```
!pip install plotly --user
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.15.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly) (23.2)
```

Importing libraries

```
import pandas as pd # data processing
import numpy as np # linear algebra
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 Data set

```
ball_data = pd.read_csv("IPL Ball-by-Ball 2008-2020.csv")
match_data = pd.read_csv("IPL Matches 2008-2020.csv")
print("Data ready for exploration")
```

```
Data ready for exploration
```

```
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils
3	335985	Mumbai	2008-	MV Boucher	Wankhede	0	Mumbai	Royal Challengers	Mumbai	bat	Royal Challengers

```
ball_data.head()
```

	id	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismissal_ki
0	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	N
1	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	N

```
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])
print(' \n Venues Played At:',match_data['city'].unique())
print(' \n Teams :',match_data['team1'].unique())
```

```
Total Matches Played: 816

 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']

 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']
```

1) Number of matches played in various seasons :

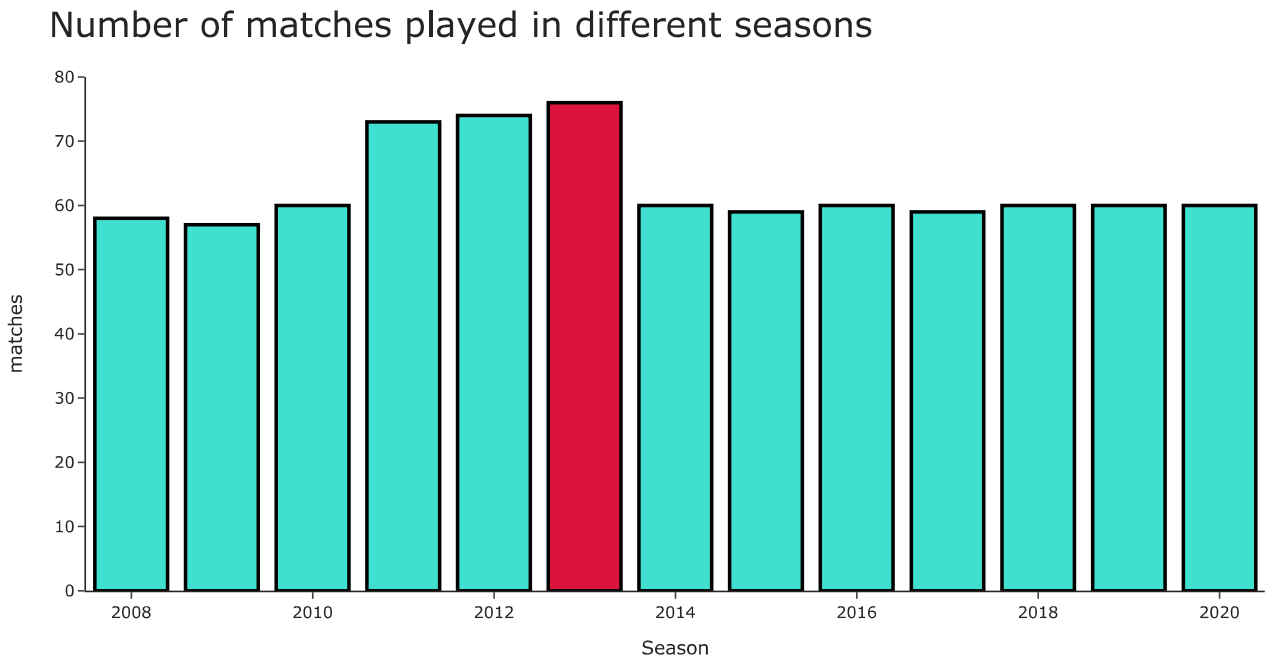
```
match_data['Season'] = pd.DatetimeIndex(match_data['date']).year
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore

```
import pandas as pd
match_data['Season'] = pd.to_datetime(match_data['date']).dt.year

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

	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()
```



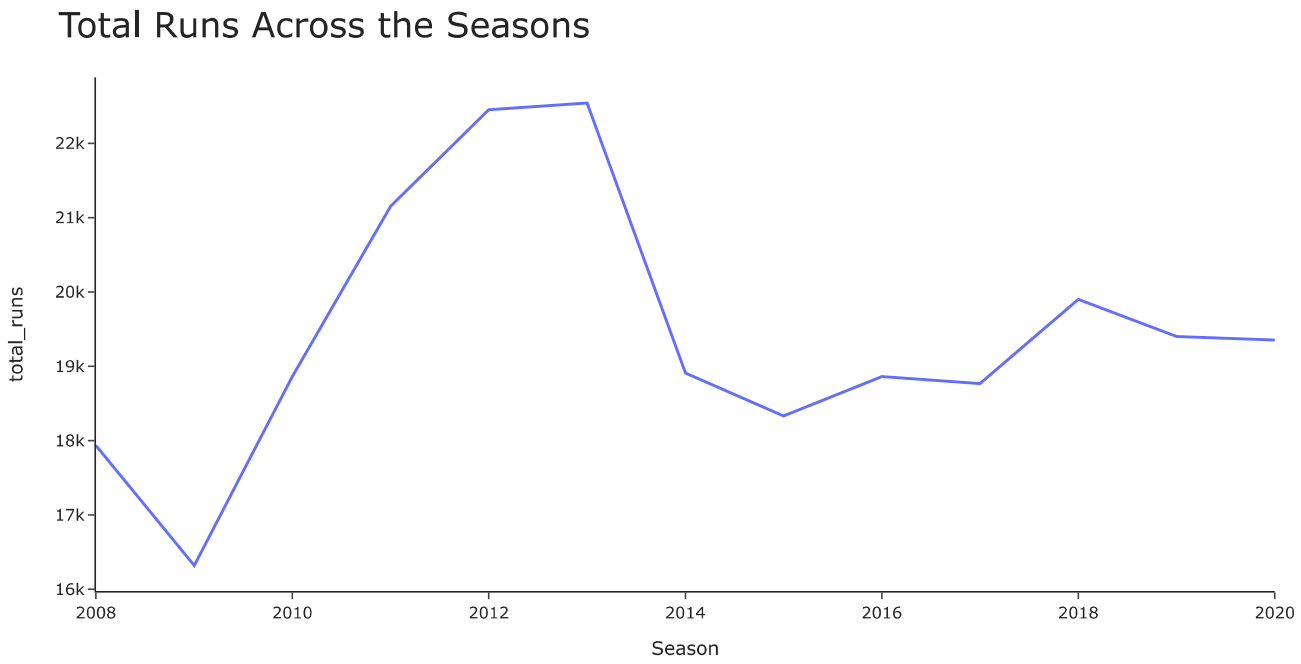
Each season, almost 60 matches were played. However, we see a spike in the number of matches from 2011 to 2013. This is because two new franchises, the Pune Warriors and Kochi Tuskers Kerala, were introduced, increasing the number of teams to 10.

2) Total number of runs scored across seasons

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

	date	Season	inning	over	ball	batsman	non_striker	bowler	batsman_runs	extra_runs	total_runs	non_boundary	is_wicket	dismi
0	2008-04-18	2008	1	6	5	RT Ponting	BB McCullum	AA Noffke	1	0	1	0	0	
1	2008-04-18	2008	1	6	6	BB McCullum	RT Ponting	AA Noffke	1	0	1	0	0	
2	2008-04-18	2008	1	7	1	BB McCullum	RT Ponting	Z Khan	0	0	0	0	0	
3	2008-	2008	1	7	2	BB	RT Ponting	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()
```



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).

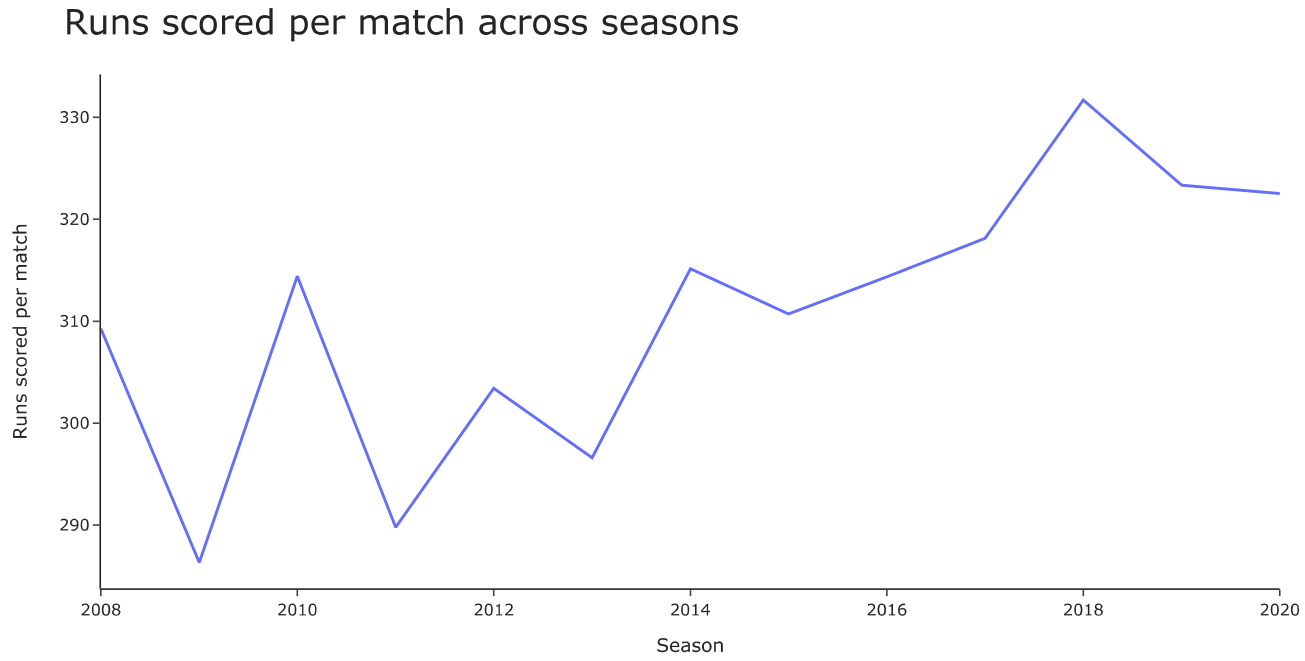
Runs scored per match across seasons

```
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)
print(runs_per_season)
```

```
        matches  total_runs  Runs scored per match
Season
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()
```



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.

