Exploratory Data Analysis - Sports

Author: Sidharth Kumar Mohanty

Data Science and Business Analytics Intern @ The Spark Foundation

Task #5 : "Exploratory Data Analysis : Sports (Indian Premier League)"

Dataset: Click here

Problem Statement :

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

1. Import Libraries

```
# import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

2. Load Datasets

```
# load matches dataset
matches_df = pd.read_csv("/content/matches.csv")

matches_df.head()
```

	id	season	city	date	team1	team2	toss_winner	toss_decision
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field
					Rising			

```
# load deliveries dataset
deliveries_df = pd.read_csv("deliveries.csv")

deliveries_df.head()
```

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

```
# merge matches & deliveries datasets
merge_df = pd.merge(deliveries_df,matches_df,left_on='match_id',right_on='id')

merge_df.head()
```

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

```
# size of each dataset
print("===========================================")
print("size of matches dataset : ",matches_df.shape )
print("===========================================")
print("size of deliveries dataset : ",deliveries_df.shape )
print("===========================================")
print("size of merge dataset : ",merge_df.shape )
print("===========================================")
```

```
===========================================
size of matches dataset :  (756, 18)
===========================================
size of deliveries dataset :  (179078, 21)
===========================================
size of merge dataset :  (179078, 39)
===========================================
```

3. EDA of Matches dataset

```
matches_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
```

```
# statistical analysis of matches_df
matches_df.describe(include='all')
```

	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_appl
756.000000	749	756	756	756	756	756	756	756.000
NaN	32	546	15	15	15	2	3	
NaN	Mumbai	2013-04-23	Mumbai Indians	Kolkata Knight Riders	Mumbai Indians	field	normal	
NaN	101	2	101	95	98	463	743	
013.444444	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.025
3.366895	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.156
008.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000
011.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000
013.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000
016.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000
019.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000

4. Handling Missing Values

```
# missing values in matches_df
matches_df.isnull().sum()
```

```
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
```

- Columns "city", "winner", "player_of_match", "umpire1", "umpire2" have missing values.
- Here "umpire3" column has maximum number of missing value present. So we should delete that column from the dataframe.

```
# drop "umpire3" column
matches_df.drop(["umpire3"],axis=1,inplace=True)
```

4.1. Handling Missing Values in "city" column

```
# find the venue name of all missing value "city"
matches_df[matches_df["city"].isnull()][["city","venue"]]
```

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

- As all missing values are from "Dubai International Cricket Stadium". So we can fill the missing value by "Dubai".

```
matches_df["city"] = matches_df["city"].fillna("Dubai")
```

- In matches_df "player_of_match", "umpirr1", and "umpire2" has 4,2,2 numbers of missing value. So we can delete these rows having missing values.

4.2. Handling Missing Values in "umpire1", "umpire2", "player_of_match" columns

```
# rows having missing values
matches_df[(matches_df["umpire1"].isnull()) | (matches_df["umpire2"].isnull()) | (matches_
```

	id	season	city	date	team1	team2	toss_winner	toss
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	
300	301	2011	Delhi	2011-05-21	Delhi Daredevils	Pune Warriors	Delhi Daredevils	
545	546	2015	Bangalore	2015-04-29	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	
570	571	2015	Bangalore	2015-05-17	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	
744	11340	2019	Bengaluru	30/04/19	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	
753	11413	2019	Visakhapatnam	08/05/19	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	

```
# delete rows having missing value in columns 'umpire1', 'umpire2', 'player_of_match'.
matches_df.dropna(subset=['umpire1', 'umpire2', 'player_of_match'],inplace=True)
```

```
# shape of updated matches_df DataFrame
matches_df.shape
```

```
(750, 17)
```

5. EDA of Deliveries Dataset

```
deliveries_df.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
```

```
# statistical analysis of deliveries dataset
deliveries_df.describe()
```

	match_id	inning	over	ball	is_super_over	
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179
mean	1802.252957	1.482952	10.162488	3.615587	0.000452	
std	3472.322805	0.502074	5.677684	1.806966	0.021263	
min	1.000000	1.000000	1.000000	1.000000	0.000000	
25%	190.000000	1.000000	5.000000	2.000000	0.000000	
50%	379.000000	1.000000	10.000000	4.000000	0.000000	
75%	567.000000	2.000000	15.000000	5.000000	0.000000	
max	11415.000000	5.000000	20.000000	9.000000	1.000000	

6. Handling Missing Values

```
# see how many missing value present each column
deliveries_df.isnull().sum()
```

```
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
```

- Here we can see column "player_dismissed", "dismissal_kind", "fielder" have maximum(more than 90%) number of missing value present.
- So we should delete these columns.

```
# drop columns "player_dismissed","dismissal_kind","fielder" from the DataFrame
deliveries_df.drop(columns=["player_dismissed","dismissal_kind","fielder"],axis=1,inplace=
```

```
# check for any missing value in deliveries_df
deliveries_df.isnull().sum().sum()
```

```
0
```

```
# check for any missing value in matches_df
matches_df.isnull().sum().sum()
```

```
0
```

Now both the datasets are clean i.e there is no missing value present.

