

# DATA ANALYSIS REPORT: PERFORMANCE INSIGHTS FOR EUROPE’S TOP FIVE FOOTBALL LEAGUES, 2023/24 SEASON

**Prepared by:** Rodricks Misasa  
**Date:** February 21, 2025  
**Prepared for:** Football Analytics Stakeholders

## Contents

DATA ANALYSIS REPORT: PERFORMANCE INSIGHTS FOR EUROPE’S TOP FIVE FOOTBALL LEAGUES, 2023/24 SEASON..... 1

Executive Summary..... 2

1. Introduction ..... 2

    1.1 Background ..... 2

    1.2 Objectives..... 2

    1.3 Methodology..... 3

2. Data Cleaning and Preparation..... 3

    2.1 Process ..... 3

3. Exploratory Data Analysis (EDA)..... 4

    3.1 General League and Match Insights..... 4

    3.2 Team Performance ..... 11

    3.3 Match Outcome Analysis ..... 17

4. Findings..... 22

5. Conclusions ..... 23

    5.1 Key Insights..... 23

    5.2 Implications ..... 23

    5.3 Recommendations for Future Research ..... 23

6. Appendices ..... 23

    6.1 Data Sources ..... 23

    6.2 Glossary..... 23

    6.3 Limitations..... 23

## Executive Summary

This report presents a comprehensive analysis of Europe's top 5 football leagues—Bundesliga, La Liga, Ligue 1, Premier League, and Serie A—based on data from the 2023/24 season. Through exploratory data analysis (EDA), data cleaning, and advanced visualizations, we uncover key trends in match outcomes, team performance, and scoring patterns. Key findings include:

- The Premier League has the highest average goals per match at 3.28, indicating its high-scoring nature.
- Home teams consistently outperform away teams, with the Premier League showing the largest disparity in wins.
- Goal differences vary, with the Premier League exhibiting the widest distribution, suggesting greater variability in match outcomes.
- Top-performing teams like Inter (Serie A) and Real Madrid (La Liga) dominate in wins, while teams like Sheffield United (Premier League) face the most losses.
- Goals are scored more frequently in the second half across all leagues, with the Premier League showing the most significant increase.
- The most common full-time score is 1-1 (213 occurrences), and draws occur in 26.4% of matches, highlighting competitive balance.

These insights provide actionable recommendations for teams, coaches, and stakeholders to refine strategies, predict outcomes, and understand league dynamics.

## 1. Introduction

### 1.1 Background

Europe's top 5 football leagues are globally renowned for their competitive intensity, attracting millions of fans and substantial investments. The 2023/24 season offered a rich dataset to analyze match outcomes, team performance, and scoring trends, providing valuable insights for teams, broadcasters, and betting markets.

### 1.2 Objectives

The primary goal of this project was to perform an exploratory data analysis (EDA) and create insightful visualizations to communicate findings. Key objectives included:

- Cleaning and preparing datasets for consistency.
- Identifying trends in match outcomes, goal differences, and team performance.
- Comparing leagues on goals scored, wins, and referee influence.
- Highlighting key insights through visualizations using Matplotlib and Seaborn.

## 1.3 Methodology

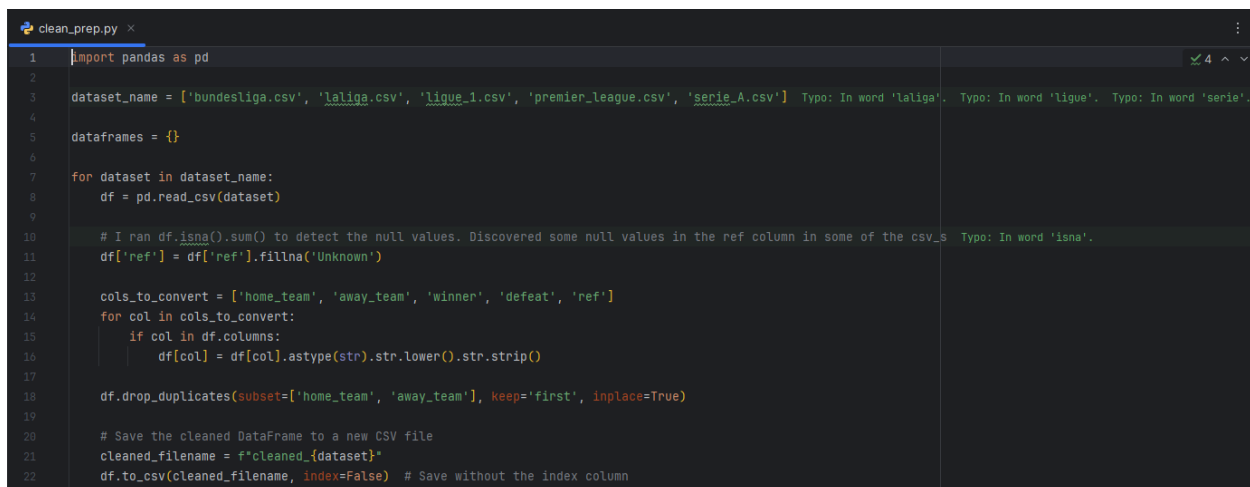
Data was sourced from CSV files for each league, cleaned for inconsistencies, and combined or analyzed separately as needed. Python libraries (pandas, matplotlib, seaborn) were used to process data and create visualizations. The analysis addressed specific questions about league performance, team metrics, and match outcomes.

## 2. Data Cleaning and Preparation

### 2.1 Process

To ensure data quality, we inspected and cleaned datasets for inconsistencies or missing values. The process involved:

- Loading datasets for each league: Bundesliga, La Liga, Ligue 1, Premier League, and Serie A.
- Identifying and handling null values, particularly in the referee ('ref') column.
- Standardizing text fields (e.g., team names, referees) by converting to lowercase and removing duplicates.



```
1 import pandas as pd
2
3 dataset_name = ['bundesliga.csv', 'laliga.csv', 'ligue_1.csv', 'premier_league.csv', 'serie_A.csv']
4
5 dataframes = {}
6
7 for dataset in dataset_name:
8     df = pd.read_csv(dataset)
9
10    # I ran df.isna().sum() to detect the null values. Discovered some null values in the ref column in some of the csv_s
11    df['ref'] = df['ref'].fillna('Unknown')
12
13    cols_to_convert = ['home_team', 'away_team', 'winner', 'defeat', 'ref']
14    for col in cols_to_convert:
15        if col in df.columns:
16            df[col] = df[col].astype(str).str.lower().str.strip()
17
18    df.drop_duplicates(subset=['home_team', 'away_team'], keep='first', inplace=True)
19
20    # Save the cleaned DataFrame to a new CSV file
21    cleaned_filename = f'cleaned_{dataset}'
22    df.to_csv(cleaned_filename, index=False) # Save without the index column
```

### Code Snippet Explanation:

This code snippet cleans and prepares datasets for Europe's top 5 football leagues (Bundesliga, La Liga, Ligue 1, Premier League, Serie A) for the 2023/24 season. It uses the pandas library to handle data manipulation, ensuring consistency and readiness for analysis:

- **Importing Libraries:** The pandas library is imported as pd to manage and manipulate data in tabular form (DataFrames).
- **Loading Datasets:** A list of CSV files (dataset\_name) representing each league's data is defined. The code loops through each file, reading it into a DataFrame using pd.read\_csv().
- **Handling Missing Values:** The df.isna().sum() method (mentioned in the comment) checks for null values, and df['ref'].fillna('Unknown') replaces any missing referee data with 'Unknown' to maintain data integrity.

- **Standardizing Text:** Columns like 'home\_team', 'away\_team', 'winner', 'defeat', and 'ref' are converted to lowercase and stripped of extra whitespace using `astype(str).str.lower().str.strip()` to ensure uniformity and avoid duplicates due to case or spacing issues.
- **Removing Duplicates:** `drop_duplicates()` removes duplicate matches based on 'home\_team' and 'away\_team', keeping the first occurrence to prevent redundancy.
- **Saving Cleaned Data:** Each cleaned DataFrame is saved as a new CSV file (e.g., `cleaned_bundesliga.csv`) using `to_csv()`, excluding the index column for cleaner output. A confirmation message is printed for tracking.

This process ensures the data is consistent, complete, and ready for comparative analysis, enabling accurate visualizations and insights in later sections.

**Outcome:** Each league's dataset was cleaned, saved as new CSVs, and prepared for comparative or individual analysis, ensuring accuracy for subsequent insights.

### 3. Exploratory Data Analysis (EDA)

#### 3.1 General League and Match Insights

##### 3.1.1 Which League Has the Highest Average Goals per Match?

We calculated the average goals per match for each league by summing home and away full-time scores and dividing by the number of matches.

```
highest_avg_goals.py x avg_goals_per_league.png
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 csv_data = ['cleaned_bundesliga.csv', 'cleaned_laliga.csv', 'cleaned_ligue_1.csv', 'cleaned_premier_league.csv', 'cleaned_serie_A.csv']
6 highest_avg_goals = []
7
8 for csv in csv_data:
9     df = pd.read_csv(csv)
10
11     goals_per_league = (df['home_score_full_time'].sum() + df['away_score_full_time'].sum()) / df.shape[0]
12
13     league = csv.replace(_old='cleaned_', _new='').replace(_old='.csv', _new='').title().replace(_old='_', _new=' ')
14
15     highest_avg_goals.append([league, round(float(goals_per_league), 2)])
16
17 dataF = pd.DataFrame(highest_avg_goals, columns=['league', 'avg_goals_per_league'])
18 plt.figure(figsize=(10, 10)) # Increase the figure size (width, height)
19 sns.barplot(x='league', y='avg_goals_per_league', data=dataF, palette='viridis', hue='league', legend=False)
20
21 # Adding the values on top of each bar
22 for index, row in dataF.iterrows():
23     plt.text(index, row['avg_goals_per_league'] + 0.02, # Adjust the value placement Expected type 'float', got 'Hashable' instead.
24             round(row['avg_goals_per_league'], 2), ha='center', va='bottom', fontsize=10)
25
26 plt.title(label='Average Goals per League', fontsize=14, fontweight='bold', color='darkblue', pad = 20)
27 plt.xlabel(xlabel='League', fontsize=14, fontweight='bold')
28 plt.gcf().set_facecolor('lightgrey')
29 plt.xticks(rotation=45, fontsize=10)
30 plt.ylabel(ylabel='Average Goals', fontsize=14, fontweight='bold')
31 plt.tight_layout()
32 plt.savefig('args: avg_goals_per_league.png', dpi = 300)
33 plt.show()
34
```

### Code Snippet Explanation:

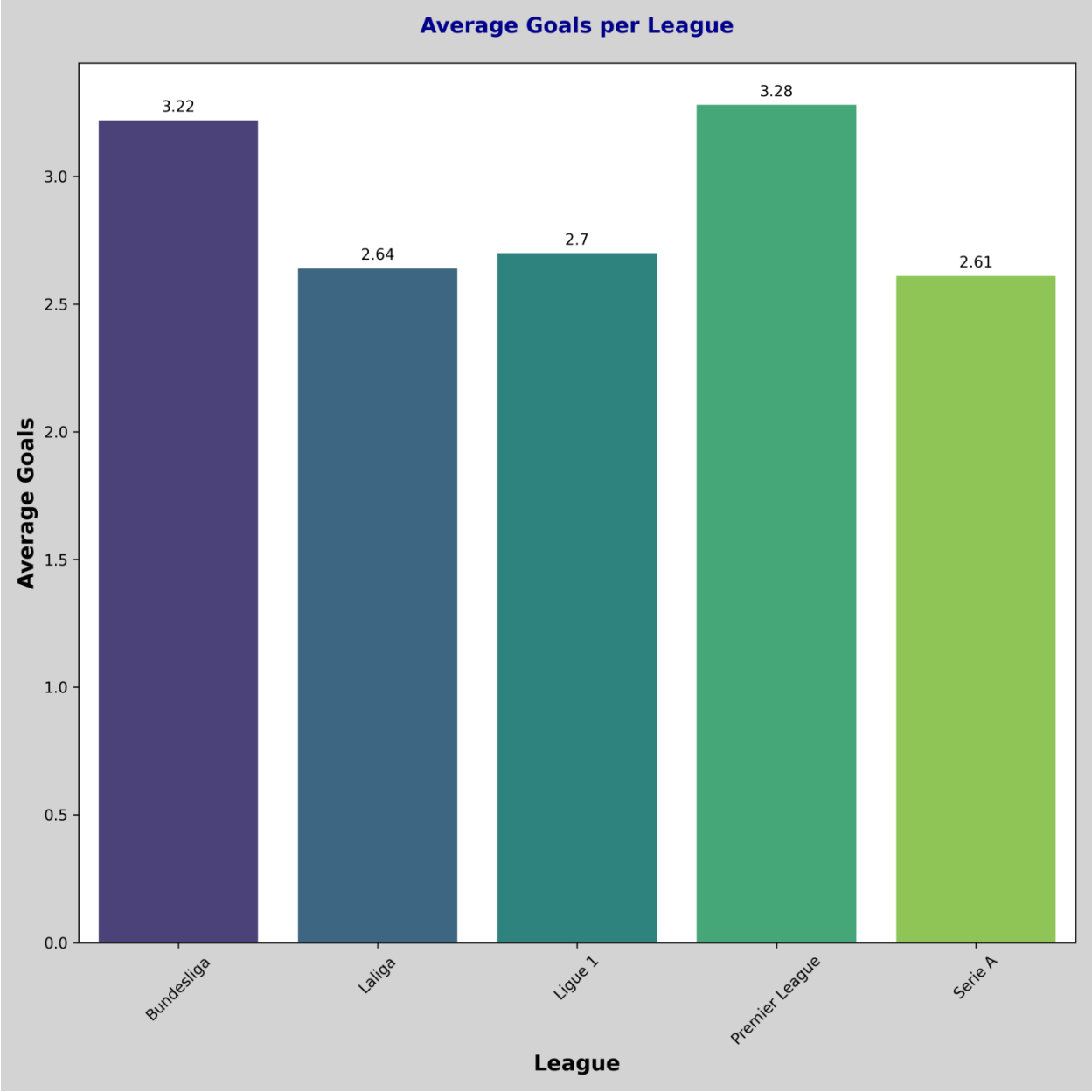
This code calculates and visualizes the average goals per match for each league, using pandas, matplotlib, and seaborn to create a barplot:

- **Importing Libraries:** matplotlib.pyplot (as plt), pandas (as pd), and seaborn (as sns) are imported for data manipulation, plotting, and enhanced visualization styling.
- **Loading Data:** A list of cleaned CSV files for each league is defined, and each file is read into a DataFrame using `pd.read_csv()`.
- **Calculating Averages:** For each league, the code sums the 'home\_score\_full\_time' and 'away\_score\_full\_time' columns, then divides by the number of matches (`df.shape[0]`) to get the average goals per match. The result is rounded to two decimal places for clarity.
- **Storing Results:** The league name (formatted by replacing file names and adding title case) and average are stored in a list, converted into a DataFrame for plotting.
- **Creating Visualization:** A barplot is generated using `sns.barplot()`, with 'league' on the x-axis and 'avg\_goals\_per\_league' on the y-axis. The `palette='viridis'` adds a color scheme, and `hue='league'` (with `legend=False`) ensures each bar is distinct. Values are added on top of bars using `plt.text()` for readability.
- **Customizing Plot:** The title, axis labels, background color (lightgrey), and rotated x-axis labels enhance clarity. The plot is saved as `avg_goals_per_league.png` at high resolution (`dpi=300`) and displayed.

This code produces a clear bar chart showing the Premier League's leadership in goals per match, aiding stakeholders in understanding league scoring dynamics.

### Average Goals per League

**Findings:** The Premier League leads with 3.28 goals per match, followed by Bundesliga at 3.22, Ligue 1 at 2.70, La Liga at 2.64, and Serie A at 2.61. This highlights the Premier League's high-scoring nature, potentially due to its competitive intensity.



### 3.1.2 How Do Home and Away Teams Perform Across the Different Leagues?

We compared home and away team wins to assess performance differences.

```
home_vs_away_wins.png home_vs_away.py x
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 csv_data = ['cleaned_bundesliga.csv', 'cleaned_laliga.csv', 'cleaned_ligue_1.csv', 'cleaned_premier_league.csv', 'cleaned_serie_A.csv'] Typo: In word 'laliga'. Typo:
6 h_v_a = []
7 # Compare perf of home vs away in terms of wins
8 for team in csv_data:
9     df = pd.read_csv(team)
10    league_name = team.replace(_old: 'cleaned_', _new: '').replace(_old: '.csv', _new: '').replace(_old: '_', _new: ' ').title()
11    home_team_wins = df.loc[df['home_team'] == df['winner']].shape[0]
12    away_team_wins = df.loc[df['away_team'] == df['winner']].shape[0]
13
14    h_v_a.append([league_name, home_team_wins, away_team_wins])
15
16 new_df = pd.DataFrame(h_v_a, columns=['league_name', 'home_team_wins', 'away_team_wins'])
17
18 new_df_melted = new_df.melt(id_vars=['league_name'], value_vars=['home_team_wins', 'away_team_wins'], var_name='Win Type', value_name='Wins')
19 new_df_melted['Win Type'] = new_df_melted['Win Type'].replace({'
20     'home_team_wins': 'Home Team Wins',
21     'away_team_wins': 'Away Team Wins'
22 })
23 print(new_df_melted)
24
25 plt.figure(figsize=(12, 8))
26 sns.barplot(x='league_name', y='Wins', data=new_df_melted, hue='Win Type')
27 # Customize plot appearance
28 plt.title(label='Home vs Away Wins in Each League', fontsize=16, fontweight='bold')
29 plt.gcf().set_facecolor('lightgrey') Typo: In word 'lightgrey'.
30 plt.xlabel(xlabel='League', fontsize=14)
31 plt.ylabel(ylabel='Number of Wins', fontsize=14)
32 plt.tight_layout()
33 plt.savefig(*args: 'home_vs_away_wins.png', dpi=300)
34 plt.show()
35
36
```

#### Code Snippet Explanation:

This code compares home and away team wins across leagues, using pandas, matplotlib, and seaborn to create a grouped barplot:

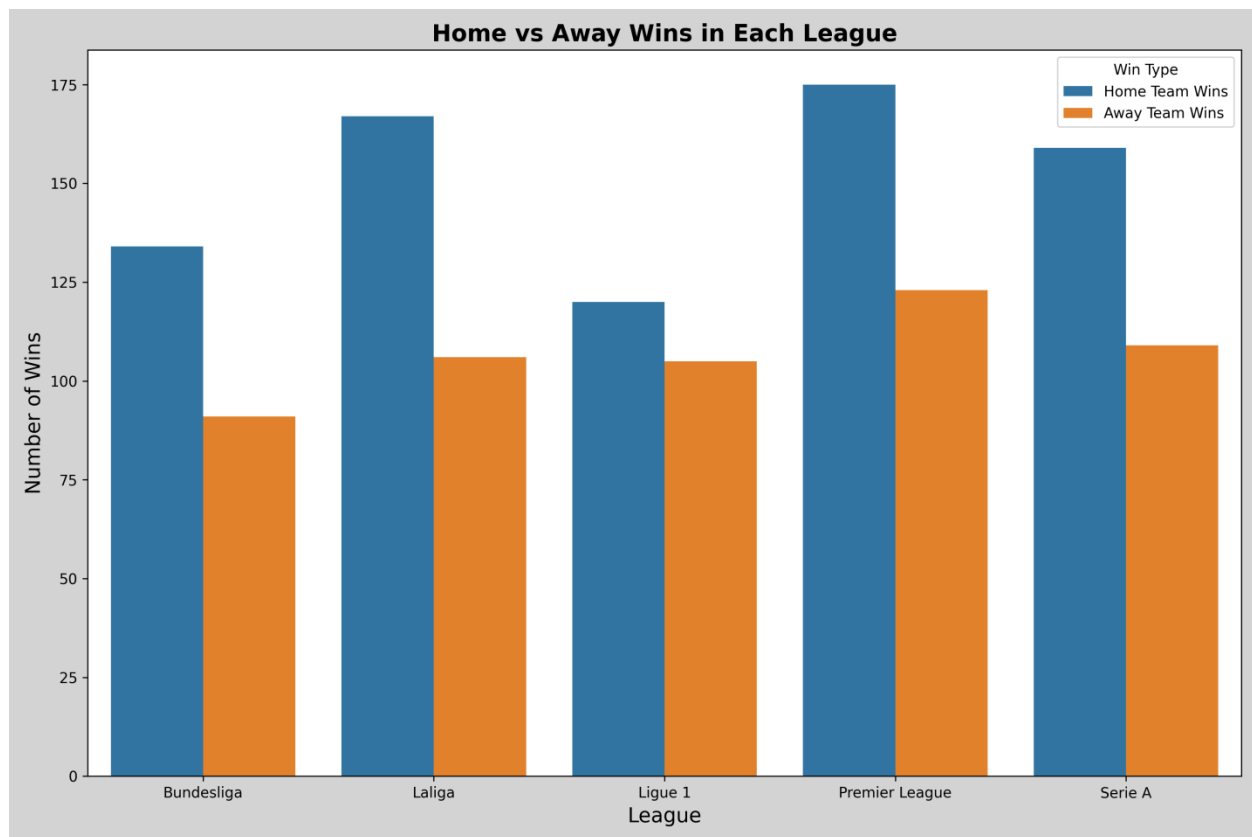
- **Importing Libraries:** The same libraries (plt, pd, sns) are imported for data handling and visualization.
- **Loading Data:** Cleaned CSV files for each league are listed and read into DataFrames.
- **Counting Wins:** For each league, the code counts home team wins (where home\_team matches winner) and away team wins (where away\_team matches winner) using loc and shape[0].
- **Storing Results:** Results are stored in a list with league names (formatted for readability) and win counts, then converted into a DataFrame.
- **Melting Data:** The DataFrame is reshaped using melt() to combine 'home\_team\_wins' and 'away\_team\_wins' into a single 'Win Type' column, renaming them for clarity (e.g., 'Home Team Wins', 'Away Team Wins').
- **Creating Visualization:** A barplot is generated with sns.barplot(), plotting 'league\_name' on the x-axis, 'Wins' on the y-axis, and 'Win Type' as hues to distinguish home vs. away wins. The palette='Set2' adds color variation.

- **Customizing Plot:** The title, axis labels, background color (lightgrey), and layout adjustments enhance readability. The plot is saved as home\_vs\_away\_wins.png at high resolution and displayed.

This code visualizes the home advantage across leagues, showing the Premier League's significant disparity, which is useful for tactical planning.

#### Home vs Away Wins in Each League

**Findings:** Home teams consistently outperform away teams, with the Premier League showing the largest gap (160 home wins vs. 125 away wins). This suggests a strong home advantage, possibly due to fan support or familiarity with home conditions.





### 3.1.3 What Is the Distribution of Goal Differences for Each League?

We examined goal differences to understand match competitiveness.

```
gd_dist.py x GD_Distribution.png
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 csv_data = ['cleaned_bundesliga.csv', 'cleaned_laliga.csv', 'cleaned_ligue_1.csv', 'cleaned_premier_league.csv', 'cleaned_serie_A.csv']
6 dfs = []
7 for csv in csv_data:
8     df = pd.read_csv(csv)
9
10    df['league'] = csv.replace(_old='cleaned_', _new='').replace(_old='_', _new=' ').replace(_old='.csv', _new='')
11    dfs.append(df)
12
13    combined_df = pd.concat(dfs, ignore_index=True)
14
15    # combined_df.to_csv('combined_table.csv', index=False)
16    plt.figure(figsize=(12, 8))
17
18    sns.boxplot(x='league', y='goal_diff', data=combined_df, palette='Set2', hue='league')
19    plt.title(label='Goal Difference Distribution by League', fontsize=16, fontweight='bold')
20    plt.gcf().set_facecolor('lightgray')
21    plt.xlabel(xlabel='League', fontsize=12, fontweight='bold')
22    plt.ylabel(ylabel='Goal Difference', fontsize=12, fontweight='bold')
23    plt.xticks(rotation=45)
24    plt.grid(axis='y', linestyle='--', alpha=0.7)
25    plt.tight_layout()
26    plt.savefig('GD_Distribution.png', dpi=300)
27    plt.show()
28
```

#### Code Snippet Explanation:

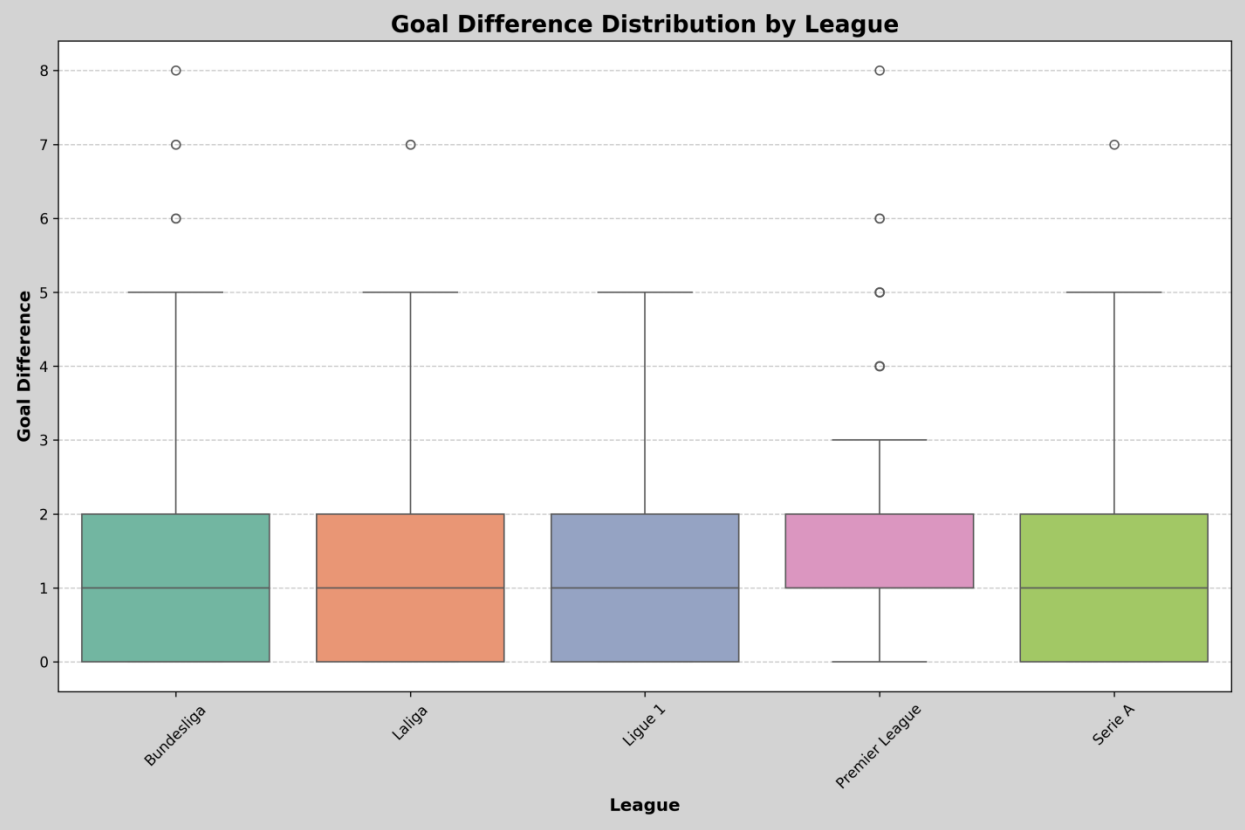
This code analyzes and visualizes the distribution of goal differences using a boxplot, combining data from all leagues:

- **Importing Libraries:** plt, pd, and sns are imported for data manipulation and plotting.
- **Loading Data:** Cleaned CSV files are listed, read into DataFrames, and each DataFrame is tagged with its league name for identification.
- **Combining Data:** All DataFrames are concatenated into a single combined\_df using pd.concat(), with ignore\_index=True to reset indices for consistency.
- **Creating Visualization:** A boxplot is generated using sns.boxplot(), with 'league' on the x-axis and 'goal\_diff' on the y-axis. The palette='Set2' adds color differentiation, and hue='league' ensures each league is distinguishable.
- **Customizing Plot:** The title, axis labels, background color (lightgray), rotated x-axis labels, and a grid (axis='y') improve readability. The plot is saved as GD\_Distribution.png at high resolution and displayed.

This code reveals variability in goal differences, with the Premier League showing the widest spread, aiding in understanding match competitiveness.

#### Goal Difference Distribution by League

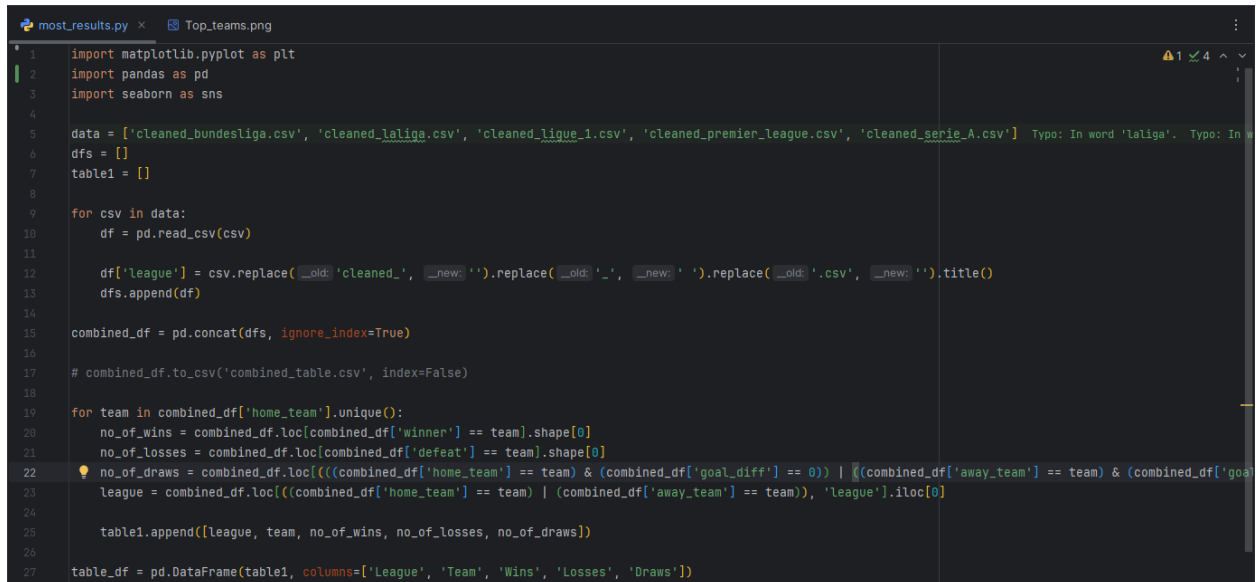
**Findings:** The Premier League shows the widest distribution (median 1.5, outliers up to 8), indicating greater variability. Other leagues like Bundesliga, La Liga, and Serie A have medians around 2, with outliers up to 7, suggesting more consistent outcomes.



## 3.2 Team Performance

### 3.2.1 Which Teams Have the Most Wins, Draws, and Losses?

We identified top-performing teams in wins, draws, and losses across all leagues.

A screenshot of a Jupyter Notebook interface. The top bar shows two tabs: 'most\_results.py' and 'Top\_teams.png'. The code is written in Python and is as follows:

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 data = ['cleaned_bundesliga.csv', 'cleaned_laliga.csv', 'cleaned_ligue1.csv', 'cleaned_premier_league.csv', 'cleaned_serie_A.csv']
6 dfs = []
7 table1 = []
8
9 for csv in data:
10     df = pd.read_csv(csv)
11
12     df['league'] = csv.replace(_old='cleaned_', _new='').replace(_old='.', _new=' ').replace(_old='.csv', _new='').title()
13     dfs.append(df)
14
15 combined_df = pd.concat(dfs, ignore_index=True)
16
17 # combined_df.to_csv('combined_table.csv', index=False)
18
19 for team in combined_df['home_team'].unique():
20     no_of_wins = combined_df.loc[combined_df['winner'] == team].shape[0]
21     no_of_losses = combined_df.loc[combined_df['defeat'] == team].shape[0]
22     no_of_draws = combined_df.loc[((combined_df['home_team'] == team) & (combined_df['goal_diff'] == 0)) | ((combined_df['away_team'] == team) & (combined_df['goal_diff'] == 0))]
23     league = combined_df.loc[((combined_df['home_team'] == team) | (combined_df['away_team'] == team)), 'league'].iloc[0]
24
25     table1.append([league, team, no_of_wins, no_of_losses, no_of_draws])
26
27 table_df = pd.DataFrame(table1, columns=['League', 'Team', 'Wins', 'Losses', 'Draws'])
```

#### Code Snippet Explanation:

This code identifies and visualizes the top-performing teams in wins, draws, and losses, using multiple barplots:

- **Importing Libraries:** `pd`, `np`, `plt`, and `sns` are imported for data handling, numerical operations, and visualization.
- **Loading Data:** Cleaned CSV files are listed, read, tagged with league names, and concatenated into `combined_df`.
- **Counting Metrics:** For each unique 'home\_team', the code counts wins (where team matches 'winner'), losses (where team matches 'defeat'), and draws (where 'goal\_diff' is 0 for either home or away team). Results are stored with league information.
- **Storing Results:** Data is organized into a DataFrame with columns for 'League', 'Team', 'Wins', 'Losses', and 'Draws'.
- **Creating Visualizations:** Three subplots are created using `plt.subplot()` for wins, draws, and losses:
  - Each uses `sns.barplot()` with 'Team' on the x-axis, the metric (e.g., 'Wins') on the y-axis, and 'League' as hues for distinction.
  - The top 10 teams are sorted by each metric and plotted, with team names formatted to title case and rotated for readability.

- **Customizing Plots:** Titles, axis labels, background color (lightblue), and font styles (Times New Roman) are customized. The layout is adjusted with `tight_layout()`, and the plot is saved as `Top_teams.png` at high resolution.

This code provides a comprehensive view of team performance, identifying leaders like Inter and underperformers like Sheffield United.

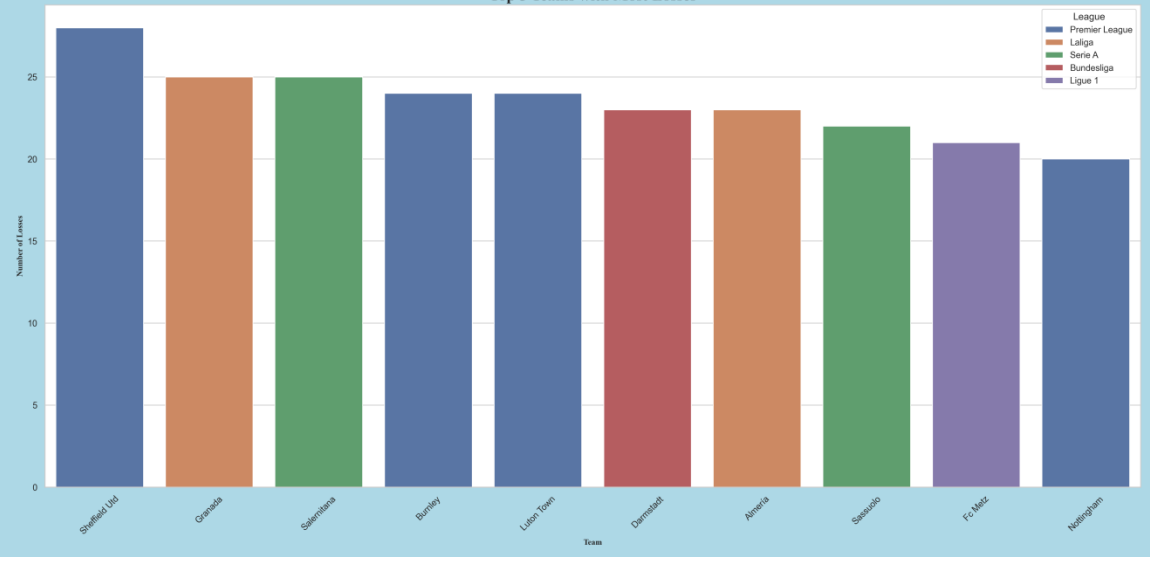
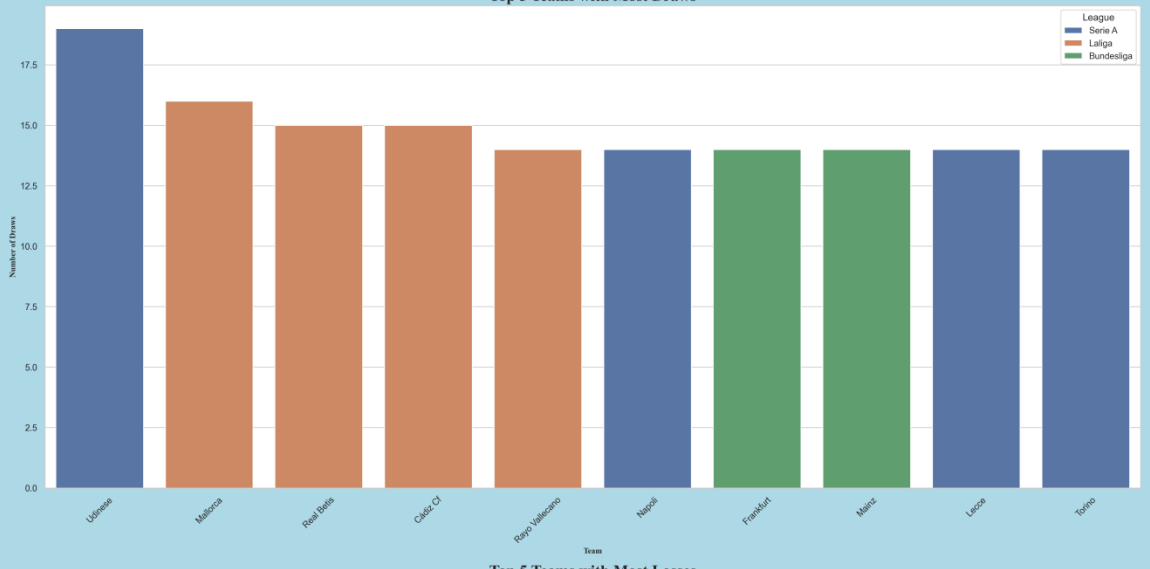
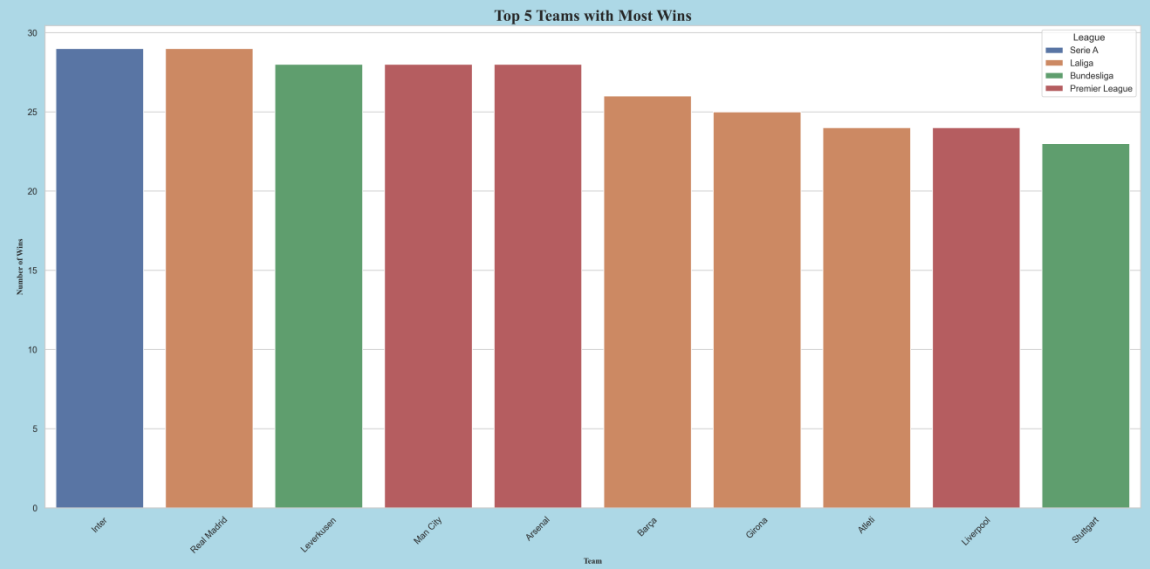
```

most_results.py x Top_teams.png
28
29 # Set plot style
30 sns.set(style='whitegrid') Function 'set' is deprecated in favor of 'set_theme'. Type: In word 'whitegrid'.
31 plt.figure(figsize=(20, 30))
32 plt.gcf().set_facecolor('lightblue')
33
34 # Top 10 teams with most wins
35 top_wins = table_df.sort_values(by='Wins', ascending=False).head(10)
36 plt.subplot(*args: 3, 1, 1)
37 sns.barplot(x='Team', y='Wins', hue='League', data=top_wins, dodge=False)
38 plt.title(label='Top 5 Teams with Most Wins', fontsize=20, fontweight='bold', fontname='Times New Roman')
39 plt.ylabel(ylabel='Number of Wins', fontsize=10, fontweight='bold', fontname='Times New Roman')
40 plt.xlabel(xlabel='Team', fontsize=10, fontweight='bold', fontname='Times New Roman')
41 plt.xticks(ticks=range(10), labels=[team.title() for team in top_wins['Team']], rotation=45) # Apply title() to team names
42
43 # Top 10 teams with most draws
44 top_draws = table_df.sort_values(by='Draws', ascending=False).head(10)
45 plt.subplot(*args: 3, 1, 2)
46 sns.barplot(x='Team', y='Draws', hue='League', data=top_draws, dodge=False)
47 plt.title(label='Top 5 Teams with Most Draws', fontsize=20, fontweight='bold', fontname='Times New Roman')
48 plt.ylabel(ylabel='Number of Draws', fontsize=10, fontweight='bold', fontname='Times New Roman')
49 plt.xlabel(xlabel='Team', fontsize=10, fontweight='bold', fontname='Times New Roman')
50 plt.xticks(ticks=range(10), labels=[team.title() for team in top_draws['Team']], rotation=45) # Apply title() to team names
51
52 # Top 10 teams with most losses
53 top_losses = table_df.sort_values(by='Losses', ascending=False).head(10)
54 plt.subplot(*args: 3, 1, 3)
55 sns.barplot(x='Team', y='Losses', hue='League', data=top_losses, dodge=False)
56 plt.title(label='Top 5 Teams with Most Losses', fontsize=20, fontweight='bold', fontname='Times New Roman')
57 plt.ylabel(ylabel='Number of Losses', fontsize=10, fontweight='bold', fontname='Times New Roman')
58 plt.xlabel(xlabel='Team', fontsize=10, fontweight='bold', fontname='Times New Roman')
59 plt.xticks(ticks=range(10), labels=[team.title() for team in top_losses['Team']], rotation=45) # Apply title() to team names
60
61 plt.tight_layout()
62 plt.savefig(*args: 'Top_teams.png', dpi=300)
63 plt.show()
64

```

## Findings:

- **Wins:** Inter (Serie A) leads with 28 wins, followed by Real Madrid (La Liga) with 27.
- **Draws:** Udinese (Serie A) tops with 18 draws, reflecting a pattern of stalemates.
- **Losses:** Sheffield United (Premier League) has the most losses at 27, indicating performance challenges.



### 3.2.2 What Is the Goal-Scoring Trend Over the Matchdays for Each Team?

We visualized how teams' goal-scoring patterns evolved over the season.

```
goal_scoring_trend.py x goal_scoring_trend.png
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 df = pd.read_csv('combined_table.csv')
6
7 goal_trend = []
8
9 # Standardize team names
10 df['home_team'] = df['home_team'].str.replace('1.', '').str.title()
11 df['away_team'] = df['away_team'].str.replace('1.', '').str.title()
12
13 for team in df['home_team'].unique():
14     match_days = df.loc[(df['home_team'] == team) | (df['away_team'] == team)].sort_values(by='matchday')
15     for _, row in match_days.iterrows():
16         if row['home_team'] == team:
17             goal_trend.append([team.title(), row['matchday'], row['home_score_full_time']])
18         else:
19             goal_trend.append([team.title(), row['matchday'], row['away_score_full_time']])
20
21 goal_trend = pd.DataFrame(goal_trend, columns=['Team', 'Match Day', 'Goals'])
22 print(goal_trend.head(10))
23
24 # Pivot the data to create a matrix of teams vs matchdays
25 goal_trend_pivot = goal_trend.pivot_table(index='Team', columns='Match Day', values='Goals', aggfunc='sum', fill_value=0)
26
27 # Create the heatmap
28 plt.figure(figsize=(20, 40))
29 sns.heatmap(goal_trend_pivot, cmap="YlGnBu", cbar_kws={'label': 'Goals Scored'}, linewidths=0.5)
30
31 plt.title(label='Goal Scoring Trend Across Matchdays (Top 5 Leagues)', fontsize=10, fontweight='bold')
32 plt.xlabel(xlabel='Matchday', fontsize=14, fontweight='bold')
33 plt.ylabel(ylabel='Team', fontsize=14, fontweight='bold')
34 plt.xticks(rotation=90) # Rotate matchday labels for clarity
35 plt.gcf().set_facecolor('lightgrey')
36 plt.tight_layout()
37
38 plt.savefig("args: 'goal_scoring_trend.png', dpi=300)
39
40 plt.show()
41
```

#### Code Snippet Explanation:

This code visualizes goal-scoring trends over match days using a heat map, tracking each team's performance:

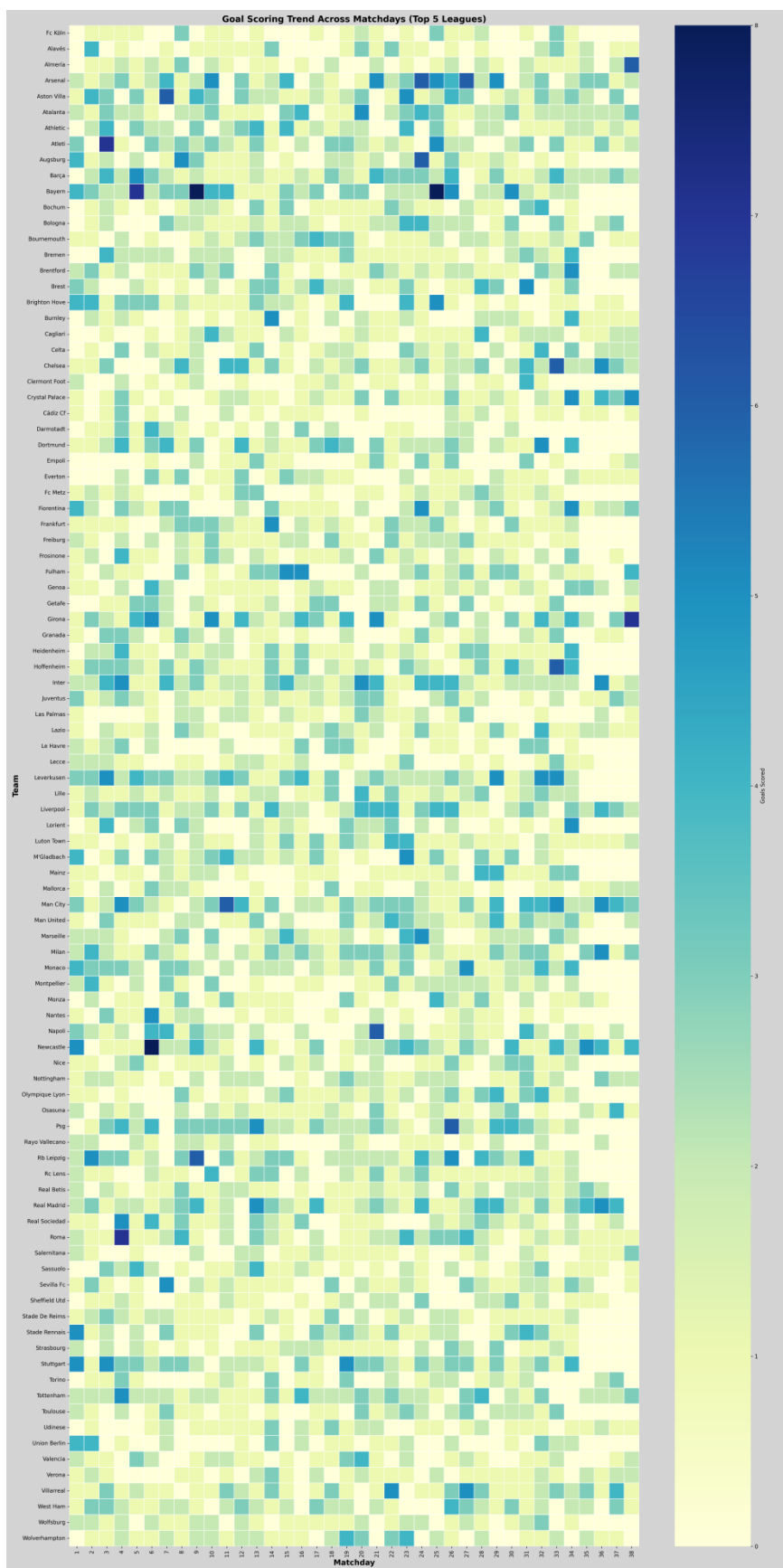
- **Importing Libraries:** plt, pd, and sns are imported for data handling and visualization.
- **Loading Data:** The combined\_table.csv file is read into a DataFrame.
- **Standardizing Team Names:** Team names in 'home\_team' and 'away\_team' columns are cleaned (removing '1.' and converting to title case) for consistency.
- **Tracking Goals:** For each unique team, matches are sorted by 'matchday'. Goals are tracked—home goals from 'home\_score\_full\_time' and away goals from 'away\_score\_full\_time'—and stored with team, matchday, and goal count.
- **Storing Results:** Data is organized into a DataFrame, then pivoted using pivot\_table() to create a matrix of teams vs. matchdays, with goals summed and zeros for missing data.

- **Creating Visualization:** A heat map is generated using `sns.heatmap()`, with 'Team' on the y-axis, 'Match Day' on the x-axis, and goal counts color-coded (YlGnBu cmap). The color bar labels goals scored.
- **Customizing Plot:** The title, axis labels, background color (lightgrey), and rotated x-axis labels improve readability. The plot is large (`figsize=(20, 40)`) to accommodate many teams and displayed (commented save function for reference).

This code shows seasonal scoring patterns, helping identify high-scoring matchdays for teams like Bayern Munich.

#### Goal Scoring Trend across Match days

**Findings:** Teams like Bayern Munich and Manchester City show high-scoring patches, while others exhibit consistency or variability, aiding in understanding seasonal performance trends.





### 3.3 Match Outcome Analysis

#### 3.3.1 How Does the Number of Goals Scored Vary Between the First and Second Halves?

We compared goal distribution across halves.

```
compare_1st_2nd_goals.py x
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 df = pd.read_csv('combined_table.csv')
6
7 df['1st Half Goals'] = df['home_score_half_time'] + df['away_score_half_time']
8 df['2nd Half Goals'] = df['home_score_full_time'] + df['away_score_full_time'] - df['1st Half Goals']
9
10 # Melt the dataframe to have '1st Half Goals' and '2nd Half Goals' in a single column
11 melted_pd = pd.melt(df, id_vars=['league'], value_vars=['1st Half Goals', '2nd Half Goals'],
12                    var_name='Half', value_name='Goals')
13
14 # Plotting the grouped barplot Type: In word 'barplot'.
15 plt.figure(figsize=(10, 6))
16 sns.barplot(x='league', y='Goals', hue='Half', data=melted_pd, palette='Set2')
17
18 # Customize plot
19 plt.title('Goals Scored in First and Second Halves by League', fontsize=16, fontweight='bold')
20 plt.xlabel('League', fontsize=12, fontweight='bold')
21 plt.ylabel('Goals Scored', fontsize=12, fontweight='bold')
22 plt.xticks(rotation=45)
23 plt.gcf().set_facecolor('lightblue')
24 plt.tight_layout()
25 plt.grid(axis='y', linestyle='--')
26 plt.savefig('1st_2nd_goals.png', dpi=300)
27
28 # Show plot
29 plt.show()
```

#### Code Snippet Explanation:

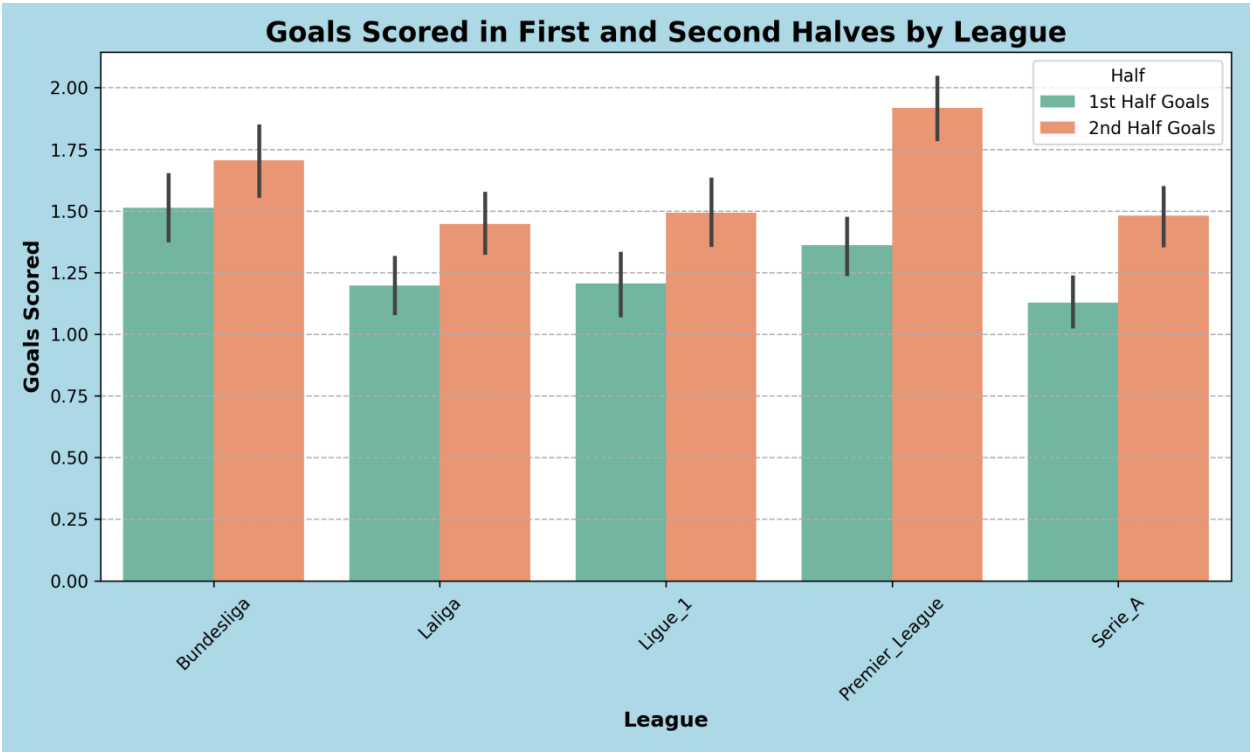
This code compares goals scored in the first and second halves, visualizing results with a grouped bar plot:

- **Importing Libraries:** plt, pd, and sns are imported for data handling and visualization.
- **Loading Data:** The combined\_table.csv file is read into a DataFrame.
- **Calculating Goals:** '1st Half Goals' are computed by summing 'home\_score\_half\_time' and 'away\_score\_half\_time'. '2nd Half Goals' are calculated as the difference between full-time and half-time totals.
- **Reshaping Data:** The DataFrame is melted using pd.melt() to combine '1st Half Goals' and '2nd Half Goals' into a single 'Goals' column, with 'Half' as a variable and 'league' as an identifier.
- **Creating Visualization:** A barplot is generated with sns.barplot(), plotting 'league' on the x-axis, 'Goals' on the y-axis, and 'Half' as hues to distinguish first vs. second halves. The palette='Set2' adds color variation.
- **Customizing Plot:** The title, axis labels, background color (lightblue), rotated x-axis labels, and a grid enhance readability. The plot is saved as 1st\_2nd\_goals.png at high resolution and displayed.

This code reveals the Premier League's significant second-half increase, informing tactical adjustments.

Goals Scored in First and Second Halves by League

**Findings:** Second halves generally see more goals, with the Premier League showing the largest increase (1.3 to 1.8), suggesting tactical shifts or fatigue impacting defenses.



### 3.3.2 What Are the Most Common Full-Time Scores?

We identified frequent match outcomes.

```
avg_ft_score.py x avg_ft_score.png
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 df = pd.read_csv('combined_table.csv')
6
7 score_counts = df.groupby(['home_score_full_time', 'away_score_full_time']).size().reset_index(name='count')
8
9
10 pivot_Table = score_counts.pivot(index='home_score_full_time', columns='away_score_full_time', values = 'count').fillna(0)
11
12 sns.heatmap(pivot_Table, cmap='coolwarm', annot=True, fmt='g', linewidths=.5, cbar_kws={'label': 'Frequency'}) Type: In word 'coolwarm'.
13 plt.title(label: 'Average Full Time Score Across Top 5 Leagues', fontsize=30, fontweight='bold', fontdict={'fontname': 'Times New Roman'})
14 plt.xlabel(xlabel: 'Away Team FT Score', fontsize=12, fontweight='bold', font = 'Times New Roman')
15 plt.ylabel(ylabel: 'Home Team FT Score', fontsize=12, fontweight='bold', font = 'Times New Roman')
16 plt.tight_layout()
17 plt.savefig('args: 'avg_ft_score.png', dpi = 300)
18 plt.show()
19
```

#### Code Snippet Explanation:

This code identifies and visualizes the most frequent full-time scores using a heat map:

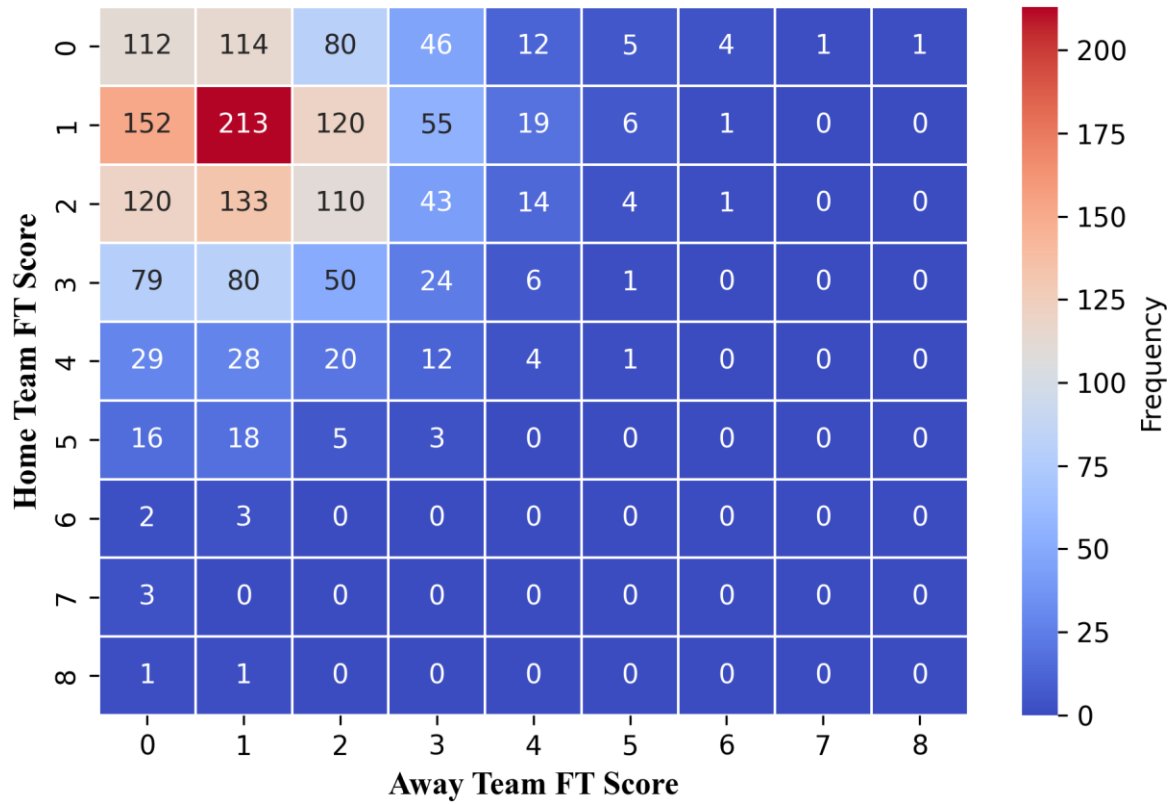
- **Importing Libraries:** plt, pd, and sns are imported for data handling and visualization.
- **Loading Data:** The combined\_table.csv file is read into a DataFrame.
- **Counting Scores:** Full-time scores are grouped by 'home\_score\_full\_time' and 'away\_score\_full\_time', with counts stored in score\_counts using groupby().size().
- **Pivoting Data:** The data is pivoted into a matrix with home scores as rows, away scores as columns, and frequencies as values, filling missing values with 0 using pivot().fillna(0).
- **Creating Visualization:** A heatmap is generated with sns.heatmap(), using 'coolwarm' for color coding, annotating cells with exact frequencies (annot=True), and formatting numbers (fmt='g'). The color bar shows frequency.
- **Customizing Plot:** The title, axis labels, and font (Times New Roman) are customized for professionalism. The plot is saved as avg\_ft\_score.png at high resolution and displayed.

This code highlights the 1-1 score's dominance, aiding in understanding typical match outcomes.

#### Average Full Time Score Across Top 5 Leagues

**Findings:** The most common score is 1-1 (213 occurrences), followed by 1-0 (152) and 0-0 (112), indicating a preference for close, competitive games.

# Full Time Scores



### 3.3.3 How Often Do Matches End in a Draw?

We quantified draw frequency.

```
draw_chances.py x draw_percentage.png
1 import matplotlib.pyplot as plt
2 import pandas as pd
3
4 df = pd.read_csv('combined_table.csv')
5
6 total_matches = df.shape[0]
7
8 draws = df[df['winner'] == 'draw'].shape[0]
9 non_Draws = total_matches - draws
10 draw_percentage = (draws / total_matches) * 100
11
12 print(f'Total Matches: {total_matches}')
13 print(f'Total Draws: {draws}')
14 print(f'Draw Percentage: {draw_percentage:.2f}%')
15
16 labels = ['Draws', 'Non-Draws']
17 sizes = [draws, non_Draws]
18 colors = ['gold', 'lightcoral'] Type: In word 'lightcoral'.
19
20 plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140, wedgeprops={'edgecolor': 'black'}) Type: In word 'edgecolor'.
21
22 centre_circle = plt.Circle(xy=(0,0), radius=0.70, fc='white')
23 fig = plt.gcf()
24 fig.gca().add_artist(centre_circle)
25
26 plt.title(label='Draw Percentage in Top 5 Leagues', fontsize=30, fontweight='bold', fontdict={'fontname': 'Times New Roman'})
27 plt.savefig(args='draw_percentage.png', dpi = 300)
28 plt.tight_layout()
29 plt.show()
```

#### Code Snippet Explanation:

This code calculates and visualizes the percentage of matches ending in a draw using a pie chart:

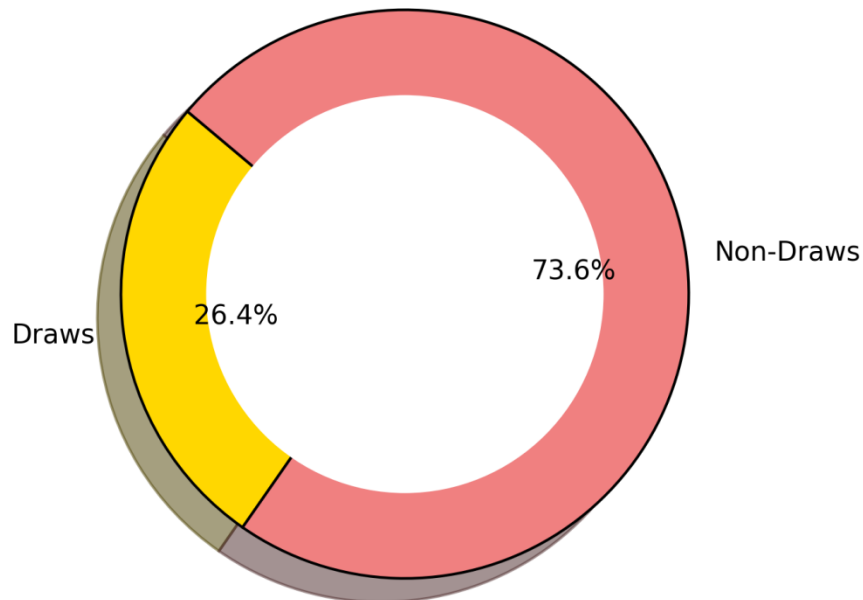
- **Importing Libraries:** plt and pd are imported for plotting and data handling.
- **Loading Data:** The combined\_table.csv file is read into a DataFrame.
- **Counting Draws:** Total matches are counted using df.shape[0]. Draws are identified where 'winner' is 'draw', and non-draws are calculated. The draw percentage is computed as (draws / total\_matches) \* 100.
- **Storing Results:** Results are printed for transparency, showing total matches, draws, and percentage.
- **Creating Visualization:** A pie chart is generated with plt.pie(), using 'Draws' and 'Non-Draws' as labels, colors (gold for draws, lightcoral for non-draws), and percentage labels (autopct='%1.1f%%'). A white center circle creates a donut effect, and black edges (wedgeprops) enhance clarity.
- **Customizing Plot:** The title, font (Times New Roman), and layout are adjusted. The plot is saved as draw\_percentage.png at high resolution and displayed.

This code quantifies the 26.4% draw rate, reflecting league competitiveness.

#### Draw Percentage in Top 5 Leagues

**Findings:** Draws occur in 26.4% of matches (500 out of 1896), a significant proportion reflecting competitive balance.

# Draw Percentage in Top 5 Leagues



## 4. Findings

- **League Competitiveness:** The Premier League's high average goals (3.28) and wide goal difference distribution suggest it is the most dynamic and unpredictable league, while Serie A and La Liga show more consistent outcomes.
- **Home Advantage:** Across all leagues, home teams win more frequently, with the Premier League showing the strongest disparity, potentially due to fan support or tactical familiarity.
- **Team Performance:** Dominant teams like Inter and Real Madrid excel in wins, while struggling teams like Sheffield United face significant losses, indicating performance disparities within leagues.
- **Scoring Patterns:** Goals increase in the second half, possibly due to tactical adjustments or fatigue, with the Premier League showing the most pronounced shift.
- **Match Outcomes:** The prevalence of 1-1 and 0-0 scores, alongside a 26.4% draw rate, highlights the competitive nature of these leagues, where tight, balanced games are common.

## 5. Conclusions

### 5.1 Key Insights

This analysis reveals the unique characteristics of each league:

- **Premier League:** Leads in goals per match and shows high variability in outcomes, making it highly entertaining but unpredictable.
- **Bundesliga, La Liga, Serie A:** Exhibit more consistent goal differences and outcomes, with strong home advantages and balanced competitions.
- **Ligue 1:** Shows the smallest gap in home vs. away wins, suggesting a more even playing field.

### 5.2 Implications

- **For Teams and Coaches:** Understanding home advantages, scoring trends, and common outcomes can guide tactical decisions, such as when to push for goals or focus on defense.
- **For Analysts and Bettors:** The high draw rate and frequent 1-1 scores provide insights for predicting match outcomes and setting odds.
- **For Broadcasters and Fans:** The dynamic nature of the Premier League and frequent draws across leagues enhance viewer engagement.

### 5.3 Recommendations for Future Research

- Investigate referee impact on outcomes by analyzing 'ref' data more deeply.
- Explore the influence of specific match days, weather, or player injuries on scoring trends.
- Conduct a longitudinal study comparing these trends with previous seasons to identify long-term shifts.

## 6. Appendices

### 6.1 Data Sources

- CSV files for each league, sourced from Kaggle.com, covering all matches in the 2023/24 season.

### 6.2 Glossary

- **Goal Difference:** The difference between goals scored and conceded in a match.
- **Match day:** A specific round or week in the season's schedule.
- **Full-Time Score:** The final score at the end of a match.

### 6.3 Limitations

- Data might exclude certain match details (e.g., injuries, weather) that could influence outcomes.
- Standardizing team names assumed consistency, but minor variations could exist.

This report provides a detailed, data-driven narrative of the 2023/24 season, offering actionable insights for stakeholders in the football industry.