HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

---------- ꕥ🕮ꕥ ----------

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: LẬP TRÌNH PYTHON

Giảng viên hướng dẫn: Kim Ngọc Bách

Nhóm lớp học: 02

Nhóm tổ bài tập lớn: 08

Thành viên nhóm:

B23DCCN646 – Nguyễn Thanh Phong

B23DCCN926 – Bùi Quang Vinh

B23DCCN208 – Phạm Việt Dũng

Hà Nội, 2025

LỜI MỞ ĐẦU

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành nhất tới thầy Kim Ngọc Bách – giảng viên môn Lập trình Python của Học viện Công nghệ Bưu chính Viễn thông. Trong học kỳ này, nhờ sự hướng dẫn tận tình và tâm huyết của thầy, chúng em đã có cơ hội tiếp cận sâu hơn với kỹ thuật lập trình hiện đại - đặc biệt là khả năng thu thập, xử lý, phân tích dữ liệu thực tế bằng ngôn ngữ Python.

Dưới sự định hướng của thầy, nhóm 08 đã thực hiện đề tài: "Thu thập, phân tích và khai thác dữ liệu cầu thủ tại giải Ngoại hạng Anh mùa giải 2024 - 2025 bằng ngôn ngữ Python". Đề tài tập trung vào việc ứng dụng các thư viện như Pandas, BeautifulSoup, Selenium, SQLite,... để xây dựng một quy trình hoàn chỉnh – từ thu thập dữ liệu cầu thủ từ các trang web uy tín, lưu trữ vào cơ sở dữ liệu, đến phân tích thống kê và trực quan hóa kết quả.

Thông qua dự án, nhóm 08 đã rèn luyện được một phần nào đó về kỹ năng lập trình, khả năng xử lý dữ liệu với các mô hình kết hợp giữa Python và công nghệ web.

Bên cạnh khía cạnh học thuật, đề tài này còn mang đến nhiều cảm hứng thú vị cho một số thành viên trong nhóm 08 - là những cổ động viên nhiệt thành của bóng đá châu Âu, đặc biệt là giải bóng đá Ngoại Hạng Anh. Do đó việc được phân tích dữ liệu cầu thủ không chỉ giúp chúng em củng cố kỹ năng lập trình, mà còn tạo cơ hội kết hợp niềm đam mê thể thao với công nghệ – biến việc học thành một trải nghiệm thực tế và gần gũi hơn.

Dù đã nỗ lực hoàn thành trong khả năng tốt nhất, chúng em nhận thấy bài làm vẫn còn một số thiếu sót do hạn chế về thời gian và kinh nghiệm. Nhóm 08 rất mong nhận được sự góp ý của thầy để có thể hoàn thiện hơn trong các dự án sau.

Các thành viên nhóm xin chân thành cảm ơn thầy, kính chúc thầy luôn mạnh khỏe, hạnh phúc và thành công trong công việc và cuộc sống!

-Nhóm 08-

MỤC LỤC

PHÂN CÔNG CÔNG VIỆC

STT	Mã sinh viên	Họ tên	Nhiệm vụ
1	B23DCCN926	Bùi Quang Vinh	Thực hiện và viết báo cáo phần I. Viết chương trình Python thu thập xử lý dữ liệu phân tích cầu thủ, Tổng hợp báo cáo nhóm 08
2	B23DCCN646	Nguyễn Thanh Phong	Thực hiện và viết báo cáo phần II. Sử dụng module Python Flask, module Request và phần III.2 đề xuất định giá cầu thủ
3	B23DCCN208	Phạm Việt Dũng	Thực hiện và viết báo cáo phần III.1 và IV. về phân tích dữ liệu cầu thủ, sử dụng các thuật toán, vẽ các biểu đồ minh họa phân tích

I.

I.1.

Viết chương trình Python thu thập dữ liệu phân tích cầu thủ với yêu cầu 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 2024-2025. (Nguồn dữ liệu: https://fbref.com/en/)

- Ghi kết quả vào một bảng trong cơ sở dữ liệu SQLite, bảng kết quả có cấu trúc như sau:

    + Mỗi cột tương ứng với một chỉ số.

    + Chỉ số nào không có hoặc không áp dụng thì để là N/a

Bước 1: Lưu tên chương trình player_data.py, import các thư viện cần thiết:

```
import pandas as pd
from io import StringIO
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import sqlite3
import time
```

- import pandas as pd: thư viện dùng để xử lý và phân tích dữ liệu dạng bảng (DataFrame), hỗ trợ đọc, ghi và làm sạch dữ liệu.

- from io import StringIO: chuyển chuỗi HTML thành đối tượng giống file, giúp pandas.read_html() có thể đọc bảng từ chuỗi.

- from bs4 import BeautifulSoup: hỗ trợ phân tích cú pháp HTML, trích xuất và tìm kiếm nội dung từ mã HTML mà Selenium đã tải về.

- from selenium import webdriver: selenium là công cụ tự động hóa trình duyệt web; webdriver giúp mở trình duyệt, truy cập trang và thao tác tự động.

- from selenium.webdriver.chrome.service import Service: service điều khiển tiến trình chạy ChromeDriver, cho phép Selenium giao tiếp với trình duyệt Chrome.

- from webdriver_manager.chrome import ChromeDriverManager: tự động tải và cài đặt phiên bản ChromeDriver phù hợp với trình duyệt Chrome hiện tại trên máy.

- from selenium.webdriver.support.ui import WebDriverWait, selenium.webdriver.support import expected_conditions as EC, selenium.webdriver.common.by import By: ba module này giúp chương trình chờ trang web tải xong phần tử cần thiết (như bảng dữ liệu) trước khi trích xuất, tránh lỗi khi trang chưa load kịp.

- import sqlite3: dùng để tạo, ghi và truy vấn cơ sở dữ liệu SQLite (.sqlite) lưu kết quả cào dữ liệu.

- import time: hỗ trợ dừng chương trình để trang web kịp tải hoặc tránh bị chặn truy cập tự động.

Bước 2: Khởi tạo 2 dataframe, file SQLite lưu dữ liệu đề bài:

```
merged_dataframe = pd.DataFrame()
goalkeeping_dataframe = pd.DataFrame()
DB_NAME = 'player_data1.sqlite'
```

- Thực hiện chia 2 dataframe tách riêng goalkeeping_dataframe (cho vị trí thủ môn), merged_dataframe (cho các vị trí khác trên sân) do đặc thù vị trí thủ môn có nhiều chỉ số các vị trí khác không có và ngược lại tương tự -> sẽ dùng để phục vụ xử lý dữ liệu dạng SQLite.

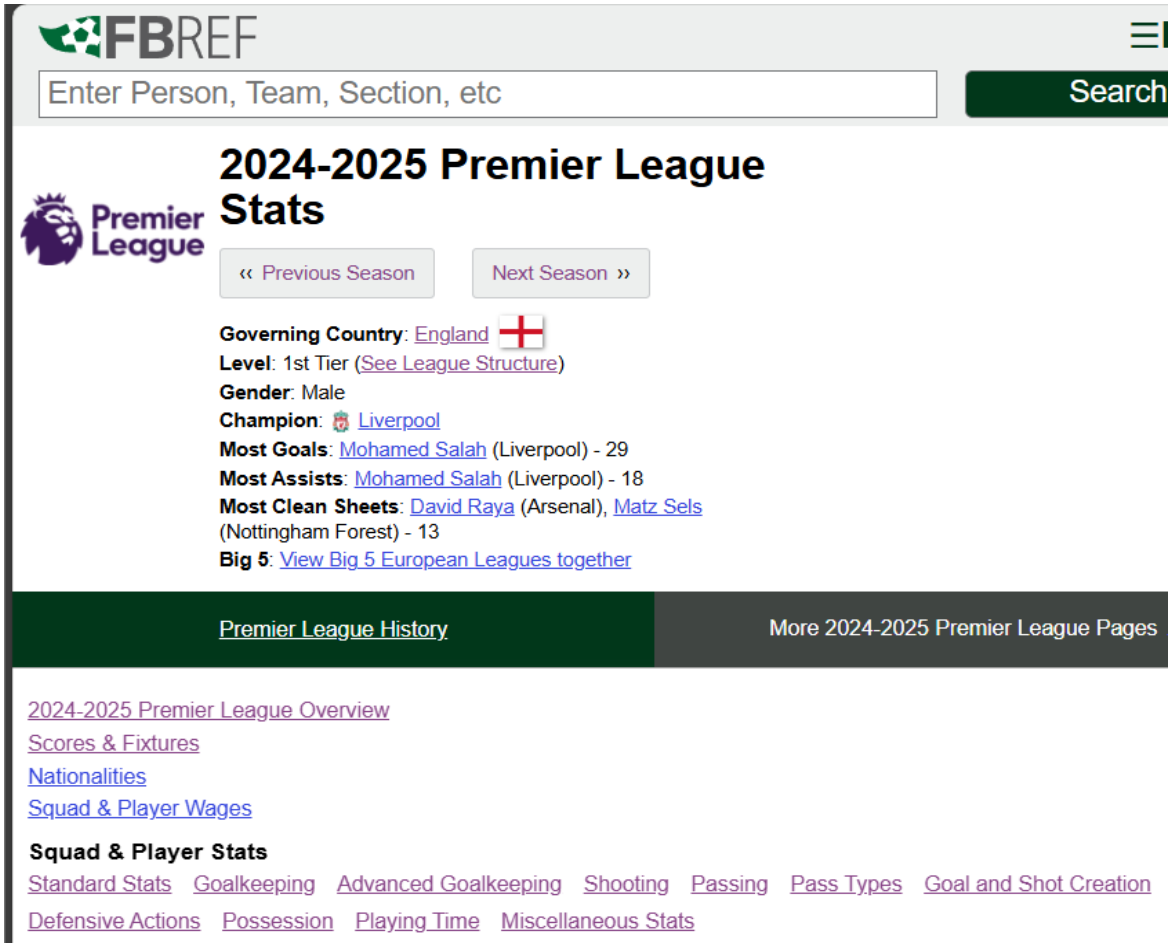
- Tạo file SQLite là player_data1.sqlite.

Bước 3: Xác định dữ liệu cần cào trên fbref.com:

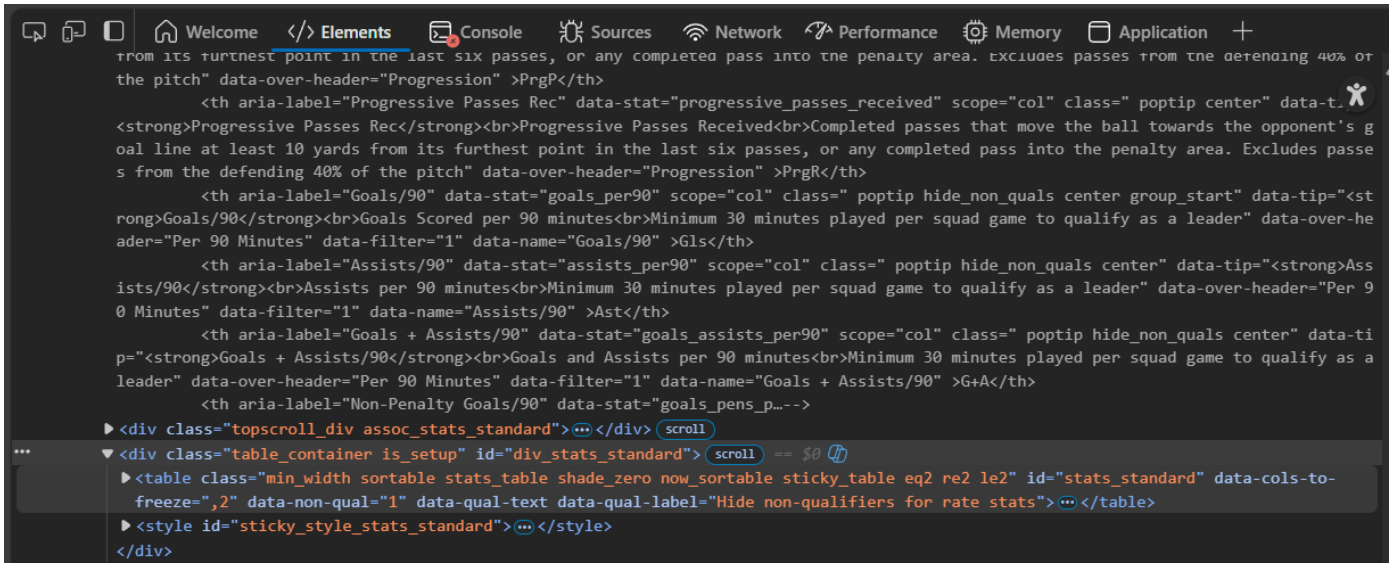
- Truy cập trang web tại địa chỉ: https://fbref.com/en/comps/9/2024-2025/2024-2025-Premier-League-Stats

- Ta sẽ thấy các bảng trong Squad & Player Stats như: Standard Stats, Goalkeeping, Shooting... -> đây chính là dữ liệu cần cào trên fbref.com để lấy thông tin các cầu thủ.

- Với từng bảng ta vào bấm vào và chọn các chỉ số phù hợp để cào dữ liệu ở chương trình python player_data.py.



- Cách để tìm thẻ id trong từng bảng với một cách làm chung cho tất cả các bảng, lấy ví dụ với bảng Standard Stats(dùng Ctrl+Shift+C và tìm thẻ id) -> tìm được thẻ id = “stats_standards”



- Với bài tập này, một số bảng như: Advanced Goalkeeping, Pass Types, Playing Time, Miscellaneous không sử dụng vì các bảng này chỉ có thêm một số chỉ số chuyên sâu của các cầu thủ -> vì chỉ cào dữ liệu cơ bản nên các bảng khác vẫn đáp ứng được đầy đủ các thông số. Sau tất cả các thao tác ta có đoạn code sau:

```
# Danh sách bảng cần cào trên FBref
table_links = {
# Các bảng Advanced Goalkeeping, Pass Types, Playing Time, Miscellaneous không sử dụng
    'Standard Stats': ('https://fbref.com/en/comps/9/2024-2025/stats/2024-2025-Premier-League-Stats', 'stats_standard'),
    'Goalkeeping': ('https://fbref.com/en/comps/9/2024-2025/keepers/2024-2025-Premier-League-Stats', 'stats_keeper'),
    'Shooting': ('https://fbref.com/en/comps/9/2024-2025/shooting/2024-2025-Premier-League-Stats', 'stats_shooting'),
    'Passing': ('https://fbref.com/en/comps/9/2024-2025/passing/2024-2025-Premier-League-Stats', 'stats_passing'),
    'Goal and Shot Creation': ('https://fbref.com/en/comps/9/2024-2025/gca/2024-2025-Premier-League-Stats', 'stats_gca'),
    'Defense': ('https://fbref.com/en/comps/9/2024-2025/defense/2024-2025-Premier-League-Stats', 'stats_defense'),
    'Possession': ('https://fbref.com/en/comps/9/2024-2025/possession/2024-2025-Premier-League-Stats', 'stats_possession'),
}
# Danh sách thuộc tính đã lựa chọn của từng bảng cho bài tập:
required_columns = [
```

```
#Trong bảng Standard Stats:
    'Player', 'Nation', 'Squad', 'Pos', 'Age', 'Playing Time Min', 'Playing Time MP',
'Playing Time Starts',
    'Performance Gls', 'Performance Ast', 'Performance Crdy', 'Performance Crdr',
      'Expected xG', 'Expected xAG', 'Progression PrgC', 'Progression PrgP',
'Progression PrgR',
    'Per 90 Minutes Gls', 'Per 90 Minutes Ast', 'Per 90 Minutes xG', 'Per 90 Minutes
xAG',
#Trong bảng Goalkeeping:
    'Performance GA90', 'Performance Save%', 'Performance CS%', 'Penalty Kicks
Save%',
#Trong bảng Shooting:
    'Standard SoT%', 'Standard SoT/90', 'Standard G/Sh', 'Standard Dist',
#Trong bảng Passing:
    'Total Cmp', 'Total Cmp%', 'Total TotDist', 'Short Cmp', 'Medium Cmp', 'Long
Cmp',
    'KP', 'Passing 1/3', 'PPA', 'CrSPA', 'PrgP',
#Trong bảng Goal and Shot Creation:
    'SCA', 'SCA SCA90', 'GCA', 'GCA GCA90',
#Trong bảng Defense:
    'Tackles Tkl', 'Tackles TklW', 'Challenges Att', 'Challenges Lost',
    'Blocks', 'Blocks Sh', 'Blocks Pass', 'Int',
#Trong bảng Possession:
    'Touches', 'Touches Def Pen', 'Touches Def 3rd', 'Touches Mid 3rd', 'Touches
Att 3rd', 'Touches Att Pen',
    'Take-Ons Att', 'Take-Ons Succ%', 'Take-Ons Tkld%',
    'Carries', 'Carries PrgDist', 'Carries PrgC', 'Carries 1/3', 'Carries CPA', 'Carries
Mis', 'Carries Dis',
    'Receiving Rec', 'Receiving PrgR',
]
```

Bước 4: Cào dữ liệu trên fbref:

```
#Cào dữ liệu trên fbref.com:
def scrape_fbref():
#Cấu hình bằng Selenium sử dụng Chrome:
    global merged_dataframe, goalkeeping_dataframe
    options = webdriver.ChromeOptions()
```

```
    try:
#Tự động tải phiên bản Chrome phù hợp và đảm bảo rằng phiên bản cài đặt phù hợp, nếu lỗi thì in thông báo:
        driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
      except Exception as e:
        print(f"Lỗi khởi tạo WebDriver: {e}")
        return False
# Duyệt qua từng bảng thống kê:
    for name, (url, table_id) in table_links.items():
# Truy cập URL tương ứng cho mỗi bảng:
        print(f\nĐang cào bảng: {name} (ID: {table_id})')
        driver.get(url)
        time.sleep(2)
# Duyệt qua từng bảng thống kê, với mỗi bảng truy cập url tương ứng:
        try:
# Chờ bảng chính được tải:, nếu bảng bị lỗi thì bỏ qua:
            WebDriverWait(driver,15).until(EC.presence_of_element_located((By.ID, table_id)))
        except Exception as e:
            print(f'Lỗi (Timeout) load bảng {name} hoặc không tìm thấy ID: {e}')
            continue
# Sử dụng BeautifulSoup trích xuất HTML bằng cách tìm các thẻ “id” trong bảng, gán table và nếu không tìm thấy thì in ra thông báo:
        soup = BeautifulSoup(driver.page_source, 'html.parser')
        table = soup.find('table', id=table_id)
        if table is None:
            print(f"Lỗi: Không tìm thấy thẻ 'table' với id='{table_id}' cho bảng {name}")
            continue
# Đọc bảng vào pandas Dataframe, nếu bị lỗi thì in ra thông báo:
        try:
            dataframe = pd.read_html(StringIO(str(table)))[0]
        except ValueError:
            print(f"Lỗi: Không thể đọc bảng {name} bằng pandas.")
            continue
```

```
 # Chuẩn hóa tên cột (nếu là MultiIndex thì gộp lại):
        if isinstance(dataframe.columns, pd.MultiIndex):
            new_cols = []
            for col in dataframe.columns:
                group = col[0].strip() if col[0].strip() and 'Unnamed' not in col[0] else ''
                subgroup = col[1].strip() if col[1].strip() else col[0].strip()
                col_name = f"{group} {subgroup}" if group and group != subgroup else
subgroup
                new_cols.append(col_name.strip())
            dataframe.columns = new_cols
        else:
            dataframe.columns = [col.strip() for col in dataframe.columns]

# Tìm và chuẩn hóa cột Player:
        player_col = next((col for col in dataframe.columns if 'player' in col.lower()
and 'rk' not in col.lower()), None)
        if not player_col:
            print(f"Lỗi: Không tìm thấy cột 'Player' trong {name}")
            continue
                    dataframe    =    dataframe.loc[dataframe[player_col].notna()    &
(dataframe[player_col] != player_col)].drop_duplicates(subset=player_col)
        dataframe = dataframe.rename(columns={player_col: 'Player'})
# Xử lý trước khi gộp dữ liệu:
        if name == 'Passing' and '1/3' in dataframe.columns:
            dataframe = dataframe.rename(columns={'1/3': 'Passing 1/3'})
        if 'G-xG' in dataframe.columns:
             dataframe = dataframe.rename(columns={'G-xG': 'Expected G-xG'})
# Nếu là vị trí thủ môn tách vào goalkeeping_dataframe:
        if name == 'Goalkeeping':
            goalkeeping_dataframe = dataframe.copy()
# Nếu là vị trí khác thì tách vào merged_dataframe:
        else:
            if merged_dataframe.empty:
                merged_dataframe = dataframe
            else:
# Loại bỏ các cột trùng lặp (trừ cột 'Player') trước khi gộp dữ liệu:
```

```
                cols_to_drop = [col for col in dataframe.columns if col in merged_dataframe.columns and col != 'Player']
            dataframe = dataframe.drop(columns=cols_to_drop, errors='ignore')
                merged_dataframe = pd.merge(merged_dataframe, dataframe, on='Player', how='outer')
    driver.quit()
    print("\n--- Hoàn thành cào dữ liệu fbref.com ---")
    return True
```

Bước 5: Làm sạch và gộp dữ liệu:

```
# Thực hiện lọc cầu thủ > 90 phút và xử lý dữ liệu thủ môn:
def clean_and_merge_data():
    global merged_dataframe, goalkeeping_dataframe
# 1. Lọc cầu thủ theo 'Playing Time Min' > 90 ở tất cả các vị trí:
    min_col = next((col for col in merged_dataframe.columns if 'playing time min' in col.lower()), None)
    if min_col:
                                                merged_dataframe = merged_dataframe[merged_dataframe[min_col].notna()].copy()
        try:
                                            merged_dataframe.loc[:, min_col] = merged_dataframe[min_col].astype(str).str.replace(',', '', regex=False).astype(float)
            merged_dataframe = merged_dataframe[merged_dataframe[min_col] > 90].copy()
            print(f"Lọc được {len(merged_dataframe)} cầu thủ chơi > 90 phút.")
        except Exception as e:
            print(f"Lỗi lọc 'Playing Time Min': {e}")
            return False
    else:
        print("Lỗi: Không tìm thấy cột 'Playing Time Min'")
        return False

# 2. Xử lý dữ liệu thủ môn và gộp dữ liệu:
    if not goalkeeping_dataframe.empty:
#Tìm kiếm tên cột Vị trí (Pos) trong bảng chính (merged_dataframe)
```

```
        pos_col = next((col for col in merged_dataframe.columns if 'pos' in col.lower()), None)
    if pos_col:
# Chỉ giữ lại thủ môn đã chơi > 90 phút:
        goalkeeping_dataframe = goalkeeping_dataframe[goalkeeping_dataframe['Player'].isin(merged_dataframe['Player'])].copy()
# Lấy vị trí từ bảng chính để lọc chỉ số thủ môn:
        pos_mapping = merged_dataframe.set_index('Player')[pos_col].to_dict()
# Loại bỏ các chỉ số thủ môn nếu cầu thủ là tiền đạo, tiền vệ, hậu vệ còn lại thì giữ nguyên:
        for col in goalkeeping_dataframe.columns:
            if col not in ['Player', 'Pos']:
#Logic phân loại cầu thủ (chuỗi “GK” hay “DF”/”MF”/”FW”)
                goalkeeping_dataframe.loc[:, col] = goalkeeping_dataframe.apply(
                    lambda row: row[col] if 'GK' in pos_mapping.get(row['Player'], '') else pd.NA, axis=1
                )
        cols_to_drop_keeper = [col for col in goalkeeping_dataframe.columns if col in merged_dataframe.columns and col != 'Player']
                        goalkeeping_dataframe.drop(columns=cols_to_drop_keeper, inplace=True, errors='ignore')


# Gộp dữ liệu thủ môn vào bảng chính
                            merged_dataframe = pd.merge(merged_dataframe, goalkeeping_dataframe, on='Player', how='left')
        print(f"Đã xử lý dữ liệu thủ môn.")
    else:
        print("Lỗi: Không tìm thấy cột 'Pos' để xử lý dữ liệu thủ môn")

# 3. Chọn và sắp xếp các cột cuối cùng
    available_columns = [col for col in required_columns if col in merged_dataframe.columns]
  must_have = ['Player', 'Nation', 'Squad', 'Pos', 'Age']
   final_columns = must_have + [col for col in available_columns if col not in must_have]
```

```
# Giữ lại các cột có trong danh sách final_columns bắt buộc phải có
    merged_dataframe = merged_dataframe[[col for col in final_columns if col in
merged_dataframe.columns]]
# Điền giá trị thiếu bằng 'N/a' theo yêu cầu
    merged_dataframe.fillna('N/a', inplace=True)
    return True
```

Bước 6: Ghi kết quả ra sqlite:

```
#Ghi ra player_data1.sqlite3
if __name__ == "__main__":
    if scrape_fbref():
        if clean_and_merge_data():
            conn = sqlite3.connect(DB_NAME)
                    merged_dataframe.to_sql('player_data1',  conn,  if_exists='replace',
index=False)
            conn.close()
            print("Ghi cầu thủ vào bảng player_data1.sqlite thành công!")
```

Kết quả đạt được:

```
Đang cào bảng: Standard Stats (ID: stats_standard)

Đang cào bảng: Goalkeeping (ID: stats_keeper)

Đang cào bảng: Shooting (ID: stats_shooting)

Đang cào bảng: Passing (ID: stats_passing)

Đang cào bảng: Goal and Shot Creation (ID: stats_gca)

Đang cào bảng: Defense (ID: stats_defense)

Đang cào bảng: Possession (ID: stats_possession)

--- Hoàn thành cào dữ liệu fbref.com ---
Lọc được 491 cầu thủ chơi > 90 phút.
Đã xử lý dữ liệu thủ môn.
d:\Coding\PYTHON_CODEPTIT\tempCodeRunnerFile.py:175:
Call result.infer_objects(copy=False) instead. To opt
  merged_dataframe.fillna('N/a', inplace=True)
Ghi cầu thủ vào bảng player_data1.sqlite thành công!
PS D:\Coding\PYTHON_CODEPTIT>
```

player_data1.sqlite

Rows: 491

TABLES

play...

	Player	N...	Squad	P.	Age	Playing Time Min	P.	P..	Perfor...
1	Aaron Cresswell	eng ENG	West Ham	DF	34	824	18	10	0
2	Aaron Ramsda...	eng ENG	Southampton	GK	26	2700	30	30	0
3	Aaron Wan-Bi...	cd COD	West Ham	DF	26	3154	36	35	2
4	Abdoulaye Do...	ml MLI	Everton	MF	31	2564	33	31	3
5	Abdukodir Kh...	uz UZB	Manchester ...	DF	20	503	6	6	0
6	Abdul Fatawu ...	gh GHA	Leicester City	FW	20	579	11	6	0
7	Adam Armstro...	eng ENG	Southampton	FW,MF	27	1248	20	15	2
8	Adam Lallana	eng ENG	Southampton	MF	36	361	14	5	0
9	Adam Smith	eng ENG	Bournemouth	DF	33	1590	25	19	0
1	Adam Webster	eng ENG	Brighton	DF	29	887	14	11	0
1	Adam Wharton	eng ENG	Crystal Palace	MF	20	1318	20	16	0
1	Adama Traoré	es ESP	Fulham	FW,MF	28	1772	36	18	2
1	Albert Grønba...	dk DEN	Southampton	FW,MF	23	143	4	2	0
1	Alejandro Gar...	ar ARG	Manchester ...	MF,FW	20	2199	36	23	6
1	Alex Iwobi	ng NGA	Fulham	FW,MF	28	2981	38	35	9
	Alex McCarthy	eng ENG	Southampton	GK	34	450	5	5	0

I.2.

- Với mỗi cầu thủ được ghi trong bảng trên, thu thập giá chuyển nhượng của cầu thủ trong mùa 2024-2025 từ trang web https://www.footballtransfers.com. Nếu không có dữ liệu thì để là N/a

- Kết quả ghi vào một bảng thứ hai trong cơ sở dữ liệu SQLite.

Chú ý:
Trong khi crawl dữ liệu, có thể gặp các tình huống như Capchar và hoặc rate-limit. Đề xuất phương pháp khắc phục nếu có
Có thể dùng module request hoặc selenium để cào dữ liệu.

Bước 1: Lưu tên chương trình là transfer_data.py, import các thư viện cần thiết:

```
import pandas as pd
from io import StringIO
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
```

```
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import sqlite3
import time
```

Bước 2: Đọc từ player_data1.sqlite:

```
DB_NAME = 'player_data1.sqlite'
conn = sqlite3.connect(DB_NAME)
```

Bước 3: Chọn cột Player từ player_data1.sqlite và đếm tổng số cầu thủ trong player_data1.sqlite:

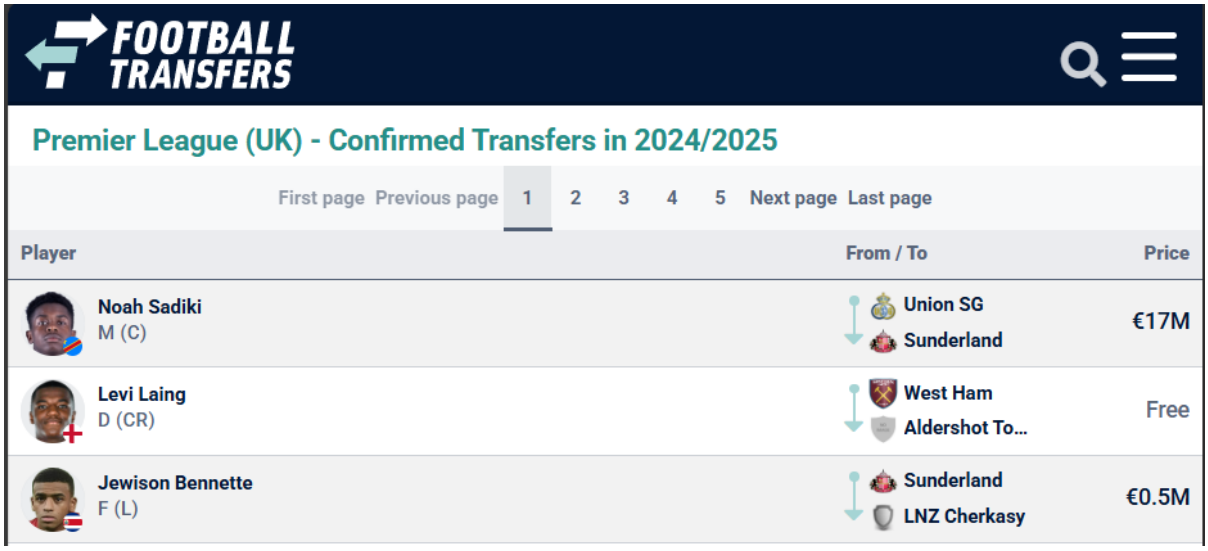
```
#Chọn cột Player từ player_data1.sqlite
player_list = pd.read_sql("SELECT Player FROM player_data1", conn)
conn.close()
players_i1 = player_list["Player"].drop_duplicates().tolist()
print(f"Đọc được {len(players_i1)} cầu thủ từ bảng player_data1.sqlite")
```

- Đoạn code chạy thành công, đọc được 491 cầu thủ từ bảng player_data1.sqlite.

