**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGUYEN TAT THANH**

**BÁO CÁO ĐỒ ÁN DỮ LIỆU LỚN**

**ĐỀ TÀI : SỬ DỤNG SPARK PHÂN TÍCH THỊ TRƯỜNG ÂM NHẠC TOÀN CẦU**

**Giảng viên hướng dẫn :Ths. Phạm Đình Tài**

**Sinh viên thực hiện : Phạm Khánh Dũ**

**MSSV : 2100009554**

**Khoá : 2021**

**Ngành/ chuyên ngành : Khoa học dữ liệu**

Tp HCM, tháng ..09.. năm 2024

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGUYEN TAT THANH**

**BÁO CÁO ĐỒ ÁN DỮ LIỆU LỚN**

**ĐỀ TÀI : SỬ DỤNG SPARK PHÂN TÍCH THỊ TRƯỜNG ÂM NHẠC TOÀN CẦU**

**Giảng viên hướng dẫn : Ths. Phạm Đình Tài**

**Sinh viên thực hiện : Phạm Khánh Dũ**

**MSSV : 2100009554**

**Khoá : 2021**

**Ngành/ chuyên ngành : Khoa học dữ liệu**

Tp HCM, tháng ..09.. năm 2024

# LỜI MỞ ĐẦU

Kính thưa Thầy Phạm Đình Tài,

Chân thành xin gửi đến Thầy những lời chào trân trọng và lòng biết ơn sâu sắc. Việc được học môn "Dữ liệu lớn" dưới sự hướng dẫn của Thầy không chỉ là một cơ hội lớn mà còn là một hành trình học thuật đầy ý nghĩa. Qua từng buổi giảng, chúng em đã nhận được những kiến thức sâu rộng, cùng sự tận tâm và nhiệt huyết của Thầy , đã giúp chúng em hiểu rõ hơn về tầm quan trọng của việc khai thác thông tin và dữ liệu trong thời đại số ngày nay.

Chúng em rất mong được thể hiện những kiến thức và kỹ năng đã học được thông qua dự án của mình, và chân thành mong muốn nhận được sự chỉ dẫn, đánh giá cũng như những góp ý quý báu từ Thầy để có thể phát triển hơn trong lĩnh vực này.

Một lần nữa, xin chân thành cảm ơn Thầy vì sự tận tâm và sự hỗ trợ không ngừng trong suốt khoảng thời gian qua.

Trân trọng!

# LỜI CẢM ƠN

Kính gửi Thầy Phạm Đình Tài,

Trong không khí tràn đầy tinh thần học thuật và nhiệt huyết, chúng em xin gửi đến Thầy lời cảm ơn chân thành nhất. Dưới sự hướng dẫn của Thầy, chúng em đã có cơ hội tiếp xúc với những khái niệm mới mẻ và ứng dụng chúng vào thực tế thông qua môn học "Dữ liệu lớn".

Chúng em không chỉ học được về kỹ thuật khai thác dữ liệu mà còn rèn luyện được tư duy phân tích và giải quyết vấn đề. Điều này là nhờ vào sự hỗ trợ và sự tận tâm của Thầy, người đã luôn tạo điều kiện tốt nhất để chúng em có thể phát triển toàn diện.

Chúng em xin bày tỏ lòng biết ơn sâu sắc và cam kết sẽ nỗ lực hết mình để trở thành những người học viên tích cực, đồng thời sẽ không quên những bài học quý báu mà Thầy đã truyền đạt.

Một lần nữa, xin chân thành cảm ơn Thầyvì tất cả những điều tốt lành mà Thầy đã mang đến cho chúng em.

Trân trọng!

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH  **TRUNG TÂM KHẢO THÍ** | **KỲ THI KẾT THÚC HỌC PHẦN**  **HỌC KỲ 7 – NĂM HỌC 2023 - 2024** |

**PHIẾU CHẤM ĐIỂM ĐỒ ÁN CƠ SỞ**

Môn thi: Dữ liệu lớn Lớp học phần: 21DTH2C

Nhóm sinh viên thực hiện :

1. Phạm Khánh Dũ-2100009554 Tham gia đóng góp:100%

2.Lê Ngô Văn Lộc-2100010394 Tham gia đóng góp:100%

Đề tài tiểu luận báo cáo của sinh viên : Sử dụng spark phân tích thị trường âm nhạc toàn cầu

Phần đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

Ngày thi: 04/09/2024 Phòng thi: L.604

Đề tài tiểu luận: Sử dụng spark phân tích thị trường âm nhạc toàn cầu

Phần đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí (theo CĐR HP)** | **Đánh giá của GV** | **Điểm tối đa** | **Điểm đạt được** |
| Cấu trúc của báo cáo |  |  |  |
| Nội dung |  |  |  |
| Các nội dung thành phần |  |  |  |
| Lập luận |  |  |  |
| Kết luận |  |  |  |
| Trình bày |  |  |  |
| **TỔNG ĐIỂM** |  |  |  |

**Giảng viên chấm thi**

*(ký, ghi rõ họ tên)*

GV.Phạm Đình Tài

# MỤC LỤC

[LỜI MỞ ĐẦU i](#_Toc176170942)

[LỜI CẢM ƠN ii](#_Toc176170943)

[MỤC LỤC iv](#_Toc176170944)

[DANH MỤC HÌNH ẢNH v](#_Toc176170945)

[**KÝ HIỆU CÁC TỪ VÀ CỤM TỪ VIẾT TẮC** vi](#_Toc176170946)

[**CHƯƠNG I: CƠ SỞ LÝ THUYẾT** 1](#_Toc176170947)

[1.1 Dữ liệu lớn (Big Data): 1](#_Toc176170948)

[1.1.1 Giới thiệu 1](#_Toc176170949)

[1.2 Kiến trúc và công cụ trong Big Data 1](#_Toc176170950)

[1.3 Giới thiệu về PySpark 1](#_Toc176170951)

[1.3.1. Các thành phần chính của Spark 2](#_Toc176170952)

[1.3.2. Lợi ích của PySpark trong phân tích dữ liệu 2](#_Toc176170953)

[1.3.3. PySpark trong phân tích thị trường âm nhạc 2](#_Toc176170954)

[1.4 Phương pháp xử lý và phân tích dữ liệu lớn 2](#_Toc176170955)

[1.4.1. Mô hình MapReduce 2](#_Toc176170956)

[1.4.2. Xử lý song song trong Spark 3](#_Toc176170957)

[1.4.3. Machine Learning trên dữ liệu lớn với MLlib 3](#_Toc176170958)

[1.4.4. Phân tích dữ liệu trực tiếp với Spark Streaming 3](#_Toc176170959)

[1.5. Áp dụng PySpark trong phân tích thị trường âm nhạc 3](#_Toc176170960)

[1.5.1. Khám phá và chuẩn bị dữ liệu 3](#_Toc176170961)

[1.5.2. Phân cụm bài hát và nghệ sĩ 4](#_Toc176170962)

[1.5.3. Phân tích xu hướng và dự đoán 4](#_Toc176170963)

[1.5.4. Hệ thống gợi ý âm nhạc 4](#_Toc176170964)

[**CHƯƠNG II : THỰC NGHIỆM VÀ KẾT QUẢ** 5](#_Toc176170965)

[2.1 Giới thiệu về đồ án 5](#_Toc176170966)

[2.1.1 Giới thiệu về dữ liệu 5](#_Toc176170967)

[2.2 *Tiền xử lý dữ liệu và 1 số thao tác với spark* 5](#_Toc176170968)

[2.2.1 Cái đặt thư viện pyspark và đọc dữ liệu 5](#_Toc176170969)

[2.2.2 Thiết lập phiên làm việc với PySpark 7](#_Toc176170970)

[**2.2.3** Đọc dữ liệu từ tệp CSV , Chuyển đổi cột và Loại bỏ các cột không cần thiết 7](#_Toc176170971)

[***2.2.4***Một số truy vấn SQL , ví dụ về cách sử dụng pyspark để lọc kết quả từ 1 ***tập dữ liệu*** 9](#_Toc176170972)

[***2.2.5*** Các kỹ thuật xử lý , phân tích dữ liệu và  kỹ thuật thống kê sử dụng để giảm chiều dữ liệu PCA***:*** 17](#_Toc176170973)

[2.2.6 Xây dựng mô hình Kmeans Cluster 24](#_Toc176170974)

[**CHƯƠNG III : KẾT LUẬN** 29](#_Toc176170975)

[3.1 Kết quả đạt được : 29](#_Toc176170976)

[3.2 Hiệu suất mô hình : 29](#_Toc176170977)

[3.3 GỢI Ý SẢN PHẨM CHO KHÁCH HÀNG : 30](#_Toc176170978)

[**3.4** **HẠN CHẾ :** 30](#_Toc176170979)

[TÀI LIỆU THAM KHẢO 30](#_Toc176170980)

# DANH MỤC HÌNH ẢNH

[Hình 2. 1 : Giới thiệu về dữ liệu 5](#_Toc176170461)

[Hình 2. 2 : Code cài đặt thư viện pyspark và đọc dữ liệu 6](#_Toc176170462)

[Hình 2. 3: Thiết lập phiên làm việc với PySpark 7](#_Toc176170463)

[Hình 2. 4: Đọc dữ liệu sử dụng spark và tiền xử lý dữ liệu 7](#_Toc176170464)

[Hình 2. 5:Đọc dữ liệu 5 dòng đầu 9](#_Toc176170465)

[Hình 2. 6:Tổng số lượng bài hát riêng biệt trong data 10](#_Toc176170466)

[Hình 2. 7: Số lượng dữ liệu sử dụng cú pháp spark và độ dài các cột 10](#_Toc176170467)

[Hình 2. 8: Top 10 nghệ sĩ Nghệ Sĩ Nổi Bật Nhất 10](#_Toc176170468)

[Hình 2. 9: CreateOrReplaceTempView 11](#_Toc176170469)

[Hình 2. 10: Top 10 nghệ sĩ phổ biến ở USA 12](#_Toc176170470)

[Hình 2. 11:Chuyển đổi dữ liệu Spark DataFrame sang Pandas DataFrame để trực quán hóa 12](#_Toc176170471)

[Hình 2. 12: Bài hát nổi bật nhất (có độ phổ biến cao nhất) từ từng thập kỷ 13](#_Toc176170472)

[Hình 2. 13: Tối ưu hóa việc lấy bài hát nổi bật nhất trong mỗi thập kỷ 14](#_Toc176170473)

[Hình 2. 14: 5 thể loại nhạc phổ biến nhất 15](#_Toc176170474)

[Hình 2. 15: Thể loại phổ biến nhất cho mỗi thập kỷ 15](#_Toc176170475)

[Hình 2. 16 : 10 bài hát phổ biến nhất của nghệ sĩ "Paulo Londra” 16](#_Toc176170476)

[Hình 2. 17 : Giá trị trung bình các đặc điểm âm thanh của các bài hát qua các thập kỷ 17](#_Toc176170477)

[Hình 2. 18: Truy vấn để thu thập dữ liệu về bài hát từ Mỹ và thể loại rap 18](#_Toc176170478)

[Hình 2. 19 : Truy vấn dữ liệu từ Mỹ 18](#_Toc176170479)

[Hình 2. 20 : Truy vấn Dữ Liệu từ Thể Loại Rap 19](#_Toc176170480)

[Hình 2. 21 : Tính toán tương đồng cosine giữa bài hát rap và bài hát Mỹ từ các tập dữ liệu khác nhau 20](#_Toc176170481)

[Hình 2. 22 Độ tương đồng cosine giữa hai bài hát rap. 21](#_Toc176170482)

[Hình 2. 23 Tỷ lệ phương sai giải thích được bởi mỗi thành phần chính và ảnh trực quan hóa 22](#_Toc176170483)

[Hình 2. 24 :Biểu Đồ Phân Tán 24](#_Toc176170484)

[Hình 2. 25: Tìm số cụm tối ưu bằng phương pháp Elbow 24](#_Toc176170485)

[Hình 2. 26: K-Means Clustering thể loại rap và usa 26](#_Toc176170486)

[Hình 2. 27 Biểu đồ phân tán sử dụng mô hình K-Means đã phân chia các bài hát thành các cụm riêng biệt 27](#_Toc176170487)

**KÝ HIỆU CÁC TỪ VÀ CỤM TỪ VIẾT TẮC**

|  |  |
| --- | --- |
| Chữ viết tắt | Ý nghĩa |
| **EDA** | **Exploratory Data Analysis** |
| **SQL** | **Structured Query Language** |

**CHƯƠNG I: CƠ SỞ LÝ THUYẾT**

## 1.1 Dữ liệu lớn (Big Data):

### 1.1.1 Giới thiệu

**Dữ liệu lớn (Big Data)** là thuật ngữ dùng để chỉ tập hợp các dữ liệu có kích thước rất lớn và phức tạp mà các công cụ truyền thống không thể xử lý một cách hiệu quả. Big Data thường được đặc trưng bởi 4 yếu tố chính, gọi là 4V:

* **Volume (Khối lượng)**: Khối lượng dữ liệu rất lớn, thường tính bằng terabyte hoặc petabyte.
* **Velocity (Tốc độ)**: Dữ liệu được tạo ra và xử lý với tốc độ cao, gần như theo thời gian thực.
* **Variety (Đa dạng)**: Dữ liệu đến từ nhiều nguồn khác nhau và ở nhiều định dạng khác nhau (cấu trúc, phi cấu trúc, bán cấu trúc).
* **Veracity (Độ tin cậy)**: Dữ liệu có thể không chính xác hoặc không đầy đủ, gây khó khăn trong việc phân tích.

Với sự bùng nổ của Internet, mạng xã hội, và các thiết bị IoT, lượng dữ liệu tạo ra hàng ngày là vô cùng lớn. Việc khai thác và phân tích dữ liệu này mang lại nhiều cơ hội trong nhiều lĩnh vực như kinh doanh, y tế, giáo dục, và giải trí.

## 1.2 Kiến trúc và công cụ trong Big Data

Để xử lý Big Data, các hệ thống cần có khả năng mở rộng và phân tán cao. Một số công nghệ và công cụ phổ biến trong lĩnh vực này bao gồm:

* **Hadoop**: Nền tảng xử lý dữ liệu phân tán phổ biến nhất, dựa trên mô hình MapReduce.
* **Spark**: Nền tảng xử lý dữ liệu nhanh và hiệu quả hơn Hadoop, hỗ trợ nhiều ngôn ngữ lập trình như Python, Java, và Scala.
* **NoSQL Databases**: Các hệ quản trị cơ sở dữ liệu phi quan hệ như MongoDB, Cassandra, được thiết kế để lưu trữ và quản lý dữ liệu không có cấu trúc.
* **Data Lakes**: Lưu trữ dữ liệu thô với chi phí thấp trước khi xử lý.

## 1.3 Giới thiệu về PySpark

**PySpark** là một giao diện lập trình của Apache Spark được viết bằng Python. Apache Spark là một nền tảng mã nguồn mở dùng để xử lý dữ liệu phân tán, được phát triển nhằm tăng tốc quá trình xử lý dữ liệu lớn so với các phương pháp truyền thống.

### 1.3.1. Các thành phần chính của Spark

* **Spark Core**: Là lõi của Spark, cung cấp các dịch vụ như quản lý tài nguyên, lập lịch trình công việc, và tối ưu hóa thực thi.
* **Spark SQL**: Cho phép xử lý dữ liệu dạng bảng với cú pháp SQL.
* **Spark Streaming**: Hỗ trợ xử lý dữ liệu trực tiếp từ các luồng dữ liệu (streaming).
* **MLlib**: Thư viện hỗ trợ các thuật toán học máy (machine learning) trên Spark.
* **GraphX**: Công cụ để xử lý và phân tích đồ thị.

### 1.3.2. Lợi ích của PySpark trong phân tích dữ liệu

* **Hiệu năng cao**: PySpark có khả năng xử lý dữ liệu nhanh chóng nhờ khả năng tối ưu hóa và quản lý bộ nhớ hiệu quả.
* **Khả năng mở rộng**: Có thể mở rộng để xử lý dữ liệu lớn trên nhiều cụm máy chủ.
* **Tích hợp tốt với hệ sinh thái Big Data**: PySpark tích hợp dễ dàng với các công cụ và nền tảng khác trong hệ sinh thái Big Data như Hadoop, Hive, và HDFS.

### 1.3.3. PySpark trong phân tích thị trường âm nhạc

Trong phân tích thị trường âm nhạc toàn cầu, PySpark được sử dụng để thực hiện các bước như:

* **EDA (Exploratory Data Analysis)**: Khám phá dữ liệu, hiểu được các xu hướng và mẫu trong dữ liệu.
* **Clustering**: Nhóm các bài hát hoặc nghệ sĩ thành các cụm dựa trên các đặc điểm như thể loại, nhịp điệu, và phổ biến.
* **Streaming Analytics**: Phân tích dữ liệu âm nhạc từ các nguồn phát trực tiếp để hiểu xu hướng trong thời gian thực.

## 1.4 Phương pháp xử lý và phân tích dữ liệu lớn

### 1.4.1. Mô hình MapReduce

**MapReduce** là mô hình lập trình cho phép xử lý dữ liệu lớn song song trên nhiều máy tính. Mô hình này chia quá trình xử lý thành hai giai đoạn chính:

* **Map**: Dữ liệu đầu vào được chia nhỏ và xử lý song song trên nhiều node để tạo ra cặp khóa-giá trị (key-value).
* **Reduce**: Kết quả từ giai đoạn Map được tổng hợp lại để cho ra kết quả cuối cùng.

MapReduce là nền tảng của nhiều hệ thống xử lý dữ liệu lớn, đặc biệt là Hadoop. Tuy nhiên, vì hiệu năng không cao trong một số trường hợp nhất định, các công cụ như Spark đã ra đời để khắc phục nhược điểm này.

### 1.4.2. Xử lý song song trong Spark

Spark cải tiến mô hình MapReduce bằng cách cho phép lưu trữ kết quả trung gian trong bộ nhớ (in-memory), giảm thiểu thời gian đọc/ghi đĩa cứng và cải thiện hiệu năng. Spark thực hiện xử lý dữ liệu lớn bằng cách chia công việc thành các task nhỏ và phân tán chúng trên nhiều node trong cluster.

### 1.4.3. Machine Learning trên dữ liệu lớn với MLlib

**MLlib** là thư viện học máy của Spark, hỗ trợ nhiều thuật toán học máy phổ biến như:

* **Classification (Phân loại)**: Như logistic regression, decision trees.
* **Clustering (Phân cụm)**: Như k-means, Gaussian mixture models.
* **Recommendation (Đề xuất)**: Như collaborative filtering, thường dùng trong các hệ thống gợi ý.

MLlib giúp thực hiện các mô hình học máy trên dữ liệu lớn một cách hiệu quả nhờ vào khả năng xử lý song song và tối ưu hóa tài nguyên.

### 1.4.4. Phân tích dữ liệu trực tiếp với Spark Streaming

**Spark Streaming** là thành phần của Spark hỗ trợ xử lý dữ liệu thời gian thực. Spark Streaming xử lý dữ liệu từ các nguồn như Kafka, Flume, hoặc socket, và cho phép ứng dụng các mô hình phân tích và học máy lên luồng dữ liệu này.

Điều này đặc biệt quan trọng trong các ứng dụng yêu cầu phản hồi nhanh, như phân tích cảm xúc trên mạng xã hội hoặc theo dõi các sự kiện trong thời gian thực.

## 1.5. Áp dụng PySpark trong phân tích thị trường âm nhạc

### 1.5.1. Khám phá và chuẩn bị dữ liệu

Khi phân tích thị trường âm nhạc toàn cầu, bước đầu tiên là khám phá và chuẩn bị dữ liệu. Dữ liệu âm nhạc thường bao gồm các thông tin về bài hát, nghệ sĩ, thể loại, lượt nghe, và cảm nhận từ người dùng. Các bước cần thực hiện bao gồm:

* **Làm sạch dữ liệu**: Xử lý dữ liệu thiếu, loại bỏ các giá trị ngoại lai.
* **Chuyển đổi dữ liệu**: Chuẩn hóa các thuộc tính, mã hóa các biến phân loại.

### 1.5.2. Phân cụm bài hát và nghệ sĩ

Một trong những ứng dụng của PySpark trong phân tích âm nhạc là phân cụm (clustering) các bài hát hoặc nghệ sĩ dựa trên các đặc điểm âm nhạc như thể loại, tempo, hoặc mức độ phổ biến. Kết quả phân cụm giúp hiểu rõ hơn về các xu hướng âm nhạc và thị hiếu của người nghe.

### 1.5.3. Phân tích xu hướng và dự đoán

Dữ liệu âm nhạc có thể được phân tích để xác định các xu hướng trong thị trường âm nhạc, ví dụ như sự phát triển của một thể loại âm nhạc cụ thể theo thời gian. Các mô hình dự đoán cũng có thể được áp dụng để dự báo sự thành công của một bài hát dựa trên các thuộc tính của nó.

### 1.5.4. Hệ thống gợi ý âm nhạc

Dựa trên các kỹ thuật học máy trong MLlib, bạn có thể xây dựng một hệ thống gợi ý âm nhạc (music recommendation system) để đề xuất các bài hát hoặc nghệ sĩ mới cho người dùng dựa trên sở thích và lịch sử nghe nhạc của họ.

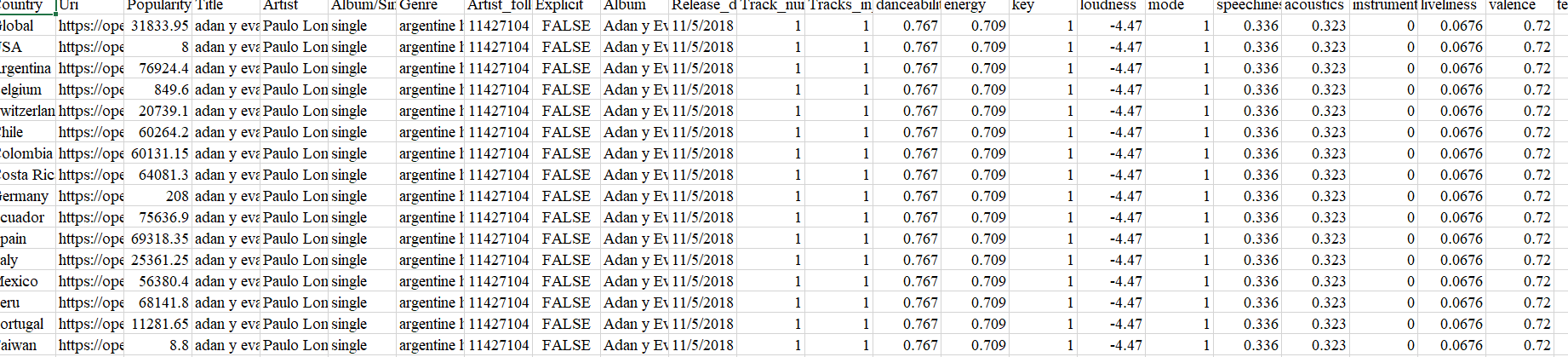
**CHƯƠNG II : THỰC NGHIỆM VÀ KẾT QUẢ**

## 2.1 Giới thiệu về đồ án

Trong chương này, chúng tôi sử dụng PySpark để tiến hành phân tích dữ liệu thị trường âm nhạc toàn cầu. Dữ liệu được lấy từ bộ dữ liệu Spotify, chứa các thuộc tính chi tiết về các bài hát như tên bài hát, nghệ sĩ, thể loại, và các thuộc tính âm nhạc như energy, danceability, tempo, valence, v.v. Mục tiêu chính của thực nghiệm là:

* Khám phá và tiền xử lý dữ liệu để chuẩn bị cho các bước phân tích tiếp theo.
* Áp dụng thuật toán phân cụm (clustering) để nhóm các bài hát có các đặc điểm tương đồng.
* Đánh giá và phân tích các kết quả để hiểu rõ hơn về xu hướng âm nhạc toàn cầu.

### 2.1.1 Giới thiệu về dữ liệu



Hình 2. 1 : Giới thiệu về dữ liệu

* Dữ liệu này được sử dụng để tính Điểm phổ biến trong "Cơ sở dữ liệu cuối cùng" (xem tệp khác), hãy xem tệp đó để biết dữ liệu đầy đủ hơn về các bài hát
* Bảng xếp hạng Top 200 hàng ngày của Spotify ghi lại hàng ngày từ tháng 1 năm 2017 đến tháng 12 năm 2020.
* Biểu đồ từ 35 quốc gia +1 (toàn cầu).
* Dữ liệu bao gồm vị trí, tiêu đề, nghệ sĩ, quốc gia, ngày và uri (được sử dụng trong Spotify API).

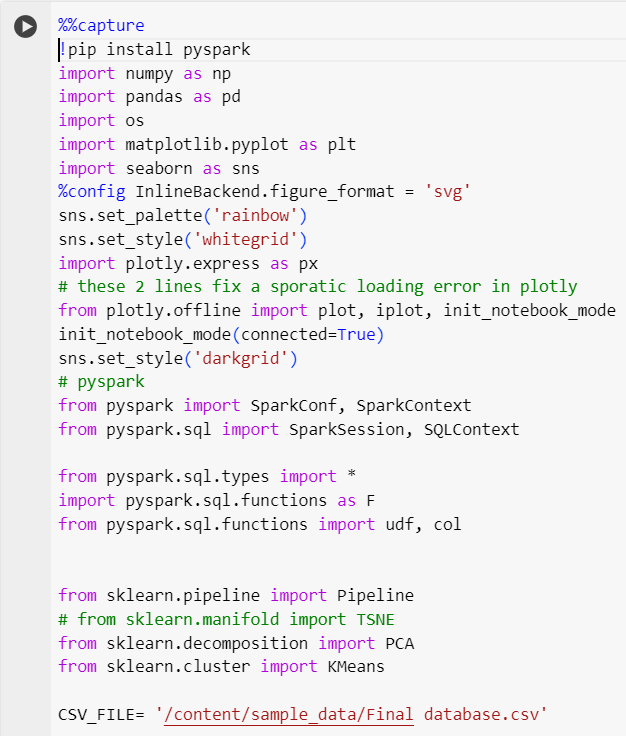
## 2.2 *Tiền xử lý dữ liệu và 1 số thao tác với spark*

### 2.2.1 Cái đặt thư viện pyspark và đọc dữ liệu

- !pip install pyspark để cài đặt thư viện

- Mã nguồn sữ dụng thư viện pandas để đọc dữ liệu từ file CSV và khai báo các thư viện

Trong đó, pd là viết tắt của pandas, read\_csv() là một hàm trong pandas được sử dụng để đọc dữ liệu từ file CSV. Trong mã nguồn trên, chúng ta đọc dữ liệu từ tệp CSV có tên là Final database.csv

******

Hình 2. 2 : Code cài đặt thư viện pyspark và đọc dữ liệu

### 2.2.2 Thiết lập phiên làm việc với PySpark



Hình 2. 3: Thiết lập phiên làm việc với PySpark

🡪Mô tả: Đoạn mã này thiết lập một phiên làm việc với Spark bằng cách cấu hình SparkContext và SparkSession và appname xậy dựng có tên là Spotify-Huge-Dataset

**2.2.3 Đọc dữ liệu từ tệp CSV , Chuyển đổi cột và Loại bỏ các cột không cần thiết**



Hình 2. 4: Đọc dữ liệu sử dụng spark và tiền xử lý dữ liệu

**Đoạn mã này là một phần của quá trình tiền xử lý dữ liệu trước khi phân tích hoặc mô hình hóa. Giải thích từng phần của đoạn mã:**

**1. Đọc dữ liệu từ tệp CSV**

**df = spark.read.option("header", True).csv(CSV\_FILE)**

**- Mô tả: Dòng mã này sử dụng PySpark để đọc một tệp CSV vào một DataFrame. Tùy chọn `"header", True` chỉ định rằng tệp CSV có dòng đầu tiên là tiêu đề cột.**

**- Ý nghĩa: Đọc dữ liệu từ tệp CSV là bước đầu tiên trong việc chuẩn bị dữ liệu để phân tích. Spark DataFrame là một cấu trúc dữ liệu mạnh mẽ, có khả năng xử lý lượng dữ liệu lớn hơn nhiều so với Pandas DataFrame thông thường.**

**2. Chuyển đổi cột `Release\_date` sang kiểu ngày tháng**

**df = df.withColumn("Release\_date", F.to\_date("Release\_date", "yyyy-MM-dd"))**

**- Mô tả: Dòng mã này chuyển đổi cột `Release\_date` từ định dạng chuỗi (string) sang định dạng ngày tháng (`date`) theo định dạng "yyyy-MM-dd".**

**- Ý nghĩa: Việc chuyển đổi dữ liệu ngày tháng về đúng kiểu dữ liệu là cần thiết để thực hiện các phân tích liên quan đến thời gian, chẳng hạn như tính số ngày kể từ khi phát hành.**

**3. Chuyển đổi các cột đặc trưng số về kiểu dữ liệu `float`**

**numerical\_features = ['danceability', 'energy', 'instrumentalness', 'valence', 'liveliness', 'speechiness', 'acoustics',**

**'instrumentalness', 'tempo', 'duration\_ms',**

**'time\_signature', 'Days\_since\_release', 'n\_words']**

**for c in numerical\_features:**

**df = df.withColumn(c, df[c].cast("float"))**

**- Mô tả: Danh sách `numerical\_features` chứa tên của các cột có giá trị số. Đoạn mã sau đó chuyển đổi các cột này sang kiểu dữ liệu `float`.**

**- Ý nghĩa: Chuyển đổi các cột số sang kiểu `float` là cần thiết để đảm bảo rằng các phép tính số học hoặc các mô hình học máy hoạt động chính xác.**

**4. Loại bỏ các cột không cần thiết**

**cols\_to\_drop = ['syuzhet\_norm', 'bing\_norm', 'afinn\_norm', 'nrc\_norm', 'syuzhet', 'bing']**

**for c in cols\_to\_drop:**

**df.drop(c).collect()**

**- Mô tả: Danh sách `cols\_to\_drop` chứa tên của các cột không cần thiết. Đoạn mã này loại bỏ các cột này khỏi DataFrame.**

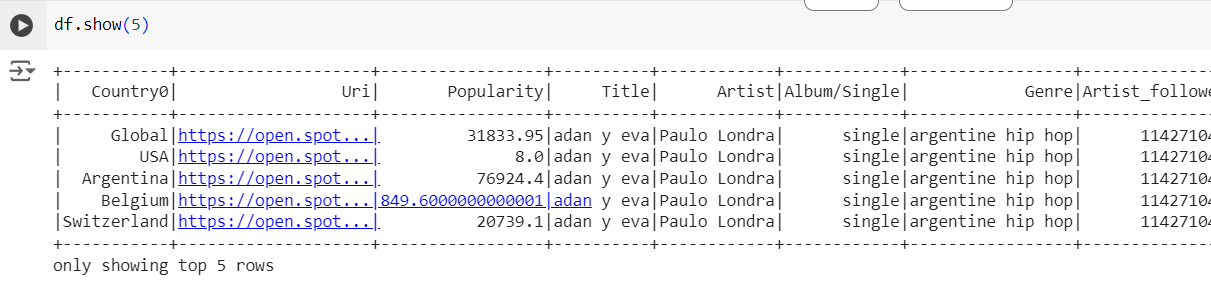
**- Ý nghĩa: Việc loại bỏ các cột không cần thiết giúp làm sạch dữ liệu, giảm kích thước bộ dữ liệu, và tập trung vào những đặc trưng quan trọng cho phân tích hoặc mô hình hóa.**

**5. In cấu trúc của DataFrame**

**df.printSchema()**

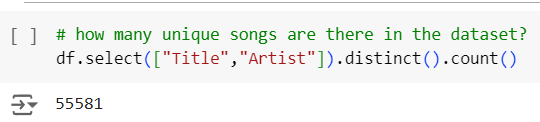
**- Mô tả: Dòng mã này in ra cấu trúc của DataFrame, bao gồm tên các cột và kiểu dữ liệu của chúng.**

**- Ý nghĩa: Việc kiểm tra cấu trúc DataFrame là bước cuối cùng để đảm bảo rằng dữ liệu đã được xử lý đúng cách và sẵn sàng cho các bước phân tích tiếp theo.**

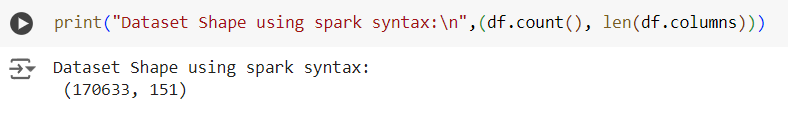
**Đây là quy trình tiền xử lý dữ liệu điển hình trong phân tích dữ liệu lớn, giúp chuẩn bị dữ liệu cho các bước phân tích và mô hình hóa tiếp theo.**

Hình 2. 5:Đọc dữ liệu 5 dòng đầu

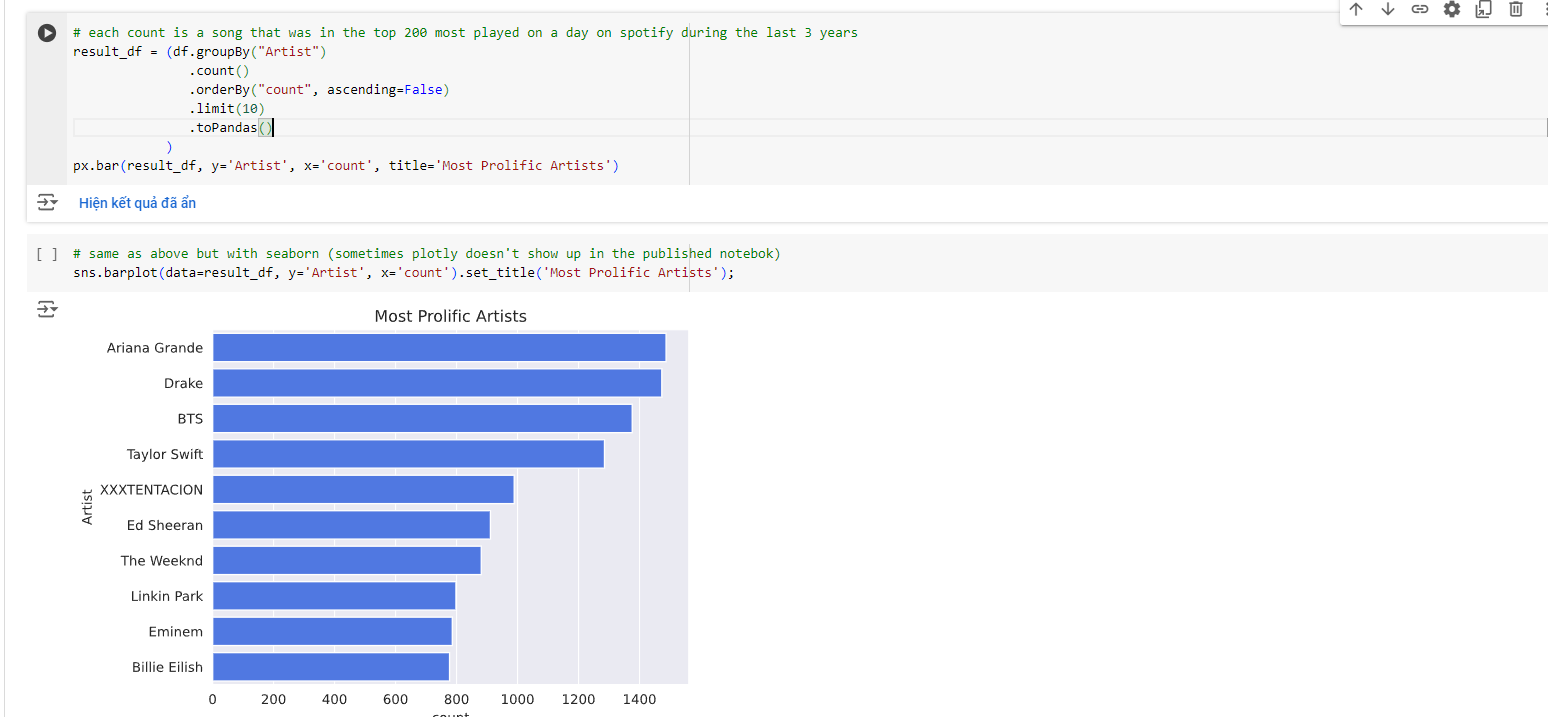
**2.2.4 Một số truy vấn SQL , ví dụ về cách sử dụng pyspark để lọc kết quả từ 1tập dữ liệu**

****

Hình 2. 6:Tổng số lượng bài hát riêng biệt trong data



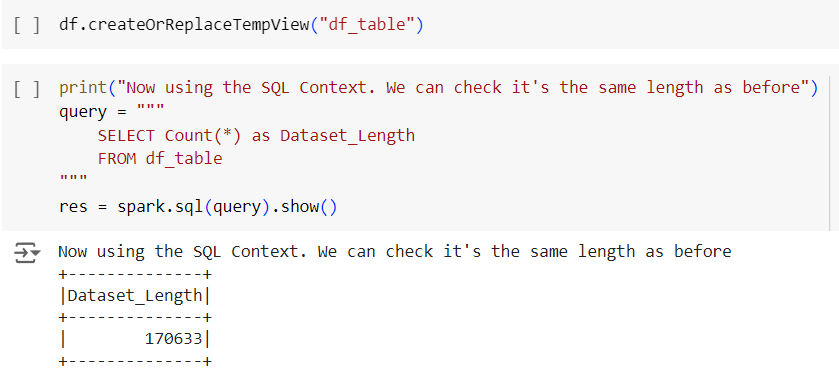
Hình 2. 7: Số lượng dữ liệu sử dụng cú pháp spark và độ dài các cột



Hình 2. 8: Top 10 nghệ sĩ Nghệ Sĩ Nổi Bật Nhất

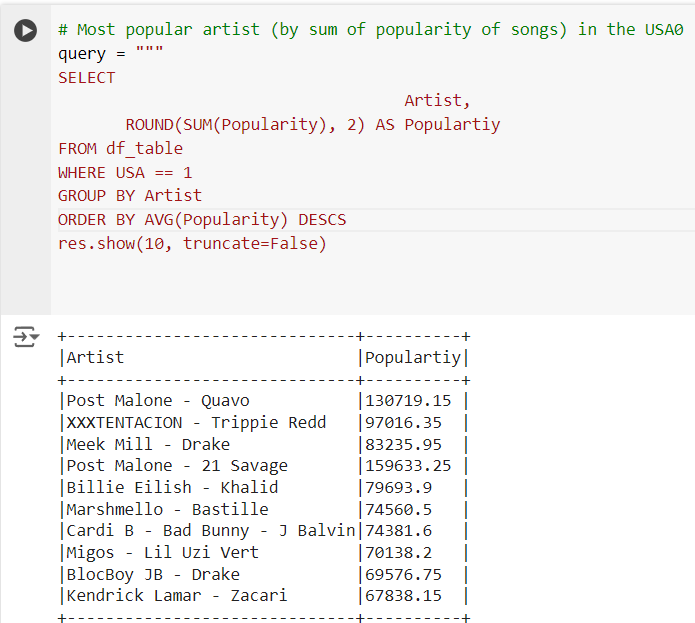
**Giải thích đoạn mã:**

* + **df.groupBy("Artist")**:Nhóm dữ liệu theo tên nghệ sĩ (Artist). Điều này có nghĩa là tất cả các bài hát của cùng một nghệ sĩ sẽ được nhóm lại với nhau.
  + **.count()**:Đếm số lượng bài hát của mỗi nghệ sĩ trong nhóm. Kết quả sẽ là số lần xuất hiện của mỗi nghệ sĩ trong dữ liệu.
  + **.orderBy("count", ascending=False)**:Sắp xếp các nghệ sĩ theo số lượng bài hát của họ từ cao đến thấp. ascending=False có nghĩa là sắp xếp theo thứ tự giảm dần.
  + **.limit(10)**:Lấy 10 nghệ sĩ đứng đầu, tức là những nghệ sĩ có số lượng bài hát cao nhất.
  + **.toPandas()**:Chuyển đổi kết quả từ một DataFrame của Spark thành một DataFrame của Pandas. Điều này cần thiết vì plotly.express (được gọi là px ở đây) làm việc với các DataFrame của Pandas.
  + **px.bar(result\_df, y='Artist', x='count', title='Most Prolific Artists')**:
  + Sử dụng plotly.express để vẽ một biểu đồ cột (bar chart).
  + y='Artist': Đặt tên nghệ sĩ trên trục y.
  + x='count': Đặt số lượng bài hát trên trục x.
  + title='Most Prolific Artists': Đặt tiêu đề cho biểu đồ là "Những Nghệ Sĩ Nổi Bật Nhất".

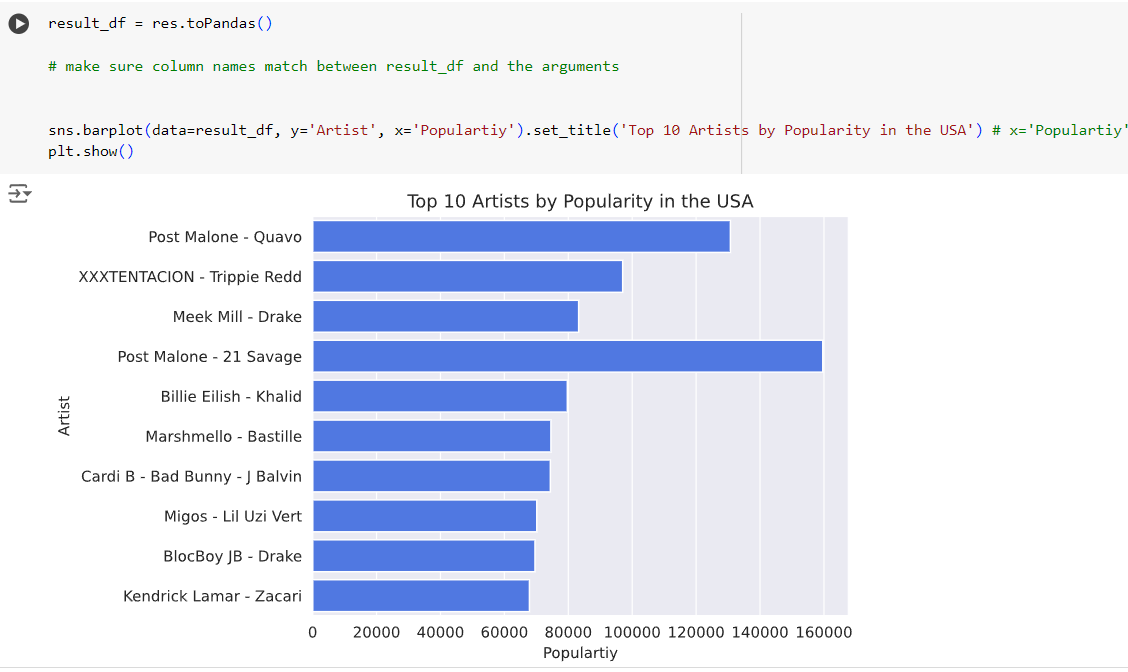


Hình 2. 9: CreateOrReplaceTempView

* + - * CreateOrReplaceTempView sẽ tạo chế độ xem tạm thời của bảng trên bộ nhớ. Nó sẽ không tồn tại, nhưng bạn có thể chạy các truy vấn SQL trên đó. Bạn luôn có thể buộc nó lưu vào bộ nhớ đệm/tồn tại với saveAsTable.

****

Hình 2. 10: Top 10 nghệ sĩ phổ biến ở USA



Hình 2. 11:Chuyển đổi dữ liệu Spark DataFrame sang Pandas DataFrame để trực quán hóa



Hình 2. 12: Bài hát nổi bật nhất (có độ phổ biến cao nhất) từ từng thập kỷ

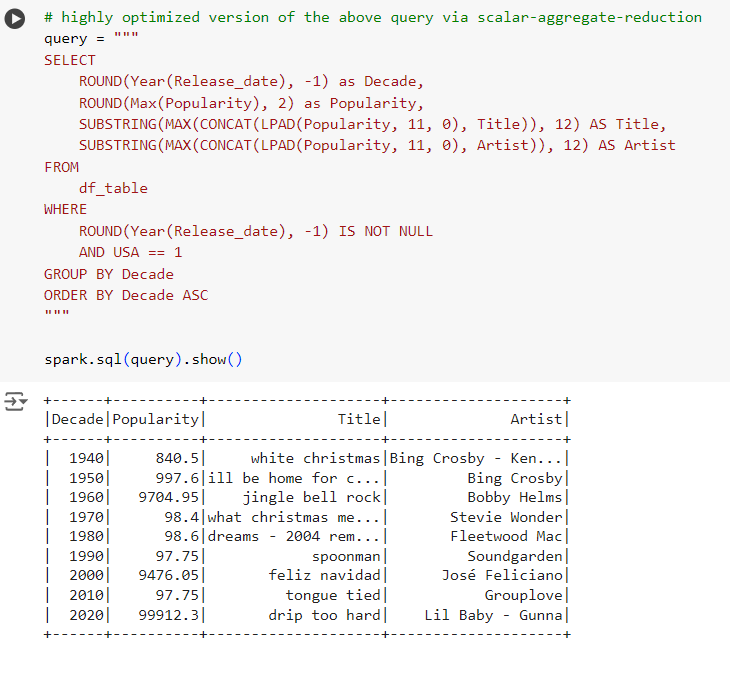
**Giải thích từng phần:**

* **ROUND(Year(Release\_date), -1) AS Decade**: Lấy năm phát hành của bài hát, làm tròn về hàng chục để xác định thập kỷ. Ví dụ, năm 1995 sẽ trở thành 1990.
* **ROUND(Popularity, 2) AS Popularity**: Làm tròn giá trị độ phổ biến của bài hát đến 2 chữ số thập phân.
* **INNER JOIN (...) AS temp**: Tạo một bảng phụ (temp) chứa độ phổ biến cao nhất (Max(Popularity)) cho từng thập kỷ từ dữ liệu chỉ từ Mỹ (USA = 1).
* **ON temp.mp = df\_table.Popularity**: Kết hợp bảng phụ temp với bảng chính df\_table dựa trên độ phổ biến, chọn các bài hát có độ phổ biến bằng giá trị tối đa từ bảng phụ.
* **ORDER BY Decade ASC, Popularity ASC**: Sắp xếp kết quả theo thập kỷ (tăng dần) và sau đó theo độ phổ biến (tăng dần) trong từng thập kỷ.

Pandas DataFrame

res.toPandas().drop\_duplicates(subset='Decade', keep="last")

* **res.toPandas()**: Chuyển đổi kết quả từ Spark DataFrame sang Pandas DataFrame để dễ dàng xử lý và trực quan hóa trong Python.
* **drop\_duplicates(subset='Decade', keep="last")**: Xóa các bản sao, giữ lại bài hát cuối cùng trong mỗi thập kỷ (theo thứ tự độ phổ biến tăng dần).



Hình 2. 13: Tối ưu hóa việc lấy bài hát nổi bật nhất trong mỗi thập kỷ

* Truy vấn này tối ưu hóa việc lấy bài hát nổi bật nhất trong mỗi thập kỷ bằng cách sử dụng các hàm tổng hợp và chuỗi để rút gọn số lượng phép toán cần thực hiện



Hình 2. 14: 5 thể loại nhạc phổ biến nhất



Hình 2. 15: Thể loại phổ biến nhất cho mỗi thập kỷ

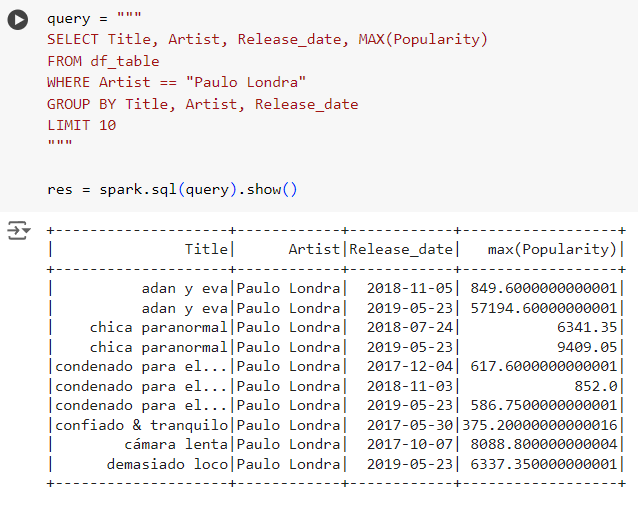
**Giải thích từng phần:**

* **ROUND(Year(Release\_date), -1) AS Decade**: Tính thập kỷ từ năm phát hành bài hát bằng cách làm tròn về hàng chục. Ví dụ, năm 1995 trở thành 1990.
* **Genre**:Chọn cột thể loại nhạc từ bảng dữ liệu.
* **COUNT(Genre) AS counts**: Đếm số lượng bài hát trong mỗi thể loại nhạc và gán tên cột đếm là counts.
* **FROM df\_table**: Chỉ định bảng dữ liệu df\_table làm nguồn dữ liệu.
* **WHERE ROUND(Year(Release\_date), -1) IS NOT NULL**: Lọc các bài hát có năm phát hành không bị NULL sau khi làm tròn về thập kỷ.
* **GROUP BY Decade, Genre**: Nhóm dữ liệu theo thập kỷ và thể loại nhạc để tính toán số lượng bài hát trong từng nhóm.
* **ORDER BY COUNT(Genre) DESC**: Sắp xếp kết quả theo số lượng bài hát (counts) theo thứ tự giảm dần, để thể loại phổ biến nhất trong từng thập kỷ xuất hiện đầu tiên.

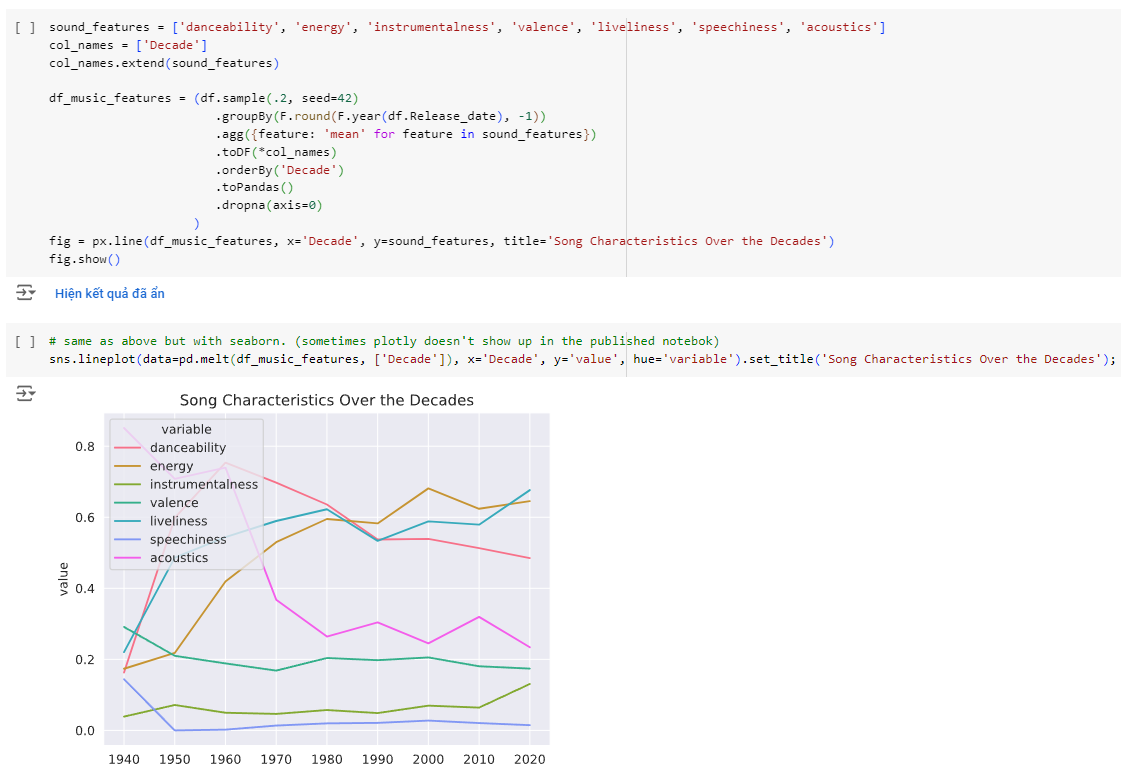
Xử lý dữ liệu với Spark:

* **spark.sql(query)**: Thực hiện truy vấn SQL trên bảng dữ liệu df\_table.
* **.dropDuplicates(subset=['Decade'])**: Xóa các bản sao, chỉ giữ lại hàng đầu tiên cho mỗi thập kỷ. Trong trường hợp này, nó giữ lại thể loại phổ biến nhất trong mỗi thập kỷ do sắp xếp theo số lượng giảm dần.
* **.orderBy('Decade')**: Sắp xếp kết quả theo thập kỷ để có cái nhìn theo thứ tự thời gian.
* **.show()**: Hiển thị kết quả.

Chuyển đổi kết quả từ Spark DataFrame sang Pandas DataFrame và loại bỏ các bản sao để chỉ giữ lại hàng đầu tiên cho mỗi thập kỷ.



Hình 2. 16 : 10 bài hát phổ biến nhất của nghệ sĩ "Paulo Londra”

****

Hình 2. 17 : Giá trị trung bình các đặc điểm âm thanh của các bài hát qua các thập kỷ

**2.2.5 Các kỹ thuật xử lý , phân tích dữ liệu và  kỹ thuật thống kê sử dụng để giảm chiều dữ liệu PCA:**

A screen shot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

Hình 2. 18: Truy vấn để thu thập dữ liệu về bài hát từ Mỹ và thể loại rap

**Giải thích từng phần:**

* + - **Truy vấn dữ liệu từ Mỹ**

**A screenshot of a computer code

Description automatically generated**

Hình 2. 19 : Truy vấn dữ liệu từ Mỹ

* **query\_usa**: Truy vấn SQL để chọn tên bài hát (Title), nghệ sĩ (Artist), và các đặc điểm số (numerical\_features) từ bảng df\_table nơi cột USA có giá trị là 1.
* **.format(', '.join(numerical\_features))**: Chèn danh sách các đặc điểm số vào truy vấn.
* **spark.sql(query\_usa)**: Thực hiện truy vấn SQL trên bảng df\_table.
* **.sample(.1)**: Chọn ngẫu nhiên 10% dữ liệu từ kết quả truy vấn.
* **.dropna()**: Loại bỏ các hàng có giá trị NaN.
* **.toPandas()**: Chuyển đổi kết quả thành Pandas DataFrame.
  + - **Truy vấn Dữ Liệu từ Thể Loại Rap**

A screenshot of a computer code

Description automatically generated

Hình 2. 20 : Truy vấn Dữ Liệu từ Thể Loại Rap

* **query\_rap**: Truy vấn SQL để chọn tên bài hát (Title), nghệ sĩ (Artist), và các đặc điểm số (numerical\_features) từ bảng df\_table nơi cột rap có giá trị là 1.
* **.format(', '.join(numerical\_features))**: Chèn danh sách các đặc điểm số vào truy vấn.
* **spark.sql(query\_rap)**: Thực hiện truy vấn SQL trên bảng df\_table.
* **.sample(.1)**: Chọn ngẫu nhiên 10% dữ liệu từ kết quả truy vấn.
* **.dropna()**: Loại bỏ các hàng có giá trị NaN.
* **.toPandas()**: Chuyển đổi kết quả thành Pandas DataFrame.
* **df\_rap\_songs.head()**: Hiển thị 5 hàng đầu tiên của DataFrame df\_rap\_songs.

**Lưu Ý Quan Trọng :**

* **toPandas()**: Chuyển đổi toàn bộ kết quả từ Spark DataFrame sang Pandas DataFrame có thể không hiệu quả với tập dữ liệu lớn vì nó có thể gây quá tải bộ nhớ. Thay vào đó, việc thực hiện mẫu dữ liệu (.sample()) trước khi chuyển đổi là một cách tiếp cận tốt hơn.
* **dropna()**: Loại bỏ các hàng có giá trị NaN sau khi mẫu dữ liệu có thể dẫn đến việc mất dữ liệu quan trọng. Cần cân nhắc việc xử lý giá trị thiếu theo cách khác trước khi loại bỏ.
  + Check out the cosine similiarty of song feature vectors

A screenshot of a computer program

Description automatically generated

Hình 2. 21 : Tính toán tương đồng cosine giữa bài hát rap và bài hát Mỹ từ các tập dữ liệu khác nhau

**Giải thích mã :**

* + **Nhập Các Thư Viện Cần Thiết**

from scipy import spatial

from sklearn.preprocessing import Normalizer

from sklearn.preprocessing import MinMaxScaler

* **scipy.spatial**: Thư viện dùng để tính toán khoảng cách giữa các vector, trong trường hợp này là khoảng cách cosine.
* **sklearn.preprocessing.Normalizer**: Dùng để chuẩn hóa dữ liệu, biến đổi các vector thành đơn vị độ dài.
* **sklearn.preprocessing.MinMaxScaler**: Dùng để chuyển đổi dữ liệu về khoảng [0, 1].

#### Khởi Tạo Các Công Cụ Tiền Xử Lý MinMaxScaler()

scaler = MinMaxScaler()

transformer = Normalizer()

* **MinMaxScaler**: Được sử dụng để đưa dữ liệu về khoảng [0, 1].
* **Normalizer**: Được sử dụng để chuẩn hóa dữ liệu sao cho tổng bình phương các giá trị của mỗi vector bằng 1.
  + **Chuyển Đổi Dữ Liệu**

scaled\_usa\_df = scaler.fit\_transform(df\_usa\_songs.iloc[:, 2:])

scaled\_rap\_df = scaler.fit\_transform(df\_rap\_songs.iloc[:, 2:])

* **df\_usa\_songs.iloc[:, 2:]**: Chọn tất cả các cột bắt đầu từ cột thứ ba (bỏ qua các cột Title và Artist) trong DataFrame df\_usa\_songs.
* **scaler.fit\_transform()**: Chuyển đổi dữ liệu bằng cách sử dụng MinMaxScaler, đưa dữ liệu về khoảng [0, 1].
  + **Tính Toán Độ Tương Đồng Cosine**

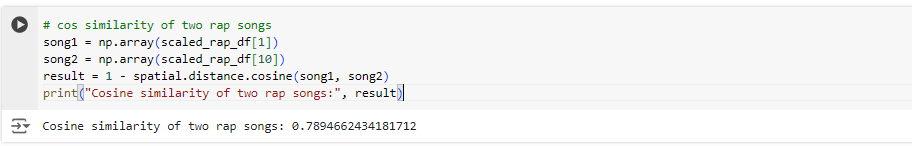
song1 = np.array(scaled\_rap\_df[1])

song2 = np.array(scaled\_usa\_df[2])

result = 1 - spatial.distance.cosine(song1, song2)

print("Cosine similarity of a rap and a usa song:", result)

* **scaled\_rap\_df[1]**: Lấy dòng thứ hai của scaled\_rap\_df (bài hát rap).
* **scaled\_usa\_df[2]**: Lấy dòng thứ ba của scaled\_usa\_df (bài hát Mỹ).
* **spatial.distance.cosine(song1, song2)**: Tính khoảng cách cosine giữa hai vector song1 và song2. Khoảng cách cosine là 1 trừ đi độ tương đồng cosine, vì vậy kết quả tính toán được lấy là 1 - cosine\_distance.
* **print()**: Hiển thị độ tương đồng cosine giữa bài hát rap và bài hát Mỹ.
  + - * **Kết quả cho thấy độ tương đồng :0.9381489410027772**



Hình 2. 22 Độ tương đồng cosine giữa hai bài hát rap.

* + **Dimensionality reduction( Giảm kích thước)**



Hình 2. 23 Tỷ lệ phương sai giải thích được bởi mỗi thành phần chính và ảnh trực quan hóa

**Giải thích mã** :

Đoạn mã trên thực hiện các bước chuẩn bị dữ liệu và phân tích chính thành phần (PCA) cho hai tập dữ liệu bài hát rap và bài hát Mỹ. Đây là các bước chi tiết:

* + Gán Các Cột One-Hot Encoding (OHE) Cho Các Thể Loại

df\_rap\_songs = df\_rap\_songs.assign(is\_rap=1, is\_usa=0)

df\_usa\_songs = df\_usa\_songs.assign(is\_rap=0, is\_usa=1)

* **assign()**: Thêm các cột mới is\_rap và is\_usa vào các DataFrame df\_rap\_songs và df\_usa\_songs. Cột is\_rap và is\_usa là các cột mã hóa một nóng (OHE) cho thể loại.
  + **is\_rap=1, is\_usa=0**: Đánh dấu bài hát rap với is\_rap là 1 và is\_usa là 0.
  + **is\_rap=0, is\_usa=1**: Đánh dấu bài hát Mỹ với is\_rap là 0 và is\_usa là 1.
  + Kết Hợp Các DataFrame

df\_rap\_and\_usa = pd.concat([df\_rap\_songs, df\_usa\_songs])

* **pd.concat()**: Kết hợp các DataFrame df\_rap\_songs và df\_usa\_songs thành một DataFrame duy nhất df\_rap\_and\_usa.
  + Tiền Xử Lý Dữ Liệu:

X = scaler.fit\_transform(df\_rap\_and\_usa.iloc[:, 2:])

* **scaler.fit\_transform()**: Chuyển đổi dữ liệu trong các cột từ cột thứ ba trở đi (bỏ qua Title, Artist, is\_rap, is\_usa) bằng cách sử dụng MinMaxScaler. Điều này đưa tất cả các giá trị về khoảng [0, 1].

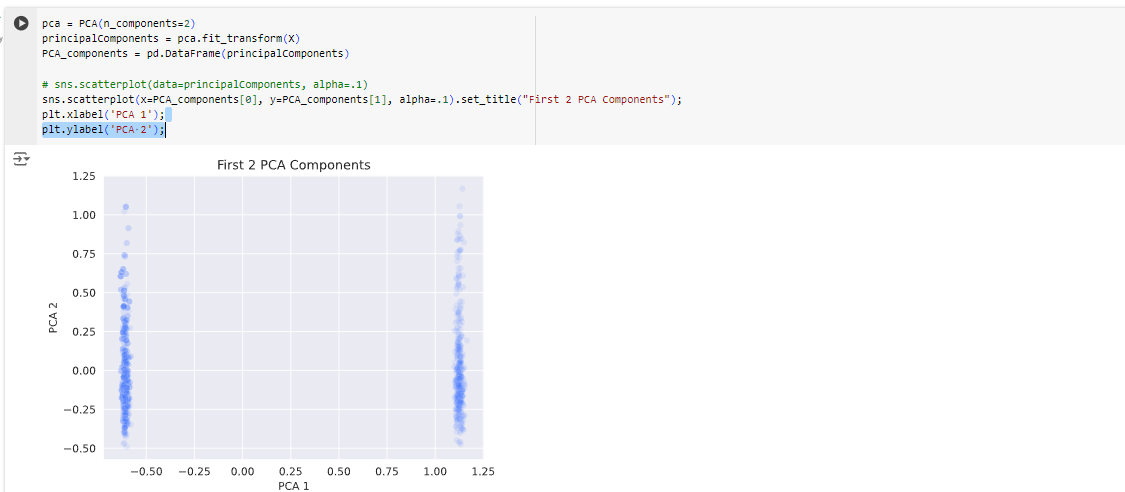
#### Phân Tích Chính Thành Phần (PCA

pca = PCA(n\_components=10)

pca.fit(X)

print(pca.explained\_variance\_ratio\_)

* **PCA(n\_components=10)**: Khởi tạo PCA để giảm số lượng thành phần xuống còn 10.
* **pca.fit(X)**: Áp dụng PCA trên dữ liệu đã chuẩn hóa.
* **print(pca.explained\_variance\_ratio\_)**: In ra tỷ lệ phương sai giải thích được bởi mỗi thành phần chính. Đây là thông số cho biết phần trăm thông tin trong dữ liệu được giải thích bởi từng thành phần.
  + **Ý Nghĩa**
* **OHE Encoding**: Việc thêm cột is\_rap và is\_usa giúp mã hóa các thể loại bài hát để PCA có thể xử lý dữ liệu theo cách mà các thể loại cũng được tính đến trong phân tích.
* **PCA**: Giảm số lượng biến trong dữ liệu từ nhiều thành phần thành 10 thành phần chính, giúp giảm độ phức tạp của dữ liệu và làm rõ các yếu tố chính ảnh hưởng đến dữ liệu.
  + **Kết Quả:**
* **pca.explained\_variance\_ratio\_**: Đây là mảng chứa tỷ lệ phương sai giải thích được bởi mỗi thành phần chính. Các giá trị này cho biết mỗi thành phần chính đóng góp bao nhiêu phần trăm vào tổng phương sai của dữ liệu.

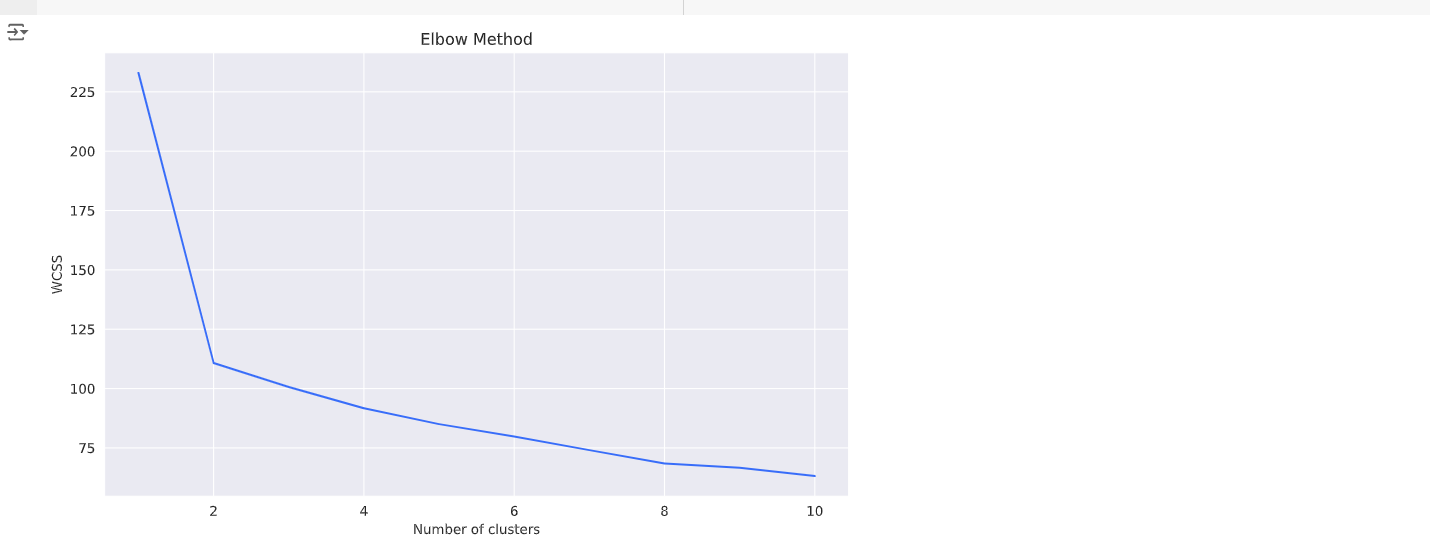


Hình 2. 24 :Biểu Đồ Phân Tán

* + **Ý Nghĩa: Biểu Đồ Phân Tán**: Hiển thị sự phân phối của dữ liệu trong không gian của hai thành phần chính đầu tiên. Giúp hiểu cách dữ liệu phân bố và xác định các nhóm hoặc mẫu đặc biệt.

**2.2.6 Xây dựng mô hình Kmeans Cluster**





Hình 2. 25: ****Tìm số cụm tối ưu**** bằng phương pháp Elbow

Giải thích chi tiết cho từng phần của đoạn mã:

1. Import các thư viện cần thiết

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import seaborn as sns

* **sklearn.cluster.KMeans**: Thư viện sklearn cung cấp hàm KMeans để thực hiện thuật toán phân cụm K-Means. Đây là một trong những phương pháp phổ biến nhất để phân nhóm dữ liệu thành các cụm dựa trên đặc trưng.
* **matplotlib.pyplot**: Thư viện matplotlib được sử dụng để tạo các biểu đồ. pyplot cung cấp các hàm để dễ dàng vẽ biểu đồ.
* **seaborn**: Seaborn là một thư viện xây dựng trên matplotlib, cung cấp các công cụ mạnh mẽ và dễ sử dụng để vẽ biểu đồ.

2. Chuẩn bị dữ liệu

# Sử dụng dữ liệu đã chuẩn hóa trước đó (X\_scaled)

# Giả sử X\_scaled đã được chuẩn hóa và là đầu vào cho mô hình

# Nếu X\_scaled chưa chuẩn hóa thì thực hiện chuẩn hóa lại

# X\_scaled = scaler.fit\_transform(X)

* **Chuẩn hóa dữ liệu (Normalization)**: Việc chuẩn hóa dữ liệu là một bước quan trọng khi sử dụng K-Means, bởi vì K-Means sử dụng khoảng cách Euclidean để đo lường sự tương đồng giữa các điểm. Nếu dữ liệu chưa được chuẩn hóa, khoảng cách giữa các điểm có thể bị chi phối bởi các đặc trưng có giá trị lớn hơn, làm sai lệch kết quả phân cụm.
* **X\_scaled**: Đây là dữ liệu đầu vào đã được chuẩn hóa, thường là một mảng numpy chứa các giá trị đặc trưng đã được chuẩn hóa.

3. Tìm số cụm tối ưu bằng Elbow Method

wcss = [] # within-cluster sum of squares

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', max\_iter=300, n\_init=10, random\_state=42)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

* **wcss**: Biến wcss (within-cluster sum of squares) lưu trữ tổng bình phương khoảng cách giữa các điểm trong cùng một cụm. Đây là một chỉ số đo lường mức độ chặt chẽ của các cụm.
* **Vòng lặp**: Vòng lặp chạy từ 1 đến 10 cụm (n\_clusters=i), huấn luyện mô hình K-Means với số lượng cụm khác nhau và lưu trữ giá trị inertia\_ (tức là WCSS) của từng mô hình vào danh sách wcss.
* **KMeans**:
  + n\_clusters=i: Số lượng cụm cho mỗi lần lặp.
  + init='k-means++': Phương pháp khởi tạo các trung tâm cụm để tăng hiệu suất phân cụm.
  + max\_iter=300: Số lần lặp tối đa cho thuật toán K-Means để hội tụ.
  + n\_init=10: Số lần thuật toán K-Means được khởi động lại với các trung tâm cụm khác nhau.
  + random\_state=42: Đảm bảo tính ngẫu nhiên có thể tái tạo được (kết quả có thể giống nhau mỗi khi chạy lại).

4. Vẽ biểu đồ Elbow

plt.figure(figsize=(10, 6))

sns.lineplot(x=range(1, 11), y=wcss)

plt.title('Elbow Method')

plt.xlabel('Number of clusters')

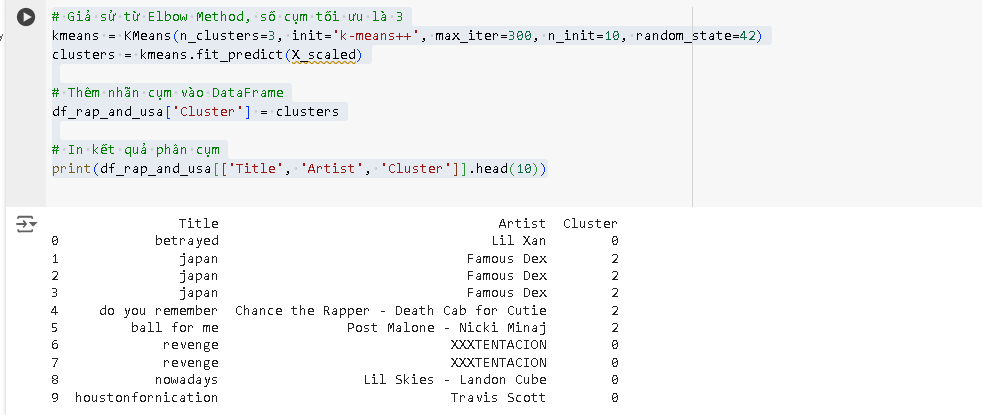
plt.ylabel('WCSS')

plt.show()

* **Biểu đồ Elbow**: Biểu đồ này cho phép bạn quan sát sự thay đổi của WCSS theo số lượng cụm (n\_clusters). Trục x biểu diễn số lượng cụm, và trục y biểu diễn giá trị WCSS.
* **Tìm "Elbow" (khuỷu tay)**: Phương pháp Elbow giúp xác định số cụm tối ưu bằng cách tìm vị trí mà WCSS giảm mạnh và sau đó chậm lại (hình dạng như khuỷu tay). Số cụm tương ứng với điểm "khuỷu tay" là lựa chọn tốt nhất cho số cụm trong mô hình của bạn.
* **sns.lineplot**: Hàm này từ thư viện Seaborn được sử dụng để vẽ biểu đồ đường.

**Tổng kết:**

* **Chuẩn bị dữ liệu**: Đảm bảo rằng dữ liệu được chuẩn hóa để tránh ảnh hưởng của các đặc trưng có giá trị lớn.
* **Tìm số cụm tối ưu**: Phương pháp Elbow giúp xác định số lượng cụm tối ưu cho mô hình K-Means, đảm bảo phân cụm hiệu quả và chính xác.
* **Biểu đồ Elbow**: Giúp trực quan hóa kết quả và đưa ra quyết định về số lượng cụm phù hợp cho bài toán của bạn.



Hình 2. 26: K-Means Clustering thể loại rap và usa

Giải thích chi tiết cho từng phần của đoạn mã:

1. Khởi tạo và áp dụng K-Means với số cụm tối ưu

kmeans = KMeans(n\_clusters=3, init='k-means++', max\_iter=300, n\_init=10, random\_state=42)

clusters = kmeans.fit\_predict(X\_scaled)

* **n\_clusters=3**: Giả sử từ Elbow Method, số cụm tối ưu đã được xác định là 3. Điều này có nghĩa là mô hình sẽ cố gắng phân loại các bài hát vào 3 nhóm (cụm) khác nhau.
* **init='k-means++'**: Phương pháp khởi tạo trung tâm cụm, giúp chọn các điểm khởi tạo tốt hơn, dẫn đến sự hội tụ nhanh hơn và phân cụm hiệu quả hơn.
* **max\_iter=300**: Giới hạn số lần lặp của thuật toán K-Means để tìm ra vị trí tối ưu cho các trung tâm cụm.
* **n\_init=10**: K-Means sẽ chạy 10 lần với các trung tâm cụm khởi tạo ngẫu nhiên khác nhau, và chọn lần có kết quả tốt nhất (dựa trên WCSS thấp nhất).
* **random\_state=42**: Đảm bảo tính ngẫu nhiên có thể tái tạo được, nghĩa là kết quả sẽ giống nhau mỗi khi bạn chạy lại mã này.
* **fit\_predict(X\_scaled)**: Hàm fit\_predict vừa huấn luyện mô hình trên dữ liệu X\_scaled, vừa trả về nhãn cụm (clusters) cho từng mẫu trong tập dữ liệu. Mỗi bài hát sẽ được gán một nhãn cụm từ 0 đến 2 (vì có 3 cụm).

2. Thêm nhãn cụm vào DataFrame

df\_rap\_and\_usa['Cluster'] = clusters

**Thêm nhãn cụm**: Sau khi thực hiện phân cụm, chúng ta thêm nhãn cụm (Cluster) vào DataFrame gốc (df\_rap\_and\_usa) để gắn kết từng bài hát với cụm mà nó thuộc về. Cột Cluster sẽ chứa các giá trị từ 0 đến 2, đại diện cho các cụm mà mỗi bài hát được phân vào.

3. In kết quả phân cụm

print(df\_rap\_and\_usa[['Title', 'Artist', 'Cluster']].head(10))

* **In kết quả**: Đoạn mã này in ra bảng dữ liệu chứa tiêu đề bài hát (Title), nghệ sĩ (Artist), và cụm mà mỗi bài hát được gán (Cluster). head(10) giới hạn kết quả in ra chỉ 10 dòng đầu tiên của bảng.

Tổng kết:

* **K-Means Clustering**: Phương pháp phân cụm được sử dụng để phân loại các bài hát thành 3 nhóm dựa trên các đặc trưng âm nhạc của chúng.
* **Gán nhãn cụm**: Mỗi bài hát được gán vào một cụm, giúp xác định xem các bài hát nào có các đặc điểm âm nhạc tương tự nhau.
* **Hiển thị kết quả**: Đoạn mã cuối cùng hiển thị một phần kết quả của quá trình phân cụm, cho thấy mỗi bài hát thuộc về cụm nào.



Hình 2. 27 Biểu đồ phân tán sử dụng mô hình K-Means đã phân chia các bài hát thành các cụm riêng biệt

giải thích chi tiết cho từng phần của đoạn mã:

1. Tính toán số lượng bài hát trong mỗi cụm

print(df\_rap\_and\_usa['Cluster'].value\_counts())

* **df\_rap\_and\_usa['Cluster']**: Đây là cột chứa nhãn cụm mà mỗi bài hát đã được gán vào sau khi thực hiện K-Means Clustering.
* **value\_counts()**: Hàm này đếm số lượng giá trị xuất hiện trong cột Cluster. Cụ thể, nó sẽ đếm số lượng bài hát thuộc về mỗi cụm (0, 1, hoặc 2).
* **Kết quả in ra**:
  + 0 117: Có 117 bài hát được gán vào cụm 0.
  + 1 92: Có 92 bài hát được gán vào cụm 1.
  + 2 58: Có 58 bài hát được gán vào cụm 2.

Điều này cho thấy, cụm 0 có nhiều bài hát nhất, tiếp theo là cụm 1 và cụm 2.

2. Vẽ biểu đồ để trực quan hóa các cụm

plt.figure(figsize=(10, 6))

sns.scatterplot(x=X\_scaled[:, 0], y=X\_scaled[:, 1], hue=clusters, palette='viridis')

plt.title('K-Means Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

* **plt.figure(figsize=(10, 6))**: Tạo một hình vẽ mới với kích thước 10x6 inch. Điều này giúp điều chỉnh kích thước của biểu đồ sao cho dễ nhìn hơn.
* **sns.scatterplot()**: Vẽ biểu đồ phân tán (scatter plot) bằng thư viện Seaborn.
  + x=X\_scaled[:, 0]: Trục x của biểu đồ sẽ biểu diễn giá trị của đặc trưng đầu tiên (cột đầu tiên) trong dữ liệu đã được chuẩn hóa.
  + y=X\_scaled[:, 1]: Trục y của biểu đồ sẽ biểu diễn giá trị của đặc trưng thứ hai (cột thứ hai) trong dữ liệu đã được chuẩn hóa.
  + hue=clusters: Màu sắc của các điểm trên biểu đồ được xác định bởi giá trị cụm (Cluster). Điều này giúp phân biệt các bài hát thuộc các cụm khác nhau bằng các màu sắc khác nhau.
  + palette='viridis': Chỉ định bảng màu viridis để vẽ biểu đồ, đây là một bảng màu gradient từ xanh lá cây sang tím.
* **plt.title()**: Đặt tiêu đề cho biểu đồ là "K-Means Clustering".
* **plt.xlabel('Feature 1')**: Đặt nhãn cho trục x là "Feature 1" (đặc trưng 1).
* **plt.ylabel('Feature 2')**: Đặt nhãn cho trục y là "Feature 2" (đặc trưng 2).
* **plt.show()**: Hiển thị biểu đồ.

Kết quả và Ý nghĩa

* **Phân bố các bài hát trong các cụm**: Biểu đồ phân tán cho phép bạn trực quan hóa cách các bài hát được nhóm lại thành các cụm dựa trên hai đặc trưng đầu tiên trong dữ liệu đã chuẩn hóa. Mỗi màu sắc trên biểu đồ đại diện cho một cụm khác nhau, và bạn có thể quan sát các điểm dữ liệu (bài hát) tập trung thành các nhóm riêng biệt như thế nào.
* **Số lượng bài hát trong mỗi cụm**: Kết quả của value\_counts() cung cấp thông tin về sự phân bố số lượng bài hát trong từng cụm, giúp bạn đánh giá sự cân bằng của các cụm và xem cụm nào có nhiều bài hát hơn.

Biểu đồ phân tán là công cụ trực quan mạnh mẽ giúp bạn thấy rõ cách các cụm được hình thành và sự khác biệt giữa các nhóm bài hát dựa trên các đặc trưng được chọn. Điều này có thể cung cấp những hiểu biết giá trị về cấu trúc của dữ liệu và cách mà mô hình K-Means đã phân chia các bài hát thành các cụm riêng biệt.

**CHƯƠNG III : KẾT LUẬN**

## 3.1 Kết quả đạt được :

Trong dự án này, chúng tôi đã sử dụng Apache Spark để phân tích và trực quan hóa dữ liệu thị trường âm nhạc toàn cầu. Các kết quả chính đạt được bao gồm:

* **Phân Tích Thể Loại Nhạc Phổ Biến**: Chúng tôi đã xác định các thể loại nhạc phổ biến nhất trên toàn cầu và trong các quốc gia cụ thể. Dữ liệu cho thấy thể loại hip hop rất phổ biến ở nhiều quốc gia, trong khi các thể loại khác như pop và electronic có sự phổ biến đáng kể ở các khu vực khác.
* **Đánh Giá Xu Hướng Theo Thập Kỷ**: Sử dụng PCA để giảm số chiều và phân tích các đặc điểm âm nhạc theo các thập kỷ, chúng tôi đã tạo ra biểu đồ mô tả sự thay đổi của các đặc điểm âm nhạc qua các thập kỷ, như danceability, energy, và valence.
* **So Sánh Giữa Các Thể Loại**: Chúng tôi đã so sánh các bài hát rap với các bài hát Mỹ (USA) để hiểu sự khác biệt về đặc điểm âm nhạc giữa hai thể loại này. Tỷ lệ tương đồng cosine giữa các bài hát từ hai thể loại này đã được tính toán và phân tích.
* **Phân Tích Thành Phần Chính (PCA)**: Sử dụng PCA, chúng tôi đã giảm số lượng biến và trực quan hóa các thành phần chính đầu tiên, giúp hiểu được sự phân bố dữ liệu và các yếu tố chính ảnh hưởng đến các bài hát.
* **Biểu đồ phân tán K-Means Clustering**: Biểu đồ phân tán cho thấy các bài hát được phân nhóm thành các cụm khác nhau dựa trên hai đặc trưng âm nhạc chính, mỗi cụm được hiển thị bằng màu sắc khác nhau, giúp bạn dễ dàng nhận diện sự phân chia giữa các nhóm

## 3.2 Hiệu suất mô hình :

* **Tốc Độ Xử Lý Dữ Liệu**: Sử dụng Apache Spark, chúng tôi đã có khả năng xử lý và phân tích khối lượng dữ liệu lớn một cách nhanh chóng và hiệu quả. Spark's DataFrame API cho phép thực hiện các phép toán phức tạp trên dữ liệu với tốc độ cao.
* **Tính Chính Xác**: Các phân tích và trực quan hóa đã được thực hiện với độ chính xác cao nhờ vào việc sử dụng các kỹ thuật phân tích dữ liệu tiên tiến và công cụ mạnh mẽ như PCA.
* **Khả Năng Mở Rộng**: Dự án có khả năng mở rộng để phân tích thêm các dữ liệu lớn khác hoặc thực hiện các phân tích nâng cao hơn, nhờ vào khả năng phân tán của Spark.
* **Tính chính xác của phân cụm**: K-Means Clustering đã phân chia dữ liệu thành ba cụm khác nhau. Sự phân chia này cho phép chúng ta nhìn nhận được cấu trúc tiềm ẩn trong dữ liệu âm nhạc. Tuy nhiên, hiệu suất của mô hình không thể được đánh giá hoàn toàn chỉ dựa trên số lượng bài hát trong từng cụm hoặc biểu đồ phân tán. Đánh giá thêm về sự hợp lý của các cụm về mặt nội dung âm nhạc và ý nghĩa của từng cụm là cần thiết.

## 3.3 GỢI Ý SẢN PHẨM CHO KHÁCH HÀNG :

**Cá Nhân Hóa Playlist**: Dựa trên phân tích thể loại và đặc điểm âm nhạc, chúng tôi có thể gợi ý các playlist cá nhân hóa cho người dùng dựa trên sở thích âm nhạc của họ. Ví dụ, người dùng yêu thích thể loại hip hop có thể được gợi ý các bài hát và album nổi bật trong thể loại này.

**Chiến Lược Tiếp Thị Toàn Cầu**: Các công ty có thể sử dụng phân tích thị trường âm nhạc toàn cầu để điều chỉnh chiến lược tiếp thị của họ, nhằm nhắm mục tiêu các thị trường cụ thể với các sản phẩm âm nhạc phù hợp với sở thích địa phương.

**Dự Đoán Xu Hướng Âm Nhạc**: Dựa trên các đặc điểm âm nhạc qua các thập kỷ, các nhà sản xuất âm nhạc và các nhà phân phối có thể dự đoán các xu hướng âm nhạc mới và chuẩn bị cho các xu hướng này.

* 1. **HẠN CHẾ :**

**Chất Lượng Dữ Liệu**: Một số dữ liệu có thể bị thiếu hoặc không chính xác, ảnh hưởng đến kết quả phân tích. Các bài hát có thể thiếu thông tin về thể loại hoặc các đặc điểm âm nhạc.

**Khả Năng Tinh Chỉnh Mô Hình**: Mặc dù PCA giúp giảm số lượng chiều, nhưng việc chọn số lượng thành phần chính phù hợp có thể là một thách thức. Cần phải cân nhắc kỹ lưỡng để đảm bảo rằng các thành phần chính lựa chọn giải thích được phần lớn phương sai trong dữ liệu.

**Khả Năng Đúng Đắn Của So Sánh**: Các phép toán cosine similarity có thể không luôn phản ánh đúng sự khác biệt thực tế giữa các bài hát từ các thể loại khác nhau, vì nó chỉ đo sự tương đồng về đặc điểm âm nhạc mà không tính đến các yếu tố khác như bối cảnh văn hóa.

**Hiệu Suất Hệ Thống**: Trong các trường hợp dữ liệu rất lớn, mặc dù Spark có khả năng mở rộng tốt, nhưng yêu cầu về tài nguyên hệ thống và thời gian xử lý có thể vẫn là một yếu tố hạn chế.

# TÀI LIỆU THAM KHẢO

1.dữ liệu: https://www.kaggle.com/code/fusshandschuhe/spotify-eda-with-pyspark-and-clustering/input

2.pyspark: https://spark.apache.org/docs/latest/api/python/index.html