Football Data Analysis Notebook

AUTHORS

1. Vishalkrishna Bhosle - 22205256
2. Shubham Sharma - 22201541

Introduction

This Python notebook focuses on analyzing football match data using the Pandas, Seaborn, and Matplotlib libraries. The dataset used in this analysis contains information about various attributes of football matches, such as goals scored, shots, cards, and more.

Importing Libraries

We begin by importing the necessary libraries for data analysis and visualization: `pandas`, `seaborn`, and `matplotlib.pyplot`.

In [16]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Loading

In this section, we load the football match data from a CSV file using Pandas and display basic information about the DataFrame.

In [17]:

```
# Load the CSV data into a Pandas DataFrame
url = "https://www.football-data.co.uk/mmz4281/2223/E0.csv"
data = pd.read_csv(url, usecols=range(24))
```

In [18]:

```
# Display basic information about the DataFrame
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380 entries, 0 to 379
Data columns (total 24 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Div       380 non-null    object
 1   Date      380 non-null    object
 2   Time      380 non-null    object
 3   HomeTeam  380 non-null    object
 4   AwayTeam  380 non-null    object
 5   FTHG      380 non-null    int64 
 6   FTAG      380 non-null    int64 
 7   FTR       380 non-null    object
 8   HTHG      380 non-null    int64 
 9   HTAG      380 non-null    int64 
 10  HTR       380 non-null    object
 11  Referee   380 non-null    object
 12  HS        380 non-null    int64 
 13  AS        380 non-null    int64 
 14  HST       380 non-null    int64 
 15  AST       380 non-null    int64 
 16  HF        380 non-null    int64 
 17  AF        380 non-null    int64 
 18  HC        380 non-null    int64 
 19  AC        380 non-null    int64 
 20  HY        380 non-null    int64 
 21  AY        380 non-null    int64 
 22  HR        380 non-null    int64 
 23  AR        380 non-null    int64 
dtypes: int64(16), object(8)
memory usage: 71.4+ KB
```

In [19]:
```
# Display the first few rows of the DataFrame
data.head()
```

Out[19]:

	Div	Date	Time	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	...	H
0	E0	05/08/2022	20:00	Crystal Palace	Arsenal	0	2	A	0	1	...	
1	E0	06/08/2022	12:30	Fulham	Liverpool	2	2	D	1	0	...	
2	E0	06/08/2022	15:00	Bournemouth	Aston Villa	2	0	H	1	0	...	
3	E0	06/08/2022	15:00	Leeds	Wolves	2	1	H	1	1	...	
4	E0	06/08/2022	15:00	Newcastle	Nott'm Forest	2	0	H	0	0	...	

5 rows × 24 columns

Exploratory Data Analysis (EDA)

Summary Statistics and Correlation Analysis

We analyze the dataset by calculating summary statistics and creating a correlation matrix heatmap.

In [5]:
```
# Summary statistics of numerical columns
data.describe()
```

Out[5]:

	FTHG	FTAG	HTHG	HTAG	HS	AS	H
count	380.000000	380.000000	380.000000	380.000000	380.000000	380.000000	380.000
mean	1.634211	1.218421	0.757895	0.563158	13.952632	11.310526	4.907
std	1.419944	1.183518	0.918480	0.746998	5.604170	4.941173	2.495
min	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000
25%	1.000000	0.000000	0.000000	0.000000	10.000000	8.000000	3.000
50%	1.000000	1.000000	1.000000	0.000000	14.000000	11.000000	5.000
75%	2.000000	2.000000	1.000000	1.000000	17.000000	15.000000	7.000
max	9.000000	6.000000	5.000000	3.000000	33.000000	30.000000	15.000

In [6]:

```python
# Correlation matrix
data.corr()
```

```
<ipython-input-6-a7b7c8db44ff>:2: FutureWarning: The default value of numeri
c_only in DataFrame.corr is deprecated. In a future version, it will default
to False. Select only valid columns or specify the value of numeric_only to
silence this warning.
  data.corr()
```

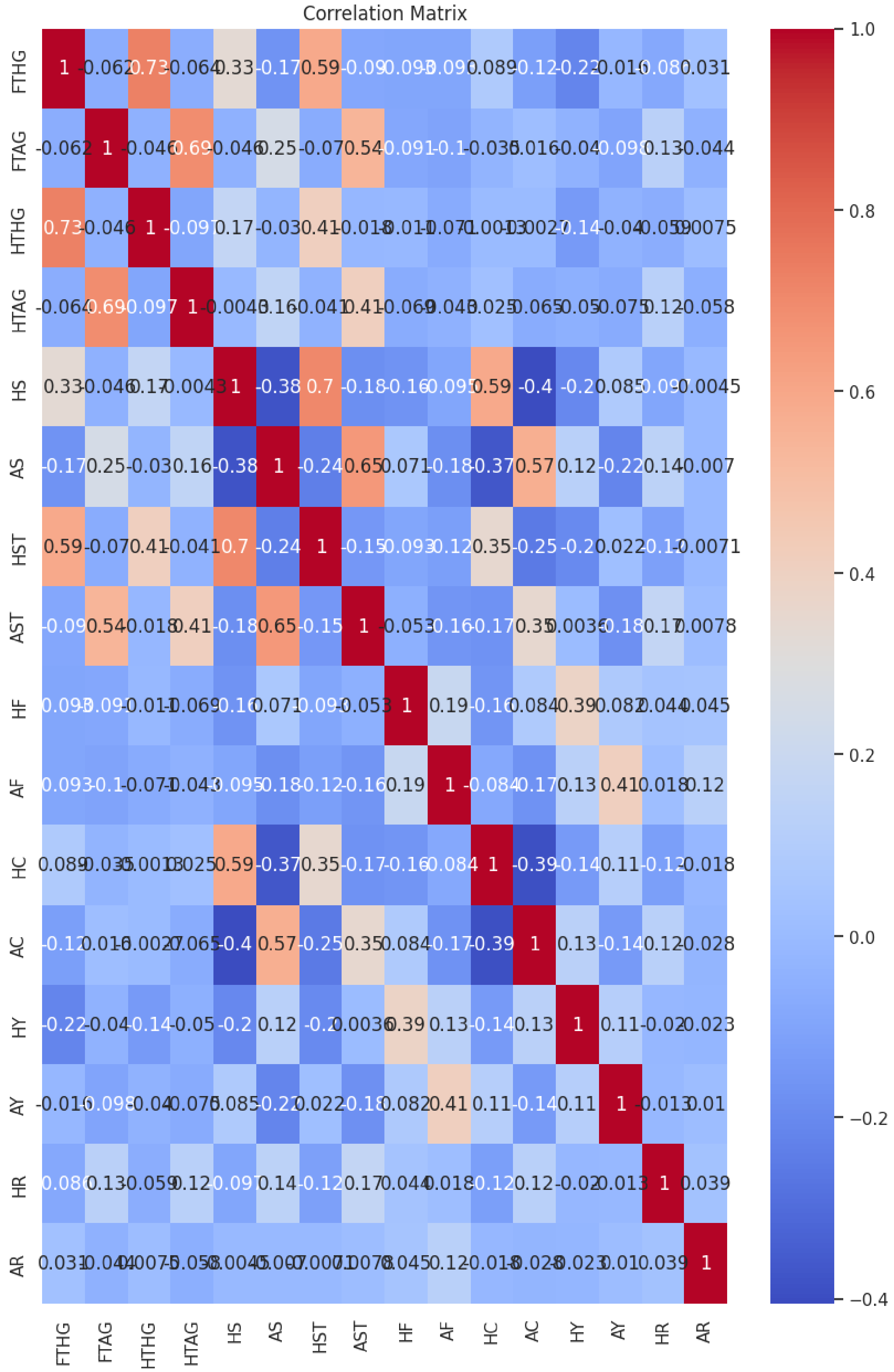
Out[6]:

	FTHG	FTAG	HTHG	HTAG	HS	AS	HST	
FTHG	1.000000	-0.062236	0.727000	-0.063982	0.334694	-0.166534	0.591428	-0.08
FTAG	-0.062236	1.000000	-0.045888	0.690181	-0.045775	0.252315	-0.070006	0.54
HTHG	0.727000	-0.045888	1.000000	-0.096870	0.167437	-0.029901	0.411608	-0.01
HTAG	-0.063982	0.690181	-0.096870	1.000000	-0.004326	0.156227	-0.041461	0.40
HS	0.334694	-0.045775	0.167437	-0.004326	1.000000	-0.382032	0.703854	-0.18
AS	-0.166534	0.252315	-0.029901	0.156227	-0.382032	1.000000	-0.239923	0.65
HST	0.591428	-0.070006	0.411608	-0.041461	0.703854	-0.239923	1.000000	-0.14
AST	-0.089661	0.540437	-0.017623	0.409373	-0.184451	0.653632	-0.149648	1.00
HF	-0.092663	-0.090573	-0.011395	-0.068577	-0.164562	0.071216	-0.092971	-0.05
AF	-0.092555	-0.104485	-0.070747	-0.043199	-0.095325	-0.182537	-0.119496	-0.15
HC	0.088588	-0.035441	-0.001318	0.024958	0.591543	-0.366457	0.347253	-0.17
AC	-0.119717	0.015748	-0.002715	-0.064945	-0.404807	0.570543	-0.254269	0.34
HY	-0.216682	-0.040491	-0.138530	-0.049761	-0.204545	0.124693	-0.196741	0.00
AY	-0.016012	-0.098492	-0.039648	-0.075355	0.085433	-0.217023	0.021838	-0.17
HR	-0.086109	0.129621	-0.059330	0.123567	-0.096663	0.141143	-0.119221	0.16
AR	0.030813	-0.044289	0.007546	-0.057992	-0.004483	-0.007014	-0.007118	0.00

In [22]:

```python
# Plot Correlation matrix
correlation_matrix = data.corr()
sns.set_theme(style="whitegrid", palette="pastel")
plt.figure(figsize=(10, 15))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-22-4b39d4cd3011>:2: FutureWarning: The default value of numer
ic_only in DataFrame.corr is deprecated. In a future version, it will defaul
t to False. Select only valid columns or specify the value of numeric_only t
o silence this warning.
  correlation_matrix = data.corr()
```



Correlation Pairs

We identify and interpret correlation pairs between various attributes.

In [8]:
```
correlation_matrix = data.corr()

# List of correlation pairs to display
correlation_pairs = [
    ('FTHG', 'HTHG'),
    ('FTHG', 'HST'),
    ('FTAG', 'HTAG'),
    ('FTAG', 'AST'),
    ('HS', 'AS'),
    ('HY', 'FTHG'),
    ('HY', 'FTAG'),
    ('HY', 'HR'),
    ('AY', 'FTAG'),
    ('AY', 'HR'),
    ('HR', 'FTAG')
]

# Display the correlation pairs and their coefficients
for var1, var2 in correlation_pairs:
    correlation = correlation_matrix.loc[var1, var2]
    print(f"Correlation between {var1} and {var2}: {correlation:.4f}")
```

```
Correlation between FTHG and HTHG: 0.7270
Correlation between FTHG and HST: 0.5914
Correlation between FTAG and HTAG: 0.6902
Correlation between FTAG and AST: 0.5404
Correlation between HS and AS: -0.3820
Correlation between HY and FTHG: -0.2167
Correlation between HY and FTAG: -0.0405
Correlation between HY and HR: -0.0195
Correlation between AY and FTAG: -0.0985
Correlation between AY and HR: -0.0128
Correlation between HR and FTAG: 0.1296
```

```
<ipython-input-8-67cf38de34e3>:1: FutureWarning: The default value of numeri
c_only in DataFrame.corr is deprecated. In a future version, it will default
to False. Select only valid columns or specify the value of numeric_only to
silence this warning.
  correlation_matrix = data.corr()
```

1. Positive Correlations:

- FTHG (Full Time Home Team Goals) has a strong positive correlation with HTHG (Half Time Home Team Goals) and HST (Home Team Shots on Target). This suggests that when a home team scores more goals in the full time, they also tend to score more goals in the first half and have more shots on target.
- FTAG (Full Time Away Team Goals) has a moderate positive correlation with HTAG (Half Time Away Team Goals) and AST (Away Team Shots on Target). Similar to the home team, higher away team goal scores are associated with more goals in the first half and more shots on target.

1. Negative Correlations:

- FTHG has a weak negative correlation with FTAG, indicating that when one team scores more goals, the other team tends to score fewer goals.

- HS (Home Team Shots) has a moderate negative correlation with AS (Away Team Shots), suggesting that when one team takes more shots, the other team tends to take fewer shots.

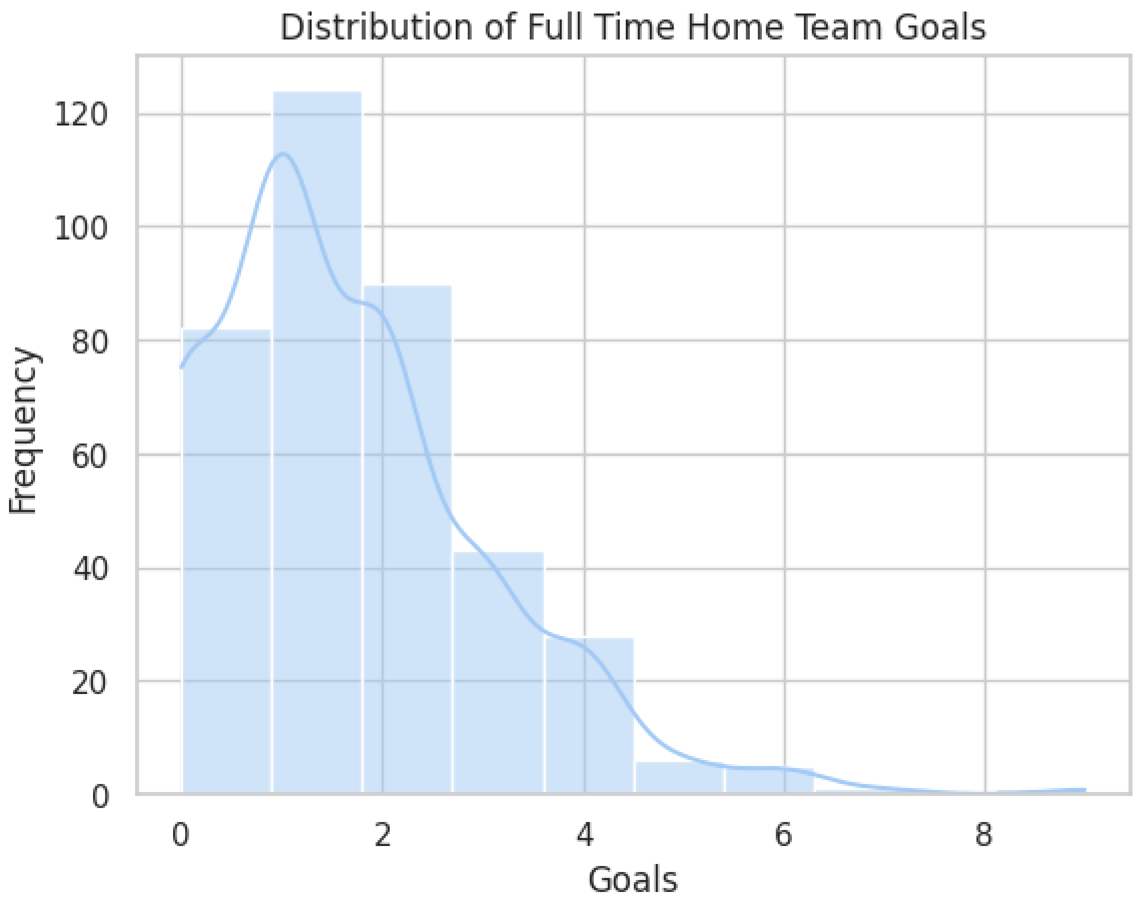
1. Other Insights:

- Yellow cards (HY, AY) show a weak negative correlation with home team goals (FTHG) and away team goals (FTAG). This suggests that higher goal-scoring teams might receive fewer yellow cards.
- Red cards (HR, AR) have a slight positive correlation with away team goals (FTAG), implying that matches with more away team goals might have slightly more red cards.

Distribution of Full Time Home Team Goals

We visualize the distribution of full-time home team goals using a histogram.

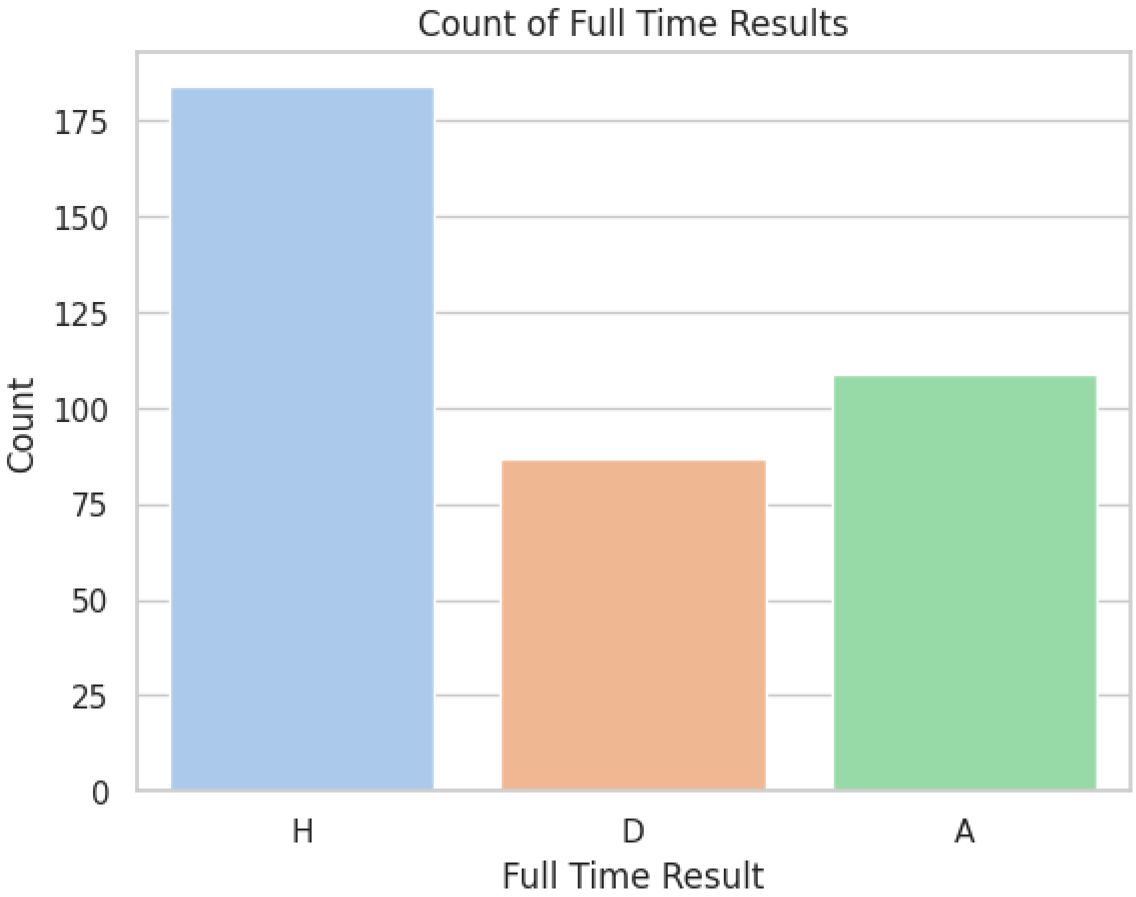
In [23]:
```
# Distribution of Full Time Home Team Goals
sns.histplot(data['FTHG'], bins=10, kde=True)
plt.title('Distribution of Full Time Home Team Goals')
plt.xlabel('Goals')
plt.ylabel('Frequency')
plt.show()
```



Count of Full Time Results

We visualize the count of full-time results using a count plot.

In [25]:
```
# Count plot of Full Time Results
sns.countplot(data=data, x='FTR', order=['H', 'D', 'A'])
plt.title('Count of Full Time Results')
plt.xlabel('Full Time Result')
plt.ylabel('Count')
plt.show()
```



Team Analysis

Goals and Wins Analysis

We analyze teams based on total goals scored and total wins.

In [26]:
```
# Calculate total goals scored by each team
home_goals = data.groupby('HomeTeam')['FTHG'].sum()
away_goals = data.groupby('AwayTeam')['FTAG'].sum()
total_goals = home_goals.add(away_goals, fill_value=0)

# Calculate total wins for each team
home_wins = data[data['FTR'] == 'H']['HomeTeam'].value_counts()
away_wins = data[data['FTR'] == 'A']['AwayTeam'].value_counts()
total_wins = home_wins.add(away_wins, fill_value=0)

# Create a DataFrame for team-wise analysis
team_stats = pd.DataFrame({
    'TotalGoals': total_goals,
    'TotalWins': total_wins,
})

# Sort teams by total goals in descending order
team_stats = team_stats.sort_values(by='TotalGoals', ascending=False)
```

```python
# Display the top teams by total goals and total wins
team_stats
```

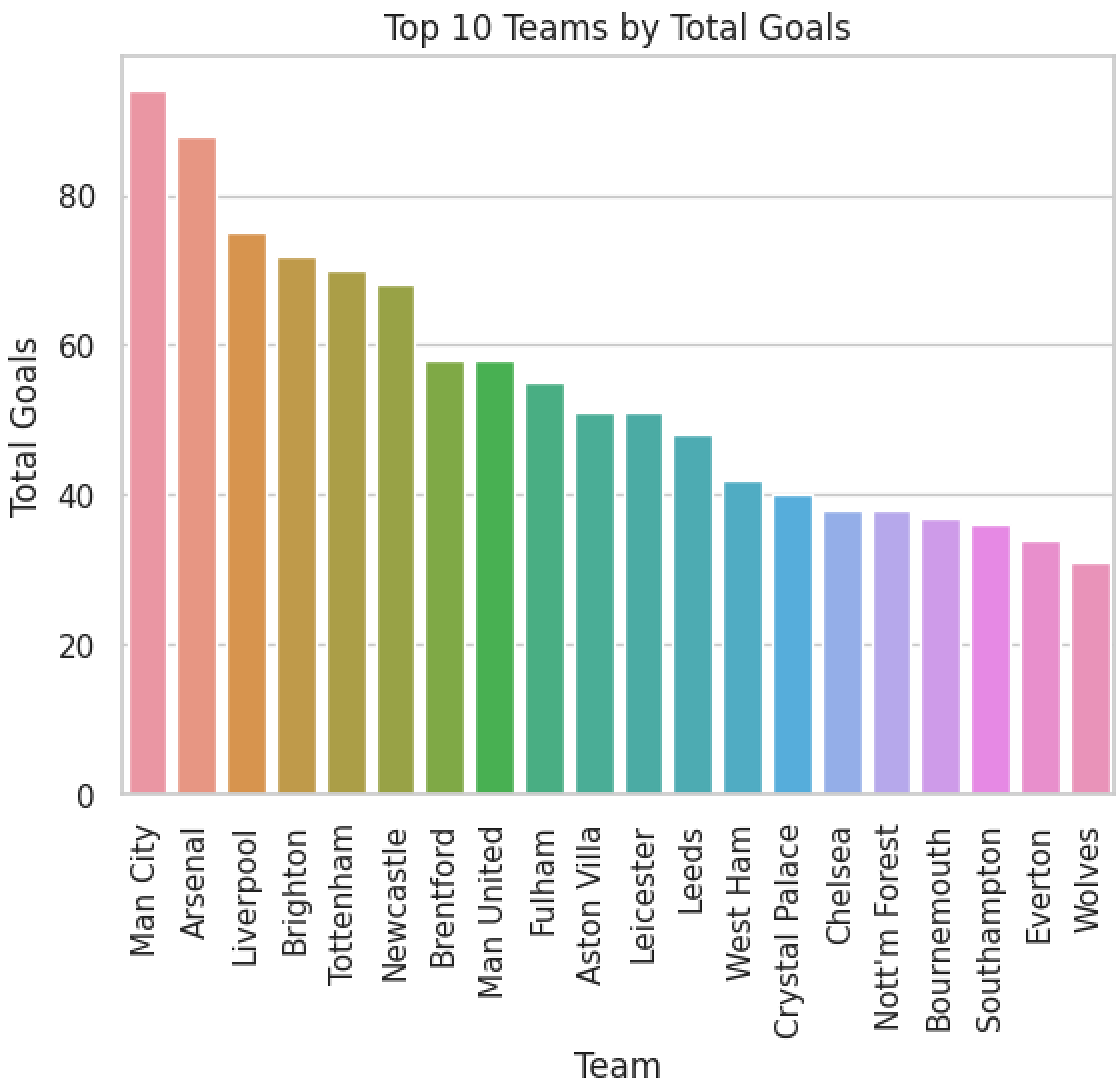
Out[26]:

	TotalGoals	TotalWins
Man City	94	28
Arsenal	88	26
Liverpool	75	19
Brighton	72	18
Tottenham	70	18
Newcastle	68	19
Brentford	58	15
Man United	58	23
Fulham	55	15
Aston Villa	51	18
Leicester	51	9
Leeds	48	7
West Ham	42	11
Crystal Palace	40	11
Chelsea	38	11
Nott'm Forest	38	9
Bournemouth	37	11
Southampton	36	6
Everton	34	8
Wolves	31	11

Top Teams Visualization

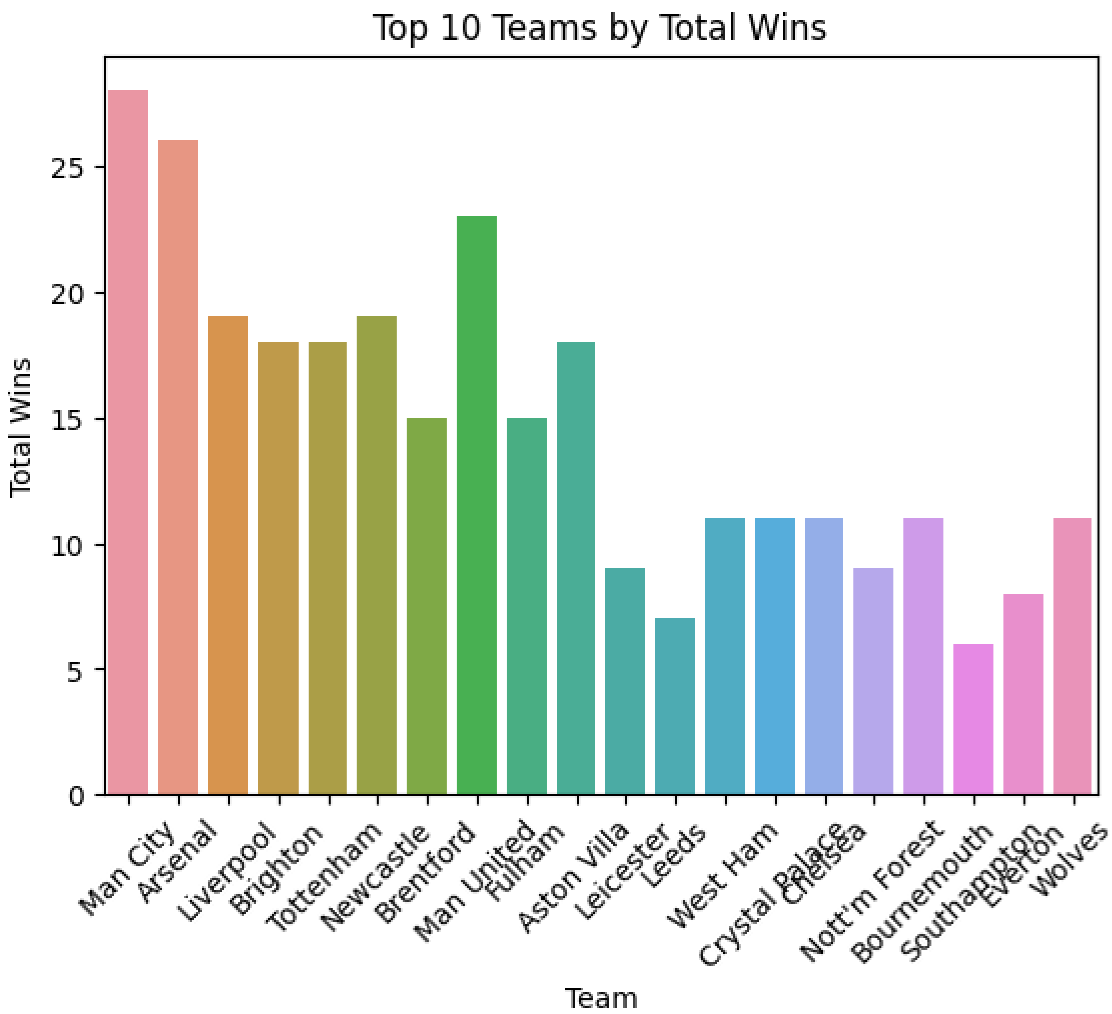
We visualize the top teams based on total goals and total wins.

In [27]:
```python
# Plotting top teams by total goals
sns.barplot(data=team_stats, x=team_stats.index, y='TotalGoals')
plt.xticks(rotation=90)
plt.title('Top 10 Teams by Total Goals')
plt.xlabel('Team')
plt.ylabel('Total Goals')
plt.show()
```



In [13]:

```
# Plotting top teams by total wins
sns.barplot(data=team_stats, x=team_stats.index, y='TotalWins')
plt.xticks(rotation=90)
plt.title('Top 10 Teams by Total Wins')
plt.xlabel('Team')
plt.ylabel('Total Wins')
plt.show()
```



Additional Statistics and Team Analysis

We calculate additional summary statistics per match and perform team-wise analysis.

In [14]:

```
# Calculate additional summary statistics
data['TotalGoals'] = data['FTHG'] + data['FTAG']
data['TotalShots'] = data['HS'] + data['AS']
data['TotalShotsOnTarget'] = data['HST'] + data['AST']
data['TotalCorners'] = data['HC'] + data['AC']
data['TotalFouls'] = data['HF'] + data['AF']
data['TotalYellowCards'] = data['HY'] + data['AY']
data['TotalRedCards'] = data['HR'] + data['AR']

# Calculate average statistics per match
team_stats = data.groupby('HomeTeam').agg(
    AverageGoals=('TotalGoals', 'mean'),
    WinPercentage=('FTR', lambda x: (x == 'H').mean() * 100),
    DrawPercentage=('FTR', lambda x: (x == 'D').mean() * 100),
    LossPercentage=('FTR', lambda x: (x == 'A').mean() * 100),
    AverageShots=('TotalShots', 'mean'),
    AverageShotsOnTarget=('TotalShotsOnTarget', 'mean'),
    AverageCorners=('TotalCorners', 'mean'),
    AverageFouls=('TotalFouls', 'mean'),
    AverageYellowCards=('TotalYellowCards', 'mean'),
    AverageRedCards=('TotalRedCards', 'mean'),
    CleanSheetPercentage=('FTAG', lambda x: (x == 0).mean() * 100)
).reset_index()

# Sort teams by AverageGoals in descending order
```

```
team_stats = team_stats.sort_values(by='AverageGoals', ascending=False)
team_stats = team_stats.reset_index(drop=True)
team_stats.index += 1

# Display team-wise statistics
team_stats
```

Out[14]:

	HomeTeam	AverageGoals	WinPercentage	DrawPercentage	LossPercentage	AverageS
1	Arsenal	4.105263	73.684211	15.789474	10.526316	26.05
2	Man City	4.052632	89.473684	5.263158	5.263158	23.47
3	Leeds	3.315789	26.315789	36.842105	36.842105	26.57
4	Liverpool	3.315789	68.421053	26.315789	5.263158	25.47
5	Tottenham	3.263158	63.157895	5.263158	31.578947	27.05
6	Fulham	3.157895	42.105263	26.315789	31.578947	24.52
7	Brighton	3.052632	52.631579	21.052632	26.315789	27.21
8	Southampton	2.947368	10.526316	26.315789	63.157895	25.21
9	Aston Villa	2.842105	63.157895	10.526316	26.315789	21.47
10	Brentford	2.789474	52.631579	36.842105	10.526316	24.78
11	Nott'm Forest	2.684211	42.105263	31.578947	26.315789	23.47
12	Newcastle	2.631579	57.894737	31.578947	10.526316	26.10
13	West Ham	2.631579	42.105263	21.052632	36.842105	25.52
14	Leicester	2.631579	26.315789	21.052632	52.631579	24.10
15	Bournemouth	2.526316	31.578947	21.052632	47.368421	25.68
16	Man United	2.421053	78.947368	15.789474	5.263158	29.05
17	Crystal Palace	2.315789	36.842105	36.842105	26.315789	24.00
18	Everton	2.263158	31.578947	15.789474	52.631579	25.47
19	Chelsea	2.052632	31.578947	36.842105	31.578947	24.68
20	Wolves	2.052632	47.368421	15.789474	36.842105	25.31

1. AverageGoals: The average number of goals scored by a team per match.
2. WinPercentage: The percentage of matches won by a team.
3. DrawPercentage: The percentage of matches drawn by a team.
4. LossPercentage: The percentage of matches lost by a team.
5. AverageShots: The average number of shots taken by a team per match.
6. AverageShotsOnTarget: The average number of shots on target by a team per match.
7. AverageCorners: The average number of corners awarded to a team per match.

8. AverageFouls: The average number of fouls committed by a team per match.

9. AverageYellowCards: The average number of yellow cards received by a team per match.

10. AverageRedCards: The average number of red cards received by a team per match.

11. AverageBookingPoints: The average number of booking points (10 for yellow, 25 for red) accumulated by a team per match.

12. CleanSheetPercentage: The percentage of matches in which a team did not concede any goals.

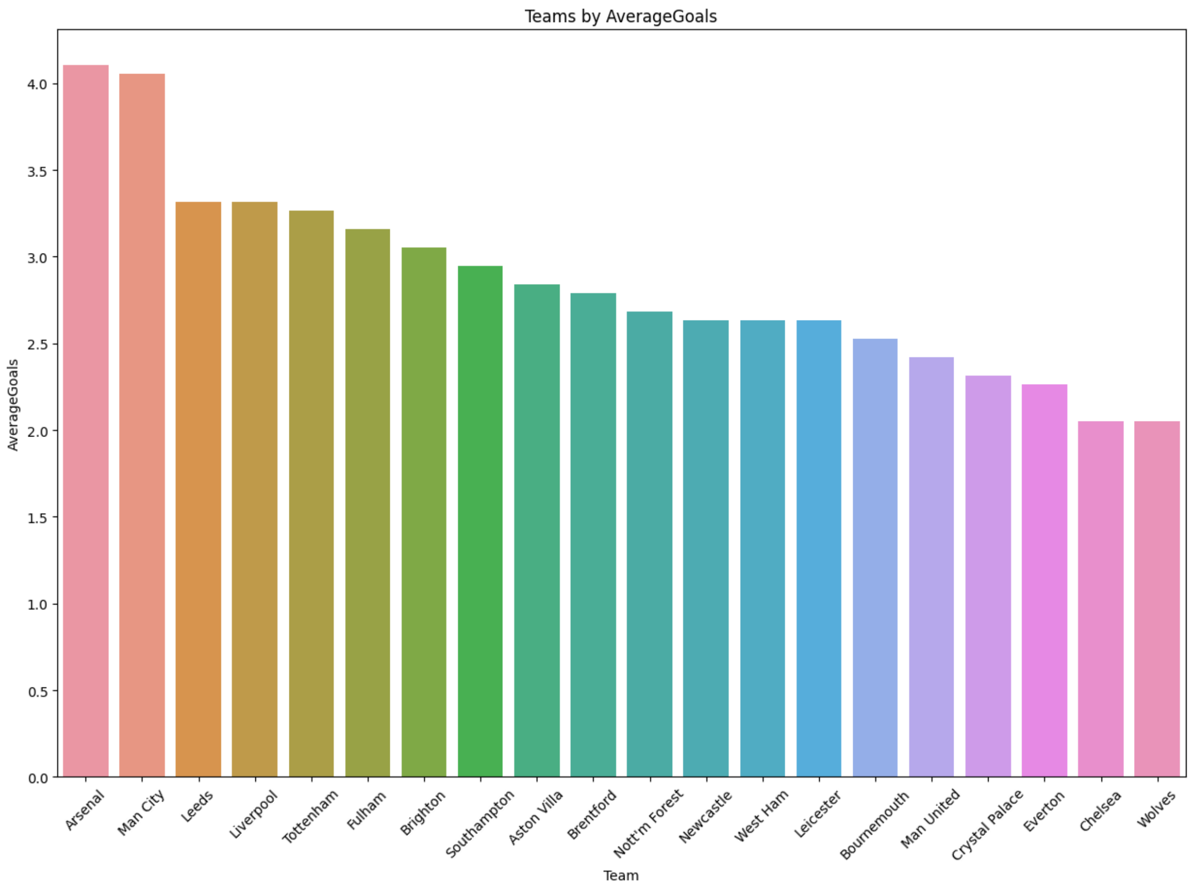
13. TotalGoals: The total number of goals scored by a team across all matches.

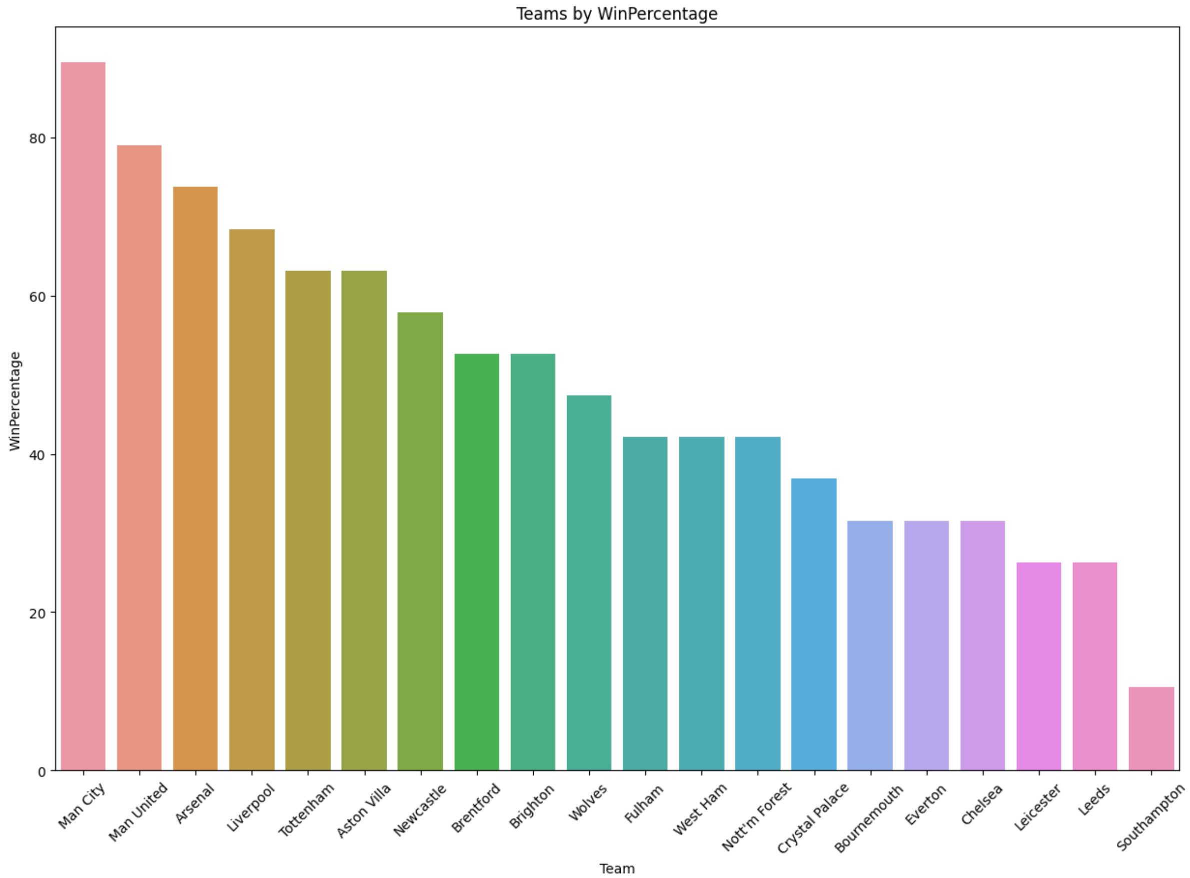
14. TotalWins: The total number of matches won by a team across all matches.

Visualization of Team Statistics

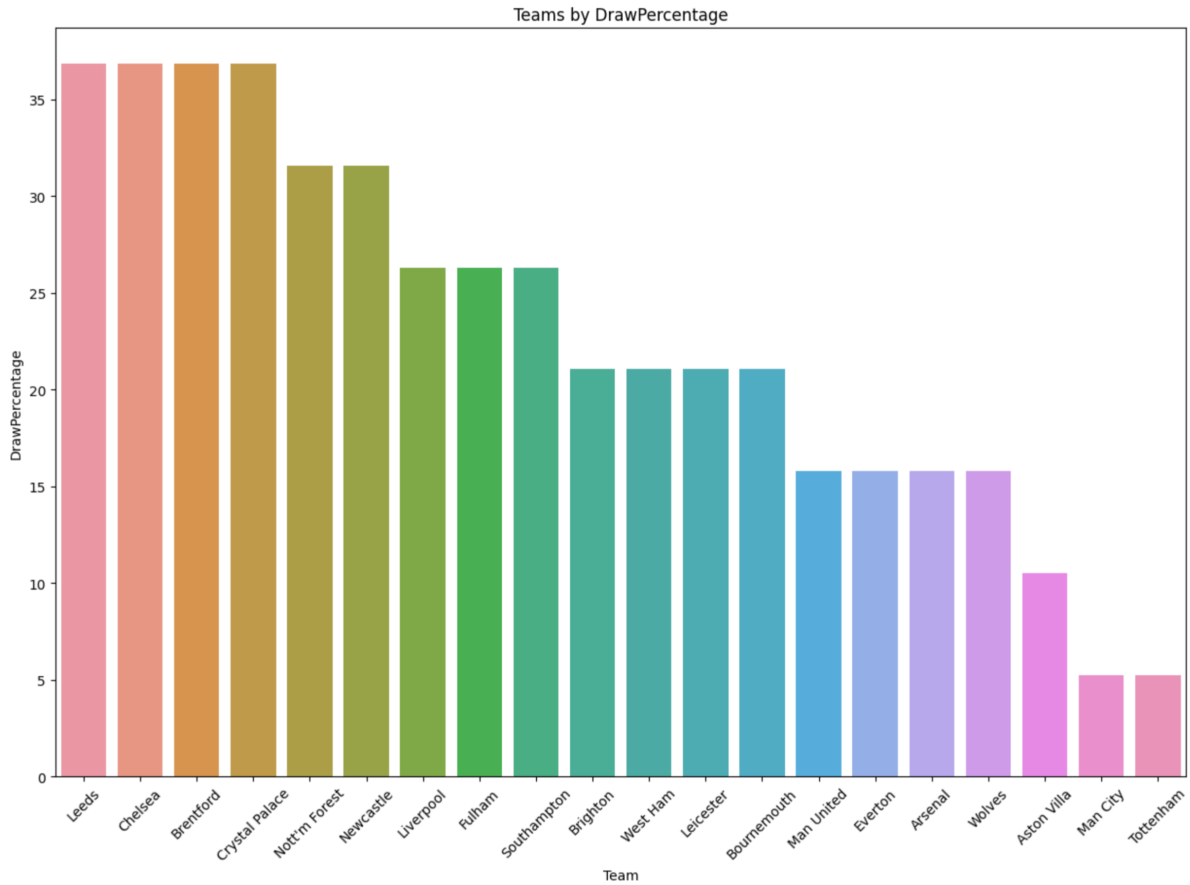
We visualize various statistics for each team.

In [15]:
```
for col in team_stats.columns[1:]:
    plt.figure(figsize=(15, 10))
    sns.barplot(data=team_stats.sort_values(by=col, ascending=False), x='Hom
    plt.xticks(rotation=45)
    plt.title(f'Teams by {col}')
    plt.xlabel('Team')
    plt.ylabel(col)
    plt.show()
```

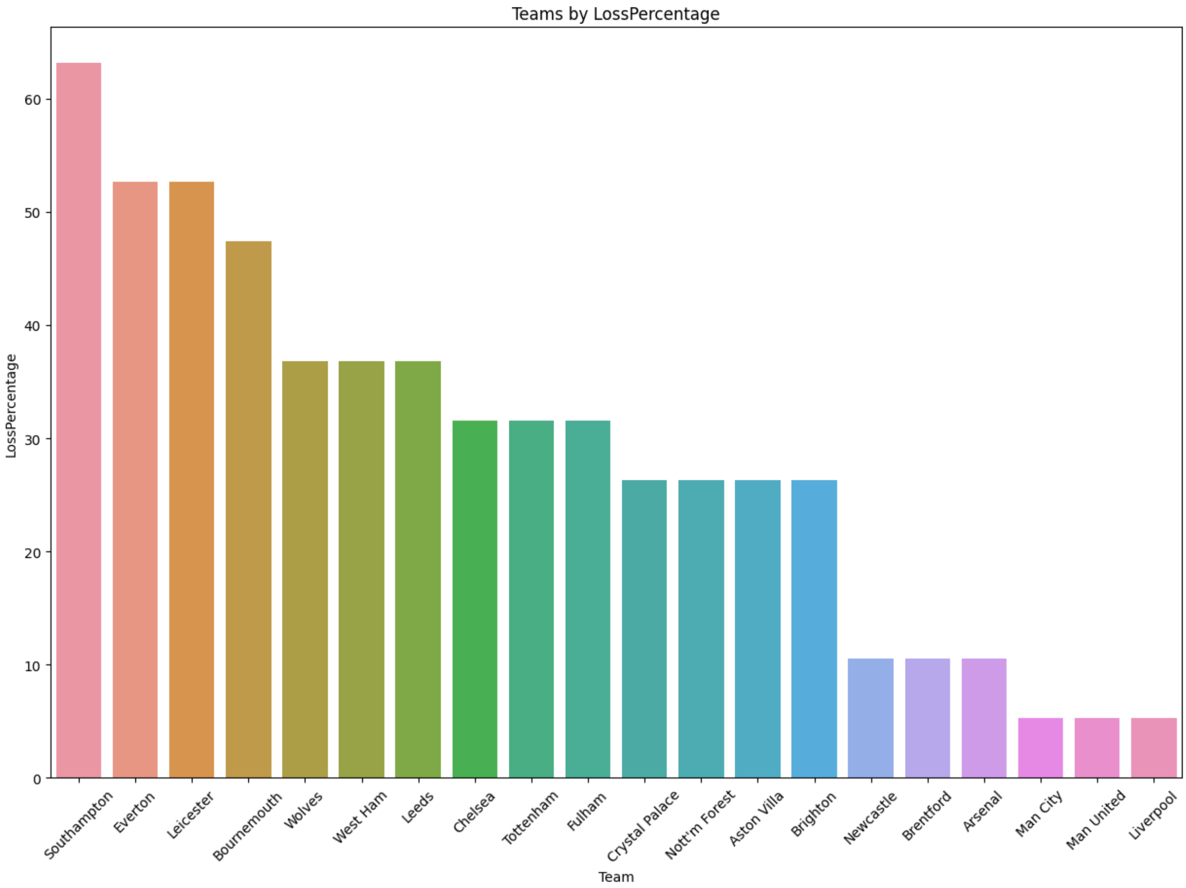


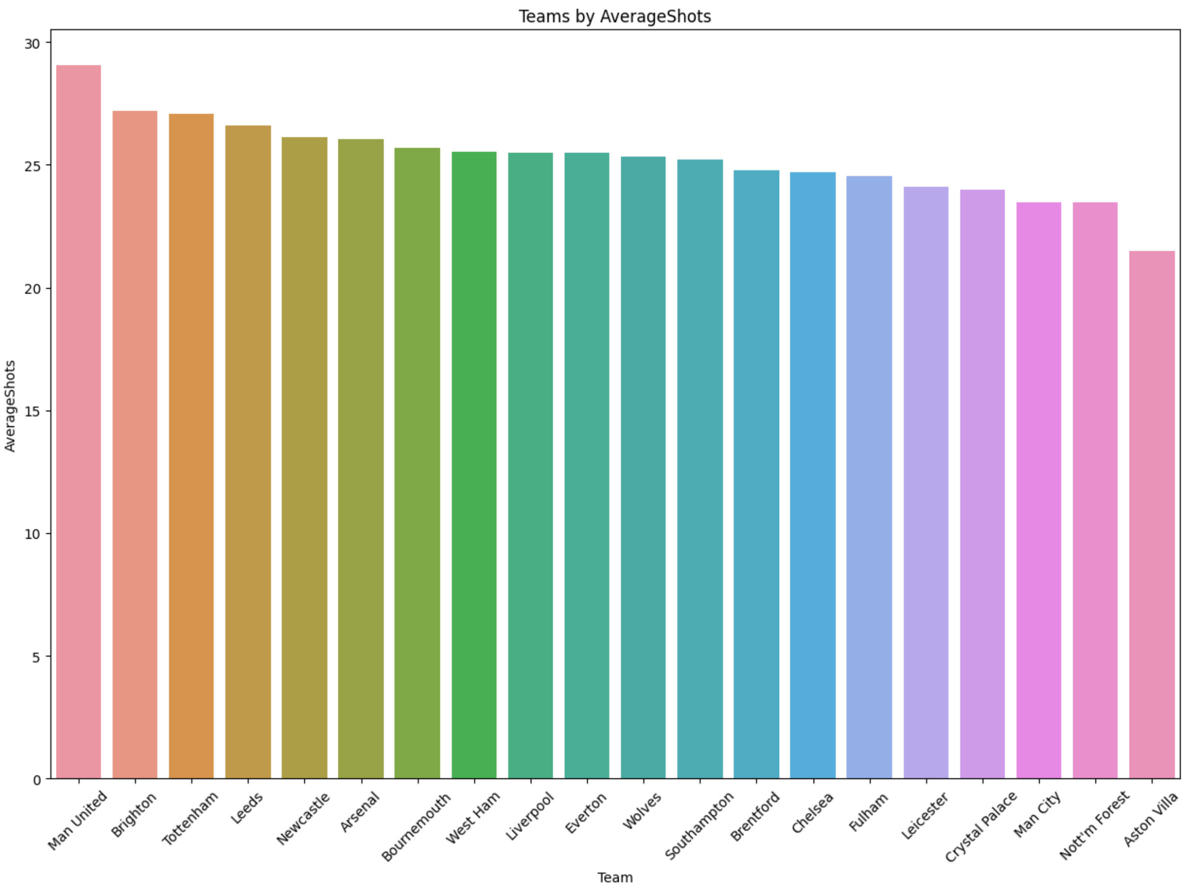


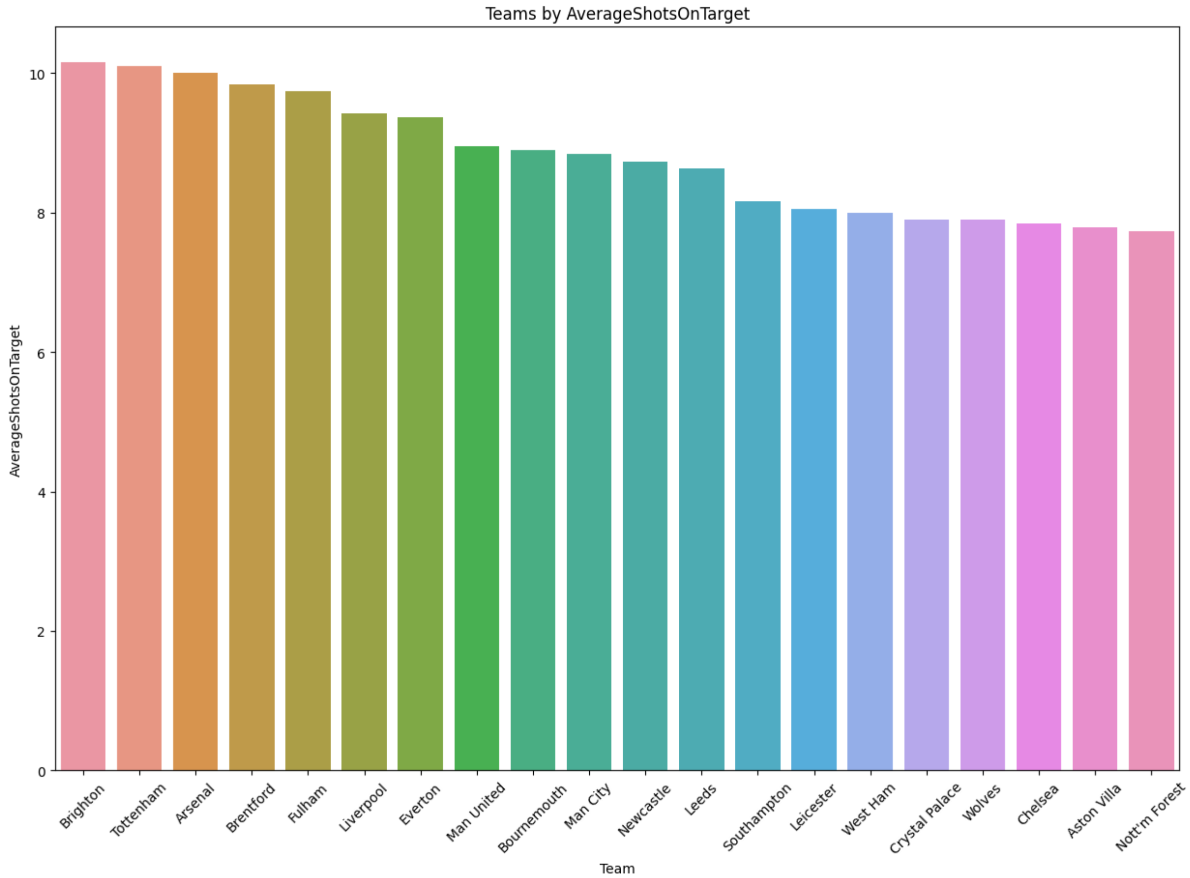
Teams by WinPercentage
WinPercentage
0
20
40
60
80
Man City
Man United
Arsenal
Liverpool
Tottenham
Aston Villa
Newcastle
Brentford
Brighton
Wolves
Fulham
West Ham
Nott'm Forest
Crystal Palace
Bournemouth
Everton
Chelsea
Leicester
Leeds
Southampton
Team

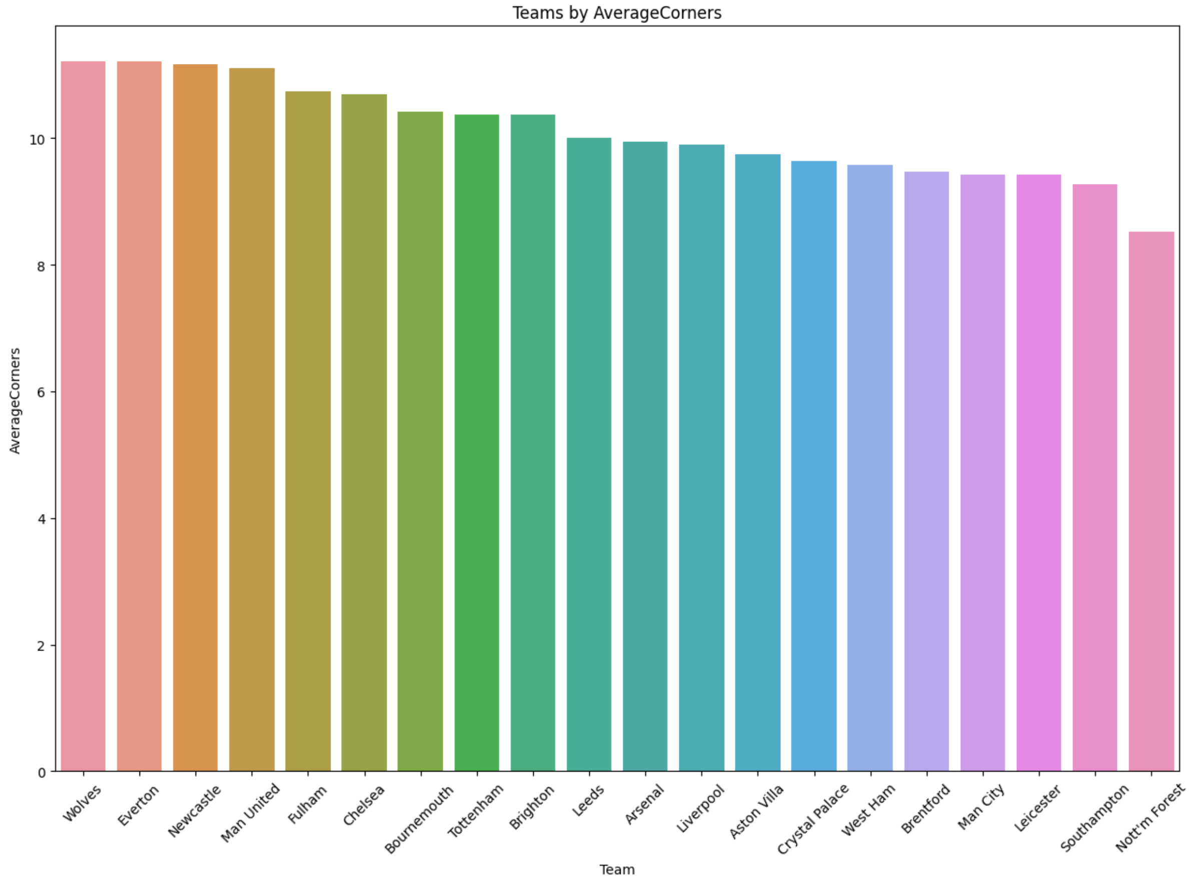


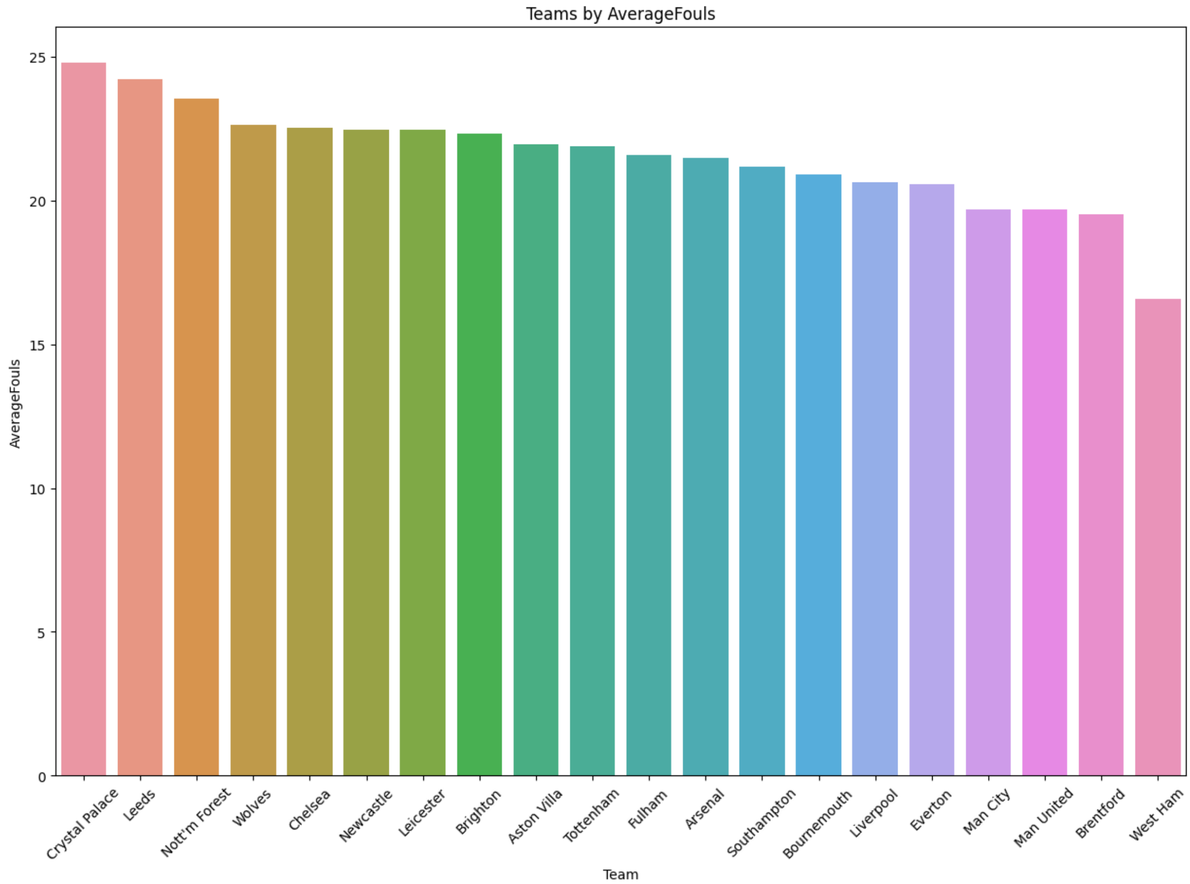
Teams by DrawPercentage
DrawPercentage
0
5
10
15
20
25
30
35
Leeds
Chelsea
Brentford
Crystal Palace
Nott'm Forest
Newcastle
Liverpool
Fulham
Southampton
Brighton
West Ham
Leicester
Bournemouth
Man United
Everton
Arsenal
Wolves
Aston Villa
Man City
Tottenham
Team

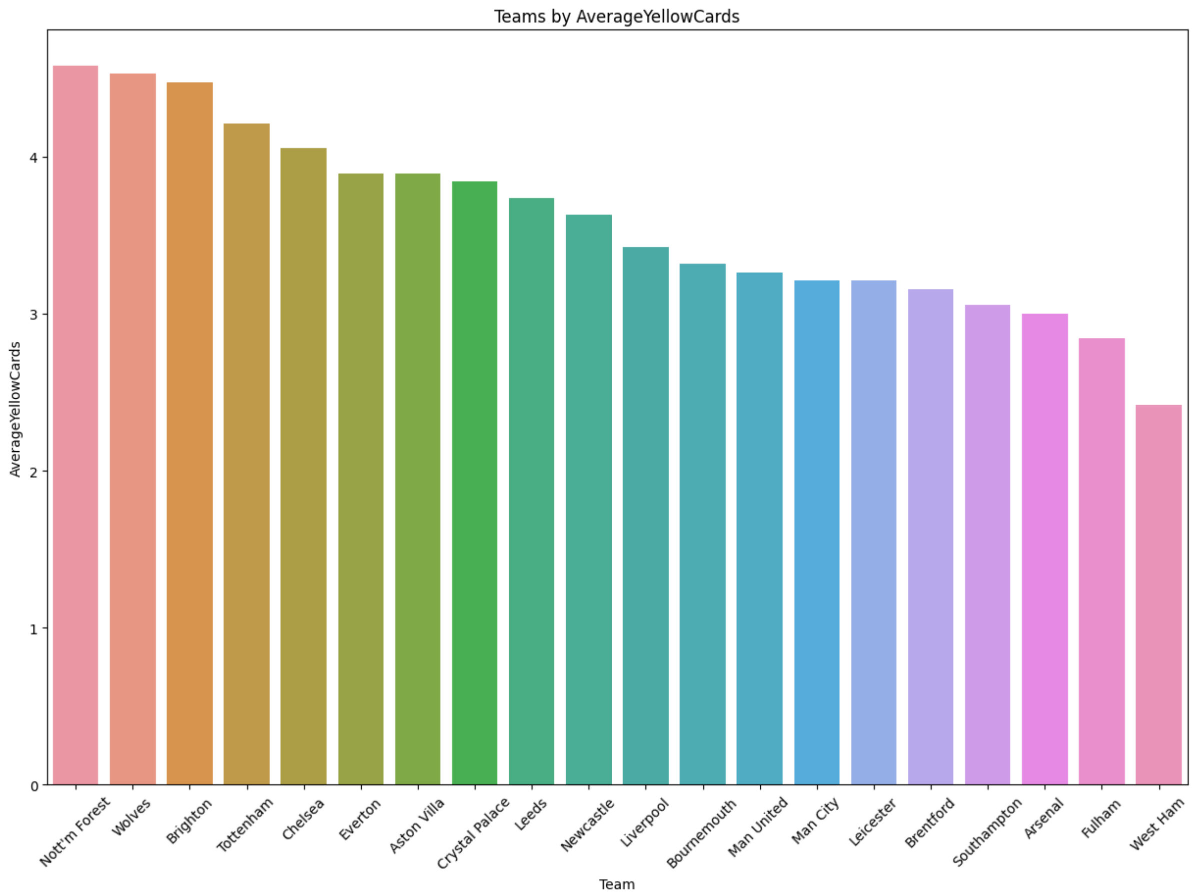

Teams by LossPercentage
LossPercentage
Team
0
10
20
30
40
50
60
Southampton
Everton
Leicester
Bournemouth
Wolves
West Ham
Leeds
Chelsea
Tottenham
Fulham
Crystal Palace
Nott'm Forest
Aston Villa
Brighton
Newcastle
Brentford
Arsenal
Man City
Man United
Liverpool

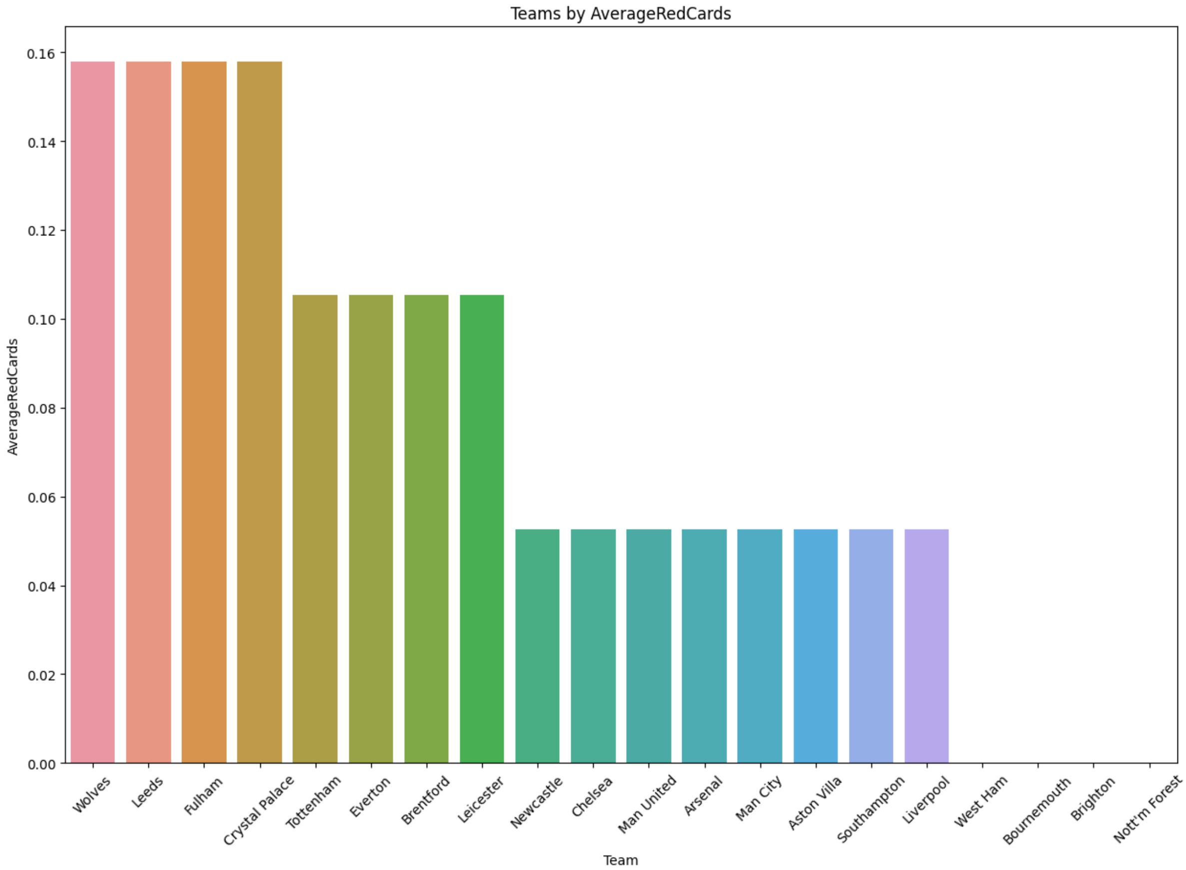

Teams by AverageShots
AverageShots
Team
0
5
10
15
20
25
30
Man United
Brighton
Tottenham
Leeds
Newcastle
Arsenal
Bournemouth
West Ham
Liverpool
Everton
Wolves
Southampton
Brentford
Chelsea
Fulham
Leicester
Crystal Palace
Man City
Nott'm Forest
Aston Villa

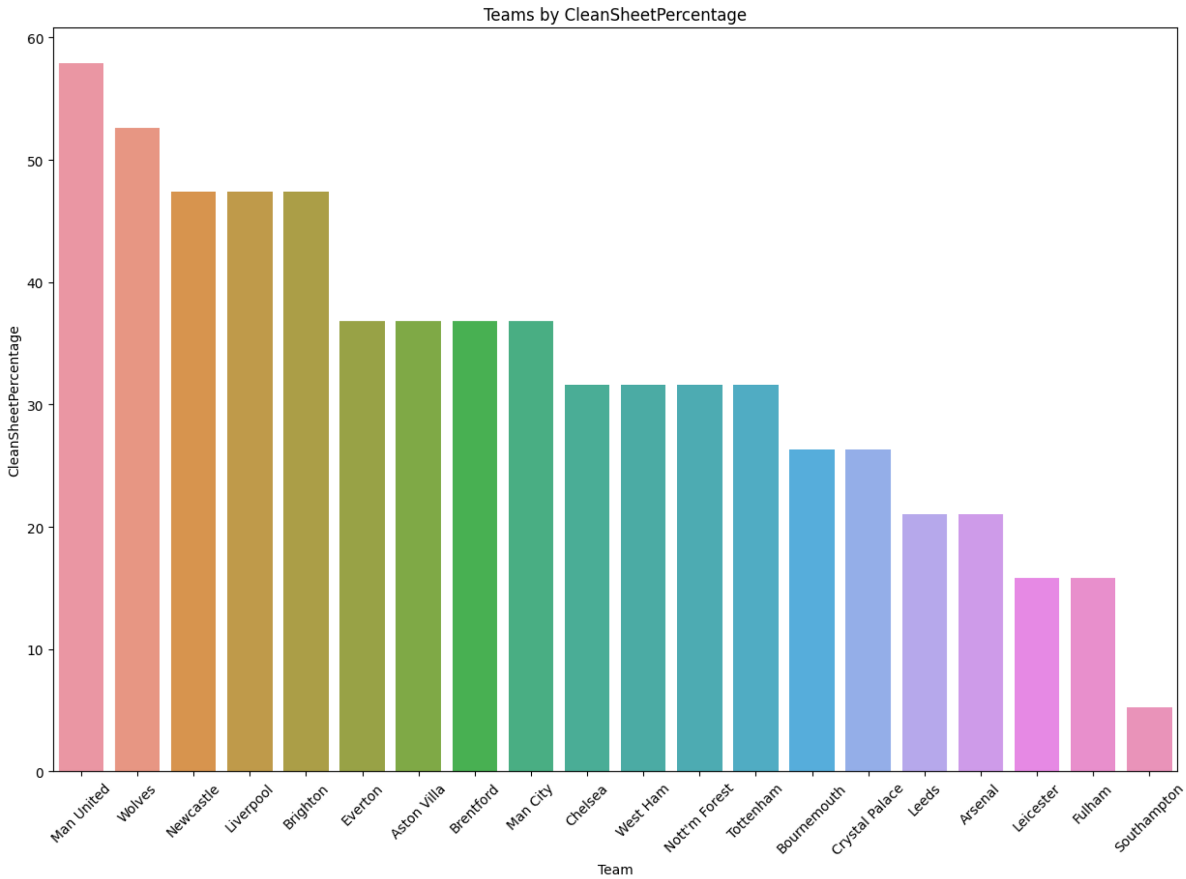

Teams by AverageShotsOnTarget
AverageShotsOnTarget
0
2
4
6
8
10
Brighton
Tottenham
Arsenal
Brentford
Fulham
Liverpool
Everton
Man United
Bournemouth
Man City
Newcastle
Leeds
Southampton
Leicester
West Ham
Crystal Palace
Wolves
Chelsea
Aston Villa
Nott'm Forest
Team


Teams by AverageCorners
AverageCorners
0
2
4
6
8
10
Wolves
Everton
Newcastle
Man United
Fulham
Chelsea
Bournemouth
Tottenham
Brighton
Leeds
Arsenal
Liverpool
Aston Villa
Crystal Palace
West Ham
Brentford
Man City
Leicester
Southampton
Nott'm Forest
Team


Teams by AverageFouls
AverageFouls
0
5
10
15
20
25
Crystal Palace
Leeds
Nott'm Forest
Wolves
Chelsea
Newcastle
Leicester
Brighton
Aston Villa
Tottenham
Fulham
Arsenal
Southampton
Bournemouth
Liverpool
Everton
Man City
Man United
Brentford
West Ham
Team


Teams by AverageYellowCards
AverageYellowCards
0
1
2
3
4
Nott'm Forest
Wolves
Brighton
Tottenham
Chelsea
Everton
Aston Villa
Crystal Palace
Leeds
Newcastle
Liverpool
Bournemouth
Man United
Man City
Leicester
Brentford
Southampton
Arsenal
Fulham
West Ham
Team





Conclusion

This notebook showcases an exploratory data analysis of football match data. It covers summary statistics, correlation analysis, team-wise analysis, and various visualizations to gain insights into team performance, goal scoring, wins, and other key attributes.