EPL 2020/21 - Project log

%matplotlib inline
—> **Allows me to get my graph/chart right after writing the code**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df = pd.read_csv('/kaggle/input/english-premier-league202021/EPL_20_21.csv')
df.head()
```

	Name	Club	Nationality	Position	Age	Matches	Starts	Mins	Goals	Assists	Passes_Attempted	Pe
0	Mason Mount	Chelsea	ENG	MF,FW	21	36	32	2890	6	5	1881	
1	Edouard Mendy	Chelsea	SEN	GK	28	31	31	2745	0	0	1007	
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	826	
3	Ben Chilwell	Chelsea	ENG	DF	23	27	27	2286	3	5	1806	
4	Reece James	Chelsea	ENG	DF	20	32	25	2373	1	2	1987	

- Shows me all the values in club and the count for each value in club

```
club_count = epl_df['Club'].value_counts()
print(club_count)
```

```
West Bromwich Albion        30
Manchester United           29
Arsenal                     29
Southampton                 29
Everton                     29
Liverpool FC                28
Fulham                      28
Chelsea                     27
Newcastle United            27
Brighton                    27
Wolverhampton Wanderers     27
Sheffield United            27
Leicester City              27
Burnley                     25
Manchester City             24
Crystal Palace              24
Tottenham Hotspur           24
West Ham United             24
Aston Villa                 24
Leeds United                23
Name: Club, dtype: int64
```

- Allows me to look at all the rows in one go

```
pd.set_option('display.max_rows', None)
print(epl_df)
```

```
                        Name                    Club Nationality  \
0                Mason Mount                 Chelsea         ENG
1              Edouard Mendy                 Chelsea         SEN
2                Timo Werner                 Chelsea         GER
3               Ben Chilwell                 Chelsea         ENG
4                Reece James                 Chelsea         ENG
5           César Azpilicueta                 Chelsea         ESP
6                N'Golo Kanté                 Chelsea         FRA
7                    Jorginho                 Chelsea         ITA
8                Thiago Silva                 Chelsea         BRA
9                  Kurt Zouma                 Chelsea         FRA
10              Mateo Kovačić                 Chelsea         CRO
11            Antonio Rüdiger                 Chelsea         GER
12          Christian Pulisic                 Chelsea         USA
13                Kai Havertz                 Chelsea         GER
14         Andreas Christensen                 Chelsea         DEN
15               Hakim Ziyech                 Chelsea         MAR
16              Tammy Abraham                 Chelsea         ENG
17              Marcos Alonso                 Chelsea         ESP
18          Callum Hudson-Odoi                 Chelsea         ENG
19              Olivier Giroud                 Chelsea         FRA
```

```
df.columns
```

```
Index(['Name', 'Club', 'Nationality', 'Position', 'Age', 'Matches', 'Starts',
       'Mins', 'Goals', 'Assists', 'Passes_Attempted', 'Perc_Passes_Completed',
       'Penalty_Goals', 'Penalty_Attempted', 'xG', 'xA', 'Yellow_Cards',
       'Red_Cards'],
      dtype='object')
```

```
df.shape
```

```
(532, 18)
```

```
df.describe()
```

	Age	Matches	Starts	Mins	Goals	Assists	Passes_Attempted	Perc_
count	532.000000	532.000000	532.000000	532.000000	532.000000	532.000000	532.000000	
mean	25.500000	19.535714	15.714286	1411.443609	1.853383	1.287594	717.750000	
std	4.319404	11.840459	11.921161	1043.171856	3.338009	2.095191	631.372522	
min	16.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	
25%	22.000000	9.000000	4.000000	426.000000	0.000000	0.000000	171.500000	
50%	26.000000	21.000000	15.000000	1345.000000	1.000000	0.000000	573.500000	
75%	29.000000	30.000000	27.000000	2303.500000	2.000000	2.000000	1129.500000	
max	38.000000	38.000000	38.000000	3420.000000	23.000000	14.000000	3214.000000	

- I have looked over all the Types and I am satisfied that none of them need changing, so I will not be needing to use

```
#pd.to_numeric(df['example:column_name_goes_here']) <<--if I want to change it to numeric Dtype
epl_df.dtypes
```

```
Name                      object
Club                      object
Nationality               object
Position                  object
Age                        int64
Matches                    int64
Starts                     int64
Mins                       int64
Goals                      int64
Assists                    int64
Passes_Attempted           int64
Perc_Passes_Completed    float64
Penalty_Goals              int64
Penalty_Attempted          int64
xG                       float64
xA                       float64
Yellow_Cards               int64
Red_Cards                  int64
dtype: object
```

- Dot info() is showing me how many entries there are that are NOT NULL and the range I am working with (0 to 531). It tells me the total amount of columns, the column names, how many entries for each

column, what Dtype it is, and it tells us the count for each Dtype (because it is football, mainly dealing with numerical data, most of the dtypes are integers and floats with only 4 being object). Lastly it tells me the Memory used by this data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 532 entries, 0 to 531
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Name                    532 non-null    object 
 1   Club                    532 non-null    object 
 2   Nationality             532 non-null    object 
 3   Position                532 non-null    object 
 4   Age                     532 non-null    int64  
 5   Matches                 532 non-null    int64  
 6   Starts                  532 non-null    int64  
 7   Mins                    532 non-null    int64  
 8   Goals                   532 non-null    int64  
 9   Assists                 532 non-null    int64  
 10  Passes_Attempted        532 non-null    int64  
 11  Perc_Passes_Completed   532 non-null    float64
 12  Penalty_Goals           532 non-null    int64  
 13  Penalty_Attempted       532 non-null    int64  
 14  xG                      532 non-null    float64
 15  xA                      532 non-null    float64
 16  Yellow_Cards            532 non-null    int64  
 17  Red_Cards               532 non-null    int64  
dtypes: float64(3), int64(11), object(4)
memory usage: 74.9+ KB
```

- Below I am checking for null accounts
- No null entries in this data (using dot sum gives me a better view).

```
epl_df.isna()
```

	Name	Club	Nationality	Position	Age	Matches	Starts	Mins	Goals	Assists	Passes_Attempted	Perc
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
527	False	False	False	False	False	False	False	False	False	False	False	
528	False	False	False	False	False	False	False	False	False	False	False	
529	False	False	False	False	False	False	False	False	False	False	False	
530	False	False	False	False	False	False	False	False	False	False	False	
531	False	False	False	False	False	False	False	False	False	False	False	

532 rows × 18 columns

```
epl_df.isna().sum()
```

```
Name                    0
Club                    0
Nationality             0
Position                0
Age                     0
Matches                 0
Starts                  0
Mins                    0
Goals                   0
Assists                 0
Passes_Attempted        0
Perc_Passes_Completed   0
Penalty_Goals           0
Penalty_Attempted       0
xG                      0
xA                      0
Yellow_Cards            0
Red_Cards               0
dtype: int64
```

- Checked for duplicated colums: zero

```
epl_df.duplicated().sum()
```

```
0
```

- I changed some of the column names

```
epl_df = epl_df.rename(columns={'Position': 'Position/s',
                                'Matches': 'Matches_played',
                                'Mins': 'Total_mins_played'})
```

Created new columns

- 2 missing data pieces I wanted were the amount of minutes each player played in all their matches, and the amount of goals scored by each player per match.
- To find out the first, I got the total amount of minutes played and divided it by the number of matches played. To find the latter, I got the total amount of goals scored and divided it by the amount of matches played.

```
epl_df['Mins_per_match'] = (epl_df['Total_mins_played'] / epl_df['Matches_played']).astype(int)
epl_df['Goals_per_match'] = (epl_df['Goals'] / epl_df['Matches_played']).astype(float)
```

```
epl_df.head()
```

_Passes_Completed	Penalty_Goals	Penalty_Attempted	xG	xA	Yellow_Cards	Red_Cards	Mins_per_match	Goals_per_match
82.3	1	1	0.21	0.24	2	0	80	0.166667
84.6	0	0	0.00	0.00	2	0	88	0.000000
77.2	0	0	0.41	0.21	2	0	74	0.171429
78.6	0	0	0.10	0.11	3	0	84	0.111111
85.0	0	0	0.06	0.12	3	0	74	0.031250

Finding out some info:

- Count of every position in the EPL

```
position_count = epl_df['Position/s'].value_counts()
print(position_count)
```

```
DF       178
MF       108
FW        81
FW,MF     47
GK        42
MF,FW     36
DF,MF     15
MF,DF     13
FW,DF      6
DF,FW      6
Name: Position/s, dtype: int64
```

- Total goals scored in the English premier league 2020/21 season: 986

```
total_goals = epl_df['Goals'].sum()
print(total_goals)
```

```
986
```

- Total penalties scored in the English premier league 2020/21 season: 102

```
total_penalties = epl_df['Penalty_Goals'].sum()
print(total_penalties)
```

```
102
```

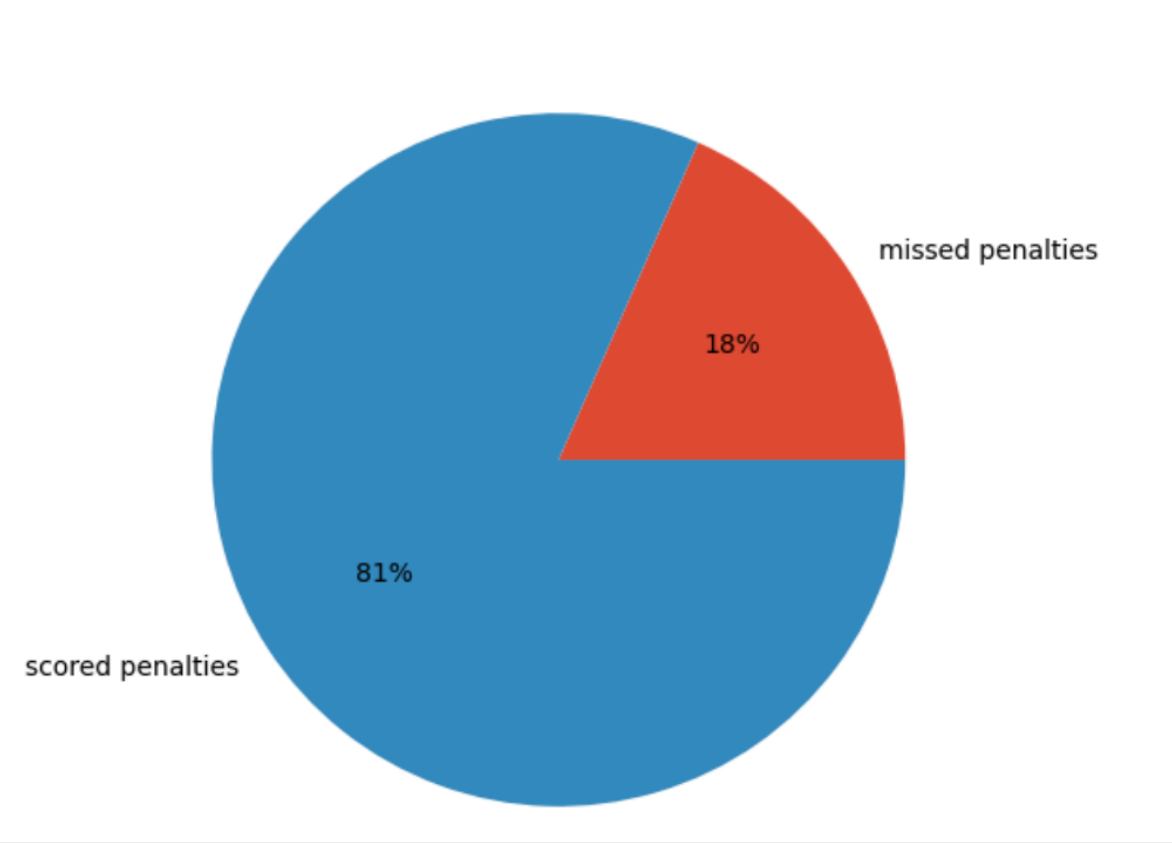
-Total penalties attempts in the English premier league 2020/21 season: 125 (A total of 23 penalties were missed in the 2020/21 eps)

```
penalty_attempts = epl_df['Penalty_Attempted'].sum()
print(penalty_attempts)
```

```
125
```

