Author: Varun SA

Task - 5 Exploratory Data Analysis

Graduate Rotational Internship Program @ THE SPARKS FOUNDATIONS

Carried out exploratory data analysis with the given sports dataset 'Indian Premier League' to find the following

- The most successful team/player
- Factors contributing win or loss
- Present with suggestions to companies in endorsing a player or a team

Technical Requirements

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Loading the dataset

In [2]:
```
matches=pd.read_csv(r"C:\Users\Varun\Desktop\GRIP\datasets\matches.csv")
print('Dataset Loaded')
```

```
Dataset Loaded
```

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

Out[3]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	NaN
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD Smith	Maharashtra Cricket Association Stadium	A Nand Kishore	S Ravi	NaN
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA Lynn	Saurashtra Cricket Association Stadium	Nitin Menon	CK Nandan	NaN
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Maxwell	Holkar Cricket Stadium	AK Chaudhary	C Shamshuddin	NaN
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Jadhav	M Chinnaswamy Stadium	NaN	NaN	NaN

In [4]:
```
matches.tail()
```

Out[4]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3
751	11347	2019	Mumbai	05/05/19	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumbai Indians	0	9	HH Pandya	Wankhede Stadium	Nanda Kishore	O Nandan	S Ravi
752	11412	2019	Chennai	07/05/19	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumbai Indians	0	6	AS Yadav	M. A. Chidambaram Stadium	Nigel Llong	Nitin Menon	Ian Gould
753	11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	0	Delhi Capitals	0	2	RR Pant	ACA-VDCA Stadium	NaN	NaN	NaN
754	11414	2019	Visakhapatnam	10/05/19	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chennai Super Kings	0	6	F du Plessis	ACA-VDCA Stadium	Sundaram Ravi	Bruce Oxenford	Chettithody Shamshuddin
755	11415	2019	Hyderabad	12/05/19	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	Mumbai Indians	1	0	JJ Bumrah	Rajiv Gandhi Intl. Cricket Stadium	Nitin Menon	Ian Gould	Nigel Llong

In [5]:
```
matches.describe()
```

Out[5]:

	id	season	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000	756.000000	756.000000
mean	1792.178571	2013.444444	0.025132	13.283069	3.350529
std	3464.478148	3.366895	0.156630	23.471144	3.387963
min	1.000000	2008.000000	0.000000	0.000000	0.000000
25%	189.750000	2011.000000	0.000000	0.000000	0.000000
50%	378.500000	2013.000000	0.000000	0.000000	4.000000
75%	567.250000	2016.000000	0.000000	19.000000	6.000000
max	11415.000000	2019.000000	1.000000	146.000000	10.000000

In [6]:
```
matches.shape
```

Out[6]:
```
(756, 18)
```

Loading the second dataset

In [7]:
```
deliveries=pd.read_csv(r"C:\Users\Varun\Desktop\GRIP\datasets\deliveries.csv")
```

In [8]:
```
deliveries.head()
```

Out[8]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs	player_dismissed	dismissal_kind	fielder
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	0	0	0	NaN	NaN	NaN
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	0	0	0	NaN	NaN	NaN
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	4	0	4	NaN	NaN	NaN
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	0	0	0	NaN	NaN	NaN
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	0	2	2	NaN	NaN	NaN

5 rows × 21 columns

In [9]:
```
deliveries.tail()
```

Out[9]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs	player_dismissed	dismissal_kind	fielder
179073	11415	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	SR Watson	SL Malinga	0	...	0	0	0	0	1	0	1	NaN	NaN	NaN
179074	11415	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	RA Jadeja	SL Malinga	0	...	0	0	0	0	2	0	2	NaN	NaN	NaN
179075	11415	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	RA Jadeja	SL Malinga	0	...	0	0	0	0	1	0	1	SR Watson	run out	KH Pandya
179076	11415	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	RA Jadeja	SL Malinga	0	...	0	0	0	0	2	0	2	NaN	NaN	NaN
179077	11415	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	RA Jadeja	SL Malinga	0	...	0	0	0	0	0	0	0	SN Thakur	lbw	NaN

5 rows × 21 columns

In [10]:
```
deliveries.describe()
```

Out[10]:

	match_id	inning	over	ball	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000
mean	1802.252957	1.482952	10.162488	3.615587	0.000452	0.036721	0.004936	0.021136	0.004183	0.000056	1.246864	0.067032	1.313897
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.251161	0.116480	0.194908	0.070492	0.016709	1.608270	0.342553	1.605422
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.000000	4.000000	5.000000	5.000000	5.000000	7.000000	7.000000	10.000000

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

Out[11]:
```
(179078, 21)
```

Merging the two datasets

In [12]:
```
#merging the two datasets
merge=pd.merge(matches, deliveries, left_on='id', right_on='match_id')
merge.head(2)
```

Out[12]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	...	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs	player_dismissed	dismissal_kind	fielder
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	...	0	0	0	0	0	0	0	NaN	NaN	NaN
1	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	...	0	0	0	0	0	0	0	NaN	NaN	NaN

2 rows × 39 columns

In [13]:
```
merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 179078 entries, 0 to 179077
Data columns (total 39 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                179078 non-null  int64 
 1   season            179078 non-null  int64 
 2   city              177378 non-null  object
 3   date              179078 non-null  object
 4   team1             179078 non-null  object
 5   team2             179078 non-null  object
 6   toss_winner       179078 non-null  object
 7   toss_decision     179078 non-null  object
 8   result            179078 non-null  object
 9   dl_applied        179078 non-null  int64 
 10  winner            178706 non-null  object
 11  win_by_runs       179078 non-null  int64
 12  win_by_wickets    179078 non-null  int64
 13  player_of_match   178706 non-null  object
 14  venue             179078 non-null  object
 15  umpire1           178578 non-null  object
 16  umpire2           178578 non-null  object
 17  umpire3           28366 non-null   object
 18  match_id          179078 non-null  int64
 19  inning            179078 non-null  int64
 20  batting_team      179078 non-null  object
 21  bowling_team      179078 non-null  object
 22  over              179078 non-null  int64
 23  ball              179078 non-null  int64
 24  batsman           179078 non-null  object
 25  non_striker       179078 non-null  object
 26  bowler            179078 non-null  object
 27  is_super_over     179078 non-null  int64
 28  wide_runs         179078 non-null  int64
 29  bye_runs          179078 non-null  int64
 30  legbye_runs       179078 non-null  int64
 31  noball_runs       179078 non-null  int64
 32  penalty_runs      179078 non-null  int64
 33  batsman_runs      179078 non-null  int64
 34  extra_runs        179078 non-null  int64
 35  total_runs        179078 non-null  int64
 36  player_dismissed  8834 non-null    object
 37  dismissal_kind    8834 non-null    object
 38  fielder           6448 non-null    object
dtypes: int64(18), object(21)
memory usage: 54.7+ MB
```

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

Out[14]:

	id	season	dl_applied	win_by_runs	win_by_wickets	match_id	inning	over	ball	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000
mean	1802.252957	2013.444510	0.017914	13.404036	3.261579	1802.252957	1.482952	10.162488	3.615587	0.000452	0.036721	0.004936	0.021136	0.004183	0.000056	1.246864	0.067032	1.313897
std	3472.322805	3.363947	0.132639	23.261007	3.347033	3472.322805	0.502074	5.677684	1.806966	0.021263	0.251161	0.116480	0.194908	0.070492	0.016709	1.608270	0.342553	1.605422
min	1.000000	2008.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	190.000000	2011.000000	0.000000	0.000000	0.000000	190.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	379.000000	2013.000000	0.000000	0.000000	3.000000	379.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
75%	567.000000	2016.000000	0.000000	19.000000	6.000000	567.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
max	11415.000000	2019.000000	1.000000	146.000000	10.000000	11415.000000	5.000000	20.000000	9.000000	1.000000	5.000000	4.000000	5.000000	5.000000	5.000000	7.000000	7.000000	10.000000

Preprocessing the Dataset

In [15]:
```
matches.head(4)
```

Out[15]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	NaN
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD Smith	Maharashtra Cricket Association Stadium	A Nand Kishore	S Ravi	NaN
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA Lynn	Saurashtra Cricket Association Stadium	Nitin Menon	CK Nandan	NaN
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Maxwell	Holkar Cricket Stadium	AK Chaudhary	C Shamshuddin	NaN

Error and missing values in the dataset

The missing values are

- umpire 1 & umpire 2 have missing values
- umpire 3 has a missing value of 94%
- missing values in winners and player of the match
- one distinct values missing from Team 1 and Team 2

In [16]:
```
matches[matches.city.isnull()][['city','venue']]
```

Out[16]:

	city	venue
461	NaN	Dubai International Cricket Stadium
462	NaN	Dubai International Cricket Stadium
466	NaN	Dubai International Cricket Stadium
468	NaN	Dubai International Cricket Stadium
469	NaN	Dubai International Cricket Stadium
474	NaN	Dubai International Cricket Stadium
476	NaN	Dubai International Cricket Stadium

- Heance the missing values of city can be replaced with Dubai

In [17]:
```
matches.city = matches.city.fillna('Dubai')
```

In [18]:
```
matches[(matches.umpire1.isnull())| (matches.umpire2.isnull())]
```

Out[18]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Jadhav	M Chinnaswamy Stadium	NaN	NaN	NaN
753	11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	0	Delhi Capitals	0	2	RR Pant	ACA-VDCA Stadium	NaN	NaN	NaN

In [19]:
```
matches=matches.drop('umpire3', axis=1)
```

- umpire 3 has been dropped since it has high missing values

In [20]:
```
city_venue= matches.groupby(['city','venue']).count()['season']
city_venue_df=pd.DataFrame(city_venue)
city_venue_df
```

Out[20]:

		season
city	**venue**	
Abu Dhabi	**Sheikh Zayed Stadium**	7
Ahmedabad	**Sardar Patel Stadium, Motera**	12
Bangalore	**M Chinnaswamy Stadium**	66
Bengaluru	**M Chinnaswamy Stadium**	7
	M. Chinnaswamy Stadium	7
Bloemfontein	**OUTsurance Oval**	2
Cape Town	**Newlands**	7
Centurion	**SuperSport Park**	12
Chandigarh	**Punjab Cricket Association IS Bindra Stadium, Mohali**	11
	Punjab Cricket Association Stadium, Mohali	35
Chennai	**M. A. Chidambaram Stadium**	8
	MA Chidambaram Stadium, Chepauk	49
Cuttack	**Barabati Stadium**	7
Delhi	**Feroz Shah Kotla**	67
	Feroz Shah Kotla Ground	7
Dharamsala	**Himachal Pradesh Cricket Association Stadium**	9
Dubai	**Dubai International Cricket Stadium**	7
Durban	**Kingsmead**	15
East London	**Buffalo Park**	3
Hyderabad	**Rajiv Gandhi International Stadium, Uppal**	56
	Rajiv Gandhi Intl. Cricket Stadium	8
Indore	**Holkar Cricket Stadium**	9
Jaipur	**Sawai Mansingh Stadium**	47
Johannesburg	**New Wanderers Stadium**	8
Kanpur	**Green Park**	4
Kimberley	**De Beers Diamond Oval**	3
Kochi	**Nehru Stadium**	5
Kolkata	**Eden Gardens**	77
Mohali	**IS Bindra Stadium**	7
	Punjab Cricket Association IS Bindra Stadium, Mohali	3
Mumbai	**Brabourne Stadium**	11
	Dr DY Patil Sports Academy	17
	Wankhede Stadium	73
Nagpur	**Vidarbha Cricket Association Stadium, Jamtha**	3
Port Elizabeth	**St George's Park**	7
Pune	**Maharashtra Cricket Association Stadium**	21
	Subrata Roy Sahara Stadium	17
Raipur	**Shaheed Veer Narayan Singh International Stadium**	6

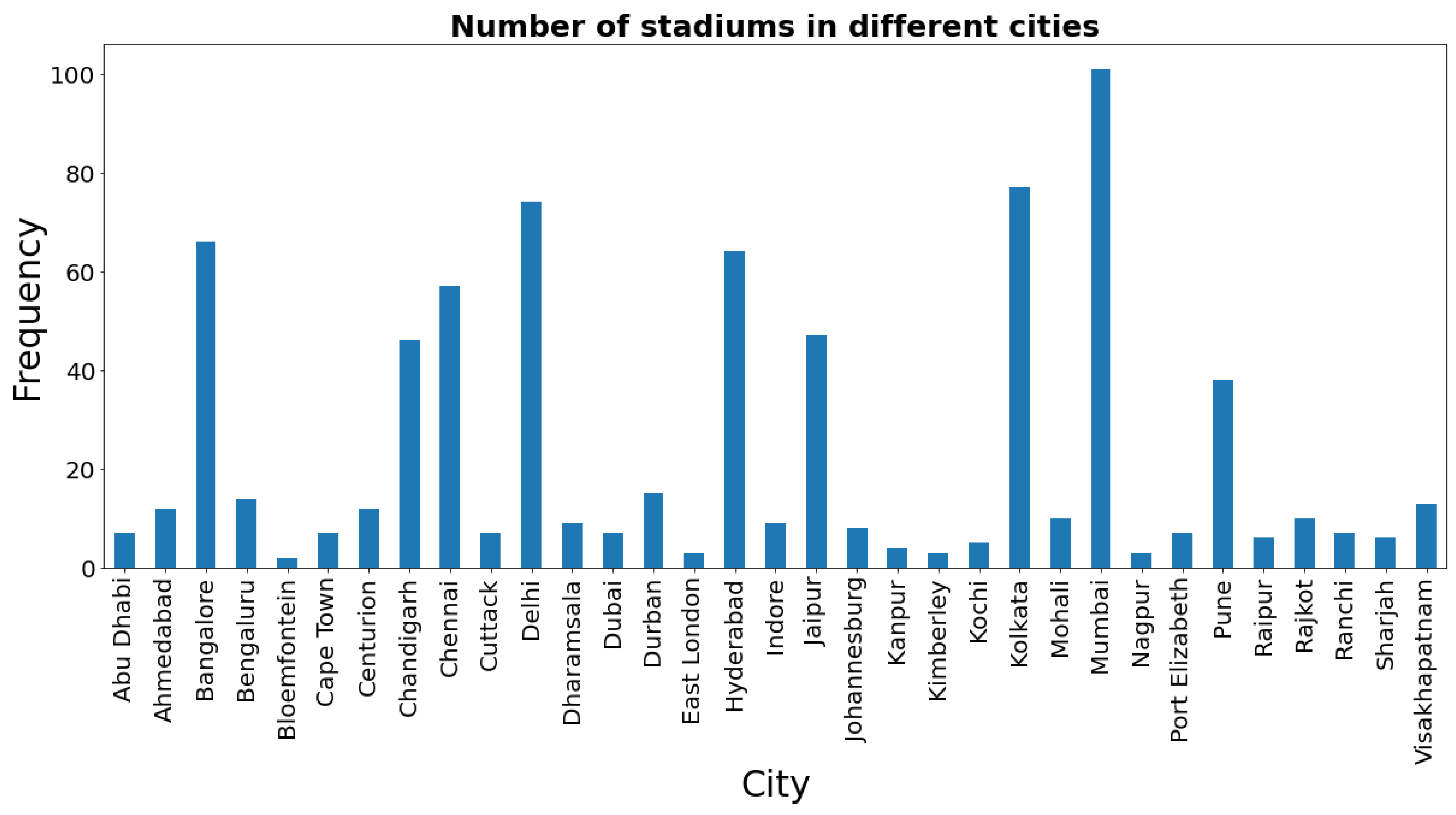
city	venue	season
Rajkot	Saurashtra Cricket Association Stadium	10
Ranchi	JSCA International Stadium Complex	7
Sharjah	Sharjah Cricket Stadium	6
Visakhapatnam	ACA-VDCA Stadium	2
	Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium	11

Observation

- repeatation observed
- new names and old names are taken as distinct values

Visualising the Dataset

In [21]:
```
#plotting the venues along with cities
v= pd.crosstab(matches['city'], matches['venue'])

#Adding a column by summing other consecutive columns
v['count']=v.sum(axis='columns')

#last column='count'
b= v['count']

#plotting the dataset
plt.figure(figsize=(20,8))
b.plot(kind='bar')
plt.title('Number of stadiums in different cities', fontsize=25, fontweight= 'bold')
plt.ylabel('Frequency',size=30)
plt.xlabel('City', size=30)
plt.yticks(size=20)
plt.xticks(size=20)
plt.show()
```

