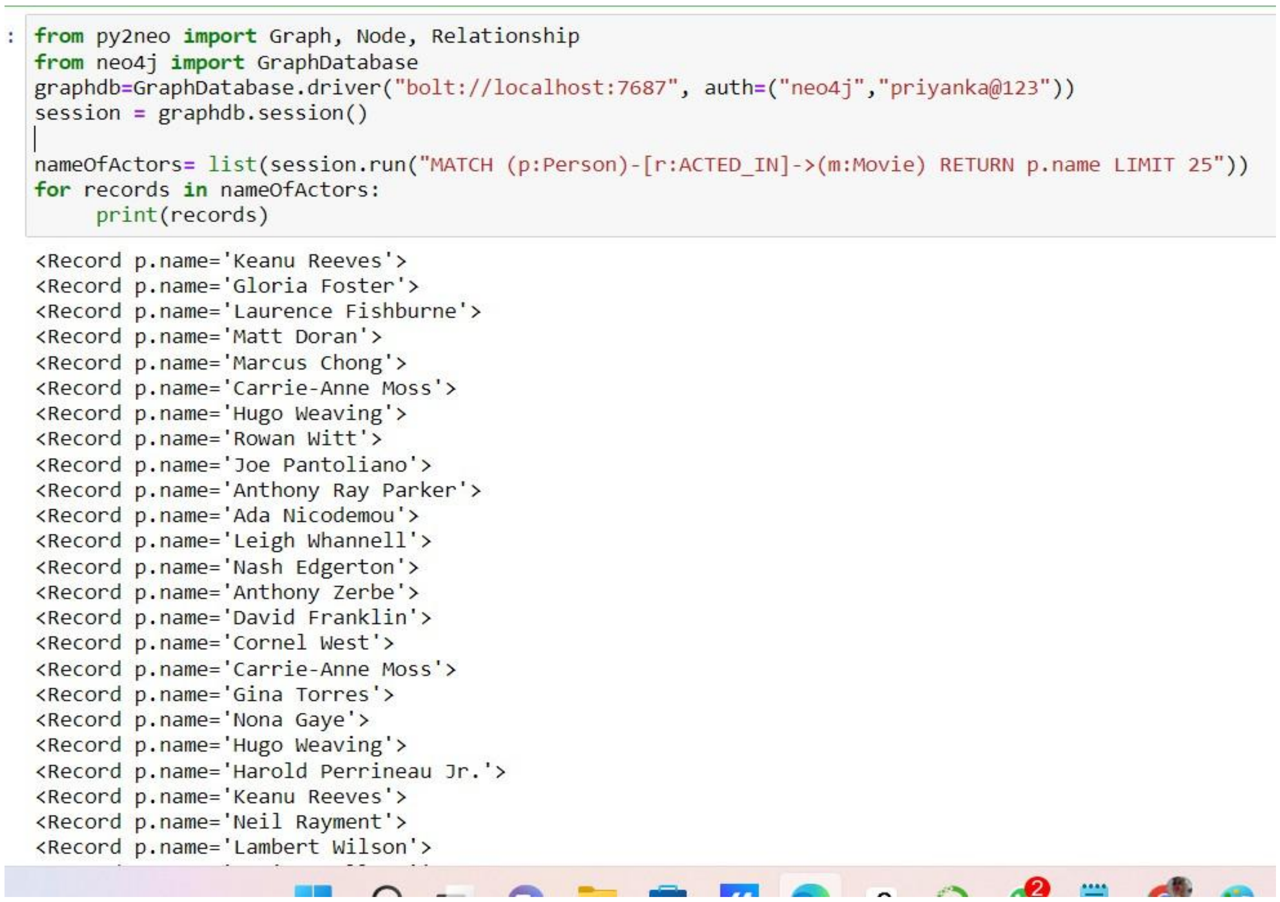
Big Data: Management and Analytics

Task 1 Recalling the first assignment task 3 part d, you were asked to create a new relationship 'FOLLOWS' between two users. As you may recognize from this task, our graph database has a label named 'User' which already has some entries. Your task was to find two users with the username 'James' and 'Thomas' and create a relationship between them. As it is mentioned before you can maintain a connection between your code in python and Neo4j with several pythons libraries e.g. py2neo.

Graph model should be build in Neo4j

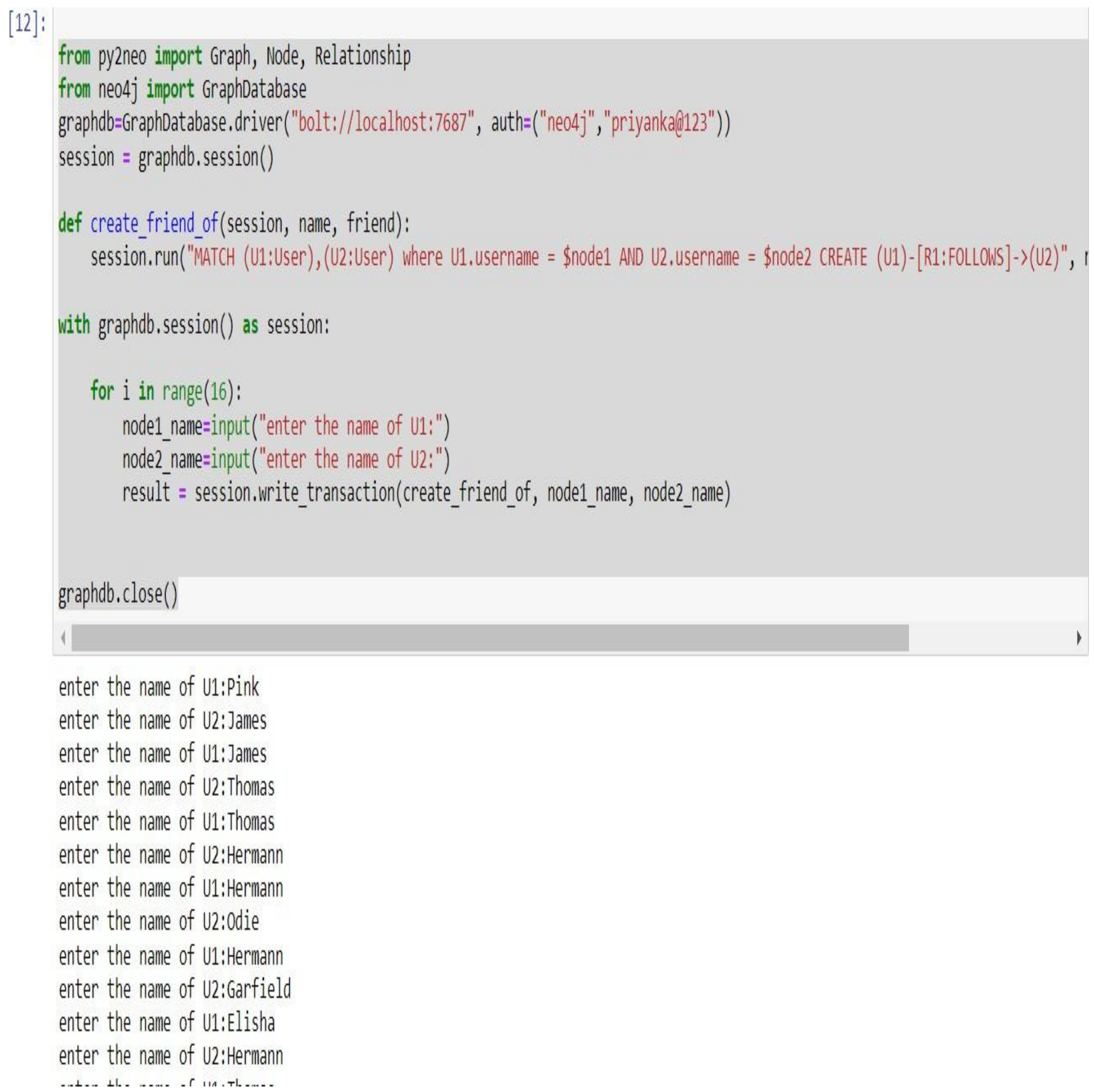
(a) Connect your python code with Neo4j using the above mentioned library or a similar one. Retrieve the name of any actor from the database.



```
from py2neo import Graph, Node, Relationship
from neo4j import GraphDatabase
graphdb=GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j","priyanka@123"))
session = graphdb.session()

nameOfActors= list(session.run("MATCH (p:Person)-[r:ACTED_IN]->(m:Movie) RETURN p.name LIMIT 25"))
for records in nameOfActors:
    print(records)
```

```
<Record p.name='Keanu Reeves'>
<Record p.name='Gloria Foster'>
<Record p.name='Laurence Fishburne'>
<Record p.name='Matt Doran'>
<Record p.name='Marcus Chong'>
<Record p.name='Carrie-Anne Moss'>
<Record p.name='Hugo Weaving'>
<Record p.name='Rowan Witt'>
<Record p.name='Joe Pantoliano'>
<Record p.name='Anthony Ray Parker'>
<Record p.name='Ada Nicodemou'>
<Record p.name='Leigh Whannell'>
<Record p.name='Nash Edgerton'>
<Record p.name='Anthony Zerbe'>
<Record p.name='David Franklin'>
<Record p.name='Cornel West'>
<Record p.name='Carrie-Anne Moss'>
<Record p.name='Gina Torres'>
<Record p.name='Nona Gaye'>
<Record p.name='Hugo Weaving'>
<Record p.name='Harold Perrineau Jr.'>
<Record p.name='Keanu Reeves'>
<Record p.name='Neil Rayment'>
<Record p.name='Lambert Wilson'>
```

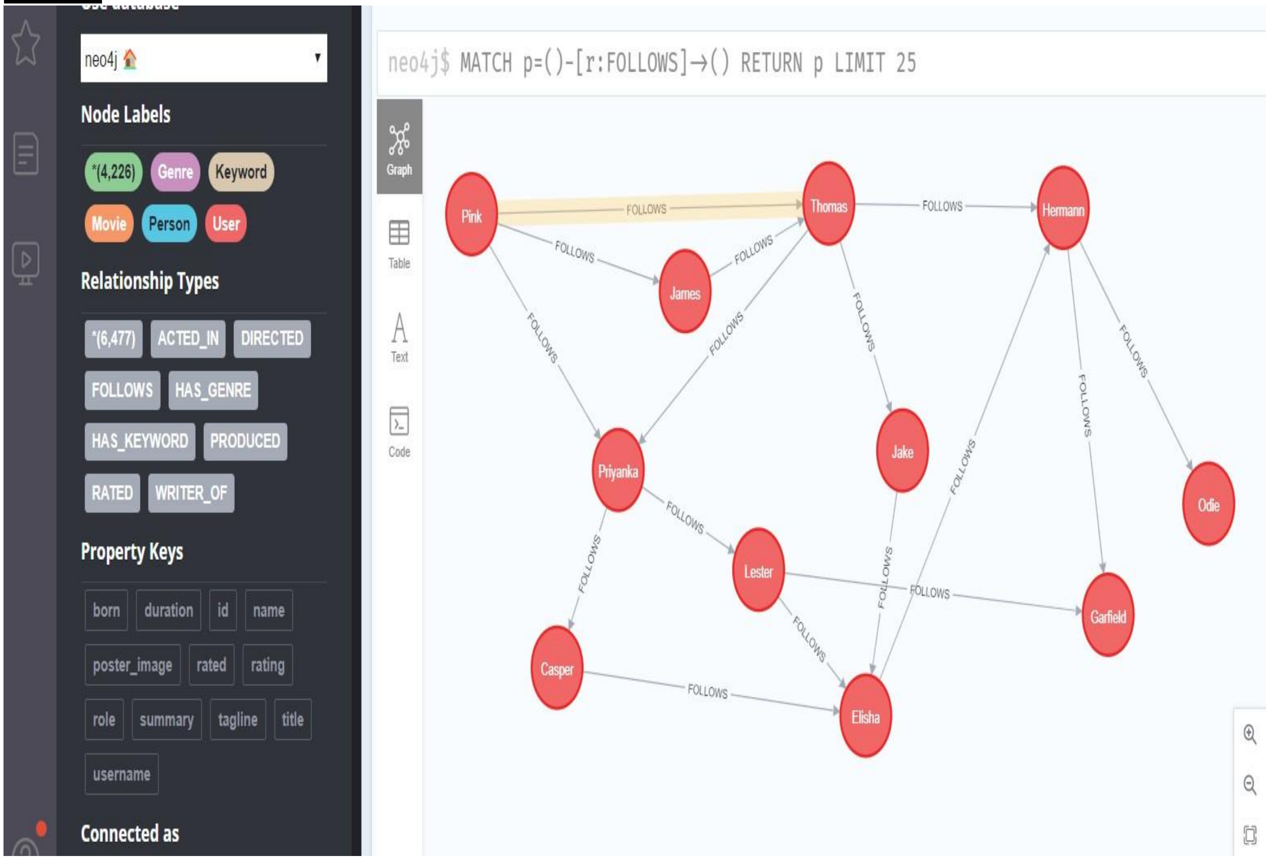
(b) **Consider the graph in Fig 1. All the Users are already implemented as nodes in our database. Insert this structure into your Neo4j database. Use your python code to access and manipulate your database to create the relationships between the nodes seen in Fig 1. Afterwards, with the Neo4j-browser, try to show the results of your insertions as your output. (This should work by using just ONE query.) Note, that all the users already exist in your graph.**



```
[12]:
from py2neo import Graph, Node, Relationship
from neo4j import GraphDatabase
graphdb=GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j","priyanka@123"))
session = graphdb.session()

def create_friend_of(session, name, friend):
    session.run("MATCH (U1:User),(U2:User) where U1.username = $node1 AND U2.username = $node2 CREATE (U1)-[R1:FOLLOWS]->(U2)",

with graphdb.session() as session:

    for i in range(16):
        node1_name=input("enter the name of U1:")
        node2_name=input("enter the name of U2:")
        result = session.write_transaction(create_friend_of, node1_name, node2_name)


graphdb.close()
```

```
enter the name of U1:Pink
enter the name of U2:James
enter the name of U1:James
enter the name of U2:Thomas
enter the name of U1:Thomas
enter the name of U2:Hermann
enter the name of U1:Hermann
enter the name of U2:Odie
enter the name of U1:Hermann
enter the name of U2:Garfield
enter the name of U1:Elisha
enter the name of U2:Hermann
```

