IPL Data Analysis Report

Index

1. Introduction

The Indian Premier League (IPL) is one of the most popular and competitive T20 cricket leagues in the world, attracting top players from across the globe. With a rich history of thrilling matches, dramatic finishes, and standout performances, the IPL generates a massive amount of data every season.

This project aims to explore and analyze IPL match data to extract meaningful insights using data science techniques. The focus is on understanding:

- How different teams have performed over the years
- Which players consistently deliver strong performances
- How factors like toss decisions, venues, and match outcomes are related

Using tools such as Python, Pandas, NumPy, Matplotlib, and Seaborn, this project performs:

- **Data Cleaning**: Preparing the dataset for analysis by handling null values, formatting columns, and merging datasets if needed.
- **Exploratory Data Analysis (EDA)**: Identifying patterns and trends in the data using visualizations and statistics.
- **Visualization**: Creating charts and graphs to better understand performance metrics and match statistics.

The analysis helps answer questions like:

- Which team has the highest number of wins?
- Who are the top batsmen and bowlers?
- Is there any advantage in winning the toss?
- What are the most successful venues?

Through this project, we aim to apply data science skills to a real-world dataset and demonstrate how data can be used to derive insights, support decision-making, and enhance our understanding of sports.

2. Dataset Description

This analysis uses the following datasets from Kaggle:

- `matches.csv` – Match-level details (season, teams, winner, toss, etc.)
- `deliveries.csv` – Ball-by-ball delivery data
- `players.csv` – Player details

- `teams.csv` – Team metadata
- `most_runs.csv` – Total runs by players
- `average_strike_rate.csv` – Player-wise average and strike rate

3. Tools and Technologies Used

- **Programming Language:** Python
- **Libraries:** Pandas, NumPy, Matplotlib, Seaborn, Plotly
- **IDE:** Jupyter Notebook / VS Code
- **Version Control:** Git & GitHub

4. Exploratory Data Analysis

Initial steps involved loading and inspecting the data:

```
import pandas as pd
matches = pd.read_csv("matches.csv")
deliveries = pd.read_csv("deliveries.csv")
```

Data was explored to understand dimensions, null values, and data types.

5. Data Cleaning and Preparation

- Removed null or irrelevant rows
- Ensured consistency in team and player names
- Created new columns like `toss_win_and_match_win`

6. Query-Based Analysis

Query 1: Total Matches Played

```
matches.shape[0]
```

output :

Total Matches Played: 756

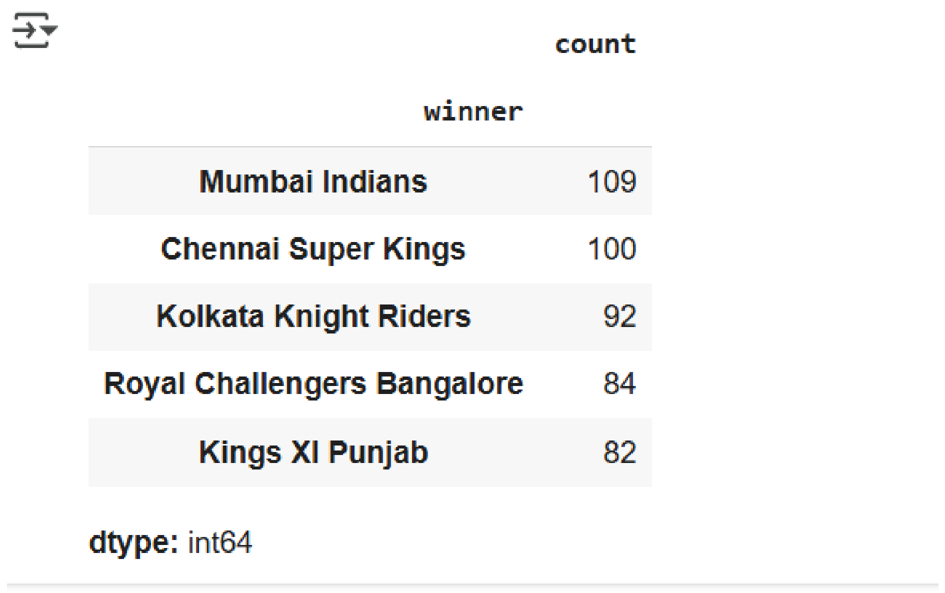
Query 2: Number of Teams Participated

```
matches['team1'].nunique()

Output: 15
```

Query 3: Most Wins by a Team

```
matches['winner'].value_counts().head(5)
```

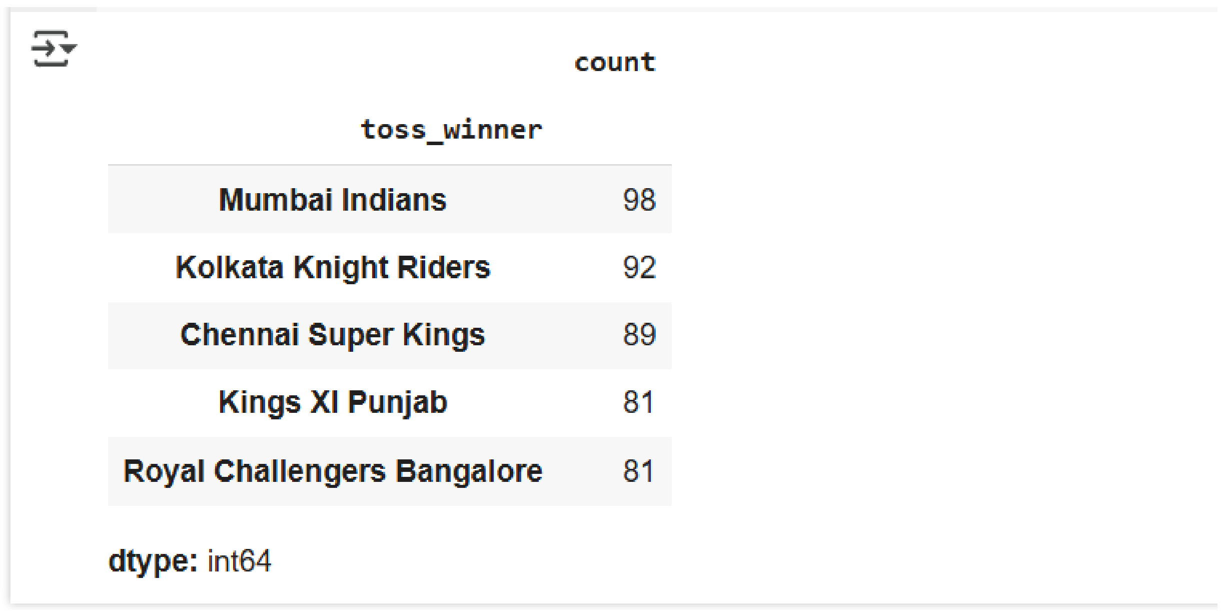


```
                             count
                  winner
         Mumbai Indians        109
     Chennai Super Kings       100
   Kolkata Knight Riders        92
Royal Challengers Bangalore     84
          Kings XI Punjab       82

dtype: int64
```

Query 4: Most Toss Wins

```
matches['toss_winner'].value_counts().head(5)
```

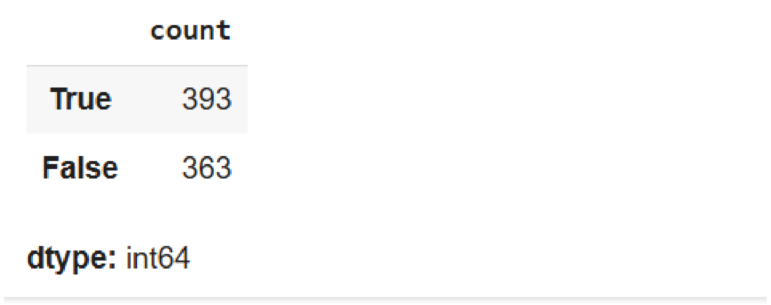


```
                              count
toss_winner
Mumbai Indians                   98
Kolkata Knight Riders            92
Chennai Super Kings              89
Kings XI Punjab                  81
Royal Challengers Bangalore      81

dtype: int64
```

Query 5: Toss Impact on Match Result

```
(matches['toss_winner'] == matches['winner']).value_counts()
```



```
       count
True     393
False    363

dtype: int64
```

Query 6: Home vs Away Wins

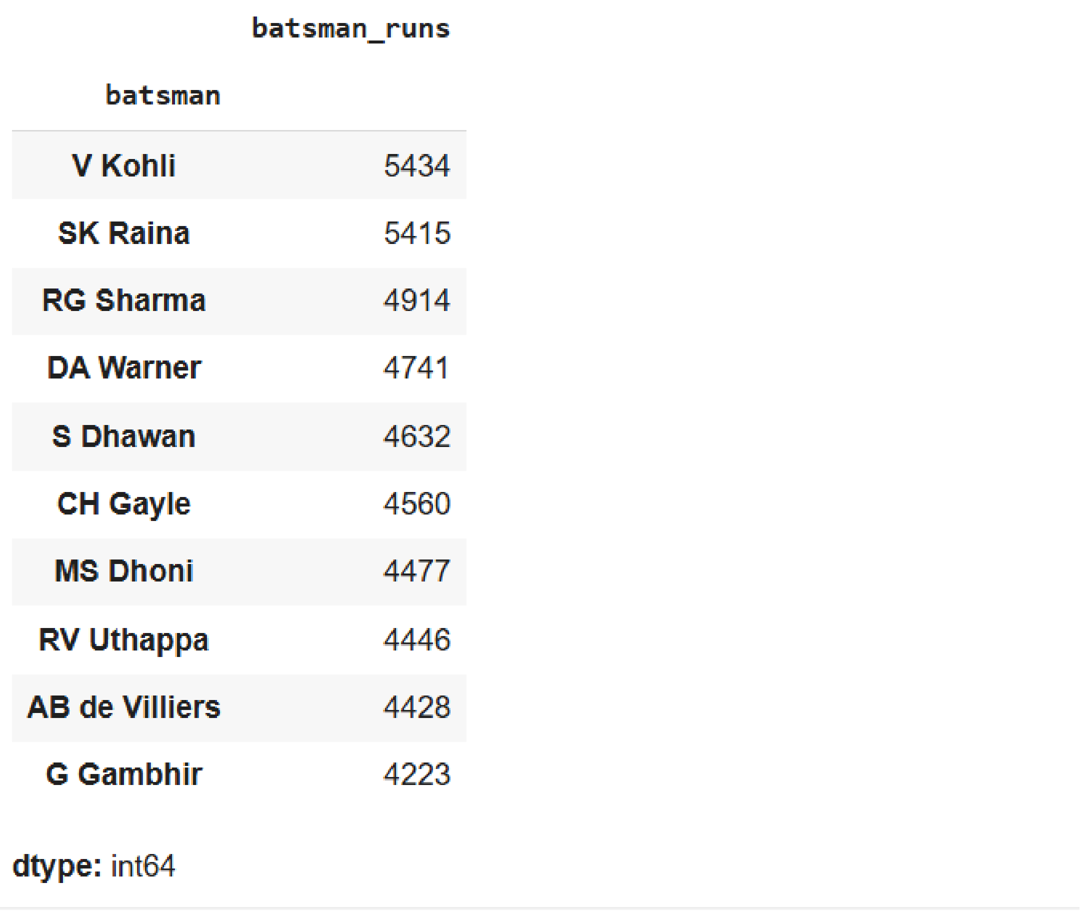
```
home_away = matches.groupby('team1')['winner'].value_counts()
```

Query 7: Top 10 Run Scorers

```
try:
    deliveries
except NameError:
    deliveries = pd.read_csv("deliveries.csv")
    # Ensure column names are lowercase if needed for consistency
    deliveries.columns = deliveries.columns.str.strip().str.lower()



deliveries.groupby('batsman')['batsman_runs'].sum().sort_values(ascending=False).
head(10)
```

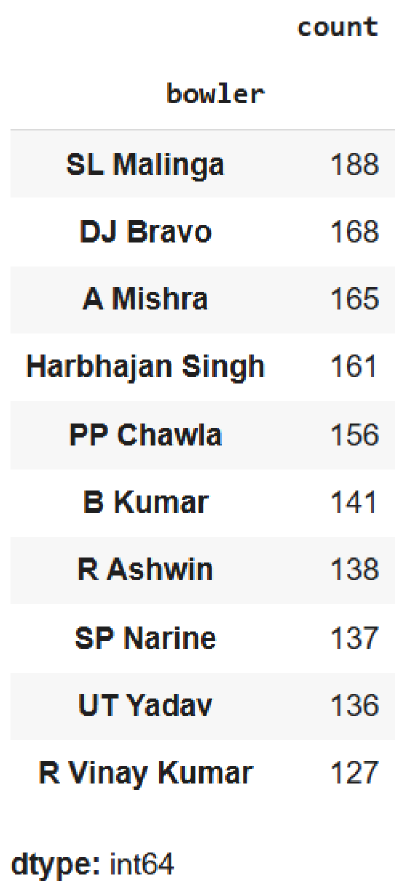


```
                 batsman_runs
      batsman
      V Kohli            5434
     SK Raina            5415
    RG Sharma            4914
    DA Warner            4741
    S Dhawan             4632
    CH Gayle             4560
    MS Dhoni             4477
   RV Uthappa            4446
AB de Villiers           4428
   G Gambhir             4223

dtype: int64
```

Query 8: Top 10 Wicket Takers

```
deliveries[deliveries['dismissal_kind'].notnull()]['bowler'].value_cou
nts().head(10)
```



	count
bowler	
SL Malinga	188
DJ Bravo	168
A Mishra	165
Harbhajan Singh	161
PP Chawla	156
B Kumar	141
R Ashwin	138
SP Narine	137
UT Yadav	136
R Vinay Kumar	127

dtype: int64

Query 9: Best Strike Rate (min 200 balls)

```
strike = deliveries.groupby('batsman').agg({'batsman_runs': 'sum',
'ball': 'count'})
strike['strike_rate'] = (strike['batsman_runs'] / strike['ball']) * 100
strike[strike['ball'] > 200].sort_values(by='strike_rate',
ascending=False).head(10)
```

batsman	batsman_runs	ball	strike_rate
AD Russell	1445	803	179.950187
SP Narine	803	481	166.943867
RR Pant	1792	1104	162.318841
J Bairstow	468	293	159.726962
GJ Maxwell	1403	902	155.543237
CH Morris	520	339	153.392330
HH Pandya	1118	736	151.902174
JC Buttler	1431	954	150.000000
V Sehwag	2728	1833	148.827059
AB de Villiers	4428	2977	148.740343

Query 10: Best Bowling Economies (min 100 balls)

```
eco = deliveries.groupby('bowler').agg({'total_runs': 'sum', 'ball':
'count'})
eco['economy'] = eco['total_runs'] / (eco['ball'] / 6)
eco[eco['ball'] > 100].sort_values('economy').head(10)
```

	total_runs	ball	economy
bowler			
Sohail Tanvir	275	265	6.226415
A Chandila	245	234	6.282051
FH Edwards	160	150	6.400000
L Ngidi	175	163	6.441718
SMSM Senanayake	211	195	6.492308
SM Pollock	307	280	6.578571
J Yadav	248	226	6.584071
A Kumble	1089	983	6.646999
DW Steyn	2454	2207	6.671500
GD McGrath	366	329	6.674772

Query 11: Most Man of the Match Awards

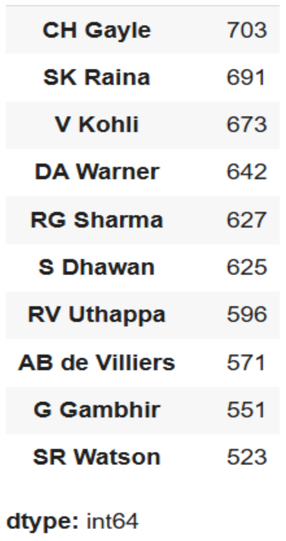
```
matches['player_of_match'].value_counts().head(10)
```

	count
player_of_match	
CH Gayle	21
AB de Villiers	20
MS Dhoni	17
RG Sharma	17
DA Warner	17
YK Pathan	16
SR Watson	15
SK Raina	14
G Gambhir	13
MEK Hussey	12

dtype: int64

Query 12: Most Boundaries (4s and 6s)

```
boundaries = deliveries[deliveries['batsman_runs'] >= 4]
boundaries['batsman'].value_counts().head(10)
```

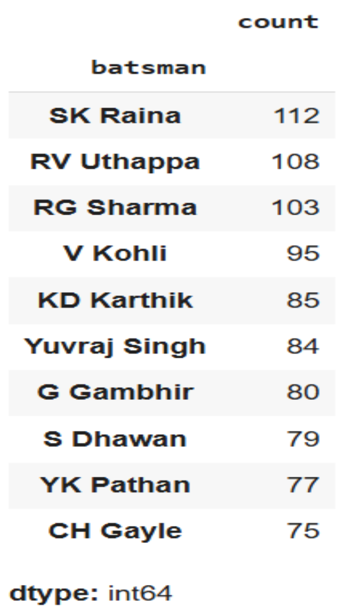


```
CH Gayle        703
SK Raina        691
V Kohli         673
DA Warner       642
RG Sharma       627
S Dhawan        625
RV Uthappa      596
AB de Villiers  571
G Gambhir       551
SR Watson       523

dtype: int64
```

Query 13: Most Ducks

```
ducks = deliveries[(deliveries['batsman_runs'] == 0) & (deliveries['dismissal_kind'] == 'caught')]
ducks['batsman'].value_counts().head(10)
```

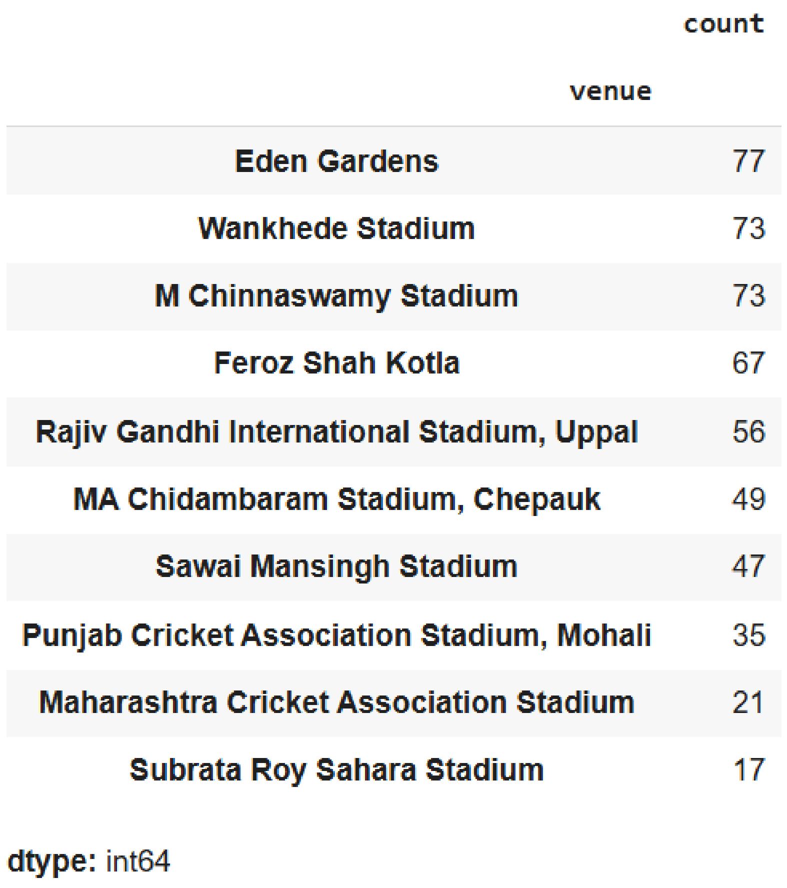


```
               count
    batsman
   SK Raina      112
 RV Uthappa      108
 RG Sharma       103
    V Kohli       95
 KD Karthik       85
Yuvraj Singh      84
 G Gambhir        80
  S Dhawan        79
  YK Pathan       77
  CH Gayle        75

dtype: int64
```

Query 14: Most Matches at a Venue

```
matches['venue'].value_counts().head(10)
```

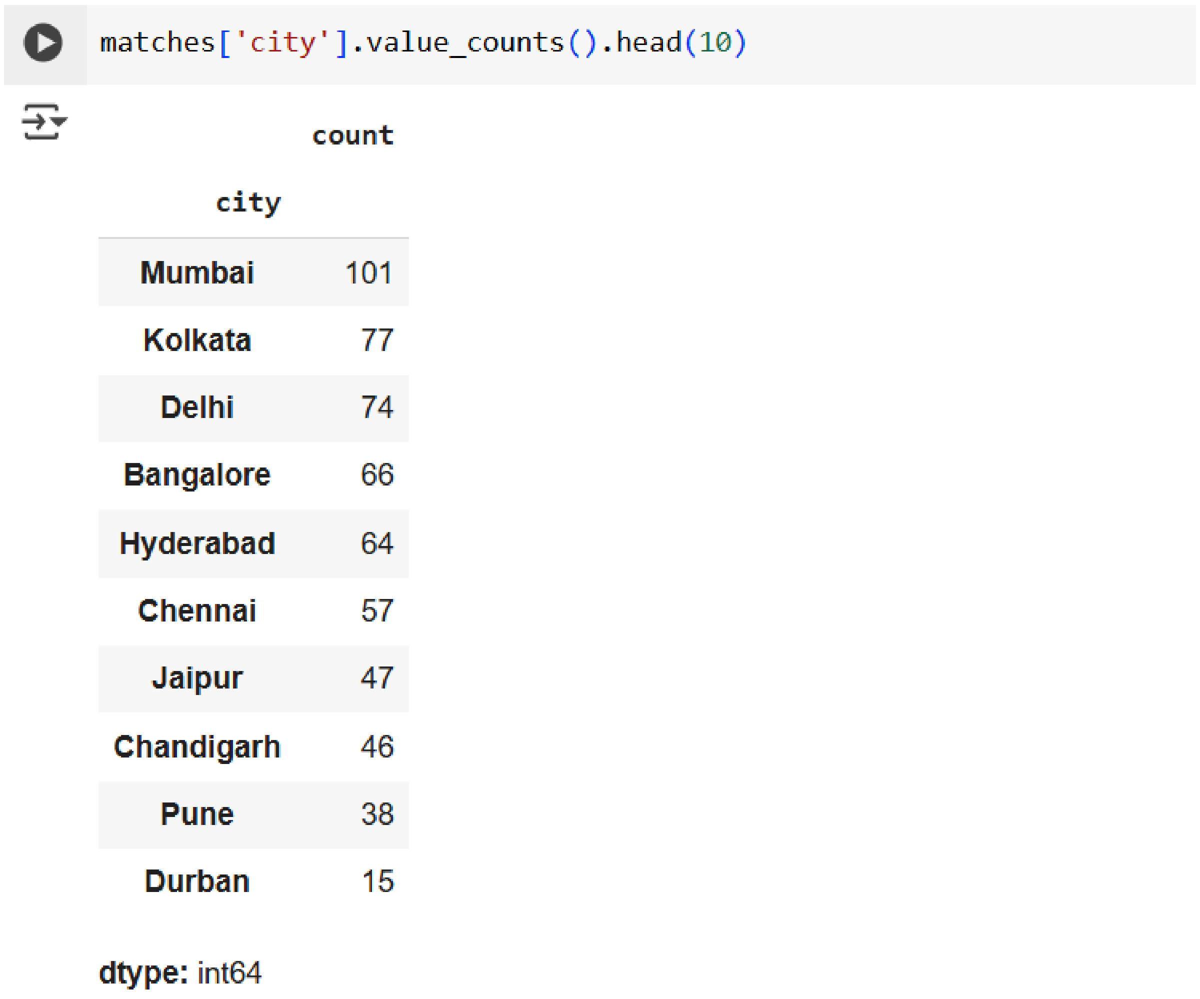


	count
venue	
Eden Gardens	77
Wankhede Stadium	73
M Chinnaswamy Stadium	73
Feroz Shah Kotla	67
Rajiv Gandhi International Stadium, Uppal	56
MA Chidambaram Stadium, Chepauk	49
Sawai Mansingh Stadium	47
Punjab Cricket Association Stadium, Mohali	35
Maharashtra Cricket Association Stadium	21
Subrata Roy Sahara Stadium	17

dtype: int64

Query 15: Cities with Most Matches

```
matches['city'].value_counts().head(10)
```



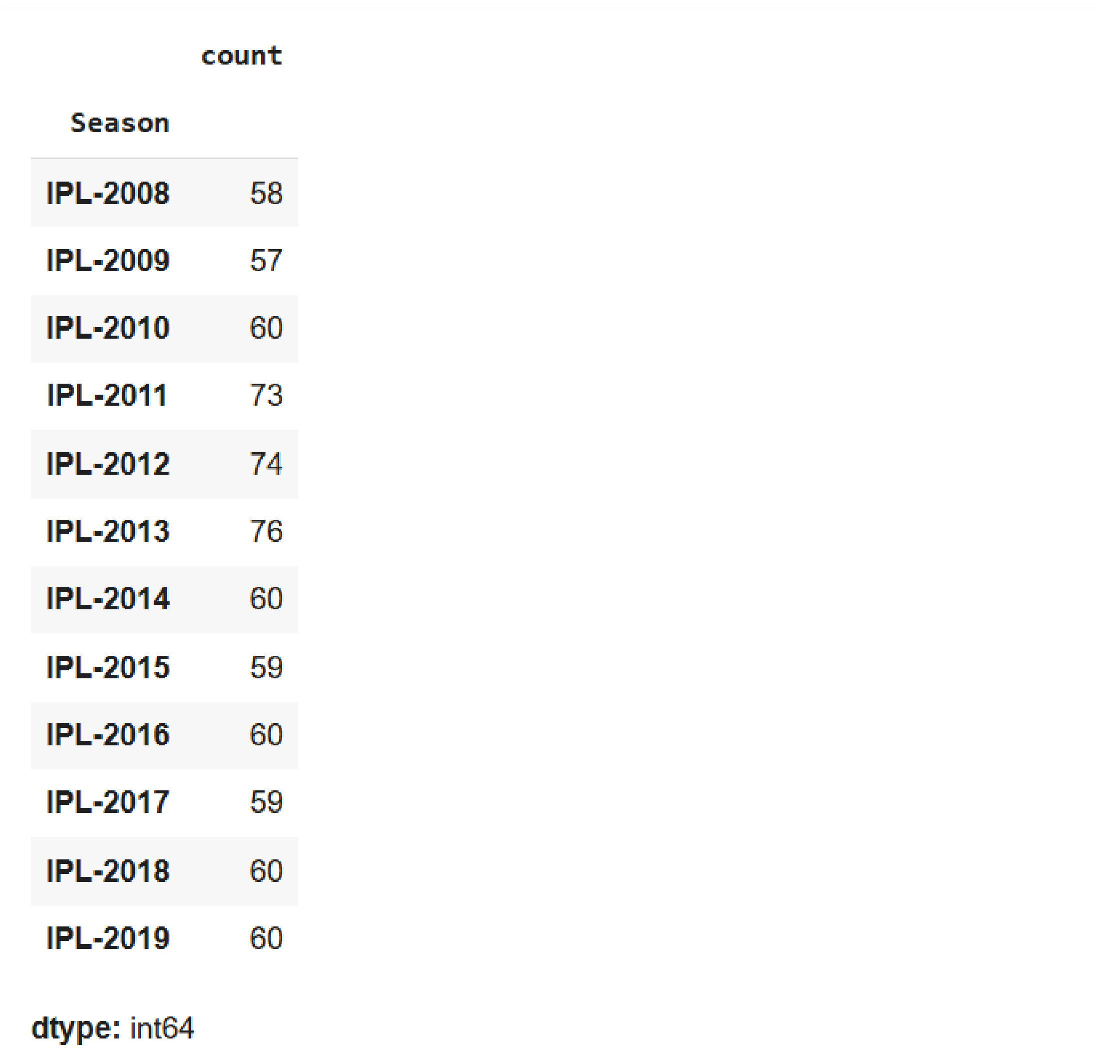
```python
matches['city'].value_counts().head(10)
```

```
                count
        city
      Mumbai      101
     Kolkata       77
       Delhi       74
   Bangalore       66
   Hyderabad       64
     Chennai       57
      Jaipur       47
  Chandigarh       46
        Pune       38
      Durban       15

dtype: int64
```

Query 16: Matches Per Season

```python
matches['season'].value_counts().sort_index()
```

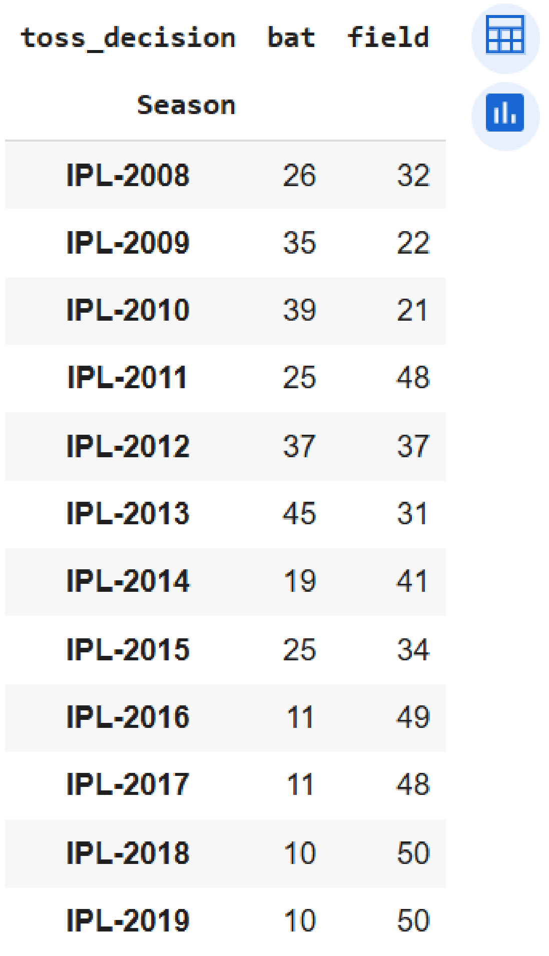


```
          count
  Season
IPL-2008     58
IPL-2009     57
IPL-2010     60
IPL-2011     73
IPL-2012     74
IPL-2013     76
IPL-2014     60
IPL-2015     59
IPL-2016     60
IPL-2017     59
IPL-2018     60
IPL-2019     60

dtype: int64
```

Query 17 : Toss Decision Trends Over Years

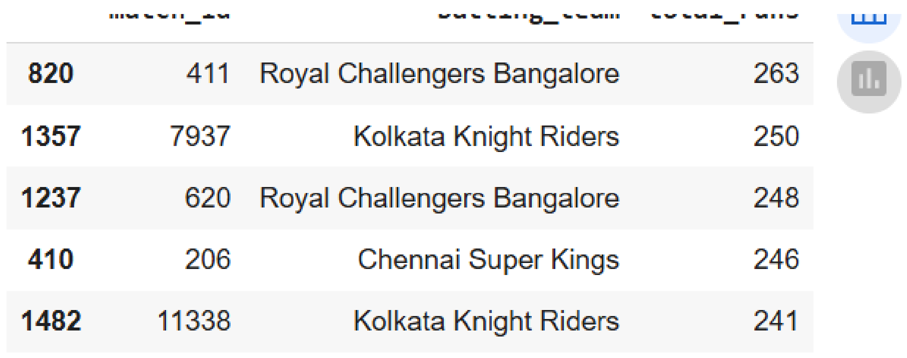
```
pd.crosstab(matches['season'], matches['toss_decision'])
```



toss_decision	bat	field
Season		
IPL-2008	26	32
IPL-2009	35	22
IPL-2010	39	21
IPL-2011	25	48
IPL-2012	37	37
IPL-2013	45	31
IPL-2014	19	41
IPL-2015	25	34
IPL-2016	11	49
IPL-2017	11	48
IPL-2018	10	50
IPL-2019	10	50

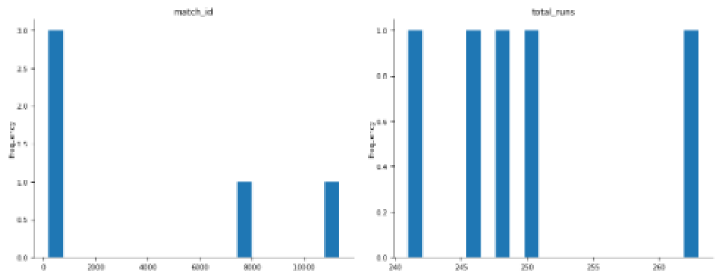
Query 18: Highest Team Score in a Match

```
match_runs = deliveries.groupby(['match_id', 'batting_team'])['total_runs'].sum().reset_index()
match_runs.sort_values('total_runs', ascending=False).head(5)
```

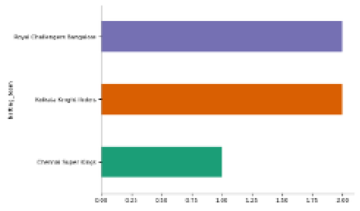


820	411	Royal Challengers Bangalore	263
1357	7937	Kolkata Knight Riders	250
1237	620	Royal Challengers Bangalore	248
410	206	Chennai Super Kings	246
1482	11338	Kolkata Knight Riders	241

Distributions

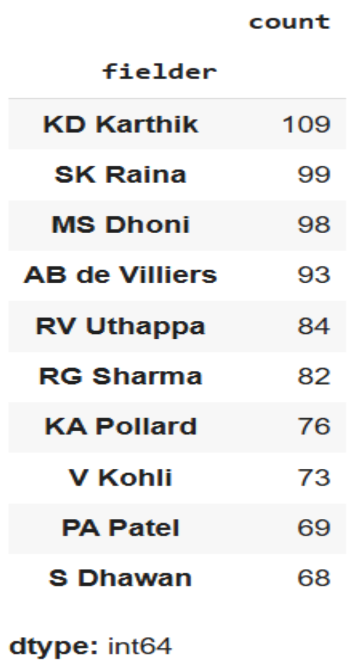


Categorical distributions



Query 19: Most Catches by a Player

```
catches = deliveries[deliveries['dismissal_kind'] == 'caught']
catches['fielder'].value_counts().head(10)
```



fielder	count
KD Karthik	109
SK Raina	99
MS Dhoni	98
AB de Villiers	93
RV Uthappa	84
RG Sharma	82
KA Pollard	76
V Kohli	73
PA Patel	69
S Dhawan	68

dtype: int64

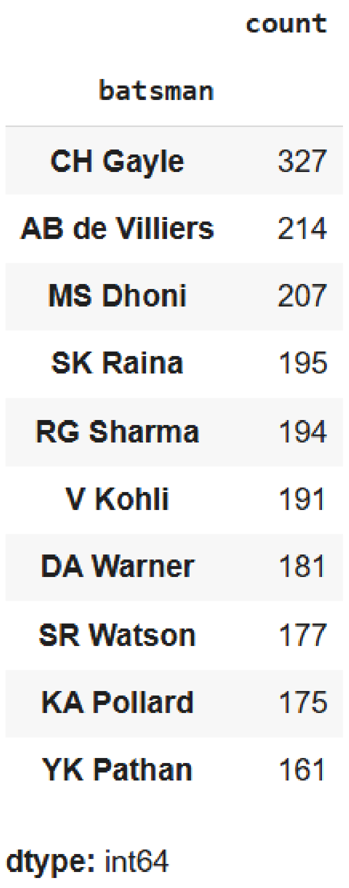
Query 20: Most Match Wins by Season

```
matches.groupby(['season','winner']).size().reset_index(name='wins').s
ort_values('wins', ascending=False).head(10)
```

	Season	winner	wins
6	IPL-2008	Rajasthan Royals	13
47	IPL-2013	Mumbai Indians	13
38	IPL-2012	Kolkata Knight Riders	12
43	IPL-2013	Chennai Super Kings	12
54	IPL-2014	Kings XI Punjab	12
80	IPL-2017	Mumbai Indians	12
49	IPL-2013	Rajasthan Royals	11
55	IPL-2014	Kolkata Knight Riders	11
36	IPL-2012	Delhi Daredevils	11
75	IPL-2016	Sunrisers Hyderabad	11

Query 21: Players with Most Sixes

```
deliveries[deliveries['batsman_runs']==6]['batsman'].value_counts().he
ad(10)
```

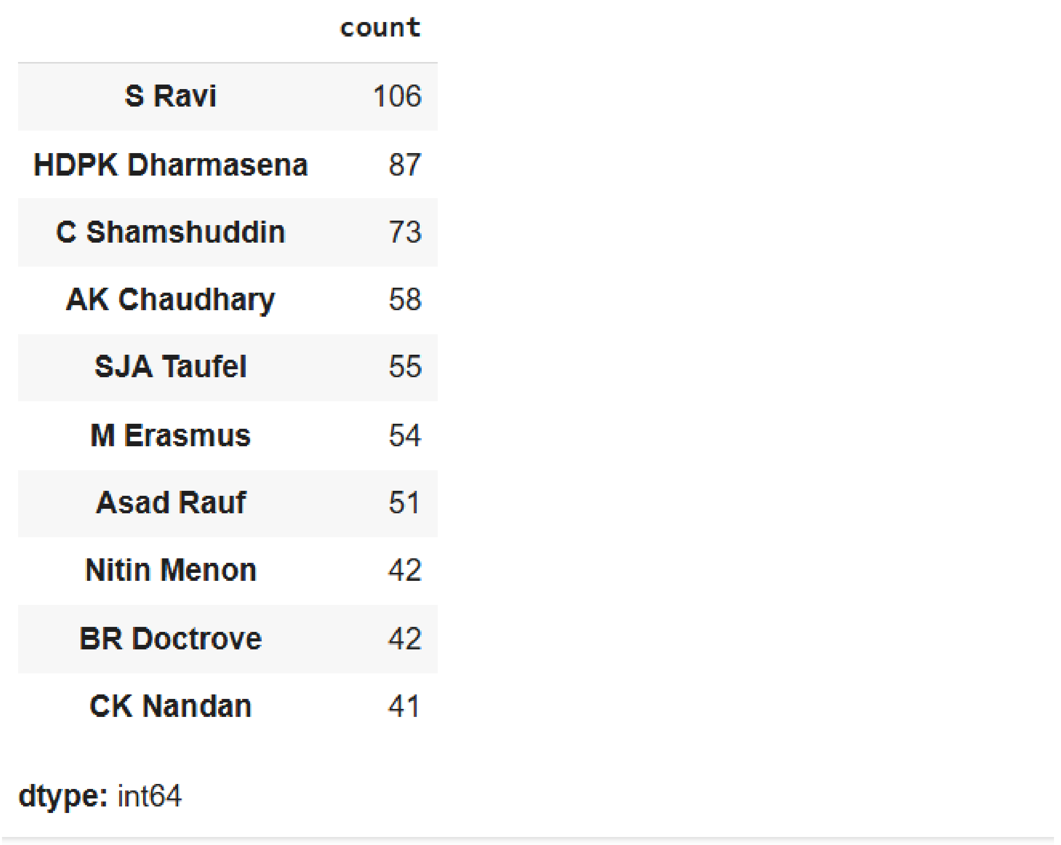


```
                count
batsman
CH Gayle          327
AB de Villiers    214
MS Dhoni          207
SK Raina          195
RG Sharma         194
V Kohli           191
DA Warner         181
SR Watson         177
KA Pollard        175
YK Pathan         161

dtype: int64
```

Query 22: Most Popular Umpires

```
umpires = pd.concat([matches['umpire1'], matches['umpire2']])
umpires.value_counts().head(10)
```

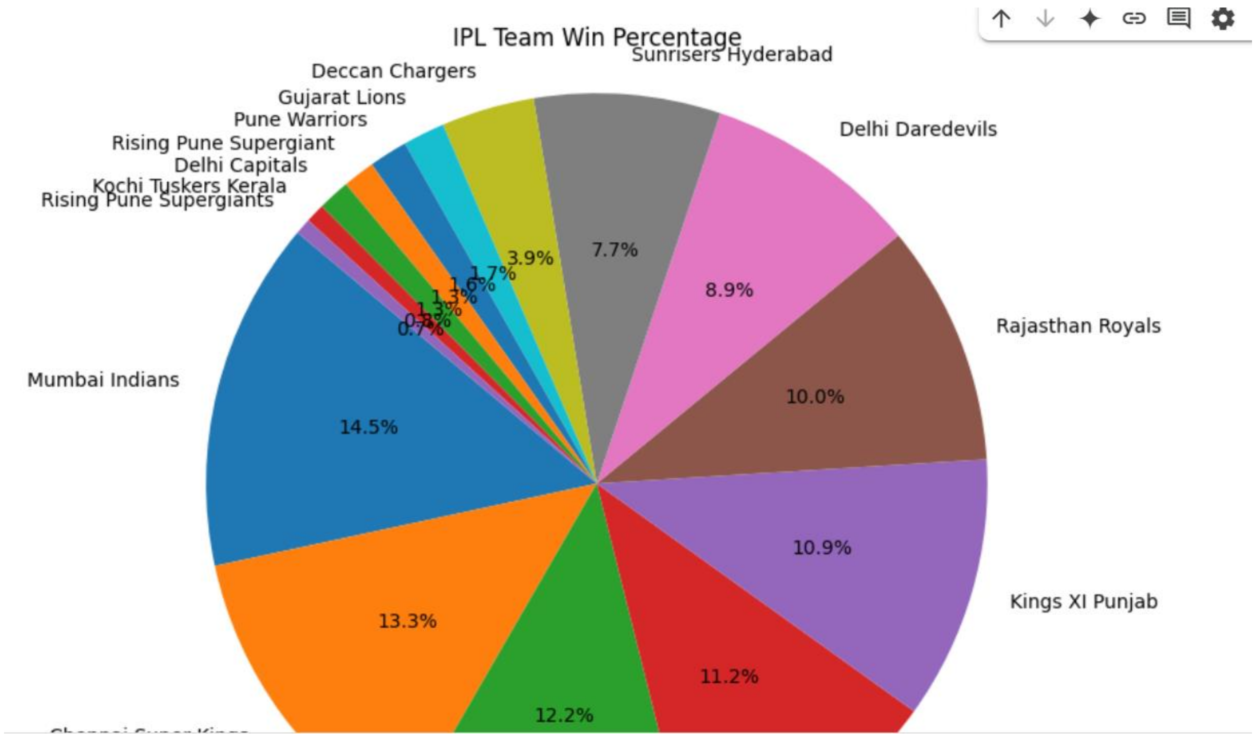


	count
S Ravi	106
HDPK Dharmasena	87
C Shamshuddin	73
AK Chaudhary	58
SJA Taufel	55
M Erasmus	54
Asad Rauf	51
Nitin Menon	42
BR Doctrove	42
CK Nandan	41

dtype: int64

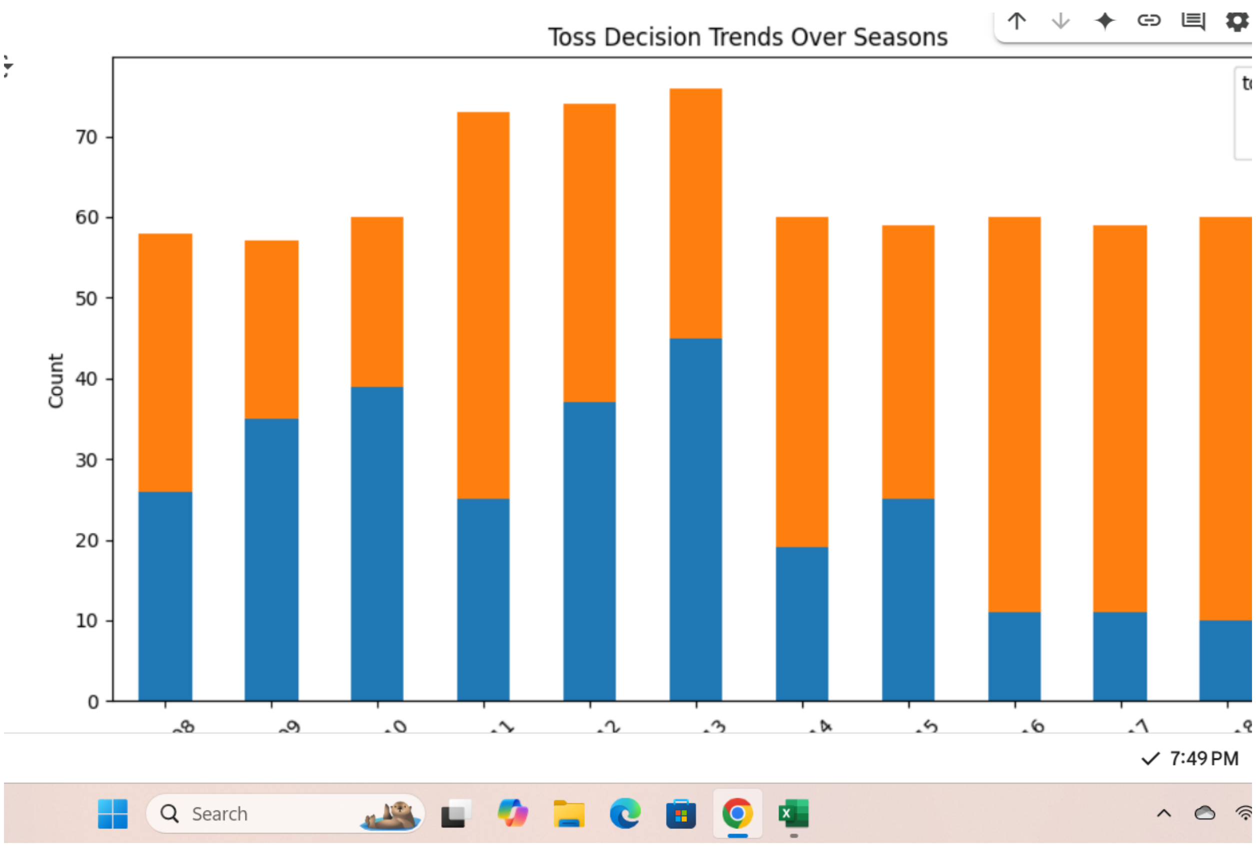
Query 23: Win Percentage of Each Team (Pie Chart)

```
import matplotlib.pyplot as plt
team_wins = matches['winner'].value_counts()
plt.figure(figsize=(8,8))
plt.pie(team_wins, labels=team_wins.index, autopct='%1.1f%%',
startangle=140)
plt.title('IPL Team Win Percentage')
plt.axis('equal')
plt.show()
```



Query 24: Toss Decision Trends Over Years (Stacked Bar Chart)

```
import seaborn as sns
import pandas as pd
cross_tab = pd.crosstab(matches['season'], matches['toss_decision'])
cross_tab.plot(kind='bar', stacked=True, figsize=(10,6))
plt.title('Toss Decision Trends Over Seasons')
plt.xlabel('Season')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

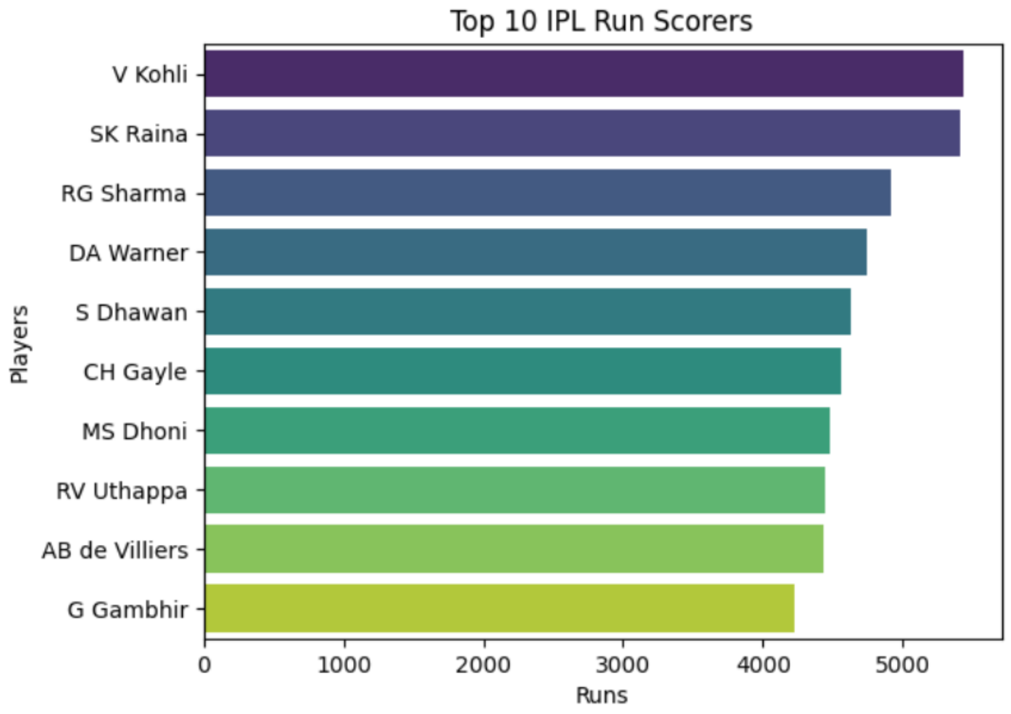


Query 25: Top Run Scorers (Bar Chart)

```
import seaborn as sns
runs =
deliveries.groupby('batsman')['batsman_runs'].sum().sort_values(ascend
ing=False).head(10)
sns.barplot(x=runs.values, y=runs.index, palette='viridis')
plt.title("Top 10 IPL Run Scorers")
plt.xlabel("Runs")
plt.ylabel("Players")
plt.show()
```

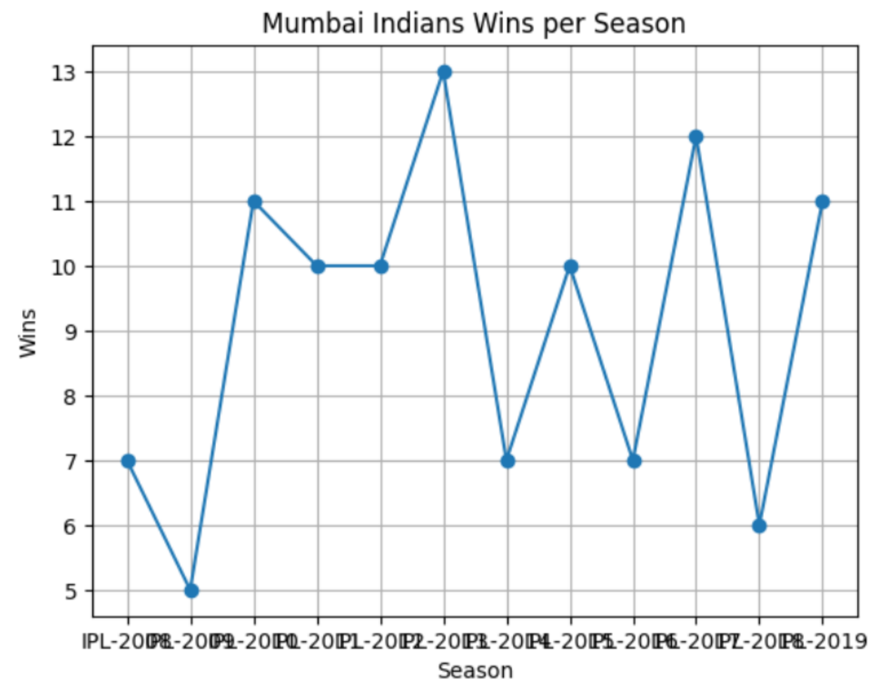
```
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the

    sns.barplot(x=runs.values, y=runs.index, palette='viridis')
```



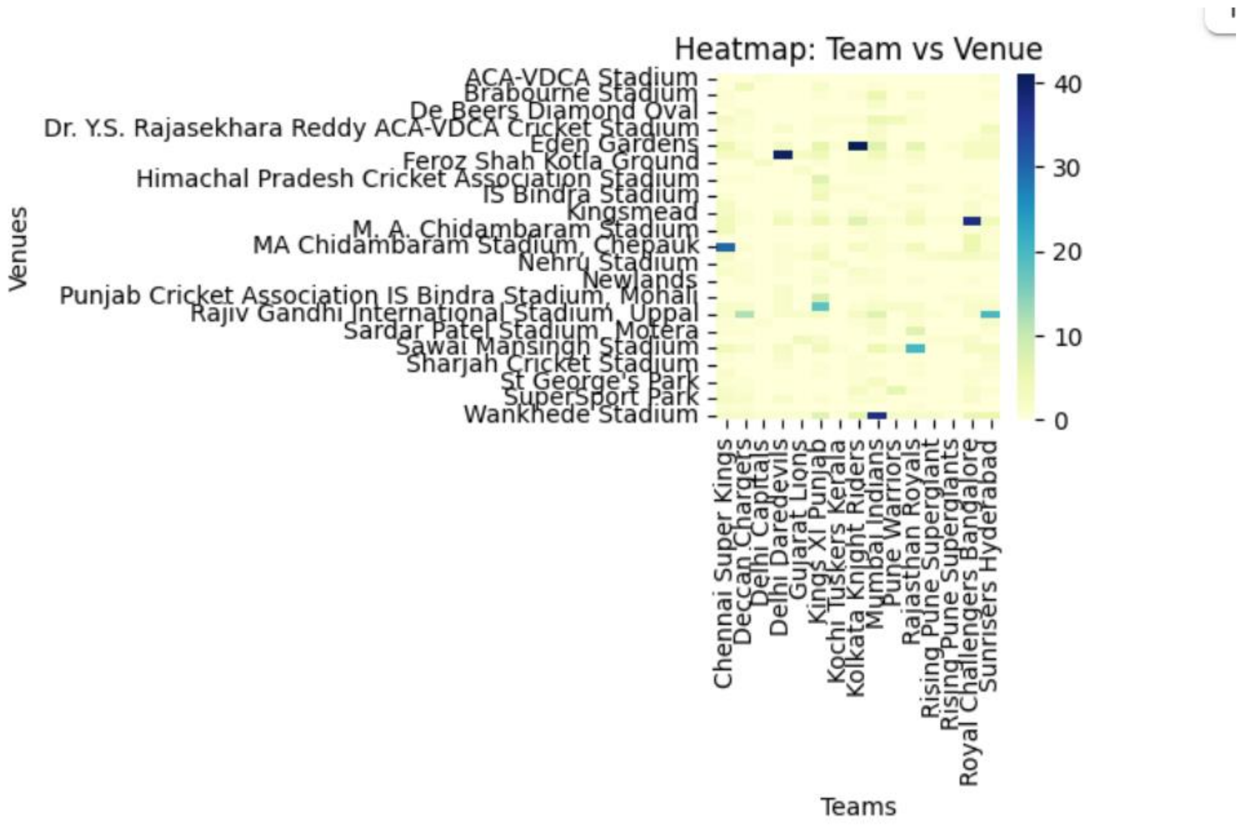
Query 26: Team Performance Over Years (Line Plot)

```
season_wins = matches.groupby(['season',
'winner']).size().reset_index(name='wins')
mi = season_wins[season_wins['winner'] == 'Mumbai Indians']
plt.plot(mi['season'], mi['wins'], marker='o')
plt.title('Mumbai Indians Wins per Season')
plt.xlabel('Season')
plt.ylabel('Wins')
plt.grid(True)
plt.show()
plt.grid(True)
plt.show()
```



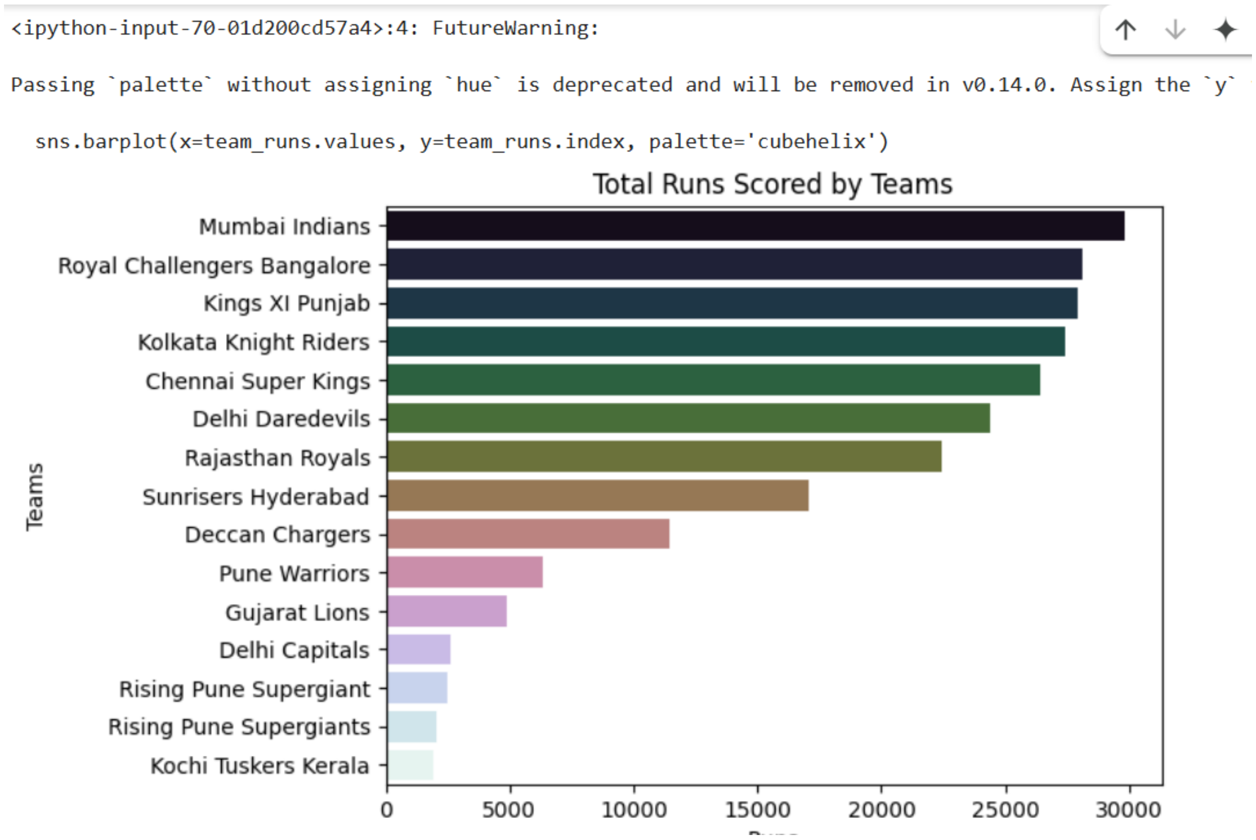
Query 27: Heatmap of Team vs Venue Matches

```
heat_data = pd.crosstab(matches['venue'], matches['team1'])
sns.heatmap(heat_data, cmap='YlGnBu')
plt.title('Heatmap: Team vs Venue')
plt.xlabel('Teams')
plt.ylabel('Venues')
plt.tight_layout()
plt.show()
```



Query 28: Total Runs by Each Team (Bar Plot)

```
team_runs                                                                =
deliveries.groupby('batting_team')['total_runs'].sum().sort_values(asc
ending=False)
sns.barplot(x=team_runs.values, y=team_runs.index, palette='cubehelix')
plt.title('Total        Runs        Scored        by        Teams')
plt.xlabel('Runs')
plt.ylabel('Teams')
plt.show()
```



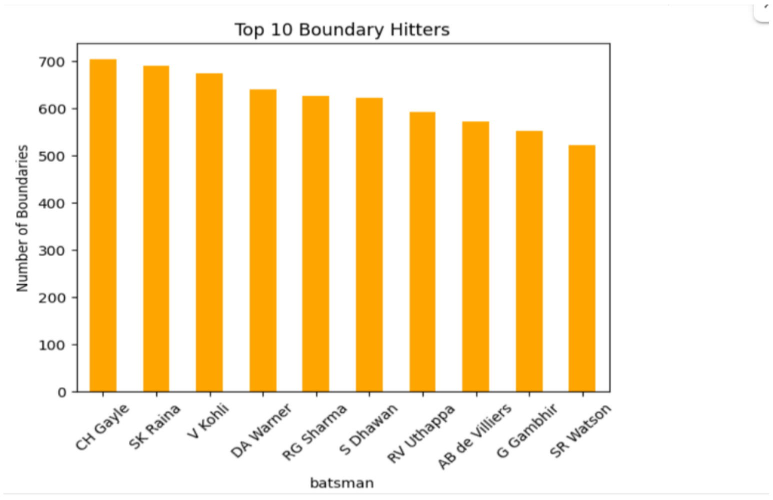
```
<ipython-input-70-01d200cd57a4>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y`

  sns.barplot(x=team_runs.values, y=team_runs.index, palette='cubehelix')
```

Query 29: Boundary Distribution by Player (4s and 6s)

```
boundaries = deliveries[deliveries['batsman_runs'].isin([4,6])]
counts =
boundaries.groupby('batsman')['batsman_runs'].count().sort_values(asce
nding=False).head(10)
counts.plot(kind='bar', color='orange')
plt.title('Top 10 Boundary Hitters')
plt.ylabel('Number of Boundaries')
plt.xticks(rotation=45)
```



```
plt.show()
```

7. Visualizations

Visuals generated include:

- Bar Charts for top scorers
- Pie Charts for win distributions

- Line Graphs for seasonal trends
- Heatmaps for strike rates
-
-
-

8. Key Insights

- Toss does not strongly correlate with wins
- Mumbai Indians & CSK are the most successful teams
- Consistent high performers include Kohli, Bravo, Dhoni

9. Conclusion

This analysis presents a deep dive into IPL history and performance metrics. The combination of statistical analysis and visualizations makes it easier to understand match dynamics and player contributions.

10. Future Scope

- Build a predictive model for match outcomes
- Create a Streamlit web dashboard

- Add comparative player reports across years

11. References

- IPL Dataset on Kaggle
- Python Docs, Pandas Docs, Matplotlib & Seaborn Docs

Author: Chanchal Gupta

Role: B.sc Data Science Student