**Hướng dẫn về khung dữ liệu của PySpark: DataFrames là gì?**

DataFrames thường đề cập đến một cấu trúc dữ liệu, về bản chất là dạng bảng. Nó đại diện cho các hàng, mỗi hàng bao gồm một số quan sát. Các hàng có thể có nhiều định dạng dữ liệu (không đồng nhất), trong khi một cột có thể có dữ liệu cùng loại (đồng nhất). DataFrames thường chứa một số siêu dữ liệu ngoài dữ liệu; ví dụ, tên cột và hàng.



Chúng ta có thể nói rằng DataFrames không là gì, ngoài các cấu trúc dữ liệu 2 chiều, tương tự như bảng SQL hoặc bảng tính. Bây giờ, hãy tiếp tục với Hướng dẫn về khung dữ liệu PySpark này và hiểu lý do tại sao chính xác chúng ta cần Pyspark Dataframe.

## ****Tại sao chúng ta cần DataFrames?****

1. **Xử lý dữ liệu có cấu trúc và bán cấu trúc**

DataFrames được thiết kế để xử lý một **tập hợp lớn dữ liệu có cấu trúc cũng như bán cấu trúc** . Các quan sát trong Spark DataFrame được tổ chức dưới các cột được đặt tên, giúp Apache Spark hiểu sơ đồ của Dataframe. Điều này giúp Spark tối ưu hóa kế hoạch thực hiện trên các truy vấn này. Nó cũng có thể xử lý petabyte dữ liệu.

1. **Cắt lát và thái hạt lựu**

API DataFrames thường hỗ trợ các phương thức phức tạp để cắt và xử lý dữ liệu. Nó bao gồm các hoạt động như "chọn" các hàng, cột và ô theo tên hoặc theo số, lọc ra các hàng, v.v. Dữ liệu thống kê thường rất lộn xộn và chứa nhiều giá trị thiếu và không chính xác và vi phạm phạm vi. Vì vậy, một tính năng cực kỳ quan trọng của DataFrames là quản lý rõ ràng dữ liệu bị thiếu.

### **Nguồn dữ liệu**

DataFrames đã hỗ trợ cho một loạt các định dạng và nguồn dữ liệu, chúng ta sẽ xem xét vấn đề này sau trong hướng dẫn Pyspark DataFrames này. Họ có thể lấy dữ liệu từ nhiều nguồn khác nhau.

### **Hỗ trợ nhiều ngôn ngữ**

Nó có hỗ trợ API cho các ngôn ngữ khác nhau như Python, R, Scala, Java, giúp mọi người có nền tảng lập trình khác nhau dễ sử dụng hơn.

## ****Các tính năng của DataFrames****

* DataFrames được **phân phối** trong tự nhiên, làm cho nó có cấu trúc dữ liệu có khả năng chịu lỗi và có tính sẵn sàng cao.
* **Đánh giá lười biếng** là một chiến lược đánh giá giữ đánh giá biểu thức cho đến khi cần giá trị của nó. Nó tránh đánh giá lặp đi lặp lại. Đánh giá lười biếng trong Spark có nghĩa là việc thực thi sẽ không bắt đầu cho đến khi một hành động được kích hoạt. Trong Spark, hình ảnh đánh giá lười biếng xuất hiện khi biến đổi Spark xảy ra.
* DataFrames là **bất biến** trong tự nhiên. Bằng cách bất biến, ý tôi là nó là một đối tượng có trạng thái **không thể sửa đổi** sau khi nó được tạo. Nhưng chúng ta có thể biến đổi các giá trị của nó bằng cách áp dụng một phép biến đổi nhất định, như trong RDD.

## ****Nguồn dữ liệu của PySpark****



Dữ liệu có thể được tải thông qua **tệp CSV, JSON, XML**  hoặc tệp Parquet. Nó cũng có thể được tạo bằng **RDD** hiện có và thông qua bất kỳ cơ sở dữ liệu nào khác, như **Hive** hoặc **Cassandra** . Nó cũng có thể lấy dữ liệu từ HDFS hoặc hệ thống tệp cục bộ.

## ****Pyspark DataFrames Ví dụ: Bộ dữ liệu FIFA World Cup****

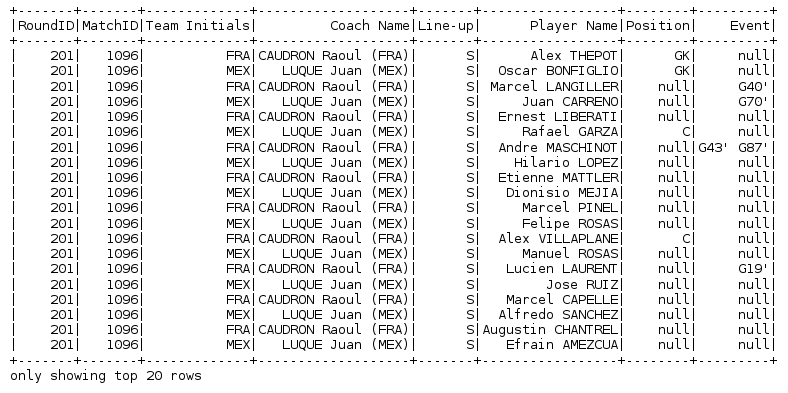
Ở đây chúng tôi đã lấy Bộ dữ liệu cầu thủ FIFA World Cup. Chúng tôi sẽ tải dữ liệu này ở định dạng CSV vào DataFrame và sau đó chúng tôi sẽ tìm hiểu về các biến đổi và hành động khác nhau có thể được thực hiện trên DataFrame này.

### **Đọc dữ liệu từ tệp CSV**

Hãy tải dữ liệu từ tệp CSV. Ở đây chúng ta sẽ sử dụng phương thức **spark.read.csv** để tải dữ liệu vào DataFrame , fifa\_df. Phương thức thực tế là **spark.read.format [csv / json]** .

fifa\_df = spark.read.csv("path-of-file/fifa\_players.csv", inferSchema = True, header = True)

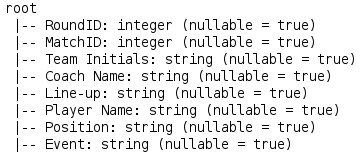
fifa\_df.show()



### **Lược đồ của DataFrame**

Để xem sơ đồ, tức là cấu trúc của DataFrame, chúng ta sẽ sử dụng phương thức **printSchema** . Điều này sẽ cung cấp cho chúng tôi các cột khác nhau trong DataFrame của chúng tôi, cùng với loại dữ liệu và các điều kiện không thể thực hiện được cho cột cụ thể đó.

Fifa\_df.printSchema()



### **Tên cột và Đếm (Hàng và Cột)**

Khi chúng tôi muốn xem tên và số lượng hàng và cột của một DataFrame cụ thể, chúng tôi sử dụng các phương pháp sau.

fifa\_df.columns //column Names

fifa\_df.count() //Row Count

len(fifa\_df.columns) //Columns Count