4) 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
```

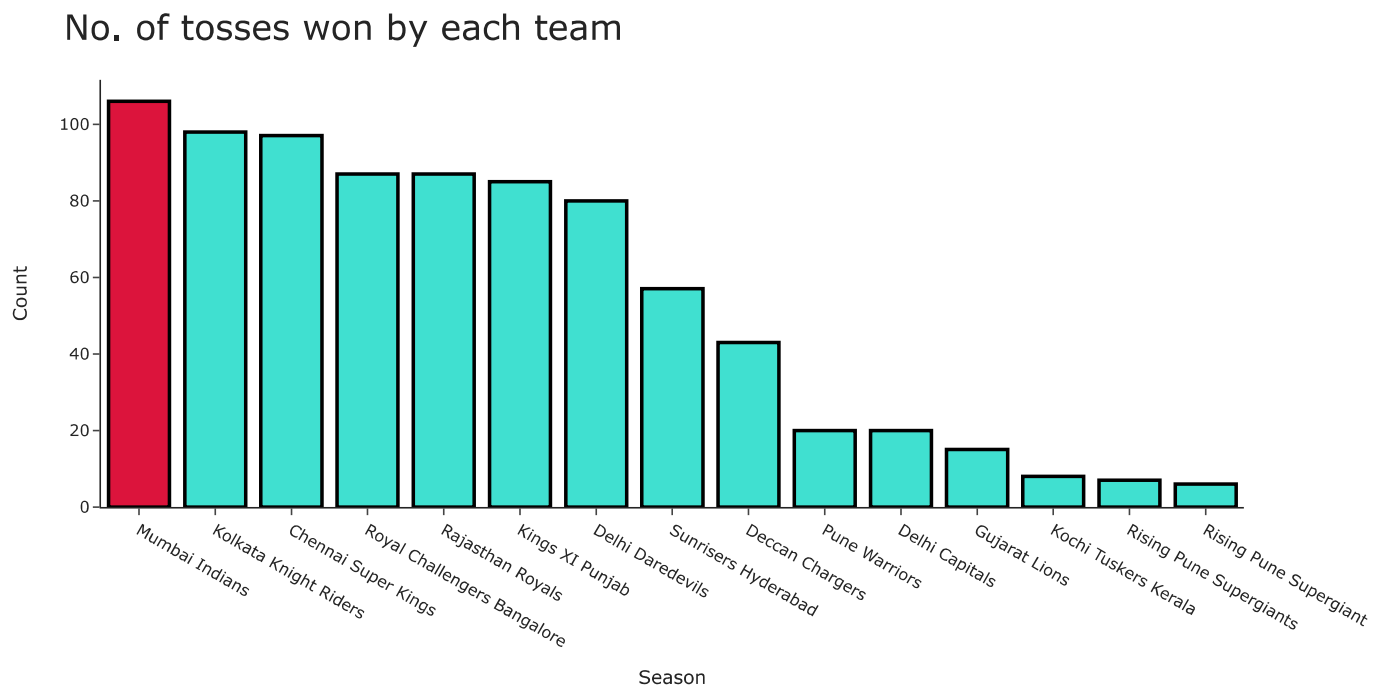
S Ravi has umpired in 121 matches, followed by Dharmasena who has umpired in 94 matches.

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
```

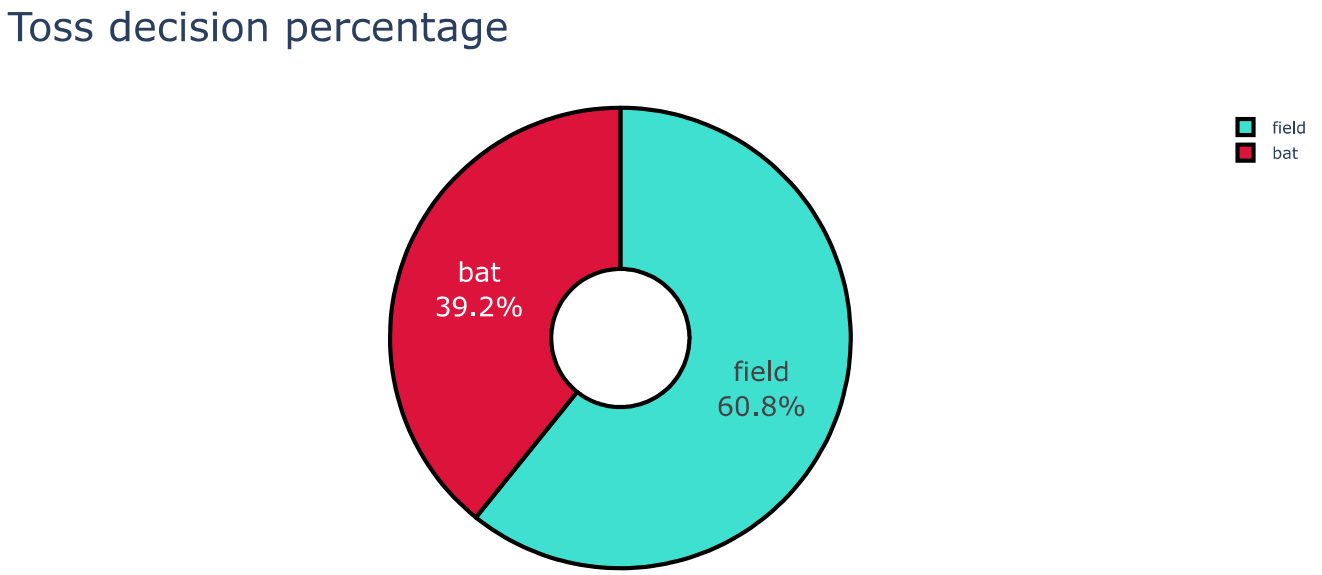
```
toss=match_data['toss_winner'].value_counts()
colors = ['turquoise',] * 15
colors[0] = 'crimson'
fig=px.bar( y=toss,x=toss.index,labels=dict(x="Season",y="Count"),)
fig.update_layout(title="No. of tosses won by each team",
                  titlefont={'size': 26},template='simple_white'
        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



Mumbai Indians have won the most tosses, followed by Kolkata Knight Riders

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()
```



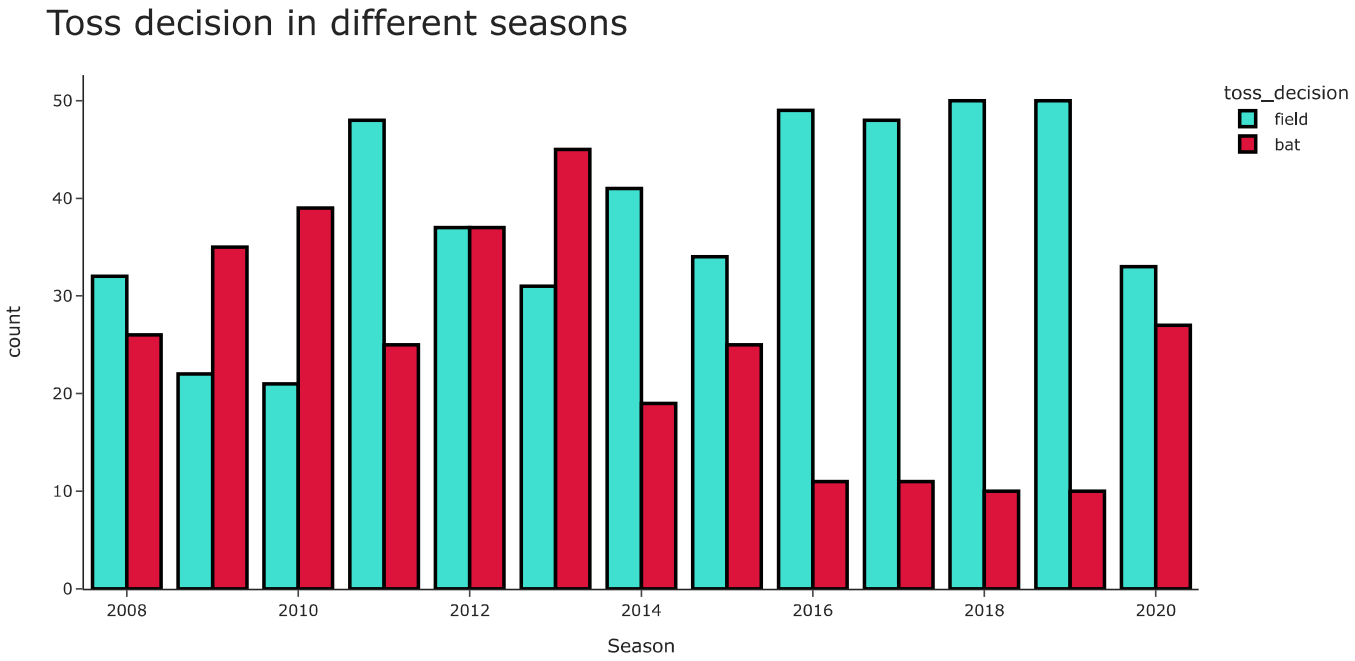
After winning the toss, team tends to field first

7) Toss decision across seasons

```
match_data.head()
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore

```
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()
```



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.

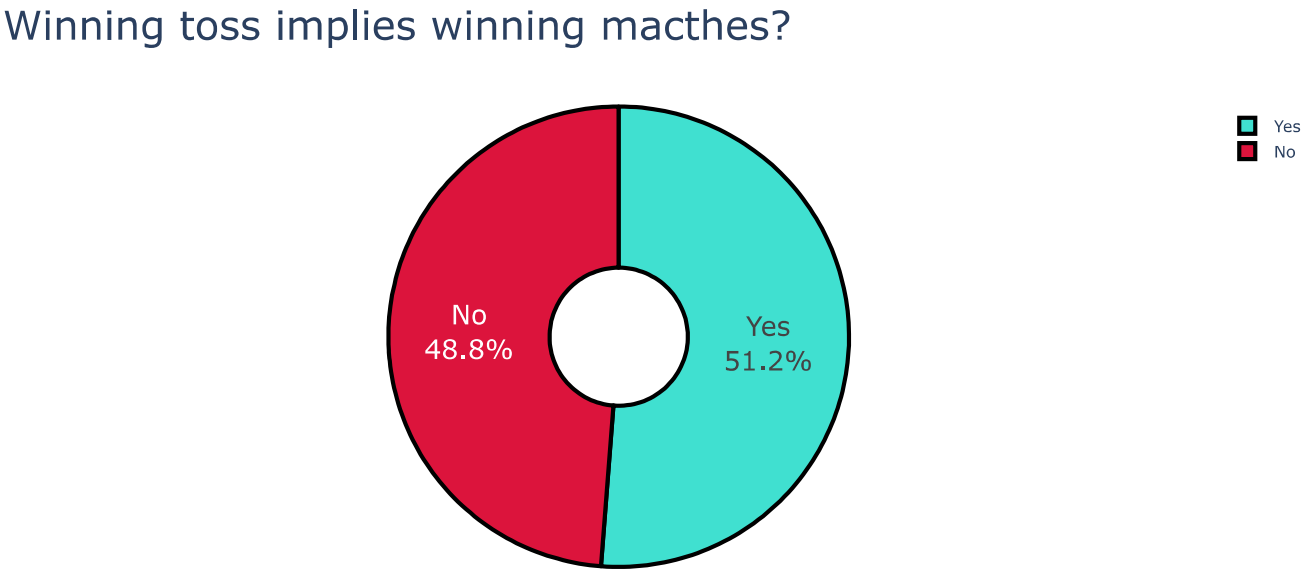
8)Winning toss implies winning game

```
match_data['toss_win_game_win'] = np.where((match_data.toss_winner == match_data.winner),'Yes','No')
match_data.head(20)
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winn
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolk Kni Rid
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chen Su Kir
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	De Daredev
3	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Ro Challeng Bangal
4	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolk Kni Rid
5	335987	Jaipur	2008-04-21	SR Watson	Sawai Mansingh Stadium	0	Rajasthan Royals	Kings XI Punjab	Kings XI Punjab	bat	Rajasth Roy
6	335988	Hyderabad	2008-04-22	V Sehwag	Rajiv Gandhi International Stadium, Uppal	0	Deccan Chargers	Delhi Daredevils	Deccan Chargers	bat	De Daredev
7	335989	Chennai	2008-04-23	ML Hayden	MA Chidambaram Stadium, Chepauk	0	Chennai Super Kings	Mumbai Indians	Mumbai Indians	field	Chen Su Kir
8	335990	Hyderabad	2008-04-24	YK Pathan	Rajiv Gandhi International Stadium, Uppal	0	Deccan Chargers	Rajasthan Royals	Rajasthan Royals	field	Rajasth Roy
9	335991	Chandigarh	2008-04-25	KC Sangakkara	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Mumbai Indians	Mumbai Indians	field	Kings Pun
10	335992	Bangalore	2008-04-26	SR Watson	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	Rajasth Roy
11	335993	Chennai	2008-04-26	JDP Oram	MA Chidambaram Stadium, Chepauk	0	Chennai Super Kings	Kolkata Knight Riders	Kolkata Knight Riders	bat	Chen Su Kir
12	335994	Mumbai	2008-04-27	AC Gilchrist	Dr DY Patil Sports Academy	0	Mumbai Indians	Deccan Chargers	Deccan Chargers	field	Decc Charg
13	335995	Chandigarh	2008-04-27	SM Katich	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Delhi Daredevils	Delhi Daredevils	bat	Kings Pun
14	335996	Bangalore	2008-04-28	MS Dhoni	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Chennai Super Kings	Chennai Super Kings	bat	Chen Su Kir
15	335997	Kolkata	2008-04-29	ST Jayasuriya	Eden Gardens	0	Kolkata Knight Riders	Mumbai Indians	Kolkata Knight Riders	bat	Mum Indi
16	335998	Delhi	2008-04-30	GD McGrath	Feroz Shah Kotla	0	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	De Daredev
17	335999	Hyderabad	2008-05-01	SE Marsh	Rajiv Gandhi International Stadium, Uppal	0	Deccan Chargers	Kings XI Punjab	Kings XI Punjab	field	Kings Pun
18	336000	Jaipur	2008-05-01	SA Asnodkar	Sawai Mansingh Stadium	0	Rajasthan Royals	Kolkata Knight Riders	Rajasthan Royals	bat	Rajasth Roy

MA Chennai

```
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()
```



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

9) Match win result

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

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

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

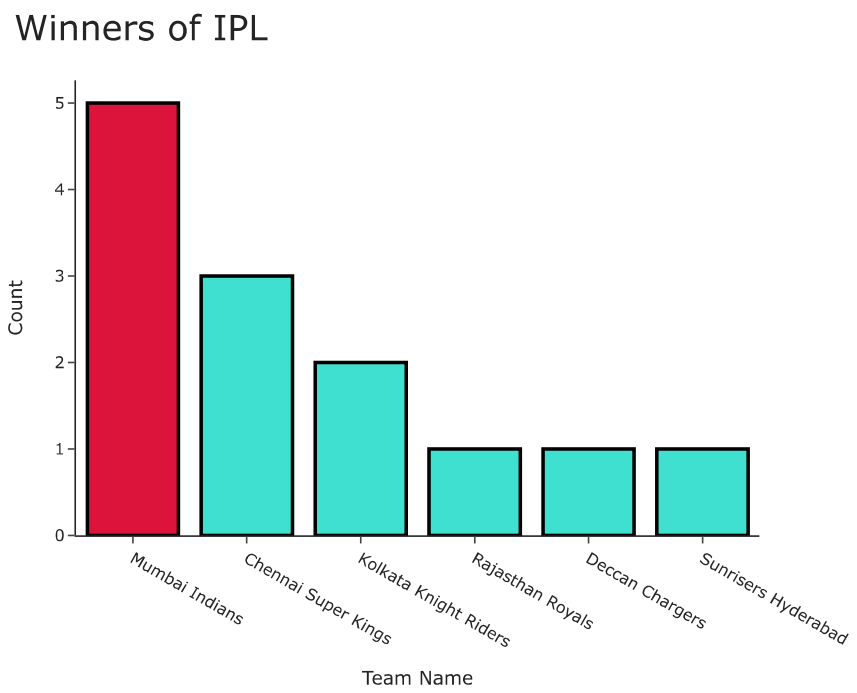
10) Number of times team have won the tournament

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

#dictionaries to get winners to each season
winners_team = {}
for i in sorted(winning_teams.Season.unique()):
    winners_team[i] = winning_teams[winning_teams.Season == i]['winner'].tail(1).values[0]
```

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


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()
```



11) 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)
```

	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
12	Rising Pune Supergiant	16
13	Kochi Tuskers Kerala	14
14	Rising Pune Supergiants	14

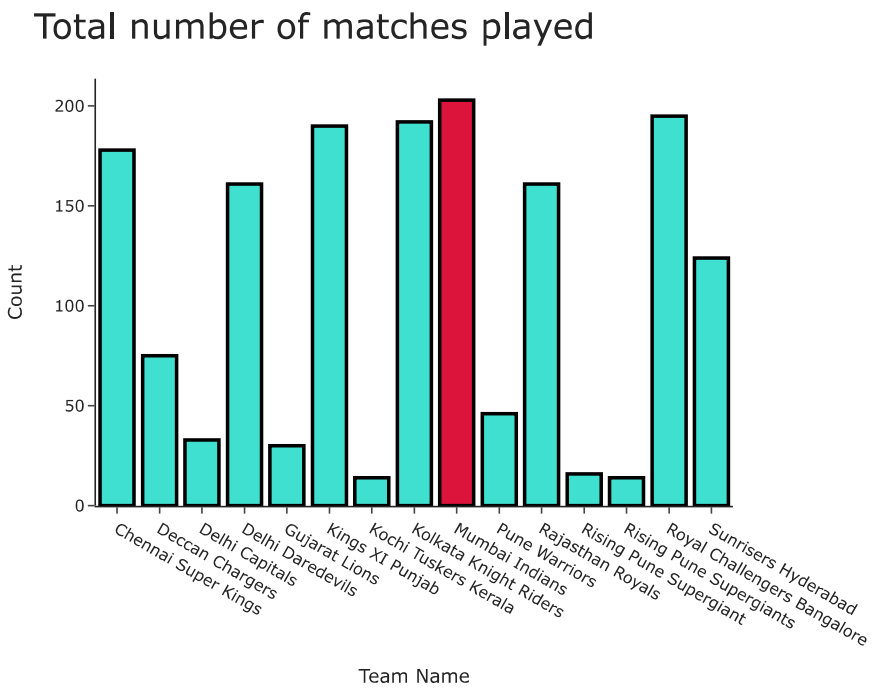
```
wins=pd.DataFrame(match_data['winner'].value_counts()).reset_index()
wins.columns=['Team Name','Wins']
wins
```

	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)
```

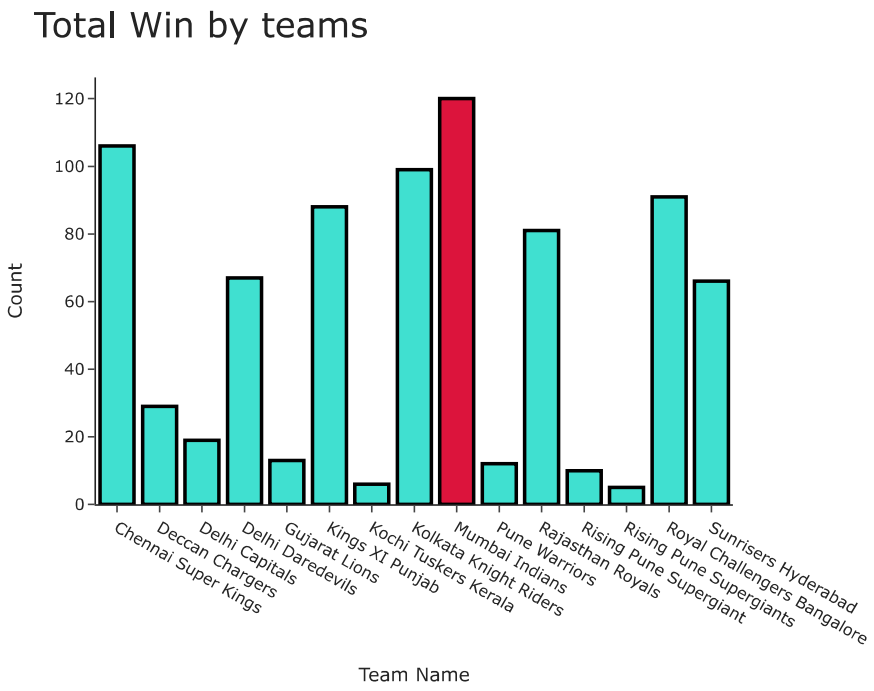
	Team Name	Total Matches played	Wins	% Win
0	Rising Pune Supergiant	16	10	62.500000
1	Chennai Super Kings	178	106	59.550562
2	Mumbai Indians	203	120	59.113300
3	Delhi Capitals	33	19	57.575758
4	Sunrisers Hyderabad	124	66	53.225806
5	Kolkata Knight Riders	192	99	51.562500
6	Rajasthan Royals	161	81	50.310559
7	Royal Challengers Bangalore	195	91	46.666667
8	Kings XI Punjab	190	88	46.315789
9	Gujarat Lions	30	13	43.333333
10	Kochi Tuskers Kerala	14	6	42.857143
11	Delhi Daredevils	161	67	41.614907
12	Deccan Chargers	75	29	38.666667
13	Rising Pune Supergiants	14	5	35.714286
14	Pune Warriors	46	12	26.086957

```
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()
```



12) Most number of wins

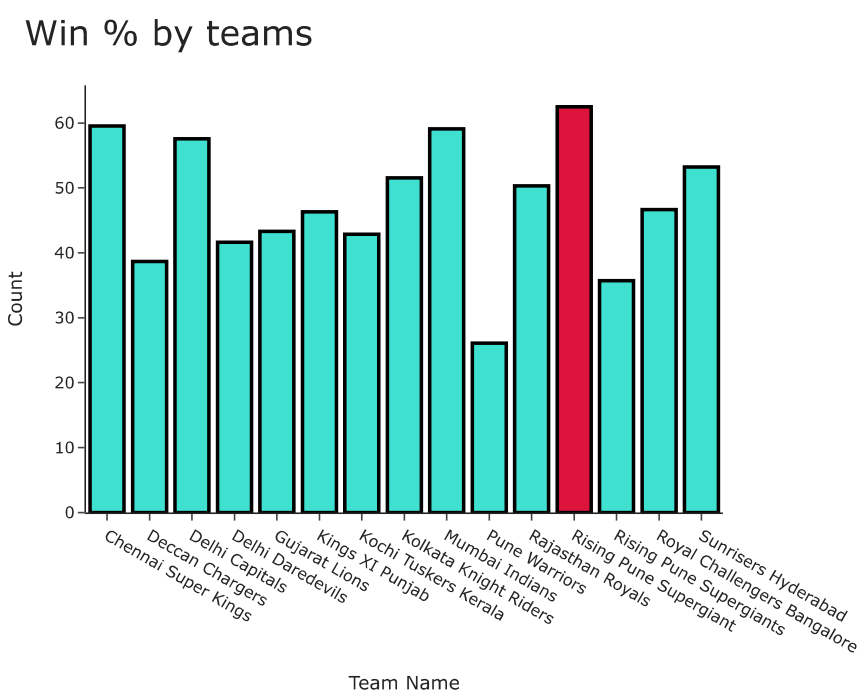
```
colors = ['turquoise',] * 15
colors[8] = 'crimson'
fig=px.bar(x=played['Team Name'],y=played['Wins'],labels=dict(x="Team Name",y="Count"),)
fig.update_layout(title="Total Win by teams",
                  titlefont={'size': 26},template='simple_white'
                  )
fig.update_traces(marker_line_color='black',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



Mumbai Indians had won the most matches(120), followed by Chennai Super Kings (106)

13) 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()
```



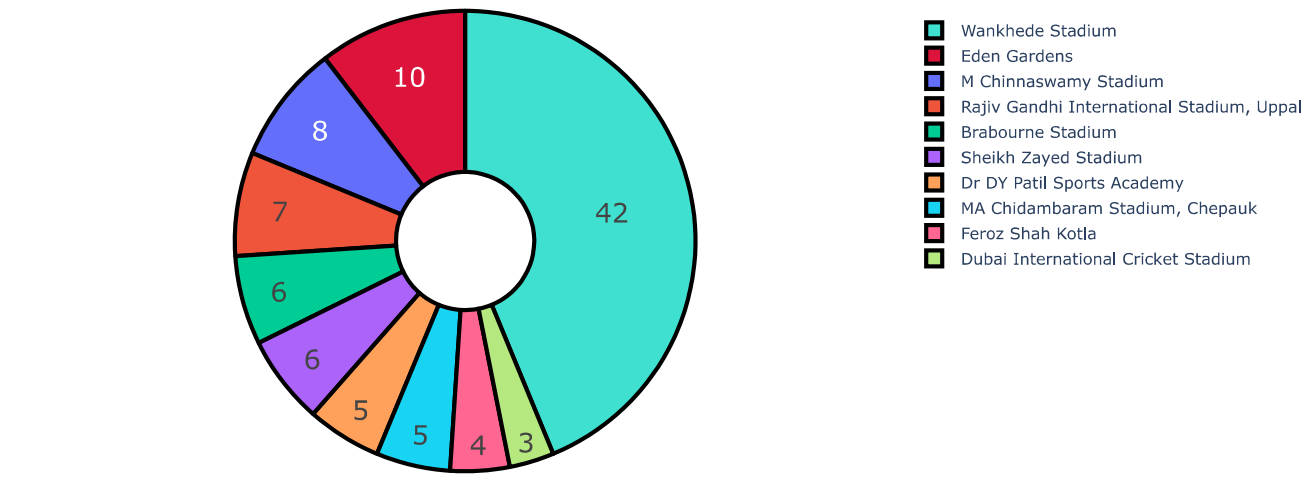
Rising Pune Supergiants have the highest win % of 62.50, followed by Chennai Super kings and Mumbai Indians.

This is largely due to the fact that they had played really few matches.

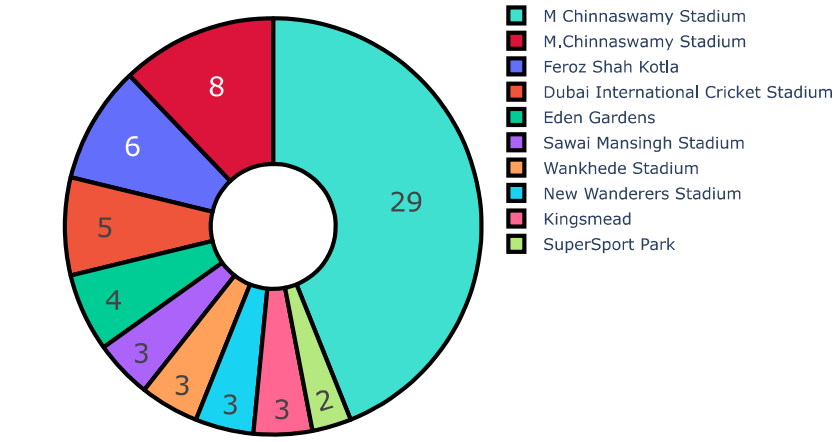
14) Lucky Venues for a team

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

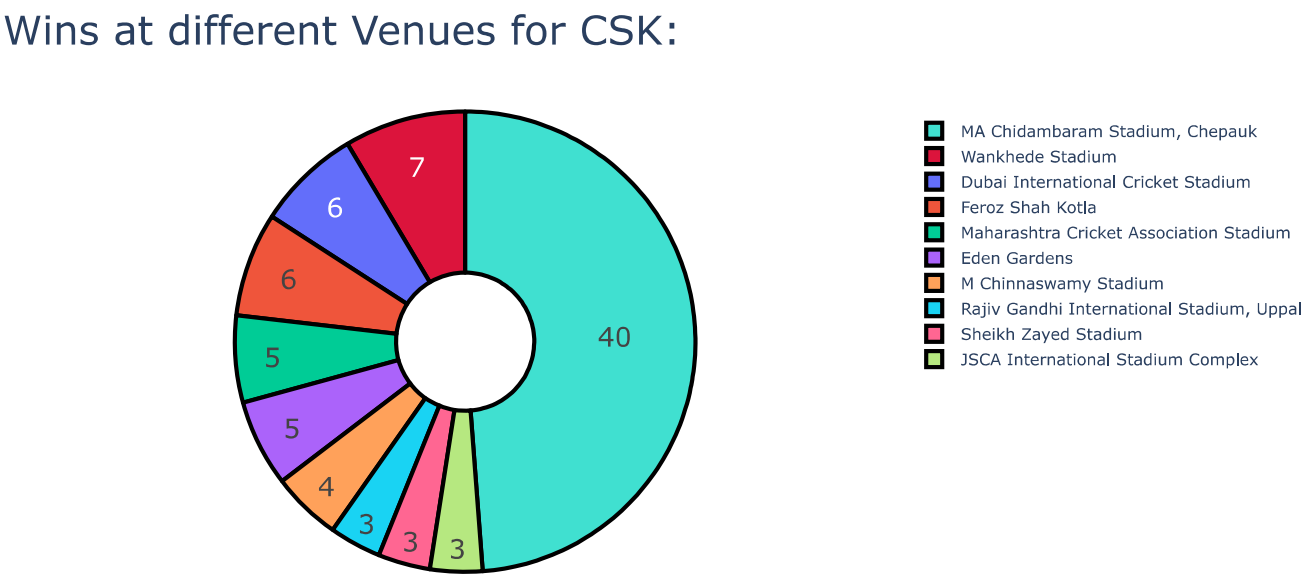
```
mi=lucky(match_data,'Mumbai Indians')
values = mi
labels=mi.index
colors = ['turquoise', 'crimson']
fig = go.Figure(data=[go.Pie(labels=labels,values=values,hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000',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




```
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()
```



```
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()
```



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

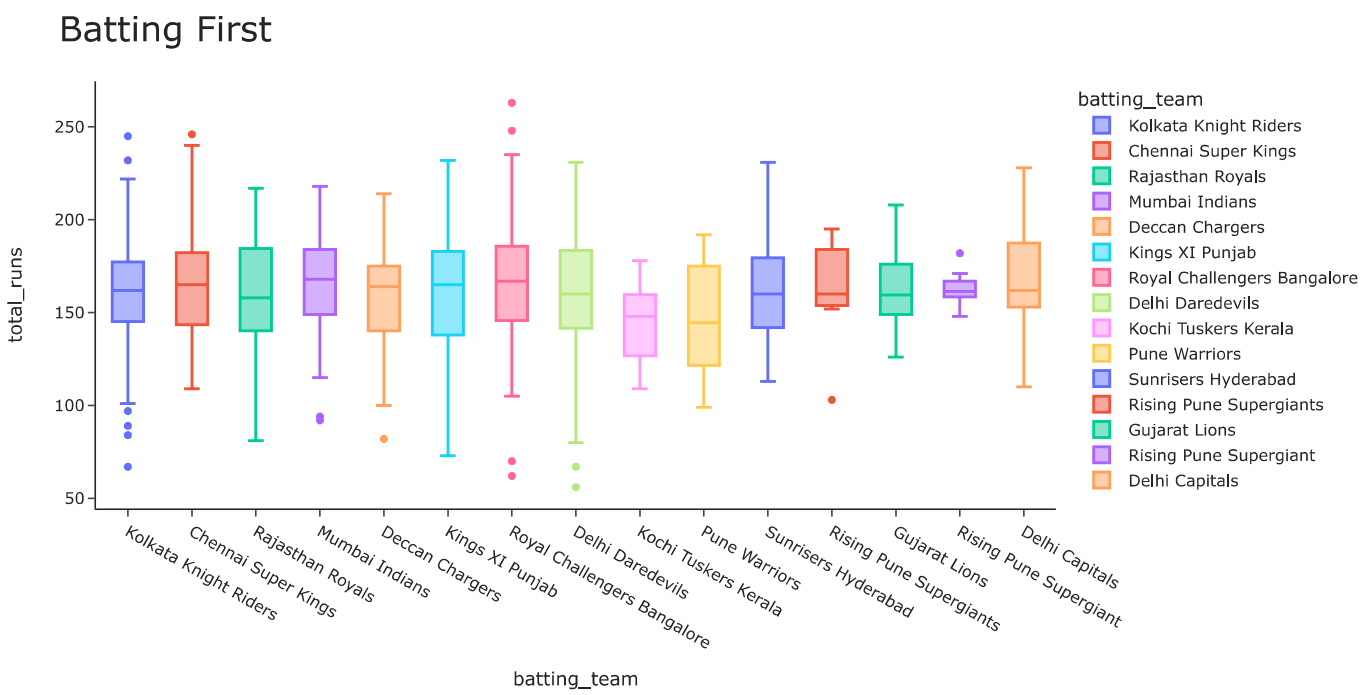
15) Innings wise comparision

```
runs=ball_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]
```

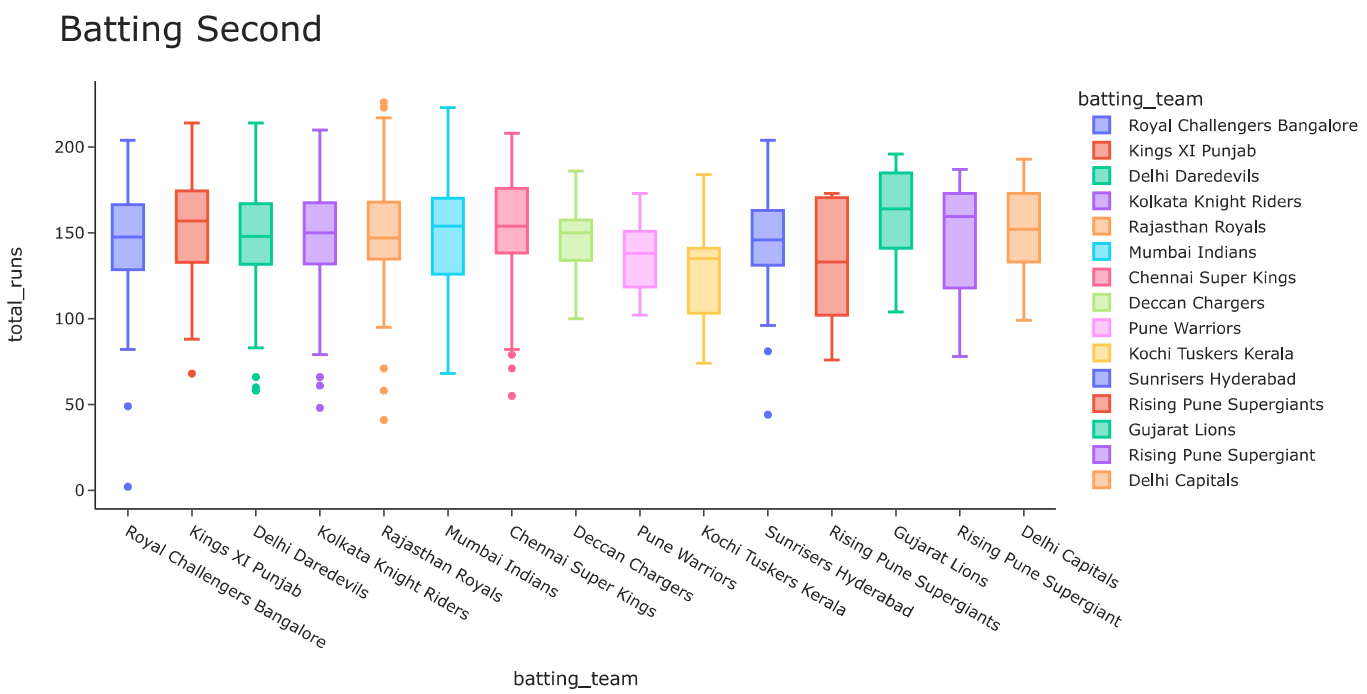
```
fig = px.box(y='total_runs',x='batting_team',data_frame=inning1,color='batting_team')

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



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.

```
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()
```



16) Scored 200+ runs

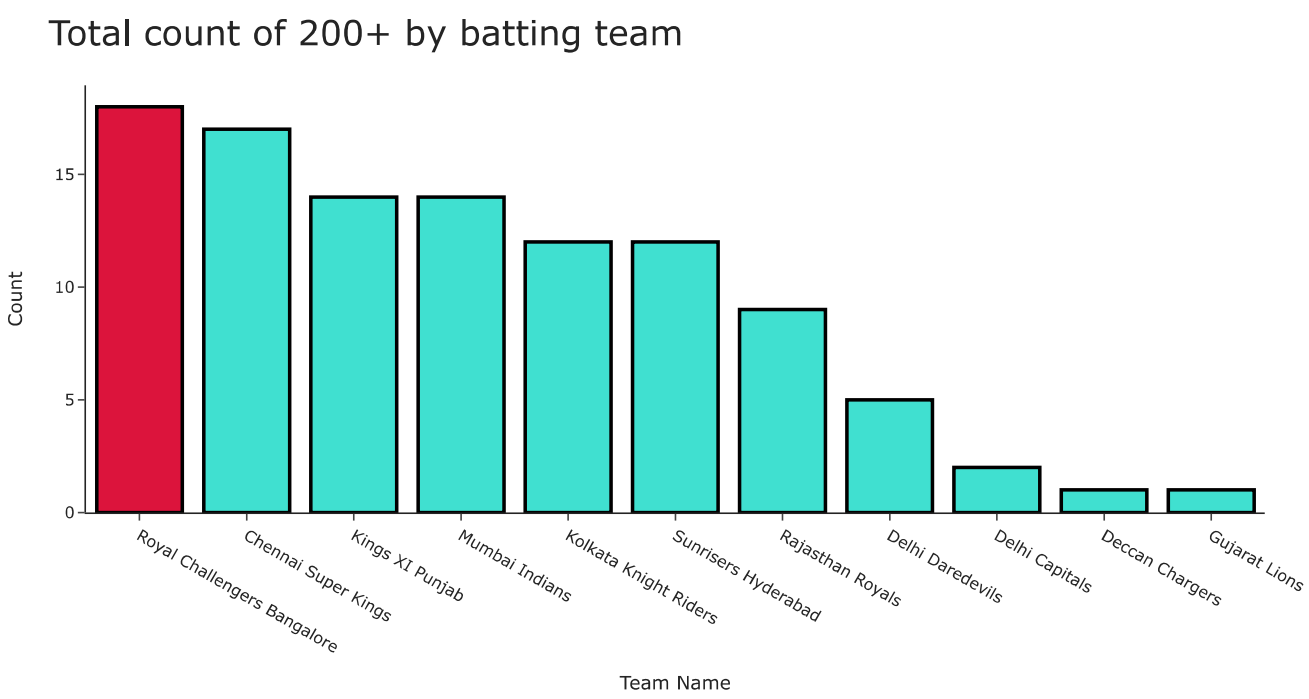
```
high_scores=ball_data.groupby(['id', 'inning','batting_team','bowling_team'])['total_runs'].sum().reset_index()
score_200=high_scores[high_scores['total_runs']>=200]
score_200.head(5)
```

	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
```

	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()
```



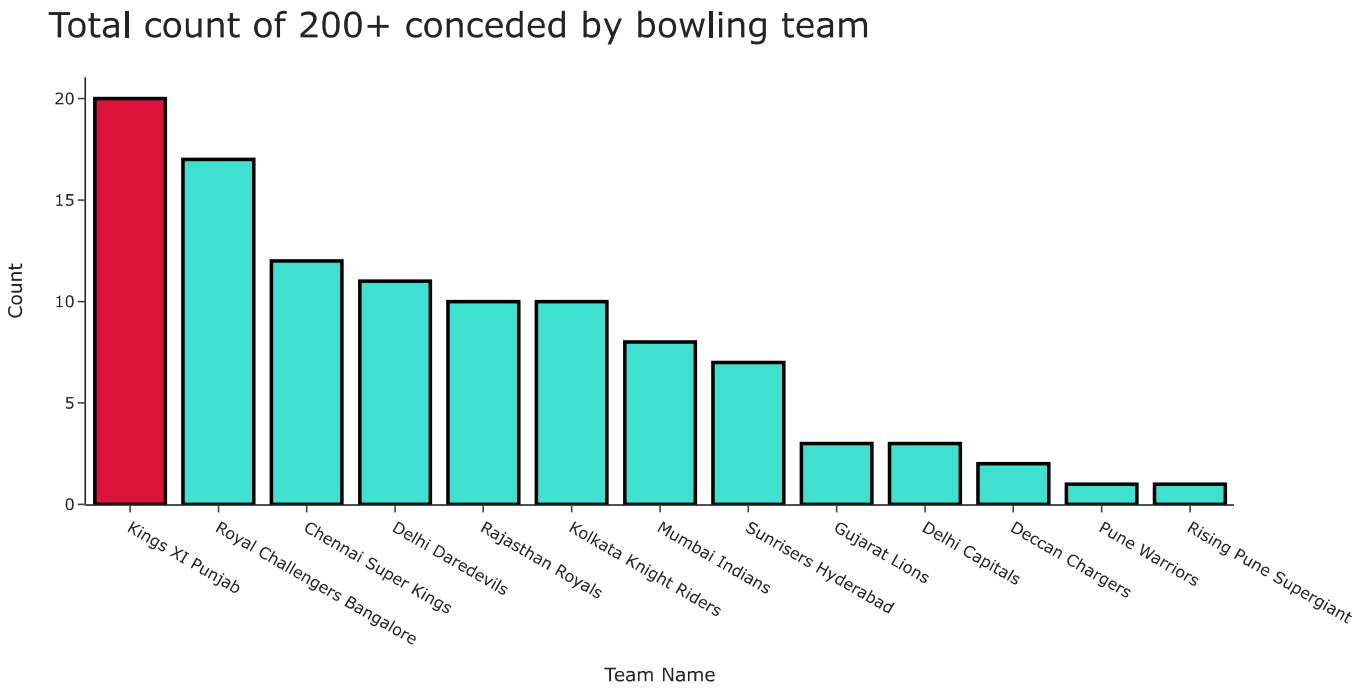
Royal Challengers Bangalore had scored the most 200+ score (18 times), followed by Chennai Super Kings who had scored 17 times.

17) Conceded 200+ runs

```
z=score_200['bowling_team'].value_counts()
z=pd.DataFrame(z)
z
```

	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()
```



Kings XI Punjab has conceded 200+ runs 20 times, followed by Royal Challengers Bangalore

18) Highest runs in an innings

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

id	inning	batting_team	bowling_team	total_runs
335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	222
335982	2	Royal Challengers Bangalore	Kolkata Knight Riders	82
335983	1	Chennai Super Kings	Kings XI Punjab	240
335983	2	Kings XI Punjab	Chennai Super Kings	207
335984	1	Rajasthan Royals	Delhi Daredevils	129
...	...	...	...	...
1237178	2	Sunrisers Hyderabad	Royal Challengers Bangalore	132
1237180	1	Delhi Capitals	Sunrisers Hyderabad	189
1237180	2	Sunrisers Hyderabad	Delhi Capitals	172
1237181	1	Delhi Capitals	Mumbai Indians	156
1237181	2	Mumbai Indians	Delhi Capitals	157

1628 rows × 4 columns

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

```
263
```

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

19) 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	result	result_m
620	1082635	Delhi	2017-05-06	LMP Simmons	Feroz Shah	0	Delhi Daredevils	Mumbai Indians	Delhi Daredevils	field	Mumbai Indians	runs	

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

20)Most balls played by a batsman

```
balls_played=ball_data.groupby(['batsman'])['ball'].count().reset_index()
balls_played.sort_values(by='ball',ascending=False).head(10)
```

	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
42	AM Rahane	3325

Virat kohli had played the most balls.

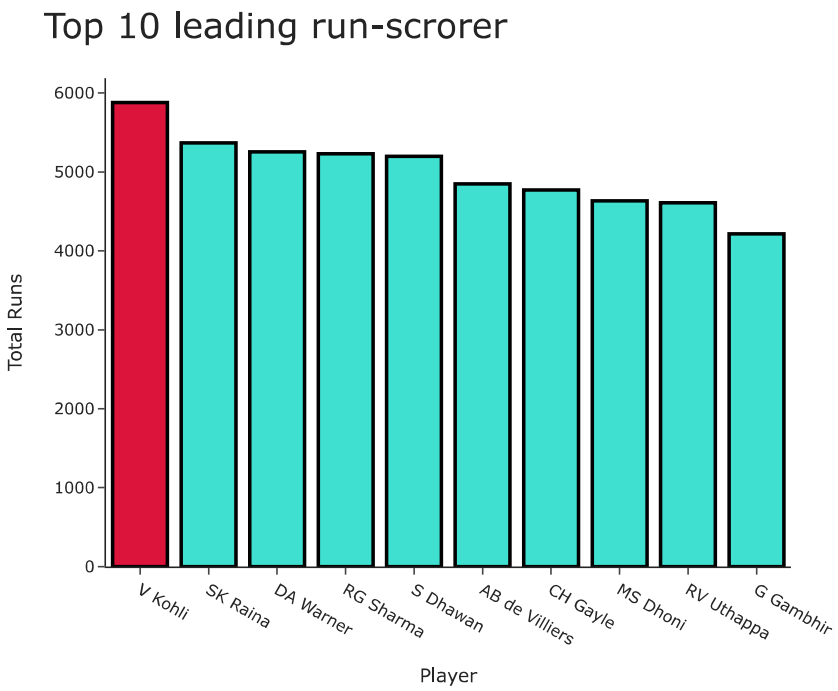
David Warner and Chris Gayle are the only two foreign players in this top 10 list.

21) Top 10 run scorer of all time

```
runs=ball_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
```

	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()
```



Virat Kohli is the leading run scorer in IPL.

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

22) Most number of 4's

```
balls_played=balls_played.merge(runs,left_on='batsman',right_on='Batsman',how='outer')
four=ball_data[ball_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)
```

	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

Shikhar Dhawan holds the record for most number of 4's

23) Most number of 6's

```
six=ball_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)
```

	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

Chris Gayle had smashed 349 sixes, most by any batsman

24) Highest Strike rate

```
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

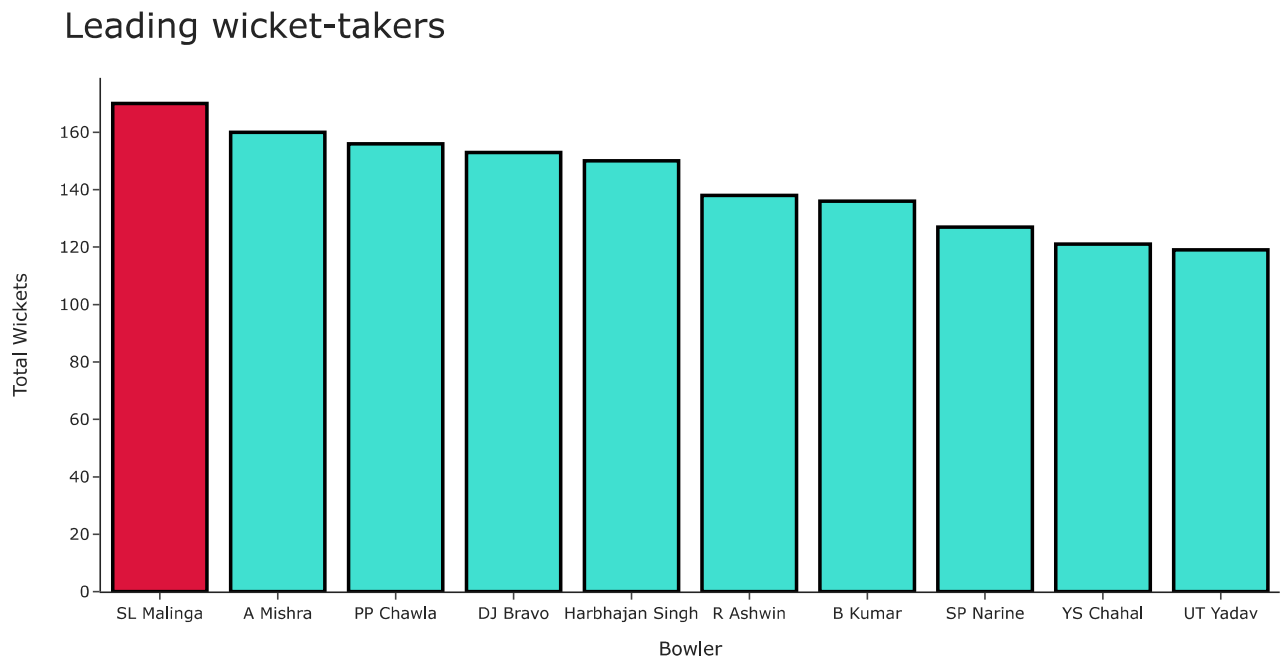
```
reqsr=sr.drop(columns=['runs','ball','4s','6s'],axis=1)
reqsr.sort_values(by='strike_rate',ascending=False).head(10)
```

	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

Andre Russel has the highest strike rate, followed by K. Gowtham and B. Cutting.

25) Highest wicket-taker

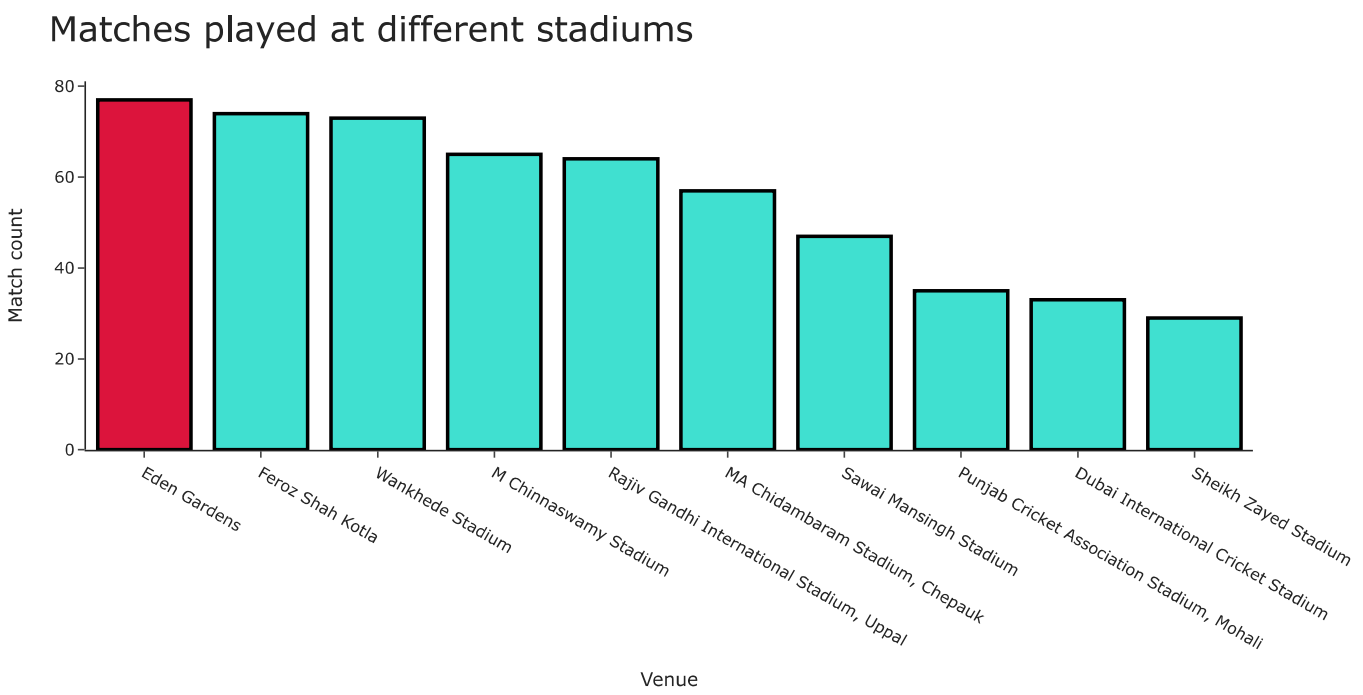
```
ball_data['dismissal_kind'].unique()
dismissal_kinds = ['caught', 'bowled', 'lbw', 'caught and bowled',
       'stumped', 'hit wicket']
hwt=ball_data[ball_data["dismissal_kind"].isin(dismissal_kinds)]
bo=hwt['bowler'].value_counts()


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()
```



L. Malinga is the leading wicket taker in IPL.

26) Total count of matches played in differnt stadiums

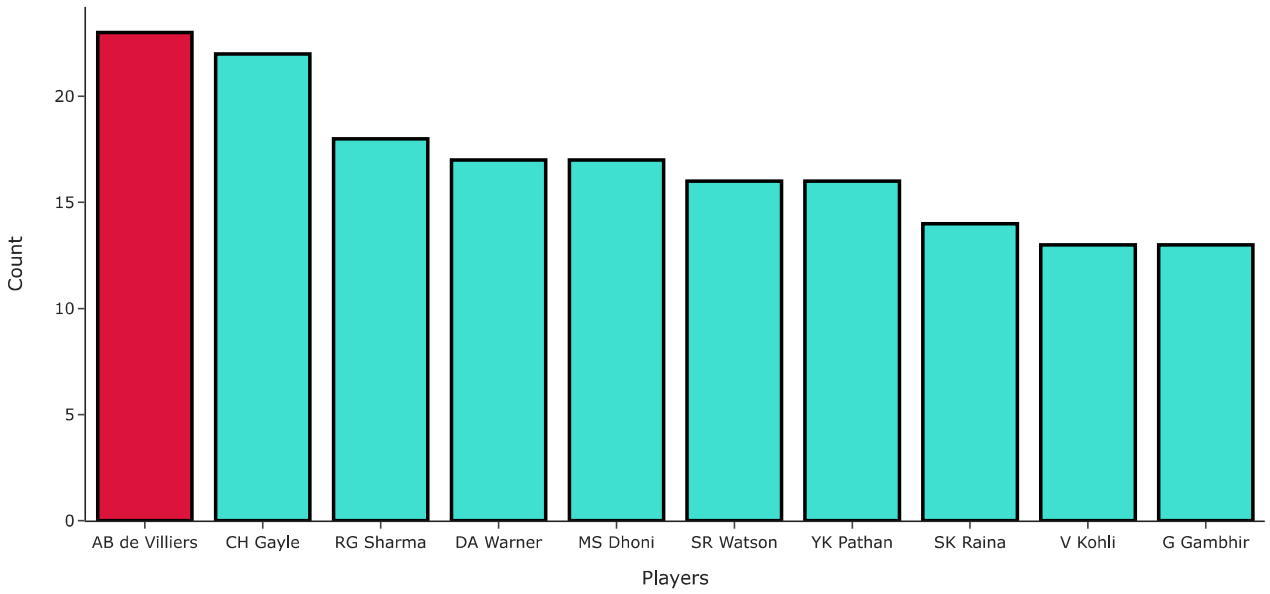
```
colors = ['turquoise',] * 13
colors[0] = 'crimson'
fig=px.bar(x=match_data['venue'].value_counts()[:10].index,y=match_data['venue'].value_counts()[:10],labels=dict(x="Venue",y="Match count"),
fig.update_layout(title="Matches played at different stadiums",
                  titlefont={'size': 26},template='simple_white'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



Eden Gardens has hosted most number of matches.

27) Man of the Match award

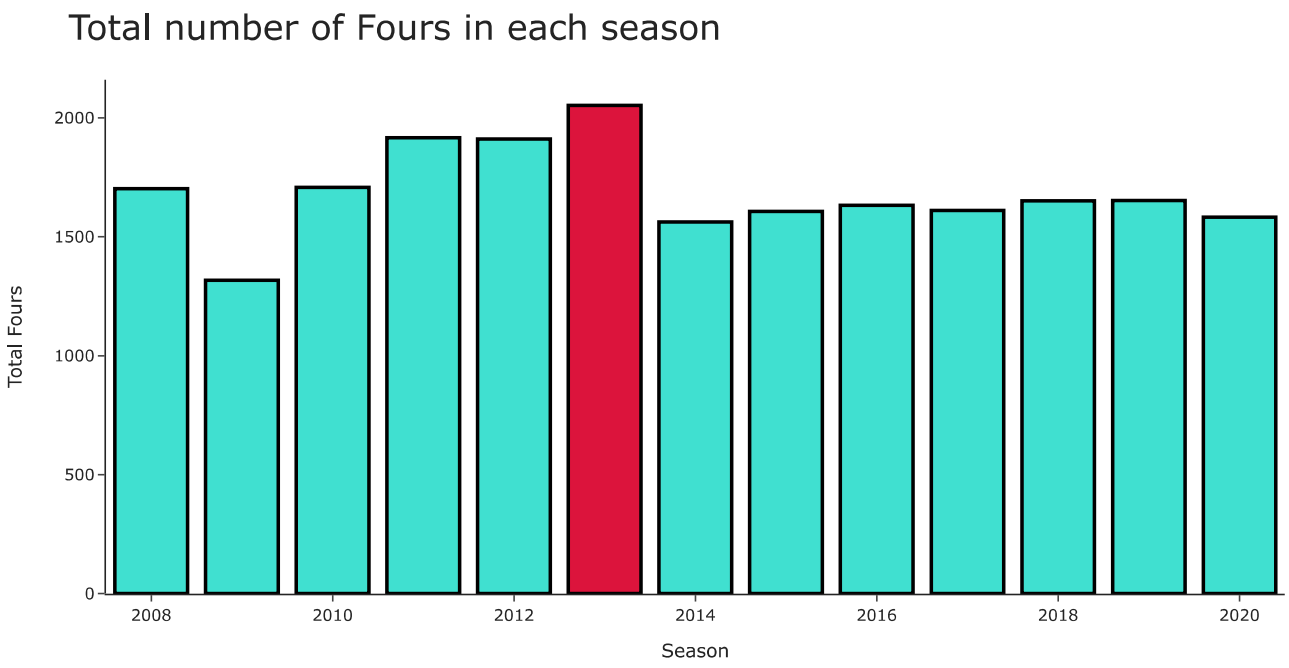
```
colors = ['turquoise',] * 11
colors[0] = 'crimson'
fig=px.bar(x=match_data.player_of_match.value_counts()[:10].index,y=match_data.player_of_match.value_counts()[:10],labels=dict(x="Players",y=
fig.update_layout(title="Top 10 MOM awardee",
                  titlefont={'size': 26},template='simple_white'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



AB de Villiers (23) had won the most MOM awards, followed by Chris Gayle (22)

28) Total number of fours in each season

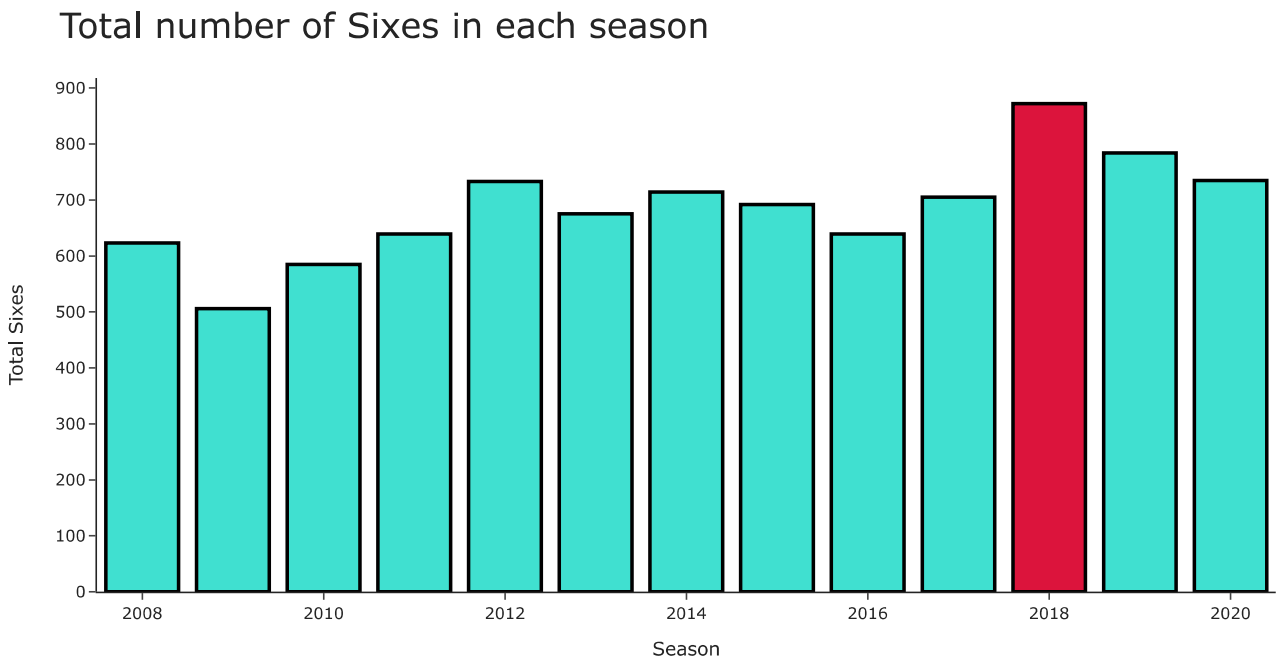
```
data_4 = match_data['Season'].unique()

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 = ball_data[ball_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()
```



Highest number of four in a season was is 2013 season while season 2009 had the lowest count of 4's.

29) Total number of sixes in each season

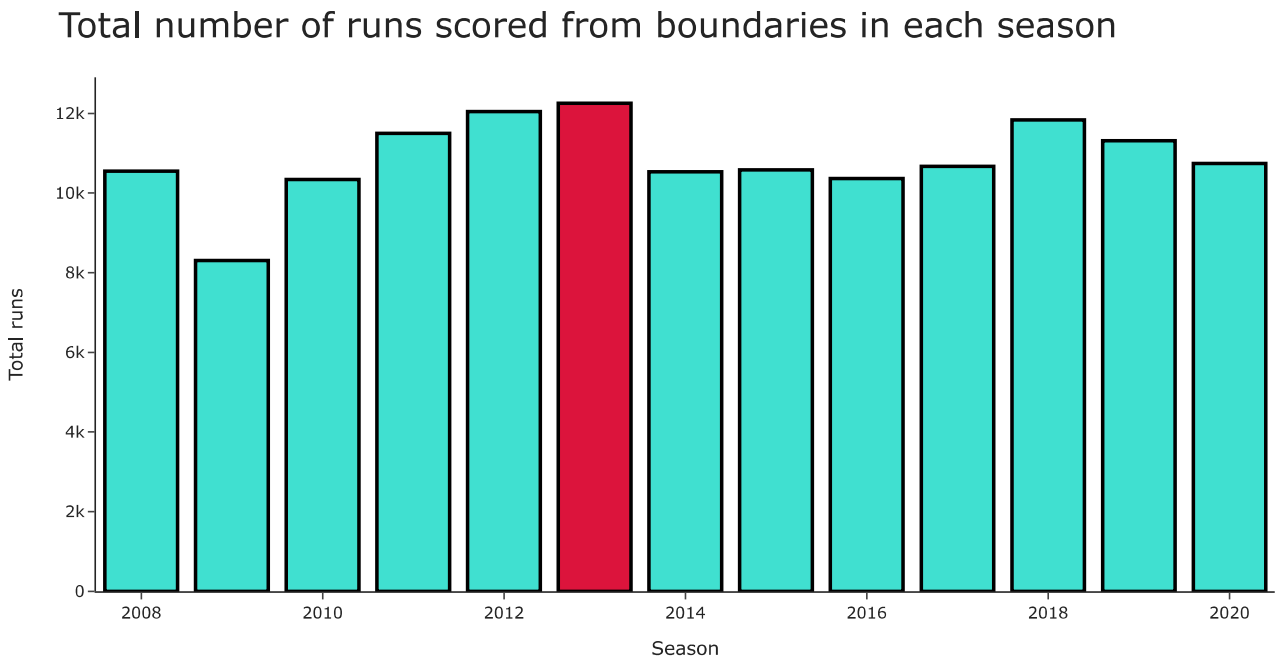
```
data_6 = match_data['Season'].unique()

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 = ball_data[ball_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()
```



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

30) 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()
```



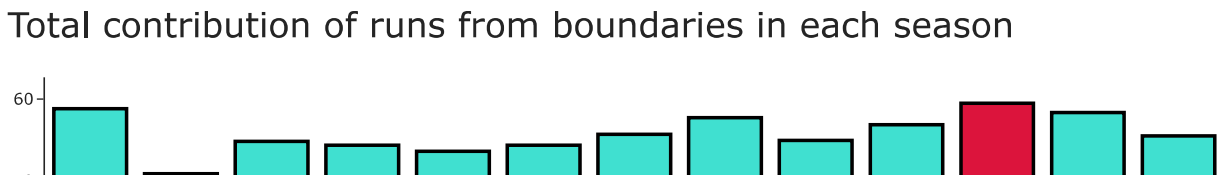
Total run scored from boundaries is lowest in season 2009 and highest in season 2013.

31) 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()
```



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.

32) Total runs scored by teams in first 6 overs

```
team = ball_data['batting_team'].unique()
team_runs = []
for var in team:
    temp_df = ball_data[ball_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()
```