## HU Extension Assignment 02 E63 Big Data Analytics

### Handed out: 09/08/2017 Due by 11:59 AM, Saturday, 09/16/2017

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 on Piazza.

You can do most of this assignment in Python, Java, R, Scala or any other language of your convenience.

**Problem 1.** The following is the content ofMovies database.Bring that database into Neo4J using curl.

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

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

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

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

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)

You might want to create a bash script which contains curl command with previous Cypher request and run that script from Cygwin or Linux (Unix) prompt. Following notes could be helpful:

1. you might want to use notepad++ to create and edit the bash file, and go to edit --> EOL conversion --> Unix.
2. Specify the user name and password like “-user neo4j:neo4jneo4j" before the local host url.
3. Add one space and one \ at each point you want to have a line break.
4. In the string of query, every double quote " should be changed to \".

(20%)

Wrote up a shell script with these curl commands

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (matrix:Movie {title : \"The Matrix\", year : '1999-03-31' }) return id(matrix)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (matrix2:Movie { title : \"The Matrix Reloaded\", year : '2003-05-07' }) return id(matrix2)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (matrix3:Movie { title : \"The Matrix Revolutions\", year : '2003-10-27' }) return id(matrix3)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (keanu:Actor { name:\"Keanu Reeves\" }) return id(keanu)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (laurence:Actor { name:\"Laurence Fishburne\" })"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (carrieanne:Actor { name:\"Carrie-Anne Moss\" })"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (keanu)-[:ACTS\_IN { role : \"Neo\" }]->(matrix1)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (keanu)-[:ACTS\_IN { role : \"Neo\" }]->(matrix2)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (keanu)-[:ACTS\_IN { role : \"Neo\" }]->(matrix3)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (laurence)-[:ACTS\_IN { role : \"Morpheus\" }]->(matrix1)"}]}'**

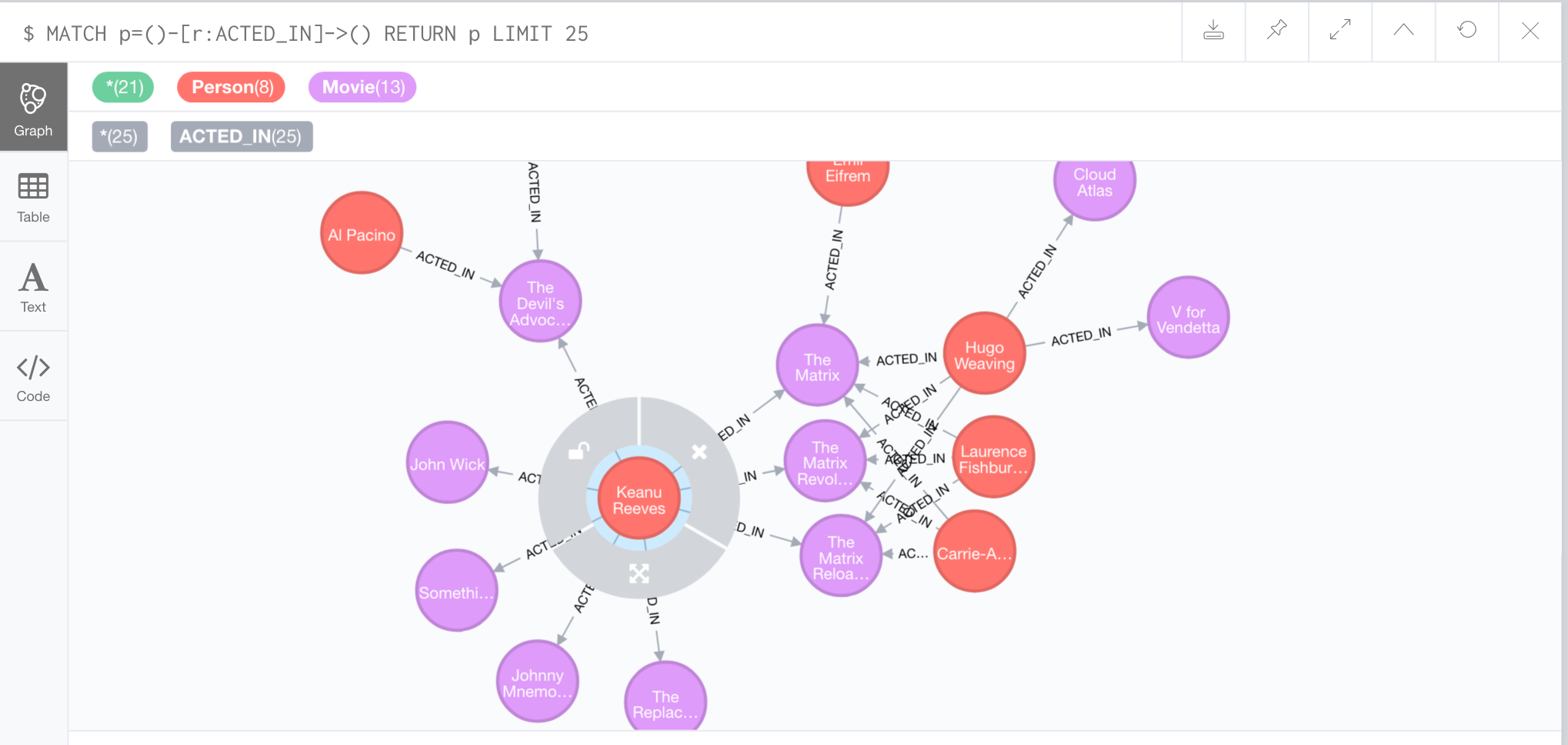
**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (laurence)-[:ACTS\_IN { role : \"Morpheus\" }]->(matrix2)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (laurence)-[:ACTS\_IN { role : \"Morpheus\" }]->(matrix3)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (carrieanne)-[:ACTS\_IN { role : \"Trinity\" }]->(matrix1)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (carrieanne)-[:ACTS\_IN { role : \"Trinity\" }]->(matrix2)"}]}'**

**curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (carrieanne)-[:ACTS\_IN { role : \"Trinity\" }]->(matrix3)"}]}'**



**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. Demonstrate that you have successfully brought data about John Wick movie into the database. You can use Cypher Browser or any other means. Delete above movie and all the cast except Keanu Reeves.

(15%)

I created the movie

CREATE (johnWick:Movie{title:'John Wick'}) return id(johnWick)

MATCH (m:Movie {title: 'John Wick'}) return m.title

╒═══════════╕

│"m.title" │

╞═══════════╡

│"John Wick"│

└───────────┘

Using Cypher

I first create the actors and directors

CREATE (p:Person{name:"Chad Stahelski"}) return id(p)

CREATE (p:Person{name:"David Leitch"}) return id(p)

CREATE (p:Person{name:"William Dafoe"}) return id(p)

CREATE (p:Person{name:"Michael Nyquist"}) return id(p)

Then I create the relationships for the directors and actors

MATCH (m:Movie { title:"John Wick" })

MATCH (p2:Person { name:"David Leitch" })

CREATE (p2)-[r2:DIRECTED]->(m)

MATCH (m:Movie { title:"John Wick" })

MATCH (p2:Person { name:"Chad Stahelski" })

CREATE (p2)-[r2:DIRECTED]->(m)

MATCH (m:Movie { title:"John Wick" })

MATCH (p1:Person { name:"William Dafoe" })

CREATE (p1)-[r1:ACTED\_IN]->(m)

MATCH (m:Movie { title:"John Wick" })

MATCH (p1:Person { name:"Michael Nyquist" })

CREATE (p1)-[r1:ACTED\_IN]->(m)

— Validate

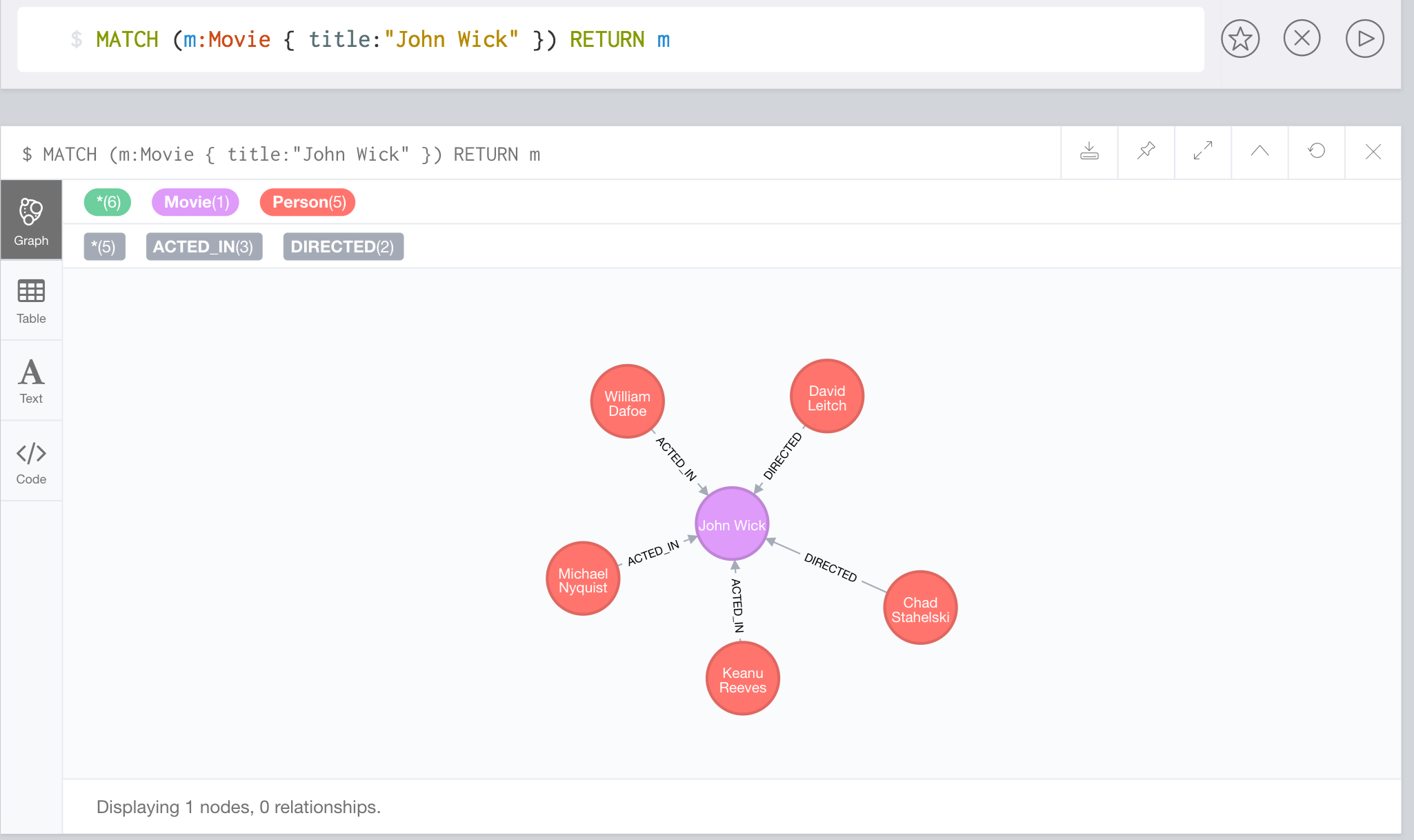
MATCH (p2:Person { name:"William Dafoe" }) RETURN p2

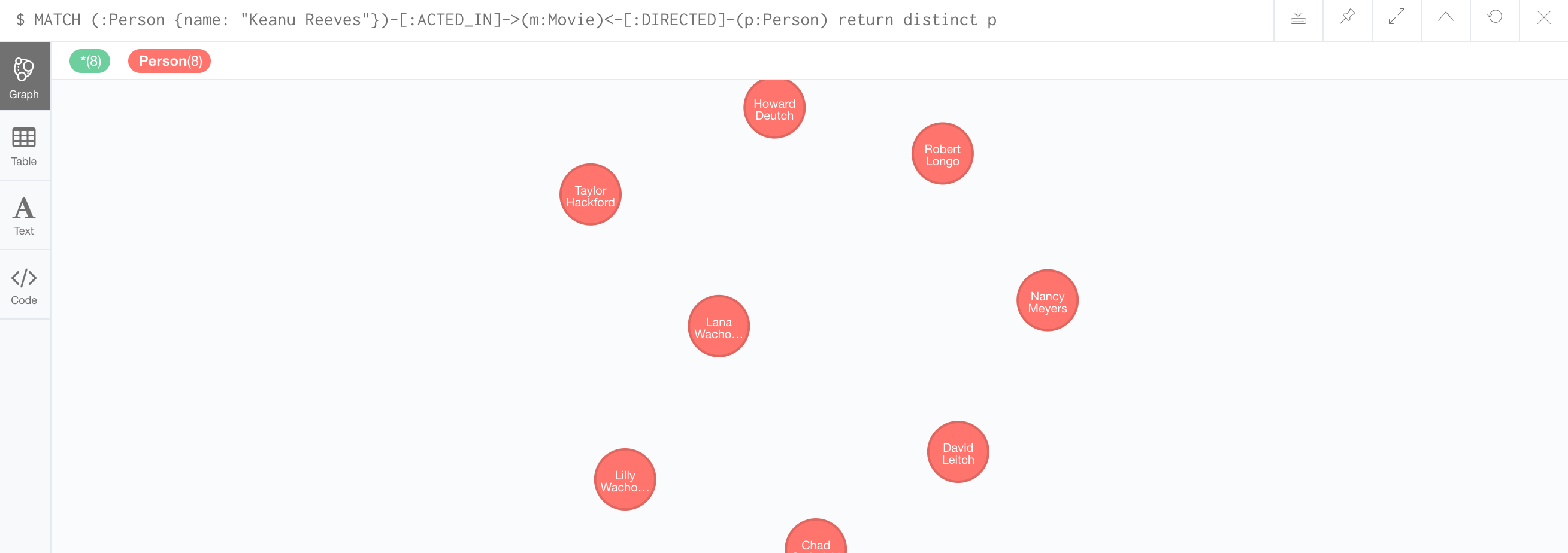
MATCH (p2:Person { name:"Michael Nyquist" }) RETURN p2

MATCH (p2:Person { name:"David Leitch" }) RETURN p2

MATCH (p2:Person { name:"Chad Stahelski" }) RETURN p2

MATCH (m:Movie { title:"John Wick" }) RETURN m





---Rollback

MATCH (p2:Person { name:"William Dafoe" }) DETACH DELETE p2

MATCH (p2:Person { name:"Michael Nyquist" }) DETACH DELETE p2

MATCH (p2:Person { name:"David Leitch" }) DETACH DELETE p2

MATCH (p2:Person { name:"Chad Stahelski" }) DETACH DELETE p2

MATCH (m:Movie { title:"John Wick" }) DETACH DELETE m

**Problem 3.** Addall the actors and the roles they played in this movie “John Wick” to the database using JAVA REST API or some other APIs for Neo4J in a language of your choice (not Curl). Demonstrate that you have successfully brought data about John Wick movie into the database. You can use Cypher Browser or any other means.

(15%)

\*\*\*Java Code\*\*\*

*/\*\**

*\* Created by smukherjee on 9/11/17.*

*\*/*

import org.neo4j.driver.v1.\*;

import static org.neo4j.driver.v1.Values.*parameters*;

public class Assignment2 {

static Session *session*;

public static void main(String[] args) {

*session* = *getNeo4JConnection*();

*createMovie*("John Wick");

*addPerson*("Chad Stahelski");

*addPerson*("David Leitch");

*addPerson*("William Dafoe");

*addPerson*("Michael Nyquist");

*addActor*("Keanu Reeves","John Wick");

*addActor*("William Dafoe","John Wick");

*addActor*("Michael Nyquist","John Wick");

*addDirector*("David Leitch","John Wick");

*addDirector*("Chad Stahelski","John Wick");

System.*exit*(1);

}

public static void addPerson(String name) {

*session*.run( "CREATE (p:Person{name:{name}})",

*parameters*( "name", name) );

}

public static void createMovie(String movieName) {

*session*.run( "CREATE (m:Movie{title:{title}})",

*parameters*( "title", movieName) );

}

public static void addActor(String person, String movie) {

*session*.run( "MATCH (m:Movie { title:{movie} })\n" +

"MATCH (p1:Person { name:{person} })\n" +

"CREATE (p1)-[r1:ACTED\_IN]->(m)",

*parameters*( "person",person,"movie",movie) );

}

public static void addDirector(String person, String movie) {

*session*.run( "MATCH (m:Movie { title:{movie} })\n" +

"MATCH (p1:Person { name:{person} })\n" +

"CREATE (p1)-[r1:DIRECTED]->(m)",

*parameters*( "person",person,"movie",movie) );

}

public static Session getNeo4JConnection() {

Driver driver = GraphDatabase.*driver*( "bolt://localhost:7687");

Session session = driver.session();

return session;

}

}

\*\*\*\*\*pom

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>assignment\_neo4j</groupId>

<artifactId>assignment\_neo4j</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.neo4j.driver</groupId>

<artifactId>neo4j-java-driver</artifactId>

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>org.neo4j</groupId>

<artifactId>neo4j-rest-graphdb</artifactId>

<version>1.8.1</version>

</dependency>

</dependencies>

<repositories>

<repository>

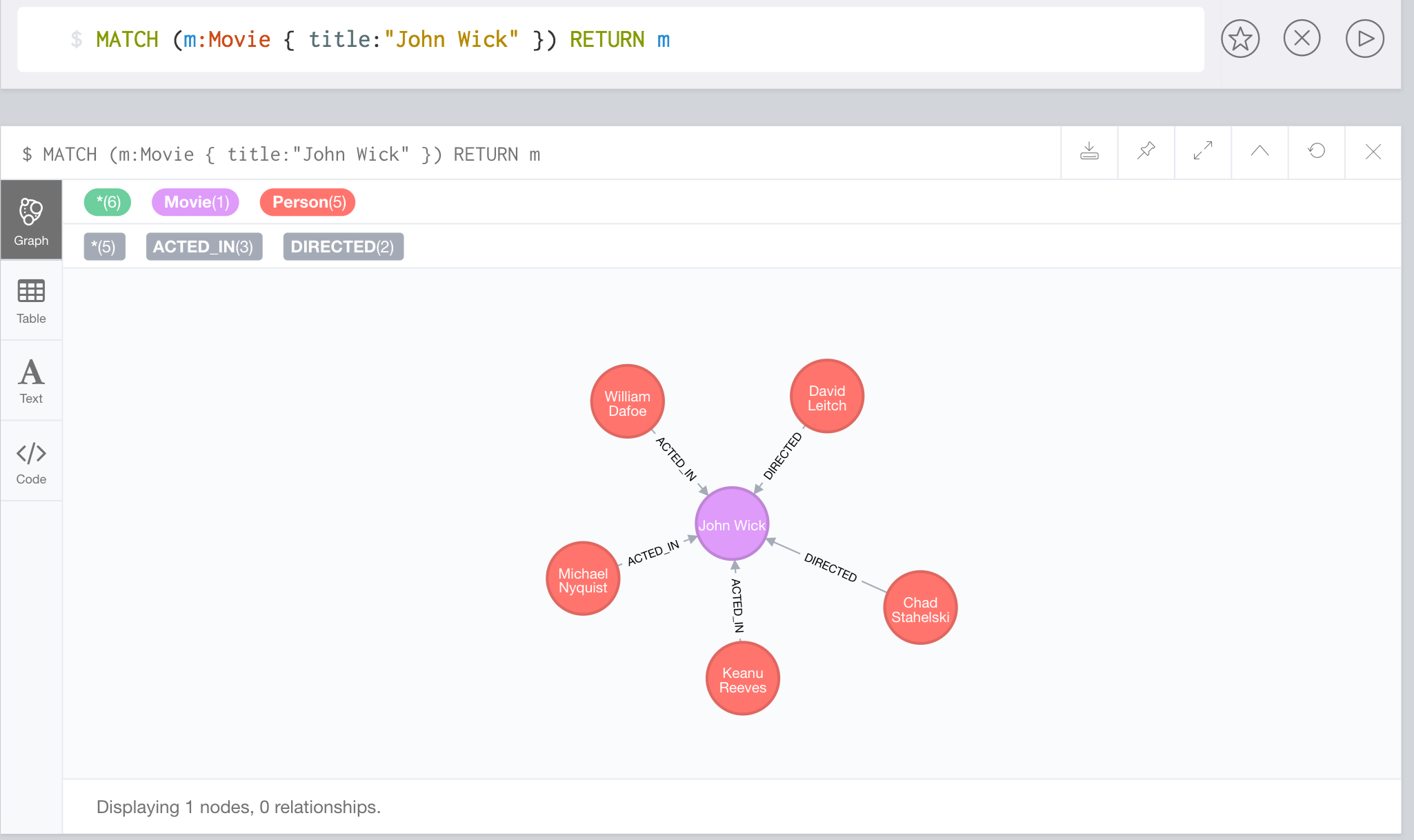
<id>mvn-neo4j</id>

<url>http://m2.neo4j.org/content/groups/everything</url>

</repository>

</repositories>

</project>



**Problem 4**. Find a list of actors playing in movies in which Keanu Reeves played. Find directors of movies in which K. Reeves played. Please use any language of your convenience (Java, Python, C#, R, curl). Verify your results using Cypher queries in Cypher Browser

(15%)

public static List<Record> findCostarsForActorMovies(String actor) {

List<String> resultList = new ArrayList<String>();

StatementResult result= *session*.run( "MATCH (:Person {name: {person}})" +

"-[:ACTED\_IN]->(m:Movie)" +

"<-[:ACTED\_IN]-(p:Person) return distinct p.name",

*parameters*( "person",actor) );

return result.list();

}

public static List<Record> findDirectorsForActorMovies(String actor) {

StatementResult result = *session*.run( "MATCH (:Person {name: {person}})" +

"-[:ACTED\_IN]->(m:Movie)" +

"<-[:DIRECTED]-(p:Person) return distinct p.name",

*parameters*( "person",actor)) ;

return result.list();

}

MATCH (:Person {name: "Keanu Reeves"})-[:ACTED\_IN]->(m:Movie)<-[:DIRECTED]-(p:Person) return distinct p

╒══════════════════════════════════════╕

│"p" │

╞══════════════════════════════════════╡

│{"name":"Chad Stahelski"} │

├──────────────────────────────────────┤

│{"name":"David Leitch"} │

├──────────────────────────────────────┤

│{"born":1949,"name":"Nancy Meyers"} │

├──────────────────────────────────────┤

│{"born":1953,"name":"Robert Longo"} │

├──────────────────────────────────────┤

│{"born":1950,"name":"Howard Deutch"} │

├──────────────────────────────────────┤

│{"born":1944,"name":"Taylor Hackford"}│

├──────────────────────────────────────┤

│{"born":1965,"name":"Lana Wachowski"} │

├──────────────────────────────────────┤

│{"born":1967,"name":"Lilly Wachowski"}│

└──────────────────────────────────────┘

MATCH (:Person {name: "Keanu Reeves"})-[:ACTED\_IN]->(m:Movie)<-[:ACTED\_IN]-(p:Person) return distinct p

"p" │

╞═════════════════════════════════════════╡

│{"name":"Michael Nyquist"} │

├─────────────────────────────────────────┤

│{"name":"William Dafoe"} │

├─────────────────────────────────────────┤

│{"born":1937,"name":"Jack Nicholson"} │

├─────────────────────────────────────────┤

│{"born":1946,"name":"Diane Keaton"} │

├─────────────────────────────────────────┤

│{"born":1968,"name":"Dina Meyer"} │

├─────────────────────────────────────────┤

│{"born":1958,"name":"Ice-T"} │

**Problem 5**. Find a way to export data from Neo4j into a set of CSV files. Delete your database and demonstrate that you can recreate the database by loading those CSV files. Please use any programming language of your convenience: Java, Python, R, C# or Scala.

(20%)

I downloaded apoc in plugins

I set apoc.export.file.enabled=true in neo4j.conf

I had to copy it here too

/Applications/Neo4j Community Edition 3.2.3.app/Contents/Resources/app/plugins >cp /Users/smukherjee/Documents/Neo4j/default.graphdb/plugins/apoc-3.2.0.4-all.jar .

**//Neo4j connection**

public static Session getNeo4JConnection() {

Driver driver = GraphDatabase.*driver*( "bolt://localhost:7687");

Session session = driver.session();

return session;

}

**//export data from neo4j to cvs files**

static void writeCSV(){

*session*.run("CALL apoc.export.csv.query(\"MATCH (p:Person) RETURN id(p) as id,p.name as name,p.born as born\", '/tmp/neo4j/data/persons.csv', {})");

*session*.run("CALL apoc.export.csv.query(\"MATCH (m:Movie) RETURN id(m) as id,m.tagline as tagline,m.title as title,m.released as released\", '/tmp/neo4j/data/movies.csv', {})");

*session*.run("CALL apoc.export.csv.query(\"MATCH (p:Person)-[act:ACTED\_IN]->(m:Movie) RETURN id(p) as personId, act.roles as role, id(m) as movieId\", '/tmp/neo4j/data/actorroles.csv', {})");

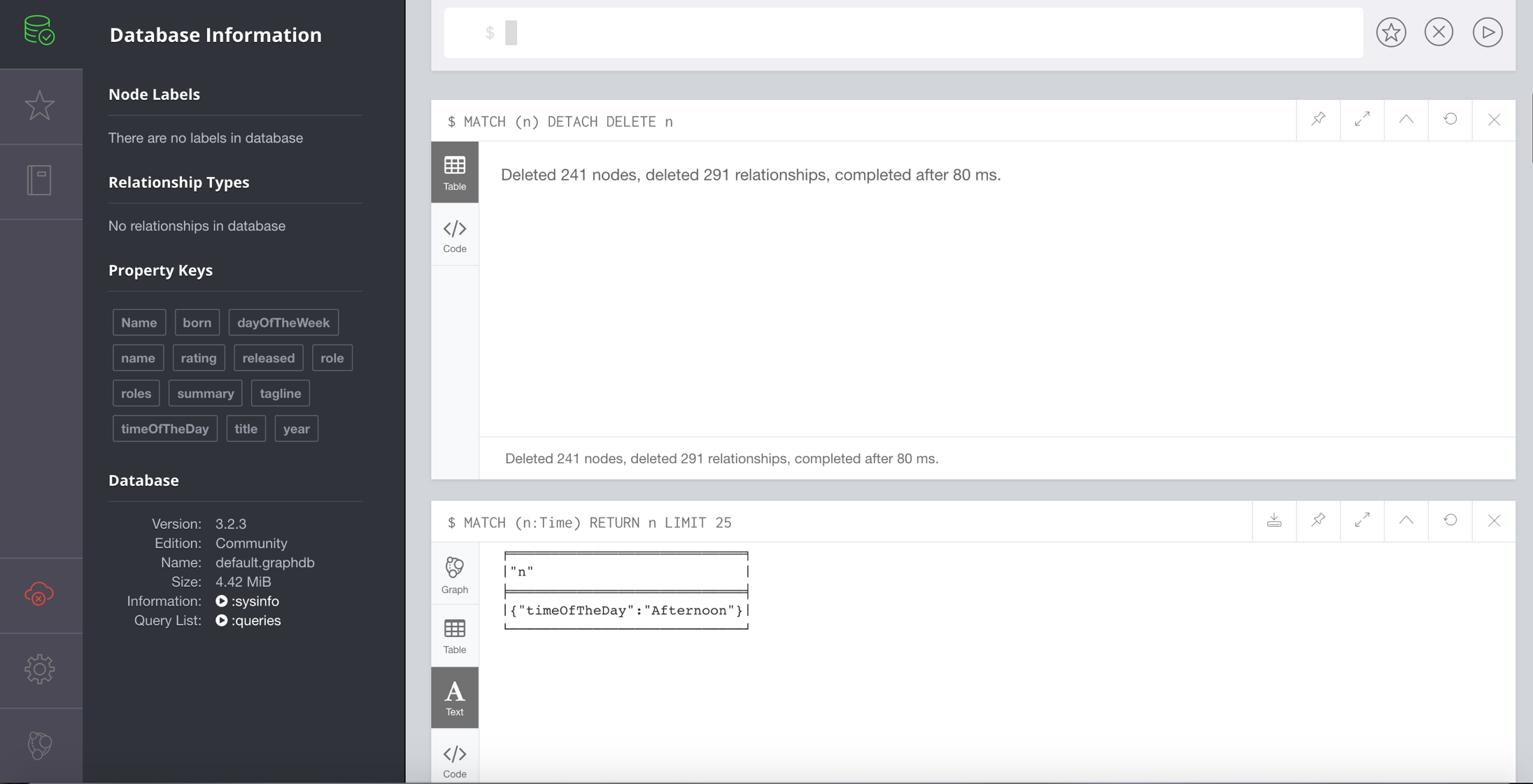
*session*.run("CALL apoc.export.csv.query(\"MATCH (p:Person)-[act:DIRECTED]->(m:Movie) RETURN id(p) as personId, id(m) as movieId\", '/tmp/neo4j/data/directorroles.csv', {})");

}

**//To Delete all nodes/relationships (ran in console)**

**MATCH (n)**

**DETACH DELETE n**

****

**//load the CSVs again to restor db**

static void loadCSV(){

*session*.run("USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM \"file:///persons.csv\" AS line\n" +

"MERGE (a:Person { id:line.id })\n" +

"ON CREATE SET a.name=line.name");

*session*.run("LOAD CSV WITH HEADERS FROM \"file:///movies.csv\" AS line\n" +

"CREATE (m:Movie { id:line.id,tagline:line.tagline,title:line.title, released:toInteger(line.released)})");

*session*.run("LOAD CSV WITH HEADERS FROM \"file:///actorroles.csv\" AS line\n" +

"MATCH (m:Movie { id:line.movieId })\n" +

"MATCH (a:Person { id:line.personId })\n" +

"CREATE (a)-[:ACTED\_IN { roles: [line.role]}]->(m)");

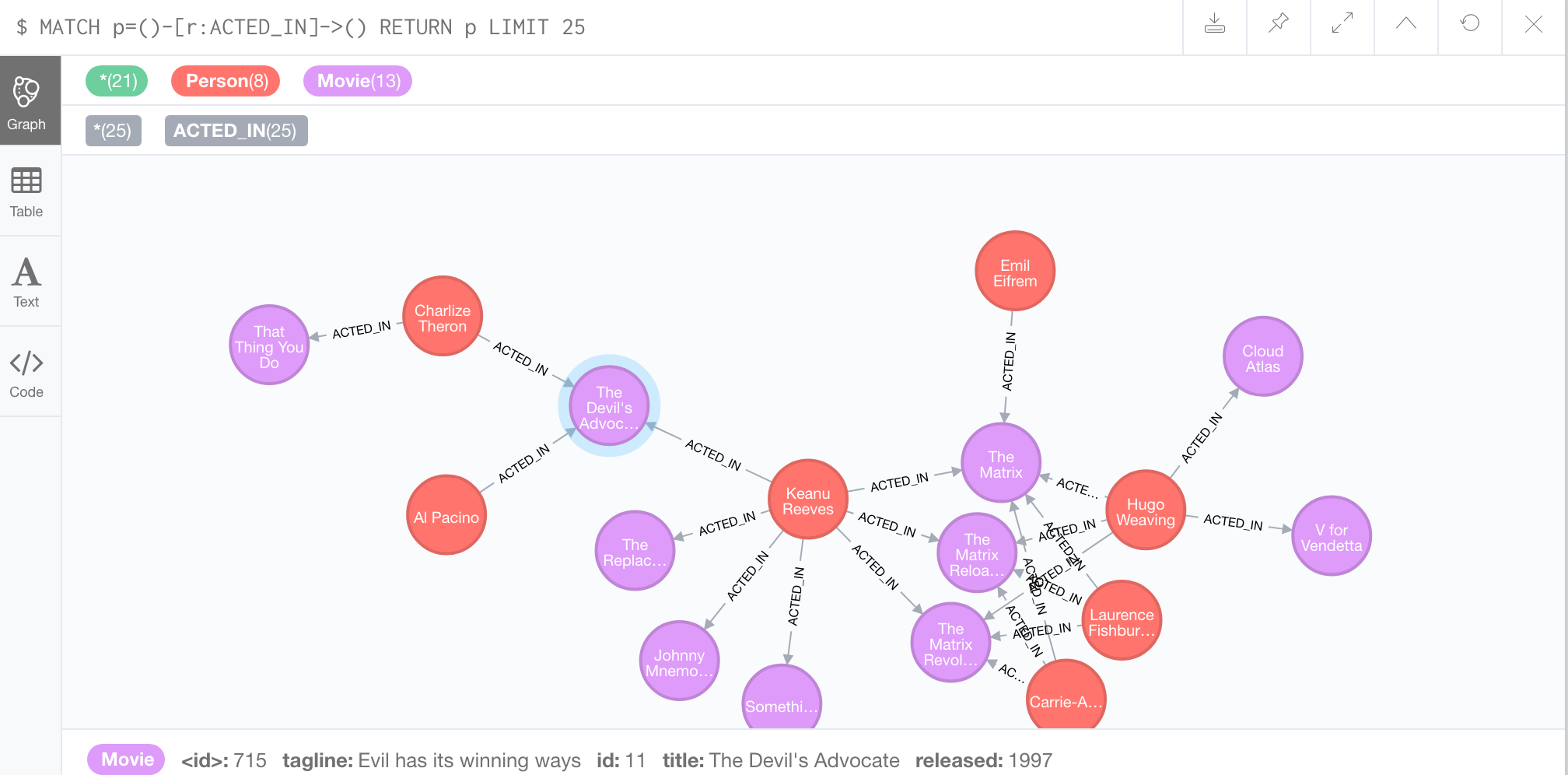
*session*.run("LOAD CSV WITH HEADERS FROM \"file:///directorroles.csv\" AS line\n" +

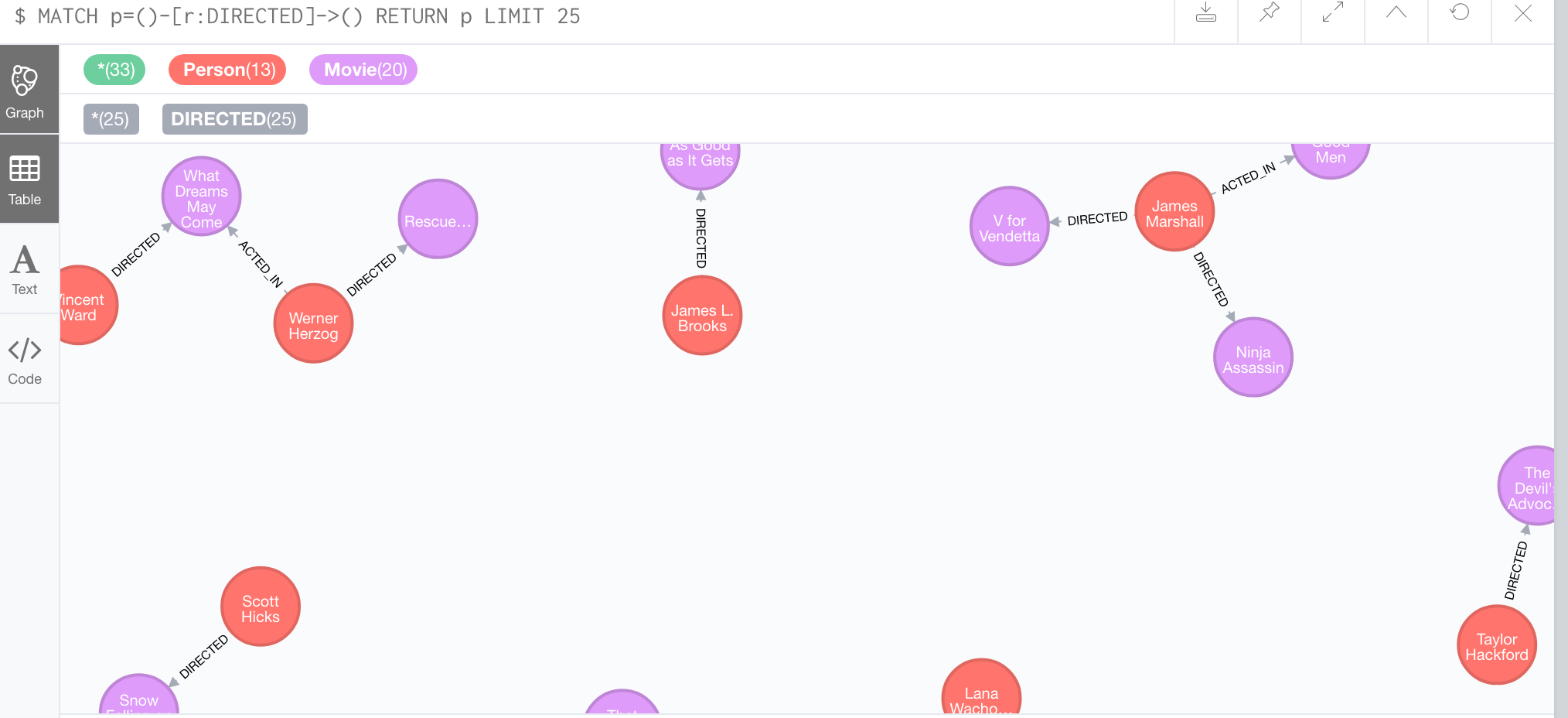
"MATCH (m:Movie { id:line.movieId })\n" +

"MATCH (a:Person { id:line.personId })\n" +

"CREATE (a)-[:DIRECTED]->(m)");

//After adding back

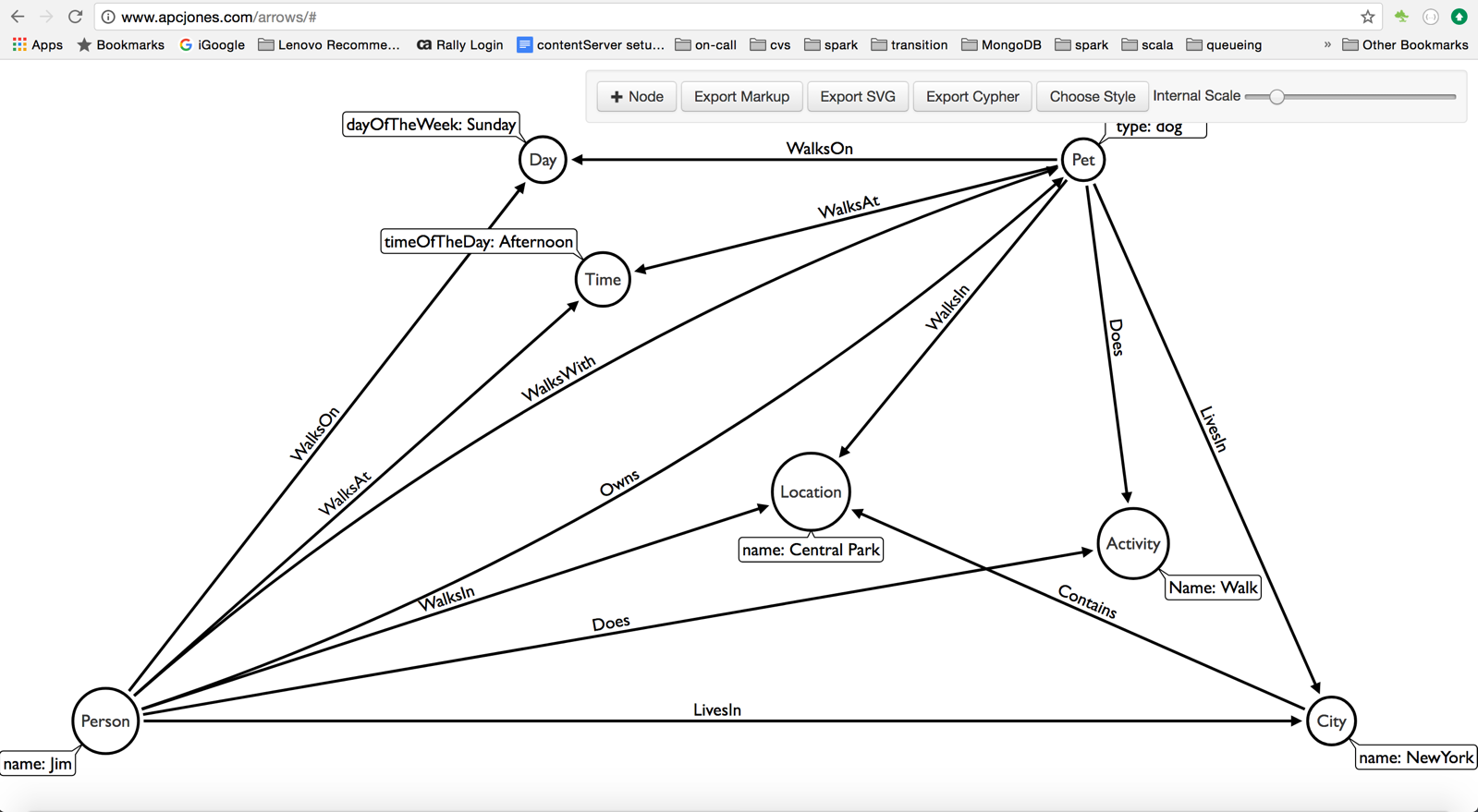


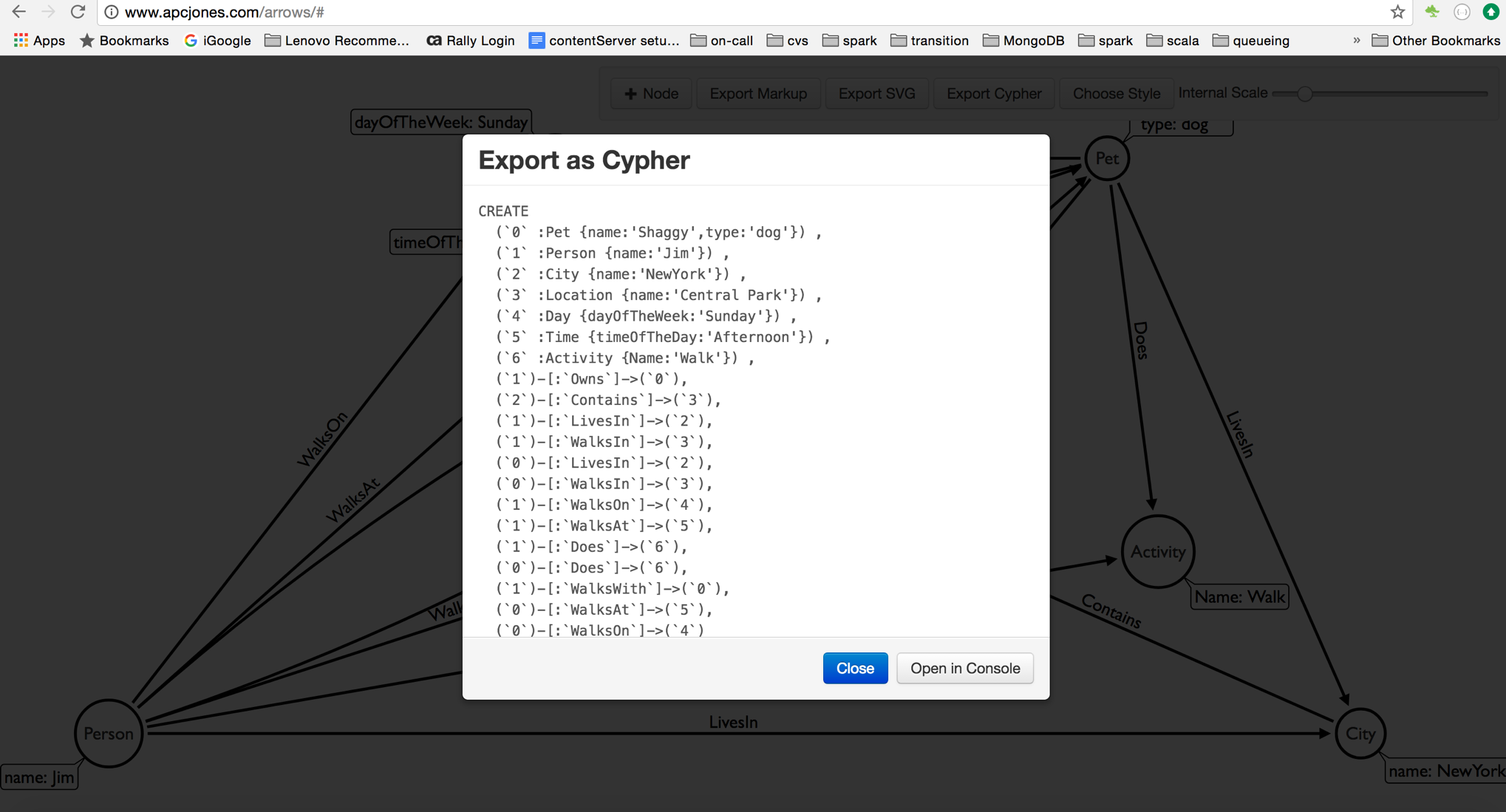


**Problem 6.** Find a way to use Arrow Tool (<http://www.apcjones.com>) to paint a relationship between a dog and his owner who live in New York and walk through the Central Park on Sunday afternoon. Add Labels and necessary properties to all nodes and relationships. Export your graph in Cypher format and then adjust (if necessary) generated Cypher so that you can create that graph in Neo4J database. Verify that your graph is indeed created using Cypher Browser.

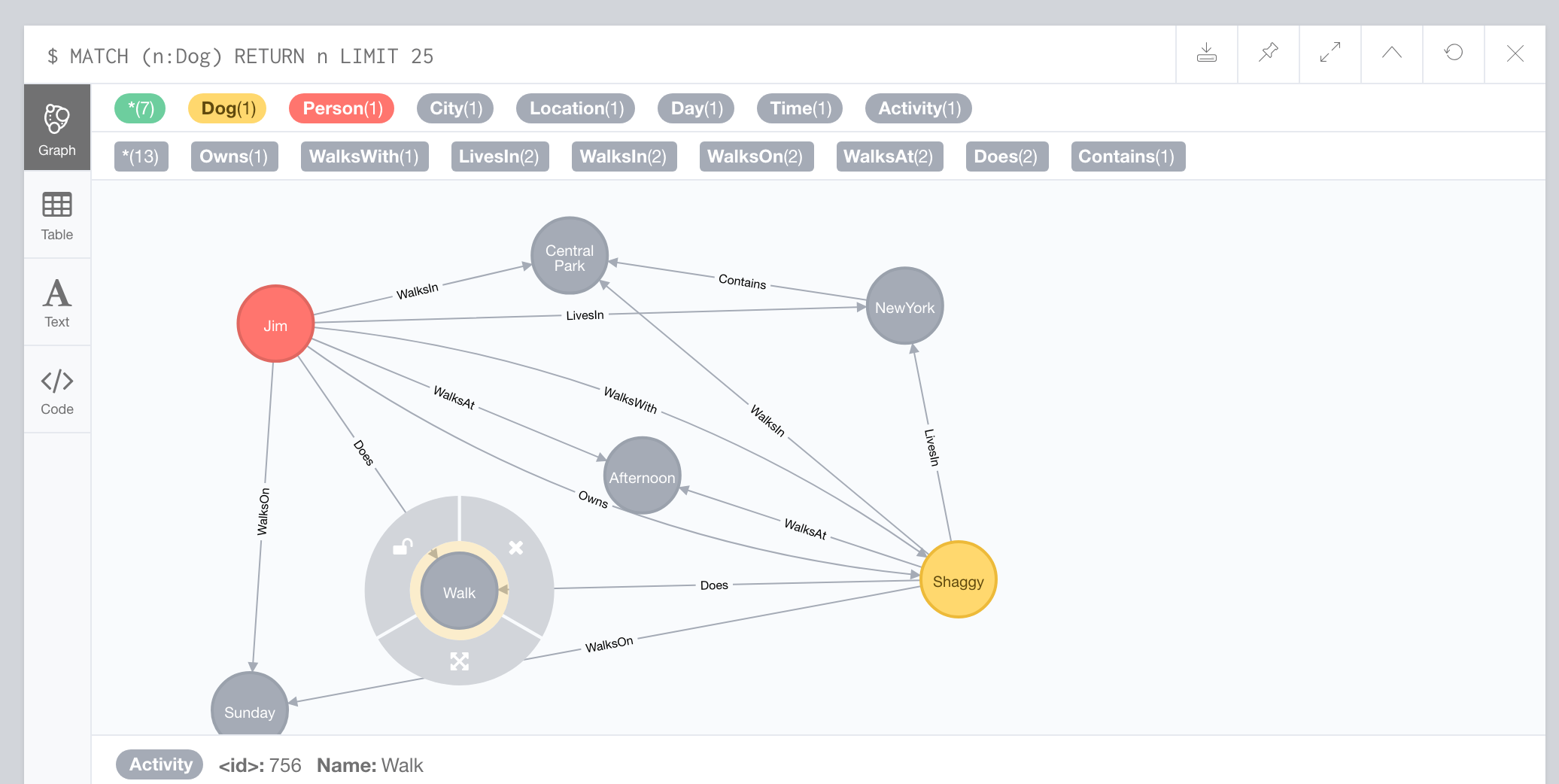
(15%)

I created the graph in Arrow Tool and then exported in Cypher format.I named the person Jim and the dog Shaggy.





CREATE (`0` :Dog {name:'Shaggy'}) , (`1` :Person {name:'Jim'}) , (`2` :City {name:'NewYork'}) , (`3` :Location {name:'Central Park'}) , (`4` :Day {dayOfTheWeek:'Sunday'}) , (`5` :Time {timeOfTheDay:'Afternoon'}) , (`6` :Activity {Name:'Walk'}) , (`1`)-[:Owns]->(`0`), (`2`)-[:`Contains`]->(`3`), (`1`)-[:LivesIn]->(`2`), (`1`)-[:WalksIn]->(`3`), (`0`)-[:LivesIn]->(`2`), (`0`)-[:WalksIn]->(`3`), (`1`)-[:WalksOn]->(`4`), (`1`)-[:WalksAt]->(`5`), (`1`)-[:Does]->(`6`), (`0`)-[:Does]->(`6`), (`1`)-[:WalksWith]->(`0`), (`0`)-[:`WalksAt`]->(`5`), (`0`)-[:`WalksOn`]->(`4`)



For Ref:

All Java Code together:

*/\*\*  
 \* Created by smukherjee on 9/11/17.  
 \*/*import org.neo4j.driver.v1.\*;  
  
import static org.neo4j.driver.v1.Values.*parameters*;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Assignment2 {  
  
 static Session *session*;  
  
 public static void main(String[] args) {  
 *session* = *getNeo4JConnection*();  
// createMovie("John Wick");  
// addPerson("Chad Stahelski");  
// addPerson("David Leitch");  
// addPerson("William Dafoe");  
// addPerson("Michael Nyquist");  
//  
// addActor("Keanu Reeves","John Wick");  
// addActor("William Dafoe","John Wick");  
// addActor("Michael Nyquist","John Wick");  
//  
// addDirector("David Leitch","John Wick");  
// addDirector("Chad Stahelski","John Wick");  
//  
// printOutPut(findCostarsForActorMovies("Keanu Reeves"));  
// printOutPut(findDirectorsForActorMovies("Keanu Reeves"));  
  
  
 *writeCSV*();  
 *loadCSV*();  
 System.*exit*(1);  
  
 }  
  
  
 static void printOutPut(List<Record> list){  
  
 for (Record s : list) {  
 System.*out*.println(s.get("p.name"));  
  
 }  
 }  
  
 public static void addPerson(String name) {  
 *session*.run( "CREATE (p:Person{name:{name}})",  
 *parameters*( "name", name) );  
  
  
 }  
  
 public static void createMovie(String movieName) {  
 *session*.run( "CREATE (m:Movie{title:{title}})",  
 *parameters*( "title", movieName) );  
  
 }  
  
  
 public static void addActor(String person, String movie) {  
 *session*.run( "MATCH (m:Movie { title:{movie} })\n" +  
 "MATCH (p1:Person { name:{person} })\n" +  
 "CREATE (p1)-[r1:ACTED\_IN]->(m)",  
 *parameters*( "person",person,"movie",movie) );  
  
 }  
  
 public static void addDirector(String person, String movie) {  
 *session*.run( "MATCH (m:Movie { title:{movie} })\n" +  
 "MATCH (p1:Person { name:{person} })\n" +  
 "CREATE (p1)-[r1:DIRECTED]->(m)",  
 *parameters*( "person",person,"movie",movie) );  
  
 }  
  
 public static List<Record> findCostarsForActorMovies(String actor) {  
 List<String> resultList = new ArrayList<String>();  
 StatementResult result= *session*.run( "MATCH (:Person {name: {person}})" +  
 "-[:ACTED\_IN]->(m:Movie)" +  
 "<-[:ACTED\_IN]-(p:Person) return distinct p.name",  
 *parameters*( "person",actor) );  
  
 return result.list();  
  
 }  
  
 public static List<Record> findDirectorsForActorMovies(String actor) {  
  
 StatementResult result = *session*.run( "MATCH (:Person {name: {person}})" +  
 "-[:ACTED\_IN]->(m:Movie)" +  
 "<-[:DIRECTED]-(p:Person) return distinct p.name",  
 *parameters*( "person",actor)) ;  
  
 return result.list();  
  
 }  
  
  
 public static Session getNeo4JConnection() {  
  
 Driver driver = GraphDatabase.*driver*( "bolt://localhost:7687");  
 Session session = driver.session();  
  
 return session;  
 }  
  
  
 static void writeCSV(){  
 *session*.run("CALL apoc.export.csv.query(\"MATCH (p:Person) RETURN id(p) as id,p.name as name,p.born as born\", '/tmp/neo4j/data/persons.csv', {})");  
 *session*.run("CALL apoc.export.csv.query(\"MATCH (m:Movie) RETURN id(m) as id,m.tagline as tagline,m.title as title,m.released as released\", '/tmp/neo4j/data/movies.csv', {})");  
 *session*.run("CALL apoc.export.csv.query(\"MATCH (p:Person)-[act:ACTED\_IN]->(m:Movie) RETURN id(p) as personId, act.roles as role, id(m) as movieId\", '/tmp/neo4j/data/actorroles.csv', {})");  
 *session*.run("CALL apoc.export.csv.query(\"MATCH (p:Person)-[act:DIRECTED]->(m:Movie) RETURN id(p) as personId, id(m) as movieId\", '/tmp/neo4j/data/directorroles.csv', {})");  
  
 }  
  
 static void loadCSV(){  
 *session*.run("USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM \"file:///persons.csv\" AS line\n" +  
 "MERGE (a:Person { id:line.id })\n" +  
 "ON CREATE SET a.name=line.name");  
  
 *session*.run("LOAD CSV WITH HEADERS FROM \"file:///movies.csv\" AS line\n" +  
 "CREATE (m:Movie { id:line.id,tagline:line.tagline,title:line.title, released:toInteger(line.released)})");  
  
  
 *session*.run("LOAD CSV WITH HEADERS FROM \"file:///actorroles.csv\" AS line\n" +  
 "MATCH (m:Movie { id:line.movieId })\n" +  
 "MATCH (a:Person { id:line.personId })\n" +  
 "CREATE (a)-[:ACTED\_IN { roles: [line.role]}]->(m)");  
  
  
 *session*.run("LOAD CSV WITH HEADERS FROM \"file:///directorroles.csv\" AS line\n" +  
 "MATCH (m:Movie { id:line.movieId })\n" +  
 "MATCH (a:Person { id:line.personId })\n" +  
 "CREATE (a)-[:DIRECTED]->(m)");  
  
 }  
  
}

pom

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>assignment\_neo4j</groupId>  
 <artifactId>assignment\_neo4j</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.neo4j.driver</groupId>  
 <artifactId>neo4j-java-driver</artifactId>  
 <version>1.2.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.neo4j</groupId>  
 <artifactId>neo4j-rest-graphdb</artifactId>  
 <version>1.8.1</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/net.sf.opencsv/opencsv -->  
 <dependency>  
 <groupId>com.opencsv</groupId>  
 <artifactId>opencsv</artifactId>  
 <version>3.8</version>  
 </dependency>  
  
 </dependencies>  
  
 <repositories>  
 <repository>  
 <id>mvn-neo4j</id>  
 <url>http://m2.neo4j.org/content/groups/everything</url>  
 </repository>  
 </repositories>  
  
</project>