Title: IPL Performance Analysis

Objective

The objective of this project is to explore detailed IPL match and ball-by-ball datasets to identify performance patterns, player statistics, team success rates, and venue-based insights. The analysis provides actionable trends to understand key success factors in IPL matches.

Tools & Technologies Used

- Python
- Pandas
- Numpy
- Matplotlib & Seaborn
- Jupyter Notebook

Dataset Overview

The project utilizes two comprehensive Kaggle datasets:

Match Data (2008 - 2023) — Match-wise records including teams, results, toss decisions, venue, margin of victory, etc.
Ball by Ball Data (2008 - 2023) — Delivery-wise details such as batsman, bowler, runs scored, wickets taken, dismissal types, etc.

These datasets enable player-level, team-level, and venue-level performance analysis across multiple IPL seasons.

Data Exploration & Visual Analysis

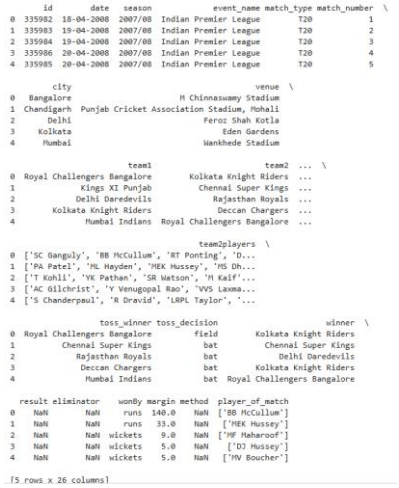
import pandas as pd

import numpy as np

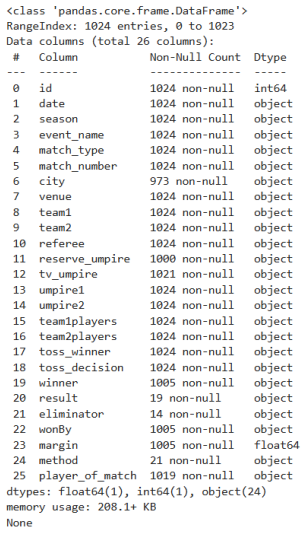
import matplotlib.pyplot as plt

import seaborn as sns

```
sns.set_style('whitegrid')

plt.rcParams['figure.figsize'] = (10,6)

# Load Data

match_info = pd.read_csv(r'E:/ipl-data-analysis/data/ipl_match_data_2008-2023.csv')

ball_data = pd.read_csv(r'E:/ipl-data-analysis/data/ipl_ball_by_ball_data_2008-2023.csv', low_memory=False)

# Basic Checks

print(match_info.head())

print(match_info.info())
```

