WsCube Tech
System For Satisfaction

IPL Capstone Project Assignment

CAPSTONE PROJECT

Data Analytics with Advanced SQL | Power BI | MySQL | DAX | Visualization | Python

PRESENTED BY SHUBHANGI SAPKALE

About project

The goal of this project is to dive into historical IPL data to uncover meaningful insights around how players perform, how teams plan their strategies, what influences match results, and how fans engage with the league.

As a Data Analyst, my focus is on helping IPL franchises make smarter decisions—whether it’s during player auctions, setting match tactics, or finding better ways to connect with their audience.

By using advanced SQL to pull key data and building interactive dashboards in Power BI/Tableau, this project brings a clear, data-backed view of how the league performs both on and off the field.

❖ Project Overview

❑ **Objective**:

Analyze IPL historical data for actionable insights

❑ **Scope**:

Player performance, team strategies, match dynamics, and fan trends

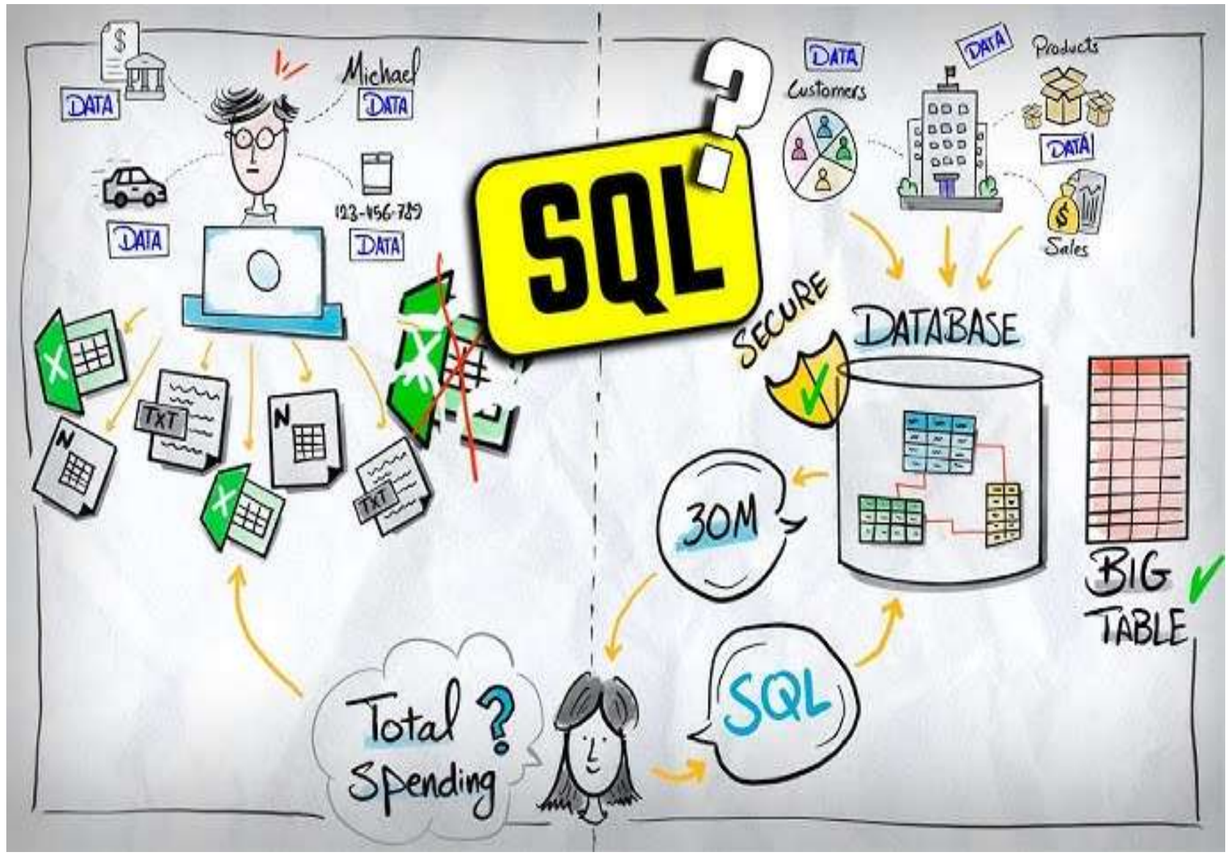
❑ **Tools Used**:

- SQL :(Data Queries)
- Power BI :(Dashboards)
- Python :(EDA & Visualization)

SR NO	SQL	POWER BI	PYTHON
1	Which batsman has scored the highest total runs?	Season Highlights	combine and clean the relevant datasets
2	Which team has conceded the least extras in the matches?	Team Performance	Analyze run trends over the years
3	What is the total number of runs scored by each team across all matches?	Player Statistics	Compare batting styles (anchor vs aggressive) via strike rate and boundary %
4	Who are the top 5 bowlers based on runs conceded per over (economy rate)?	Venue Analysis	Study bowling consistency (dot balls, economy rate, average)
5	Which batsman has faced the most balls?	Match Situations	Visualize performance in different overs (Powerplay, Death)
6	Which bowling team has taken part in the most overs?	Season-wise Trends	Compare venue behavior in high-scoring vs low-scoring matches
7	What is the average number of runs scored per ball by each batsman?		
8	Which teams have the best win percentage according to the team performance table?		
9	List all players who are all-rounders (both batting and bowling style available).		
10	What is the average number of runs scored per over by each batting team?		

Phase 1:
SQL Analysis - Advanced Queries

MySQL



1.Which batsman has scored the highest total runs?

```sql
select striker as batsman,sum(runs_scored) as total_runs
from ipl_ballbyball2008_2024_updated
group by striker
order by total_runs desc
limit 1;
```

Result Grid | Filter Rows:

batsman	total_runs
V Kohli	7773

2. Which team has conceded the least extras in the matches?

```sql
select bowling_team as team,sum(extras) as total_extras_conceded
from ipl_ballbyball2008_2024_updated
group by Bowling_team
order by total_extras_conceded asc
limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

team	total_extras_conceded
Rising Pune Supergiants	108

3.What is the total number of runs scored by each team across all matches?

```sql
select Batting_team as team, sum(runs_scored + extras) as total_runs_scored
from ipl_ballbyball2008_2024_updated
group by Batting_team
order by total_runs_scored desc;
```

Result Grid | Filter Rows:

team	total_runs_scored
Mumbai Indians	41522
Kolkata Knight Riders	38492
Chennai Super Kings	37930
Royal Challengers Bangalore	37692
Rajasthan Royals	33748
Kings XI Punjab	30064
Sunrisers Hyderabad	27868
Delhi Daredevils	24296
Delhi Capitals	14331
Deccan Chargers	11463
Punjab Kings	8857
Gujarat Titans	7379
Lucknow Super Giants	6805
Pune Warriors	6358
Gujarat Lions	4862
Rising Pune Supergiant	2470
Rising Pune Supergiants	2063
Royal Challengers Bengaluru	1960
Kochi Tuskers Kerala	1901

4.Who are the top 5 bowlers based on runs conceded per over (economy rate)?

```sql
SELECT
    Bowler,
    ROUND(SUM(runs_scored + extras) / (SUM(CASE
                                        WHEN type_of_extras NOT IN ('wide', 'no-ball') OR type_of_extras IS NULL
                                        THEN 1
                                        ELSE 0
                                    END) / 6.0), 2) AS economy_rate
FROM ipl_ballbyball2008_2024_updated
GROUP BY Bowler
HAVING SUM(CASE
              WHEN type_of_extras NOT IN ('wide', 'no-ball') OR type_of_extras IS NULL
              THEN 1
              ELSE 0
          END) >= 6
ORDER BY economy_rate ASC
LIMIT 5;
```

Result Grid | Filter Rows:

Bowler	economy_rate
R Ravindra	3.50
NB Singh	4.32
Sachin Baby	4.80
AM Rahane	5.00
LA Carseldine	5.14

5.Which batsman has faced the most balls?

```sql
SELECT
    Bowler,
    ROUND(SUM(runs_scored + extras) / (SUM(CASE
                                           WHEN type_of_extras NOT IN ('wide', 'no-ball') OR type_of_extras IS NULL
                                           THEN 1
                                           ELSE 0
                                       END) / 6.0), 2) AS economy_rate
FROM ipl_ballbyball2008_2024_updated
GROUP BY Bowler
HAVING SUM(CASE
               WHEN type_of_extras NOT IN ('wide', 'no-ball') OR type_of_extras IS NULL
               THEN 1
               ELSE 0
           END) >= 6
ORDER BY economy_rate ASC
LIMIT 5;
```

Result Grid | Filter Rows:

batsman	balls_faced
V Kohli	5929

6.Which bowling team has taken part in the most overs?

```sql
SELECT
    Bowling_team,
    COUNT(*) / 6 AS total_overs
FROM
    ipl_ballbyball2008_2024_updated
GROUP BY
    Bowling_team
ORDER BY
    total_overs DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export:

Bowling_team	total_overs
Mumbai Indians	5170.0000

7.What is the average number of runs scored per ball by each batsman?

```sql
SELECT Striker AS Batsman,SUM(runs_scored) AS Total_Runs,COUNT(*) AS Balls_Faced,
ROUND(SUM(runs_scored) * 1.0 / COUNT(*), 3) AS Runs_Per_Ball
FROM IPL_BallByBall2008_2024_Updated
WHERE type_of_extras IS NULL
GROUP BY Striker
HAVING Balls_Faced > 0
ORDER BY Runs_Per_Ball DESC;
```

Result Grid | Filter Rows: | Export:

Batsman	Total_Runs	Balls_Faced	Runs_Per_Ball
L Wood	9	3	3.000
R Sai Kishore	13	5	2.600
KMDN Kulasekara	5	2	2.500
B Stanlake	5	2	2.500
J Fraser-McGurk	253	106	2.387
Umar Gul	39	17	2.294
VRV Singh	4	2	2.000
RS Sodhi	4	2	2.000
Naman Dhir	50	25	2.000
Ashutosh Sharma	153	50	1.962
R Shepherd	114	59	1.932
WG Jacks	170	88	1.932
ER Dwivedi	24	13	1.846
PD Salt	609	338	1.802
Shahid Afridi	81	45	1.800
TM Head	542	304	1.783
AD Russell	2422	1368	1.770
I Malhotra	7	4	1.750
TU Deshpande	21	12	1.750
TH David	628	359	1.749
H Klaasen	804	460	1.748
KK Cooper	115	66	1.742
BCJ Cutting	238	137	1.737
LJ Wright	102	59	1.729
SP Narine	1416	823	1.721

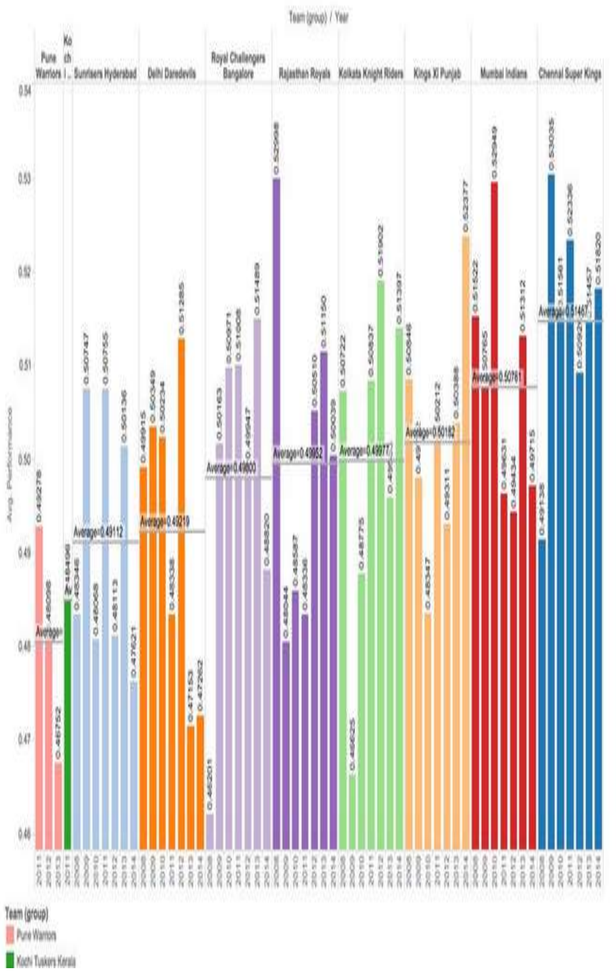
Result 8

8.Which teams have the best win percentage according to the team performance table?

```sql
WITH match_counts AS (
    SELECT
        Team,
        COUNT(*) AS Matches_Played
    FROM (
        SELECT
            SUBSTRING_INDEX(Teams, ' vs ', 1) AS Team
        FROM team_performance_dataset_2008to2024
        UNION ALL
        SELECT
            SUBSTRING_INDEX(Teams, ' vs ', -1) AS Team
        FROM team_performance_dataset_2008to2024
    ) AS all_teams
    GROUP BY Team
),
```

```sql
win_counts AS (
    SELECT
        Match_Winner AS Team,
        COUNT(*) AS Matches_Won
    FROM team_performance_dataset_2008to2024
    WHERE Match_Winner IS NOT NULL AND Match_Winner != ''
    GROUP BY Match_Winner
)
SELECT
    m.Team,
    m.Matches_Played,
    w.Matches_Won,
    ROUND((w.Matches_Won / m.Matches_Played) * 100, 2) AS Win_Percentage
FROM
    match_counts m
LEFT JOIN
    win_counts w ON m.Team = w.Team
ORDER BY
    Win_Percentage DESC;
```

Team	Matches_Played	Matches_Won	Win_Percentage
Gujarat Titans	43	27	62.79
Rising Pune Supergiant	16	10	62.50
Chennai Super Kings	234	136	58.12
Lucknow Super Giants	40	23	57.50
Mumbai Indians	257	141	54.86
Kolkata Knight Riders	246	125	50.81
Rajasthan Royals	215	109	50.70
Delhi Capitals	88	43	48.86
Royal Challengers Bangalore	240	114	47.50
Sunrisers Hyderabad	175	83	47.43
Kings XI Punjab	190	85	44.74
Punjab Kings	52	23	44.23
Gujarat Lions	30	13	43.33
Kochi Tuskers Kerala	14	6	42.86
Delhi Daredevils	161	67	41.61
Deccan Chargers	75	29	38.67
Rising Pune Supergiants	14	5	35.71
Royal Challengers Bengaluru	10	3	30.00
Pune Warriors	46	12	26.09



9.List all players who are all-rounders (both batting and bowling style available).

```sql
SELECT
    Player_Name,Team_Name,Player_Role,Batting_Style,Bowling_Style
FROM  players_info_2024
WHERE
    Batting_Style IS NOT NULL
        AND Batting_Style != ''
        AND Bowling_Style IS NOT NULL
        AND Bowling_Style != '';
```

Player_Name	Team_Name	Player_Role	Batting_Style	Bowling_Style
MS Dhoni	CSK	Wicketkeeper Batter	Right hand Bat	Right arm Medium
Devon Conway	CSK	Wicketkeeper Batter	Left hand Bat	Right arm Medium
Ruturaj Gaikwad	CSK	Batter	Right hand Bat	Right arm Offbreak
Ajinkya Rahane	CSK	Top order Batter	Right hand Bat	Right arm Medium
Shaik Rasheed	CSK	Batter	Right hand Bat	Legbreak
Sameer Rizvi	CSK	Batter	Right hand Bat	Right arm Offbreak
Avanish Rao Aravelly	CSK	Wicketkeeper Batter	Left hand Bat	No Bowling Style
Ravindra Jadeja	CSK	Allrounder	Left hand Bat	Slow Left arm Orthodox
Mitchell Santner	CSK	Bowling Allrounder	Left hand Bat	Slow Left arm Orthodox
Moeen Ali	CSK	Batting Allrounder	Left hand Bat	Right arm Offbreak
Shivam Dube	CSK	Allrounder	Left hand Bat	Right arm Medium
Nishant Sindhu	CSK	Allrounder	Left hand Bat	Slow Left arm Orthodox
Ajay Mandal	CSK	Allrounder	Left hand Bat	Slow Left arm Orthodox
Rachin Ravindra	CSK	Batting Allrounder	Left hand Bat	Slow Left arm Orthodox
Shardul Thakur	CSK	Bowler	Right hand Bat	Right arm Medium
Daryl Mitchell	CSK	Batting Allrounder	Right hand Bat	Right arm Medium
Rajvardhan Hangar...	CSK	Bowling Allrounder	Right hand Bat	Right arm Fast medium
Deepak Chahar	CSK	Bowler	Right hand Bat	Right arm Medium
Maheesh Theekshana	CSK	Bowler	Right hand Bat	Right arm Offbreak
Mukesh Choudhary	CSK	Bowler	Left hand Bat	Left arm Medium
Mustafizur Rahman	CSK	Bowler	Left hand Bat	Left arm Fast medium

10.What is the average number of runs scored per over by each batting team?

```sql
SELECT
  Batting_Team,
  ROUND(SUM(runs_Scored + extras) / (SUM(CASE
                                      WHEN type_of_extras NOT IN ('wide', 'no-ball') OR type_of_extras IS NULL
                                      THEN 1
                                      ELSE 0
                                    END) / 6.0), 2) AS avg_runs_per_over
FROM ipl_ballbyball2008_2024_updated
GROUP BY Batting_Team
ORDER BY avg_runs_per_over DESC;
```

Batting_Team	avg_runs_per_over
Royal Challengers Bengaluru	9.55
Gujarat Titans	8.44
Punjab Kings	8.35
Lucknow Super Giants	8.30
Gujarat Lions	8.18
Delhi Capitals	8.13
Chennai Super Kings	8.08
Mumbai Indians	8.04
Royal Challengers Bangalore	8.02
Kings XI Punjab	7.97
Kolkata Knight Riders	7.96
Sunrisers Hyderabad	7.95
Rajasthan Royals	7.94
Rising Pune Supergiants	7.83
Rising Pune Supergiant	7.80
Delhi Daredevils	7.76
Deccan Chargers	7.61
Kochi Tuskers Kerala	7.21
Pune Warriors	7.01

IPL SQL Query Insights - Summary

This analysis extract key players and team performance matrics from IPL data using SQL.

- ❖ Top Performers –Identify the highest run-scorer and most ball-faced batsman.
- ❖ Team Analysis-Evaluate team by total runs, extras conceded, win percentage and average run rate.
- ❖ Bowling Insights- Highlight top 5 economical bowler and team with the most overs bowled
- ❖ All –Rounders- players with both batting and bowling styles.

Scoring Efficiency- Analyze batsman run per ball and team runs per over.

- ➢ These Queries provide a well-rounded view of individual brilliance and team strategies in the IPL.

Phase 2: Data Visualization - Power BI

Power BI

Season Highlights

Season
All

Team
All

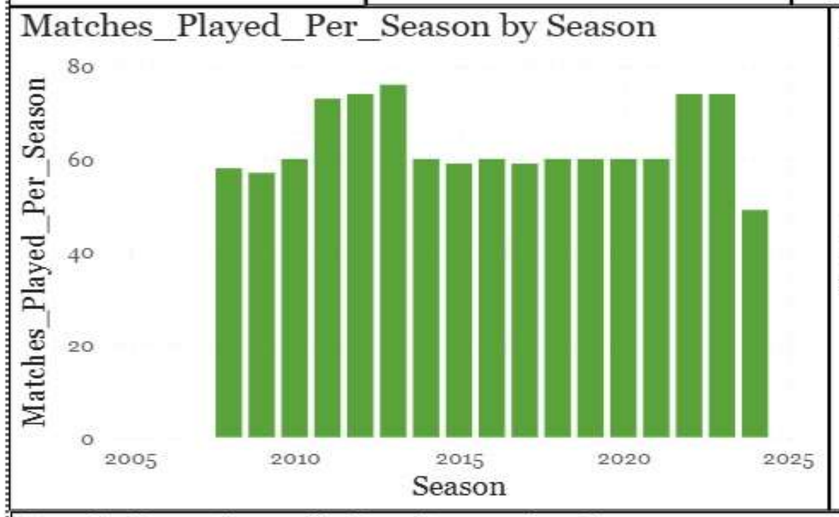
1073
Matches_Played_Per_Seas...

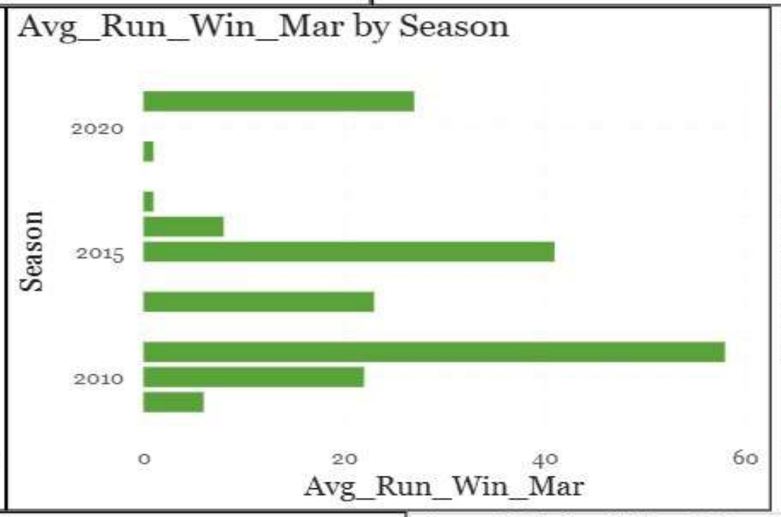
20.78
Avg_Run_Win_Mar

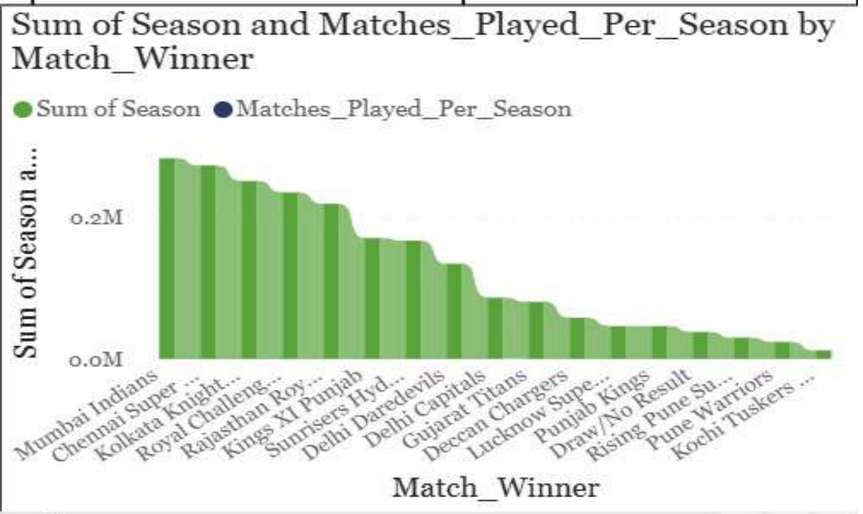
340K
Total_Run_Scored_By_Sea...

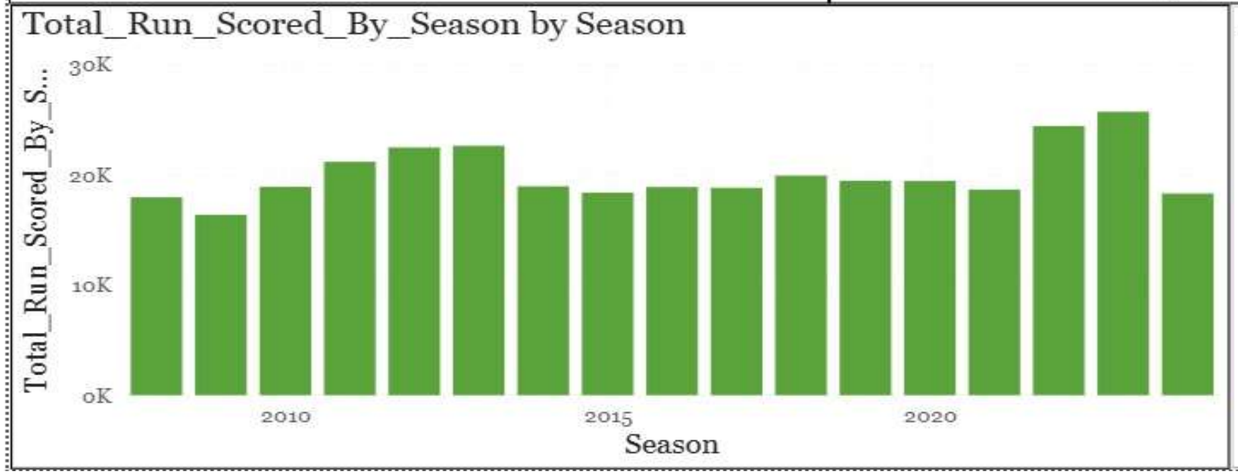
29K
Four_Per_Season

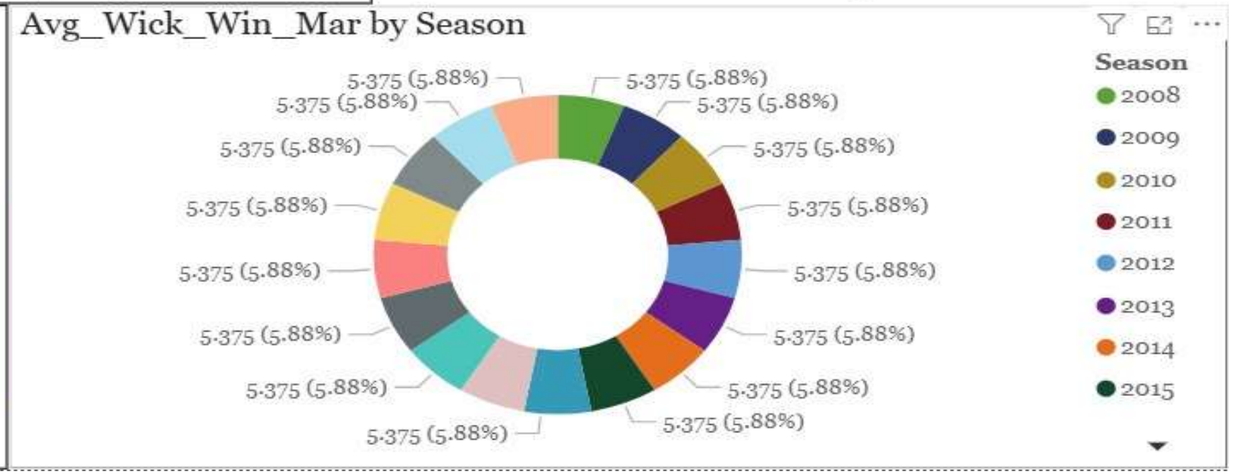
13K
SIXES_PER_SEASON











❖ Season Highlights Dashboard Overview

This dashboard provides a comprehensive summary of cricket match statistics over 8 seasons. Key highlights include:

- **Total Matches Played:** 1,073
- **Total Runs Scored:** 340,000
- **Total Boundaries:** 29,000 fours and 13,000 sixes
- **Average Win Margin:** 20.78 (runs or wickets)

❑ Seasonal Insights:

- A steady increase in matches played, runs scored, and boundaries hit across seasons.
- Season 8 marks the peak in all three categories, indicating growing competitiveness and performance

.

❑ Team Performance:

- The "Top Winning Teams" bar chart highlights the most successful teams by number of wins.
- Winning margins are broken down by season, showing fluctuations in match dominance.

❑ Win Margin Analysis:

- Bar and donut charts show win margins by runs and wickets, with most wins having a margin between 11% to 16%.

This visual representation helps track team progress, seasonal trends, and performance metrics, aiding data-driven decision-making for future strategies.

Team Performance

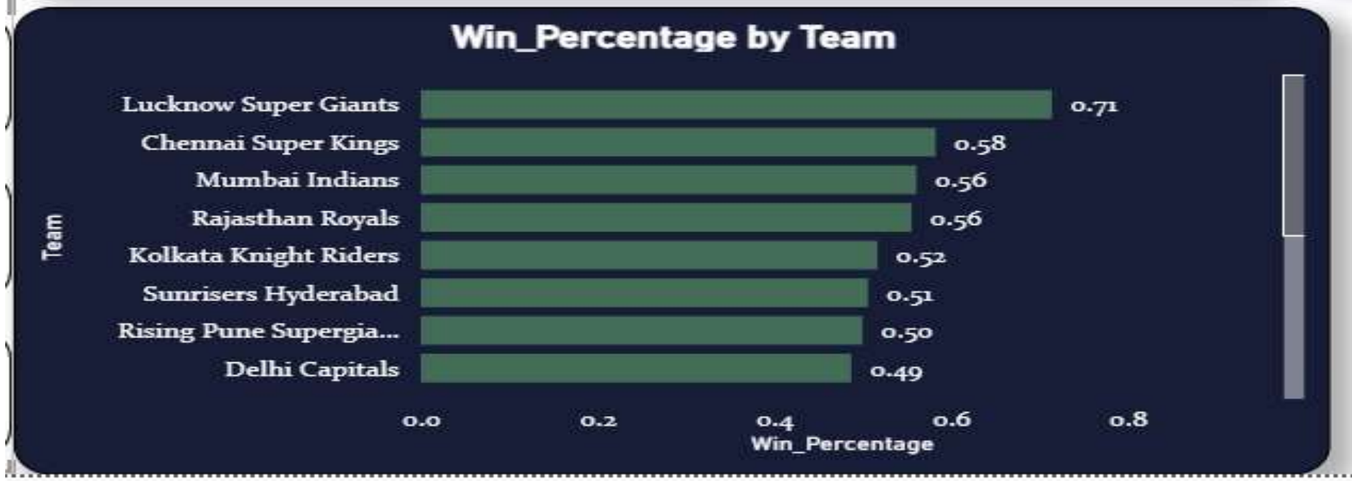
Chennai Supper King

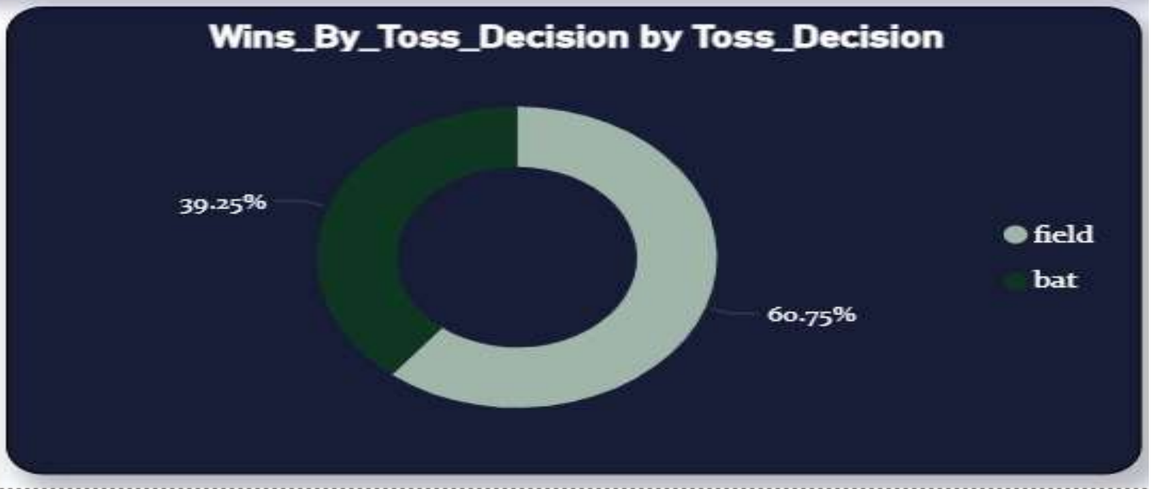
First Final Looser

Rajasthan Royals

First Final Winner

Winner	Opponent	H2H TotalMatches	H2H Win
Chennai Super Kings	Chennai Super Kings	107	4
Chennai Super Kings	Deccan Chargers	36	2
Chennai Super Kings	Delhi Capitals	49	2
Chennai Super Kings	Delhi Daredevils	76	4
Chennai Super Kings	Gujarat Titans	38	1
Chennai Super Kings	Kings XI Punjab	98	5
Chennai Super Kings	Kochi Tuskers Kerala	7	
Chennai Super Kings	Kolkata Knight Riders	128	6
Chennai Super Kings	Lucknow Super Giants	19	
Chennai Super Kings	Mumbai Indians	134	6
Chennai Super Kings	Pune Warriors	23	1
Chennai Super Kings	Rajasthan Royals	116	5





❖ Team performance Dashboard overview

- Chennai Super Kings lead significantly, followed by: Mumbai Indians Kolkata Knight Riders ,Royal Challengers Bangalore, and others.2.

- CSK has played the most matches against Mumbai Indians (134) and Kolkata Knight Riders (128).Best win ratio against Kochi Tuskers Kerala (6 out of 7 matches).

- Top performers : Lucknow Super Giants – Highest win percentage (0.71)Chennai Super Kings – Second (0.58)Others include Mumbai Indians, Rajasthan Royals, and KKR Useful to compare performance efficiency, not just total wins.4.

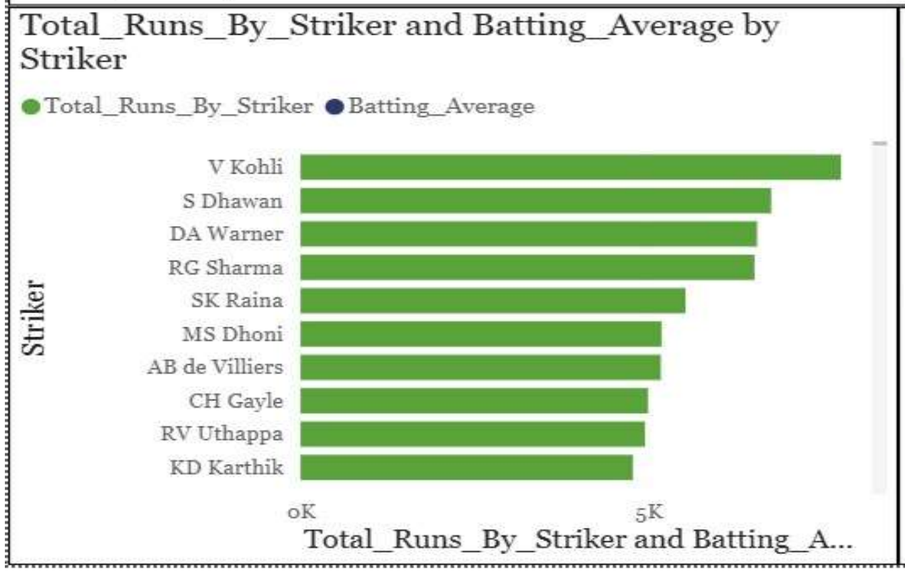
- Wins By Toss Decision showing CSK's win percentage based on toss decisions ,When they choose to field, they win 60.75% of the time.
- When they choose to bat, they win 39.25% of the time .Indicates CSK performs better when chasing a target.
- Overall Insights: Chennai Super Kings are one of the most successful teams by total wins.
- Despite losing the first final, they have remained consistently strong win percentage is also among the top teams ,Better win rate when fielding first, a key strategic insight.Head-to-head stats show a strong record against almost every opponent.

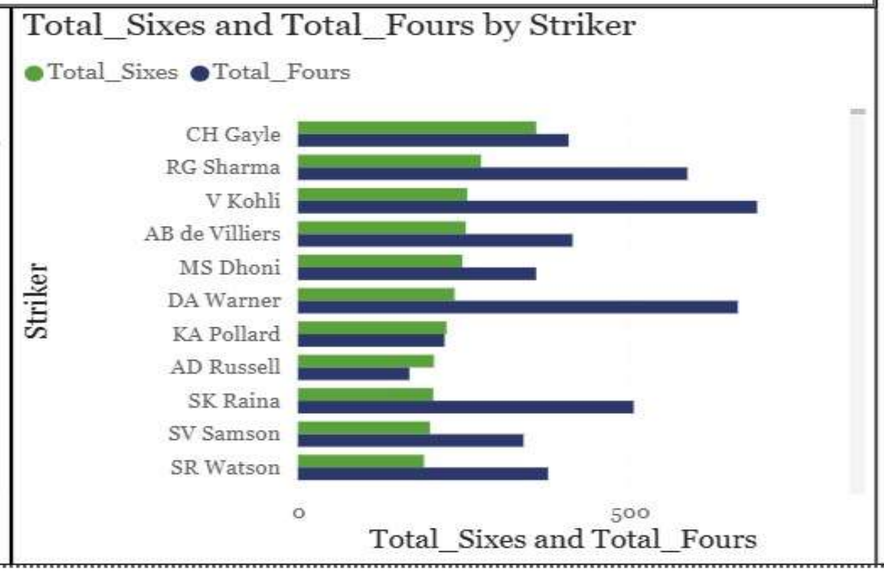
Player Statistics

Team
All

Season
All

AB de Villiers
Top_Player_of_Match





Bowler	Total_Wickets	Economy_Rate
A Ashish Reddy	19	9.16
A Badoni	2	5.68
A Chandila	11	6.28
A Choudhary	5	8.55
A Dananjaya		11.75
A Flintoff	2	9.64
A Kumble	49	6.77
A Mishra	183	7.46
A Mithun	8	9.94
A Nehra	121	7.98
A Nel	1	10.33
A Nortje	62	9.19
A Singh	31	8.11
A Symonds	23	7.90
A Uniyal	3	11.17
A Zampa	30	8.05
AA Chavan	8	8.20
AA Jhunjhunwala	1	9.18
AA Kazi	1	10.50
AA Noffke	1	10.25
AB Agarkar	33	9.01
AB Dinda	82	8.32
AB McDonald	14	8.48
Abdul Samad	2	12.56
Abdur Razzak		14.50
Abhishek Sharma	9	9.07
AC Gilchrist	1	0.00
AC Thomas	16	7.92
AC Voges	1	8.44
AD Mascarenhas	19	7.11
AD Mathews	28	8.31
AD Russell	115	9.53
AF Milne	9	9.83
AG Murtaza	12	7.32
AJ Finch	1	9.35
AJ Hosein	1	10.25
AJ Tye	48	8.80
AK Markram	2	8.70
Akash Deep	8	11.85
Total	**12651**	**8.28**

❖ Player statistics Dashboard overview

This dashboard provides a detailed analysis of player performance. highlighting his batting performance and key statistics:

•**Top Performer:** AB de Villiers is recognized as the top player of the match.

•**Total Rounds Score by Striker:** Displays consistent scoring by various batters, with AB among the top performers.

•**Total Runs by Striker and Batting Position:** Highlights AB's run contribution across positions, with other top scorers included for comparison.

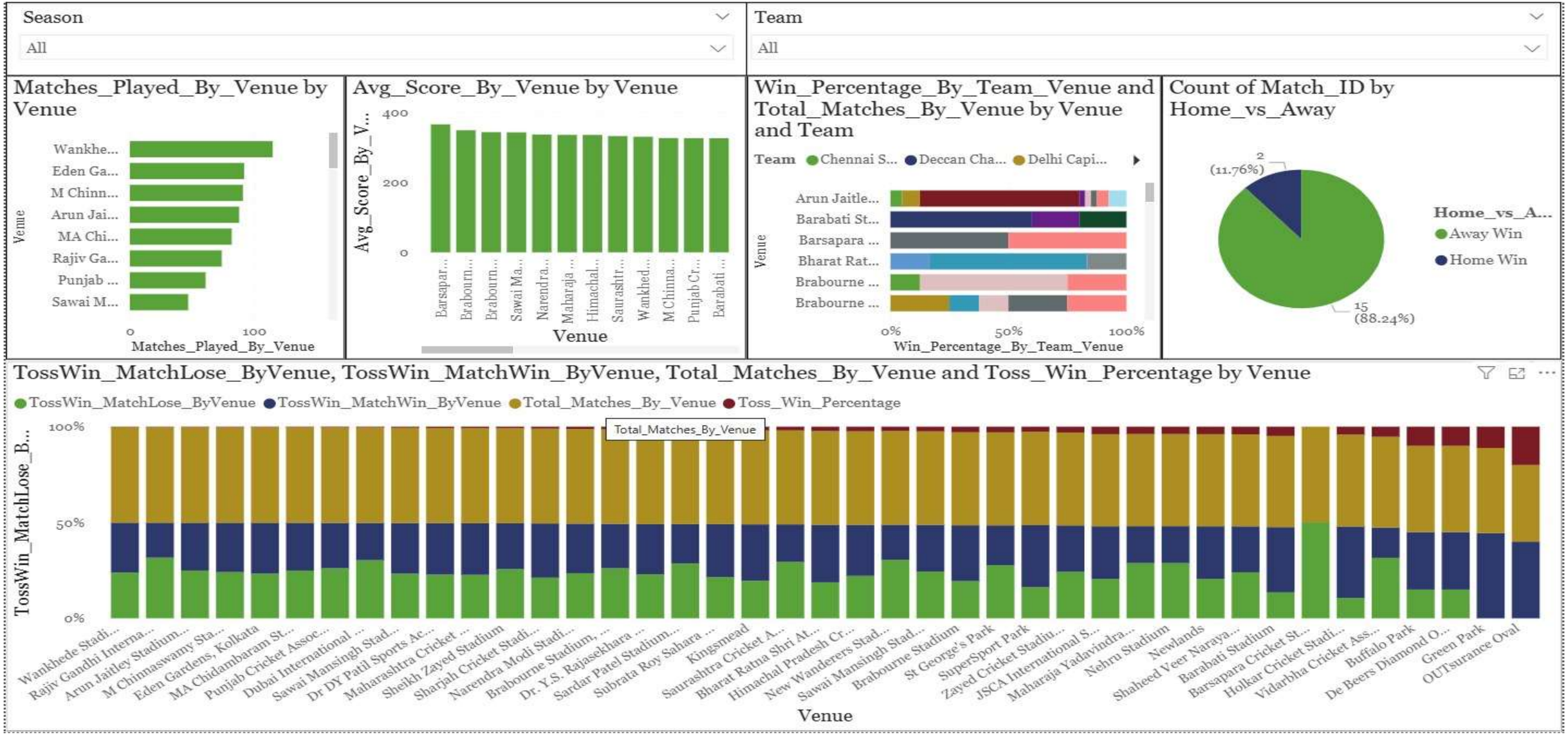
•**Sixes and Fours Analysis:** Tracks the total number of boundaries (sixes and fours) scored by each player.

•**Bowling Statistics:** Lists top bowlers by total wickets and economy rate, with As high Reddy leading in wickets taken.

•**Donut Chart Summary:** Visualizes total sixes and fours by top strikers, with individual contributions highlighted for better comparison.

This dashboard provides a well-rounded view of individual player impact, both in batting and bowling, while showcasing AB de Villiers' consistent dominance as a key match-winner.

Venue Analysis



❖ Venue Analysis Dashboard Overview

This dashboard provides a comprehensive analysis of IPL match data with a focus on venue-based insights. It enables users to explore team performances, match outcomes, and scoring trends across different stadiums. Key components of the dashboard include:

- **Matches Played by Venue**
  Highlights the number of IPL matches hosted at each venue, with Wankhede Stadium and Eden Gardens leading in frequency.

- **Average Score by Venue**
  Displays the average total runs scored at each venue, helping identify high- and low-scoring grounds.

- **Win Percentage by Team and Venue**
  Illustrates team-wise win percentages at various venues, showcasing performance dominance or struggles at specific locations.

- **Home vs. Away Wins**
  A pie chart showing the distribution of match wins between home and away teams. Notably, away wins are significantly higher in the sample.

- **Toss Win vs Match Outcome Analysis by Venue**
  A stacked bar chart representing:
  - Toss wins that led to match wins or losses.
  - Total matches played per venue.
  - Toss win percentages.

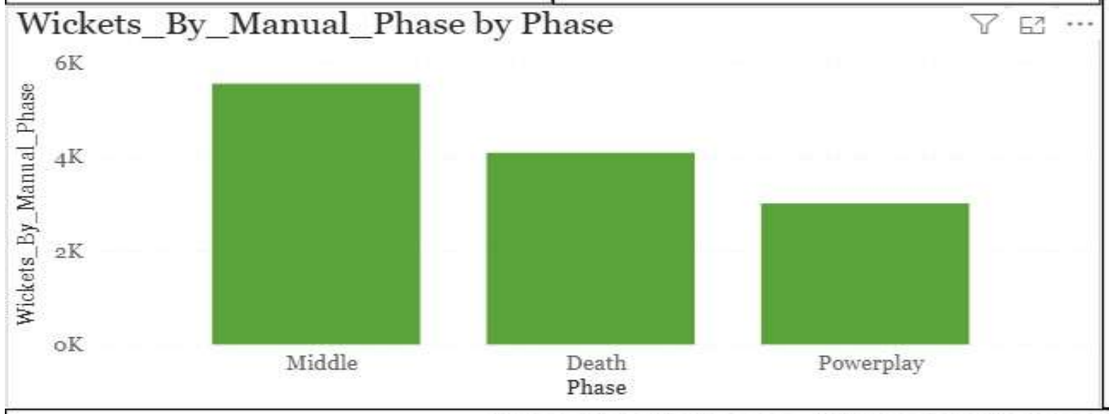
  This visual helps evaluate the impact of winning the toss on match results.

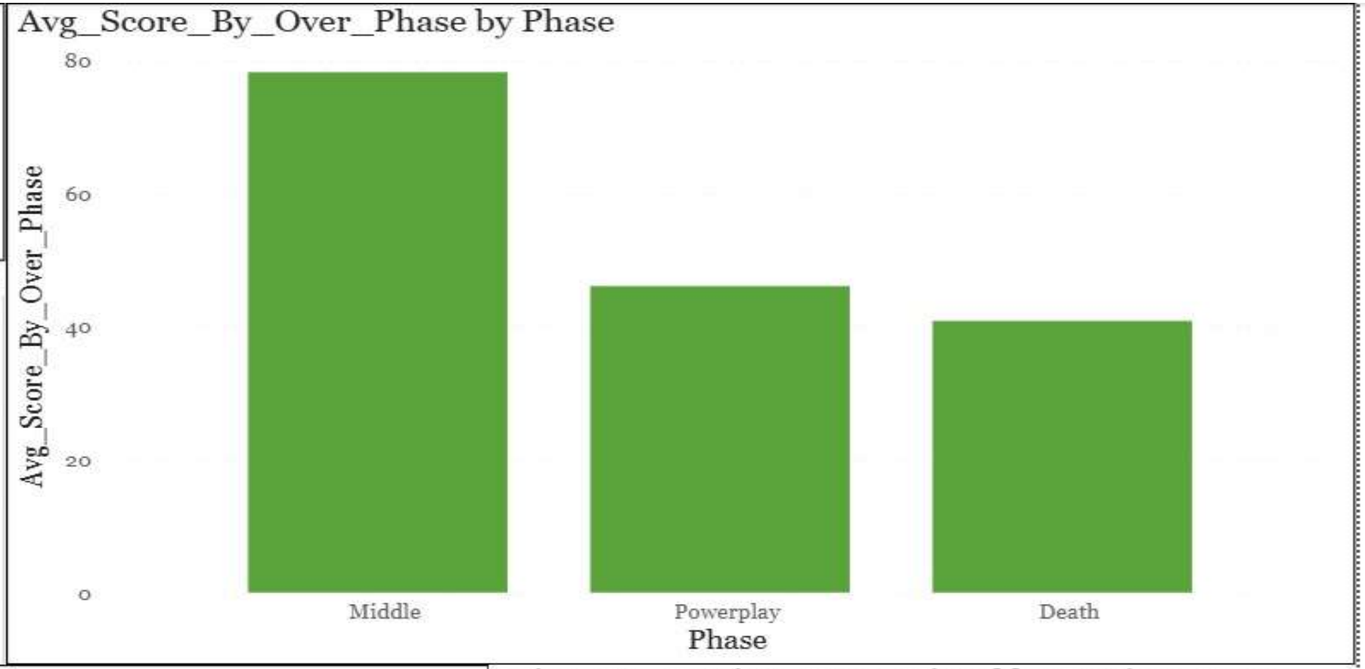
Match Situations

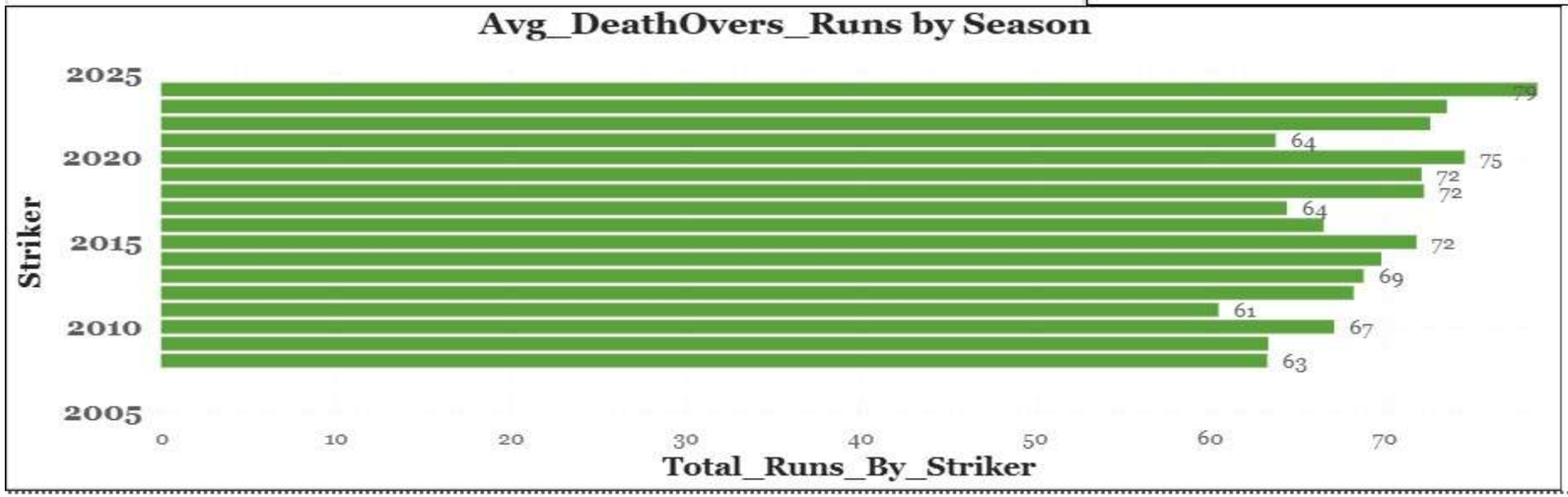
Season
All

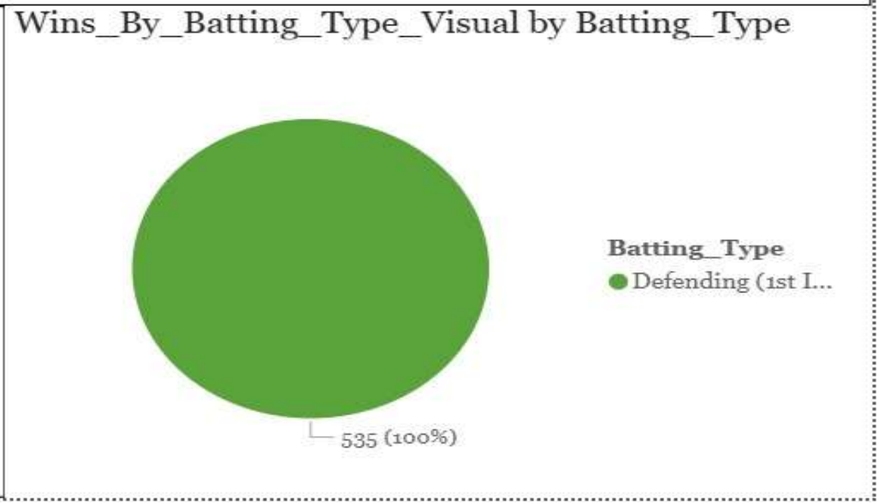
Team
All

14
Super_Over_Matches









❖ Match situation Dashboard overview

- This dashboard provides a comprehensive overview of match phases and performance metrics in IPL matches, focusing on wickets, scoring trends, and match outcomes based on batting type.

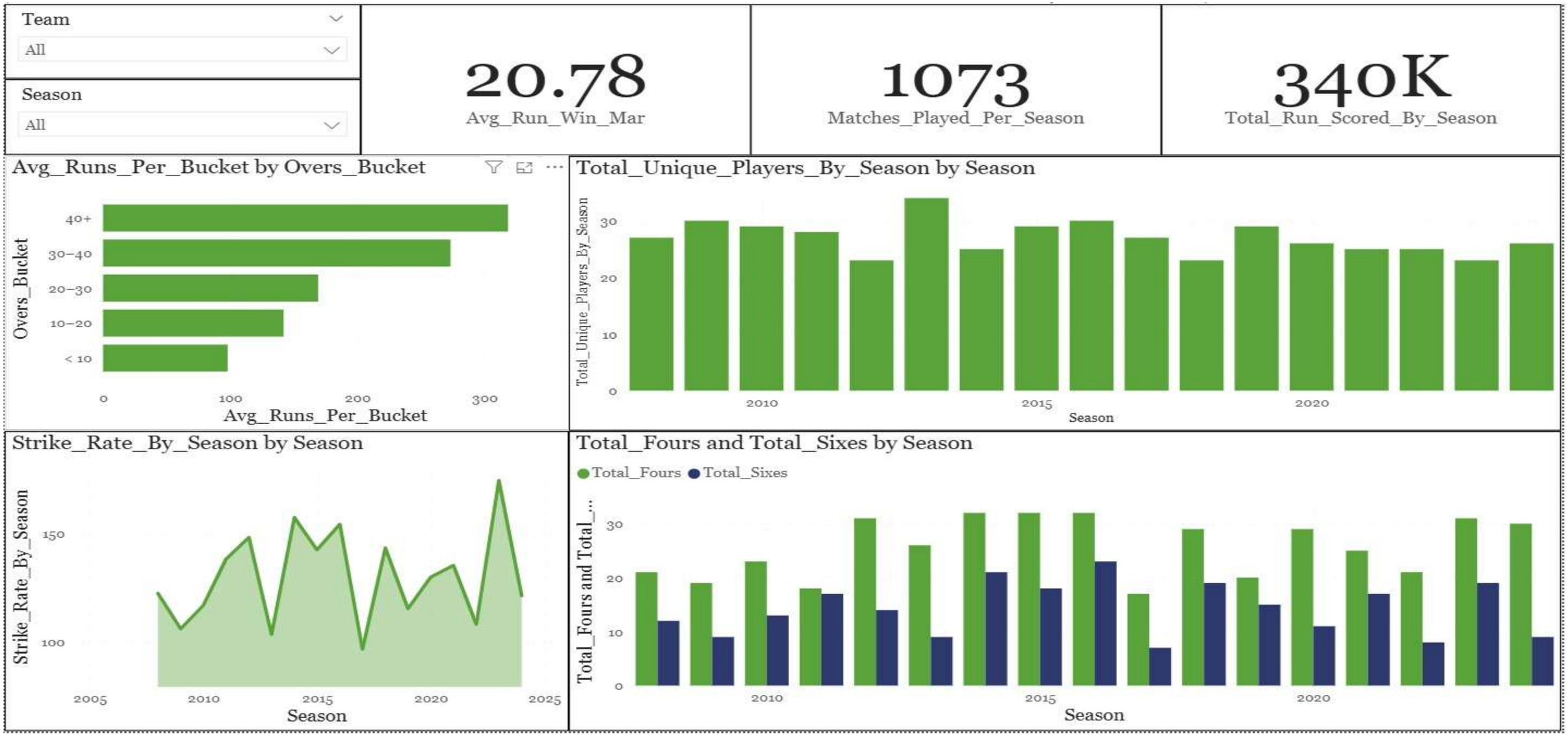
- Super Over Matches:14 Super Over Matches have occurred in the dataset across all seasons.

