IPL DATA ANALYSIS

PRESENTED BY

Syed Abadul Rahaman

Title Winner

Mumbai Indians

SK Raina

4548

SL Malinga

170

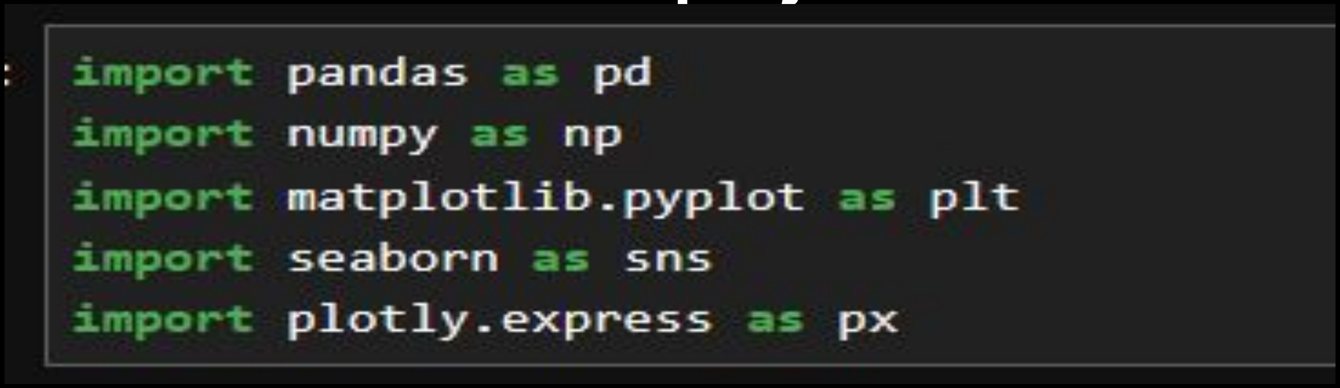
INTRODUCTION

In this Project , we will work on IPL Data Analysis and Visualization Project using Python where we will explore interesting insights from the data of IPL matches like most run by a player, most wicket taken by a player, and much more from IPL season 2008-2017.

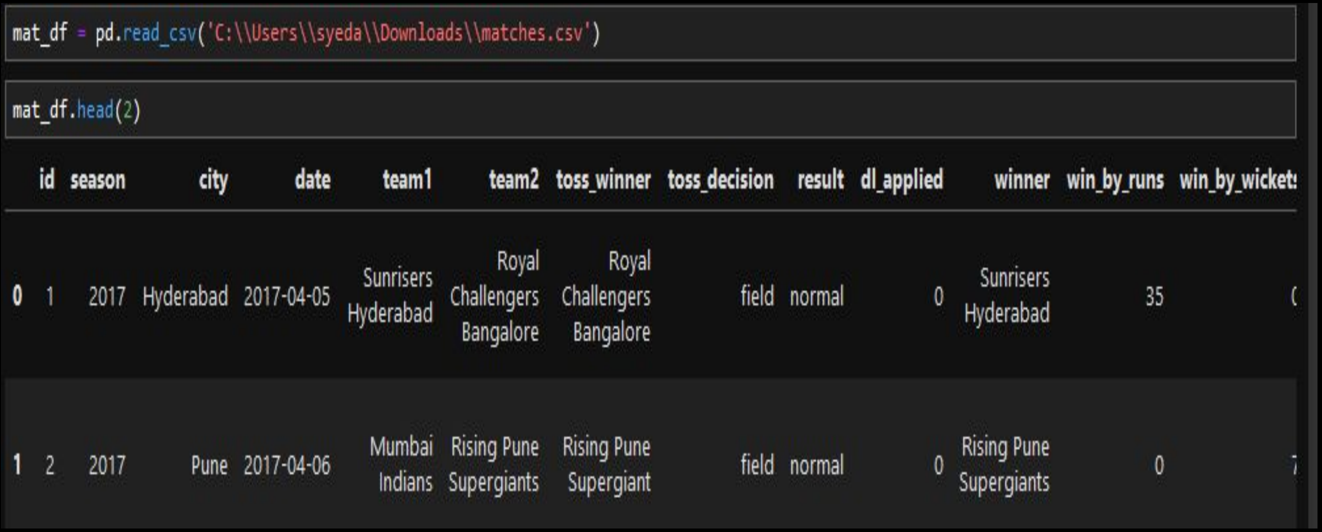
LOADING DATA ……

Libraries and Dataset Used for this project :--

List of Libraries
1. Pandas
2. NumPy
3. Matplotlib
4. Seaborn
5. Statistics



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```



```
mat_df = pd.read_csv('C:\\Users\\syeda\\Downloads\\matches.csv')
```

```
mat_df.head(2)
```

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wicket
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	(
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiants	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiants	0	7

Dataset
1. matches.csv
2. deliveries.csv

DATA UNDERSTANDING

1. match.csv

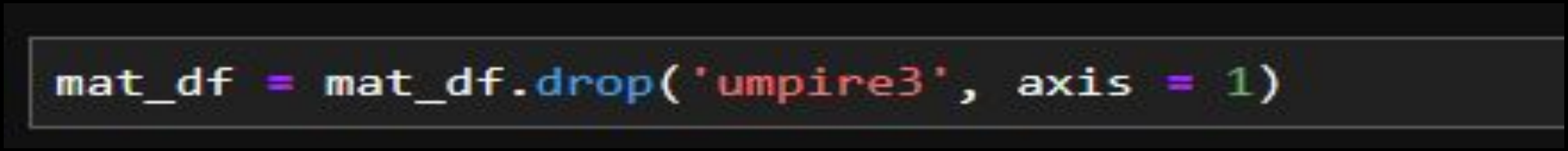
- This dataset provides matches information in IPL till 2017. It gives information on teams, cities, venues, toss decisions, winners and umpires.
- There are a total of 636 rows and 18 columns in the dataframe.
- There are 5 columns with a numeric data-type and 13 columns with an object datatype.

2. deliveries.csv

- This dataset provides deliveries information in IPL till 2017. It gives ball by ball details of all matches in IPL along with total runs scored by each batsman, wickets taken by each bowler and extras provided in each match.
- There are a total of 150460 rows and 21 columns in the dataframe.
- There are 13 columns with a numeric data-type and 8 columns with an object datatype.

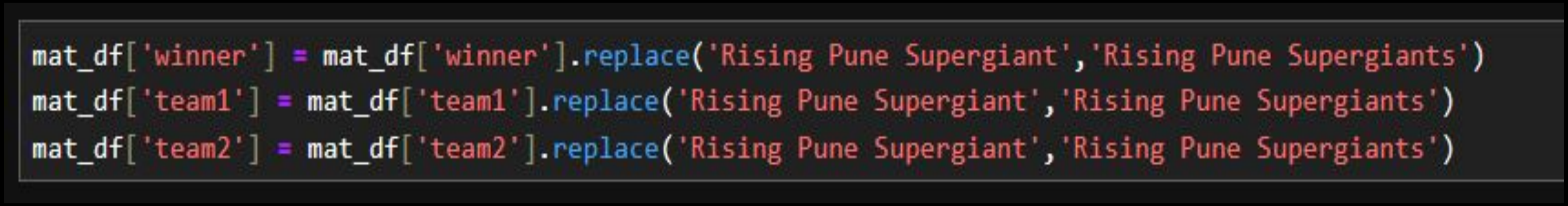
DATA CLEANING

umpire3 column has 100% missing values. hence dropping that column

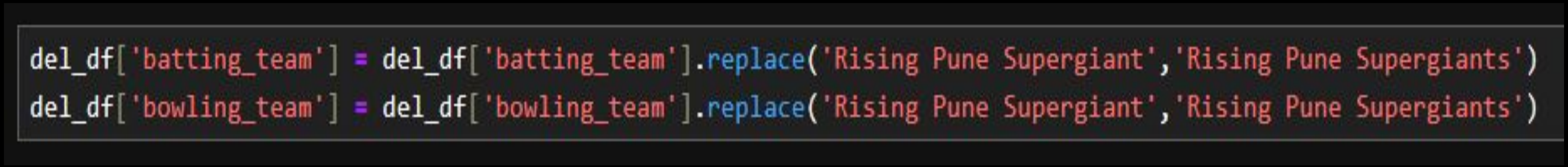


```
mat_df = mat_df.drop('umpire3', axis = 1)
```

team1, team2 and winner all 3 columns have rising pune supergiant as well as rising pune supergiants. so we can just keep one of the names

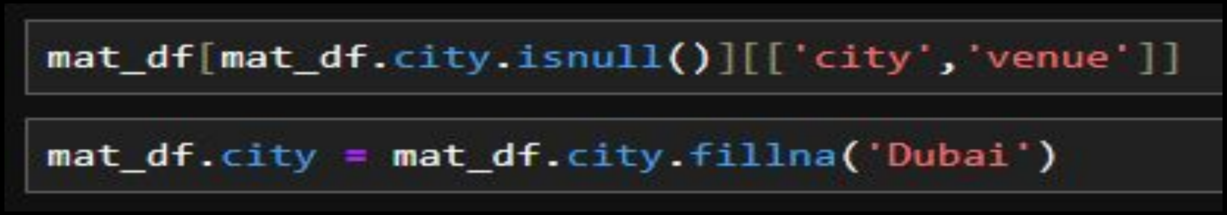


```
mat_df['winner'] = mat_df['winner'].replace('Rising Pune Supergiant','Rising Pune Supergiants')
mat_df['team1'] = mat_df['team1'].replace('Rising Pune Supergiant','Rising Pune Supergiants')
mat_df['team2'] = mat_df['team2'].replace('Rising Pune Supergiant','Rising Pune Supergiants')
```



```
del_df['batting_team'] = del_df['batting_team'].replace('Rising Pune Supergiant','Rising Pune Supergiants')
del_df['bowling_team'] = del_df['bowling_team'].replace('Rising Pune Supergiant','Rising Pune Supergiants')
```

city has missing 7 values



```
mat_df[mat_df.city.isnull()][['city','venue']]
```

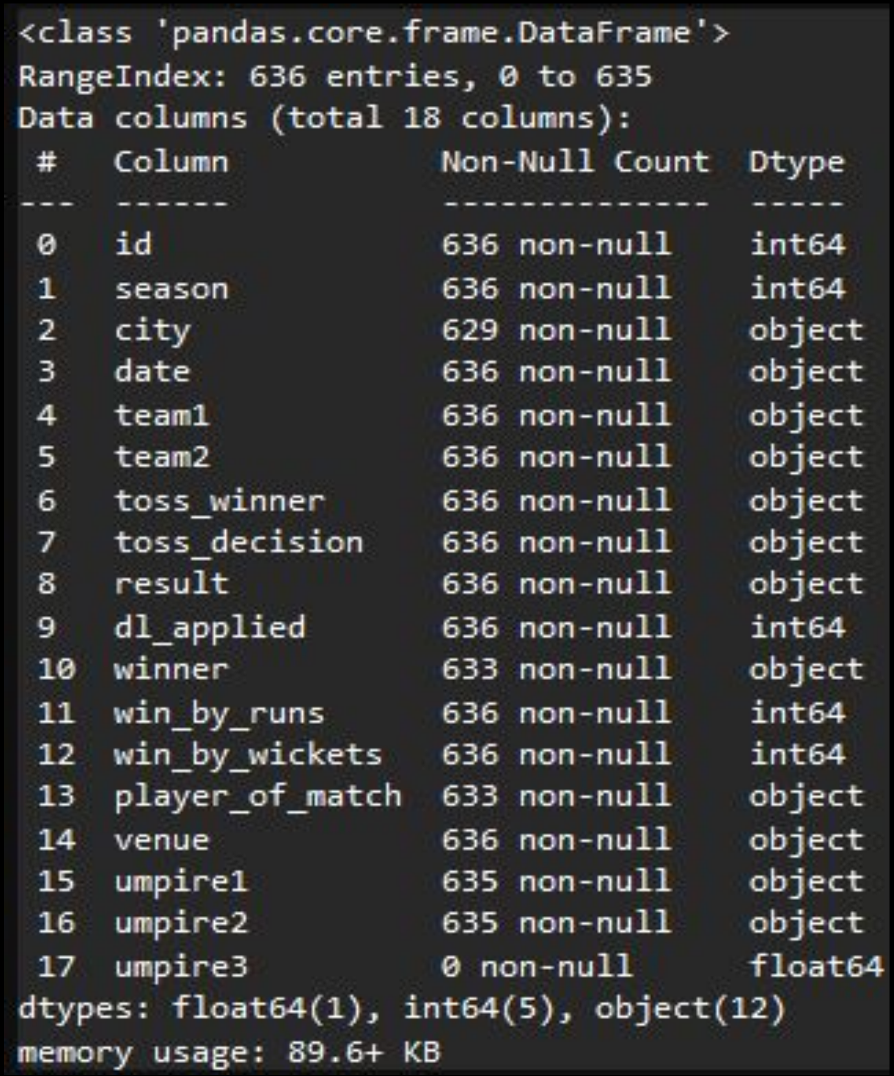
```
mat_df.city = mat_df.city.fillna('Dubai')
```

DATA INFORMATION

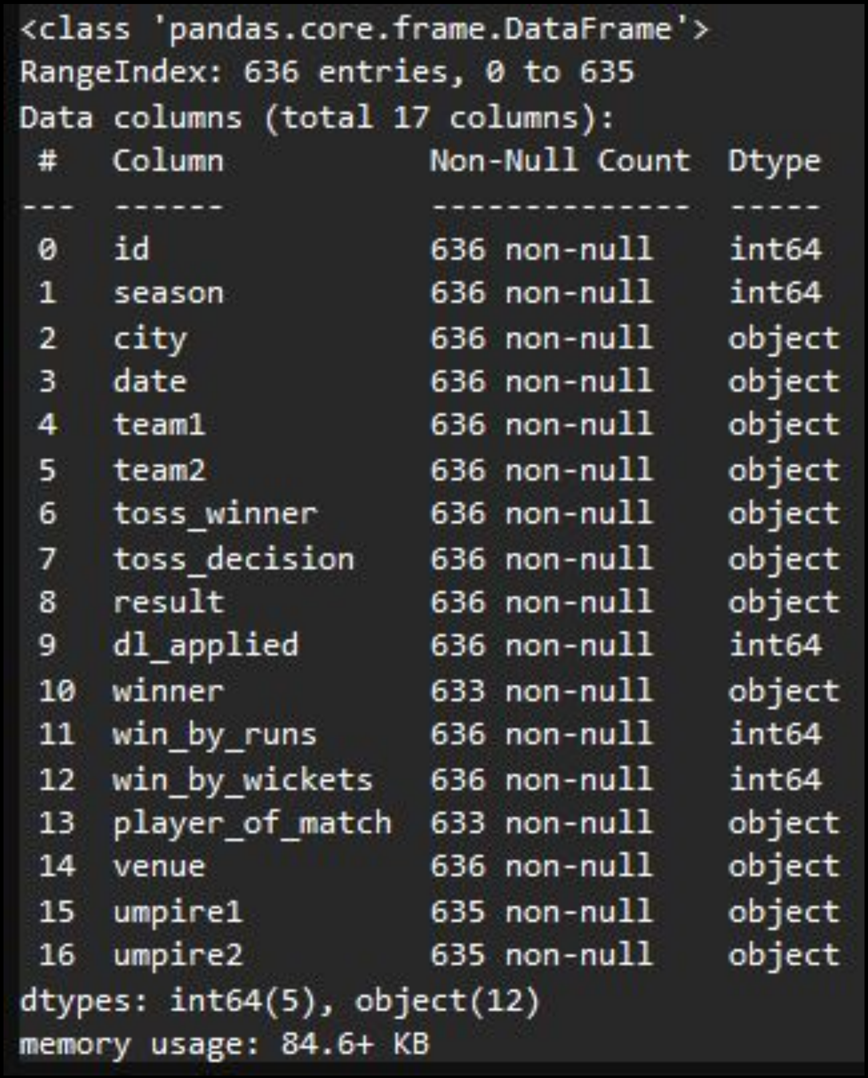
BEFORE

match.info()

AFTER



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               636 non-null    int64  
 1   season           636 non-null    int64  
 2   city             629 non-null    object 
 3   date             636 non-null    object 
 4   team1            636 non-null    object 
 5   team2            636 non-null    object 
 6   toss_winner      636 non-null    object 
 7   toss_decision    636 non-null    object 
 8   result           636 non-null    object 
 9   dl_applied       636 non-null    int64  
 10  winner           633 non-null    object 
 11  win_by_runs      636 non-null    int64  
 12  win_by_wickets   636 non-null    int64  
 13  player_of_match  633 non-null    object 
 14  venue            636 non-null    object 
 15  umpire1          635 non-null    object 
 16  umpire2          635 non-null    object 
 17  umpire3          0 non-null      float64
dtypes: float64(1), int64(5), object(12)
memory usage: 89.6+ KB
```



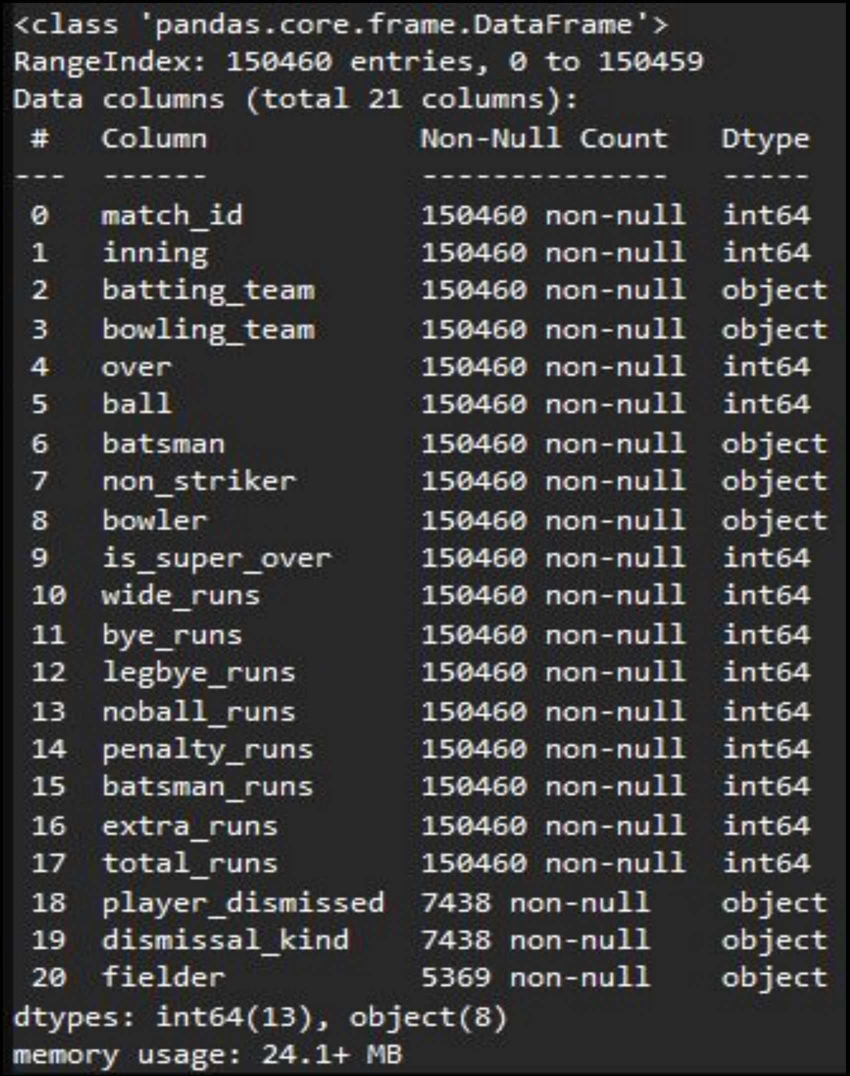
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               636 non-null    int64 
 1   season           636 non-null    int64 
 2   city             636 non-null    object
 3   date             636 non-null    object
 4   team1            636 non-null    object
 5   team2            636 non-null    object
 6   toss_winner      636 non-null    object
 7   toss_decision    636 non-null    object
 8   result           636 non-null    object
 9   dl_applied       636 non-null    int64 
 10  winner           633 non-null    object
 11  win_by_runs      636 non-null    int64 
 12  win_by_wickets   636 non-null    int64 
 13  player_of_match  633 non-null    object
 14  venue            636 non-null    object
 15  umpire1          635 non-null    object
 16  umpire2          635 non-null    object
dtypes: int64(5), object(12)
memory usage: 84.6+ KB
```

DATA INFORMATION

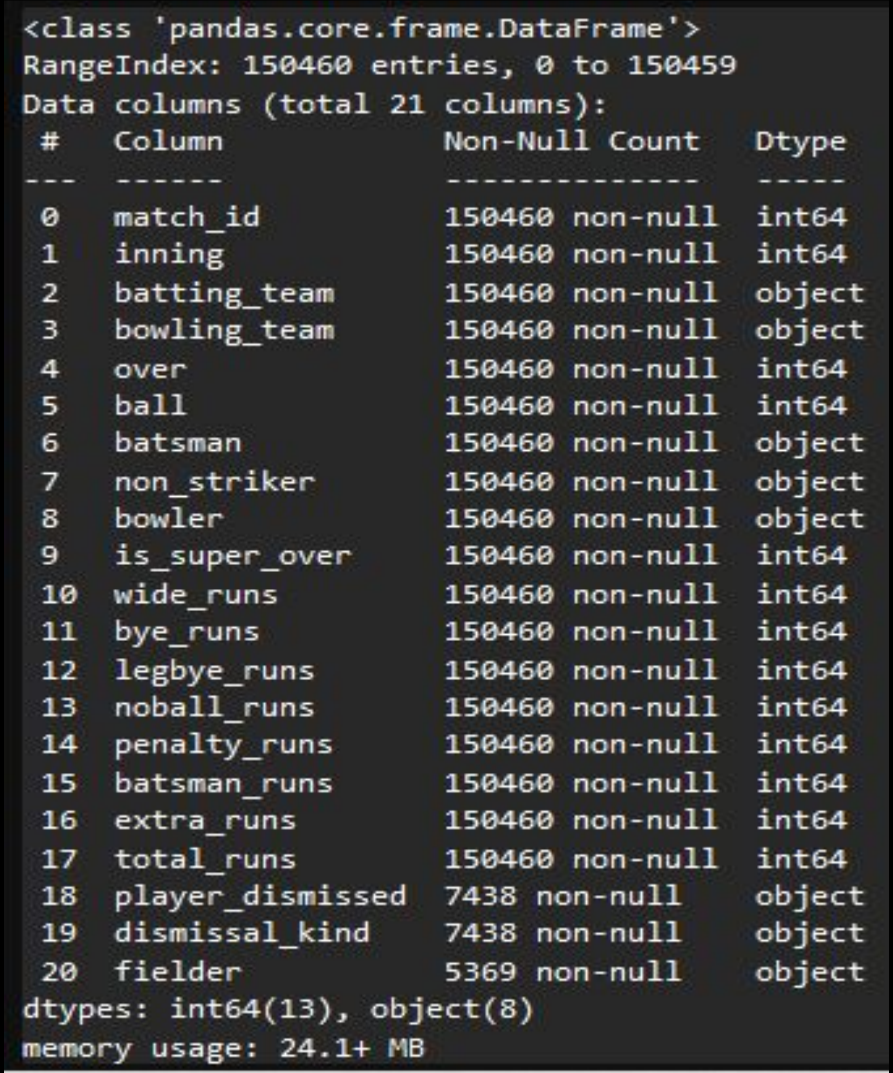
BEFORE

deliveries.info()

AFTER

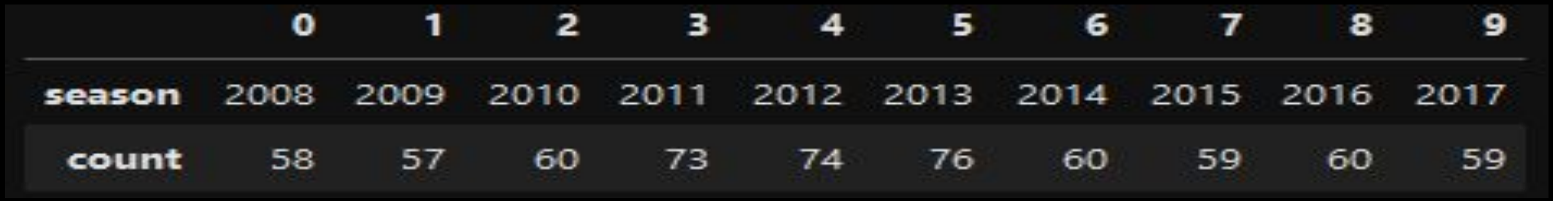


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150460 entries, 0 to 150459
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   match_id          150460 non-null  int64 
 1   inning            150460 non-null  int64 
 2   batting_team      150460 non-null  object
 3   bowling_team      150460 non-null  object
 4   over              150460 non-null  int64 
 5   ball              150460 non-null  int64 
 6   batsman           150460 non-null  object
 7   non_striker       150460 non-null  object
 8   bowler            150460 non-null  object
 9   is_super_over     150460 non-null  int64 
 10  wide_runs         150460 non-null  int64 
 11  bye_runs          150460 non-null  int64 
 12  legbye_runs       150460 non-null  int64 
 13  noball_runs       150460 non-null  int64 
 14  penalty_runs      150460 non-null  int64 
 15  batsman_runs      150460 non-null  int64 
 16  extra_runs        150460 non-null  int64 
 17  total_runs        150460 non-null  int64 
 18  player_dismissed  7438 non-null    object
 19  dismissal_kind    7438 non-null    object
 20  fielder           5369 non-null    object
dtypes: int64(13), object(8)
memory usage: 24.1+ MB
```



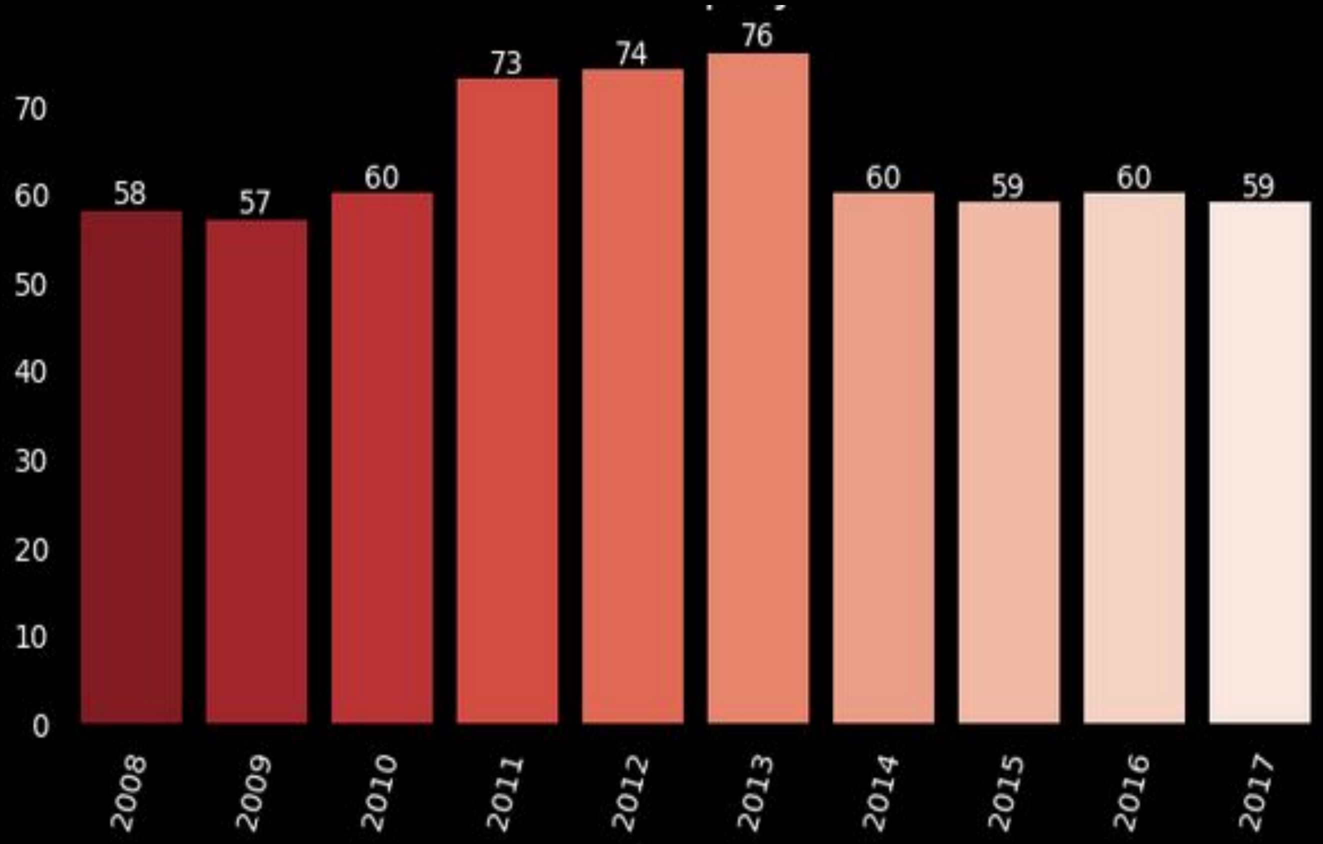
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150460 entries, 0 to 150459
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   match_id          150460 non-null  int64 
 1   inning            150460 non-null  int64 
 2   batting_team      150460 non-null  object
 3   bowling_team      150460 non-null  object
 4   over              150460 non-null  int64 
 5   ball              150460 non-null  int64 
 6   batsman           150460 non-null  object
 7   non_striker       150460 non-null  object
 8   bowler            150460 non-null  object
 9   is_super_over     150460 non-null  int64 
 10  wide_runs         150460 non-null  int64 
 11  bye_runs          150460 non-null  int64 
 12  legbye_runs       150460 non-null  int64 
 13  noball_runs       150460 non-null  int64 
 14  penalty_runs      150460 non-null  int64 
 15  batsman_runs      150460 non-null  int64 
 16  extra_runs        150460 non-null  int64 
 17  total_runs        150460 non-null  int64 
 18  player_dismissed  7438 non-null    object
 19  dismissal_kind    7438 non-null    object
 20  fielder           5369 non-null    object
dtypes: int64(13), object(8)
memory usage: 24.1+ MB
```

MATCHES PLAYED EACH SEASON

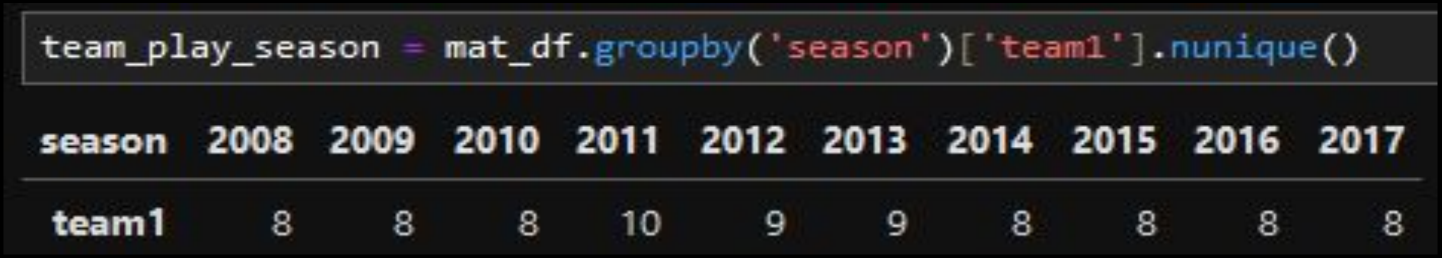


	0	1	2	3	4	5	6	7	8	9
season	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
count	58	57	60	73	74	76	60	59	60	59

- **2011-2013** have more matches being played than other seasons
- All other seasons have approximately 58-60 matches while 2011-2013 have more than 70 matches



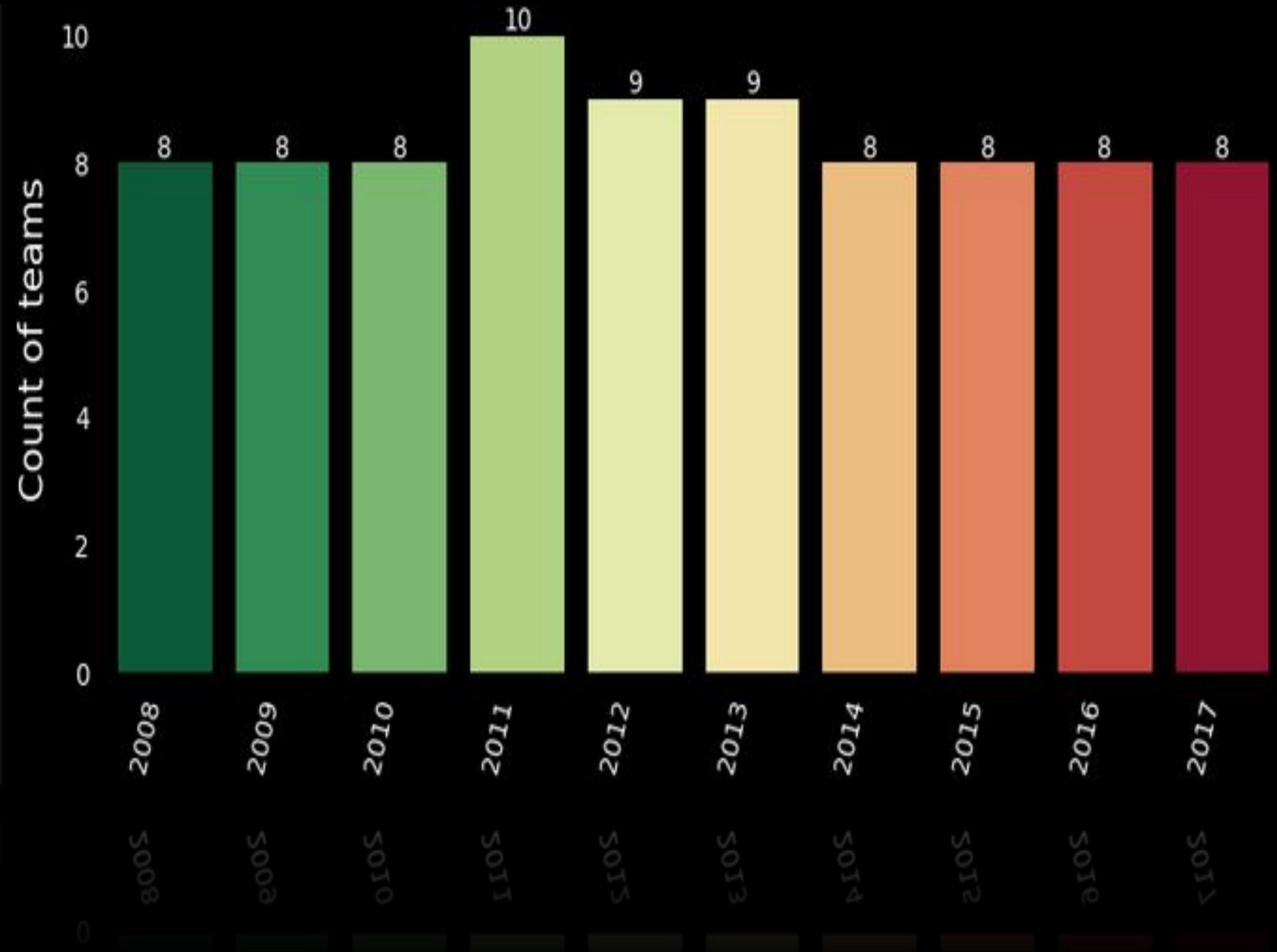
TEAMS PLAYED IN EACH SEASON



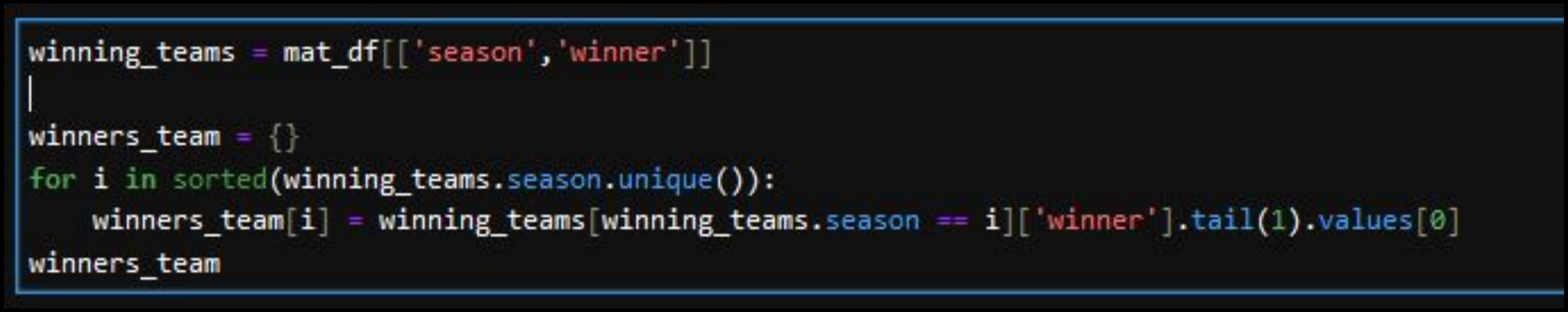
```python
team_play_season = mat_df.groupby('season')['team1'].nunique()
```

season	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
team1	8	8	8	10	9	9	8	8	8	8

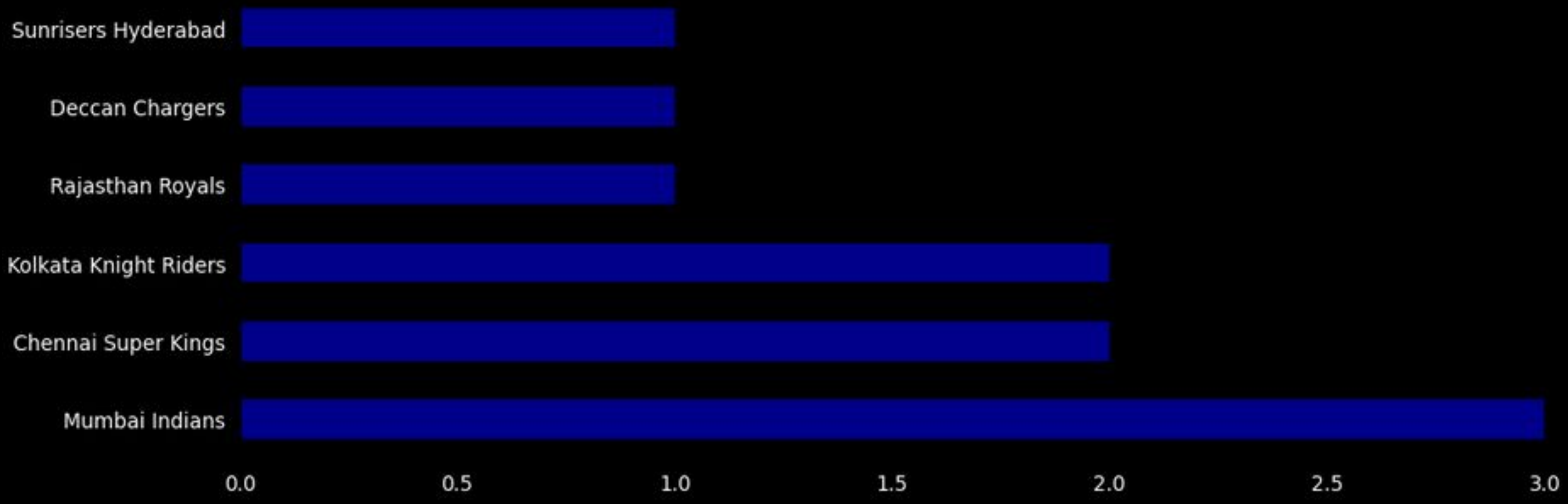
- **10 teams** played in 2011 and **9 teams** each in 2012 and 2013
- This explains why 2011-2013 have seen more matches being played than other seasons



WINNER ACROSS 10 SEASON 2008-2017

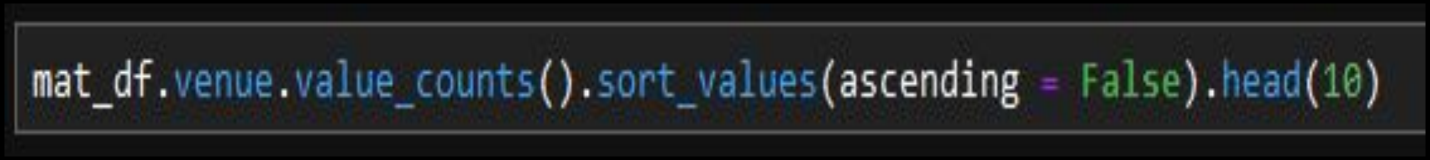


```
winning_teams = mat_df[['season','winner']]

winners_team = {}
for i in sorted(winning_teams.season.unique()):
    winners_team[i] = winning_teams[winning_teams.season == i]['winner'].tail(1).values[0]
winners_team
```

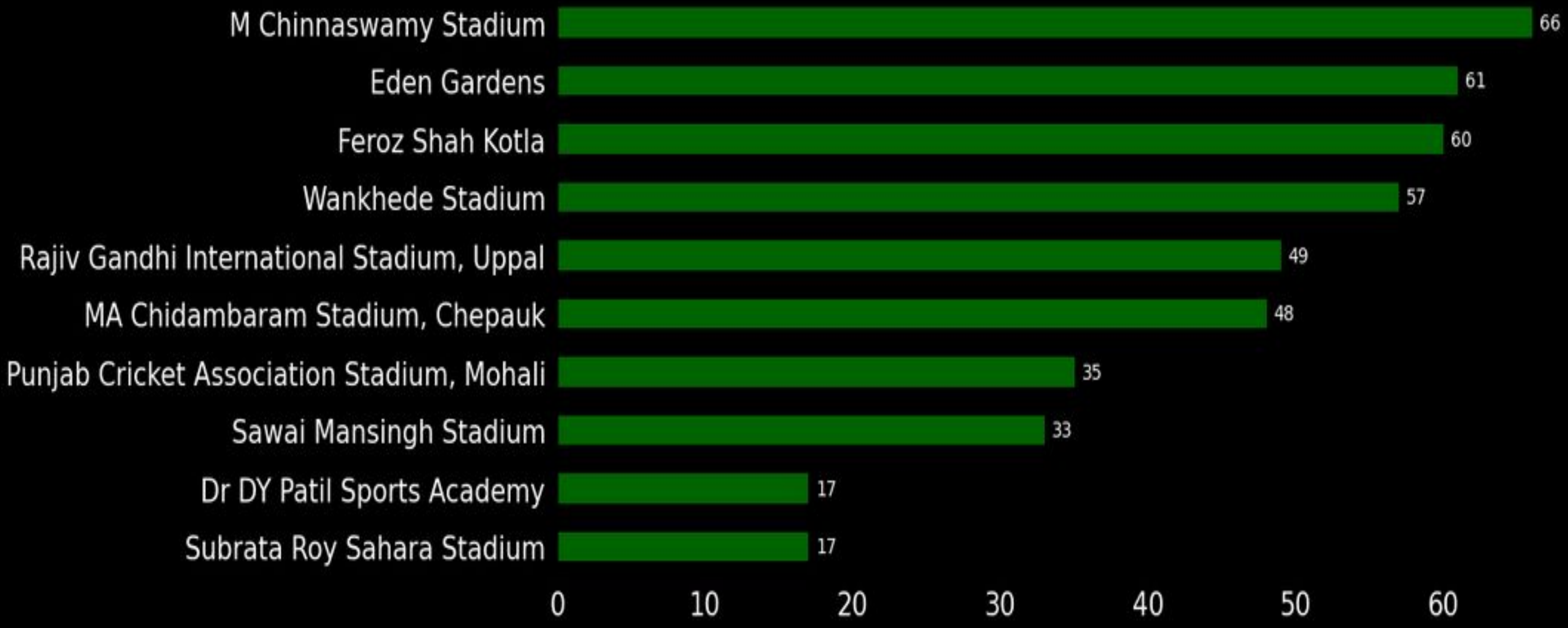


- **MI** has won 3 times.
- **CSK** and **KKR** have both won 2 times each.
- Actually Hyderabad team has also won 2 matches under 2 franchise name - Deccan Chargers and Sunrisers Hyderabad

Top Venue for IPL Matches

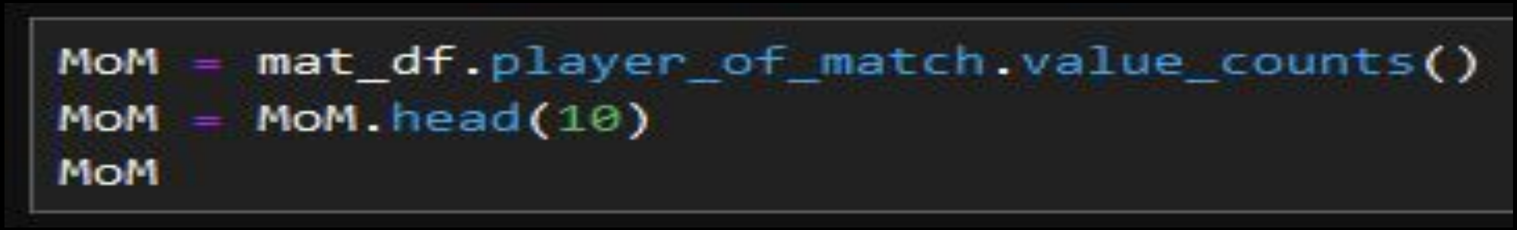


```
mat_df.venue.value_counts().sort_values(ascending = False).head(10)
```



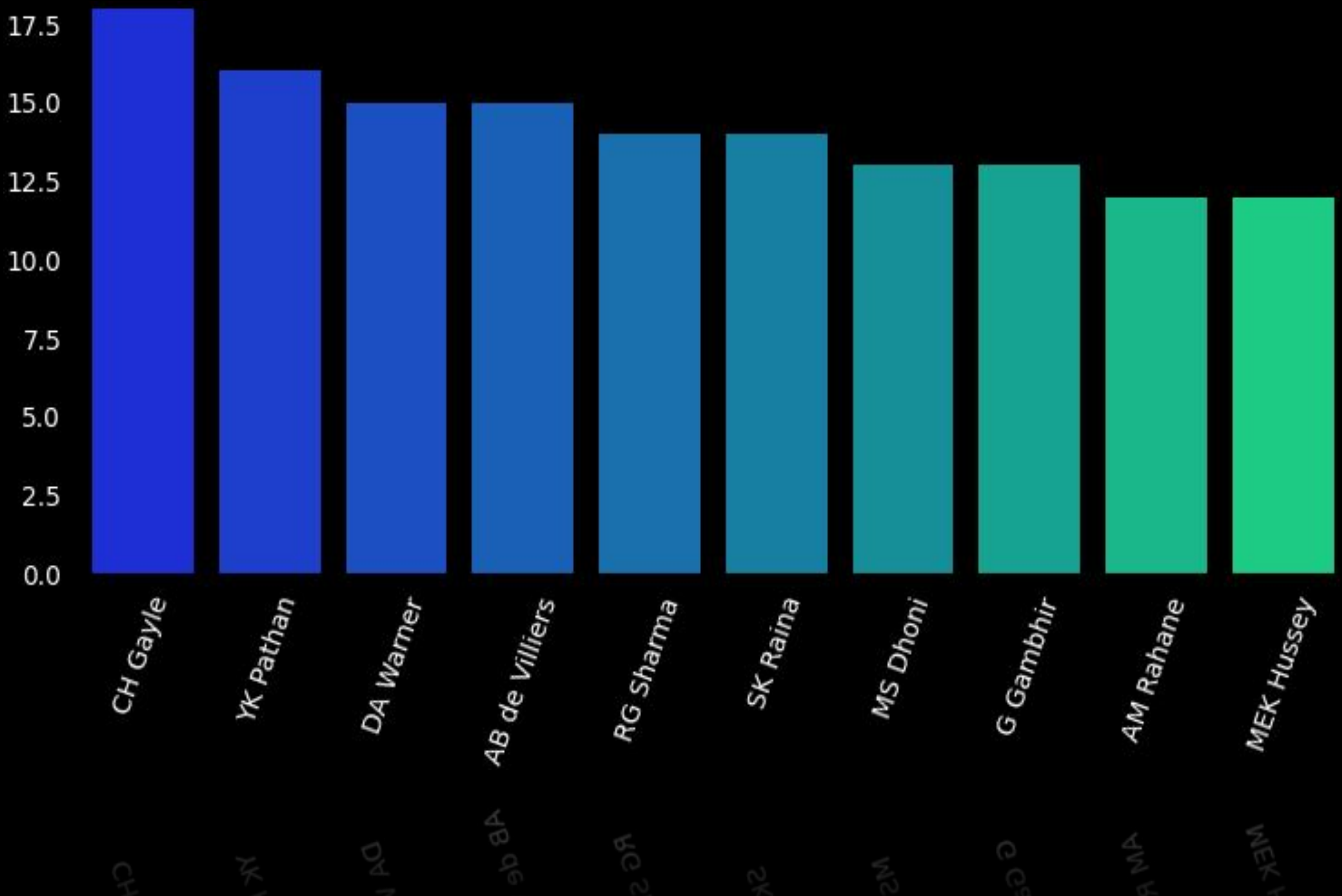
- **M Chinnaswamy Stadium in Bengaluru has hosted the highest number of matches so far in IPL followed by Eden Gardens in Kolkata**

PLAYERS WITH MOST MoM AWARDS

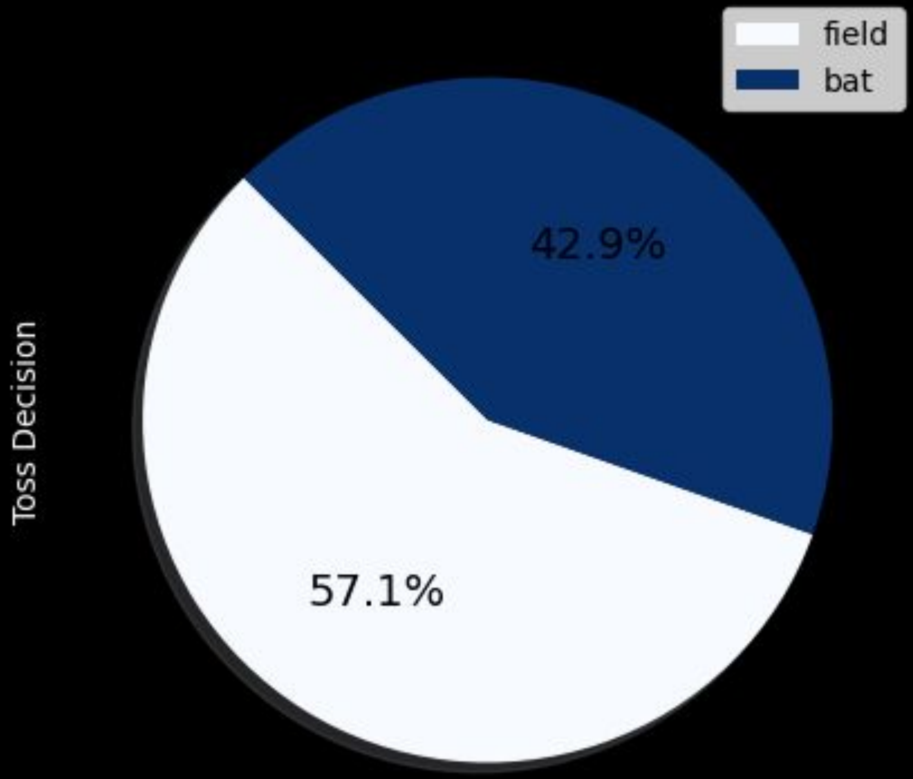


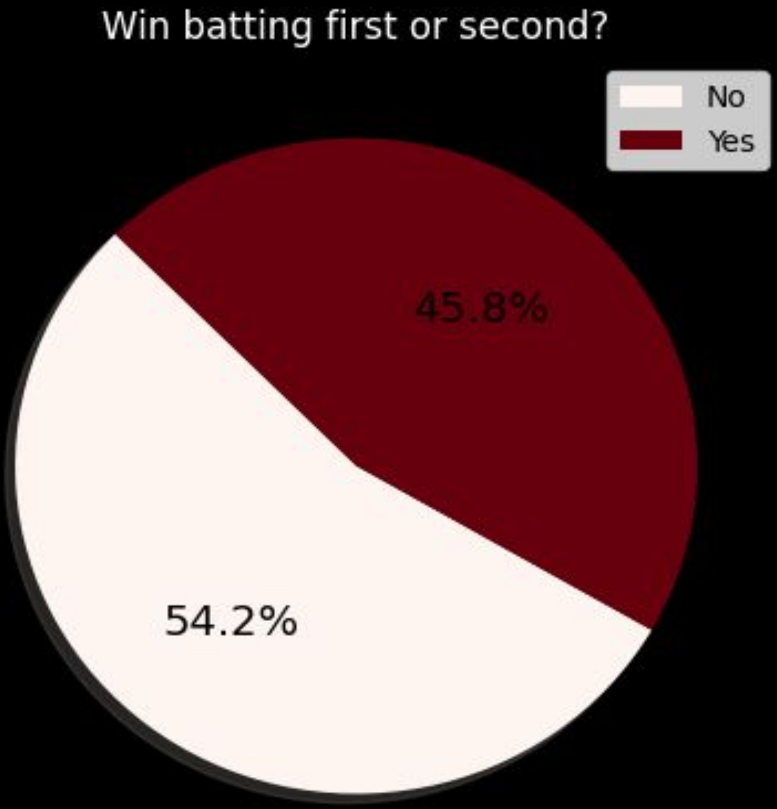
```
MoM = mat_df.player_of_match.value_counts()
MoM = MoM.head(10)
MoM
```

- Chris Gayle has so far won the most number of MoM awards followed by YK Pathan.
- Players like MS Dhoni and Gautam Gambhir, who have also been team captains, appear on the list, indicating that leadership and experience can also correlate with match-winning performances.



TOSS DECISION BY CAPTAIN

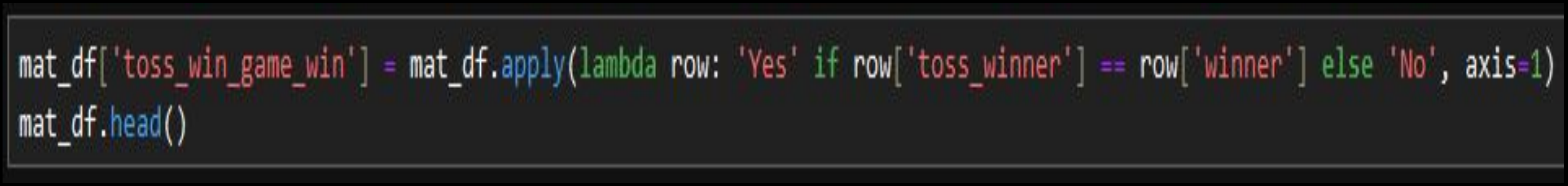




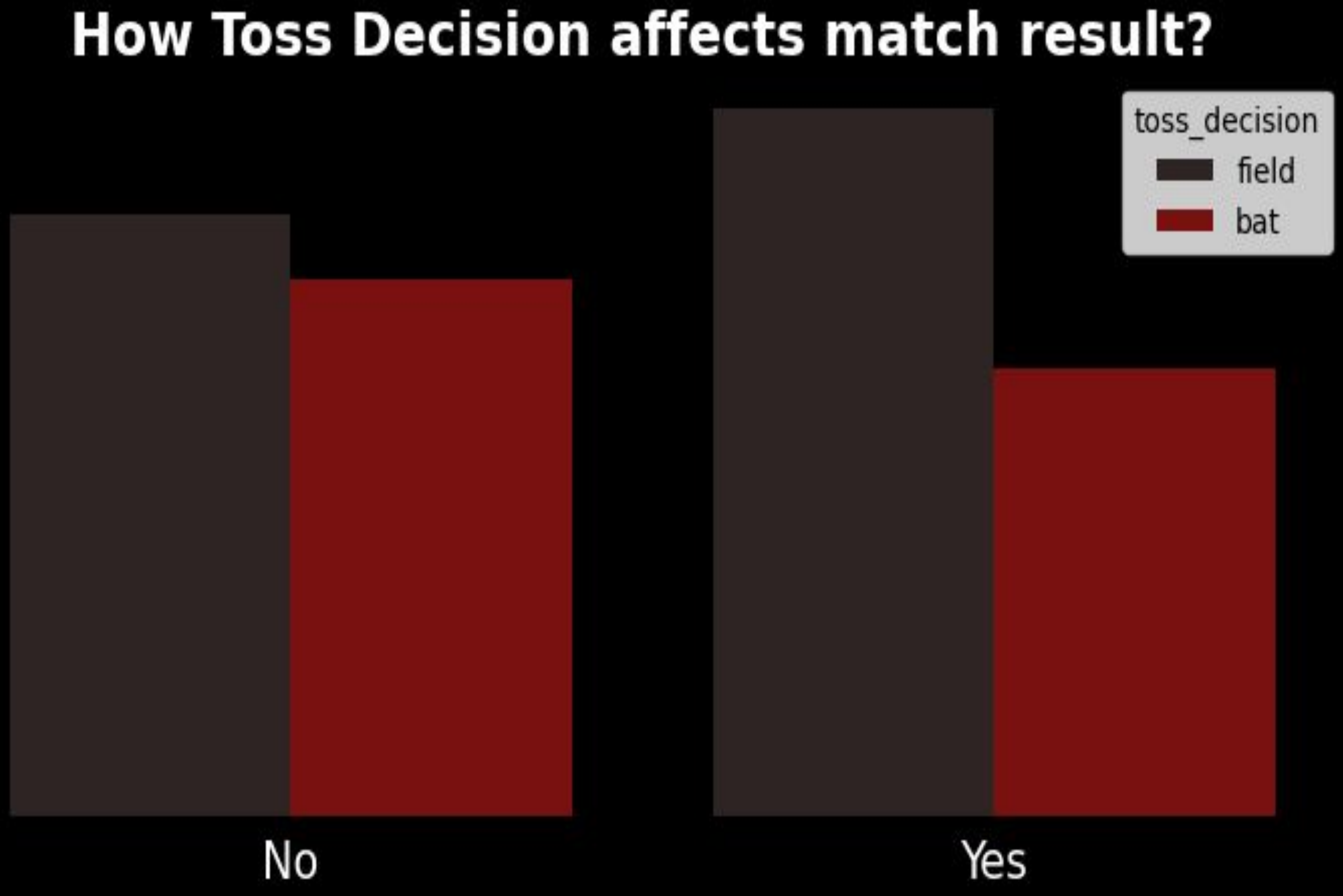
- Close to 60% times teams who have won tosses have decided to chase down

- Teams batting second have won 54% times.

TOSS CHOICES AND MATCH SUCCESS



```
mat_df['toss_win_game_win'] = mat_df.apply(lambda row: 'Yes' if row['toss_winner'] == row['winner'] else 'No', axis=1)
mat_df.head()
```



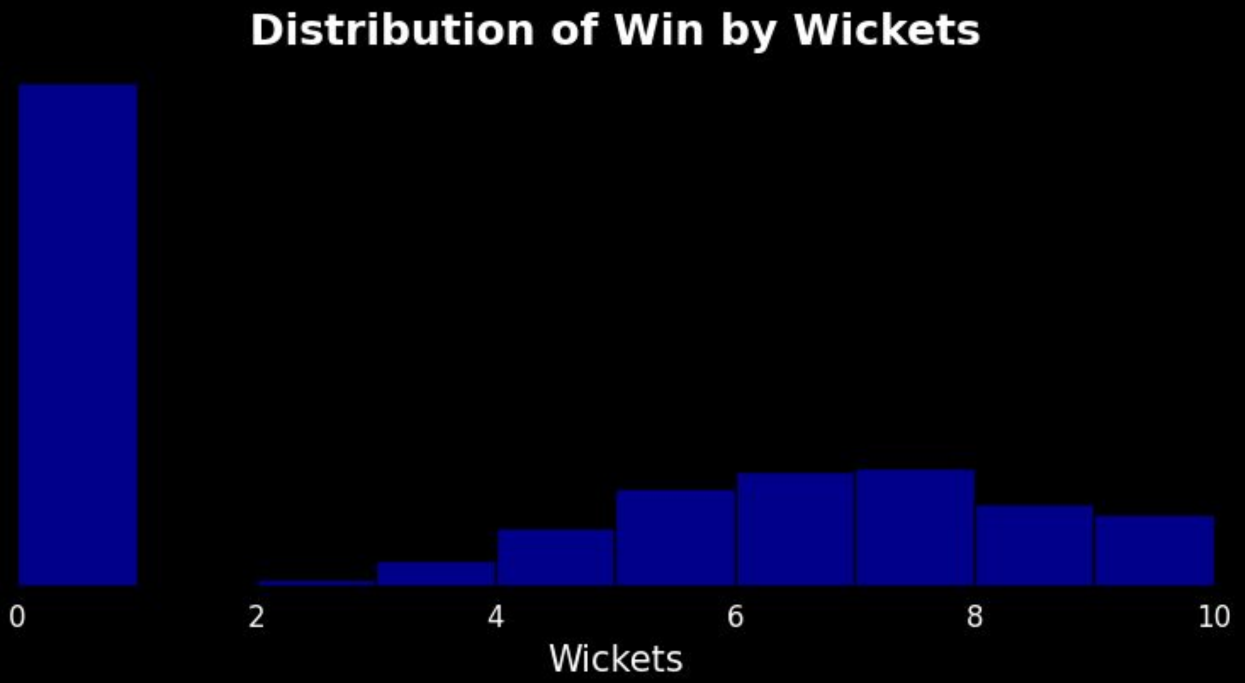
How Toss Decision affects match result?

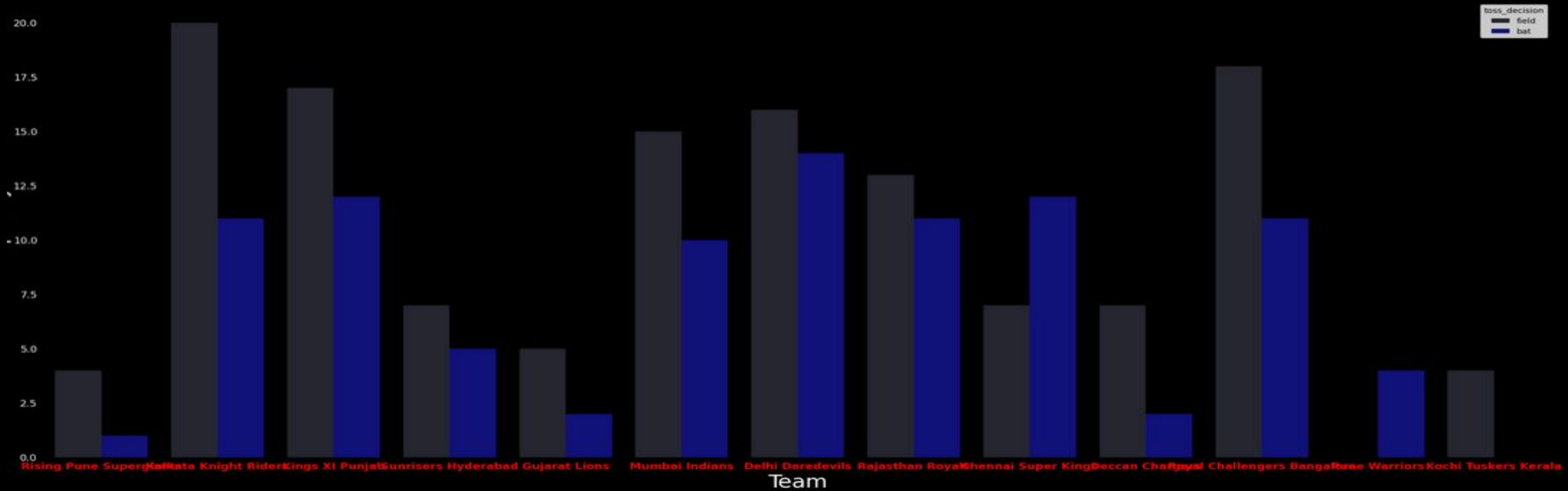
- Teams winning tosses and electing to field first have won most number of times.

TEAMS WINNING BY BIG WICKET MARGIN

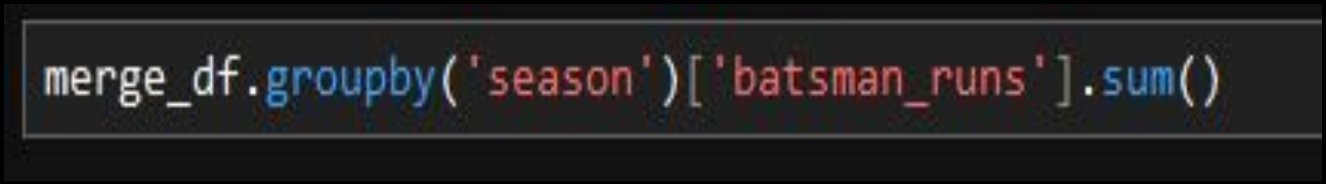
- The distribution is right-skewed, which indicates that most matches are won by a relatively small number of wickets.

- KKR, Kings XI Punjab, Delhi Daredevils and RCB have won by good wicket margins over the years and they have all decided to field first after winning toss

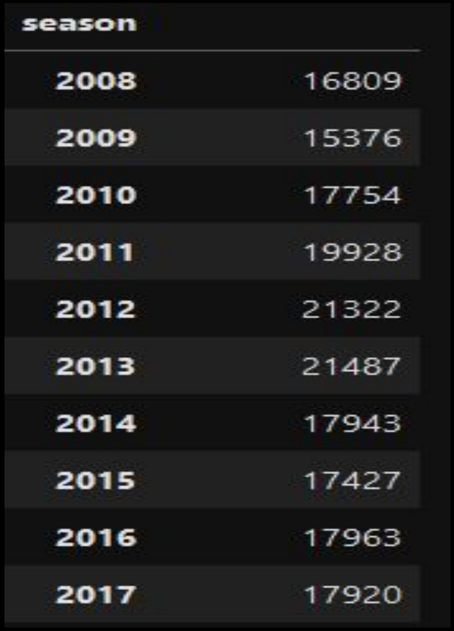




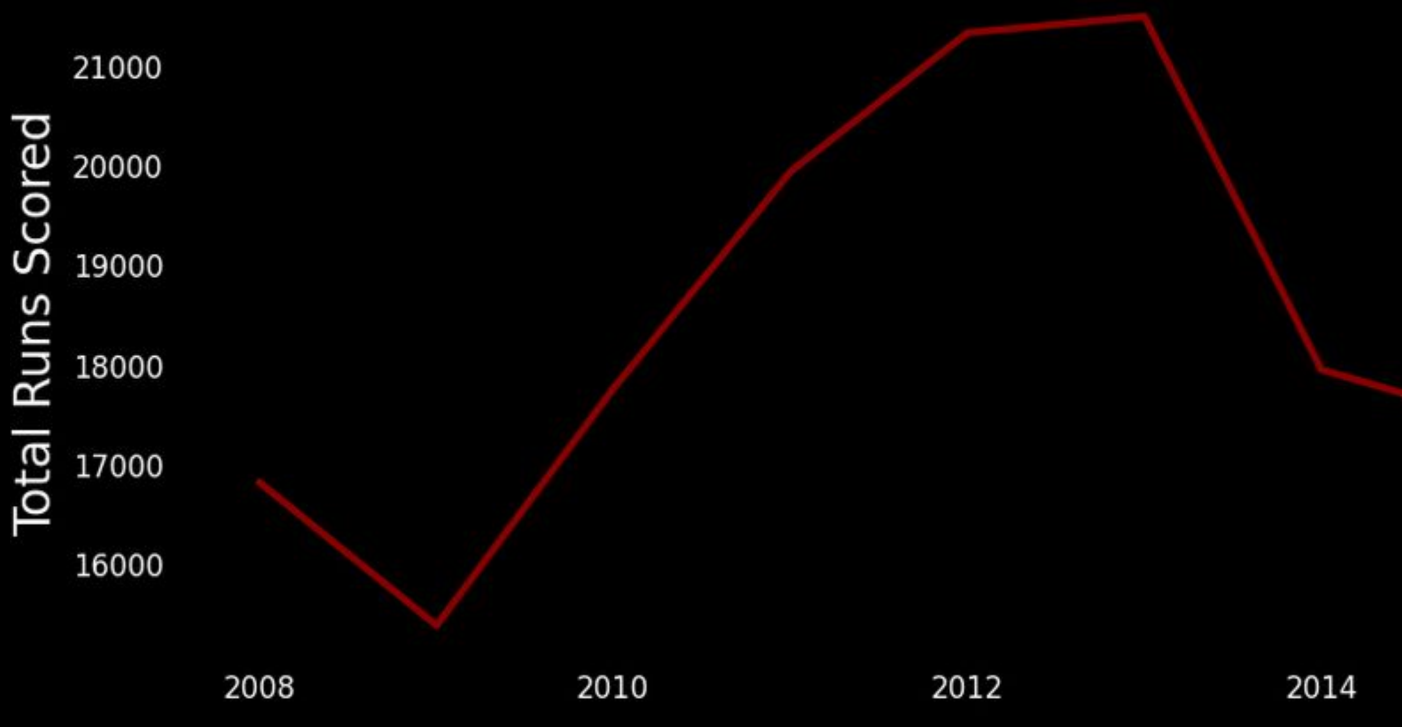
RUNS OVER THE YEARS



```
merge_df.groupby('season')['batsman_runs'].sum()
```

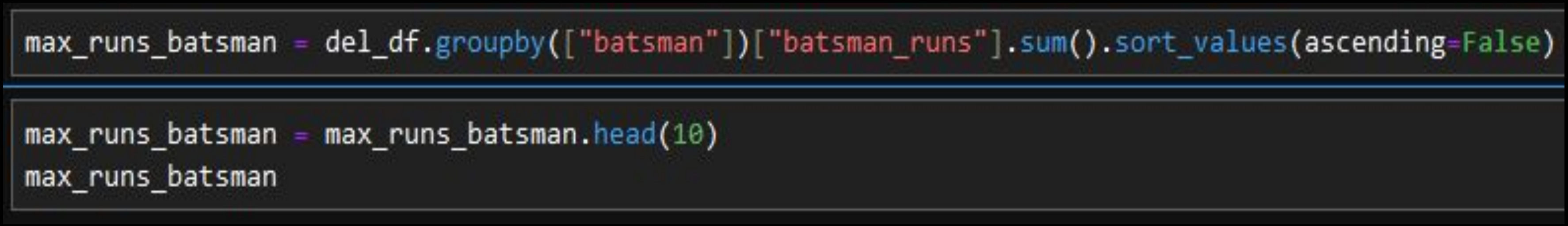


season	
2008	16809
2009	15376
2010	17754
2011	19928
2012	21322
2013	21487
2014	17943
2015	17427
2016	17963
2017	17920



- There was a decline in total runs from 2008 to 2009. But there after there was a substantial increase in runs in every season until 2013, but from next season there was a slump in the total runs. But the number of matches are not equal in all seasons. We should check the average runs per match in each season:

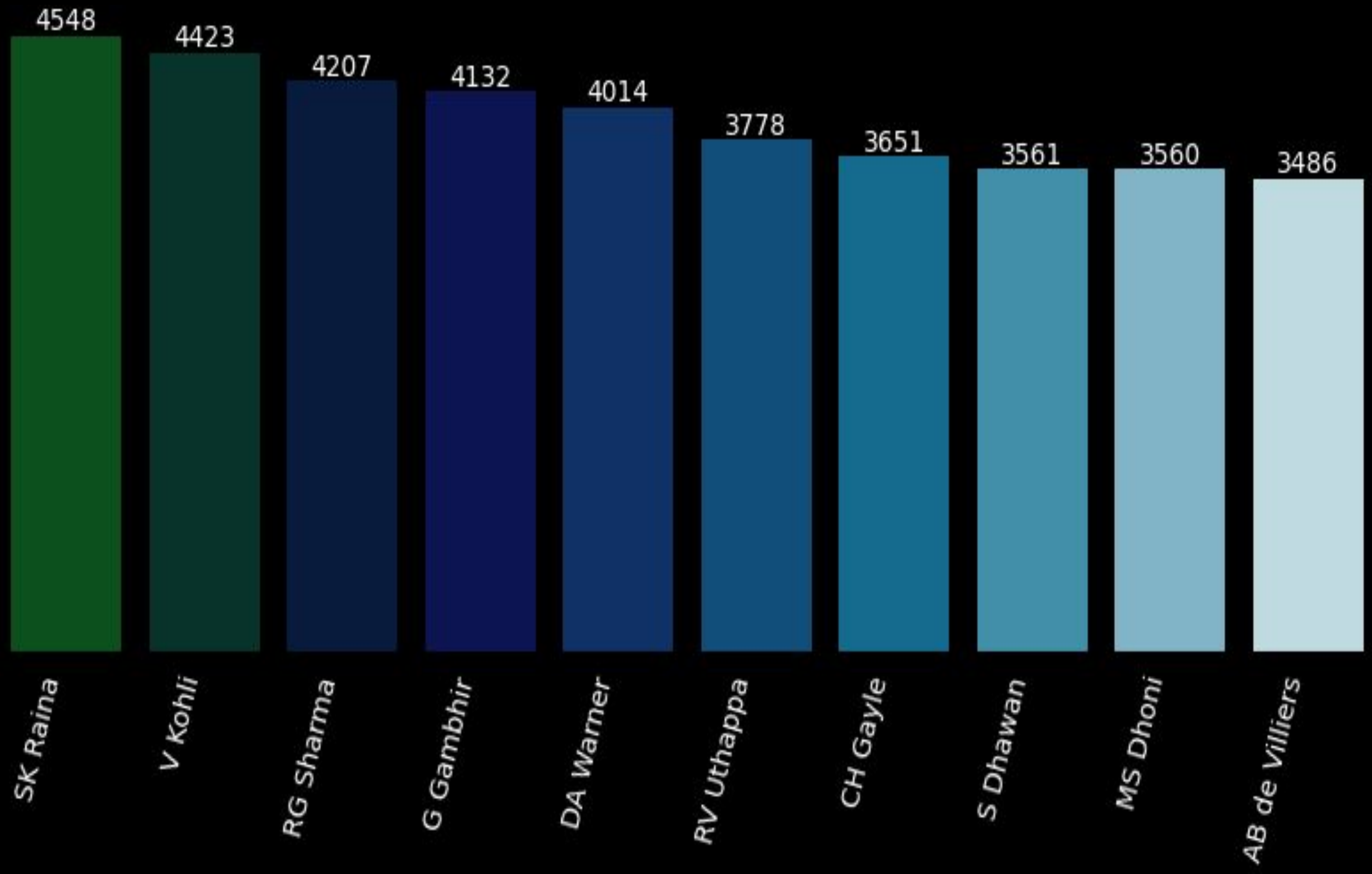
BATSMAN WITH THE MOST RUNS



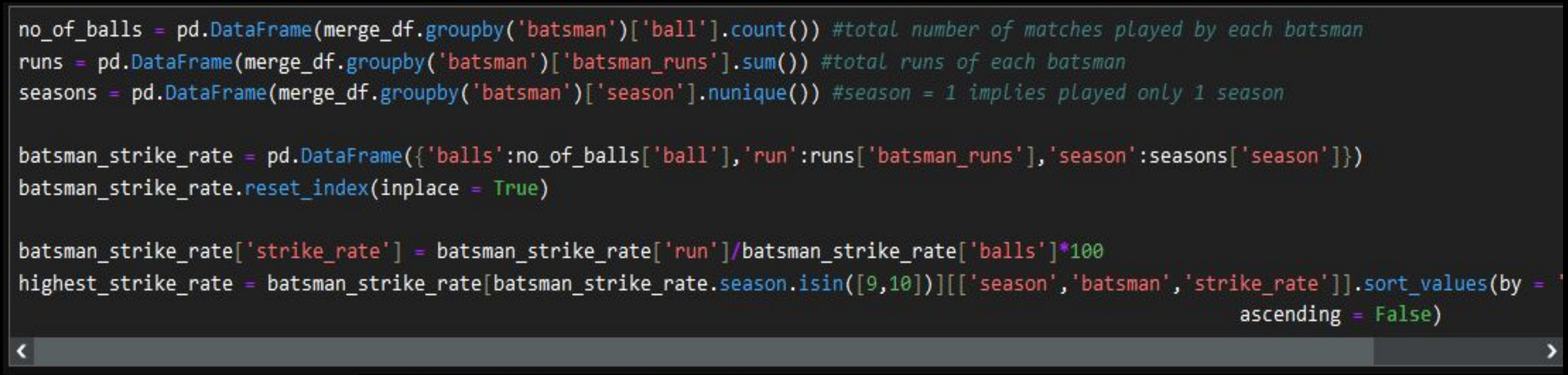
```
max_runs_batsman = del_df.groupby(["batsman"])["batsman_runs"].sum().sort_values(ascending=False)
```

```
max_runs_batsman = max_runs_batsman.head(10)
max_runs_batsman
```

- **Suresh Raina** is the highest run getter in IPL followed by **Virat Kholi**

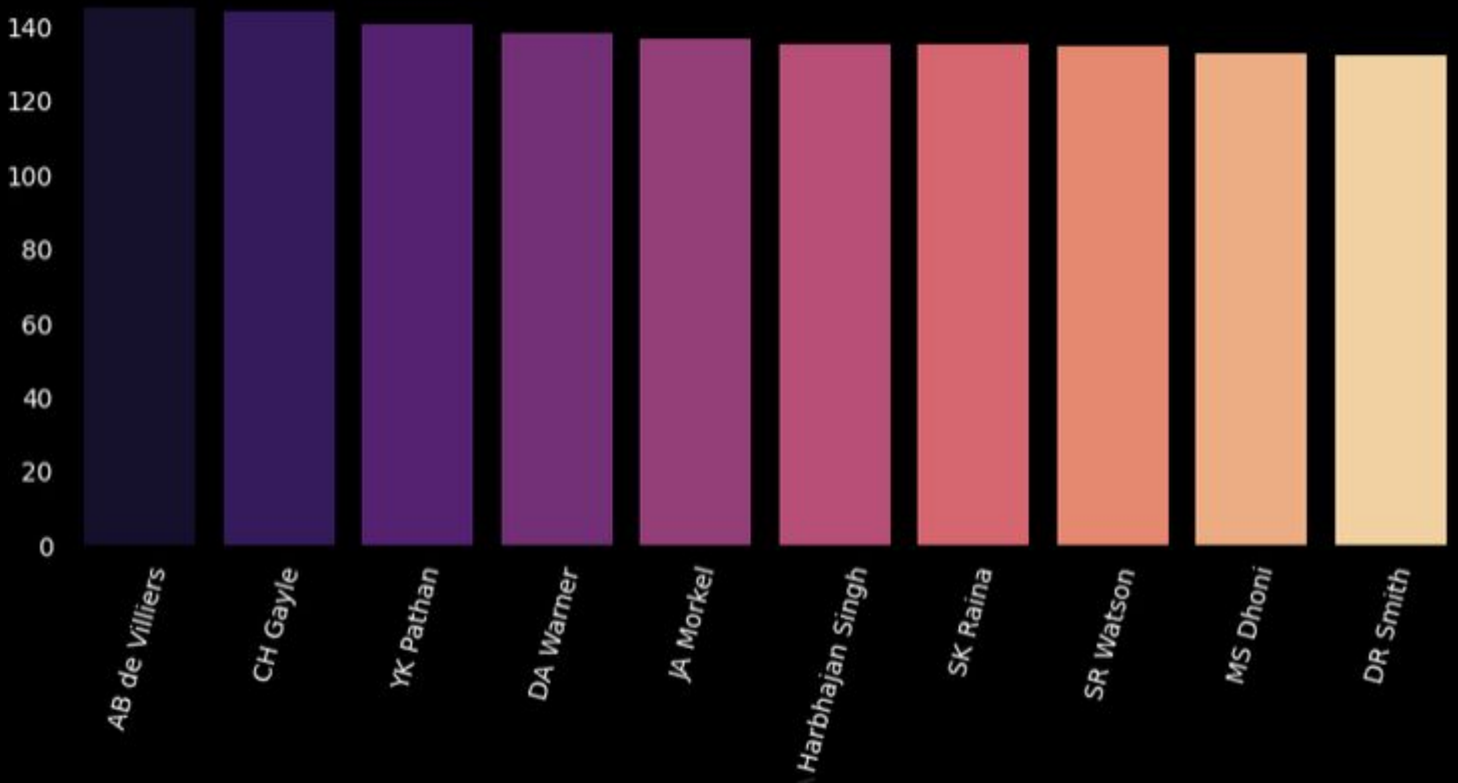


HIGHEST STRIKE RATES BY BATTER

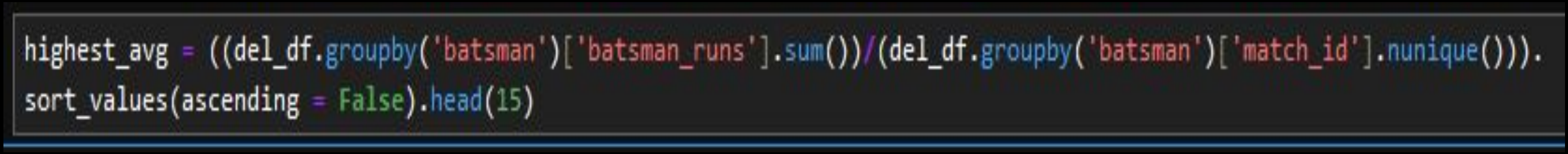


```
no_of_balls = pd.DataFrame(merge_df.groupby('batsman')['ball'].count()) #total number of matches played by each batsman
runs = pd.DataFrame(merge_df.groupby('batsman')['batsman_runs'].sum()) #total runs of each batsman
seasons = pd.DataFrame(merge_df.groupby('batsman')['season'].nunique()) #season = 1 implies played only 1 season

batsman_strike_rate = pd.DataFrame({'balls':no_of_balls['ball'],'run':runs['batsman_runs'],'season':seasons['season']})
batsman_strike_rate.reset_index(inplace = True)

batsman_strike_rate['strike_rate'] = batsman_strike_rate['run']/batsman_strike_rate['balls']*100
highest_strike_rate = batsman_strike_rate[batsman_strike_rate.season.isin([9,10])][['season','batsman','strike_rate']].sort_values(by = '
                                                                                                                                  ascending = False)
```

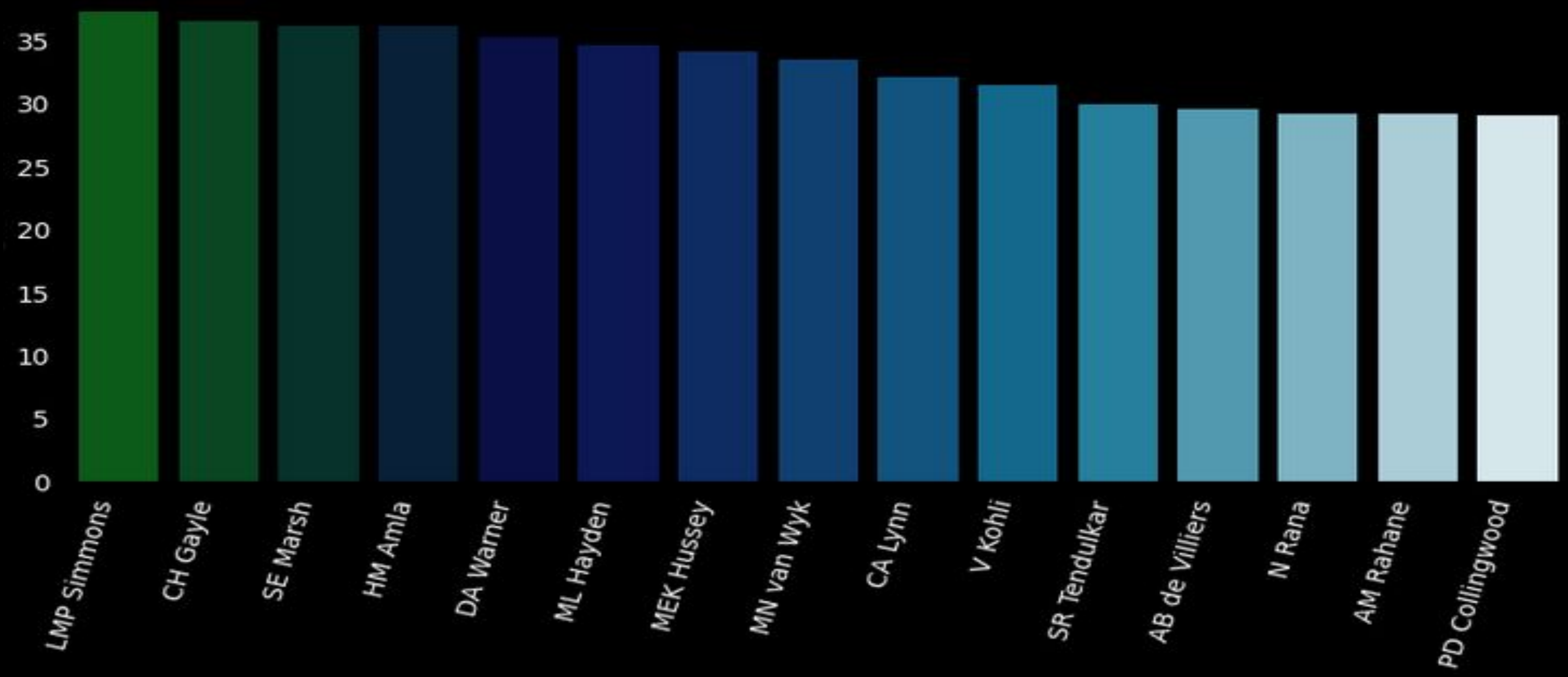
- **AB de Villiers**, **Gayle** have the highest strike rates in IPL. They are the big hitters and can win any match on their day
- One surprise here is that **Harbhajan Singh** who is a bowler has a strike rate of 130+ and comes before Sk Raina in raking



HIGHEST BATTING AVERAGES IN IPL

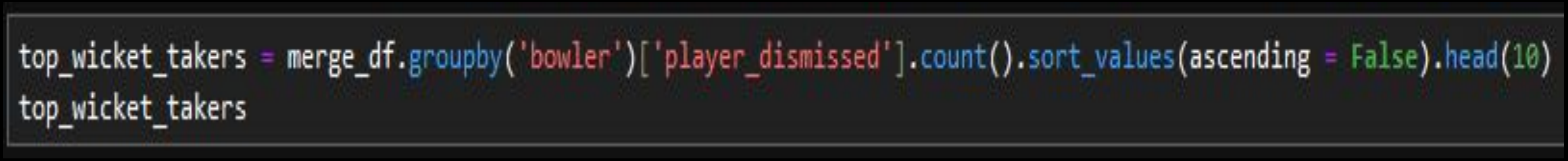


```
highest_avg = ((del_df.groupby('batsman')['batsman_runs'].sum())/(del_df.groupby('batsman')['match_id'].nunique())).
sort_values(ascending = False).head(15)
```



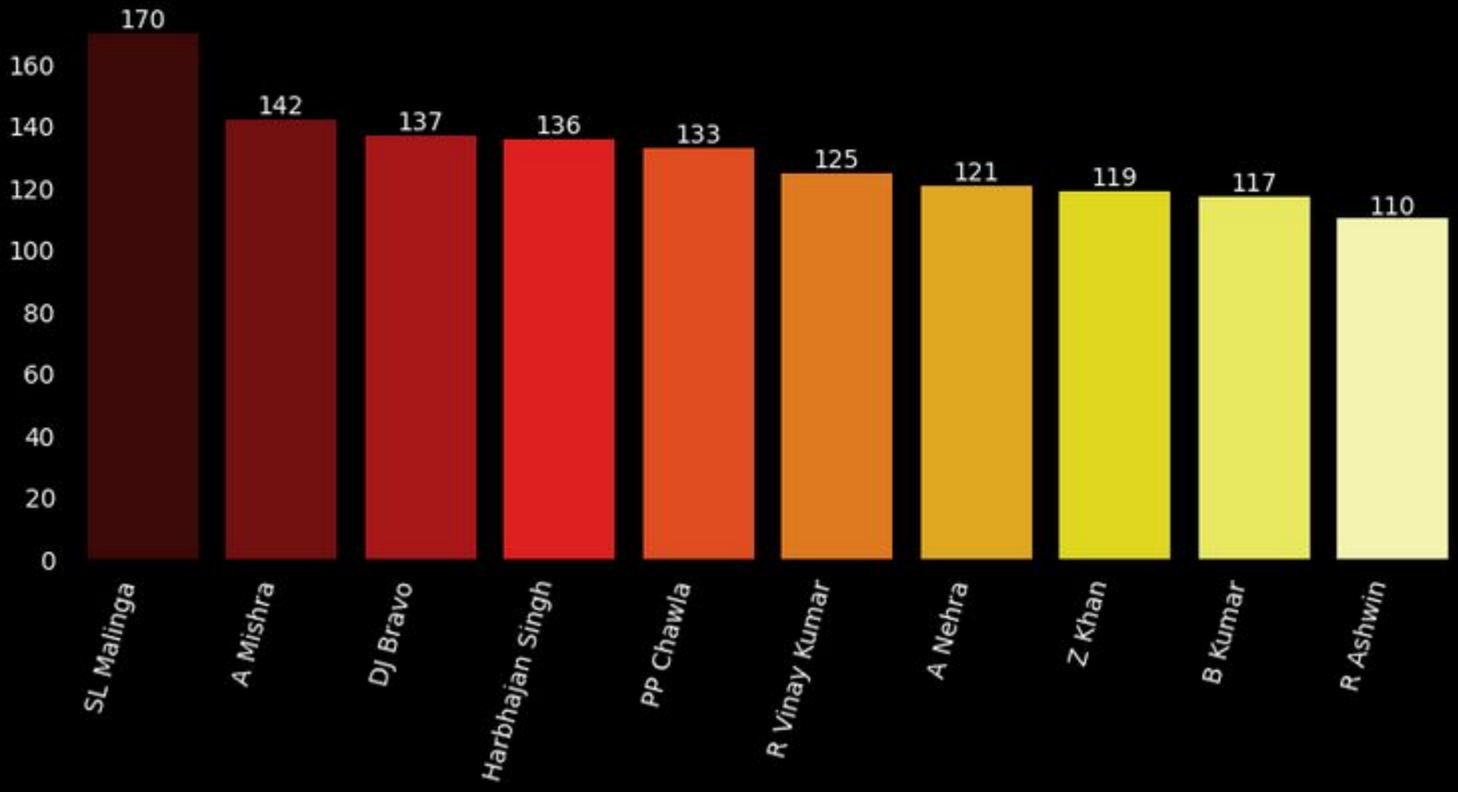
- **LMP Simmons** has the highest average followed by **CH Gayle** and **SE Marsh**
- Rahane, Warner and Hayden might not be in the top 10 run getters but have maintained a good average over the years.

MOST WICKETS TAKERS

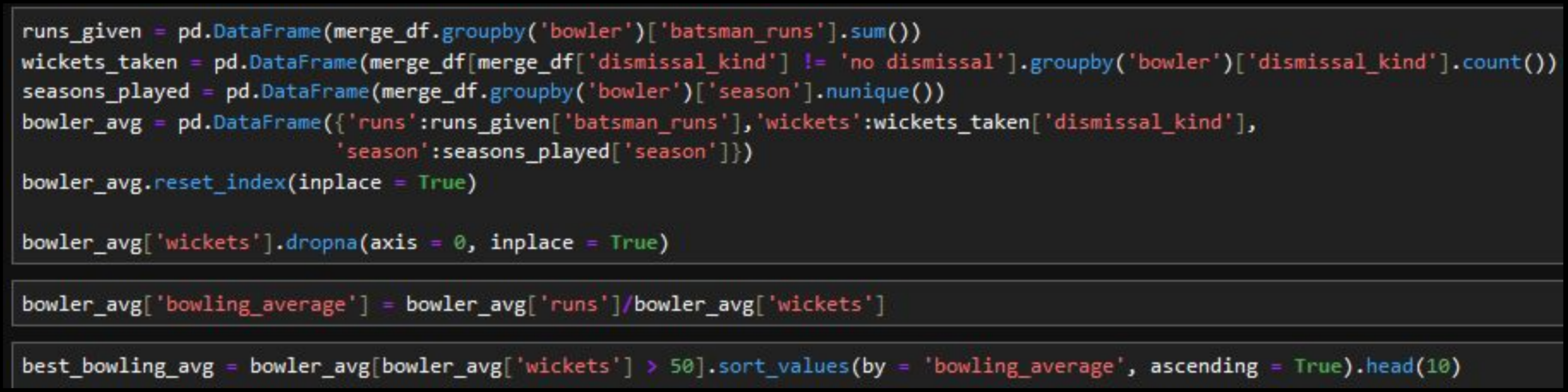


```
top_wicket_takers = merge_df.groupby('bowler')['player_dismissed'].count().sort_values(ascending = False).head(10)
top_wicket_takers
```

- **Malinga** has taken the most number of wickets in IPL followed by Amit Mishra and Bravo



BEST BOWLING AVERAGES

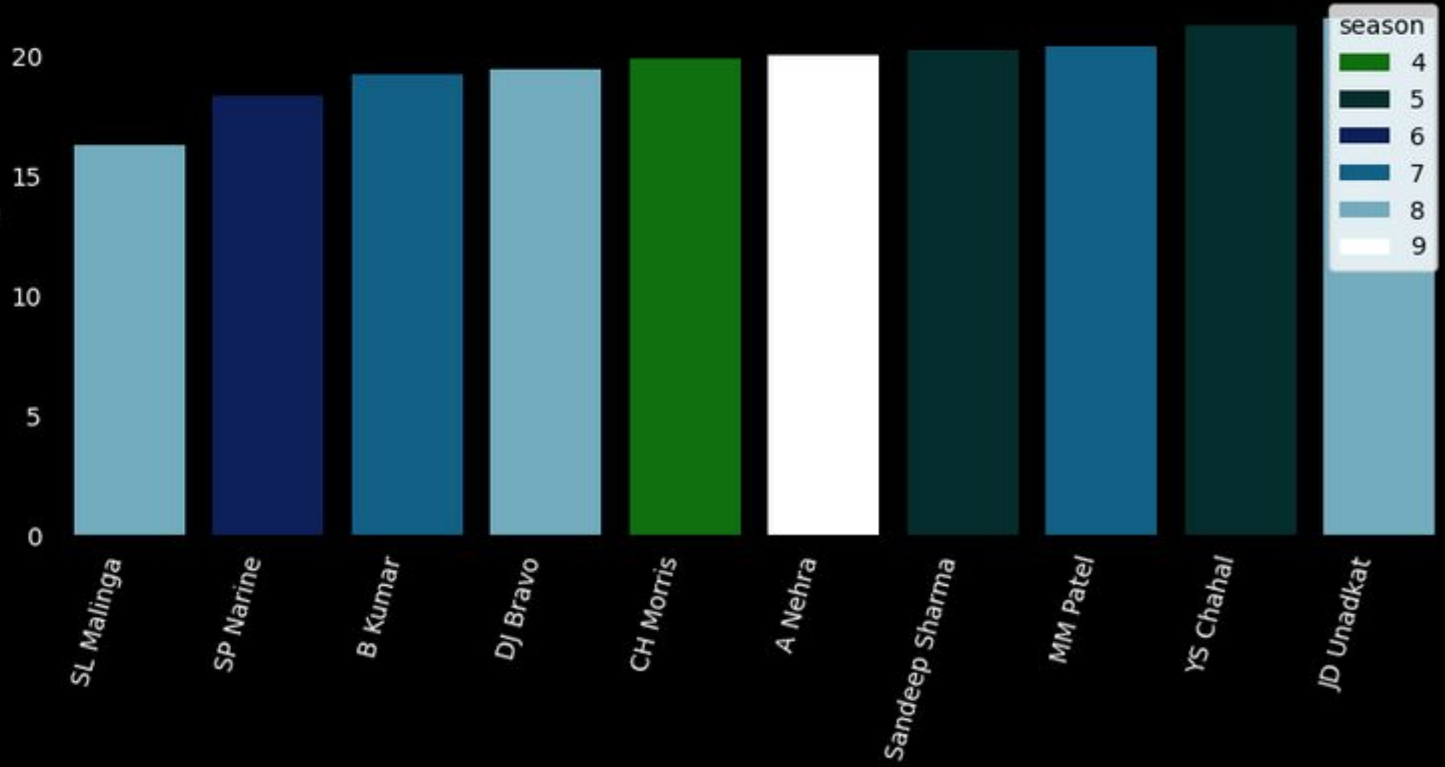


```
runs_given = pd.DataFrame(merge_df.groupby('bowler')['batsman_runs'].sum())
wickets_taken = pd.DataFrame(merge_df[merge_df['dismissal_kind'] != 'no dismissal'].groupby('bowler')['dismissal_kind'].count())
seasons_played = pd.DataFrame(merge_df.groupby('bowler')['season'].nunique())
bowler_avg = pd.DataFrame({'runs':runs_given['batsman_runs'],'wickets':wickets_taken['dismissal_kind'],
                           'season':seasons_played['season']})
bowler_avg.reset_index(inplace = True)

bowler_avg['wickets'].dropna(axis = 0, inplace = True)
```

```
bowler_avg['bowling_average'] = bowler_avg['runs']/bowler_avg['wickets']
```

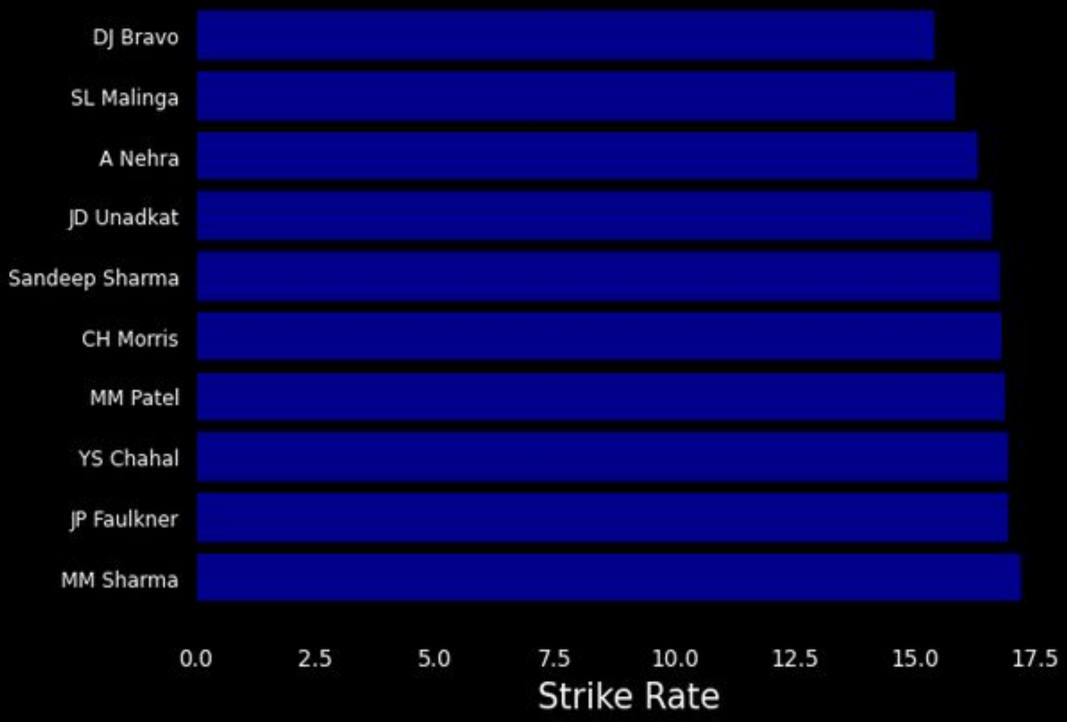
```
best_bowling_avg = bowler_avg[bowler_avg['wickets'] > 50].sort_values(by = 'bowling_average', ascending = True).head(10)
```

- Malinga has taken the most wickets at the best avg of 16.27.
- Malinga, Bravo, Nehra have played 8 and above seasons and hence have taken more wickets and improved average

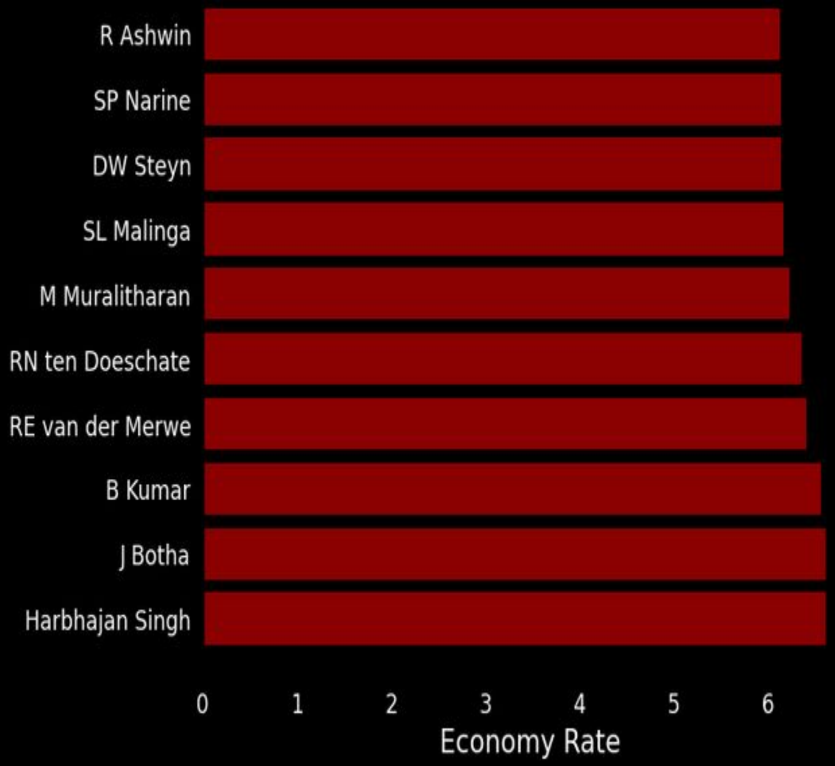


BOWLER

STRIKE RATE

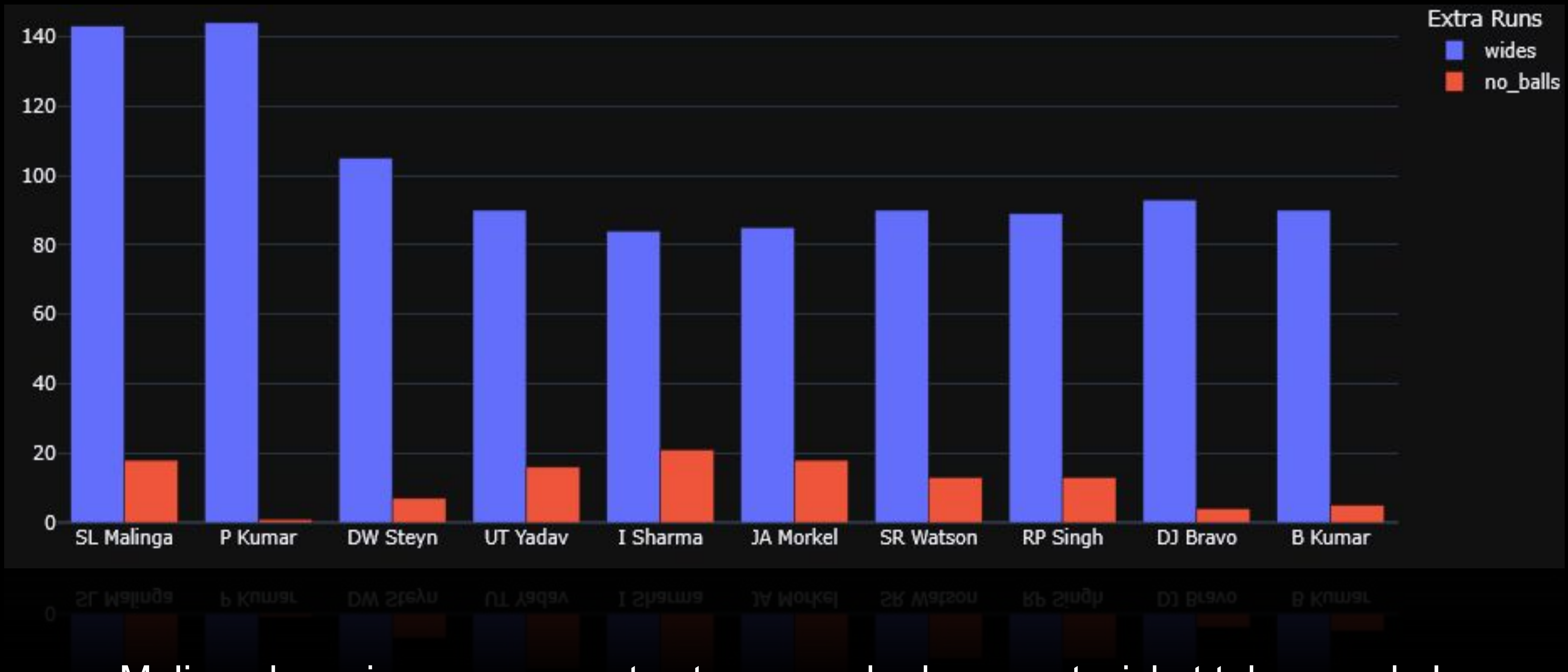


ECONOMY RATE



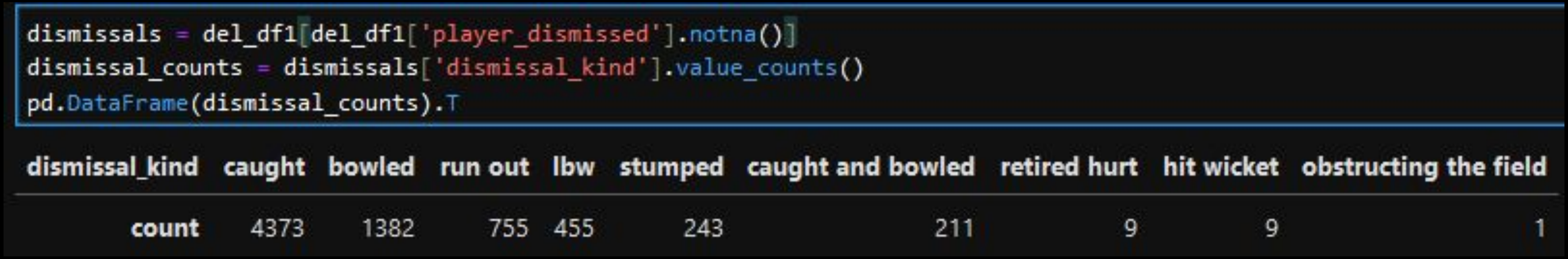
- SL Malinga appears on both lists, indicating his effectiveness both in taking wickets frequently and in controlling the run rate. This makes him an exceptionally valuable bowler.
- Bowlers like DJ Bravo and A Nehra are more focused on taking wickets frequently (strike rate)
- Bowlers like R Ashwin and SP Narine are more focused on restricting runs (economy rate)

EXTRAS RUN GIVEN BY BOWLER



- Malinga has given away most extras even he has most wicket takers and also good economy and strike rate
- Top 10 in this list are all pacers

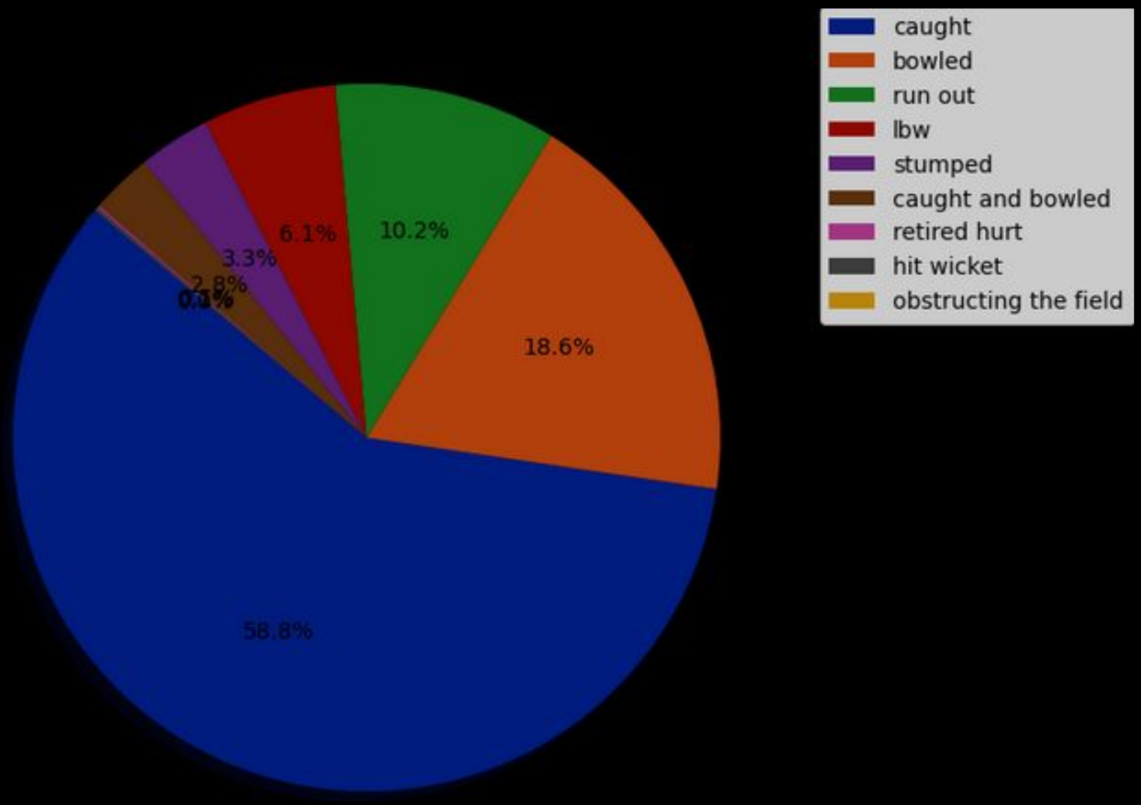
DISTRIBUTION OF DISMISSAL TYPES



```python
dismissals = del_df1[del_df1['player_dismissed'].notna()]
dismissal_counts = dismissals['dismissal_kind'].value_counts()
pd.DataFrame(dismissal_counts).T
```

dismissal_kind	caught	bowled	run out	lbw	stumped	caught and bowled	retired hurt	hit wicket	obstructing the field
count	4373	1382	755	455	243	211	9	9	1

- The high percentages of 'caught,' 'bowled,' and 'lbw' dismissals suggest that bowlers are either inducing batsmen to make errors or are bowling with precision
- The presence of run-out dismissals highlights the importance of good fielding and careful running between the wickets.
- The distribution shows that while batsmen are often aggressive (leading to being caught)



CONCLUSION

- Most teams decide to chase down totals after winning toss except for CSK which goes well with the fact that the have won most games (by good margin) by defending
- 54% times teams who have chased irrespective of winning or losing toss have won matches. But teams winning tosses and electing to field first have won most number of times. It has been uniform across all venues. Particularly KKR, KIX Punjab, DD and RCB have won by big wicket margin
- MoM awards have mostly been received by batsmen implying t20 is a more batsmen-oriented game.
- Suresh Raina has been most consistent batsmen among top run getters while AB has had the highest strike rate among all players who have played 10 or more seasons
- Gayle has had the best average of 36.4 among all batsmen with more than 9 seasons. He has been sensational with most number of Man of the Match awards
- Rahane, Watson are not in top 10 run getters but have maintained a good average across the seasons
- Malinga has been the most impressive bowler in IPL with more than 170 wickets at an average of 16.27, economy of 6.16 and strike rate of 15.84
- Spinners generally do not give away many extra runs and all bowlers in top 10 of that list are pacers.

THANKS