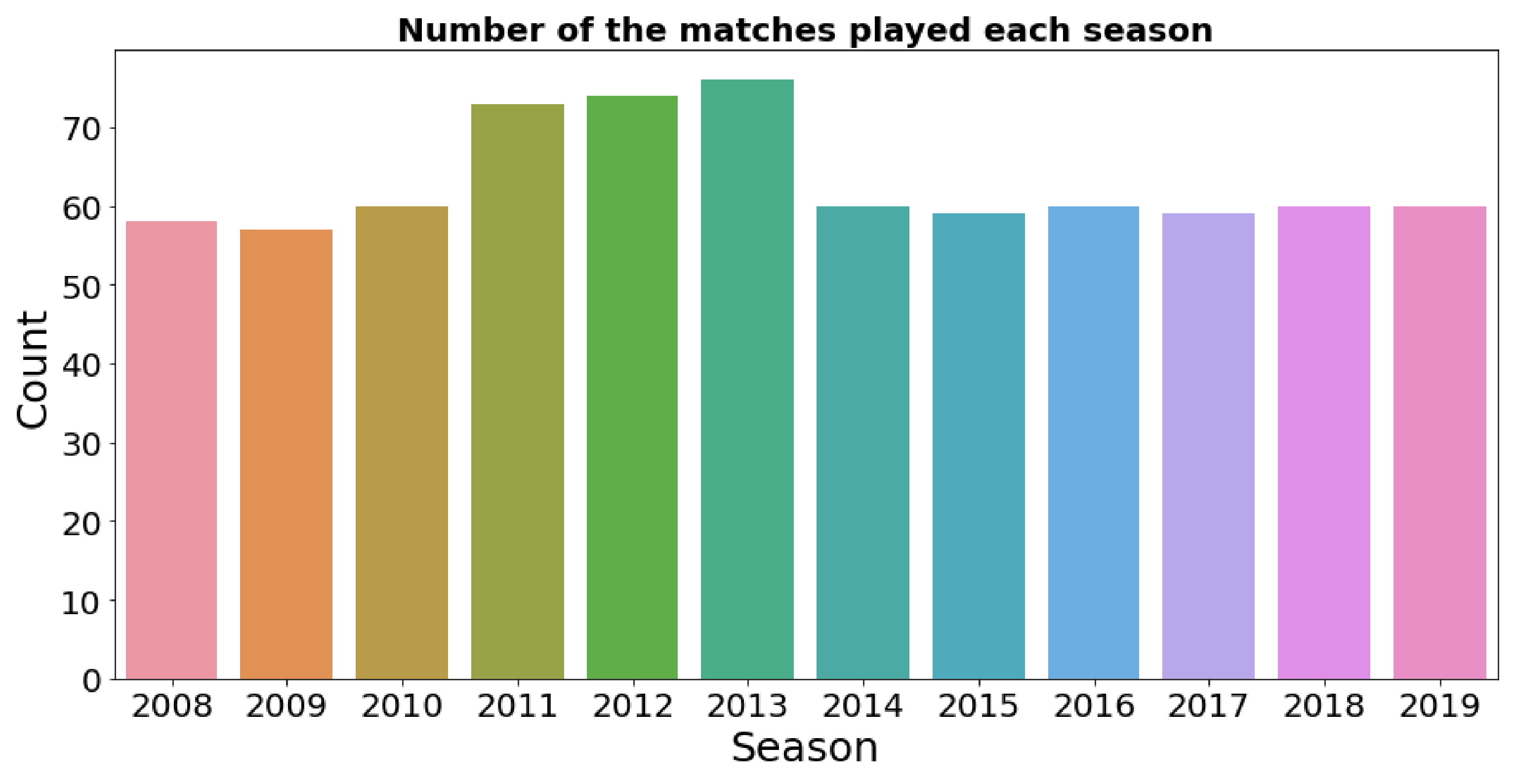


Number of matches played in each season

In [22]:
```
plt.figure(figsize=(15,7))
sns.countplot('season',data=matches)
plt.xlabel('Season',fontsize=25)
plt.ylabel('Count',size=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title('Number of the matches played each season', fontsize=20,
fontweight="bold")
```

```
C:\Users\Varun\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

Out[22]:
```
Text(0.5, 1.0, 'Number of the matches played each season')
```

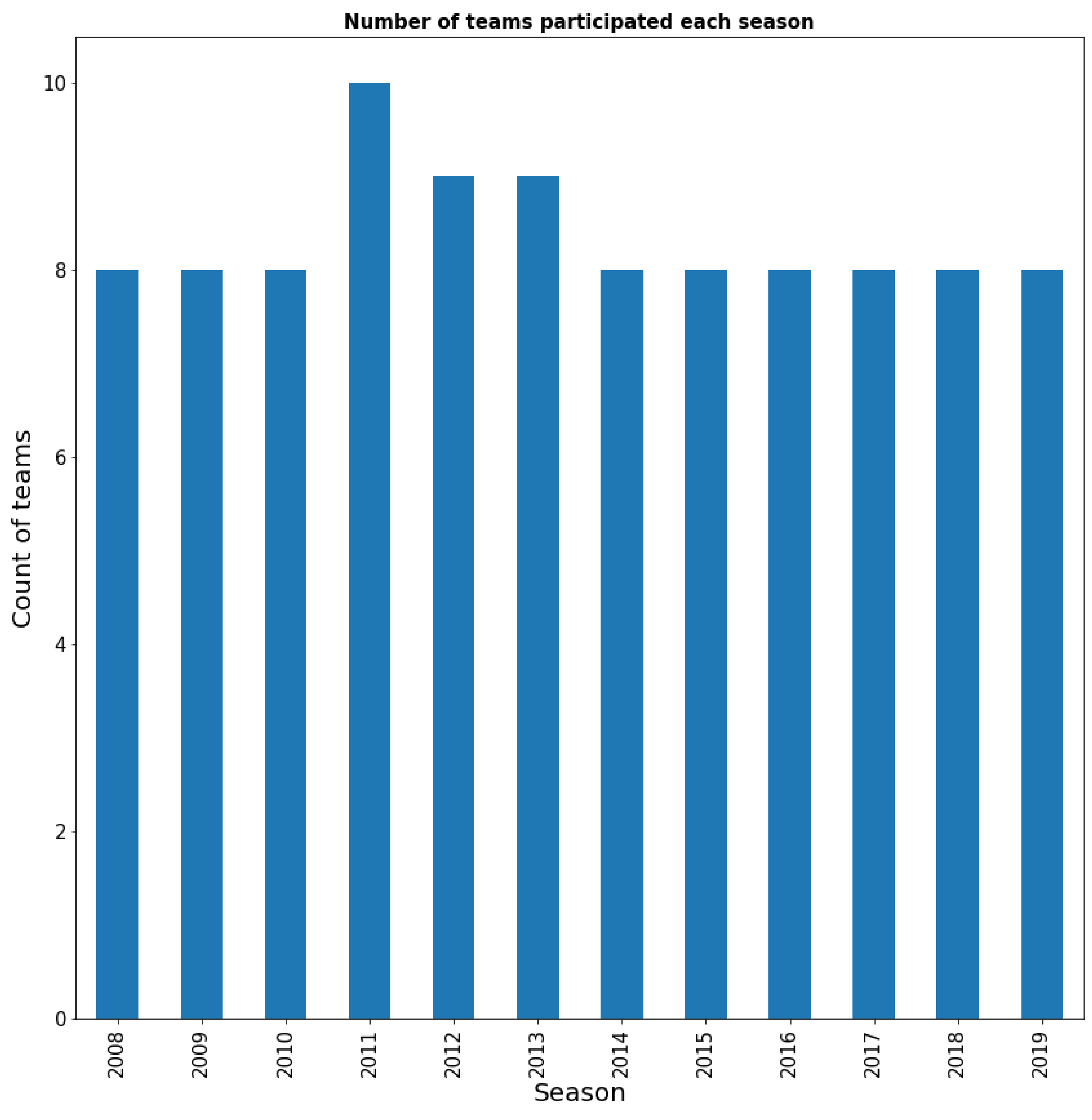


- The highest number of matches played is between the year 2011-2013

Number of Teams played in each season

In [23]:
```
matches.groupby('season')['team1'].nunique().plot(kind='bar', figsize=(14,14))
plt.title('Number of teams participated each season', fontsize=15,fontweight='bold')
plt.xlabel('Season',size=20)
plt.ylabel('Count of teams', size=20)
plt.yticks(size=15)
plt.xticks(size=15)
```

Out[23]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, '2008'),
  Text(1, 0, '2009'),
  Text(2, 0, '2010'),
  Text(3, 0, '2011'),
  Text(4, 0, '2012'),
  Text(5, 0, '2013'),
  Text(6, 0, '2014'),
  Text(7, 0, '2015'),
  Text(8, 0, '2016'),
  Text(9, 0, '2017'),
  Text(10, 0, '2018'),
  Text(11, 0, '2019')])
```

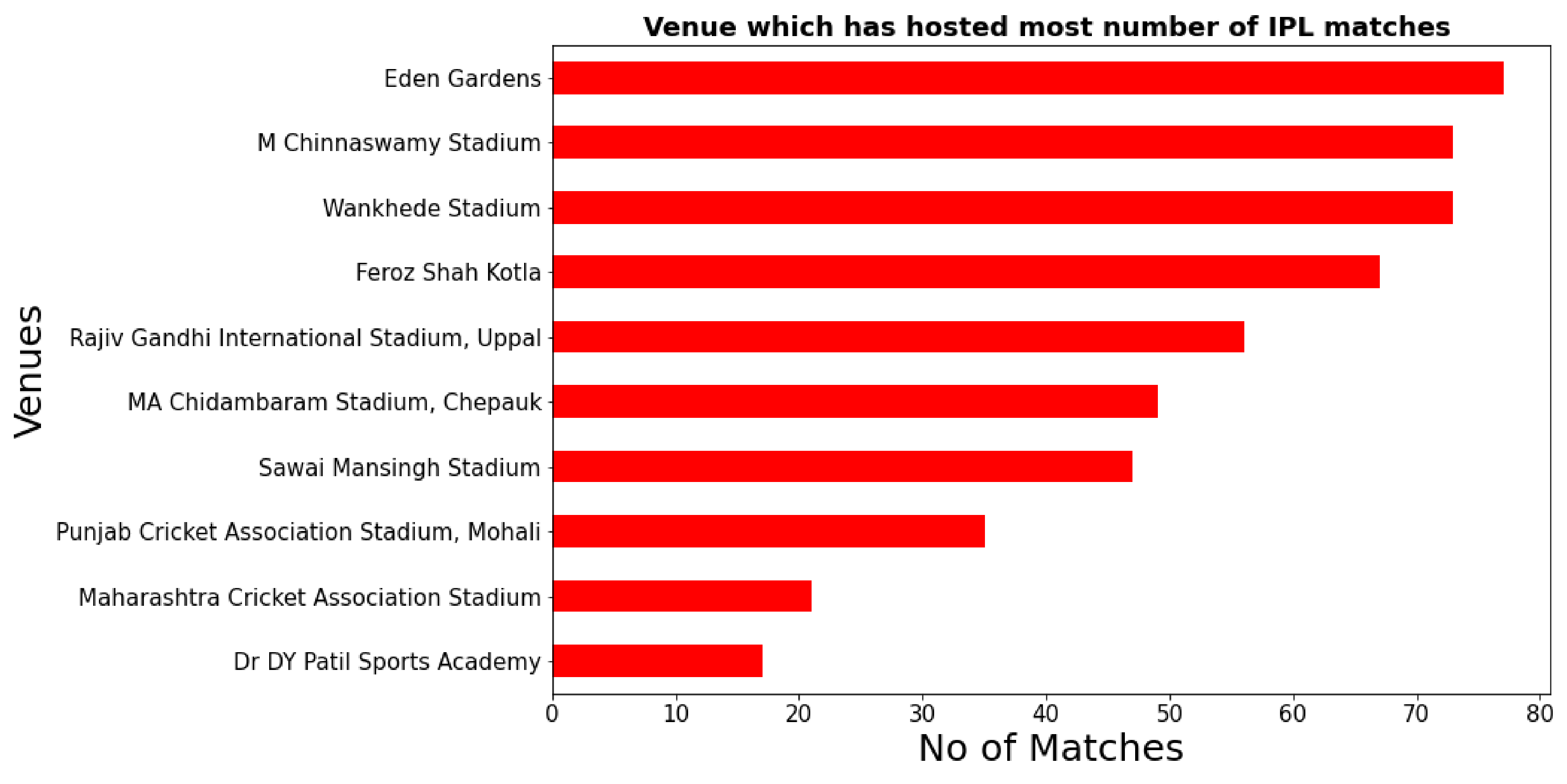


- 10 teams have played in the year 2011 and 9 teams respectively in the year 2012 & 2013

Venues with Highest Match Hostings

In [24]:
```
matches.venue.value_counts().sort_values(ascending=True).tail(10).plot(kind='barh',figsize=(12,8), fontsize=15, color='red')
plt.title('Venue which has hosted most number of IPL matches ',fontsize=18, fontweight='bold')
plt.xlabel('No of Matches', size=25)
plt.ylabel('Venues', size=25)
```

Out[24]:
```
Text(0, 0.5, 'Venues')
```



- EDEN GARDENS stadium in kolkata has hosted highest number of matches

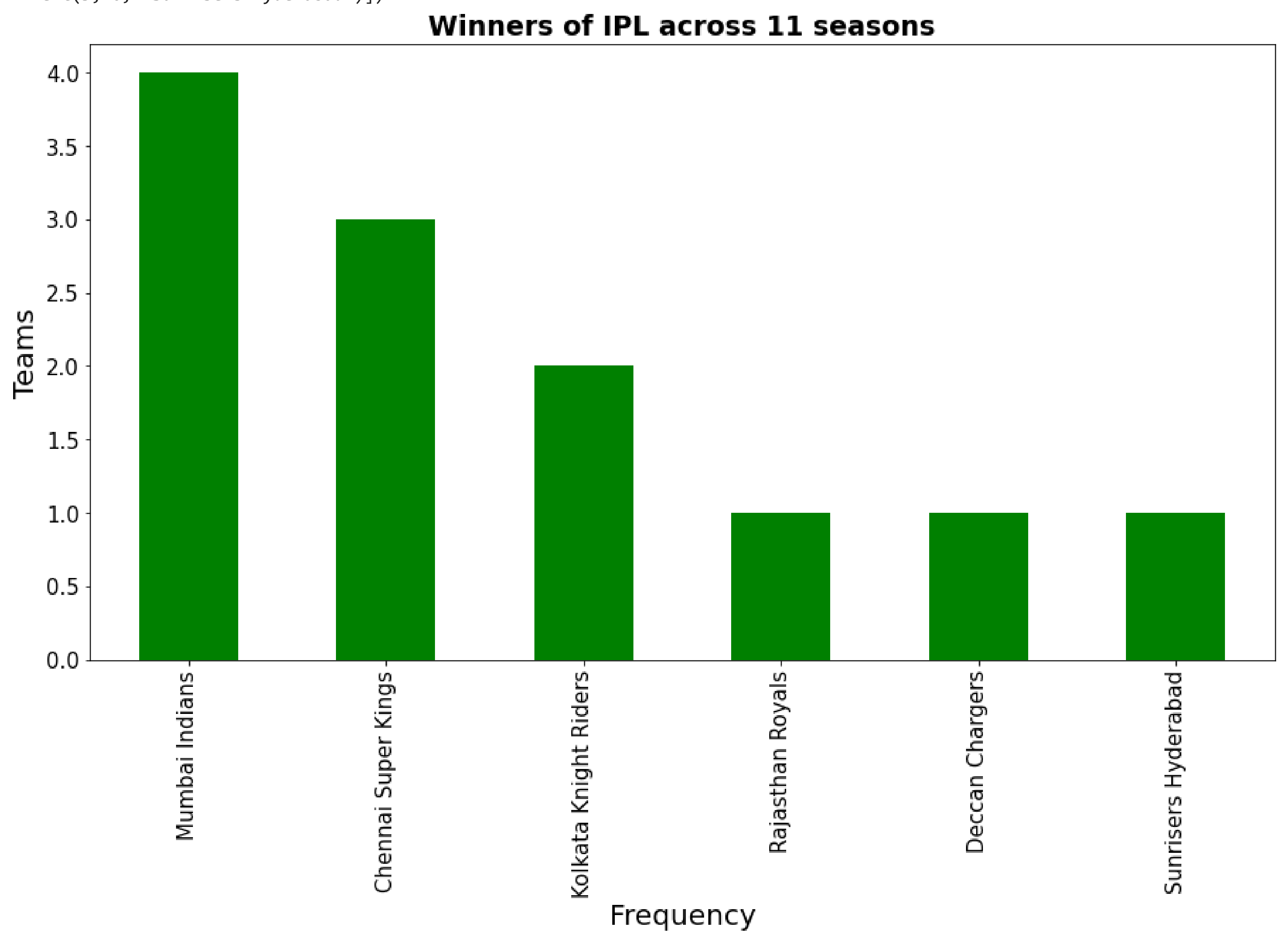
Teams with maximum number of wins in IPL

