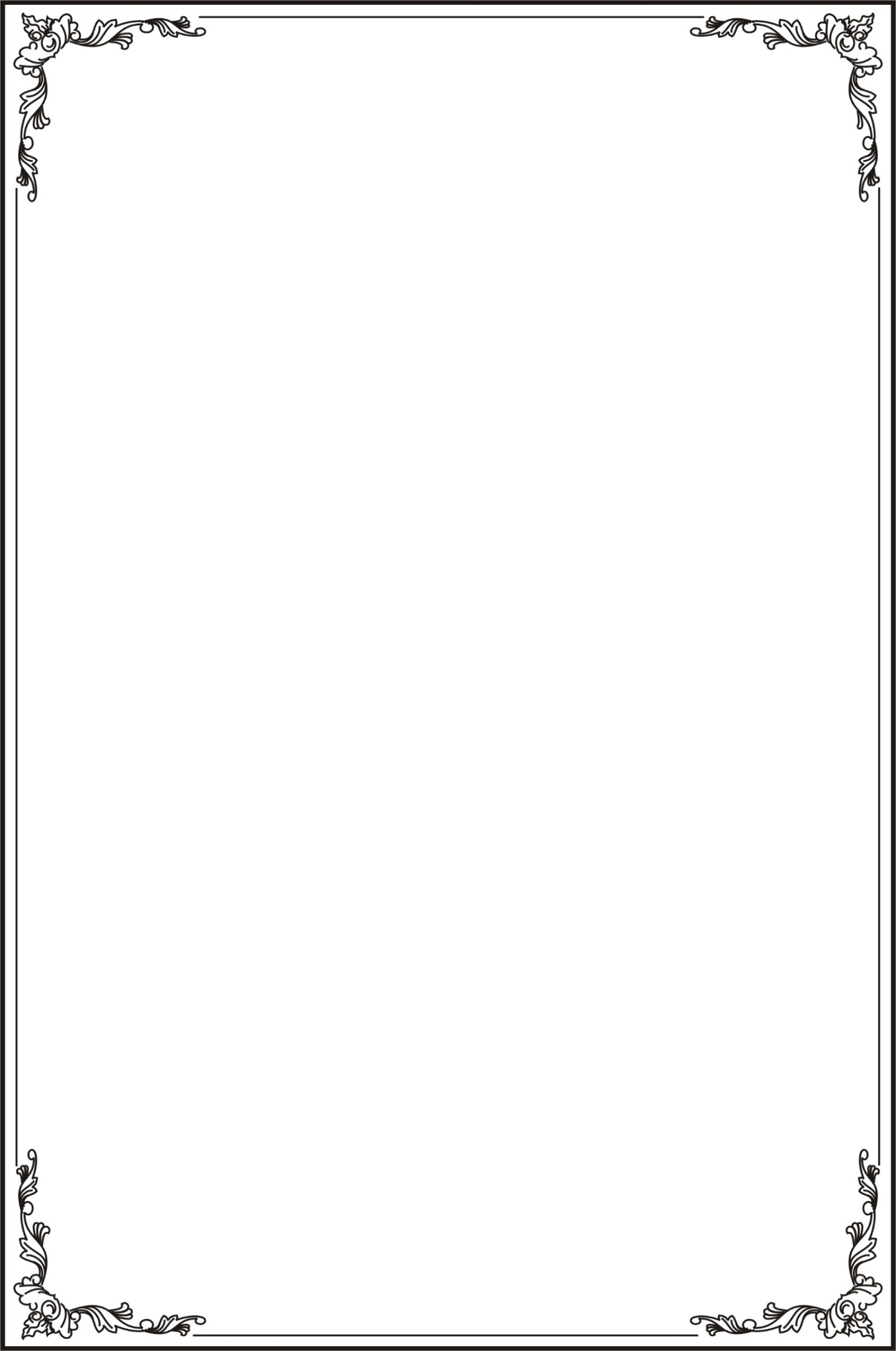
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT**

**KHOA CÔNG NGHỆ THÔNG TIN**



***Học phần:***

**BIG DATA**

***Đề tài: Đề Tài Số 1***

**Giảng viên hướng dẫn: Khuất Thị Ngọc Ánh**

**Lớp: 71DCTT21**

**Thành viên nhóm gồm: Vũ Nguyễn**

**Nguyễn Khương Duy**

**Phạm Giang Nam**

**Nguyễn Văn Sơn**

**Hà Nội, 2023**

**Bảng Phân Công Công Việc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Số Thứ Tự** | **Họ và Tên** | **Công Việc** | **Đóng góp** | **Đánh giá** |
| 1 | Vũ Nguyễn |  | 25% | Tốt |
| 2 | Phạm Giang Nam | Tìm hiểu tài liệu | 25% | Tốt |
| 3 | Nguyễn Khương Duy |  | 25% | Tốt |
| 4 | Nguyễn Văn Sơn |  | 25% | Tốt |

MỤC LỤC

[1.Bigdata 1](#_Toc147131114)

[1.1 Khái Niệm 1](#_Toc147131115)

[1.2 Đặc Trưng 1](#_Toc147131116)

[1.3 Phân Loại 3](#_Toc147131117)

[2. Spark 4](#_Toc147131118)

[2.1 Khái Niệm 4](#_Toc147131119)

[2.2 Các Thành Phần 4](#_Toc147131120)

[2.3 Ưu Nhược Điểm 6](#_Toc147131121)

[3. Sử dụng Spark RDD hoặc Spark SQL để trả lời một số câu hỏi 7](#_Toc147131122)

[3.1 Đếm số lượng cầu thủ 8](#_Toc147131123)

[3.2 Tính số lượng câu lạc bộ trong danh sách 8](#_Toc147131124)

[3.3 Tính độ tuổi trung bình của các cầu thủ? Cầu thủ có tuổi lớn nhất và nhỏ nhất là bao nhiêu? 8](#_Toc147131125)

[3.4 Tính độ tuổi trung bình của các cầu thủ? Cầu thủ có tuổi lớn nhất và nhỏ nhất là bao nhiêu? 8](#_Toc147131126)

[3.5 Liệt kê số lượng cầu thủ tương ứng với từng quốc gia 9](#_Toc147131127)

[3.6 Tìm cầu thủ người Brazil có giá trị cao nhất 9](#_Toc147131128)

[3.7 Tính số lượng cầu thủ Brazil trong các câu lạc bộ 9](#_Toc147131129)

[3.8 Hiển thị danh sách top 5 các cầu thủ ghi nhiều bàn thắng nhất 10](#_Toc147131130)

[3.9 Tạo một cột mới nếu: Age < 25 điền là Y, 25 <= Age <= 30 điền là T, 30 < Age điền là O 10](#_Toc147131131)

[3.10 Tạo một cột mới và điền thông tin là 2 ký tự đầu của cột Nationality 10](#_Toc147131132)

[4. Thực hành streaming kết quả 3 đội bóng có thứ hạng cao nhất theo năm 11](#_Toc147131133)

# 1.Bigdata

## 1.1 Khái Niệm

Big data là các tập dữ liệu rất lớn và phức tạp, rất khó để quản lý, lưu trữ và phân tích bằng các công cụ xử lý dữ liệu truyền thống. Điều quan trọng của big data là khả năng phân tích và tìm hiểu thông tin từ những tập dữ liệu này, vì chúng thường chứa nhiều thông tin tiềm ẩn và giá trị quan trọng.

Ví dụ như phương pháp cá nhân hóa nội dung đề xuất cho mỗi người trên các nền tảng Spotify, Netflix, Youtube, các nền tảng Thương mại điện tử như Shopee, Lazada,...

## 1.2 Đặc Trưng

Có rất nhiều định nghĩa khác nhau về dữ liệu lớn (Big Data), theo thời gian đặc tính của Big Data cũng được bổ sung nhiều chữ V hơn, từ định nghĩa truyền thống Big Data = 3V (Volume, Variety, Velocity), đến SAS định nghĩa Big Data = 5V (3V + Variability, Veracity), rồi khi dữ liệu bắt đầu trở thành “mỏ vàng”, thành nguồn doanh thu mới của mỗi doanh nghiệp thì Big Data = 6V (5V + Value). Trong rất nhiều các định nghĩa khác nhau thì có vẻ định nghĩa của Google bao phủ tất cả các ý “Big data refers to data that would typically be too expensive to store, manage, and analyze using traditional (relational and/or monolithic) database systems. Usually, such systems are cost-inefficient because of their inflexibility for storing unstructured data (such as images, text, and video), accommodating “high-velocity” (real-time) data, or scaling to support very large (petabyte-scale) data volumes”, còn hiểu một cách đơn giản thì Big Data là khái niệm đề cập đến việc thu thập, lưu trữ, xử lý “dữ liệu lớn” mà các hệ thống truyền thống không thể xử lý được. Vậy dữ liệu thế nào gọi là “lớn”? “Dữ liệu lớn” có 6 đặc tính với 6 chữ V ở trên: Volume, Variety, Velocity, Variability, Veracity, và Value.

A diagram of a big data

Description automatically generated

**Volume**: khối lượng dữ liệu được tạo ra, lưu trữ và xử lý. Bao nhiêu thì là lớn? Ít nhất nó nên được tính bằng hàng tram Terabyte, còn thông thường thì nó là Petabyte hoặc Exabyte, thậm chí Zettabyte. Một số ví dụ: Facebook tạo ra khoảng 500TB dữ liệu mỗi ngày, con số này ở Twitter là khoảng 8TB.

**Velocity:** tốc độ dữ liệu được tạo ra. Câu hỏi cũ, bao nhiêu thì là lớn? Câu trả lời thì rất đa dạng, vì nó phụ thuộc vào loại dữ liệu mà bạn đang xử lý, nhưng có thể lấy một vài ví dụ trực quan của các tên tuổi lớn như 90 triệu bức ảnh được upload lên Facebook mỗi ngày, con số cho Twitter là 500 triệu tweets được post, 0.4 triệu giờ video được upload lên Youtube hay 3.5 tỷ lượt tìm kiếm được thực hiện mỗi ngày trên Google.

**Variety:** tính đa dạng của dữ liệu. Cái này chắc dễ hiểu rồi, Big Data là không ngán dạng nào, từ dữ liệu có cấu trúc (structure) như các bảng nơi có hàng và cột trong cơ sở dữ liệu quan hệ RDBMS hay bảng tính excel; đến dữ liệu phi cấu trúc (unstructured) như văn bản (text), ảnh (pictures), video, audio, …; và thậm chí cả dữ liệu bán cấu trúc (semi-structure) như file json hay file xml.

**Variability:** tính biến thiên của dữ liệu phản ánh sự không nhất quán trong dữ liệu, đặc biệt là trong xử lý ngôn ngữ tự nhiên, cùng một từ vựng nhưng trong các ngữ cảnh khác nhau nó sẽ mang các ý nghĩa khác nhau. Tính biến thiên của dữ liệu cũng còn được thể hiện qua sự biến thiên về kích thước và tốc độ dữ liệu được sinh ra, do dữ liệu được thu thập từ nhiều nguồn khác nhau.

**Veracity:** mức độ tin cậy của dữ liệu. Đặc tính này đi ngược chiều với các đặc tính khác của Big Data, khi khối lượng dữ liệu ngày càng tăng, tính đa dạng của dữ liệu ngày càng phong phú và tính biến thiên của dữ liệu ngày càng lớn thì mức độ tin cậy của dữ liệu ngày càng giảm xuống.

**Value:** giá trị của dữ liệu. Sẽ không thể nhắc đến Big Data nếu không thể get value từ dữ liệu. Hàng loạt các công ty đã khai thác “mỏ vàng mới” dữ liệu và phát triển mạnh mẽ: Google, Facebook, Amazon, …

## 1.3 Phân Loại

Big data thường được phân loại dựa trên 3 yếu tố dữ liệu phổ biến: Dữ liệu có cấu trúc, dữ liệu phi cấu trúc và dữ liệu bán cấu trúc.

**Dữ liệu có cấu trúc**

Dữ liệu có cấu trúc được xem là dữ liệu đơn giản nhất để quản lý và tìm kiếm. Nó là những dữ liệu có thể truy cập, lưu trữ và xử lý ở định dạng cố định. Các thành phần của dữ liệu có cấu trúc được phân loại dễ dàng, cho phép các nhà thiết kế và quản trị viên cơ sở dữ liệu xác định các thuật toán đơn giản để tìm kiếm và phân tích.

**Dữ liệu phi cấu trúc**

Dữ liệu phi cấu trúc là bất kỳ tập hợp dữ liệu nào không được tổ chức hoặc xác định rõ ràng. Loại dữ liệu này hỗn loạn, khó xử lý, khó hiểu và đánh giá. Nó không có cấu trúc cố định và có thể thay đổi vào những thời điểm khác nhau. Dữ liệu phi cấu trúc bao gồm các nhận xét, tweet, lượt chia sẻ, bài đăng trên mạng xã hội, video trên YouTube mà người dùng xem,...

**Dữ liệu bán cấu trúc**

Dữ liệu bán cấu trúc là sự kết hợp giữa dữ liệu có cấu trúc và dữ liệu phi cấu trúc. Email là một ví dụ điển hình vì chúng bao gồm dữ liệu phi cấu trúc trong nội dung thư, cũng như nhiều thuộc tính tổ chức khác như người gửi, người nhận, chủ đề và ngày tháng. Các thiết bị sử dụng gắn thẻ địa lý, thời gian cũng có thể cung cấp dữ liệu có cấu trúc bên cạnh nội dung phi cấu trúc.

# 2. Spark

## 2.1 Khái Niệm

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được ( Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,…. Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json…) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

## 2.2 Các Thành Phần

Several blue squares with text

Description automatically generated

Gồm 4 Thành phần chính là:

**Spark SQL**: là một thành phần nằm trên Spark Core, giới thiệu một khái niệm trừu tượng hóa dữ liệu mới gọi là SchemaRDD, cung cấp hỗ trợ cho dữ liệu có cấu trúc và bán cấu trúc.

**Spark Streaming**: tận dụng khả năng lập lịch memory-base của Spark Core để thực hiện streaming analytics. Nó lấy dữ liệu theo mini-batches và thực hiện các phép biến đổi RDD (Bộ dữ liệu phân tán có khả năng phục hồi) trên các mini-batches dữ liệu đó.

**MLlib (Machine Learning Library):** là một framework machine learning phân tán trên Spark tận dụng khả năng tính toán tốc độ cao nhờ distributed memory-based của kiến ​​trúc Spark.

**GraphX:** ​​là một framework xử lý đồ thị phân tán. Nó cung cấp một API để thực hiện tính toán biểu đồ có thể mô hình hóa các biểu đồ do người dùng xác định bằng cách sử dụng API đã được tối ưu sẵn.

**Spark Core**: là engine thực thi chung làm nền tảng cho Spark. Tất cả các chức năng khác được xây dựng dựa trên base là Spark Core. Nó cung cấp khả năng tính toán trên bộ nhớ RAM và cả bộ dữ liệu tham chiếu trong các hệ thống external storage.

## 2.3 Ưu Nhược Điểm

**Ưu điểm:**

**Advanced Analytics**: Spark không chỉ hỗ trợ "Map" và "Reduce ", nó còn hỗ trợ Spark truy vấn SQL, Streaming data, Machine learning (ML) và các thuật toán xử lý đồ thị đóng vai trò như một bộ công cụ phân tích dữ liệu cực kì mạnh mẽ.

**Speed**: Spark giúp chạy một ứng dụng với tốc độ rất nhanh. So với Hadoop cluster, Spark Application nến chạy trên bộ nhớ nhanh hơn tới 100 lần và nhanh hơn 10 lần khi chạy trên đĩa. Điều này có được nhờ giảm số lượng các hoạt động đọc / ghi vào ổ đĩa.

**Supports multiple languages:** Spark cung cấp built-in APIs phổ biến từ Java, Scala đến Python, R. Do đó, có thể code Spark applications với nhiều lựa chọn về ngôn ngữ lập trình. Bên cạnh đó Spark còn cung cấp rất nhiều high-level operators cho việc truy vấn dữ liệu...

**Nhược điểm:**

Spark không có hệ thống Filesystem riêng, do đó, nó phụ thuộc vào một số nền tảng khác như Hadoop hoặc một nền tảng dựa trên đám mây (S3, Google Cloud Storage,...).

Apache Spark đòi hỏi rất nhiều RAM để chạy trong bộ nhớ, do đó chi phí của Spark khá cao.

Spark Streaming không thực sự real-time

Việc tối ưu hóa, tinh chỉnh để phù hợp với các bộ dữ liệu cụ thể cần có kinh nghiệm và vẫn cần thực hiện thủ công.

# 3. Sử dụng Spark RDD hoặc Spark SQL để trả lời một số câu hỏi

**Phân tích các cột dữ liệu**

1. **sofifa\_id**: ID duy nhất của cầu thủ trên trang web Sofifa.
2. **player\_url**: URL của trang thông tin cầu thủ trên Sofifa.
3. **short\_name**: Tên ngắn gọn của cầu thủ.
4. **long\_name**: Tên đầy đủ của cầu thủ.
5. **age**: Tuổi của cầu thủ.
6. **dob**: Ngày tháng năm sinh của cầu thủ.
7. **height\_cm**: Chiều cao của cầu thủ (đơn vị: cm).
8. **weight\_kg**: Cân nặng của cầu thủ (đơn vị: kg).
9. **nationality**: Quốc tịch của cầu thủ.
10. **club**: Câu lạc bộ mà cầu thủ hiện đang thi đấu cho.
11. **overall**: Điểm tổng quan (Overall Rating) của cầu thủ, thường được tính dựa trên nhiều yếu tố khác nhau liên quan đến kỹ thuật và hiệu suất của cầu thủ.
12. **potential**: Tiềm năng của cầu thủ, tức là điểm cao nhất mà cầu thủ có thể đạt được trong tương lai.
13. **value\_eur**: Giá trị của cầu thủ trong Euro.
14. **wage\_eur**: Lương hàng tuần của cầu thủ trong Euro.
15. **player\_positions**: Vị trí(s) chơi bóng mà cầu thủ có thể đảm nhận.
16. **preferred\_foot**: Chân thuận của cầu thủ (Left - trái, Right - phải).
17. **international\_reputation**: Uy tín quốc tế của cầu thủ.
18. **weak\_foot**: Chân yếu của cầu thủ.
19. **skill\_moves**: Số lần di chuyển kỹ thuật mà cầu thủ có thể thực hiện.
20. **work\_rate**: Mức độ làm việc của cầu thủ.
21. **body\_type**: Loại cơ thể của cầu thủ.
22. **real\_face**: Có sử dụng hình ảnh khuôn mặt thật của cầu thủ không (Yes - Có, No - Không).
23. **release\_clause\_eur**: Mức phí phá hợp đồng của cầu thủ trong Euro.
24. **player\_tags**: Các tag mô tả về cầu thủ.
25. **team\_position**: Vị trí chơi trong đội bóng của cầu thủ.
26. **team\_jersey\_number**: Số áo cầu thủ trong đội bóng.
27. **loaned\_from**: Thông tin về việc mượn cầu thủ từ một đội khác (nếu có).
28. **joined**: Ngày gia nhập câu lạc bộ hiện tại.
29. **contract\_valid\_until**: Ngày hết hạn hợp đồng hiện tại của cầu thủ với câu lạc bộ.
30. Cột 30-45: Điểm số chi tiết của cầu thủ trong các khả năng và yếu tố khác nhau liên quan đến bóng đá, chẳng hạn như tốc độ (pace), khả năng dứt điểm (shooting), khả năng chuyền bóng (passing), khả năng điều bóng (dribbling), khả năng phòng thủ (defending), thể lực (physic), và khả năng thủ môn (goalkeeping)

## 3.1 Đếm số lượng cầu thủ

num\_players = df.count()

print("3.1")

print("Số lượng cầu thủ:", num\_players)

## 3.2 Tính số lượng câu lạc bộ trong danh sách

num\_clubs = df.select("club").distinct().count()

print("3.2")

print("Số lượng câu lạc bộ:", num\_clubs)

## 3.3 Tính độ tuổi trung bình của các cầu thủ? Cầu thủ có tuổi lớn nhất và nhỏ nhất là bao nhiêu?

avg\_age = df.selectExpr("avg(age)").first()[0]

max\_age = df.selectExpr("max(age)").first()[0]

min\_age = df.selectExpr("min(age)").first()[0]

print("3.3")

print("Độ tuổi trung bình của các cầu thủ:", round(avg\_age, 2))

print("Cầu thủ có tuổi lớn nhất:", max\_age)

print("Cầu thủ có tuổi nhỏ nhất:", min\_age)

## 3.4 Tính độ tuổi trung bình của các cầu thủ? Cầu thủ có tuổi lớn nhất và nhỏ nhất là bao nhiêu?

avg\_height = df.selectExpr("avg(height\_cm)").first()[0]

max\_height = df.selectExpr("max(height\_cm)").first()[0]

min\_height = df.selectExpr("min(height\_cm)").first()[0]

print("3.4")

print("Chiều cao trung bình của các cầu thủ:",round(avg\_height, 2) ,"cm")

print("Cầu thủ có chiều cao lớn nhất:", max\_height,"cm")

print("Cầu thủ có chiều cao nhỏ nhất:", min\_height,"cm")

## 3.5 Liệt kê số lượng cầu thủ tương ứng với từng quốc gia

so\_cau\_thu = df.groupBy("nationality").count()

# Hiển thị kết quả

print("3.5")

so\_cau\_thu.show(so\_cau\_thu.count())

## 3.6 Tìm cầu thủ người Brazil có giá trị cao nhất

brazilian\_players = df.filter(df["nationality"] == "Brazil")

# Tìm giá trị `value\_eur` lớn nhất trong số cầu thủ Brazil

max\_value\_eur = brazilian\_players.select(max(col("value\_eur").cast("float"))).collect()[0][0]

# Lọc ra các cầu thủ Brazil có `value\_eur` bằng giá trị lớn nhất

highest\_valued\_brazilian\_players = brazilian\_players.filter(col("value\_eur").cast("float") == max\_value\_eur)

# Chọn chỉ các cột cần hiển thị

selected\_columns = ["sofifa\_id", "player\_url", "short\_name", "long\_name", "age", "dob", "height\_cm", "weight\_kg","value\_eur"]

result\_df = highest\_valued\_brazilian\_players.select(\*selected\_columns)

# In ra thông tin các cầu thủ Brazil có `value\_eur` cao nhất

print("3.6: Cầu Thủ Có Giá Trị Cao Nhất")

result\_df.show()

## 3.7 Tính số lượng cầu thủ Brazil trong các câu lạc bộ

brazilian\_players\_in\_clubs = df.filter(df["nationality"] == "Brazil").groupBy("club").count()

# In ra số lượng cầu thủ Brazil trong các câu lạc bộ

print("3.7 Số lượng cầu thủ Brazil trong các câu lạc bộ:")

brazilian\_players\_in\_clubs.show(brazilian\_players\_in\_clubs.count())

# Lọc ra câu lạc bộ có số lượng cầu thủ Brazil lớn nhất

max\_brazilian\_players\_count = brazilian\_players\_in\_clubs.select(max(col("count"))).collect()[0][0]

club\_with\_most\_brazilian\_players = brazilian\_players\_in\_clubs.filter(col("count") == max\_brazilian\_players\_count)

# In ra câu lạc bộ có số lượng cầu thủ Brazil lớn nhất

print(" Câu lạc bộ có số lượng cầu thủ Brazil lớn nhất:")

club\_with\_most\_brazilian\_players.show()

## 3.8 Hiển thị danh sách top 5 các cầu thủ ghi nhiều bàn thắng nhất

top\_scorers = df.orderBy(col("shooting").desc()).limit(5)

# Chọn chỉ các cột cần hiển thị

selected\_columns = ["sofifa\_id", "player\_url", "short\_name", "long\_name", "age", "dob", "height\_cm", "weight\_kg","shooting"]

# Hiển thị danh sách top 5 cầu thủ ghi nhiều bàn thắng nhất

print('3.8: danh sách top 5 cầu thủ ghi nhiều bàn thắng nhất:')

top\_scorers.select(\*selected\_columns).show()

## 3.9 Tạo một cột mới nếu: Age < 25 điền là Y, 25 <= Age <= 30 điền là T, 30 < Age điền là O

df = df.withColumn("age\_group", when(df["age"] < 25, "Y").when((df["age"] >= 25) & (df["age"] <= 30), "T").otherwise("O"))

# In ra cột mới 'age\_group'

print("3.9: Cột mới 'age\_group':")

# Lấy 20 dòng đầu tiên và chỉ hiển thị các trường thông tin được yêu cầu

top\_20\_players = df.select("sofifa\_id", "short\_name", "long\_name", "age","age\_group").limit(20)

# In ra thông tin của 20 cầu thủ

top\_20\_players.show()

## 3.10 Tạo một cột mới và điền thông tin là 2 ký tự đầu của cột Nationality

df = df.withColumn("nationality\_prefix", substring(df["nationality"], 1, 2))

# In ra cột mới 'nationality\_prefix'

# Chọn các trường thông tin bạn muốn hiển thị

selected\_columns = ["sofifa\_id", "player\_url", "short\_name", "long\_name", "age", "dob", "height\_cm", "weight\_kg","nationality", "nationality\_prefix"]

# Hiển thị thông tin của các cầu thủ chỉ với các trường bạn đã chọn

print("3.10: 2 ký tự đầu của trường 'nationality' ")

df.select(\*selected\_columns).show()

# 4. Thực hành streaming kết quả 3 đội bóng có thứ hạng cao nhất theo năm

**Phân tích dữ liệu**

- Bộ dữ liệu gồm 5 file: Premium\_2015.csv, Premium\_2016.csv, Premium\_2017.csv, Premium\_2018.csv, Premium\_2019.csv, Premium\_2020.csv

1. **Div**: Đây là viết tắt của "Division" (Bảng), nó chỉ ra giải đấu hoặc hạng đội bóng mà trận đấu thuộc về, ví dụ: E0 cho Premier League.
2. **Date**: Ngày diễn ra trận đấu.
3. **HomeTeam**: Đội chủ nhà trong trận đấu.
4. **AwayTeam**: Đội khách trong trận đấu.
5. **FTHG**: Số bàn thắng của đội chủ nhà (Full Time Home Goals).
6. **FTAG**: Số bàn thắng của đội khách (Full Time Away Goals).
7. **FTR**: Kết quả cuối cùng của trận đấu (Full Time Result),
8. **HTHG**: Số bàn thắng của đội chủ nhà vào hiệp 1 (Half Time Home Goals).
9. **HTAG**: Số bàn thắng của đội khách vào hiệp 1 (Half Time Away Goals).
10. **HTR**: Kết quả tại hiệp 1, H (Home win), A (Away win), D (Draw).
11. **Referee**: Tên trọng tài của trận đấu.
12. **HS**: Số lần sút của đội chủ nhà (Home Shots).
13. **AS**: Số lần sút của đội khách (Away Shots).
14. **HST**: Số lần sút trúng khung thành của đội chủ nhà (Home Shots on Target).
15. **AST**: Số lần sút trúng khung thành của đội khách (Away Shots on Target).
16. **HF**: Số lần phạm lỗi của đội chủ nhà (Home Fouls).
17. **AF**: Số lần phạm lỗi của đội khách (Away Fouls).
18. **HC**: Số quả góc của đội chủ nhà (Home Corners).
19. **AC**: Số quả góc của đội khách (Away Corners).
20. **HY**: Số thẻ vàng của đội chủ nhà (Home Yellow Cards).
21. **AY**: Số thẻ vàng của đội khách (Away Yellow Cards).
22. **HR**: Số thẻ đỏ của đội chủ nhà (Home Red Cards).
23. **AR**: Số thẻ đỏ của đội khách (Away Red Cards).
24. Từ cột 24 đến cuối là tỉ lệ cược của các nhà cái

import pyspark.sql.functions as F

from pyspark.sql import SparkSession

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

import os

# Khởi tạo SparkContext

sc = SparkContext("local[2]", "FileContent")

# Khởi tạo StreamingContext với batch interval là 20 giây

ssc = StreamingContext(sc, 20)

# Đường dẫn đến thư mục "input"

input\_directory = "input"

# Tạo một DStream từ thư mục "input"

input\_stream = ssc.textFileStream(input\_directory)

# Khởi tạo SparkSession

spark = SparkSession.builder.appName("PremierLeagueRanking").getOrCreate()

# Hàm để xác định thứ hạng của các đội bóng

def calculate\_rank(data\_frame):

    ranked\_data = data\_frame.groupBy("HomeTeam").agg(

        F.sum(F.when(data\_frame["FTR"] == "H", 3)

              .when(data\_frame["FTR"] == "D", 1)

              .otherwise(0)).alias("HomePoints")

    ).orderBy(F.desc("HomePoints"))

    top\_3\_teams = ranked\_data.limit(3)  # Giới hạn kết quả cho 3 đội hàng đầu

    return top\_3\_teams

# Hàm để in ra nội dung của các file trong thư mục

def print\_file\_contents(rdd):

    file\_names = os.listdir(input\_directory)

    for file\_name in file\_names:

        file\_path = os.path.join(input\_directory, file\_name)

        data = spark.read.csv(file\_path, header=True, inferSchema=True)

        print("File:", file\_name)

        top3 = calculate\_rank(data)

        top3.show()

# Áp dụng hàm print\_file\_contents cho mỗi RDD trong DStream

input\_stream.foreachRDD(print\_file\_contents)

# Khởi động Spark Streaming

ssc.start()

# Đợi cho quá trình tính toán kết thúc

ssc.awaitTermination()

Tài Liệu Tham Khảo

1. <https://topdev.vn/blog/big-data/>
2. <https://demanejar.github.io/posts/spark-streaming/>
3. <https://www.databricks.com/glossary/what-is-spark-streaming>