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 of Movies database. Bring that database into Neo4J using curl.

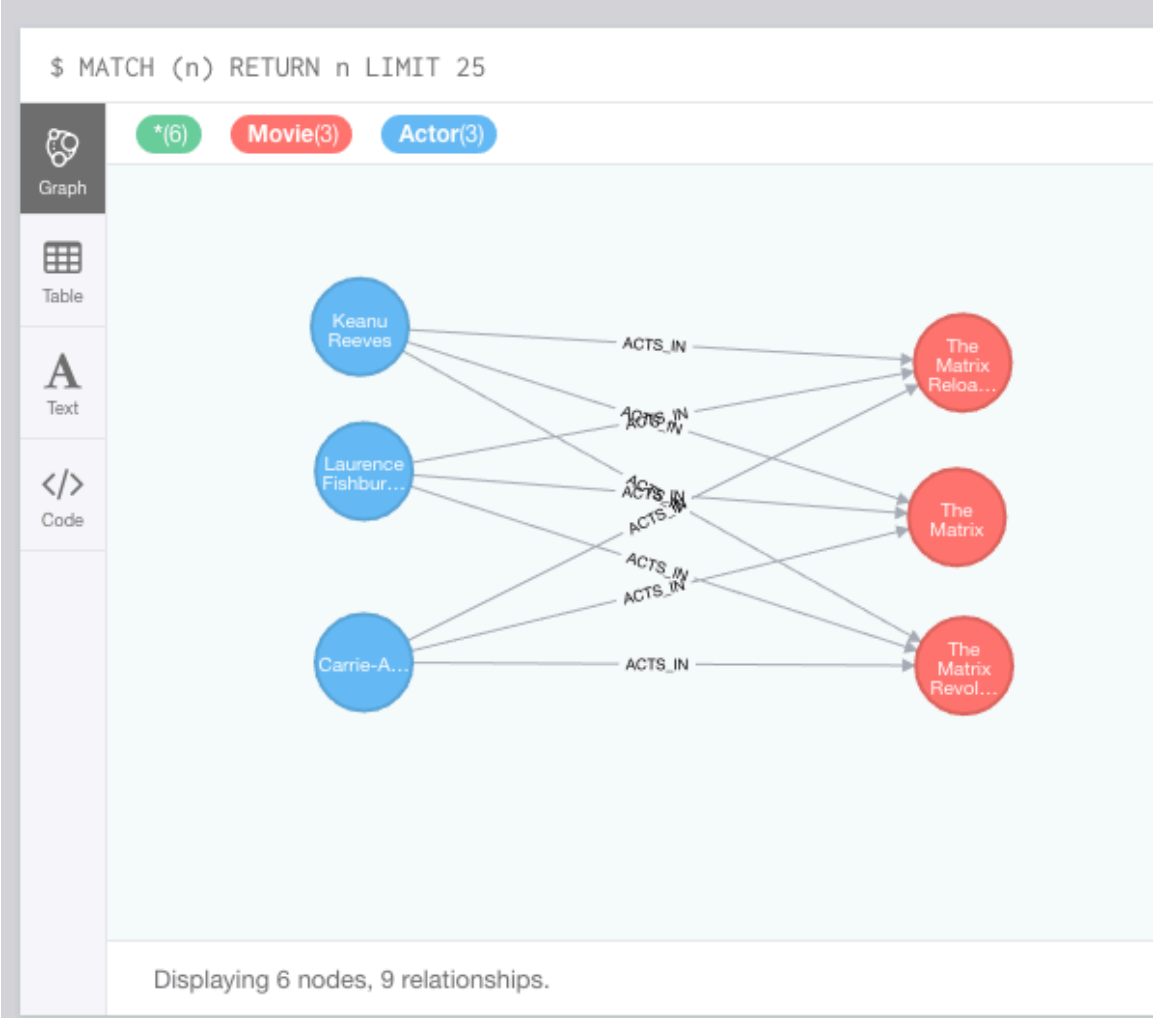
```
CREATE (matrix1:Movie { title : 'The Matrix', year : '1999-03-31' })
return id(matrix1)

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%)

