GRAPH THEORY AND ITS APPLICATIONS

ASSIGNMENT NAME:

Flipkart Purchase Using Neo4j

.

Flipkart Purchase USING NEO4J

The aim of this assignment is to build a recommendation system which recommends Products to Customers based on the dataset using neo4j.

Introduction To The Dataset

Flipkaít is one of India's leading e-commeíce maíketplaces. Flipkaít has 100 million íegisteíed useís and moíe than a million selleís on its electíonic commeíce platfoím. l'o ensuíe píompt deliveíy to its customeís, the company has invested in setting up waíehouses in 21 states.

It has an enormous amount of data/Products that can be used to make an optimal model. We have used a raw data set of Flipkart. This data set contains 12 different features that can be used for recommending an app to a user.

The different columns in the csv file are _id, _labels, name, lastname, Reviews, Size, Installs, Type, Price, _start, _end, _type, rating.

1. The column "_id" provides the id of a particular row.
2. The column "_labels" provides the label for the given name.It can be either :Product or :Customer or :Category.
3. The column "name" provides the name of either Product or Customer or Category.
4. The column "Review" provides the number of reviews for an app.
5. The column "Size" provides the size of an Product.
6. The colums "Purchased" provides the total number of Purchases of a particular Product.
7. The column "Type" provies the type of app, which is either free or paid.
8. The column "Price" provides the price of the Product.
9. The column "_start" is used for mapping of relationships, where _start contains the id of thestarting(source) node.
10. The column "_type" provides the type of relationship, which can be either HAS_Product or HAS_CATEGORY
11. The column "_end" is used for mapping of relationships, where _end contains the id of the ending(destination) node.
12. The column "rating"is used along with a ralationship where gives the rating a user has given for a particular app.

Products will be recommended to users by using both Content-Based Filtering and Collaborative Filtering .

Content-based filtering

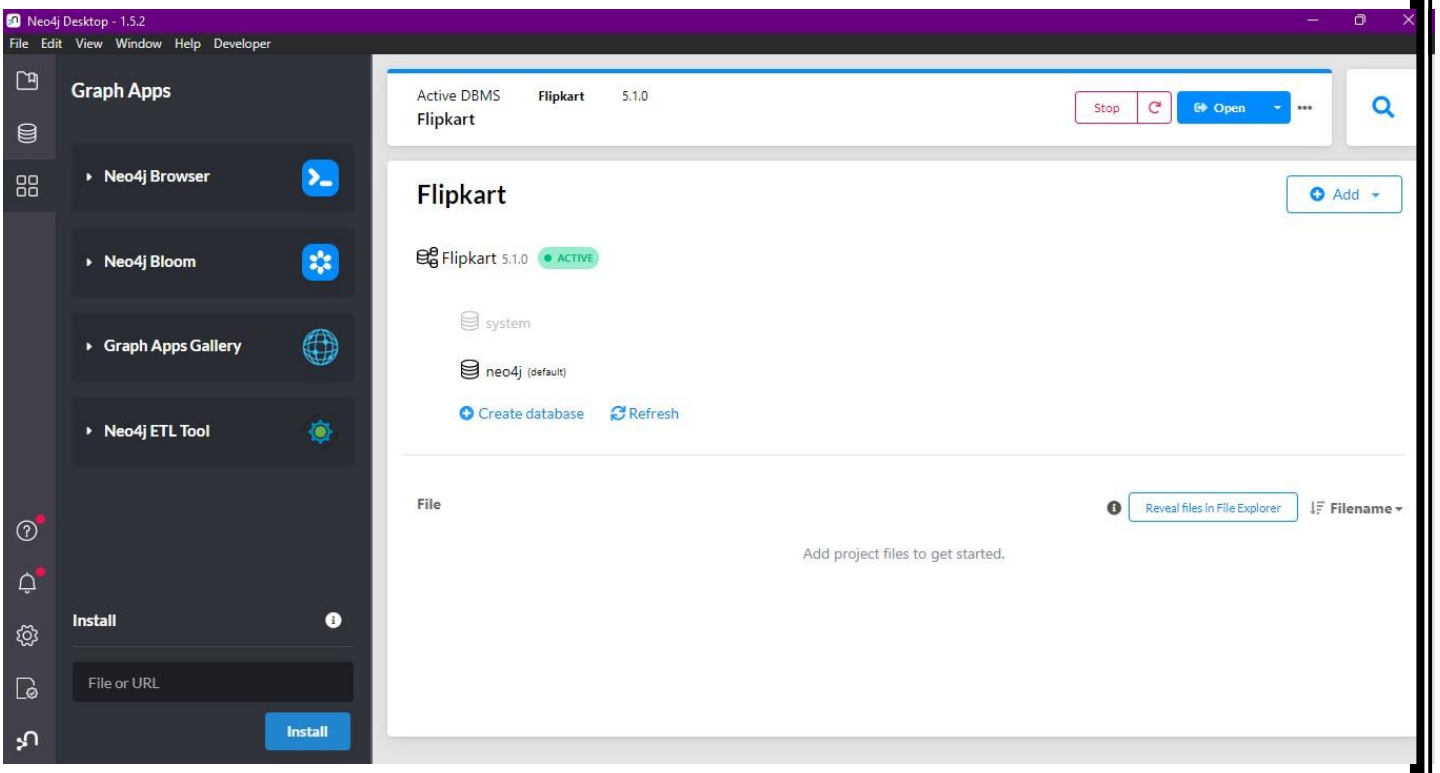
Content-based filtering uses item features to recommend other products similar to what the user likes,based on their previous actions or explicit feedback.

Collaborative filtering

Collaborative filtering recommends a user the products on the basis of the preferences of the other users with similar tastes.

Neo4j commands for constructing a graph based on the datasetStep1:

First create a project in Neo4j



Step2: Create nodes based on the dataset

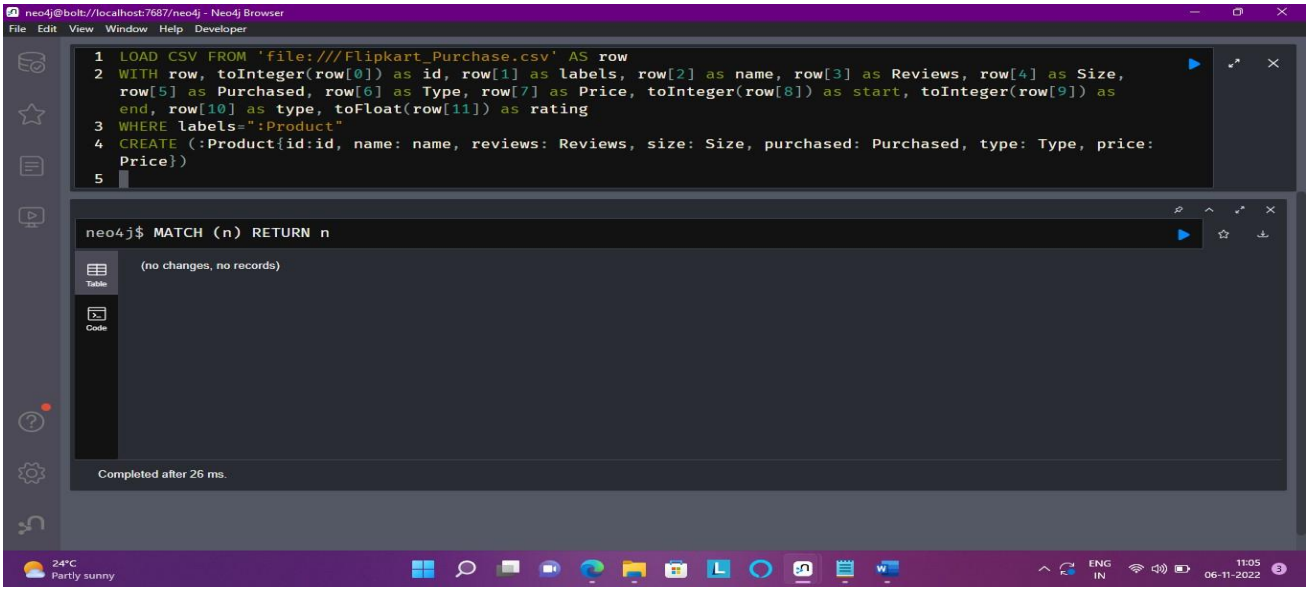
Cypher Query for Creating “Product” nodes

LOAD CSV FROM 'file:///Flipkart_Purchase.csv' AS row

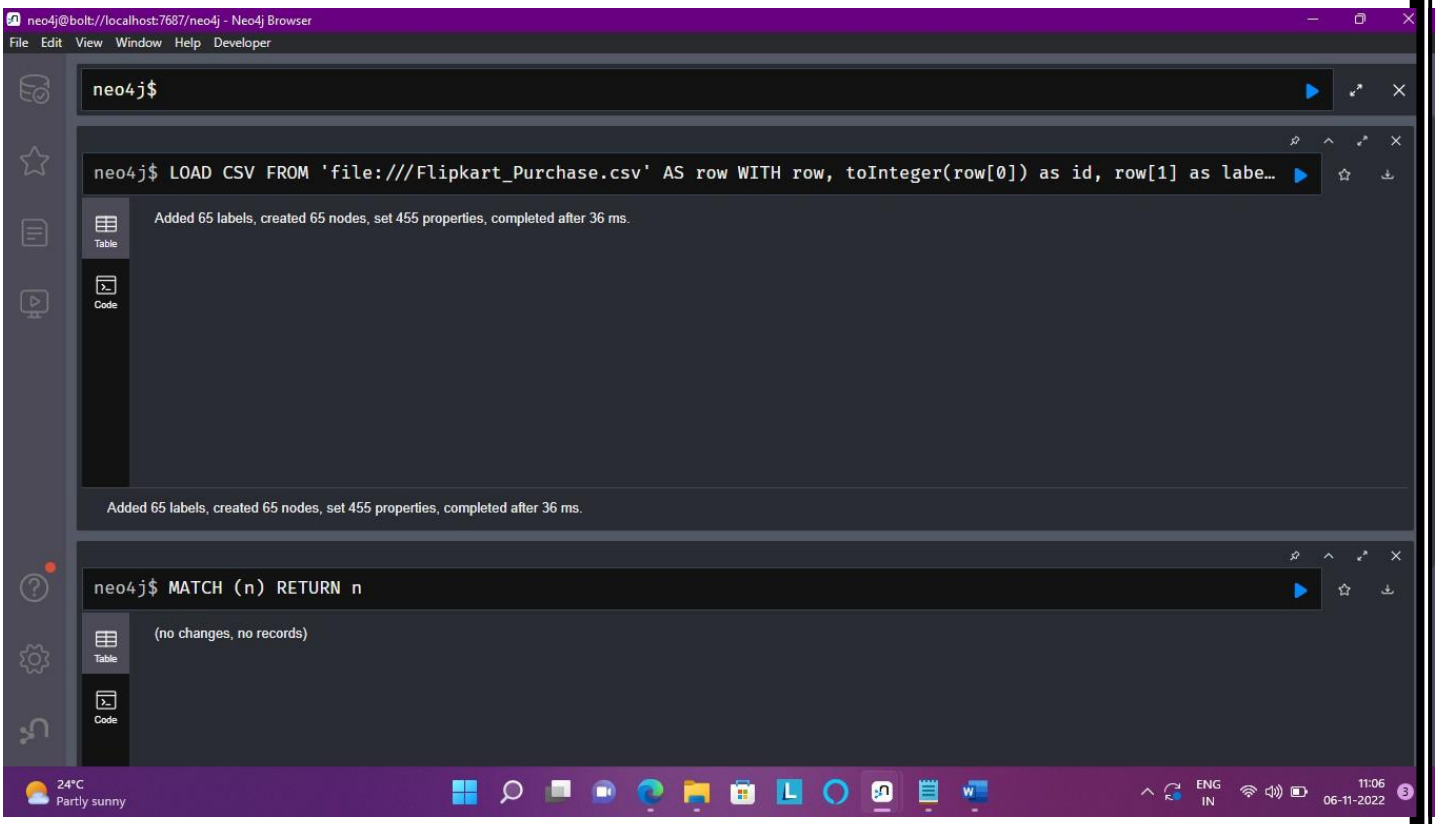
WITH row, toInteger(row[0]) as id, row[1] as labels, row[2] as name, row[3] as Reviews, row[4] as Size, row[5] as Purchased, row[6] as Type, row[7] as Price, toInteger(row[8]) as start, toInteger(row[9]) as end, row[10] as type, toFloat(row[11]) as rating

WHERE labels=":Product"

