* **Resilient Distributed Datasets (RDDs):**RDDs là cấu trúc dữ liệu cốt lõi trong Spark. Chúng là tập hợp dữ liệu phân tán và bất biến có thể được xử lý song song trên nhiều node. RDDs được tạo trong Spark Context, được phân phối trên các worker nodes và được lưu trữ trong bộ nhớ cache để tăng hiệu suất.
* **Dataframe** (hay còn gọi là khung dữ liệu) là một cấu trúc dữ liệu hai chiều được sử dụng phổ biến trong khoa học dữ liệu, học máy và điện toán khoa học. Nó được ví như một bảng tính với các hàng và cột, mỗi hàng đại diện cho một bản ghi dữ liệu và mỗi cột đại diện cho một thuộc tính của bản ghi đó. Dataframe có thể chứa nhiều loại dữ liệu khác nhau, bao gồm số, chuỗi, thời gian, v.v.

**Đặc điểm chính của Dataframe:**

* **Tổ chức dữ liệu hai chiều:** Dữ liệu được sắp xếp theo hàng và cột, giúp dễ dàng truy cập và thao tác.
* **Có nhãn:** Mỗi hàng và cột đều có nhãn để xác định dữ liệu mà nó chứa.
* **Hỗ trợ nhiều loại dữ liệu:** Dataframe có thể chứa nhiều loại dữ liệu khác nhau, bao gồm số, chuỗi, thời gian, v.v.
* **Dễ dàng thao tác:** Dataframe cung cấp nhiều hàm và phương thức để thao tác với dữ liệu, chẳng hạn như lọc, sắp xếp, tổng hợp, v.v.
* **Hiệu quả:** Dataframe được tối ưu hóa để xử lý dữ liệu lớn một cách hiệu quả.

**Dataset, hay bộ dữ liệu**, là một tập hợp các dữ liệu có tổ chức. Dữ liệu này có thể được biểu diễn dưới nhiều dạng khác nhau, chẳng hạn như:

* Số
* Chữ
* Hình ảnh
* Âm thanh
* Video

Dataset thường được lưu trữ dưới dạng các bảng, file văn bản, hoặc các định dạng chuyên biệt khác. Nó là nguồn nguyên liệu thô cung cấp thông tin cần thiết để thực hiện phân tích dữ liệu, học máy, trí tuệ nhân tạo (AI), và các lĩnh vực liên quan.

**Một số đặc điểm quan trọng của dataset:**

* Có cấu trúc: Dữ liệu trong dataset được tổ chức theo một cấu trúc nhất định, giúp dễ dàng truy cập, xử lý và phân tích.
* Kích thước: Dataset có thể có kích thước khác nhau, từ vài KB đến hàng Petabyte tùy thuộc vào lĩnh vực và mục đích sử dụng.
* Chất lượng: Chất lượng của dataset ảnh hưởng trực tiếp đến kết quả phân tích. Một dataset chất lượng cao, chính xác và đầy đủ sẽ giúp đưa ra những kết luận chính xác hơn.

So Sánh

**Điểm tương đồng giữa tất cả:**

* Khả năng chịu lỗi: Có thể tiếp tục hoạt động ngay cả khi một số node bị lỗi.
* Phân tán: Được phân phối trên nhiều node trong một cụm, cho phép xử lý dữ liệu lớn một cách hiệu quả.
* Xử lý song song trong bộ nhớ: Thực hiện các phép tính song song trên nhiều node, tận dụng tối đa sức mạnh tính toán của cụm.
* Không thay đổi: Dữ liệu không thể thay đổi sau khi được tạo, đảm bảo tính nhất quán và khả năng tái tạo của các kết quả tính toán.
* Đánh giá chậm (Lazy Evaluation): Các phép tính chỉ được thực hiện khi cần thiết, giúp tối ưu hóa hiệu suất và tiết kiệm tài nguyên.
* Được xử lý nội bộ bằng API RDD: Sử dụng RDD làm nền tảng cơ bản để xử lý dữ liệu, cung cấp các tính năng cốt lõi như tính chịu lỗi, khả năng phân tán và xử lý song song.

**Khác biệt:**

* Cả RDD và Dataset đều cung cấp API theo phong cách lập trình hướng đối tượng (OOP), trong khi DataFrame cung cấp API theo phong cách SQL.
* Với RDD, chúng ta chỉ định cho engine Spark cách thực hiện một tác vụ nhất định, trong khi với DataFrame và Dataset, chúng ta chỉ định những gì cần làm và Spark Engine sẽ xử lý phần còn lại. Đây là lý do tại sao DataFrame và Dataset vốn có các kỹ thuật tối ưu hóa.
* Trong RDD, chỉ các đối tượng trên heap được sử dụng, trong khi trong DataFrame và Dataset, cả bộ nhớ trên heap và ngoài heap đều có thể được tận dụng. Các đối tượng ngoài heap được sử dụng khi có thêm dữ liệu trong bộ nhớ.
* Do RDD chỉ sử dụng các đối tượng trên heap, nên việc tuần tự hóa (serialization) là không thể tránh khỏi vì dữ liệu bổ sung cần được truyền từ RAM sang đĩa. Điều này có thể tránh được trong DataFrame và Dataset do có vùng nhớ ngoài heap.
* Trong RDD, garbage collection (GC) ảnh hưởng đến hiệu suất, nhưng trong DataFrame và Dataset, tác động của GC đã được giải quyết.
* GC - Garbage Collector: Khi bộ nhớ đầy trong RDD, GC sẽ bắt đầu quét toàn bộ bộ nhớ và loại bỏ dữ liệu cũ và lỗi thời.
* Cả RDD và Dataset đều cung cấp tính an toàn kiểu dữ liệu mạnh mẽ, nghĩa là chúng sẽ báo lỗi ngay khi viết code nếu có gì sai sót, do đó cung cấp lỗi biên dịch (compile time error). DataFrame không có tính an toàn kiểu dữ liệu, vì vậy lỗi sẽ chỉ được biết khi code được thực thi (runtime error).

Câu 2

So sánh Chế độ Temporary view và Global Temporary Views trong Spark SQL

**Phạm vi:**

* Temporary view: Chỉ hiển thị trong SparkSession đã tạo ra nó. Nó sẽ biến mất khi phiên đó kết thúc.
* Global Temporary Views: Hiển thị trên tất cả SparkSession trong cùng một ứng dụng Spark. Nó tồn tại cho đến khi ứng dụng kết thúc.

**Khả năng hiển thị:**

* Temporary view: Không hiển thị với các sổ ghi hoặc phiên khác.
* Global Temporary Views: Hiển thị với các sổ ghi và phiên khác trong cùng một ứng dụng Spark.

**Vòng đời:**

* Temporary view: Giới hạn trong vòng đời của SparkSession.
* Global Temporary Views: Tồn tại cho đến khi ứng dụng Spark kết thúc.

**Cách sử dụng:**

* Temporary view: Phù hợp cho các tác vụ riêng lẻ hoặc phân tích riêng biệt trong một phiên.
* Global Temporary Views: Lý tưởng để chia sẻ dữ liệu trên nhiều sổ ghi hoặc phiên, thúc đẩy cộng tác và khả năng sử dụng lại.

**Khả năng truy cập:**

* Temporary view: Không thể truy cập bằng tên đầy đủ.
* Global Temporary Views: Được truy cập bằng tên đầy đủ (ví dụ: global temp. viewName).

**Đóng gói Sơ đồ:**

Cả hai chế độ**:** Đóng gói lược đồ DataFrame, đơn giản hóa các truy vấn SQL và giảm độ phức tạp của mã.

**Điểm chính:**

* Tính bền vững: Global Temporary Views tồn tại lâu hơn, mang lại sự linh hoạt để tái sử dụng.
* Chia sẻ: Global Temporary Views tạo điều kiện thuận lợi cho cộng tác và chia sẻ trên nhiều phiên.
* Khả năng truy cập: Global Temporary Views được truy cập bằng tên đầy đủ.

**Lựa chọn Chế độ Phù hợp:**

* Sử dụng Temporary view cho các tác vụ riêng biệt, cụ thể cho phiên.
* Lựa chọn Global Temporary Views khi cần chia sẻ dữ liệu trên nhiều phiên.