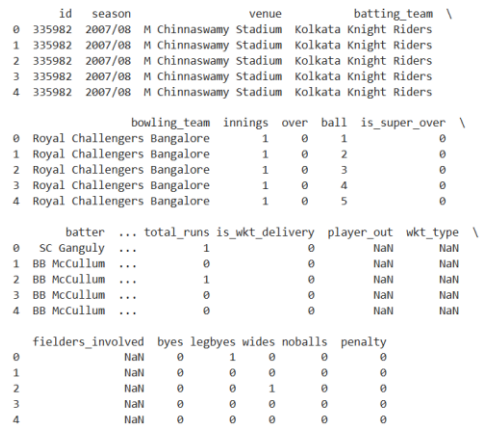


```
      id       date   season            event_name match_type match_number  \
0  335982  18-04-2008  2007/08  Indian Premier League        T20            1
1  335983  19-04-2008  2007/08  Indian Premier League        T20            2
2  335984  19-04-2008  2007/08  Indian Premier League        T20            3
3  335986  20-04-2008  2007/08  Indian Premier League        T20            4
4  335985  20-04-2008  2007/08  Indian Premier League        T20            5

        city                                venue  \
0  Bangalore                 M Chinnaswamy Stadium
1  Chandigarh  Punjab Cricket Association Stadium, Mohali
2      Delhi                      Feroz Shah Kotla
3    Kolkata                        Eden Gardens
4     Mumbai                      Wankhede Stadium

                         team1                  team2  ... \
0  Royal Challengers Bangalore  Kolkata Knight Riders  ...
1            Kings XI Punjab    Chennai Super Kings  ...
2             Delhi Daredevils      Rajasthan Royals  ...
3        Kolkata Knight Riders     Deccan Chargers  ...
4               Mumbai Indians  Royal Challengers Bangalore  ...

                                        team2players  \
0  ['SC Ganguly', 'BB McCullum', 'RT Ponting', 'D...
1  ['PA Patel', 'ML Hayden', 'MEK Hussey', 'MS Dh...
2  ['T Kohli', 'YK Pathan', 'SR Watson', 'M Kaif'...
3  ['AC Gilchrist', 'Y Venugopal Rao', 'VVS Laxma...
4  ['S Chanderpaul', 'R Dravid', 'LRPL Taylor', '...

                 toss_winner toss_decision                    winner  \
0  Royal Challengers Bangalore       field      Kolkata Knight Riders
1          Chennai Super Kings         bat        Chennai Super Kings
2            Rajasthan Royals         bat          Delhi Daredevils
3             Deccan Chargers         bat      Kolkata Knight Riders
4              Mumbai Indians         bat  Royal Challengers Bangalore

  result eliminator    wonBy margin method  player_of_match
0    NaN        NaN     runs  140.0    NaN  ['BB McCullum']
1    NaN        NaN     runs   33.0    NaN   ['MEK Hussey']
2    NaN        NaN  wickets    9.0    NaN  ['MF Maharoof']
3    NaN        NaN  wickets    5.0    NaN    ['DJ Hussey']
4    NaN        NaN  wickets    5.0    NaN  ['MV Boucher']

[5 rows x 26 columns]
```

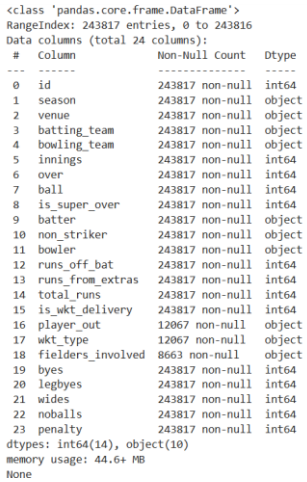


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               1024 non-null   int64
 1   date             1024 non-null   object
 2   season           1024 non-null   object
 3   event_name       1024 non-null   object
 4   match_type       1024 non-null   object
 5   match_number     1024 non-null   object
 6   city             973 non-null    object
 7   venue            1024 non-null   object
 8   team1            1024 non-null   object
 9   team2            1024 non-null   object
 10  referee          1024 non-null   object
 11  reserve_umpire   1000 non-null   object
 12  tv_umpire        1021 non-null   object
 13  umpire1          1024 non-null   object
 14  umpire2          1024 non-null   object
 15  team1players     1024 non-null   object
 16  team2players     1024 non-null   object
 17  toss_winner      1024 non-null   object
 18  toss_decision    1024 non-null   object
 19  winner           1005 non-null   object
 20  result           19 non-null     object
 21  eliminator       14 non-null     object
 22  wonBy            1005 non-null   object
 23  margin           1005 non-null   float64
 24  method           21 non-null     object
 25  player_of_match  1019 non-null   object
dtypes: float64(1), int64(1), object(24)
memory usage: 208.1+ KB
None
```

```
print(ball_data.head())

print(ball_data.info())
```

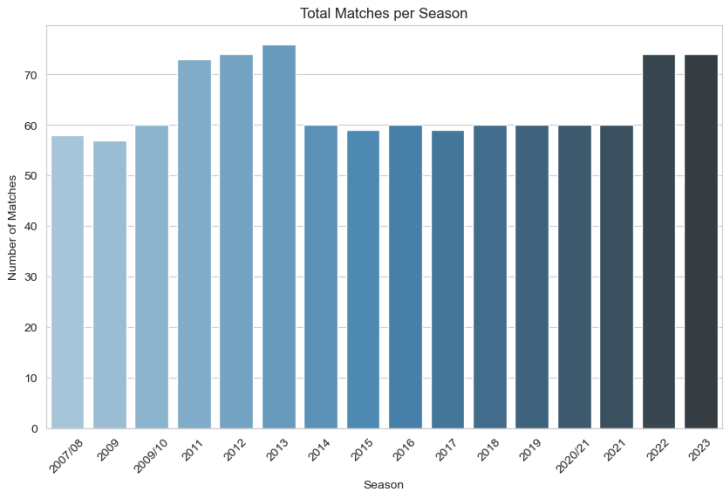


```
      id   season                venue           batting_team  \
0  335982  2007/08  M Chinnaswamy Stadium  Kolkata Knight Riders
1  335982  2007/08  M Chinnaswamy Stadium  Kolkata Knight Riders
2  335982  2007/08  M Chinnaswamy Stadium  Kolkata Knight Riders
3  335982  2007/08  M Chinnaswamy Stadium  Kolkata Knight Riders
4  335982  2007/08  M Chinnaswamy Stadium  Kolkata Knight Riders

                bowling_team  innings  over  ball  is_super_over  \
0  Royal Challengers Bangalore        1     0     1              0
1  Royal Challengers Bangalore        1     0     2              0
2  Royal Challengers Bangalore        1     0     3              0
3  Royal Challengers Bangalore        1     0     4              0
4  Royal Challengers Bangalore        1     0     5              0

       batter  ... total_runs is_wkt_delivery  player_out  wkt_type  \
0  SC Ganguly  ...          1               0         NaN       NaN
1  BB McCullum  ...          0               0         NaN       NaN
2  BB McCullum  ...          1               0         NaN       NaN
3  BB McCullum  ...          0               0         NaN       NaN
4  BB McCullum  ...          0               0         NaN       NaN

  fielders_involved  byes legbyes wides noballs  penalty
0               NaN     0       1     0       0        0
1               NaN     0       0     0       0        0
2               NaN     0       0     1       0        0
3               NaN     0       0     0       0        0
4               NaN     0       0     0       0        0

[5 rows x 24 columns]
```



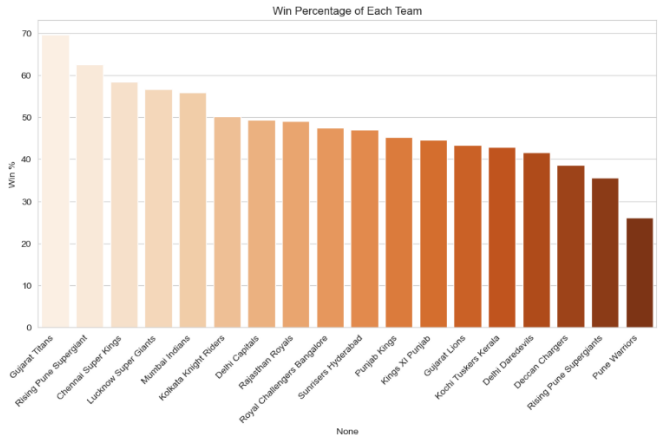
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243817 entries, 0 to 243816
Data columns (total 24 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 243817 non-null  int64
 1   season             243817 non-null  object
 2   venue              243817 non-null  object
 3   batting_team       243817 non-null  object
 4   bowling_team       243817 non-null  object
 5   innings            243817 non-null  int64
 6   over               243817 non-null  int64
 7   ball               243817 non-null  int64
 8   is_super_over      243817 non-null  int64
 9   batter             243817 non-null  object
 10  non_striker        243817 non-null  object
 11  bowler             243817 non-null  object
 12  runs_off_bat       243817 non-null  int64
 13  runs_from_extras   243817 non-null  int64
 14  total_runs         243817 non-null  int64
 15  is_wkt_delivery    243817 non-null  int64
 16  player_out         12067 non-null   object
 17  wkt_type           12067 non-null   object
 18  fielders_involved  8663 non-null    object
 19  byes               243817 non-null  int64
 20  legbyes            243817 non-null  int64
 21  wides              243817 non-null  int64
 22  noballs            243817 non-null  int64
 23  penalty            243817 non-null  int64
dtypes: int64(14), object(10)
memory usage: 44.6+ MB
None
```

```
# Column Cleanup
match_info.columns = match_info.columns.str.strip().str.lower()
ball_data.columns = ball_data.columns.str.strip().str.lower()
# Merge Datasets
merged_data = pd.merge(ball_data, match_info, left_on='id', right_on='id', how='left')
# visualization
# Total Matches per Season
plt.figure(figsize=(10,6))
matches_per_season = match_info['season'].value_counts().sort_index()
sns.barplot(x=matches_per_season.index, y=matches_per_season.values, palette='Blues_d')
plt.title("Total Matches per Season")
plt.xlabel("Season")
plt.ylabel("Number of Matches")
plt.xticks(rotation=45)
plt.savefig(r'E:/ipl-data-analysis/output/graphs/total_matches_per_season.png',
bbox_inches='tight')
plt.show()
```



Win % of Each Team

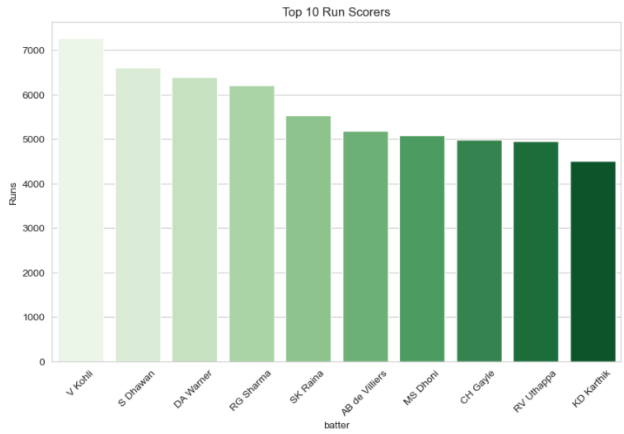
```
plt.figure(figsize=(12,6))

total_matches = match_info['team1'].value_counts() + match_info['team2'].value_counts()

wins = match_info['winner'].value_counts()

win_percentage = (wins / total_matches) * 100

win_percentage = win_percentage.dropna().sort_values(ascending=False)

sns.barplot(x=win_percentage.index, y=win_percentage.values, palette='Oranges')

plt.title("Win Percentage of Each Team")

plt.ylabel("Win %")

plt.xticks(rotation=45, ha='right')

plt.savefig(r'E:/ipl-data-analysis/output/graphs/win_percentage_each_team.png', bbox_inches='tight')

plt.show()
```



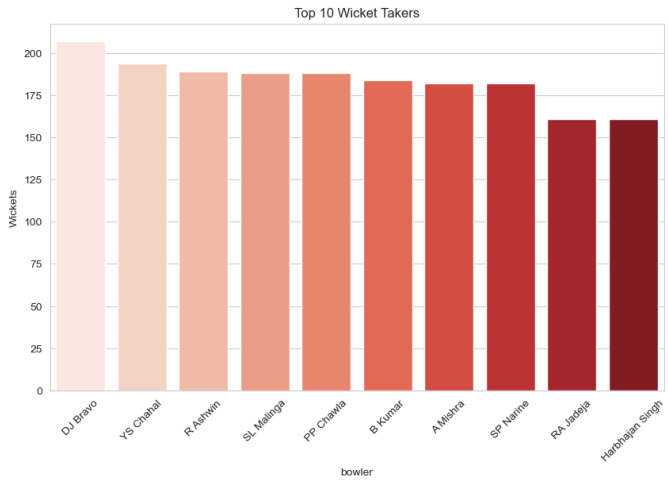
Top Run Scorers

```
plt.figure(figsize=(10,6))

top_scorers = merged_data.groupby('batter')['runs_off_bat'].sum().sort_values(ascending=False).head(10)

sns.barplot(x=top_scorers.index, y=top_scorers.values, palette='Greens')

plt.title("Top 10 Run Scorers")

plt.ylabel("Runs")

plt.xticks(rotation=45)
```

```
plt.savefig(r'E:/ipl-data-analysis/output/graphs/top_10_run_scorers.png', bbox_inches='tight')
plt.show()
```

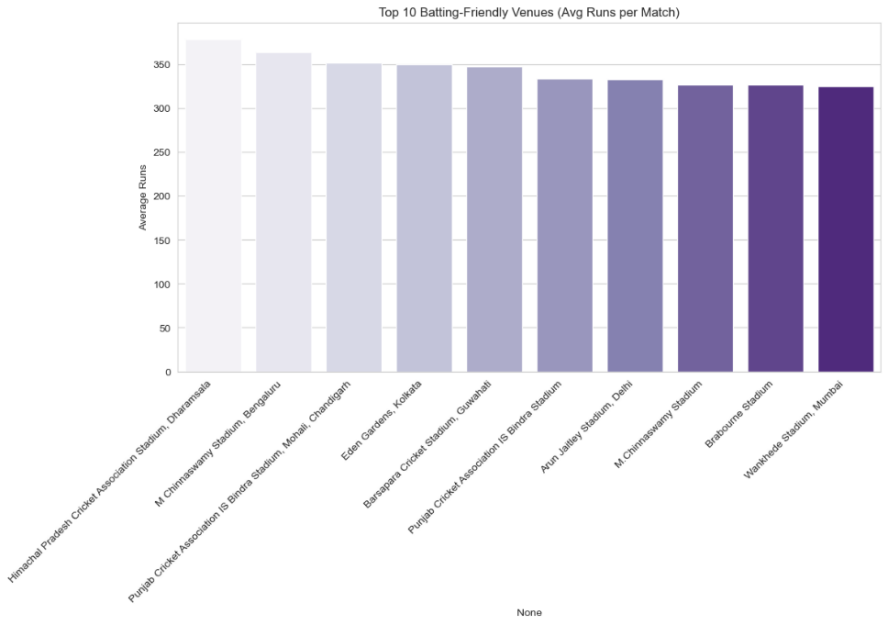



```
# Most Wickets Taken
plt.figure(figsize=(10,6))
wickets = merged_data[merged_data['wkt_type'].notnull()]
top_wicket_takers = wickets['bowler'].value_counts().head(10)
sns.barplot(x=top_wicket_takers.index, y=top_wicket_takers.values, palette='Reds')
plt.title("Top 10 Wicket Takers")
plt.ylabel("Wickets")
plt.xticks(rotation=45)
plt.savefig(r'E:/ipl-data-analysis/output/graphs/top_10_wicket_takers.png', bbox_inches='tight')
plt.show()
```



Venue Analysis - Batting Friendly Venues

```
plt.figure(figsize=(12,6))
venue_runs = merged_data.groupby('venue_x')['runs_off_bat'].sum()
venue_matches = match_info['venue'].value_counts()
avg_runs_per_venue = (venue_runs / venue_matches).sort_values(ascending=False).head(10)
sns.barplot(x=avg_runs_per_venue.index, y=avg_runs_per_venue.values, palette='Purples')
plt.title("Top 10 Batting-Friendly Venues (Avg Runs per Match)")
plt.ylabel("Average Runs")
plt.xticks(rotation=45, ha='right')
plt.savefig(r'E:/ipl-data-analysis/output/graphs/top_10_batting_friendly_venues.png', bbox_inches='tight')
plt.show()
```



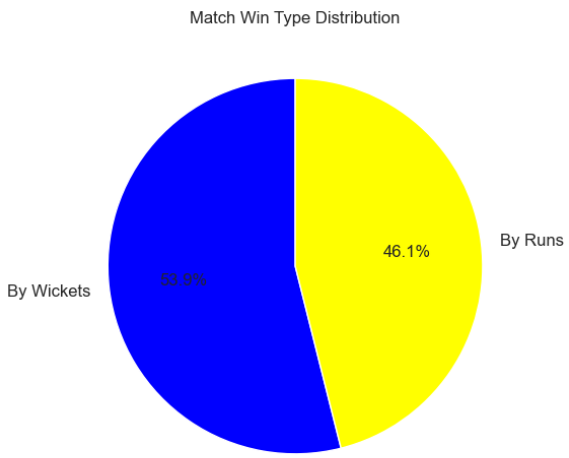
Toss Impact

```
toss_wins = match_info[match_info['toss_winner'] == match_info['winner']]
toss_effect = (len(toss_wins) / len(match_info)) * 100
print(f"Toss winning team also won the match in {toss_effect:.2f}% cases.")
```

```
Toss winning team also won the match in 50.49% cases.
```

Match Result Trends

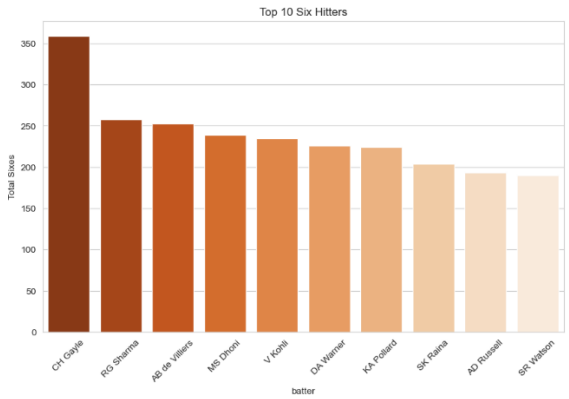
```
plt.figure(figsize=(6,6))

result_data = match_info[['wonby', 'margin']].copy()

result_data = result_data.dropna()

result_data['win_type'] = result_data['wonby'].apply(lambda x: 'By Runs' if str(x).lower() == 'runs' else 'By Wickets')

result_data['win_type'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['blue', 'yellow'], startangle=90, textprops={'fontsize': 12})

plt.title("Match Win Type Distribution")

plt.ylabel("")

plt.savefig(r'E:/ipl-data-analysis/output/graphs/match_win_type_distribution.png', bbox_inches='tight')

plt.show()
```



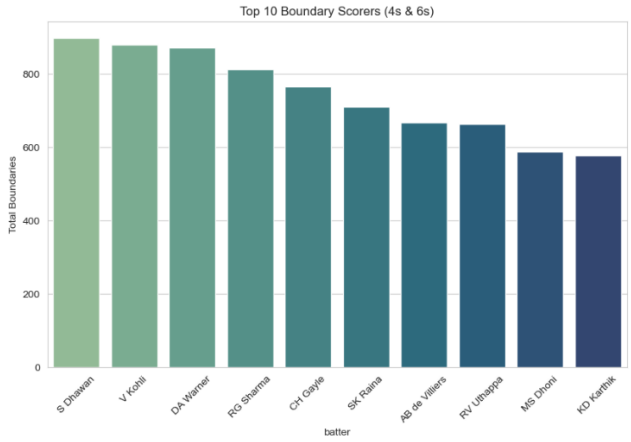
Most Sixes Hit by Players

```
plt.figure(figsize=(10,6))

sixes = merged_data[merged_data['runs_off_bat'] == 6]

top_six_hitters = sixes['batter'].value_counts().head(10)

sns.barplot(x=top_six_hitters.index, y=top_six_hitters.values, palette='Oranges_r')

plt.title("Top 10 Six Hitters")

plt.ylabel("Total Sixes")
```

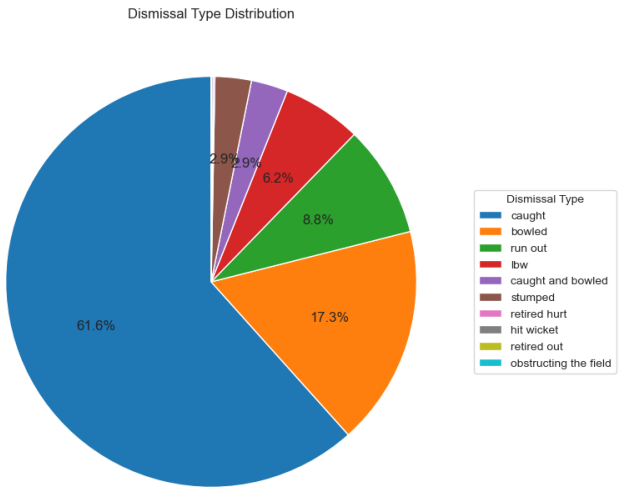
```
plt.xticks(rotation=45)
plt.savefig(r'E:/ipl-data-analysis/output/graphs/top_10_six_hitters.png', bbox_inches='tight')
plt.show()
```



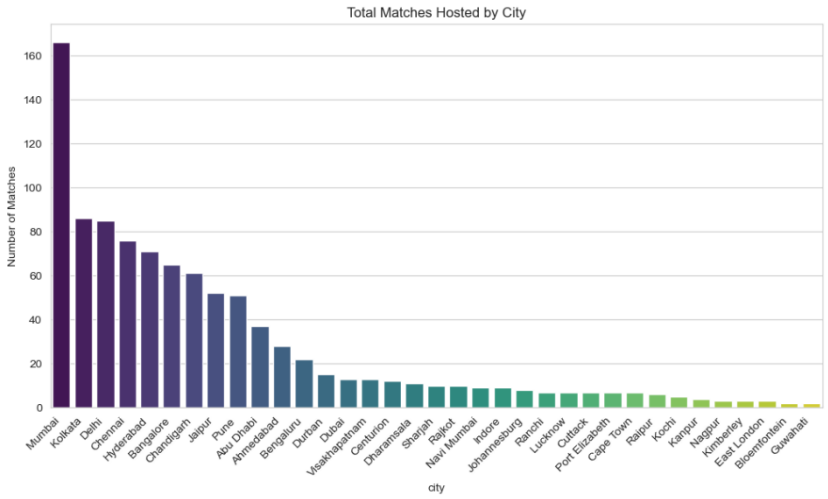

```
# Top Boundary Scorers (4s + 6s)
plt.figure(figsize=(10,6))
boundaries = merged_data[merged_data['runs_off_bat'].isin([4,6])]
boundary_counts = boundaries['batter'].value_counts().head(10)

sns.barplot(x=boundary_counts.index, y=boundary_counts.values, palette='crest')
plt.title("Top 10 Boundary Scorers (4s & 6s)")
plt.ylabel("Total Boundaries")
plt.xticks(rotation=45)
plt.savefig(r'E:/ipl-data-analysis/output/graphs/top_10_boundary_scorers.png',
bbox_inches='tight')
plt.show()
```



```
# Dismissal Type Distribution
plt.figure(figsize=(8,8))
dismissals = merged_data[merged_data['wkt_type'].notnull()]
dismissals_count = dismissals['wkt_type'].value_counts()

def autopct_format(pct):
    return ('%1.1f%%' % pct) if pct >= 1 else ''

wedges, texts, autotexts =
plt.pie(dismissals_count,autopct=autopct_format,startangle=90,textprops={'fontsize': 12})

plt.legend(wedges, dismissals_count.index, title="Dismissal Type", loc="center left",
bbox_to_anchor=(1, 0, 0.5, 1))
plt.title("Dismissal Type Distribution")
plt.savefig(r'E:/ipl-data-analysis/output/graphs/dismissal_type_distribution.png',
bbox_inches='tight')
plt.show()
```



```
# City-wise Total Matches Hosted
plt.figure(figsize=(12,6))
city_match_counts = match_info['city'].value_counts().sort_values(ascending=False)

sns.barplot(x=city_match_counts.index, y=city_match_counts.values, palette='viridis')
plt.title("Total Matches Hosted by City")
plt.ylabel("Number of Matches")
plt.xticks(rotation=45, ha='right')
plt.savefig(r'E:/ipl-data-analysis/output/graphs/total_matches_by_city.png', bbox_inches='tight')
plt.show()
```



Summary

The dataset provides granular insights into player and team performances
'Caught' is the most common mode of dismissal
Certain venues consistently support higher run-scoring matches
Winning the toss slightly increases the chances of winning the match
Some players consistently dominate in runs, sixes, and wickets across seasons

---

Conclusion

The IPL Performance Analysis project showcases how detailed cricket data can be used to extract meaningful trends, player performances, and strategic insights. Such analyses assist teams, broadcasters, and fans in understanding patterns that influence match outcomes and player success.

Future extensions could include predictive models to forecast match winners or player performance using machine learning.