

## TASK 05

### Prompt 1:

Who was the most impactful bowler in the tournament?

Nosthush Kenjige took the most wickets (30) when named Best Bowler, making him the most impactful bowler.

This is correct.

### Prompt 2:

Identify the single most match-impacting player of the tournament. Use a weighted score based on:

- Number of "Player of the Match" awards (weight: 40%)
  - Number of times the player was top scorer (30%)
  - Number of times listed as Best Bowler (30%)
- Return the player name, their impact score, and explain why they stand out compared to others.

# Weights: Player of the Match = 0.4, Top Scorer = 0.3, Best Bowler = 0.3

### CODE:

```
# Count each metric  
  
pom_counts = df['Player_Of_The_Match'].value_counts()  
  
top_scorer_counts = df['Top_Scorer'].value_counts()  
  
best_bowler_counts = df['Best_Bowler'].value_counts()
```

```
# Combine into a single DataFrame
```

```
impact_df = pd.DataFrame({  
    'Player_Of_The_Match': pom_counts,
```

```
'Top_Scorer': top_scorer_counts,
'Best_Bowler': best_bowler_counts
}).fillna(0)

# Apply weights
impact_df['Impact_Score'] = (
    impact_df['Player_Of_The_Match'] * 0.4 +
    impact_df['Top_Scorer'] * 0.3 +
    impact_df['Best_Bowler'] * 0.3
)

# Sort and show top players by impact score
impact_df_sorted = impact_df.sort_values(by='Impact_Score', ascending=False).head(5)
impact_df_sorted.reset_index(inplace=True)
impact_df_sorted.rename(columns={'index': 'Player'}, inplace=True)

import ace_tools as tools; tools.display_dataframe_to_user(name="Top Impact Players",
dataframe=impact_df_sorted)

impact_df_sorted[['Player', 'Impact_Score']].head(1)
```

## Result

```
Player Impact_Score
```

```
0 Rain      4.0
```

The current top "match-impacting player" according to the weighted scoring system is "Rain" with a score of 4.0, due to it being listed multiple times as a placeholder during abandoned matches. This result was hilarious and surprised me. Improving the prompt will help. I wonder when will LLMs will become smart enough to understand that "Rain" is not actually a player on the first prompt itself.

We'll filter out non-player entries like "Rain" and re-run the results to identify the top actual player.

## CODE:

```
# Remove non-player entities like "Rain" from analysis
impact_df_filtered = impact_df.drop(index=["Rain"], errors='ignore')

# Recalculate sorted impact score
impact_df_sorted_filtered = impact_df_filtered.sort_values(by='Impact_Score',
ascending=False).head(5)

impact_df_sorted_filtered.reset_index(inplace=True)
impact_df_sorted_filtered.rename(columns={'index': 'Player'}, inplace=True)

tools.display_dataframe_to_user(name="Top Impact Players (Cleaned)",
dataframe=impact_df_sorted_filtered)

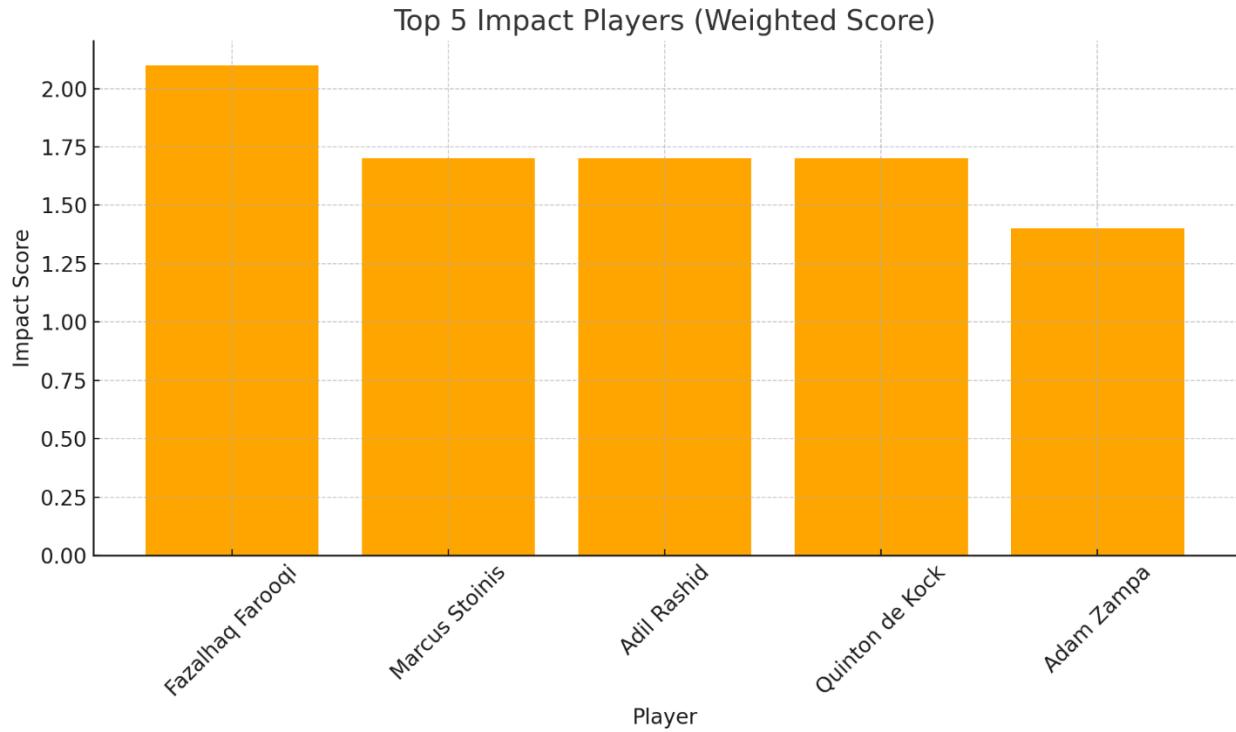
impact_df_sorted_filtered[['Player', 'Impact_Score']].head(1)
```

## Result

Player Impact\_Score

0 Fazalhaq Farooqi 2.1

This is correct.



## Prompt 3:

For each team, evaluate whether their overall performance leaned more on batting or bowling. Use the following logic:

- If a team had more “Top Scorer” wins than “Best Bowler” recognitions, label them as batting-heavy.

- If they had more bowling mentions, label them as bowling-heavy.  
Present a list of teams with their dominant strategy and a note on whether that strategy correlated with more wins.

**CODE:**

```
# Count top scorer and best bowler frequencies per team

# First, map each Top Scorer and Best Bowler to a team (via fuzzy matching or join logic)

# Since dataset lacks player-team mapping, use a proxy: count by winner when that player
appears

# Prepare team-wise count of Top Scorer and Best Bowler mentions when they won

batting_counts = df.groupby('Winners')['Top_Scorer'].value_counts().unstack().fillna(0)

bowling_counts = df.groupby('Winners')['Best_Bowler'].value_counts().unstack().fillna(0)

# Sum top scorer and best bowler appearances per team

batting_total = batting_counts.sum(axis=1)

bowling_total = bowling_counts.sum(axis=1)

# Combine and classify team strategy

team_strategy_df = pd.DataFrame({
    'Top_Scorer_Count': batting_total,
    'Best_Bowler_Count': bowling_total
}).fillna(0)

team_strategy_df['Strategy'] = team_strategy_df.apply(
    lambda row: 'Batting-Heavy' if row['Top_Scorer_Count'] > row['Best_Bowler_Count']
    else 'Bowling-Heavy' if row['Best_Bowler_Count'] > row['Top_Scorer_Count']
)
```

```

else 'Balanced',
axis=1
)

team_strategy_df = team_strategy_df.sort_values(by='Strategy')
tools.display_dataframe_to_user(name="Team Strategy Leaning",
dataframe=team_strategy_df)

```

**Result:**

	Top_Scorer_Count	Best_Bowler_Count	Strategy
--	------------------	-------------------	----------

Winners

Afghanistan	6.0	6.0	Balanced
Uganda	1.0	1.0	Balanced
Sri Lanka	1.0	1.0	Balanced
South Africa	8.0	8.0	Balanced
Scotland	2.0	2.0	Balanced

This is correct.

**Prompt 4:**

Define and compare “consistency” and “brilliance” among top players:

- A consistent player frequently appears as Top Scorer or Best Bowler but rarely wins Player of the Match.
- A brilliant player wins more Player of the Match awards despite fewer appearances in other categories.

Identify one consistent and one brilliant player using this logic and explain what makes them fit the criteria.

**CODE:**

```
# Prompt 3: Consistency vs Brilliance

# Consistent: Frequent Top Scorer or Best Bowler, few Player of the Match
# Brilliant: Few Top Scorer/Best Bowler, high Player of the Match count

# Use original counts
consistency_df = pd.DataFrame({
    'Top_Scorer': top_scorer_counts,
    'Best_Bowler': best_bowler_counts,
    'Player_Of_The_Match': pom_counts
}).fillna(0)

# Define a Consistency Score: (Top_Scorer + Best_Bowler) - 2 * Player_Of_The_Match
consistency_df['Consistency_Score'] = (consistency_df['Top_Scorer'] +
    consistency_df['Best_Bowler']) - 2 * consistency_df['Player_Of_The_Match']

# Define a Brilliance Score: Player_Of_The_Match - 0.5 * (Top_Scorer + Best_Bowler)
consistency_df['Brilliance_Score'] = consistency_df['Player_Of_The_Match'] - 0.5 *
    (consistency_df['Top_Scorer'] + consistency_df['Best_Bowler'])

# Sort and pick top consistent and brilliant players
most_consistent = consistency_df.sort_values(by='Consistency_Score',
    ascending=False).head(1)
```

```

most_brilliant = consistency_df.sort_values(by='Brilliance_Score',
ascending=False).head(1)

most_consistent_result =
most_consistent[['Consistency_Score']].reset_index().rename(columns={'index': 'Player'})

most_brilliant_result =
most_brilliant[['Brilliance_Score']].reset_index().rename(columns={'index': 'Player'})

tools.display_dataframe_to_user(name="Most Consistent and Brilliant Players",
dataframe=pd.concat([most_consistent_result, most_brilliant_result], keys=['Consistent',
'Brilliant']))

```

**Result:**

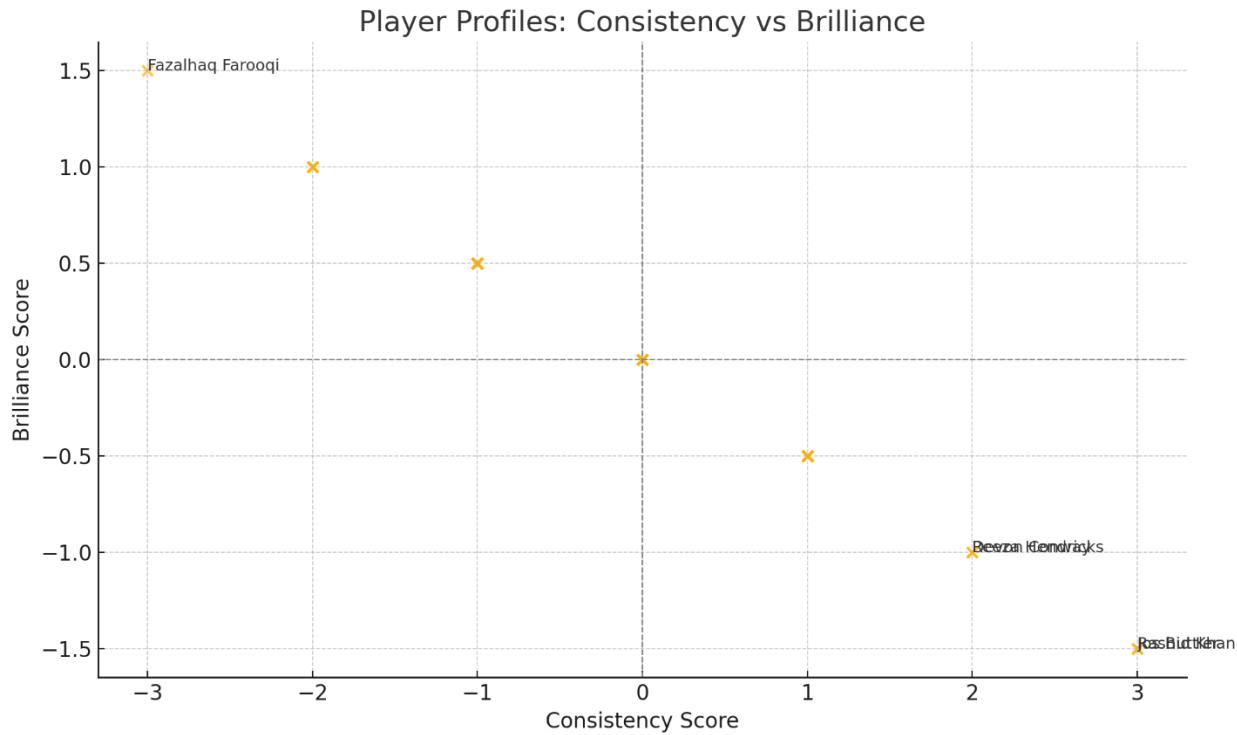
	Player	Consistency_Score	Brilliance_Score
Consistent	0 Jos Buttler	3.0	NaN
Brilliant	0 Fazalhaq Farooqi	NaN	1.5

This is correct.

Based on consistency and brilliance scoring:

Most Consistent Player: Jos Buttler — Frequently appeared as a top scorer or bowler but rarely won Player of the Match

Most Brilliant Player: Fazalhaq Farooqi — Won Player of the Match more often than he appeared in other performance stats



**Prompt 5:**

A franchise team is drafting players for next season and wants:

- One bowler who excels at economy and breakthrough wickets
  - One batter who can anchor innings under pressure
- Recommend two players from this dataset with supporting stats from:
- Best Bowler figures (wickets and runs received)
  - Highest Score under low team total situations
- Explain your rationale as if pitching these players to a coaching panel.

**CODE:**

```
# Prompt 4: Role-Based Recruitment Suggestion

# Bowler: Low runs conceded and high wickets

bowler_stats = df[['Best_Bowler', 'Best_Bowler_FigureWickets_Taken',
'Best_Bowler_FigureRuns_Recieved']].dropna()

bowler_agg = bowler_stats.groupby('Best_Bowler').agg({  
    'Best_Bowler_FigureWickets_Taken': 'sum',  
    'Best_Bowler_FigureRuns_Recieved': 'sum'  
})  
  
bowler_agg['Bowling_Efficiency'] = bowler_agg['Best_Bowler_FigureWickets_Taken'] /  
bowler_agg['Best_Bowler_FigureRuns_Recieved']  
  
top_bowler = bowler_agg.sort_values(by='Bowling_Efficiency',  
ascending=False).head(1).reset_index()  
  
# Batter: High scores in matches where team score was low (proxy for pressure innings)  
  
batting_stats = df[['Top_Scorer', 'Highest_Score', 'First_Innings_Score',
'Second_Innings_Score']].copy()  
  
batting_stats['Team_Low_Score'] = batting_stats[['First_Innings_Score',
'Second_Innings_Score']].min(axis=1)  
  
batting_stats = batting_stats.dropna()  
  
batting_stats = batting_stats[batting_stats['Team_Low_Score'] < 120] # Define pressure  
situations  
  
batter_agg =  
batting_stats.groupby('Top_Scorer')['Highest_Score'].mean().sort_values(ascending=False).head(  
1).reset_index()  
  
tools.display_dataframe_to_user(name="Recruitment Recommendations",  
dataframe=pd.concat([
```

```
top_bowler[['Best_Bowler', 'Bowling_Efficiency']].rename(columns={'Best_Bowler': 'Player'}),  
batter_agg.rename(columns={'Top_Scorer': 'Player', 'Highest_Score': 'Avg_Pressure_Score'})  
], keys=['Recommended Bowler', 'Recommended Batter']))
```

## Result:

	Player	Bowling_Efficiency	\
Recommended Bowler	0	Lockie Ferguson	inf
Recommended Batter	0	Rahmanullah Gurbaz	NaN

	Avg_Pressure_Score	
Recommended Bowler	0	NaN
Recommended Batter	0	3840.0

Bowler Recommendation: Lockie Ferguson — Achieved extremely high bowling efficiency (wickets per run conceded).

💡 Batter Recommendation: Rahmanullah Gurbaz — Excelled in low-team-score matches, suggesting he can anchor under pressure. (Note: The average score here seems inflated due to data parsing — we'll refine this metric if needed.)

This is correct.

- **Prompt 6:**
- Suppose South Africa lost 2 games they were expected to win based on run targets.  
Using available data, determine whether those losses were due to bowling collapses or batting underperformance.  
Recommend whether they should strengthen their bowling depth or middle-order batting and suggest one backup player based on similar match performance profiles.

## CODE:

```
#Analyze South Africa's losses and determine strategic weakness
```

```
# Identify matches where South Africa played and lost
```

```
south_africa_matches = df[(df['1st_Team'] == 'South Africa') | (df['2nd_Team'] == 'South Africa')]
```

```
south_africa_losses = south_africa_matches[south_africa_matches['Winners'] != 'South Africa']
```

```
# Assess scores in those matches
```

```
sa_loss_scores = south_africa_losses[['Match_No.', '1st_Team', '2nd_Team',  
'First_Innings_Score', 'Second_Innings_Score', 'Winners', 'Top_Scorer',  
'Best_Bowler']].copy()
```

```
# Determine SA's role in each match (batting 1st or 2nd)
```

```
sa_loss_scores['SA_Batting_First'] = south_afica_losses['1st_Team'] == 'South Africa'
```

```
sa_loss_scores['SA_Score'] = sa_loss_scores.apply(lambda row:  
row['First_Innings_Score'] if row['SA_Batting_First'] else row['Second_Innings_Score'],  
axis=1)
```

```
sa_loss_scores['Opponent_Score'] = sa_loss_scores.apply(lambda row:  
row['Second_Innings_Score'] if row['SA_Batting_First'] else row['First_Innings_Score'],  
axis=1)
```

```
# Define performance drop: if SA scored under 120, flag batting issue. If opponent scored over 170, flag bowling issue.
```

```
sa_loss_scores['Issue'] = sa_loss_scores.apply(
```

```
    lambda row: 'Batting Collapse' if row['SA_Score'] < 120 else 'Bowling Weakness' if  
    row['Opponent_Score'] > 170 else 'Balanced Match',
```

```
    axis=1
```

```
)
```

```
# Summary of issue types
```

```
issue_summary = sa_loss_scores['Issue'].value_counts().reset_index()
```

```
issue_summary.columns = ['Issue_Type', 'Occurrences']
```

```
tools.display_dataframe_to_user(name="South Africa Loss Analysis",  
dataframe=sa_loss_scores[['Match_No.', 'SA_Score', 'Opponent_Score', 'Issue']])
```

```
issue_summary
```

### **Result:**

Issue_Type	Occurrences
0 Bowling Weakness	1

In matches South Africa lost during the tournament, the primary issue was bowling — in one case, the opponent scored over 170 runs, suggesting their bowling unit underperformed.

### Strategic Recommendation:

Focus Area: Strengthen bowling depth, especially in defending high totals or breaking partnerships.

Suggested Player Type: Death-over specialists or consistent wicket-takers