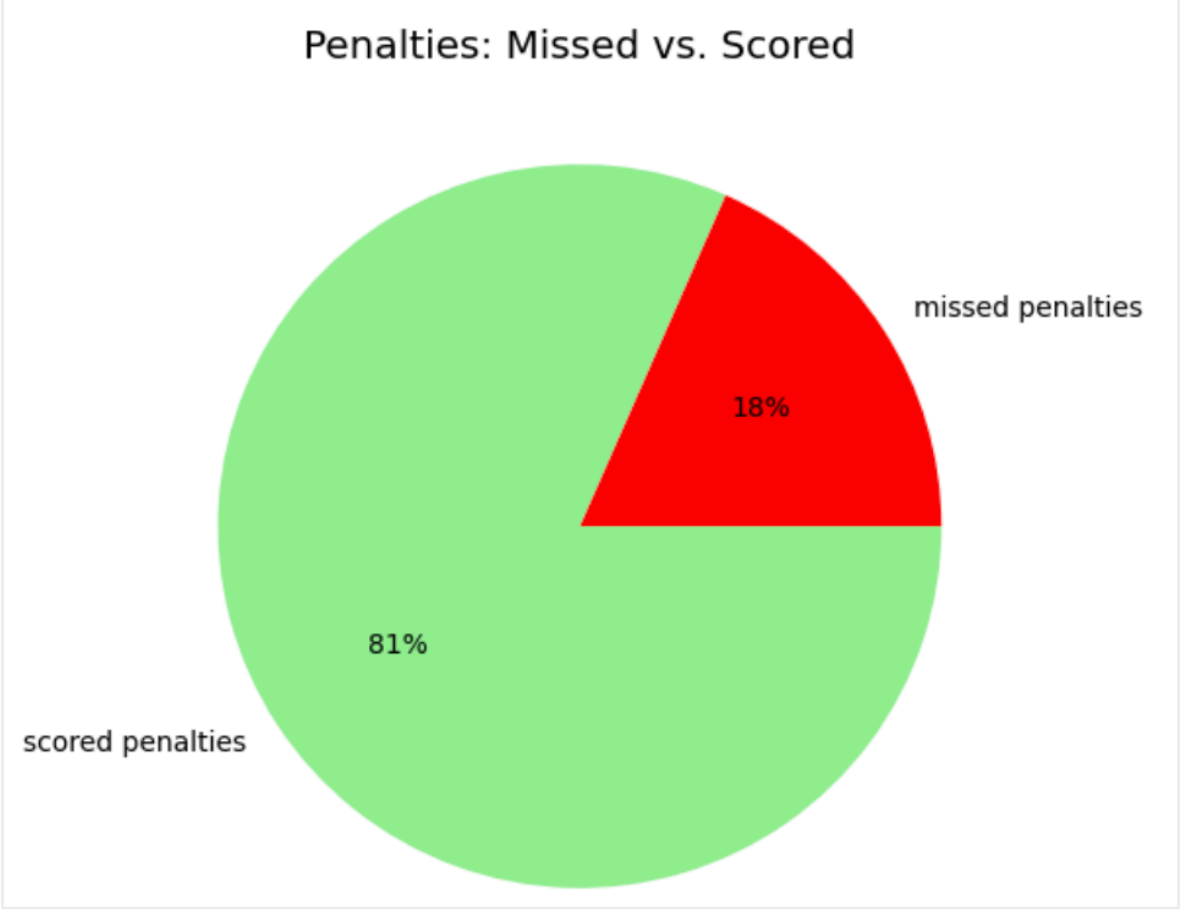
- Creating a pie chart showing the amount of penalties missed vs penalties scored

```
plt.figure(figsize=(13,6))
PL_not_Scored = epl_df['Penalty_Attempted'].sum() - total_penalties
data = [PL_not_Scored, total_penalties]
labels = ['missed penalties', 'scored penalties']
plt.pie(data, labels = labels, autopct = '%d%%')
plt.show()
```



- I Added a title and I played around with the colour - changed the colours of the pie chart segments to match specific connotations that people often associate with the colours (red: fails/missed goals and light green/positives/scored penalties/football)

```
plt.figure(figsize=(13,6))
PL_not_Scored = epl_df['Penalty_Attempted'].sum() - total_penalties
data = [PL_not_Scored, total_penalties]
labels = ['missed penalties', 'scored penalties']
colors = ['red', 'lightgreen']
plt.pie(data, labels = labels,colors = colors, autopct = '%d%%')
plt.title('Penalties: Missed vs. Scored')
plt.show()
```

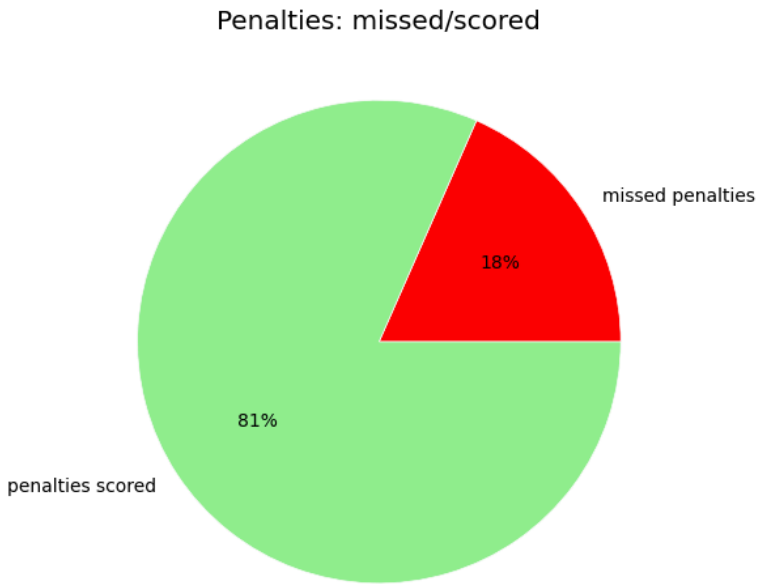


– **I found the total number of unique positions in the EPL**

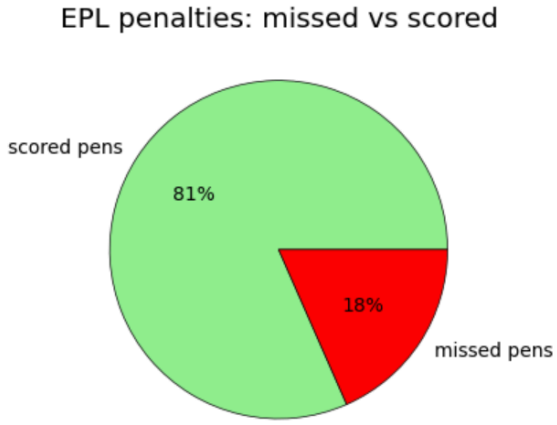
```
epl_df['Position/s'].unique()
```

```
array(['MF,FW', 'GK', 'FW', 'DF', 'MF', 'FW,MF', 'FW,DF', 'DF,MF',
       'MF,DF', 'DF,FW'], dtype=object)
```

```
penalty_total = epl_df['Penalty_Goals'].sum()
penalty_missed = epl_df['Penalty_Attempted'].sum() - penalty_total
data = [penalty_missed, penalty_total]
labels = ['missed penalties', 'penalties scored']
colors = ['red', 'lightgreen']

plt.figure(figsize=(6,6))
plt.pie(data, labels = labels, colors=colors, autopct='%d%%', wedgeprops={'edgecolor': 'white'})
plt.title('Penalties: missed/scored')
plt.show()
```

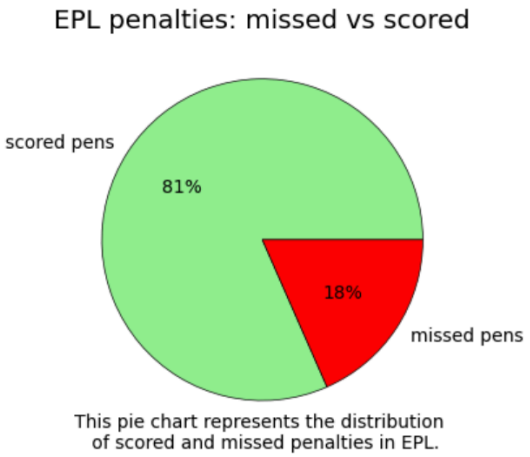



```
total_pens_scored = epl_df['Penalty_Goals'].sum()
total_pens_missed = epl_df['Penalty_Attempted'].sum() - total_pens_scored
data = [total_pens_scored,total_pens_missed]
labels = ['scored pens', 'missed pens']
colors = ['lightgreen', 'red']

plt.figure(figsize=(4,4))
plt.pie(data, labels=labels, colors=colors, autopct='%d%%', wedgeprops={'edgecolor':'black'})
plt.title('EPL penalties: missed vs scored')
description = "This pie chart represents the distribution of scored and missed penalties in EPL."
plt.text(0,-1.3 , description, ha='center', fontsize=10)
plt.show()
```



This pie chart represents the distribution of scored and missed penalties in EPL.

```
total_pens_scored = epl_df['Penalty_Goals'].sum()
total_pens_missed = epl_df['Penalty_Attempted'].sum() - total_pens_scored
data = [total_pens_scored,total_pens_missed]
labels = ['scored pens', 'missed pens']
colors = ['lightgreen', 'red']

plt.figure(figsize=(4,4))
plt.pie(data, labels=labels, colors=colors, autopct='%d%%', wedgeprops={'edgecolor':'black'})
plt.title('EPL penalties: missed vs scored')
description = "This pie chart represents the distribution \n of scored and missed penalties in EPL."
plt.text(0,-1.3 , description, ha='center', fontsize=10)
plt.show()
```




```
#In the code I provided, nunique() is a method in pandas that stands for "number of unique."

total_amount_of_positions = epl_df['Position/s'].nunique()
print('total amount of positions is:', total_amount_of_positions)
```

```
total amount of positions is: 10
```

- I found the number of players in the EPL who play in the FW position (forward)
- There are 81 rows so I know there are 81 players who play as Forwards in the EPL

```
#players that play as FW
epl_df[epl_df['Position/s'] == 'FW']
```

	Name	Club	Nationality	Position/s	Age	Matches_played	Starts	Total_mins_played	Goals	Assists	Passes_Att
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	
16	Tammy Abraham	Chelsea	ENG	FW	22	22	12	1040	6	1	
19	Olivier Giroud	Chelsea	FRA	FW	33	17	8	748	4	0	
23	Ruben Loftus-Cheek	Chelsea	ENG	FW	24	1	1	60	0	0	
30	Raheem Sterling	Manchester City	ENG	FW	25	31	28	2536	10	7	
...	...	...	...	...	...	...	...	...	...	...	
516	Oliver Burke	Sheffield United	SCO	FW	23	25	14	1269	1	1	
518	Oliver McBurnie	Sheffield United	SCO	FW	24	23	12	1324	1	0	
519	Rhian Brewster	Sheffield United	ENG	FW	20	27	12	1128	0	0	
523	Billy Sharp	Sheffield United	ENG	FW	34	16	7	735	3	0	
526	Daniel Jebbison	Sheffield United	ENG	FW	17	4	3	284	1	0	

81 rows × 20 columns

- Showing me the same as the above but in different ways by using different functions etc

```
epl_df['Position/s'] == 'FW'
```

```
0       False
1       False
2        True
3       False
4       False
        ...
527     False
528     False
529     False
530     False
531     False
Name: Position/s, Length: 532, dtype: bool
```

+ Code + Markdown

```
#players that play as FW
epl_df[epl_df['Position/s'] == 'FW'].sum
```

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of                    Name
Club Nationality Position/s  Age  \
2           Timo Werner           Chelsea         GER         FW   24
16        Tammy Abraham           Chelsea         ENG         FW   22
19       Olivier Giroud           Chelsea         FRA         FW   33
23   Ruben Loftus-Cheek           Chelsea         ENG         FW   24
30      Raheem Sterling   Manchester City         ENG         FW   25
..                  ...               ...         ...        ...  ...
516        Oliver Burke  Sheffield United         SCO         FW   23
518     Oliver McBurnie  Sheffield United         SCO         FW   24
519      Rhian Brewster  Sheffield United         ENG         FW   20
523         Billy Sharp  Sheffield United         ENG         FW   34
526     Daniel Jebbison  Sheffield United         ENG         FW   17

     Matches_played  Starts  Total_mins_played  Goals  Assists  \
2                35      29               2602      6        8
16               22      12               1040      6        1
19               17       8                748      4        0
23                1       1                 60      0        0
30               31      28               2536     10        7
..              ...     ...                ...    ...      ...
516              25      14               1269      1        1
518              23      12               1324      1        0
519              27      12               1128      0        0
523              16       7                735      3        0
526               4       3                284      1        0

     Passes_Attempted  Perc_Passes_Completed  Penalty_Goals  \
2                 826                   77.2              0
16                218                   68.3              0
19                217                   74.2              0
23                 16                   68.8              0
30               1127                   85.4              0
..                ...                    ...            ...
516               262                   70.6              0
518               426                   62.9              0
519               225                   69.3              0
523               123                   69.9              2
526                34                   70.6              0

     Penalty_Attempted    xG    xA  Yellow_Cards  Red_Cards  Mins_per_match  \
2                    0  0.41  0.21             2          0              74
16                   0  0.56  0.07             0          0              47
19                   0  0.58  0.09             1          0              44
```

– Unique number of nationalities in the EPL

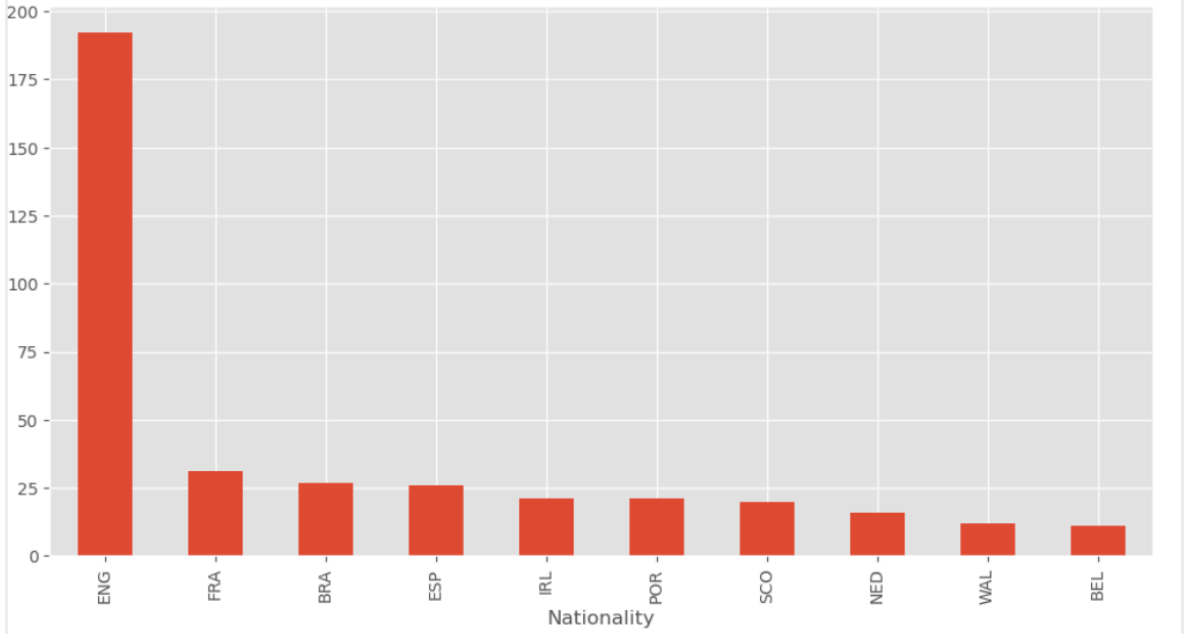
```
#Im going to use the numpy function called
np.size(epl_df['Nationality'].unique())
```

```
59
```

Note: Fig size = Width and height

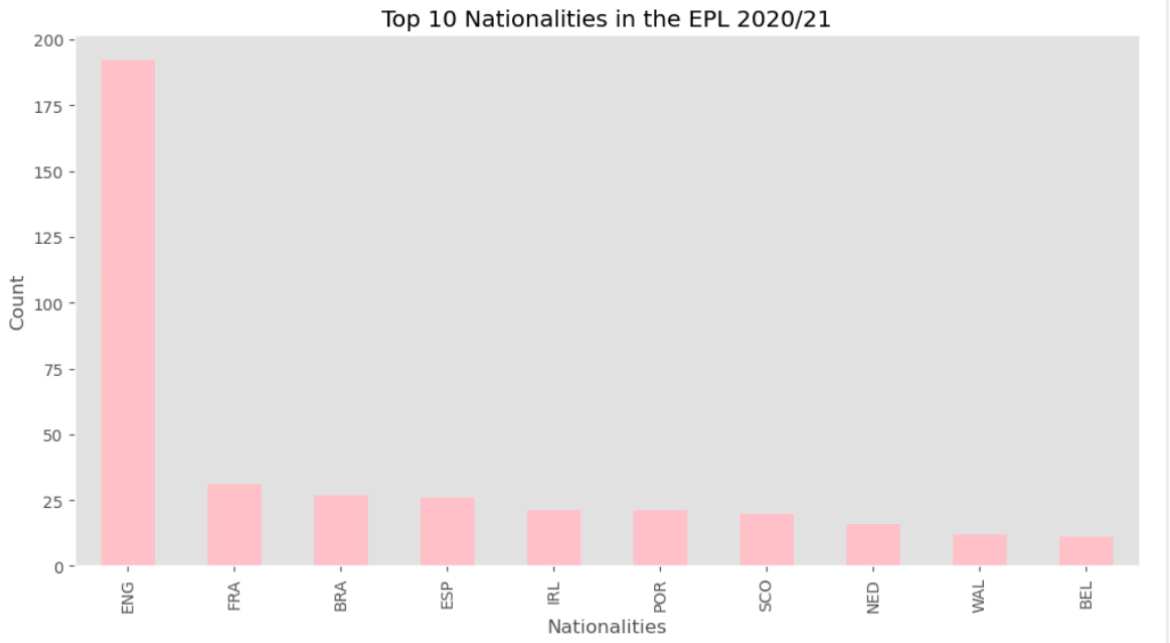
```
nationality = epl_df.groupby('Nationality').size().sort_values(ascending = False)
nationality.head(10).plot(kind='bar', figsize=(12,6))
```

```
<Axes: xlabel='Nationality'>
```



```
nationality = epl_df.groupby('Nationality').size().sort_values(ascending = False)
nationality.head(10).plot(kind='bar',color = 'pink', grid=False, figsize=(12,6))
plt.title('Top 10 Nationalities in the EPL 2020/21')
plt.xlabel('Nationalities')
plt.ylabel('Count')
```

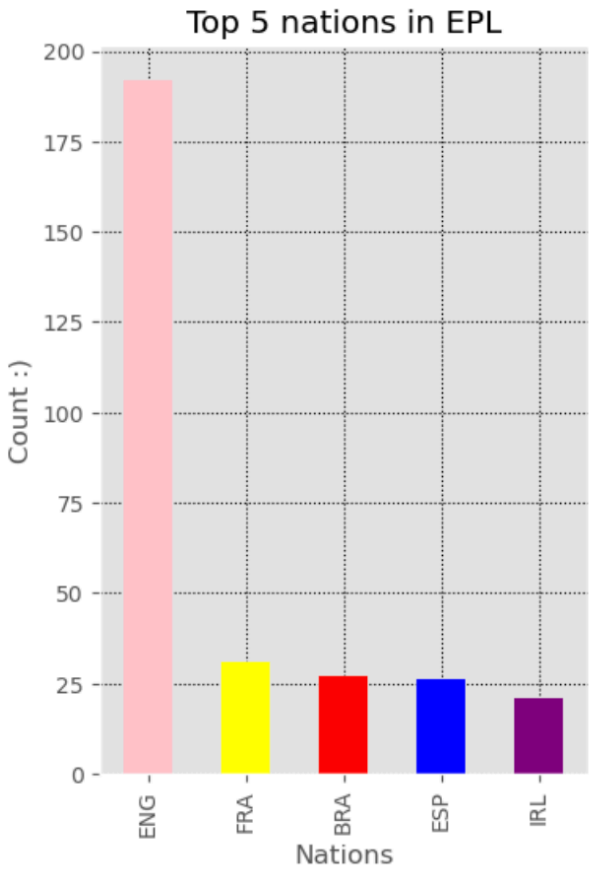
```
Text(0, 0.5, 'Count')
```



- With the below code, I was playing around with the grid using matplotlib.pyplot (plt)
- plt.grid()

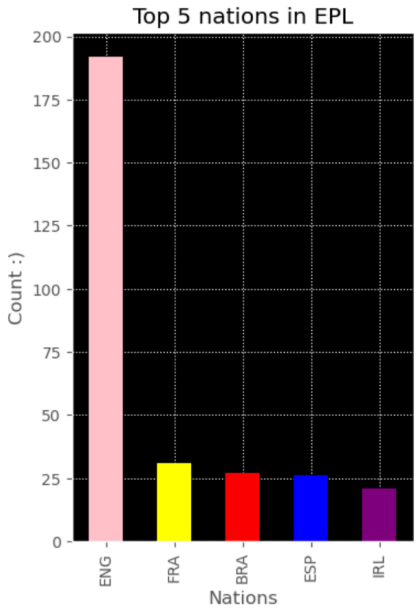
```
nationality = epl_df.groupby('Nationality').size().sort_values(ascending=False)
colors = ['pink', 'yellow', 'red', 'blue', 'purple']
nationality.head(5).plot(kind='bar', grid=False, figsize=(4,6), color=colors)

plt.grid(color='black', linestyle='dotted')
plt.title('Top 5 nations in EPL')
plt.xlabel('Nations')
plt.ylabel('Count :)')
```

71… `Text(0, 0.5, 'Count :)')`



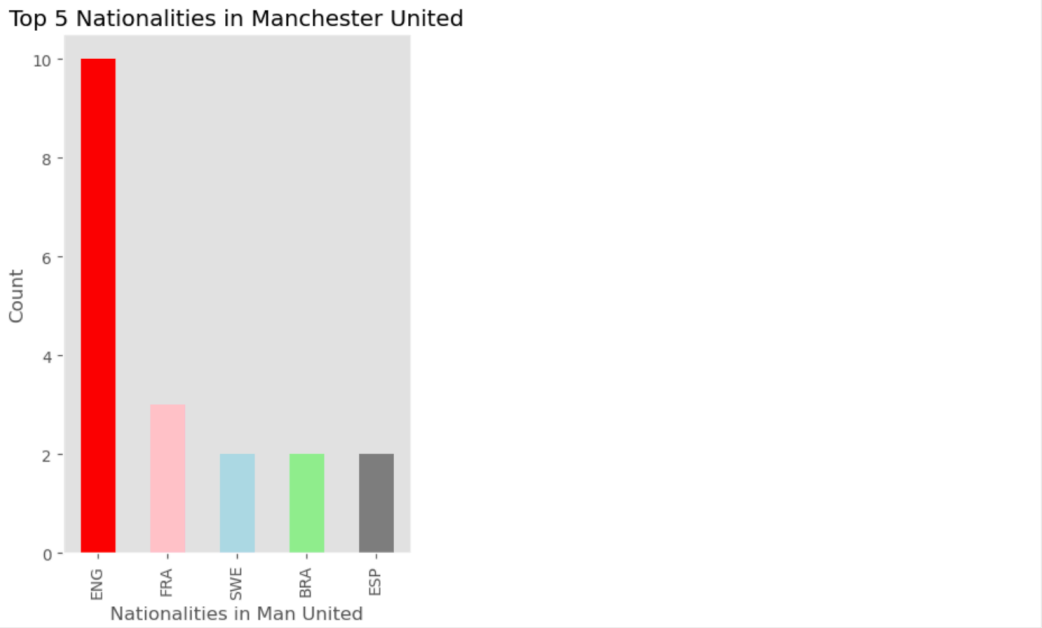
```
nationality = epl_df.groupby('Nationality').size().sort_values(ascending=False)
colors = ['pink', 'yellow', 'red', 'blue', 'purple']
nationality.head(5).plot(kind='bar', grid=False, figsize=(4,6), color=colors)

