In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import plotly.express as px
import seaborn as sns
```

# analyzing the data

In [2]:
```
df = pd.read_csv('IPL Matches 2008-2020.csv')
df_ = pd.read_csv('IPL Ball-by-Ball 2008-2020.csv')
```

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
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
```

In [4]:
```
df_.isnull().sum()
```

Out[4]:
```
id                    0
inning                0
over                  0
ball                  0
batsman               0
non_striker           0
bowler                0
batsman_runs          0
extra_runs            0
total_runs            0
non_boundary          0
is_wicket             0
dismissal_kind       183973
player_dismissed     183973
fielder              186684
extras_type          183235
batting_team          0
bowling_team          191
dtype: int64
```

In [5]:
```
df.head(1)
```

Out[5]:

| | id | city | date | player_of_match | venue | neutral_venue | team1 | team2 |
|---|---|---|---|---|---|---|---|---|
| **0** | 335982 | Bangalore | 2008-04-18 | BB McCullum | M Chinnaswamy Stadium | 0 | Royal Challengers Bangalore | Kolkata Knight Riders |

In [6]:
```
df_.head(1)
```

Out[6]:

| | id | inning | over | ball | batsman | non_striker | bowler | batsman_runs | extra_runs | total_ru |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 335982 | 1 | 6 | 5 | RT Ponting | BB McCullum | AA Noffke | 1 | 0 | |

In [7]:
```
df[['season', 'month', 'day']] = df['date'].str.split('-', expand=True)
df['year'] = df['season'].astype(int)
df['month'] = df['month'].astype(int)
df['day'] = df['day'].astype(int)
```

In [8]:
```
df.columns.unique()
```

Out[8]:
```
Index(['id', 'city', 'date', 'player_of_match', 'venue', 'neutral_venue',
       'team1', 'team2', 'toss_winner', 'toss_decision', 'winner', 'result',
       'result_margin', 'eliminator', 'method', 'umpire1', 'umpire2', 'seaso
n',
       'month', 'day', 'year'],
      dtype='object')
```

In [9]:
```
df_.columns.unique()
```

Out[9]:
```
Index(['id', 'inning', 'over', 'ball', 'batsman', 'non_striker', 'bowler',
       'batsman_runs', 'extra_runs', 'total_runs', 'non_boundary', 'is_wicke
t',
       'dismissal_kind', 'player_dismissed', 'fielder', 'extras_type',
       'batting_team', 'bowling_team'],
      dtype='object')
```

## Q-1 | total count of matches playes in each season

In [10]:
```
matches_per_season = df.groupby(['season'])['id'].count()
print(matches_per_season)
```

```
season
2008    58
2009    57
2010    60
2011    73
2012    74
2013    76
2014    60
2015    59
2016    60
2017    59
2018    60
2019    60
2020    60
Name: id, dtype: int64
```

## Q-2 | runs scored in each season

In [11]:
```
runs_in_each_season = df[['id','season']].merge(df_,left_on ='id',right_on =
runs_in_each_season.head()
```

Out[11]:

| | season | inning | over | ball | batsman | non_striker | bowler | batsman_runs | extra_runs | total_r |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2008 | 1 | 6 | 5 | RT Ponting | BB McCullum | AA Noffke | 1 | 0 | |
| **1** | 2008 | 1 | 6 | 6 | BB McCullum | RT Ponting | AA Noffke | 1 | 0 | |
| **2** | 2008 | 1 | 7 | 1 | BB McCullum | RT Ponting | Z Khan | 0 | 0 | |
| **3** | 2008 | 1 | 7 | 2 | BB McCullum | RT Ponting | Z Khan | 1 | 0 | |
| **4** | 2008 | 1 | 7 | 3 | RT Ponting | BB McCullum | Z Khan | 1 | 0 | |

In [12]:
```
match_runs = runs_in_each_season.groupby(['season'])['total_runs'].sum().reset
match_runs
```

Out[12]:

| | season | total_runs |
|---|---|---|
| **0** | 2008 | 17937 |
| **1** | 2009 | 16320 |
| **2** | 2010 | 18864 |
| **3** | 2011 | 21154 |
| **4** | 2012 | 22453 |
| **5** | 2013 | 22541 |
| **6** | 2014 | 18909 |
| **7** | 2015 | 18332 |
| **8** | 2016 | 18862 |
| **9** | 2017 | 18769 |
| **10** | 2018 | 19901 |
| **11** | 2019 | 19400 |
| **12** | 2020 | 19352 |

In [13]:
```
match_runs_graph = px.line(match_runs,x = 'season',y = 'total_runs',title = '
match_runs_graph
```

## Q-3 | runs scored per match in each season

In [14]:
```
df__ = df[['id','season']].merge(df_,left_on ='id',right_on = 'id',how = 'left
df__['Runs_Scored'] = df__['total_runs'] + df__['extra_runs']
runs_per_match_per_season = df__.groupby(['season','id'])['Runs_Scored'].sum()
Runs_scored = pd.DataFrame(runs_per_match_per_season , columns = ['season','id
Runs_scored
```

Out[14]:

| | season | id | Runs_Scored |
|---|---|---|---|
| **0** | 2008 | 335982 | 340 |
| **1** | 2008 | 335983 | 464 |
| **2** | 2008 | 335984 | 278 |
| **3** | 2008 | 335985 | 347 |
| **4** | 2008 | 335986 | 260 |
| **...** | ... | ... | ... |
| **811** | 2020 | 1216547 | 417 |
| **812** | 2020 | 1237177 | 353 |
| **813** | 2020 | 1237178 | 277 |
| **814** | 2020 | 1237180 | 375 |
| **815** | 2020 | 1237181 | 321 |

