

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**--------------------**

****

**BÁO CÁO ĐỒ ÁN**

Môn học: Mạng Xã Hội

Học kỳ II (2019-2020)

**TÌM HIỂU VÀ ỨNG DỤNG NEO4J VÀO**

**PHÂN TÍCH DỮ LIỆU BÁN LẺ CỦA UK**

**Giáo viên hướng dẫn: Nguyễn Thị Kim Phụng**

**Sinh viên thực hiện:**

Nguyễn Đình Văn Khoa - 14520426

Phạm Quốc Cường – 16520157

Nguyễn Văn Trí - 16521287

Lớp: IS353.K21

TP. Hồ Chí Minh, ngày 23 tháng 06 năm 2020

**LỜI CẢM ƠN**

Lời đầu tiên, nhóm xin gửi lời cảm ơn chân thành đến quý Thầy Cô trường Đại học Công nghệ thông tin đặc biệt là quý Thầy Cô Khoa Hệ thống thông tin - những người đã dùng tri thức và tâm huyết của mình để truyền đạt cho chúng em vốn kiến thức vô cùng quý báu trong khoảng thời gian học tập tại trường. Những kiến thức mà Thầy Cô truyền đạt là bước đệm quan trọng giúp chúng em có thể hoàn thành đề tài tốt hơn.

Nhóm xin gửi lời cảm ơn đặc biệt chân thành tới Cô Nguyễn Thị Kim Phụng - giảng viên môn Mạng xã hội đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn nhóm trong suốt quá trình làm đồ án môn học. Nhờ đó, chúng em đã tiếp thu được nhiều kiến thức bổ ích trong việc vận dụng cũng như kỹ năng làm đồ án.

Trải qua thời gian một học kỳ thực hiện đề tài. Với sự hướng dẫn tận tình cùng những đóng góp quý báu của Cô và các bạn giúp nhóm hoàn thành tốt hơn báo cáo môn học của mình. Bên cạnh việc vận dụng những kiến thức được học trên lớp đồng thời kết hợp với việc học hỏi và tìm hiểu những kiến thức mới. Từ đó, nhóm đã vận dụng tối đa những gì đã tiếp thu được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, không tránh khỏi những sai sót. Dó đó, rất mong nhận được những sự góp ý từ Cô nhằm giúp nhóm hoàn thiện những kiến thức đã học tập và cũng là hành trang để nhóm thực hiện tiếp các đề tài khác trong tương lai.

Xin chân thành cảm ơn quý Thầy Cô và các bạn!

*Nhóm sinh viên thực hiện*

Mục Lục

[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI 4](#_Toc45353270)

[I. Lý do chọn đề tài 4](#_Toc45353271)

[II. Mô tả bài toán: 4](#_Toc45353272)

[CHƯƠNG 2: TÌM HIỂU VỀ NEO4J 5](#_Toc45353273)

[I. Neo4J là gì 5](#_Toc45353274)

[II. Cách tổ chức dữ liệu trong Neo4J 6](#_Toc45353275)

[III. Các điểm nổi bật 7](#_Toc45353276)

[IV. Cài đặt Neo4J 9](#_Toc45353277)

[CHƯƠNG 3: ỨNG DỤNG NEO4J VÀO PHÂN TÍCH MẠNG XÃ HỘI: 13](#_Toc45353278)

[I. Dataset 13](#_Toc45353279)

[II. Import dữ liệu 14](#_Toc45353280)

[III. Các thuật toán được sử dụng 18](#_Toc45353281)

[1. Jaccard Similarity 18](#_Toc45353282)

[2. Louvain 18](#_Toc45353283)

[3. Page Rank 20](#_Toc45353284)

[4. Closeness 21](#_Toc45353285)

[IV. Phân tích dữ liệu 21](#_Toc45353286)

[1. Gom cụm các khách hàng tương đồng với nhau: 21](#_Toc45353287)

[2. Tính similarity , closeness của các cụm cộng đồng: 26](#_Toc45353288)

[3. Thực hiện các câu truy vấn: 29](#_Toc45353289)

[CHƯƠNG 4: KẾT LUẬN 35](#_Toc45353290)

[1. Những điều đã làm được: 35](#_Toc45353291)

[2. Hạn chế: 35](#_Toc45353292)

[CHƯƠNG 5: TÀI LIỆU THAM KHẢO 36](#_Toc45353293)

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

## Lý do chọn đề tài

* Việc kinh doanh bán lẻ là xu thế phát triển mạnh và điều không thể thiếu trong thời kì ngày nay. Các nhà bán lẻ đưa sản phẩm của mình tiếp cận với khách hàng thông qua nhiều hình thức, điển hình như thông qua các trang thương mại điện tử.
* Nhóm em lựa chọn dữ liệu bán lẻ ở UK do nhóm tác giả được dẫn đầu bởi Dr Daquing chen thu thập để phục vụ cho đồ án này. Qua việc áp dụng thuật toán Louvain và các độ đo Similar, Page Rank, Closeness giải quyết các vấn đề:
  + Phân tích hành vi mua hành của các cộng đồng khách hàng
  + Đề xuất các sản phẩm khách hàng nên mua.
  + Các sản phẩm bán chạy cũng như sản phẩm khó tiêu thụ.
* Xét trong bối cảnh dữ liệu được thu thập từ 1/12/2010 đến 9/12/2011, dữ liệu được thu thập theo từng giờ từng phút và bao gồm 541911 dòng.

- Công cụ hỗ trợ: Excel,Noe4j.

## Mô tả bài toán:

Nhóm sẽ sử dụng Neo4j để giải quyết bài toán trên. Sau khi đã import dữ liệu vào Neo4j, nhóm sẽ sử dụng độ đo Jaccard Similarity để tìm ra các khách hàng tương đồng với nhau trong việc mua hàng. Sau đó sẽ tiến hành gom cụm các khách hành tương đồng với nhau bằng thuật toán Louvain. Sau khi đã gom được các cụm cộng đồng sẽ tính độ tương đồng Similarity của các sản phẩm thuộc từng cụm cộng đồng. Để xem xét những sản phẩm hay được các khách hàng trong từng cộng đồng mua. Từ đó suy ra được hành vi mua hàng của từng cụm cộng đồng. Kế tiếp nhóm sẽ tính độ đo Closeness và Page Rank của các sản phẩm thuộc từng cụm cộng đồng để có thể biết đâu là sản phẩm bán chạy và đâu là sản phẩm khó tiêu thụ.

# CHƯƠNG 2: TÌM HIỂU VỀ NEO4J

## Neo4J là gì

Neo4j là một cơ sở dữ liệu NoSql dạng đồ thị mã nguồn mở xây dựng bằng Java và Scala do tập đoàn Neo Technology phát triển, là cơ sở dữ liệu giao dịch ACID có thể lưu trữ và xử lý đồ thị nguyên bản. Mô hình dữ liệu theo kiểm Nodes – Relationships

Đối với neo4j đối tượng được mô tả thành các đỉnh của đồ thị, đặc điểm của đối tượng được mô tả qua thuộc tính của đỉnh và mối quan hệ giữu các đối tượng được mô tả bằng liên kết có hướng giữa các đỉnh.

Neo4j là cơ sở dữ liệu đồ thị phổ biến nhất theo xếp hạng DB-Engines. Neo4j có 02 chế độ:

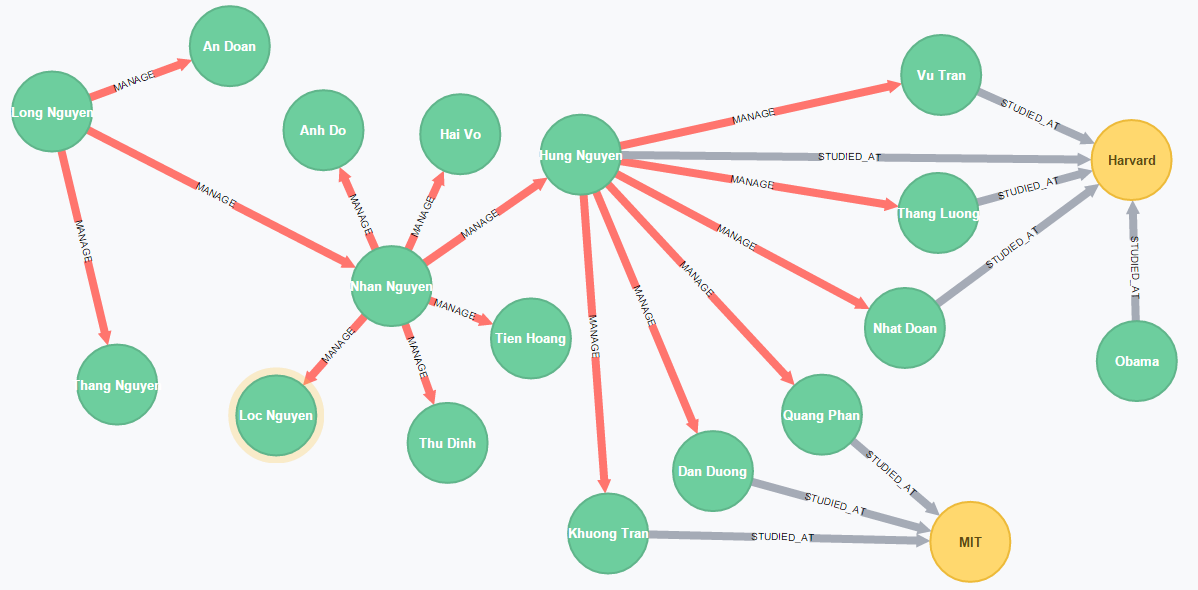
* Quản trị dữ liệu và thể hiện nội dung trên nền tảng riêng (Neo4j Server)
* Nhúng vào các nền tảng khác: Neo4j chỉ đóng vai trò lưu trữ (Graph Database)

**Một số tính năng trong Neo4J như:**

* Được viết bên trong JVM giảm thời gian biên dịch.
* Lưu trữ nhỏ gọn và bộ nhớ đệm cho các đồ thị, vì vậy có thể khả năng mở rộng lưu trữ hàng tỉ nút trong một cơ sở dữ liệu trên một hệ thống phần cứng vừa phải.
* Tất cả quan hệ trong Neo4j đều quan trọng và nhanh chóng, khiến nó có thể trở thành thực thể và dử dụng các mối quan hệ mới sau này trên "shortcut" và tăng tốc độ dữ liệu tên miền khi có nhu cầu phát sinh.
* Bỏ qua các hằng số thời gian cho các quan hệ kể cả về chiều sâu và chiều rộng do cách biều diễn các nút và quan hệ có khả năng.
* Cụ thể hóa các mối quan hệ tại thời điểm tạo, kết quả là không có cho truy vấn thời gian phức tạp.

## Cách tổ chức dữ liệu trong Neo4J

* **Node**: là một trong những đơn vị cơ bản trong Neo4J, chứa nội dung, thuộc tính của một đối tượng .
* **Label**: một node có thể có 1 hoặc nhiều label, dùng để dịnh danh một node và là một thành phần không thể thiếu trong truy vấn dữ liệu.
* **Relationship**: có thể hiểu như là những cạnh nối giữa các node, cho biết mối liên hệ của các node đó là gì.

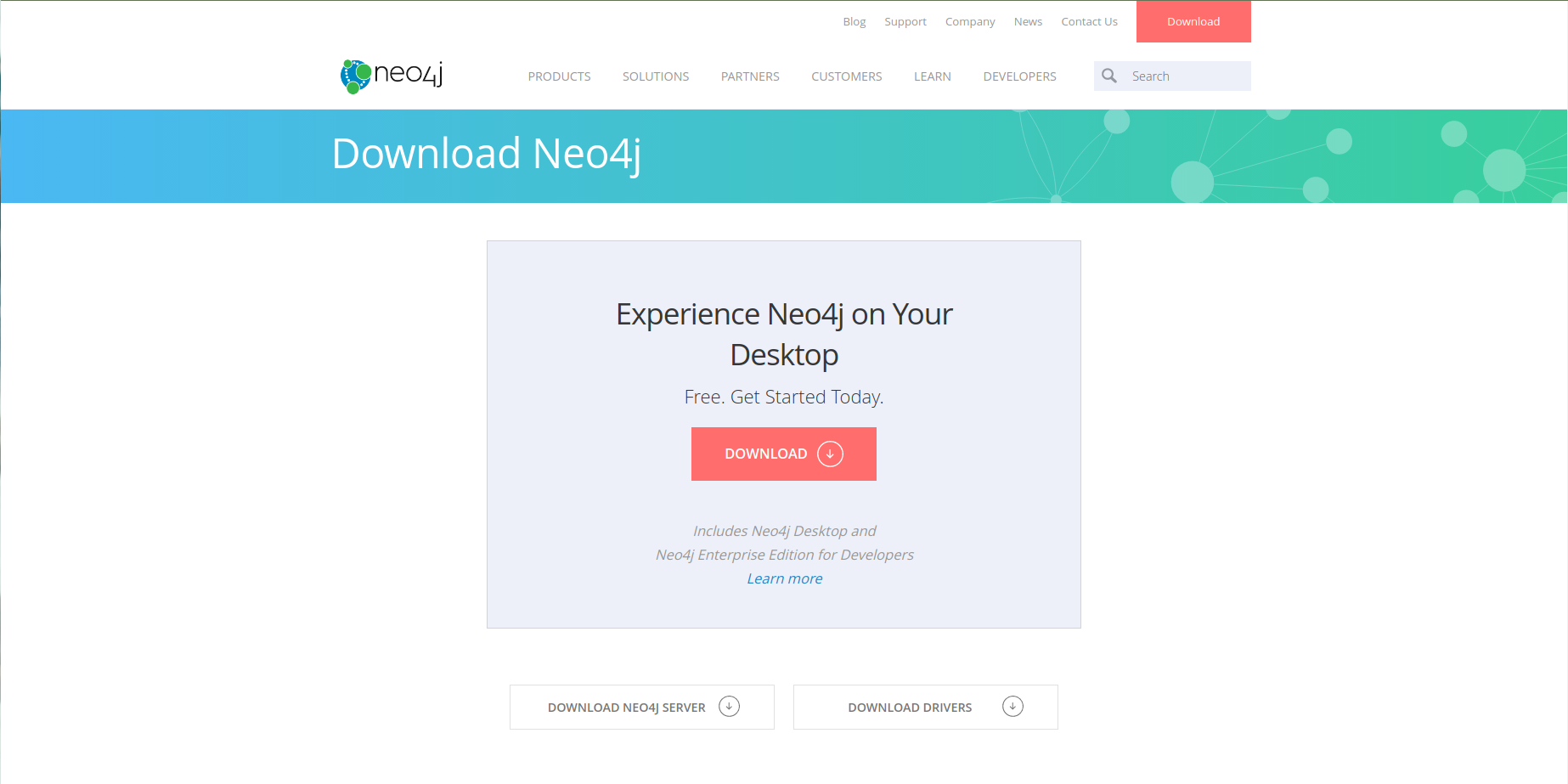


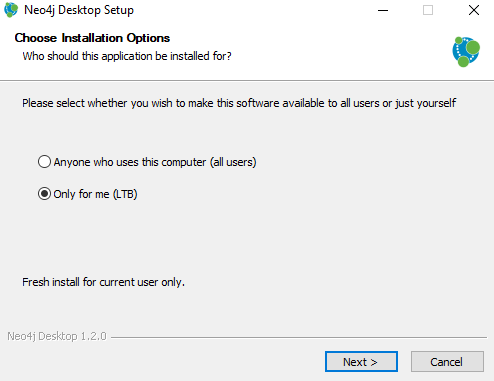
Từ cách tổ chức dữ liệu này chúng ta có thể thấy rằng Neo4J rất phù hợp trong nhữngCSDL có nhiều mối liên hệ, routing... ví dụ như CSDL trong mạng xã hội.

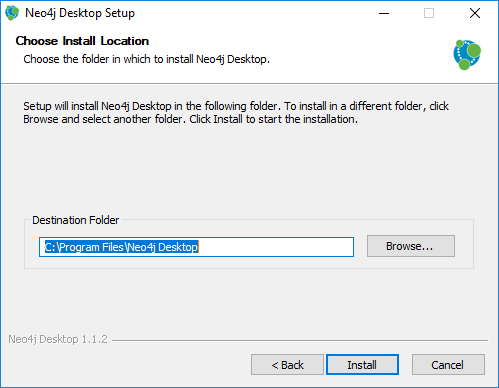
## Các điểm nổi bật

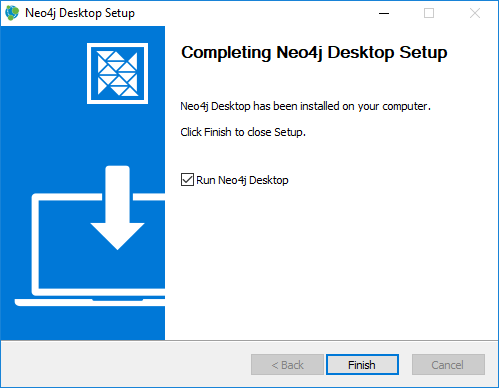
* Cơ sở dữ liệu đồ thị gốc:
* Neo4j được xây dựng từ dưới lên để trở thành một cơ sở dữ liệu đồ thị. Kiến trúc được thiết kế để tối ưu hóa việc quản lý, lưu trữ và truyền tải nhanh các nút và mối quan hệ. Trong Neo4j, các mối quan hệ là các lớp đầu tiên đại diện cho các kết nối được thực hiện trước giữa các thực thể. Một hoạt động công khai trong cơ sở dữ liệu quan hệ được xem như một phép nối, mà hiệu năng của nó giảm xuống theo cấp số nhân với số lượng các mối quan hệ, được biểu diễn bởi Neo4j như điều hướng từ nút này sang nút khác, có hiệu suất là tuyến tính.
* Cách tiếp cận khác này để lưu trữ và truy vấn các kết nối giữa các thực thể cung cấp hiệu suất truyền tải lên đến 4 triệu bước nhảy mỗi giây và lõi. Vì hầu hết các tìm kiếm đồ thị là cục bộ cho vùng lân cận lớn hơn của một nút, tổng lượng dữ liệu được lưu trữ trong cơ sở dữ liệu sẽ không ảnh hưởng đến thời gian thực thi của hoạt động. Quản lý bộ nhớ chuyên dụng, và hoạt động hiệu quả cao và khả năng mở rộng bộ nhớ, góp phần vào lợi ích.
* Bảng trắng thân thiện:
  + Cách tiếp cận của Property Graphs cho phép sử dụng đồng nhất cùng một mô hình trong suốt quan niệm, thiết kế, triển khai, lưu trữ và hiển thị của bất kì domain hoặc use case nào. Với mô hình otional schema, mô hình domain có thể được phát triển liên tục khi thay đổi yêu cầu.
  + Ngôn ngữ truy vấn biểu đồ Cypher được thiết kế để biểu thị trực quan các mẫu biểu đồ của các nút và các mối quan hệ. Ngôn ngữ truy vấn có khả năng cao nhưng dễ đọc, tập trung vào các mẫu thể hiện các khái niệm hoặc câu hỏi từ một domain cụ thể. Cypher cũng có thể được mở rộng để tối ưu hóa hẹp cho các trường hợp sử dụng cụ thể.
* Hỗ trợ phát triển nhanh chóng: Neo4j hỗ trợ phát triển nhanh các hệ thống đồ thị được hỗ trợ. Sự phát triển của Neo4j bắt nguồn từ sự cần thiết phải thực thi các câu truy vấn thời gian thực trên các thông tin có tính liên quan cao mà không có cơ sở dữ liệu nào khác có thể cung cấp. Những tính năng độc đáo này giúp bạn khởi động, thực thi nhanh và duy trì phát triển ứng dụng nhanh chóng cho các ứng dụng có khả năng mở rộng cao.
* Cung cấp sự an toàn dữ liệu thông qua các giao dịch ACID: Neo4j sử dụng các giao dịch ACID để đảm bảo răng dữ liệu được duy trì trong trường hợp lỗi phần cứng hoặc sự cố hệ thống.
  + Được thiết kế cho các hoạt động kinh doanh quan trọng và hiệu suất cao: Neo4j có thể lưu giữ hàng trăm nghìn tỷ thực thể cho các tập dữ liệu cực kì lớn trong bộ nhớ nhỏ gọn. Neo4j có thể được triển khai như một cụm máy có khả năng mở rộng, có khả năng chịu lỗi. Do khả năng mở rộng cao, các cụm Neo4j chỉ yêu cầu hàng chục máy, không phải hàng trăm hoặc hàng nghìn, tiết kiệm chi phí và độ phức tạp của hoạt động. Các tính năng khác cho các ứng dụng sản xuất bao gồm các bản sao lưu nóng và giám sát rộng rãi.

