IPL All Seasons (2008-2018)

About IPL

The Indian Premier League (IPL) is a professional Twenty20 cricket league in India contested during March or April and May of every year by eight teams representing eight different cities in India.The league was founded by the Board of Control for Cricket in India (BCCI) in 2008. IPL has an exclusive window in ICC Future Tours Programme.

The IPL is the most-attended cricket league in the world and in 2014 ranked sixth by average attendance among all sports leagues.In 2010, the IPL became the first sporting event in the world to be broadcast live on YouTube. The brand value of IPL in 2018 was US$6.3 billion, according to Duff & Phelps. According to BCCI, the 2015 IPL season contributed ₹11.5 billion to the GDP of the Indian economy.

There have been twelve seasons of the IPL tournament. The current IPL title holders are the Mumbai Indians, who won the 2019 season.

About Dataset

This data set contains the following data:

1. Match_date
2. Match_number
3. Match_venue
4. Match_time
5. Toss_winner
6. Toss_decision
7. Team1
8. Team1_score
9. Team2
10. Team2_score
11. Winning_team
12. Winning_margin

Total numberof rows: 704

About Project

This project is about creating meaningful Data Visualizations to understand trends in IPL.

The activites invovled are:

1. Data cleaning
2. Data preprocessing
3. Handling missing values
4. Exploratory data analysis (Various Visualizations)
5. Conclusions

In [1]:

```
import warnings
warnings.filterwarnings('ignore')

#General Libraries
import os
import sqlite3
import pandas as pd
from pandas import read_excel
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

In [2]:

```
plt.rcParams["figure.figsize"] = (10,10)
sns.set_style("whitegrid")
```

Reading data from Excel file

In [3]:

```
raw_data = pd.read_excel('iplallseasons_refined.xlsx',sheet_name='Sheet1')
```

In [4]:

```
raw_data.head(5)
```

Out[4]:

	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2_scc
0	Apr 18, 2008	1st match	M Chinnaswamy Stadium, Bangalore	night	Royal Challengers Bangalore	field	Kolkata Knight Riders	222/3	RCB	
1	Apr 19, 2008	2nd match	Punjab Cricket Association Stadium, Mohali, Ch...	day/night	Chennai Super Kings	bat	Chennai Super Kings	240/5	Kings XI Punjab	207
2	Apr 19, 2008	3rd match	Feroz Shah Kotla, Delhi	night	Rajasthan Royals	bat	Rajasthan Royals	129/8	Delhi Daredevils	132
3	Apr 20, 2008	4th match	Eden Gardens, Kolkata	day/night	Deccan Chargers	bat	Deccan Chargers	110	Kolkata Knight Riders	112
4	Apr 20, 2008	5th match	Wankhede Stadium, Mumbai	night	Mumbai Indians	bat	Mumbai Indians	165/6	RCB	166

Raw data: Issue with data and what sort of cleansining is required

- The name to teams are not in sync at all the columns, for eg, at some places it is written as RCB and at some places Royal challengers banglore, same is true for Deccan Chargers and Sunrisers Hyderabad. We need to bring this in sync and better idea is to just address the team as a state/city name they belong to.
- Changes needs to be made to spilt the year from the date for mathematical funtions.Also special characters from the team scores needs to be removed.
- Winning marging data also needs to be handled, to represent the data better.

In [5]:

```
raw_data.count()
```

Out[5]:

```
Match_date        704
Match_number      704
Match_venue       704
Match_time        704
Toss_winner       702
Toss_decision     702
Team1             704
Team1_score       696
Team2             704
Team2_score       694
Winning_team      704
Winning_margin    686
dtype: int64
```

Observation: We can observe here that few of the fields are missing the data like, toss_winner, toss_decission

Observation: We can observe here that few of the fields are missing the data like, toss_winner, toss_decission, team1_score, team2_score, winning_margin

In [6]:

```
#Function to Replace different team names by state or city name
def CleanTeamName(x):

    if (x == 'Royal Challengers Bangalore' or x == 'RCB'):
        team = 'Banglore'
    elif (x == 'Rajasthan Royals' or x == 'RR'):
        team = 'Rajasthan'
    elif (x == 'Chennai Super Kings' or x == 'CSK'):
        team = 'Chennai'
    elif (x == 'Deccan Chargers' or x == 'Sunrisers Hyderabad' or x == 'SRH'):
        team = 'Hyderabad'
    elif (x == 'Mumbai Indians'):
        team = 'Mumbai'
    elif (x == 'Kings XI Punjab'):
        team = 'Punjab'
    elif (x == 'Kolkata Knight Riders' or x == 'KKR'):
        team = 'Kolkata'
    elif (x == 'Delhi Daredevils'):
        team = 'Delhi'
    elif (x == 'Kochi Tuskers Kerala'):
        team = 'Kerala'
    elif (x == 'Pune Warriors' or x == 'Rising Pune Supergiants' or x == 'Rising Pune Supergiant'):
        team = 'Pune'
    elif (x == 'no toss'):
        team = 'Match Abandoned'
    elif (x == 'Gujarat Lions'):
        team = 'Gujarat'
    elif (x == 'Match Tie'):
        team = 'Match Tie'
    else:
        team = 'Recheck'
    return team
```

In [7]:

```
New = []
for i in raw_data['Toss_winner']:
    New_item = CleanTeamName(i)
    New.append(New_item)
raw_data['Toss_winner'] = New
```

In [8]:

```
print("Number of data points in our data", raw_data.shape)
print(raw_data['Toss_winner'].value_counts())
```

```
Number of data points in our data (704, 12)
Mumbai             90
Kolkata            87
Hyderabad          85
Delhi              80
Banglore           78
Chennai            77
Punjab             75
Rajasthan          68
Pune               33
Gujarat            15
Kerala              8
Match Abandoned     6
Recheck             2
Name: Toss_winner, dtype: int64
```

Removing irrelevant rows

In [9]:

```
raw_data.loc[raw_data.Toss_winner == 'Recheck']
```

Out[9]:

	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2_sco
488	Apr 26, 2015	25th match	Eden Gardens, Kolkata	day/night	Recheck	NaN	Kolkata Knight Riders	NaN	Rajasthan Royals	Na
492	Apr 29, 2015	29th match	M Chinnaswamy Stadium, Bangalore	night	Recheck	NaN	RCB	200/7	Rajasthan Royals	Na

In [10]:

```
raw_data.loc[raw_data.Toss_winner == 'Match Abandoned']
```

Out[10]:

	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2_
46	May 22, 2008	47th match	Feroz Shah Kotla, Delhi	night	Match Abandoned	no	Delhi Daredevils	NaN	Kolkata Knight Riders	
65	Apr 21, 2009	7th match	Kingsmead, Durban	day/night	Match Abandoned	no	Mumbai Indians	NaN	Rajasthan Royals	
71	Apr 25, 2009	13th match	Newlands, Cape Town	day/night	Match Abandoned	no	Chennai Super Kings	NaN	Kolkata Knight Riders	
197	Apr 19, 2011	20th match	M Chinnaswamy Stadium, Bangalore	night	Match Abandoned	no	RCB	NaN	Rajasthan Royals	
283	Apr 24, 2012	32nd match	Eden Gardens, Kolkata	night	Match Abandoned	no	Kolkata Knight Riders	NaN	Deccan Chargers	
612	Apr 25, 2017	29th match	M Chinnaswamy Stadium, Bangalore	night	Match Abandoned	no	RCB	NaN	Sunrisers Hyderabad	

Total 8 matched are abandoned and hence these do not contribute to our data in a meaniful way, so we can remove these 8 rows from our data.

In [11]:

```
refine_data = raw_data.drop(raw_data[(raw_data.Toss_winner == 'Match Abandoned') | (raw_data.Toss_w
inner == 'Recheck')].index)
```

In [12]:

```
refine_data.shape
```

Out[12]:

```
(696, 12)
```

In [13]:

```
refine_data.count()
```

Out[13]:

```
Match_date          696
Match_number        696
```

```
Match_venue       696
Match_time        696
Toss_winner       696
Toss_decision     696
Team1             696
Team1_score       695
Team2             696
Team2_score       694
Winning_team      696
Winning_margin    686
dtype: int64
```

Now we can see that there are two instances where Team2_score is not matching with Team1_score and Also there is a difference in total counts of Winning_margin but this could be due to a tie. Lets Check !

In [14]:

```
refine_data.loc[refine_data.Team2_score.isnull()]
```

Out[14]:

	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2_sc
245	May 21, 2011	68th match	Feroz Shah Kotla, Delhi	night	Delhi	bat	Delhi Daredevils	56/3	Pune Warriors	N
285	Apr 25, 2012	34th match	M Chinnaswamy Stadium, Bangalore	night	Banglore	field	RCB	NaN	Chennai Super Kings	N

So, as we can see this are also matches with no result, these can be dropped too.

In [15]:

```
refine_data = refine_data.drop(refine_data[(refine_data.Team2_score.isnull())].index)
```

In [16]:

```
refine_data.shape
```

Out[16]:

```
(694, 12)
```

In [17]:

```
refine_data.loc[refine_data.Winning_margin.isnull()]
```

Out[17]:

	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2_
68	Apr 23, 2009	10th match	Newlands, Cape Town	day/night	Kolkata	field	Rajasthan Royals	150/6	Kolkata Knight Riders	
133	Mar 21, 2010	16th match	MA Chidambaram Stadium, Chepauk, Chennai	night	Chennai	field	Kings XI Punjab	136/8	Chennai Super Kings	
334	Apr 7, 2013	7th match	Rajiv Gandhi International Stadium, Uppal, Hyd...	night	Banglore	bat	RCB	130/8	Sunrisers Hyderabad	
348	Apr 16, 2013	21st match	M Chinnaswamy Stadium, Bangalore	night	Banglore	field	Delhi Daredevils	152/5	RCB	
422	Apr 29,	19th match	Sheikh Zayed Stadium, Abu	night	Rajasthan	bat	Rajasthan	152/5	Kolkata Knight	

	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2_
481	Apr 21, 2015	18th match	Sardar Patel Stadium, Motera, Ahmedabad	night	Punjab	field	Rajasthan Royals	191/6	Kings XI Punjab	
518	May 17, 2015	55th match	M Chinnaswamy Stadium, Bangalore	day/night	Banglore	field	Delhi Daredevils	187/5	RCB	
618	Apr 29, 2017	35th match	Saurashtra Cricket Association Stadium, Rajkot	night	Gujarat	bat	Gujarat Lions	153/9	Mumbai Indians	

Apart from one match that has Winning_team = 'No result' rest all of the matches are tie, hence its valid to have NaN in Winning_margin for those matches. We shall drop the row that contains Winning_team = 'No result'

In [18]:

```
refine_data = refine_data.drop(refine_data[(refine_data.Winning_team == 'No result')].index).reset_index()
```

In [19]:

```
refine_data.shape
```

Out[19]:

```
(693, 13)
```

Creating uniformity in data

In [20]:

```
tie_ind = refine_data[refine_data['Winning_team'].str.contains("Match tied")==True].Winning_team.index
```

In [21]:

```
for i in tie_ind:
    refine_data.Winning_team[i] = 'Match Tie'
```

In [22]:

```
refine_data.loc[refine_data.Winning_margin.isnull()]
```

Out[22]:

	index	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2
66	68	Apr 23, 2009	10th match	Newlands, Cape Town	day/night	Kolkata	field	Rajasthan Royals	150/6	Kolkata Knight Riders
130	133	Mar 21, 2010	16th match	MA Chidambaram Stadium, Chepauk, Chennai	night	Chennai	field	Kings XI Punjab	136/8	Chennai Super Kings
327	334	Apr 7, 2013	7th match	Rajiv Gandhi International Stadium, Uppal, Hyd...	night	Banglore	bat	RCB	130/8	Sunrisers Hyderabad
341	348	Apr 16, 2013	21st match	M Chinnaswamy Stadium, Bangalore	night	Banglore	field	Delhi Daredevils	152/5	RCB
415	422	Apr 29,	19th match	Sheikh Zayed Stadium, Abu	night	Rajasthan	bat	Rajasthan	152/5	Kolkata Knight

	index	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2
474	481	Apr 21, 2015	18th match	Sardar Patel Stadium, Motera, Ahmedabad	night	Punjab	field	Rajasthan Royals	191/6	Kings XI Punjab
607	618	Apr 29, 2017	35th match	Saurashtra Cricket Association Stadium, Rajkot	night	Gujarat	bat	Gujarat Lions	153/9	Mumbai Indians

Applying function to Replace different team names by state or city name in other columns as well

In [23]:

```
New = []
for i in refine_data['Team1']:
    New_item = CleanTeamName(i)
    New.append(New_item)
refine_data['Team1'] = New

New = []
for i in refine_data['Team2']:
    New_item = CleanTeamName(i)
    New.append(New_item)
refine_data['Team2'] = New

New = []
for i in refine_data['Winning_team']:
    New_item = CleanTeamName(i)
    New.append(New_item)
refine_data['Winning_team'] = New
```

In [24]:

```
refine_data.head(2)
```

Out[24]:

	index	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2
0	0	Apr 18, 2008	1st match	M Chinnaswamy Stadium, Bangalore	night	Banglore	field	Kolkata	222/3	Banglore	
1	1	Apr 19, 2008	2nd match	Punjab Cricket Association Stadium, Mohali, Ch...	day/night	Chennai	bat	Chennai	240/5	Punjab	

Cleaning Date format :

We need only "Year of the Match" and given date format will not help.

In [25]:

```
for i in range(len(refine_data)):
    refine_data.Match_date[i] = refine_data.Match_date[i][-4:]
```

In [114]:

```
refine_data.head(2)
```

Out[114]:

	index	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2

	index	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2
0	0	2008	1st match	M Chinnaswamy Stadium, Bangalore	night	Banglore	field	Kolkata	222/3	Banglore	
1	1	2008	2nd match	Punjab Cricket Association Stadium, Mohali, Ch...	day/night	Chennai	bat	Chennai	240/5	Punjab	

Winning Margin Segmentation:

We need to different columns to identify if we have won by a margin of runs or wickets

In [27]:

```
wicket_ind = refine_data[refine_data['Winning_margin'].str.contains("wickets")==True].Winning_margi
n.index
run_ind = refine_data[refine_data['Winning_margin'].str.contains("runs")==True].Winning_margin.inde
x
```

In [28]:

```
refine_data['Winning_margin_wicket']=np.nan
refine_data['Winning_margin_runs']=np.nan
```

In [29]:

```
for i in wicket_ind:
    word = refine_data.Winning_margin[i].split()
    refine_data.Winning_margin_wicket[i] = word[0]

for i in run_ind:
    word = refine_data.Winning_margin[i].split()
    refine_data.Winning_margin_runs[i] = word[0]
```

Team Scores Slicing based on runs and wickets

In [30]:

```
refine_data['Team1_runs'] = ' '
refine_data['Team1_wickets'] = ' '
refine_data['Team2_runs'] = ' '
refine_data['Team2_wickets'] = ' '
```

In [31]:

```
Team1_run_ind = refine_data[refine_data['Team1_score'].str.contains("/")==False].Team1_score.index
Team1_wic_ind = refine_data[refine_data['Team1_score'].str.contains("/")==True].Team1_score.index
Team2_run_ind = refine_data[refine_data['Team2_score'].str.contains("/")==False].Team2_score.index
Team2_wic_ind = refine_data[refine_data['Team2_score'].str.contains("/")==True].Team2_score.index
```

In [32]:

```
for i in Team1_wic_ind:
    refine_data.Team1_wickets[i]= refine_data.Team1_score[i][-1:]
    refine_data.Team1_runs[i] = refine_data.Team1_score[i][0:-2]

for i in Team1_run_ind:
    refine_data.Team1_runs[i] = refine_data.Team1_score[i]

for i in Team2_wic_ind:
    refine_data.Team2_wickets[i]= refine_data.Team2_score[i][-1:]
    refine_data.Team2_runs[i] = refine_data.Team2_score[i][0:-2]

for i in Team2_run_ind:
    refine_data.Team2_runs[i] = refine_data.Team2_score[i]
```

In [33]:

```
refine_data.head(2)
```

Out[33]:

	index	Match_date	Match_number	Match_venue	Match_time	Toss_winner	Toss_decision	Team1	Team1_score	Team2	Team2
0	0	2008	1st match	M Chinnaswamy Stadium, Bangalore	night	Banglore	field	Kolkata	222/3	Banglore	
1	1	2008	2nd match	Punjab Cricket Association Stadium, Mohali, Ch...	day/night	Chennai	bat	Chennai	240/5	Punjab	

Questions this data can answer: Visualizations

[1] Which team had scored the highest runs in total

In [34]:

```
df1 = pd.DataFrame()
df2 = pd.DataFrame()
df1['Team'] = refine_data['Team1']
df1['Runs'] = refine_data.Team1_runs.astype(int)
df2['Team'] = refine_data['Team2']
df2['Runs'] = refine_data.Team2_runs.astype(int)
```

In [35]:

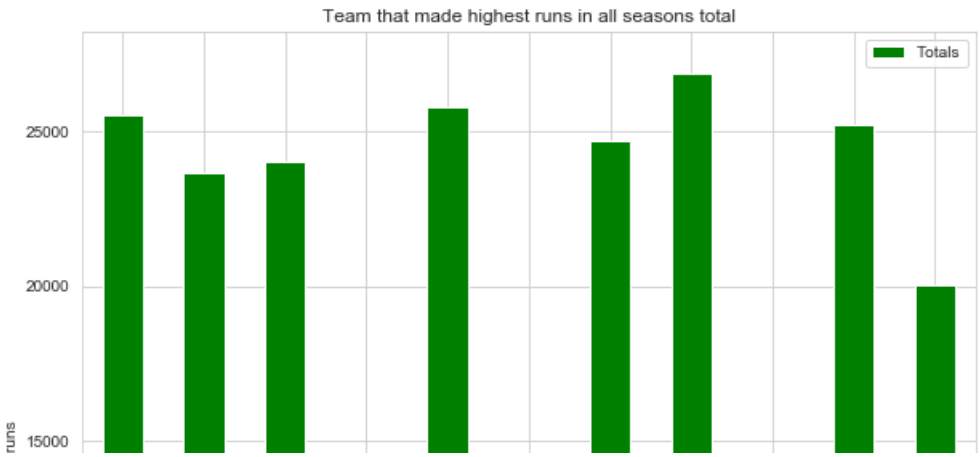
```
df1 = df1.groupby('Team').Runs.agg('sum').reset_index()
df2 = df2.groupby('Team').Runs.agg('sum').reset_index()
df1.columns = ['Team','Totals']
df2.columns = ['Team','Totals']
```

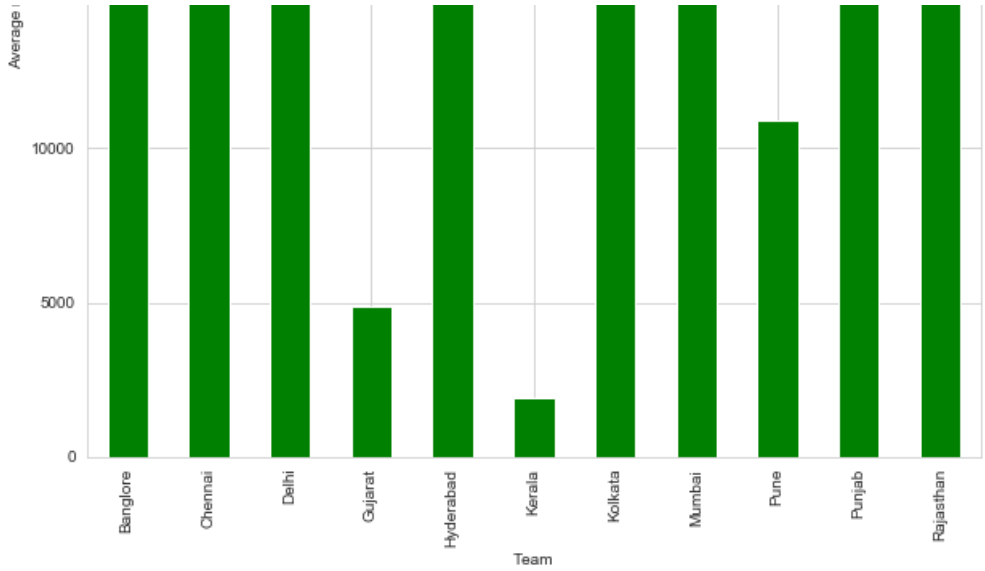
In [36]:

```
df3 = df1.groupby('Team').sum().add(df2.groupby('Team').sum(), fill_value=0).reset_index()
df3.set_index("Team",drop=True,inplace=True)
```

In [37]:

```
df3.plot.bar(color='green') # Use the plot.bar method on the counts data frame
plt.title('Team that made highest runs in all seasons total') # Give the plot a main title
plt.xlabel('Team') # Set text for the x axis
plt.ylabel('Average runs')# Set text for y axis
plt.show()
```





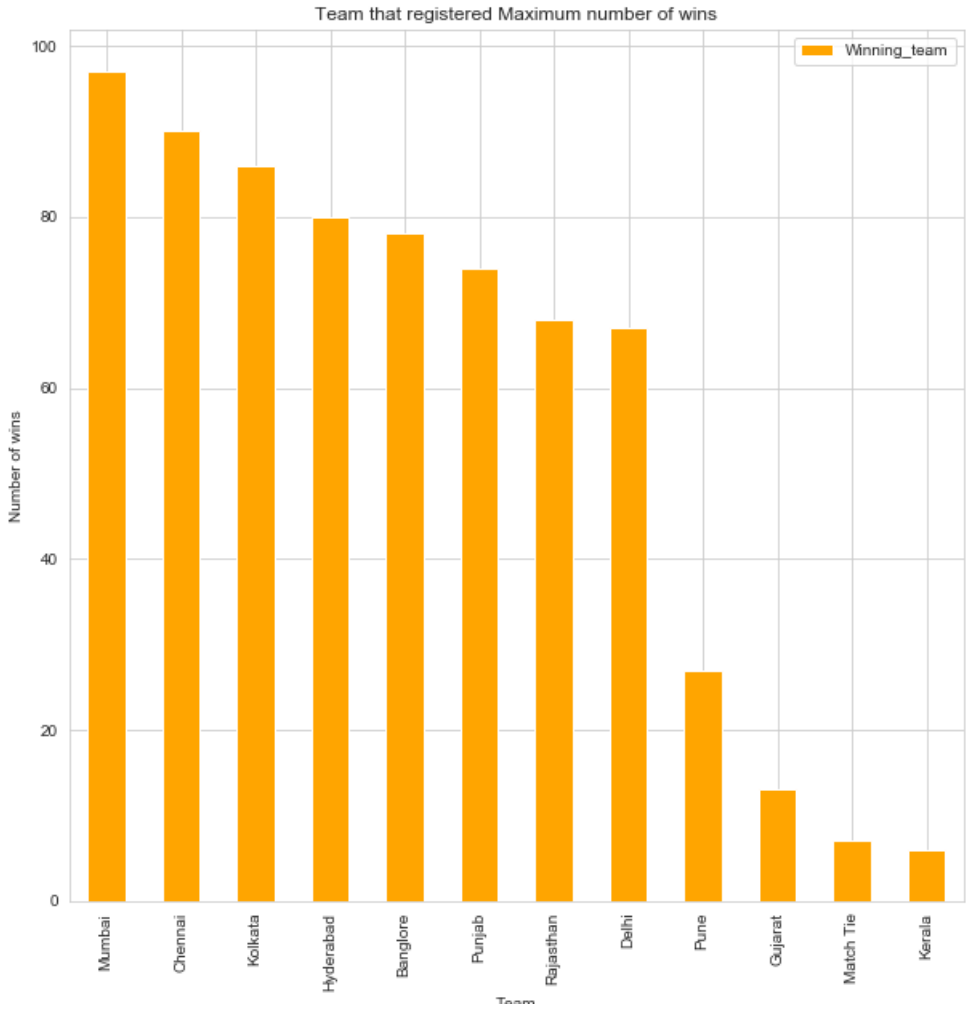
[2] Which Team has won maximum number of matches

In [38]:

```
Win_stats = pd.DataFrame(refine_data['Winning_team'].value_counts())
```

In [39]:

```
Win_stats.plot.bar(color='orange') # Use the plot.bar method on the counts data frame
plt.title('Team that registered Maximum number of wins') # Give the plot a main title
plt.xlabel('Team') # Set text for the x axis
plt.ylabel('Number of wins')# Set text for y axis
plt.show()
```



[3] Which stadium hosted maximum number of matches

In [40]:

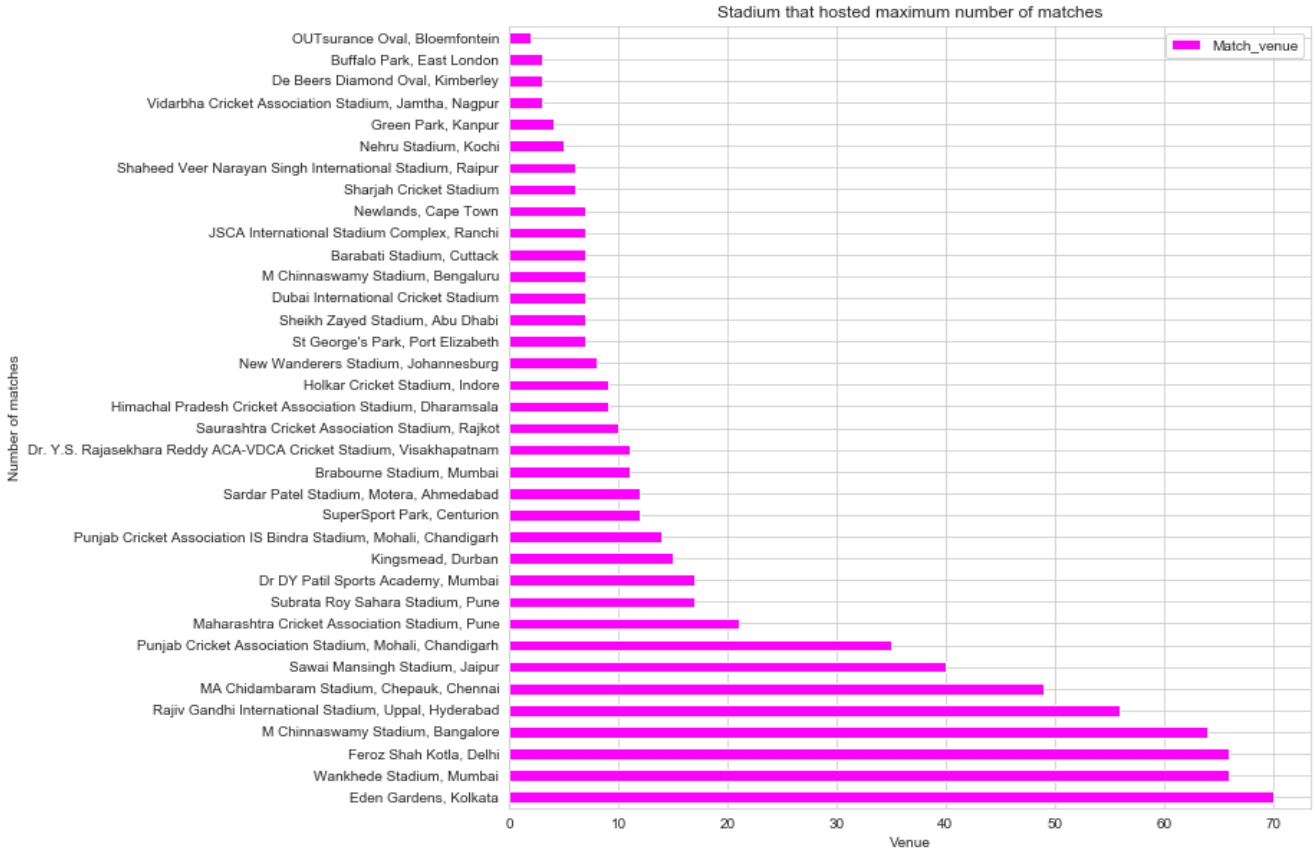
```
venue_stats = pd.DataFrame(refine_data['Match_venue'].value_counts())
venue_stats.head(2)
```

Out[40]:

	Match_venue
Eden Gardens, Kolkata	70
Wankhede Stadium, Mumbai	66

In [41]:

```
venue_stats.plot.barh(align='center',color='magenta') # Use the plot.bar method on the counts data frame
plt.title('Stadium that hosted maximum number of matches') # Give the plot a main title
plt.xlabel('Venue') # Set text for the x axis
plt.ylabel('Number of matches')# Set text for y axis
plt.show()
```



[4] What helps to win the match, to bat first or to field first?

In [42]:

```
Toss_stat = refine_data[['Toss_winner','Toss_decision','Winning_team']]
Toss_stat.head(2)
```

Out[42]:

	Toss_winner	Toss_decision	Winning_team

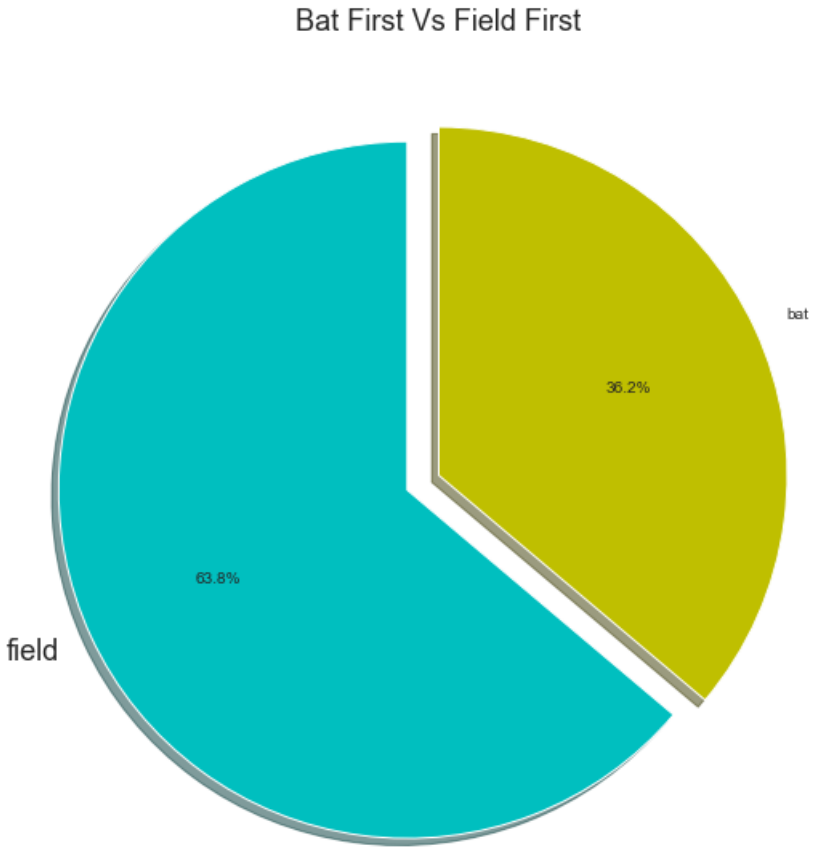
	Toss_winner	Toss_decision	Winning_team
0	Bangiore	field	Kolkata
1	Chennai	bat	Chennai

In [43]:

```
Toss_stat = Toss_stat.drop(Toss_stat[(Toss_stat.Winning_team != Toss_stat.Toss_winner)].index)
Toss_stats = pd.DataFrame(Toss_stat['Toss_decision'].value_counts()).reset_index()
Toss_stats.columns = ['Toss','counts']
```

In [44]:

```
sizes = Toss_stats['counts']
labels = Toss_stats['Toss']
colors = ['c','y']
patches,texts,autotexts = plt.pie(sizes,labels=labels,colors=colors, shadow=True, autopct='%1.1f%%'
,startangle=90,explode =(0.1,0))
texts[0].set_fontsize(18)
plt.title('Bat First Vs Field First',fontsize=18)
plt.show()
```



[5] Individual performance of team on their toss decision

In [45]:

```
Team_toss = refine_data[['Toss_winner','Toss_decision','Winning_team']]
good_toss = Team_toss.drop(Team_toss[(Team_toss.Winning_team != Team_toss.Toss_winner)].index)
bad_toss = Team_toss.drop(Team_toss[(Team_toss.Winning_team == Team_toss.Toss_winner)].index)
```

In [46]:

```
good_toss = good_toss.drop('Winning_team',1)
bad_toss = bad_toss.drop('Winning_team',1)
good = pd.DataFrame(good_toss['Toss_winner'].value_counts()).reset_index()
good.columns = ['Team','counts']
bad = pd.DataFrame(bad_toss['Toss_winner'].value_counts()).reset_index()
```

```
bad = pd.DataFrame(bad_toss['Toss_Winner'].value_counts()).reset_index()
bad.columns = ['Team','counts']
print(good.head(2),bad.head(2))
```

```
      Team  counts
0   Mumbai      50
1  Chennai      50         Team  counts
0  Hyderabad      45
1      Delhi      44
```

In [47]:

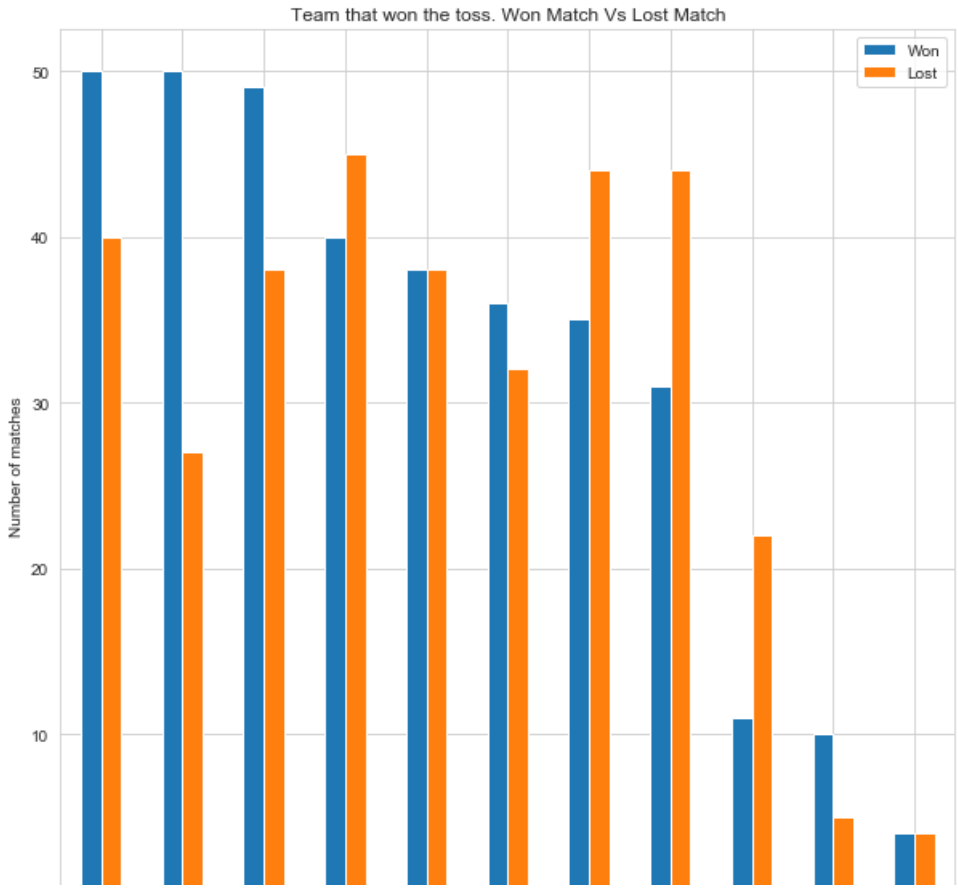
```
good.set_index("Team",drop=True,inplace=True)
bad.set_index("Team",drop=True,inplace=True)
dec = pd.merge(good,bad,on='Team').reset_index()
dec.columns= ['Team','Won','Lost']
dec.set_index("Team",drop=True,inplace=True)
dec.head()
```

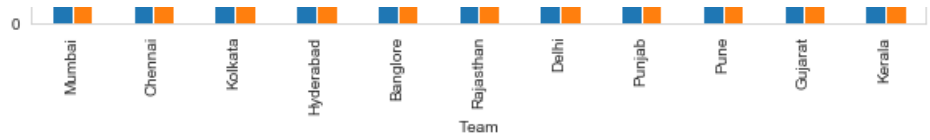
Out[47]:

Team	Won	Lost
Mumbai	50	40
Chennai	50	27
Kolkata	49	38
Hyderabad	40	45
Banglore	38	38

In [48]:

```
dec.plot.bar() # Use the plot.bar method on the counts data frame
plt.title('Team that won the toss. Won Match Vs Lost Match') # Give the plot a main title
plt.xlabel('Team') # Set text for the x axis
plt.ylabel('Number of matches')# Set text for y axis
plt.show()
```





[6] Team to reach maximum number of finals

In [49]:

```
reach_final = refine_data.drop(refine_data[(refine_data['Match_number'] != 'Final')].index)
```

In [50]:

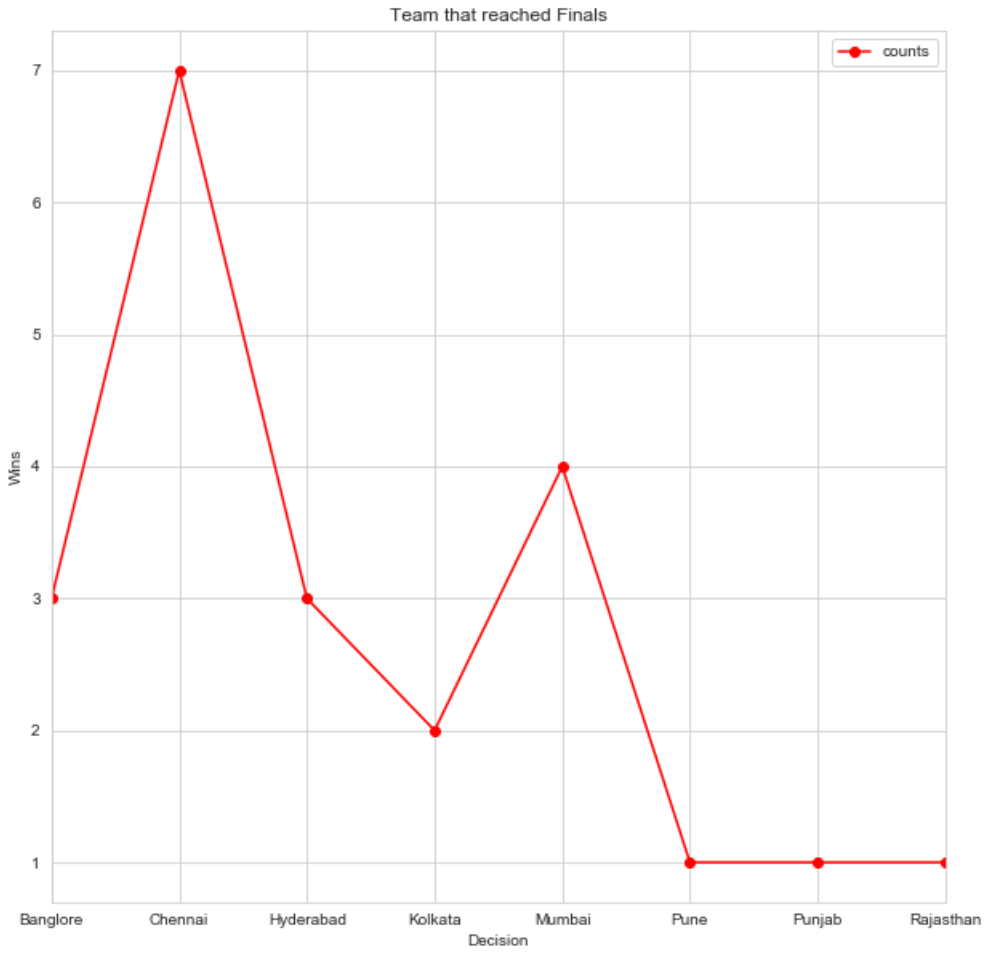
```
rf1 = pd.DataFrame(reach_final['Team1'].value_counts()).reset_index()
rf1.columns= ['Team','counts']
rf2 =  pd.DataFrame(reach_final['Team2'].value_counts()).reset_index()
rf2.columns= ['Team','counts']
```

In [51]:

```
frames = [rf1,rf2]
rf= pd.DataFrame(pd.concat(frames).groupby('Team').agg('sum')).reset_index()
rf.columns=['Team','counts']
```

In [52]:

```
rf.plot.line(x='Team', y='counts', linestyle='-', marker='o',color='red')
plt.title('Team that reached Finals') # Give the plot a main title
plt.xlabel('Decision') # Set text for the x axis
plt.ylabel('Wins')# Set text for y axis
plt.show()
```



[7] Winning percentages in home ground

In [53]:

```
## Function to map state or city name to the Venue column
def MapVenue(x):
    if x == 'M Chinnaswamy Stadium, Bangalore' or x == 'M Chinnaswamy Stadium, Bengaluru':
        venue = 'Banglore'
    elif x == 'Punjab Cricket Association Stadium, Mohali, Chandigarh' or x == 'Punjab Cricket
Association IS Bindra Stadium, Mohali, Chandigarh':
        venue = 'Punjab'
    elif x == 'Feroz Shah Kotla, Delhi':
        venue = 'Delhi'
    elif x == 'Eden Gardens, Kolkata':
        venue = 'Kolkata'
    elif x == 'Wankhede Stadium, Mumbai'or x == 'Dr DY Patil Sports Academy, Mumbai' or x == 'Brabo
urne Stadium, Mumbai':
        venue = 'Mumbai'
    elif x == 'Sawai Mansingh Stadium, Jaipur':
        venue = 'Rajasthan'
    elif x == 'Rajiv Gandhi International Stadium, Uppal, Hyderabad':
        venue = 'Rajasthan'
    elif x == 'MA Chidambaram Stadium, Chepauk, Chennai':
        venue = 'Chennai'
    elif x == 'Sardar Patel Stadium, Motera, Ahmedabad'or x == 'Saurashtra Cricket Association
Stadium, Rajkot':
        venue = 'Gujarat'
    elif x == 'Nehru Stadium, Kochi':
        venue = 'Kerala'
    elif x == 'Subrata Roy Sahara Stadium, Pune' or x == 'Maharashtra Cricket Association Stadium,
Pune':
        venue = 'Pune'
    else:
        venue = 'Not Home ground for any Team'

    return venue
```

In [54]:

```
Venue_stat = refine_data[['Match_venue','Winning_team']]
```

In [55]:

```
New = []
for i in Venue_stat['Match_venue']:
    New_venue = MapVenue(i)
    New.append(New_venue)
Venue_stat['Match_venue'] = New
Venue_stat.head()
```

Out[55]:

	Match_venue	Winning_team
0	Banglore	Kolkata
1	Punjab	Chennai
2	Delhi	Delhi
3	Kolkata	Kolkata
4	Mumbai	Banglore

In [56]:

```
Venue_stat.shape
```

Out[56]:

```
(693, 2)
```

In [57]:

```
Venue_stat = Venue_stat.drop(Venue_stat[(Venue_stat['Match_venue'] == 'Not Home ground for any Tea
m')].index)
```

In [58]:

```
good_at_hg = Venue_stat.drop(Venue_stat[(Venue_stat['Match_venue'] != Venue_stat['Winning_team'])]
.index)
bad_at_hg = Venue_stat.drop(Venue_stat[(Venue_stat['Match_venue'] == Venue_stat['Winning_team'])].
index)
```

In [59]:

```
good = pd.DataFrame(good_at_hg['Winning_team'].value_counts()).reset_index()
good.columns = ['Match_venue','Winning_team']
bad = pd.DataFrame(bad_at_hg['Winning_team'].value_counts()).reset_index()
bad.columns = ['Match_venue','Winning_team']
hg_per = pd.merge(good,bad,on="Match_venue")
hg_per.columns = ['Home_Ground', 'Won_Match','Lost_Match']
hg_per
```

Out[59]:

	Home_Ground	Won_Match	Lost_Match
0	Mumbai	48	40
1	Kolkata	42	34
2	Chennai	34	39
3	Banglore	33	31
4	Rajasthan	31	28
5	Delhi	27	21
6	Punjab	25	26
7	Pune	9	15
8	Gujarat	3	8
9	Kerala	2	3

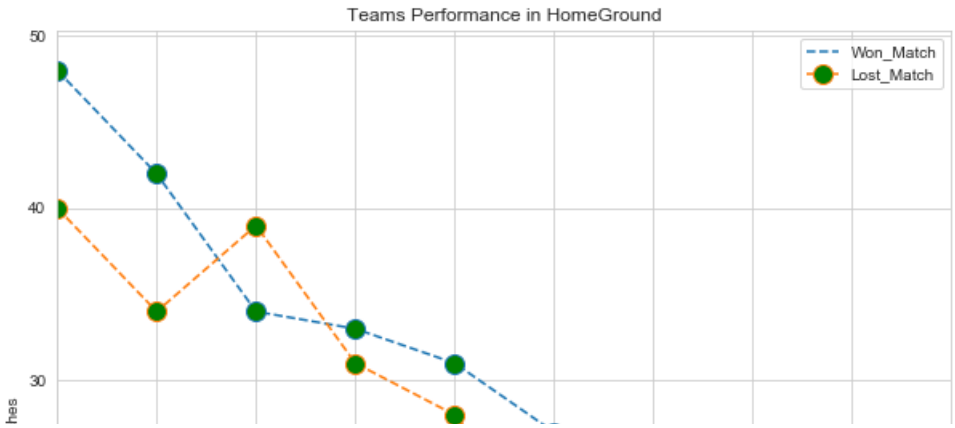
In [ ]:

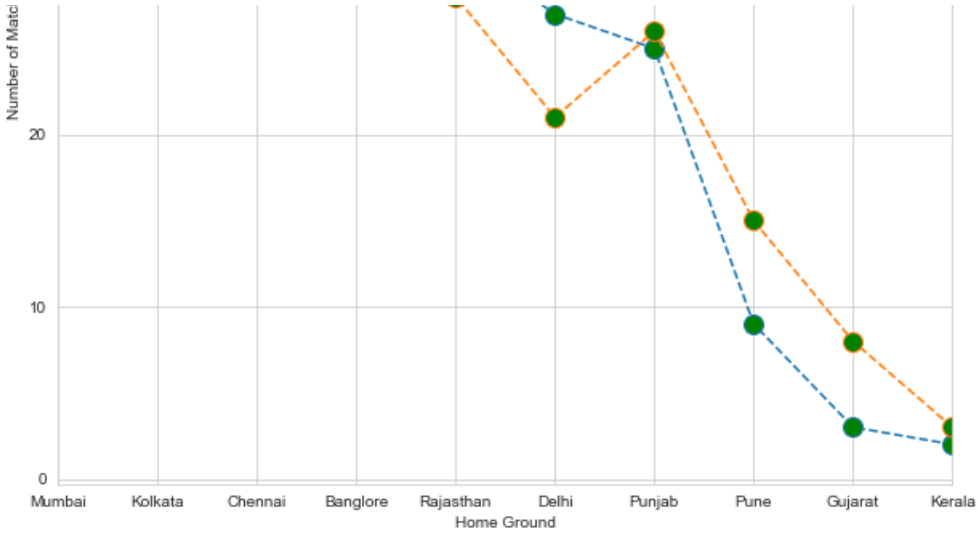
```

```

In [60]:

```
fig, ax = plt.subplots()
for cols in ['Won_Match', 'Lost_Match']:
    hg_per.plot('Home_Ground', cols,ax=ax, linestyle='--', marker='o', markerfacecolor='green', mar
kersize=12)
plt.title('Teams Performance in HomeGround') # Give the plot a main title
plt.xlabel('Home Ground') # Set text for the x axis
plt.ylabel('Number of Matches')# Set text for y axis
plt.show()
```





[8] Teams out performed with highest margin - RUNS

In [61]:

```
margin = pd.DataFrame(refine_data[['Winning_team','Winning_margin_runs']])
```

In [62]:

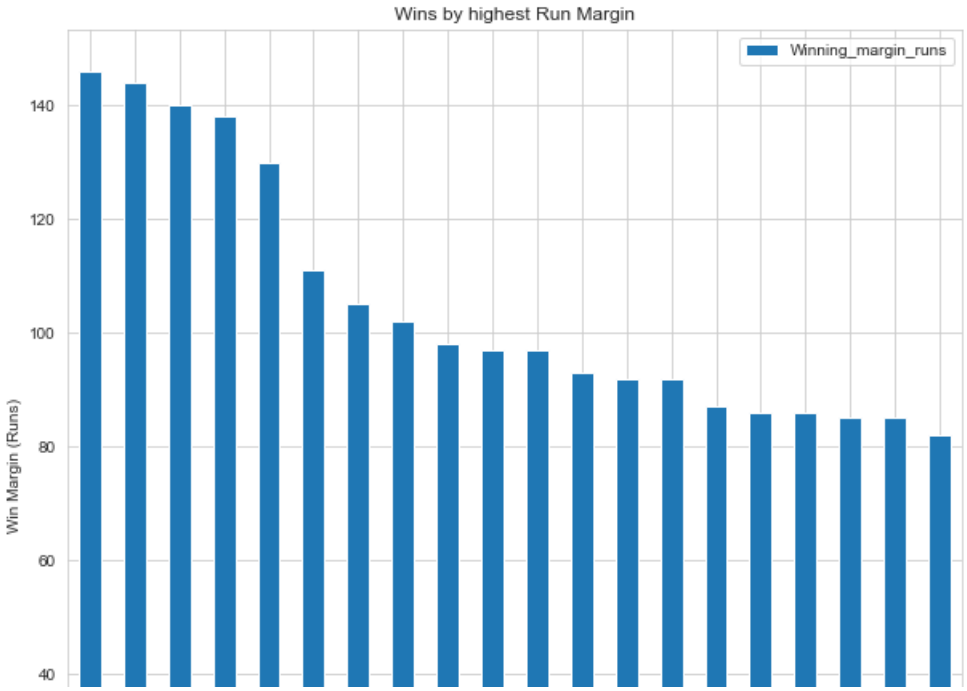
```
margin = margin.dropna()
margin['Winning_margin_runs'] = pd.to_numeric(margin['Winning_margin_runs'])
margin_sort = margin.sort_values('Winning_margin_runs',ascending=False)
```

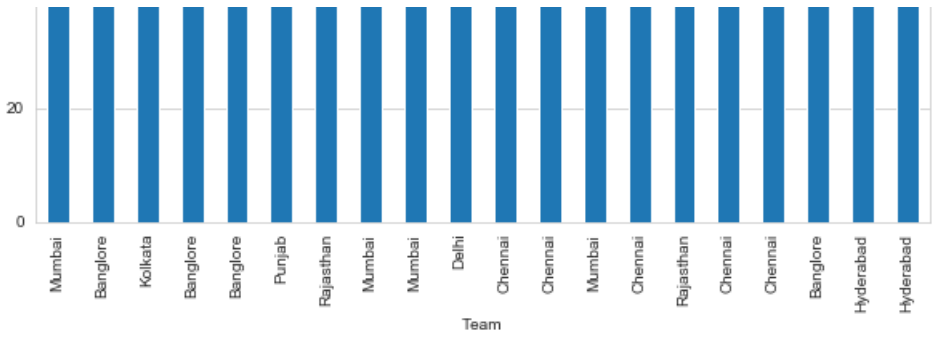
In [63]:

```
best_run_margin =pd.DataFrame(margin_sort.head(20))
best_run_margin.set_index('Winning_team',drop=True,inplace=True)
```

In [64]:

```
best_run_margin.plot.bar()
plt.title('Wins by highest Run Margin')
plt.xlabel('Team')
plt.ylabel('Win Margin (Runs)')
plt.show()
```





[9] Teams out performed with highest margin - Wickets

In [65]:

```
margin = pd.DataFrame(refine_data[['Winning_team','Winning_margin_wicket']])
```

In [66]:

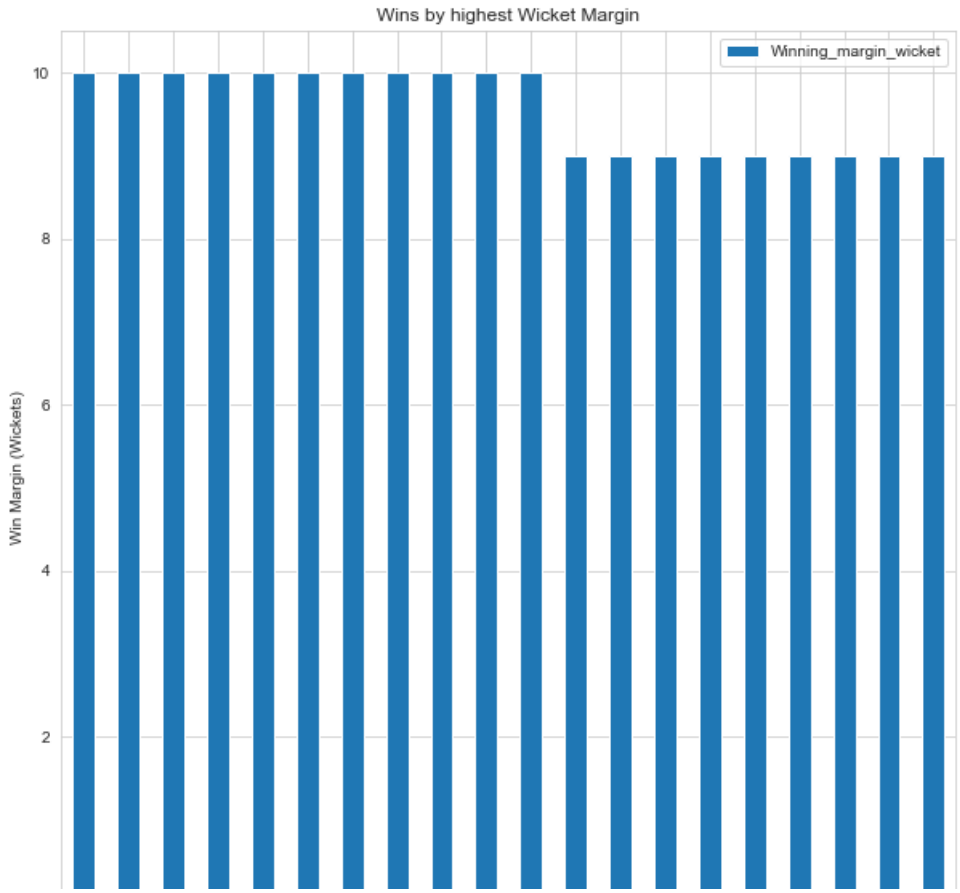
```
margin = margin.dropna()
margin['Winning_margin_wicket'] = pd.to_numeric(margin['Winning_margin_wicket'])
margin_sort = margin.sort_values('Winning_margin_wicket',ascending=False)
```

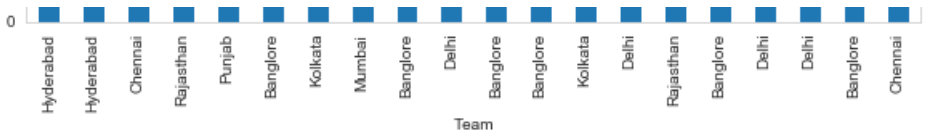
In [67]:

```
best_run_margin =pd.DataFrame(margin_sort.head(20))
best_run_margin.set_index('Winning_team',drop=True,inplace=True)
```

In [68]:

```
best_run_margin.plot.bar()
plt.title('Wins by highest Wicket Margin')
plt.xlabel('Team')
plt.ylabel('Win Margin (Wickets)')
plt.show()
```





[11] Performance of Top 5 teams through out these years (Based on number of wins)

In [94]:

```
yearly_stat = pd.DataFrame(refine_data[['Match_date','Winning_team']])
```

In [96]:

```
df_ban = yearly_stat[yearly_stat.Winning_team == 'Banglore']
ban_df = pd.DataFrame(df_ban['Match_date'].value_counts()).reset_index()
ban_df.columns = ['year','counts']
ban = ban_df.sort_values('year',ascending=True)
```

In [100]:

```
df_chen = yearly_stat[yearly_stat.Winning_team == 'Chennai']
chen_df = pd.DataFrame(df_chen['Match_date'].value_counts()).reset_index()
chen_df.columns = ['year','counts']
chen = chen_df.sort_values('year',ascending=True)
```

In [102]:

```
df_mum = yearly_stat[yearly_stat.Winning_team == 'Mumbai']
mum_df = pd.DataFrame(df_mum['Match_date'].value_counts()).reset_index()
mum_df.columns = ['year','counts']
mum = mum_df.sort_values('year',ascending=True)
```

In [104]:

```
df_kol = yearly_stat[yearly_stat.Winning_team == 'Kolkata']
kol_df = pd.DataFrame(df_kol['Match_date'].value_counts()).reset_index()
kol_df.columns = ['year','counts']
kol = kol_df.sort_values('year',ascending=True)
```

In [106]:

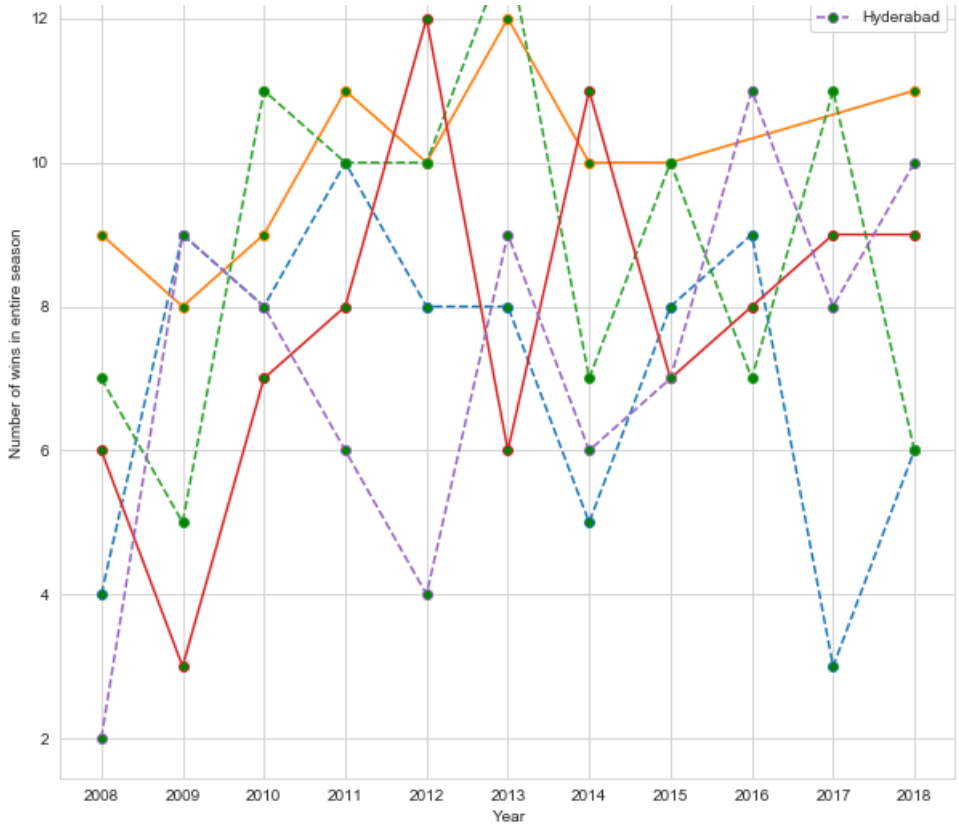
```
df_hyd = yearly_stat[yearly_stat.Winning_team == 'Hyderabad']
hyd_df = pd.DataFrame(df_hyd['Match_date'].value_counts()).reset_index()
hyd_df.columns = ['year','counts']
hyd = hyd_df.sort_values('year',ascending=True)
```

In [112]:

```
plt.plot(ban.year,ban.counts,linestyle='--', marker='o', markerfacecolor='green',label='Banglore')
plt.plot(chen.year,chen.counts,linestyle='-', marker='o', markerfacecolor='green',label='Chennai')
plt.plot(mum.year,mum.counts,linestyle='--', marker='o', markerfacecolor='green',label='Mumbai')
plt.plot(kol.year,kol.counts,linestyle='-', marker='o', markerfacecolor='green',label='Kolkata')
plt.plot(hyd.year,hyd.counts,linestyle='--', marker='o', markerfacecolor='green',label='Hyderabad')
plt.title('Performance of TOP 5 teams over the years (Total number of wins per year)')
plt.xlabel('Year')
plt.ylabel('Number of wins in entire season')
plt.legend()
```

Out[112]:

```
<matplotlib.legend.Legend at 0x234ac8edef0>
```


Hyderabad
Number of wins in entire season
Year
12
10
8
6
4
2
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018

In [ ]: