TY B.Tech. (CSE) – II [2022-23]

5CS372: Advanced Database System Lab.

Assignment No. 11

Name : Tanaya Mukund Bhide

PRN : 2020BTECS00011

Batch : T5

Branch: T.Y CSE

Neo4j Graph Database

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 sub-classifications “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.
2. 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.
3. Load this data using Neo4j Data Browser
4. Design the python based desktop application for any kind of search on above database. The application should able to answer queries like

a) Does paper A cite paper B? If not directly, does paper A cite a paper which in its turn cites paper B? And so on, in several levels.

b) Show the full classification of a paper (for example, Databases / Relational)

Introduction:

A graph database stores nodes and relationships instead of tables, or documents. Data is stored just like you might sketch ideas on a whiteboard. Your data is stored without restricting it to a pre-defined model, allowing a very flexible way of thinking about and using it.

Theory:

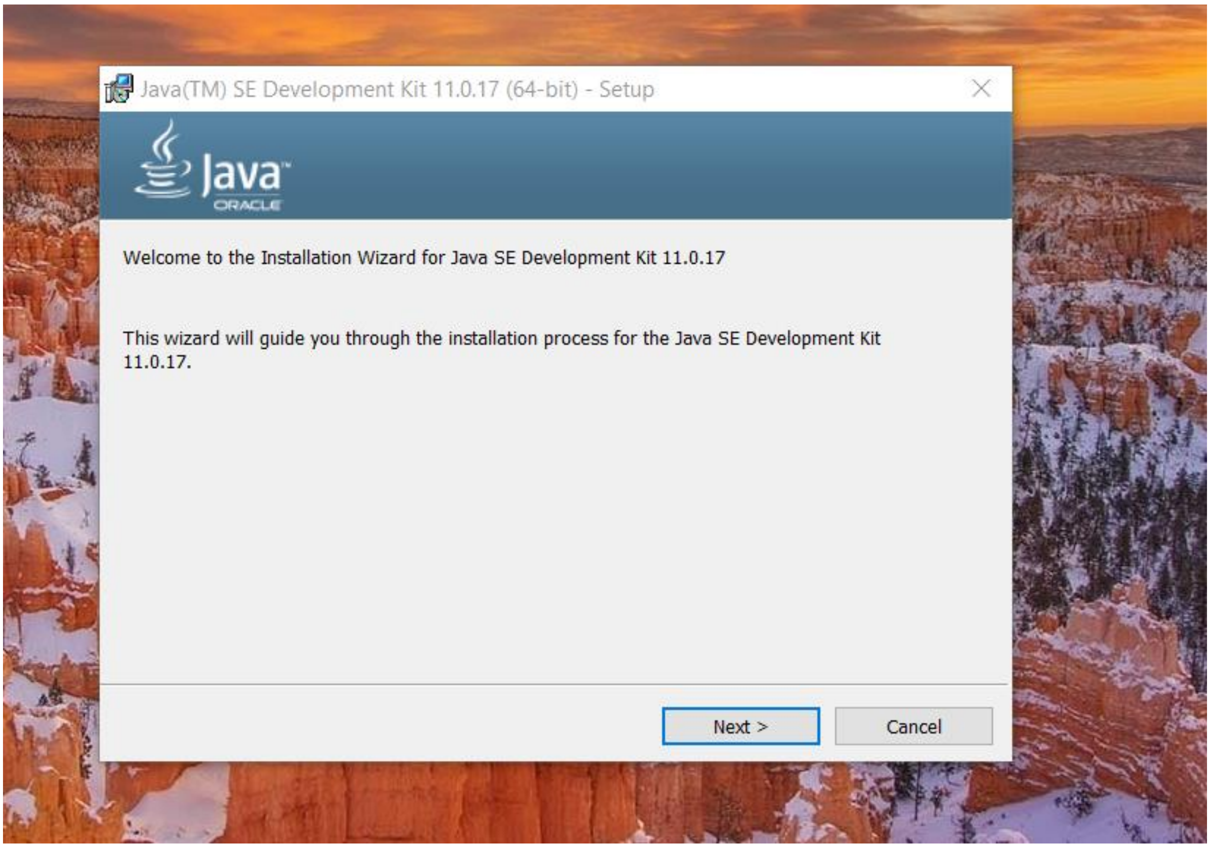
Neo4j is the world's leading graph database. The architecture is designed for optimal management, storage, and traversal of nodes and relationships. The graph database takes a property graph approach, which is beneficial for both traversal performance and operations runtime.

Cypher

Cypher is Neo4j's graph query language that allows users to store and retrieve data from the graph database. It is a declarative, SQL-inspired language for describing visual patterns in graphs using ASCII-art syntax. The syntax provides a visual and logical way to match patterns of nodes and relationships in the graph. Cypher has been designed to be easy to learn, understand, and use for everyone, but also incorporate the power and functionality of other standard data access languages

Procedure:

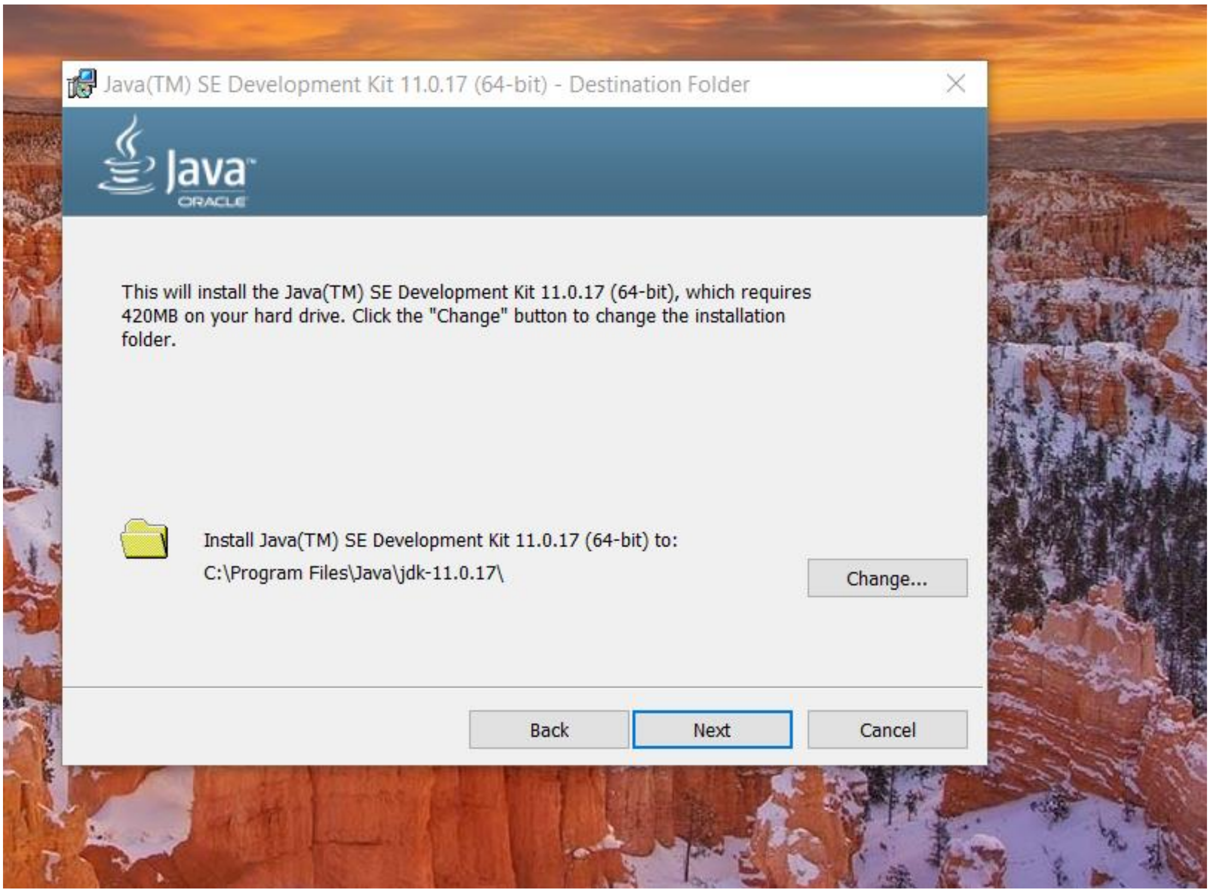
Downloaded JDK11 as per software requirements.



Java(TM) SE Development Kit 11.0.17 (64-bit) - Setup

Welcome to the Installation Wizard for Java SE Development Kit 11.0.17

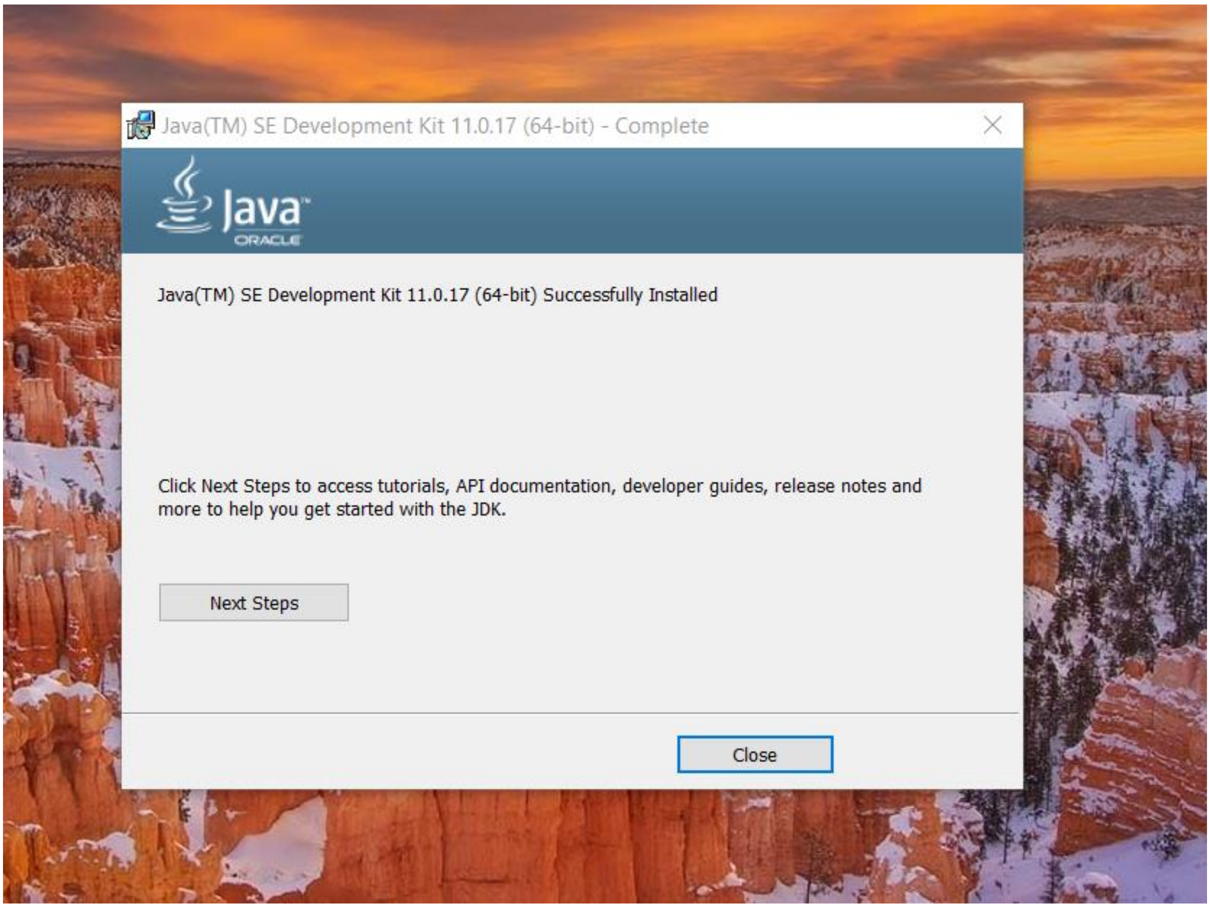
This wizard will guide you through the installation process for the Java SE Development Kit 11.0.17.


Java(TM) SE Development Kit 11.0.17 (64-bit) - Destination Folder
Java
ORACLE

This will install the Java(TM) SE Development Kit 11.0.17 (64-bit), which requires 420MB on your hard drive. Click the "Change" button to change the installation folder.

Install Java(TM) SE Development Kit 11.0.17 (64-bit) to:
C:\Program Files\Java\jdk-11.0.17\

Change...
Back
Next
Cancel

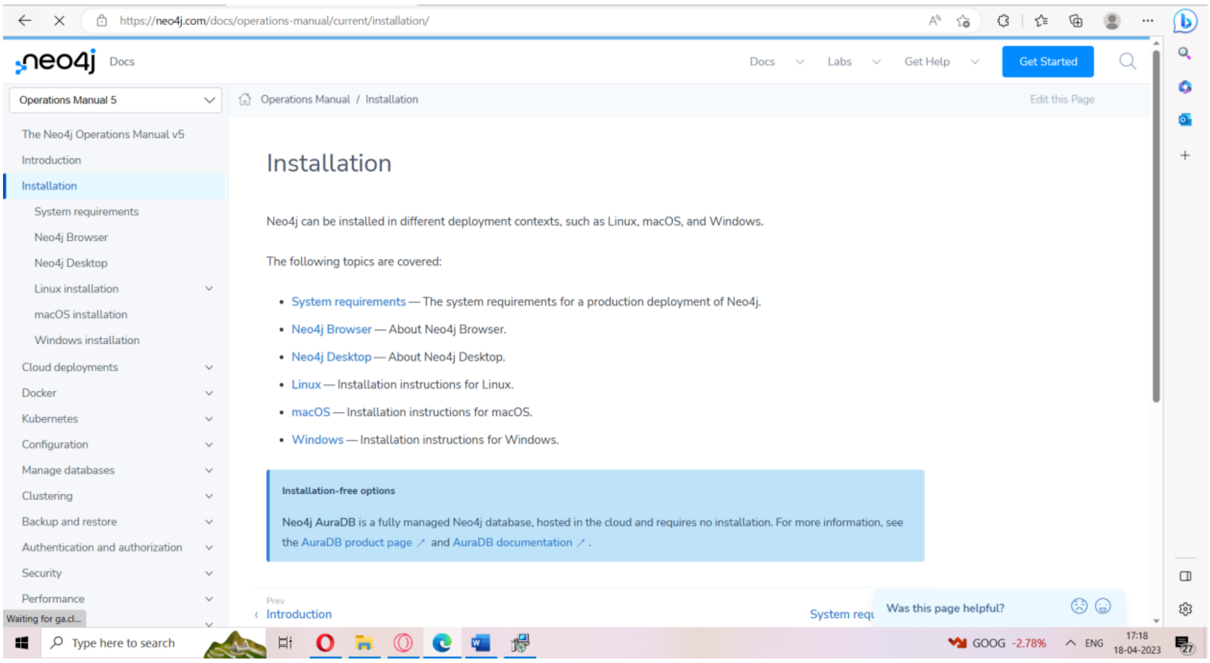


Java(TM) SE Development Kit 11.0.17 (64-bit) - Complete

Java(TM) SE Development Kit 11.0.17 (64-bit) Successfully Installed

Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

Downloaded neo4j community server edition



https://neo4j.com/docs/operations-manual/current/installation/

neo4j Docs — Docs — Labs — Get Help — Get Started

Operations Manual 5

- The Neo4j Operations Manual v5
- Introduction
- Installation
  - System requirements
  - Neo4j Browser
  - Neo4j Desktop
  - Linux installation
  - macOS installation
  - Windows installation
- Cloud deployments
- Docker
- Kubernetes
- Configuration
- Manage databases
- Clustering
- Backup and restore
- Authentication and authorization
- Security
- Performance

Operations Manual / Installation — Edit this Page

Installation

Neo4j can be installed in different deployment contexts, such as Linux, macOS, and Windows.

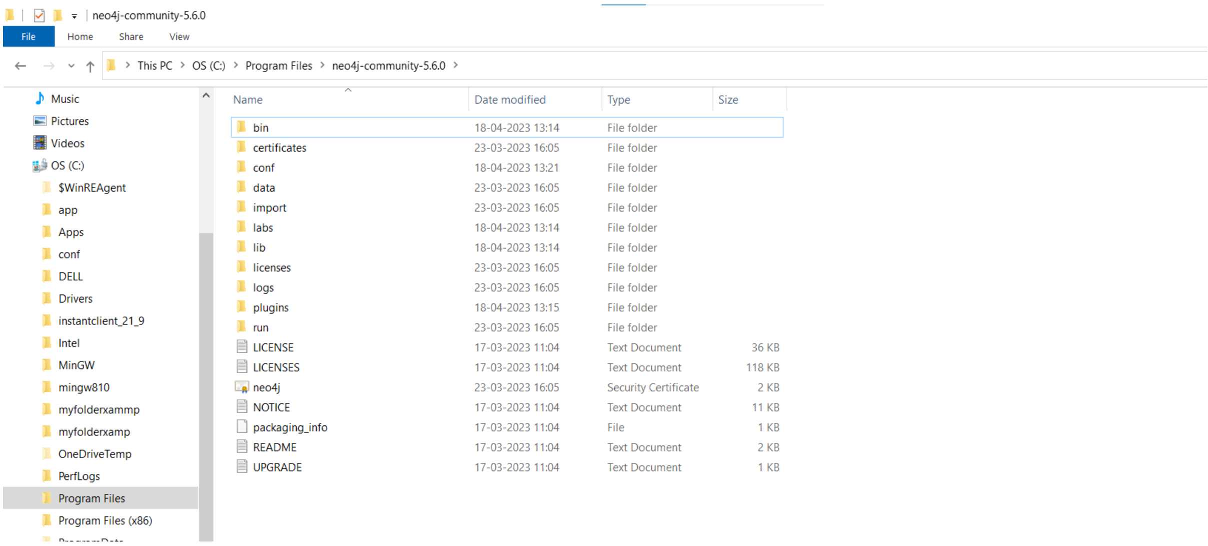
The following topics are covered:

- System requirements — The system requirements for a production deployment of Neo4j.
- Neo4j Browser — About Neo4j Browser.
- Neo4j Desktop — About Neo4j Desktop.
- Linux — Installation instructions for Linux.
- macOS — Installation instructions for macOS.
- Windows — Installation instructions for Windows.

Installation-free options

Neo4j AuraDB is a fully managed Neo4j database, hosted in the cloud and requires no installation. For more information, see the AuraDB product page ↗ and AuraDB documentation ↗ .

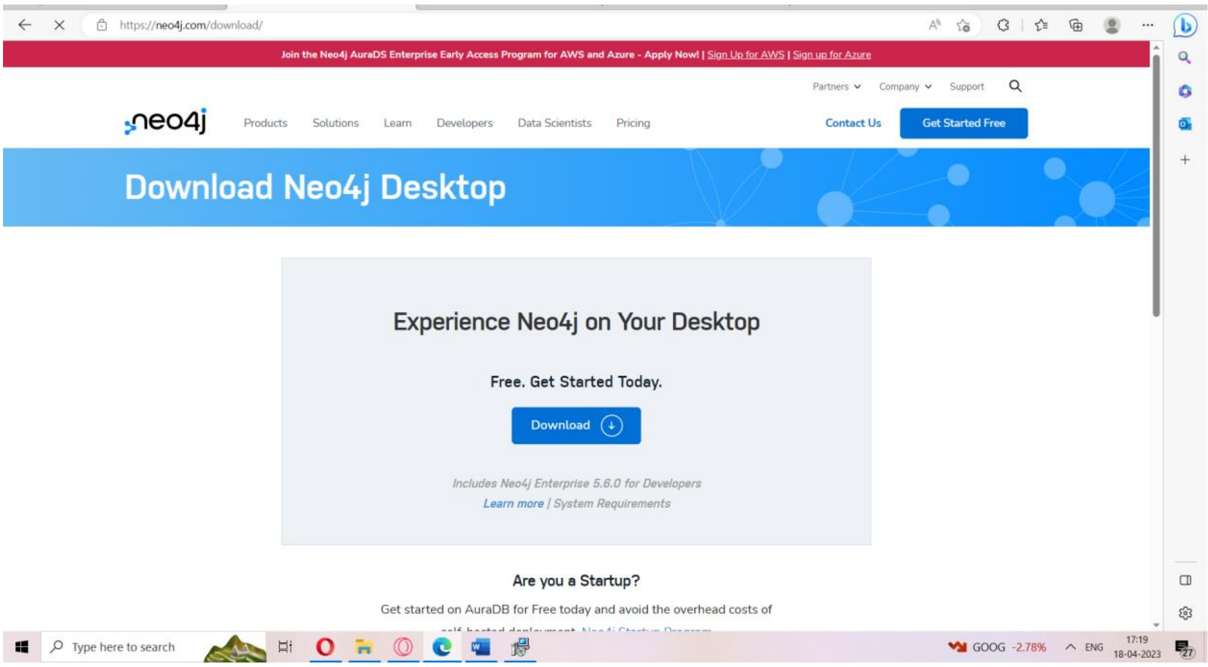
Prev Introduction — System requ — Was this page helpful?



neo4j-community-5.6.0

Name	Date modified	Type	Size
bin	18-04-2023 13:14	File folder	
certificates	23-03-2023 16:05	File folder	
conf	18-04-2023 13:21	File folder	
data	23-03-2023 16:05	File folder	
import	23-03-2023 16:05	File folder	
labs	18-04-2023 13:14	File folder	
lib	18-04-2023 13:14	File folder	
licenses	23-03-2023 16:05	File folder	
logs	23-03-2023 16:05	File folder	
plugins	18-04-2023 13:15	File folder	
run	23-03-2023 16:05	File folder	
LICENSE	17-03-2023 11:04	Text Document	36 KB
LICENSES	17-03-2023 11:04	Text Document	118 KB
neo4j	23-03-2023 16:05	Security Certificate	2 KB
NOTICE	17-03-2023 11:04	Text Document	11 KB
packaging_info	17-03-2023 11:04	File	1 KB
README	17-03-2023 11:04	Text Document	2 KB
UPGRADE	17-03-2023 11:04	Text Document	1 KB

Downloaded neo4j Desktop



Join the Neo4j AuraDS Enterprise Early Access Program for AWS and Azure - Apply Now! | Sign Up for AWS | Sign up for Azure

Download Neo4j Desktop

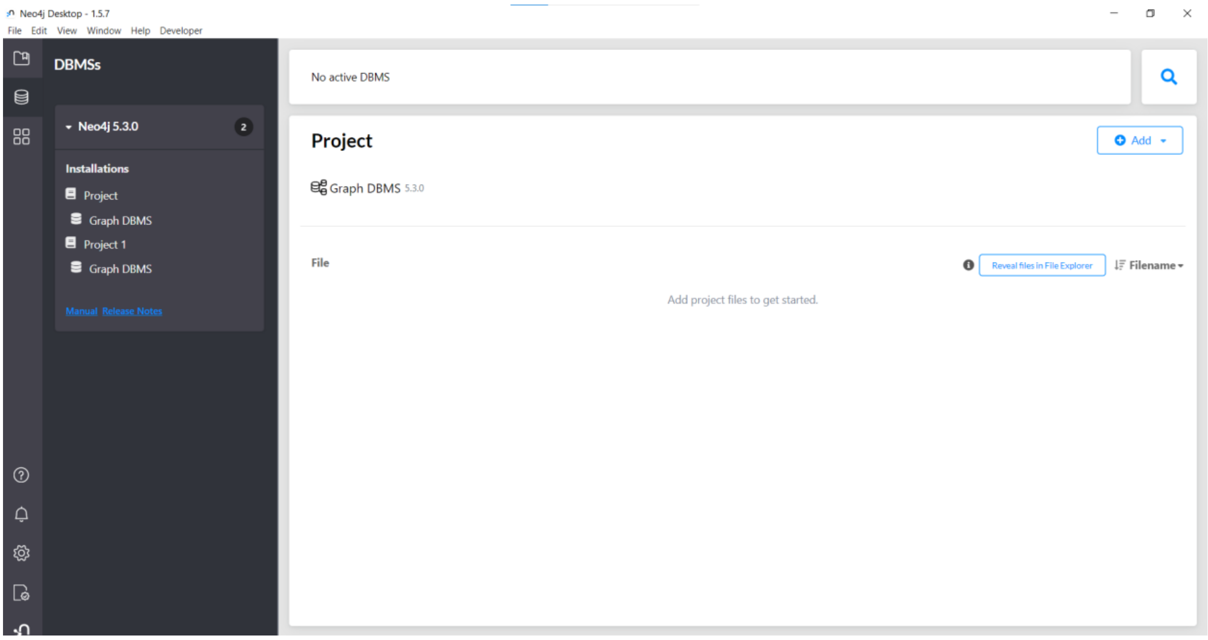
Experience Neo4j on Your Desktop

Free. Get Started Today.

Includes Neo4j Enterprise 5.6.0 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



No active DBMS

Project

Graph DBMS 5.3.0

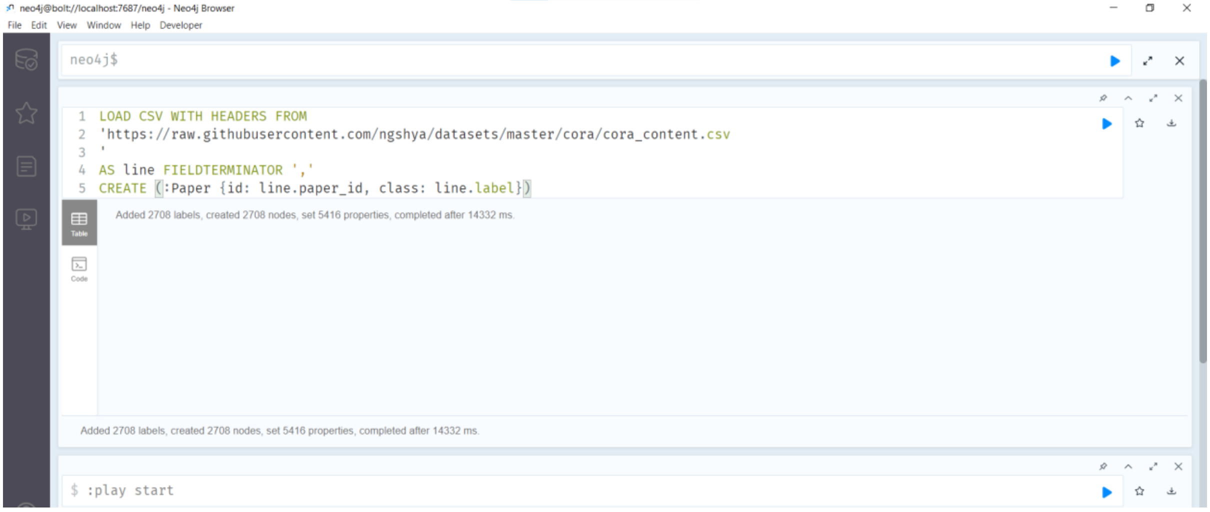
File

Add project files to get started.

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})
```

The above Cypher query loads data from a CSV file hosted on the specified URL and creates a Paper node for each row in the CSV file.


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
Table
Code

```
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})
```

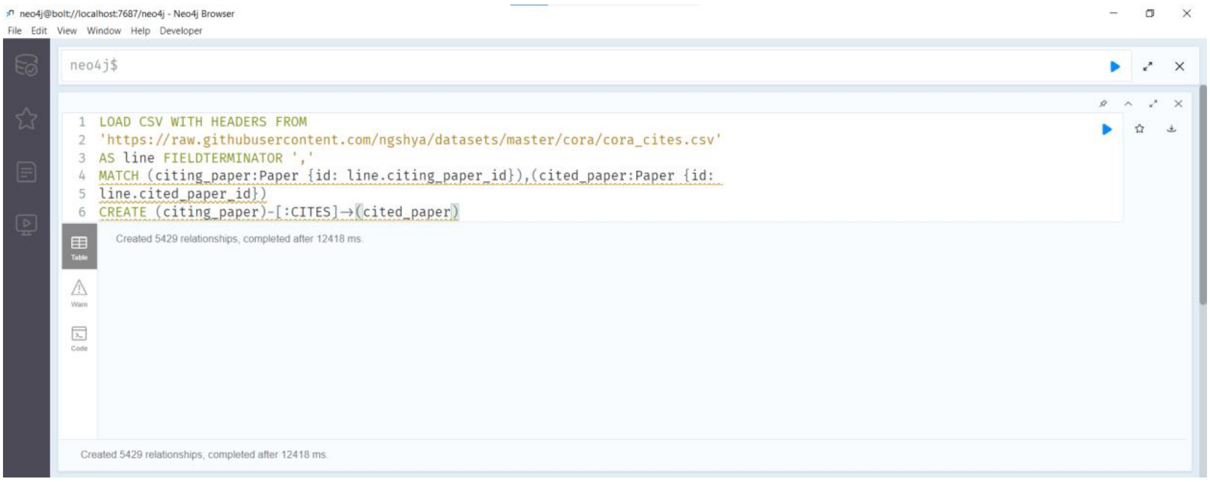
```
Added 2708 labels, created 2708 nodes, set 5416 properties, completed after 14332 ms.
```

```
Added 2708 labels, created 2708 nodes, set 5416 properties, completed after 14332 ms.
```

```
$ :play start
```

```
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)
```

The above Cypher query loads data from a CSV file hosted on the specified URL and creates a CITES relationship between Paper nodes based on the data in the CSV file.


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$
Table
Warn
Code

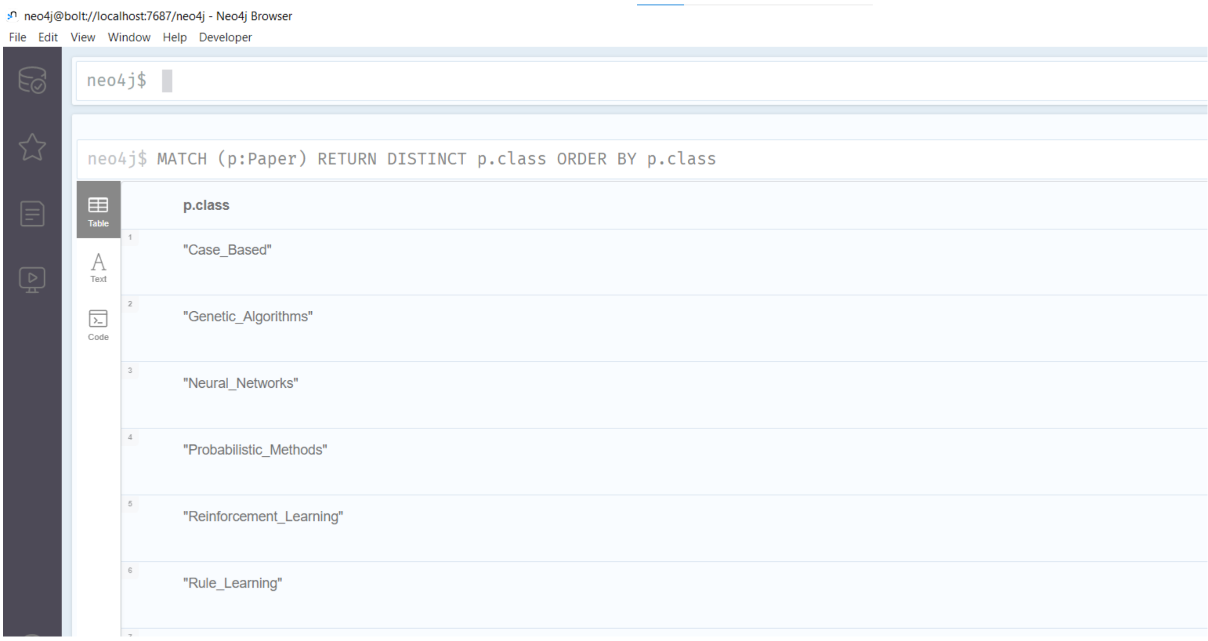
```
LOAD CSV WITH HEADERS FROM
'https://raw.githubusercontent.com/ngshya/datasets/master/cora/cora_cites.csv'
AS line FIELDTERMINATOR ','
MATCH (citing_paper:Paper {id: line.citing_paper_id}),(cited_paper:Paper {id:
line.cited_paper_id})
CREATE (citing_paper)-[:CITES]→(cited_paper)
```

```
Created 5429 relationships, completed after 12418 ms.
```

```
Created 5429 relationships, completed after 12418 ms.
```

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

The above Cypher query retrieves all distinct classes of Paper nodes in the database and sorts them in ascending order.



neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Table
Text
Code

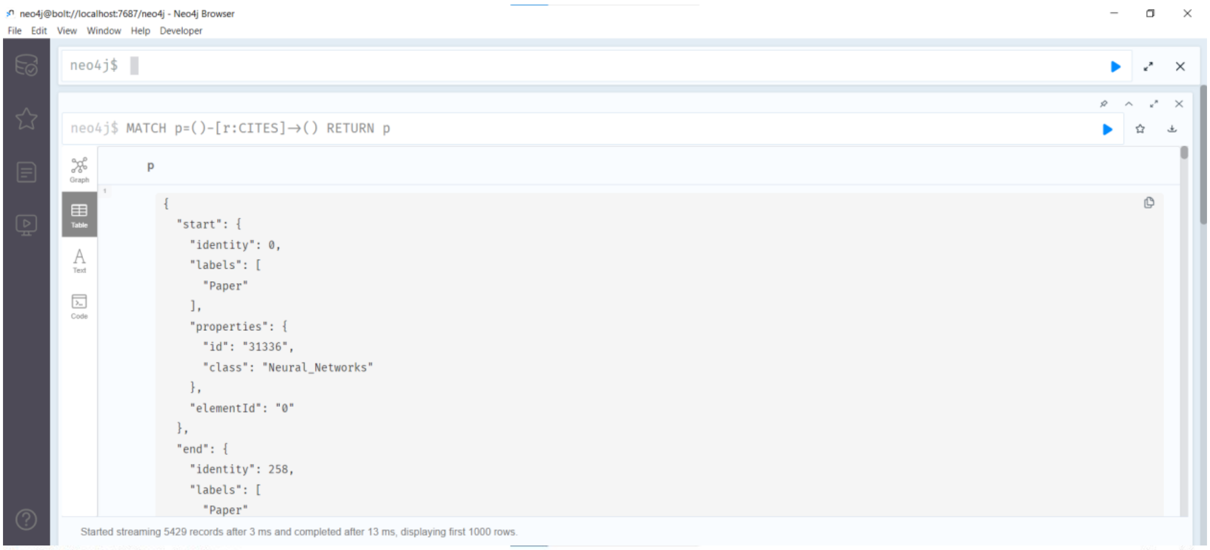
```
neo4j$
```

```
neo4j$ MATCH (p:Paper) RETURN DISTINCT p.class ORDER BY p.class
```

	p.class
1	"Case_Based"
2	"Genetic_Algorithms"
3	"Neural_Networks"
4	"Probabilistic_Methods"
5	"Reinforcement_Learning"
6	"Rule_Learning"

MATCH p=()-[r:CITES]->() RETURN p

The above Cypher query retrieves all CITES relationships in the database and returns them as paths (p)

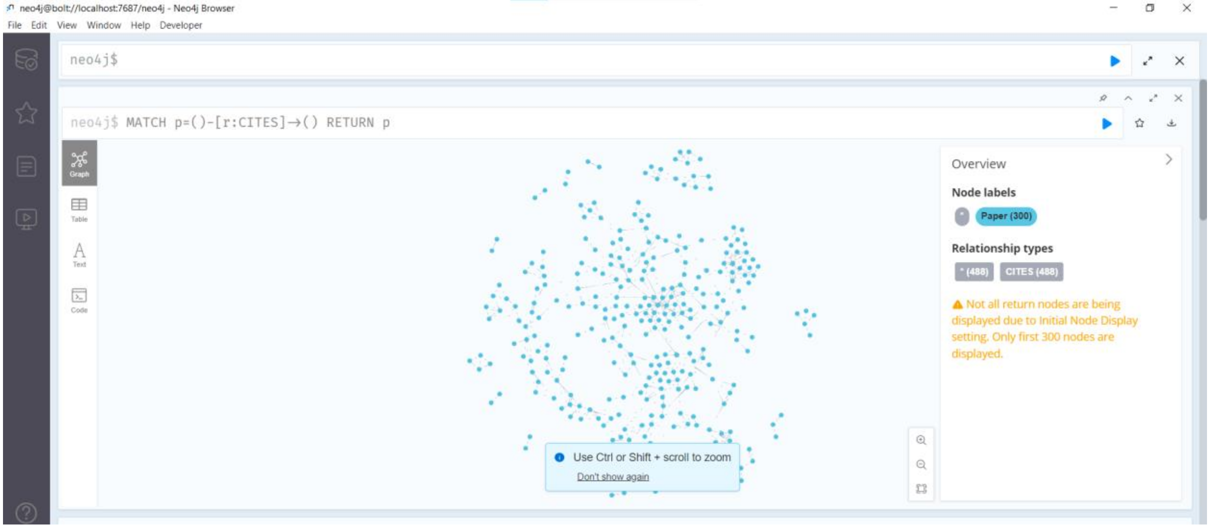

neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Graph
Table
Text
Code
p

```
neo4j$
```

```
neo4j$ MATCH p=()-[r:CITES]→() RETURN p
```

```
{
  "start": {
    "identity": 0,
    "labels": [
      "Paper"
    ],
    "properties": {
      "id": "31336",
      "class": "Neural_Networks"
    },
    "elementId": "0"
  },
  "end": {
    "identity": 258,
    "labels": [
      "Paper"
```

Started streaming 5429 records after 3 ms and completed after 13 ms, displaying first 1000 rows.



```
neo4j$
```

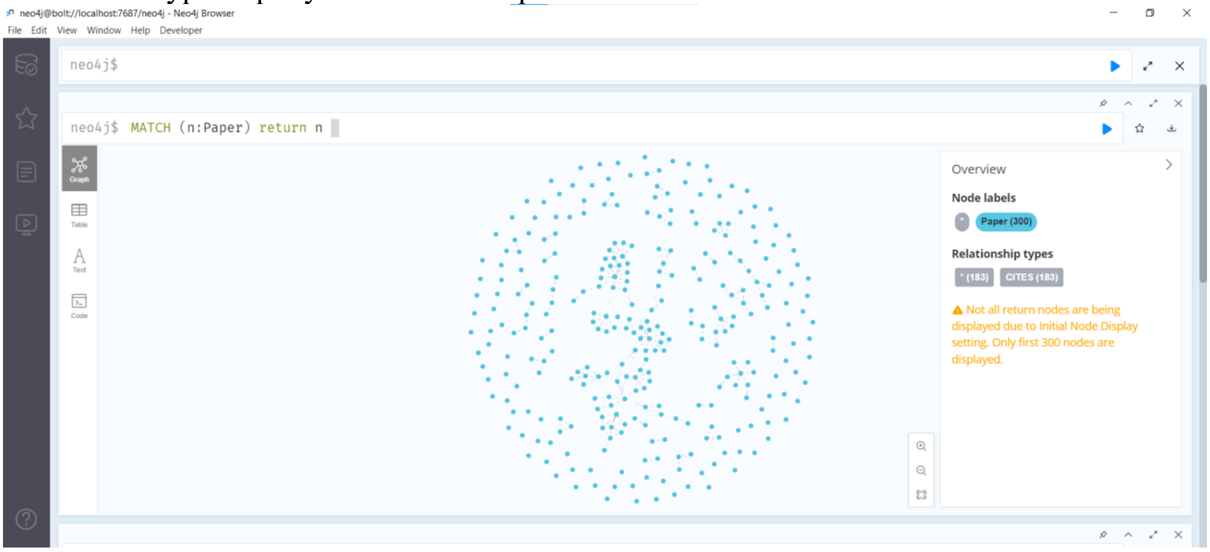
```
neo4j$ MATCH p=()-[r:CITES]→() RETURN p
```

⚠ Not all return nodes are being displayed due to Initial Node Display setting. Only first 300 nodes are displayed.

Use Ctrl or Shift + scroll to zoom

MATCH (n:Paper) return n

The above Cypher query retrieves all Paper nodes in the database and returns them.



neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Graph
Table
Text
Code

```
neo4j$
neo4j$ MATCH (n:Paper) return n
```

Overview

Node labels

* Paper (300)

Relationship types

* (183) CITES (183)

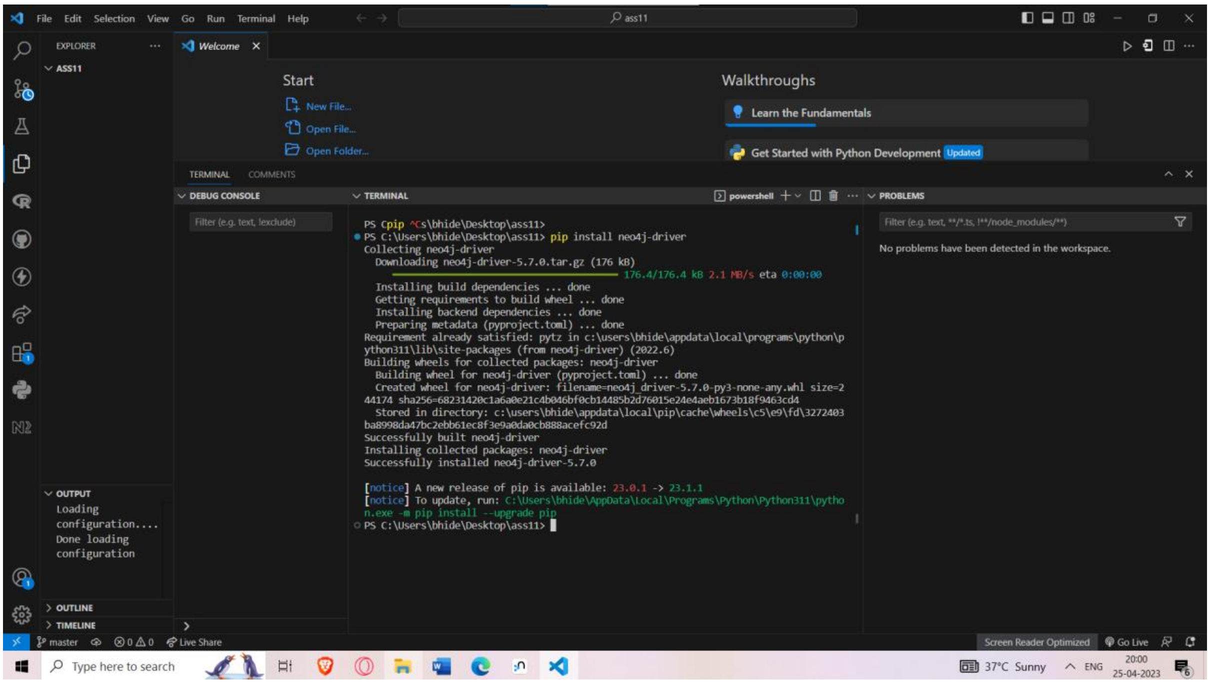
⚠ Not all return nodes are being displayed due to Initial Node Display setting. Only first 300 nodes are displayed.

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

a) Does paper A cite paper B? If not directly, does paper A cite a paper which in its turn cites paper B? And so on, in several levels.

b) Show the full classification of a paper (for example, Databases / Relational)

Installed neo4j-driver to run python application



```
PS Cpip ^Cs\bhide\Desktop\ass11>
PS C:\Users\bhide\Desktop\ass11> pip install neo4j-driver
Collecting neo4j-driver
  Downloading neo4j-driver-5.7.0.tar.gz (176 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 176.4/176.4 kB 2.1 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: pytz in c:\users\bhide\appdata\local\programs\python\p
ython311\lib\site-packages (from neo4j-driver) (2022.6)
Building wheels for collected packages: neo4j-driver
  Building wheel for neo4j-driver (pyproject.toml) ... done
  Created wheel for neo4j-driver: filename=neo4j_driver-5.7.0-py3-none-any.whl size=2
44174 sha256=68231420c1a6a0e21c4b046bf0cb14485b2d76015e24e4aeb1673b18f9463cd4
  Stored in directory: c:\users\bhide\appdata\local\pip\cache\wheels\c5\e9\fd\3272403
ba8998da47bc2ebb61ec8f3e9a0da0cb888acefc92d
Successfully built neo4j-driver
Installing collected packages: neo4j-driver
Successfully installed neo4j-driver-5.7.0

[notice] A new release of pip is available: 23.0.1 -> 23.1.1
[notice] To update, run: C:\Users\bhide\AppData\Local\Programs\Python\Python311\pytho
n.exe -m pip install --upgrade pip
PS C:\Users\bhide\Desktop\ass11> 
```

No problems have been detected in the workspace.

```
Loading
configuration....
Done loading
configuration
```

Python Desktop Application:

Code :

```
import sys
import os
import tkinter as tk
from tkinter import *
import tkinter.messagebox

# For Neo4j Connection
from neo4j import GraphDatabase
class Neo4jConnection:

    def __init__(self, uri, user, pwd):
        self.__uri = uri
        self.__user = user
        self.__pwd = pwd
        self.__driver = None
        try:
            self.__driver = GraphDatabase.driver(self.__uri, auth=(self.__user, self.__pwd))
        except Exception as e:
            print("Failed to create the driver:", e)

    def close(self):
        if self.__driver is not None:
            self.__driver.close()

    def query(self, query, db=None):
        assert self.__driver is not None, "Driver not initialized!"
        session = None
        response = None
        try:
            session = self.__driver.session(database=db) if db is not None else
self.__driver.session()
            response = list(session.run(query))
        except Exception as e:
            print("Query failed:", e)
        finally:
            if session is not None:
                session.close()
        return response
conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="newpass123")
# ^ Neo4j Connected


window = tk.Tk()
window.title("Neo4j Desktop App")
window.geometry("700x500")
window.configure(bg="grey")
```

```
blog=tk.StringVar()
blog_title=tk.StringVar()
direct_id1=tk.StringVar()
direct_id2=tk.StringVar()
recur_id1=tk.StringVar()
recur_id2=tk.StringVar()

#submitting query
def submit():
   query_string = blog_title.get()
   result = conn.query(query_string, db='neo4j')
   result_label.config(text=result)  # Update the label text with the query result
   blog.set("")  # Clear the blog_title entry widget



def direct_check():
   id1=direct_id1.get()
   id2=direct_id2.get()
   query_string = '''MATCH p=(:Paper{id:"'''+id1+'''"})-[r:CITES]->(:Paper{id:"'''+id2+'''"})
RETURN p'''
   result = conn.query(query_string, db='neo4j')
   if(result):
      Label(window,text="YES", fg="blue",font=("Arial", 15),width=37).grid(row=160)
   else:
      Label(window,text="NO", fg="RED",font=("Arial", 15),width=37).grid(row=160)
   blog.set("")

def indirect_check():
   id1=recur_id1.get()
   id2=recur_id2.get()
   query_string = '''MATCH p=(:Paper{id:"'''+id1+'''"})-[r:CITES]->() MATCH
q=(:Paper{id:"'''+id2+'''"}) RETURN q'''
   result = conn.query(query_string, db='neo4j')
   if(result):
      Label(window,text="YES", fg="blue",font=("Arial", 15),width=37).grid(row=220)
   else:
      Label(window,text="NO", fg="RED",font=("Arial", 15),width=37).grid(row=220)
   blog.set("")

#tkinter window
title_label = tk.Label(window,text="Neo4j Python Desktop Application",
fg="black",font=("Arial", 25, 'bold'),width=37)
title_label.grid(row=0,column=0, pady=10)

name_label = tk.Label(window, text='Query', font=('calibre',10, 'bold'))
name_label.grid(row=70, pady=10)

name_entry = tk.Entry(window, textvariable=blog_title, font=('calibre',10,'normal'),
width=70)
name_entry.grid(row=80, pady=5)
```

```
sub_btn = tk.Button(window, text='Run Query', command=submit)
sub_btn.grid(row=110, pady=10)

result_label = tk.Label(window, text='', font=('calibre', 12, 'normal'))
result_label.grid(row=90, pady=20)


name_label = tk.Label(window, text='Does Paper with id1 cite id2 directly?',
font=('calibre',10,'bold')).grid(row=120)
name_entry1 = tk.Entry(window, textvariable=direct_id1, font=('calibre',10,'normal'))
name_entry1.grid(row=130, pady=5)
name_entry2 = tk.Entry(window, textvariable=direct_id2, font=('calibre',10,'normal'))
name_entry2.grid(row=140, pady=5)
sub_btn = tk.Button(window, text='Check', command=direct_check).grid(row=150, pady=10)


name_label = tk.Label(window, text='Does Paper with id1 cites id2 indirectly?',
font=('calibre',10,'bold')).grid(row=180)
name_entry1 = tk.Entry(window, textvariable=recur_id1, font=('calibre',10,'normal'))
name_entry1.grid(row=190, pady=5)
name_entry2 = tk.Entry(window, textvariable=recur_id2, font=('calibre',10,'normal'))
name_entry2.grid(row=200, pady=5)
sub_btn = tk.Button(window, text='Check', command=indirect_check).grid(row=210,
pady=10)


window.mainloop()
```

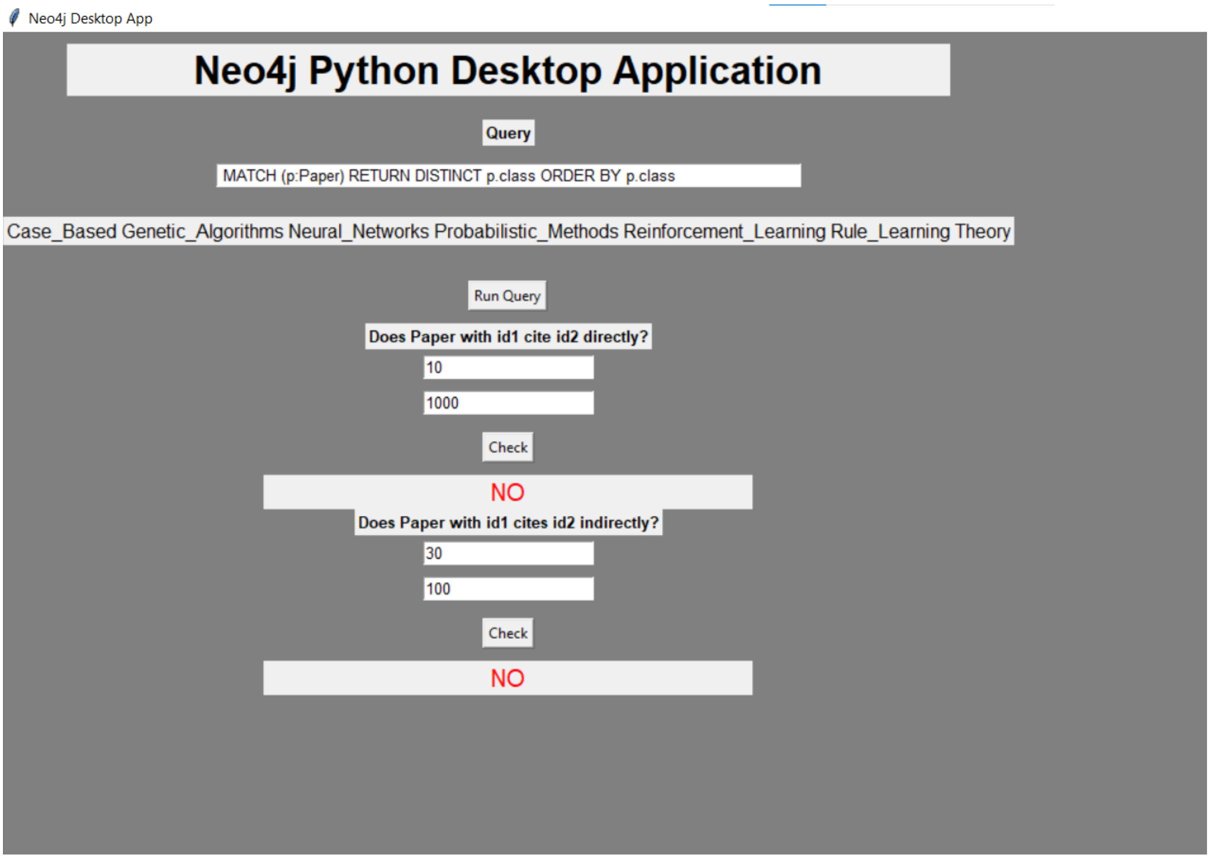
This is a Python desktop application that provides a GUI for interacting with a Neo4j database. It allows the user to submit queries to the database and retrieve results. The application has two functions:

direct_check(): This function checks whether a paper with a given ID (direct_id1) directly cites another paper with a given ID (direct_id2). If the citation exists, it displays "YES" in blue, and if not, it displays "NO" in red.

indirect_check(): This function checks whether a paper with a given ID (recur_id1) cites another paper with a given ID (recur_id2) indirectly. If the citation exists, it displays "YES" in blue, and if not, it displays "NO" in red.

The submit() function is called when the user clicks the "Run Query" button, and it executes the query entered by the user (blog_title) and displays the result in a label (result_label).

The application uses the tkinter library to create the GUI and the neo4j library to connect to the Neo4j database.



Neo4j Desktop App

Neo4j Python Desktop Application

Query

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

```
Case_Based Genetic_Algorithms Neural_Networks Probabilistic_Methods Reinforcement_Learning Rule_Learning Theory
```

Does Paper with id1 cite id2 directly?

10

1000

NO

Does Paper with id1 cites id2 indirectly?

30

100

NO