```
swaite@Rmt-mac-swaite:~/stirling/CSIE-63/assignment-2|master⚡
⇒  ./ps1.sh
Usage load_json.sh 'http://json.api.com?params=values'
import_json.cypher
Use {data} as parameter in your query for the JSON data
{"results":[{"columns":["matrix1","matrix2","matrix3","keanu","la
urence","carrieanne"],"data":[{"row":[{"year":"1999-03-
31","title":"The Matrix"},{"year":"2003-05-07","title":"The
Matrix Reloaded"},{"year":"2003-10-27","title":"The Matrix
Revolutions"},{"name":"Keanu Reeves"},{"name":"Laurence
Fishburne"},{"name":"Carrie-Anne
Moss"}],"meta":[{"id":16,"type":"node","deleted":false},{"id":17,
"type":"node","deleted":false},{"id":18,"type":"node","deleted":f
alse},{"id":19,"type":"node","deleted":false},{"id":20,"type":"no
de","deleted":false},{"id":21,"type":"node","deleted":false}]}]}]
,"errors":[]}
```



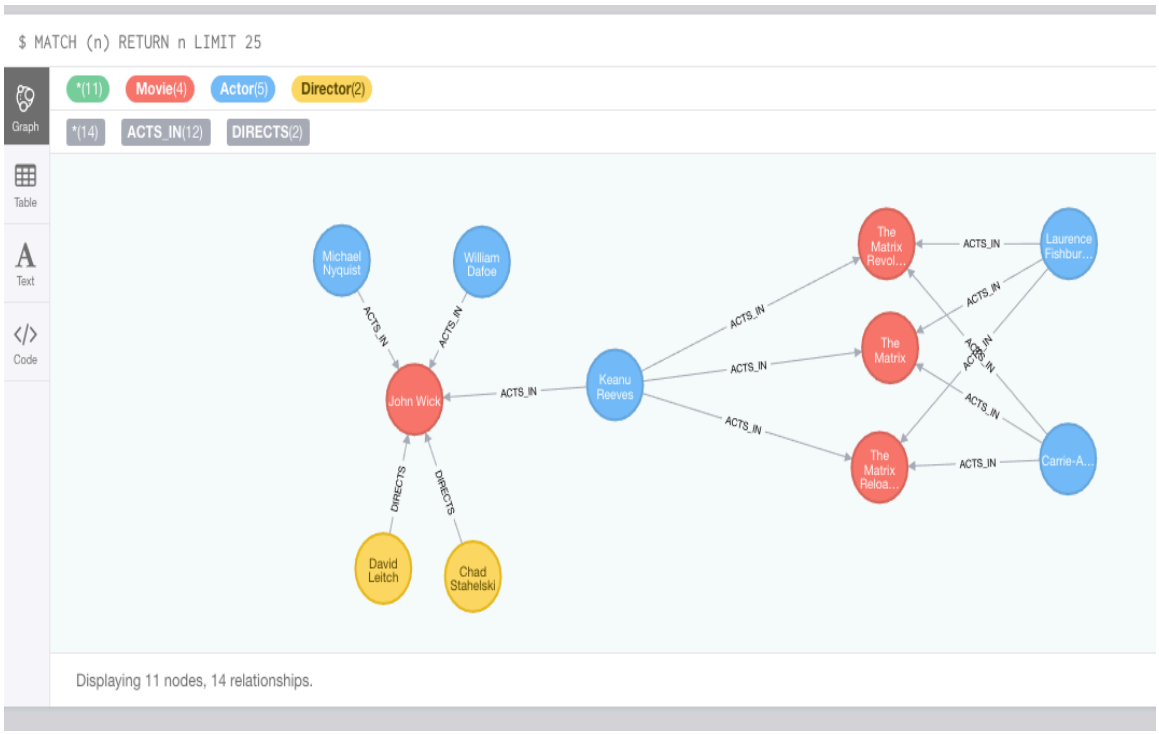
Verify

```
swaite@Rmt-mac-swaite:~/stirling/CSIE-63/assignment-2|master⚡
⇒  ./ps1a.sh
HTTP/1.1 200 OK
Date: Sat, 16 Sep 2017 01:43:53 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Content-Length: 638
Server: Jetty(9.2.z-SNAPSHOT)