#plt.gca stands for ' get current axis' and set_facecolor allows us to choose a color for the background
plt.gca().set_facecolor('black')
plt.grid(color='white', linestyle='dotted')
plt.title('Top 5 nations in EPL')
plt.xlabel('Nations')
plt.ylabel('Count :)')
```

```
Text(0, 0.5, 'Count :)')
```

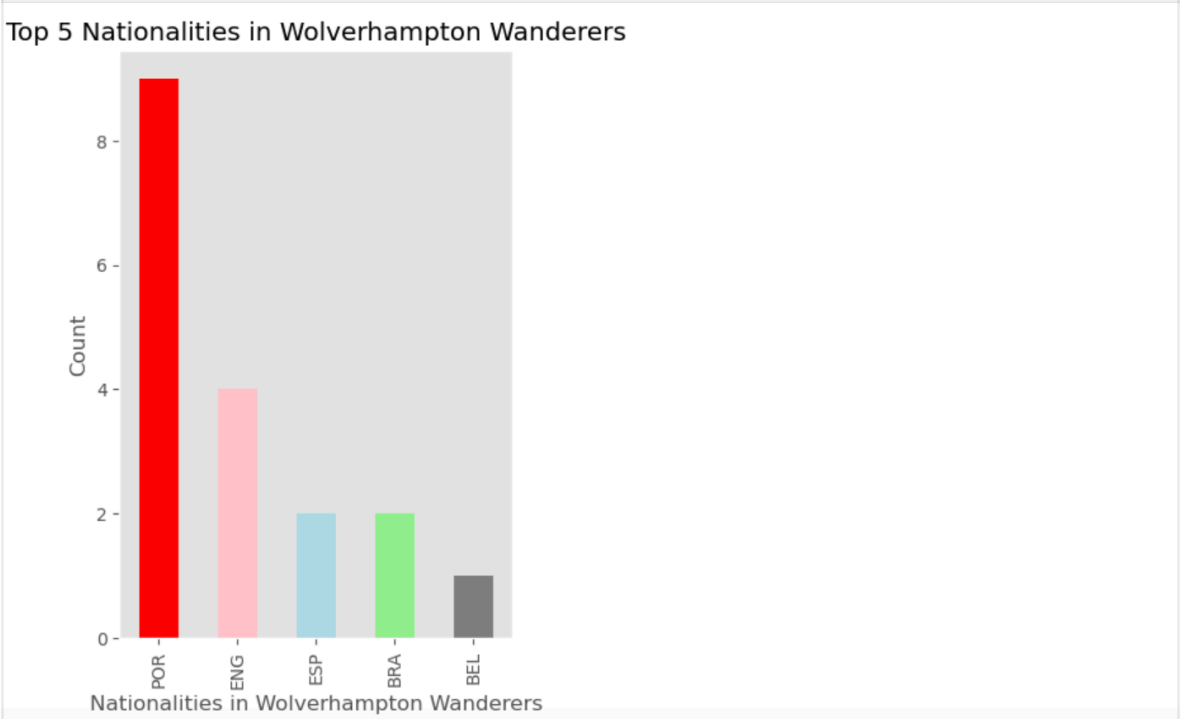


- Did the same as the above but narrowed it down to one team from my dataset
- Top 5 nationalities in Manchester United

```
selected_team = 'Manchester United'
filtered_data = epl_df[epl_df['Club'] == selected_team]
nationality_counts = filtered_data['Nationality'].value_counts()
colors = ['red', 'pink', 'lightblue', 'lightgreen', 'grey']
top_10_nationalities = nationality_counts.head(5).plot(kind='bar', color=colors, grid=False, figsize=(4, 6))

plt.title(f'Top 5 Nationalities in {selected_team}')
plt.xlabel('Nationalities in Man United')
plt.ylabel('Count')

plt.show()
```



```
selected_team = 'Wolverhampton Wanderers'
filtered_data = epl_df[epl_df['Club'] == selected_team]
nationality_counts = filtered_data['Nationality'].value_counts()
colors = ['red', 'pink', 'lightblue', 'lightgreen', 'grey']
top_5_nationalities = nationality_counts.head(5).plot(kind='bar', color=colors, grid=False, figsize=(4, 6))

plt.title(f'Top 5 Nationalities in {selected_team}')
plt.xlabel(f'Nationalities in {selected_team}')
plt.ylabel('Count')

plt.show()
```

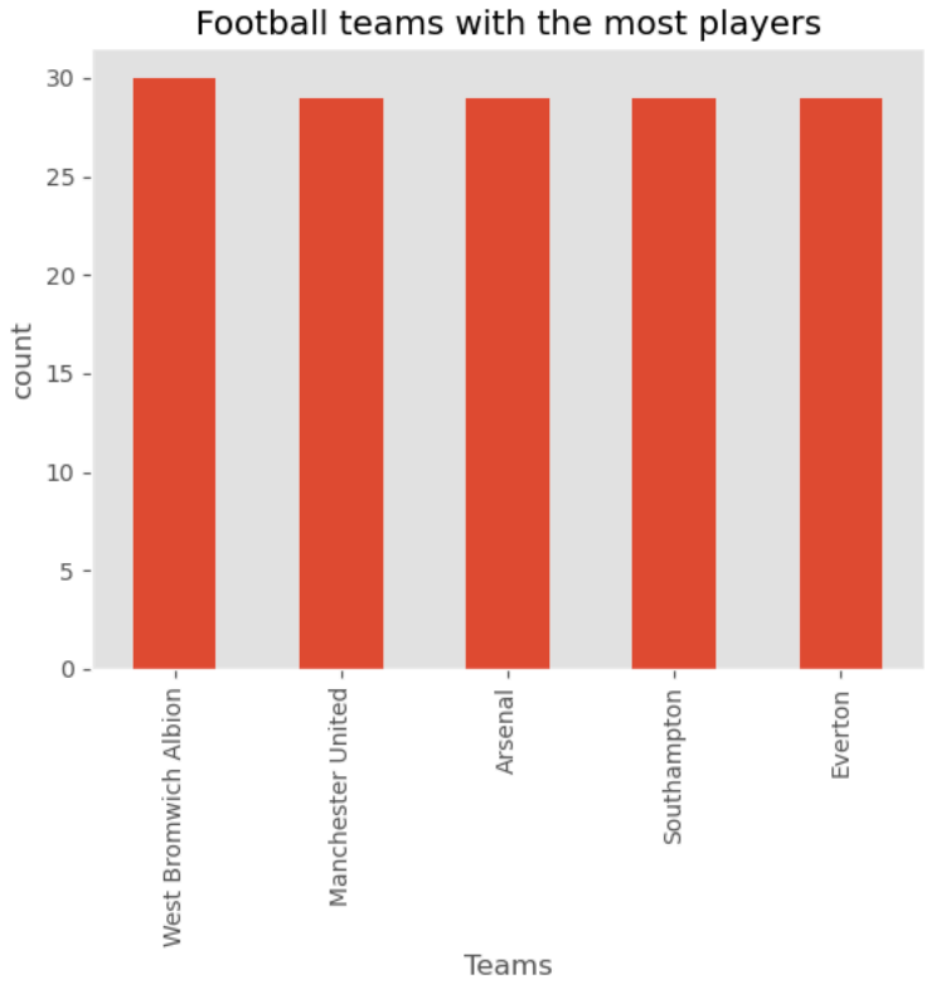


- Looking at teams with the most players. Same result with using ‘nlargest(5)’ and ‘head(5)’
- Next, .nlargest(5) in Code 1 and .head(5) in Code 2 are used to extract the top 5 clubs with the most players. Both methods achieve the same result, providing the top 5 clubs based on player counts.

```
epl_df['Club'].value_counts().head(5).plot(kind='bar', grid=False)
plt.title('Football teams with the most players')
plt.xlabel('Teams')
plt.ylabel('count')
```

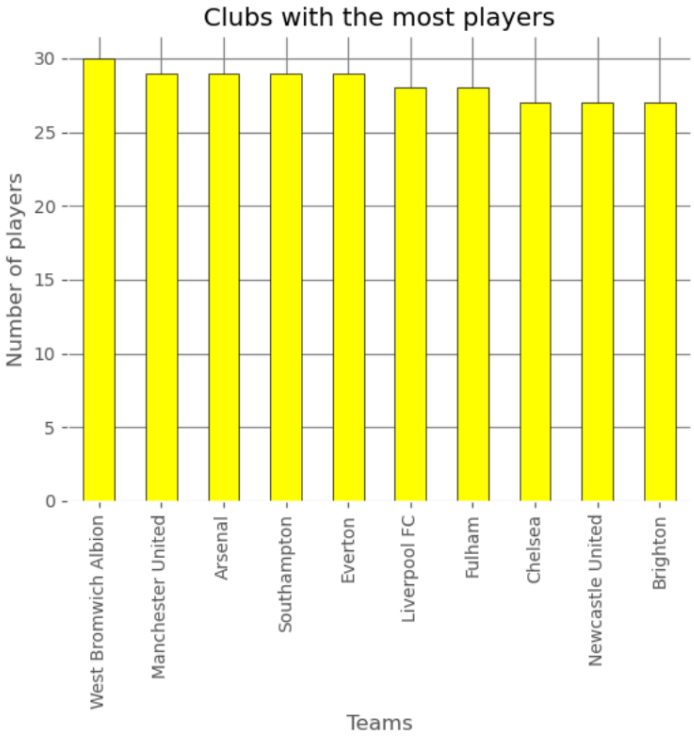
```
#clubs with max players in their squad dsx

# choose the club column from the epl_df and call on the value counts from that
# use the nlargest function to get the top 5 clubs and then plot it.

epl_df['Club'].value_counts().nlargest(5).plot(kind='bar', grid=False)
plt.title('Football teams with the most players')
plt.xlabel('Teams')
plt.ylabel('count')
```

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



```
epl_df['Club'].value_counts().nlargest(10).plot(kind='bar', grid=False, edgecolor='black', color='yellow')
plt.grid(color='grey', linestyle='solid')
plt.gca().set_facecolor(color='white')
plt.title('Clubs with the most players')
plt.xlabel('Teams')
plt.ylabel('Number of players')
```

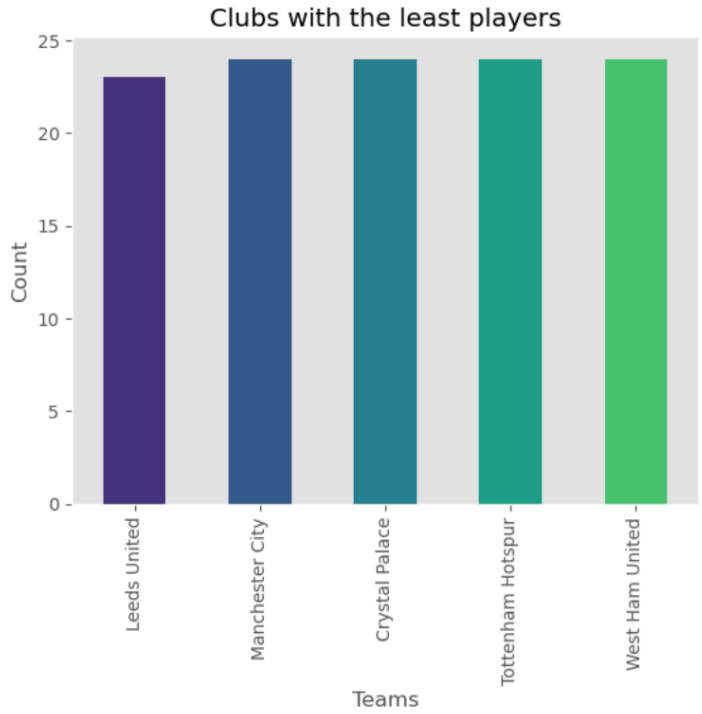
```
Text(0, 0.5, 'Number of players')
```



- The opposite of the above: clubs with the least players (I used the nsmallest function for this)
- I used a color palette from seaborn

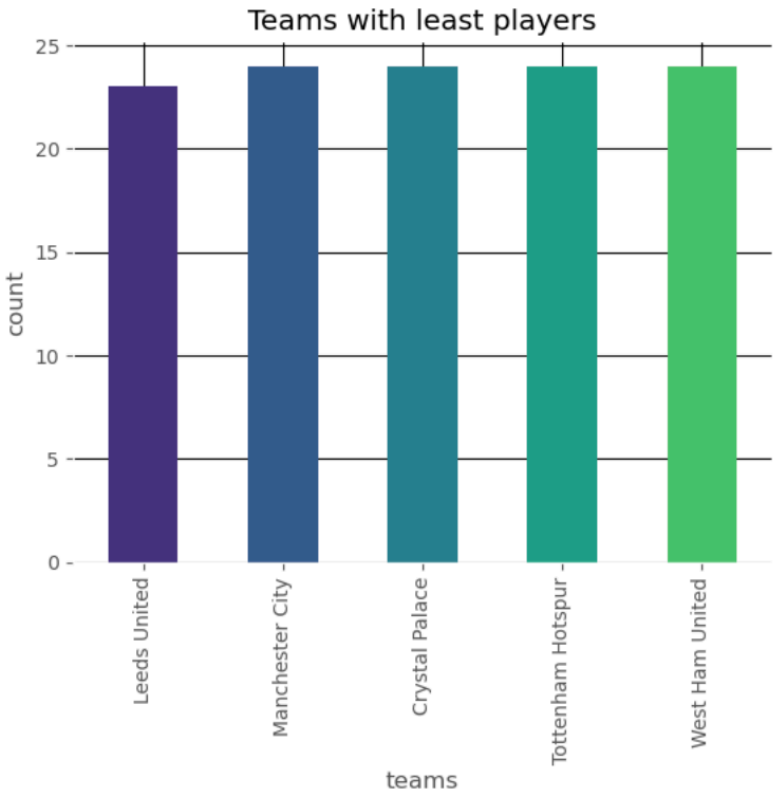
```
#clubs with the least players in their team
epl_df['Club'].value_counts().nsmallest(5).plot(kind='bar', grid=False, color=sns.color_palette("viridis"))
plt.title('Clubs with the least players')
plt.xlabel('Teams')
plt.ylabel('Count')
```

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



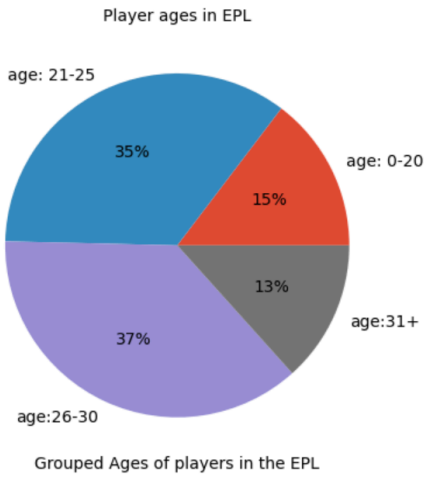
```
epl_df['Club'].value_counts().nsmallest(5).plot(kind='bar', color=sns.color_palette('viridis'))

plt.grid(color='black', linestyle='solid')
plt.gca().set_facecolor(color='white')

plt.title('Teams with least players')
plt.xlabel('teams')
plt.ylabel('count')
```

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



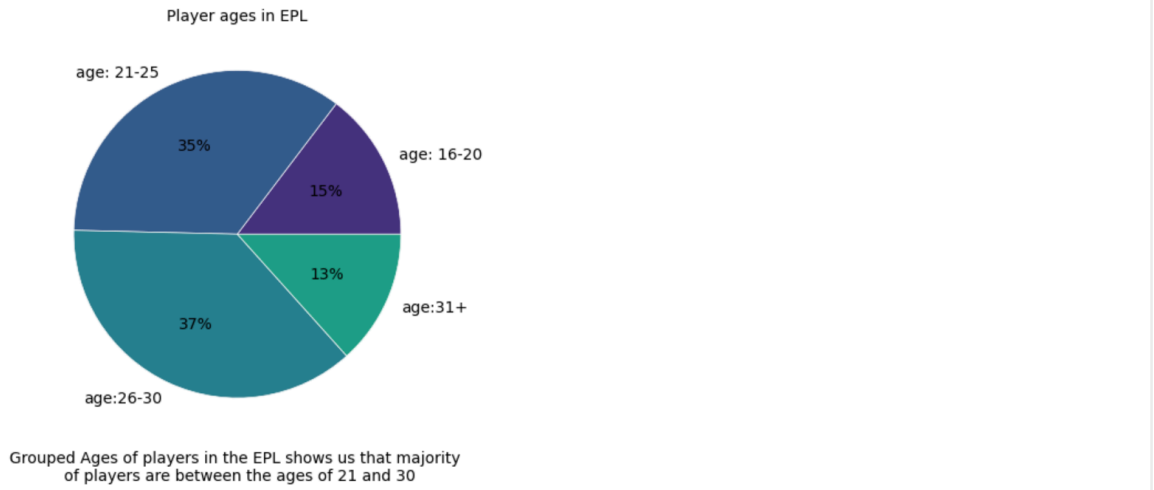
Players based on age groups!

```
#players based on age group

under20 = epl_df[epl_df['Age'] <=20]
age20_25 = epl_df[(epl_df['Age'] >20) & (epl_df['Age'] <=25)]
age25_30 = epl_df[(epl_df['Age'] >25) & (epl_df['Age'] <=30)]
over30 = epl_df[epl_df['Age'] >30]

#now i can create my pie chart for the above

x = np.array([under20['Name'].count(), age20_25['Name'].count(), age25_30['Name'].count(), over30['Name'].count()])
labels = ['age: 0-20', 'age: 21-25', 'age:26-30', 'age:31+']
plt.title('Player ages in EPL', fontsize=10)
plt.pie(x, labels=labels, autopct='%.f%%')
description = 'Grouped Ages of players in the EPL'
plt.text(0, -1.3, description, ha='center')
plt.show()
```



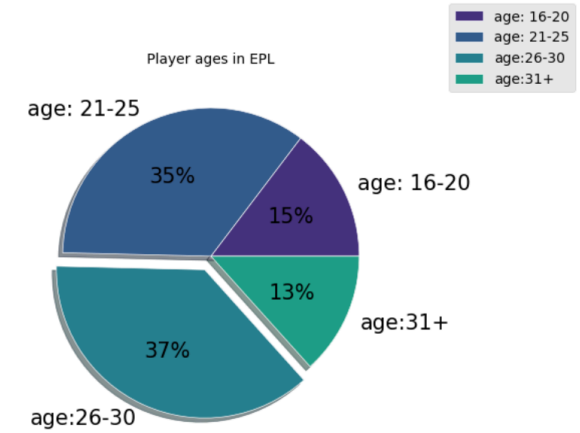
Below is the same as the above. I have just played around with the colors and description

```
#players based on age group

under20 = epl_df[epl_df['Age'] <=20]
age20_25 = epl_df[(epl_df['Age'] >20) & (epl_df['Age'] <=25)]
age25_30 = epl_df[(epl_df['Age'] >25) & (epl_df['Age'] <=30)]
over30 = epl_df[epl_df['Age'] >30]

#now i can create my pie chart for the above

x = np.array([under20['Name'].count(), age20_25['Name'].count(), age25_30['Name'].count(), over30['Name'].count()])
labels = ['age: 16-20', 'age: 21-25', 'age:26-30', 'age:31+']
plt.pie(x, labels=labels, autopct='%.f%%', colors=sns.color_palette('viridis'),wedgeprops={'edgecolor':'white'})
plt.title('Player ages in EPL', fontsize=10)
description = 'Grouped Ages of players in the EPL shows us that majority \n of players are between the ages of 21 and 30'
plt.text(0, -1.5, description, ha='center')
plt.show()
```



Below is the same as the above. I have just added plt.legend to indicate what each colour represents, ‘explode’ to visually separate one of the pies (the highest percentage) from the rest and shadow, and the font size of my labels

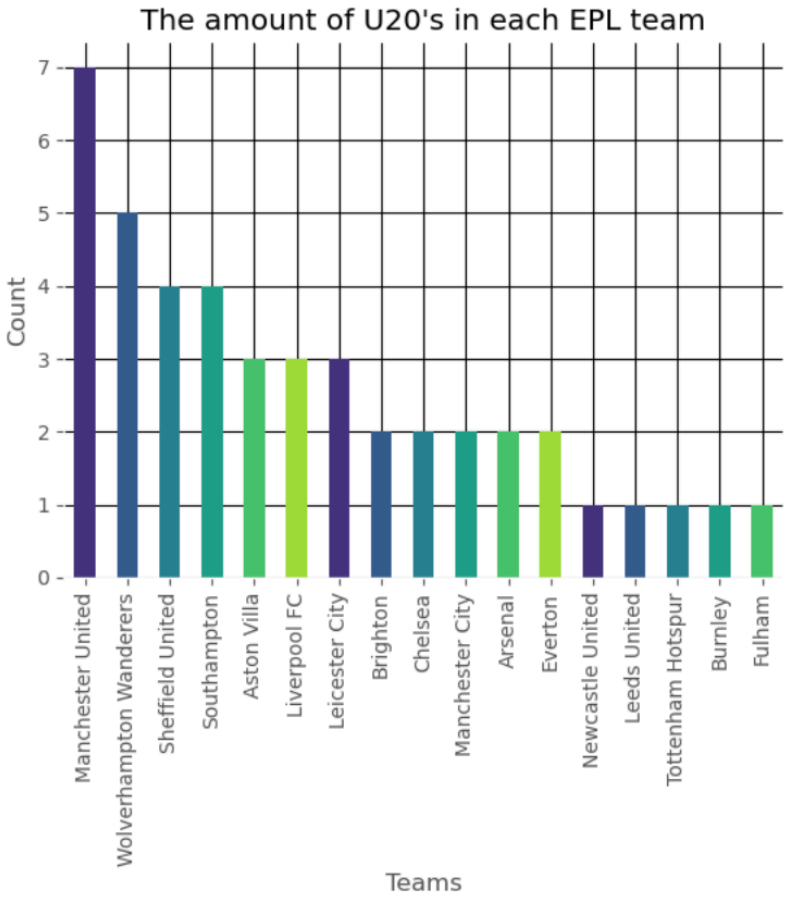
```
#players based on age group

under20 = epl_df[epl_df['Age'] <=20]
age20_25 = epl_df[(epl_df['Age'] >20) & (epl_df['Age'] <=25)]
age25_30 = epl_df[(epl_df['Age'] >25) & (epl_df['Age'] <=30)]
over30 = epl_df[epl_df['Age'] >30]

#now i can create my pie chart for the above

x = np.array([under20['Name'].count(), age20_25['Name'].count(), age25_30['Name'].count(), over30['Name'].count()])
labels = ['age: 16-20', 'age: 21-25', 'age:26-30', 'age:31+']
description = 'Grouped Ages of players in the EPL shows us that majority \n of players are between the ages of 21 and 30'
explode = [0, 0, 0.1, 0]


plt.pie(x, labels=labels, autopct='%.f%%', colors=sns.color_palette('viridis'),
        wedgeprops={'edgecolor':'white'},textprops={'fontsize': 15},
        explode=explode, shadow=True)
plt.legend(loc='upper right', labels=labels, bbox_to_anchor=(1.5,1.2))
plt.title('Player ages in EPL', fontsize=10)
plt.text(0, -1.8, description, ha='center')

plt.show()
```



Grouped Ages of players in the EPL shows us that majority
of players are between the ages of 21 and 30

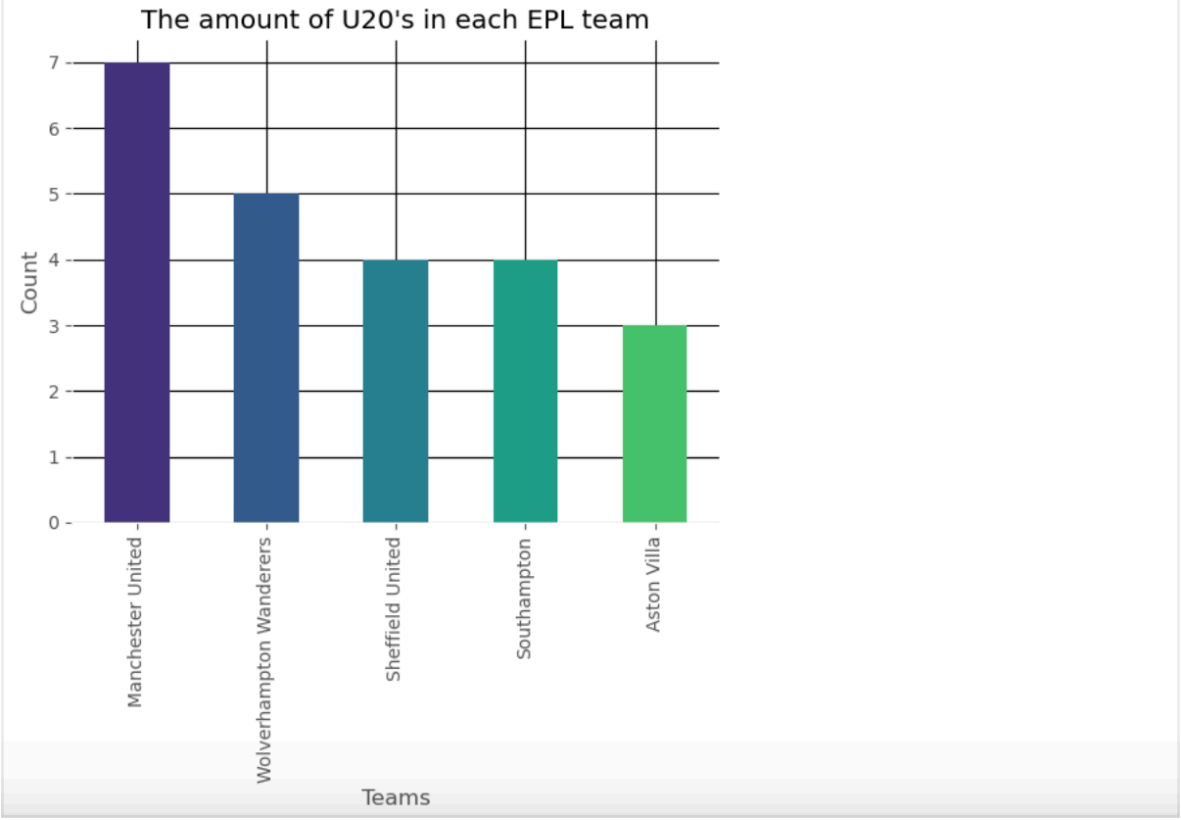
Total under 20 players from each EPL club

```
#total under 20 players from each epl club

players_under_20 = epl_df[epl_df['Age'] <20]
players_under_20['Club'].value_counts().plot(kind='bar', color=sns.color_palette('viridis'))

plt.title("The amount of U20's in each EPL team")
plt.xlabel('Teams')
plt.ylabel('Count')
plt.grid(color='black', linestyle='solid')
plt.gca().set_facecolor(color='white')
```



Used nlargest(5) to show me top 5 clubs with the most u20s

```
#total under 20 players from each epl club

players_under_20 = epl_df[epl_df['Age'] <20]
players_under_20['Club'].value_counts().nlargest(5).plot(kind='bar', color=sns.color_palette('viridis'))

plt.title("The amount of U20's in each EPL team")
plt.xlabel('Teams')
plt.ylabel('Count')
plt.grid(color='black', linestyle='solid')
plt.gca().set_facecolor(color='white')
```



Players in Manchester United who are u20

```
#under 20's players in a specific team
players_under_20[players_under_20['Club'] == 'Manchester United']
```

[24…

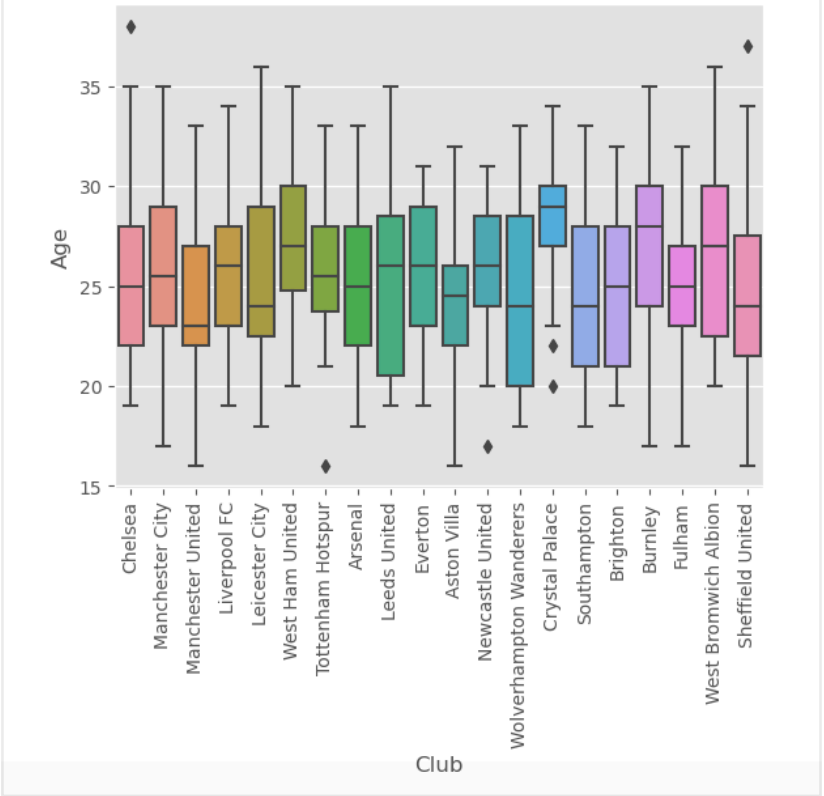
	Name	Club	Nationality	Position/s	Age	Matches_played	Starts	Total_mins_played	Goals	Assists	Passes_Attempted	Perc_Passes_Completed	Per
61	Mason Greenwood	Manchester United	ENG	FW	18	31	21	1822	7	2	732	83.1	
72	Brandon Williams	Manchester United	ENG	DF	19	4	2	188	0	0	140	85.7	
73	Amad Diallo	Manchester United	CIV	FW	18	3	2	166	0	1	64	84.4	
74	Anthony Elanga	Manchester United	SWE	FW	18	2	2	155	1	0	53	81.1	
76	Shola Shoretire	Manchester United	ENG	FW	16	2	0	11	0	0	8	75.0	
78	Hannibal Mejbri	Manchester United	FRA	MF	17	1	0	9	0	0	3	100.0	
79	William Thomas Fish	Manchester United	ENG	DF	17	1	0	1	0	0	1	0.0	

+ Code + Markdown

- Average age of players in each club
- I used a box plot using seaborn to show average age of players from each club + the min/max age in each club

```
#Average age of players in each club
sns.boxplot(x = 'Club', y = 'Age', data = epl_df)
plt.xticks(rotation = 90)
```

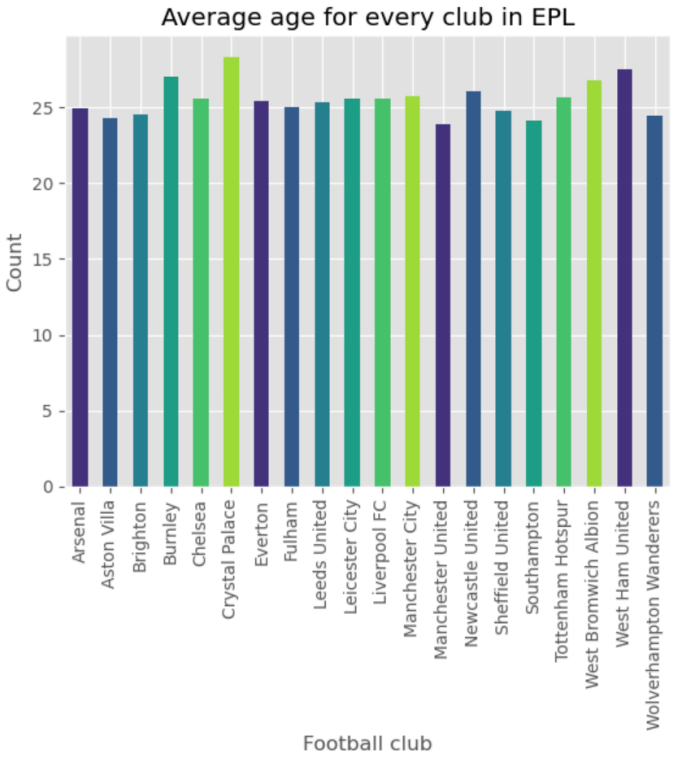
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19]),
 [Text(0, 0, 'Chelsea'),
  Text(1, 0, 'Manchester City'),
  Text(2, 0, 'Manchester United'),
  Text(3, 0, 'Liverpool FC'),
  Text(4, 0, 'Leicester City'),
  Text(5, 0, 'West Ham United'),
  Text(6, 0, 'Tottenham Hotspur'),
  Text(7, 0, 'Arsenal'),
  Text(8, 0, 'Leeds United'),
  Text(9, 0, 'Everton'),
  Text(10, 0, 'Aston Villa'),
  Text(11, 0, 'Newcastle United'),
  Text(12, 0, 'Wolverhampton Wanderers'),
  Text(13, 0, 'Crystal Palace'),
  Text(14, 0, 'Southampton'),
  Text(15, 0, 'Brighton'),
  Text(16, 0, 'Burnley'),
  Text(17, 0, 'Fulham'),
  Text(18, 0, 'West Bromwich Albion'),
  Text(19, 0, 'Sheffield United')])
```



- Showed just the average age for each football team in EPL

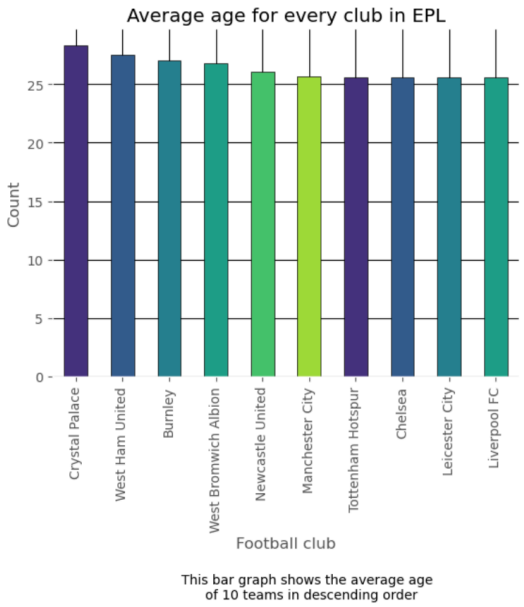
```
average_age_by_club = epl_df.groupby('Club')['Age'].mean().plot(kind='bar', color=sns.color_palette('viridis'))

plt.title('Average age for every club in EPL')
plt.xlabel('Football club')
plt.ylabel('Count')
plt.figure(figsize=(6,6))
```

```
<Figure size 600x600 with 0 Axes>
```



```
plt.grid(color='black', linestyle='solid')
plt.gca().set_facecolor(color='white')
average_age_by_club = epl_df.groupby('Club')['Age'].mean().nlargest(10).plot(kind='bar', color=sns.color_palette('viridis'), edgecolor='black')

plt.title('Average age for every club in EPL')
description = 'This bar graph shows the average age \n of 10 teams in descending order'
plt.text(5, -19, description, ha='center')
plt.xlabel('Football club')
plt.ylabel('Count')
plt.figure(figsize=(6,6))
```

```
<Figure size 600x600 with 0 Axes>
```



- Below's code shows me the average age of each club in the EPL.

```
number_of_players = epl_df.groupby('Club').size()
data = (epl_df.groupby('Club')['Age'].sum())/number_of_players
data.sort_values(ascending = False)
```

```
[52…  Club
Crystal Palace              28.333333
West Ham United             27.500000
Burnley                     27.040000
West Bromwich Albion        26.766667
Newcastle United            26.074074
Manchester City             25.708333
Tottenham Hotspur           25.625000
Chelsea                     25.592593
Leicester City              25.592593
Liverpool FC                25.571429
Everton                     25.413793
Leeds United                25.347826
Fulham                      25.035714
Arsenal                     24.965517
Sheffield United            24.814815
Brighton                    24.555556
Wolverhampton Wanderers     24.444444
Aston Villa                 24.291667
Southampton                 24.137931
Manchester United           23.862069
dtype: float64
```

Bellow shows me the top 5 most populated age years. It shows that there are 46 twenty six year olds in the EPL. To do this I can use nlargest() or head()

```
age_count = epl_df['Age'].value_counts().nlargest()
print(age_count)
```

```
26    47
28    46
23    44
22    40
27    40
Name: Age, dtype: int64
```

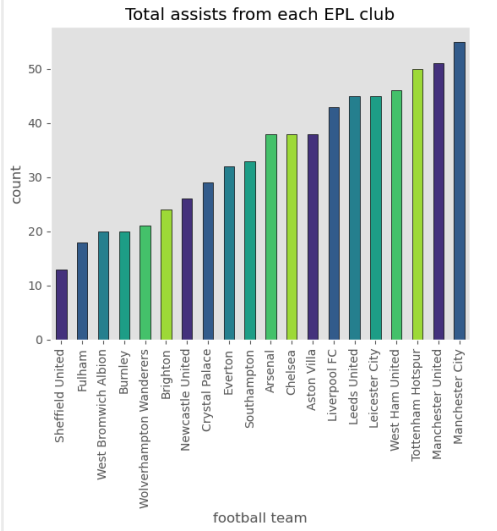
```
age_count = epl_df['Age'].value_counts().head()
print(age_count)
```

```
26    47
28    46
23    44
22    40
27    40
Name: Age, dtype: int64
```

- Totalling the assists column by clubs

```
# total assists from each club

total_assists_by_club = epl_df.groupby('Club')['Assists'].sum().nsmallest(20).plot(kind='bar', color=sns.color_palette('viridis'), grid=False, edgecolor='black')
plt.title('Total assists from each EPL club')
plt.xlabel('football team')
plt.ylabel('count')
description = 'This bar graph shows the amount of assists from each premier league team \n My guess was that manchester city or liverpool would be leading'
plt.text(10, -43, description, ha='center')
```

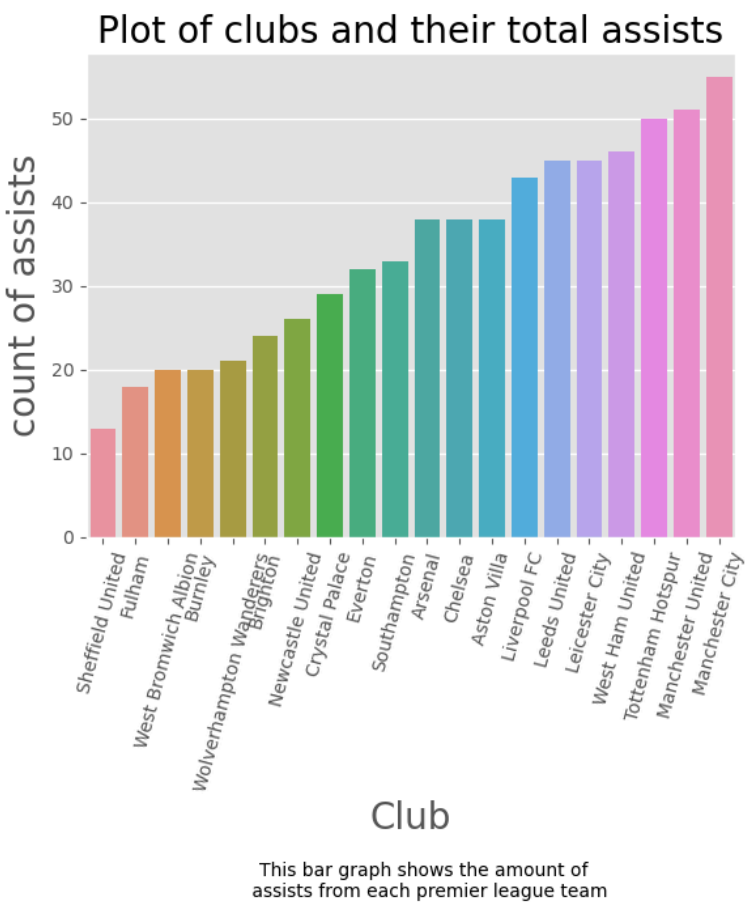
```
Text(10, -43, 'This bar graph shows the amount of assists from each premier league team \n My guess was that manchester city or liverpool would be leading')
```



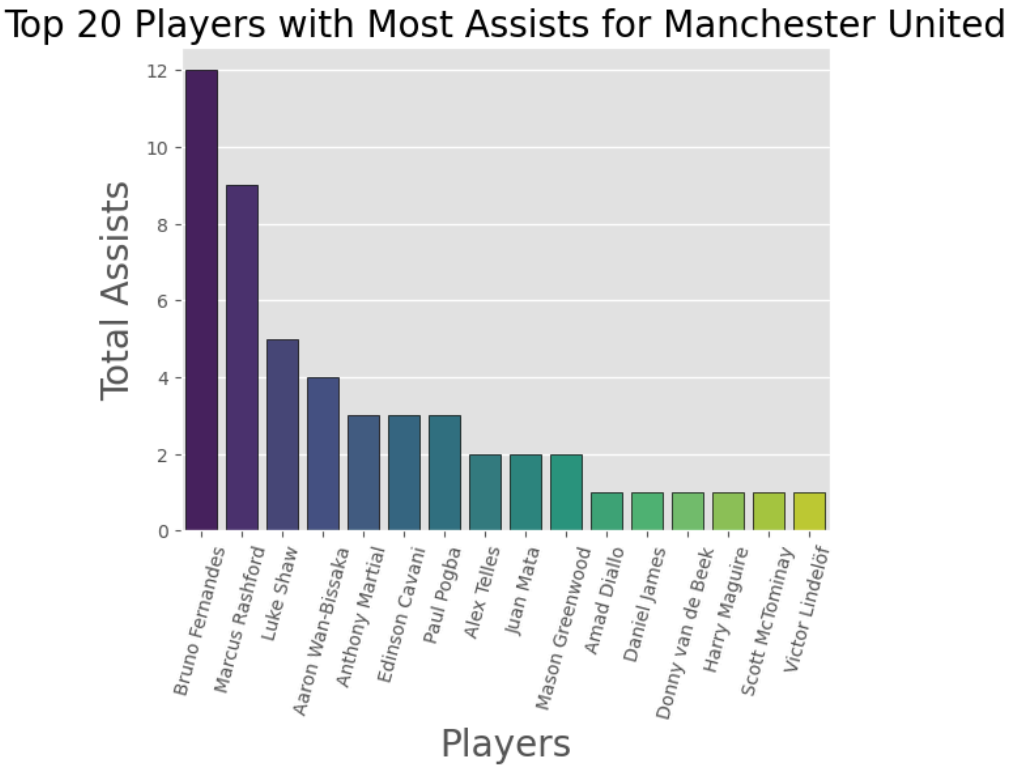
This bar graph shows the amount of assists from each premier league team
My guess was that manchester city or liverpool would be leading

```
total_assists_by_club = epl_df.groupby('Club')['Assists'].sum().reset_index()
my_graph = sns.barplot(x = 'Club', y = 'Assists', data = total_assists_by_club.sort_values('Assists'))
my_graph.set_xlabel('Club', fontsize=20)
my_graph.set_ylabel('count of assists', fontsize=20)
plt.xticks(rotation=75)
plt.title('Plot of clubs and their total assists', fontsize=20)
description = 'This bar graph shows the amount of \n assists from each premier league team'
plt.text(10, -43, description, ha='center')
```

```
Text(10, -43, 'This bar graph shows the amount of \n assists from each premier league team')
```



```python
# Filter the data for the specific club you want to focus on
specific_club = 'Manchester United'
specific_club_data = epl_df[epl_df['Club'] == specific_club]

# Grouping by 'name' (player) and summing the 'Assists' for each player within the specific club
total_assists_by_player = specific_club_data.groupby('Name')['Assists'].sum().reset_index()

# Selecting the top 20 players with the most assists for the specific club
top_players_by_assists = total_assists_by_player.nlargest(16, 'Assists')

# Creating the bar plot
graph = sns.barplot(x='Name', y='Assists', data=top_players_by_assists,
                    palette='viridis',edgecolor='black')

graph.set_xlabel('Players', fontsize=20)
graph.set_ylabel('Total Assists', fontsize=20)
plt.xticks(rotation=75)
plt.title(f'Top 20 Players with Most Assists for {specific_club}', fontsize=20)

# Adding a description text
description = f'This bar graph shows the top 20 players with the most assists for {specific_club}.'
plt.text(8, -8, description, ha='center')

plt.show()
```



This bar graph shows the top 20 players with the most assists for Manchester United.

- Showing top 10 assisting players in the EPL

```
#top 10 players with the most assists

# Selecting the top ten players with most assists and specific columns to show with it
top_10_assists = epl_df.nlargest(n=10, columns='Assists')[['Name', 'Club', 'Assists','Goals', 'Matches_played']]

# Displaying the top ten players with most assist
top_10_assists
```

	Name	Club	Assists	Goals	Matches_played
162	Harry Kane	Tottenham Hotspur	14	23	35
34	Kevin De Bruyne	Manchester City	12	6	25
51	Bruno Fernandes	Manchester United	12	18	37
161	Son Heung-min	Tottenham Hotspur	10	17	37
273	Jack Grealish	Aston Villa	10	6	26
54	Marcus Rashford	Manchester United	9	11	37
110	Jamie Vardy	Leicester City	9	15	34
220	Raphael Dias Belloli	Leeds United	9	6	30
2	Timo Werner	Chelsea	8	6	35
136	Aaron Cresswell	West Ham United	8	0	36

- Finding out goals per match
- Earlier on I made 2 new columns: mins per match and goals per match column

```
epl_df['Mins_per_match'] = (epl_df['Total_mins_played'] / epl_df['Matches_played']).astype(int)
epl_df['Goals_per_match'] = (epl_df['Goals'] / epl_df['Matches_played']).astype(float)
```

```
epl_df.head()
```

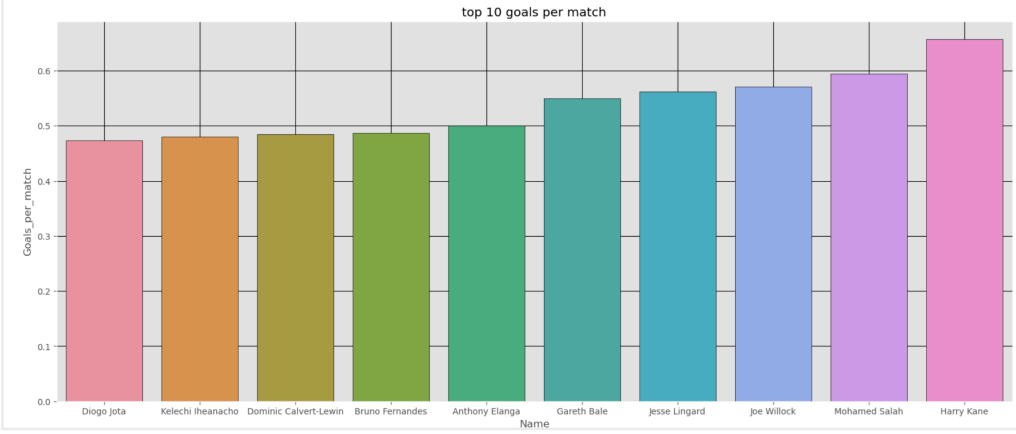
_Passes_Completed	Penalty_Goals	Penalty_Attempted	xG	xA	Yellow_Cards	Red_Cards	Mins_per_match	Goals_per_match
82.3	1	1	0.21	0.24	2	0	80	0.166667
84.6	0	0	0.00	0.00	2	0	88	0.000000
77.2	0	0	0.41	0.21	2	0	74	0.171429
78.6	0	0	0.10	0.11	3	0	84	0.111111
85.0	0	0	0.06	0.12	3	0	74	0.031250

```
#Goals per match

top10goals_per_match = epl_df[['Name', 'Goals_per_match','Matches_played', 'Goals']].nlargest(n=10, columns='Goals_per_match')
top10goals_per_match
```

	Name	Goals_per_match	Matches_played	Goals
162	Harry Kane	0.657143	35	23
81	Mohamed Salah	0.594595	37	22
307	Joe Willock	0.571429	14	8
145	Jesse Lingard	0.562500	16	9
175	Gareth Bale	0.550000	20	11
74	Anthony Elanga	0.500000	2	1
51	Bruno Fernandes	0.486486	37	18
237	Dominic Calvert-Lewin	0.484848	33	16
120	Kelechi Iheanacho	0.480000	25	12
92	Diogo Jota	0.473684	19	9

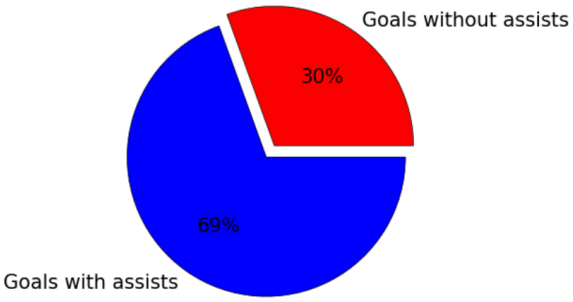
```
#Goals per match

top10goals_per_match = epl_df[['Name', 'Goals_per_match','Matches_played', 'Goals']].nlargest(n=10, columns='Goals_per_match')
top10goals_per_match

my_graph = sns.barplot(x='Name', y='Goals_per_match', data = top10goals_per_match.sort_values('Goals_per_match'), edgecolor='black')
my_graph.set_xlabel('Name')
my_graph.set_ylabel('Goals_per_match')
my_graph.grid(False)

plt.grid(color='black', linestyle='solid')


plt.title('top 10 goals per match')
```

```
Text(0.5, 1.0, 'top 10 goals per match')
```



Pie chart - goals without assists and goals with assists

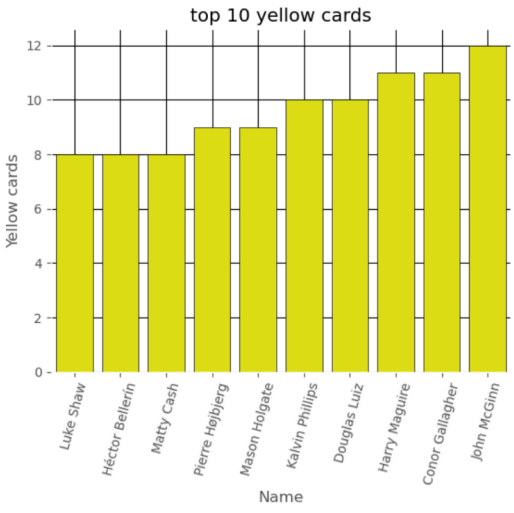
```
# Pie chart - goals without assists and goals with assists

total_goals = epl_df['Goals'].sum()
assists = epl_df['Assists'].sum()

data = [total_goals - assists, assists]
labels = ['Goals without assists', 'Goals with assists' ]
color = ('red', 'blue')

plt.pie(data, labels=labels, colors=color, autopct = '%.d%%', wedgeprops={'edgecolor':'black'}, textprops={'fontsize': 15}, explode=explode)
plt.show()
explode = [0, 0.1]
```



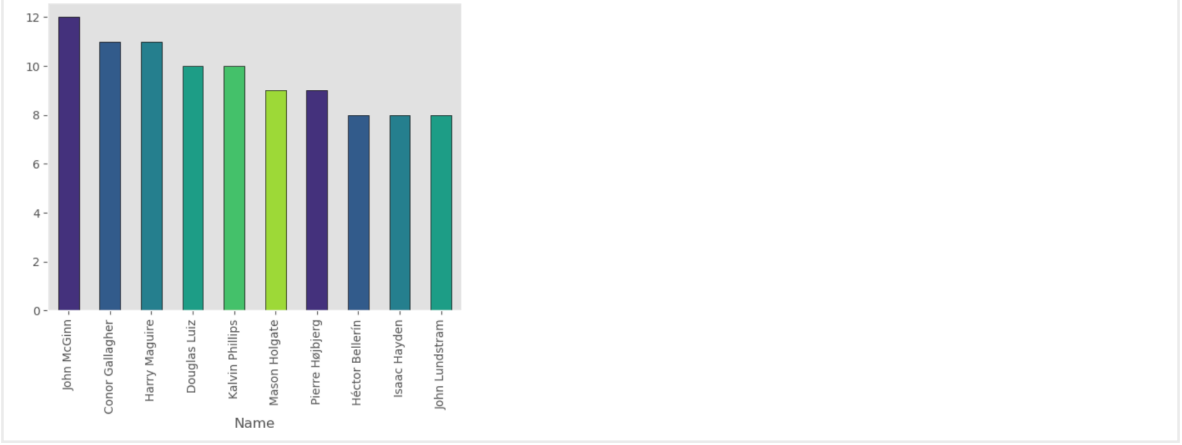
#Top 10 players with most yellow cards - barplot

```
#Top 10 players with most yellow cards

top10players_yellowcards = epl_df[['Name', 'Yellow_Cards',]].nlargest(n=10, columns='Yellow_Cards')

my_graph = sns.barplot(x='Name', y='Yellow_Cards', data = top10players_yellowcards.sort_values('Yellow_Cards'), edgecolor='black', color='yellow')
my_graph.set_xlabel('Name')
my_graph.set_ylabel('Yellow cards')
my_graph.grid(False)

plt.grid(color='black', linestyle='solid')
plt.xticks(rotation=75)
plt.gca().set_facecolor(color='white')


plt.title('top 10 yellow cards')
```

```
Text(0.5, 1.0, 'top 10 yellow cards')
```

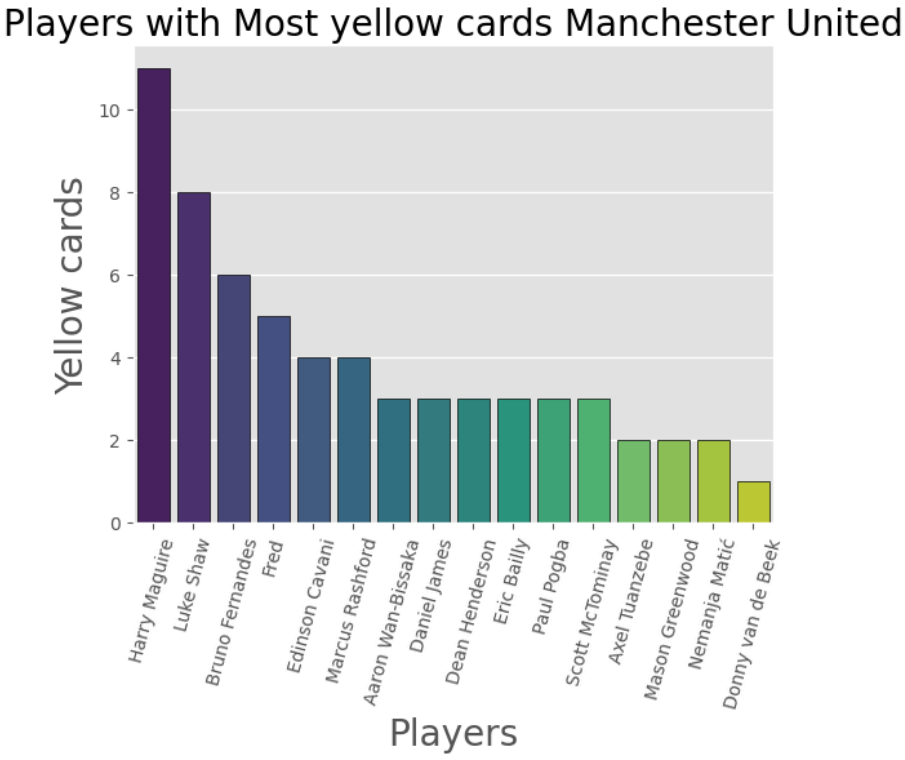


```
]:
#Top 10 players with most yellow cards

top_yellows_clubs = epl_df.groupby('Name')['Yellow_Cards'].sum().nlargest(10).plot(kind='bar', color=sns.color_palette('viridis'), grid=False, edgecolor='black')
```



– Players from a specific club with the most yellow cards.

```
# Filter the data for the specific club you want to focus on
specific_club = 'Manchester United'
specific_club_data = epl_df[epl_df['Club'] == specific_club]

# Grouping by 'name' (player) and summing the 'yellow cards' for each player within the specific club
total_yellow = specific_club_data.groupby('Name')['Yellow_Cards'].sum().reset_index()

# Selecting the top 20 players with the most assists for the specific club
total_yellow = total_yellow.nlargest(16, 'Yellow_Cards')

# Creating the bar plot
graph = sns.barplot(x='Name', y='Yellow_Cards', data=total_yellow,
                    palette='viridis',edgecolor='black')

graph.set_xlabel('Players', fontsize=20)
graph.set_ylabel('Yellow cards', fontsize=20)
plt.xticks(rotation=75)
plt.title(f'Players with Most yellow cards {specific_club}', fontsize=20)

# Adding a description text
description = f'This bar graph shows the top 20 players with the most yellows for {specific_club}.'
plt.text(8, -8, description, ha='center')

plt.show()
```



This bar graph shows the top 20 players with the most yellows for Manchester United.

Useful notes to self:

1. **Round Brackets ()**:
   - Function Calls: Use round brackets when calling functions or methods. For example, when you use methods like `df.head()`, `df.info()`, or `pd.read_csv()`.
   - Grouping: Round brackets are also used for grouping expressions in mathematical calculations or conditions. For instance, `(2 + 3) * 4` or `(x > 5) & (y < 10)`.
2. **Square Brackets []**:
   - Indexing: Square brackets are used for indexing and accessing elements from a pandas DataFrame or Series. For example, `df['column_name']` or `series[3]`.
   - Slicing: You can also use square brackets for slicing data from a DataFrame or Series. For instance, `df[1:5]` will return rows 1 to 4 (excluding row 5).
   - Lists: Square brackets are used to create lists in Python. For example, `my_list = [1, 2, 3]`.
3. **Curly Brackets {}**:
   - Dictionaries: Curly brackets are used to create dictionaries in Python. Dictionaries are a data structure that stores key-value pairs. For example, `my_dict = {'name': 'John', 'age': 30}`.

)

1. Named palettes: Seaborn provides a set of named color palettes that you can directly use. Some of the named palettes are:
    - "deep"
    - "muted"
    - "pastel"
    - "bright"
    - "dark"
    - "colorblind"
2. Matplotlib colormaps: You can use any of the colormaps available in Matplotlib by passing the colormap name as a string. Some popular colormaps are:
    - "viridis"
    - "plasma"
    - "inferno"
    - "magma"
    - "cividis"
    - "coolwarm"
    - "rainbow"
    - "autumn"
    - "cool"