Indian Premier League Analysis

One of the most well-known and avidly watched sporting events in India and across the world is the Indian Premier competition (IPL), a professional Twenty20 cricket competition. The Indian Premier League, which was founded in 2008, consists of franchise clubs from different Indian cities. It's a spectacular spectacle for viewers, bringing together world-class cricket talent with high-pressure, short-format contests. The league has a big influence on the media, fan involvement, and economics of cricket as well as providing a stage for both seasoned and up-and-coming players to display their abilities. Beyond only cricket, the Indian Premier League (IPL) has cultural and social importance as well. It brings together the top players in an exciting and fast-paced style that continues to captivate cricket fans across the world, and it unites millions of people in celebration and competition.

Importing Libraries

```
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, plot, iplot
from plotly import tools
from warnings import filterwarnings
filterwarnings('ignore')
```

Importing Dataset

```
deliveries_data = pd.read_csv('/content/IPL Ball-by-Ball 2008-2020.csv')
match_data = pd.read_csv('/content/IPL Matches 2008-2020.csv')
```

Basic Data Exploration

```
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_d
0	335982	Bangalore	4/18/2008	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	
1	335983	Chandigarh	4/19/2008	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	
2	335984	Delhi	4/19/2008	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	
3	335985	Mumbai	4/20/2008	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	
4	335986	Kolkata	4/20/2008	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	

```
deliveries_data.head()
```

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissa
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	
2	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	
3	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	0	0	
4	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	0	0	

```
match_data.isnull().sum()
```

```
id                   0
city                13
date                 0
player_of_match      4
venue                0
neutral_venue        0
team1                0
team2                0
toss_winner          0
toss_decision        0
winner               4
result               4
result_margin       17
eliminator           4
method             797
umpire1              0
umpire2              0
dtype: int64
```

```
match_data.shape
```

```
(816, 17)
```

```
match_data.columns
```

```
Index(['id', 'city', 'date', 'player_of_match', 'venue', 'neutral_venue',
       'team1', 'team2', 'toss_winner', 'toss_decision', 'winner', 'result',
       'result_margin', 'eliminator', 'method', 'umpire1', 'umpire2'],
      dtype='object')
```

```
print('Total Matches Played:',match_data.shape[0])
```

```
Total Matches Played: 816
```

```
print('Venues Played At:',match_data['city'].unique())
```

```
Venues Played At: ['Bangalore' 'Chandigarh' 'Delhi' 'Mumbai' 'Kolkata' 'Jaipur' 'Hyderabad'
 'Chennai' 'Cape Town' 'Port Elizabeth' 'Durban' 'Centurion' 'East London'
 'Johannesburg' 'Kimberley' 'Bloemfontein' 'Ahmedabad' 'Cuttack' 'Nagpur'
 'Dharamsala' 'Kochi' 'Indore' 'Visakhapatnam' 'Pune' 'Raipur' 'Ranchi'
 'Abu Dhabi' nan 'Rajkot' 'Kanpur' 'Bengaluru' 'Dubai' 'Sharjah']
```

```
print('Teams :',match_data['team1'].unique())
```

```
Teams : ['Royal Challengers Bangalore' 'Kings XI Punjab' 'Delhi Daredevils'
 'Mumbai Indians' 'Kolkata Knight Riders' 'Rajasthan Royals'
 'Deccan Chargers' 'Chennai Super Kings' 'Kochi Tuskers Kerala'
 'Pune Warriors' 'Sunrisers Hyderabad' 'Gujarat Lions'
 'Rising Pune Supergiants' 'Rising Pune Supergiant' 'Delhi Capitals']
```

Number of Matches Played in Various Seasons

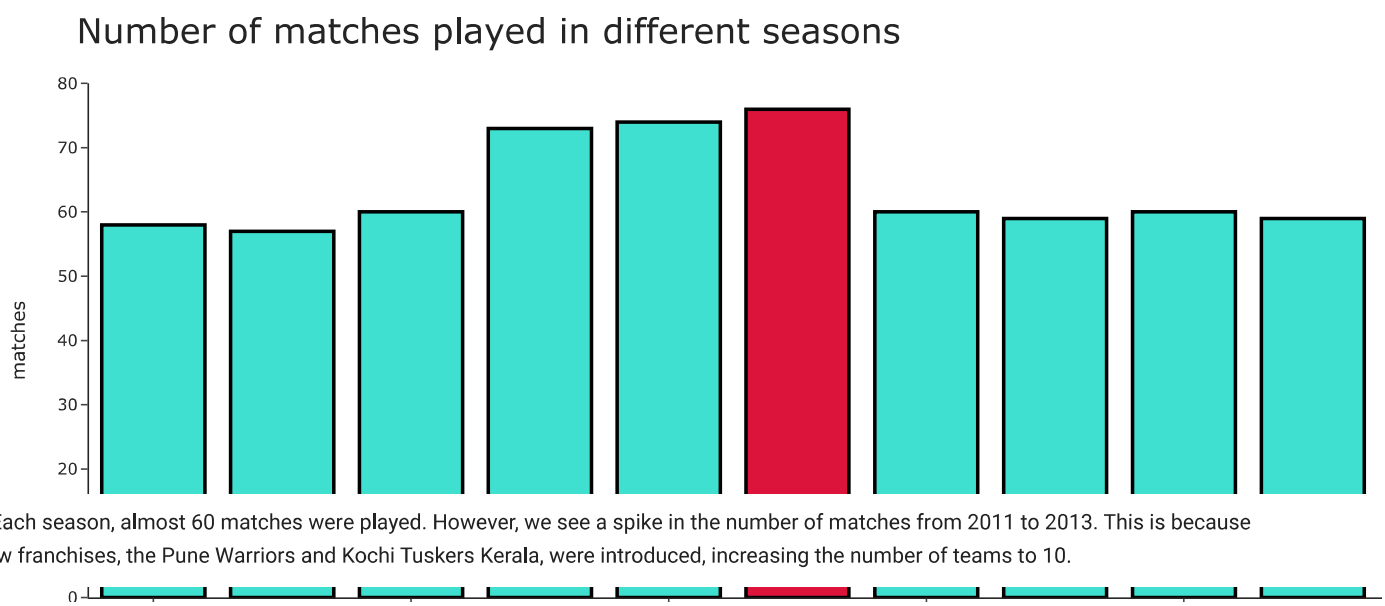
```
match_data['Season'] = pd.DatetimeIndex(match_data['date']).year
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decisio
0	335982	Bangalore	4/18/2008	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	fiel
1	335983	Chandigarh	4/19/2008	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	ba
2	335984	Delhi	4/19/2008	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	ba
3	335985	Mumbai	4/20/2008	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	ba
4	335986	Kolkata	4/20/2008	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	ba

```
match_per_season=match_data.groupby(['Season'])['id'].count().reset_index().rename(columns={'id':'matches'})
match_per_season.style.background_gradient(cmap='PuBu')
```

	Season	matches
0	2008	58
1	2009	57
2	2010	60
3	2011	73
4	2012	74
5	2013	76
6	2014	60
7	2015	59
8	2016	60
9	2017	59
10	2018	60
11	2019	60
12	2020	60

```
colors = ['turquoise',] * 13
colors[5] = 'crimson'
fig=px.bar(data_frame=match_per_season,x=match_per_season.Season,y=match_per_season.matches,labels=dict(x="Season",y="Count"),)
fig.update_layout(title="Number of matches played in different seasons ",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



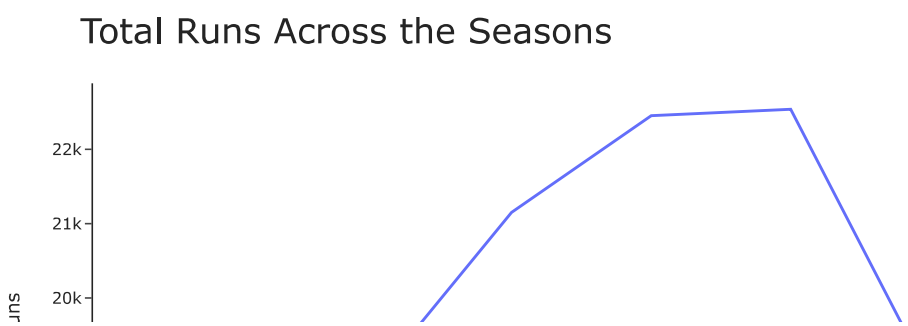
Note: Each season, almost 60 matches were played. However, we see a spike in the number of matches from 2011 to 2013. This is because two new franchises, the Pune Warriors and Kochi Tuskers Kerala, were introduced, increasing the number of teams to 10.

Total Number of Runs Scored Across Seasons

```
season_data=match_data[['id','Season']].merge(deliveries_data, left_on = 'id', right_on = 'id', how = 'left').drop('id', axis = 1)
season_data.head()
```

	Season	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissa
0	2008	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	
1	2008	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	
2	2008	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	
3	2008	1	7	2	BB McCullum	RT Ponting	Z Khan	1	0	1	0	0	
4	2008	1	7	3	RT Ponting	BB McCullum	Z Khan	1	0	1	0	0	

```
season=season_data.groupby(['Season'])['total_runs'].sum().reset_index()
p=season.set_index('Season')
fig = px.line(p, x=p.index, y="total_runs")
fig.update_layout(title="Total Runs Across the Seasons ",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.show()
```



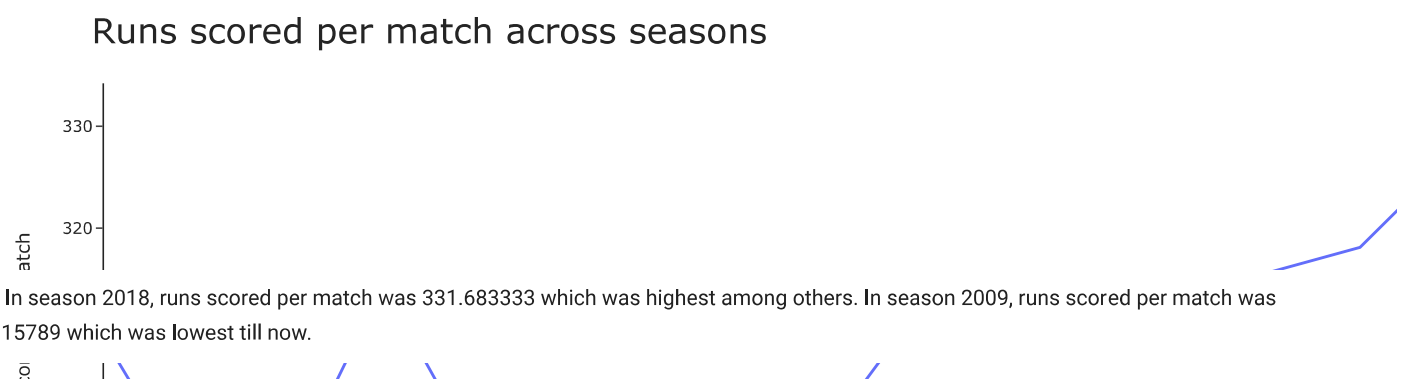
Note: Season 2013 was the highest scoring season (22,541 runs), followed by 2012 (22,453 runs) Season 2009 was the lowest scoring season (16,320 runs).

Runs Scored Per Match Across Seasons

```
runs_per_season=pd.concat([match_per_season,season.iloc[:,1]],axis=1)
runs_per_season['Runs scored per match']=runs_per_season['total_runs']/runs_per_season['matches']
runs_per_season.set_index('Season',inplace=True)
runs_per_season.style.background_gradient(cmap='PuBu',subset=['Runs scored per match'])
```

Season	matches	total_runs	Runs scored per match
2008	58	17937	309.258621
2009	57	16320	286.315789
2010	60	18864	314.400000
2011	73	21154	289.780822
2012	74	22453	303.418919
2013	76	22541	296.592105
2014	60	18909	315.150000
2015	59	18332	310.711864
2016	60	18862	314.366667
2017	59	18769	318.118644
2018	60	19901	331.683333
2019	60	19400	323.333333
2020	60	19352	322.533333

```
fig = px.line(runs_per_season, x=runs_per_season.index, y="Runs scored per match")
fig.update_layout(title="Runs scored per match across seasons",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.show()
```



Note: In season 2018, runs scored per match was 331.683333 which was highest among others. In season 2009, runs scored per match was 286.315789 which was lowest till now.

Count of Matches By Umpires

```
ump=pd.concat([match_data['umpire1'],match_data['umpire2']])
ump=ump.value_counts()
umps=ump.to_frame().reset_index()
ump.head(10)
```

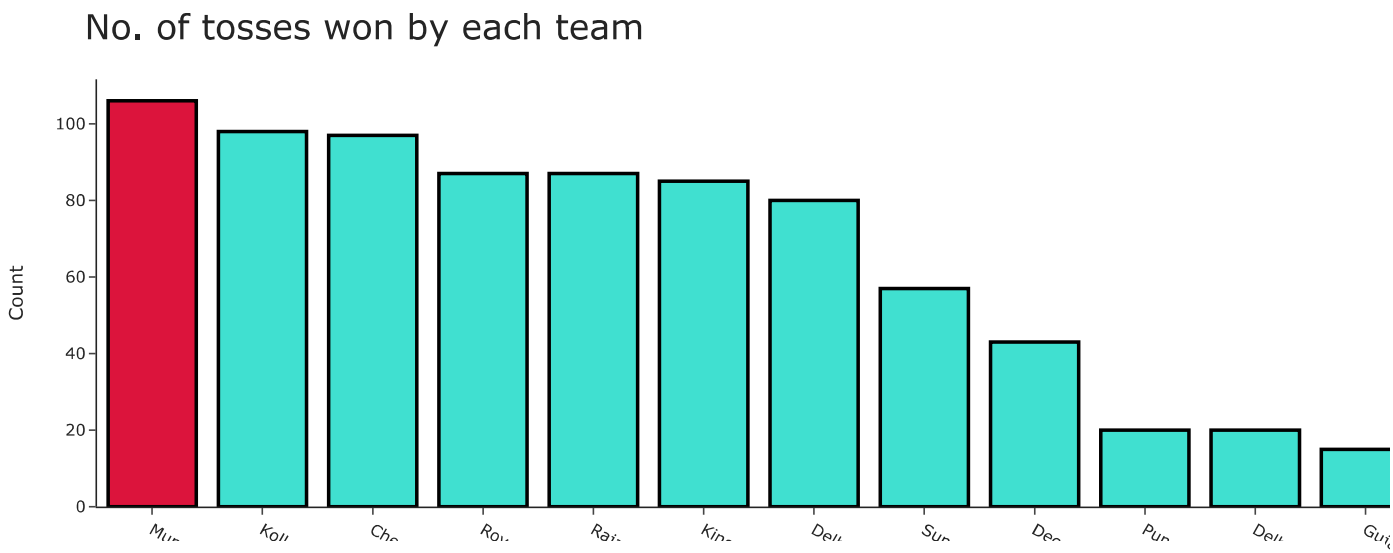
```
S Ravi             121
HDPK Dharmasena     94
AK Chaudhary        87
C Shamshuddin       82
M Erasmus           65
CK Nandan           57
Nitin Menon         57
SJA Taufel          55
Asad Rauf           51
VA Kulkarni         50
dtype: int64
```

Number of Tosses Won by Teams

```
match_data['toss_winner'].value_counts()
```

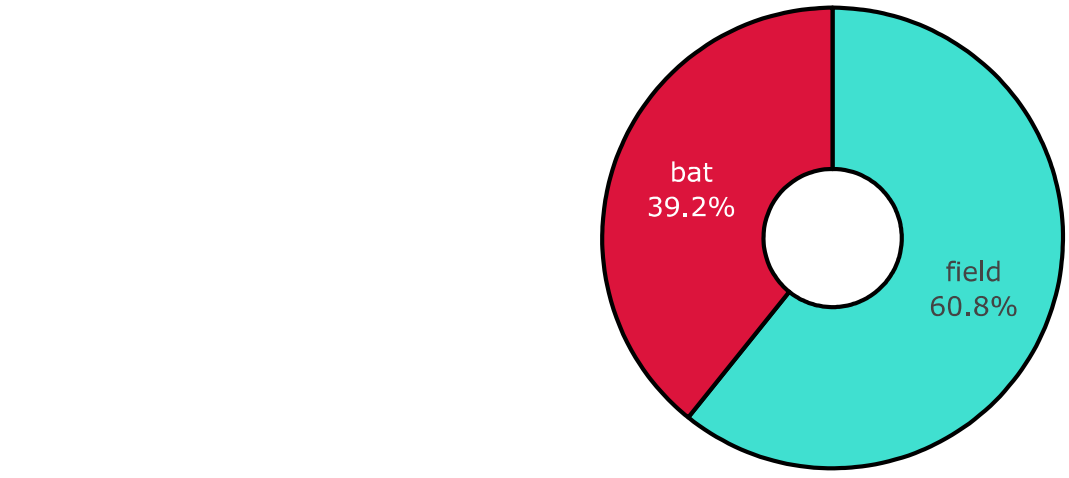
```
Mumbai Indians                106
Kolkata Knight Riders          98
Chennai Super Kings            97
Royal Challengers Bangalore    87
Rajasthan Royals               87
Kings XI Punjab                85
Delhi Daredevils               80
Sunrisers Hyderabad            57
Deccan Chargers                43
Pune Warriors                  20
Delhi Capitals                 20
Gujarat Lions                  15
Kochi Tuskers Kerala            8
Rising Pune Supergiants         7
Rising Pune Supergiant          6
Name: toss_winner, dtype: int64
```

```
toss=match_data['toss_winner'].value_counts()
colors = ['turquoise',] * 15
colors[0] = 'crimson'
fig=px.bar( y=toss,x=toss.index,labels=dict(x="Season",y="Count"),)
fig.update_layout(title="No. of tosses won by each team",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Decision Made After Winning the Toss

```
temp_series = match_data.toss_decision.value_counts()
labels = (np.array(temp_series.index))
values = (np.array((temp_series / temp_series.sum())*100))
colors = ['turquoise', 'crimson']
fig = go.Figure(data=[go.Pie(labels=labels,
                             values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='label+percent', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Toss decision percentage",
                  titlefont={'size': 30},
                  )
fig.show()
```



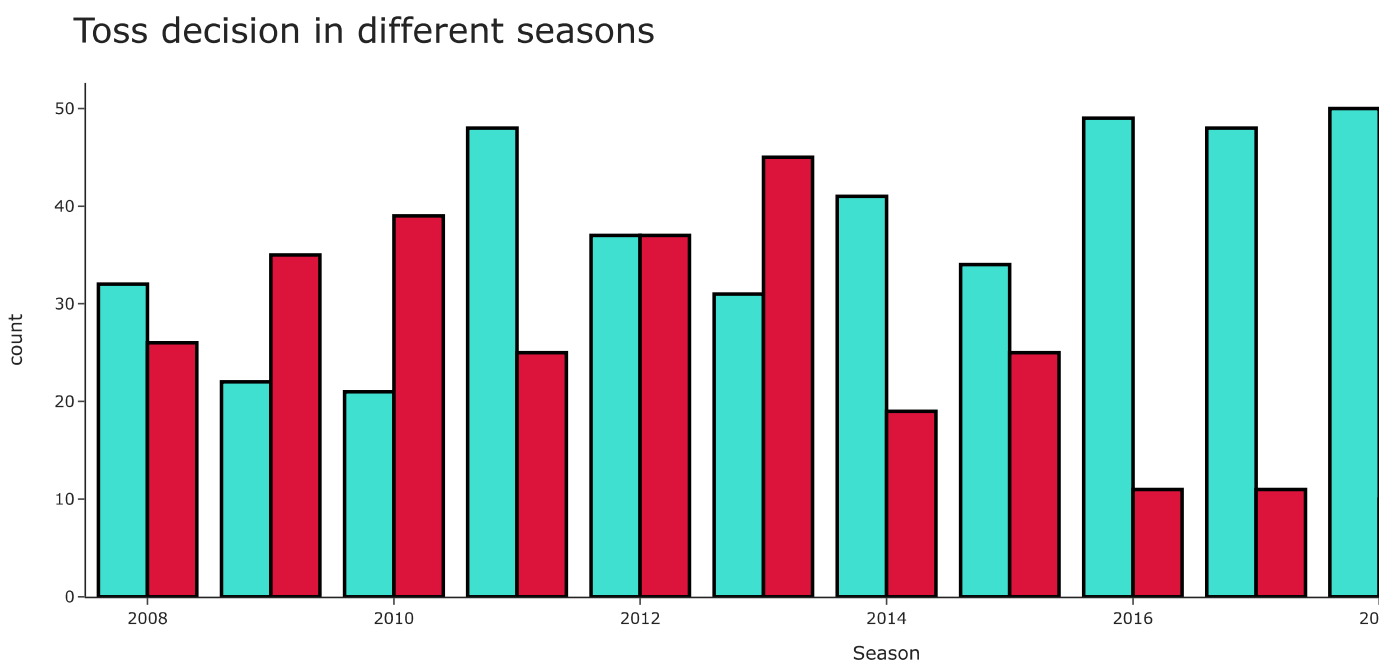
Note: After winning the toss, team tends to field first

Toss Decision Across Seasons

```
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decisio
0	335982	Bangalore	4/18/2008	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	fiel
1	335983	Chandigarh	4/19/2008	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	ba
2	335984	Delhi	4/19/2008	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	ba
3	335985	Mumbai	4/20/2008	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	ba
4	335986	Kolkata	4/20/2008	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	ba

```
fig=px.histogram(data_frame=match_data,x='Season',color='toss_decision',color_discrete_sequence=colors,barmode='group')
fig.update_layout(title="Toss decision in different seasons",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1)
fig.show()
```



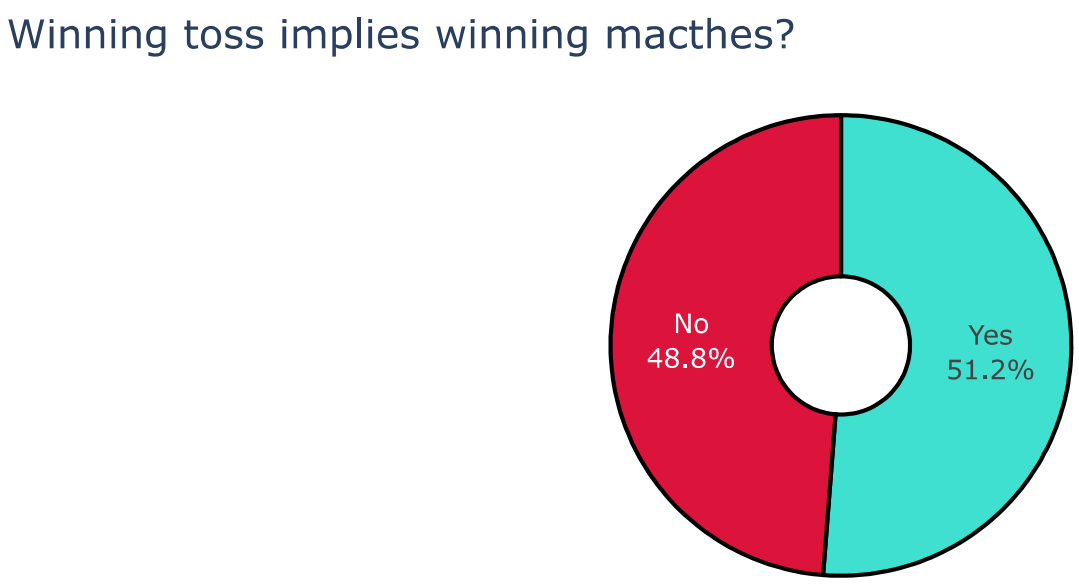
Note: Most of the times, teams decide to feild first except in season 2009,2010,2013 where teams decided to bat first mostly. Since 2014, teams have overwhelmingly chosen to bat second. Especially since 2016, teams have chosen to field for more than 80% of the times except in season 2020.

Winning Toss Implies Winning Game ?

```
match_data['toss_win_game_win'] = np.where((match_data.toss_winner == match_data.winner),'Yes','No')
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winn
0	335982	Bangalore	4/18/2008	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolka Knig Ride
1	335983	Chandigarh	4/19/2008	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chenr Sup Kin
2	335984	Delhi	4/19/2008	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	De Daredev
3	335985	Mumbai	4/20/2008	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Roy Challenge Bangalc
4	335986	Kolkata	4/20/2008	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolka Knig Ride

```
labels =["Yes",'No']
values = match_data['toss_win_game_win'].value_counts()
colors = ['turquoise', 'crimson']
fig = go.Figure(data=[go.Pie(labels=labels,
                             values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='label+percent', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Winning toss implies winning macthes?",
                  titlefont={'size': 30},
                  )
fig.show()
```



Note: Though winning toss gives you an advantage but it doesn't significantly implies that winning the toss helps in winning the game.

Match Win Result

```
match_data['result'].value_counts()
```

```
wickets    435
runs       364
tie         13
Name: result, dtype: int64
```

Note: We can see that 435 out of 816 matches was won by team batting second while 364 matches was won by team batting first.

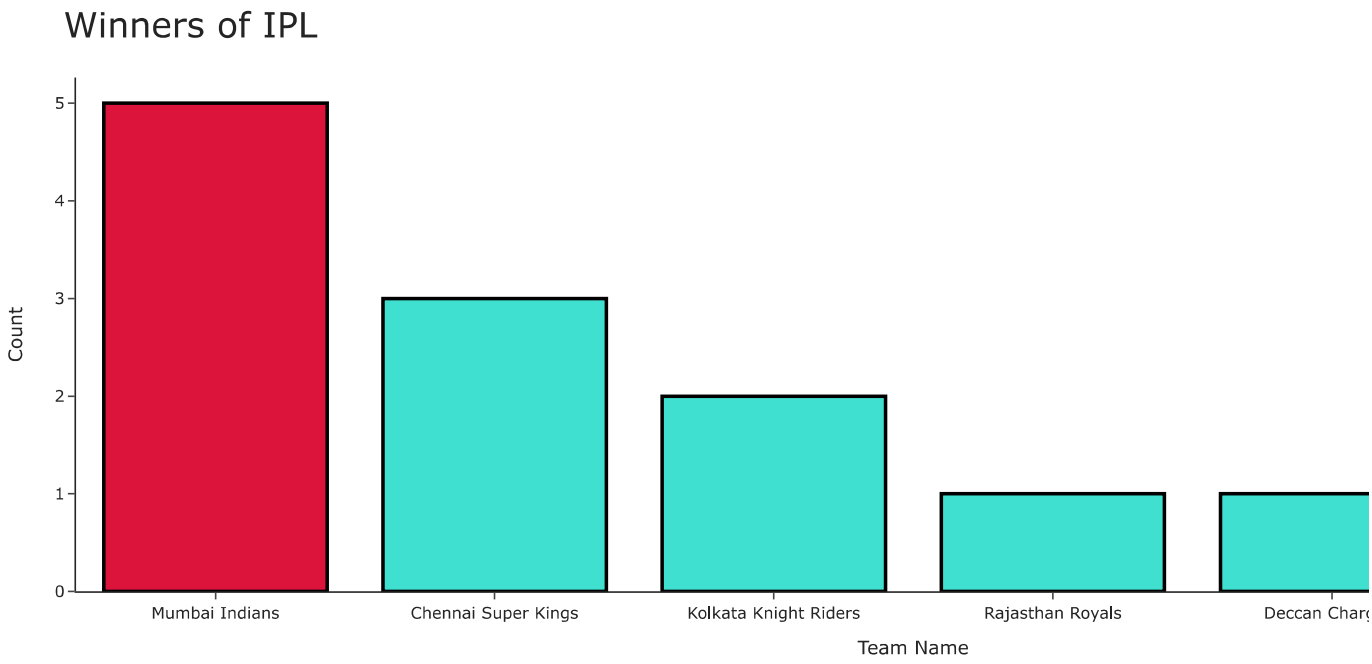
Number of times team have won the tournament

```
winning_teams = match_data[['Season','winner']]
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'])
```

```
winners_of_IPL.value_counts().index
```

```
MultiIndex([(       'Mumbai Indians',),
            (  'Chennai Super Kings',),
            ('Kolkata Knight Riders',),
            (      'Deccan Chargers',),
            (     'Rajasthan Royals',),
            (  'Sunrisers Hyderabad',)],
           names=['team'])
```

```
colors = ['turquoise',] * 6
colors[0] = 'crimson'
fig=px.bar( y=winners_of_IPL['team'].value_counts(),x=winners_of_IPL['team'].value_counts().index,labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Winners of IPL",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Total Number of Matches Played By A Team

```
matches_played_byteams=pd.concat([match_data['team1'],match_data['team2']],axis=1)
teams=(matches_played_byteams['team1'].value_counts()+matches_played_byteams['team2'].value_counts()).reset_index()
teams.columns=['Team Name','Total Matches played']
teams.sort_values(by=['Total Matches played'],ascending=False).reset_index().drop('index',axis=1).style.background_gradient(cmap='PuBu')
```

	Team Name	Total Matches played
0	Mumbai Indians	203
1	Royal Challengers Bangalore	195
2	Kolkata Knight Riders	192
3	Kings XI Punjab	190
4	Chennai Super Kings	178
5	Delhi Daredevils	161
6	Rajasthan Royals	161
7	Sunrisers Hyderabad	124
8	Deccan Chargers	75
9	Pune Warriors	46
10	Delhi Capitals	33
11	Gujarat Lions	30

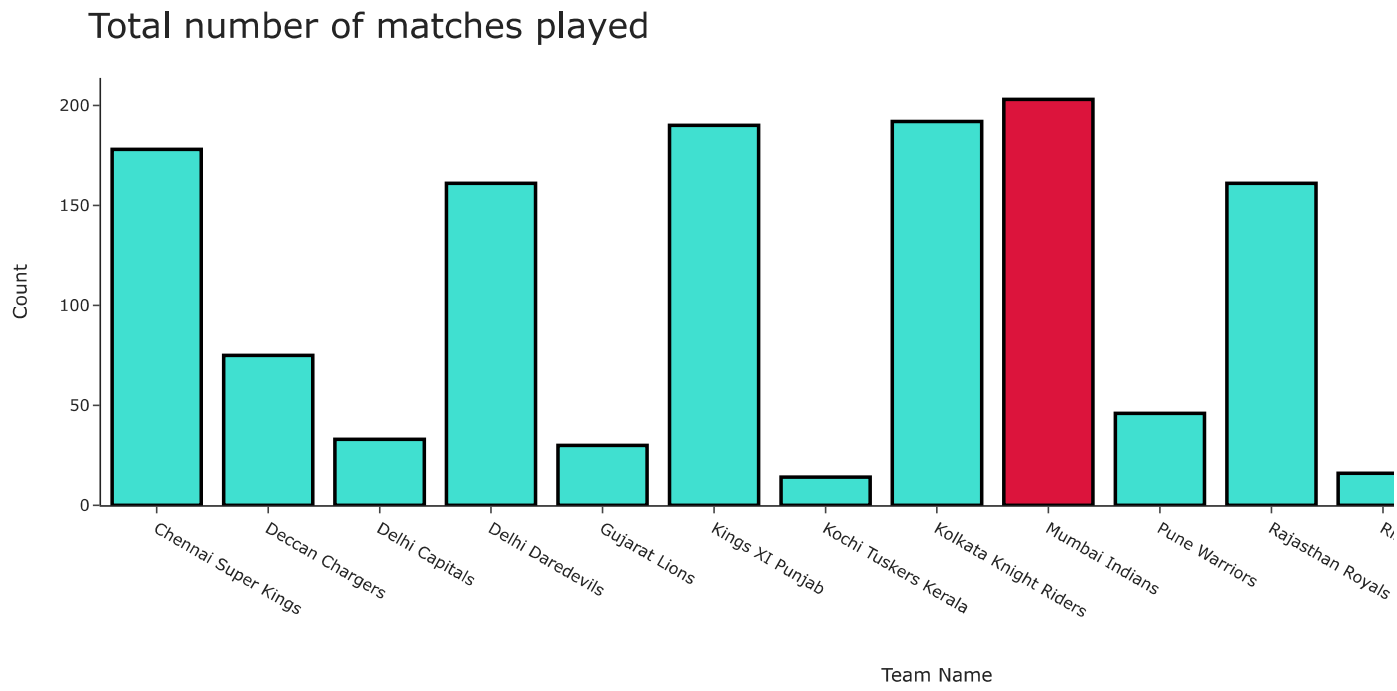
```
wins=pd.DataFrame(match_data['winner'].value_counts()).reset_index()
wins.columns=['Team Name','Wins']
wins.style.background_gradient(cmap='PuBu')
```

	Team Name	Wins
0	Mumbai Indians	120
1	Chennai Super Kings	106
2	Kolkata Knight Riders	99
3	Royal Challengers Bangalore	91
4	Kings XI Punjab	88
5	Rajasthan Royals	81
6	Delhi Daredevils	67
7	Sunrisers Hyderabad	66
8	Deccan Chargers	29
9	Delhi Capitals	19
10	Gujarat Lions	13
11	Pune Warriors	12
12	Rising Pune Supergiant	10
13	Kochi Tuskers Kerala	6
14	Rising Pune Supergiants	5

```
played=teams.merge(wins,left_on='Team Name',right_on='Team Name',how='inner')
played['% Win']=(played['Wins']/played['Total Matches played'])*100
played.sort_values(by=['% Win'],ascending=False).reset_index().drop('index',axis=1).style.background_gradient(cmap='PuBu',subset=['% Win'])
```

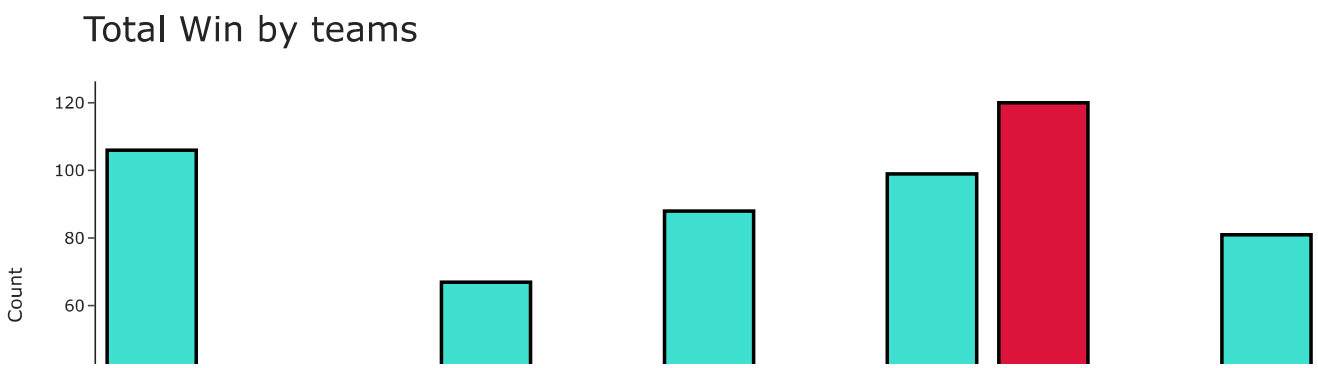
	Team Name	Total Matches played	Wins	% Win
0	Rising Pune Supergiant	16	10	62.500000
1	Chennai Super Kings	178	106	59.550562
2	Mumbai Indians	203	120	59.113300
3	Delhi Capitals	33	19	57.575758

```
colors = ['turquoise',] * 15
colors[8] = 'crimson'
fig=px.bar(x=played['Team Name'],y=played['Total Matches played'],labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Total number of matches played",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



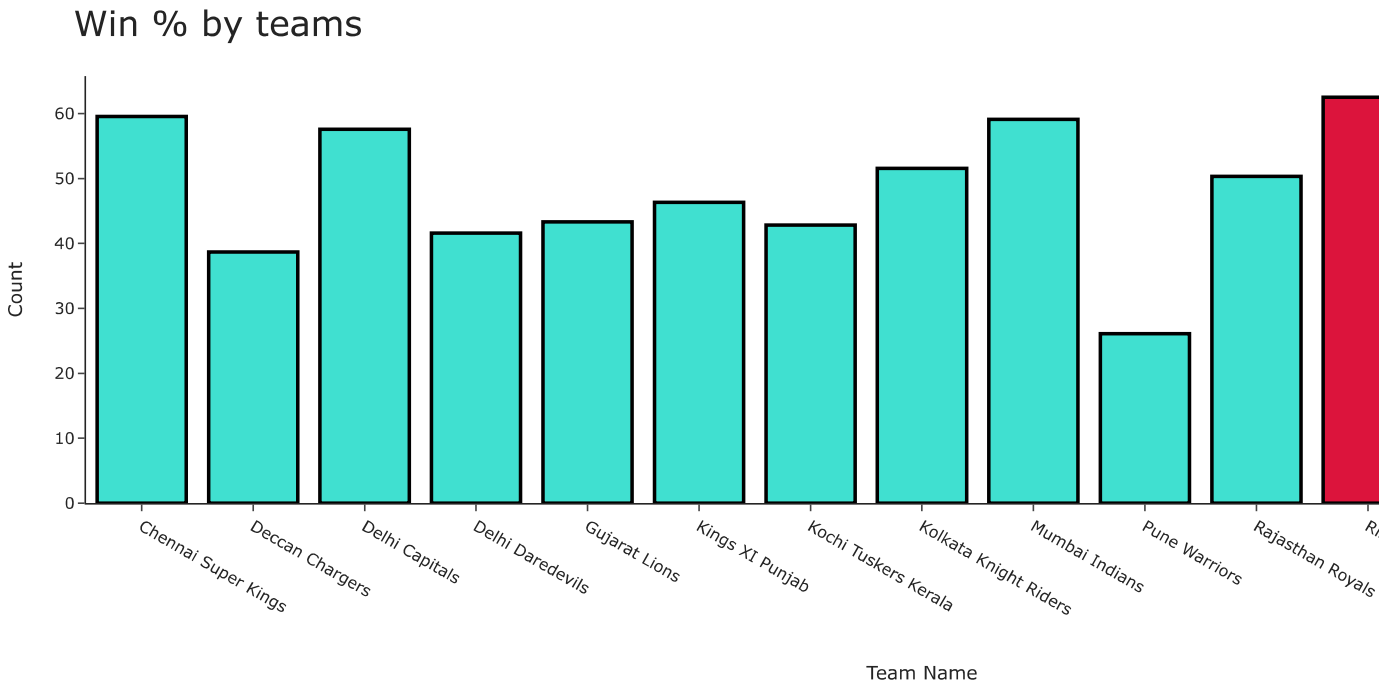
Most Number of Wins

```
colors = ['turquoise',] * 15
colors[8] = 'crimson'
fig=px.bar(x=played['Team Name'],y=played['Wins'],labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Total Win by teams",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Win % by Teams

```
colors = ['turquoise',] * 15
colors[-4] = 'crimson'
fig=px.bar(x=played['Team Name'],y=played['% Win'],labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Win % by teams",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: Rising Pune Supergiants have the highest win % of 62.50, followed by Chennai Super kings and Mumbai Indians. This is largely due to the fact that they had played really few matches.

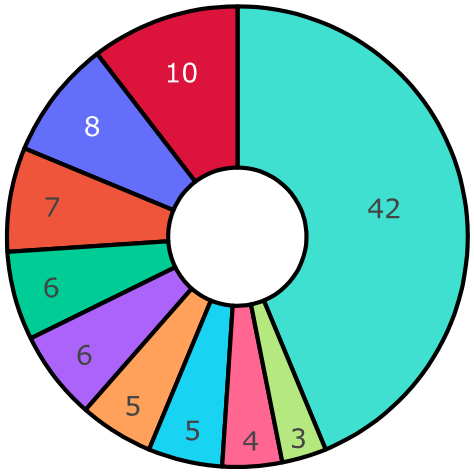
Lucky Venues For a Team

```
def lucky(match_data,team_name):
    return match_data[match_data['winner']==team_name]['venue'].value_counts().nlargest(10)


mi=lucky(match_data,'Mumbai Indians')
values = mi
labels=mi.index
colors = ['turquoise', 'crimson']
fig = go.Figure(data=[go.Pie(labels=labels,values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
```

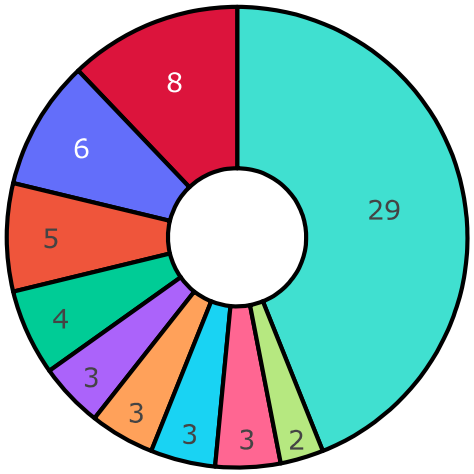
```
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Wins at different Venues for MI:",
                  titlefont={'size': 30},
                  )
fig.show()
```

Wins at different Venues for MI:

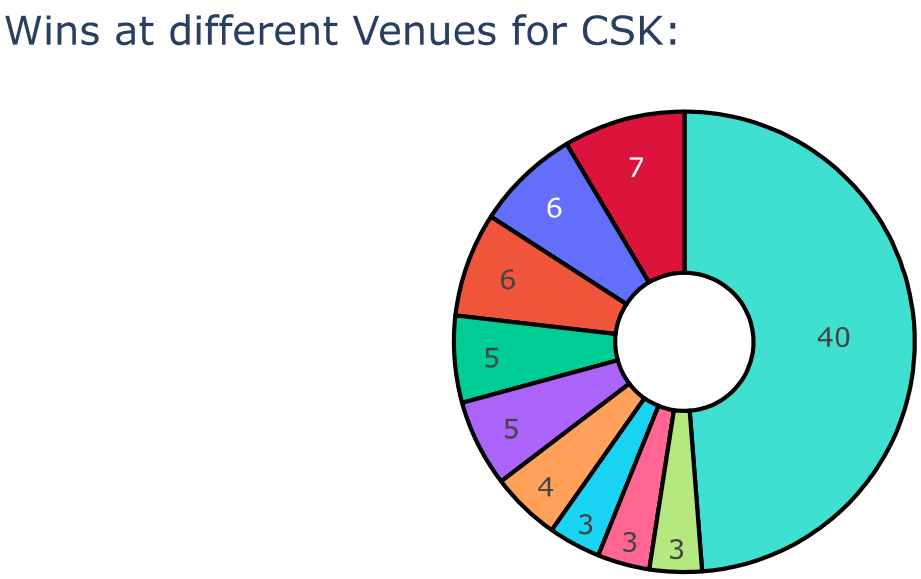



```
rcb=lucky(match_data,'Royal Challengers Bangalore')
values = rcb
labels=rcb.index
colors = ['turquoise', 'crimson']
fig = go.Figure(data=[go.Pie(labels=labels,values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Wins at different Venues for RCB:",
                  titlefont={'size': 30},
                  )
fig.show()
```

Wins at different Venues for RCB:



```
csk=lucky(match_data,'Chennai Super Kings')
values = csk
labels=csk.index
colors = ['turquoise', 'crimson']
fig = go.Figure(data=[go.Pie(labels=labels,values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Wins at different Venues for CSK:",
                  titlefont={'size': 30},
                  )
fig.show()
```



Note: It can be easily seen that team have won the most of its matches at their home venues.

Comparision Between Two Teams

```
def comparison(team1,team2):
    compare=match_data[((match_data['team1']==team1)|(match_data['team2']==team1))&((match_data['team1']==team2)|(match_data['team2']==team2)
    fig=px.histogram(data_frame=compare,x='Season',color='winner',labels=dict(x="Team Name",y="Count"),barmode='group',nbins=16,color_discret
    fig.update_layout(title="Team Comparision:",
                      titlefont={'size': 26},template='simple_white'
                      )
    fig.update_traces(marker_line_color='black',
                      marker_line_width=2.5, opacity=1)
    fig.show()


comparison('Mumbai Indians','Chennai Super Kings')
```

Team Comparision:

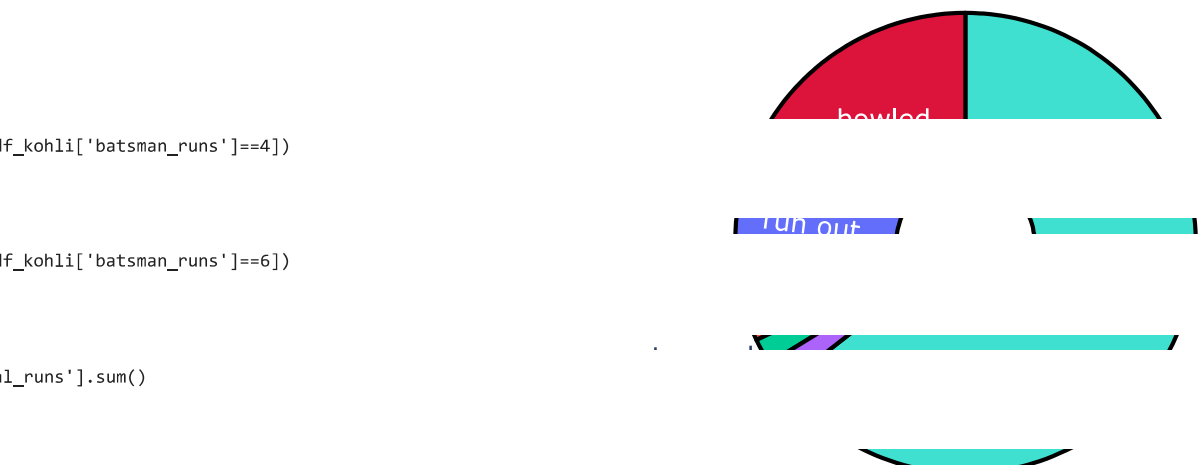
Note: Mumbai Indians is slight ahead of Chennai Super Kings in this heavy clash. The dominance was especially seen in the 2019 season, where Mumbai defeated Chennai 4 out of 4 times they met, including the playoff and final.

Particular Batsman Analysis

```
filt=(deliveries_data['batsman']=='V Kohli')
df_kohli=deliveries_data[filt]
df_kohli.head()
```

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_
211	335982	2	1	2	V Kohli	W Jaffer	I Sharma	0	0	0	0	0	
212	335982	2	1	3	V Kohli	W Jaffer	I Sharma	0	4	4	0	0	
213	335982	2	1	4	V Kohli	W Jaffer	I Sharma	1	0	1	0	0	
216	335982	2	2	1	V Kohli	W Jaffer	AB Dinda	0	0	0	0	0	
217	335982	2	2	2	V Kohli	W Jaffer	AB Dinda	0	0	0	0	1	bc

```
values = df_kohli['dismissal_kind'].value_counts()
labels=df_kohli['dismissal_kind'].value_counts().index
fig = go.Figure(data=[go.Pie(labels=labels,values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='label', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Dismissal Type",
                  titlefont={'size': 30},
                  )
fig.show()
```




```
len(df_kohli[df_kohli['batsman_runs']==4])
```

```
504
```

```
len(df_kohli[df_kohli['batsman_runs']==6])
```

```
202
```

```
df_kohli['total_runs'].sum()
```

```
6081
```

```
def count(df_kohli,runs):
    return len(df_kohli[df_kohli['batsman_runs']==runs])*runs


print("Runs scored from 1's :",count(df_kohli,1))
print("Runs scored from 2's :",count(df_kohli,2))
print("Runs scored from 3's :",count(df_kohli,3))
print("Runs scored from 4's :",count(df_kohli,4))
print("Runs scored from 6's :",count(df_kohli,6))
```

```
Runs scored from 1's : 1919
Runs scored from 2's : 692
Runs scored from 3's : 39
Runs scored from 4's : 2016
Runs scored from 6's : 1212
```

```
values=[1919,692,39,2016,1212]
labels=[1,2,3,4,6]
fig = go.Figure(data=[go.Pie(labels=labels,values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='label+value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=3)))
fig.update_layout(title="Virat Kohli total runs contribution",
                  titlefont={'size': 30},
                  )
fig.show()
```

Innings Wise Comparision

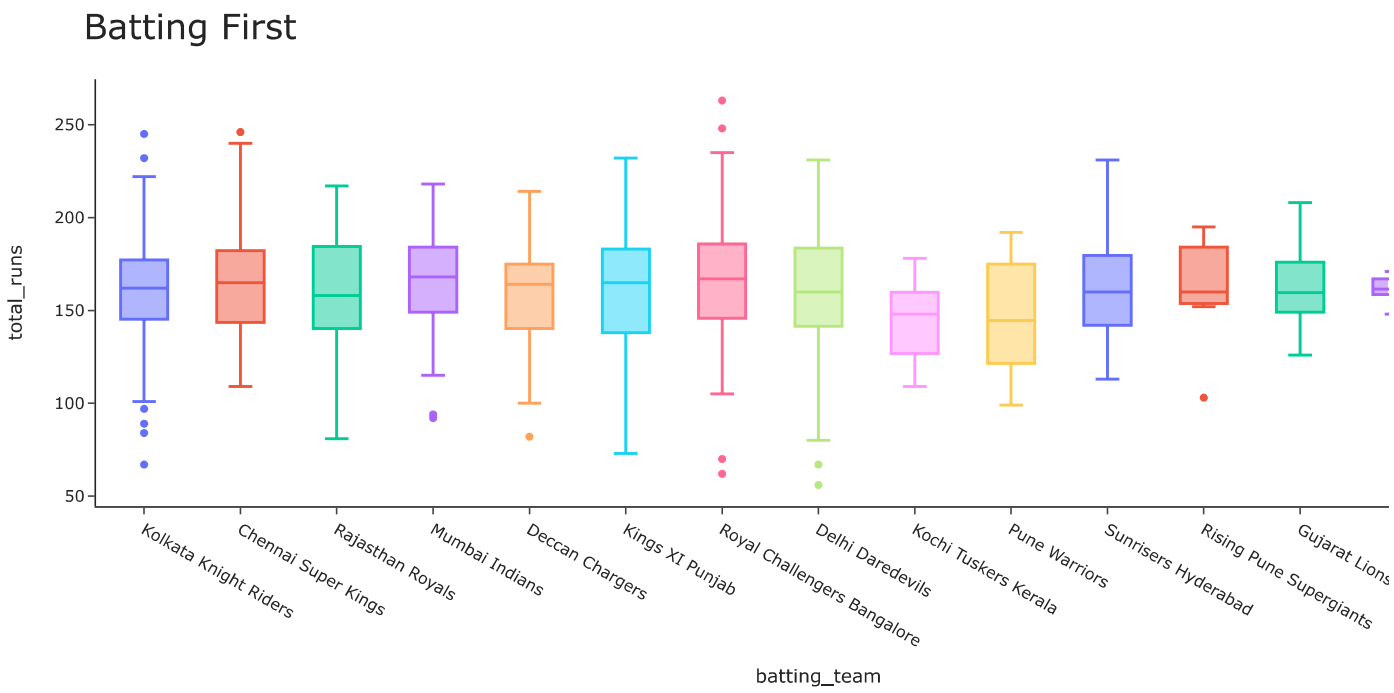
```
runs=deliveries_data.groupby(['id','inning','batting_team'])[['total_runs']].sum().reset_index()
runs.drop('id',axis=1,inplace=True)
runs.head()
```

	inning	batting_team	total_runs
0	1	Kolkata Knight Riders	222
1	2	Royal Challengers Bangalore	82
2	1	Chennai Super Kings	240
3	2	Kings XI Punjab	207
4	1	Rajasthan Royals	129

```
inning1=runs[runs['inning']==1]
inning2=runs[runs['inning']==2]
```

Batting First

```
fig = px.box(y='total_runs',x='batting_team',data_frame=inning1,color='batting_team')
fig.update_layout(title="Batting First",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.show()
```

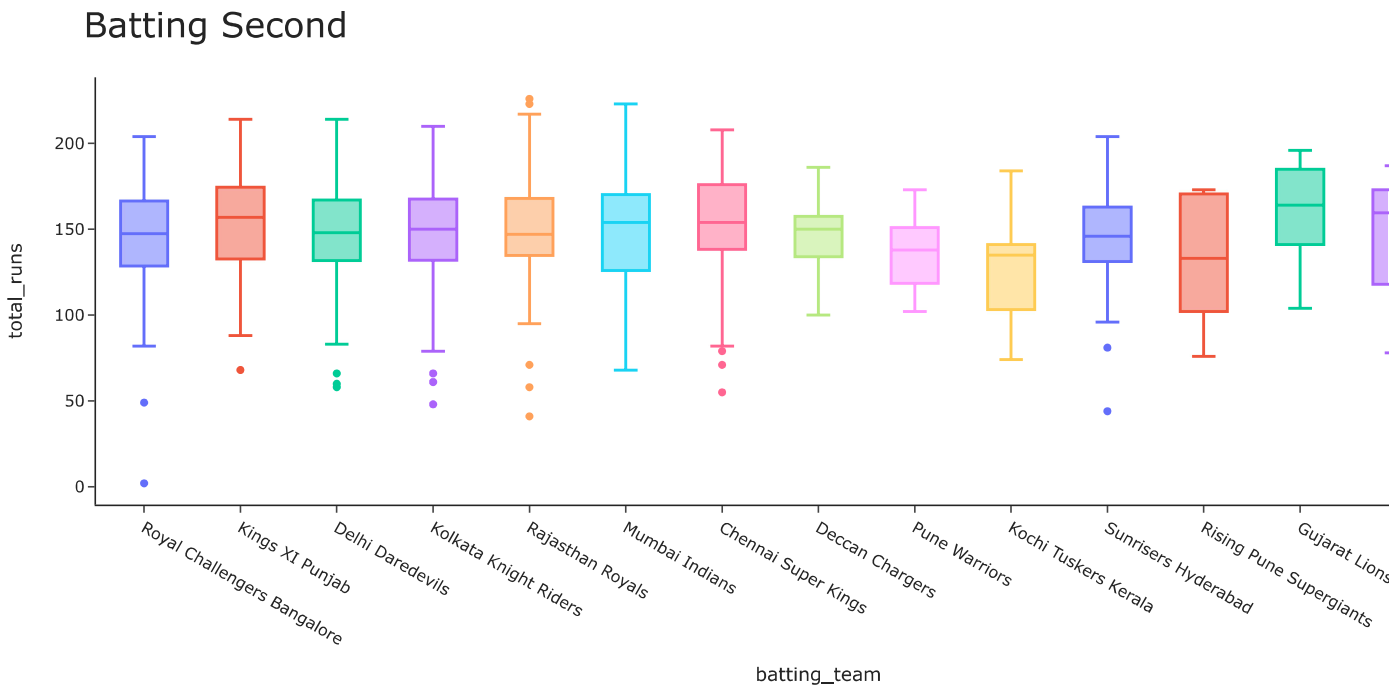


Note: Royal Challengers Bangalore and Mumbai Indians median value is better than other teams while batting first. Royal Challengers Bangalore had scored 250+ in a single match and is the only team to achieve that feat.

Batting Second

```
fig = px.box(y='total_runs',x='batting_team',data_frame=inning2,color='batting_team')

fig.update_layout(title="Batting Second",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.show()
```



Scored 200+ runs

```
high_scores=deliveries_data.groupby(['id', 'inning','batting_team','bowling_team'])['total_runs'].sum().reset_index()
score_200=high_scores[high_scores['total_runs']>=200]
score_200.head()
```

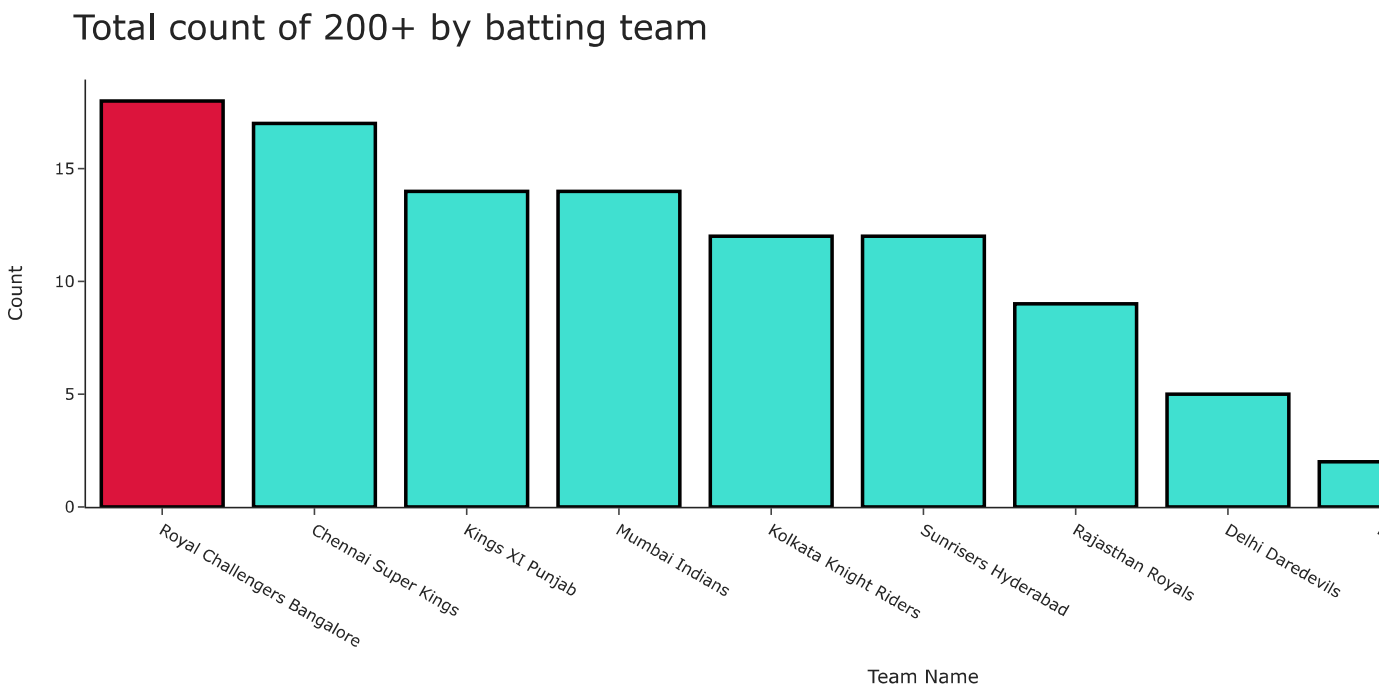
	id	inning	batting_team	bowling_team	total_runs
0	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	222
2	335983	1	Chennai Super Kings	Kings XI Punjab	240
3	335983	2	Kings XI Punjab	Chennai Super Kings	207
14	335989	1	Chennai Super Kings	Mumbai Indians	208
15	335989	2	Mumbai Indians	Chennai Super Kings	202

```
x1=score_200['batting_team'].value_counts()
x1=pd.DataFrame(x1)
x1.style.background_gradient(cmap='PuBu')
```

	batting_team
Royal Challengers Bangalore	18
Chennai Super Kings	17
Kings XI Punjab	14
Mumbai Indians	14
Kolkata Knight Riders	12
Sunrisers Hyderabad	12
Rajasthan Royals	9
Delhi Daredevils	5
Delhi Capitals	2
Deccan Chargers	1
Gujarat Lions	1

```
colors = ['turquoise',] * 11
colors[0] = 'crimson'
```

```
fig=px.bar(x=x1.index,y=x1['batting_team'],labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Total count of 200+ by batting team",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: Royal Challengers Bangalore had scored the most 200+ score (18 times), followed by Chennai Super Kings who had scored 17 times.

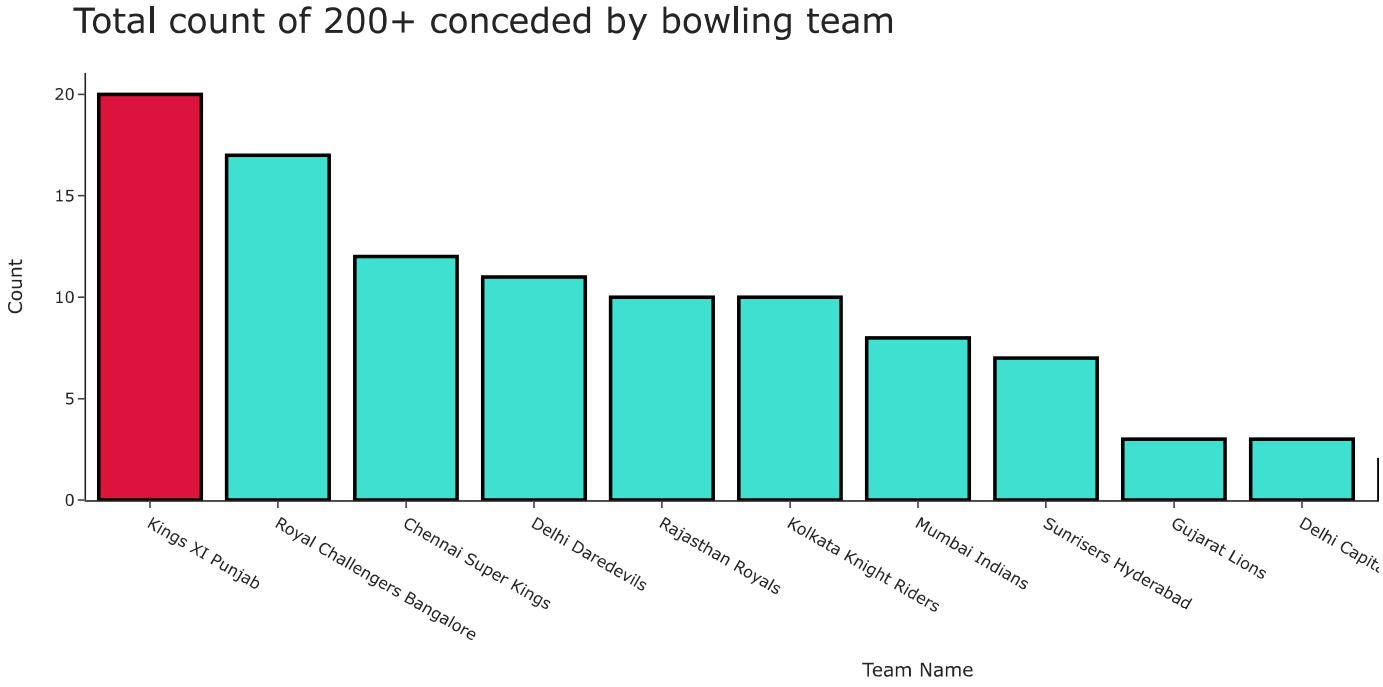
Conceded 200+ Runs

```
z=score_200['bowling_team'].value_counts()
z=pd.DataFrame(z)
z.style.background_gradient(cmap='PuBu')
```

	bowling_team
Kings XI Punjab	20
Royal Challengers Bangalore	17
Chennai Super Kings	12
Delhi Daredevils	11
Rajasthan Royals	10
Kolkata Knight Riders	10
Mumbai Indians	8
Sunrisers Hyderabad	7
Gujarat Lions	3
Delhi Capitals	3
Deccan Chargers	2
Pune Warriors	1
Rising Pune Supergiant	1

```
colors = ['turquoise',] * 13
colors[0] = 'crimson'
fig=px.bar(x=z.index,y=z['bowling_team'],labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Total count of 200+ conceded by bowling team",
```

```
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



*Highest Runs in an Innings *

```
high_200=deliveries_data.groupby(['id', 'inning','batting_team','bowling_team'])['total_runs'].sum().reset_index()
high_200.set_index(['id'],inplace=True)
```

```
high_200['total_runs'].max()
```

```
263
```

Note: In season 2013, Royal Challengers Bangalore scored 263/5 against Pune Warriors India.

Biggest Win in Terms of Run Margin

```
match_data[match_data['result_margin']==match_data['result_margin'].max()]
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner
620	1082635	Delhi	5/6/2017	LMP Simmons	Feroz Shah Kotla	0	Delhi Daredevils	Mumbai Indians	Delhi Daredevils	field	Mumbai Indians

Note: In season 2017, Mumbai Indians had defeated Delhi Daredevils by a huge margin of 146 runs.

Most Balls Played By a Batsman

```
balls_played=deliveries_data.groupby(['batsman'])['ball'].count().reset_index()
balls_played.sort_values(by='ball',ascending=False).head(10).style.background_gradient(cmap='PuBu')
```

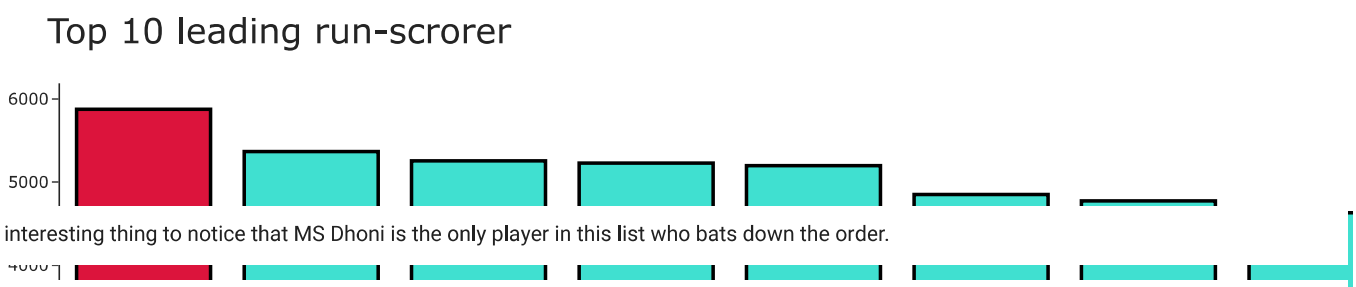
	batsman	ball
505	V Kohli	4609
407	S Dhawan	4208
379	RG Sharma	4088
438	SK Raina	4041
116	DA Warner	3819
398	RV Uthappa	3658
154	G Gambhir	3524
301	MS Dhoni	3493
96	CH Gayle	3342

Top 10 Run Scorer of All Time

```
runs=deliveries_data.groupby(['batsman'])['batsman_runs'].sum().reset_index()
runs.columns=['Batsman','runs']
y=runs.sort_values(by='runs',ascending=False).head(10).reset_index().drop('index',axis=1)
y.style.background_gradient(cmap='PuBu')
```

	Batsman	runs
0	V Kohli	5878
1	SK Raina	5368
2	DA Warner	5254
3	RG Sharma	5230
4	S Dhawan	5197
5	AB de Villiers	4849
6	CH Gayle	4772
7	MS Dhoni	4632
8	RV Uthappa	4607
9	G Gambhir	4217

```
colors = ['turquoise',] * 13
colors[0] = 'crimson'
fig=px.bar(x=y['Batsman'],y=y['runs'],labels=dict(x="Player",y="Total Runs"),)
fig.update_layout(title="Top 10 leading run-scrorer",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: One interesting thing to notice that MS Dhoni is the only player in this list who bats down the order.

Most Number of 4's

```
balls_played=balls_played.merge(runs,left_on='batsman',right_on='Batsman',how='outer')
four=deliveries_data[deliveries_data['batsman_runs']==4]
runs_4=four.groupby('batsman')['batsman_runs'].count().reset_index()
runs_4.columns=['Batsman','4s']
runs_4.sort_values(by='4s',ascending=False).head(10).reset_index().drop('index',axis=1).style.background_gradient(cmap='PuBu')
```

	Batsman	4s
0	S Dhawan	591
1	DA Warner	510
2	V Kohli	504
3	SK Raina	493
4	G Gambhir	492
5	RG Sharma	458
6	RV Uthappa	454
7	AM Rahane	416
8	AB de Villiers	390
9	CH Gayle	384

Most Number of 6's

```
six=deliveries_data.groupby('batsman')['batsman_runs'].agg(lambda x: (x==6).sum()).reset_index()
six.columns=['Batsman','6s']
six.sort_values(by='6s',ascending=False).head(10).reset_index().drop('index',axis=1).style.background_gradient(cmap='PuBu')
```

	Batsman	6s
0	CH Gayle	349
1	AB de Villiers	235
2	MS Dhoni	216
3	RG Sharma	214
4	V Kohli	202
5	KA Pollard	198
6	DA Warner	195
7	SK Raina	194
8	SR Watson	190
9	RV Uthappa	163

Highest Strike Rate (Minimum 100 Balls)

```
player=pd.concat([runs,balls_played.iloc[:,1],runs_4.iloc[:,1],six.iloc[:,1]],axis=1)
player['strike_rate']=player['runs']/player['ball']*100
```

```
player['4s'].fillna(0,inplace=True)
player.isnull().values.any()
```

```
False
```

```
player.sort_values(by='strike_rate',ascending=False).head(10)
```

	Batsman	runs	ball	4s	6s	strike_rate
72	B Stanlake	5	2	3.0	0	250.000000
504	Umar Gul	39	19	0.0	5	205.263158
395	RS Sodhi	4	2	1.0	0	200.000000
470	Shahid Afridi	81	46	0.0	6	176.086957
175	I Malhotra	7	4	13.0	0	175.000000
498	TU Deshpande	21	12	0.0	1	175.000000
33	AD Russell	1517	882	24.0	129	171.995465
253	LJ Wright	106	63	4.0	3	168.253968
57	Abdul Samad	111	66	28.0	6	168.181818
235	KMDN Kulasekara	5	3	156.0	0	166.666667

```
sr=player[player.ball > 100]
sr.sort_values(by='strike_rate',ascending=False).head(10)
```

	Batsman	runs	ball	4s	6s	strike_rate
33	AD Russell	1517	882	24.0	129	171.995465
217	K Gowtham	186	113	247.0	12	164.601770
80	BCJ Cutting	238	146	40.0	19	163.013699
317	N Pooran	521	323	6.0	39	161.300310
453	SP Narine	892	573	0.0	52	155.671902
293	MM Ali	309	199	23.0	23	155.276382
97	CH Morris	551	360	116.0	30	153.055556
192	JC Archer	195	128	18.0	14	152.343750
106	CR Brathwaite	181	120	11.0	16	150.833333
88	Bipul Sharma	187	124	5.0	9	150.806452

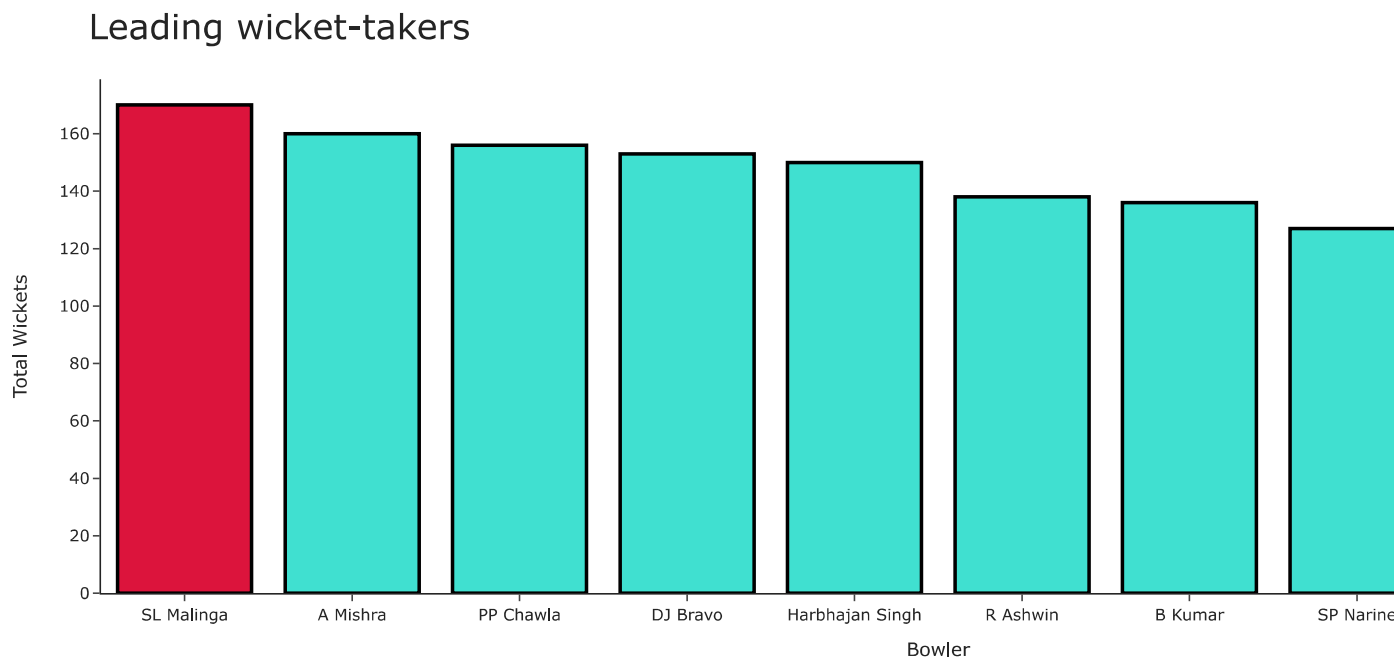
```
reqsr=sr.drop(columns=['runs','ball','4s','6s'],axis=1)
reqsr.sort_values(by='strike_rate',ascending=False).head(10).style.background_gradient(cmap='PuBu')
```

	Batsman	strike_rate
33	AD Russell	171.995465
217	K Gowtham	164.601770
80	BCJ Cutting	163.013699
317	N Pooran	161.300310
453	SP Narine	155.671902
293	MM Ali	155.276382
97	CH Morris	153.055556
192	JC Archer	152.343750
106	CR Brathwaite	150.833333
88	Bipul Sharma	150.806452

Highest Wicket-Taker

```
deliveries_data['dismissal_kind'].unique()
dismissal_kinds = ['caught', 'bowled', 'lbw', 'caught and bowled',
       'stumped', 'hit wicket']
hwt=deliveries_data[deliveries_data["dismissal_kind"].isin(dismissal_kinds)]
bo=hwt['bowler'].value_counts()
```

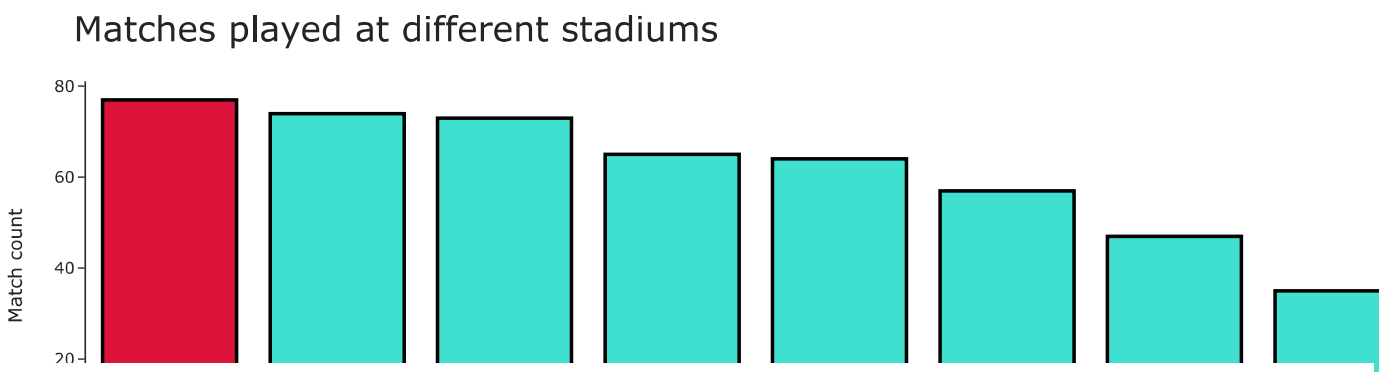
```
colors = ['turquoise',] * 13
colors[0] = 'crimson'
fig=px.bar(x=bo[:10].index,y=bo[:10],labels=dict(x="Bowler",y="Total Wickets"),)
fig.update_layout(title="Leading wicket-takers",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: 6 out of top 10 are spin bowlers.

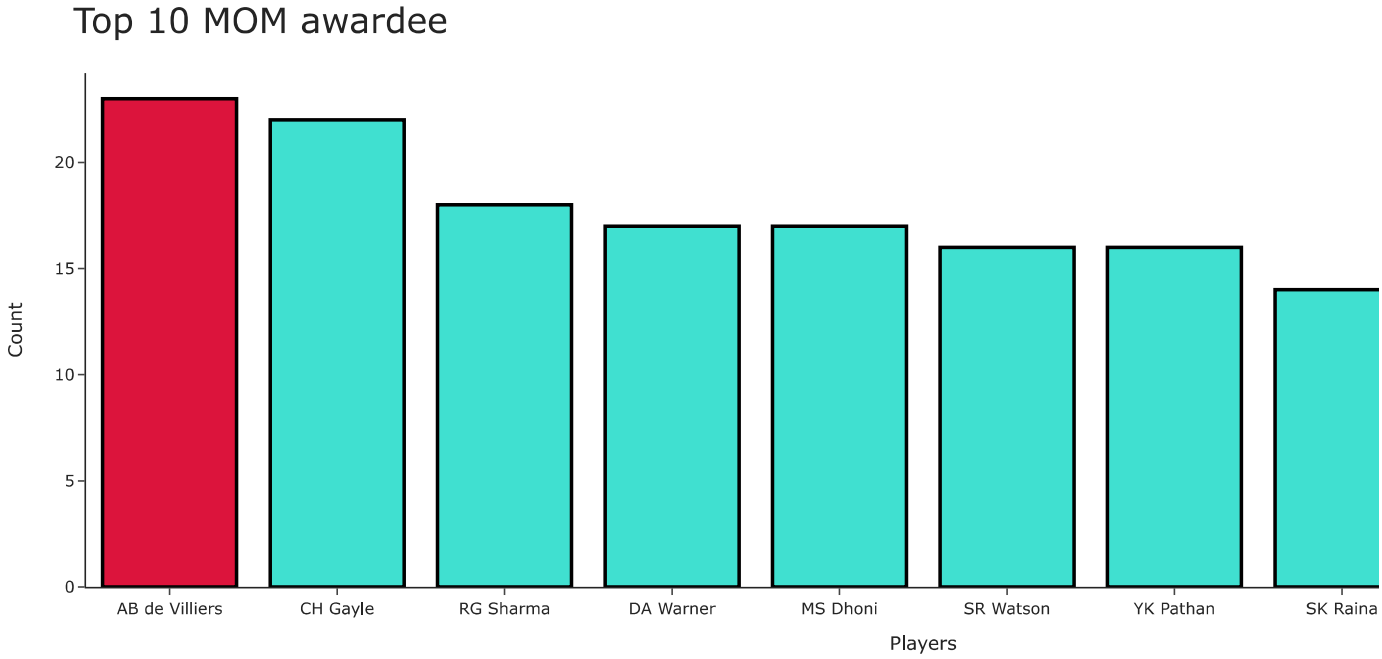
Total Count of Matches Played in Differnt Stadiums

```
colors = ['turquoise',] * 13
colors[0] = 'crimson'
fig=px.bar(x=match_data['venue'].value_counts()[:10].index,y=match_data['venue'].value_counts()[:10],labels=dict(x="Venue",y="Match count"),)
fig.update_layout(title="Matches played at different stadiums",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Man of the Match Award

```
colors = ['turquoise',] * 11
colors[0] = 'crimson'
fig=px.bar(x=match_data.player_of_match.value_counts()[:10].index,y=match_data.player_of_match.value_counts()[:10],labels=dict(x="Players",y=
fig.update_layout(title="Top 10 MOM awardee",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



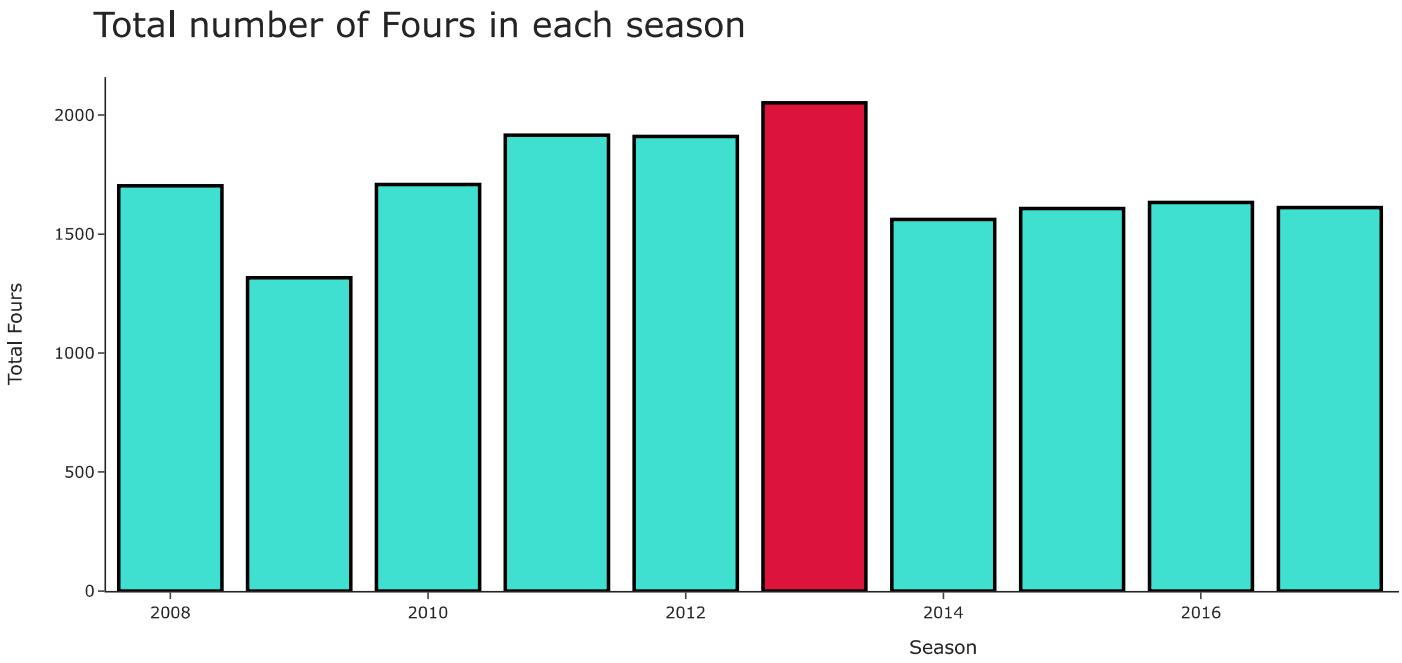
Total Number of Fours in Each Season

```
data_4 = match_data['Season'].unique()

fours_list = []
for var in data_4:
    new_df = match_data[match_data['Season']==var]
    total_fours = 0
    for i in new_df['id'].values:
        temp_df = deliveries_data[deliveries_data['id']==i]
        fours = temp_df[temp_df['batsman_runs']==4]['batsman_runs'].count()
        total_fours+=fours
    fours_list.append(total_fours)


colors = ['turquoise',] * 14
```

```
colors[5] = 'crimson'
fig=px.bar(x=data_4, y=fours_list,labels=dict(x="Season",y="Total Fours"),)
fig.update_layout(title="Total number of Fours in each season",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```

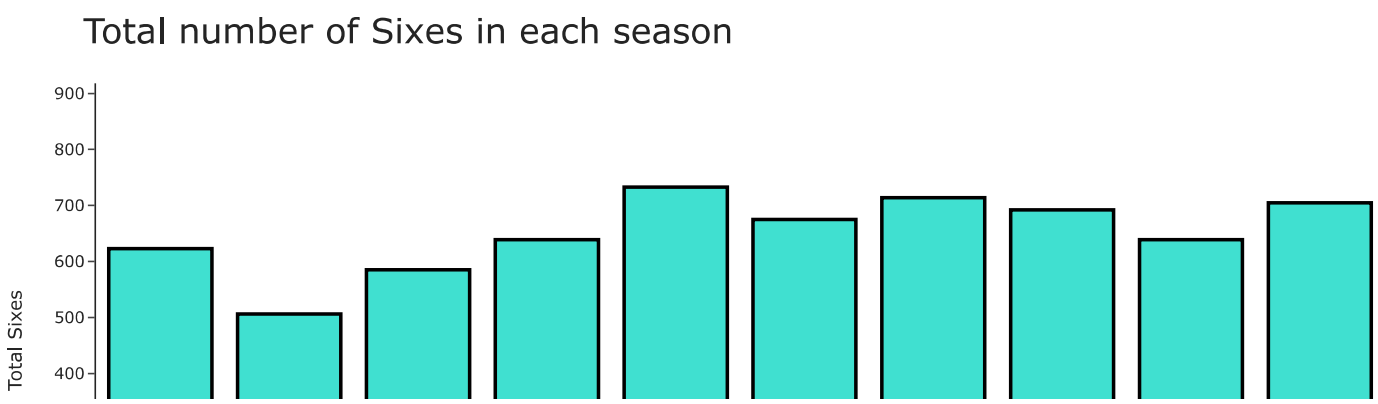


Total Number of Sixes in Each Season

```
data_6 = match_data['Season'].unique()

sixes_list = []
for var in data_6:
    new_df = match_data[match_data['Season']==var]
    total_sixes = 0
    for i in new_df['id'].values:
        temp_df = deliveries_data[deliveries_data['id']==i]
        sixes = temp_df[temp_df['batsman_runs']==6]['batsman_runs'].count()
        total_sixes+=sixes
    sixes_list.append(total_sixes)

colors = ['turquoise',] * 14
colors[-4] = 'crimson'
fig=px.bar(x=data_4, y=sixes_list,labels=dict(x="Season",y="Total Sixes"),)
fig.update_layout(title="Total number of Sixes in each season",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```

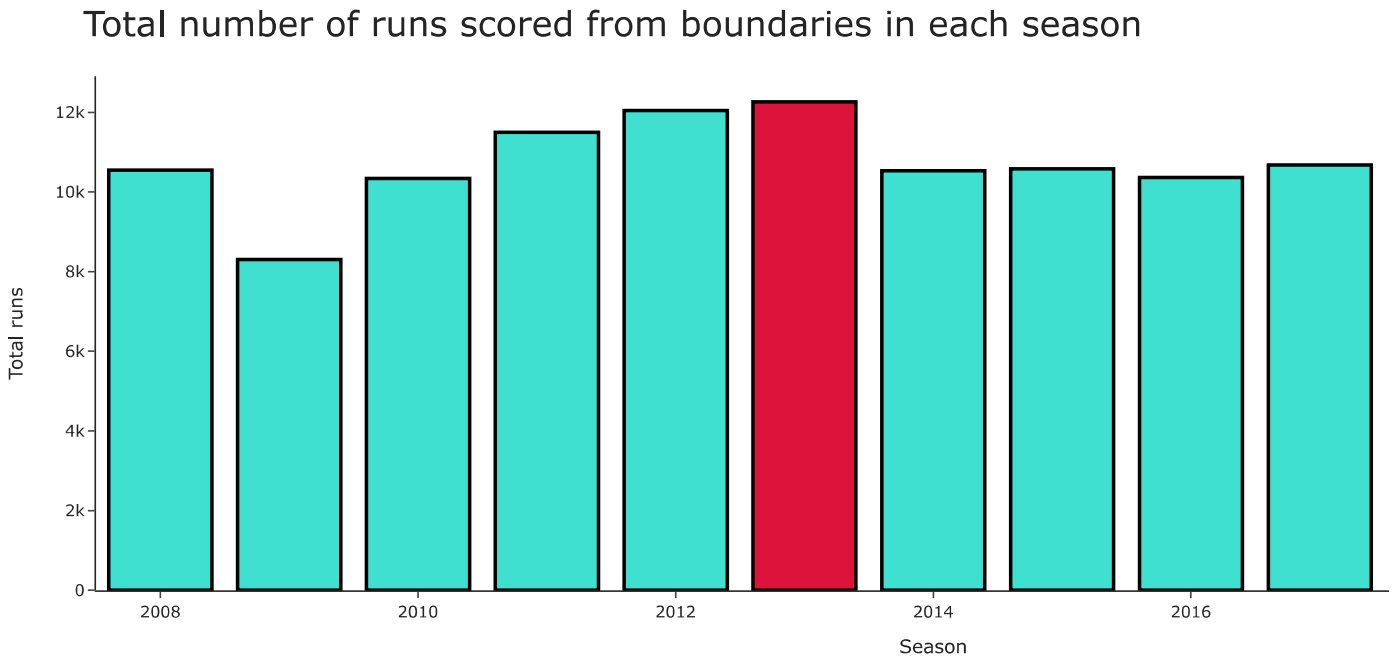


Note: In season 2018, the maximum number of sixes were hit while the lowest was observed in season 2009.

Total Runs Scored From Boundaries in Each Season

```
runs4=np.dot(fours_list,4)
runs6=np.dot(sixes_list,6)


k=runs4+runs6
Y=match_data['Season'].unique()
colors = ['turquoise',] * 14
colors[5] = 'crimson'
fig=px.bar(x=Y,y=k,labels=dict(x="Season",y="Total runs"),)
fig.update_layout(title="Total number of runs scored from boundaries in each season",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: Total run scored from boundaries is lowest in season 2009 and highest in season 2013.

