IPL Data analysis and visualization using Python

Project by : Sanketh S

Data courtesy : Kaggle

Importing packages

In [1]:
```
import pandas as pd
import plotly.express as px
from plotly.subplots import make_subplots as ms
import plotly.graph_objs as go
```

Reading data

In [2]:
```
deliveries =pd.read_csv('C:/Users/ACER/Downloads/IPL Ball-by-Ball 2008-2020.csv')
matches=pd.read_csv('C:/Users/ACER/Downloads/IPL Matches 2008-2020.csv')
```

cleaning data redundency

In [3]:
```
deliveries.at[deliveries['batting_team']=='Rising Pune Supergiant','batting_team']='Rising Pune Supergiants'
deliveries.at[deliveries['bowling_team']=='Rising Pune Supergiant','bowling_team']='Rising Pune Supergiants'

deliveries.at[deliveries['batting_team']=='Delhi Daredevils','batting_team']='Delhi Capitals'
deliveries.at[deliveries['bowling_team']=='Delhi Daredevils','bowling_team']='Delhi Capitals'

deliveries.at[deliveries['batting_team']=='Deccan Chargers','batting_team']='Sunrisers Hyderabad'
deliveries.at[deliveries['bowling_team']=='Deccan Chargers','bowling_team']='Sunrisers Hyderabad'
```

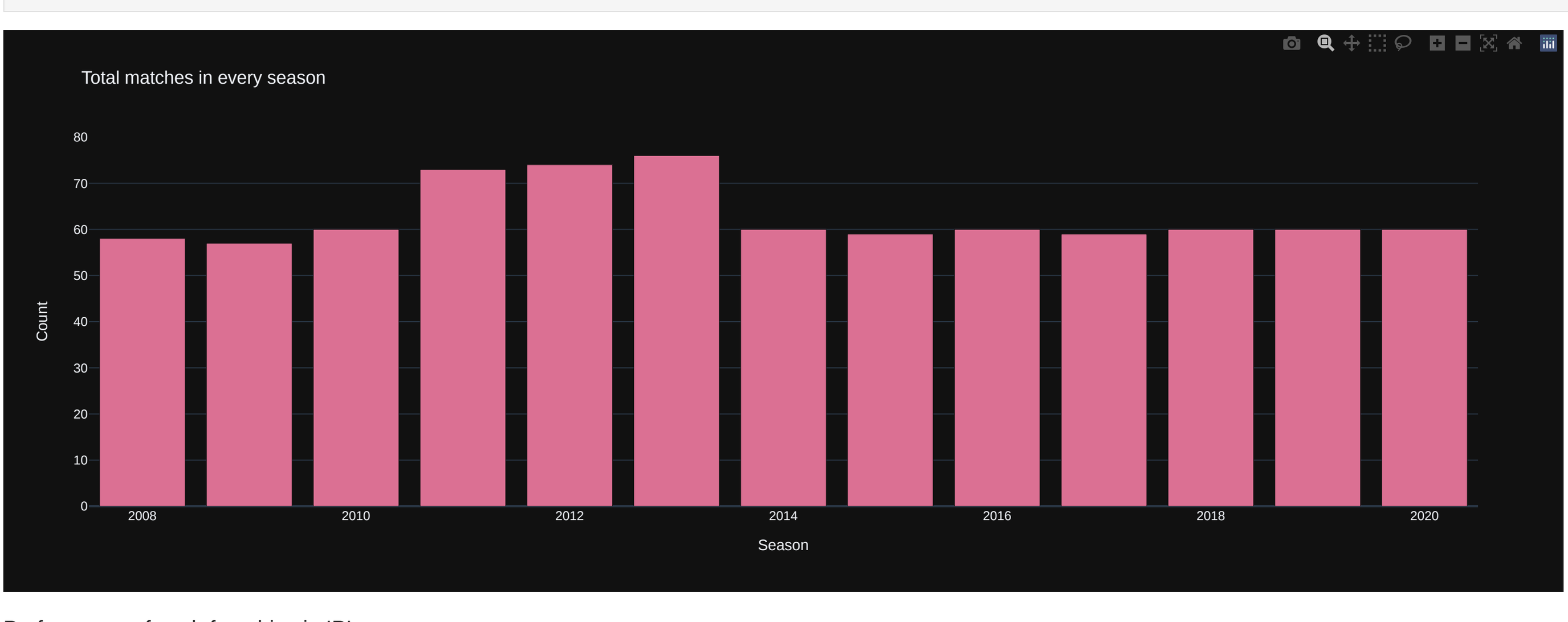
In [4]:
```
matches.at[matches['team1']=='Rising Pune Supergiant','team1']='Rising Pune Supergiants'
matches.at[matches['team2']=='Rising Pune Supergiant','team2']='Rising Pune Supergaints'
matches.at[matches['winner']=='Rising Pune Supergiant','winner']='Rising Pune Supergiants'
matches.at[matches['team1']=='Delhi Daredevils','team1']='Delhi Capitals'
matches.at[matches['team2']=='Delhi Daredevils','team2']='Delhi Capitals'
matches.at[matches['winner']=='Delhi Daredevils','winner']='Delhi Capitals'
matches.at[matches['team1']=='Deccan Chargers','team1']='Sunrisers Hyderabad'
matches.at[matches['team2']=='Deccan Chargers','team2']='Sunrisers Hyderabad'
matches.at[matches['winner']=='Deccan Chargers','winner']='Sunrisers Hyderabad'
```

In [5]:
```
x=['Royal Challengers Bangalore' ,'Kings XI Punjab'  ,  'Mumbai Indians' , 'Rajasthan Royals', 'Chennai Super Kings', 'Gujarat Lions', 'Pune Warriors' ,'Delhi Capitals',
   'Kochi Tuskers Kerala','Sunrisers Hyderabad','Kolkata Knight Riders','Rising Pune Supergiants']
y=['RCB','KXIP','MI','RR','CSK','GL','PW','DC','KTK','SRH','KKR','RPS']
matches.replace(x,y,inplace = True)
deliveries.replace(x,y,inplace = True)
```

In [6]:
```
matches["Season"]=matches["date"].str[:4].astype(int)
```

Total match counts in every season

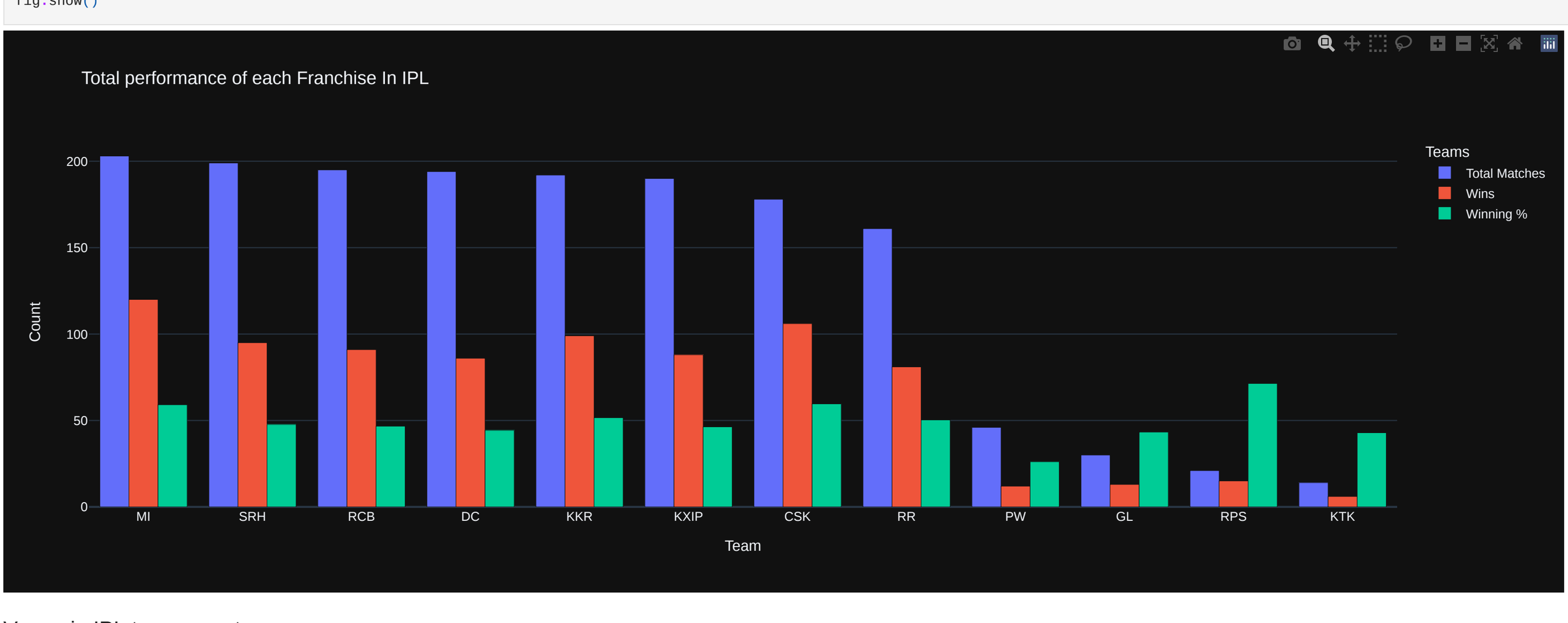
In [7]:
```
df1=matches["Season"].value_counts().rename_axis("Season").reset_index(name="Count")
fig=px.bar(df1,x="Season",y="Count",template="plotly_dark",title="Total matches in every season",
           color_discrete_sequence=["palevioletred"])
fig.show()
```



Performance of each franchise in IPL

In [8]:
```
df_team1=matches["team1"].value_counts().rename_axis("Team").reset_index(name="Total Matches")
df_team2=matches["team2"].value_counts().rename_axis("Team").reset_index(name="Total Matches")
new=pd.merge(df_team1,df_team2, on ="Team")
new["Total Matches"]=new["Total Matches_x"]+new["Total Matches_y"]
total_matches=new[["Team","Total Matches"]].sort_values(by="Total Matches",ascending = False)
wins=matches["winner"].value_counts().rename_axis("Team").reset_index(name="Wins")
df=pd.merge(total_matches,wins,on="Team")
df["Winning %"]=round((df["Wins"]/df["Total Matches"])*100,1)
mdf=df.melt(id_vars ="Team",value_vars=["Total Matches","Wins","Winning %"],var_name="Teams",value_name="Count")
```

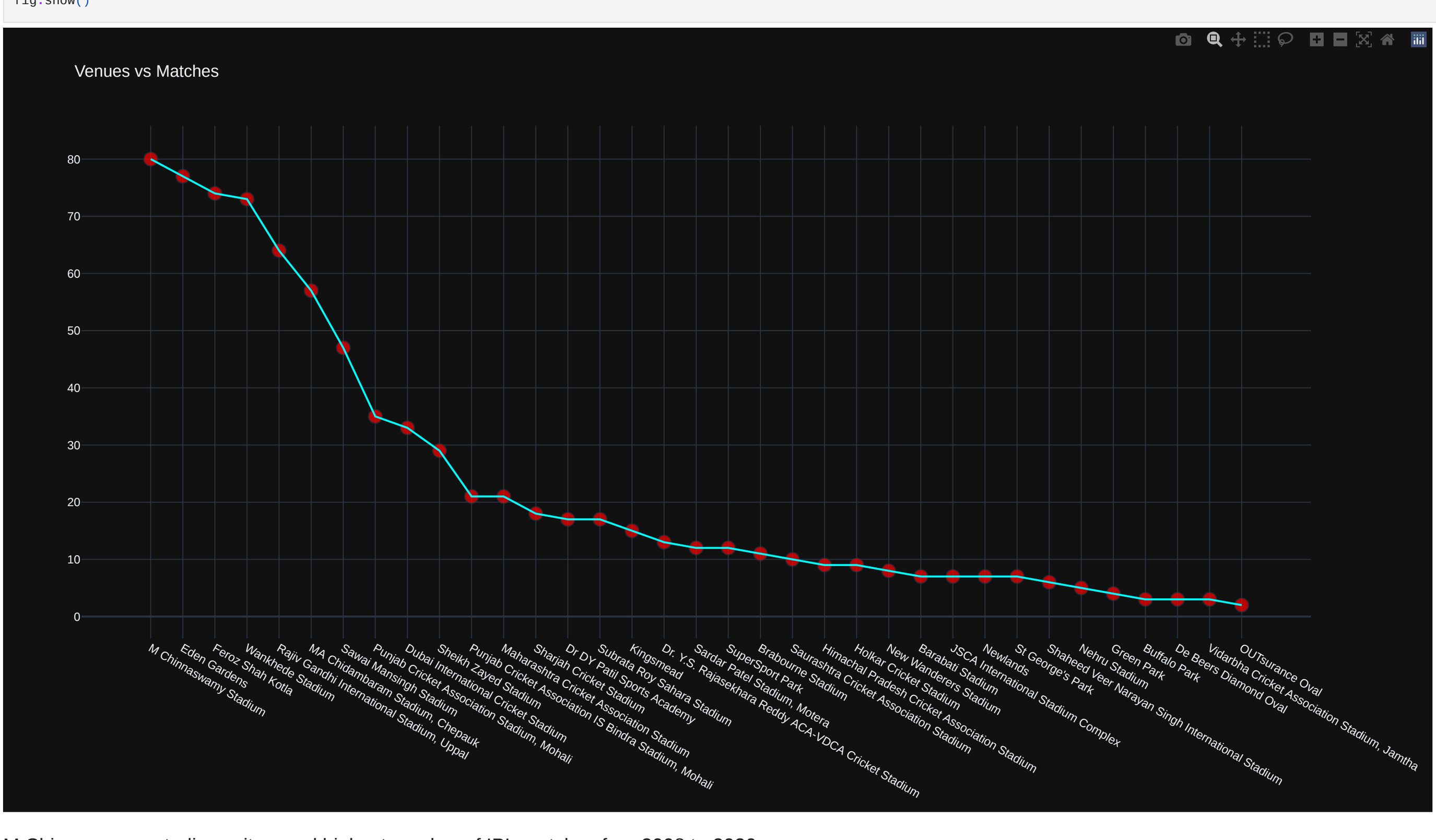
In [9]:
```
fig=px.bar(mdf,x="Team",y="Count",color="Teams",title= "Total performance of each Franchise In IPL",
          template="plotly_dark",barmode="group")
fig.show()
```



Venue in IPL tournamnets

In [10]:
```
matches.at[matches['venue']=='M.Chinnaswamy Stadium','venue']='M Chinnaswamy Stadium'
```

In [11]:
```
venues=matches["venue"].value_counts().rename_axis("Venues").reset_index(name="Count")
venues["size"]=1
fig1=px.scatter(venues,x="Venues",y="Count",template="plotly_dark",color_discrete_sequence=["red"],
                size='size',size_max=10)
fig2=px.line(venues,x="Venues",y="Count",template="plotly_dark",color_discrete_sequence=["cyan"])
fig=go.Figure(fig1["data"]+fig2["data"])
fig.update_layout(template="plotly_dark",height=800,title="Venues vs Matches")
fig.show()
```

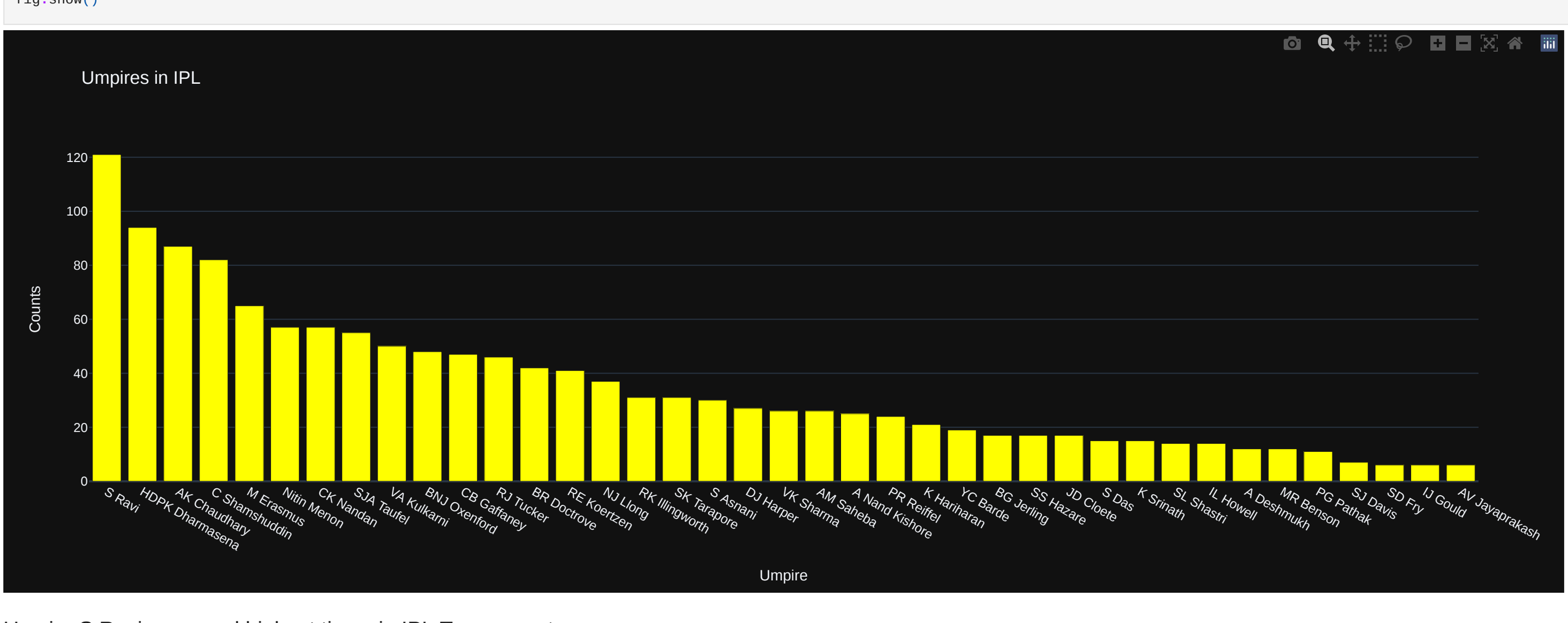


M.Chinnaswamy stadium witnessed highest number of IPL matches from2008 to 2020

Umpires appeard for matches

In [12]:
```
u1=matches["umpire1"].value_counts().rename_axis("Umpire").reset_index(name="Counts")
u2=matches["umpire2"].value_counts().rename_axis("Umpire").reset_index(name="Counts")
mumpire=pd.merge(u1,u2, on ="Umpire")
mumpire["Counts"]=mumpire["Counts_x"]+mumpire["Counts_y"]
Umpires=mumpire[["Umpire","Counts"]].sort_values(by="Counts",ascending = False)
```

In [13]:
```
fig=px.bar(Umpires,x="Umpire",y="Counts",color_discrete_sequence=["yellow"],title= "Umpires in IPL",
          template="plotly_dark")
fig.show()
```

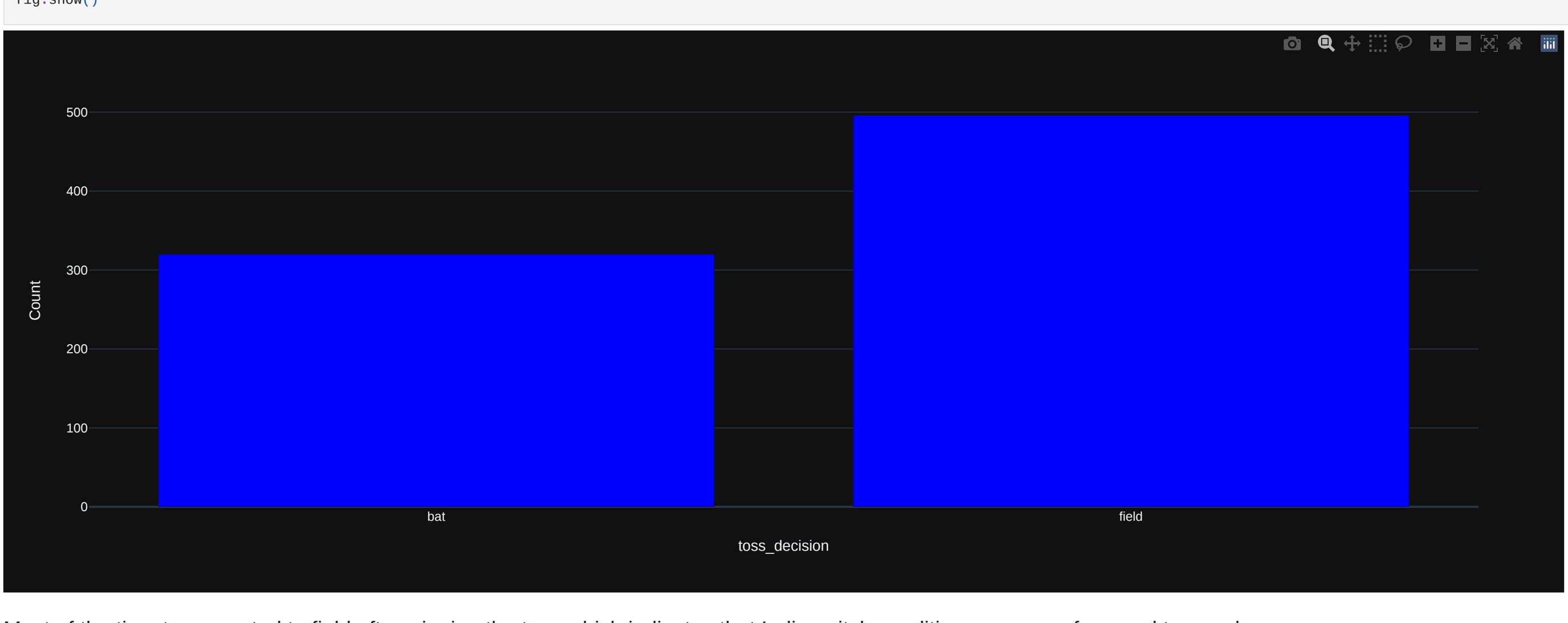


Umpire S Ravi appeared highest times in IPL Tournaments

Team decision after winning toss

In [14]:
```
field = matches[matches["toss_decision"] == "field"]["toss_decision"].value_counts()
bat = matches[matches["toss_decision"] == "bat"]["toss_decision"].value_counts()
toss = pd.DataFrame({"bat":list(bat), "field":list(field)})
mtoss = toss.melt(value_vars = ["bat", "field"], var_name = "toss_decision", value_name = "Count")
```

In [15]:
```
fig=px.bar(mtoss,x="toss_decision",y="Count",template="plotly_dark",color_discrete_sequence=["blue"])
fig.show()
```

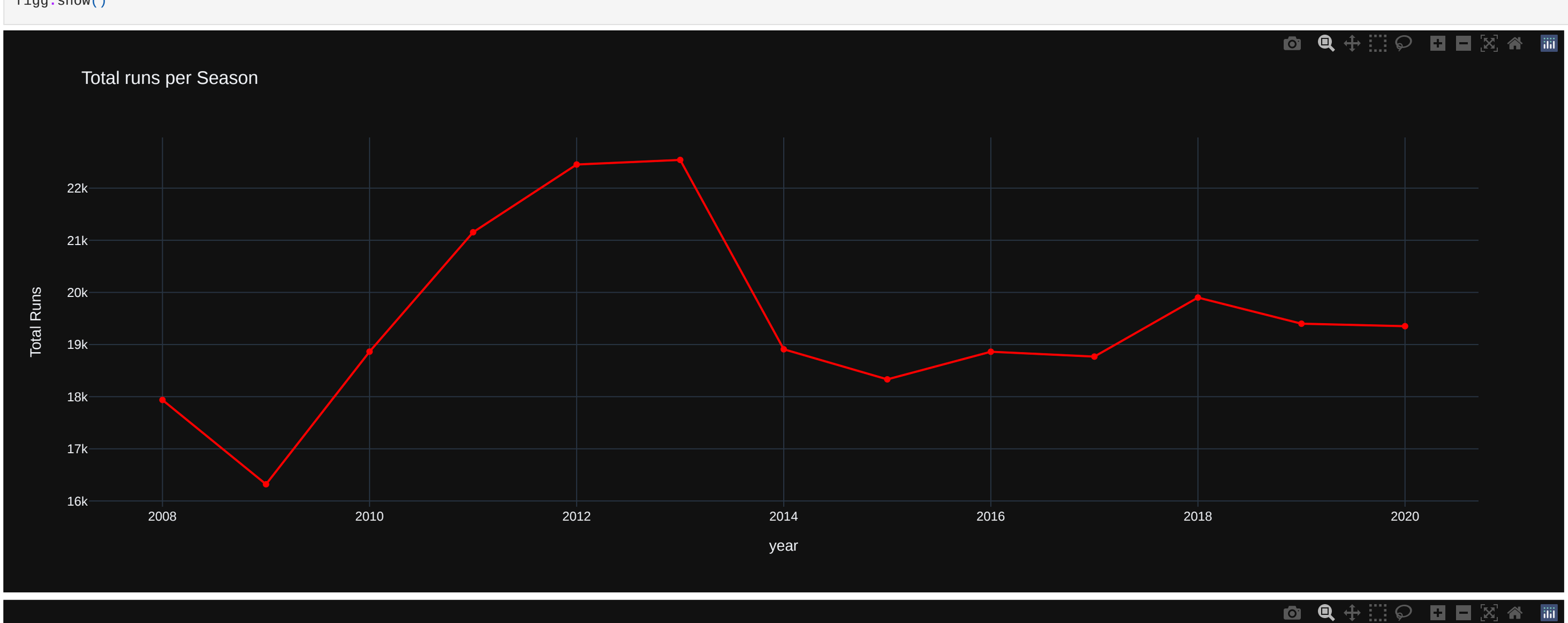


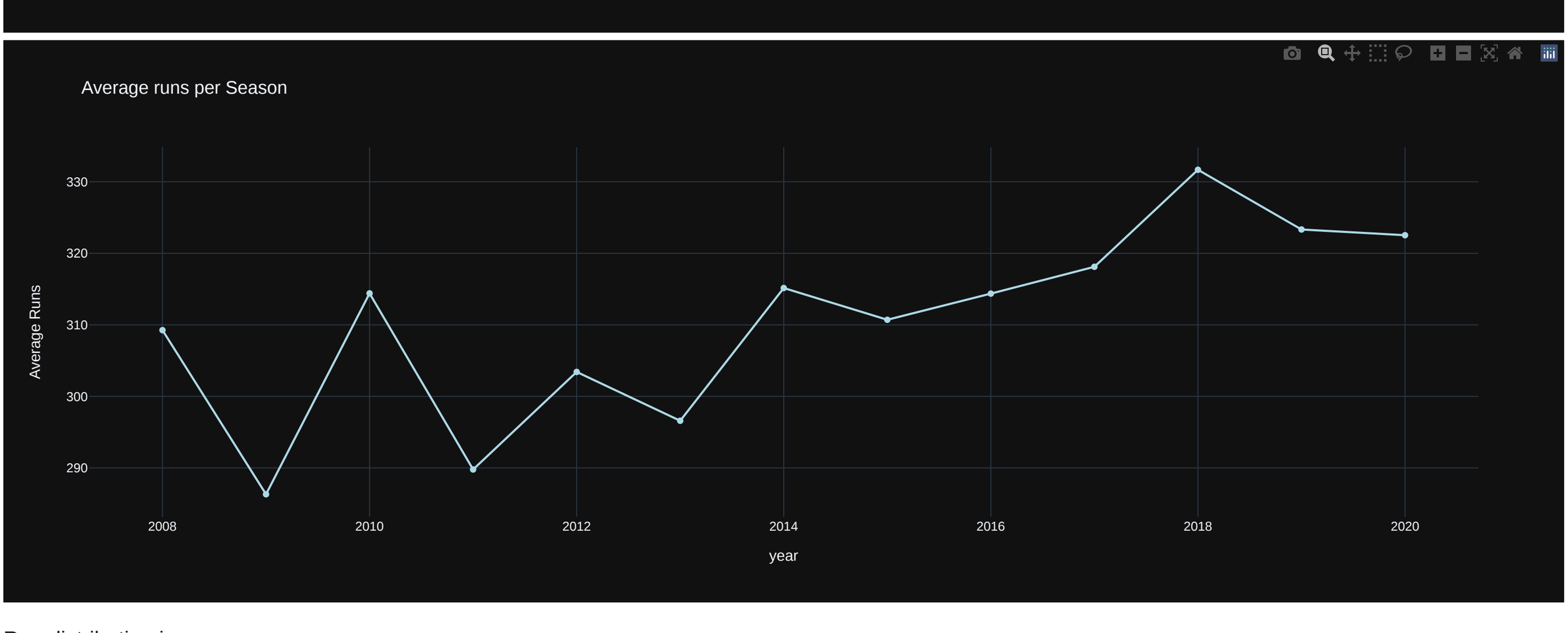
Most of the time teams opted to field after winning the toss which indicates that Indian pitch conditions are more favoured to run chase

Total and Average runs in every season

In [16]:
```
mergedf=pd.merge(matches,deliveries,on="id")
Season1=mergedf.pivot_table(index="Season",values="total_runs",aggfunc="sum").reset_index()
Season2=mergedf.pivot_table(index=["Season","id"],values="total_runs",aggfunc="sum").reset_index()
dff=Season2["Season"].value_counts().rename_axis("Season").reset_index(name="total matches").sort_values(by="Season")
newdf=pd.merge(Season1,dff,on="Season")
newdf["average runs"]=newdf["total_runs"]/newdf["total matches"]
```

In [17]:
```
fig1=px.line(newdf,x="Season",y="total_runs",template="plotly_dark",color_discrete_sequence=["red"])
fig2=px.scatter(newdf,x="Season",y="total_runs",template="plotly_dark",color_discrete_sequence=["red"])
fig=go.Figure(fig1["data"]+fig2["data"])
fig.update_layout(template="plotly_dark",title="Total runs per Season")
fig.update_xaxes(title="year")
fig.update_yaxes(title="Total Runs")
fig.show()
fig11=px.line(newdf,x="Season",y="average runs",template="plotly_dark",color_discrete_sequence=["lightblue"])
fig22=px.scatter(newdf,x="Season",y="average runs",template="plotly_dark",color_discrete_sequence=["lightblue"])
figg=go.Figure(fig11["data"]+fig22["data"])
figg.update_layout(template="plotly_dark",title="Average runs per Season")
figg.update_xaxes(title="year")
figg.update_yaxes(title="Average Runs")
figg.show()
```

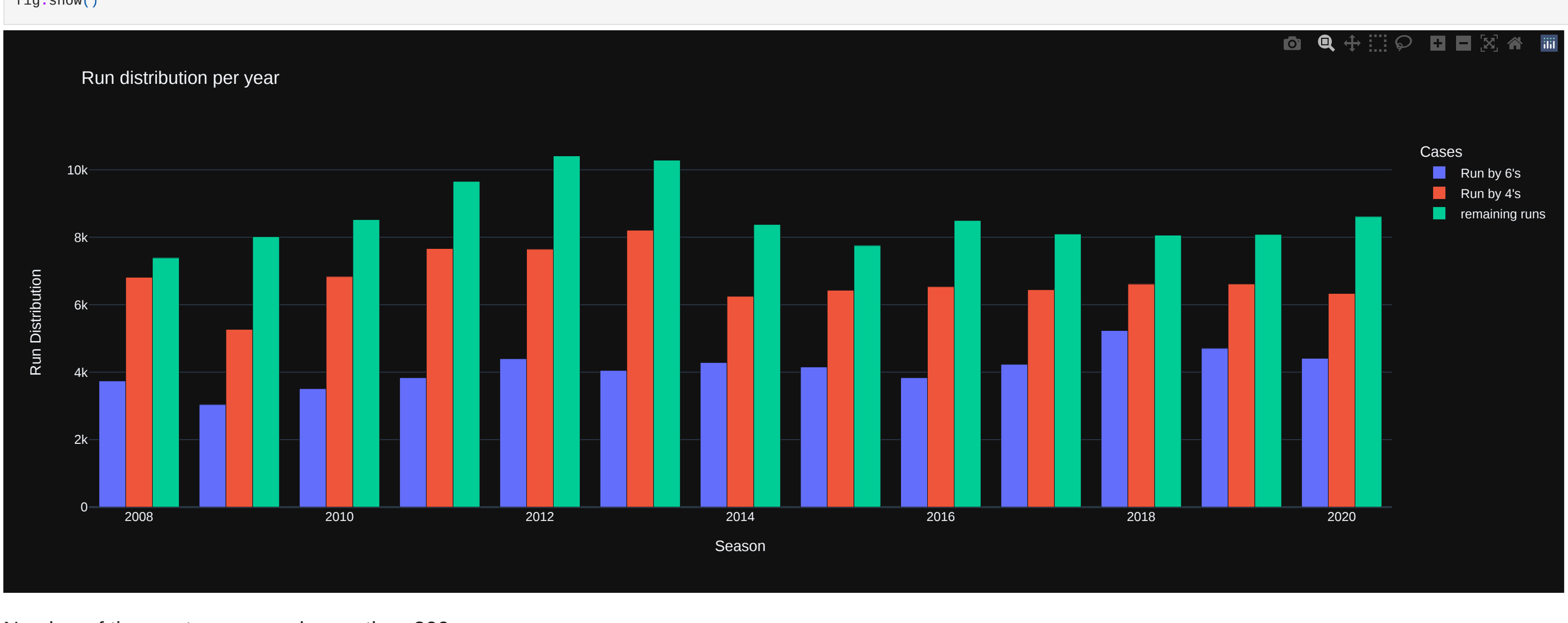




Run distribution in every season

In [18]:
```
df1=mergedf[mergedf["batsman_runs"] ==  6].pivot_table(index=["Season","id"],values="batsman_runs",
    aggfunc = "sum").reset_index().pivot_table(index = ["Season"],
     values = "batsman_runs", aggfunc = "sum").reset_index()
df2=mergedf[mergedf["batsman_runs"] ==  4].pivot_table(index=["Season","id"],values="batsman_runs",
    aggfunc = "sum").reset_index().pivot_table(index = ["Season"],
     values = "batsman_runs", aggfunc = "sum").reset_index()
df3=mergedf.pivot_table(index=["Season","id"],values="total_runs",
    aggfunc = "sum").reset_index().pivot_table(index = ["Season"],
     values = "total_runs", aggfunc = "sum").reset_index()
new=pd.merge(df1,df2,on="Season").merge(df3,on="Season")
new["remaining runs"]=new["total_runs"]-(new["batsman_runs_x"]+new["batsman_runs_y"])
new.rename(columns={
    "batsman_runs_x":"Run by 6's",
    "batsman_runs_y":"Run by 4's"
},inplace =True)
mnew=new.melt(id_vars="Season",value_vars=["Run by 6's","Run by 4's","remaining runs"],
          var_name="Cases",value_name="Run Distribution")
```

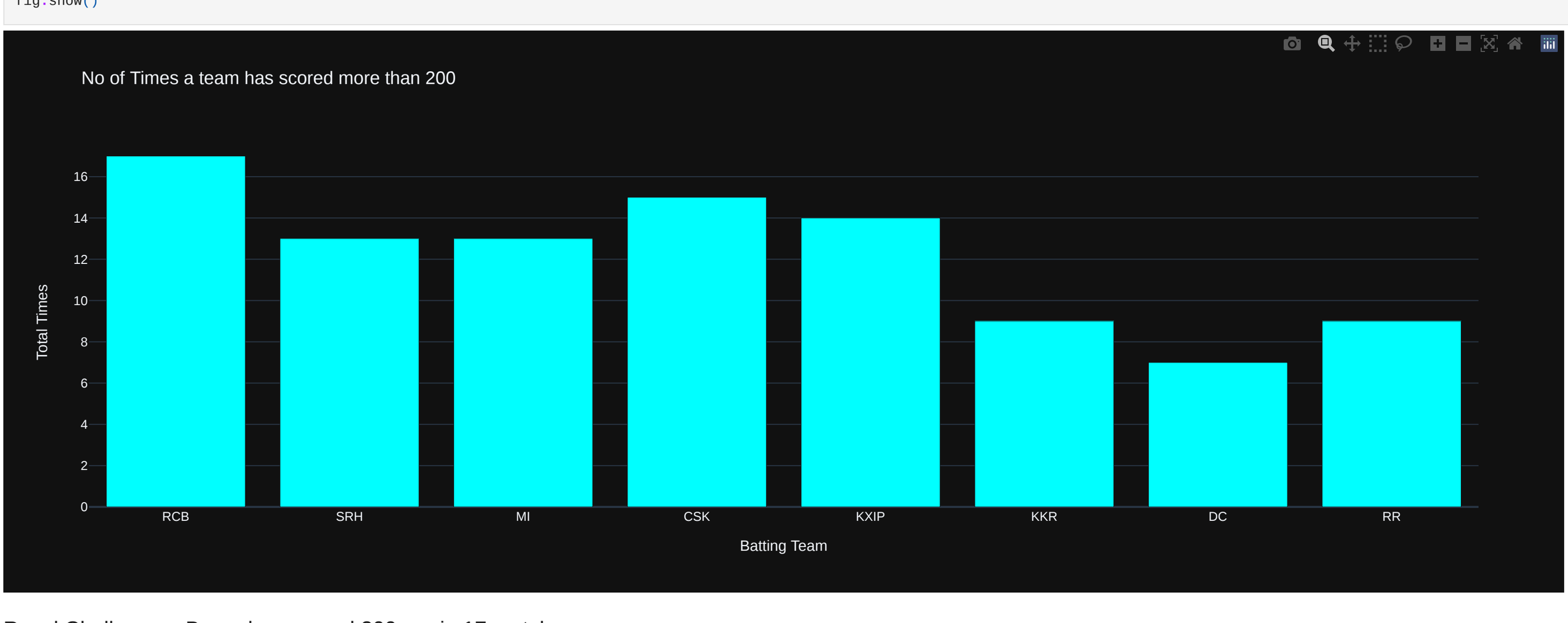
In [19]:
```
fig=px.bar(mnew,color="Cases",x="Season",y="Run Distribution",template="plotly_dark",barmode="group",title="Run distribution per year ")
fig.show()
```



Number of times a team scored more then 200

In [20]:
```
df1=mergedf[mergedf["inning"] ==  1].pivot_table(index=["inning","id","batting_team"],values="total_runs",
    aggfunc = "sum").reset_index()
df2=mergedf[mergedf["inning"] ==  2].pivot_table(index=["inning","id","batting_team"],values="total_runs",
    aggfunc = "sum").reset_index()
new=pd.merge(df1,df2,on="id")
op1=new[new["total_runs_x"]>200]["batting_team_x"].value_counts().rename_axis("Batting Team").reset_index(name="Total Times1")
op2=new[new["total_runs_y"]>200]["batting_team_y"].value_counts().rename_axis("Batting Team").reset_index(name="Total Times2")
j= pd.merge(op1,op2,on="Batting Team")
j["Total Times"]=j["Total Times1"]+j["Total Times2"]
newj=j[["Batting Team","Total Times"]]
```

In [21]:
```
fig=px.bar(newj,x="Batting Team",y="Total Times",template ="plotly_dark",color_discrete_sequence=["cyan"],
          title ="No of Times a team has scored more than 200")
fig.show()
```

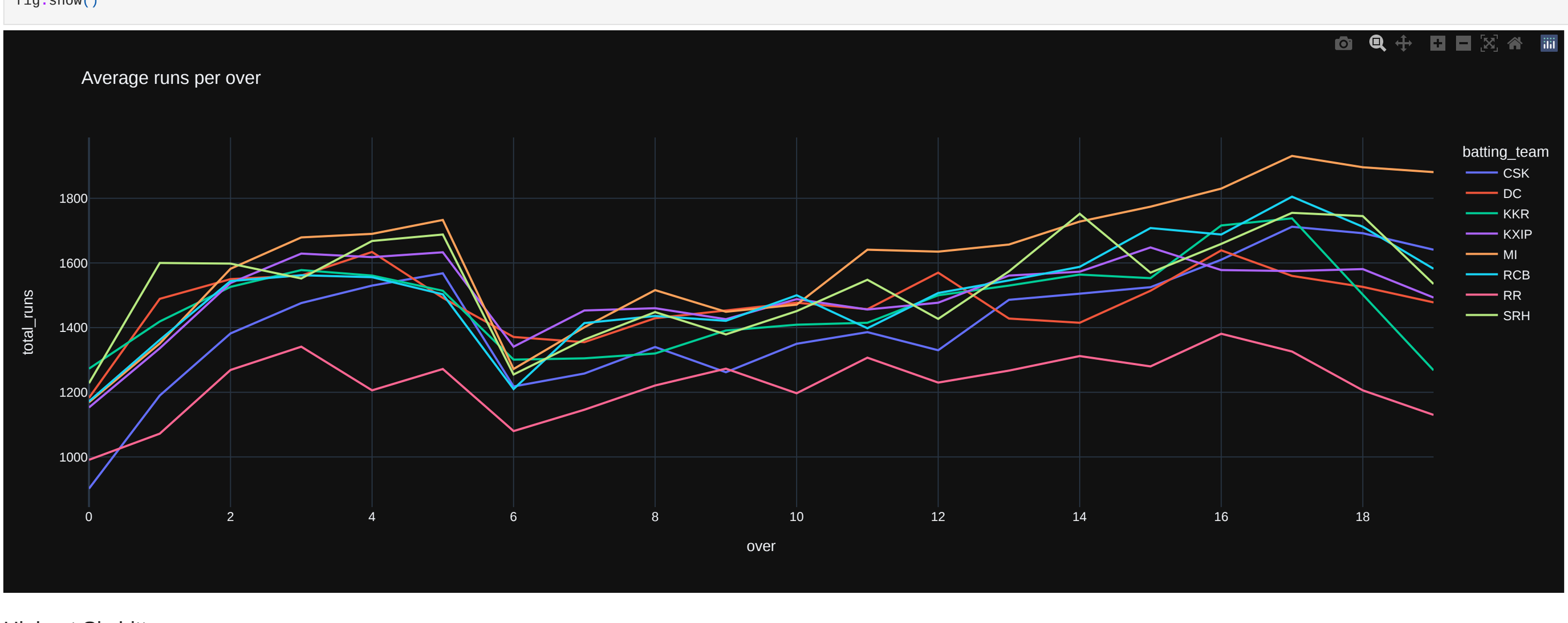


Royal Challengers Bangaluru scored 200 run in 17 matches

Average run scored by IPL teams

In [22]:
```
over=mergedf.pivot_table(index=["batting_team","over"],values="total_runs",aggfunc="sum").reset_index()
over2=over[(over["batting_team"] !="KTK") & (over["batting_team"] !="GL") & (over["batting_team"] !="PW")&
          (over["batting_team"] !="RPS")]
```

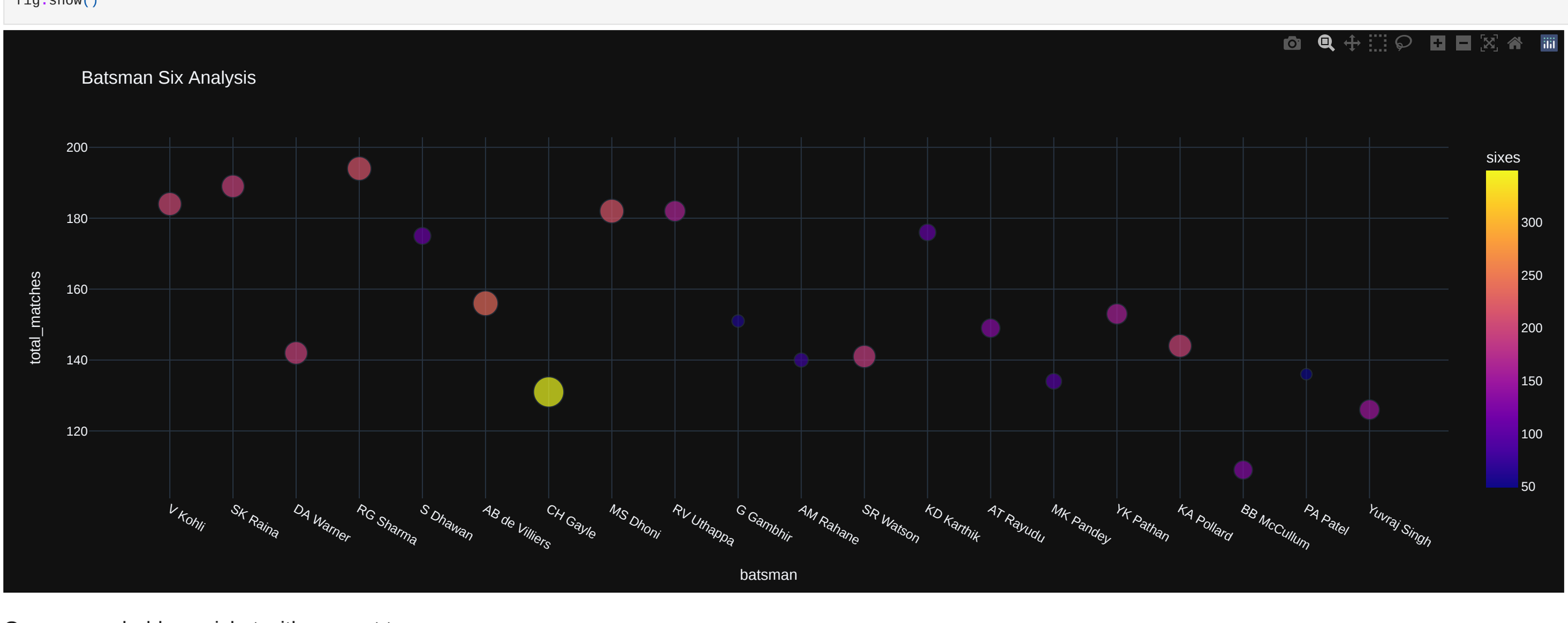
In [23]:
```
fig=px.line(over2,x="over",y="total_runs",template ="plotly_dark",color="batting_team",
           title ="Average runs per over")
fig.show()
```



Highest Six hitters

In [24]:
```
a1=mergedf.pivot_table(index="batsman",values=["batsman_runs"],aggfunc="sum").reset_index().sort_values(by="batsman_runs",ascending=False)
a2=deliveries[["batsman","ball"]]
a23=a2["batsman"].value_counts().rename_axis("batsman").reset_index(name="balls")
df1=pd.merge(a1,a23,on="batsman")
df1["strike rate"]=df1["batsman_runs"]/df1["balls"]*100
a3=deliveries.pivot_table(index=["batsman","id"],values="total_runs",aggfunc="sum").reset_index()
a4=a3["batsman"].value_counts().rename_axis("batsman").reset_index(name="total_matches")
df3=pd.merge(df1,a4,on="batsman")
df3["average score"]=df3["batsman_runs"]/df3["total_matches"]
a5=mergedf[mergedf["batsman_runs"]==6].pivot_table(index="batsman",values="total_runs",aggfunc="sum").reset_index()
a5["sixes"]=a5["total_runs"]//6
a55=a5[["batsman","sixes"]]
a6=mergedf[mergedf["batsman_runs"]==4].pivot_table(index="batsman",values="total_runs",aggfunc="sum").reset_index()
a6["fours"]=a6["total_runs"]//4
a66=a6[["batsman","fours"]]
result=pd.merge(df3,a55,on="batsman").merge(a66,on="batsman").head(20)
```

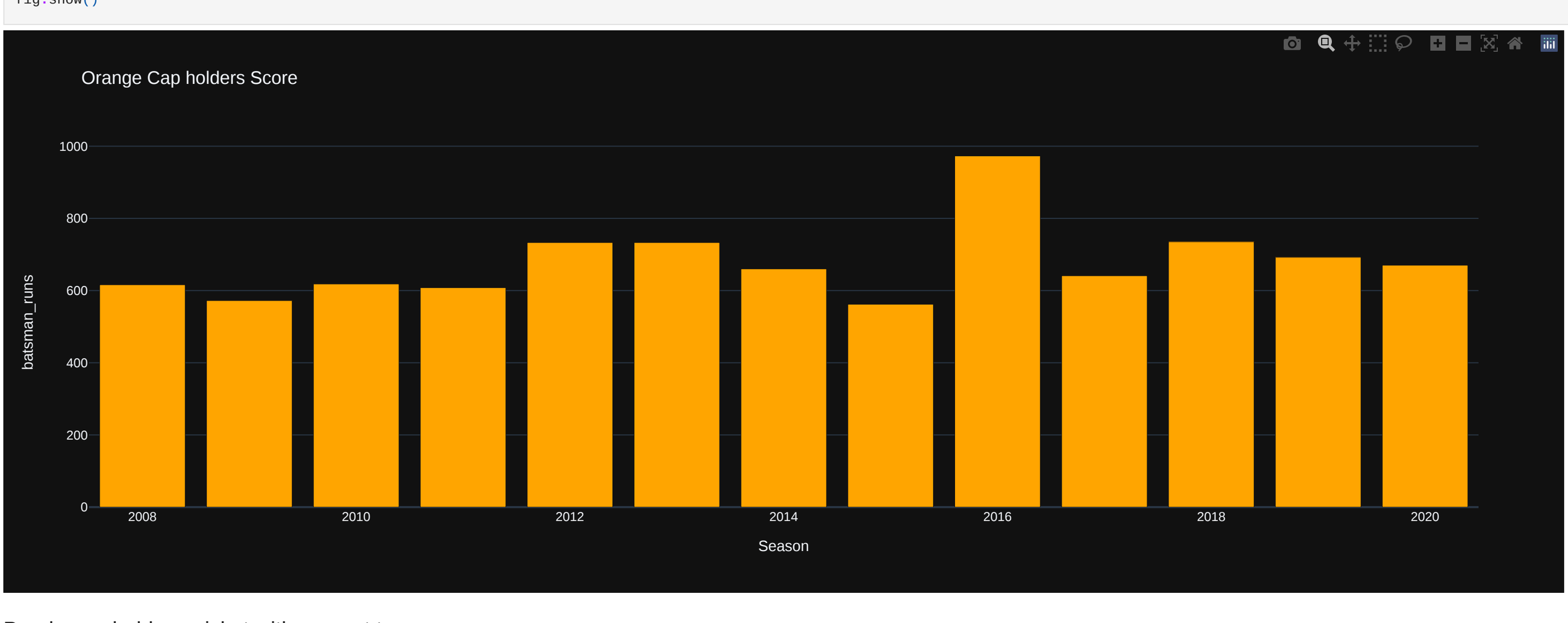
In [25]:
```
fig=px.scatter(result,x="batsman",y="total_matches",size="sixes",template ="plotly_dark",color="sixes",
          title ="Batsman Six Analysis",size_max=20)
fig.show()
```



Orange cap holders wicket with respect to year

In [26]:
```
df1=mergedf.pivot_table(index=["Season","batsman"],values="batsman_runs",aggfunc="sum").reset_index().sort_values(by="batsman_runs",ascending=False)
orange=df1.drop_duplicates(subset=["Season"],keep="first").sort_values(by="Season")
```

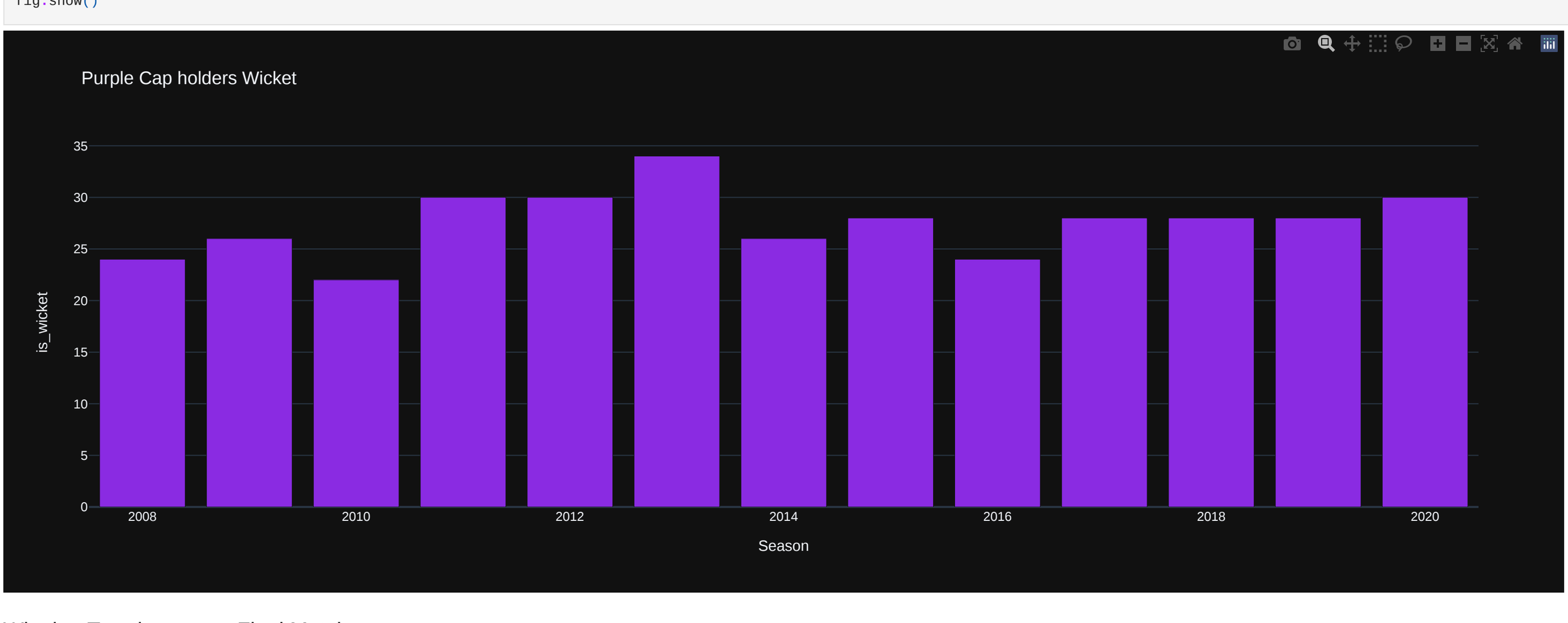
In [27]:
```
fig=px.bar(orange,x="Season",y="batsman_runs",template ="plotly_dark",color_discrete_sequence=["orange"],
          title ="Orange Cap holders Score")
fig.show()
```



Purple cap holders wicket with respect to year

In [28]:
```
df1=mergedf[mergedf["dismissal_kind"]!="run_out"].pivot_table(index=["Season","bowler"],values=["is_wicket"],
    aggfunc="sum").reset_index().sort_values(by="is_wicket",ascending=False).drop_duplicates(subset=["Season"],
      keep="first").sort_values(by="Season")
```

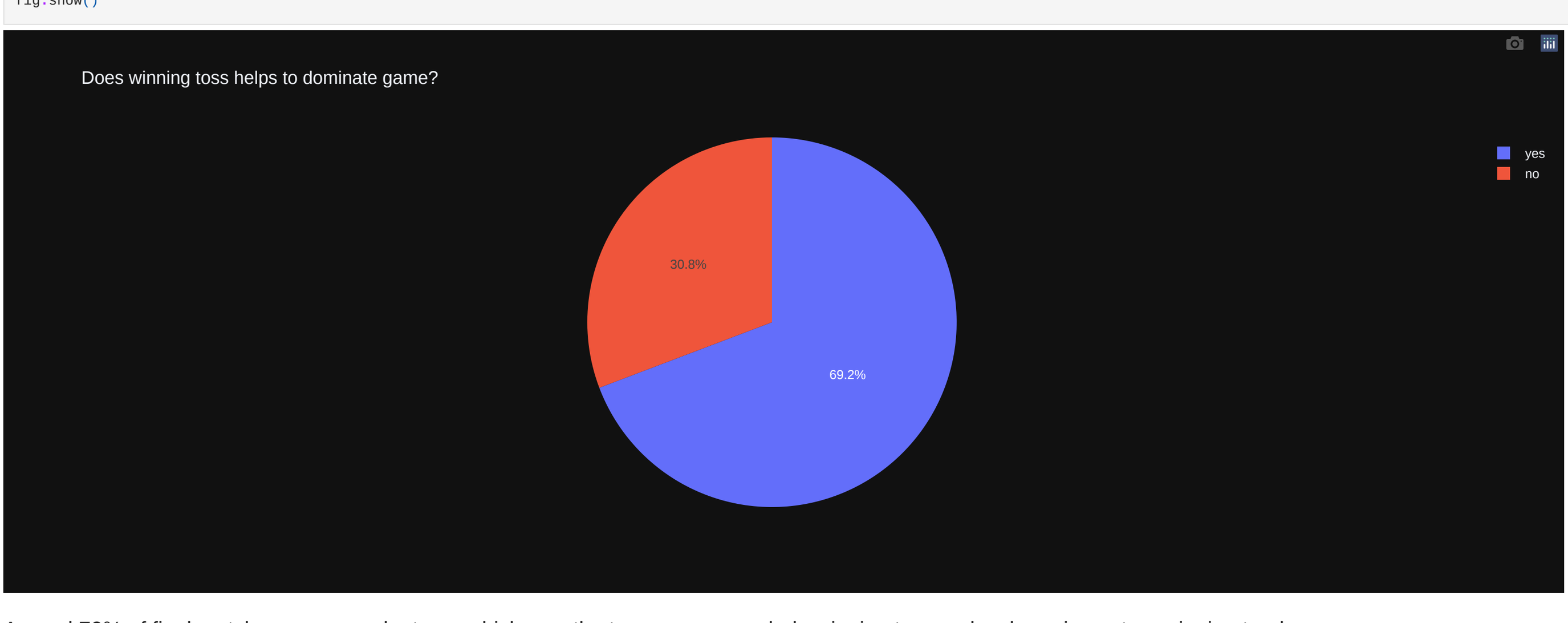
In [29]:
```
fig=px.bar(df1,x="Season",y="is_wicket",template ="plotly_dark",color_discrete_sequence=["blueviolet"],
          title ="Purple Cap holders Wicket ")
fig.show()
```



Winning Toss impact on Final Matches

In [30]:
```
df1=matches.drop_duplicates(subset=["Season"],keep="last")
w1=df1[df1["toss_winner"]==df1["winner"]][["winner"]].value_counts().sum()
w2=df1[df1["toss_winner"]!=df1["winner"]][["winner"]].value_counts().sum()
result=pd.DataFrame({"yes":[w1],
      "no":[w2]}).melt(value_vars=["yes",
        "no"],value_name="Count",var_name="Cases")
```

In [31]:
```
fig=px.pie(result,names="Cases",values="Count",template="plotly_dark",title="Does winning toss helps to dominate game?")
fig.show()
```

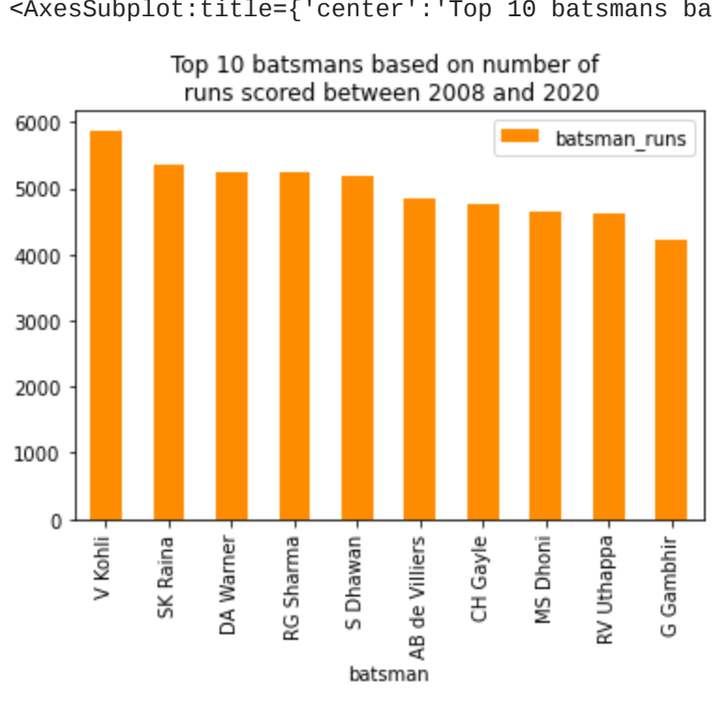


Around 70% of final matches were won by team which won the toss ,so we conclude winning toss makes huge impact on winning trophy.

Finding Highest run scorers

In [32]:
```
data_top10_batsman_split = pd.DataFrame(deliveries,columns=['batsman','batsman_runs'])
data_top10_batsman_bfr_sort_tmp = data_top10_batsman_split.groupby(['batsman']).agg({'batsman_runs':[sum]}).reset_index()
file_obj  = open("./df_agg_bowler_tmp.csv","w+")
data_top10_batsman_bfr_sort_tmp.to_csv("./data_top10_batsman.csv",index=False)
data_top10_batsman=pd.read_csv("./data_top10_batsman.csv",skiprows=[1]).sort_values(by='batsman_runs',ascending=False).head(10).set_index('batsman')
data_top10_batsman.plot.bar(alpha=1,rot=90,title='Top 10 batsmans based on number of \n runs scored between 2008 and 2020',color='darkorange')
```

Out[32]:
```
<AxesSubplot:title={'center':'Top 10 batsmans based on number of \n runs scored between 2008 and 2020'}, xlabel='batsman'>
```



Virat Kohli is the highest run scorer of IPL tournament from 2008 to 2020.