Author - AYUSH CHHOKER ¶

DATA SCIENCE AND BUSINESS ANALYTICS INTERN

Task 5 : Exploratory Data Analysis - Sports

1. Perform ‘Exploratory Data Analysis’ on dataset ‘Indian Premier League’
2. As a sports analysts, find out the most successful teams, players and factors contributing win or loss of a team.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Dataset 1

In [2]:

```
matches = pd.read_csv("D:\matches.csv")
```

In [5]:

```
matches.head()
```

Out[5]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal

In [6]:

```
matches.tail()
```

Out[6]:

	id	season	city	date	team1	team2	toss_winner	toss_decisior
751	11347	2019	Mumbai	05/05/19	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	fielc
752	11412	2019	Chennai	07/05/19	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	ba
753	11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	fielc
754	11414	2019	Visakhapatnam	10/05/19	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	fielc
755	11415	2019	Hyderabad	12/05/19	Mumbai Indians	Chennai Super Kings	Mumbai Indians	ba

In [7]:

```
matches.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               756 non-null    int64 
 1   season           756 non-null    int64 
 2   city             749 non-null    object
 3   date             756 non-null    object
 4   team1            756 non-null    object
 5   team2            756 non-null    object
 6   toss_winner      756 non-null    object
 7   toss_decision    756 non-null    object
 8   result           756 non-null    object
 9   dl_applied       756 non-null    int64 
 10  winner           752 non-null    object
 11  win_by_runs      756 non-null    int64 
 12  win_by_wickets   756 non-null    int64 
 13  player_of_match  752 non-null    object
 14  venue            756 non-null    object
 15  umpire1          754 non-null    object
 16  umpire2          754 non-null    object
 17  umpire3          119 non-null    object
dtypes: int64(5), object(13)
memory usage: 106.4+ KB
```

In [8]:

```
matches.shape
```

Out[8]:

```
(756, 18)
```

In [9]:

```
matches.describe()
```

Out[9]:

	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

Dataset 2

In [11]:

```
deliveries = pd.read_csv("D:\deliveries.csv")
deliveries.head()
```

Out[11]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills

5 rows × 21 columns

In [14]:

```
deliveries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179078 entries, 0 to 179077
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   match_id          179078 non-null  int64 
 1   inning            179078 non-null  int64 
 2   batting_team      179078 non-null  object
 3   bowling_team      179078 non-null  object
 4   over              179078 non-null  int64 
 5   ball              179078 non-null  int64 
 6   batsman           179078 non-null  object
 7   non_striker       179078 non-null  object
 8   bowler            179078 non-null  object
 9   is_super_over     179078 non-null  int64 
 10  wide_runs         179078 non-null  int64 
 11  bye_runs          179078 non-null  int64 
 12  legbye_runs       179078 non-null  int64 
 13  noball_runs       179078 non-null  int64 
 14  penalty_runs      179078 non-null  int64 
 15  batsman_runs      179078 non-null  int64 
 16  extra_runs        179078 non-null  int64 
 17  total_runs        179078 non-null  int64 
 18  player_dismissed  8834 non-null    object
 19  dismissal_kind    8834 non-null    object
 20  fielder           6448 non-null    object
dtypes: int64(13), object(8)
memory usage: 28.7+ MB
```

In [13]:

```
deliveries.shape
```

Out[13]:

```
(179078, 21)
```

In [12]:

```
deliveries.describe()
```

Out[12]:

	match_id	inning	over	ball	is_super_over	wide
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.0
mean	1802.252957	1.482952	10.162488	3.615587	0.000452	0.0
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.2
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.0
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.0
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.0
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.0
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.0

We will merge the 2 datasets for better insights from the data

In [16]:

```
merge = pd.merge(deliveries,matches, left_on='match_id', right_on ='id')
merge.head(5)
```

Out[16]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills

5 rows × 39 columns

In [17]:

```
merge.info()
```

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

In [18]:

```
merge.describe()
```

Out[18]:

	match_id	inning	over	ball	is_super_over	wide
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.0
mean	1802.252957	1.482952	10.162488	3.615587	0.000452	0.0
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.2
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.0
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.0
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.0
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.0
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.0

In [19]:

```
matches.id.is_unique
```

Out[19]:

```
True
```

id is unique we can set this as our index

In [20]:

```
matches.set_index('id', inplace=True)
```

In [21]:

```
#Summary statistics of matches data
matches.describe(include = 'all')
```

Out[21]:

	season	city	date	team1	team2	toss_winner	toss_decision	resul
count	756.000000	749	756	756	756	756	756	75
unique	NaN	32	546	15	15	15	2	
top	NaN	Mumbai	2010-04-07	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	field	norma
freq	NaN	101	2	101	95	98	463	74
mean	2013.444444	NaN	NaN	NaN	NaN	NaN	NaN	Na
std	3.366895	NaN	NaN	NaN	NaN	NaN	NaN	Na
min	2008.000000	NaN	NaN	NaN	NaN	NaN	NaN	Na
25%	2011.000000	NaN	NaN	NaN	NaN	NaN	NaN	Na
50%	2013.000000	NaN	NaN	NaN	NaN	NaN	NaN	Na
75%	2016.000000	NaN	NaN	NaN	NaN	NaN	NaN	Na
max	2019.000000	NaN	NaN	NaN	NaN	NaN	NaN	Na

Data Preprocessing

Here we will perform Data Preprocessing on our matches dataset first, to make the data usable for EDA.

In [22]:

```
matches.head()
```

Out[22]:

	season	city	date	team1	team2	toss_winner	toss_decision	result	c
id									
1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	

From Pre profiling, we found that:

city has missing values

team1 and team2 columns have 14 distinct values but winner has 15 distinct values

umpire1 and umpire2 have 1 missing value each

umpire3 has 94% missing values

city has 33 distinct values while venue has 35 distinct values

So, missing values can be filled with Dubai

In [23]:

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

umpire1 and umpire2 columns have one missing value each.

In [24]:

```
matches[(matches.umpire1.isnull()) | (matches.umpire2.isnull())]
```

Out[24]:

	season	city	date	team1	team2	toss_winner	toss_decision
id							
5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat
11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field

Umpire3 column has close to 92% missing values. hence dropping that column

In [25]:

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

In [26]:

```
#city has 33 distinct values while we have 35 venues.
#Let's find out venues grouped by cities to see which cities have multiple venues

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

Out[26]:

		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

city	venue	season
	Subrata Roy Sahara Stadium	17
Raipur	Shaheed Veer Narayan Singh International Stadium	6
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

Observations

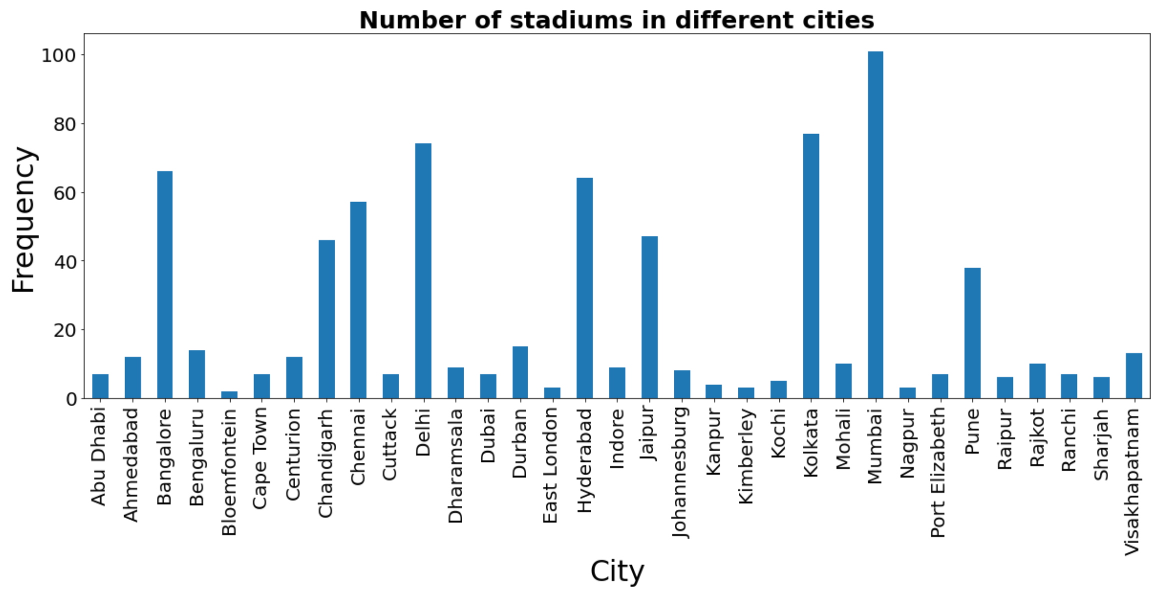
- Bengaluru and Bangalore both are in the data when they are same. So we need to keep one of them
- Chandigarh and Mohali are same and there is just one stadium Punjab Cricket Association IS Bindra Stadium, Mohali whose value has not been entered correctly. We need to have either Chandigarh or Mohali as well as correct name of the stadium there
- Mumbai has 3 stadiums/venues used for IPL
- Pune has 2 venues for IPL
  ### Visual representation of number of venues in each city .

In [28]:

```
#Plotting venues along with cities
v = pd.crosstab(matches['city'],matches['venue'])
v.replace(v[v!=0],1, inplace = True)

#Adding a column by summing each columns
v['count'] = v.sum(axis = 'columns')
#We will just keep last column = 'count'
b = v['count']

#Plotting
plt.figure(figsize = (20,7))
b.plot(kind = 'bar')
plt.title("Number of stadiums in different cities", fontsize = 25, fontweight = 'bold')
plt.xlabel("City", size = 30)
plt.ylabel("Frequency", size = 30)
plt.xticks(size = 20)
plt.yticks(size = 20)
```

Out[28]:

```
(array([  0.,  20.,  40.,  60.,  80., 100., 120.]),
 <a list of 7 Text major ticklabel objects>)
```



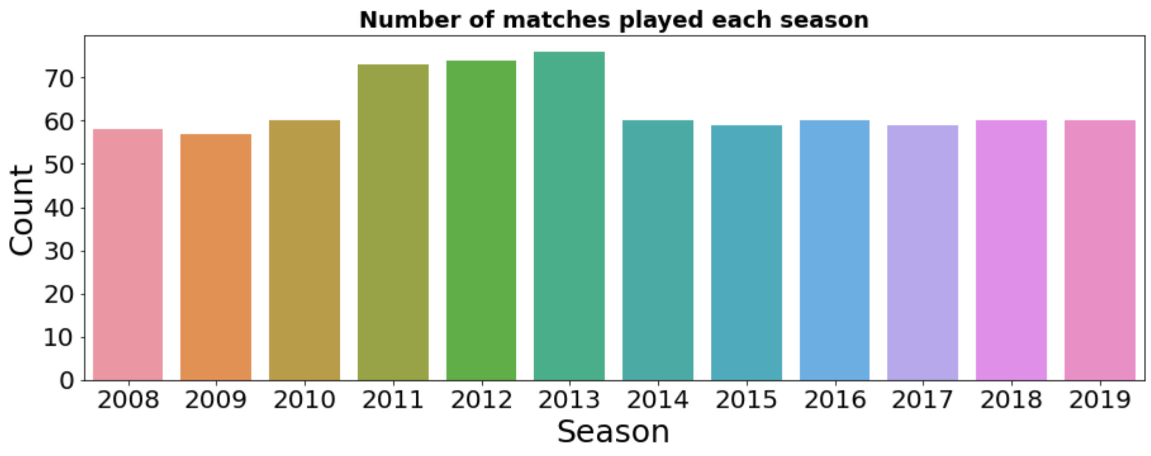
EDA

In [29]:

```
plt.figure(figsize=(15,5))
sns.countplot('season', data = matches)
plt.title("Number of matches played each season",fontsize=18,fontweight="bold")
plt.ylabel("Count", size = 25)
plt.xlabel("Season", size = 25)
plt.xticks(size = 20)
plt.yticks(size = 20)
```

Out[29]:

```
(array([ 0., 10., 20., 30., 40., 50., 60., 70., 80.]),
 <a list of 9 Text major ticklabel objects>)
```



2011-2013 have more matches being played than other seasons

All other seasons have approximately 58-60 matches while 2011-2013 have more than 70 matches.

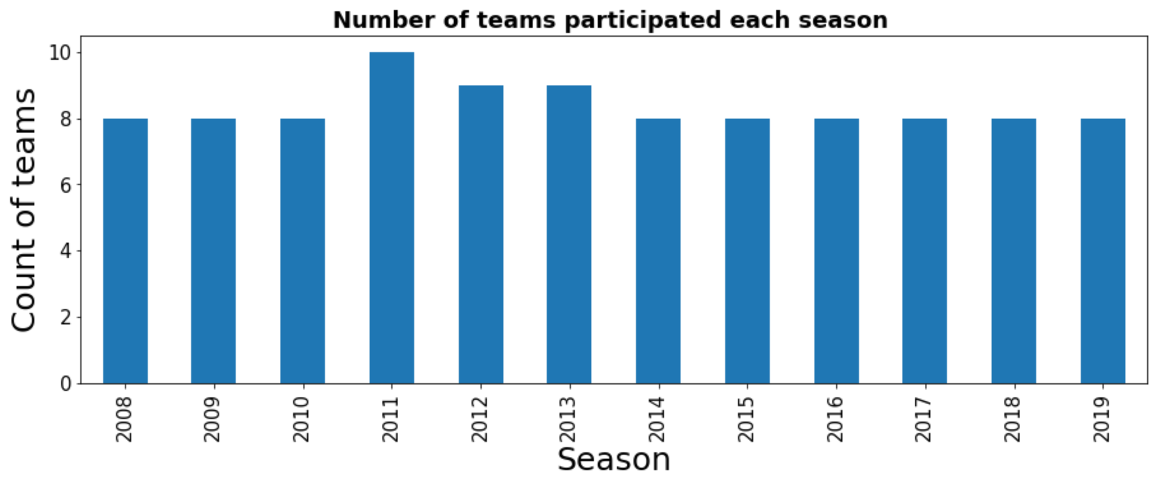
How many teams played in each season?

In [30]:

```
matches.groupby('season')['team1'].nunique().plot(kind = 'bar', figsize=(15,5))
plt.title("Number of teams participated each season ",fontsize=18,fontweight="bold")
plt.ylabel("Count of teams", size = 25)
plt.xlabel("Season", size = 25)
plt.xticks(size = 15)
plt.yticks(size = 15)
```

Out[30]:

```
(array([ 0.,  2.,  4.,  6.,  8., 10., 12.]),
 <a list of 7 Text major ticklabel objects>)
```



10 teams played in 2011 and 9 teams each in 2012 and 2013

This explains why 2011-2013 have seen more matches being played than other seasons

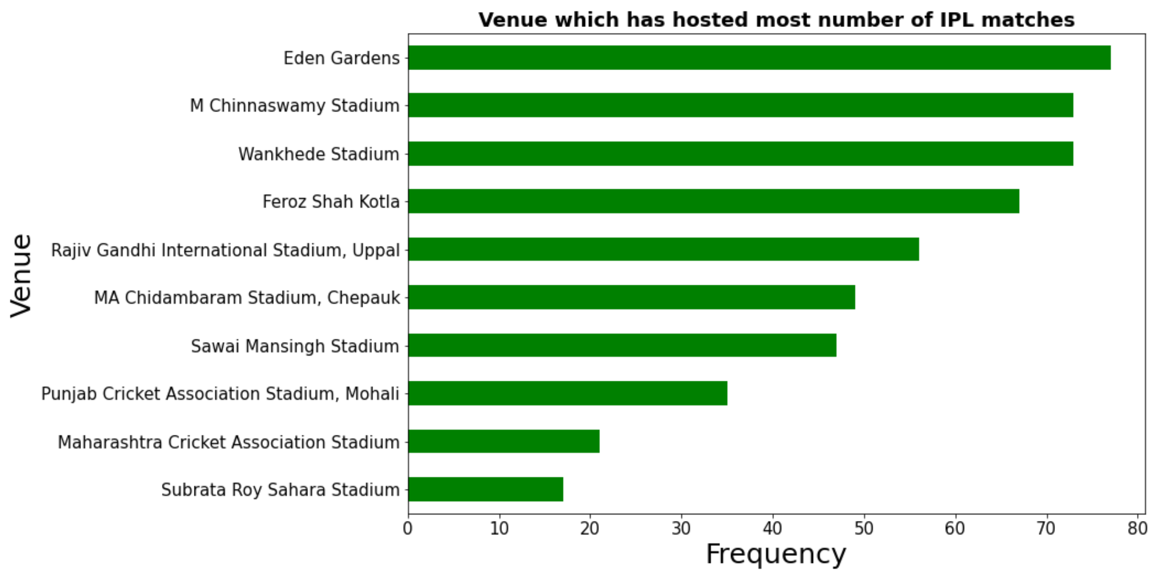
Venue which has hosted most number of IPL matches

In [31]:

```
matches.venue.value_counts().sort_values(ascending = True).tail(10).plot(kind = 'barh',
figsize=(12,8), fontsize=15, color='green')
plt.title("Venue which has hosted most number of IPL matches",fontsize=18,fontweight="b
old")
plt.ylabel("Venue", size = 25)
plt.xlabel("Frequency", size = 25)
```

Out[31]:

```
Text(0.5, 0, 'Frequency')
```



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

Which team has maximum wins in IPL so far?

In [32]:

```
#creating a dataframe with season and winner columns
winning_teams = matches[['season','winner']]
```

In [33]:

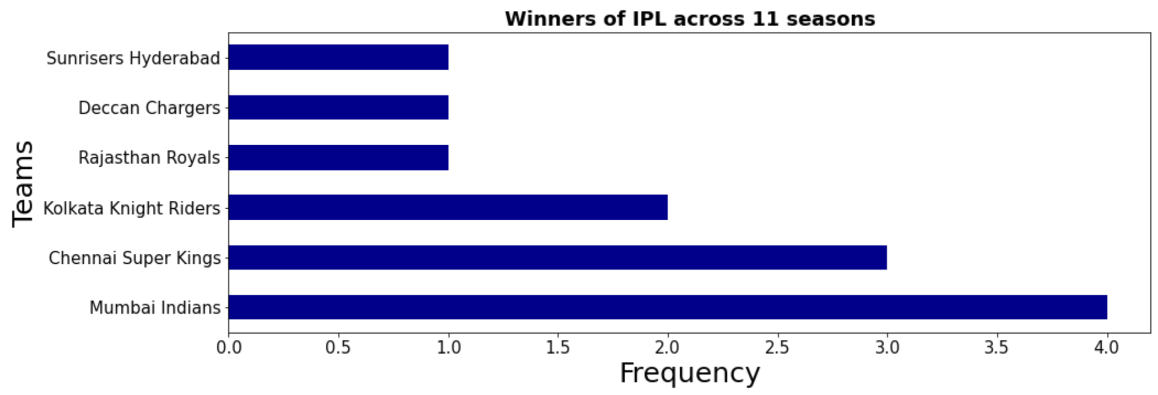
```
#dictionaries to get winners to each season
winners_team = {}
for i in sorted(winning_teams.season.unique()):
    winners_team[i] = winning_teams[winning_teams.season == i]['winner'].tail(1).values
[0]

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

In [34]:

```
winners_of_IPL['team'].value_counts().plot(kind = 'barh', figsize = (15,5), color = 'da
rkblue')
plt.title("Winners of IPL across 11 seasons",fontsize=18,fontweight="bold")
plt.ylabel("Teams", size = 25)
plt.xlabel("Frequency", size = 25)
plt.xticks(size = 15)
plt.yticks(size = 15)
```

Out[34]:

```
(array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)
```



MI and CSK have both won 3 times each followed by KKR who has won 2 times.

Hyderabad team has also won 2 matches under 2 franchise name - Deccan Chargers and Sunrisers Hyderabad

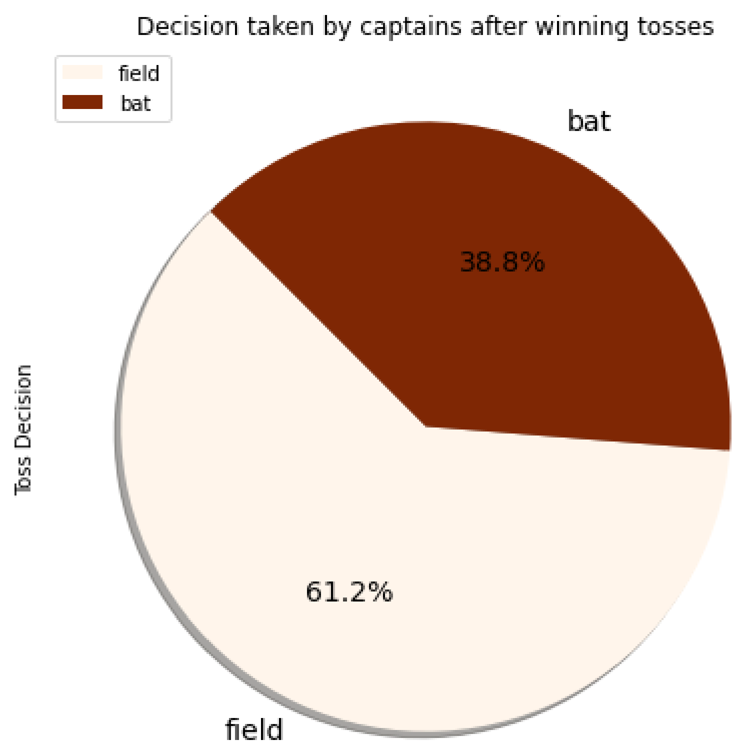
Does teams choosed to bat or field first, after winning toss?

In [35]:

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

plt.ylabel('Toss Decision')
plt.title('Decision taken by captains after winning tosses')
```

Out[35]:

```
Text(0.5, 1.0, 'Decision taken by captains after winning tosses')
```



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

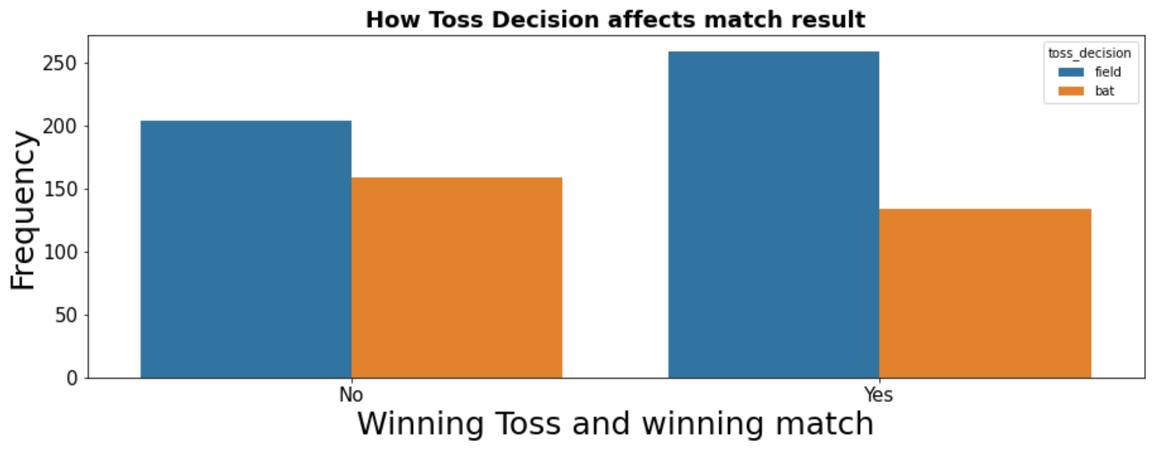
How toss decision affects match results?

In [37]:

```
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 result", fontsize=18,fontweight="bold")
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel("Winning Toss and winning match", fontsize = 25)
plt.ylabel("Frequency", fontsize = 25)
```

Out[37]:

```
Text(0, 0.5, 'Frequency')
```



Teams winning tosses and electng to field first have won most number of time

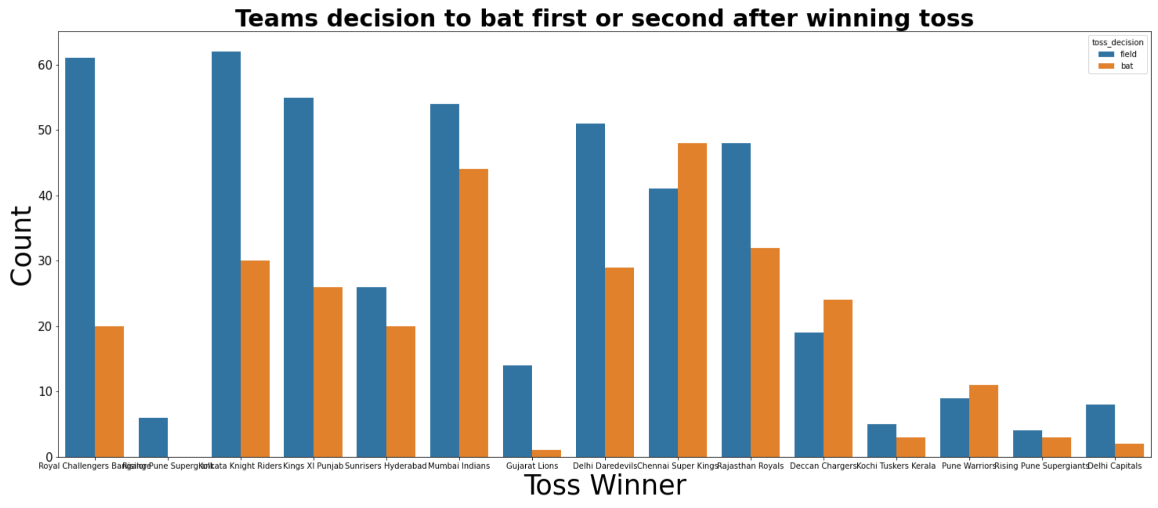
Individual teams decision to choose bat or field after winning toss.

In [36]:

```
plt.figure(figsize = (25,10))
sns.countplot('toss_winner', data = matches, hue = 'toss_decision')
plt.title("Teams decision to bat first or second after winning toss", size = 30, fontwe
ight = 'bold')
plt.xticks(size = 10)
plt.yticks(size = 15)
plt.xlabel("Toss Winner", size = 35)
plt.ylabel("Count", size = 35)
```

Out[36]:

```
Text(0, 0.5, 'Count')
```



Most teams field first after winning toss except for Chennai Super Kings who has mostly opted to bat first. Deccan Chargers and Pune Warriors also show the same trend.

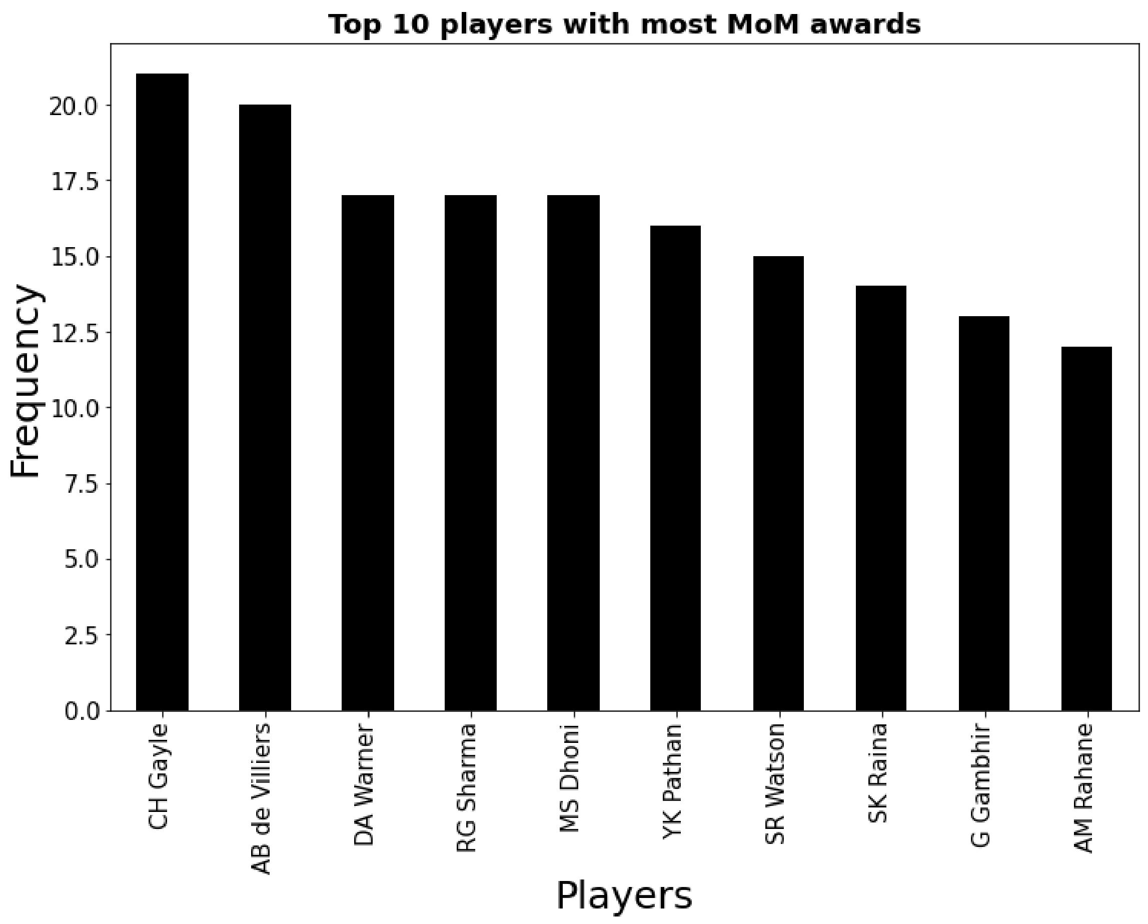
Which player's performance has mostly led team's win?

In [38]:

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

Out[38]:

```
Text(0.5, 0, 'Players')
```



Chris Gayle has so far won the most number of MoM awards followed by AB de Villiers.

Also, all top 10 are batsmen which kind of hints that in IPL batsmen have mostly dictated the matches

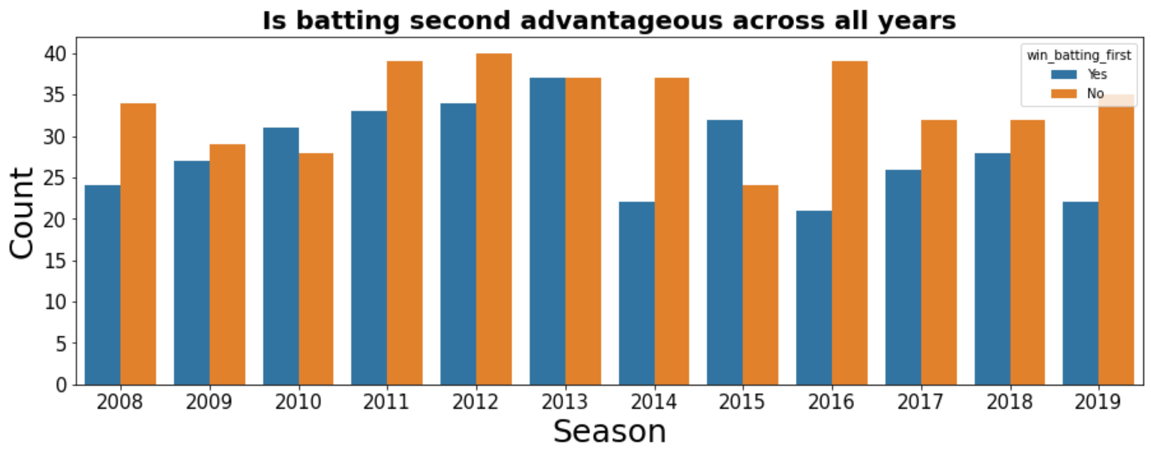
Is batting second advantageous across all years?

In [40]:

```
plt.figure(figsize = (15,5))
sns.countplot('season', data = new_matches, hue = 'win_batting_first')
plt.title("Is batting second advantageous across all years", fontsize=20,fontweight="bo
ld")
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel("Season", fontsize = 25)
plt.ylabel("Count", fontsize = 25)
```

Out[40]:

```
Text(0, 0.5, 'Count')
```



Exceptt for 2010 and 2015, in all other years it can be seen that teams batting second have won more matches

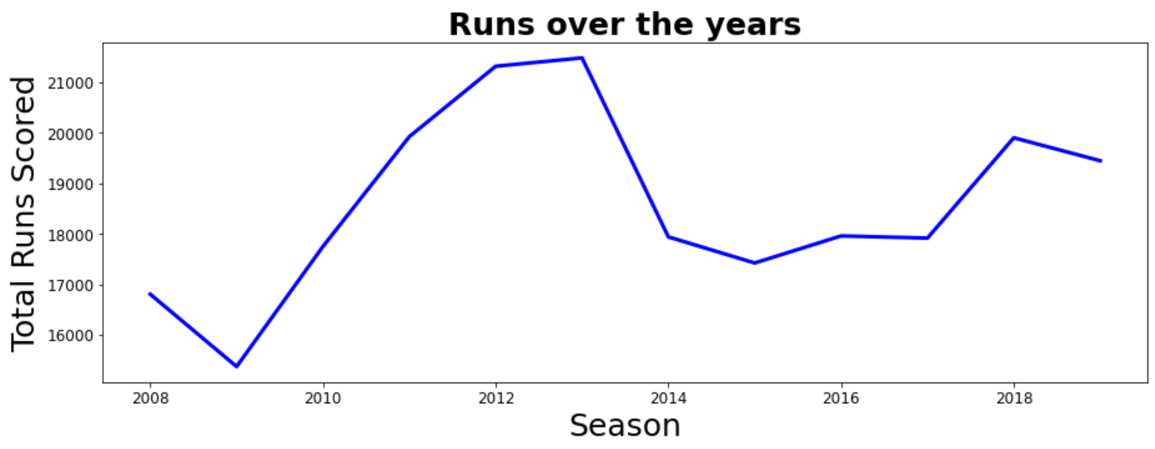
Teams total scoring runs, over the years?

In [41]:

```
merge.groupby('season')['batsman_runs'].sum().plot(kind = 'line', linewidth = 3, figsiz
e =(15,5),

color = 'blue')
plt.title("Runs over the years",fontsize= 25, fontweight = 'bold')
plt.xlabel("Season", size = 25)
plt.ylabel("Total Runs Scored", size = 25)
plt.xticks(size = 12)
plt.yticks(size = 12)
```

Out[41]:

```
(array([15000., 16000., 17000., 18000., 19000., 20000., 21000., 22000.]),
 <a list of 8 Text major ticklabel objects>)
```



Run scoring has gone up from the start of the IPL in 2008.

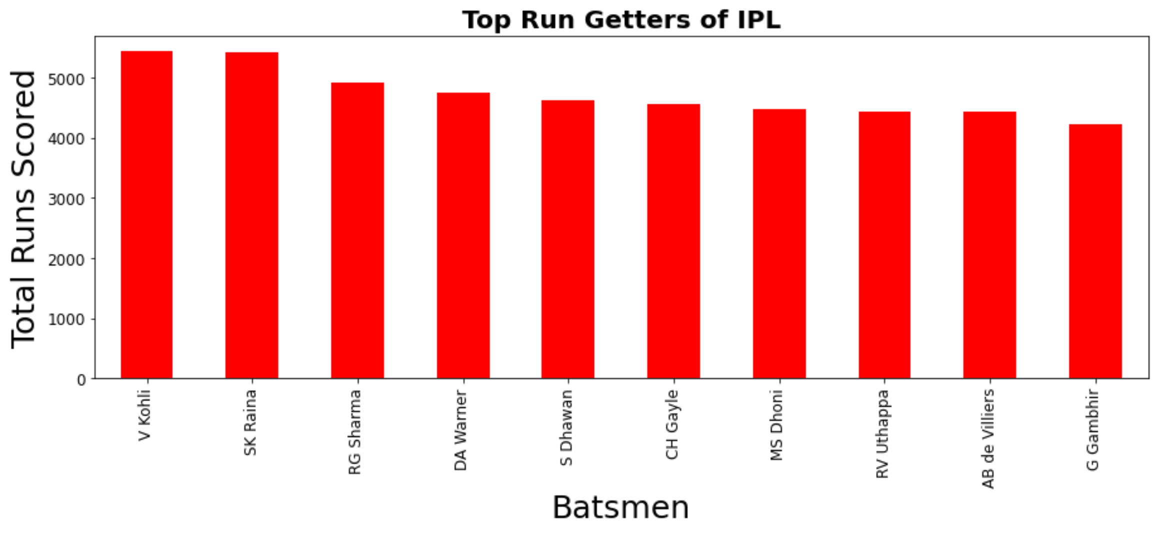
Top Run Getters of IPL.

In [42]:

```
#let's plot the top 10 run getter so far in IPL
merge.groupby('batsman')['batsman_runs'].sum().sort_values(ascending = False).head(10).
plot(kind = 'bar', color = 'red',

figsize = (15,5))
plt.title("Top Run Getters of IPL", fontsize = 20, fontweight = 'bold')
plt.xlabel("Batsmen", size = 25)
plt.ylabel("Total Runs Scored", size = 25)
plt.xticks(size = 12)
plt.yticks(size = 12)
```

Out[42]:

```
(array([   0., 1000., 2000., 3000., 4000., 5000., 6000.]),
 <a list of 7 Text major ticklabel objects>)
```



Except for MS Dhoni, all other top run getters are either openers or come in 3rd or 4th positions to bat

Suresh Raina is the highest run getter in IPL

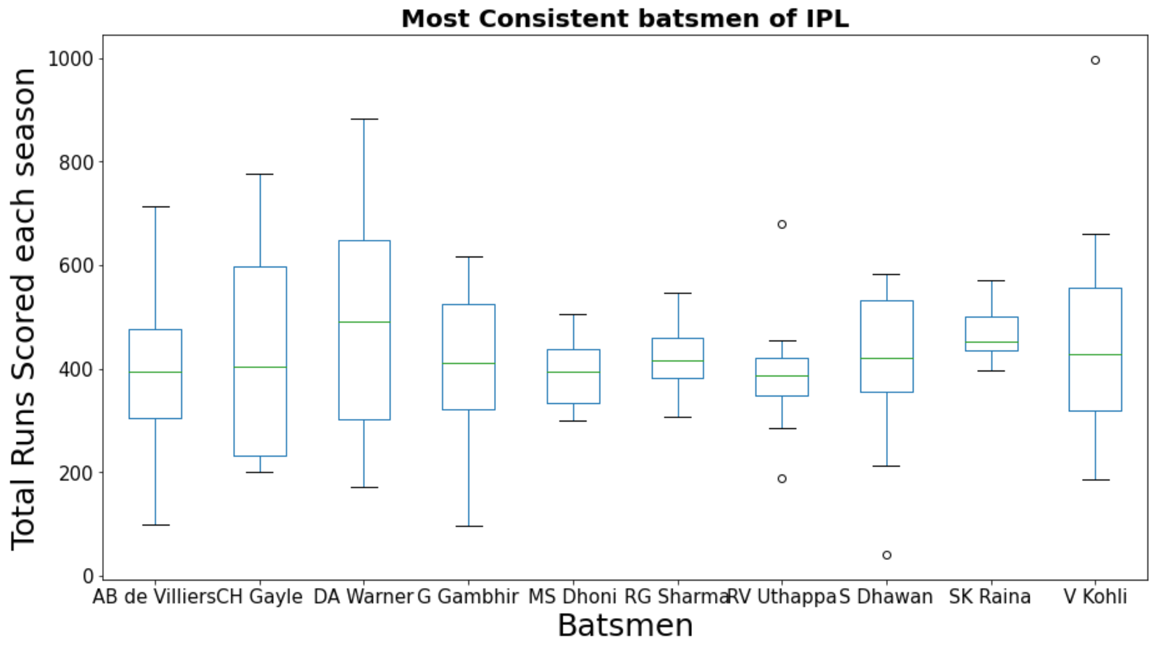
Which batsman has been most consistent among top 10 run getters?

In [43]:

```
consistent_batsman = merge[merge.batsman.isin(['SK Raina', 'V Kohli','RG Sharma','G Gam
bhir',
                                              'RV Uthappa', 'S Dhawan','CH Gayle', 'MS Dh
oni',
                                              'DA Warner', 'AB de Villiers'])][['batsman'
,'season','total_runs']]

consistent_batsman.groupby(['season','batsman'])['total_runs'].sum().unstack().plot(kin
d = 'box', figsize = (15,8))
plt.title("Most Consistent batsmen of IPL", fontsize = 20, fontweight = 'bold')
plt.xlabel("Batsmen", size = 25)
plt.ylabel("Total Runs Scored each season", size = 25)
plt.xticks(size = 15)
plt.yticks(size = 15)
```

Out[43]:

```
(array([-200.,    0.,  200.,  400.,  600.,  800., 1000., 1200.]),
 <a list of 8 Text major ticklabel objects>)
```



Median score for Raina is above all the top 10 run getters. He has the highest lowest run among all the batsmen across 11 seasons. Considering the highest and lowest season totals and spread of runs, it seems Raina has been most consistent among all.

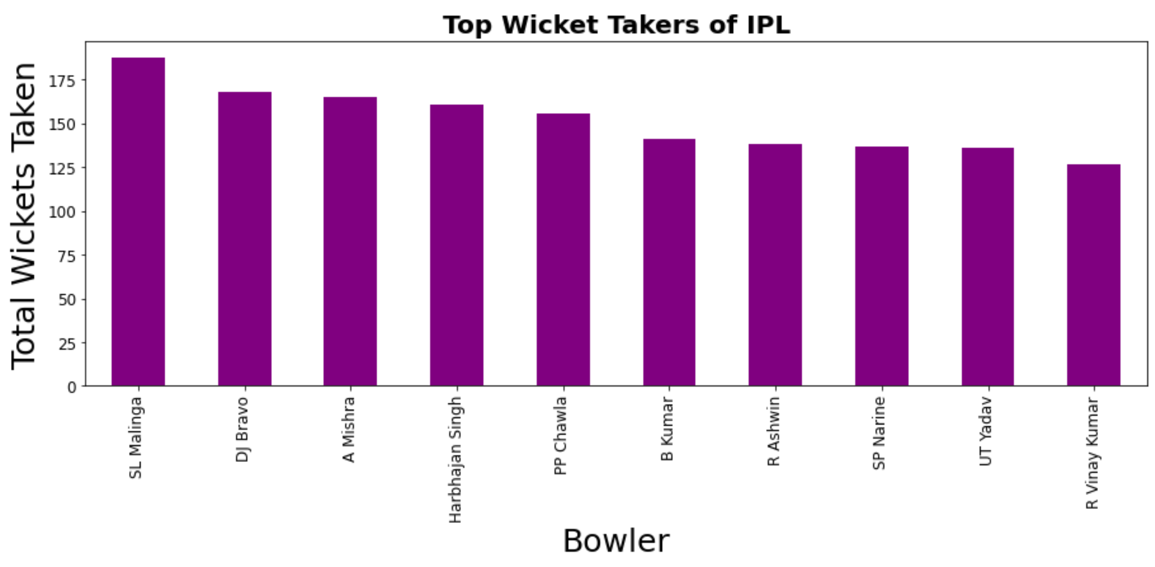
Which bowlers have performed the best?

In [44]:

```
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 = 20, fontweight = 'bold')
plt.xlabel("Bowler", size = 25)
plt.ylabel("Total Wickets Taken", size = 25)
plt.xticks(size = 12)
plt.yticks(size = 12)
```

Out[44]:

```
(array([  0.,  25.,  50.,  75., 100., 125., 150., 175., 200.]),
 <a list of 9 Text major ticklabel objects>)
```



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

In top 10 bowlers, 5 are fast and medium pacers while the other 5 are spinners

All 5 spinners are right arm spinners and 2 are leg spinners while 3 are off spinners

All 5 pacers are right arm pacers

Batsmen with the best strike rates over the years

In [46]:

```
#We will consider players who have played 10 or more seasons
no_of_balls = pd.DataFrame(merge.groupby('batsman')['ball'].count()) #total number of m
atches played by each batsman
runs = pd.DataFrame(merge.groupby('batsman')['batsman_runs'].sum()) #total runs of each
batsman
seasons = pd.DataFrame(merge.groupby('batsman')['season'].nunique()) #season = 1 implie
s played only 1 season

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

batsman_strike_rate['strike_rate'] = batsman_strike_rate['run']/batsman_strike_rate['ba
lls']*100
highest_strike_rate = batsman_strike_rate[batsman_strike_rate.season.isin([10,11])][['s
eason','batsman','strike_rate']].sort_values(by = 'strike_rate',

ascending = False)

highest_strike_rate.head(10)
```

Out[46]:

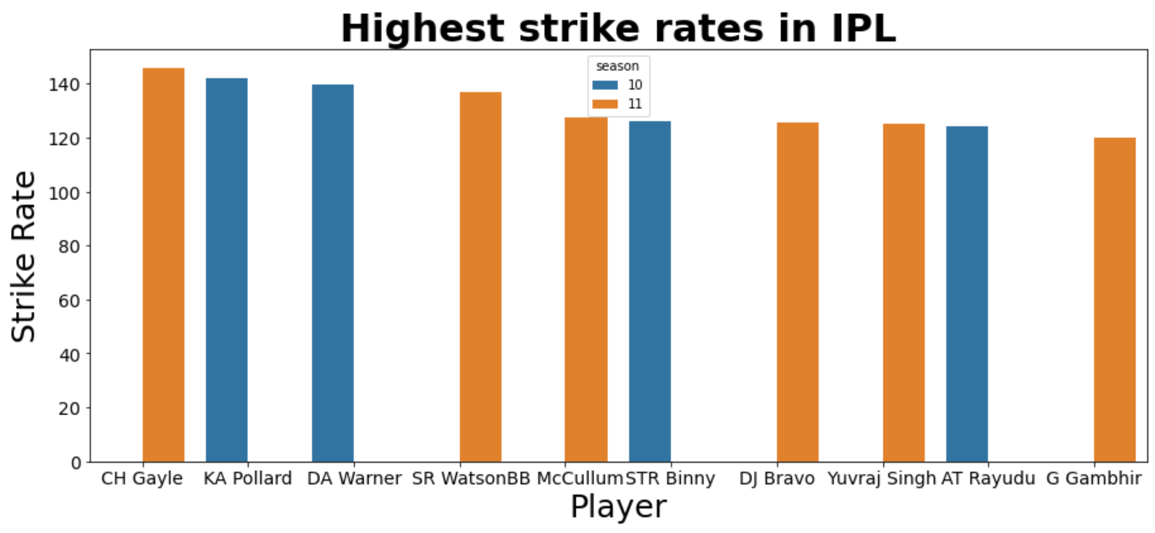
	season	batsman	strike_rate
92	11	CH Gayle	145.640370
213	10	KA Pollard	141.751527
112	10	DA Warner	139.523249
444	11	SR Watson	136.945813
72	11	BB McCullum	127.332746
449	10	STR Binny	126.000000
118	11	DJ Bravo	125.565801
514	11	Yuvraj Singh	125.283190
53	10	AT Rayudu	124.058187
147	11	G Gambhir	119.835414

In [47]:

```
plt.figure(figsize = (15,6))
sns.barplot(x='batsman', y='strike_rate', data = highest_strike_rate.head(10), hue = 's
eason')
plt.title("Highest strike rates in IPL",fontsize= 30, fontweight = 'bold')
plt.xlabel("Player", size = 25)
plt.ylabel("Strike Rate", size = 25)
plt.xticks(size = 14)
plt.yticks(size = 14)
```

Out[47]:

```
(array([  0.,  20.,  40.,  60.,  80., 100., 120., 140., 160.]),
 <a list of 9 Text major ticklabel objects>)
```



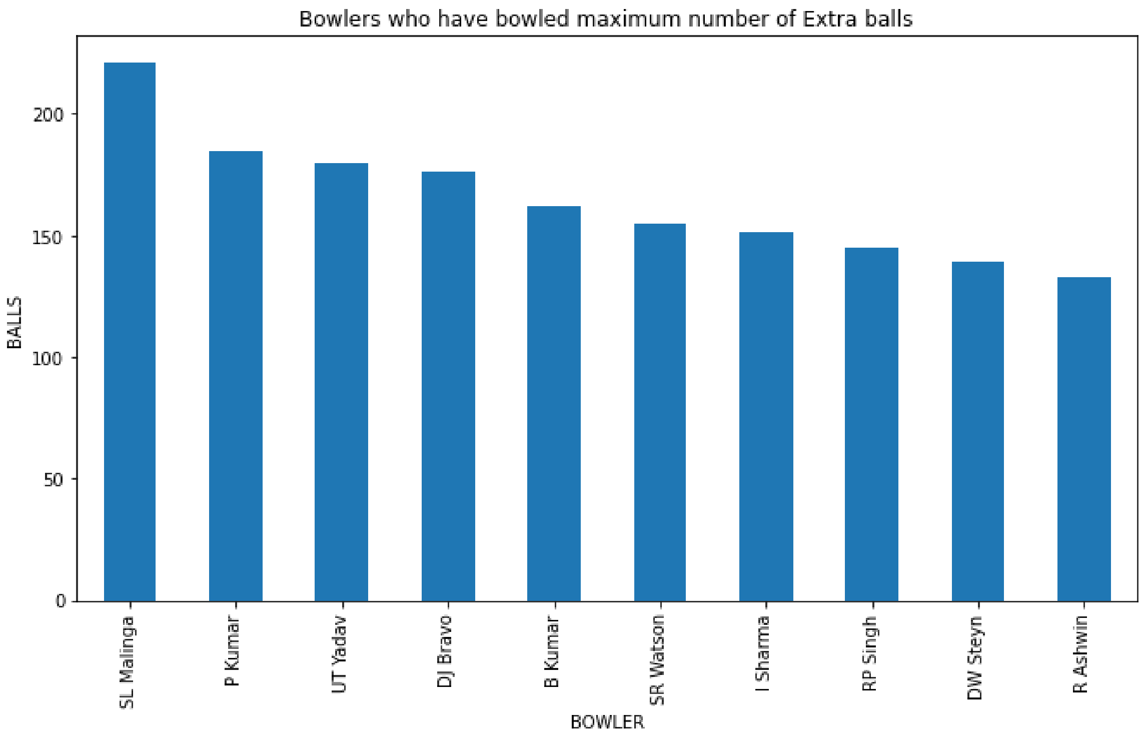
AB de Villiers, Gayle have the highest strike rates in IPL. They are the big hitters and can win any match on their day

One surprise here is that Harbhajan Singh who is a bowler has a strike rate of 130+ and comes before Rohit Sharma in ranking

Bowlers with maximum number of extras.

In [48]:

```
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')
plt.show()

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



Out[48]:

	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

Which bowlers have picked up wickets more frequently?

In [49]:

```
#strike_rate = balls bowled by wickets taken
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 [50]:

```
bowler_strike_rate['strike_rate'] = bowler_strike_rate['balls']/bowler_strike_rate['wic
kets']
def highlight_cols(s):
    color = 'skyblue'
    return 'background-color: %s' % color
#Strike rate for bowlers who have taken more than 50 wickets
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[50]:

	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

Q1. As a sports analysts, The most successful teams, players & factors contributing win or loss of a team:

- Mumbai Indians is the most successful team in IPL and has won the most number of toss.
- There were more matches won by chasing the total(419 matches) than defending(350 matches).
- When defending a total, the biggest victory was by 146 runs(Mumbai Indians defeated Delhi Daredevils by 146 runs on 06 May 2017 at Feroz Shah Kotla stadium, Delhi).
- When chasing a target, the biggest victory was by 10 wickets(without losing any wickets) and there were 11 such instances.
- The Mumbai city has hosted the most number of IPL matches.
- Chris Gayle has won the maximum number of player of the match title.
- S. Ravi(Sundaram Ravi) has officiated the most number of IPL matches on-field.
- Eden Gardens has hosted the maximum number of IPL matches.
- If a team wins a toss choose to field first as it has highest probablity of winning

Q2. Teams or Players a company should endorse for its products

- If the franchise is looking for a consistant batsman who needs to score good amount of runs then go for V Kohli, S Raina, Rohit Sharma , David Warner...
- If the franchise is looking for a game changing batsman then go for Chris Gayle, AB deVillers, R Sharma , MS Dhoni...
- If the franchise is looking for a batsman who could score good amount of runs every match the go for DA Warner, CH Gayle, V Kohli,AB de Villiers,S Dhawan
- If the franchise needs the best finisher in lower order having good strike rate then go for CH Gayle,KA Pollard, DA Warner,SR Watson,BB McCullum
- If the franchise need a experienced bowler then go for Harbhajan Singh ,A Mishra,PP Chawla ,R Ashwin,SL Malinga,DJ Bravo
- If the franchise need a wicket taking bowler then go for SL Malinga,DJ Bravo,A Mishra ,Harbhajan Singh, PP Chawla
- If the franchise need a bowler bowling most number of dot balls then go for Harbhajan Singh,SL Malinga,B Kumar,A Mishra,PP Chawla
- If the franchise need a bowler with good economy then go for DW Steyn ,M Muralitharan ,R Ashwin,SP Narine ,Harbhajan Singh

MERCI

In [ ]: