In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
matches= pd.read_csv('C:/Users/saurabh/matches.csv')
deliveries= pd.read_csv('C:/Users/saurabh/deliveries.csv')
```

In [3]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

DATASETS

In [4]:
```
matches.head()
```

Out[4]:

	season	team1	team2	date	match_number	venue	city	toss_winner	toss_decisio
0	2024	Canada	United States of America	01/06/2024	1	Grand Prairie Stadium	Dallas	United States of America	fiel
1	2024	Papua New Guinea	West Indies	02/06/2024	2	Providence Stadium	Providence	West Indies	fiel
2	2024	Oman	Namibia	02/06/2024	3	Kensington Oval	Bridgetown	Namibia	fiel
3	2024	Sri Lanka	South Africa	03/06/2024	4	Nassau County International Cricket Stadium	New York	Sri Lanka	ba
4	2024	Afghanistan	Uganda	03/06/2024	5	Providence Stadium	Providence	Uganda	fiel

In [5]:
```
deliveries.head()
```

Out[5]:

match_id	season	start_date	venue	innings	ball	batting_team	bowling_team	striker	non_striker	bo
0	1	2024	2024-06-02	Providence Stadium, Guyana	1	0.1	Papua New Guinea	West Indies	TP Ura	A Vala	H
1	1	2024	2024-06-02	Providence Stadium, Guyana	1	0.2	Papua New Guinea	West Indies	TP Ura	A Vala	H
2	1	2024	2024-06-02	Providence Stadium, Guyana	1	0.3	Papua New Guinea	West Indies	A Vala	TP Ura	H
3	1	2024	2024-06-02	Providence Stadium, Guyana	1	0.4	Papua New Guinea	West Indies	A Vala	TP Ura	H
4	1	2024	2024-06-02	Providence Stadium, Guyana	1	0.5	Papua New Guinea	West Indies	A Vala	TP Ura	H

FILLING SUPER OVER WINNERS

In [6]:
```
matches.loc[(matches['team1'] == 'Oman') & (matches['team2'] == 'Namibia'), 'winner'] =
matches.loc[(matches['team1'] == 'Pakistan') & (matches['team2'] == 'United States of Am
```

1. MATCHES ANALYSIS

1.1 Which team won the most matches?

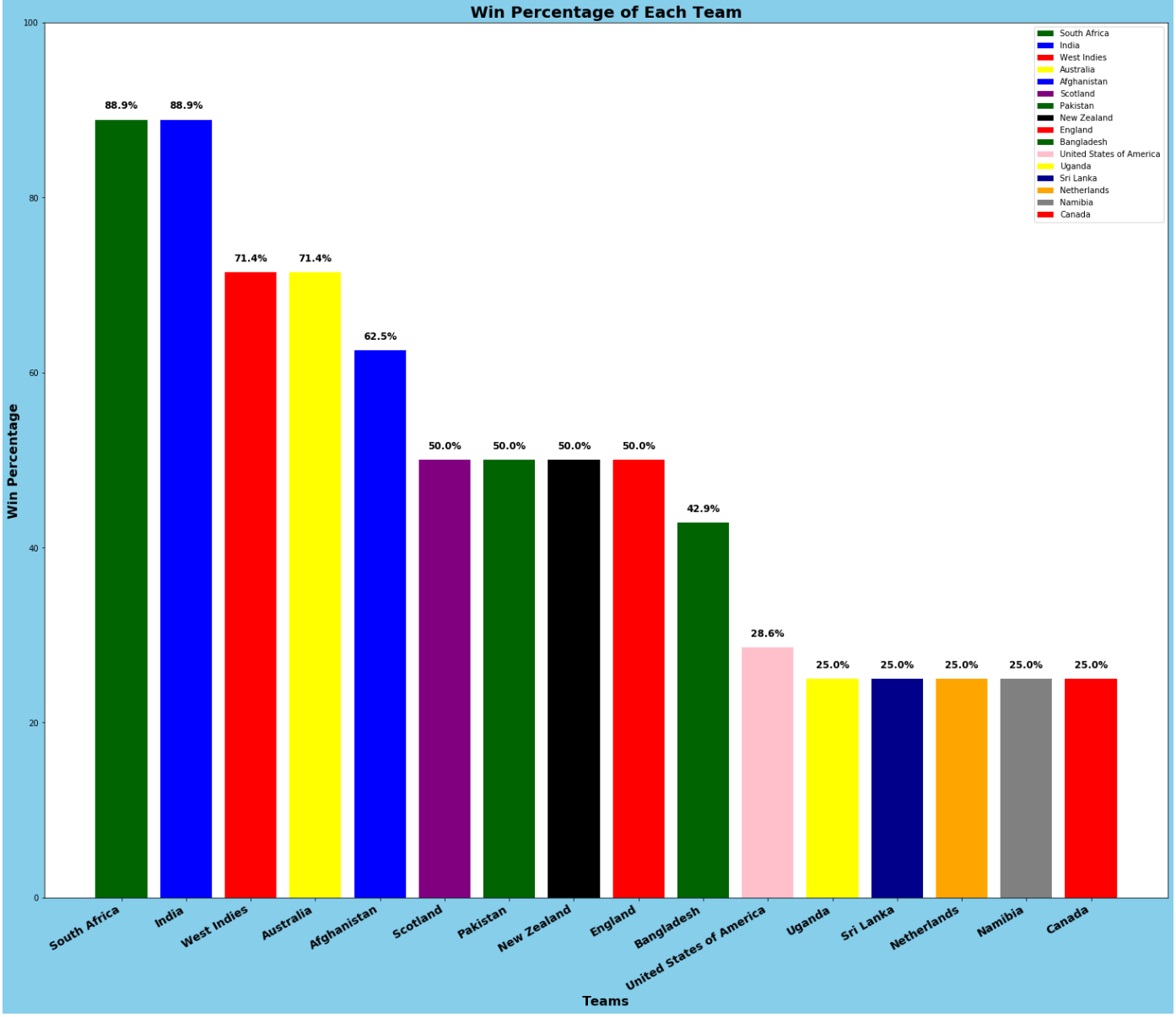
In [7]:
```
win_counts = matches['winner'].value_counts()

max_wins = win_counts.max()
teams_with_most_wins = win_counts[win_counts == max_wins].index.tolist()

teams_str = ", ".join(teams_with_most_wins)
print(f"Teams with the most wins: {teams_str} ({max_wins} wins)")
```

```
Teams with the most wins: South Africa, India (8 wins)
```

1.2 What is the win percentage of each team?

In [8]:
```
all_teams = pd.concat([matches['team1'], matches['team2']])

total_matches = all_teams.value_counts()

team_wins = matches['winner'].value_counts()

win_percentage = (team_wins / total_matches) * 100

win_percentage_sorted = win_percentage.sort_values(ascending=False)

win_percentage_sorted = win_percentage_sorted[win_percentage_sorted > 0]


nation_colors = {
    'India': 'blue',
```

```
    'Afghanistan': 'blue',
    'Uganda': 'yellow',
    'Australia': 'yellow',
    'New Zealand': 'black',
    'South Africa': 'darkgreen',
    'Pakistan': 'darkgreen',
    'Bangladesh': 'darkgreen',
    'West Indies': 'red',
    'Scotland': 'purple',
    'Netherlands': 'orange',
    'United States of America': 'pink',
    'Sri Lanka': 'darkblue',
    'Canada': 'red',
    'England': 'red'
}
colors = [nation_colors.get(team, 'gray') for team in win_percentage_sorted.index]

plt.figure(figsize=(25, 20), facecolor='skyblue')
bars=plt.bar(win_percentage_sorted.index, win_percentage_sorted, color=colors)
plt.xlabel('Teams', fontsize=16, weight='bold')
plt.ylabel('Win Percentage', fontsize=16, weight='bold')
labels=[f'{team}' for team in win_percentage_sorted.index]
plt.legend(bars,labels)
plt.title('Win Percentage of Each Team', fontsize=20, weight='bold')
plt.xticks(rotation=30, ha='right', fontsize=14, weight='bold')
plt.ylim(0, 100)

for bar, percentage in zip(bars, win_percentage_sorted):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 1, f'{percentage:.1f}'
             ha='center', va='bottom', fontsize=12, weight='bold')
```



1.3 How does the toss outcome affect the match result?

In [9]:
```
toss_match_outcome = matches[matches['toss_winner'] == matches['winner']]

toss_win_and_match_win_count = toss_match_outcome.shape[0]
total_matches_count = matches.shape[0]

print(f"Toss winner also won the match {toss_win_and_match_win_count} times out of {tota
```

```
Toss winner also won the match 28 times out of 55 matches.
```

1.4 How often do teams win after winning the toss and choosing to bat or bowl?

In [10]:
```
chosen_to_bat = matches[(matches['toss_decision'] == 'bat') & (matches['toss_winner'] ==
chosen_to_field = matches[(matches['toss_decision'] == 'field') & (matches['toss_winner'

batting_win_percentage = (chosen_to_bat.shape[0] / matches.shape[0]) * 100
fielding_win_percentage = (chosen_to_field.shape[0] / matches.shape[0]) * 100
```

```
print(f"Percentage of wins after choosing to bat: {batting_win_percentage:.2f}%")
print(f"Percentage of wins after choosing to field: {fielding_win_percentage:.2f}%")
```

```
Percentage of wins after choosing to bat: 9.09%
Percentage of wins after choosing to field: 41.82%
```

1.5 How does team performance vary between batting first and chasing targets?

In [11]:
```
batting_first = matches[matches['toss_decision'] == 'bat']
chasing_target = matches[matches['toss_decision'] == 'field']

batting_first_win_percentage = (batting_first[batting_first['winner'] == batting_first['
chasing_win_percentage = (chasing_target[chasing_target['winner'] == chasing_target['tea

print(f"Win percentage batting first: {batting_first_win_percentage:.2f}%")
print(f"Win percentage chasing target: {chasing_win_percentage:.2f}%")
```

```
Win percentage batting first: 50.00%
Win percentage chasing target: 54.76%
```

1.6 What was the average number of sixes and fours per match?

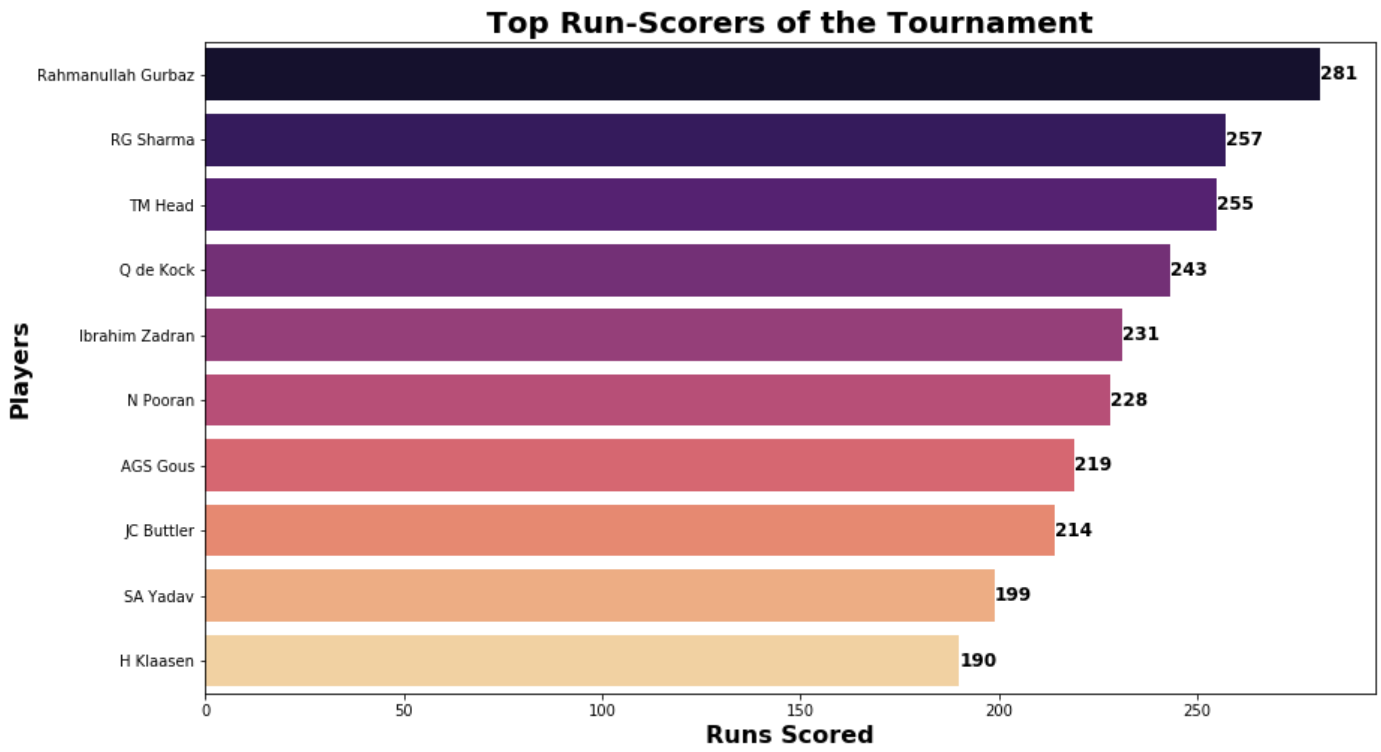
In [12]:
```
fours=deliveries[deliveries['runs_off_bat']==4]
sixes=deliveries[deliveries['runs_off_bat']==6]
avg_fours_per_match= fours.shape[0]/matches.shape[0]
avg_sixes_per_match=sixes.shape[0]/matches.shape[0]

print(f"Average Number of fours per match: {avg_fours_per_match: .0f}")
print(f"Average Number of sixes per match: {avg_sixes_per_match: .0f}")
```

```
Average Number of fours per match:  18
Average Number of sixes per match:  9
```

2 Player Performance Analysis

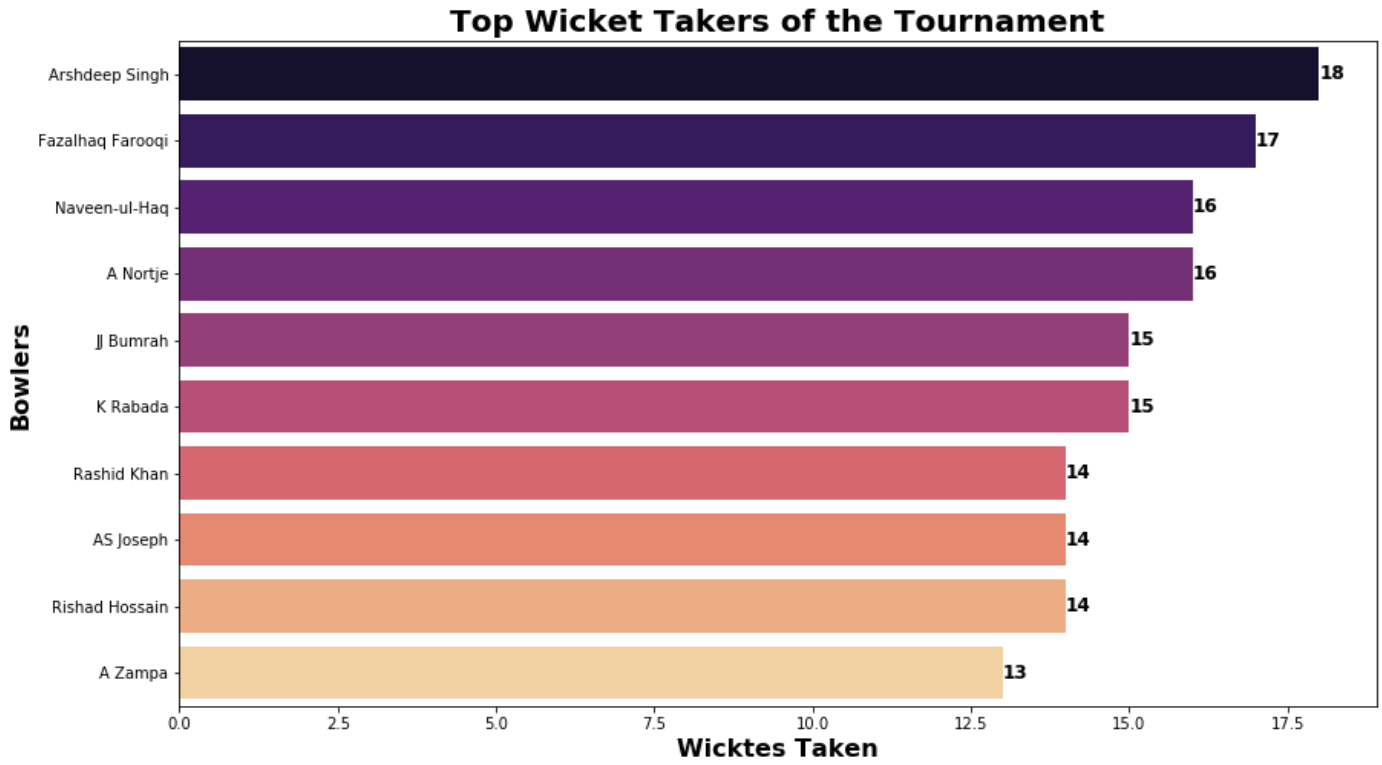
2.1 Top Run Scorers of the Tournament

In [13]:
```
top_run_scorers = deliveries.groupby('striker')['runs_off_bat'].sum().sort_values(ascend
plt.figure(figsize=(14, 8))
barplot = sns.barplot(x=top_run_scorers.values, y=top_run_scorers.index, palette='magma'
plt.title('Top Run-Scorers of the Tournament', fontsize=20, weight='bold')
plt.xlabel('Runs Scored', fontsize=16, weight='bold')
plt.ylabel('Players', fontsize=16, weight='bold')
for index, value in enumerate(top_run_scorers.values):
    plt.text(value, index, f'{value}', va='center', ha='left', fontsize=12, color='black
```



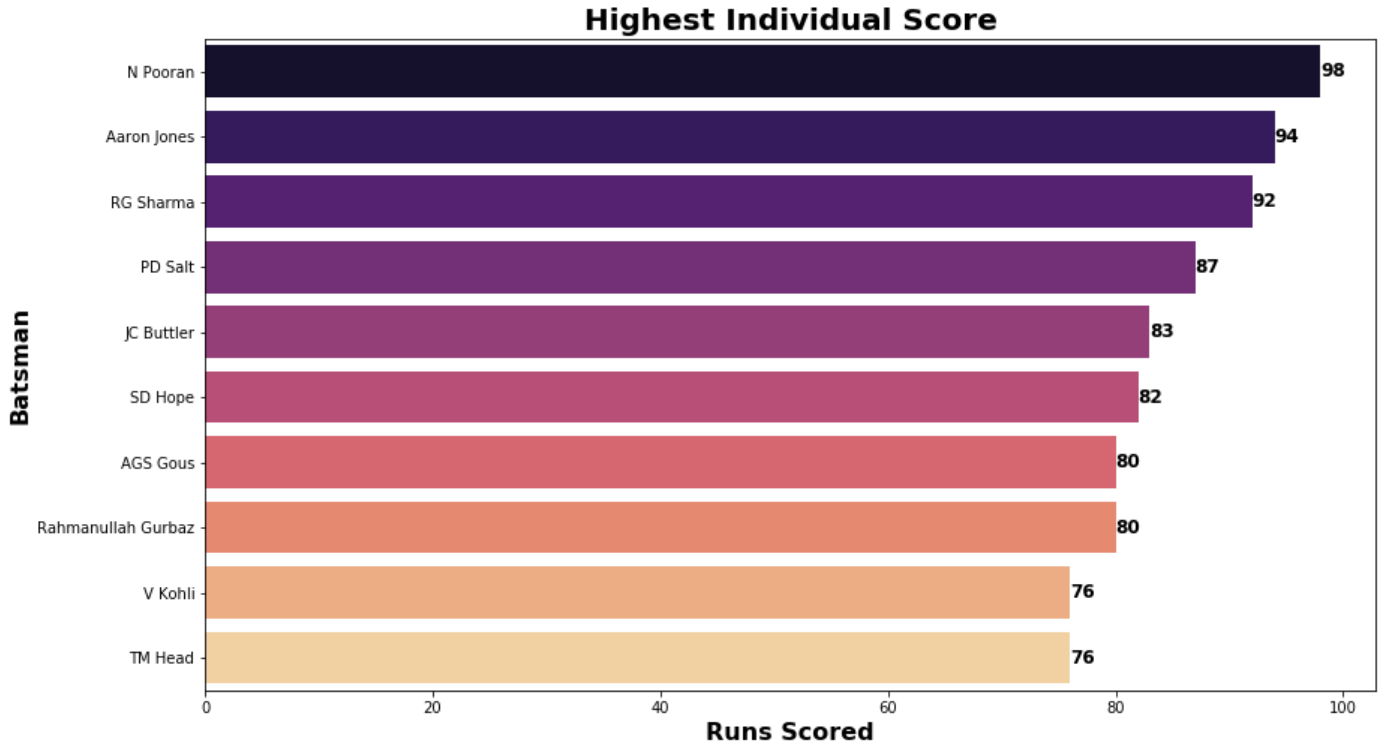
2.2 Top Wicket-Takers of the Tournament

In [14]:
```
top_wicket_takers = deliveries[deliveries['wicket_type'].notnull()].groupby('bowler').si
plt.figure(figsize=(14, 8))
barplot = sns.barplot(x=top_wicket_takers.values, y=top_wicket_takers.index, palette='ma
plt.title('Top Wicket Takers of the Tournament', fontsize=20, weight='bold')
plt.xlabel('Wicktes Taken', fontsize=16, weight='bold')
plt.ylabel('Bowlers', fontsize=16, weight='bold')
for index, value in enumerate(top_wicket_takers.values):
    plt.text(value, index, f'{value}', va='center', ha='left', fontsize=12, color='black
```



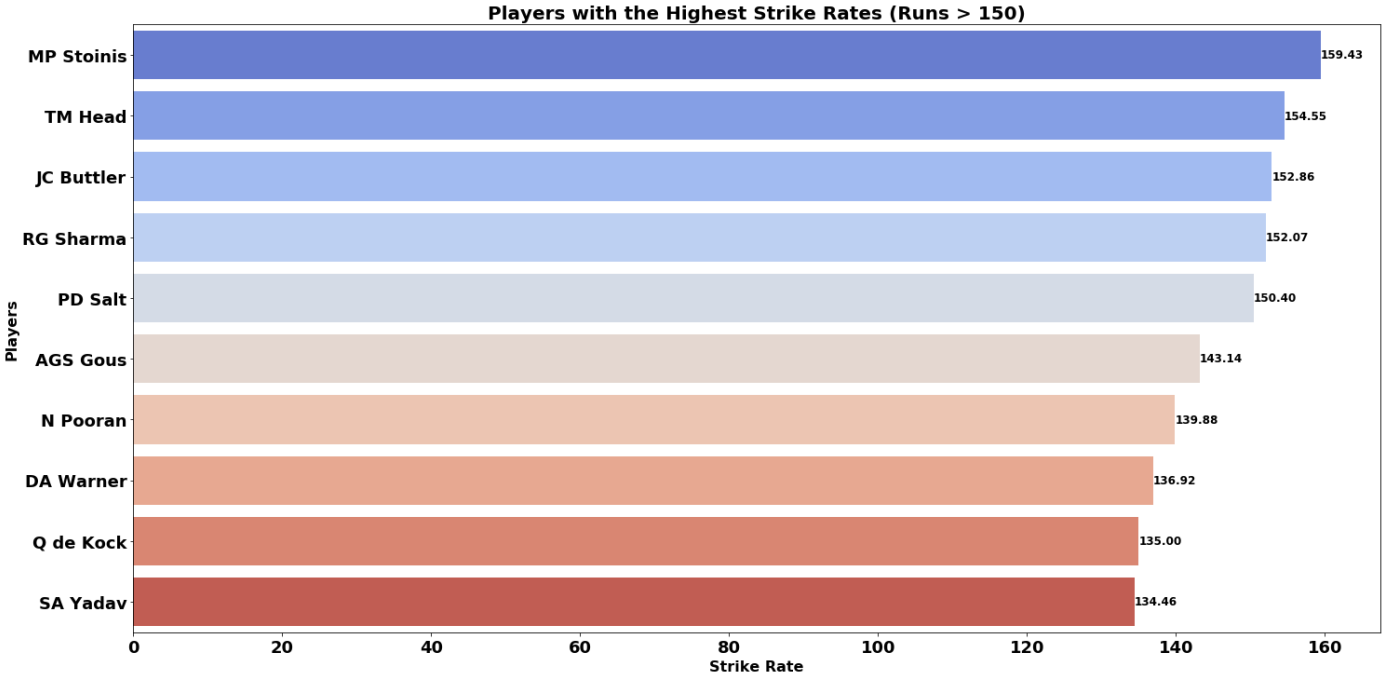
2.3 Top 10 highest individual player scores

In [15]:
```
top_individual_scorer=deliveries.groupby(['match_id','striker'])['runs_off_bat'].sum().r
plt.figure(figsize=(14, 8))

bars=sns.barplot(x=top_individual_scorer['runs_off_bat'],y=top_individual_scorer['strike
plt.title('Highest Individual Score', fontsize=20, weight='bold')
plt.xlabel('Runs Scored', fontsize=16, weight='bold')
plt.ylabel('Batsman', fontsize=16, weight='bold')
for index, (value, name) in enumerate(zip(top_individual_scorer['runs_off_bat'], top_ind
    bars.text(value, index, f'{value}', va='center', ha='left', fontsize=12, color='blac
```



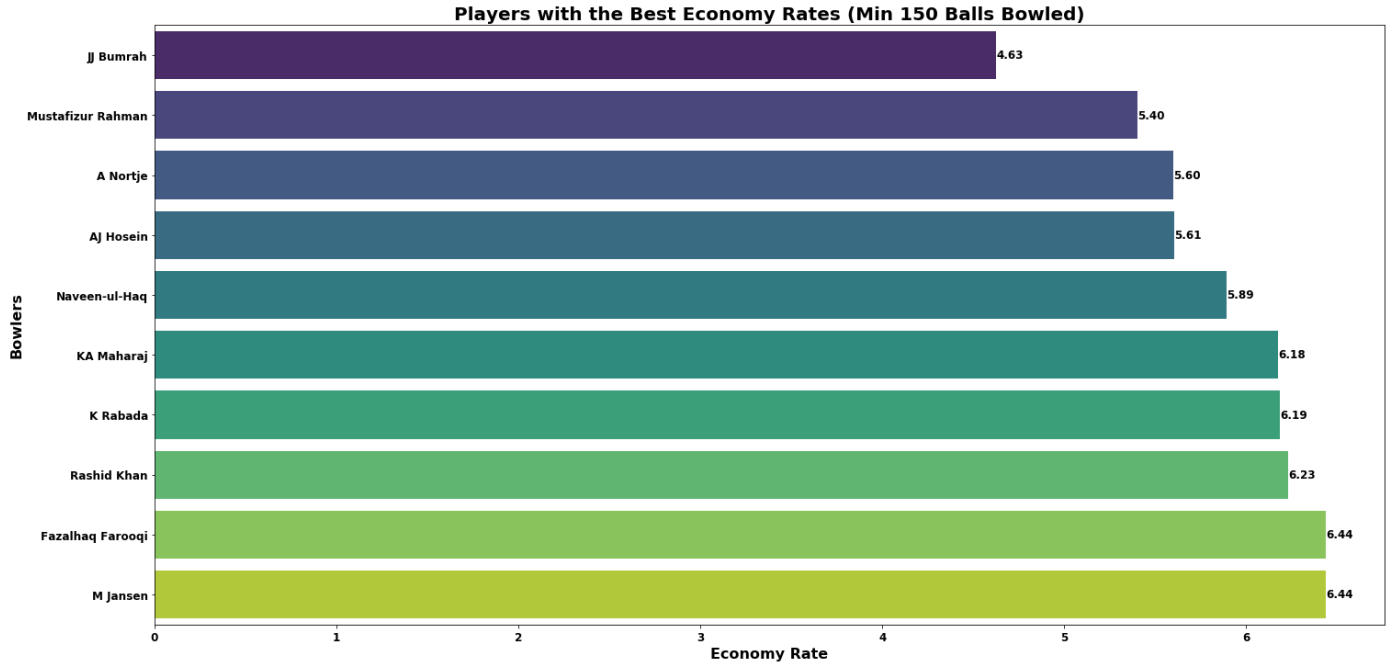
2.4 Players with the Highest Strike Rates(Runs > 150)

In [16]:
```
balls_faced = deliveries.groupby('striker').size()
runs_scored = deliveries.groupby('striker')['runs_off_bat'].sum()

strike_rate = (runs_scored / balls_faced) * 100

filtered_strike_rate = strike_rate[runs_scored > 150]

top_strike_rates = filtered_strike_rate.sort_values(ascending=False).head(10)

plt.figure(figsize=(24, 12))
barplot = sns.barplot(x=top_strike_rates.values, y=top_strike_rates.index, palette='cool
plt.title('Players with the Highest Strike Rates (Runs > 150)', fontsize=20, weight='bol
plt.xlabel('Strike Rate', fontsize=16, weight='bold')
plt.ylabel('Players', fontsize=16, weight='bold')

for index, value in enumerate(top_strike_rates.values):
    plt.text(value, index, f'{value:.2f}', va='center', ha='left', fontsize=12, color='b

plt.xticks(fontsize=18, weight='bold')
plt.yticks(fontsize=18, weight='bold')
plt.show()
```



2.5 Players with the Best Economy Rates(Minimum 150 Balls)

In [17]:
```
balls_bowled = deliveries.groupby('bowler').size()
runs_conceded = deliveries.groupby('bowler')['runs_off_bat'].sum() + deliveries.groupby(
economy_rate = (runs_conceded / (balls_bowled / 6))

filtered_economy_rate = economy_rate[balls_bowled >= 150]

best_economy_rates = filtered_economy_rate.sort_values().head(10)

plt.figure(figsize=(24, 12))

barplot = sns.barplot(x=best_economy_rates.values, y=best_economy_rates.index, palette='
plt.title('Players with the Best Economy Rates (Min 150 Balls Bowled)', fontsize=20, wei
plt.xlabel('Economy Rate', fontsize=16, weight='bold')
plt.ylabel('Bowlers', fontsize=16, weight='bold')

for index, value in enumerate(best_economy_rates.values):
    plt.text(value, index, f'{value:.2f}', va='center', ha='left', fontsize=12, color='b

plt.xticks(fontsize=12, weight='bold')
plt.yticks(fontsize=12, weight='bold')

plt.show()
```



2.6 Consistent Batters

In [18]:
```
consistent_batsmen = deliveries.groupby(['match_id', 'striker'])['runs_off_bat'].sum().g

consistent_batsmen = consistent_batsmen.reset_index().rename(columns={"striker": "Batsma
consistent_batsmen
```

Out[18]:

	Batsman	Striking Rate
0	BJ McMullen	46.666667
1	AGS Gous	36.500000
2	TM Head	36.428571
3	HC Brook	36.250000
4	SD Hope	35.666667
5	Rahmanullah Gurbaz	35.125000
6	KR Mayers	35.000000
7	RD Berrington	34.000000
8	MP Stoinis	33.800000
9	NR Kirton	33.666667

2.7 Consistent Bowlers

In [19]:
```
consistent_bowlers = deliveries[deliveries['wicket_type'].notnull()].groupby(['match_id'
consistent_bowlers = consistent_bowlers.reset_index().rename(columns={"bowler": "Bowler"
consistent_bowlers
```

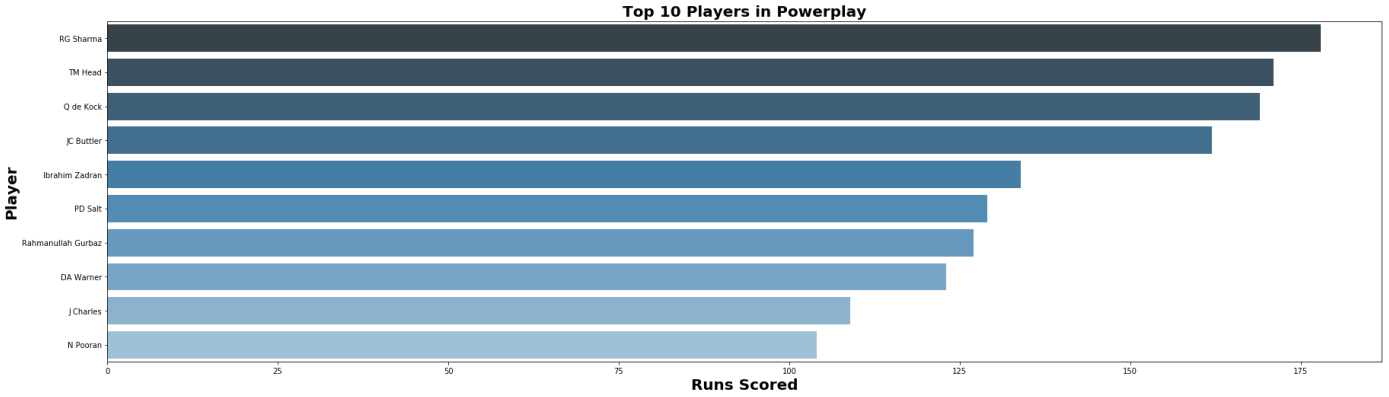
Out[19]:

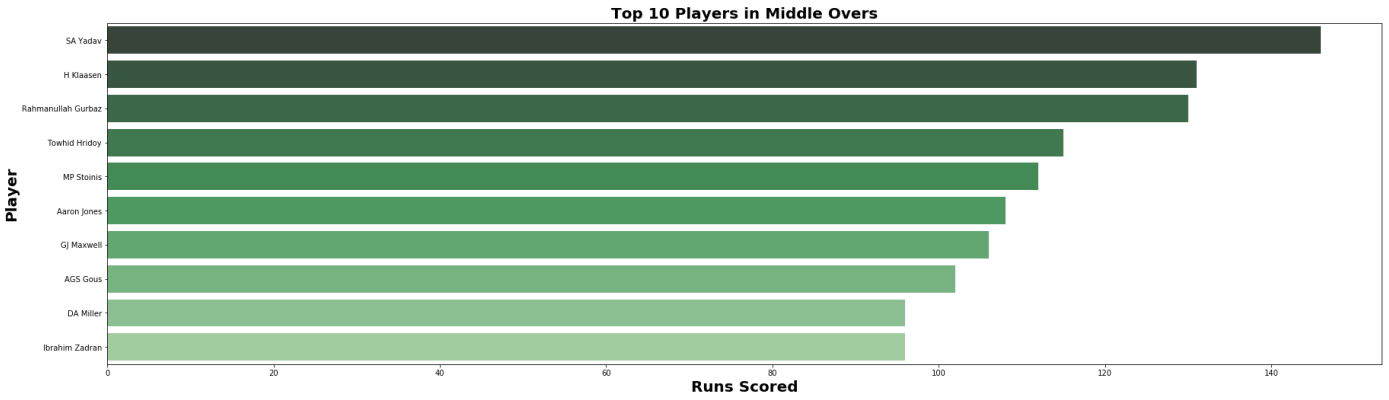
	Bowler	Consistency in Economy
0	K Bhurtel	4.000000
1	Imad Wasim	3.000000
2	J Miyaji	3.000000

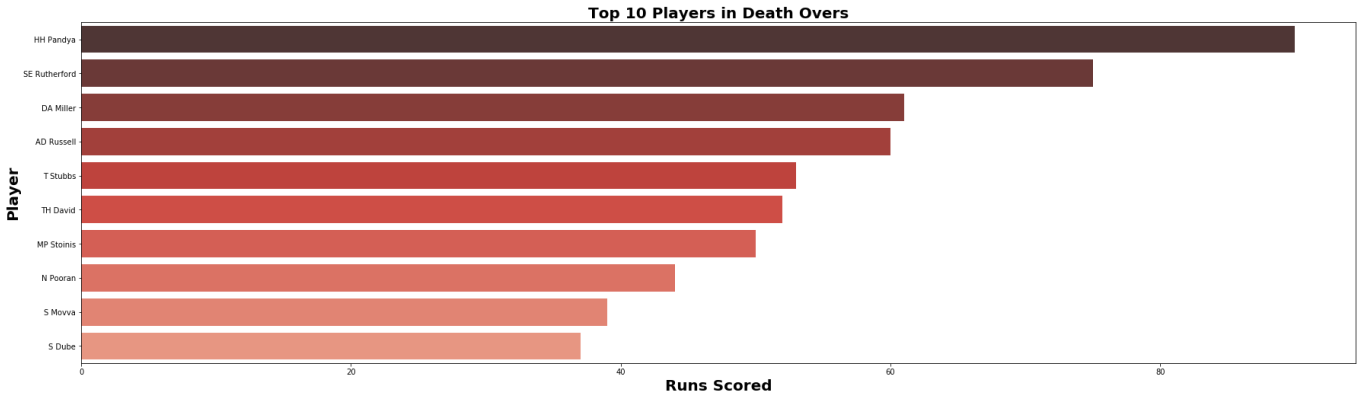
3	S Lamichhane	3.000000
4	NP Kenjige	3.000000
5	OC McCoy	3.000000
6	Fazalhaq Farooqi	2.833333
7	Rashid Khan	2.800000
8	T Shamsi	2.750000
9	CJ Jordan	2.750000

2.8 Player Performances in Powerplay, Middle Overs, and Death Overs

In [20]:
```
powerplay = deliveries[deliveries['ball'].between(0.1, 6.6)]
middle_overs = deliveries[deliveries['ball'].between(7.1, 15.6)]
death_overs = deliveries[deliveries['ball'].between(16.1, 20.6)]

powerplay_performance = powerplay.groupby('striker')['runs_off_bat'].sum().sort_values(a
middle_overs_performance = middle_overs.groupby('striker')['runs_off_bat'].sum().sort_va
death_overs_performance = death_overs.groupby('striker')['runs_off_bat'].sum().sort_valu

fig, axes = plt.subplots(3, 1, figsize=(30, 28))


sns.barplot(x=powerplay_performance .values, y=powerplay_performance.index, ax=axes[0],
axes[0].set_title('Top 10 Players in Powerplay', fontsize=20, weight='bold')
axes[0].set_xlabel('Runs Scored', fontsize=20, weight='bold')
axes[0].set_ylabel('Player', fontsize=20, weight='bold')

sns.barplot(x=middle_overs_performance.values, y=middle_overs_performance.index, ax=axes
axes[1].set_title('Top 10 Players in Middle Overs', fontsize=20, weight='bold')
axes[1].set_xlabel('Runs Scored', fontsize=20, weight='bold')
axes[1].set_ylabel('Player', fontsize=20, weight='bold')

sns.barplot(x=death_overs_performance.values, y=death_overs_performance.index, ax=axes[2
axes[2].set_title('Top 10 Players in Death Overs', fontsize=20, weight='bold')
axes[2].set_xlabel('Runs Scored', fontsize=20, weight='bold')
axes[2].set_ylabel('Player', fontsize=20, weight='bold')

plt.show()
```





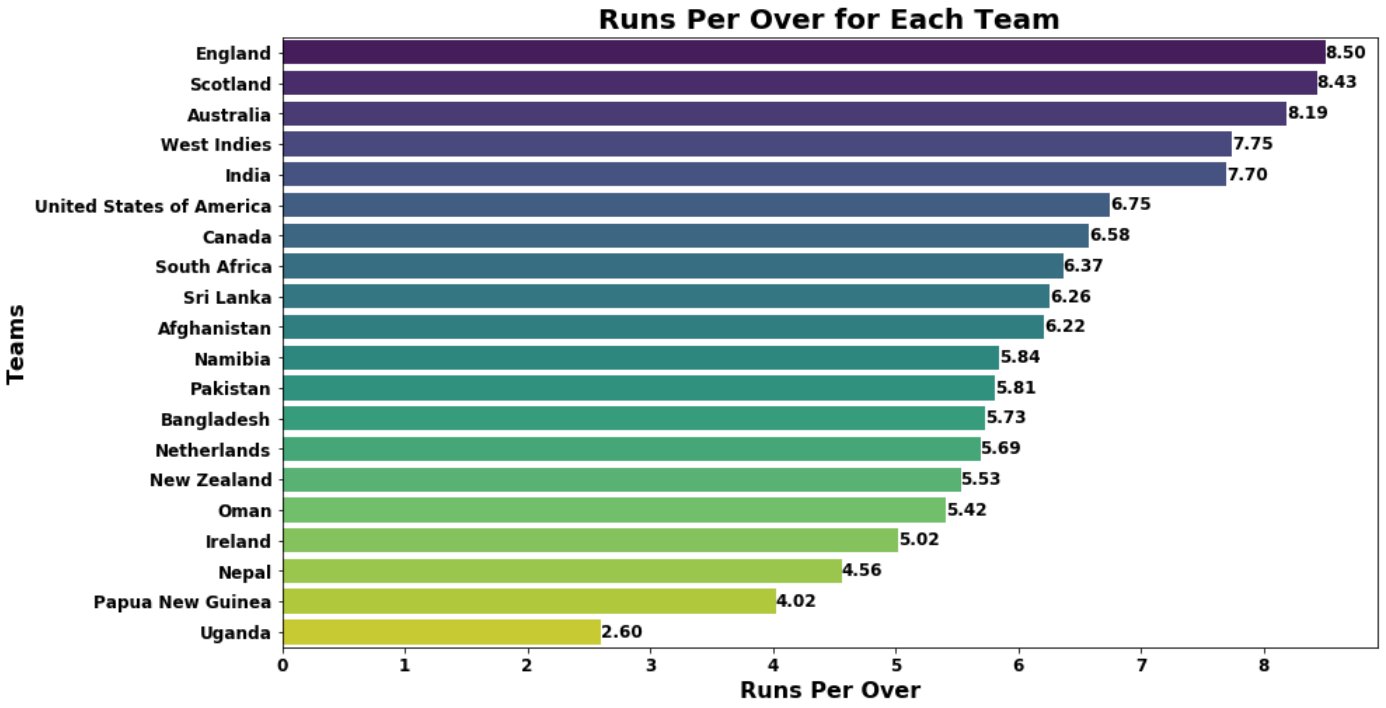


3 Batting Performance Analysis

In [21]:

```
team_runs = deliveries.groupby('batting_team')['runs_off_bat'].sum()
team_balls = deliveries.groupby('batting_team').size()

runs_per_over = team_runs / (team_balls / 6)
most_runs_per_over = runs_per_over.sort_values(ascending=False)

plt.figure(figsize=(14, 8))

barplot = sns.barplot(x=most_runs_per_over.values, y=most_runs_per_over.index, palette='
plt.title('Runs Per Over for Each Team', fontsize=20, weight='bold')
plt.xlabel('Runs Per Over', fontsize=16, weight='bold')
plt.ylabel('Teams', fontsize=16, weight='bold')

for index, value in enumerate(most_runs_per_over.values):
    plt.text(value, index, f'{value:.2f}', va='center', ha='left', fontsize=12, color='b

plt.xticks(fontsize=12, weight='bold')
plt.yticks(fontsize=12, weight='bold')
```

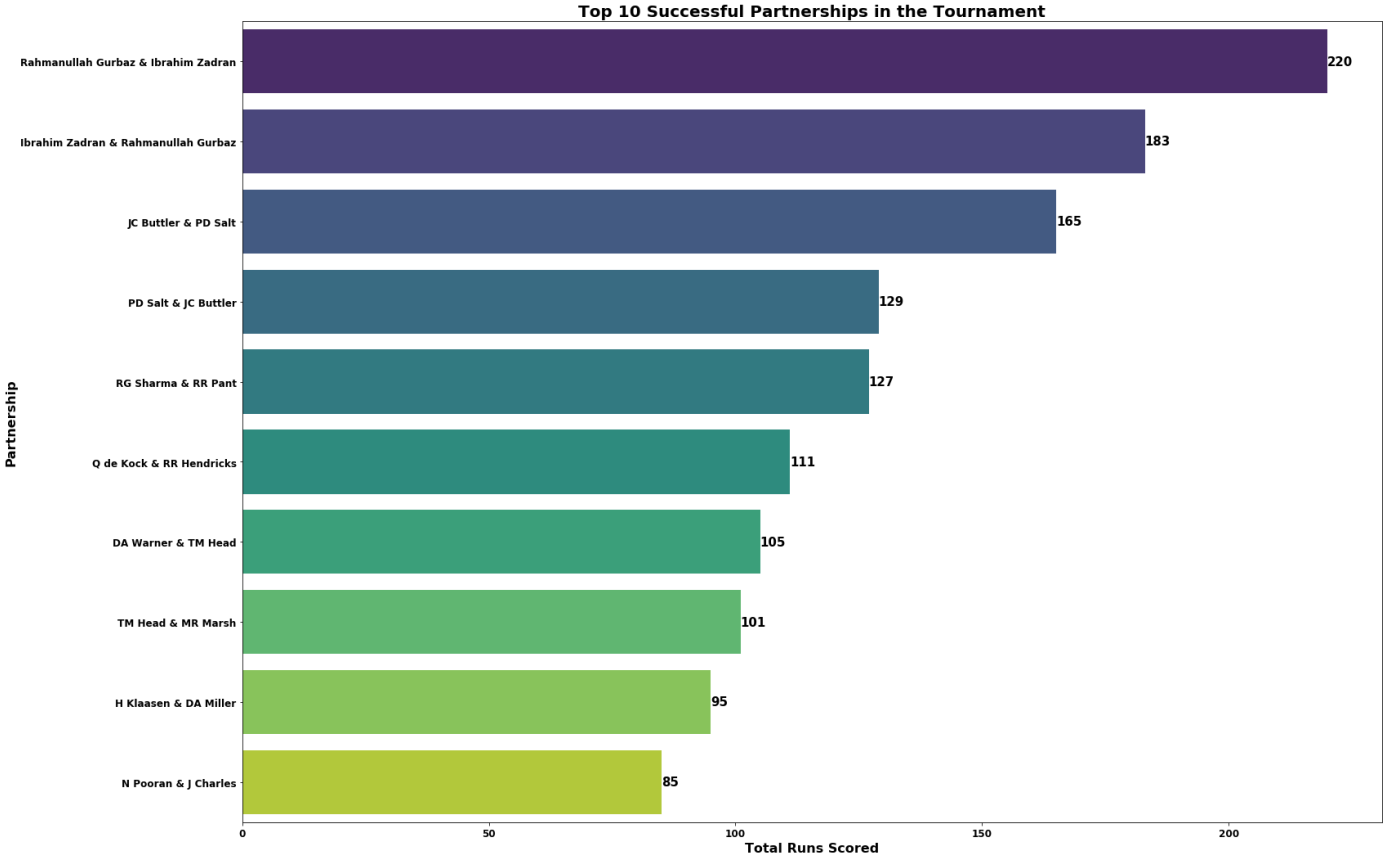
```
plt.show()
```



3.2 Most Successful Batting Partnerships

In [22]:

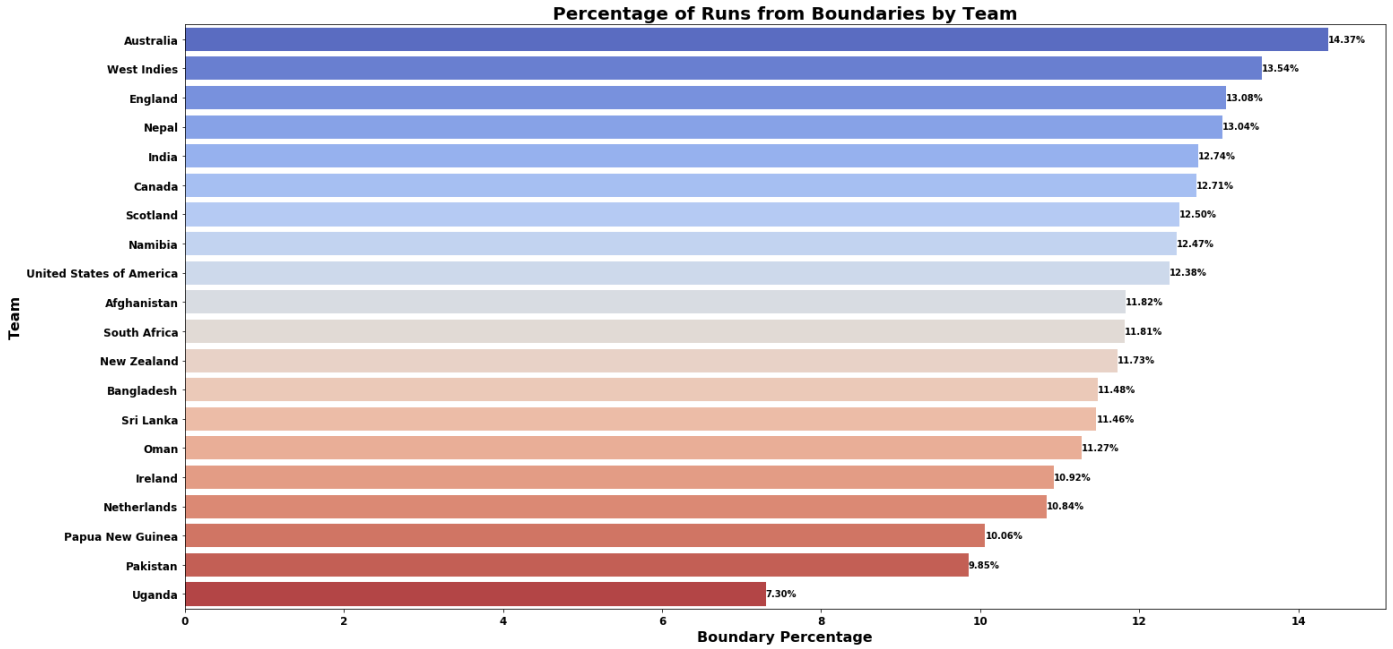
```
partnerships = deliveries.groupby(['match_id', 'striker', 'non_striker'])['runs_off_bat'

successful_partnerships = partnerships.groupby(['striker', 'non_striker']).sum().sort_va

top_partnerships = successful_partnerships.head(10)

top_partnerships['partnership'] = top_partnerships['striker'] + ' & ' + top_partnerships

plt.figure(figsize=(25, 18))

barplot = sns.barplot(data=top_partnerships, x='runs_off_bat', y='partnership', palette=
plt.title('Top 10 Successful Partnerships in the Tournament', fontsize=20, weight='bold'
plt.xlabel('Total Runs Scored', fontsize=16, weight='bold')
plt.ylabel('Partnership', fontsize=16, weight='bold')

plt.xticks(fontsize=12, weight='bold')
plt.yticks(fontsize=12, weight='bold')


for index, value in enumerate(top_partnerships['runs_off_bat']):
    plt.text(value, index, f'{value}', ha='left', va='center', fontsize=15, color='black


plt.show()
```



3.3 The Frequency of Boundaries Affect the Overall Score

In [23]:

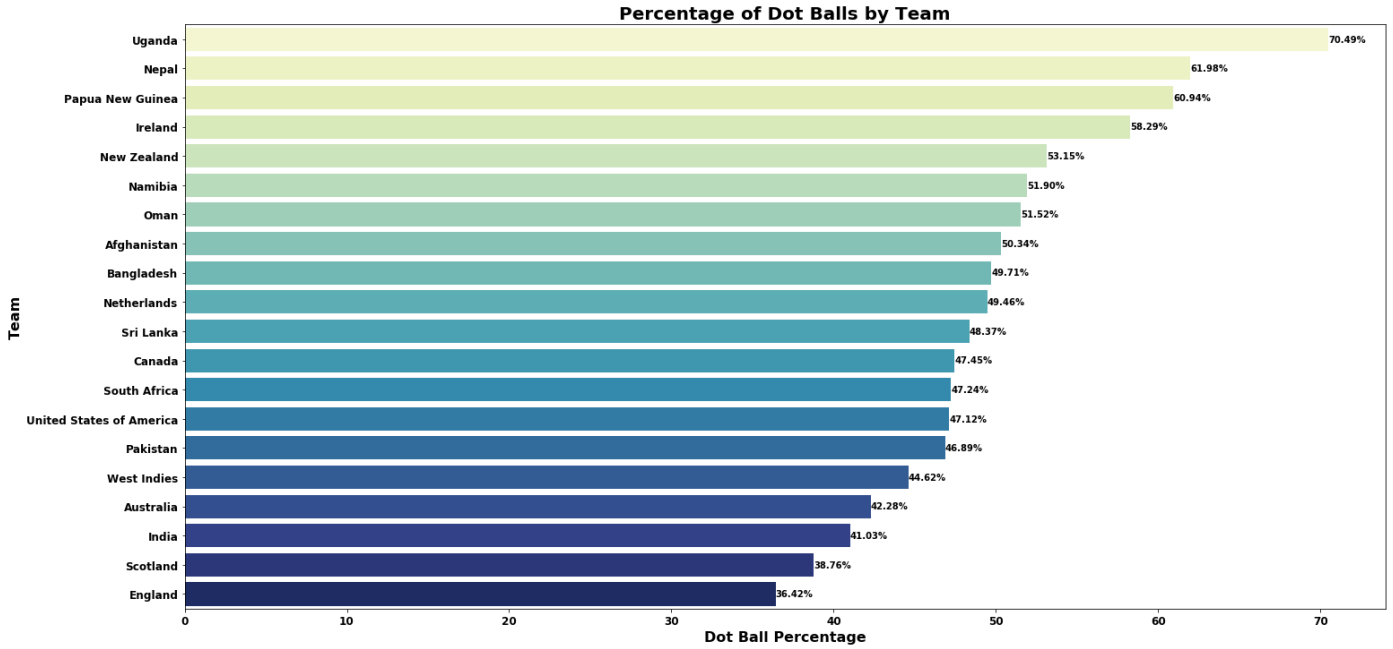
```
boundaries = deliveries[deliveries['runs_off_bat'].isin([4, 6])]
team_boundaries = boundaries.groupby('batting_team').size()
team_total_runs = deliveries.groupby('batting_team')['runs_off_bat'].sum()

boundary_percentage = (team_boundaries / team_total_runs) * 100

boundary_percentage = boundary_percentage.reset_index()
boundary_percentage.columns = ['batting_team', 'boundary_percentage']
boundary_percentage = boundary_percentage.sort_values(by='boundary_percentage', ascendin

plt.figure(figsize=(24, 12))

barplot = sns.barplot(data=boundary_percentage, x='boundary_percentage', y='batting_team
plt.title('Percentage of Runs from Boundaries by Team', fontsize=20, weight='bold')
plt.xlabel('Boundary Percentage', fontsize=16, weight='bold')
plt.ylabel('Team', fontsize=16, weight='bold')

plt.xticks(fontsize=12, weight='bold')
plt.yticks(fontsize=12, weight='bold')


for index, value in enumerate(boundary_percentage['boundary_percentage']):
    plt.text(value, index, f'{value:.2f}%', ha='left', va='center', fontsize=10, color='

plt.show()
```



3.4 Percentage of Balls are Dot Balls

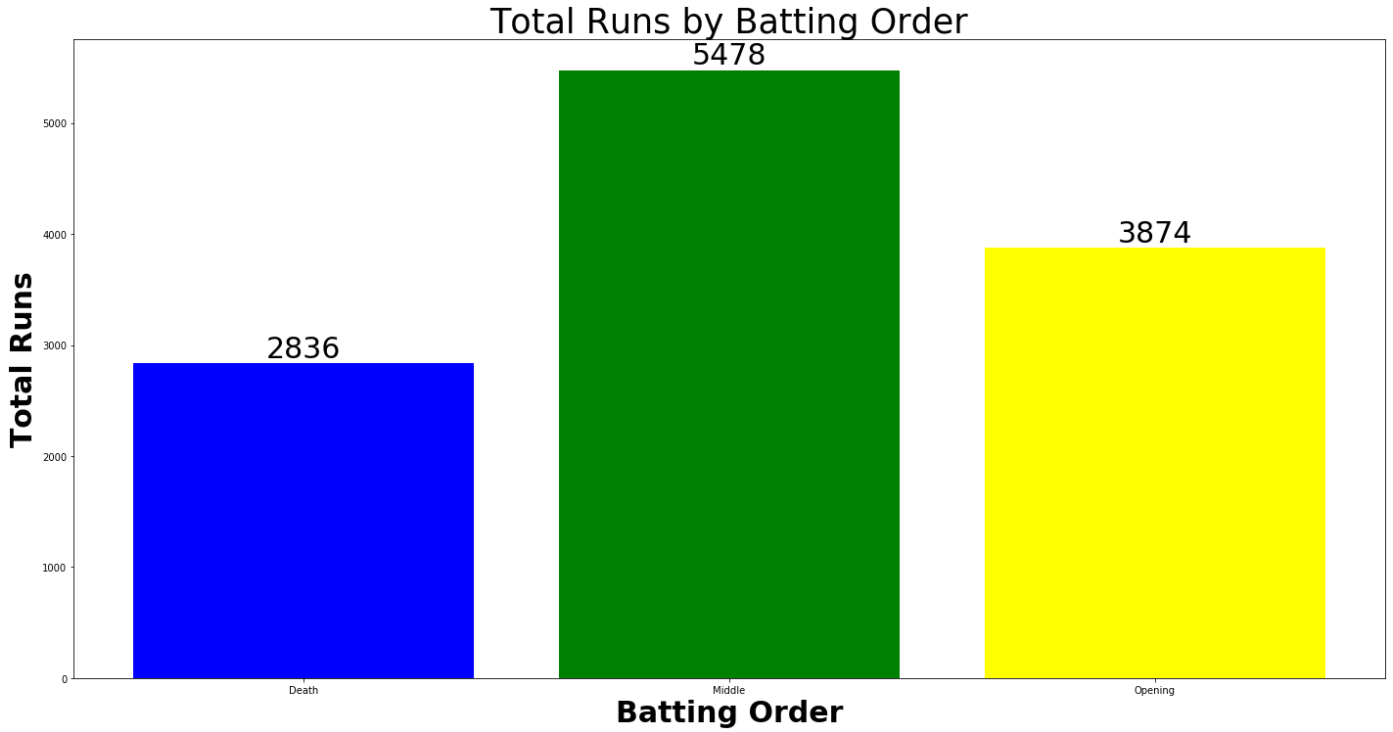
In [26]:

```
dot_balls = deliveries[deliveries['runs_off_bat'] == 0]
team_dot_balls = dot_balls.groupby('batting_team').size()

dot_ball_percentage = (team_dot_balls / team_balls) * 100

dot_ball_percentage = dot_ball_percentage.reset_index()
dot_ball_percentage.columns = ['batting_team', 'dot_ball_percentage']
dot_ball_percentage = dot_ball_percentage.sort_values(by='dot_ball_percentage', ascendin

plt.figure(figsize=(24,12))


barplot = sns.barplot(data=dot_ball_percentage, x='dot_ball_percentage', y='batting_team
plt.title('Percentage of Dot Balls by Team', fontsize=20, weight='bold')
plt.xlabel('Dot Ball Percentage', fontsize=16, weight='bold')
plt.ylabel('Team', fontsize=16, weight='bold')

plt.xticks(fontsize=12, weight='bold')
plt.yticks(fontsize=12, weight='bold')




for index, value in enumerate(dot_ball_percentage['dot_ball_percentage']):
    plt.text(value, index, f'{value:.2f}%', ha='left', va='center', fontsize=10, color='


plt.show()
```



Which phase of the innings—opening, middle, or death—contributed the highest percentage of the total runs scored in the tournament?

In [41]:
```
deliveries['batting_order']=deliveries['ball'].apply(lambda x: 'Opening' if x<6 else 'Mi
runs=deliveries.groupby('batting_order')['runs_off_bat'].sum()

plt.figure(figsize=(24,12))
bars=plt.bar(runs.index,runs.values,color=['Blue','Green','Yellow'])
plt.title('Total Runs by Batting Order',fontsize=35)
plt.xlabel('Batting Order', fontsize=30, weight='bold')
plt.ylabel('Total Runs', fontsize=30, weight='bold')


for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), ha='center', va='bot
```


Total Runs by Batting Order
5478
3874
2836
Total Runs
0
1000
2000
3000
4000
5000
Death
Middle
Opening
Batting Order

In [ ]: