Major Project 2: Data Analysis And Visualization - IPL

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 Ball-by-Ball 2008-2020.csv')
```

In [3]:
```
df
```

Out[3]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_kind	player_dis
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	NaN	
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	NaN	
2	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	NaN	
3	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	0	0	NaN	
4	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	0	0	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
193463	1237181	1	12	5	RR Pant	SS Iyer	NM Coulter-Nile	0	0	0	0	0	NaN	
193464	1237181	1	12	6	RR Pant	SS Iyer	NM Coulter-Nile	1	0	1	0	0	NaN	
193465	1237181	1	13	1	RR Pant	SS Iyer	KH Pandya	0	1	1	0	0	NaN	
193466	1237181	1	13	2	RR Pant	SS Iyer	KH Pandya	1	0	1	0	0	NaN	
193467	1237181	1	13	3	SS Iyer	RR Pant	KH Pandya	1	0	1	0	0	NaN	

193468 rows × 18 columns

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

In [5]:
```
df1
```

Out[5]:

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner	result	result
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	
4	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
811	1216547	Dubai	2020-09-28	AB de Villiers	Dubai International Cricket Stadium	0	Royal Challengers Bangalore	Mumbai Indians	Mumbai Indians	field	Royal Challengers Bangalore	tie	
812	1237177	Dubai	2020-11-05	JJ Bumrah	Dubai International Cricket Stadium	0	Mumbai Indians	Delhi Capitals	Delhi Capitals	field	Mumbai Indians	runs	
813	1237178	Abu Dhabi	2020-11-06	KS Williamson	Sheikh Zayed Stadium	0	Royal Challengers Bangalore	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	
814	1237180	Abu Dhabi	2020-11-08	MP Stoinis	Sheikh Zayed Stadium	0	Delhi Capitals	Sunrisers Hyderabad	Delhi Capitals	bat	Delhi Capitals	runs	
815	1237181	Dubai	2020-11-10	TA Boult	Dubai International Cricket Stadium	0	Delhi Capitals	Mumbai Indians	Delhi Capitals	bat	Mumbai Indians	wickets	

816 rows × 17 columns

Studing df

In [6]:
```
df
```

Out[6]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_kind	player_dis
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	NaN	
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	NaN	
2	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	NaN	
3	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	0	0	NaN	
4	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	0	0	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
193463	1237181	1	12	5	RR Pant	SS Iyer	NM Coulter-Nile	0	0	0	0	0	NaN	
193464	1237181	1	12	6	RR Pant	SS Iyer	NM Coulter-Nile	1	0	1	0	0	NaN	
193465	1237181	1	13	1	RR Pant	SS Iyer	KH Pandya	0	1	1	0	0	NaN	
193466	1237181	1	13	2	RR Pant	SS Iyer	KH Pandya	1	0	1	0	0	NaN	
193467	1237181	1	13	3	SS Iyer	RR Pant	KH Pandya	1	0	1	0	0	NaN	

193468 rows × 18 columns

```
In [7]: df['id'].isnull().sum()
Out[7]: 0

In [8]: df['inning'].isnull().sum()
Out[8]: 0

In [9]: df['over'].isnull().sum()
Out[9]: 0

In [10]: df['ball'].isnull().sum()
Out[10]: 0

In [11]: df['batsman'].isnull().sum()
Out[11]: 0

In [12]: df['non_striker'].isnull().sum()
Out[12]: 0

In [13]: df['bowler'].isnull().sum()
Out[13]: 0

In [14]: df['batsman_runs'].isnull().sum()
Out[14]: 0

In [15]: df['extra_runs'].isnull().sum()
Out[15]: 0

In [16]: df['total_runs'].isnull().sum()
Out[16]: 0
```

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

Out[17]: 0

In [18]:
```
df['is_wicket'].isnull().sum()
```

Out[18]: 0

In [19]:
```
df['dismissal_kind'].isnull().sum()  #deleted
```

Out[19]: 183973

In [20]:
```
df['player_dismissed'].isnull().sum()  #deleted
```

Out[20]: 183973

In [21]:
```
df['fielder'].isnull().sum()
```

Out[21]: 186684

In [22]:
```
df['extras_type'].isnull().sum()  #deleted
```

Out[22]: 183235

In [23]:
```
df['batting_team'].isnull().sum()
```

Out[23]: 0

In [24]:
```
df['bowling_team'].isnull().sum()
```

Out[24]: 191

In [25]:
```
del df['fielder']
```

In [26]:
```
del df['dismissal_kind']
```

In [27]:
```
del df['player_dismissed']
```

In [28]:
```
del df['extras_type']
```

In [29]:
```
df
```

Out[29]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	batting_team	bowling_tea
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
2	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
3	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
4	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
193463	1237181	1	12	5	RR Pant	SS Iyer	NM Coulter-Nile	0	0	0	0	0	Delhi Capitals	Mumb Indiar
193464	1237181	1	12	6	RR Pant	SS Iyer	NM Coulter-Nile	1	0	1	0	0	Delhi Capitals	Mumb Indiar
193465	1237181	1	13	1	RR Pant	SS Iyer	KH Pandya	0	1	1	0	0	Delhi Capitals	Mumb Indiar
193466	1237181	1	13	2	RR Pant	SS Iyer	KH Pandya	1	0	1	0	0	Delhi Capitals	Mumb Indiar
193467	1237181	1	13	3	SS Iyer	RR Pant	KH Pandya	1	0	1	0	0	Delhi Capitals	Mumb Indiar

193468 rows × 14 columns

In [30]:
```
#Filling NULL values in the bowling_team column
df['bowling_team'] = df['bowling_team'].fillna('Anonymous')
```

In [31]:
```
df
```

Out[31]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	batting_team	bowling_tea
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
2	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
3	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
4	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
193463	1237181	1	12	5	RR Pant	SS Iyer	NM Coulter-Nile	0	0	0	0	0	Delhi Capitals	Mumb Indiar
193464	1237181	1	12	6	RR Pant	SS Iyer	NM Coulter-Nile	1	0	1	0	0	Delhi Capitals	Mumb Indiar
193465	1237181	1	13	1	RR Pant	SS Iyer	KH Pandya	0	1	1	0	0	Delhi Capitals	Mumb Indiar
193466	1237181	1	13	2	RR Pant	SS Iyer	KH Pandya	1	0	1	0	0	Delhi Capitals	Mumb Indiar
193467	1237181	1	13	3	SS Iyer	RR Pant	KH Pandya	1	0	1	0	0	Delhi Capitals	Mumb Indiar

193468 rows × 14 columns

Studing df1

In [32]:
```
df1
```

Out[32]:

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner	result	result
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	
4	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
811	1216547	Dubai	2020-09-28	AB de Villiers	Dubai International Cricket Stadium	0	Royal Challengers Bangalore	Mumbai Indians	Mumbai Indians	field	Royal Challengers Bangalore	tie	
812	1237177	Dubai	2020-11-05	JJ Bumrah	Dubai International Cricket Stadium	0	Mumbai Indians	Delhi Capitals	Delhi Capitals	field	Mumbai Indians	runs	
813	1237178	Abu Dhabi	2020-11-06	KS Williamson	Sheikh Zayed Stadium	0	Royal Challengers Bangalore	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	
814	1237180	Abu Dhabi	2020-11-08	MP Stoinis	Sheikh Zayed Stadium	0	Delhi Capitals	Sunrisers Hyderabad	Delhi Capitals	bat	Delhi Capitals	runs	
815	1237181	Dubai	2020-11-10	TA Boult	Dubai International Cricket Stadium	0	Delhi Capitals	Mumbai Indians	Delhi Capitals	bat	Mumbai Indians	wickets	

816 rows × 17 columns

In [33]:
```
df1['id'].isnull().sum()
```

Out[33]:
```
0
```

In [34]:
```
b= df1[df1['player_of_match'].isnull()].index.tolist()
```

In [35]:
```
b
```

Out[35]:
```
[241, 486, 511, 744]
```

In [36]:
```
c=df1[df1['result'].isnull()].index.tolist()
```

In [37]:
```
c
```

Out[37]:
```
[241, 486, 511, 744]
```

In [38]:
```
df1=df1.drop(241)
```

In [39]:
```
df1
```

Out[39]:

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner	result	result
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	
4	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
811	1216547	Dubai	2020-09-28	AB de Villiers	Dubai International Cricket Stadium	0	Royal Challengers Bangalore	Mumbai Indians	Mumbai Indians	field	Royal Challengers Bangalore	tie	
812	1237177	Dubai	2020-11-05	JJ Bumrah	Dubai International Cricket Stadium	0	Mumbai Indians	Delhi Capitals	Delhi Capitals	field	Mumbai Indians	runs	
813	1237178	Abu Dhabi	2020-11-06	KS Williamson	Sheikh Zayed Stadium	0	Royal Challengers Bangalore	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	
814	1237180	Abu Dhabi	2020-11-08	MP Stoinis	Sheikh Zayed Stadium	0	Delhi Capitals	Sunrisers Hyderabad	Delhi Capitals	bat	Delhi Capitals	runs	
815	1237181	Dubai	2020-11-10	TA Boult	Dubai International Cricket Stadium	0	Delhi Capitals	Mumbai Indians	Delhi Capitals	bat	Mumbai Indians	wickets	

815 rows × 17 columns

In [40]:
```
df1['city'].isnull().sum()
```

Out[40]: 13

In [41]:
```
df1['date'].isnull().sum()
```

Out[41]: 0

In [42]:
```
df1['player_of_match'].isnull().sum()
```

Out[42]: 3

In [43]:
```
df1['venue'].isnull().sum()
```

Out[43]: 0

In [44]:
```
df1['neutral_venue'].isnull().sum()  #deleted
```

Out[44]: 0

In [45]:
```
df1['team1'].isnull().sum()
```

Out[45]: 0

In [46]:
```
df1['team2'].isnull().sum()
```

Out[46]: 0

In [47]:
```
df1['toss_winner'].isnull().sum()
```

Out[47]: 0

In [48]:
```
df1['toss_decision'].isnull().sum()
```

Out[48]: 0

In [49]:
```
df1['winner'].isnull().sum()
```

Out[49]: 3

In [50]:
```
df1['result'].isnull().sum()
```

Out[50]: 3

In [51]:
```
df1['result_margin'].isnull().sum()   #deleted
```

Out[51]: 16

In [52]:
```
df1['eliminator'].isnull().sum()   #deleted
```

Out[52]: 3

In [53]:
```
df1['method'].isnull().sum()   #deleted
```

Out[53]: 796

In [54]:
```
df1['umpire1'].isnull().sum()
```

Out[54]: 0

In [55]:
```
df1['umpire2'].isnull().sum()
```

Out[55]: 0

In [56]:
```
del df1['method']
```

In [57]:
```
del df1['eliminator']
```

In [58]:
```
del df1['neutral_venue']
```

In [59]:
```
del df1['result_margin']
```

In [60]:
```
#Filling NULL values in the winner column
df1['winner'].fillna('Anonymous')
```

Out[60]:
```
0            Kolkata Knight Riders
1              Chennai Super Kings
2                 Delhi Daredevils
3      Royal Challengers Bangalore
4            Kolkata Knight Riders
                    ...
811    Royal Challengers Bangalore
812                 Mumbai Indians
813            Sunrisers Hyderabad
814                 Delhi Capitals
815                 Mumbai Indians
Name: winner, Length: 815, dtype: object
```

In [61]:
```
df1['player_of_match']=df1['player_of_match'].fillna('Anonymous')
```

In [62]:
```
#Filling NULL values in the city column
df1['city'].fillna('Anonymous')
```

Out[62]:
```
0       Bangalore
1      Chandigarh
2           Delhi
3          Mumbai
4         Kolkata
          ...
811         Dubai
812         Dubai
813     Abu Dhabi
814     Abu Dhabi
815         Dubai
Name: city, Length: 815, dtype: object
```

In [63]:
```
#Filling NULL values in the result column
df1['result'].fillna('Not Known')
```

Out[63]:
```
0         runs
1         runs
2      wickets
3      wickets
4      wickets
        ...
811        tie
812       runs
813    wickets
814       runs
815    wickets
Name: result, Length: 815, dtype: object
```

In [64]:
```
df.isnull().sum()
```

Out[64]:
```
id              0
inning          0
over            0
ball            0
batsman         0
non_striker     0
bowler          0
batsman_runs    0
extra_runs      0
total_runs      0
non_boundary    0
is_wicket       0
batting_team    0
bowling_team    0
dtype: int64
```

In [65]:
```
df1.isnull().sum()
```

Out[65]:
```
id                 0
city              13
date               0
player_of_match    0
venue              0
team1              0
team2              0
toss_winner        0
toss_decision      0
winner             3
result             3
umpire1            0
umpire2            0
dtype: int64
```

1) What was the count of matches played in each season?

In [66]:
```
df.to_csv('ipl_1.csv')
df1.to_csv('ipl_2.csv')
```

In [67]:
```
#Adding year column to the data frame
#step 1 - converting the str date column into int and separating year and inserting them into a list.
year = []
for i in df1['date']:
    year.append(int(i[:-6]))
print(year)
```

