1. Spark Properties.

Thuộc tính Spark kiểm soát hầu hết các cài đặt ứng dụng và được cấu hình riêng cho từng ứng dụng. Các thuộc tính này có thể được đặt trực tiếp trên *SparkConf* và được chuyển đến *SparkContext* của bạn. *SparkConf* cho phép bạn cấu hình một số thuộc tính chung (Ví dụ: Master URL, Tên ứng dụng), cũng như các cặp (khóa - giá trị) thông qua phương thức *set()*.

Ví dụ: Khởi tạo một ứng dụng với 2 luồng.

```
val conf = new SparkConf()
             .setMaster("local[2]")
             .setAppName("CountingSheep")
val sc = new SparkContext(conf)
```

- Các thuộc tính thời gian được cấu hình như sau:

```
25ms (milliseconds)
5s (seconds)
10m or 10min (minutes)
3h (hours)
5d (days)
1y (years)
```

- Các thuộc tính chỉ định kích thước được cấu hình như sau:

```
1b (bytes)
1k or 1kb (kibibytes = 1024 bytes)
1m or 1mb (mebibytes = 1024 kibibytes)
1g or 1gb (gibibytes = 1024 mebibytes)
1t or 1tb (tebibytes = 1024 gibibytes)
1p or 1pb (pebibytes = 1024 tebibytes)
```

2. Dynamically Loading Spark Properties.

Dynamically Loading Spark Properties cho phép chúng ta quyết định cách khởi động ứng dụng. Ví dụ: Nếu bạn muốn chạy cùng 1 ứng dụng với các bản khác nhau hoặc số lượng bộ nhớ khác nhau Spark cho phép bạn tạo một *SparkConf* trống:

```
val sc = new SparkContext(new SparkConf())
```

Sau đó bạn có thể cấu hình các giá trị trong thời gian chạy:

```
./bin/spark-submit --name "My app" --master local[4] --conf spark.eventLog.enabled=false
  --conf "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" myApp.jar
```

Người dùng có thể sử dụng *spark-submit* để chỉ định bất kỳ thuộc tính nào bằng cách sử dụng *–conf/-c* sau đó chỉ định đối số của thuộc tính đó sau dấu "=". Chạy lệnh *–help* sẽ hiển thị tất cả các tùy chọn.

- Quyền ưu tiên.

  Dynamically Loading Spark Properties được ưu tiên theo hai cách sau:

  - Tải thuộc tính theo các cách khác nhau: Các thuộc tính được đặt trực tiếp trên SparkConf có độ ưu tiên cao nhất sau đó đến các flag được chuyển đến các spark-submit hoặc spark-shell và cuối cùng là các thuộc tính tùy chọn trong tệp "*spark-defaults.conf*".

  - Tải thuộc tính có tên được cập nhật: Khi phiên bản mới được cập nhật thì một số tên khóa cấu hình sẽ bị thay đổi, nhưng các khóa cũ này vẫn sẽ được chấp nhận nhưng chúng được ưu tiên thấp hơn so với những khóa mới.

3. Resilient Distributed Datasets:

Resilient Distributed Datasets (RDD) là một cấu trúc dữ liệu cơ bản của Spark. Nó là một tập hợp bất biến phân tán của một đối tượng. Mỗi dataset trong RDD được chia ra thành nhiều phần vùng logical. Có thể được tính toán trên các node khác nhau của một cụm máy chủ (cluster).

RDDs có thể chứa bất kỳ kiểu dữ liệu nào của Python, Java, hoặc đối tượng Scala, bao gồm các kiểu dữ liệu do người dùng định nghĩa. Thông thường, RDD chỉ cho phép đọc, phân mục tập hợp của các bản ghi. RDDs có thể được tạo ra qua điều khiển xác định trên dữ liệu trong bộ nhớ hoặc RDDs, RDD là một tập hợp có khả năng chịu lỗi mỗi thành phần có thể được tính toán song song.

Có hai cách để tạo RDDs:

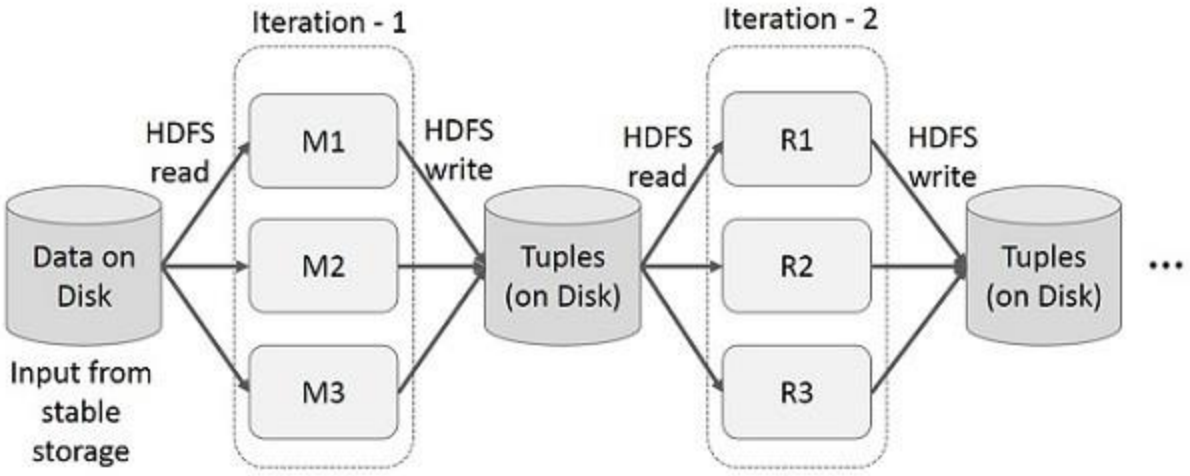
- Tạo từ một tập hợp dữ liệu có sẵn trong ngôn ngữ sử dụng như Java, Python, Scala.
- Lấy từ dataset hệ thống lưu trữ bên ngoài như HDFS, Hbase hoặc các cơ sở dữ liệu quan hệ.

3.1. Thực thi trên MapReduce:

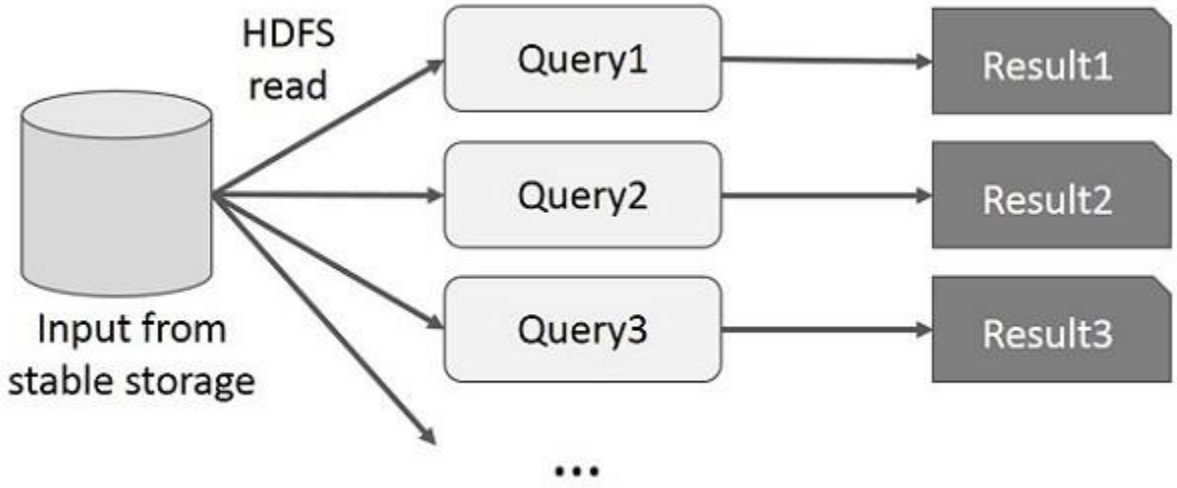
MapReduce được áp dụng rộng rãi để xử lý và tạo các bộ dữ liệu lớn với thuật toán xử lý phân tán song song trên một cụm. Nó cho phép người dùng viết các tính toán song song, sử dụng một tập hợp các toán tử cấp cao, mà không phải lo lắng về xử lý/phân phối công việc và khả năng chịu lỗi.

Cả hai ứng dụng Lặp (Iterative) và Tương tác (Interactive) đều yêu cầu chia sẻ truy cập và xử lý dữ liệu nhanh hơn trên các công việc song song. Chia sẻ dữ liệu chậm trong MapReduce do sao chép tuần tự và tốc độ I/O của ổ đĩa. Về hệ thống lưu trữ, hầu hết các ứng dụng Hadoop, cần dành hơn 90% thời gian để thực hiện các thao tác đọc-ghi HDFS.

- Iterative Operation trên MapReduce:



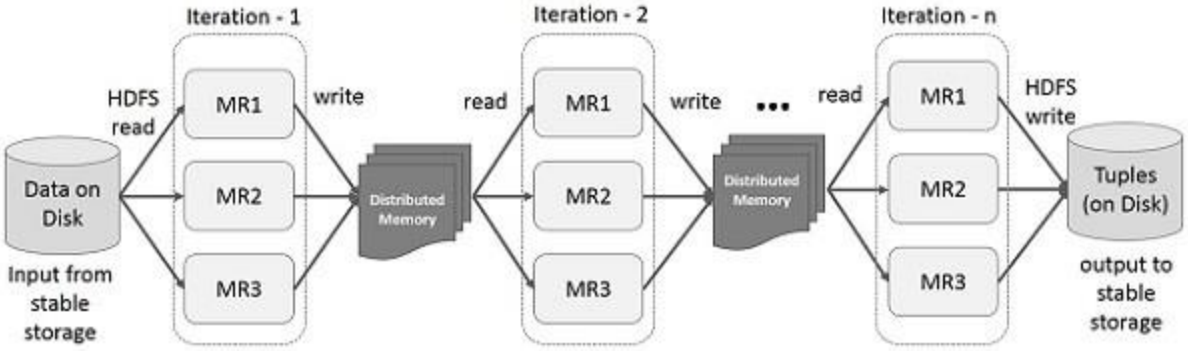
- Iteractive Operations trên MapReduce:



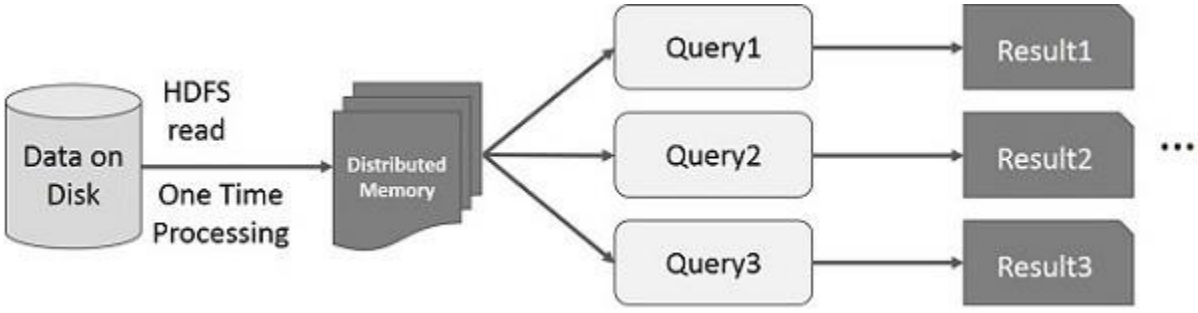
3.2. Thực thi trên Spark RDD:

Để khắc phục được vấn đề về MapRedure, các nhà nghiên cứu đã phát triển một framework chuyên biệt gọi là Apache Spark. Ý tưởng chính của Spark là Resilient Distributed Datasets (RDD); nó hỗ trợ tính toán xử lý trong bộ nhớ. Điều này có nghĩa, nó lưu trữ trạng thái của bộ nhớ dưới dạng một đối tượng trên các công việc và đối tượng có thể chia sẻ giữa các công việc đó. Việc xử lý dữ liệu trong bộ nhớ nhanh hơn 10 đến 100 lần so với network và disk.

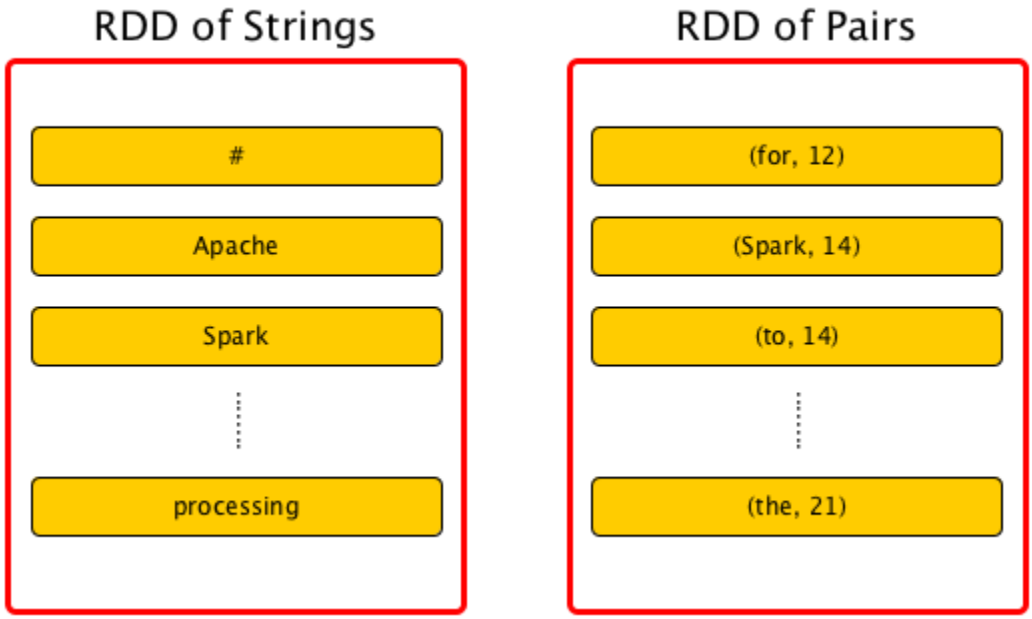
- Iterative Operation trên Spark RDD:



- Interactive Operations trên Spark RDD:

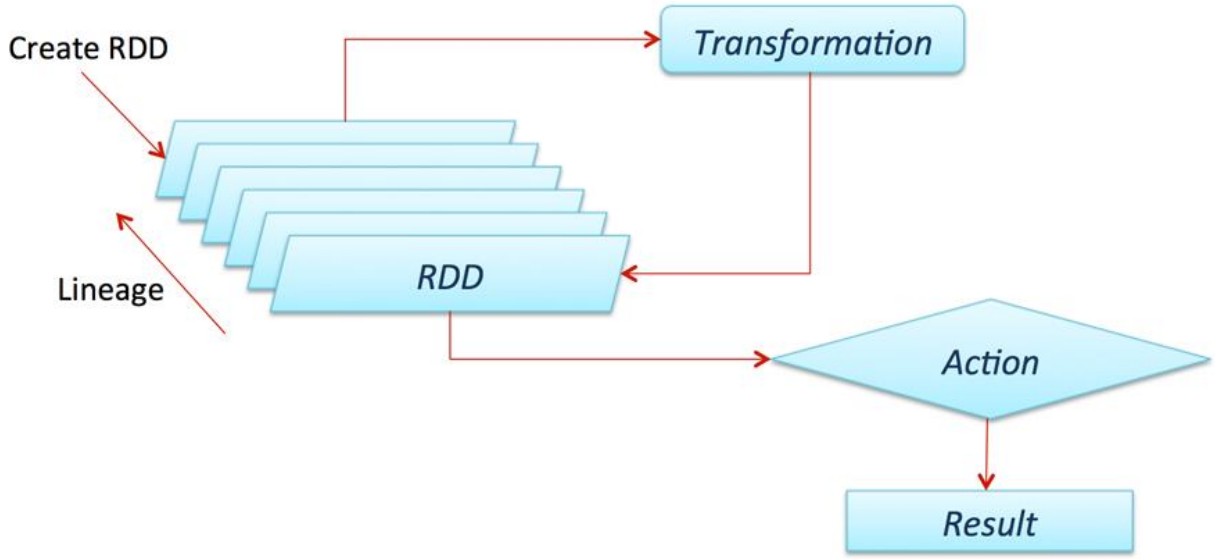


3.3. Các loại RDD:



3.4. Các transformation và action với RDD:

RDD cung cấp các transformation và action hoạt động giống như DataFrame lẫn DataSets. Transformation xử lý các thao tác lazily và Action xử lý thao tác cần xử lý tức thời.



- Một số transformation:

Transformation	Meaning
map(*func*)	Return a new distributed dataset formed by passing each element of the source through a function *func*.
filter(*func*)	Return a new dataset formed by selecting those elements of the source on which *func* returns true.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item).
mapPartitions(*func*)	Similar to map, but runs separately on each partition (block) of the RDD, so *func* must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
mapPartitionsWithIndex(*func*)	Similar to mapPartitions, but also provides *func* with an integer value representing the index of the partition, so *func* must be of type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(*withReplacement*, *fraction*, *seed*)	Sample a fraction *fraction* of the data, with or without replacement, using a given random number generator seed.
union(*otherDataset*)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(*otherDataset*)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
distinct([*numPartitions*]))	Return a new dataset that contains the distinct elements of the source dataset.

- Một số action:

Action	Meaning
reduce(*func*)	Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to take(1)).
take(*n*)	Return an array with the first *n* elements of the dataset.
takeSample(*withReplacement*, *num*, [*seed*])	Return an array with a random sample of *num* elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
takeOrdered(*n*, *[ordering]*)	Return the first *n* elements of the RDD using either their natural order or a custom comparator.
saveAsTextFile(*path*)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call toString on each element to convert it to a line of text in the file.
saveAsSequenceFile(*path*) (Java and Scala)	Write the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem, HDFS or any other Hadoop-supported file system. This is available on RDDs of key-value pairs that implement Hadoop's Writable interface. In Scala, it is also available on types that are implicitly convertible to Writable (Spark includes conversions for basic types like Int, Double, String, etc).

4. RDD Persistence

Thường thì mỗi transformation sẽ thực hiện chạy lại mỗi khi bạn chạy actions. Tuy nhiên, bạn có thể persist 1 RDD trong bộ nhớ sử dụng phương thức persist (hoặc caching). Điều này rất tiện lợi cho việc tính toán ngay tại bộ nhớ trong và tái sử dụng chúng cho actions khác trên tập dữ liệu, nó làm cho việc tính toán thực hiện nhanh hơn và khả năng chịu lỗi tốt . Cụ thể, lần đầu tiên tính toán với 1 actions, nó sẽ lưu trữ tại bộ nhớ trong trên các node. Nếu 1 phần nào đó của RDD bị mất, nó sẽ tự động tính toán lại sử dụng transformation được tạo ban đầu.

Mỗi RDD được lưu trữ với các mức khác nhau. Ví dụ, bạn có thể persist dữ liệu trên ổ cứng, bộ nhớ trong, hoặc qua việc sao lặp qua các node. Các mức lưu trữ tùy thuộc truyền đối tượng *org.apache.spark.storage.StorageLevel* vào hàm *persist()*. Phương thức *cache()* là cách viết tắt cho phương thức persist với mức lưu trữ mặc định, cụ thể là *StorageLevel.MEMORY_ONLY*.

Storage Level	Meaning
MEMORY_ONLY	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they're needed. This is the default level.
MEMORY_AND_DISK	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.
MEMORY_ONLY_SER (Java and Scala)	Store RDD as *serialized* Java objects (one byte array per partition). This is generally more space-efficient than deserialized objects, especially when using a fast serializer, but more CPU-intensive to read.
MEMORY_AND_DISK_SER (Java and Scala)	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store the RDD partitions only on disk.
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc.	Same as the levels above, but replicate each partition on two cluster nodes.
OFF_HEAP (experimental)	Similar to MEMORY_ONLY_SER, but store the data in off-heap memory. This requires off-heap memory to be enabled.

5. Spark DataFrame:

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.

DESCRIPTION	COLUMN ONE	COLUMN TWO	COLUMN THREE	COLUMN FOUR
First Feature				
Second Feature				
Third Feature				
Fourth Feature				
Fifth Feature				

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.

5.1. Đọc dữ liệu từ tệp CSV.

```
fifa_df = spark.read.csv("path-of-file/fifa_players.csv", inferSchema = True, header = True)

fifa_df.show()
```

```
+-------+-------+-------------+--------------------+-------+-----------------+--------+---------+
|RoundID|MatchID|Team Initials|         Coach Name|Line-up|      Player Name|Position|    Event|
+-------+-------+-------------+--------------------+-------+-----------------+--------+---------+
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|      Alex THEPOT|      GK|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|  Oscar BONFIGLIO|      GK|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S| Marcel LANGILLER|    null|     G40'|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Juan CARRENO|    null|     G70'|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Ernest LIBERATI|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Rafael GARZA|       C|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Andre MASCHINOT|    null|G43' G87'|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|    Hilario LOPEZ|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Etienne MATTLER|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|   Dionisio MEJIA|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|     Marcel PINEL|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Felipe ROSAS|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Alex VILLAPLANE|       C|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Manuel ROSAS|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|   Lucien LAURENT|    null|     G19'|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|        Jose RUIZ|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|   Marcel CAPELLE|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|  Alfredo SANCHEZ|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|Augustin CHANTREL|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|   Efrain AMEZCUA|    null|     null|
+-------+-------+-------------+--------------------+-------+-----------------+--------+---------+
only showing top 20 rows
```

5.2. Lược đồ của DataFrame:

```
fifa_df.printSchema()
```

```
root
 |-- RoundID: integer (nullable = true)
 |-- MatchID: integer (nullable = true)
 |-- Team Initials: string (nullable = true)
 |-- Coach Name: string (nullable = true)
 |-- Line-up: string (nullable = true)
 |-- Player Name: string (nullable = true)
 |-- Position: string (nullable = true)
 |-- Event: string (nullable = true)
```

5.3. Tên cột và Đếm (Hàng và Cột):

```
fifa_df.columns //Column Names

fifa_df.count() // Row Count

len(fifa_df.columns) //Column Count
```

```
Out[5]: ['RoundID',
         'MatchID',
         'Team Initials',
         'Coach Name',
         'Line-up',
         'Player Name',
         'Position',
         'Event']
```

```
37784

8
```

5.4. Mô tả một cột đặc biệt:

Nếu chúng ta muốn xem tóm tắt về bất kỳ cột cụ thể nào của DataFrame, chúng tôi sử dụng describe phương thức này. Phương pháp này cung cấp cho chúng tôi bản tóm tắt thống kê của cột đã cho, nếu không được chỉ định, nó cung cấp tóm tắt thống kê của DataFrame.

```
fifa_df.describe('Coach Name').show()
fifa_df.describe('Position').show()
```

```
+-------+------------------+ +-------+--------+
|summary|         Coach Name| |summary|Position|
+-------+------------------+ +-------+--------+
|  count|             37784| |  count|    4143|
|   mean|              null| |   mean|    null|
| stddev|              null| | stddev|    null|
|    min|ACOSTA Nelson (URU)| |    min|       C|
|    max|         ZICO (BRA)| |    max|     GKC|
+-------+------------------+ +-------+--------+
```

5.5. Lọc dữ liệu:

```
fifa_df.filter(fifa_df.MatchID=='1096').show()

fifa_df.filter(fifa_df.MatchID=='1096').count()  //to get the count
```

```
+-------+-------+-------------+--------------------+-------+-----------------+--------+---------+
|RoundID|MatchID|Team Initials|         Coach Name|Line-up|      Player Name|Position|    Event|
+-------+-------+-------------+--------------------+-------+-----------------+--------+---------+
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|      Alex THEPOT|      GK|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|  Oscar BONFIGLIO|      GK|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S| Marcel LANGILLER|    null|      G40'|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Juan CARRENO|    null|      G70'|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Ernest LIBERATI|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Rafael GARZA|       C|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Andre MASCHINOT|    null|G43' G87'|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|    Hilario LOPEZ|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S| Etienne MATTLER|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|  Dionisio MEJIA|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|     Marcel PINEL|    null|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Felipe ROSAS|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S| Alex VILLAPLANE|       C|     null|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|     Manuel ROSAS|    null|     null|
|    201|   1096|          FRA|CAUDRON Raoul (FRA)|      S|  Lucien LAURENT|    null|      G19'|
|    201|   1096|          MEX|   LUQUE Juan (MEX)|      S|        Jose RUIZ|    null|     null|
```

5.7. Lọc dữ liệu (Nhiều tham số):

```
fifa_df.filter((fifa_df.Position=='C') && (fifa_df.Event=="G40'")).show()
```

```
+-------+-------+-------------+--------------------+-------+---------------+--------+-----+
|RoundID|MatchID|Team Initials|         Coach Name|Line-up|    Player Name|Position|Event|
+-------+-------+-------------+--------------------+-------+---------------+--------+-----+
|    201|   1089|          PAR|DURAND LAGUNA Jos...|      S|Luis VARGAS PENA|       C| G40'|
|    429|   1175|          HUN|  DIETZ Karoly (HUN)|      S|  Gyorgy SAROSI|       C| G40'|
+-------+-------+-------------+--------------------+-------+---------------+--------+-----+
```

Nguồn tham khảo:

- https://helpex.vn/article/rdd-trong-spark-la-gi-va-tai-sao-chung-ta-can-no-5c6afe5bae03f628d053a84c
- https://dataartblog.wordpress.com/2015/02/26/huong-dan-chuong-trinh-spark-part-2-rdds/
- https://spark.apache.org/docs/2.3.0/sql-programming-guide.html
- https://www.educative.io/edpresso/what-are-dynamically-loading-properties-in-spark
- https://laptrinh.vn/books/apache-spark/page/apache-spark-rdd