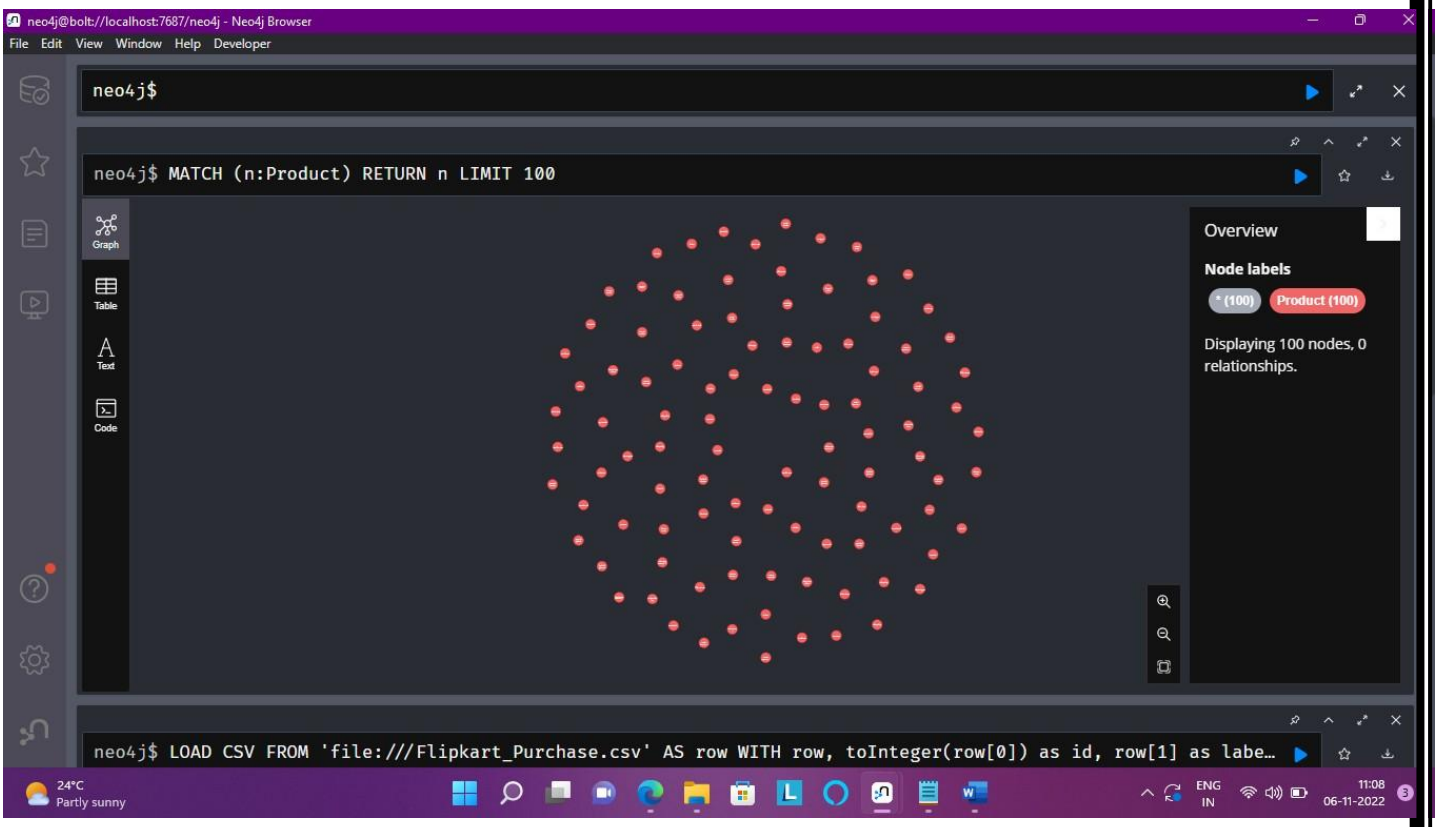
CREATE (:Product{id:id, name: name, reviews: Reviews, size: Size, purchased: Purchased, type: Type, price: Price})



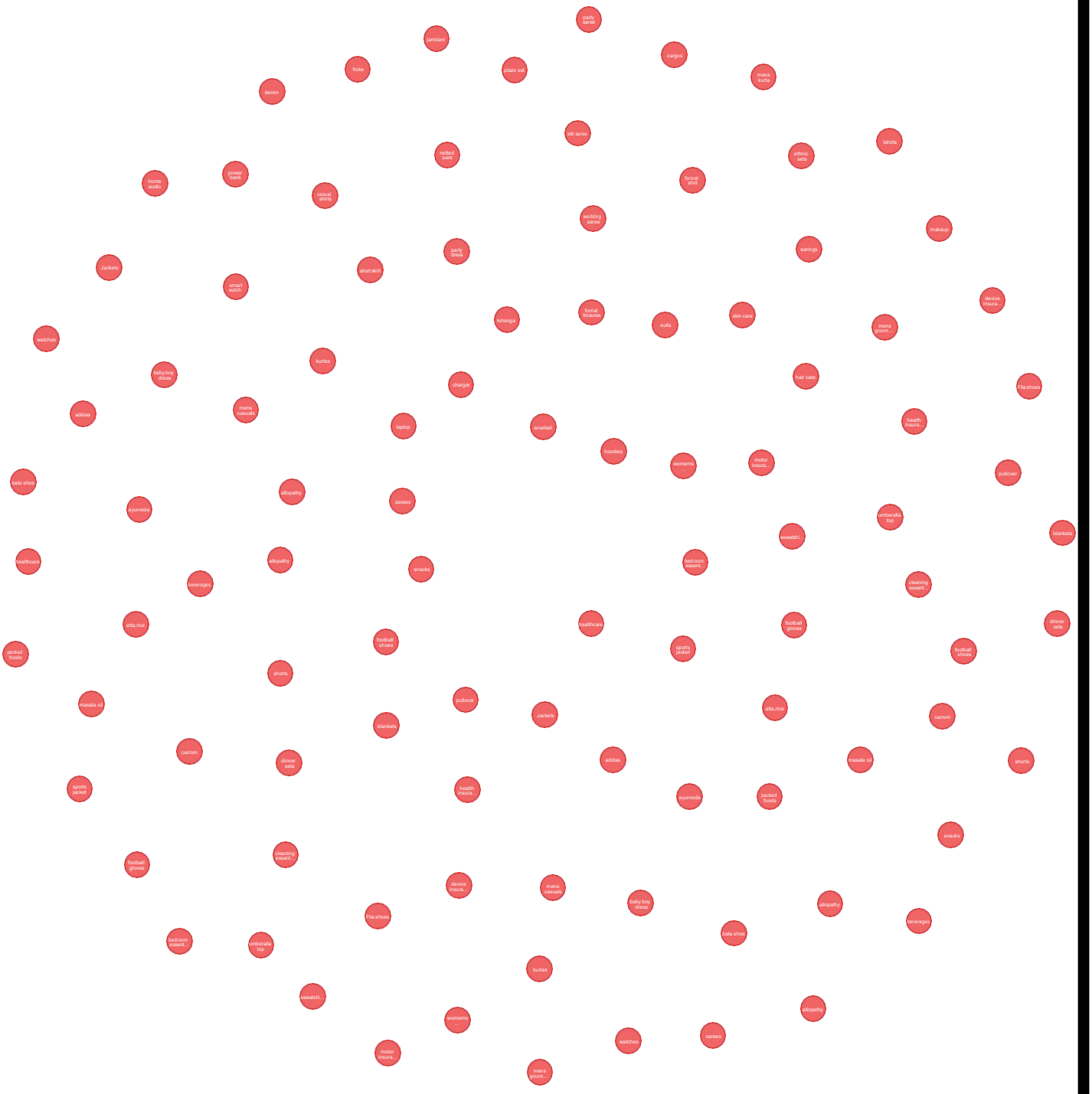
Output:



Output graph

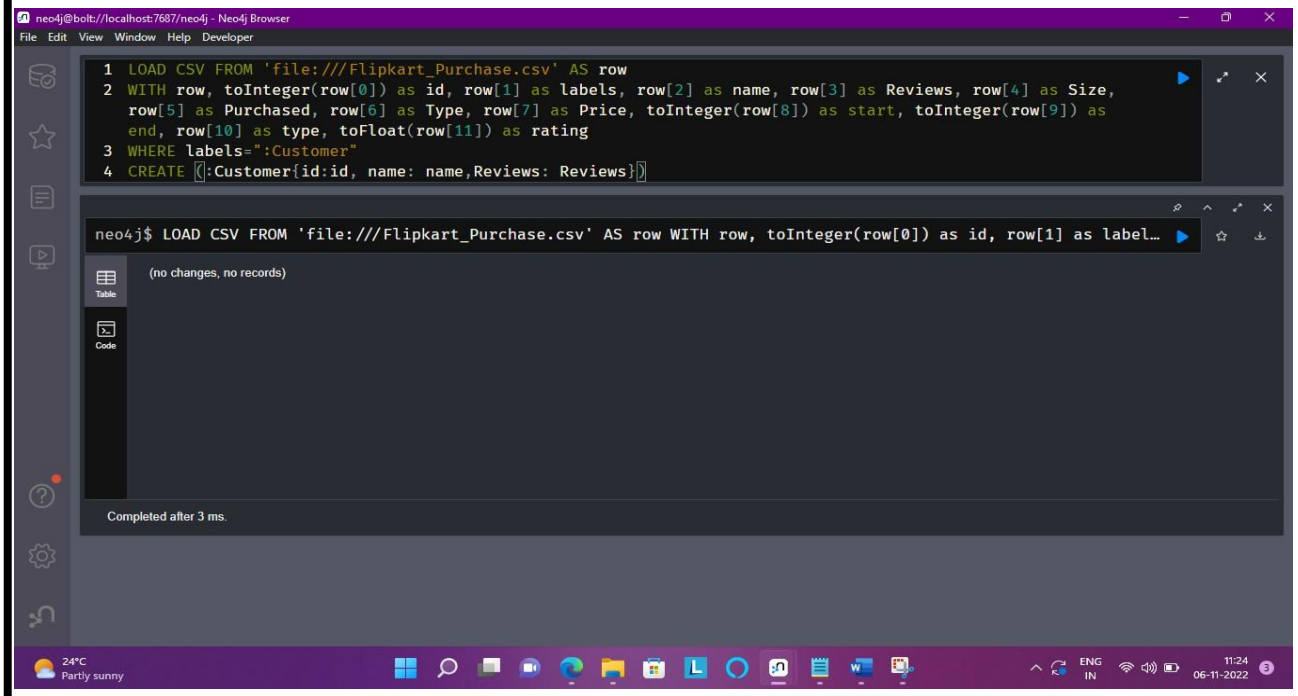


Exported graph:

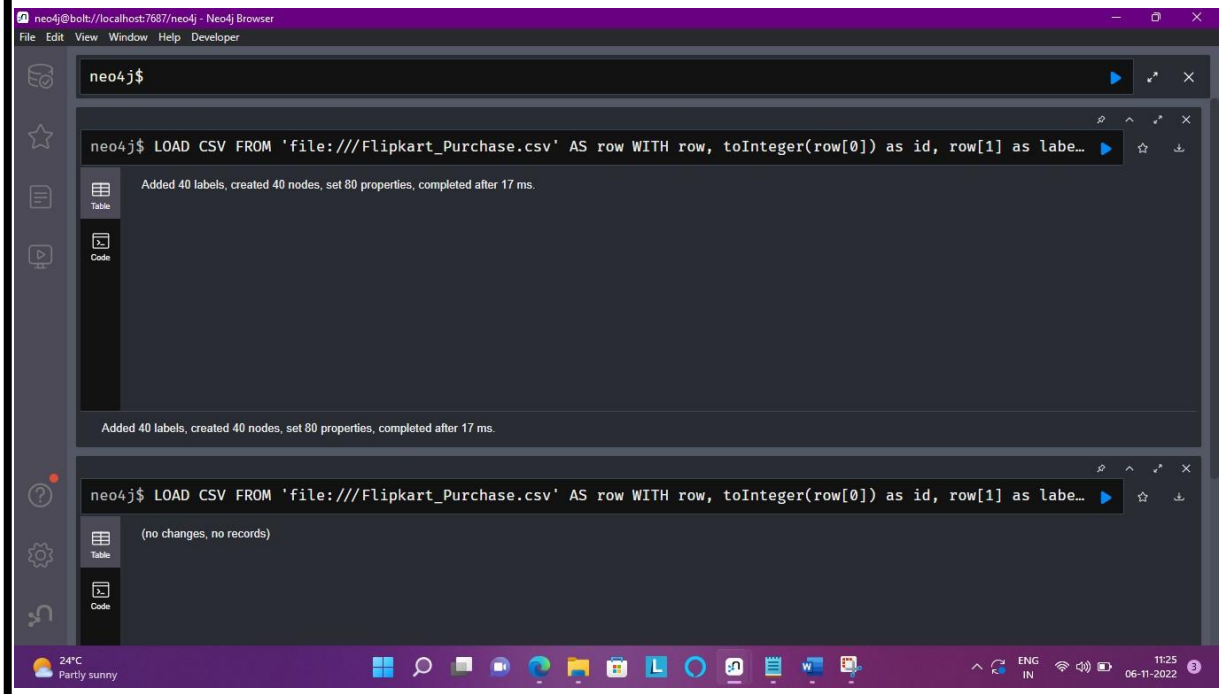


Cypher Query for Creating User Nodes

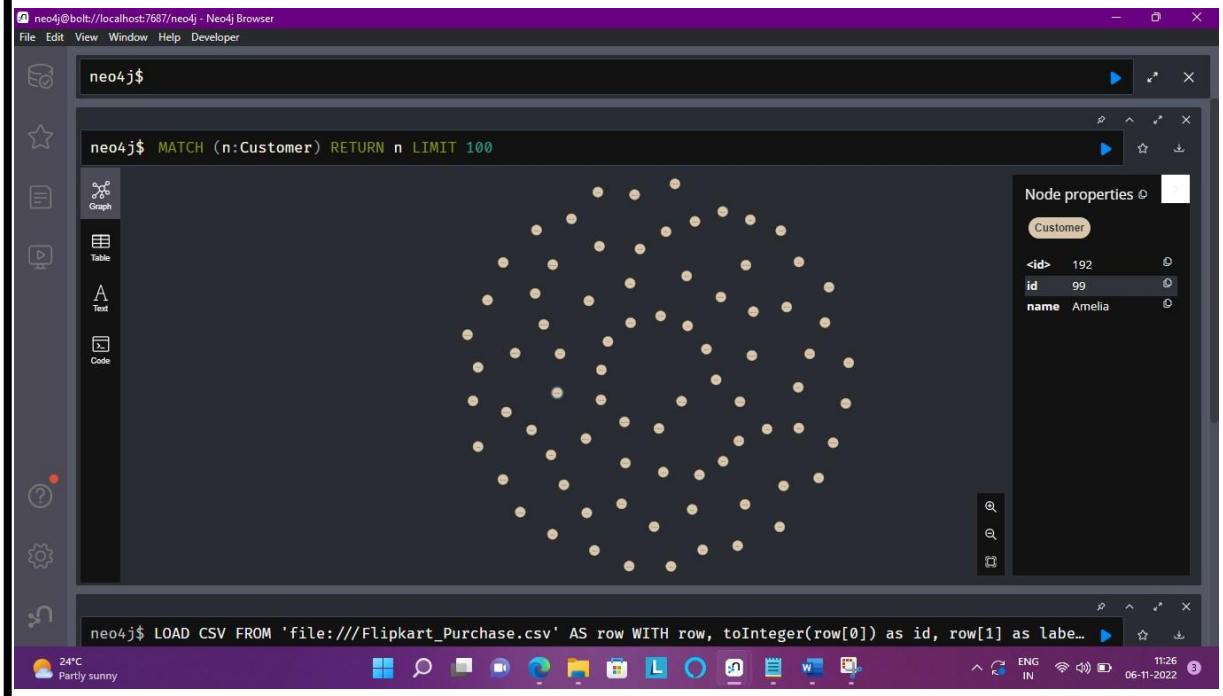
```
LOAD CSV FROM 'file:///Flipkart_Purchase.csv' AS row
WITH row, toInteger(row[0]) as id, row[1] as labels, row[2] as name, row[3] as Reviews, row[4] as Size, row[5] as Purchased, row[6] as Type, row[7] as Price, toInteger(row[8]) as start, toInteger(row[9]) as end, row[10] as type, toFloat(row[11]) as rating
WHERE labels="::Customer"
CREATE (::Customer{id:id, name: name,Reviews: Reviews})
```



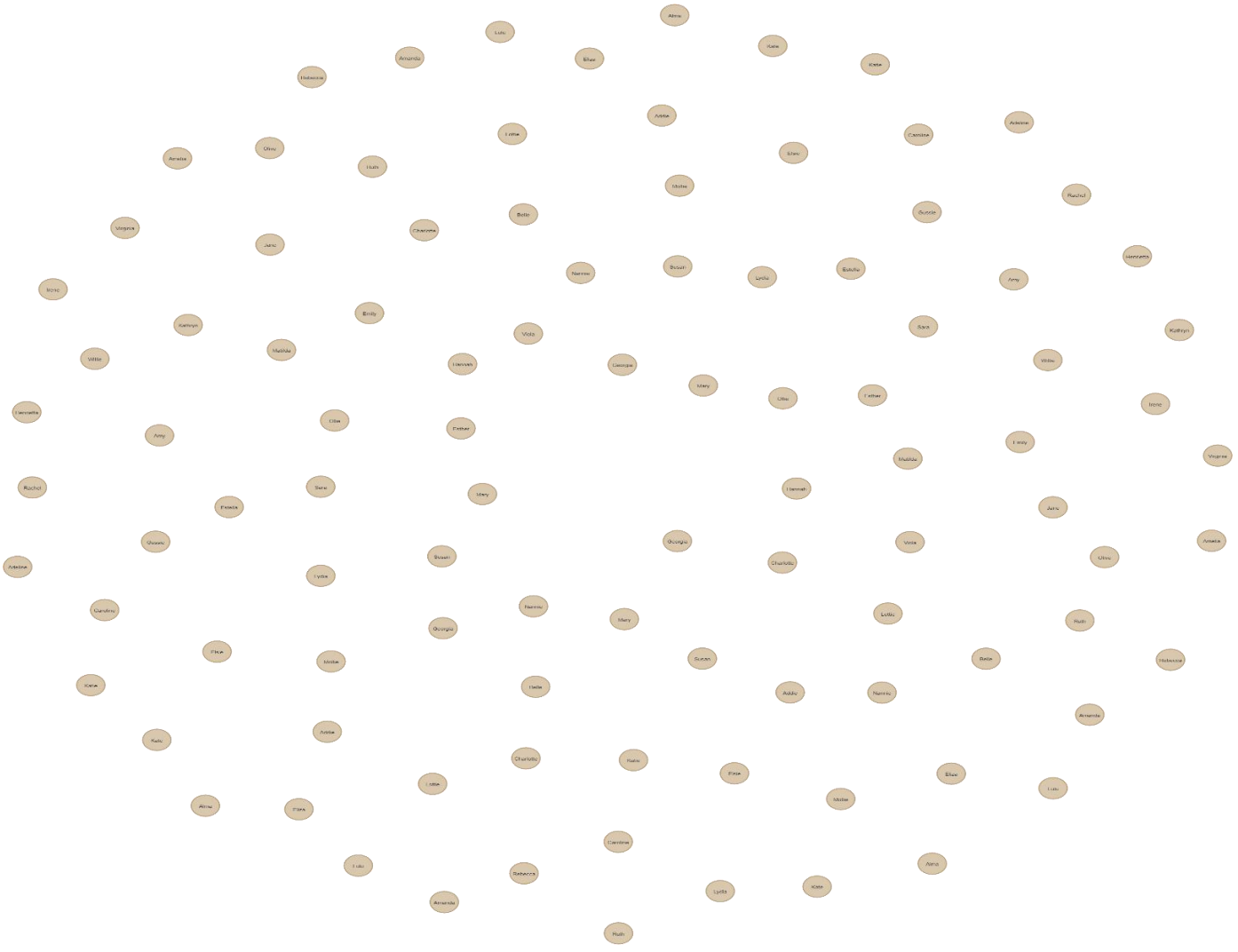
OUTPUT:



Output graph:

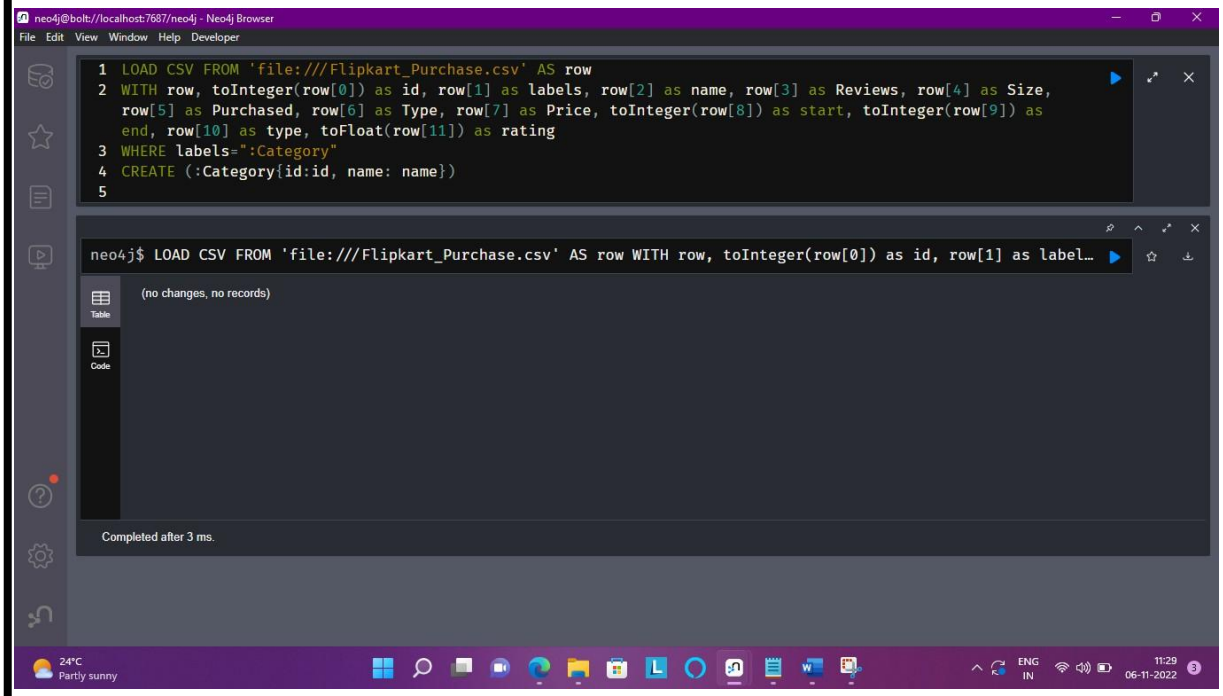


Exported graph:

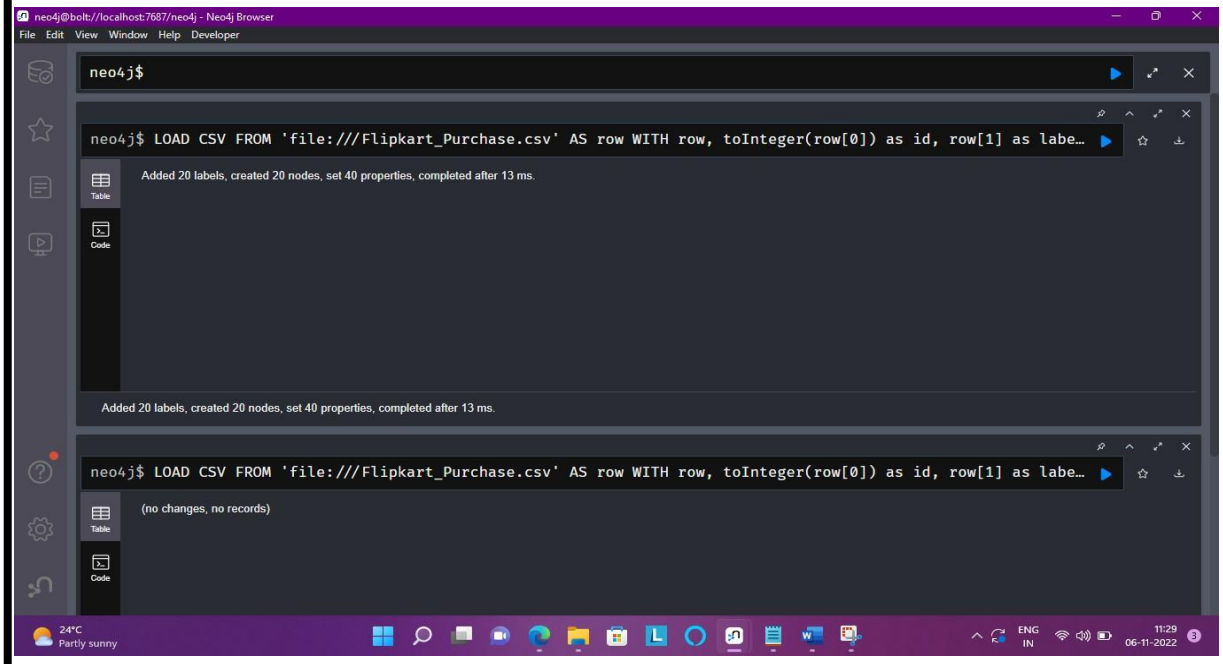


Cypher Query for Creating “Category” Nodes

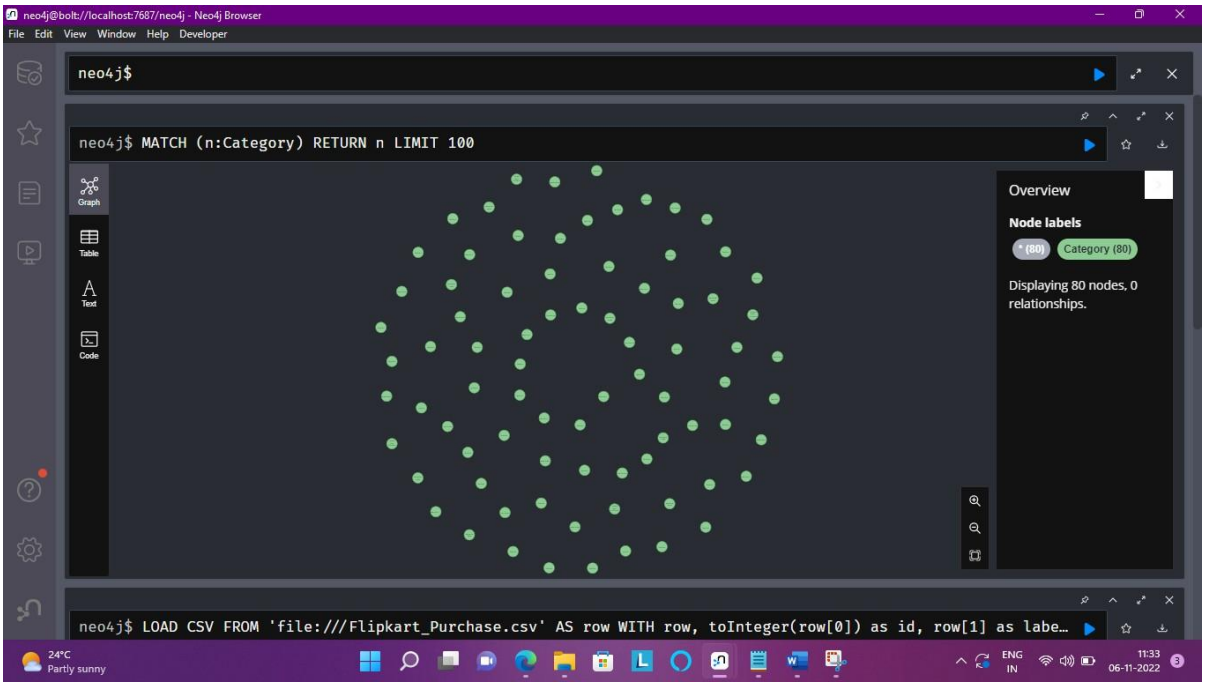
```
LOAD CSV FROM 'file:///Flipkart_Purchase.csv' AS row
WITH row, toInteger(row[0]) as id, row[1] as labels, row[2] as name, row[3] as Reviews, row[4] as Size, row[5] as Purchased, row[6] as Type, row[7] as Price, toInteger(row[8]) as start, toInteger(row[9]) as end, row[10] as type, toFloat(row[11]) as rating
WHERE labels=":Category"
CREATE (:Category{id:id, name: name})
```



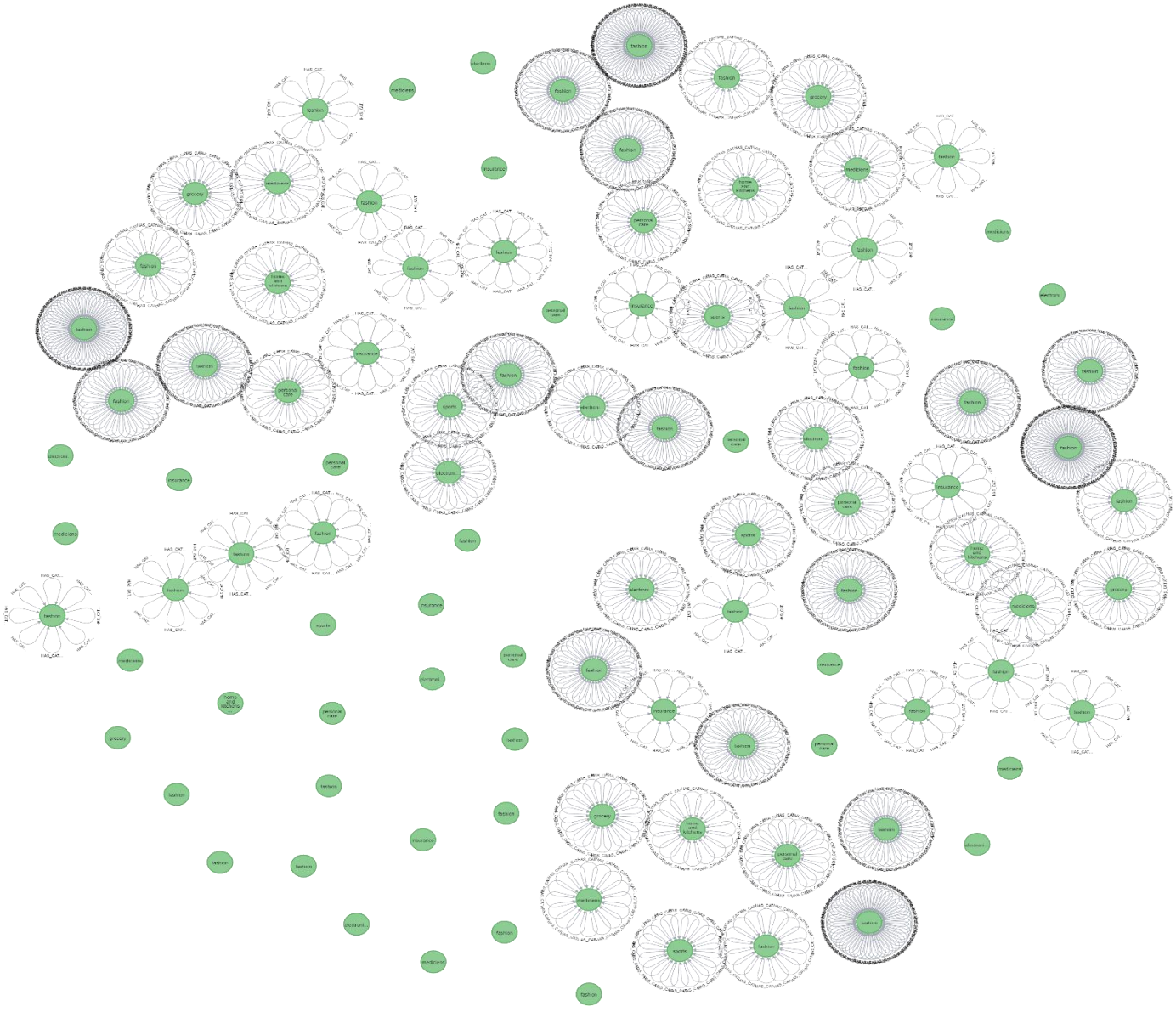
OUTPUT:



OUTPUT GRAPH:



Exported graph:



Step3: Create realtionships based on the dataset

Cypher Query for Creating relationship as “HAS_Product”

LOAD CSV FROM 'file:///Flipkart_Purchase.csv' AS row

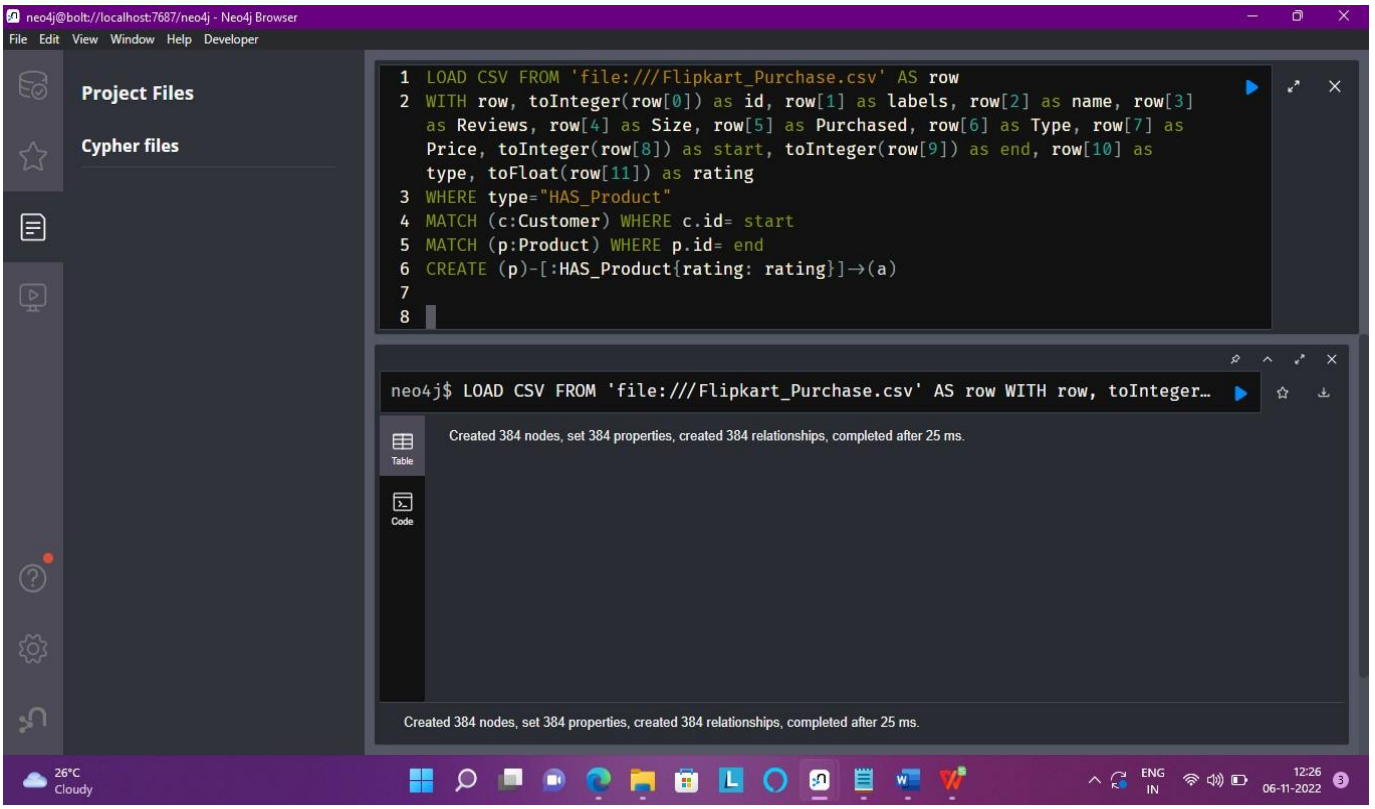
WITH row, toInteger(row[0]) as id, row[1] as labels, row[2] as name, row[3] as Reviews, row[4] as Size, row[5] as Purchased, row[6] as Type, row[7] as Price, toInteger(row[8]) as start, toInteger(row[9]) as end, row[10] as type, toFloat(row[11]) as rating

WHERE type="HAS_Product"

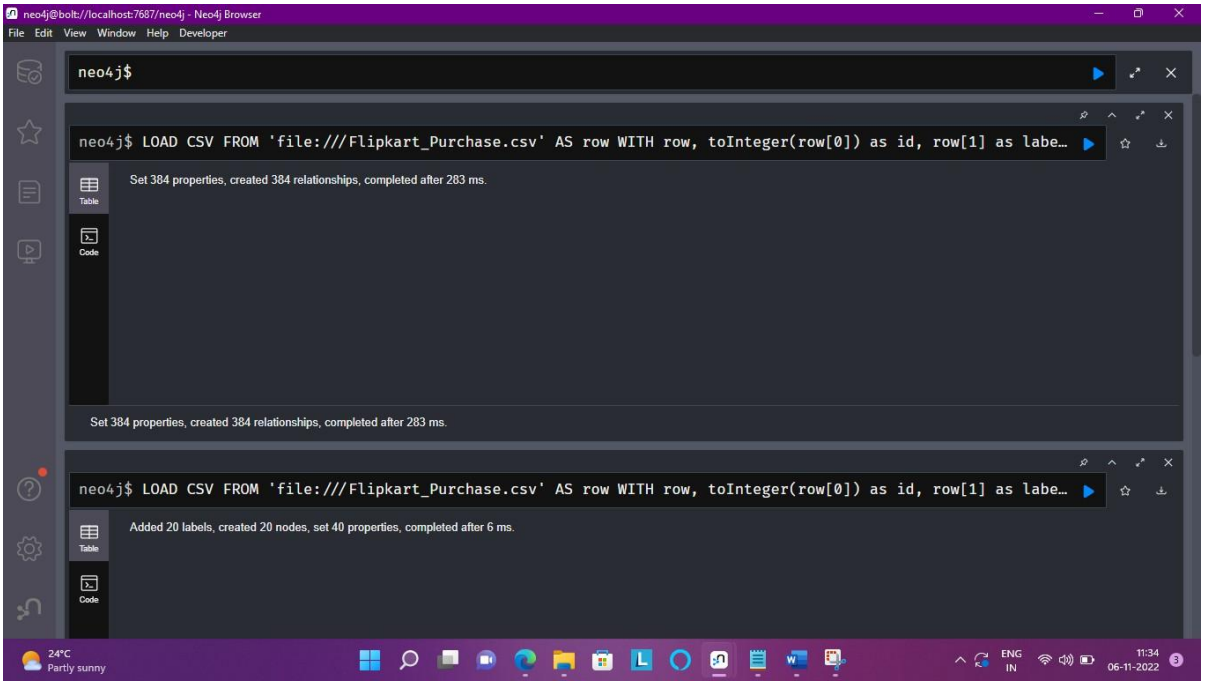
MATCH (c:Customer) WHERE c.id= start

MATCH (p:Product) WHERE p.id= end

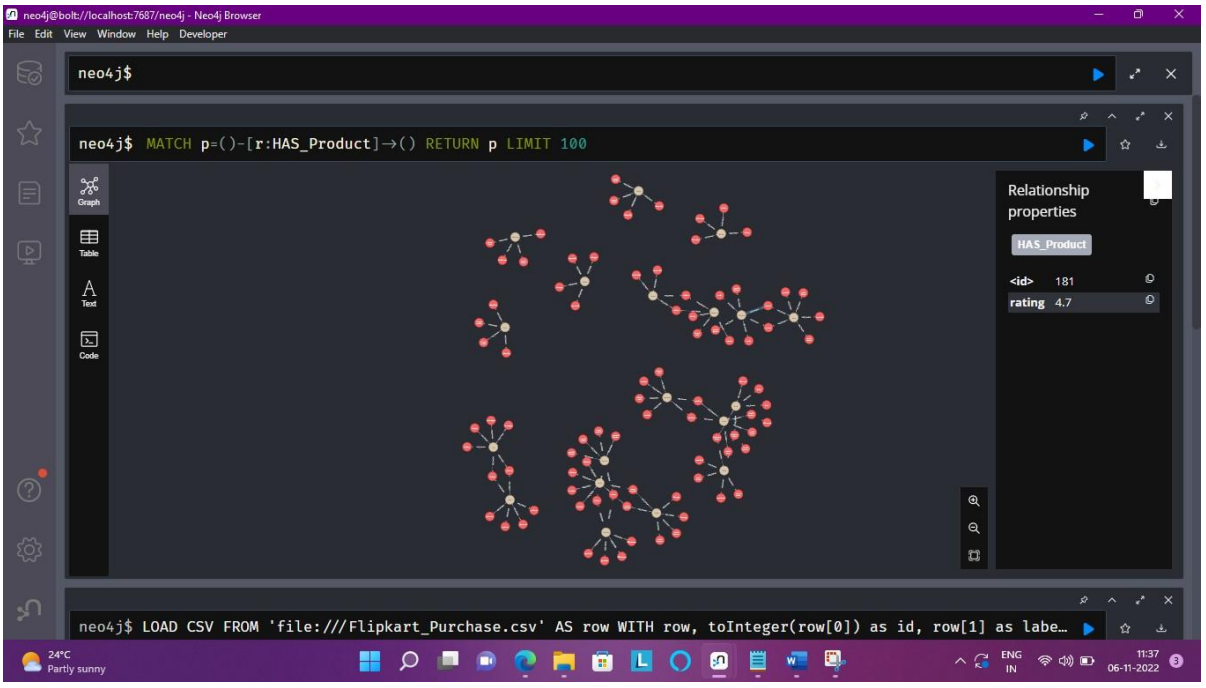
CREATE (p)-[:HAS_Product{rating: rating}]->(a)



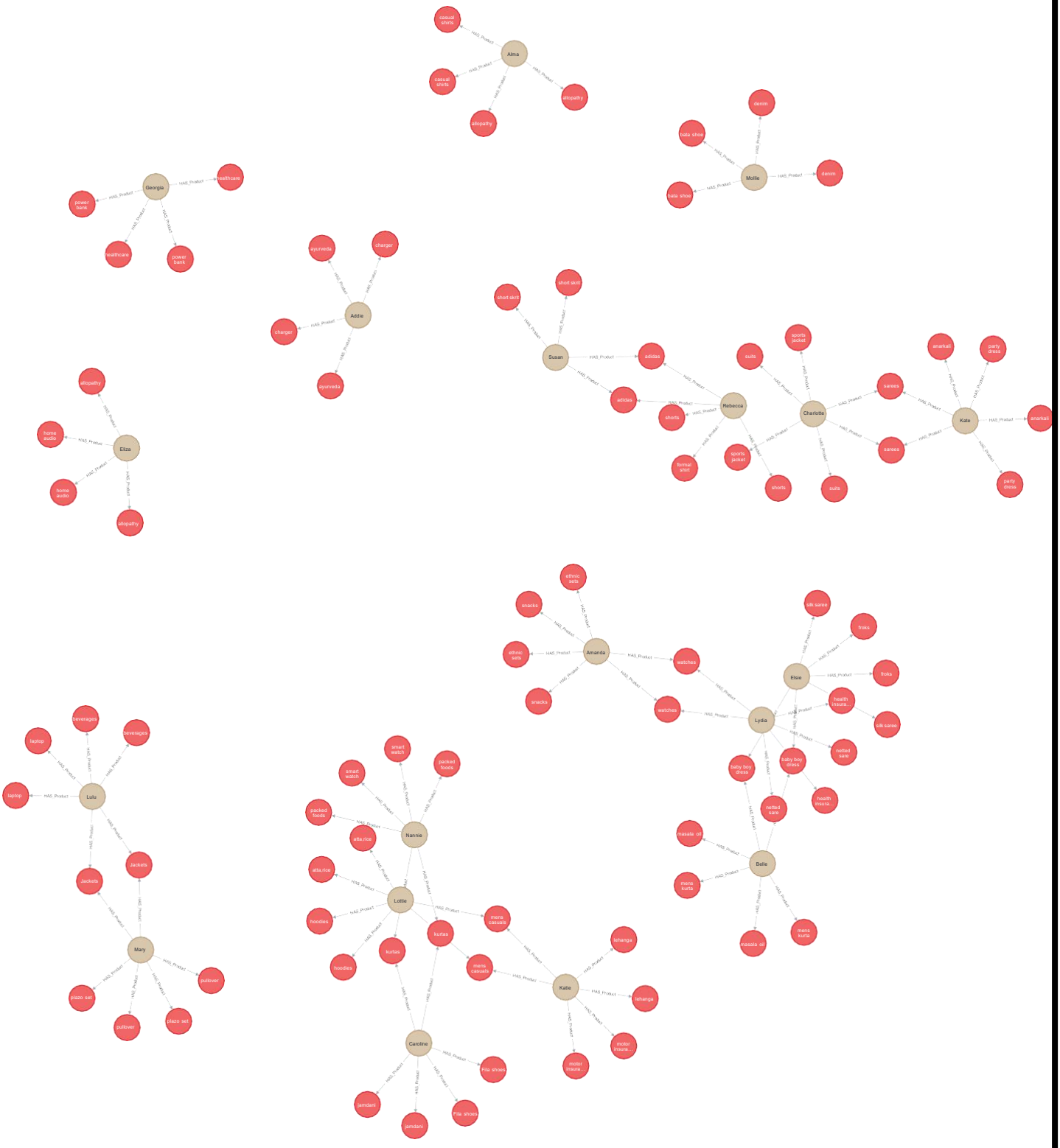
OUTPUT:



OUTPUT GRAPH:

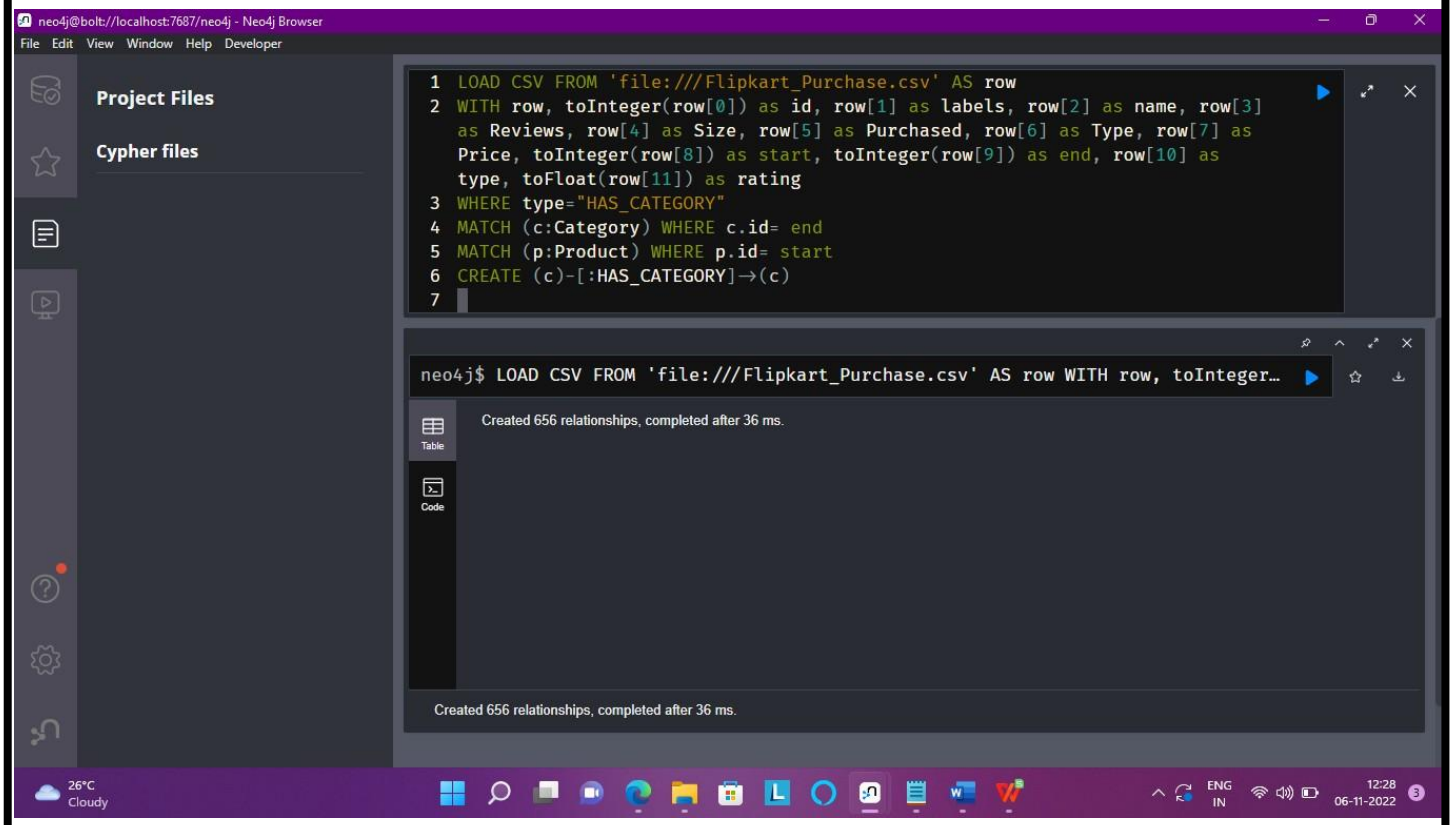


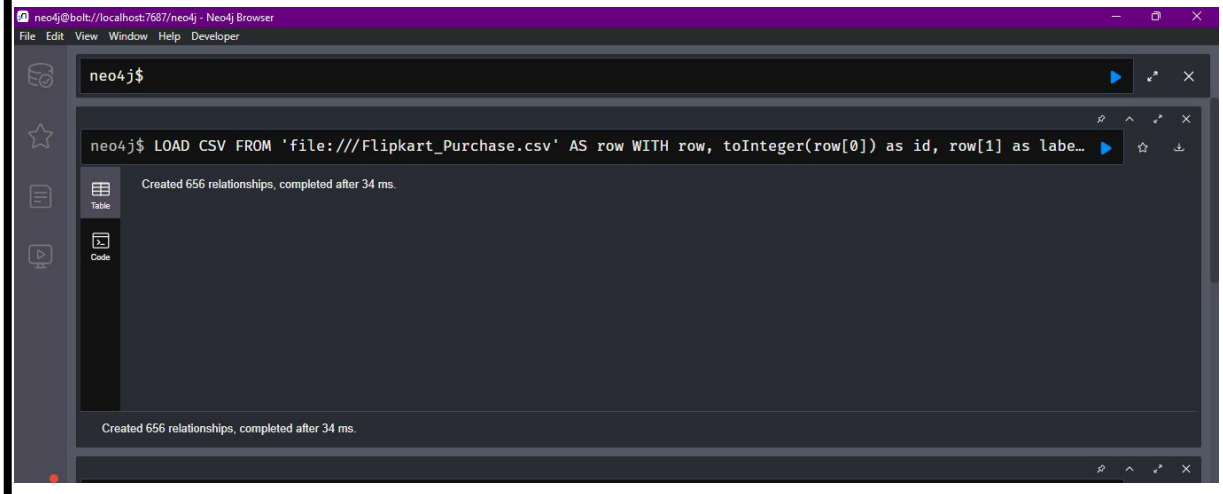
EXPORTED GRAPH:



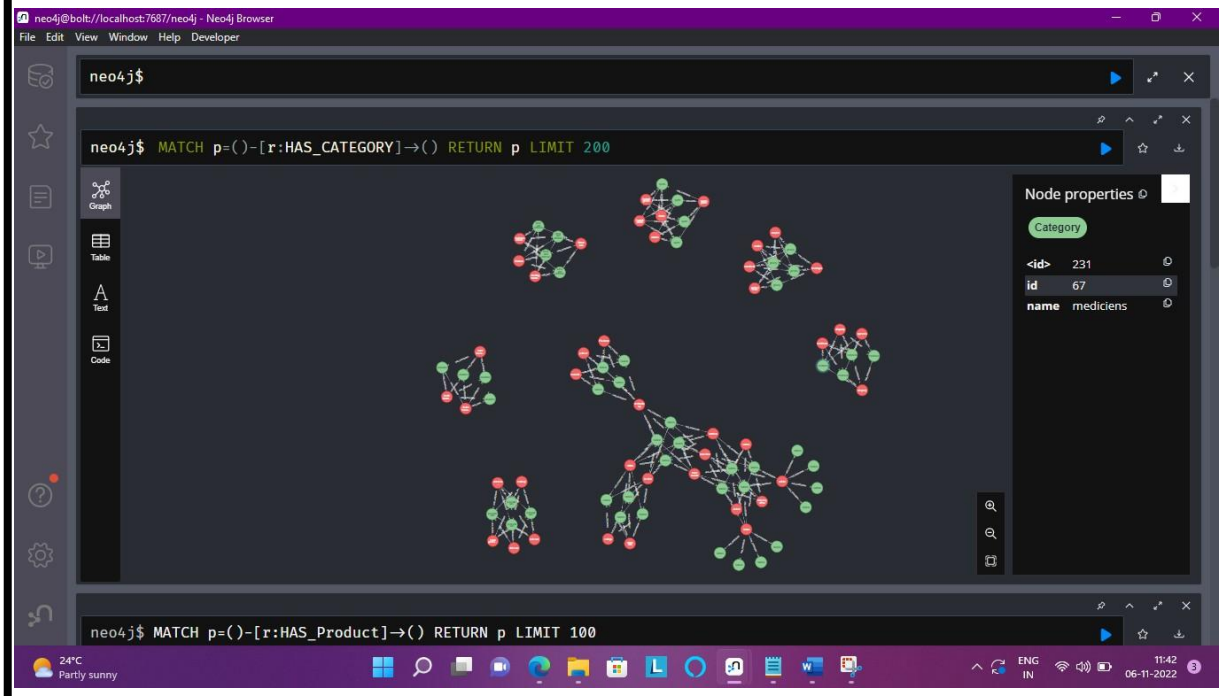
Cypher Query for Creating relationship as “HAS_CATEGORY”

```
LOAD CSV FROM 'file:///Flipkart_Purchase.csv' AS row
WITH row, toInteger(row[0]) as id, row[1] as labels, row[2] as name, row[3] as Reviews, row[4] as Size, row[5] as Purchased, row[6] as Type, row[7] as Price, toInteger(row[8]) as start, toInteger(row[9]) as end, row[10] as type, toFloat(row[11]) as rating
WHERE type="HAS_CATEGORY"
MATCH (c:Category) WHERE c.id= end
MATCH (p:Product) WHERE p.id= start
CREATE (c)-[:HAS_CATEGORY]->(c)
```

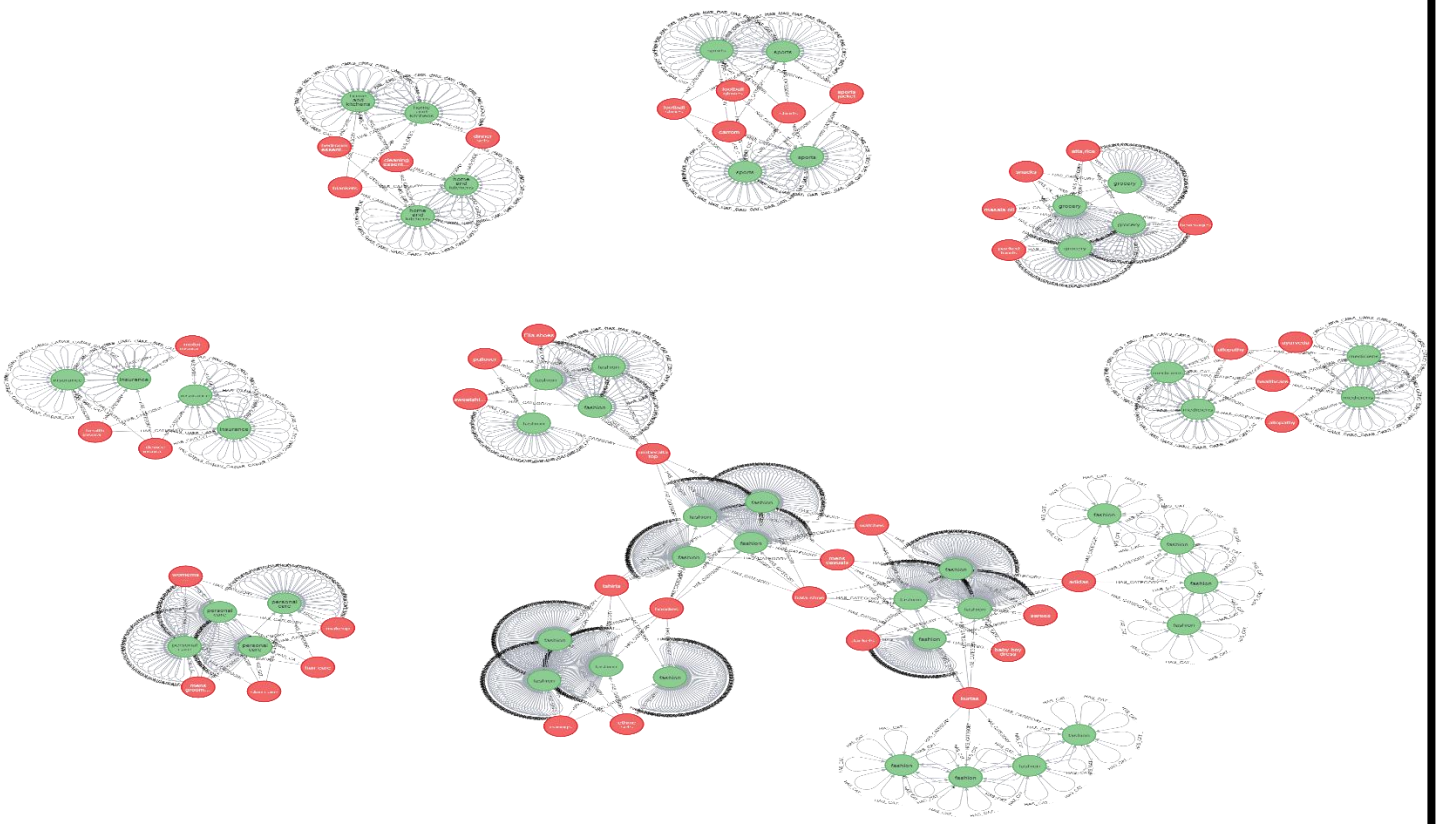




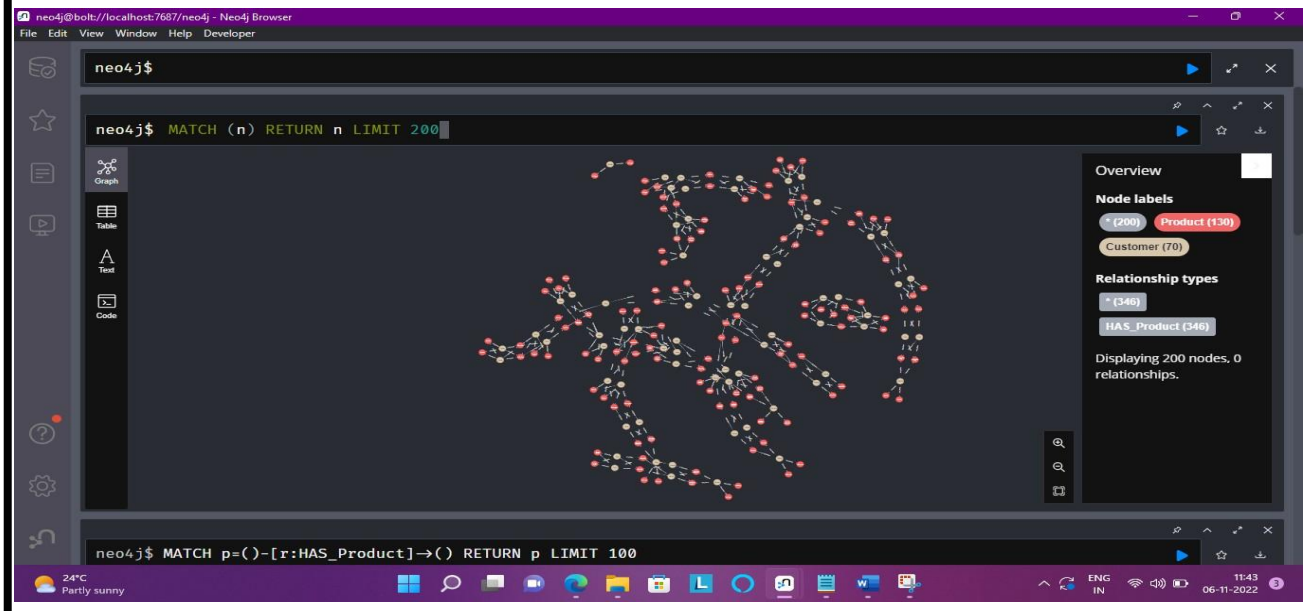
OUTPUT GRAPH:



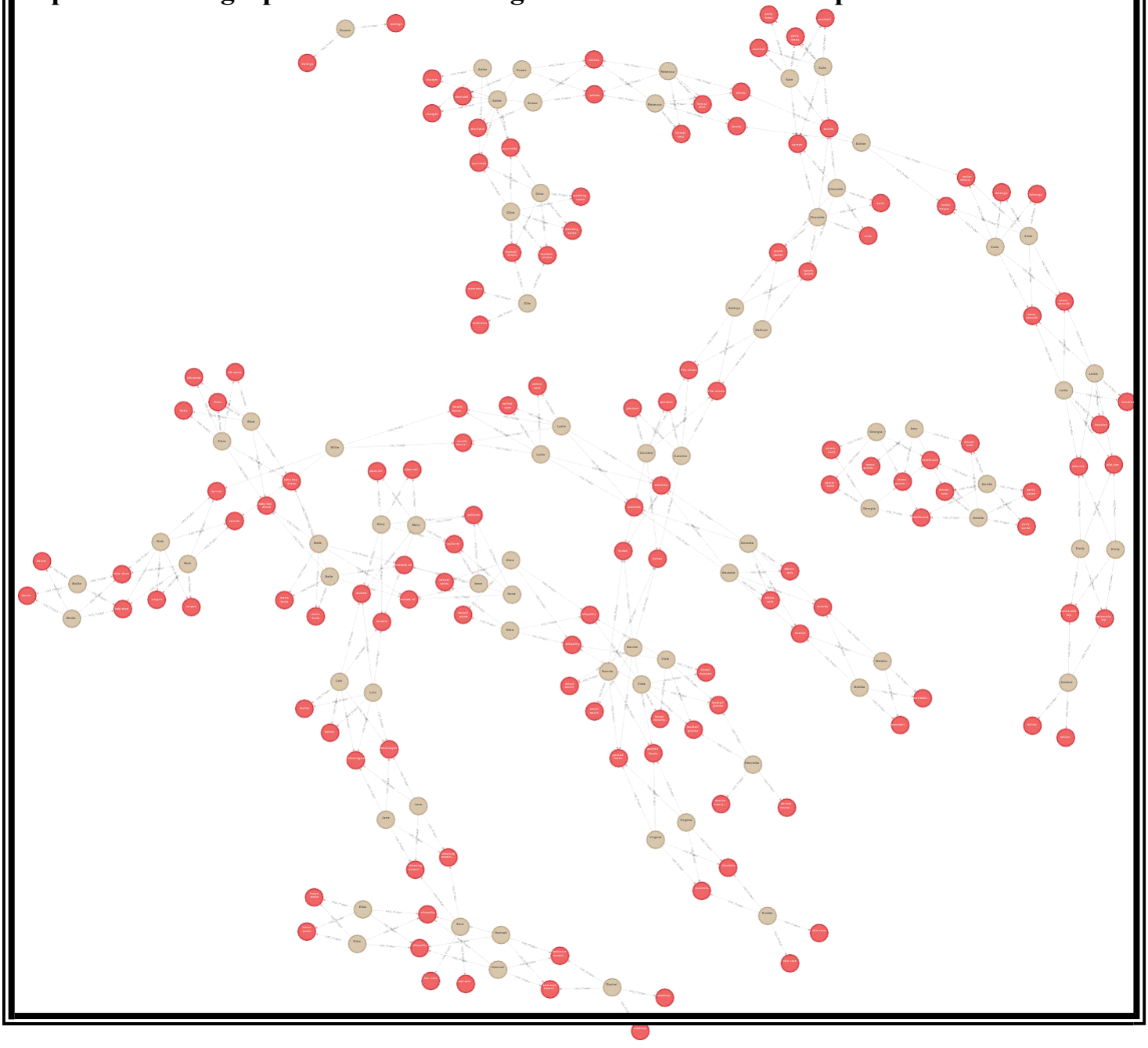
EXPORTED GRAPH:



Entire Graph screenshot:



Exported entire graph after constructing all nodes and relationships



Content- Based Filtering

Cypher query:

MATCH(u:Customer{name:"Belle", lastname:"Iona"})-[:HAS_Product]->(a:Product)-[:HAS_CATEGORY]->(c:Category)<-[:HAS_CATEGORY]-(z:Product)

WHERE NOT EXISTS ((u)-[:HAS_Product]->(z))

WITH a, z, COUNT(c) AS intersection

MATCH (a)-[:HAS_CATEGORY]->(ac:Category)

WITH a, z, intersection, COLLECT(ac.name) AS a1

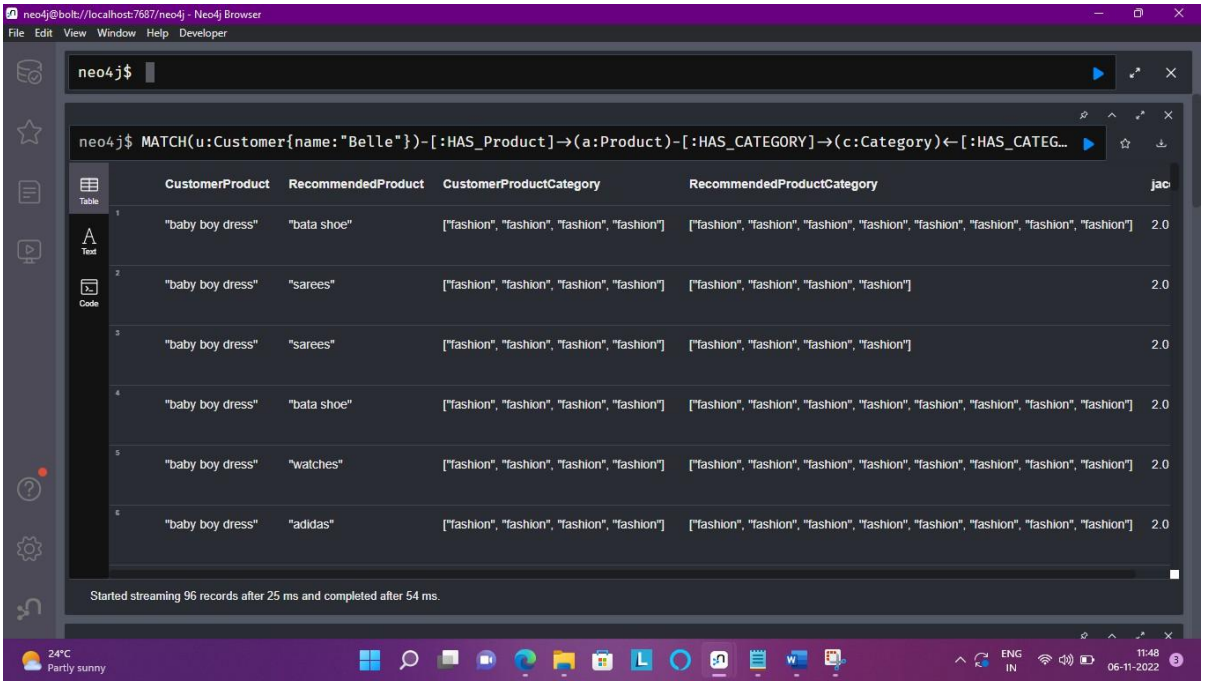
MATCH (z)-[:HAS_CATEGORY]->(zc:Category)

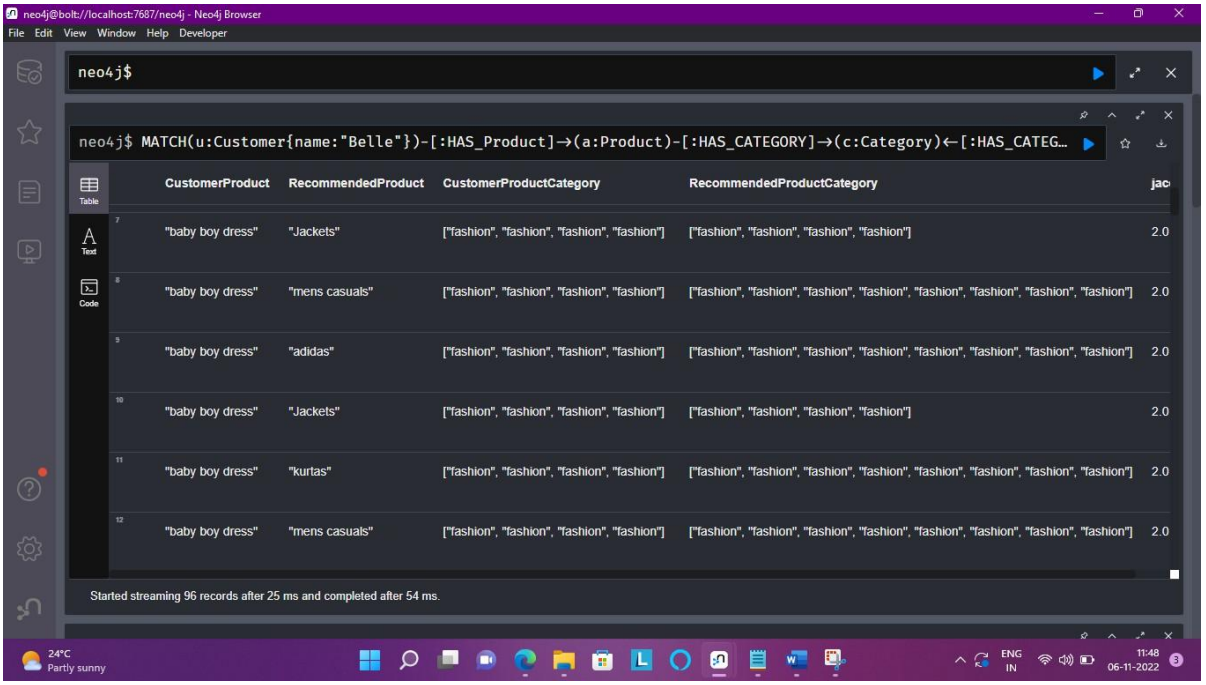
WITH a, z, a1, intersection, COLLECT(zc.name) AS a2

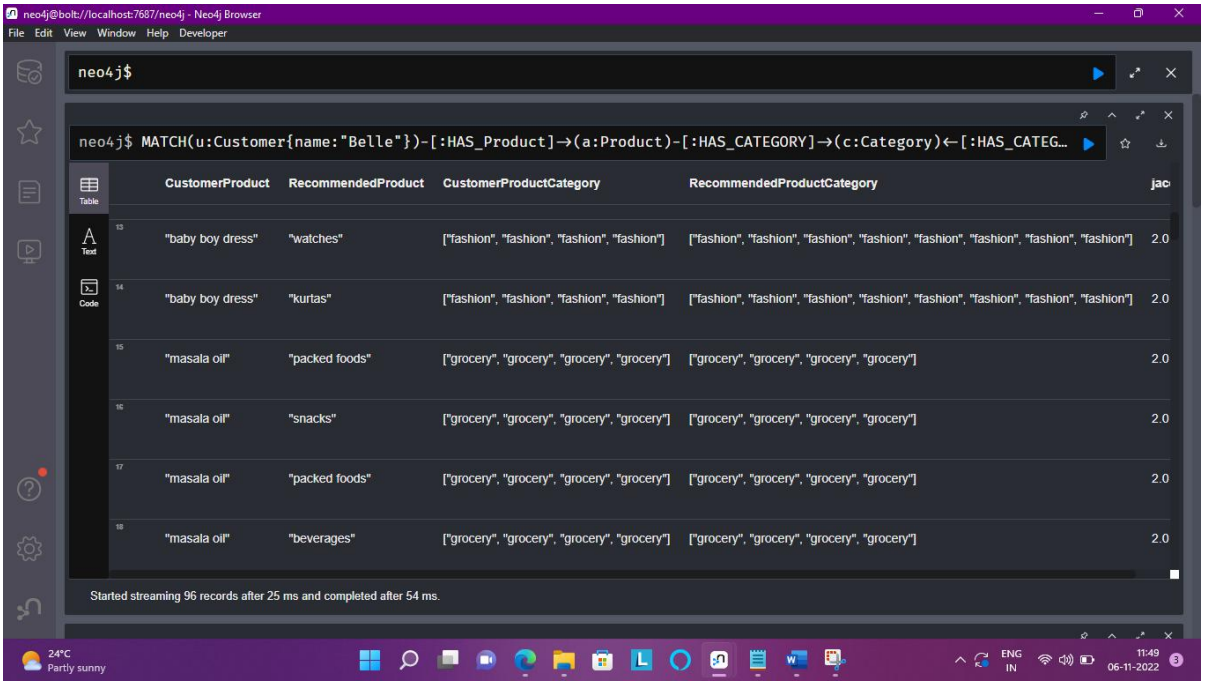
WITH a, z, intersection, a1+[x IN a2 WHERE NOT x IN a1] AS union, a1, a2

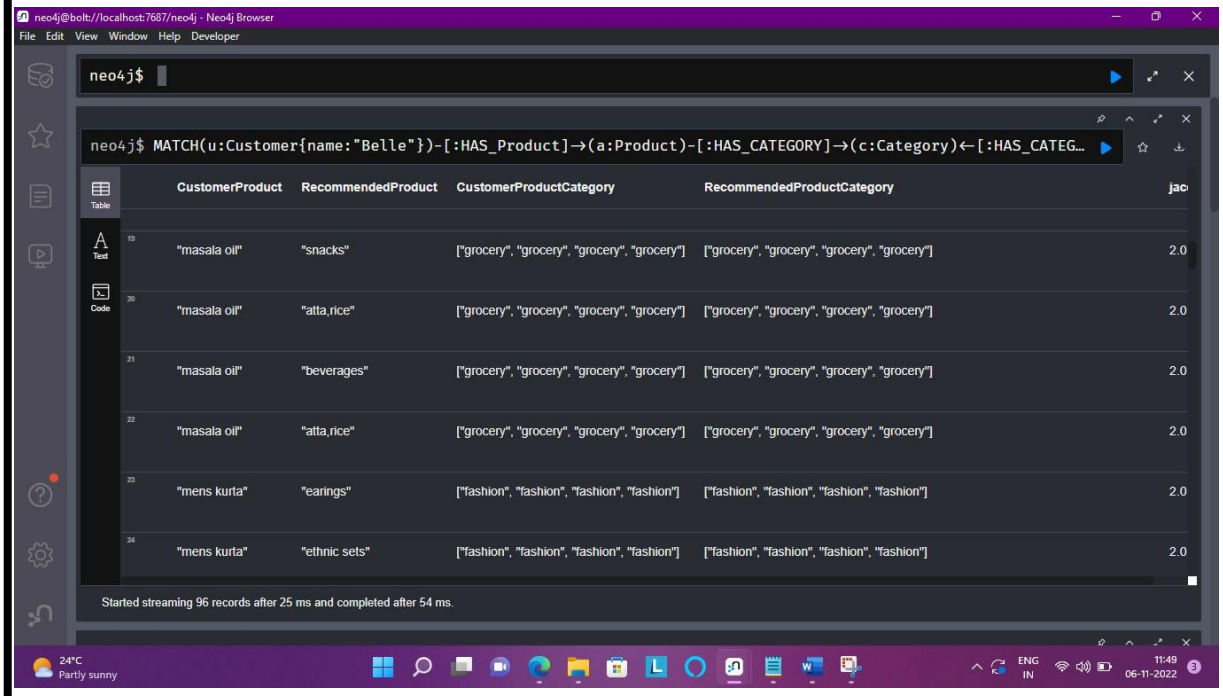
RETURN a.name as CustomerProduct, z.name as RecommendedProduct, a1 as CustomerProductCategory, a2 as RecommendedProductCategory, ((1.0*intersection)/SIZE(union)) AS jaccard ORDER BY jaccard DESC

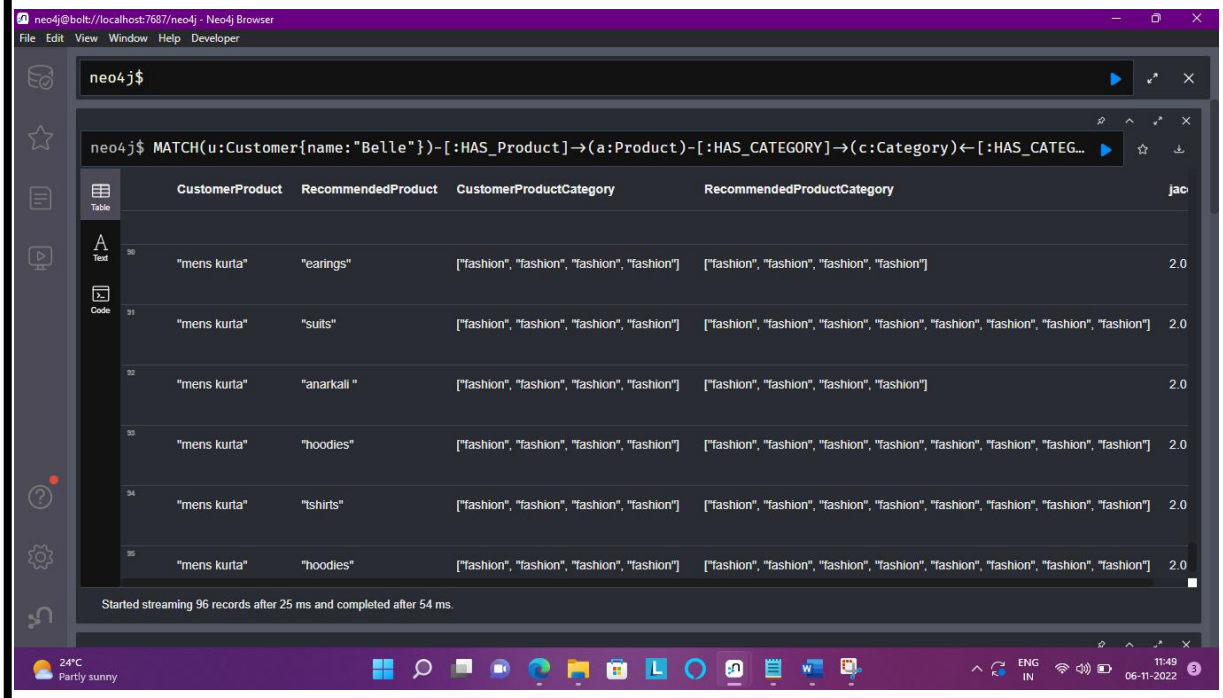
Output in table form:



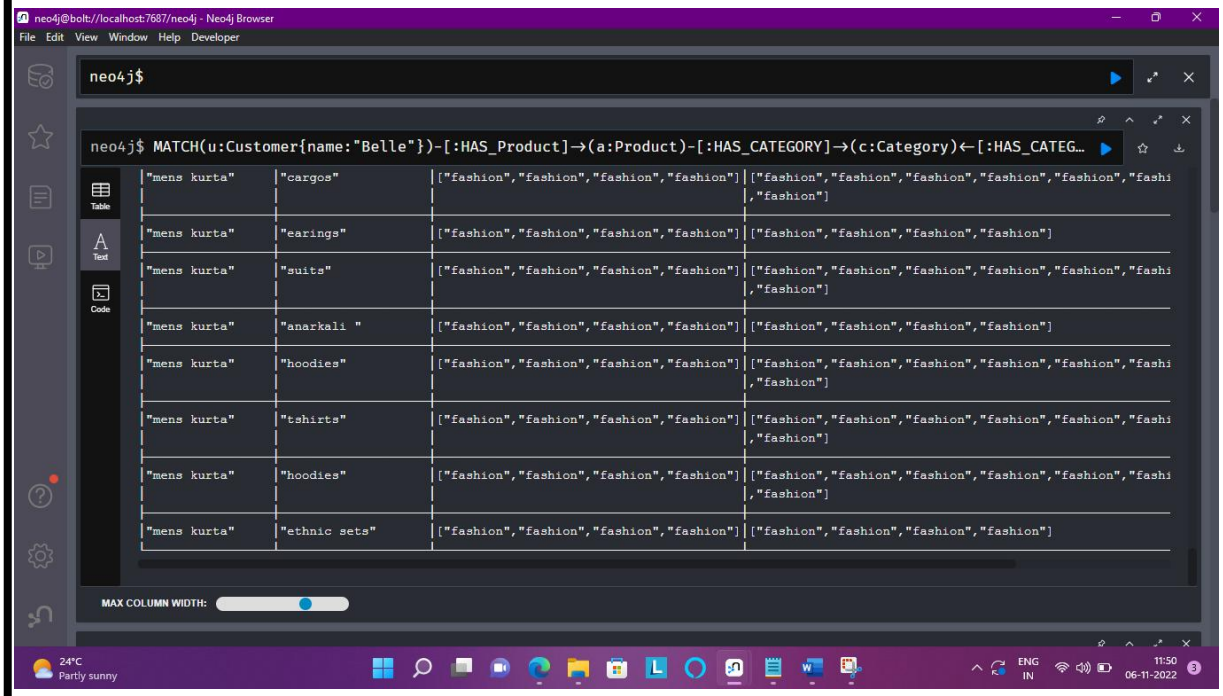

neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
neo4j$ MATCH(u:Customer{name:"Belle"})-[:HAS_Product]→(a:Product)-[:HAS_CATEGORY]→(c:Category)←[:HAS_CATEG...
Table
Text
Code
CustomerProduct
RecommendedProduct
CustomerProductCategory
RecommendedProductCategory
jac
7 "baby boy dress" "Jackets" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion"] 2.0
8 "baby boy dress" "mens casuals" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
9 "baby boy dress" "adidas" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
10 "baby boy dress" "Jackets" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion"] 2.0
11 "baby boy dress" "kurtas" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
12 "baby boy dress" "mens casuals" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
Started streaming 96 records after 25 ms and completed after 54 ms.
24°C
Partly sunny
ENG
IN
11:48
06-11-2022


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
neo4j$ MATCH(u:Customer{name:"Belle"})-[:HAS_Product]→(a:Product)-[:HAS_CATEGORY]→(c:Category)←[:HAS_CATEG...
Table
Text
Code
CustomerProduct
RecommendedProduct
CustomerProductCategory
RecommendedProductCategory
jac
13 "baby boy dress" "watches" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
14 "baby boy dress" "kurtas" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
15 "masala oil" "packed foods" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
16 "masala oil" "snacks" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
17 "masala oil" "packed foods" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
18 "masala oil" "beverages" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
Started streaming 96 records after 25 ms and completed after 54 ms.
24°C
Partly sunny
ENG
IN
11:49
06-11-2022


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
neo4j$ MATCH(u:Customer{name:"Belle"})-[:HAS_Product]→(a:Product)-[:HAS_CATEGORY]→(c:Category)←[:HAS_CATEG...
Table
Text
Code
CustomerProduct RecommendedProduct CustomerProductCategory RecommendedProductCategory jac
19 "masala oil" "snacks" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
20 "masala oil" "atta,rice" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
21 "masala oil" "beverages" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
22 "masala oil" "atta,rice" ["grocery", "grocery", "grocery", "grocery"] ["grocery", "grocery", "grocery", "grocery"] 2.0
23 "mens kurta" "earings" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion"] 2.0
24 "mens kurta" "ethnic sets" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion"] 2.0
Started streaming 96 records after 25 ms and completed after 54 ms.
24°C
Partly sunny
ENG
IN
11:49
06-11-2022


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
neo4j$ MATCH(u:Customer{name:"Belle"})-[:HAS_Product]→(a:Product)-[:HAS_CATEGORY]→(c:Category)←[:HAS_CATEG...
Table
Text
Code
CustomerProduct RecommendedProduct CustomerProductCategory RecommendedProductCategory jac
90 "mens kurta" "earings" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion"] 2.0
91 "mens kurta" "suits" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
92 "mens kurta" "anarkali " ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion"] 2.0
93 "mens kurta" "hoodies" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
94 "mens kurta" "tshirts" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
95 "mens kurta" "hoodies" ["fashion", "fashion", "fashion", "fashion"] ["fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion", "fashion"] 2.0
Started streaming 96 records after 25 ms and completed after 54 ms.
24°C
Partly sunny
ENG
IN
11:49
06-11-2022

Output represented in the form of text



Exported excel sheetoutput for content-based filtering:

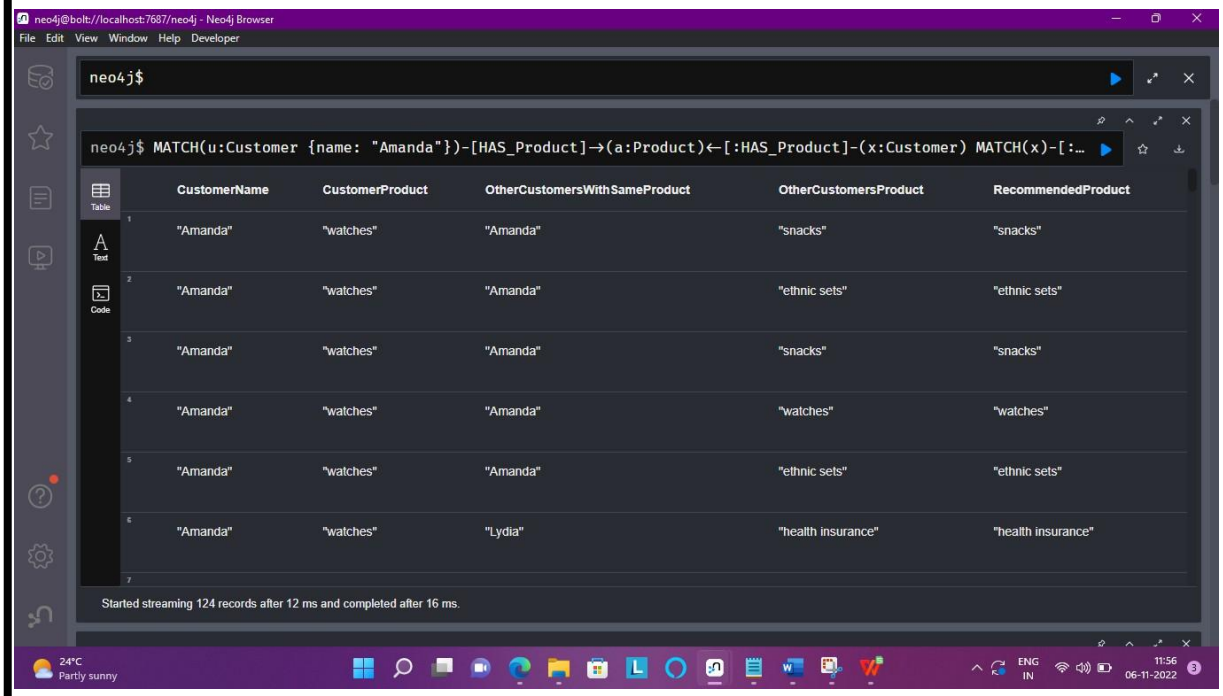
	A	B	C	D	E	F
1	CustomerP	Recommen	CustomerP	Recommen	jaccard	
2	baby boy di	bata shoe	[fashion,fa	[fashion,fa	2	
3	baby boy di	sarees	[fashion,fa	[fashion,fa	2	
4	baby boy di	sarees	[fashion,fa	[fashion,fa	2	
5	baby boy di	bata shoe	[fashion,fa	[fashion,fa	2	
6	baby boy di	watches	[fashion,fa	[fashion,fa	2	
7	baby boy di	adidas	[fashion,fa	[fashion,fa	2	
8	baby boy di	Jackets	[fashion,fa	[fashion,fa	2	
9	baby boy di	mens casua	[fashion,fa	[fashion,fa	2	
10	baby boy di	adidas	[fashion,fa	[fashion,fa	2	
11	baby boy di	Jackets	[fashion,fa	[fashion,fa	2	
12	baby boy di	kurtas	[fashion,fa	[fashion,fa	2	
13	baby boy di	mens casua	[fashion,fa	[fashion,fa	2	
14	baby boy di	watches	[fashion,fa	[fashion,fa	2	
15	baby boy di	kurtas	[fashion,fa	[fashion,fa	2	
16	masala oil	packed foo	[grocery,gr	[grocery,gr	2	
17	masala oil	snacks	[grocery,gr	[grocery,gr	2	
18	masala oil	packed foo	[grocery,gr	[grocery,gr	2	
19	masala oil	beverages	[grocery,gr	[grocery,gr	2	
20	masala oil	snacks	[grocery,gr	[grocery,gr	2	
21	masala oil	atta,rice	[grocery,gr	[grocery,gr	2	
22	masala oil	beverages	[grocery,gr	[grocery,gr	2	
23	masala oil	atta,rice	[grocery,gr	[grocery,gr	2	
24	mens kurta	earings	[fashion,fa	[fashion,fa	2	

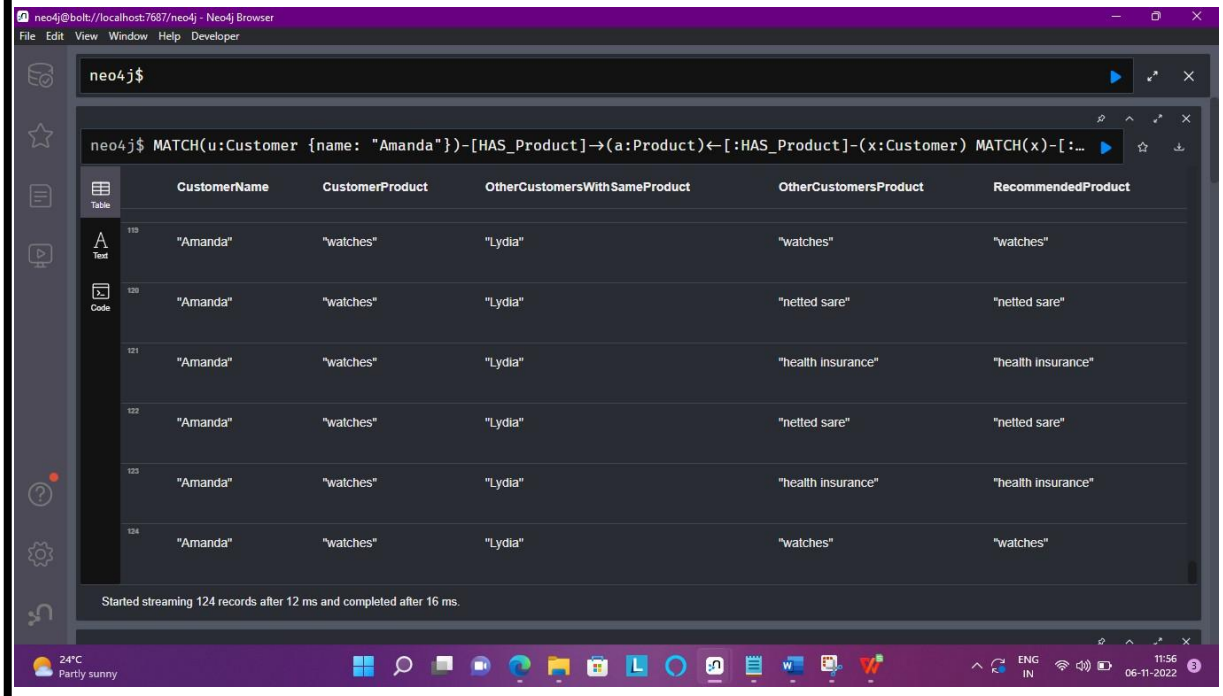
Collaborative Filtering

Cypher Query:

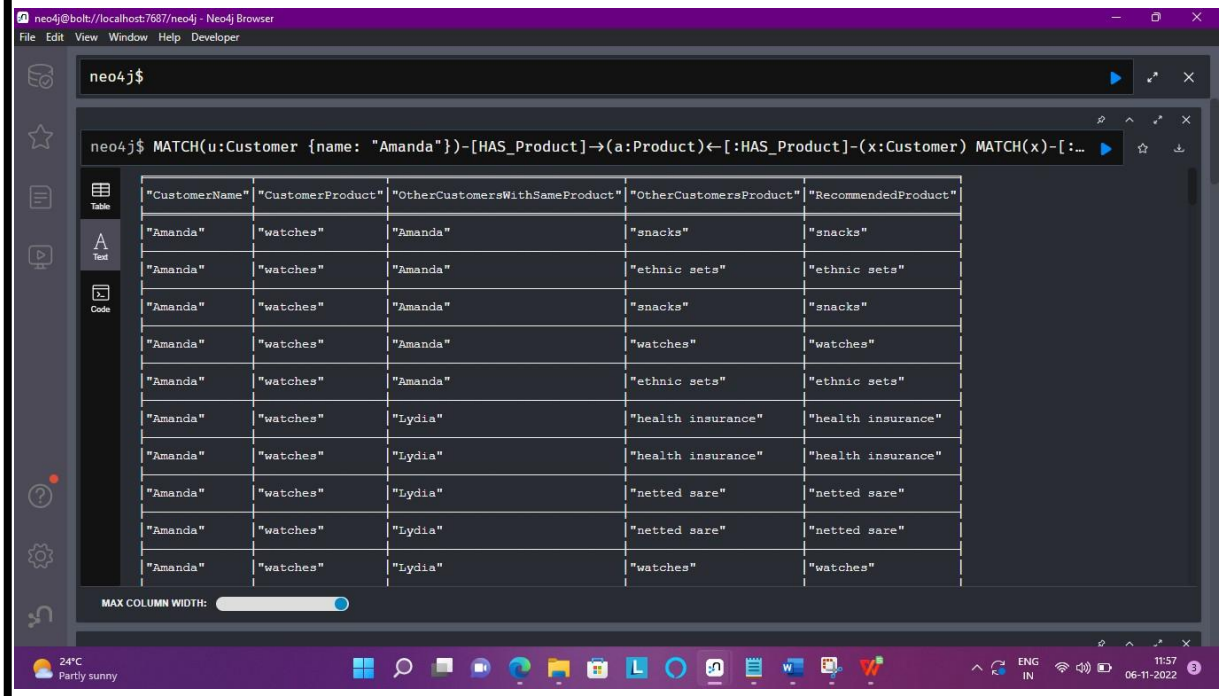
```
MATCH(u:Customer {name: "Amanda"})-[HAS_Product]->(a:Product)<-[:HAS_Product]-(x:Customer)
MATCH(x)-[:]->(z:Product) WHERE NOT exists((u)-[HAS_Product]->(z))
RETURN u.name as CustomerName, a.name as CustomerProduct, x.name as OtherCustomersWithSameProduct, z.name
as OtherCustomersProduct, z.name as RecommendedProduct
```

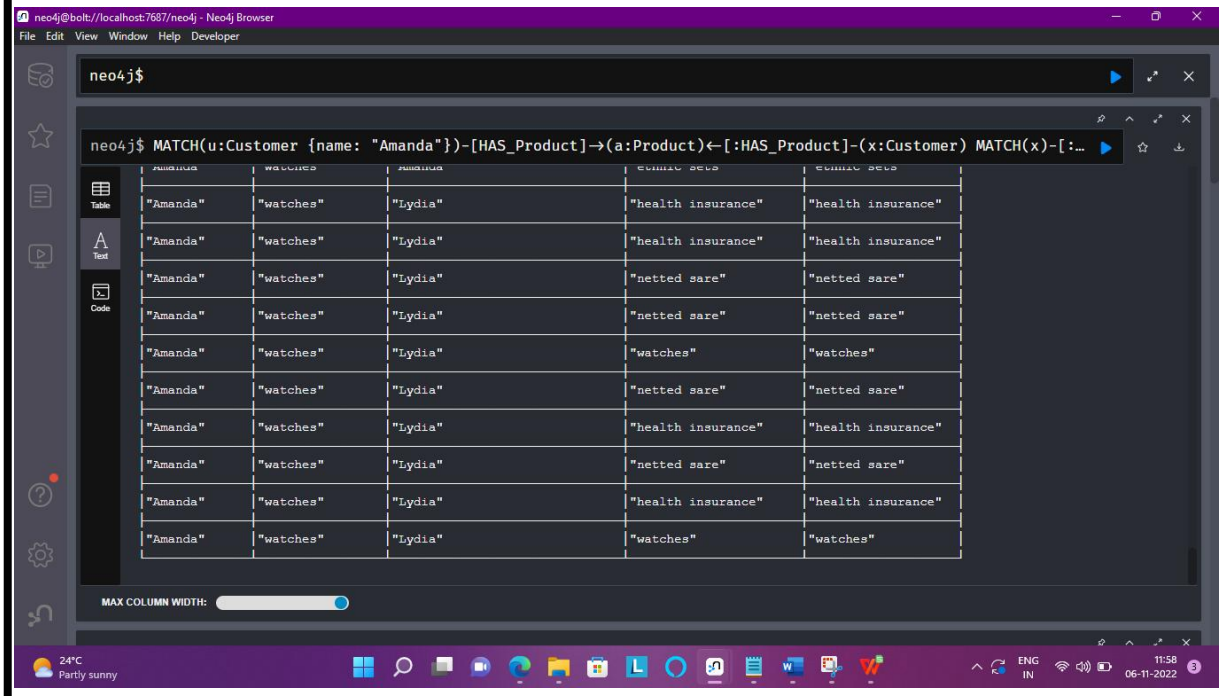
Output in table format:





Output in text format:





Centrality Measure

In graph analytics, Centrality is a very important concept in identifying important nodes in a graph. Itis used to measure the importance (or "centrality" as in how "central" a node is in the graph) of various nodesin a graph. Now, each node could be important from an angle depending on how "importance" is defined. Centrality comes in different flavors and each flavor or a metric defines importance of a node from a differentperspective and further provides relevant analytical information about the graph and its nodes.

Degree Centrality

In a **non-directed graph**, degree of a node is defined as the number of direct connections a node haswith other nodes

In a **directed graph** (each edge has a direction), degree of a node is further divided into In-degree andOut-degree. In-degree refers to the number of edges/connections incident on it and Out-degree refers to the number of edges/connections from it to other nodes.Lets look at a sample Twitter graph below where nodes are individuals and edges with arrows indicate the "Follows" relationship:

Closeness Centrality

Closeness centrality metric defines the importance of a node in a graph as being measured by how close it is to all other nodes in the graph.For a node, it is defined as the sum of the geodesic distance betweenthat node to all other nodes in the network.

Betweenness Centrality

Betweenness Centrality metric defines and measures the importance of a node in a network basedupon how many times it occurs in the shortest path between all pairs of nodes in a graph.

Eigen Vector Centrality

Eigen Vector Centrality metric measures the importance of a node in a graph as a function of the importance of its neighbors. If a node is connected to highly important nodes, it will have a higher EigenVector Centrality score as compared to a node which is connected to lesser important nodes.

According to our graph the nodes with highest centrality measures are "free" and "education".

Analysis/ Inference

By the constructed graph we can analyse the apps that a user like and then recommend it to them by considering the attributes

Analysis of the working of Content-based Filtering

Based on the Cypher query executed previously

- In Content-based filtering the apps were recommended to a user based on the apps which the user hasinstalled.
- This done by checking the category of the app installed and then matching that category with appswith similar category and then recommending it to the user.

Analysis of the working of Collaborative Filtering

Based on the Cypher query executed previously

- In Collaborative filtering the apps were recommended to a user based on which other users have thesame app as our user.
- This done by checking the app name of the app installed by our user and then matching that app namewith other users app.
- Now if a user has same app, then the other apps installed by that user is recommended to our user

Conclusion:

We can conclude that by using Neo4j we can do both Content-based Filtering and Collaborative Filtering easily. Then by using the above created graphs we can implement this in real-time to provide recommendation for users. We can also find the popularity of an app based on this. These methodologies canalso be used in other recommendation system.