```
[2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008,
2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2
008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2009, 2009, 2009, 2009, 2009, 20
09, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 200
9, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 200
9, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 201
0, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 201
0, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 201
0, 2010, 2010, 2010, 2010, 2010, 2010, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 201
1, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 201
1, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 201
1, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2012, 2012, 2012, 2012, 2012, 2012, 201
2, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 201
2, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 201
2, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 201
2, 2012, 2012, 2012, 2012, 2012, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 201
3, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 201
3, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 201
3, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2014, 2014, 2014, 201
4, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 201
4, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 201
4, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2015, 2015, 2015, 2015, 2015, 2015, 201
5, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 201
5, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 201
5, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 201
6, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 201
6, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 201
6, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 201
7, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 201
7, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 201
7, 2017, 2017, 2017, 2017, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 201
8, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 201
8, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 201
8, 2018, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 201
9, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 201
9, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2020, 202
0, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 202
0, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 202
0, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020]
```

In [68]:
```
# step 2 - Adding year column to the data frame
df1.loc[:, "Year"] = year
```

In [69]:
```
df1
```

Out[69]:

	id	city	date	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	umpire1	umpire2
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	Asad Rauf	RE Koertzer
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	MR Benson	SL Shastr
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	Aleem Dar	GA Pratapkuma
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	SJ Davis	DJ Harpe
4	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	BF Bowden	K Hariharar
...	...	...	...	...	...	...	...	...	...	...	...	...	..
811	1216547	Dubai	2020-09-28	AB de Villiers	Dubai International Cricket Stadium	Royal Challengers Bangalore	Mumbai Indians	Mumbai Indians	field	Royal Challengers Bangalore	tie	Nitin Menon	PR Reiffe
812	1237177	Dubai	2020-11-05	JJ Bumrah	Dubai International Cricket Stadium	Mumbai Indians	Delhi Capitals	Delhi Capitals	field	Mumbai Indians	runs	CB Gaffaney	Nitin Menor
813	1237178	Abu Dhabi	2020-11-06	KS Williamson	Sheikh Zayed Stadium	Royal Challengers Bangalore	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	PR Reiffel	S Rav
814	1237180	Abu Dhabi	2020-11-08	MP Stoinis	Sheikh Zayed Stadium	Delhi Capitals	Sunrisers Hyderabad	Delhi Capitals	bat	Delhi Capitals	runs	PR Reiffel	S Rav
815	1237181	Dubai	2020-11-10	TA Boult	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals	bat	Mumbai Indians	wickets	CB Gaffaney	Nitin Menor

815 rows × 14 columns

In [70]:
```
def Yearwise_matches( year,season):
    name = []
    for i in df1['Year']:
        if i == year:
            name.append(i)
    print("Number of matches in ", year, ",i.e, season",season,":",len(name))
```

In [71]:
```
df1['Year'].max()
```

Out[71]: 2020

In [72]:
```
print("*"*100)
Yearwise_matches(2008,1)
print("-"*100)
Yearwise_matches(2009,2)
print("-"*100)
Yearwise_matches(2010,3)
print("-"*100)
Yearwise_matches(2011,4)
print("-"*100)
Yearwise_matches(2012,5)
print("-"*100)
Yearwise_matches(2013,6)
print("-"*100)
Yearwise_matches(2014,7)
print("-"*100)
Yearwise_matches(2015,8)
print("-"*100)
Yearwise_matches(2016,9)
print("-"*100)
Yearwise_matches(2017,10)
print("-"*100)
Yearwise_matches(2018,11)
print("-"*100)
Yearwise_matches(2019,12)
print("-"*100)
Yearwise_matches(2020,13)
print("*"*100)
```

```
**********************************************************************************************
Number of matches in  2008 ,i.e, season 1 : 58
----------------------------------------------------------------------------------------------
Number of matches in  2009 ,i.e, season 2 : 57
----------------------------------------------------------------------------------------------
Number of matches in  2010 ,i.e, season 3 : 60
----------------------------------------------------------------------------------------------
Number of matches in  2011 ,i.e, season 4 : 72
----------------------------------------------------------------------------------------------
Number of matches in  2012 ,i.e, season 5 : 74
----------------------------------------------------------------------------------------------
Number of matches in  2013 ,i.e, season 6 : 76
----------------------------------------------------------------------------------------------
Number of matches in  2014 ,i.e, season 7 : 60
----------------------------------------------------------------------------------------------
Number of matches in  2015 ,i.e, season 8 : 59
----------------------------------------------------------------------------------------------
Number of matches in  2016 ,i.e, season 9 : 60
----------------------------------------------------------------------------------------------
Number of matches in  2017 ,i.e, season 10 : 59
----------------------------------------------------------------------------------------------
Number of matches in  2018 ,i.e, season 11 : 60
----------------------------------------------------------------------------------------------
Number of matches in  2019 ,i.e, season 12 : 60
----------------------------------------------------------------------------------------------
Number of matches in  2020 ,i.e, season 13 : 60
**********************************************************************************************
```

2] How many runs were scored in each season?

In [73]:
```
# Merge datasets
merged_df = pd.merge(df, df1, on='id')
merged_df
```

Out[73]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	...	venue	team1	team2	toss_winner
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	...	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	...	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
2	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	...	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
3	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	...	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
4	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	...	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
193400	1237181	1	12	5	RR Pant	SS Iyer	NM Coulter-Nile	0	0	0	...	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals
193401	1237181	1	12	6	RR Pant	SS Iyer	NM Coulter-Nile	1	0	1	...	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals
193402	1237181	1	13	1	RR Pant	SS Iyer	KH Pandya	0	1	1	...	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals
193403	1237181	1	13	2	RR Pant	SS Iyer	KH Pandya	1	0	1	...	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals
193404	1237181	1	13	3	SS Iyer	RR Pant	KH Pandya	1	0	1	...	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals

193405 rows × 27 columns

In [74]:
```
runs_per_season = merged_df.groupby('Year')['total_runs'].sum()     #grouping and adding total runs based on match year
runs = runs_per_season.tolist()
yr=2008
print('--------Runs per season--------')
for i in runs:
    print(yr,':',i)
    print('-------------')
    yr+=1
```

```
--------Runs per season--------
2008 : 17937
-------------
2009 : 16320
-------------
2010 : 18864
-------------
2011 : 21098
-------------
2012 : 22453
-------------
2013 : 22541
-------------
2014 : 18909
-------------
2015 : 18332
-------------
2016 : 18862
-------------
2017 : 18769
-------------
2018 : 19901
-------------
2019 : 19400
-------------
2020 : 19352
-------------
```

3] What were the runs scored per match in different seasons?

In [75]:
```
runs_per_match = merged_df.groupby('id')['total_runs'].sum()
run = runs_per_match.tolist()
len(run)
```

Out[75]: 815

In [76]:
```
df1['runs_per_match']=run
```

In [77]:
```
df1.head()
```

Out[77]:

	id	city	date	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	umpire1	umpire2	Y
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	Asad Rauf	RE Koertzen	2
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	MR Benson	SL Shastri	2
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	Aleem Dar	GA Pratapkumar	2
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	SJ Davis	DJ Harper	2
4	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	BF Bowden	K Hariharan	2

In [78]:
```
data = df1.values
```

In [79]:
```
data.shape
```

Out[79]: (815, 15)

In [80]:
```
def match_per_season(y:int):                    #function to print number of matches and runs scored in each matches as per season
    print('---- Year: ',y,'----')
    print('match_id : no_of_runs')
    print('----------------')
    for i in data:
        if i[-2]==y:
            print( i[0], '  :', i[-1])
            print('----------------')
```

In [81]:
```
match_per_season(2008)
```

```
---- Year:  2008 ----
match_id : no_of_runs
----------------
335982   : 304
----------------
335983   : 447
----------------
335984   : 261
----------------
335985   : 331
----------------
335986   : 222
----------------
335987   : 334
----------------
335988   : 285
----------------
335989   : 410
----------------
```

In [82]:
```
match_per_season(2009)
```

```
---- Year:  2009 ----
match_id : no_of_runs
---------------
392181   : 311
---------------
392182   : 191
---------------
392183   : 162
---------------
392184   : 205
---------------
392185   : 266
---------------
392186   : 237
---------------
392188   : 344
---------------
392189   : 369
---------------
```

In [83]:
```
match_per_season(2010)
```

```
---- Year:  2010 ----
match_id : no_of_runs
---------------
419106   : 311
---------------
419107   : 420
---------------
419108   : 288
---------------
419109   : 271
---------------
419110   : 349
---------------
419111   : 283
---------------
419112   : 407
---------------
419113   : 273
---------------
```

In [84]:
```
match_per_season(2011)
```

```
---- Year:  2011 ----
match_id : no_of_runs
----------------
501198   : 304
----------------
501199   : 278
----------------
501200   : 323
----------------
501201   : 194
----------------
501202   : 225
----------------
501203   : 317
----------------
501204   : 303
----------------
501205   : 283
----------------
```

In [85]:
```
match_per_season(2012)
```

```
---- Year:  2012 ----
match_id : no_of_runs
----------------
548306   : 227
----------------
548307   : 197
----------------
548308   : 230
----------------
548309   : 351
----------------
548310   : 294
----------------
548311   : 312
----------------
548312   : 306
----------------
548313   : 310
----------------
```

In [86]:
```
match_per_season(2013)
```

```
---- Year:  2013 ----
match_id : no_of_runs
----------------
597998   : 257
----------------
597999   : 310
----------------
598000   : 230
----------------
598001   : 325
----------------
598002   : 287
----------------
598003   : 199
----------------
598004   : 260
----------------
598005   : 269
----------------
```

In [87]:
```
match_per_season(2014)
```

```
---- Year:  2014 ----
match_id : no_of_runs
----------------
729279   : 285
----------------
729281   : 291
----------------
729283   : 411
----------------
729285   : 268
----------------
729287   : 231
----------------
729289   : 333
----------------
729291   : 384
----------------
729293   : 261
----------------
```

In [88]:
```
match_per_season(2015)
```

```
---- Year:  2015 ----
match_id : no_of_runs
----------------
829705   : 338
----------------
829707   : 299
----------------
829709   : 298
----------------
829711   : 373
----------------
829713   : 356
----------------
829715   : 370
----------------
829717   : 336
----------------
829719   : 338
----------------
```

In [89]:
```
match_per_season(2016)
```

```
---- Year:  2016 ----
match_id : no_of_runs
----------------
980901   : 247
----------------
980903   : 197
----------------
980905   : 323
----------------
980907   : 409
----------------
980909   : 375
----------------
980911   : 327
----------------
980913   : 224
----------------
980915   : 288
----------------
```

In [90]:
```
match_per_season(2017)
```

```
---- Year:  2017 ----
match_id : no_of_runs
----------------
1082591   : 379
----------------
1082592   : 371
----------------
1082593   : 367
----------------
1082594   : 327
----------------
1082595   : 299
----------------
1082596   : 275
----------------
1082597   : 358
----------------
1082598   : 298
----------------
```

In [91]:
```
match_per_season(2018)
```

```
---- Year:  2018 ----
match_id : no_of_runs
----------------
1136561   : 334
----------------
1136562   : 333
----------------
1136563   : 353
----------------
1136564   : 252
----------------
1136565   : 407
----------------
1136566   : 213
----------------
1136567   : 298
----------------
1136568   : 314
----------------
```

In [92]:
```
match_per_season(2019)
```

```
---- Year:  2019 ----
match_id : no_of_runs
----------------
1175356   : 141
----------------
1175357   : 364
----------------
1175358   : 389
----------------
1175359   : 354
----------------
1175360   : 297
----------------
1175361   : 408
----------------
1175362   : 368
----------------
1175363   : 399
----------------
```

In [93]:
```
match_per_season(2020)
```

```
---- Year:  2020 ----
match_id : no_of_runs
----------------
1216492   : 328
----------------
1216493   : 314
----------------
1216494   : 169
----------------
1216495   : 300
----------------
1216496   : 416
----------------
1216497   : 329
----------------
1216498   : 240
----------------
1216499   : 330
----------------
```

4] Who has umpired the most?

In [94]:
```
#grouping by umpire name and sorting by number of matches umpired in descending order
count=df1.groupby('umpire1').size().sort_values(ascending=False)
#.idxmax() returns the index (in this case, the name of the umpire) corresponding to the maximum value in the Series,
# which is the umpire with the highest count.
highest_count = count.idxmax()
print('----------------------')
print(highest_count,':', count.max())
print('----------------------')
```

```
----------------------
HDPK Dharmasena : 78
----------------------
```

In [95]:
```
#grouping by umpire name and sorting by number of matches umpired in descending order
count1=df1.groupby('umpire2').size().sort_values(ascending=False)
#.idxmax() returns the index (in this case, the name of the umpire) corresponding to the maximum value in the Series,
# which is the umpire with the highest count.
highest_count1 = count1.idxmax()
print('-------------------')
print(highest_count1,':', count1.max())
print('-------------------')
```

```
-------------------
S Ravi : 84
-------------------
```

In [96]:
```
print('-'*125)
print('According to the data from both columns (umpire1 and umpire 2) the umpire who had umpired most number of matches is - ')
print('-'*125)
print(highest_count1,':',count1.max(),'(number of matches umpired)')
print('-'*125)
```

```
-----------------------------------------------------------------------------------------------------------------------------
According to the data from both columns (umpire1 and umpire 2) the umpire who had umpired most number of matches is -
-----------------------------------------------------------------------------------------------------------------------------
S Ravi : 84 (number of matches umpired)
-----------------------------------------------------------------------------------------------------------------------------
```

5] Which team has won the most tosses?

In [97]:
```
#grouping by toss_winner name and sorting by number of tosses won in descending order
tcount=df1.groupby('toss_winner').size().sort_values(ascending=False)
#.idxmax() returns the index (in this case, the name of the team) corresponding to the maximum value in the Series,
# which is the team with the highest count.
thighest_count = tcount.idxmax()
print('-------------------------------------------------')
print('The team who has won highest number of tosses is -')
print('-------------------------------------------------')
print(thighest_count,':', tcount.max(),'(number of tosses won)')
print('-------------------------------------------------')
```