In [25]:
```
#creating a dataframe containing the season and winner columns
winning_teams = matches [['season', 'winner']]
```

In [26]:
```
winners_team={}
for i  in sorted(winning_teams.season.unique()):
    winners_team[i]=winning_teams[winning_teams.season==i]['winner'].tail(1).values[0]

winners_of_IPL = pd.Series(winners_team)
winners_of_IPL = pd.DataFrame(winners_of_IPL, columns=['team'])
```

In [27]:
```
winners_of_IPL['team'].value_counts().plot(kind='bar', figsize=(15,8), color='green')
plt.title('Winners of IPL across 11 seasons', fontsize=19,fontweight='bold')
plt.xlabel('Frequency',size=20)
plt.ylabel('Teams', size=20)
plt.yticks(size=15)
plt.xticks(size=15)
```

Out[27]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Mumbai Indians'),
  Text(1, 0, 'Chennai Super Kings'),
  Text(2, 0, 'Kolkata Knight Riders'),
  Text(3, 0, 'Rajasthan Royals'),
  Text(4, 0, 'Deccan Chargers'),
  Text(5, 0, 'Sunrisers Hyderabad')])
```

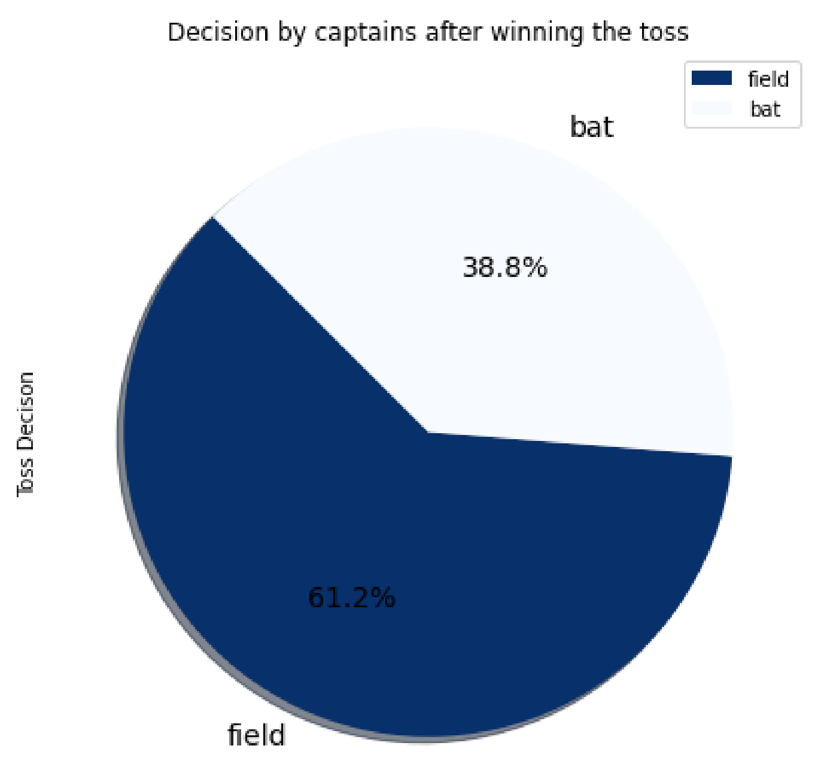


- Mumbai Indians 4 Wins
- Chennai Super Kings 3 Wins

Batting VS Fielding (Teams's Choice)

In [28]:
```
matches['toss_decision'].value_counts().plot(kind='pie', fontsize=14, autopct='%3.1f%%', figsize=(10,7), shadow = True, startangle=135, legend=True, cmap='Blues_r')

plt.ylabel('Toss Decison')
plt.title('Decision by captains after winning the toss')
```

Out[28]:
```
Text(0.5, 1.0, 'Decision by captains after winning the toss')
```



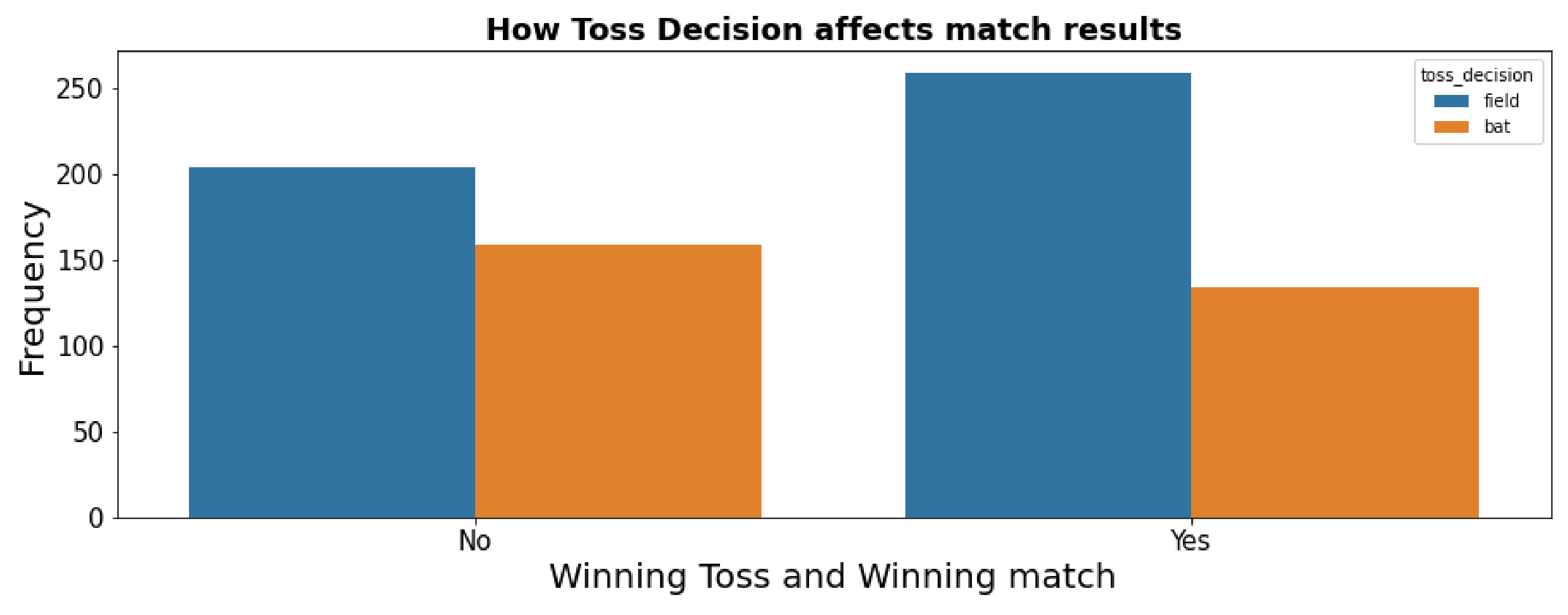
Factor affecting match results

- Toss decision
- Team decision
- player's performance

In [29]:
```
matches['toss_win_game_win']= np.where((matches.toss_winner == matches.winner),'Yes','No')
plt.figure(figsize=(15,5))
sns.countplot('toss_win_game_win', data= matches, hue='toss_decision')
plt.title('How Toss Decision affects match results', fontsize=18, fontweight='bold')
plt.xlabel('Winning Toss and Winning match',size=20)
plt.ylabel('Frequency', size=20)
plt.yticks(size=15)
plt.xticks(size=15)
```

```
C:\Users\Varun\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

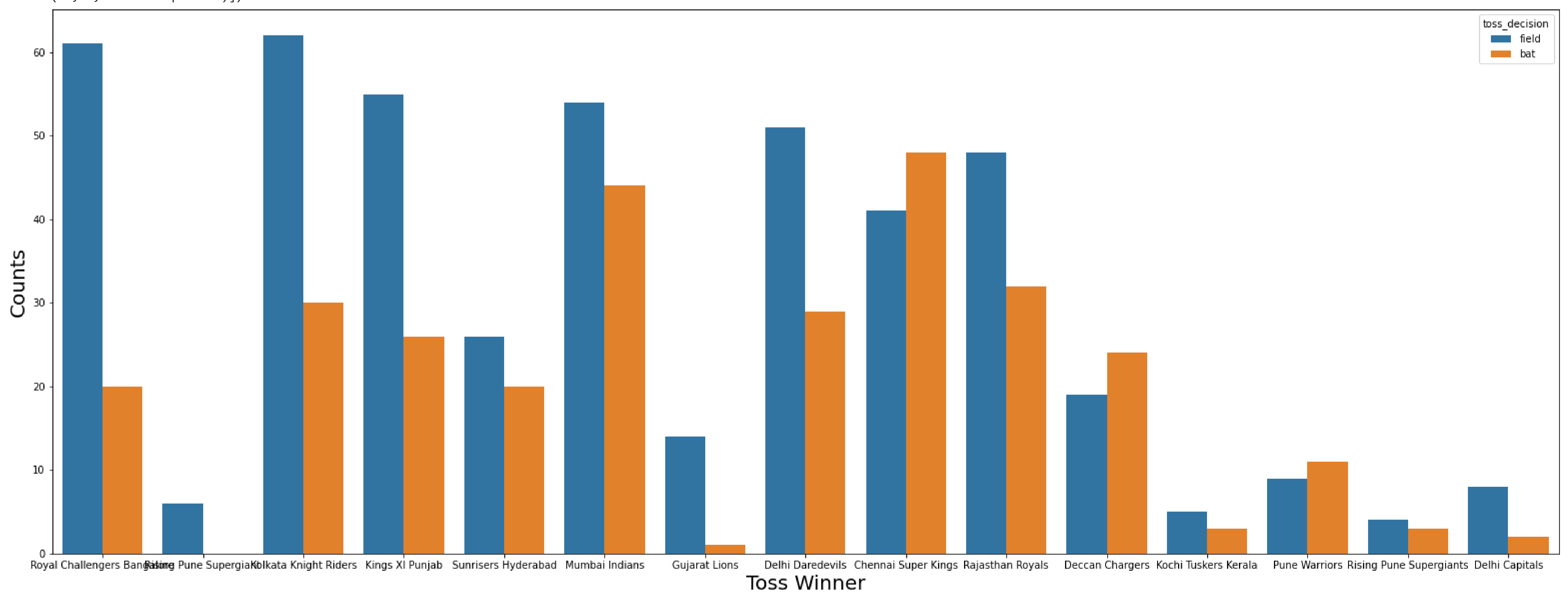
Out[29]:
```
(array([0, 1]), [Text(0, 0, 'No'), Text(1, 0, 'Yes')])
```



In [30]:
```
plt.figure(figsize=(27,10))
sns.countplot('toss_winner', data=matches,hue='toss_decision')
plt.xlabel('Toss Winner',size=20)
plt.ylabel('Counts', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

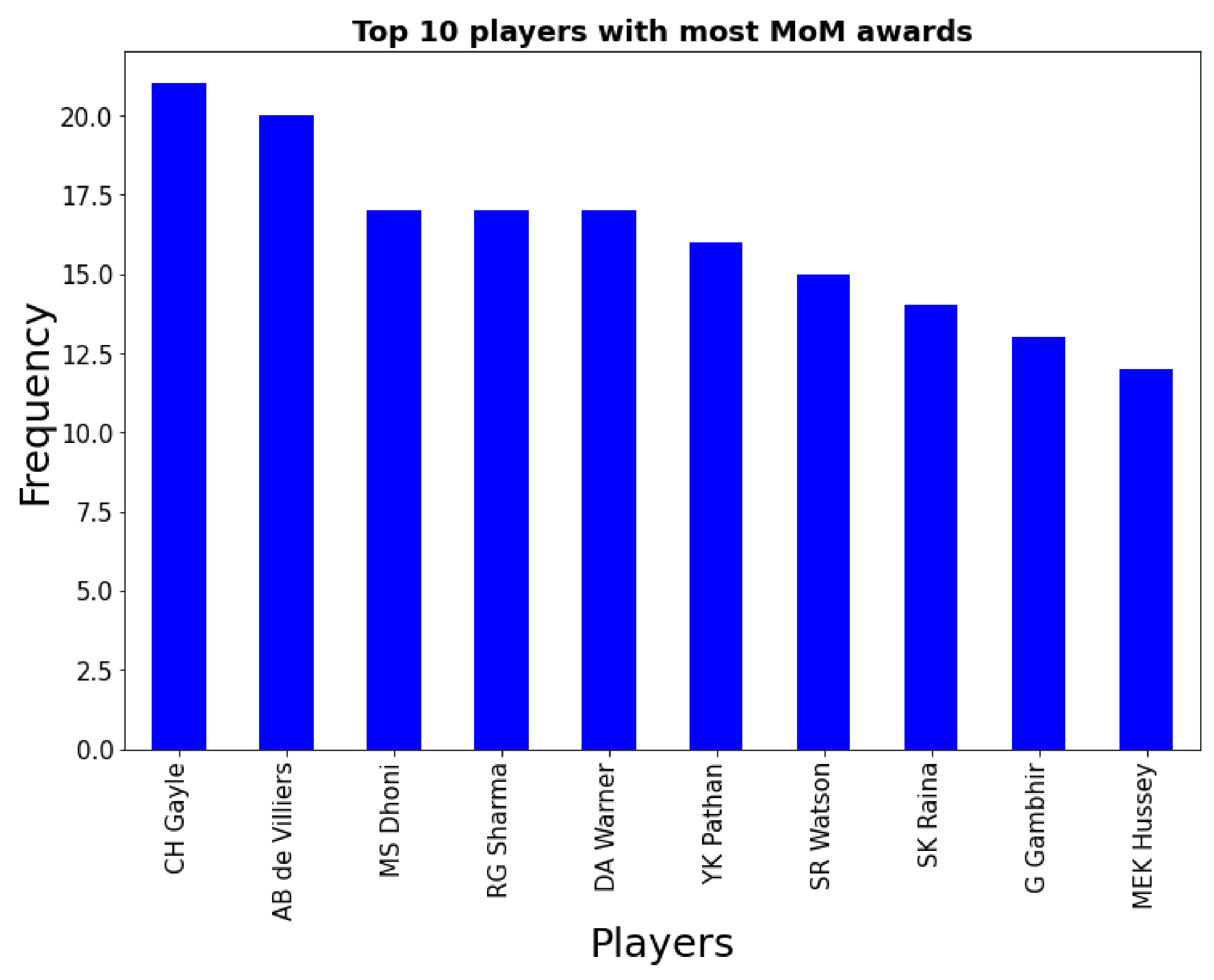
```
C:\Users\Varun\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[30]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(0, 0, 'Royal Challengers Bangalore'),
  Text(1, 0, 'Rising Pune Supergiant'),
  Text(2, 0, 'Kolkata Knight Riders'),
  Text(3, 0, 'Kings XI Punjab'),
  Text(4, 0, 'Sunrisers Hyderabad'),
  Text(5, 0, 'Mumbai Indians'),
  Text(6, 0, 'Gujarat Lions'),
  Text(7, 0, 'Delhi Daredevils'),
  Text(8, 0, 'Chennai Super Kings'),
  Text(9, 0, 'Rajasthan Royals'),
  Text(10, 0, 'Deccan Chargers'),
  Text(11, 0, 'Kochi Tuskers Kerala'),
  Text(12, 0, 'Pune Warriors'),
  Text(13, 0, 'Rising Pune Supergiants'),
  Text(14, 0, 'Delhi Capitals')])
```



In [31]:
```
MoM = matches['player_of_match'].value_counts()
MoM.head(10).plot(kind ='bar', figsize=(12,8), fontsize=15,color='blue')
plt.title('Top 10 players with most MoM awards',fontsize=18, fontweight='bold')
plt.xlabel('Players',size=25)
plt.ylabel('Frequency',size=25)
```

Out[31]:
```
Text(0, 0.5, 'Frequency')
```

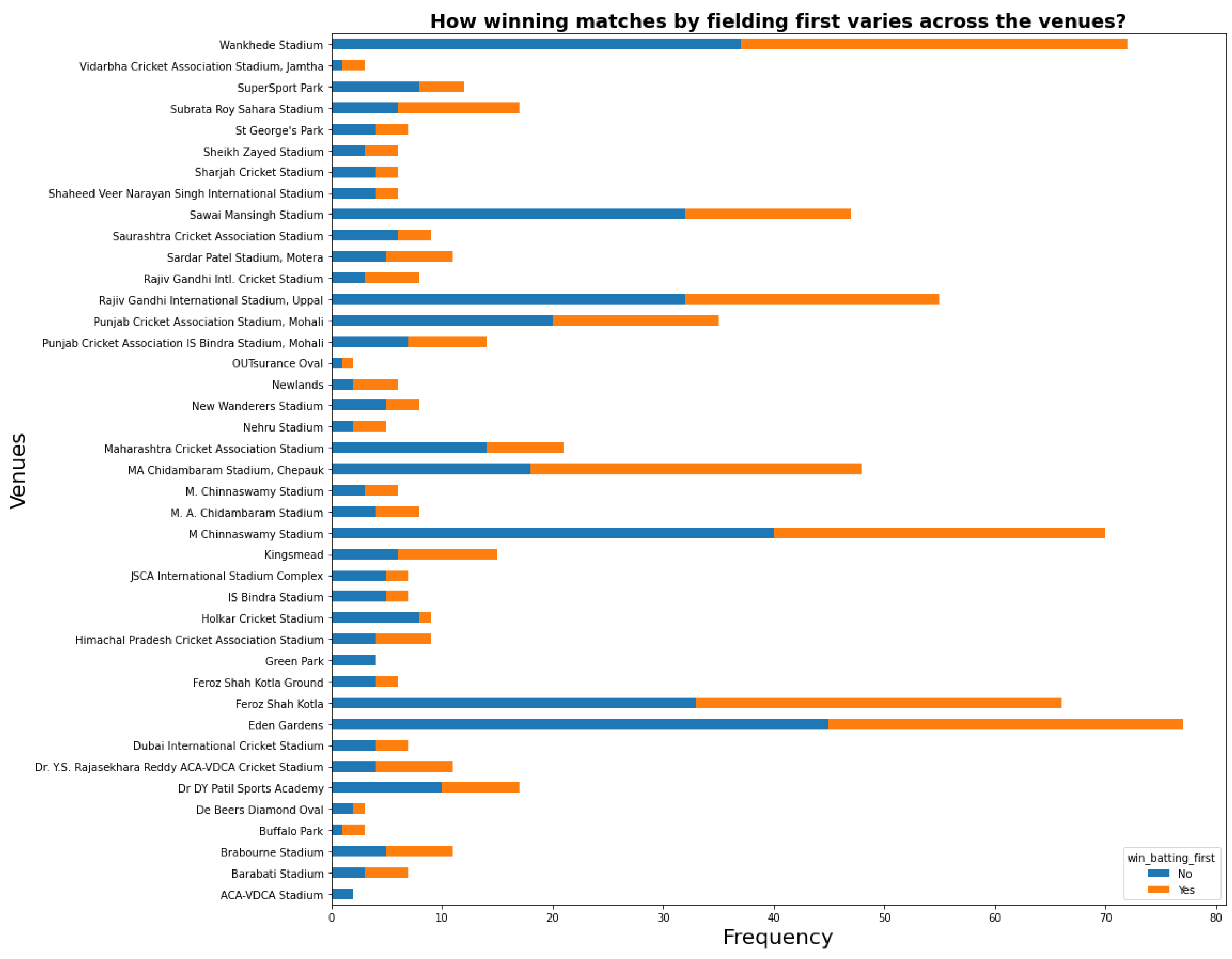


Fielding first varies across venues

In [32]:
```
new_matches= matches[matches['result']=='normal']
new_matches['win_batting_first']=np.where((new_matches.win_by_runs>0), 'Yes','No')
new_matches.groupby('venue')['win_batting_first'].value_counts().unstack().plot(kind='barh', stacked=True,figsize=(15,15))
plt.title('How winning matches by fielding first varies across the venues?', fontsize=18,fontweight='bold')
plt.xlabel('Frequency',size=20)
plt.ylabel('Venues', size=20)
```

```
<ipython-input-32-d1a6a57a527c>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_matches['win_batting_first']=np.where((new_matches.win_by_runs>0), 'Yes','No')
```

Out[32]:
```
Text(0, 0.5, 'Venues')
```

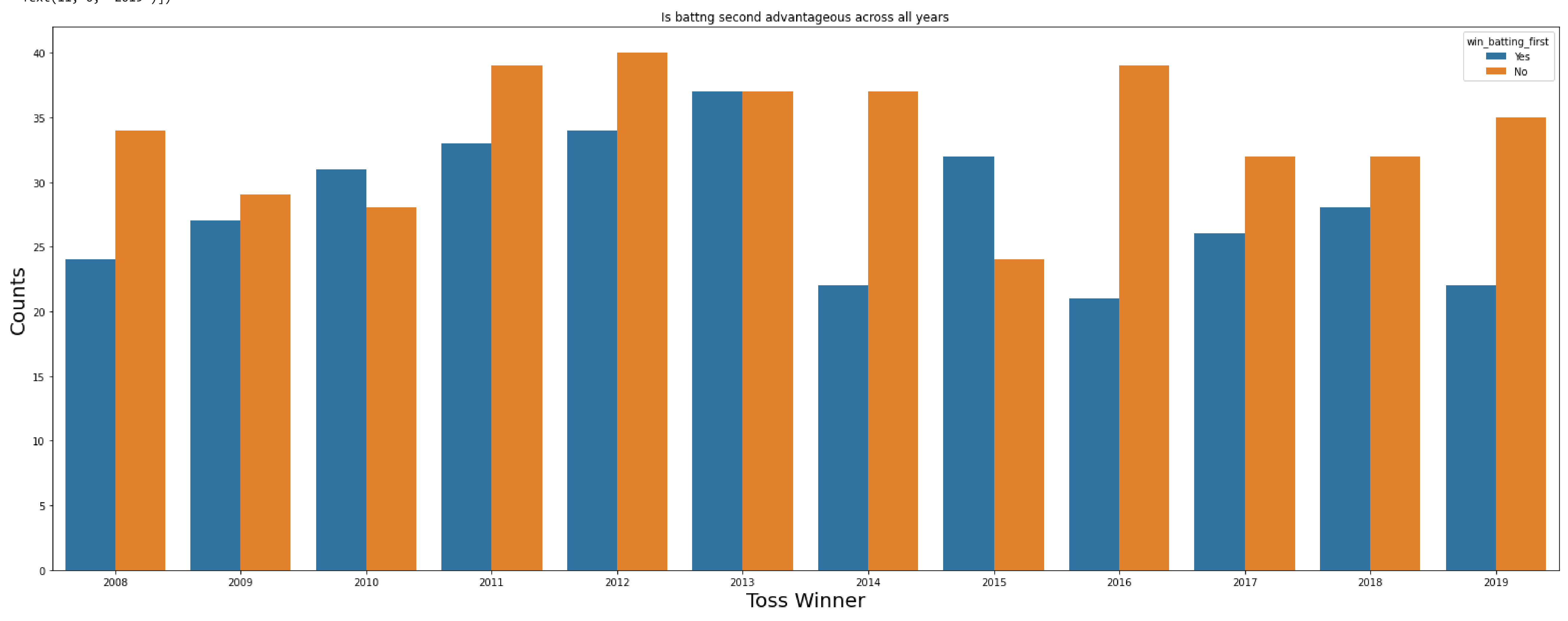


Was batting second advantageous?

In [33]:
```
plt.figure(figsize=(27,10))
sns.countplot('season', data=new_matches,hue='win_batting_first')
plt.title('Is battng second advantageous across all years')
plt.xlabel('Toss Winner',size=20)
plt.ylabel('Counts', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

```
C:\Users\Varun\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

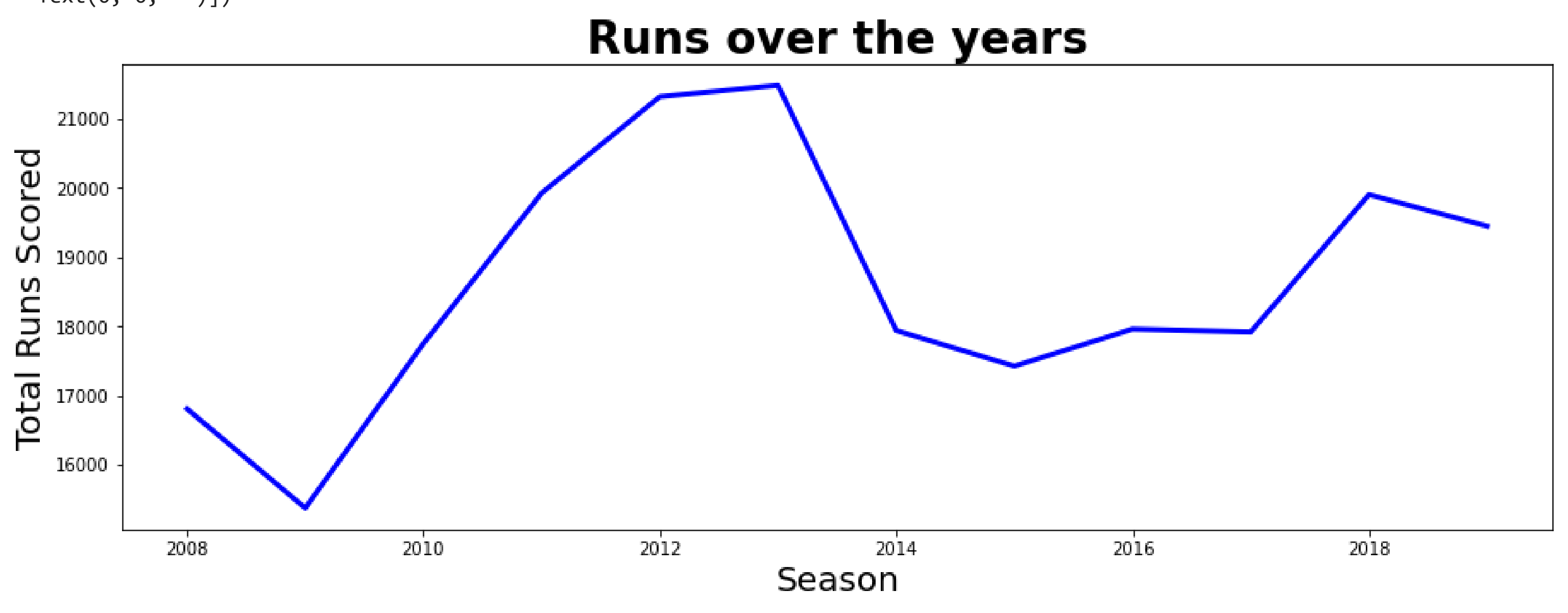
Out[33]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, '2008'),
  Text(1, 0, '2009'),
  Text(2, 0, '2010'),
  Text(3, 0, '2011'),
  Text(4, 0, '2012'),
  Text(5, 0, '2013'),
  Text(6, 0, '2014'),
  Text(7, 0, '2015'),
  Text(8, 0, '2016'),
  Text(9, 0, '2017'),
  Text(10, 0, '2018'),
  Text(11, 0, '2019')])
```



Teams Total Scoring

In [34]:
```
merge.groupby('season')['batsman_runs'].sum().plot(kind='line',linewidth=3,figsize=(15,5),color='blue')
plt.title('Runs over the years', fontsize=26,fontweight='bold')
plt.xlabel('Season',size=20)
plt.ylabel('Total Runs Scored', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

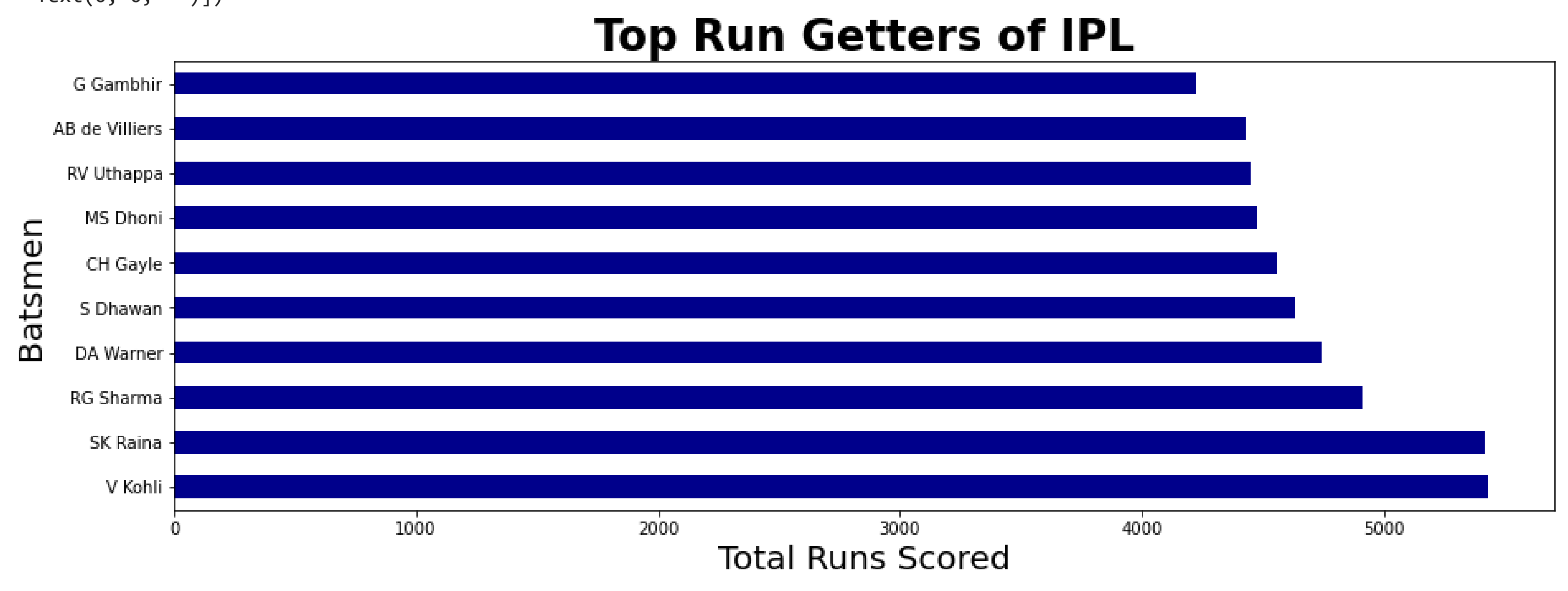
Out[34]:
```
(array([2006., 2008., 2010., 2012., 2014., 2016., 2018., 2020.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



Top Run scores of IPL

In [35]:
```
merge.groupby('batsman')['batsman_runs'].sum().sort_values(ascending=False).head(10).plot(kind='barh', color='darkblue',figsize=(15,5))
plt.title('Top Run Getters of IPL', fontsize=26,fontweight='bold')
plt.xlabel('Total Runs Scored',size=20)
plt.ylabel('Batsmen', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

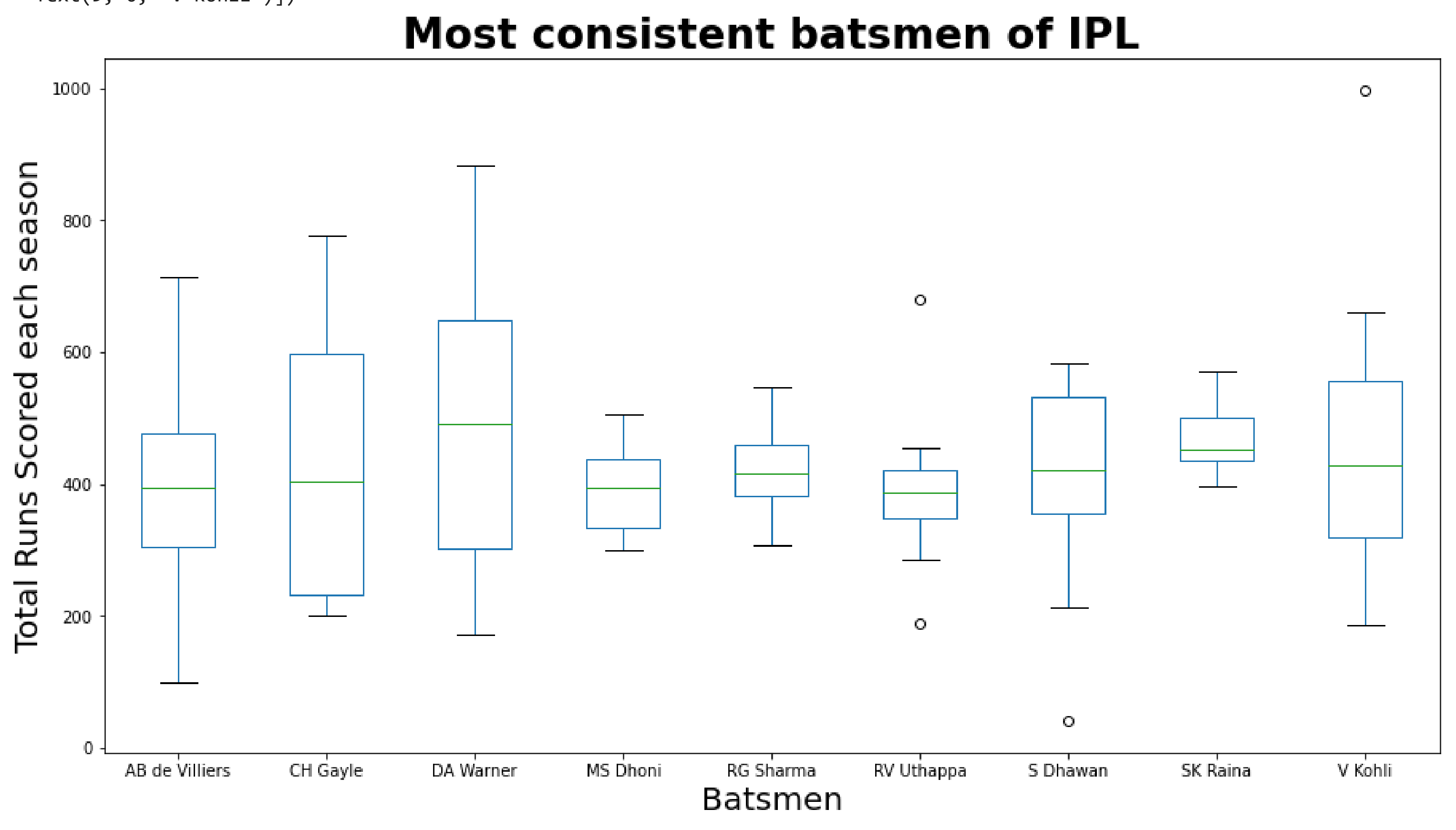
Out[35]:
```
(array([   0., 1000., 2000., 3000., 4000., 5000., 6000.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



Consistent Player among the top 10 run scorers

In [36]:
```
consistent_batsman= merge[merge.batsman.isin(['SK Raina', 'V Kohli','RG Sharma','G Gambir','RV Uthappa','S Dhawan','CH Gayle','MS Dhoni','DA Warner','AB de Villiers'])][['batsman','season','total_runs']]
consistent_batsman.groupby(['season','batsman'])['total_runs'].sum().unstack().plot(kind='box',figsize=(15,8))
plt.title('Most consistent batsmen of IPL', fontsize=26,fontweight='bold')
plt.xlabel('Batsmen',size=20)
plt.ylabel('Total Runs Scored each season', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

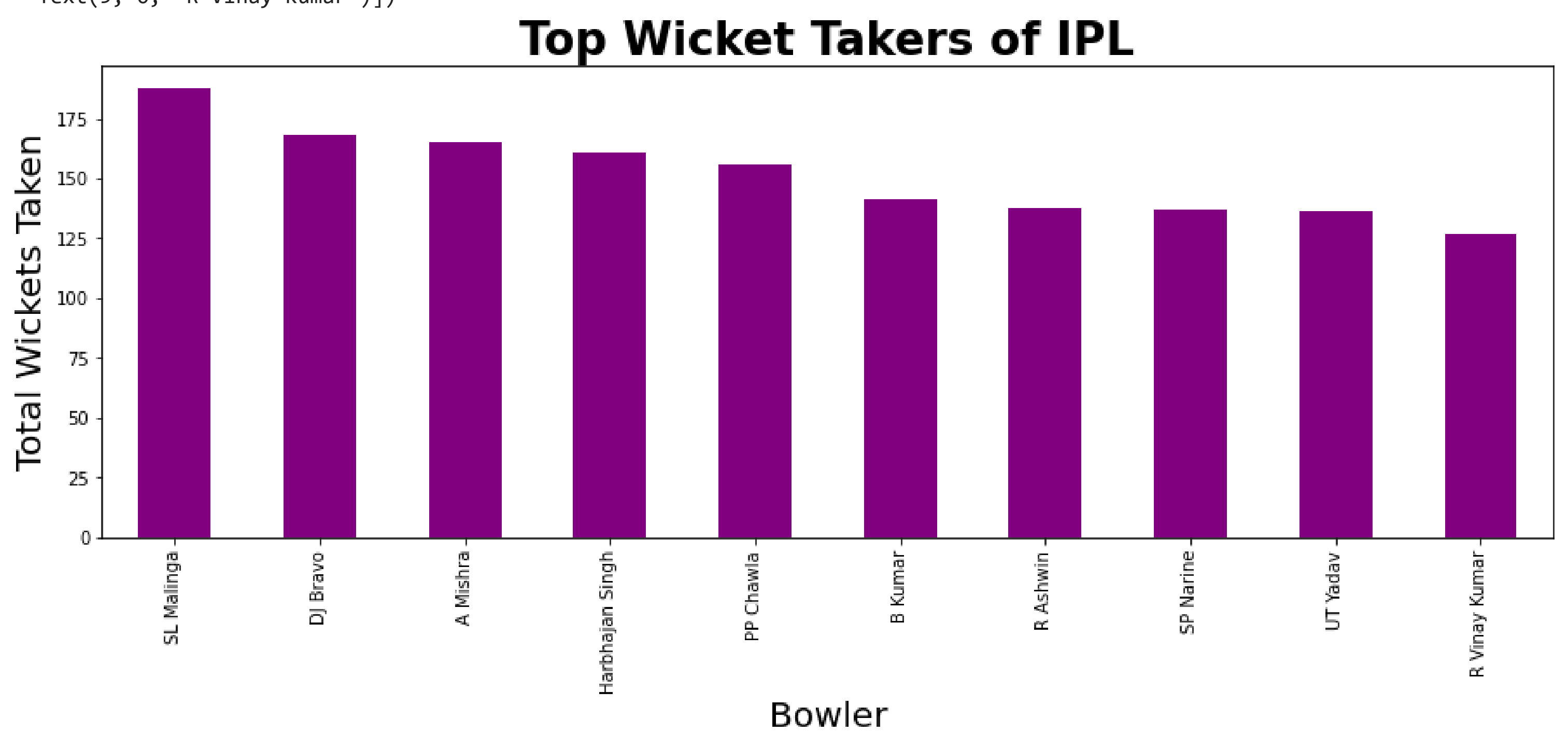
Out[36]:
```
(array([1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(1, 0, 'AB de Villiers'),
  Text(2, 0, 'CH Gayle'),
  Text(3, 0, 'DA Warner'),
  Text(4, 0, 'MS Dhoni'),
  Text(5, 0, 'RG Sharma'),
  Text(6, 0, 'RV Uthappa'),
  Text(7, 0, 'S Dhawan'),
  Text(8, 0, 'SK Raina'),
  Text(9, 0, 'V Kohli')])
```



Best Bowlers

In [37]:
```
merge.groupby('bowler')['player_dismissed'].count().sort_values(ascending=False).head(10).plot(kind='bar', color='purple',figsize=(15,5))
plt.title('Top Wicket Takers of IPL', fontsize=26,fontweight='bold')
plt.xlabel('Bowler',size=20)
plt.ylabel('Total Wickets Taken', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

Out[37]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'SL Malinga'),
  Text(1, 0, 'DJ Bravo'),
  Text(2, 0, 'A Mishra'),
  Text(3, 0, 'Harbhajan Singh'),
  Text(4, 0, 'PP Chawla'),
  Text(5, 0, 'B Kumar'),
  Text(6, 0, 'R Ashwin'),
  Text(7, 0, 'SP Narine'),
  Text(8, 0, 'UT Yadav'),
  Text(9, 0, 'R Vinay Kumar')])
```



Batsmen with a best strike rate

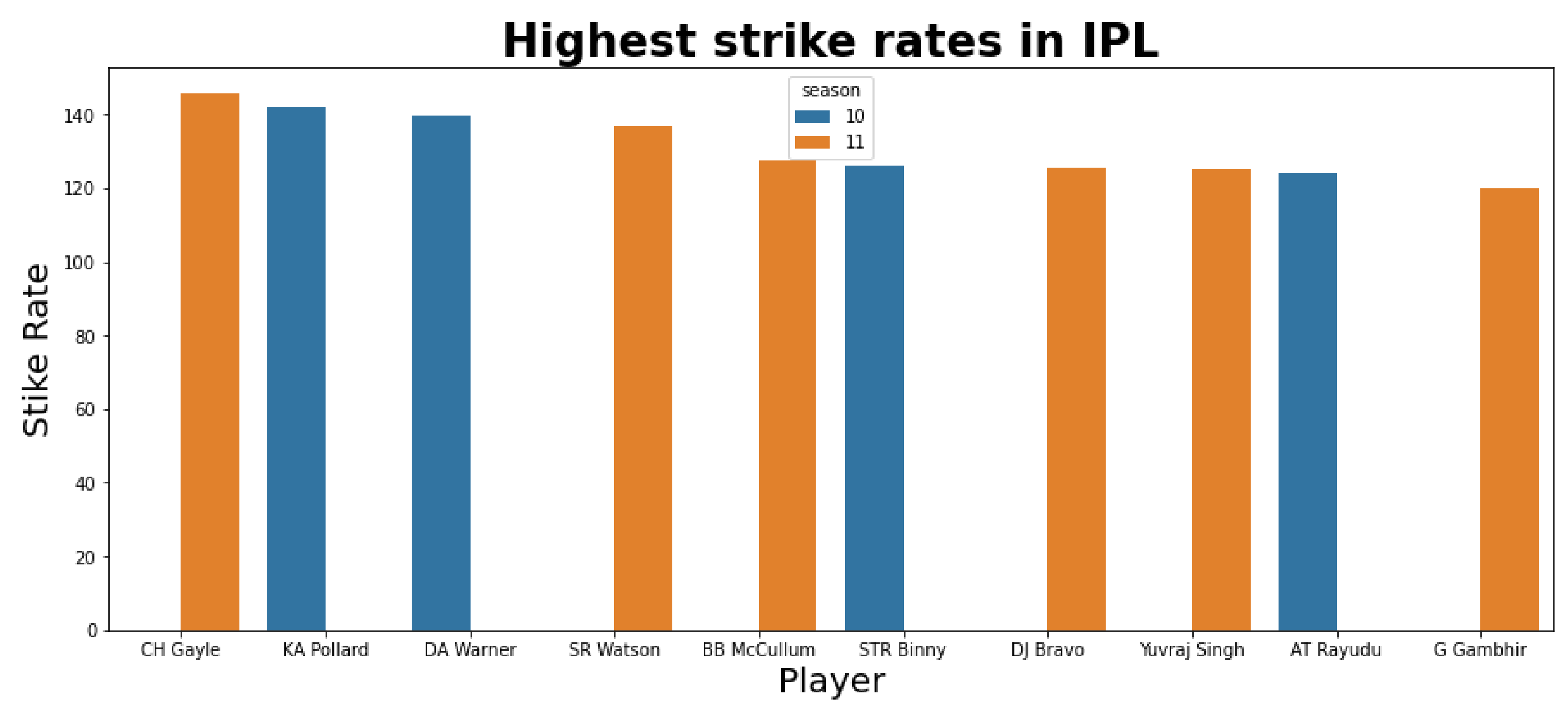
In [38]:
```
no_of_balls=pd.DataFrame(merge.groupby('batsman')['ball'].count())
runs=pd.DataFrame(merge.groupby('batsman')['batsman_runs'].sum())
seasons=pd.DataFrame(merge.groupby('batsman')['season'].nunique())

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

batsman_strike_rate['strike_rate']=batsman_strike_rate['run']/batsman_strike_rate['balls']*100
highest_strike_rate= batsman_strike_rate[batsman_strike_rate.season.isin([10,11])][['season','batsman','strike_rate']].sort_values(by='strike_rate', ascending= False)

highest_strike_rate.head(10)
```

Out[38]:

	season	batsman	strike_rate
92	11	CH Gayle	145.640370
213	10	KA Pollard	141.751527
112	10	DA Warner	139.523249
444	11	SR Watson	136.945813
72	11	BB McCullum	127.332746
449	10	STR Binny	126.000000
118	11	DJ Bravo	125.565801
514	11	Yuvraj Singh	125.283190
53	10	AT Rayudu	124.058187
147	11	G Gambhir	119.835414

In [39]:
```
plt.figure(figsize=(15,6))
sns.barplot(x='batsman', y='strike_rate', data= highest_strike_rate.head(10), hue='season')
plt.title('Highest strike rates in IPL', fontsize=26,fontweight='bold')
plt.xlabel('Player',size=20)
plt.ylabel('Stike Rate', size=20)
plt.yticks(size=10)
plt.xticks(size=10)
```

Out[39]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'CH Gayle'),
  Text(1, 0, 'KA Pollard'),
  Text(2, 0, 'DA Warner'),
  Text(3, 0, 'SR Watson'),
  Text(4, 0, 'BB McCullum'),
  Text(5, 0, 'STR Binny'),
  Text(6, 0, 'DJ Bravo'),
  Text(7, 0, 'Yuvraj Singh'),
  Text(8, 0, 'AT Rayudu'),
  Text(9, 0, 'G Gambhir')])
```

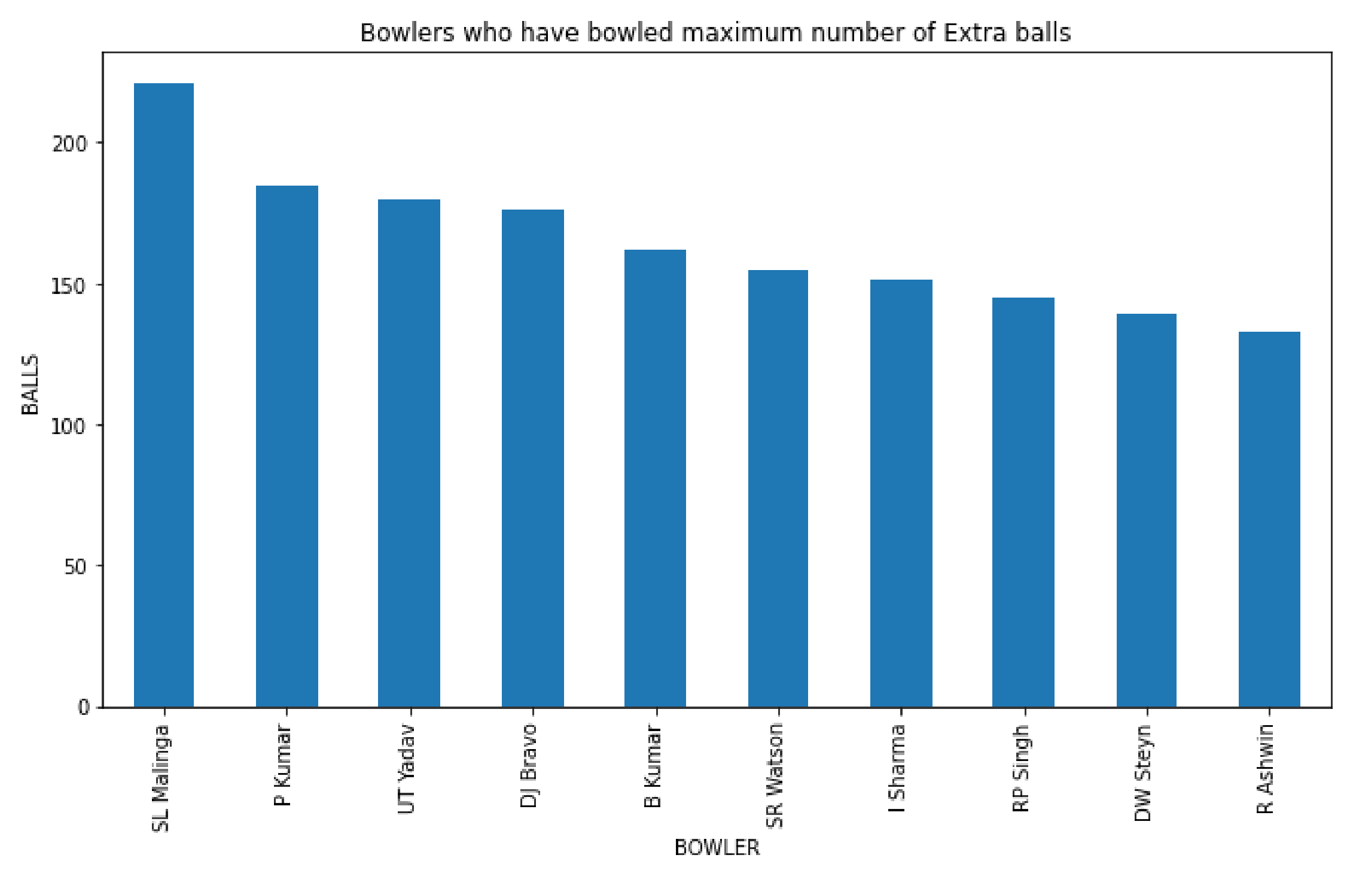


Bowlers with maximum number of extras

In [40]:
```
extra = deliveries[deliveries['extra_runs']!=0]['bowler'].value_counts()[:10]
extra.plot(kind='bar', figsize=(11,6),title='Bowlers who have bowled maximum number of Extra balls')

plt.xlabel('BOWLER')
plt.ylabel('BALLS')

extra = pd.DataFrame(extra)
extra.T
```

Out[40]:

	SL Malinga	P Kumar	UT Yadav	DJ Bravo	B Kumar	SR Watson	I Sharma	RP Singh	DW Steyn	R Ashwin
bowler	221	185	180	176	162	155	151	145	139	133



In [41]:
```
balls_bowled= pd.DataFrame(merge.groupby('bowler')['ball'].count())
wickets_taken= pd.DataFrame(merge[merge['dismissal_kind']!='no dismissal'].groupby('bowler')['dismissal_kind'].count())
seasons_played= pd.DataFrame(merge.groupby('bowler')['season'].nunique())
bowler_strike_rate= pd.DataFrame({'balls':balls_bowled['ball'],'wickets':wickets_taken['dismissal_kind'], 'season':seasons_played['season']})
bowler_strike_rate.reset_index(inplace=True)
```

In [42]:
```
bowler_strike_rate['strike_rate']=bowler_strike_rate['balls']/bowler_strike_rate['wickets']
def highlight_cols(s):
    color='skyblue'
    return 'background-color: %s' % color
best_bowling_strike_rate=bowler_strike_rate[bowler_strike_rate['wickets']>50].sort_values(by='strike_rate',ascending=True)
best_bowling_strike_rate.head().style.applymap(highlight_cols, subset=pd.IndexSlice[:,['bowler','wickets','strike_rate']])
```

Out[42]:

	bowler	balls	wickets	season	strike_rate
134	Imran Tahir	1249	82	6	15.231707
340	SL Malinga	2974	188	9	15.819149
93	DJ Bravo	2711	168	10	16.136905
9	A Nehra	1974	121	9	16.314050
225	MM Patel	1382	82	7	16.853659

Conclusion

- The exploratory data analysis task was carried out successfully to analyze and visualize the best team, player, match, etc. from the given IPL dataset
- The outcomes are listed below

Part-A[Outcomes from the Data]

- Mumbai Indians were the best team with highest number of wins.
- When chasing a target, the biggest victory was by 10 wickets and there were 11 such instances.
- Most of IPL matches were held at EDEN GARDENS,KOLKATA.
- Teams choosing the first field option had the highest winning probability.
- Surender Ravi has officiated the most number of IPL matches on field.
- Chris Gyle has the maximum number of match titles.
- Biggest victory was by 146 runs (defending)

part -B[Suggestions to the company for hiring the best player]

- Consistent Batsman: Virat kohli, Suresh Raina, Rohit Sharma, David Warner.
- Game Changing Batsman: Chris Gayle, AB de Villiers, Rohit Sharma, David Warner.
- Batsman scoring good runs(Every match): DA Warner, Chris Gayle, Virat Kohli, AB de Villiers, Shiker Dhawan.
- Best Finishers(with good strike rate): Chris Gayle, KA Pollard, DA Warner, SR Warson, BB McCulum
- Experienced Bowler: Harbhajan Singh, A Mishra, PP Chawla, R Ashwin, SL Malinga, DJ Bravo
- Best Wicket taking Bowlers: SL Malinga, DJ Bravo, A Mishra, PP Chawla
- Bowlers with Highest dot balls: Harbhajan Singh, SL Malinga, B Kumar, A Mishra, PP chawla
- Bowlers with good economy: DW Steyn, M Muralitharan, R Ashwin, SP Narine, Harbhajan Singh