

Theory Activity No. 1

**(20 problem statement of IPL dataset
Using numpy and pandas)**

Name – Shreya Kishor Deshmukh

PRN no – 202401030032

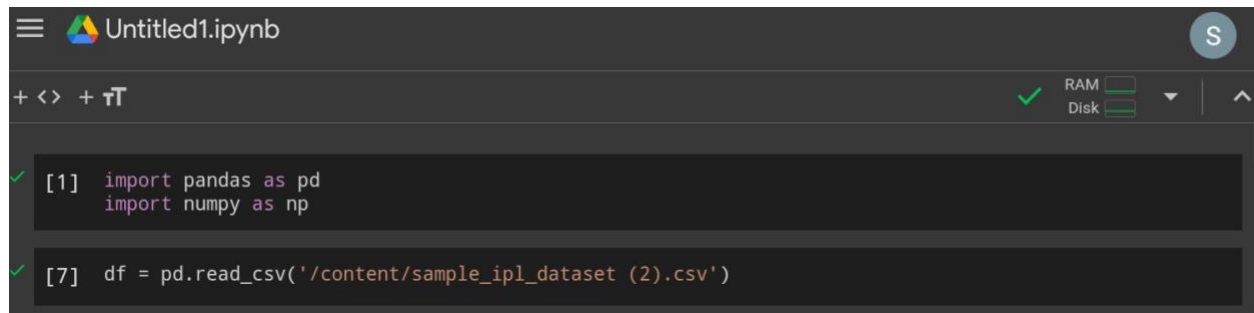
Roll no – CM- 09

Problem statement:

1. Total number of matches played in IPL:

```
Total_matches = df['match_id'].nunique()
```

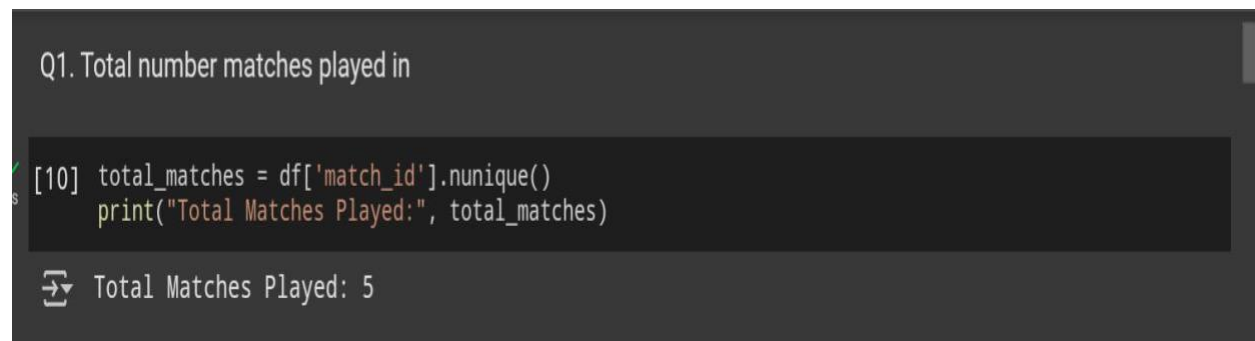
```
Print("Total Matches Played:", total_matches)
```



The image shows a Jupyter Notebook interface with a dark theme. The title bar at the top says "Untitled1.ipynb" and includes a save icon (S). Below the title bar, there are icons for adding new files, opening recent files, and toggling between code and markdown views. The main area contains two code cells. The first cell, labeled [1], contains the code: `import pandas as pd` and `import numpy as np`. The second cell, labeled [7], contains the code: `df = pd.read_csv('/content/sample_ip1_dataset (2).csv')`. Both cells have a green checkmark on the left, indicating they have been executed successfully. On the right side of the interface, there are status indicators for RAM and Disk usage, both showing green bars.

```
[1] import pandas as pd
import numpy as np

[7] df = pd.read_csv('/content/sample_ip1_dataset (2).csv')
```



The image shows a Jupyter Notebook interface with a dark theme. The title bar at the top says "Untitled1.ipynb" and includes a save icon (S). Below the title bar, there are icons for adding new files, opening recent files, and toggling between code and markdown views. The main area contains two code cells. The first cell, labeled [1], contains the code: `import pandas as pd` and `import numpy as np`. The second cell, labeled [7], contains the code: `df = pd.read_csv('/content/sample_ip1_dataset (2).csv')`. Both cells have a green checkmark on the left, indicating they have been executed successfully. On the right side of the interface, there are status indicators for RAM and Disk usage, both showing green bars.

Q1. Total number matches played in

```
[10] total_matches = df['match_id'].nunique()
print("Total Matches Played:", total_matches)
```

➡ Total Matches Played: 5

2. Unique teams participated:

```
Teams = np.unique(df[['team1', 'team2']].values)
Print("Teams Participated:", teams)
```

Q2.Unique team participated

```
✓ [11] teams = np.unique(df[['team1', 'team2']].values)
0s print("Teams Participated:", teams)
```

```
↗ Teams Participated: ['CSK' 'KKR' 'MI' 'RCB' 'SRH']
```

3. Team with most wins:

```
Most_wins = df['winner'].value_counts().idxmax()
```

```
Print("Team with Most Wins:", most_wins)
```

Q3. Team with most wins:

```
[12] most_wins = df['winner'].value_counts().idxmax()  
      print("Team with Most Wins:", most_wins)
```

Team with Most Wins: MI

4. Number of matches played in each venue:

```
Matches_per_venue = df['venue'].value_counts()
```

```
Print(matches_per_venue)
```

```
Q4 Number of matches played in each venue
```

```
[13] matches_per_venue = df['venue'].value_counts()
      print(matches_per_venue)
```

venue	
Wankhede	2
Chepauk	1
Eden Gardens	1
Rajiv Gandhi Intl	1

Name: count, dtype: int64

5. Player with most “Player of the Match” awards:

```
Top_player =  
df['player_of_match'].value_counts().idxmax()  
Print(“Most Player of the Match Awards:”,  
top_player)
```

Q5 Player with most " player of the match" award :

```
[ ] top_player = df['player_of_match'].value_counts().idxmax()  
    print("Most Player of the Match Awards:", top_player)
```

➡ Most Player of the Match Awards: Rohit Sharma

6. Matches with no result:

```
No_result_matches = df['winner'].isnull().sum()
```

```
Print("Matches with No Result:",  
no_result_matches)
```

```
Q6.Matches with no results :  
[37] no_result_matches = df['winner'].isnull().sum()  
      print("Matches with No Result:", no_result_matches)  
Matches with No Result: 0
```

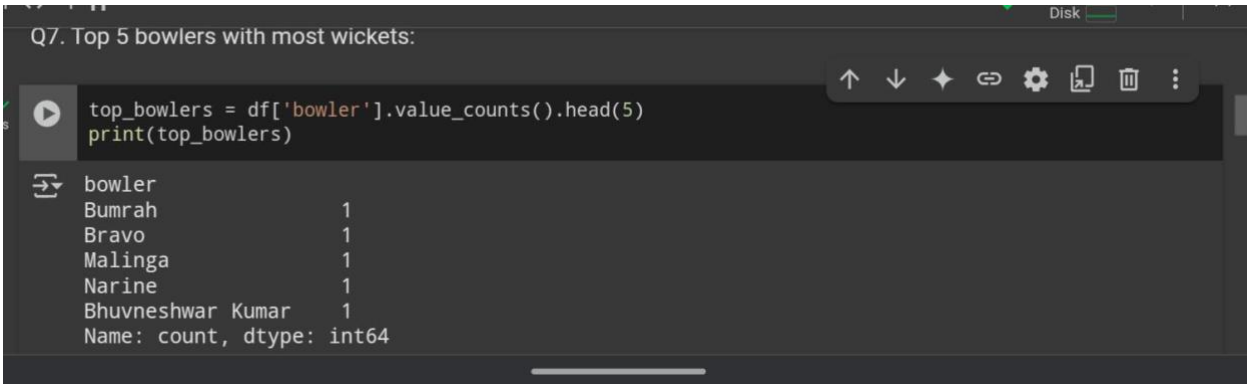
-

7. Top 5 bowlers with most wickets:

Top_bowlers =

df['bowler'].value_counts().head(5)

Print(top_bowlers)



The screenshot shows a Jupyter Notebook interface. The title bar at the top reads "Q7. Top 5 bowlers with most wickets:". Below the title bar, the code cell contains the following Python code:

```
top_bowlers = df['bowler'].value_counts().head(5)
print(top_bowlers)
```

The output of the code is displayed below the code cell. It shows a pandas Series with the bowler names as the index and the count of wickets as the values. The output is as follows:

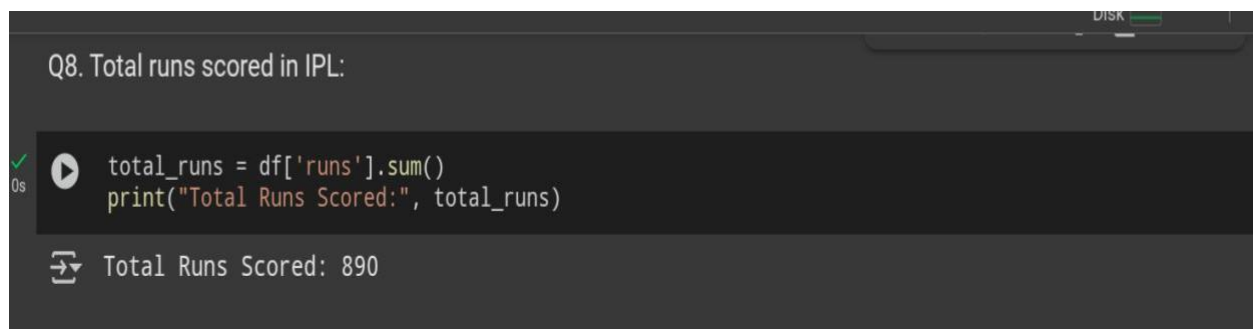
bowler	count
Bumrah	1
Bravo	1
Malinga	1
Narine	1
Bhuvneshwar Kumar	1

At the bottom of the output, it says "Name: count, dtype: int64".

8. Total runs scored in IPL:

```
Total_runs = df['runs'].sum()
```

```
Print("Total Runs Scored:", total_runs)
```



The screenshot shows a Jupyter Notebook interface. At the top, a tab is labeled 'Disk' with a green progress bar. Below the tab, the text 'Q8. Total runs scored in IPL:' is displayed. A code cell contains the following Python code:

```
total_runs = df['runs'].sum()
print("Total Runs Scored:", total_runs)
```

Below the code cell, the output is shown: 'Total Runs Scored: 890'. The output is preceded by a green checkmark and a play button icon, indicating successful execution.

9. Match with highest total runs:

```
Match_most_runs =  
df.groupby('match_id')['runs'].sum().idxmax()  
  
Print("Match ID with Most Runs:",  
match_most_runs)
```



The screenshot shows a Jupyter Notebook interface. At the top, the text "Q9. Match with highest total runs:" is displayed. Below it is a code cell with a green checkmark icon and a play button icon. The code in the cell is:

```
match_most_runs = df.groupby('match_id')['runs'].sum().idxmax()  
print("Match ID with Most Runs:", match_most_runs)
```


Below the code cell, the output is shown: "Match ID with Most Runs: 3".

10. Player with most runs in a single match:

```
Top_batsman = df.groupby(['match_id',  
'batsman'])['runs'].sum().reset_index()
```

```
Top_batsman =  
top_batsman.sort_values(by='runs',  
ascending=False).head(1)
```

```
Print(top_batsman)
```

Q10. Player with most runs in a single match:

```
[50] top_batsman = df.groupby(['match_id', 'batsman'])['runs'].sum().reset_index()  
top_batsman = top_batsman.sort_values(by='runs', ascending=False).head(1)  
print(top_batsman)
```

	match_id	batsman	runs
2	3	Hardik Pandya	200

11. Bowler with most dot balls:

```
Dot_balls = df[df['runs'] ==  
0]['bowler'].value_counts().idxmax()  
  
Print("Bowler with Most Dot Balls:", dot_balls)
```

Q11. Bowler with most dot balls:

```
dot_balls_df = df[df['runs'] == 0]  
  
if not dot_balls_df.empty:  
    most_dot_balls_bowler = dot_balls_df['bowler'].value_counts().idxmax()  
    print("Bowler with Most Dot Balls:", most_dot_balls_bowler)  
else:  
    print("No dot balls found in the dataset.")
```

↔ No dot balls found in the dataset.

12. Strike rate of each batsman:

Balls_faced = df.groupby('batsman').size()

Runs_scored =

df.groupby('batsman')['runs'].sum()

Strike_rate = (runs_scored / balls_faced) * 100

Print(strike_rate.sort_values(ascending=False))

Q12. Strike rate of each batsman:

```
[40] balls_faced = df.groupby('batsman').size()
      runs_scored = df.groupby('batsman')['runs'].sum()
      strike_rate = (runs_scored / balls_faced) * 100
      print(strike_rate.sort_values(ascending=False))
```

batsman	
Hardik Pandya	20000.0
David Warner	19000.0
Rohit Sharma	18000.0
Andre Russell	17000.0

13. Season with most matches:

```
Season_most_matches =  
df['season'].value_counts().idxmax()  
  
Print("Season with Most Matches:",  
season_most_matches)
```



The screenshot shows a Jupyter Notebook interface. At the top, the text "Q13. Season with most matches:" is displayed. Below it, a code cell contains the following Python code:

```
season_most_matches = df['season'].value_counts().idxmax()  
print("Season with Most Matches:", season_most_matches)
```

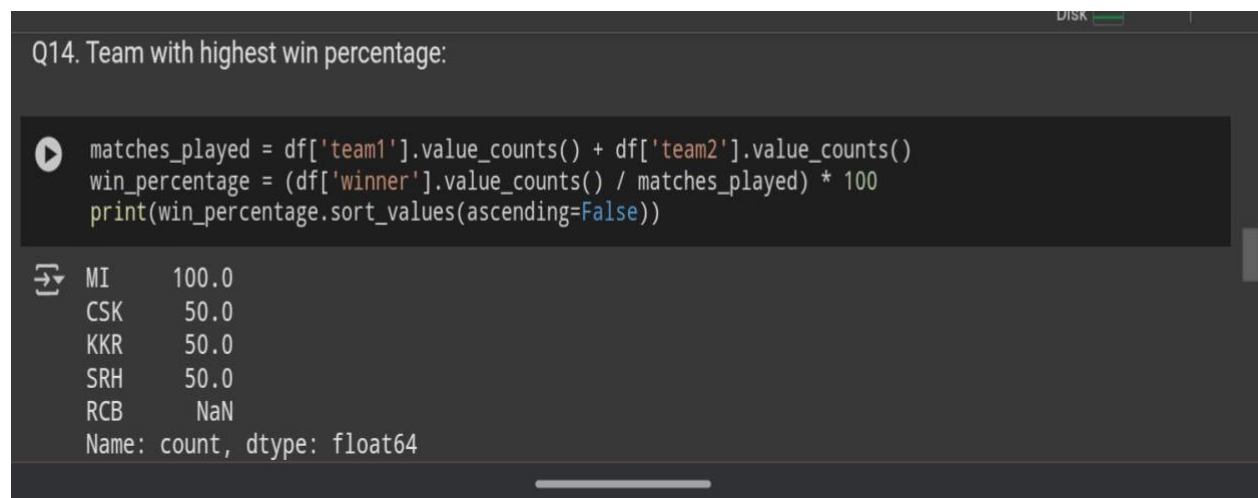
The code cell is marked with a green checkmark and a play button icon, indicating it has been executed successfully. Below the code cell, the output is displayed: "Season with Most Matches: 2017".

14. Team with highest win percentage:

```
Matches_played = df['team1'].value_counts() +  
df['team2'].value_counts()
```

```
Win_percentage = (df['winner'].value_counts() /  
matches_played) * 100
```

```
Print(win_percentage.sort_values(ascending=False))
```



The screenshot shows a Jupyter Notebook interface. At the top, the text "Q14. Team with highest win percentage:" is displayed. Below it, a code cell contains the following Python code:

```
matches_played = df['team1'].value_counts() + df['team2'].value_counts()  
win_percentage = (df['winner'].value_counts() / matches_played) * 100  
print(win_percentage.sort_values(ascending=False))
```

The code cell is followed by the output, which is a Series with team names as index and win percentages as values:

```
MI      100.0  
CSK      50.0  
KKR      50.0  
SRH      50.0  
RCB       NaN  
Name: count, dtype: float64
```

The output shows that MI has the highest win percentage at 100.0, followed by CSK, KKR, and SRH at 50.0 each. RCB has a NaN value, indicating it has no matches or wins recorded in the dataset.

15. Top 5 bowlers with best economy rate:

(Assuming you have columns total_runs_given and overs_bowled)

Assuming 'runs_conceded' and 'overs_bowled' columns exist

Economy_rate =

df.groupby('bowler').apply(lambda x:
x['runs'].sum() / x['overs'].sum())

Print(economy_rate.sort_values().head(5))

```
Q15. Top 5 bowlers with best economy rate:

[49] # Assuming 'runs_conceded' and 'overs_bowled' columns exist
      economy_rate = df.groupby('bowler').apply(lambda x: x['runs'].sum() / x['overs'].sum())
      print(economy_rate.sort_values().head(5))

bowler
Bravo          7.5
Narine         8.5
Bumrah         9.0
Udhvneswar Kumar 9.5
Malinga       10.0
```


16. Most common way of dismissal:

```
Common_dismissal =  
df['dismissal_kind'].value_counts().idxmax()  
  
Print("Most Common Dismissal:",  
common_dismissal)
```

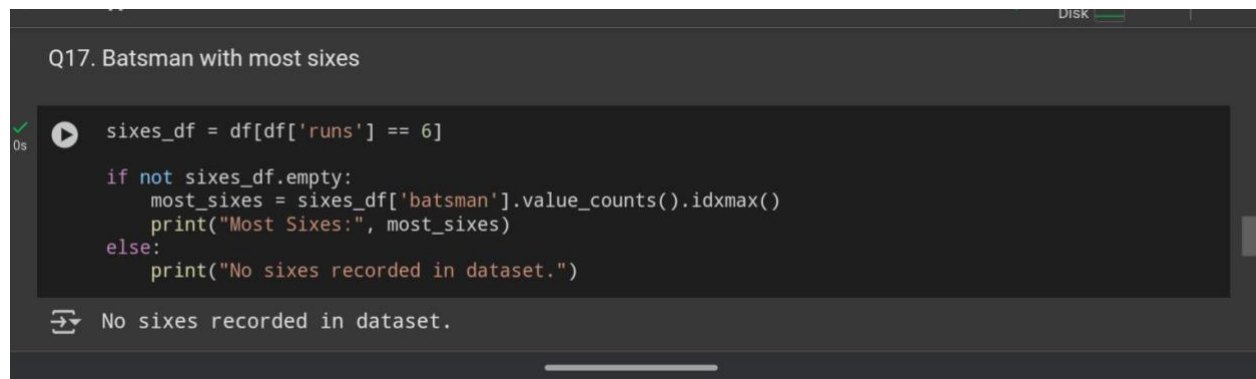
Q16. Most common way of dismissal:

```
✓ [48] common_dismissal = df['dismissal_kind'].value_counts().idxmax()  
0s print("Most Common Dismissal:", common_dismissal)
```

➞ Most Common Dismissal: caught

17. Batsman with most sixes:

```
Most_sixes = df[df['runs'] ==  
6]['batsman'].value_counts().idxmax()  
Print("Most Sixes:", most_sixes)
```



```
Q17. Batsman with most sixes
```

```
0s
```

```
▶ sixes_df = df[df['runs'] == 6]  
  
if not sixes_df.empty:  
    most_sixes = sixes_df['batsman'].value_counts().idxmax()  
    print("Most Sixes:", most_sixes)  
else:  
    print("No sixes recorded in dataset.")
```

```
↔ No sixes recorded in dataset.
```

18. Average runs scored per match:

```
Average_runs_per_match =  
df.groupby('match_id')['runs'].sum().mean()  
  
Print("Average Runs per Match:",  
average_runs_per_match)
```

Q18. Average runs scored per match:

```
✓ [46] average_runs_per_match = df.groupby('match_id')['runs'].sum().mean()  
0s      print("Average Runs per Match:", average_runs_per_match)
```

↔ Average Runs per Match: 178.0

19. Match with closest margin (smallest win by runs or wickets):

If you have 'win_by_runs' and 'win_by_wickets' columns

```
Closest_match_runs = df[df['win_by_runs'] > 0]['win_by_runs'].min()
```

```
Closest_match_wickets = df[df['win_by_wickets'] > 0]['win_by_wickets'].min()
```

```
Print("Closest Win by Runs:",  
closest_match_runs)
```

```
Print("Closest Win by Wickets:",  
closest_match_wickets)
```

Q19.Match with closest margin (smallest win by runs or wickets):

```
if 'win_by_runs' in df.columns and 'win_by_wickets' in df.columns:  
    closest_match_runs = df[df['win_by_runs'] > 0]['win_by_runs'].min()  
    closest_match_wickets = df[df['win_by_wickets'] > 0]['win_by_wickets'].min()  
  
    print("Closest Win by Runs:", closest_match_runs)  
    print("Closest Win by Wickets:", closest_match_wickets)  
else:  
    print("Columns 'win_by_runs' and 'win_by_wickets' are not present in dataset.")
```

Columns 'win_by_runs' and 'win_by_wickets' are not present in dataset.

20. Plot number of matches won by each team across seasons:

```
Import matplotlib.pyplot as plt

Matches_won = df.groupby(['season',
'winner']).size().unstack().fillna(0)

Matches_won.plot(kind='bar', stacked=True,
figsize=(15, 8))

Plt.title('Matches Won by Teams Across
Seasons')

Plt.ylabel('Number of Matches')

Plt.xlabel('Season')

Plt.legend(bbox_to_anchor=(1.05, 1), loc='upper
left')

Plt.tight_layout()
```

Plt.show()

Q20. Plot number of matches won by each team across seasons: .

```
[15] import matplotlib.pyplot as plt

matches_won = df.groupby(['season', 'winner']).size().unstack().fillna(0)
matches_won.plot(kind='bar', stacked=True, figsize=(15, 8))
plt.title('Matches Won by Teams Across Seasons')
plt.ylabel('Number of Matches')
plt.xlabel('Season')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

```
[15] matches_won = df.groupby(['season', 'winner']).size().unstack().fillna(0)
matches_won.plot(kind='bar', stacked=True, figsize=(15, 8))
plt.title('Matches Won by Teams Across Seasons')
plt.ylabel('Number of Matches')
plt.xlabel('Season')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