```
-------------------------------------------------
The team who has won highest number of tosses is -
-------------------------------------------------
Mumbai Indians : 106 (number of tosses won)
-------------------------------------------------
```

6] What does team decide after winning the toss?

In [98]:
```
data1= np.array(df1)
```

In [99]:
```
data1=data1.tolist()
```

In [100]:
```
count=[]
count1=[]
for i in df1.groupby('toss_winner').size():
     count.append(i)
for i in df1.groupby('toss_decision').size():
     count1.append(i)
```

In [101]:
```
count
```

Out[101]:
```
[97, 43, 20, 79, 15, 85, 8, 98, 106, 20, 87, 6, 7, 87, 57]
```

In [102]:
```
#groupby('toss winner'): This part of the code groups the DataFrame df2 by the unique values in the 'toss winner' column.
#Each group corresponds to a unique team that has won the toss.|
#['toss decision']: After grouping by the 'toss winner' column, this part selects the 'toss decision' column from each group.
#This column contains information about the decision made by the team that won the toss, whether to bat or field.|
#value_counts(): This function calculates the frequency of each unique value in the selected column.
#In this case, it calculates how many times each value ('Bat' or 'Field') appears in the 'toss decision' column for each group (e
toss_decision_counts= df1.groupby('toss_winner')['toss_decision'].value_counts()
temp_df = pd.DataFrame(toss_decision_counts)
```

In [103]:
```
temp_df
```

Out[103]:

		count
toss_winner	**toss_decision**	
Chennai Super Kings	**bat**	51
	field	46
Deccan Chargers	**bat**	24
	field	19
Delhi Capitals	**field**	13
	bat	7
Delhi Daredevils	**field**	51
	bat	28
Gujarat Lions	**field**	14
	bat	1
Kings XI Punjab	**field**	58
	bat	27
Kochi Tuskers Kerala	**field**	5
	bat	3
Kolkata Knight Riders	**field**	64
	bat	34
Mumbai Indians	**field**	58
	bat	48
Pune Warriors	**bat**	11
	field	9
Rajasthan Royals	**field**	53
	bat	34
Rising Pune Supergiant	**field**	6
Rising Pune Supergiants	**field**	4
	bat	3
Royal Challengers Bangalore	**field**	63
	bat	24

		count
toss_winner	toss_decision	
Sunrisers Hyderabad	field	33
	bat	24

7] How does the toss decisions vary across seasons?

In [104]:
```
toss_decision_season1 = df1.groupby(['Year', 'toss_decision']).size()
```

In [105]:
```
lst = toss_decision_season1.tolist()

bat=[]
c=0
for i in lst:
    if c%2 == 0:
        bat.append(i)
    c=c+1
```

In [106]:
```
bat
```

Out[106]: [26, 35, 39, 24, 37, 45, 19, 25, 11, 11, 10, 10, 27]

In [107]:
```
field=[]
c=0
for i in lst:
    if c%2 != 0:
        field.append(i)
    c=c+1
```

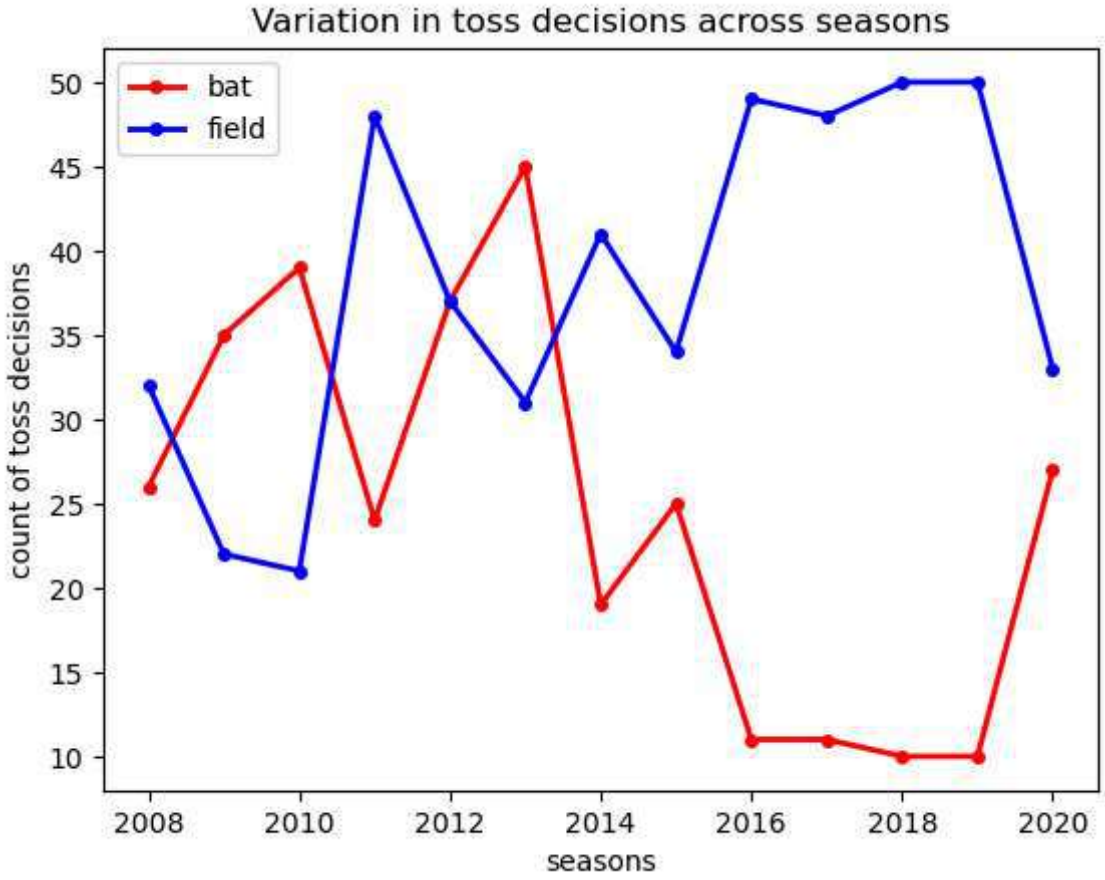
In [108]:
```
field
```

Out[108]: [32, 22, 21, 48, 37, 31, 41, 34, 49, 48, 50, 50, 33]

In [109]:
```
season=[]
for i in df1['Year'].unique():
    season.append(i)
season
```

Out[109]: [2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

In [110]:
```
plt.plot(season,bat,label='bat', color='r',linewidth=2,markersize=8, marker='.')
plt.plot(season,field,label='field',color='blue',linewidth=2,markersize=8, marker='.')
plt.title('Variation in toss decisions across seasons')
plt.xlabel('seasons')
plt.ylabel('count of toss decisions')
plt.legend()
plt.show()
```



8] Does winning the toss imply winning the game?

In [111]:
```
c = 0
d = 0
for i, row in df1.iterrows():
    if row['toss_winner']==row['winner']:
        c+=1
    if row['toss_winner'] and row['toss_winner']!= row['winner']:
        d+=1
#win_rate_toss_winners = toss_win_match_win / total_toss_wins
win_rate_toss_winners = 418/812

#Calculate the overall match win rate for teams not winning the toss
win_rate_non_toss_winners = 1 - win_rate_toss_winners

print('-----------------------------------------------------------------------------------------------')
print('Toss_win_match_win:',c)
print('win_rate_toss_winners:',win_rate_toss_winners)
print('-----------------------------------------------------------------------------------------------')
print('Toss_lose_match_win',d)
print('win_rate_non_toss_winners :',win_rate_non_toss_winners )
print('-----------------------------------------------------------------------------------------------')
```

```
-----------------------------------------------------------------------------------------------
Toss_win_match_win: 418
win_rate_toss_winners: 0.5147783251231527
-----------------------------------------------------------------------------------------------
Toss_lose_match_win 397
win_rate_non_toss_winners : 0.4852216748768473
-----------------------------------------------------------------------------------------------
```

In [112]:
```
print('-------------------------------------------------------------------------------------------------------------------')
print('As per calculations the win rate of both toss winners and non toss winners is almost the same with a difference of 3%-4%.
print('So we can safely conclude that winning the toss does not imply winning the game.')
print('But if accurate conclusion or marginal values are concerned then due to number of toss winners winning is higher than numb
print('-------------------------------------------------------------------------------------------------------------------')
```

```
-------------------------------------------------------------------------------------------------------------------
As per calculations the win rate of both toss winners and non toss winners is almost the same with a difference of 3%-4%.
So we can safely conclude that winning the toss does not imply winning the game.
But if accurate conclusion or marginal values are concerned then due to number of toss winners winning is higher than number of
non toss winners slightly
-------------------------------------------------------------------------------------------------------------------
```

9] How many times has the chasing team won the match?

In [113]:
```
temp1=[]
for i, row in df1.iterrows():            #teams who choose to field and won the match
    if row['toss_winner']==row['winner']:
        if row['toss_decision']=='field':
            temp1.append(row['winner'])
len(temp1)  #273

temp2=[] #187
temp3=[] #132
for i, row in df1.iterrows():
    if row['toss_decision']!=row['winner']:
        if row['toss_decision']=='bat':    #teams who lost the toss toss decision=bat(decided by toss winner) and won
            if row['team1']==row['winner']:
                temp2.append(row['winner'])
            elif row['team2']==row['winner']:
                temp3.append(row['winner'])

total=temp1+temp2+temp3
print('------------------------------------------------------------')
print('The number of times chasing team had won the match:',len(total))
print('------------------------------------------------------------')
```

```
------------------------------------------------------------
The number of times chasing team had won the match: 592
------------------------------------------------------------
```

10] Which all teams had won this tournament?

In [114]:
```
won = df1['winner'].unique()
won = won.tolist()
```

In [115]:
```
print('All the teams who had won the tournament are:')
print('-----------------------------------------------')
for i in won:
    print(i)
    print('-----------------------------------------------')
```

```
All the teams who had won the tournament are:
-----------------------------------------------
Kolkata Knight Riders
-----------------------------------------------
Chennai Super Kings
-----------------------------------------------
Delhi Daredevils
-----------------------------------------------
Royal Challengers Bangalore
-----------------------------------------------
Rajasthan Royals
-----------------------------------------------
Kings XI Punjab
-----------------------------------------------
Deccan Chargers
-----------------------------------------------
Mumbai Indians
-----------------------------------------------
Pune Warriors
-----------------------------------------------
Kochi Tuskers Kerala
-----------------------------------------------
Sunrisers Hyderabad
-----------------------------------------------
nan
-----------------------------------------------
Rising Pune Supergiants
-----------------------------------------------
Gujarat Lions
-----------------------------------------------
Rising Pune Supergiant
-----------------------------------------------
Delhi Capitals
-----------------------------------------------
```

11] Which teams have played most number of matches?

In [116]:
```
# Count the occurrences of each team in 'team1' and 'team2' columns
#value_counts() method, which returns a Series containing the counts of unique values in a column.
team1_counts = df1['team1'].value_counts()
team2_counts = df1['team2'].value_counts()
```

In [117]:
```
# Combine the counts for each team
#Combining counts for each team: After counting the occurrences of each team in both columns, the code combines these counts to g
#This is done using the add() method, which adds two Series together.
#team1_counts.add(team2_counts, fill_value=0): This line adds the counts of team occurrences in both 'team1' and 'team2' columns
#The fill_value=0 parameter ensures that if a team appears in only one column and not the other, its count is treated as 0.
total_matches = team1_counts.add(team2_counts, fill_value=0)
```

In [118]:
```
# Sort the teams based on the total number of matches played
#sorts the total_matches Series in descending order (ascending=False), so the team with the highest count (i.e., most matches pl
most_matches_played = total_matches.sort_values(ascending=False)
```

In [119]:
```
print('Teams with the most number of matches played:')
temp1_df = pd.DataFrame(most_matches_played)
temp1_df
```

```
Teams with the most number of matches played:
```

Out[119]:

	count
Mumbai Indians	203
Royal Challengers Bangalore	195
Kolkata Knight Riders	192
Kings XI Punjab	190
Chennai Super Kings	178
Rajasthan Royals	161
Delhi Daredevils	160
Sunrisers Hyderabad	124
Deccan Chargers	75
Pune Warriors	45
Delhi Capitals	33
Gujarat Lions	30
Rising Pune Supergiant	16
Kochi Tuskers Kerala	14
Rising Pune Supergiants	14

12] Which team has won the most number of times?

In [120]:
```
winner_counts = df1['winner'].value_counts()
```

In [121]:
```
#.idxmax() returns the index (in this case, the name of the team) corresponding to the maximum value in the Series,
# which is the team with the highest count.
most_win = winner_counts.idxmax()
print('------------------------------------------------------------')
print('The team which has won most number of times is -')
print(most_win,':', winner_counts.max())
print('------------------------------------------------------------')
```

```
------------------------------------------------------------
The team which has won most number of times is -
Mumbai Indians : 120
------------------------------------------------------------
```

13] Which team has the highest winning percentage?

In [122]:
```
# Count the occurrences of each team in 'team1' and 'team2' columns
#value_counts() method, which returns a Series containing the counts of unique values in a column.
team1_counts = df1['team1'].value_counts()
team2_counts = df1['team2'].value_counts()
```

In [123]:
```
# Combine the counts for each team
total_matches = team1_counts.add(team2_counts, fill_value=0)
```

In [124]:
```
# Calculate the total number of matches won by each team
win_counts = df1['winner'].value_counts()
```

In [125]:
```
# Calculate the winning percentage for each team
winning_percentage = (win_counts / total_matches) * 100
```

In [126]:
```
#Sorting values of winning percentage in descending order
most_win = winning_percentage.sort_values(ascending=False)
```

In [127]:
```
# Find the team with the highest winning percentage
team_highest_percentage = winning_percentage.idxmax()
highest_percentage = winning_percentage.max()
```

In [128]:
```
print('------------------------------------------------------------------')
print("Team with the highest winning percentage:", team_highest_percentage)
print("Winning percentage:", highest_percentage)
print('------------------------------------------------------------------')
```

```
------------------------------------------------------------------
Team with the highest winning percentage: Rising Pune Supergiant
Winning percentage: 62.5
------------------------------------------------------------------
```

14] Is there any lucky venue for a particular team?

In [129]:
```
def lucky_venue_for_team(team_name):
#Filtering the dataset: The code filters the dataset df2 to include only matches involving the specified team
#It selects rows where either 'team1' or 'team2' is equal to the specified team name.
    team_matches = df1[(df1['team1'] == team_name) | (df1['team2'] == team_name)]
#-----------------------------------------------------------------------------------------------------------------
#Calculating match counts at each venue: The code calculates the total number of matches played by the specified team at each ver
#It uses the value_counts() method to count the occurrences of each venue in the filtered dataset.
    venue_counts = team_matches['venue'].value_counts()
#-----------------------------------------------------------------------------------------------------------------
#Calculating match win counts at each venue: Similarly, the code calculates the total number of matches won by the specified tean
#It first filters the dataset to include only matches where the specified team is the winner, and then counts the occurrences of
    venue_win_counts = team_matches[team_matches['winner'] == team_name]['venue'].value_counts()
#Calculating win percentage at each venue: The code calculates the win percentage for the specified team at each venue by dividir
    win_percentage = (venue_win_counts / venue_counts) * 100
#Finding the lucky venue: Finally, the code finds the venue with the highest win percentage for the specified team using the idxr
    lucky_venue = win_percentage.idxmax()
#It also retrieves the highest win percentage using the max() method.
    highest_win_percentage = win_percentage.max()
#Printing the result
    print("Lucky venue for", team_name, ":", lucky_venue)
    print("Win percentage at", lucky_venue, ":", highest_win_percentage)
```

In [130]:
```
lst=df1['team1'].unique().tolist()
```

In [131]:
```
lst
```

Out[131]:
```
['Royal Challengers Bangalore',
 'Kings XI Punjab',
 'Delhi Daredevils',
 'Mumbai Indians',
 'Kolkata Knight Riders',
 'Rajasthan Royals',
 'Deccan Chargers',
 'Chennai Super Kings',
 'Kochi Tuskers Kerala',
 'Pune Warriors',
 'Sunrisers Hyderabad',
 'Gujarat Lions',
 'Rising Pune Supergiants',
 'Rising Pune Supergiant',
 'Delhi Capitals']
```

In [132]:
```
for team_name in lst:
    print('-------------------------------------------------------------------------------------------------')
    lucky_venue_for_team(team_name)
print('-------------------------------------------------------------------------------------------------')
```

```
--------------------------------------------------------------------------------------------------
Lucky venue for Royal Challengers Bangalore : Brabourne Stadium
Win percentage at Brabourne Stadium : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Kings XI Punjab : De Beers Diamond Oval
Win percentage at De Beers Diamond Oval : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Delhi Daredevils : Buffalo Park
Win percentage at Buffalo Park : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Mumbai Indians : Holkar Cricket Stadium
Win percentage at Holkar Cricket Stadium : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Kolkata Knight Riders : Barabati Stadium
Win percentage at Barabati Stadium : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Rajasthan Royals : Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium
Win percentage at Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Deccan Chargers : Himachal Pradesh Cricket Association Stadium
Win percentage at Himachal Pradesh Cricket Association Stadium : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Chennai Super Kings : Buffalo Park
Win percentage at Buffalo Park : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Kochi Tuskers Kerala : Eden Gardens
Win percentage at Eden Gardens : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Pune Warriors : Feroz Shah Kotla
Win percentage at Feroz Shah Kotla : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Sunrisers Hyderabad : Brabourne Stadium
Win percentage at Brabourne Stadium : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Gujarat Lions : Eden Gardens
Win percentage at Eden Gardens : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Rising Pune Supergiants : Feroz Shah Kotla
Win percentage at Feroz Shah Kotla : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Rising Pune Supergiant : Eden Gardens
Win percentage at Eden Gardens : 100.0
--------------------------------------------------------------------------------------------------
Lucky venue for Delhi Capitals : Eden Gardens
```

```
Win percentage at Eden Gardens : 100.0
--------------------------------------------------------------------------------------------------
```

15] Innings wise comparison between teams

In [133]:
```
grouped_data = df.groupby(['inning', 'batting_team'])

# Step 2: Calculate Metrics
innings_comparison = grouped_data.agg({
    #we calculate the total runs scored by each team in each inning using the agg() function, specifying that we want to sum the
    'total_runs': 'sum'  # Total runs scored
}).reset_index()
```

In [134]:
```
innings_comparison
```

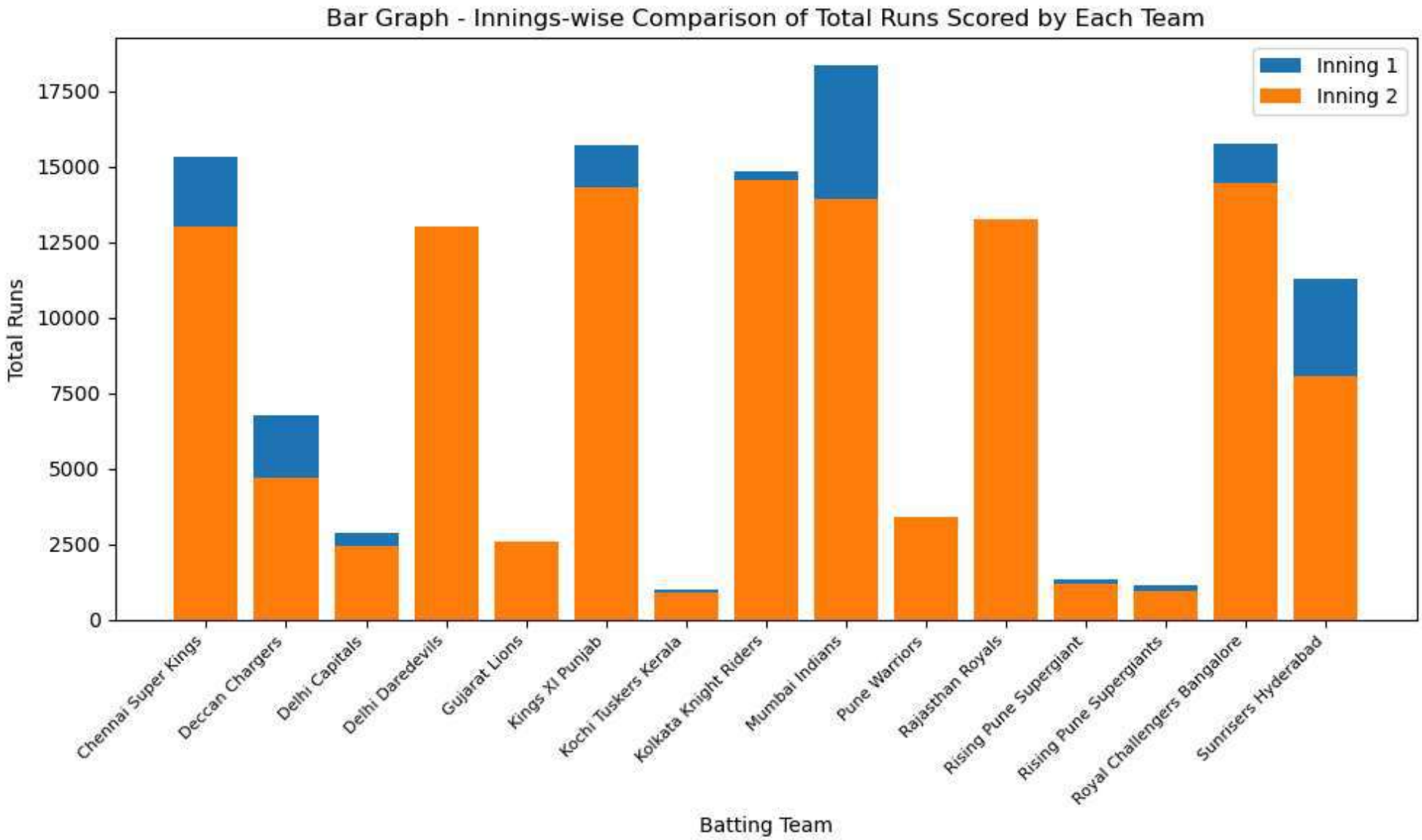
Out[134]:

	inning	batting_team	total_runs
0	1	Chennai Super Kings	15344
1	1	Deccan Chargers	6765
2	1	Delhi Capitals	2860
3	1	Delhi Daredevils	11247
4	1	Gujarat Lions	2267
5	1	Kings XI Punjab	15710
6	1	Kochi Tuskers Kerala	1009
7	1	Kolkata Knight Riders	14842
8	1	Mumbai Indians	18338
9	1	Pune Warriors	2973
10	1	Rajasthan Royals	11236
11	1	Rising Pune Supergiant	1304
12	1	Rising Pune Supergiants	1123
13	1	Royal Challengers Bangalore	15774
14	1	Sunrisers Hyderabad	11277
15	2	Chennai Super Kings	13019
16	2	Deccan Chargers	4698
17	2	Delhi Capitals	2436
18	2	Delhi Daredevils	13038
19	2	Gujarat Lions	2589
20	2	Kings XI Punjab	14307
21	2	Kochi Tuskers Kerala	892
22	2	Kolkata Knight Riders	14541
23	2	Mumbai Indians	13948
24	2	Pune Warriors	3385
25	2	Rajasthan Royals	13271
26	2	Rising Pune Supergiant	1166
27	2	Rising Pune Supergiants	940

	inning	batting_team	total_runs
28	2	Royal Challengers Bangalore	14440
29	2	Sunrisers Hyderabad	8055

In [135]:
```
# Step 3: Visualize Data
plt.figure(figsize=(10, 6))
for inning in innings_comparison['inning'].unique():
    inning_data = innings_comparison[innings_comparison['inning'] == inning]
    plt.bar(inning_data['batting_team'], inning_data['total_runs'], label=f"Inning {inning}")

plt.xlabel('Batting Team')
plt.ylabel('Total Runs')
plt.title('Bar Graph - Innings-wise Comparison of Total Runs Scored by Each Team')
plt.legend()
plt.xticks(rotation=45, ha='right', fontsize=8)
plt.tight_layout()
plt.show()
```



In [136]:
```
i1=[]
i2=[]
for i,row in innings_comparison.iterrows():
    if(row['inning']==1):
        i1.append(row['total_runs'])
```

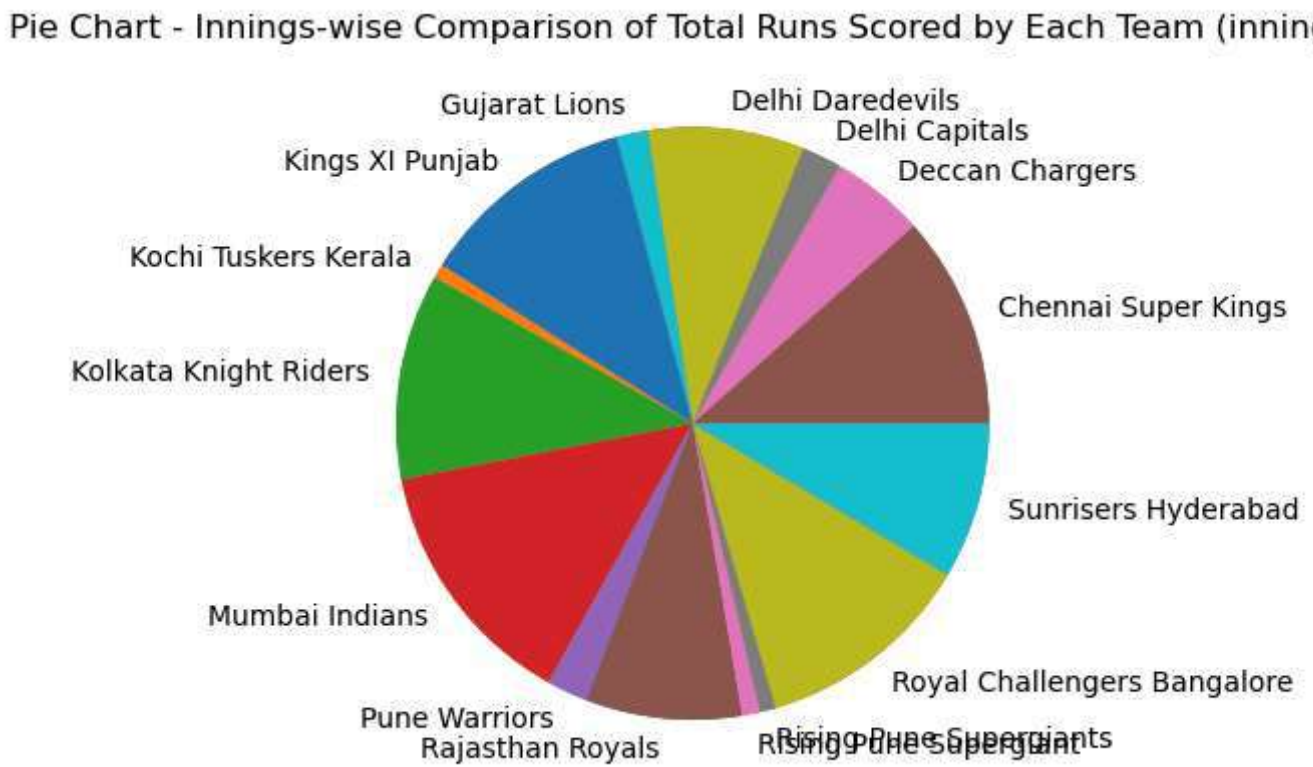
In [137]:
```
for i,row in innings_comparison.iterrows():
    if(row['inning']==2):
        i2.append(row['total_runs'])
```

In [138]:
```
teams=[]
for i,row in innings_comparison.iterrows():
    teams.append(row['batting_team'])
    if i==14:
        break
```

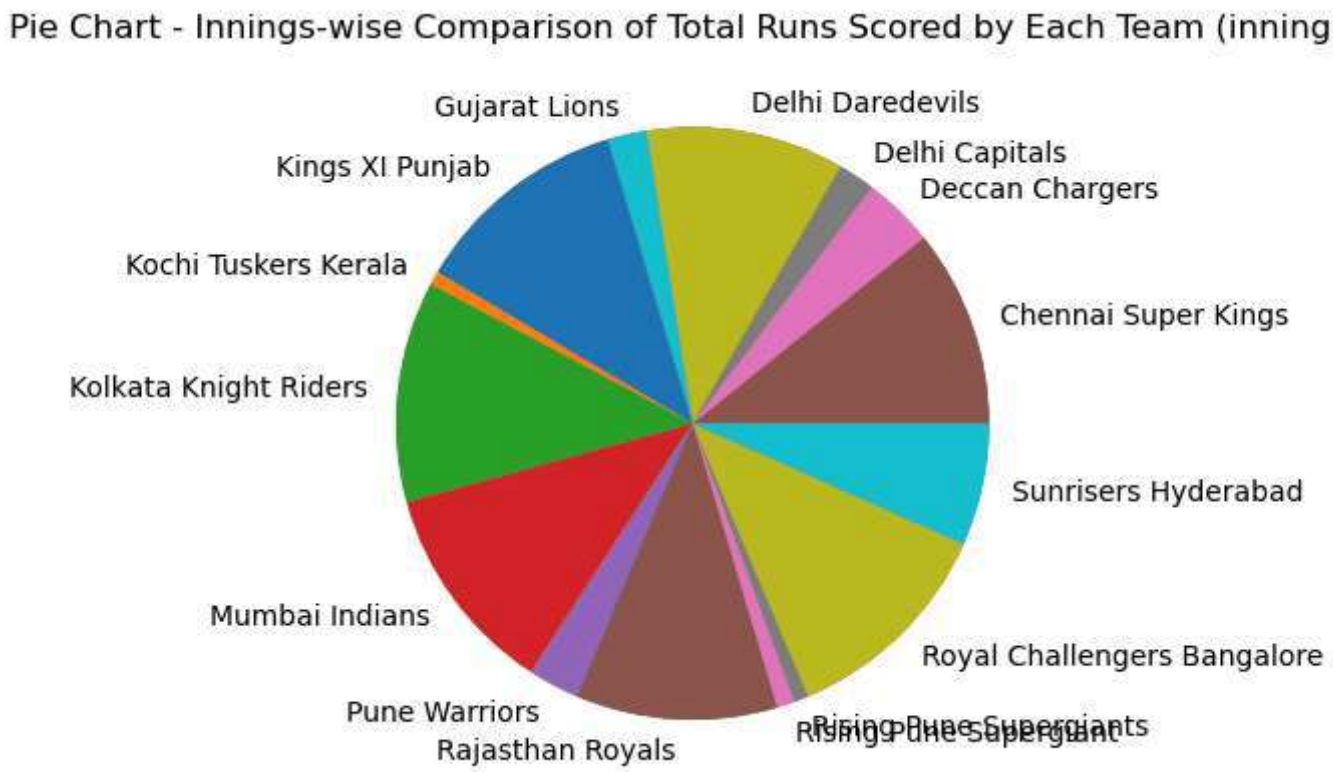
In [139]:
```
teams
```

Out[139]:
```
['Chennai Super Kings',
 'Deccan Chargers',
 'Delhi Capitals',
 'Delhi Daredevils',
 'Gujarat Lions',
 'Kings XI Punjab',
 'Kochi Tuskers Kerala',
 'Kolkata Knight Riders',
 'Mumbai Indians',
 'Pune Warriors',
 'Rajasthan Royals',
 'Rising Pune Supergiant',
 'Rising Pune Supergiants',
 'Royal Challengers Bangalore',
 'Sunrisers Hyderabad']
```

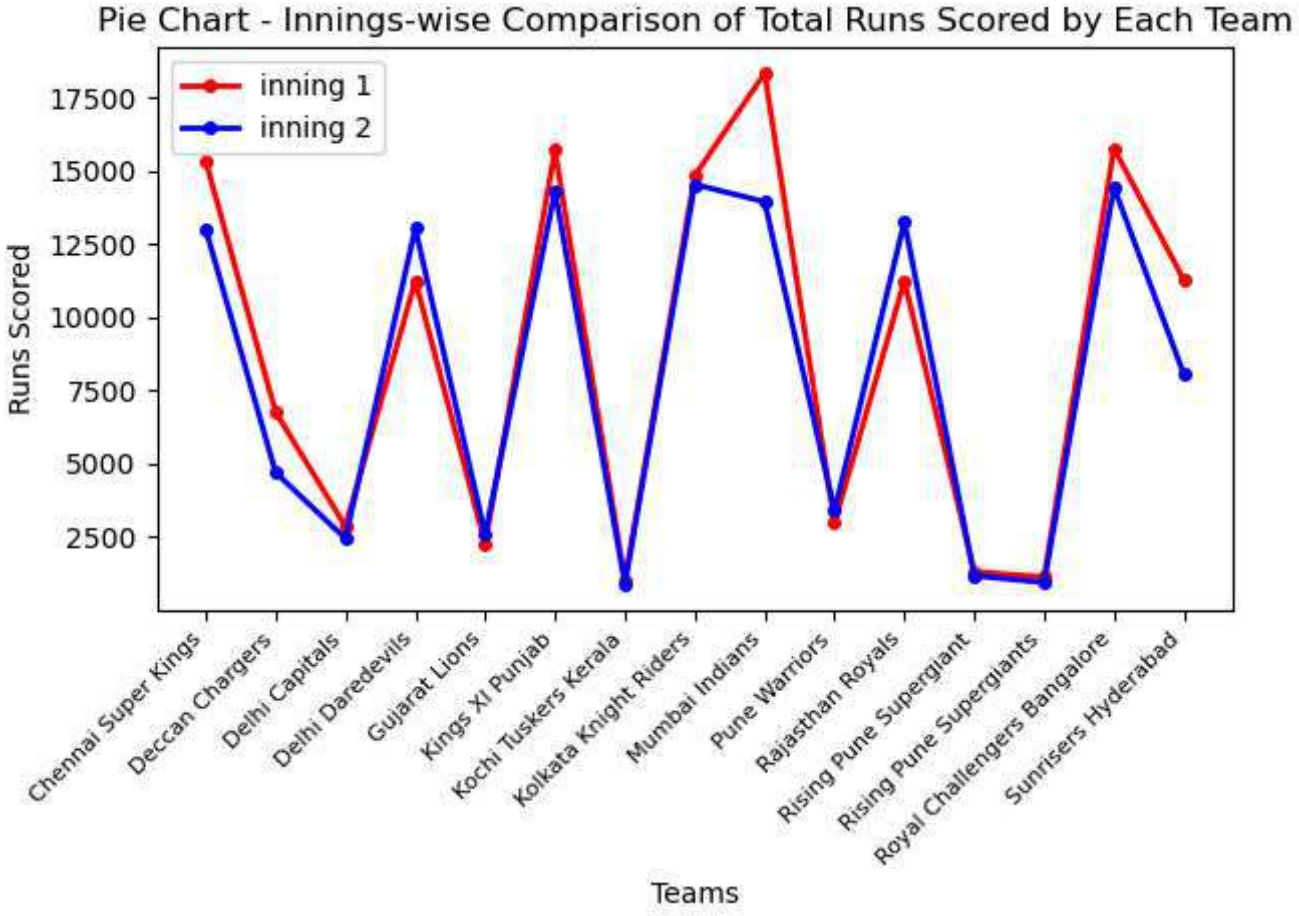
In [140]:
```
plt.pie(i1)
labels=['Chennai Super Kings',
 'Deccan Chargers',
 'Delhi Capitals',
 'Delhi Daredevils',
 'Gujarat Lions',
 'Kings XI Punjab',
 'Kochi Tuskers Kerala',
 'Kolkata Knight Riders',
 'Mumbai Indians',
 'Pune Warriors',
 'Rajasthan Royals',
 'Rising Pune Supergiant',
 'Rising Pune Supergiants',
 'Royal Challengers Bangalore',
 'Sunrisers Hyderabad']
plt.pie(i1,labels=labels)
plt.title('Pie Chart - Innings-wise Comparison of Total Runs Scored by Each Team (inning 1)')
plt.show()
```



In [141]:
```
plt.pie(i2)
labels=['Chennai Super Kings',
 'Deccan Chargers',
 'Delhi Capitals',
 'Delhi Daredevils',
 'Gujarat Lions',
 'Kings XI Punjab',
 'Kochi Tuskers Kerala',
 'Kolkata Knight Riders',
 'Mumbai Indians',
 'Pune Warriors',
 'Rajasthan Royals',
 'Rising Pune Supergiant',
 'Rising Pune Supergiants',
 'Royal Challengers Bangalore',
 'Sunrisers Hyderabad']
plt.pie(i2,labels=labels)
plt.title('Pie Chart - Innings-wise Comparison of Total Runs Scored by Each Team (inning 2)')
plt.show()
```



In [142]:
```
plt.plot(teams,i1,label='inning 1', color='r',linewidth=2,markersize=8, marker='.')
plt.plot(teams,i2,label='inning 2',color='blue',linewidth=2,markersize=8, marker='.')
plt.title('Pie Chart - Innings-wise Comparison of Total Runs Scored by Each Team')
plt.xlabel('Teams')
plt.ylabel('Runs Scored')
plt.legend()
plt.xticks(rotation=45, ha='right', fontsize=8)
plt.tight_layout()
plt.show()
```



16] Which team has scored the most 200+ scores?

In [143]:
```
# Group the dataset by match and batting team, and sum the total runs scored
runs_by_team_per_match = df.groupby(['id', 'batting_team'])['total_runs'].sum().reset_index()
runs_by_team_per_match
```

Out[143]:

	id	batting_team	total_runs
0	335982	Kolkata Knight Riders	222
1	335982	Royal Challengers Bangalore	82
2	335983	Chennai Super Kings	240
3	335983	Kings XI Punjab	207
4	335984	Delhi Daredevils	132
...	...	...	...
1625	1237178	Sunrisers Hyderabad	132
1626	1237180	Delhi Capitals	189
1627	1237180	Sunrisers Hyderabad	172
1628	1237181	Delhi Capitals	156
1629	1237181	Mumbai Indians	157

1630 rows × 3 columns

In [144]:
```
batting=[]
bowling=[]
batting_runs=[]
bowling_runs=[]
for i,row in runs_by_team_per_match.iterrows():
    if i % 2:
        batting.append(row['batting_team'])
        batting_runs.append(row['total_runs'])
    else :
        bowling.append(row['batting_team'])
        bowling_runs.append(row['total_runs'])
```

In [145]:
```
df1['batting_team']=batting
df1['batting_team_runs']=batting_runs
df1['bowling_team']=bowling
df1['bowling_team_runs']=bowling_runs
```

In [146]:
```
temp1=[]
temp2=[]
for i,row in df1.iterrows():
    if row['batting_team_runs'] >= 200 :
        temp1.append([row['batting_team'], row['batting_team_runs']])

for i,row in df1.iterrows():
    if row['bowling_team_runs'] >= 200 :
        temp2.append([row['bowling_team'], row['bowling_team_runs']])
```

In [147]:
```
temp_1=pd.DataFrame(temp1, columns =['batting_team','batting_team_runs'])
temp_2=pd.DataFrame(temp2, columns =['bowling_team','bowling_team_runs'])
```

In [148]:
```
temp1_df = temp_1['batting_team'].value_counts()
```

In [149]:
```
temp2_df = temp_2['bowling_team'].value_counts()
```

In [150]:
```
temp1_df.max(),temp2_df.max()
```

Out[150]: (13, 11)

In [151]:
```
print('-------------------------------------------------------')
print('The team which has scored the most 200+ runs is')
print('-------------------------------------------------------')
print(temp2_df.idxmax(),':',temp2_df.max())
print('-------------------------------------------------------')
```

```
-------------------------------------------------------
The team which has scored the most 200+ runs is
-------------------------------------------------------
Kings XI Punjab : 11
-------------------------------------------------------
```

17] Which team has conceded 200+ scores the most?

In [152]:
```
lst1 = []
for i, row in df1.iterrows():
    if row['toss_decision'] == 'bat':
        lst1.append(  row['toss_winner'])
    elif row['toss_decision'] == 'field':
        if row['toss_winner']==row['team1']:
            lst1.append(row['team2'])
        elif row['toss_winner']==row['team2']:
            lst1.append(row['team1'])
```

In [153]:
```
df1['batting_team']=lst1
```

In [154]:
```
lst2=[]
for i,row in df1.iterrows():
    if row['batting_team']==row['team1']:
        lst2.append(row['team2'])
    elif row['batting_team']==row['team2']:
        lst2.append(row['team1'])
```

In [155]:
```
df1['bowling_team']=lst2
```

In [156]:
```
temp1=[]
for i,row in df1.iterrows():
    if row['batting_team_runs'] >= 200 :
        temp1.append([row['batting_team'], row['batting_team_runs'], row['bowling_team']])
```

In [157]:
```
temp_df1 = pd.DataFrame(temp1,columns=['batting_team', 'batting_team_runs', 'bowling_team'])
```

In [158]:
```
temp_df1
```

Out[158]:

	batting_team	batting_team_runs	bowling_team
0	Chennai Super Kings	207	Kings XI Punjab
1	Chennai Super Kings	202	Mumbai Indians
2	Deccan Chargers	217	Rajasthan Royals
3	Kolkata Knight Riders	204	Deccan Chargers
4	Rajasthan Royals	211	Chennai Super Kings
5	Rajasthan Royals	211	Kings XI Punjab
6	Mumbai Indians	208	Rajasthan Royals
7	Kings XI Punjab	204	Royal Challengers Bangalore
8	Mumbai Indians	218	Delhi Daredevils
9	Chennai Super Kings	223	Rajasthan Royals
10	Kolkata Knight Riders	200	Kings XI Punjab
11	Delhi Daredevils	202	Kings XI Punjab
12	Royal Challengers Bangalore	205	Kings XI Punjab
13	Chennai Super Kings	205	Royal Challengers Bangalore
14	Royal Challengers Bangalore	208	Chennai Super Kings
15	Delhi Daredevils	207	Mumbai Indians
16	Chennai Super Kings	222	Delhi Daredevils
17	Chennai Super Kings	200	Kolkata Knight Riders
18	Chennai Super Kings	223	Sunrisers Hyderabad
19	Chennai Super Kings	205	Kings XI Punjab
20	Sunrisers Hyderabad	211	Kings XI Punjab
21	Kings XI Punjab	202	Chennai Super Kings
22	Chennai Super Kings	209	Sunrisers Hyderabad
23	Mumbai Indians	209	Royal Challengers Bangalore
24	Royal Challengers Bangalore	200	Rajasthan Royals
25	Royal Challengers Bangalore	226	Kings XI Punjab
26	Sunrisers Hyderabad	201	Rajasthan Royals
27	Royal Challengers Bangalore	235	Mumbai Indians

	batting_team	batting_team_runs	bowling_team
28	Mumbai Indians	202	Chennai Super Kings
29	Royal Challengers Bangalore	248	Gujarat Lions
30	Mumbai Indians	206	Delhi Daredevils
31	Royal Challengers Bangalore	211	Kings XI Punjab
32	Sunrisers Hyderabad	208	Royal Challengers Bangalore
33	Sunrisers Hyderabad	207	Royal Challengers Bangalore
34	Royal Challengers Bangalore	213	Gujarat Lions
35	Sunrisers Hyderabad	207	Kings XI Punjab
36	Sunrisers Hyderabad	209	Kolkata Knight Riders
37	Gujarat Lions	208	Delhi Daredevils
38	Mumbai Indians	212	Delhi Daredevils
39	Kings XI Punjab	223	Mumbai Indians
40	Kolkata Knight Riders	202	Chennai Super Kings
41	Kolkata Knight Riders	200	Delhi Daredevils
42	Royal Challengers Bangalore	205	Chennai Super Kings
43	Mumbai Indians	210	Kolkata Knight Riders
44	Kolkata Knight Riders	245	Kings XI Punjab
45	Royal Challengers Bangalore	204	Sunrisers Hyderabad
46	Kolkata Knight Riders	218	Kings XI Punjab
47	Rajasthan Royals	201	Sunrisers Hyderabad
48	Sunrisers Hyderabad	231	Royal Challengers Bangalore
49	Royal Challengers Bangalore	205	Kolkata Knight Riders
50	Royal Challengers Bangalore	213	Kolkata Knight Riders
51	Royal Challengers Bangalore	202	Kings XI Punjab
52	Sunrisers Hyderabad	212	Kings XI Punjab
53	Rajasthan Royals	216	Chennai Super Kings
54	Delhi Capitals	210	Kolkata Knight Riders
55	Sunrisers Hyderabad	219	Delhi Capitals
56	Kings XI Punjab	226	Rajasthan Royals

	batting_team	batting_team_runs	bowling_team
57	Sunrisers Hyderabad	201	Kings XI Punjab
58	Royal Challengers Bangalore	201	Mumbai Indians
59	Mumbai Indians	200	Delhi Capitals

In [159]:
```
temp_ = temp_df1['bowling_team'].value_counts()
```

In [160]:
```
print('--------------------------------------------------------------------------------')
print("Team that has conceded the most 200+ scores:", temp_.idxmax())
print('--------------------------------------------------------------------------------')
print("Number of times they conceded 200+ scores:", temp_.max())
print('--------------------------------------------------------------------------------')
```

```
--------------------------------------------------------------------------------
Team that has conceded the most 200+ scores: Kings XI Punjab
--------------------------------------------------------------------------------
Number of times they conceded 200+ scores: 15
--------------------------------------------------------------------------------
```

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

In [161]:
```
df1['batting_team_runs'].max()
```

Out[161]: 248

In [162]:
```
df1['bowling_team_runs'].max()
```

Out[162]: 263

In [163]:
```
for i,row in df1.iterrows():

        if row['bowling_team_runs']==df1['bowling_team_runs'].max():
            print('-------------------------------------------------------')
            print('The highest run scored by a team in a single match is')
            print('-------------------------------------------------------')
            print(row['winner'],':',df1['bowling_team_runs'].max())
            print('-------------------------------------------------------')
```

```
-------------------------------------------------------
The highest run scored by a team in a single match is
-------------------------------------------------------
Kings XI Punjab : 263
-------------------------------------------------------
```

19] Which is the biggest win in terms of run margin?

In [164]:
```
#creating a column named run margin in df1
temp1=[]
for i,row in df1.iterrows():
    if row['batting_team_runs'] > row['bowling_team_runs']:
        temp1.append(row['batting_team_runs'] - row['bowling_team_runs'])
    elif row['batting_team_runs'] < row['bowling_team_runs']:
        temp1.append(row['bowling_team_runs'] - row['batting_team_runs'])
    else:
        temp1.append(0)
```

In [165]:
```
df1['run_margin']=temp1
```

In [166]:
```
for i, row in df1.iterrows():
    print('-----------------------------------------------------------------')
    print('The biggest win in terms of run margin is', df1['run_margin'].max())
    print('-----------------------------------------------------------------')
    print('The winner team is', row['winner'])
    print('-----------------------------------------------------------------')
    break
```

```
-----------------------------------------------------------------
The biggest win in terms of run margin is 185
-----------------------------------------------------------------
The winner team is Kolkata Knight Riders
-----------------------------------------------------------------
```

20] Which batsmen have played the most number of balls?

In [167]:
```
#groups the data in DataFrame df by the 'batsman' column using the groupby() function.
#This step creates separate groups for each unique value in the 'batsman' column.
#Within each group, the code counts the number of balls faced by each batsman using the count() function applied to the 'ball' co
#This step creates a Series where the index is the batsman's name and the values are the number of balls faced by each batsman.
#Resetting index: The code resets the index of the resulting Series using the reset_index() method to convert the Series to a Dat
#This step ensures that the batsman names become a regular column in the DataFrame.
batsman_balls = df.groupby('batsman')['ball'].count().reset_index()
```

In [168]:
```
#meaningful names to the columns representing the batsman's name and the number of balls faced.
batsman_balls.columns = ['Batsman', 'Balls Faced']
```

In [169]:
```
# sorts the DataFrame in descending order based on the number of balls faced by each batsman.
batsman_balls = batsman_balls.sort_values(by='Balls Faced', ascending=False)
```

In [170]:
```
# prints the top batsmen who have faced the most number of balls.
print('------------------------------------------------------------')
print("Batsmen who have played the most number of balls:")
for i,row in batsman_balls.iterrows():
    print(row['Batsman'],':',row['Balls Faced'])
    break
print('------------------------------------------------------------')
```

```
------------------------------------------------------------
Batsmen who have played the most number of balls:
V Kohli : 4609
------------------------------------------------------------
```

21] Who are the leading run-scorers of all the time?

In [171]:
```
# Group the data by batsman and sum the runs scored by each batsman
batsman_score = df.groupby('batsman')['batsman_runs'].sum().reset_index()

# Sort the data in descending order based on runs scored
leading_run_scorers = batsman_score.sort_values(by='batsman_runs', ascending=False).reset_index()

# Print the leading run-scorers
print('----------------------------------------------------')
print("Leading run-scorers of all time:")
for i,row in leading_run_scorers.iterrows():
    print('----------------------------------------------------')
    print(row['batsman'],':', row['batsman_runs'])
    if i>5:
        break
print('----------------------------------------------------')
```

```
----------------------------------------------------
Leading run-scorers of all time:
----------------------------------------------------
V Kohli : 5878
----------------------------------------------------
SK Raina : 5368
----------------------------------------------------
DA Warner : 5254
----------------------------------------------------
RG Sharma : 5230
----------------------------------------------------
S Dhawan : 5197
----------------------------------------------------
AB de Villiers : 4849
----------------------------------------------------
CH Gayle : 4772
----------------------------------------------------
```

22] Who has hit the most number of 4's?

In [172]:
```
#This line filters the dataset df to include only those deliveries where the batsman scored 4 runs.
#It creates a new DataFrame fours containing only the deliveries where 4 runs were scored.
fours = df[df['batsman_runs'] == 4]

# Count the occurrences of each batsman hitting a four
most_fours = fours['batsman'].value_counts()

# Identify the batsman with the most number of fours
batsman_most_fours = most_fours.idxmax()
num_fours = most_fours.max()

print('----------------------------------------------------------------------')
print("Batsman who has hit the most number of fours:", batsman_most_fours)
print('----------------------------------------------------------------------')
print("Number of fours:", num_fours)
print('----------------------------------------------------------------------')
```

```
----------------------------------------------------------------------
Batsman who has hit the most number of fours: S Dhawan
----------------------------------------------------------------------
Number of fours: 591
----------------------------------------------------------------------
```

23] Who has hit the most number of 6's?

In [173]:
```
#This line filters the dataset df to include only those deliveries where the batsman scored 6 runs.
#It creates a new DataFrame six containing only the deliveries where 6 runs were scored.
six = df[df['batsman_runs'] == 6]

# Count the occurrences of each batsman hitting six
most_six = six['batsman'].value_counts()

# Identify the batsman with the most number of fours
batsman_most_six = most_six.idxmax()
num_six = most_six.max()

print('----------------------------------------------------------------------')
print("Batsman who has hit the most number of six:", batsman_most_six)
print('----------------------------------------------------------------------')
print("Number of six:", num_six)
print('----------------------------------------------------------------------')
```

```
----------------------------------------------------------------------
Batsman who has hit the most number of six: CH Gayle
----------------------------------------------------------------------
Number of six: 349
----------------------------------------------------------------------
```

24] Who has the highest strike rate ?

In [174]:
```
# Calculate the total runs scored by each batsman
total_runs = df.groupby('batsman')['batsman_runs'].sum()

# Calculate the total balls faced by each batsman
total_balls_faced = df['batsman'].value_counts()

# Calculate the strike rate for each batsman
strike_rate = (total_runs / total_balls_faced) * 100

# Identify the batsman with the highest strike rate
highest_strike_rate_batsman = strike_rate.idxmax()
highest_strike_rate = strike_rate.max()

print('-----------------------------------------------------------------')
print("Batsman with the highest strike rate:", highest_strike_rate_batsman)
print('-----------------------------------------------------------------')
print("Highest strike rate:", highest_strike_rate)
print('-----------------------------------------------------------------')
```

```
-----------------------------------------------------------------
Batsman with the highest strike rate: B Stanlake
-----------------------------------------------------------------
Highest strike rate: 250.0
-----------------------------------------------------------------
```

25] Who is leading wicket-taker?

In [175]:
```
# Filter the dataset to include only deliveries where a wicket was taken
wickets = df[df['is_wicket'] == 1]

# Count the occurrences of each bowler taking a wicket
leading_wicket_taker = wickets['bowler'].value_counts()

# Identify the bowler with the most number of wickets
leading_wicket_taker_name = leading_wicket_taker.idxmax()
leading_wicket_count = leading_wicket_taker.max()

print('----------------------------------------------------')
print("Leading wicket-taker:", leading_wicket_taker_name)
print('----------------------------------------------------')
print("Number of wickets:", leading_wicket_count)
print('----------------------------------------------------')
```

```
----------------------------------------------------
Leading wicket-taker: SL Malinga
----------------------------------------------------
Number of wickets: 188
----------------------------------------------------
```

26] Which stadium has hosted the most number of matches?

In [176]:
```
most_matches_stadium = df1['venue'].value_counts()
```

In [177]:
```
most_matches_stadium=most_matches_stadium.sort_values(ascending=False).reset_index()
```

In [178]:
```
print('------------------------------------------------------------')
for i,row in most_matches_stadium.iterrows():
    print('The stadium which has hosted most number of matches is')
    print(row['venue'],':',row['count'])
    break
print('------------------------------------------------------------')
```

```
------------------------------------------------------------
The stadium which has hosted most number of matches is
Eden Gardens : 77
------------------------------------------------------------
```

27] Who has won the most MOM awards?

In [179]:
```
most_mom= df1['player_of_match'].value_counts()
```

In [180]:
```
most_mom= most_mom.sort_values(ascending=False).reset_index()
```

In [181]:
```
print('------------------------------------------------')
print('The most man of the match(MOM) awards are won by')
print('------------------------------------------------')
for i,row in most_mom.iterrows():
    print(row['player_of_match'],':',row['count'])
    break
print('------------------------------------------------')
```

```
------------------------------------------------
The most man of the match(MOM) awards are won by
------------------------------------------------
AB de Villiers : 23
------------------------------------------------
```

28] What is the count of fours hit in each season?

In [182]:
```
four = df[df['batsman_runs']==4]
```

In [183]:
```
four
```

Out[183]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	batting_team	bowling_tea
21	335982	1	10	2	RT Ponting	BB McCullum	JH Kallis	4	0	4	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
33	335982	1	12	2	BB McCullum	DJ Hussey	JH Kallis	4	0	4	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
37	335982	1	12	6	BB McCullum	DJ Hussey	JH Kallis	4	0	4	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
45	335982	1	14	1	DJ Hussey	BB McCullum	CL White	4	0	4	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
48	335982	1	14	4	BB McCullum	DJ Hussey	CL White	4	0	4	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
193402	1237181	1	2	5	SS Iyer	S Dhawan	TA Boult	4	0	4	0	0	Delhi Capitals	Mumb Indiar
193411	1237181	1	4	2	SS Iyer	RR Pant	TA Boult	4	0	4	0	0	Delhi Capitals	Mumb Indiar
193414	1237181	1	4	5	SS Iyer	RR Pant	TA Boult	4	0	4	0	0	Delhi Capitals	Mumb Indiar
193420	1237181	1	5	5	SS Iyer	RR Pant	NM Coulter-Nile	4	0	4	0	0	Delhi Capitals	Mumb Indiar
193454	1237181	1	11	3	RR Pant	SS Iyer	KA Pollard	4	0	4	0	0	Delhi Capitals	Mumb Indiar

21908 rows × 14 columns

In [184]:
```
season1=[]
season2=[]
season3=[]
season4=[]
season5=[]
season6=[]
season7=[]
season8=[]
season9=[]
season10=[]
season11=[]
season12=[]
season13=[]

#creating a function to create list of ids for each season
def fours_per_season(year,lst):
    for i,row in df1.iterrows():
        if row['Year']==year:
            lst.append(row['id'])
    print([min(lst), max(lst)])
```

In [185]:
```
fours_per_season(2008,season1)
```

```
[335982, 336040]
```

In [186]:
```
fours_per_season(2009,season2)
```

```
[392181, 392239]
```

In [187]:
```
fours_per_season(2010,season3)
```

```
[419106, 419165]
```

In [188]:
```
fours_per_season(2011,season4)
```

```
[501198, 501271]
```

In [189]:
```
fours_per_season(2012,season5)
```

```
[548306, 548381]
```

In [190]:
```
fours_per_season(2013,season6)
```

```
[597998, 598073]
```

In [191]:
```
fours_per_season(2014,season7)
```

```
[729279, 734049]
```

In [192]:
```
fours_per_season(2015,season8)
```

```
[829705, 829823]
```

In [193]:
```
fours_per_season(2016,season9)
```

```
[980901, 981019]
```

In [194]:
```
fours_per_season(2017,season10)
```

```
[1082591, 1082650]
```

In [195]:
```
fours_per_season(2018,season11)
```

```
[1136561, 1136620]
```

In [196]:
```
fours_per_season(2019,season12)
```

```
[1175356, 1181768]
```

In [197]:
```
fours_per_season(2020,season13)
```

```
[1216492, 1237181]
```

In [198]:
```
def no_of_fours_per_season(lst,season,year):

    count=0
    for i,row in four.iterrows():
        if(row['id']>= min(lst) and row['id']<=max(lst)):
            count =count+1
    print('----------------------------------------------------------------------')
    print(season,'(i.e., year',year,')',' :', count)
```

In [199]:
```
print('*************** Count of Fours scored in each season********************')
fours =[]
fours.append([no_of_fours_per_season(season1,'season 1',2008)])
fours.append([no_of_fours_per_season(season2,'season 2',2009)])
fours.append([no_of_fours_per_season(season3,'season 3',2010)])
fours.append([no_of_fours_per_season(season4,'season 4',2011)])
fours.append([no_of_fours_per_season(season5,'season 5',2012)])
fours.append([no_of_fours_per_season(season6,'season 6',2013)])
fours.append([no_of_fours_per_season(season7,'season 7',2014)])
fours.append([no_of_fours_per_season(season8,'season 8',2015)])
fours.append([no_of_fours_per_season(season9,'season 9',2016)])
fours.append([no_of_fours_per_season(season10,'season 10',2017)])
fours.append([no_of_fours_per_season(season11,'season 11',2018)])
fours.append([no_of_fours_per_season(season12,'season 12',2019)])
fours.append([no_of_fours_per_season(season13,'season 13',2020)])
print('----------------------------------------------------------------------')
```

```
*************** Count of Fours scored in each season********************
----------------------------------------------------------------------
season 1 (i.e., year 2008 )  : 1703
----------------------------------------------------------------------
season 2 (i.e., year 2009 )  : 1317
----------------------------------------------------------------------
season 3 (i.e., year 2010 )  : 1708
----------------------------------------------------------------------
season 4 (i.e., year 2011 )  : 1916
----------------------------------------------------------------------
season 5 (i.e., year 2012 )  : 1911
----------------------------------------------------------------------
season 6 (i.e., year 2013 )  : 2052
----------------------------------------------------------------------
season 7 (i.e., year 2014 )  : 1562
----------------------------------------------------------------------
season 8 (i.e., year 2015 )  : 1607
----------------------------------------------------------------------
season 9 (i.e., year 2016 )  : 1633
----------------------------------------------------------------------
season 10 (i.e., year 2017 )  : 1611
----------------------------------------------------------------------
season 11 (i.e., year 2018 )  : 1652
----------------------------------------------------------------------
season 12 (i.e., year 2019 )  : 1653
----------------------------------------------------------------------
season 13 (i.e., year 2020 )  : 1583
----------------------------------------------------------------------
```

29] What is the count of sixes hit in each season? What is the count of runs scored from boundaries in each season?

In [200]:
```
six = df[df['batsman_runs']==6]
```

In [201]:
```
six
```

Out[201]:

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	batting_team	bowling_tea
18	335982	1	9	5	BB McCullum	RT Ponting	SB Joshi	6	0	6	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
23	335982	1	10	4	RT Ponting	BB McCullum	JH Kallis	6	0	6	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
27	335982	1	11	2	BB McCullum	RT Ponting	SB Joshi	6	0	6	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
47	335982	1	14	3	BB McCullum	DJ Hussey	CL White	6	0	6	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
51	335982	1	14	7	BB McCullum	DJ Hussey	CL White	6	0	6	0	0	Kolkata Knight Riders	Roy Challenge Bangalo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
193352	1237181	2	14	4	Ishan Kishan	RG Sharma	MP Stoinis	6	0	6	0	0	Mumbai Indians	Delhi Capita
193384	1237181	1	19	5	SS Iyer	K Rabada	NM Coulter-Nile	6	0	6	0	0	Delhi Capitals	Mumb Indiar
193441	1237181	1	9	2	RR Pant	SS Iyer	KH Pandya	6	0	6	0	0	Delhi Capitals	Mumb Indiar
193444	1237181	1	9	5	RR Pant	SS Iyer	KH Pandya	6	0	6	0	0	Delhi Capitals	Mumb Indiar
193458	1237181	1	11	7	SS Iyer	RR Pant	KA Pollard	6	0	6	0	0	Delhi Capitals	Mumb Indiar

8902 rows × 14 columns

In [202]:
```
def no_of_sixes_per_season(lst,season,year):

    count=0
    for i,row in six.iterrows():
        if(row['id']>= min(lst) and row['id']<=max(lst)):
            count =count+1
    print('----------------------------------------------------------------------')
    print(season,'(i.e., year',year,')',' :', count)
```

In [203]:
```
print('*************** Count of sixes scored in each season********************')
sixes =[]
sixes.append([no_of_sixes_per_season(season1,'season 1',2008)])
sixes.append([no_of_sixes_per_season(season2,'season 2',2009)])
sixes.append([no_of_sixes_per_season(season3,'season 3',2010)])
sixes.append([no_of_sixes_per_season(season4,'season 4',2011)])
sixes.append([no_of_sixes_per_season(season5,'season 5',2012)])
sixes.append([no_of_sixes_per_season(season6,'season 6',2013)])
sixes.append([no_of_sixes_per_season(season7,'season 7',2014)])
sixes.append([no_of_sixes_per_season(season8,'season 8',2015)])
sixes.append([no_of_sixes_per_season(season9,'season 9',2016)])
sixes.append([no_of_sixes_per_season(season10,'season 10',2017)])
sixes.append([no_of_sixes_per_season(season11,'season 11',2018)])
sixes.append([no_of_sixes_per_season(season12,'season 12',2019)])
sixes.append([no_of_sixes_per_season(season13,'season 13',2020)])
print('----------------------------------------------------------------------')
```

```
*************** Count of sixes scored in each season********************
----------------------------------------------------------------------
season 1 (i.e., year 2008 )  : 623
----------------------------------------------------------------------
season 2 (i.e., year 2009 )  : 506
----------------------------------------------------------------------
season 3 (i.e., year 2010 )  : 585
----------------------------------------------------------------------
season 4 (i.e., year 2011 )  : 639
----------------------------------------------------------------------
season 5 (i.e., year 2012 )  : 733
----------------------------------------------------------------------
season 6 (i.e., year 2013 )  : 675
----------------------------------------------------------------------
season 7 (i.e., year 2014 )  : 714
----------------------------------------------------------------------
season 8 (i.e., year 2015 )  : 692
----------------------------------------------------------------------
season 9 (i.e., year 2016 )  : 639
----------------------------------------------------------------------
season 10 (i.e., year 2017 )  : 705
----------------------------------------------------------------------
season 11 (i.e., year 2018 )  : 872
----------------------------------------------------------------------
season 12 (i.e., year 2019 )  : 784
----------------------------------------------------------------------
season 13 (i.e., year 2020 )  : 735
----------------------------------------------------------------------
```

In [204]:
```
boundary=df[(df['non_boundary'] == 0) & ((df['batsman_runs'] == 4) | (df['batsman_runs'] == 6))]
```

In [205]:
```
boundary=boundary.reset_index()
```

In [206]:
```
boundary
```

Out[206]:

	index	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	batting_team	bowl
0	18	335982	1	9	5	BB McCullum	RT Ponting	SB Joshi	6	0	6	0	0	Kolkata Knight Riders	C
1	21	335982	1	10	2	RT Ponting	BB McCullum	JH Kallis	4	0	4	0	0	Kolkata Knight Riders	C
2	23	335982	1	10	4	RT Ponting	BB McCullum	JH Kallis	6	0	6	0	0	Kolkata Knight Riders	C
3	27	335982	1	11	2	BB McCullum	RT Ponting	SB Joshi	6	0	6	0	0	Kolkata Knight Riders	C
4	33	335982	1	12	2	BB McCullum	DJ Hussey	JH Kallis	4	0	4	0	0	Kolkata Knight Riders	C
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
30789	193420	1237181	1	5	5	SS Iyer	RR Pant	NM Coulter-Nile	4	0	4	0	0	Delhi Capitals	
30790	193441	1237181	1	9	2	RR Pant	SS Iyer	KH Pandya	6	0	6	0	0	Delhi Capitals	
30791	193444	1237181	1	9	5	RR Pant	SS Iyer	KH Pandya	6	0	6	0	0	Delhi Capitals	
30792	193454	1237181	1	11	3	RR Pant	SS Iyer	KA Pollard	4	0	4	0	0	Delhi Capitals	
30793	193458	1237181	1	11	7	SS Iyer	RR Pant	KA Pollard	6	0	6	0	0	Delhi Capitals	

30794 rows × 15 columns

In [207]:
```
def no_of_boundary_per_season(lst,season,year):

    add=0
    for i,row in boundary.iterrows():
        if(row['id']>= min(lst) and row['id']<=max(lst)):
            add = add + row['batsman_runs']
    print('-----------------------------------------------------------------------')
    print(season,'(i.e., year',year,')',' :', add)
```

In [208]:
```
print('*************** Count of runs scored from boundaries in each season********************')
runs =[]
runs.append([no_of_boundary_per_season(season1,'season 1',2008)])
runs.append([no_of_boundary_per_season(season2,'season 2',2009)])
runs.append([no_of_boundary_per_season(season3,'season 3',2010)])
runs.append([no_of_boundary_per_season(season4,'season 4',2011)])
runs.append([no_of_boundary_per_season(season5,'season 5',2012)])
runs.append([no_of_boundary_per_season(season6,'season 6',2013)])
runs.append([no_of_boundary_per_season(season7,'season 7',2014)])
runs.append([no_of_boundary_per_season(season8,'season 8',2015)])
runs.append([no_of_boundary_per_season(season9,'season 9',2016)])
runs.append([no_of_boundary_per_season(season10,'season 10',2017)])
runs.append([no_of_boundary_per_season(season11,'season 11',2018)])
runs.append([no_of_boundary_per_season(season12,'season 12',2019)])
runs.append([no_of_boundary_per_season(season13,'season 13',2020)])
print('-----------------------------------------------------------------------')
```

