Today I am going to go through a random dataset I found on Kaggle, which is about the Indian Premier League(IPL). The dataset usability has been shown 10 out of 10 so I thought why not give it a try? I tried to do a proper Exploratory Data Analysis(EDA) on the dataset and this is the first part as the EDA is going to be bigger than I thought.

The dataset which I am working with contains all the information about IPL from season 2008 to 2016, meaning nine seasons in total. First of all, the dataset contains two CSV files, one is deliveries.csv another is matches.csv. But the thing is proper description is not provided in the data card by the author. The file named deliveries.csv is the ball-by-ball data of all the IPL matches including data of the batting team, batsman, bowler, non-striker, runs scored, etc. So there are 636 match data for each batsman and bowler along with their teams. After that, the other file matches.csv contains details related to the match such as location, contesting teams, umpires, results, etc. Not wasting our time anymore let us dive into EDA!

We need totally basic Python libraries, which are numpy, pandas, matplotlib and seaborn.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

I ran some basic codes seeing the datasets head and shapes. It has been found that the delivery dataset has 150460 rows and 21 columns. So it is indeed a huge data to do analysis. After seeing the information summary with '.info' method I found there eight columns containing object type data and other 13 columns with numerical value. But three of the columns has too many null values, which is needed to be handled.
So, readers, this part is totally for the 'delivery.csv' file.

To check the number of null values I wrote:

```
delivery.isnull.sum()
```

The three columns with null values that is found are:

```
player_dismissed    143022
dismissal_kind      143022
fielder             145091
```

I am interested in the batting teams and the batsmen for now. So let me check out the teams with '.unique' method.

```
delivery['batting_team'].unique()
```

I got many teams and I felt nostalgic to see the names because some of the teams of previous season dont exist or has been changed to other name.
What I got is:

```
array(['Sunrisers Hyderabad', 'Royal Challengers Bangalore',
       'Mumbai Indians', 'Rising Pune Supergiant', 'Gujarat Lions',
       'Kolkata Knight Riders', 'Kings XI Punjab', 'Delhi Daredevils',
       'Chennai Super Kings', 'Rajasthan Royals', 'Deccan Chargers',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Rising Pune Supergiants'],
      dtype=object)
```

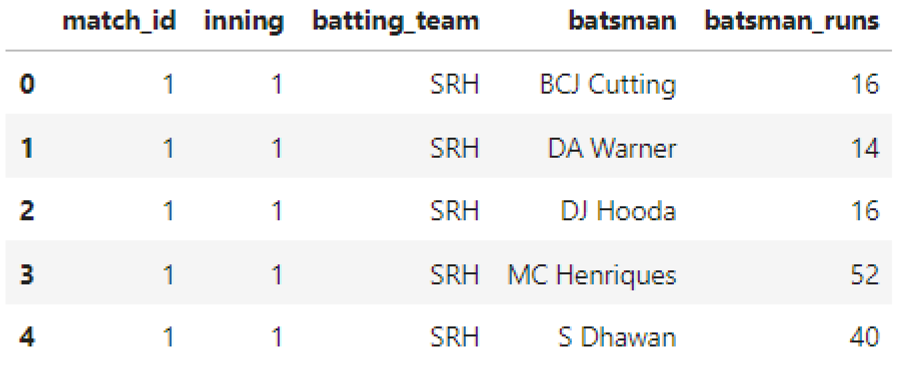
It will be somewhat irritating to work with this long string types so let me make it smaller with their short names. So I will use the '.replace' method here.

```
delivery.replace(['Mumbai Indians','Kolkata Knight Riders','Royal Challengers Bangalore','Deccan Chargers','Chennai Super Kings',
                 'Rajasthan Royals','Delhi Daredevils','Gujarat Lions','Kings XI Punjab',
                 'Sunrisers Hyderabad','Rising Pune Supergiants','Kochi Tuskers Kerala','Pune Warriors','Rising Pune Supergiant']
,['MI','KKR','RCB','DC','CSK','RR','DD','GL','KXIP','SRH','RPS','KTK','PW','RPS'],inplace=True)
```

Now at this point let me take the batsmen group only, further I will be dealing with the bowlers section. So to get the batsmen I grouped the columns "match_id", "inning", "batting_team", "batsman" to a new dataframe named batsman_grp. I just want to see the team and batsmen.

```
batsman_grp = delivery.groupby(["match_id", "inning", "batting_team", "batsman"])
batsmen = batsman_grp["batsman_runs"].sum().reset_index()
```

So the dataframe looks like this:

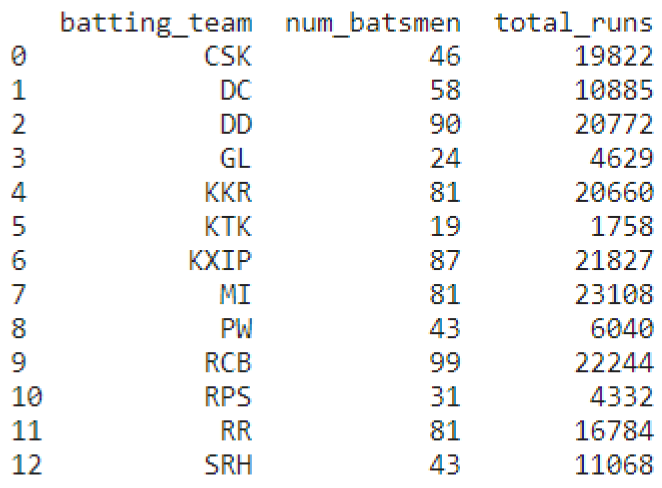


	match_id	inning	batting_team	batsman	batsman_runs
0	1	1	SRH	BCJ Cutting	16
1	1	1	SRH	DA Warner	14
2	1	1	SRH	DJ Hooda	16
3	1	1	SRH	MC Henriques	52
4	1	1	SRH	S Dhawan	40

Let us go see some answers to some questions.

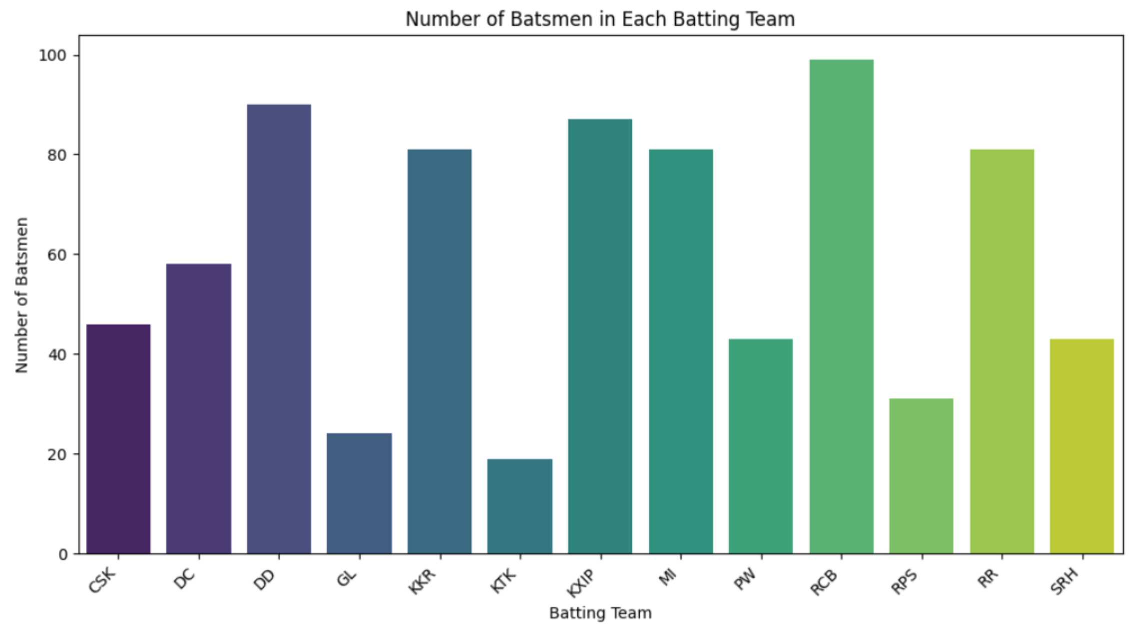
Question 1. How many runs in total each team has made and how many unique players contributed to the total runs?

Code:
```
team_stats = batsmen.groupby('batting_team').agg(
    num_batsmen=('batsman', 'nunique'),  # Count of unique batsmen
    total_runs=('batsman_runs', 'sum')  # Total runs made by the team
).reset_index()

print(team_stats)
```
Result:

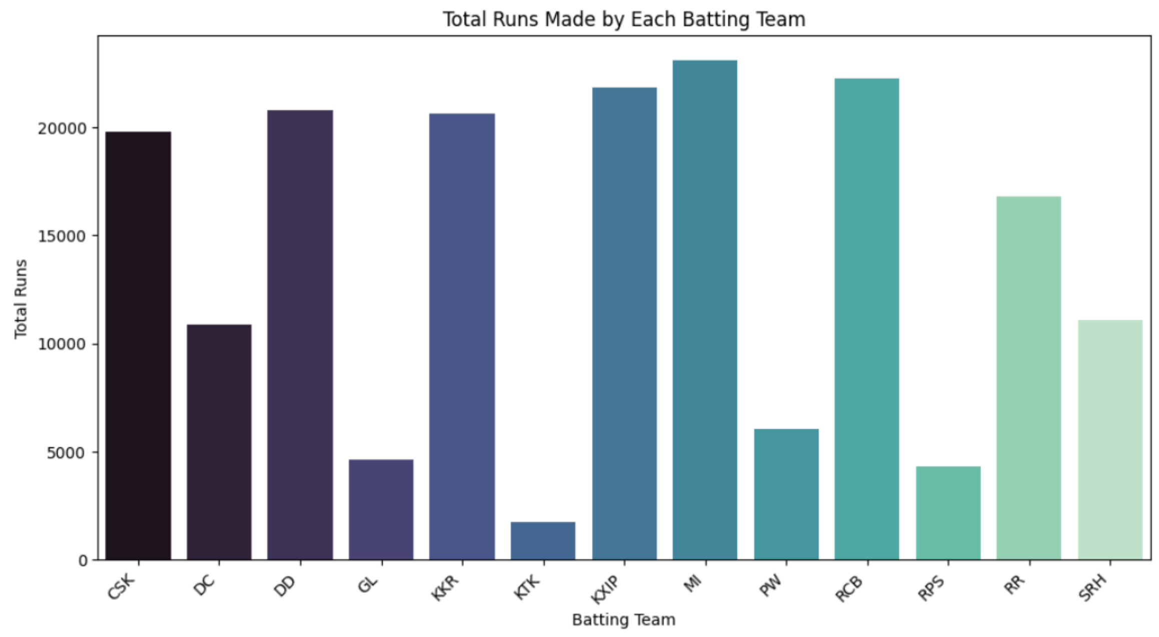


```
   batting_team  num_batsmen  total_runs
0           CSK           46       19822
1            DC           58       10885
2            DD           90       20772
3            GL           24        4629
4           KKR           81       20660
5           KTK           19        1758
6          KXIP           87       21827
7            MI           81       23108
8            PW           43        6040
9           RCB           99       22244
10          RPS           31        4332
11           RR           81       16784
12          SRH           43       11068
```

Let me try to visualize this by using barchart. It will be great.
```
# Creating a bar chart for num_batsmen
plt.figure(figsize=(12, 6))
sns.barplot(x='batting_team', y='num_batsmen', data=team_stats, palette='viridis')
plt.title('Number of Batsmen in Each Batting Team')
plt.xlabel('Batting Team')
plt.ylabel('Number of Batsmen')
plt.xticks(rotation=45, ha='right')
plt.show()
```



```
# Creating a bar chart for total_runs
plt.figure(figsize=(12, 6))
sns.barplot(x='batting_team', y='total_runs', data=team_stats, palette='mako')
plt.title('Total Runs Made by Each Batting Team')
plt.xlabel('Batting Team')
plt.ylabel('Total Runs')
plt.xticks(rotation=45, ha='right')
plt.show()
```



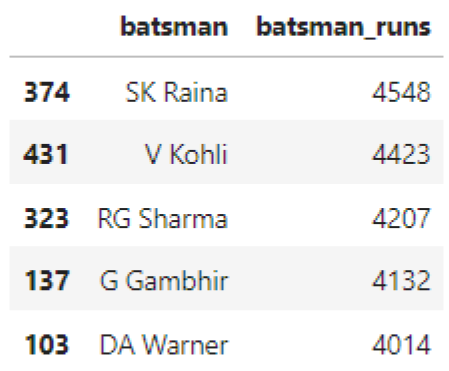
From the above analysis, we can find some insights. The unique number of batsmen for a team is the highest for Royal Challengers Bangalore, they have given chance to 99 players to bat! Again we can see that Mumbai Indians with Kings XI Punjab are the highest run-makers in total.

Question 2. Can you tell the total runs made by each batsman?

Yes obviously it is possible by this code:

```
batsman_stats = batsmen.groupby('batsman')['batsman_runs'].sum().reset_index()
batsman_stats = batsman_stats.sort_values(by='batsman_runs', ascending=False)

batsman_stats.head()
```

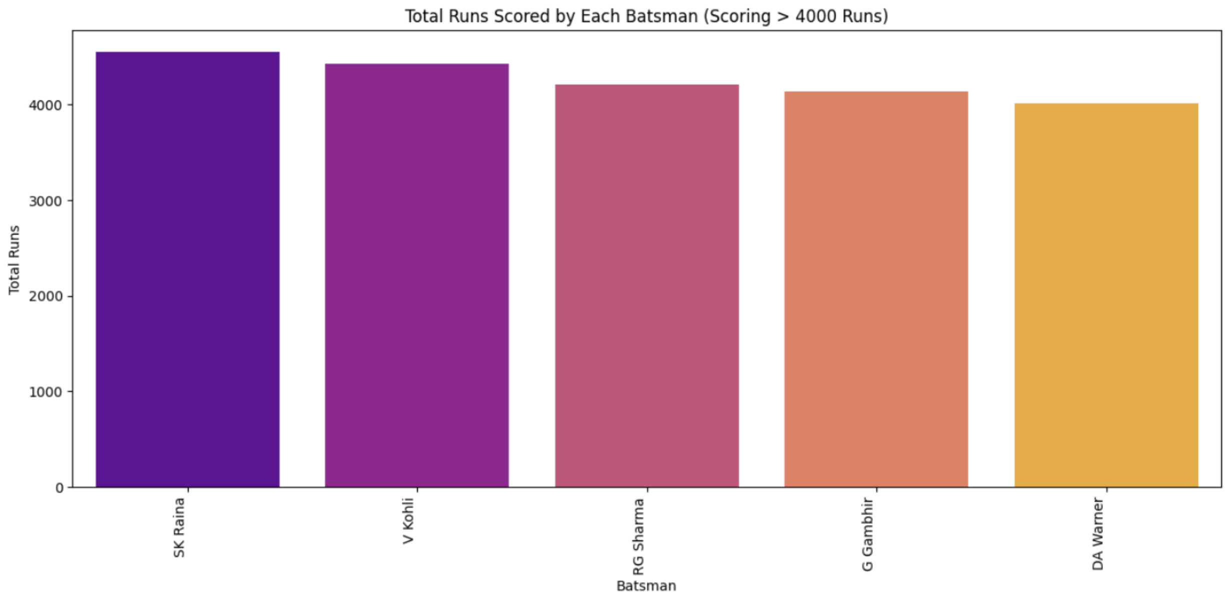
So I get,



	batsman	batsman_runs
374	SK Raina	4548
431	V Kohli	4423
323	RG Sharma	4207
137	G Gambhir	4132
103	DA Warner	4014

Question 3. It seems that several hard-hitting batsmen have scored a total of 4000 or more. So can you show the batsman who has scored 4000 runs or more? (Many will say they are GOAT!)

```
batsman_stats = batsmen.groupby('batsman')['batsman_runs'].sum().reset_index()

#batsmen with total runs greater than 4000
top_batsmen = batsman_stats[batsman_stats['batsman_runs'] > 4000]
top_batsmen = top_batsmen.sort_values(by='batsman_runs', ascending=False)

plt.figure(figsize=(15, 6))
sns.barplot(x='batsman', y='batsman_runs', data=top_batsmen, palette='plasma')
plt.title('Total Runs Scored by Each Batsman (Scoring > 4000 Runs)')
plt.xlabel('Batsman')
plt.ylabel('Total Runs')
plt.xticks(rotation=90, ha='right')
plt.show()
```

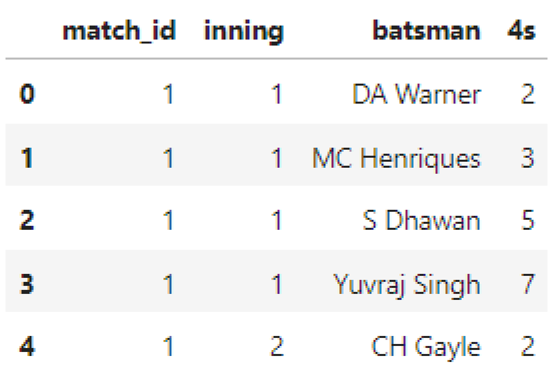


Owh! Thats great! The top scorers have been found, they are:
Suresh Raina
Virat Kohli
RG Sharma
Gautam Gambhir
David Warner

Question 4. Can you show me how many fours or sixes have been hit by each of the batsmen in any consecutive innings? And if you can find it then show the highest number of fours and sixes in any innings by which batsman?

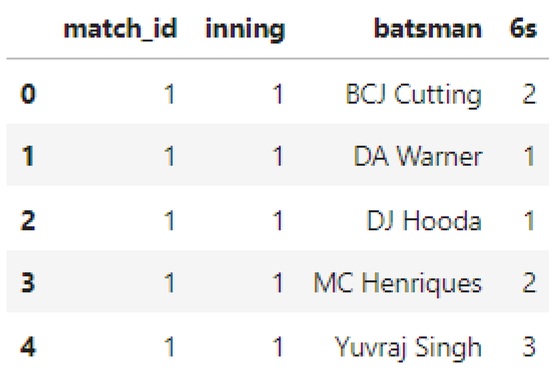
It seems tough. But the thing is we can go for the easy way by making two individual dataframes named fours and sixes with only taking the 4s and 6s columns. Then grouping it by match_is, innings and batsman just put the count method and it will do the rest.

```
fours = delivery[ delivery["batsman_runs"] == 4]
sixes = delivery[ delivery["batsman_runs"] == 6]

fours_per_batsman = fours.groupby(["match_id", "inning",
"batsman"])["batsman_runs"].count().reset_index()
sixes_per_batsman = sixes.groupby(["match_id", "inning",
"batsman"])["batsman_runs"].count().reset_index()

fours_per_batsman.columns = ["match_id", "inning", "batsman", "4s"]
sixes_per_batsman.columns = ["match_id", "inning", "batsman", "6s"]

fours_per_batsman.head()
```



	match_id	inning	batsman	4s
0	1	1	DA Warner	2
1	1	1	MC Henriques	3
2	1	1	S Dhawan	5
3	1	1	Yuvraj Singh	7
4	1	2	CH Gayle	2

```
sixes_per_batsman.head()
```



	match_id	inning	batsman	6s
0	1	1	BCJ Cutting	2
1	1	1	DA Warner	1
2	1	1	DJ Hooda	1
3	1	1	MC Henriques	2
4	1	1	Yuvraj Singh	3

```
# Find the batsman with the most fours
most_fours_batsman = fours_per_batsman.loc[fours_per_batsman["4s"].idxmax()]

# Find the batsman with the most sixes
most_sixes_batsman = sixes_per_batsman.loc[sixes_per_batsman["6s"].idxmax()]
print("Batsman with the most fours:")
print(most_fours_batsman)
print("\nBatsman with the most sixes:")
print(most_sixes_batsman)
```

Result:

```
Batsman with the most fours:
match_id            243
inning                2
batsman     PC Valthaty
4s                   19
Name: 2226, dtype: object

Batsman with the most sixes:
match_id         411
inning             1
batsman     CH Gayle
```

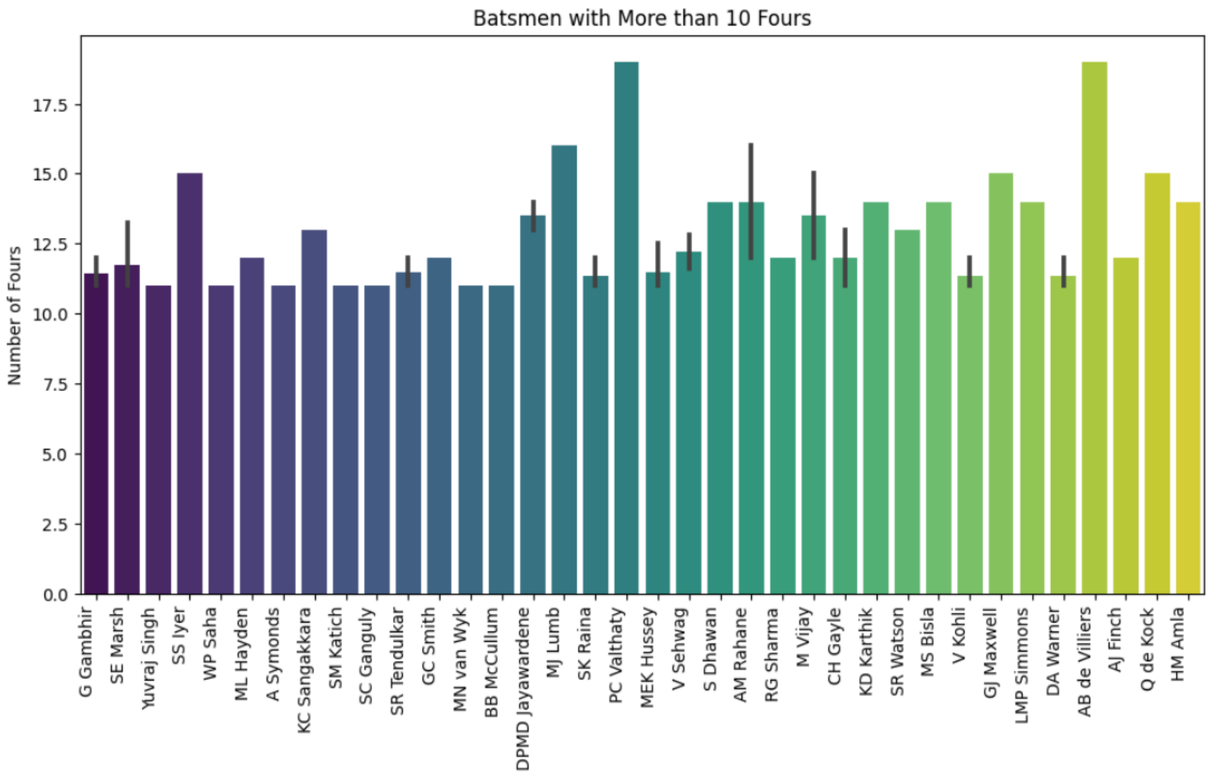
```
6s            17
Name: 2120, dtype: object
```

Question 5. Now I am interested in those batsmen who have scored more than 10 fours in an innings and also those batsmen who hit more than 10 sixes in an innings.

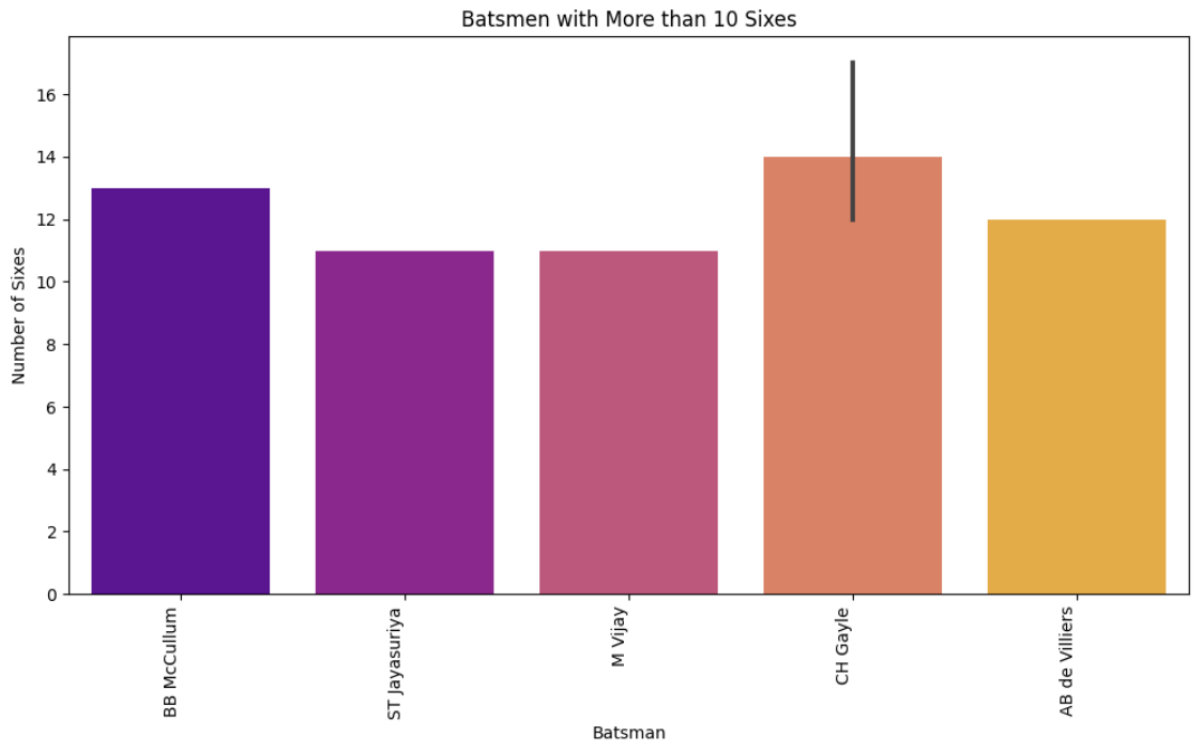
Code:

```
# Filter batsmen with more than 10 fours
batsmen_with_more_than_10_fours = fours_per_batsman[fours_per_batsman["4s"] > 10]

# Filter batsmen with more than 10 sixes
batsmen_with_more_than_10_sixes = sixes_per_batsman[sixes_per_batsman["6s"] > 10]

# Plotting a bar chart for batsmen with more than 10 fours
plt.figure(figsize=(12, 6))
sns.barplot(x='batsman', y='4s', data=batsmen_with_more_than_10_fours, palette='viridis')
plt.title('Batsmen with More than 10 Fours')
plt.xlabel('Batsman')
plt.ylabel('Number of Fours')
plt.xticks(rotation=90, ha='right')
plt.show()

# Plotting a bar chart for batsmen with more than 10 sixes
plt.figure(figsize=(12, 6))
sns.barplot(x='batsman', y='6s', data=batsmen_with_more_than_10_sixes, palette='plasma')
plt.title('Batsmen with More than 10 Sixes')
plt.xlabel('Batsman')
plt.ylabel('Number of Sixes')
plt.xticks(rotation=90, ha='right')
plt.show()
```

Result:



There has been a close contest between PC Valthathy and AB De Villiers. They are hard hitter batsmen one must admit! Let us see the six-hitters.



Again the universe boss Chris Gayle, but Brendon McCullum and AB De Villiers are also in the race!

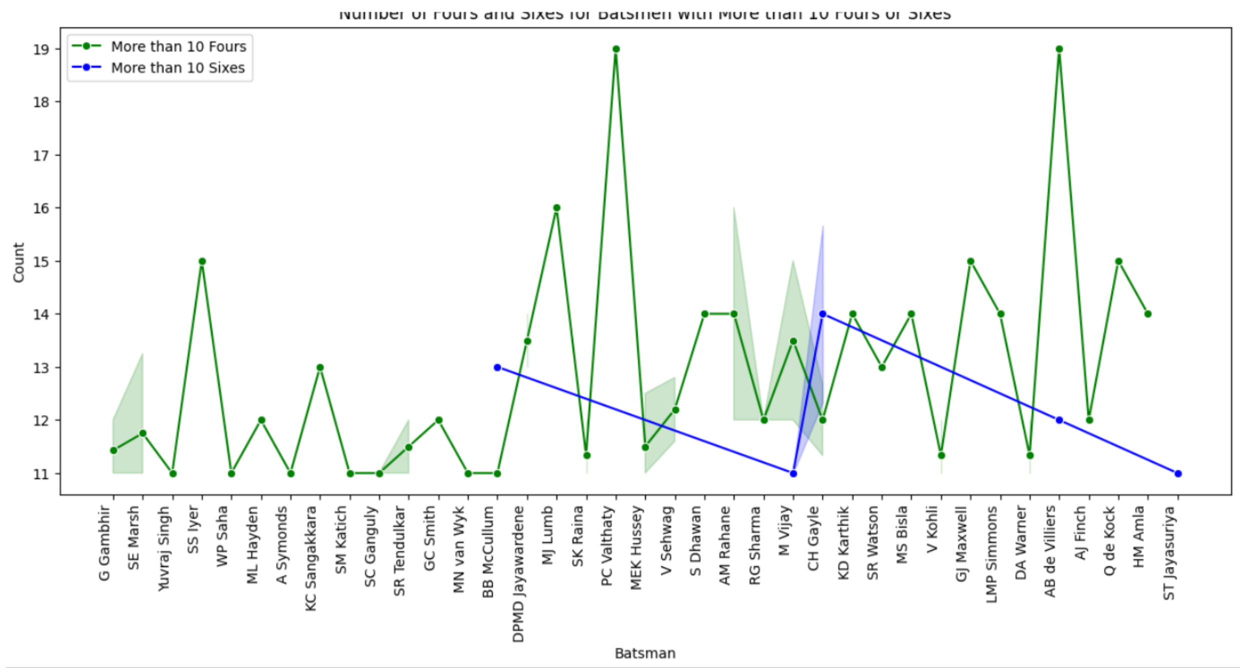
Question 6. Is it possible to see the above-mentioned criteria to see together?

Yes why not? We can merge the two dataframe with fours and sixes more than 10 and plot it into a line chart.

Code:

```
merged_data = pd.merge(batsmen_with_more_than_10_fours, batsmen_with_more_than_10_sixes,
on='batsman', how='outer')

# Plotting a line chart for both scenarios
plt.figure(figsize=(15, 6))
sns.lineplot(x='batsman', y='4s', data=merged_data, label='More than 10 Fours', marker='o',
color='green')
sns.lineplot(x='batsman', y='6s', data=merged_data, label='More than 10 Sixes', marker='o',
color='blue')

plt.title('Number of Fours and Sixes for Batsmen with More than 10 Fours or Sixes')
plt.xlabel('Batsman')
plt.ylabel('Count')
plt.xticks(rotation=90, ha='right')
plt.legend()
plt.show()
```

Result:


Number of Fours and Sixes for Batsmen with More than 10 Fours or Sixes
More than 10 Fours
More than 10 Sixes
Count
G Gambhir
SE Marsh
Yuvraj Singh
SS Iyer
WP Saha
ML Hayden
A Symonds
KC Sangakkara
SM Katich
SC Ganguly
SR Tendulkar
GC Smith
MN van Wyk
BB McCullum
DPMD Jayawardene
MJ Lumb
SK Raina
PC Valthaty
MEK Hussey
V Sehwag
S Dhawan
AM Rahane
RG Sharma
M Vijay
CH Gayle
KD Karthik
SR Watson
MS Bisla
V Kohli
GJ Maxwell
LMP Simmons
DA Warner
AB de Villiers
AJ Finch
Q de Kock
HM Amla
ST Jayasuriya
Batsman