**Analysis of Twitter Data with the help of Neo4j Graph Database and Python**

A step-by-step tutorial of how to create, manage and query/process graphs



Photo from [internetdevels](https://internetdevels.com/blog/using-neo4j-graph-database-part1)

In today's data-driven world, the amount of information we generate is growing at an unprecedented rate. Traditional databases are great at handling structured data, but they struggle with the sheer volume, variety, and complexity of data that we generate. This is where graph databases come in.

but, what exactly are the *graph databases?*

**Graph databases**

Graph databases are designed to handle highly connected and complex data, making them an ideal choice for applications such as social networks, recommendation engines, and fraud detection systems. In a graph database, each node represents an entity, such as a person or a product, and each edge represents a relationship between those entities, such as a purchase or a friend connection.

**Neo4j** is a popular graph database that is designed to store and manage data in the form of graphs. In Neo4j, nodes and relationships can have various properties or attributes that provide additional information about them. One of the key benefits of Neo4j is that it is highly scalable and flexible, making it suitable for handling complex data relationships. It is also optimized for querying graph data, which allows for fast and efficient retrieval of information.

In this tutorial, we will explore how to use Neo4j and Python to analyze Twitter data. Specifically, we will discuss about:

* Installation and setup of Neo4j locally
* Retrieve Twitter data from a MongoDB database using Python
* Build nodes and relationships to populate the Neo4j Graph
* Perform queries to our graph

● ● ●

**Installation and setup of Neo4j**

Firstly, let’s download and install Neo4j for desktop. You can find the link [here](https://neo4j.com/download/).

For Neo4j Desktop Edition, you need to fill in the below form and then press Download Desktop

Graphical user interface, application, Word

Description automatically generated

An .exe file will be downloaded inside your Downloads folder. The installation steps are pretty straightforward.

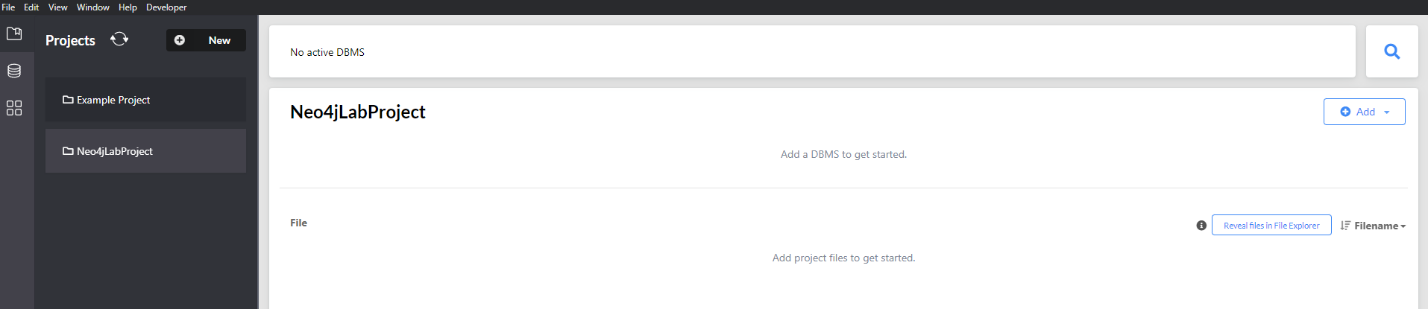
When you open the Neo4j Desktop app, firstly you need to choose a folder to store the application data. After that, we are ready to create our first Neo4j project.

To do so, we press the “New” button and then we choose “Create Project”. We will name our project “Neo4jLabProject”.

A picture containing text

Description automatically generated

This will be the interface of our new project:

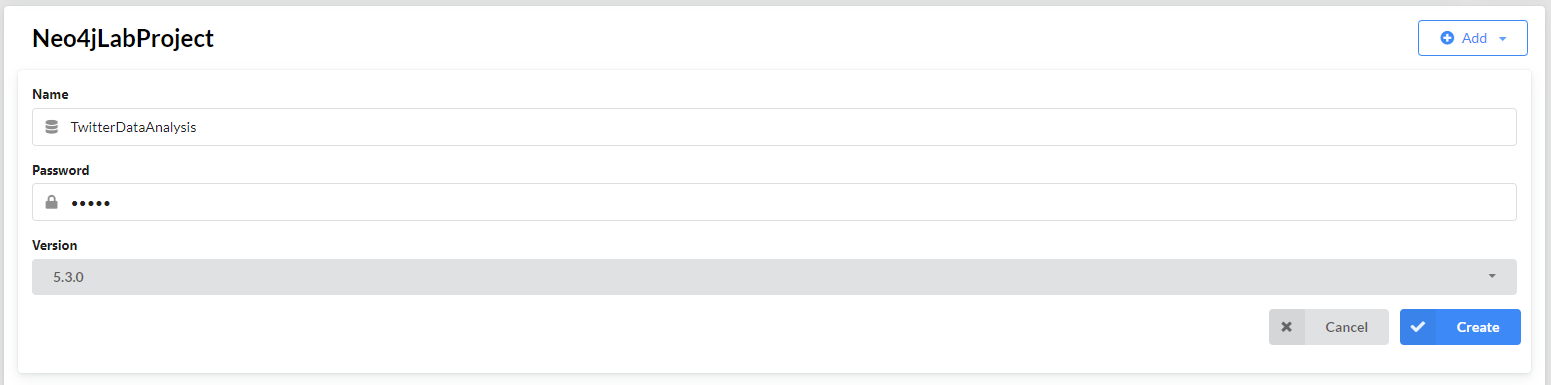


Now, we need to create a new local Database Management System (DBMS) that will be used to manage our graph.

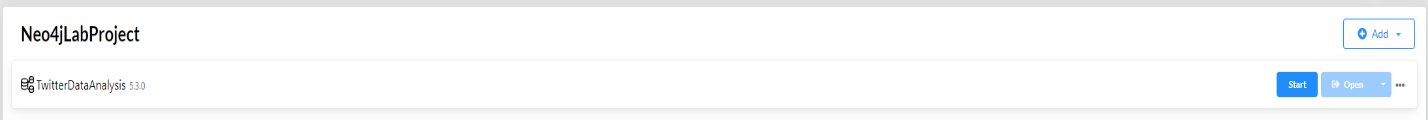
To achieve that, we press the “Add” button and we choose “Local DBMS”. We give a DBMS name and password and then we press “Create”.

**Be careful!**

***Write down the DBMS name and password as we will need them later in order to connect to the database.***



Now we are ready to start our database, by pressing the blue “Start” button at the right side.



This may take a few seconds. You will get notified when the database is active.

Now, we can examine our dataset.

● ● ●

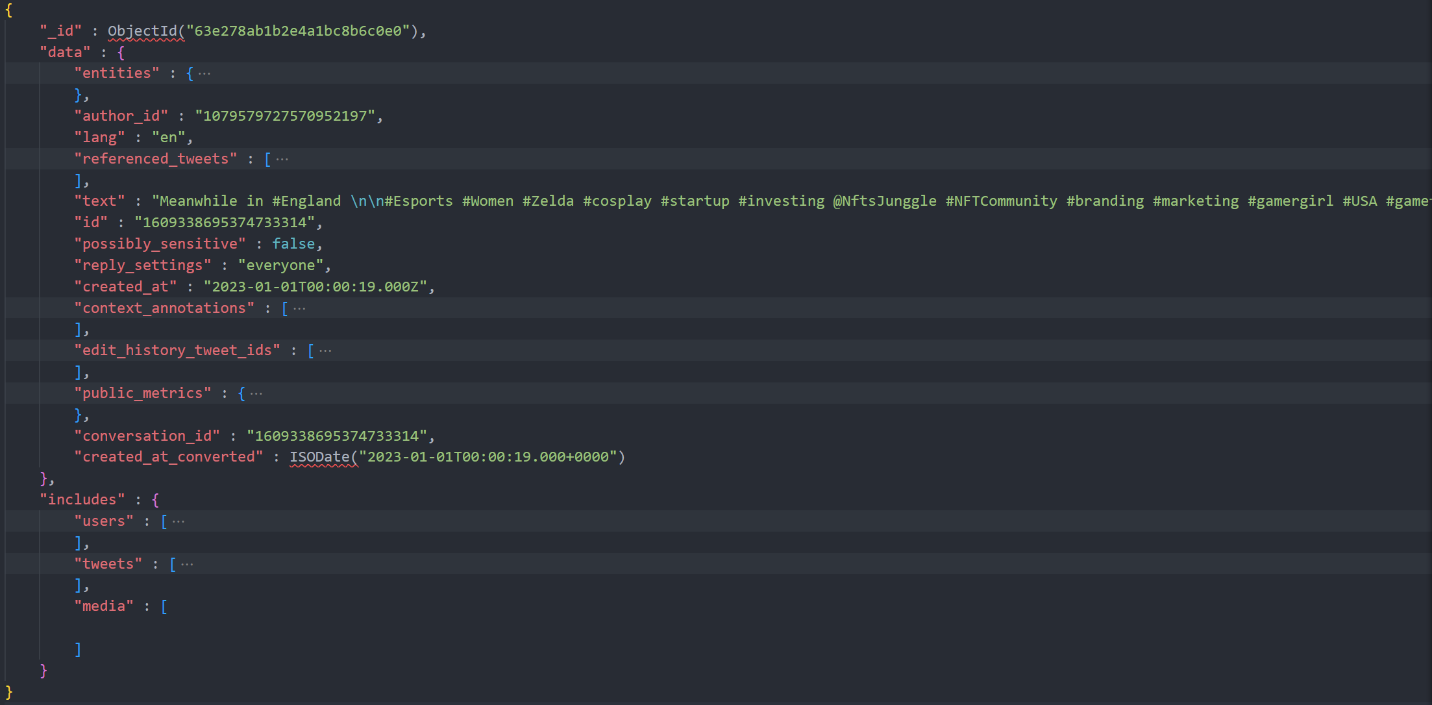
**Dataset**

The dataset is a JSON file, which contains 33,223 tweets retrieved by using Twitter API.

Each tweet has a unique *id* and 2 main fields, which contain more fields:

* *data* (contains 14 more fields)
* *includes* (contains 3 fields: *users, tweets, media*, which also contain more fields)

Let’s see an example for better understanding:



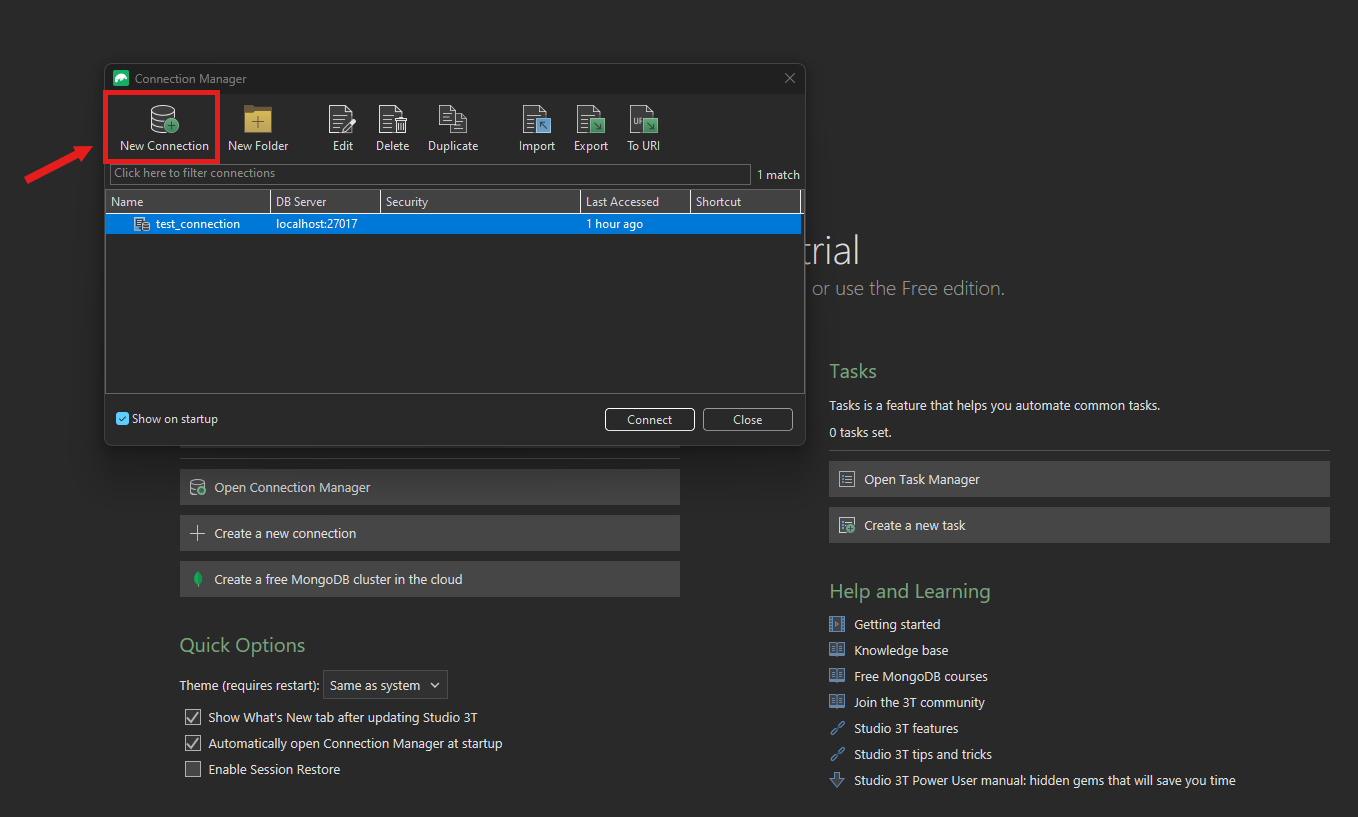
In the above picture, we can see the unique *id* of the tweet, the *data* branch with the 14 fields and the *includes* branch that contains the *users, tweets and media* branches.

**MongoDB**

We will use MongoDB to extract the data from the JSON file. You can download MongoDB from the link [here](https://www.mongodb.com/try/download/community). We also need a GUI in order to have a clear view of the dataset. We use [Studio 3T](https://studio3t.com/), but also [MongoDB Compass](https://www.mongodb.com/products/compass) is a quite popular choice.

Once the installation has been completed, you need to create a 3T account and then we are ready to create our connection.

The Connection Manager windows appears automatically on startup. Click “New Connection” and then select “Manually configure my connection settings”. Use the default properties and just choose a *Connection name*. Click “Save” and our new connection is ready!

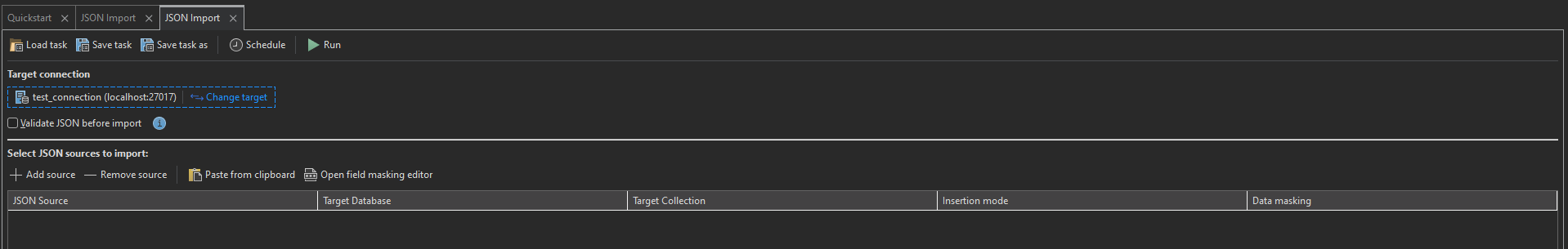


A screenshot of a computer

Description automatically generated with medium confidence

Now we are ready to import the dataset.

In the Global toolbar, click “Import” and then choose JSON. Click “Configure” and you will see the below interface:



In the Target Connection, choose the connection that we made before. Then, press “+ Add source” and select the JSON file with the dataset. Then click “Run” and wait until the process finishes.

Once it finishes, you will see a new database inside the connection branch, with the name of the folder that JSON is inside.

Graphical user interface, text, application

Description automatically generated

**Populate Graph**

In order to populate the Neo4j graph, we chose to extract the dataset into a .csv file and then use this file for the creation of the nodes and the relationships.

Open the dropdown menu of the new database that we made before, and inside the “Collections” folder, choose the JSON file (dataset). Then click the “Export” button in the Global toolbar and select CSV.

A screenshot of a computer

Description automatically generated

When you press “Configure”, you will see a similar interface like before, where we chose “Import”. In this interface, you can choose the destination path of the .csv file and which fields you want to be available to you (columns). Then, click “Run” and wait until the process finishes.

Now, it’s about time to introduce the Nodes and the Relationships of our graph.

● ● ●

**Nodes & Relationships**

|  |  |
| --- | --- |
| Node name | Description |
| Tweet | Represents a tweet (can be tweet, retweet, quote, reply) |
| User | Represents a Twitter user |
| Link | Represents a URL |
| Hashtag | Represents a hashtag |