(Vivekanand)

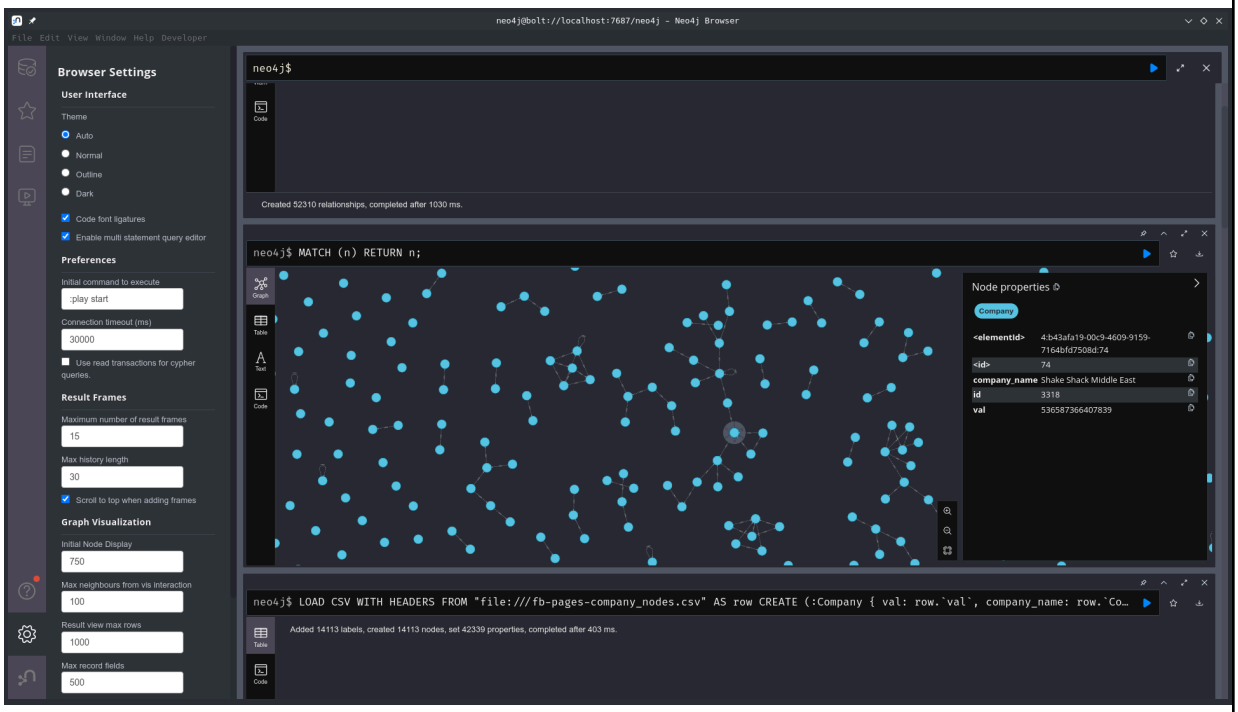
```
CREATE CONSTRAINT FOR (c:Company) REQUIRE c.id IS UNIQUE;
```

```
LOAD CSV WITH HEADERS FROM "file:///fb-pages-company_nodes.csv" AS row
CREATE (:Company {
        val: row.`val`,
        company_name: row.`Company Name`,
        id: row.id
})
```

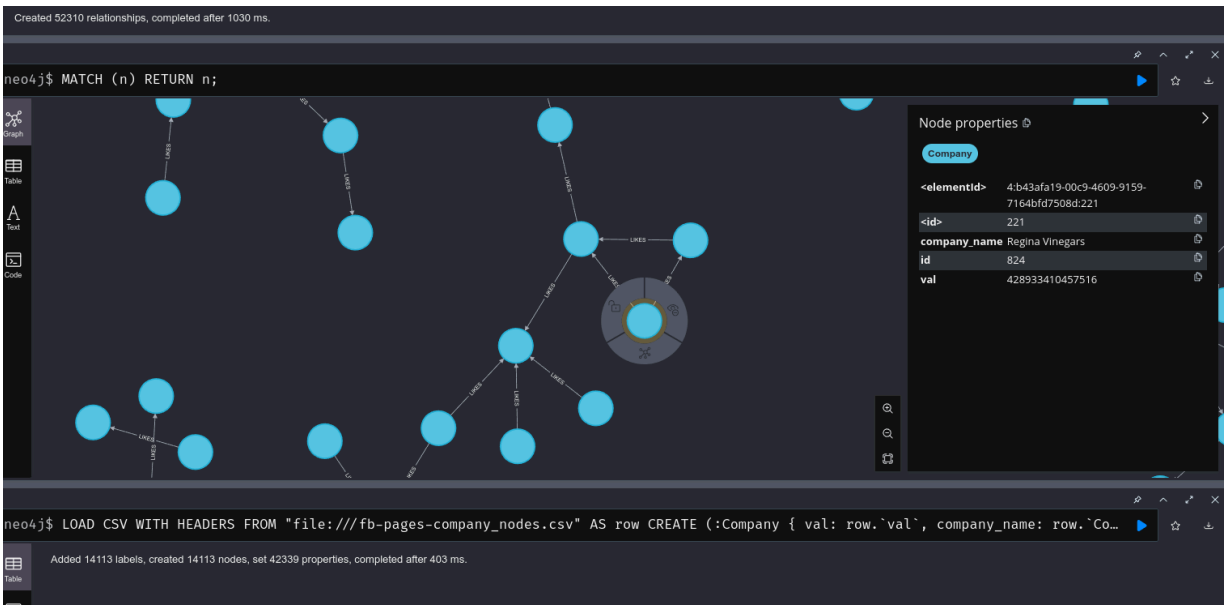
Note: The key is not to include any spaces in the header of the csv file as follows:
val,Company Name,id

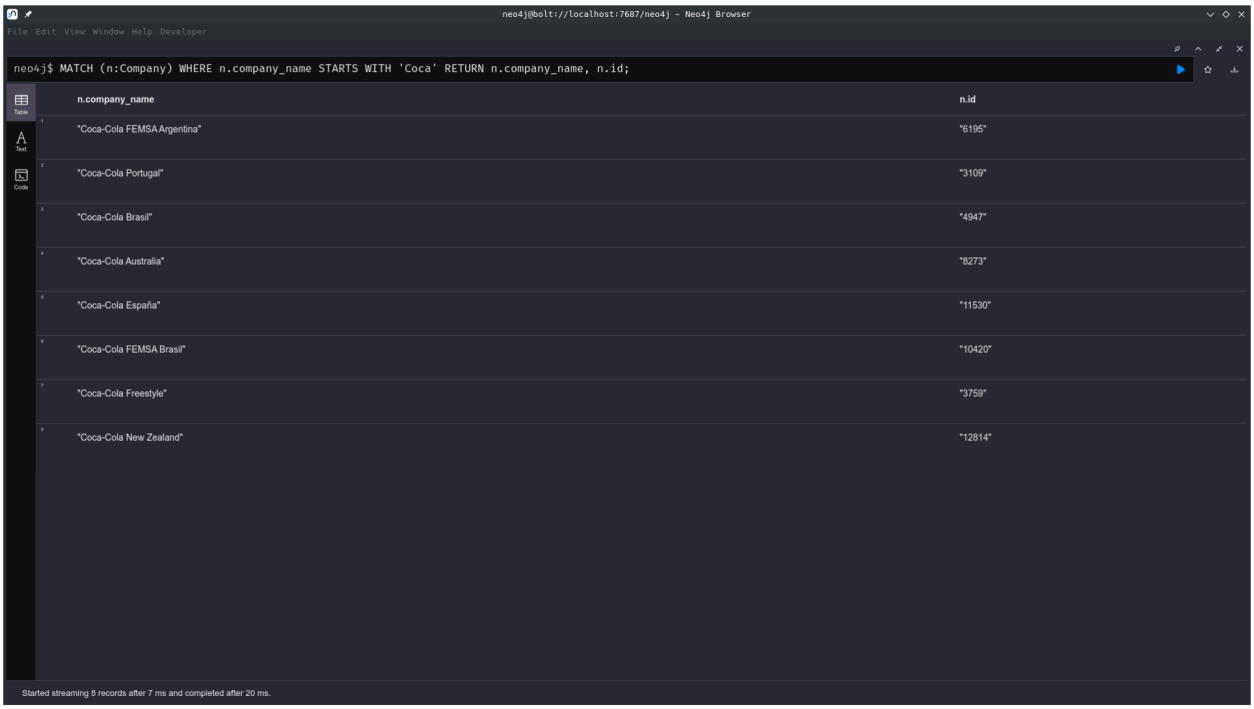
```
LOAD CSV WITH HEADERS FROM "file:///fb-pages-company.edges.csv" AS row
MATCH (c1:Company {id:row.id_from}), (c2:Company {id:row.id_to})
CREATE (c1)-[:LIKES]->(c2);
```

```
Created 52310 relationships, completed after 1030 ms.
```



Interesting Visualizations !






```
MATCH (n:Company) WHERE n.company_name STARTS WITH 'Coca' RETURN n.company_name, n.id;
```

Finding Connected Pages/Companies:

```
MATCH (sony:Company {company_name: "Sony"})-[:LIKES]->(otherCompanies) RETURN sony, otherCompanies;
```

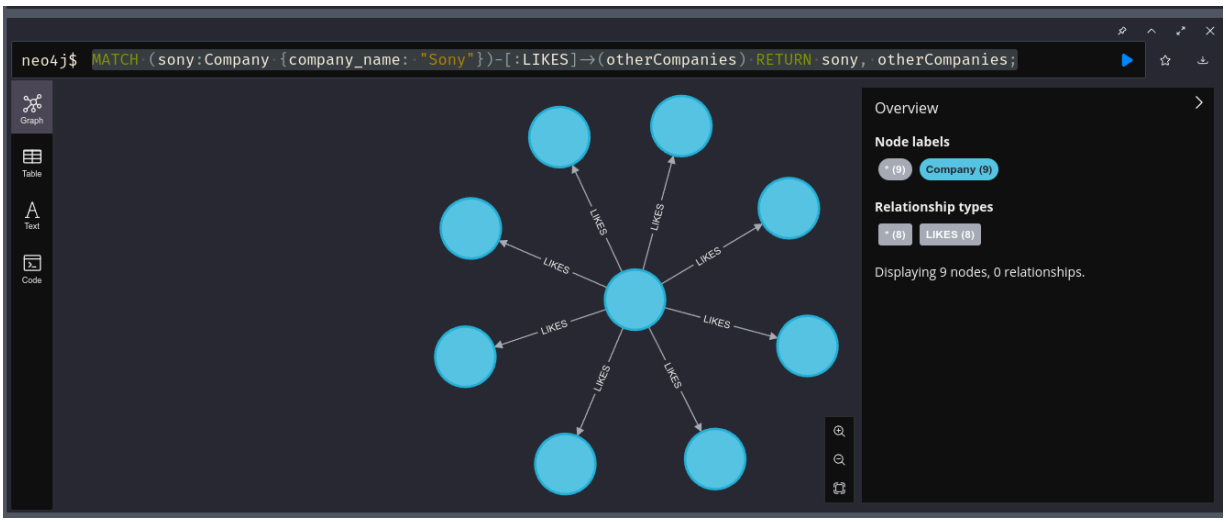
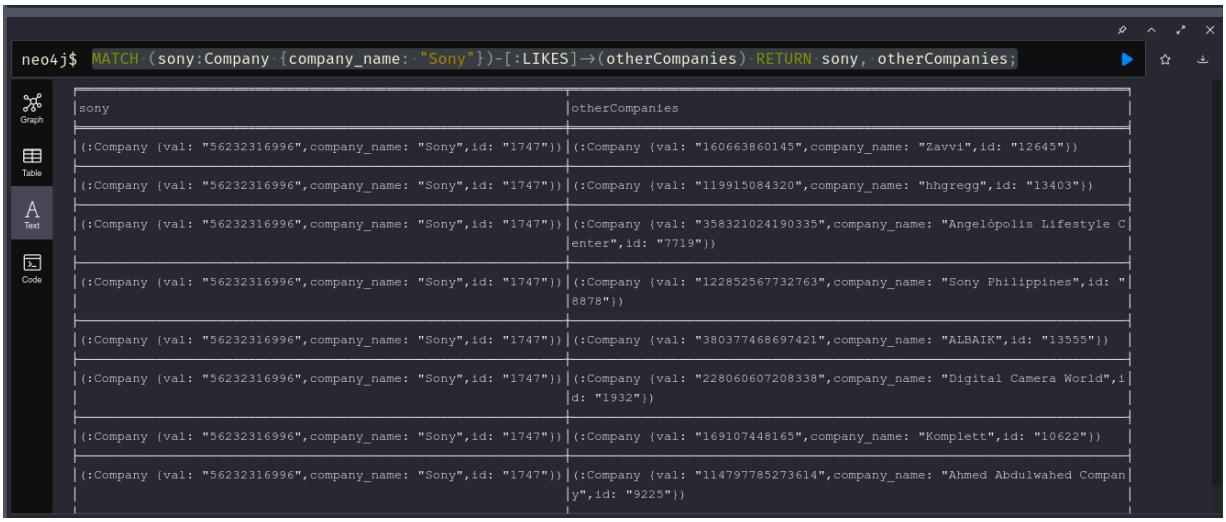


Table Format:



Walmart command:

Finding Connected Pages/Companies:

```
MATCH (walmart:Company {company_name: "Walmart"})-[:LIKES]->(otherCompanies) RETURN walmart, otherCompanies;
```

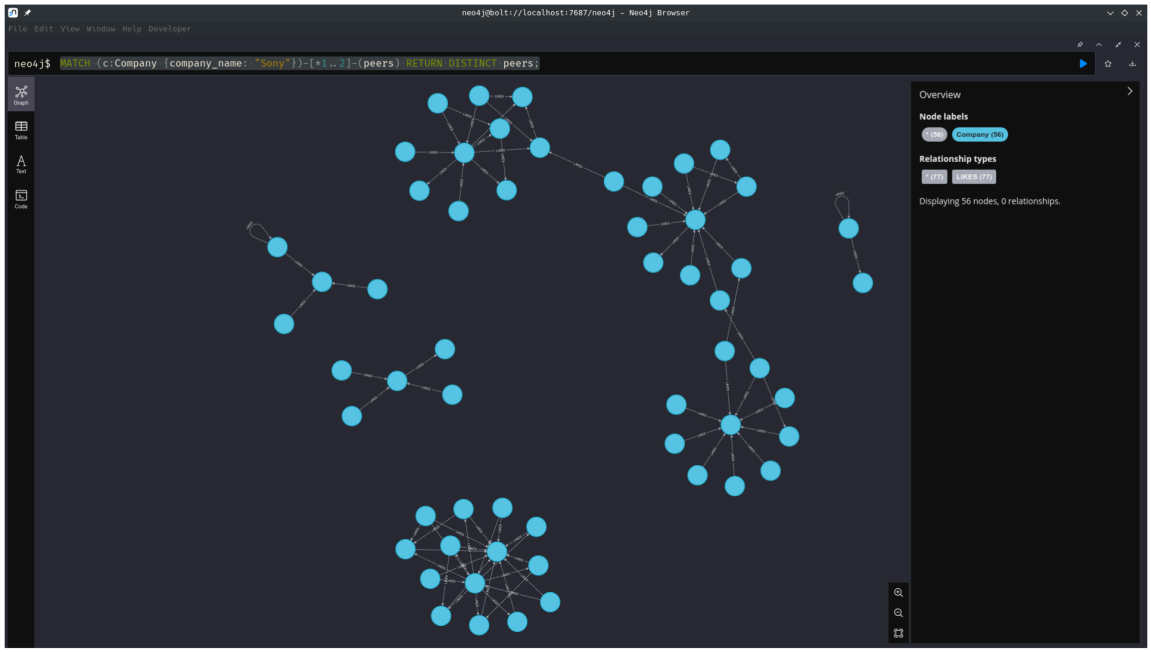
Table Format(Abbreviated):

```
walmart
otherCompanies
```

classic "Six Degrees of Kevin Bacon". That is simply the shortest path between two nodes, called the "Bacon Path".

```
MATCH (c:Company {company_name: "Sony"})-[*1..2]-(peers) RETURN DISTINCT peers;
```

Graph:



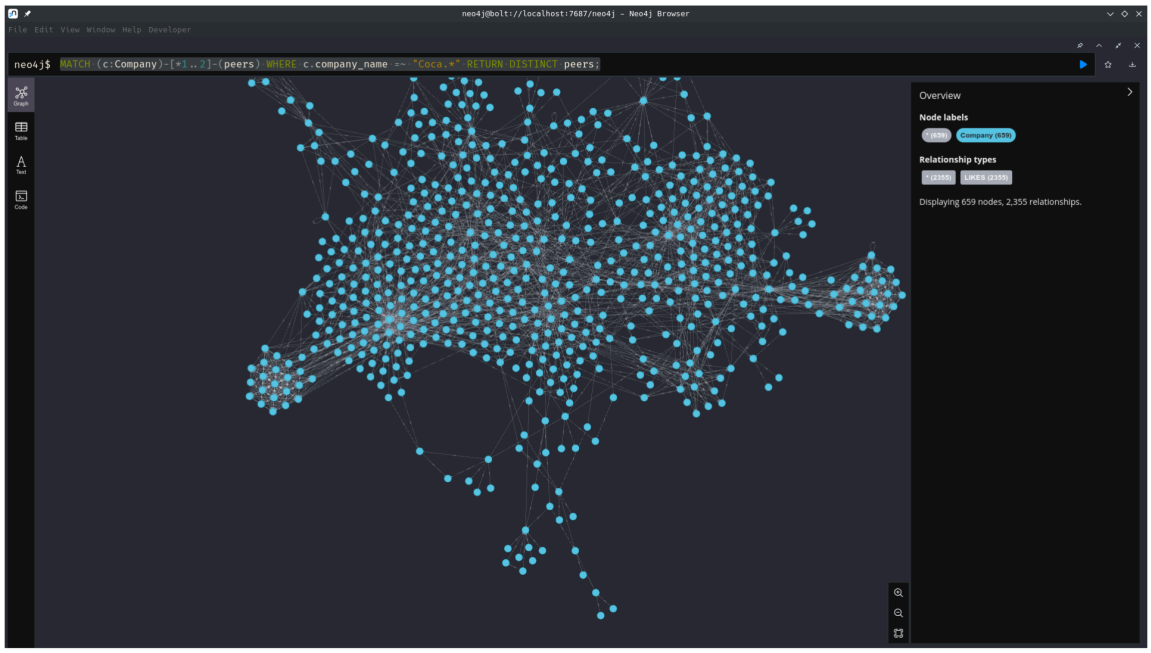
Short Excerpt of Table:

peers
(:Company {val: "160663860145",company_name: "Zavvi",id: "12645"})
(:Company {val: "119915084320",company_name: "hhgregg",id: "13403"})
(:Company {val: "358321024190335",company_name: "Angelópolis Lifestyle Center",id: "7719"})
(:Company {val: "122852567732763",company_name: "Sony Philippines",id: "8878"})
(:Company {val: "380377468697421",company_name: "ALBAIK",id: "13555"})
(:Company {val: "228060607208338",company_name: "Digital Camera World",id: "1932"})
(:Company {val: "169107448165",company_name: "Komplett",id: "10622"})

```
(:Company {val: "114797785273614",company_name: "Ahmed Abdulwahed Company",id: "9225"})
(:Company {val: "86850570385",company_name: "WD",id: "920"})
(:Company {val: "393364824033060",company_name: "Dell",id: "759"})
(:Company {val: "12442500122",company_name: "Verizon",id: "13642"})
(:Company {val: "9291921501",company_name: "Kenneth Cole",id: "13193"})
(:Company {val: "10084673031",company_name: "Cisco",id: "11131"})
(:Company {val: "43877262927",company_name: "APC by Schneider Electric",id: "11157"})
(:Company {val: "120487647987301",company_name: "97.7 QLZ",id: "7616"})
(:Company {val: "20528438720",company_name: "Microsoft",id: "3221"})
(:Company {val: "147546801935011",company_name: "Crock-Pot Slow Cooker",id: "2001"})
(:Company {val: "107593572595009",company_name: "Samsung Gulf",id: "5692"})
```

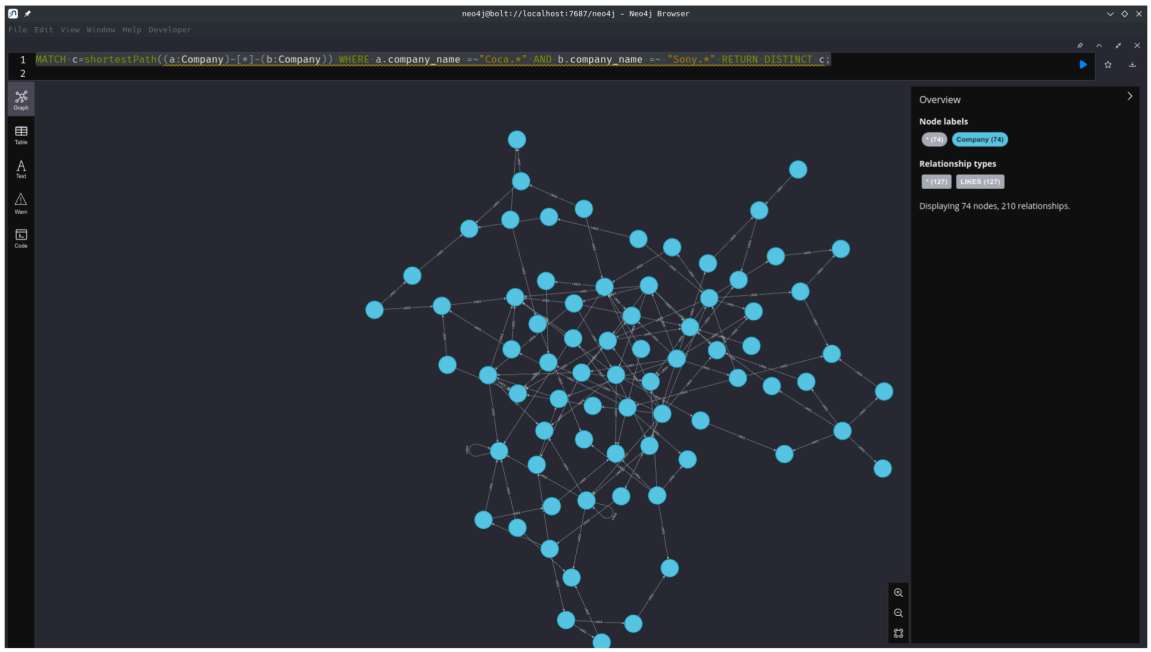
Matching All companies up to 2 hops away from starting with Coco or all the child/sister companies of Coca Cola.

```
MATCH (c:Company)-[*1..2]-(peers) WHERE c.company_name =~ "Coca.*" RETURN DISTINCT peers;
```



Shortest Path Algorithm or the the “Bacon Path” from all the sister companies of Coca Cola to Sony

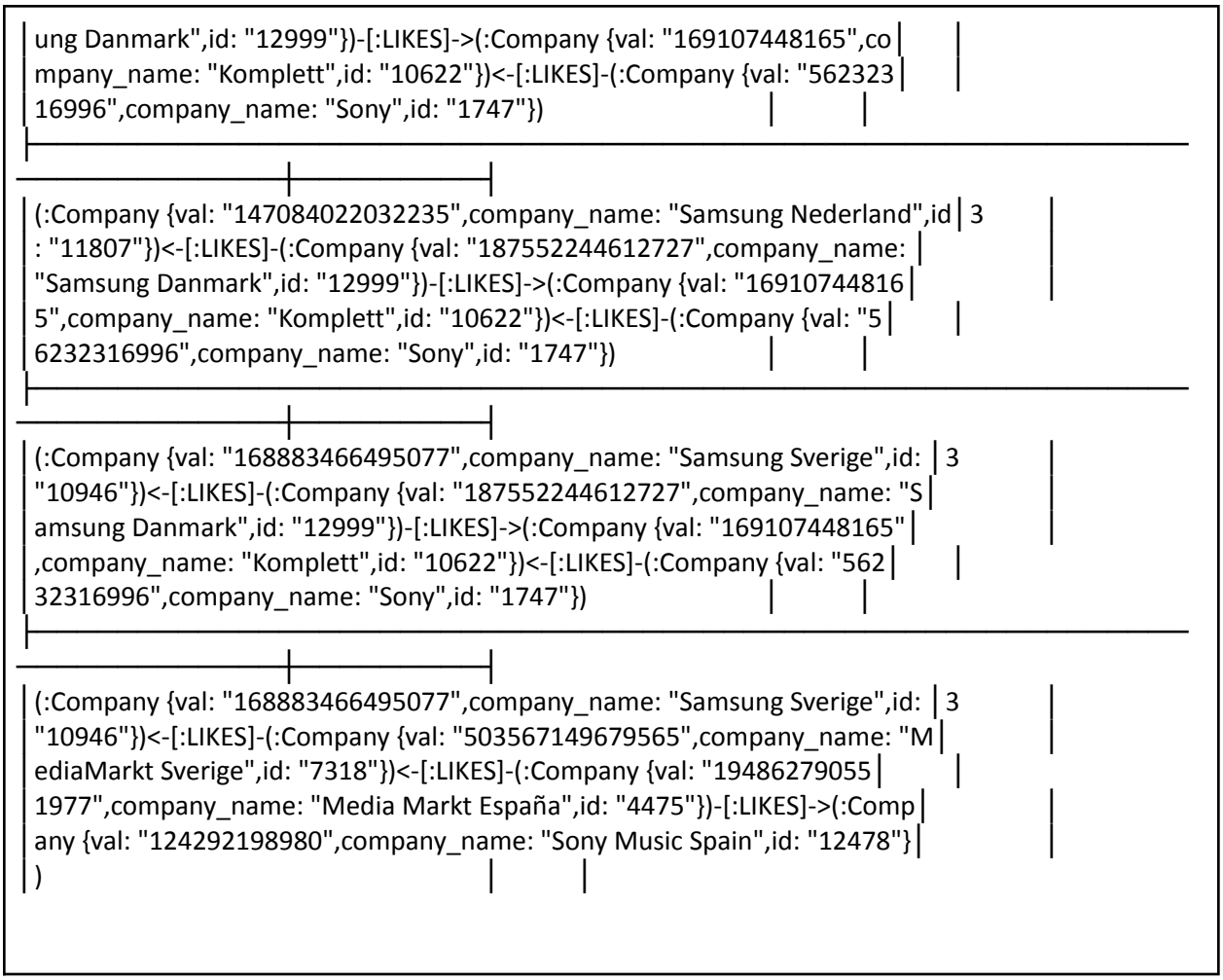
```
MATCH c=shortestPath((a:Company)-[*]-(b:Company)) WHERE a.company_name =~"Coca.*" AND b.company_name =~ "Sony.*" RETURN DISTINCT c;
```



The same shortest path sorted by length:

```
MATCH c=shortestPath((a:Company)-[*]-(b:Company)) WHERE a.company_name =~"Samsung.*" AND b.company_name =~ "Sony.*" RETURN DISTINCT c, length(c) ORDER BY length(c) ;
```

c	length(c)
(:Company {val: "107593572595009",company_name: "Samsung Gulf",id: "5692"})-[:LIKES]->(:Company {val: "114797785273614",company_name: "Ahmed Abdulwahed Company",id: "9225"})<-[:LIKES]-(:Company {val: "56232316996",company_name: "Sony",id: "1747"})	2
(:Company {val: "187552244612727",company_name: "Samsung Danmark",id: "12999"})-[:LIKES]->(:Company {val: "169107448165",company_name: "Komplett",id: "10622"})<-[:LIKES]-(:Company {val: "56232316996",company_name: "Sony",id: "1747"})	2
(:Company {val: "201450226534069",company_name: "Samsung Norge",id: "7645"})-[:LIKES]->(:Company {val: "187552244612727",company_name: "Sams	3




```
CYPHER runtime=parallel MATCH (c:Company)-[*1..2]-(peers) WHERE c.company_name =~ "Apple.*"
RETURN DISTINCT peers;
```

Return all of Apple’s peers up to two connections away.

(Elias)

Louvain

//Setting parameters the algorithm

```
:param limit => ( 1000);
:param config => ({
  relationshipWeightProperty: null,
  includeIntermediateCommunities: false,
  seedProperty: ''
});
:param communityNodeLimit => ( 10);
:param graphConfig => ({
  nodeProjection: 'Company',
  relationshipProjection: {
    relType: {
      type: 'LIKES',
      orientation: 'NATURAL',
      properties: {}
    }
  }
});
:param generatedName => ('in-memory-graph-1733199753404');


// run algorithm and create graph
CALL gds.graph.project($generatedName, $graphConfig.nodeProjection,
$graphConfig.relationshipProjection, {})

CALL gds.louvain.stream($generatedName, $config)
YIELD nodeId, communityId AS community, intermediateCommunityIds AS communities
WITH gds.util.asNode(nodeId) AS node, community, communities
WITH community, communities, collect(node) AS nodes
RETURN community, communities, nodes[0..$communityNodeLimit] AS nodes, size(nodes) AS size
ORDER BY size DESC
LIMIT toInteger($limit)
```

Page Rank:

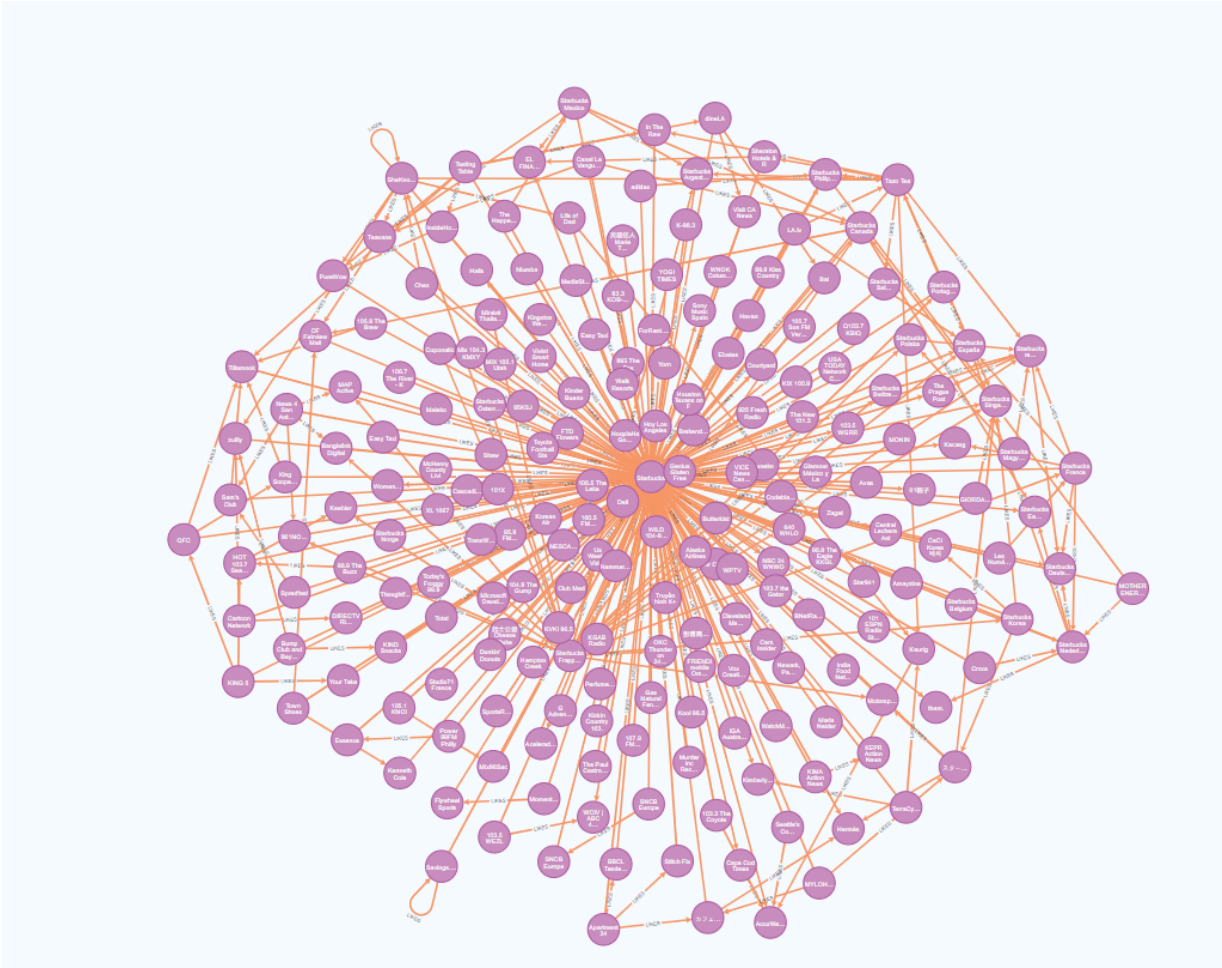
```
//create new in memory graph
CALL gds.graph.project(
  'fb-graph',
  'Company',
  {
    LIKES: {   // Relationship type
      orientation: 'NATURAL',
      properties: {}
    }
  }
);



// run pageRank on graph
CALL gds.pageRank.stream('fb-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).company_name AS Company, score
ORDER BY score DESC
LIMIT 10;
```

	Company	score
1	"Starbucks"	16.502117702809496
2	"Kioski"	11.69393679450167
3	"Iberia"	11.640303179025295
4	"AccuWeather"	11.042386838917222
5	"ESPN Fans"	10.040812511760707
6	"Top Gear"	9.83690576250258
7	"Mashable"	9.293250622585813
8	"CBS News"	9.271085875558652
9	"Airbus"	8.667069865051065
10	"Skittles"	8.317016671274345

```
MATCH (starbucks:Company {company_name: "Starbucks"})-[:LIKES]->(otherCompanies) RETURN starbucks, otherCompanies;
```



Count Direct Connections (Degree)

The degree of a node is the total number of direct neighbors (both incoming and outgoing connections). For "Starbucks":

```
neo4j$ MATCH (n:Company {company_name: 'Starbucks'})--(neighbor) RETURN n.company_name AS Node, count(neighbor) AS TotalConnections;
```

	Node	TotalConnections
1	"Starbucks"	212

Compare it with the top 20 highly connected nodes

```
MATCH (n:Company)--(neighbor)
RETURN n.company_name AS Company, count(neighbor) AS TotalConnections
ORDER BY TotalConnections DESC
LIMIT 20;
```

	Company	TotalConnections
1	"L'OCCITANE en Provence"	1293
2	"Digicel"	587
3	"ABB"	581
4	"Crocs"	500
5	"Microsoft"	423

And it actually is on position 15.

now take a look at the incoming edges:

```
MATCH (n:Company {company_name: 'Starbucks'})<-[:LIKES]-(neighbor)
RETURN n.company_name AS Node, count(neighbor) AS IncomingConnections;
```

	Node	IncomingConnections
1	"Starbucks"	181

Compare it with the top 20 highly nodes incoming edges:

	Company	IncomingConnections
5	"Wiko"	215
6	"ESET"	207
7	"Microsoft"	187
8	"WD"	184
9	"Starbucks"	181
10	"Volkswagen"	173

Started streaming 20 records after 8 ms and completed after 67 ms

It is actually pretty good. It is on position 9

Now let's find out if it has a lot of influence friends:

MATCH (n:Company {company_name: 'Starbucks'})-[:LIKES]-(neighbor)
CALL gds.pageRank.stream('fb-graph') YIELD nodeId, score
WHERE id(neighbor) = nodeId
RETURN neighbor.company_name AS Neighbor, score AS PageRank
ORDER BY PageRank DESC;

```
1 MATCH (n:Company {company_name: 'Starbucks'})-[:LIKES]-(neighbor)
2 CALL gds.pageRank.stream('fb-graph') YIELD nodeId, score
3 WHERE id(neighbor) = nodeId
4 RETURN neighbor.company_name AS Neighbor, score AS PageRank
5 ORDER BY PageRank DESC;
6
```

	Neighbor	PageRank
1	"AccuWeather"	11.042386838917224
2	"Hermès"	5.669334399936937
3	"EL FINANCIERO"	5.291873732707627
4	"100.5 KISS FM"	4.514079938281597
5	"Kacang"	4.110341431745122
6	"Kimberly-Clark Corporation (United States)"	3.298486758743769

And it has a lot of neighbours with a really high PageRank. So it actually has some very influential neighbours.

Now even go further add the degree of the node into the PageRank analyze:
MATCH (n:Company)

```
CALL gds.pageRank.stream('fb-graph') YIELD nodeId, score
WHERE id(n) = nodeId
WITH n, score
MATCH (n)--(neighbor)
RETURN n.company_name AS Node, count(neighbor) AS Degree, score
ORDER BY score DESC
LIMIT 10;
```

	Node	Degree	score
1	"Starbucks"	212	16.5021177028095
2	"Kioski"	76	11.69393679450167
3	"Iberia"	21	11.640303179025295
4	"AccuWeather"	90	11.042386838917222
5	"ESPN Fans"	8	10.04081251176071
6	"Top Gear"	49	9.83690576250258
7			

Table with pageRank incoming edges and outgoing edges

```
neo4j$ MATCH (n:Company) CALL gds.pageRank.stream('fb-graph') YIELD nodeId, score…
```

Table
Text
Warn
Code

	Company	PageRank	IncomingConnections	OutgoingConnections
15	"CHANEL"	7.361649167158827	88	4
16	"Conversations with Richard Fidler"	7.077964890926975	21	0
17	"ABC Books"	6.286154201944108	64	1
18	"Yenda"	6.274521321631802	4	1
19	"Qatar Airways"	6.268563132572963	38	3
20	"Red Bull"	6.2552812390410475	132	53

Conclusion
High PageRank:

- Indicates Starbucks receives a substantial amount of influence from other nodes.
- The influence is amplified if its incoming edges come from other high-PageRank nodes.

High IncomingConnections:

- Suggests Starbucks is "liked" or connected by many nodes in the graph, making it a central entity.

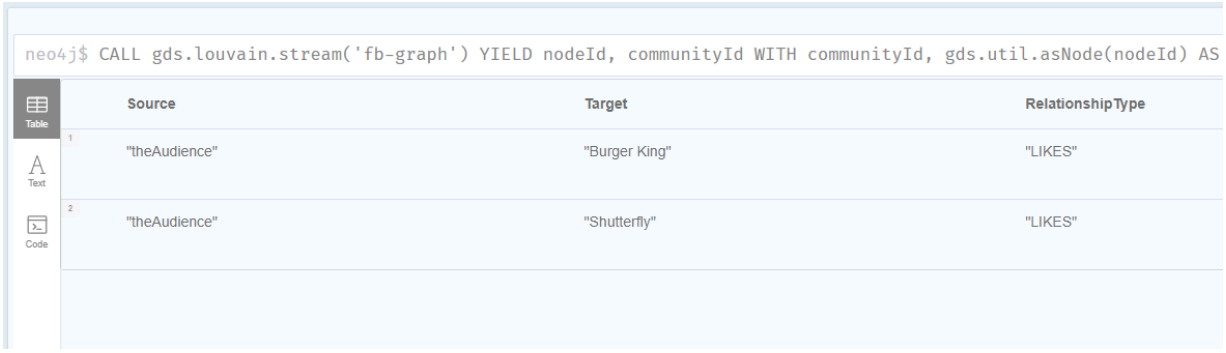
Balanced OutgoingConnections:

- If Starbucks has outgoing edges, their count influences how its PageRank is distributed to other nodes.

To look evain more into it i looked at the lauvain community of starbacks:

```
//get comunity_id
CALL gds.louvain.stream('fb-graph')
YIELD nodeId, communityId
WHERE gds.util.asNode(nodeId).company_name = 'Starbucks'
RETURN communityId;
```

```
CALL gds.louvain.stream('fb-graph')
YIELD nodeId, communityId
WITH communityId, gds.util.asNode(nodeId) AS node
WHERE communityId = 256
MATCH (node)-[r]-(neighbor)
RETURN node.company_name AS Source, neighbor.company_name AS Target, type(r) AS
RelationshipType
ORDER BY Source, Target;
```



Example Case:

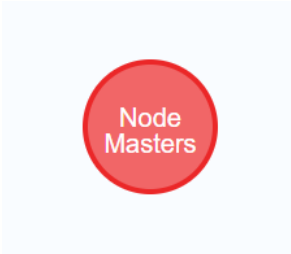
Create a new Company and try to get the most influence with all the network analysis we get to find out who we should like and who we should get to like us.

Goal

- **High Visibility**: To become widely recognized in the network.
- **Central Position**: To integrate deeply into key communities.
- **Strong Reputation**: To be associated with respected, highly influential pages.

1. **Create Your Company**

```
CREATE (:Company {company_name: 'Node Masters', id: '1193499999', val: '22092443056'});
```



2. **Find smaller Companies to start with**

```
MATCH (n:Company) RETURN n.company_name AS Company, size((n)--()) AS TotalConnections
ORDER BY TotalConnections ASC LIMIT 10;
```

```
1 MATCH (n:Company)
2 RETURN n.company_name AS Company, count{(n)--()} AS TotalConnections
3 ORDER BY TotalConnections ASC
4 LIMIT 15;
```

	Company	TotalConnections
1	"Node Masters"	0
2	"MLC Australia"	1
3	"Where We Live"	1
4	"GoAir"	1
5	"Fickle Fish Films"	1
6	"Castello"	1

3. **Like Small Companies and Get Them to Like You**

```
MATCH (n:Company)

WHERE n.company_name IN ['Thrive Global', 'SimonBooks', 'International Delight',
```

```
'Headspace']

MATCH (yourCompany:Company {company_name: 'Node Masters'})

CREATE (yourCompany)-[:LIKES]->(n);
```

4. **Identify the most Influential Nodes (pageRank)**

```
CALL gds.pageRank.stream('fb-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).company_name AS Company, score AS PageRank
ORDER BY PageRank DESC
LIMIT 20;
```

	Company	PageRank
1	"Starbucks"	16.5021177028095
2	"Kioski"	11.693936794501672
3	"Iberia"	11.640303179025295
4	"AccuWeather"	11.042386838917222
5	"ESPN Fans"	10.04081251176071
6	"Top Gear"	9.836905762502582

5. **Like the influential Pages**

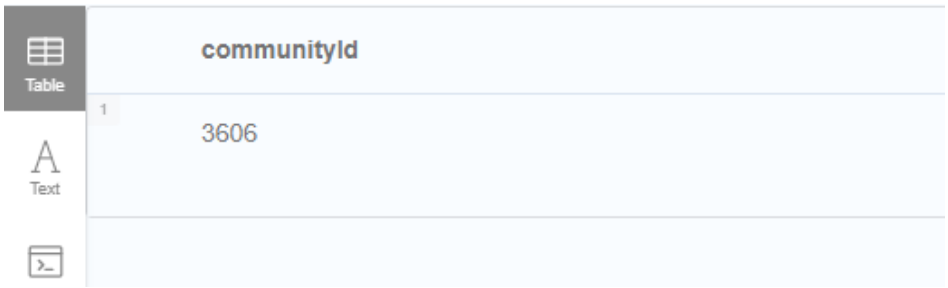
```
MATCH (influential:Company)
WHERE influential.company_name IN ['Starbucks', 'Kioski','Starbucks', 'Iberia','AccuWeather']  //
Replace with actual top page names
MATCH (yourCompany:Company {company_name: 'Node Masters'})
CREATE (influential)-[:LIKES]->(yourCompany);
```

7. Get Liked by Influential Page (just one for this example)

```
MATCH (influential:Company)
WHERE influential.company_name IN ['Starbucks']
MATCH (yourCompany:Company {company_name: 'Node Masters'})
CREATE (influential)-[:LIKES]->(yourCompany);
```

6. **Find your community (Louvain)**

CALL gds.louvain.stream('fb-graph') YIELD nodeId, communityId WHERE gds.util.asNode(nodeId).company_name = 'YourCompany' RETURN communityId;



7. **Display the community**

```
CALL gds.louvain.stream('fb-graph')
YIELD nodeId, communityId
WITH communityId, gds.util.asNode(nodeId) AS node
WHERE communityId = 3606
MATCH (node)-[r]-(neighbor)
RETURN node.company_name AS Source, neighbor.company_name AS Target, type(r) AS
RelationshipType
ORDER BY Source, Target;
```

Source	Target	RelationshipType
"HomeAway"	"Airfarewatchdog"	"LIKES"
"HomeAway"	"Cheapflights"	"LIKES"
"HomeAway"	"Telegraph Travel"	"LIKES"
"HomeAway"	"The North Face"	"LIKES"

8. **Add Likes to Community Members**:

MATCH (member:Company) WHERE member.communityId =3606 MATCH (yourCompany:Company {company_name: 'YourCompany'}) CREATE (yourCompany)-[:LIKES]->(member);

9. **Find Key Bridges (Nodes with High Betweenness)**

```
CALL gds.betweenness.stream('fb-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).company_name AS Company, score
```

```
ORDER BY score DESC
LIMIT 5;
```

	Company	score
1	"CNN"	441293.5877407348
2	"Red Bull"	365830.6710642946
3	"BBC News"	337846.6567719818
4	"TIME"	335331.4597432648
5	"Target"	245795.5794957561

10. **Build connection with them**

```
MATCH (bridge:Company)
WHERE bridge.company_name IN ['CNN', 'Red Bull', 'BBC News', 'TIME', 'Target']
MATCH (yourCompany:Company {company_name: 'Node Masters'})
CREATE (yourCompany)-[:LIKES]->(bridge);
```

11. **Engage with Followers of Influential Pages (find followers of initial Pages)**

```
MATCH (influential:Company)-[:LIKES]->(follower:Company) WHERE influential.company_name =
'TopPage1' RETURN follower.company_name AS Follower LIMIT 10;
```

	Follower
1	"Hermès"
2	"Starbucks Singapore"
3	"WCIV | ABC News 4"
4	"Hoy Los Angeles"
5	"Cape Cod Times"
6	"100.5 KISS FM"

12. **Build Connection with them:**

```
MATCH (follower:Company)
WHERE follower.company_name IN ['Hermès', 'Starbucks Singapore', 'WCIV | ABC News 4', 'Hoy Los
```

```
Angeles', 'Cape Cod Times', '100.5 KISS FM']
MATCH (yourCompany:Company {company_name: 'Node Masters'})
CREATE (yourCompany)-[:LIKES]->(follower);
```

10. Calculate the Final PageRank

```
CALL gds.pageRank.stream('fb-graph-simulation-1')
YIELD nodeId, score
WHERE gds.util.asNode(nodeId).company_name = 'Node Masters'
RETURN score AS PageRank;
```

Result : 27.613599029629306

(Vivekanand)

Advanced Analysis:

Returning subgraphs of company starting with name Sony from a maximum of 2 hops away

```
MATCH (c:Company) WHERE c.company_name starts with "Sony" CALL apoc.path.expand(c, "LIKES", null, 1, 2) YIELD path RETURN path, length(path) AS hops order by hops;
```

MATCH (c:Company) WHERE c.company_name starts with $c CALL apoc.path.expand(c, "LIKES", null, 1, $n) YIELD path RETURN path, length(path) AS hops order by hops;

Display companies a number of hops away from a company

Match the path between two companies **a** and **b**.

Company relations between $from and $to

cypher runtime=parallel MATCH c=shortestPath((a:Company)-[*]-(b:Company)) WHERE a.company_name contains $from AND b.company_name contains $to RETURN DISTINCT c, length(c) ORDER BY length(c) ;

Return the nodes (In progress):

```
MATCH (c:Company) MATCH (end:Company) WHERE c.company_name starts with "Sony" AND
end.company_name in ["Samsung"] WITH
c, collect(end) as endNodes
CALL apoc.path.subgraphNodes(c,{
relationshipFilter:"LIKES",
terminatorNodes: endNodes,
minLevel: 0
})
YIELD node
RETURN node;
```

Speed up queries:

Google speed (millisecond results)

```
CREATE INDEX node_range_company_name_index FOR (c:Company) ON (c.company_name)
```