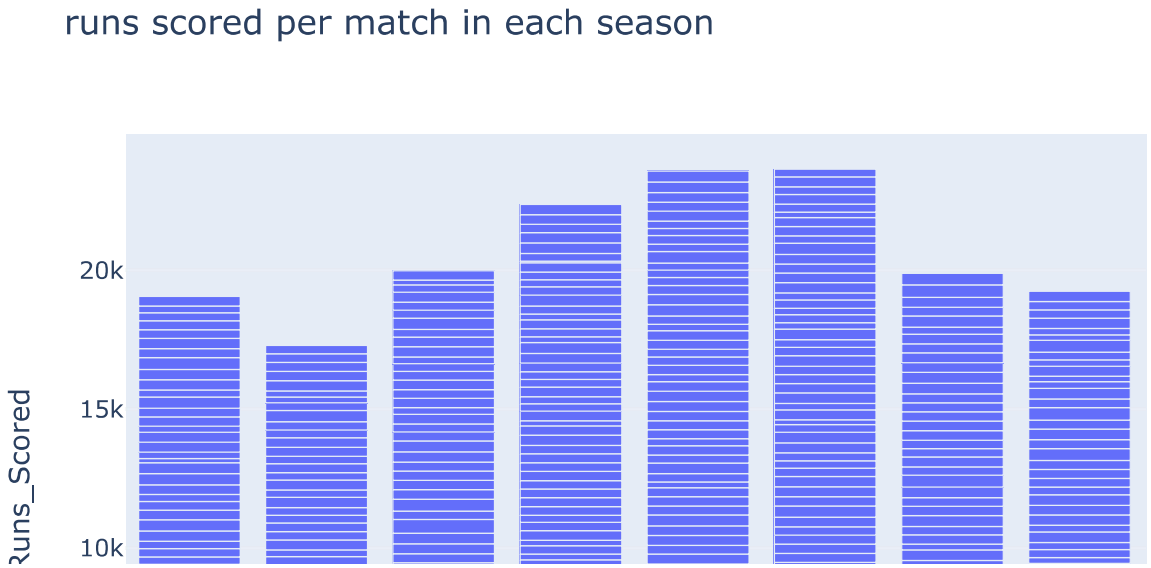
816 rows × 3 columns

In [15]:
```
df__.head(1)
```

Out[15]:

| | id | season | inning | over | ball | batsman | non_striker | bowler | batsman_runs | extra_runs |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 335982 | 2008 | 1 | 6 | 5 | RT Ponting | BB McCullum | AA Noffke | 1 | 0 |

In [16]:
```
px.bar(Runs_scored , x ='season', y='Runs_Scored' , title = 'runs scored per r
```



## Q-4 | who has umpired the most

In [17]:
```
df.head()
```

Out[17]:

| | id | city | date | player_of_match | venue | neutral_venue | team1 | te |
|---|---|---|---|---|---|---|---|---|
| **0** | 335982 | Bangalore | 2008-04-18 | BB McCullum | M Chinnaswamy Stadium | 0 | Royal Challengers Bangalore | Ko K R |
| **1** | 335983 | Chandigarh | 2008-04-19 | MEK Hussey | Punjab Cricket Association Stadium, Mohali | 0 | Kings XI Punjab | Che S K |
| **2** | 335984 | Delhi | 2008-04-19 | MF Maharoof | Feroz Shah Kotla | 0 | Delhi Daredevils | Rajas R( |
| **3** | 335985 | Mumbai | 2008-04-20 | MV Boucher | Wankhede Stadium | 0 | Mumbai Indians | F Challer Bang |
| **4** | 335986 | Kolkata | 2008-04-20 | DJ Hussey | Eden Gardens | 0 | Kolkata Knight Riders | De Chai |

5 rows × 21 columns

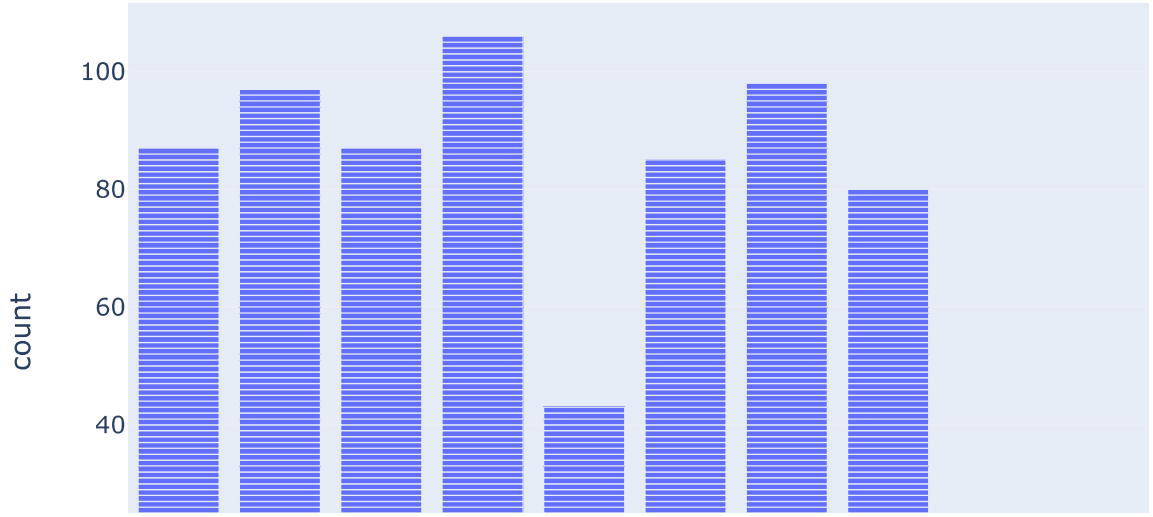
In [18]:
```
umpire_data = pd.concat([df['umpire1'],df['umpire2']])
umpire_data.value_counts().head()
```

Out[18]:
```
S Ravi              121
HDPK Dharmasena      94
AK Chaudhary         87
C Shamshuddin        82
M Erasmus            65
dtype: int64
```

## Q-5 | which team has won the most tosses

In [19]:
```
toss_data = df[['toss_winner']]
```

In [20]:
```
px.bar(toss_data )
```




mumbai indians has won the most tosses

## Q-6 | team decision after winning the tosses

In [21]:
```
df['toss_decision'].unique()
```

Out[21]:
```
array(['field', 'bat'], dtype=object)
```

the team has to choose between two decisions.it has to be either 'field' or 'bat'

lets analyze which of the decisions is choosen by most of teams

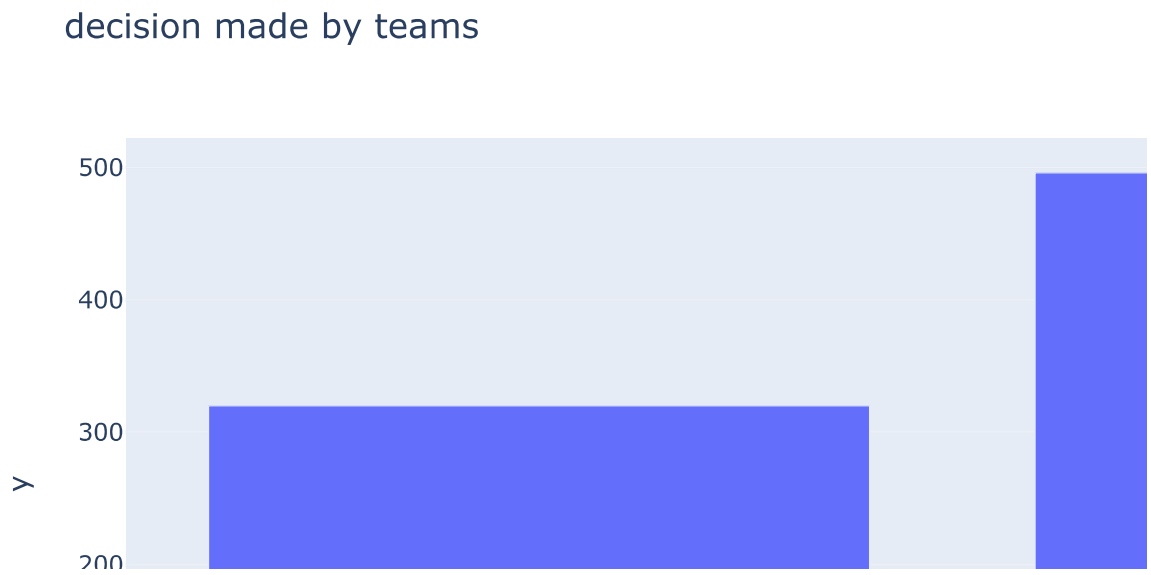
In [22]:
```
data = df[['toss_winner','toss_decision']]
bat = len(data[data['toss_decision'] == 'bat'])
field = len(data[data['toss_decision'] == 'field'])
```

In [23]:
```
bat = len(data[data['toss_decision'] == 'bat'])
field = len(data[data['toss_decision'] == 'field'])
```

In [24]:
```
print('choose to bat    :' , bat)
print('choose to field  :' , field)
print('most no of teams choosen to field first')
```

```
choose to bat    : 320
choose to field  : 496
most no of teams choosen to field first
```

In [25]:
```
name = 'decision made by teams'
label = ['bat' , 'field']
px.bar(y = [bat , field] ,labels = label, title = name)
```



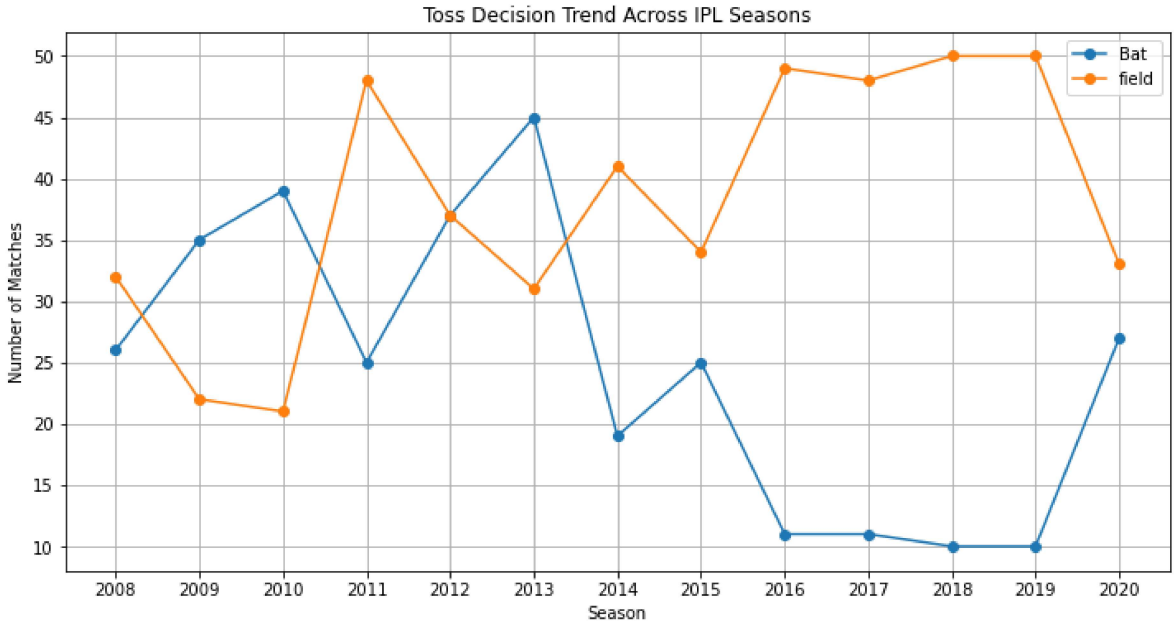
## Q-7| how does the toss decision vary accross the season

