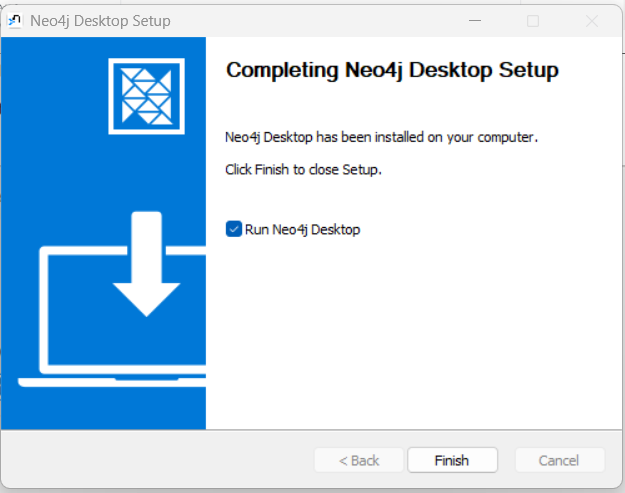
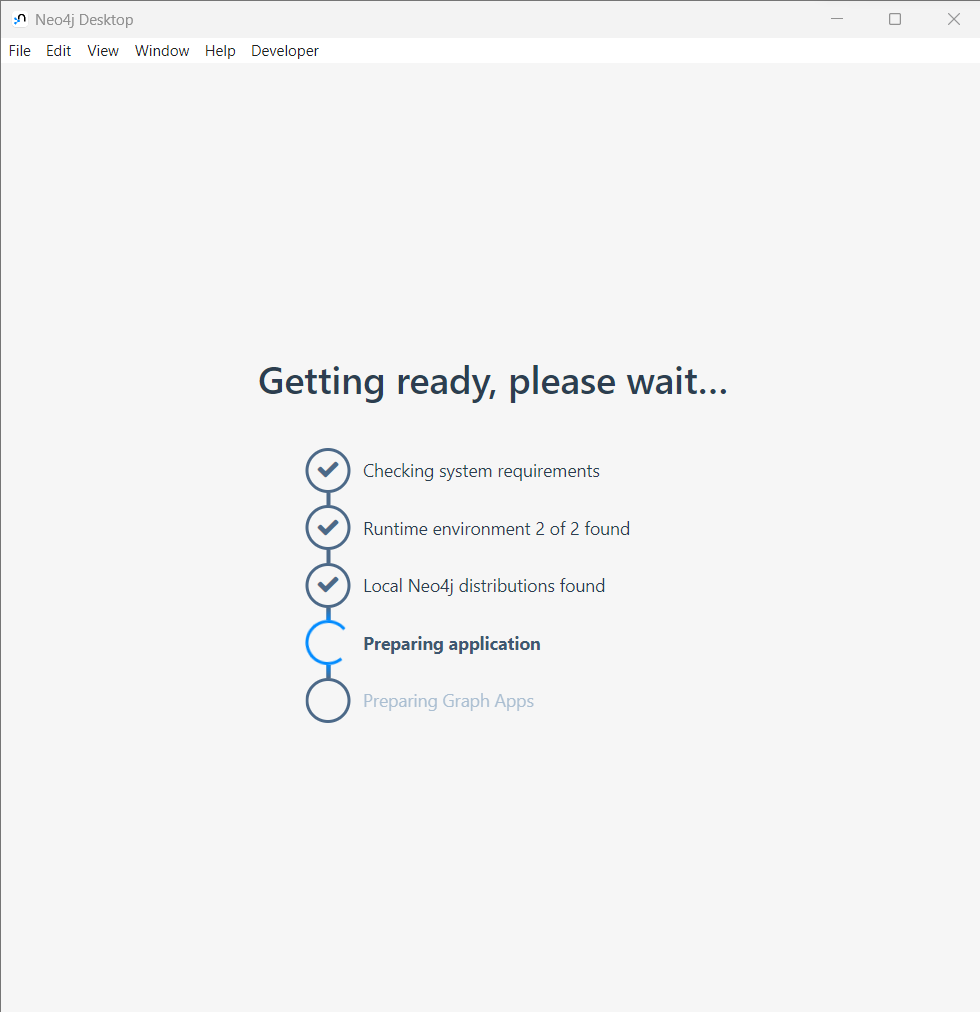
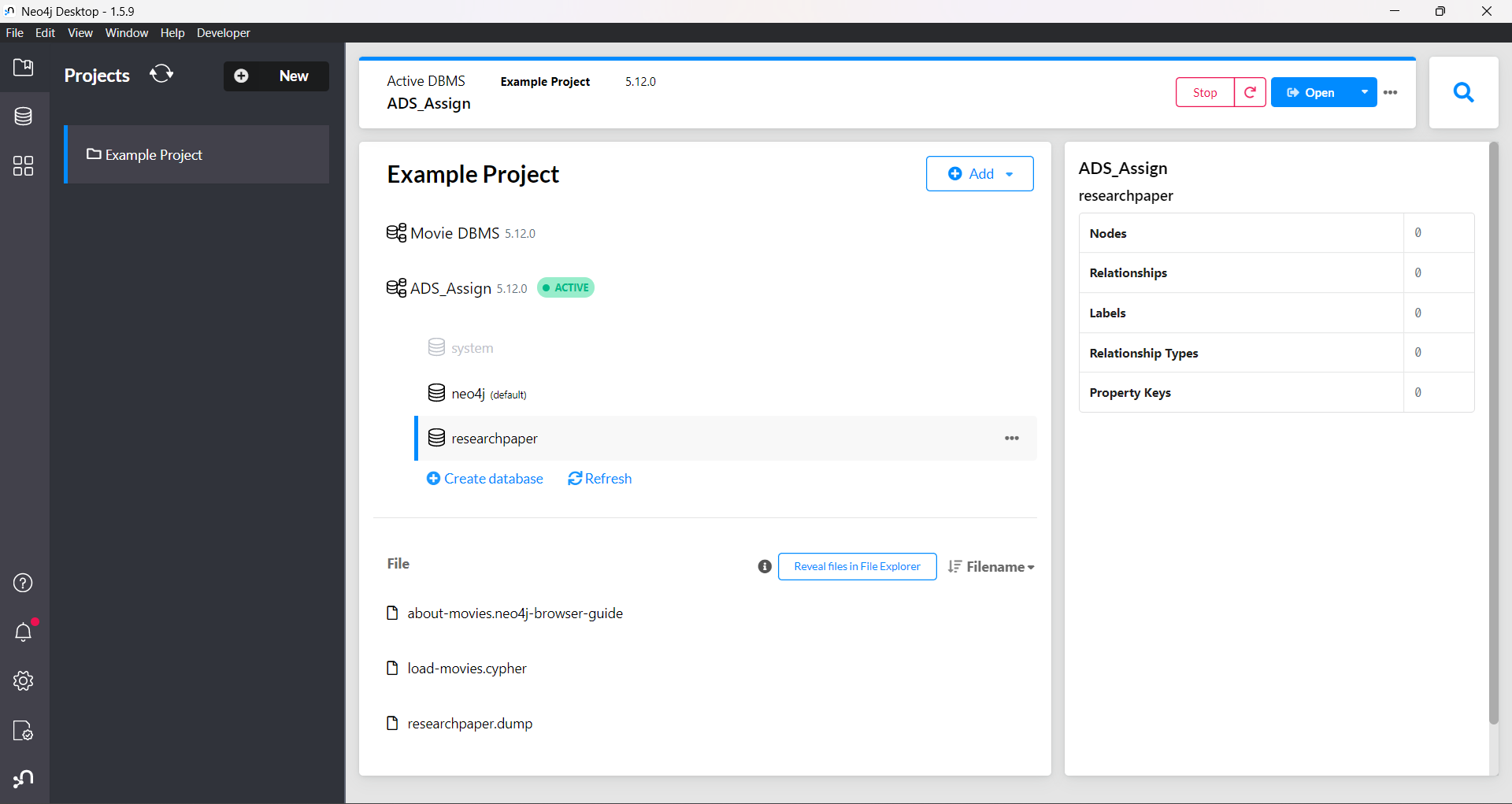
**Assignment - 11**

Consider the “**Research Papers Database”** scenario as follows :

The research papers have authors (often more than one). Most papers have a classification (what the paper is about). The classifications form a hierarchy in several levels (for example, the classification “Databases” has the subclassifications “Relational” and “Object-Oriented”). A paper usually has a list of references, which are other papers. These are called citations.

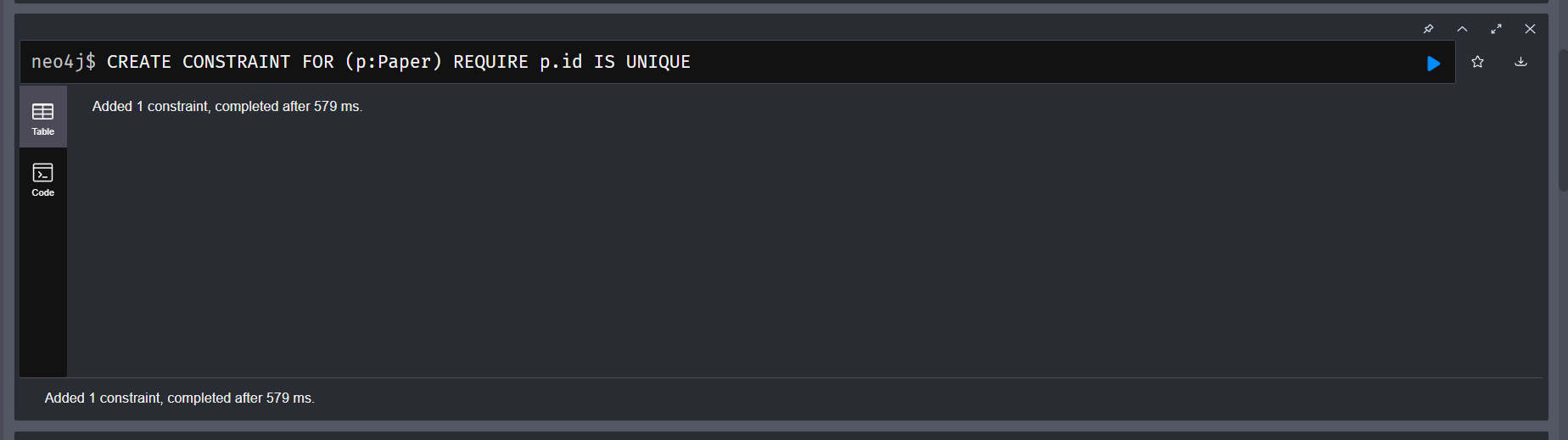


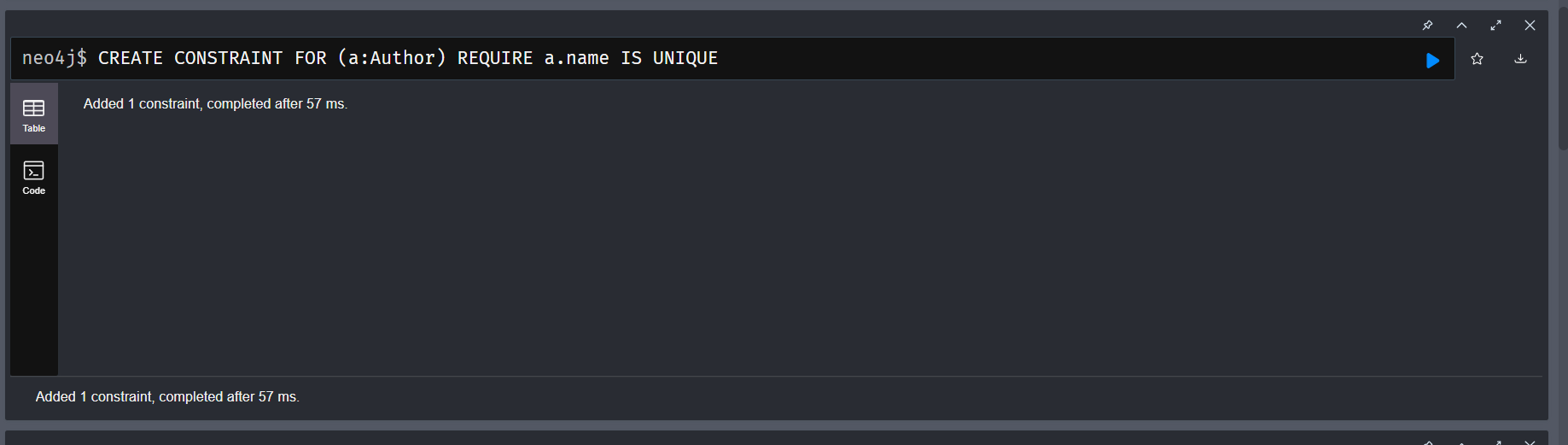


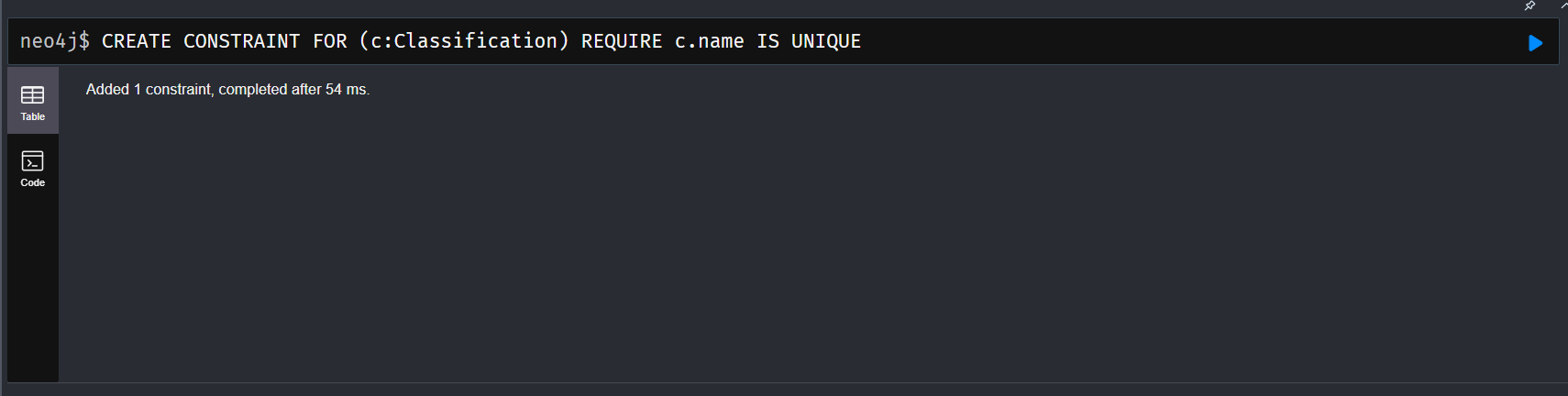


1. Design/model the graph database using Neo4j for above scenario.

1. Download the raw data from **Cora Research Paper Classification Project** : <http://people.cs.umass.edu/~mccallum/data.html>The database contains approximately 25,000 authors, 37,000 papers and 220,000 relationships.

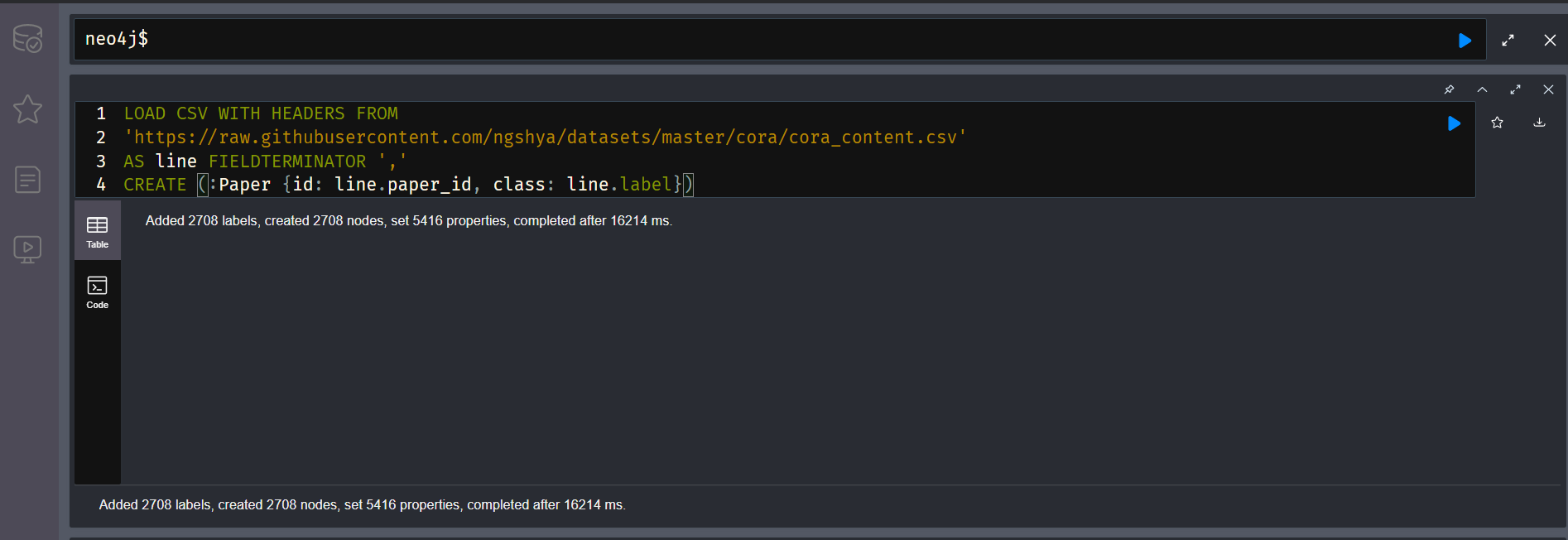








1. Load this data using Neo4j Data Browser

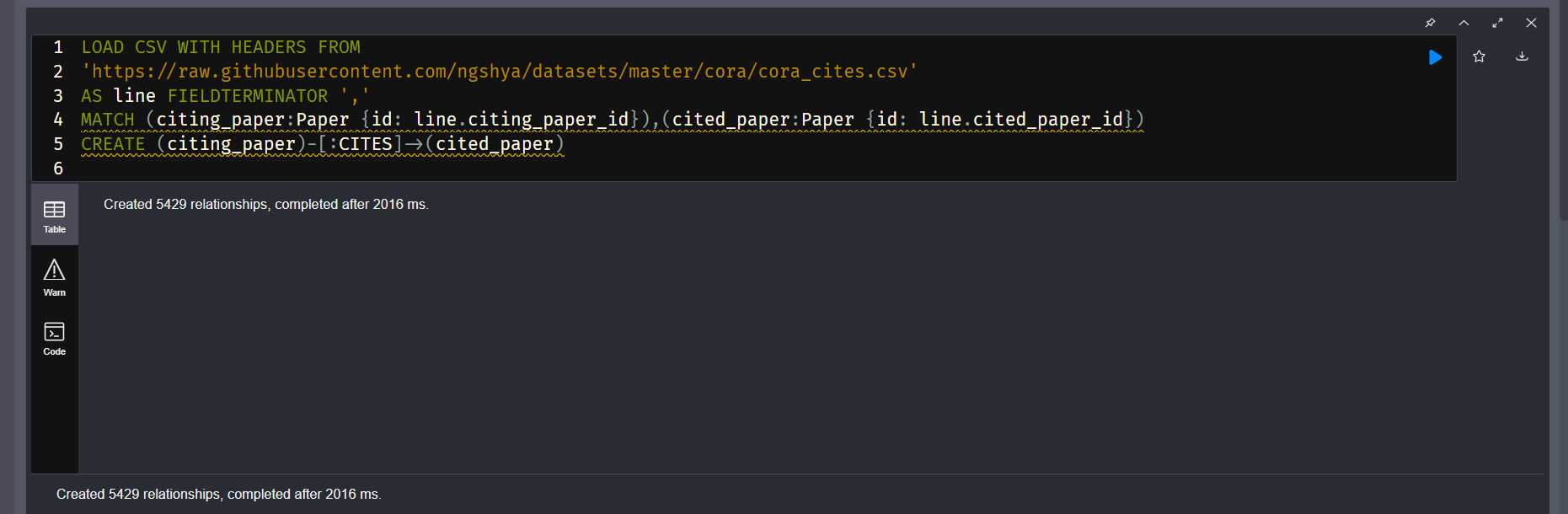


LOAD CSV WITH HEADERS FROM

'https://raw.githubusercontent.com/ngshya/datasets/master/cora/cora\_content.csv'

AS line FIELDTERMINATOR ','

CREATE (:Paper {id: line.paper\_id, class: line.label})



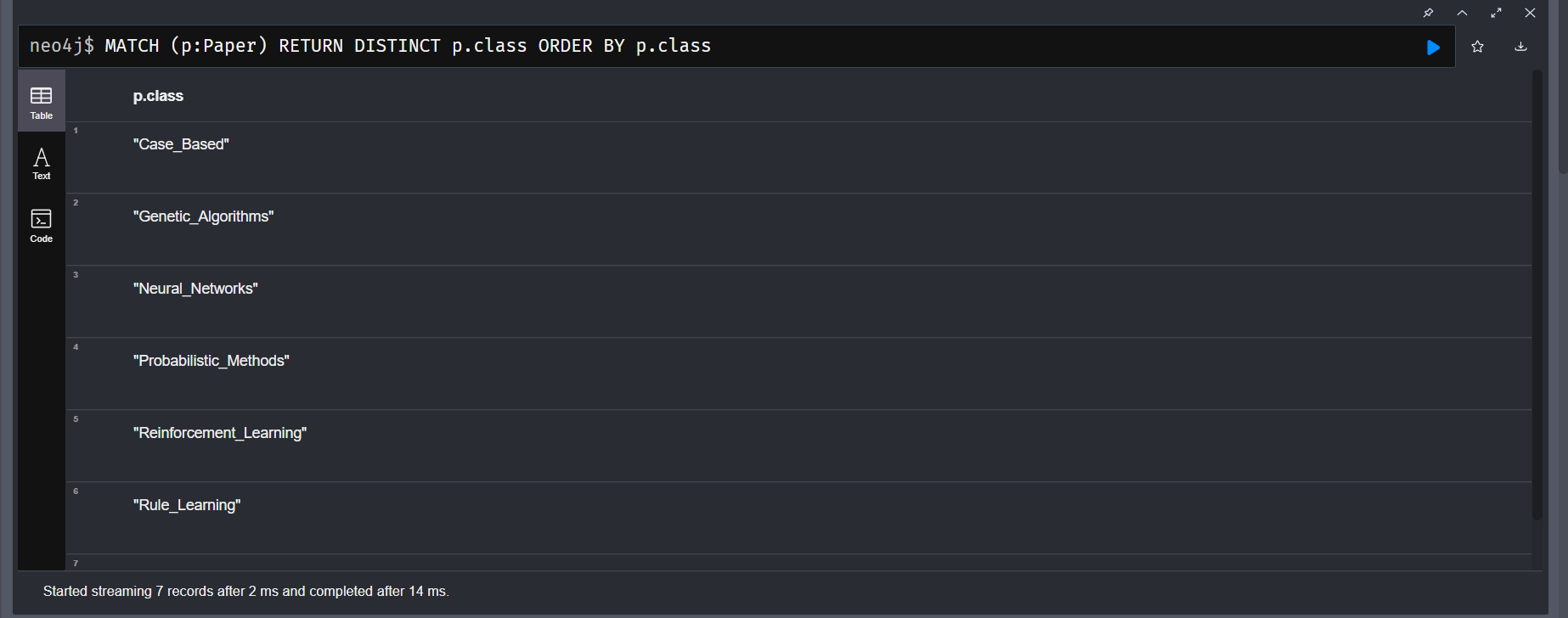
LOAD CSV WITH HEADERS FROM

'https://raw.githubusercontent.com/ngshya/datasets/master/cora/cora\_cites.csv'

AS line FIELDTERMINATOR ','

MATCH (citing\_paper:Paper {id: line.citing\_paper\_id}),(cited\_paper:Paper {id: line.cited\_paper\_id})

CREATE (citing\_paper)-[:CITES]->(cited\_paper)

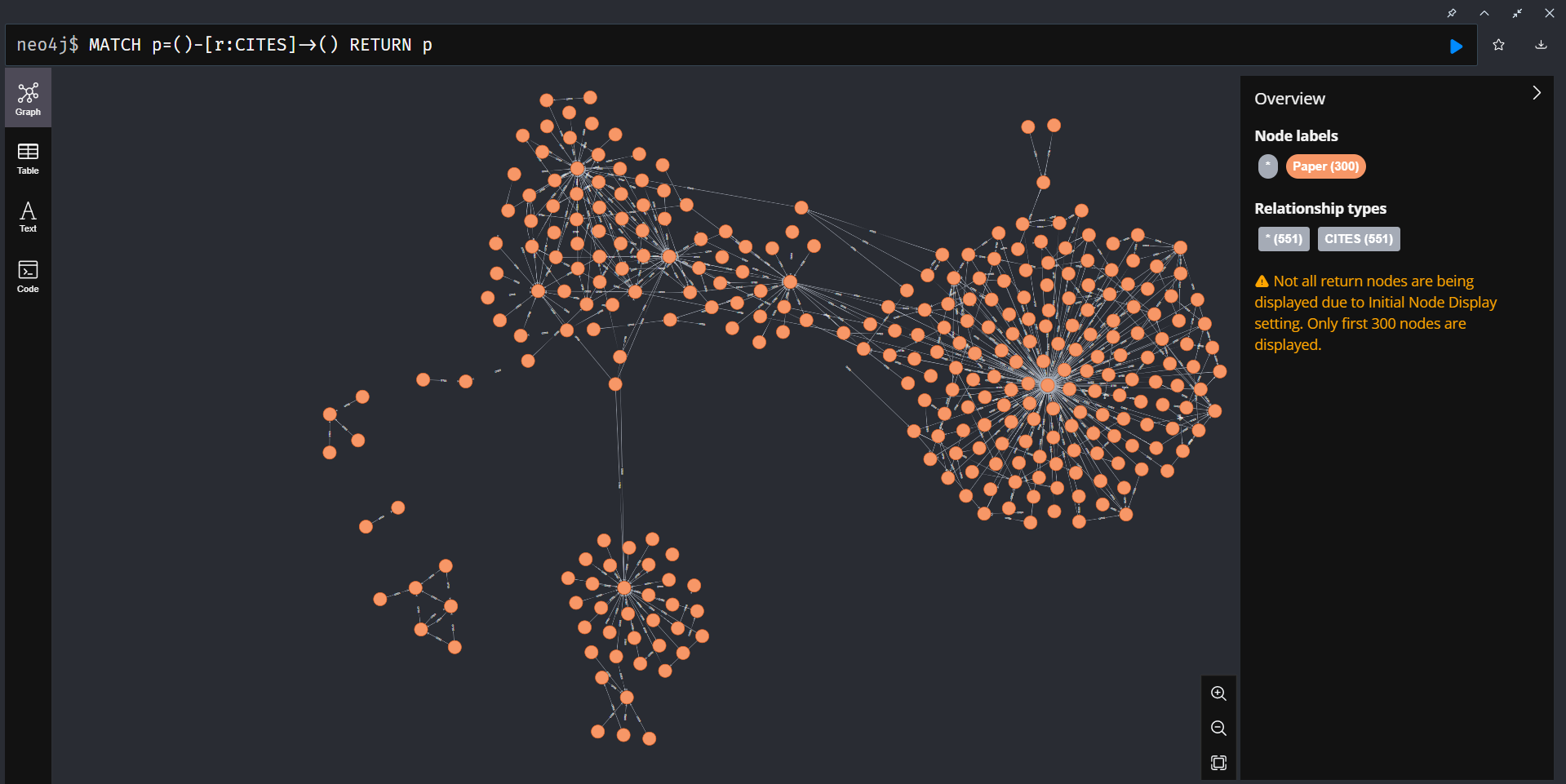


MATCH (p:Paper) RETURN DISTINCT p.class ORDER BY p.class



MATCH p=()-[r:CITES]->() RETURN p

1. Design the python-based desktop application for any kind of search on above database. The application should able to answer queries like



PYTHON APPLICATION:

import tkinter as tk

from neo4j import GraphDatabase

# Neo4j database connection details

uri = "bolt://127.0.0.1:7687"

username = "neo4j"

password = "adsassign"

# Initialize the Neo4j driver

try:

    driver = GraphDatabase.driver(uri, auth=(username, password))

except Exception as e:

    print(f"Failed to connect to Neo4j: {e}")

    exit()

# Function to check if paper A cites paper B

def does\_paper\_a\_cite\_paper\_b(tx, paper\_a\_id, paper\_b\_id):

    query = (

        "MATCH (a:Paper {paper\_id: $paper\_a\_id})-[:CITATION\*..3]->(b:Paper {paper\_id: $paper\_b\_id}) "

        "RETURN count(\*) > 0"

    )

    result = tx.run(query, paper\_a\_id=paper\_a\_id, paper\_b\_id=paper\_b\_id)

    return result.single()[0]

# Function to get the full classification of a paper

def get\_classification\_of\_paper(tx, paper\_id):

    query = (

        "MATCH (p:Paper {paper\_id: $paper\_id})-[:CLASSIFICATION\*]->(c:Classification) "

        "RETURN c.name"

    )

    result = tx.run(query, paper\_id=paper\_id)

    return [record['c.name'] for record in result]

# Function to handle search button click

def search():

    # Get the paper IDs from the entry widgets

    paper\_a\_id = entry\_paper\_a\_id.get()

    paper\_b\_id = entry\_paper\_b\_id.get()

    paper\_id = entry\_paper\_id.get()

    # Validate input

    if not paper\_a\_id or not paper\_b\_id or not paper\_id:

        label\_a\_b.config(text="Please enter all paper IDs")

        label\_a\_cite\_b.config(text="")

        label\_classification.config(text="")

        return

    # Open a new Neo4j session

    with driver.session() as session:

        try:

            # Check if paper A cites paper B

            result\_a\_b = does\_paper\_a\_cite\_paper\_b(session, paper\_a\_id, paper\_b\_id)

            label\_a\_b.config(text="Yes" if result\_a\_b else "No")

            # Check if paper A cites a paper that cites paper B

            result\_a\_cite\_b = False

            for i in range(3):

                result\_a\_cite\_b = does\_paper\_a\_cite\_paper\_b(session, paper\_a\_id, paper\_b\_id)

                if result\_a\_cite\_b:

                    break

            label\_a\_cite\_b.config(text="Yes" if result\_a\_cite\_b else "No")

            # Get the full classification of the paper

            result\_classification = get\_classification\_of\_paper(session, paper\_id)

            label\_classification.config(text="/".join(result\_classification))

        except Exception as e:

            print(f"Query execution failed: {e}")

            label\_a\_b.config(text="")

            label\_a\_cite\_b.config(text="")

            label\_classification.config(text="Error executing query")

# Create the main window

window = tk.Tk()

window.title("Research Papers Database")

# Create the widgets

label\_paper\_a\_id = tk.Label(window, text="Paper A ID:")

entry\_paper\_a\_id = tk.Entry(window)

label\_paper\_b\_id = tk.Label(window, text="Paper B ID:")

entry\_paper\_b\_id = tk.Entry(window)

button\_search\_citations = tk.Button(window, text="Search Citations", command=search)

label\_a\_b = tk.Label(window, text="")

label\_a\_cite\_b = tk.Label(window, text="")

label\_paper\_id = tk.Label(window, text="Paper ID:")

entry\_paper\_id = tk.Entry(window)

button\_search\_classification = tk.Button(window, text="Search Classification", command=search)

label\_classification = tk.Label(window, text="")

# Pack the widgets

label\_paper\_a\_id.pack()

entry\_paper\_a\_id.pack()

label\_paper\_b\_id.pack()

entry\_paper\_b\_id.pack()

button\_search\_citations.pack()

label\_a\_b.pack()

label\_a\_cite\_b.pack()

label\_paper\_id.pack()

entry\_paper\_id.pack()

button\_search\_classification.pack()

label\_classification.pack()

# Run the main loop

window.mainloop()

