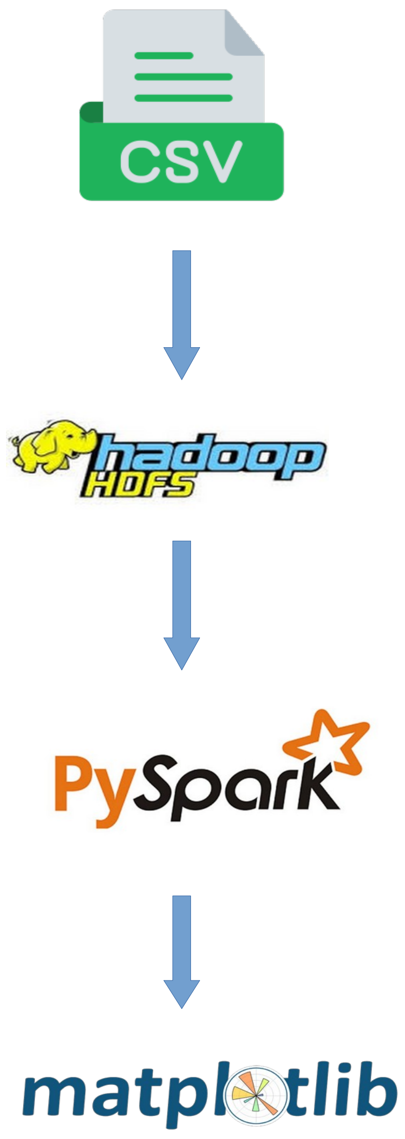
English Premier League Matches 2023/2024

Spark data analytics project

SPARK

- Spark commonly refers to Apache Spark, an open-source, distributed computing system that provides a fast and general-purpose cluster-computing framework for big data processing.
- Apache Spark is designed to be fast and flexible, with support for various programming languages such as Scala, Java, Python, and R.
- Spark supports various data processing tasks, including batch processing, interactive queries, streaming analytics, machine learning, and graph processing.

PROJECT WORK FLOW



CSV

- CSV stands for Comma-Separated Values. It is a simple and widely used file format for storing tabular data, such as a spreadsheet or a database.
- CSV files are commonly used for importing and exporting data between different software applications, databases, and spreadsheet programs.
- They are human-readable, easy to create and manipulate, and supported by many tools and programming languages for data processing.

HDFS

- HDFS stands for Hadoop Distributed File System. It is a distributed file system designed to store, manage and process large data datatsets.
- HDFS a robust and scalable solution for handling the storage needs of big data applications in a distributed and fault-tolerant manner.
- HDFS is designed to run on commodity hardware, making it a cost-effective solution for storing massive datasets.

PYSPARK

- PySpark is the Python API for Apache Spark, a powerful open-source distributed computing system.
- Apache Spark is designed for large-scale data processing and analytics, providing a fast and general-purpose cluster computing framework.
- PySpark allows developers to write Spark applications using Python programming language.
- PySpark includes Spark SQL, which allows developers to query structured data using SQL-like syntax. It supports the creation and manipulation of DataFrames, providing a more high-level, tabular abstraction for working with structured data.

MATPLOTLIB

- Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in data science, machine learning, and scientific computing.
- It provides a wide range of plotting options and customization features, making it a versatile tool for creating high-quality visualizations.

ABOUT DATASET

- The dataset is from kaggle, a well known online platform that provides users to share and discover datasets .Users can upload, explore, and download datasets for use in their own projects or analyses.

- https://www.kaggle.com/datasets/mertbayraktar/english-premier-league-matches-20232024-season

- The dataset contains 28 columns with the following column specifications:

Unnamed: 0: An index or identifier column.

Date: The date when the match took place.

Time: The kickoff time of the match.

Comp: The competition name, which is the Premier League for the rows displayed.

Round: The matchweek or round of the competition.

Day: The day of the week the match was played.

Venue: Indicates whether the team was playing at home or away.

Result: The outcome of the match from the perspective of the team mentioned at the end (W= Win, D = Draw, L = Loss).

GF (Goals For): The number of goals scored by the team.

GA (Goals Against): The number of goals conceded by the team.

Opponent: The name of the opposing team.

xG: Expected goals for the team.

xGA: Expected goals against the team.

Poss: Possession percentage during the match.

Attendance: The number of spectators present at the venue.

Captain: The name of the team captain.

Formation: The team's formation.

Referee: The name of the match referee.

Match Report: A link or reference to a detailed match report.

Notes: Any additional notes about the match.

Sh (Shots): Total number of shots taken by the team.

SoT (Shots on Target): Number of shots on target.

Dist: Average distance (likely in meters) from which shots were taken.

FK: Number of free kicks taken.

PK (Penalty Kicks): Number of penalty kicks scored.

PKatt (Penalty Kicks Attempted): Number of penalty kicks attempted.

Season: The season year.

Team: The team the data row is about.

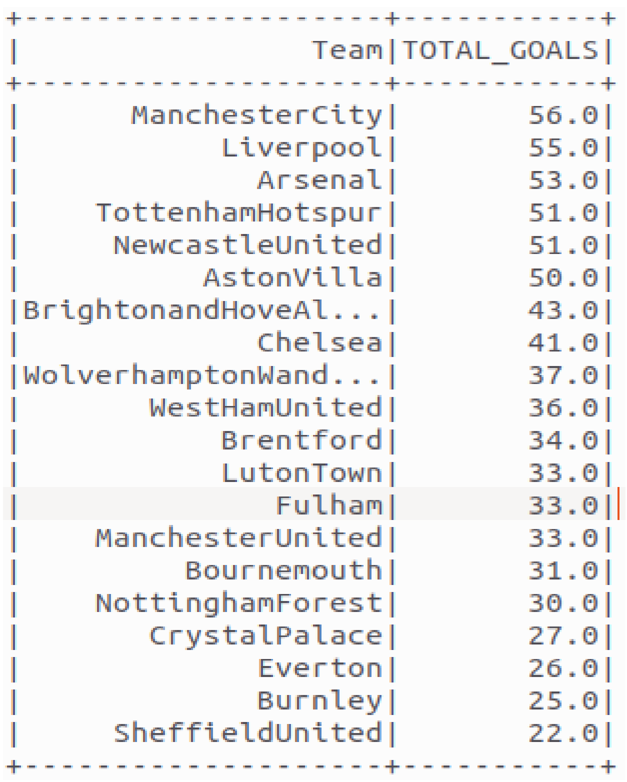
ANALYSIS 1

- **WHICH TEAM SCORED THE MOST NUMBER OF GOALS IN THE SEASON?**

CODE :

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as f
spark=SparkSession.builder.appName('PLproject').getOrCreate()
myrdd=spark.read.csv('hdfs://localhost:9000/sparkproject/matches.csv',header=True,inferSchema=
True)
goals_rdd=myrdd.groupBy('Team').agg(f.sum('GF').alias('TOTAL_GOALS'))
final_out=goals_rdd.sort(f.desc('TOTAL_GOALS'))
final_out.show()
```

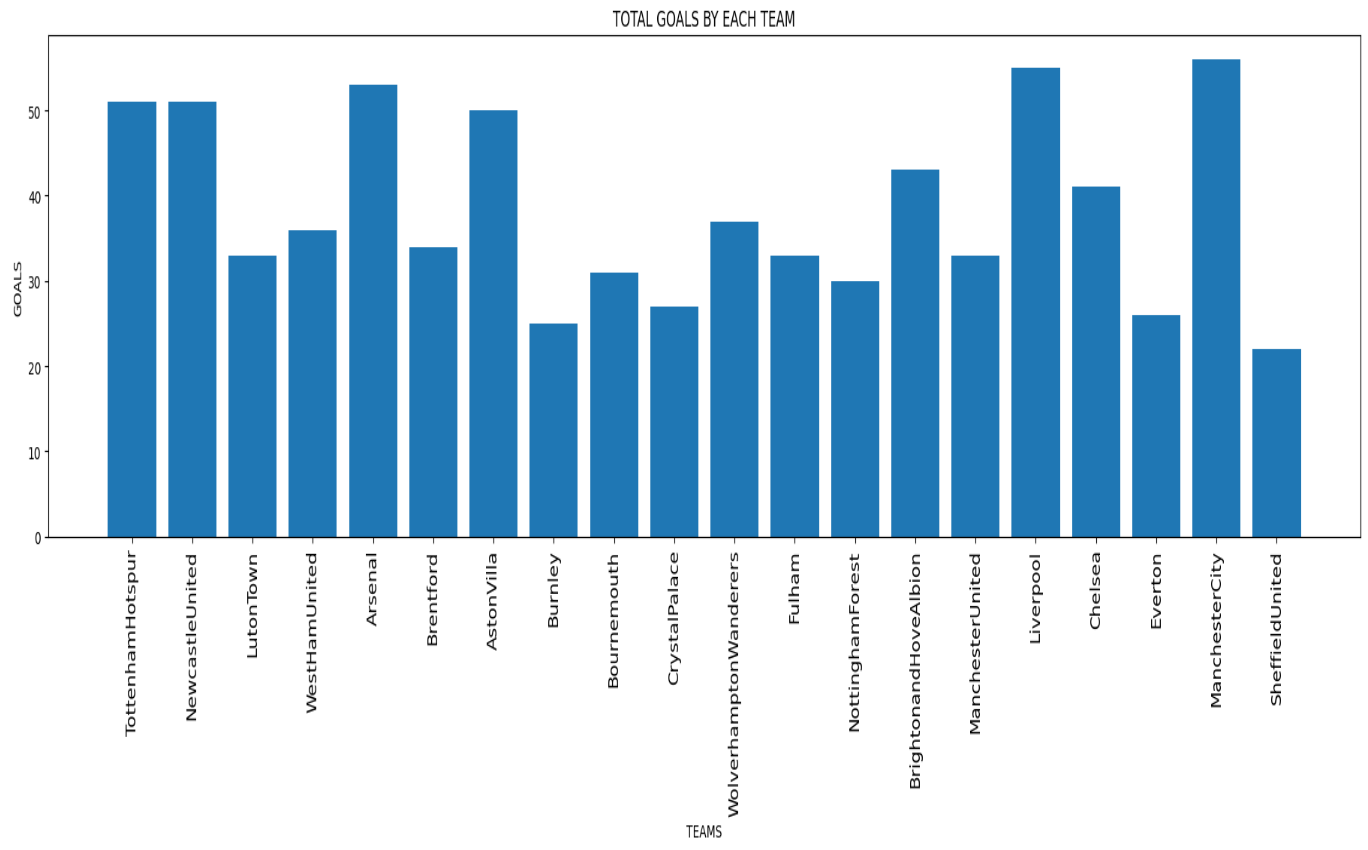
RESULT:



```
+--------------------+-----------+
|                Team|TOTAL_GOALS|
+--------------------+-----------+
|     ManchesterCity|       56.0|
|           Liverpool|       55.0|
|             Arsenal|       53.0|
|    TottenhamHotspur|       51.0|
|    NewcastleUnited|       51.0|
|          AstonVilla|       50.0|
|BrightonandHoveAl...|       43.0|
|             Chelsea|       41.0|
|WolverhamptonWand...|       37.0|
|       WestHamUnited|       36.0|
|           Brentford|       34.0|
|           LutonTown|       33.0|
|              Fulham|       33.0|
|    ManchesterUnited|       33.0|
|         Bournemouth|       31.0|
|    NottinghamForest|       30.0|
|       CrystalPalace|       27.0|
|             Everton|       26.0|
|             Burnley|       25.0|
|     SheffieldUnited|       22.0|
+--------------------+-----------+
```

PLOTTING:

```
import matplotlib.pyplot as plt
import pandas as pd
goal_graph=goals_rdd.toPandas()
plt.bar(goal_graph['Team'],goal_graph['TOTAL_GOALS'])
plt.xticks(rotation='vertical',fontsize=8)
plt.tight_layout()
plt.xlabel('TEAMS')
plt.ylabel('GOALS')
plt.title('TOTAL GOALS BY EACH TEAM')
plt.show()
```



CONCLUSION:

Top Performers: Manchester City stands out as the top-performing team, securing the lead with an impressive 56 goals. Liverpool closely trails with 55 goals, showcasing a fierce competition for the top spot. Arsenal completes the top three, demonstrating a formidable offensive presence with 53 goals.

Mid-Table Dynamics: Teams like Tottenham Hotspur, Newcastle United, and Aston Villa fall within the mid-range, emphasizing the competitive balance in this segment. Their goal-scoring performances, hovering around the 50-51 range, indicate closely contested matches and a balanced middle tier.

Challenges for Bottom Half: The bottom half of the table features teams like Wolverhampton Wanderers, West Ham United, Brentford, Luton Town, Fulham, and Manchester United, each facing unique challenges in goal-scoring. Their goal totals range f rom 36 to 33, pointing to potential areas for improvement and tactical adjustments.

Competitive Balance: The season witnessed a competitive balance, with several teams closely matched in terms of total goals. This balance suggests that matches were fiercely contested, contributing to an engaging and unpredictable football season.

ANALYSIS 2

- **CREATE A POINTS TABLE BASED ON THE DATA?**

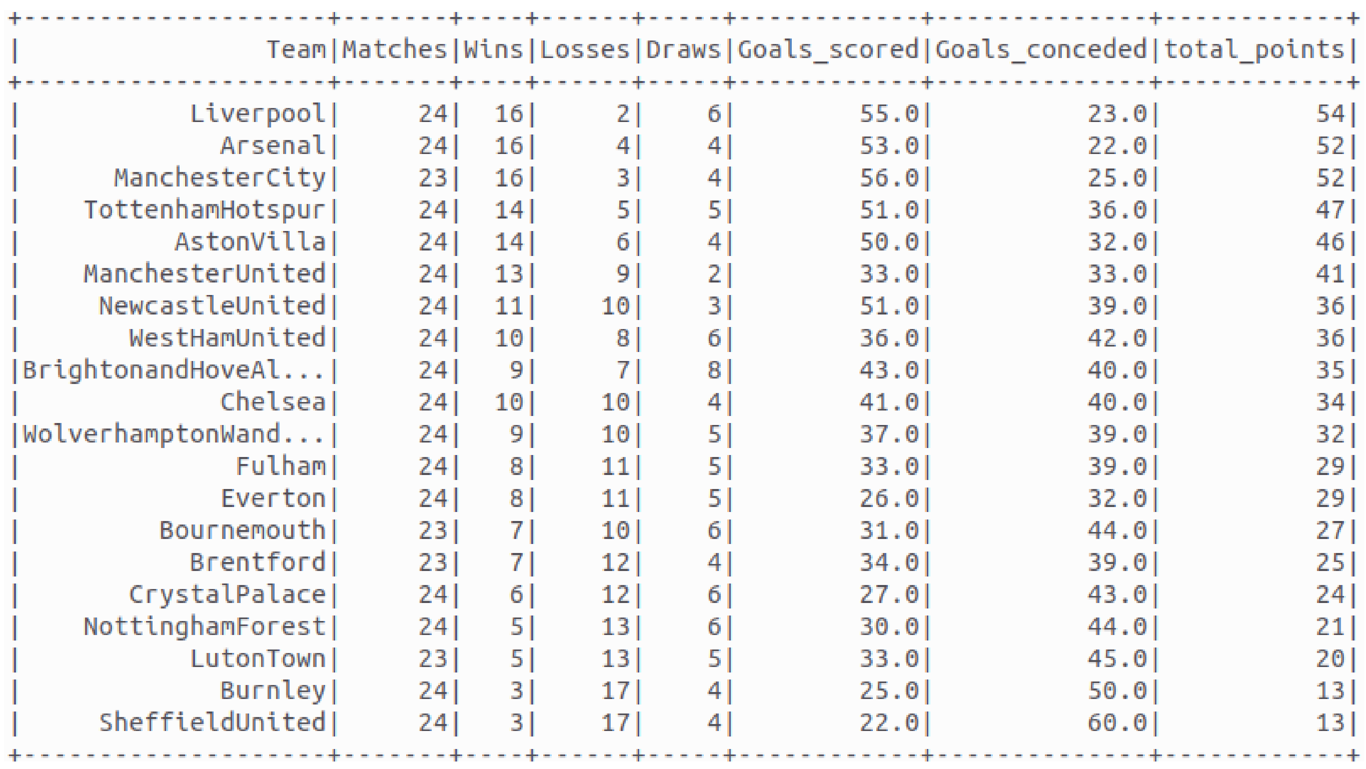
POINTS TABLE SHOULD HAVE THE FOLLOWING COLUMNS :

- TEAM,MATCHES PLAYED,WON,LOST,DRAW,GOALS SCORED,GOALS CONCEDED,POINTS
- ASSUMING FOR EACH MATCHES WON THE TEAM GETS 3 POINTS ,IF ITS A DRAW THE TEAM GET 1 POINT AND 0 IF LOST

CODE:

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as f
spark=SparkSession.builder.appName('PLproject').getOrCreate()
myrdd=spark.read.csv('hdfs://localhost:9000/sparkproject/matches.csv',header=True,inferSchema=
True)
select_rdd=myrdd.select('Team','Result','GF','GA')
count_result=select_rdd.withColumn('points',f.when(select_rdd.Result=='W',3).
                        when(select_rdd.Result=='D',1).otherwise(0))
totals_table = (count_result.groupBy('Team').agg(f.count('Team').alias('Matches'),
                        f.sum(f.when(count_result.Result == 'W', 1).otherwise(0)).alias('Wins'),
                        f.sum(f.when(count_result.Result == 'L', 1).otherwise(0)).alias('Losses'),
                        f.sum(f.when(count_result.Result == 'D', 1).otherwise(0)).alias('Draws'),
                        f.sum('GF').alias('Goals_scored'),f.sum('GA').alias('Goals_conceded'),
                        f.sum(count_result.points).alias('total_points')))
totals_table.sort(f.desc('total_points')).show()
```

RESULT:



```
+--------------------+-------+----+------+-----+------------+--------------+------------+
|                Team|Matches|Wins|Losses|Draws|Goals_scored|Goals_conceded|total_points|
+--------------------+-------+----+------+-----+------------+--------------+------------+
|           Liverpool|     24|  16|     2|    6|        55.0|          23.0|          54|
|             Arsenal|     24|  16|     4|    4|        53.0|          22.0|          52|
|       ManchesterCity|     23|  16|     3|    4|        56.0|          25.0|          52|
|     TottenhamHotspur|     24|  14|     5|    5|        51.0|          36.0|          47|
|          AstonVilla|     24|  14|     6|    4|        50.0|          32.0|          46|
|    ManchesterUnited|     24|  13|     9|    2|        33.0|          33.0|          41|
|     NewcastleUnited|     24|  11|    10|    3|        51.0|          39.0|          36|
|        WestHamUnited|     24|  10|     8|    6|        36.0|          42.0|          36|
|BrightonandHoveAl...|     24|   9|     7|    8|        43.0|          40.0|          35|
|             Chelsea|     24|  10|    10|    4|        41.0|          40.0|          34|
|WolverhamptonWand...|     24|   9|    10|    5|        37.0|          39.0|          32|
|              Fulham|     24|   8|    11|    5|        33.0|          39.0|          29|
|             Everton|     24|   8|    11|    5|        26.0|          32.0|          29|
|          Bournemouth|     23|   7|    10|    6|        31.0|          44.0|          27|
|           Brentford|     23|   7|    12|    4|        34.0|          39.0|          25|
|        CrystalPalace|     24|   6|    12|    6|        27.0|          43.0|          24|
|     NottinghamForest|     24|   5|    13|    6|        30.0|          44.0|          21|
|           LutonTown|     23|   5|    13|    5|        33.0|          45.0|          20|
|             Burnley|     24|   3|    17|    4|        25.0|          50.0|          13|
|      SheffieldUnited|     24|   3|    17|    4|        22.0|          60.0|          13|
+--------------------+-------+----+------+-----+------------+--------------+------------+
```

CONCLUSION:

Points Leaders:

- **Liverpool:** Leading with 54 points, Liverpool showcases a strong balance of offensive and defensive capabilities. Their impressive goal difference, coupled with 16 wins and 6 draws, positions them as a formidable force.
- **Arsenal:** Following closely with 52 points, Arsenal's consistent wins (16) and a robust defensive record (only 22 goals conceded) underscore their competitiveness.

Manchester City's Efficiency:

- Despite playing one game less, Manchester City demonstrates efficiency with 52 points. Their high goal-scoring rate (56 goals) suggests offensive prowess, though a relatively higher number of goals conceded (25) raises considerations.

Mid-Table Battles:

- Teams like Newcastle United, West Ham United, Brighton and Hove Albion, and Chelsea are engaged in tight mid-table battles, where small performance improvements could lead to significant shifts in standings.

Struggles for Burnley and Sheffield United:

- **Burnley** and Sheffield United encounter difficulties, reflected in their low points (13 each) and substantial goals conceded. Rebuilding defensive structures is imperative for their resurgence.

Draw Impact on Points:

- Teams with a considerable number of draws, such as Fulham and Crystal Palace, face a dilemma as draws impact total points. Converting draws into wins could elevate their standings.

Goal Difference as Performance Indicator:

- The correlation between positive goal differences and higher points is evident. Teams with solid defensive capabilities and efficient goal-scoring performances tend to be at the top of the table.

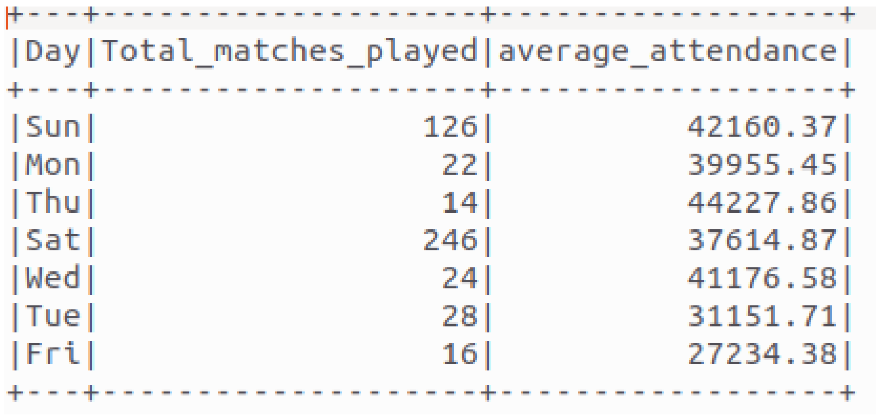
ANALYSIS 3

- **DURING WHICH DAY OF THE WEEK MOST MATCHES TOOK PLACE AND ALSO THE AVERAGE ATTENDANCE DURING EACH DAY?**

CODE:

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as f
spark=SparkSession.builder.appName('PLproject').getOrCreate()
myrdd=spark.read.csv('hdfs://localhost:9000/sparkproject/matches.csv',header=True,inferSchema=
True)
week_day=myrdd.groupBy('Day').agg((f.count('Day').alias('Total_matches_played')) ,
            (f.round(f.avg('Attendance'),2).alias('average_attendance')))
week_day.show()
```

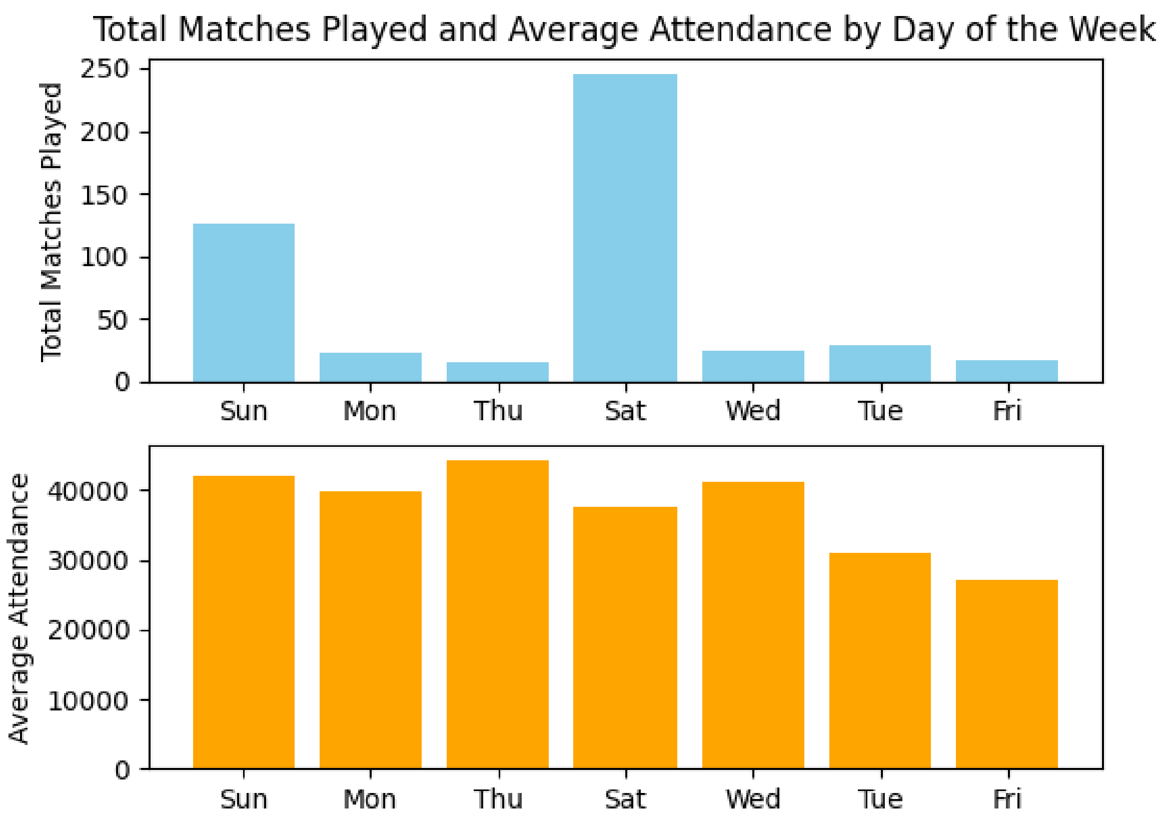
RESULT:



```
+---+--------------------+------------------+
|Day|Total_matches_played|average_attendance|
+---+--------------------+------------------+
|Sun|                 126|          42160.37|
|Mon|                  22|          39955.45|
|Thu|                  14|          44227.86|
|Sat|                 246|          37614.87|
|Wed|                  24|          41176.58|
|Tue|                  28|          31151.71|
|Fri|                  16|          27234.38|
+---+--------------------+------------------+
```

PLOTTING:

```
df =week_day.toPandas()

import matplotlib.pyplot as plt
import pandas as pd

ax1 = plt.subplot2grid((2, 1), (0, 0))
ax1.bar(df['Day'], df['Total_matches_played'], color='skyblue')
ax1.set_ylabel('Total Matches Played')
ax1.set_title('Total Matches Played and Average Attendance by Day of the Week')

ax2 = plt.subplot2grid((2, 1), (1, 0))
ax2.bar(df['Day'], df['average_attendance'], color='orange')
ax2.set_ylabel('Average Attendance')
plt.show()
```



CONCLUSION:

Most Popular Days:

- Saturday has the highest total matches played (246) and a relatively high average attendance (37614.87), indicating it might be the most popular day for matches.
- Sunday also has a significant number of matches (126) with a relatively high average attendance (42160.37).

Weekday Analysis:

- Weekdays (Monday to Friday) generally have fewer matches compared to weekends.
- Thursday has the highest average attendance (44227.86) among weekdays, followed by Wednesday (41176.58) and Monday (39955.45).

Low Attendance Days:

- Friday has the lowest total matches played (16) and the lowest average attendance (27234.38), suggesting it might be a less popular day for matches.
- There's a noticeable variability in average attendance across different days, indicating that certain days might attract more fans than others.

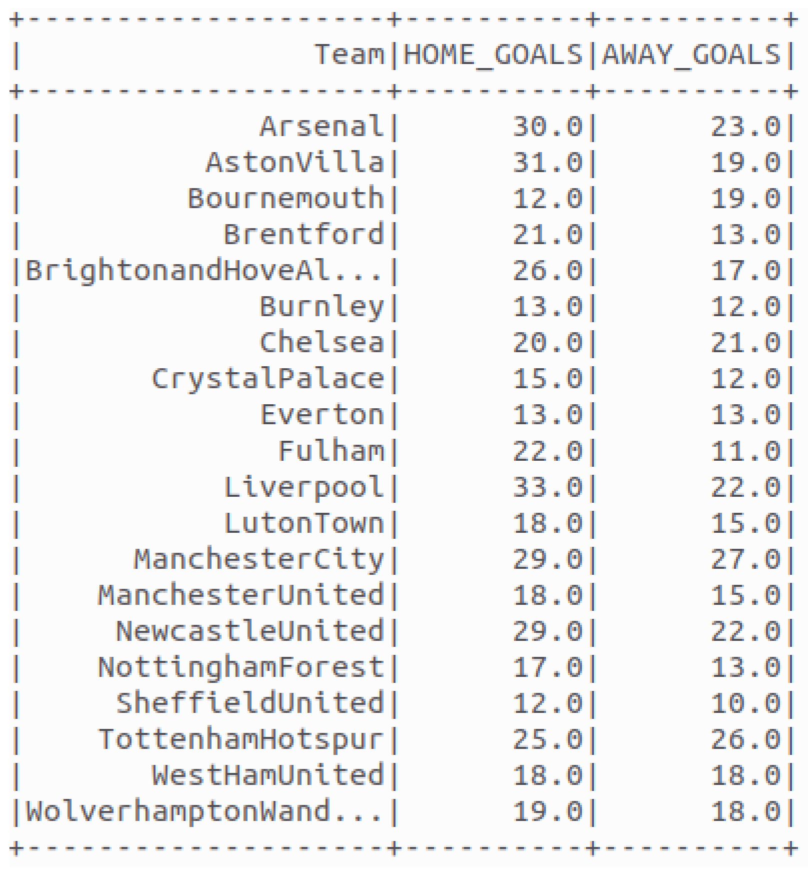
ANALYSIS 4:

- **TOTAL GOALS SCORED IN HOME AND AWAY MATCHES BY EACH TEAM?**

CODE:

```
from pyspark.sql import SparkSession

from pyspark.sql import functions f

spark=SparkSession.builder.appName('Plproject').getOrCreate()

myrdd=spark.read.csv('hdfs://localhost:9000/sparkproject/matches.csv',header=True,inferSchema=True)

home_goals=myrdd.filter(myrdd['Venue']=='Home').groupBy('Team').agg(f.sum('GF').alias('HOME_GOALS'))

away_goals=myrdd.filter(myrdd['Venue']=='Away').groupBy('Team').agg(f.sum('GF').alias('AWAY_GOALS'))

goal_count=home_goals.join(away_goals,'Team','full')

goal_count.show()
```

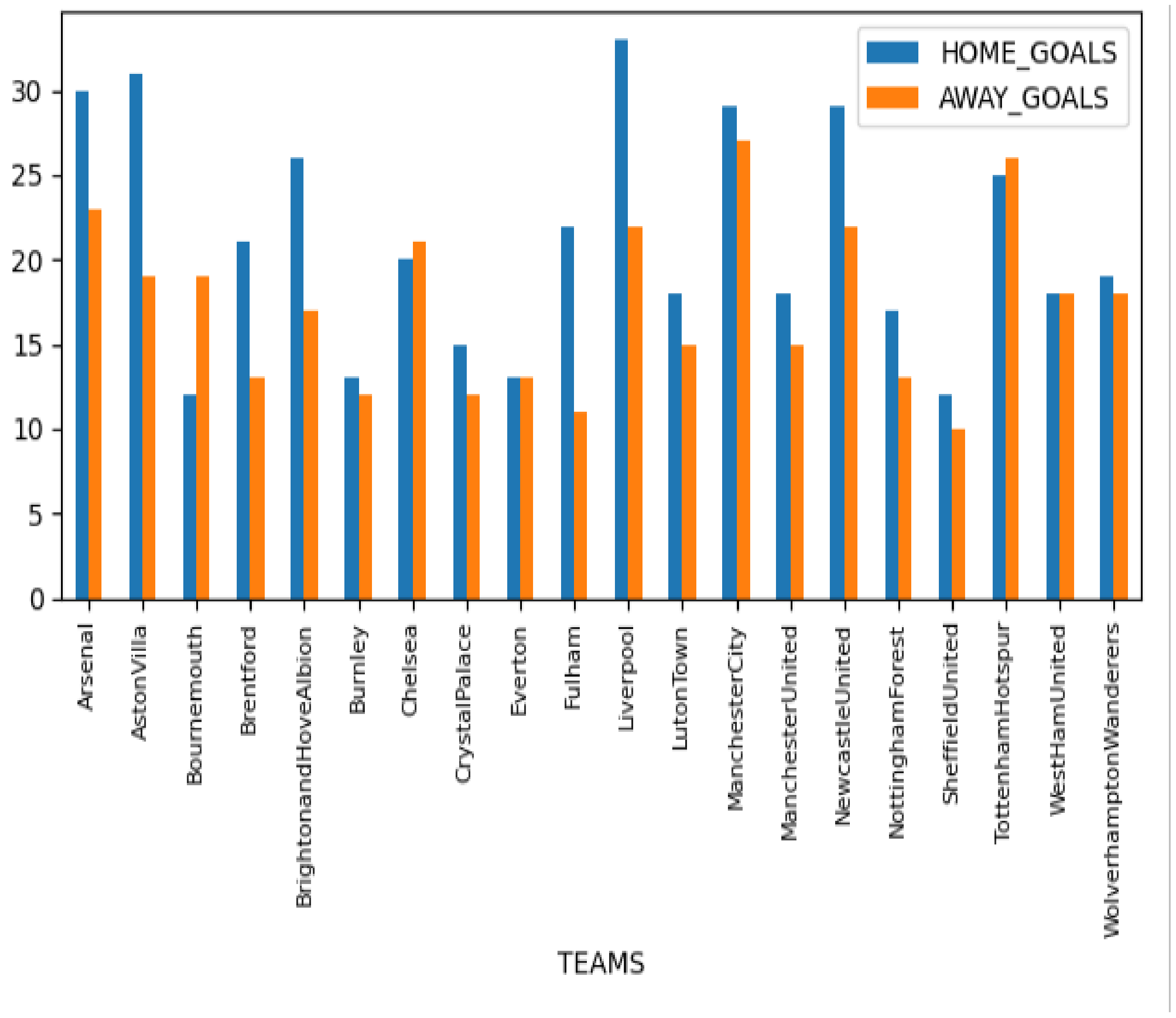
RESULT:



```
+--------------------+----------+----------+
|                Team|HOME_GOALS|AWAY_GOALS|
+--------------------+----------+----------+
|             Arsenal|      30.0|      23.0|
|          AstonVilla|      31.0|      19.0|
|         Bournemouth|      12.0|      19.0|
|           Brentford|      21.0|      13.0|
|BrightonandHoveAl...|      26.0|      17.0|
|             Burnley|      13.0|      12.0|
|             Chelsea|      20.0|      21.0|
|       CrystalPalace|      15.0|      12.0|
|             Everton|      13.0|      13.0|
|              Fulham|      22.0|      11.0|
|           Liverpool|      33.0|      22.0|
|           LutonTown|      18.0|      15.0|
|     ManchesterCity|       29.0|      27.0|
|   ManchesterUnited|       18.0|      15.0|
|    NewcastleUnited|       29.0|      22.0|
|   NottinghamForest|       17.0|      13.0|
|    SheffieldUnited|       12.0|      10.0|
|   TottenhamHotspur|       25.0|      26.0|
|       WestHamUnited|       18.0|      18.0|
|WolverhamptonWand...|      19.0|      18.0|
+--------------------+----------+----------+
```

PLOTTING:

```
import matplotlib.pyplot as plt
import pandas as pd
total_goals_rdd=goal_count.toPandas()
total_goals_rdd.plot(kind='bar', x='Team', y=['HOME_GOALS', 'AWAY_GOALS'])

plt.xticks(rotation='vertical',fontsize=8)
plt.tight_layout()
plt.xlabel('TEAMS')
plt.ylabel('GOALS')
plt.title('TOTAL HOME AND AWAY GOALS')
plt.show()
```



CONCLUSION:

Goal Disparities:

- There are variations in the number of goals scored by teams at home and away. Some teams score more goals at home, while others perform better in away matches.

Home Advantage:

- Teams like Liverpool, Arsenal, and Manchester City seem to have a strong home advantage, scoring significantly more goals at home compared to away.

Consistency:

- Some teams, such as Everton and Manchester United, have a relatively balanced performance in terms of goals scored at home and away.

Competitiveness:

- The competitiveness of a team can be gauged by comparing their home and away goal numbers. Teams with similar figures for both home and away goals may be considered more consistent and adaptable.

Potential Strategies:

- Coaches and analysts may use this data to develop strategies based on team performance in different match scenarios. For instance, understanding which teams tend to score more goals away might influence defensive strategies when facing them at their home ground.

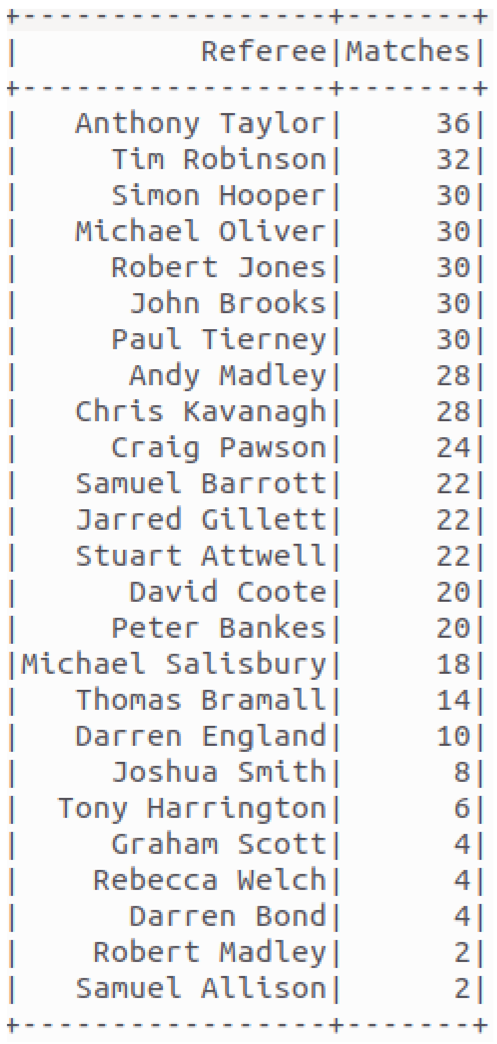
ANALYSIS 5:

- **WHICH REFEREE OFFICIATED MOST NUMBER OF GAMES IN THE SEASON?**

CODE:

```
referee_rdd=myrdd.groupBy('Referee').agg(f.count('Referee').alias('Matches'))
final_referee_rdd=referee_rdd.sort(f.desc('Matches'))
final_referee_rdd.show(final_referee_rdd.count())
```

RESULT:



```
+-----------------+-------+
|          Referee|Matches|
+-----------------+-------+
|   Anthony Taylor|     36|
|     Tim Robinson|     32|
|     Simon Hooper|     30|
|   Michael Oliver|     30|
|     Robert Jones|     30|
|      John Brooks|     30|
|     Paul Tierney|     30|
|      Andy Madley|     28|
|   Chris Kavanagh|     28|
|     Craig Pawson|     24|
|   Samuel Barrott|     22|
|   Jarred Gillett|     22|
|   Stuart Attwell|     22|
|      David Coote|     20|
|     Peter Bankes|     20|
|Michael Salisbury|     18|
|   Thomas Bramall|     14|
|   Darren England|     10|
|     Joshua Smith|      8|
|  Tony Harrington|      6|
|     Graham Scott|      4|
|    Rebecca Welch|      4|
|      Darren Bond|      4|
|    Robert Madley|      2|
|   Samuel Allison|      2|
+-----------------+-------+
```

CONCLUSION:

- Analyzing the data on the number of matches officiated by referees in the given season reveals several interesting patterns. Anthony Taylor emerges as the most active referee, overseeing 36 matches, showcasing a high level of consistency and reliability in his officiating duties. Tim Robinson follows closely with 32 matches, contributing significantly to the referee lineup.

- The range of match counts among referees is notable, ranging from the highest at 36 to the lowest at 2. This variance might be attributed to factors such as referee experience, performance assessments, or specific scheduling considerations.

- The distribution of match counts among referees in this season raises intriguing questions about the factors influencing their assignments and the potential implications for officiating dynamics. Further investigation into performance metrics, experience levels, and league policies would contribute to a more comprehensive understanding of the referee landscape in the given season.