HU Extension Assignment 10 E63 Big Data Analytics

Handed out: 04/09/2016 Due by 11:30PM EST on Friday, 04/15/2016

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document with explanations of the purpose of those commands. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. For issues and comments visit the class Discussion Board. You are not obliged to use Java or Eclipse. You are welcome to use any language and any IDE of your choice.

Problem 1. The following is the content of Movies database. Bring that database into Neo4J using curl.

```
CREATE (matrix1:Movie { title : 'The Matrix', year : '1999-03-31' })

CREATE (matrix2:Movie { title : 'The Matrix Reloaded', year : '2003-05-07' })

CREATE (matrix3:Movie { title : 'The Matrix Revolutions', year : '2003-10-27' })

CREATE (keanu:Actor { name:'Keanu Reeves' })

CREATE (laurence:Actor { name:'Laurence Fishburne' })

CREATE (carrieanne:Actor { name:'Carrie-Anne Moss' })

CREATE (keanu)-[:ACTS_IN { role : 'Neo' }]->(matrix1)

CREATE (keanu)-[:ACTS_IN { role : 'Neo' }]->(matrix2)

CREATE (keanu)-[:ACTS_IN { role : 'Neo' }]->(matrix3)

CREATE (laurence)-[:ACTS_IN { role : 'Morpheus' }]->(matrix1)

CREATE (laurence)-[:ACTS_IN { role : 'Morpheus' }]->(matrix2)

CREATE (laurence)-[:ACTS_IN { role : 'Morpheus' }]->(matrix3)

CREATE (carrieanne)-[:ACTS_IN { role : 'Trinity' }]->(matrix1)

CREATE (carrieanne)-[:ACTS_IN { role : 'Trinity' }]->(matrix2)

CREATE (carrieanne)-[:ACTS_IN { role : 'Trinity' }]->(matrix3)
```

Solution:
Download and install Neo4j as was discussed in the notes.

There are two options to deal with Authorization. One option is to change the auth to access Neo4j located in the file neo4j-server.properties as in

```
# Require (or disable the requirement of) auth to access Neo4j
#dbms.security.auth_enabled=true
dbms.security.auth_enabled=false
```

Another option is to place the user name and password in the curl command directly as i

```
http://neo4j:adfasf@localhost:7474/db/data/transaction/commit
```

Where neo4j is username and adfasf is password. See the entire relevant curl command in the command section.

Download a window version of *curl, i.e.* curl-7.48.0-win64-mingw from https://curl.haxx.se/download.html

To run the commands assigned in this problem I have written the following code:
The relevant part of the code are presented in larger font.

```
{
"statements":[
{
"statement":"CREATE (matrix1:Movie{props} ) RETURN matrix1" ,
"parameters": {
"props" :{
"title" : "The Matrix", "year" : "1999-03-31"
}
}
}
,
{
"statement":"CREATE (matrix2:Movie{props} ) RETURN matrix2" ,
"parameters": {
"props" :{
"title" : "The Matrix Reloaded", "year" : "2003-05-07"
}
}
}
,
{
"statement":"CREATE (matrix3:Movie{props} ) RETURN matrix3" ,
"parameters": {
"props" :{
"title" : "The Matrix Revolutions", "year" : "2003-10-27 "
}
}
}
,
{
"statement":"CREATE (keanu:Actor{props} ) RETURN keanu" ,
"parameters": {
"props" :{
"name" : "Keanu Reeves"
}
}
}

,
{
"statement":"CREATE (laurence:Actor{props} ) RETURN laurence" ,
"parameters": {
"props" :{
"name" : "Laurence Fishburne"
}
}
}
,

{
"statement":"CREATE (carrieanne:Actor{props} ) RETURN carrieanne" ,
```

```
"parameters": {
"props" :{
"name" : "Carrie-Anne Moss"
}
}
}
,
{
"statement": "CREATE (keanu)-[:ACTS_IN { role : 'Neo' }]->(matrix1)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}

,
{
"statement": "CREATE (keanu)-[:ACTS_IN { role : 'Neo' }]->(matrix2)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}

,
{
"statement": "CREATE (keanu)-[:ACTS_IN { role : 'Neo' }]->(matrix3)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}
,
{
"statement": "CREATE (laurence)-[:ACTS_IN { role : 'Morpheus' }]->(matrix1)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}
,
{
"statement": "CREATE (laurence)-[:ACTS_IN { role : 'Morpheus' }]->(matrix2)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}
,
{
"statement": "CREATE (laurence)-[:ACTS_IN { role : 'Morpheus' }]->(matrix3)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}
,
{
"statement": "CREATE (carrieanne)-[:ACTS_IN { role : 'Trinity' }]->(matrix1)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}
,
{
"statement": "CREATE (carrieanne)-[:ACTS_IN { role : 'Trinity' }]->(matrix2)",
"resultDataContents": [
"row",
"graph"
],
"includeStats": true
}
,
{
"statement": "CREATE (carrieanne)-[:ACTS_IN { role : 'Trinity' }]->(matrix2)",
"resultDataContents": [
```

```
"row",
"graph"
],
"includeStats": true
}
 ]
}
```

Command to run:
CD to location of Curl:

```
C:\Users\SK\Documents\MyClasses\Harvard_cscie55\E63_Spring2016-2\Downloads\curl-7.48.0-win64-mingw\curl-7.48.0-win64-mingw\bin>
```

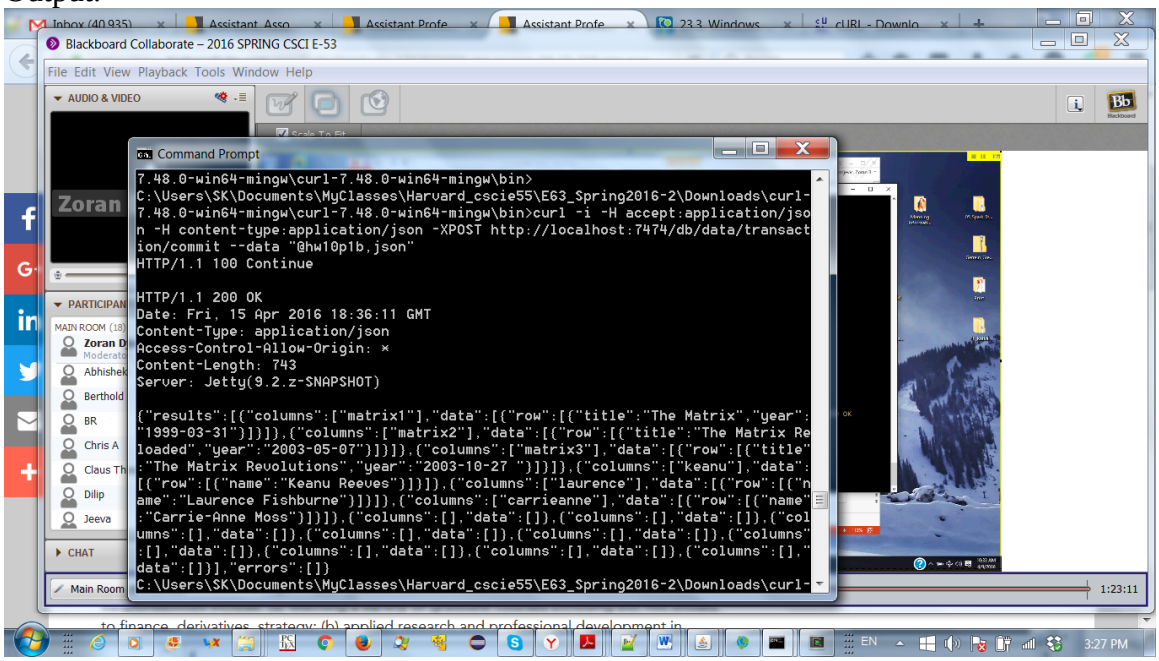
If authorization is disabled Then enter comman in one line

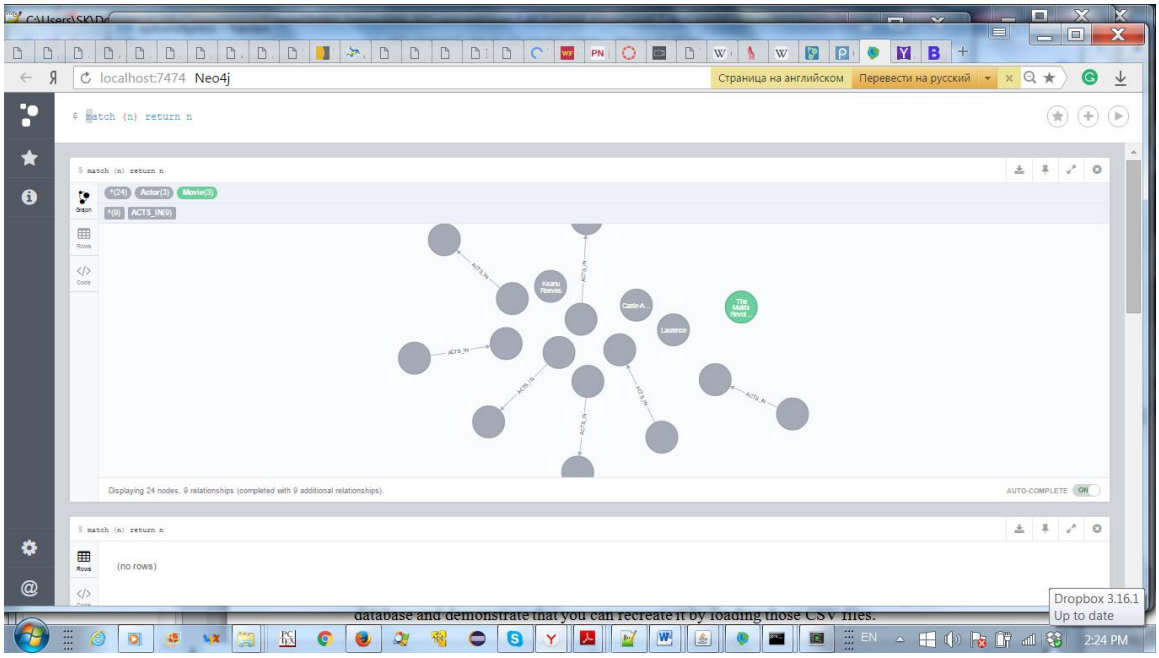
```
curl -i -H accept:application/json -H content-type:application/json -XPOST
http://localhost:7474/db/data/transaction/commit --data "@hw10p1b,json"
```

Or, if the username and password are required for authorization then:

```
curl  -i -H  accept:application/json  -H content-type:application/json -XPOST
http://neo4j:absdfd@localhost:7474/db/data/transaction/commit --data "@hw10p1b,json"
```

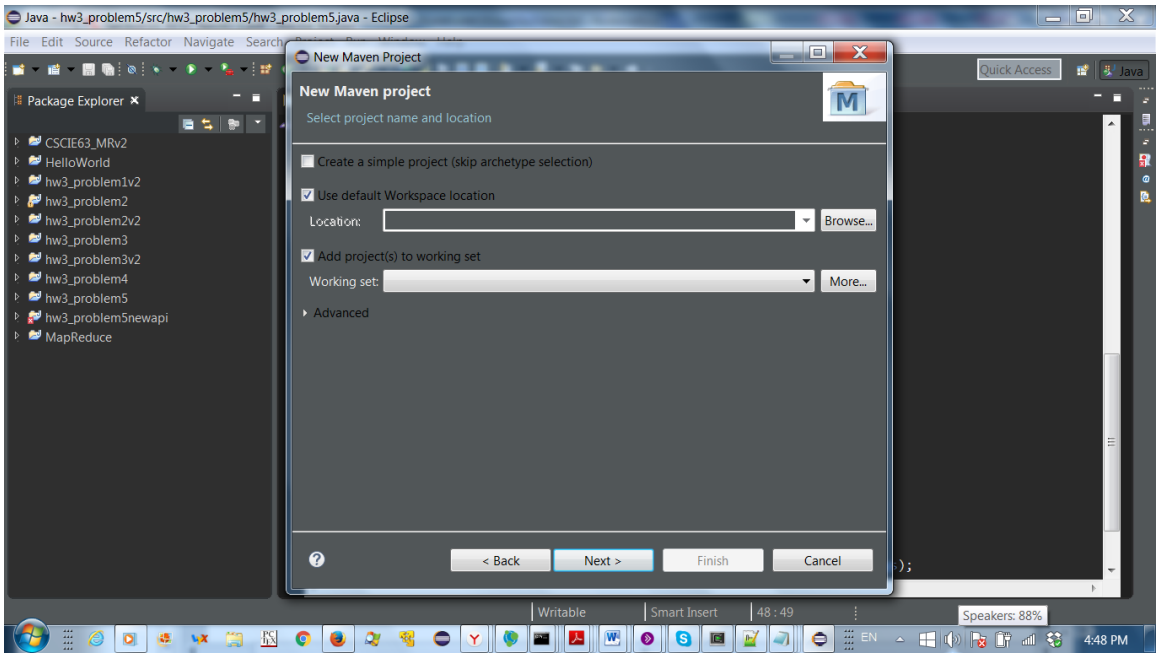
Output:

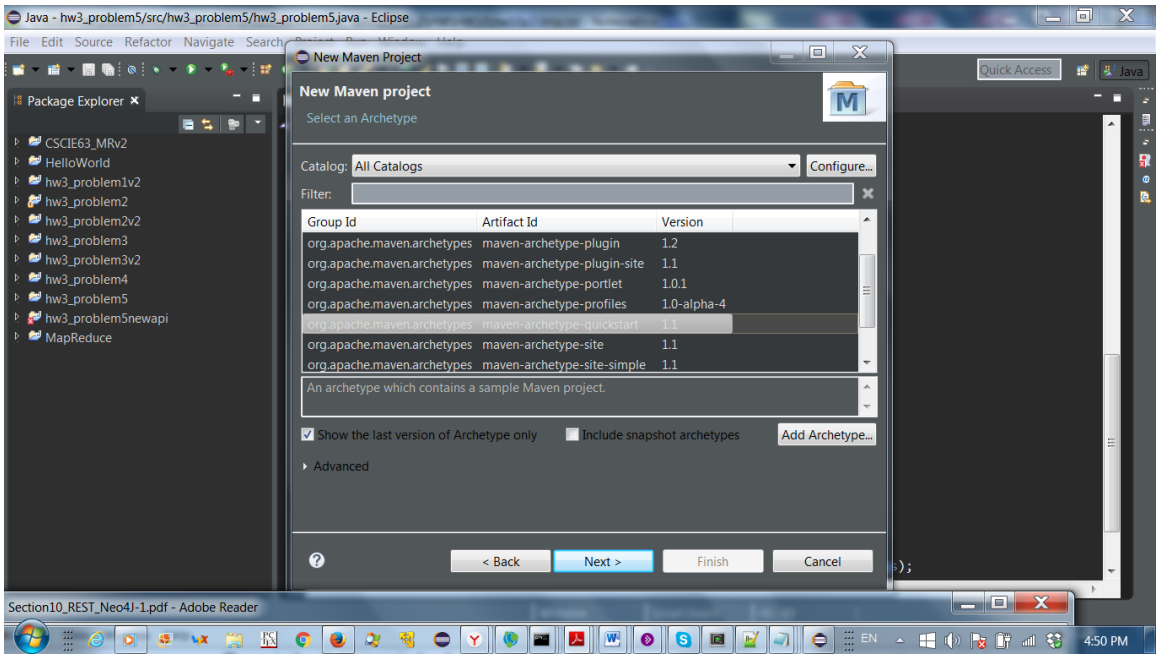


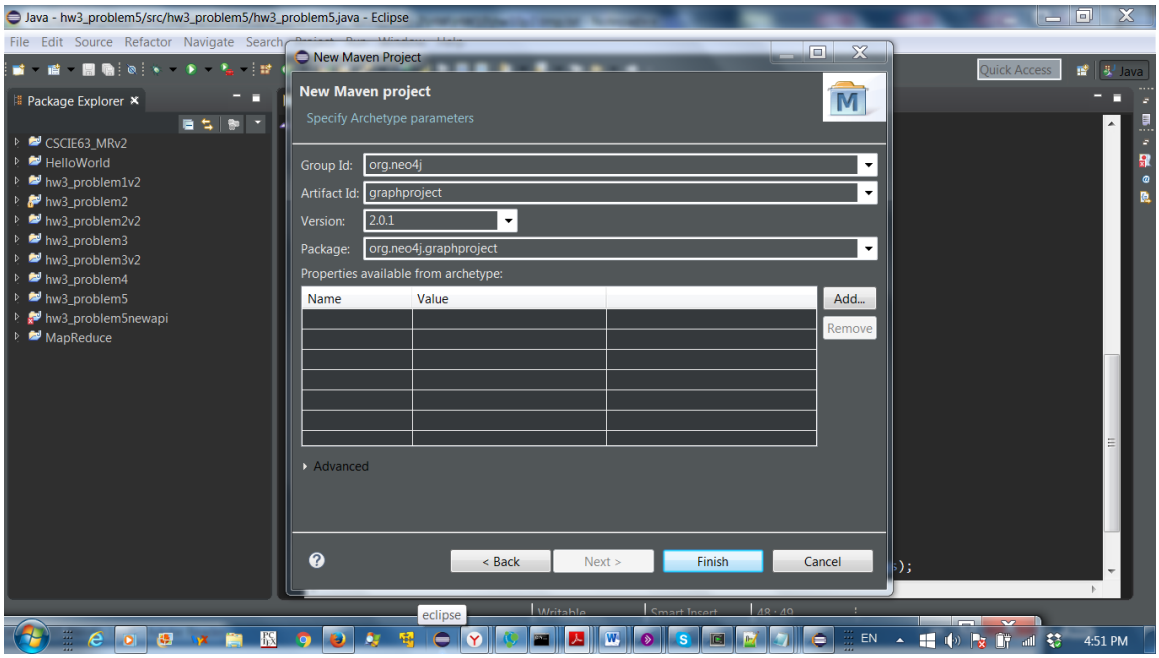


Problem 2. Keanu Reeves acted in the movie "John Wick" which is not in the database. That movie was directed by Chad Stahelski and David Leitch. Cast of the movie included William Dafoe and Michael Nyquist.

Add all of those people and the roles they played in this movie to the database using JAVA REST API or one of other RESTful APIs for Neo4J in a language of your choice. Demonstrate that you have successfully brought data about John Wick movie into the database. You can use Cypher Browser or any other means.



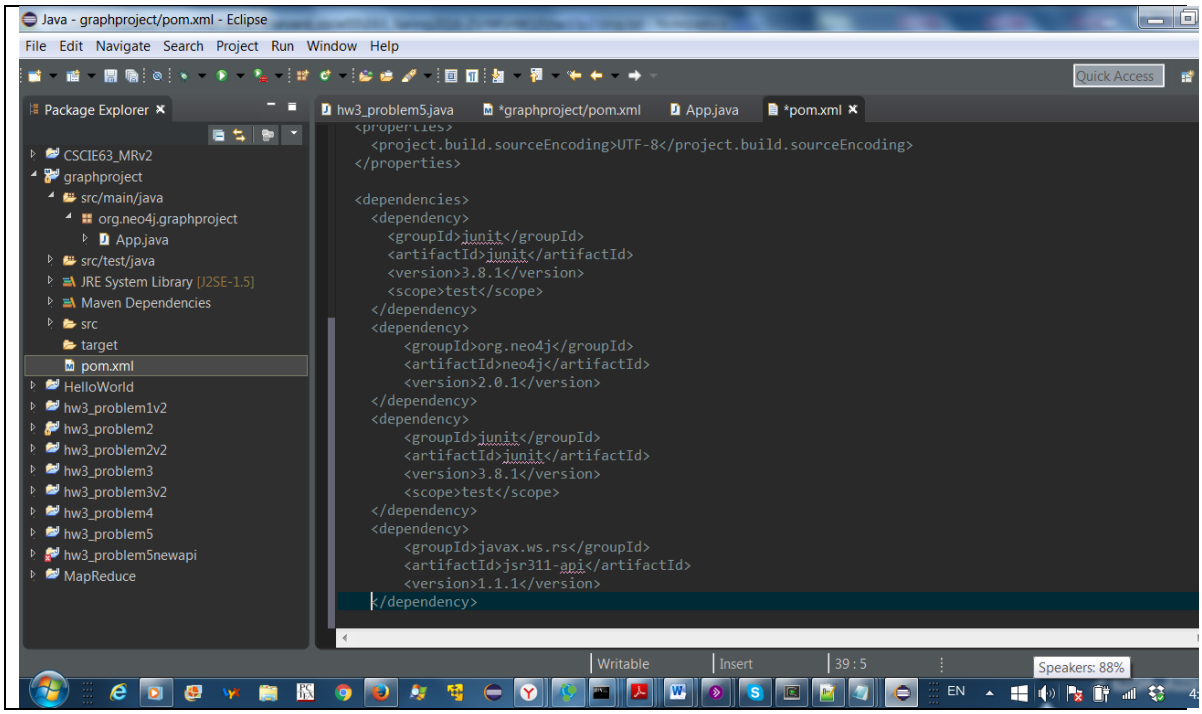




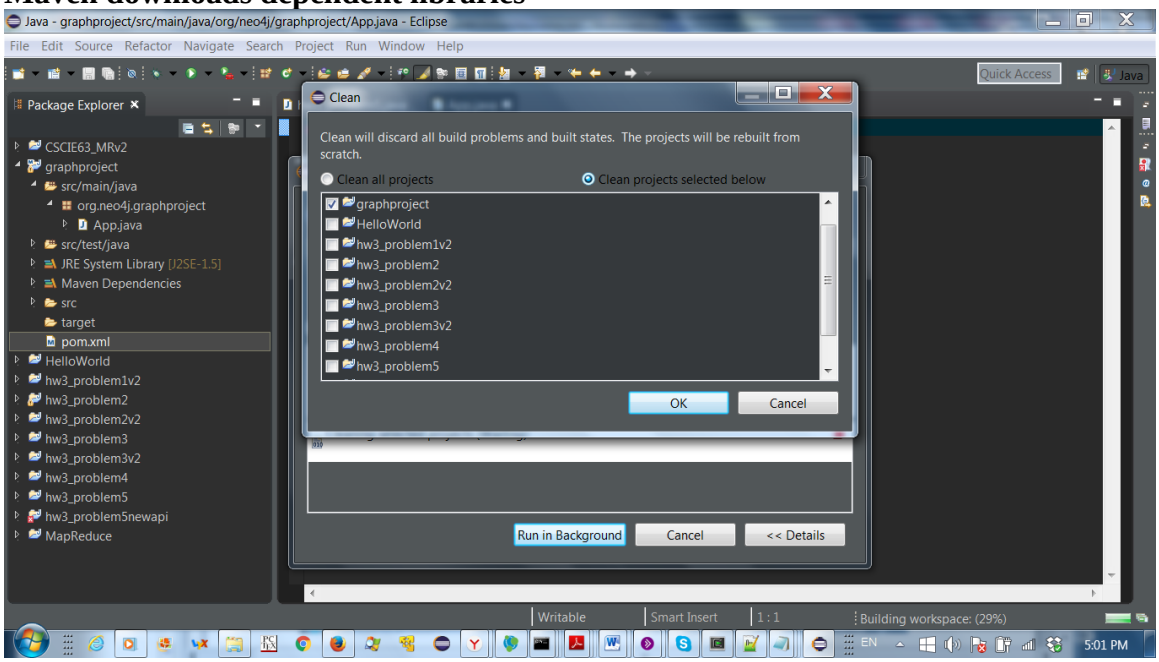
Added dependencies in the pom.xml file

```
<dependency>
		<groupId>org.neo4j</groupId>
		<artifactId>neo4j</artifactId>
		<version>2.0.1</version>
	</dependency>
	<dependency>
		<groupId>junit</groupId>
		<artifactId>junit</artifactId>
		<version>3.8.1</version>
```

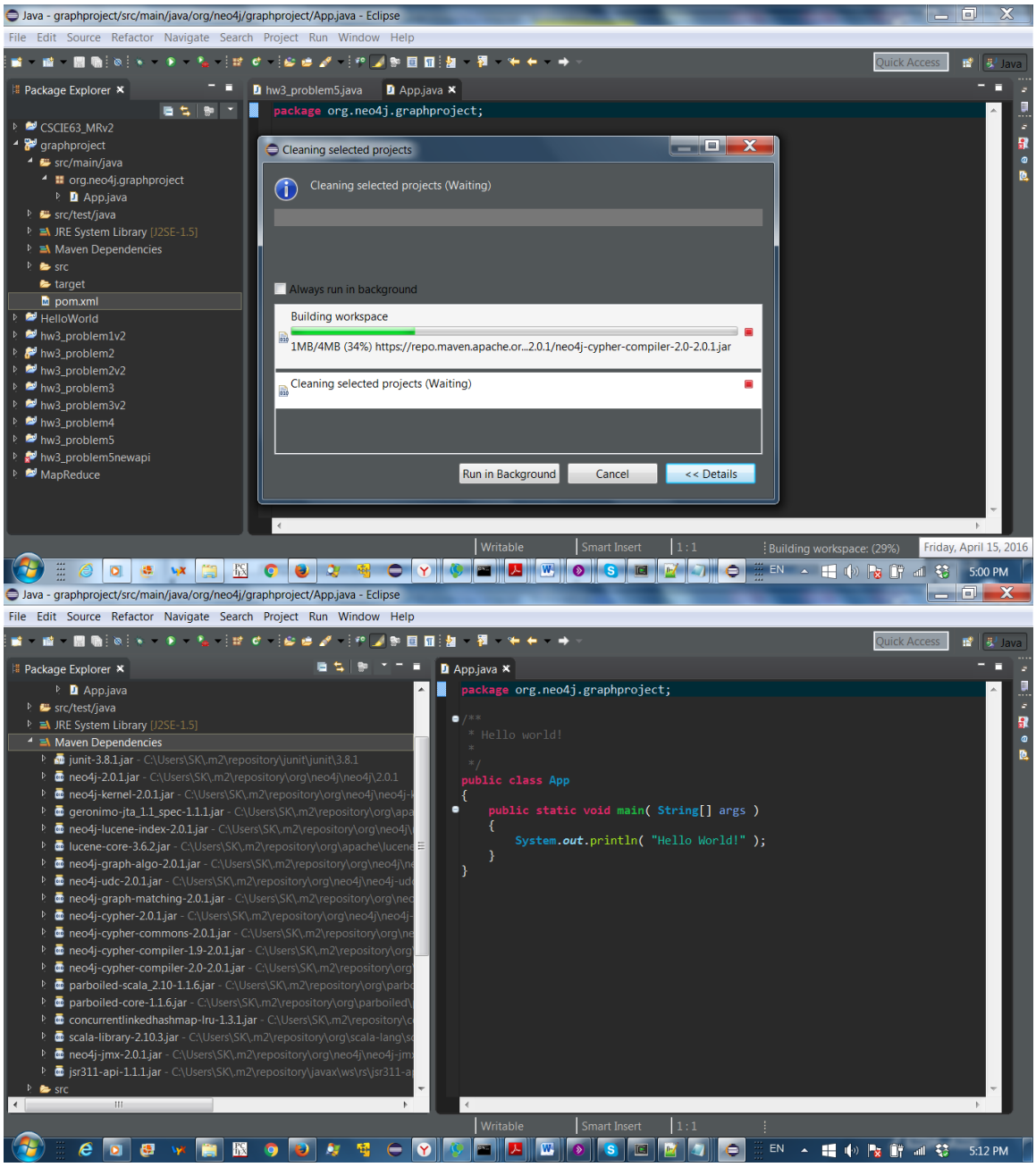
```
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>javax.ws.rs</groupId>
            <artifactId>jsr311-api</artifactId>
            <version>1.1.1</version>
        </dependency>
```



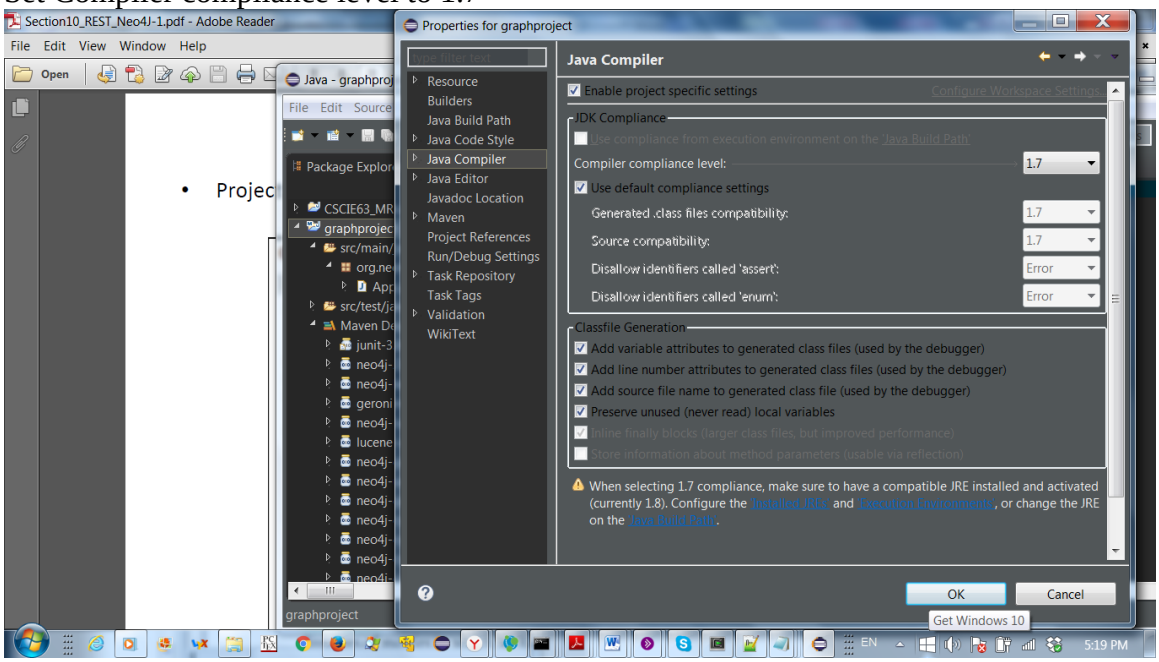
Maven downloads dependent libraries



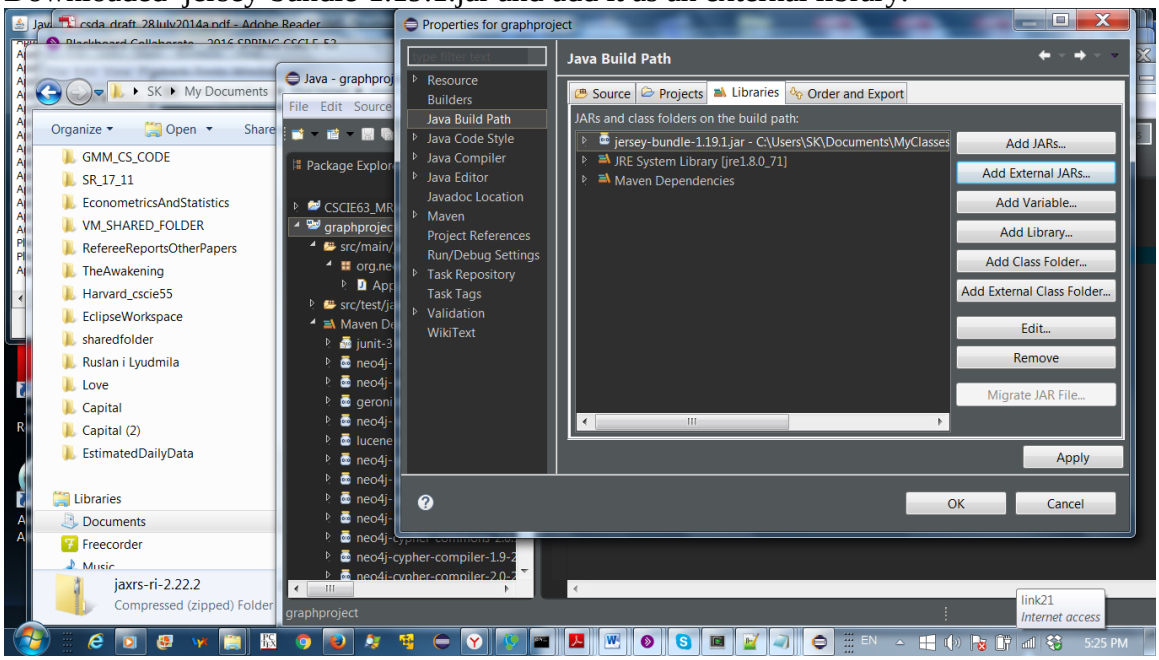
Maven dependencies



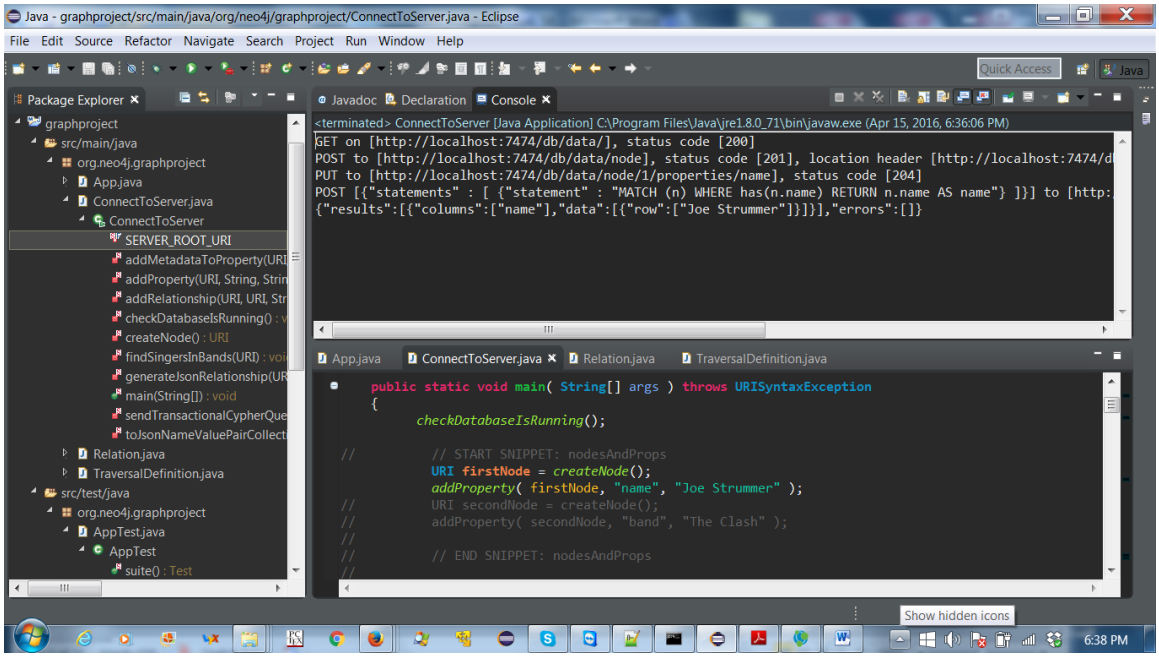
Set Compiler compliance level to 1.7



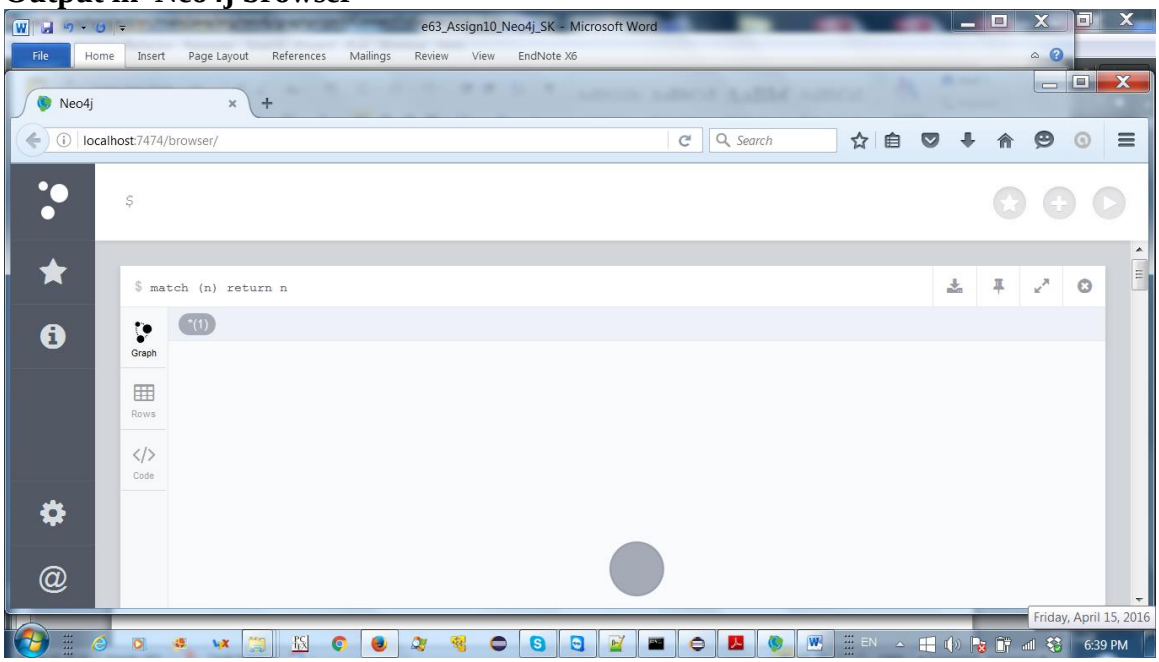
Downloaded jersey-bundle-1.19.1.jar and add it as an external library.



Run the code and get the response:



Output in Neo4j browser



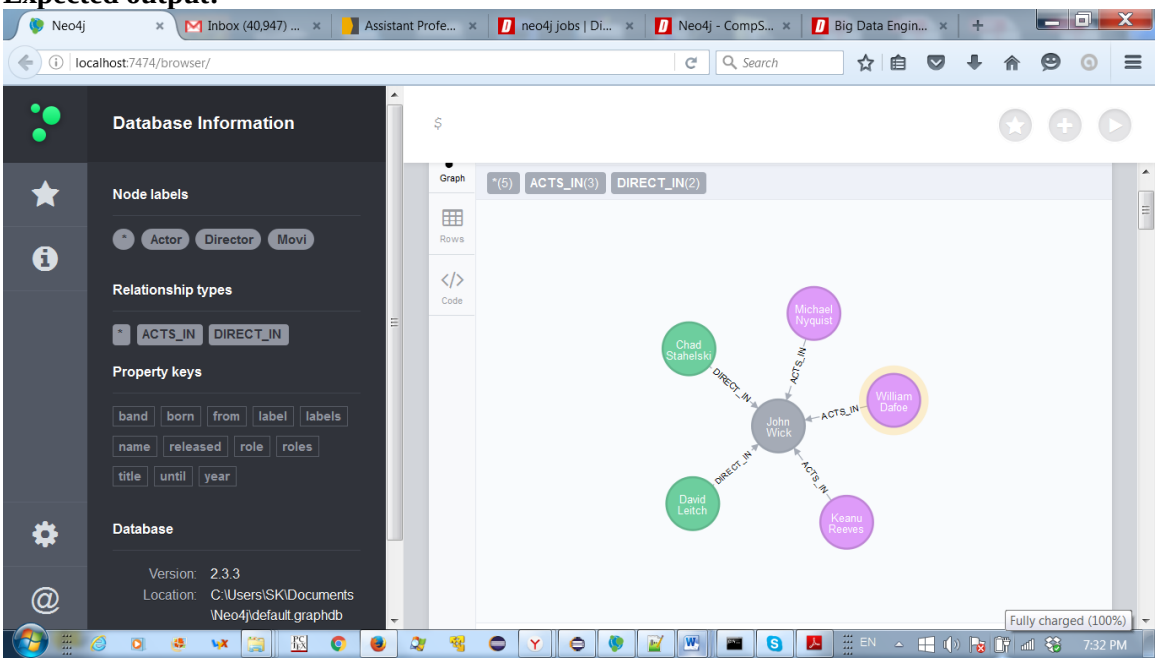
Keanu Reeves acted in the movie “John Wick” which is not in the database. That movie was directed by Chad Stahelski and David Leitch. Cast of the movie included William Dafoe and Michael Nyquist.

The set of commands:

```
CREATE (m:Movi { name: 'John Wick ' });
MATCH  (m: Movi { name: 'John Wick ' }) CREATE (:Actor { name: 'Keanu Reeves '})  - [:ACTS_IN ]->(m);
MATCH  (m: Movi { name: 'John Wick ' })  CREATE (:Actor { name: 'William Dafoe '}) - [:ACTS_IN ]->(m);
MATCH  (m: Movi { name: 'John Wick ' })  CREATE (:Actor { name: 'Michael Nyquist '}) - [:ACTS_IN ]->(m);
```

```
MATCH  (m: Movi { name: 'John Wick ' })  CREATE (:Director { name: 'Chad Stahelski '}) - [:DIRECT_IN ]->(m);
MATCH  (m: Movi{ name: 'John Wick ' })  CREATE (:Director { name: 'David Leitch '})  - [:DIRECT_IN ]->(m);
```

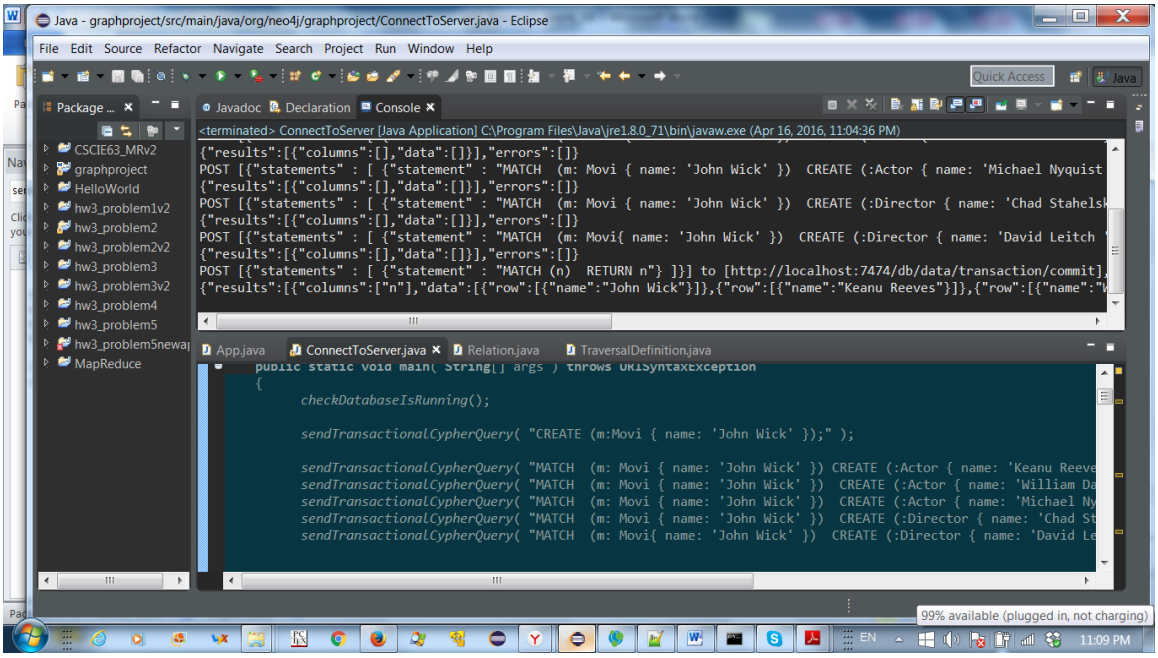
Expected output:

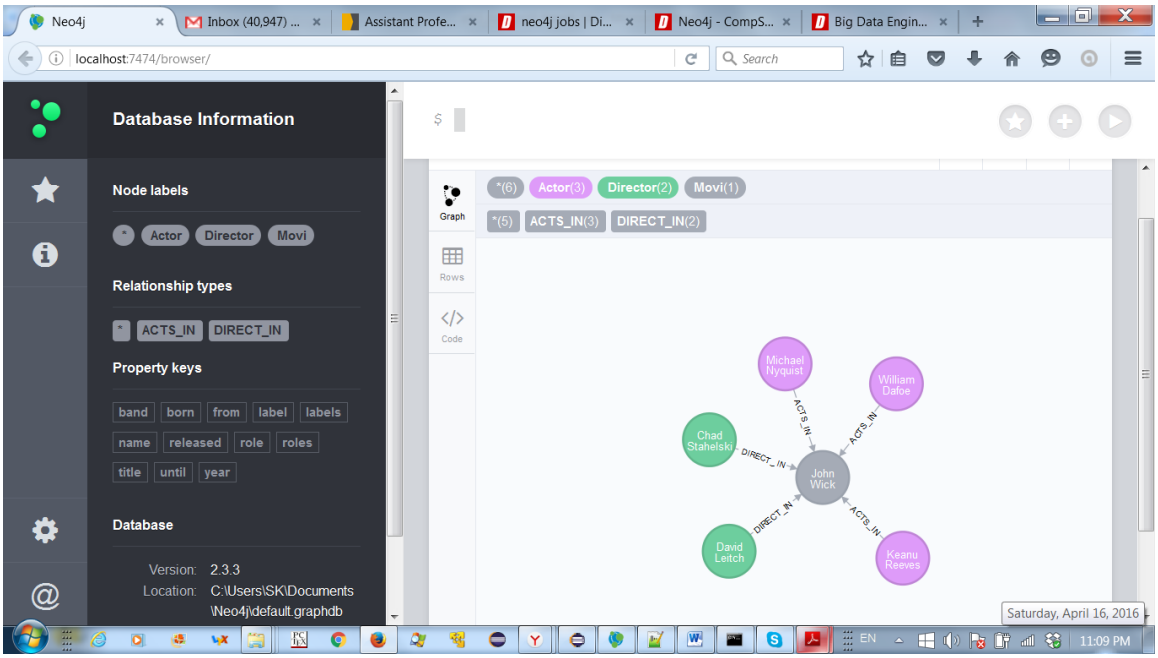


JAVA: Critical changes

Important: Use comma like this for internal input ' ' a la 'John Wick' in this case no change of function *sendTransactionalCypherQuery()* is required.

```
public static void main( String[] args ) throws URISyntaxException
    {
        checkDatabaseIsRunning();

sendTransactionalCypherQuery( "CREATE (m:Movi { name: 'John Wick' });" );

sendTransactionalCypherQuery( "MATCH  (m: Movi { name: 'John Wick' }) CREATE (:Actor { name: 'Keanu Reeves'})  - [:ACTS_IN ]->(m); " );

sendTransactionalCypherQuery( "MATCH  (m: Movi { name: 'John Wick' })  CREATE (:Actor { name: 'William Dafoe '}) - [:ACTS_IN ]->(m); " );

sendTransactionalCypherQuery( "MATCH  (m: Movi { name: 'John Wick' })  CREATE (:Actor { name: 'Michael Nyquist '}) - [:ACTS_IN ]->(m);" );

sendTransactionalCypherQuery( "MATCH  (m: Movi { name: 'John Wick' })  CREATE (:Director { name: 'Chad Stahelski '}) - [:DIRECT_IN ]->(m);" );
sendTransactionalCypherQuery( "MATCH  (m: Movi{ name: 'John Wick' })  CREATE (:Director { name: 'David Leitch '})  - [:DIRECT_IN ]->(m);" );

sendTransactionalCypherQuery( "MATCH (n)  RETURN n" );
    }
```





Note: there was also an attempt to change the function sendTransactionalCypherQuery(..)
Before I realized that the comma must be changed. For historical purposes I keep this function here. However, it is no more required for this specific assignment.

```
sendTransactionalCypherQuery("CREATE  (m:Movi", ")",  "name", "John Wick");
```

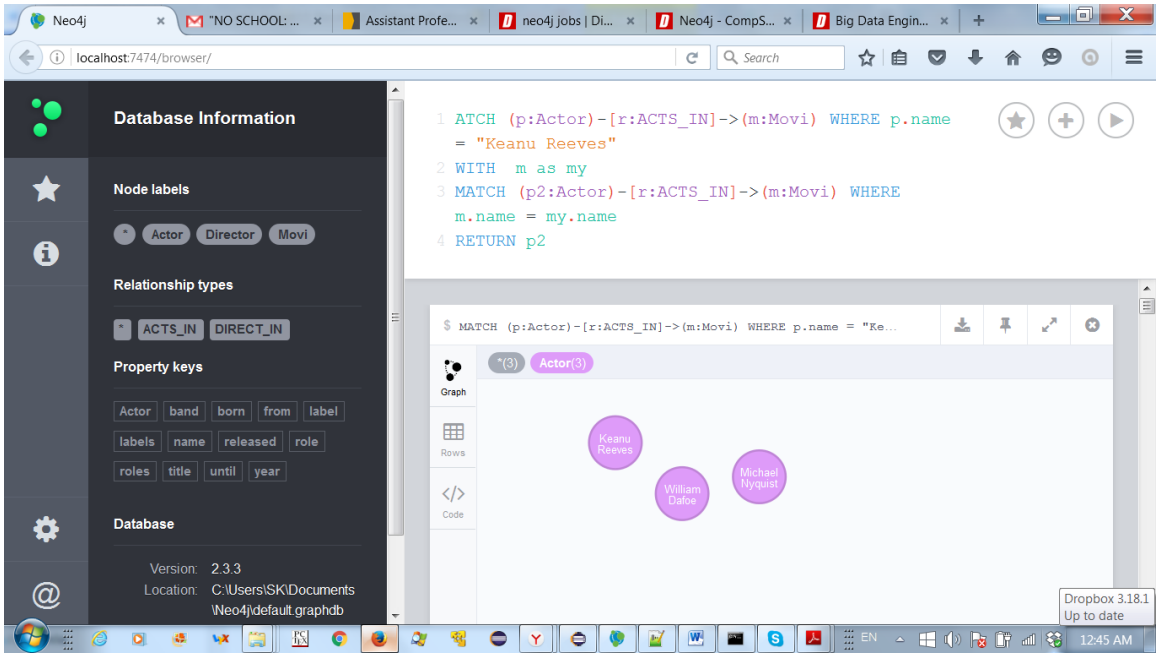
```
private static void sendTransactionalCypherQuery(String queryPart1, String queryPart2, String props_key, String props_value) {
    // START SNIPPET: queryAllNodes
    final String txUri = SERVER_ROOT_URI + "transaction/commit";
    WebResource resource = Client.create().resource( txUri );
```

```
// CRITICAL CHANGE

// Example of use: sendTransactionalCypherQuery("CREATE  (m:Movi", ")",  "name", "John Wick");
String payload = "{\"statements\" : [ {\"statement\" : \" "  + queryPart1 + "{props}"  + queryPart2 +  "\" ,\"parameters\":{ \"props\":{
\"" +props_key +"\" : \""+ props_value+"\"}}} ]}";


    ClientResponse response = resource
            .accept( MediaType.APPLICATION_JSON )
            .type( MediaType.APPLICATION_JSON )
            .entity( payload )
            .post( ClientResponse.class );

    System.out.println( String.format(
            "POST [%s] to [%s], status code [%d], returned data: "
                    + System.lineSeparator() + "%s",
            payload, txUri, response.getStatus(),
            response.getEntity( String.class ) ) );

    response.close();
    // END SNIPPET: queryAllNodes
```

Problem 3. Find a list of actors playing in movies in which Keanu Reeves played.

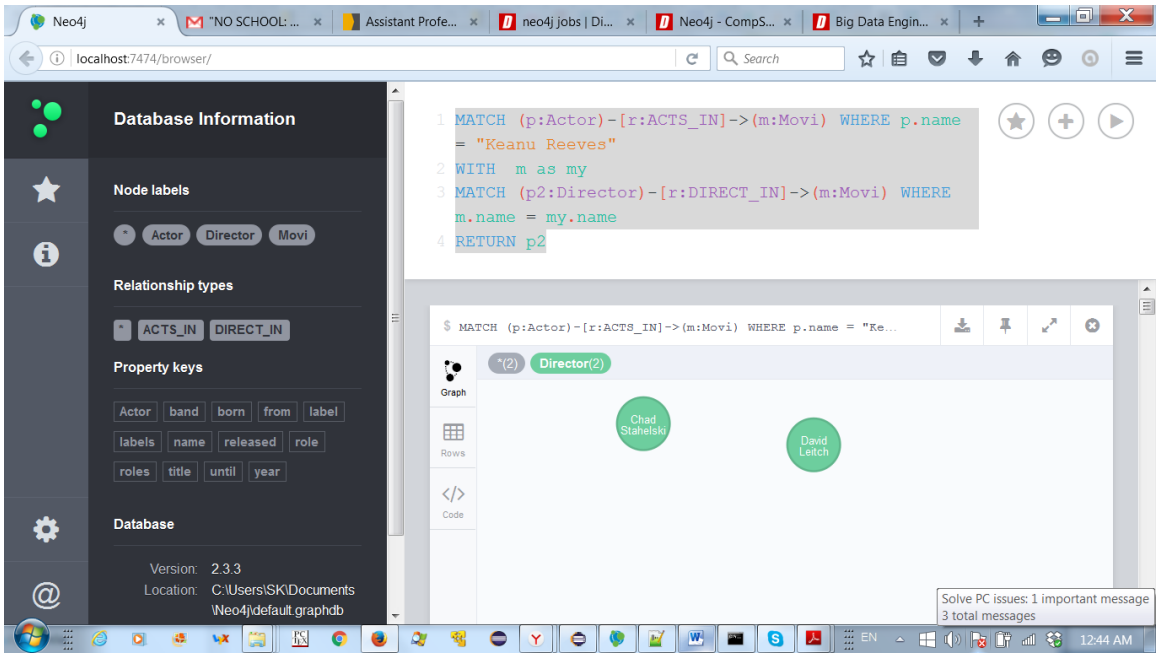
Answer to the question:

```
MATCH (p:Actor)-[r:ACTS_IN]->(m:Movi) WHERE p.name = "Keanu Reeves"
WITH  m as my
MATCH (p2:Actor)-[r:ACTS_IN]->(m:Movi) WHERE m.name = my.name
RETURN p2
```



Find directors of movies in which K. Reeves played.

```
MATCH (p:Actor)-[r:ACTS_IN]->(m:Movi) WHERE p.name = "Keanu Reeves"
WITH  m as my
MATCH (p2:Director)-[r:DIRECT_IN]->(m:Movi) WHERE m.name = my.name
RETURN p2
```



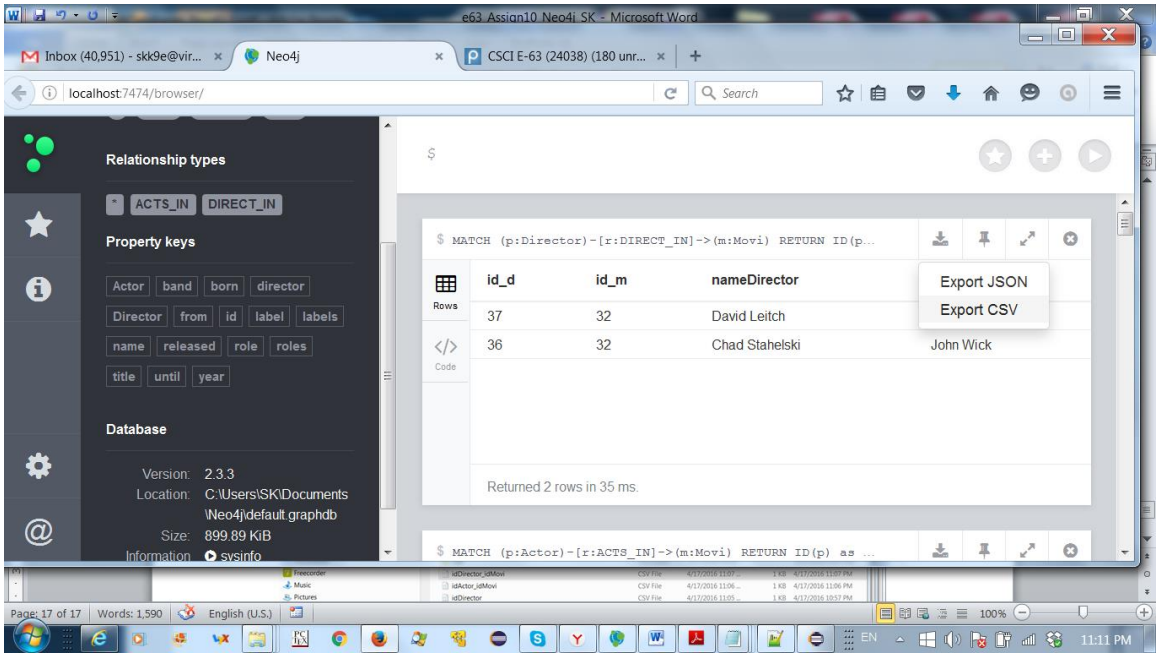
Problem 4. Find a way to export data from Neo4j into **a set of CSV files**. Delete your database and demonstrate that you can recreate it by loading those CSV files.

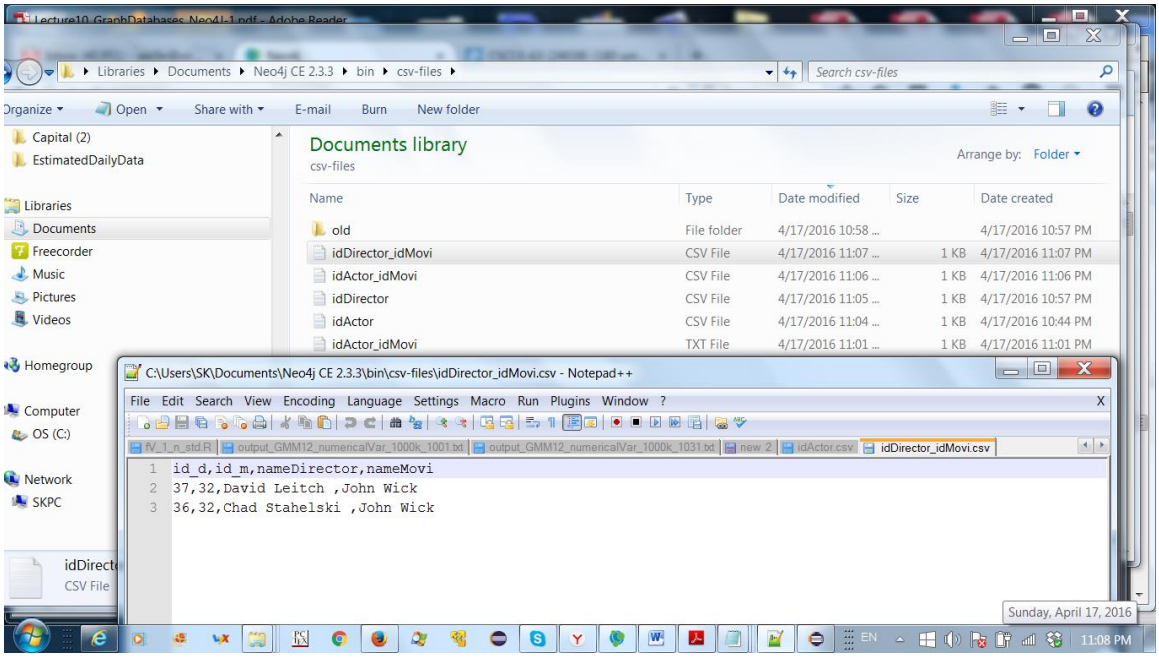
To export the tables in csv file

1. Create table using queries like (one by one):

```
MATCH (n:Actor) RETURN ID(n) as id_a , n.name as nameActor

MATCH(n:Director) RETURN ID(n) as id_d, n.name as nameDirector


MATCH (p:Actor)-[r:ACTS_IN]->(m:Movi)
RETURN ID(p) as id_a , ID(m) as id_m, p.name as nameActor, m.name as nameMovi

MATCH (p:Director)-[r:DIRECT_IN]->(m:Movi)
RETURN ID(p) as id_d , ID(m) as id_m, p.name as nameDirector, m.name as nameMovi
```

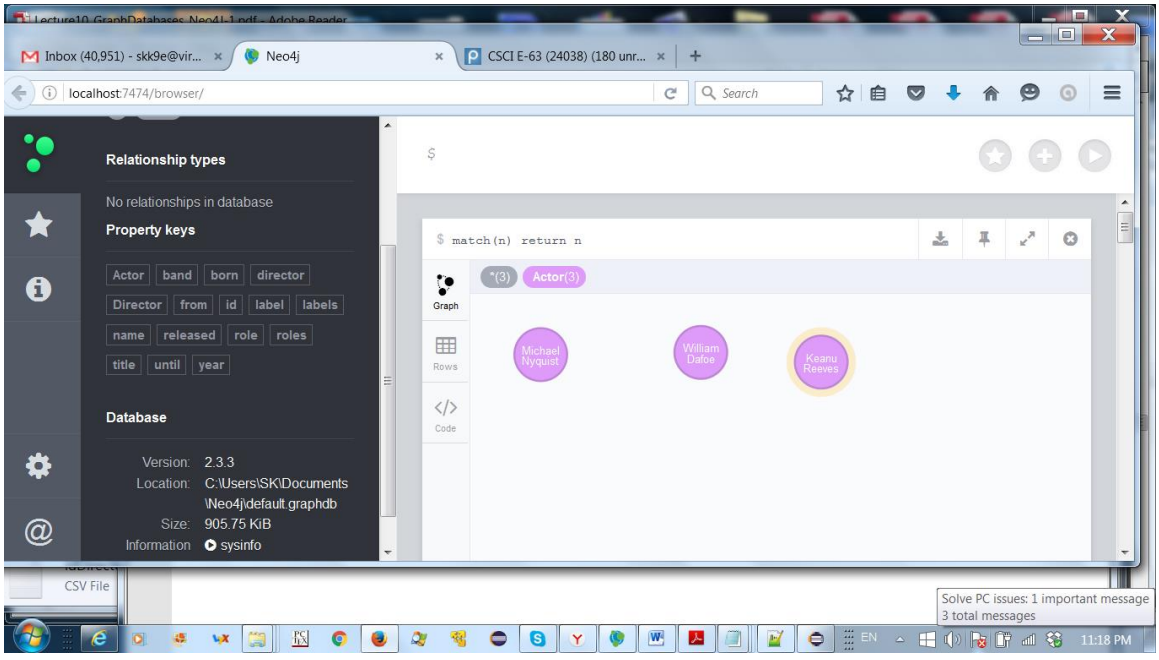
And after each query export the result in a csv file using Export CSV option



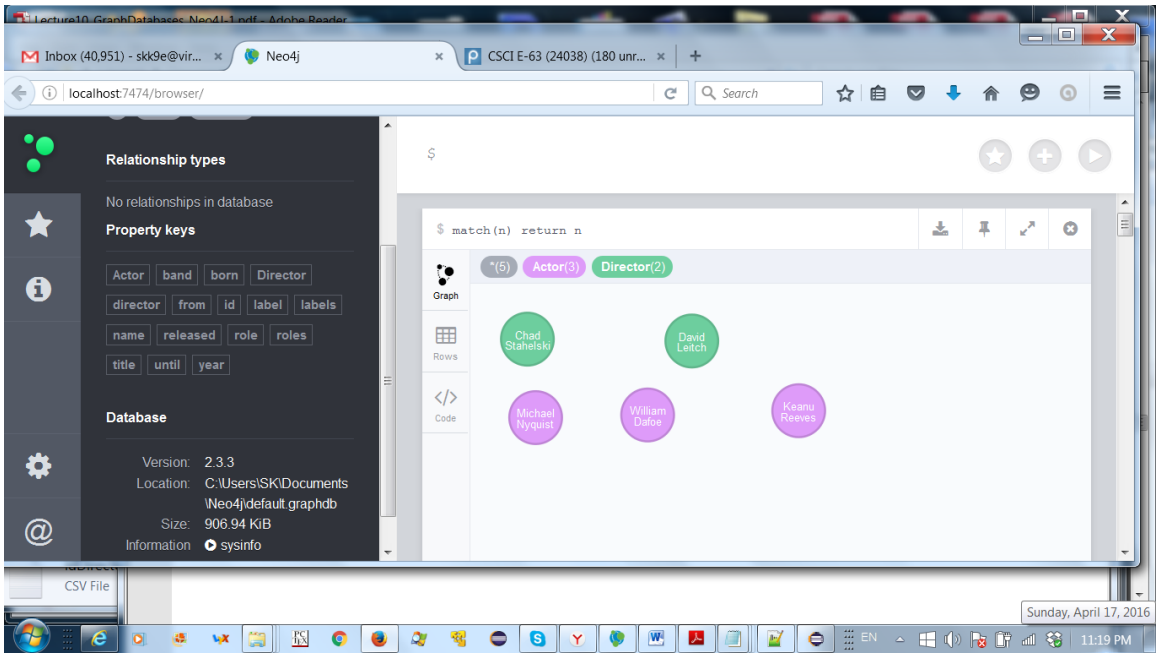


To import file

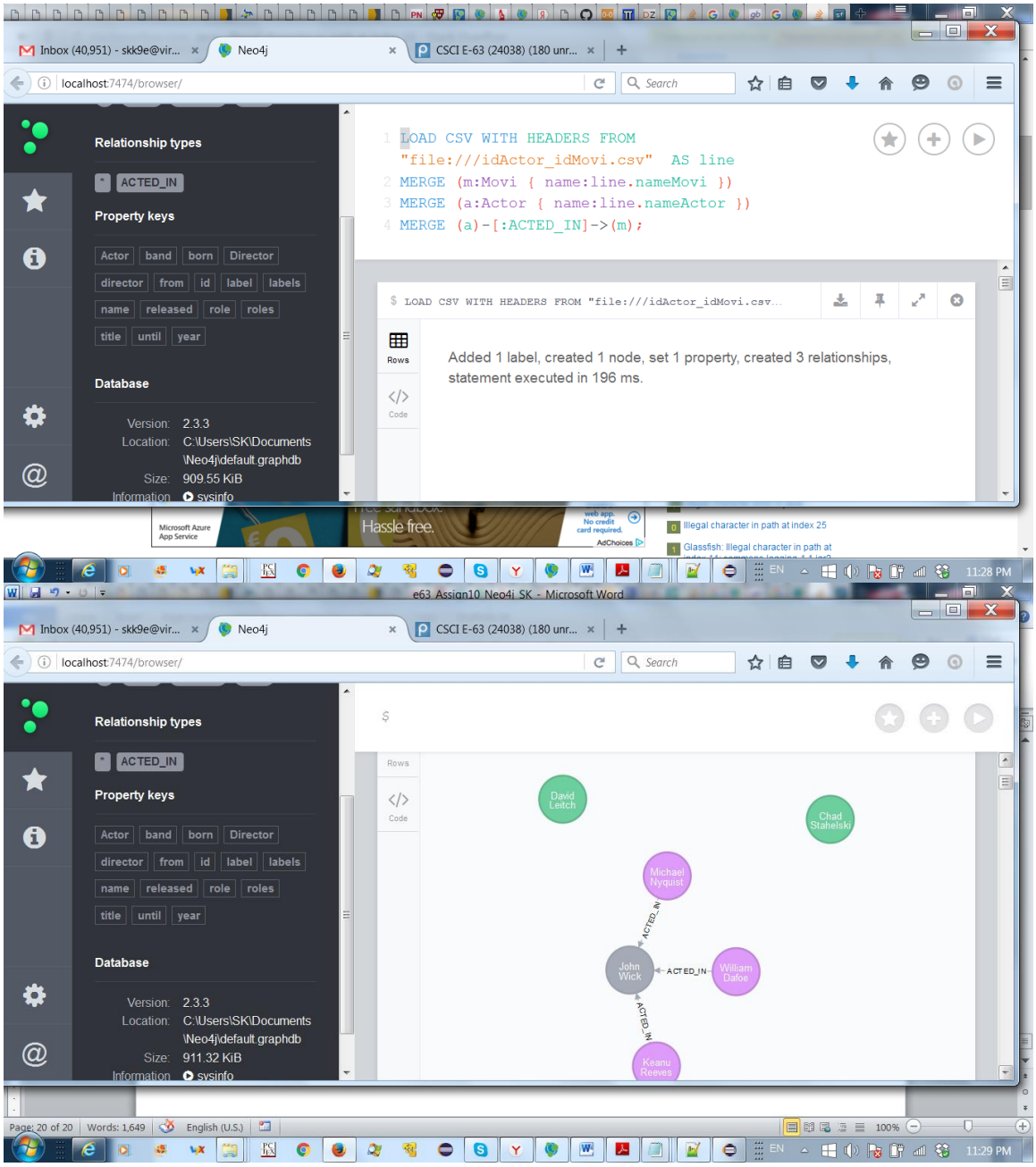
```
LOAD CSV WITH HEADERS FROM "file:///idActor.csv" AS line
MERGE (a:Actor { id:line.id_a })
ON CREATE SET a.name=line.nameActor;
```



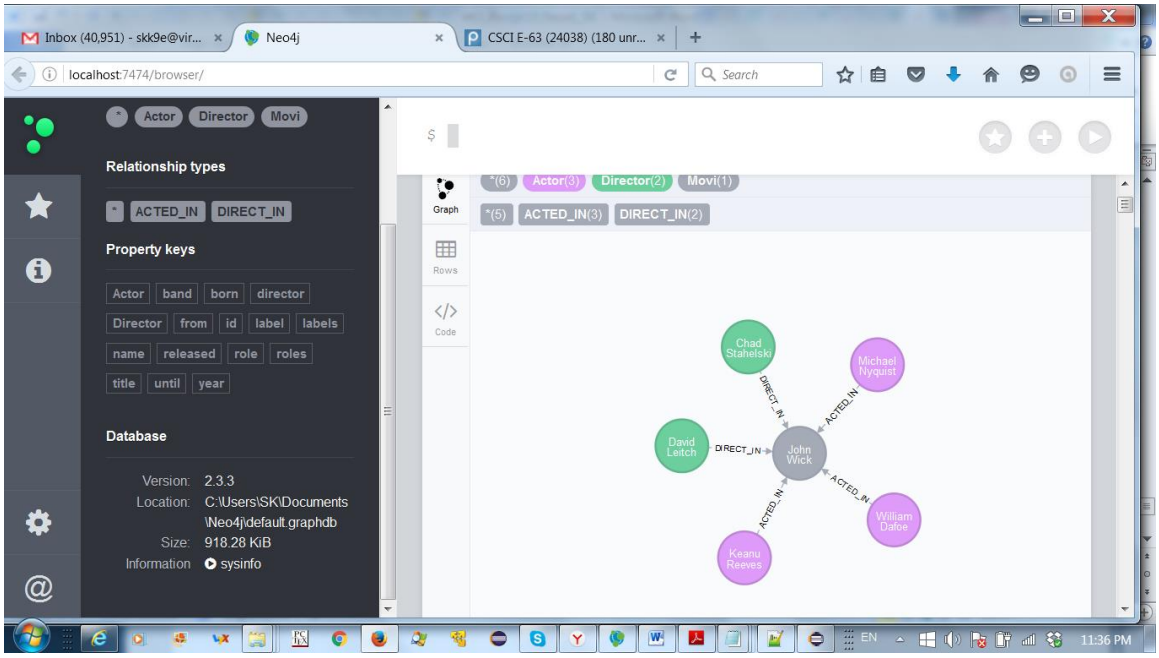
```
LOAD CSV WITH HEADERS FROM "file:///idDirector.csv" AS line
MERGE (a:Director { id:line.id_d })
ON CREATE SET a.name=line.nameDirector;
```




```
LOAD CSV WITH HEADERS FROM "file:///idActor_idMovi.csv"  AS line
MERGE (m:Movi { name:line.nameMovi })
MERGE (a:Actor { name:line.nameActor })
MERGE (a)-[:ACTED_IN]->(m);
```



LOAD CSV WITH HEADERS FROM "**file:///idDirector_idMovi.csv**" AS line
MERGE (m:Movi { name:line.nameMovi})
MERGE (a:Director { name:line. nameDirector})
MERGE (a)-[: DIRECT_IN]->(m);


Inbox (40,951) - skk9e@vir...
Neo4j
CSCI E-63 (24038) (180 unr...
localhost:7474/browser/
Search
Actor
Director
Movi
Relationship types
ACTED_IN
DIRECT_IN
Property keys
Actor
band
born
director
Director
from
id
label
labels
name
released
role
roles
title
until
year
Database
Version: 2.3.3
Location: C:\Users\SK\Documents\Neo4j\default.graphdb
Size: 918.28 KiB
Information sysinfo
Graph
Rows
Code
*(6)
Actor(3)
Director(2)
Movi(1)
*(5)
ACTED_IN(3)
DIRECT_IN(2)
Chad Stahelski
Michael Nyquist
David Leitch
John Wick
William Dafoe
Keanu Reeves
DIRECT_IN
ACTED_IN
11:36 PM