Project :- Web scrapping of the IPL 2023 Point table from News18 website.

Step 1:-

Here we performs web scraping using the BeautifulSoup library, requests module & Pandas Modules.

*Step 1.1 :- The code imports the BeautifulSoup library and assigns it an alias "bf". This library allows the code to parse HTML and XML documents, extract useful information and navigate through their structure.

*Step 1.2 :- Requests module is imported, which enables the code to send HTTP/1.1 requests and obtain their responses.

*Step 1.3 :- A URL variable is defined as "https://www.news18.com/cricketnext/ipl-2023/points-table.html" (https://www.news18.com/cricketnext/ipl-2023/points-table.html") which points to the web page that the code intends to scrape.

*Step 1.4 :- A request is sent to the web server to retrieve the content of the web page using the requests.get() method. The method takes the URL as an argument and returns a response object r that contains the HTML content of the web page.

The code will proceed to extract the data from the HTML content using the BeautifulSoup library.

In [16]:

```
from bs4 import BeautifulSoup as bf
import requests
url="https://www.news18.com/cricketnext/ipl-2023/points-table.html"
r=requests.get(url)
print(r)
```

```
<Response [200]>
```

Step 2:-

The below code consists of two lines:

*Step 2.1 :- The first line assigns the content attribute of the Response object r to the variable htmlcontent. In HTTP, a response is sent from a server to a client after receiving a request. The response object contains the server's response to the client request, which typically includes an HTTP status code, headers, and the response body (which is the actual content of the response).

The content attribute of the Response object represents the raw content of the response, typically in bytes. It is useful when you want to get the response content in its original form, without any decoding or processing.

So, in this line, the htmlcontent variable is assigned the raw response content.

*Step 2.2 :-The second line print(htmlcontent) prints the content of the htmlcontent variable to the console. This is useful when you want to check the content of the response, such as when you are testing a web scraping script or an API client.

In [3]:

```
htmlcontent=r.content
print(htmlcontent)
```

```
ext/static/chunks/31516-3d4f1ce4cf6703ddaf75.js,/_next/static/chunks/306
53-f40262ff37a17c7dd3b2.js,/_next/static/chunks/23228-4718f64af473a4f195
20.js,/_next/static/chunks/84482-0ca8b4e2f9fa3a7f049b.js,/_next/static/c
hunks/29861.c6031b0b68fadd0d5d36.js,/_next/static/chunks/2962-9afcf162f5
13270ff827.js,/_next/static/chunks/89569.9865cfd89f045bd8d488.js,/_next/
static/chunks/81342.cca4e0d05a26b7dc146a.js,/_next/static/chunks/83229.0
3b0027d5cdf7580a5ad.js,/_next/static/chunks/32206.062bcc2bb9eeaba9f6c1.j
s,/_next/static/chunks/28950.47d79fa1fbe001b319d1.js,/_next/static/chunk
s/15772.15d0b0d40dcbedaeedda.js\';\n        var _mydynchunks = dynnewchu
nks.split(\',\');\n        window.addEventListener(\'load\', () => {\n
//console.log("window load _mydynchunks: ", _mydynchunks);\n          fo
r (var i = 0; i < _mydynchunks.length; i++) {\n            let ref = (do
cument.getElementsByTagName(\'head\')[0] || document.getElementsByTagNam
e(\'body\')[0]);\n            let runtimejs = _mydynchunks[i];\n
var script = document.createElement("script");\n            script.src =
runtimejs;\n            script.defer = true;\n            script.crossor
igin = "anonymous";\n            ref.appendChild(script);\n          }\n
})</script><script type="text/javascript">\n        let newchunks = \'/_
next/static/chunks/webpack-2b314b6012799b9a6a5a.js,/_next/static/chunks/
framework-a43af322d6c0069d5762.js,/_next/static/chunks/main-6f892ba847d0
```

Step 3 :-

The below code consists of two lines:

*Step 3.1 :- The first line of code creates a BeautifulSoup object by passing two arguments - the HTML content of a web page (stored as a string in the variable htmlcontent), and the parser to use ("html.parser" in this case).

*Step 3.2 :- The second line of code prints the "prettified" version of the HTML document. soup.prettify() is a method that takes no arguments and returns a string that formats the HTML content in an indented and human-readable way.

This makes it easier to visually inspect the structure of the HTML document and locate specific elements for further processing, such as extracting text or links.

In [4]:

```
soup=bf(htmlcontent,"html.parser")
print(soup.prettify())
```

```
        setTimeout(() => {
            var purl = window.location.href;
            var url = '//ads.pubmatic.com/AdServer/js/pwt/113941/826';
            var profileVersionId = '';
            if (purl.indexOf('pwtv=') > 0) {
                var regexp = /pwtv=(.*?)(&|$)/g;
                var matches = regexp.exec(purl);
                if (matches.length >= 2 && matches[1].length > 0) {
                    profileVersionId = '/' + matches[1];
                }
            }
            scriptLoader(url + profileVersionId + "/pwt.js", () => {
                scriptLoader("https://securepubads.g.doubleclick.net/ta
g/js/gpt.js");
                setTimeout(() => {
                    adRecoverScript();
                }, 2000);
            });
        }, 1000);
        setTimeout(() => {
```

Step 4 :-

The given code is using the BeautifulSoup library in Python to extract the title of a web page and then print it to the console.

Here's what's happening in each line of the code:

*Step 4.1 :-title=soup.title - This line creates a variable called title and assigns it the value of the title of the web page. The soup object is assumed to be an instance of the BeautifulSoup class, which is initialized with the HTML content of the web page.

*Step 4.2 :- print(title.string) - This line prints the string attribute of the title variable to the console. The string attribute is a string representation of the title of the web page. The print() function is used to output the string to the console.

Overall, the code is a simple example of how to use BeautifulSoup to extract data from an HTML page.

In [5]:

```
title=soup.title
print(title.string)
```

```
IPL 2023 Points Table: Updated Ranking After Today Match
```

Step 5 :-

Soup is a variable that represents the parsed HTML code of the webpage. It is likely created using BeautifulSoup(html_content, 'html.parser'), where html_content is the raw HTML code of the webpage.

The code then uses the find method on the soup object to search for a specific HTML element on the webpage. In this case, it is looking for a div element with the CSS class super_group_table.

The result of this search is stored in a variable called table, which can then be used to extract more information from the webpage, such as its text content or child elements. If no matching element is found, table will be assigned the value None.

In [6]:

```
table=soup.find("div",class_="super_group_table")
```

Step 6 :-

The below code is likely part of a Python script and it is used the find() method on an object called table.

In general, the find() method is used to search for a specific element within a larger container, such as a string or an HTML document. It takes a single argument, which is the element to search for, and returns the first occurrence of that element.

In this case, table is a BeautifulSoup object that represents an HTML table. The find() method is being used to search for the tbody element within the table, which represents the body of the table (as opposed to the header or footer).

Once the tbody element has been found, the table object is updated to point to the tbody element rather than the original table object. This allows the script to easily work with the contents of the table body without having to deal with the table header or footer.

In [7]:

```
table=table.find("tbody")
```

Step 7 :-

The below code is using the BeautifulSoup library to find all the HTML table rows (tr) within an HTML table element.

Here is a breakdown of the code:

table is likely an instance of the BeautifulSoup Tag class that represents an HTML table element. find_all() is a method provided by the BeautifulSoup library that searches for all matching HTML elements based on specified filters. In this case, the filter is looking for all tr elements.

The second argument, style="", is an optional filter that limits the search to only those tr elements that have an empty "style" attribute. This means that the code will only select tr elements that don't have any inline styles applied to them.

The resulting output of this code will be a list of all the tr elements within the HTML table that meet the specified criteria.

In [8]:

```
rows=table.find_all("tr",style="")
```

In [9]:

```
1 print(rows)
```

[<tr><th><h3>POS</h3></th><th><h3>TEAMS</h3></th><th><h3>PLAYED</h3></th><th><h3>WON</h3></th><th><h3>LOST</h3></th><th><h3>N/R</h3></th><th><h3>TIED</h3></th><th><h3>NET RR</h3></th><th><h3>POINTS</h3></th></tr>, <tr><td>1</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/2955.png" style="width:54px;height:30px" type="image/png"></object> Gujarat Titans </td><td>9</td><td>6</td><td>3</td><td>0</td><td>0</td><td>+0.532</td><td>12</td></tr>, <tr><td>2</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/2954.png" style="width:54px;height:30px" type="image/png"></object> Lucknow Super Giants </td><td>10</td><td>5</td><td>4</td><td>1</td><td>0</td><td>+0.639</td><td>11</td></tr>, <tr><td>3</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1108.png" style="width:54px;height:30px" type="image/png"></object> Chennai Super Kings </td><td>10</td><td>5</td><td>4</td><td>1</td><td>0</td><td>+0.329</td><td>11</td></tr>, <tr><td>4</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1110.png" style="width:54px;height:30px" type="image/png"></object> Rajasthan Royals </td><td>9</td><td>5</td><td>4</td><td>0</td><td>0</td><td>+0.800</td><td>10</td></tr>, <tr><td>5</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1105.png" style="width:54px;height:30px" type="image/png"></object> Royal Challengers Bangalore </td><td>9</td><td>5</td><td>4</td><td>0</td><td>0</td><td>-0.030</td><td>10</td></tr>, <tr><td>6</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1111.png" style="width:54px;height:30px" type="image/png"></object> Mumbai Indians </td><td>9</td><td>5</td><td>4</td><td>0</td><td>0</td><td>-0.373</td><td>10</td></tr>, <tr><td>7</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1107.png" style="width:54px;height:30px" type="image/png"></object> Punjab Kings </td><td>10</td><td>5</td><td>5</td><td>0</td><td>0</td><td>-0.472</td><td>10</td></tr>, <tr><td>8</td><td> <object data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1106.png" style="width:54px;height:30px" type="image/png"></object> Kolkata Knight Riders </td><td>10</td><td>4</td><td>6</td><td>0</td><td>0</td><td>-0.103</td><td>8</td></t

```
r>, <tr><td>9</td><td><div class="super_team_name"><object data="https://x
mlns.cricketnext.com/cktnxt/scorecard/crk_player_images/flags/160x90/1379.
png" style="width:54px;height:30px" type="image/png"><img alt="Sunrisers H
yderabad" height="30px" src="https://images.news18.com/static_news18/pix/i
bnhome/news18/default-flag.jpg" width="54px"/></object><p><a href="/cricke
tnext/ipl-2023/sunrisers-hyderabad-squad-1379.html">Sunrisers Hyderabad</a
></p></div></td><td>9</td><td>3</td><td>6</td><td>0</td><td>0</td><td>-0.5
40</td><td>6</td></tr>, <tr><td>10</td><td><div class="super_team_name"><o
bject data="https://xmlns.cricketnext.com/cktnxt/scorecard/crk_player_imag
es/flags/160x90/1109.png" style="width:54px;height:30px" type="image/png">
<img alt="Delhi Capitals" height="30px" src="https://images.news18.com/sta
tic_news18/pix/ibnhome/news18/default-flag.jpg" width="54px"/></object><p>
<a href="/cricketnext/ipl-2023/delhi-capitals-squad-1109.html">Delhi Capit
als</a></p></div></td><td>9</td><td>3</td><td>6</td><td>0</td><td>0</td><t
d>-0.768</td><td>6</td></tr>]
```

Step 8:-

This code appears to be a web scraping script in Python that extracts data from HTML rows and appends them to a list named IPL_Point_Table. Here is a step-by-step explanation of what the code does:

1. It creates an empty list called IPL_Point_Table to store the extracted data.
2. It loops through each row of the HTML page, which is presumably stored in the rows variable.
3. For each row, it uses the find_all method to extract all the elements within the row and store them in a list called teams.
4. It then uses a list comprehension to extract the text content of each element in teams and store them in a list called teams_info.
5. Finally, it appends the teams_info list to the IPL_Point_Table list.
6. The script then prints the IPL_Point_Table list, which contains a list of lists where each inner list corresponds to the extracted data from each HTML row.

In summary, this code extracts data from HTML rows and stores them in a list of lists, which can then be used for further analysis or processing.

In [10]:

```
IPL_Point_Table=[]
for i in rows:
    teams=i.find_all("td")
    teams_info=[t.text for t in teams]
    IPL_Point_Table.append(teams_info)
print(IPL_Point_Table)
```

```
[[], ['1', 'Gujarat Titans', '9', '6', '3', '0', '0', '+0.532', '12'],
['2', 'Lucknow Super Giants', '10', '5', '4', '1', '0', '+0.639', '11'],
['3', 'Chennai Super Kings', '10', '5', '4', '1', '0', '+0.329', '11'],
['4', 'Rajasthan Royals', '9', '5', '4', '0', '0', '+0.800', '10'], ['5',
'Royal Challengers Bangalore', '9', '5', '4', '0', '0', '-0.030', '10'],
['6', 'Mumbai Indians', '9', '5', '4', '0', '0', '-0.373', '10'], ['7', 'P
unjab Kings', '10', '5', '5', '0', '0', '-0.472', '10'], ['8', 'Kolkata Kn
ight Riders', '10', '4', '6', '0', '0', '-0.103', '8'], ['9', 'Sunrisers H
yderabad', '9', '3', '6', '0', '0', '-0.540', '6'], ['10', 'Delhi Capital
s', '9', '3', '6', '0', '0', '-0.768', '6']]
```

Step 9 :-

The below code defines 9 empty lists, each of which has a different name:

1. POS: This list will be used to store the position of each team in a league table. Typically, the team at the top of the table will have position 1, the team in second place will have position 2, and so on.
2. Teams: This list will be used to store the names of the teams that are participating in the league.
3. Played: This list will be used to store the number of matches that each team has played in the league.
4. Won: This list will be used to store the number of matches that each team has won in the league.
5. Lost: This list will be used to store the number of matches that each team has lost in the league.
6. NR: This list will be used to store the number of matches that each team has had no result (i.e. either abandoned or tied).
7. Tied: This list will be used to store the number of matches that each team has tied in the league.
8. NET_RR: This list will be used to store the net run rate (RR) of each team. Net RR is calculated as the average runs scored per over by a team, minus the average runs scored per over against the team.
9. Points: This list will be used to store the total number of points that each team has earned in the league. Typically, teams earn 2 points for a win, 1 point for a tie, and 0 points for a loss.

In [11]:

```
POS=[]
Teams=[]
Played=[]
Won=[]
Lost=[]
NR=[]
Tied=[]
NET_RR=[]
Points=[]
```

Step 10 :-

The below code is extracting data from a list IPL_Point_Table and storing it in different lists POS, Teams, Played, Won, Lost, NR, Tied, NET_RR, and Points. It then prints the contents of each of these lists.

In [12]:

```
for i in IPL_Point_Table[1:]:
    POS.append(i[0])
    Teams.append(i[1])
    Played.append(i[2])
    Won.append(i[3])
    Lost.append(i[4])
    NR.append(i[5])
    Tied.append(i[6])
    NET_RR.append(i[7])
    Points.append(i[8])

print(POS)
print(Teams)
print(Played)
print(Won)
print(Lost)
print(NR)
print(Tied)
print(NET_RR)
print(Points)
```

```
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
['Gujarat Titans', 'Lucknow Super Giants', 'Chennai Super Kings', 'Rajasth
an Royals', 'Royal Challengers Bangalore', 'Mumbai Indians', 'Punjab King
s', 'Kolkata Knight Riders', 'Sunrisers Hyderabad', 'Delhi Capitals']
['9', '10', '10', '9', '9', '9', '10', '10', '9', '9']
['6', '5', '5', '5', '5', '5', '5', '4', '3', '3']
['3', '4', '4', '4', '4', '4', '5', '6', '6', '6']
['0', '1', '1', '0', '0', '0', '0', '0', '0', '0']
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
['+0.532', '+0.639', '+0.329', '+0.800', '-0.030', '-0.373', '-0.472', '-
0.103', '-0.540', '-0.768']
['12', '11', '11', '10', '10', '10', '10', '8', '6', '6']
```

Step 11:-

The below code is creating a Python dictionary called Point_Table with keys and values assigned to it.

Each key corresponds to a particular type of data related to the IPL point table: "POS", "Teams", "Played", "Won", "Lost", "NR", "Tied", "NET_RR", and "Points".

In [13]:

```
Point_Table={"POS":POS,
      "Teams":Teams,
      "Played":Played,
      "Won":Won,
      "Lost":Lost,
      "NR":NR,
      "Tied":Tied,
      "NET_RR":NET_RR,
      "Points":Points,
}
```

Step 12:-

In below code, we imports the Pandas library and creates a DataFrame object info using the pd.DataFrame() function. The Point_Table is assumed to be a list, tuple, or dictionary containing the data to be displayed in the DataFrame.

The pd.DataFrame() function takes the Point_Table as input and creates a tabular representation of the data in the form of rows and columns. Each row in the DataFrame corresponds to a record or observation, and each column corresponds to a variable or attribute.

The resulting DataFrame info is printed using the print() function, which displays the tabular data in a nicely formatted output. The DataFrame will contain headers for each column based on the original data in Point_Table, and the rows will be numbered starting from 0.

Pandas is a powerful library for data analysis and provides many functions to manipulate and analyze tabular data. The DataFrame() function is just one of the many functions available in Pandas for working with data. By creating a DataFrame, we can easily manipulate and analyze the data using various built-in functions in Pandas.

In [14]:

```
import pandas as pd
info = pd.DataFrame(Point_Table)
print(info)
```

```
  POS                         Teams Played Won Lost NR Tied  NET_RR Points
0   1                Gujarat Titans      9   6    3  0    0  +0.532     12
1   2          Lucknow Super Giants     10   5    4  1    0  +0.639     11
2   3           Chennai Super Kings     10   5    4  1    0  +0.329     11
3   4              Rajasthan Royals      9   5    4  0    0  +0.800     10
4   5  Royal Challengers Bangalore      9   5    4  0    0  -0.030     10
5   6                Mumbai Indians      9   5    4  0    0  -0.373     10
6   7                  Punjab Kings     10   5    5  0    0  -0.472     10
7   8         Kolkata Knight Riders     10   4    6  0    0  -0.103      8
8   9           Sunrisers Hyderabad      9   3    6  0    0  -0.540      6
9  10                Delhi Capitals      9   3    6  0    0  -0.768      6
```

Step 13:-

The Last Step of code (info.to_csv("IPL_2023_Point_Table",index=False)) is saves the contents of the info dataframe to a CSV (Comma Separated Values) file named "IPL_2023_Point_Table".

The to_csv() method is a function of the Pandas library in Python that is used to save data in a dataframe to a CSV file. The first argument passed to to_csv() is the filename and path to which the CSV file will be saved. In this case, the filename is "IPL_2023_Point_Table", which means that the CSV file will be saved in the current working directory with the specified filename.

The second argument index=False is optional and tells Pandas not to include the index column in the output CSV file. If we don't set index=False, the output CSV file will contain an extra column with the index values of the dataframe.

In summary, this line of code saves the info dataframe to a CSV file named "IPL_2023_Point_Table" without including the index column in the output.

In [15]:

```
1 info.to_csv("IPL_2023_Point_Table",index=False)
```