Author: ATANU DAS

Data Science And Business Analytics Intern

GRIP: The Sparks Foundation

Task 5: Exploratory Data Analysis- Sports

Level - Advanced

- Problem Statement: Perform Exploratory Data Analysis on 'Indian Premiere League'
- As a sports analysts, find out the most successful teams, players and factors contributing win or loss of a team.
- Suggest teams or players a company should endorse for its products.

Importing the Libraries

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

Loading 1st Dataset

In [ ]:
```
matches = pd.read_csv("matches.csv")
```

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

Out[ ]:

id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wicke

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wicke
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	

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

Out[ ]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by
751	11347	2019	Mumbai	05/05/19	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumbai Indians	0	
752	11412	2019	Chennai	07/05/19	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumbai Indians	0	
753	11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	0	Delhi Capitals	0	
754	11414	2019	Visakhapatnam	10/05/19	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chennai Super Kings	0	

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by
755	11415	2019	Hyderabad	12/05/19	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	Mumbai Indians	1	

Data information

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

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

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

Out[ ]:

	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

Loading 2nd dataset

In [ ]:
```
deliveries=pd.read_csv("deliveries.csv")
```

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

Out[ ]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	wide_runs	bye_runs	legbye_runs	
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	1.0	DA Warner	S Dhawan	TS Mills	0.0	0.0	0.0	0.0	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	2.0	DA Warner	S Dhawan	TS Mills	0.0	0.0	0.0	0.0	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	3.0	DA Warner	S Dhawan	TS Mills	0.0	0.0	0.0	0.0	
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	4.0	DA Warner	S Dhawan	TS Mills	0.0	0.0	0.0	0.0	

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	wide_runs	bye_runs	legbye_runs	
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	5.0	DA Warner	S Dhawan	TS Mills	0.0	2.0	0.0	0.0	

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

Out[ ]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	wide_runs	bye_runs	legbye_r
93052	393	2	Pune Warriors	Rajasthan Royals	10.0	2.0	LRPL Taylor	AJ Finch	SK Trivedi	0.0	0.0	0.0	
93053	393	2	Pune Warriors	Rajasthan Royals	10.0	3.0	AJ Finch	LRPL Taylor	SK Trivedi	0.0	0.0	0.0	
93054	393	2	Pune Warriors	Rajasthan Royals	10.0	4.0	AJ Finch	LRPL Taylor	SK Trivedi	0.0	0.0	0.0	
93055	393	2	Pune Warriors	Rajasthan Royals	10.0	5.0	LRPL Taylor	AJ Finch	SK Trivedi	0.0	0.0	0.0	
93056	393	2	Pune Warr	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

Data information

In [ ]:
```
deliveries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93057 entries, 0 to 93056
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   match_id          93057 non-null  int64  
 1   inning            93057 non-null  int64  
 2   batting_team      93057 non-null  object 
 3   bowling_team      93056 non-null  object 
 4   over              93056 non-null  float64
 5   ball              93056 non-null  float64
```

```
 6   batsman           93056 non-null  object
 7   non_striker       93056 non-null  object
 8   bowler            93056 non-null  object
 9   is_super_over     93056 non-null  float64
 10  wide_runs         93056 non-null  float64
 11  bye_runs          93056 non-null  float64
 12  legbye_runs       93056 non-null  float64
 13  noball_runs       93056 non-null  float64
 14  penalty_runs      93056 non-null  float64
 15  batsman_runs      93056 non-null  float64
 16  extra_runs        93056 non-null  float64
 17  total_runs        93056 non-null  float64
 18  player_dismissed  4645 non-null   object
 19  dismissal_kind    4645 non-null   object
 20  fielder           3338 non-null   object
dtypes: float64(11), int64(2), object(8)
memory usage: 14.9+ MB
```

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

Out[ ]:
```
(93057, 21)
```

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

Out[ ]:

	match_id	inning	over	ball	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs	penalty_
count	93057.000000	93057.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.00
mean	197.532996	1.481844	10.126580	3.619810	0.000494	0.038547	0.005384	0.022739	0.004664	0.00
std	113.371454	0.501605	5.670758	1.810329	0.022228	0.261940	0.118391	0.201970	0.075182	0.01
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	100.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	197.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	296.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
max	393.000000	4.000000	20.000000	9.000000	1.000000	5.000000	4.000000	5.000000	5.000000	5.00

Merging the two datasets for better insights from the data

In [ ]:
```
merge = pd.merge(deliveries,matches, left_on='match_id', right_on ='id')
merge.head(2)
```

Out[ ]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	wide_runs	bye_runs	legbye_runs
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	1.0	DA Warner	S Dhawan	TS Mills	0.0	0.0	0.0	0.0
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1.0	2.0	DA Warner	S Dhawan	TS Mills	0.0	0.0	0.0	0.0

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 93057 entries, 0 to 93056
Data columns (total 39 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   match_id          93057 non-null  int64
 1   inning            93057 non-null  int64
 2   batting_team      93057 non-null  object
 3   bowling_team      93056 non-null  object
 4   over              93056 non-null  float64
 5   ball              93056 non-null  float64
 6   batsman           93056 non-null  object
 7   non_striker       93056 non-null  object
 8   bowler            93056 non-null  object
 9   is_super_over     93056 non-null  float64
 10  wide_runs         93056 non-null  float64
 11  bye_runs          93056 non-null  float64
 12  legbye_runs       93056 non-null  float64
 13  noball_runs       93056 non-null  float64
 14  penalty_runs      93056 non-null  float64
 15  batsman_runs      93056 non-null  float64
 16  extra_runs        93056 non-null  float64
```

```
 17  total_runs        93056 non-null  float64
 18  player_dismissed  4645 non-null   object
 19  dismissal_kind    4645 non-null   object
 20  fielder           3338 non-null   object
 21  id                93057 non-null  int64
 22  season            93057 non-null  int64
 23  city              93057 non-null  object
 24  date              93057 non-null  object
 25  team1             93057 non-null  object
 26  team2             93057 non-null  object
 27  toss_winner       93057 non-null  object
 28  toss_decision     93057 non-null  object
 29  result            93057 non-null  object
 30  dl_applied        93057 non-null  int64
 31  winner            92994 non-null  object
 32  win_by_runs       93057 non-null  int64
 33  win_by_wickets    93057 non-null  int64
 34  player_of_match   92994 non-null  object
 35  venue             93057 non-null  object
 36  umpire1           92809 non-null  object
 37  umpire2           92809 non-null  object
 38  umpire3           0 non-null      object
dtypes: float64(11), int64(7), object(21)
memory usage: 28.4+ MB
```

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

Out[ ]:

	match_id	inning	over	ball	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs	penalty_
count	93057.000000	93057.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.000000	93056.00
mean	197.532996	1.481844	10.126580	3.619810	0.000494	0.038547	0.005384	0.022739	0.004664	0.00
std	113.371454	0.501605	5.670758	1.810329	0.022228	0.261940	0.118391	0.201970	0.075182	0.01
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	100.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	197.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	296.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
max	393.000000	4.000000	20.000000	9.000000	1.000000	5.000000	4.000000	5.000000	5.000000	5.00

In [ ]:
```
matches.id.is_unique
```

Out[ ]:
```
True
```

Since id is unique we can set this as our index

In [ ]:
```
matches.set_index('id', inplace=True)
```

Summary statistics of matches data

In [ ]:
```
matches.describe(include='all')
```

Out[ ]:

	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	p
count	756.000000	749	756	756	756	756	756	756	756.000000	752	756.000000	756.000000	
unique	NaN	32	546	15	15	15	2	3	NaN	15	NaN	NaN	
top	NaN	Mumbai	2013-04-20	Mumbai Indians	Kolkata Knight Riders	Mumbai Indians	field	normal	NaN	Mumbai Indians	NaN	NaN	
freq	NaN	101	2	101	95	98	463	743	NaN	109	NaN	NaN	
mean	2013.444444	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.025132	NaN	13.283069	3.350529	
std	3.366895	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.156630	NaN	23.471144	3.387963	
min	2008.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	0.000000	0.000000	
25%	2011.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	0.000000	0.000000	
50%	2013.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	0.000000	4.000000	
75%	2016.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	19.000000	6.000000	
max	2019.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	146.000000	10.000000	

Data Preprocessing

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

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

Out[ ]:

id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets
1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0
2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7
3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10
4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6
5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0

In [ ]:
```
matches.isnull().sum()
```

Out[ ]:
```
season          0
city            7
date            0
team1           0
team2           0
toss_winner     0
toss_decision   0
result          0
dl_applied      0
winner          4
win_by_runs     0
```

```
win_by_wickets       0
player_of_match      4
venue                0
umpire1              2
umpire2              2
umpire3            637
dtype: int64
```

This shows:

1. City has 7 missing values
2. Umpire 1, 2 and 3 have 2, 2 and 637 missing values
3. City has 32 distict values whereas venue has 41 distinct values.

Filling in the missing values:

Finding the venues corresponding to values of city which are empty

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

Out[ ]:

id	city	venue
462	NaN	Dubai International Cricket Stadium
463	NaN	Dubai International Cricket Stadium
467	NaN	Dubai International Cricket Stadium
469	NaN	Dubai International Cricket Stadium
470	NaN	Dubai International Cricket Stadium
475	NaN	Dubai International Cricket Stadium
477	NaN	Dubai International Cricket Stadium

Since the venue is Dubai International Cricket Stadium , filling the null values to Dubai.

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

Umpire 1 and 2 having 2 missing values each.

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

Out[ ]:

id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by
5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	
11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	0	Delhi Capitals	0	

Umpire 3 has 637 missing values hence dropping that column.

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

City has 32 distict values whereas venue has 41 distinct values. Finding out which cities have multiple venues:

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

Out[ ]:

city	venue	season
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

city	venue	season
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

city	venue	season
	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
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

1. Bengaluru and Bangalore both are in the data when they are same. So we need to keep one of them
2. 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

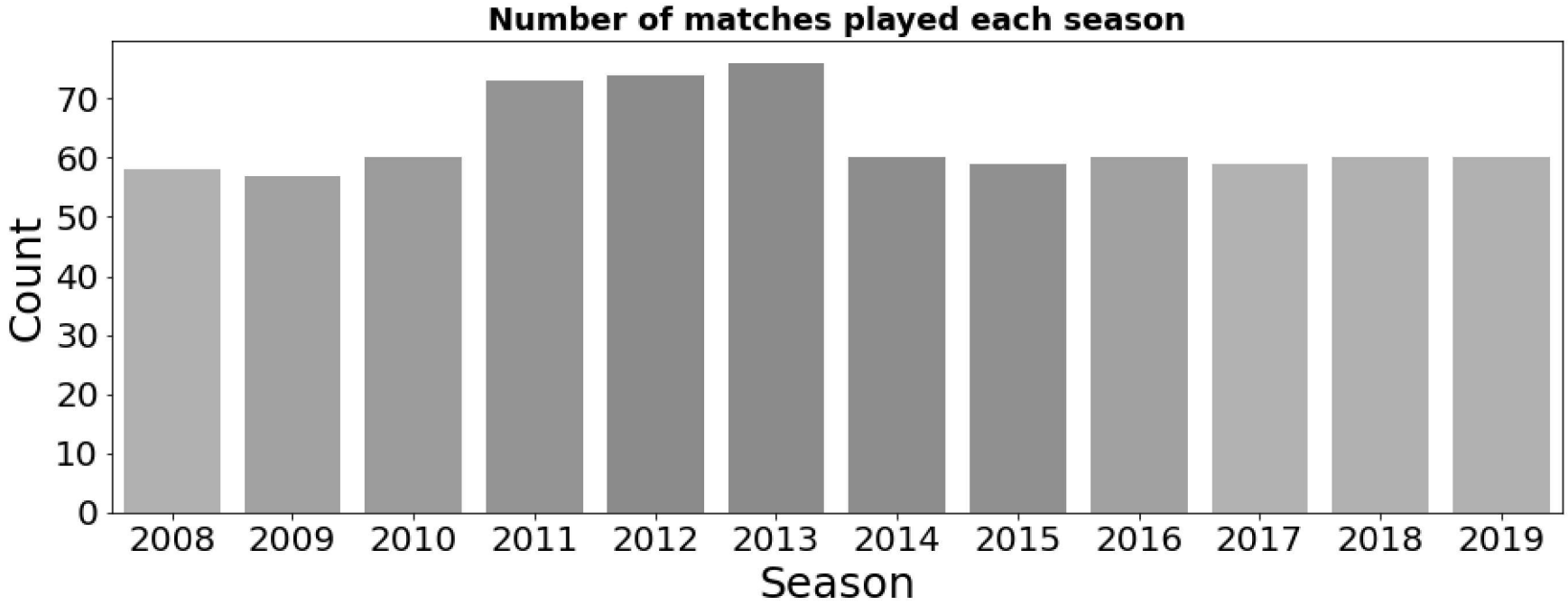
NUMBER OF MATCHES PLAYED IN EACH SEASON:

In [ ]:
```
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)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: 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 expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

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



- 2011-13 have more number of matches being played compared to other seasons.

NUMBER OF MATCHES PLAYED IN EACH CITY

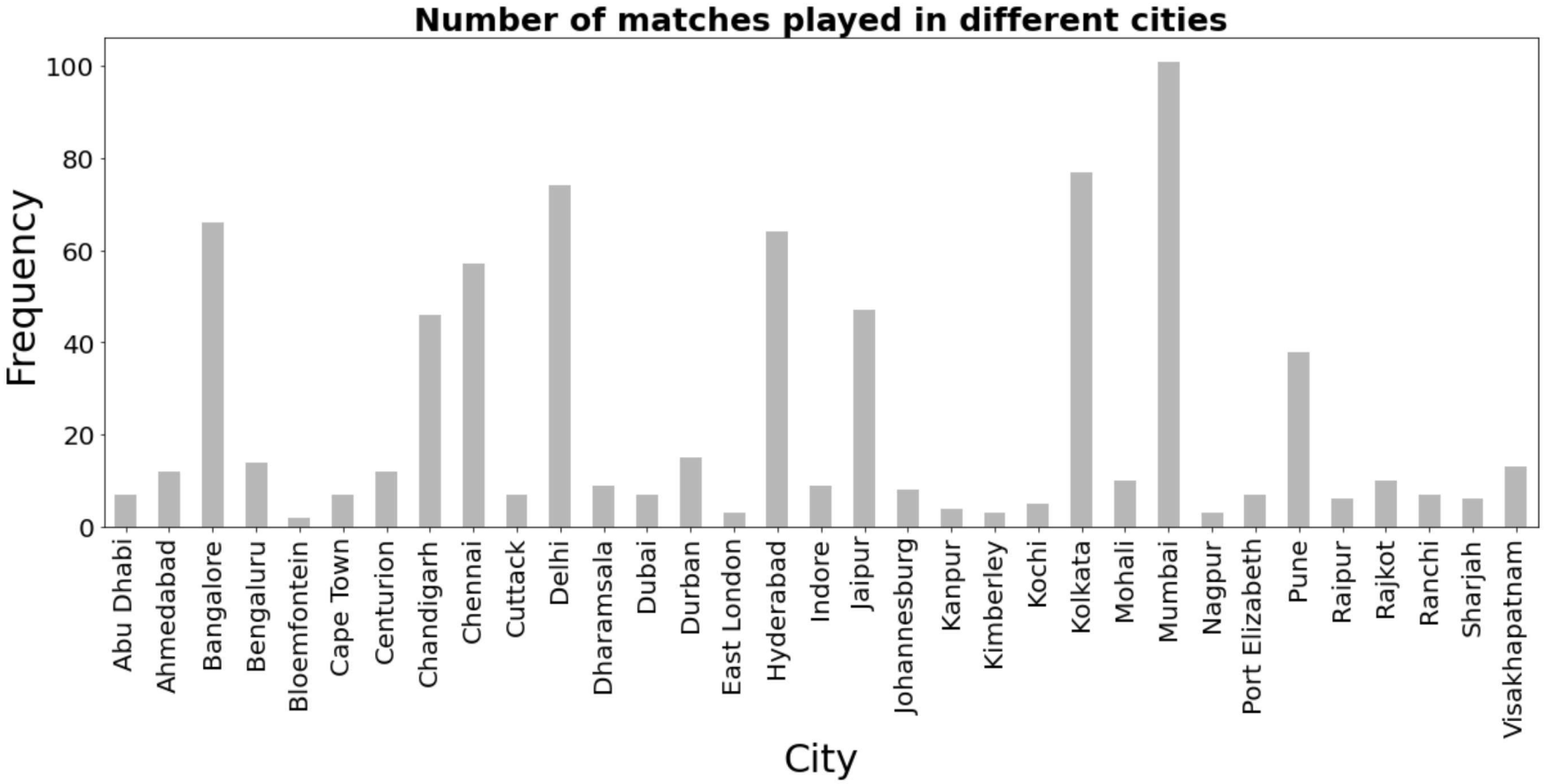
In [ ]:
```
#Plotting venues along with cities
v = pd.crosstab(matches['city'],matches['venue'])

#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', color='lightsalmon')
plt.title("Number of matches played 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[ ]:
```
(array([  0.,  20.,  40.,  60.,  80., 100., 120.]),
 <a list of 7 Text major ticklabel objects>)
```

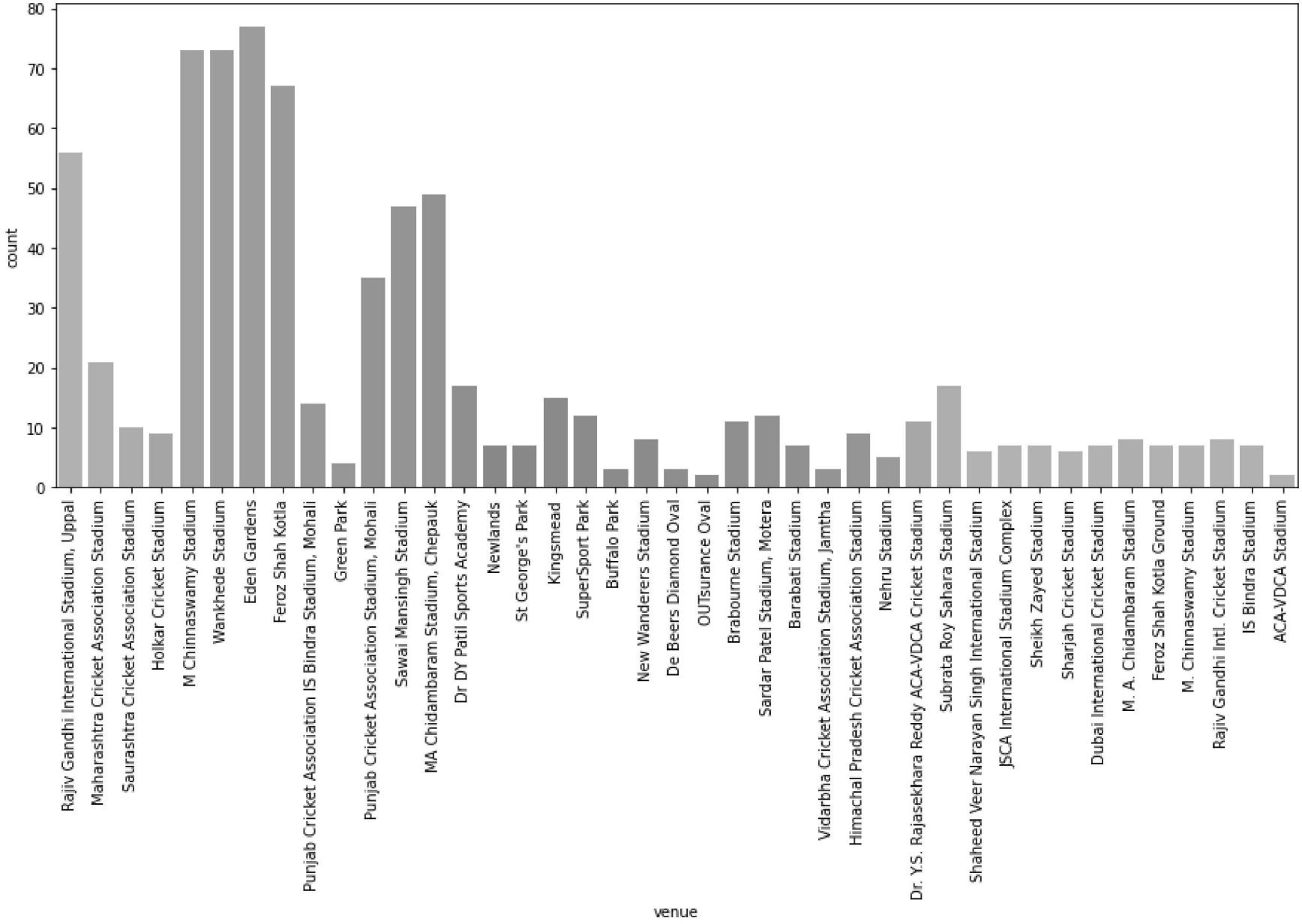


- Mumbai has hosted most number of matches.

VENUE WITH MOST NUMBER OF MATCHES

In [ ]:
```
plt.figure(figsize = (15,6))
plt.xticks(rotation=90)
sns.countplot(x = 'venue', data = matches)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e289f3c90>
```



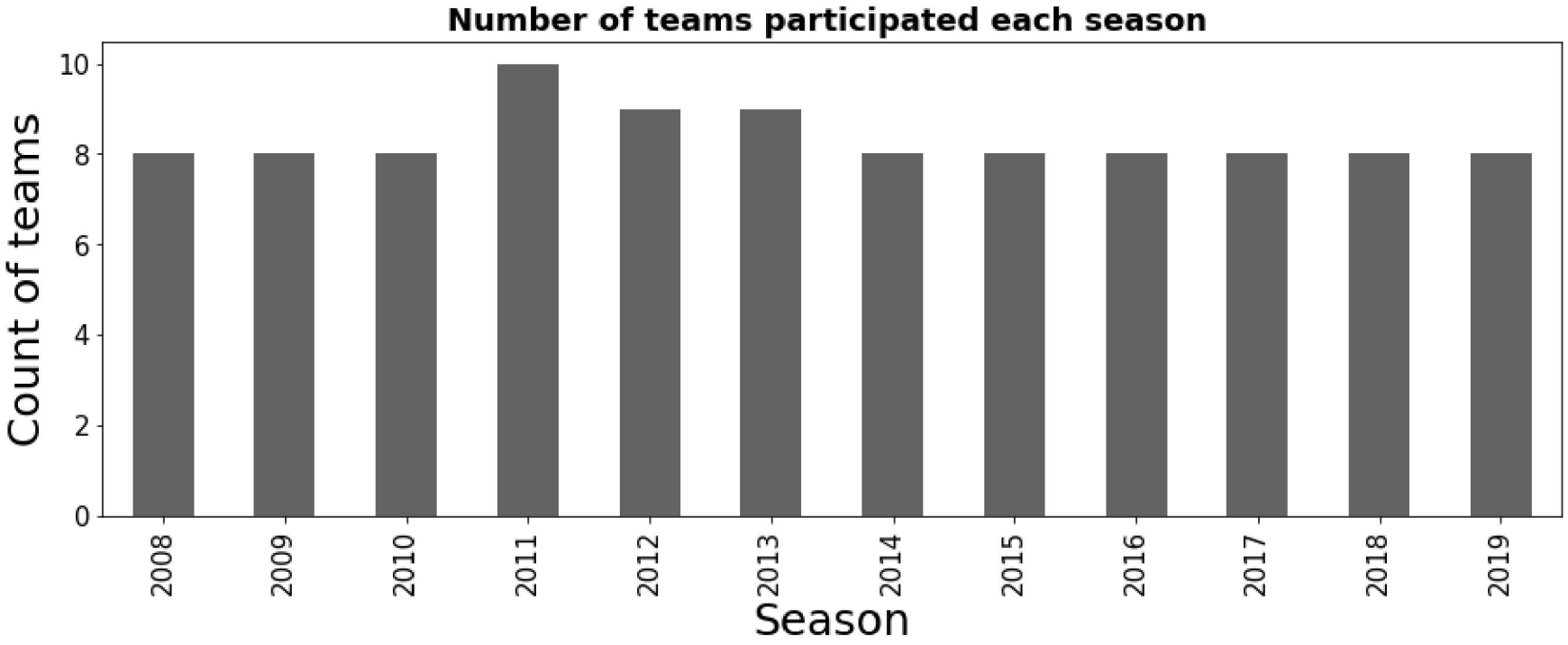
- Eden garden has hosted most number of matches followed by M Chinnaswamy stadium and Wankhede Stadium

NUMBER OF TEAMS PARTICIPATED IN EACH SEASON

In [ ]:

```
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[ ]:
```
(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 explaining comparatively high number of matches being played in these three seasons.

TEAM WITH MAXIMUM NUMBER OF WINS

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

In [ ]:
```
#dictionaries to get winners to each season
```

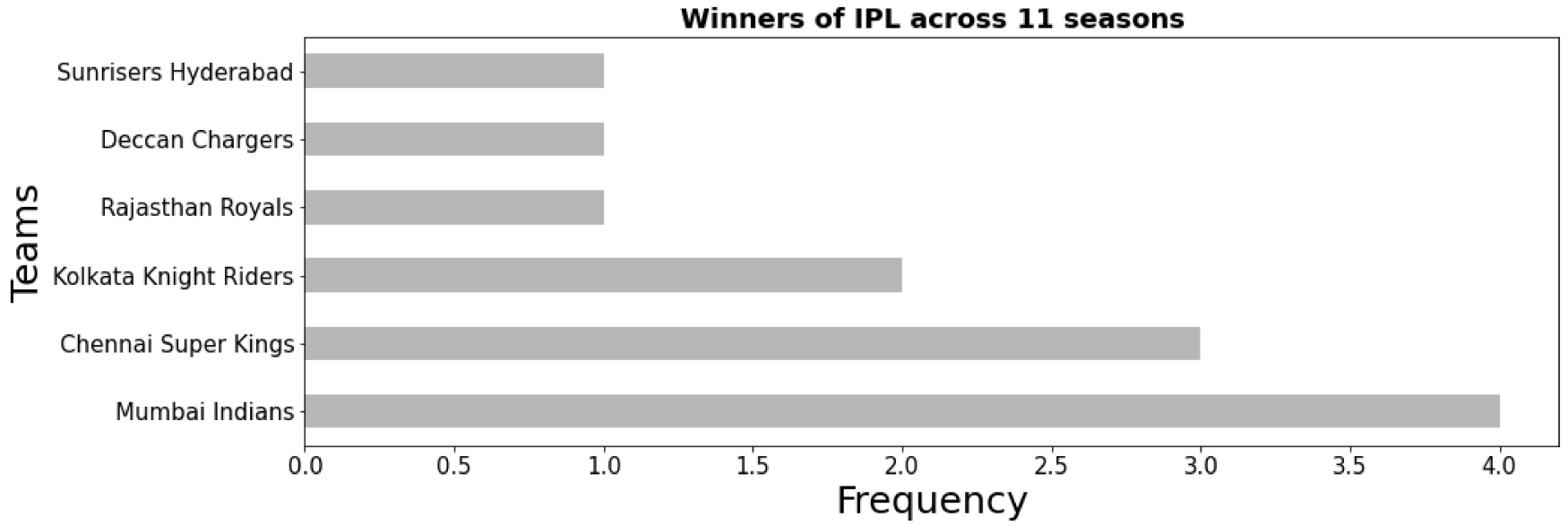
```
winners_team = {}
for i in sorted(winning_teams.season.unique()):
    winners_team[i] = winning_teams[winning_teams.season == i]['winner'].tail(1).values[0]

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

In [ ]:

```
winners_of_IPL['team'].value_counts().plot(kind = 'barh', figsize = (15,5), color = 'lightsalmon')
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[ ]: (array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)



- Mumbai Indians won most number of matches followed by Chennai Super Kings and then kolkata knight Riders.

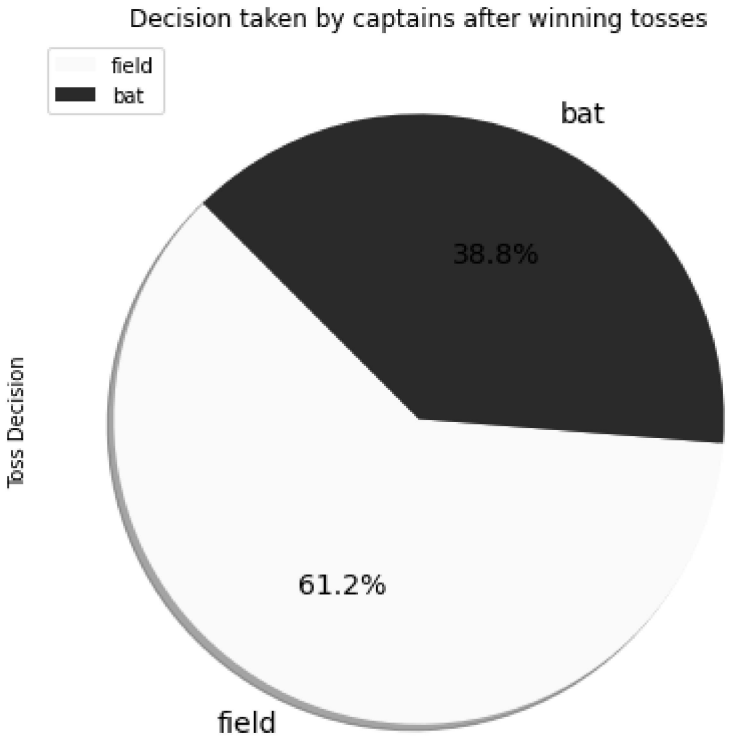
DECISION TAKEN AFTER WINING THE TOSS

In [ ]:

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

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

Out[ ]:
```
Text(0.5, 1.0, 'Decision taken by captains after winning tosses')
```



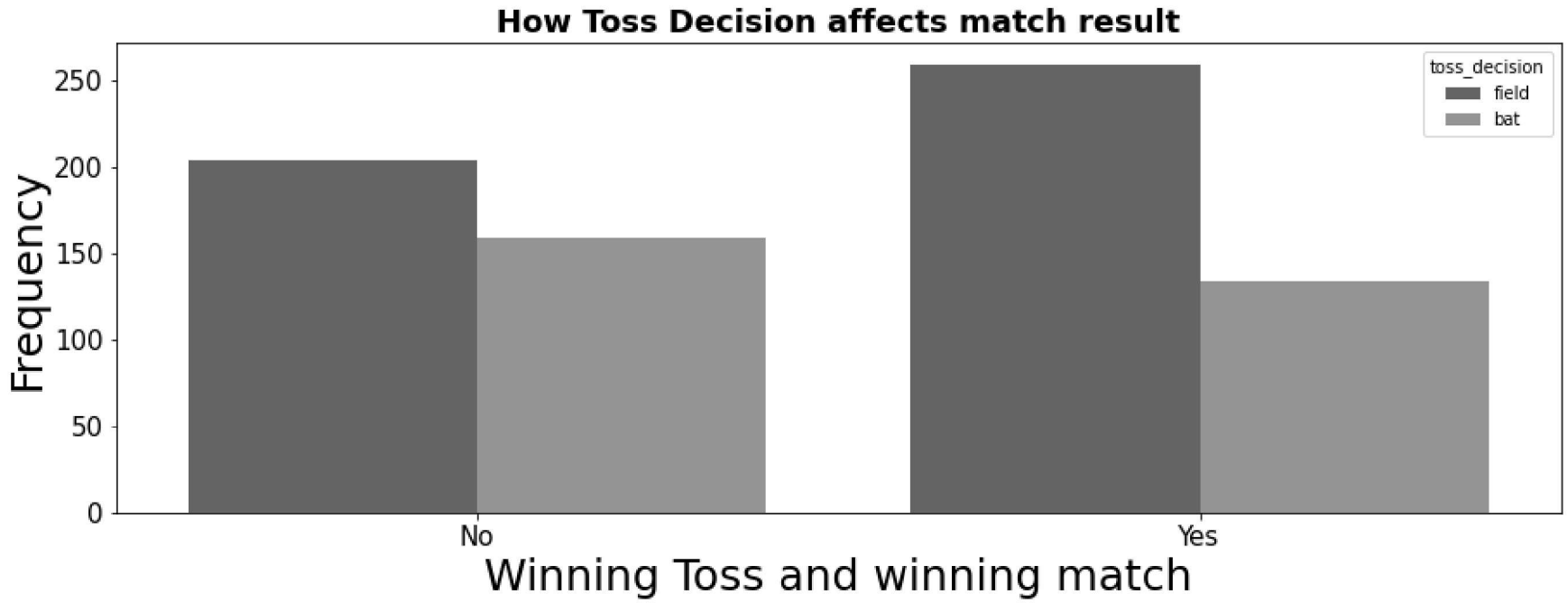
- 61.2% times teams who have won tosses have decided to chase down

How toss decision affects match results?

In [ ]:
```
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)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: 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 expl
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

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



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

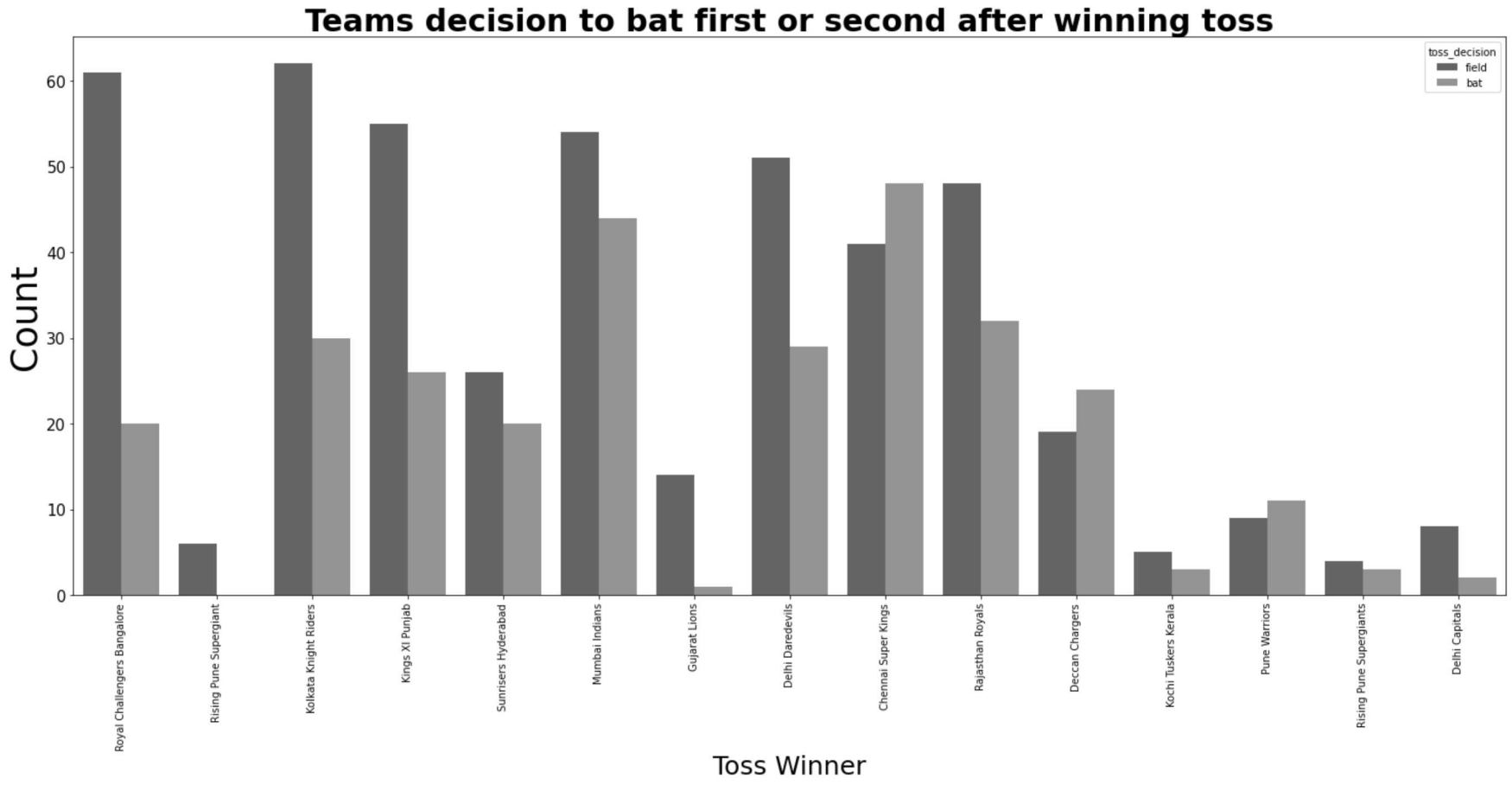
INDIVIDUAL TEAMS DECISION AFTER WINNING THE TOSS

In [ ]:
```
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, fontweight = 'bold')
plt.xticks(size = 10, rotation=90)
plt.yticks(size = 15)
plt.xlabel("Toss Winner", size = 25)
plt.ylabel("Count", size = 35)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: 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 expl
```

```
icit keyword will result in an error or misinterpretation.
  FutureWarning
```

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

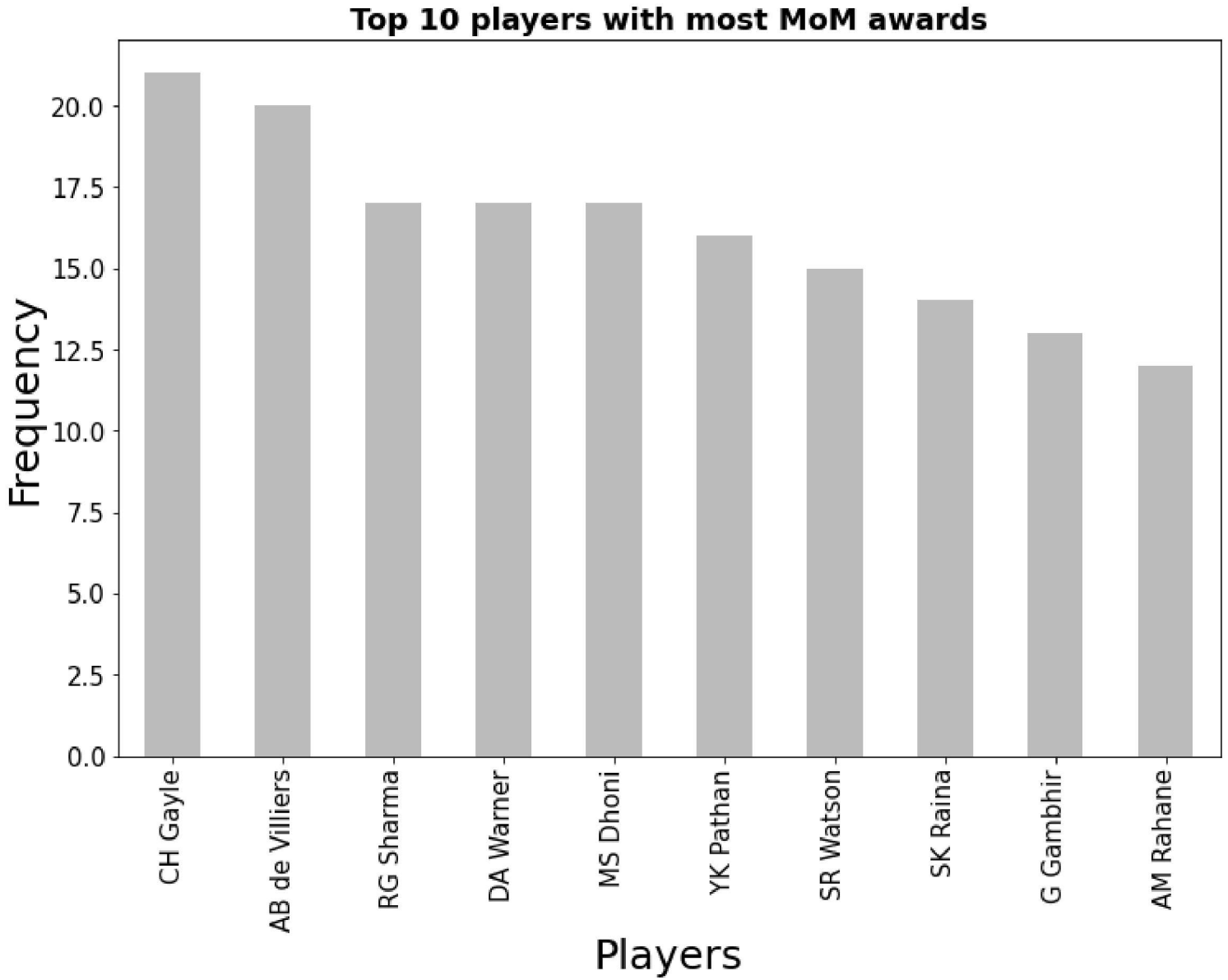


- Most team chooses to field first on winning the toss except for Chennai Super Kings, Deccan Charges and Pune Warriors

PLAYERS WITH MOST MOM AWARDS

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

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



- Chris Gayle has so far won the most number of MoM awards followed by AB de Villiers.
- All Top 10 players are batsmen. This shows IPL batsmen have mostly dictated the matches.

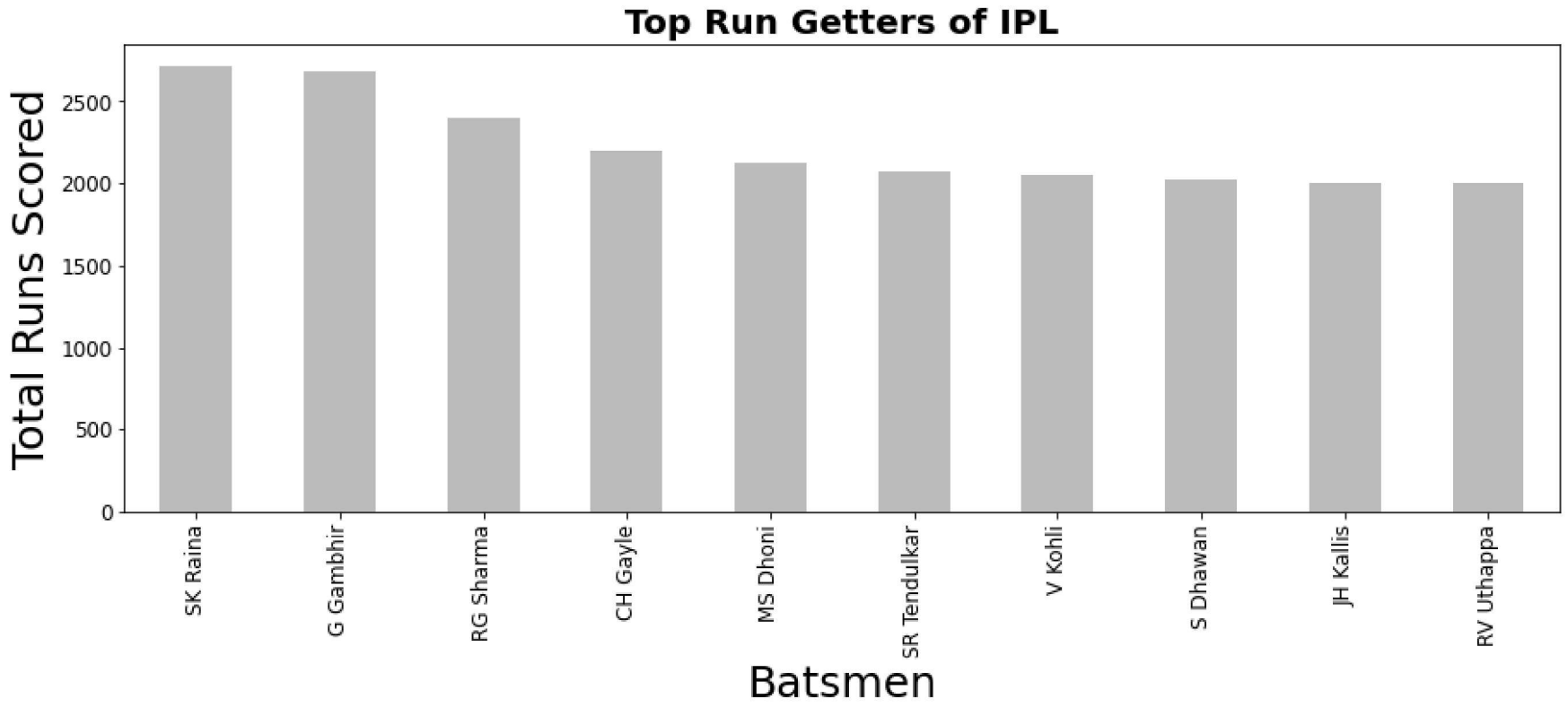
TOP RUN GETTERS

In [ ]:
```
merge.groupby('batsman')['batsman_runs'].sum().sort_values(ascending = False).head(10)
```

Out[ ]:
```
batsman
SK Raina        2714.0
G Gambhir       2685.0
RG Sharma       2401.0
CH Gayle        2205.0
MS Dhoni        2123.0
SR Tendulkar    2071.0
V Kohli         2057.0
S Dhawan        2019.0
JH Kallis       2004.0
RV Uthappa      2001.0
Name: batsman_runs, dtype: float64
```

In [ ]:
```
#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 = 'skyblu
                                                                                           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[ ]:
```
(array([   0.,  500., 1000., 1500., 2000., 2500., 3000.]),
 <a list of 7 Text major ticklabel objects>)
```



- SK Raina has scored most number of runs(2714) followed by G Gambhir (2685) and then RG Sharma (2401).
- Except for MS Dhoni, all other top run getters are either openers or come in 3rd or 4th positions to bat

MOST CONSISTENT BATSMAN

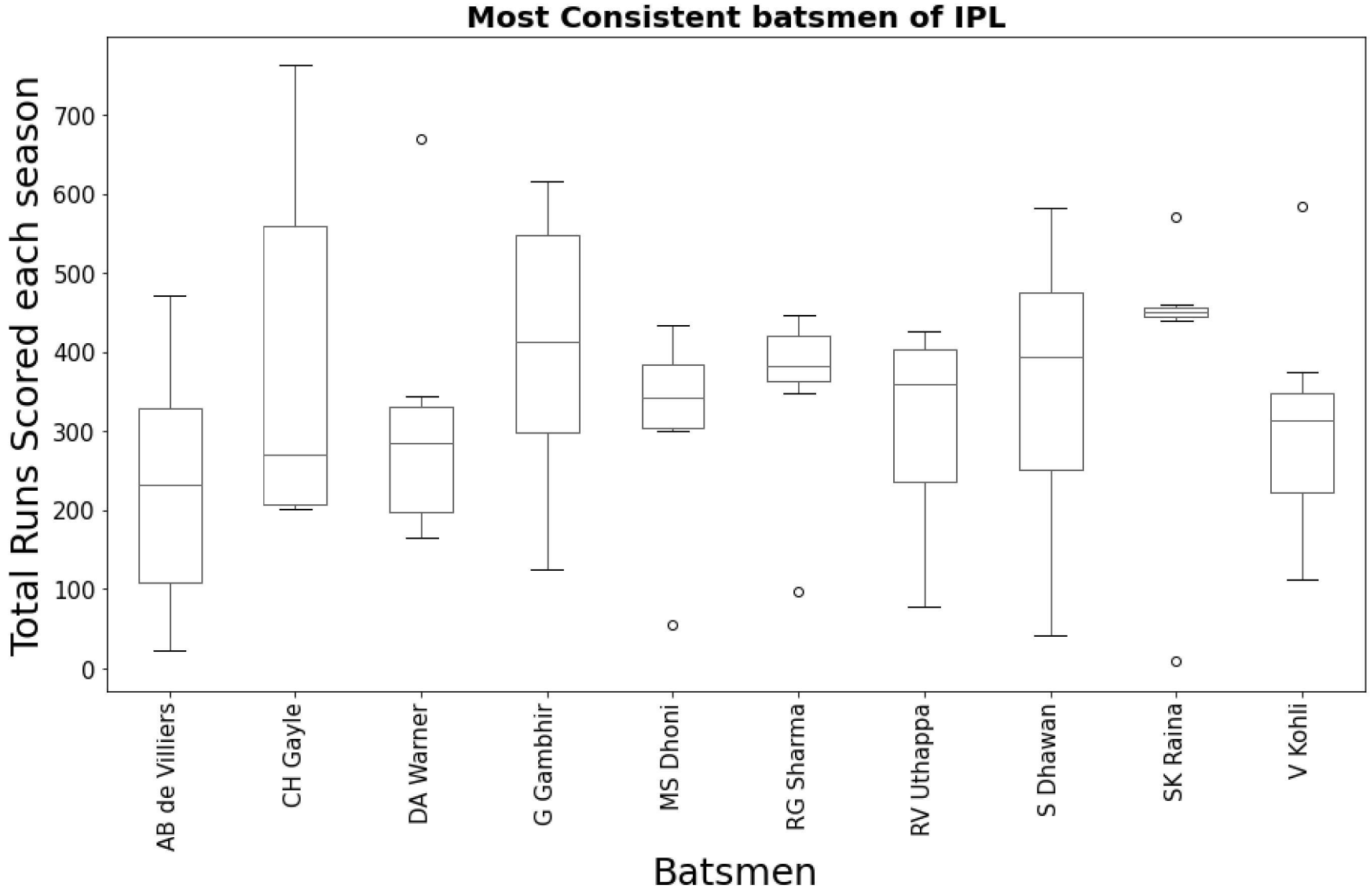
In [ ]:
```
consistent_batsman = merge[merge.batsman.isin(['SK Raina', 'V Kohli','RG Sharma','G Gambhir',
                                               'RV Uthappa', 'S Dhawan','CH Gayle', 'MS Dhoni',
                                               'DA Warner', 'AB de Villiers'])][['batsman','season','total_runs']]

consistent_batsman.groupby(['season','batsman'])['total_runs'].sum().unstack().plot(kind = 'box', figsize = (15,8))
plt.title("Most Consistent batsmen of IPL", fontsize = 20, fontweight = 'bold')
plt.xlabel("Batsmen", size = 25)
plt.ylabel("Total Runs Scored each season", size = 25)
plt.xticks(size = 15, rotation=90)
plt.yticks(size = 15)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from rag
```

```
ged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprec
ated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

Out[ ]:
```
(array([-100.,    0.,  100.,  200.,  300.,  400.,  500.,  600.,  700.,
        800.]), <a list of 10 Text major ticklabel objects>)
```

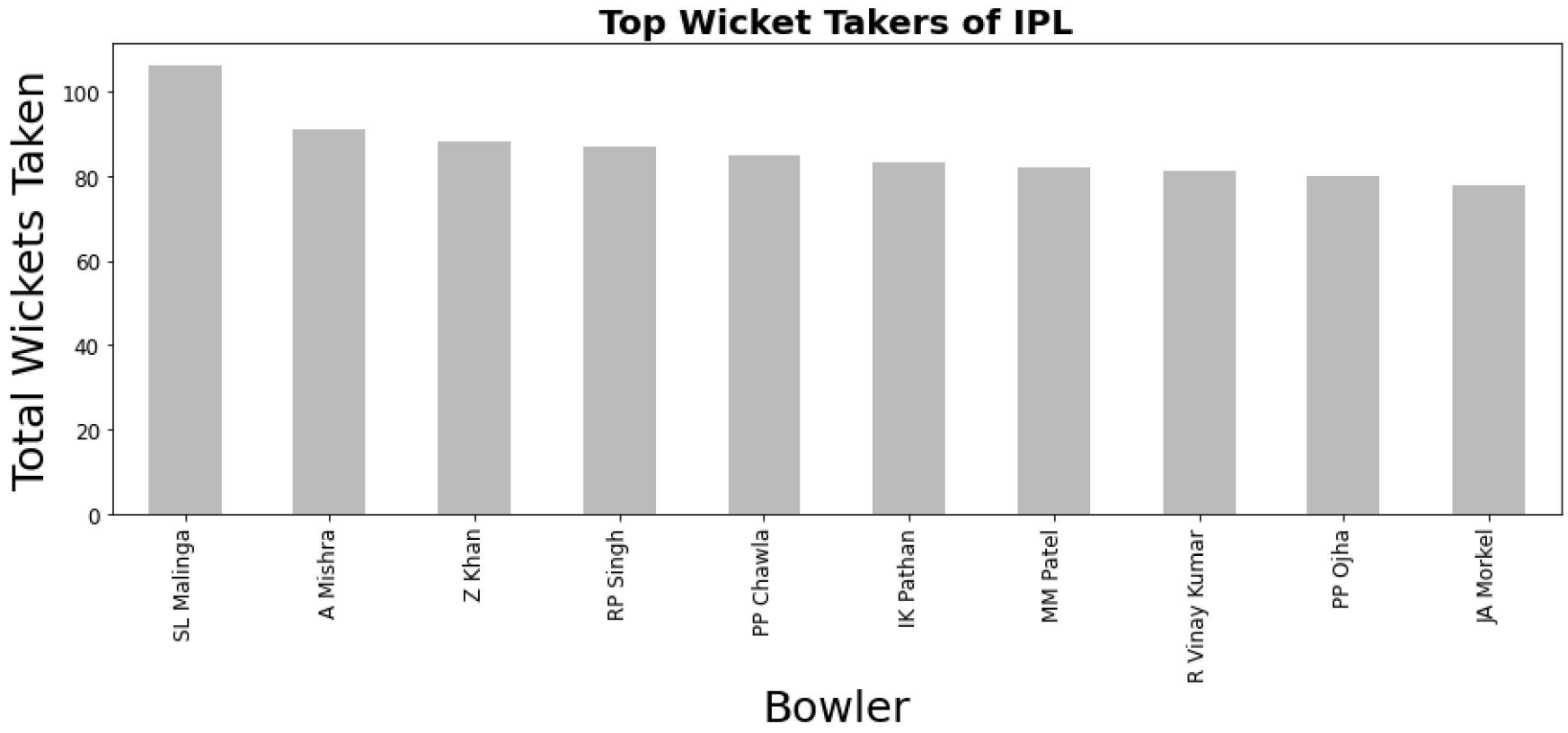


- Median score for Raina is above all the top 10 run getters. He has the highest lowest run among all the batsmen across 11 seasons. Considering the highest and lowest season totals and spread of runs, it seems Raina has been most consistent among all.

BEST BOWLERS

In [ ]:
```
merge.groupby('bowler')['player_dismissed'].count().sort_values(ascending = False).head(10).plot(kind = 'bar',
                                                color = 'skyblue', 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[ ]:
```
(array([  0.,  20.,  40.,  60.,  80., 100., 120.]),
 <a list of 7 Text major ticklabel objects>)
```



- Malinga has taken the most number of wickets in IPL followed by Amit Mishra and Z Khan.
- 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

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

- 2011-13 have more number of matches being played compared to other seasons.
- 10 teams played in 2011 and 9 teams each in 2012 and 2013 explaining comparatively high number of matches being played in these three seasons.
- Eden garden has hosted most number of matches.
- Mumbai Indians won most number of matches followed by Chennai Super Kings and then kolkata knight Riders.
- Most times teams who have won tosses have decided to chase down
- Most team chooses to field first on winning the toss except for Chennai Super Kings, Deccan Charges and Pune Warriors
- Teams winning tosses and choosing to field first have won most number of times.

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

- From the Data,it can be concluded Team Mumbai Indians have won most number of matches and therefore is preferable for endorsement of products by the companies. Other than MI, Chennai Super Kings and Kolkata Knight Riders have performed well.
- In Case of Individual performances, Chris Gayle with most number of MoM awards, SK Raina with most number of scored runs, SL Malinga taking most number of wickets and other consistent players like G Gambhir, RG Sharma and V Kohli should be endorsed by a company for its products.

THANKYOU