Output



```
neo4j$ MATCH p=()-[r:FOLLOWS]→() RETURN p LIMIT 25
```

neo4j

Node Labels

*(4,226) Genre Keyword Movie Person User

Relationship Types

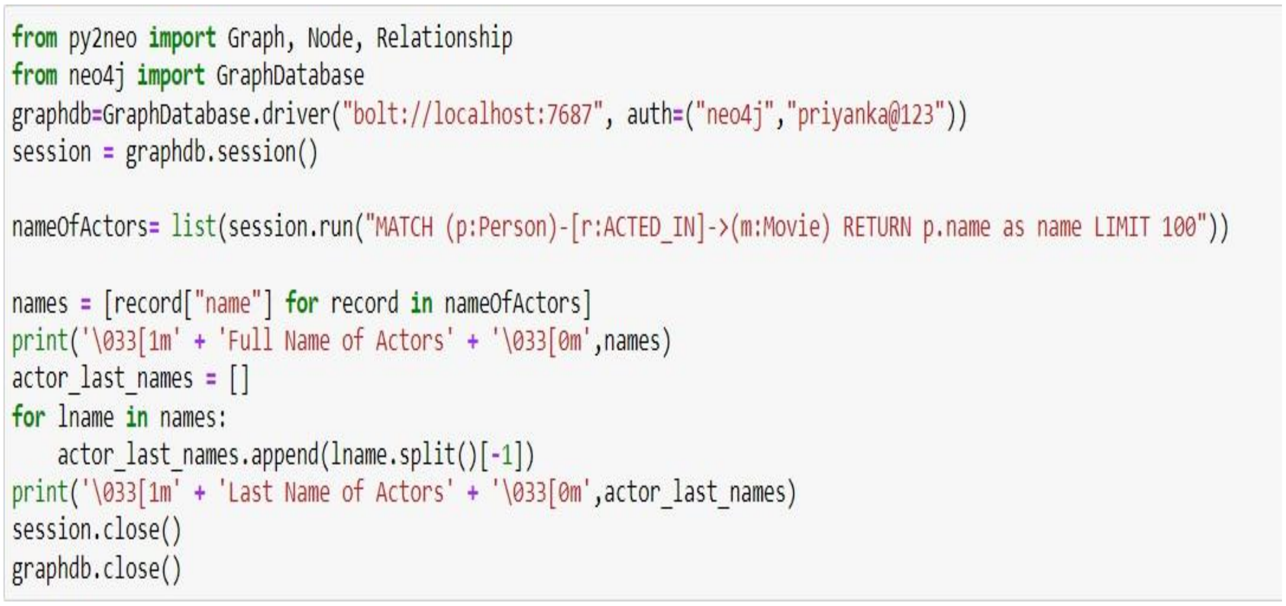
*(6,477) ACTED_IN DIRECTED FOLLOWS HAS_GENRE HAS_KEYWORD PRODUCED RATED WRITER_OF

Property Keys

born duration id name poster_image rated rating role summary tagline title username

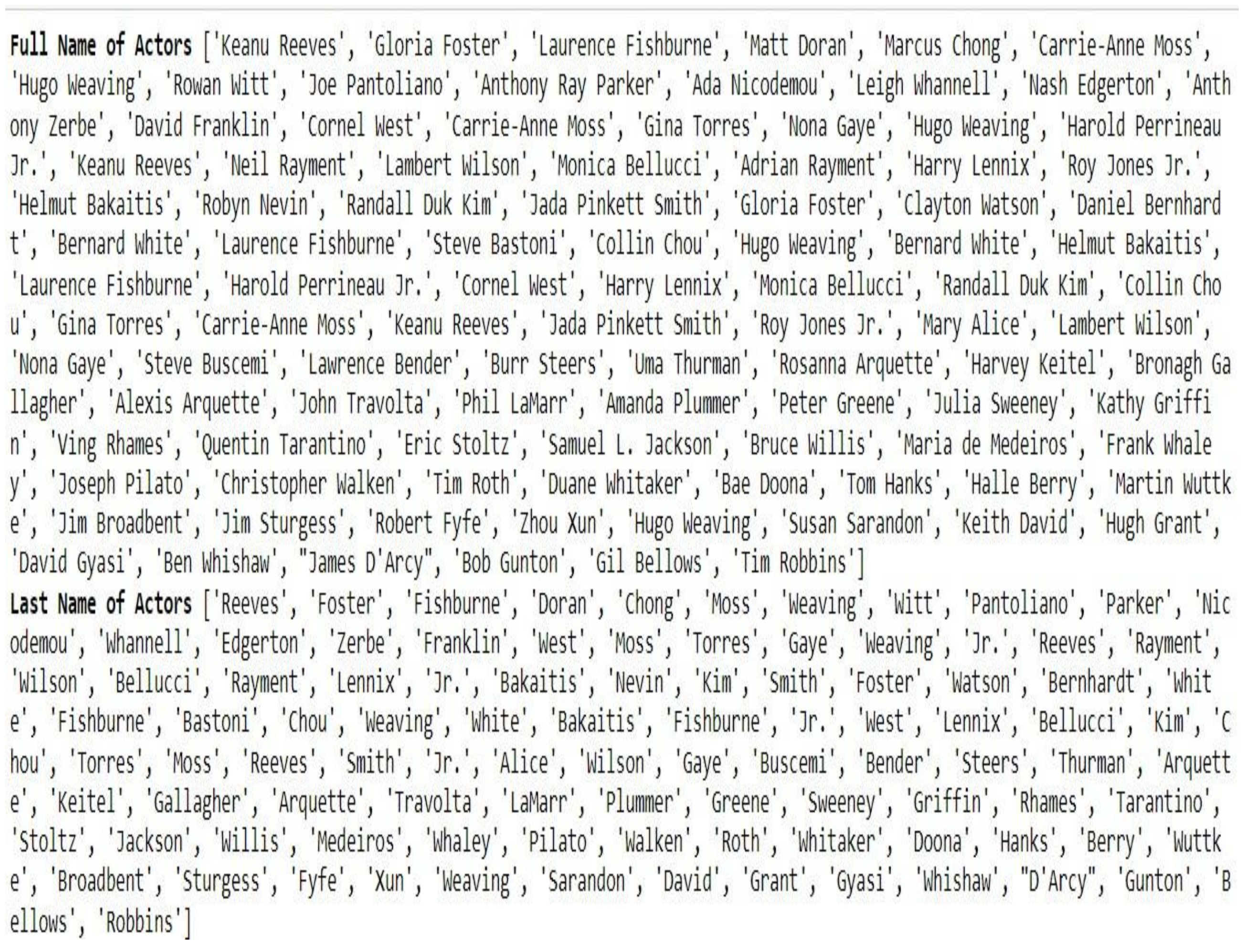
Connected as

(c) **Write a python code that retrieve 100 actor's names from Neo4j database and make a list with them. With the help of regex exclude the firstname from each of their (full-)names and make a new list with just their lastnames. Hint: You can read about regex here: https://docs.python.org/3/library/re. html**



```python
from py2neo import Graph, Node, Relationship
from neo4j import GraphDatabase
graphdb=GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j","priyanka@123"))
session = graphdb.session()

nameOfActors= list(session.run("MATCH (p:Person)-[r:ACTED_IN]->(m:Movie) RETURN p.name as name LIMIT 100"))

names = [record["name"] for record in nameOfActors]
print('\033[1m' + 'Full Name of Actors' + '\033[0m',names)
actor_last_names = []
for lname in names:
    actor_last_names.append(lname.split()[-1])
print('\033[1m' + 'Last Name of Actors' + '\033[0m',actor_last_names)
session.close()
graphdb.close()
```

OutPut



```
Full Name of Actors ['Keanu Reeves', 'Gloria Foster', 'Laurence Fishburne', 'Matt Doran', 'Marcus Chong', 'Carrie-Anne Moss',
'Hugo Weaving', 'Rowan Witt', 'Joe Pantoliano', 'Anthony Ray Parker', 'Ada Nicodemou', 'Leigh Whannell', 'Nash Edgerton', 'Anth
ony Zerbe', 'David Franklin', 'Cornel West', 'Carrie-Anne Moss', 'Gina Torres', 'Nona Gaye', 'Hugo Weaving', 'Harold Perrineau
Jr.', 'Keanu Reeves', 'Neil Rayment', 'Lambert Wilson', 'Monica Bellucci', 'Adrian Rayment', 'Harry Lennix', 'Roy Jones Jr.',
'Helmut Bakaitis', 'Robyn Nevin', 'Randall Duk Kim', 'Jada Pinkett Smith', 'Gloria Foster', 'Clayton Watson', 'Daniel Bernhard
t', 'Bernard White', 'Laurence Fishburne', 'Steve Bastoni', 'Collin Chou', 'Hugo Weaving', 'Bernard White', 'Helmut Bakaitis',
'Laurence Fishburne', 'Harold Perrineau Jr.', 'Cornel West', 'Harry Lennix', 'Monica Bellucci', 'Randall Duk Kim', 'Collin Cho
u', 'Gina Torres', 'Carrie-Anne Moss', 'Keanu Reeves', 'Jada Pinkett Smith', 'Roy Jones Jr.', 'Mary Alice', 'Lambert Wilson',
'Nona Gaye', 'Steve Buscemi', 'Lawrence Bender', 'Burr Steers', 'Uma Thurman', 'Rosanna Arquette', 'Harvey Keitel', 'Bronagh Ga
llagher', 'Alexis Arquette', 'John Travolta', 'Phil LaMarr', 'Amanda Plummer', 'Peter Greene', 'Julia Sweeney', 'Kathy Griffi
n', 'Ving Rhames', 'Quentin Tarantino', 'Eric Stoltz', 'Samuel L. Jackson', 'Bruce Willis', 'Maria de Medeiros', 'Frank Whale
y', 'Joseph Pilato', 'Christopher Walken', 'Tim Roth', 'Duane Whitaker', 'Bae Doona', 'Tom Hanks', 'Halle Berry', 'Martin Wuttk
e', 'Jim Broadbent', 'Jim Sturgess', 'Robert Fyfe', 'Zhou Xun', 'Hugo Weaving', 'Susan Sarandon', 'Keith David', 'Hugh Grant',
'David Gyasi', 'Ben Whishaw', "James D'Arcy", 'Bob Gunton', 'Gil Bellows', 'Tim Robbins']
Last Name of Actors ['Reeves', 'Foster', 'Fishburne', 'Doran', 'Chong', 'Moss', 'Weaving', 'Witt', 'Pantoliano', 'Parker', 'Nic
odemou', 'Whannell', 'Edgerton', 'Zerbe', 'Franklin', 'West', 'Moss', 'Torres', 'Gaye', 'Weaving', 'Jr.', 'Reeves', 'Rayment',
'Wilson', 'Bellucci', 'Rayment', 'Lennix', 'Jr.', 'Bakaitis', 'Nevin', 'Kim', 'Smith', 'Foster', 'Watson', 'Bernhardt', 'Whit
e', 'Fishburne', 'Bastoni', 'Chou', 'Weaving', 'White', 'Bakaitis', 'Fishburne', 'Jr.', 'West', 'Lennix', 'Bellucci', 'Kim', 'C
hou', 'Torres', 'Moss', 'Reeves', 'Smith', 'Jr.', 'Alice', 'Wilson', 'Gaye', 'Buscemi', 'Bender', 'Steers', 'Thurman', 'Arquett
e', 'Keitel', 'Gallagher', 'Arquette', 'Travolta', 'LaMarr', 'Plummer', 'Greene', 'Sweeney', 'Griffin', 'Rhames', 'Tarantino',
'Stoltz', 'Jackson', 'Willis', 'Medeiros', 'Whaley', 'Pilato', 'Walken', 'Roth', 'Whitaker', 'Doona', 'Hanks', 'Berry', 'Wuttk
e', 'Broadbent', 'Sturgess', 'Fyfe', 'Xun', 'Weaving', 'Sarandon', 'David', 'Grant', 'Gyasi', 'Whishaw', "D'Arcy", 'Gunton', 'B
ellows', 'Robbins']
```

Task 2

In this task, we will practice the user-defined procedures and functions in Neo4j. You can read more about extending Neo4j under this 2 URLs: https:// neo4j.com/docs/java-reference/current/extending-neo4j/ and https://neo4j. com/docs/java-reference/current/extending-neo4j/functions/

(a) **Explain in your own words the advantages of using user-defined procedures in Neo4j.**

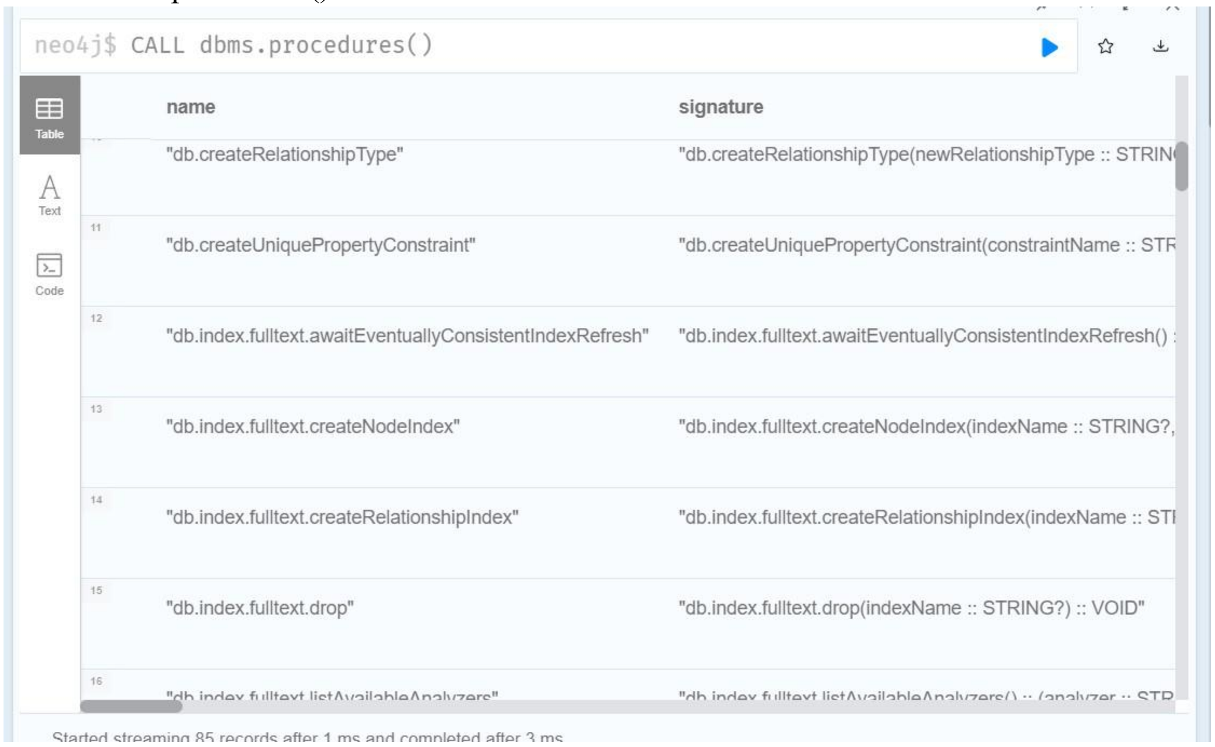
The following are some of the advantages of using user defined functions:

• Users may easily integrate with other data stores, expand import/export functionalities, and develop shiny graph algorithms using Neo4j's user-defined procedures.

• APOC, a community repository, is an excellent example. There's a great source of method and function examples there. These additions are not only quite valuable and appreciated, but the process of writing and publishing them is also incredibly easy.

• If a new procedure or function is invalid, the user will receive comprehensive error feedback while deploying it (in annotations, parameter or return types or injection points).

(b) **Write a Cypher query to retrieve all procedures**
CALL dbms.procedures()



```
neo4j$ CALL dbms.procedures()
```

	name	signature
	"db.createRelationshipType"	"db.createRelationshipType(newRelationshipType :: STRIN
11	"db.createUniquePropertyConstraint"	"db.createUniquePropertyConstraint(constraintName :: STR
12	"db.index.fulltext.awaitEventuallyConsistentIndexRefresh"	"db.index.fulltext.awaitEventuallyConsistentIndexRefresh() :
13	"db.index.fulltext.createNodeIndex"	"db.index.fulltext.createNodeIndex(indexName :: STRING?,
14	"db.index.fulltext.createRelationshipIndex"	"db.index.fulltext.createRelationshipIndex(indexName :: STI
15	"db.index.fulltext.drop"	"db.index.fulltext.drop(indexName :: STRING?) :: VOID"
16	"db.index.fulltext.listAvailableAnalyzers"	"db.index.fulltext.listAvailableAnalyzers() :: (analyzer :: STR

Started streaming 85 records after 1 ms and completed after 3 ms

(c) **Write a function that returns the movie name by passing a parameter (movie id), deploy it to Neo4j and run it as a Cypher statement in Neo4j**.
MATCH (m: MOVIE) WHERE m.id = 28 RETURN org.neo4j.exampleprogram.
movieFun(collect(m.title))

```
package exampleprogram;
import org.neo4j.procedure.Name;
import org.neo4j.procedure.Procedure;
import org.neo4j.procedure.UserFunction;

public class movieFun
{
   @UserFunction
   public String moviefun(@Name("strings") List<String> strings)
}
```

(d) **Write a function that returns the names of all actors that acted in a specific movie (actor id is a parameter), deploy it to Neo4j and run it as a Cypher statement.**

MATCH (m:Movie),(p:Person) WHERE m.id = 1 return
org.neo4j.eActor.Actorjoin(collect(p.name))

```
package eActor;
import org.neo4j.procedure.Name;
import org.neo4j.procedure.Procedure;
import org.neo4j.procedure.UserFunction;

public class ActorJoin
{
    @UserFunction
    public String actorjoin(
    @Name("strings") List<String> strings)
    @Name(value = "delimiter", defaultValue = ",") String delimiter)
      {
         if (strings == null || delimiter == null)
         {
           return null;
         }
       return String.actorjoin(delimiter, strings);
    }

}
```