Installing Neo4j

Table of Contents

[**Installing Neo4j** 3](#_Toc457319609)

[Installing Neo4j on Windows… 3](#_Toc457319610)

[Installing Neo4j on a Mac… 6](#_Toc457319611)

[Setting a Database Password and Testing out Your Installation 8](#_Toc457319612)

[**Getting Started With Neo4j** 10](#_Toc457319613)

[Cypher 10](#_Toc457319614)

[Loading Data Directly into the Console 11](#_Toc457319615)

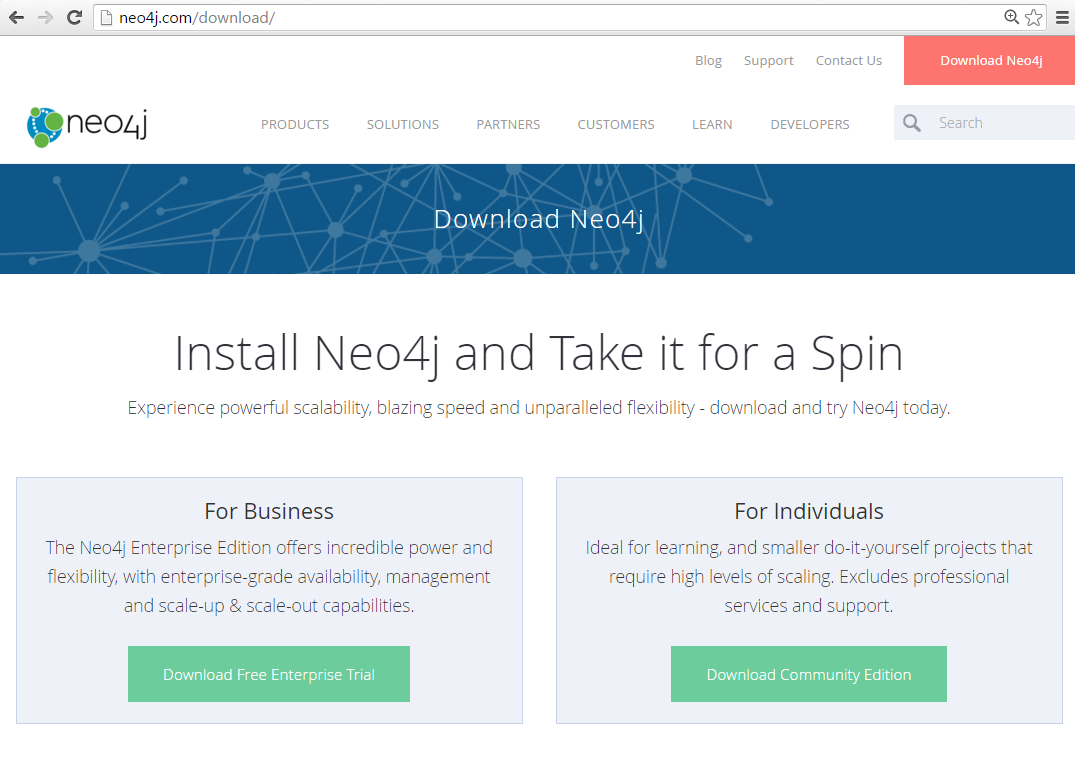
[Visualization with the Console 13](#_Toc457319616)

[Appendix: Deleting Data 13](#_Toc457319617)

# **Installing Neo4j**

1. Go to the website <http://neo4j.com/> and download Neo4j for your operating system.

The Download Neo4j link is in the upper right hand corner: https://neo4j.com/download/

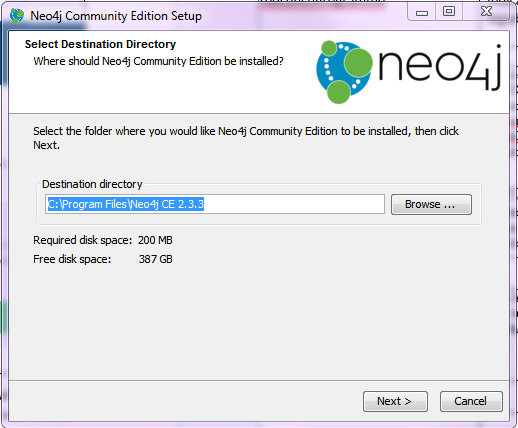


Once you are on this page, click “Download Community Edition” under “For Individuals”, on the right. This will give you a free version of Neo4j that is perfect for learning about graph databases and getting you started with smaller projects. Your download should start automatically, and it will take you to a page with instructions on how to install Neo4j on your operating system.

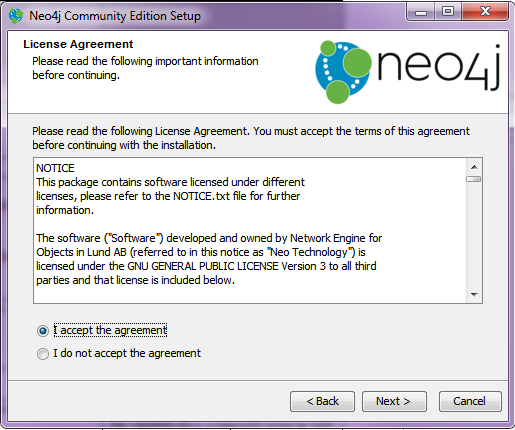
1. Run the Neo4j installer.

## Installing Neo4j on Windows…

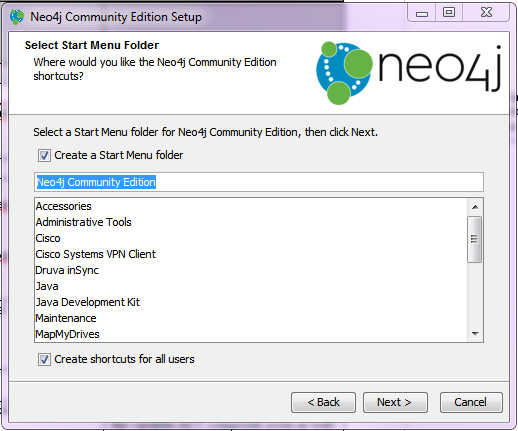
Once you have downloaded the installer .exe file, double-click it to run it.



Accept the default installation location, inside “Program Files”.

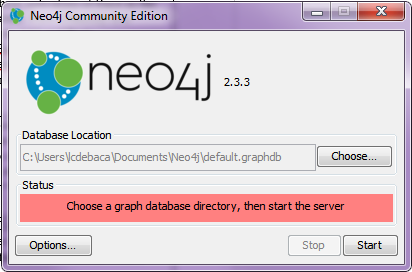


Accept the terms.



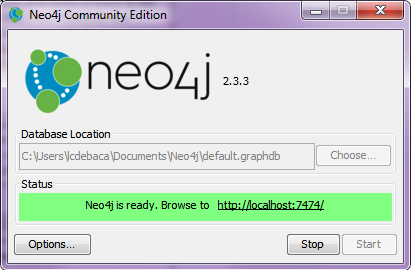
Accept the defaults.

Once Neo4j is installed, it will open up a program you can use to control the server on your computer.



To start, click the “Start” button to accept the default database location and start a server that will read from and write to the default database.

Once the server has started, the command application will look like this:

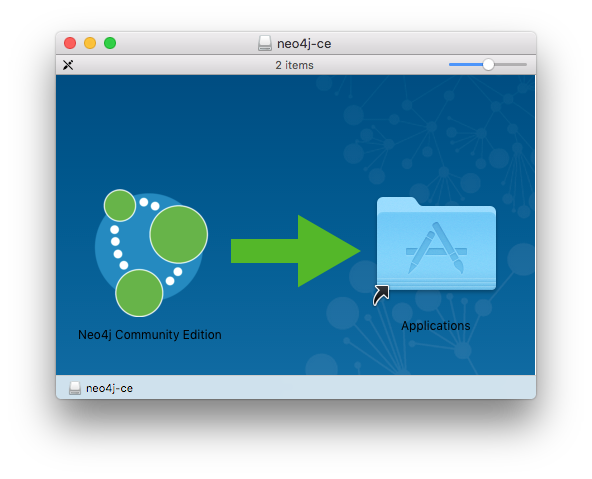


## Installing Neo4j on a Mac…

*The following is adapted from the page that loads when you download Neo4j for Mac (the exact link changes from version to version – the latest at the time of writing was https://neo4j.com/download-thanks/?edition=community&flavour=osx&release=3.0.3).*

To start, open the “.dmg” disk image file you just downloaded. It will be named something like “neo4j-community\_macos\_3\_0\_3.dmg”.

Opening the “.dmg” file will result in a Finder window that looks like this:



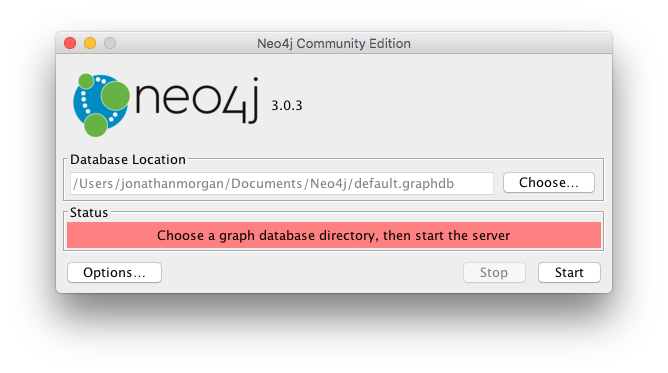
To install Neo4j, click on and drag the “Neo4j Community Edition” icon on the left and drop it onto the “Applications” folder on the right.

Close this Finder window (and unmount the disk image).

Then, open your Applications folder and run the Application named “Neo4j Community Edition”.

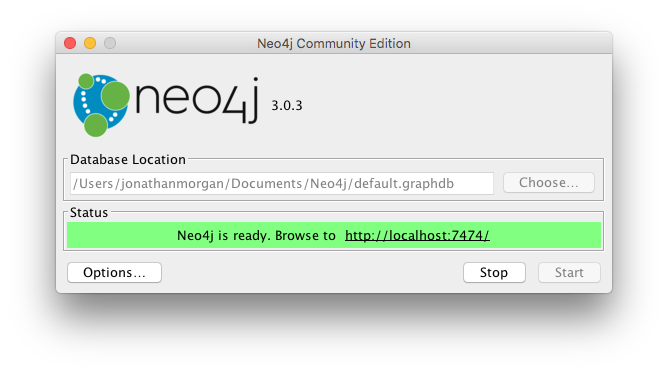
You might need to click “Open” to acknowledge that you downloaded the Application from the Internet and you are sure you want to run it.

This will cause the same control program used to control Neo4j on Windows to load, allowing you to start and stop the Neo4j database server, change the database file it is using, and adjust options. When you first open the control program, Neo4j will not be running:



To start, click the “Start” button to accept the default database location and start a server that will read from and write to the default database.

Once the server has started, the command application will look like this:



## Setting a Database Password and Testing out Your Installation

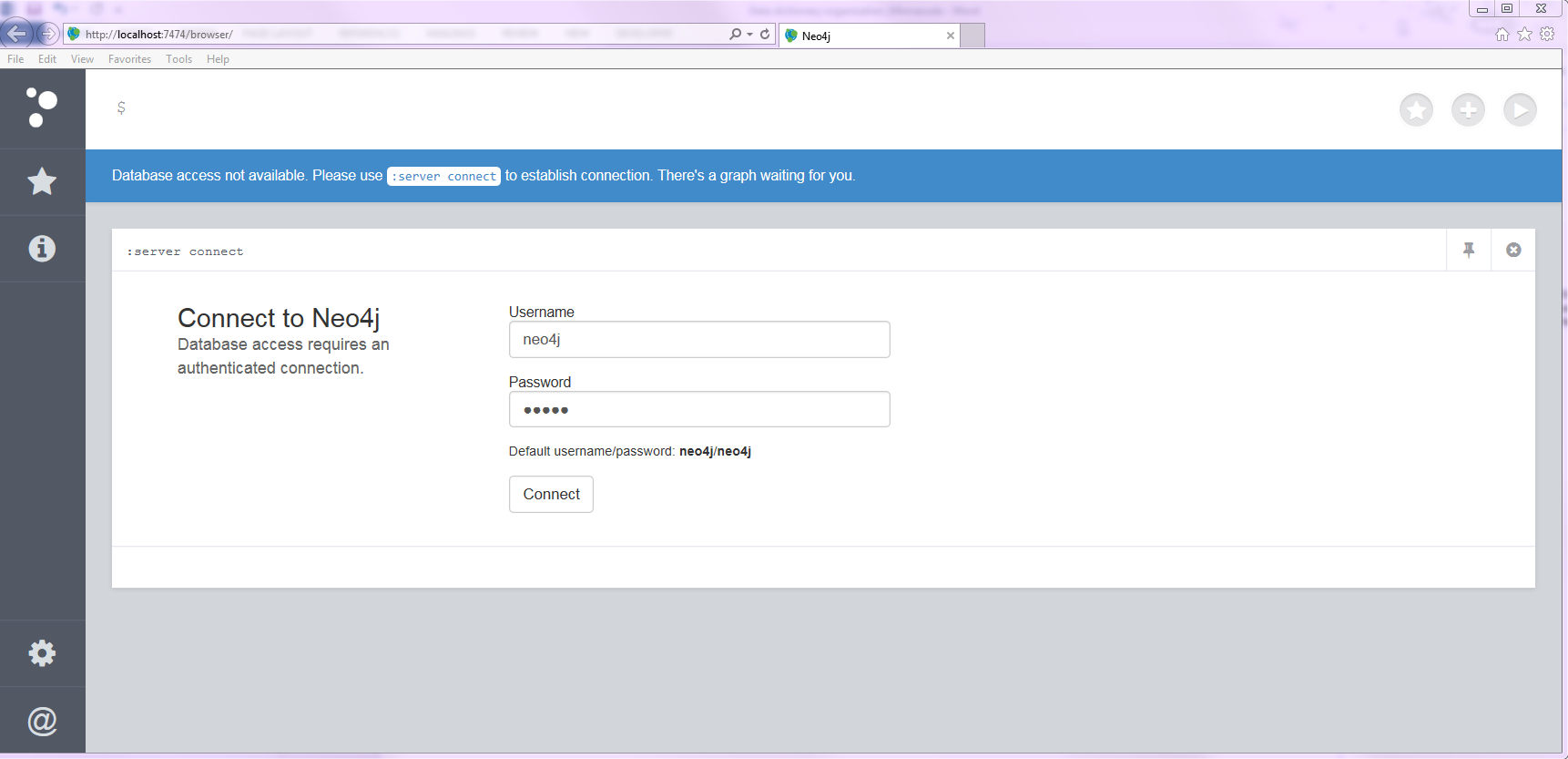
***A quick side note about servers:***  In the simplest terms, servers are pieces of software that are designed to provide information to other programs, but not necessarily designed for easy use by people. You have probably heard the term server when talking about internet sites or web sites. When you look at a web page, instructions on what to display in your browser window are sent to your computer over the Internet by a web server. The server contains all the code for a given web site and sends sections of the site (pages) to you as you request them. Your computer doesn’t hold all that information at one time. It only receives the pieces or pages of a web site that it requests (based on you clicking on links and so asking for a new part of the site).

In the context of a web site, your browser acts as a “client” to the server. A “client” is a program that knows how to interact with a server and then make use of the data it receives. Your web browser receives information in a format called HTML, then renders it into the viewable, interactive format you see when you view a web page.

Your browser also can serve as a client for more complicated servers that “speak” HTML. For example, when you run a Jupyter IPython notebook in a browser window, your browser is a client for a Python server running on your local machine. Jupyter IPython notebooks allow you to enter Python code in the browser, send that code to the Python server to be run, and then view the results in the same browser window.

The Neo4j database you just installed is also a server, and while it is designed to be used with programs written in languages like Python, Neo4j also provides a client that can be run in a web browser.

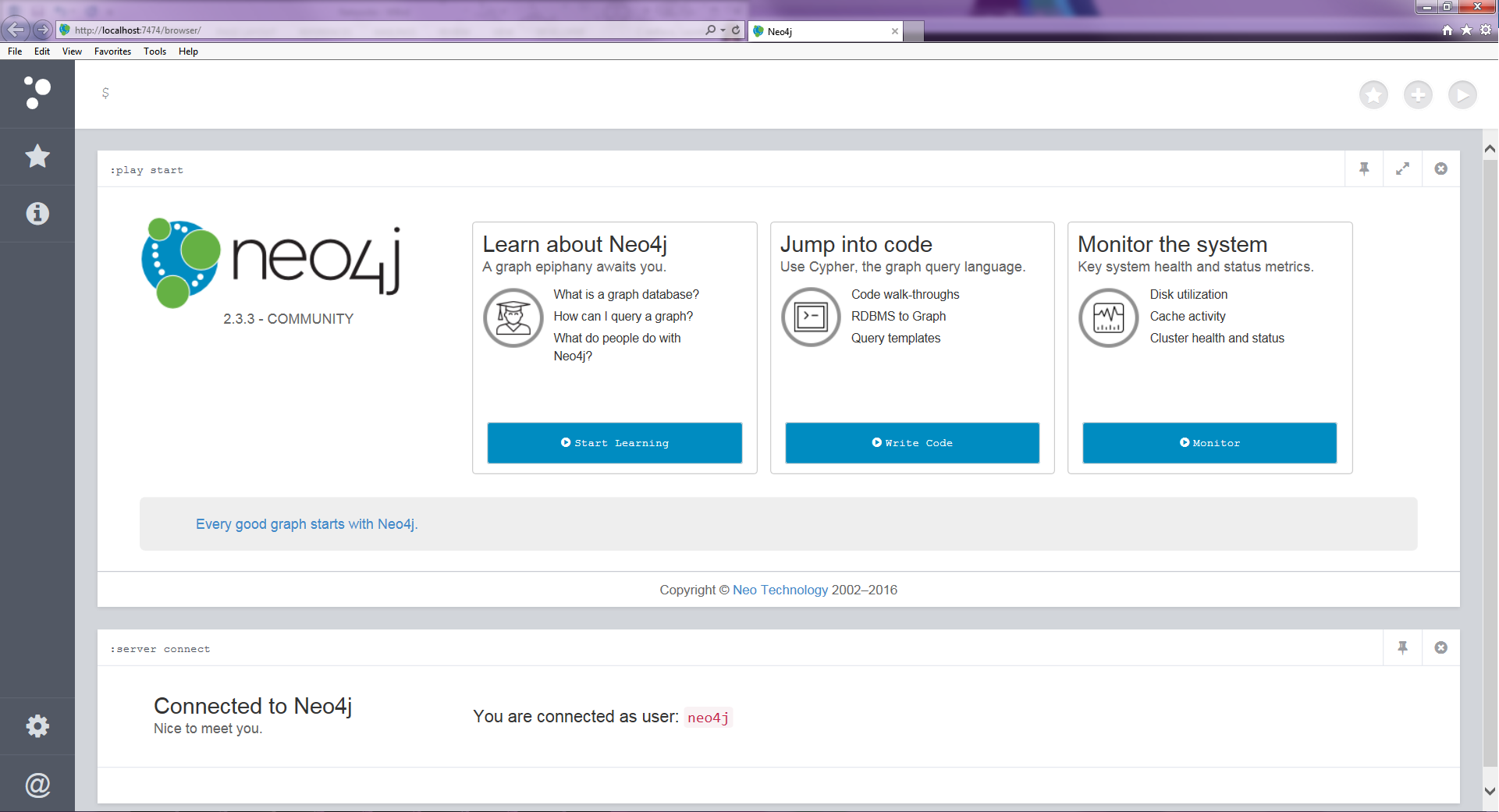
We will use this client to run a few basic tests to verify that our neo4j server was installed correctly and is working. To access the Neo4j web client, copy the link <http://localhost:7474/> into your browser.



You will first see a login screen for the default database. Enter “neo4j” as the password on this screen.

You will then be asked to set a new password. Set the password, and make sure to store it somewhere safe, else you will be unable to subsequently access Neo4j, and you’ll have to either switch to a new database file or re-install.

Once you set a new password, if you see the following page, Neo4j is installed and configured correctly.



Congratulations you are connected to neo4j!

# **Getting Started With Neo4j**

Before we start using python to interact with Neo4j, it is a good idea to get accustomed to what the Neo4j console has to offer.

The Neo4j console can be used to pass commands to the database to create and visualize data. It also has tools for managing the database and learning to use neo4j. If you are interested in learning more about Neo4j, make sure to take a look at the tutorials that Neo4j makes available through the console.

## Cypher

Neo4j uses the Cypher Query language to communicate with the database. If you are familiar with SQL commands, cypher should not look too much different. For the exercises in this chapter all the cypher commands will be provided, but here is a super quick introduction to the basics of cypher in case you need to write your own.

Cypher represents relationships between nodes. Names of nodes are enclosed in parentheses, and relationships are represented by arrows made up of hyphens and angle brackets (“-->” and “<--”). For instance the symbol **(n)** represents a node named n. (note, n is just a variable name, it can be called anything). The symbol **-->** or **<--** representrelationships. Putting it together the notation **(a) --> (b)** would denote the node **a** has a relationship with node **b**.

This relationship notation is used in conjunction with cypher commands to allow you to interact with network data in Neo4j. The most common cypher commands are listed below:

* MATCH
  + Essentially the same thing as SELECT in SQL – lets you choose sets of nodes and relationships that match criteria you specify.
* CREATE
  + Allows you to create a node or a relationship.
* MERGE
  + Allows you to reference a node or relationship and have the entity be automatically created if a matching node or relationship does not already exist.
* RETURN
  + Used to tell Neo4j what information you want returned. Needed in every query.
* LIMIT
  + Lets you limit the number of results that are returned (just like the LIMIT keyword in SQL).

## Loading Data Directly into the Console

The Data we will be using in the notebooks is from the fictional university, Roke College. Roke College is a research institution where the professors apply to grants to fund their research. The data set we will be using shows the relationship between these awards and the people who work on them.

Currently the data for this workbook is stored in the network workbook’s folder in three CSV files: employee\_data.csv, student\_data.csv, and award\_data.csv. Download these data files to your computer, storing them in a location that you’ll be able to find as you work through the notebooks.

The Employee and Student Data look like this,



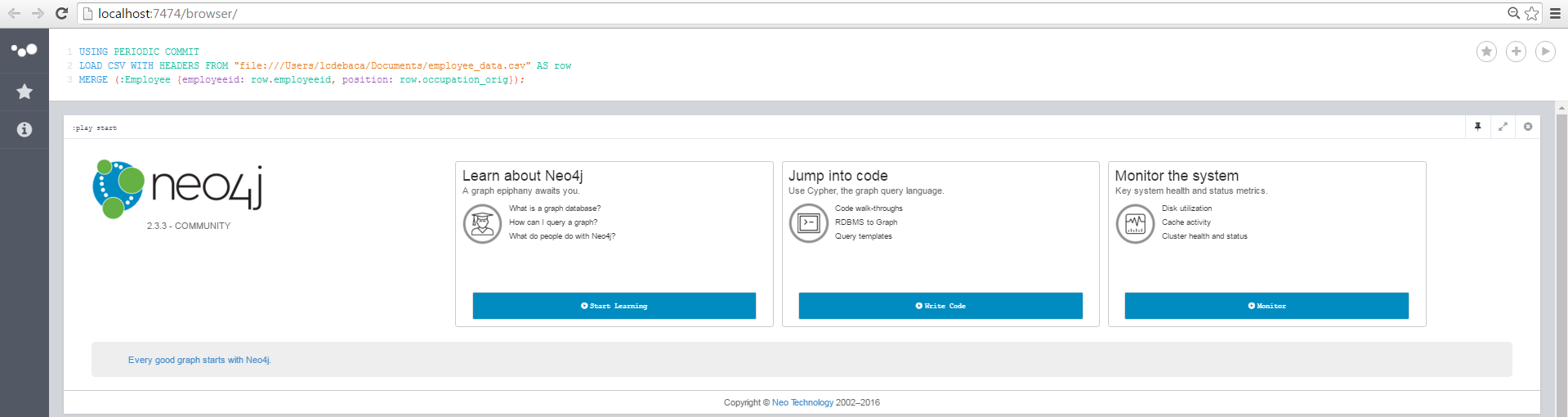
Where each Facility member and student have an employee id and an occupation identifier (FACILTY or STUDENT). Each student and facility member appears only once in their respective lists.

The Awards data looks like this,



Where the employeeid of the Student or facility member who worked on an award appears next to the award number for that award. An award and a person can appear multiple times in this file.

To load the data type the following commands in the top portion of the console (the box highlighted below),



Commands

1. Load in the Employee Nodes from the employee\_data.csv. *Note: in the FROM clauses below you will have to replace “<PATH TO DATA>” with the actual path to the location on your machine where you saved the CSV Files. You should also avoid spaces in any parts of the path to files you want to import.*

The following cypher LOAD CSV command will create a node for each row in the data and give the node the properties of employeeid and position based on the data in that row.

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:<PATH TO DATA>/employee\_data.csv" AS row

MERGE (:Employee {employeeid: row.employeeid, position: row.occupation\_orig});

Windows example:

On Windows, you should structure the file path as the full, absolute path from the root of your drive to the data file. The pattern is “file:<drive\_letter>:/path/to/data.csv”, where <drive\_letter> is the letter of the drive on which the file is stored (your boot drive is usually C:, for example). So an example where the file is in “C:\work\employee\_data.csv”:

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:c:/work/employee\_data.csv" AS row

MERGE (:Employee {employeeid: row.employeeid, position: row.occupation\_orig});

Mac example:

On the mac, you’ll follow the pattern: <file:///path/to/data.csv>. By default, Neo4j on Mac expects you to make an “import” folder inside your data directory and place files you want to import there, and then the [file:///](NULL) reference is relative to the import folder. So for a user jonathanmorgan whose data directory is “/Users/jonathanmorgan/Documents/Neo4j/default.graphdb”, you’d create directory “/Users/jonathanmorgan/Documents/Neo4j/default.graphdb/import” if it doesn’t already exist, then place the file employee\_data.csv in the import directory: “/Users/jonathanmorgan/Documents/Neo4j/default.graphdb/import/employee\_data.csv”. Then in the resulting command you’d just reference the part of the path inside the “import” folder – In this case, “employee\_data.csv”:

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///employee\_data.csv" AS row

MERGE (:Employee {employeeid: row.employeeid, position: row.occupation\_orig});

For more information on importing data from CSV files,, see the Neo4j documentation on loading data from CSV files: <https://neo4j.com/developer/guide-import-csv/>

1. Load in the Award Nodes from the award\_data.csv. (Note, you will have to change the path of the csv to match the location of where you saved the data on your machine.) This commands will create a node for each row in the data if it does not already exist and gives the node the prosperity of award\_num based on the data in that row.

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:<PATH TO DATA>/award\_data.csv" AS row

MERGE (:Award {award\_num: row.uniqueawardnumber});

1. Create relationships between Facility members and the Awards they worked on. This command will go back through the award\_data.csv file and for each row, each worked on relationship, find the existing nodes that are part of that relationship, then formally create that relationship. These will be names WORKED\_ON relationships.

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:<PATH TO DATA>/award\_data.csv" AS row

MATCH (a:Award {award\_num: row.uniqueawardnumber})

MATCH (e:Employee {employeeid: row.employeeid})

MERGE (e)-[r:WORKED\_ON]->(a);

1. Create relationships between Facility members who worked on the same awards. This Command finds the pattern where two nodes have a WORKED\_ON relationship with the same award node. If a WORKED\_WITH relationship does not already exist, the command creates the relationship, WORKED\_WITH between those nodes.

MATCH (n1)-[:WORKED\_ON]->(a:Award)<-[:WORKED\_ON]-(n2)

MERGE (n1)-[r:WORKED\_WITH]-(n2);

At this point you have successfully create a neo4j database. We did not load the Student’s data in the above commands. This will be done later in the notebooks.

## Visualization with the Console

The Neo4j console is a convenient platform to visually explore data.

Try running the following commands to see a few of the patterns and relationships in your database.

* MATCH ()-[r:WORKED\_ON]->() RETURN r LIMIT 25
* MATCH ()-[r:WORKED\_WITH]->() RETURN r LIMIT 25
* MATCH ()-[r]->() RETURN r LIMIT 25

You can always return to the console to experiment with commands or to visually explore your data. If you think you are comfortable with the console, you can move on to the notebooks where we will learn how to use python to interface with our data set.

# Appendix: Deleting Data

If you find yourself in a situation where you would like to delete all relationships and nodes from the database and start over, the following command will remove all entities from the database. You can then run the commands in the section above to repopulate the data from the CSVs

* MATCH (n)

DETACH DELETE n