<https://neo4j.com/blog/getting-started-data-analysis-neo4j/>

1. Importing data from a JSON file (sample file)

CALL apoc.load.json('https://raw.githubusercontent.com/arjuntherajeev/neo4j\_issuu\_data\_analysis/master/issuu\_sample.json')

YIELD value

UNWIND value.items AS item

WITH item

WHERE NOT item.env\_doc\_id IS NULL

MERGE (document:Document

{doc\_uuid:item.env\_doc\_id})

MERGE (visitor:Visitor

{visitor\_uuid:item.visitor\_uuid})

ON CREATE SET visitor.visitor\_country

= item.visitor\_country

MERGE (visitor)-[:VIEWED{type:item.event\_type}]->(document)

1. Importing data from a JSON file (2,295 nodes and 2,171 relationships)

MATCH (n) DETACH DELETE n;

CALL apoc.load.json('https://raw.githubusercontent.com/arjuntherajeev/neo4j\_issuu\_data\_analysis/master/issuu\_cw2.json')

YIELD value

UNWIND value.items AS item

WITH item

WHERE NOT item.env\_doc\_id IS NULL

MERGE (document:Document

{doc\_uuid:item.env\_doc\_id})

MERGE (visitor:Visitor

{visitor\_uuid:item.visitor\_uuid})

ON CREATE SET visitor.visitor\_country

= item.visitor\_country

MERGE (visitor)-[:VIEWED{type:item.event\_type}]->(document)

#### Added 2293 labels, created 2293 nodes, set 5749 properties,

#### Created 2170 relationships, statement executed in 15523 ms.

1. To check whether the graph was populated successfully, we can run the Cypher query:

MATCH (n) RETURN (n) LIMIT 200

Query 1. Find the number of visitors from each country and display them in the descending order of count.

MATCH (v:Visitor)

RETURN v.visitor\_country AS Country, count(v) AS Count

ORDER BY count(v) DESC

LIMIT 10

Query 2. For a given document, find the number of visitors from each country.

MATCH (d:Document)<-[:VIEWED]-(v:Visitor)

WHERE d.doc\_uuid='140228101942-d4c9bd33cc299cc53d584ca1a4bf15d9'

RETURN v.visitor\_country AS Country, count(v.visitor\_country) AS Count

ORDER BY count(v.visitor\_country) DESC

Query 3. Find the number of occurrences for each type of viewership activity.

MATCH (d:Document)<-[r:VIEWED]-(v:Visitor)

RETURN r.type AS Type, count(d.doc\_uuid) AS Count

ORDER BY Count ASC

Query 4. Find the visitors for each document and display the top three in the descending order of number of visitors.

MATCH (d:Document)<-[r:VIEWED]-(v:Visitor)

RETURN DISTINCT d.doc\_uuid AS DocUUID, collect(DISTINCT v.visitor\_uuid) AS Visitors, count(DISTINCT v.visitor\_uuid) AS Count

ORDER BY Count DESC

LIMIT 3

Query 5. For a given document, find recommendations of other documents like it.

Example 1: document UUID = 130902223509-8fed6b88ae0937c1c43fb30cb9f87ad8

MATCH (d:Document)<-[r:VIEWED]-(v:Visitor)-[r1:VIEWED]->(d1:Document)

WHERE d1<>d

AND

d.doc\_uuid='130902223509-8fed6b88ae0937c1c43fb30cb9f87ad8'

RETURN d1 AS Recommendations, count(\*) AS Views,

sum(

CASE r1.type

WHEN "impression" THEN 1

WHEN "pageread" THEN 1.5

WHEN "pagereadtime" THEN 1.5

WHEN "read" THEN 2

WHEN "click" THEN 0.5

ELSE 0

END

) as Score

ORDER BY Score DESC