IPL TEAM MANAGEMENT

Hello Everyone I am Gulzar Ahmed As we knows that we got a project for player performance analysis and for selecting those player who got better performance from 2016 to 2022 and release their name for auction for 2024IPL according to their bit value as i assigned bit value for best batsman is 50000000 and for best bowler 750000000 and as i performed in this project statistical analysis of players and then select them in two way as of best batsman and best bowler i selected top 10 batsman and top 10 bowler

Here are code for my analysis for data exploration,scrapping data and forming team ,handling with value redundancy and missling values i did everything according to my approach as i could approach

as of my reference for my analysis , exploring data there is so many resources some of as kaggle and bcci for data collection and for my analytical skill

at first i gathered my data and then setting up my enviroment for coding and analytical skill and then first i imported my required library for data handling and loading i imported pandas for data handling and matplotlib for visualization

Library Importing

In [16]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

handling Missing Values

after setting required library then i started my coding part as of i first created a list name as dfs which contains file paths of multiple csv files that are related to players score of batting and bowling this file are from different year 2016 to 2022

in the next line defining a function named as check_missing_value to read a csv file or check for missing values and print it

the function is started by reading the csv file using pd.read_csv(file_path)

and then prints a message indicating which file is being processed: print(f"\nChecking missing values for file: {file_path}")

Next, it prints the column names that have missing values using the df.columns[df.isnull().any()] This will display only the columns that contain at least one missing value.

then calculates and prints the total number of missing values in each column using "df.isnull()"".sum()". The "isnull()" function checks for missing (NaN) values, and sum() computes the number of missing values for each column.

Finally, it calculates and prints the total number of missing values in the entire DataFrame using df.isnull().sum().sum().

In the if **name** == "**main**": block, the code iterates through each CSV file path in the dfs list and calls the check_missing_values after running whole this section code for checking null value or missing value it output and code behind its as shown below.

In [17]:
```
dfs = [r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IP
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2017.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2018.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2019.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2020.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2021.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2022.
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv"]

def check_missing_values(file_path):
    df = pd.read_csv(file_path)
    print(f"\nChecking missing values for file: {file_path}")
    print("Columns with missing values:")
    print(df.columns[df.isnull().any()])
    print(f"Total missing values in each column:\n{df.isnull().sum()}")
    print(f"Total missing values in the entire dataframe: {df.isnull().sum().sum()}")

if __name__ == "__main__":
    for csv_file in dfs:
        check_missing_values(csv_file)
```

```
Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2016.csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2017.csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2018.csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
```

```
Stats\BATTING STATS - IPL_2019.csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2020.csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2021.csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2022.csv
```

```
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
6s        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
```

```
POS       0
Player    0
Mat       0
Inns      0
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
```

```
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0

Checking missing values for file: C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.
csv
Columns with missing values:
Index([], dtype='object')
Total missing values in each column:
POS       0
Player    0
Mat       0
Inns      0
Ov        0
Runs      0
Wkts      0
BBI       0
Avg       0
Econ      0
SR        0
4w        0
5w        0
dtype: int64
Total missing values in the entire dataframe: 0
```

Finding Total Number Of column and rows

again in this section of code i handling with finding the total number of columns and rows are present in this data set for this check i did this code whose explanation of code are

first again i define a function called get_rows_and_columns that takes a file_path as an argument This function is used to read a CSV file, determine whether it's a batting or bowling file, extract the year from the file name, and then print the file type, year, number of rows, and number of columns.

The function starts by reading the CSV file using pd.read_csv(file_path). As mentioned earlier

The function uses a conditional expression to determine the file type (Batting or Bowling) based on whether the string 'Batting' is present in the file_path. If 'Batting' is found, file_type is set to 'Batting'; otherwise, it is set to 'Bowling'.

The function extracts the year from the file_path using string manipulation. It first splits the file_path string by underscores ('_') and selects the last part of the resulting list using [-1]. Then, it further splits the selected part by periods ('.') and takes the first part of that split using [0]. This should extract the year from the file name.

the function then prints the file type and year using the extracted information: print(f"\nFile: {file_type} {year}")

Next, it prints the number of rows in the DataFrame using the shape[0]. The shape returns a tuple with the number of rows and columns, and shape[0] gives the number of rows

Similarly, it prints the number of columns in the DataFrame using the shape[1] attribute. shape[1] gives the number of columns.

In the if **name** == "**main**": block, the code iterates through each CSV file path in the dfs list and calls the get_rows_and_columns function for each file This will print information about the file type (Batting or Bowling), the year, the number of rows, and the number of columns for each CSV file

In [18]:

```
def get_rows_and_columns(file_path):
    df = pd.read_csv(file_path)
    file_type = 'Batting' if 'Batting' in file_path else 'Bowling'
    year = file_path.split('_')[-1].split('.')[0]
    print(f"\nFile: {file_type} {year}")
    print(f"Number of Rows: {df.shape[0]}")
    print(f"Number of Columns: {df.shape[1]}")

if __name__ == "__main__":
    for csv_file in dfs:
        get_rows_and_columns(csv_file)
```

```
File: Batting 2016
Number of Rows: 136
Number of Columns: 14

File: Batting 2017
Number of Rows: 143
Number of Columns: 14

File: Batting 2018
Number of Rows: 138
Number of Columns: 14

File: Batting 2019
Number of Rows: 144
Number of Columns: 14

File: Batting 2020
Number of Rows: 133
Number of Columns: 14

File: Batting 2021
Number of Rows: 149
Number of Columns: 14

File: Batting 2022
Number of Rows: 162
Number of Columns: 14

File: Bowling 2016
Number of Rows: 86
Number of Columns: 13

File: Bowling 2017
Number of Rows: 90
Number of Columns: 13

File: Bowling 2018
Number of Rows: 82
Number of Columns: 13

File: Bowling 2019
Number of Rows: 87
Number of Columns: 13

File: Bowling 2020
Number of Rows: 78
Number of Columns: 13

File: Bowling 2021
Number of Rows: 89
Number of Columns: 13

File: Bowling 2022
Number of Rows: 103
Number of Columns: 13
```

To know All column Names

at next section of code of my project i started with again by defining function a function called identify_columns that takes a file_path as an argument reading the CSV file using pd.read_csv(file_path). As mentioned earlier It prints a message indicating the file for which the columns are being identified: print(f"\nColumns in {file_path}:") Next, it prints the column names present in the DataFrame using df.columns if **name** == "**main**": block, the code iterates through each CSV file path in the dfs list and calls the identify_columns function for each file.

In [19]:
```
def identify_columns(file_path):
    df = pd.read_csv(file_path)
    print(f"\nColumns in {file_path}:")
    print(df.columns)

if __name__ == "__main__":
    for csv_file in dfs:
        identify_columns(csv_file)
```

```
Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2016.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2017.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2018.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2019.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2020.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2021.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - I
PL_2022.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF', 'SR',
       '100', '50', '4s', '6s'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')
```

```
Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')

Columns in C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv:
Index(['POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts', 'BBI', 'Avg',
       'Econ', 'SR', '4w', '5w'],
      dtype='object')
```

To gain Dataframe Information

function that i define every it use to take argument and call for our use cases DataFrame for which the information is being displayed: print(f"\nInformation about these DataFrame {file_path}:")

necxt it prints the information about DataFrame using df.info() method of a dataframe gives a concise summary of the dataframe including the number of non-null value and data types of each column

In [20]:
```
def dataframe_information(file_path):
    df = pd.read_csv(file_path)
    print(f"\nInformation about these DataFrame {file_path}:")
    print(df.info())

if __name__ == "__main__":
    for csv_file in dfs:
        dataframe_information(csv_file)
```

```
Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2016.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 136 entries, 0 to 135
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     136 non-null    int64  
 1   Player  136 non-null    object 
 2   Mat     136 non-null    int64  
 3   Inns    136 non-null    int64  
 4   NO      136 non-null    int64  
 5   Runs    136 non-null    int64  
 6   HS      136 non-null    object 
 7   Avg     136 non-null    float64
 8   BF      136 non-null    int64  
 9   SR      136 non-null    float64
 10  100     136 non-null    int64  
 11  50      136 non-null    int64  
 12  4s      136 non-null    int64  
 13  6s      136 non-null    int64  
dtypes: float64(2), int64(10), object(2)
memory usage: 15.0+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2017.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143 entries, 0 to 142
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     143 non-null    int64  
 1   Player  143 non-null    object 
 2   Mat     143 non-null    int64  
 3   Inns    143 non-null    int64  
 4   NO      143 non-null    int64  
 5   Runs    143 non-null    int64  
 6   HS      143 non-null    object 
 7   Avg     143 non-null    float64
 8   BF      143 non-null    int64  
 9   SR      143 non-null    float64
 10  100     143 non-null    int64  
 11  50      143 non-null    int64  
 12  4s      143 non-null    int64  
 13  6s      143 non-null    int64  
dtypes: float64(2), int64(10), object(2)
memory usage: 15.8+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2018.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 138 entries, 0 to 137
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     138 non-null    int64  
 1   Player  138 non-null    object 
 2   Mat     138 non-null    int64  
 3   Inns    138 non-null    int64  
 4   NO      138 non-null    int64  
 5   Runs    138 non-null    int64  
 6   HS      138 non-null    object 
 7   Avg     138 non-null    float64
 8   BF      138 non-null    int64  
 9   SR      138 non-null    float64
 10  100     138 non-null    int64
 11  50      138 non-null    int64
 12  4s      138 non-null    int64
 13  6s      138 non-null    int64
dtypes: float64(2), int64(10), object(2)
memory usage: 15.2+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2019.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   POS     144 non-null    int64
 1   Player  144 non-null    object
 2   Mat     144 non-null    int64
 3   Inns    144 non-null    int64
 4   NO      144 non-null    int64
 5   Runs    144 non-null    int64
 6   HS      144 non-null    object
 7   Avg     144 non-null    float64
 8   BF      144 non-null    int64
 9   SR      144 non-null    float64
 10  100     144 non-null    int64
 11  50      144 non-null    int64
 12  4s      144 non-null    int64
 13  6s      144 non-null    int64
dtypes: float64(2), int64(10), object(2)
memory usage: 15.9+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2020.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 133 entries, 0 to 132
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   POS     133 non-null    int64
 1   Player  133 non-null    object
 2   Mat     133 non-null    int64
 3   Inns    133 non-null    int64
 4   NO      133 non-null    int64
 5   Runs    133 non-null    int64
 6   HS      133 non-null    object
 7   Avg     133 non-null    float64
 8   BF      133 non-null    int64
 9   SR      133 non-null    float64
 10  100     133 non-null    int64
 11  50      133 non-null    int64
 12  4s      133 non-null    int64
 13  6s      133 non-null    int64
dtypes: float64(2), int64(10), object(2)
memory usage: 14.7+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2021.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   POS     149 non-null    int64
 1   Player  149 non-null    object
 2   Mat     149 non-null    int64  
 3   Inns    149 non-null    int64  
 4   NO      149 non-null    int64  
 5   Runs    149 non-null    int64  
 6   HS      149 non-null    object 
 7   Avg     149 non-null    float64
 8   BF      149 non-null    int64  
 9   SR      149 non-null    float64
 10  100     149 non-null    int64  
 11  50      149 non-null    int64  
 12  4s      149 non-null    int64  
 13  6s      149 non-null    int64  
dtypes: float64(2), int64(10), object(2)
memory usage: 16.4+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting
Stats\BATTING STATS - IPL_2022.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162 entries, 0 to 161
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     162 non-null    int64  
 1   Player  162 non-null    object 
 2   Mat     162 non-null    int64  
 3   Inns    162 non-null    int64  
 4   NO      162 non-null    int64  
 5   Runs    162 non-null    int64  
 6   HS      162 non-null    object 
 7   Avg     162 non-null    object 
 8   BF      162 non-null    int64  
 9   SR      162 non-null    float64
 10  100     162 non-null    int64  
 11  50      162 non-null    int64  
 12  4s      162 non-null    int64  
 13  6s      162 non-null    int64  
dtypes: float64(1), int64(10), object(3)
memory usage: 17.8+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.
csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86 entries, 0 to 85
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     86 non-null     int64  
 1   Player  86 non-null     object 
 2   Mat     86 non-null     int64  
 3   Inns    86 non-null     int64  
 4   Ov      86 non-null     int64  
 5   Runs    86 non-null     int64  
 6   Wkts    86 non-null     int64  
 7   BBI     86 non-null     object 
 8   Avg     86 non-null     float64
 9   Econ    86 non-null     float64
 10  SR      86 non-null     float64
 11  4w      86 non-null     int64  
 12  5w      86 non-null     int64  
dtypes: float64(3), int64(8), object(2)
memory usage: 8.9+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.
csv:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     90 non-null     int64  
 1   Player  90 non-null     object 
 2   Mat     90 non-null     int64  
 3   Inns    90 non-null     int64  
 4   Ov      90 non-null     int64  
 5   Runs    90 non-null     int64  
 6   Wkts    90 non-null     int64  
 7   BBI     90 non-null     object 
 8   Avg     90 non-null     float64
 9   Econ    90 non-null     float64
 10  SR      90 non-null     float64
 11  4w      90 non-null     int64  
 12  5w      90 non-null     int64  
dtypes: float64(3), int64(8), object(2)
memory usage: 9.3+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.
csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82 entries, 0 to 81
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     82 non-null     int64  
 1   Player  82 non-null     object 
 2   Mat     82 non-null     int64  
 3   Inns    82 non-null     int64  
 4   Ov      82 non-null     int64  
 5   Runs    82 non-null     int64  
 6   Wkts    82 non-null     int64  
 7   BBI     82 non-null     object 
 8   Avg     82 non-null     float64
 9   Econ    82 non-null     float64
 10  SR      82 non-null     float64
 11  4w      82 non-null     int64  
 12  5w      82 non-null     int64  
dtypes: float64(3), int64(8), object(2)
memory usage: 8.5+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.
csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87 entries, 0 to 86
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     87 non-null     int64  
 1   Player  87 non-null     object 
 2   Mat     87 non-null     int64  
 3   Inns    87 non-null     int64  
 4   Ov      87 non-null     int64  
 5   Runs    87 non-null     int64  
 6   Wkts    87 non-null     int64  
 7   BBI     87 non-null     object 
 8   Avg     87 non-null     float64
 9   Econ    87 non-null     float64
 10  SR      87 non-null     float64
 11  4w      87 non-null     int64  
 12  5w      87 non-null     int64  
dtypes: float64(3), int64(8), object(2)
```

```
memory usage: 9.0+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.
csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     78 non-null     int64  
 1   Player  78 non-null     object 
 2   Mat     78 non-null     int64  
 3   Inns    78 non-null     int64  
 4   Ov      78 non-null     int64  
 5   Runs    78 non-null     int64  
 6   Wkts    78 non-null     int64  
 7   BBI     78 non-null     object 
 8   Avg     78 non-null     float64
 9   Econ    78 non-null     float64
 10  SR      78 non-null     float64
 11  4w      78 non-null     int64  
 12  5w      78 non-null     int64  
dtypes: float64(3), int64(8), object(2)
memory usage: 8.0+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.
csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89 entries, 0 to 88
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     89 non-null     int64  
 1   Player  89 non-null     object 
 2   Mat     89 non-null     int64  
 3   Inns    89 non-null     int64  
 4   Ov      89 non-null     int64  
 5   Runs    89 non-null     int64  
 6   Wkts    89 non-null     int64  
 7   BBI     89 non-null     object 
 8   Avg     89 non-null     float64
 9   Econ    89 non-null     float64
 10  SR      89 non-null     float64
 11  4w      89 non-null     int64  
 12  5w      89 non-null     int64  
dtypes: float64(3), int64(8), object(2)
memory usage: 9.2+ KB
None

Information about these DataFrame C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.
csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   POS     103 non-null    int64  
 1   Player  103 non-null    object 
 2   Mat     103 non-null    int64  
 3   Inns    103 non-null    int64  
 4   Ov      103 non-null    float64
 5   Runs    103 non-null    int64  
 6   Wkts    103 non-null    int64  
 7   BBI     103 non-null    object 
 8   Avg     103 non-null    float64
 9   Econ    103 non-null    float64
 10  SR      103 non-null    float64
 11  4w      103 non-null    int64
 12  5w      103 non-null    int64
dtypes: float64(4), int64(7), object(2)
memory usage: 10.6+ KB
None
```

Summary statics of numerical Columns

in this section of code i try to prints the summary statistics for numerical columns in the DataFrame using df.describe(). The describe() method of a DataFrame provides various summary statistics, including count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum values for each numerical column

In [21]:
```
def summary_statics(file_path):
    df = pd.read_csv(file_path)
    print(f"\nSummary statistics for numerical columns {file_path}:")
    print(df.describe())

if __name__ == "__main__":
    for csv_file in dfs:
        summary_statics(csv_file)
```

```
Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
atting Stats\BATTING STATS - IPL_2016.csv:
              POS         Mat        Inns          NO        Runs         Avg  \
count  136.000000  136.000000  136.000000  136.000000  136.000000  136.000000
mean    68.500000    8.970588    6.463235    1.573529  132.073529   18.716691
std     39.403892    4.877676    4.854792    1.533007  170.831068   15.969887
min      1.000000    1.000000    1.000000    0.000000    0.000000    0.000000
25%     34.750000    4.000000    2.000000    0.000000   12.500000    6.210000
50%     68.500000    9.000000    5.000000    1.000000   57.000000   16.900000
75%    102.250000   14.000000   10.250000    3.000000  188.750000   26.550000
max    136.000000   17.000000   17.000000    8.000000  973.000000   81.080000

               BF          SR         100          50          4s          6s
count  136.000000  136.000000  136.000000  136.000000  136.000000  136.000000
mean   100.500000  115.034118    0.051471    0.808824   12.000000    4.691176
std    121.995264   45.284237    0.371580    1.621680   17.081504    6.668349
min      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%     13.000000  100.000000    0.000000    0.000000    1.000000    0.000000
50%     40.500000  120.645000    0.000000    0.000000    4.000000    2.000000
75%    143.250000  138.600000    0.000000    1.000000   17.000000    6.500000
max    640.000000  233.330000    4.000000    9.000000   88.000000   38.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
atting Stats\BATTING STATS - IPL_2017.csv:
             POS         Mat        Inns          NO        Runs         Avg  \
count  143.00000  143.000000  143.000000  143.000000  143.000000  143.000000
mean    72.00000    8.790210    6.377622    1.426573  125.223776   17.709510
std     41.42463    4.712664    4.757210    1.633667  146.982581   14.645015
min      1.00000    1.000000    1.000000    0.000000    0.000000    0.000000
25%     36.50000    5.000000    2.000000    0.000000    9.500000    4.165000
50%     72.00000    9.000000    5.000000    1.000000   51.000000   16.250000
75%    107.50000   13.000000   10.000000    2.000000  241.000000   27.680000
max    143.00000   17.000000   16.000000    9.000000  641.000000   60.000000

               BF          SR         100          50          4s          6s
count  143.000000  143.000000  143.000000  143.000000  143.000000  143.000000
mean    93.902098  112.154825    0.034965    0.664336   11.244755    4.930070
std    105.711960   45.458629    0.219236    1.093798   14.520125    6.525843
min      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%     12.000000   87.405000    0.000000    0.000000    0.000000    0.000000
50%     41.000000  120.930000    0.000000    0.000000    4.000000    1.000000
75%    166.000000  139.970000    0.000000    1.000000   20.500000    8.000000
max    452.000000  233.330000    2.000000    5.000000   63.000000   26.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
atting Stats\BATTING STATS - IPL_2018.csv:
              POS         Mat        Inns          NO        Runs         Avg  \
count  138.000000  138.000000  138.000000  138.000000  138.000000  138.000000
mean    69.500000    9.318841    6.753623    1.536232  138.391304   18.570000
std     39.981246    4.916805    4.953524    1.557461  175.839402   16.187522
min      1.000000    1.000000    1.000000    0.000000    0.000000    0.000000
25%     35.250000    5.000000    2.250000    0.000000   11.000000    6.125000
50%     69.500000    9.000000    5.000000    1.000000   58.000000   15.455000
75%    103.750000   14.000000   11.000000    2.000000  209.750000   26.150000
max    138.000000   17.000000   17.000000    9.000000  735.000000   75.830000

               BF          SR         100          50          4s          6s
count  138.000000  138.000000  138.000000  138.000000  138.000000  138.000000
mean   100.340580  114.951522    0.036232    0.731884   11.971014    6.318841
std    119.582882   50.099941    0.223098    1.467606   16.798046    8.855684
min      2.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%     11.250000   85.947500    0.000000    0.000000    1.000000    0.000000
50%     43.500000  124.230000    0.000000    0.000000    4.000000    2.000000
75%    159.250000  143.555000    0.000000    1.000000   17.750000    8.750000
max    516.000000  300.000000    2.000000    8.000000   68.000000   37.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
```

```
atting Stats\BATTING STATS - IPL_2019.csv:
              POS         Mat        Inns          NO        Runs         Avg  \
count  144.000000  144.000000  144.000000  144.000000  144.000000  144.000000   
mean    72.500000    8.687500    6.263889    1.534722  128.986111   17.302847   
std     41.713307    5.102404    4.997882    1.672121  166.700770   16.367185   
min      1.000000    1.000000    1.000000    0.000000    0.000000    0.000000   
25%     36.750000    4.000000    2.000000    0.000000    6.750000    3.375000   
50%     72.500000    9.000000    4.000000    1.000000   41.000000   15.165000   
75%    108.250000   13.000000   10.250000    2.000000  211.500000   28.230000   
max    144.000000   17.000000   17.000000    8.000000  692.000000   83.200000   

               BF          SR         100          50          4s          6s  
count  144.000000  144.000000  144.000000  144.000000  144.000000  144.000000  
mean    96.500000  109.617500    0.041667    0.736111   11.479167    5.444444  
std    119.138575   56.982403    0.200524    1.414145   15.899144    8.347168  
min      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000  
25%      8.000000   82.287500    0.000000    0.000000    0.000000    0.000000  
50%     37.000000  120.000000    0.000000    0.000000    2.500000    2.000000  
75%    158.000000  139.737500    0.000000    1.000000   18.250000    7.000000  
max    481.000000  333.330000    1.000000    8.000000   64.000000   52.000000  

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
atting Stats\BATTING STATS - IPL_2020.csv:
             POS         Mat        Inns          NO        Runs         Avg  \
count  133.00000  133.000000  133.000000  133.000000  133.000000  133.000000   
mean    67.00000    9.631579    6.631579    1.616541  139.157895   19.366241   
std     38.53786    4.893523    5.030925    1.550766  167.293103   18.053343   
min      1.00000    1.000000    1.000000    0.000000    0.000000    0.000000   
25%     34.00000    5.000000    2.000000    0.000000   10.000000    6.000000   
50%     67.00000   10.000000    5.000000    1.000000   59.000000   15.000000   
75%    100.00000   14.000000   11.000000    2.000000  232.000000   29.900000   
max    133.00000   17.000000   17.000000    7.000000  670.000000  101.000000   

               BF          SR         100          50          4s          6s  
count  133.000000  133.000000  133.000000  133.000000  133.000000  133.000000  
mean   105.714286  107.364737    0.037594    0.827068   11.894737    5.518797  
std    122.253870   44.584031    0.227170    1.351269   15.521375    7.393283  
min      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000  
25%     12.000000   88.750000    0.000000    0.000000    0.000000    0.000000  
50%     53.000000  116.840000    0.000000    0.000000    5.000000    2.000000  
75%    169.000000  137.500000    0.000000    1.000000   20.000000    9.000000  
max    518.000000  191.420000    2.000000    5.000000   67.000000   30.000000  

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
atting Stats\BATTING STATS - IPL_2021.csv:
              POS         Mat        Inns          NO        Runs         Avg  \
count  149.000000  149.000000  149.000000  149.000000  149.000000  149.000000   
mean    75.000000    8.536913    6.248322    1.442953  118.899329   17.265772   
std     43.156691    5.126143    4.737581    1.556948  153.332563   15.230611   
min      1.000000    1.000000    1.000000    0.000000    0.000000    0.000000   
25%     38.000000    4.000000    2.000000    0.000000   10.000000    3.000000   
50%     75.000000    8.000000    5.000000    1.000000   49.000000   14.400000   
75%    112.000000   14.000000   10.000000    2.000000  160.000000   28.550000   
max    149.000000   17.000000   17.000000    9.000000  635.000000   75.660000   

               BF          SR         100          50          4s          6s  
count  149.000000  149.000000  149.000000  149.000000  149.000000  149.000000  
mean    93.637584  105.055638    0.026846    0.597315   10.389262    4.610738  
std    113.916115   46.698607    0.162177    1.235289   15.229644    6.093540  
min      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000  
25%     11.000000   82.600000    0.000000    0.000000    0.000000    0.000000  
50%     46.000000  113.190000    0.000000    0.000000    3.000000    2.000000  
75%    140.000000  131.910000    0.000000    1.000000   13.000000    7.000000  
max    471.000000  261.110000    1.000000    6.000000   64.000000   30.000000  

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\IPL Player Stats\B
atting Stats\BATTING STATS - IPL_2022.csv:
```

```
              POS         Mat        Inns          NO        Runs          BF  \
count  162.000000  162.000000  162.000000  162.000000  162.000000  162.000000   
mean    81.500000    9.160494    7.141975    1.586420  142.296296  106.061728   
std     46.909487    5.047493    4.816982    1.819989  161.252161  114.571234   
min      1.000000    1.000000    1.000000    0.000000    1.000000    1.000000   
25%     41.250000    5.000000    3.000000    0.000000   14.500000   14.000000   
50%     81.500000    9.500000    7.000000    1.000000   65.000000   47.500000   
75%    121.750000   14.000000   11.000000    2.000000  241.500000  176.500000   
max    162.000000   17.000000   17.000000   10.000000  863.000000  579.000000   

               SR         100          50          4s          6s  
count  162.000000  162.000000  162.000000  162.000000  162.000000  
mean   120.406235    0.049383    0.679012   12.450617    6.555556  
std     45.192799    0.366322    1.172428   15.314529    8.070343  
min     16.660000    0.000000    0.000000    0.000000    0.000000  
25%     94.082500    0.000000    0.000000    1.000000    0.000000  
50%    123.620000    0.000000    0.000000    5.000000    3.000000  
75%    144.247500    0.000000    1.000000   21.000000   11.000000  
max    400.000000    4.000000    5.000000   83.000000   45.000000  

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2016.csv:
             POS        Mat       Inns         Ov        Runs       Wkts  \
count  86.000000  86.000000  86.000000  86.000000   86.000000  86.000000   
mean   43.500000   8.639535   7.872093  25.174419  204.988372   6.872093   
std    24.969982   4.591329   4.523914  16.698849  129.725275   5.523845   
min     1.000000   1.000000   1.000000   1.000000    8.000000   1.000000   
25%    22.250000   4.000000   4.000000  12.250000  103.750000   2.000000   
50%    43.500000   8.000000   7.000000  17.500000  172.000000   5.000000   
75%    64.750000  12.000000  11.000000  39.000000  307.000000  10.750000   
max    86.000000  17.000000  17.000000  66.000000  494.000000  23.000000   

              Avg       Econ          SR         4w         5w  
count   86.000000  86.000000   86.000000  86.000000  86.000000  
mean    39.708837   8.266977   28.186860   0.151163   0.011628  
std     26.674725   1.286748   17.003731   0.360308   0.107833  
min      4.000000   4.800000    5.000000   0.000000   0.000000  
25%     24.287500   7.577500   18.915000   0.000000   0.000000  
50%     32.165000   8.130000   23.000000   0.000000   0.000000  
75%     44.300000   8.982500   32.495000   0.000000   0.000000  
max    147.000000  13.200000  102.000000   1.000000   1.000000  

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2017.csv:
             POS        Mat       Inns         Ov        Runs       Wkts  \
count  90.000000  90.000000  90.000000  90.000000   90.000000  90.000000   
mean   45.500000   8.255556   7.522222  23.933333  196.444444   7.211111   
std    26.124701   4.428105   4.222007  15.788490  123.615943   5.974655   
min     1.000000   1.000000   1.000000   2.000000   15.000000   1.000000   
25%    23.250000   5.000000   4.000000  10.250000   97.500000   2.000000   
50%    45.500000   8.000000   6.000000  20.000000  176.000000   5.000000   
75%    67.750000  12.000000  12.000000  38.750000  299.500000  11.000000   
max    90.000000  17.000000  16.000000  59.000000  507.000000  26.000000   

              Avg       Econ          SR         4w         5w  
count   90.000000  90.000000   90.000000  90.000000  90.000000  
mean    37.913444   8.527333   25.874111   0.077778   0.033333  
std     31.082148   1.600797   17.647388   0.269322   0.180511  
min     11.750000   3.750000    8.000000   0.000000   0.000000  
25%     22.175000   7.565000   17.012500   0.000000   0.000000  
50%     28.900000   8.455000   21.500000   0.000000   0.000000  
75%     42.750000   9.370000   29.650000   0.000000   0.000000  
max    248.000000  14.200000  144.000000   1.000000   1.000000  

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2018.csv:
             POS        Mat      Inns          Ov        Runs       Wkts  \
count  82.000000  82.000000  82.00000  82.000000   82.000000  82.000000
mean   41.500000   9.243902   8.52439  27.536585  234.658537   8.073171
std    23.815261   4.641629   4.61138  17.437150  136.882505   5.751608
min     1.000000   2.000000   1.00000   2.000000   14.000000   1.000000
25%    21.250000   5.250000   5.00000  12.250000  118.500000   3.000000
50%    41.500000   9.000000   7.50000  25.500000  220.500000   6.000000
75%    61.750000  14.000000  12.75000  41.000000  332.750000  12.000000
max    82.000000  17.000000  17.00000  68.000000  547.000000  24.000000

              Avg       Econ         SR         4w         5w
count   82.000000  82.000000  82.000000  82.000000  82.000000
mean    34.496463   8.992683  22.900488   0.097561   0.012195
std     16.436058   1.792103   9.497781   0.403985   0.110432
min     11.000000   5.860000   9.250000   0.000000   0.000000
25%     23.495000   7.815000  16.960000   0.000000   0.000000
50%     28.080000   8.720000  21.260000   0.000000   0.000000
75%     44.420000   9.877500  27.727500   0.000000   0.000000
max    108.000000  16.950000  63.000000   3.000000   1.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2019.csv:
             POS        Mat       Inns         Ov        Runs       Wkts  \
count  87.000000  87.000000  87.000000  87.000000   87.000000  87.000000
mean   44.000000   8.218391   7.632184  25.839080  212.965517   7.344828
std    25.258662   4.767685   4.764488  18.237371  138.345493   6.296599
min     1.000000   1.000000   1.000000   2.000000   11.000000   1.000000
25%    22.500000   3.500000   3.000000   8.500000   78.000000   2.000000
50%    44.000000   8.000000   7.000000  23.000000  194.000000   5.000000
75%    65.500000  12.000000  11.500000  42.000000  339.000000  11.000000
max    87.000000  17.000000  17.000000  64.000000  482.000000  26.000000

              Avg       Econ          SR         4w         5w
count   87.000000  87.000000   87.000000  87.000000  87.000000
mean    37.652644   8.695747   25.811149   0.103448   0.011494
std     23.805384   1.557568   15.167603   0.404466   0.107211
min     11.000000   5.500000    8.660000   0.000000   0.000000
25%     23.610000   7.485000   17.025000   0.000000   0.000000
50%     31.710000   8.700000   22.000000   0.000000   0.000000
75%     44.500000   9.595000   30.000000   0.000000   0.000000
max    166.000000  13.500000  120.000000   2.000000   1.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2020.csv:
             POS        Mat       Inns         Ov        Runs       Wkts  \
count  78.000000  78.000000  78.000000  78.000000   78.000000  78.000000
mean   39.500000   9.012821   8.461538  28.820513  234.564103   8.012821
std    22.660538   4.876377   4.785509  18.286851  133.956534   6.744875
min     1.000000   1.000000   1.000000   4.000000   42.000000   1.000000
25%    20.250000   5.000000   5.000000  14.250000  133.250000   3.000000
50%    39.500000   9.000000   7.000000  25.000000  211.000000   6.000000
75%    58.750000  14.000000  13.000000  49.000000  344.000000  11.000000
max    78.000000  17.000000  17.000000  65.000000  548.000000  30.000000

              Avg       Econ         SR         4w         5w
count   78.000000  78.000000  78.000000  78.000000  78.000000
mean    40.252051   8.623077  27.818333   0.089744   0.012821
std     22.497176   1.791806  13.578477   0.367013   0.113228
min     14.960000   5.370000  10.660000   0.000000   0.000000
25%     24.055000   7.487500  18.975000   0.000000   0.000000
50%     33.000000   8.350000  23.780000   0.000000   0.000000
75%     46.150000   9.407500  33.750000   0.000000   0.000000
max    133.000000  16.000000  72.000000   2.000000   1.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2021.csv:
             POS        Mat       Inns         Ov        Runs       Wkts  \
count  89.000000  89.000000  89.000000  89.000000   89.000000  89.000000
```

```
mean   45.000000   8.078652   7.471910  25.067416  197.561798   7.516854
std    25.836021   4.848207   4.668779  18.061006  137.806349   6.528108
min     1.000000   1.000000   1.000000   2.000000   16.000000   1.000000
25%    23.000000   3.000000   3.000000  11.000000   95.000000   2.000000
50%    45.000000   8.000000   7.000000  17.000000  139.000000   5.000000
75%    67.000000  12.000000  11.000000  38.000000  328.000000  12.000000
max    89.000000  17.000000  17.000000  68.000000  527.000000  32.000000

             Avg       Econ         SR         4w         5w
count  89.000000  89.000000  89.000000  89.000000  89.000000
mean   33.754045   8.087303  24.810787   0.112360   0.033708
std    19.152758   1.349384  13.241954   0.351562   0.181499
min     8.000000   4.000000  10.360000   0.000000   0.000000
25%    20.400000   7.250000  16.630000   0.000000   0.000000
50%    29.250000   8.080000  22.000000   0.000000   0.000000
75%    42.000000   9.000000  28.000000   0.000000   0.000000
max    98.000000  11.830000  72.000000   2.000000   1.000000

Summary statistics for numerical columns C:\Users\ggulz\Downloads\archive\BOWLING STATS - IP
L_2022.csv:
              POS         Mat        Inns          Ov        Runs        Wkts  \
count  103.000000  103.000000  103.000000  103.000000  103.000000  103.000000
mean    52.000000    8.815534    7.941748   27.166019  227.213592    8.242718
std     29.877528    4.872386    4.764709   18.447714  146.147498    6.484470
min      1.000000    1.000000    1.000000    0.300000    8.000000    1.000000
25%     26.500000    5.000000    4.000000   11.500000  100.500000    3.000000
50%     52.000000    9.000000    7.000000   23.000000  210.000000    6.000000
75%     77.500000   13.000000   12.000000   43.000000  349.500000   12.500000
max    103.000000   17.000000   17.000000   68.000000  551.000000   27.000000

              Avg        Econ          SR          4w          5w
count  103.000000  103.000000  103.000000  103.000000  103.000000
mean    33.685437    8.821262   23.226117    0.203883    0.038835
std     15.984617    1.698188   11.030324    0.450691    0.194146
min      5.500000    5.500000    3.000000    0.000000    0.000000
25%     21.705000    7.660000   15.740000    0.000000    0.000000
50%     29.620000    8.530000   20.150000    0.000000    0.000000
75%     40.415000    9.655000   27.000000    0.000000    0.000000
max     84.000000   16.000000   66.000000    2.000000    1.000000
```

For Columns name is used for And their Alliases of short form for Bowlers

a dictionary named column_aliases_of_Bowler,This dictionary contains column names found in a dataset of bowler statistics to their corresponding descriptive names (values).

for loop iterates through each key-value pair in the column_aliases_of_Bowler dictionary.

it prints the key (column name) followed by a colon and then the value (alias/descriptive name) for that column.

The loop also adds a comma (,) after each key-value pair, which means the output will have line breaks after each comma creating a more readable and organized The loop iterates over all the key-value pairs in the dictionary, and each pair is printed as a separate line.

In [22]:
```
column_aliases_of_Bowler = {
    'POS': "Player's rank based on most wickets",
    'Player': "Player's name",
    'Mat': "Matches played",
    'Inns': "Innings Played",
    'Ov': "Overs",
    'Runs': "Total runs given by bowler",
    'Wkts': "Total Wickets taken",
    'BBI': "Best Bowling in Innings",
    'Avg': "Average",
    'Econ': "Economy",
    'SR': "Strike Rate",
    '4w': "4 wickets haul",
    '5w': "5 wickets haul"
}
print("Bowlers'column Name Aliases are :\n\n")
# Print the dictionary with line breaks after each comma
for key, value in column_aliases_of_Bowler.items():
    print(f"{key}: {value},")
```

```
Bowlers'column Name Aliases are :


POS: Player's rank based on most wickets,
Player: Player's name,
Mat: Matches played,
Inns: Innings Played,
Ov: Overs,
Runs: Total runs given by bowler,
Wkts: Total Wickets taken,
BBI: Best Bowling in Innings,
Avg: Average,
Econ: Economy,
SR: Strike Rate,
4w: 4 wickets haul,
5w: 5 wickets haul,
```

Alliases and their uses in dataframe of Batsaman

the use case line of code also same as bowler but the common aliases are difenrent in the batsman dataset so the aliases are diferently declared for it but writting of code is same

In [23]:
```
column_aliases_of_batsman={
'POS': "Player's rank based on most runs",
'Player' : "Player's name",
'Mat' : "Matches played",
'Inns' : "Innings Played",
'NO' : "Number of Not Out in innings",
'Runs' : "Total Runs scored by a player",
'HS' : "Highest Score in innings [* -- Not Out in that Innings]",
'Avg' : "Average",
'BF' : "Bowls faced",
'SR' : "Strike Rate",
'100' : "No of times 100 scored",
'50' : "No of the times 50 scored",
'4s' : "Total Fours Scored",
'6s' : "Total Sixes Scored"
}

print("Batsman's column Name Aliases are :\n\n")
# Print the dictionary with line breaks after each comma
for key, value in column_aliases_of_batsman.items():
    print(f"{key}: {value},")
```

```
Batsman's column Name Aliases are :


POS: Player's rank based on most runs,
Player: Player's name,
Mat: Matches played,
Inns: Innings Played,
NO: Number of Not Out in innings,
Runs: Total Runs scored by a player,
HS: Highest Score in innings [* -- Not Out in that Innings],
Avg: Average,
BF: Bowls faced,
SR: Strike Rate,
100: No of times 100 scored,
50: No of the times 50 scored,
4s: Total Fours Scored,
6s: Total Sixes Scored,
```

first 5 rows of data frame from batsman files

inn this section of code i defined a function for batsman dataset and store or give all batsaman dataset path and seprated of all path giving(,)in the different paths btw and the creating a enpty dataframe for storing new data that extracted from dataset

then prints a message indicating the data for which IPL season is being displayed. It extracts the year from the file name using string manipulation (splitting the file name and selecting the last part)to display the ipl season.

After that, it prints the first 5 rows of the DataFrame using df_season.head(5). This gives a preview of the data for the corresponding IPL season.

After the loop completes, it concatenates all the DataFrames in the data_frames_Batsman list into a single DataFrame named combined_data. The pd.concat function is used to concatenate the DataFrames along the rows, effectively combining all the data into a single DataFrame.

after the loop, the variable combined_data contains the combined data of all IPL seasons from 2016 to 2022 for batsman statistics.

