IPL DATA ANALYSIS

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
```

In [2]:
```
df = pd.read_csv('IPL Matches 2008-2020.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	eliminator	method	umpires	umpire1	umpire2
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Royal Challengers Bangalore	runs	140	N	NaN	Asad Rauf	RE Koertzen	
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Chennai Super Kings	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33	N	NaN	MR Benson	SL Shamshi	
2	335984	Delhi	2008-04-19	SPD Smith	Feroz Shah Kotla	0	Rajasthan Royals	Delhi Daredevils	Rajasthan Royals	bat	Delhi Daredevils	runs	59	N	NaN	DJ Harper	PG Pathak	
3	335985	Mumbai	2008-04-20	V Sehwag	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Royal Challengers Bangalore	bat	Mumbai Indians	wickets	5	N	NaN	SJ Davis	DJ Harper	
4	335986	Kolkata	2008-04-20	BJ Hodge	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5	N	NaN	AM Saheba	GAV Baxter	

In [4]:
```
df1 = pd.read_csv('IPL Ball-by-Ball 2008-2020.csv')
```

In [5]:
```
df1.head()
```

Out[5]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_kind	player_dismissed	fielder	extra_type	batting_team	bowling_team
0	335982	1	6	2	SC Ganguly	BB McCullum	P Kumar	1	0	1	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
1	335982	1	6	3	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
2	335982	1	6	4	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
3	335982	1	6	5	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
4	335982	1	7	1	SC Ganguly	BB McCullum	P Kumar	1	0	1	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore

In [6]:
```
df1.columns
```

Out[6]:
```
Index(['id', 'inning', 'over', 'ball', 'batsman', 'non_striker',
       'bowler',
       'batsman_runs', 'extra_runs', 'total_runs', 'non_boundary',
       'is_wicket',
       'dismissal_kind', 'player_dismissed', 'fielder', 'extra_type',
       'batting_team', 'bowling_team',
       dtype='object')
```

In [7]:
```
df1.shape
df = df1
df.rename(columns={'id':'match_id'},inplace=True)
df.head()
```

Out[7]:

	match_id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_kind	player_dismissed	fielder	extra_type	batting_team	bowling_team
0	335982	1	6	2	SC Ganguly	BB McCullum	P Kumar	1	0	1	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
1	335982	1	6	3	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
2	335982	1	6	4	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
3	335982	1	6	5	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
4	335982	1	7	1	SC Ganguly	BB McCullum	P Kumar	1	0	1	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore

In [8]:
```
df['total_runs'] = df['total_runs']
```

In [9]:
```
df.head()
```

Out[9]:

	match_id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_kind	player_dismissed	fielder	extra_type	batting_team	bowling_team
0	335982	1	6	2	SC Ganguly	BB McCullum	P Kumar	1	0	1	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
1	335982	1	6	3	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
2	335982	1	6	4	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
3	335982	1	6	5	BB McCullum	SC Ganguly	P Kumar	0	0	0	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore
4	335982	1	7	1	SC Ganguly	BB McCullum	P Kumar	1	0	1	0	0	NaN	NaN	NaN	NaN	Kolkata Knight Riders	Royal Challengers Bangalore

Exploring data and Data Preprocessing

In [11]:
```
df.shape
```

Out[11]:
```
(215, 18)
```

In [12]:
```
df.info()
```

Out[12]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 215
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           215 non-null    int64
 1   city         210 non-null    object
 2   date         215 non-null    object
 3   player_of_match  215 non-null    object
 4   venue        215 non-null    object
 5   neutral_venue  215 non-null    int64
 6   team1        215 non-null    object
 7   team2        215 non-null    object
 8   toss_winner  215 non-null    object
 9   toss_decision  215 non-null    object
 10  winner       212 non-null    object
 11  result       210 non-null    object
 12  result_margin  207 non-null    float64
 13  eliminator   215 non-null    object
 14  method       19 non-null     object
 15  umpire1      215 non-null    object
 16  umpire2      215 non-null    object
 17  total_runs   215 non-null    int64
dtypes: float64(1), int64(4), object(13)
memory usage: 32.5 KB
```

In [13]:
```
df.columns
```

Out[13]:
```
Index(['id', 'city', 'date', 'player_of_match', 'venue',
       'neutral_venue',
       'team1', 'team2', 'toss_winner', 'toss_decision',
       'winner', 'result',
       'result_margin', 'eliminator', 'method', 'umpire1', 'umpire2',
       'total_runs',
       dtype='object')
```

In [14]:
```
df.describe()
```

Out[14]:

	id	neutral_venue	result_margin	total_runs
count	2.150000e+02	215.000000	207.000000	215.000000
mean	1.304065e+06	0.046512	17.231884	165.790698
std	3.454862e+05	0.211081	21.559207	34.749767
min	3.335982e+05	0.000000	1.000000	36.000000
25%	1.216047e+06	0.000000	5.000000	279.000000
50%	1.304116e+06	0.000000	8.000000	314.000000
75%	1.312081e+06	0.000000	19.500000	325.000000
max	1.370353e+06	1.000000	146.000000	349.000000

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
id                 0
city               5
date               0
player_of_match    0
venue              0
neutral_venue      0
team1              0
team2              0
toss_winner        0
toss_decision      0
winner             3
result             5
result_margin      8
eliminator         0
method           196
umpire1            0
umpire2            0
total_runs         0
dtype: int64
```

