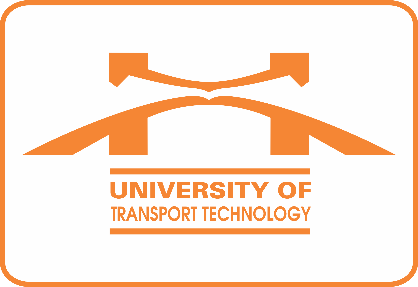
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÀI TẬP NHÓM**

**BÁO CÁO TỔNG KẾT HỌC PHẦN**

**MÔN HỌC: BIG DATA**

**Ngành: Công nghệ thông tin**

**Giảng viên hướng dẫn:** Đỗ Bảo Sơn

**Lớp:** 73DCTT23

**Sinh viên thực hiện:** 1. Trương Ngọc Duy

2. Nguyễn Huyền Nhung

3. Bùi Thu Phương

4. Đỗ Thị Phương

5. Nguyễn Như Xuân

**HÀ NỘI, tháng 4 / 2025**

# Mục lục

[Mục lục i](#_Toc196206305)

[Bảng phân công công việc ii](#_Toc196206306)

[Danh mục hình ảnh iii](#_Toc196206307)

[Danh mục từ viết tắt iv](#_Toc196206308)

[Chương 1: Tổng quan về big data và nền tảng Apache Spark 1](#_Toc196206309)

[1.1. Big data 1](#_Toc196206310)

[1.1.1. Khái niệm về big data 1](#_Toc196206311)

[1.1.2. Đặc trưng của big data 1](#_Toc196206312)

[1.1.3. Sự khác biệt giữa big data và dữ liệu truyền thống 2](#_Toc196206313)

[1.2. Apache Spark 3](#_Toc196206314)

[1.2.1. Giới thiệu về Apache Spark 3](#_Toc196206315)

[1.2.2. RDD 4](#_Toc196206316)

[1.2.3. Spark SQL 6](#_Toc196206317)

[1.2.4. Spark Streaming 7](#_Toc196206318)

[Chương 2: Câu hỏi đề tài số 3 10](#_Toc196206319)

[2.1. Câu hỏi 1 10](#_Toc196206320)

[2.2. Câu hỏi 2 11](#_Toc196206321)

[2.3. Câu hỏi 3 16](#_Toc196206322)

[2.4. Câu hỏi 4 23](#_Toc196206323)

[Tài liệu tham khảo 26](#_Toc196206324)

# Bảng phân công công việc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Mã sinh viên** | **Công việc** | **Đánh giá** |
| 1 | Trương Ngọc Duy | 73DCTT22150 | - Làm câu hỏi 3  - Tìm hiểu 1.2.3  - Soạn báo cáo Word | 20% |
| 2 | Nguyễn Huyền Nhung | 73DCTT22446 | - Làm câu hỏi 1  - Tìm hiểu 1.2.2 | 20% |
| 3 | Bùi Thu Phương | 73DCTT22404 | - Làm câu hỏi 4  - Tìm hiểu 1.1 | 20% |
| 4 | Đỗ Thị Phương | 73DCTT23303 | - Làm câu hỏi 4  - Tìm hiểu 1.2.4 | 20% |
| 5 | Nguyễn Như Xuân | 73DCTT22304 | - Làm câu hỏi 2  - Tìm hiểu 1.2.1 | 20% |

# Danh mục hình ảnh

[Hình 2.1. Các thành phần của Spark 13](#_Toc196212183)

[Hình 2.2. Kết quả 3.1 bằng Spark SQL 17](#_Toc196212184)

[Hình 2.3. Kết quả 3.2 bằng Spark SQL 17](#_Toc196212185)

[Hình 2.4. Kết quả 3.3 bằng Spark SQL 18](#_Toc196212186)

[Hình 2.5. Kết quả 3.4 bằng Spark SQL 18](#_Toc196212187)

[Hình 2.6. Kết quả 3.5 bằng Spark SQL 19](#_Toc196212188)

[Hình 2.7. Kết quả 3.6 bằng Spark SQL 19](#_Toc196212189)

[Hình 2.8. Kết quả 3.7 bằng Spark SQL 20](#_Toc196212190)

[Hình 2.9. Kết quả 3.8 bằng Spark SQL 21](#_Toc196212191)

[Hình 2.10. Kết quả 3.9 bằng Spark SQL 22](#_Toc196212192)

[Hình 2.11. Kết quả 3.1 bằng Spark SQL 22](#_Toc196212193)

[Hình 2.12. Kết quả câu 4 25](#_Toc196212194)

# Danh mục từ viết tắt

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Từ đầy đủ** | **Ý nghĩa** |
| RDD | Resilient distributed datasets | Bộ dữ liệu phân tán có khả năng phục hồi |
| SQL | Stucture query language | Ngôn ngữ truy vấn mang tính cấu trúc |
| API | Application programming interface | Giao diện chương trình ứng dụng |

# Tổng quan về big data và Apache Spark

## 1.1. Big data

### 1.1.1. Khái niệm về big data

Big data là những nguồn thông tin có đặc điểm chung là có khối lượng dữ liệu lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau, do đó muốn khai thác được đòi hỏi phải có hình thức xử lý mới để đưa ra quyết định, khám phá và tối ưu hoá quy trình.

### 1.1.2. Đặc trưng của big data

Big data có 5 đặc trưng cơ bản như sau (mô hình 5V):

(1) Khối lượng dữ liệu lớn (Volume):

Đây là đặc điểm tiêu biểu nhất của dữ liệu lớn, khối lượng dữ liệu rất lớn. Kích cỡ của Big data đang từng ngày tăng lên, và tính đến năm 2012 thì nó có thể nằm trong khoảng vài chục terabyte cho đến nhiều petabyte (1 petabyte = 1024 terabyte) chỉ cho một tập hợp dữ liệu. Dữ liệu truyền thống có thể lưu trữ trên các thiết bị đĩa mềm, đĩa cứng. Nhưng với dữ liệu lớn chúng ta sẽ sử dụng công nghệ “đám mây” mới đáp ứng khả năng lưu trữ được dữ liệu lớn.

(2) Tốc độ (Velocity):

Tốc độ có thể hiểu theo 2 khía cạnh: (a) Khối lượng dữ liệu gia tăng rất nhanh (mỗi giây có tới 72.9 triệu các yêu cầu truy cập tìm kiếm trên web bán hàng của Amazon); (b) Xử lý dữ liệu nhanh ở mức thời gian thực (real-time), có nghĩa dữ liệu được xử lý ngay tức thời ngay sau khi chúng phát sinh (tính đến bằng mili giây). Các ứng dụng phổ biến trên lĩnh vực Internet, Tài chính, Ngân hàng, Hàng không, Quân sự, Y tế – Sức khỏe như hiện nay phần lớn dữ liệu lớn được xử lý real-time. Công nghệ xử lý dữ liệu lớn ngày nay đã cho phép chúng ta xử lý tức thì trước khi chúng được lưu trữ vào cơ sở dữ liệu.

(3) Đa dạng (Variety):

Đối với dữ liệu truyền thống chúng ta hay nói đến dữ liệu có cấu trúc, thì ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc (tài liệu, blog, hình ảnh, vi deo, bài hát, dữ liệu từ thiết bị cảm biến vật lý, thiết bị chăm sóc sức khỏe…). Big data cho phép liên kết và phân tích nhiều dạng dữ liệu khác nhau. Ví dụ, với các bình luận của một nhóm người dùng nào đó trên Facebook với thông tin video được chia sẻ từ Youtube và Twitter.

(4) Độ tin cậy/chính xác (Veracity):

Một trong những tính chất phức tạp nhất của Dữ liệu lớn là độ tin cậy/chính xác của dữ liệu. Với xu hướng phương tiện truyền thông xã hội (Social Media) và mạng xã hội (Social Network) ngày nay và sự gia tăng mạnh mẽ tính tương tác và chia sẻ của người dùng Mobile làm cho bức tranh xác định về độ tin cậy & chính xác của dữ liệu ngày một khó khăn hơn. Bài toán phân tích và loại bỏ dữ liệu thiếu chính xác và nhiễu đang là tính chất quan trọng của Big data.

(5) Giá trị (Value):

Giá trị là đặc điểm quan trọng nhất của dữ liệu lớn, vì khi bắt đầu triển khai xây dựng dữ liệu lớn thì việc đầu tiên chúng ta cần phải làm đó là xác định được giá trị của thông tin mang lại như thế nào, khi đó chúng ta mới có quyết định có nên triển khai dữ liệu lớn hay không. Nếu chúng ta có dữ liệu lớn mà chỉ nhận được 1% lợi ích từ nó, thì không nên đầu tư phát triển dữ liệu lớn. Kết quả dự báo chính xác thể hiện rõ nét nhất về giá trị của dữ liệu lớn mang lại. Ví dụ, từ khối dữ liệu phát sinh trong quá trình khám, chữa bệnh sẽ giúp dự báo về sức khỏe được chính xác hơn, sẽ giảm được chi phí điều trị và các chi phí liên quan đến y tế.

### 1.1.3. Sự khác biệt giữa big data và dữ liệu truyền thống

Big data khác với dữ liệu truyền thống ở 4 điểm cơ bản: dữ liệu đa dạng hơn, lưu trữ dữ liệu lớn hơn, truy vấn dữ liệu nhanh hơn, độ chính xác cao hơn.

(1) Dữ liệu đa dạng hơn: Khi khai thác dữ liệu truyền thống (dữ liệu có cấu trúc), chúng ta thường phải trả lời các câu hỏi: Dữ liệu lấy ra kiểu gì? định dạng dữ liệu như thế nào? Đối với dữ liệu lớn, không phải trả lời các câu hỏi trên. Hay nói khác, khi khai thác, phân tích dữ liệu lớn chúng ta không cần quan tâm đến kiểu dữ liệu và định dạng của chúng; điều quan tâm là giá trị mà dữ liệu mang lại có đáp ứng được cho công việc hiện tại và tương lai hay không.

(2) Lưu trữ dữ liệu lớn hơn: Lưu trữ dữ liệu truyền thống vô cùng phức tạp và luôn đặt ra câu hỏi lưu như thế nào? dung lượng kho lưu trữ bao nhiêu là đủ? gắn kèm với câu hỏi đó là chi phí đầu tư tương ứng. Công nghệ lưu trữ dữ liệu lớn hiện nay đã phần nào có thể giải quyết được vấn đề trên nhờ những công nghệ lưu trữ đám mây, phân phối lưu trữ dữ liệu phân tán và có thể kết hợp các dữ liệu phân tán lại với nhau một cách chính xác và xử lý nhanh trong thời gian thực.

(3) Truy vấn dữ liệu nhanh hơn: Dữ liệu lớn được cập nhật liên tục, trong khi đó kho dữ liệu truyền thống thì lâu mới được cập nhật và trong tình trạng không theo dõi thường xuyên gây ra tình trạng lỗi cấu trúc truy vấn dẫn đến không tìm kiếm được thông tin đáp ứng theo yêu cầu.

(4) Độ chính xác cao hơn: Dữ liệu lớn khi đưa vào sử dụng thường được kiểm định lại dữ liệu với những điều kiện chặt chẽ, số lượng thông tin được kiểm tra thông thường rất lớn, và đảm bảo về nguồn lấy dữ liệu không có sự tác động của con người vào thay đổi số liệu thu thập.

## 1.2. Apache Spark

### 1.2.1. Giới thiệu về Apache Spark

Apache Spark là một nền tảng điện toán cụm được thiết kế để phân tích dữ liệu nhanh chóng.

Về mặt tốc độ, Spark mở rộng mô hình MapReduce phổ biến để hỗ trợ hiệu quả nhiều loại tính toán hơn, bao gồm truy vấn tương tác và xử lý luồng. Tốc độ rất quan trọng trong việc xử lý các tập dữ liệu lớn, vì nó có nghĩa là sự khác biệt giữa việc khám phá dữ liệu theo cách tương tác và chờ đợi nhiều thời gian. Một trong những tính năng chính mà Spark cung cấp để tăng tốc là khả năng chạy các phép tính trong bộ nhớ, nhưng hệ thống cũng hiệu quả hơn MapReduce đối với các ứng dụng phức tạp chạy trên đĩa.

Về tổng quát, Spark được thiết kế để xử lý nhiều khối lượng công việc trước đây yêu cầu các hệ thống phân tán riêng biệt, bao gồm các ứng dụng hàng loạt, thuật toán lặp, truy vấn tương tác và phát trực tuyến. Bằng cách hỗ trợ các khối lượng công việc này trong cùng một công cụ, Spark giúp dễ dàng và không tốn kém để kết hợp các loại xử lý khác nhau, thường cần thiết trong các quy trình phân tán dữ liệu sản xuất. Ngoài ra, nó làm giảm gánh nặng quản lý khi duy trì các công cụ riêng biệt.

Spark được thiết kế để có khả năng truy cập cao, cung cấp các API đơn giản trong Python, Java, Scala và SQL, cùng với các thư viện tích hợp phong phú. Nó cũng tích hợp chặt chẽ với các công cụ Big Data khác. Đặc biệt, Spark có thể chạy trong các cụm Hadoop và truy cập bất kỳ nguồn dữ liệu Hadoop nào, bao gồm cả Casandra.

### 1.2.2. RDD

a. Khái niệm về RDD

Trong Spark, một RDD đơn giản chỉ là một tập hợp các đối tượng phân tán không thay đổi. Mỗi RDD được chia thành nhiều phân vùng, có thể được tính toán trên các nút khác nhau của cụm. RDD có thể chứa bất kỳ loại đối tượng Python, Java hoặc Scala nào, bao gồm các lớp do người dùng định nghĩa.

b. Tạo RDD

Thông thường, người dùng sẽ tạo RDD theo hai cách: tải một tập dữ liệu bên ngoài hoặc phân phối một tập hợp các đối tượng trong chương trình.

Cách đơn giản nhất để tạo RDD là lấy một tập hợp hiện có trong chương trình và truyền nó đến phương thức parallelize() của SparkContext,. Cách tiếp cận này giúp nhanh chóng tạo một RDD và thực hiện các thao tác trên chúng.

Ví dụ: Tạo một RDD bằng parallelize() trong Python

data = [1, 2, 3, 4, 5]

rdd = sc.parallelize(data)

Một cách phổ biến hơn để tạo RDD là tải dữ liệu từ bộ nhớ lưu trữ bên ngoài.

Ví dụ: Tạo một RDD bằng textFile() trong Python

lines = sc.textFile(“đường\_dẫn\_đến\_file”)

c. Các phương thức của RDD

RDD hỗ trợ hai loại hoạt động: transformation và action. Transformation là các hoạt động trên RDD trả về một RDD mới, chẳng hạn như map() và filter(). Action là các hoạt động trả về kết quả cho chương trình hoặc ghi kết quả đó vào bộ nhớ và bắt đầu tính toán, chẳng hạn như count() và first().

Spark xử lý các transformation và action rất khác nhau, vì vậy việc hiểu hai loại hoạt động này thực hiện như nào rất quan trọng. Nếu bối rối một hàm là transformation hay action, ta có thể xem kiểu trả về của nó: transformation trả về một RDD. Trong khi action trả về một số kiểu dữ liệu khác.

d. Transformation

Transformation là các hoạt động trên RDD trả về một RDD mới. Giả sử chúng ta có một tệp nhật ký log.txt, với một số thông báo và chúng ta muốn chỉ chọn các thông báo “error”. Chúng ta sử dụng transformation filter().

Ví dụ:

inputRDD = sc.textFile(“log.txt”)

errorsRDD = inputRDD.filter(lambda x: “error” in x)

Lưu ý rằng filter() không làm thay đổi inputRDD hiện tại. Thay vào đó, nó trả về một con trỏ đến một RDD hoàn toàn mới. inputRDD vẫn có thể được sử dụng lại sau này trong chương trình. Giả sử chúng ta sử dụng inputRDD một lần nữa để tìm các thông báo “warning”, sau đó sử dụng transformation union() để in ra số dòng chứa lỗi hoặc cảnh báo.

Ví dụ:

warningRDD = input.filter(lambda x: “warning” in x)

badLinesRDD = errorsRDD.union(warningRDD)

e. Action

Action trả về giá trị cuối cùng cho chương trình hoặc ghi dữ liệu vò hệ thống lưu trữ bên ngoài. Các action buộc phải đánh giá các transformation cần thiết cho RDD mà chúng được gọi, vì chúng cần tạo kết quả đầu ra.

Tiếp tục với ví dụ trước, chúng ta có thể in ra một số thông tin về badLinesRDD. Để làm điều đó, chúng ta sẽ sửu dụng hai action, count() trả về số đếm dưới dạng số và take() sẽ thu thập một phần tử từ RDD.

Ví dụ:

print “Input had ” + badLinesRDD.count() + “ concerning lines”

print “Here are 10 examples:”

for line in badLinesRDD.take(10):

print line

Trong ví dụ này, take() được sử dụng để lấy một số lượng nhỏ các phần tử trong RDD tại chương trình, sau đó in chúng ra. RDD cũng có hàm collect() để lấy toàn bộ RDD. Điều này có thể hữu ích nếu chương trình lọc RDD xuống kích thước rất nhỏ và muốn xử lý nó cục bộ. Tuy nhiên, toàn bộ dữ liệu phải vừa với bộ nhớ trên một máy duy nhất để sử dụng collect() trên đó, vì vậy không nên sử dụng collect() trên các tập dữ liệu lớn.

### 1.2.3. Spark SQL

a. Sử dụng SQL trong chương trình

Cách hiệu quả nhất để sử dụng Spark SQL là sử dụng bên trong chương trình Spark. Điều này giúp chúng ta có khả năng dễ dàng tải dữ liệu và truy vấn dữ liệu bằng SQL, đồng thời kết hợp dữ liệu với mã lệnh thông thường trong Python, Java hoặc Scala.

Để sử dụng Spark SQL, cần thêm một số thư viện vào chương trình.

Ví dụ:

# Import Spark SQL

from pyspark.sql import HiveContext, Row

# Or if you can't include the hive requirements

from pyspark.sql import SQLContext, Row

Sau khi thêm thư viện, chúng ta cần tạo HiveContext hoặc SQLContext . Cả hai lớp này đều sử dụng SparkContext để chạy.

Ví dụ:

hiveCtx = HiveContext(sc)

b. Một số truy vấn cơ bản

Để thực hiện truy vấn đối với một bảng, ta gọi phương thức sql() trên HiveContext hoặc SQLContext. Đầu tiên, chúng ta cần cho Spark SQL biết về dữ liệu truy vấn. Giả sử, chúng ta tải một số dữ liệu Twitter từ JSON và đặt tên cho dữ liệu đó bằng cách đặt dữ liệu đó là bảng tạm thời để chúng ta truy vấn dữ liệu đó bằng SQL. Sau đó, có thể chọn các tweet hàng đầu theo retweetCount.

Ví dụ:

input = hiveCtx.jsonFile(inputFile)

input.registerTempTable(“tweets”)

topTweet = hiveCtx.sql(“SELECT text, retweetCount FROM tweets ORDER BY retweetCount LIMIT 10”)

### 1.2.4. Spark Streaming

Spark Streaming là một thành phần quan trọng của Apache Spark, cho phép xử lý dữ liệu trực tiếp và liên tục từ nhiều nguồn khác nhau như Kafka, Flume, Kinesis, hoặc socket TCP/IP.

Dưới đây là một số điểm quan trọng cần hiểu về Spark Streaming:

a. Xử lý dữ liệu liên tục:

Spark Streaming cho phép xử lý dữ liệu liên tục, tức là dữ liệu được xử lý và phân tích ngay khi nó được sinh ra. Điều này rất hữu ích cho các ứng dụng yêu cầu phản hồi thời gian thực hoặc phân tích dữ liệu liên tục như giao dịch tài chính, giám sát hệ thống, và xử lý luồng truyền thông xã hội.

Để minh họa cách xử lý dữ liệu liên tục bằng Spark Streaming, hãy xem xét một ví dụ đơn giản về việc đếm từ trong dữ liệu nhận được từ một nguồn text liên tục.

1. Khởi tạo SparkSession và SparkContext:

from pyspark.sql import SparkSession

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

sc = spark.sparkContext

2. Tạo Dstream từ nguồn dữ liệu:

from pyspark.streaming import StreamingContext

# Tạo một StreamingContext với khoảng thời gian nhỏ nhất 1 giây

ssc = StreamingContext(sc, 1)

# Kết nối đến cổng localhost 9999 để nhận dữ liệu

lines = ssc.socketTextStream("localhost", 9999)

3. Xử lý dữ liệu:

# Phân tách các từ từ dòng dữ liệu

words = lines.flatMap(lambda line: line.split(" "))

# Đếm số lượng từ

word\_counts = words.map(lambda word: (word, 1)).reduceByKey(lambda x, y: x + y)

4. In kết quả:

word\_counts.pprint()

# Khởi động quá trình xử lý dữ liệu liên tục

ssc.start()

# Đợi quá trình kết thúc

ssc.awaitTermination()

5. Gửi dữ liệu đến cổng 9999 bằng terminal:

nc -lk 9999

6. Kết quả:

Khi nhập các câu văn vào terminal, Spark Streaming sẽ đếm số lần xuất hiện của mỗi từ và in kết quả lên màn hình mỗi giây.

Ví dụ:

$ nc -lk 9999

hello world

hello spark

hello streaming

Kết quả trên màn hình của Spark Streaming sẽ là:

-------------------------------------------

Time: ...

-------------------------------------------

('world', 1)

('hello', 1)

('hello', 2)

('spark', 1)

('hello', 3)

('streaming', 1)

b Micro-batch Processing:

Spark Streaming sử dụng mô hình micro-batch processing, trong đó dữ liệu đến trong một khoảnh khắc nhất định được nhóm lại thành các micro-batch và xử lý bằng các quá trình xử lý tương tự như xử lý dữ liệu tĩnh trong Apache Spark.

c DStream:

DStream (Discretized Stream) là một thành phần quan trọng của Spark Streaming, là một chuỗi liên tục các RDDs (Resilient Distributed Datasets) được tạo ra bởi việc chia dữ liệu nhận được thành các micro-batch.

d. Tích hợp dễ dàng

Spark Streaming tích hợp chặt chẽ với các thành phần khác của hệ sinh thái Spark như Spark SQL, MLlib và GraphX, cho phép bạn kết hợp các phương tiện phân tích dữ liệu khác nhau để thực hiện các nhiệm vụ phức tạp trên dữ liệu liên tục.

e. Các nguồn dữ liệu đa dạng

Spark Streaming hỗ trợ nhiều nguồn dữ liệu đầu vào như Kafka, Flume, Kinesis, và socket TCP/IP. Điều này cho phép bạn kết nối và xử lý dữ liệu từ nhiều nguồn khác nhau một cách linh hoạt.

f. Tích hợp với dữ liệu tĩnh

Spark Streaming cũng tích hợp chặt chẽ với dữ liệu tĩnh trong Apache Spark, cho phép bạn kết hợp xử lý dữ liệu liên tục và dữ liệu tĩnh trong cùng một ứng dụng.

# Câu hỏi đề tài số 3

## 2.1. Câu hỏi 1

Câu hỏi: Trình bày về khái niệm RDD và các cách thức tạo ra RDD?

Trả lời:

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 có thể hoạt động song.

Đặc điểm:

Immutable (Bất biến): RDD không thể thay đổi sau khi được tạo. Bạn chỉ có thể tạo RDD mới thông qua các phép biến đối (transformations).

Fault-tolerant (Chịu lỗi): Spark có thể tự động tái tạo lại RDD trong trường hợp xảy ra lỗi nhờ thông tin lineage.

Lazy evaluation (Đánh giá lười): Các phép biến đổi không được thực thi ngay lập tức mà chỉ chạy khi gọi một hành động (action).

Low-level operations: Cho phép kiểm soát chi tiết quá trình biến đổi dữ liệu

Có hai cách để tạo RDDs:

Tạo từ một tập hợp dữ liệu có sẵn trong ngôn ngữ sử dụng như Java, Python, Scala: lines = sc.parallelize (["pandas", "i like pandas"])

Lấy từ dataset hệ thống lưu trữ bên ngoài như HDFS, Hbase hoặc các cơ sở dữ liệu quan hệ: lines = sc. textFile("README. md")

## 2.2. Câu hỏi 2

Câu hỏi: Trình bày về khái niệm và đặc điểm của Apache Spark?

Trả lời:

Apache Spark là một hệ thống tính toán phân tán mã nguồn mở, được thiết kế để xử lý dữ liệu nhanh chóng và đa năng. Khác với các hệ thống truyền thống như Hadoop MapReduce, Spark mang lại cải thiện hiệu suất đáng kể, đặc biệt là đối với các công việc phức tạp, nhờ vào việc xử lý dữ liệu trong bộ nhớ (in-memory). Điều này giúp các phép toán được thực hiện nhanh chóng hơn nhiều, điều này rất quan trọng khi xử lý các bộ dữ liệu lớn, nơi tốc độ là yếu tố quan trọng.

Các tính năng chính của Apache Spark bao gồm:

1. Tốc độ: Khả năng xử lý trong bộ nhớ giúp Spark tăng tốc đáng kể so với MapReduce, vốn phải ghi các kết quả trung gian vào đĩa. Điều này khiến Spark trở nên đặc biệt hữu ích cho các ứng dụng yêu cầu truy vấn dữ liệu nhanh và xử lý thời gian thực, chẳng hạn như các truy vấn tương tác và xử lý luồng dữ liệu.

2. Đa năng: Spark được thiết kế để xử lý nhiều loại công việc mà trước đây cần các hệ thống phân tán riêng biệt. Điều này bao gồm xử lý theo lô (batch processing), các thuật toán lặp đi lặp lại (như trong học máy), truy vấn tương tác, và xử lý luồng dữ liệu. Việc hỗ trợ tất cả những công việc này trong cùng một hệ thống giúp đơn giản hóa các đường ống phân tích dữ liệu và giảm bớt gánh nặng quản lý các hệ thống riêng biệt.

3. Dễ sử dụng: Spark được thiết kế để dễ tiếp cận với nhiều đối tượng người dùng khác nhau, cung cấp các API đơn giản bằng các ngôn ngữ phổ biến như Python, Java, Scala và SQL. Nó cũng cung cấp các thư viện tích hợp sẵn phong phú, chẳng hạn như Spark MLlib cho học máy và Spark SQL cho truy vấn dữ liệu có cấu trúc, làm cho nó trở thành công cụ linh hoạt cho cả nhà phát triển và nhà khoa học dữ liệu.

4. Tích hợp với các công cụ Big Data khác: Spark tích hợp tốt với các hệ sinh thái Big Data khác, đặc biệt là Hadoop. Nó có thể chạy trên các cụm Hadoop và truy cập các nguồn dữ liệu Hadoop như HDFS, HBase và Cassandra, khiến Spark trở thành lựa chọn lý tưởng cho các tổ chức đã sử dụng Hadoop.

Tóm lại, Apache Spark cung cấp một nền tảng xử lý dữ liệu lớn thống nhất, hiệu quả và dễ sử dụng, có khả năng xử lý các tác vụ tính toán đa dạng với tốc độ và tính linh hoạt cao.

Cấu Trúc Tích Hợp Đơn Nhất

Dự án Spark bao gồm nhiều thành phần được tích hợp chặt chẽ với nhau. Tại đó, Spark là một "công cụ tính toán" chịu trách nhiệm lên lịch, phân phối và giám sát các ứng dụng gồm nhiều tác vụ tính toán trên nhiều máy làm việc, hoặc cụm máy tính. Vì lõi của Spark vừa nhanh chóng vừa đa năng, nó cung cấp nền tảng cho nhiều thành phần cấp cao hơn, được chuyên biệt cho các loại công việc khác nhau, chẳng hạn như SQL hoặc học máy. Những thành phần này được thiết kế để tương tác chặt chẽ với nhau, giúp bạn kết hợp chúng như các thư viện trong một dự án phần mềm.

Một triết lý tích hợp chặt chẽ mang lại nhiều lợi ích. Đầu tiên, tất cả các thư viện và thành phần cấp cao trong cấu trúc này đều hưởng lợi từ các cải tiến ở các lớp thấp hơn. Ví dụ, khi công cụ lõi của Spark thêm một tối ưu hóa, các thư viện SQL và học máy sẽ tự động nhanh hơn. Thứ hai, chi phí liên quan đến việc vận hành cấu trúc này được tối thiểu hóa, vì thay vì chạy 5–10 hệ thống phần mềm độc lập, một tổ chức chỉ cần chạy một hệ thống duy nhất. Các chi phí này bao gồm triển khai, bảo trì, thử nghiệm, hỗ trợ, và các chi phí khác. Điều này cũng có nghĩa là mỗi khi một thành phần mới được thêm vào cấu trúc của Spark, mọi tổ chức sử dụng Spark sẽ ngay lập tức có thể thử nghiệm thành phần mới này. Điều này thay đổi chi phí thử nghiệm một loại phân tích dữ liệu mới từ việc tải xuống, triển khai và học một dự án phần mềm mới sang việc nâng cấp Spark.

Cuối cùng, một trong những lợi thế lớn nhất của sự tích hợp chặt chẽ là khả năng xây dựng các ứng dụng kết hợp liền mạch các mô hình xử lý khác nhau. Ví dụ, trong Spark, bạn có thể viết một ứng dụng sử dụng học máy để phân loại dữ liệu theo thời gian thực khi nó được nạp từ các nguồn luồng. Đồng thời, các nhà phân tích có thể truy vấn dữ liệu kết quả, cũng trong thời gian thực, qua SQL (ví dụ, để kết hợp dữ liệu với các tệp log không cấu trúc). Ngoài ra, các kỹ sư dữ liệu và nhà khoa học dữ liệu có thể truy cập cùng dữ liệu qua shell Python để phân tích tạm thời. Những người khác có thể truy cập dữ liệu trong các ứng dụng theo lô độc lập. Trong suốt quá trình này, đội ngũ IT chỉ cần duy trì một hệ thống duy nhất.

Dưới đây, chúng ta sẽ giới thiệu sơ lược về các thành phần của Spark, được hiển thị trong hình:

Several blue squares with black text

AI-generated content may be incorrect.

Hình .. Các thành phần của Spark

1. Spark Core

Spark Core chứa các chức năng cơ bản của Spark, bao gồm các thành phần cho việc lên lịch tác vụ, quản lý bộ nhớ, phục hồi lỗi, tương tác với các hệ thống lưu trữ, và nhiều hơn nữa. Spark Core cũng là nơi chứa API định nghĩa các tập dữ liệu phân tán bền vững (Resilient Distributed Datasets - RDDs), đây là trừu tượng lập trình chính của Spark. RDDs đại diện cho một tập hợp các phần tử được phân tán trên nhiều nút tính toán và có thể được xử lý song song. Spark Core cung cấp nhiều API để xây dựng và thao tác với các tập hợp này.

2. Spark SQL

Spark SQL là gói của Spark để làm việc với dữ liệu có cấu trúc. Nó cho phép truy vấn dữ liệu qua SQL cũng như biến thể SQL của Apache Hive—gọi là Hive Query Language (HQL)—và hỗ trợ nhiều nguồn dữ liệu, bao gồm bảng Hive, Parquet, và JSON. Ngoài việc cung cấp một giao diện SQL cho Spark, Spark SQL còn cho phép các nhà phát triển kết hợp các truy vấn SQL với các thao tác dữ liệu lập trình hỗ trợ bởi RDDs trong Python, Java và Scala, tất cả trong một ứng dụng duy nhất, do đó kết hợp SQL với phân tích phức tạp. Sự tích hợp chặt chẽ này với môi trường tính toán phong phú mà Spark cung cấp khiến Spark SQL trở nên khác biệt với bất kỳ công cụ kho dữ liệu nguồn mở nào khác. Spark SQL đã được thêm vào Spark trong phiên bản 1.0.

Shark là một dự án SQL-on-Spark cũ từ Đại học California, Berkeley, đã sửa đổi Apache Hive để chạy trên Spark. Shark hiện đã được thay thế bằng Spark SQL để cung cấp sự tích hợp tốt hơn với công cụ Spark và các ngôn ngữ API.

3. Spark Streaming

Spark Streaming là một thành phần của Spark cho phép xử lý các luồng dữ liệu trực tiếp. Các ví dụ về luồng dữ liệu bao gồm các tệp log được tạo ra bởi các máy chủ web sản xuất, hoặc các hàng đợi tin nhắn chứa các cập nhật trạng thái do người dùng của dịch vụ web gửi lên. Spark Streaming cung cấp một API để thao tác với các luồng dữ liệu tương tự như API RDD của Spark Core, giúp lập trình viên dễ dàng học và di chuyển giữa các ứng dụng thao tác với dữ liệu lưu trữ trong bộ nhớ, trên đĩa, hoặc đến theo thời gian thực. Dưới API của nó, Spark Streaming được thiết kế để cung cấp cùng mức độ độ tin cậy, thông lượng và khả năng mở rộng như Spark Core.

4. MLlib

Spark đi kèm với một thư viện chứa các chức năng học máy (ML) phổ biến, gọi là MLlib. MLlib cung cấp nhiều loại thuật toán học máy, bao gồm phân loại, hồi quy, phân cụm và lọc hợp tác, cùng với các chức năng hỗ trợ như đánh giá mô hình và nhập dữ liệu. Nó cũng cung cấp một số nguyên lý học máy cấp thấp hơn, bao gồm thuật toán tối ưu hóa gradient descent tổng quát. Tất cả các phương pháp này được thiết kế để mở rộng trên một cụm máy tính.

5. GraphX

GraphX là một thư viện dùng để thao tác với đồ thị (ví dụ: đồ thị bạn bè của một mạng xã hội) và thực hiện các tính toán song song đồ thị. Giống như Spark Streaming và Spark SQL, GraphX mở rộng API RDD của Spark, cho phép chúng ta tạo một đồ thị có hướng với các thuộc tính tùy ý gắn liền với mỗi đỉnh và cạnh. GraphX cũng cung cấp nhiều toán tử để thao tác với đồ thị (ví dụ: subgraph và mapVertices) và một thư viện các thuật toán đồ thị phổ biến (ví dụ: PageRank và đếm tam giác).

6. Cluster Managers

Spark được thiết kế để mở rộng hiệu quả từ một đến hàng ngàn nút tính toán. Để đạt được điều này trong khi tối đa hóa tính linh hoạt, Spark có thể chạy trên nhiều loại trình quản lý cụm (cluster managers), bao gồm Hadoop YARN, Apache Mesos và một trình quản lý cụm đơn giản được tích hợp sẵn trong Spark gọi là Standalone Scheduler. Nếu bạn chỉ cài đặt Spark trên một bộ máy tính trống, Standalone Scheduler cung cấp một cách dễ dàng để bắt đầu; tuy nhiên, nếu bạn đã có một cụm Hadoop YARN hoặc Mesos, sự hỗ trợ của Spark đối với các trình quản lý cụm này cho phép các ứng dụng của bạn cũng có thể chạy trên chúng.

Vì Spark là một framework đa năng cho tính toán cụm, nó được sử dụng cho một loạt các ứng dụng đa dạng. Trong phần giới thiệu, chúng ta đã đề cập đến hai nhóm đối tượng độc giả mà cuốn sách này nhắm đến: các nhà khoa học dữ liệu và kỹ sư. Hãy cùng tìm hiểu kỹ hơn về từng nhóm và cách họ sử dụng Spark. Như bạn có thể tưởng tượng, các trường hợp sử dụng điển hình sẽ khác nhau giữa hai nhóm, nhưng chúng ta có thể phân loại chúng thành hai nhóm chính: khoa học dữ liệu và ứng dụng dữ liệu.

Tất nhiên, những phân ngành và mô hình sử dụng này không phải lúc nào cũng rõ ràng, và nhiều người có kỹ năng từ cả hai lĩnh vực, đôi khi đóng vai trò của một nhà khoa học dữ liệu đang nghiên cứu, và sau đó "đổi mũ" để viết các ứng dụng xử lý dữ liệu đã được kiểm nghiệm. Tuy nhiên, việc xem xét hai nhóm này và các trường hợp sử dụng tương ứng có thể mang lại cái nhìn sâu sắc.

## 2.3. Câu hỏi 3

Câu hỏi: Cho tập dữ liệu về danh sách hành khách trên chuyến tàu có nội dung như bên dưới (File đính kèm), sử dụng Spark RDD hoặc Spark SQL để trả lời một số câu hỏi sau:

3.1.Tổng cộng có bao nhiêu hành khách trên chuyến tàu?

3.2. Trong số các hành khách bao nhiêu người là Nam, bao nhiêu người là Nữ?

3.3. Số lượng hành khách là trẻ em (Tuổi < 13)?

3.4. Tính số lượng hành khách là trẻ em nam và trẻ em nữ?

3.5. Tuổi lớn nhất và nhỏ nhất của tất cả hành khách trên chuyến tàu là bao nhiêu?

3.6. Có bao nhiêu hành khách có cùng tuổi lớn nhất, có cùng tuổi nhỏ nhất?

3.7. Tính tổng số tiền vé thu được? Giá vé trung bình ?

3.8. Liệt kê danh sách hành khách theo điều kiện giới tính là nam, mua vé có giá lớn hơn hoặc bằng mức giá trung bình, có tuổi lớn hơn hoặc bằng độ tuổi trung bình

3.9. Tạo một cột mới nếu: Pclass = 1 điền là “VIP”, Pclass = 2 điền là “Luxury”, còn lại điền là “Normal”

3.10. Tạo một cột mới nếu thông tin cột Sex là female và Sibsp là 1 điền là “wife”, nếu Sex là male điền là “husband”, còn lại điền là “Other”

Trả lời: Sử dụng Spark SQL

from pyspark.sql import SparkSession

#Khởi tạo Spark Session

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

data\_path = "D:\Môn học\Big data\Đề tài 3\Spark\_Titanic\Spark\_Titanic.csv"

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

#Tạo một view tạm thời

df.createOrReplaceTempView("Passengers")

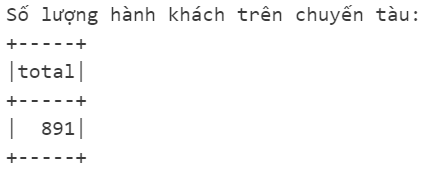
# 3.1. Tổng cộng có bao nhiêu hành khách trên chuyến tàu?

total\_passengers = spark.sql("SELECT COUNT(\*) AS total FROM Passengers")

print("Số lượng hành khách trên chuyến tàu:")

total\_passengers.show()

#Kết quả:



Hình .. Kết quả 3.1 bằng Spark SQL

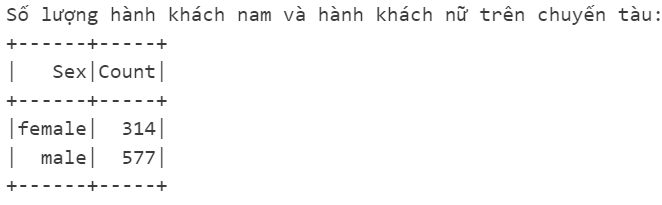
# 3.2. Trong số các hành khách bao nhiêu người là Nam, bao nhiêu người là Nữ?

gender\_count = spark.sql("SELECT Sex, Count(\*) as Count FROM Passengers GROUP BY Sex")

print("Số lượng hành khách nam và hành khách nữ trên chuyến tàu:")

gender\_count.show()

#Kết quả:



Hình .. Kết quả 3.2 bằng Spark SQL

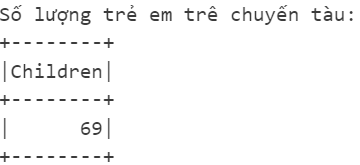
# 3.3. Số lượng hành khách là trẻ em (Tuổi < 13)?

children\_passengers = spark.sql("SELECT COUNT(\*) AS Children FROM Passengers WHERE Age < 13")

print("Số lượng trẻ em trê chuyến tàu:")

children\_passengers.show()

#Kết quả:



Hình .. Kết quả 3.3 bằng Spark SQL

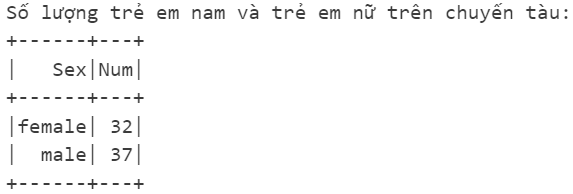
# 3.4. Tính số lượng hành khách là trẻ em nam và trẻ em nữ?

children\_gender = spark.sql("SELECT Sex, COUNT(\*) AS Num FROM Passengers WHERE Age < 13 GROUP BY Sex")

print("Số lượng trẻ em nam và trẻ em nữ trên chuyến tàu:")

children\_gender.show()

#Kết quả:



Hình .. Kết quả 3.4 bằng Spark SQL

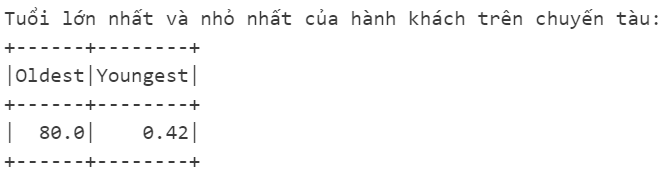
# 3.5. Tuổi lớn nhất và nhỏ nhất của tất cả hành khách trên chuyến tàu là bao nhiêu?

oldest\_youngest = spark.sql("SELECT MAX(Age) AS Oldest, MIN(Age) AS Youngest FROM Passengers WHERE Age IS NOT NULL")

print("Tuổi lớn nhất và nhỏ nhất của hành khách trên chuyến tàu:")

oldest\_youngest.show()

#Kết quả:



Hình .. Kết quả 3.5 bằng Spark SQL

# 3.6. Có bao nhiêu hành khách có cùng tuổi lớn nhất, có cùng tuổi nhỏ nhất?

oldest\_people = spark.sql("SELECT COUNT(\*) AS Num\_Oldest FROM Passengers WHERE Age = (SELECT MAX(Age) FROM Passengers)")

print("Số hành khách có tuổi cao nhất:")

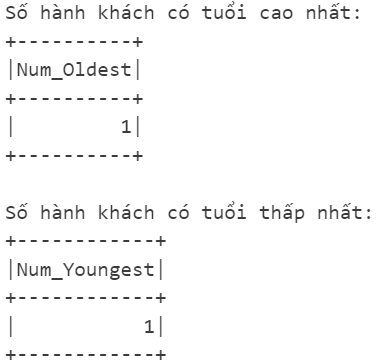
oldest\_people.show()

youngest\_people = spark.sql("SELECT COUNT(\*) AS Num\_Youngest FROM Passengers WHERE Age = (SELECT MIN(Age) FROM Passengers)")

print("Số hành khách có tuổi thấp nhất:")

youngest\_people.show()

#Kết quả:



Hình .. Kết quả 3.6 bằng Spark SQL

# 3.7. Tính tổng số tiền vé thu được? Giá vé trung bình ?

sum\_fare = spark.sql("SELECT SUM(Fare) AS Sum FROM Passengers")

print("Tổng số tiền vé thu được:")

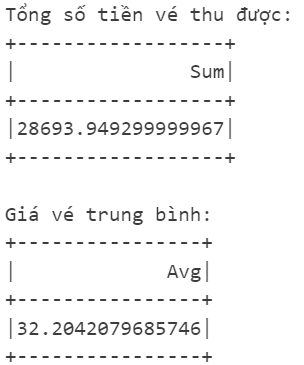
sum\_fare.show()

avg\_fare = spark.sql("SELECT AVG(Fare) AS Avg FROM Passengers")

print("Giá vé trung bình:")

avg\_fare.show()

#Kết quả:



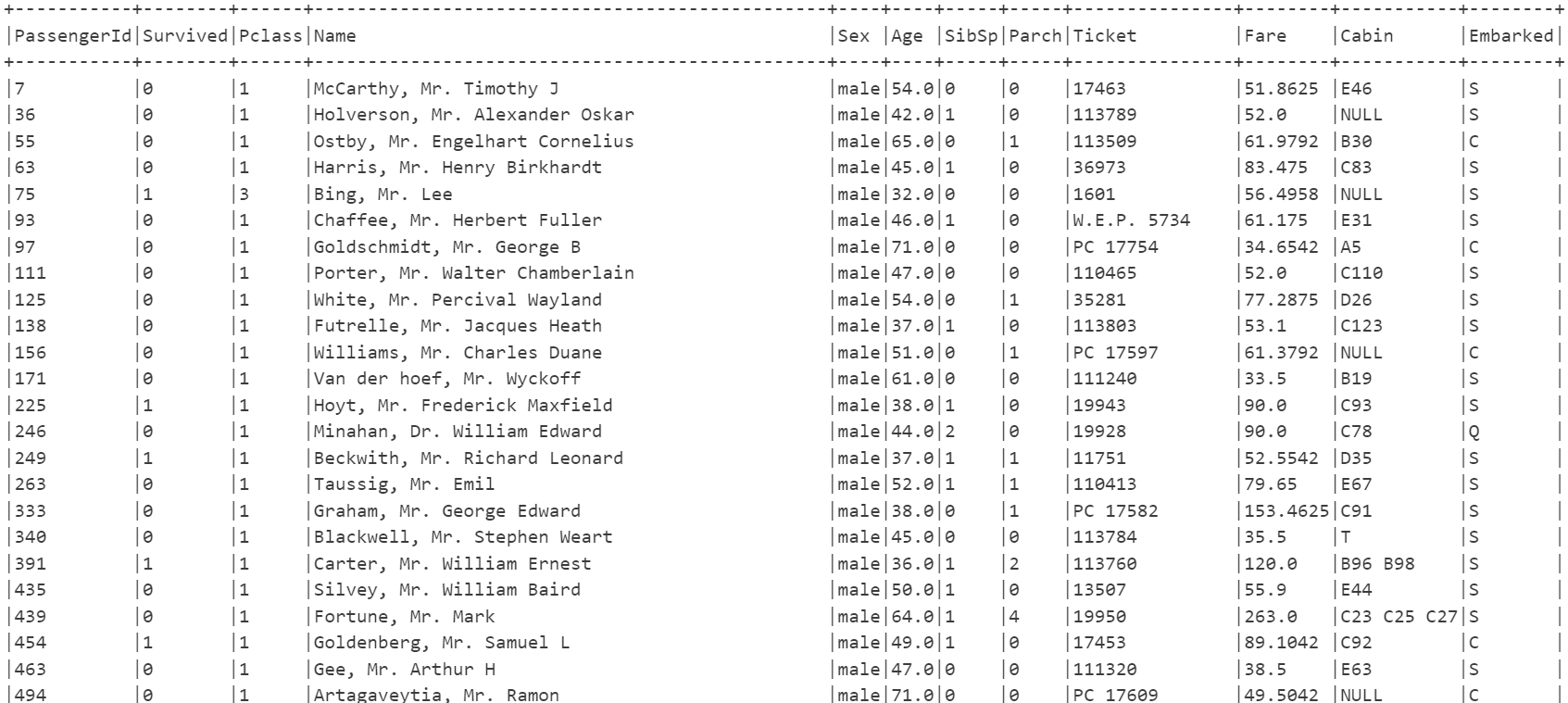
Hình .. Kết quả 3.7 bằng Spark SQL

# 3.8. Liệt kê danh sách hành khách theo điều kiện giới tính là nam, mua vé có giá lớn hơn hoặc bằng mức giá trung bình, có tuổi lớn hơn hoặc bằng độ tuổi trung bình

passengers\_list = spark.sql("SELECT \* FROM Passengers WHERE Sex = 'male' AND Fare >= (SELECT AVG(Fare) FROM Passengers) AND Age >= (SELECT AVG(Age) FROM Passengers)")

passengers\_list.show(n=passengers\_list.count(),truncate=False)

#Kết quả:



Hình .. Kết quả 3.8 bằng Spark SQL

# 3.9. Tạo một cột mới nếu: Pclass = 1 điền là “VIP”, Pclass = 2 điền là “Luxury”, còn lại điền là “Normal”

new\_col = spark.sql("""

SELECT \*,

CASE

WHEN Pclass = 1 THEN 'VIP'

WHEN Pclass = 2 THEN 'Luxury'

ELSE 'Normal'

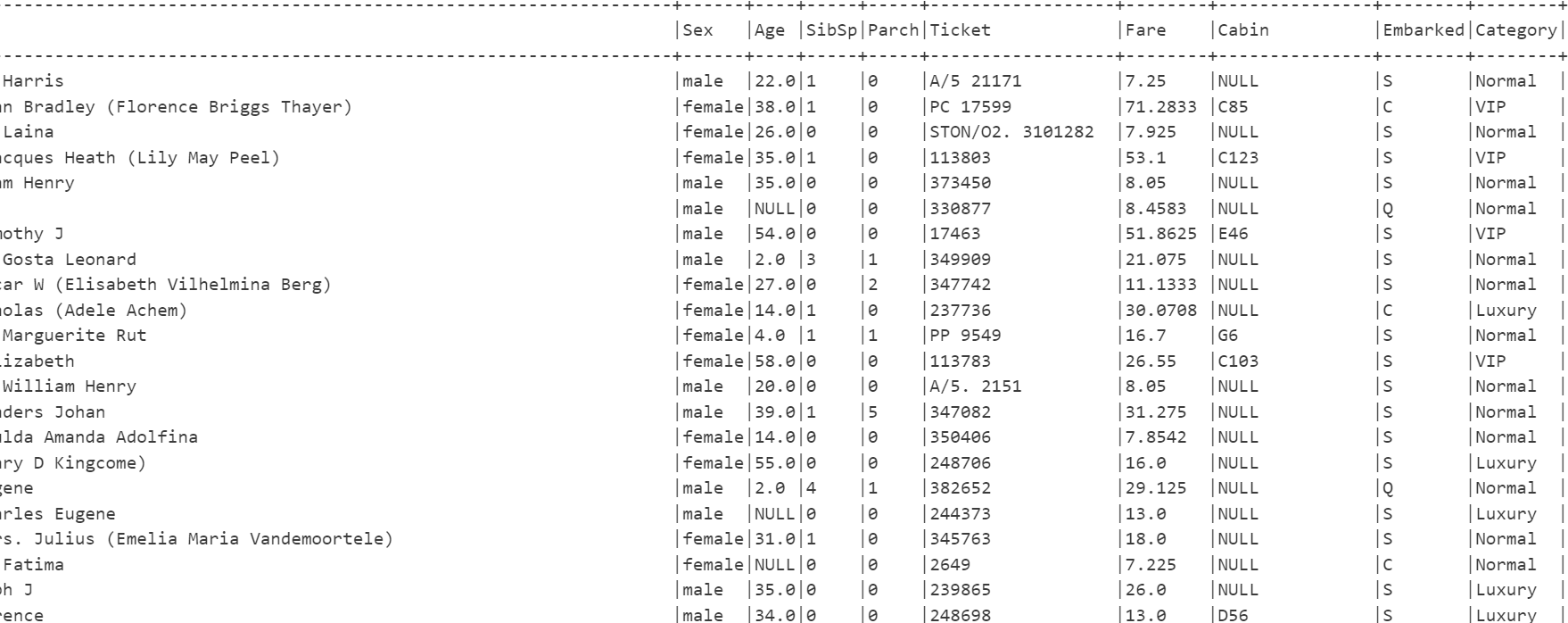
END AS Category

FROM Passengers

""")

new\_col.show(n=new\_col.count(),truncate=False)

#Kết quả:



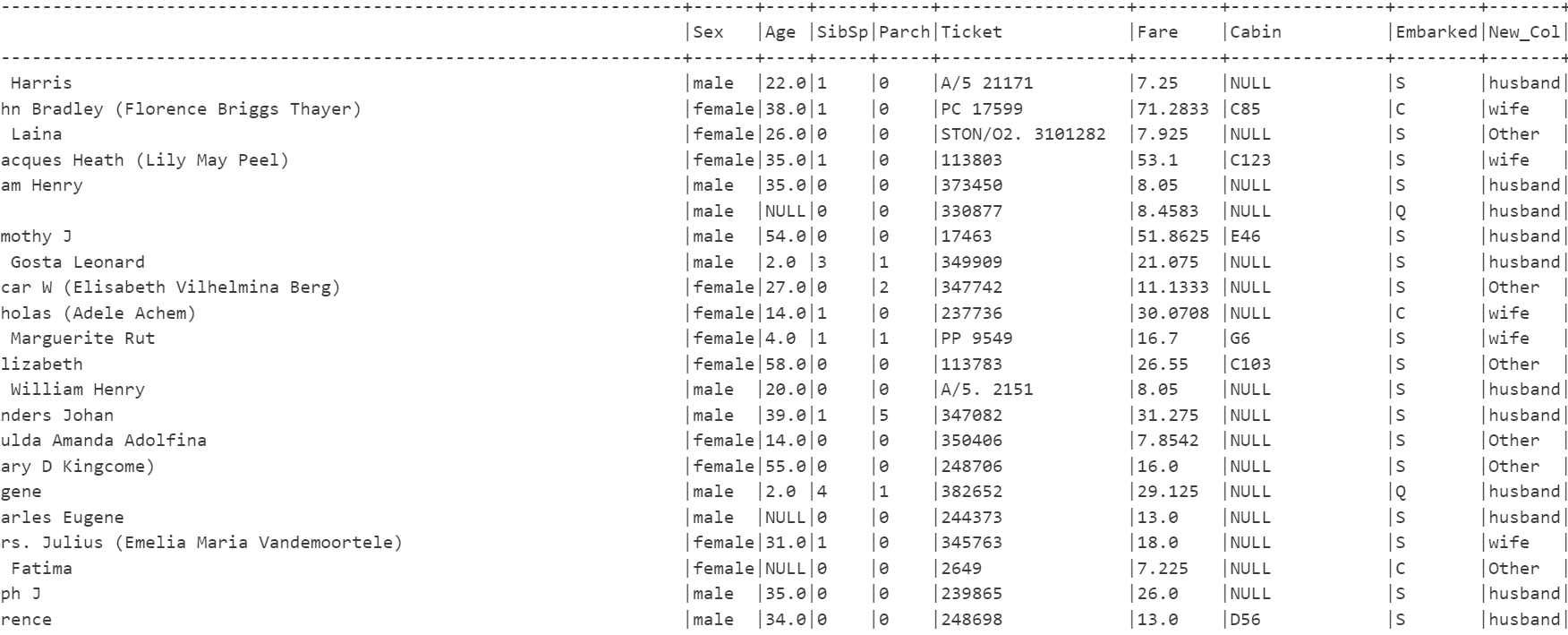
Hình .. Kết quả 3.9 bằng Spark SQL

# 3.10. Tạo một cột mới nếu thông tin cột Sex là female và Sibsp là 1 điền là “wife”, nếu Sex là male điền là “husband”, còn lại điền là “Other”

new\_col2 = spark.sql("SELECT \*, CASE WHEN (Sex = 'female' AND Sibsp = 1) THEN 'wife' WHEN (Sex = 'male') THEN 'husband' ELSE 'Other' END AS New\_Col FROM Passengers")

new\_col2.show(n=new\_col2.count(),truncate=False)

#Kết quả:



Hình .. Kết quả 3.1 bằng Spark SQL

## 2.4. Câu hỏi 4

Câu hỏi: Sử dụng dữ liệu giải đấu Championship từ năm 2015 đến 2020 (file đính kèm) để thực hành streaming kết quả tổng số bàn thắng và hiệu suất ghi bàn của đội Wigan Athletic theo năm.

Trả lời:

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

from pyspark.sql import SparkSession

import os

import time

# Cấu hình Spark

sc = SparkContext(appName="WiganStreaming")

sc.setLogLevel("ERROR") # Chỉ hiển thị lỗi

ssc = StreamingContext(sc, batchDuration=5)

# Tạo SparkSession

spark = SparkSession(sc)

# Thư mục chứa file CSV

data\_folder = "."

# Danh sách file CSV

file\_names = [f"Championship\_{year}.csv" for year in range(2015, 2021)]

def process\_batch(time, rdd):

if not rdd.isEmpty():

file\_path = rdd.collect()[0]

year = int(os.path.basename(file\_path).split('\_')[1].split('.')[0])

try:

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

df.createOrReplaceTempView("matches")

wigan\_home\_goals = spark.sql("SELECT FTHG FROM matches WHERE TRIM(HomeTeam) = 'Wigan'").rdd.map(lambda row: int(row[0]) if row[0].isdigit() else 0).sum()

wigan\_home\_matches = spark.sql("SELECT count(\*) FROM matches WHERE TRIM(HomeTeam) = 'Wigan'").collect()[0][0]

wigan\_away\_goals = spark.sql("SELECT FTAG FROM matches WHERE TRIM(AwayTeam) = 'Wigan'").rdd.map(lambda row: int(row[0]) if row[0].isdigit() else 0).sum()

wigan\_away\_matches = spark.sql("SELECT count(\*) FROM matches WHERE TRIM(AwayTeam) = 'Wigan'").collect()[0][0]

total\_goals = wigan\_home\_goals + wigan\_away\_goals

total\_matches = wigan\_home\_matches + wigan\_away\_matches

print(f"-------------------- Batch Time: {time} --------------------")

print(f"Dang xu ly du lieu nam: {year}")

print(f"Nam {year}: San nha: {wigan\_home\_matches}, San khach: {wigan\_away\_matches}")

if total\_matches > 0:

goals\_per\_match = total\_goals / total\_matches

print(f"Nam {year}: Tong so ban thang: {total\_goals}, So tran da dau: {total\_matches}, Hieu suat: {goals\_per\_match:.2f}")

else:

print(f"Nam {year}: Wigan khong thi dau hoac khong co du lieu.")

except Exception as e:

print(f"Loi khi xu ly file {os.path.basename(file\_path)}: {e}")

file\_stream = ssc.queueStream([sc.parallelize([os.path.join(data\_folder, file)]) for file in file\_names])

file\_stream.foreachRDD(process\_batch)

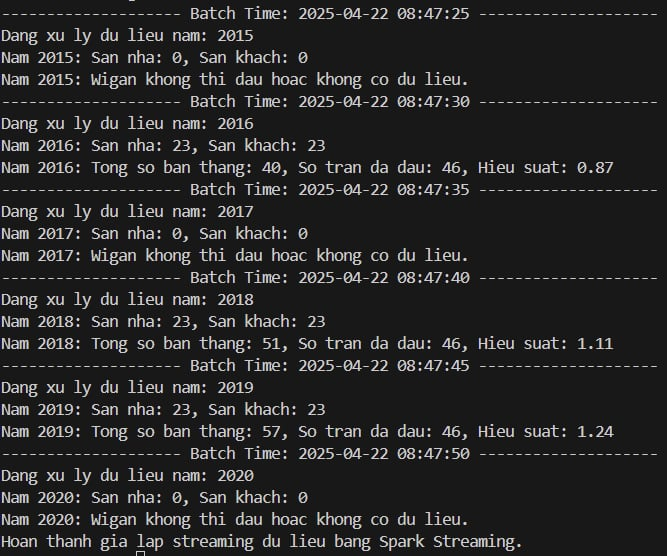
ssc.start()

time.sleep(len(file\_names) \* 5)

ssc.stop(stopSparkContext=True, stopGraceFully=True)

print("Hoan thanh gia lap streaming du lieu bang Spark Streaming.")

#Kết quả:



Hình .. Kết quả câu 4

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | B. W. J. S. D. T. D. Denny Lee, “Learning Spark : lightning-fast data analytics,” trong *Learning Spark : lightning-fast data analytics*, O'Reilly Media, 2020, pp. 47-60. |
| [2] | N. C. Hoan, “TỔNG QUAN VỀ DỮ LIỆU LỚN (BIGDATA),” trong *Kỷ yếu Hội thảo khoa học “Thống kê Nhà nước với Dữ liệu lớn”*, 2015. |
| [3] | “VIBLO,” 3 Tháng sáu 2024. [Trực tuyến]. Available: https://viblo.asia/p/spark-streaming-trong-apache-spark-y37Ld1m2Vov. |