In [24]:
```
dfs_of_Batsman = [r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2017.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2018.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2019.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2020.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2021.
r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BATTING STATS - IPL_2022.

data_frames_Batsman = []

for dfs_of_Batsman in dfs_of_Batsman:
    df_season = pd.read_csv(dfs_of_Batsman)
    print(f"\nData for {dfs_of_Batsman.split('_')[-1].split('.')[0]} IPL Season:")
    print(df_season.head(5))
    data_frames_Batsman.append(df_season)

combined_data = pd.concat(data_frames_Batsman)
```

```
Data for 2016 IPL Season:
   POS           Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  \
0    1      Virat Kohli   16    16   4   973   113  81.08  640  152.03    4   
1    2     David Warner   17    17   3   848   93*  60.57  560  151.42    0   
2    3  AB de Villiers   16    16   3   687  129*  52.84  407  168.79    1   
3    4  Gautam Gambhir   15    15   2   501   90*  38.53  411  121.89    0   
4    5  Shikhar Dhawan   17    17   4   501   82*  38.53  429  116.78    0   

   50  4s  6s  
0   7  83  38  
1   9  88  31  
2   6  57  37  
3   5  54   6  
4   4  51   8  

Data for 2017 IPL Season:
   POS           Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  50  \
0    1     David Warner   14    14   3   641  126  58.27  452  141.81    1   4   
1    2  Gautam Gambhir   16    16   4   498  76*  41.50  389  128.02    0   4   
2    3  Shikhar Dhawan   14    14   1   479   77  36.84  376  127.39    0   3   
3    4      Steve Smith   15    15   3   472  84*  39.33  387  121.96    0   3   
4    5     Suresh Raina   14    14   3   442   84  40.18  307  143.97    0   3   

   4s  6s  
0  63  26  
1  61   7  
2  53   9  
3  38  12  
4  42  13  

Data for 2018 IPL Season:
   POS            Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  \
0    1  Kane Williamson   17    17   3   735    84  52.50  516  142.44    0   
1    2      Rishabh Pant   14    14   1   684  128*  52.61  394  173.60    1   
2    3          KL Rahul   14    14   2   659   95*  54.91  416  158.41    0   
3    4     Ambati Rayudu   16    16   2   602  100*  43.00  402  149.75    1   
4    5      Shane Watson   15    15   1   555  117*  39.64  359  154.59    2   

   50  4s  6s  
0   8  64  28  
1   5  68  37  
2   6  66  32  
3   3  53  34  
4   2  44  35  

Data for 2019 IPL Season:
   POS            Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  \
0    1      David Warner   12    12   2   692  100*  69.20  481  143.86    1   
1    2          KL Rahul   14    14   3   593  100*  53.90  438  135.38    1   
2    3  Quinton de Kock   16    16   1   529    81  35.26  398  132.91    0   
3    4   Shikhar Dhawan   16    16   1   521   97*  34.73  384  135.67    0   
4    5    Andre Russell   14    13   4   510   80*  56.66  249  204.81    0   

   50  4s  6s  
0   8  57  21  
1   6  49  25  
2   4  45  25  
3   5  64  11  
4   4  31  52  

Data for 2020 IPL Season:
   POS           Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  \
0    1         KL Rahul   14    14   2   670  132*  55.83  518  129.34    1   
1    2  Shikhar Dhawan   17    17   3   618  106*  44.14  427  144.73    2   
2    3    David Warner   16    16   2   548   85*  39.14  407  134.64    0   
3    4     Shreyas Iyer   17    17   2   519   88*  34.60  421  123.27    0   
4    5     Ishan Kishan   14    13   4   516    99  57.33  354  145.76    0   
```

```
   50  4s  6s
0   5  58  23
1   4  67  12
2   4  52  14
3   3  40  16
4   4  36  30

Data for 2021 IPL Season:
   POS           Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  \
0    1  Ruturaj Gaikwad   16    16   2   635  101*  45.35  466  136.26    1
1    2    Faf du Plessis   16    16   2   633   95*  45.21  458  138.20    0
2    3          KL Rahul   13    13   3   626   98*  62.60  451  138.80    0
3    4    Shikhar Dhawan   16    16   1   587    92  39.13  471  124.62    0
4    5     Glenn Maxwell   15    14   2   513    78  42.75  356  144.10    0

   50  4s  6s
0   4  64  23
1   6  60  23
2   6  48  30
3   3  63  16
4   6  48  21

Data for 2022 IPL Season:
   POS           Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  \
0    1      Jos Buttler   17    17   2   863   116  57.53  579  149.05    4
1    2        K L Rahul   15    15   3   616  103*  51.33  455  135.38    2
2    3  Quinton De Kock   15    15   1   508  140*  36.29  341  148.97    1
3    4    Hardik Pandya   15    15   4   487   87*  44.27  371  131.26    0
4    5     Shubman Gill   16    16   2   483    96   34.5  365  132.32    0

   50  4s  6s
0   4  83  45
1   4  45  30
2   3  47  23
3   4  49  12
4   4  51  11
```

first five rows of bowler from dataframe

this section of coding is also same like above but the use case of this to print first five row of dataset from each datafile and the codding lline is also same like above because using to print first five row of data

In [25]:
```python
dfs_of_Bowler = [r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv"]
data_frames_Batsman = []

for dfs_of_Bowler in dfs_of_Bowler:
    df_season_Bowling = pd.read_csv(dfs_of_Bowler)
    print(f"\nData for {dfs_of_Bowler.split('_')[-1].split('.')[0]} IPL Season:")
    print(df_season_Bowling.head(5))
    data_frames_Batsman.append(df_season_Bowling)

combined_data = pd.concat(data_frames_Batsman)
```

```
Data for 2016 IPL Season:
   POS                Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ  \
0    1     Bhuvneshwar Kumar   17    17  66   490    23  5/19  21.30  7.42   
1    2      Yuzvendra Chahal   13    13  49   401    21  4/25  19.09  8.15   
2    3          Shane Watson   16    16  56   485    20  4/29  24.25  8.58   
3    4       Dhawal Kulkarni   14    14  49   364    18  4/14  20.22  7.42   
4    5  Mitchell McClenaghan   14    14  53   436    17  4/21  25.64  8.17   

      SR  4w  5w  
0  17.21   1   0  
1  14.04   1   0  
2  16.95   1   0  
3  16.33   1   0  
4  18.82   1   0  

Data for 2017 IPL Season:
   POS                Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ  \
0    1     Bhuvneshwar Kumar   14    14  52   369    26  5/19  14.19  7.05   
1    2        Jaydev Unadkat   12    12  46   322    24  5/25  13.41  7.02   
2    3        Jasprit Bumrah   16    16  59   439    20  4/14  21.95  7.39   
3    4  Mitchell McClenaghan   14    14  54   507    19  4/21  26.68  9.38   
4    5           Imran Tahir   12    12  47   369    18  4/12  20.50  7.85   

      SR  4w  5w  
0  12.07   0   1  
1  11.45   0   1  
2  17.80   0   0  
3  17.05   0   0  
4  15.66   0   0  

Data for 2018 IPL Season:
   POS         Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ     SR  \
0    1     Andrew Tye   14    14  56   448    24  5/17  18.66  8.00  14.00   
1    2  Siddarth Kaul   17    17  66   547    21  4/29  26.04  8.28  18.85   
2    3    Rashid Khan   17    17  68   458    21  3/19  21.80  6.73  19.42   
3    4    Umesh Yadav   14    14  53   418    20  4/24  20.90  7.86  15.95   
4    5  Hardik Pandya   13    13  42   381    18  3/20  21.16  8.92  14.22   

   4w  5w  
0   3   0  
1   0   0  
2   0   0  
3   0   0  
4   0   0  

Data for 2019 IPL Season:
   POS          Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ     SR  \
0    1     Imran Tahir   17    17  64   431    26  4/12  16.57  6.69  14.84   
1    2   Kagiso Rabada   12    12  47   368    25  4/21  14.72  7.82  11.28   
2    3   Deepak Chahar   17    17  64   482    22  4/13  21.90  7.47  17.59   
3    4   Shreyas Gopal   14    14  48   347    20  4/16  17.35  7.22  14.40   
4    5  Mohammad Shami   14    14  54   469    19  3/15  24.68  8.68  17.05   

   4w  5w  
0   2   0  
1   2   0  
2   0   0  
3   0   0  
4   0   0  

Data for 2020 IPL Season:
   POS            Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ     SR  \
0    1     Kagiso Rabada   17    17  65   548    30  4/21  18.26  8.34  13.13   
1    2    Jasprit Bumrah   15    15  60   404    27  4/14  14.96  6.73  13.33   
2    3       Trent Boult   15    15  57   457    25  4/18  18.28  7.97  13.76   
3    4     Anrich Nortje   16    16  61   512    22  3/33  23.27  8.39  16.63   
4    5  Yuzvendra Chahal   15    15  57   405    21  4/25  19.28  7.08  16.33   
```

```
   4w  5w
0   2   0
1   2   0
2   1   0
3   0   0
4   0   0

Data for 2021 IPL Season:
   POS          Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ     SR  \
0    1   Harshal Patel   15    15  56   459    32  5/27  14.34  8.14  10.56   
1    2      Avesh Khan   16    16  61   450    24  3/13  18.75  7.37  15.25   
2    3  Jasprit Bumrah   14    14  55   410    21  4/14  19.52  7.45  15.71   
3    4  Shardul Thakur   16    16  60   527    21  3/19  25.09  8.80  17.09   
4    5  Mohammad Shami   14    14  52   395    19  3/15  20.78  7.50  16.63   

   4w  5w
0   1   1
1   0   0
2   0   0
3   0   0
4   0   0

Data for 2022 IPL Season:
   POS             Player  Mat  Inns    Ov  Runs  Wkts   BBI    Avg  Econ  \
0    1   Yuzvendra Chahal   17    17  68.0   527    27  40/5  19.51  7.75   
1    2  Wanindu Hasaranga   16    16  57.0   430    26  18/5  16.53  7.54   
2    3      Kagiso Rabada   13    13  48.0   406    23  33/4  17.65  8.45   
3    4        Umran Malik   14    14  49.1   444    22  25/5  20.18  9.03   
4    5      Kuldeep Yadav   14    14  49.4   419    21  14/4  19.95  8.43   

      SR  4w  5w
0  15.11   1   1
1  13.15   1   1
2  12.52   2   0
3  13.40   1   1
4  14.19   2   0
```

Top 10 performing player in each role batsman and as bowler

in this section of code i print top perfroming players in a particular year

here i defines a function called load_csv that takes a file_path as an argument and tries to read the CSV file using pd.read_csv(file_path)

If the file is found, it returns the DataFrame read from the CSV. If the file is not found, it prints a message indicating that and returns None

and then defines a function called find_best_player that takes a DataFrame df, a metric (performance metric) to evaluate players, and an optional num_players argument (default value is 1) to specify the number of top players to be returned. The function sorts the DataFrame based on the specified metric in descending order and returns the top num_players players.

defines the main function form_new_team, which takes the year, batting_file_path, and bowling_file_path as arguments it loads the batting and bowling CSV files using the load_csv function.

It checks if either the batting or bowling DataFrame is None, which would indicate that there was an issue loading the CSV files, If so, the function returns without proceeding further

The function then proceeds to find the best batsman and best bowler for the new team based on their batting and bowling averages using the find_best_player function.

After finding the best batsman and best bowler, it prints the results for the given year, including the best batsman's information and the best bowler's information.

In the if **name** == "**main**": block, the code creates a list years_list containing the years from 2016 to 2022

It creates a dictionary file_paths_by_year, where each key represents a year, and the value is another dictionary containing the file paths for the corresponding batting and bowling CSV files.

The code then iterates through each year in years_list, gets the corresponding batting and bowling file paths from the file_paths_by_year dictionary, and calls the form_new_team function for each year, displaying the best batsman and best bowler for each IPL season.

In [26]:
```
# Load CSV files
def load_csv(file_path):
    try:
        return pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"File not found: {file_path}")
        return None

# Find the best player for each role based on performance metrics
def find_best_player(df, metric, num_players=1):
    sorted_df = df.sort_values(by=[metric], ascending=False)
    return sorted_df.head(num_players)

# Main function to form a new team
def form_new_team(year, batting_file_path, bowling_file_path):
    batting_df = load_csv(batting_file_path)
    bowling_df = load_csv(bowling_file_path)

    if batting_df is None or bowling_df is None:
        return

    # Finding the best batsman and best bowler for the new team
    best_batsman = find_best_player(batting_df, 'Avg', num_players=1)
    best_bowler = find_best_player(bowling_df, 'Avg', num_players=1)

    # Printing the results
    print(f"Year: {year}")
    print("Best Batsman:")
    print(best_batsman)
    print("Best Bowler:")
    print(best_bowler)

if __name__ == "__main__":
    years_list = [2016, 2017, 2018, 2019, 2020, 2021, 2022]
    file_paths_by_year = {
        2016: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
        },
        2017: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
        },
        2018: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
        },
        2019: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
        },
        2020: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
        },
        2021: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
        },
        2022: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv",
        },

    }

    for year in years_list:
        batting_file_path = file_paths_by_year[year]['batting']
        bowling_file_path = file_paths_by_year[year]['bowling']
        form_new_team(year, batting_file_path, bowling_file_path)
```

```
Year: 2016
Best Batsman:
   POS       Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  50  \
0    1  Virat Kohli   16    16   4   973  113  81.08  640  152.03    4   7

   4s  6s
0  83  38
Best Bowler:
    POS         Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ    SR  \
85   86  Harshal Patel    5     5  14   147     1  5/27  147.0  10.5  84.0

    4w  5w
85   0   0
Year: 2017
Best Batsman:
   POS       Player  Mat  Inns  NO  Runs    HS   Avg   BF      SR  100  50  \
5    6  Hashim Amla   10    10   3   420  104*  60.0  288  145.83    2   2

   4s  6s
5  40  17
Best Bowler:
    POS            Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg   Econ  \
89   90  Moises Henriques   12    12  24   248     1  3/12  248.0  10.33

       SR  4w  5w
89  144.0   0   0
Year: 2018
Best Batsman:
    POS    Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  50  4s  \
12   13  MS Dhoni   16    15   9   455  79*  75.83  302  150.66    0   3  24

    6s
12  30
Best Bowler:
    POS            Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg   Econ  \
72   73  Mitchell Johnson    6     6  21   216     2  3/26  108.0  10.28

      SR  4w  5w
72  63.0   0   0
Year: 2019
Best Batsman:
    POS    Player  Mat  Inns  NO  Runs   HS   Avg   BF      SR  100  50  4s  \
12   13  MS Dhoni   15    12   7   416  84*  83.2  309  134.62    0   3  22

    6s
12  23
Best Bowler:
    POS              Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ  \
77   78  Krishnappa Gowtham    7     7  20   166     1  2/12  166.0   8.3

       SR  4w  5w
77  120.0   0   0
Year: 2020
Best Batsman:
    POS        Player  Mat  Inns  NO  Runs   HS    Avg  BF      SR  100  50  \
57   58  Deepak Hooda    7     5   4   101  62*  101.0  71  142.25    0   1

    4s  6s
57   5   5
Best Bowler:
    POS      Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ    SR  4w  \
67   68  Dale Steyn    3     3  11   133     1  3/11  133.0  11.4  70.0   0

    5w
67   0
Year: 2021
Best Batsman:
    POS           Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  \
30   31  Ravindra Jadeja   16    12   9   227  62*  75.66  156  145.51    0   

    50  4s  6s  
30   1  19   9  
Best Bowler:
    POS        Player  Mat  Inns  Ov  Runs  Wkts   BBI   Avg  Econ    SR  4w  \
88   89  Nathan Ellis    3     3  12    98     1  1/20  98.0  8.16  72.0   0   

    5w  
88   0  
Year: 2022
Best Batsman:
    POS        Player  Mat  Inns  NO  Runs  HS   Avg  BF      SR  100  50  4s  \
84   85  Jason Holder   12     8   2    58  16  9.67  44  131.81    0   0   2   

    6s  
84   6  
Best Bowler:
    POS          Player  Mat  Inns    Ov  Runs  Wkts   BBI   Avg  Econ    SR  \
72   73  Shahbaz Ahmed   16    14  35.0   336     4  26/2  84.0   9.6  52.5   

    4w  5w  
72   0   0  
```

uses of useful columns instead of droping or deleteing columns

in this section i use usefull columns and ignore those columns that aren't neccessary and print after sorting the players according to performance .

in this code defines a function called find_best_player that takes a DataFrame df, a metric (performance metric) to evaluate players, an optional num_players argument (default value is 1) to specify the number of top players to be returned, and a batting argument (default value is True) to indicate whether to find the best batsman or bowler. Depending on the value of the batting argument, it selects different columns as required (either batting columns or bowling columns).

Inside the find_best_player function, it sorts the DataFrame based on the specified metric in descending order using df.sort_values(by=[metric], ascending=False)

then checks whether batting is True. If True, it selects the required columns for the best batsman and stores them in the required_columns list. Otherwise, it selects the required columns for the best bowler and stores them in the required_columns list

The function returns the top num_players players along with the selected required columns using sorted_df.head(num_players)[required_columns].

defines the main function form_new_team, which takes the year, batting_file_path, and bowling_file_path as arguments. Inside the function, it loads the batting and bowling CSV files using the load_csv function

In [27]:
```python
def load_csv(file_path):
    try:
        return pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"File not found: {file_path}")
        return None

# Find the best player for each role based on performance metrics
```

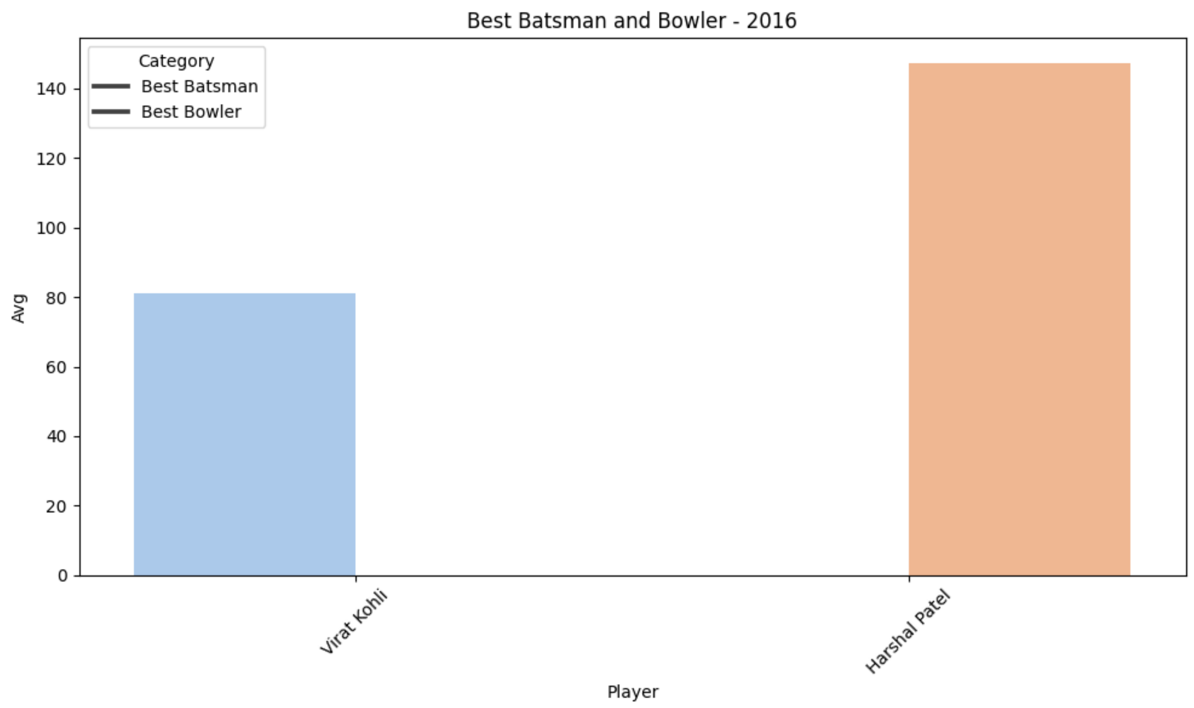
```
def find_best_player(df, metric, num_players=1, batting=True):
    sorted_df = df.sort_values(by=[metric], ascending=False)
    if batting:
        required_columns = ['Player', 'HS', 'Avg', 'SR', 'Runs']
    else:
        required_columns = ['Player', 'BBI', 'Avg', 'Econ', 'SR', 'Wkts']
    return sorted_df.head(num_players)[required_columns]

# Main function to form a new team
def form_new_team(year, batting_file_path, bowling_file_path):
    batting_df = load_csv(batting_file_path)
    bowling_df = load_csv(bowling_file_path)

    if batting_df is None or bowling_df is None:
        return

    # Finding the best batsman and best bowler for the new team
    best_batsman = find_best_player(batting_df, 'Avg', num_players=1, batting=True)
    best_bowler = find_best_player(bowling_df, 'Avg', num_players=1, batting=False)

    # Printing the results
    print(f"Year: {year}")
    print("Best Batsman:")
    print(best_batsman)
    print("\nBest Bowler:")
    print(best_bowler)

if __name__ == "__main__":
    years_list = [2016, 2017, 2018, 2019, 2020, 2021, 2022]
    file_paths_by_year = {
        2016: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
        },
        2017: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
        },
        2018: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
        },
        2019: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
        },
        2020: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
        },
        2021: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
        },
        2022: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv",
        },

    }

    for year in years_list:
        batting_file_path = file_paths_by_year[year]['batting']
        bowling_file_path = file_paths_by_year[year]['bowling']
        form_new_team(year, batting_file_path, bowling_file_path)
```

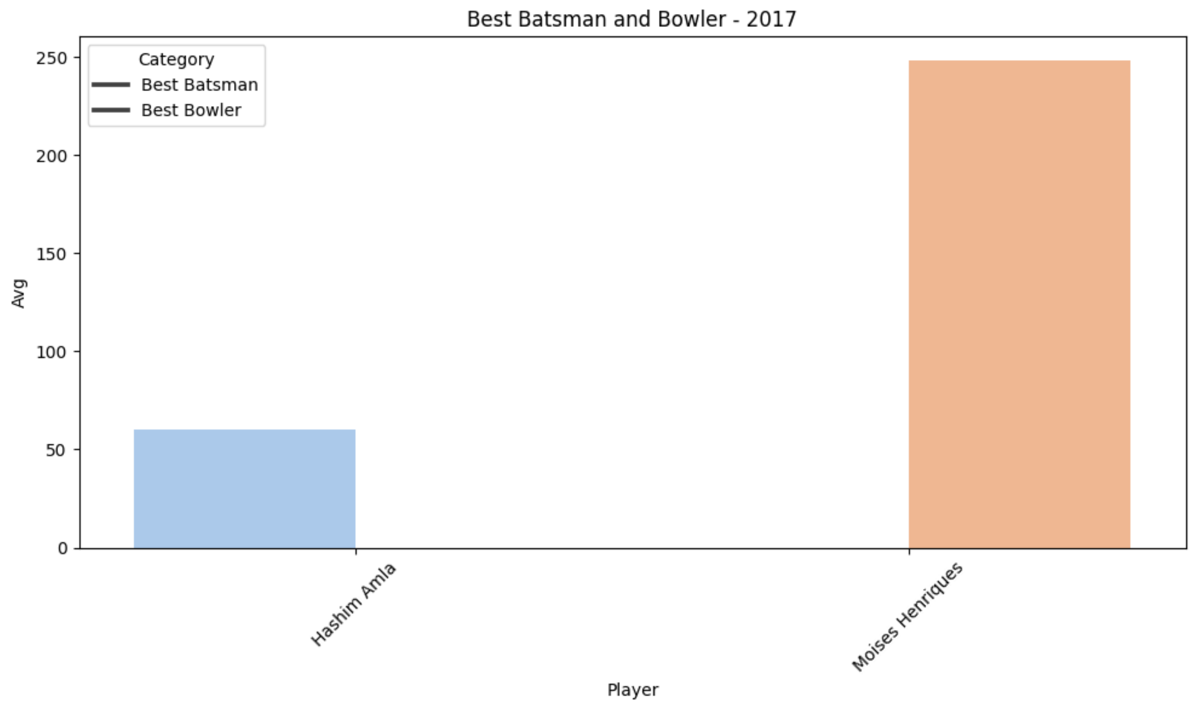
```
Year: 2016
Best Batsman:
        Player   HS    Avg      SR  Runs
0  Virat Kohli  113  81.08  152.03   973

Best Bowler:
           Player   BBI    Avg  Econ    SR  Wkts
85  Harshal Patel  5/27  147.0  10.5  84.0     1
Year: 2017
Best Batsman:
        Player    HS   Avg      SR  Runs
5  Hashim Amla  104*  60.0  145.83   420

Best Bowler:
              Player   BBI    Avg   Econ     SR  Wkts
89  Moises Henriques  3/12  248.0  10.33  144.0     1
Year: 2018
Best Batsman:
      Player   HS    Avg      SR  Runs
12  MS Dhoni  79*  75.83  150.66   455

Best Bowler:
              Player   BBI    Avg   Econ    SR  Wkts
72  Mitchell Johnson  3/26  108.0  10.28  63.0     2
Year: 2019
Best Batsman:
      Player   HS   Avg      SR  Runs
12  MS Dhoni  84*  83.2  134.62   416

Best Bowler:
                Player   BBI    Avg  Econ     SR  Wkts
77  Krishnappa Gowtham  2/12  166.0   8.3  120.0     1
Year: 2020
Best Batsman:
          Player   HS    Avg      SR  Runs
57  Deepak Hooda  62*  101.0  142.25   101

Best Bowler:
        Player   BBI    Avg  Econ    SR  Wkts
67  Dale Steyn  3/11  133.0  11.4  70.0     1
Year: 2021
Best Batsman:
             Player   HS    Avg      SR  Runs
30  Ravindra Jadeja  62*  75.66  145.51   227

Best Bowler:
          Player   BBI   Avg  Econ    SR  Wkts
88  Nathan Ellis  1/20  98.0  8.16  72.0     1
Year: 2022
Best Batsman:
          Player  HS   Avg      SR  Runs
84  Jason Holder  16  9.67  131.81    58

Best Bowler:
           Player   BBI   Avg  Econ    SR  Wkts
72  Shahbaz Ahmed  26/2  84.0   9.6  52.5     4
```

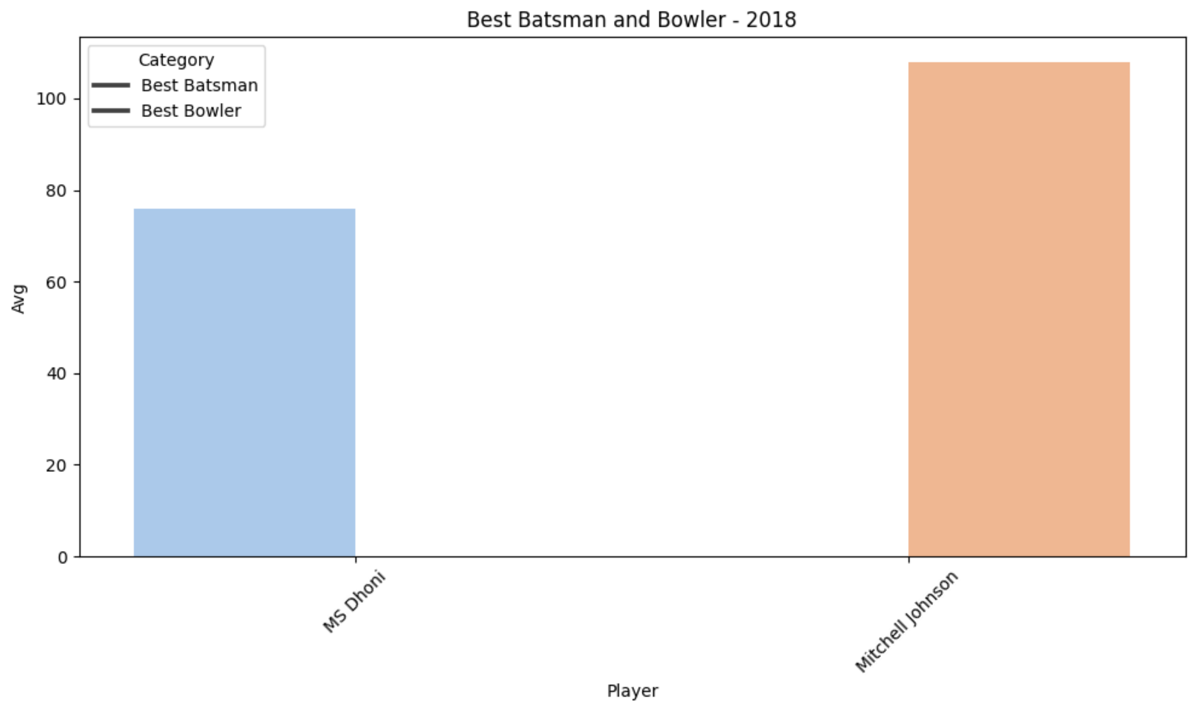
Visualizing player top 10 batsman and bowlers Performance

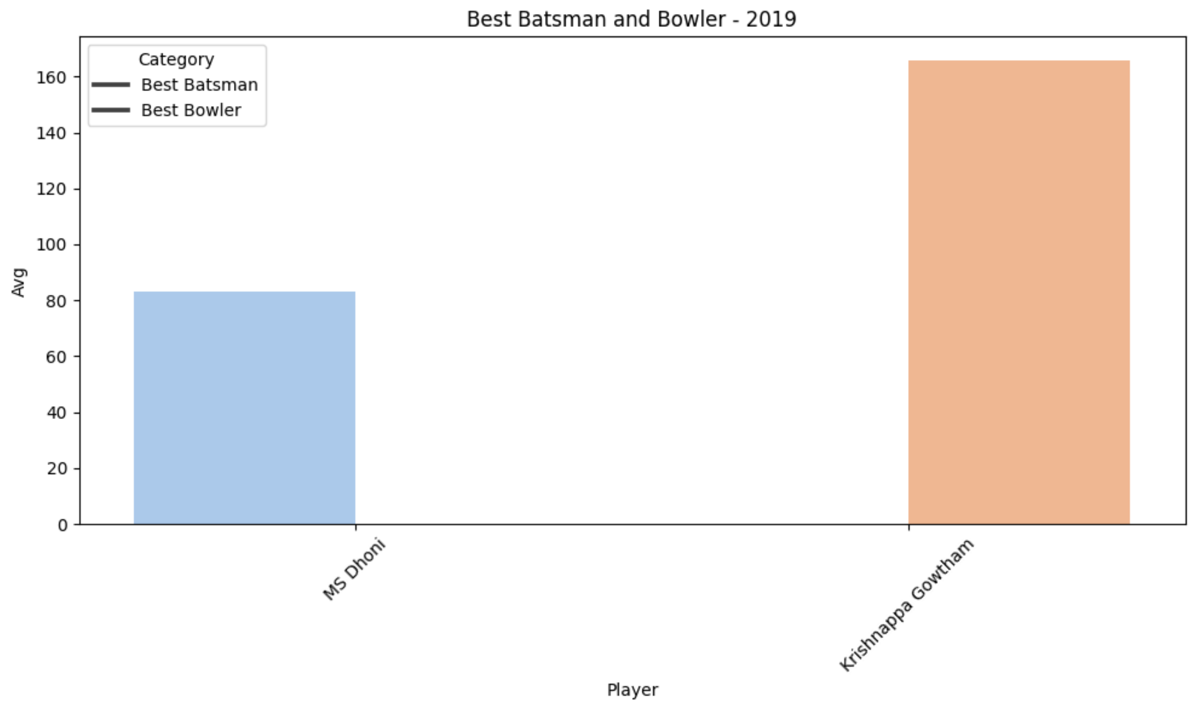
in this section of code i visualize the players performance using bar graph it will show year wise player batsman and bowlers bar in a single graph and so on for all players

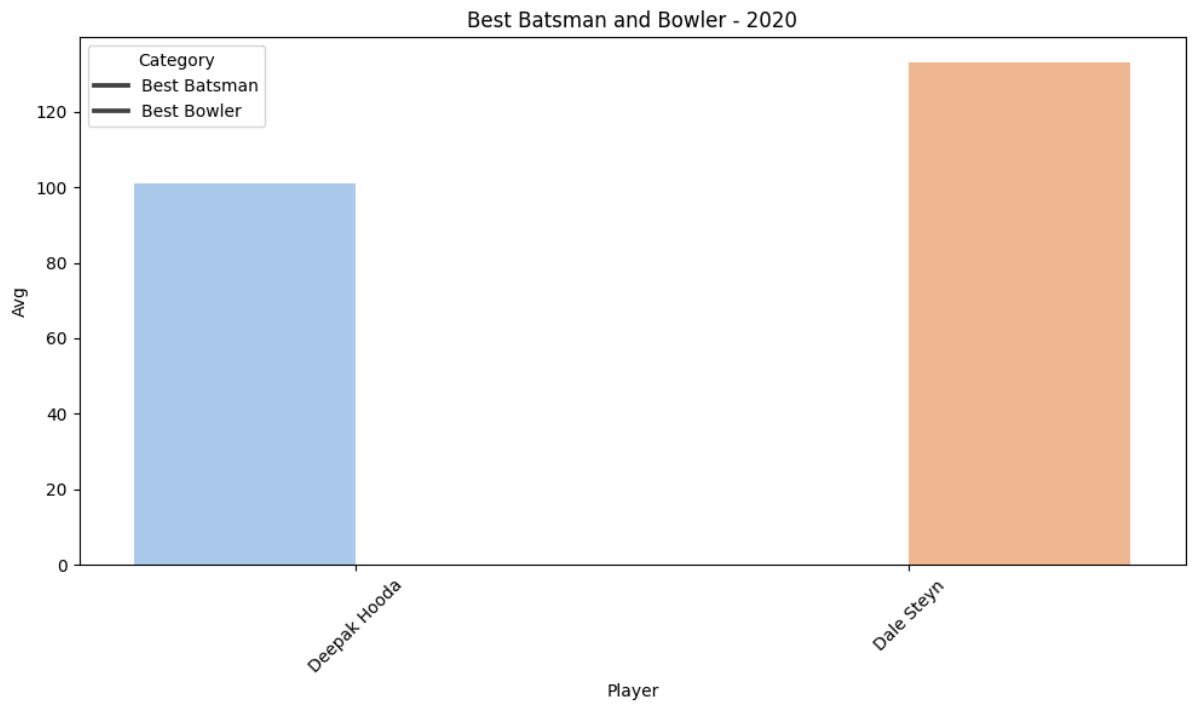
In [28]:
```
def load_csv(file_path):
    try:
        return pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"File not found: {file_path}")
        return None

def find_best_player(df, metric, num_players=1):
    return df.sort_values(by=[metric], ascending=False).head(num_players)

def form_new_team(year, batting_file_path, bowling_file_path):
    batting_df = load_csv(batting_file_path)
    bowling_df = load_csv(bowling_file_path)

    if batting_df is None or bowling_df is None:
        return

    best_batsman = find_best_player(batting_df, 'Avg', num_players=1)
    best_bowler = find_best_player(bowling_df, 'Avg', num_players=1)

    best_batsman['Avg'] = pd.to_numeric(best_batsman['Avg'])
    best_bowler['Avg'] = pd.to_numeric(best_bowler['Avg'])

    # Concatenate the DataFrames
    best_players = pd.concat([best_batsman, best_bowler])

    plt.figure(figsize=(10, 6))
    sns.barplot(data=best_players, x='Player', y='Avg', hue='POS', palette='pastel')
    plt.title(f"Best Batsman and Bowler - {year}")
    plt.xticks(rotation=45)
    plt.legend(title='Category', loc='upper left', labels=["Best Batsman", "Best Bowler"])

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    years_list = [2016, 2017, 2018, 2019, 2020, 2021, 2022]
    file_paths_by_year = {
        2016: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
        },
        2017: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
        },
        2018: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
        },
        2019: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
        },
        2020: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
        },
        2021: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
        },
        2022: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv",
        },
    }


    for year in years_list:
        form_new_team(year, file_paths_by_year[year]['batting'], file_paths_by_year[year][
```

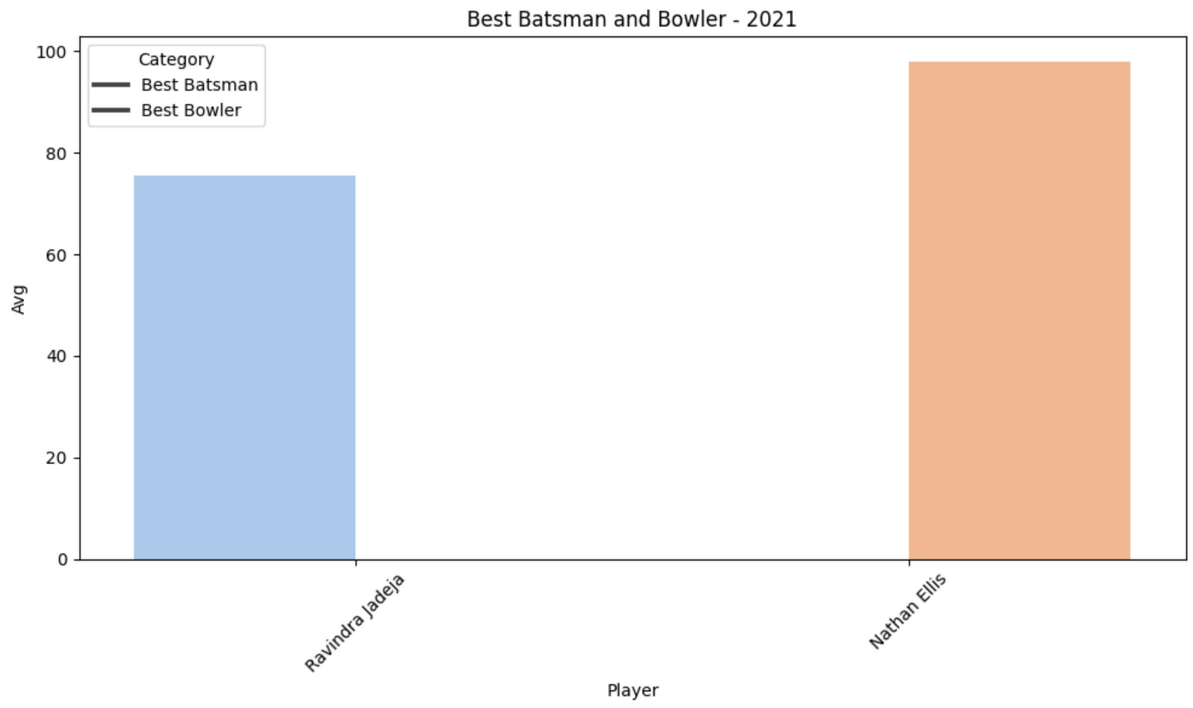


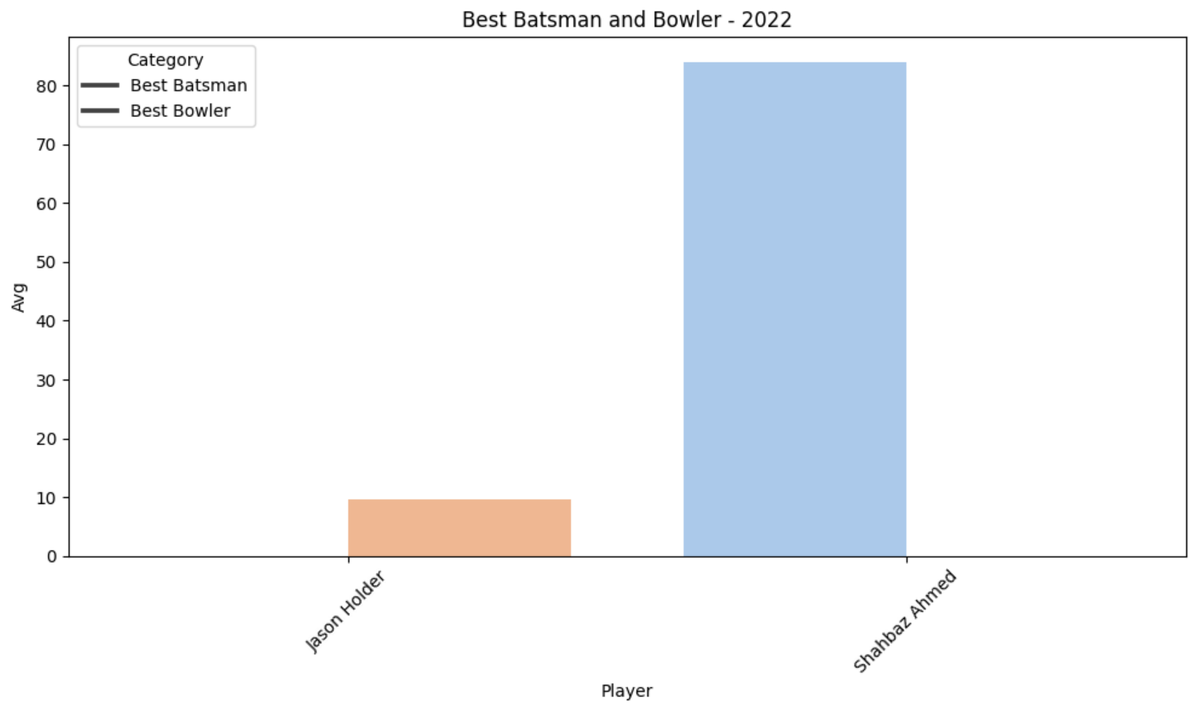



Best Batsman and Bowler - 2018
Category
Best Batsman
Best Bowler
Avg
0
20
40
60
80
100
MS Dhoni
Mitchell Johnson
Player


Best Batsman and Bowler - 2019
Category
Best Batsman
Best Bowler
Avg
0
20
40
60
80
100
120
140
160
MS Dhoni
Krishnappa Gowtham
Player


Best Batsman and Bowler - 2020
Category
Best Batsman
Best Bowler
Avg
0
20
40
60
80
100
120
Deepak Hooda
Dale Steyn
Player

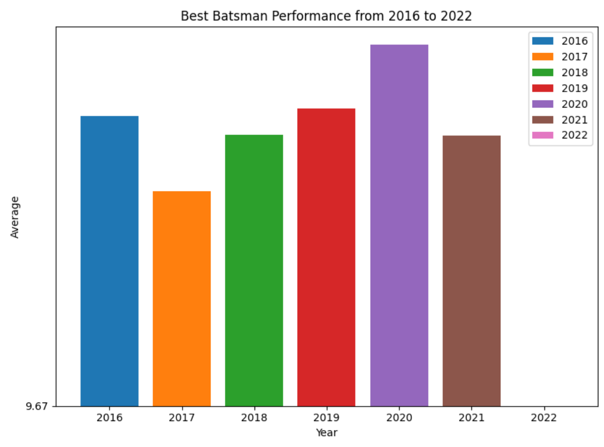

Best Batsman and Bowler - 2021
Category
Best Batsman
Best Bowler
Avg
0
20
40
60
80
100
Ravindra Jadeja
Nathan Ellis
Player

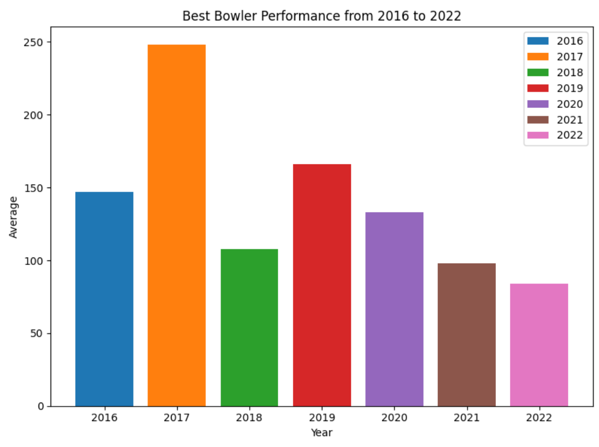


Visualizing player performance according to perfromance stats

in this section of code i visualize all players statics by their performance matrices it helps to compare to those players that played before visualization are of yearwise and the stats of bowler and batsman are seprated

In [29]:
```
# Load CSV files
def load_csv(file_path):
    try:
        return pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"File not found: {file_path}")
        return None

# Find the best player for each role based on performance metrics
def find_best_player(df, metric):
    best_player = df[df[metric] == df[metric].max()]
    return best_player

# Main function to form a new team
def form_new_team(year, batting_file_path, bowling_file_path):
    batting_df = load_csv(batting_file_path)
    bowling_df = load_csv(bowling_file_path)

    if batting_df is None or bowling_df is None:
        return

    # Finding the best batsman and best bowler for the new team
    best_batsman = find_best_player(batting_df, 'Avg')
    best_bowler = find_best_player(bowling_df, 'Avg')

    return best_batsman, best_bowler

if __name__ == "__main__":
    years_list = [2016, 2017, 2018, 2019, 2020, 2021, 2022]
    file_paths_by_year = {
        2016: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
        },
        2017: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
        },
        2018: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
        },
        2019: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
        },
        2020: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
        },
        2021: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
        },
        2022: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv",
        },
    }

    # Store the best batsmen and bowlers along with their performance stats
    best_batsmen_stats = []
    best_bowlers_stats = []
    for year in years_list:
        batting_file_path = file_paths_by_year[year]['batting']
        bowling_file_path = file_paths_by_year[year]['bowling']
        best_batsman, best_bowler = form_new_team(year, batting_file_path, bowling_file_pat
        best_batsmen_stats.append(best_batsman)
        best_bowlers_stats.append(best_bowler)

    # Visualization
    plt.figure(figsize=(16, 6))

    # Best Batsmen Performance
    plt.subplot(1, 2, 1)
    for i, year in enumerate(years_list):
        plt.bar(str(year), best_batsmen_stats[i]['Avg'], label=str(year))
    plt.xlabel('Year')
    plt.ylabel('Average')
    plt.title('Best Batsman Performance from 2016 to 2022')
    plt.legend()

    # Best Bowlers Performance
    plt.subplot(1, 2, 2)
    for i, year in enumerate(years_list):
        plt.bar(str(year), best_bowlers_stats[i]['Avg'], label=str(year))
    plt.xlabel('Year')
    plt.ylabel('Average')
    plt.title('Best Bowler Performance from 2016 to 2022')
    plt.legend()

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



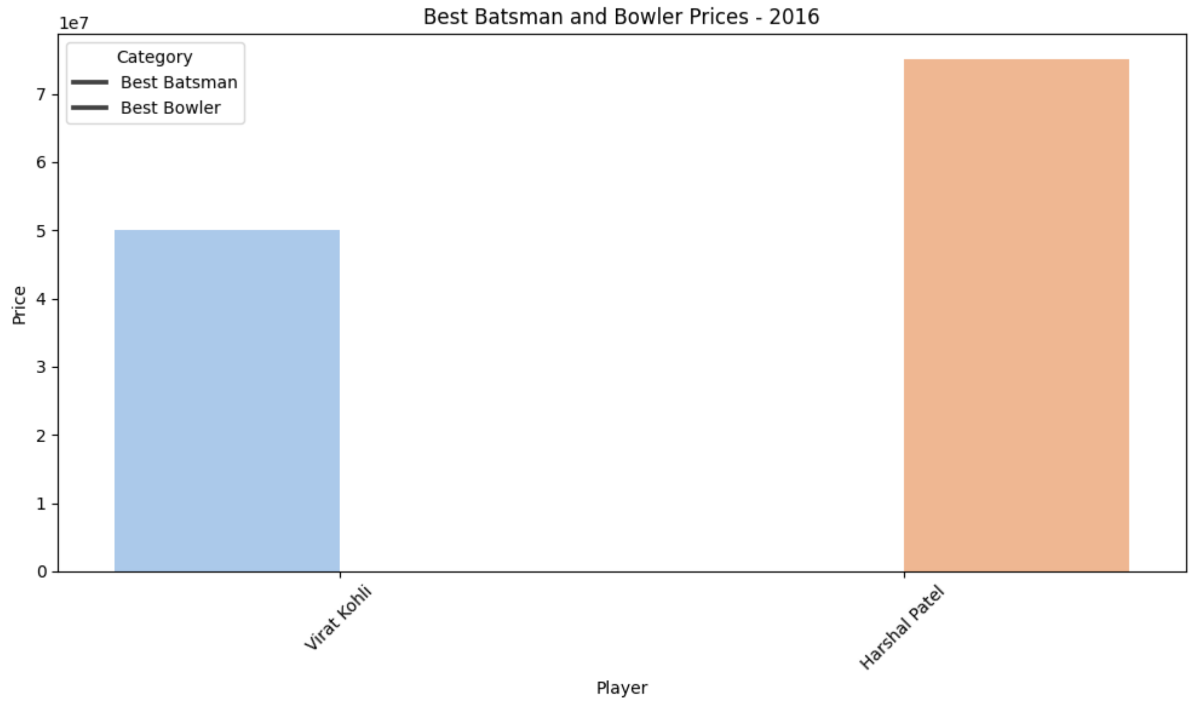


Assigning Player Price for Auction Every best Batsman And Best Bowler Price starts for bitting from 50000000 to 75000000 and their bar visualization

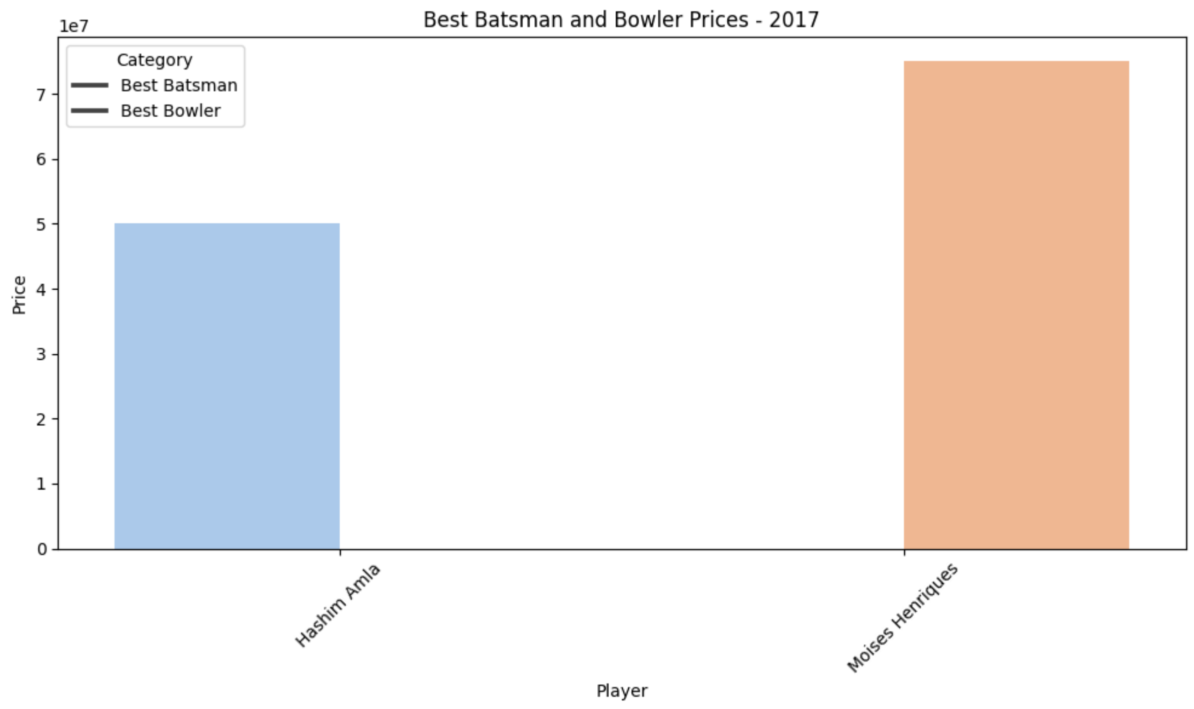
in this section i assign value price of players for the starts for everyone ,that is 5000000 for best batsman and 75000000 for best bowler also i visualize their price and name for whom

In [30]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load CSV files
def load_csv(file_path):
    try:
        return pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"File not found: {file_path}")
        return None

# Find the best player for each role based on performance metrics
def find_best_player(df, metric, num_players=1):
    sorted_df = df.sort_values(by=[metric], ascending=False)
    return sorted_df.head(num_players)

# Main function to form a new team
def form_new_team(year, batting_file_path, bowling_file_path):
    batting_df = load_csv(batting_file_path)
    bowling_df = load_csv(bowling_file_path)

    if batting_df is None or bowling_df is None:
        return

    # Finding the best batsman and best bowler for the new team
    best_batsman = find_best_player(batting_df, 'Avg', num_players=1)
    best_bowler = find_best_player(bowling_df, 'Avg', num_players=1)

    # Printing the results
    print(f"Year: {year}")
    print("Best Batsman:")
    print(best_batsman)
    print("Best Bowler:")
    print(best_bowler)

    # Player prices (replace with actual prices)
    player_prices = {
        best_batsman.iloc[0]['Player']: 50000000,
        best_bowler.iloc[0]['Player']: 75000000,
    }

    # Adding player prices to the DataFrames
    best_batsman['Price'] = best_batsman['Player'].map(player_prices)
    best_bowler['Price'] = best_bowler['Player'].map(player_prices)

    # Concatenate the DataFrames
    best_players = pd.concat([best_batsman, best_bowler])

    # Plotting the prices of the best players
    plt.figure(figsize=(10, 6))
    sns.barplot(data=best_players, x='Player', y='Price', hue='POS', palette='pastel')
    plt.title(f"Best Batsman and Bowler Prices - {year}")
    plt.xticks(rotation=45)
    plt.legend(title='Category', loc='upper left', labels=["Best Batsman", "Best Bowler"])

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    years_list = [2016, 2017, 2018, 2019, 2020, 2021, 2022]
    file_paths_by_year = {
        2016: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2016.csv",
        },
        2017: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2017.csv",
        },
        2018: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2018.csv",
        },
        2019: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2019.csv",
        },
        2020: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2020.csv",
        },
        2021: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2021.csv",
        },
        2022: {
            'batting': r"C:\Users\ggulz\Downloads\archive\IPL Player Stats\Batting Stats\BA
            'bowling': r"C:\Users\ggulz\Downloads\archive\BOWLING STATS - IPL_2022.csv",
        },

    }

    for year in years_list:
        batting_file_path = file_paths_by_year[year]['batting']
        bowling_file_path = file_paths_by_year[year]['bowling']
        form_new_team(year, batting_file_path, bowling_file_path)
```

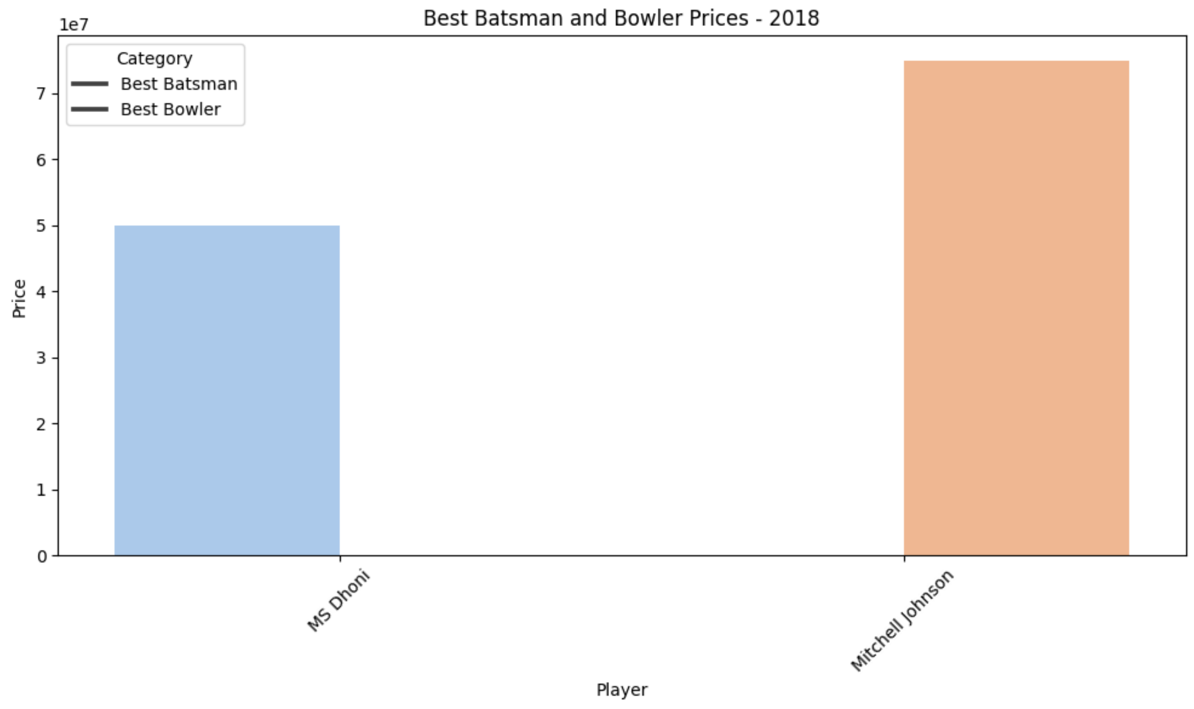
```
Year: 2016
Best Batsman:
   POS       Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  50  \
0    1  Virat Kohli   16    16   4   973  113  81.08  640  152.03    4   7   

   4s  6s  
0  83  38  
Best Bowler:
    POS         Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ    SR  \
85   86  Harshal Patel    5     5  14   147     1  5/27  147.0  10.5  84.0   

    4w  5w  
85   0   0  
```



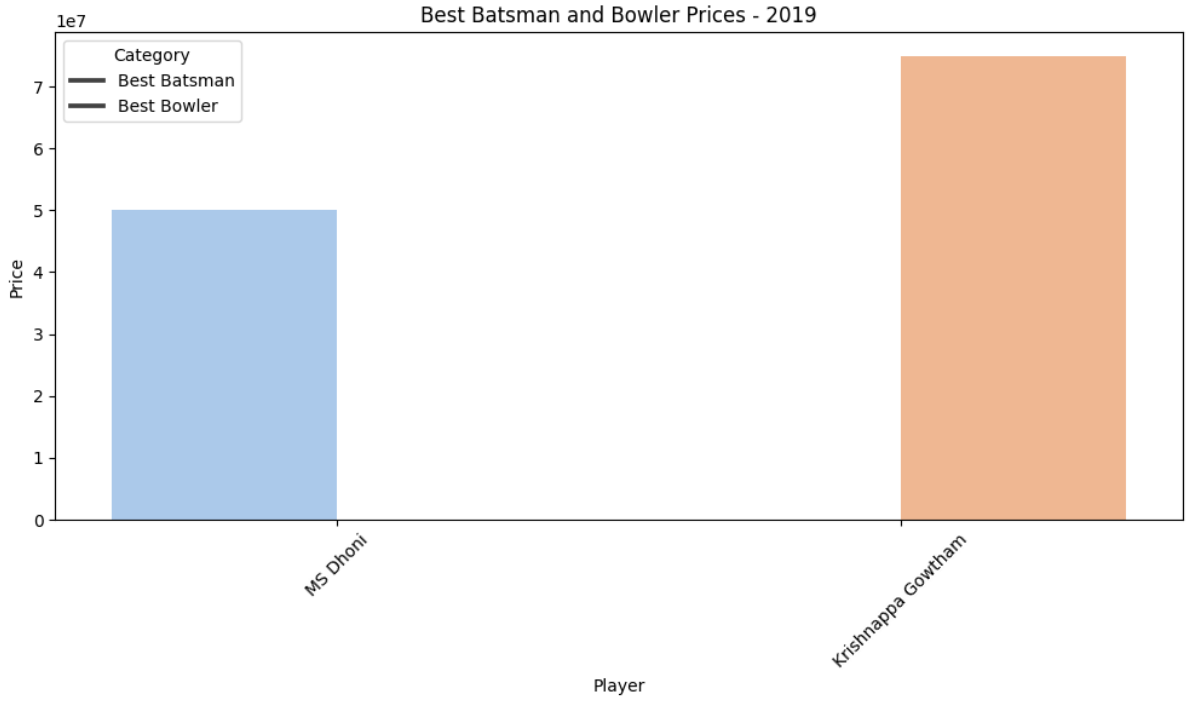
```
Year: 2017
Best Batsman:
   POS       Player  Mat  Inns  NO  Runs    HS   Avg   BF      SR  100  50  \
5    6  Hashim Amla   10    10   3   420  104*  60.0  288  145.83    2   2   

   4s  6s  
5  40  17  
Best Bowler:
    POS            Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg   Econ  \
89   90  Moises Henriques   12    12  24   248     1  3/12  248.0  10.33   

       SR  4w  5w  
89  144.0   0   0  
```



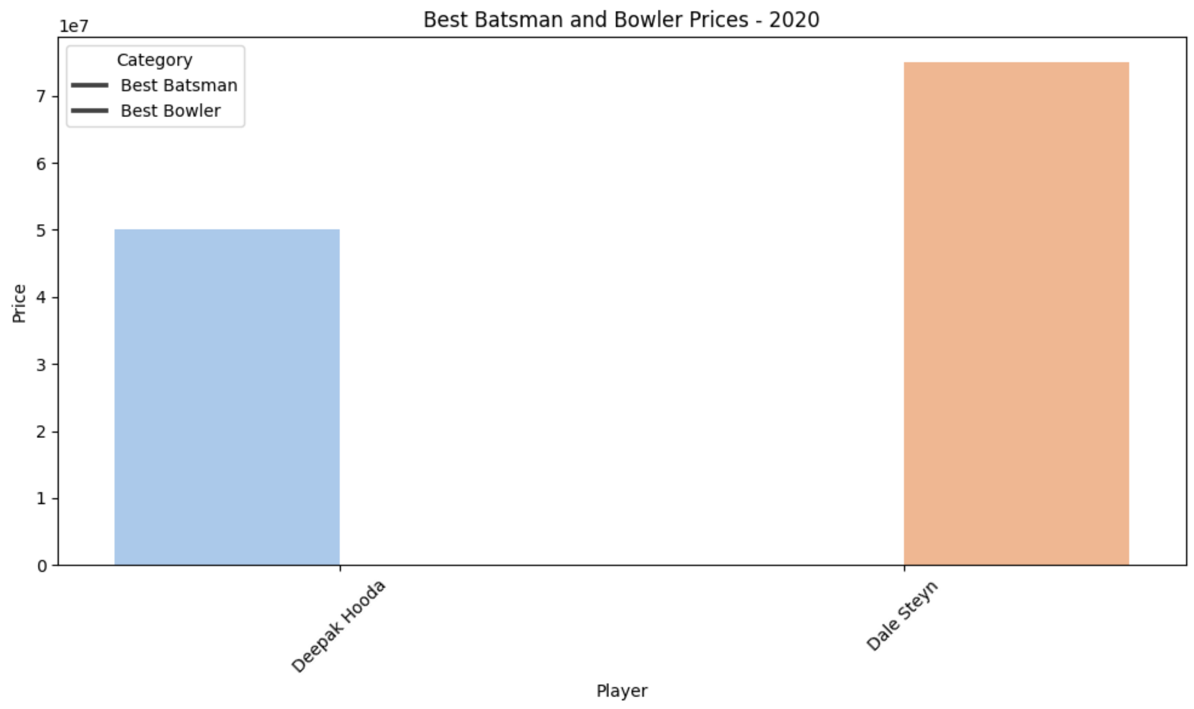
```
Year: 2018
Best Batsman:
    POS    Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  50  4s  \
12   13  MS Dhoni   16    15   9   455  79*  75.83  302  150.66    0   3  24   

    6s  
12  30  
Best Bowler:
    POS           Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg   Econ  \
72   73  Mitchell Johnson    6     6  21   216     2  3/26  108.0  10.28   

      SR  4w  5w  
72  63.0   0   0  
```



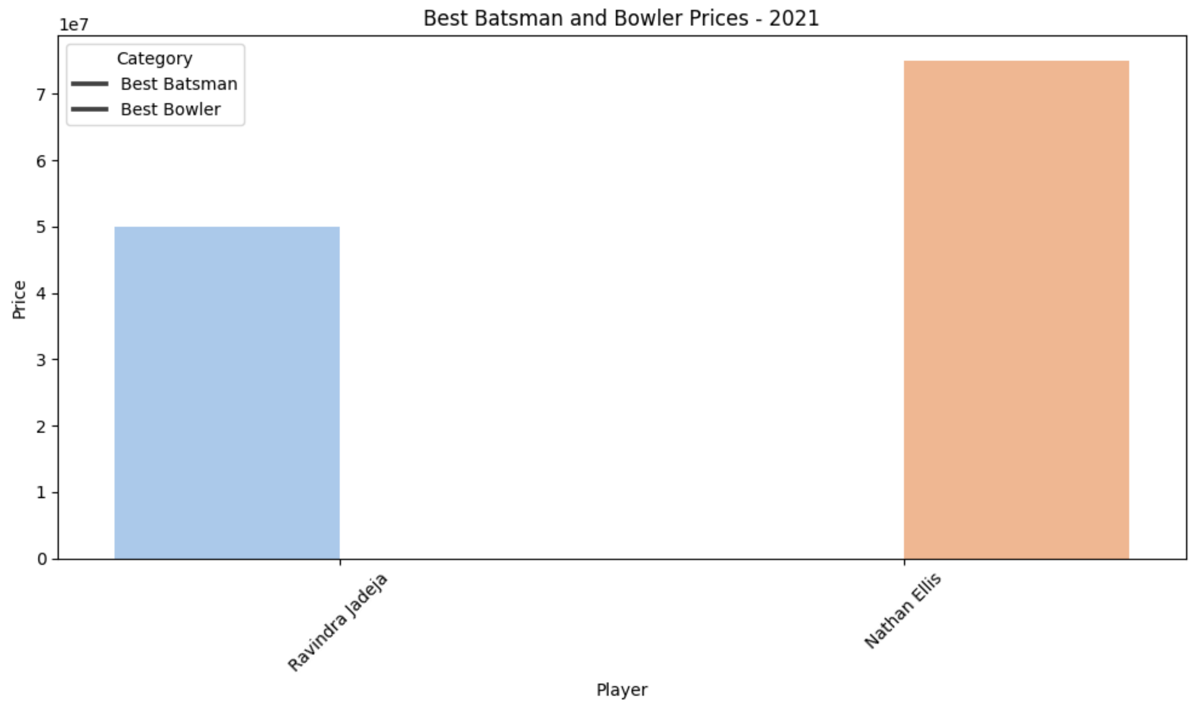
```
Year: 2019
Best Batsman:
    POS    Player  Mat  Inns  NO  Runs   HS   Avg   BF      SR  100  50  4s  \
12   13  MS Dhoni   15    12   7   416  84*  83.2  309  134.62    0   3  22   

    6s  
12  23  
Best Bowler:
    POS              Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ  \
77   78  Krishnappa Gowtham    7     7  20   166     1  2/12  166.0   8.3   

       SR  4w  5w  
77  120.0   0   0  
```



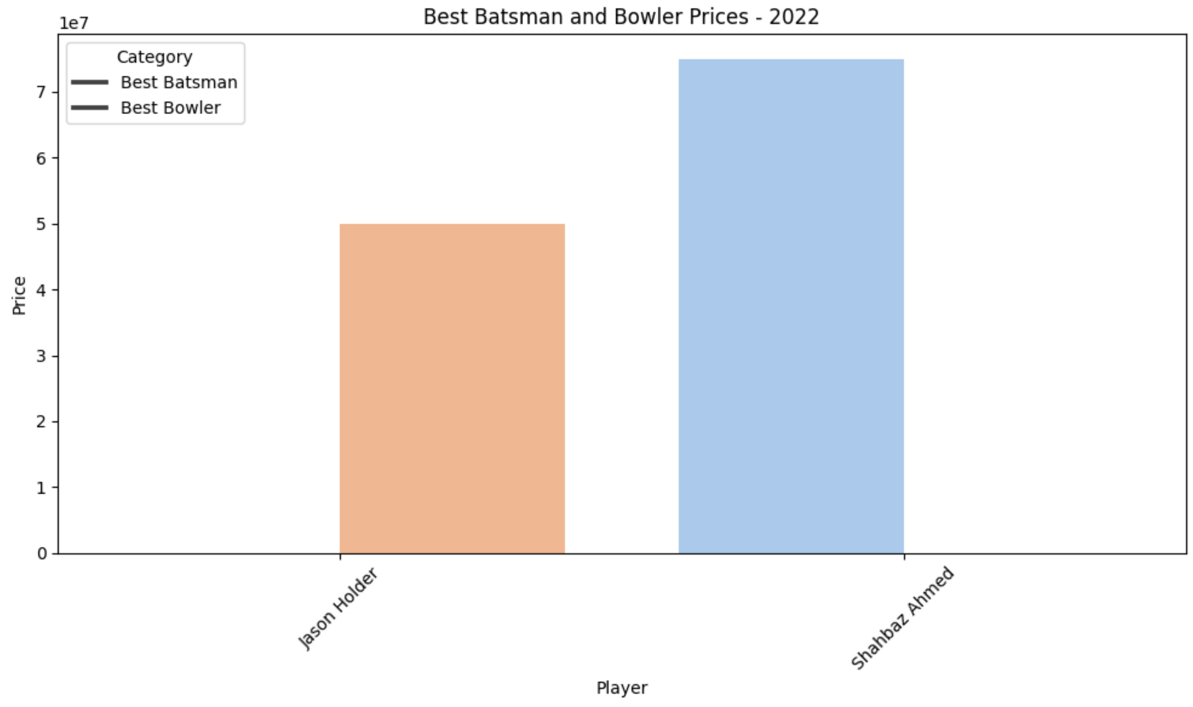
```
Year: 2020
Best Batsman:
    POS        Player  Mat  Inns  NO  Runs   HS    Avg  BF      SR  100  50  \
57   58  Deepak Hooda    7     5   4   101  62*  101.0  71  142.25    0   1   

    4s  6s  
57   5   5  
Best Bowler:
    POS      Player  Mat  Inns  Ov  Runs  Wkts   BBI    Avg  Econ    SR  4w  \
67   68  Dale Steyn    3     3  11   133     1  3/11  133.0  11.4  70.0   0   

    5w  
67   0  
```



```
Year: 2021
Best Batsman:
    POS           Player  Mat  Inns  NO  Runs   HS    Avg   BF      SR  100  \
30   31  Ravindra Jadeja   16    12   9   227  62*  75.66  156  145.51    0   

    50  4s  6s  
30   1  19   9  
Best Bowler:
    POS        Player  Mat  Inns  Ov  Runs  Wkts   BBI   Avg  Econ    SR  4w  \
88   89  Nathan Ellis    3     3  12    98     1  1/20  98.0  8.16  72.0   0   

    5w  
88   0  
```



```
Year: 2022
Best Batsman:
    POS        Player  Mat  Inns  NO  Runs  HS   Avg  BF      SR  100  50  4s  \
84   85  Jason Holder   12     8   2    58  16  9.67  44  131.81    0   0   2   

    6s  
84   6  
Best Bowler:
    POS          Player  Mat  Inns    Ov  Runs  Wkts   BBI   Avg  Econ    SR  \
72   73  Shahbaz Ahmed   16    14  35.0   336     4  26/2  84.0   9.6  52.5   

    4w  5w  
72   0   0  
```



```
In [ ]:
```