Project Name: IPL 2025 Data Visualization

Develop By: Rudra Rathod

Dataset Year: 2025

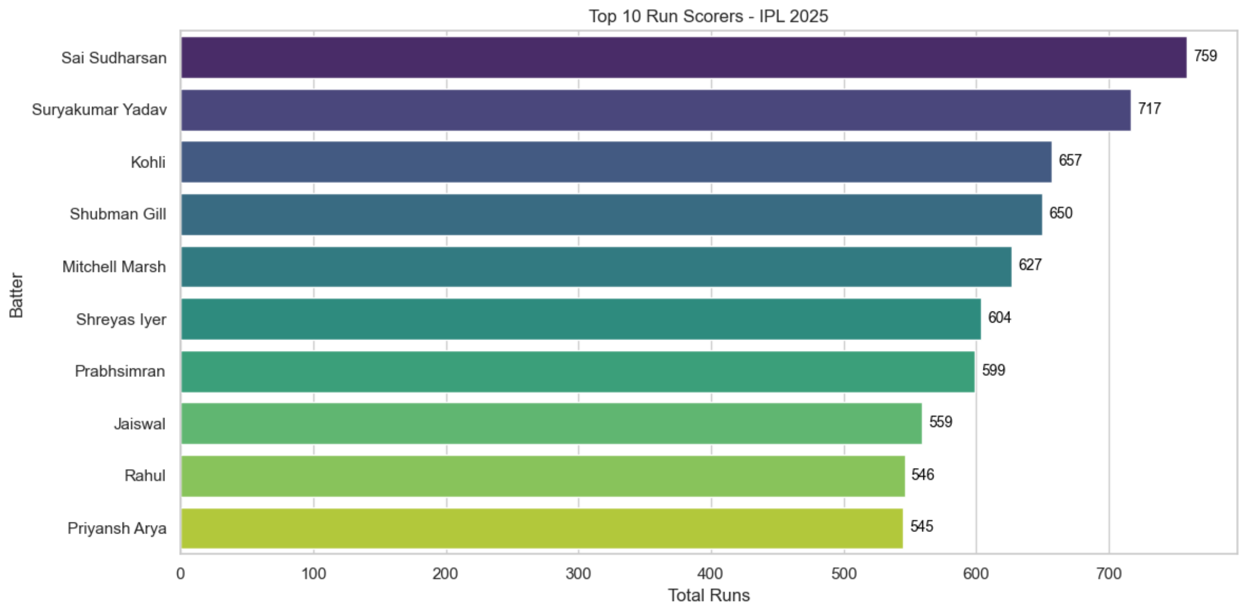
```
# Step 0: Import libraries and load data
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import numpy as np
import warnings
warnings.filterwarnings('ignore')
# For better plots
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (10,6)
# Load dataset
df = pd.read_csv('ipl_2025.csv')
# Quick peek
print(df.head())
print(f"Dataset shape: {df.shape}")
```

```
   match_id  season        phase  match_no          date  \
0    202501    2025  Group Stage         1  Mar 22, 2025
1    202501    2025  Group Stage         1  Mar 22, 2025
2    202501    2025  Group Stage         1  Mar 22, 2025
3    202501    2025  Group Stage         1  Mar 22, 2025
4    202501    2025  Group Stage         1  Mar 22, 2025

                  venue batting_team bowling_team  innings  over  ...
\
0  Eden Gardens, Kolkata          KKR          RCB        1   0.1  ...

1  Eden Gardens, Kolkata          KKR          RCB        1   0.2  ...

2  Eden Gardens, Kolkata          KKR          RCB        1   0.3  ...

3  Eden Gardens, Kolkata          KKR          RCB        1   0.4  ...

4  Eden Gardens, Kolkata          KKR          RCB        1   0.5  ...


     bowler runs_of_bat  extras  wide  legbyes  byes  noballs
wicket_type  \
0  Hazlewood           0       0     0        0     0        0
```

```
NaN
1  Hazlewood          4       0     0        0     0        0
NaN
2  Hazlewood          0       0     0        0     0        0
NaN
3  Hazlewood          0       0     0        0     0        0
NaN
4  Hazlewood          0       0     0        0     0        0
caught

  player_dismissed        fielder
0              NaN            NaN
1              NaN            NaN
2              NaN            NaN
3              NaN            NaN
4          de Kock  Jitesh Sharma

[5 rows x 21 columns]
Dataset shape: (17246, 21)
```

Top 10 Run Scorers - IPL 2025

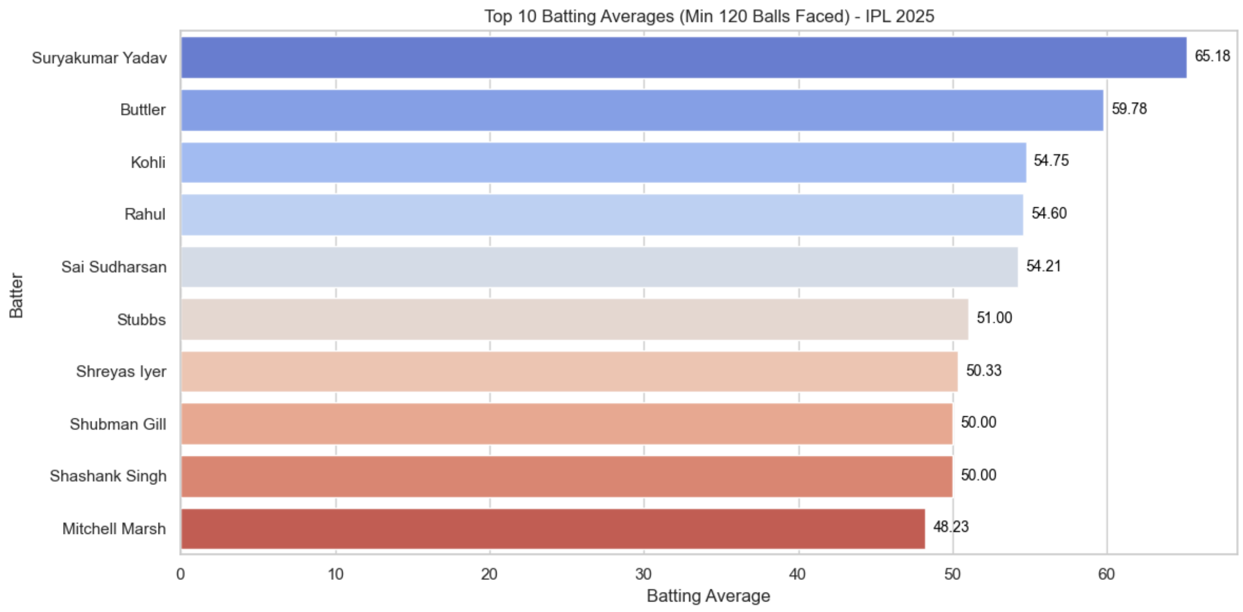
```
top_batters = df.groupby('striker')['runs_of_bat'].sum().reset_index()
# Step 2: Sort and select top 10
top_batters = top_batters.sort_values(by='runs_of_bat',
ascending=False).head(10)
# Step 3: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_batters, x='runs_of_bat', y='striker',
palette='viridis')
# Add text labels (run values) to the right side of bars
for i in ax.patches:
    plt.text(i.get_width() + 5,             # x-position (slightly
after the bar)
             i.get_y() + i.get_height() / 2,  # y-position (middle of
the bar)
             int(i.get_width()),              # the run value
             fontsize=10, color='black', va='center')
plt.title('Top 10 Run Scorers - IPL 2025')
plt.xlabel('Total Runs')
plt.ylabel('Batter')
plt.tight_layout()
plt.show()
```



Top 10 Batting Averages (Min 120 Balls Faced) - IPL 2025

```
# Step 1: Total runs per batter
total_runs = df.groupby('striker')
['runs_of_bat'].sum().reset_index(name='total_runs')
# Step 2: Number of dismissals per batter
dismissals = df[df['player_dismissed'].notna()]
dismissal_counts =
dismissals['player_dismissed'].value_counts().reset_index()
dismissal_counts.columns = ['striker', 'dismissals']
# Step 3: Merge runs and dismissals
batting_stats = pd.merge(total_runs, dismissal_counts, on='striker',
how='left')
batting_stats['dismissals'] = batting_stats['dismissals'].fillna(0)
batting_stats = batting_stats[batting_stats['dismissals'] > 0]
# Step 4: Calculate batting average
batting_stats['batting_average'] = batting_stats['total_runs'] /
batting_stats['dismissals']
# Step 5: Filter players with minimum balls faced (e.g., 120)
valid_balls = df[df['wide'].isna() | (df['wide'] == 0)]
balls_faced =
valid_balls.groupby('striker').size().reset_index(name='balls_faced')
batting_stats = pd.merge(batting_stats, balls_faced, on='striker',
how='left')
min_balls = 120
top_batting_average = batting_stats[batting_stats['balls_faced'] >=
min_balls]
```

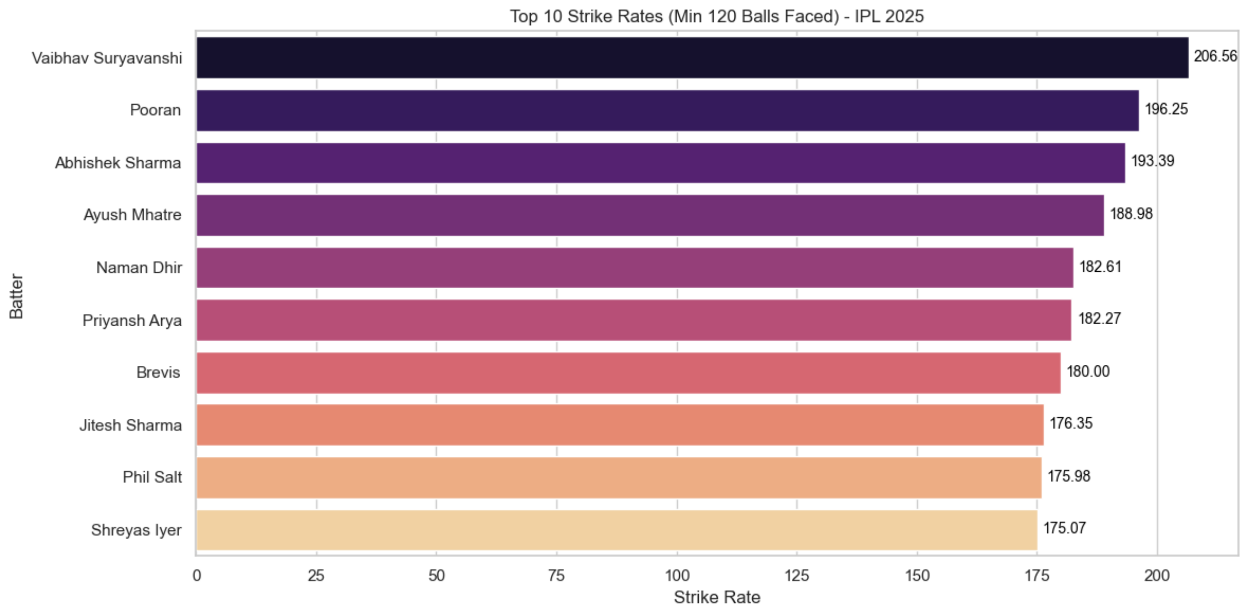
```
# Step 6: Sort by batting average descending and get top 10
top_batting_average =
top_batting_average.sort_values(by='batting_average',
ascending=False).head(10)
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_batting_average, x='batting_average',
y='striker', palette='coolwarm')
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.2f}",
             va='center', fontsize=10, color='black')
plt.title('Top 10 Batting Averages (Min 120 Balls Faced) - IPL 2025')
plt.xlabel('Batting Average')
plt.ylabel('Batter')
plt.tight_layout()
plt.show()
```



Top 10 Strike Rates (Min 120 Balls Faced) - IPL 2025

```
valid_balls = df[df['wide'].isna() | (df['wide'] == 0)]
# Step 1: Calculate total runs and balls faced for each batter
batter_stats = valid_balls.groupby('striker').agg(
    runs=('runs_of_bat', 'sum'),
    balls_faced=('striker', 'count')  # each row = a ball faced
).reset_index()
# Step 2: Calculate strike rate
```

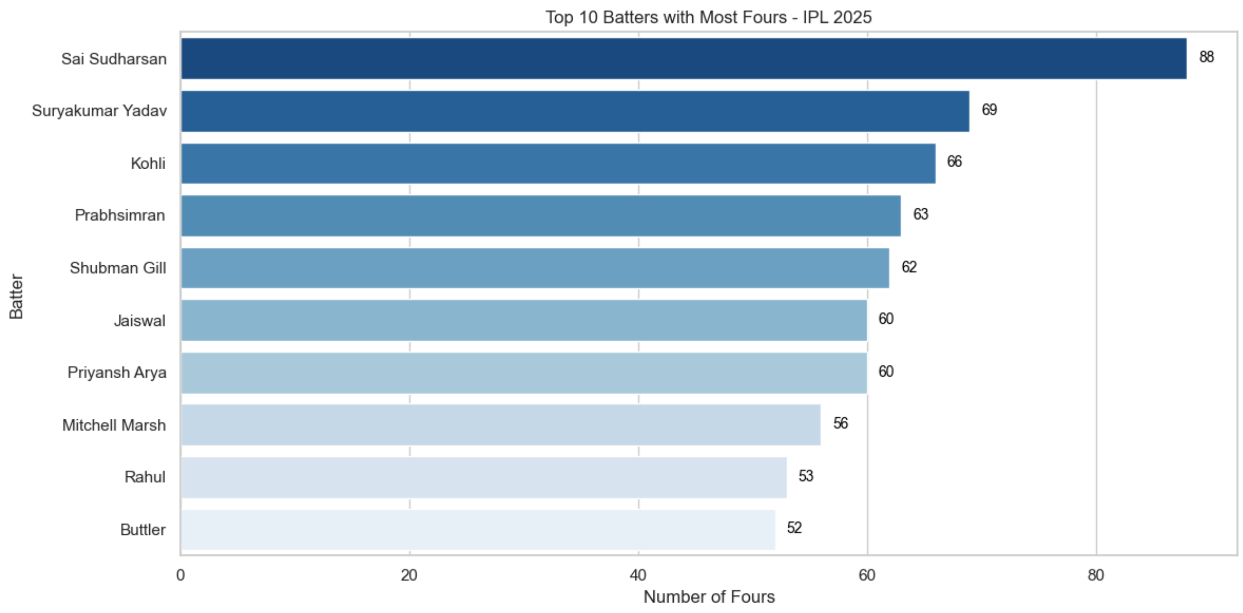
```
batter_stats['strike_rate'] = (batter_stats['runs'] /
batter_stats['balls_faced']) * 100
# Step 3: Filter batters with at least 120 balls faced
qualified_batters = batter_stats[batter_stats['balls_faced'] >= 120]
# Step 4: Sort by strike rate descending and get top 10
top_strike_rates = qualified_batters.sort_values(by='strike_rate',
ascending=False).head(10)
# Plotting the top strike rates
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_strike_rates, x='strike_rate', y='striker',
palette='magma')
# Add strike rate labels at the end of bars
for bar in ax.patches:
    plt.text(bar.get_width() + 1,                         # x-position
             bar.get_y() + bar.get_height() / 2,    # y-position
             f"{bar.get_width():.2f}",              # strike rate
value
             va='center', fontsize=10, color='black')
plt.title('Top 10 Strike Rates (Min 120 Balls Faced) - IPL 2025')
plt.xlabel('Strike Rate')
plt.ylabel('Batter')
plt.tight_layout()
plt.show()
```



Top 10 Batters with Most Fours - IPL 2025

```
# Step 1: Filter only fours
fours_df = df[df['runs_of_bat'] == 4]
# Step 2: Count fours per batter
```

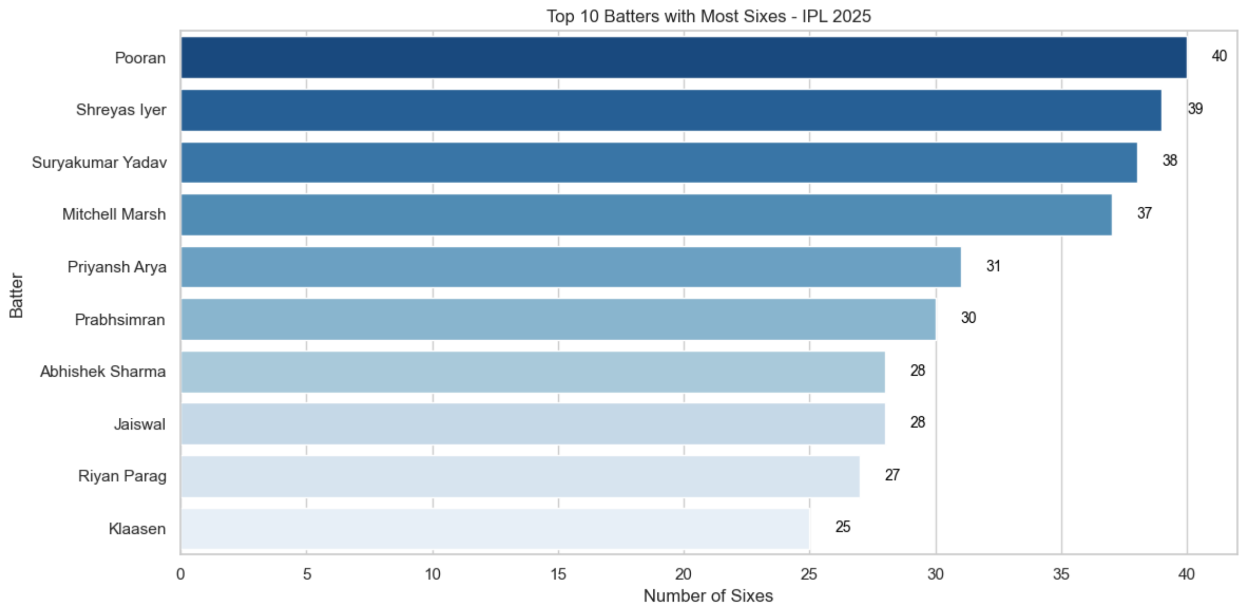
```python
fours_count =
fours_df.groupby('striker').size().reset_index(name='fours')
# Step 3: Sort and get top 10
top_fours = fours_count.sort_values(by='fours',
ascending=False).head(10)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_fours, x='fours', y='striker',
palette='Blues_r')
# Add labels to bars
for bar in ax.patches:
    plt.text(bar.get_width() + 1,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=10, color='black')
plt.title('Top 10 Batters with Most Fours - IPL 2025')
plt.xlabel('Number of Fours')
plt.ylabel('Batter')
plt.tight_layout()
plt.show()
```



Top 10 Batters with Most Sixes - IPL 2025

```python
# Step 1: Filter only sixes
sixes_df = df[df['runs_of_bat'] == 6]
# Step 2: Count sixes per batter
sixes_count =
sixes_df.groupby('striker').size().reset_index(name='sixes')
# Step 3: Sort and get top 10
```

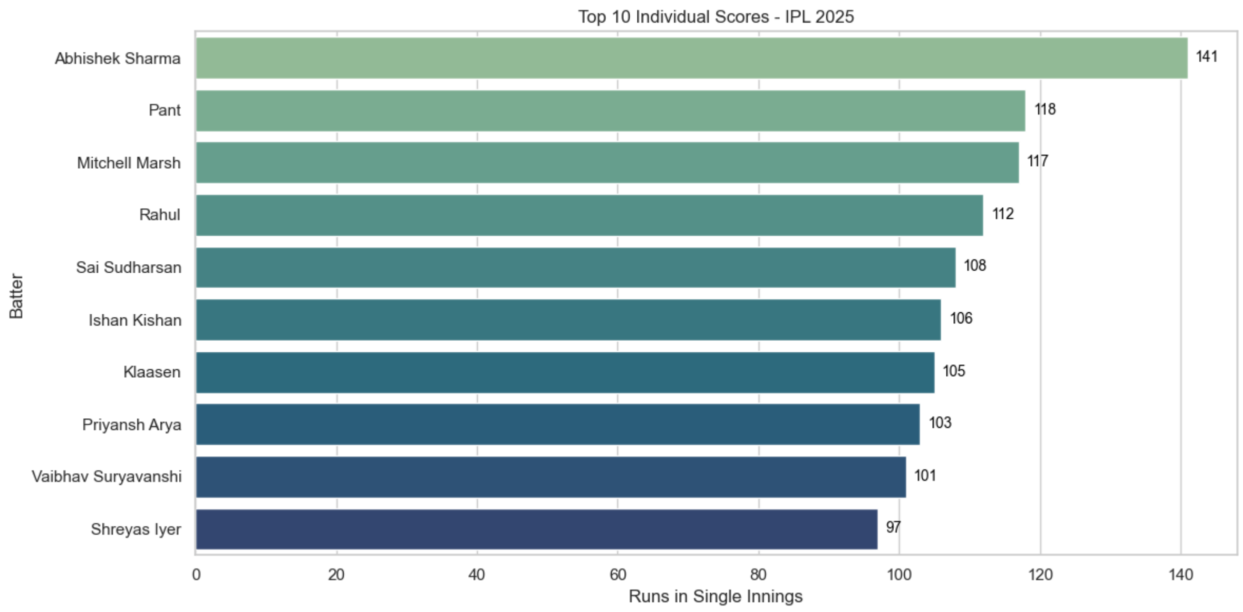
```
top_sixes = sixes_count.sort_values(by='sixes',
ascending=False).head(10)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_sixes, x='sixes', y='striker',
palette='Blues_r')
# Add labels to bars
for bar in ax.patches:
    plt.text(bar.get_width() + 1,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=10, color='black')
plt.title('Top 10 Batters with Most Sixes - IPL 2025')
plt.xlabel('Number of Sixes')
plt.ylabel('Batter')
plt.tight_layout()
plt.show()
```



Top 10 Individual Scores in a Single Innings - IPL 2025

```
individual_scores = df.groupby(['match_id', 'innings', 'striker'])
['runs_of_bat'].sum().reset_index()
# Step 2: Sort by runs scored in descending order
top_scores = individual_scores.sort_values(by='runs_of_bat',
ascending=False).head(10)
# Step 3: Plotting
plt.figure(figsize=(12, 6))
```

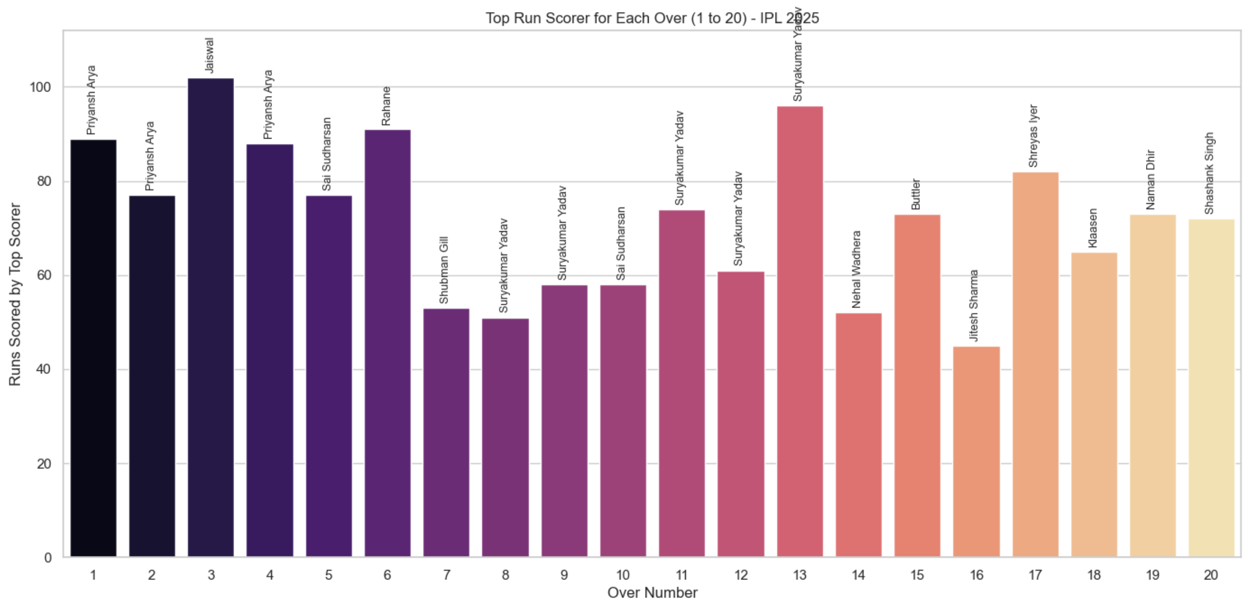
```
ax = sns.barplot(data=top_scores, x='runs_of_bat', y='striker',
palette='crest')
# Add score labels to bars
for bar in ax.patches:
    plt.text(bar.get_width() + 1,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=10, color='black')
plt.title('Top 10 Individual Scores - IPL 2025')
plt.xlabel('Runs in Single Innings')
plt.ylabel('Batter')
plt.tight_layout()
plt.show()
```



Top Run Scorer for Each Over (1 to 20) - IPL 2025

```
# Extract integer over number (e.g., 0.1 -> 0, 19.6 -> 19)
df['over_int'] = df['over'].astype(float).apply(int)
# Group by integer over and striker, sum runs
runs_per_over_player = df.groupby(['over_int', 'striker'])
['runs_of_bat'].sum().reset_index()
# For each over, find the player with max runs
top_scorers_per_over = runs_per_over_player.loc[
    runs_per_over_player.groupby('over_int')['runs_of_bat'].idxmax()
].reset_index(drop=True)
# Sort by over number
top_scorers_per_over = top_scorers_per_over.sort_values('over_int')
```

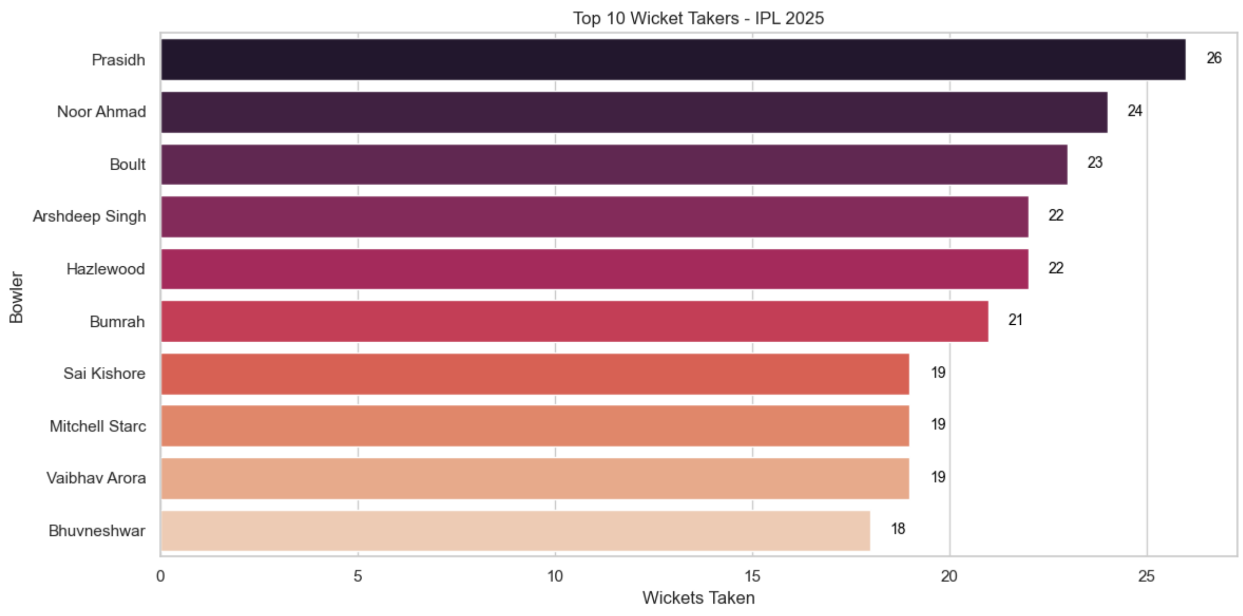
```
# Rename 'over_int' to 'over' for clarity and shift over by 1
top_scorers_per_over =
top_scorers_per_over.rename(columns={'over_int': 'over'})
top_scorers_per_over['over'] = top_scorers_per_over['over'] + 1
# Plot horizontal bar chart
plt.figure(figsize=(14, 7))
ax = sns.barplot(data=top_scorers_per_over, x='over', y='runs_of_bat',
palette='magma')
# Add player names on the bars
for index, row in top_scorers_per_over.iterrows():
    ax.text(row['over'] - 1,  # bar x-position (0-based index)
            row['runs_of_bat'] + 1,  # slightly above the bar
            row['striker'],
            rotation=90, fontsize=9, ha='center', va='bottom')
plt.title('Top Run Scorer for Each Over (1 to 20) - IPL 2025')
plt.xlabel('Over Number')
plt.ylabel('Runs Scored by Top Scorer')
plt.xticks(range(0, 20), range(1, 21))  # show 1 to 20 as ticks
plt.ylim(0, top_scorers_per_over['runs_of_bat'].max() + 10)
plt.tight_layout()
plt.show()
```



Top 10 Wicket Takers - IPL 2025

```
wickets = df[df['player_dismissed'].notna()]
# Step 2: Count wickets per bowler
wicket_counts =
wickets.groupby('bowler').size().reset_index(name='wickets')
# Step 3: Sort and get top 10 wicket takers
```

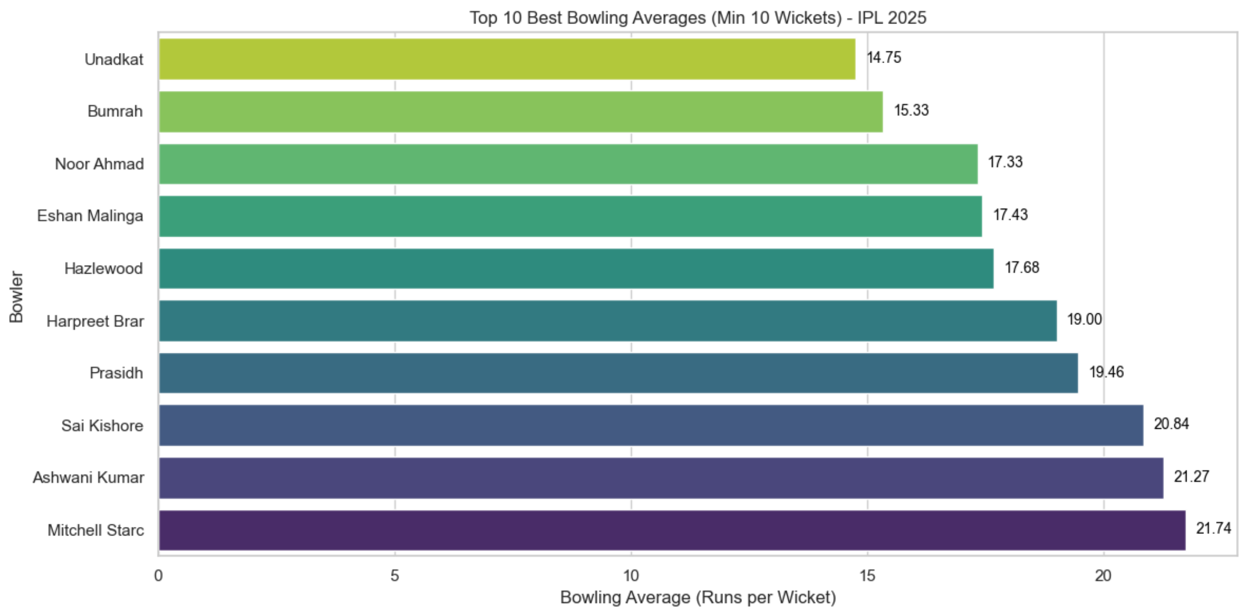
```
top_wicket_takers = wicket_counts.sort_values(by='wickets',
ascending=False).head(10)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_wicket_takers, x='wickets', y='bowler',
palette='rocket')
# Add wicket count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=10, color='black')

plt.title('Top 10 Wicket Takers - IPL 2025')
plt.xlabel('Wickets Taken')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



Top 10 Best Bowling Averages (Min 10 Wickets) - IPL 2025

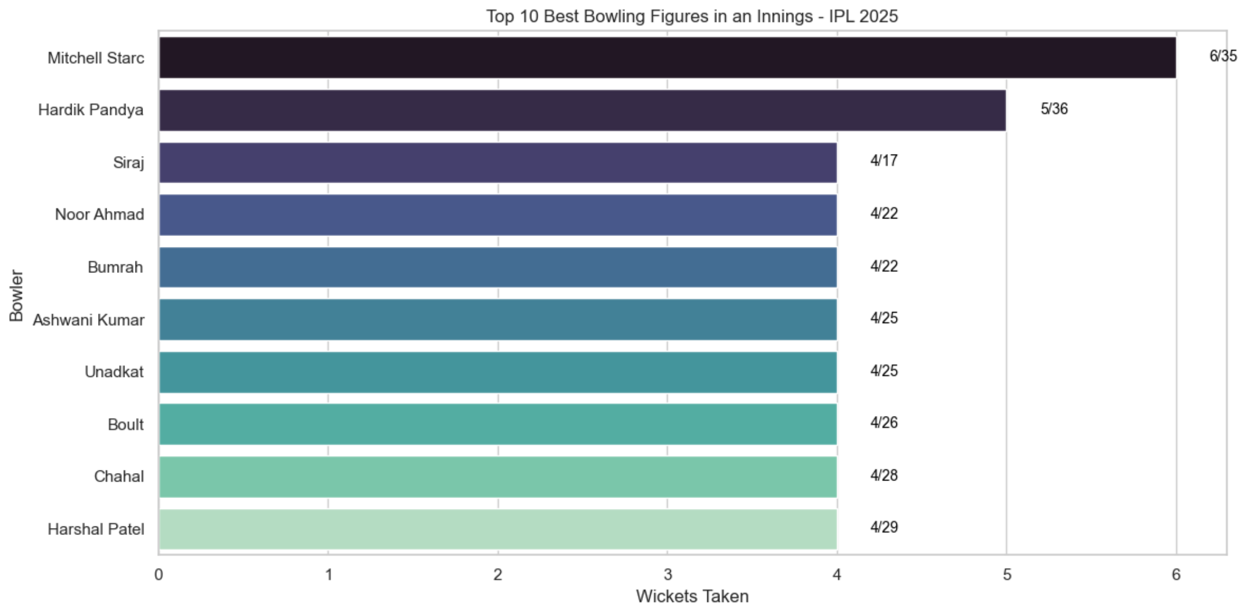
```
# Step 1: Calculate runs conceded per bowler (including extras)
df['total_runs_conceded'] = df['runs_of_bat'] + df['extras'].fillna(0)
runs_conceded = df.groupby('bowler')
['total_runs_conceded'].sum().reset_index()
# Step 2: Count wickets taken by each bowler
wickets = df[df['player_dismissed'].notna()]
```

```
wicket_counts =
wickets.groupby('bowler').size().reset_index(name='wickets')
# Step 3: Merge runs conceded and wickets taken
bowling_stats = pd.merge(runs_conceded, wicket_counts, on='bowler',
how='inner')
# Step 4: Filter bowlers with minimum wickets (e.g., 10)
min_wickets = 10
bowling_stats = bowling_stats[bowling_stats['wickets'] >= min_wickets]
# Step 5: Calculate bowling average
bowling_stats['bowling_average'] =
bowling_stats['total_runs_conceded'] / bowling_stats['wickets']
# Step 6: Sort by bowling average ascending (best averages first)
best_bowling_average =
bowling_stats.sort_values(by='bowling_average').head(10)
# Step 7: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=best_bowling_average, x='bowling_average',
y='bowler', palette='viridis_r')
# Add labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.2f}",
             va='center', fontsize=10, color='black')
plt.title('Top 10 Best Bowling Averages (Min 10 Wickets) - IPL 2025')
plt.xlabel('Bowling Average (Runs per Wicket)')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



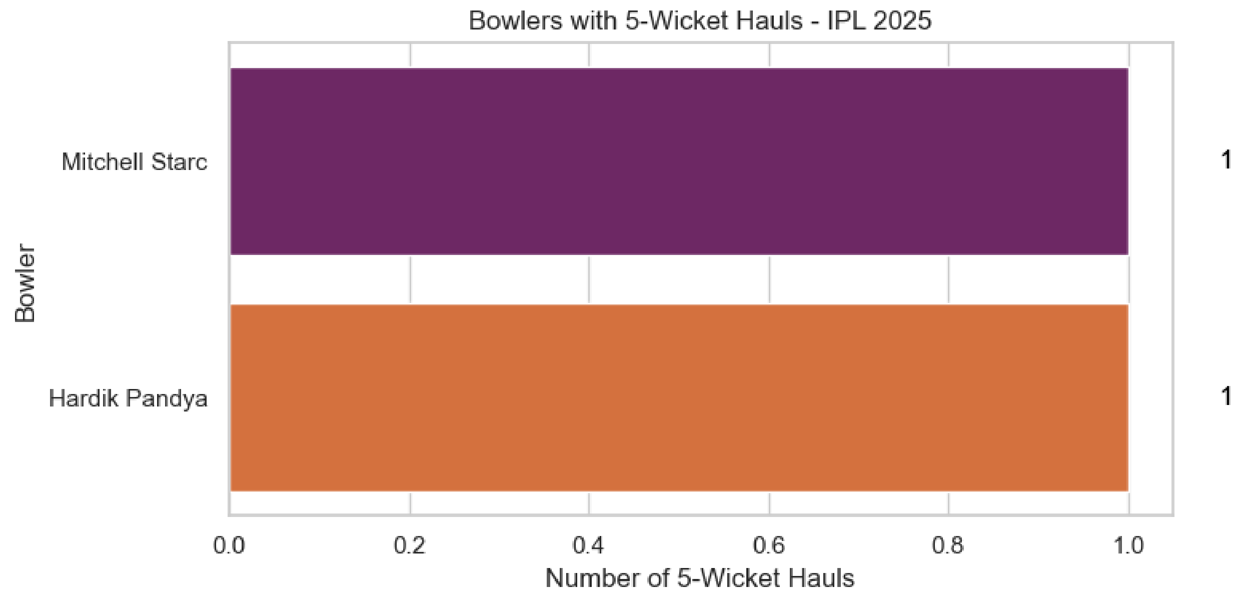
Top 10 Best Bowling Figures in an Innings - IPL 2025

```python
# Step 1: Calculate runs conceded per ball (runs_of_bat + extras)
df['total_runs_conceded'] = df['runs_of_bat'] + df['extras'].fillna(0)
# Step 2: Calculate wickets per ball: mark 1 if player dismissed, else
0
df['wicket'] = df['player_dismissed'].notna().astype(int)
# Step 3: Group by match, innings, and bowler to get runs and wickets
in that innings
bowling_innings = df.groupby(['match_id', 'innings', 'bowler']).agg(
    runs_conceded=('total_runs_conceded', 'sum'),
    wickets=('wicket', 'sum')
).reset_index()
# Step 4: Filter to only bowlers who took at least 1 wicket
bowling_innings = bowling_innings[bowling_innings['wickets'] > 0]
# Step 5: Sort by wickets descending, then runs ascending to get best
bowling figures
best_figures = bowling_innings.sort_values(by=['wickets',
'runs_conceded'], ascending=[False, True]).head(10)
# Step 6: Create a combined "figures" string like '5/24'
(wickets/runs)
best_figures['figures'] = best_figures['wickets'].astype(str) + '/' +
best_figures['runs_conceded'].astype(int).astype(str)
# Step 7: Plotting best bowling figures
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=best_figures, x='wickets', y='bowler',
palette='mako')
# Add runs conceded and figure label next to bars
for i, bar in enumerate(ax.patches):
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             f"{best_figures.iloc[i]['figures']}",
             va='center', fontsize=10, color='black')
plt.title('Top 10 Best Bowling Figures in an Innings - IPL 2025')
plt.xlabel('Wickets Taken')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



Bowlers with 5-Wicket Hauls - IPL 2025

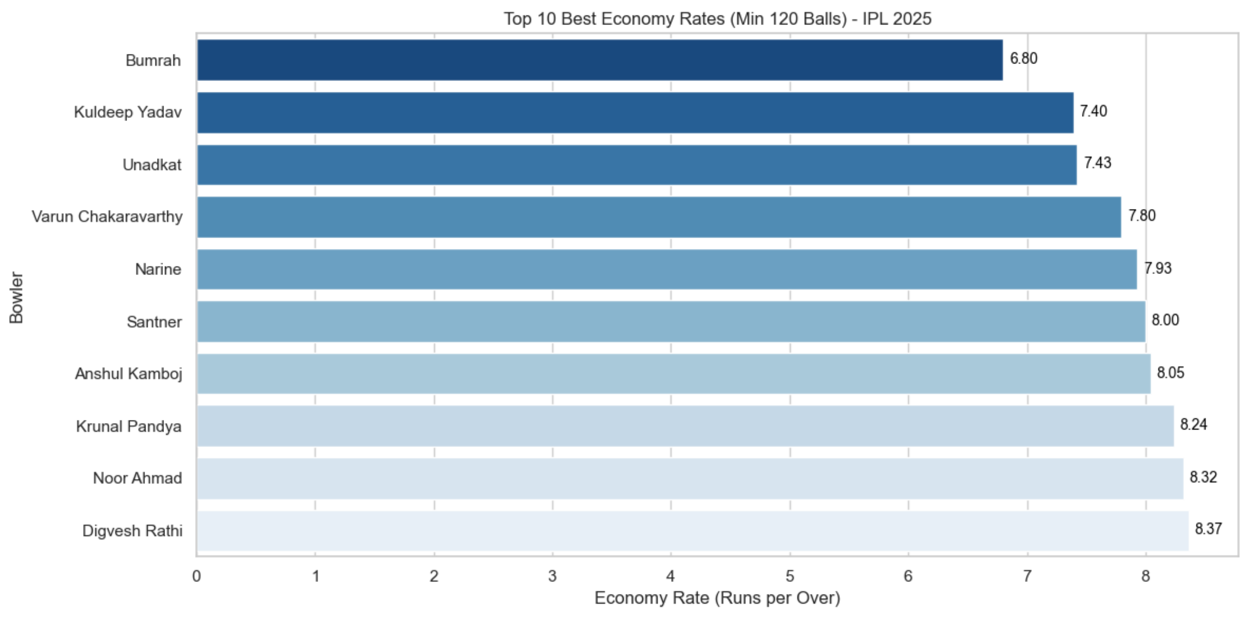
```
# Step 1: Mark wickets (1 if player dismissed else 0)
df['wicket'] = df['player_dismissed'].notna().astype(int)
# Step 2: Group by match, innings, bowler to get wickets per innings
wickets_per_innings = df.groupby(['match_id', 'innings', 'bowler'])
['wicket'].sum().reset_index()
# Step 3: Filter innings where wickets >= 5
five_wicket_hauls = wickets_per_innings[wickets_per_innings['wicket']
>= 5]
# Step 4: Count number of 5-wicket hauls per bowler
five_wicket_counts =
five_wicket_hauls['bowler'].value_counts().reset_index()
five_wicket_counts.columns = ['bowler', 'five_wicket_hauls']
# Step 5: Sort descending and get top 10
top_five_wicket_bowlers =
five_wicket_counts.sort_values(by='five_wicket_hauls',
ascending=False).head(10)
plt.figure(figsize=(8, 4))  # Smaller figure since only 2 bars
ax = sns.barplot(data=top_five_wicket_bowlers, x='five_wicket_hauls',
y='bowler', palette='inferno')
for bar in ax.patches:
    plt.text(bar.get_width() + 0.1,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Bowlers with 5-Wicket Hauls - IPL 2025')
plt.xlabel('Number of 5-Wicket Hauls')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



Top 10 Best Economy Rates (Min 120 Balls) - IPL 2025

```
# Step 1: Calculate total runs conceded per ball
df['total_runs_conceded'] = df['runs_of_bat'] + df['extras'].fillna(0)
# Step 2: Filter out illegal deliveries for balls count (exclude wides
and no-balls)
valid_balls = df[(df['wide'].isna() | (df['wide'] == 0)) &
(df['noballs'].isna() | (df['noballs'] == 0))]
# Step 3: Calculate balls bowled per bowler (valid balls only)
balls_bowled =
valid_balls.groupby('bowler').size().reset_index(name='balls_bowled')
# Step 4: Calculate runs conceded per bowler (including extras)
runs_conceded = df.groupby('bowler')
['total_runs_conceded'].sum().reset_index()
# Step 5: Merge balls bowled and runs conceded
bowling_stats = pd.merge(runs_conceded, balls_bowled, on='bowler',
how='inner')
# Step 6: Filter bowlers with minimum balls bowled (e.g., 120 balls =
20 overs)
min_balls = 120
bowling_stats = bowling_stats[bowling_stats['balls_bowled'] >=
min_balls]
# Step 7: Calculate economy rate (runs per over)
bowling_stats['economy_rate'] = bowling_stats['total_runs_conceded'] /
```

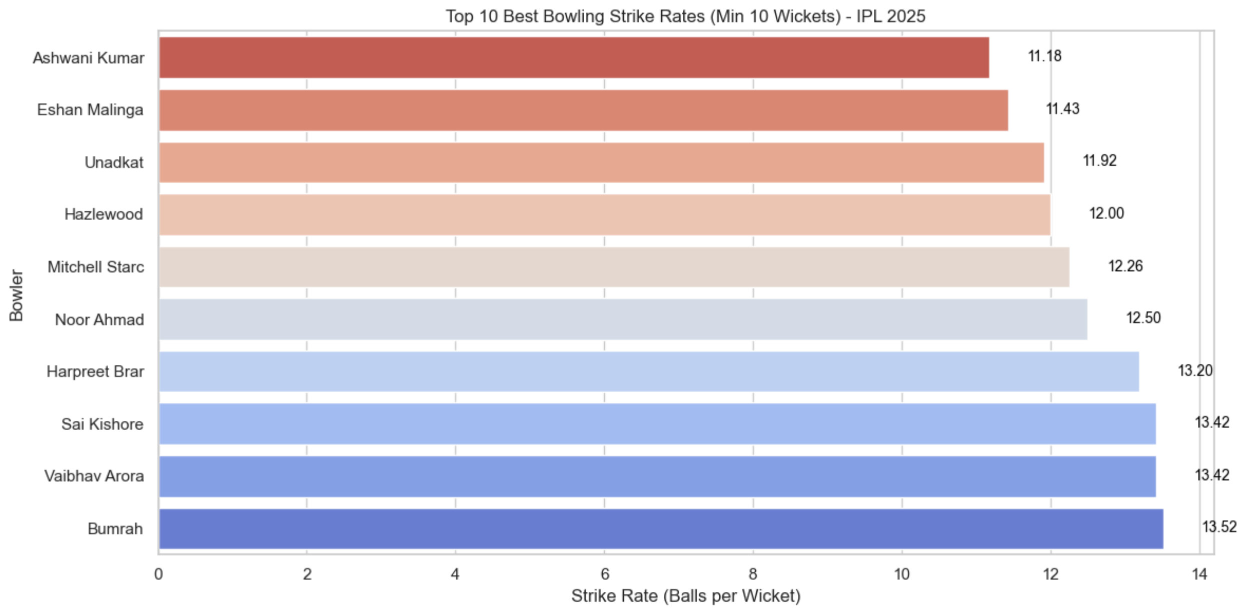
```
(bowling_stats['balls_bowled'] / 6)
# Step 8: Sort by economy rate ascending (best economy first)
best_economy = bowling_stats.sort_values(by='economy_rate').head(10)
# Step 9: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=best_economy, x='economy_rate', y='bowler',
palette='Blues_r')
# Add labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.05,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.2f}",
             va='center', fontsize=10, color='black')

plt.title('Top 10 Best Economy Rates (Min 120 Balls) - IPL 2025')
plt.xlabel('Economy Rate (Runs per Over)')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



Top 10 Best Bowling Strike Rates (Min 10 Wickets) - IPL 2025

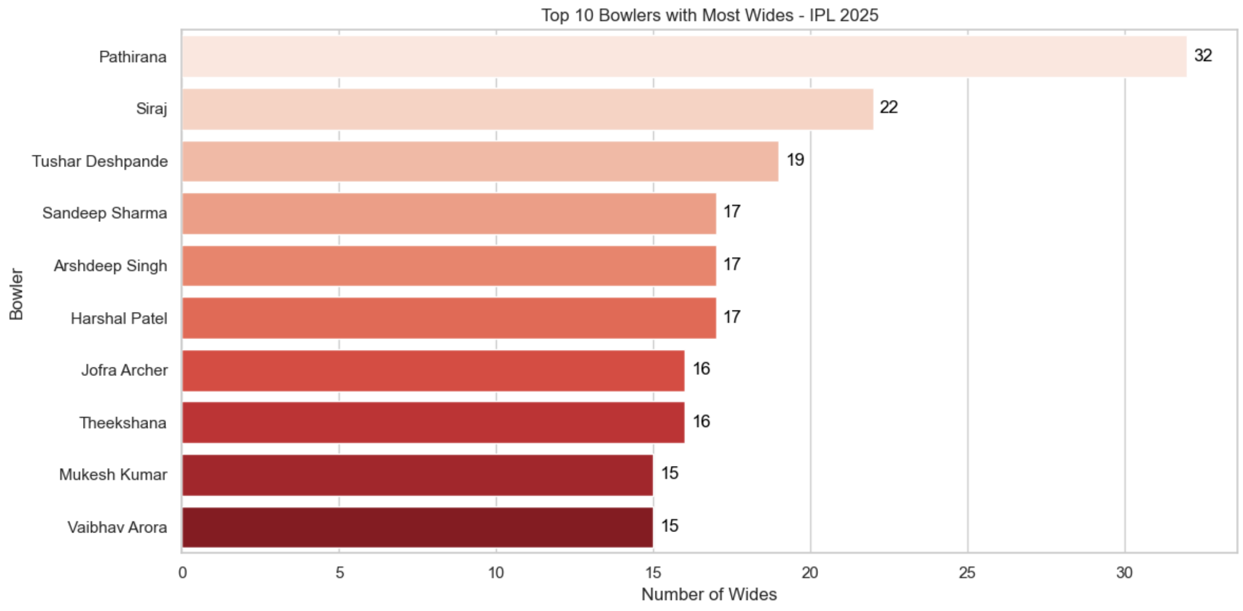
```
# Step 1: Mark wickets per delivery
df['wicket'] = df['player_dismissed'].notna().astype(int)
# Step 2: Filter out illegal deliveries (exclude wides and no-balls)
for counting balls bowled
valid_balls = df[(df['wide'].isna() | (df['wide'] == 0)) &
```

```python
(df['noballs'].isna() | (df['noballs'] == 0))]
# Step 3: Calculate balls bowled per bowler
balls_bowled =
valid_balls.groupby('bowler').size().reset_index(name='balls_bowled')
# Step 4: Calculate wickets taken per bowler
wickets_taken = df.groupby('bowler')['wicket'].sum().reset_index()
# Step 5: Merge balls bowled and wickets taken
bowling_stats = pd.merge(balls_bowled, wickets_taken, on='bowler',
how='inner')
# Step 6: Filter bowlers with minimum wickets (e.g., 10)
min_wickets = 10
bowling_stats = bowling_stats[bowling_stats['wicket'] >= min_wickets]
# Step 7: Calculate bowling strike rate
bowling_stats['strike_rate'] = bowling_stats['balls_bowled'] /
bowling_stats['wicket']
# Step 8: Sort by strike rate ascending (best first)
best_strike_rate =
bowling_stats.sort_values(by='strike_rate').head(10)
# Step 9: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=best_strike_rate, x='strike_rate', y='bowler',
palette='coolwarm_r')
# Add labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.2f}",
             va='center', fontsize=10, color='black')
plt.title('Top 10 Best Bowling Strike Rates (Min 10 Wickets) - IPL
2025')
plt.xlabel('Strike Rate (Balls per Wicket)')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



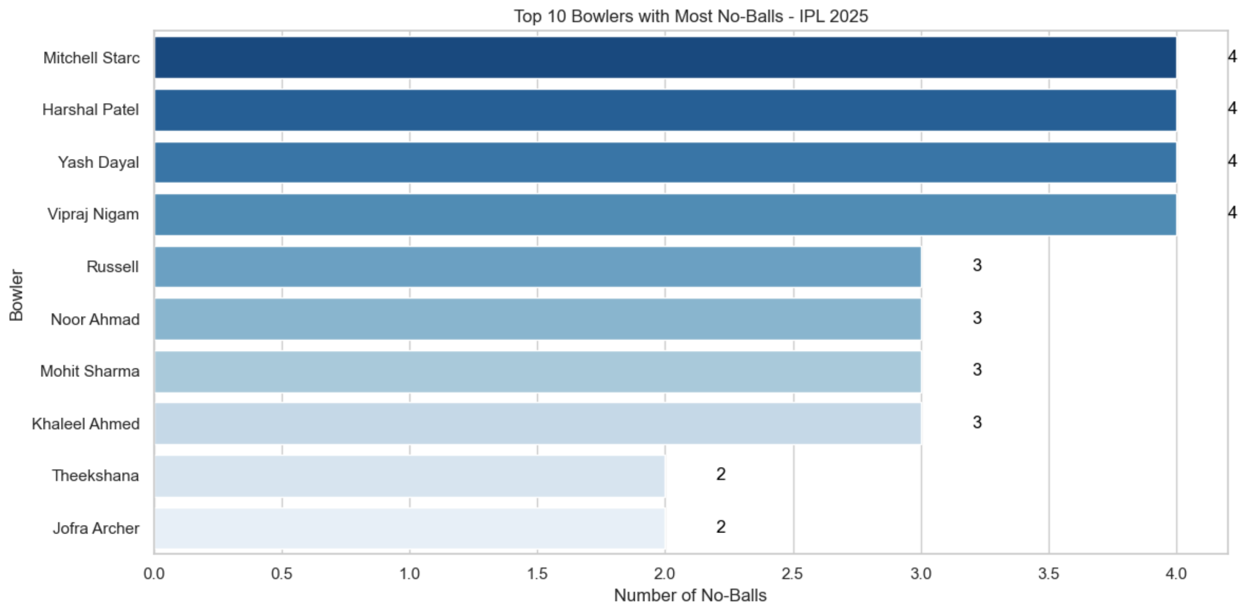
Top 10 Bowlers with Most Wides - IPL 2025

```
# Step 1: Filter deliveries where wides were bowled (wide > 0)
wides = df[df['wide'] > 0]
# Step 2: Count wides by bowler
wides_per_bowler =
wides.groupby('bowler').size().reset_index(name='wides')
# Step 3: Sort descending to find the bowler with the most wides
wides_per_bowler = wides_per_bowler.sort_values(by='wides',
ascending=False)
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=wides_per_bowler.head(10), x='wides',
y='bowler', palette='Reds')
for bar in ax.patches:
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Top 10 Bowlers with Most Wides - IPL 2025')
plt.xlabel('Number of Wides')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



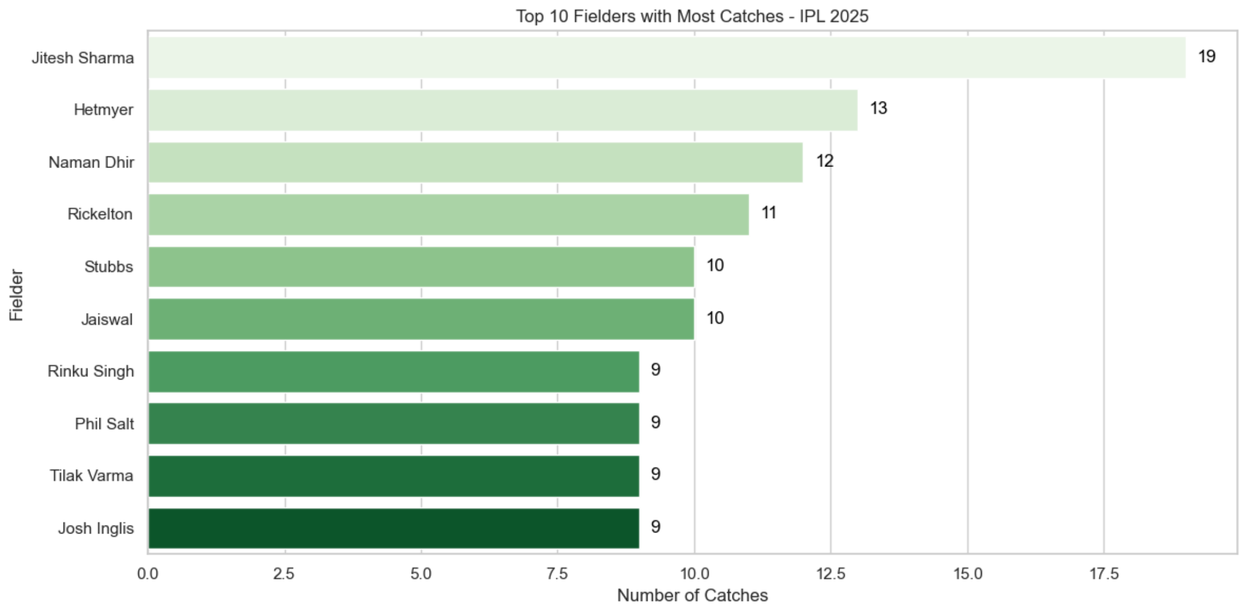
Top 10 Bowlers with Most No-Balls - IPL 2025

```
# Filter deliveries in 2025 season where no-balls were bowled (noballs
> 0)
no_balls = df[(df['season'] == 2025) & (df['noballs'] > 0)]
# Count no-balls by bowler
no_balls_per_bowler =
no_balls.groupby('bowler').size().reset_index(name='no_balls')
# Sort descending to find bowlers with most no-balls
no_balls_per_bowler = no_balls_per_bowler.sort_values(by='no_balls',
ascending=False)
# Plot top 10 bowlers with most no-balls in 2025
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=no_balls_per_bowler.head(10), x='no_balls',
y='bowler', palette='Blues_r')
# Add numbers on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')

plt.title('Top 10 Bowlers with Most No-Balls - IPL 2025')
plt.xlabel('Number of No-Balls')
plt.ylabel('Bowler')
plt.tight_layout()
plt.show()
```



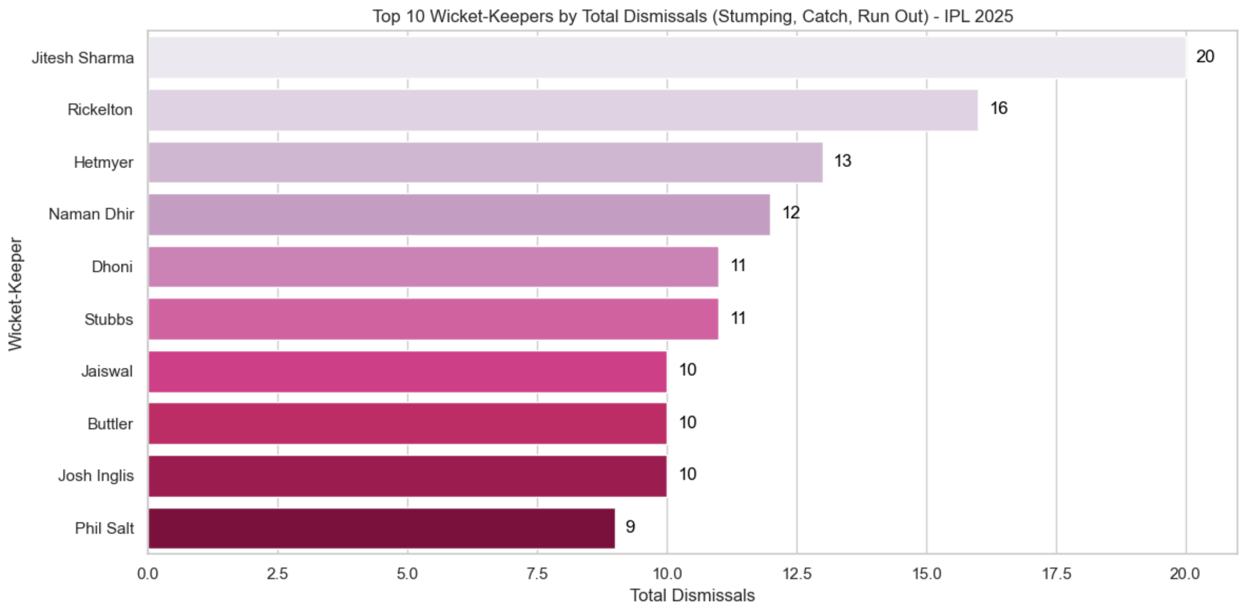
Top 10 Fielders with Most Catches - IPL 2025

```
# Step 1: Filter deliveries where wicket_type is 'caught' and fielder
is not null
caught_deliveries = df[(df['wicket_type'] == 'caught') &
(df['fielder'].notna())]
# Step 2: Count catches per fielder
catch_counts =
caught_deliveries['fielder'].value_counts().reset_index()
catch_counts.columns = ['fielder', 'catches']
# Step 3: Get top 10 catchers
top_catchers = catch_counts.head(10)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_catchers, x='catches', y='fielder',
palette='Greens')
# Add catch count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Top 10 Fielders with Most Catches - IPL 2025')
plt.xlabel('Number of Catches')
plt.ylabel('Fielder')
plt.tight_layout()
plt.show()
```



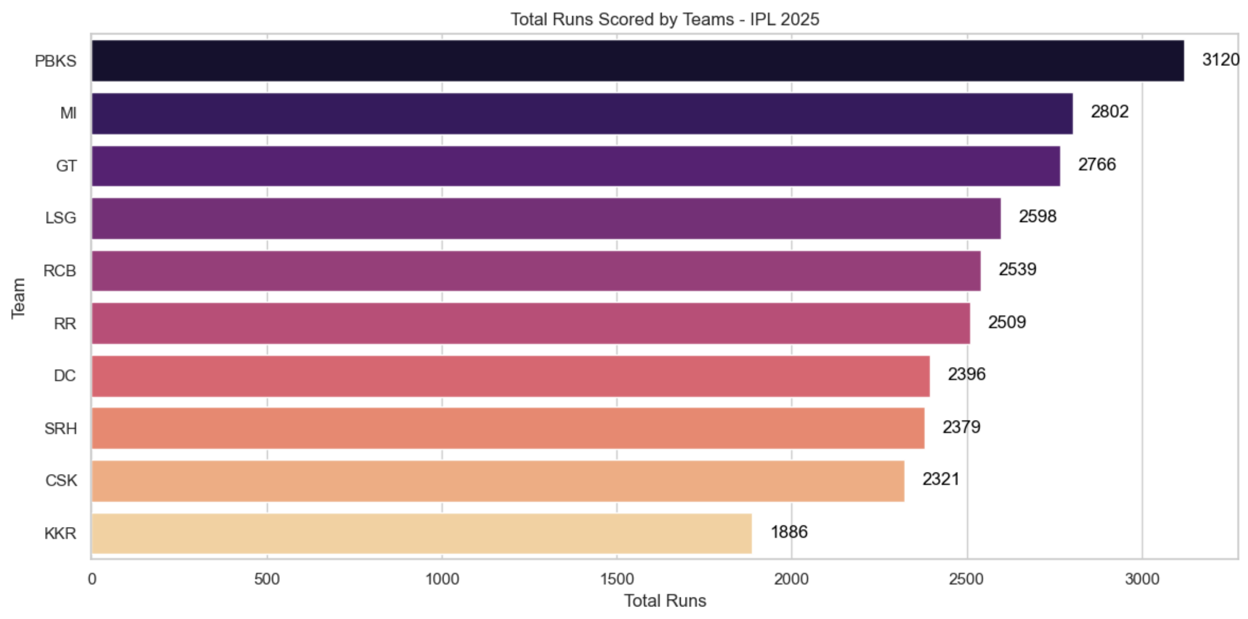
Top 10 Wicket-Keepers by Total Dismissals (Stumping, Catch, Run Out) - IPL 2025

```
# Step 1: Filter dismissal types for wicketkeepers
dismissal_types = ['stumped', 'caught', 'run out']
wk_dismissals = df[df['wicket_type'].isin(dismissal_types) &
df['fielder'].notna()]
# Step 2: Count total dismissals per fielder (keeper)
dismissals_count =
wk_dismissals['fielder'].value_counts().reset_index()
dismissals_count.columns = ['wicket_keeper', 'total_dismissals']
# Step 3: Get top 10 wicket-keepers
top_wicket_keepers = dismissals_count.head(10)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=top_wicket_keepers, x='total_dismissals',
y='wicket_keeper', palette='PuRd')
for bar in ax.patches:
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Top 10 Wicket-Keepers by Total Dismissals (Stumping, Catch,
Run Out) - IPL 2025')
plt.xlabel('Total Dismissals')
plt.ylabel('Wicket-Keeper')
plt.tight_layout()
plt.show()
```



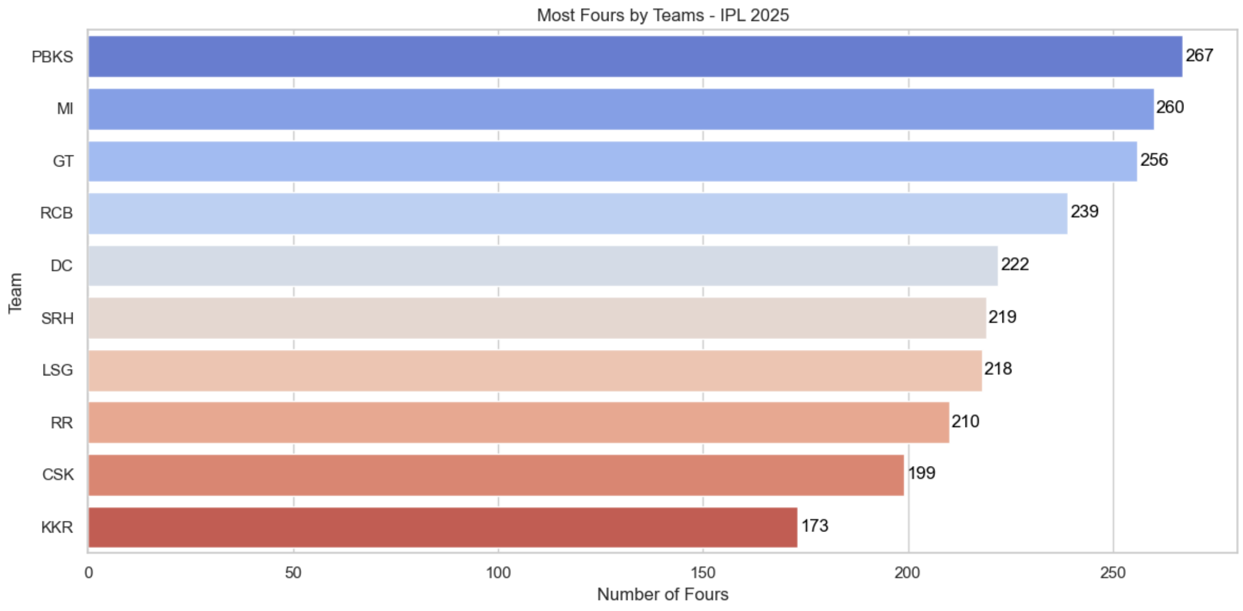
Total Runs Scored by Teams - IPL 2025

```
# Step 1: Sum runs by batting team
team_runs = df.groupby('batting_team')
['runs_of_bat'].sum().reset_index()
# Step 2: Sort descending by runs
team_runs = team_runs.sort_values(by='runs_of_bat', ascending=False)
# Step 3: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=team_runs, x='runs_of_bat', y='batting_team',
palette='magma')
# Add run count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 50,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Total Runs Scored by Teams - IPL 2025')
plt.xlabel('Total Runs')
plt.ylabel('Team')
plt.tight_layout()
plt.show()
```



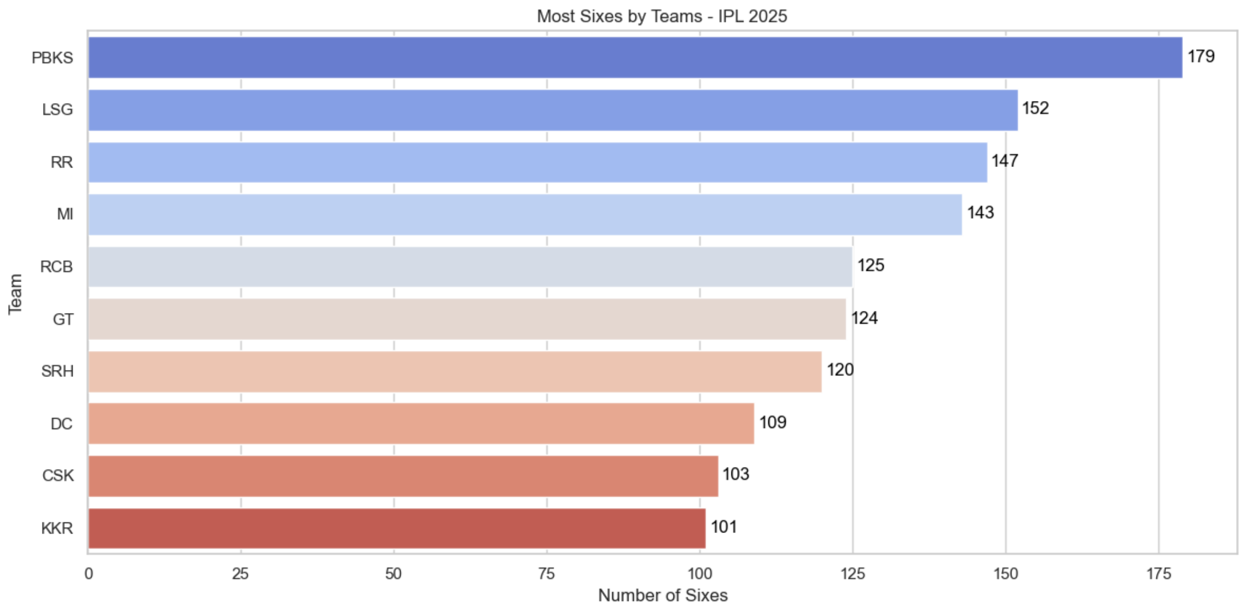
Most Fours by Teams - IPL 2025

```
# Step 1: Filter deliveries where runs_of_bat is 4 (fours)
fours = df[df['runs_of_bat'] == 4]
# Step 2: Count fours by batting team
team_fours =
fours.groupby('batting_team').size().reset_index(name='fours_count')
# Step 3: Sort descending
team_fours = team_fours.sort_values(by='fours_count', ascending=False)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=team_fours, x='fours_count', y='batting_team',
palette='coolwarm')
# Add count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Most Fours by Teams - IPL 2025')
plt.xlabel('Number of Fours')
plt.ylabel('Team')
plt.tight_layout()
plt.show()
```



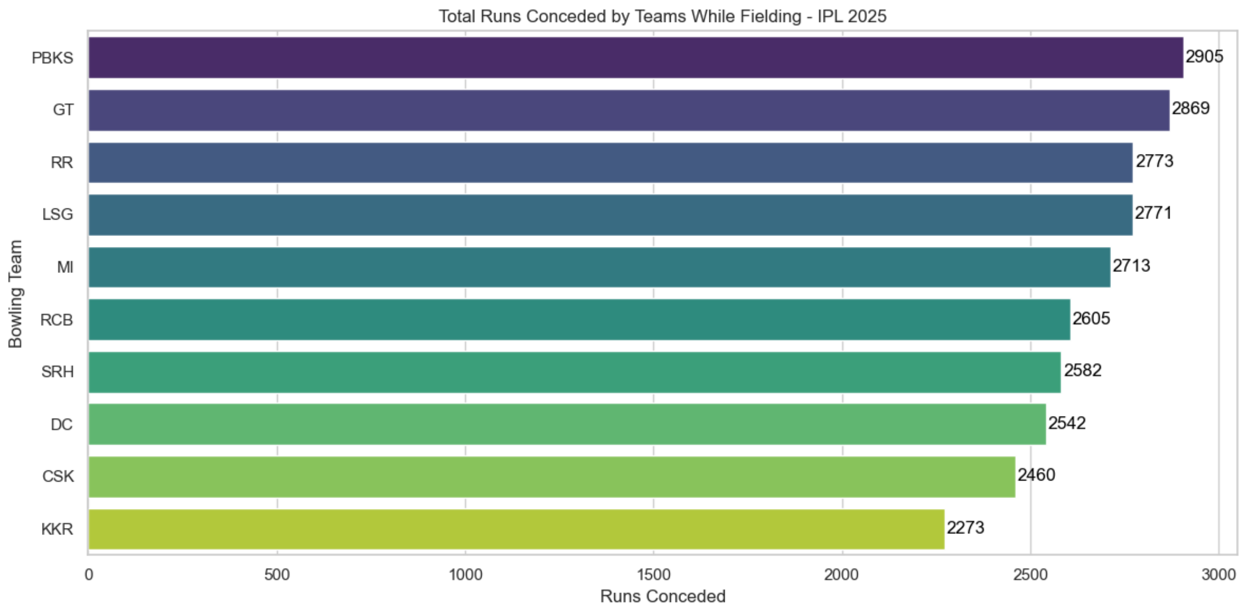
Most Sixes by Teams - IPL 2025

```
# Step 1: Filter deliveries where runs_of_bat is 6 (sixes)
sixes = df[df['runs_of_bat'] == 6]
# Step 2: Count sixes by batting team
team_sixes =
sixes.groupby('batting_team').size().reset_index(name='sixes_count')
# Step 3: Sort descending
team_sixes = team_sixes.sort_values(by='sixes_count', ascending=False)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=team_sixes, x='sixes_count', y='batting_team',
palette='coolwarm')
# Add count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Most Sixes by Teams - IPL 2025')
plt.xlabel('Number of Sixes')
plt.ylabel('Team')
plt.tight_layout()
plt.show()
```



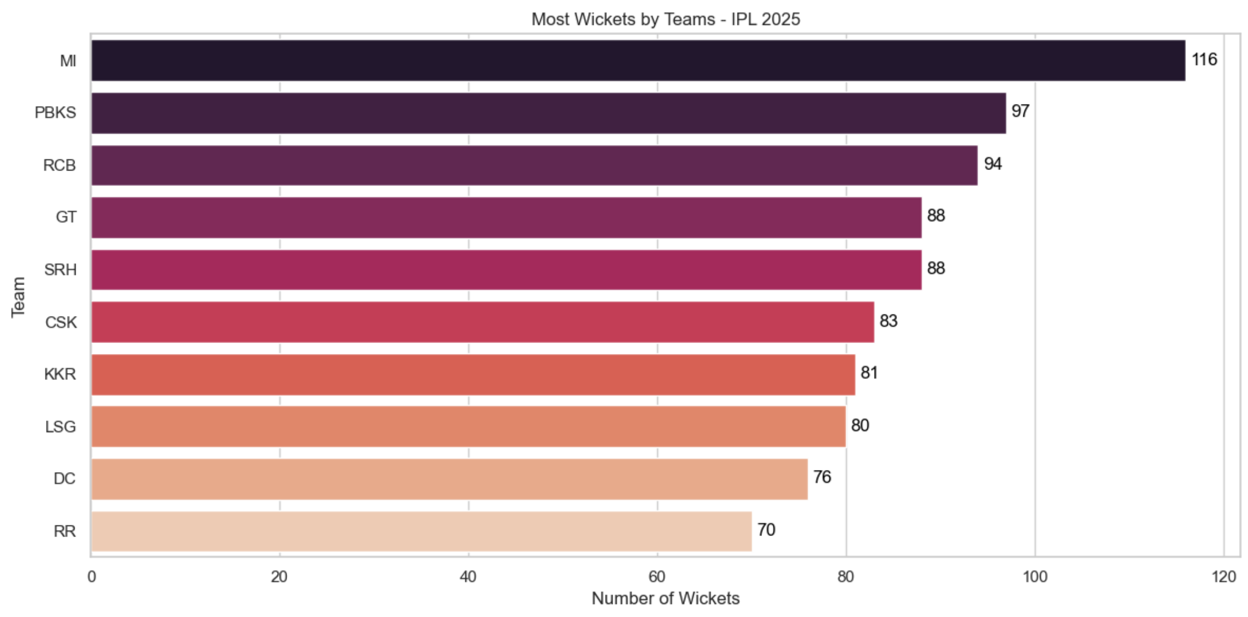
Total Runs Conceded by Teams While Fielding - IPL 2025

```
# Step 1: Calculate total runs per delivery
df['total_runs'] = df['runs_of_bat'] + df['extras']
# Step 2: Group by bowling_team and sum total_runs conceded
runs_conceded_by_team = df.groupby('bowling_team')
['total_runs'].sum().reset_index()
# Step 3: Sort descending by runs conceded
runs_conceded_by_team =
runs_conceded_by_team.sort_values(by='total_runs', ascending=False)
# Step 4: Plot
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=runs_conceded_by_team, x='total_runs',
y='bowling_team', palette='viridis')
# Add labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 5,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Total Runs Conceded by Teams While Fielding - IPL 2025')
plt.xlabel('Runs Conceded')
plt.ylabel('Bowling Team')
plt.tight_layout()
plt.show()
```



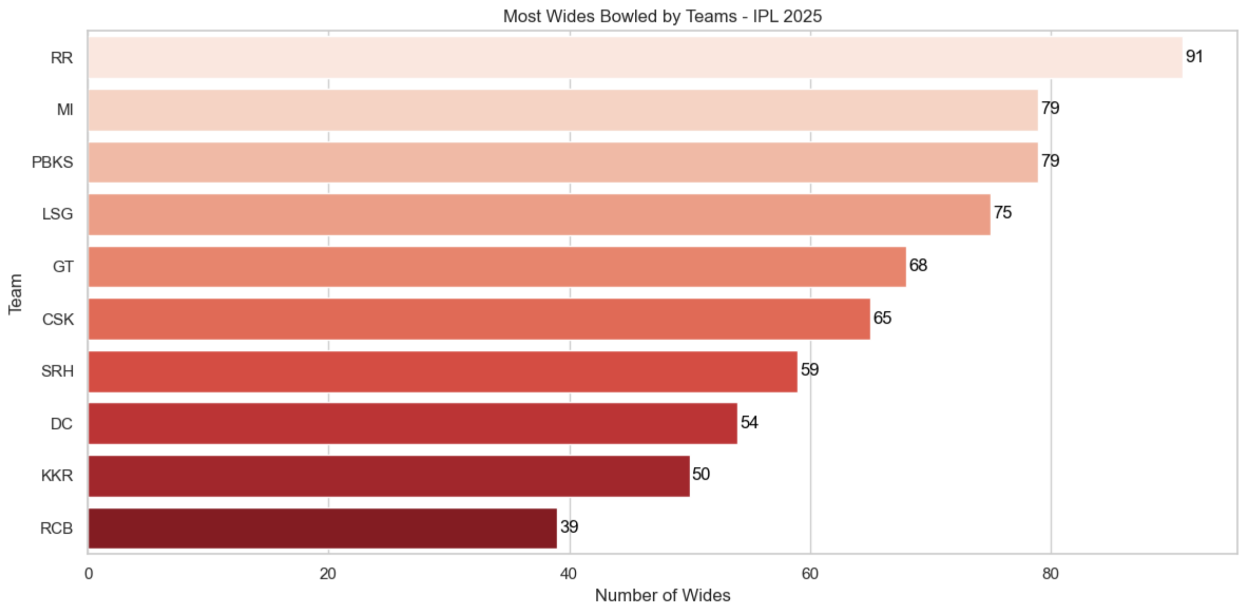
Most Wickets by Teams - IPL 2025

```
# Step 1: Filter rows where a wicket fell (player_dismissed is not
null)
wickets = df[df['player_dismissed'].notna()]
# Step 2: Count wickets by bowling team
team_wickets =
wickets.groupby('bowling_team').size().reset_index(name='wickets')
# Step 3: Sort descending
team_wickets = team_wickets.sort_values(by='wickets', ascending=False)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=team_wickets, x='wickets', y='bowling_team',
palette='rocket')
# Add wicket count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Most Wickets by Teams - IPL 2025')
plt.xlabel('Number of Wickets')
plt.ylabel('Team')
plt.tight_layout()
plt.show()
```



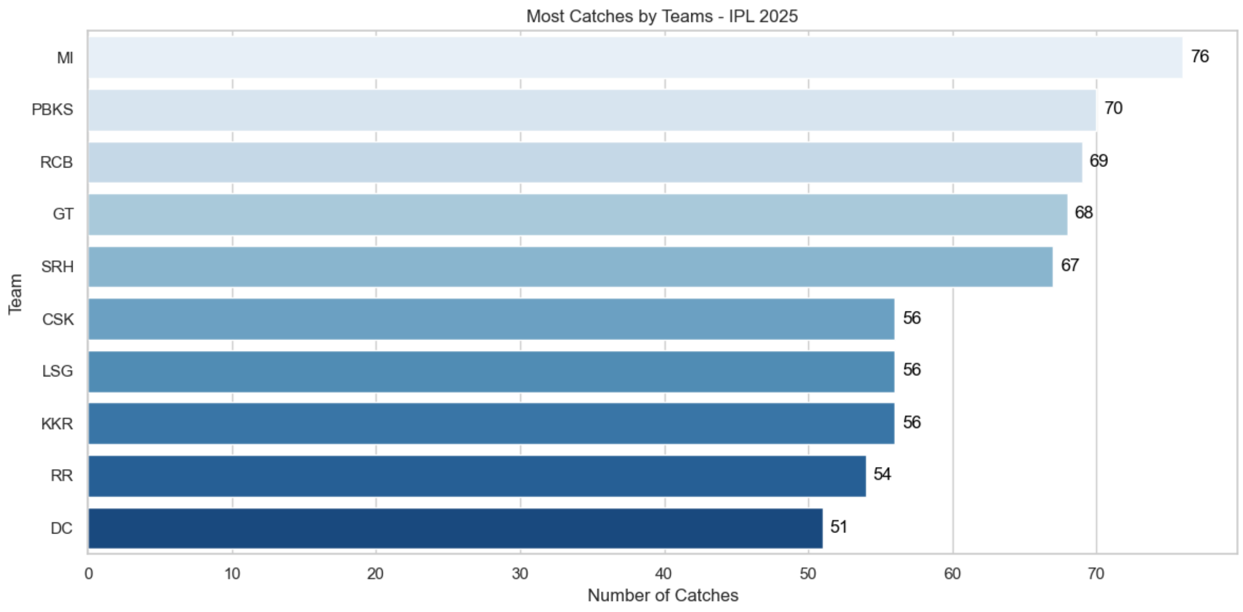
Most Wides Bowled by Teams - IPL 2025

```
# Step 1: Sum wides by bowling team
# wides column indicates number of wides per delivery (usually 1 or 0)
team_wides = df.groupby('bowling_team')['wide'].sum().reset_index()
# Step 2: Sort descending
team_wides = team_wides.sort_values(by='wide', ascending=False)
# Step 3: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=team_wides, x='wide', y='bowling_team',
palette='Reds')
# Add wides count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.2,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Most Wides Bowled by Teams - IPL 2025')
plt.xlabel('Number of Wides')
plt.ylabel('Team')
plt.tight_layout()
plt.show()
```



Most Catches by Teams - IPL 2025

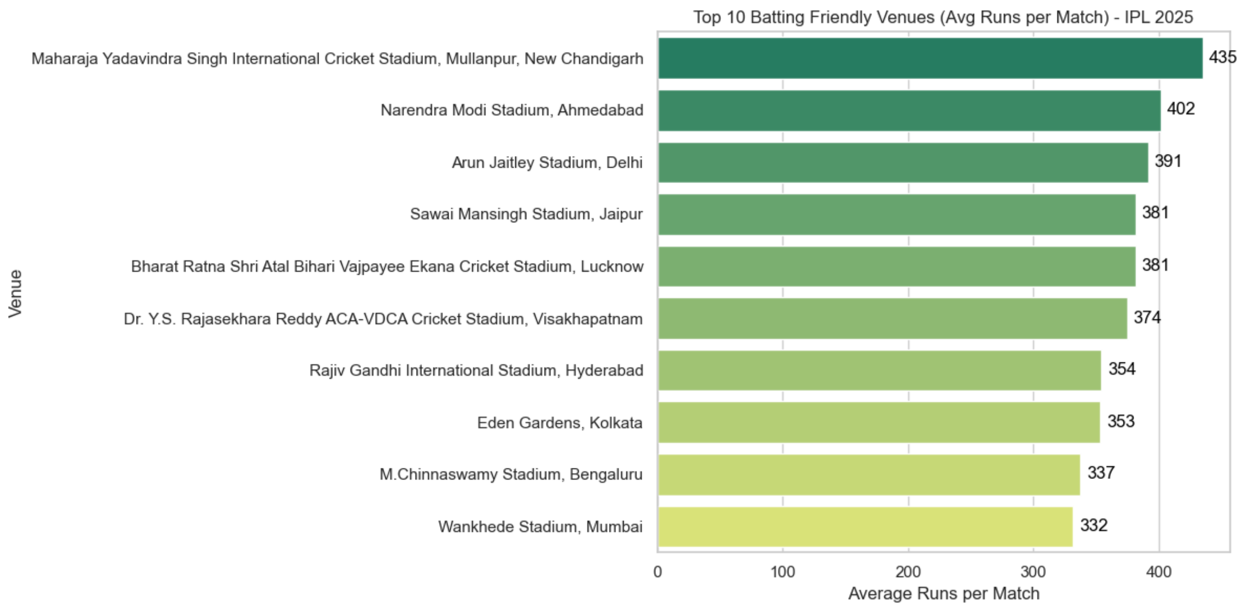
```
# Step 1: Filter rows where wicket_type is 'caught' and fielder is not
null
caught_deliveries = df[(df['wicket_type'] == 'caught') &
(df['fielder'].notna())]
# Step 2: Count catches by bowling team
team_catches =
caught_deliveries.groupby('bowling_team').size().reset_index(name='cat
ches')
# Step 3: Sort descending
team_catches = team_catches.sort_values(by='catches', ascending=False)
# Step 4: Plotting
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=team_catches, x='catches', y='bowling_team',
palette='Blues')
# Add catch count labels on bars
for bar in ax.patches:
    plt.text(bar.get_width() + 0.5,
             bar.get_y() + bar.get_height() / 2,
             int(bar.get_width()),
             va='center', fontsize=12, color='black')
plt.title('Most Catches by Teams - IPL 2025')
plt.xlabel('Number of Catches')
plt.ylabel('Team')
plt.tight_layout()
plt.show()
```



Top 10 Batting Friendly Venues (Avg Runs per Match) - IPL 2025

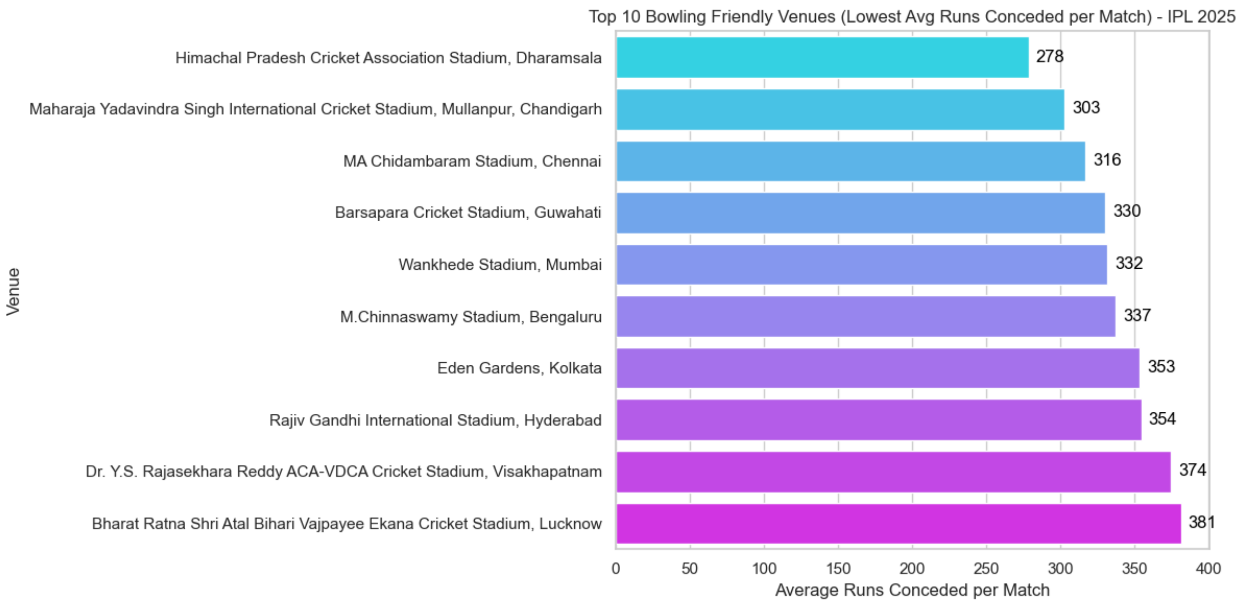
```
# Step 1: Calculate total runs per delivery
df['total_runs'] = df['runs_of_bat'] + df['extras']
# Step 2: Total runs scored per venue
runs_per_venue = df.groupby('venue')['total_runs'].sum().reset_index()
# Step 3: Number of matches per venue
matches_per_venue = df.groupby('venue')
['match_id'].nunique().reset_index(name='matches')
# Step 4: Merge runs and matches data
venue_stats = runs_per_venue.merge(matches_per_venue, on='venue')
# Step 5: Calculate average runs per match at each venue
venue_stats['avg_runs_per_match'] = venue_stats['total_runs'] /
venue_stats['matches']
# Step 6: Sort by average runs descending
venue_stats = venue_stats.sort_values(by='avg_runs_per_match',
ascending=False)
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=venue_stats.head(10), x='avg_runs_per_match',
y='venue', palette='summer')
for bar in ax.patches:
    plt.text(bar.get_width() + 5,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.0f}",
             va='center', fontsize=12, color='black')
plt.title('Top 10 Batting Friendly Venues (Avg Runs per Match) - IPL
2025')
```

```
plt.xlabel('Average Runs per Match')
plt.ylabel('Venue')
plt.tight_layout()
plt.show()
```



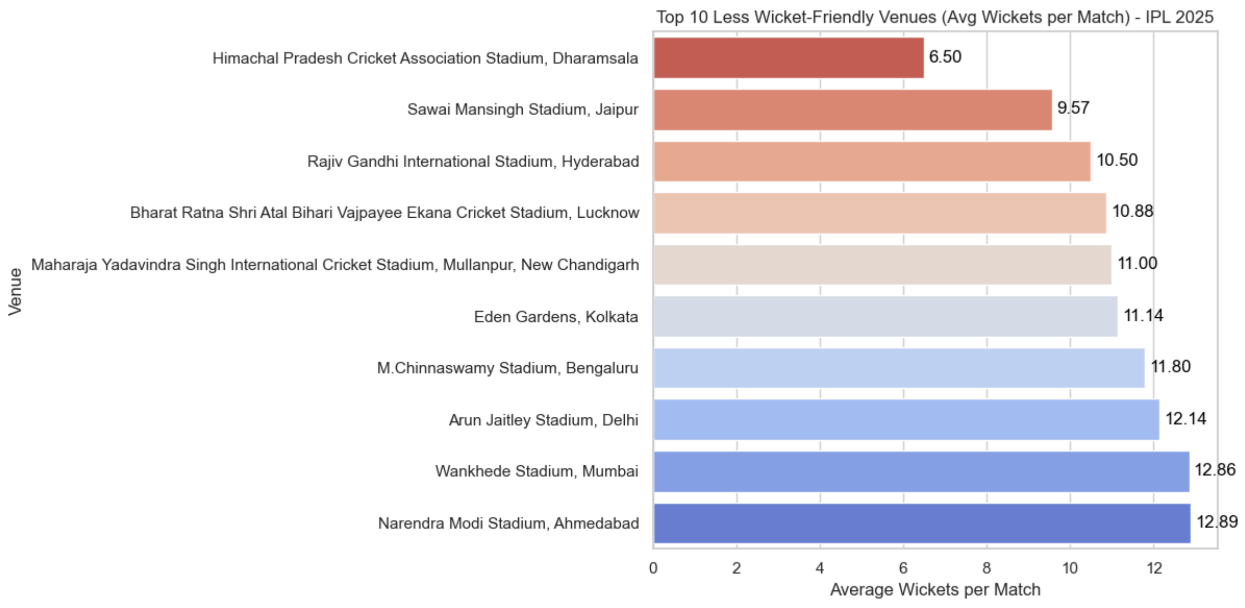
Top 10 Bowling Friendly Venues (Lowest Avg Runs Conceded per Match) - IPL 2025

```
# Step 1: Calculate total runs per delivery
df['total_runs'] = df['runs_of_bat'] + df['extras']
# Step 2: Total runs conceded per venue (same as total runs scored)
runs_per_venue = df.groupby('venue')['total_runs'].sum().reset_index()
# Step 3: Number of matches per venue
matches_per_venue = df.groupby('venue')
['match_id'].nunique().reset_index(name='matches')
# Step 4: Merge runs and matches data
venue_stats = runs_per_venue.merge(matches_per_venue, on='venue')
# Step 5: Calculate average runs conceded per match at each venue
venue_stats['avg_runs_conceded_per_match'] = venue_stats['total_runs']
/ venue_stats['matches']
# Step 6: Sort by average runs conceded ascending (lower is better
bowling wicket)
venue_stats =
venue_stats.sort_values(by='avg_runs_conceded_per_match')
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=venue_stats.head(10),
x='avg_runs_conceded_per_match', y='venue', palette='cool')
for bar in ax.patches:
    plt.text(bar.get_width() + 5,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.0f}",
             va='center', fontsize=12, color='black')
plt.title('Top 10 Bowling Friendly Venues (Lowest Avg Runs Conceded
per Match) - IPL 2025')
plt.xlabel('Average Runs Conceded per Match')
plt.ylabel('Venue')
plt.tight_layout()
plt.show()
```



Top 10 Less Wicket-Friendly Venues (Avg Wickets per Match) - IPL 2025

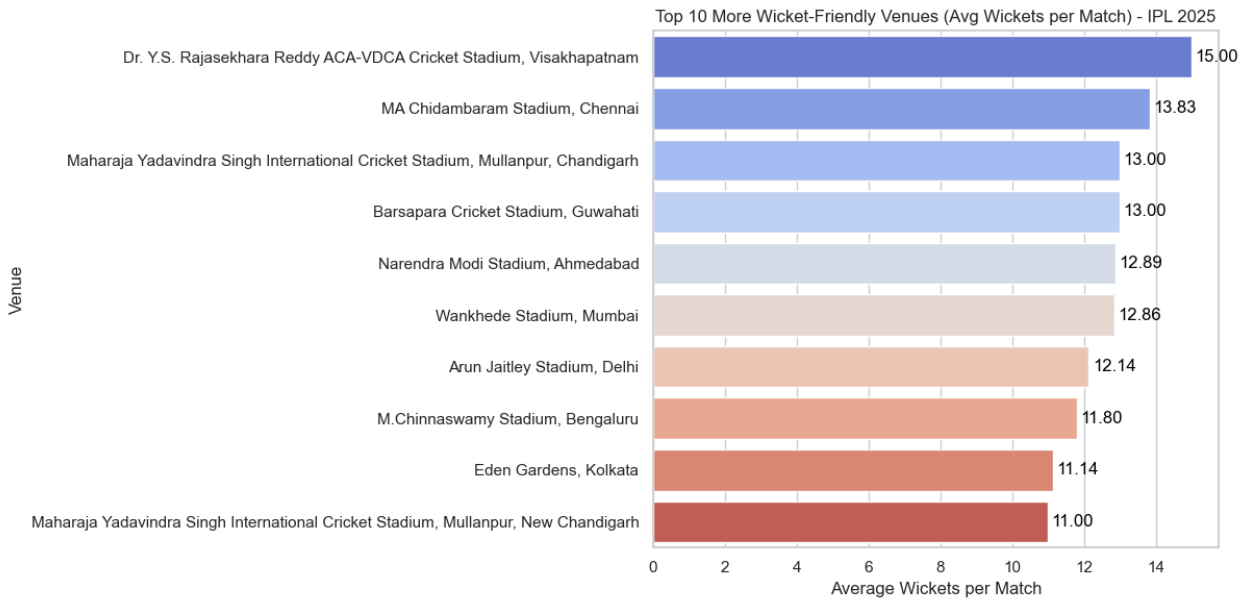
```
# Step 1: Filter deliveries where wickets fell (wicket_type not null
or empty)
wickets = df[df['wicket_type'].notna() & (df['wicket_type'] != '')]
# Step 2: Count wickets per venue
wickets_per_venue =
wickets.groupby('venue').size().reset_index(name='wickets')
# Step 3: Count matches per venue
matches_per_venue = df.groupby('venue')
['match_id'].nunique().reset_index(name='matches')
# Step 4: Merge wickets and matches data
venue_wicket_stats = wickets_per_venue.merge(matches_per_venue,
on='venue')
# Step 5: Calculate average wickets per match at each venue
venue_wicket_stats['avg_wickets_per_match'] =
venue_wicket_stats['wickets'] / venue_wicket_stats['matches']
# Step 6: Sort by average wickets ascending (least wickets first)
venue_wicket_stats =
venue_wicket_stats.sort_values(by='avg_wickets_per_match',
ascending=True)
# Step 7: Plot top 10 venues with least wickets per match (less
wicket-friendly)
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=venue_wicket_stats.head(10),
x='avg_wickets_per_match', y='venue', palette='coolwarm_r')
for bar in ax.patches:
    plt.text(bar.get_width() + 0.1,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.2f}",
             va='center', fontsize=12, color='black')
plt.title('Top 10 Less Wicket-Friendly Venues (Avg Wickets per Match)
- IPL 2025')
plt.xlabel('Average Wickets per Match')
plt.ylabel('Venue')
plt.tight_layout()
plt.show()
```



Top 10 More Wicket-Friendly Venues (Avg Wickets per Match) - IPL 2025

```
# Step 1: Filter deliveries where wickets fell (wicket_type not null
or empty)
wickets = df[df['wicket_type'].notna() & (df['wicket_type'] != '')]
```

```python
# Step 2: Count wickets per venue
wickets_per_venue =
wickets.groupby('venue').size().reset_index(name='wickets')
# Step 3: Count matches per venue
matches_per_venue = df.groupby('venue')
['match_id'].nunique().reset_index(name='matches')
# Step 4: Merge wickets and matches data
venue_wicket_stats = wickets_per_venue.merge(matches_per_venue,
on='venue')
# Step 5: Calculate average wickets per match at each venue
venue_wicket_stats['avg_wickets_per_match'] =
venue_wicket_stats['wickets'] / venue_wicket_stats['matches']
# Step 6: Sort by average wickets descending (more wickets first)
venue_wicket_stats =
venue_wicket_stats.sort_values(by='avg_wickets_per_match',
ascending=False)
# Step 7: Plot top 10 venues with most wickets per match (more wicket-
friendly)
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=venue_wicket_stats.head(10),
x='avg_wickets_per_match', y='venue', palette='coolwarm')
for bar in ax.patches:
    plt.text(bar.get_width() + 0.1,
             bar.get_y() + bar.get_height() / 2,
             f"{bar.get_width():.2f}",
             va='center', fontsize=12, color='black')
plt.title('Top 10 More Wicket-Friendly Venues (Avg Wickets per Match)
- IPL 2025')
plt.xlabel('Average Wickets per Match')
plt.ylabel('Venue')
plt.tight_layout()
plt.show()
```


Top 10 More Wicket-Friendly Venues (Avg Wickets per Match) - IPL 2025
Venue
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium, Visakhapatnam
MA Chidambaram Stadium, Chennai
Maharaja Yadavindra Singh International Cricket Stadium, Mullanpur, Chandigarh
Barsapara Cricket Stadium, Guwahati
Narendra Modi Stadium, Ahmedabad
Wankhede Stadium, Mumbai
Arun Jaitley Stadium, Delhi
M.Chinnaswamy Stadium, Bengaluru
Eden Gardens, Kolkata
Maharaja Yadavindra Singh International Cricket Stadium, Mullanpur, New Chandigarh
15.00
13.83
13.00
13.00
12.89
12.86
12.14
11.80
11.14
11.00
0
2
4
6
8
10
12
14
Average Wickets per Match