In [1]:
```
# ============================================================================
# 1. Setup and Data Loading
# ============================================================================

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

# Set a consistent style for plots
plt.style.use('fivethirtyeight')
sns.set_style('whitegrid')

# Load the datasets
try:
    df_ball_by_ball = pd.read_csv('IPL_BallByBall2008_2024(Updated).csv')
    df_team_performance = pd.read_csv('team_performance_dataset_2008to2024.csv')
    df_players = pd.read_csv('Players_Info_2024.csv')
    df_teams = pd.read_csv('ipl_teams_2024_info.csv')
    print("All datasets loaded successfully.")
except FileNotFoundError as e:
    print(f"Error: {e}. Please ensure the CSV files are in the correct directory
```

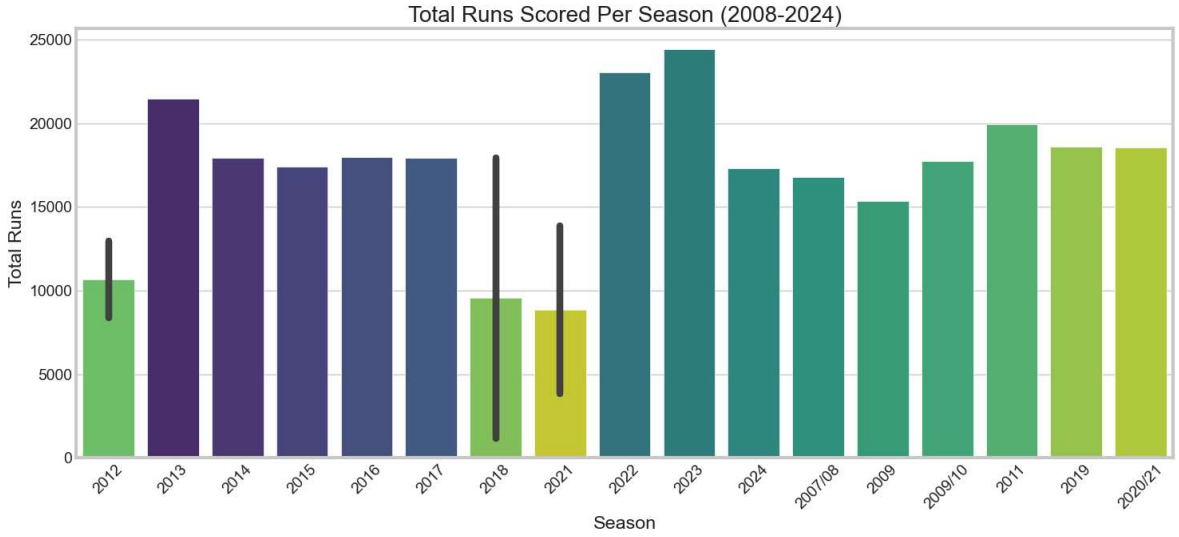
```
All datasets loaded successfully.
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\3513067361.py:17: DtypeWarning:
Columns (2) have mixed types. Specify dtype option on import or set low_memory=Fa
lse.
  df_ball_by_ball = pd.read_csv('IPL_BallByBall2008_2024(Updated).csv')
```

In [2]:
```
# Initial data cleaning and preparation
# Renaming columns for easier use
df_ball_by_ball = df_ball_by_ball.rename(columns={
    'Match id': 'match_id', 'Ball No': 'ball_no', 'runs_scored': 'runs_scored',
    'Striker': 'striker', 'Bowler': 'bowler', 'Season': 'season',
    'Batting team': 'batting_team', 'Bowling team': 'bowling_team',
    'Innings No': 'innings_no', 'extras': 'extras'
})

df_team_performance = df_team_performance.rename(columns={
    'Match_ID': 'match_id', 'Date': 'date', 'Venue': 'venue',
    'First_Innings_Score': 'first_innings_score',
    'Second_Innings_Score': 'second_innings_score', 'Match_Winner': 'match_winne
})

# Convert 'Date' column to datetime objects
df_team_performance['date'] = pd.to_datetime(df_team_performance['date'], dayfir

# Merge datasets for combined analysis
# Merging ball-by-ball data with match details
df_combined = pd.merge(df_ball_by_ball, df_team_performance[['match_id', 'venue'
df_combined['year'] = df_combined['date'].dt.year
```

```
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\2201781317.py:17: UserWarning:
Parsing dates in %Y-%m-%d format when dayfirst=True was specified. Pass `dayfirst
=False` or specify a format to silence this warning.
  df_team_performance['date'] = pd.to_datetime(df_team_performance['date'], dayfi
rst=True)
```