{"results":[{"columns":["m"],"data":[{"row":[{"year":"1999-03-
31","title":"The
Matrix"}],"meta":[{"id":16,"type":"node","deleted":false}]},{"row
":[{"year":"2003-05-07","title":"The Matrix
Reloaded"}],"meta":[{"id":17,"type":"node","deleted":false}]},{"r
ow":[{"year":"2003-10-27","title":"The Matrix
Revolutions"}],"meta":[{"id":18,"type":"node","deleted":false}]},
{"row":[{"name":"Keanu
Reeves"}],"meta":[{"id":19,"type":"node","deleted":false}]},{"row
```

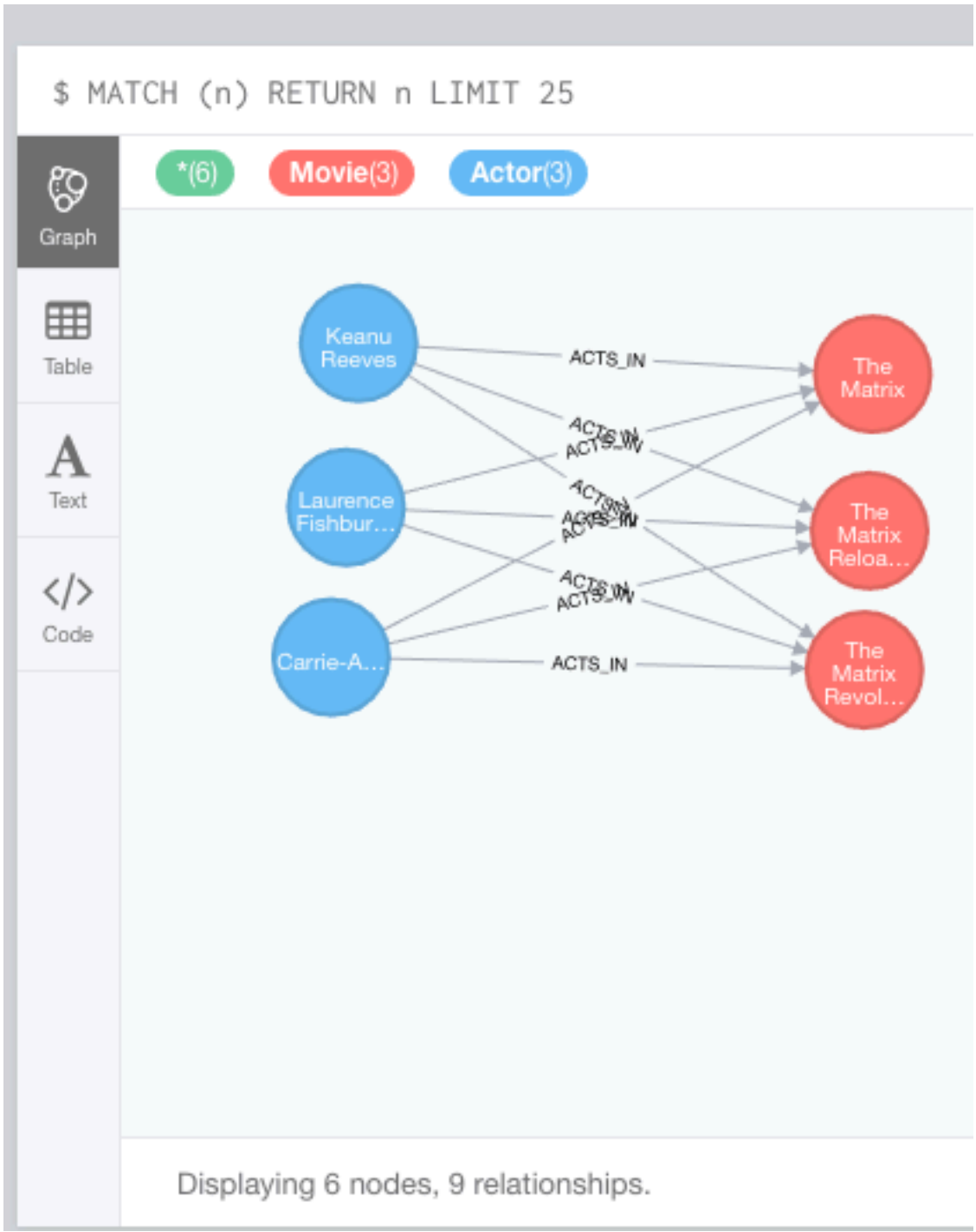
```
":[{"name":"Laurence
Fishburne"}],"meta":[{"id":20,"type":"node","deleted":false}]},{"
row":[{"name":"Carrie-Anne
Moss"}],"meta":[{"id":21,"type":"node","deleted":false}]}]},"err
ors":[]} row":[{"name":"Carrie-Anne
Moss"}],"meta":[{"id":21,"type":"node","deleted":false}]}]},"err
ors":[]}
```

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%)

```
from neo4j.v1 import GraphDatabase, basic_auth

driver =
GraphDatabase.driver("bolt://localhost:7687",auth=basic_auth("neo
4j", "neo4jj"))
session = driver.session()

# create new movie and attach directors, actors and roles
# https://neo4j.com/developer/python/#neo4j-python-driver
##
session.run("MATCH (keanu:Actor { name:'Keanu Reeves'}) "
"CREATE (wick:Movie { title:'John Wick', year:'2014-05-11'}) "
"CREATE (keanu)-[:ACTS_IN {role: 'Wick'}]->(wick) "
"CREATE (dafoe:Actor { name:'William Dafoe' }) "
"CREATE (dafoe)-[:ACTS_IN {role: 'Marcus'}]->(wick) "
"CREATE (nyquist:Actor { name:'Michael Nyquist' }) "
"CREATE (nyquist)-[:ACTS_IN {role: 'Viggo'}]->(wick) "
"CREATE (chad:Director { name:'Chad Stahelski' }) "
"CREATE (chad)-[:DIRECTS]->(wick) "
"CREATE (leitch:Director { name:'David Leitch' }) "
"CREATE (leitch)-[:DIRECTS]->(wick) ")

result = session.run("MATCH (m:Movie) WHERE m.title = {title} "
"RETURN m.title AS title, m.year AS year",
{"title": "John Wick"})

for record in result:
        print("%s %s" %(record["title"], record["year"]))

session.close()
```

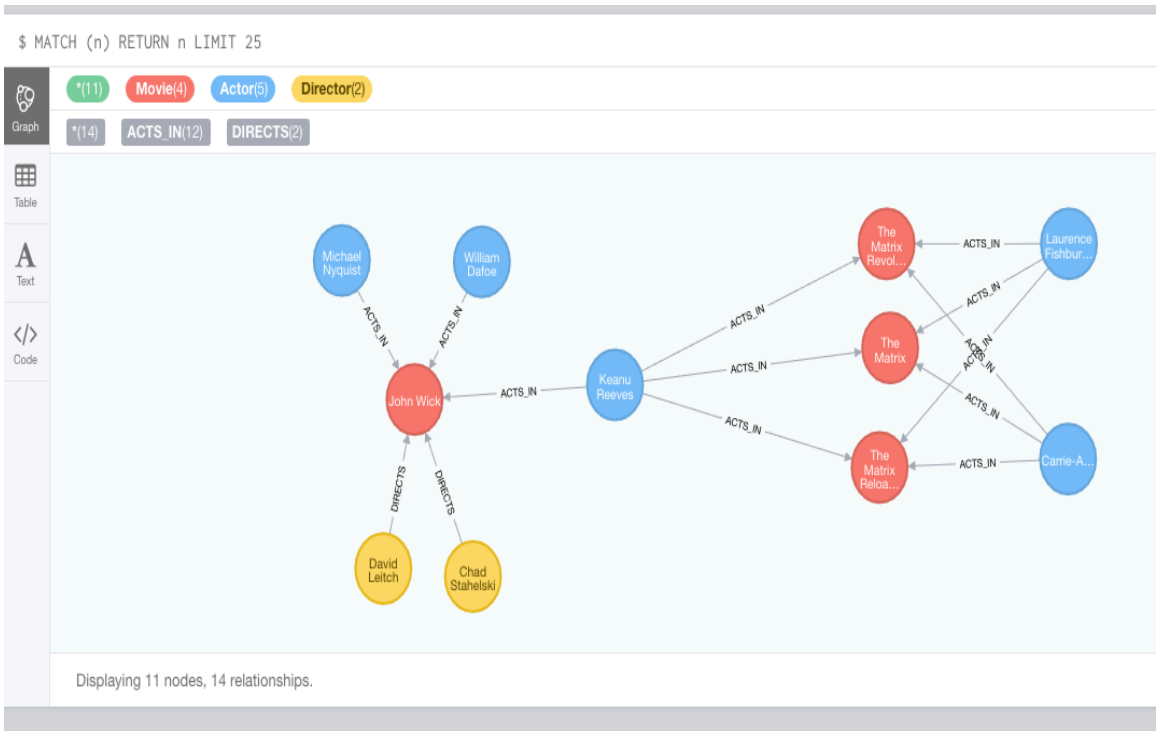



```
session.run("""MATCH (n { name: 'David Leitch'}) DETACH DELETE
n""")
session.run("""MATCH (n { name: 'Chad Stahelski'}) DETACH DELETE
n""")
session.run("""MATCH (n { name: 'Michael Nyquist' }) DETACH
DELETE n""")
session.run("""MATCH (n { name: 'William Dafoe' }) DETACH DELETE
n""")

session.run("""MATCH (n { title: 'John Wick'}) DETACH DELETE
n""")

session.run("MATCH (n:Director) REMOVE n:Director")
session.run("MATCH p=()-[r:DIRECTS]->() DELETE r")

session.close()
```



Problem 3. Add all 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%)

```
from neo4j.v1 import GraphDatabase, basic_auth

driver =
GraphDatabase.driver("bolt://localhost:7687",auth=basic_auth("neo
4j", "neo4jj"))
session = driver.session()

# create new movie and attach directors, actors and roles
# https://neo4j.com/developer/python/#neo4j-python-driver
##
session.run("MATCH (keanu:Actor { name:'Keanu Reeves'}) "
"CREATE (wick:Movie { title:'John Wick', year:'2014-05-11'}) "
"CREATE (keanu)-[:ACTS_IN {role: 'Wick'}]->(wick) "
"CREATE (dafoe:Actor { name:'William Dafoe' }) "
"CREATE (dafoe)-[:ACTS_IN {role: 'Marcus'}]->(wick) "
"CREATE (nyquist:Actor { name:'Michael Nyquist' }) "
"CREATE (nyquist)-[:ACTS_IN {role: 'Viggo'}]->(wick) "
"CREATE (chad:Director { name:'Chad Stahelski' }) "
"CREATE (chad)-[:DIRECTS]->(wick) "
"CREATE (leitch:Director { name:'David Leitch' }) "
"CREATE (leitch)-[:DIRECTS]->(wick) ")

result = session.run("MATCH (m:Movie) WHERE m.title = {title} "
"RETURN m.title AS title, m.year AS year",
{"title": "John Wick"})

for record in result:
        print("%s %s" %(record["title"], record["year"]))

session.close()
```



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%)

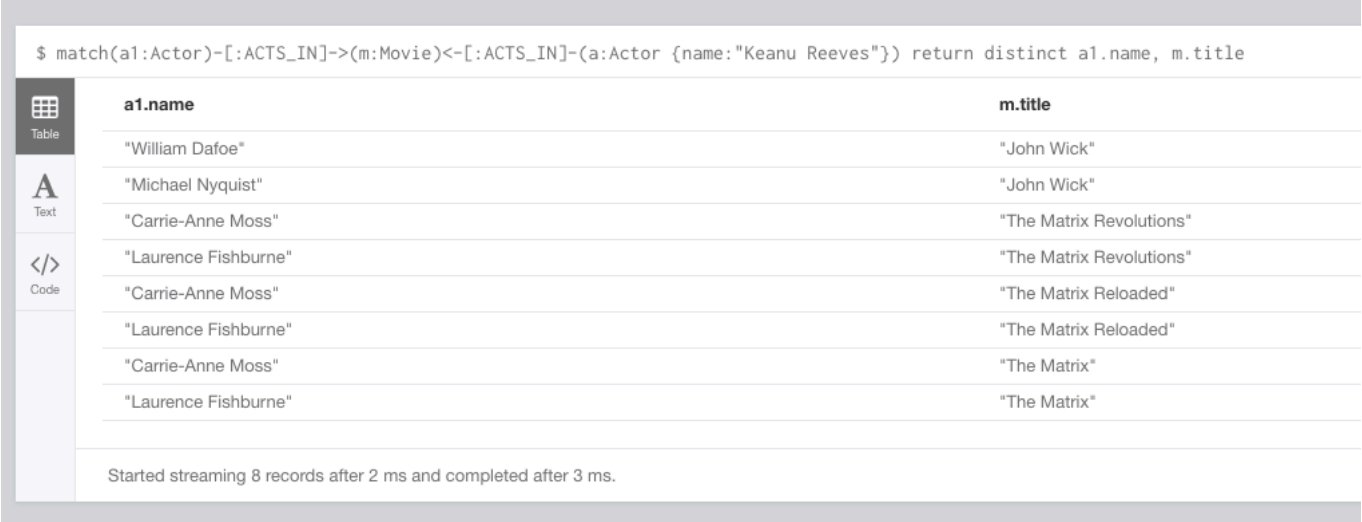
```
from neo4j.v1 import GraphDatabase, basic_auth

driver =
GraphDatabase.driver("bolt://localhost:7687",auth=basic_auth("neo
4j", "neo4j"))
session = driver.session()

keanu = "Keanu Reeves"

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

results = session.run("match(a1:Actor)-[:ACTS_IN]->(m:Movie)<-
[:ACTS_IN]-(a:Actor {name:\""+keanu+"\"}) return distinct a1.name
as name, m.title as title")

print("==== List of actors playing in movies in which %s played
====" %keanu)
for record in results:
```

```
    print("%s (Movie: %s)" %(record["name"], record["title"]))

print("\n")

