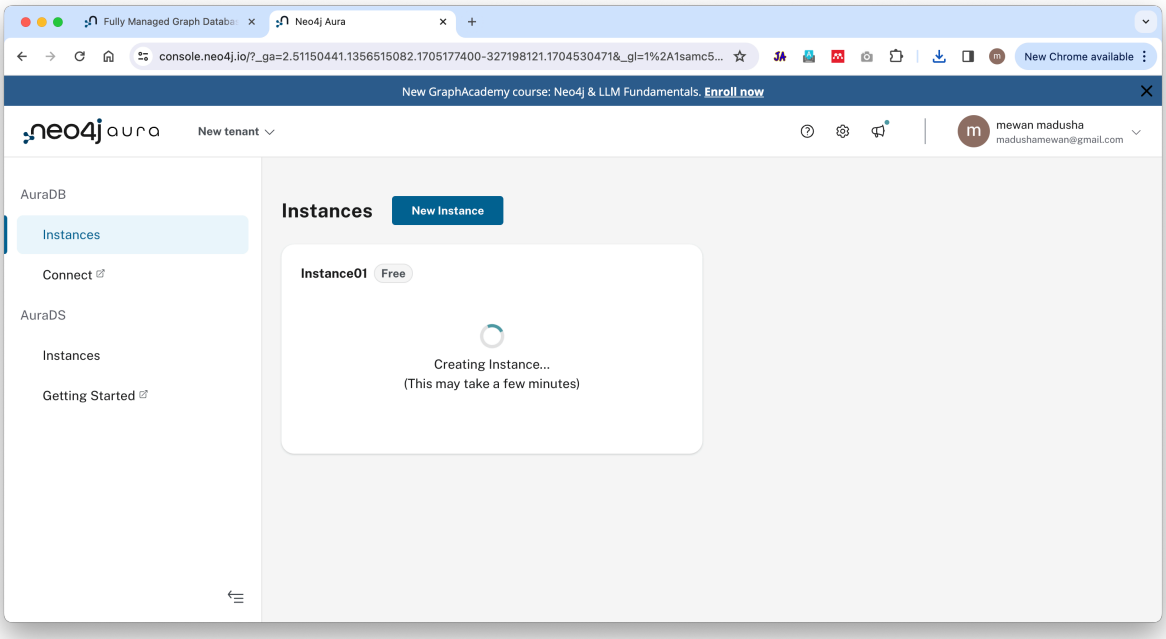
Student name: Pothumulla Kankanamge Mewan Madhusha
NB! Grading system. Assignment contains 15 tasks. Each task completion brings you 1 point regardless of complexity. If task completed partially grade might be smaller, for example, 0.5 for half of task completed. Maximum number of points to earn for this assignment is 15.

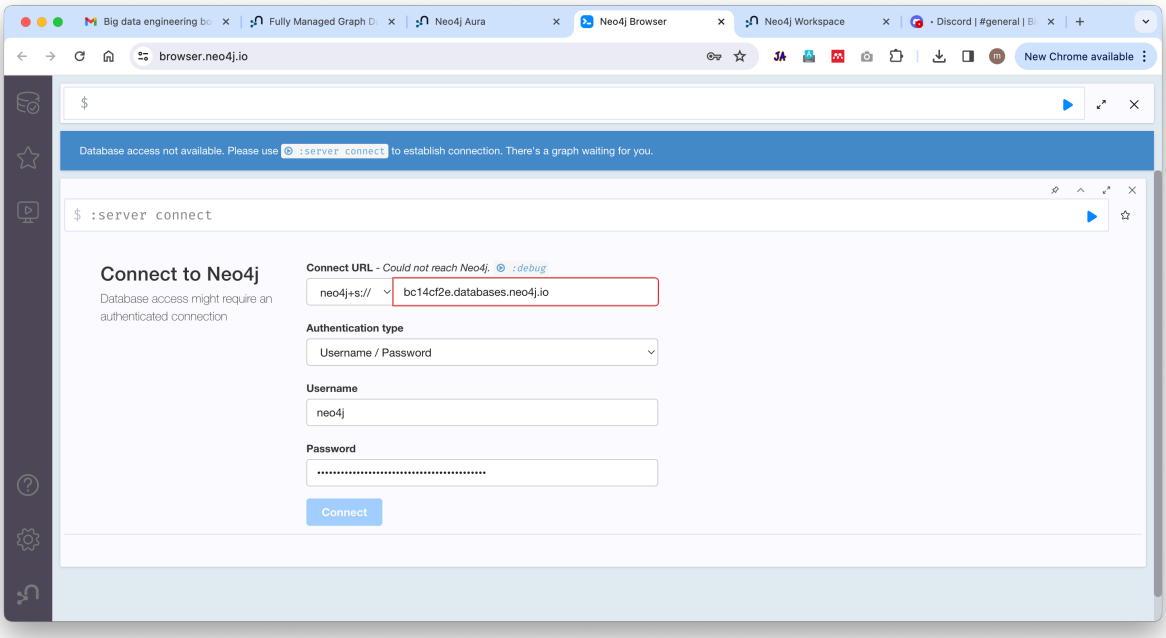
Neo4J task.

TASK 1.

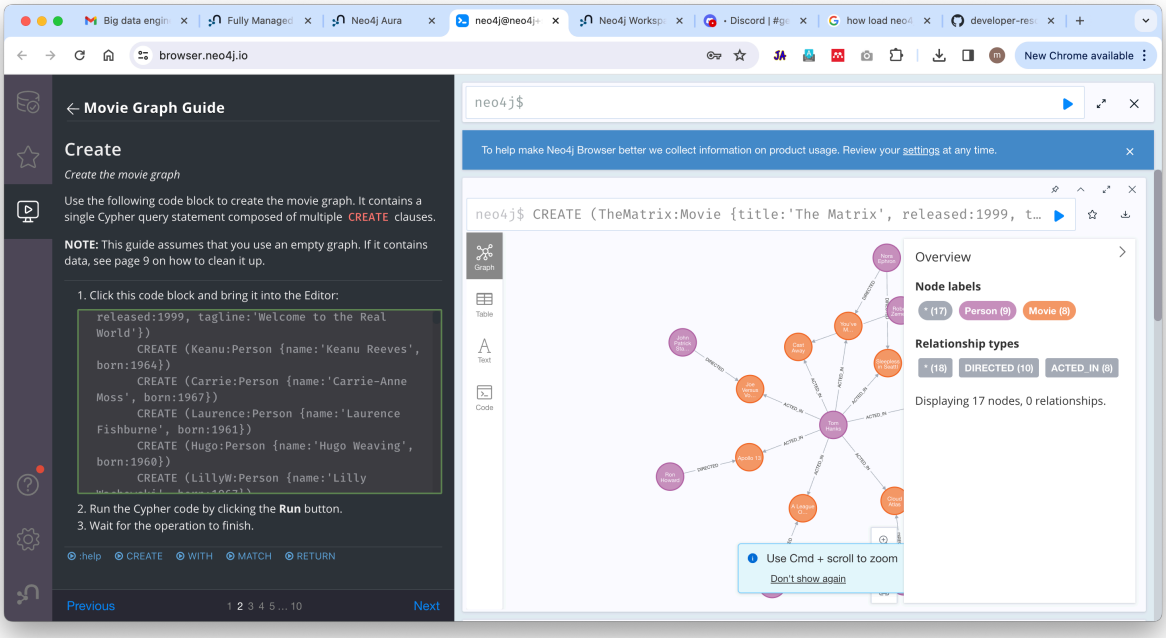
1. Go to neo4j.io
2. Set up Neo4j free account to use Aura DB.
3. Create new AuraDB instance with Movies dataset.

Alternative: Download and install Desktop version, then download free Movies dataset.
Insert screenshot of completed assignment here:
Creating Free AuraDB account





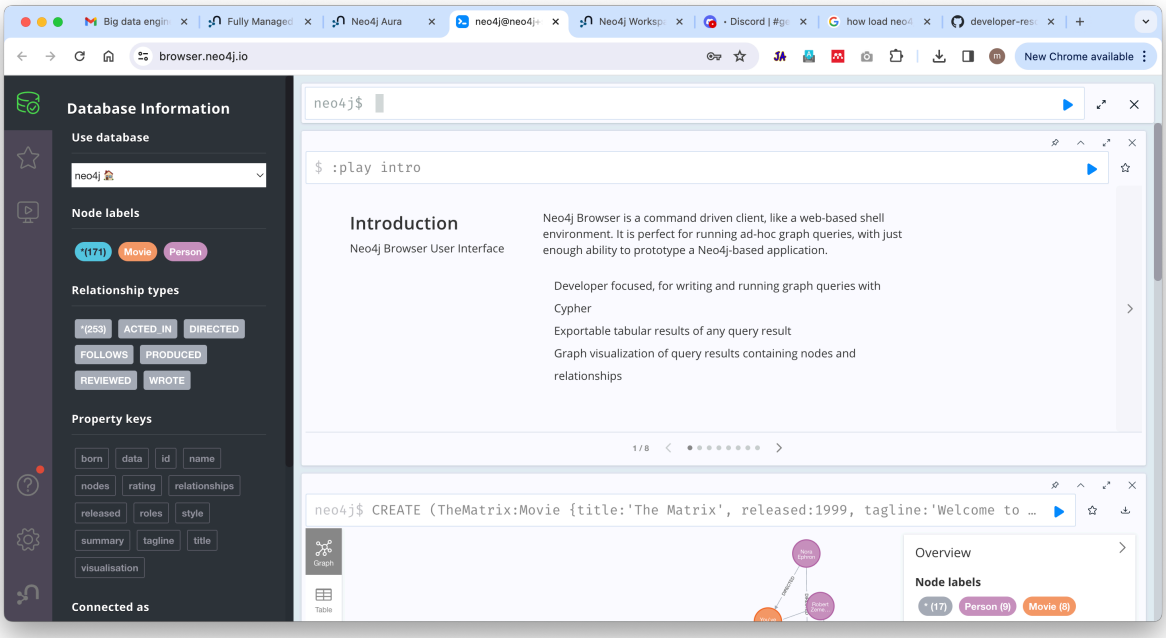
Create Movie dataset



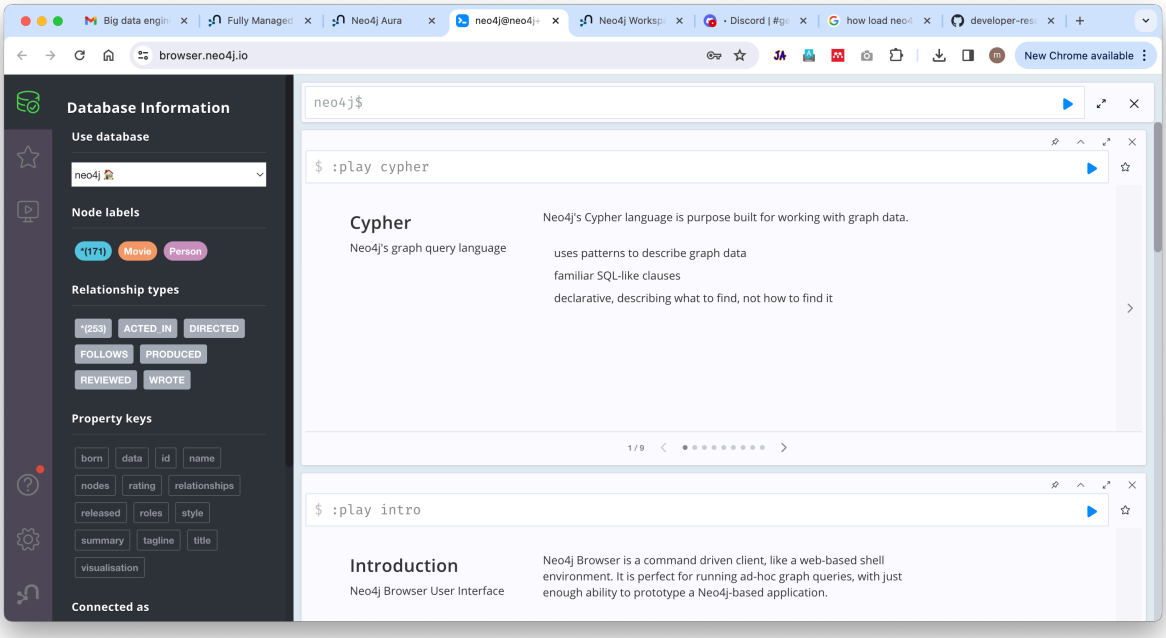
(points: 1)

TASK 2.

Refresh your impression of Neo4j with https://youtu.be/_D19h5s73Co
We'll be taking advantage of built-in learning materials. Type these commands in console and read information there:
$:play intro

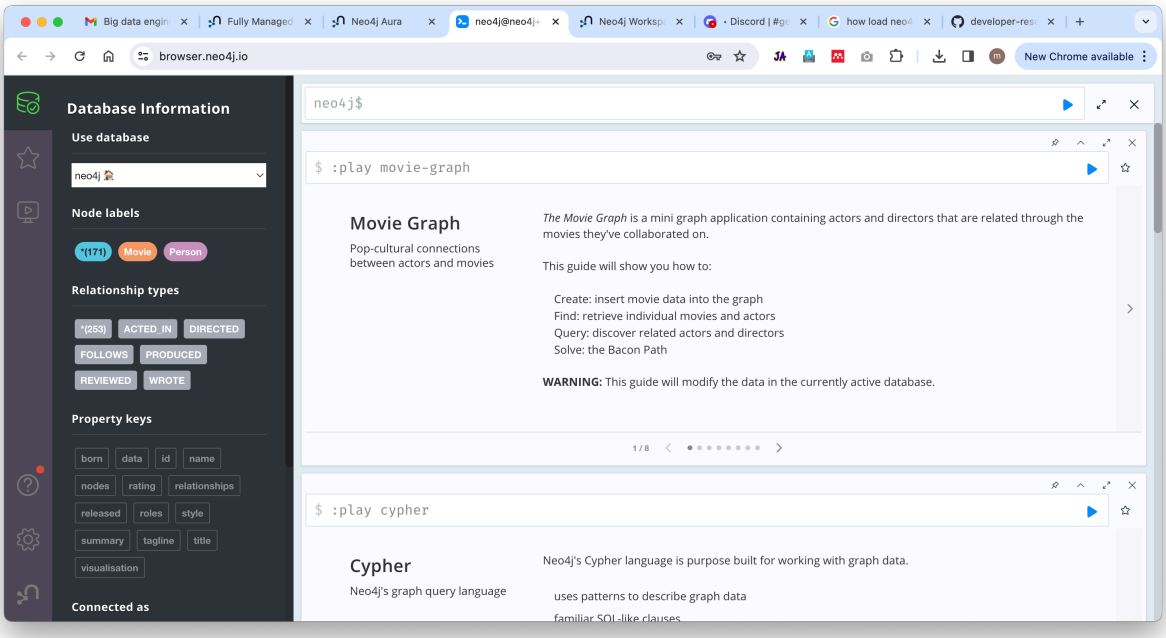


$:play cypher



Now time for practice - navigate to the “Movie Graph” section:

$:play movie-graph



We'll be working through the examples provided in this section for working with a graph of data relations between actors, directors and movies, with some additional tasks.

What is Cypher?

Cypher is a graph query language that is used to query the Neo4j Database. Just like you use SQL to query a MySQL database, you would use Cypher to query the Neo4j Database.

A simple cypher query can look something like this:

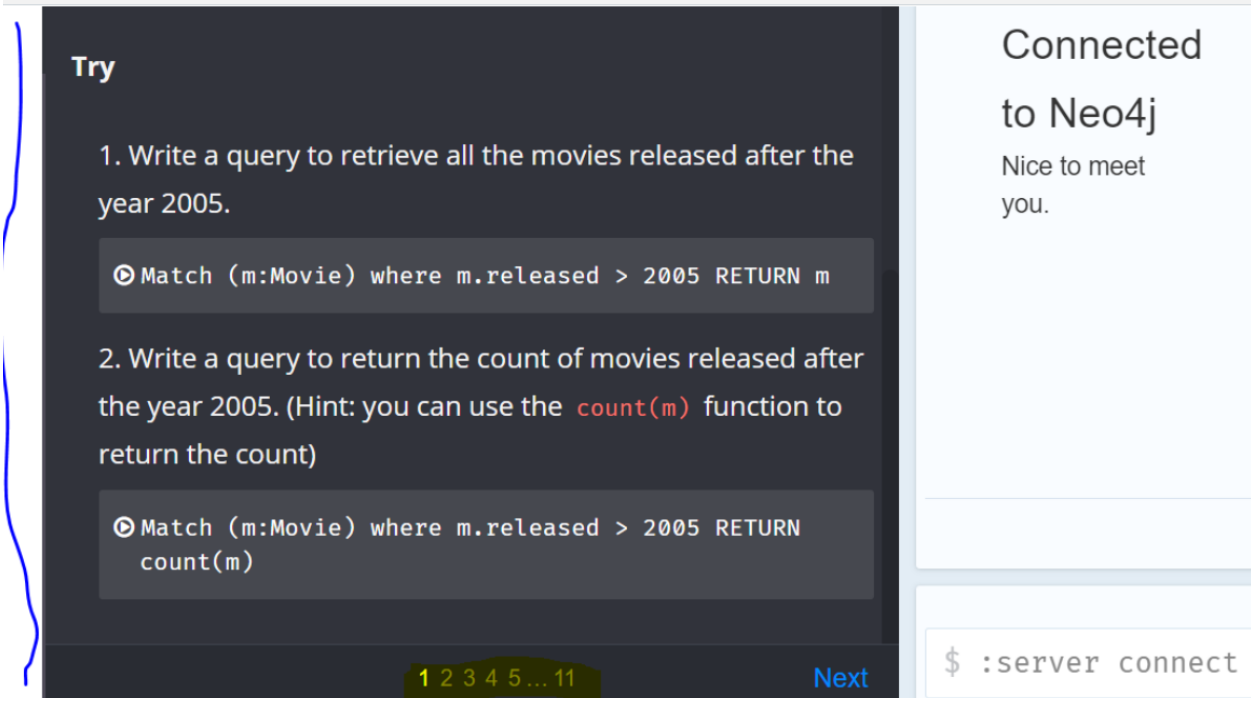
Match (m:Movie) where m.released > 2000 RETURN m limit 5

Please look at this video:

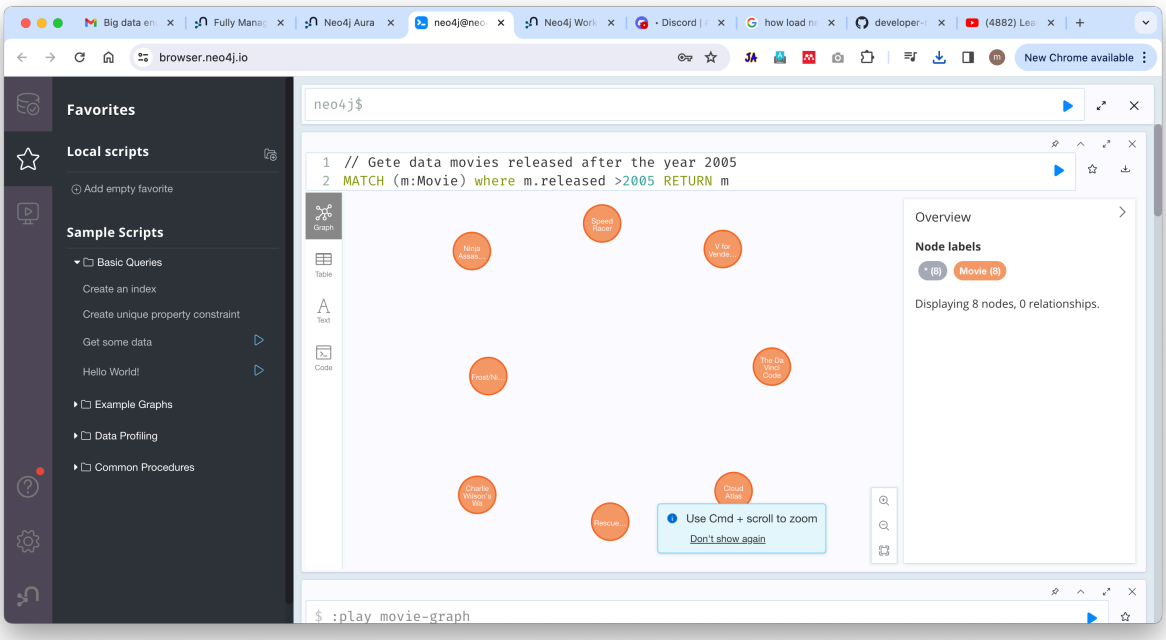
https://www.youtube.com/watch?v=3sgNbgRjZSU

Cypher tutorial starts automatically when you connect to Neo4j instance and choose to use Movies dataset.

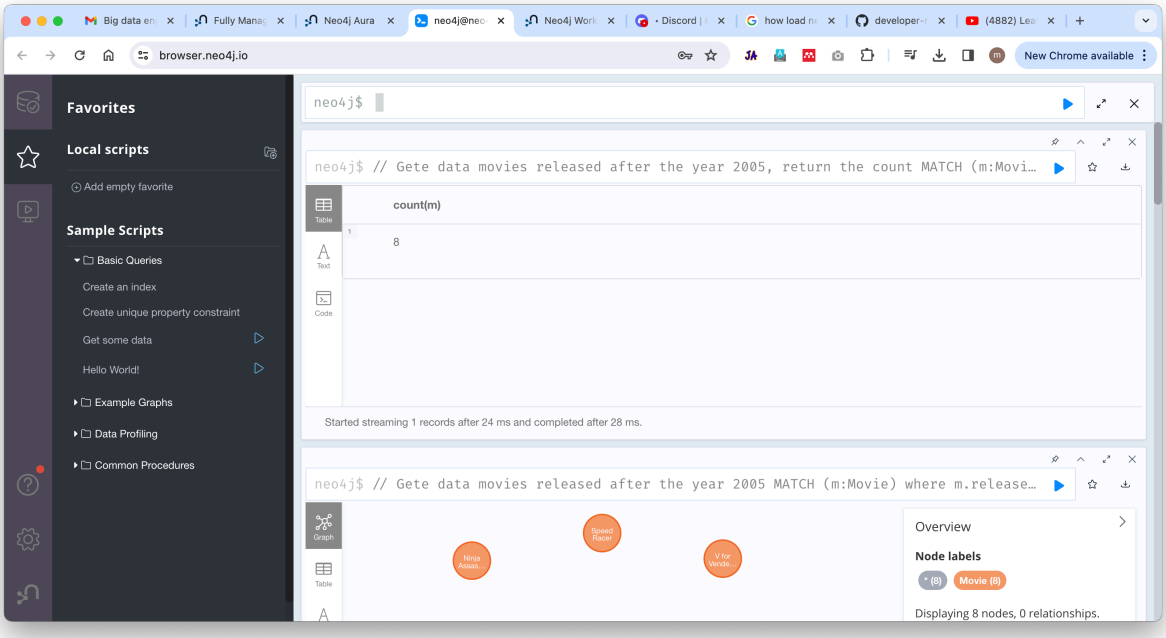
Finish Cypher tutorial.


Try
1. Write a query to retrieve all the movies released after the year 2005.
Match (m:Movie) where m.released > 2005 RETURN m
2. Write a query to return the count of movies released after the year 2005. (Hint: you can use the count(m) function to return the count)
Match (m:Movie) where m.released > 2005 RETURN count(m)
1 2 3 4 5 ... 11
Next
Connected to Neo4j
Nice to meet you.
$:server connect

QUERY 01


browser.neo4j.io
New Chrome available
Favorites
Local scripts
Add empty favorite
Sample Scripts
Basic Queries
Create an index
Create unique property constraint
Get some data
Hello World!
Example Graphs
Data Profiling
Common Procedures
neo4j$
// Gete data movies released after the year 2005
MATCH (m:Movie) where m.released >2005 RETURN m
Graph
Table
Text
Code
Overview
Node labels
Movie (8)
Displaying 8 nodes, 0 relationships.
Use Cmd + scroll to zoom
Don't show again
$:play movie-graph

QUERY 02



What are advantages and disadvantages of Cypher

Advantages	Disadvantages
Pattern matching - Easy to write queries that match specific structures within the graph. **Readability** **Traversal and path finding - Cypher provides efficient syntax for traversing relationships and finding paths within the graph**	**Learning of Cypher – It take much time to deep dive into advanced query.** **Less support and difference in outside the graph database when considering neo4j and other systems.** **Performance consideration when writing queries (Different from each other)**

TASK 3.

What differences did you notice between CYPHER and SQL?
Paste your answer here: I have identify several differences in CYPHER and SQL in different criteria.

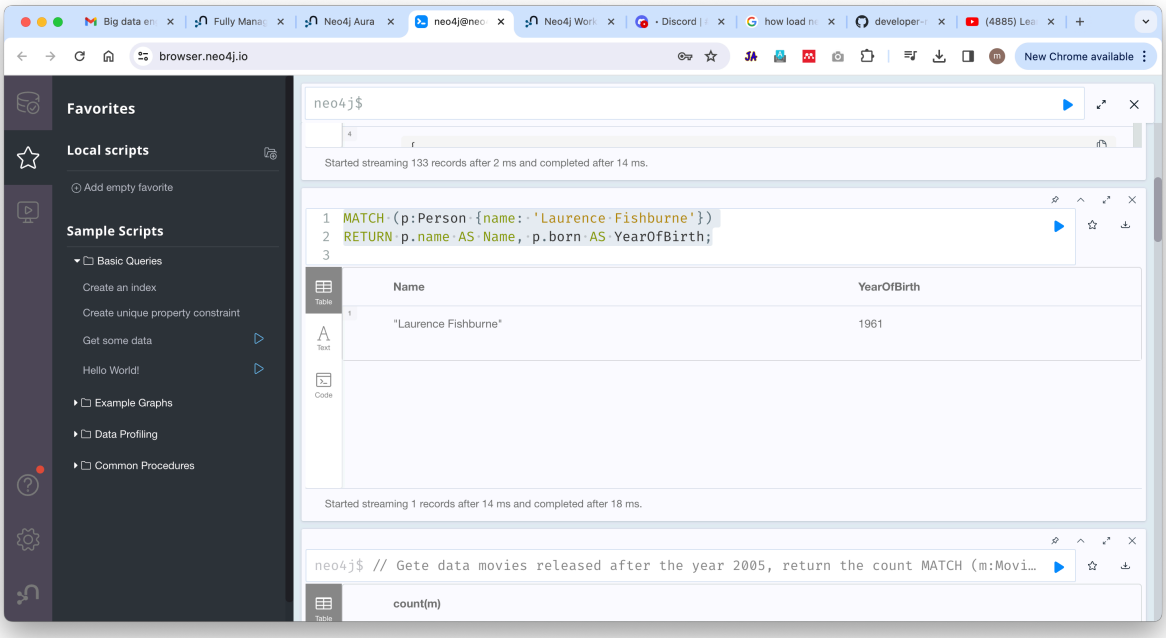
Data model	Cypher Designed for querying graph databases that use the property graph model but in SQL Designed for querying relational databases that use tables with rows and columns. It is well-suited for working with structured data
Syntax	Cypher syntax more similar to Uses ASCII art-like syntax to represent patterns in the graph but in SQL it based on keywords and Sequence of using keywords
Usecase	Primarily used for querying graph databases where relationships are as important as the entities themselves. Commonly used in scenarios involving social networks, recommendation engines While SQL used for querying relational databases where structured tabular data is the norm. Commonly used in business applications

TASK 4.

Create and execute the query that returns name and year of birth of one person of your choosing.

Paste the query and output as an answer.

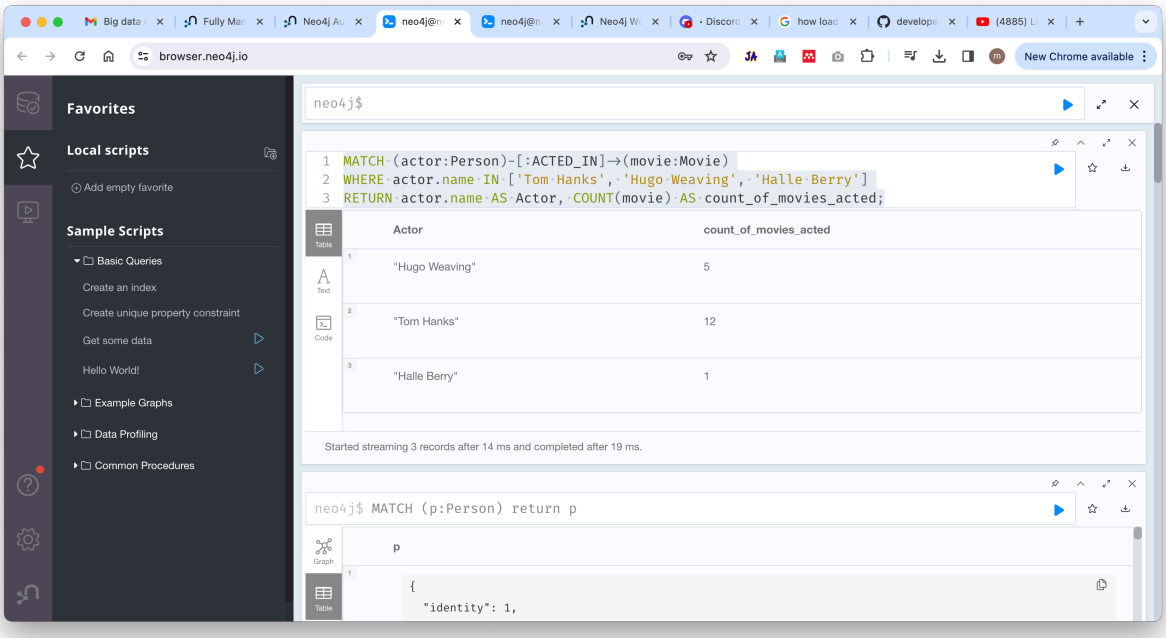
```
MATCH (p:Person {name: 'Laurence Fishburne'})
RETURN p.name AS Name, p.born AS YearOfBirth;
```



TASK 5.

Query DB and check in how many movies acted Tom Hanks, Hugo Weaving, Halle Berrie

```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)
WHERE actor.name IN ['Tom Hanks', 'Hugo Weaving', 'Halle Berry']
RETURN actor.name AS Actor, COUNT(movie) AS count_of_movies_acted;
```

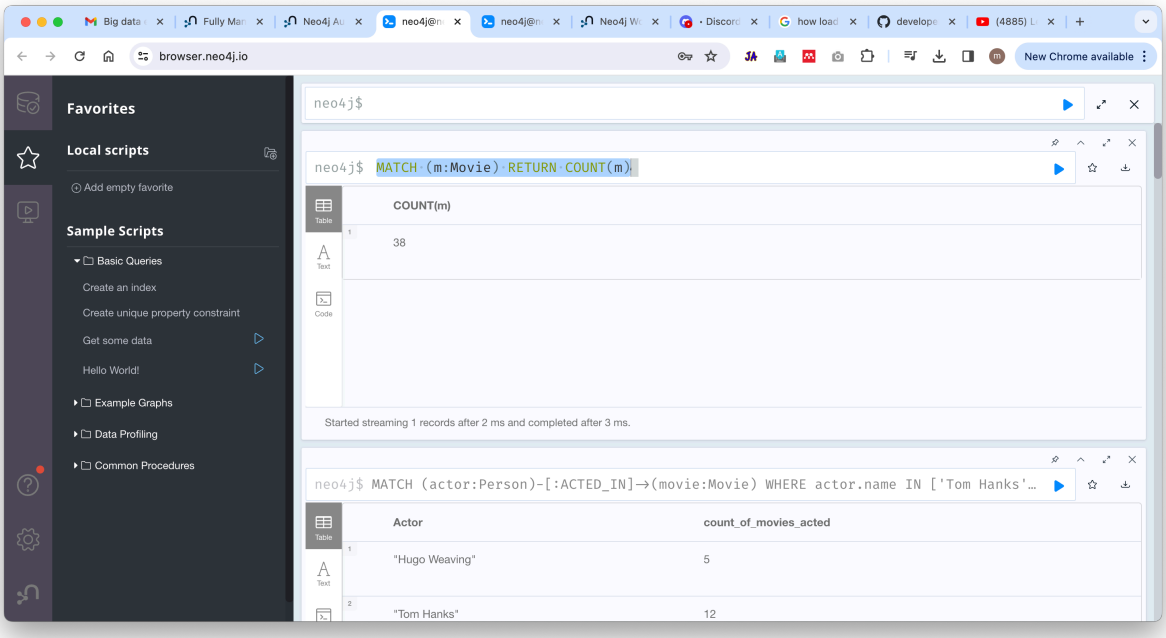


Paste the query and output as an answer.

TASK 6.

How many movies are in DB?

MATCH (m:Movie) RETURN COUNT(m)

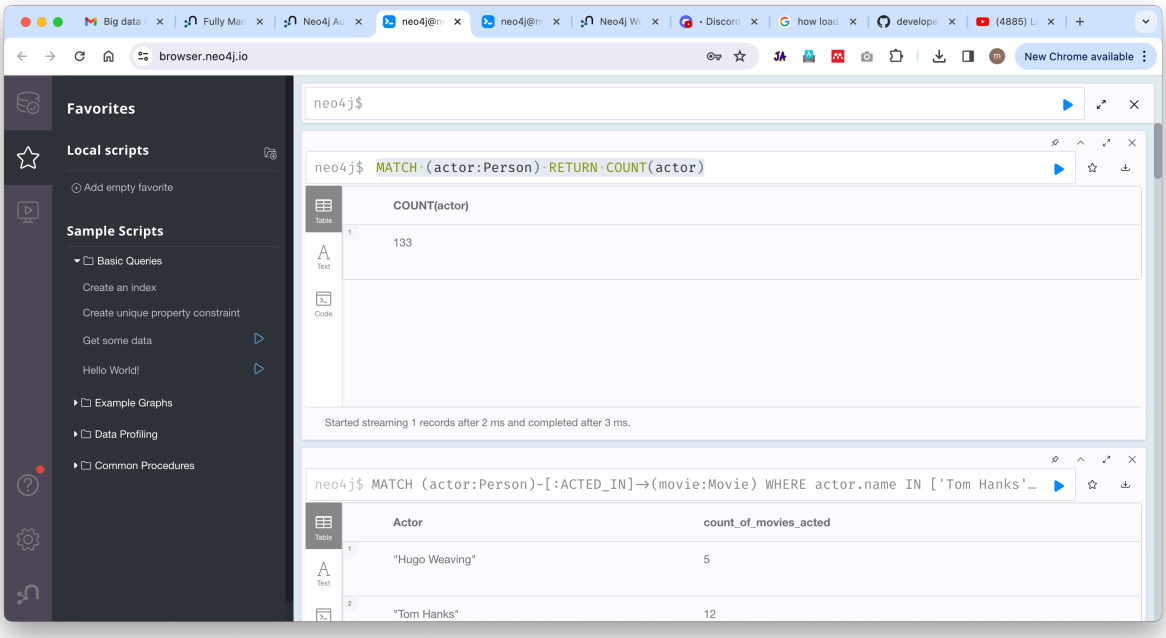


Paste the query and output as an answer.

TASK 7.

How many actors are in DB?

MATCH (actor:Person) RETURN COUNT(actor)



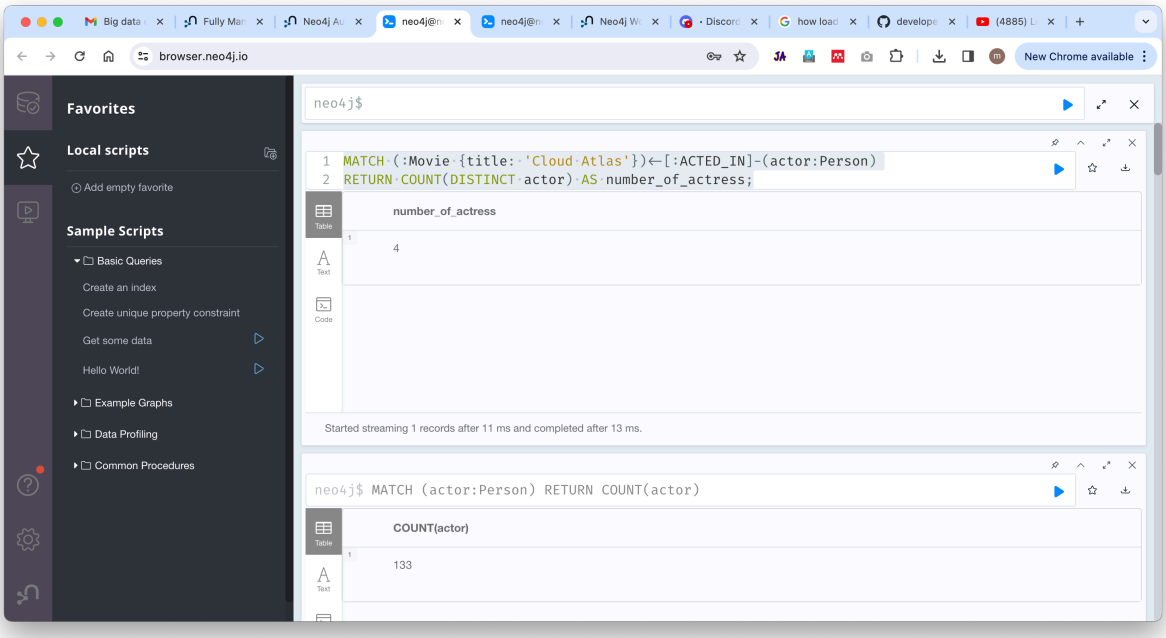
Paste the query and output as an answer.

TASK 8.

How many actors played in Cloud Atlas?

Paste the query and output as an answer.

```
MATCH (:Movie {title: 'Cloud Atlas'})<-[:ACTED_IN]-(actor:Person)
RETURN COUNT(DISTINCT actor) AS number_of_actress;
```

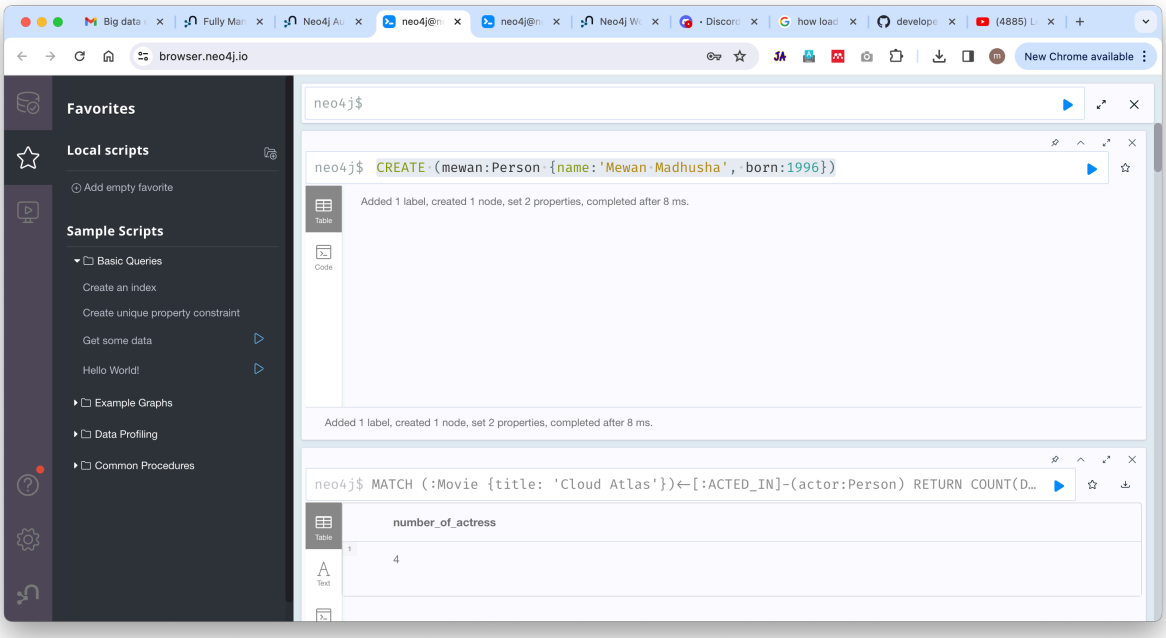


TASK 9.

Add node with any new name to the Database.

Paste the query and output as an answer.

```
CREATE (mewan:Person {name:'Mewan Madhusha', born:1996})
```

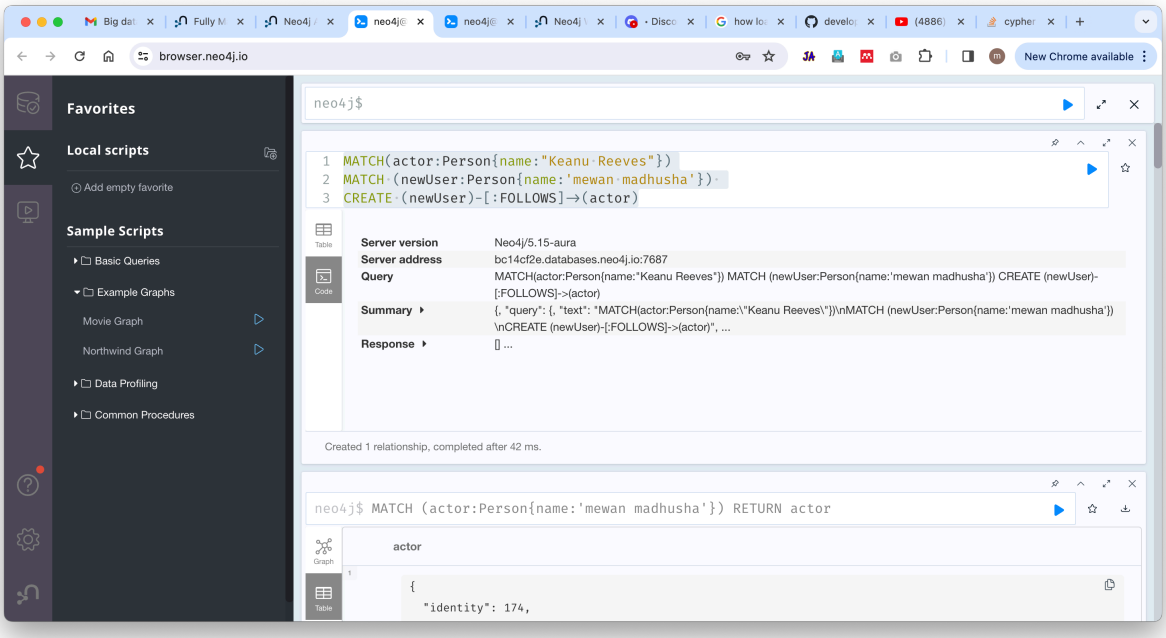


TASK 10.

Add connection between user created name and favorite actor

Paste the query and output as an answer.

```
MATCH(actor:Person{name:"Keanu Reeves"})
MATCH (newUser:Person{name:'mewan madhusha'})
CREATE (newUser)-[:FOLLOWS]->(actor)
```

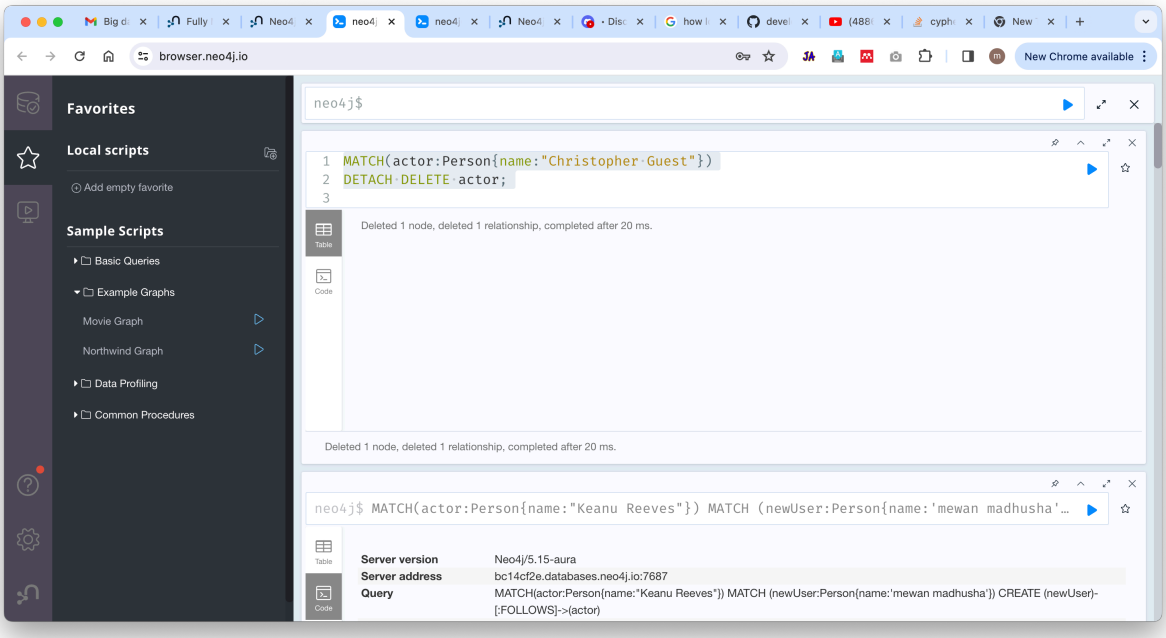


TASK 11.

Remove any actor from DB

Paste the query and output as an answer.

```
MATCH(actor:Person{name:"Christopher Guest"})
DETACH DELETE actor;
```

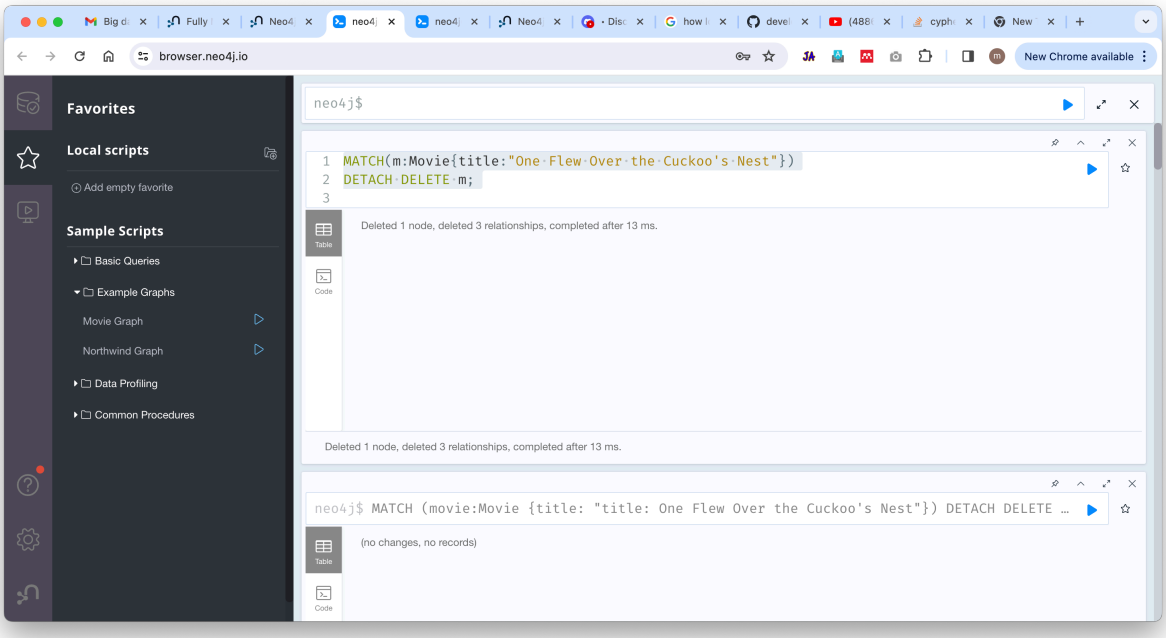


TASK 12.

Remove any movie from DB

Paste the query and output as an answer.

```
MATCH(m:Movie{title:"One Flew Over the Cuckoo's Nest"})
DETACH DELETE m;
```

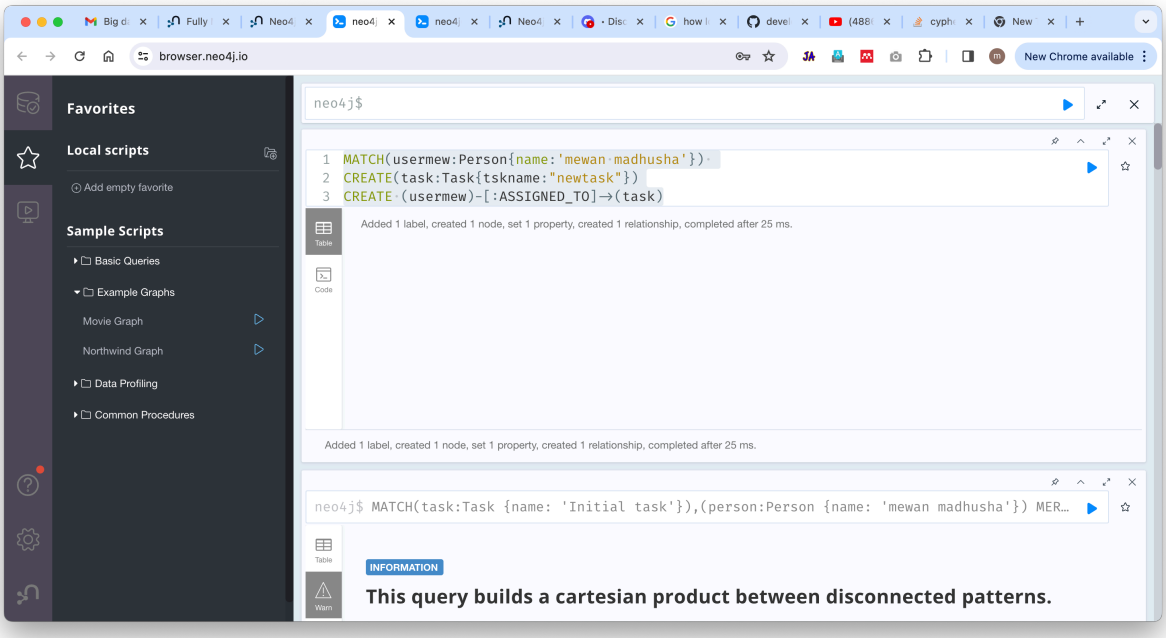


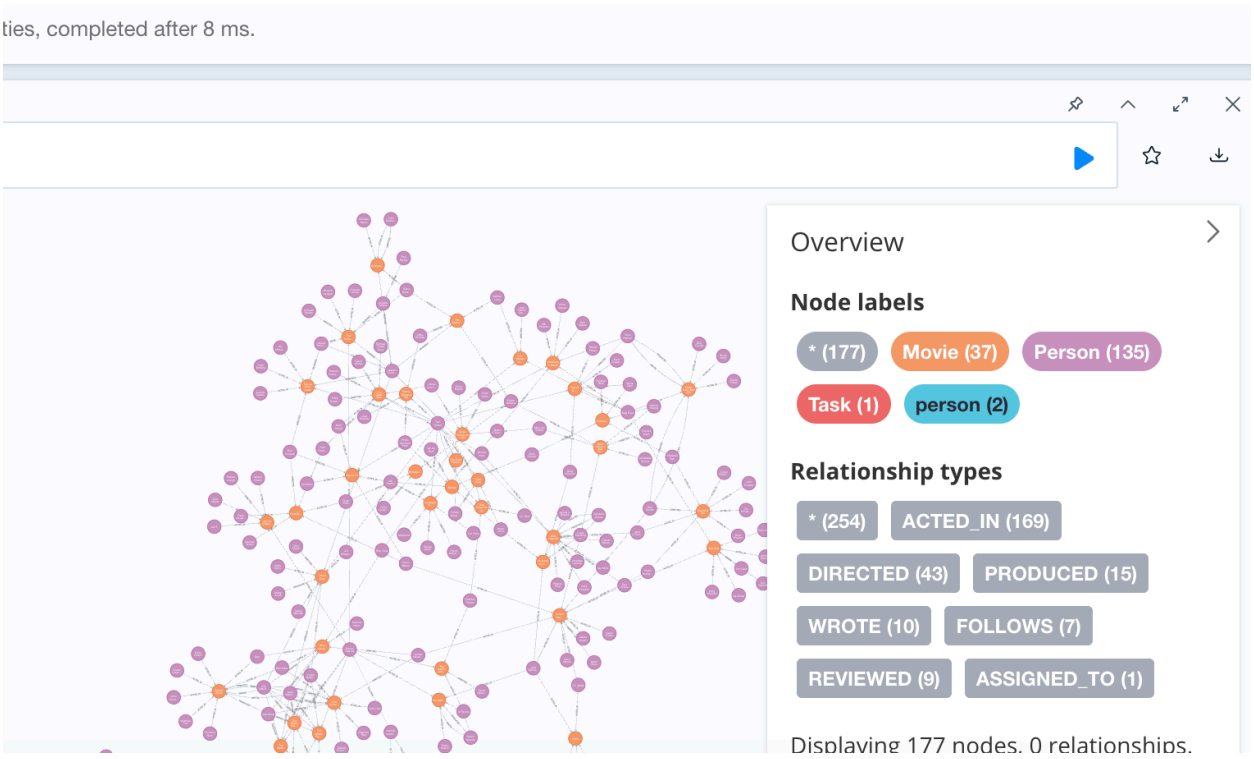
TASK 13.

Extend graph with new information. To do so please attach node from Neo4j TASK to the database with the relationships. If relationships are not there yet, please create them.

Provide code and cypher output as the result.

```
MATCH(usermew:Person{name:'mewan madhusha'})
CREATE(task:Task{tskname:"newtask"})
CREATE (usermew)-[:ASSIGNED_TO]->(task)
```


browser.neo4j.io
New Chrome available
Favorites
Local scripts
Add empty favorite
Sample Scripts
Basic Queries
Example Graphs
Movie Graph
Northwind Graph
Data Profiling
Common Procedures
neo4j$
1 MATCH(usermew:Person{name:'mewan madhusha'})
2 CREATE(task:Task{tskname:"newtask"})
3 CREATE (usermew)-[:ASSIGNED_TO]→(task)
Table
Code
Added 1 label, created 1 node, set 1 property, created 1 relationship, completed after 25 ms.
Added 1 label, created 1 node, set 1 property, created 1 relationship, completed after 25 ms.
neo4j$ MATCH(task:Task {name: 'Initial task'}),(person:Person {name: 'mewan madhusha'}) MER…
Table
Warn
INFORMATION
This query builds a cartesian product between disconnected patterns.


ties, completed after 8 ms.
Overview
Node labels
* (177)
Movie (37)
Person (135)
Task (1)
person (2)
Relationship types
* (254)
ACTED_IN (169)
DIRECTED (43)
PRODUCED (15)
WROTE (10)
FOLLOWS (7)
REVIEWED (9)
ASSIGNED_TO (1)
Displaying 177 nodes, 0 relationships.

TASK 14.

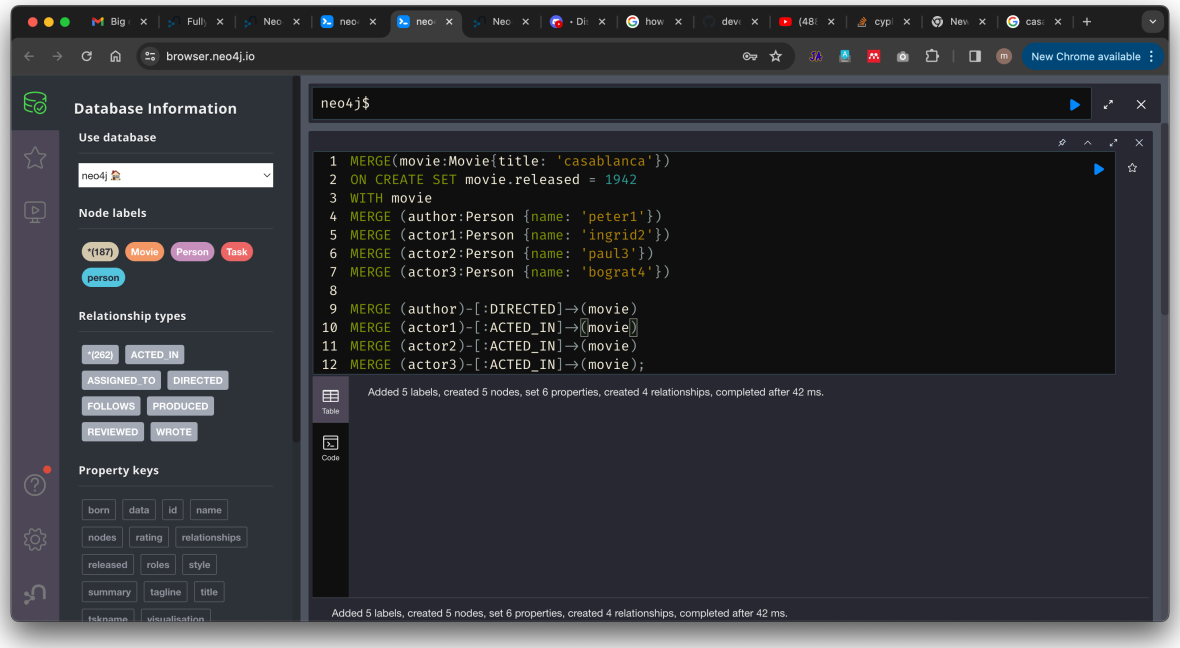
Express a repeatable (idempotent) update operation. Check if movie Casablanca exists. If not exists - create one with corresponding relationships and add authors / actors as well. Movie should have at least one author and three actors.

NB! Use the most appropriate CYPHER feature to cover this task.

As a result please provide query and output. Explain why you have used particular clause(s).

```
MERGE(movie:Movie{title: 'casablanca'})
ON CREATE SET movie.released = 1942
WITH movie
MERGE (author:Person {name: 'peter1'})
MERGE (actor1:Person {name: 'ingrid2'})
MERGE (actor2:Person {name: 'paul3'})
MERGE (actor3:Person {name: 'bograt4'})

MERGE (author)-[:DIRECTED]->(movie)
MERGE (actor1)-[:ACTED_IN]->(movie)
MERGE (actor2)-[:ACTED_IN]->(movie)
MERGE (actor3)-[:ACTED_IN]->(movie);
```



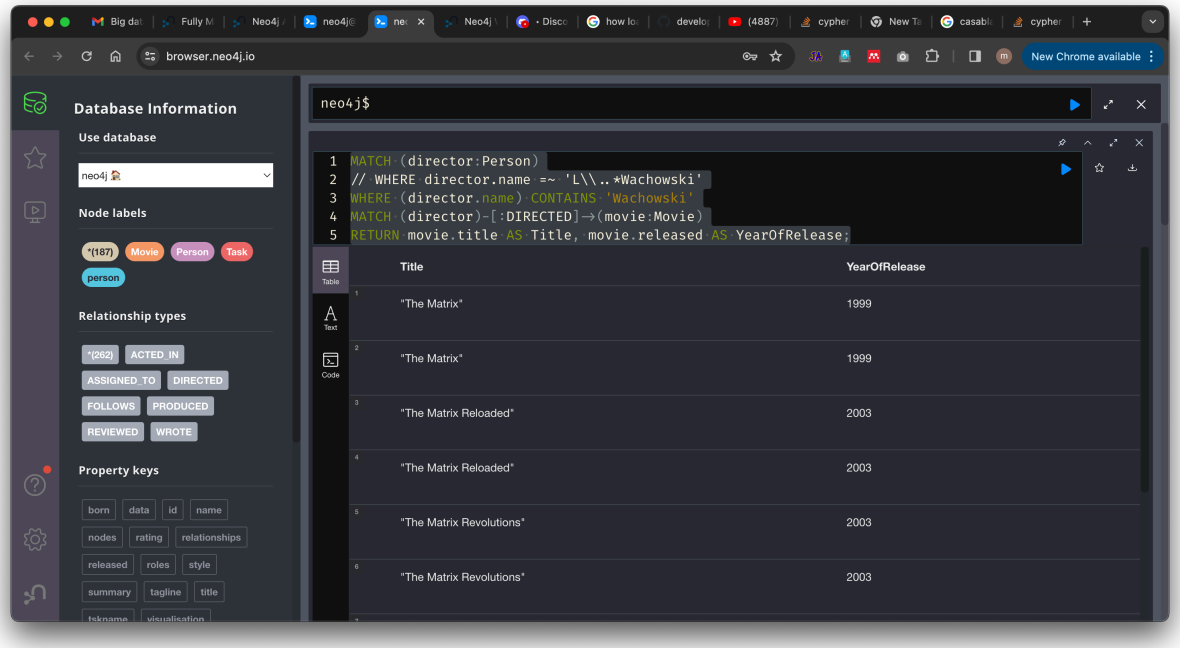
The MERGE (movie:Movie {title: 'Casablanca'}) checks if the movie node with the title 'Casablanca' exists. If it doesn't, it creates one. And then adding released data properties also if the movie node created , to do that we can use The ON CREATE SET movie.year = 1942, It's create only if new movie create. Apart from that other merge statements ensure that author and actor nodes exist, and the relationships (DIRECTED and ACTED_IN) are created between them and the movie node.

TASK 15.

Build a Cypher statement to find the title and year of release for every :Movie that L.Wachowski has :DIRECTED. Try to use CYPHER practices you already know to identify proper director name.

Provide query and output.

```
MATCH (director:Person)
// WHERE director.name =~ 'L\\..*Wachowski'
WHERE (director.name) CONTAINS 'Wachowski'
MATCH (director)-[:DIRECTED]->(movie:Movie)
RETURN movie.title AS Title, movie.released AS YearOfRelease;
```


browser.neo4j.io
New Chrome available
Database Information
Use database
neo4j
Node labels
*(187)
Movie
Person
Task
person
Relationship types
*(282)
ACTED_IN
ASSIGNED_TO
DIRECTED
FOLLOWS
PRODUCED
REVIEWED
WROTE
Property keys
born
data
id
name
nodes
rating
relationships
released
roles
style
summary
tagline
title
neo4j$
1 MATCH (director:Person)
2 // WHERE director.name =~ 'L\\..*Wachowski'
3 WHERE (director.name) CONTAINS 'Wachowski'
4 MATCH (director)-[:DIRECTED]→(movie:Movie)
5 RETURN movie.title AS Title, movie.released AS YearOfRelease;
Table
Text
Code
Title
YearOfRelease
1 "The Matrix" 1999
2 "The Matrix" 1999
3 "The Matrix Reloaded" 2003
4 "The Matrix Reloaded" 2003
5 "The Matrix Revolutions" 2003
6 "The Matrix Revolutions" 2003