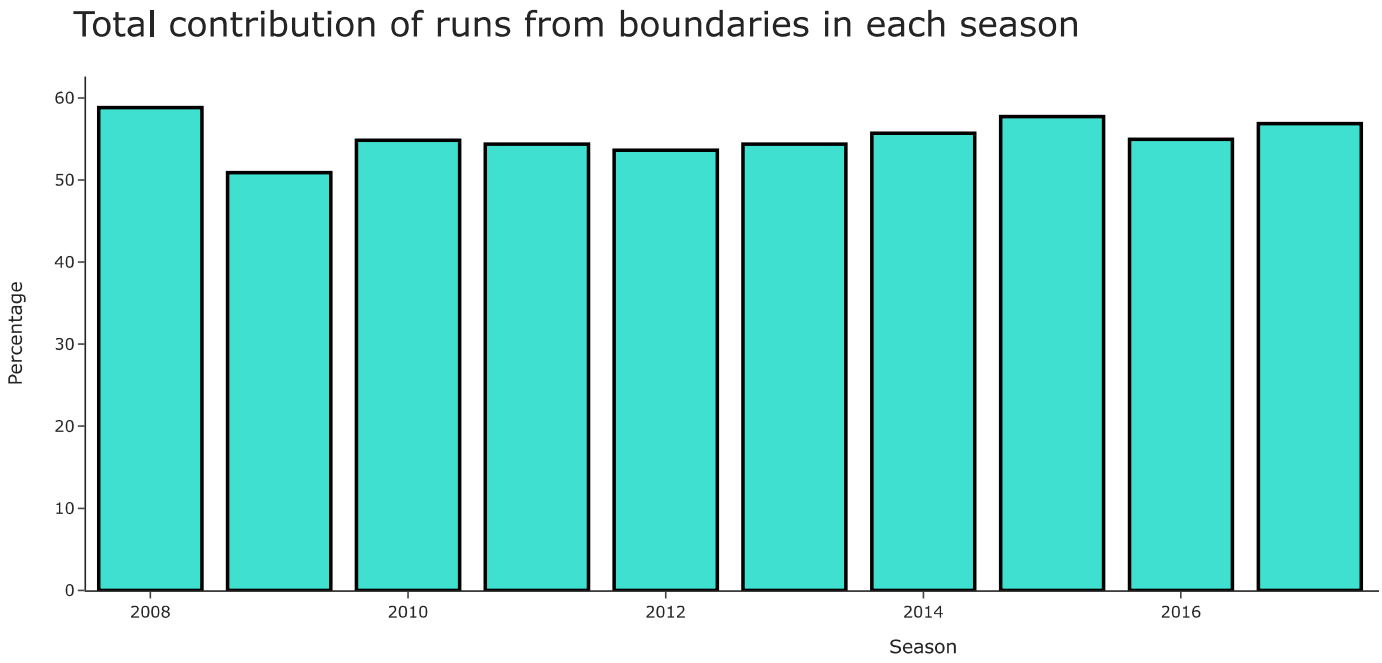
Total Contribution of Runs From Boundaries in Each Season

```
totruns=np.array(season['total_runs'])
res=(k/totruns)*100


res

     array([58.81697051, 50.88235294, 54.82400339, 54.35378652, 53.63203135,
            54.3809059 , 55.6983447 , 57.71328824, 54.95705652, 56.87037136,
            59.49449776, 58.32989691, 55.50847458])


colors = ['turquoise',] * 14
colors[10] = 'crimson'
fig=px.bar(x=Y,y=res,labels=dict(x="Season",y="Percentage"),)
fig.update_layout(title="Total contribution of runs from boundaries in each season",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



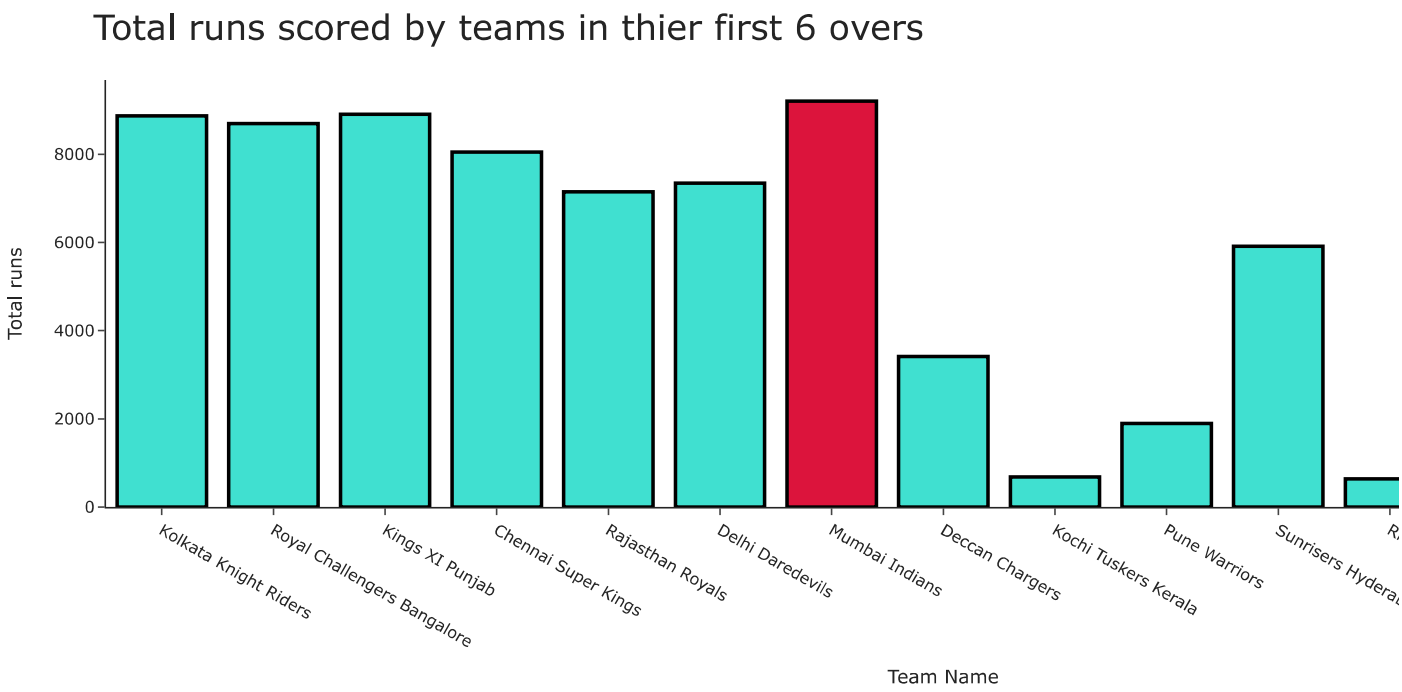
Note: In season 2018, 59.49 % runs of the total runs came from boundaries while 50.88 % runs came from boundaries in season 2009 which is lowest till now.

Total Runs Scored by Teams in First 6 Overs

```
team = deliveries_data['batting_team'].unique()
team_runs = []
for var in team:
    temp_df = deliveries_data[deliveries_data['batting_team']==var]
    temp_df = temp_df[temp_df['over'].isin([0,1,2,3,4,5])]
    runs = temp_df['total_runs'].sum()
    team_runs.append(runs)
team = pd.DataFrame(data=team_runs, index=team,columns=['Runs In First 6 Overs'])
#team.sort_values('Runs In First 6 Overs', ascending=False, inplace=True)
team.index.name = 'Team'

colors = ['turquoise',] * 15
colors[6] = 'crimson'
fig=px.bar(x=team.index,y=team['Runs In First 6 Overs'],labels=dict(x="Team Name",y="Total runs"),)
fig.update_layout(title="Total runs scored by teams in thier first 6 overs",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
```

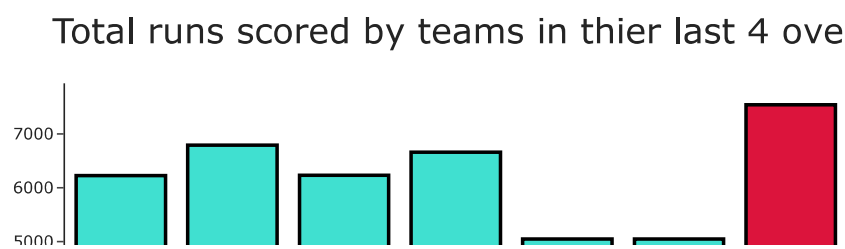
```
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: Mumbai Indians had scored the most runs in first 6 overs, followed by Kings XI Punjab.

Total Runs Scored by Teams in Last 4 Overs

```
team1 = deliveries_data['batting_team'].unique()
team_runs1 = []
for var in team1:
    temp_df = deliveries_data[deliveries_data['batting_team']==var]
    temp_df = temp_df[temp_df['over'].isin([19,18,17,16])]
    runs1 = temp_df['total_runs'].sum()
    team_runs1.append(runs1)
team1 = pd.DataFrame(data=team_runs1, index=team1,columns=['Runs In Last 4 Overs'])
team1.index.name = 'Team'
colors = ['turquoise',] * 15
colors[6] = 'crimson'
fig=px.bar(x=team1.index,y=team1['Runs In Last 4 Overs'],labels=dict(x="Team Name",y="Total runs"),)
fig.update_layout(title="Total runs scored by teams in thier last 4 overs",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```

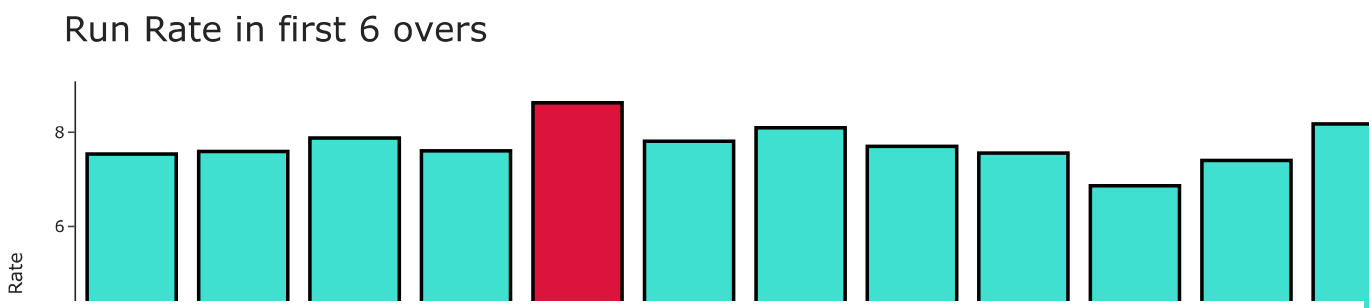


Highest Scoring Run-Rate in First 6 Overs


```
played1=played.merge(team,left_on='Team Name',right_on='Team',how='inner')
played3=played1.merge(team1,left_on='Team Name',right_on='Team',how='inner')
mintoover=np.dot(played3['Total Matches played'],6)
mintoover1=np.dot(played3['Total Matches played'],4)
played3['RR in first 6 overs']=(played3['Runs In First 6 Overs']/mintoover)
played3['RR in last 4 overs']=(played3['Runs In Last 4 Overs']/mintoover1)
played3
```

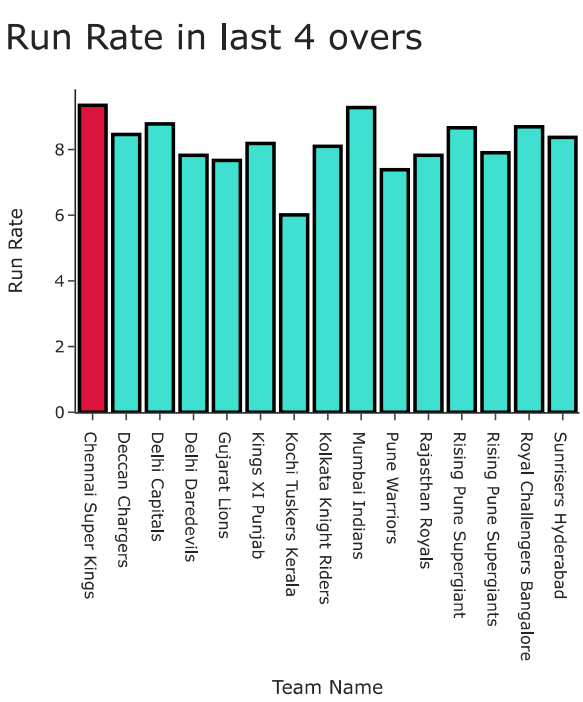
	Team Name	Total Matches played	Wins	% Win	Runs In First 6 Overs	Runs In Last 4 Overs	RR in first 6 overs	RR
0	Chennai Super Kings	178	106	59.550562	8048	6655	7.535581	
1	Deccan Chargers	75	29	38.666667	3417	2539	7.593333	
2	Delhi Capitals	33	19	57.575758	1560	1160	7.878788	
3	Delhi Daredevils	161	67	41.614907	7349	5043	7.607660	
4	Gujarat Lions	30	13	43.333333	1553	921	8.627778	
5	Kings XI Punjab	190	88	46.315789	8907	6227	7.813158	
6	Kochi Tuskers Kerala	14	6	42.857143	680	337	8.095238	
7	Kolkata Knight Riders	192	99	51.562500	8871	6224	7.700521	
8	Mumbai Indians	203	120	59.113300	9204	7538	7.556650	
9	Pune Warriors	46	12	26.086957	1895	1360	6.865942	
10	Rajasthan Royals	161	81	50.310559	7151	5043	7.402692	
11	Rising Pune Supergiant	16	10	62.500000	785	555	8.177083	
12	Rising Pune Supergiants	14	5	35.714286	638	443	7.595238	
13	Royal Challengers Bangalore	195	91	46.666667	8699	6787	7.435043	
14	Sunrisers Hyderabad	124	66	53.225806	5917	4155	7.952957	

```
colors = ['turquoise',] * 15
colors[4] = 'crimson'
fig=px.bar(x=played3['Team Name'],y=played3['RR in first 6 overs'],labels=dict(x="Team Name",y="Run Rate"),)
fig.update_layout(title="Run Rate in first 6 overs",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Highest Scoring Rate in Last 4 Overs

```
colors = ['turquoise',] * 15
colors[0] = 'crimson'
fig=px.bar(x=played3['Team Name'],y=played3['RR in last 4 overs'],labels=dict(x="Team Name",y="Run Rate"),)
fig.update_layout(title="Run Rate in last 4 overs",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',
                  marker_line_width=2.5, opacity=1,marker_color=colors)
fig.show()
```



Note: Chennai Super Kings has the highest scoring run-rate in last 4 overs, followed by Mumbai Indians.

Conclusion

The Indian Premier League (IPL) has a significant influence on the cricket globe and beyond, according to studies. As a result of its ability to blend entertainment value with top-notch cricket skill, the IPL has developed into a global sports phenomenon that draws spectators from all over the world. In addition to showcasing exceptional cricketing abilities, the league has drawn large financial contributions from sponsors and broadcasters and grown to become a major economic force. It has cultivated young talent, produced cricket legends, and changed the T20 cricket scene. The IPL's cultural and sociological relevance is demonstrated by its capacity to captivate spectators and cross cultural divides. Furthermore, the league's focus on technology and data analytics has given cricket analysts a fresh perspective. Even though the Indian Premier League (IPL) has encountered difficulties including match-fixing scandals and schedule conflicts, its adaptability and fortitude have helped it to continue being one of the world's top cricket leagues. In summary, the IPL is still influencing cricket's future and has left a lasting impression on the game, its participants, and the millions of ardent fans who look forward to each new season."