Bước 4: Cấu hình Selenium để cào dữ liệu:

```
#Cấu hình bằng Selenium sử dụng Chrome:
options = webdriver.ChromeOptions()
#Tự động tải phiên bản Chrome phù hợp và đảm bảo rằng phiên bản cài đặt phù hợp, nếu lỗi thì in thông báo:
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=options)
```

Bước 5: Truy cập trang web tại địa chỉ để xem các cuộc chuyển nhượng trong mùa giải 2024-2025 cùng giá trị chuyển nhượng của các cầu thủ liên quan: https://www.footballtransfers.com/en/transfers/confirmed/most-recent/2024-2025/uk-premier-league




```
#Tạo 2 cột cho bảng SQLite:
players = []
prices = []
#Duyệt qua 17 trang cầu thủ có giá chuyển nhượng trong mùa giải 2024-2025 tại Ngoại hạng Anh
for page in range(1, 17):
    if page == 1:
            url  =  "https://www.footballtransfers.com/en/transfers/confirmed/most-recent/2024-2025/uk-premier-league"
    elif(page >= 2):
            url  =  f"https://www.footballtransfers.com/en/transfers/confirmed/most-recent/2024-2025/uk-premier-league/{page}"
    print(f"Đang cào trang {page}")
    driver.get(url)
    try:
        WebDriverWait(driver,10).until(EC.presence_of_element_located((By.CSS_SELECTOR, "tbody#player-table-body")))
    except:
        print(f"Không tìm thấy bảng dữ liệu ở trang {page}")
        continue
```

Bước 6: Sử dụng BeautifulSoup xử lý dữ liệu HTML:

```
#Sử dụng BeautifulSoup trích xuất HTML và xử lý dữ liệu:
    soup = BeautifulSoup(driver.page_source, "html.parser")
    rows = soup.select("tbody#player-table-body tr")
    for row in rows:
```

```
        name_tag = row.select_one("td.td-player span.d-none")
        player_name = name_tag.get_text(strip=True) if name_tag else "N/a"
        price_tag = row.select_one("td.td-price span")
        price = price_tag.get_text(strip=True) if price_tag else "N/a"
        if player_name != "N/a":
            players.append(player_name)
            prices.append(price)
driver.quit()
```

Bước 7: Gộp thêm bảng player_data1.sqlite:

```
# Gộp thêm bảng player_data1.sqlite3:
df_transfer = pd.DataFrame({
    "Player": players,
    "Price": prices
}).drop_duplicates(subset="Player")
print(f"Cào được tổng cộng là {len(df_transfer)} cầu thủ có dữ liệu giá chuyển nhượng")
final_df = pd.merge(player_list, df_transfer, on="Player", how="left")
# Điền các cầu thủ không có giá bằng 'N/a'
final_df["Price"] = final_df["Price"].fillna("N/a")
print(f"Tổng số cầu thủ: {len(final_df)}")
```

- Khi thực hiện chạy thành công thì có thông báo “Cào được tổng cộng ... cầu thủ có dữ liệu giá trị chuyển nhượng” trong “Tổng số cầu thủ: 491 cầu thủ”

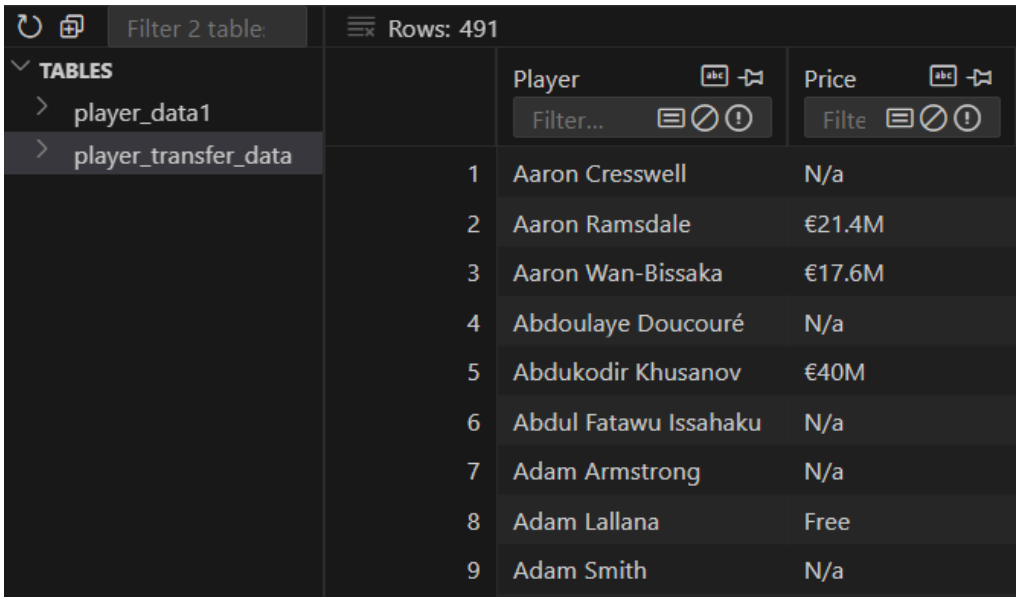
Bước 8: Ghi dữ liệu ra bảng SQLite thứ 2 đúng như yêu cầu bài tập:

```
#Ghi ra player_transfer_data.sqlite3
conn = sqlite3.connect(DB_NAME)
final_df.to_sql("player_transfer_data", conn, if_exists="replace", index=False)
conn.close()
print("Ghi giá chuyển nhượng vào bảng player_transfer_data.sqlite thành công!")
```

- Ghi thông báo ghi SQLite thành công.

Kết quả đạt được:

```
Đọc được 491 cầu thủ từ bảng player_data1.sqlite
Đang cào trang 1
Đang cào trang 2
Đang cào trang 3
Đang cào trang 4
Đang cào trang 5
Đang cào trang 6
Đang cào trang 7
Đang cào trang 8
Đang cào trang 9
Đang cào trang 10
Đang cào trang 11
Đang cào trang 12
Đang cào trang 13
Đang cào trang 14
Đang cào trang 15
Đang cào trang 16
Cào được tổng cộng là 380 cầu thủ có dữ liệu giá chuyển nhượng
Tổng số cầu thủ: 491
Ghi giá chuyển nhượng vào bảng player_transfer_data.sqlite thành công!
PS D:\Coding\PYTHON_CODEPTIT>
```



- Trong mùa giải 2024-2025 có 111 cầu thủ vẫn ở lại các câu lạc bộ chủ quản và số cầu thủ có các hoạt động trên thị trường chuyển nhượng.

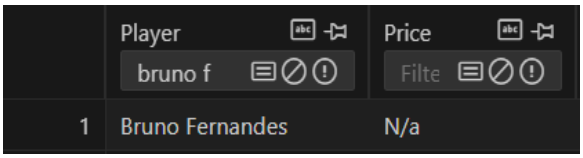
+ N/a là cầu thủ vẫn ở câu lạc bộ cũ, không chuyển đi đâu nên không liên quan đến giá chuyển nhượng

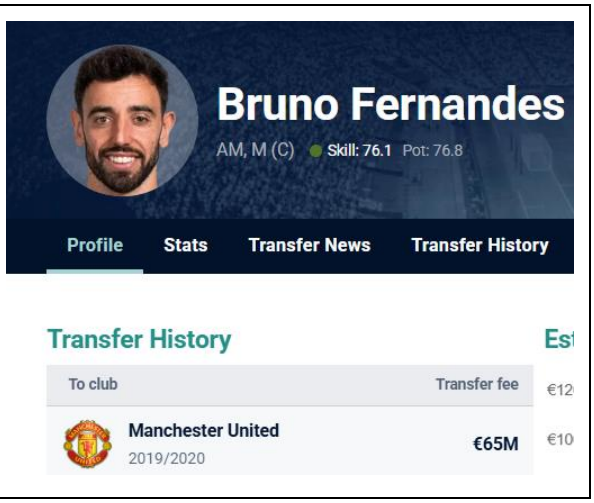
+ Free là chuyển nhượng tự do không mất phí cho câu lạc bộ, các câu lạc bộ dùng lương hay các ưu đãi khác để đàm phán hợp đồng với cầu thủ.

+ Các giá chuyển nhượng cầu thủ khác được tính theo đơn vị Bảng Anh (€)

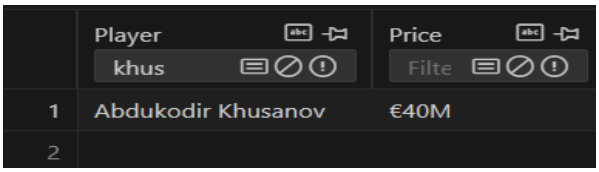
Kiểm nghiệm kết quả với các trường hợp thực tế:

TH1: Bruno Fernandes (Manchester United) chuyển đến câu lạc bộ này từ mùa giải 2019-2020 đến nay và chưa rời đi -> và với trang web https://www.footballtransfers.com/ mùa giải 2024-2025 của anh ta không có chuyển nhượng nên dữ liệu là N/a

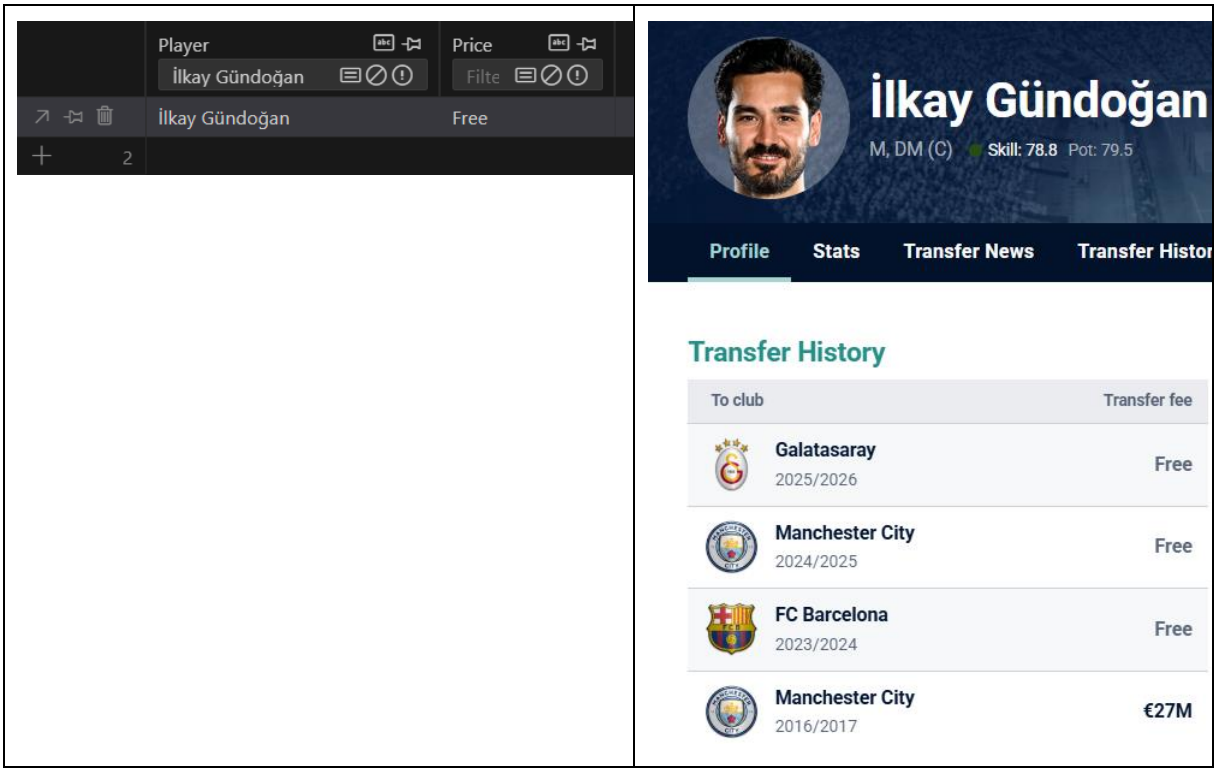




TH2: Abdukodir Khusanov (Manchester City) chuyển đến câu lạc bộ này vào mùa đông năm 2024-2025 từ RC Lens-> mức giá 40€



TH3: İlkay Gündoğan (Galatasaray) chuyển nhượng tự do từ Barcelona đến Manchester City vào mùa giải 2024-2025 -> giá trị chuyển nhượng là Free



Đề xuất các phương pháp đối phó với Captcha và Rate-Limit khi thực hiện cào dữ liệu:

1. Khắc phục Rate-Limit (Giới hạn Tốc độ Truy cập):

- Đề xuất: Thêm độ trễ ngẫu nhiên (time.sleep) giữa các lần truy cập URL hoặc thao tác.

- Mục đích: Giúp mô phỏng hành vi truy cập không đồng đều của người dùng thực, làm giảm nguy cơ bị hệ thống phát hiện và chặn vì tốc độ cào quá nhanh.

- Áp dụng trong code: Thay vì dùng time.sleep(2) cố định như trong báo cáo, nên sử dụng một giá trị ngẫu nhiên trong một khoảng an toàn (ví dụ: time.sleep(random.uniform(3, 8))).

2. Khắc phục CAPTCHA:

- Đề xuất: Tận dụng Selenium để giải quyết.

- Mục đích: Vì Selenium mô phỏng trình duyệt thực, nếu gặp CAPTCHA, chương trình sẽ hiển thị cửa sổ trình duyệt cho phép người dùng giải thủ công.

3. Xử lý Lỗi Tải Trang và Timeout:

- Đề xuất: Sử dụng WebDriverWait và xử lý ngoại lệ (try-except).

- Mục đích: Đảm bảo rằng chương trình chờ các phần tử quan trọng (như bảng dữ liệu) được tải xong hoàn toàn trước khi cố gắng trích xuất. Điều này ngăn ngừa lỗi khi mạng chậm hoặc trang web có nội dung được tải động.

- Áp dụng trong code: Dự án đã triển khai tính năng này bằng cách sử dụng WebDriverWait để chờ table_id được tìm thấy. Lỗi Timeout cũng được bắt bằng khối try-except.

II.

II.1.

Sử dụng module Python Flask để tạo các Restful API tra cứu chỉ số cầu thủ trong cơ sở dữ liệu vừa tạo ở câu 1 theo các tiêu chí sau:

- Tên cầu thủ: trả về toàn bộ chỉ số của cầu thủ có tên tương ứng.

- Câu lạc bộ: Trả về toàn bộ chỉ số của các cầu thủ thuộc về câu lạc bộ đó.

Bước 1: Lưu tên chương trình app.py, import các thư viện cần thiết:

```
from flask import Flask, request, jsonify
from flask_restful import Api, Resource
import sqlite3
import pandas as pd
```

- Flask: tạo ứng dụng web.
- request: dùng để lấy tham số từ URL (ví dụ ?name=ederson).
- jsonify: trả dữ liệu dạng JSON (ở đây không dùng trực tiếp, vì Flask-RESTful tự lo).
- flask_restful.Api, Resource: hỗ trợ định nghĩa các endpoint RESTful gọn hơn.
- sqlite3: làm việc với cơ sở dữ liệu SQLite.
- pandas: để đọc kết quả truy vấn SQL và xử lý/trả dữ liệu dễ dàng.

Bước 2: Khởi tạo Flask app và Restful API:

```
app = Flask(__name__)
api = Api(app)
```

- app: đối tượng Flask
- api: đối tượng Api dùng để đăng ký các "resource" (chính là endpoint như /player hay /club).

Bước 3: Cấu hình đường dẫn cơ sở dữ liệu:

```
DB_PATH = "player_data1.sqlite"
TABLE_NAME = "player_data1"
```

- Đây là file SQLite chứa dữ liệu cầu thủ.
- TABLE_NAME là tên bảng đã lưu dữ liệu vào.

Bước 4: Hàm truy vấn cơ sở dữ liệu:

```
def query_database(query, params=()):
```

```
    conn = sqlite3.connect(DB_PATH)
    df = pd.read_sql_query(query, conn, params=params)
    conn.close()
    return df
```

- Kết nối tới SQLite.
- Dùng pandas.read_sql_query() để thực thi câu lệnh SQL và trả về DataFrame.
- Đóng kết nối sau khi xong.

Bước 5: API tra cứu cầu thủ:

```
class PlayerLookup(Resource):
    def get(self):
        player_name = request.args.get("name")
```

* PlayerLookup là một lớp con của Resource (thuộc thư viện flask_restful)
- Mỗi lớp như thế này tương ứng với một endpoint API (ở đây là /player).
- Trong Flask-RESTful, mỗi HTTP method (GET, POST, PUT, DELETE) được định nghĩa bằng hàm trùng tên trong class.
-> Ở đây ta định nghĩa def get(self): nghĩa là xử lý yêu cầu GET (tức là khi người dùng truy cập /player?... trên trình duyệt)
* Lấy tham số trên URL:

```
player_name = request.args.get("name")
```

- request là đối tượng chứa toàn bộ thông tin của HTTP request (URL, headers, dữ liệu...).
- request.args là dictionary chứa các tham số truyền trên URL (query string).
-> Nếu người dùng không gửi tham số name, thì request.args.get("name") sẽ trả về None.
* Kiểm tra tham số bị thiếu:

```
if not player_name:
    return {"error": "Thiếu tham số ?name=<tên cầu thủ>"}, 400
```

- Nếu người dùng không nhập ?name=... → trả về lỗi 400 là HTTP status code cho *Bad Request* (yêu cầu không hợp lệ).
- Flask-RESTful cho phép ta trả về tuple dạng: (dictionary, status_code)
-> Flask tự động chuyển dictionary thành JSON.
* Thực hiện truy vấn trong try block:

```
try:
    df = query_database(
        f"SELECT * FROM {TABLE_NAME} WHERE Name LIKE ?",
(f"%{player_name}%",)
    )
```

```
except Exception as e:
    return {"error": f"Lỗi truy vấn cơ sở dữ liệu: {e}"}, 500
```

- df là DataFrame chứa kết quả truy vấn.
- Nếu có lỗi khi truy vấn → trả về lỗi 500 (Internal Server Error).

* Kiểm tra kết quả truy vấn:

```
if df.empty:
    return {"message": f"Không tìm thấy cầu thủ '{player_name}'"}, 404
return df.to_dict(orient="records"), 200
```

- df.empty trả về True nếu không có dòng nào trong DataFrame → không tìm thấy cầu thủ.
- 404 là mã lỗi "Not Found".
- df.to_dict(orient="records"): chuyển DataFrame thành list các dictionary, mỗi dòng là một cầu thủ.
- Trả về JSON cùng mã 200 (OK) nếu có kết quả.

Bước 6: API tra cứu câu lạc bộ:

```
class ClubLookup(Resource):
    def get(self):
        club_name = request.args.get("club")
        if not club_name:
            return {"error": "Thiếu tham số ?club=<tên câu lạc bộ>"}, 400
        try:
            df = query_database(
                f"SELECT * FROM {TABLE_NAME} WHERE Squad LIKE ?",
(f"%{club_name}%",)
            )
        except Exception as e:
            return {"error": f"Lỗi truy vấn cơ sở dữ liệu: {e}"}, 500
        if df.empty:
            return {"message": f"Không tìm thấy câu lạc bộ '{club_name}'"}, 404
        return df.to_dict(orient="records"), 200
```

- class ClubLookup(Resource): Đây là API tra cứu cầu thủ theo tên câu lạc bộ (endpoint /club)
- def get(self): Hàm xử lý khi người dùng gửi yêu cầu GET

* Các bước chính:

- Lấy tham số trên URL:
club_name = request.args.get("club") -> Lấy tên CLB từ ?club=....
- Kiểm tra thiếu tham số:
Nếu không có club → trả lỗi 400 Bad Request.
- Truy vấn cơ sở dữ liệu:
VD: SELECT * FROM player_data1 WHERE Squad LIKE '%Manchester%'
-> Tìm tất cả cầu thủ có Squad chứa tên CLB nhập vào.
- Bắt lỗi truy vấn:
Nếu có lỗi khi truy vấn SQLite -> trả lỗi 500 Internal Server Error.
- Không tìm thấy dữ liệu:
Nếu không có cầu thủ nào thuộc CLB đó -> trả 404 Not Found.
- Trả kết quả JSON:
Nếu có kết quả -> chuyển DataFrame thành list dict rồi trả JSON + mã 200 OK.

Bước 7: Đăng ký các API endpoint:

```
api.add_resource(PlayerLookup, "/player")
api.add_resource(ClubLookup, "/club")
```

/player → xử lý bởi lớp PlayerLookup

/club → xử lý bởi lớp ClubLookup

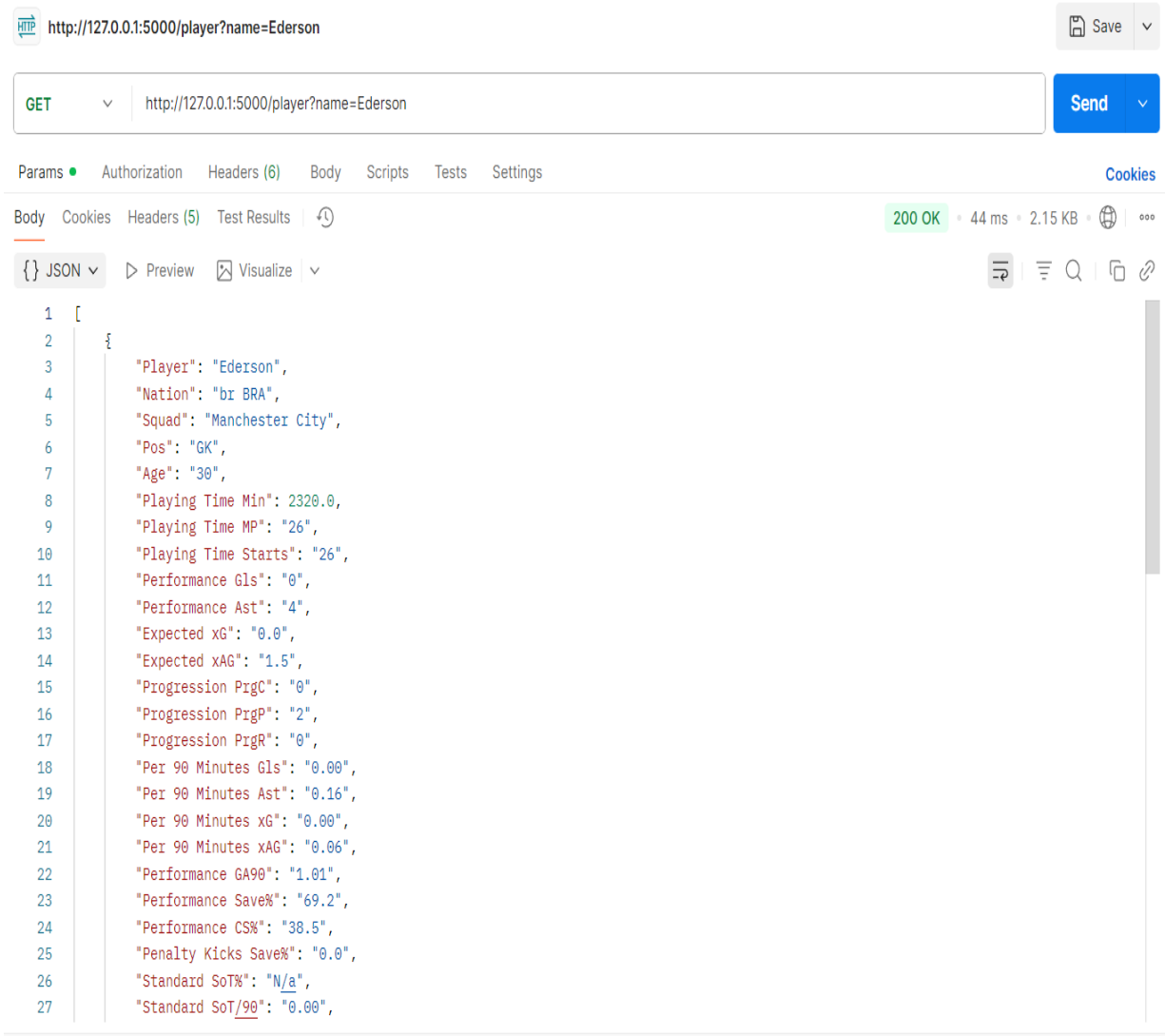
Bước 8: Chạy ứng dụng Flask:

```
if __name__ == "__main__":
    print("Server Flask đang chạy tại http://127.0.0.1:5000")
    print("Tra cứu cầu thủ: http://127.0.0.1:5000/player?name=<Tên_cầu_thủ>")
    print("Tra cứu CLB: http://127.0.0.1:5000/club?club=<Tên_câu_lạc_bộ>")
    app.run(debug=True)
```

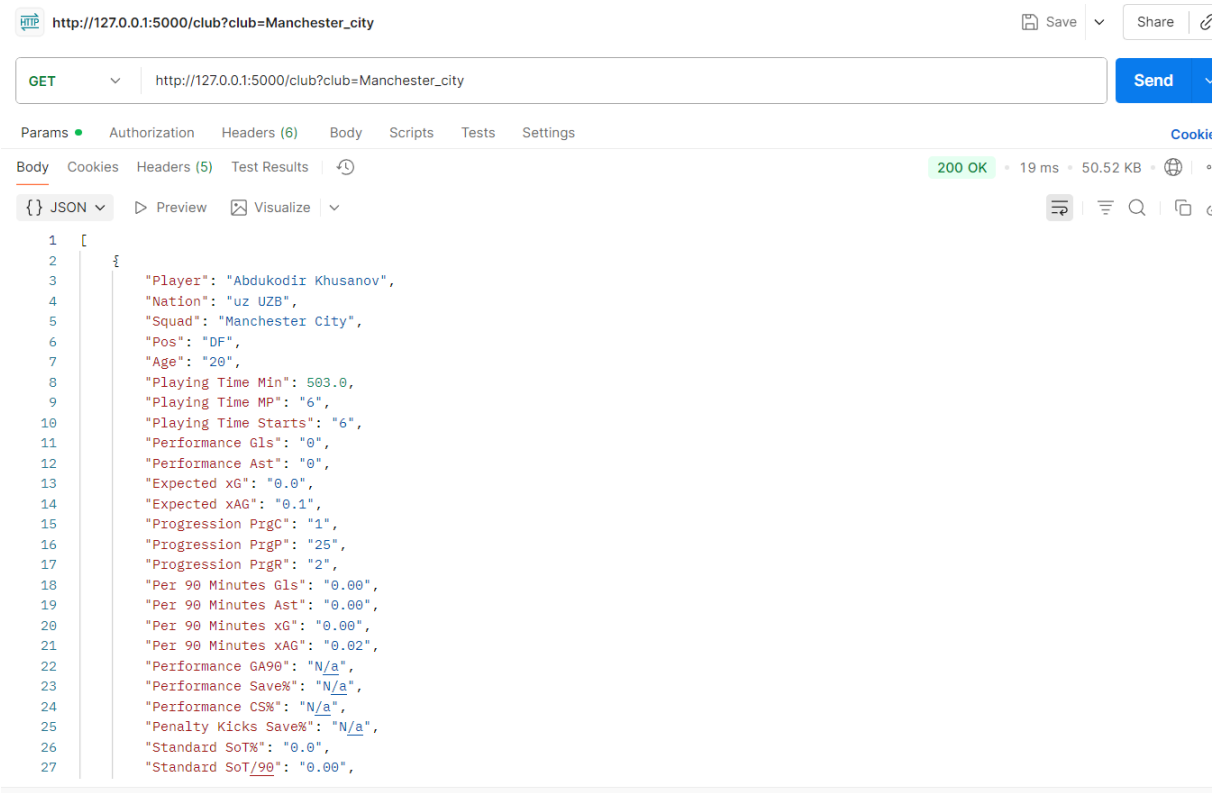
Kết quả đạt được:

```
PS C:\Users\ADMIN\Documents\nhap> python -u "c:\Users\ADMIN\Documents\nhap\app.py"
 Server Flask đang chạy tại http://127.0.0.1:5000
 Tra cứu cầu thủ: http://127.0.0.1:5000/player?name=<Tên_cầu_thủ>
 Tra cứu CLB: http://127.0.0.1:5000/club?club=<Tên_câu_lạc_bộ>
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 Server Flask đang chạy tại http://127.0.0.1:5000
 Tra cứu cầu thủ: http://127.0.0.1:5000/player?name=<Tên_cầu_thủ>
 Tra cứu CLB: http://127.0.0.1:5000/club?club=<Tên_câu_lạc_bộ>
 * Debugger is active!
 * Debugger PIN: 117-711-306
```

Tra cứu cầu thủ:



Tra cứu câu lạc bộ:



II.2.

Sử dụng module Request, viết 1 chương trình python để tra cứu dữ liệu cầu thủ theo 2 tiêu chí như trên. Tên cầu thủ hoặc câu lạc bộ được truyền vào lúc chạy chương trình có cú pháp như sau:
./> python lookup.py --name <tên cầu thủ> --club <tên câu lạc bộ>
- Kết quả trả ra được in ra màn hình dưới dạng pretty-printed json, và 1 file CSV. Tên file được đặt theo tên cầu thủ hoặc câu lạc bộ, tùy theo sử dụng tham số đầu vào.

Bước 1: Lưu tên chương trình lookup.py, import các thư viện cần thiết:

```
import argparse
import requests
import json
import pandas as pd
```

- argparse: dùng để nhận tham số từ dòng lệnh (command line).
- requests: thư viện gửi HTTP request (GET, POST, ...).
- json: để in kết quả JSON đẹp và dễ đọc.

- pandas: dùng để lưu kết quả thành file CSV

Bước 2: Địa chỉ API server:

```
BASE_URL = "http://127.0.0.1:5000"
```

- Đây là địa chỉ server Flask đang chạy.

Bước 3: Tạo parser để đọc tham số dòng lệnh:

```
parser = argparse.ArgumentParser(description="Tra cứu dữ liệu cầu thủ hoặc câu lạc bộ (Premier League)")
parser.add_argument("--name", help="Tên cầu thủ cần tra cứu")
parser.add_argument("--club", help="Tên câu lạc bộ cần tra cứu")
args = parser.parse_args()
```

- argparse.ArgumentParser(...): tạo trình phân tích lệnh.
- add_argument("--name"): cho phép nhập tùy chọn --name <tên>.
- add_argument("--club"): cho phép nhập --club <tên_clb>.
- parse_args() lấy giá trị người dùng nhập và lưu trong args.

Bước 4: Kiểm tra nếu không nhập gì:

```
if not args.name and not args.club:
    print(" Bạn cần nhập ít nhất một tham số: --name <tên> hoặc --club <tên>")
    exit()
```

- Nếu người dùng không nhập cả hai thì báo lỗi và dừng chương trình.

Bước 5: Xác định endpoint và tên file xuất:

```
if args.name:
    endpoint = "/player"
    params = {"name": args.name}
    filename = args.name.replace(" ", "_") + ".csv"
elif args.club:
    endpoint = "/club"
    params = {"club": args.club}
    filename = args.club.replace(" ", "_") + ".csv"
```

Bước 6: Gửi request tới API Flask:

```
url = BASE_URL + endpoint
print(f" Đang gửi request tới: {url} với tham số {params}\n")
```

Bước 7: Kiểm tra mã phản hồi HTTP, đọc dữ liệu JSON trả về, in kết quả đẹp ra màn hình:

```
try:
    response = requests.get(url, params=params)
    if response.status_code != 200:
```

```
        print(f"Lỗi truy vấn ({response.status_code})")
        print(response.text)
        exit()
    data = response.json()
    # 📄 In ra dạng pretty-printed JSON
    print("Kết quả dạng JSON:")
    print(json.dumps(data, indent=4, ensure_ascii=False))
```

Bước 8: Ghi dữ liệu ra file CSV:

```
    if isinstance(data, list) and len(data) > 0:
        df = pd.DataFrame(data)
        df.to_csv(filename, index=False)
        print(f"\n Đã lưu kết quả vào file: {filename}")
    else:
        print(" Không có dữ liệu để lưu vào CSV.")
```

- Kiểm tra xem data có phải là danh sách không và có dữ liệu không?

+ Nếu có dữ liệu -> chuyển list thành DataFrame, ghi ra file CSV.

+ Nếu không có dữ liệu -> báo “Không có dữ liệu để lưu”.

Bước 9: Bắt lỗi khi gửi request:

```
except requests.exceptions.RequestException as e:
    print(f" Lỗi khi gửi request: {e}")
```

- Nếu server không chạy, mất kết nối, hoặc URL sai → requests ném lỗi.
- Phần except sẽ in thông báo lỗi ra màn hình.

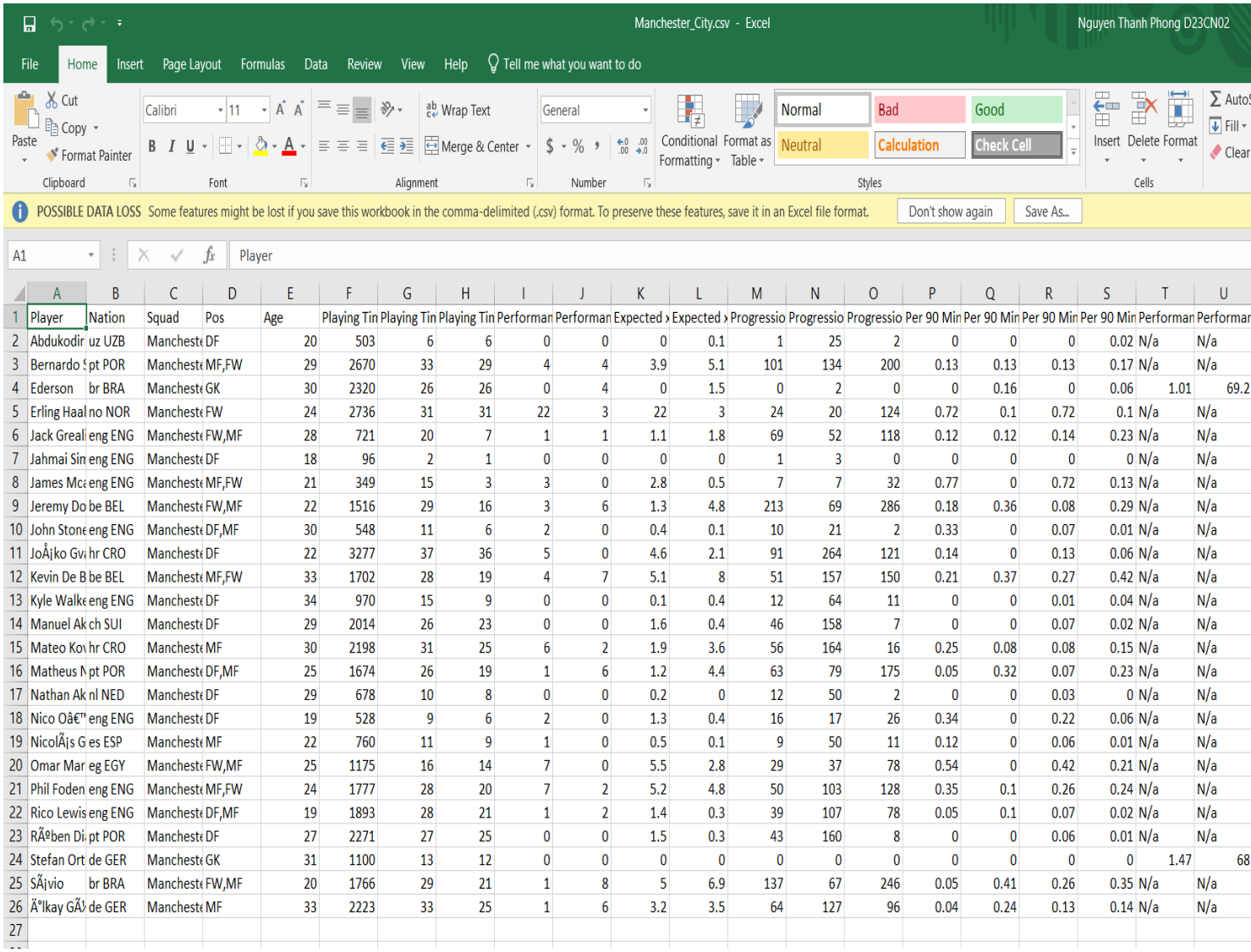
Kết quả đạt được:

python lookup.py --name "Ederson"

```
PS C:\Users\ADMIN\Documents\nhap> python lookup.py --name "Ederson"
 Đang gửi request tới: http://127.0.0.1:5000/player với tham số {'name': 'Ederson'}

📘 Kết quả dạng JSON:
[
    {
        "Player": "Ederson",
        "Nation": "br BRA",
        "Squad": "Manchester City",
        "Pos": "GK",
        "Age": "30",
        "Playing Time Min": 2320.0,
        "Playing Time MP": "26",
        "Playing Time Starts": "26",
        "Performance Gls": "0",
        "Performance Ast": "4",
        "Expected xG": "0.0",
        "Expected xAG": "1.5",
        "Progression PrgC": "0",
        "Progression PrgP": "2",
        "Progression PrgR": "0",
        "Per 90 Minutes Gls": "0.00",
        "Per 90 Minutes Ast": "0.16",
        "Per 90 Minutes xG": "0.00",
        "Per 90 Minutes xAG": "0.06",
        "Performance GA90": "1.01",
        "Performance Save%": "69.2",
        "Performance CS%": "38.5",
        "Penalty Kicks Save%": "0.0",
        "Standard SoT%": "N/a",
        "Standard SoT/90": "0.00",
        "Standard G/Sh": "N/a",
        "Standard Dist": "N/a",
        "Total Cmp": "811",
        "Total Cmp%": "87.9",
        "Total TotDist": "19885",
        "Short Cmp": "142",
        "Medium Cmp": "471",
        "Long Cmp": "194",
        "KP": "4",
        "Passing 1/3": "21",
```

python lookup.py --club "Manchester City"



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Player	Nation	Squad	Pos	Age	Playing Tin	Playing Tin	Playing Tin	Performan	Performan	Expected ›	Expected ›	Progressio	Progressio	Progressio	Per 90 Min	Per 90 Min	Per 90 Min	Per 90 Min	Performan	Performan
2	Abdukodir	uz UZB	Manchest	DF	20	503	6	6	0	0	0	0.1	1	25	2	0	0	0	0.02	N/a	N/a
3	Bernardo S	pt POR	Manchest	MF,FW	29	2670	33	29	4	4	3.9	5.1	101	134	200	0.13	0.13	0.13	0.17	N/a	N/a
4	Ederson	br BRA	Manchest	GK	30	2320	26	26	0	4	0	1.5	0	2	0	0	0.16	0	0.06	1.01	69.2
5	Erling Haal	no NOR	Manchest	FW	24	2736	31	31	22	3	22	3	24	20	124	0.72	0.1	0.72	0.1	N/a	N/a
6	Jack Greal	eng ENG	Manchest	FW,MF	28	721	20	7	1	1	1.1	1.8	69	52	118	0.12	0.12	0.14	0.23	N/a	N/a
7	Jahmai Sin	eng ENG	Manchest	DF	18	96	2	1	0	0	0	0	1	3	0	0	0	0	0	N/a	N/a
8	James Mc	eng ENG	Manchest	MF,FW	21	349	15	3	3	0	2.8	0.5	7	7	32	0.77	0	0.72	0.13	N/a	N/a
9	Jeremy Do	be BEL	Manchest	FW,MF	22	1516	29	16	3	6	1.3	4.8	213	69	286	0.18	0.36	0.08	0.29	N/a	N/a
10	John Ston	eng ENG	Manchest	DF,MF	30	548	11	6	2	0	0.4	0.1	10	21	2	0.33	0	0.07	0.01	N/a	N/a
11	JoÅ¡ko Gv	hr CRO	Manchest	DF	22	3277	37	36	5	0	4.6	2.1	91	264	121	0.14	0	0.13	0.06	N/a	N/a
12	Kevin De B	be BEL	Manchest	MF,FW	33	1702	28	19	4	7	5.1	8	51	157	150	0.21	0.37	0.27	0.42	N/a	N/a
13	Kyle Walke	eng ENG	Manchest	DF	34	970	15	9	0	0	0.1	0.4	12	64	11	0	0	0.01	0.04	N/a	N/a
14	Manuel Ak	ch SUI	Manchest	DF	29	2014	26	23	0	0	1.6	0.4	46	158	7	0	0	0.07	0.02	N/a	N/a
15	Mateo Ko	hr CRO	Manchest	MF	30	2198	31	25	6	2	1.9	3.6	56	164	16	0.25	0.08	0.08	0.15	N/a	N/a
16	Matheus N	pt POR	Manchest	DF,MF	25	1674	26	19	1	6	1.2	4.4	63	79	175	0.05	0.32	0.07	0.23	N/a	N/a
17	Nathan Ak	nl NED	Manchest	DF	29	678	10	8	0	0	0.2	0	12	50	2	0	0	0.03	0	N/a	N/a
18	Nico Oâ€™	eng ENG	Manchest	DF	19	528	9	6	2	0	1.3	0.4	16	17	26	0.34	0	0.22	0.06	N/a	N/a
19	NicolÃ¡s G	es ESP	Manchest	MF	22	760	11	9	1	0	0.5	0.1	9	50	11	0.12	0	0.06	0.01	N/a	N/a
20	Omar Mar	eg EGY	Manchest	FW,MF	25	1175	16	14	7	0	5.5	2.8	29	37	78	0.54	0	0.42	0.21	N/a	N/a
21	Phil Foden	eng ENG	Manchest	MF,FW	24	1777	28	20	7	2	5.2	4.8	50	103	128	0.35	0.1	0.26	0.24	N/a	N/a
22	Rico Lewis	eng ENG	Manchest	DF,MF	19	1893	28	21	1	2	1.4	0.3	39	107	78	0.05	0.1	0.07	0.02	N/a	N/a
23	RÃºben Di	pt POR	Manchest	DF	27	2271	27	25	0	0	1.5	0.3	43	160	8	0	0	0.06	0.01	N/a	N/a
24	Stefan Ort	de GER	Manchest	GK	31	1100	13	12	0	0	0	0	0	0	0	0	0	0	0	1.47	68
25	SÃ¡vio	br BRA	Manchest	FW,MF	20	1766	29	21	1	8	5	6.9	137	67	246	0.05	0.41	0.26	0.35	N/a	N/a
26	Ã°lkay GÃ¼	de GER	Manchest	MF	33	2223	33	25	1	6	3.2	3.5	64	127	96	0.04	0.24	0.13	0.14	N/a	N/a
27																					

III.

III.1.

- Tìm trung vị, trung bình và độ lệch chuẩn của mỗi chỉ số các cầu thủ của mỗi đội. Ghi kết quả vào 1 file CSV.

- 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 2024-2025

A. Tìm trung vị, trung bình và độ lệch chuẩn của mỗi chỉ số các cầu thủ của mỗi đội. Ghi kết quả vào 1 file CSV:

Bước 1: Lưu tên chương trình TrungVi.py, import các thư viện cần thiết:

```
import pandas as pd
import sqlite3

# Kết nối đến cơ sở dữ liệu SQLite
DB_NAME = 'player_data1.sqlite'

# Kết nối database và đọc dữ liệu
conn = sqlite3.connect(DB_NAME)

# Đọc dữ liệu từ bảng player_data1 vào DataFrame
data = pd.read_sql_query("SELECT * FROM player_data1", conn)

# Đóng kết nối database
conn.close()
```

- Để thực hiện yêu cầu tìm trung vị, trung bình, độ lệch chuẩn của mỗi chỉ số, ta thực hiện phân tích thống kê chi tiết các chỉ số của cầu thủ từ dữ liệu đã thu thập ở câu I.1 qua file kết quả là 'player_data1.sqlite':

+ Ý tưởng: sử dụng thư viện pandas để thao tác dữ liệu dạng bảng.

+ Mục đích: nhập thư viện cần thiết (pandas) và đọc dữ liệu từ file 'player_data1.sqlite (file kết quả của câu 1) để sử dụng cho phân tích.

Bước 2: Xử lý dữ liệu trước khi tiến hành tính toán:

- Lấy các cột có giá trị là số: dòng code dưới đây chuyển đổi tất cả các cột có thể sang kiểu số và lưu tên các cột này vào biến ‘number_attributes’:

```
# Chuyển đổi tất cả các cột có thể sang kiểu số
numeric_df = data.copy()
for col in data.columns:
    # Bỏ qua các cột không cần chuyển đổi
    if col not in ['Player', 'Nation', 'Squad', 'Pos']:
        # Thử chuyển đổi cột sang kiểu số
        try:
            # Xử lý chuỗi có dấu phẩy và các ký tự đặc biệt
            numeric_df[col] = numeric_df[col].replace('N/a', pd.NA)
                                                    numeric_df[col]          =
pd.to_numeric(numeric_df[col].astype(str).str.replace(',', ''), errors='coerce')
        except:
            continue

# Chọn các cột số sau khi đã chuyển đổi
number_attributes = numeric_df.select_dtypes(include=[float, int])
```

- Khởi tạo một từ điển để lưu kết quả: ‘results’ được khởi tạo với một cột tên là ‘Team’ bắt đầu với giá trị là ‘all’ để ghi nhận các kết quả cho toàn bộ các chỉ số cho toàn giải đấu trước khi tính toán riêng cho từng đội.

```
# Khởi tạo từ điển kết quả với cột "Team"
results = {"Team": ["all"]}
```

Bước 3: Tiến hành tính toán:

- Tính toán trung vị, trung bình và độ lệch chuẩn cho từng chỉ số cho các cầu thủ trong toàn giải:

```
# Khởi tạo các list cho các chỉ số thống kê
for col in number_attributes.columns:
    # Tính toán cho toàn bộ giải đấu trung bình, trung vị, độ lệch chuẩn
                            results[f"Median          of          {col}"]          =
[f"{number_attributes[col].median(skipna=True):.2f}"]
                             results[f"Mean          of          {col}"]          =
[f"{number_attributes[col].mean(skipna=True):.2f}"]
    results[f"Std of {col}"] = [f"{number_attributes[col].std(skipna=True):.2f}"]
```

- Vòng lặp for được sử dụng để lần lượt xử lý từng chỉ số nằm trong tập number_attributes. Ở mỗi lượt, chương trình sẽ tính ba đại lượng thống kê quan trọng gồm trung vị, giá trị trung bình và độ lệch chuẩn.

+ 'number_attributes[x].median(skipna=True)': trả về giá trị trung vị của cột *x*, đồng thời bỏ qua các ô bị thiếu dữ liệu.

+ 'number_attributes[x].mean(skipna=True)': tính giá trị trung bình của cột *x* mà không xét đến các giá trị NaN.

+ 'number_attributes[x].std(skipna=True)': xác định độ lệch chuẩn của cột *x*, cũng với cơ chế bỏ qua dữ liệu thiếu.

-> Các kết quả thống kê thu được sẽ được định dạng về hai chữ số thập phân trước khi lưu vào cấu trúc results để phục vụ cho việc tổng hợp sau này.

- Tính toán trung vị, trung bình và độ lệch chuẩn cho từng chỉ số cho các cầu thủ trong từng đội bóng:

```
# Nhóm dữ liệu theo từng đội bóng và tính các chỉ số thống kê
for squad, group in data.groupby("Squad"):
    results["Team"].append(squad)
    for col in number_attributes.columns:
        # Lọc dữ liệu cho đội hiện tại
        team_data = number_attributes[col][data['Squad'] == squad]
        # Thêm các chỉ số thống kê cho đội
                                                    results[f"Median                of
{col}"].append(f"{team_data.median(skipna=True):.2f}")
        results[f"Mean of {col}"].append(f"{team_data.mean(skipna=True):.2f}")
        results[f"Std of {col}"].append(f"{team_data.std(skipna=True):.2f}")
```

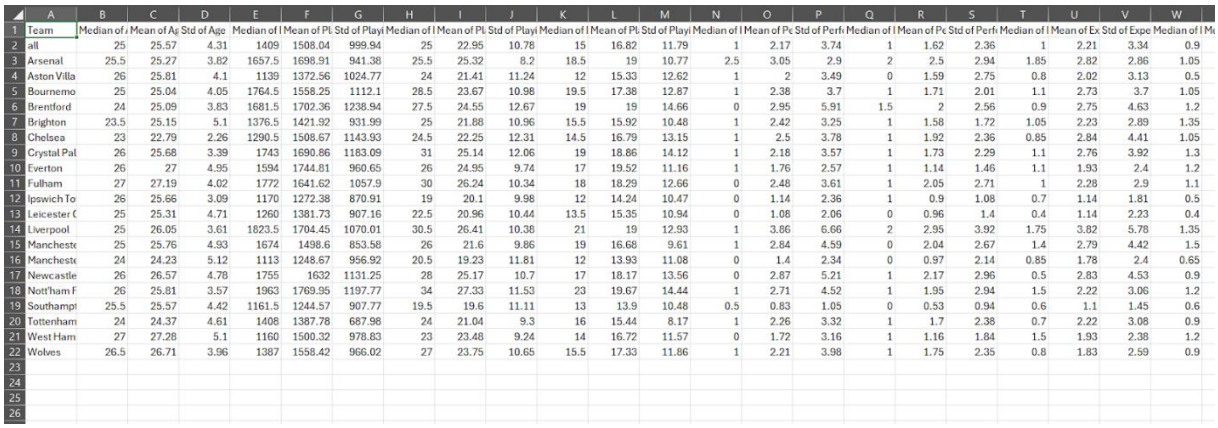
+ Trong bước này, dữ liệu được gom nhóm theo cột Team bằng hàm *groupby* để tính trung vị, trung bình và độ lệch chuẩn cho từng chỉ số trong number_attributes của mỗi đội bóng, sau đó toàn bộ kết quả được đưa vào cấu trúc results.

Bước 4: Ghi kết quả ra file CSV:

```
# Chuyển kết quả thành DataFrame
results_df = pd.DataFrame(results)

# Lưu kết quả
results_df.to_csv("results2.csv", index=False)
print("Đã tính toán và lưu kết quả thống kê thành công!")
```

Kết quả đạt được:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Team	Median of /	Mean of Ag	Std of Age	Median of I	Mean of Pl	Std of Playi	Median of I	Mean of Pl	Std of Playi	Median of I	Mean of Pl	Std of Playi	Median of I	Mean of Pc	Std of Perfi	Median of I	Mean of Pc	Std of Perfi	Median of I	Mean of Ex	Std of Expe	Median of I
2	all	25	25.57	4.31	1409	1508.04	999.94	25	22.95	10.78	15	16.82	11.79	1	2.17	3.74	1	1.62	2.36	1	2.21	3.34	0.9
3	Arsenal	25.5	25.27	3.82	1657.5	1698.91	941.38	25.5	25.32	8.2	18.5	19	10.77	2.5	3.05	2.9	2	2.5	2.94	1.85	2.82	2.86	1.05
4	Aston Villa	26	25.81	4.1	1139	1372.56	1024.77	24	21.41	11.24	12	15.33	12.62	1	2	3.49	0	1.59	2.75	0.8	2.02	3.13	0.5
5	Bournemo	25	25.04	4.05	1764.5	1558.25	1112.1	28.5	23.67	10.98	19.5	17.38	12.87	1	2.38	3.7	1	1.71	2.01	1.1	2.73	3.7	1.05
6	Brentford	24	25.09	3.83	1681.5	1702.36	1238.94	27.5	24.55	12.67	19	19	14.66	0	2.95	5.91	1.5	2	2.56	0.9	2.75	4.63	1.2
7	Brighton	23.5	25.15	5.1	1376.5	1421.92	931.99	25	21.88	10.96	15.5	15.92	10.48	1	2.42	3.25	1	1.58	1.72	1.05	2.23	2.89	1.35
8	Chelsea	23	22.79	2.26	1290.5	1508.67	1143.93	24.5	22.25	12.31	14.5	16.79	13.15	1	2.5	3.78	1	1.92	2.36	0.85	2.84	4.41	1.05
9	Crystal Pal	26	25.68	3.39	1743	1690.86	1183.09	31	25.14	12.06	19	18.86	14.12	1	2.18	3.57	1	1.73	2.29	1.1	2.76	3.92	1.3
10	Everton	26	27	4.95	1594	1744.81	960.65	26	24.95	9.74	17	19.52	11.16	1	1.76	2.57	1	1.14	1.46	1.1	1.93	2.4	1.2
11	Fulham	27	27.19	4.02	1772	1641.62	1057.9	30	26.24	10.34	18	18.29	12.66	0	2.48	3.61	1	2.05	2.71	1	2.28	2.9	1.1
12	Ipswich To	26	25.66	3.09	1170	1272.38	870.91	19	20.1	9.98	12	14.24	10.47	0	1.14	2.36	1	0.9	1.08	0.7	1.14	1.81	0.5
13	Leicester (	25	25.31	4.71	1260	1381.73	907.16	22.5	20.96	10.44	13.5	15.35	10.94	0	1.08	2.06	0	0.96	1.4	0.4	1.14	2.23	0.4
14	Liverpool	25	26.05	3.61	1823.5	1704.45	1070.01	30.5	26.41	10.38	21	19	12.93	1	3.86	6.66	2	2.95	3.92	1.75	3.82	5.78	1.35
15	Manchest	25	25.76	4.93	1674	1498.6	853.58	26	21.6	9.86	19	16.68	9.61	1	2.84	4.59	0	2.04	2.67	1.4	2.79	4.42	1.5
16	Manchest	24	24.23	5.12	1113	1248.67	956.92	20.5	19.23	11.81	12	13.93	11.08	0	1.4	2.34	0	0.97	2.14	0.85	1.78	2.4	0.65
17	Newcastle	26	26.57	4.78	1755	1632	1131.25	28	25.17	10.7	17	18.17	13.56	0	2.87	5.21	1	2.17	2.96	0.5	2.83	4.53	0.9
18	Nott'ham F	26	25.81	3.57	1963	1769.95	1197.77	34	27.33	11.53	23	19.67	14.44	1	2.71	4.52	1	1.95	2.94	1.5	2.22	3.06	1.2
19	Southampt	25.5	25.57	4.42	1161.5	1244.57	907.77	19.5	19.6	11.11	13	13.9	10.48	0.5	0.83	1.05	0	0.53	0.94	0.6	1.1	1.45	0.6
20	Tottenham	24	24.37	4.61	1408	1387.78	687.98	24	21.04	9.3	16	15.44	8.17	1	2.26	3.32	1	1.7	2.38	0.7	2.22	3.08	0.9
21	West Ham	27	27.28	5.1	1160	1500.32	978.83	23	23.48	9.24	14	16.72	11.57	0	1.72	3.16	1	1.16	1.84	1.5	1.93	2.38	1.2
22	Wolves	26.5	26.71	3.96	1387	1558.42	966.02	27	23.75	10.65	15.5	17.33	11.86	1	2.21	3.98	1	1.75	2.35	0.8	1.83	2.59	0.9
23																							
24																							
25																							
26																							

B. 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 2024-2025?

* Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số:

Bước 1: Lưu tên chương trình là BestTeam.py, import các thư viện cần thiết

```
import pandas as pd
import sqlite3
# Kết nối đến cơ sở dữ liệu SQLite
DB_NAME = 'player_data1.sqlite'
# Kết nối database và đọc dữ liệu
conn = sqlite3.connect(DB_NAME)
# Đọc dữ liệu từ bảng player_data1 vào DataFrame
df = pd.read_sql_query("SELECT * FROM player_data1", conn)
# Đóng kết nối database
conn.close()
```

- Sử dụng thư viện ‘pandas’ để xử lý và phân tích dữ liệu.

- Đọc dữ liệu từ file sqlite có tên là ‘player_data1.sqlite’ (file kết quả của câu 1) vào một DataFrame có tên là df.

Bước 2: Xử lý dữ liệu:

```
# Chuyển đổi tất cả các cột có thể sang kiểu số
for col in df.columns:
    # Bỏ qua các cột không cần chuyển đổi
    if col not in ['Player', 'Nation', 'Squad', 'Pos']:
        # Thử chuyển đổi cột sang kiểu số
```

```
        try:
            df[col] = df[col].replace('N/a', pd.NA)
                        df[col]   =   pd.to_numeric(df[col].astype(str).str.replace(',',   ''),
errors='coerce')
        except:
            continue
# Nhóm dữ liệu theo cột 'Squad'
teams = df.groupby('Squad')
# Khởi tạo DataFrame rỗng để lưu kết quả
best_teams = pd.DataFrame(columns=['Attribute', 'Best Team', 'Value'])
# Lọc ra các cột chứa dữ liệu dạng số
number_columns = df.select_dtypes(include=[float, int]).columns
```

- Nhóm dữ liệu theo cột ‘Squad’ để có thể tính toán các chỉ số cho từng đội bóng.

- Khởi tạo một DataFrame rỗng để lưu kết quả và gồm có các cột là ‘Attribute’, ‘Best Team’, ‘Value’.

- Đồng thời, ta lọc các cột chứa dữ liệu dạng số và bỏ qua các giá trị ‘N/a’.

Bước 3: Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số:

```
# Tìm đội có điểm số cao nhất cho từng chỉ số
for column in number_columns:
    # Tính giá trị trung bình của mỗi đội cho cột đang xét
    team_means = teams[column].mean()
    # Loại bỏ các giá trị NaN
    team_means = team_means.dropna()
    if not team_means.empty:
        best_team = team_means.idxmax()
        best_value = team_means.max()

        # Tạo một hàng dữ liệu mới
            row  =  pd.DataFrame({'Attribute':  [column],  'Best  Team':  [best_team],
'Value': [f"{best_value:.2f}"]})
        # Thêm hàng này vào best_teams
        best_teams = pd.concat([best_teams, row], ignore_index=True)
```

- Ở bước này, ta tiến hành tính giá trị trung bình của từng chỉ số cho mỗi đội bóng, sau đó xác định đội có mức trung bình cao nhất - đây chính là đội đứng đầu ở chỉ số đó.

- Vòng lặp for sẽ lần lượt duyệt qua từng chỉ số, tính toán trung bình theo đội, rồi chọn ra đội có giá trị lớn nhất; tên đội được gán vào ‘best_team’, còn giá trị tương ứng được lưu vào ‘best_value’.

- Tiếp theo, một dòng dữ liệu mới (row) được tạo ra để chứa đầy đủ thông tin gồm tên chỉ số, đội dẫn đầu và giá trị tốt nhất; dòng này sau đó được ghép vào DataFrame ‘best_teams’ để tổng hợp kết quả.

Bước 4: Ghi kết quả ra file CSV:

```
 # In kết quả ra màn hình
print("Kết quả thống kê:")
print(best_teams)
# Lưu kết quả vào file CSV
best_teams.to_csv('results_BestTeam.csv', index=False)
print("\nĐã lưu kết quả vào file 'results_BestTeam.csv'")
```

Kết quả đạt được:

	A	B	C
1	Attribute	Best Team	Value
2	Age	West Ham	27.28
3	Playing Time Min	Nott'ham Forest	1769.95
4	Playing Time MP	Nott'ham Forest	27.33
5	Playing Time Starts	Nott'ham Forest	19.67
6	Performance Gls	Liverpool	3.86
7	Performance Ast	Liverpool	2.95
8	Expected xG	Liverpool	3.82
9	Expected xAG	Liverpool	2.83
10	Progression PrgC	Manchester City	45.76
11	Progression PrgP	Liverpool	87.45
12	Progression PrgR	Liverpool	86.82
13	Per 90 Minutes Gls	Manchester City	0.18
14	Per 90 Minutes Ast	Liverpool	0.14
15	Per 90 Minutes xG	Bournemouth	0.2
16	Per 90 Minutes xAG	Chelsea	0.15
17	Performance GA90	Leicester City	2.6
18	Performance Save%	Bournemouth	79.5
19	Performance CS%	Aston Villa	60.8
20	Penalty Kicks Save%	Everton	100
21	Standard SoT%	Nott'ham Forest	38.23
22	Standard SoT/90	Bournemouth	0.58
23	Standard G/Sh	Arsenal	0.15
24	Standard Dist	Nott'ham Forest	18.56
25	Total Cmp	Manchester City	857.88
26	Total Cmp%	Manchester City	86.48

* Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2024-2025:

- Để đánh giá phong độ của một đội bóng, cả khả năng tấn công lẫn sự vững chắc trong phòng ngự đều đóng vai trò quan trọng. Vì vậy, khi xác định đội có phong độ tốt nhất mùa giải Ngoại Hạng Anh 2024–2025, ta cần xem xét các chỉ số phản ánh hiệu quả ở cả hai mặt trận này.

- Điểm phong độ tổng hợp của mỗi đội được tạo nên bằng cách tính trung bình các chỉ số liên quan đến tấn công và phòng ngự, sau đó kết hợp hai nhóm điểm này lại để đưa ra một đánh giá tổng quát.

Bước 1: Lưu tên chương trình là PhongDo.py, import các thư viện cần thiết:

```
import pandas as pd
```

```
import sqlite3
# Kết nối đến cơ sở dữ liệu SQLite
DB_NAME = 'player_data1.sqlite'

# Kết nối database và đọc dữ liệu
conn = sqlite3.connect(DB_NAME)
# Đọc dữ liệu từ bảng player_data1 vào DataFrame
df = pd.read_sql_query("SELECT * FROM player_data1", conn)
# Đóng kết nối database
conn.close()
# Chuyển đổi tất cả các cột có thể sang kiểu số
for col in df.columns:
    if col not in ['Player', 'Nation', 'Squad', 'Pos']:
        try:
            df[col] = df[col].replace('N/a', pd.NA)
                        df[col]    =    pd.to_numeric(df[col].astype(str).str.replace(',',    ''),
errors='coerce')
        except:
            continue
teams = df.groupby("Squad")
```

- Sử dụng thư viện ‘pandas’ để xử lý và phân tích dữ liệu.

- Đọc dữ liệu từ file sqlite có tên là ‘player_data1.sqlite’ (file kết quả của câu I.1) vào một DataFrame có tên là df.

- Nhóm dữ liệu theo tên đội bóng để thực hiện việc tính toán cho từng đội.

Bước 2: Xác định các chỉ số tấn công và phòng ngự cần thiết cho tính toán:

```
 # Các chỉ số tấn công và phòng ngự
attack_metrics = ['Performance Gls', 'Performance Ast', 'Expected xG', 'Expected
xAG', 'Per 90 Minutes Gls', 'Per 90 Minutes Ast']
defensive_metrics = ['Tackles Tkl', 'Tackles TklW', 'Touches Def 3rd', 'Touches
Att 3rd', 'Blocks', 'Blocks Sh', 'Blocks Pass']
```

Bước 3: Tính điểm phong độ tấn công và phòng ngự:

```
# Tính trung bình các chỉ số tấn công và phòng ngự cho từng đội
team_attack_performance = teams[attack_metrics].mean().mean(axis=1)
```

```
team_defensive_performance = teams[defensive_metrics].mean().mean(axis=1) / 100
```

- Điểm tấn công được tính bằng cách tính trung bình cộng của từng chỉ số tấn công cho mỗi đội bóng và tiếp tục lấy trung bình của các chỉ số đó.

- Điểm phòng ngự cũng được làm tương tự nhưng chia thêm cho 100 để chuẩn hoá dữ liệu phòng ngự với điểm tấn công.

Bước 4: Tính điểm phong độ chung:

```
# Tạo bảng tổng hợp điểm phong độ
team_performance = pd.DataFrame({
    'Attack Performance': team_attack_performance,
    'Defensive Performance': team_defensive_performance
})

# Tính điểm phong độ chung bằng trung bình cộng của điểm tấn công và phòng ngự
team_performance['Overall Performance'] = team_performance.mean(axis=1)

# Định dạng các cột số với 2 chữ số thập phân
for col in team_performance.columns:
    team_performance[col] = team_performance[col].apply(lambda x: f"{x:.2f}")

# Sắp xếp theo thứ tự giảm dần của điểm phong độ chung
team_performance = team_performance.sort_values(by='Overall Performance', ascending=False)
```

- Một bảng dữ liệu được tạo ra để lưu trữ điểm tấn công và điểm phòng ngự của từng đội.

- Điểm phong độ tổng thể sau đó được tính bằng cách lấy trung bình giữa hai giá trị này và ghi vào cột ‘Overall Performance’.

- Cuối cùng, bảng kết quả được sắp xếp theo thứ tự giảm dần của điểm tổng hợp, từ đó xác định đội bóng có phong độ cao nhất.

Bước 5: Ghi dữ liệu ra file CSV:

```
# Lưu kết quả vào file results_PhongDo.csv
team_performance.to_csv('results_PhongDo.csv')
```

```
print("\nĐiểm phong độ của các đội bóng:")
print(team_performance)

# Tìm đội có phong độ cao nhất
best_team = team_performance.index[0]
best_team_score = float(team_performance.iloc[0]['Overall Performance'])

print(f"\nĐội bóng có phong độ tốt nhất của giải bóng đá Ngoại Hạng Anh mùa 2024-2025 là: {best_team}")
print(f"Với các chỉ số:")
print(f"- Điểm tấn công: {team_performance.loc[best_team, 'Attack Performance']}")
print(f"- Điểm phòng ngự: {team_performance.loc[best_team, 'Defensive Performance']}")
print(f"- Điểm tổng hợp: {team_performance.loc[best_team, 'Overall Performance']}")
```

- Kết quả được lưu vào file ‘'results_PhongDo.csv’.

- Đồng thời ta tìm ra đội bóng có phong độ tốt nhất và điểm số phong độ của đội bóng đó và in ra màn hình.

- Đội bóng có phong độ tốt nhất mùa giải Ngoại Hạng Anh 2024-2025 là Liverpool

Kết quả đạt được:

```
Đội bóng có phong độ tốt nhất của giải bóng đá Ngoại Hạng Anh mùa 2024-2025 là: Liverpool
Với các chỉ số:
- Điểm tấn công: 2.30
- Điểm phòng ngự: 1.06
- Điểm tổng hợp: 1.68
```

III.2.

Dựa vào kết quả từ câu I, đề xuất một phương pháp định giá cầu thủ.

- Dựa trên dữ liệu cầu thủ đã thu thập ở phần I, nhóm đề xuất phương pháp định giá cầu thủ dựa theo kết quả tổng hợp của ba yếu tố chính là: hiệu suất thi đấu, mức độ đóng góp và tiềm năng phát triển.

+ Hiệu suất thi đấu: bao gồm số bàn thắng, kiến tạo, khả năng dứt điểm và các chỉ số kỳ vọng (xG, xAG). Đây là yếu tố phản ánh trực tiếp năng lực chuyên môn.

+ Đóng góp cho đội bóng: thể hiện qua số phút thi đấu, tỷ lệ chuyền chính xác, mức độ tham gia vào lối chơi. Yếu tố này phản ánh tầm ảnh hưởng của cầu thủ trong chiến thuật chung.
+ Tuổi và tiềm năng: cầu thủ trẻ thường có khả năng phát triển cao hơn và được định giá cao hơn, trong khi cầu thủ lớn tuổi có xu hướng giảm giá trị chuyển nhượng.
- Sau khi có điểm tổng hợp cho mỗi cầu thủ dựa trên ba yếu tố trên, tiến hàng so sánh với các cầu thủ có giá trị chuyển nhượng thực tế để ước lượng giá trị hợp lý.
- Phương pháp này giúp việc định giá trở nên khách quan hơn, không phụ thuộc cảm tính, đồng thời có thể mở rộng và cải thiện bằng các mô hình học máy trong tương lai.

IV.

- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số tương tự nhau.
- Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao (gợi ý: sử dụng biểu đồ Elbow và Silhoute, vẽ 2 biểu đồ này)? Bạn có Nhận xét gì về kết quả.
- Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, 3 chiều, vẽ hình scatter plot phân cụm các điểm dữ liệu trên mặt 2D, và khối 3D.

IV.1. Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau:

- Để làm bài tập này, em sử dụng thuật toán K-means tìm số cụm tối ưu thông qua phương pháp Elbow.

Bước 1: Lưu tên chương trình Kmeans.py, import các thư viện cần thiết:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import sqlite3
import matplotlib.pyplot as plt
```

- pandas: Dùng để đọc và xử lý dữ liệu dạng bảng (DataFrame).

- numpy: Dùng để tính toán số học nhanh chóng, đặc biệt cho mảng.
- sklearn.cluster.KMeans: Thuật toán phân cụm K-Means.
- sklearn.preprocessing.StandardScaler: Chuẩn hóa dữ liệu (đưa các chỉ số về cùng thang đo).
- sqlite3: Kết nối và truy vấn cơ sở dữ liệu SQLite.
- matplotlib.pyplot: Dùng để vẽ đồ thị (biểu đồ Elbow trong trường hợp này).

Bước 2: Chuẩn bị dữ liệu:

```
# Kết nối đến cơ sở dữ liệu SQLite
DB_NAME = 'player_data1.sqlite'

# Kết nối database và đọc dữ liệu
conn = sqlite3.connect(DB_NAME)
data = pd.read_sql_query("SELECT * FROM player_data1", conn)
conn.close()

# Chọn các chỉ số quan trọng để phân cụm
features = [
    'Age',                  # Tuổi
    'Playing Time MP',      # Số trận đấu
    'Playing Time Min',     # Số phút thi đấu
    'Performance Gls',      # Số bàn thắng
    'Performance Ast',      # Số kiến tạo
    'Expected xG',          # Chỉ số xG
    'Expected xAG',         # Chỉ số xAG
    'Per 90 Minutes Gls',   # Bàn thắng/90 phút
    'Per 90 Minutes Ast'    # Kiến tạo/90 phút
]

# Chuẩn bị dữ liệu
X = data[features].copy()
```

- Đọc dữ liệu từ file sqlite: Ta tiến hành đọc dữ liệu từ file 'player_data1.sqlite' và lưu vào DataFrame có tên là 'data'.
- Đây là các cột chứa chỉ số thể hiện phong độ cầu thủ.
- Các chỉ số này sẽ được dùng để nhóm các cầu thủ có phong độ tương tự nhau.
- Lấy ra các cột cần thiết từ data thành bảng mới X.
- Dùng .copy() để đảm bảo thao tác không ảnh hưởng đến data gốc.

Bước 3: Làm sạch dữ liệu:

```
# Xử lý dữ liệu thiếu và chuyển đổi sang kiểu số
for col in X.columns:
    X[col] = pd.to_numeric(X[col].astype(str).str.replace(',', ''), errors='coerce')
X = X.fillna(X.mean())
```

- Vòng lặp qua từng cột để:

+ astype(str): Chuyển giá trị sang chuỗi, phòng khi dữ liệu bị lẫn kiểu.

+ .str.replace(',', ''): Xóa dấu phẩy (ví dụ "1,234" → "1234").

+ pd.to_numeric(..., errors='coerce'): Chuyển sang kiểu số, nếu có giá trị không hợp lệ thì đổi thành NaN.

+ X.fillna(X.mean()): Điền giá trị trung bình của từng cột vào các ô bị thiếu (NaN).

Bước 4: Chuẩn hóa dữ liệu:

```
# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- StandardScaler() chuẩn hóa mỗi cột sao cho:

+ Trung bình = 0

+ Độ lệch chuẩn = 1

- fit_transform(X) học thông số chuẩn hóa và áp dụng vào X.

- Kết quả X_scaled là mảng số thực (numpy array) dùng để huấn luyện mô hình K-Means.

Bước 5: Tìm số cụm tối ưu bằng phương pháp Elbow:

```
# Tìm k tối ưu bằng phương pháp Elbow
wcss = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

- wcss: danh sách chứa giá trị Within-Cluster Sum of Squares (tổng bình phương sai số trong cụm).

Tức là WCSS càng nhỏ thì các điểm trong cụm càng gần nhau.

- K = range(1, 11): Thử nghiệm số cụm từ 1 đến 10.

- Với mỗi giá trị k:

  - Khởi tạo mô hình KMeans:
    - n_clusters=k: số cụm.

- init='k-means++': cách chọn tâm cụm tối ưu hơn khởi tạo ngẫu nhiên.
- max_iter=300: số lần lặp tối đa.
- n_init=10: chạy lại 10 lần để chọn kết quả tốt nhất.
- random_state=42: để kết quả có thể tái lập.

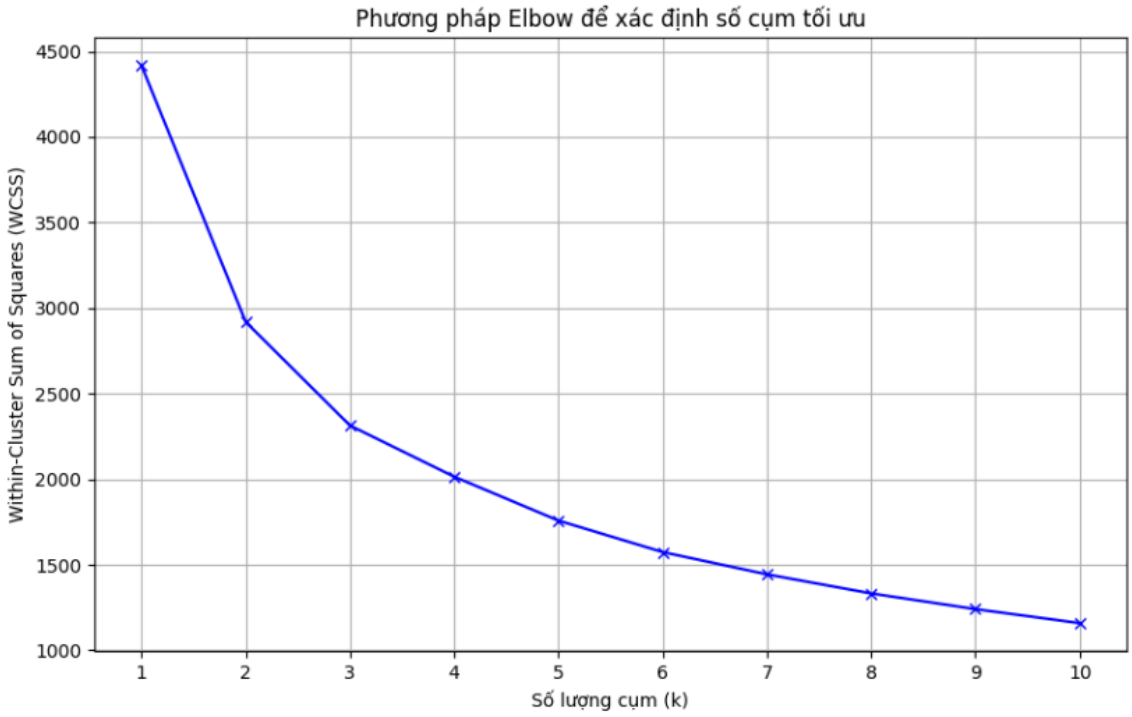
- fit(X_scaled): huấn luyện mô hình.
- kmeans.inertia_: chính là WCSS mô hình, thêm vào danh sách wcss.

Bước 6: Vẽ biểu đồ Elbow và hiển thị:

```
# Vẽ biểu đồ Elbow
plt.figure(figsize=(10, 6))
plt.plot(K, wcss, 'bx-')
plt.xlabel('Số lượng cụm (k)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.title('Phương pháp Elbow để xác định số cụm tối ưu')
plt.xticks(K)
plt.grid(True)
plt.show()
```

- Tạo biểu đồ với trục X là k, trục Y là WCSS.
- 'bx-': vẽ đường nối giữa các điểm bằng dấu X màu xanh.
- Gắn nhãn, tiêu đề, bật lưới.

Bước 7: Xác định số cụm tối ưu: Dựa vào biểu đồ Elbow ta vẽ được, ta có thể thấy với k = 3 là giá trị gần như tối ưu nhất -> độ dốc giảm chậm



IV.2. Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có nhận xét gì về kết quả:

- Dựa vào kết quả biểu đồ Elbow, ta có thể thấy tại k = 3 là điểm "gấp khúc" rõ rệt, thể hiện rằng khi tăng số cụm lớn hơn 3 thì độ giảm của WCSS (Within-Cluster Sum of Squares) không còn đáng kể. Do đó, k = 3 được xem là số cụm tối ưu cho bài toán này.
- Trong bóng đá, mỗi cầu thủ có thể đảm nhiệm nhiều vai trò và sở hữu những phong cách thi đấu rất đa dạng. Do đó, việc phân chia họ thành 3 nhóm dựa trên mức độ tương đồng về các chỉ số thống kê là hợp lý hơn so với cách phân loại truyền thống dựa trên vị trí thi đấu cố định trên sân.
- Nhận xét về kết quả:

Kết quả phân cụm nhìn chung khá hợp lý — mô hình đã chia dữ liệu thành 3 nhóm có đặc điểm tương đồng cao, giúp việc phân tích và đánh giá cầu thủ trở nên có hệ thống hơn. Tuy nhiên, do một số chỉ số trong dữ liệu còn thiếu hoặc chưa thật đồng nhất (ví dụ các giá trị "N/A"), kết quả phân nhóm có thể chưa phản ánh hoàn toàn chính xác thực tế. Dù vậy, kết quả này vẫn mang ý nghĩa tham khảo tốt và cho thấy mô hình của thuật toán K-means hoạt động đúng hướng.

IV.3. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, 3 chiều và vẽ hình phân cụm các điểm dữ liệu trên mặt 2D, khối 3D:

* Vẽ hình phân cụm các điểm dữ liệu trên mặt 2D:

- Dựa vào kết quả của bài K-means vừa rồi, ta tiến hành giảm số chiều dữ liệu xuống 2 chiều bằng thuật toán PCA.

Bước 1: Lưu tên chương trình PCA_2D.py, import các thư viện cần thiết:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import sqlite3
import matplotlib.pyplot as plt
```

- pandas as pd: thao tác dữ liệu bảng (DataFrame).

- numpy as np: toán học trên mảng, tính toán số học (ví dụ np.cumsum).

- KMeans: thuật toán phân cụm K-Means.

- StandardScaler: chuẩn hóa (mean=0, std=1). Quan trọng để các biến có đơn vị khác nhau không làm lệch kết quả.

- PCA: phân tích thành phần chính — giảm chiều dữ liệu.

- sqlite3: kết nối và truy vấn database SQLite.

- matplotlib.pyplot as plt: vẽ đồ thị.

Bước 2: Chuẩn bị dữ liệu:

```
# Kết nối đến cơ sở dữ liệu SQLite
DB_NAME = 'player_data1.sqlite'

# Kết nối database và đọc dữ liệu
conn = sqlite3.connect(DB_NAME)
data = pd.read_sql_query("SELECT * FROM player_data1", conn)
conn.close()

# Chọn các chỉ số quan trọng để phân tích
features = [
    'Age',                  # Tuổi
    'Playing Time MP',      # Số trận đấu
    'Playing Time Min',     # Số phút thi đấu
    'Performance Gls',      # Số bàn thắng
    'Performance Ast',      # Số kiến tạo
    'Expected xG',          # Chỉ số xG
    'Expected xAG',         # Chỉ số xAG
    'Per 90 Minutes Gls',   # Bàn thắng/90 phút
    'Per 90 Minutes Ast'    # Kiến tạo/90 phút
]

# Chuẩn bị dữ liệu
X = data[features].copy()
```

- DB_NAME = 'player_data1.sqlite': tên file DB.

- sqlite3.connect(DB_NAME): mở kết nối tới file SQLite.

- pd.read_sql_query("SELECT * FROM player_data1", conn): chạy câu lệnh SQL lấy toàn bộ bảng player_data1 vào DataFrame data.

- conn.close(): đóng kết nối — luôn nên làm để tránh leak/khóa file.

- Danh sách tên cột dùng để xây ma trận đặc trưng X.

- Chọn features có ý nghĩa với phân cụm (độ tương tự về chơi bóng).

- Lấy một bản sao (.copy()) của các cột quan tâm từ data vào X để thao tác không làm thay đổi data gốc.

Bước 3: Xử lý dữ liệu thiếu và ép kiểu số:

```
# Xử lý dữ liệu thiếu và chuyển đổi sang kiểu số
for col in X.columns:
    X[col] = pd.to_numeric(X[col].astype(str).str.replace(',', ''), errors='coerce')
X = X.fillna(X.mean())
```

- Vòng lặp qua từng cột col trong X:

+ X[col].astype(str): chuyển mọi giá trị thành chuỗi (để thao tác str.replace an toàn).

+ str.replace(',', ''): loại bỏ dấu phẩy nếu dữ liệu ở dạng "1,234" — tránh lỗi khi chuyển sang số.

+ pd.to_numeric(..., errors='coerce'): chuyển chuỗi thành số; nếu không chuyển được (vd: chữ), sẽ đặt là NaN (do errors='coerce').

+ X.fillna(X.mean()): thay tất cả giá trị NaN bằng **trung bình cột tương ứng**.

Bước 4: Chuẩn hóa dữ liệu:

```
# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- StandardScaler() tính mean và std cho từng cột.

- fit_transform(X) vừa học các tham số chuẩn hóa từ X, vừa biến X về dạng có mean≈0 và std≈1.

- X_scaled là array numpy, sẵn sàng để dùng cho PCA và K-Means.

- Vì K-Means và PCA nhạy với tỷ lệ, chuẩn hóa trước là bắt buộc nếu các biến có đơn vị khác nhau.

Bước 5: PCA — giảm chiều về 2 thành phần chính:

```
# Áp dụng PCA để giảm chiều dữ liệu xuống 2 chiều
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)
```

- PCA(n_components=2): khởi tạo PCA để chiết xuất 2 thành phần chính (2 trục mới).

- pca.fit_transform(X_scaled): học các thành phần chính từ X_scaled và trả về biến đổi (ma trận kích thước n_samples x 2).

- principal_components chứa tọa độ của mỗi cầu thủ trên 2 trục chính (PC1, PC2).

Bước 6: Phần trăm phương sai được giải thích:

```
# Tính phần trăm phương sai giải thích được
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)
```

- pca.explained_variance_ratio_: array có 2 phần tử, mỗi phần tử là tỉ lệ phương sai (variance) mà mỗi PC giải thích. Ví dụ [0.45, 0.25] nghĩa là PC1 giải thích 45% và PC2 giải thích 25% tổng phương sai.

- np.cumsum(...): tính tổng cộng dồn (useful nếu bạn muốn biết PC1+PC2 giải thích bao nhiêu %).

Bước 7: Phân cụm bằng K-means:

```
# Phân cụm với k = 3
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
```

- KMeans:

+ n_clusters=3: chọn 3 cụm.

+ init='k-means++': khởi tạo tâm cụm hiệu quả (giảm khả năng hội tụ vào cực tệ).

+ max_iter=300: số vòng lặp tối đa cho mỗi khởi tạo.

+ n_init=10: chạy thuật toán 10 lần với các khởi tạo khác nhau rồi chọn kết quả có inertia nhỏ nhất.

+ random_state=42: để kết quả tái lập được.

+ kmeans.fit_predict(X_scaled): vừa huấn luyện mô hình trên X_scaled, vừa trả về clusters là mảng nhãn (0..4) cho mỗi mẫu.

Bước 8: Tạo DataFrame chứa kết quả PCA + cluster + thông tin cầu thủ:

```
# Tạo DataFrame cho dữ liệu PCA
data_pca = pd.DataFrame(data=principal_components, columns=['Principal Component 1', 'Principal Component 2'])
data_pca['Cluster'] = clusters
data_pca['Player'] = data['Player']
data_pca['Position'] = data['Pos']
data_pca['Squad'] = data['Squad']
```

- Tạo DataFrame data_pca với 2 cột PC1/PC2.

- Thêm cột Cluster gán nhãn cụm cho mỗi hàng.

- data['Player'], data['Pos'], data['Squad'] — copy thêm thông tin mô tả (tên, vị trí, đội).

Bước 9: Vẽ biểu đồ phân cụm:

```
# Vẽ biểu đồ phân cụm
plt.figure(figsize=(12, 8))

# Định nghĩa màu sắc cho từng cụm
colors = ['#FF9999', '#66B2FF', '#99FF99']

# Vẽ scatter plot với màu sắc theo cụm
for i in range(3):
    cluster_data = data_pca[data_pca['Cluster'] == i]
        plt.scatter(cluster_data['Principal Component 1'], cluster_data['Principal Component 2'], c=[colors[i]], label=f'Cụm {i+1}', alpha=0.6)

    # Thêm tên một số cầu thủ tiêu biểu trong mỗi cụm
    for idx, row in cluster_data.head(3).iterrows():
            plt.annotate(row['Player'], (row['Principal Component 1'], row['Principal Component 2']), xytext=(5, 5), textcoords='offset points', fontsize=8, alpha=0.7)
```

- Tạo figure mới kích thước 12x8 inch.

- Lặp i từ 0 tới 2 (tương ứng 3 cụm).

- cluster_data = data_pca[data_pca['Cluster'] == i]: trích dữ liệu của cụm i.

- plt.scatter(...):

    - Trục x = PC1, trục y = PC2.
    - c=[colors[i]]: chỉ định màu cho cụm đó. (Đặt c là danh sách sao cho matplotlib không coi mỗi điểm là một màu khác nhau.)
    - label=f'Cụm {i+1}': nhãn cho legend.
    - alpha=0.6: trong suốt 0.6 để nhìn tốt khi các điểm chồng lên nhau.

- cluster_data.head(3).iterrows(): lấy ba hàng đầu của từng cụm để gắn nhãn tên cầu thủ (ví dụ muốn hiển thị những cầu thủ "tiêu biểu").

    - plt.annotate(...) đặt tên tại vị trí điểm, với offset (5,5) để chữ không chồng trực tiếp lên marker.

Bước 10: Thêm tiêu đề, nhãn, legend, lưới, hiển thị kết quả:

```
plt.title('Phân cụm cầu thủ sau khi giảm chiều dữ liệu bằng PCA', pad=20)
plt.xlabel(f'Principal Component 1 ({explained_variance_ratio[0]:.1%} variance explained)')
```

```
plt.ylabel(f'Principal Component 2 ({explained_variance_ratio[1]:.1%} variance explained)')
# Thêm legend với tên các cụm
plt.legend(title="Phân loại cầu thủ", bbox_to_anchor=(1.05, 1), loc='upper left')
# Thêm lưới
plt.grid(True, linestyle='--', alpha=0.7)
```

- plt.title(...) đặt tiêu đề, pad=20 đẩy tiêu đề ra xa trục.

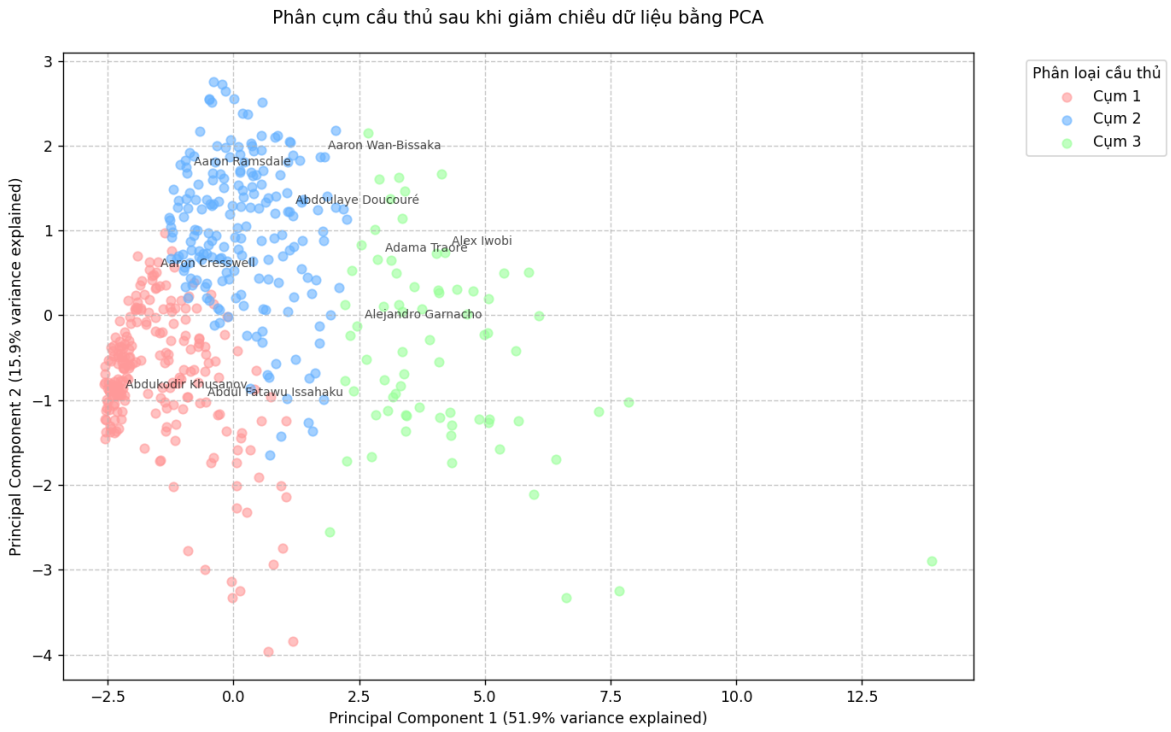
- plt.xlabel(...) và plt.ylabel(...) gắn tên trục kèm phần trăm phương sai PC giải thích (format :.1% hiển thị dạng phần trăm với 1 chữ số thập phân).

- plt.legend(...) đặt legend bên ngoài biểu đồ (với bbox_to_anchor=(1.05,1) legend nằm ở bên phải).

- plt.grid(...) bật lưới với style đứt nét và alpha.

```
plt.show()
```

Kết quả đạt được:



* Vẽ hình phân cụm các điểm dữ liệu trên mặt 3D:

- Dựa vào kết quả của bài K-means vừa rồi, ta tiến hành giảm số chiều dữ liệu xuống 3 chiều bằng thuật toán PCA.

Bước 1: Lưu tên chương trình PCA_3D.py, import các thư viện cần thiết:

```
import pandas as pd
```

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import sqlite3
import matplotlib.pyplot as plt
```

- pandas as pd: thao tác dữ liệu bảng (DataFrame).

- numpy as np: toán học trên mảng, tính toán số học (ví dụ np.cumsum).

- KMeans: thuật toán phân cụm K-Means.

- StandardScaler: chuẩn hóa (mean=0, std=1). Quan trọng để các biến có đơn vị khác nhau không làm lệch kết quả.

- PCA: phân tích thành phần chính — giảm chiều dữ liệu.

- sqlite3: kết nối và truy vấn database SQLite.

- matplotlib.pyplot as plt: vẽ đồ thị.

Bước 2: Chuẩn bị dữ liệu:

```
# Kết nối đến cơ sở dữ liệu SQLite
DB_NAME = 'player_data1.sqlite'
# Kết nối database và đọc dữ liệu
conn = sqlite3.connect(DB_NAME)
data = pd.read_sql_query("SELECT * FROM player_data1", conn)
conn.close()
# Chọn các chỉ số quan trọng để phân tích
features = [
    'Age',                  # Tuổi
    'Playing Time MP',      # Số trận đấu
    'Playing Time Min',     # Số phút thi đấu
    'Performance Gls',      # Số bàn thắng
    'Performance Ast',      # Số kiến tạo
    'Expected xG',          # Chỉ số xG
    'Expected xAG',         # Chỉ số xAG
    'Per 90 Minutes Gls',   # Bàn thắng/90 phút
    'Per 90 Minutes Ast'    # Kiến tạo/90 phút
]

# Chuẩn bị dữ liệu
```

```
X = data[features].copy()
```

- DB_NAME = 'player_data1.sqlite': tên file DB.

- sqlite3.connect(DB_NAME): mở kết nối tới file SQLite.

- pd.read_sql_query("SELECT * FROM player_data1", conn): chạy câu lệnh SQL lấy toàn bộ bảng player_data1 vào DataFrame data.

- conn.close(): đóng kết nối — luôn nên làm để tránh leak/khóa file.

- Danh sách tên cột dùng để xây ma trận đặc trưng X.

- Chọn features có ý nghĩa với phân cụm (độ tương tự về chơi bóng).

- Lấy một bản sao (.copy()) của các cột quan tâm từ data vào X để thao tác không làm thay đổi data gốc.

Bước 3: Xử lý dữ liệu thiếu và ép kiểu số:

```
# Xử lý dữ liệu thiếu và chuyển đổi sang kiểu số
for col in X.columns:
    X[col] = pd.to_numeric(X[col].astype(str).str.replace(',', ''), errors='coerce')
X = X.fillna(X.mean())
```

- Vòng lặp qua từng cột col trong X:

+ X[col].astype(str): chuyển mọi giá trị thành chuỗi (để thao tác str.replace an toàn).

+ str.replace(',', ''): loại bỏ dấu phẩy nếu dữ liệu ở dạng "1,234" — tránh lỗi khi chuyển sang số.

+ pd.to_numeric(..., errors='coerce'): chuyển chuỗi thành số; nếu không chuyển được (vd: chữ), sẽ đặt là NaN (do errors='coerce').

+ X.fillna(X.mean()): thay tất cả giá trị NaN bằng trung bình cột tương ứng.

Bước 4: Chuẩn hóa dữ liệu:

```
# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- StandardScaler() tính mean và std cho từng cột.

- fit_transform(X) vừa học các tham số chuẩn hóa từ X, vừa biến X về dạng có mean≈0 và std≈1.

- X_scaled là array numpy, sẵn sàng để dùng cho PCA và K-Means.

- Vì K-Means và PCA nhạy với tỷ lệ, chuẩn hóa trước là bắt buộc nếu các biến có đơn vị khác nhau.

Bước 5: PCA — giảm chiều về 3 thành phần chính:

```
# Áp dụng PCA để giảm chiều dữ liệu xuống 3 chiều
pca = PCA(n_components=3)
principal_components = pca.fit_transform(X_scaled)
```

- PCA(n_components=3): yêu cầu 3 thành phần chính (PC1, PC2, PC3).
- pca.fit_transform(X_scaled): học ma trận chuyển đổi từ X_scaled và trả về tọa độ của mỗi mẫu trong không gian 3 chiều mới.
- principal_components có kích thước (n_samples, 3).

Bước 6: Phần trăm phương sai được giải thích:

```
# Tính phần trăm phương sai giải thích được
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)
```

- explained_variance_ratio_: mảng 3 phần tử, mỗi phần tử là tỉ lệ phần trăm phương sai mà mỗi PC giải thích.
- np.cumsum(...): tính tổng cộng dồn (PC1 + PC2 + PC3) — cho biết 3 PC giữ lại bao nhiêu thông tin ban đầu.

Bước 7: Phân cụm bằng K-means:

```
# Phân cụm với k = 3
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
```

- Khởi tạo KMeans:
    - n_clusters=3: giả thiết muốn 3 cụm. (Bạn có thể chọn khác bằng Elbow/Silhouette).
    - init='k-means++': khởi tạo trọng tâm tốt hơn random.
    - max_iter=300, n_init=10: giới hạn vòng lặp và số lần chạy để chọn kết quả tốt nhất.
    - random_state=42: để kết quả có thể tái lập.
- fit_predict(X_scaled): vừa huấn luyện vừa trả về mảng nhãn clusters (0.4) cho từng mẫu.

Bước 8: Tạo DataFrame chứa kết quả PCA + cluster + thông tin cầu thủ:

```
# Tạo DataFrame cho dữ liệu PCA
data_pca = pd.DataFrame(
    data=principal_components,
    columns=['Principal Component 1', 'Principal Component 2', 'Principal Component 3']
)
data_pca['Cluster'] = clusters
```

```
data_pca['Player'] = data['Player']
data_pca['Position'] = data['Pos']
data_pca['Squad'] = data['Squad']
```

- Tạo DataFrame data_pca với 3 cột mới (PC1, PC2, PC3).
- Thêm cột Cluster chứa nhãn cụm.
- Thêm thông tin mô tả (Player, Pos, Squad) từ data để khi hiển thị dễ nhận biết.

Bước 9: Chuẩn bị figure 3D và trục:

```
# Tạo biểu đồ 3D
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
```

- plt.figure(figsize=(12,8)): tạo figure kích thước 12×8 inch.
- fig.add_subplot(111, projection='3d'): thêm một subplot 3D; ax là đối tượng trục 3D (sử dụng mpl_toolkits.mplot3d nội bộ của matplotlib).

Bước 10: Vẽ scatter plot 3D theo cụm và annotate:

```
# Định nghĩa màu sắc cho từng cụm
colors = ['#FF9999', '#66B2FF', '#99FF99']

# Vẽ scatter plot 3D với màu sắc theo cụm
for i in range(3):
    cluster_data = data_pca[data_pca['Cluster'] == i]
    ax.scatter(
        cluster_data['Principal Component 1'],
        cluster_data['Principal Component 2'],
        cluster_data['Principal Component 3'],
        c=[colors[i]],
        label=f'Cụm {i+1}',
        alpha=0.6
    )

    # Thêm tên một số cầu thủ tiêu biểu trong mỗi cụm
    for idx, row in cluster_data.head(3).iterrows():
        ax.text(
            row['Principal Component 1'],
            row['Principal Component 2'],
            row['Principal Component 3'],
            row['Player'],
            fontsize=8,
```

```
        alpha=0.7
    )
```

- Lặp i từ 0 tới 2:

+ cluster_data = data_pca[data_pca['Cluster'] == i]: lấy các hàng thuộc cụm i.

+ ax.scatter(...) vẽ điểm trong không gian 3 chiều:

  - trục x = PC1, y = PC2, z = PC3.
  - c=[colors[i]]: gán màu cho điểm; dùng list nhỏ là ổn để matplotlib nhận một màu cho cả nhóm điểm.
  - label=f'Cụm {i+1}': nhãn dùng trong legend.
  - alpha=0.6: độ trong suốt.

+ cluster_data.head(3).iterrows(): chọn 3 hàng đầu của cụm để gắn nhãn tên cầu thủ (ví dụ tiêu biểu).

+ ax.text(...): đặt văn bản 3D tại tọa độ tương ứng; ax.text tiện cho 3D (khác annotate hay 2D).

Bước 11: Tiêu đề, nhãn trục, legend, điều chỉnh góc nhìn, hiển thị kết quả:

```
# Thêm tiêu đề và nhãn
ax.set_title('Phân cụm cầu thủ trong không gian 3D sau khi giảm chiều dữ liệu bằng PCA', pad=20)
ax.set_xlabel(f'PC1 ({explained_variance_ratio[0]:.1%} variance)')
ax.set_ylabel(f'PC2 ({explained_variance_ratio[1]:.1%} variance)')
ax.set_zlabel(f'PC3 ({explained_variance_ratio[2]:.1%} variance)')

# Thêm legend
ax.legend(title="Phân loại cầu thủ", bbox_to_anchor=(1.05, 1), loc='upper left')

# Điều chỉnh góc nhìn
ax.view_init(elev=20, azim=45)

plt.tight_layout()
```

- set_title(...): tiêu đề biểu đồ (3D). pad=20 đẩy tiêu đề ra xa trục.

- set_xlabel/ylabel/zlabel(...): gán tên trục kèm phần trăm phương sai PC giải thích (format :.1%).

- ax.legend(...): thêm legend; bbox_to_anchor=(1.05,1) cố gắng đặt legend bên phải đồ họa. (Trong 3D, vị trí legend có thể cần tinh chỉnh).

- ax.view_init(elev=20, azim=45): thiết lập góc nhìn camera:

  - elev: độ cao camera (elevation).
  - azim: góc quay quanh trục z (azimuth).

- Thử nhiều elev/azim để tìm góc nhìn tốt nhất.
- plt.tight_layout(): tự động chỉnh margin để tránh các label/legend bị cắt.

```
plt.show()
```

Kết quả đạt được: