Analysis of EPL Data for the 2020-2021 Season.

Project Name- EPL 2020-2021 Data Anlaysis

Project By- Abhishek Kumar Upadhyay

Date- 24/02/2022

About the Dataset- This dataset is a collection of basic and vital stats for the 2020-21 English Premier League season. The dataset includes every player who has played in the EPL and standard stats like Goals, Assists, xG, xA, Passes_Attempted and Pass Accuracy and more.

Content-

Position- Each player has a certain position, in which he plays regularly. The position in this dataset are, FW - Forward, MF - Midfield, DF - Defensive, GK - Goalkeeper

Starts- The number of times the player was named in the starting 11 by the manager.

Mins- The number of minutes played by the player.

Goals- The number of Goals scored by the player.

Assists- The number of times the player has assisted other player in scoring the goal.

Passes_Attempted- The number of passes attempted by the player.

Perc_Passes_Completed- The number of passes that the player accurately passed to his teammate.

xG- Expected number of goals from the player in a match.

xA- Expected number of assists from the player in a match.

Yellow_Cards- The players get a yellow card from the referee for indiscipline, technical fouls, or other minor fouls.

Red Cards- The players get a red card for accumulating 2 yellow cards in a single game, or for a major foul.

Imports the necessary libraries

In [2]:
```
# Import the libraries

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

Reads a CSV file

In [3]:
```
# load Dataset

epl_df = pd.read_csv('F:\old desktop data\Python\EPL 2020-21 Data Analysis\EPL 20_21.csv')
```

Display the first 5 rows of the DataFrame epl_df

In [4]:
```
# for the first 5 rows

epl_df.head()
```

Out[4]:

	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
0	Mason Mount	Chelsea	ENG	MF,FW	21	36	32	2890	6	5	1881	82.3	1	1	0.21	0.24	2	0
1	Edouard Mendy	Chelsea	SEN	GK	28	31	31	2745	0	0	1007	84.6	0	0	0.00	0.00	2	0
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	826	77.2	0	0	0.41	0.21	2	0
3	Ben Chilwell	Chelsea	ENG	DF	23	27	27	2286	3	5	1806	78.6	0	0	0.10	0.11	3	0
4	Reece James	Chelsea	ENG	DF	20	32	25	2373	1	2	1987	85.0	0	0	0.06	0.12	3	0

Display structure and content of the DataFrame

In [5]:
```
# for the total no. of column

epl_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
```

Display statistics of the epl_df DataFrame

In [6]:
```
# for the summary statistics of our dataset of each numeric column
epl_df.describe() # describe function only works on numeric column
```

Out[6]:

	Age	Matches	Starts	Mins	Goals	Assists	Passes_Attempted	Perc_Passes_Completed	Penalty_Goals	Penalty_Attempted	xG	xA	Yellow_Cards	Red_Cards
count	532.000000	532.000000	532.000000	532.000000	532.000000	532.000000	532.000000	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	77.823872	0.191729	0.234962	0.113289	0.072650	2.114662	0.090226
std	4.319404	11.840459	11.921161	1043.171856	3.338009	2.095191	631.372522	13.011631	0.850881	0.975818	0.148174	0.090072	2.269094	0.293268
min	16.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	22.000000	9.000000	4.000000	426.000000	0.000000	0.000000	171.500000	73.500000	0.000000	0.000000	0.010000	0.000000	0.000000	0.000000
50%	26.000000	21.000000	15.000000	1345.000000	1.000000	0.000000	573.500000	79.200000	0.000000	0.000000	0.060000	0.050000	2.000000	0.000000
75%	29.000000	30.000000	27.000000	2303.500000	2.000000	2.000000	1129.500000	84.625000	0.000000	0.000000	0.150000	0.110000	3.000000	0.000000
max	38.000000	38.000000	38.000000	3420.000000	23.000000	14.000000	3214.000000	100.000000	9.000000	10.000000	1.160000	0.900000	12.000000	2.000000

Identifying any missing values in the dataset

In [7]:
```
# total no of null value that are there in each coloumn
epl_df.isna().sum()
```

Out[7]:
```
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
```

Create two new column

In [8]:
```
# create two new column
epl_df['MinsPerMatch'] = (epl_df['Mins']/epl_df['Matches']).astype(int)
epl_df['GoalsPerMatch'] = (epl_df['Goals']/epl_df['Matches']).astype(float)
epl_df.head() # from this code we will see the two new column at last of our dataset
```

Out[8]:

	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	MinsPerMatch
0	Mason Mount	Chelsea	ENG	MF,FW	21	36	32	2890	6	5	1881	82.3	1	1	0.21	0.24	2	0	80
1	Edouard Mendy	Chelsea	SEN	GK	28	31	31	2745	0	0	1007	84.6	0	0	0.00	0.00	2	0	88
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	826	77.2	0	0	0.41	0.21	2	0	74
3	Ben Chilwell	Chelsea	ENG	DF	23	27	27	2286	3	5	1806	78.6	0	0	0.10	0.11	3	0	84
4	Reece James	Chelsea	ENG	DF	20	32	25	2373	1	2	1987	85.0	0	0	0.06	0.12	3	0	74

Display the number of rows and columns in the DataFrame

In [9]:
```
# Number of rows and columns in the DataFrame.
"""The first element of the tuple is the number of rows, and the second element is the number of columns. Using this command to
quickly get an idea of the size of your data, which can be helpful for understanding its structure and planning data analysis
tasks."""
epl_df.shape
```

Out[9]:
```
(532, 20)
```

Calculate total goals

In [10]:
```
# Total Goals

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

```
986
```

Calculates the total number of penalty goals scored in the 2020-21 English Premier League

In [11]:
```
# Penalty Goals

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

```
102
```

Calculates the total number of penalty kicks attempted in the 2020-21 English Premier League season

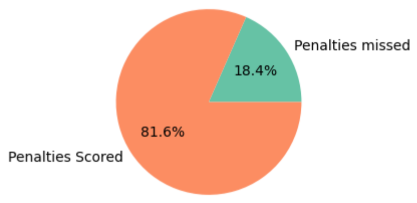
In [12]:
```
# Penalty Attempts

Total_Penalty_Attempts = epl_df["Penalty_Attempted"].sum()
print(Total_Penalty_Attempts)
```

```
125
```

Visualize the proportion of penalty kicks missed and scored in the 2020-21 English Premier League season.

In [13]:
```
# Pie chart for penalties missed vs scored

# Figure size
plt.figure(figsize=(5,3))
```

```
# calculates the total number of penalty kicks not scored
pl_not_scored = epl_df['Penalty_Attempted'].sum() - Total_Penalty_Goals
#You can write this because you already have Total_Penality_Attempts- pl_not_scored = Total_Penality_Attempts-Total_Penalty_Goals

# creates a list called data with two elements to showing the proportion of penalties missed versus penalties scored
data = [pl_not_scored, Total_Penalty_Goals]

# Add labels
labels = ['Penalties missed', 'Penalties Scored']

# creates a color palette
color = sns.color_palette("Set2")

# Create pie charts
plt.pie(data, labels = labels, colors = color , autopct = '%.1f%%')
# Without decimal points- plt.pie(data, labels = labels, colors = color , autopct = '%.1f%%')

# Display the pie chart
plt.show()
```



Note-

☛ `autopct` parameter in the pie() function of matplotlib library is used to format the percentages displayed on the pie chart. In this case, the % symbol indicates that the values should be displayed as percentages, and the .0f format specifier indicates that the values should be displayed with no decimal places. So, `autopct = '%.0f%%'` means that the percentages on the pie chart will be displayed as integers with a % symbol added to the end. For example, if the percentage of penalties scored is 75%, it will be displayed as "75%". If the percentage is 100%, it will be displayed as "100%".

☛ If you want to display decimal values for the percentages, you can modify the autopct parameter to include a format string that specifies the number of decimal places you want to display. For example, autopct='%.1f%%' will display one decimal place for the percentages.

Here's the modified code with `autopct='%.1f%%'` : Just like above

List the Name of total unique position in epl dataframe

In [14]:
```
# Unique Position of each of the player
epl_df['Position'].unique()
```

Out[14]:
```
array(['MF,FW', 'GK', 'FW', 'DF', 'MF', 'FW,MF', 'FW,DF', 'DF,MF',
       'MF,DF', 'DF,FW'], dtype=object)
```

Creates a Pandas DataFrame called 'Player_Name_df' that contains a single column with the unique player names from the 'Name' column of the 'epl_df' DataFrame.

In [15]:
```
#Get the unique list of players and sort it in alphabetical order
Total_Unique_Player = epl_df['Name'].unique()
Total_Unique_Player.sort()

# Create a DataFrame with the sorted player names in a single column
Player_Name_df = pd.DataFrame(Total_Unique_Player, columns=['Player_Name'])

# Print the DataFrame
Player_Name_df
```

Out[15]:

	Player_Name
0	Aaron Connolly
1	Aaron Cresswell
2	Aaron Ramsdale
3	Aaron Wan-Bissaka
4	Abdoulaye Doucouré
...	...
519	Zack Steffen
520	Çağlar Söyüncü
521	Érik Lamela
522	İlkay Gündoğan
523	Łukasz Fabiański

524 rows × 1 columns

List the total player who played for forwand position in EPL football 'FW'

In [16]:
```
# find the total player who played for forwand position in EPL football 'FW'
total_FW_player = epl_df[epl_df['Position'] == 'FW']

# display the total_FW_player
total_FW_player
```

Out[16]:

	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	MinsPer
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	826	77.2	0	0	0.41	0.21	2	0	
16	Tammy Abraham	Chelsea	ENG	FW	22	22	12	1040	6	1	218	68.3	0	0	0.56	0.07	0	0	
19	Olivier Giroud	Chelsea	FRA	FW	33	17	8	748	4	0	217	74.2	0	0	0.58	0.09	1	0	
23	Ruben Loftus-Cheek	Chelsea	ENG	FW	24	1	1	60	0	0	16	68.8	0	0	0.00	0.00	0	0	
30	Raheem Sterling	Manchester City	ENG	FW	25	31	28	2536	10	7	1127	85.4	0	1	0.43	0.17	4	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
516	Oliver Burke	Sheffield United	SCO	FW	23	25	14	1269	1	1	262	70.6	0	0	0.17	0.13	2	0	
518	Oliver McBurnie	Sheffield United	SCO	FW	24	23	12	1324	1	0	426	62.9	0	0	0.21	0.07	2	0	
519	Rhian Brewster	Sheffield United	ENG	FW	20	27	12	1128	0	0	225	69.3	0	0	0.14	0.13	1	0	
523	Billy Sharp	Sheffield United	ENG	FW	34	16	7	735	3	0	123	69.9	2	2	0.33	0.07	1	0	
526	Daniel Jebbison	Sheffield United	ENG	FW	17	4	3	284	1	0	34	70.6	0	0	0.50	0.01	0	0	

81 rows × 20 columns

Sort the 'total_FW_player' dataframe by Name in ascending order

In [17]:
```python
# sort the total_FW_player by 'Name'
total_FW_player = epl_df[epl_df['Position'] == 'FW'].sort_values(by='Name')

