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

In [2]:
```
#import first dataset
df = pd.read_csv(r'C:\Users\aishw\Downloads\Indian Premier League\matches data 1.csv
```

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

Out[3]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_app
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 [4]:
```
df.shape
```

Out[4]:
```
(756, 18)
```

In [5]:
```
df.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 [6]:
```
df.describe()
```

Out[6]:

	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 [7]:
```
df.isnull().sum()
```

Out[7]:
```
id                   0
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
```

In [8]:
```
# import second data
df1 = pd.read_csv(r'C:\Users\aishw\Downloads\Indian Premier League\deliveries data.c
```

In [9]:
```
df1.head()
```

Out[9]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_supe
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	

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_supe
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 [10]:
```
df1.shape
```

Out[10]: (179078, 21)

In [11]:
```
df1.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 [12]:
```
df1.describe()
```

Out[12]:

	match_id	inning	over	ball	is_super_over	wide_runs
count	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
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.251161
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.000000

	match_id	inning	over	ball	is_super_over	wide_runs
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.000000
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.000000
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.000000

In [13]:
```
df1.isnull().sum()
```

Out[13]:
```
match_id                0
inning                  0
batting_team            0
bowling_team            0
over                    0
ball                    0
batsman                 0
non_striker             0
bowler                  0
is_super_over           0
wide_runs               0
bye_runs                0
legbye_runs             0
noball_runs             0
penalty_runs            0
batsman_runs            0
extra_runs              0
total_runs              0
player_dismissed    170244
dismissal_kind      170244
fielder             172630
dtype: int64
```

In [14]:
```
# merge two dataset.
merge_data = pd.merge(df1,df,left_on='match_id',right_on='id')
merge_data.head()
```

Out[14]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_supe
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 [15]:
```
merge_data.describe()
```

Out[15]:

	match_id	inning	over	ball	is_super_over	wide_runs
count	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
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.251161
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.000000
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.000000
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.000000
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.000000

In [16]:
```
merge_data.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 [17]:
```
merge_data.isnull().sum()
```

Out[17]:
```
match_id                0
inning                  0
batting_team            0
bowling_team            0
over                    0
ball                    0
batsman                 0
non_striker             0
bowler                  0
is_super_over           0
wide_runs               0
bye_runs                0
legbye_runs             0
noball_runs             0
penalty_runs            0
batsman_runs            0
extra_runs              0
total_runs              0
player_dismissed    170244
dismissal_kind      170244
fielder             172630
id                       0
season                   0
city                  1700
date                     0
team1                    0
team2                    0
toss_winner              0
toss_decision            0
result                   0
dl_applied               0
winner                 372
win_by_runs              0
win_by_wickets           0
player_of_match        372
venue                    0
umpire1                500
umpire2                500
umpire3             150712
dtype: int64
```

In [18]:
```
df.id.is_unique
```

Out[18]:
```
True
```

In [19]:
```
#since id is unique we have it as index
df.set_index('id', inplace=True)
```

In [20]:
```
df.describe()
```

Out[20]:

	season	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000	756.000000
mean	2013.444444	0.025132	13.283069	3.350529
std	3.366895	0.156630	23.471144	3.387963
min	2008.000000	0.000000	0.000000	0.000000

	season	dl_applied	win_by_runs	win_by_wickets
25%	2011.000000	0.000000	0.000000	0.000000
50%	2013.000000	0.000000	0.000000	4.000000
75%	2016.000000	0.000000	19.000000	6.000000
max	2019.000000	1.000000	146.000000	10.000000

In [21]:
```
df.describe(include='all')
```

Out[21]:

	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_ap
count	756.000000	749	756	756	756	756	756	756	756.0
unique	NaN	32	546	15	15	15	2	3	
top	NaN	Mumbai	2013-04-07	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	field	normal	
freq	NaN	101	2	101	95	98	463	743	
mean	2013.444444	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
std	3.366895	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.1
min	2008.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
25%	2011.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
50%	2013.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
75%	2016.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
max	2019.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0

In [22]:
```
#data preprocessing
df.head()
```

Out[22]:

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

id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applie
5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	

Conclusion 1. Umpire 1 and Umpire 2 having 1 each missing value. But for Umpire 3 having maximum no of missing value. 2. Team 1 and Team 2 having 14 Distinct Values. 3. Winners having 15 distinct values. 4. City having 33 distinct values. 5. Venue having 35 distinct values. 6. city is having maximum no missing value.

In [23]:
```
#Find that venue for which city value is missing
df[df.city.isnull()][['city' , 'venue']]
```

Out[23]:

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

Now we replace missing value with dubai

In [24]:
```
df.city = df.city.fillna('Dubai')
```

In [25]:
```
#Umpire1 and Umpire2 having 1 missing value
df[(df.umpire1.isnull())| (df.umpire2.isnull())]
```

Out[25]:

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

In [26]:
```
#umpire 3 is having lot of missing value so we drop it
df = df.drop('umpire3', axis=1)
```

In [27]:
```
df.head(10)
```

Out[27]:

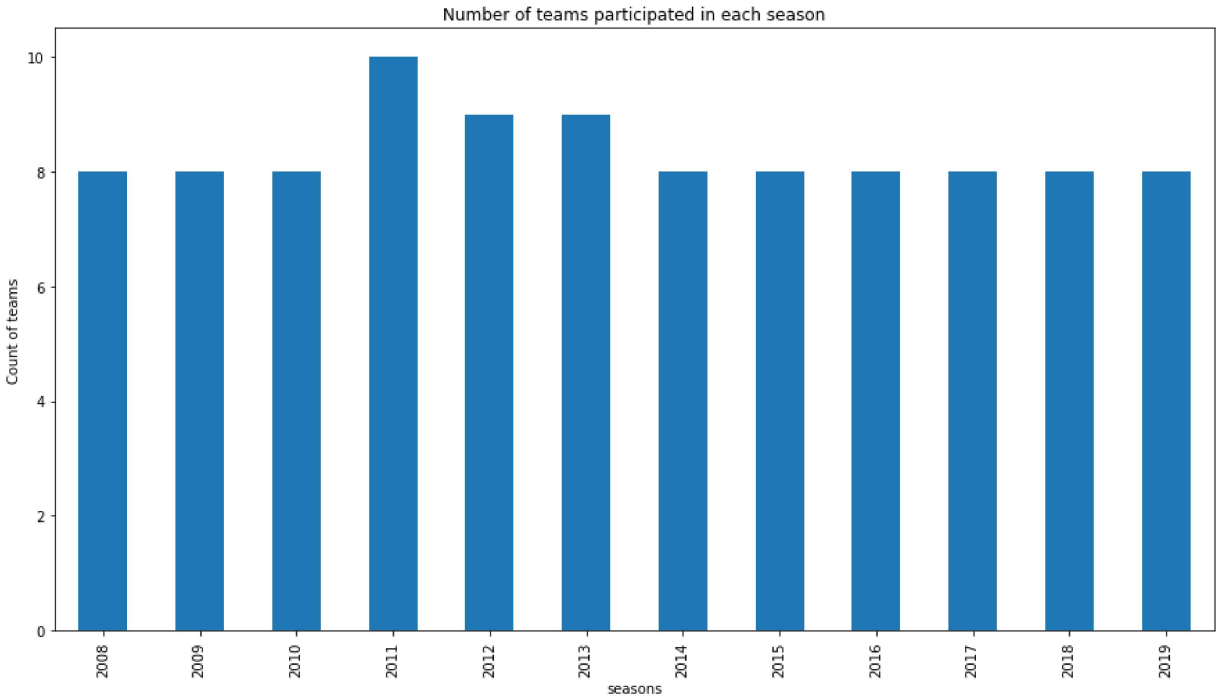
id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applie

id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applie
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	
6	2017	Hyderabad	2017-04-09	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	
7	2017	Mumbai	2017-04-09	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	
8	2017	Indore	2017-04-10	Royal Challengers Bangalore	Kings XI Punjab	Royal Challengers Bangalore	bat	normal	
9	2017	Pune	2017-04-11	Delhi Daredevils	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
10	2017	Mumbai	2017-04-12	Sunrisers Hyderabad	Mumbai Indians	Mumbai Indians	field	normal	

Perform Exploratory Data Analysis

How many teams played?

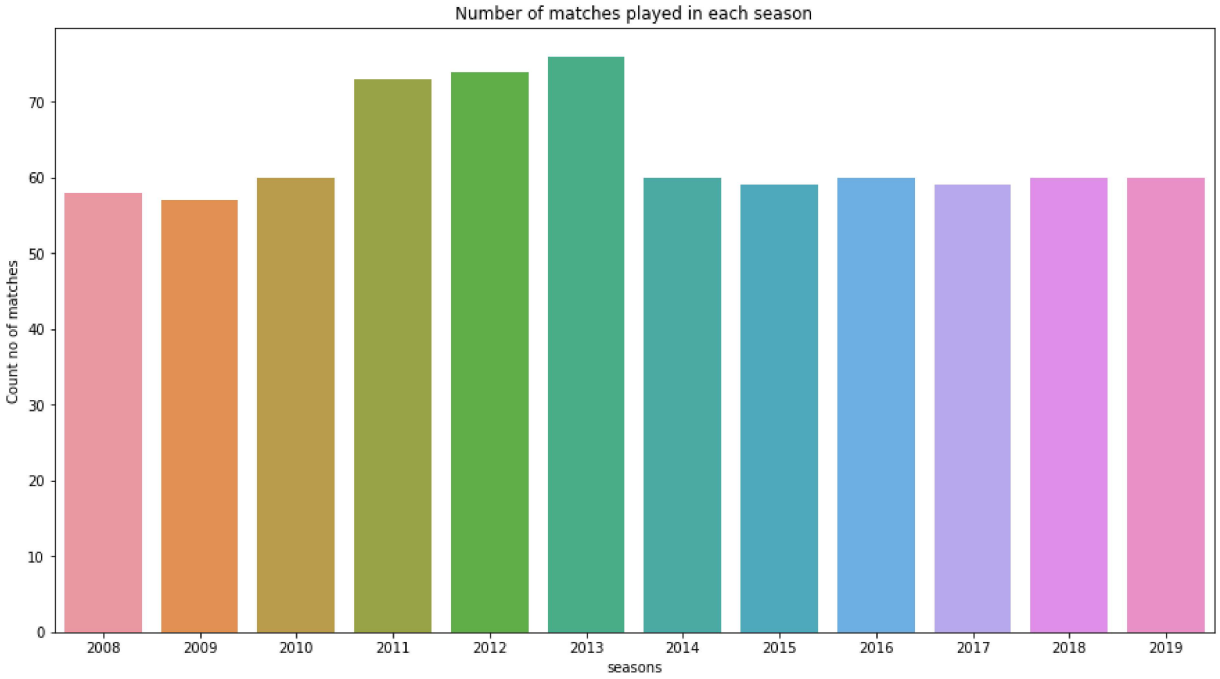
In [28]:
```
df.groupby('season')['team1'].nunique().plot(kind='bar', figsize=(15,8))
plt.title('Number of teams participated in each season')
plt.ylabel('Count of teams')
plt.xlabel('seasons')
plt.show()
```



Conclusion: 1. In year 2011 10 teams have played and In the year 2012 and 2013 9 teams have played. 2. Maximum no of matches are played in the year 2011 to 2013No of matches played in each season

In [29]:
```
plt.figure(figsize=(15,8))
sns.countplot('season', data = df)
plt.title('Number of matches played in each season')
plt.ylabel('Count no of matches')
plt.xlabel('seasons')
plt.show()
```

```
C:\ProgramData\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 p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

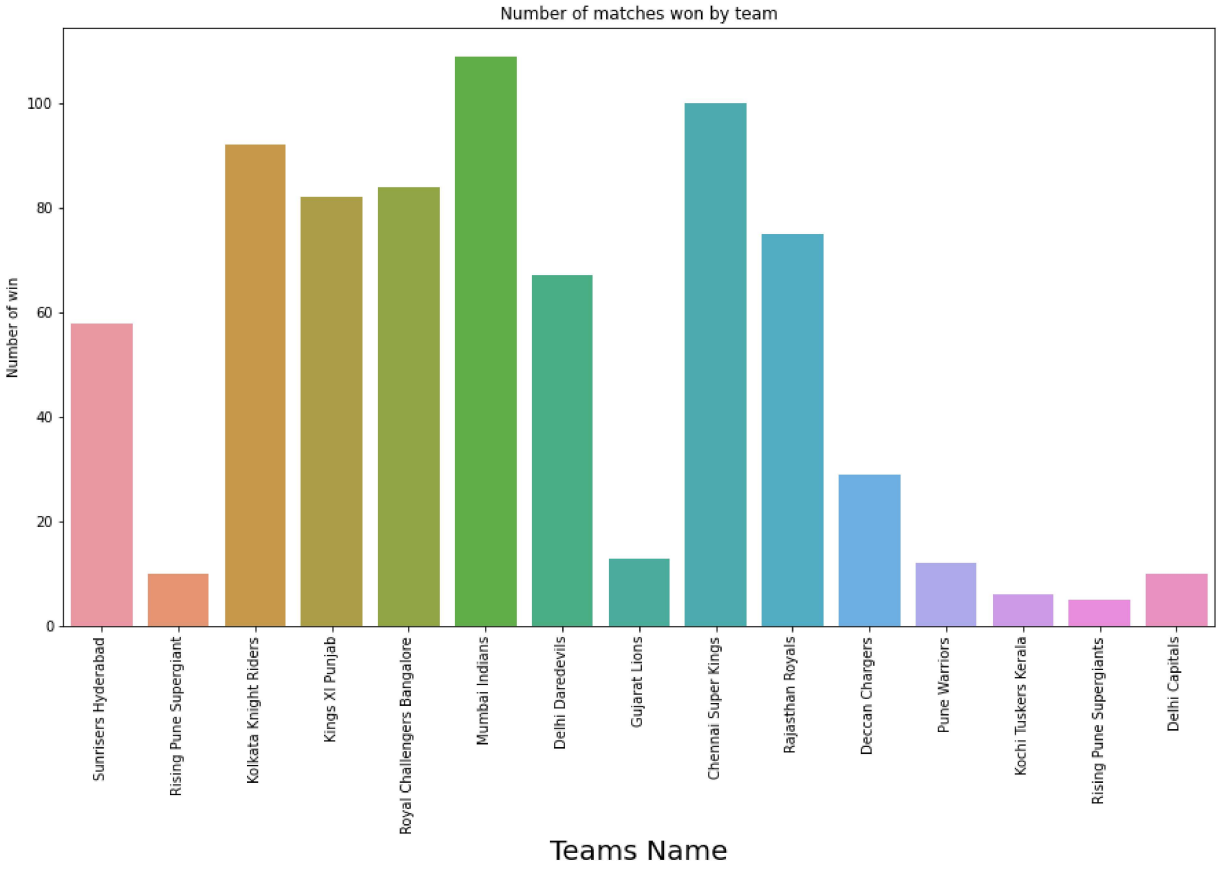


Conclusion: 1. In the year 2011,2012 and 2013 maximum above 70 matches have played. 2. Rest of all season have played matches on an average between 55 to 65.Number of matches won by team