```
*************** Count of runs scored from boundaries in each season********************
-----------------------------------------------------------------------
season 1 (i.e., year 2008 )  : 10544
-----------------------------------------------------------------------
season 2 (i.e., year 2009 )  : 8300
-----------------------------------------------------------------------
season 3 (i.e., year 2010 )  : 10342
-----------------------------------------------------------------------
season 4 (i.e., year 2011 )  : 11486
-----------------------------------------------------------------------
season 5 (i.e., year 2012 )  : 12030
-----------------------------------------------------------------------
season 6 (i.e., year 2013 )  : 12248
-----------------------------------------------------------------------
season 7 (i.e., year 2014 )  : 10532
-----------------------------------------------------------------------
season 8 (i.e., year 2015 )  : 10580
-----------------------------------------------------------------------
season 9 (i.e., year 2016 )  : 10356
-----------------------------------------------------------------------
season 10 (i.e., year 2017 )  : 10666
-----------------------------------------------------------------------
season 11 (i.e., year 2018 )  : 11836
-----------------------------------------------------------------------
season 12 (i.e., year 2019 )  : 11316
-----------------------------------------------------------------------
season 13 (i.e., year 2020 )  : 10732
-----------------------------------------------------------------------
```

30] What is the run contribution from boundaries in each season?

In [209]:
```
def boundary_run_contribution_per_season(lst,season,year):

    add=0
    total_runs_per_season=0
    for i,row in boundary.iterrows():
        if(row['id']>= min(lst) and row['id']<=max(lst)):
            add = add + row['batsman_runs']
    for i,row in df1.iterrows():
        if(row['Year']==year):
            total_runs_per_season = total_runs_per_season+row['runs_per_match']
            boundary_contribution_per_season = (add / total_runs_per_season) * 100
    print('-------------------------------------------------------------------------------')
    print('The run contibution from boundaries in each season is')
    print(season,'(i.e.,year,',year,' ) : ',boundary_contribution_per_season,'%')
```

In [210]:
```
print('********** Count of runs contibution from boundaries in each season**********')
contribution =[]
contribution.append([boundary_run_contribution_per_season(season1,'season 1',2008)])
contribution.append([boundary_run_contribution_per_season(season2,'season 2',2009)])
contribution.append([boundary_run_contribution_per_season(season3,'season 3',2010)])
contribution.append([boundary_run_contribution_per_season(season4,'season 4',2011)])
contribution.append([boundary_run_contribution_per_season(season5,'season 5',2012)])
contribution.append([boundary_run_contribution_per_season(season6,'season 6',2013)])
contribution.append([boundary_run_contribution_per_season(season7,'season 7',2014)])
contribution.append([boundary_run_contribution_per_season(season8,'season 8',2015)])
contribution.append([boundary_run_contribution_per_season(season9,'season 9',2016)])
contribution.append([boundary_run_contribution_per_season(season10,'season 10',2017)])
contribution.append([boundary_run_contribution_per_season(season11,'season 11',2018)])
contribution.append([boundary_run_contribution_per_season(season12,'season 12',2019)])
contribution.append([boundary_run_contribution_per_season(season13,'season 13',2020)])
print('--------------------------------------------------------------------------------')
```

********** Count of runs contibution from boundaries in each season**********

The run contibution from boundaries in each season is
season 1 (i.e.,year, 2008) : 58.7835200981212 %

The run contibution from boundaries in each season is
season 2 (i.e.,year, 2009) : 50.857843137254896 %

The run contibution from boundaries in each season is
season 3 (i.e.,year, 2010) : 54.82400339270568 %

The run contibution from boundaries in each season is
season 4 (i.e.,year, 2011) : 54.4411792586975 %

The run contibution from boundaries in each season is
season 5 (i.e.,year, 2012) : 53.57858638043914 %

The run contibution from boundaries in each season is
season 6 (i.e.,year, 2013) : 54.33654230069651 %

The run contibution from boundaries in each season is
season 7 (i.e.,year, 2014) : 55.698344703580304 %

The run contibution from boundaries in each season is
season 8 (i.e.,year, 2015) : 57.71328823914467 %

The run contibution from boundaries in each season is
season 9 (i.e.,year, 2016) : 54.904039868518716 %

The run contibution from boundaries in each season is
season 10 (i.e.,year, 2017) : 56.8277478821461 %

The run contibution from boundaries in each season is
season 11 (i.e.,year, 2018) : 59.474398271443654 %

The run contibution from boundaries in each season is
season 12 (i.e.,year, 2019) : 58.329896907216494 %

The run contibution from boundaries in each season is
season 13 (i.e.,year, 2020) : 55.456800330715176 %

31] Which team has scored the most runs in first 6 overs?

In [211]:
```
first_6_overs=df[df['over']<=6]
```

In [212]:
```
runs_in_first_6_overs=first_6_overs.groupby('batting_team')['batsman_runs'].sum()
```

In [213]:
```
most_runs_in_first_6_overs = runs_in_first_6_overs.sort_values(ascending=False).reset_index()
```

In [214]:
```
most_runs_in_first_6_overs
```

Out[214]:

	batting_team	batsman_runs
0	Mumbai Indians	9734
1	Kings XI Punjab	9677
2	Kolkata Knight Riders	9494
3	Royal Challengers Bangalore	9357
4	Chennai Super Kings	8732
5	Delhi Daredevils	7957
6	Rajasthan Royals	7750
7	Sunrisers Hyderabad	6308
8	Deccan Chargers	3651
9	Pune Warriors	2008
10	Delhi Capitals	1718
11	Gujarat Lions	1717
12	Rising Pune Supergiant	835
13	Rising Pune Supergiants	709
14	Kochi Tuskers Kerala	697

In [215]:
```
for i,row in most_runs_in_first_6_overs.iterrows():
    print('-------------------------------------------------------')
    print('The team which has scored most runs in first 6 overs is')
    print(row['batting_team'] ,':',row['batsman_runs'])
    print('-------------------------------------------------------')
    break
```

```
-------------------------------------------------------
The team which has scored most runs in first 6 overs is
Mumbai Indians : 9734
-------------------------------------------------------
```

32] Which team has scored the most runs in last 4 overs?

In [216]:
```
last_4_overs=df[df['over']>=17]
```

In [217]:
```
runs_in_last_4_overs=last_4_overs.groupby('batting_team')['batsman_runs'].sum()
```

In [218]:
```
most_runs_in_last_4_overs = runs_in_last_4_overs.sort_values(ascending=False).reset_index()
```

In [219]:
```
most_runs_in_last_4_overs
```

Out[219]:

	batting_team	batsman_runs
0	Mumbai Indians	5410
1	Royal Challengers Bangalore	4843
2	Chennai Super Kings	4784
3	Kings XI Punjab	4398
4	Kolkata Knight Riders	4277
5	Delhi Daredevils	3524
6	Rajasthan Royals	3453
7	Sunrisers Hyderabad	3024
8	Deccan Chargers	1738
9	Pune Warriors	993
10	Delhi Capitals	800
11	Gujarat Lions	623
12	Rising Pune Supergiant	412
13	Rising Pune Supergiants	312
14	Kochi Tuskers Kerala	236

In [220]:
```
for i,row in most_runs_in_last_4_overs.iterrows():
    print('-------------------------------------------------')
    print('The team which has scored most runs in last 4 overs is')
    print(row['batting_team'] ,':',row['batsman_runs'])
    print('-------------------------------------------------')
    break
```

```
-------------------------------------------------
The team which has scored most runs in last 4 overs is
Mumbai Indians : 5410
-------------------------------------------------
```

33] Which team has the best scoring run-rate in the first 6 overs ?

In [221]:
```
first_6_overs=df[df['over']<=6]
```

In [222]:
```
runs_in_first_6_overs=first_6_overs.groupby('batting_team')['batsman_runs'].sum().reset_index()
```

In [223]:
```
most_runs_in_first_6_overs = runs_in_first_6_overs.sort_values(by='batsman_runs',ascending=False).reset_index()
```

In [224]:
```
most_runs_in_first_6_overs
```

Out[224]:

	index	batting_team	batsman_runs
0	8	Mumbai Indians	9734
1	5	Kings XI Punjab	9677
2	7	Kolkata Knight Riders	9494
3	13	Royal Challengers Bangalore	9357
4	0	Chennai Super Kings	8732
5	3	Delhi Daredevils	7957
6	10	Rajasthan Royals	7750
7	14	Sunrisers Hyderabad	6308
8	1	Deccan Chargers	3651
9	9	Pune Warriors	2008
10	2	Delhi Capitals	1718
11	4	Gujarat Lions	1717
12	11	Rising Pune Supergiant	835
13	12	Rising Pune Supergiants	709
14	6	Kochi Tuskers Kerala	697

In [225]:
```
unique_teams=[]
for i,row in most_runs_in_first_6_overs.iterrows():
    unique_teams.append(row['batting_team'])
```

In [226]:
```
def run_rate_6(team):
    count=0
    for i,row in first_6_overs.iterrows():
        for i in unique_teams:
            if (row['batting_team'] == team):
                count=count+ 1
    return count
```

In [227]:
```
a=[]
for i in unique_teams:
    a.append(run_rate_6(i))
```

In [228]:
```
unique_teams
```

Out[228]:
```
['Mumbai Indians',
 'Kings XI Punjab',
 'Kolkata Knight Riders',
 'Royal Challengers Bangalore',
 'Chennai Super Kings',
 'Delhi Daredevils',
 'Rajasthan Royals',
 'Sunrisers Hyderabad',
 'Deccan Chargers',
 'Pune Warriors',
 'Delhi Capitals',
 'Gujarat Lions',
 'Rising Pune Supergiant',
 'Rising Pune Supergiants',
 'Kochi Tuskers Kerala']
```

In [229]:
```
a
```

Out[229]:
```
[133485,
 124200,
 125565,
 126480,
 116220,
 105060,
 103995,
 80880,
 49170,
 29370,
 21405,
 19635,
 10455,
 9135,
 9330]
```

In [230]:
```
rate=[]
for i,row in most_runs_in_first_6_overs.iterrows():
    for i in a:
        run_rate=row['batsman_runs']/i
        rate.append(run_rate)
        break
```

In [231]:
```
print('------------------------------------------------------------------')
print('The team which has the best scoring run rate in first 6 overs is')
print(unique_teams[0],':',max(rate))
print('------------------------------------------------------------------')
```

```
------------------------------------------------------------------
The team which has the best scoring run rate in first 6 overs is
Mumbai Indians : 0.07292205116679777
------------------------------------------------------------------
```

34] Which team has the best scoring run-rate in the last 4 overs ?

In [232]:
```
last_4_overs=df[df['over']>=17]
```

In [233]:
```
runs_in_last_4_overs=last_4_overs.groupby('batting_team')['batsman_runs'].sum()
```

In [234]:
```
most_runs_in_last_4_overs = runs_in_last_4_overs.sort_values(ascending=False).reset_index()
```

In [235]:
```
most_runs_in_last_4_overs
```

Out[235]:

	batting_team	batsman_runs
0	Mumbai Indians	5410
1	Royal Challengers Bangalore	4843
2	Chennai Super Kings	4784
3	Kings XI Punjab	4398
4	Kolkata Knight Riders	4277
5	Delhi Daredevils	3524
6	Rajasthan Royals	3453
7	Sunrisers Hyderabad	3024
8	Deccan Chargers	1738
9	Pune Warriors	993
10	Delhi Capitals	800
11	Gujarat Lions	623
12	Rising Pune Supergiant	412
13	Rising Pune Supergiants	312
14	Kochi Tuskers Kerala	236

In [236]:
```
unique_teams=[]
for i,row in most_runs_in_last_4_overs.iterrows():
    unique_teams.append(row['batting_team'])
```

In [237]:
```
def run_rate_4(team):
    count=0
    for i,row in last_4_overs.iterrows():
        for i in unique_teams:
            if (row['batting_team'] == team):
                count=count+ 1
    return count
```

In [238]:
```
a1=[]
for i in unique_teams:
    a1.append(run_rate_4(i))
```

In [239]:
```
unique_teams
```

Out[239]:
```
['Mumbai Indians',
 'Royal Challengers Bangalore',
 'Chennai Super Kings',
 'Kings XI Punjab',
 'Kolkata Knight Riders',
 'Delhi Daredevils',
 'Rajasthan Royals',
 'Sunrisers Hyderabad',
 'Deccan Chargers',
 'Pune Warriors',
 'Delhi Capitals',
 'Gujarat Lions',
 'Rising Pune Supergiant',
 'Rising Pune Supergiants',
 'Kochi Tuskers Kerala']
```

In [240]:
```
a1
```

Out[240]:
```
[49110,
 43140,
 43110,
 44550,
 41880,
 33780,
 35790,
 29505,
 18255,
 10950,
 8220,
 6570,
 3900,
 2790,
 2610]
```

In [241]:
```
rate1=[]
for i,row in most_runs_in_last_4_overs.iterrows():
    for i in a1:
        run_rate=row['batsman_runs']/i
        rate1.append(run_rate)
        break
```

In [242]:
```
print('-----------------------------------------------------------------')
print('The team which has the best scoring run rate in last 4 overs is')
print(unique_teams[0],':',max(rate1))
print('-----------------------------------------------------------------')
```

```
-----------------------------------------------------------------
The team which has the best scoring run rate in last 4 overs is
Mumbai Indians : 0.1101608633679495
-----------------------------------------------------------------
```