# GRIP - The Sparks Foundation

## Task - 5 : Exploratory Data Analysis - Sports

### Perform Exploratory Data Analysis on 'Indian Premiere League

**Objective::**

● Perform ‘Exploratory Data Analysis’ on dataset ‘Indian Premier 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.

● Dataset:https://bit.ly/34SRn3b (https://bit.ly/34SRn3b)

In [1]:

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

In [2]:

```
matches_df = pd.read_csv("matches[1].csv")
score_df = pd.read_csv("deliveries[1].csv")
```

In [3]:

```
matches_df.head()
```

Out[3]:

| | 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 [4]:

```
matches_df.head()
```

Out[4]:

| | 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 [5]:

```
# DATA INFORMATION
print(matches_df.info())
print(score_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
None
<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
None
```

In [6]:

```
matches_df["umpire3"].isnull().sum()
```

Out[6]:

```
637
```

In [7]:

```
matches_df["umpire3"].tail(10)
```

Out[7]:

```
746              Nanda Kishore
747      KN Ananthapadmanabhan
748                Nitin Menon
749               Ulhas Gandhe
750             Bruce Oxenford
751                    S Ravi
752                 Ian Gould
753                       NaN
754    Chettithody Shamshuddin
755                Nigel Llong
Name: umpire3, dtype: object
```

In [8]:

```
matches_df["umpire3"].tail(10)
```

Out[8]:

```
746              Nanda Kishore
747      KN Ananthapadmanabhan
748                Nitin Menon
749               Ulhas Gandhe
750             Bruce Oxenford
751                    S Ravi
752                 Ian Gould
753                       NaN
754    Chettithody Shamshuddin
755                Nigel Llong
Name: umpire3, dtype: object
```

In [9]:

```
matches_df["umpire3"].tail(10)
```

Out[9]:

```
746              Nanda Kishore
747      KN Ananthapadmanabhan
748                Nitin Menon
749               Ulhas Gandhe
750             Bruce Oxenford
751                    S Ravi
752                 Ian Gould
753                       NaN
754    Chettithody Shamshuddin
755                Nigel Llong
Name: umpire3, dtype: object
```

In [10]:

```
# Seasons we have got in the dataset
matches_df['season'].unique()
```

Out[10]:

```
array([2017, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2018,
       2019], dtype=int64)
```

### Team won by Maximum Runs

In [11]:

```
matches_df.iloc[matches_df['win_by_runs'].idxmax()]
```

Out[11]:

```
id                              44
season                        2017
city                         Delhi
date                    2017-05-06
team1                Mumbai Indians
team2              Delhi Daredevils
toss_winner        Delhi Daredevils
toss_decision                field
result                      normal
dl_applied                       0
winner               Mumbai Indians
win_by_runs                    146
win_by_wickets                   0
player_of_match          LMP Simmons
venue              Feroz Shah Kotla
umpire1                 Nitin Menon
umpire2                   CK Nandan
umpire3                         NaN
Name: 43, dtype: object
```

In [12]:

```
matches_df.iloc[matches_df['win_by_runs'].idxmax()]['winner']
```

Out[12]:

```
'Mumbai Indians'
```

### Team won by Maximum Wickets

In [13]:

```
matches_df.iloc[matches_df['win_by_wickets'].idxmax()]['winner']
```

Out[13]:

```
'Kolkata Knight Riders'
```

### Team won by minimum runs

In [14]:

```
matches_df.iloc[matches_df[matches_df['win_by_runs'].ge(1)].win_by_runs.idxmin()]['winner']
```

Out[14]:

```
'Mumbai Indians'
```

### Team won by Minimum Wickets

In [15]:

```
matches_df.iloc[matches_df[matches_df['win_by_wickets'].ge(1)].win_by_wickets.idxmin()]
```

Out[15]:

```
id                                560
season                           2015
city                          Kolkata
date                       2015-05-09
team1                 Kings XI Punjab
team2             Kolkata Knight Riders
toss_winner           Kings XI Punjab
toss_decision                     bat
result                         normal
dl_applied                          0
winner            Kolkata Knight Riders
win_by_runs                         0
win_by_wickets                      1
player_of_match            AD Russell
venue                     Eden Gardens
umpire1                   AK Chaudhary
umpire2                HDPK Dharmasena
umpire3                            NaN
Name: 559, dtype: object
```

In [16]:

```
matches_df.iloc[matches_df[matches_df['win_by_wickets'].ge(1)].win_by_wickets.idxmin()]['wi
```

Out[16]:

```
'Kolkata Knight Riders'
```

## observation :

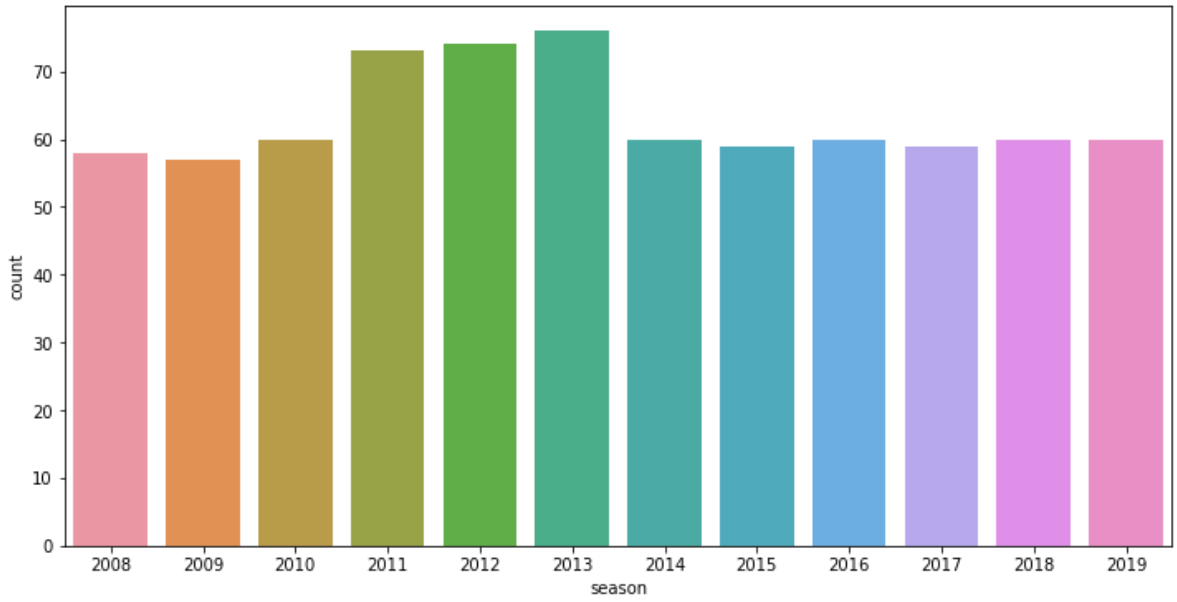
> Mumbai Indians is the team which won by maximum and minimum runs

> Kolkata Knight Riders is the team which won by maximum and minimum wickets

## Season Which had most number of matches

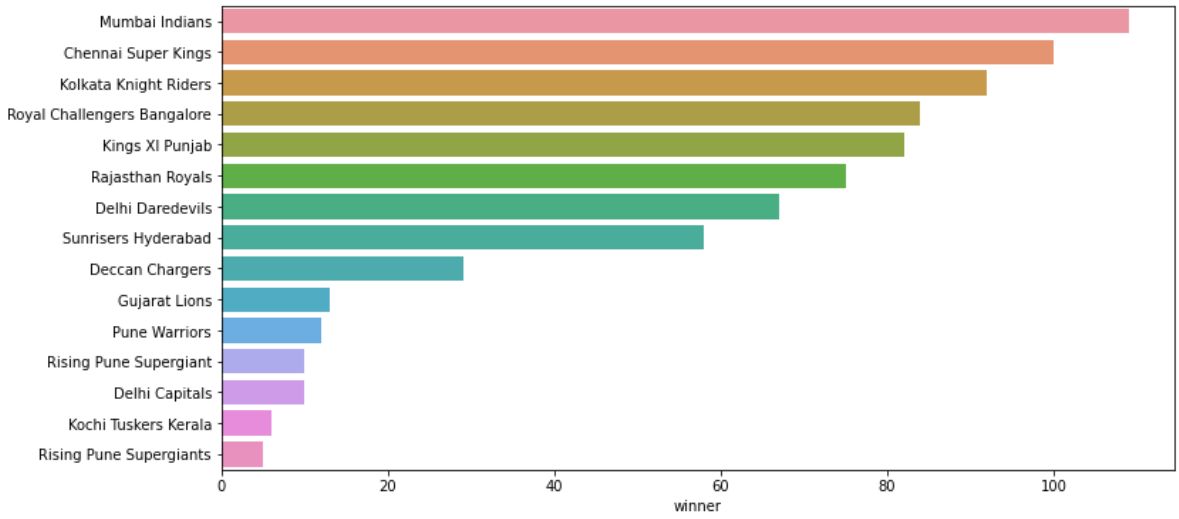
In [17]:

```
plt.figure(figsize=(12,6))
sns.countplot(x='season', data=matches_df)
plt.show()
```



In [18]:

```
plt.figure(figsize=(12,6))
data = matches_df.winner.value_counts()
sns.barplot(y = data.index, x = data, orient='h')
plt.show()
```

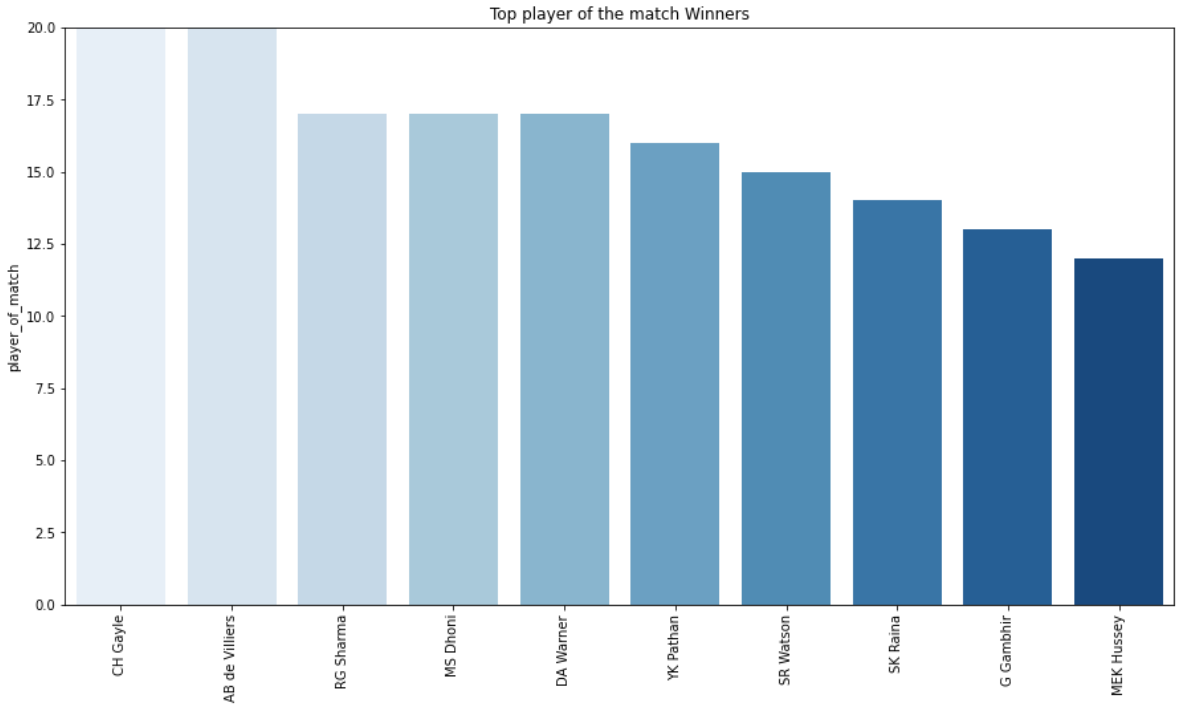



Mumbai Indians are the winners in most of the matches

## Top Player of the match winners

In [19]:

```
top_players = matches_df.player_of_match.value_counts()[:10]
#sns.barplot(x="day", y="total_bill", data=df)
fig, ax = plt.subplots(figsize=(15,8))
ax.set_ylim([0,20])
ax.set_ylabel("Count")
ax.set_title("Top player of the match Winners")
top_players.plot.bar()
sns.barplot(x = top_players.index, y = top_players, orient='v', palette="Blues");
plt.show()
```



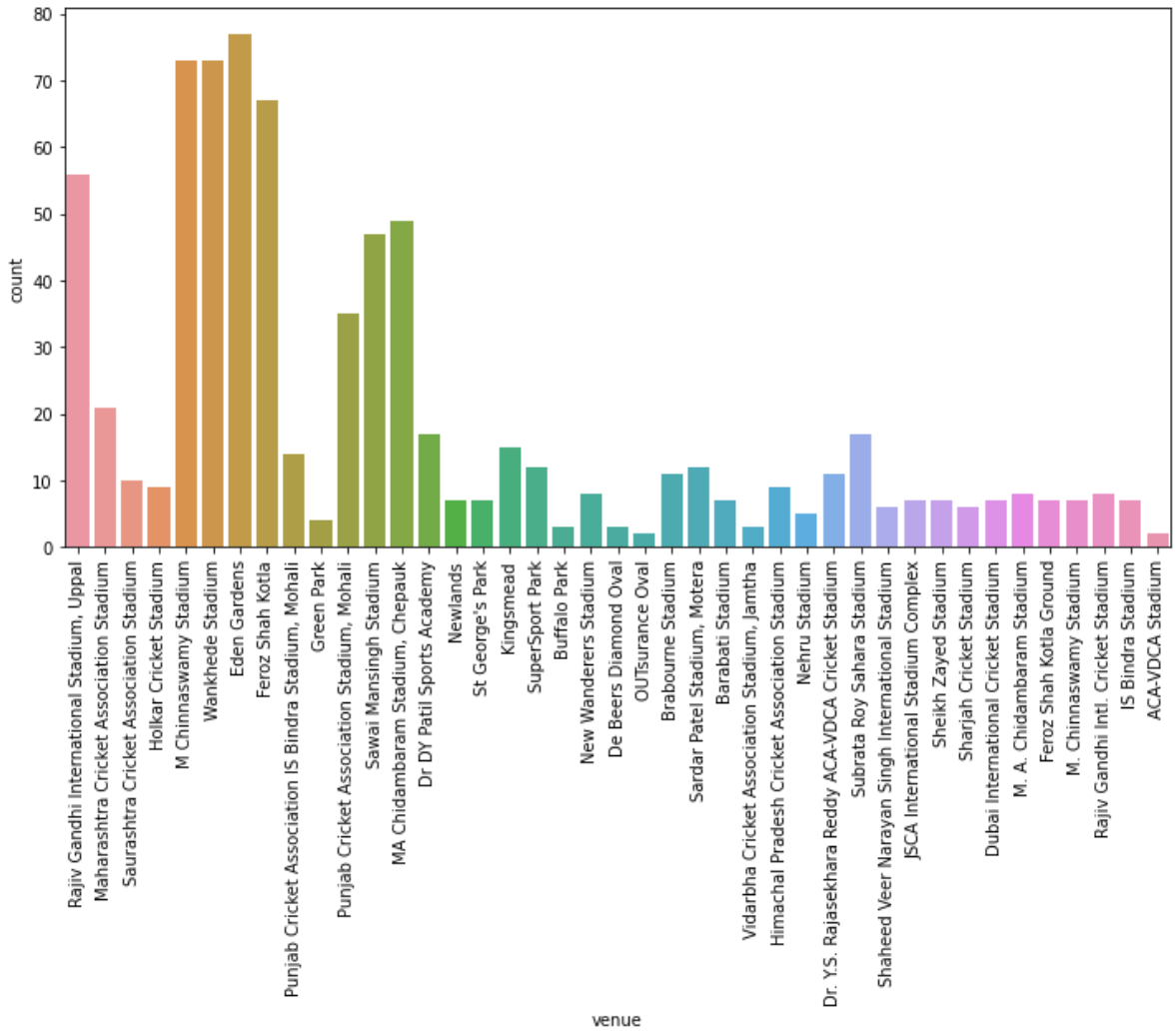

***CH Gayle***

is the most Successful player in all match winners

**Number of matches in each venue:**

In [20]:

```
plt.figure(figsize=(12,6))
sns.countplot(x='venue', data=matches_df)
plt.xticks(rotation='vertical')
plt.show()
```

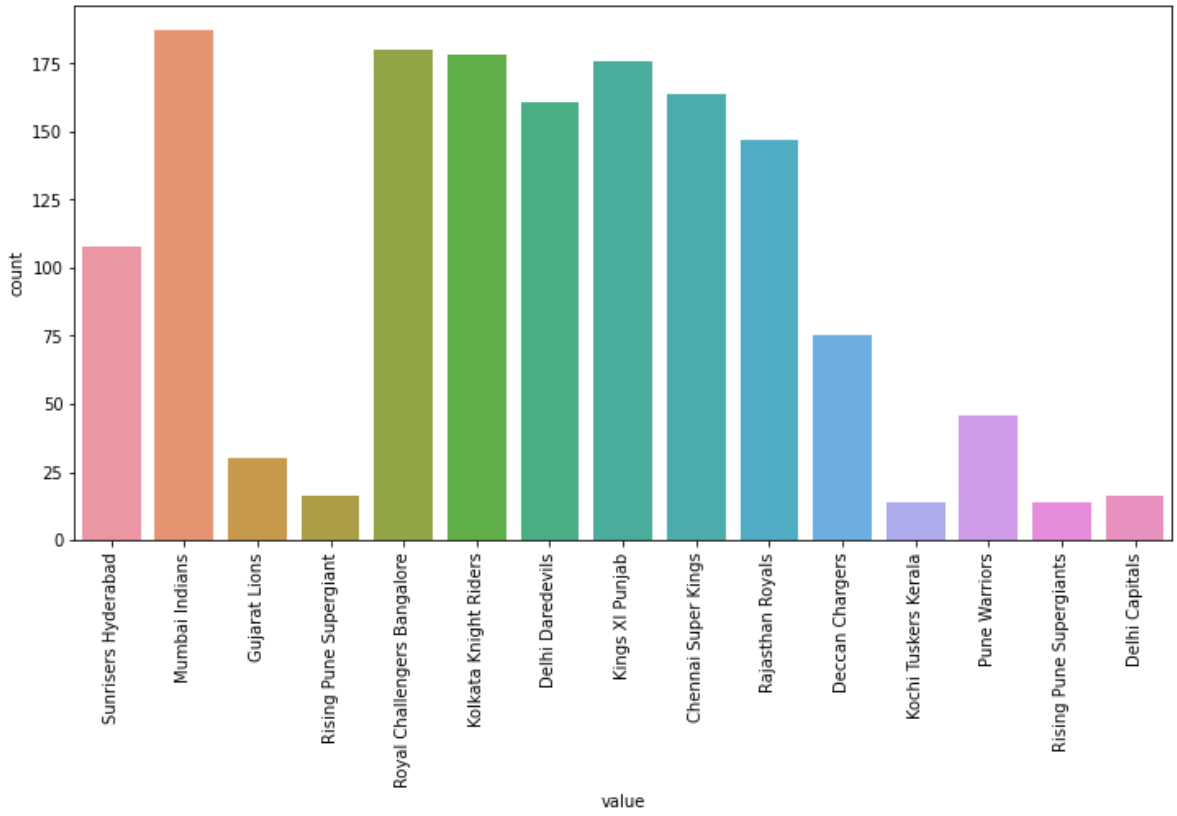



There are quite a few venues present in the data with "M Chinnaswamy Stadium" being the one with most number of matches followed by "Eden Gardens"

**Number of matches played by each team:**

In [21]:

```
temp_df = pd.melt(matches_df, id_vars=['id','season'], value_vars=['team1', 'team2'])

plt.figure(figsize=(12,6))
sns.countplot(x='value', data=temp_df)
plt.xticks(rotation='vertical')
plt.show()
```

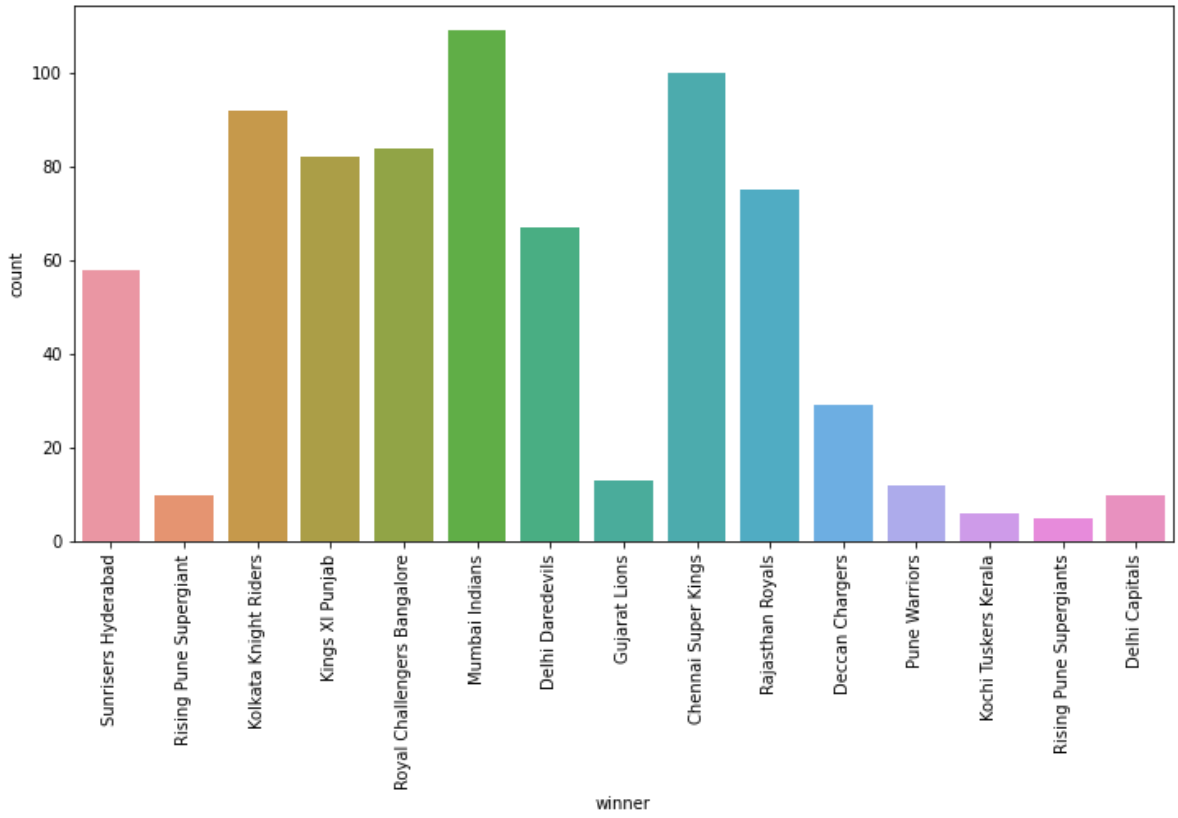



**"Mumbai Indians"** lead the pack with most number of matches played followed by **"Royal Challengers Bangalore"**. There are also teams with very few matches like 'Rising Pune Supergiants', 'Gujarat Lions' as they are new teams that came in only last season.

**Number of wins per team:**

In [22]:

```
plt.figure(figsize=(12,6))
sns.countplot(x='winner', data=matches_df)
plt.xticks(rotation=90)
plt.show()
```




**MI** again leads the pack followed by CSK.

## Champions each season:

Now let us see the champions in each season

In [23]:

```
temp_df = matches_df.drop_duplicates(subset=['season'], keep='last')[['season', 'winner']].
temp_df
```

Out[23]:

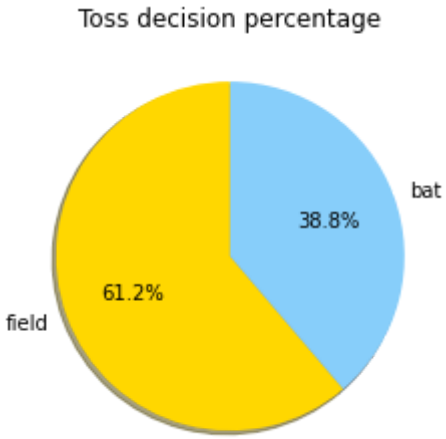
| | season | winner |
|---|---|---|
| **0** | 2017 | Mumbai Indians |
| **1** | 2008 | Rajasthan Royals |
| **2** | 2009 | Deccan Chargers |
| **3** | 2010 | Chennai Super Kings |
| **4** | 2011 | Chennai Super Kings |
| **5** | 2012 | Kolkata Knight Riders |
| **6** | 2013 | Mumbai Indians |
| **7** | 2014 | Kolkata Knight Riders |
| **8** | 2015 | Mumbai Indians |
| **9** | 2016 | Sunrisers Hyderabad |
| **10** | 2018 | Chennai Super Kings |
| **11** | 2019 | Mumbai Indians |

> Toss decision:

Let us see the toss decisions taken so far.

In [24]:

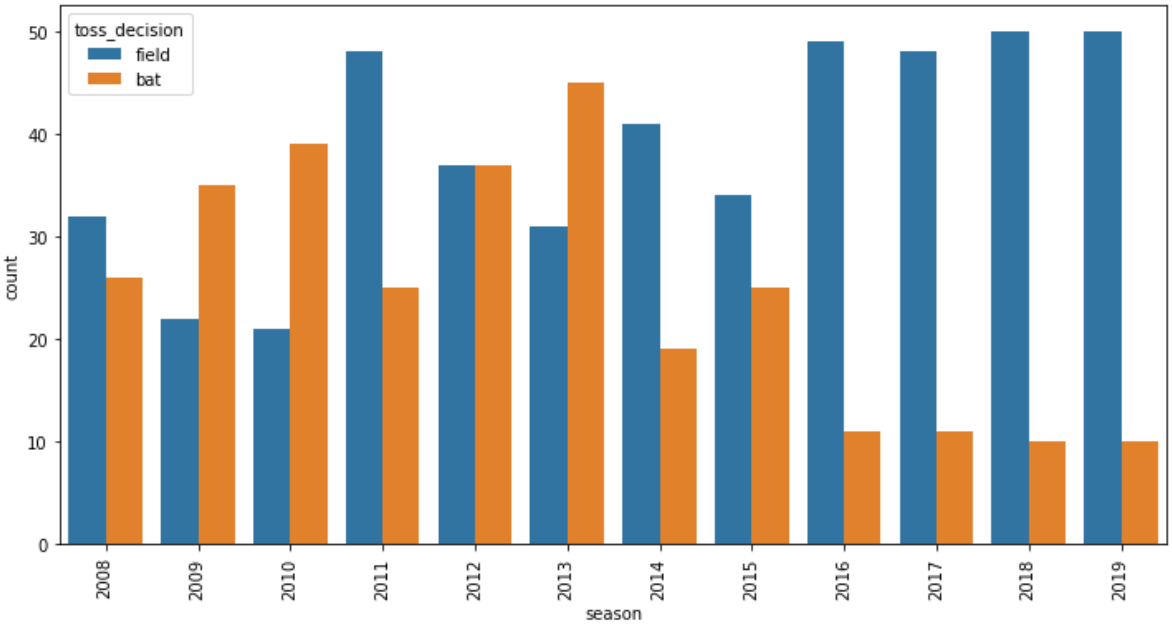
```
temp_series = matches_df.toss_decision.value_counts()
labels = (np.array(temp_series.index))
sizes = (np.array((temp_series / temp_series.sum())*100))
colors = ['gold', 'lightskyblue']
plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)
plt.title("Toss decision percentage")
plt.show()
```




Almost 55% of the toss decisions are made to field first. Now let us see how this decision varied over time.

In [25]:

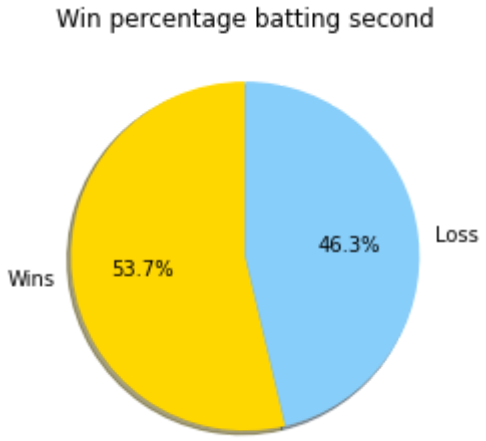
```
plt.figure(figsize=(12,6))
sns.countplot(x='season', hue='toss_decision', data=matches_df)
plt.xticks(rotation='vertical')
plt.show()
```



It seems during the initial years, teams wanted to bat first. Voila.! Look at the 2016 season, most of the toss decisions are to field first.

In [26]:

```
# Since there is a very strong trend towards batting second let us see the win percentage o
num_of_wins = (matches_df.win_by_wickets>0).sum()
num_of_loss = (matches_df.win_by_wickets==0).sum()
labels = ["Wins", "Loss"]
total = float(num_of_wins + num_of_loss)
sizes = [(num_of_wins/total)*100, (num_of_loss/total)*100]
colors = ['gold', 'lightskyblue']
plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)
plt.title("Win percentage batting second")
plt.show()
```




So percentage of times teams batting second has won is 53.2. Now let us split this by year and see the distribution.

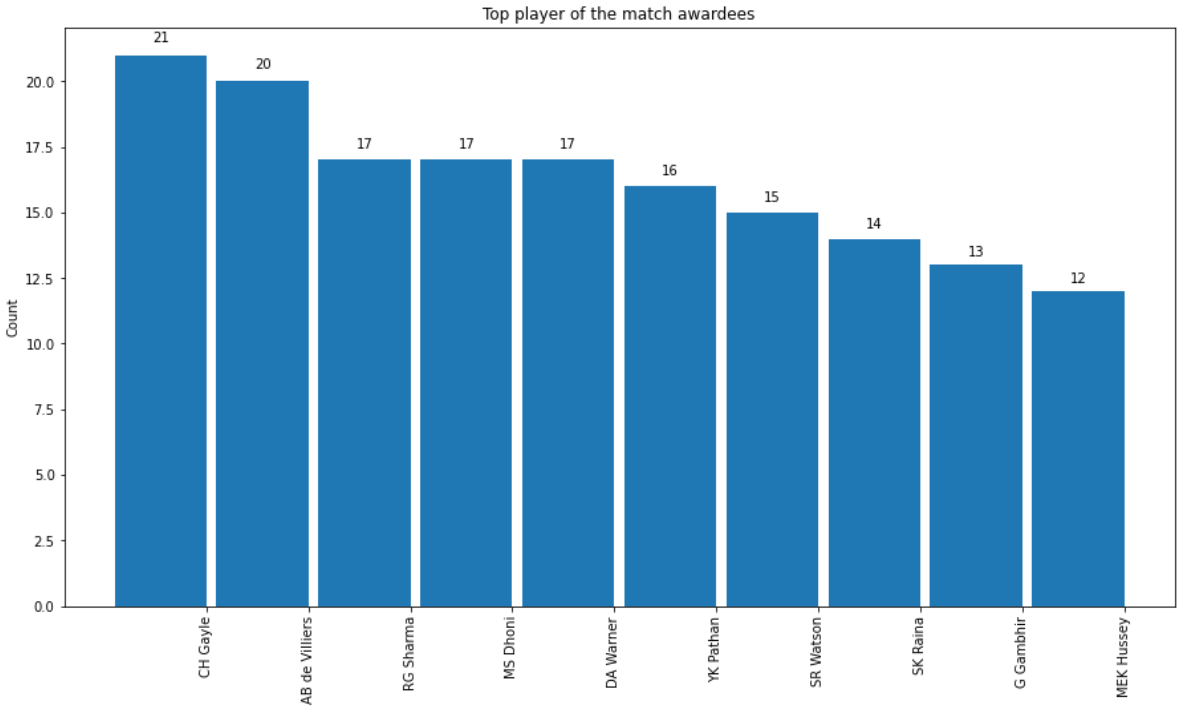
**Top players of the match:**

In [27]:

```
# create a function for labeling #
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.02*height,
                '%d' % int(height),
                ha='center', va='bottom')
```

In [28]:

```
temp_series = matches_df.player_of_match.value_counts()[:10]
labels = np.array(temp_series.index)
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_series), width=width)
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Top player of the match awardees")
autolabel(rects)
plt.show()
```

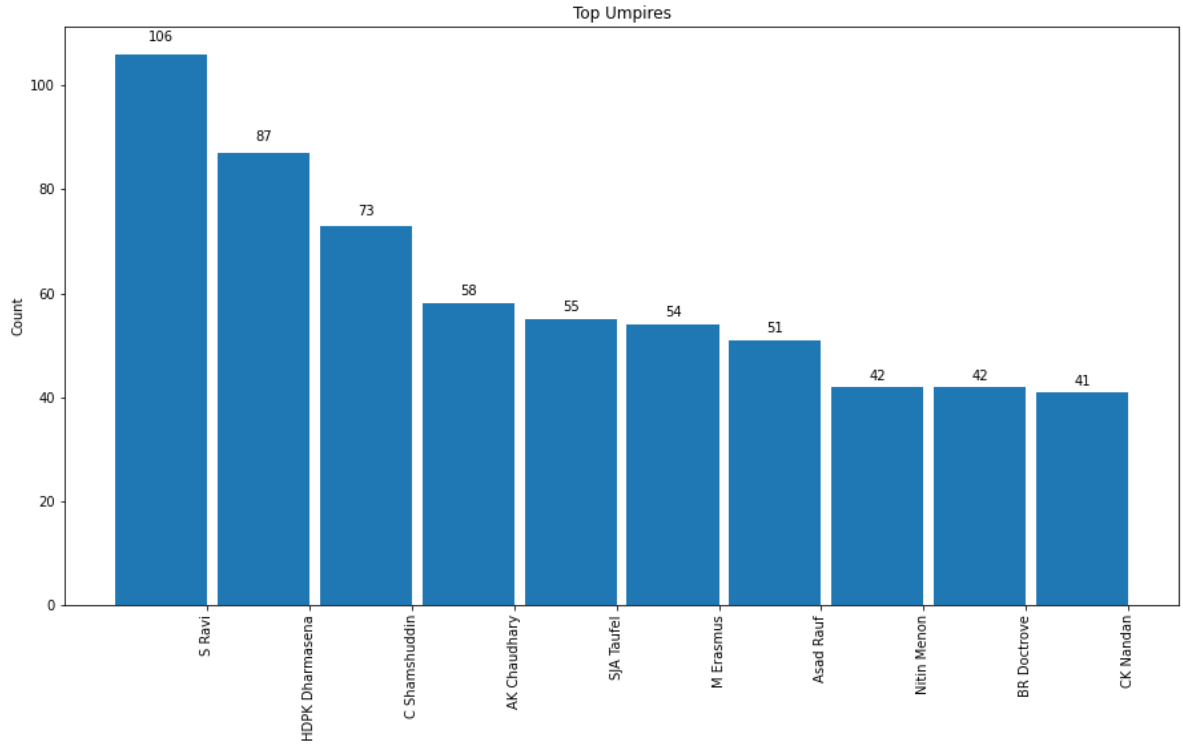



**CH Gayle** is the top player of the match awardee in all the seasons of IPL.

**Top Umpires:**

In [29]:

```
temp_df = pd.melt(matches_df, id_vars=['id'], value_vars=['umpire1', 'umpire2'])

temp_series = temp_df.value.value_counts()[:10]
labels = np.array(temp_series.index)
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_series), width=width,)
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Top Umpires")
autolabel(rects)
plt.show()
```




**Dharmasena** seems to be the most sought after umpire for IPL matches followed by Ravi. Others are fairly close to each other.

# Score Data Set

In [30]:

```
score_df.head()
```

Out[30]:

| | match_id | inning | batting_team | bowling_team | over | ball | batsman | non_striker | bowler | is_ |
|---|---|---|---|---|---|---|---|---|---|---|
| **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 | |

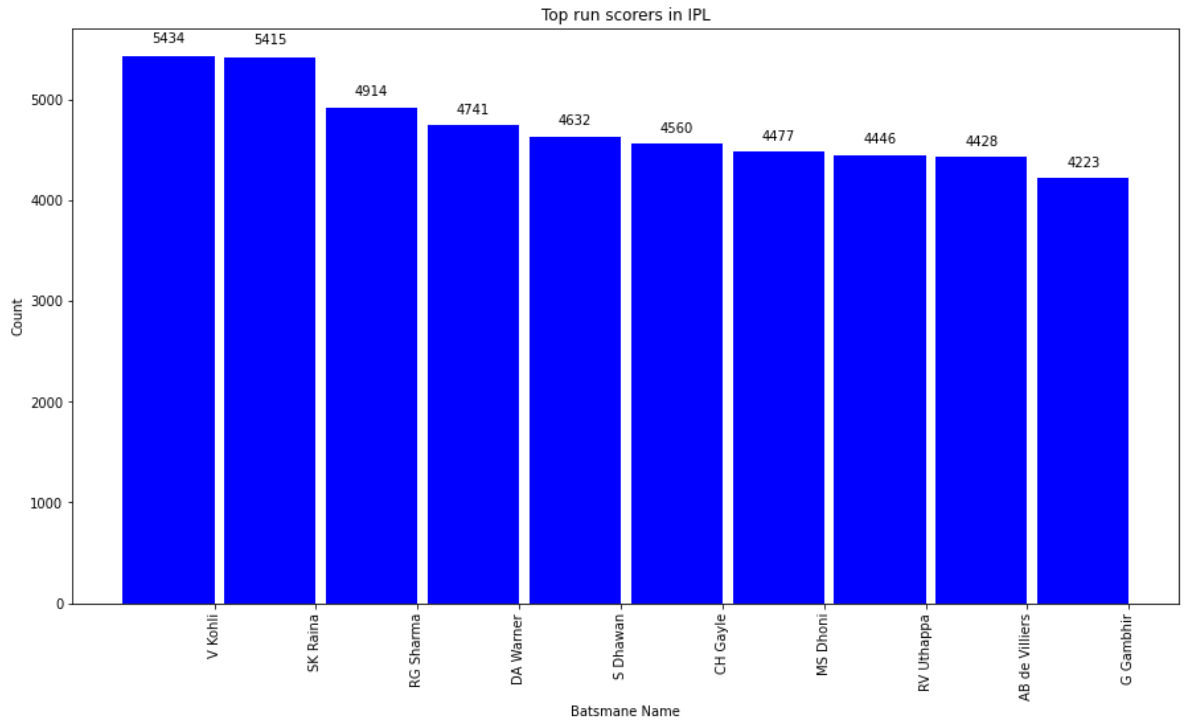
5 rows × 21 columns

**Batsman analysis:**

Let us start our analysis with batsman. Let us first see the ones with most number of IPL runs under their belt.

In [31]:

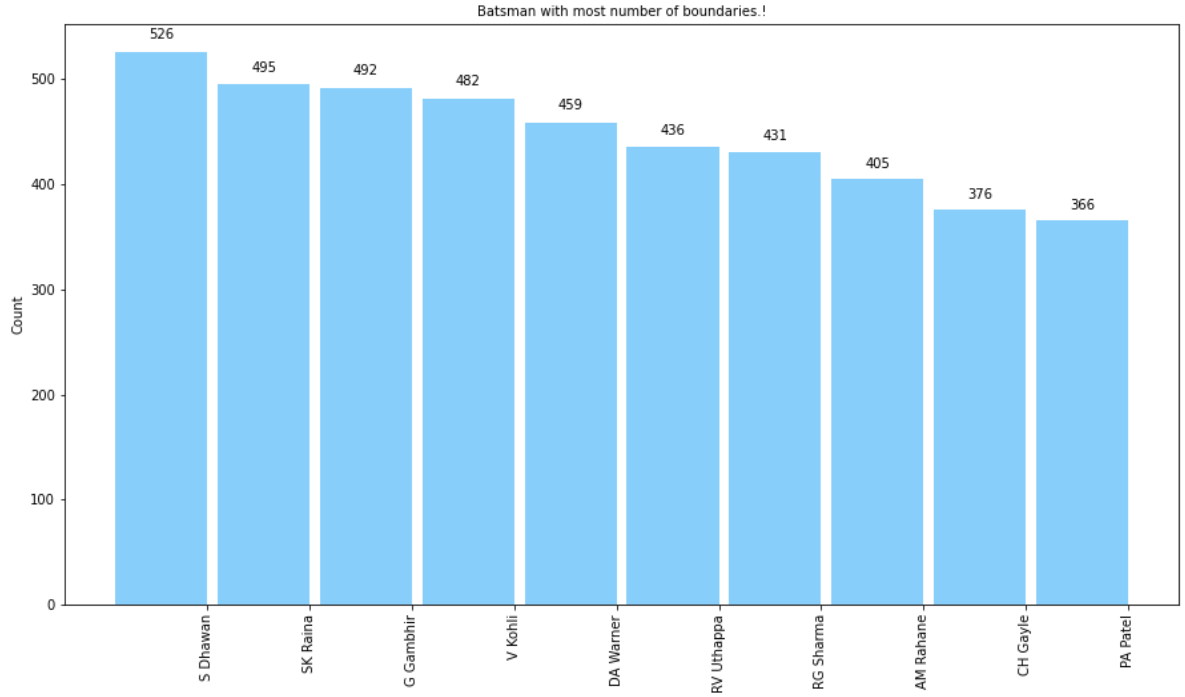
```
temp_df = score_df.groupby('batsman')['batsman_runs'].agg('sum').reset_index().sort_values(
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['batsman'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['batsman_runs']), width=width, color='blue')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Top run scorers in IPL")
ax.set_xlabel('Batsmane Name')
autolabel(rects)
plt.show()
```




**Virat Kohli** is leading the chart followed closely by Raina. Gayle is the top scorer among foreign players.

In [32]:

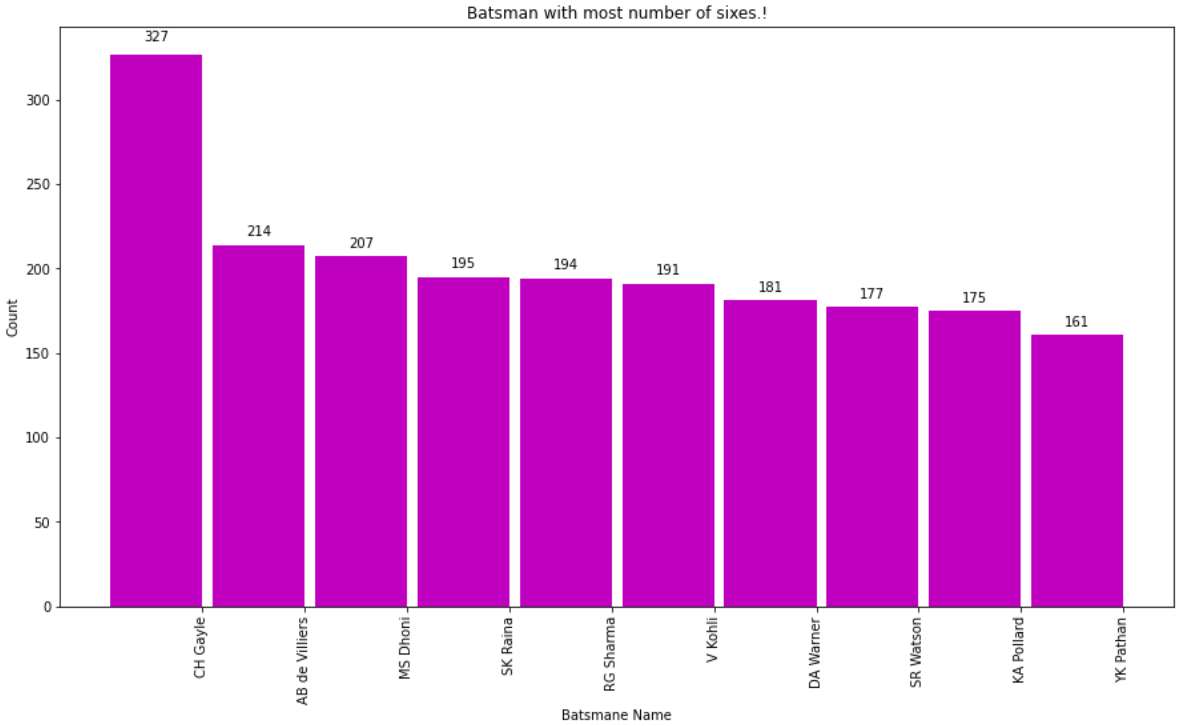
```
# Now let us see the players with more number of boundaries in IPL.
temp_df = score_df.groupby('batsman')['batsman_runs'].agg(lambda x: (x==4).sum()).reset_ind
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['batsman'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['batsman_runs']), width=width, color='lightskyblue')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Batsman with most number of boundaries.!",fontsize = 10)
autolabel(rects)
plt.show()
```




**Gambhir** is way ahead of others - almost 60 boundaries more than Kohli.! Nice to Sachin in the top 10 list :)

In [33]:

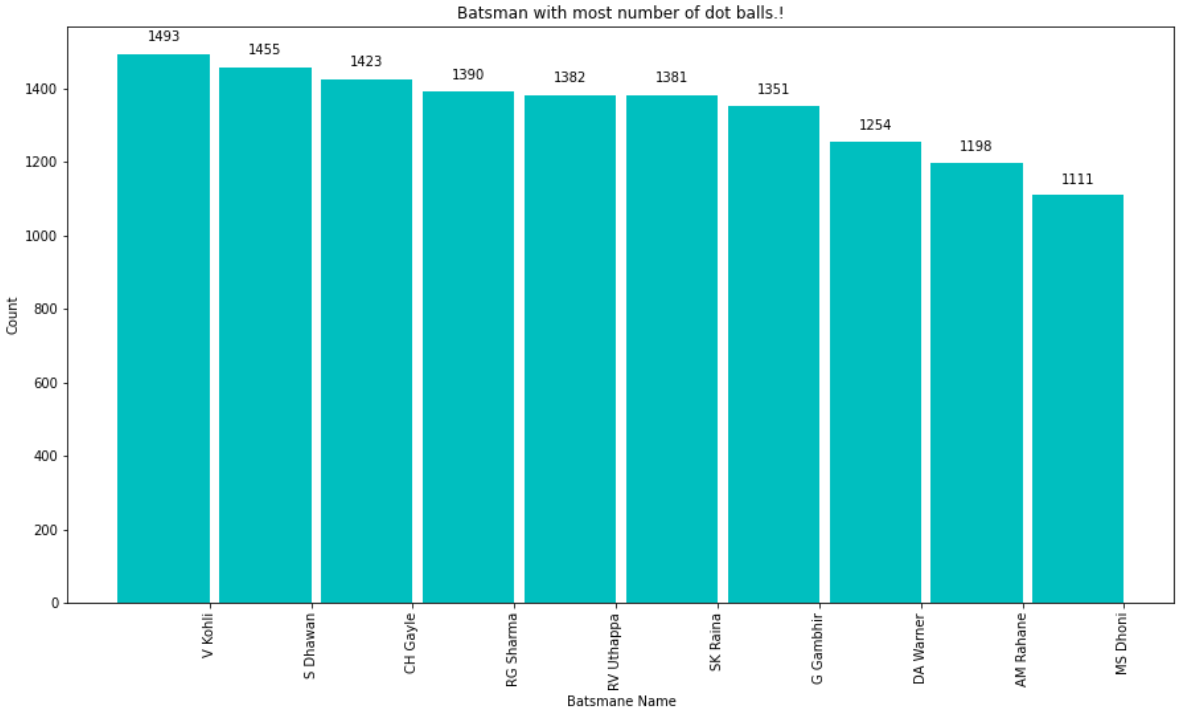
```
# Now let us check the number of 6's
temp_df = score_df.groupby('batsman')['batsman_runs'].agg(lambda x: (x==6).sum()).reset_ind
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['batsman'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['batsman_runs']), width=width, color='m')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation=90)
ax.set_ylabel("Count")
ax.set_title("Batsman with most number of sixes.!")
ax.set_xlabel('Batsmane Name')
autolabel(rects)
plt.show()
```




**Raina** is third in both number of 4's and 6's

In [34]:

```
# Now let us see the batsman who has played the most number of dot balls.
temp_df = score_df.groupby('batsman')['batsman_runs'].agg(lambda x: (x==0).sum()).reset_ind
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['batsman'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['batsman_runs']), width=width, color='c')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Batsman with most number of dot balls.!")
ax.set_xlabel('Batsmane Name')
autolabel(rects)
plt.show()
```

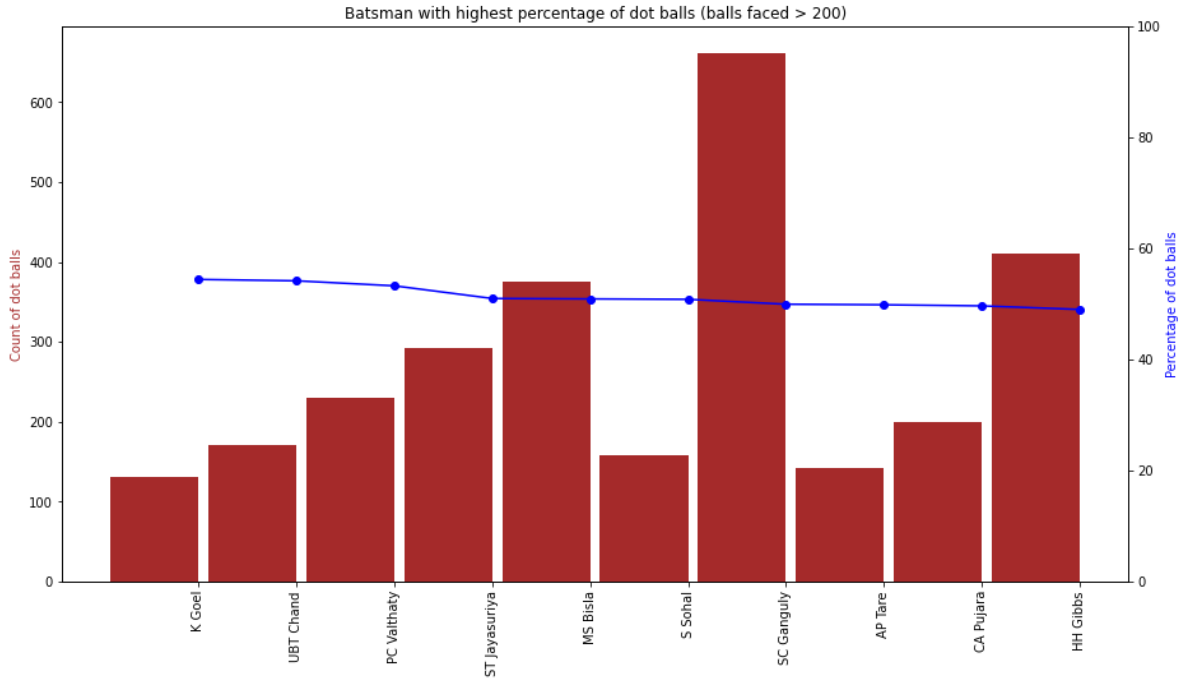



It is interesting to see that the same names repeat again here as well. I think since these guys have played more number of balls, they have more dot balls as well.

In [35]:

```
# Let us check the percentage distribution now.
def balls_faced(x):
    return len(x)

def dot_balls(x):
    return (x==0).sum()

temp_df = score_df.groupby('batsman')['batsman_runs'].agg([balls_faced, dot_balls]).reset_i
temp_df = temp_df.loc[temp_df.balls_faced>200,:]
temp_df['percentage_of_dot_balls'] = (temp_df['dot_balls'] / temp_df['balls_faced'])*100.
temp_df = temp_df.sort_values(by='percentage_of_dot_balls', ascending=False).reset_index(dr
temp_df = temp_df.iloc[:10,:]

fig, ax1 = plt.subplots(figsize=(15,8))
ax2 = ax1.twinx()
labels = np.array(temp_df['batsman'])
ind = np.arange(len(labels))
width = 0.9
rects = ax1.bar(ind, np.array(temp_df['dot_balls']), width=width, color='brown')
ax1.set_xticks(ind+((width)/2.))
ax1.set_xticklabels(labels, rotation='vertical')
ax1.set_ylabel("Count of dot balls", color='brown')
ax1.set_title("Batsman with highest percentage of dot balls (balls faced > 200)")
ax2.plot(ind+0.45, np.array(temp_df['percentage_of_dot_balls']), color='b', marker='o')
ax2.set_ylabel("Percentage of dot balls", color='b')
ax2.set_ylim([0,100])
ax2.grid(b=False)
plt.show()
```

```
C:\Users\vanda\AppData\Local\Temp/ipykernel_26008/1546379463.py:27: Matplotl
ibDeprecationWarning: The 'b' parameter of grid() has been renamed 'visible'
since Matplotlib 3.5; support for the old name will be dropped two minor rel
eases later.
  ax2.grid(b=False)
```



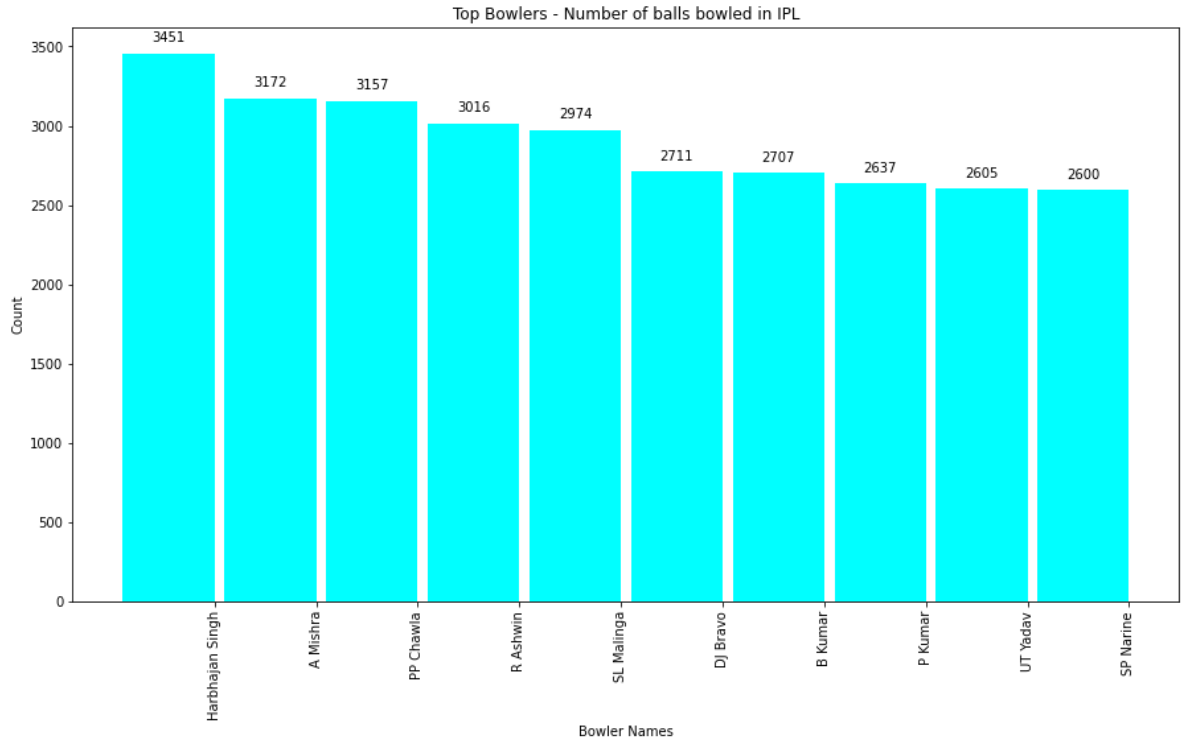
Batsman with more than 300 balls faced in taken and the ones with higher percentage of dot balls are seen. It is interesting to see Ganguly with more than 1000 balls and nearly half of them are dot balls. It is surprising to see names like Jayasuriya and Gibbs in there.!

## Bowler Analysis:

Now let us see the bowlers who has bowled most number of balls in IPL.

In [36]:

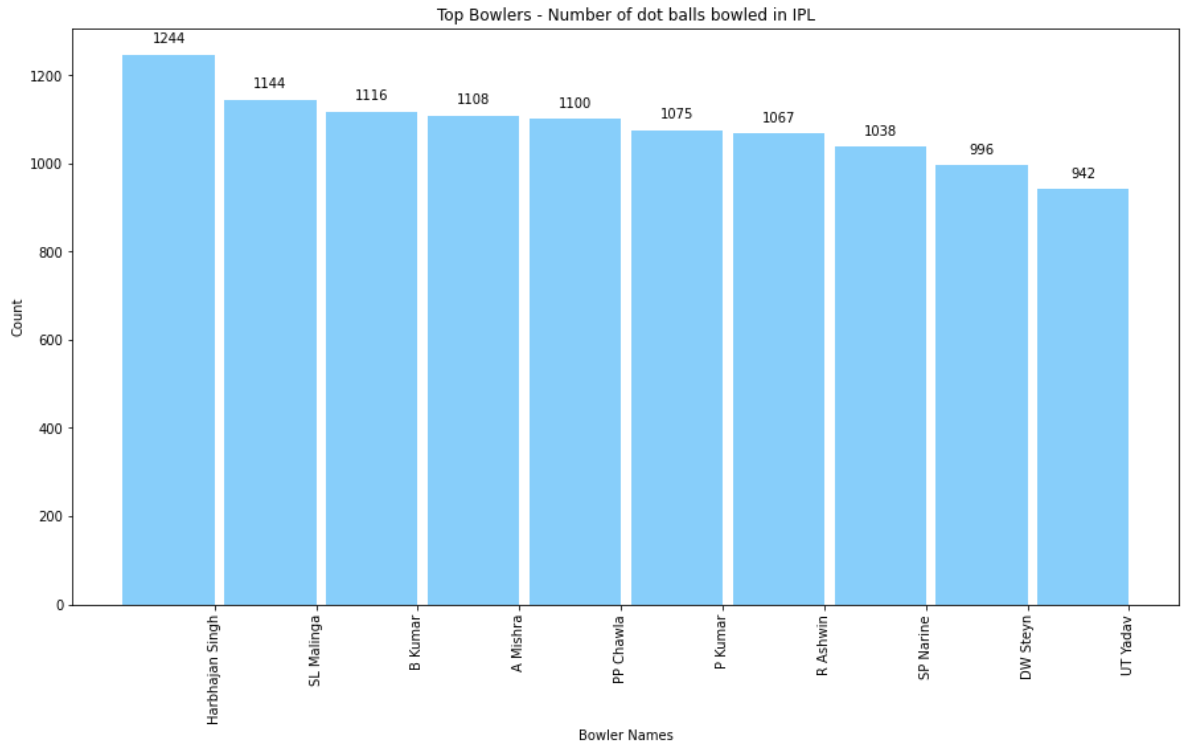
```
temp_df = score_df.groupby('bowler')['ball'].agg('count').reset_index().sort_values(by='bal
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['bowler'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['ball']), width=width, color='cyan')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Top Bowlers - Number of balls bowled in IPL")
ax.set_xlabel('Bowler Names')
autolabel(rects)
plt.show()
```




Harbhajan Singh is the the bowler with most number of balls bowled in IPL matches. Now let us see the bowler with more number of dot balls.

In [37]:

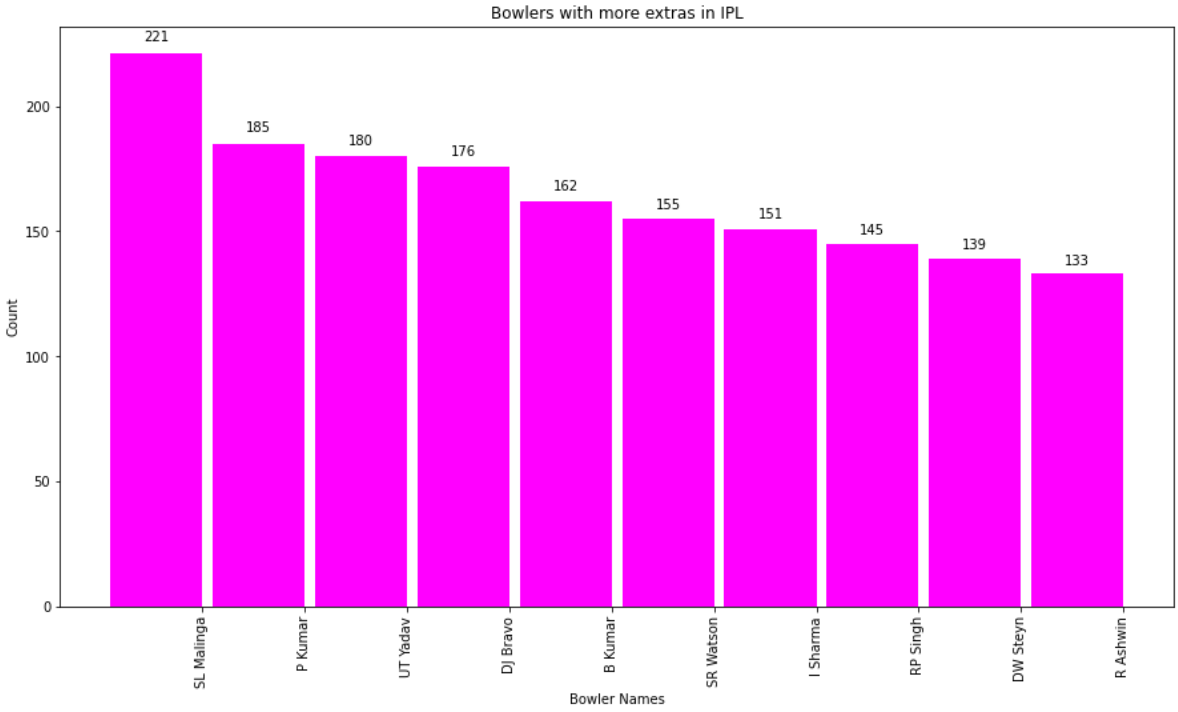
```
temp_df = score_df.groupby('bowler')['total_runs'].agg(lambda x: (x==0).sum()).reset_index(
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['bowler'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['total_runs']), width=width, color='lightskyblue')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Top Bowlers - Number of dot balls bowled in IPL")
ax.set_xlabel('Bowler Names')
autolabel(rects)
plt.show()
```




**Pravin Kumar** is the one with more number of dot balls followed by Steyn and Malinga

In [38]:

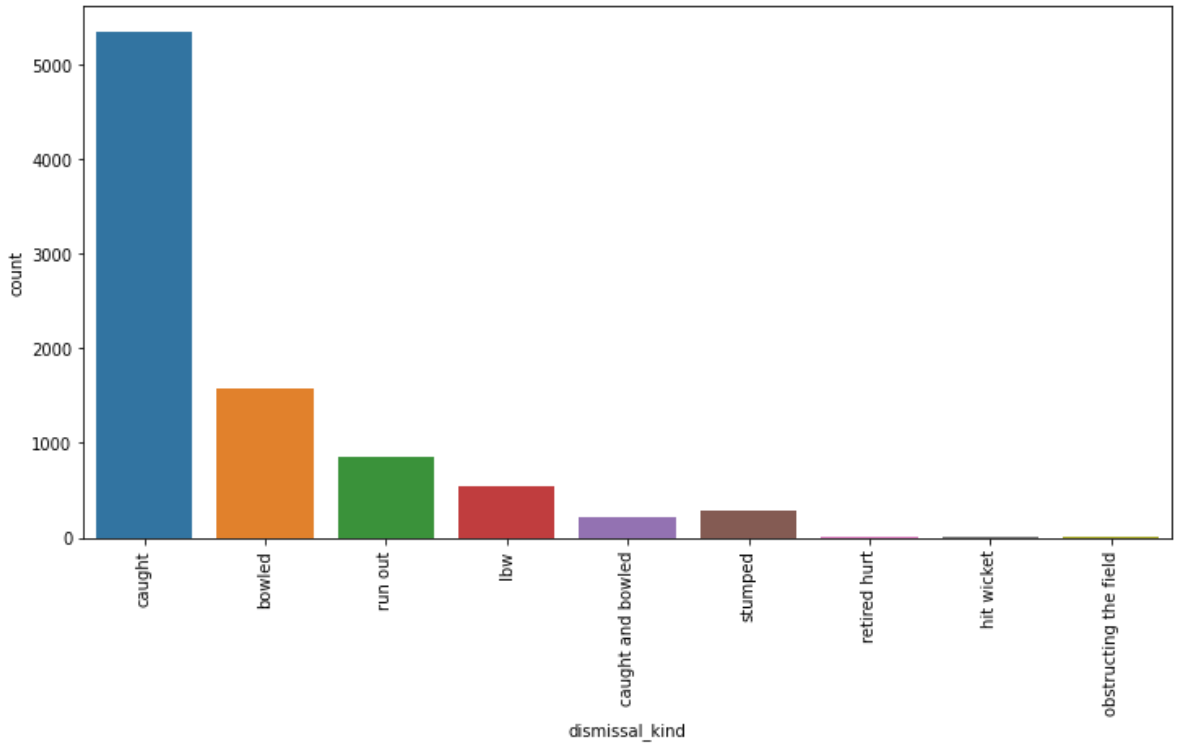
```
# Now let us see the bowlers who has bowled more number of extras in IPL.
temp_df = score_df.groupby('bowler')['extra_runs'].agg(lambda x: (x>0).sum()).reset_index()
temp_df = temp_df.iloc[:10,:]

labels = np.array(temp_df['bowler'])
ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['extra_runs']), width=width, color='magenta')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Bowlers with more extras in IPL")
ax.set_xlabel('Bowler Names')
autolabel(rects)
plt.show()
```




**Malinga** tops the chart with 221 extra runs followed by Pravin Kumar.

In [39]:

```
# Now let us see most common dismissal types in IPL.
plt.figure(figsize=(12,6))
sns.countplot(x='dismissal_kind', data=score_df)
plt.xticks(rotation='vertical')
plt.show()
```




**Caught** is the most common dismissal type in IPL followed by Bowled. There are very few instances of hit wicket as well. 'Obstructing the field' is one of the dismissal type as well in IPL.!