In [30]:
```
plt.figure(figsize=(15,8))
sns.countplot('winner', data=df)
plt.title('Number of matches won by team')
```

```
plt.xlabel('Teams Name', fontsize=20)
plt.xticks(rotation=90, fontsize=10)
plt.ylabel('Number of win')
plt.yticks(fontsize=10)
plt.show()
```

```
C:\ProgramData\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 p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

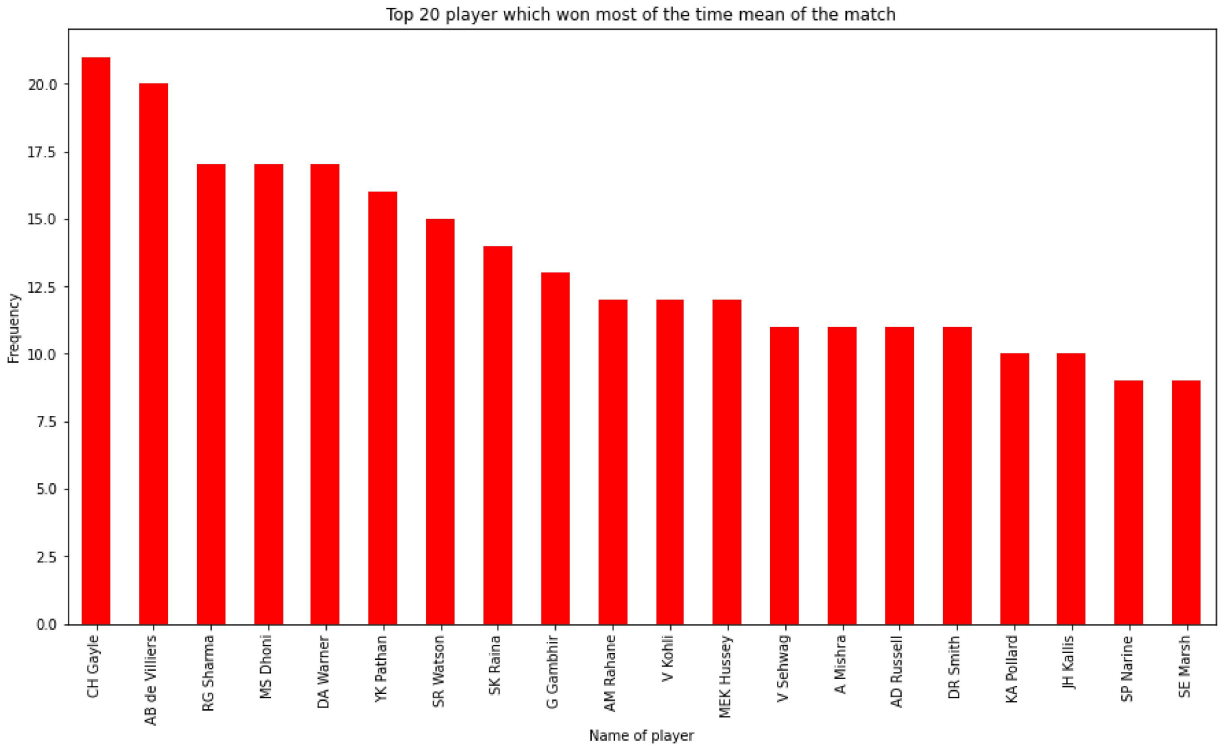


Conclusion: 1. Mumbai indians win the match maximum no of times. 2. Rising Pune Supergiants wins less no of matches.Players having 'Man of match Award'

In [31]:
```
df['player_of_match'].value_counts()
```

Out[31]:
```
CH Gayle          21
AB de Villiers    20
RG Sharma         17
MS Dhoni          17
DA Warner         17
                  ..
Mohammed Shami     1
PD Collingwood     1
M Ur Rahman        1
S Badrinath        1
M Kartik           1
Name: player_of_match, Length: 226, dtype: int64
```

In [32]:
```
Manofthematch = df['player_of_match'].value_counts()
Manofthematch.head(20).plot(kind='bar', figsize=(15,8), color='red')
plt.title('Top 20 player which won most of the time mean of the match')
plt.xlabel('Name of player')
plt.ylabel('Frequency')
plt.show()
```

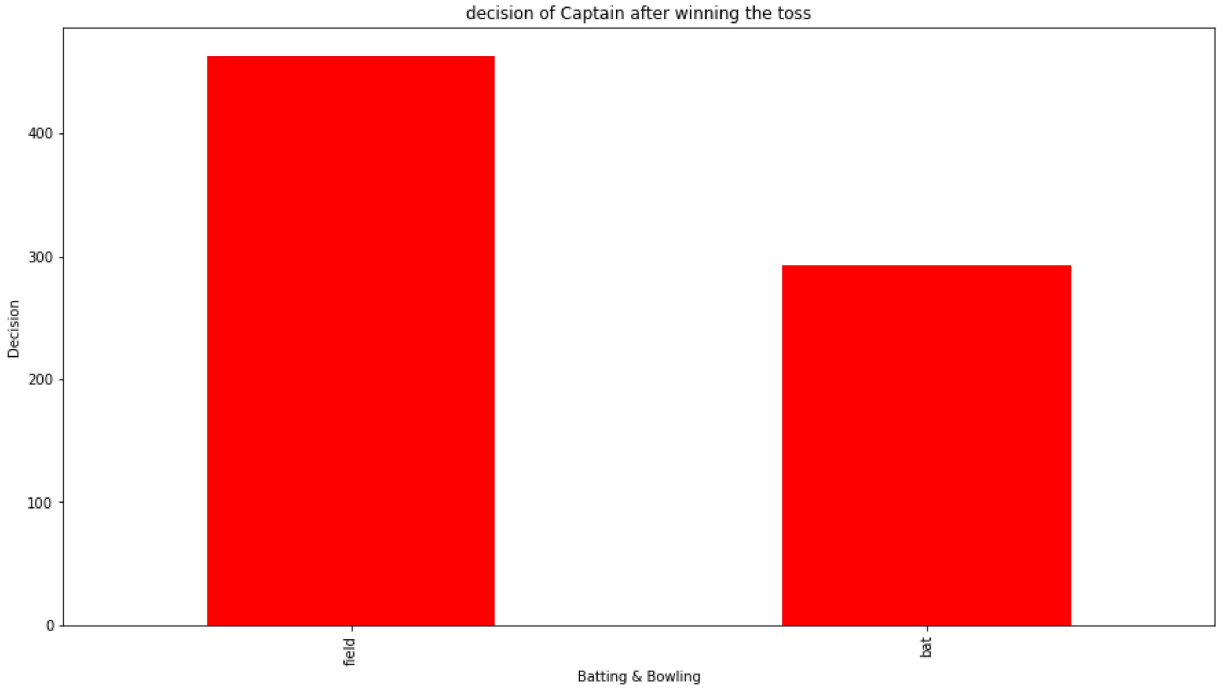


Conclusion: 1. CH Gayle won maximum no of time Man of the match award 2. AT Raydu and SE Marsh won minimum no of time Man of the match award. probability that deciding batting and bowling after winning the toss.

In [33]:
```
df['toss_decision'].value_counts()
```

Out[33]:
```
field    463
bat      293
Name: toss_decision, dtype: int64
```

In [34]:
```
toss_dec = df['toss_decision'].value_counts()
toss_dec.plot(kind='bar', figsize=(15,8), color='red')
plt.title('decision of Captain after winning the toss')
plt.xlabel('Batting & Bowling')
plt.ylabel('Decision')
plt.show()
```

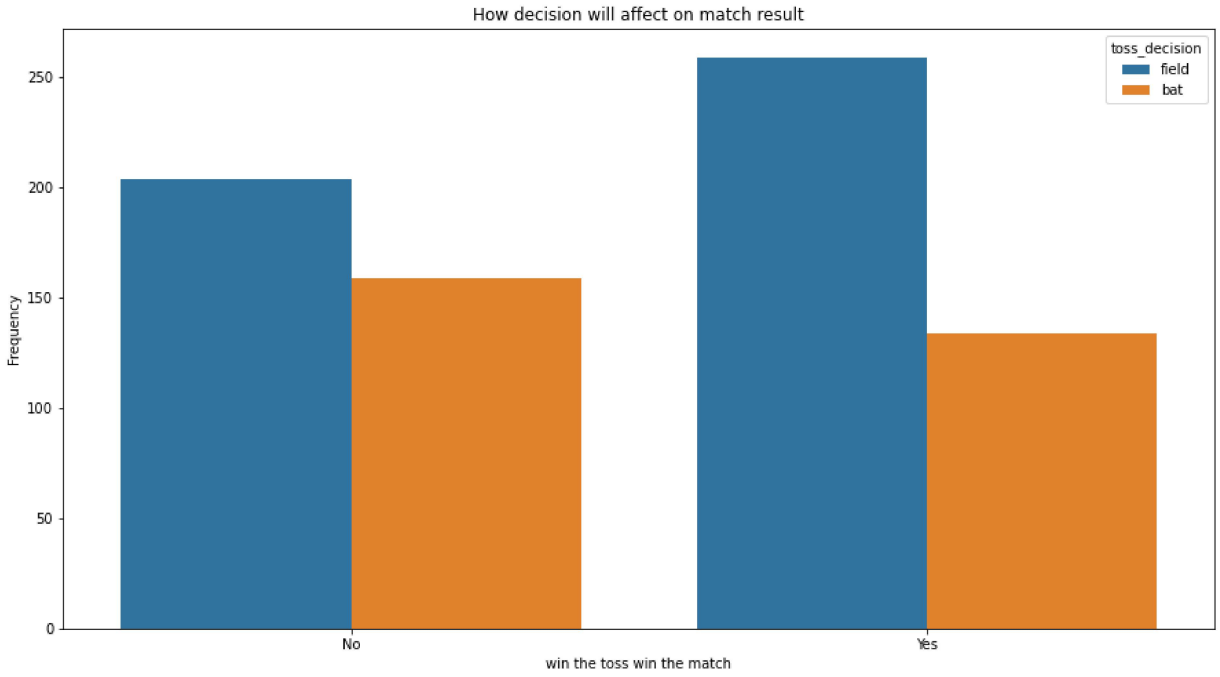


Conclusion: 1. Maximum no of time Captain decided to take fielding first.Number of time matches won by taking batting or bowling firstplt.figure(figsize=(15,8)) sns.countplot('season',hue='win_by_wickets',data=df)

plt.title('Number of time matches won by taking batting or bowling first') plt.xlabel('Number of season') plt.ylabel('Number of count') plt.show()Effect of toss decision on the match result

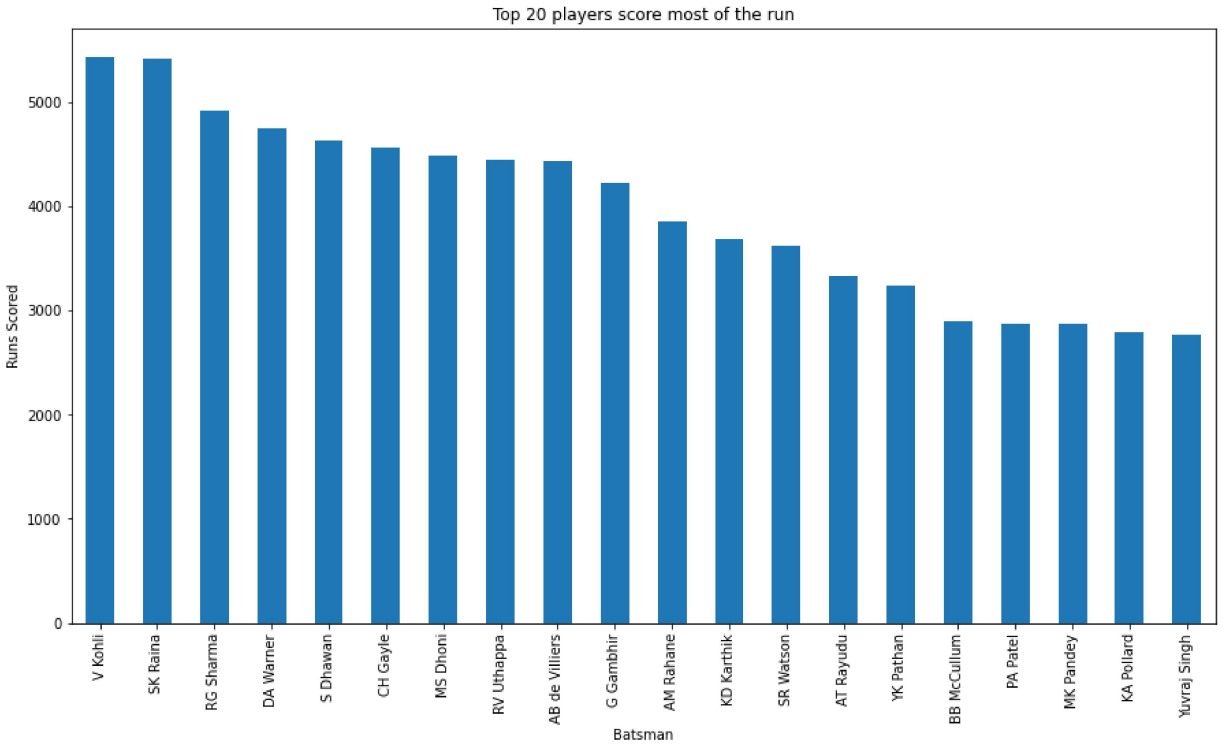
In [35]:
```
df['toss_win_game_win'] = np.where((df.toss_winner == df.winner),'Yes','No')
plt.figure(figsize=(15,8))
sns.countplot('toss_win_game_win',data=df,hue='toss_decision')
plt.title('How decision will affect on match result')
plt.xlabel('win the toss win the match')
plt.ylabel('Frequency')
plt.show()
```

```
C:\ProgramData\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 p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



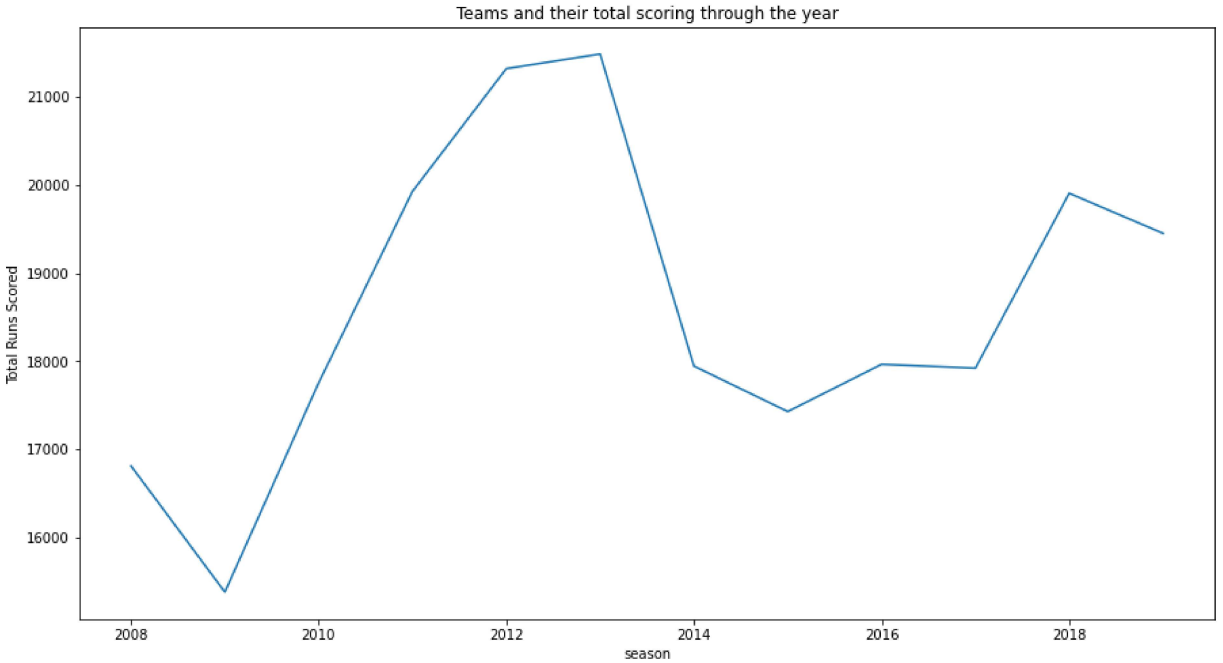
The captain should choose field first that probably will win the matchTop players who scored most of the run in IPL

In [36]:
```
merge_data.groupby('batsman')['batsman_runs'].sum().sort_values(ascending=False).hea
plt.title('Top 20 players score most of the run')
plt.xlabel('Batsman')
plt.ylabel('Runs Scored')
plt.show()
```



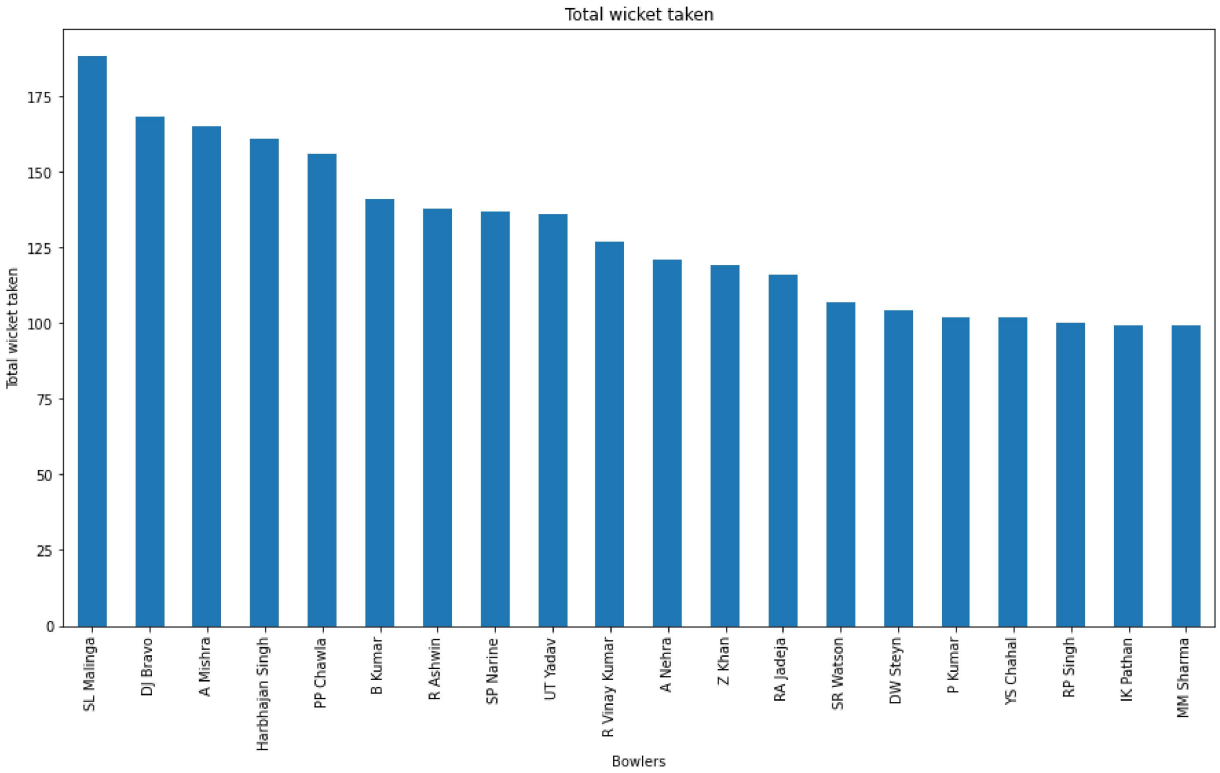
V.Kohli and SK Raina acored maximum run and Yuvraj Singh scored minimum runs.Teams and their total scoring through the season

In [37]:
```
merge_data.groupby('season')['batsman_runs'].sum().plot(kind='line',figsize=(15,8))
plt.title('Teams and their total scoring through the year')
plt.xlabel('season')
plt.ylabel('Total Runs Scored')
plt.show()
```



In 2012 and 2013 teams and their total score rate is maximum In 2009 teams and their total score rate is minimumplayers who have maximum wicket in IPL

In [38]:
```
merge_data.groupby('bowler')['player_dismissed'].count().sort_values(ascending=False
plt.title('Total wicket taken')
plt.xlabel('Bowlers')
plt.ylabel('Total wicket taken')
plt.show()
```

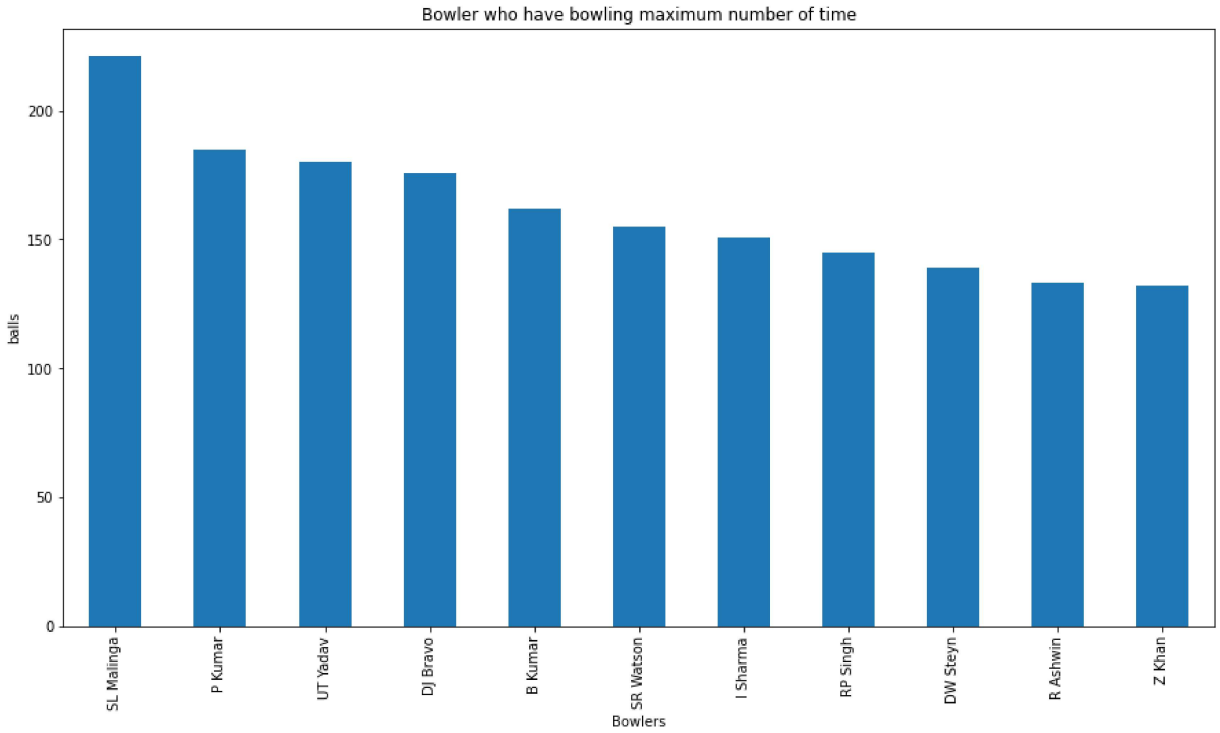


1.SLmalinga took highesh wicket in IPL 2.MM sharma and IK pathan very less wicket in IPLfind out extras bowlers

In [39]:
```
extras = df1[df1['extra_runs']!=0]['bowler'].value_counts()[:11]
print(extras)
```

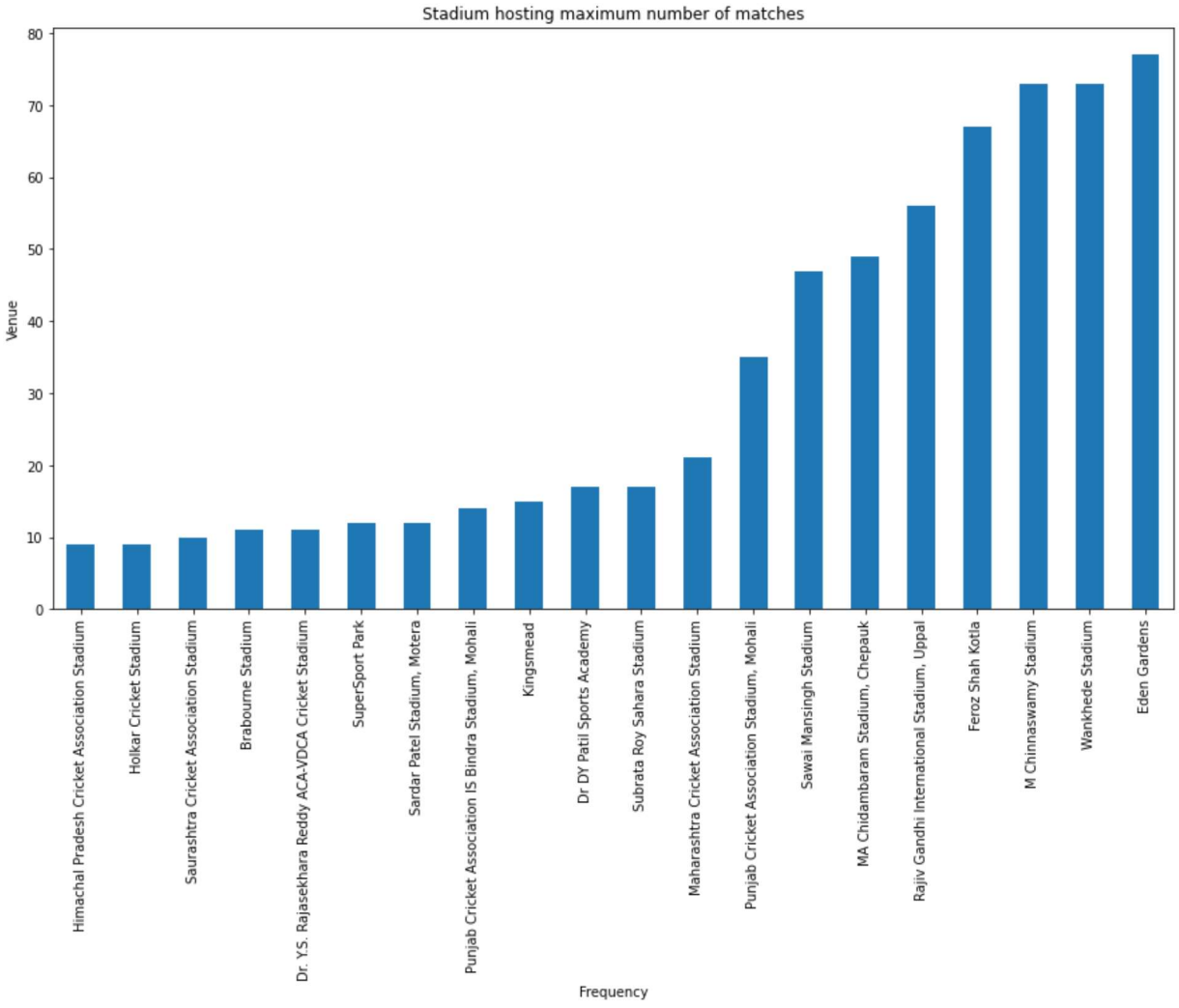
```
SL Malinga    221
P Kumar       185
UT Yadav      180
DJ Bravo      176
B Kumar       162
SR Watson     155
I Sharma      151
RP Singh      145
DW Steyn      139
R Ashwin      133
Z Khan        132
Name: bowler, dtype: int64
```

In [40]:
```
extras.plot(kind='bar',figsize=(15,8))
plt.title('Bowler who have bowling maximum number of time')
plt.xlabel('Bowlers')
plt.ylabel('balls')
plt.show()
```



1. SL Malinga who have done bowling maximum number of time. 2. Zkhan who have minimum number of bowling.Venue and Stadium that are hosted High matches of IPL

In [41]:
```
df.venue.value_counts().sort_values(ascending=True).tail(20).plot(kind='bar',figsize
plt.title('Stadium hosting maximum number of matches')
plt.xlabel('Frequency')
plt.ylabel('Venue')
plt.show()
```



1. Eden Garden hosted maximum number of matches 2. Himachal Pradesh Cricket Association Stadium and Holkar Cricket Stadium hosted minimum matchesConclusion: 1. In year 2011 10 teams have played and In the year 2012 and 2013 9 teams have played. 2. In the year 2011,2012 and 2013 maximum above 70 matches have played. 3. Mumbai indians win the match maximum no of times. 4. CH Gayle won maximum no of time Man of the match award. 5. Maximum no of time Captain decided to take fielding first. 6. The captain should choose field first that probably will win the match. 7. V.Kohli and SK Raina acored maximum run and Yuvraj Singh scored minimum runs. 8. In 2012 and 2013 teams and their total score rate is maximum. 9. SLmalinga took highesh wicket in IPL. 10. SL Malinga who have done bowling maximum number of time. 11. Eden Garden hosted maximum number of matches.Suggestions: 1. If companyis looking for a batsmanwith exceptional runs they should go for Chris Gayle, AB Devillers, Virat Kohli and R sharma 2.If company experienced bowlers they should go with Malinga, Pkumar and UT Jadhav

Thanku