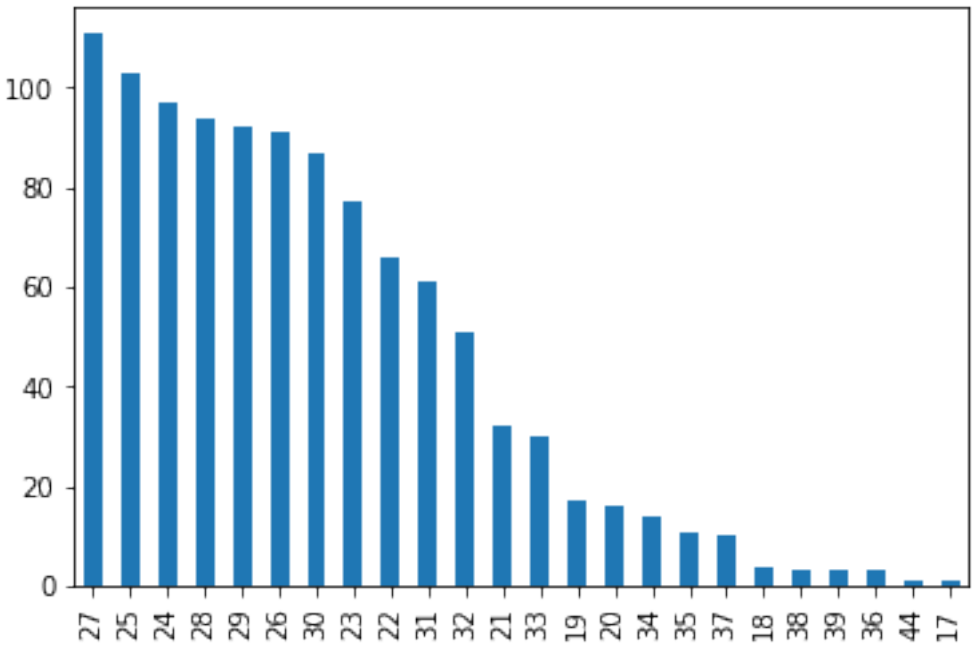
Below is a bar chart showing the distrubiton of rating among the different players. The "Rating" stat in FIFA is based of the over 30 indivudual ratings also in the dataset. Each rating beit the overall Rating or a rating specefic skill is on a scale of 1 - 100 where the higher number means the player is either better overall or better at the specefic skill.

```
In [31]: refinedData['Rating'].value_counts().plot(kind="bar", figsize = (10, 10))
         plt.show()
```



Below is the age distribution and as you can see there are more 27 year olds than any other age but not by a wide margin.

```
In [32]: refinedData['Age'].value_counts().plot(kind="bar")
         plt.show()
```



Below is a pie chart showing the distribution between right dominated foot players and left. There is a clear majority of righties than lefties which is not a total surprise as there are more righties in the world as a whole.

```
In [33]: refinedData['Preffered_Foot'].value_counts().plot(kind="pie")
         plt.show()
```

Below is a table produced by using groupBy which shows how many righties and lefties play at each position when they are playing for their national teams.

```
In [34]: byPositionAndFoot = refinedData[['National_Position', 'Preffered_Foot']].copy()
         byPositionAndFoot = byPositionAndFoot.groupby(['National_Position', 'Preffered_Foot']).
         byPositionAndFoot.reset_index(name='Count')
```

Out[34]:

	National_Position	Preffered_Foot	Count
0	CAM	Left	7
1	CAM	Right	12
2	CB	Left	1
3	CB	Right	8
4	CDM	Left	1
5	CDM	Right	8
6	CM	Left	1
7	CM	Right	8
8	GK	Left	4
9	GK	Right	43
10	LAM	Left	2
11	LAM	Right	2
12	LB	Left	31
13	LB	Right	8
14	LCB	Left	16
15	LCB	Right	32
16	LCM	Left	3
17	LCM	Right	22
18	LDM	Left	4
19	LDM	Right	15
20	LF	Right	3
21	LM	Left	7
22	LM	Right	25
23	LS	Left	2
24	LS	Right	16
25	LW	Left	1
26	LW	Right	6
27	LWB	Left	3
28	LWB	Right	1
29	RAM	Left	1
30	RAM	Right	3
31	RB	Right	38
32	RCB	Left	4
33	RCB	Right	42
34	RCM	Left	3
35	RCM	Right	22
36	RDM	Left	2
37	RDM	Right	16
38	RF	Right	3
39	RM	Left	8

```
40                RM          Right     26
41                RS           Left      3
42                RS          Right     15
43                RW           Left      3
44                RW          Right      4
45               RWB          Right      4
46                ST           Left     11
47                ST          Right     19
48               Sub           Left    117
49               Sub          Right    439
```

Below is a table produced by using groupBy which shows how many righties and lefties play at each position when they are playing for their national teams.

```
In [35]: byClubPositionAndFoot = refinedData[['Club_Position', 'Preffered_Foot']].copy()
         byClubPositionAndFoot = byClubPositionAndFoot.groupby(['Club_Position', 'Preffered_Foot
         byClubPositionAndFoot.reset_index(name='Count')
```

Out[35]:

	Club_Position	Preffered_Foot	Count
0	CAM	Left	8
1	CAM	Right	23
2	CB	Left	1
3	CB	Right	5
4	CDM	Left	3
5	CDM	Right	9
6	CF	Right	1
7	CM	Left	1
8	CM	Right	6
9	GK	Left	9
10	GK	Right	71
11	LAM	Right	3
12	LB	Left	35
13	LB	Right	3
14	LCB	Left	20
15	LCB	Right	32
16	LCM	Left	12
17	LCM	Right	19
18	LDM	Right	14
19	LF	Left	1
20	LF	Right	1
21	LM	Left	13
22	LM	Right	23
23	LS	Left	5
24	LS	Right	7
25	LW	Left	2
26	LW	Right	14
27	LWB	Left	5
28	RAM	Right	3

```
29            RB          Right     44
30           RCB           Left      1
31           RCB          Right     48
32           RCM          Right     23
33           RDM          Right     16
34            RF           Left      2
35            RM           Left      6
36            RM          Right     21
37            RS           Left      2
38            RS          Right     13
39            RW           Left      7
40            RW          Right     11
41           RWB          Right      4
42           Res           Left     43
43           Res          Right    188
44            ST           Left      2
45            ST          Right     39
46           Sub           Left     57
47           Sub          Right    199
```

Below is a table produced using groupBy that shows the correleation between height and overall rating. This produces a lot of table entries because the height stat in the dataSet is an exact figure so even a player is a cm different from another than there is a whole new row in the table. In the future possbily refining the height stat in this dataset to produce the height stat as a few ranges of height might yeild better results in this table.

```
In [36]: # height with overall rating
         byHeightAndRating = refinedData[['Height', 'Rating']].copy()
         byHeightAndRating = byHeightAndRating.groupby(['Height', 'Rating']).size()
         byHeightAndRating.reset_index(name='Count')
```

Out[36]:

	Height	Rating	Count
0	160 cm	64	1
1	162 cm	71	1
2	163 cm	84	1
3	164 cm	65	1
4	164 cm	72	1
5	165 cm	66	1
6	165 cm	68	1
7	165 cm	71	1
8	165 cm	86	1
9	166 cm	61	1
10	166 cm	80	1
11	167 cm	69	1
12	167 cm	70	1
13	167 cm	83	1
14	168 cm	65	1
15	168 cm	71	3

```
16   168 cm      73      2
17   168 cm      74      1
18   168 cm      75      1
19   168 cm      76      1
20   168 cm      78      1
21   168 cm      80      1
22   168 cm      83      1
23   169 cm      63      1
24   169 cm      67      1
25   169 cm      73      2
26   169 cm      74      2
27   169 cm      77      1
28   169 cm      78      1
29   169 cm      79      1
..      ...     ...     ...
524  194 cm      77      1
525  194 cm      78      1
526  194 cm      81      1
527  195 cm      58      1
528  195 cm      67      1
529  195 cm      68      1
530  195 cm      69      1
531  195 cm      74      1
532  195 cm      76      2
533  195 cm      77      1
534  195 cm      81      1
535  196 cm      71      1
536  196 cm      73      1
537  196 cm      74      2
538  196 cm      75      2
539  196 cm      76      1
540  196 cm      77      1
541  196 cm      79      2
542  196 cm      80      1
543  196 cm      81      2
544  196 cm      83      1
545  197 cm      61      1
546  197 cm      66      1
547  197 cm      76      1
548  198 cm      72      1
549  198 cm      75      2
550  199 cm      79      1
551  199 cm      89      1
552  201 cm      78      1
553  203 cm      77      1

[554 rows x 3 columns]
```

Below is a table produced using groupBy that shows the correleation between weight and overall rating. This produces a lot of table entries because the weight stat in the dataSet is an exact figure so even a player is a kg different from another than there is a whole new row in the table. In the future possbily refining the weight stat in this dataset to produce the weight stat as a few ranges of weight like suggested for the height stat might yeild better results in this table.

```
In [37]: # weight with overall rating
         byWeightAndRating = refinedData[['Weight', 'Rating']].copy()
         byWeightAndRating = byWeightAndRating.groupby(['Weight', 'Rating']).size()
         byWeightAndRating.reset_index(name='Count')
```

Out[37]:

	Weight	Rating	Count
0	55 kg	64	1
1	58 kg	62	1
2	58 kg	80	1
3	59 kg	68	1
4	59 kg	73	1
5	59 kg	84	1
6	60 kg	65	1
7	60 kg	71	1
8	60 kg	72	1
9	60 kg	73	1
10	60 kg	74	1
11	60 kg	86	1
12	61 kg	73	1
13	61 kg	78	1
14	61 kg	82	1
15	61 kg	85	1
16	62 kg	65	1
17	62 kg	72	1
18	62 kg	73	1
19	62 kg	77	1
20	62 kg	78	1
21	62 kg	83	1
22	62 kg	88	1
23	63 kg	70	1
24	63 kg	74	1
25	63 kg	75	1
26	63 kg	80	1
27	63 kg	82	1
28	64 kg	63	1
29	64 kg	71	2
..	...	...	...
540	91 kg	83	2
541	91 kg	85	1
542	91 kg	88	1
543	91 kg	89	1
544	92 kg	65	1

```
545  92 kg      67      1
546  92 kg      72      1
547  92 kg      76      1
548  92 kg      78      1
549  92 kg      80      1
550  92 kg      82      1
551  92 kg      83      1
552  92 kg      88      1
553  92 kg      89      1
554  92 kg      92      1
555  93 kg      69      1
556  93 kg      70      1
557  93 kg      75      1
558  93 kg      78      2
559  93 kg      84      1
560  94 kg      85      1
561  95 kg      71      2
562  95 kg      74      1
563  95 kg      75      1
564  95 kg      81      1
565  96 kg      69      1
566  96 kg      77      1
567  97 kg      66      1
568  97 kg      77      1
569  98 kg      79      1

[570 rows x 3 columns]
```

Below is table produced using groupBy that shows the correlation between age and rating and from looking at is looks like the majority of the high ratings reside in the middle of the age range.

```
In [38]: byAgeAndRating = refinedData[['Age', 'Rating']].copy()
         byAgeAndRating = byAgeAndRating.groupby(['Age', 'Rating']).size()
         byAgeAndRating.reset_index(name='Count')
```

Out[38]:

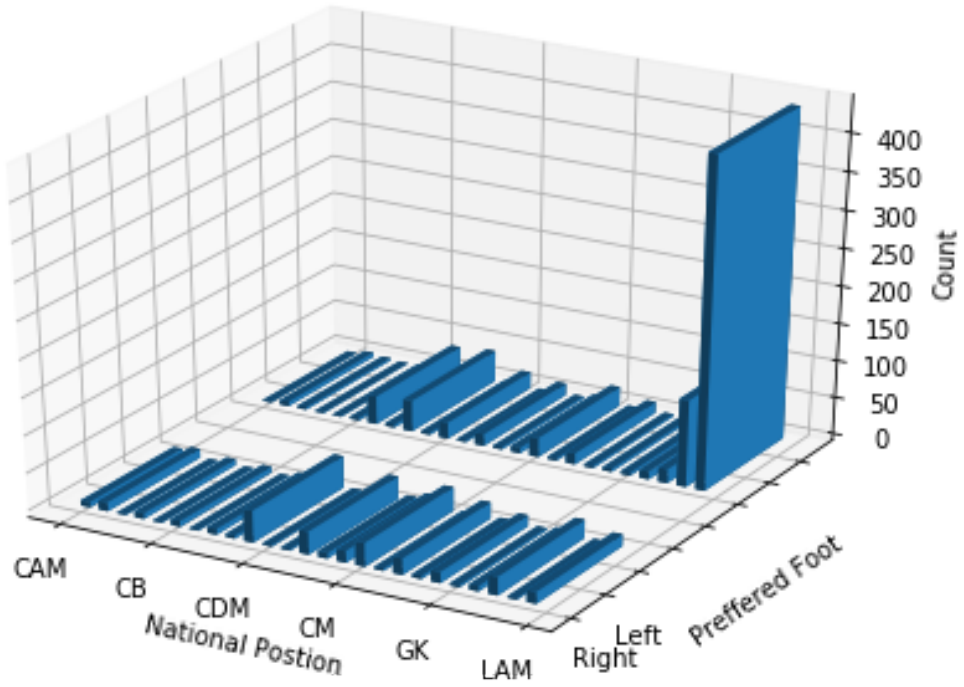
	Age	Rating	Count
0	17	63	1
1	18	62	1
2	18	75	1
3	18	77	1
4	18	79	1
5	19	53	1
6	19	54	1
7	19	62	1
8	19	64	2
9	19	65	1
10	19	67	1
11	19	70	2

12	19	71	1
13	19	72	1
14	19	75	2
15	19	77	1
16	19	78	2
17	19	81	1
18	20	52	1
19	20	59	1
20	20	61	1
21	20	64	1
22	20	67	1
23	20	69	1
24	20	70	2
25	20	71	1
26	20	74	1
27	20	75	1
28	20	76	1
29	20	77	1
..	...	...	...
363	34	77	1
364	34	80	1
365	34	83	1
366	34	88	1
367	35	69	1
368	35	70	1
369	35	71	1
370	35	73	2
371	35	74	3
372	35	75	1
373	35	79	1
374	35	86	1
375	36	69	1
376	36	76	2
377	37	59	1
378	37	69	1
379	37	70	1
380	37	71	1
381	37	72	1
382	37	73	1
383	37	74	1
384	37	75	1
385	37	76	2
386	38	70	1
387	38	73	1
388	38	79	1
389	39	63	1
390	39	77	1
391	39	88	1

```
392   44      74      1

[393 rows x 3 columns]
```

Below is a 3D graph of the table showing the relationship between national position and preffered foot. The x axis is the national position, the y axis is the preffered foot stat, and finally the z axis is the the count when these two variables intersect.

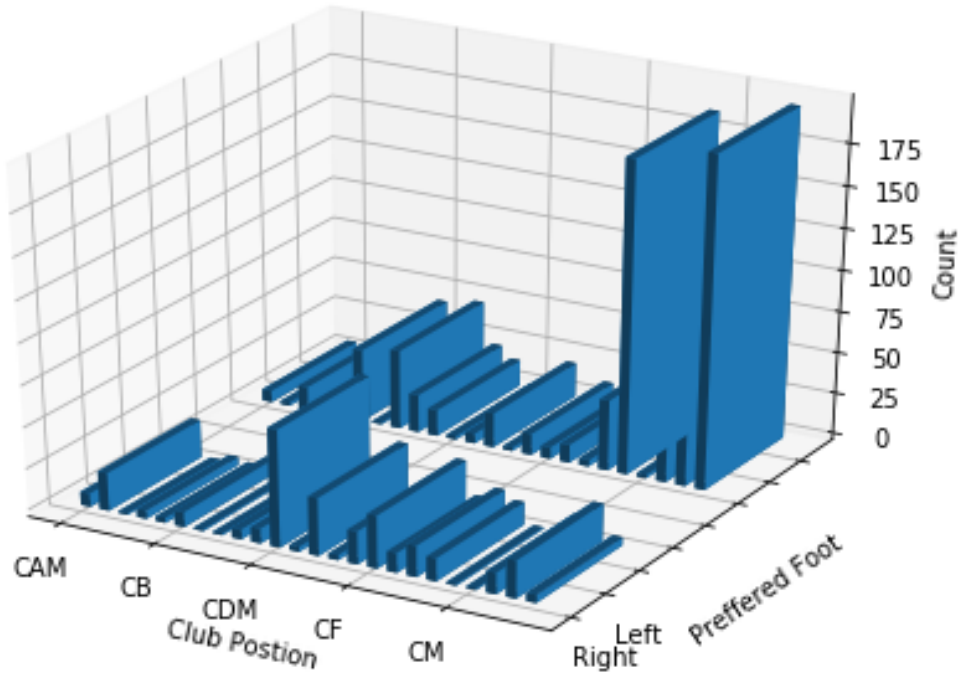
```
In [39]: # national position and preffered foot - 3d graph
         national_position_array = ["CAM", "CB", "CDM", "CM", "GK", "LAM", "LB", "LCB", "LCM", "
         preffered_foot_array = ["Right", "Left"]

         fig = plt.figure()
         ax = fig.add_subplot(111, projection = '3d')
         ax = Axes3D(fig)
         ax.set_xlabel("National Postion")
         ax.set_ylabel("Preffered Foot")
         ax.set_zlabel("Count")

         byPositionAndFoot = refinedData[['National_Position', 'Preffered_Foot']].copy()
         byPositionAndFoot = byPositionAndFoot.groupby(['National_Position', 'Preffered_Foot'])

         z = byPositionAndFoot.size().tolist()
         axes = byPositionAndFoot.groups.keys()
         axes = sorted(axes, key=itemgetter(1))
         axes = sorted(axes, key=itemgetter(0))

         x = list(range(0, len(national_position_array)))
         y = list(range(0, len(preffered_foot_array)))
         X, Y = np.meshgrid(x,y)
         zs = np.array(z)
         Z = zs.reshape(Y.shape)

         values = np.linspace(0.2,1.,X.ravel().shape[0])
         colours = plt.cm.Spectral(values)

         ax.bar3d(X.ravel(), Y.ravel(), Z.ravel()*0, dx=0.5, dy=0.5, dz=Z.ravel())

         ax.set_xticklabels(np.array(national_position_array))
         ax.set_yticklabels(np.array(preffered_foot_array))

         plt.show()
```



Below is a 3D graph showing the relationship between preffered foot and position just like the above chart but in this case club positions are looked at instead of national positions. This was done not only because it is good to compare between national positions and club positions but as well as you can see in the national positions chart, most players are substitutes which is not the case for the club positions. As you can see the distribution of players among different positions and not just the substitue role is much greater in this chart. This can espcially be seen with the center defense midfielder position where there are many more players that play that position for their club team and not their national team.

```
In [40]: # club position and foot
         club_position_array = ['CAM', 'CB', 'CDM', 'CF', 'CM', 'GK', 'LAM', 'LB', 'LCB', 'LCM',
         foot_array = ["Right", "Left"]


         fig = plt.figure()
         ax = fig.add_subplot(111, projection = '3d')
         ax = Axes3D(fig)
         ax.set_xlabel("Club Postion")
         ax.set_ylabel("Preffered Foot")
         ax.set_zlabel("Count")

         byClubPositionAndFoot = refinedData[['Club_Position', 'Preffered_Foot']].copy()
         byClubPositionAndFoot = byClubPositionAndFoot.groupby(['Club_Position', 'Preffered_Foot
```

```
z = byClubPositionAndFoot.size().tolist()
axes = byPositionAndFoot.groups.keys()
axes = sorted(axes, key=itemgetter(1))
axes = sorted(axes, key=itemgetter(0))

x = list(range(0, len(club_position_array)))
y = list(range(0, len(foot_array)))

X, Y = np.meshgrid(x,y)
# len(national_position_array)
zs = np.array(z)
# print(X)
# print(Y)
Z = zs.reshape(Y.shape)

values = np.linspace(0.2,1.,X.ravel().shape[0])
colours = plt.cm.Spectral(values)

ax.bar3d(X.ravel(), Y.ravel(), Z.ravel()*0, dx=0.5, dy=0.5, dz=Z.ravel())

ax.set_xticklabels(np.array(club_position_array))
ax.set_yticklabels(np.array(foot_array))

plt.show()
```



Below is a widget using ipywdigets that allows you to selected a specefic player and see all of there stats and other data that pertains to them like which team they play for and their position.

```
In [58]: def update (Player = list(refinedData['Name'].unique())):
             rating = refinedData[(refinedData['Name'] == Player)]
             display(rating)

         interact(update);
```

```
               Name Nationality National_Position  National_Kit         Club  \
0  Cristiano Ronaldo    Portugal                LS             7  Real Madrid

  Club_Position  Club_Kit Club_Joining  Contract_Expiry  Rating     ...       \
0            LW         7   07/01/2009            2,021      94     ...

  Long_Shots Curve Freekick_Accuracy Penalties  Volleys GK_Positioning  \
0         90    81                76        85       88             14

  GK_Diving  GK_Kicking  GK_Handling  GK_Reflexes
0         7          15           11           11

[1 rows x 53 columns]
```

```
In [ ]:
```