Data: /results.csv
File crawl_data: /crawl_data.ipynb

Data Information:

```
df.info
```

```
<bound method DataFrame.info of                Player   Nation    Pos          Squad   Age  Playing_Time_MP  \
0      Aaron Cresswell  eng ENG  DF,FW       West Ham  33.0              11
1       Aaron Ramsdale  eng ENG     GK        Arsenal  25.0               6
2    Aaron Wan-Bissaka  eng ENG     DF  Manchester Utd  25.0              22
3         Aaron Hickey  sct SCO     DF       Brentford  21.0               9
4         Aaron Ramsey  eng ENG  MF,FW         Burnley  20.0              14
..                 ...      ...    ...             ...   ...             ...
488      Yves Bissouma   ml MLI     MF       Tottenham  26.0              28
489       Zeki Amdouni   ch SUI     FW         Burnley  22.0              34
490        Álex Moreno   es ESP     DF     Aston Villa  30.0              21
491     Đorđe Petrović   rs SRB     GK         Chelsea  23.0              23
492   Łukasz Fabiański   pl POL     GK        West Ham  38.0              10

     Playing_Time_Starts  Playing_Time_Min  Performance_Ast  Performance_G-PK  \
0                      4               436                0                 0
1                      6               540                0                 0
2                     20              1780                2                 0
3                      9               713                0                 0
4                      5               527                0                 0
..                   ...               ...              ...               ...
488                   26              2068                0                 0
489                   27              1953                1                 4
490                   11              1031                0                 2
491                   22              1987                0                 0
492                    7               721                0                 0
...
490                4                10              28.6
491                5                 0             100.0
492                2                 0             100.0

[493 rows x 172 columns]>
```

- Gồm 493 cầu thủ, mỗi cầu thủ có 172 thuộc tính, chia làm 20 đội bóng ở ngoại hạng anh.

TOP 3 CẦU THỦ CÓ ĐIỂM CAO - THẤP NHẤT Ở MỖI CHỈ SỐ:
path: /result2_1.txt
Ảnh minh họa:

```
Chỉ số: Age
Top 3 cầu thủ có điểm cao nhất:
              Player  Age
    Ashley Young 38.0
    Thiago Silva 38.0
Łukasz Fabiański 38.0

Top 3 cầu thủ có điểm thấp nhất:
      Player  Age
Leon Chiwome 17.0
 Lewis Miley 17.0
  David Ozoh 18.0
```

```
Chỉ số: Playing_Time_MP
Top 3 cầu thủ có điểm cao nhất:
        Player  Playing_Time_MP
   André Onana              38
    Bernd Leno              38
Carlton Morris              38

Top 3 cầu thủ có điểm thấp nhất:
       Player  Playing_Time_MP
   Alex Iwobi               2
   Ionuț Radu               2
Matheus Nunes               2
```

Solution:

```
top3 = {}

for col in df.select_dtypes(include='number').columns:
    top_3_highest = df.nlargest(3, col)[['Player', col]]
    top_3_lowest = df.nsmallest(3, col)[['Player', col]]

    top3[col] = {
        'Top 3 Highest': top_3_highest,
        'Top 3 Lowest': top_3_lowest
    }

with open('result2_1.txt', 'w') as f:
    for stat, values in top3.items():
        f.write(f"Chỉ số: {stat}\n")
        f.write("Top 3 cầu thủ có điểm cao nhất:\n")
        f.write(values['Top 3 Highest'].to_string(index=False) + "\n\n")
        f.write("Top 3 cầu thủ có điểm thấp nhất:\n")
        f.write(values['Top 3 Lowest'].to_string(index=False) + "\n\n")
        f.write("-" * 50 + "\n\n")
```

TRUNG VỊ MỖI CHỈ SỐ, TRUNG BÌNH VÀ ĐỘ LỆCH CHUẨN MỖI CHỈ SỐ CHO CÁC CẦU THỦ TRONG TOÀN GIẢI VÀ CỦA MỖI ĐỘI:

Ảnh minh họa:

	Team	Median of Age	Mean of Age	Std of Age	Median of Playing_Time_MP	Mean of Playing_Time_MP	Std of Playing_Time_MP	Median of Playing_Time_Starts	Mean of Playing_Time_Starts	Std of Playing_Time_Starts
0	all	25.0	25.498986	4.127355	23.0	22.657201	10.136975	16.0	16.941176	11.167179
1	West Ham	27.5	28.272727	3.869069	23.5	23.363636	10.825655	21.0	19.000000	13.511900
2	Arsenal	24.0	24.761905	2.547641	27.0	26.809524	10.191266	18.0	19.857143	13.093073
3	Manchester Utd	25.5	25.269231	4.414138	22.0	21.500000	10.052860	15.0	16.038462	11.039858
4	Brentford	26.0	25.800000	3.593976	26.0	22.960000	10.346014	15.0	16.720000	10.883933
5	Burnley	24.0	24.071429	3.838678	16.0	20.392857	9.346575	14.0	14.928571	10.014540
6	Everton	26.0	26.347826	4.858064	28.0	23.304348	11.561829	23.0	18.173913	13.720099
7	Brighton	23.5	24.785714	5.698324	20.0	20.928571	8.751417	15.0	14.892857	8.603786
8	Bournemouth	24.5	25.038462	3.538144	25.5	22.076923	11.852166	13.0	16.038462	12.732575
9	Crystal Palace	25.5	25.166667	4.280051	22.5	22.458333	9.477567	17.5	17.416667	10.993740
10	Fulham	27.0	27.904762	3.360130	29.0	27.238095	7.993152	18.0	19.904762	10.084170
11	Luton Town	26.0	26.320000	3.051229	23.0	22.840000	9.163696	16.0	16.720000	10.159232
12	Newcastle Utd	25.5	26.125000	4.875070	21.0	22.875000	8.679373	14.5	17.333333	10.773021
13	Liverpool	24.0	25.318182	3.822071	28.0	25.863636	8.993624	17.0	18.954545	8.283531
14	Chelsea	22.0	23.000000	3.905125	23.0	21.880000	9.404432	18.0	16.720000	11.066165
15	Sheffield Utd	24.0	25.166667	4.259540	14.5	18.800000	10.584308	11.0	13.933333	10.550154
16	Nott'ham Forest	25.5	25.900000	3.880544	20.0	19.000000	9.955071	15.0	13.933333	8.642052
17	Tottenham	25.5	25.125000	3.530150	27.5	23.750000	10.927628	15.5	17.416667	12.693431
18	Manchester City	27.0	26.000000	4.024922	29.0	24.952381	9.351343	24.0	19.904762	11.330952
19	Aston Villa	26.0	25.956522	3.548089	27.0	24.173913	11.109587	20.0	18.130435	12.392462
20	Wolves	24.0	24.680000	4.422669	25.0	22.480000	11.930773	11.0	16.720000	13.358892

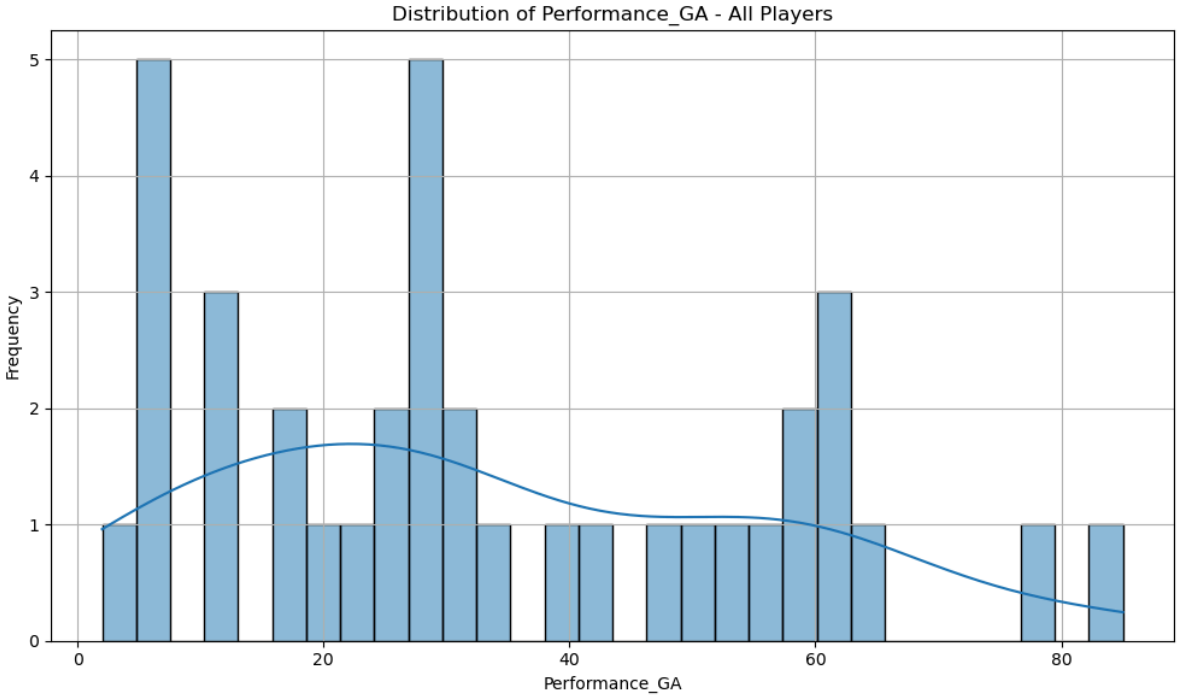
Path: /result2_2.csv

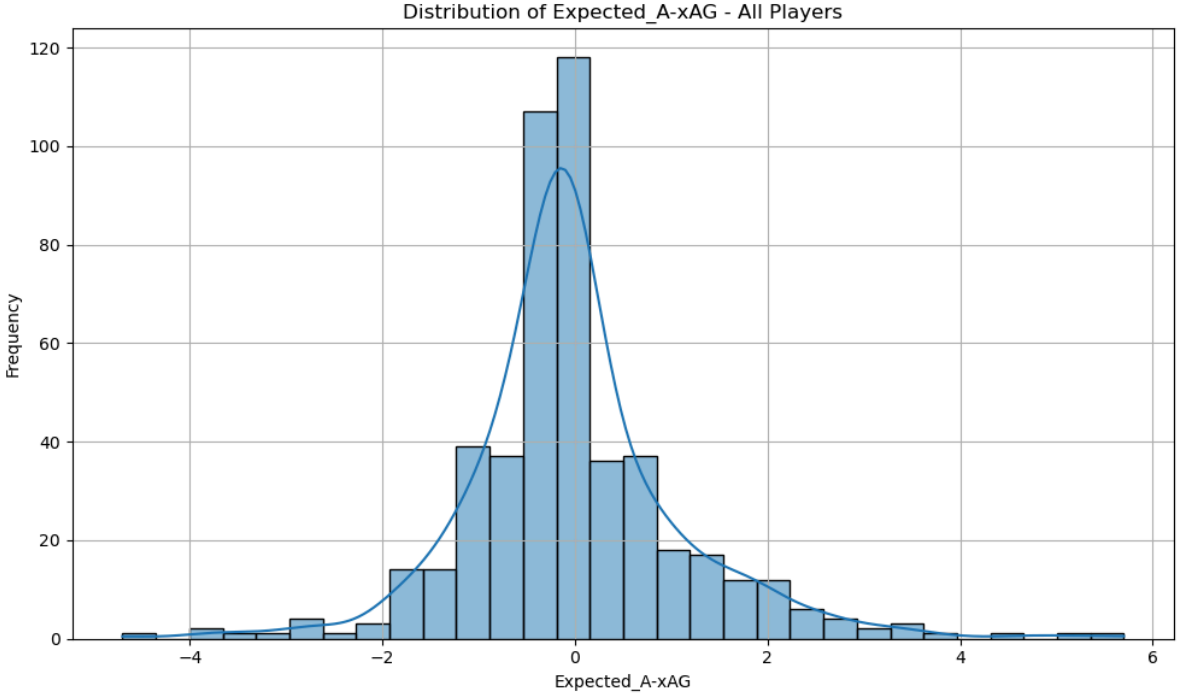
Solution:

```python
# trung vị của mỗi chỉ số, trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ tro
numeric_columns = df.select_dtypes(include='number').columns

results2_2 = pd.DataFrame()

all_stats = {
    'Team': 'all'
}
for col in numeric_columns:
    all_stats[f'Median of {col}'] = df[col].median()
    all_stats[f'Mean of {col}'] = df[col].mean()
    all_stats[f'Std of {col}'] = df[col].std()

results2_2 = pd.concat([results2_2, pd.DataFrame([all_stats])], ignore_index=True)

for team in df['Squad'].unique():
    team_stats = {
        'Team': team
    }
    team_data = df[df['Squad'] == team]
    for col in numeric_columns:
        team_stats[f'Median of {col}'] = team_data[col].median()
        team_stats[f'Mean of {col}'] = team_data[col].mean()
        team_stats[f'Std of {col}'] = team_data[col].std()

    results2_2 = pd.concat([results2_2, pd.DataFrame([team_stats])], ignore_index=True)

results2_2.to_csv('results2_2.csv', index=False)
```

HISTOGRAM PHÂN BỐ CỦA MỖI CHỈ SỐ CỦA CÁC CẦU THỦ TRONG TOÀN GIẢI VÀ MỖI ĐỘI

path: /result2_3/
Ảnh minh họa:


Distribution of Performance_GA - All Players
Frequency
Performance_GA
0
1
2
3
4
5
0
20
40
60
80


Distribution of Expected_A-xAG - All Players
Frequency
Expected_A-xAG
0
20
40
60
80
100
120
−4
−2
0
2
4
6

Solution:

```
output_dir = 'result2_3'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

for col in numeric_columns:
    plt.figure(figsize=(10, 6))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col} - All Players')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.tight_layout()
    name = normalize(col)
    plt.savefig(os.path.join(output_dir, f'all_players_{name}.png'))
    plt.close()

for team in df['Squad'].unique():
    team_data = df[df['Squad'] == team]

    team_dir = os.path.join(output_dir, team)
    if not os.path.exists(team_dir):
        os.makedirs(team_dir)

    for col in numeric_columns:
        plt.figure(figsize=(10, 6))
        sns.histplot(team_data[col], kde=True, bins=30)
        plt.title(f'Distribution of {col} - {team}')
        plt.xlabel(col)
        plt.ylabel('Frequency')
        plt.grid(True)
        plt.tight_layout()
        name = normalize(col)
        plt.savefig(os.path.join(team_dir, f'{team}_{name}.png'))
        plt.close()
```

ĐỘI BÓNG CÓ PHONG ĐỘ CAO NHẤT: ARSENAL

ĐỘI BÓNG CÓ CHỈ SỐ CAO NHẤT Ở MỖI CHỈ SỐ:

Path: /result2_4.txt

Ảnh minh họa:

```
Teams with the highest score in each stat:
Playing_Time_MP: Manchester Utd
Playing_Time_Starts: Manchester Utd
Playing_Time_Min: Manchester Utd
Performance_Ast: Aston Villa
Performance_G-PK: Manchester City
Performance_PK: Chelsea
Performance_CrdY: Fulham
Performance_CrdR: Sheffield Utd
Expected_xG_x: Manchester City
Expected_npxG_x: Manchester City
Expected_xAG: Manchester Utd
Progression_PrgC: Manchester City
Progression_PrgP: Manchester City
Progression_PrgR: Arsenal
Per_90_Minutes_Gls: Wolves
Per_90_Minutes_Ast: Tottenham
Per_90_Minutes_G+A: Fulham
Per_90_Minutes_G-PK: Wolves
Per_90_Minutes_G+A-PK: Fulham
Per_90_Minutes_xG: Bournemouth
Per_90_Minutes_xAG: Manchester City
Per_90_Minutes_xG+xAG: Bournemouth
Per_90_Minutes_npxG: Bournemouth
Per_90_Minutes_npxG+xAG: Bournemouth
Performance_GA: Luton Town
Performance_GA90: Bournemouth
Performance_SoTA: Luton Town
Performance_Saves: Manchester Utd
Performance_Save%: Burnley
Performance_W: Manchester City
Performance_D: West Ham
Performance_L: Luton Town
Performance_CS: Arsenal
Performance_CS%: Manchester City
Penalty_Kicks_PKatt: Fulham
Penalty_Kicks_PKA: Fulham
```

Solution:

```
best_teams_by_stat = {}
for col in df.columns[5:]:
    max_idx = df[col].idxmax()
    best_team = df.loc[max_idx, 'Squad']
    best_teams_by_stat[col] = best_team
```

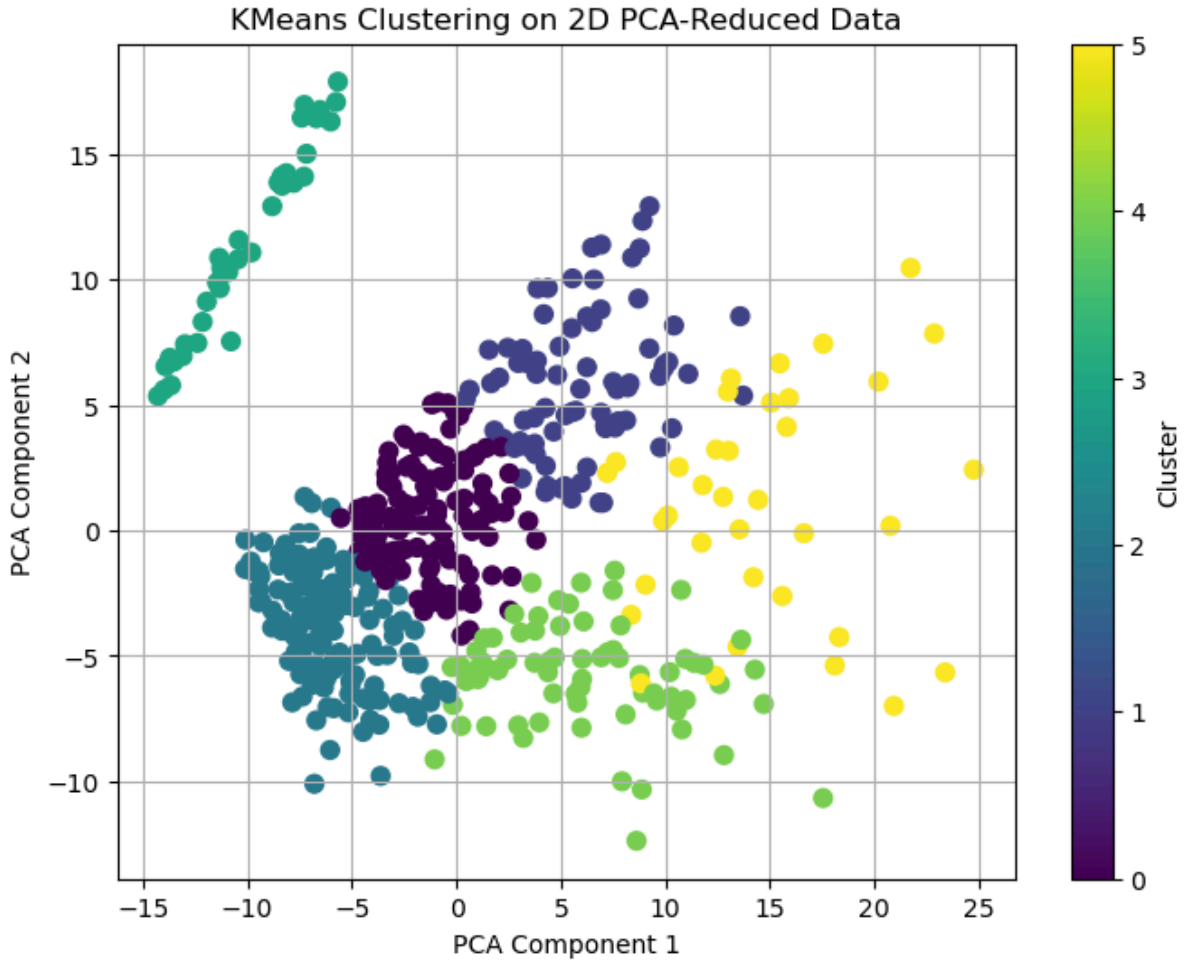
K-MEANS PHÂN LOẠI CÁC CẦU THỦ THÀNH CÁC NHÓM GIỐNG NHAU:

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

# Chọn các cột có số liệu liên quan để phân nhóm
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
data = df[numerical_columns].fillna(-999)  # Thay NaN bằng giá trị -999

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

if np.isnan(data_scaled).sum() == 0:
    kmeans = KMeans(n_clusters=9, random_state=42)
    df['Cluster'] = kmeans.fit_predict(data_scaled)

    with open("player_clusters.txt", "w") as f:
        for cluster in range(9):
            f.write(f"Cluster {cluster}:\n")
            cluster_players = df[df['Cluster'] == cluster]['Player'].tolist()
            for player in cluster_players:
                f.write(f"- {player}\n")
            f.write("\n")
else:
    print("Error: Data contains NaN values after scaling.")
```

nhận xét:

- Dù có chia làm bao nhiêu nhóm thì luôn có 1 nhóm gồm toàn các thủ môn.
- Việc chọn số lượng nhóm nên phụ thuộc vào mục đích ban đầu (chia theo vị trí, đánh giá vị trí dựa trên chỉ số,...)

SỬ DỤNG PCA, GIẢM CHIỀU DỮ LIỆU CÒN 2 CHIỀU VÀ PHÂN CỤM THÀNH 6 CLUSTERS:



Solution:

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
data = df[numerical_columns].fillna(-999)

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

kmeans = KMeans(n_clusters=6, random_state=42)
df['Cluster'] = kmeans.fit_predict(data_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=df['Cluster'], cmap='viridis', s=50)
plt.title('KMeans Clustering on 2D PCA-Reduced Data')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()
```

BIỂU ĐỒ RADAR CHART ĐỂ SO SÁNH CÁC CẦU THỦ:

Solution:

```
from math import pi
from sklearn.preprocessing import MinMaxScaler

def radar_chart(df, player1, player2, attributes):
    # Lấy dữ liệu của hai cầu thủ
    player1_data = df[df['Player'] ==
player1][attributes].values.flatten()
    player2_data = df[df['Player'] ==
player2][attributes].values.flatten()
    print(player1_data, player2_data)

    # Chuẩn hóa dữ liệu về khoảng 0-1 để dễ so sánh
    data = np.array([player1_data, player2_data])
    scaler = MinMaxScaler()
    data_normalized = scaler.fit_transform(data)

    # Dữ liệu chuẩn hóa của 2 cầu thủ
```

```
    player1_data = data_normalized[0]
    player2_data = data_normalized[1]

    # Tạo các nhãn cho các chỉ số
    labels = attributes
    num_vars = len(labels)

    # Tạo các góc cho biểu đồ radar
    angles = np.linspace(0, 2 * np.pi, num_vars,
endpoint=False).tolist()

    # Đóng hình tròn cho các góc
    player1_data = np.concatenate((player1_data, [player1_data[0]]))
    player2_data = np.concatenate((player2_data, [player2_data[0]]))
    angles += angles[:1]

    # Thiết lập biểu đồ
    fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))

    # Vẽ radar chart cho cầu thủ 1
    ax.fill(angles, player1_data, color='blue', alpha=0.15)
    ax.plot(angles, player1_data, color='blue', linewidth=2,
label=player1)

    # Vẽ radar chart cho cầu thủ 2
    ax.fill(angles, player2_data, color='red', alpha=0.15)
    ax.plot(angles, player2_data, color='red', linewidth=2,
label=player2)

    # Đặt các nhãn cho các trục
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(labels)

    # Tinh chỉnh hiển thị
    plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
    plt.title(f"Comparison between {player1} and {player2}")
    plt.show()

# Chọn cầu thủ và các chỉ số so sánh
player1 = "Mohamed Salah"
player2 = "Erling Haaland"
attributes = ["Expected_xG_x", "Performance_G-PK", "Playing_Time_Min"]
```

```
if player1 not in df['Player'].values:
    print(f"Player {player1} not found in the dataset.")
elif player2 not in df['Player'].values:
    print(f"Player {player2} not found in the dataset.")
else:
    # Vẽ biểu đồ radar so sánh hai cầu thủ
    radar_chart(df, player1, player2, attributes)
```