## Cài đặt Neo4J

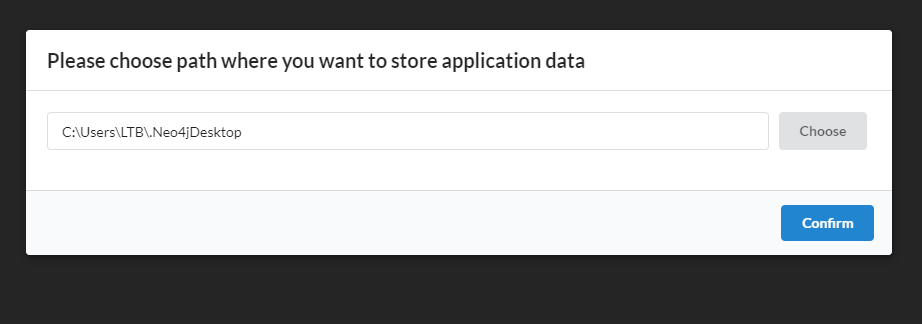
* Đường dẫn: <https://neo4j.com/download/>
* Cài đặt theo hướng dẫn



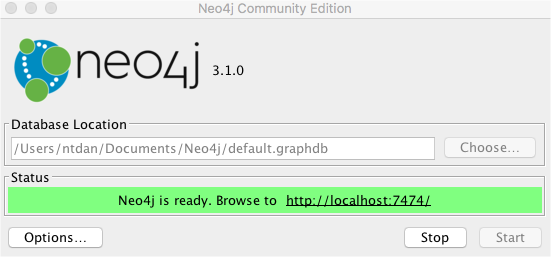




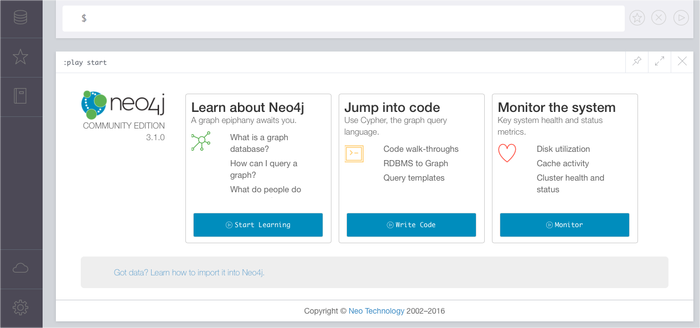
* Chạy neo4j và cấu hình nơi lưu dữ liệu



* Nhấn nút start để mở server neo4j và đây là giao diện sau khi chạy



* Nhấn chọn liên kết dưới để mở giao diện web của neo4j, đăng nhập bằng tài khoản neo4j/neo4j



* Như vậy là mọi thứ đã sẵn sàn để thiết kế dữ liệu đồ thị.

# CHƯƠNG 3: ỨNG DỤNG NEO4J VÀO PHÂN TÍCH MẠNG XÃ HỘI:

## Dataset

Dữ liệu được nhóm lấy từ nguồn UCI Machine Learning Repository. Đây là dữ liệu về Online Retailer in UK – Dữ liệu bán lẻ online. Dữ liệu được Dr Daqing Chen thu thập và phân tích trong topic “Data mining for the online retail industry”.

Dữ liệu bao gồm các thuộc tính:

InvoiceNo : Số hóa đơn bao gồm 6 số, nếu nó bắt đầu bằng chữ C thì có nghĩa là hóa đơn ấy bị hủy bỏ.

StockCode: Mã sản phẩm bao gồm 5 số. Và có thể kèm theo 1 chữ.

Description: Tên sản phẩm

Quantity: Số lượng sản phẩm trong mỗi lần giao dịch.

InvoiceDate: Ngày và giờ giao dịch.

UnitPrice: Giá tiền sản phẩm. Được tính trên mỗi sản phẩm

CustomerID: Mã khách hàng thực hiện giao dịch bao gồm 5 số.

Country: Tên quốc gia nơi khách hàng cư trú.

Dữ liệu gốc sẽ bao gồm 581587 dòng.

Ví dụ về Dataset:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Invoice | StockCode | Description | Quantity | InvoiceDate | Price | Customer ID |
| 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01/12/2010 8:26 | 2.55 | 17850 |
| 536365 | 71053 | WHITE METAL LANTERN | 6 | 01/12/2010 8:26 | 3.39 | 17850 |
| C536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 01/12/2010 8:26 | 2.75 | 17850 |

## Import dữ liệu

Để import dữ liệu vào neo4j một cách tốt nhất. Nhóm đã tách file trên thành cách file nhỏ. Cụ thể như sau:

* Các file để import Node:

UniqueCategories: Loại sản phẩm đây là thuộc tính nhóm đưa vào để phục vụ các câu truy vấn sau này nếu cần thiết .

UniqueItems: Bao gồm StockCode và Description.

UniqueCountries: Bao gồm Country.

UniqueHouseholds:Bao gồm CustomerID.

UniqueTransactions: Bao gồm TransactionID(Đổi tên InvoiceNo), InvoiceDay, và Epochtime (Convert từ Ngày, Giờ sang một dãy số ).

* Các file để import Relationship:

Item-Category: Bao gồm StockCode và Category.

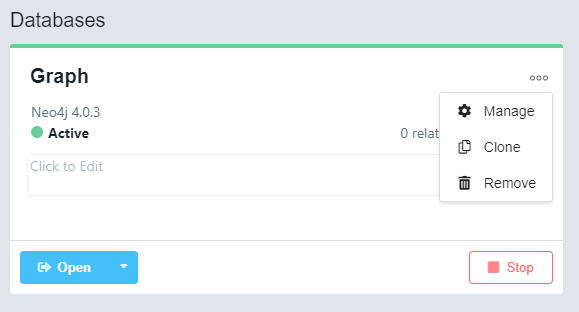
Household-Transaction: Bao gồm CustomerID và TransactionID.

Household-Country: Bao gồm CustomerID và Country.

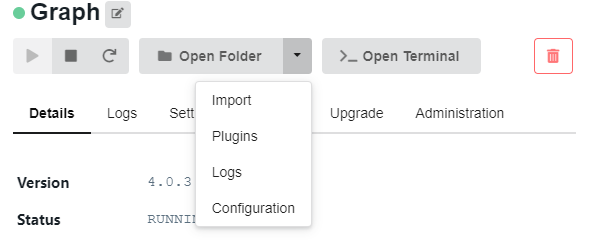
Customer-Item: Bao gồm StockCode CustomerID và Quantity.

Transaction-Item: Bao gồm TransactionID, StockCode, Quantity và Price

* Chọn manage để thiết lập database trong Neo4j:



* Chọn import để chuyển đến thư mục import sau đó copy các file bên trên vào thư mục này:



* Khởi động Neo4j và chạy các câu query sau để import dữ liệu từ csv sang database:

LOAD CSV WITH HEADERS FROM "file:///UniqueCategories.csv" AS row

WITH row.ITEMCATEGORY as ItemCategory

MERGE (c:Category{Category:ItemCategory})

RETURN COUNT (c);

LOAD CSV WITH HEADERS FROM "file:///UniqueItems.csv" AS row

WITH toInteger(row.StockCode) as StockCode, row.Description as Description where StockCode is not null

MERGE (i:Item{StockCode: StockCode, Description:Description})

RETURN COUNT (i);

LOAD CSV WITH HEADERS FROM "file:///UniqueCountries.csv" AS row

WITH row.Country as CountryName

MERGE (c:Country{Country:CountryName})

RETURN COUNT (c);

LOAD CSV WITH HEADERS FROM "file:///UniqueHouseholds.csv" AS row

WITH toInteger(row.CustomerID) as CustomerID

MERGE (c:Customer{CustomerID:CustomerID})

RETURN COUNT (c);

LOAD CSV WITH HEADERS FROM "file:///UniqueTransactions.csv" AS row

WITH toInteger(row.Transaction\_ID) as TransactionID, row.InvoiceDate as InvoiceDate, toInteger(row.epochtime) as EpochTime

MERGE (t:Transaction{TransactionID:TransactionID, InvoiceDate:InvoiceDate, EpochTime:EpochTime})

RETURN COUNT (t);

//Add relationships

:auto

USING PERIODIC COMMIT 500

LOAD CSV WITH HEADERS FROM "file:///item-category.csv" as row

WITH toInteger (row.StockCode) as StockCode, row.CATEGORY as Category

MATCH (i:Item{StockCode:StockCode})

MATCH (c:Category{Category:Category})

MERGE (i)-[:TYPE]->(c);

:auto

USING PERIODIC COMMIT 500

LOAD CSV WITH HEADERS FROM "file:///household-transaction.csv" as row

WITH toInteger(row.CustomerID) as CustomerID, toInteger(row.Transaction\_ID) as TransactionID

MATCH (c:Customer{CustomerID:CustomerID})

MATCH (t:Transaction{TransactionID:TransactionID})

MERGE (c)-[:MADE\_TRANSACTION]->(t);

:auto

USING PERIODIC COMMIT 500

LOAD CSV WITH HEADERS FROM "file:///household-country.csv" as row

WITH toInteger(row.CustomerID) as CustomerID, row.Country as Country

MATCH (c:Customer{CustomerID:CustomerID})

MATCH (c2:Country{Country:Country})

MERGE (c)-[:FROM]->(c2);

:auto

USING PERIODIC COMMIT 500

LOAD CSV WITH HEADERS FROM "file:///customer-item.csv" as row

WITH toInteger(row.NumberPurchased) as NumberPurchase, toInteger(row.CustomerID) as CustomerID, tointeger (row.StockCode) as StockCode

MATCH (c:Customer{CustomerID:CustomerID})

MATCH (i:Item {StockCode:StockCode})

MERGE (c)-[:BOUGHT{Quantity:NumberPurchase}]->(i);

:auto

USING PERIODIC COMMIT 500

LOAD CSV WITH HEADERS FROM "file:///transaction-item.csv" as row

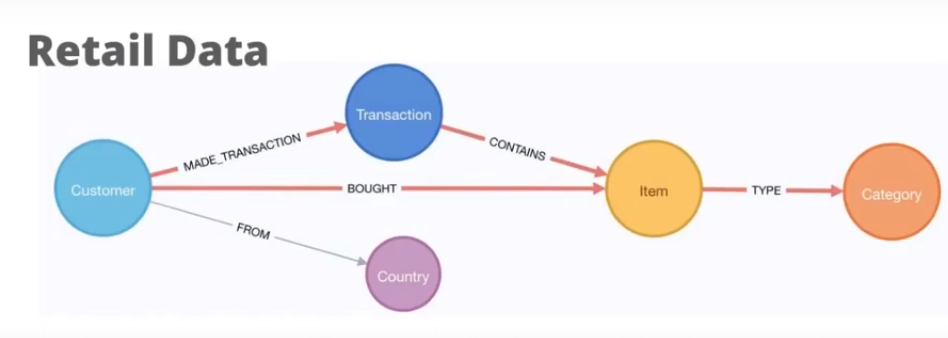
WITH tointeger (row.StockCode) as StockCode, toFloat(row.Price) as Price, toInteger(row.Transaction\_ID) as TransactionID, toInteger(row.Quantity) as Quantity

MATCH (i:Item{StockCode:StockCode})

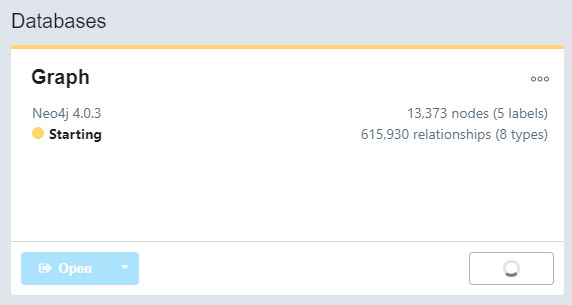
MATCH (t:Transaction{TransactionID:TransactionID})

MERGE (t)-[:CONTAINS{Quantity:Quantity, Price:Price}]->(i);

* Sau khi import thì sơ đồ database của chúng ta sẽ như sau:



* Database sẽ có 13373 node và 615930 liên kết



## Các thuật toán được sử dụng

### Jaccard Similarity

Chỉ số Jaccard là một chỉ số so sánh các thành phần trong hai bộ để xem thành viên nào được chia sẻ chung và thành phần nào phân biệt. Nó đo được mức độ tương tự của hai bộ dữ liệu phạm vi từ 0% đến 100%.

Có công thức là: J(X,Y) = |X∩Y| / |X∪Y|

Ví dụ chúng ta có 4 khách hàng là A, B, C và D. Chúng ta có 7 sản phẩm là HP1 (Harry Porter 1), HP2 (Harry Porter 2), HP3 (Harry Porter 3), TW (Twilight).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| HP1 | 4 | 5 |  |  |
| HP2 |  | 5 |  | 3 |
| HP3 |  | 5 |  |  |
| TW | 5 |  | 2 |  |

User A và user B đều mua sản phẩm HP1 nên rA ∩ rB = 1. User A và user B mua tổng cộng 4 sản phẩm nên rA U rB = 4. Do đó, điểm số tương đồng của User A và User B là J(A,B) = 1/4.

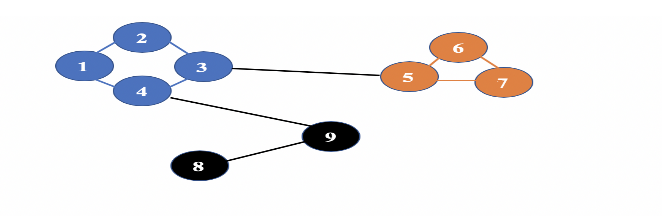
### Louvain

#### Tính mô đun (modularity)

Tính mô đun (Modularity) là một thước đo cấu trúc của mạng hoặc đồ thị. Nó được thiết kế để đo lường sức mạnh của việc phân chia mạng thành các mô-đun (còn được gọi là nhóm, cụm hoặc cộng đồng). Các mạng có tính mô đun cao có các kết nối dày đặc giữa các nút trong các mô-đun của chúng, nhưng sẽ kết nối thưa thớt với các nút trong các mô-đun khác. Tính mô đun thường được sử dụng trong các phương pháp tối ưu hóa để phát hiện cấu trúc cộng đồng trong các mạng.

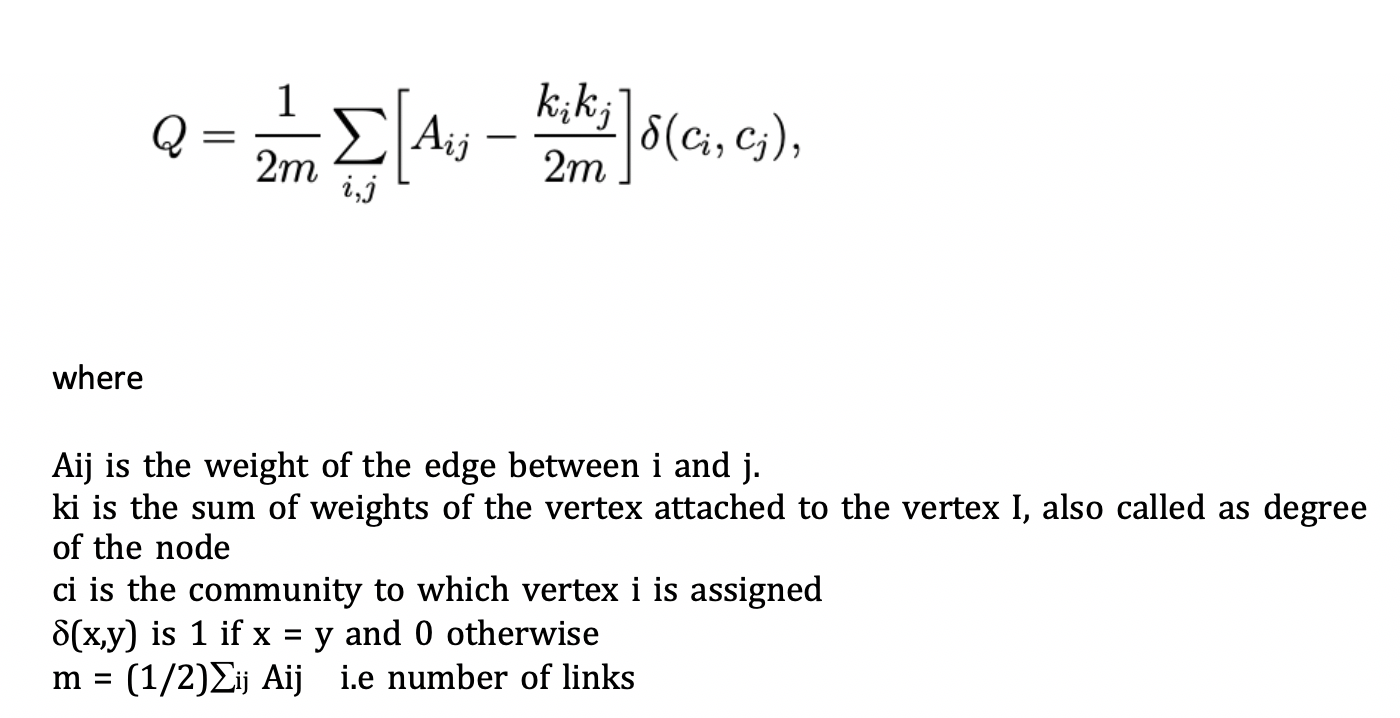
#### Phát hiện cộng đồng (Community Detection)

Phát hiện cộng đồng trong các mạng là một trong những chủ đề phổ biến nhất của khoa học mạng hiện đại. Các cộng đồng, hoặc cụm, thường là các nhóm đỉnh có xác suất kết nối với nhau cao hơn so với các thành viên của các nhóm khác. Xác định các cộng đồng là một vấn đề không xác định một cách rõ ràng. Không có giao thức phổ quát về các thành phần cơ bản, như định nghĩa của chính khái niệm cộng đồng, cũng như các vấn đề quan trọng khác, như xác nhận các thuật toán và so sánh hiệu suất của chúng.



Hình : ví dụ về phát hiện cộng đồng

Modularity đo lường chất lượng cho một phân vùng cụ thể của mạng. định lượng sức mạnh cộng đồng bằng cách so sánh phân số của các cạnh trong cộng đồng với phần như vậy khi kết nối ngẫu nhiên giữa các nút được thực hiện. Giả định ban đầu là một cộng đồng nên có nhiều mối liên hệ giữa bản than nó hơn là lấy kết nối ngẫu nhiên một số cụm.Độ đo modularity được tính như sau:



A: là ma trận kề thông thường.

K : là bật của node.

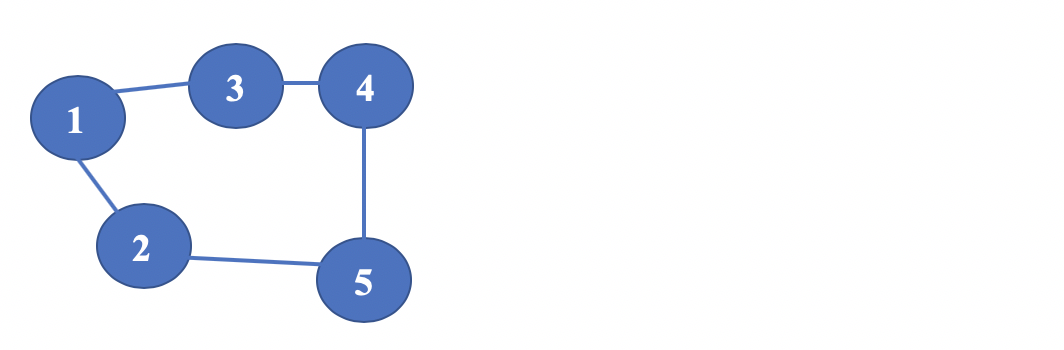
M : tổng số lượng liên kết.

delta δ(ci,cj) : bằng 1 nếu i và j cùng nhóm, bằng 0 nếu khác nhóm.

ki/2m : là trọng số node i chia cho tổng trọng số của toàn mạng.

Hiệu số Aij – kikj/2m nếu âm thì chứng tỏ i và j không nên ở cùng nhóm.

Ví dụ ta có các node sau :



Giả sử chúng ta chia các node trên làm 2 nhóm:

Nhóm 1: node 1,3,4.

Nhóm 2: node 2,5.

Số liên kết: m=5

Ki,kj là bật của node (2 trong ví dụ này)

D(x,y) bằng 0 nếu node nằm cùng trong một nhóm nếu không thì bằng 1.

*Sử dụng công thức ta có:*

Q = 1 / (2 \* 5) ( (0–2 \* 2 / (2 \* 5) ) \* 1 + # node 1 với node 1 -> không liên kết, cùng nhóm

(1–2 \* 2 / (2 \* 5) ) \* 0 + # node 1 với 2 -> có liên kết, khác nhóm

(0–2 \* 2 / (2 \* 5) ) \* 1 + # node 1 to 4 -> không liên kết, cùng nhóm

(0–2 \* 2 / (2 \* 5) ) \* 0 + # node 1 to 5 -> không liên kết, khác nhóm

(1–2 \* 2 / (2 \* 5) ) \* 1 + # node 1 to 3 -> có liên kết, cùng nhóm

Tiếp tục với phần còn lại ta rút gọn:

Q = 1 / 10 ( 7 \* (0–2 / 5) + 6 \* (1–2 / 5) ) = 4 / 50 = 0.08

#### Thuật toán Louvain

* Lấy một node và cố gắng gom nó vào chung với những hàng xóm của nó
* Chọn hàng xóm sao cho tối đa được điểm modularity.
* Lặp lại qua tất cả các node. Chúng ta sẽ hợp các node lại với nhau tạo thành các cộng đồng.
* Những node đã gộp thành cộng đồng sẽ trở thành đầu vào mới cho thuật toán. Coi mỗi cộng đồng là một node và cố gắng gom chúng lại thành những cộng đồng lớn hơn.
* Thuật toán dừng khi không thể tìm hàm modurarity nào lớn hơn được nữa.

### Page Rank

Pagerank là thuật toán phân tích các liên kết được dùng trong Google Search để xếp hạng các trang web. Thuật toán này chỉ định giá trị nhất định cho mỗi thành phần của một tập hợp các văn bản liên kết với nhau, ví dụ như World Wide Web. Mục đích "đo" tầm quan trọng tương đối của các liên kết trong tập hợp đó.

Giả sử một nhóm gồm 4 trang web: A, B, C, D. những liên kết từ một trang đến chính nó không được tính, mỗi trang web có 1 đường dẫn duy nhất đến 1 trang web khác. Giá trị Pagerank của các trang ban đầu được cho là bằng nhau. Do đó giá trị ban đầu cho mỗi trang là 0.25. Nếu các liên kết duy nhất trong hệ thống từ các trang B, C và D tới A, mỗi liên kết sẽ chuyển giá trị bằng 0.25 Pagerank A khi tính trong lần tiếp theo, tổng cộng là 0,75.

PR(A)=PR(B)+PR(C)+PR(D).

Khác với ví dụ trên, B có liên kết đến trang C và A, trong khi D có các link đến cả ba trang. Như vậy trong bước tiếp theo, trang B sẽ chuyển tải một nửa giá trị của mình, tương đương với 0.125 tới trang A và 0.125 tới trang C. Khi trang D có 3 liên kết trỏ đi, có nghĩa nó sẽ chuyển 1/3 giá trị của mình, tương đương với 0.083 tới A.

PR(A)=PR(B)/2 +PR(C)/1 +PR(D)/3.

### Closeness

Độ đo Closeness của mỗi nút bằng tổng khoảng cách của nó với tất cả các nút khác,được tính dựa trên việc tính toán các đường dẫn ngắn nhất giữa tất cả các cặp nút. Tổng kết quả sau đó được đảo ngược để xác định điểm số Closeness cho nút đó. Điểm số Closeness càng cao thì nút ấy đi đến các nút khác trong mạng càng nhanh.

## Phân tích dữ liệu

### Gom cụm các khách hàng tương đồng với nhau:

Bước 1: chúng ta sẽ tạo ra một graph để lưu dữ liệu cũng như thao tác trên nó. Các node của graph bao gồm node Customer, node Item và các Relationship của chúng :

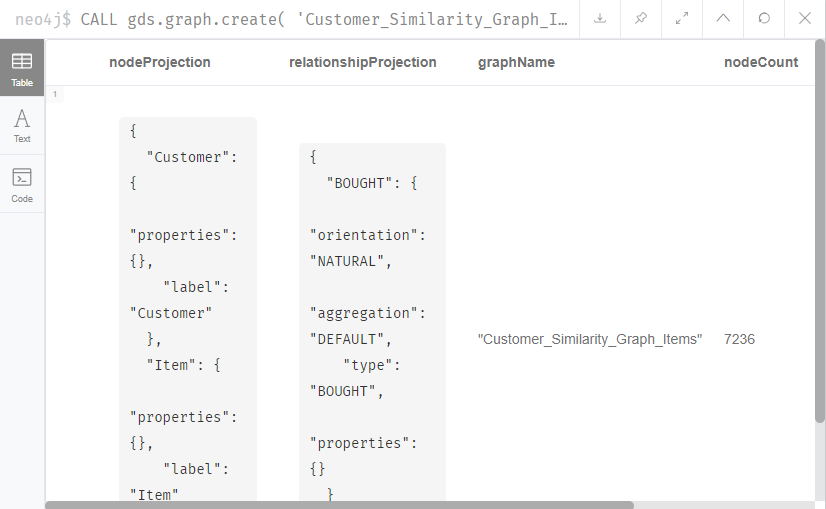
CALL gds.graph.create(

'Customer\_Similarity\_Graph\_Items',

['Customer','Item'],

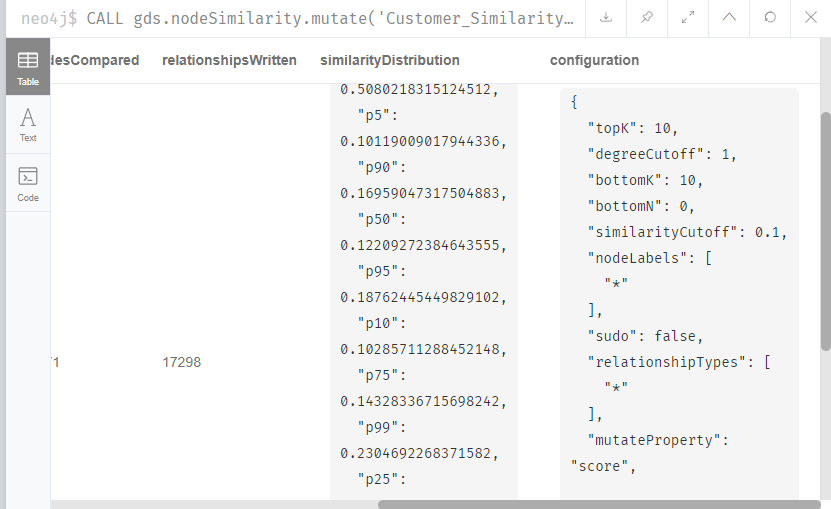
'BOUGHT'

)



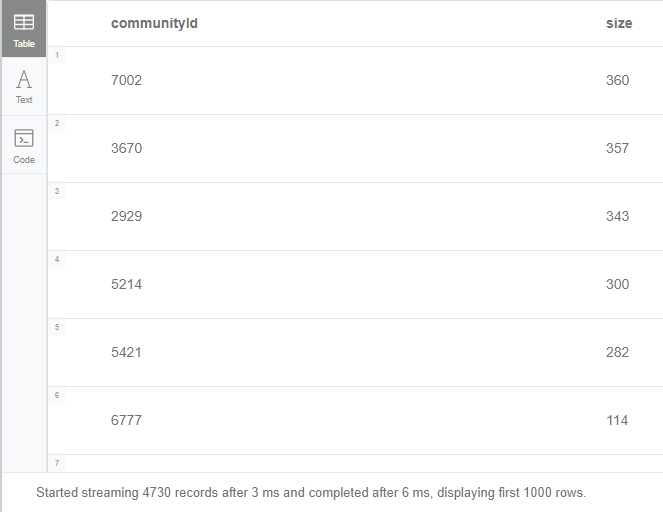
Bước 2: chúng ta sẽ tìm những node Customer có độ tương đồng (Similar) với nhau trong việc mua sản phẩm bằng thuật toán Jaccard Similarity. Ở đây ta sẽ sử dụng lệnh mutate tạo Relationship tên ‘Similar’ trên graph 'Customer\_Similarity\_Graph\_Items’, chứ không tạo Relationship trên database chính. Kế đó ta có các tham số như topK:10, nghĩa là ta sẽ so sánh 1 Customer với 10 Item bởi vì nếu không giới hạn thì có thể sẽ xảy ra lỗi. Similarity Cut Off: 0.1 nghĩa là chỉ cần customer mua chung 1 Item duy nhất (1/10) thì cũng được tính là tương đồng vì ta không muốn cộng đồng của mình quá nhỏ. Thực hiện Query sau:

CALL gds.nodeSimilarity.mutate('Customer\_Similarity\_Graph\_Items', {mutateRelationshipType:'Similar', mutateProperty:'score',topK:10, similarityCutoff:0.1})



Bước 3: chúng ta sẽ sử dụng thuật toán Louvain – phát hiện cộng đồng. Để gom các node khách hàng tương đồng với nhau thành 1 cụm riêng biệt. Để đảm bảo kết quả xuất ra gom được các cộng đồng lớn chứ không phải các cộng đồng lẻ tẻ, ta cần sử dụng câu lệnh stream của Neo4j để xem kết quả trả về.

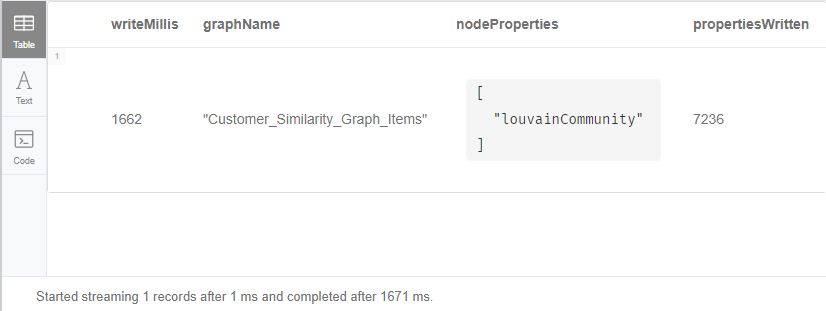
CALL gds.louvain.stream('Customer\_Similarity\_Graph\_Items',{relationshipTypes:['Similar'], relationshipWeightProperty:'score'}) YIELD nodeId, communityId RETURN communityId, count(nodeId) as size ORDER by size DESC



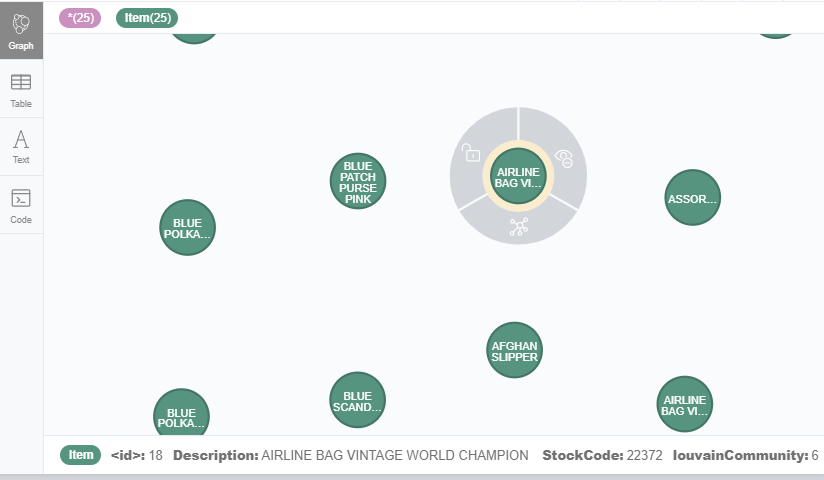
Bước 4: Kết quả trên đã gom được những cộng đồng lớn, nên tiếp theo chúng ta sẽ ghi tên cộng đồng LouvainCommunity này vào các node trên graph 'Customer\_Similarity\_Graph\_Items' bằng lệnh mutate. Và ghi lên các node trong database chính bằng lệnh writeNodePerties.

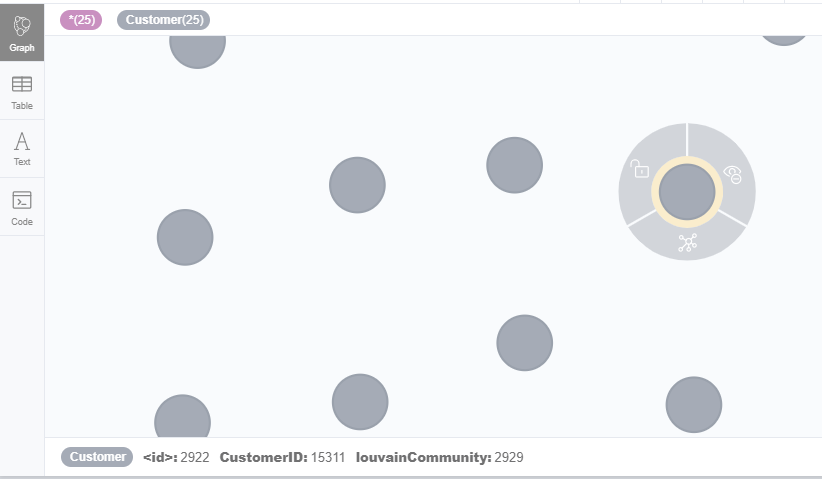
CALL gds.louvain.mutate('Customer\_Similarity\_Graph\_Items',{relationshipTypes:['Similar'], relationshipWeightProperty:'score', mutateProperty:'louvainCommunity'})

CALL gds.graph.writeNodeProperties('Customer\_Similarity\_Graph\_Items',['louvainCommunity'])



Bước 5: Kiểm tra lại các node trên database chính thì chỉ số louvain community này đã được ghi thẳng vào các node



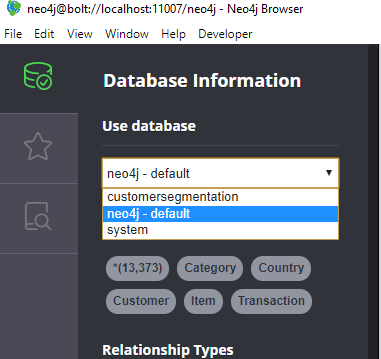


Bước 6: Để có thể xem rõ ràng hơn các kết quả của thuật toán Similarity và Lovain chúng ta sẽ xuất dữ liệu Graph 'Customer\_Similarity\_Graph\_Items' sang một database mới tên là CustomerSegmentation. Database gốc của chúng là neo4j.

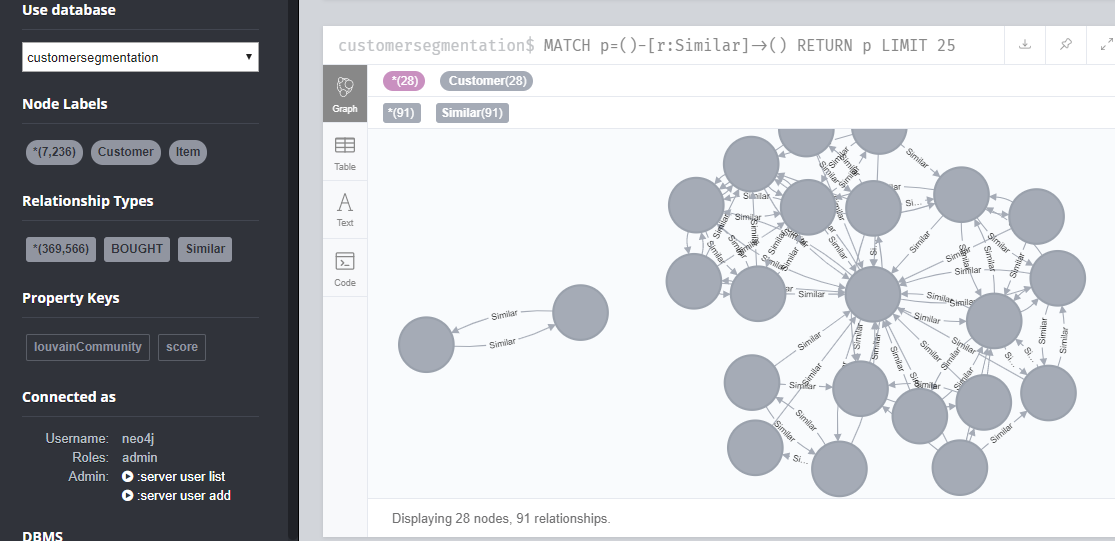
CALL gds.beta.graph.export('Customer\_Similarity\_Graph\_Items',{dbName:'CustomerSegmentation'})

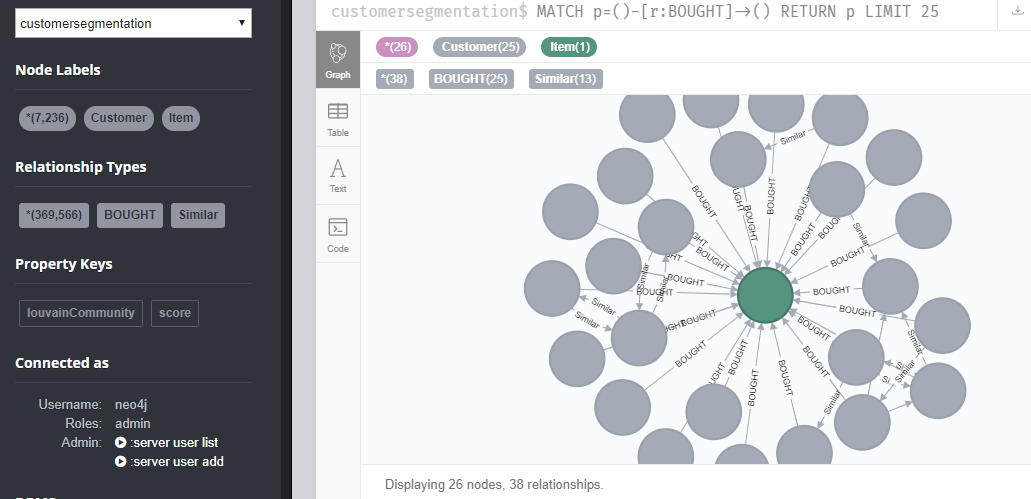
: use system

Create database CustomerSegmentation



Bước 7: Nhấn vào Relationship Types Similar và Bought ta sẽ thấy rõ các cộng đồng khách hàng tương đồng với các customer node được nối với nhau bằng Relationship Similar





### Tính similarity , closeness của các cụm cộng đồng:

Bước 1: Tính similarity (độ tương đồng) của các item được mua bởi các customer thuộc cụm cộng đồng 2929,5214,5421. Lần lượt thay 2929, 5214, 5421 vào thông số c.louvainCommunity:

CALL gds.graph.create.cypher(

'Item\_Similarity\_2929',

'MATCH (n) WHERE n:Transaction or n:Item return id(n) as id',

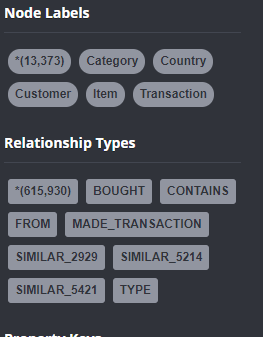
'MATCH (t:Transaction)<--(c:Customer) WHERE c.louvainCommunity = **2929** WITH t MATCH (i:Item)<--(t) RETURN distinct id(i) as source, id(t) as target'

);

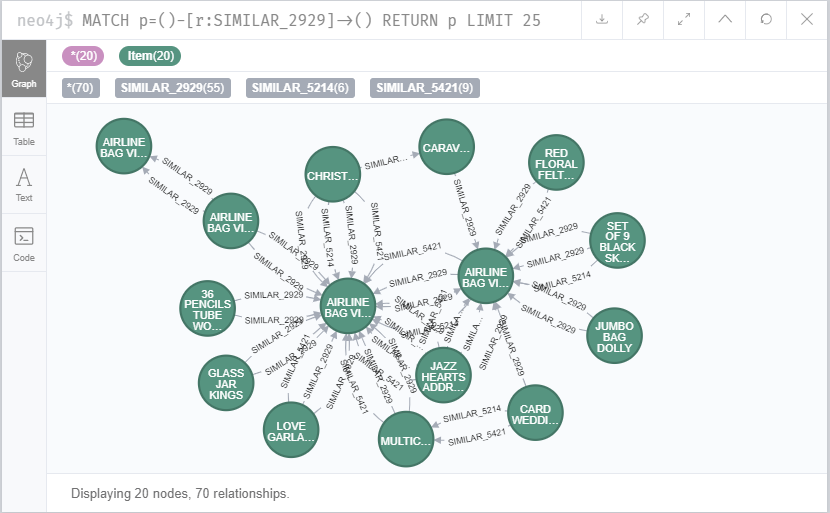
CALL gds.nodeSimilarity.mutate('Item\_Similarity\_2929',{mutateRelationshipType:'SIMILAR\_2929', mutateProperty:'score'});

CALL gds.graph.writeRelationship('Item\_Similarity\_2929','SIMILAR\_2929','score');

Như vậy ta sẽ tạo ra được 3 Realtionship mới lần lượt là SIMILAR\_2929, SIMILAR\_5241 và SIMILAR\_5421



Bước 2: Nhấn vào Relationship Types SIMILAR\_2929. Ta sẽ thấy những sản phẩm được mua bởi cộng đồng 2929. Từ kết quả này ta có thể đề xuất các sản phẩm cho cộng đồng này khi cần.

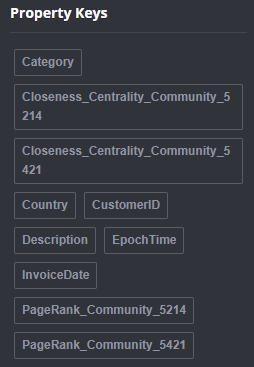
Bước 3: Tính PageRank và Closeness của 2 item được mua bởi các customer của cộng đồng 5214 và 5421. Lần lượt thay 5421, 5241 vào các thông số c.louvainCommunity:

CALL gds.graph.create.cypher('item\_copurchase\_5214','MATCH (i:Item)<--(c:Customer) where c.louvainCommunity = **5214** RETURN DISTINCT id(i) as id','MATCH (i1:Item)<--(c:Customer)-->(i2:Item) WHERE c.louvainCommunity = **5214** RETURN distinct id(i1) as source, id(i2) as target, count(c) as weight');

CALL gds.alpha.closeness.write('item\_copurchase\_5214',{writeProperty:'Closeness\_Centrality\_Community\_5214'});

CALL gds.pageRank.write('item\_copurchase\_5214',{relationshipWeightProperty:'weight',writeProperty:'PageRank\_Community\_5214'});

Ở câu lệnh tạo Graph trên ta đã chỉ cho Neo4j rằng hãy lấy quan hệ Item<-Customer->Item. Nghĩa là xét trường hợp mỗi customer sẽ mua 2 item. Và kiểm tra lại ta sẽ thấy các độ đo đã được Neo4j tính toán xong.



### Thực hiện các câu truy vấn:

Bước 1: Tìm các cặp Item tương đồng đều được mua bởi cộng đồng 5214 mà không được mua bởi cộng đồng 2929

MATCH p=(i1:Item)-[r:SIMILAR\_5214]->(i2)

WHERE NOT (i1:Item)-[r:SIMILAR\_2929]->(i2)

AND r.score > .1 AND r.score < .8

RETURN i1.Description, i2.Description, r.score as score order by score desc



Bước 2: Tìm các cặp Item tương đồng đều được mua bởi cộng đồng 2929 mà không được mua bởi cộng đồng 5214

MATCH p=(i1:Item)-[r:SIMILAR\_2929]->(i2)

WHERE NOT (i1:Item)-[r:SIMILAR\_5214]->(i2)

AND r.score > .1 AND r.score < .8

RETURN i1.Description, i2.Description, r.score as score order by score desc



Từ 2 câu query trên ta có thể thấy được sự khác nhau trong hành vi mua hàng của khách hàng. Nêu như cộng đồng 5214 mua những món hàng dường như dành cho tiệc sinh nhật thì cộng đồng 2929 lại mua những món hàng dành cho việc trang trí làm vườn. Từ những đặc điểm này ta có thể xây dựng chiến lược chăm sóc khách hàng hoặc những chiến lược bán hàng đặc trưng theo từng cụm cộng đồng khác nhau.

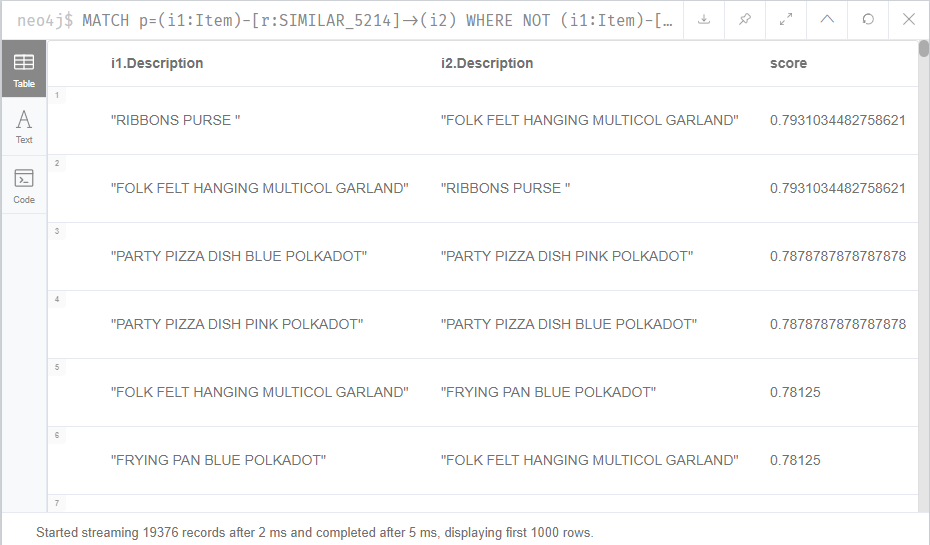
Bước 3: Tìm các cặp Item tương đồng được mua bởi cộng đồng 5214 mà không được mua bởi cộng đồng 5421

MATCH p=(i1:Item)-[r:SIMILAR\_5214]->(i2)

WHERE NOT (i1:Item)-[r:SIMILAR\_5421]->(i2)

AND r.score > .1 AND r.score < .8

RETURN i1.Description, i2.Description, r.score as score order by score desc



Bước 4: Tìm các cặp Item tương đồng được mua bởi cộng đồng 5421 mà không được mua bởi cộng đồng 5214

MATCH p=(i1:Item)-[r:SIMILAR\_5421]->(i2)

WHERE NOT (i1:Item)-[r:SIMILAR\_5214]->(i2)

AND r.score > .1 AND r.score < .8

RETURN i1.Description, i2.Description, r.score as score order by score desc



Từ 2 câu query trên ta có thể thấy rằng dường như khách hàng thuộc cộng đồng 5421 và cộng đồng 5214 có xu hướng mua hàng khá giống nhau. Tiếp theo ta sẽ tìm hiểu xem ở 2 cụm cộng đồng này liệu rằng xu hướng mua ấy liệu có hoàn toàn giống nhau hay không. Và nếu là giống nhau thì ta có nên gom hai cộng đồng này thành một hay không.

Bước 5: Tìm những Item có pagerank\_5214 lớn nhưng lại có pagerank\_5421 thấp

MATCH (i:Item)

WHERE i.PageRank\_Community\_5214 > 1.5

AND i.PageRank\_Community\_5421 < .75

RETURN i.Description, i.PageRank\_Community\_5214, i.PageRank\_Community\_5421



Từ câu truy vấn trên, ta có thể nhận xét rằng ở 2 cụm cộng đồng 5214 và 5412 vẫn có nhiều item mà điểm pagerank của chúng rất cao. Nghĩa là tầm ảnh hưởng của Item này ở cộng đồng 5214 rất lớn nhưng Item này khi ở cộng đồng 5421 thì tầm ảnh hưởng thì lại rất thấp. Vậy ta có thể kết luận rằng hành vi mua hàng ở 2 cụm cộng đồng 5214 và 5412 vẫn có điểm riêng biệt, không hoàn toàn giống nhau. Và chúng ta đã biết được các sản phẩm thật sự đặc biệt ở hai cụm cộng đồng này. Từ đây ta có thể có các chiến lược bán hàng với các sản phẩm này. Tiếp theo chúng ta sẽ tìm xem ở mỗi cụm cộng đồng thì những sản phẩm nào sẽ có tầm ảnh hưởng cao nhất, và thấp nhất. Và liệu rằng cửa hàng có nên loại bỏ những sản phẩm có tầm ảnh hưởng thấp hay không.

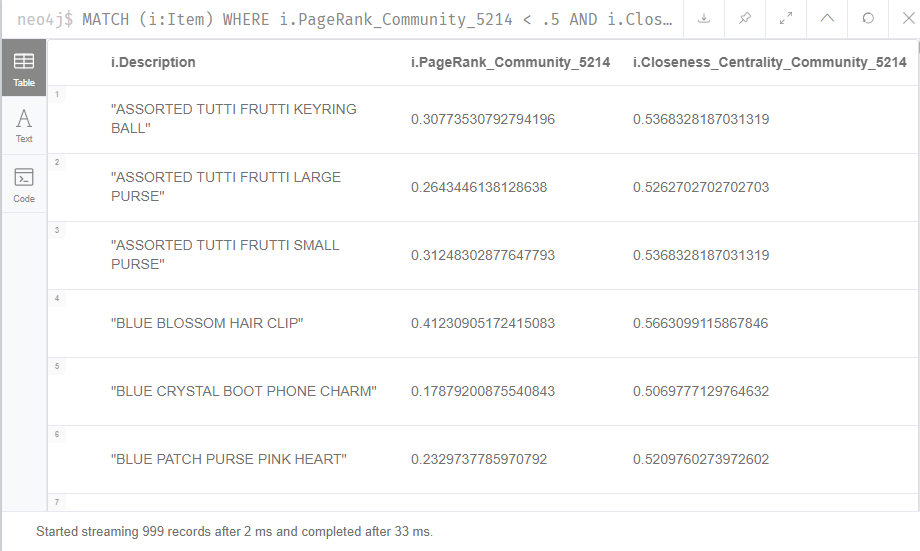
Bước 6: Tìm các Item có điểm Closeness cao nhưng Pagerank thấp ở cộng đồng 5214:

MATCH (i:Item)

WHERE i.PageRank\_Community\_5214 < .5

AND i.Closeness\_Centrality\_Community\_5214 > .5

RETURN i.Description, i.PageRank\_Community\_5214, i.Closeness\_Centrality\_Community\_5214



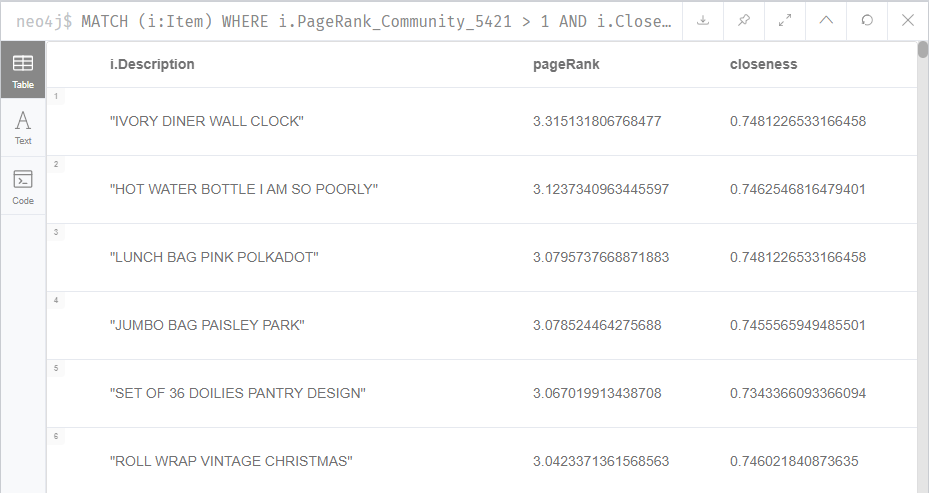
Bước 7: Tìm các Item có điểm Closeness thấp nhưng Pagerank cao ở cộng đồng 5214:

MATCH (i:Item)

WHERE i.PageRank\_Community\_5421 > 1

AND i.Closeness\_Centrality\_Community\_5421 < 0.75

RETURN i.Description, i.PageRank\_Community\_5421 as pageRank, i.Closeness\_Centrality\_Community\_5421 as closeness order by pageRank DESC



Từ 2 câu query trên ta có thể nhận thấy rằng. Điểm closeness càng cao thì các node ở có đường đi ngắn nhất đến các node khác càng ngắn. Vậy những món hàng có điểm closeness cao và pagerank thấp là những món hàng mà bất kỳ ai cũng có thể mua, cũng như tiện tay là gom khi ở cửa hàng. Như những món đồ phụ kiện, trang sức như kẹp tóc,móc điện thoại… những món hàng này đều có số lượng lớn và dễ dàng tìm thấy ở bất kỳ cửa tiệm nào. Trong khi đó những món hàng với điểm pagerank cao và closeness thấp. Là những món hàng tuy cần thiết nhưng số lượng của chúng khi ta mua ở cửa hàng thì cực kỳ thấp, chỉ cần một hoặc hai món hàng này. Đó là đồng hồ, bình giữ nhiệt, giỏ đựng đồ ăn trưa hay đồ trang trí giáng sinh. Như vậy, ta có thể nhận ra rằng những món đồ có page rank hay closeness thấp đều quan trọng. Tùy nhu cầu của mỗi khách hàng mà tầm quan trọng của các món đồ này cũng khác nhau. Từ những đặc điểm này ta có thể đưa ra những chiến lược như gom những item bán chạy lại thành 1 set, khuyến mãi combo. Còn những sản phẩm cần thiết nhưng khó bán, thì cần vận động bán hàng bằng cách giảm giá hay tạo ra các ưu đãi như mua một tặng một …

# CHƯƠNG 4: KẾT LUẬN

### Những điều đã làm được:

* Tìm hiểu về Neo4j và một số thuật toán, độ đo của nó
* Gom cụm thành các cộng đồng riêng biệt
* Thực hiện các truy vấn và trả lời được các câu hỏi như :
  + Phân tích hành vi của một số cộng đồng khách hàng
  + Có thể recommend cho khách hàng nên mua những món hàng nào
  + Từ những phân tích về các món hàng trong mỗi cộng đồng có thể từ đó đưa ra những chiến lược khuyến mãi.

### Hạn chế:

* Chưa xử lý được hết tất cả cộng đồng
* Dữ liệu phân tích được chỉ ở mức chung chung. Chưa làm rõ ràng được.
* Chưa thể nêu rõ về giải thuật Louvain.

# CHƯƠNG 5: TÀI LIỆU THAM KHẢO

[1] Neo4j Documentary <https://neo4j.com/docs/graph-data-science/current/introduction/>

[2] Dataset: <https://archive.ics.uci.edu/ml/datasets/Online+Retail+II>