In [26]:
```
df.columns.unique()
```

Out[26]:
```
Index(['id', 'city', 'date', 'player_of_match', 'venue', 'neutral_venue',
       'team1', 'team2', 'toss_winner', 'toss_decision', 'winner', 'result',
       'result_margin', 'eliminator', 'method', 'umpire1', 'umpire2', 'seaso
n',
       'month', 'day', 'year'],
      dtype='object')
```

In [27]:
```
# Convert 'season' to categorical for better ordering on the plot
df['season'] = pd.Categorical(df['season'])
# Count the frequency of each toss decision for each season
decision_counts = pd.crosstab(index=df['season'], columns=df['toss_decision'])
# Plotting the trend using a line chart
plt.figure(figsize=(12, 6))
plt.plot(decision_counts.index, decision_counts['bat'], marker='o', label='Bat
plt.plot(decision_counts.index, decision_counts['field'], marker='o', label='

plt.title('Toss Decision Trend Across IPL Seasons')
plt.xlabel('Season')
plt.ylabel('Number of Matches')
plt.legend()
plt.grid(True)
plt.show()
```



## Q-8 | does winning the toss imply winning the game

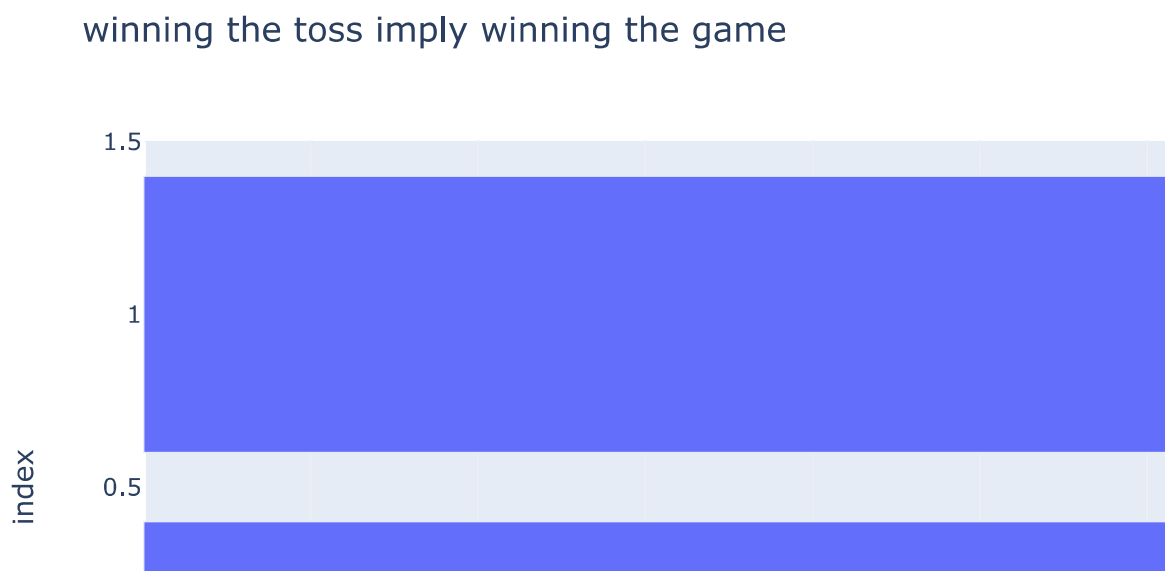
In [28]:
```
winning = df[['toss_winner','winner']]
```

In [29]:
```
won = 0
for i in winning.values:
    if(i[0] == i[1]):
        won += 1
lost = len(winning)- won
print('won in both toss and match :',won)
print('didnt win in both toss and match :',lost)
print('teams won in both toss and match are more than the teams didnt win in b
```

```
won in both toss and match : 418
didnt win in both toss and match : 398
teams won in both toss and match are more than the teams didnt win in both t
oss and match
```

In [30]:
```
decision = [[won , lost]]
```

In [31]:
```
px.bar(x = decision , title ='winning the toss imply winning the game' )
```



## Q-9 | how many times has the chasing team won the match

In [32]:
```
winner = pd.DataFrame(df[['id','winner']])
```

In [33]:
```
bowling_team =pd.DataFrame(df_[['id','bowling_team']]).reset_index()
```

In [34]:
```
bowling__team = pd.DataFrame (bowling_team.drop_duplicates(subset = 'id'))
```

In [35]:
```
bowling__team
```

Out[35]:

| | index | id | bowling_team |
|---|---|---|---|
| **0** | 0 | 335982 | Royal Challengers Bangalore |
| **225** | 225 | 335983 | Chennai Super Kings |
| **473** | 473 | 335984 | Delhi Daredevils |
| **692** | 692 | 335985 | Royal Challengers Bangalore |
| **938** | 938 | 335986 | Deccan Chargers |
| **...** | ... | ... | ... |
| **192241** | 192241 | 1216547 | Mumbai Indians |
| **192489** | 192489 | 1237177 | Mumbai Indians |
| **192736** | 192736 | 1237178 | Sunrisers Hyderabad |
| **192983** | 192983 | 1237180 | Sunrisers Hyderabad |
| **193233** | 193233 | 1237181 | Mumbai Indians |

816 rows × 3 columns

In [36]:
```
merged_data = pd.merge(winner,bowling__team, on='id')
chasing_data = pd.DataFrame(merged_data)
chasing_team_wins = merged_data[merged_data['bowling_team'] == merged_data['wi
print(f'The chasing team won {chasing_team_wins} times.combining  in all seasc
```

```
The chasing team won 470 times.combining  in all seasons.
```

## Q-10 | WHICH ALL TEAMS HAD WON THE TOURNAMENT

In [37]:
```
winning_teams = df.groupby('season')['winner'].apply(lambda x: x.value_counts(
print(winning_teams)
```

```
   season                   wins
0    2008       Rajasthan Royals
1    2009       Delhi Daredevils
2    2010         Mumbai Indians
3    2011    Chennai Super Kings
4    2012  Kolkata Knight Riders
5    2013         Mumbai Indians
6    2014        Kings XI Punjab
7    2015    Chennai Super Kings
8    2016    Sunrisers Hyderabad
9    2017         Mumbai Indians
10   2018    Chennai Super Kings
11   2019         Mumbai Indians
12   2020         Mumbai Indians
```

## Q-11 | which team has played most number of matches

In [38]:
```
all_teams = pd.concat([df['team1'],df['team2']])
```

In [39]:
```
all_teams.value_counts()
```

Out[39]:
```
Mumbai Indians                 203
Royal Challengers Bangalore    195
Kolkata Knight Riders          192
Kings XI Punjab                190
Chennai Super Kings            178
Delhi Daredevils               161
Rajasthan Royals               161
Sunrisers Hyderabad            124
Deccan Chargers                 75
Pune Warriors                   46
Delhi Capitals                  33
Gujarat Lions                   30
Rising Pune Supergiant          16
Kochi Tuskers Kerala            14
Rising Pune Supergiants         14
dtype: int64
```

## mumbai indians has played most number of matches

## Q-12 | which team has won most no of times

In [40]:
```
win_counts = (df['winner'].value_counts())
print(win_counts)
print('-'*40)
print('Mumbai Indians has won most no of times')
print('-'*40)
```

```
Mumbai Indians                 120
Chennai Super Kings            106
Kolkata Knight Riders           99
Royal Challengers Bangalore     91
Kings XI Punjab                 88
Rajasthan Royals                81
Delhi Daredevils                67
Sunrisers Hyderabad             66
Deccan Chargers                 29
Delhi Capitals                  19
Gujarat Lions                   13
Pune Warriors                   12
Rising Pune Supergiant          10
Kochi Tuskers Kerala             6
Rising Pune Supergiants          5
Name: winner, dtype: int64
----------------------------------------
Mumbai Indians has won most no of times
----------------------------------------
```

## Q-13 | which team has the highest winning percentage

In [41]:
```
matches_count = all_teams.value_counts()
wins_count = (df['winner'].value_counts())
```

In [42]:
```
# Calculate the winning percentage for each team
winning_percentage = wins_count / matches_count * 100
print(winning_percentage)
```

```
Chennai Super Kings            59.550562
Deccan Chargers                38.666667
Delhi Capitals                 57.575758
Delhi Daredevils               41.614907
Gujarat Lions                  43.333333
Kings XI Punjab                46.315789
Kochi Tuskers Kerala           42.857143
Kolkata Knight Riders          51.562500
Mumbai Indians                 59.113300
Pune Warriors                  26.086957
Rajasthan Royals               50.310559
Rising Pune Supergiant         62.500000
Rising Pune Supergiants        35.714286
Royal Challengers Bangalore    46.666667
Sunrisers Hyderabad            53.225806
dtype: float64
```

In [43]:
```
# Identify the team with the highest winning percentage
max_winning_percentage_team = winning_percentage.idxmax()
max_winning_percentage = winning_percentage.max()

print(f"The team with the highest winning percentage is {max_winning_percentag
```

```
The team with the highest winning percentage is Rising Pune Supergiant with
a winning percentage of 62.50%.
```

## Q-14 | is there any lucky venue to any team

In [44]:
```
df.columns.unique()
```

Out[44]:
```
Index(['id', 'city', 'date', 'player_of_match', 'venue', 'neutral_venue',
       'team1', 'team2', 'toss_winner', 'toss_decision', 'winner', 'result',
       'result_margin', 'eliminator', 'method', 'umpire1', 'umpire2', 'seaso
n',
       'month', 'day', 'year'],
      dtype='object')
```

In [45]:
```
lucky_venue = pd.DataFrame(df[['team1' , 'team2','winner','venue']])
```

In [46]:
```
venue_wins = pd.crosstab(index=[lucky_venue['team1'], lucky_venue['venue']], 
venue_winning_percentage = venue_wins.div(venue_wins.sum(axis=1), axis=0) * 1
venue_winning_percentage.reset_index(inplace=True)

# Identify the lucky venue for each team
lucky_venues = venue_winning_percentage.groupby('team1').apply(lambda x: x[x[

print("Lucky Venues for Each Team:")
print(lucky_venues)
```

```
Lucky Venues for Each Team:
team1
Chennai Super Kings                                    Brabourne Sta
dium
Deccan Chargers                                                Newl
ands
Delhi Capitals                Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket S
t...
Delhi Daredevils                                          Buffalo
Park
Gujarat Lions                                               Green
Park
Kings XI Punjab                                                Newl
ands
Kochi Tuskers Kerala                                Holkar Cricket Sta
dium
Kolkata Knight Riders                                  Barabati Sta
dium
Mumbai Indians                                             Eden Gar
dens
Pune Warriors                                  Dr DY Patil Sports Aca
demy
Rajasthan Royals                                       Feroz Shah K
otla
Rising Pune Supergiant             Maharashtra Cricket Association Sta
dium
Rising Pune Supergiants       Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket S
t...
Royal Challengers Bangalore              MA Chidambaram Stadium, Che
pauk
Sunrisers Hyderabad                                    Feroz Shah K
otla
dtype: object
```

# Q-15 | innings wise comparision between teams

In [47]:
```
print(df_.columns.unique())
```

```
Index(['id', 'inning', 'over', 'ball', 'batsman', 'non_striker', 'bowler',
       'batsman_runs', 'extra_runs', 'total_runs', 'non_boundary', 'is_wicke
t',
       'dismissal_kind', 'player_dismissed', 'fielder', 'extras_type',
       'batting_team', 'bowling_team'],
      dtype='object')
```

In [48]:
```
innings_data = pd.DataFrame(df_[['id','inning','total_runs','batting_team','bo

grouped_innings = df_.groupby(['id','inning','batting_team'])

innings_totals = grouped_innings['total_runs'].sum().reset_index()


innings_pivot = innings_totals.pivot(index='id', columns=['inning', 'batting_t

innings_statistics = innings_pivot.describe()
```

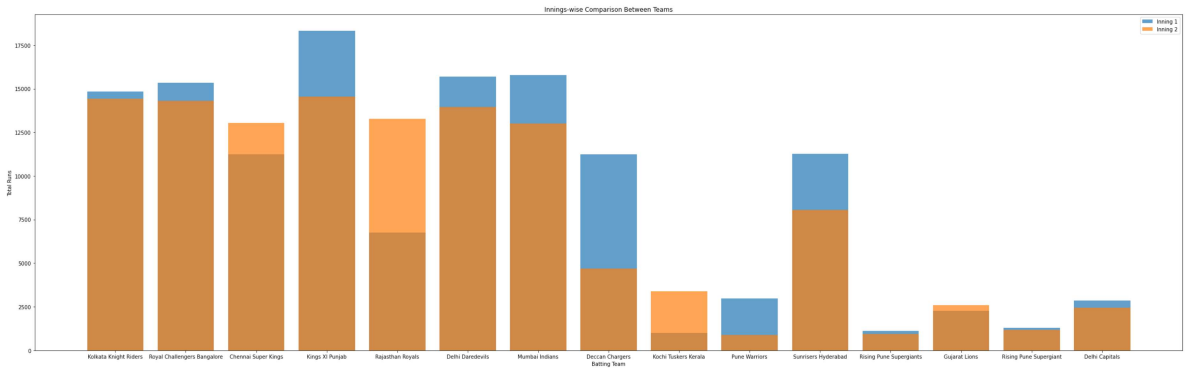
In [49]:
```
plt.figure(figsize=(40, 12))

# Get the unique teams for the x-axis
teams = innings_totals['batting_team'].unique()

# Plotting each inning's total runs for each team
for inning in innings_pivot.columns.levels[0]:
    plt.bar(
        x=teams,
        height=innings_pivot[inning].sum(),
        label=f'Inning {inning}',
        alpha=0.7
    )

# Set plot labels and title
plt.xlabel('Batting Team')
plt.ylabel('Total Runs')
plt.title('Innings-wise Comparison Between Teams')
plt.legend()

# Show the plot
plt.show()
```




## Q-16 | which team has scored the most no of 200+ score

In [ ]:
```
```

In [51]:
```
Runs_scored.head()
```

Out[51]:

| | season | id | Runs_Scored |
|---|---|---|---|
| **0** | 2008 | 335982 | 340 |
| **1** | 2008 | 335983 | 464 |
| **2** | 2008 | 335984 | 278 |
| **3** | 2008 | 335985 | 347 |
| **4** | 2008 | 335986 | 260 |

In [52]:
```
team1 = pd.DataFrame(df['team1'])

Runs_scored['team'] = team1

Runs_scored.sort_values('Runs_Scored',ascending = False).head()
```

Out[52]:

| | season | id | Runs_Scored | team |
|---|---|---|---|---|
| **146** | 2010 | 419137 | 490 | Chennai Super Kings |
| **679** | 2018 | 1136604 | 471 | Kings XI Punjab |
| **626** | 2017 | 1082641 | 471 | Mumbai Indians |
| **791** | 2020 | 1216527 | 467 | Kings XI Punjab |
| **1** | 2008 | 335983 | 464 | Kings XI Punjab |

In [53]:
```
score_200_plus = Runs_scored[Runs_scored['Runs_Scored'] >= 200]

team_200_plus_counts = score_200_plus['team'].value_counts()

most_200_plus_team = team_200_plus_counts.idxmax()

print(f'the team has scored the most no of 200+ score is {most_200_plus_team}')
```

```
the team has scored the most no of 200+ score is Royal Challengers Bangalore
```

In [54]:
```
score_200_plus
```

Out[54]:

| | season | id | Runs_Scored | team |
|---|---|---|---|---|
| **0** | 2008 | 335982 | 340 | Royal Challengers Bangalore |
| **1** | 2008 | 335983 | 464 | Kings XI Punjab |
| **2** | 2008 | 335984 | 278 | Delhi Daredevils |
| **3** | 2008 | 335985 | 347 | Mumbai Indians |
| **4** | 2008 | 335986 | 260 | Kolkata Knight Riders |
| **...** | ... | ... | ... | ... |
| **811** | 2020 | 1216547 | 417 | Royal Challengers Bangalore |
| **812** | 2020 | 1237177 | 353 | Mumbai Indians |
| **813** | 2020 | 1237178 | 277 | Royal Challengers Bangalore |
| **814** | 2020 | 1237180 | 375 | Delhi Capitals |
| **815** | 2020 | 1237181 | 321 | Delhi Capitals |

795 rows × 4 columns

# Q-17| which team has conceded 200+ runs the most

In [55]:
```
Runs_scored['bowling_team'] = df['team2']
```

In [56]:
```
Runs_scored.head(1)
```

Out[56]:

| | season | id | Runs_Scored | team | bowling_team |
|---|---|---|---|---|---|
| **0** | 2008 | 335982 | 340 | Royal Challengers Bangalore | Kolkata Knight Riders |

In [57]:
```
conceded_200_plus = Runs_scored['bowling_team'].value_counts()

conceded_team = conceded_200_plus.idxmax()
print('-'*68)
print(f'the team which has conceded most no of 200+ score is {conceded_team}')
print('-'*68)
```

```
--------------------------------------------------------------------
the team which has conceded most no of 200+ score is Mumbai Indians
--------------------------------------------------------------------
```

# Q-18 | what is the highest runs scored by a team in a single match

In [58]:
```
Runs_scored.sort_values('Runs_Scored' , ascending = False).head(1)
```

Out[58]:

| | season | id | Runs_Scored | team | bowling_team |
|---|---|---|---|---|---|
| **146** | 2010 | 419137 | 490 | Chennai Super Kings | Rajasthan Royals |

# Q-19 | which is the biggest win in terms of run margin

In [59]:
```
df['total_runs'] = Runs_scored['Runs_Scored']

run_margin = df[['team1','team2','total_runs','result_margin']]

run_margin.sort_values('result_margin',ascending = False).head(1)
```

Out[59]:

| | team1 | team2 | total_runs | result_margin |
|---|---|---|---|---|
| **620** | Delhi Daredevils | Mumbai Indians | 300 | 146.0 |

# Q-20 | which batsman have played the most no of

In [60]:
```
most_no_of_balls = df_['batsman'].value_counts()
```

In [61]:
```
print('-'*37)
print(most_no_of_balls.head(1))
print('-'*37)
```

```
-------------------------------------
V Kohli    4609
Name: batsman, dtype: int64
-------------------------------------
```

## Q-21 | who are the leading run scorers of all time

In [62]:
```
total_runs = df_.groupby('batsman')['total_runs'].sum().reset_index()
```

In [63]:
```
total_runs.sort_values(by = 'total_runs',ascending = False).head()
```

Out[63]:

| | batsman | total_runs |
|---|---|---|
| **505** | V Kohli | 6081 |
| **438** | SK Raina | 5604 |
| **116** | DA Warner | 5522 |
| **407** | S Dhawan | 5452 |
| **379** | RG Sharma | 5394 |

## Q-22 | who has hit the most no of '4's

In [64]:
```
batsman_4s =  df_[df_['total_runs'] == 4]
```

In [65]:
```
print('*'*15)
print(batsman_4s['batsman'].value_counts().head(1))
print('*'*15)
```

```
***************
S Dhawan    595
Name: batsman, dtype: int64
***************
```

## Q-23 | who has hit the mot no of '6's

In [66]:
```
batsman_6s = df_[df_['total_runs'] == 6]
```

In [67]:
```
print('*'*15)
print(batsman_6s['batsman'].value_counts().head(1))
print('*'*15)
```

```
***************
CH Gayle    344
Name: batsman, dtype: int64
***************
```

## Q-24 | who has the highest strikerate

In [68]:
```
df_['Strike Rate'] = (df_['total_runs'] / df_['ball']) * 100
```

In [69]:
```
max_strike_rate_player = df_.groupby('batsman')['Strike Rate'].max().reset_inc
```

In [70]:
```
highest_strike_rate_player = max_strike_rate_player.loc[max_strike_rate_player

print(f"The player with the highest strike rate is: {highest_strike_rate_playe
```

```
The player with the highest strike rate is: CH Morris with a strike rate of
700.00.
```

## Q-25 | who is leading wicket taker

In [71]:
```
wicket_taker = df_[df_['is_wicket'] == 1]
```

In [72]:
```
print('-'*27)
print(wicket_taker['bowler'].value_counts().head(1))
print('-'*27)
```

```
---------------------------
SL Malinga    188
Name: bowler, dtype: int64
---------------------------
```

## Q-26 | which stadium has hosted the most no of matches

In [73]:
```
df['venue'].value_counts().head()
```

Out[73]:
```
Eden Gardens                                77
Feroz Shah Kotla                            74
Wankhede Stadium                            73
M Chinnaswamy Stadium                       65
Rajiv Gandhi International Stadium, Uppal    64
Name: venue, dtype: int64
```

## Q-27 | who has won the most MOM (man of the match) awards

In [74]:
```
df['player_of_match'].value_counts().head()
```

Out[74]:
```
AB de Villiers    23
CH Gayle          22
RG Sharma         18
DA Warner         17
MS Dhoni          17
Name: player_of_match, dtype: int64
```

## Q-28 | what is the count of 4's hit in each season

In [75]:
```
df_['season'] = runs_in_each_season['season']
```

In [76]:
```
fours_data = df_[df_['total_runs'] == 4]

fours_per_season = fours_data.groupby('season')['total_runs'].count().reset_ir

print('='*31)
print('count of 4_s hit in each season')
print('='*31)
print(fours_per_season)
print('='*31)
```

```
===============================
count of 4_s hit in each season
===============================
   season  total_runs
0    2008        1726
1    2009        1337
2    2010        1728
3    2011        1950
4    2012        1932
5    2013        2081
6    2014        1590
7    2015        1628
8    2016        1643
9    2017        1624
10   2018        1673
11   2019        1681
12   2020        1594
===============================
```

In [ ]:

# Q-29 | what is the count of 6's hit in each season

In [77]:
```
sixs_data = df_[df_['total_runs'] == 6]

sixs_per_season = sixs_data.groupby('season')['total_runs'].count().reset_inde

print('='*31)
print('count of 6_s hit in each season')
print('='*31)
print(sixs_per_season)
print('='*31)
```

```
==============================
count of 6_s hit in each season
==============================
    season  total_runs
0     2008         618
1     2009         502
2     2010         583
3     2011         634
4     2012         728
5     2013         671
6     2014         710
7     2015         689
8     2016         638
9     2017         701
10    2018         869
11    2019         776
12    2020         731
==============================
```

# what is the count of runs scored by boundaries in each season

In [78]:
```
boundaries_data = df_[df_['batsman_runs'].isin([4, 6])]

boundaries_per_season = boundaries_data.groupby('season')['batsman_runs'].cou

print('='*31)
print('count of boundaries hit in each season')
print('='*31)
print(boundaries_per_season)
print('='*31)
```

```
===============================
count of boundaries hit in each season
===============================
   season  batsman_runs
0    2008          2326
1    2009          1823
2    2010          2293
3    2011          2555
4    2012          2644
5    2013          2727
6    2014          2276
7    2015          2299
8    2016          2272
9    2017          2316
10   2018          2524
11   2019          2437
12   2020          2318
===============================
```

## Q-30 | what is the run contribution from boundaries in each season

In [79]:
```
boundary_runs_per_season = boundaries_data.groupby('season')['batsman_runs'].s

print('='*47)
print('contribution from boundaries hit in each season')
print('='*47)
print(boundary_runs_per_season)
print('='*47)
```

```
===============================================
contribution from boundaries hit in each season
===============================================
   season  batsman_runs
0    2008         10550
1    2009          8304
2    2010         10342
3    2011         11498
4    2012         12042
5    2013         12258
6    2014         10532
7    2015         10580
8    2016         10366
9    2017         10674
10   2018         11840
11   2019         11316
12   2020         10742
===============================================
```

## Q-31 | which team has scored the most runs in the first 6 overs

In [80]:
```
powerplay_data = df_[df_['over'] <= 6]

runs_in_powerplay_per_team = powerplay_data.groupby('batting_team')['total_ru

most_runs_in_powerplay_team = runs_in_powerplay_per_team.loc[runs_in_powerplay

print(f"The team that has scored the most runs in the first 6 overs is: {most_
```

```
The team that has scored the most runs in the first 6 overs is: Mumbai India
ns with 10476 runs.combining in all seasons
```

## Q-32 | which team has scored the most runs in the last 4 overs

In [81]:
```
last_4_overs_data = df_[df_['over'] >= (df_['over'].max() - 3)]

runs_in_last_4_overs_per_team = last_4_overs_data.groupby('batting_team')['to

most_runs_in_last_4_overs_team = runs_in_last_4_overs_per_team.loc[runs_in_las

print(f"The team that has scored the most runs in the last 4 overs is: {most_
```

```
The team that has scored the most runs in the last 4 overs is: Mumbai Indian
s with 7538 runs.
```

## Q-33 | which team has the best scoring run rate in first 6 overs

In [82]:
```
powerplay_data = df_[df_['over'] <= 6].copy().dropna()

powerplay_data['run_rate'] = powerplay_data['total_runs']/powerplay_data['over

average_run_rate_per_team = powerplay_data.groupby('batting_team')['run_rate']

best_run_rate_team = average_run_rate_per_team.loc[average_run_rate_per_team[

print(f"The team with the best scoring run rate in the first 6 overs is: {best
```

```
The team with the best scoring run rate in the first 6 overs is: Delhi Dared
evils
```

# Q-34 | which team has the best scoring run rate in last 4 overs

In [83]:
```
last_4_overs_data = df_[df_['over'] >= (df_['over'].max() - 3)].copy().dropna(

last_4_overs_data['Run Rate'] = last_4_overs_data['total_runs'] / last_4_overs

average_run_rate_per_team = last_4_overs_data.groupby('batting_team')['Run Rat

best_run_rate_team = average_run_rate_per_team.loc[average_run_rate_per_team[

print(f"The team with the best scoring run rate in the last 4 overs is: {best_
```

```
The team with the best scoring run rate in the last 4 overs is: Rajasthan Ro
yals with an average run rate of 0.11 runs per over.
```