- Wickets By Manual Phase: total wickets taken during three key match phases ,Middle Phase (7–15 overs): ~5,800 wickets most wickets fall during this phase .Death Phase (16–20 overs): ~4,000 wickets .Powerplay (1–6 overs): ~3,000 wickets lowest wickets.
- Insight: Bowlers tend to pick up more wickets during the middle overs, likely due to spinners and changes in pace disrupting rhythm.

- Avg Score By Over Phase: Middle Phase again leads (~78 runs) – surprisingly the most productive phase Powerplay – ~50 runs .Death Overs – ~45 runs.
- Insight: Contrary to popular belief, middle overs yield the highest runs, indicating teams accelerate early and maintain momentum through rotation and boundaries.

- Avg Death Overs Runs :
- 2025 Highest (~79 runs).2020: Lower at ~64 runs.2010–2019: Fluctuates between 63 and 72.
- Insight: Scoring in death overs has increased over time, reflecting better finishing skills or rule changes favoring batters.
- Wins By Batting Type Visual All wins (100%) shown here are from teams defending (batting 1st).Value: 535 wins when batting first.

Season-wise Trends



❖ Season-wise trends Dashboard overview

This Dashboard is designed to support data-driven analysis of player and team performance using filters for **Team** and **Season**.

➢ **Key Metrics Displayed:**

- **Avg Run Win Mar**: 20.78
  Average run margin for wins across seasons.

- **Matches Played Per Season**: 1073
  Total number of IPL matches played.

- **Total Run Scored By Season**: 340K
  Cumulative runs scored across all season

Phase 3:
Exploratory Data Analysis (EDA)

❖ Exploratory Data Analysis for IPL

GITHUB LINK :https://github.com/shubhangi96804/IPL-capstone-project

(click above link to see EDA report for IPL)

❖ Content

1.Combine and clean relevant datasets

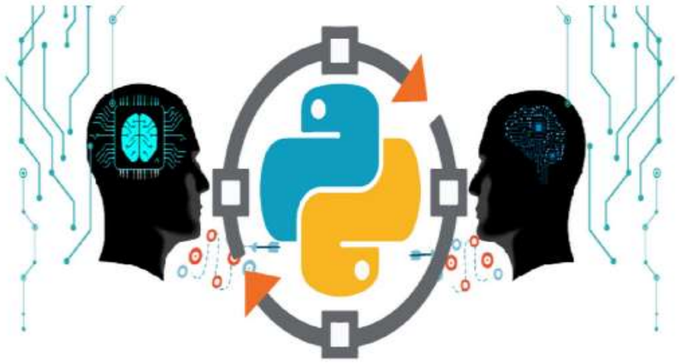
2.Analyze ren trends over the year

3.Compare batting styles

4.Study bowling consistency

5.Visualize performance in different overs

6.Compare venue behaviour in high scoring vs low scoring matches



1.Combine and clean the relevant dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
import pandas as pd

# Load ball-by-ball dataset
ball_by_ball = pd.read_csv(
    r"D:\capstone ipl\IPL_BallByBall2008_2024_Updated.csv",
    encoding='unicode_escape',
    on_bad_lines='skip',
    low_memory=False
)

# Load players info
players_info = pd.read_csv(
    r"D:\capstone ipl\Players_Info_2024.csv",
    encoding='unicode_escape',
    on_bad_lines='skip'
)

# Load team performance data
team_perf = pd.read_csv(
    r"D:\capstone ipl\team_performance_dataset_2008to2024.csv",
    encoding='unicode_escape',
    on_bad_lines='skip'
)
```

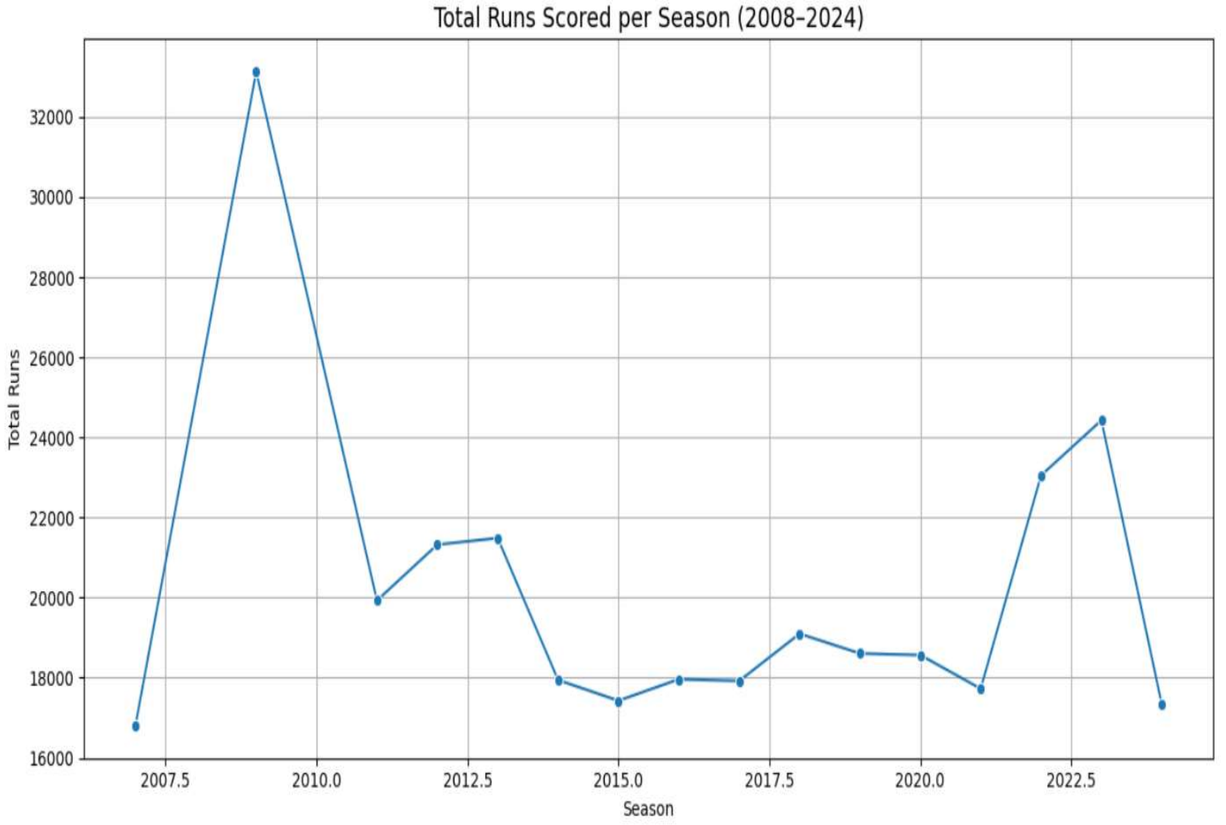
```
# Clean column names
def clean_columns(df):
    df.columns = df.columns.str.strip().str.lower().str.replace(' ', '').str.replace('-', '')
    return df

ball_by_ball = clean_columns(ball_by_ball)
players_info = clean_columns(players_info)
team_perf = clean_columns(team_perf)



# Show column names
print("ball_by_ball columns:", ball_by_ball.columns.tolist())
print("players_info columns:", players_info.columns.tolist())
print("team_perf columns:", team_perf.columns.tolist())



# Null summary
print("Ball by Ball nulls:\n", ball_by_ball.isnull().sum())
print("Players Info nulls:\n", players_info.isnull().sum())
print("Team Perf nulls:\n", team_perf.isnull().sum())
```

```python
# Merge player info with ball-by-ball using 'striker' → 'player_name'
ball_by_ball = ball_by_ball.merge(players_info, left_on='striker', right_on='playername', how='left')

# Drop 'player' column if it exists
if 'player' in ball_by_ball.columns:
    ball_by_ball = ball_by_ball.drop(columns=['player'])

# Summary
print("Merged shape:", ball_by_ball.shape)
print("Nulls after merge:\n", ball_by_ball.isnull().sum())

# If 'season' not present, extract from match_date
if 'season' not in ball_by_ball.columns and 'match_date' in ball_by_ball.columns:
    ball_by_ball['match_date'] = pd.to_datetime(ball_by_ball['match_date'], errors='coerce')
    ball_by_ball['season'] = ball_by_ball['match_date'].dt.year

# Ensure 'season' is cleaned (even if already present)
def extract_year(value):
    try:
        if isinstance(value, str) and '/' in value:
            return int(value.split('/')[0])
        return int(value)
    except:
        return None

ball_by_ball['season'] = ball_by_ball['season'].apply(extract_year)
```

```
Merged shape: (255759, 29)
Nulls after merge:
 matchid                  0
date                      0
season                    0
battingteam               0
bowlingteam               0
inningsno                 0
ballno                    0
bowler                    0
striker                   0
nonstriker                0
runs_scored               0
extras                    0
typeofextras         241936
score                     0
score/wicket              0
wicket_confirmation       0
wicket_type          243108
fielders_involved    246637
playerout            243108
playername           237445
teamname             237445
playernationality    237445
dateofbirth          237445
playerrole           237445
ipldebut             237698
about                237445
battingstyle         237445
bowlingstyle         237445
playersalary         237445
dtype: int64
```

2.Analyse Runs Trends over the year

```
# Group and plot total runs per season
yearly_runs = ball_by_ball.groupby('season')['runs_scored'].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(data=yearly_runs, x='season', y='runs_scored', marker='o')
plt.title('Total Runs Scored per Season (2008-2024)', fontsize=14)
plt.xlabel('Season')
plt.ylabel('Total Runs')
plt.grid(True)
plt.tight_layout()
plt.show()
```



3.Bating style comparison

```
#PART 3: Batting Style Comparison
# Filter only legal deliveries
legal_deliveries['is_boundary'] = legal_deliveries['runs_scored'].isin([4, 6])
```

```
print(legal_deliveries.columns.tolist())
```

```
['matchid', 'date', 'season', 'battingteam', 'bowlingteam', 'inningsno', 'ballno', 'bowler', 'striker', 'nonstriker', 'runs_scored', 'extras', 'typeofextras', 'score', 'score/wicket', 'wicket_confirmation', 'wicket_type', 'fielders_involved', 'playerout', 'playername', 'teamname', 'playernationality', 'dateofbirth', 'playerrole', 'ipldebut', 'about', 'battingstyle', 'bowlingstyle', 'playersalary', 'over', 'over_phase', 'is_dot', 'is_wicket', 'is_boundary']
```

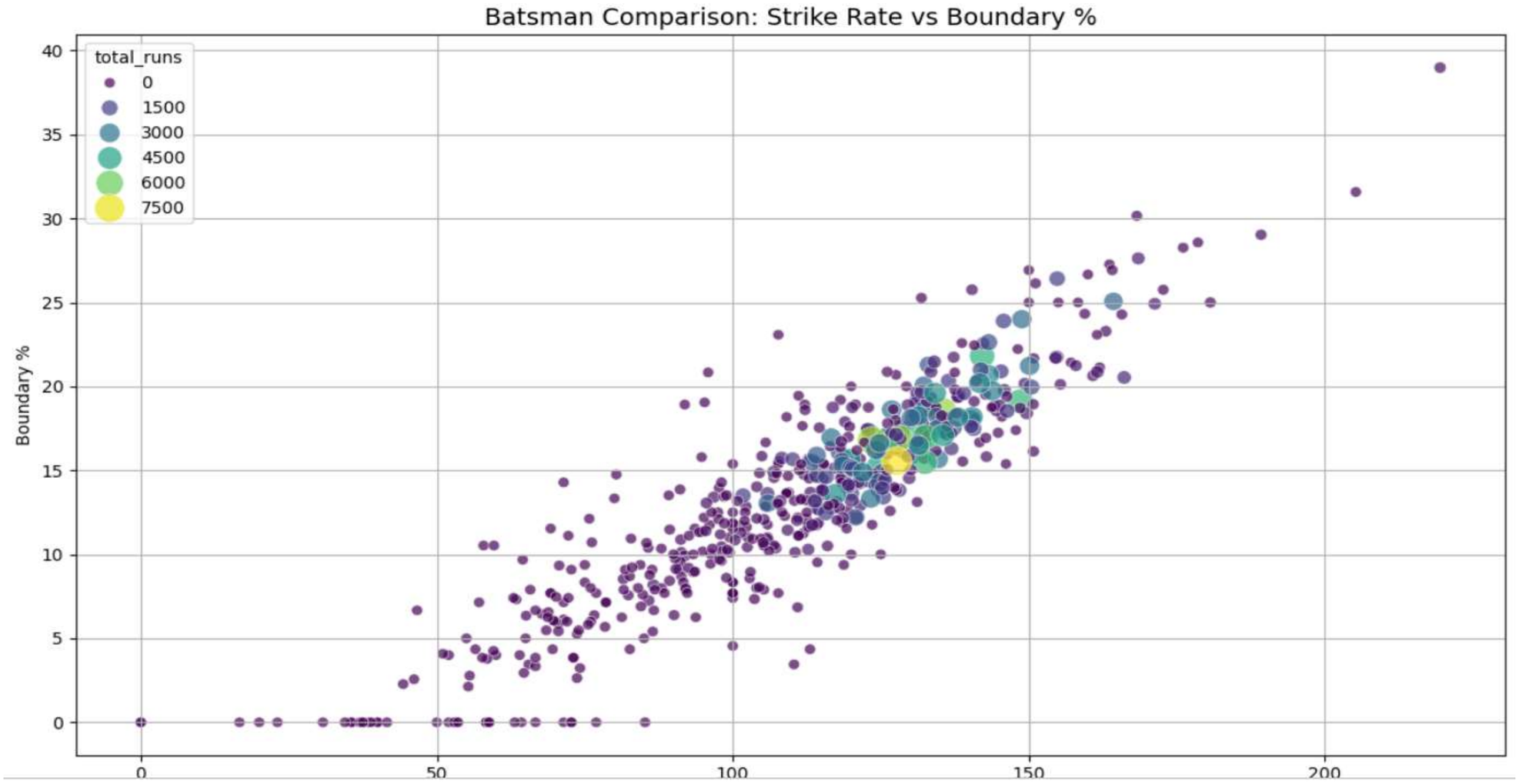
```
# Group by batsman
batting_summary = legal_deliveries.groupby('striker').agg(
    total_runs=('runs_scored', 'sum'),
    balls_faced=('striker', 'count'),
    boundaries=('is_boundary', 'sum')
).reset_index()
```

```
# Calculate strike rate and boundary %
batting_summary['strike_rate'] = (batting_summary['total_runs'] / batting_summary['balls_faced']) * 100
batting_summary['boundary_percent'] = (batting_summary['boundaries'] / batting_summary['balls_faced']) * 100
```

```
# Filter for minimum 100 balls faced
batting_summary = batting_summary[batting_summary['balls_faced'] >= 10]
```

```
# Merge player batting style
batting_summary = batting_summary.merge(
    players_info[['playername', 'battingstyle']],
    left_on='striker',
    right_on='playername',
    how='left'
)
```

```
# Plot strike rate vs boundary %
plt.figure(figsize=(12, 7))
sns.scatterplot(
    data=batting_summary,
    x='strike_rate',
    y='boundary_percent',
    size='total_runs',
    hue='total_runs',
    sizes=(40, 300),
    palette='viridis',
    legend='brief',
    alpha=0.7
)
plt.title('Batsman Comparison: Strike Rate vs Boundary %', fontsize=14)
plt.xlabel('Strike Rate')
plt.ylabel('Boundary %')
plt.grid(True)
plt.tight_layout()
plt.show()
```



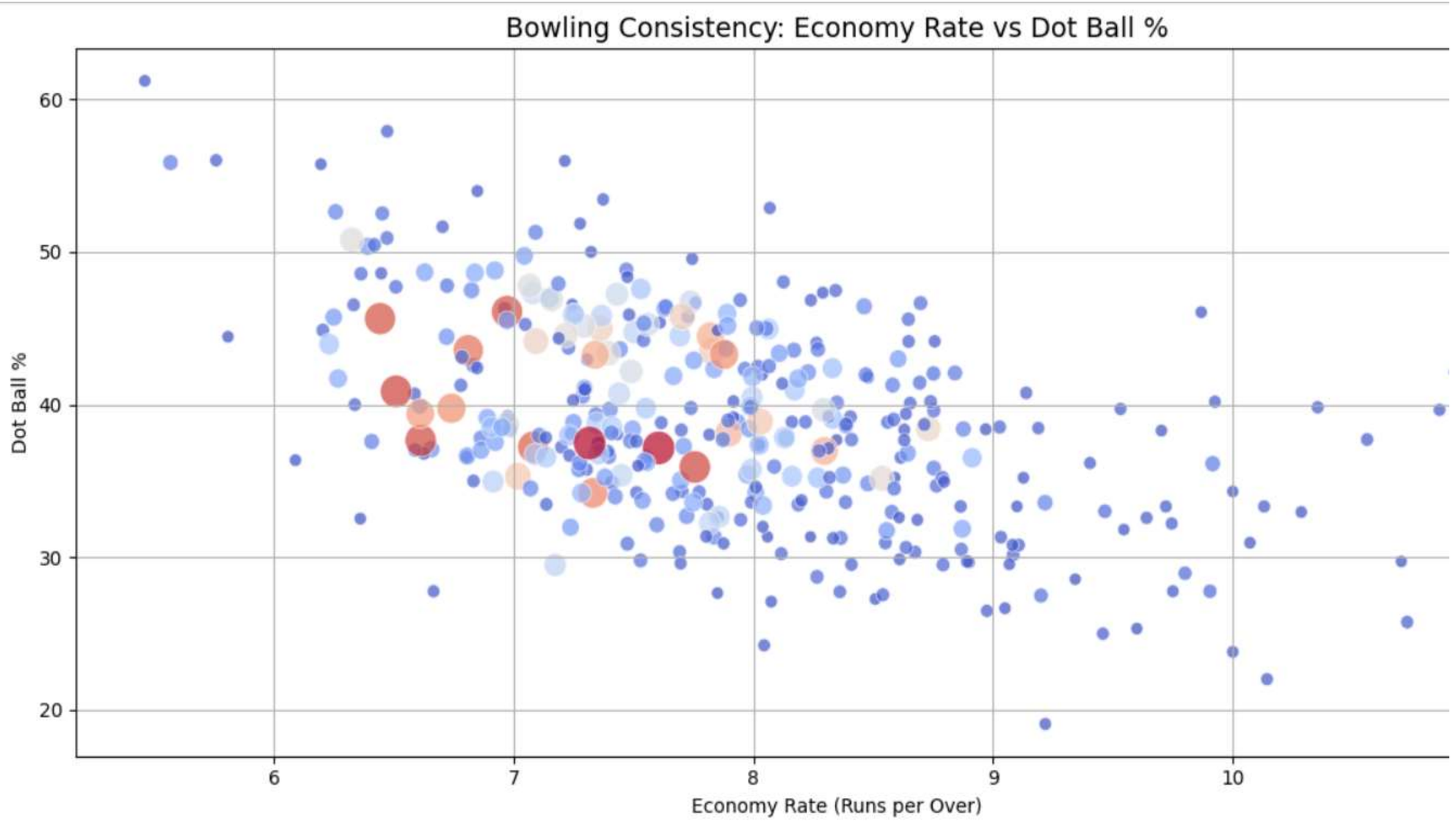

```
: print(legal_deliveries.columns)
```

```
Index(['matchid', 'date', 'season', 'battingteam', 'bowlingteam', 'inningsno',
       'ballno', 'bowler', 'striker', 'nonstriker', 'runs_scored', 'extras',
       'typeofextras', 'score', 'score/wicket', 'wicket_confirmation',
       'wicket_type', 'fielders_involved', 'playerout', 'playername',
       'teamname', 'playernationality', 'dateofbirth', 'playerrole',
       'ipldebut', 'about', 'battingstyle', 'bowlingstyle', 'playersalary',
       'over', 'over_phase', 'is_dot', 'is_wicket', 'is_boundary'],
      dtype='object')
```

4.BOWLING CONSISTENCY MATRIX

```python
# 1. Work with all deliveries
legal_deliveries = ball_by_ball.copy()

# 2. Create is_dot and is_wicket columns
legal_deliveries['is_dot'] = legal_deliveries['runs_scored'] == 0
legal_deliveries['is_wicket'] = legal_deliveries['wicket_type'].notna()

# 3. Group by bowler
bowling_summary = legal_deliveries.groupby('bowler').agg(
    total_balls=('bowler', 'size'),
    dot_balls=('is_dot', 'sum'),
    total_runs=('runs_scored', 'sum'),
    total_wickets=('is_wicket', 'sum')
).reset_index()

# 4. Calculate metrics
bowling_summary['overs_bowled'] = bowling_summary['total_balls'] / 6
bowling_summary['dot_ball_pct'] = (bowling_summary['dot_balls'] / bowling_summary['total_balls']) * 100
bowling_summary['economy_rate'] = bowling_summary['total_runs'] / bowling_summary['overs_bowled']
bowling_summary['bowling_average'] = bowling_summary['total_runs'] / bowling_summary['total_wickets']
bowling_summary.replace([float('inf'), -float('inf')], pd.NA, inplace=True)

# 5. Filter: At least 60 balls bowled
bowling_summary = bowling_summary[bowling_summary['total_balls'] >= 60]

# 6. Plot
plt.figure(figsize=(12, 6))
sns.scatterplot(
    data=bowling_summary,
    x='economy_rate',
    y='dot_ball_pct',
    size='total_wickets',
```

```python
.t.figure(figsize=(12, 6))
ıs.scatterplot(
    data=bowling_summary,
    x='economy_rate',
    y='dot_ball_pct',
    size='total_wickets',
    hue='total_wickets',
    palette='coolwarm',
    sizes=(40, 300),
    alpha=0.7,
    legend='brief'
)
plt.title("Bowling Consistency: Economy Rate vs Dot Ball %", fontsize=14)
plt.xlabel("Economy Rate (Runs per Over)")
plt.ylabel("Dot Ball %")
plt.grid(True)
plt.tight_layout()
plt.show()
```



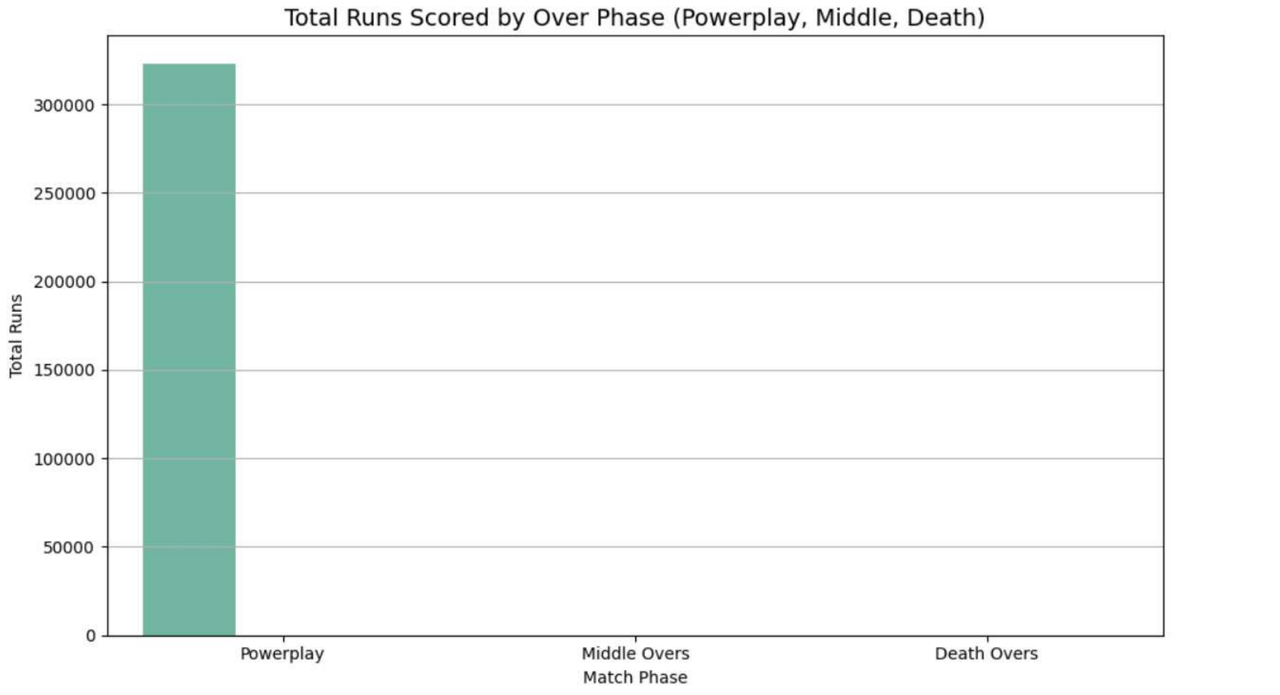

```
print(ball_by_ball.columns.tolist())
```

```
['matchid', 'date', 'season', 'battingteam', 'bowlingteam', 'inningsno', 'ballno', 'bowler', 'striker', 'nonstriker', 'runs_scored', 'extras', 'typeofextras', 'score', 'score/wicket', 'wicket_confirmation', 'wicket_type', 'fielders_involved', 'playerout', 'playername', 'teamname', 'playernationality', 'dateofbirth', 'playerrole', 'ipldebut', 'about', 'battingstyle', 'bowlingstyle', 'playersalary']
```

5.Performance by over phase (power play, middle, Death)

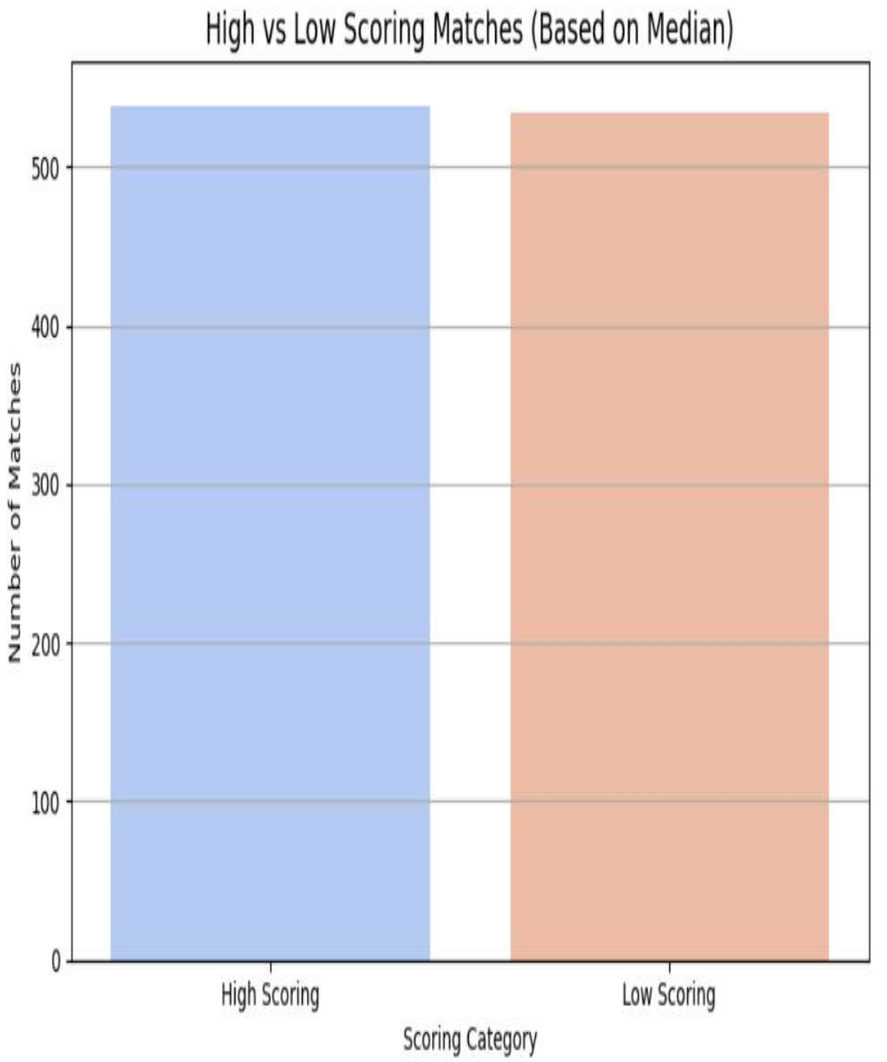
```python
# Extract over number
ball_by_ball['over'] = (ball_by_ball['ballno'] // 6 + 1).astype(int)

# Classify overs into phases
# Classify match phase
def label_phase(over):
    if 1 <= over <= 6:
        return 'Powerplay'
    elif 7 <= over <= 15:
        return 'Middle Overs'
    elif 16 <= over <= 20:
        return 'Death Overs'
    else:
        return 'Other'

ball_by_ball['over_phase'] = ball_by_ball['over'].apply(label_phase)

# Group by phase
phase_summary = ball_by_ball.groupby('over_phase')['runs_scored'].sum().reset_index()

# Sort & plot
import seaborn as sns
import matplotlib.pyplot as plt

phase_order = ['Powerplay', 'Middle Overs', 'Death Overs']
phase_summary['over_phase'] = pd.Categorical(phase_summary['over_phase'], categories=phase_order, ordered=True)
```

```python
# Sort & plot
import seaborn as sns
import matplotlib.pyplot as plt

phase_order = ['Powerplay', 'Middle Overs', 'Death Overs']
phase_summary['over_phase'] = pd.Categorical(phase_summary['over_phase'], categories=phase_order, ordered=True)
phase_summary = phase_summary.sort_values('over_phase')

plt.figure(figsize=(10, 6))
sns.barplot(data=phase_summary, x='over_phase', y='runs_scored', hue='over_phase', palette='Set2', legend=False)
plt.title('Total Runs Scored by Over Phase (Powerplay, Middle, Death)', fontsize=14)
plt.xlabel('Match Phase')
plt.ylabel('Total Runs')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



6.Venue Behaviour in High VS Low Scoring matches

```python
#  PART 6: Venue Behavior in High vs Low Scoring Matches

# Step 1: Total runs per match
match_runs = ball_by_ball.groupby('matchid')['runs_scored'].sum().reset_index()

# Step 2: Median split
median_runs = match_runs['runs_scored'].median()

# Step 3: Classify match type
match_runs['scoring_type'] = match_runs['runs_scored'].apply(
    lambda x: 'High Scoring' if x >= median_runs else 'Low Scoring'
)

# Step 4: Count match types
summary = match_runs['scoring_type'].value_counts().reset_index()
summary.columns = ['Scoring Type', 'Match Count']

# Step 5: Plot
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
sns.barplot(
    data=summary,
    x='Scoring Type',
    y='Match Count',
    hue='Scoring Type',
    palette='coolwarm',
    legend=False
)
plt.title('High vs Low Scoring Matches (Based on Median)', fontsize=14)
plt.xlabel('Scoring Category')
plt.ylabel('Number of Matches')
plt.grid(axis='y')
```



[ ]:

❖ IPL Python EDA Summary:

- ❑ Using python libraries like pandas and matplotlib ,clean and merges IPL dataset for consistent structure .Tracks run trends across season and overs (powerplay to death)
- ❑ Compares batting style using strike rate and boundary %
- ❑ Evaluate bowlers on dot balls ,economy and consistency.
- ❑ Analyzes venue behaviour in high vs low scouring matches
- ❑ Build a basic match-winner prediction model.

❖Key Findings

- **Top Performers**: Consistent players highlighted season-wise
- **Venue Advantage**: Certain grounds show high/low scoring biases
- **Team Strategy**: Toss impact, chasing vs defending stats
- **Performance Trends**: Evolution of strike rates, economy, and total scores

❖Conclusion

- ❑ Comprehensive analysis using SQL, Power BI, and Python
- ❑ Support :
  - ➢ **Team decision-making**
  - ➢ **Performance forecasting**
  - ➢ **Fan engagement strategies**
- ❑ Demonstrate the **value of data analytics in sports**

“You can have data without information,
but you cannot have information without data.”

- Daniel Keys Moran