NoSQL Graph
Centrality Algorithms

Data Engineering

UCI Division of Continuing Education

Graph Centrality Algorithms

- Degree centrality
- Closeness centrality
- Betweenness centrality
- PageRank

Degree Centrality

- Measures the number of relationships a node has in the graph
  - Incoming
  - Outgoing
- Compare a node to statistics for all nodes of the graph or subgraph
  - Average, median, minimum, maximum, standard deviation

Closeness Centrality

- Measures the average of the shortest path distances between a node and all other nodes
- Node with high closeness
  - Have the shortest distance to other nodes
  - Able to spread information most efficiently
- Compare a node to statistics for all nodes of the graph
  - Average, median, minimum, maximum, standard deviation

Betweenness Centrality

- Find all pairs’ shortest paths (weighted)
- For each node, how many paths pass through the node?
- Node with high betweenness
  - High number of paths passing through the node
  - Bridge
  - More influence over the flow within the graph
- Pivotal node: lies on every path between two other nodes

PageRank, Part I

- Larry Page of Google
- Overall influence of a node in a graph
  - Direct influence of a node
  - Influence of incoming relationships
  - Influence of incoming relationships of the incoming relationships
- Knowing a lot of influential people makes you more influential

PageRank, Part II

- Relationships
  - Directional
  - Weighted
  - Incoming relationships increase a node’s influence score
- Algorithm
  - Score each node by weighted incoming relationships
  - Iterate: each pass passes scores along to the outgoing relationships
  - Stop when scores converge or when a predetermined number of iterations has been reached

PageRank, Part III

- Random surfer
  - Basic algorithms assumes surfers are following links
  - Surfer does not follow links, moves on to something else
  - Solution: damping factor
- Rank sinks
  - No outbound relationships
  - Solution: random teleporting to another node

PageRank, Personalized

- Perspective from a single node
- What’s important to a specific user
- Target recommendations to a specific user

What centrality should be applied?

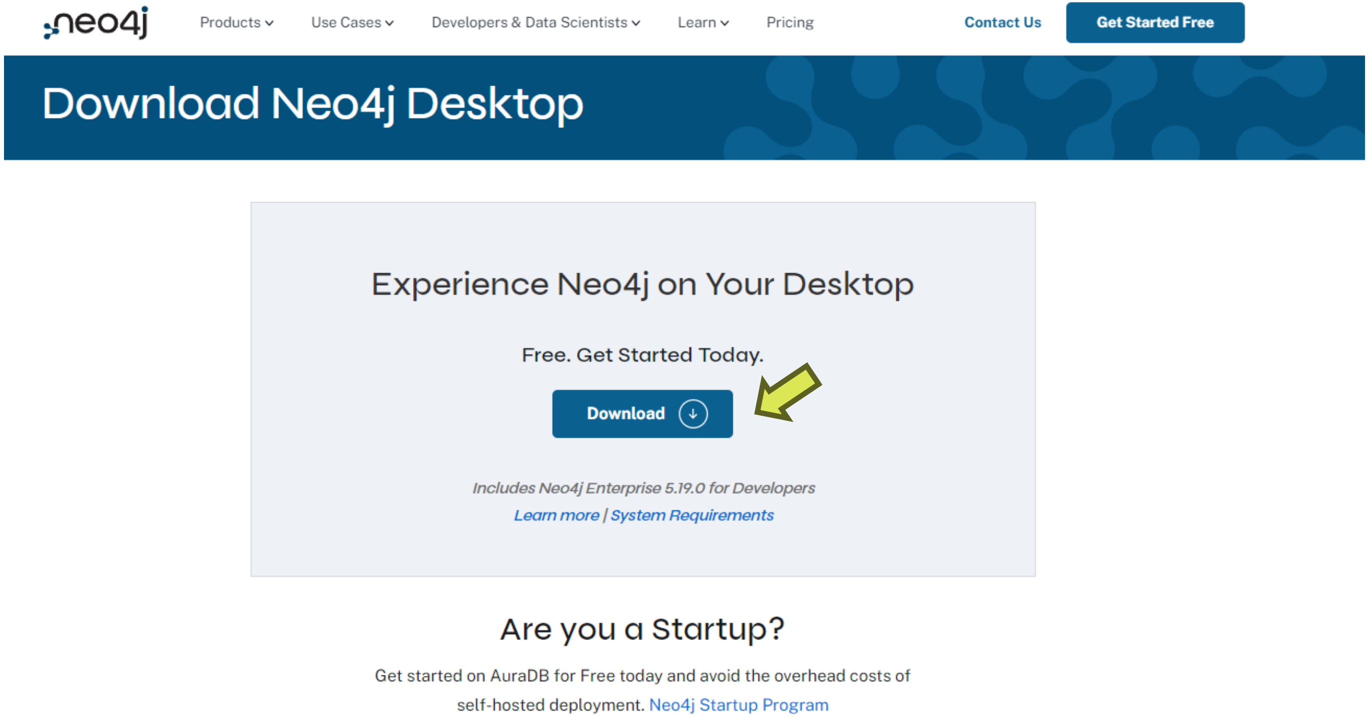
- Virus Infection
- Failure detection on Computer Network
- Interdisciplinary Employees
  - ➢ Betweenness
  - ➢ Degree centrality, in-degree vs. out-degree?
  - ➢ closeness

Centrality in Neo4j

- Virus Infection
- Failure detection on Computer Network
- Interdisciplinary Employees
  - Betweenness
  - Degree centrality, in-degree vs. out-degree?
  - closeness

neo4j.com

Download Neo4j Desktop



Download Neo4j Desktop

Experience Neo4j on Your Desktop

Free. Get Started Today.

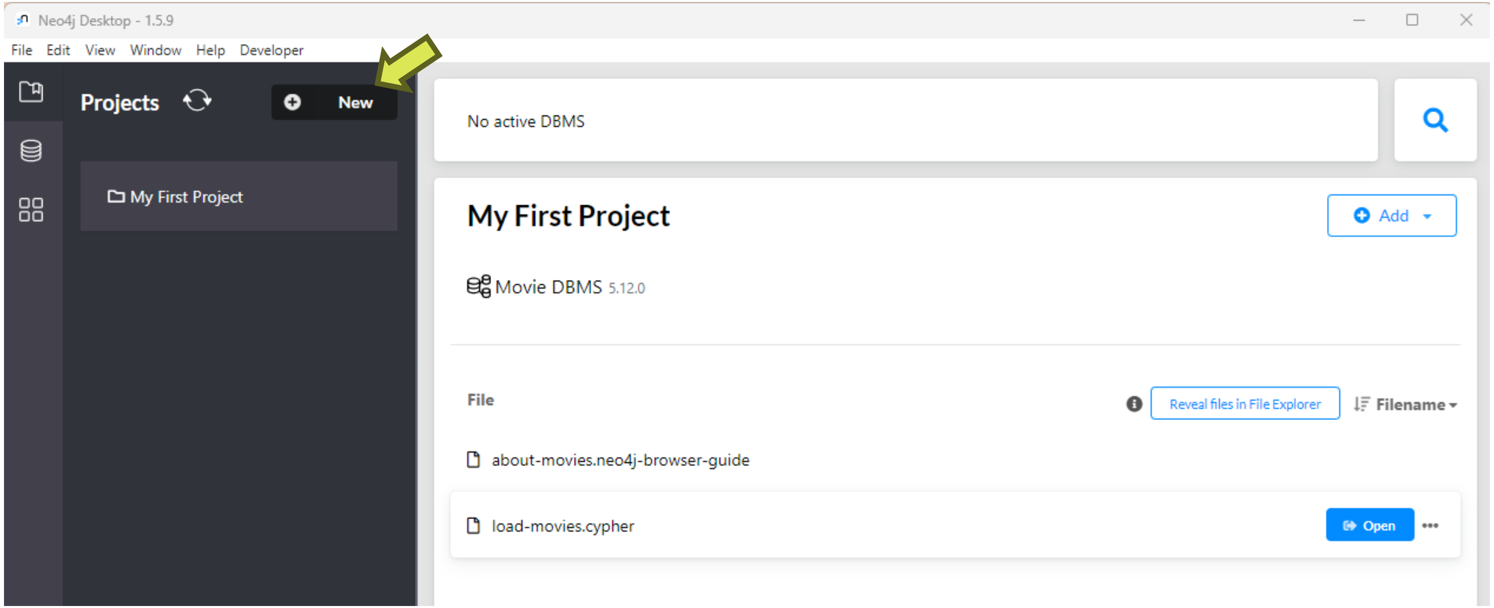
Includes Neo4j Enterprise 5.19.0 for Developers

Learn more | System Requirements

Are you a Startup?

Get started on AuraDB for Free today and avoid the overhead costs of self-hosted deployment. Neo4j Startup Program

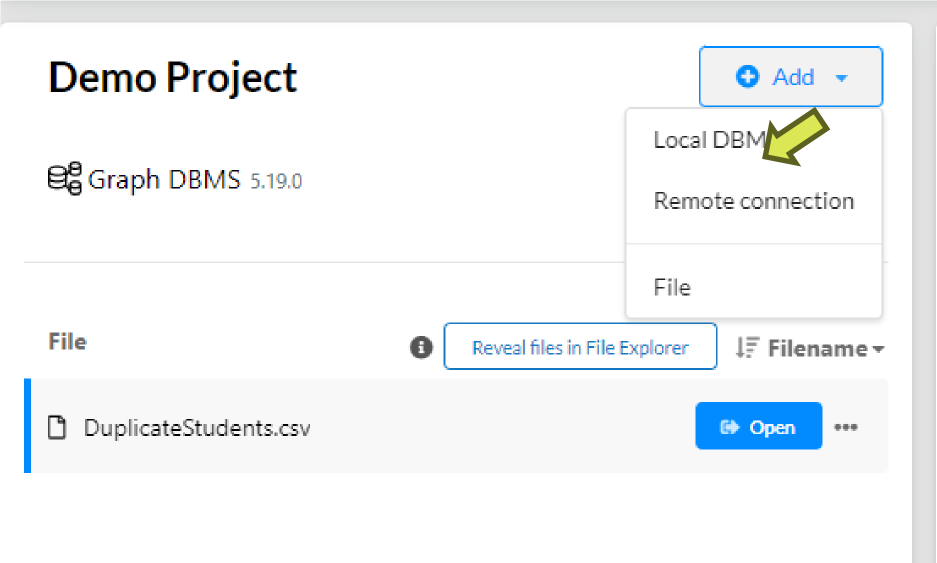
Open the Desktop & Create A New Project

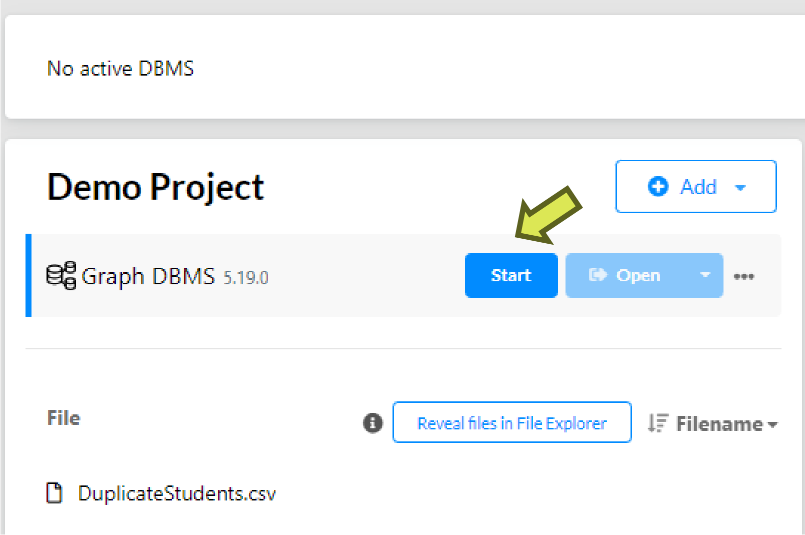


My First Project

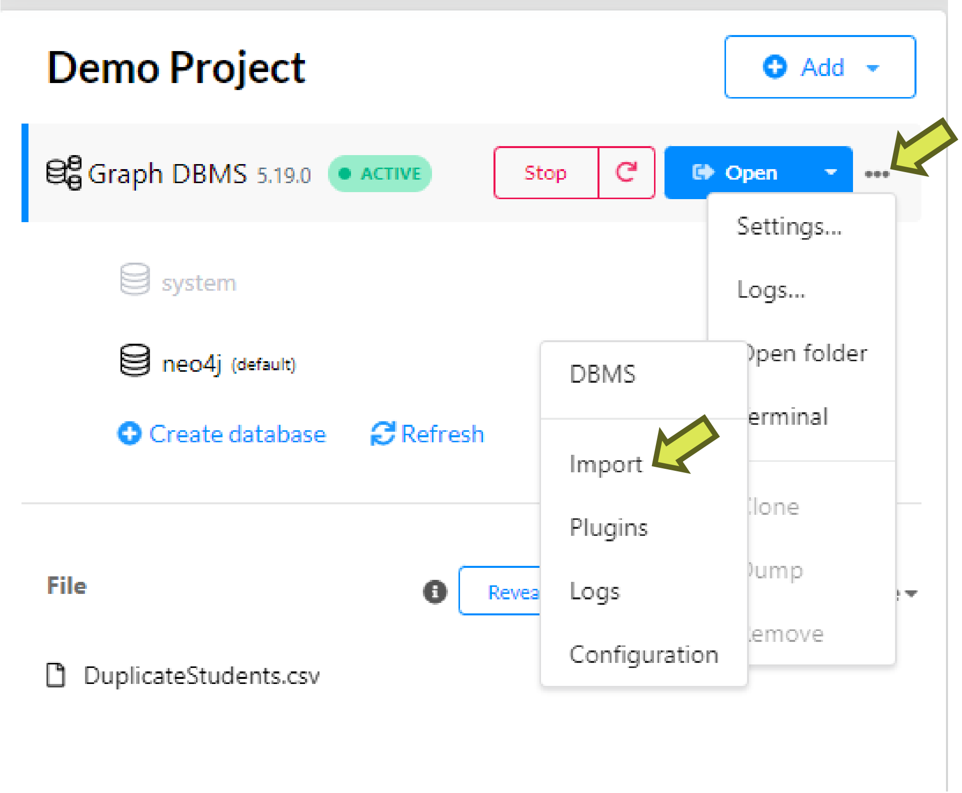
Add
Movie DBMS 5.12.0
File
Reveal files in File Explorer
Filename
about-movies.neo4j-browser-guide
load-movies.cypher
Open

Connect Local DBMS and Start It



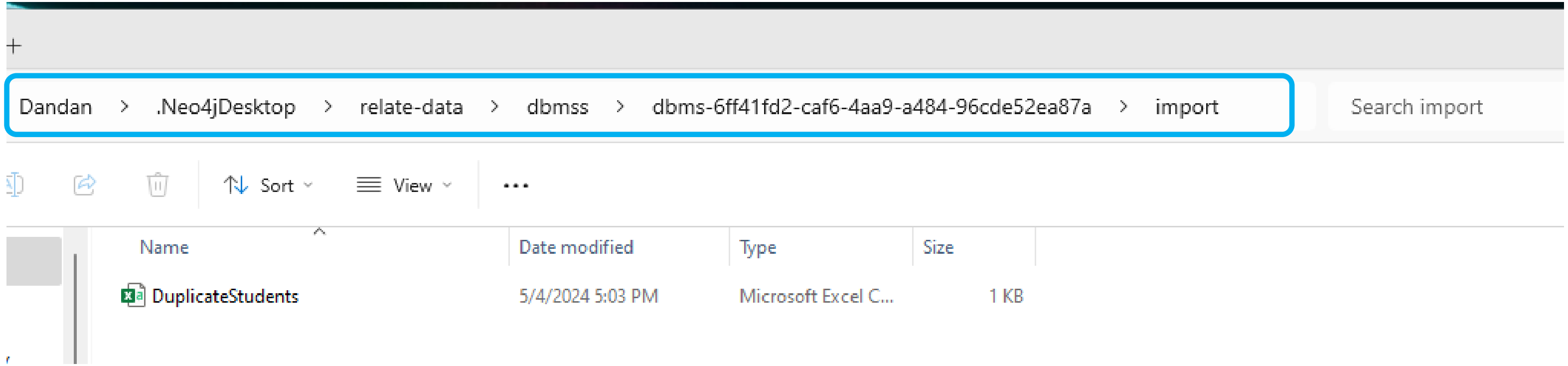


Import csv to DBMS



Import csv to DBMS

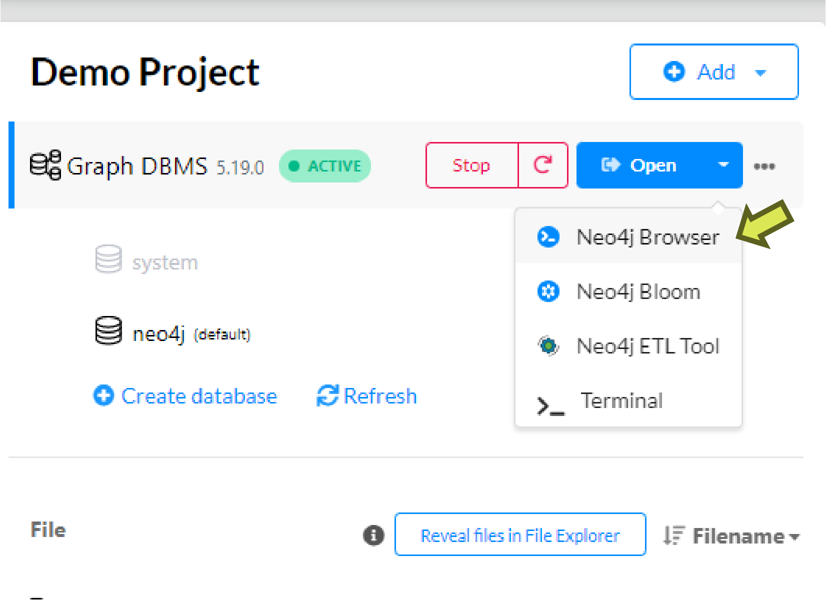
Save the dataset to the import folder



Dandan > .Neo4jDesktop > relate-data > dbmss > dbms-6ff41fd2-caf6-4aa9-a484-96cde52ea87a > import

Name	Date modified	Type	Size
DuplicateStudents	5/4/2024 5:03 PM	Microsoft Excel C...	1 KB

Open Neo4j Browser



Using Movie Database

Degree centrality

```
MATCH (actor:Person)-[:ACTED_IN]->(movie)
WITH actor, COUNT(*) AS numberOfMovies
RETURN actor.name AS actorName, numberOfMovies
ORDER BY numberOfMovies DESC
```

Calculate betweenness centrality for actors (simplified)

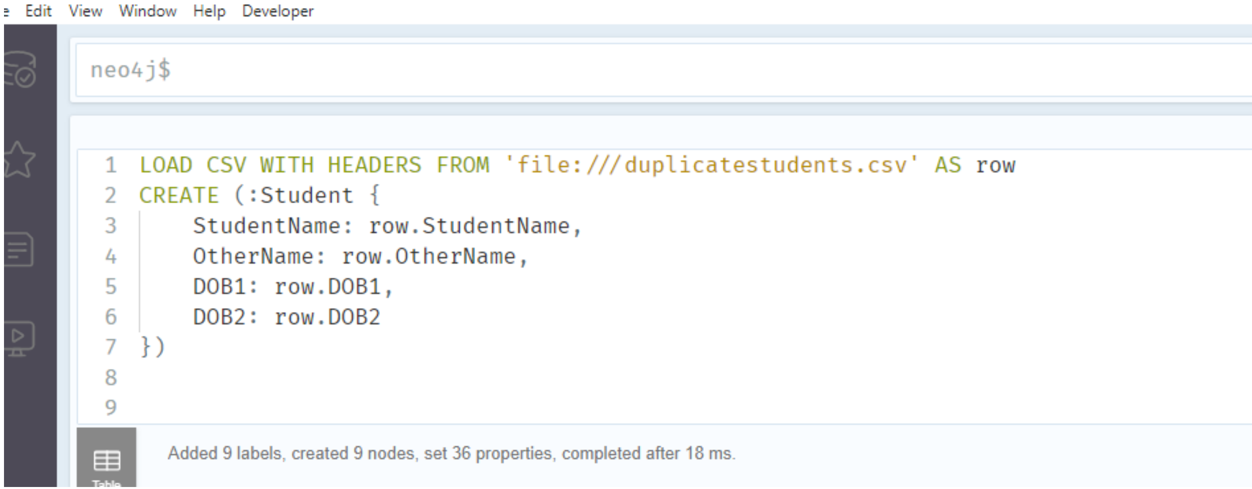
```
MATCH (a1:Person), (a2:Person)
WHERE id(a1) < id(a2)
MATCH p = shortestPath((a1)-[*]-(a2))
WITH a1, a2, p
UNWIND nodes(p) AS actor
WITH a1, a2, actor
WHERE actor:Person // Ensure that only actor nodes are considered
RETURN actor.name AS actorName, count(DISTINCT a1) + count(DISTINCT a2) AS betweennessCentrality
ORDER BY betweennessCentrality DESC
```

Calculate Closeness (simplified)

```
MATCH (actor:Person)
WITH collect(actor) AS actors
UNWIND actors AS actor
WITH actor, size(actors) - 1 AS numActors
LIMIT 1000 // Limit the number of actors to process at once
MATCH (actor)-[*1..3]-(otherActor:Person) // Limit the maximum path length and only consider other actors
WHERE actor <> otherActor
WITH actor, numActors, sum(length(shortestPath((actor)-[*]-(otherActor))) + 1) AS totalShortestPathLength
WITH actor, numActors, totalShortestPathLength, numActors - 1 AS numActorsMinusOne
RETURN actor.name AS actorName, numActorsMinusOne / totalShortestPathLength AS closenessCentrality
ORDER BY closenessCentrality DESC
```

Using Uploaded Dataset

Load and Create A Table

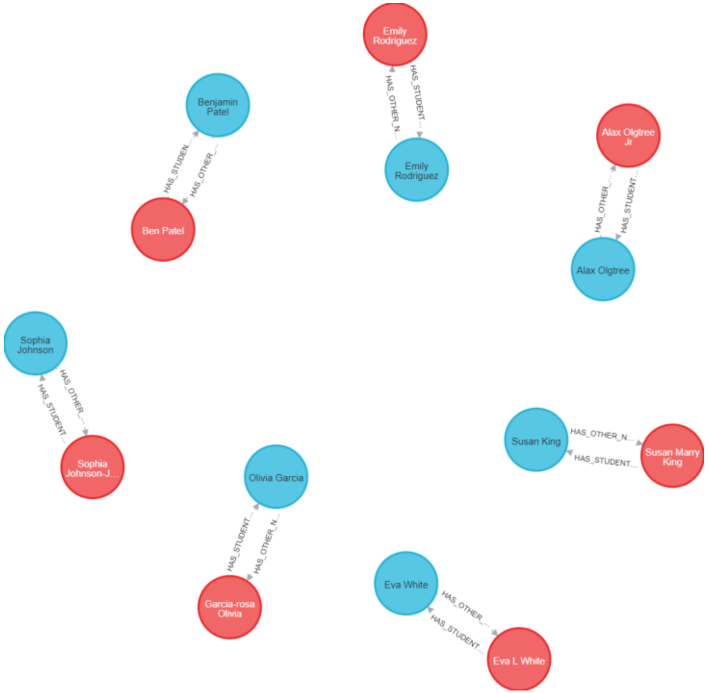


```
LOAD CSV WITH HEADERS FROM 'file:///duplicatestudents.csv' AS row
CREATE (:Student {
    StudentName: row.StudentName,
    OtherName: row.OtherName,
    DOB1: row.DOB1,
    DOB2: row.DOB2
})
```

Added 9 labels, created 9 nodes, set 36 properties, completed after 18 ms.

Build relationships

```
LOAD CSV WITH HEADERS FROM 'file:///duplicatest
udents.csv' AS row
MERGE (student:Student {name: row.StudentName})
MERGE (other:Other {name: row.OtherName})
MERGE (student)-[:HAS_DOB]-
>(:DOB {date: row.DOB1})
MERGE (other)-[:HAS_DOB]-
>(:DOB {date: row.DOB2})
WITH student, other
WHERE student.name <> other.name
MERGE (student)-[:HAS_OTHER_NAME]->(other)
MERGE (other)-[:HAS_STUDENT_NAME]->(student)
RETURN student, other
```



Stop the project when you are done