# Find directors of movies in which K. Reeves played

results = session.run("match(d:Director)-[:DIRECTS]->(m:Movie)<-
[:ACTS_IN]-(a:Actor {name:\""+keanu+"\"}) return d.name as name,
m.title as title ")

print("==== Directors of movies in which %s played ====" %keanu)
for record in results:
        print("%s (Movie: %s)" %(record["name"],
record["title"]))

session.close()
```

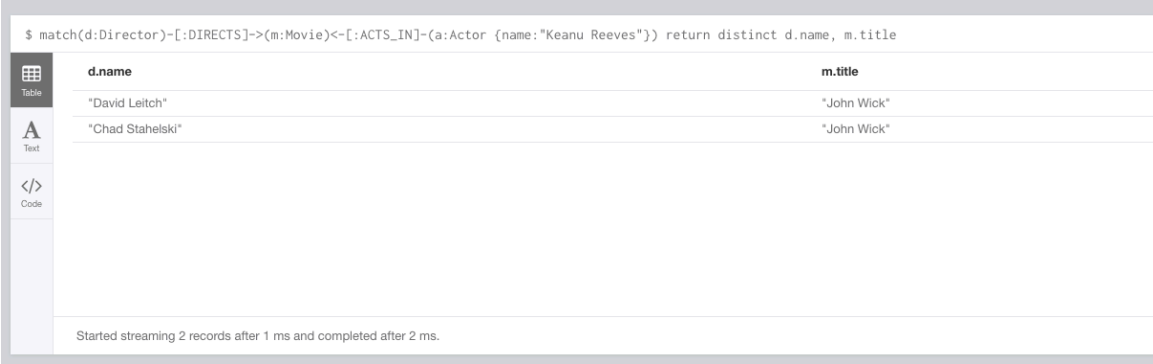
```
swaite@Rmt-mac-swaite:~/stirling/CSIE-63/assignment-2|master⚡
⇒  python2.7 p4.py
==== List of actors playing in movies in which Keanu Reeves played ====
William Dafoe (Movie: John Wick)
Michael Nyquist (Movie: John Wick)
Carrie-Anne Moss (Movie: The Matrix Revolutions)
Laurence Fishburne (Movie: The Matrix Revolutions)
Carrie-Anne Moss (Movie: The Matrix Reloaded)
Laurence Fishburne (Movie: The Matrix Reloaded)
Carrie-Anne Moss (Movie: The Matrix)
Laurence Fishburne (Movie: The Matrix)


==== Directors of movies in which Keanu Reeves played ====
David Leitch (Movie: John Wick)
Chad Stahelski (Movie: John Wick)
```

Verification

```
match(a1:Actor)-[:ACTS_IN]->(m:Movie)<-[:ACTS_IN]-(a:Actor {name:"Keanu
Reeves"}) return distinct a1.name, m.title
```




```
match(d:Director)-[:DIRECTS]->(m:Movie)<-[:ACTS_IN]-(a:Actor
{name:"Keanu Reeves"}) return distinct d.name, m.title
```



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%)

p5-export.py

```
from neo4j.v1 import GraphDatabase, basic_auth

driver =
GraphDatabase.driver("bolt://localhost:7687",auth=basic_auth("neo
4j", "neo4jj"))
session = driver.session()
path = "/Users/swaite/stirling/CSIE-63/assignment-2/data/input/"
```

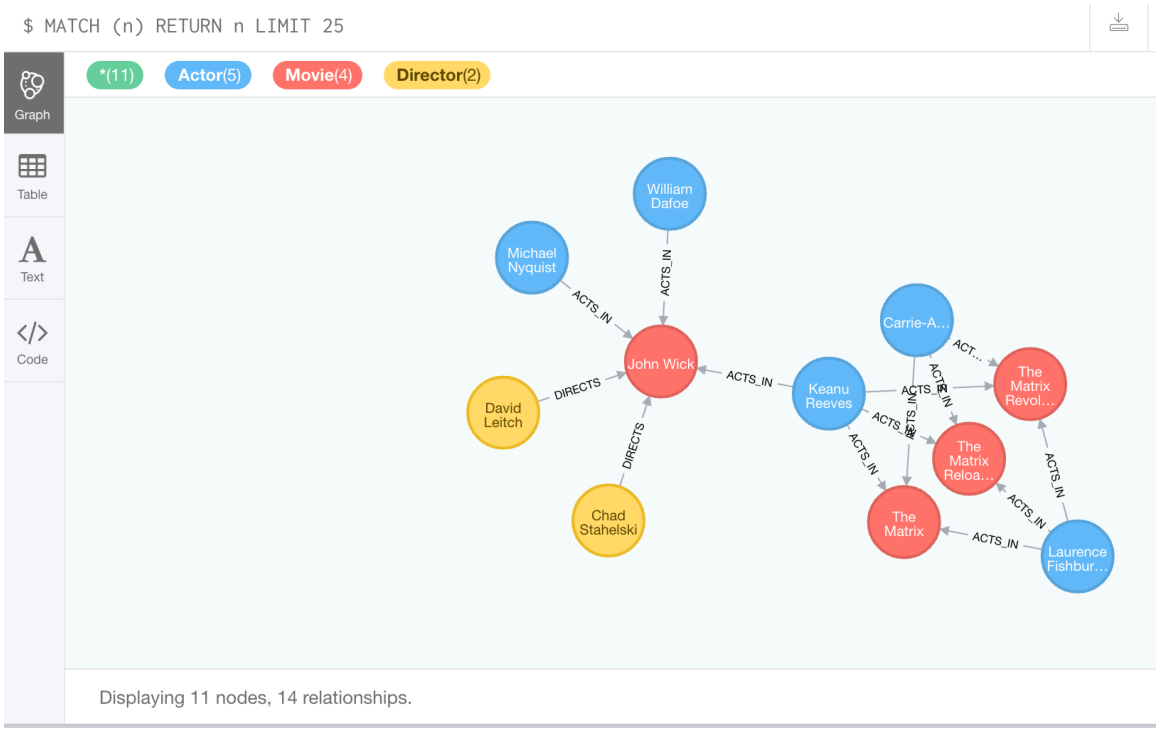
```
### Actors ###

result = session.run("""
MATCH (n:Actor)
RETURN n.name as name;
""")

f = open(path + 'actors.csv','w')
f.write("name\n")
for record in result:
    f.write(record["name"] +"\n")
f.close()

### Directors ###

result = session.run("""
MATCH (d:Director)
RETURN d.name as name;
""")

f = open(path + 'directors.csv','w')
f.write("name\n")
for record in result:
    f.write(record["name"] +"\n")
f.close()


### Movies ###
result = session.run("""
MATCH (m:Movie)
RETURN m.title as title, m.year as year
""")

f = open(path +'movies.csv','w')
f.write("title,year\n")
for record in result:
    f.write(record["title"] + "," + str(record["year"]) + "\n")
f.close()


### Relationships - ACTS IN  ###
f = open(path + 'relationships-acts-in.csv','w')
f.write("name,role,year,title\n")
result = session.run("""
MATCH p=(a:Actor)-[r:ACTS_IN]->(m:Movie)
RETURN a.name as name, r.role as role, m.year as year, m.title as
title
""")
```

```
for record in result:
    f.write(record["name"] + "," + record["role"] + "," +
str(record["year"]) + "," + record["title"] + "\n")
f.close()

### Relationships - DIRECTS  ###
f = open(path + 'directs.csv','w')
f.write("name,year,title\n")
result = session.run("""
MATCH p=(d:Director)-[r:DIRECTS]->(m:Movie)
RETURN d.name as name, m.year as year, m.title as title
""")
for record in result:
    f.write(record["name"] + "," + str(record["year"]) + "," +
record["title"] + "\n")
f.close()


session.close()
```

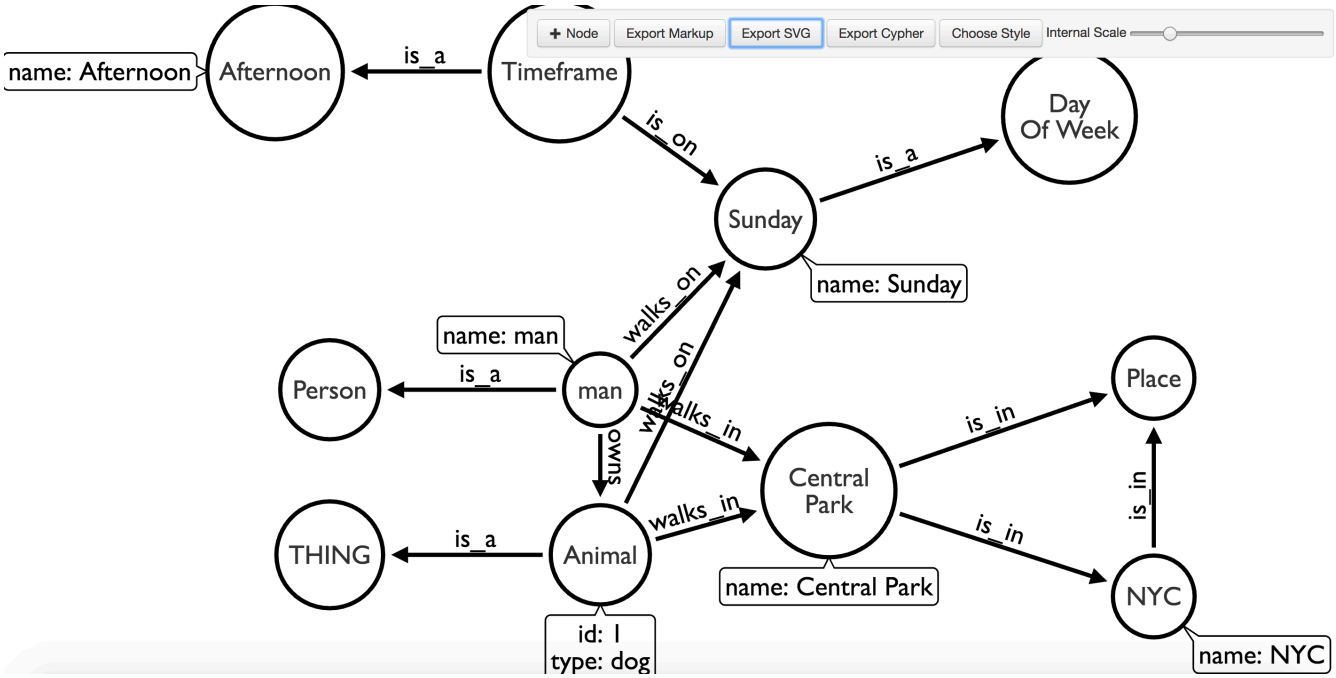
p5-import.py

```
import csv
from neo4j.v1 import GraphDatabase, basic_auth

driver =
GraphDatabase.driver("bolt://localhost:7687",auth=basic_auth("neo
4j", "neo4jj"))
session = driver.session()
path = "file:///Users/swaite/stirling/CSIE-63/assignment-
2/data/input/"

### CLEAR DATABASE ###
session.run(""" match(n) detach delete n """)

#### Actor ###
session.run("""
LOAD CSV WITH HEADERS FROM "file:///actors.csv" AS line
CREATE (a:Actor {name:line.name});
""")

#### Movies ###
session.run("""
LOAD CSV WITH HEADERS FROM 'file:///movies.csv' AS line
CREATE (m:Movie {title:line.title, year:line.year});
""")

### Directors ###
session.run("""
```

```
LOAD CSV WITH HEADERS FROM "file:///directors.csv" AS line
CREATE (d:Director {name:line.name});
""")

###  RELATIONSHIPS - ACTS_IN ###
session.run("""
LOAD CSV WITH HEADERS FROM "file:///relationships-acts-in.csv" AS
line
MATCH (m:Movie) WHERE m.title = line.title
MATCH (a:Actor) WHERE a.name = line.name
CREATE (a)-[:ACTS_IN {role:line.role}]->(m)
""")

###  RELATIONSHIPS - DIRECTS ###
session.run("""
LOAD CSV WITH HEADERS FROM "file:///directs.csv" AS line
MATCH (d:Director) WHERE d.name = line.name
MATCH (m:Movie) WHERE m.title = line.title and m.year=line.year
CREATE (d)-[:DIRECTS]->(m)
""")

session.close()
```



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%)



CYPHER

```
CREATE
 (`0` :Person ) ,
 (`1` :THING ) ,
 (`2` :Place ) ,
 (`4` :man {name:'man'}) ,
 (`5` :Animal {id:'1',type:'dog'}) ,
 (`6` :NYC {name:'NYC'}) ,
 (`7` :`Central Park` {name:'Central Park'}) ,
 (`8` :`Day Of Week` ) ,
 (`9` :Sunday {name:'Sunday'}) ,
 (`10` :Timeframe ) ,
 (`11` :Afternoon {name:'Afternoon'}) ,
 (`4`)-[:`is_a`]->(`0`),
 (`5`)-[:`is_a`]->(`1`),
 (`4`)-[:`owns`]->(`5`),
 (`6`)-[:`is_in`]->(`2`),
 (`4`)-[:`walks_in`]->(`7`),
```

(`7`)-[:`is_in`]->(`2`),
(`7`)-[:`is_in`]->(`6`),
(`5`)-[:`walks_in`]->(`7`),
(`9`)-[:`is_a`]->(`8`),
(`10`)-[:`is_on`]->(`9`),
(`10`)-[:`is_a`]->(`11`),
(`5`)-[:`walks_on`]->(`9`),
(`4`)-[:`walks_on`]->(`9`)