In [ ]:

```python
#Import the necessary libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

Full_df=pd.read_csv(r'C:\Users\B InfoSoft\Desktop\Code Projects\PANDAS\TheArsenal.csv') #Read th
#Set display options to make the output clearer and less cluttered
pd.set_option('display.max_rows',8)
pd.set_option('display.max_columns',25)

Full_df
```

Out[ ]:

	Matchweek	Day	Time	Venue	Attendance	Referee	Opponent	Result	Possession	Goals for	Go agai
0	Matchweek 1	Fri	20:00	Away	25,286	Anthony Taylor	Crystal Palace	W	44	2	
1	Matchweek 2	Sat	15:00	Home	60,033	Darren England	Leicester City	W	50	4	
2	Matchweek 3	Sat	17:30	Away	10,423	Craig Pawson	Bournemouth	W	57	3	
3	Matchweek 4	Sat	17:30	Home	60,164	Jarred Gillett	Fulham	W	71	2	
...	...	...	...	...	...	...	...	...	...	...	
34	Matchweek 35	Sun	16:30	Away	50,267	Chris Kavanagh	Newcastle Utd	W	45	2	
35	Matchweek 36	Sun	16:30	Home	60,139	Andy Madley	Brighton	L	41	0	
36	Matchweek 37	Sat	17:30	Away	29,514	Anthony Taylor	Nott'ham Forest	L	81	0	
37	Matchweek 38	Sun	16:30	Home	60,095	Andre Marriner	Wolves	W	51	5	

38 rows × 25 columns

This dataframe contains all the matchweeks for Arsenal but we only need the data for the top 6 teams. We will first look at how the top six teams are registered in the dataframe (their name can be found in the 'Opponent' column). Then, we create a list and pass it into a filter where it displays only the data containing the names of those teams inside the previously mentioned list.

Finally, we create a new abridged dataframe containing only the data versus the top teams in the Premier League.

In [ ]:

```python
#Look into the opponent column
Full_df['Opponent'].unique()
```

```
Out[ ]: array(['Crystal Palace', 'Leicester City', 'Bournemouth', 'Fulham',
               'Aston Villa', 'Manchester Utd', 'Brentford', 'Tottenham',
               'Liverpool', 'Leeds United', 'Southampton', "Nott'ham Forest",
               'Chelsea', 'Wolves', 'West Ham', 'Brighton', 'Newcastle Utd',
               'Everton', 'Manchester City'], dtype=object)
```

We need the following teams: Manchester City, Manchester Utd, Tottenham, Chelsea, Liverpool

```
In [ ]: #Create the the list containing the team names as they are written in the dataframe
        Top_6=['Manchester City','Manchester Utd','Chelsea','Liverpool','Tottenham']
        #Create a filter
        Filter= Full_df['Opponent'].isin(Top_6)
        #Pass the filter to the full dataframe
        Full_df[Filter]
```

Out[ ]:

	Matchweek	Day	Time	Venue	Attendance	Referee	Opponent	Result	Possession	Goals for	Goals against
5	Matchweek 6	Sun	16:30	Away	73,431	Paul Tierney	Manchester Utd	L	60	1	3
7	Matchweek 9	Sat	12:30	Home	60,278	Anthony Taylor	Tottenham	W	64	3	1
8	Matchweek 10	Sun	16:30	Home	60,059	Michael Oliver	Liverpool	W	43	3	2
12	Matchweek 15	Sun	12:00	Away	40,142	Michael Oliver	Chelsea	W	55	1	0
...	...	...	...	...	...	...	...	...	...	...	...
21	Matchweek 12	Wed	19:30	Home	60,276	Anthony Taylor	Manchester City	L	63	1	3
29	Matchweek 30	Sun	16:30	Away	53,267	Paul Tierney	Liverpool	D	41	2	2
32	Matchweek 33	Wed	20:00	Away	53,482	Michael Oliver	Manchester City	L	48	1	4
33	Matchweek 34	Tue	20:00	Home	60,144	Robert Jones	Chelsea	W	55	3	1

10 rows × 25 columns

Now that we have made a filter. Let's create a new dataframe containing only the data we need

```
In [ ]: #Create the new dataframe
        df= Full_df[Filter]
```

Now, let's have a look at the columns and their types for any needed transformation

```
In [ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10 entries, 5 to 33
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Matchweek           10 non-null     object 
 1   Day                 10 non-null     object 
 2   Time                10 non-null     object 
 3   Venue               10 non-null     object 
 4   Attendance          10 non-null     object 
 5   Referee             10 non-null     object 
 6   Opponent            10 non-null     object 
 7   Result              10 non-null     object 
 8   Possession          10 non-null     int64  
 9   Goals for           10 non-null     int64  
 10  Goals against       10 non-null     int64  
 11  Expectec goals      10 non-null     float64
 12  Shots               10 non-null     int64  
 13  Shots on target     10 non-null     int64  
 14  Passes              10 non-null     int64  
 15  Passes completed    10 non-null     int64  
 16  Short passes        10 non-null     int64  
 17  Medium passes       10 non-null     int64  
 18  Long passes         10 non-null     int64  
 19  Corners             10 non-null     int64  
 20  Tackles             10 non-null     int64  
 21  Tackles won         10 non-null     int64  
 22  Fouls               10 non-null     int64  
 23  Penalties attempted  10 non-null     int64  
 24  Penalties made      10 non-null     int64  
dtypes: float64(1), int64(16), object(8)
memory usage: 2.0+ KB
```

From the following output, we can classify the changes that need to be made as the following:

- The columns Day and Result can be transformed from object to category to improve and optimize the performance.
- The column Attendance is more appropriate being an integer rather than an object to perform numerical aggregations.

Further changes can be made to make the performance better. For example, the integer columns can be changed from 64 bit to 16 or 8 since they don't contain a large range of numbers but since this is already a small dataset, it's not necessary, yet it's worth noting that it's a beneficial habit to embed in your workflow

In [ ]:
```
#Change the type of columns
df['Day']=df['Day'].astype('category')
df['Result']=df['Result'].astype('category')

df['Attendance']=df['Attendance'].str.replace(',','').astype(int)
```

```
C:\Users\B InfoSoft\AppData\Local\Temp\ipykernel_12696\1138655964.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
exing.html#returning-a-view-versus-a-copy
  df['Day']=df['Day'].astype('category')
C:\Users\B InfoSoft\AppData\Local\Temp\ipykernel_12696\1138655964.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
exing.html#returning-a-view-versus-a-copy
  df['Result']=df['Result'].astype('category')
C:\Users\B InfoSoft\AppData\Local\Temp\ipykernel_12696\1138655964.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
exing.html#returning-a-view-versus-a-copy
  df['Attendance']=df['Attendance'].str.replace(',','').astype(int)
```

Now that the dataframe is ready for analysis. We are going to perform some comparison between the performance vs the top 6. The following insights that I'm aiming to extract are:

- The number of goals Arsenal scored vs the top six
- The number of goals Arsenal scored vs the top six
- The number of wins, draws and losses vs the top six
- The total amount of points gained vs the top six teams
- The total number of attendances
- The average possession percentage vs the top six
- The average passes made vs the top six
- The average expected goals vs top six
- The match result based on the opponents
- The match result based on the venue
- The percentage of goals for against top 6 from the total goals scored
- The percentage of goals against against top 6 from the total goals conceded
- The average possession vs the top six compared to the overall average possession
- The average of goals scored and conceded against top six compared to the total average
- The shot accuracy percentage
- The passing accuracy percentage
- The percentage of attendences for the top 6 teams compared to the total number of attendances

I already have data available for the overall stats from a previous project I have made so all that is left is comparing the data and see any useful insights.

First off, let's create a filter for home matches and away matches since it's going to be used on a number of insights we listed

In [ ]:
```python
#Creating a home filter
Home_matches= df['Venue']=='Home'
#Creating an away filter
Away_matches= df['Venue']=='Away'
```

In [ ]:

```python
#Create a filter for numerical columns only to perform aggregate functions
numerical_columns = df.select_dtypes(include=['int', 'float'])
#Create a view for the aggregate functions sum and mean
numerical_columns.aggregate(['sum','mean'])
```

Out[ ]:

	Attendance	Possession	Goals for	Goals against	Expectec goals	Shots	Shots on target	Passes	Passes completed	Short passes	Medi pas
sum	583274.0	535.0	20.0	18.0	18.60	143.0	46.0	5107.0	4158.0	2089.0	21
mean	58327.4	53.5	2.0	1.8	1.86	14.3	4.6	510.7	415.8	208.9	2

A portion of the questions we listed above fall in the bin of either sum (total) or mean (average). Let's have a look at the following output:

- Arsenal managed to score 20 goals versus the top six teams and conceded 18, resulting in a goal difference of a positive 2.
- Arsenal culminated a total attendance record of 583k spectators in the 10 matches against the top teams.
- Arsenal had an average possession of 53.5, meaning that they had a small edge in ball controlling terms.
- On average, Arsenal had an XG performance of 1.8 ,suggesting that the team had a solid offensive effort against the top 6 teams from various variables like attacking tactics, player positioning, and overall gameplay were effective in generating high quality scoring chances.

In [ ]:

```python
#Calculation the results versus the top teams
df['Result'].value_counts()
```

Out[ ]:

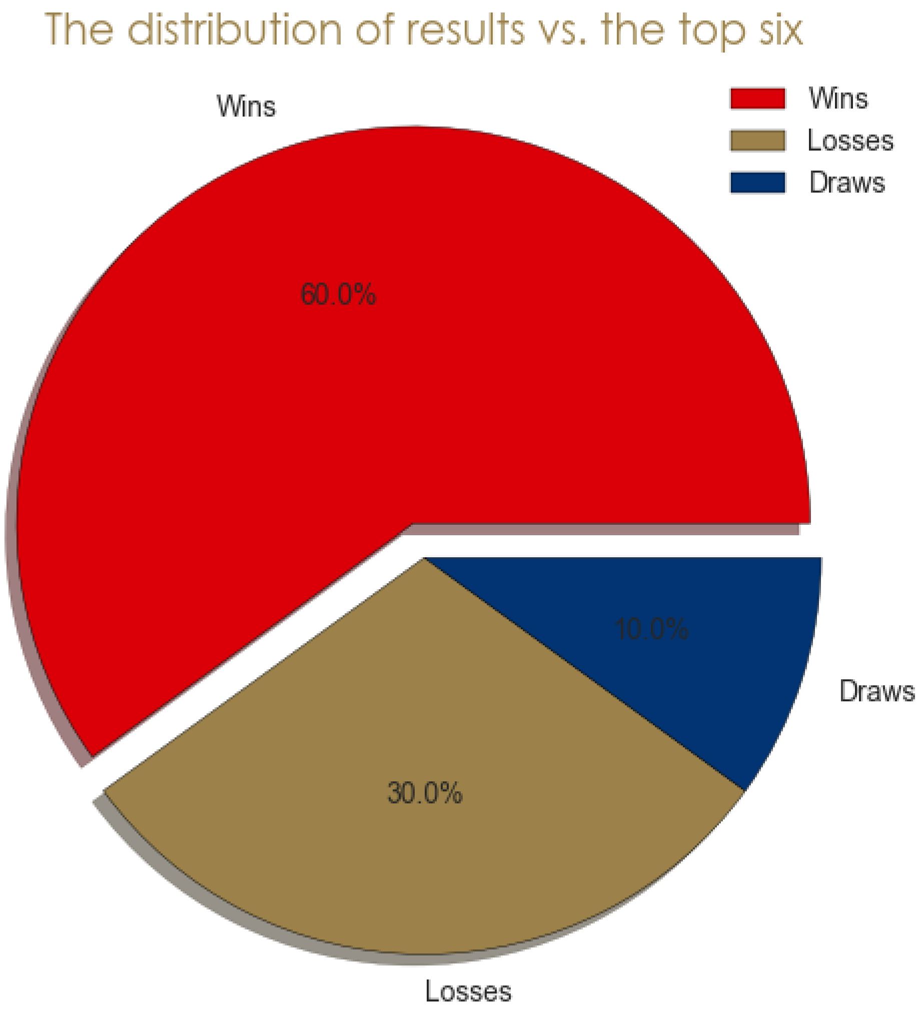
```
W    6
L    3
D    1
Name: Result, dtype: int64
```

- From the 10 matches against the top six teams, Arsenal won more than half the matches, 6 matches in total.
- Arsenal lost 6 matches during the season, half of which were against the big teams.
- Only one draw was recorded against the top six.

In [ ]:

```python
#Pie chart representing the results of the matches

plt.style.use('seaborn')

Results=df['Result'].value_counts().tolist()
Colors = ['#DB0007','#9C824A', '#023474']
Labels= ['Wins','Losses','Draws']

plt.pie(Results,labels=Labels,colors=Colors,wedgeprops={'edgecolor':'black'},shadow=True,autopct
        explode=[0.09,0,0])

plt.legend()

plt.title('The distribution of results vs. the top six',fontdict={'fontname':'Century Gothic','s
plt.tight_layout()
plt.show()
```

```
C:\Users\B InfoSoft\AppData\Local\Temp\ipykernel_12696\2866080438.py:3: MatplotlibDeprecationWarn
ing: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer corresp
ond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0_8-<style
>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn')
```



In [ ]:
```
#Create a column of points gained from each match
df['Points gained']=df['Result'].map({'W':3,'D':1,'L':0}).astype(int)
df['Points gained'].sum()
```

```
C:\Users\B InfoSoft\AppData\Local\Temp\ipykernel_12696\1417208088.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
exing.html#returning-a-view-versus-a-copy
  df['Points gained']=df['Result'].map({'W':3,'D':1,'L':0}).astype(int)
```

Out[ ]:
```
19
```

Out of 10 matches, Arsenal picked up 19 points

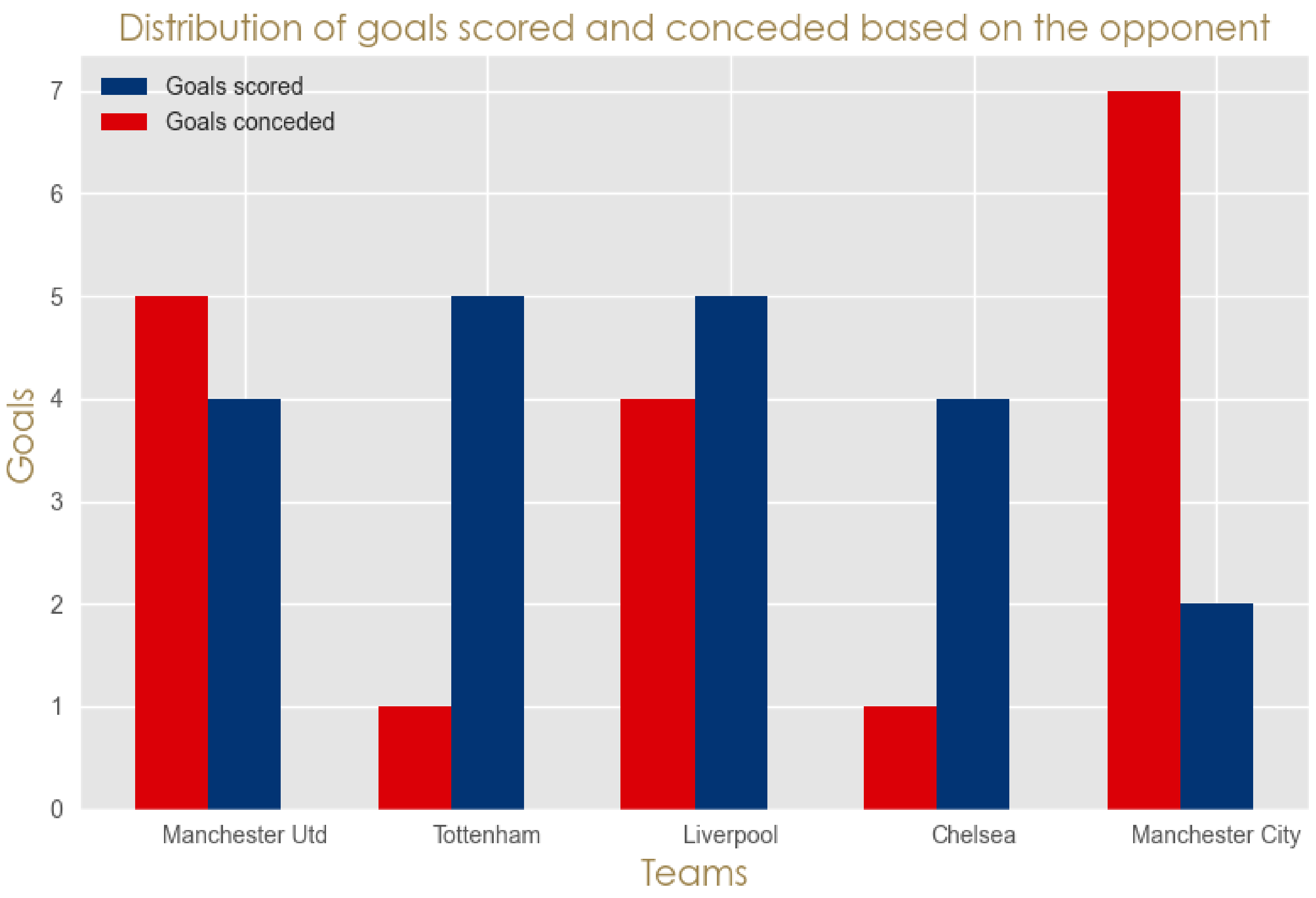
In [ ]:
```
#Calculating the results based on the opponent and venue

df[['Opponent','Result']].value_counts()
```

```
Out[ ]: Opponent          Result
        Chelsea           W         2
        Manchester City   L         2
        Tottenham         W         2
        Liverpool         D         1
                          W         1
        Manchester Utd    L         1
                          W         1
        dtype: int64
```

Based on the following output we can conclude to:

- Arsenal managed to do the double on chelsea and Tottenham, winning both in the home and away matches.
- Manchester City did the double on Arsenal, winning against the Gunners in both occasions which was a vital point to win the league.
- Arsenal managed to win against Liverpool in the home fixture and also scrapped out a 2-2 draw in the away fixture.
- Arsenal and Manchester United both got 3 points from each other with Arsenal losing the away fixture 3-1 then making a comeback in the Emirates to win 3-2

```
In [ ]: plt.style.use('ggplot')

        # colors = ['#9C824A','#DB0007', '#023474']

        Teams = df['Opponent'].tolist()
        Goals_scored = df['Goals for'].tolist()
        Goals_conceded = df['Goals against'].tolist()

        Team_goals_scored = {}
        Team_goals_conceded = {}

        for team, goal in zip(Teams, Goals_scored):
            if team not in Team_goals_scored:
                Team_goals_scored[team] = []
            Team_goals_scored[team].append(goal)


        for team, goal in zip(Teams, Goals_conceded):
            if team not in Team_goals_conceded:
                Team_goals_conceded[team] = []
            Team_goals_conceded[team].append(goal)

        Total_goals_scored = [sum(Team_goals_scored_list) for Team_goals_scored_list in Team_goals_score
        Total_goals_conceded = [sum(Team_goals_conceded_list) for Team_goals_conceded_list in Team_goals_

        x_indexes=np.arange(1,6)
        width=0.3

        plt.bar(x_indexes, Total_goals_scored, label='Goals scored', color='#023474',width=width)
        plt.bar(x_indexes-width, Total_goals_conceded, label='Goals conceded', color='#DB0007',width=wid


        plt.xlabel('Teams',fontdict={'fontname': 'Century Gothic', 'size': 15, 'color': '#9C824A'})
        plt.xticks(ticks=x_indexes,labels=Team_goals_conceded.keys())
        plt.ylabel('Goals',fontdict={'fontname': 'Century Gothic', 'size': 15, 'color': '#9C824A'})
        plt.title('Distribution of goals scored and conceded based on the opponent',
                  fontdict={'fontname': 'Century Gothic', 'size': 15, 'color': '#9C824A'})
        plt.legend()
        plt.tight_layout()
        plt.show()
```



```
In [ ]: df[Home_matches]['Result'].value_counts()
```

```
Out[ ]: W    4
        L    1
        D    0
        Name: Result, dtype: int64
```

Arsenal had a solid and an almost perfect winning record against the top six in the Emirates, where they picked up 4 wins and had only one loss which was against Manchester City

```
In [ ]: df[Away_matches]['Result'].value_counts()
```

```
Out[ ]: L    2
        W    2
        D    1
        Name: Result, dtype: int64
```

In away terms, Arsenal picked up two wins, one in Stanford Bridge against Chelsea and the other against Tottenham. The two losses came from the two Manchester teams, City and United, while the draw was picked up in Anfield against Liverpool

```
In [ ]: # Calculationg some percentages

Total_goals_for= 88
Total_goals_against=43

Goal_for_percentage= (df['Goals for'].sum()/ Total_goals_for)* 100
```

```
Goal_against_percentage= (df['Goals against'].sum()/ Total_goals_against)* 100

print(f'The percentage of goals scored vs the top six is: {Goal_for_percentage}')
print(f'The percentage of goals conceded vs the top six is: {Goal_against_percentage}')
```

```
The percentage of goals scored vs the top six is: 22.727272727272727
The percentage of goals conceded vs the top six is: 41.86046511627907
```

- Out of the 88 goals Arsenal scored in the season, 22% of them were scored against the top teams in the Premier League.
- Out of the 43 goals Arsenal conceded in the season, 41% of them were conceded against the top six, suggesting that the 10 matches against the top six culminated in almost half of the total goals conceded in the season

During my previous analysis on the season as a whole. Arsenal averaged a possession percentage of 59%. Compared to the 53% we extracted earlier, we can conclude to the fact that Arsenal had less of a dominance in the matches against the top six when put against the total average.

In [ ]:
```
#Comparing the averages of the goals scored and conceded between the top six and the entire seas

Total_for_avg=2.31
Total_against_avg= 1.13

df['Goals for'].mean(),df['Goals against'].mean()
```

Out[ ]:
```
(2.0, 1.8)
```

- The average of goals scored during the entire season is 2.31 and for the top teams specifically it's 2 on the dot, meaning that offensively, Arsenal performed just as well against the top six when comparated to the season as a whole.
- The case for the conceded goals is different as against the top six teams, Arsenal had an average of 1.8 goals scored against them which is significantly higher than the overall average which stands at 1.13.

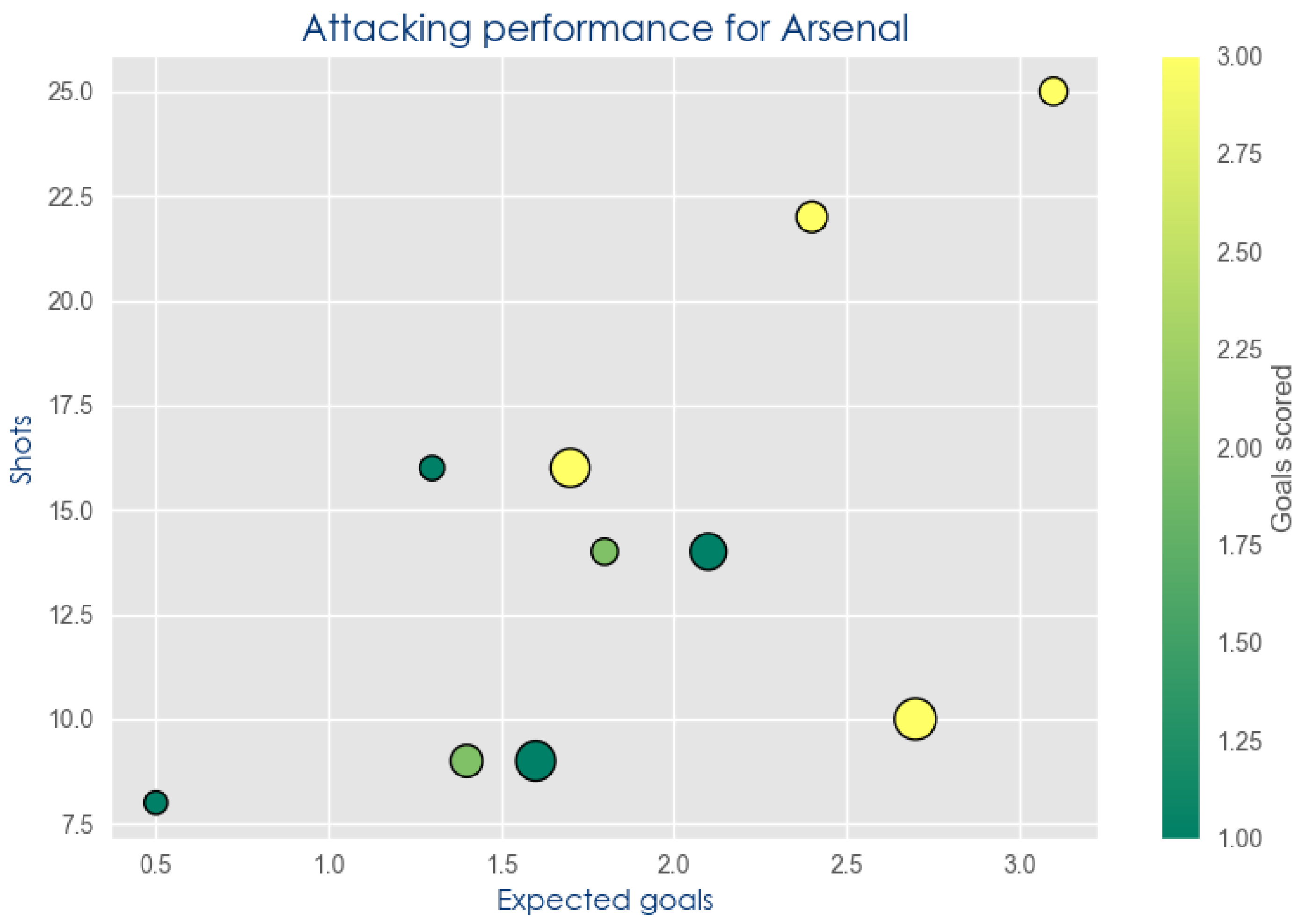
In [ ]:
```
#Calculating the shot accuracy

Total_shots=df['Shots'].sum()
Total_shots_on_target=df['Shots on target'].sum()

Shot_accuracy= (Total_shots_on_target / Total_shots) * 100

Shot_accuracy
```

Out[ ]:
```
32.16783216783217
```

Arsenal had a shot accuracy of 32%. They had culminated 143 shots against the top six teams and out of those total shots, 46 of them were on target. We can conclude to the fact that Arsenal's offensive outputs were of great caliber.

In [ ]:
```
# Scatter plot for goals scored and expected goals

Expected_goals=df['Expectec goals'].tolist()
Shots=df['Shots'].tolist()

Sizes=np.random.randint(80,300,10)
```

```
plt.scatter(Expected_goals,Shots,edgecolors='black',linewidth=1,s=Sizes,c=Goals_scored,cmap='summ

cbar=plt.colorbar()
cbar.set_label('Goals scored')


plt.title('Attacking performance for Arsenal',fontdict={'fontname':'Century Gothic','size':15,'co
plt.xlabel('Expected goals',fontdict={'fontname':'Century Gothic','size':12,'color':'#023474'})
plt.ylabel('Shots',fontdict={'fontname':'Century Gothic','size':12,'color':'#023474'})

plt.tight_layout()
plt.show()
```



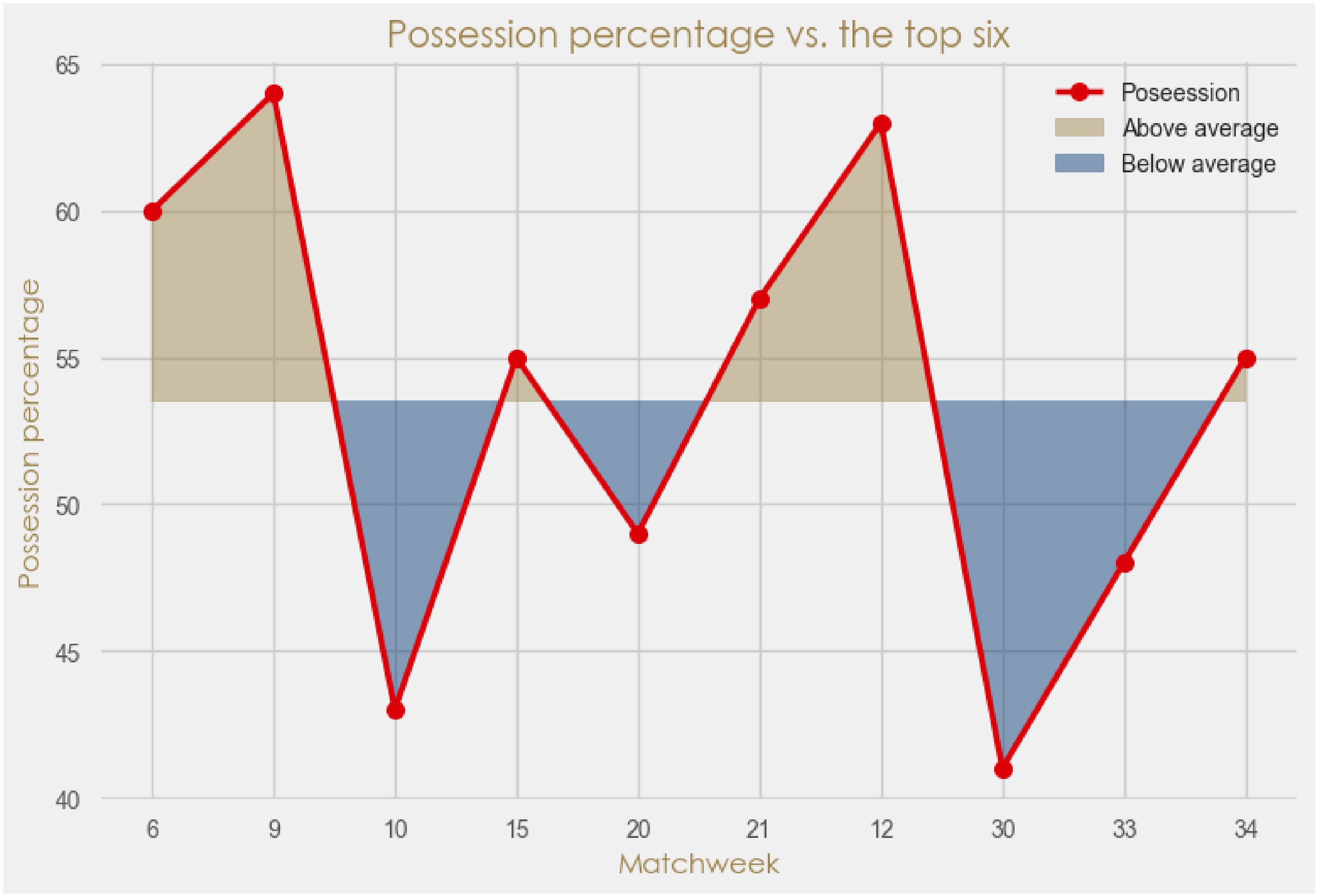
In [ ]:
```
#Calculating the passing accuracy

Total_passes=df['Passes'].sum()
Total_completed_passes=df['Passes completed'].sum()

Pass_accuracy= (Total_completed_passes / Total_passes) * 100

Pass_accuracy
```

Out[ ]:
```
81.4176620325044
```

Arsenal had a passing accuracy of 81%. This reflects the quality they had in midfield from players such as Partey, Xhaka and Ødegaard. This stat also ties into the possession average where they had a slight advantage in dominating the ball and controlling the ball.

In [ ]:
```
#Line graph representing the possession
plt.style.use('fivethirtyeight')
```

```
Possession=df['Possession'].tolist()
Matchweeks=df['Matchweek'].tolist()
Ticks=[]
Colors = ['#DB0007','#9C824A', '#023474']

Average_poss=df['Possession'].mean()

for x in range(len(Matchweeks)):
    match=Matchweeks[x][10:12]
    Ticks.append(match)

plt.plot(Matchweeks,Possession,label='Poseession',color=Colors[0],marker='o',linewidth=2.5
         ,markeredgecolor='green',markersize=8)

plt.fill_between(Matchweeks,df['Possession'],Average_poss,alpha=0.45,where=(Possession>Average_po
                 interpolate=True,color='#9C824A',label='Above average')
plt.fill_between(Matchweeks,df['Possession'],Average_poss,alpha=0.45,where=(Possession<Average_po
                 interpolate=True,color='#023474',label='Below average')

plt.title('Possession percentage vs. the top six',fontdict={'fontname':'Century Gothic','size':1
plt.ylabel('Possession percentage',fontdict={'fontname':'Century Gothic','size':12,'color':'#9C8
plt.xlabel('Matchweek',fontdict={'fontname':'Century Gothic','size':12,'color':'#9C824A'})

plt.xticks(ticks=Matchweeks,labels=Ticks)

plt.legend()

plt.tight_layout()
plt.show()
```



In [ ]:
```
#Calculating the percentage of attendances
```

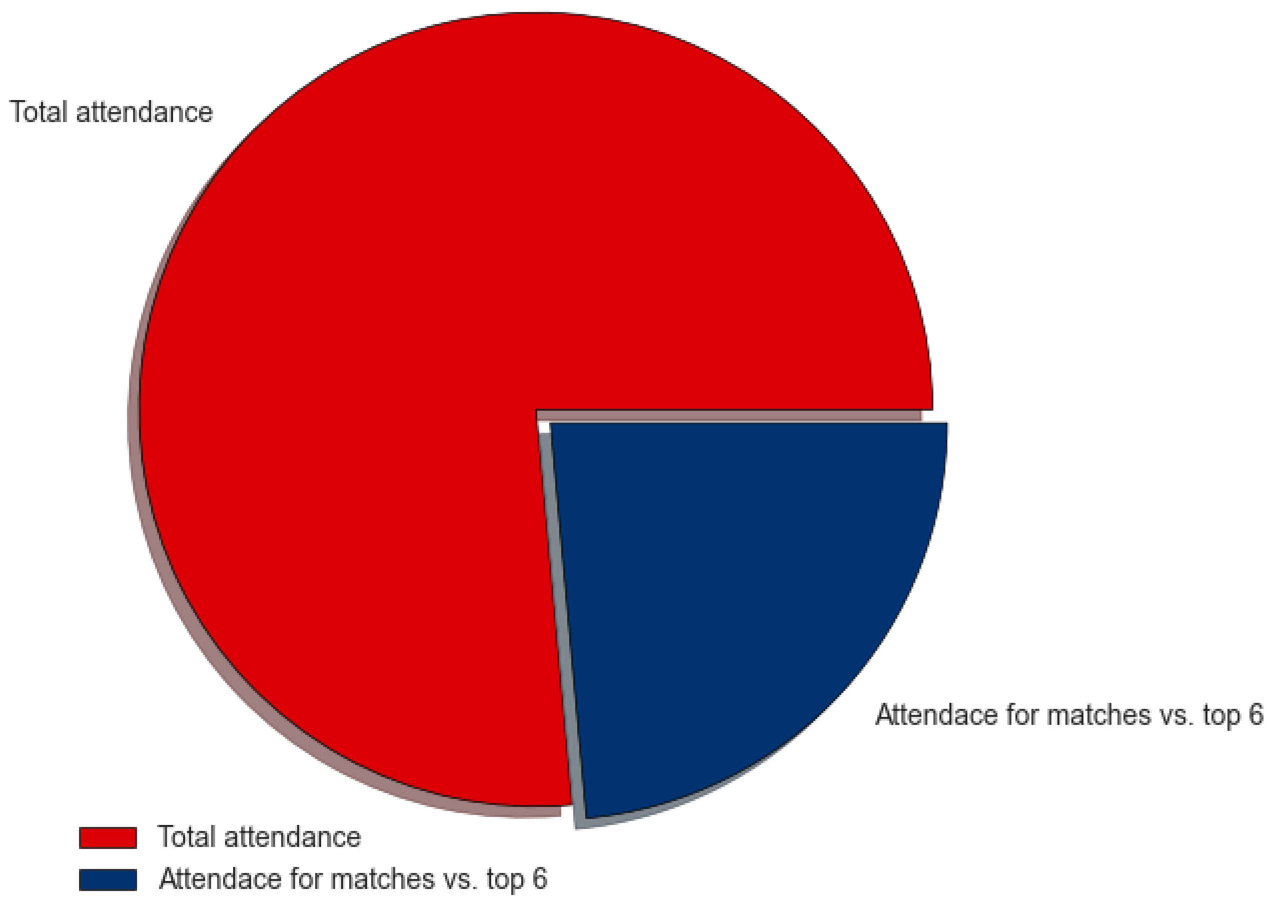
```
Total_attendance= 1893060
Top_6_attendance= df['Attendance'].sum()

Attendance_percentage= (Top_6_attendance / Total_attendance) * 100

Attendance_percentage
```

```
Out[ ]: 30.81117344405354
```

Arsenal had 10 matches against the top six and they represented about 30% of the total attendance percentage. The fact that the top six matches represented around 30% of the total attendance percentage emphasizes the impact of these matches on fan engagement. This statistic highlights that despite making up only a fraction of the total matches in the league season, the top six matches played a substantial role in drawing fans to the stadiums.

```
In [ ]: #Attendance percentage
plt.style.use('ggplot')

Whole=[Total_attendance,Top_6_attendance]

Colors = ['#DB0007','#023474']
Labels= ['Total attendance','Attendace for matches vs. top 6']

plt.pie(Whole,labels=Labels,colors=Colors,wedgeprops={'edgecolor':'black'},shadow=True,
        explode=[0.05,0])

plt.legend()
plt.title('Distribution of attendance numbers',fontdict={'fontname':'Century Gothic','size':15,'

plt.tight_layout()
plt.show()
```

Distribution of attendance numbers


Total attendance
Attendace for matches vs. top 6
🟥 Total attendance
🟦 Attendace for matches vs. top 6