Eléonor KIOULOU
Adéle MONTOYA
Paul RUNAVOT

TP NEO4J

Download neo4j on Docker:

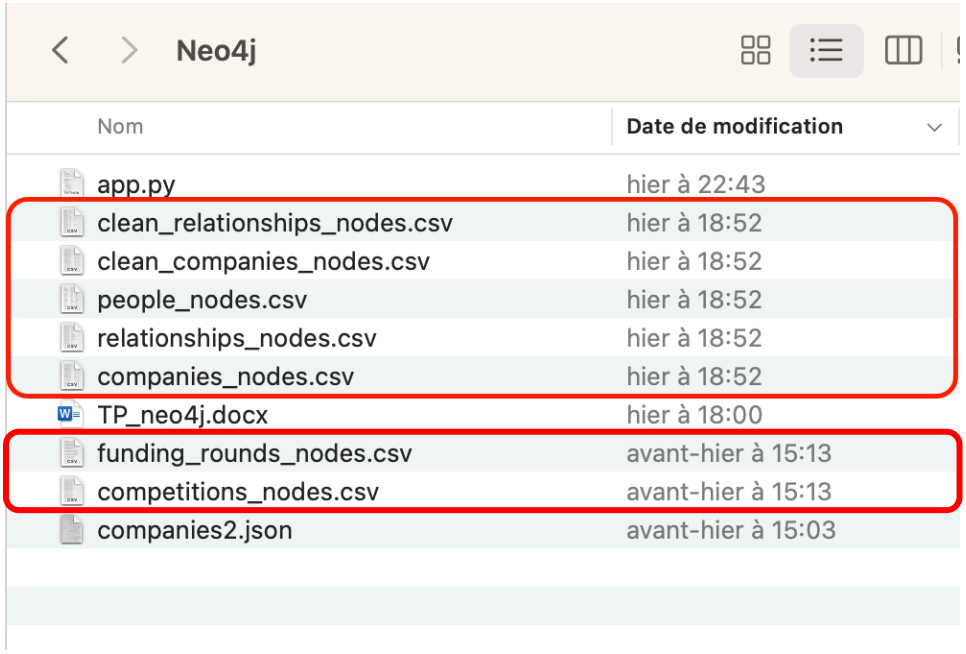
```
❯ docker pull neo4j

Using default tag: latest
latest: Pulling from library/neo4j
66dbba0fb1b5: Pull complete
22fd3d2fa564: Pull complete
8c86bd7229f0: Pull complete
8eed4276cd40: Pull complete
10d0f080131c: Pull complete
Digest: sha256:986e76ae57e8b107a132997cce59705141b0954afb9e2ebfed3026b5cc412b61
Status: Downloaded newer image for neo4j:latest
docker.io/library/neo4j:latest
(base)
~ took 11s
❯
```

```
(base)
~
❯ docker run --publish=7474:7474 --publish=7687:7687 --env NEO4J_AUTH=neo4j/your
-password neo4j

Changed password for user 'neo4j'. IMPORTANT: this change will only take effect
if performed before the database is started for the first time.
2023-03-16 13:31:44.302+0000 INFO  Starting...
2023-03-16 13:31:44.561+0000 INFO  This instance is ServerId{05d5313e} (05d5313e
-c66e-4115-ad3c-64f60cda251c)
2023-03-16 13:31:44.953+0000 INFO  ======== Neo4j 5.5.0 ========
2023-03-16 13:31:45.827+0000 INFO  Bolt enabled on 0.0.0.0:7687.
2023-03-16 13:31:46.167+0000 INFO  Remote interface available at http://localhos
t:7474/
2023-03-16 13:31:46.170+0000 INFO  id: B11AB5AFCFFEAE1DA89C953F8B9A831EE8BC6DABA
E0C9AAF9DF6B78E567D47A6
2023-03-16 13:31:46.170+0000 INFO  name: system
2023-03-16 13:31:46.170+0000 INFO  creationDate: 2023-03-16T13:31:45.229Z
2023-03-16 13:31:46.170+0000 INFO  Started.
```

Modify JSON to several csv:

```python
def clean_csv(input_file, output_file):
    df = pd.read_csv(input_file, dtype=str)

    # Remove quotes from the entire DataFrame
    df = df.applymap(lambda x: x.replace('"', '') if isinstance(x, str) else x)
    # Identify the correct id column
    id_column = "_id" if "_id" in df.columns else "company_id"
    # Filter out rows with null values in the 'id' field
    df = df.dropna(subset=[id_column])
    # Save the cleaned DataFrame to a new CSV file without quotes around the column names
    df.to_csv(output_file, index=False, quoting=csv.QUOTE_NONE, escapechar='\\')
# Load the JSON data from file
with open("companies2.json", "r") as file:
    data = [json.loads(line) for line in tqdm(file)]
# Create DataFrames for nodes and relationships
companies_df = pd.DataFrame(columns=["_id", "name", "category_code", "founded_year"])
relationships_df = pd.DataFrame(columns=["company_id", "person", "title"])
people_df = pd.DataFrame(columns=["permalink", "first_name", "last_name"])
# Iterate through the data and populate the DataFrames
for entry in tqdm(data):
    company_id = entry["_id"]["$oid"]
    company = pd.DataFrame({
        "_id": [company_id],
        "name": [entry["name"]],
        "category_code": [entry["category_code"]],
        "founded_year": [entry.get("founded_year", None)]})
    companies_df = pd.concat([companies_df, company], ignore_index=True)
    for relationship in entry["relationships"]:
        relationship_df = pd.DataFrame({
            "company_id": [company_id],
            "person": [relationship["person"]["permalink"]],
            "title": [relationship.get("title", None)]})
        relationships_df = pd.concat([relationships_df, relationship_df], ignore_index=True)
        person = relationship["person"]
        person_df = pd.DataFrame({
            "permalink": [person["permalink"]],
            "first_name": [person.get("first_name", None)],
            "last_name": [person.get("last_name", None)]})
        people_df = pd.concat([people_df, person_df], ignore_index=True)
people_df = people_df.drop_duplicates(subset=["permalink"])
# Export to CSV files
companies_df.to_csv("companies_nodes.csv", index=False)
relationships_df.to_csv("relationships_nodes.csv", index=False)
people_df.to_csv("people_nodes.csv", index=False)
# Clean the CSV files
clean_csv("companies_nodes.csv", "clean_companies_nodes.csv")
clean_csv("relationships_nodes.csv", "clean_relationships_nodes.csv")
```

Eléonor KIOULOU
Adéle MONTOYA
Paul RUNAVOT



Import:

```
~
❯ docker cp ./companies_nodes.csv esilv-neo4j:/var/lib/neo4j/import/ && docker c
p ./competitions_nodes.csv esilv-neo4j:/var/lib/neo4j/import/ && docker cp ./fun
ding_rounds_nodes.csv esilv-neo4j:/var/lib/neo4j/import/ && docker cp ./relation
ships_nodes.csv esilv-neo4j:/var/lib/neo4j/import/ && docker cp ./clean_companie
s_nodes.csv esilv-neo4j:/var/lib/neo4j/import/ &&  docker cp ./clean_relationshi
ps_nodes.csv esilv-neo4j:/var/lib/neo4j/import/
```

```
1 LOAD CSV WITH HEADERS FROM 'file:///clean_relationships_nodes.csv' AS row
2 MATCH (c:Company {id: row.company_id}), (p:Person {permalink: row.person})
3 MERGE (c)-[r:HAS_RELATIONSHIP]→(p)
4 SET r.title = row.title;
```

```
1 LOAD CSV WITH HEADERS FROM 'file:///people_nodes.csv' AS row
2 MERGE (p:Person {permalink: row.permalink})
3 SET p.first_name = row.first_name, p.last_name = row.last_name;
```

Keys:

Convert "founded_years" in Company into integer (useful for query 2):

```
1 LOAD CSV WITH HEADERS FROM 'file:///clean_companies_nodes.csv' AS row
2 MERGE (c:Company { _id: row._id })
3 ON CREATE SET c.name = row.name, c.category_code = row.category_code, c.founded_year =
  toInteger(row.founded_year)
4
```

Table

(no changes, no records)

Eléonor KIOULOU
Adéle MONTOYA
Paul RUNAVOT

```
MATCH (c:Company)
WHERE c.founded_year IS NOT NULL
SET c.founded_year = toInteger(c.founded_year)
```

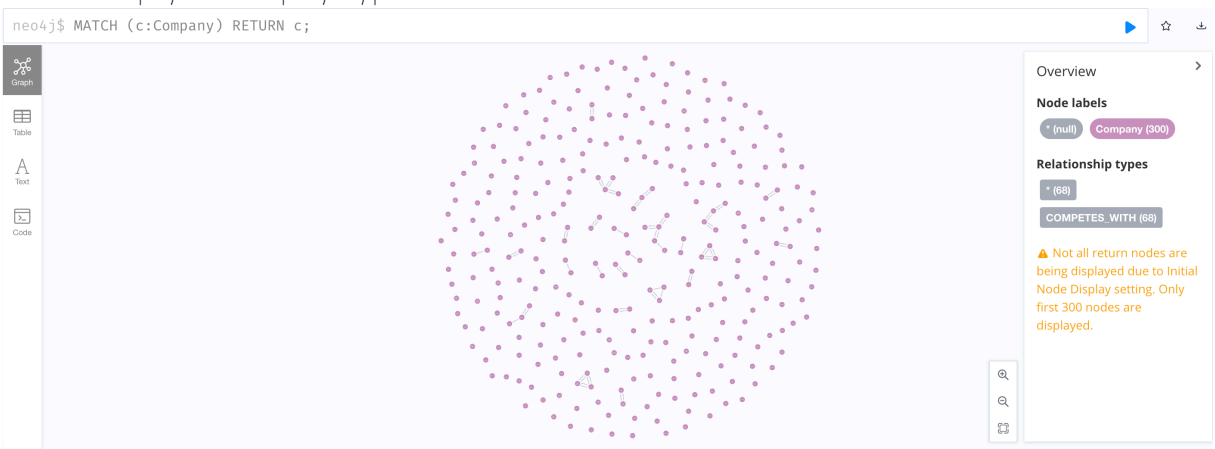
Table

Set 18801 properties, completed after 218 ms.

Simples Queries :

1. Display all "Company" type nodes:

```
neo4j$ MATCH (c:Company) RETURN c;
```



2. Show all companies founded after 2000 (display 5):

```
MATCH (c:Company)
WHERE c.founded_year > 2000
RETURN c.name, c.founded_year LIMIT 5
```

Table
Text
Code

	c.name	c.founded_year
1	"Orbis Biosciences"	2008
2	"Incogna"	2007
3	"Urbantastic!"	2008
4	"Cellopoint"	2003
5	"Writer4me"	2006

Started streaming 5 records after 1 ms and completed after 13 ms.

3. Find companies competing with a specific company "Vidyo":

Eléonor KIOULOU
Adéle MONTOYA
Paul RUNAVOT

```
neo4j$ MATCH (c1:Company {name: "Vidyo"})-[:COMPETES_WITH]-(c2:Company) RETURN c2.name
```

	c2.name
1	"Dimdim"
2	"ooVoo"
3	"Dimdim"

Started streaming 3 records after 2 ms and completed after 26 ms.

4. Find companies that participated in a "private_equity" round of financing:

```
1 MATCH (c:Company)-[:HAS_FUNDING_ROUND]→(f:FundingRound {round_code: "private_equity"})
2 RETURN c.name, f.round_code
```

	c.name	f.round_code
1	"GoodGuide"	"private_equity"
2	"Ascentis"	"private_equity"
3	"GrubHub"	"private_equity"
4	"Datapipe"	"private_equity"
5	"Kineto Wireless"	"private_equity"
6	"Pixelligent"	"private_equity"
7		

Started streaming 121 records in less than 1 ms and completed after 1 ms.

5. Returns the total number of relationships between Company and Person type nodes linked by the HAS_RELATIONSHIP relationship:

```
1 MATCH (c:Company)-[:HAS_RELATIONSHIP]→(person)
2 RETURN COUNT(*) as total_relationships;
```

	total_relationships
1	89159

6. Retrieve all people names stored in the database:

Eléonor KIOULOU
Adéle MONTOYA
Paul RUNAVOT

```
MATCH (p:Person)
RETURN p.first_name LIMIT 10
```

	p.first_name
1	"Mohsen"
2	"Eric"
3	"Margaret"
4	"Ulf"
5	"Peter"
6	"Ian"

Complex Queries:

1. Find everyone who has held a leadership position at Google:

```
MATCH (p:Person)-[:HAS_RELATIONSHIP]→(c:Company)
WHERE c.name = 'Google' AND p.title CONTAINS 'Director'
RETURN p.first_name, p.last_name, p.title
```

(no changes, no records)

2. Find all companies founded since 2000 that received type "b" funding:

```
MATCH (c:Company)-[:HAS_FUNDING_ROUND]→(f:FundingRound)
WHERE c.founded_year > 2000 AND f.round_code = 'b'
RETURN c.name, f.round_code
```

(no changes, no records)

3. Find the companies that have obtained type "b" financing and the list of people working for each of these companies:

```
MATCH (c:Company)-[:HAS_FUNDING_ROUND]→(f:FundingRound {round_code: 'b'})←
[:HAS_FUNDING_ROUND]-(comp:Company)←[:HAS_RELATIONSHIP]-(p:Person)
RETURN comp.name, collect(p.name) AS employees
```

(no changes, no records)

Hard Query :

Retrieves the names of the companies that have obtained the most type "a" or "b" funding and the names of the people working for these companies with the title "CEO":

Eléonor KIOULOU
Adéle MONTOYA
Paul RUNAVOT

```
1 MATCH (c:Company)-[:HAS_FUNDING_ROUND]→(f:FundingRound)
2 WHERE f.round_code IN ['a', 'b']
3 WITH c, COUNT(f) AS num_rounds
4 ORDER BY num_rounds DESC
5 LIMIT 5
6 MATCH (c)-[:HAS_RELATIONSHIP]→(p:Person)
7 WHERE p.title CONTAINS 'CEO'
8 RETURN c.name, p.name, p.title
9
```

Table

(no changes, no records)