```
matches_df.tail()
```

id	season	city	date	team1	team2	toss_winner	toss_decision	resu
346	2019	Mohali	05/05/19	Chennai Super	Kings XI Punjab	Kings XI Punjab	field	norr

```
deliveries_df.head()
```

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

7. Number of Teams Participated Each Season

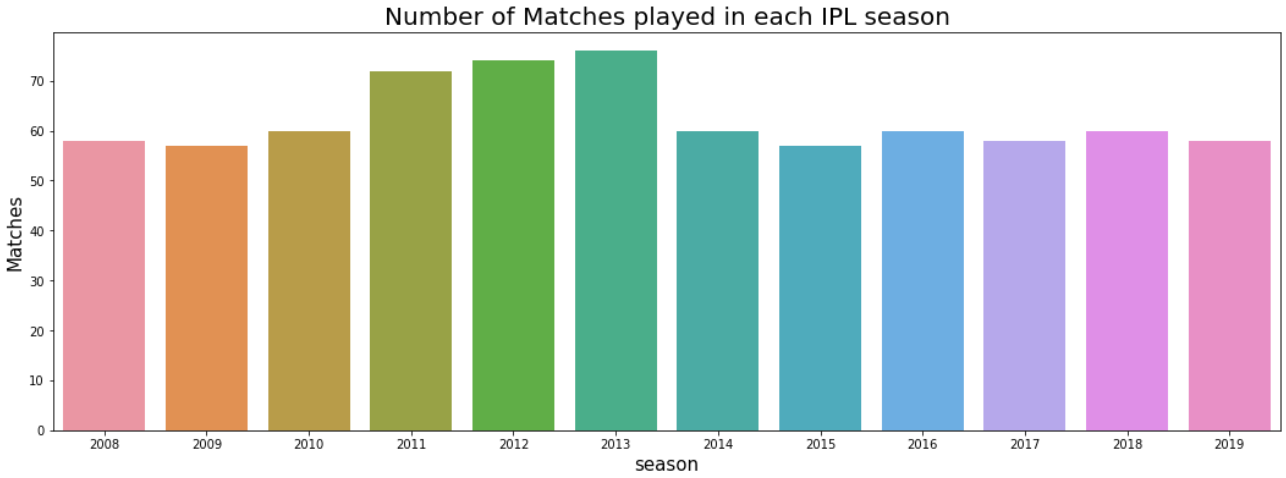
```
matches_df.groupby('season')['team1'].nunique().plot(kind = 'bar', figsize=(15,5),color =
plt.title("Number of teams participated each season ",fontsize=18,fontweight="bold")
plt.ylabel("Count of teams", size = 25)
plt.xlabel("Season", size = 25)
plt.xticks(size = 15)
plt.yticks(size = 15)
```

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

- In the year of 2011, 2012, 2013, there were 10,9,9 teams participated while in other seasons participated teams were 8.

8. Matches Played in Each Season

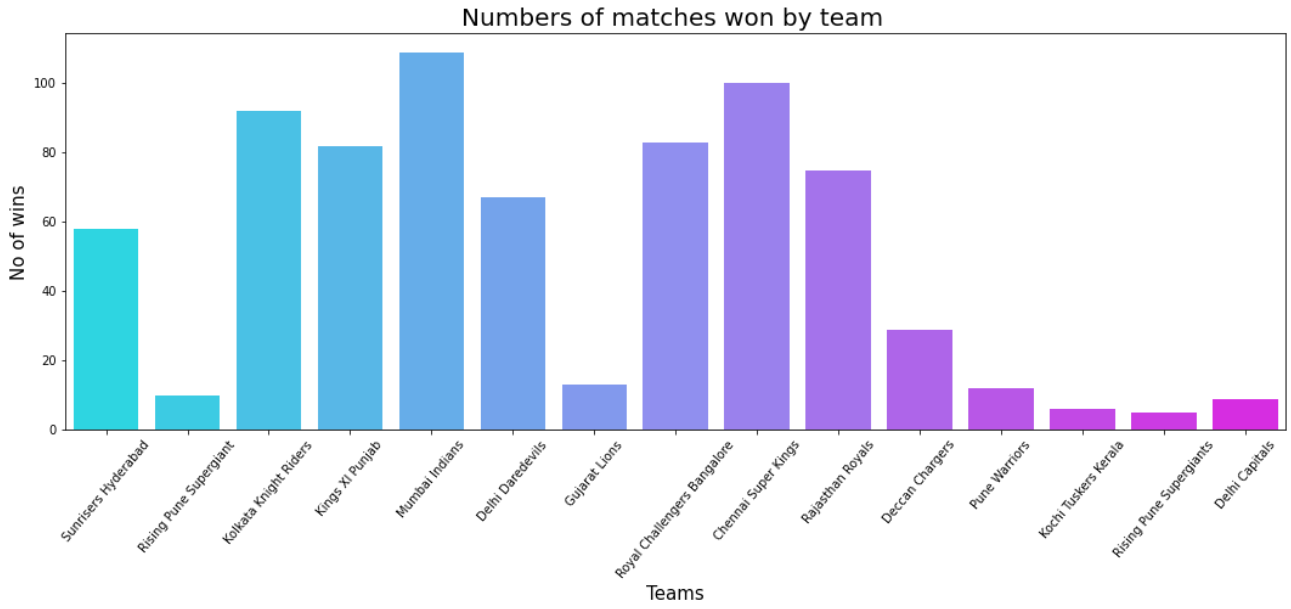
```
plt.figure(figsize = (18,6))
sns.countplot('season',data=matches_df,)
plt.title("Number of Matches played in each IPL season",fontsize=20)
plt.xlabel("season",fontsize=15)
plt.ylabel('Matches',fontsize=15)
plt.show()
```



9. Number of Matches Won by Team

```
plt.figure(figsize = (18,6))
sns.countplot(x='winner',data=matches_df, palette='cool')
plt.title("Numbers of matches won by team ",fontsize=20)
plt.xticks(rotation=50)
plt.xlabel("Teams",fontsize=15)
plt.ylabel("No of wins",fontsize=15)
```

```
plt.show()
```



- Mumbai Indians has maximum number of winning matches followed by Chennai Super Kings.
- In matches_df DataFrame, "city" column has 32 unique values while "venue" column has 41 distinct values.
- Let's find out which city has many number of venues.

```
# find how many stadium present in each cities
city_venue = matches_df.groupby(['city','venue']).count()['season']
city_venue_df = pd.DataFrame(city_venue)
city_venue_df
```

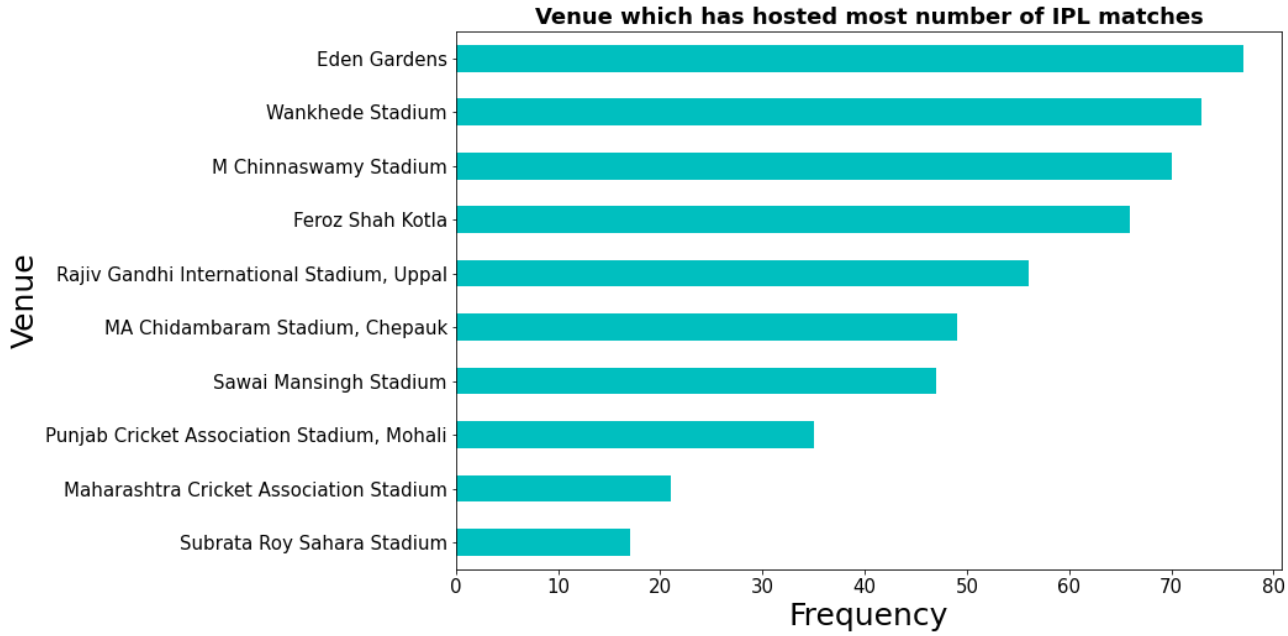
city	venue	season
Abu Dhabi	Sheikh Zayed Stadium	7
Ahmedabad	Sardar Patel Stadium, Motera	12
Bangalore	M Chinnaswamy Stadium	63
Bengaluru	M Chinnaswamy Stadium	7
	M. Chinnaswamy Stadium	6
Bloemfontein	OUTsurance Oval	2
Cape Town	Newlands	7
Centurion	SuperSport Park	12
Chandigarh	Punjab Cricket Association 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	66
	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

	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

10. Venue which has hosted most number of IPL matches

```
# matches_df["venue"].value_counts().sort_values(ascending = True).tail(10)
matches_df["venue"].value_counts().sort_values(ascending = True).tail(10).plot(kind = 'bar
plt.title("Venue which has hosted most number of IPL matches",fontsize=18,fontweight="bold
plt.ylabel("Venue", size = 25)
plt.xlabel("Frequency", size = 25)
```

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

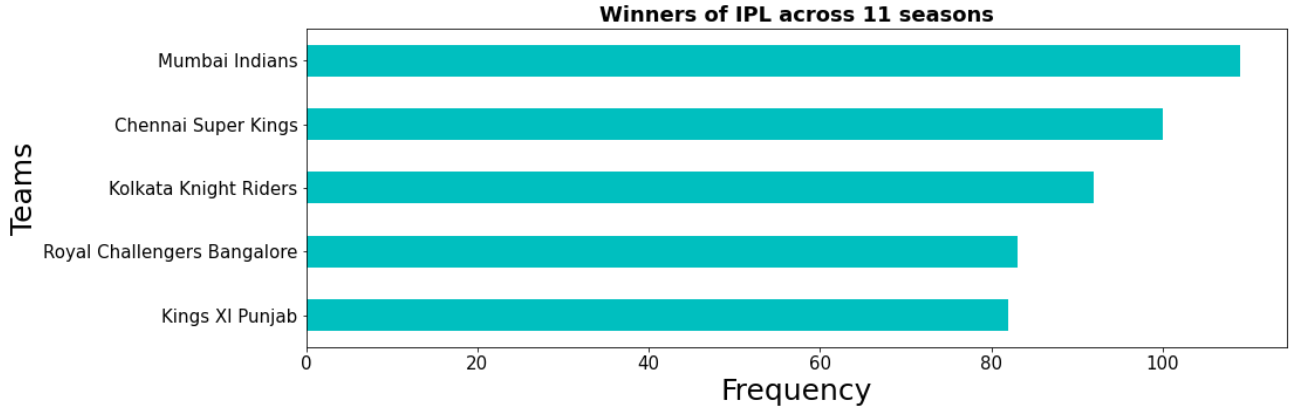


11. Which Team has maximum number of win in IPL so far

```
matches_df["winner"].value_counts().sort_values(ascending = True).tail().plot(kind = 'barh
plt.title("Winners of IPL across 11 seasons",fontsize=18,fontweight="bold")
plt.ylabel("Teams", size = 25)
plt.xlabel("Frequency", size = 25)
```

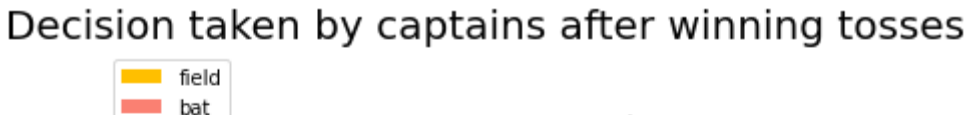
```
plt.xticks(size = 15)
plt.yticks(size = 15)
```

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



12. Does teams choose to bat or field first, after winning toss ?

```
colors = ['#FFBF00', '#FA8072']
matches_df['toss_decision'].value_counts().plot(kind='pie', fontsize=14, autopct='%3.1f%%'
                                                figsize=(10,7), shadow=True, startangle=135
plt.ylabel('Toss Decision')
plt.title('Decision taken by captains after winning tosses', size = 20)
plt.show()
```

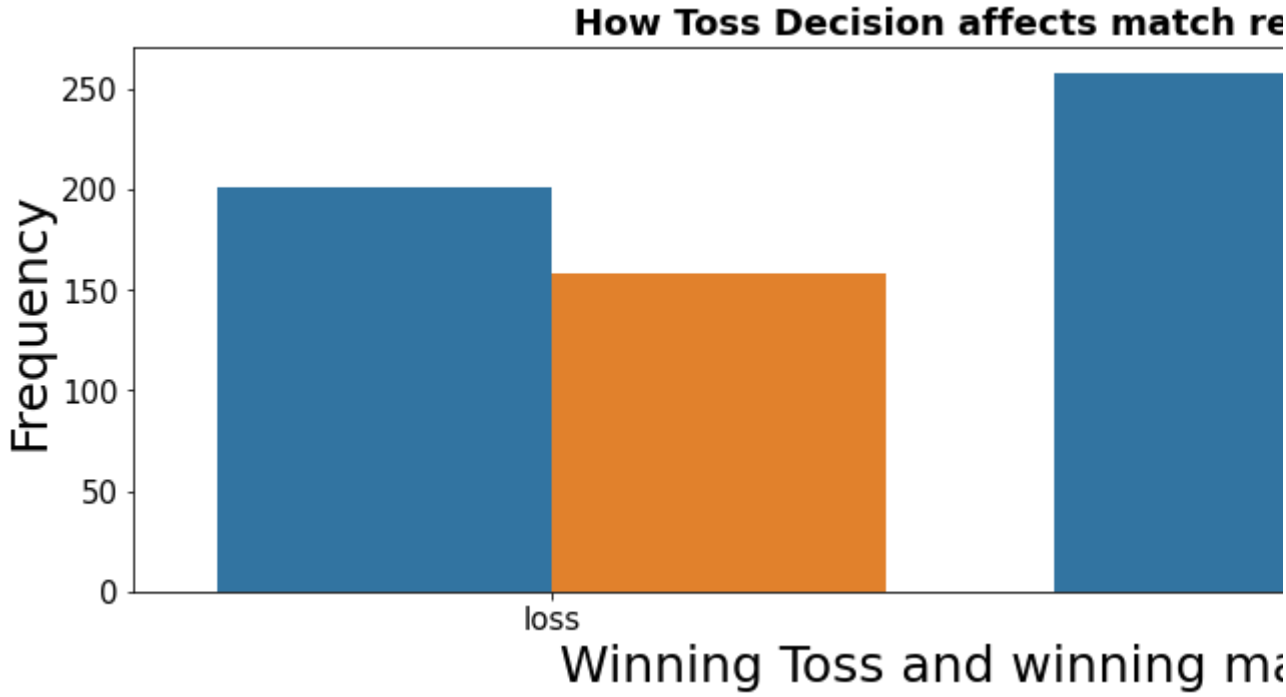


- Usually after winning the toss, team choose to field first.

13. How toss decision affects match results ?

```
# create a column which store 'win' if a team win a match &
matches_df['toss_win_game_win'] = np.where((matches_df.toss_winner == matches_df.winner),'
plt.figure(figsize = (15,5))
sns.countplot('toss_win_game_win', data=matches_df, 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)
```

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

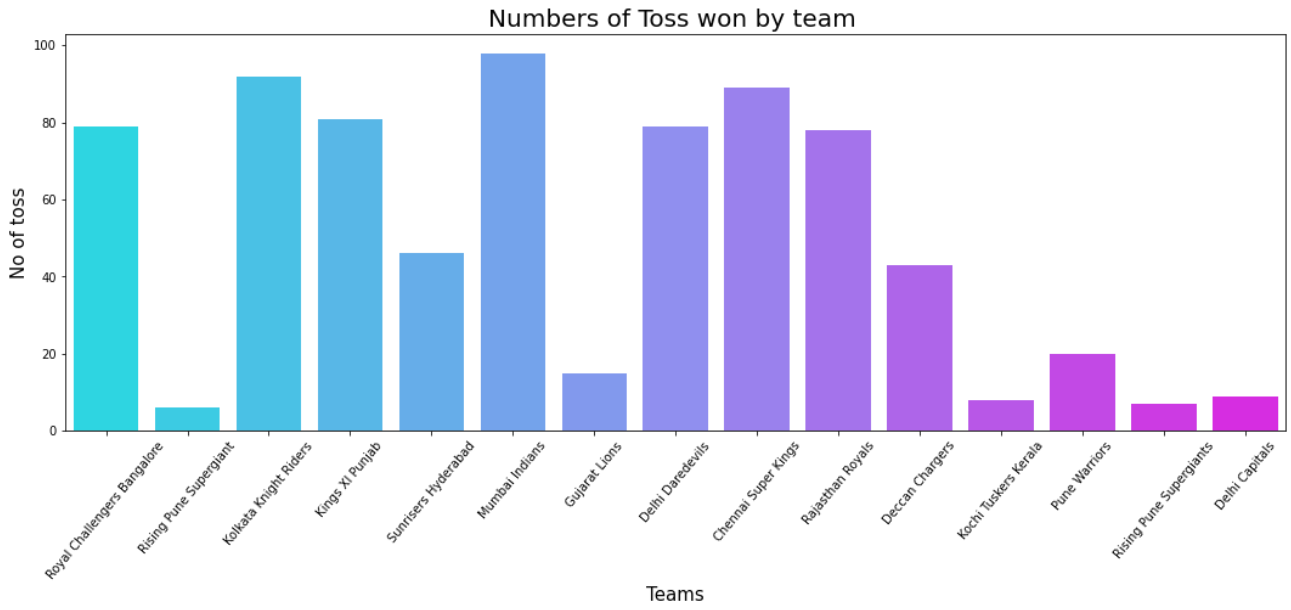


- After winning the toss the team who choose to field first has higher probability of winning the match.

14. Number of Toss won by individual team

```
plt.figure(figsize = (18,6))
sns.countplot(x='toss_winner',data=matches_df, palette='cool')
plt.title("Number of Toss won by team ",fontsize=20)
plt.xticks(rotation=50)
plt.xlabel("Teams",fontsize=15)
```

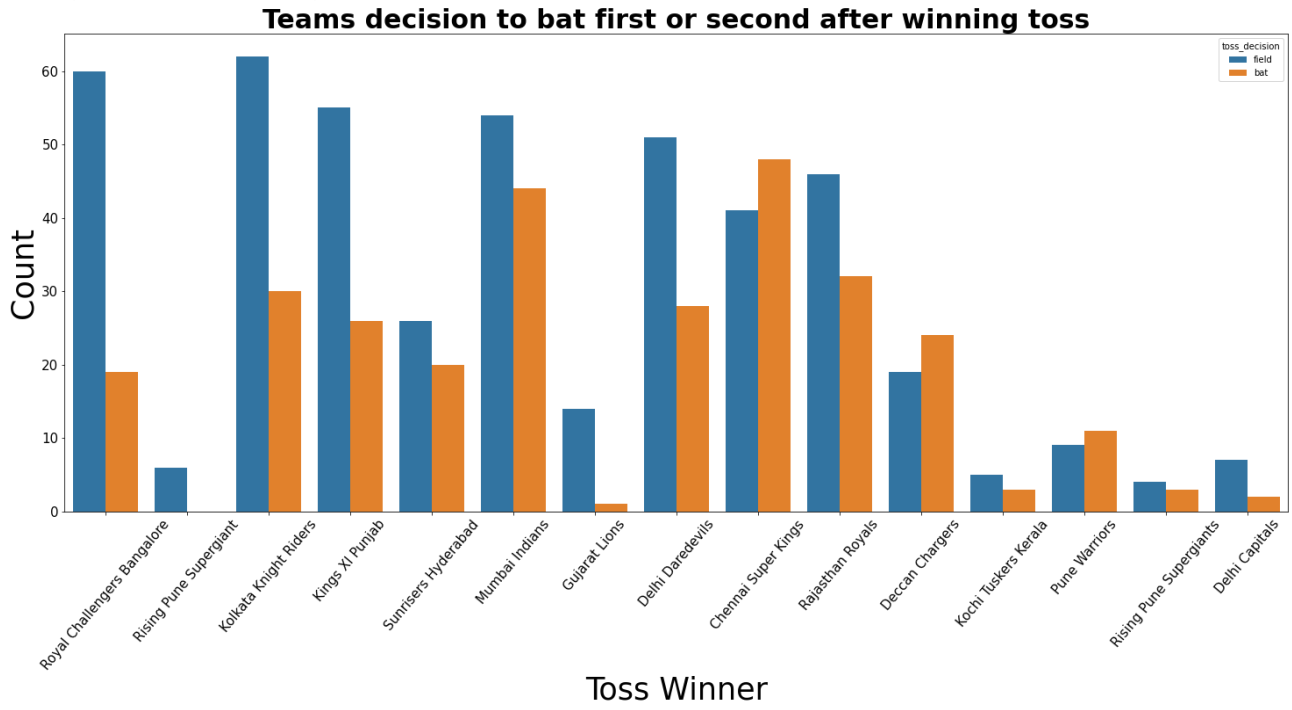
```
plt.ylabel("No of toss",fontsize=15)
plt.show()
```



15. Individual teams decision to choose bat first or second after winning toss

```
plt.figure(figsize = (25,10))
sns.countplot('toss_winner', data = matches_df, hue = 'toss_decision')
plt.title("Teams decision to bat first or second after winning toss", size = 30, fontweigh
plt.xticks(size = 15, rotation=50)
plt.yticks(size = 15)
plt.xlabel("Toss Winner", size = 35)
plt.ylabel("Count", size = 35)
```

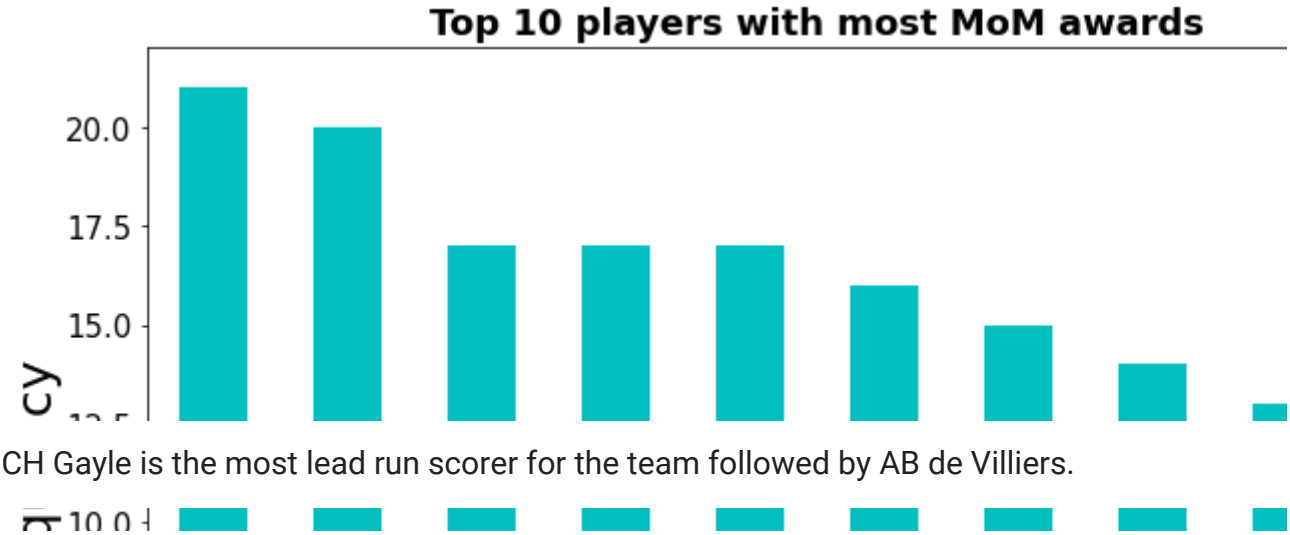
```
Text(0, 0.5, 'Count')
```



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

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

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

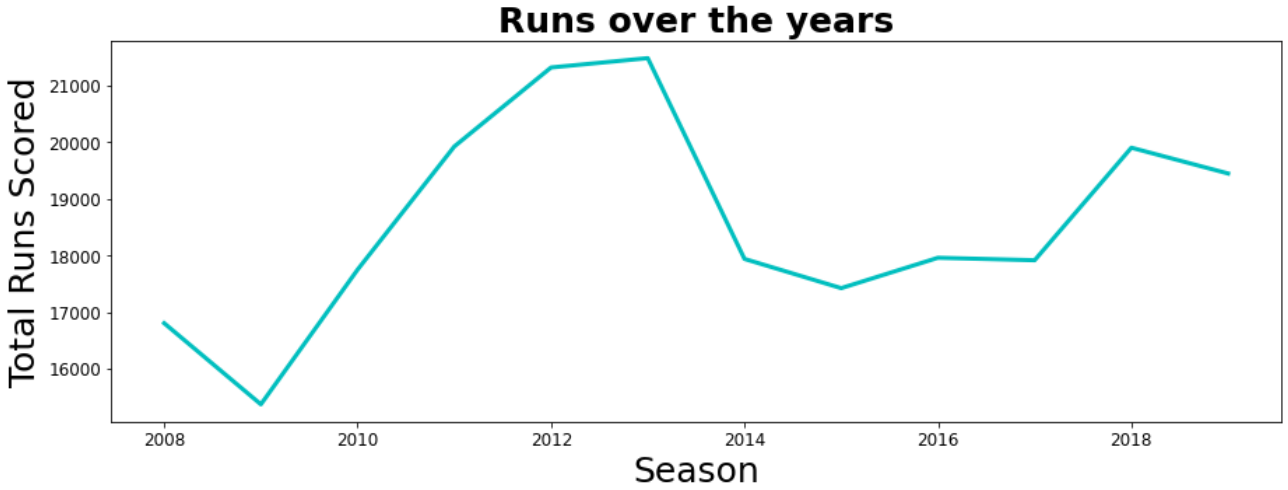


- CH Gayle is the most lead run scorer for the team followed by AB de Villiers.

17. Teams total scoring runs over the years?

```
merge_df.groupby('season')['batsman_runs'].sum().plot(kind = 'line', linewidth = 3, figsiz

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)
```

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

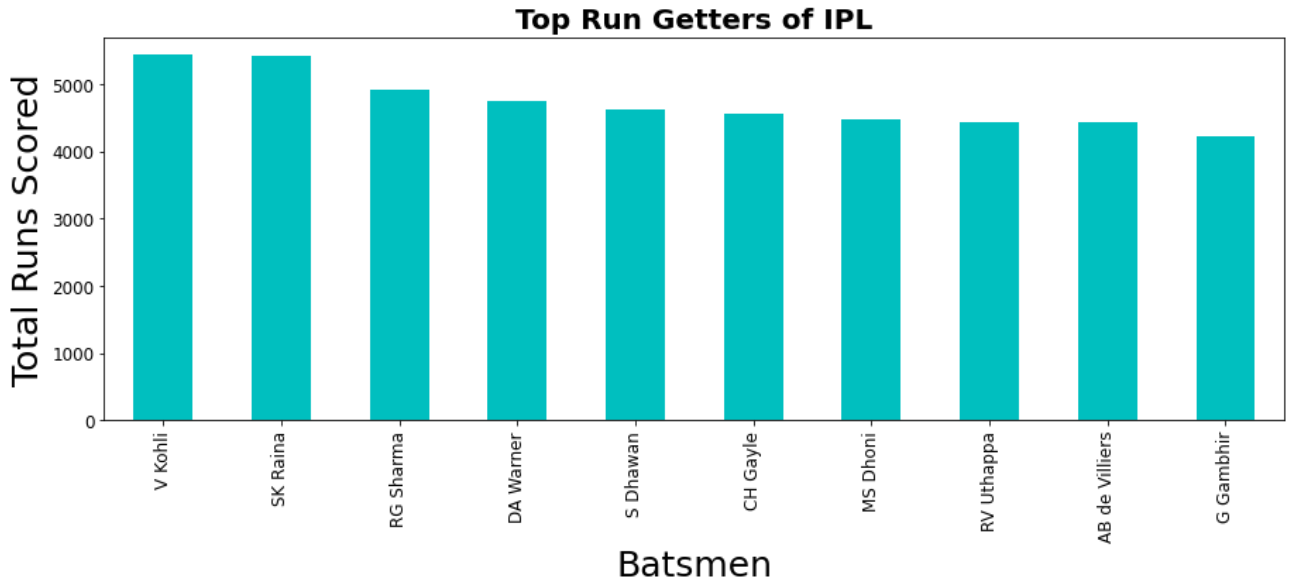


18. Top Run Getters of IPL

```
#let's plot the top 10 run getter so far in IPL
```

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

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)
```

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

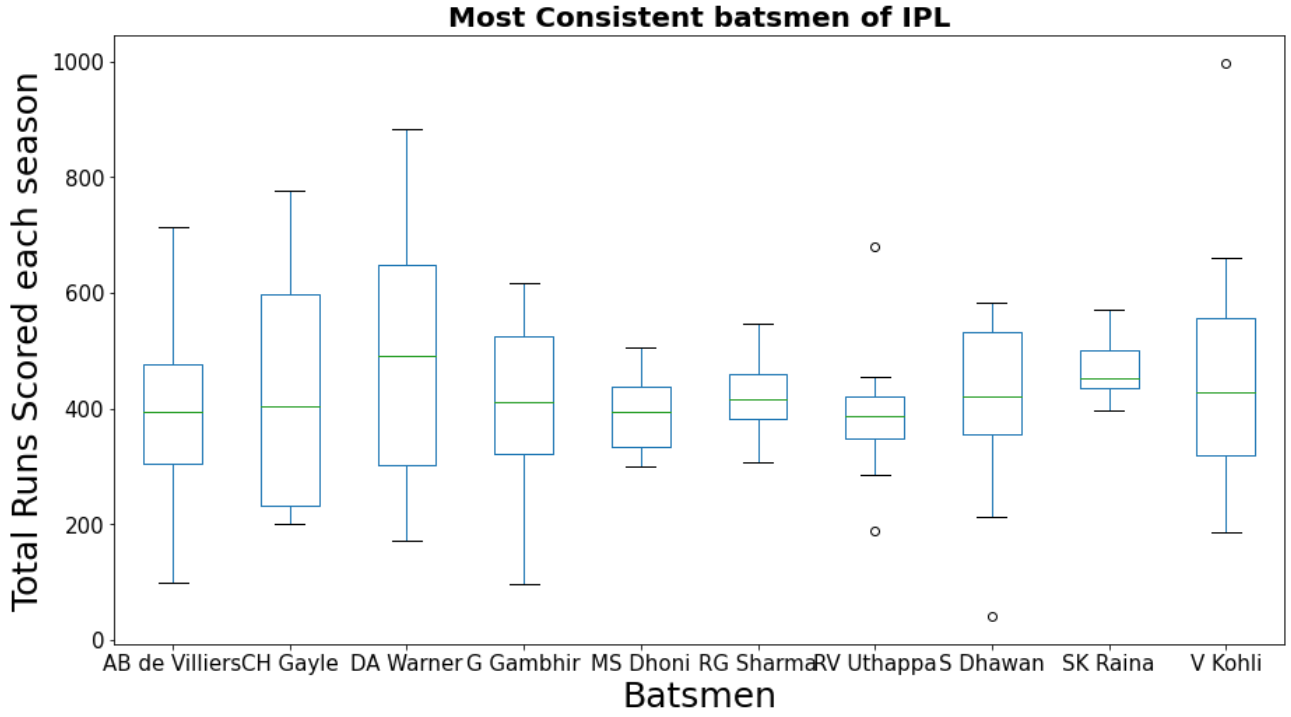


- Virat Kohli is the top run getter of IPL in all over the seasons

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

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

consistent_batsman.groupby(['season','batsman'])['total_runs'].sum().unstack().plot(kind =
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)
```

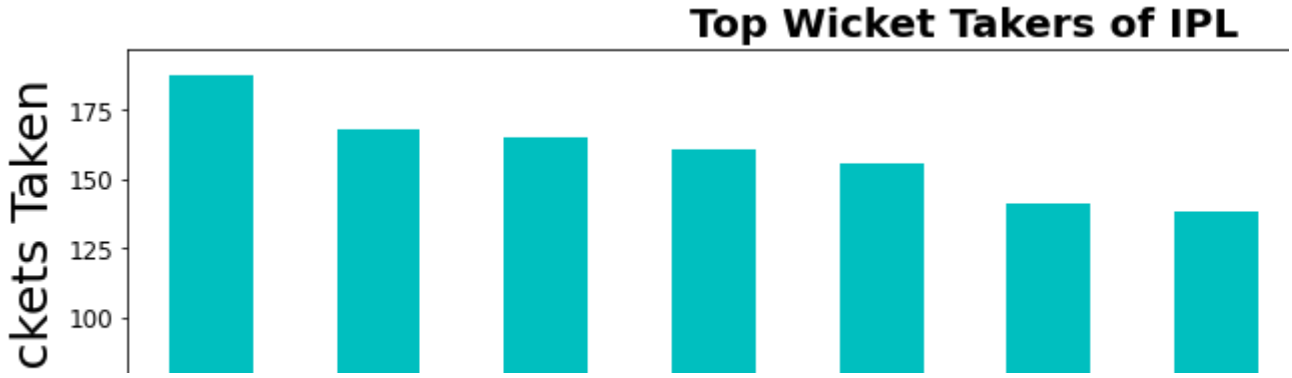
```
(array([-200.,    0.,  200.,  400.,  600.,  800., 1000., 1200.]),
 <a list of 8 Text major ticklabel objects>)
```



20. Top Wicket Takers of IPL

```
merge_df.groupby('bowler')['player_dismissed'].count().sort_values(ascending = False).head
                                                color = 'c', 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)
```

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



- SL Malinga is the top wicket taker of IPL

21. Batsmen with the best strike rates over the years

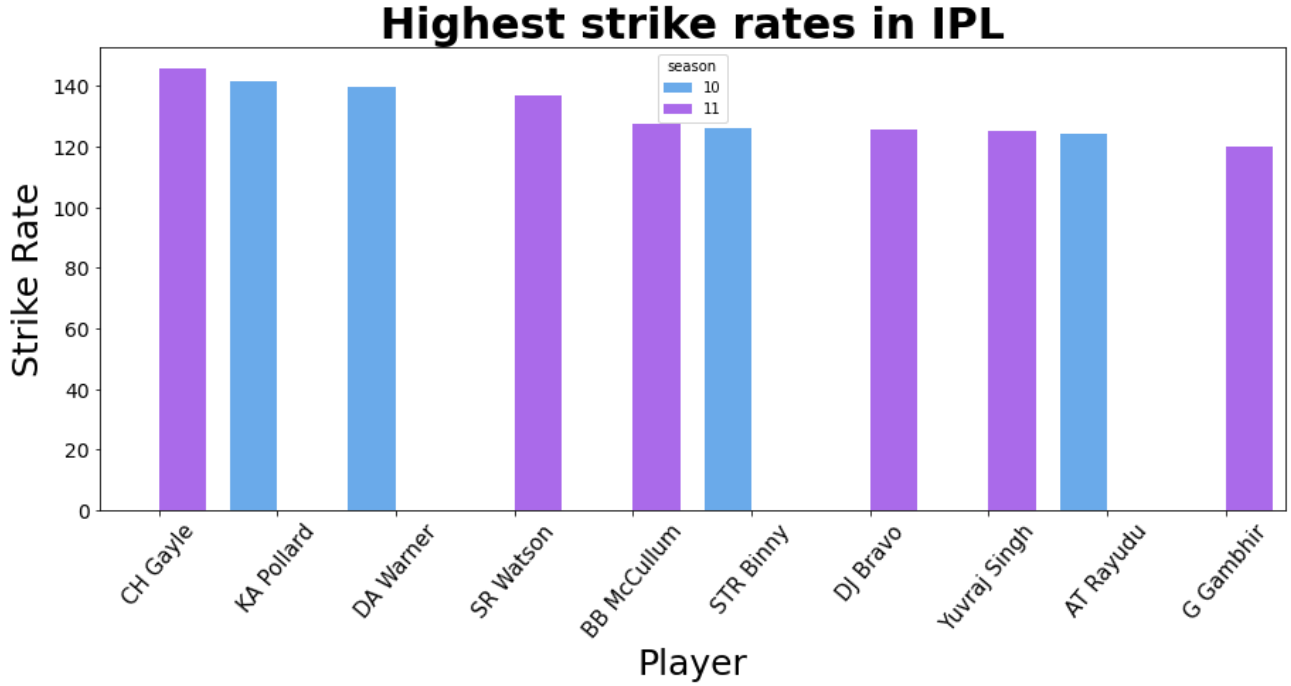
```
#We will consider players who have played 10 or more seasons
no_of_balls = pd.DataFrame(merge_df.groupby('batsman')['ball'].count()) #total number of m
runs = pd.DataFrame(merge_df.groupby('batsman')['batsman_runs'].sum()) #total runs of each
seasons = pd.DataFrame(merge_df.groupby('batsman')['season'].nunique())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

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

batsman_strike_rate['strike_rate'] = batsman_strike_rate['run']/batsman_strike_rate['balls
highest_strike_rate = batsman_strike_rate[batsman_strike_rate.season.isin([10,11])][['seas


highest_strike_rate.head(10)
```

	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

```
plt.figure(figsize = (15,6))
sns.barplot(x='batsman', y='strike_rate', data = highest_strike_rate.head(10), hue = 'seas
```

```
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 = 15, rotation=50)
plt.yticks(size = 14)
```

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



Q-1: As a sports analysts, find out the most successful teams, players and 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.
- 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

Q-2: Suggest 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.

Happy Learning!!!

✓ 0s completed at 11:50 PM