# display the total_FW_player sorted by 'Name'
total_FW_player
```

Out[17]:

	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	Mi
412	Aaron Connolly	Brighton	IRL	FW	20	17	9	791	2	1	101	78.2	0	0	0.40	0.02	0	0	
320	Adama Traoré	Wolverhampton Wanderers	ESP	FW	24	37	28	2649	2	2	879	65.9	0	0	0.08	0.18	4	0	
460	Aleksandar Mitrović	Fulham	SRB	FW	25	27	13	1402	3	3	384	76.0	1	2	0.42	0.17	3	0	
191	Alexandre Lacazette	Arsenal	FRA	FW	29	31	22	1923	13	2	524	78.2	3	3	0.46	0.13	3	0	
73	Amad Diallo	Manchester United	CIV	FW	18	3	2	166	0	1	64	84.4	0	0	0.02	0.26	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
261	Theo Walcott	Everton	ENG	FW	31	1	0	13	0	0	1	100.0	0	0	0.00	0.00	0	0	
2	Timo Werner	Chelsea	GER	FW	24	35	29	2602	6	8	826	77.2	0	0	0.41	0.21	2	0	
276	Trézéguet	Aston Villa	EGY	FW	25	21	12	1166	2	1	328	69.5	0	0	0.29	0.15	0	0	
344	Wilfried Zaha	Crystal Palace	CIV	FW	27	30	29	2612	11	2	779	75.9	2	2	0.26	0.11	6	0	
328	Willian José	Wolverhampton Wanderers	BRA	FW	28	17	12	1110	1	0	306	81.4	0	0	0.15	0.05	0	0	

81 rows × 20 columns

Calulate the total Unique nation who participate in EPL 2020-2021

In [18]:
```python
# calculate the total no of unique nations from where players played
total_unique_nationality = np.size(epl_df['Nationality'].unique())

# Display the unique nationality
total_unique_nationality
```

Out[18]:
```
59
```

Count the player from each Country

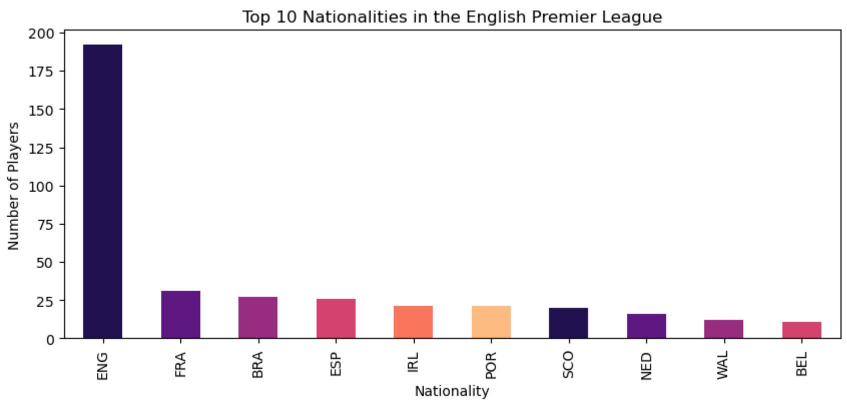
In [47]:
```python
# list player from which country
Nationality = epl_df.groupby('Nationality').size().sort_values(ascending = False)

# Display all the natiobality with highest player
Nationality
```

Out[47]:
```
Nationality
ENG    192
FRA     31
BRA     27
ESP     26
IRL     21
POR     21
SCO     20
NED     16
WAL     12
BEL     11
GER      9
ARG      8
CIV      8
NGA      7
DEN      6
SUI      6
USA      6
SEN      5
EGY      5
COL      5
SWE      5
TUR      5
GHA      5
POL      5
NIR      5
ITA      5
SRB      4
AUS      4
NOR      3
ALG      3
JAM      3
CZE      3
ISL      3
RSA      2
COD      2
CRO      2
MAR      2
GAB      2
SVK      2
PAR      2
CMR      2
UKR      2
JPN      2
MLI      2
CAN      1
URU      1
BFA      1
AUT      1
BIH      1
KOR      1
SKN      1
GRE      1
NZL      1
GUI      1
IRN      1
MTN      1
MKD      1
MEX      1
ZIM      1
dtype: int64
```

Visualize the nationality who is having highest no of players

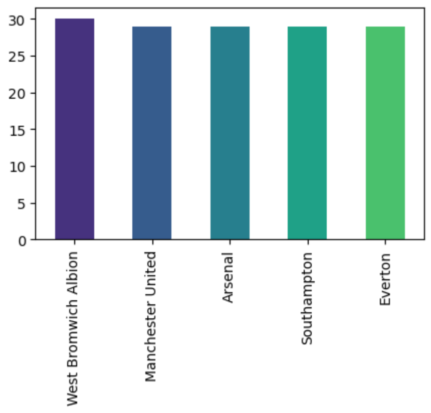
In [20]:
```python
# Visualize the Nationality  of highest player
Nationality = epl_df.groupby('Nationality').size().sort_values(ascending = False)
Nationality.head(10).plot(kind = 'bar', figsize=(10,4), color = sns.color_palette("magma"))

# Add a title and axis labels to the plot
plt.title('Top 10 Nationalities in the English Premier League')
plt.xlabel('Nationality')
plt.ylabel('Number of Players')
```

Out[20]:
```
Text(0, 0.5, 'Number of Players')
```



Find largest 5 Clubs with max players

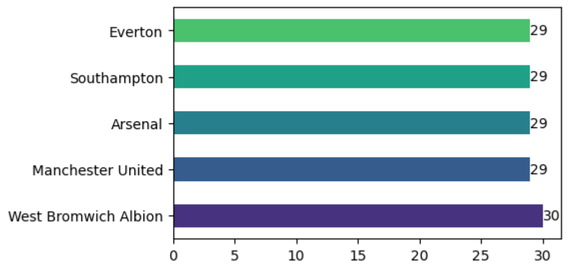
In [21]:
```python
# Club with maximum players in their squad
Max_player_club = epl_df['Club'].value_counts().nlargest(5).plot(kind='bar',figsize=(5,3), color = sns.color_palette("viridis"))
```



OR

You can plot on bar chart

In [22]:
```python
# Club with maximum players in their squad

# You can plot this above horizontally and add the count label on each bar

# 'barh' to create a horizontal bar plot.
Max_player_club = epl_df['Club'].value_counts().nlargest(5).plot(kind='barh',figsize=(5,3), color=sns.color_palette("viridis"))

# Add count labels above each bar
for i, v in enumerate(epl_df['Club'].value_counts().nlargest(5)):
    Max_player_club.text(v, i, str(v), color='black', fontsize=10, ha='left', va='center')
```



Count the players by age distribution

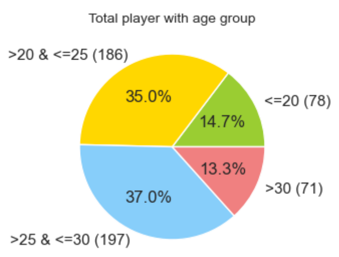
In [23]:
```python
# Players based on age group

import numpy as np

# Define the age groups
age_bins = [0, 20, 25, 30, np.inf]
age_labels = ['<20', '20-25', '25-30', '>30']

# Create a new column with the age group of each player
epl_df['Age Group'] = pd.cut(epl_df['Age'], bins=age_bins, labels=age_labels)

# Group the players by age group and count the number of players in each group
players_by_age = epl_df.groupby('Age Group')['Name'].count()

# Display the results
print(players_by_age)
```

```
Age Group
<20       78
20-25    186
25-30    197
>30       71
Name: Name, dtype: int64
```

OR

You can plot on pie chart with '%' & total of each group

In [54]:
```python
# Partition by players by 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)]
above30 = epl_df[epl_df['Age'] > 30]

# create an array to count the player by age grouped
x = np.array([Under20['Name'].count(), age20_25['Name'].count(), age25_30['Name'].count(), above30['Name'].count()])

# provide labels to each group
my_labels = [f"<=20 ({x[0]})", f">20 & <=25 ({x[1]})", f">25 & <=30 ({x[2]})", f">30 ({x[3]})"]

# Provide title and font size
plt.title("Total player with age group", fontsize=10)

# Define colors for each wedge in the pie chart
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']

# create pie chart
plt.pie(x, labels=my_labels, colors=colors, autopct="%.1f%%")

# display pie chart
plt.show()
```



The f-string is used to format a string with variables or expressions. In the code provided, f-strings are used to insert the respective count of each age group in the label for each group. This makes the pie chart more informative by displaying the total count of players in each age group alongside the respective percentage. The f-string format syntax is used to embed the value of a variable or expression inside a string. The syntax is as follows:

```
f"string {expression} more string {variable}"
```

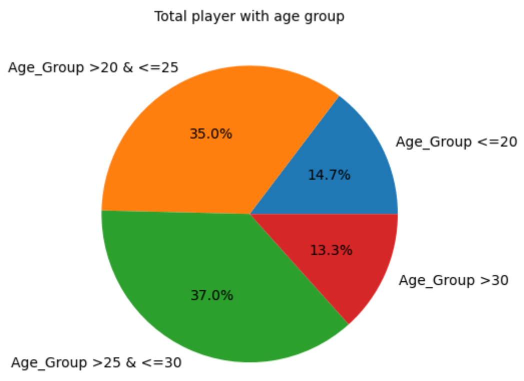
By enclosing the string inside the f" " quotation marks, the expressions and variables can be placed inside the string by enclosing them in curly braces {}.

OR

You can plot on pie chart with % of each age group

In [56]:
```python
# Partition by players by age group

# Data Frame of age- Under20
Under20 = epl_df[epl_df['Age'] <= 20]

# Data Frame of age- age20_25
age20_25 = epl_df[(epl_df['Age'] > 20) & (epl_df['Age'] <= 25)]

# Data Frame of age- age25_30
age25_30 = epl_df[(epl_df['Age'] > 25) & (epl_df['Age'] <= 30)]

# Data Frame of age- above30
above30 = epl_df[epl_df['Age'] > 30]
```

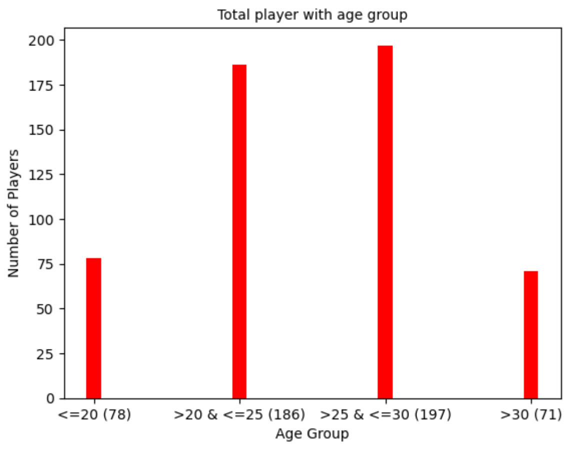
In [26]:
```python
# create a array to count the player by age grouped
x = np.array([Under20['Name'].count(),age20_25['Name'].count(),age25_30['Name'].count(),above30['Name'].count()])

# provide labels to each group
my_labels = ["Age_Group <=20", "Age_Group >20 & <=25", "Age_Group >25 & <=30", "Age_Group >30"]

# Provide title and font size
plt.title("Total player with age group", fontsize = 10)

# create pie chart
plt.pie(x, labels = my_labels, autopct = "%.1f%%")

# display pie chart
plt.show()
```



OR

You can plot on bar chart

In [27]:
```python
# Partition by players by 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)]
above30 = epl_df[epl_df['Age'] > 30]

# create an array to count the player by age grouped
x = np.array([Under20['Name'].count(), age20_25['Name'].count(), age25_30['Name'].count(), above30['Name'].count()])

# provide labels to each group
my_labels = [f"<=20 ({x[0]})", f">20 & <=25 ({x[1]})", f">25 & <=30 ({x[2]})", f">30 ({x[3]})"]

# create bar chart
plt.bar(my_labels, x, width = 0.1, color="red")

# Add a title and axis labels to the plot
plt.title("Total player with age group", fontsize=10)
plt.xlabel('Age Group')
plt.ylabel('Number of Players')

# display bar chart
plt.show()
```



Find the total unique club in EPL 2020-2021

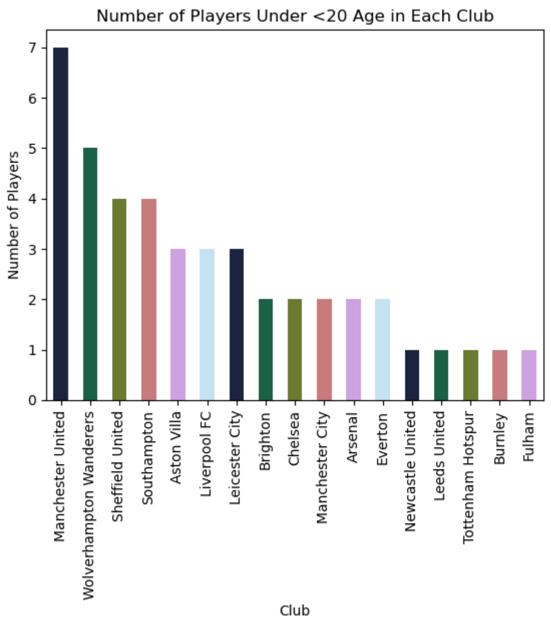
In [28]:
```
# find the total unique club


Total_unique_club = np.size(epl_df['Club'].unique())

# Print the number of unique names
print("There are",Total_unique_club,"unique Club in dataframe")
```

```
There are 20 unique Club in dataframe
```

Count the total no of players in each 'Club' whose age is <20

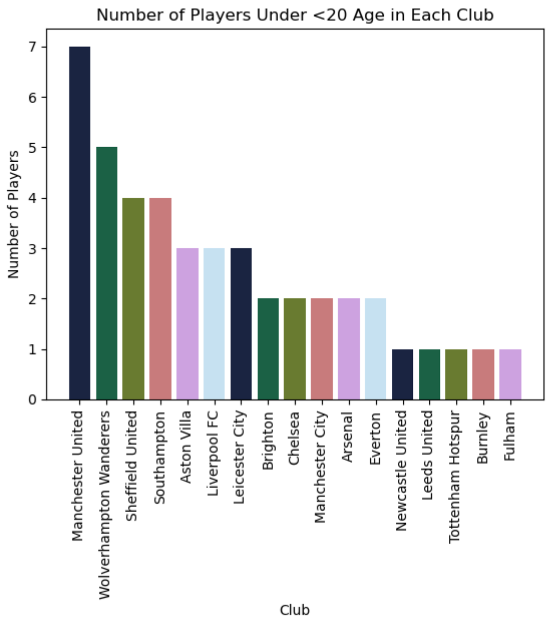
In [29]:
```
# find the total no of player in each club where age of each play is <20

# Select players under the age of <20
players_under_age20 = epl_df[epl_df['Age'] < 20]

# Count the number of players in each club for the selected age group and plot the bar chart
players_under_age20['Club'].value_counts().plot(kind = 'bar', color = sns.color_palette("cubehelix"))

# Add axis labels and a title
plt.xlabel('Club')
plt.ylabel('Number of Players')
plt.title('Number of Players Under <20 Age in Each Club')
```

Out[29]:
```
Text(0.5, 1.0, 'Number of Players Under <20 Age in Each Club')
```



OR

In [30]:
```
# Select players under the age of <20
players_under_age20 = epl_df[epl_df['Age'] < 20]

# Count the number of players under the age of 20 in each club
club_counts = players_under_age20['Club'].value_counts()

# Create a bar chart
plt.bar(x=club_counts.index, height=club_counts.values, color=sns.color_palette("cubehelix"))

# Add axis labels and a title
plt.xlabel('Club')
plt.ylabel('Number of Players')
plt.title('Number of Players Under <20 Age in Each Club')

# Rotate the x-axis labels for better readability
plt.xticks(rotation=90)

# Display the chart
plt.show()
```



Find the total player who played for 'Manchester United' Club and age is <20

In [31]:
```
# Select players who played for Manchester United and are under the age of 20
players_under_age20_Manchester = players_under_age20[players_under_age20['Club'] == 'Manchester United']

# Get the count of players under 20 who played for Manchester United
num_players = len(players_under_age20_Manchester)

# Print the result
print("Total players from Manchester United under the age of 20: ", num_players)
```

```
Total players from Manchester United under the age of 20:  7
```

OR

In [32]:
```
# find the player who played by 'Manchester United' club and age is <20

# players_under_age20 dataframe is already created above
Manchester_Under_20_Players = players_under_age20[players_under_age20["Club"]=='Manchester United']

# Print the result
Manchester_Under_20_Players
```

Out[32]:

	Name	Club	Nationality	Position	Age	Matches	Starts	Mins	Goals	Assists	...	Perc_Passes_Completed	Penalty_Goals	Penalty_Attempted	xG	xA	Yellow_Cards	Red_Cards	MinsPerMatch	GoalsP
61	Mason Greenwood	Manchester United	ENG	FW	18	31	21	1822	7	2	...	83.1	0	0	0.37	0.09	2	0	58	(
72	Brandon Williams	Manchester United	ENG	DF	19	4	2	188	0	0	...	85.7	0	0	0.05	0.01	0	0	47	(
73	Amad Diallo	Manchester United	CIV	FW	18	3	2	166	0	1	...	84.4	0	0	0.02	0.26	0	0	55	(
74	Anthony Elanga	Manchester United	SWE	FW	18	2	2	155	1	0	...	81.1	0	0	0.16	0.02	0	0	77	(
76	Shola Shoretire	Manchester United	ENG	FW	16	2	0	11	0	0	...	75.0	0	0	0.00	0.00	0	0	5	(
78	Hannibal Mejbri	Manchester United	FRA	MF	17	1	0	9	0	0	...	100.0	0	0	0.00	0.00	0	0	9	(
79	William Thomas Fish	Manchester United	ENG	DF	17	1	0	1	0	0	...	0.0	0	0	0.00	0.00	0	0	1	(

7 rows × 21 columns

Find the total player who played for 'Chelsea' Club and age is <20

In [58]:
```
# find the player who played by 'Chelsea' club and age is <20

# players_under_age20 dataframe is already created above

Chelsea_Under_20_Players = players_under_age20[players_under_age20['Club']== 'Sheffield United']

# print the result
Chelsea_Under_20_Players
```

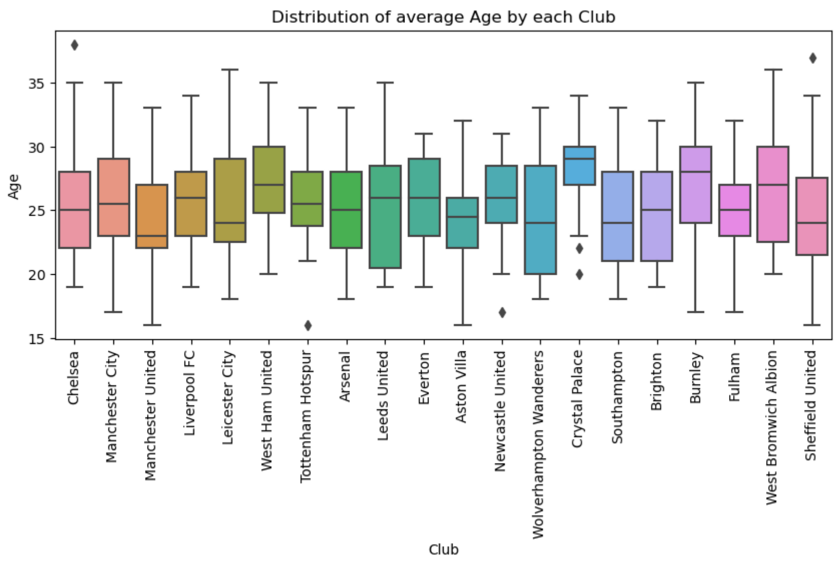
Out[58]:

	Name	Club	Nationality	Position	Age	Matches	Starts	Mins	Goals	Assists	...	Perc_Passes_Completed	Penalty_Goals	Penalty_Attempted	xG	xA	Yellow_Cards	Red_Cards	MinsPerMatch	GoalsPerM
513	Ethan Ampadu	Sheffield United	WAL	DF,MF	19	25	23	2089	0	0	...	80.5	0	0	0.01	0.04	3	0	83	
526	Daniel Jebbison	Sheffield United	ENG	FW	17	4	3	284	1	0	...	70.6	0	0	0.50	0.01	0	0	71	
530	Antwoine Hackford	Sheffield United	ENG	DF,FW	16	1	0	11	0	0	...	100.0	0	0	1.16	0.00	0	0	11	
531	Femi Seriki	Sheffield United	ENG	DF	17	1	0	1	0	0	...	-1.0	0	0	0.00	0.00	0	0	1	

4 rows × 21 columns

Find the distribution of average age of players in each club

In [34]:
```
# find the distribution of average age of players in each club

# Plot the figure size
plt.figure(figsize=(10,4))

# define the data in this we use original dataframe that 'epl_df'
sns.boxplot(x = 'Club', y = 'Age', data = epl_df)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=90)

# Add title
plt.title('Distribution of average Age by each Club')

# Add axis label
plt.xlabel('Club')
plt.ylabel('Age')


# Display the chart
plt.show()
```



OR

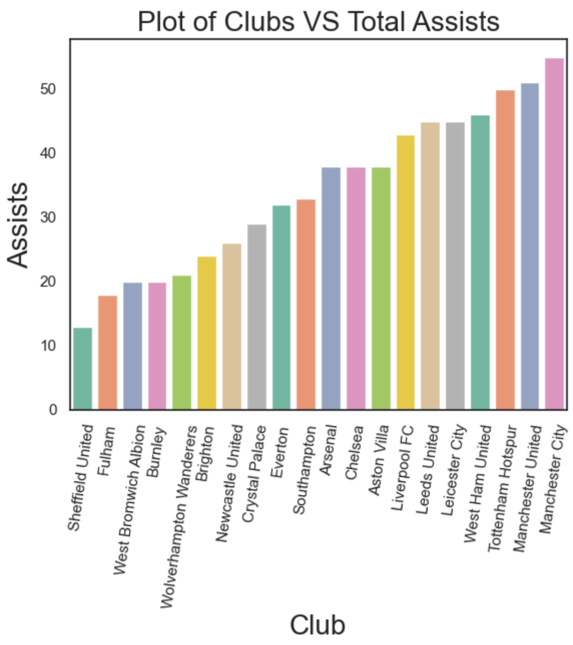
To very above see the below code

In [35]:
```
# to verify the club with average age of players

"""This code first groups the epl_df DataFrame by each club, and then calculates the total number of players in each club
using size() function. Next, it calculates the sum of ages of players in each club using sum() function, and then divides
it by the number of players to get the average age of players in each club. Finally, it sorts the resulting Series in
descending order to get the clubs with the highest average age at the top."""

# Group the DataFrame by 'Club' column and get the size of each group to count the number of players in each club
num_player = epl_df.groupby('Club').size()

# Calculate the average age of players for each club by dividing the sum of 'Age' column by the number of players in that club
data = (epl_df.groupby('Club')['Age'].sum()) /num_player

# Sort the data in descending order to get the clubs with the highest average age at the top
data.sort_values(ascending = False)
```

Out[35]:
```
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
```

Create a bar chart and calculate the total 'Assists' from each club

In [36]:
```
# Find the total assist from each club

# groups the data by 'Club' column and sums up the 'Assists' column. The results are stored in a new DataFrame called 'Assists_by_club'.
Assists_by_club = pd.DataFrame(epl_df.groupby('Club', as_index=False)['Assists'].sum())

# set seaborn theme to whitegrid with color codes
sns.set_theme(style = 'white', color_codes = True)

# create bar plot with Club on x-axis, Assists on y-axis and data from 'Assists_by_club' DataFrame
# sorted by total assists in ascending order
ax = sns.barplot(x = 'Club', y = 'Assists', data = Assists_by_club.sort_values(by = 'Assists'), palette = 'Set2')

# Add title
plt.title('Plot of Clubs VS Total Assists', fontsize = 20)

# Add axis label
ax.set_xlabel('Club', fontsize = 20)
ax.set_ylabel('Assists',fontsize = 20)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=82)

# fig size
plt.rcParams['figure.figsize'] = (8,3)
```



Display the top 10 players with the highest number of assists in the English Premier League.

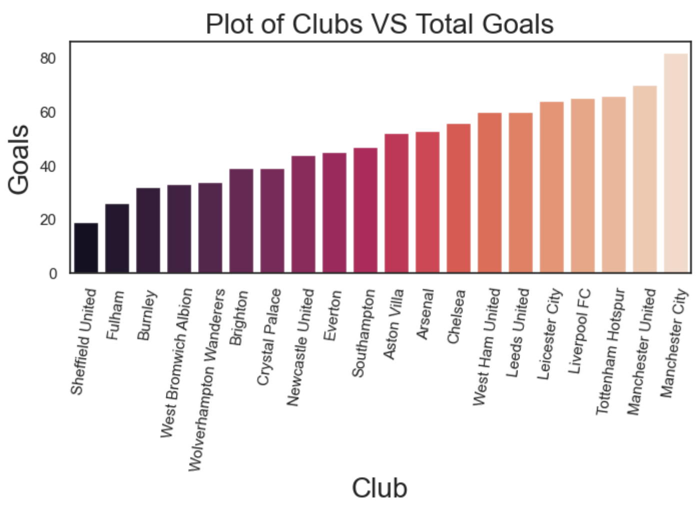
In [37]:
```
# selecting the top 10 players with the highest number of assists in the English Premier League.

""" First, it creates a new DataFrame called 'top_10_Assists' by selecting the columns 'Name', 'Age', 'Club', 'Assists', and
'Matches' from the original DataFrame 'epl_df'.

Then, it uses the 'nlargest()' function to select the top 10 players with the highest number of assists by sorting the
'Assists' column in descending order.

Finally, it displays the details of the top 10 players including their name, age, club, number of assists, and matches
played. """

# Top 10 Assists
top_10_Assists = epl_df[['Name','Age','Club','Assists', 'Matches']].nlargest(n = 10, columns = 'Assists')

# display the details
top_10_Assists
```

Out[37]:

	Name	Age	Club	Assists	Matches
162	Harry Kane	27	Tottenham Hotspur	14	35
34	Kevin De Bruyne	29	Manchester City	12	25
51	Bruno Fernandes	25	Manchester United	12	37
161	Son Heung-min	28	Tottenham Hotspur	10	37
273	Jack Grealish	24	Aston Villa	10	26
54	Marcus Rashford	22	Manchester United	9	37
110	Jamie Vardy	33	Leicester City	9	34
220	Raphael Dias Belloli	23	Leeds United	9	30
2	Timo Werner	24	Chelsea	8	35
136	Aaron Cresswell	30	West Ham United	8	36

Calculate the 'Total_goals' from each 'Club'

In [38]:
```
# Find the total goals from each club

# groups the data by 'Club' column and sums up the 'Goals' column. The results are stored in a new DataFrame called 'Goals_by_club'.
Goals_by_clubs = pd.DataFrame(epl_df.groupby('Club', as_index = False)['Goals'].sum())

# set seaborn theme to whitegrid with color codes
sns.set_theme(style = 'white', color_codes = True)

# create bar plot with Club on x-axis, Goals on y-axis and data from 'Goals_by_club' DataFrame
# sorted by total goals in ascending order
ax = sns.barplot(x = 'Club', y = 'Goals', data = Goals_by_clubs.sort_values(by='Goals'),palette='rocket')

# Add title
plt.title('Plot of Clubs VS Total Goals', fontsize = 20)

# Add axis label
ax.set_xlabel('Club', fontsize = 20)
ax.set_ylabel('Goals',fontsize = 20)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=82)

# fig size
plt.rcParams['figure.figsize'] = (8,3)
```



List the top 10 players with highest no of goals

In [39]:
```
# find the players who have highest goals

"""First, it selects the columns 'Name', 'Club', 'Goals', and 'Matches' from the DataFrame epl_df. Then, it uses the nlargest()
method to find the  10 players with the highest number of goals. The argument n specifies  the number of rows to return, and
the argument columns specifies the  column to sort by.

Finally, the resulting DataFrame containing the top 10 players is stored in the variable top_10_goals, and is printed to the
console using the print() function."""

# top 10 goals
top_10_goals = epl_df[['Name', 'Age', 'Club', 'Goals', 'Matches']].nlargest(n = 10, columns = 'Goals')

# display the output
top_10_goals
```

Out[39]:

	Name	Age	Club	Goals	Matches
162	Harry Kane	27	Tottenham Hotspur	23	35
81	Mohamed Salah	28	Liverpool FC	22	37
51	Bruno Fernandes	25	Manchester United	18	37
161	Son Heung-min	28	Tottenham Hotspur	17	37
214	Patrick Bamford	26	Leeds United	17	38
237	Dominic Calvert-Lewin	23	Everton	16	33
110	Jamie Vardy	33	Leicester City	15	34
267	Ollie Watkins	24	Aston Villa	14	37
33	İlkay Gündoğan	29	Manchester City	13	28
191	Alexandre Lacazette	29	Arsenal	13	31

In [40]:
```
# Find the top 10 'Goals' per 'match' by players
```

In [41]:
```
# find the goals per match

"""First, it selects the columns Name','Age','GoalsPerMatch','Matches','Goals from the DataFrame epl_df. Then, it uses the nlargest()
method to find the  10 players with the highest number of GoalsPerMatch. The argument n specifies  the number of rows to return, and
the argument columns specifies the  column to sort by.

Finally, the resulting DataFrame containing the top 10 players is stored in the variable top_10_goals_per_match, and is printed to the
console using the print() function. """

#top_10_goals_per_match
top_10_goals_per_match = epl_df[['Name','Age','GoalsPerMatch','Matches','Goals']].nlargest(n = 10, columns = 'GoalsPerMatch')

# display the output
top_10_goals_per_match
```

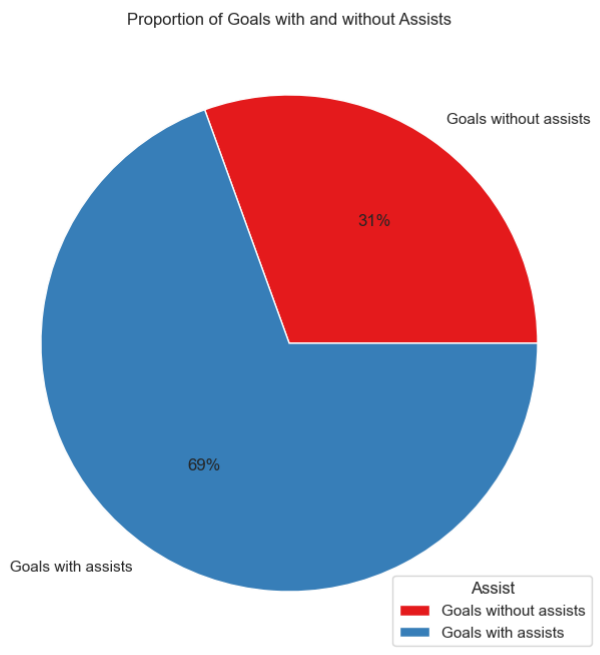
Out[41]:

	Name	Age	GoalsPerMatch	Matches	Goals
162	Harry Kane	27	0.657143	35	23
81	Mohamed Salah	28	0.594595	37	22
307	Joe Willock	20	0.571429	14	8
145	Jesse Lingard	27	0.562500	16	9
175	Gareth Bale	31	0.550000	20	11
74	Anthony Elanga	18	0.500000	2	1
51	Bruno Fernandes	25	0.486486	37	18
237	Dominic Calvert-Lewin	23	0.484848	33	16
120	Kelechi Iheanacho	23	0.480000	25	12
92	Diogo Jota	23	0.473684	19	9

Create pie chart- To displat the Goals with assist and without assist

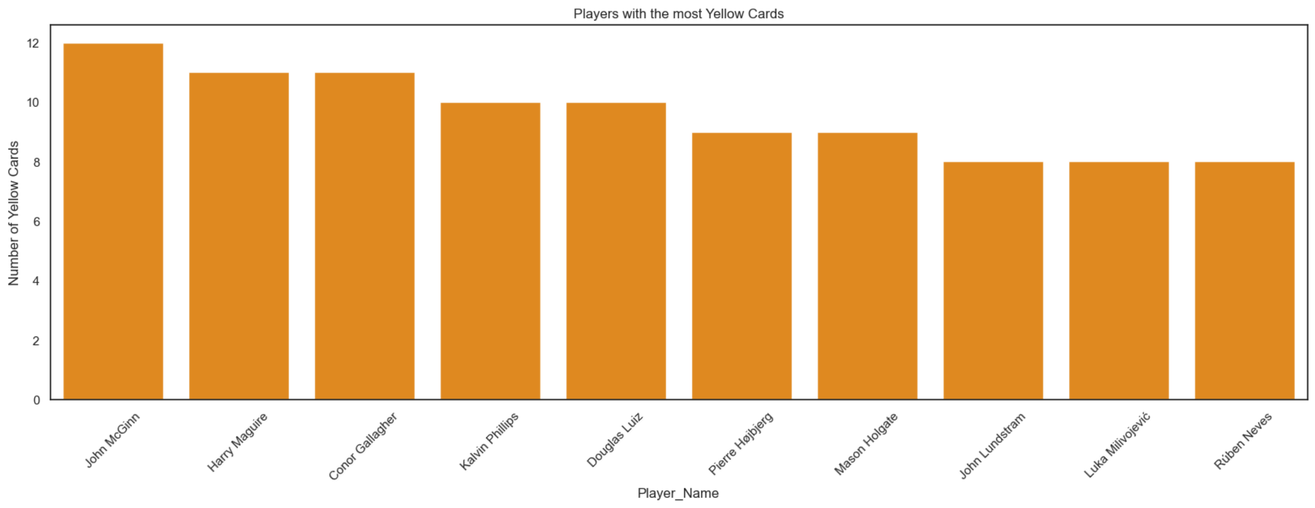
In [42]:
```
# create pie chart- Goals with assist and without assist

# Create a dataframe 'Assists' with the help of 'epl_df' by adding 'Assists'
Assists = epl_df['Assists'].sum()

# Define color palette
color = sns.color_palette('Set1')

# Total goals without assists
data = [Total_Goals - Assists, Assists]

# Add Labels
labels = ['Goals without assists', 'Goals with assists']
```

```
# figure size
plt.figure(figsize=(8,9.9))

# create pie chart
plt.pie(data, labels = labels, colors = color, autopct = '%.0f%%')

# Add title
plt.title('Proportion of Goals with and without Assists', fontsize = 12)

# Add legend
plt.legend(title='Assist', loc='lower right')

# Display the chart
plt.show()
```



In [43]:
```
# List the top 10 players with 'Yellow_Cards' and plot it into the bar chart
```

In [44]:
```
# Top 10 players with most yellow cards

# Create a dataframe 'epl_yellow_card' by sorting the original dataframe by the 'Yellow_Cards' column and selecting the top 10 rows
epl_yellow_card = epl_df.sort_values(by='Yellow_Cards', ascending=False)[:10]

# rename the 'Name' column to 'Player_Name' using the pandas rename method.
epl_yellow_card = epl_yellow_card.rename(columns={'Name': 'Player_Name'})

# Set the figure size
plt.figure(figsize=(20,6))

# Add title to the plot
plt.title("Players with the most Yellow Cards")

# Create a barplot with player names on x-axis and yellow card counts on y-axis
c = sns.barplot(x=epl_yellow_card['Player_Name'], y=epl_yellow_card['Yellow_Cards'], label='Players', color='darkorange')

# Add y-axis label
plt.ylabel('Number of Yellow Cards')

# Rotate the x-axis labels for better readability
c.set_xticklabels(c.get_xticklabels(), rotation=45)

# Display the bar chart
plt.show()
```



In [ ]: