Descriptive Analytics Project (Football Player Performance)

May 7, 2025

1 IMPORTING NECESSARY LIBRARIES

[3]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

2 IMPORTING DATASET

[5]:
```
data = pd.read_csv('Football.csv', encoding='ISO-8859-1')
df = pd.DataFrame(data)
df.head(10)
```

[5]:
```
      Id                Name     Season  Weight(kg)  Height(cm)  Age  \
0  14705     Aaron Cresswell  2022-2023        66.0       170.0   33
1  14705     Aaron Cresswell  2023-2024        66.0       170.0   34
2  14705     Aaron Cresswell  2024-2025        66.0       170.0   35
3  14634      Aaron Ramsdale  2022-2023        87.0       188.0   25
4  14634      Aaron Ramsdale  2023-2024        87.0       188.0   26
5  14634      Aaron Ramsdale  2024-2025        87.0       191.0   26
6  14634      Aaron Ramsdale  2024-2025        87.0       191.0   26
7  14710  Aaron Wan-Bissaka  2022-2023        72.0       183.0   25
8  14710  Aaron Wan-Bissaka  2023-2024        72.0       183.0   26
9  14710  Aaron Wan-Bissaka  2024-2025        72.0       183.0   27

  Citizenship                Team  Jersey    Position  …  OffSides  \
0     England     West Ham United       3    Defender  …       3.0
1     England     West Ham United       3    Defender  …       0.0
2     England     West Ham United       3    Defender  …       0.0
3     England             Arsenal       1  Goalkeeper  …       NaN
4     England             Arsenal       1  Goalkeeper  …       0.0
5     England         Southampton      30  Goalkeeper  …       2.0
6     England             Arsenal       1  Goalkeeper  …       0.0
7     England   Manchester United      29    Defender  …       2.0
8     England   Manchester United      29    Defender  …       2.0
9     England     West Ham United      29    Defender  …       1.0
```

	YellowCards	RedCards	GoalAssists	ShotsOnTarget	TotalShots	TotalGoals	\
0	3	0	1	1	9	0	
1	1	0	0	0	0	0	
2	1	0	0	0	1	0	
3	1	0	0	0	0	0	
4	0	0	0	0	0	0	
5	1	0	0	0	0	0	
6	0	0	0	0	0	0	
7	2	0	0	2	10	0	
8	4	0	2	1	3	0	
9	0	0	0	3	11	2	

	GoalsConceded	ShotsFaced	UpdateTime
0	31	0	2023-07-30T19:56:55Z
1	10	0	5-20-24 16:23
2	0	0	01-05-2025 05:43
3	42	290	2023-07-30T19:56:55Z
4	5	51	5-20-24 16:23
5	27	200	01-05-2025 05:43
6	0	0	10-14-24 06:59
7	10	0	2023-07-30T19:56:59Z
8	34	0	5-20-24 16:23
9	35	0	01-05-2025 05:43

[10 rows x 27 columns]

3 DATASET DESCRIPTION

- **Id**: A unique identifier for each player.
- **Name**: The name of the football player.
- **Season**: The football season during which the data was recorded (e.g., 2022-2023, 2023-2024).
- **Weight(kg)**: The weight of the player in kilograms.
- **Height(cm)**: The height of the player in centimeters.
- **Age**: The age of the player during the respective season.
- **Citizenship**: The country of citizenship of the player.
- **Team**: The football team the player is associated with.
- **Jersey**: The jersey number of the player.

- **Position**: The position the player occupies on the field (e.g., Goalkeeper, Defender, Midfielder, Forward).
- **Appearances**: Total games played by the player during the season.
- **GoalAssists**: Total assists made by the player in the given season.
- **ShotsOnTarget**: Total shots on target made by the player, i.e., shots that would have resulted in goals if not for the goalkeeper's intervention.
- **TotalShots**: Total number of shots taken by the player during the season.
- **OffSides**: Total number of offsides committed by the player during the season.
- **YellowCards**: Total number of yellow cards received by the player during the season.
- **RedCards**: Total number of red cards received by the player during the season.
- **ShotsFaced**: Total number of shots faced by the player (relevant for goalkeepers).
- **GoalsConceded**: Total number of goals conceded by the player during the season (relevant for goalkeepers).
- **TotalGoals**: Total number of goals scored by the player in the given season.
- **TotalPlayTime**: Total number of minutes played by the player during the season.
- **AveragePlayTime**: Average number of minutes the player spent on the field per match.
- **OwnGoals**: The number of goals scored by the player on his own team.
- **FoulsCommitted**: Total number of fouls committed by the player during the season.
- **FoulsSuffered**: Total number of fouls suffered (committed against the player) during the season.
- **SubIns**: Total number of times the player was subbed in in place of another player.
- **UpdateTime**: The timestamp indicating when the data was last updated.

[8]:
```
# Shape or Size of the Dataset
df.shape
```

[8]: (800, 27)

[9]:
```
# Description of the dataset(numerical columns)
df.describe()
```

[9]:

```
                 Id  Weight(kg)  Height(cm)         Age      Jersey  \
count    800.000000  771.000000  793.000000  800.000000  800.000000
mean   14722.728750   75.252918  182.493064   26.955000   17.768750
std     1722.640933    7.614723    6.992078    4.343565   13.822503
min    14115.000000   54.000000  163.000000   17.000000    0.000000
25%    14268.250000   69.000000  178.000000   24.000000    7.000000
50%    14469.000000   74.000000  183.000000   27.000000   16.000000
75%    14599.250000   81.000000  188.000000   30.000000   24.000000
max    25076.000000   94.000000  201.000000   39.000000   82.000000

       Appearances     SubIns  Total PlayTime (min)  AveragePlayTime (min)  \
count   800.000000  800.00000            800.000000             800.000000
mean     20.581250    4.44250           1537.527500              67.388750
std      11.152191    4.78098           1026.360233              26.391395
min       0.000000    0.00000              0.000000               0.000000
25%      12.750000    1.00000            712.500000              54.000000
50%      20.000000    3.00000           1461.000000              75.000000
75%      31.000000    7.00000           2265.250000              88.000000
max      38.000000   27.00000           3745.000000             103.000000

       FoulsCommitted  …   OwnGoals    OffSides  YellowCards    RedCards  \
count      800.000000  …  800.00000  781.000000   800.000000  800.000000
mean        15.033750  …    0.06750    2.513444     2.890000    0.080000
std         13.102947  …    0.27025    2.840889     2.612265    0.289318
min          0.000000  …    0.00000    0.000000     0.000000    0.000000
25%          4.000000  …    0.00000    0.000000     1.000000    0.000000
50%         12.000000  …    0.00000    2.000000     2.000000    0.000000
75%         22.250000  …    0.00000    4.000000     4.000000    0.000000
max         66.000000  …    2.00000   20.000000    13.000000    2.000000

       GoalAssists  ShotsOnTarget  TotalShots  TotalGoals  GoalsConceded  \
count   800.000000     800.000000  800.000000  800.000000     800.000000
mean      1.815000       6.905000   20.278750    2.422500      21.678750
std       2.533535       9.170665   23.124435    3.965306      15.412981
min       0.000000       0.000000    0.000000    0.000000       0.000000
25%       0.000000       1.000000    3.000000    0.000000      10.000000
50%       1.000000       3.000000   12.000000    1.000000      20.000000
75%       3.000000       9.000000   31.000000    3.000000      31.000000
max      16.000000      60.000000  125.000000   36.000000      67.000000

       ShotsFaced
count  800.000000
mean    14.676250
std     64.790614
min      0.000000
25%      0.000000
50%      0.000000
```

```
75%      0.000000
max    506.000000

[8 rows x 21 columns]
```

[10]:
```
# Missing values in each column of the dataset
missing=pd.DataFrame(df.isnull().sum())
missing
```

[10]:
```
                         0
Id                       0
Name                     0
Season                   0
Weight(kg)              29
Height(cm)               7
Age                      0
Citizenship              0
Team                     0
Jersey                   0
Position                 0
Appearances              0
SubIns                   0
Total PlayTime (min)     0
AveragePlayTime (min)    0
FoulsCommitted           0
FoulsSuffered            0
OwnGoals                 0
OffSides                19
YellowCards              0
RedCards                 0
GoalAssists              0
ShotsOnTarget            0
TotalShots               0
TotalGoals               0
GoalsConceded            0
ShotsFaced               0
UpdateTime               0
```

[11]:
```
# Buiding a feature matrix function to better understand the dataset and to␣
 ↪call whenver needed

def feature_matrix(df):
    features = []
    count = []
    dtypes=[]
    unique = []
    missing  = []
```

```
    missing_percentage = []
    for i in df.columns:
        features.append(i)
        count.append((df[i].shape[0]))
        dtypes.append(df[i].dtypes)
        unique.append(len(df[i].unique()))
        missing.append(df[i].isnull().sum())
        missing_percentage.append(f"{(df[i].isnull().sum())/df.shape[0]*100:.
↪2f} %")

    dataFrame = pd.DataFrame({'Features':features,
                              'Count':count,
                              'Dtypes':dtypes,
                              'Unique':unique,
                              'Missing':missing,
                              'Missing Percentage':missing_percentage})
    return dataFrame
```

[12]:
```
feature_matrix(df)
```

[12]:

	Features	Count	Dtypes	Unique	Missing	Missing Percentage
0	Id	800	int64	262	0	0.00 %
1	Name	800	object	262	0	0.00 %
2	Season	800	object	3	0	0.00 %
3	Weight(kg)	800	float64	31	29	3.62 %
4	Height(cm)	800	float64	18	7	0.88 %
5	Age	800	int64	23	0	0.00 %
6	Citizenship	800	object	45	0	0.00 %
7	Team	800	object	22	0	0.00 %
8	Jersey	800	int64	57	0	0.00 %
9	Position	800	object	4	0	0.00 %
10	Appearances	800	int64	39	0	0.00 %
11	SubIns	800	int64	26	0	0.00 %
12	Total PlayTime (min)	800	int64	685	0	0.00 %
13	AveragePlayTime (min)	800	int64	95	0	0.00 %
14	FoulsCommitted	800	int64	55	0	0.00 %
15	FoulsSuffered	800	int64	69	0	0.00 %
16	OwnGoals	800	int64	3	0	0.00 %
17	OffSides	800	float64	19	19	2.38 %
18	YellowCards	800	int64	14	0	0.00 %
19	RedCards	800	int64	3	0	0.00 %
20	GoalAssists	800	int64	16	0	0.00 %
21	ShotsOnTarget	800	int64	46	0	0.00 %
22	TotalShots	800	int64	93	0	0.00 %
23	TotalGoals	800	int64	24	0	0.00 %
24	GoalsConceded	800	int64	66	0	0.00 %
25	ShotsFaced	800	int64	55	0	0.00 %

```
26              UpdateTime    800    object       38         0               0.00 %
```

4 DATASET CLEANING

[14]:
```python
# Dropping the unnecessary columns
df.drop('UpdateTime', axis=1, inplace=True)
```

[15]:
```python
# Impute the missing values in the Weight(kg), Height(cm) and OffSides columns␣
↪using median(as they are numerical columns)

df['Weight(kg)'].fillna(df['Weight(kg)'].median(), inplace=True)
df['Weight(kg)'] = df['Weight(kg)'].astype(int)

df['Height(cm)'].fillna(df['Height(cm)'].median(), inplace=True)
df['Height(cm)']=df['Height(cm)'].astype(int)
```

[16]:
```python
# Checking the missing values in the dataset after imputation

feature_matrix(df)
```

[16]:

	Features	Count	Dtypes	Unique	Missing	Missing Percentage
0	Id	800	int64	262	0	0.00 %
1	Name	800	object	262	0	0.00 %
2	Season	800	object	3	0	0.00 %
3	Weight(kg)	800	int32	30	0	0.00 %
4	Height(cm)	800	int32	17	0	0.00 %
5	Age	800	int64	23	0	0.00 %
6	Citizenship	800	object	45	0	0.00 %
7	Team	800	object	22	0	0.00 %
8	Jersey	800	int64	57	0	0.00 %
9	Position	800	object	4	0	0.00 %
10	Appearances	800	int64	39	0	0.00 %
11	SubIns	800	int64	26	0	0.00 %
12	Total PlayTime (min)	800	int64	685	0	0.00 %
13	AveragePlayTime (min)	800	int64	95	0	0.00 %
14	FoulsCommitted	800	int64	55	0	0.00 %
15	FoulsSuffered	800	int64	69	0	0.00 %
16	OwnGoals	800	int64	3	0	0.00 %
17	OffSides	800	float64	19	19	2.38 %
18	YellowCards	800	int64	14	0	0.00 %
19	RedCards	800	int64	3	0	0.00 %
20	GoalAssists	800	int64	16	0	0.00 %
21	ShotsOnTarget	800	int64	46	0	0.00 %
22	TotalShots	800	int64	93	0	0.00 %
23	TotalGoals	800	int64	24	0	0.00 %
24	GoalsConceded	800	int64	66	0	0.00 %

```
25             ShotsFaced    800     int64      55          0              0.00 %
```

[17]:
```
# Filling in the missing values in the OffSides column with 0 as it missing␣
↪values may represent that the player has no offsides

df['OffSides'].fillna(0, inplace=True)
```

[18]:
```
feature_matrix(df)
```

[18]:

	Features	Count	Dtypes	Unique	Missing	Missing Percentage
0	Id	800	int64	262	0	0.00 %
1	Name	800	object	262	0	0.00 %
2	Season	800	object	3	0	0.00 %
3	Weight(kg)	800	int32	30	0	0.00 %
4	Height(cm)	800	int32	17	0	0.00 %
5	Age	800	int64	23	0	0.00 %
6	Citizenship	800	object	45	0	0.00 %
7	Team	800	object	22	0	0.00 %
8	Jersey	800	int64	57	0	0.00 %
9	Position	800	object	4	0	0.00 %
10	Appearances	800	int64	39	0	0.00 %
11	SubIns	800	int64	26	0	0.00 %
12	Total PlayTime (min)	800	int64	685	0	0.00 %
13	AveragePlayTime (min)	800	int64	95	0	0.00 %
14	FoulsCommitted	800	int64	55	0	0.00 %
15	FoulsSuffered	800	int64	69	0	0.00 %
16	OwnGoals	800	int64	3	0	0.00 %
17	OffSides	800	float64	18	0	0.00 %
18	YellowCards	800	int64	14	0	0.00 %
19	RedCards	800	int64	3	0	0.00 %
20	GoalAssists	800	int64	16	0	0.00 %
21	ShotsOnTarget	800	int64	46	0	0.00 %
22	TotalShots	800	int64	93	0	0.00 %
23	TotalGoals	800	int64	24	0	0.00 %
24	GoalsConceded	800	int64	66	0	0.00 %
25	ShotsFaced	800	int64	55	0	0.00 %

5 FEATURE ENGINEERING

[20]:
```
# Making new attributes using the old attributes in such a way that they donot␣
↪contain any missing values

df['GoalsPerMatch']=df.apply(lambda row:0 if row['TotalGoals']==0 else␣
↪row['TotalGoals']/row['Appearances'], axis=1)
df['GoalsPerMatch']=df['GoalsPerMatch'].round(2)
```

```
df['ShotAccuracy']=df.apply(lambda row:0 if row['TotalShots']==0 or␣
↪row['ShotsOnTarget']==0 else row['ShotsOnTarget']/row['TotalShots'], axis=1)
df['ShotAccuracy']=df['ShotAccuracy'].round(2)
```

[21]:

```
# Creating a new column to categorize the players based on their age

def age_categorize(a):
    if a<25:
        return 'Young'
    elif a>=25 and a<=33:
        return 'Mid-aged'
    else:
        return 'Old'

df['AgeGroup']=df['Age'].apply(age_categorize)
```

[22]:

```
# Feature to measure the Goalkeeper's accuracy
# The accuracy is calculated as the ratio of goals conceded to shots faced. If␣
↪the shots faced is 0, we set the accuracy to 0
# as the player may not be a goalkeeper.

df['GoalKeeperAccuracy']=df.apply(lambda row:0 if row['ShotsFaced']==0 else␣
↪row['GoalsConceded']/row['ShotsFaced'], axis=1)
df['GoalKeeperAccuracy']=df['GoalKeeperAccuracy'].round(2)
```

[23]:

```
# features for denfending players

df['FoulsInvolvement']=df.apply(lambda row:0 if row['FoulsCommitted']==0 and␣
↪row['FoulsSuffered']==0 else row['FoulsCommitted']+row['FoulsSuffered'],␣
↪axis=1)

df['TotalCards']=df.apply(lambda row:0 if row['RedCards']==0 and␣
↪row['YellowCards']==0 else row['RedCards']+row['YellowCards'], axis=1)
```

[24]:

```
feature_matrix(df)
```

[24]:

	Features	Count	Dtypes	Unique	Missing	Missing Percentage
0	Id	800	int64	262	0	0.00 %
1	Name	800	object	262	0	0.00 %
2	Season	800	object	3	0	0.00 %
3	Weight(kg)	800	int32	30	0	0.00 %
4	Height(cm)	800	int32	17	0	0.00 %
5	Age	800	int64	23	0	0.00 %
6	Citizenship	800	object	45	0	0.00 %
7	Team	800	object	22	0	0.00 %
8	Jersey	800	int64	57	0	0.00 %
9	Position	800	object	4	0	0.00 %

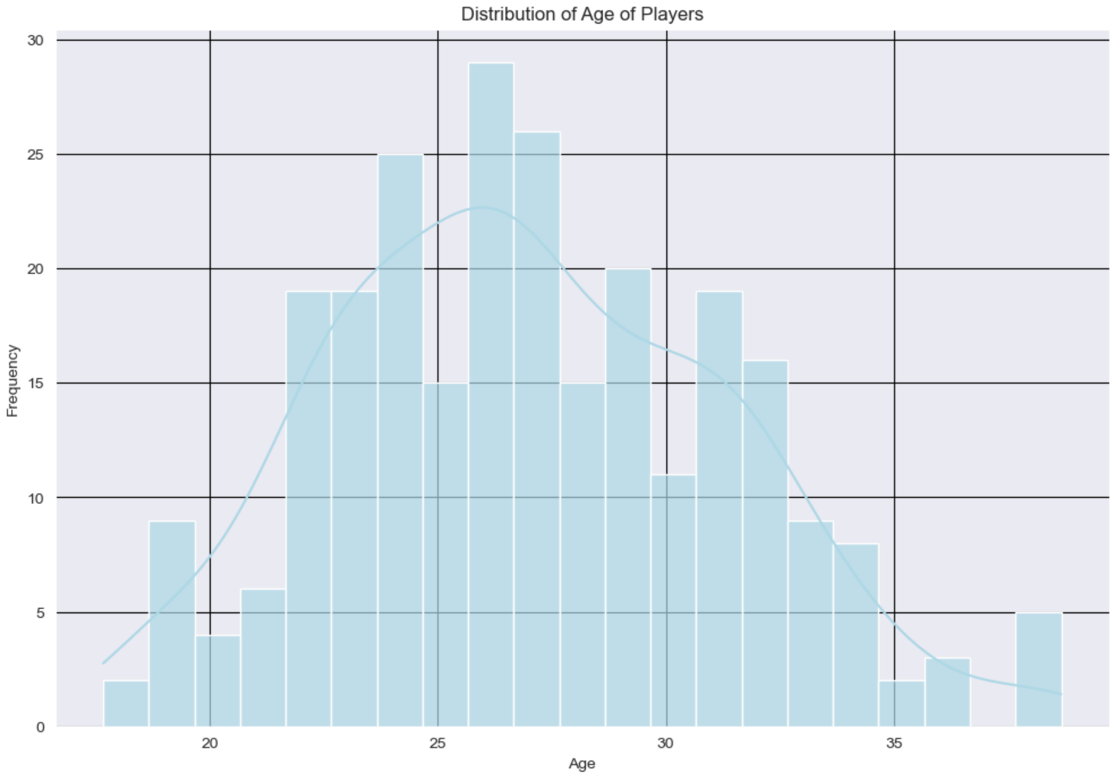
```
10          Appearances  800    int64   39    0       0.00 %
11               SubIns  800    int64   26    0       0.00 %
12    Total PlayTime (min)  800    int64  685    0       0.00 %
13  AveragePlayTime (min)  800    int64   95    0       0.00 %
14       FoulsCommitted  800    int64   55    0       0.00 %
15        FoulsSuffered  800    int64   69    0       0.00 %
16             OwnGoals  800    int64    3    0       0.00 %
17             OffSides  800  float64   18    0       0.00 %
18          YellowCards  800    int64   14    0       0.00 %
19             RedCards  800    int64    3    0       0.00 %
20          GoalAssists  800    int64   16    0       0.00 %
21        ShotsOnTarget  800    int64   46    0       0.00 %
22           TotalShots  800    int64   93    0       0.00 %
23           TotalGoals  800    int64   24    0       0.00 %
24        GoalsConceded  800    int64   66    0       0.00 %
25           ShotsFaced  800    int64   55    0       0.00 %
26        GoalsPerMatch  800  float64   62    0       0.00 %
27         ShotAccuracy  800  float64   59    0       0.00 %
28             AgeGroup  800   object    3    0       0.00 %
29   GoalKeeperAccuracy  800  float64   23    0       0.00 %
30      FoulsInvolvement  800    int64  108    0       0.00 %
31           TotalCards  800    int64   14    0       0.00 %
```

6 VISUALISATIONS

[26]:
```python
# improting the libraries for visualization

import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

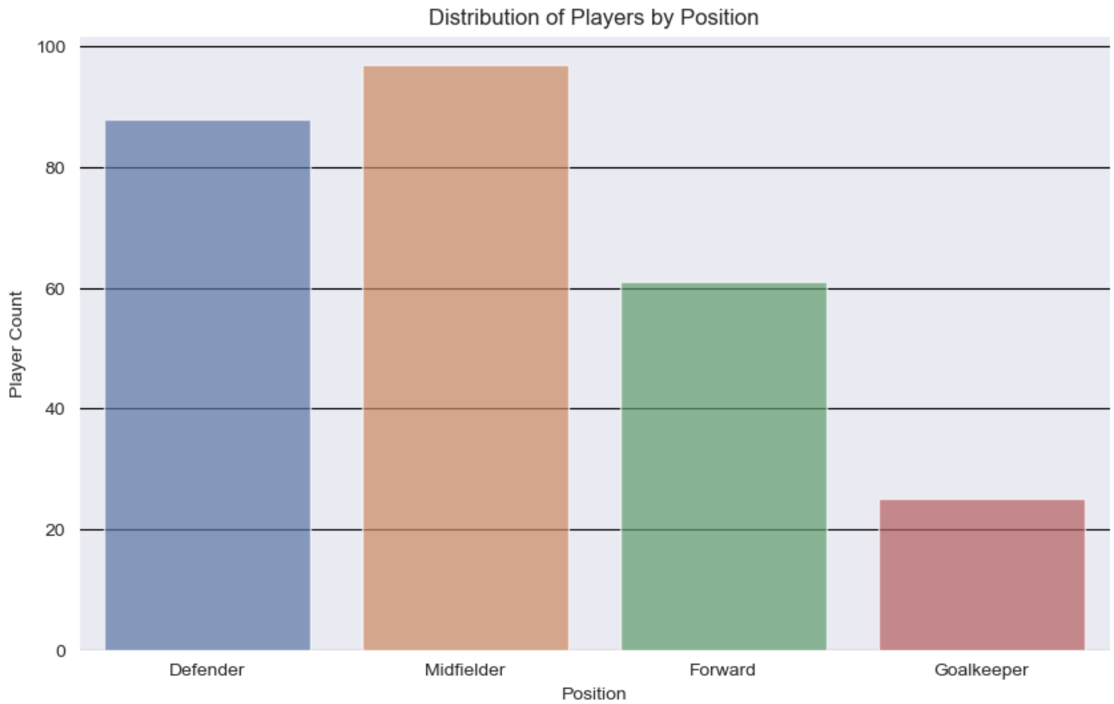
6.1 Histogram Plot

[28]:
```python
grouped=df.groupby('Id')['Age'].mean().reset_index()

sns.set_style('darkgrid',{'grid.color':'black'})
plt.figure(figsize=(12,8))
sns.histplot(grouped['Age'],bins=21,kde=True,color='lightblue',alpha=0.7)
plt.title('Distribution of Age of Players')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



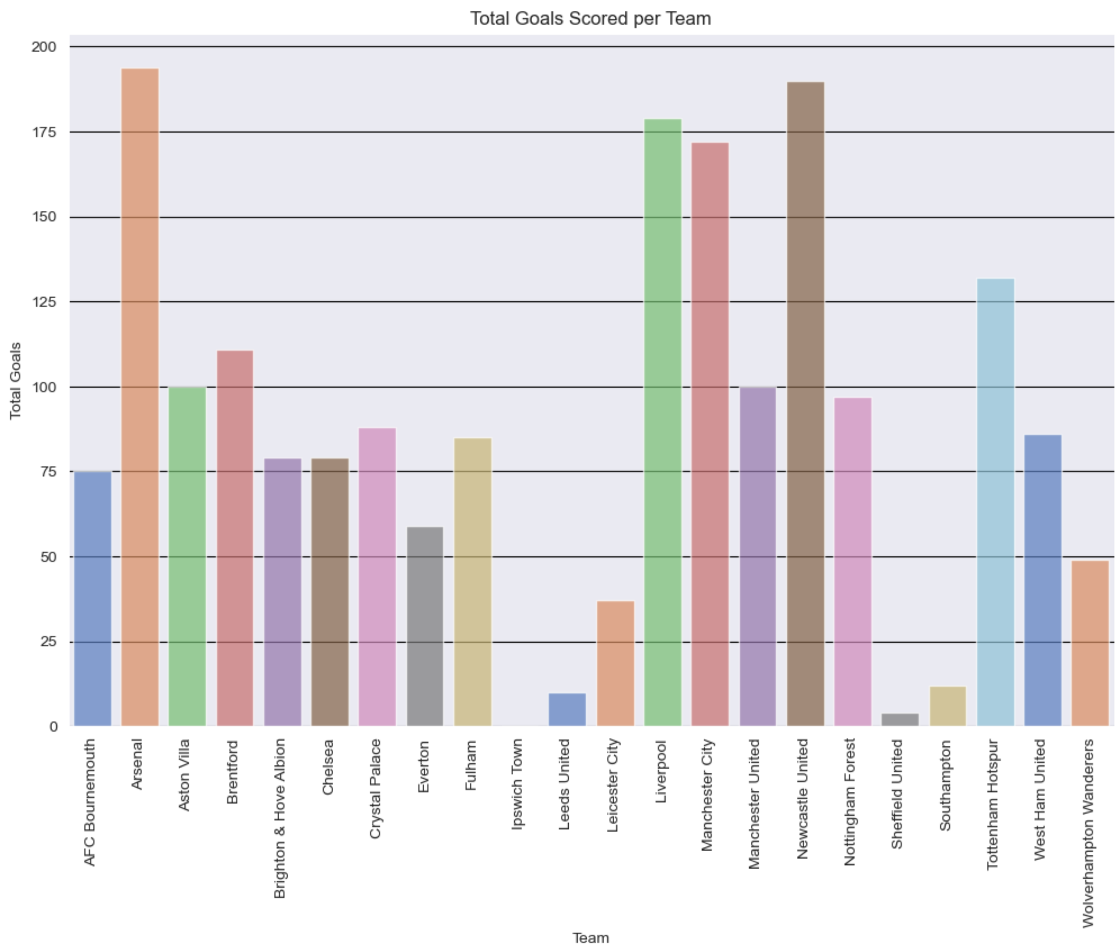
6.2 Count Plot

[30]:
```
grouped=df.groupby('Id')['Position'].value_counts().reset_index()

plt.figure(figsize=(10,6))
sns.countplot(x='Position',data=grouped,palette='deep',alpha=0.7)
plt.title('Distribution of Players by Position')
plt.xlabel('Position')
plt.ylabel('Player Count')
plt.show()
```

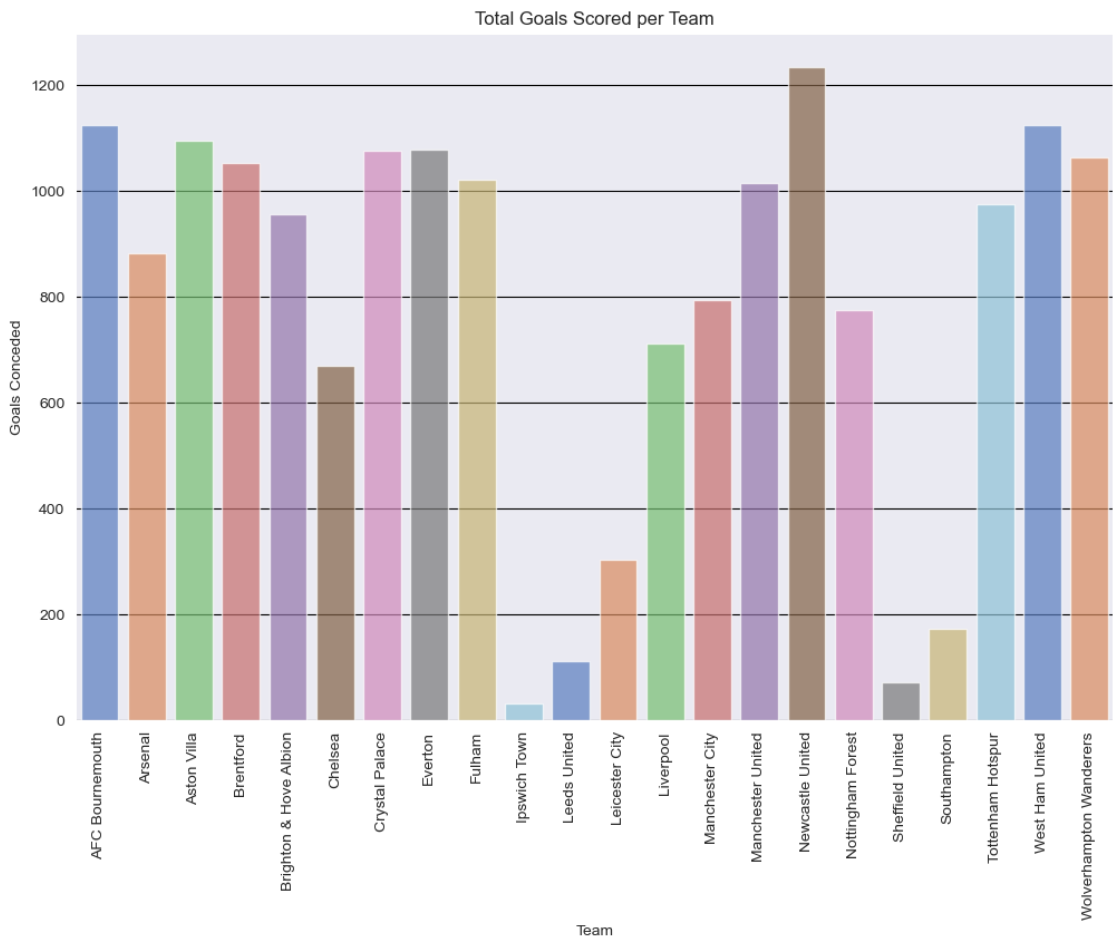


6.3 Bar Plots

[32]:
```
team=df.groupby('Team')['TotalGoals'].sum().reset_index()

plt.figure(figsize=(12,8))
sns.barplot(x='Team',y='TotalGoals',data=team,palette='muted',alpha=0.7)
plt.title('Total Goals Scored per Team')
plt.xlabel('Team')
plt.ylabel('Total Goals')
plt.xticks(rotation=90)
plt.show()
```

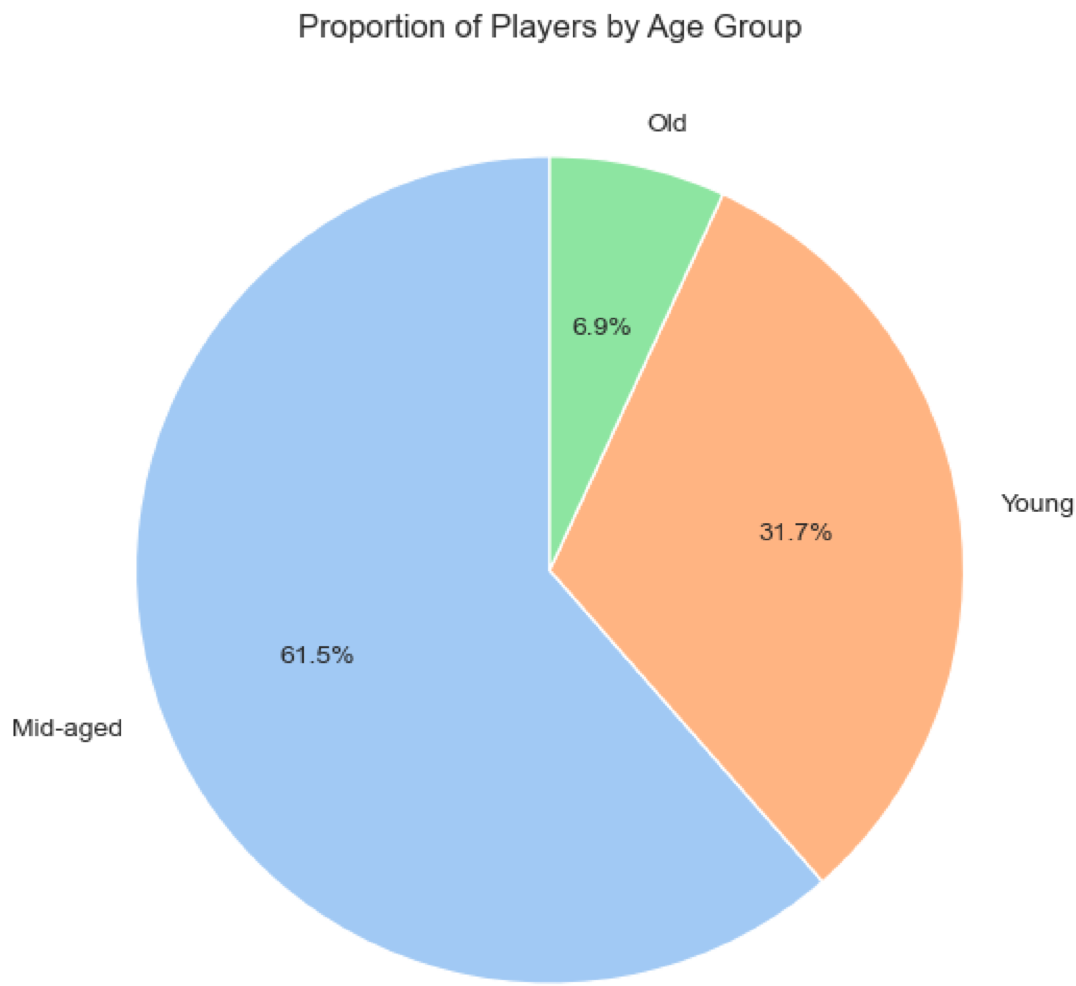


[33]:
```
team=df.groupby('Team')['GoalsConceded'].sum().reset_index()

plt.figure(figsize=(12,8))
sns.barplot(x='Team',y='GoalsConceded',data=team,palette='muted',alpha=0.7)
plt.title('Total Goals Scored per Team')
plt.xlabel('Team')
plt.ylabel('Goals Conceded')
plt.xticks(rotation=90)
plt.show()
```



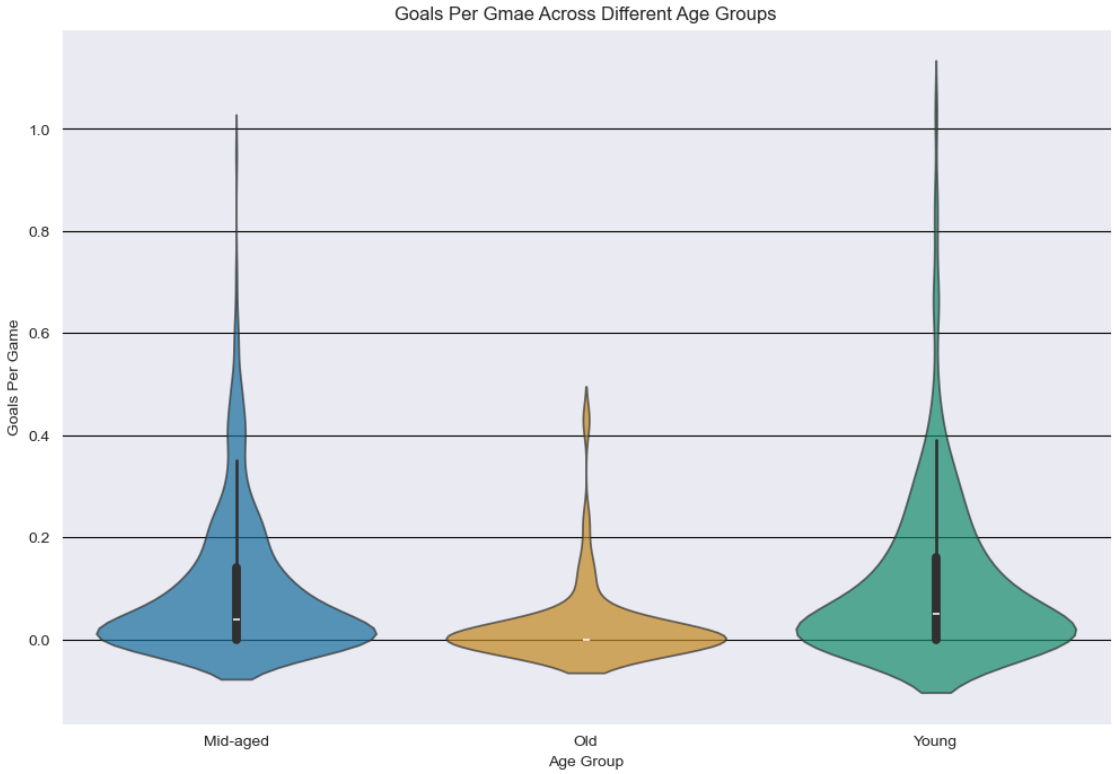
6.4 Pie Chart

[35]:
```
grouped=df.groupby('Id').agg({
    'AgeGroup': lambda x: x.mode()[0]
}).reset_index()

age=grouped['AgeGroup'].value_counts()

plt.figure(figsize=(7, 7))
age.plot.pie(autopct='%1.1f%%', startangle=90, colors=sns.
 ↪color_palette('pastel', n_colors=len(age)))
plt.title('Proportion of Players by Age Group')
plt.ylabel('')
plt.show()
```



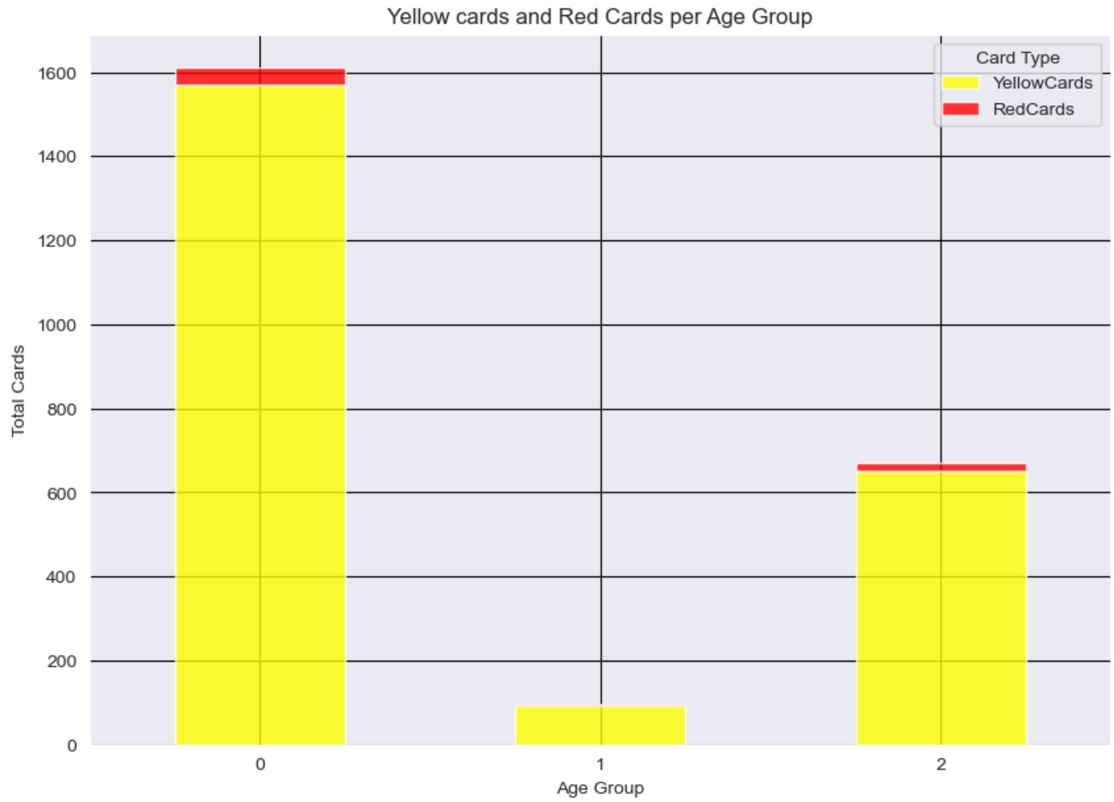
6.5 Violin Plot

[37]:
```
plt.figure(figsize=(12, 8))
sns.
 ↪violinplot(x=df['AgeGroup'],y=df['GoalsPerMatch'],palette='colorblind',alpha=0.
 ↪7)
plt.title('Goals Per Gmae Across Different Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Goals Per Game')
plt.show()
```



6.6 Stacked Bar Plot

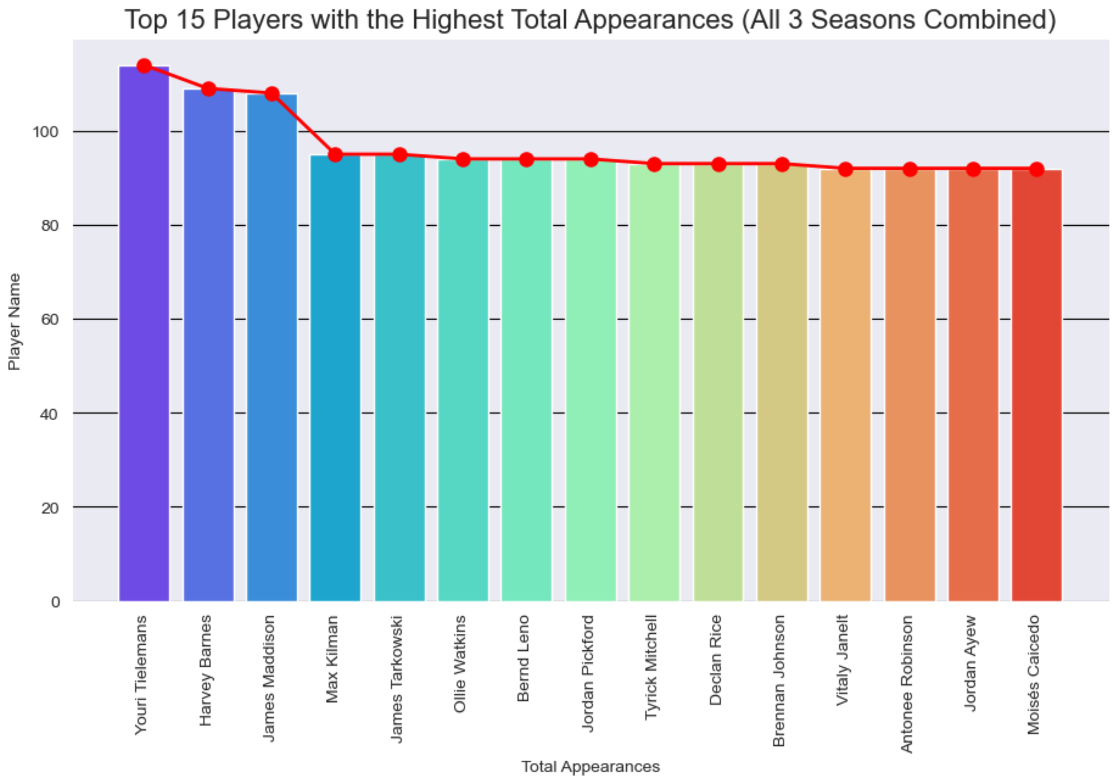
[39]:
```
grouped=df.groupby('AgeGroup')[['YellowCards','RedCards']].sum().reset_index()

grouped.
 ↪plot(kind='bar',stacked=True,figsize=(10,7),color=['yellow','red'],alpha=0.8)
plt.title('Yellow cards and Red Cards per Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Cards')
plt.xticks(rotation=0)
plt.legend(title='Card Type',loc='upper right')
plt.show()
```



6.7 Combined Plot

[41]:
```
appearances=df[['Name','Season','Appearances']]

combined_app=appearances.groupby('Name')['Appearances'].sum().reset_index()

combined_app=combined_app.sort_values(by='Appearances',ascending=False).head(15)

plt.figure(figsize=(11,6))
sns.
  ↪barplot(x='Name',y='Appearances',data=combined_app,palette='rainbow',label='Appearances')
plt.plot(combined_app['Name'], combined_app['Appearances'], color='red',␣
  ↪marker='o', markersize=8, linestyle='-', linewidth=2, label='Trend Line')
plt.title('Top 15 Players with the Highest Total Appearances (All 3 Seasons␣
  ↪Combined)',fontsize=16)
plt.xlabel('Total Appearances')
plt.ylabel('Player Name')
plt.xticks(rotation=90)
plt.show()

combined_app.reset_index(drop=True,inplace=True)
```

```
combined_app
```

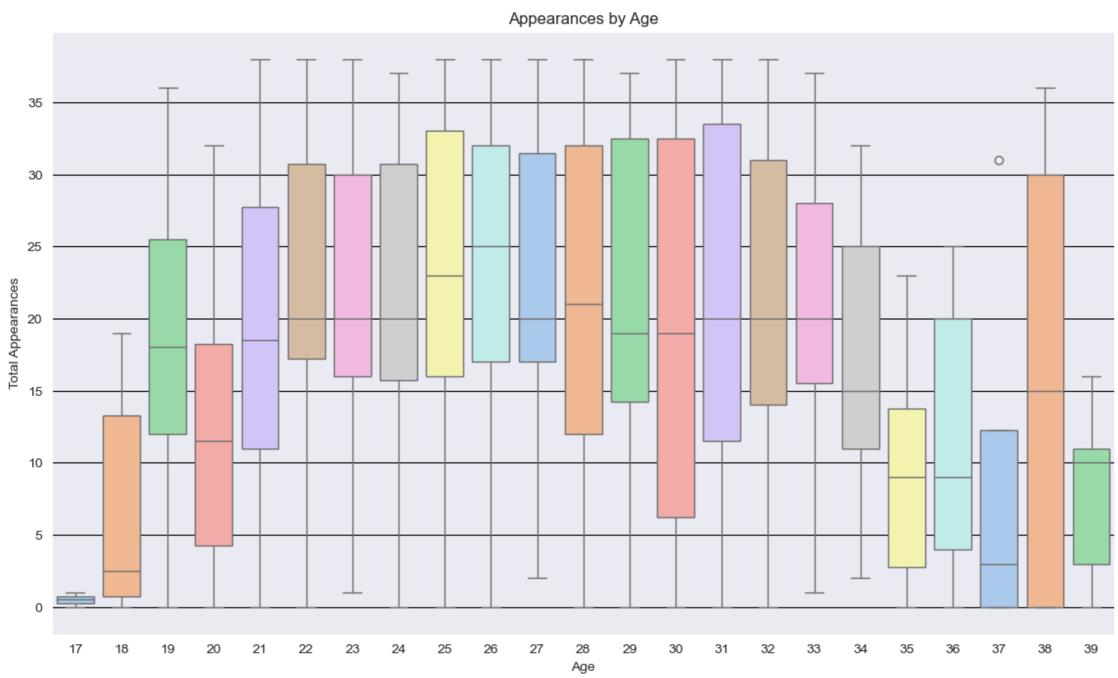


[41]:

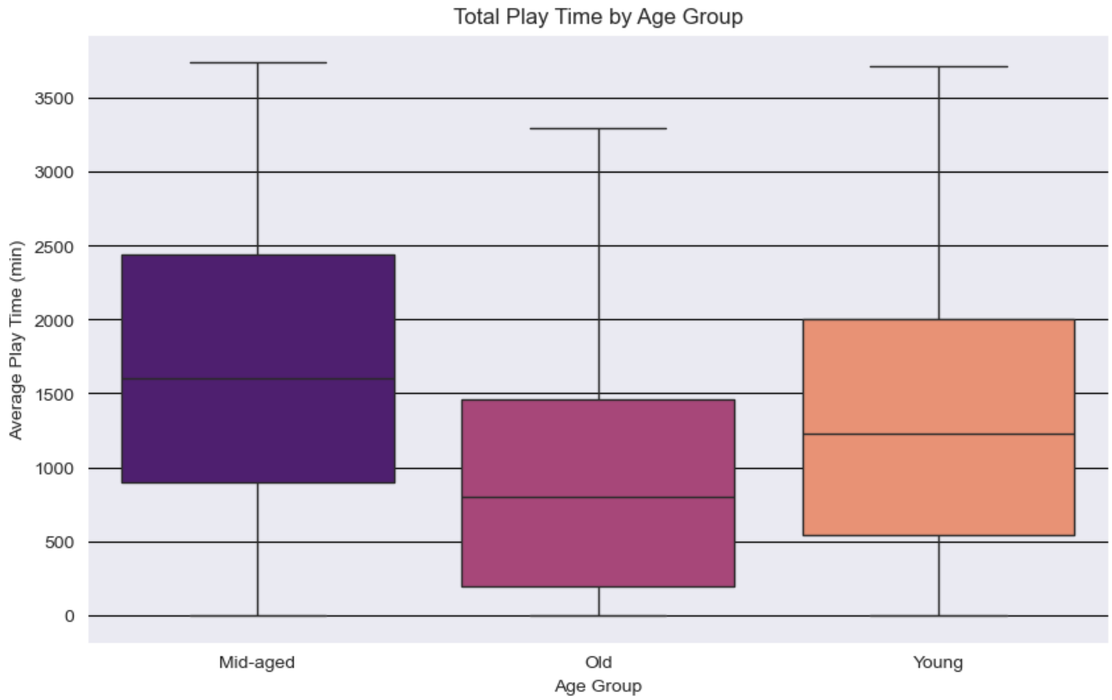
	Name	Appearances
0	Youri Tielemans	114
1	Harvey Barnes	109
2	James Maddison	108
3	Max Kilman	95
4	James Tarkowski	95
5	Ollie Watkins	94
6	Bernd Leno	94
7	Jordan Pickford	94
8	Tyrick Mitchell	93
9	Declan Rice	93
10	Brennan Johnson	93
11	Vitaly Janelt	92
12	Antonee Robinson	92
13	Jordan Ayew	92
14	Moisés Caicedo	92

6.8 Box Plots

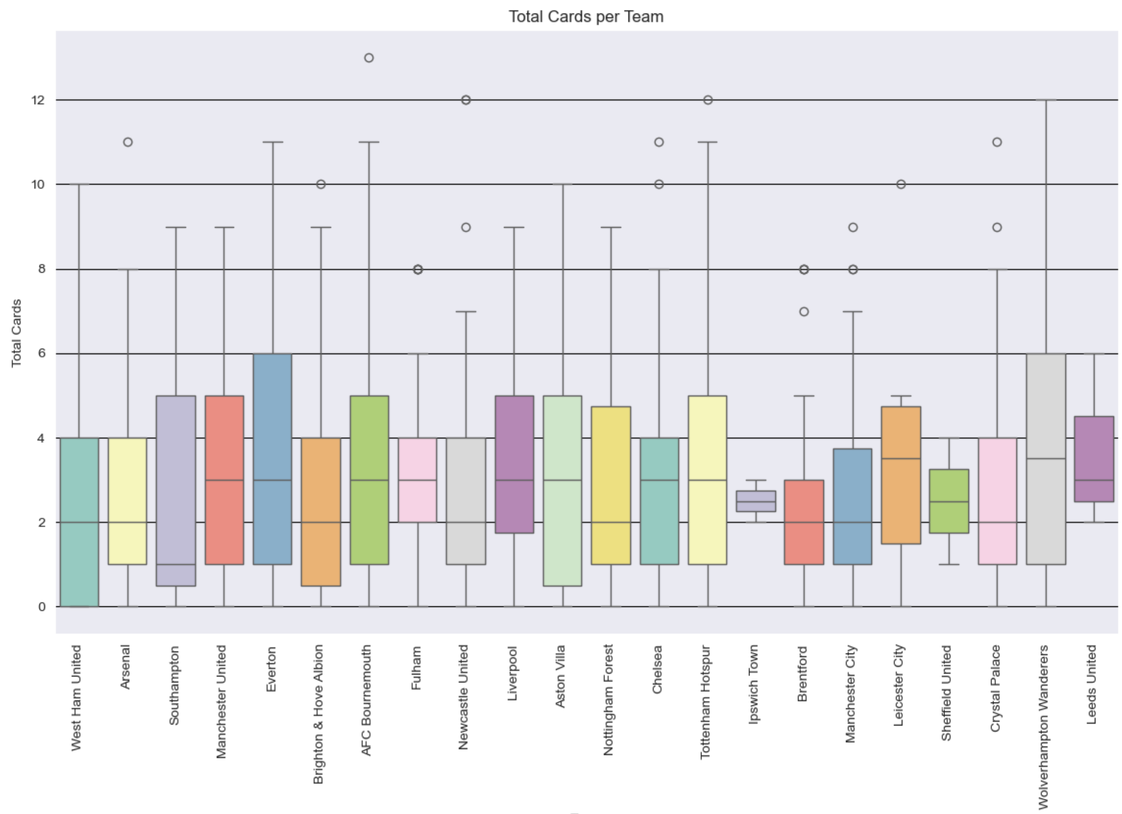
[43]:
```
plt.figure(figsize=(14,8))
sns.boxplot(x='Age', y='Appearances',data=df,palette='pastel')
plt.title('Appearances by Age')
plt.xlabel('Age')
plt.ylabel('Total Appearances')
plt.show()
```



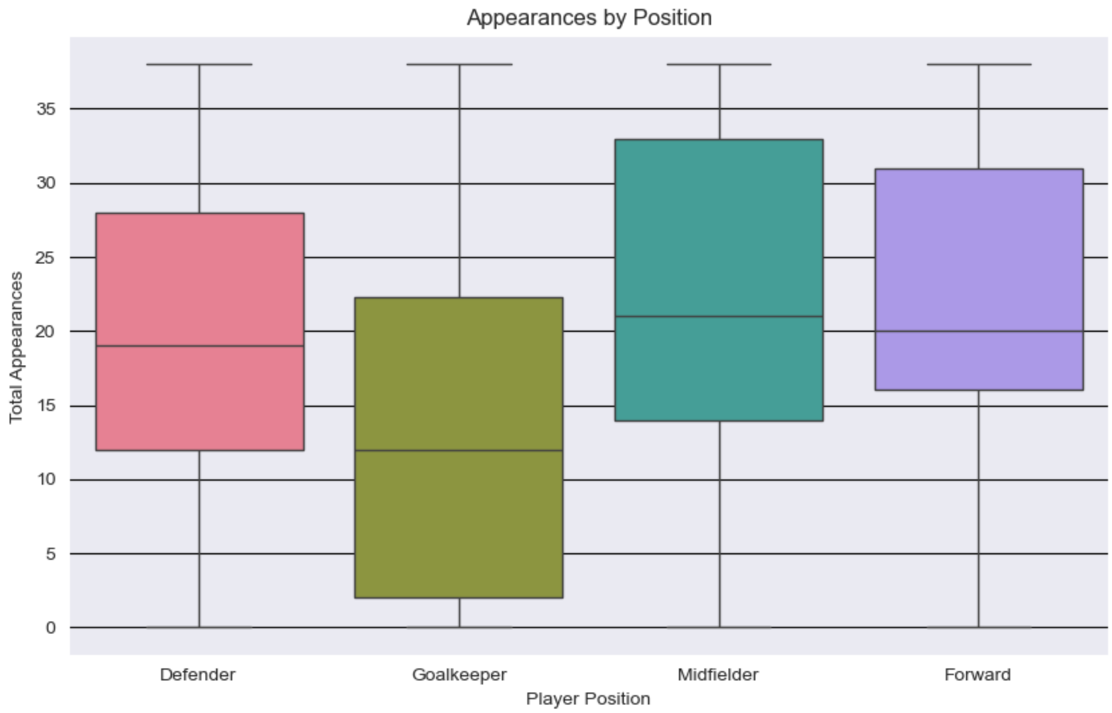
[44]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='AgeGroup',y='Total PlayTime (min)',data=df,palette='magma')
plt.title('Total Play Time by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Play Time (min)')
plt.show()
```



[45]:
```
plt.figure(figsize=(14,8))
sns.boxplot(x='Team',y='TotalCards',data=df,palette='Set3')
plt.title('Total Cards per Team')
plt.xlabel('Team')
plt.ylabel('Total Cards')
plt.xticks(rotation=90)
plt.show()
```

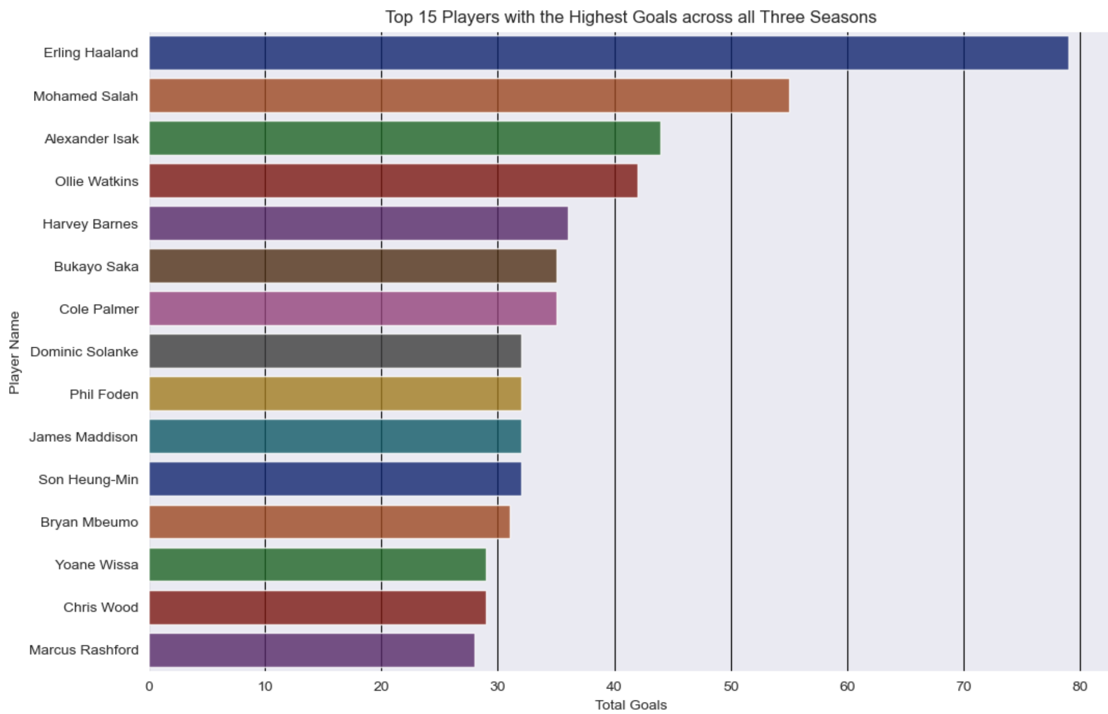


[46]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='Position',y='Appearances',data=df,palette='husl')
plt.title('Appearances by Position')
plt.xlabel('Player Position')
plt.ylabel('Total Appearances')
plt.show()
```



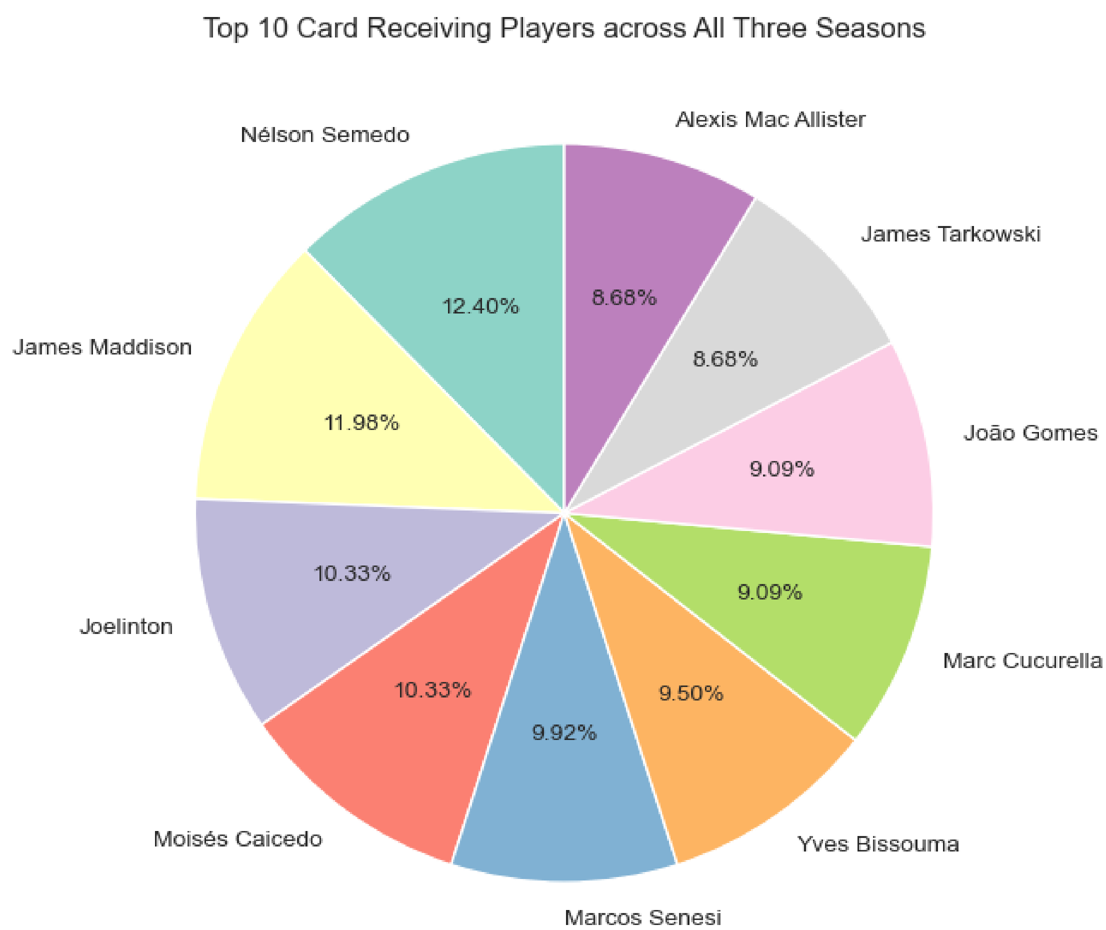
6.9 Horizontal Bar Plot

[48]:
```
topscorers=df.groupby('Id').agg({
    'Name': 'first',
    'TotalGoals': 'sum'
}).reset_index()

topscorers=topscorers.sort_values(by='TotalGoals',ascending=False).head(15)

plt.figure(figsize=(12, 8))
sns.barplot(x=topscorers['TotalGoals'],y=topscorers['Name'],␣
 ↪palette='dark',alpha=0.8)
plt.title('Top 15 Players with the Highest Goals across all Three Seasons')
plt.xlabel('Total Goals')
plt.ylabel('Player Name')
plt.show()
```



6.10 Pie Chart

[50]:
```
topoffenders=df.groupby('Id').agg({
    'Name': 'first',
    'TotalCards': 'sum'
}).reset_index()

topoffenders=topoffenders.sort_values(by='TotalCards',ascending=False).head(10)

plt.figure(figsize=(7, 7))
topoffenders.set_index('Name')['TotalCards'].plot.pie(autopct='%1.
↪2f%%',startangle=90,colors=sns.
↪color_palette('Set3',n_colors=len(topoffenders)))
plt.title('Top 10 Card Receiving Players across All Three Seasons')
plt.ylabel('')
plt.show()

topoffenders.reset_index(drop=True,inplace=True)
topoffenders
```

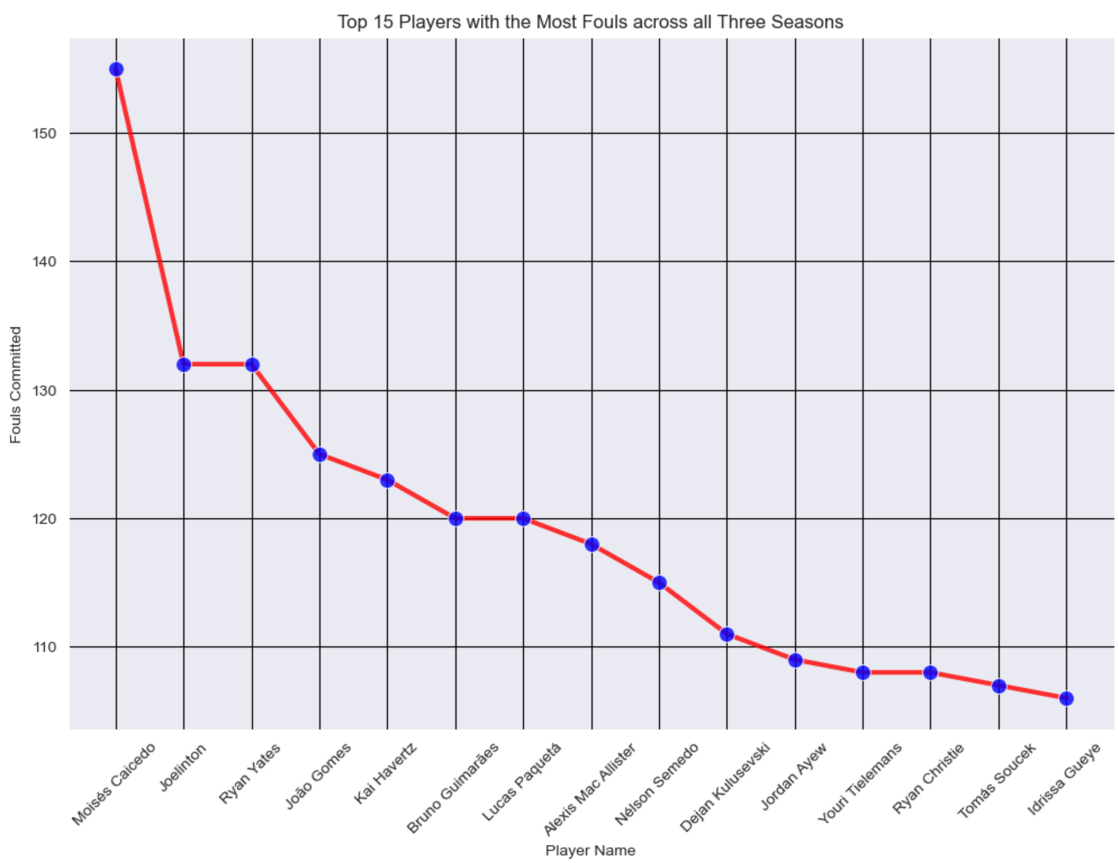


[50]:

	Id	Name	TotalCards
0	14735	Nélson Semedo	30
1	14685	James Maddison	29
2	14601	Joelinton	25
3	14304	Moisés Caicedo	25
4	14116	Marcos Senesi	24
5	14687	Yves Bissouma	23
6	14292	Marc Cucurella	22
7	14752	João Gomes	22
8	14365	James Tarkowski	21
9	14493	Alexis Mac Allister	21

6.11 Line Plot

[52]:
```
fouls=df.groupby('Id').agg({
    'Name': 'first',
    'FoulsCommitted': 'sum'
}).reset_index()

fouls=fouls.sort_values(by='FoulsCommitted',ascending=False).head(15)

plt.figure(figsize=(12, 8))
sns.
 ↪lineplot(x=fouls['Name'],y=fouls['FoulsCommitted'],marker='o',color='red',linewidth=3,marke
 ↪8)
plt.title('Top 15 Players with the Most Fouls across all Three Seasons')
plt.xlabel('Player Name')
plt.ylabel('Fouls Committed')
plt.xticks(rotation=45)
plt.show()

fouls.reset_index(drop=True,inplace=True)
fouls
```