In [3]:
```
# 2. Analyze run trends over the years
# ============================================================================

print("\n--- 2. Analysis of Run Trends Over the Years ---")
# Group by season and calculate total runs scored
season_runs = df_combined.groupby('season')['runs_scored'].sum()
season_runs_df = season_runs.reset_index()
season_runs_df.columns = ['season', 'total_runs']

# Visualize run trends
plt.figure(figsize=(15, 7))
sns.barplot(x='season', y='total_runs', data=season_runs_df, palette='viridis')
plt.title('Total Runs Scored Per Season (2008-2024)')
plt.xlabel('Season')
plt.ylabel('Total Runs')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

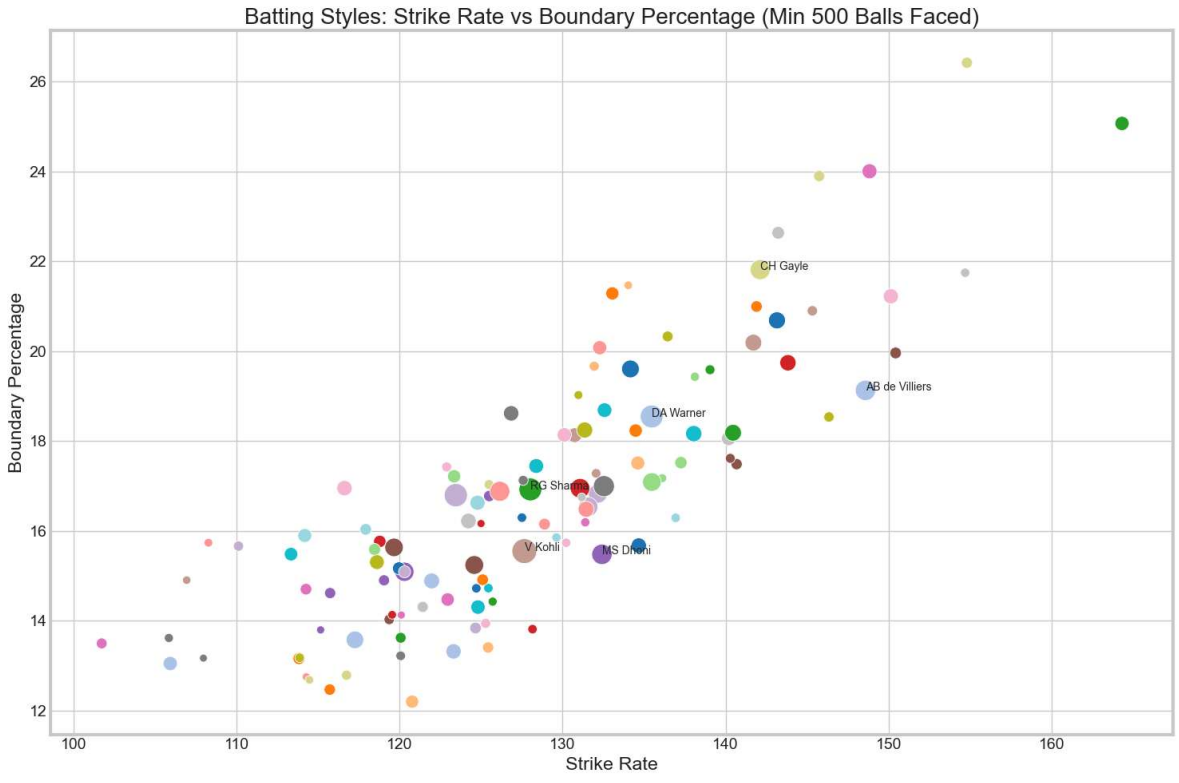
```
--- 2. Analysis of Run Trends Over the Years ---
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\1295995484.py:12: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x='season', y='total_runs', data=season_runs_df, palette='viridis')
```



In [4]:
```
# ============================================================================
# 3. Compare batting styles (anchor vs aggressive)
# ============================================================================

print("\n--- 3. Comparison of Batting Styles ---")
# Define metrics for anchor vs aggressive
# We'll filter for players who have faced at least 500 balls to ensure a reasona
min_balls_faced = 500

# Calculate total runs and balls faced for each striker
striker_stats = df_ball_by_ball.groupby('striker').agg(
    total_runs=('runs_scored', 'sum'),
    balls_faced=('ball_no', 'count'),
    total_boundaries=('runs_scored', lambda x: (x == 4).sum() + (x == 6).sum())
```

```
).reset_index()

# Filter for players who have faced a minimum number of balls
striker_stats = striker_stats[striker_stats['balls_faced'] >= min_balls_faced]

# Calculate strike rate and boundary percentage
striker_stats['strike_rate'] = (striker_stats['total_runs'] / striker_stats['bal
striker_stats['boundary_percent'] = (striker_stats['total_boundaries'] / striker

# Visualize the comparison
plt.figure(figsize=(15, 10))
sns.scatterplot(x='strike_rate', y='boundary_percent', data=striker_stats, hue='
plt.title('Batting Styles: Strike Rate vs Boundary Percentage (Min 500 Balls Fac
plt.xlabel('Strike Rate')
plt.ylabel('Boundary Percentage')

# Add annotations for some prominent players
prominent_players = ['V Kohli', 'MS Dhoni', 'RG Sharma', 'AB de Villiers', 'CH G
for player in prominent_players:
    if player in striker_stats['striker'].values:
        player_data = striker_stats[striker_stats['striker'] == player].iloc[0]
        plt.text(player_data['strike_rate'], player_data['boundary_percent'], pl

plt.tight_layout()
plt.show()
```

```
--- 3. Comparison of Batting Styles ---
```



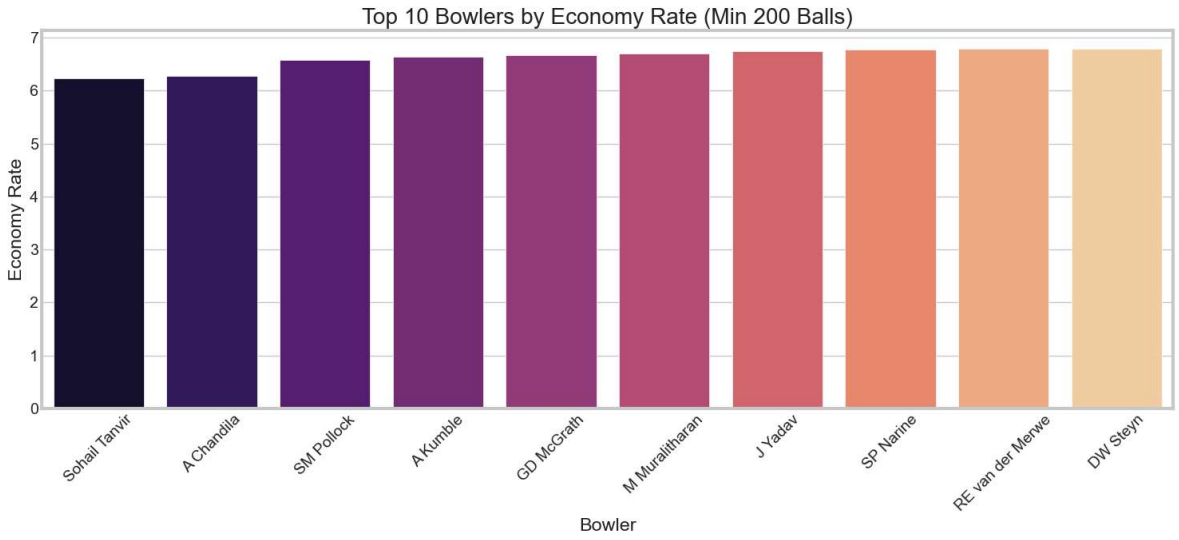
In [5]:

```
# ================================================================================
# 4. Study bowling consistency
# ================================================================================

print("\n--- 4. Study of Bowling Consistency ---")
# Calculate key bowling metrics
bowler_stats = df_ball_by_ball.groupby('bowler').agg(
    total_balls_bowled=('ball_no', 'count'),
    total_runs_conceded=('runs_scored', 'sum'),
```

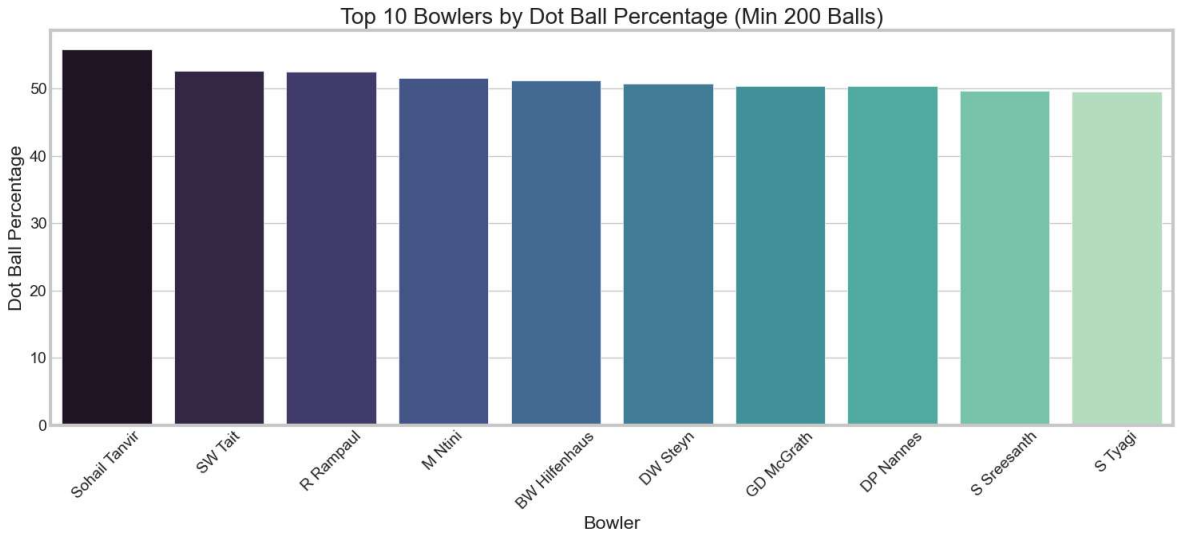
```
    dot_balls=('runs_scored', lambda x: (x == 0).sum()),
    extras_conceded=('extras', 'sum')
).reset_index()

# Add wickets taken, need to count wickets from the main dataset
wickets_taken = df_ball_by_ball[df_ball_by_ball['wicket_confirmation'] == 1].gro
bowler_stats = pd.merge(bowler_stats, wickets_taken, on='bowler', how='left').fi

# Filter for bowlers with a minimum of 200 balls bowled
min_balls_bowled = 200
bowler_stats = bowler_stats[bowler_stats['total_balls_bowled'] >= min_balls_bowl

# Calculate Economy Rate, Dot Ball Percentage, and Bowling Average
bowler_stats['economy_rate'] = (bowler_stats['total_runs_conceded'] + bowler_sta
bowler_stats['dot_ball_percent'] = (bowler_stats['dot_balls'] / bowler_stats['to
bowler_stats['bowling_average'] = np.where(bowler_stats['wickets'] == 0, np.nan,

# Visualize top bowlers by economy rate
top_bowlers_eco = bowler_stats.sort_values('economy_rate').head(10)
plt.figure(figsize=(15, 7))
sns.barplot(x='bowler', y='economy_rate', data=top_bowlers_eco, palette='magma')
plt.title('Top 10 Bowlers by Economy Rate (Min 200 Balls)')
plt.xlabel('Bowler')
plt.ylabel('Economy Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Visualize top bowlers by dot ball percentage
top_bowlers_dots = bowler_stats.sort_values('dot_ball_percent', ascending=False)
plt.figure(figsize=(15, 7))
sns.barplot(x='bowler', y='dot_ball_percent', data=top_bowlers_dots, palette='ma
plt.title('Top 10 Bowlers by Dot Ball Percentage (Min 200 Balls)')
plt.xlabel('Bowler')
plt.ylabel('Dot Ball Percentage')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
--- 4. Study of Bowling Consistency ---
```

```
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\381974974.py:30: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x='bowler', y='economy_rate', data=top_bowlers_eco, palette='magm
a')
```




```
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\381974974.py:41: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x='bowler', y='dot_ball_percent', data=top_bowlers_dots, palette='m
ako')
```



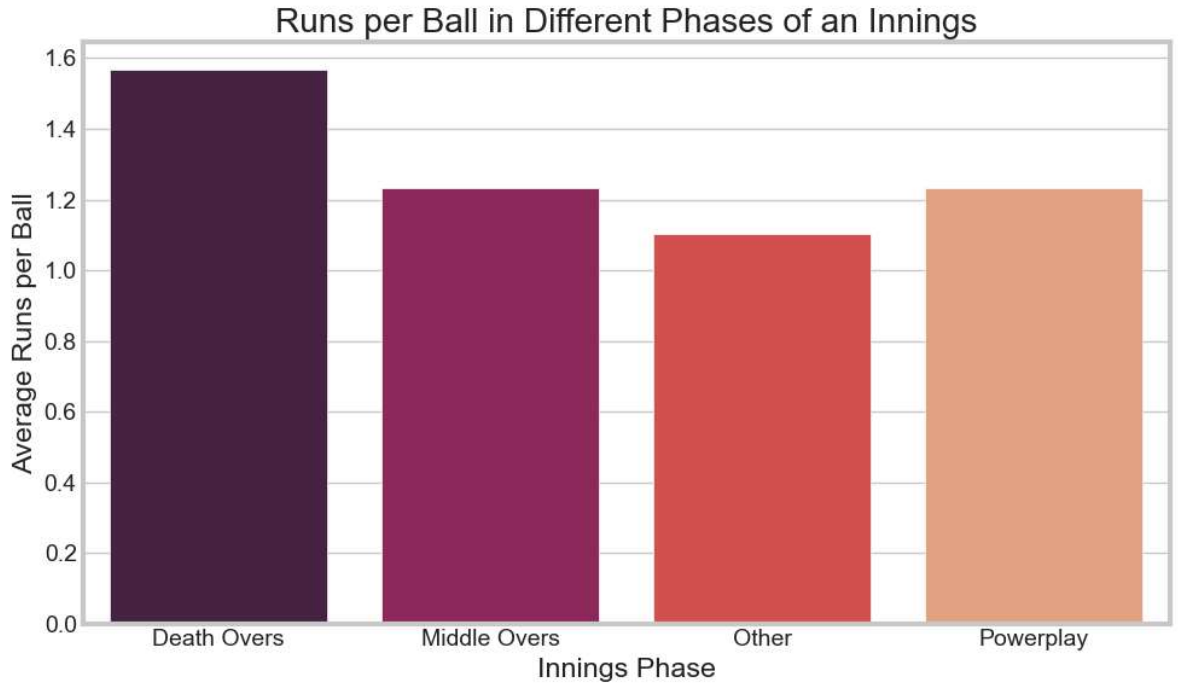
In [6]:

```
# ============================================================================
# 5. Visualize performance in different overs (Powerplay, Death)
# ============================================================================

print("\n--- 5. Performance in Different Overs (Powerplay vs Death) ---")
# Create a phase column (Powerplay, Middle, Death)
def get_phase(ball_no):
    if 1 <= ball_no <= 6:
        return 'Powerplay'
    elif 7 <= ball_no <= 16:
        return 'Middle Overs'
    elif 17 <= ball_no <= 20:
        return 'Death Overs'
    return 'Other'

df_ball_by_ball['phase'] = df_ball_by_ball['ball_no'].apply(get_phase)

# Calculate runs and wickets per phase
```

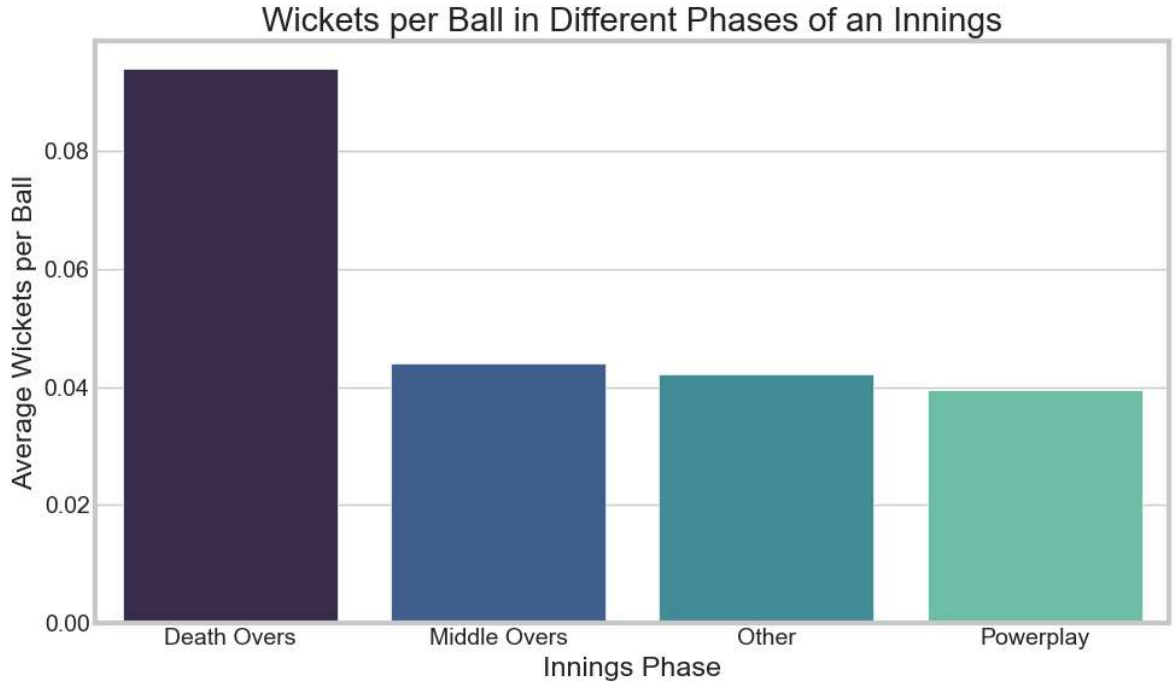
```
phase_performance = df_ball_by_ball.groupby('phase').agg(
    total_runs=('runs_scored', 'sum'),
    total_wickets=('wicket_confirmation', 'sum'),
    total_balls=('ball_no', 'count')
).reset_index()

# Calculate runs per ball and wickets per ball
phase_performance['runs_per_ball'] = phase_performance['total_runs'] / phase_per
phase_performance['wickets_per_ball'] = phase_performance['total_wickets'] / pha

# Visualize runs per ball in each phase
plt.figure(figsize=(10, 6))
sns.barplot(x='phase', y='runs_per_ball', data=phase_performance, palette='rocke
plt.title('Runs per Ball in Different Phases of an Innings')
plt.xlabel('Innings Phase')
plt.ylabel('Average Runs per Ball')
plt.tight_layout()
plt.show()

# Visualize wickets per ball in each phase
plt.figure(figsize=(10, 6))
sns.barplot(x='phase', y='wickets_per_ball', data=phase_performance, palette='ma
plt.title('Wickets per Ball in Different Phases of an Innings')
plt.xlabel('Innings Phase')
plt.ylabel('Average Wickets per Ball')
plt.tight_layout()
plt.show()
```

```
--- 5. Performance in Different Overs (Powerplay vs Death) ---
```

```
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\4012147532.py:31: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x='phase', y='runs_per_ball', data=phase_performance, palette='rock
et')
```



```
C:\Users\wasim\AppData\Local\Temp\ipykernel_29168\4012147532.py:40: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x='phase', y='wickets_per_ball', data=phase_performance, palette='m
ako')
```



In [7]:
```python
# ============================================================================
# 6. Compare venue behavior in high-scoring vs low-scoring matches
# ============================================================================

print("\n--- 6. Venue Behavior in High-Scoring vs Low-Scoring Matches ---")
# Define thresholds for high and low scoring matches
high_score_threshold = 180
low_score_threshold = 140

# Calculate total runs per match
match_scores = df_ball_by_ball.groupby('match_id').agg(
    total_runs=('runs_scored', 'sum')
).reset_index()

# Merge with match details to get venue
match_scores_with_venue = pd.merge(match_scores, df_team_performance[['match_id'

# Categorize matches as high or low scoring
match_scores_with_venue['score_category'] = np.select(
    [match_scores_with_venue['total_runs'] >= high_score_threshold, match_scores
    ['High Scoring', 'Low Scoring'],
    default='Medium Scoring'
)

# Count the number of high and low scoring matches per venue
venue_behavior = match_scores_with_venue.groupby(['venue', 'score_category']).si

# Filter for venues with a significant number of matches
venue_behavior['total_matches'] = venue_behavior.sum(axis=1)
```

```
venue_behavior = venue_behavior[venue_behavior['total_matches'] > 20] # Filter v
venue_behavior = venue_behavior.sort_values('total_matches', ascending=False)

# Visualize venue behavior
venue_behavior[['High Scoring', 'Low Scoring']].plot(kind='bar', figsize=(15, 8)
plt.title('High vs. Low Scoring Matches by Venue (Min 20 Matches)')
plt.xlabel('Venue')
plt.ylabel('Number of Matches')
plt.xticks(rotation=90)
plt.legend(title='Score Category')
plt.tight_layout()
plt.show()
```

```
--- 6. Venue Behavior in High-Scoring vs Low-Scoring Matches ---
```