HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA……

BÁO CÁO BÀI TẬP MÔN LẬP TRÌNH VỚI PYTHON

BÀI TẬP LỚN

Sinh viên thực hiện : **Lưu Xuân Dũng**

Mã sinh viên : **B22DCCN129**

Khoa : **Công nghệ thông tin 1**

Chuyên ngành : **Công nghệ thông tin**

Lớp : **D22CQCN09-B**

Giảng viên giảng dạy : **Kim Ngọc Bách**

HÀ NỘI- 2024

Mục lục

I. Đề bài

Bài 1: Viết chương trình Python thu thập dữ liệu phân tích cầu thủ với yêu cầu như sau:

- Thu thập dữ liệu thống kê [*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá ngoại hạng Anh mùa 2023-2024.
- Nguồn dữ liệu: https://fbref.com/en/
- Ghi kết quả ra file "cau_1.csv", bảng kết quả có cấu trúc như sau:
  + Mỗi cột tương ứng với một chỉ số.
  + Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.
  + Chỉ số nào không có hoặc không áp dụng thì để là N/a
- [*] Các chỉ số cần thu thập như sau:
  + Nation
  + Team
  + Position
  + Age
  + Playing time: matches played, starts, minutes
  + Performance: non-Penalty Goals, Penalty Goals, Assists, Yellow Cards, Red Cards
  + Expected: xG, npxG, xAG,
  + Progression: PrgC, PrgP, PrgR
  + Per 90 minutes: Gls, Ast, G+A, G-PK, G+A-PK, xG, xAG, xG + xAG, npxG, npxG + xAG
  + Goalkeeping:
    . Performance: GA, GA90, SoTA, Saves, Save%, W, D, L, CS, CS%
    . Penalty Kicks: PKatt, PKA, PKsv, PKm, Save%
  + Shooting:
    . Standard: Gls, Sh, SoT, SoT%, Sh/90, SoT/90, G/Sh, G/SoT, Dist, FK, PK, PKatt

. Expected: xG, npxG, npxG/Sh, G-xG, np:G-xG

+ Passing:

. Total: Cmp, Att, Cmp%, TotDist, PrgDist

. Short: Cmp, Att, Cmp%

. Medium: Cmp, Att, Cmp%

. Long: Cmp, Att, Cmp%

. Expected: Ast, xAG, xA, A-xAG, KP, 1/3, PPA, CrsPA, PrgP

+ Pass Types:

. Pass Types: Live, Dead, FK, TB, Sw, Crs, TI, CK

. Corner Kicks: In, Out, Str

.Outcomes: Cmp, Off, Blocks

+ Goal and Shot Creation:

. SCA: SCA, SCA90

. SCA Types: PassLive, PassDead, TO, Sh, Fld, Def

. GCA: GCA, GCA90

. GCA Types: PassLive, PassDead, TO, Sh, Fld, Def

+ Defensive Actions:

. Tackles: Tkl, TklW, Def 3rd, Mid 3rd, Att 3rd

. Challenges: Tkl, Att, Tkl%, Lost

. Blocks: Blocks, Sh, Pass, Int, Tkl + Int, Clr, Err

+ Possession:

. Touches: Touches, Def Pen, Def 3rd, Mid 3rd, Att 3rd, Att Pen, Live

. Take-Ons: Att, Succ, Succ%, Tkld, Tkld%

. Carries: Carries, TotDist, ProDist, ProgC, 1/3, CPA, Mis, Dis

. Receiving: Rec, PrgR

+ Playing Time

. Starts: Starts, Mn/Start, Compl

. Subs: Subs, Mn/Sub, unSub

. Team Success: PPM, onG, onGA

. Team Success xG: onxG, onxGA

+ Miscellaneous Stats:

. Performance: Fls, Fld, Off, Crs, OG, Recov

. Aerial Duels: Won, Lost, Won%

+ Tham khảo: https://fbref.com/en/squads/822bd0ba/2023-2024//en/squads/822bd0ba/2023-2024/Liverpool-Stats

Bài 2:

- Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.
- Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội. Ghi kết quả ra file "results2.csv", format như sau:

		Median of Attribute 1	Mean of Attribute 1	Std of Attribute 1	…	…
0	all					
1	Team 1					
…						
n	Team n					

- Vẽ histogram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.
- Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024
- Histogram Plot: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

Bài 3:

- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.
- Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có Nhận xét gì về kết quả.
- Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.
- Viết chương trình python vẽ biểu đồ radar (radar chart) so sánh cầu thủ với đầu vào như sau:

+ python radarChartPlot.py --p1 <player Name 1> --p2 <player NaAttribute <att1,att2,...,att_n>

+ --p1: là tên cầu thủ thứ nhất

+ --p2: là tên cầu thủ thứ hai

+ --Attribute: là danh sách các chỉ số cần so sánh

+ Radar chart: https://matplotlib.org/stable/gallery/specialty_plots/radar_chart.html

Bài 4:

- Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web https://www.footballtransfers.com
- Đề xuất phương pháp định giá cầu thủ.

II. Phương án giải quyết và xây dựng chương trình

1. Bài 1:

➔ Đầu tiên thêm những thư viện cần dùng:

```
import requests
from bs4 import BeautifulSoup, Comment
import pandas as pd
import numpy as np
```

- Thư viện requests được sử dụng để gửi yêu cầu HTTP tới các trang web và nhận lại dữ liệu HTML.
- Thư viện BeautifulSoup là thư viện để phân tích cú pháp HTML hoặc XML, giúp dễ dàng trích xuất các thành phần từ cấu trúc trang web, ví dụ như bảng, danh sách, liên kết.
- Comment là một lớp trong bs4 giúp lọc và xử lý các đoạn chú thích HTML, giúp loại bỏ hoặc xử lý các nội dung nằm trong chú thích nếu cần.
- Thư viện pandas là thư viện cho thao tác và phân tích dữ liệu dạng bảng, cung cấp cấu trúc DataFrame để xử lý dữ liệu dễ dàng. Trong bài, pandas được dùng để lưu dữ liệu đã trích xuất từ HTML thành các bảng dữ liệu, chỉnh sửa, xử lý và xuất dữ liệu ra tệp CSV.
- Thư viện numpy là thư viện tính toán số học, hỗ trợ các phép toán nhanh và hiệu quả trên mảng dữ liệu và ma trận. Ở đây, numpy được dùng để thực hiện các phép tính thống kê và chuẩn bị dữ liệu cho biểu đồ.

➔ Điều chỉnh cách hiển thị dữ liệu trong DataFrame khi in và xem dữ liệu trong Jupyter Notebook. Cụ thể:

- Hiển thị tất cả các dòng:

```
pd.set_option('display.max_rows', None)
```

- Hiển thị tất cả các cột:

```
pd.set_option('display.max_columns', None)
```

- Đặt độ dài tối đa của cột:

```
pd.set_option('display.max_colwidth', None)
```

➔ Lấy url của 20 đội bóng từ trang web gốc:

- Ý tưởng: từ trang web gốc sẽ lấy url riêng cho mỗi đội bóng để dễ dàng hơn trong việc truy cập và lấy dữ liệu.

- url = 'https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats'
- Gửi yêu cầu HTTP GET tới trang web đã cho và đối tượng response, chứa phản hồi từ máy chủ, sau đó dùng thư viện BeautifulSoup để phân tích và chuyển đổi nội dung HTML này thành một đối tượng soup dễ dàng cho việc truy cập và phân tích cấu trúc HTML.

```
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
```

- Tìm bảng với id cụ thể "results2023-202491_overall", và khởi tạo danh sách để lưu tên các đội bóng và liên kết.

```
table = soup.find('table', {'id': 'results2023-202491_overall'})
team_names = []
team_links = []
```

- Tìm tên tất cả các hàng trong phần thân của bảng: Tìm ô chứ tên đội bóng, sau đó loại bỏ khoảng trắng đầu cuối rồi lấy ra tên và liên kết của các đội bóng.

```
rows = table.find('tbody').find_all('tr')

for row in rows:
    team_cell = row.find('td', {'data-stat': 'team'})
    if team_cell:
        team_name = team_cell.text.strip()
        team_names.append(team_name)
        link_tag = team_cell.find('a')
        if link_tag and 'href' in link_tag.attrs:
            team_link = 'https://fbref.com' + link_tag['href']
        else:
            team_link = None
        team_links.append(team_link)
```

- Tạo DataFrame từ danh sách, lưu vào file "squad_url.csv" mà không có cột index, sau đó hiển thị nội dung DataFrame để kiểm tra:

```
df = pd.DataFrame({
    'team': team_names,
    'url': team_links
})

df.to_csv('squad_url.csv', index=False)



df
```

- Kết quả:

	team	url
0	Manchester City	https://fbref.com/en/squads/b8fd03ef/2023-2024/Manchester-City-Stats
1	Arsenal	https://fbref.com/en/squads/18bb7c10/2023-2024/Arsenal-Stats
2	Liverpool	https://fbref.com/en/squads/822bd0ba/2023-2024/Liverpool-Stats
3	Aston Villa	https://fbref.com/en/squads/8602292d/2023-2024/Aston-Villa-Stats
4	Tottenham	https://fbref.com/en/squads/361ca564/2023-2024/Tottenham-Hotspur-Stats
5	Chelsea	https://fbref.com/en/squads/cff3d9bb/2023-2024/Chelsea-Stats
6	Newcastle Utd	https://fbref.com/en/squads/b2b47a98/2023-2024/Newcastle-United-Stats
7	Manchester Utd	https://fbref.com/en/squads/19538871/2023-2024/Manchester-United-Stats
8	West Ham	https://fbref.com/en/squads/7c21e445/2023-2024/West-Ham-United-Stats
9	Crystal Palace	https://fbref.com/en/squads/47c64c55/2023-2024/Crystal-Palace-Stats
10	Brighton	https://fbref.com/en/squads/d07537b9/2023-2024/Brighton-and-Hove-Albion-Stats
11	Bournemouth	https://fbref.com/en/squads/4ba7cbea/2023-2024/Bournemouth-Stats
12	Fulham	https://fbref.com/en/squads/fd962109/2023-2024/Fulham-Stats
13	Wolves	https://fbref.com/en/squads/8cec06e1/2023-2024/Wolverhampton-Wanderers-Stats
14	Everton	https://fbref.com/en/squads/d3fd31cc/2023-2024/Everton-Stats
15	Brentford	https://fbref.com/en/squads/cd051869/2023-2024/Brentford-Stats
16	Nott'ham Forest	https://fbref.com/en/squads/e4a775cb/2023-2024/Nottingham-Forest-Stats
17	Luton Town	https://fbref.com/en/squads/e297cd13/2023-2024/Luton-Town-Stats
18	Burnley	https://fbref.com/en/squads/943e8050/2023-2024/Burnley-Stats
19	Sheffield Utd	https://fbref.com/en/squads/1df6b87e/2023-2024/Sheffield-United-Stats

➔ Chia thành 2 DataFrame để dễ dàng crawl data:

- Ý tưởng: Do nếu lấy dữ liệu của cả 20 đội thì sẽ quá nhiều yêu cầu gửi tới trang web và sẽ chỉ nhận lại được dữ liệu của tầm 12 đội bóng, do đó sẽ chia thành 2 dataframe để có thể lấy được hết dữ liệu.
- Lấy số dòng, tính chỉ số của nửa số dòng, sau đó tạo df1 là nửa số dòng đầu và df2 là nửa số dòng còn lại.

```
num_rows = len(df)
half = num_rows // 2
df1 = df.iloc[:half]
df2 = df.iloc[half:]
```

- Khởi tạo một danh sách trống để lưu trữ DataFrames của từng nhóm

```
teams_df = df1
all_data = []
```

- Lặp qua từng nhóm, gửi yêu cầu GET đến trang web của nhóm, sau đó phân tích nội dung bằng BeautifulSoup, tìm bảng số liệu thống kê của các cầu thủ:

```
for index, row in teams_df.iterrows():
    team_name = row['team']
    url = row['url']
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'id': lambda x: x and x.startswith('stats_standard')})
```

- Kiểm tra xem bảng có tồn tại không, nếu có đọc bảng HTML vào một dataframe pandas, thêm tên nhóm vào cột mới và thêm dataframe vào danh sách; nếu không có in ra "Could not find the stats table for {team_name}".
- Kết hợp tất cả các dataframe của nhóm thành một dataframe duy nhất, sau đó hiển thị dataframe kết hợp

```
combined_df = pd.concat(all_data, ignore_index=True)
combined_df.head()
```

- Lưu 10 đội đầu vào file "squads1.csv" mà không có cột index.
- Làm tương tự với 10 đội bóng sau và lưu vào file "squads2.csv".

➔ Chỉnh sửa và kết hợp rồi xuất ra file "sq.csv":

- Hiển thị 5 hàng đầu để xem và chỉnh sửa

```
sq1 = pd.read_csv("squads1.csv")
sq1.head()
```

- Loại bỏ hàng thừa đầu tiên gồm unnamed, performance,...: Lấy giá trị dòng đầu tiên làm tên cột mới, sau đó bỏ dòng đầu tiên vì nó đã trở thành tên cột, rồi hiển thị và kiểm tra lại. Làm tương tự với dataframe 2

```
sq1.columns = sq1.iloc[0]
sq1 = sq1[1:].reset_index(drop=True)
sq1.head()
```

- Kết hợp 2 dataframe sq1 và sq2 thành sq, sau đó đổi tên cột không xác định thành "Team"

```
sq = 0
sq = pd.concat([sq1, sq2], ignore_index=True)
sq = sq.rename(columns={np.nan: 'Team'})
```

```
sq.info()
```

- Xóa các dòng Squad Total và Opponent Total

```
sq = pd.read_csv("sq.csv")
sq = sq[~sq['Player'].str.contains("Squad Total|Opponent Total", na=False)]
```

- Lưu lại thông tin tất cả cầu thủ

```
sq.to_csv('sq.csv', index=False)
```

- Kiểm tra xem cột Player, Team có thể làm key cho bảng được không

```
is_unique = sq.duplicated(subset=['Player', 'Team']).sum() == 0
print(is_unique)
```

```
True
```

```
sq.columns
```

```
Index(['Player', 'Nation', 'Pos', 'Age', 'MP', 'Starts', 'Min', '90s', 'Gls',
       'Ast', 'G+A', 'G-PK', 'PK', 'PKatt', 'CrdY', 'CrdR', 'xG', 'npxG',
       'xAG', 'npxG+xAG', 'PrgC', 'PrgP', 'PrgR', 'Gls.1', 'Ast.1', 'G+A.1',
       'G-PK.1', 'G+A-PK', 'xG.1', 'xAG.1', 'xG+xAG', 'npxG.1', 'npxG+xAG.1',
       'Matches', 'Team', 'MP.1'],
      dtype='object')
```

➔ Truy cập bảng "Squad Goalkeeping" và lấy dữ liệu (2023-2024 Premier League Goalkeeper Stats | FBref.com)

- Đọc dữ liệu từ file "sq.csv" và lưu vào dataframe sq để làm việc

```
sq = pd.read_csv("sq.csv")
```

- Gửi yêu cầu GET tới trang web và phân tích cú pháp HTML của trang web

```
url_goalkeeping = 'https://fbref.com/en/comps/9/2023-2024/keepers/2023-2024-Premier-League-Stats#all_stats_keeper'
response = requests.get(url_goalkeeping)
soup = BeautifulSoup(response.content, 'html.parser')
```

- Vì một số bảng trên trang web có thể bị ẩn trong các comment HTML => Tìm tất cả các comment trong trang: Dùng vòng lặp FOR duyệt qua các comment, nếu comment có chứa bảng thì tìm bảng với id “stats_keeper”, sau đó sử dụng pandas để đọc bảng HTML thành dataframe và thoát khỏi vòng lặp khi đã tìm thấy bảng

```
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_keeper'})
        if table:
            df_goalkeeping = pd.read_html(str(table))[0]
            break
```

- Giữ lại lớp thứ 2 làm tên cột và hiển thị để kiểm tra

```
df_goalkeeping.columns = df_goalkeeping.columns.map('_'.join)

df_goalkeeping.head()
```

	Unnamed: 0_level_0_Rk	Unnamed: 1_level_0_Player	Unnamed: 2_level_0_Nation	Unnamed: 3_level_0_Pos	Unnamed: 4_level_0_Squad	Unnamed: 5_level_0_Age	Unnamed: 6_level_0_Born	Playing Time_MP	Playing Time_Starts	Playing Time_Min	Playing Time_90s	Performance_GA	Perform
0	1	Alisson	br BRA	GK	Liverpool	30	1992	28	28	2520	28.0	30	
1	2	Alphonse Areola	fr FRA	GK	West Ham	30	1993	31	31	2699	30.0	53	
2	3	Daniel Bentley	eng ENG	GK	Wolves	30	1993	5	3	383	4.3	7	
3	4	Martin Dúbravka	sk SVK	GK	Newcastle Utd	34	1989	23	22	1985	22.1	42	
4	5	Ederson	br BRA	GK	Manchester City	29	1993	33	33	2785	30.9	27	

- Đổi tên trực tiếp các cột “Player” và “Team”, sau đó chọn các cột cần thiết theo đề yêu cầu trong df_goalkeeping và kết hợp vào sq theo 2 cột chung “Player” và “Team”

```
df_goalkeeping.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
df_goalkeeping.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)

# GA, GA90, SoTA, Saves, Save%, W, D, L, CS, CS%, PKatt, PKA, PKsv, PKm, Save%
df_goalkeeping = df_goalkeeping[['Player', 'Team', 'Performance_GA',
       'Performance_GA90', 'Performance_SoTA', 'Performance_Saves',
       'Performance_Save%', 'Performance_W', 'Performance_D', 'Performance_L',
       'Performance_CS', 'Performance_CS%', 'Penalty Kicks_PKatt',
       'Penalty Kicks_PKA', 'Penalty Kicks_PKsv', 'Penalty Kicks_PKm',
       'Penalty Kicks_Save%']]
sq = sq.merge(df_goalkeeping, on=['Player', 'Team'], how='left')
```

```
sq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 746 entries, 0 to 745
Data columns (total 51 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Player               746 non-null    object 
 1   Nation               744 non-null    object 
 2   Pos                  746 non-null    object 
 3   Age                  745 non-null    float64
 4   MP                   446 non-null    float64
 5   Starts               746 non-null    int64  
 6   Min                  580 non-null    float64
 7   90s                  582 non-null    float64
 8   Gls                  582 non-null    float64
 9   Ast                  582 non-null    float64
 10  G+A                  582 non-null    float64
 11  G-PK                 582 non-null    float64
 12  PK                   582 non-null    float64
 13  PKatt                582 non-null    float64
 14  CrdY                 582 non-null    float64
 15  CrdR                 582 non-null    float64
 16  xG                   580 non-null    float64
 17  npxG                 580 non-null    float64
 18  xAG                  580 non-null    float64
 19  npxG+xAG             580 non-null    float64
...
 49  Penalty Kicks_PKm    40 non-null     object 
 50  Penalty Kicks_Save%  29 non-null     object 
dtypes: float64(30), int64(1), object(20)
memory usage: 297.4+ KB
```

- Lưu lại toàn bộ cầu thủ và Goalkeeping vào file “sq_goalkeeping.csv”

➔ Làm tương tự truy cập vào bảng "Squad Shooting" (2023-2024 Premier League Shooting Stats | FBref.com) và lưu lại vào file "sq_shooting.csv"

- url_Shooting = "https://fbref.com/en/comps/9/2023-2024/shooting/2023-2024-Premier-League-Stats#all_stats_shooting"
- id="stats_shooting"

Shooting

```
sq = pd.read_csv("sq_goalkeeping.csv")
```

```
url_Shooting = 'https://fbref.com/en/comps/9/2023-2024/shooting/2023-2024-Premier-League-Stats#all_stats_shooting'
response = requests.get(url_Shooting)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_shooting'})
        if table:
            df_Shooting = pd.read_html(str(table))[0]
            break

df_Shooting.columns = df_Shooting.columns.map('_'.join)
df_Shooting.head()
```

- Kiểm tra tên cột

```
df_Shooting.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'Standard_Gls',
       'Standard_Sh', 'Standard_SoT', 'Standard_SoT%', 'Standard_Sh/90',
       'Standard_SoT/90', 'Standard_G/Sh', 'Standard_G/SoT', 'Standard_Dist',
       'Standard_FK', 'Standard_PK', 'Standard_PKatt', 'Expected_xG',
       'Expected_npxG', 'Expected_npxG/Sh', 'Expected_G-xG',
       'Expected_np:G-xG', 'Unnamed: 25_level_0_Matches'],
      dtype='object')
```

- Đổi tên trực tiếp các cột "Player" và "Team", sau đó chọn các cột cần thiết theo đề yêu cầu trong df_Shooting và kết hợp vào sq theo 2 cột chung "Player" và "Team"

- Kiểm tra lại

```
sq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 746 entries, 0 to 745
Data columns (total 68 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Player              746 non-null    object
 1   Nation              744 non-null    object
 2   Pos                 746 non-null    object
 3   Age                 745 non-null    float64
 4   MP                  446 non-null    float64
 5   Starts              746 non-null    int64
 6   Min                 580 non-null    float64
 7   90s                 582 non-null    float64
 8   Gls                 582 non-null    float64
 9   Ast                 582 non-null    float64
 10  G+A                 582 non-null    float64
 11  G-PK                582 non-null    float64
 12  PK                  582 non-null    float64
 13  PKatt               582 non-null    float64
 14  CrdY                582 non-null    float64
 15  CrdR                582 non-null    float64
 16  xG                  580 non-null    float64
 17  npxG                580 non-null    float64
 18  xAG                 580 non-null    float64
 19  npxG+xAG            580 non-null    float64
...
 66  Expected_G-xG       580 non-null    object
 67  Expected_np:G-xG    580 non-null    object
dtypes: float64(45), int64(1), object(22)
memory usage: 396.4+ KB
```

- Lưu lại vào file “sq_shooting.csv”

```
sq.to_csv('sq_shooting.csv', index=False)
```

➔ Làm tương tự truy cập vào bảng "Squad Passing" (2023-2024 Premier League Passing Stats | FBref.com) và lưu lại vào file "sq_shooting.csv"

- url_passing = 'https://fbref.com/en/comps/9/2023-2024/passing/2023-2024-Premier-League-Stats#all_stats_passing'
- id="stats_passing"

Passing

```
sq = pd.read_csv("sq_shooting.csv")
```

```
url_passing = 'https://fbref.com/en/comps/9/2023-2024/passing/2023-2024-Premier-League-Stats#all_stats_passing'
response = requests.get(url_passing)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_passing'})
        if table:
            df_passing = pd.read_html(str(table))[0]
            break

df_passing.columns = df_passing.columns.map('_'.join)
df_passing.head()
```

```
df_passing = pd.read_html(str(table))[0]
```

	Unnamed: 0_level_0_Rk	Unnamed: 1_level_0_Player	Unnamed: 2_level_0_Nation	Unnamed: 3_level_0_Pos	Unnamed: 4_level_0_Squad	Unnamed: 5_level_0_Age	Unnamed: 6_level_0_Born	Unnamed: 7_level_0_90s	Total_Cmp	Total_Att	Total_Cmp%	Total_TotDist	Total_Pr
0	1	Max Aarons	eng ENG	DF	Bournemouth	23	2000	13.7	450	581	77.5	7402	
1	2	Joshua Acheampong	eng ENG	DF	Chelsea	17	2006	0.1	1	2	50.0	18	
2	3	Tyler Adams	us USA	MF	Bournemouth	24	1999	1.3	59	71	83.1	973	
3	4	Tosin Adarabioyo	eng ENG	DF	Fulham	25	1997	18.0	1028	1216	84.5	21010	
4	5	Elijah Adebayo	eng ENG	FW	Luton Town	25	1998	15.8	144	204	70.6	1847	

- Hiển thị tên cột để kiểm tra và sửa lại tên

```
df_passing.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'Total_Cmp',
       'Total_Att', 'Total_Cmp%', 'Total_TotDist', 'Total_PrgDist',
       'Short_Cmp', 'Short_Att', 'Short_Cmp%', 'Medium_Cmp', 'Medium_Att',
       'Medium_Cmp%', 'Long_Cmp', 'Long_Att', 'Long_Cmp%',
       'Unnamed: 22_level_0_Ast', 'Unnamed: 23_level_0_xAG', 'Expected_xA',
       'Expected_A-xAG', 'Unnamed: 26_level_0_KP', 'Unnamed: 27_level_0_1/3',
       'Unnamed: 28_level_0_PPA', 'Unnamed: 29_level_0_CrsPA',
       'Unnamed: 30_level_0_PrgP', 'Unnamed: 31_level_0_Matches'],
      dtype='object')
```

```
df_passing.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_passing.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
df_passing.rename(columns={'Unnamed: 22_level_0_Ast': 'Expected_Ast'}, inplace=True)
df_passing.rename(columns={'Unnamed: 23_level_0_xAG': 'Expected_xAG'}, inplace=True)
df_passing.rename(columns={'Unnamed: 26_level_0_KP': 'Expected_KP'}, inplace=True)
df_passing.rename(columns={'Unnamed: 27_level_0_1/3': 'Expected_1/3'}, inplace=True)
df_passing.rename(columns={'Unnamed: 28_level_0_PPA': 'Expected_PPA'}, inplace=True)
df_passing.rename(columns={'Unnamed: 29_level_0_CrsPA': 'Expected_CrsPA'}, inplace=True)
df_passing.rename(columns={'Unnamed: 30_level_0_PrgP': 'Expected_PrgP'}, inplace=True)
df_passing.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Player', 'Unnamed: 2_level_0_Nation',
       'Unnamed: 3_level_0_Pos', 'Team', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'Total_Cmp',
       'Total_Att', 'Total_Cmp%', 'Total_TotDist', 'Total_PrgDist',
       'Short_Cmp', 'Short_Att', 'Short_Cmp%', 'Medium_Cmp', 'Medium_Att',
       'Medium_Cmp%', 'Long_Cmp', 'Long_Att', 'Long_Cmp%', 'Expected_Ast',
       'Expected_xAG', 'Expected_xA', 'Expected_A-xAG', 'Expected_KP',
       'Expected_1/3', 'Expected_PPA', 'Expected_CrsPA', 'Expected_PrgP',
       'Unnamed: 31_level_0_Matches'],
      dtype='object')
```

- Chọn các cột cần thiết theo đề yêu cầu trong df_passing và kết hợp vào sq theo 2 cột chung “Player” và “Team”
- Hiển thị kết quả để kiểm tra và lưu vào file “sq_passing.csv”

```
sq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 746 entries, 0 to 745
Data columns (total 91 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Player              746 non-null    object 
 1   Nation              744 non-null    object 
 2   Pos                 746 non-null    object 
 3   Age                 745 non-null    float64
 4   MP                  446 non-null    float64
 5   Starts              746 non-null    int64  
 6   Min                 580 non-null    float64
 7   90s                 582 non-null    float64
 8   Gls                 582 non-null    float64
 9   Ast                 582 non-null    float64
 10  G+A                 582 non-null    float64
 11  G-PK                582 non-null    float64
 12  PK                  582 non-null    float64
 13  PKatt               582 non-null    float64
 14  CrdY                582 non-null    float64
 15  CrdR                582 non-null    float64
 16  xG                  580 non-null    float64
 17  npxG                580 non-null    float64
 18  xAG                 580 non-null    float64
 19  npxG+xAG            580 non-null    float64
...
 89  Expected_CrsPA      580 non-null    object 
 90  Expected_PrgP       580 non-null    object 
dtypes: float64(62), int64(1), object(28)
memory usage: 530.5+ KB
```

```
sq.to_csv('sq_passing.csv', index=False)
```

➔ Bảng Pass Types (2023-2024 Premier League Passing Stats | FBref.com)

- url_pass_type= "https://fbref.com/en/comps/9/2023-2024/passing_types/2023-2024-Premier-League-Stats#all_stats_passing_types"
- id="stats_passing_types"

```
sq = pd.read_csv("sq_passing.csv")
```

```
url_pass_type = 'https://fbref.com/en/comps/9/2023-2024/passing_types/2023-2024-Premier-League-Stats#all_stats_passing_types'
response = requests.get(url_pass_type)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_passing_types'})
        if table:
            df_pass_type = pd.read_html(str(table))[0]
            break

df_pass_type.columns = df_pass_type.columns.map('_'.join)
df_pass_type.head()
```

	Unnamed: 0_level_0_Rk	Unnamed: 1_level_0_Player	Unnamed: 2_level_0_Nation	Unnamed: 3_level_0_Pos	Unnamed: 4_level_0_Squad	Unnamed: 5_level_0_Age	Unnamed: 6_level_0_Born	Unnamed: 7_level_0_90s	Unnamed: 8_level_0_Att	Pass Types_Live	Pass Types_Dead	Pass Types_FK	Pa Ty
0	1	Max Aarons	eng ENG	DF	Bournemouth	23	2000	13.7	581	453	127	11	
1	2	Joshua Acheampong	eng ENG	DF	Chelsea	17	2006	0.1	2	1	1	0	
2	3	Tyler Adams	us USA	MF	Bournemouth	24	1999	1.3	71	67	4	3	
3	4	Tosin Adarabioyo	eng ENG	DF	Fulham	25	1997	18.0	1216	1181	33	28	
4	5	Elijah Adebayo	eng ENG	FW	Luton Town	25	1998	15.8	204	202	1	0	

- Hiển thị tên cột để kiểm tra và sửa lại

```
df_pass_type.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s',
       'Unnamed: 8_level_0_Att', 'Pass Types_Live', 'Pass Types_Dead',
       'Pass Types_FK', 'Pass Types_TB', 'Pass Types_Sw', 'Pass Types_Crs',
       'Pass Types_TI', 'Pass Types_CK', 'Corner Kicks_In', 'Corner Kicks_Out',
       'Corner Kicks_Str', 'Outcomes_Cmp', 'Outcomes_Off', 'Outcomes_Blocks',
       'Unnamed: 23_level_0_Matches'],
      dtype='object')
```

```
df_pass_type.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_pass_type.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
```

- Chọn các cột cần thiết theo đề yêu cầu trong df_pass_type và kết hợp vào sq theo 2 cột chung "Player" và "Team"

```
df_pass_type.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_pass_type.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)

df_pass_type = df_pass_type[['Player', 'Team', 'Pass Types_Live', 'Pass Types_Dead',
       'Pass Types_FK', 'Pass Types_TB', 'Pass Types_Sw', 'Pass Types_Crs',
       'Pass Types_TI', 'Pass Types_CK', 'Corner Kicks_In', 'Corner Kicks_Out',
       'Corner Kicks_Str', 'Outcomes_Cmp', 'Outcomes_Off', 'Outcomes_Blocks']]
sq = sq.merge(df_pass_type, on=['Player', 'Team'], how='left')
```

- Lưu lại vào file "sq_pass_type.csv"

```
sq.to_csv('sq_pass_types.csv', index=False)
```

➔ Bảng Squad Goal and Shot Creation (2023-2024 Premier League Goal and Shot Creation | FBref.com)

- url_goal_and_shot = "https://fbref.com/en/comps/9/2023-2024/gca/2023-2024-Premier-League-Stats#all_stats_gca"

```
sq = pd.read_csv("sq_pass_types.csv")
```

Python

```
url_goal_and_shot = 'https://fbref.com/en/comps/9/2023-2024/gca/2023-2024-Premier-League-Stats#all_stats_gca'
response = requests.get(url_goal_and_shot)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_gca'})
        if table:
            df_goal_and_shot = pd.read_html(str(table))[0]
            break

df_goal_and_shot.columns = df_goal_and_shot.columns.map('_'.join)
df_goal_and_shot.head()
```

Python

```
C:\Users\admin\AppData\Local\Temp\ipykernel_9428\4191782279.py:22: FutureWarning: Passing literal html to 'read_html' is deprecated and will be removed in a future version.
  df_goal_and_shot = pd.read_html(str(table))[0]
```

	Unnamed: 0_level_0_Rk	Unnamed: 1_level_0_Player	Unnamed: 2_level_0_Nation	Unnamed: 3_level_0_Pos	Unnamed: 4_level_0_Squad	Unnamed: 5_level_0_Age	Unnamed: 6_level_0_Born	Unnamed: 7_level_0_90s	SCA_SCA	SCA_SCA90	SCA Types_PassLive	SCA Types_PassDead
0	1	Max Aarons	eng ENG	DF	Bournemouth	23	2000	13.7	23	1.68	16	4
1	2	Joshua Acheampong	eng ENG	DF	Chelsea	17	2006	0.1	1	15.00	0	1
2	3	Tyler Adams	us USA	MF	Bournemouth	24	1999	1.3	3	2.25	2	1
3	4	Tosin Adarabioyo	eng ENG	DF	Fulham	25	1997	18.0	10	0.56	6	0
4	5	Elijah Adebayo	eng ENG	FW	Luton Town	25	1998	15.8	31	1.97	19	0

```
df_goal_and_shot.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'SCA_SCA',
       'SCA_SCA90', 'SCA Types_PassLive', 'SCA Types_PassDead', 'SCA Types_TO',
       'SCA Types_Sh', 'SCA Types_Fld', 'SCA Types_Def', 'GCA_GCA',
       'GCA_GCA90', 'GCA Types_PassLive', 'GCA Types_PassDead', 'GCA Types_TO',
       'GCA Types_Sh', 'GCA Types_Fld', 'GCA Types_Def',
       'Unnamed: 24_level_0_Matches'],
      dtype='object')
```

```
df_goal_and_shot.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_goal_and_shot.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)

df_goal_and_shot = df_goal_and_shot[['Player', 'Team', 'SCA_SCA',
       'SCA_SCA90', 'SCA Types_PassLive', 'SCA Types_PassDead', 'SCA Types_TO',
       'SCA Types_Sh', 'SCA Types_Fld', 'SCA Types_Def', 'GCA_GCA',
       'GCA_GCA90', 'GCA Types_PassLive', 'GCA Types_PassDead', 'GCA Types_TO',
       'GCA Types_Sh', 'GCA Types_Fld', 'GCA Types_Def']]
sq = sq.merge(df_goal_and_shot, on=['Player', 'Team'], how='left')
```

```
sq.info()
```

- Hiển thị tên cột để kiểm tra và sửa lại. Chọn các cột trong df_goal_and_shot theo yêu cầu của đề bài và kết hợp vào sq theo 2 cột chung “Player” và “Team”. Lưu lại vào file “sq_goal_and_shot_creation.csv”

```
sq.to_csv('sq_goal_and_shot_creation.csv', index=False)
```

➔ Bảng Squad Defensive Actions (2023-2024 Premier League Defensive Action Stats | FBref.com)

- url_defense = “https://fbref.com/en/comps/9/2023-2024/defense/2023-2024-Premier-League-Stats#all_stats_defense”

```
sq = pd.read_csv("sq_goal_and_shot_creation.csv")
```

Python

```
url_defense = 'https://fbref.com/en/comps/9/2023-2024/defense/2023-2024-Premier-League-Stats#all_stats_defense'
response = requests.get(url_defense)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_defense'})
        if table:
            df_defense = pd.read_html(str(table))[0]
            break

df_defense.columns = df_defense.columns.map('_'.join)
df_defense.head()
```

Python

```
C:\Users\admin\AppData\Local\Temp\ipykernel_9428\3810547583.py:23: FutureWarning: Passing literal html to 'read_html' is deprecated and will be removed in a future version.
  df_defense = pd.read_html(str(table))[0]
```

	Unnamed: 0_level_0_Rk	Unnamed: 1_level_0_Player	Unnamed: 2_level_0_Nation	Unnamed: 3_level_0_Pos	Unnamed: 4_level_0_Squad	Unnamed: 5_level_0_Age	Unnamed: 6_level_0_Born	Unnamed: 7_level_0_90s	Tackles_Tkl	Tackles_TklW	Tackles_Def 3rd	Tackles_Mid 3rd	Tack 3rd
0	1	Max Aarons	eng ENG	DF	Bournemouth	23	2000	13.7	29	19	20	7	
1	2	Joshua Acheampong	eng ENG	DF	Chelsea	17	2006	0.1	0	0	0	0	
2	3	Tyler Adams	us USA	MF	Bournemouth	24	1999	1.3	4	3	1	3	
3	4	Tosin Adarabioyo	eng ENG	DF	Fulham	25	1997	18.0	21	11	16	5	
4	5	Elijah Adebayo	eng ENG	FW	Luton Town	25	1998	15.8	4	1	1	0	

- Hiển thị tên và sửa đổi

```
df_defense.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'Tackles_Tkl',
       'Tackles_TklW', 'Tackles_Def 3rd', 'Tackles_Mid 3rd', 'Tackles_Att 3rd',
       'Challenges_Tkl', 'Challenges_Att', 'Challenges_Tkl%',
       'Challenges_Lost', 'Blocks_Blocks', 'Blocks_Sh', 'Blocks_Pass',
       'Unnamed: 20_level_0_Int', 'Unnamed: 21_level_0_Tkl+Int',
       'Unnamed: 22_level_0_Clr', 'Unnamed: 23_level_0_Err',
       'Unnamed: 24_level_0_Matches'],
      dtype='object')
```

```
df_defense.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_defense.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
df_defense.rename(columns={'Unnamed: 20_level_0_Int': 'Blocks_Int'}, inplace=True)
df_defense.rename(columns={'Unnamed: 21_level_0_Tkl+Int': 'Blocks_Tkl+Int'}, inplace=True)
df_defense.rename(columns={'Unnamed: 22_level_0_Clr': 'Blocks_Clr'}, inplace=True)
df_defense.rename(columns={'Unnamed: 23_level_0_Err': 'Blocks_Err'}, inplace=True)
df_defense.columns
```

- Chọn các cột cần thiết theo đề yêu cầu trong df_defense và kết hợp vào sq theo 2 cột chung “Player” và “Team”

```
df_defense = df_defense[['Player', 'Team', 'Tackles_Tkl',
       'Tackles_TklW', 'Tackles_Def 3rd', 'Tackles_Mid 3rd', 'Tackles_Att 3rd',
       'Challenges_Tkl', 'Challenges_Att', 'Challenges_Tkl%',
       'Challenges_Lost', 'Blocks_Blocks', 'Blocks_Sh', 'Blocks_Pass',
       'Blocks_Int', 'Blocks_Tkl+Int', 'Blocks_Clr', 'Blocks_Err']]
sq = sq.merge(df_defense, on=['Player', 'Team'], how='left')
```

- Lưu vào file “sq_Defensive Actions.csv”

```
sq.to_csv('sq_Defensive Actions.csv', index=False)
```

➔ Bảng Squad Possession (2023-2024 Premier League Possession Stats | FBref.com)

- url_possesion = “https://fbref.com/en/comps/9/2023-2024/possession/2023-2024-Premier-League-Stats#all_stats_possession”
- id=”stats_possession”

```
sq = pd.read_csv("sq_Defensive Actions.csv")
```

```
url_possesion = 'https://fbref.com/en/comps/9/2023-2024/possession/2023-2024-Premier-League-Stats#all_stats_possession'
response = requests.get(url_possesion)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_possession'})
        if table:
            df_possesion = pd.read_html(str(table))[0]
            break

df_possesion.columns = df_possesion.columns.map('_'.join)
df_possesion.head()
```

- Hiển thị tên cột và sửa lại

```
df_possesion.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'Touches_Touches',
       'Touches_Def Pen', 'Touches_Def 3rd', 'Touches_Mid 3rd',
       'Touches_Att 3rd', 'Touches_Att Pen', 'Touches_Live', 'Take-Ons_Att',
       'Take-Ons_Succ', 'Take-Ons_Succ%', 'Take-Ons_Tkld', 'Take-Ons_Tkld%',
       'Carries_Carries', 'Carries_TotDist', 'Carries_PrgDist', 'Carries_PrgC',
       'Carries_1/3', 'Carries_CPA', 'Carries_Mis', 'Carries_Dis',
       'Receiving_Rec', 'Receiving_PrgR', 'Unnamed: 30_level_0_Matches'],
      dtype='object')
```

```
df_possesion.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_possesion.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
```

- Chọn các cột cần thiết theo đề yêu cầu trong df_possesion và kết hợp vào sq theo 2 cột chung “Player” và “Team”. Lưu vào file “sq_Possession.csv”

```
df_possesion = df_possesion[['Player', 'Team', 'Touches_Touches',
       'Touches_Def Pen', 'Touches_Def 3rd', 'Touches_Mid 3rd',
       'Touches_Att 3rd', 'Touches_Att Pen', 'Touches_Live', 'Take-Ons_Att',
       'Take-Ons_Succ', 'Take-Ons_Succ%', 'Take-Ons_Tkld', 'Take-Ons_Tkld%',
       'Carries_Carries', 'Carries_TotDist', 'Carries_PrgDist', 'Carries_PrgC',
       'Carries_1/3', 'Carries_CPA', 'Carries_Mis', 'Carries_Dis',
       'Receiving_Rec', 'Receiving_PrgR']]
sq = sq.merge(df_possesion, on=['Player', 'Team'], how='left')
```

```
sq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 746 entries, 0 to 745
Columns: 159 entries, Player to Receiving_PrgR
dtypes: float64(131), int64(1), object(27)
memory usage: 926.8+ KB
```

```
sq.to_csv('sq_Possession.csv', index=False)
```

➔ Bảng Squad Playing Time (2023-2024 Premier League Playing Time | FBref.com)

- url_playing_time = “https://fbref.com/en/comps/9/2023-2024/playingtime/2023-2024-Premier-League-Stats#all_stats_playing_time”
- id=”stats_playing_time”

```
sq = pd.read_csv("sq_Possession.csv")
```

```
url_playing_time = 'https://fbref.com/en/comps/9/2023-2024/playingtime/2023-2024-Premier-League-Stats#all_stats_playing_time'
response = requests.get(url_playing_time)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_playing_time'})
        if table:
            df_playing_time = pd.read_html(str(table))[0]
            break

df_playing_time.columns = df_playing_time.columns.map('_'.join)
df_playing_time.head()
```

- Hiển thị tên cột và sửa lại

```
df_playing_time.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Playing Time_MP', 'Playing Time_Min',
       'Playing Time_Mn/MP', 'Playing Time_Min%', 'Playing Time_90s',
       'Starts_Starts', 'Starts_Mn/Start', 'Starts_Compl', 'Subs_Subs',
       'Subs_Mn/Sub', 'Subs_unSub', 'Team Success_PPM', 'Team Success_onG',
       'Team Success_onGA', 'Team Success_+/-', 'Team Success_+/-90',
       'Team Success_On-Off', 'Team Success (xG)_onxG',
       'Team Success (xG)_onxGA', 'Team Success (xG)_xG+/-',
       'Team Success (xG)_xG+/-90', 'Team Success (xG)_On-Off',
       'Unnamed: 29_level_0_Matches'],
      dtype='object')
```

```
df_playing_time.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_playing_time.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
```

- Chọn các cột cần thiết theo đề yêu cầu trong df_playing_time và kết hợp vào sq theo 2 cột chung “Player” và “Team”. Lưu vào file “sq_playing_time.csv”

```
df_playing_time = df_playing_time[['Player', 'Team', 'Starts_Starts', 'Starts_Mn/Start', 'Starts_Compl', 'Subs_Subs',
       'Subs_Mn/Sub', 'Subs_unSub','Team Success_PPM', 'Team Success_onG',
       'Team Success_onGA','Team Success (xG)_onxG',
       'Team Success (xG)_onxGA']]
sq = sq.merge(df_playing_time, on=['Player', 'Team'], how='left')
```

```
sq.info()
```

```
sq.to_csv('sq_playing_time.csv', index=False)
```

➔ Bảng Squad Miscellaneous Stats (2023-2024 Premier League Miscellaneous Stats | FBref.com)

- url_Miscellaneous_stats = “https://fbref.com/en/comps/9/2023-2024/misc/2023-2024-Premier-League-Stats#all_stats_misc”
- id=”stats_misc”

```
sq = pd.read_csv("sq_playing_time.csv")
```

```
url_Miscellaneous_stats = 'https://fbref.com/en/comps/9/2023-2024/misc/2023-2024-Premier-League-Stats#all_stats_misc'
response = requests.get(url_Miscellaneous_stats)
soup = BeautifulSoup(response.content, 'html.parser')
comments = soup.find_all(string=lambda text: isinstance(text, Comment))
for comment in comments:
    if 'table' in comment:
        comment_soup = BeautifulSoup(comment, 'html.parser')
        table = comment_soup.find('table', {'id': 'stats_misc'})
        if table:
            df_Miscellaneous_stats = pd.read_html(str(table))[0]
            break

df_Miscellaneous_stats.columns = df_Miscellaneous_stats.columns.map('_'.join)
df_Miscellaneous_stats.head()
```

- Hiển thị tên cột và sửa lại

```
df_Miscellaneous_stats.columns
```

```
Index(['Unnamed: 0_level_0_Rk', 'Unnamed: 1_level_0_Player',
       'Unnamed: 2_level_0_Nation', 'Unnamed: 3_level_0_Pos',
       'Unnamed: 4_level_0_Squad', 'Unnamed: 5_level_0_Age',
       'Unnamed: 6_level_0_Born', 'Unnamed: 7_level_0_90s', 'Performance_CrdY',
       'Performance_CrdR', 'Performance_2CrdY', 'Performance_Fls',
       'Performance_Fld', 'Performance_Off', 'Performance_Crs',
       'Performance_Int', 'Performance_TklW', 'Performance_PKwon',
       'Performance_PKcon', 'Performance_OG', 'Performance_Recov',
       'Aerial Duels_Won', 'Aerial Duels_Lost', 'Aerial Duels_Won%',
       'Unnamed: 24_level_0_Matches'],
      dtype='object')
```

```
df_Miscellaneous_stats.rename(columns={'Unnamed: 1_level_0_Player': 'Player'}, inplace=True)
df_Miscellaneous_stats.rename(columns={'Unnamed: 4_level_0_Squad': 'Team'}, inplace=True)
```

- Chọn các cột cần thiết theo đề yêu cầu trong df_Miscellaneous_stats và kết hợp vào sq theo 2 cột chung "Player" và "Team". Lưu vào file "sq_Miscellaneous_stats.csv"

```
df_Miscellaneous_stats = df_Miscellaneous_stats[['Player', 'Team', 'Performance_Fls',
       'Performance_Fld', 'Performance_Off', 'Performance_Crs','Performance_OG', 'Performance_Recov',
       'Aerial Duels_Won', 'Aerial Duels_Lost', 'Aerial Duels_Won%']]
sq = sq.merge(df_Miscellaneous_stats, on=['Player', 'Team'], how='left')
```

```
sq.info()
```

```
sq.to_csv('sq_Miscellaneous_stats.csv', index=False)
```

➔ Xử lí thứ tự cột và tên cột

- Đọc dữ liệu từ file "sq_Miscellaneous_stats.csv" và lưu vào dataframe sq để làm việc

```
sq = pd.read_csv("sq_Miscellaneous_stats.csv")
```

- Xóa các cột thừa

```
sq = sq.drop(['90s', 'Gls', 'G+A', 'PKatt', 'npxG+xAG', 'Matches', 'MP.1'], axis=1)
```

- Đổi tên các cột theo đề

```
sq.rename(columns={'Pos': 'Position'}, inplace=True)
sq.rename(columns={'MP': 'Playing time_matches played'}, inplace=True)
sq.rename(columns={'Starts': 'Playing time_starts'}, inplace=True)
sq.rename(columns={'Min': 'Playing time_minutes'}, inplace=True)

sq.rename(columns={'G-PK': 'Performance_non-Penalty Goals'}, inplace=True)
sq.rename(columns={'PK': 'Performance_Penalty Goals'}, inplace=True)
sq.rename(columns={'Ast': 'Performance_Assists'}, inplace=True)
sq.rename(columns={'CrdY': 'Performance_Yellow Cards'}, inplace=True)
sq.rename(columns={'CrdR': 'Performance_Red Cards'}, inplace=True)

sq.rename(columns={'xG': 'Expected_xG'}, inplace=True)
sq.rename(columns={'npxG': 'Expected_npxG'}, inplace=True)
sq.rename(columns={'xAG': 'Expected_xAG'}, inplace=True)

sq.rename(columns={'PrgC': 'Progression_PrgC'}, inplace=True)
sq.rename(columns={'PrgP': 'Progression_PrgP'}, inplace=True)
sq.rename(columns={'PrgR': 'Progression_PrgR'}, inplace=True)

sq.rename(columns={'Gls.1': 'Per 90 minutes_Gls'}, inplace=True)
sq.rename(columns={'Ast.1': 'Per 90 minutes_Ast'}, inplace=True)
sq.rename(columns={'G+A.1': 'Per 90 minutes_G+A'}, inplace=True)
sq.rename(columns={'G-PK.1': 'Per 90 minutes_G-PK'}, inplace=True)
sq.rename(columns={'G+A-PK': 'Per 90 minutes_G+A-PK'}, inplace=True)
sq.rename(columns={'xG.1': 'Per 90 minutes_xG'}, inplace=True)
sq.rename(columns={'xAG.1': 'Per 90 minutes_xAG'}, inplace=True)
sq.rename(columns={'xG+xAG': 'Per 90 minutes_xG + xAG'}, inplace=True)
sq.rename(columns={'npxG.1': 'Per 90 minutes_npxG'}, inplace=True)
sq.rename(columns={'npxG+xAG.1': 'Per 90 minutes_npxG + xAG'}, inplace=True)
```

- Xóa các kí tự trống và kí tự in thường

```
sq['Nation'] = sq['Nation'].str.replace(r'[a-z ]', '', regex=True)
sq.head()
```

- Lọc các cầu thủ thi đấu nhiều hơn 90 phút trở lên

```
sq = sq[sq['Playing time_minutes'] > 90]
sq.info()
```

- Sắp xếp các cầu thủ theo yêu cầu

```
sq = sq.sort_values(by=['Player', 'Age'], ascending=[True, False])
```

- Thay thế các giá trị trống hoặc NaN trong các cột chỉ số thành "N/a"

```
sq = sq.fillna("N/a")
```

- Hiển thị dataframe đã sắp xếp để kiểm tra

```
sq.head(10)
```

- Lưu kết quả và xuất ra file “cau_1.csv”

```
sq.to_csv('cau_1.csv', index=False)
```

2. Bài 2.

➔ Đầu tiên import thư viện cần thiết

```
import requests
from bs4 import BeautifulSoup, Comment
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
```

- Thư viện os ở bài này sẽ được dùng để kiểm tra thư mục histogram đã tồn tại chưa, nếu chưa sẽ tạo thư mục mới lưu trữ các biểu đồ histogram
- Thư viện malplotlib.pyplot sẽ được gọi khi vẽ histogram cho từng chỉ số (cột) của các cầu thủ và cho từng đội bóng

➔ Hiển thị tất cả các dòng, các cột, và đặt độ dài tối đa của cột

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)
```

➔ Đọc dữ liệu từ file “cau_1.csv” và lưu vào dataframe df để làm việc

```
df = pd.read_csv("cau_1.csv")
```

➔ Chuyển đổi từng cột thành số thực trong khi vẫn giữ nguyên giá trị “N/a”

```
columns_to_exclude = ['Player', 'Nation', 'Position', 'Team']

for col in df.columns:
    if col not in columns_to_exclude:
        df[col] = pd.to_numeric(df[col], errors='coerce')
```

➔ Chuyển đổi DataFrame trở lại để giữ lại chuỗi 'N/a' nếu chúng có trong dữ liệu gốc

```
df = df.fillna('N/a')
```

➔ Hiển thị các kiểu dữ liệu kết quả và DataFrame

```
df.dtypes
```

```
Player                              object
Nation                              object
Position                            object
Age                                float64
Playing time_matches played         object
Playing time_starts                  int64
Playing time_minutes               float64
Performance_Assists                float64
Performance_non-Penalty Goals      float64
Performance_Penalty Goals          float64
Performance_Yellow Cards           float64
Performance_Red Cards              float64
Expected_xG                        float64
Expected_npxG                      float64
Expected_xAG                       float64
Progression_PrgC                   float64
Progression_PrgP                   float64
Progression_PrgR                   float64
Per 90 minutes_Gls                 float64
Per 90 minutes_Ast                 float64
Per 90 minutes_G+A                 float64
Per 90 minutes_G-PK                float64
Per 90 minutes_G+A-PK              float64
Per 90 minutes_xG                  float64
Per 90 minutes_xAG                 float64
...
Performance_Recov                  float64
Aerial Duels_Won                   float64
Aerial Duels_Lost                  float64
Aerial Duels_Won%                   object
dtype: object
```

➔ Chuyển đổi các giá trị 'N/a' thành NaN để dễ xử lý. Lấy danh sách các cột là chỉ số (loại trừ các cột “Player”,”Nation”, “Position”, “Team”)

```
df = df.replace('N/a', pd.NA)
columns_to_check = df.columns.difference(['Player', 'Nation', 'Position', 'Team'])
```

➔ Tạo một DataFrame mới để lưu kết quả

```
result_df = pd.DataFrame(columns=columns_to_check)
```

➔ Lặp qua từng cột chỉ số: Sắp xếp các giá trị trong cột chỉ số theo thứ tự giảm dần và loại bỏ các giá trị NaN, sau đo lấy tên cầu thủ và các giá trị của họ

```
for column in columns_to_check:
    top_players = df[['Player', column]].dropna(subset=[column]).sort_values(by=column, ascending=False).head(3)
    result_df.loc['Top 1', column] = top_players.iloc[0]['Player']
    result_df.loc['Top 2', column] = top_players.iloc[1]['Player'] if len(top_players) > 1 else None
    result_df.loc['Top 3', column] = top_players.iloc[2]['Player'] if len(top_players) > 2 else None
    result_df.loc['Top 1 Value', column] = top_players.iloc[0][column]
    result_df.loc['Top 2 Value', column] = top_players.iloc[1][column] if len(top_players) > 1 else None
    result_df.loc['Top 3 Value', column] = top_players.iloc[2][column] if len(top_players) > 2 else None
```

➔ Hiển thị kết quả để kiểm tra

```
result_df
```

Python

	Aerial Duels_Lost	Aerial Duels_Won	Aerial Duels_Won%	Age	Blocks_Blocks	Blocks_Clr	Blocks_Err	Blocks_Int	Blocks_Pass	Blocks_Sh	Blocks_Tkl+Int	Carries_1/3	Carries_CPA	Carries_Carri
Top 1	Carlton Morris	James Tarkowski	Łukasz Fabiański	Łukasz Fabiański	James Tarkowski	Joachim Andersen	Wes Foderingham	Antonee Robinson	Alexis Mac Allister	James Tarkowski	João Palhinha	Rodri	Jeremy Doku	Roc
Top 2	Dominic Calvert-Lewin	Virgil van Dijk	Emiliano Martínez	Thiago Silva	Casemiro	James Tarkowski	Destiny Udogie	Lewis Cook	Tyrick Mitchell	Craig Dawson	Antonee Robinson	Martin Ødegaard	Alejandro Garnacho	Lewis Dur
Top 3	Dominic Solanke	Dominic Calvert-Lewin	Ivan Perišić	Ashley Young	Alexis Mac Allister	Murillo	José Sá	Teden Mengi	João Palhinha	Fabian Schär	Vinicius Souza	Alejandro Garnacho	Dejan Kulusevski	Pascal Grc
Top 1 Value	169.0	140.0	100.0	38.0	82.0	207.0	6.0	80.0	57.0	56.0	198.0	88.0	147.0	2513
Top 2 Value	144.0	140.0	100.0	38.0	74.0	191.0	5.0	65.0	53.0	39.0	173.0	83.0	102.0	2417
Top 3 Value	135.0	137.0	100.0	38.0	65.0	188.0	5.0	57.0	48.0	38.0	157.0	83.0	82.0	2284

➔ Tạo một bản sao của df và đặt nó vào df2, xóa 2 cột “Nation” và “Position” khỏi df2

```
df2 = df
df2 = df2.drop(['Nation', 'Position'], axis=1)
```

➔ Xác định vị trí cột “Team” và “Player”, lấy danh sách các cột chỉ số (loại trừ cột “Team” và “Player”)

```
team_col = 'Team'
player_col = 'Player'
attribute_columns = [col for col in df2.columns if col not in [team_col, player_col]]
```

➔ Khởi tạo dictionary để lưu kết quả

```
results = {'': ['all']}
```

➔ Tính toán thống kê cho tất cả các cầu thủ trong giải

```
for col in attribute_columns:
    results[f'Median of {col}'] = [df2[col].median(skipna=True)]
    results[f'Mean of {col}'] = [df2[col].mean(skipna=True)]
    results[f'Std of {col}'] = [df2[col].std(skipna=True)]
```

➔ Tính toán thống kê cho từng đội bóng

```
for team in df2[team_col].unique():
    team_df2 = df2[df2[team_col] == team]
    results[''].append(team)
    for col in attribute_columns:
        results[f'Median of {col}'].append(team_df2[col].median(skipna=True))
        results[f'Mean of {col}'].append(team_df2[col].mean(skipna=True))
        results[f'Std of {col}'].append(team_df2[col].std(skipna=True))
```

➔ Chuyển kết quả thành df2Frame, ghi kết quả ra file CSV

```
results_df = pd.DataFrame(results)
results_df.to_csv('results2.csv', index=False)
```

➔ Tạo một bản sao của df và đặt nó vào df3, xóa 2 cột “Nation” và “Position” khỏi df3

```
df3 = df
df3 = df3.drop(['Nation', 'Position'], axis=1)
```

➔ Tạo thư mục gốc "histogram" nếu chưa tồn tại

```
os.makedirs("histogram", exist_ok=True)
```

➔ Lấy danh sách các cột chỉ số (loại trừ cột “Team” và “Player”)

```
attribute_columns = [col for col in df3.columns if col not in ['Team', 'Player']]
```

➔ Chuyển các cột chỉ số sang kiểu số, các lỗi sẽ được thay thành NaN

```
df3[attribute_columns] = df3[attribute_columns].apply(pd.to_numeric, errors='coerce')
```

➔ Vẽ và lưu histogram cho toàn giải:

```
for col in attribute_columns:
    plt.figure(figsize=(4, 2.5))
    plt.hist(df3[col].dropna(), bins=20, edgecolor='black')
    plt.title(f'Phân bố {col} - Toàn giải')
    plt.xlabel(col)
    plt.ylabel('Số lượng cầu thủ')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    filename = f"histogram/{col}_ToanGiai.png"
    os.makedirs(os.path.dirname(filename), exist_ok=True)
    plt.savefig(filename)
    plt.close()
```

- Xác định tên file và đảm bảo đường dẫn tồn tại

```
filename = f"histogram/{col}_ToanGiai.png"
```

- Đảm bảo thư mục con tồn tại

```
os.makedirs(os.path.dirname(filename), exist_ok=True)
```

- Lưu hình ảnh

```
plt.savefig(filename)
plt.close()
```

➔ Vẽ và lưu histogram cho từng đội

```
for team in df3['Team'].unique():
    team_df3 = df3[df3['Team'] == team]
    for col in attribute_columns:
        plt.figure(figsize=(4, 2.5))
        plt.hist(team_df3[col].dropna(), bins=20, edgecolor='black')
        plt.title(f'Phân bố {col} - Đội {team}')
        plt.xlabel(col)
        plt.ylabel('Số lượng cầu thủ')
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        filename = f"histogram/{col}_Doi_{team}.png"
        os.makedirs(os.path.dirname(filename), exist_ok=True)
        plt.savefig(filename)
        plt.close()
```

➔ Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số

- URL của trang web: url = "https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats"
- Gửi yêu cầu đến trang web: Tìm bảng cụ thể (thường có id hoặc lớp duy nhất), điều chỉnh id nếu cần và phân tích bảng

```
response = requests.get(url)
if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')

    # Tìm bảng cụ thể (thường có id hoặc lớp duy nhất)
    table = soup.find('table', {'id': 'results2023-202491_overall'})  # Điều chỉnh id bảng nếu cần

    # Phân tích bảng
    df_overall = pd.read_html(str(table))[0]
df_overall.head()
```

➔ Tạo cột “Form” bằng cách lấy số điểm “Pts” chia cho số trận đã chơi “MP” của mỗi đội bóng hoặc cầu thủ

```
df_overall['Form'] = df_overall['Pts'] / df_overall['MP']
df_overall.head()
```

Python

	Rk	Squad	MP	W	D	L	GF	GA	GD	Pts	Pts/MP	xG	xGA	xGD	xGD/90	Attendance	Top Team Scorer	Goalkeeper	Notes	Form
0	1	Manchester City	38	28	7	3	96	34	62	91	2.39	80.5	35.6	44.9	1.18	53012	Erling Haaland - 27	Ederson	→ Champions League via league finish	2.394737
1	2	Arsenal	38	28	5	5	91	29	62	89	2.34	76.1	27.9	48.2	1.27	60236	Bukayo Saka - 16	David Raya	→ Champions League via league finish	2.342105
2	3	Liverpool	38	24	10	4	86	41	45	82	2.16	87.8	45.7	42.0	1.11	55979	Mohamed Salah - 18	Alisson	→ Champions League via league finish	2.157895
3	4	Aston Villa	38	20	8	10	76	61	15	68	1.79	63.3	59.9	3.4	0.09	41858	Ollie Watkins - 19	Emiliano Martínez	→ Champions League via league finish	1.789474
4	5	Tottenham	38	20	6	12	74	61	13	66	1.74	68.2	63.4	4.8	0.13	61482	Son Heung-min - 17	Guglielmo Vicario	→ Europa League via league finish	1.736842

➔ Hiển thị ra đội có phông độ cao nhất: Manchester city

```
highest_form = df_overall[df_overall["Form"] == df_overall["Form"].max()]
highest_form[['Squad', 'Form']]
```

	Squad	Form
0	Manchester City	2.394737

3. Bài 3.

➔ Đầu tiên import các thư viện cần thiết

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

- Nhập thuật toán K-Means từ thư viện sklearn (scikit-learn)
- Dùng thư viện matplotlib.pyplot để vẽ đồ thị và trực quan hóa dữ liệu
- Nhập StandardScaler từ thư viện sklearn.preprocessing để làm công cụ chuẩn hóa dữ liệu bằng cách đưa các giá trị về khoảng trung bình 0 và độ lệch chuẩn 1

➔ Đọc dữ liệu từ file “cau_1.csv” và lưu vào dataframe df để làm việc

```
df = pd.read_csv('cau_1.csv')
```

➔ Thay thế các giá trị 'N/a' bằng 0

```
df.replace('N/a', 0, inplace=True)
```

➔ Xóa 4 cột “Nation”, “Position”, “Team”, “Player” và kiểm tra

```
df = df.drop(['Nation', 'Position', 'Team', 'Player'], axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 493 entries, 0 to 492
Columns: 168 entries, Age to Aerial Duels_Won%
dtypes: float64(135), int64(5), object(28)
memory usage: 647.2+ KB
```

➔ Chuẩn hóa dữ liệu, chuẩn hóa tất cả các cột

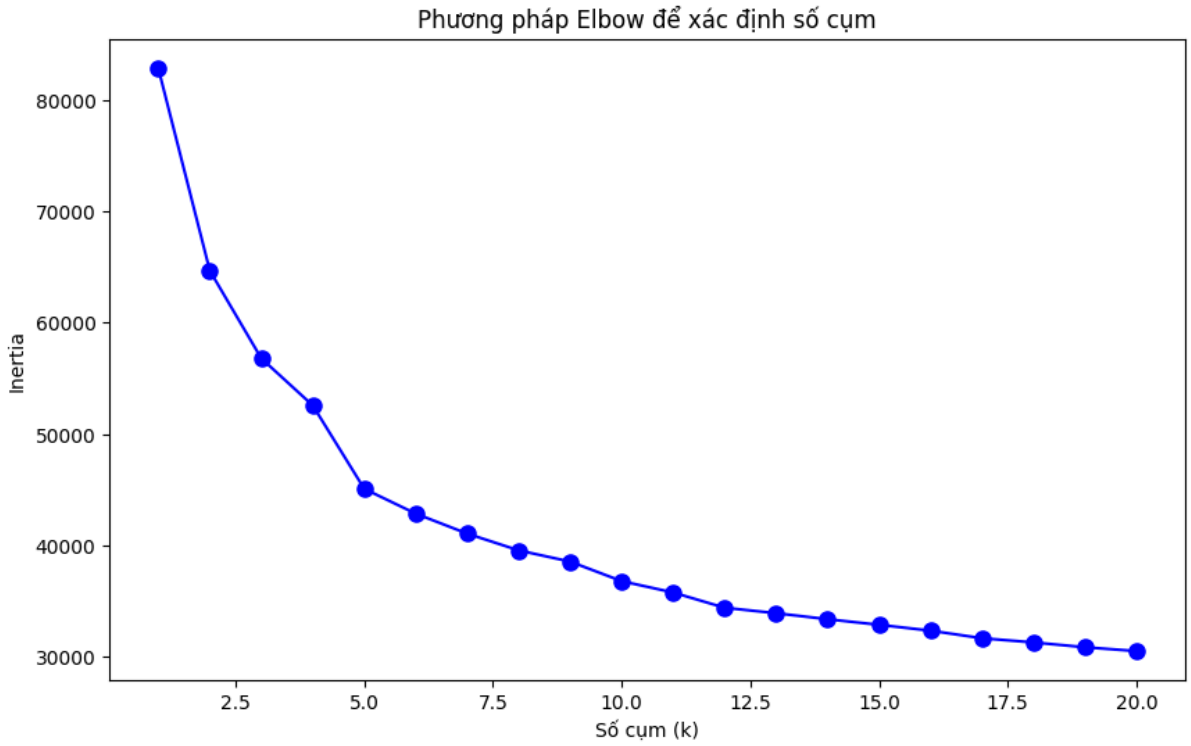
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df)
```

➔ Xác định số cụm k bằng phương pháp Elbow

```
inertia = []
K = range(1, 21)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

➔ Vẽ biểu đồ Elbow để chọn số cụm

```
plt.figure(figsize=(10, 6))
plt.plot(K, inertia, 'bo-', markersize=8)
plt.xlabel('Số cụm (k)')
plt.ylabel('Inertia')
plt.title('Phương pháp Elbow để xác định số cụm')
plt.show()
```



➔ Áp dụng K-means với số cụm tối ưu

```
optimal_k = 5
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['cluster'] = kmeans.fit_predict(scaled_features)
```

➔ Hiển thị kết quả phân cụm, in kết quả với cột "cluster" mới được thêm

```
df.head()
```

Python

	Age	Playing time_matches played	Playing time_starts	Playing time_minutes	Performance_Assists	Performance_non-Penalty Goals	Performance_Penalty Goals	Performance_Yellow Cards	Performance_Red Cards	Expected_xG	...	Performance_Fls
0	33.0	11.0	4	436.0	0.0	0.0	0.0	1.0	0.0	0.0	...	2.0
1	21.0	9.0	9	713.0	0.0	0.0	0.0	5.0	0.0	0.2	...	10.0
2	25.0	6.0	6	540.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
3	20.0	14.0	5	527.0	0.0	0.0	0.0	1.0	0.0	0.3	...	7.0
4	25.0	22.0	20	1780.0	2.0	0.0	0.0	4.0	0.0	0.1	...	17.0

5 rows × 169 columns

➔ Dựa vào phương pháp Elbow, xác định chia thành 5 cụm

- Nhập lớp PCA từ thư viện sklearn.decomposition, lớp PCA giúp giảm số chiều của dữ liệu xuống ít hơn mà vẫn giữ lại được nhiều thông tin nhất có thể, điều này giúp tăng tốc độ tính toán và đôi khi còn làm tăng độ chính xác của các mô hình
- Nhập thư viện seaborn hỗ trợ trực quan hóa dữ liệu, được xây dựng trên nền matplotlib nhưng cung cấp các biểu đồ phức tạp và thẩm mỹ hơn.

```
from sklearn.decomposition import PCA
import seaborn as sns
```

- Giảm số chiều dữ liệu xuống 2 chiều bằng PCA

```
pca = PCA(n_components=2)
pca_features = pca.fit_transform(scaled_features)
```

- Tạo DataFrame mới để dễ dàng làm việc với dữ liệu giảm chiều

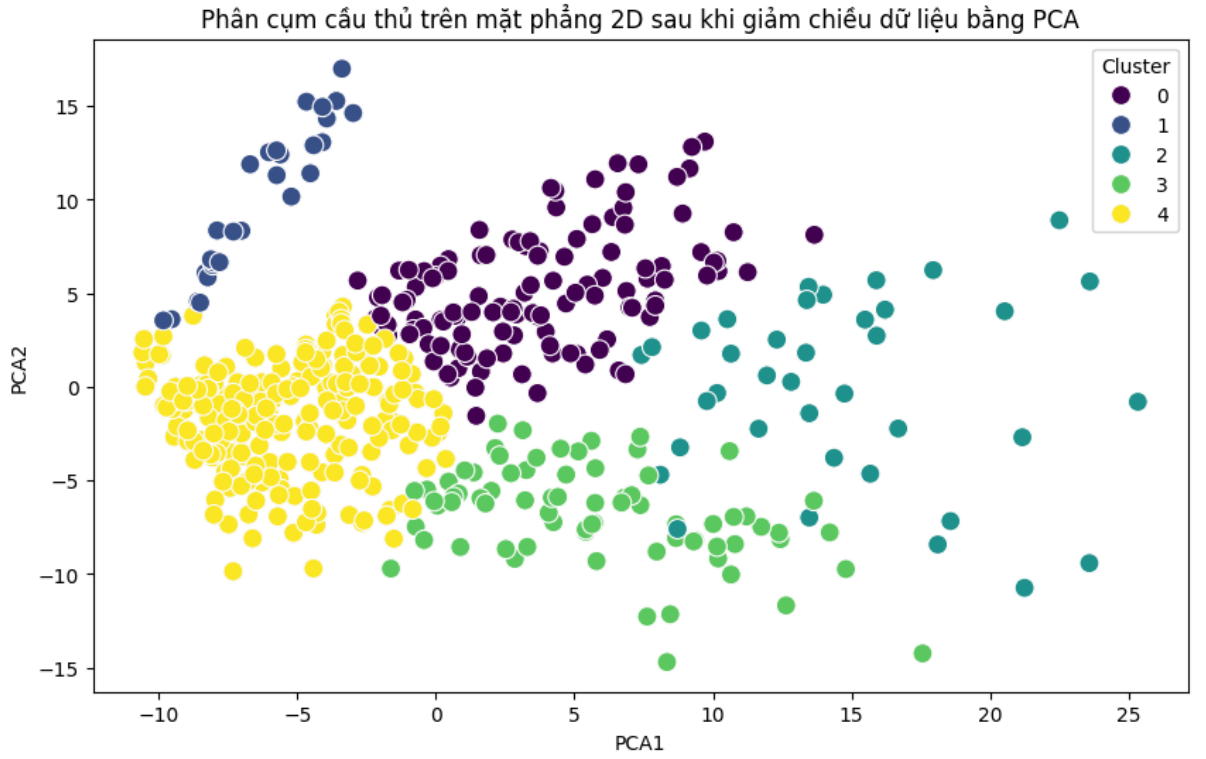
```
pca_df = pd.DataFrame(data=pca_features, columns=['PCA1', 'PCA2'])
```

- Thêm cột 'cluster' để nhận diện cụm

```
pca_df['cluster'] = df['cluster']
```

➔ Vẽ biểu đồ phân cụm trên mặt phẳng 2D

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='PCA1', y='PCA2', hue='cluster', data=pca_df, palette='viridis', s=100)
plt.title('Phân cụm cầu thủ trên mặt phẳng 2D sau khi giảm chiều dữ liệu bằng PCA')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend(title="Cluster")
plt.show()
```



➔ Import thư viện để vẽ biểu đồ radar

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import argparse
```

➔ Viết hàm vẽ biểu đồ radar

- Tạo số lượng góc cho các chỉ số

```
num_vars = len(attributes)
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
angles += angles[:1]  # Đóng vòng tròn
```

- Tạo dữ liệu cho từng cầu thủ, đóng vòng tròn cho dữ liệu

```
player1_values = player1_data[attributes].values.flatten().tolist()
player1_values += player1_values[:1]

player2_values = player2_data[attributes].values.flatten().tolist()
player2_values += player2_values[:1]
```

- Vẽ biểu đồ radar

```
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
```

- Vẽ đường và điền màu cho player 1

```
ax.plot(angles, player1_values, linewidth=2, linestyle='solid', label=player1_name)
ax.fill(angles, player1_values, alpha=0.25)
```

- Vẽ đường và điền màu cho player 2

```
ax.plot(angles, player2_values, linewidth=2, linestyle='solid', label=player2_name)
ax.fill(angles, player2_values, alpha=0.25)
```

- Cấu hình các thuộc tính

```
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(attributes, fontsize=12)
```

- Hiển thị tên cầu thủ

```
plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1))
plt.title(f'Comparison of {player1_name} vs {player2_name}', size=15, color='darkblue', weight='bold')
plt.show()
```

➔ Xử lý tham số dòng lệnh

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Compare two players using a radar chart.')
    parser.add_argument('--p1', type=str, required=True, help='Name of the first player')
    parser.add_argument('--p2', type=str, required=True, help='Name of the second player')
    parser.add_argument('--Attribute', type=str, required=True, help='Comma-separated list of attributes to compare')

    args = parser.parse_args()
    player1_name = args.p1
    player2_name = args.p2
    attributes = args.Attribute.split(',')
    df = pd.read_csv("cau_1.csv")
    if player1_name not in df['Player'].values or player2_name not in df['Player'].values:
        print("One or both players not found in the dataset.")
    else:
        # Lấy dữ liệu của hai cầu thủ
        player1_data = df[df['Player'] == player1_name].iloc[0]
        player2_data = df[df['Player'] == player2_name].iloc[0]

        # Vẽ biểu đồ radar
        radar_chart(player1_data, player2_data, attributes, player1_name, player2_name)
```

- Đọc dữ liệu từ file

```
df = pd.read_csv("cau_1.csv")
```

- Kiểm tra xem cầu thủ có trong dữ liệu không

```
if player1_name not in df['Player'].values or player2_name not in df['Player'].values:
    print("One or both players not found in the dataset.")
else:
    # Lấy dữ liệu của hai cầu thủ
    player1_data = df[df['Player'] == player1_name].iloc[0]
    player2_data = df[df['Player'] == player2_name].iloc[0]

    # Vẽ biểu đồ radar
    radar_chart(player1_data, player2_data, attributes, player1_name, player2_name)
```

III. Lời kết

➔ Trong quá trình xây dựng bài tập em cơ bản đã đạt được các mục tiêu quan trọng bao gồm:

- Đã sử dụng được các kiến thức của ngôn ngữ Python như làm việc với các thư viện khoa học dữ liệu; lập trình hướng đối tượng với các lớp như: KMeans, PCA, và StandardScaler; biểu đồ và trực quan hóa dữ liệu; giảm chiều dữ liệu;...
- Triển khai đúng với các yêu cầu logic đề ra ban đầu và hoàn thiện bài ở mức độ tối thiểu.

➔ Tuy nhiên bài tập em còn nhiều điểm còn cần phải cải thiện như:

- Vẫn chưa hoàn thành câu 4
- Tối ưu hóa code
- Đặt các điều kiện ngay từ đầu để giảm bớt lượng dữ liệu đầu vào