

Experiment no 12

Name:- Nehal D. Ughade

PRN no :- 21520009

Course name:- ADSL Lab

Date- 27/04/2023

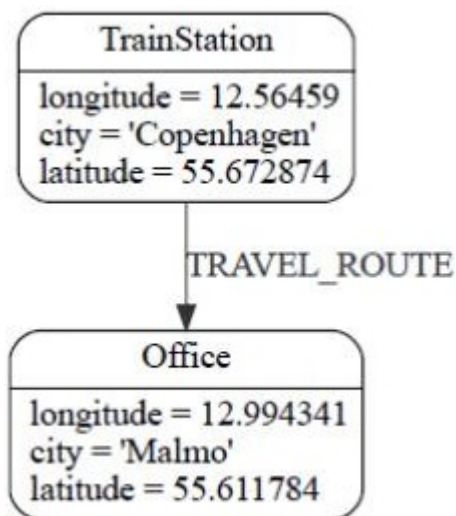
Spatial and Geographic Data

Geospatial is the natural domain for Graph Database Use Neo4j and Neo4j Spatial

Problem Statement : Finding Things Close to Other Things.

Task :

1. Use Neo4j graph database installed in previous assignments.
2. Install/configure Neo4jSpatial (<https://github.com/neo4j-contrib/spatial>) from GitHub. It is the Neo4j plug-in that facilitates geospatial operations on data stored in Neo4j.
3. Write CQL (Cypher Query Language) script to add randomly 10,000 location points as follows. Assume any data.

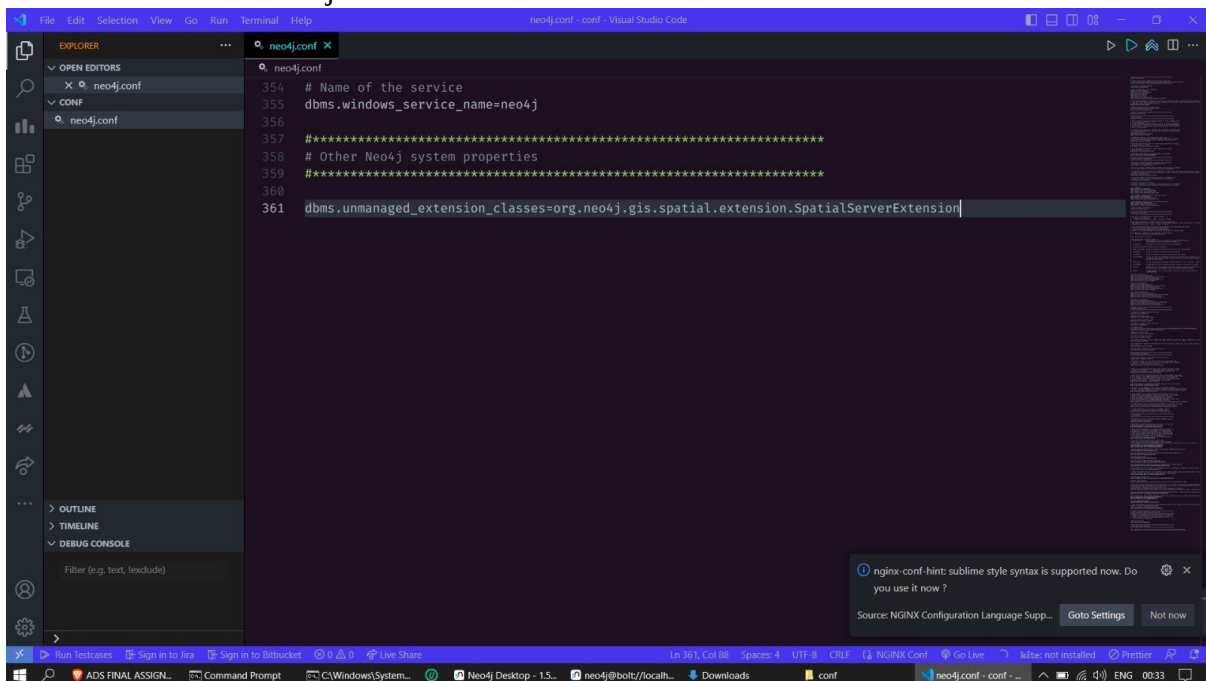


4. Use the point() , distance() function of Neo4j to answer the queries “which things close/nearest to which other things”.
5. Demonstrate the result by firing different cypher queries (write CQL statement Application in : location-based services on the web

Task

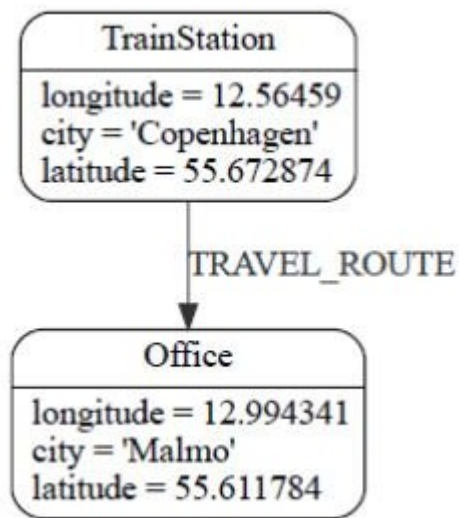
2. Configuration of the Neospatial master

- Download Neo4jSpatial from GitHub at <https://github.com/neo4j-contrib/spatial/releases>.
- You can download the latest release or a specific version that you want to use.
- Extract the downloaded file to a directory of your choice.
- Copy the extracted "neo4j-spatial-" directory to the "plugins" directory of your Neo4j installation.
- This directory is typically located at "neo4j-installation-directory/plugins/".
- Open the "neo4j.conf" file located in the "conf" directory of your Neo4j installation.
- Add the following line to the end of the file:
- Copy code
- `dbms.unmanaged_extension_classes=org.neo4j.gis.spatial.extension.SpatialServerExtension`
- This line enables the SpatialServerExtension, which provides the REST API for Neo4jSpatial.
- Save and close the "neo4j.conf" file.



- Restart Neo4j to apply the changes.

3. Write CQL (Cypher Query Language) script to add randomly 10,000 location points as follows. Assume any data.

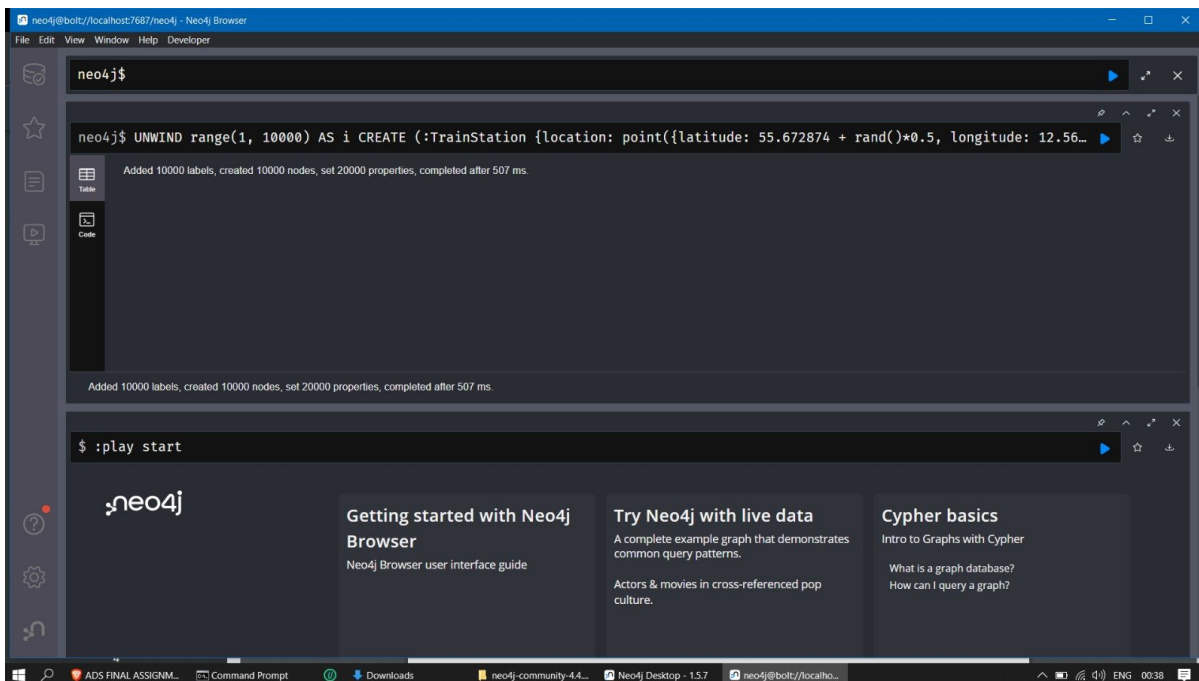


Cypher script :

```
// Create 10,000 Train Station nodes
```

```
UNWIND range(1, 10000) AS i
```

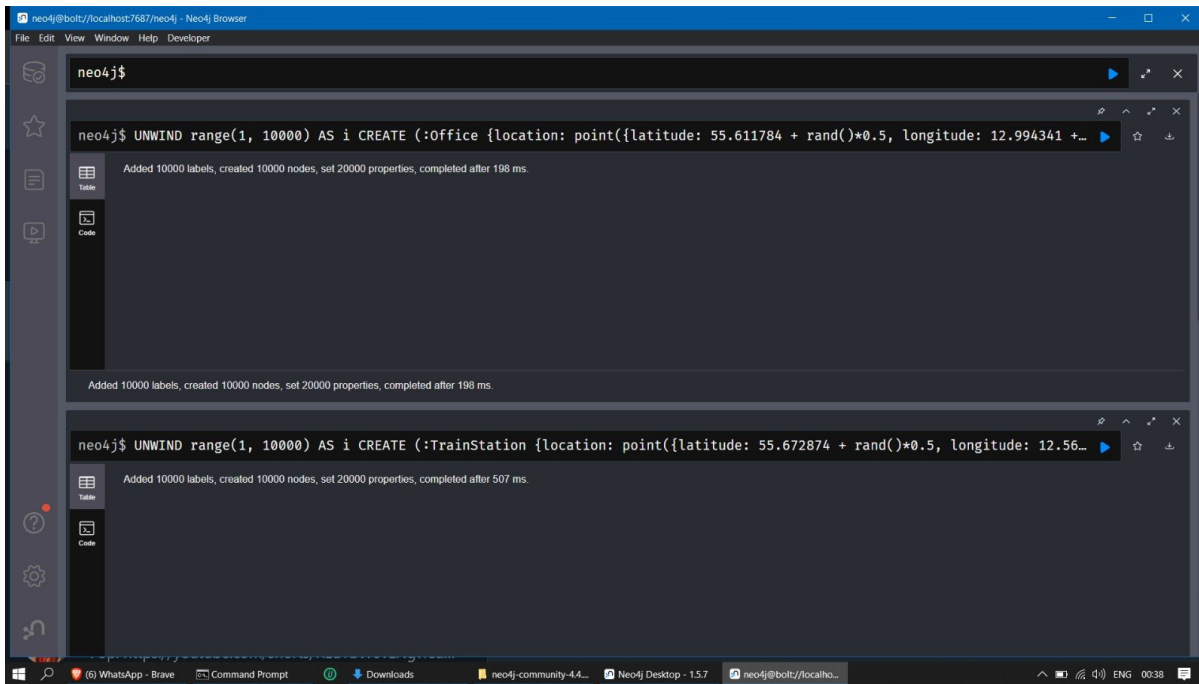
```
CREATE (:TrainStation {location: point({latitude: 55.672874 + rand()*0.5, longitude: 12.56459 + rand()*0.5}), city: 'Copenhagen'});
```



```
// Create 10,000 Office nodes
```

```
UNWIND range(1, 10000) AS i
```

```
CREATE (:Office {location: point({latitude: 55.611784 + rand()*0.5, longitude: 12.994341 + rand()*0.5}), city: 'Malmo'});
```



2. Find the closest Office to each Train Station in Copenhagen:

The screenshot displays the Neo4j Browser interface with two queries and their corresponding visualizations.

Query 1:
`neo4j$ MATCH (o1:Office {city : 'Malmo'}) MATCH (o2:Office {city : 'Malmo'}) WHERE o1<>o2 RETURN o1,o2 order by point.distance(...)`

Visualization 1: A graph view showing 20 orange circular nodes representing offices in Malmo. The nodes are scattered across the central workspace. The right sidebar shows "Overview" with "Node labels" as "Office (20)" and "Displaying 20 nodes, 0 relationships."

Query 2:
`neo4j$ MATCH (o:Office {city : 'Malmo'}) MATCH (t:TrainStation {city : 'Copenhagen'}) return t,o order by point.distance(t.loca...`

Visualization 2: A graph view showing two blue circular nodes representing a train station and an office. The right sidebar shows "Overview" with "Node labels" as "TrainStation (1)" and "Office (1)".

3. Find the closest Office to each other Office in Malmo.

The screenshot shows the Neo4j Browser interface. The top bar indicates the connection to 'neo4j@bolt://localhost:7687/neo4j'. The main query editor contains the following Cypher query:

```
neo4j$ MATCH (o1:Office {city:'Malmo'}) MATCH (o2:Office {city:'Malmo'}) WHERE o1 < o2 RETURN o1,o2 order by point.distance(...)
```

The results are displayed in a table view with two columns, o1 and o2. The first result shows two office nodes with their properties, including location coordinates. The second result shows the same two nodes in reverse order. A status message at the bottom of the results area states: 'Started streaming 20 records after 28 ms and completed after 60277 ms.'

Below the first query, a second query is visible in the editor:

```
neo4j$ MATCH (o:Office {city:'Malmo'}) MATCH (t:TrainStation {city:'Copenhagen'}) return t,o order by point.distance(t.loca...
```

The bottom of the image shows the Windows taskbar with various open applications, including 'ADS FINAL ASSIGNM...', 'Command Prompt', 'Downloads', 'neo4j-community-4.4...', 'Neo4j Desktop - 1.5.7', 'neo4j@bolt://localho...', and '12 - Notepad'. The system clock shows 'ENG 0051'.