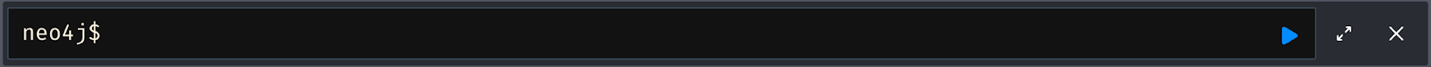
**Download Neo4J**[**Neo4J Desktop Version Download Link**](https://www.googleadservices.com/pagead/aclk?sa=L&ai=CARRBYxxnY6ScGsrBowbG2rcQrsSTqGnrxPLL-A7Z6uioqRMIABABILlUKAJgyYb_h_Cj7BKgAYrz9MkDyAEByAPYIKoEX0_QXcuFm_pC0wne63pU1yRrvSvnZtnR1xRFc4xc1UR5UciD1RT0RYqhnCm6gy6q2ceMCKMCgk3E4gdvWar9NeMVyaiWVL9AkPwJ-SdARVUvJ2ghG5OnHXNfOLYrMDvjwASjoaaw-AOABZBOiAWuwo3kOKAGZoAH3oyLNogHAZAHAagHpr4bqAe5mrECqAfz0RuoB-7SG6gH_5yxAqgHytwbqAePqbECqAfYprECqAfbqrECqAfQqrECoAiU0qYEsAgB0ggaEAIghAEyBIPAgA46CoCAgICAgICoAQJCAQSaCSVodHRwczovL25lbzRqLmNvbS9kb3dubG9hZC1uZW80ai1ub3cvsQmET4-FNkmHFbkJ2Ooam_oud134CQGYCwGqDAIIAbgMAegMBoIUGggBEhZuZW80aiBkZXNrdG9wIGRvd25sb2FkiBQHyBSVuaWuqven1GzQFQGYFgH4FgGAFwGSFwgSBggBEAMYduAXAg&ae=2&ved=2ahUKEwibs6jHwZj7AhUeEmIAHdNABsYQ0Qx6BAgHEAE&nis=8&dct=1&cid=CAASFeRo_0nbhiuhrYa2oASoxZkt6i4UUg&dblrd=1&sival=AF15MEB6fBM9mYPAR4BYSAEiopVsSQsRDeeP_18dPR2T8wtp61cUptJOFTfDBcG-iJscx8-J2AOG-D6koVmNZZn0Z9-TOO0a51TttrP-bmDYicQcYKZQfrsSGcDxzHpbf7Lptul73c8FpaShHBk3-6GXZVb3BZVK9VMCxYkoLMP3lfBRDGHpPRxW4fsFhCpRiZ93HvuIJO3W&sig=AOD64_2CbDcwBo9LMICCQaomzYz9oulZHw&adurl=https://neo4j.com/download-neo4j-now/%3Futm_program%3Dna-prospecting%26utm_source%3Dgoogle%26utm_medium%3Dcpc%26utm_campaign%3Dna-search-offers%26utm_adgroup%3Ddesktop-download%26utm_content%3Ddesktop-download%26utm_placement%3D%26utm_keyword%3Dneo4j%2520desktop%2520download%26utm_network%3Dg)

Google: Neo4J Desktop Installation

1. Open the package and install Neo4J Desktop.
2. Create new project
3. Create a new database
4. Open the project directory folder by hovering the active database and selecting the menu on the far right. Select Open Folder then Import. A new window should open with the project’s directory.  
   Graphical user interface, text, application, email, Teams

   Description automatically generatedGraphical user interface, text, application

   Description automatically generated
5. Import the files downloaded from the previous section into this folder. The files should appear in the new directory.
6. Close the import window.
7. Start the database by clicking start. This may take a couple of minutes.
8. Once it has been started, click Open. This opens the Neo4J Desktop browser. It allows data viewing in either graphical or tabular format by querying a database.
9. Make sure your Neo4j browser is connected to the database that was just created.Graphical user interface, text

   Description automatically generated
10. Begin by clicking into an empty cell.
11. Create a database by running **Query 1**. After pasting this query into the empty cell, click the play button. Fair warning: this may take up to 30 minutes. While Neo4J creates the database, it will show progress below the cell. Here is what will be displayed when the process is completed.Text

    Description automatically generated

**Query 1: Create Database**

|  |
| --- |
| //0. Create Everything MATCH (n) DETACH DELETE n;  //If Memory Error //CREATE OR REPLACE DATABASE <name>   //Create Tweets from TweetA :auto USING PERIODIC COMMIT 1000 LOAD CSV WITH HEADERS FROM 'file:///TweetToTweetData.csv' AS row WITH row where not row.UserA is null MERGE (tweet:Tweet { TweetID:toInteger(row.TweetTokenA), Tweet:row.A\_text\_clean, AuthorToken:toInteger(row.AuthorTokenA), Username:row.UserA, CreatedAt:row.A\_created\_at, Retweets:toInteger(row.A\_retweet\_count), Replies:toInteger(row.A\_reply\_count), Likes:toInteger(row.A\_like\_count), Quotes:toInteger(row.A\_quote\_count), Aspect:row.AspectsA, NegativeTweet:row.A\_is\_neg, NegativityScore:toFloat(row.A\_neg\_score), PositiveTweet:row.A\_is\_pos, PositivityScore:toFloat(row.A\_pos\_score), NeutralityScore:toFloat(row.A\_neutral\_score), CompoundScore:toFloat(row.A\_compound\_score) }); //Create Tweets from TweetB :auto USING PERIODIC COMMIT 1000 LOAD CSV WITH HEADERS FROM 'file:///TweetToTweetData.csv' AS row WITH row where not row.UserB is null MERGE (tweet:Tweet { TweetID:toInteger(row.TweetTokenB), Tweet:row.B\_text\_clean, AuthorToken:toInteger(row.AuthorTokenB), Username:row.UserB, CreatedAt:row.B\_created\_at, Retweets:toInteger(row.B\_retweet\_count), Replies:toInteger(row.B\_reply\_count), Likes:toInteger(row.B\_like\_count), Quotes:toInteger(row.B\_quote\_count), Aspect:row.AspectsB, NegativeTweet:row.B\_is\_neg, NegativityScore:toFloat(row.B\_neg\_score), PositiveTweet:row.B\_is\_pos, PositivityScore:toFloat(row.B\_pos\_score), NeutralityScore:toFloat(row.B\_neutral\_score), CompoundScore:toFloat(row.B\_compound\_score) }); //Create Tweet to Tweet relationship :auto USING PERIODIC COMMIT 1000 LOAD CSV WITH HEADERS FROM 'file:///TweetToTweetData.csv' AS row WITH row where not row.UserB is null  MATCH (a:Tweet {TweetID:toInteger(row.TweetTokenA)}), (b:Tweet {TweetID:toInteger(row.TweetTokenB)}) MERGE (a)-[:Similar{Score:toFloat(row.SimilarityScore)}]->(b);   //Author:Author Create N + R  //Create Author from AuthorA :auto USING PERIODIC COMMIT 1000 LOAD CSV WITH HEADERS FROM 'file:///AuthorToAuthorData.csv' AS row WITH row where not row.UserA is null  MERGE (author:Author { AuthorToken:toInteger(row.AuthorTokenA), NumTweets:toInteger(row.NumTweetsA), Username:row.UserA, Verified:row.A\_is\_verified, ProfilePic:row.A\_prof\_img\_url, Bio:row.A\_description, Followers: row.A\_follower\_count, Following: row.A\_following\_count });  //Create Author from AuthorB :auto USING PERIODIC COMMIT 1000 LOAD CSV WITH HEADERS FROM 'file:///AuthorToAuthorData.csv' AS row WITH row where not row.UserB is null  MERGE (author:Author { AuthorToken:toInteger(row.AuthorTokenB), NumTweets:toInteger(row.NumTweetsB), Username:row.UserB, Verified:row.B\_is\_verified, ProfilePic:row.B\_prof\_img\_url, Bio:row.B\_description, Followers: row.B\_follower\_count, Following: row.B\_following\_count });  //Create Author to Author Relationship :auto USING PERIODIC COMMIT 1000 LOAD CSV WITH HEADERS FROM 'file:///AuthorToAuthorData.csv' AS row WITH row where not row.UserB is null  MATCH (a:Author {AuthorToken:toInteger(row.AuthorTokenA)}), (b:Author {AuthorToken:toInteger(row.AuthorTokenB)}) MERGE (a)-[:Similar{ Score:toFloat(row.MeanSimilarity\_Score), MutuallySimilarTweets:row.NumMutualSimiliarTweets, NumCombinedTweets:row.NumTweetsCombined } ] ->(b);     //Create Author to Tweet Relationship  MATCH (a:Author),(b:Tweet) WHERE a.AuthorToken = b.AuthorToken  MERGE (a)-[:Tweeted]->(b) |

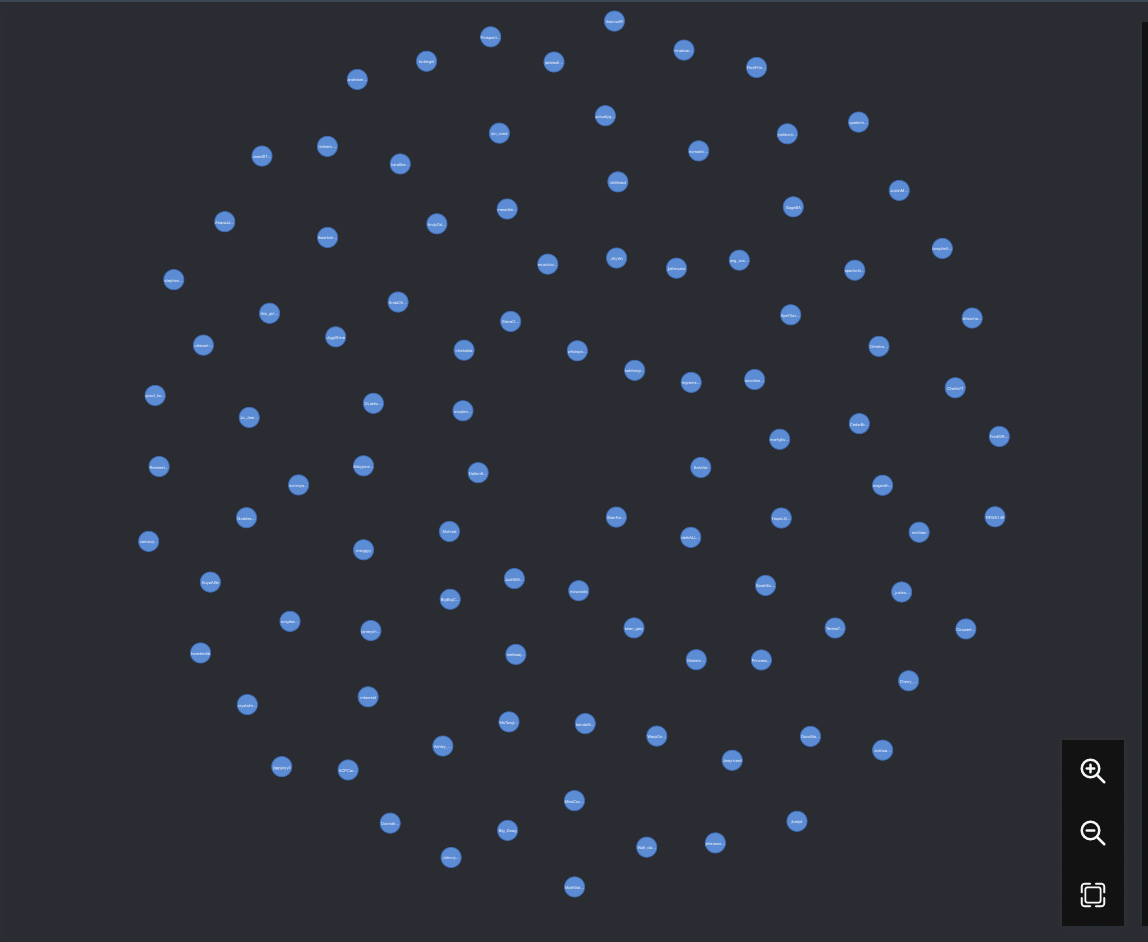
**Query 2: Show Tweets**

|  |
| --- |
| MATCH(n:Tweet) RETURN n LIMIT 100; |

****

**Query 3: Show Authors**

|  |
| --- |
| MATCH(n:Author) RETURN n LIMIT 100; |

****

**Query 4: Show Tweets and Authors**

|  |
| --- |
| MATCH(n:Author) RETURN n LIMIT 100; |

**Background pattern

Description automatically generated**

**Query 5: Show Tweets with a keyword**

|  |
| --- |
| MATCH(n) WHERE n.Tweet =~ '.\*delay.\*' RETURN n; |

**A picture containing background pattern

Description automatically generated**

**Query 6: Similar Tweets from one Tweet ID**

|  |
| --- |
| MATCH (n:Tweet {TweetID:1560045014729777154})-[:Similar]-(tweet2:Tweet) RETURN tweet2 |

**Chart, bubble chart

Description automatically generated**

**Query 7: Similar Tweets from one Tweet ID with relationship strength threshold**

|  |
| --- |
| MATCH (n:Tweet {TweetID:1560045014729777154} )-[s:Similar]-(tweet2:Tweet) WHERE s.Score >0.9 RETURN n,tweet2 |

**Bubble chart

Description automatically generated with medium confidence**

**Query 8: Author Attributes from tweet similarity**

|  |
| --- |
| MATCH (n:Tweet{TweetID:1584729987860795392})-[s:Similar]-(similarTweet:Tweet)-[t:Tweeted]-(a:Author) RETURN DISTINCT a.Username |

**Graphical user interface, application

Description automatically generated**

**Query 9: Show Tweets with greater than 10 likes and 5 retweets**

|  |
| --- |
| MATCH(n:Tweet) WHERE n.Likes > 10 and n.Retweets > 5 RETURN n |

**Graphical user interface

Description automatically generated**

**Query 10: Show very similar tweets**

|  |
| --- |
| MATCH (tweet1:Tweet)-[s:Similar]-(tweet2:Tweet) WHERE s.Score > 0.99 RETURN tweet1.Tweet,tweet2.Tweet LIMIT 100; |

**Text

Description automatically generated**

**Query 11: Get author attributes from very similar tweets**

|  |
| --- |
| MATCH (a1:Author)-[:Tweeted]-(tweet1:Tweet)-[s:Similar]-(tweet2:Tweet)-[:Tweeted]-(a2:Author) WHERE s.Score > 0.99 RETURN a1.Username,a1.NumTweets,a2.Username,a2.NumTweets LIMIT 100; |

**Calendar

Description automatically generated**

**Query 12: Case insensitive search**

*Username and Tweets of Tweets containing  “delay” + “phoenix”*

|  |
| --- |
| MATCH(n:Tweet)-[:Tweeted]-(a:Author) WHERE n.Tweet =~ '.\*delay.\*' AND (n.Tweet =~ '.\*Phoenix.\*') RETURN a.Username, n.Tweet; |

**Text

Description automatically generated**

**Demo Query 1: Find Negative Tweet Clusters**

|  |
| --- |
| //1. Find Negative Tweet Clusters  MATCH(n:Tweet) WHERE n.NegativeTweet = 'True' RETURN n Limit 100; |

**Demo Query 2: Identify authors responsible for negative Tweets**

|  |
| --- |
| //2. Identify authors responsible for negative Tweets match(n:Tweet)-[:Tweeted]-(a:Author)-[:Tweeted]-(n2:Tweet) WHERE n.NegativeTweet = 'True' return a.Followers,a.Following,a.Username,n2.CreatedAt,n2.Tweet Limit 100; |

**Demo Query 3: View authors' tweets**

|  |
| --- |
| //3. View authors' tweets match(a:Author)-[:Tweeted]-(n:Tweet) where a.Username = 'RickPrince\_' or a.Username = 'politicalteach' return n,a//n.Username,n.Tweet |

**Demo Query 4: View authors' tweets & Attributes**

|  |
| --- |
| **//4. View authors' tweets & Attributes match(a:Author)-[:Tweeted]-(n:Tweet) where a.Username = 'RickPrince\_' or a.Username = 'politicalteach' return a.Username as Username, a.Followers as Followers,a.Following as Following,n.CompoundScore asCompoundScore,n.CreatedAt as CreatedAt,n.Tweet as Tweet Order by a.Username,n.CompoundScore asc** |

**Demo Query 5: Show negative tweets with a lot of exposure**

|  |
| --- |
| //Show negative tweets with a lot of exposure MATCH(n) WHERE n.NegativeTweet = 'True' AND n.Likes > 5 AND n.Retweets > 3 RETURN n.Likes, n.Retweets,n.AuthorHandle,n.CreatedAt, n.Tweet ORDER BY n.Likes DESC, n.Retweets DESC LIMIT 50; |