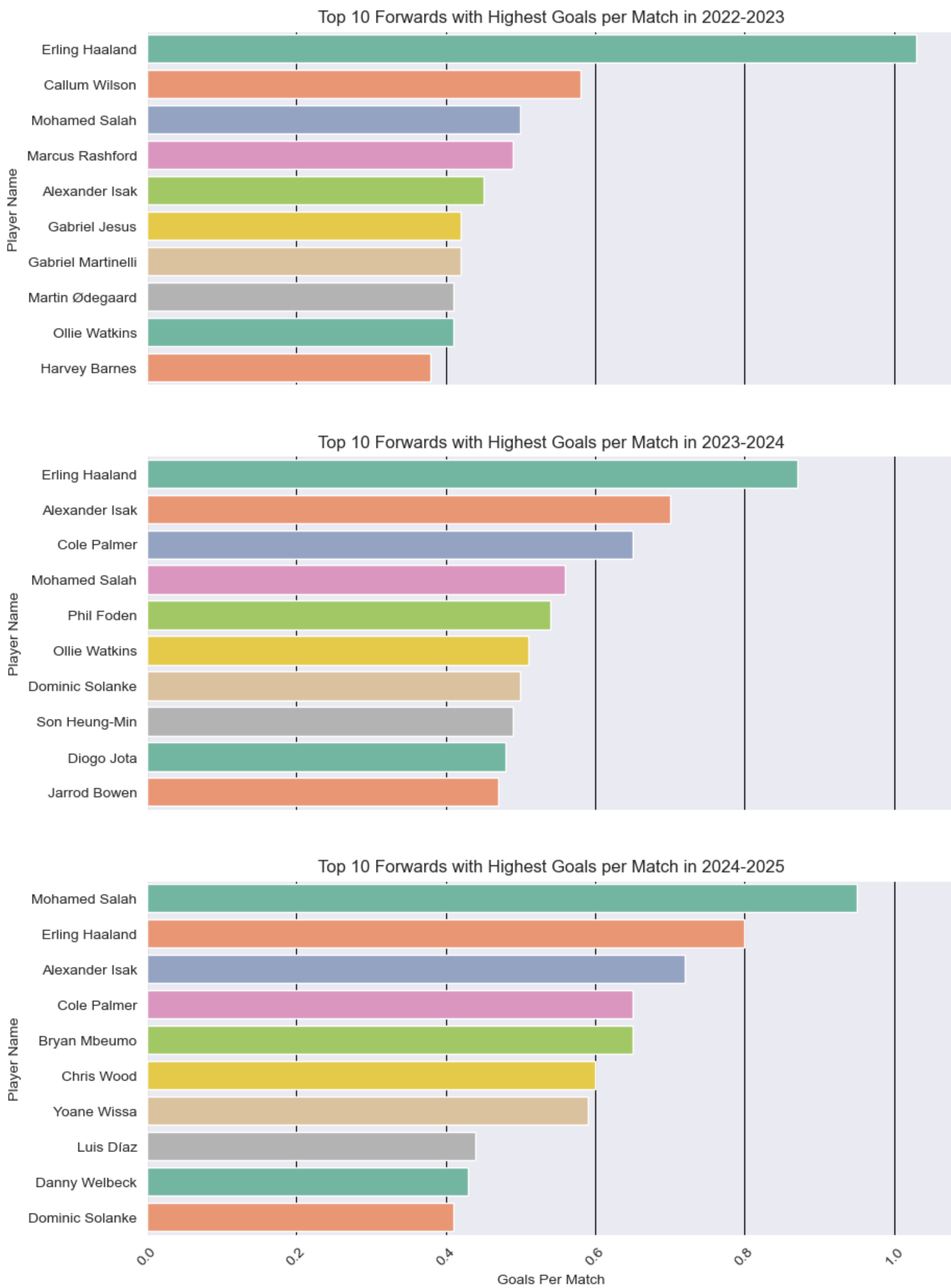


[52]:

	Id	Name	FoulsCommitted
0	14304	Moisés Caicedo	155
1	14601	Joelinton	132
2	14620	Ryan Yates	132
3	14752	João Gomes	125
4	14174	Kai Havertz	123
5	14594	Bruno Guimarães	120
6	14717	Lucas Paquetá	120
7	14493	Alexis Mac Allister	118
8	14735	Nélson Semedo	115
9	14688	Dejan Kulusevski	111
10	14469	Jordan Ayew	109
11	14199	Youri Tielemans	108
12	14128	Ryan Christie	108
13	14716	Tomás Soucek	107
14	14372	Idrissa Gueye	106

6.12 Multiple Horizontal Bar Charts

[54]:
```
players=df[df['Position'].isin(['Forward','Midfielder'])]

seasons=players['Season'].unique()

fig,axes=plt.subplots(nrows=len(seasons),figsize=(10,␣
↪5*len(seasons)),sharex=True)

for i, season in enumerate(seasons):
    season_data=players[players['Season']==season]

    top_forwards=season_data.sort_values(by='GoalsPerMatch', ascending=False).
↪head(10)

    sns.barplot(x='GoalsPerMatch',y='Name',data=top_forwards,␣
↪palette='Set2',ax=axes[i])

    axes[i].set_title(f'Top 10 Forwards with Highest Goals per Match in␣
↪{season}')
    axes[i].set_xlabel('Goals Per Match')
    axes[i].set_ylabel('Player Name')
    axes[i].tick_params(axis='x', rotation=45)

plt.show()
```

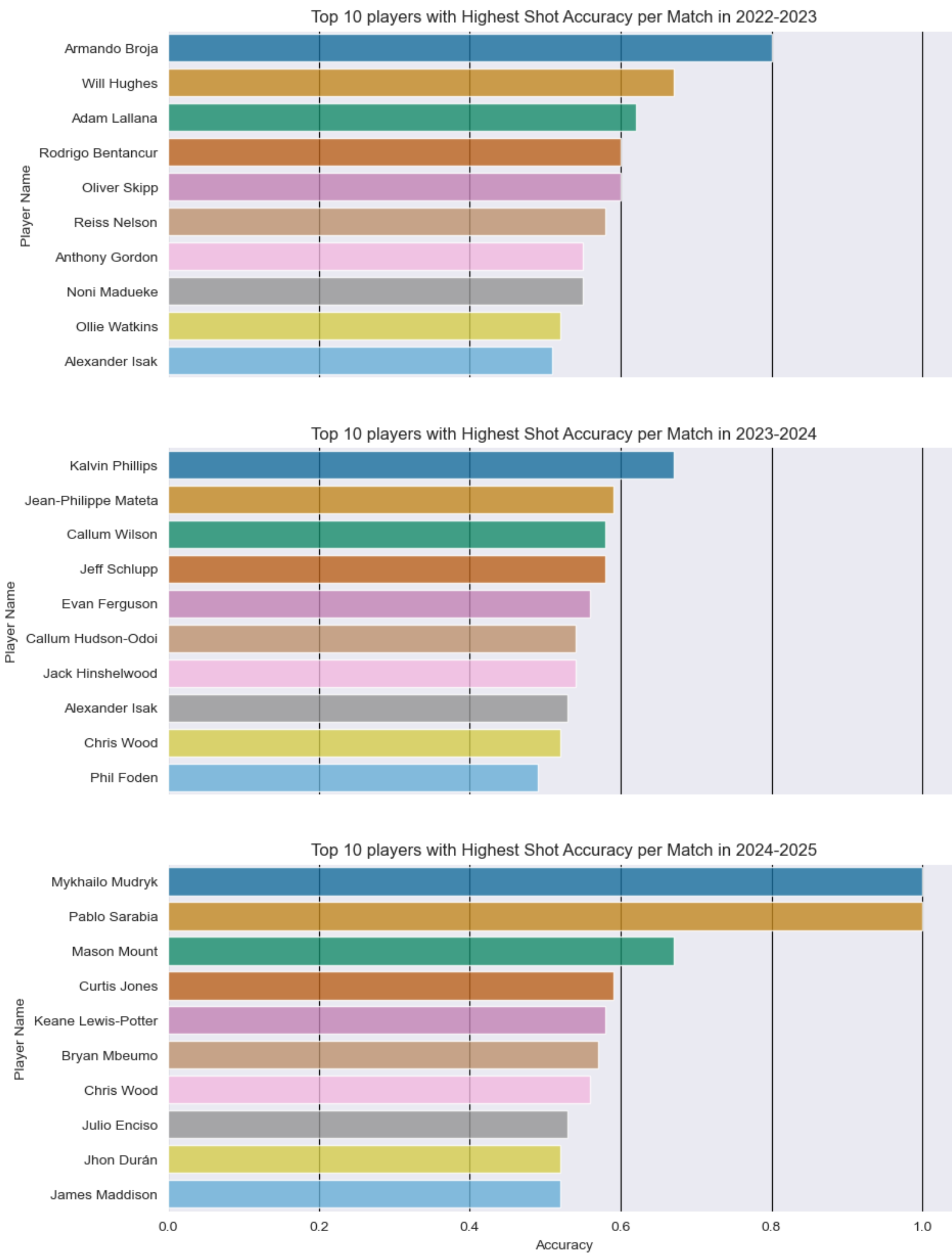


[55]:
```python
goalkeepers=df[df['Position'] == 'Goalkeeper']
```

```
seasons=goalkeepers['Season'].unique()

fig,axes=plt.subplots(nrows=len(seasons),figsize=(10,␣
↪5*len(seasons)),sharex=True)

for i, season in enumerate(seasons):
    season_data=goalkeepers[goalkeepers['Season']==season]

    top_goalkeeprs=season_data.sort_values(by='GoalKeeperAccuracy',␣
↪ascending=False).head(10)

    sns.
↪barplot(x='GoalKeeperAccuracy',y='Name',data=top_goalkeeprs,palette='Set1',alpha=0.
↪8,ax=axes[i])

    axes[i].set_title(f'Top 10 Goalkeepers with Highest Save Accuracy per Match␣
↪in {season}')
    axes[i].set_xlabel('Accuracy')
    axes[i].set_ylabel('Player Name')
    axes[i].tick_params(axis='x')

plt.show()
```

[56]:
```
players=df[df['Position'].isin(['Forward','Midfielder'])]

seasons=players['Season'].unique()
```

```
fig,axes=plt.subplots(nrows=len(seasons),figsize=(10,␣
↪5*len(seasons)),sharex=True)

for i, season in enumerate(seasons):
    season_data=players[players['Season']==season]

    top_players=season_data.sort_values(by='ShotAccuracy', ascending=False).
↪head(10)

    sns.
↪barplot(x='ShotAccuracy',y='Name',data=top_players,palette='colorblind',alpha=0.
↪8,ax=axes[i])

    axes[i].set_title(f'Top 10 players with Highest Shot Accuracy per Match in␣
↪{season}')
    axes[i].set_xlabel('Accuracy')
    axes[i].set_ylabel('Player Name')
    axes[i].tick_params(axis='x')

plt.show()
```


Top 10 players with Highest Shot Accuracy per Match in 2022-2023
Player Name
Armando Broja
Will Hughes
Adam Lallana
Rodrigo Bentancur
Oliver Skipp
Reiss Nelson
Anthony Gordon
Noni Madueke
Ollie Watkins
Alexander Isak
Top 10 players with Highest Shot Accuracy per Match in 2023-2024
Player Name
Kalvin Phillips
Jean-Philippe Mateta
Callum Wilson
Jeff Schlupp
Evan Ferguson
Callum Hudson-Odoi
Jack Hinshelwood
Alexander Isak
Chris Wood
Phil Foden
Top 10 players with Highest Shot Accuracy per Match in 2024-2025
Player Name
Mykhailo Mudryk
Pablo Sarabia
Mason Mount
Curtis Jones
Keane Lewis-Potter
Bryan Mbeumo
Chris Wood
Julio Enciso
Jhon Durán
James Maddison
0.0
0.2
0.4
0.6
0.8
1.0
Accuracy

7 HYPOTHESIS TESTING

[58]:
```
# import neccessary libraries for hypothesis testing

from scipy import stats
```

7.1 T-Test (2 Sample)

[60]:
```
# T-Test for Comparing Means
# Null Hypothesis (H0): The mean Shot Accuracy of Forwards and Midfielders is␣
↪the same.
# Alternative Hypothesis (H1): The mean Shot Accuracy of Forwards and␣
↪Midfielders is different.

forwards=df[df['Position']=='Forward']
midfielders=df[df['Position']=='Midfielder']

t_stat,p_value=stats.ttest_ind(forwards['ShotAccuracy'].dropna(),␣
↪midfielders['ShotAccuracy'].dropna())

print(f'T-statistic: {t_stat},\nP-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis. The mean Shot Accuracy of Forwards and␣
↪Midfielders is significantly different.")
else:
    print("Fail to reject the null hypothesis. The mean Shot Accuracy of␣
↪Forwards and Midfielders is not significantly different.")
```

```
T-statistic: 4.48890810786535,
P-value: 9.077942040353678e-06
Reject the null hypothesis. The mean Shot Accuracy of Forwards and Midfielders
is significantly different.
```

[61]:
```
# We can compare the Total Playtime between two groups, Goalkeepers and␣
↪Outfield Players(Defenders, Midfielders, Forwards).
# Null Hypothesis (H0): There is no significant difference in Total Playtime␣
↪between Goalkeepers and Outfield Players.
# Alternative Hypothesis (H1): There is a significant difference in Total␣
↪Playtime between Goalkeepers and Outfield Players.

goalkeepers=df[df['Position']=='Goalkeeper']['Total PlayTime (min)']
outfield_players=df[df['Position']!='Goalkeeper']['Total PlayTime (min)']

t_stat,p_value=stats.ttest_ind(goalkeepers,outfield_players)
```

```
print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

if p_value<0.05:
    print("Reject the null hypothesis: There is a significant difference in␣
↪Total Playtime between Goalkeepers and Outfield Players.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
↪difference in Total Playtime between Goalkeepers and Outfield Players.")
```

```
T-statistic: -1.655333852218366
P-value: 0.09824997606856764
Fail to reject the null hypothesis: There is no significant difference in Total
Playtime between Goalkeepers and Outfield Players.
```

7.2 ANOVA Test

[63]:
```
# One-Way ANOVA (comparing Appearance across Positions)
# Null Hypothesis (H0): There is no significant difference in the number of␣
↪appearances across the positions.
# Alternative Hypothesis (H1): There is a significant difference in the number␣
↪of appearances across the positions.

positions=df['Position'].unique()

app_by_position=[df[df['Position']==pos]['Appearances'] for pos in positions]

f_stat,p_value=stats.f_oneway(*app_by_position)

print(f"F-statistic: {f_stat}")
print(f"P-value: {p_value}")

if p_value<0.05:
    print("Reject the null hypothesis: There is a significant difference in␣
↪Appearances across player positions.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
↪difference in Appearances across player positions.")
```

```
F-statistic: 11.614025760401699
P-value: 1.8667339921200415e-07
Reject the null hypothesis: There is a significant difference in Appearances
across player positions.
```

[64]:
```
# One-Way ANOVA (comparing Mean ShotAccuracy across Positions)
# Null Hypothesis (H0): The mean Shot Accuracy is the same across all Positions.
# Alternative Hypothesis (H1): At least one Position has a different mean Shot␣
↪Accuracy.
```

```
positions=['Forward','Midfielder','Defender']
position_data=[df[df['Position']==pos]['ShotAccuracy'].dropna() for pos in␣
 ↪positions]

f_stat,p_value=stats.f_oneway(*position_data)

print(f'F-statistic: {f_stat},\nP-value: {p_value}')

if p_value<alpha:
    print("Reject the null hypothesis. The mean Shot Accuracy is different␣
 ↪across positions.")
else:
    print("Fail to reject the null hypothesis. The mean Shot Accuracy is not␣
 ↪different across positions.")
```

```
F-statistic: 12.909657442908538,
P-value: 3.1026059534054243e-06
Reject the null hypothesis. The mean Shot Accuracy is different across
positions.
```

7.3 Chi-Square Test

[66]:
```
# Chi-Square Test
# Null Hypothesis (H0): There is no association between Position and AgeGroup␣
 ↪(they are independent).
# Alternative Hypothesis (H1): There is an association between Position and␣
 ↪AgeGroup (they are dependent).

contingency=pd.crosstab(df['Position'], df['AgeGroup'])

chi2,p_value,dof,expected=stats.chi2_contingency(contingency)

print(f'Chi2-statistic: {chi2},\nP-value: {p_value}')
print(f'Degrees of Freedom: {dof}')
print()
print("Expected Frequencies:")
print(expected)
print()
alpha = 0.05
if p_value<alpha:
    print("Reject the null hypothesis. There is an association between Position␣
 ↪and AgeGroup.")
else:
    print("Fail to reject the null hypothesis. Position and AgeGroup are␣
 ↪independent.")
```

```
Chi2-statistic: 107.61435435318108,
P-value: 6.436196503023455e-21
Degrees of Freedom: 6

Expected Frequencies:
[[164.67     17.49     81.84    ]
 [110.40375  11.72625  54.87    ]
 [ 48.6525    5.1675   24.18    ]
 [175.27375  18.61625  87.11    ]]

Reject the null hypothesis. There is an association between Position and
AgeGroup.
```

8 CORRELATION

[68]:
```
correlation_matrix=df[['Age','Appearances','AveragePlayTime␣
↪(min)','TotalGoals','GoalsConceded','GoalsPerMatch','ShotAccuracy','GoalKeeperAccuracy','Fo
↪corr(method='pearson')
correlation_dataframe=pd.DataFrame(correlation_matrix)
correlation_dataframe.round(2)
```

[68]:

	Age	Appearances	AveragePlayTime (min)	TotalGoals
Age	1.00	-0.04	0.12	-0.10
Appearances	-0.04	1.00	0.57	0.44
AveragePlayTime (min)	0.12	0.57	1.00	0.22
TotalGoals	-0.10	0.44	0.22	1.00
GoalsConceded	0.02	0.83	0.68	0.33
GoalsPerMatch	-0.11	0.31	0.20	0.94
ShotAccuracy	-0.08	0.29	0.20	0.34
GoalKeeperAccuracy	0.24	-0.07	0.24	-0.15
FoulsInvolvement	-0.12	0.72	0.43	0.47
TotalCards	-0.01	0.61	0.45	0.17

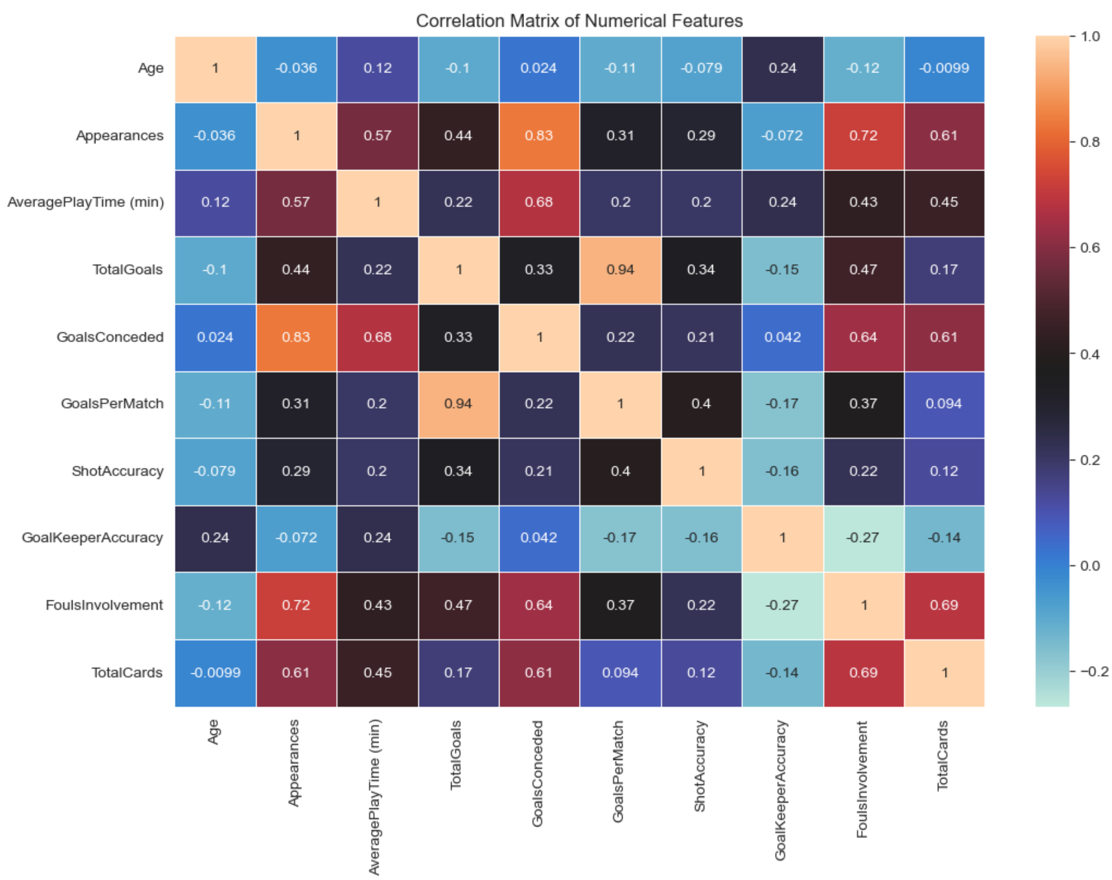
	GoalsConceded	GoalsPerMatch	ShotAccuracy
Age	0.02	-0.11	-0.08
Appearances	0.83	0.31	0.29
AveragePlayTime (min)	0.68	0.20	0.20
TotalGoals	0.33	0.94	0.34
GoalsConceded	1.00	0.22	0.21
GoalsPerMatch	0.22	1.00	0.40
ShotAccuracy	0.21	0.40	1.00
GoalKeeperAccuracy	0.04	-0.17	-0.16
FoulsInvolvement	0.64	0.37	0.22
TotalCards	0.61	0.09	0.12

	GoalKeeperAccuracy	FoulsInvolvement	TotalCards
Age	0.24	-0.12	-0.01

```
Appearances                   -0.07        0.72      0.61
AveragePlayTime (min)          0.24        0.43      0.45
TotalGoals                    -0.15        0.47      0.17
GoalsConceded                  0.04        0.64      0.61
GoalsPerMatch                 -0.17        0.37      0.09
ShotAccuracy                  -0.16        0.22      0.12
GoalKeeperAccuracy             1.00       -0.27     -0.14
FoulsInvolvement              -0.27        1.00      0.69
TotalCards                    -0.14        0.69      1.00
```

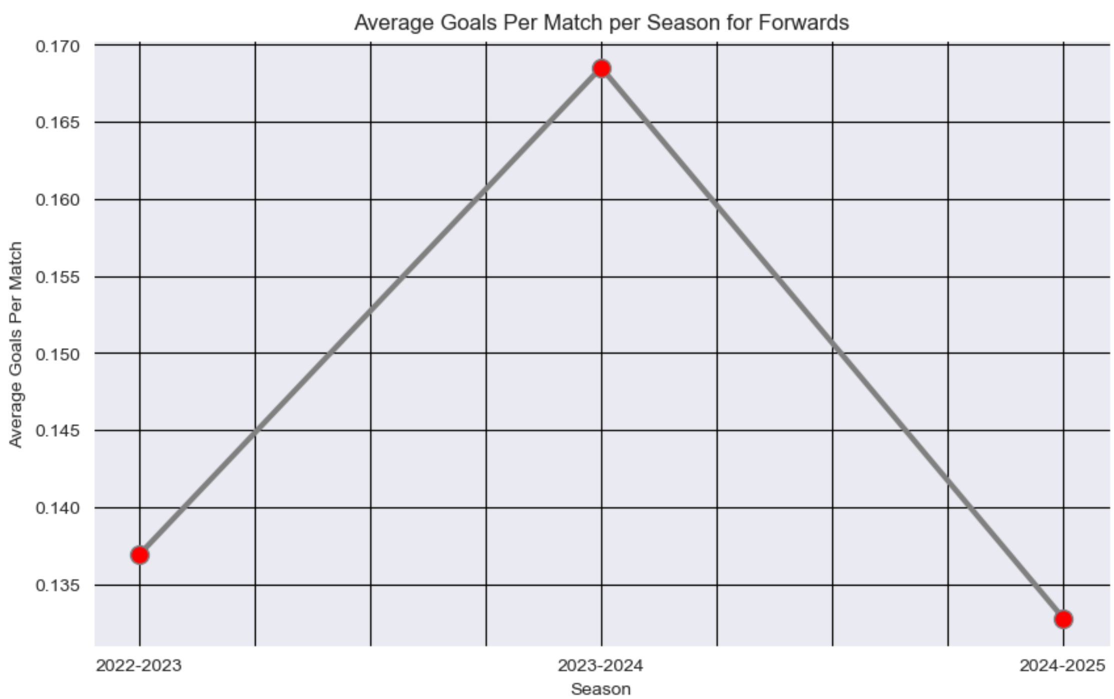
[69]:

```python
plt.figure(figsize=(12,8))
sns.heatmap(correlation_matrix,annot=True,cmap='icefire',linewidths=0.5)
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```



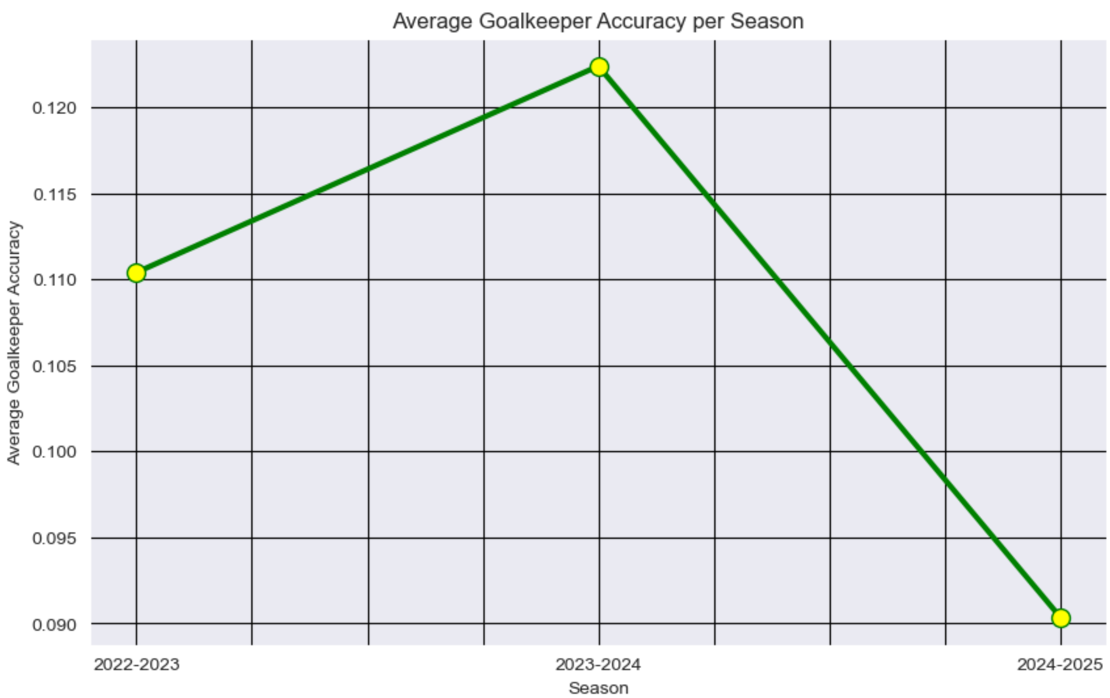
9 TIME SERIES

[71]:
```
# Plotting the average goals per match for forwards and midfielders across␣
  ↪seasons

forward=df[df['Position'].isin(['Forward','Midfielder'])]
season_goals=forward.groupby('Season')['GoalsPerMatch'].mean()

plt.figure(figsize=(10, 6))
season_goals.
  ↪plot(kind='line',marker='o',color='grey',linewidth=3,markersize=10,markerfacecolor='red')
plt.title('Average Goals Per Match per Season for Forwards')
plt.xlabel('Season')
plt.ylabel('Average Goals Per Match')
plt.show()
```

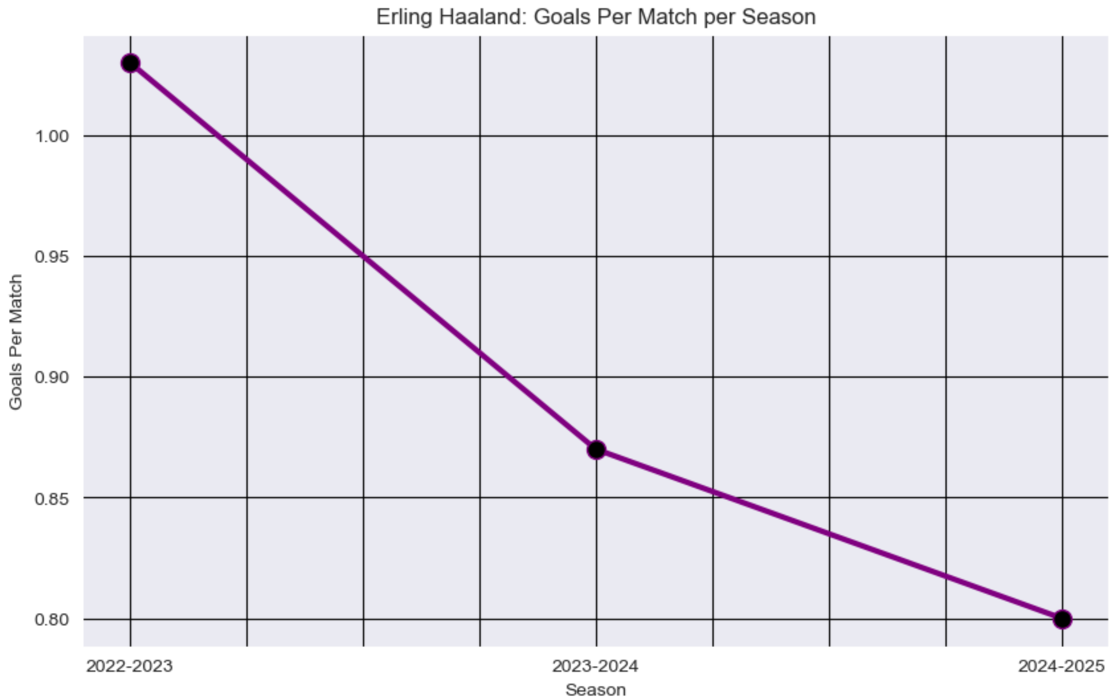


[72]:
```
# Plotting the average goalkeeper accuracy across seasons

goalkeepers_df=df[df['Position']=='Goalkeeper']

season_goalkeeper_accuracy=goalkeepers_df.
  ↪groupby('Season')['GoalKeeperAccuracy'].mean()

plt.figure(figsize=(10,6))
```

```
season_goalkeeper_accuracy.
  ↪plot(kind='line',marker='o',color='green',linewidth=3,markersize=10,markerfacecolor='yellow
plt.title('Average Goalkeeper Accuracy per Season')
plt.xlabel('Season')
plt.ylabel('Average Goalkeeper Accuracy')
plt.show()
```



[73]:
```
# Filter the data for Erling Haaland

haaland=df[df['Name'] == 'Erling Haaland']

season_haaland_goals=haaland.groupby('Season')['GoalsPerMatch'].mean()

plt.figure(figsize=(10, 6))
season_haaland_goals.
  ↪plot(kind='line',marker='o',color='purple',linewidth=3,markersize=10,markerfacecolor='black
plt.title('Erling Haaland: Goals Per Match per Season')
plt.xlabel('Season')
plt.ylabel('Goals Per Match')
plt.show()
```



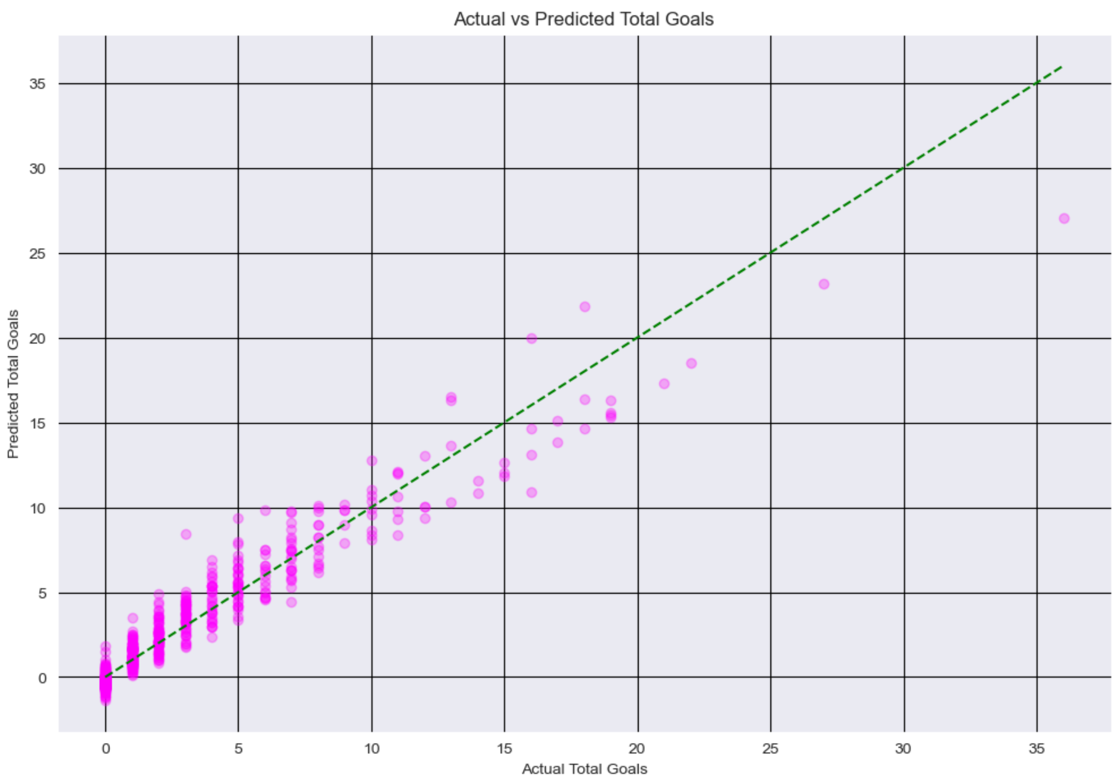
10 LINEAR REGRESSION

[75]:
```
# importing the libraries for regression analysis (linear regression)

import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

[76]:
```
df_reg= df.groupby('Id').filter(lambda x: len(x) == 3)

X = df_reg[['Age','TotalShots','FoulsInvolvement','GoalsPerMatch','ShotAccuracy','YellowCards',
y = df_reg['TotalGoals']
model=LinearRegression()
model.fit(X, y)
y_pred=model.predict(X)
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print(f'R-Squared: {r2}')
print(f'Mean Squared Error: {mse}')
```

```
plt.figure(figsize=(12,8))
plt.scatter(y, y_pred,color='Magenta',alpha=0.3)
plt.plot([min(y),max(y)], [min(y),max(y)],color='green',linestyle='--')
plt.title('Actual vs Predicted Total Goals')
plt.xlabel('Actual Total Goals')
plt.ylabel('Predicted Total Goals')
plt.show()
```

```
R-Squared: 0.9268291272555538
Mean Squared Error: 1.1645696361913669
```



[77]:
```
next_season=df_reg[df_reg['Season']=='2024-2025'][['Age','TotalShots','FoulsInvolvement','Goal
    ↪values

player=df_reg[df_reg['Season']=='2024-2025'][['Id','Name']]

predicted_goals=model.predict(next_season)

predicted_df=pd.DataFrame({
    'Player Id': player['Id'],
    'Player Name': player['Name'],
```

```
    'Predicted TotalGoals (2025-2026)': predicted_goals
})

predicted_df['Predicted TotalGoals (2025-2026)']=predicted_df['Predicted␣
↪TotalGoals (2025-2026)'].round(0)
predicted_df['Predicted TotalGoals (2025-2026)']=predicted_df['Predicted␣
↪TotalGoals (2025-2026)'].astype(int)
predicted_df.sort_values(by='Predicted TotalGoals␣
↪(2025-2026)',ascending=False,inplace=True)
predicted_df.reset_index(drop=True, inplace=True)
predicted_df.head(20)
```

[77]:

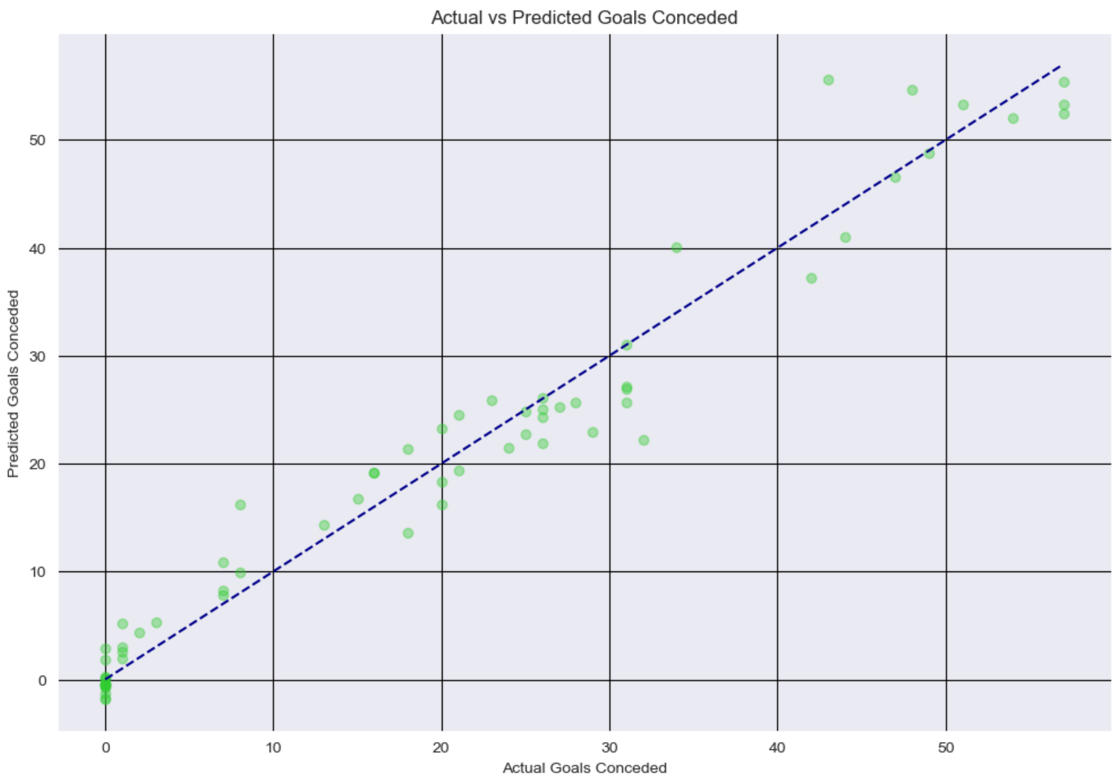
```
    Player Id      Player Name  Predicted TotalGoals (2025-2026)
0       14500    Mohamed Salah                                22
1       14536   Erling Haaland                                20
2       14602   Alexander Isak                                17
3       14306      Cole Palmer                                16
4       14238     Bryan Mbeumo                                14
5       14232      Yoane Wissa                                13
6       14624       Chris Wood                                13
7       14209    Ollie Watkins                                10
8       14504        Luis Díaz                                10
9       14410     Raúl Jiménez                                10
10      14696  Dominic Solanke                                10
11      14174      Kai Havertz                                10
12      14277    Danny Welbeck                                10
13      14697  Brennan Johnson                                 9
14      14141  Antoine Semenyo                                 9
15      14176      Bukayo Saka                                 8
16      14319     Noni Madueke                                 8
17      14211       Jhon Durán                                 8
18      14503      Cody Gakpo                                  8
19      14693   Son Heung-Min                                  7
```

[78]:

```
goalkeepers=df[df['Position']=='Goalkeeper']
df_reg=goalkeepers.groupby('Id').filter(lambda x: len(x) == 3)

X =␣
↪df_reg[['Age','ShotsFaced','FoulsInvolvement','GoalKeeperAccuracy','YellowCards','RedCards'
y = df_reg['GoalsConceded']
model=LinearRegression()
model.fit(X, y)
y_pred=model.predict(X)
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print(f'R-Squared: {r2}')
```

```
print(f'Mean Squared Error: {mse}')

plt.figure(figsize=(12,8))
plt.scatter(y, y_pred, color='LimeGreen', alpha=0.4)
plt.plot([min(y),max(y)], [min(y),max(y)],color='darkblue',linestyle='--')
plt.title('Actual vs Predicted Goals Conceded')
plt.xlabel('Actual Goals Conceded')
plt.ylabel('Predicted Goals Conceded')
plt.show()
```

```
R-Squared: 0.9612034186055136
Mean Squared Error: 12.186259159743633
```



[79]:
```
next_season=df_reg[df_reg['Season']=='2024-2025'][['Age','ShotsFaced','FoulsInvolvement','Goal
↪values

gkp=df_reg[df_reg['Season']=='2024-2025'][['Id','Name']]

conceded_goals=model.predict(next_season)

conceded_df = pd.DataFrame({
    'Player Id': gkp['Id'],
```

```python
    'Player Name': gkp['Name'],
    'Predicted GoalsConceded (2025-2026)': conceded_goals
})

conceded_df['Predicted GoalsConceded (2025-2026)']=conceded_df['Predicted␣
↪GoalsConceded (2025-2026)'].round(0)
conceded_df['Predicted GoalsConceded (2025-2026)']=conceded_df['Predicted␣
↪GoalsConceded (2025-2026)'].astype(int)
conceded_df.sort_values(by='Predicted GoalsConceded (2025-2026)',␣
↪ascending=False, inplace=True)
conceded_df.reset_index(drop=True, inplace=True)
conceded_df.head(15)
```

[79]:

	Player Id	Player Name	Predicted GoalsConceded (2025-2026)
0	14179	Emiliano Martínez	27
1	14327	Dean Henderson	26
2	14359	Jordan Pickford	25
3	14283	Robert Sánchez	25
4	14391	Bernd Leno	25
5	14578	Nick Pope	23
6	14732	José Sá	22
7	14146	David Raya	21
8	14702	Lukasz Fabianski	19
9	14703	Alphonse Areola	19
10	14670	Fraser Forster	17
11	14511	Ederson	14
12	14510	Stefan Ortega	10
13	14575	Martin Dubravka	5
14	14673	Brandon Austin	4