In [16]:
```
df = df.dropna()
```

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
id                 0
city               0
date               0
player_of_match    0
venue              0
neutral_venue      0
team1              0
team2              0
toss_winner        0
toss_decision      0
winner             0
result             0
result_margin      0
eliminator         0
method             0
umpire1            0
umpire2            0
total_runs         0
dtype: int64
```

Analysing the data

1. What are the count of match played in each season

In [18]:
```
year = []
for i in df['date']:
    year.append(int(i[0:4]))
df['season'] = year
matches_per_season = df.groupby('season').size()
print(matches_per_season)
```

Out[18]:
```
Season
2008    58
2009    57
2010    60
2011    73
2012    74
2013    76
2014    60
2015    59
2016    60
2017    59
2018    60
2019    60
2020    60
dtype: int64
```

2. How many runs were scored in each season

In [19]:
```
Runs_Scored = df.groupby('Season')['total_runs'].sum()
print(Runs_Scored)
```

Out[19]:
```
Season
2008    17937
2009    16222
2010    18592
2011    21154
2012    22453
2013    21977
2014    18909
2015    17921
2016    18862
2017    18699
2018    19821
2019    18655
2020    17958
Name: total_runs, dtype: int64
```

3. What were the runs scored per match in different seasons

In [20]:
```
season_matches = []
for i in df['Season'].unique():
    a = df.shape[0]/df['Season'].sum()
    df['a'] = df.a/df['total_runs']

    a = 0
    for i in df.a.values:
        season_matches.append(round(i,2))
        break
df = pd.DataFrame(season_matches, columns = ['Season', 'Match no.', 'Total runs'])
df.T
```

Out[20]:

	Season	Match no.	Total runs
0	2008	0	2008
1	2009	0	2009
2	2010	0	2010
3	2011	0	2011
4	2012	0	2012
...	...	...	...
196	2020	52	2020
197	2020	52	2020
198	2020	52	2020
199	2020	52	2020
200	2020	52	2020

200 rows × 3 columns

4. Who has Umpired the Most

In [24]:
```
print('Most number of Umpire in Umpire 1')
umpire1 = df
for i in df['umpire1']:umpire1
    umpire1.append(i,i,i,i,i,i,i,i,i,i,i,i,i)
df_u1 = pd.DataFrame(umpire1, columns = ['Name', 'Freq_u1'])
df_u1 = df_u1.sort_values(by = 'Name', ignore_index = True)
df_u1.iloc[:5]
```

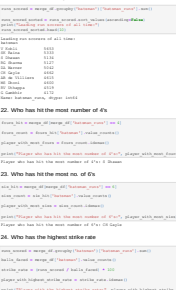
```
Most number of Umpire in Umpire 1
Name
0    NDTV Cameraman
1    Nitin Menon
Name: 1.1, dtype: object
```

In [25]:
```
print('Most number of Umpire in Umpire 2')
umpire2 = df
for i in df['umpire2']:umpire2
    umpire2.append(i,i,i,i,i,i,i,i,i,i,i,i,i)
df_u2 = pd.DataFrame(umpire2, columns = ['Name', 'Freq'])
df_u2 = df_u2.sort_values(by = 'Name')
df_u2.iloc[:5]
```

Out[25]:
```
Most number of Umpire in Umpire 2
Name
Gaffaney    1
Name: 1.1, dtype: object
```

5. Which Team Was one the most Tosses

In [26]:
```
toss_winner = df.groupby('toss_winner').size()
toss_winner.plot(kind = 'bar')
plt.ylabel('No. of Winner')
plt.title('Finding out the Team which won MAX Tosses')
plt.show()
```



According to the Above Visualization it is clear that 'Mumbai Indians' is the team which won Tosses maximum time

6. What does the team decide after winning the toss

In [29]:
```
toss_decision = df.groupby('toss_decision').size()
toss_decision.plot(kind = 'bar', color = 'g')
plt.ylabel('No. of times')
plt.title('Comparison between toss decision')
```

Out[29]:
```
Text(0.5, 1.0, 'Comparison between toss decision')
```


Max number of times the Toss winning team selects to FIELD

7. How does the toss decision vary across the seasion

In [30]:
```
toss_decision_counts = df.groupby(['Season', 'toss_decision']).size().reset_index(name='count')

toss_decision_pivot = toss_decision_counts.pivot(index='Season', columns='toss_decision', values='count')

toss_decision_pivot['total'] = toss_decision_pivot.sum(axis=1)

toss_decision_pivot['batting_percentage'] = (toss_decision_pivot['bat'] / toss_decision_pivot['total']) * 100
toss_decision_pivot['fielding_percentage'] = (toss_decision_pivot['field'] / toss_decision_pivot['total']) * 100

# Display the updated DataFrame
toss_decision_pivot
```

Out[30]:

toss_decision

Season	bat	field	total	batting_percentage	fielding_percentage
2008	25	33	58	43.103448	56.896552
2009	25	32	57	43.859649	56.140351
2010	25	35	60	41.666667	58.333333
2011	22	51	73	30.136986	69.863014
2012	27	47	74	36.486486	63.513514
2013	22	53	75	29.333333	70.666667
2014	22	38	60	36.666667	63.333333
2015	25	34	59	42.372881	57.627119
2016	24	36	60	40.000000	60.000000
2017	22	36	58	37.931034	62.068966
2018	19	41	60	31.666667	68.333333
2019	16	44	60	26.666667	73.333333
2020	15	45	60	25.000000	75.000000

8. Does winning the toss imply winning the game

In [31]:
```
toss_win_match_result = df[df['toss_winner'] == df['winner']]['winner'].value_counts()
total_toss_match_results = df[df['toss_winner'] == df['winner']]['winner'].value_counts()

total_matches_won = toss_win_match_result.sum()
total_matches_lost = toss_win_match_result.sum()

win_percentage = (total_matches_won / (total_matches_won + total_matches_lost)) * 100
lost_percentage = (total_matches_lost / (total_matches_won + total_matches_lost)) * 100
print("Percentage of matches won after winning the toss: {:.2f}%".format(win_percentage))
print("Percentage of matches lost after winning the toss: {:.2f}%".format(lost_percentage))
```

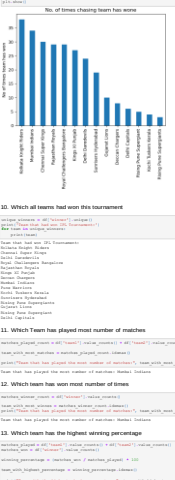
```
Percentage of matches won after winning the toss: 51.81%
Percentage of matches lost after winning the toss: 48.19%
```

9. How many time has the chasing team won the match

In [32]:
```
df_tw = df[df['toss_decision'] == 'field']

winner = df.loc[df['toss_winner'] == df['winner']]['winner'].value_counts()
winner.plot(kind = 'bar')

plt.ylabel('No. of times team has won')
plt.title('No. of times chasing team has won')
plt.show()
```



10. Which all teams had won this tournament

In [33]:
```
unique_winners = df['winner'].unique()
print("Teams that won IPL Tournament:")
for team in unique_winners:
    print(team)
```

Out[33]:
```
Teams that won IPL Tournament:
Kolkata Knight Riders
Royal Challengers
Delhi Daredevils
Royal Challengers Bangalore
Kings XI Punjab
Rajasthan Royals
Chennai Super Kings
Deccan Chargers
Mumbai Indians
Pune Warriors
Kochi Tuskers Kerala
Sunrisers Hyderabad
Rising Pune Supergiants
Gujarat Lions
Rising Pune Supergiant
Delhi Capitals
```

11. Which Team has played most number of matches

In [34]:
```
matches_played_count = df['team1'].value_counts() + df['team2'].value_counts()

team_with_most_matches = matches_played_count.idxmax()

print("Team that has played the most number of matches:", team_with_most_matches)
```

Out[34]:
```
Team that has played the most number of matches: Mumbai Indians
```

12. Which team has won most number of times

In [35]:
```
matches_winner = df['winner'].value_counts()

team_with_most_wins = matches_winner.idxmax()
print("Team that has played the most number of matches:", team_with_most_wins)
```

Out[35]:
```
Team that has played the most number of matches: Mumbai Indians
```

13. Which team has the highest winning percentage

In [37]:
```
matches_played = df['team1'].value_counts() + df['team2'].value_counts()
matches_won = df['winner'].value_counts()

winning_percentage = (matches_won / matches_played) * 100

team_with_highest_percentage = winning_percentage.idxmax()

print("Team with the highest winning percentage:", team_with_highest_percentage)
```

```
Team with the highest winning percentage: Rising Pune Supergiant
```

14. Is there any lucky venue for a particular team

In [38]:
```
for i in matches_played.index:

    team_name = i
    team_matches = df[(df['team1'] == team_name) | (df['team2'] == team_name)]

    venue_wins = team_matches[team_matches['winner'] == team_name]['venue'].value_counts()

    if len(venue_wins.empty):
        pass
        max_wins = venue_wins.max()
        lucky_venue = venue_wins[venue_wins == max_wins]
        total_matches = matches_played[team_name]
        for venue in lucky_venue:
            print(venue)
        print("-"*40)
    else:
        print("No data available for the team", team_name)
```

Out[38]:
```
Lucky venue for Chennai Super Kings
MA Chidambaram Stadium, Chepauk
----------------------------------------
Lucky venue for Deccan Chargers
Rajiv Gandhi International Stadium, Uppal
----------------------------------------
Lucky venue for Delhi Capitals
Feroz Shah Kotla
Sharjah Cricket Stadium
----------------------------------------
Lucky venue for Delhi Daredevils
Feroz Shah Kotla
----------------------------------------
Lucky venue for Gujarat Lions
Saurashtra Cricket Association Stadium
----------------------------------------
Lucky venue for Kings XI Punjab
Punjab Cricket Association Stadium, Mohali
----------------------------------------
Lucky venue for Kochi Tuskers Kerala
Nehru Stadium
----------------------------------------
Lucky venue for Kolkata Knight Riders
Eden Gardens
----------------------------------------
Lucky venue for Mumbai Indians
Wankhede Stadium
----------------------------------------
Lucky venue for Pune Warriors
Subrata Roy Sahara Stadium
----------------------------------------
Lucky venue for Rajasthan Royals
Sawai Mansingh Stadium
----------------------------------------
Lucky venue for Rising Pune Supergiant
Maharashtra Cricket Association Stadium
----------------------------------------
Lucky venue for Rising Pune Supergiants
Dr. D.Y. Patil Sports Academy DY Patil Stadium
Maharashtra Cricket Association Stadium
----------------------------------------
Lucky venue for Royal Challengers Bangalore
M Chinnaswamy Stadium
----------------------------------------
Lucky venue for Sunrisers Hyderabad
Rajiv Gandhi International Stadium, Uppal
----------------------------------------
```

15. Innings wise comparison between teams

In [39]:
```
merge_df = pd.merge(df1,df[['id','team1','team2']],
 inning0_name = merge_df.groupby(['inning','team1'], as_index = False)['total_runs'].sum(),
 inning0_name_pivot = inning0_name.pivot(index='inning', columns='team1', values='total_runs')
 inning0_name_pivot.T.sort_values(by=0,ascending=False)
 inning0_name_pivot
```

Out[39]:
Inning=Value comparison between teams:

team1	Chennai Super Kings	Deccan Chargers	Delhi Capitals	Gujarat Lions	Kochi Tuskers Kerala	Kolkata Knight Riders	Mumbai Indians	Pune Warriors	Rajasthan Royals	Rising Pune Supergiant	Rising Pune Supergiants	Royal Challengers Bangalore	Sunrisers Hyderabad
inning													
1	14768	9890	2988	5879	2246	15855	16220	3195	13471	1138	4176	16771	8997
2	13705	9896	2988	5879	2246	15855	16220	3195	13471	1138	4176	15202	7921

16. Which team has scored most number of 200+ score

In [40]:
```
high_scores = merge_df[merge_df['total_runs'] >= 200]
team_high_scores = high_scores['team1'].value_counts() + high_scores['team2'].value_counts()
team_with_most_high_scores = team_high_scores.idxmax()
print("Team with the most number of 200+ scores:", team_with_most_high_scores)
```

Out[40]:
```
Team with the most number of 200+ scores: Mumbai Indians
```

17. Which team has conceded 200+ score the most

In [41]:
```
high_scores = merge_df[merge_df['total_runs'] >= 200]

team_high_scores_conceded = high_scores.groupby('bowling_team').size()

team_with_most_high_scores_conceded = team_high_scores_conceded.idxmax()

print("Team that has conceded the most number of 200+ scores:", team_with_most_high_scores_conceded)
```

Out[41]:
```
Team that has conceded the most number of 200+ scores: Mumbai Indians
```

In [42]:
```
team_with_most_high_scores_conceded
```

Out[42]:
```
'Mumbai Indians'
```

18. What was the highest run scored by a team in single match

In [43]:
```
highest_runs = merge_df.groupby(['id', 'inning', 'team1'], as_index=False)['total_runs_y'].sum().groupby('id')['total_runs_y'].max()

highest_run_scorer = highest_runs.max()

print("Highest run scored by a team in a single match", highest_run_scorer)
```

Out[43]:
```
Highest run scored by a team in a single match: 240
```

19. Which is the biggest win in terms of run margine

In [44]:
```
match_run_margin = merge_df.groupby('id')['total_runs_y'].max() - merge_df.groupby('id')['total_runs_y'].min(0.)[1:]

biggest_run_margin_match = match_run_margin.idxmax()

biggest_run_margin = match_run_margin[biggest_run_margin_match]
print("Biggest win in terms of run margin")
print("Match ID", biggest_run_margin_match)
print("Run Margin", biggest_run_margin)
```

Out[44]:
```
Biggest win in terms of run margin
Match ID 335987
Run Margin 231.0
```

20. Which batsmen have played most number of balls

In [45]:
```
balls_faced = merge_df['batsman'].value_counts()

batsman_with_most_balls_faced = balls_faced.idxmax()

print("Batsman who has played the most number of balls:", batsman_with_most_balls_faced)
```

Out[45]:
```
Batsman who has played the most number of balls: V Kohli
```

21. Who are the leading run scorers of all time

In [46]:
```
runs_scored = merge_df.groupby('batsman')['batsman_runs'].sum()

runs_scored_sorted = runs_scored.sort_values(ascending=False)
print("Leading run scorers of all time:")
print(runs_scored_sorted.head(10))
```

Out[46]:
```
Leading run scorers of all time:
V Kohli    5053
SK Raina   4548
JC Buttler 3115
RG Sharma  3127
DA Warner  3092
CH Gayle   2942
AB de Villiers 2936
MS Dhoni   4020
KD Karthik 4015
AM Rahane  3127
Name: batsman_runs, dtype: int64
```

22. Who has hit the most number of 4's

In [47]:
```
fours_hit = merge_df[merge_df['batsman_runs'] == 4]

fours_count = fours_hit['batsman'].value_counts()

player_with_most_fours = fours_count.idxmax()

print("Player who has hit the most number of 4's:", player_with_most_fours)
```

Out[47]:
```
Player who has hit the most number of 4's: S Dhawan
```

23. Who has hit the most no. of 6's

In [48]:
```
six_hit = merge_df[merge_df['batsman_runs'] == 6]

sixes_count = six_hit['batsman'].value_counts()

player_with_most_sixes = sixes_count.idxmax()

print("Player who has hit the most number of 6's:", player_with_most_sixes)
```

Out[48]:
```
Player who has hit the most number of 6's: CH Gayle
```

24. Who has the highest strike rate

In [49]:
```
runs_scored = merge_df.groupby('batsman')['batsman_runs'].sum()

balls_faced = merge_df['batsman'].value_counts()

strike_rate = (runs_scored / balls_faced) * 100

player_with_highest_strike_rate = strike_rate.idxmax()

print("Player with the highest strike rate:", player_with_highest_strike_rate)
```

Out[49]:
```
Player with the highest strike rate: S Dwalish
```

25. Who is leading wicket taker

In [50]:
```
wickets_taken = merge_df.groupby('bowler')['is_wicket'].sum()

wickets_count = wickets_taken['bowler'].value_counts()

leading_wicket_taker = wickets_count.idxmax()

print("Leading wicket-taker:", leading_wicket_taker)
```

Out[50]:
```
Leading wicket-taker: SL Malinga
```

26. Which stadium has hosted most no. of matches

In [51]:
```
matches_count = df['venue'].value_counts()

stadium_with_most_matches = matches_count.idxmax()

print("Stadium that has hosted the most number of matches:", stadium_with_most_matches)
```

Out[51]:
```
Stadium that has hosted the most number of matches: Eden Gardens
```

In [ ]: