Project – 1 Documentation

**Scraping and Analysis of Statistics of Sports Players from Real-Time Webpage**

**OVERVIEW:** A project that gives you a better understanding of scraping data from websites and how to analyze them. Usage of various libraries as NumPy, Mat Plot, Pandas.

In the course of completing the project, you use the web scraping function, converting the extracted data into a pandas data Frame, and Storing the analyzed data.

**Problem Statement**

Web scrape basketball statistics from Wikipedia of some of the greatest basketball players and export it as a CSV file format.

**Software Requirements**

1. Programming Language: Python

2. Environment: Jupyter Notebooks / Google Collab

3. Database: CSV (export type)

4. Operation System: Windows XP or above

5. Libraries Used: Beautiful Soup, requests, Pandas, NumPy, boto 3, Matplotlib, display

1. **Open a New Notebook and import the required libraires**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | import bs4  import requests  import pandas as pd  import numpy as np  ! pip install boto3  import boto3  import matplotlib.pyplot as plt  from IPython.display import display | |  |  | | --- | --- | |  | import bs4  import requests  import pandas as pd  import numpy as np  ! pip install boto3  import boto3  import matplotlib.pyplot as plt  from IPython.display import display | |

**Description:**

Importing Libraries. When we **import** modules, we're able to call functions that are not built into **Python**. Some modules are installed as part of **Python**, and some we will install through pip. Making use of modules allows us to make our programs more robust and powerful as we're leveraging existing code

1. **Reading the webpage**

def get\_basketball\_stats(link='https://en.wikipedia.org/wiki/Michael\_Jordan'):

response = requests.get(link)

soup = bs4. BeautifulSoup (response.text, 'html.parser')

**Description:**

Starting of with a function we have taken the URL and passed to the request library and then we have received the data in the format of html which in turn passed it into the Beatuifulsoup 4 library for html.

The request module allows you to send HTTP quests in python.

1. **Main Function Process**

|  |  |
| --- | --- |
| table = soup.find(class\_='wikitable sortable')  headers = table.tr  titles = headers.find\_all('abbr')  data = {title['title']: [] for title in titles}  for row in table.find\_all('tr')[1:]:  for key, a in zip(data.keys(),row.find\_all('td')[2:]):  data[key].append(''.join(c for c in a.text if (c.isdigit() or c == '.' )))  Min = min([len(x) for x in data.values()])  for key in data.keys():  data[key] = list(map(lambda x: float(x), data[key][:Min]))  return data  **Description:**  The player stats are defined with the attribute CSS class set to ‘wikitable sortable’. By having this class, we get access to tables in webpage. We ask the soup variable to find the table. This function is very handy for finding all the values at once, but you have to check that all the information collected is relevant. Sometimes one same tag can contain completely different data. That is why it is important to be as specific as possible when choosing the tags. Therefore, we create a tag object “table” to access the given table. We create a dictionary and pass the table headers as the keys. Storing each column as a list in a dictionary, and header of the column will be the dictionary key. A for loop on dictionary iterates over its key by default. We iterate over each table row by finding each table tag(tr) and assign it to the object. Now by iterating over each cell in the table, as each cell corresponds to a different column, we all obtain the key corresponding the column using zip function. We remove unnecessary list by removing the smallest list. In order to extract only numerical values, we declare a loop in which we take each and every part i.e.; even dot as a numerical value. Since all this information is stored in the form of string, we need to convert it into float or integer type for data representation.   1. **Declaring links and names of the personals to scrap the data** |  |
| links=['https://en.wikipedia.org/wiki/Michael\_Jordan'\         ,'https://en.wikipedia.org/wiki/Kobe\_Bryant'\         ,'https://en.wikipedia.org/wiki/Ben\_Simmons'\         ,'https://en.wikipedia.org/wiki/Jayson\_Tatum'\         ,'https://en.wikipedia.org/wiki/Paul\_George'\         ,'https://en.wikipedia.org/wiki/Zion\_Williamson'\         ,'https://en.wikipedia.org/wiki/Jamal\_Murray'\         ,'https://en.wikipedia.org/wiki/Jaylen\_Brown'\         ,'https://en.wikipedia.org/wiki/Kyrie\_Irving'\         ,'https://en.wikipedia.org/wiki/Russell\_Westbrook',]  names=['Michael Jordan','Kobe Bryant','Ben simmon','Jayson Tatum','Paul george','Zion Williamson', 'Jamal Murray', 'Jaylen Brown', 'Kyrie Irving', 'Russell Westbrook'] |  |

michael\_jordan\_dict = get\_basketball\_stats(links[0])

kobe\_bryant\_dict = get\_basketball\_stats(links[1])

ben\_simmon\_dict = get\_basketball\_stats(links[2])

jayson\_tatum\_dict = get\_basketball\_stats(links[3])

paul\_george\_dict = get\_basketball\_stats(links[4])

zion\_williamson\_dict = get\_basketball\_stats(links[5])

jamal\_murray\_dict = get\_basketball\_stats(links[6])

jaylen\_brown\_dict = get\_basketball\_stats(links[7])

kyrie\_irving\_dict = get\_basketball\_stats(links[8])

russell\_westbrook\_dict = get\_basketball\_stats(links[9])

mj\_table = pd.DataFrame(michael\_jordan\_dict)

kb\_table = pd.DataFrame(kobe\_bryant\_dict)

bs\_table = pd.DataFrame(ben\_simmon\_dict)

jt\_table = pd.DataFrame(jayson\_tatum\_dict)

pg\_table = pd.DataFrame(paul\_george\_dict)

zw\_table = pd.DataFrame(zion\_williamson\_dict)

jm\_table = pd.DataFrame(jamal\_murray\_dict)

jb\_table = pd.DataFrame(jaylen\_brown\_dict)

ki\_table = pd.DataFrame(kyrie\_irving\_dict)

rw\_table = pd.DataFrame(russell\_westbrook\_dict)

list\_table =[mj\_table, kb\_table, bs\_table, jt\_table, pg\_table,zw\_table,jm\_table,jb\_table,ki\_table,rw\_table]

i = 0

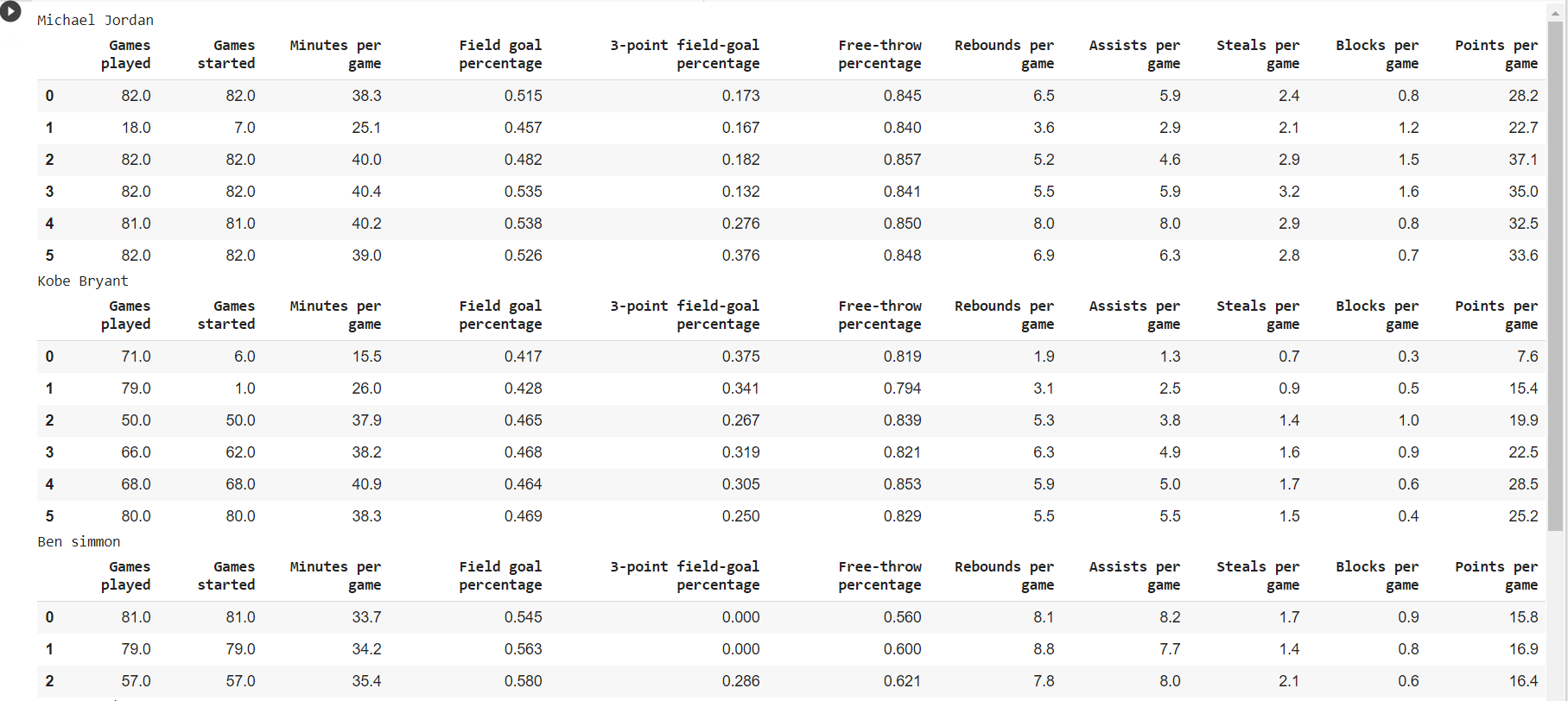
for name in names:

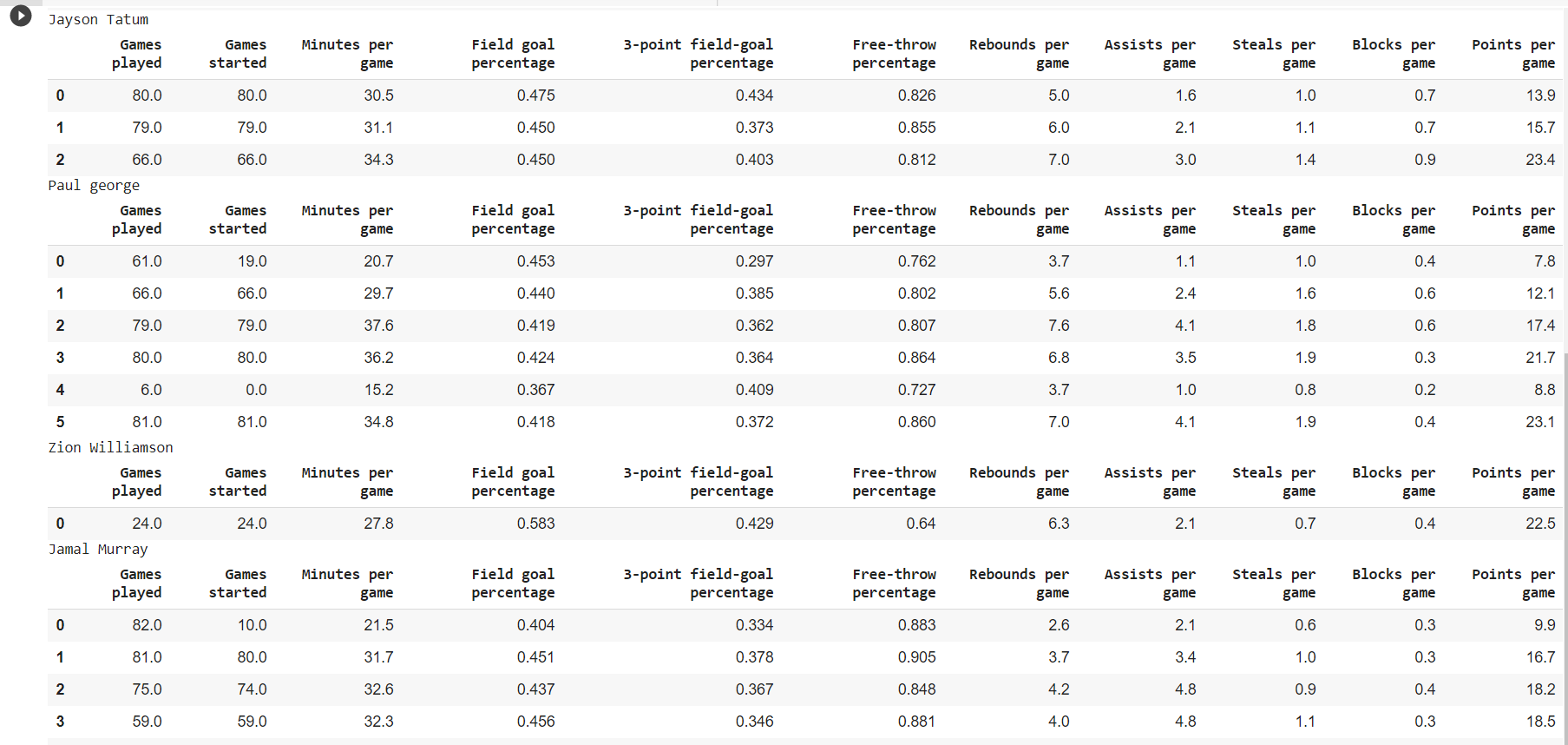
    print(name)

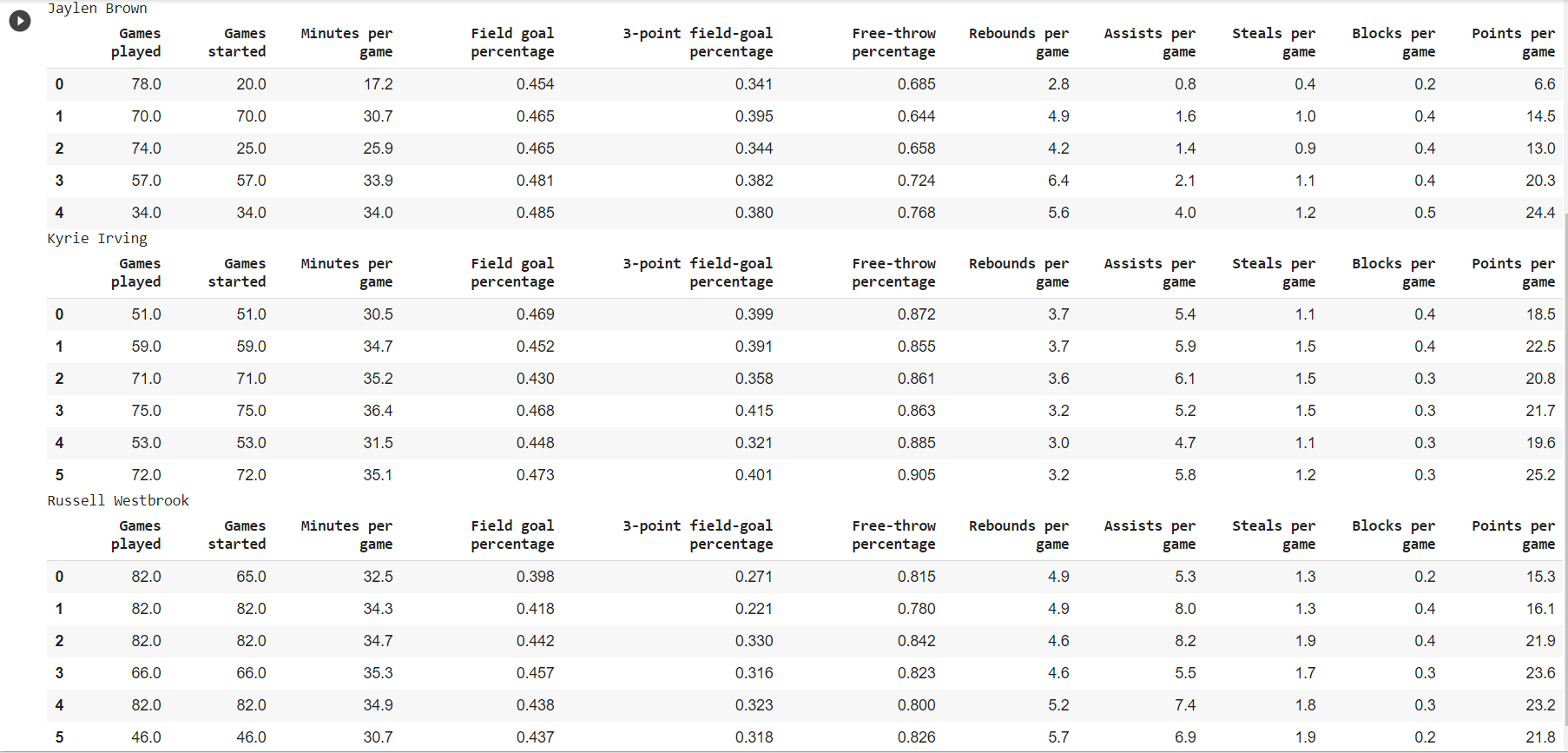
    display(list\_table[i].head(6))

    i += 1

**Output:**

****

****

****

**Description:**

We take new list and store all links of website and we create names according to links. By creating dictionary to each and every player and pass their links to function so that we get data of particular player. For representing data in data frames, we use pandas, it converts all the data in dictionaries into data frames.

**6.Making a plot using matplotlib**

j = 0

for name in names:

    plt.plot(list\_table[j][['Points per game']],label=name)

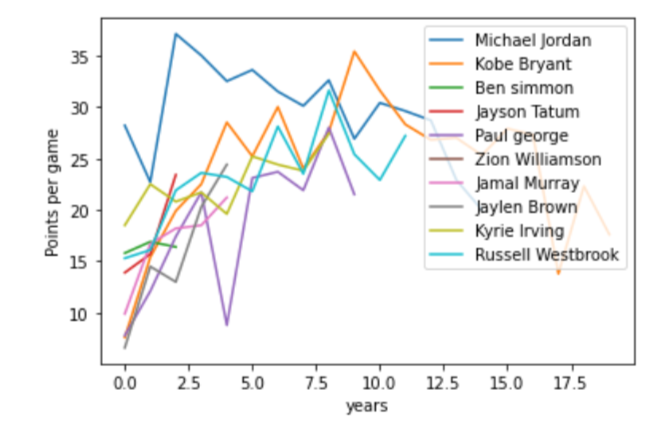
    plt.legend()

    plt.xlabel('years')

    plt.ylabel('Points per game')

    j += 1

**Output:**

****

**Description:**

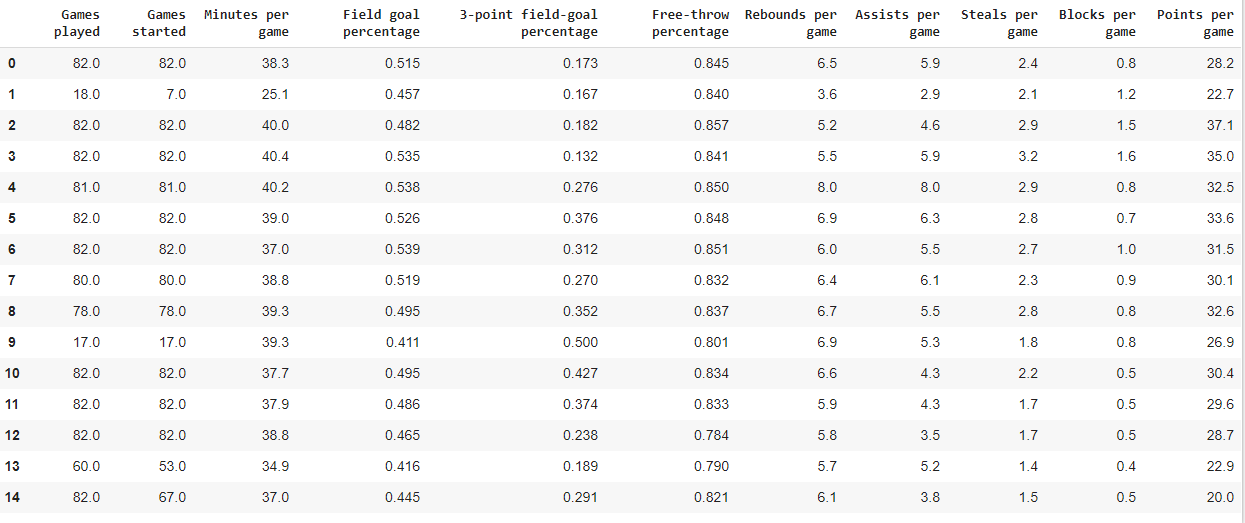
Here we plot **Points per game vs years** graph. We are visualizing the data using matplotlib.

**7.Storing the Player Statistics in Object Storage**

csv\_name = 'MJ1.csv'

mj\_table.to\_csv(csv\_name)

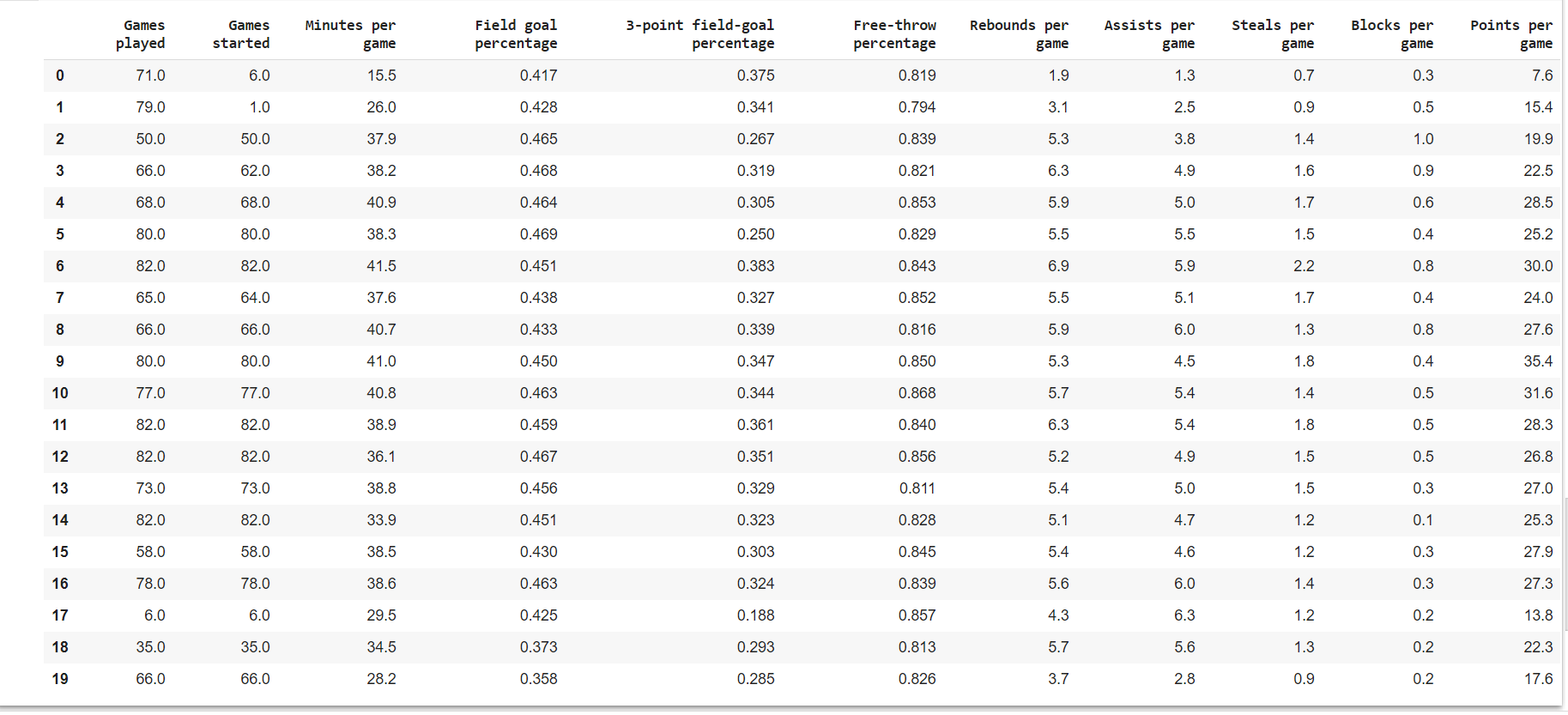
mj\_table



csv\_name1 = 'kb1.csv'

kb\_table.to\_csv(csv\_name1)

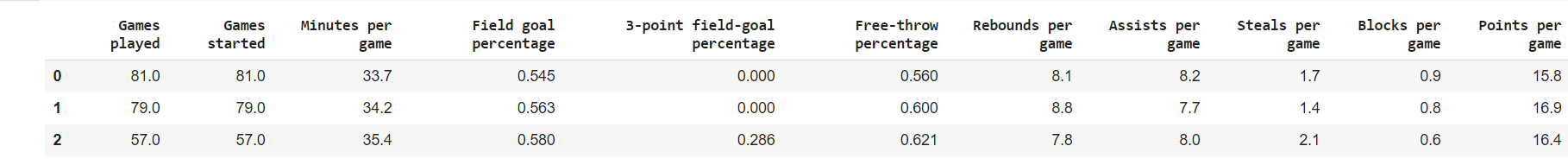
kb\_table



csv\_name2 = 'bs1.csv'

bs\_table.to\_csv(csv\_name2)

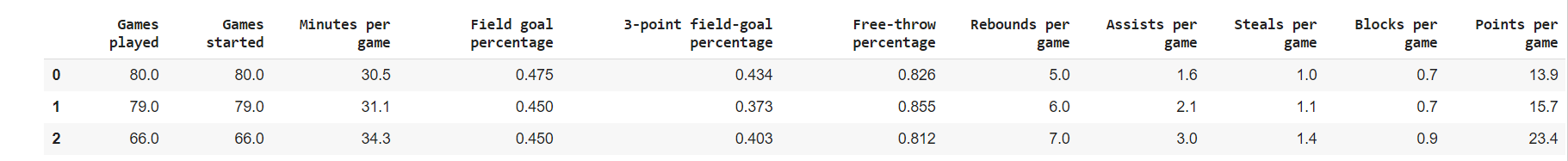
bs\_table



csv\_name3 = 'jt1.csv'

jt\_table.to\_csv(csv\_name3)

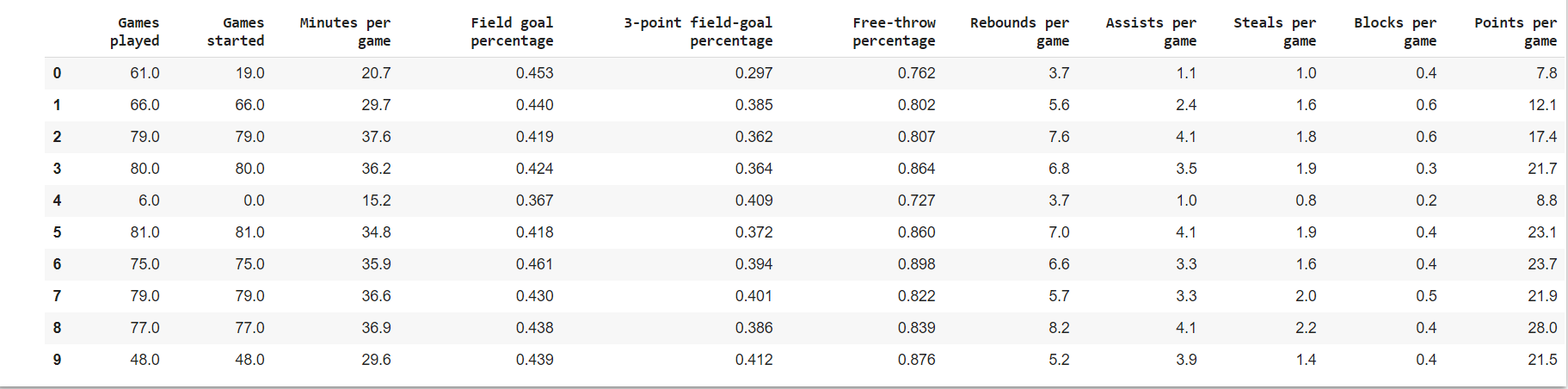
jt\_table



csv\_name4 = 'pg1.csv'

pg\_table.to\_csv(csv\_name4)

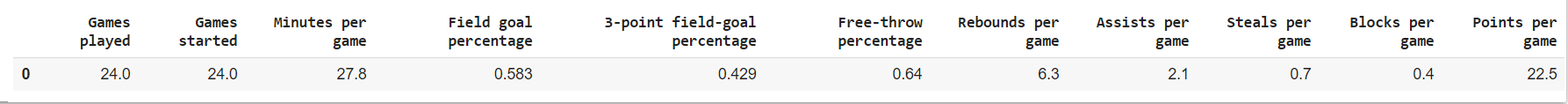
pg\_table

****

csv\_name5 = 'zw1.csv'

zw\_table.to\_csv(csv\_name5)

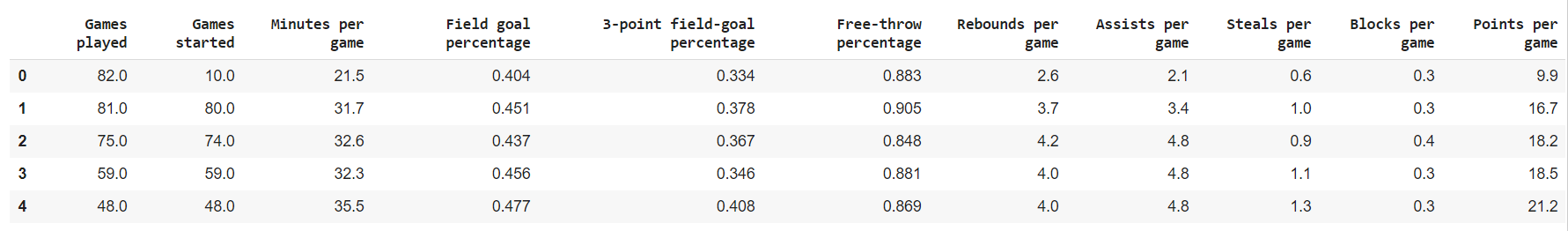
zw\_table



csv\_name6 = 'jm1.csv'

jm\_table.to\_csv(csv\_name6)

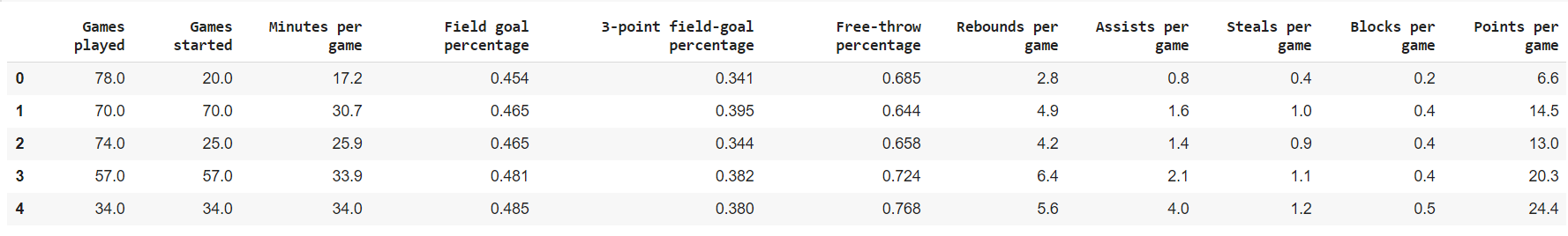
jm\_table



csv\_name7 = 'jb1.csv'

jb\_table.to\_csv(csv\_name7)

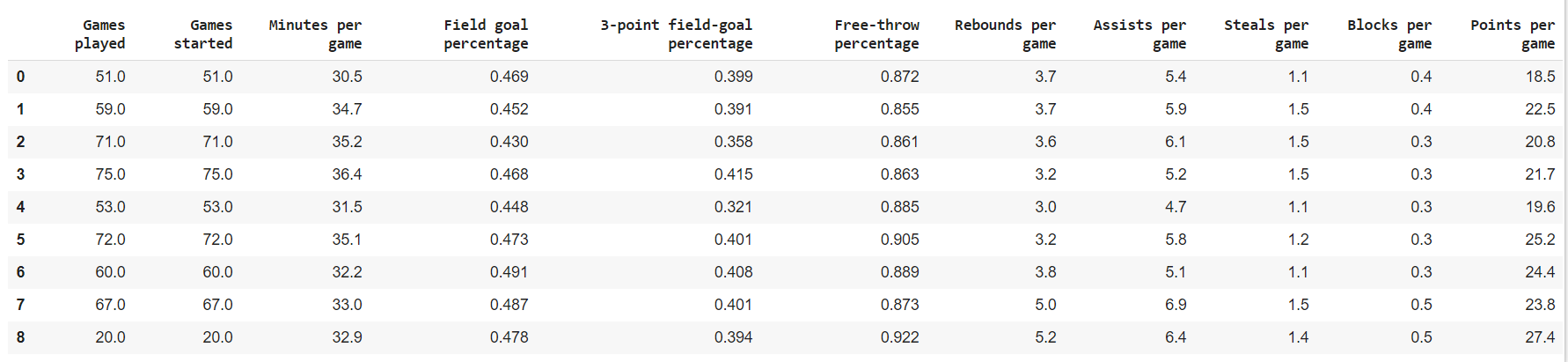
jb\_table



csv\_name8 = 'ki1.csv'

ki\_table.to\_csv(csv\_name8)

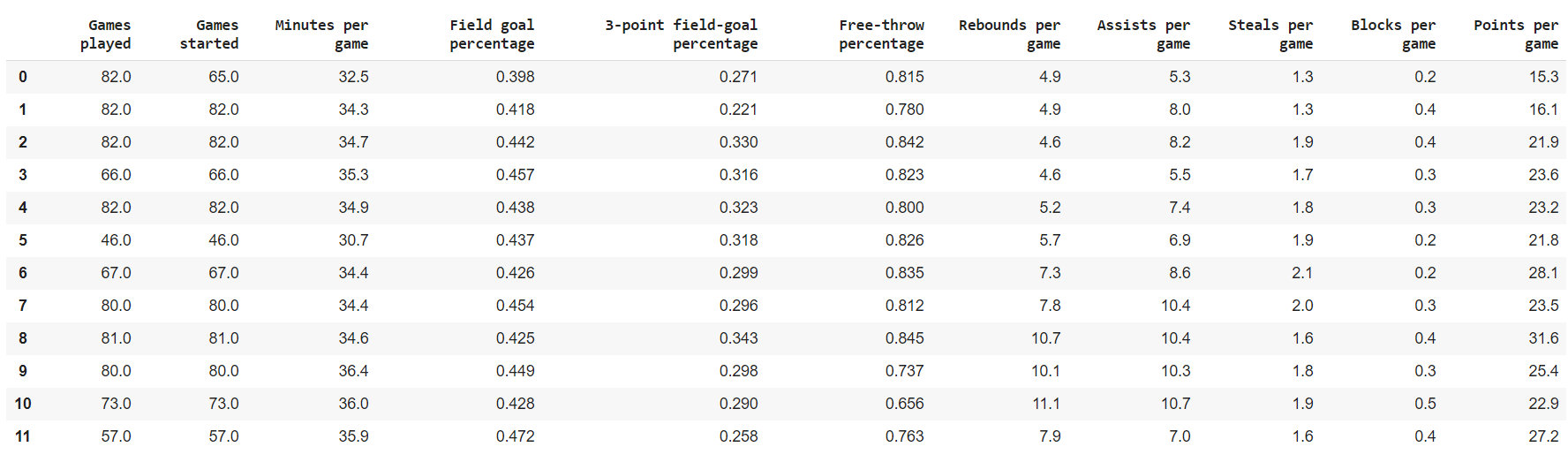
ki\_table



csv\_name9 = 'rw1.csv'

rw\_table.to\_csv(csv\_name9)

rw\_table



**Description:**

A file format is a standard way in which information is encoded for storage in a file. First, the file format specifies whether the file is a binary or ASCII file. Second, it shows how the information is organized. Here we use CSV.

Comma-separated values (CSV) file format stores tabular data in plain text. Each line in CSV file represents an observation or commonly called a record. Each record may contain one or more fields which are separated by a comma.

|  |
| --- |
|  |

**Conclusion:**

Data Scraping is widely used across numerous industries for a variety different purpose. These practices extend to market research, a better understanding of trends, research and development, and understanding customer preferences.

Collecting data from websites using an automated process. Extracting the information from table and representing the data based on their header. Data is represented in data frames where we come across the usage of panda Finally, based on the specific HTML elements you requested the web crawler to retrieve it would export those elements containing match information into a downloadable CSV file for you in milliseconds.

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |