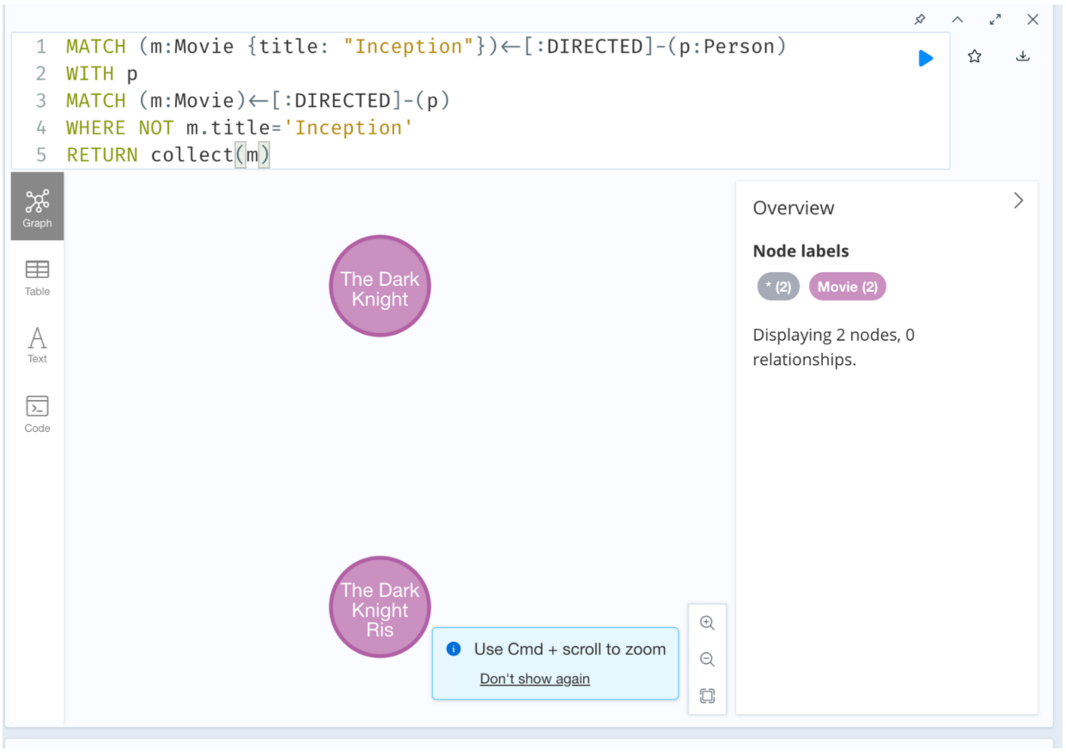
SWEN432 Assignment 1

Question 1

(a) Queries:

```
MATCH (m:Movie {title: "Inception"})<-[:DIRECTED]-(p:Person)
WITH p
MATCH (m:Movie)<-[:DIRECTED]-(p)
WHERE NOT m.title='Inception'
RETURN collect(m)
```

Screenshots:



```
MATCH (m:Movie {title: "Inception"})←[:DIRECTED]-(p:Person)
WITH p
MATCH (m:Movie)←[:DIRECTED]-(p)
WHERE NOT m.title='Inception'
RETURN collect(m)
```

Overview

Node labels

* (2) Movie (2)

Displaying 2 nodes, 0 relationships.

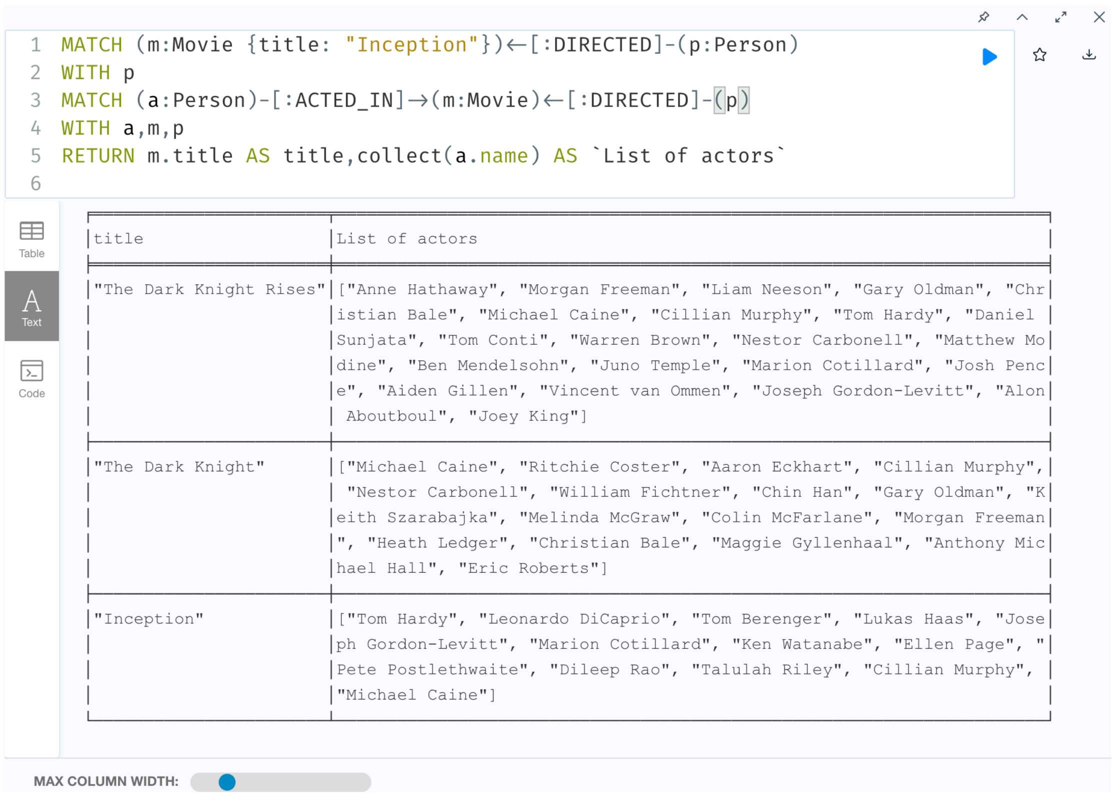
Use Cmd + scroll to zoom

Don't show again

(b) Queries:

```
MATCH (m:Movie {title: "Inception"})<-[:DIRECTED]-(p:Person)
WITH p
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(p)
WITH a, m, p
RETURN m.title AS title, collect(a.name) AS `List of actors`
```

Table and Screenshots:



```
MATCH (m:Movie {title: "Inception"})←[:DIRECTED]-(p:Person)
WITH p
MATCH (a:Person)-[:ACTED_IN]→(m:Movie)←[:DIRECTED]-(p)
WITH a,m,p
RETURN m.title AS title,collect(a.name) AS `List of actors`
```

```
╒════════════════════════╤════════════════════════════════════════════════════════════════════════╕
│title                   │List of actors                                                          │
╞════════════════════════╪════════════════════════════════════════════════════════════════════════╡
│"The Dark Knight Rises" │["Anne Hathaway", "Morgan Freeman", "Liam Neeson", "Gary Oldman", "Chr│
│                        │istian Bale", "Michael Caine", "Cillian Murphy", "Tom Hardy", "Daniel │
│                        │Sunjata", "Tom Conti", "Warren Brown", "Nestor Carbonell", "Matthew Mo│
│                        │dine", "Ben Mendelsohn", "Juno Temple", "Marion Cotillard", "Josh Penc│
│                        │e", "Aiden Gillen", "Vincent van Ommen", "Joseph Gordon-Levitt", "Alon│
│                        │ Aboutboul", "Joey King"]                                               │
├────────────────────────┼────────────────────────────────────────────────────────────────────────┤
│"The Dark Knight"       │["Michael Caine", "Ritchie Coster", "Aaron Eckhart", "Cillian Murphy",│
│                        │ "Nestor Carbonell", "William Fichtner", "Chin Han", "Gary Oldman", "K│
│                        │eith Szarabajka", "Melinda McGraw", "Colin McFarlane", "Morgan Freeman│
│                        │", "Heath Ledger", "Christian Bale", "Maggie Gyllenhaal", "Anthony Mic│
│                        │hael Hall", "Eric Roberts"]                                             │
├────────────────────────┼────────────────────────────────────────────────────────────────────────┤
│"Inception"             │["Tom Hardy", "Leonardo DiCaprio", "Tom Berenger", "Lukas Haas", "Jose│
│                        │ph Gordon-Levitt", "Marion Cotillard", "Ken Watanabe", "Ellen Page", "│
│                        │Pete Postlethwaite", "Dileep Rao", "Talulah Riley", "Cillian Murphy", │
│                        │"Michael Caine"]                                                        │
└────────────────────────┴────────────────────────────────────────────────────────────────────────┘
```

(c) Queries:

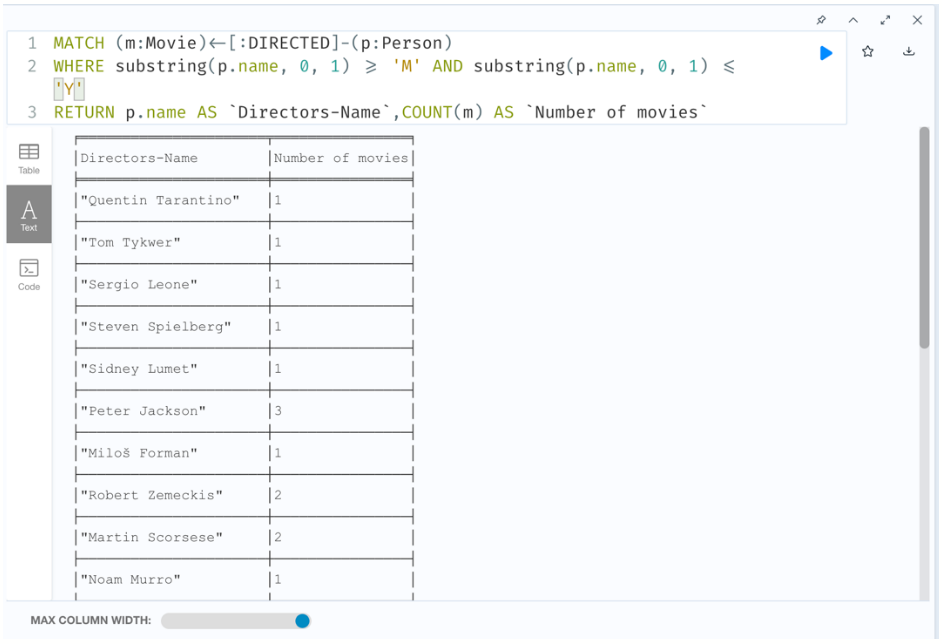
```
MATCH (m:Movie)<-[:DIRECTED]-(p:Person)
WHERE substring(p.name, 0, 1) >= 'M' AND substring(p.name, 0, 1) <= 'Y'
RETURN p.name AS `Directors-Name`,COUNT(m) AS `Number of movies`
```

Table:

Directors-Name	Number of movies
"Quentin Tarantino"	1
"Tom Tykwer"	1
"Sergio Leone"	1
"Steven Spielberg"	1
"Sidney Lumet"	1
"Peter Jackson"	3
"Miloš Forman"	1
"Robert Zemeckis"	2
"Martin Scorsese"	2
"Noam Murro"	1
"Phil Lord"	1
"Spike Jonze"	1
"Steve McQueen"	1
"Neill Blomkamp"	1
"Rob Minkoff"	1
"Tommy Wirkola"	1
"Tony Bancroft"	1

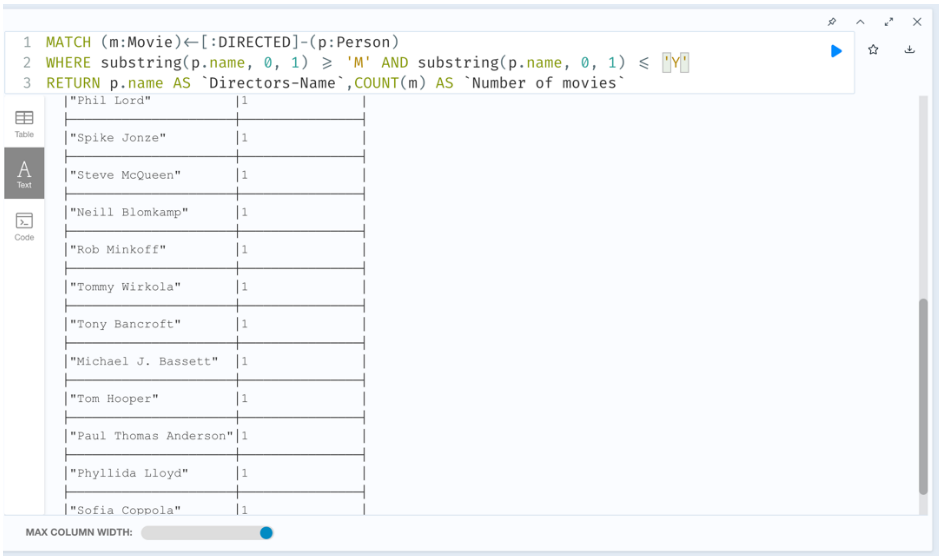
"Michael J. Bassett"	1
"Tom Hooper"	1
"Paul Thomas Anderson"	1
"Phyllida Lloyd"	1
"Sofia Coppola"	1

Screenshots:



```
MATCH (m:Movie)←[:DIRECTED]-(p:Person)
WHERE substring(p.name, 0, 1) ≥ 'M' AND substring(p.name, 0, 1) ≤
'Y'
RETURN p.name AS `Directors-Name`,COUNT(m) AS `Number of movies`
```

```
│Directors-Name         │Number of movies│
├───────────────────────┼────────────────┤
│"Quentin Tarantino"    │1               │
├───────────────────────┼────────────────┤
│"Tom Tykwer"           │1               │
├───────────────────────┼────────────────┤
│"Sergio Leone"         │1               │
├───────────────────────┼────────────────┤
│"Steven Spielberg"     │1               │
├───────────────────────┼────────────────┤
│"Sidney Lumet"         │1               │
├───────────────────────┼────────────────┤
│"Peter Jackson"        │3               │
├───────────────────────┼────────────────┤
│"Miloš Forman"         │1               │
├───────────────────────┼────────────────┤
│"Robert Zemeckis"      │2               │
├───────────────────────┼────────────────┤
│"Martin Scorsese"      │2               │
├───────────────────────┼────────────────┤
│"Noam Murro"           │1               │
```



```
MATCH (m:Movie)←[:DIRECTED]-(p:Person)
WHERE substring(p.name, 0, 1) ≥ 'M' AND substring(p.name, 0, 1) ≤ 'Y'
RETURN p.name AS `Directors-Name`,COUNT(m) AS `Number of movies`
```

```
│"Phil Lord"           │1               │
├──────────────────────┼────────────────┤
│"Spike Jonze"         │1               │
├──────────────────────┼────────────────┤
│"Steve McQueen"       │1               │
├──────────────────────┼────────────────┤
│"Neill Blomkamp"      │1               │
├──────────────────────┼────────────────┤
│"Rob Minkoff"         │1               │
├──────────────────────┼────────────────┤
│"Tommy Wirkola"       │1               │
├──────────────────────┼────────────────┤
│"Tony Bancroft"       │1               │
├──────────────────────┼────────────────┤
│"Michael J. Bassett"  │1               │
├──────────────────────┼────────────────┤
│"Tom Hooper"          │1               │
├──────────────────────┼────────────────┤
│"Paul Thomas Anderson"│1               │
├──────────────────────┼────────────────┤
│"Phyllida Lloyd"      │1               │
├──────────────────────┼────────────────┤
│"Sofia Coppola"       │1               │
```

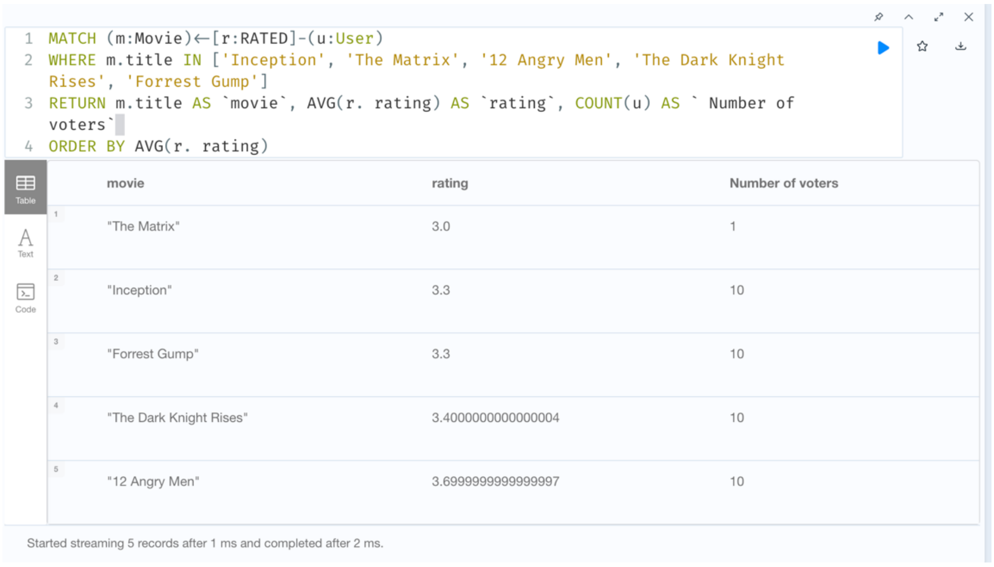
(d) Regarding the sorting in this question, it is assumed that the movies are sorted according to their average rating. This is also how most movie sites are sorted.

Queries:

```
MATCH (m:Movie)<-[r:RATED]-(u:User)
```

WHERE m.title IN ['Inception', 'The Matrix', '12 Angry Men', 'The Dark Knight Rises', 'Forrest Gump']
RETURN m.title AS `movie`, AVG(r. rating) AS `rating`, COUNT(u) AS ` Number of voters`
ORDER BY AVG(r. rating)

Screenshots:



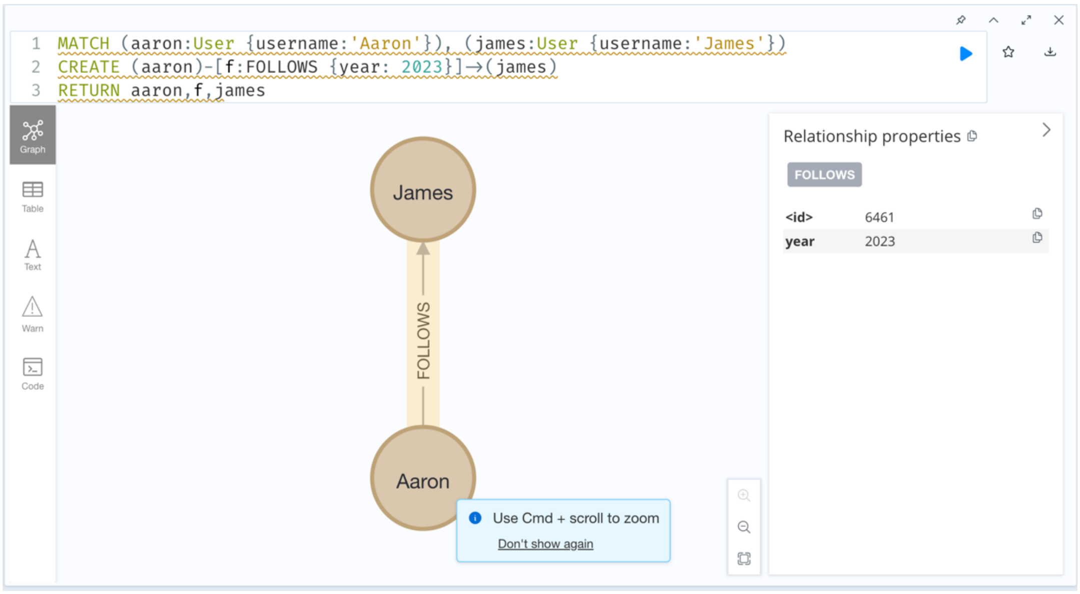
```
MATCH (m:Movie)←[r:RATED]-(u:User)
WHERE m.title IN ['Inception', 'The Matrix', '12 Angry Men', 'The Dark Knight
Rises', 'Forrest Gump']
RETURN m.title AS `movie`, AVG(r. rating) AS `rating`, COUNT(u) AS ` Number of
voters`
ORDER BY AVG(r. rating)
```

	movie	rating	Number of voters
1	"The Matrix"	3.0	1
2	"Inception"	3.3	10
3	"Forrest Gump"	3.3	10
4	"The Dark Knight Rises"	3.4000000000000004	10
5	"12 Angry Men"	3.6999999999999997	10

Started streaming 5 records after 1 ms and completed after 2 ms.

(e) Queries:
MATCH (aaron:User {username:'Aaron'}), (james:User {username:'James'})
CREATE (aaron)-[f:FOLLOWS {year: 2023}]->(james)
RETURN aaron,f,james

Screenshots:



```
MATCH (aaron:User {username:'Aaron'}), (james:User {username:'James'})
CREATE (aaron)-[f:FOLLOWS {year: 2023}]→(james)
RETURN aaron,f,james
```

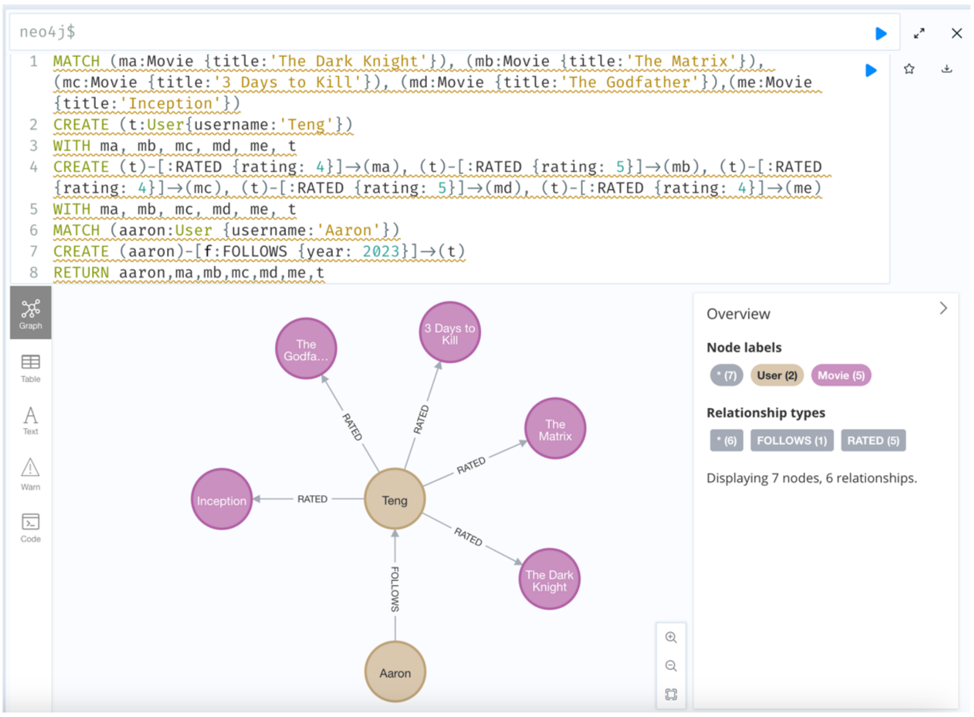
Relationship properties

FOLLOWS

<id>	6461
year	2023

(f) MATCH (ma:Movie {title:'The Dark Knight'}), (mb:Movie {title:'The Matrix'}), (mc:Movie {title:'3 Days to Kill'}), (md:Movie {title:'The Godfather'}),(me:Movie {title:'Inception'})
CREATE (t:User{username:'Teng'})
WITH ma, mb, mc, md, me, t
CREATE (t)-[:RATED {rating: 4}]->(ma), (t)-[:RATED {rating: 5}]->(mb), (t)-[:RATED {rating: 4}]->(mc), (t)-[:RATED {rating: 5}]->(md), (t)-[:RATED {rating: 4}]->(me)
WITH ma, mb, mc, md, me, t
MATCH (aaron:User {username:'Aaron'})
CREATE (aaron)-[f:FOLLOWS {year: 2023}]->(t)
RETURN aaron,ma,mb,mc,md,me,t

Screenshots:



```
neo4j$
MATCH (ma:Movie {title:'The Dark Knight'}), (mb:Movie {title:'The Matrix'}),
(mc:Movie {title:'3 Days to Kill'}), (md:Movie {title:'The Godfather'}),(me:Movie
{title:'Inception'})
CREATE (t:User{username:'Teng'})
WITH ma, mb, mc, md, me, t
CREATE (t)-[:RATED {rating: 4}]→(ma), (t)-[:RATED {rating: 5}]→(mb), (t)-[:RATED
{rating: 4}]→(mc), (t)-[:RATED {rating: 5}]→(md), (t)-[:RATED {rating: 4}]→(me)
WITH ma, mb, mc, md, me, t
MATCH (aaron:User {username:'Aaron'})
CREATE (aaron)-[f:FOLLOWS {year: 2023}]→(t)
RETURN aaron,ma,mb,mc,md,me,t
```

Overview

Node labels

* (7) User (2) Movie (5)

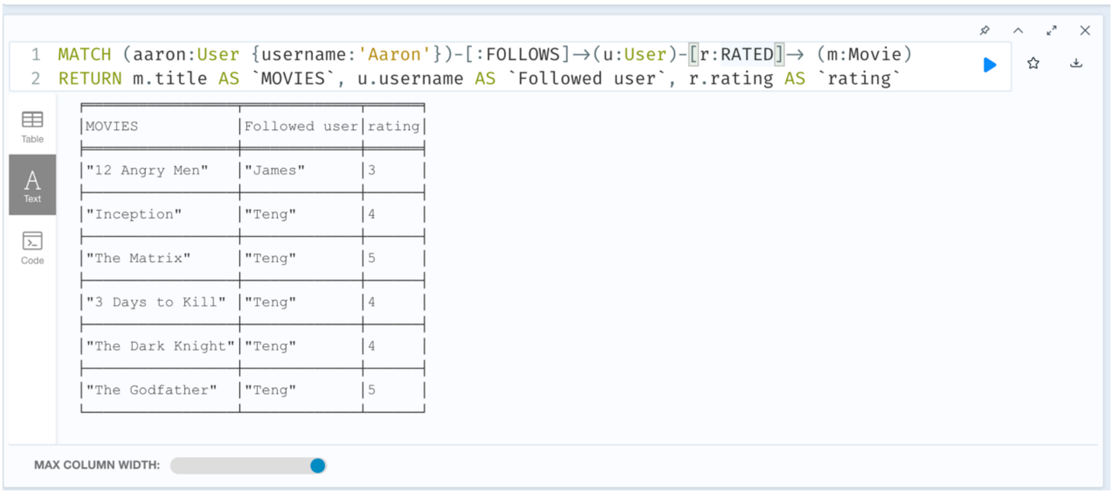
Relationship types

* (6) FOLLOWS (1) RATED (5)

Displaying 7 nodes, 6 relationships.

(g) MATCH (aaron:User {username:'Aaron'})-[:FOLLOWS]->(u:User)-[r:RATED]->(m:Movie)
RETURN m.title AS `MOVIES`, u.username AS `Followed user`, r.rating AS `rating`

Screenshots:



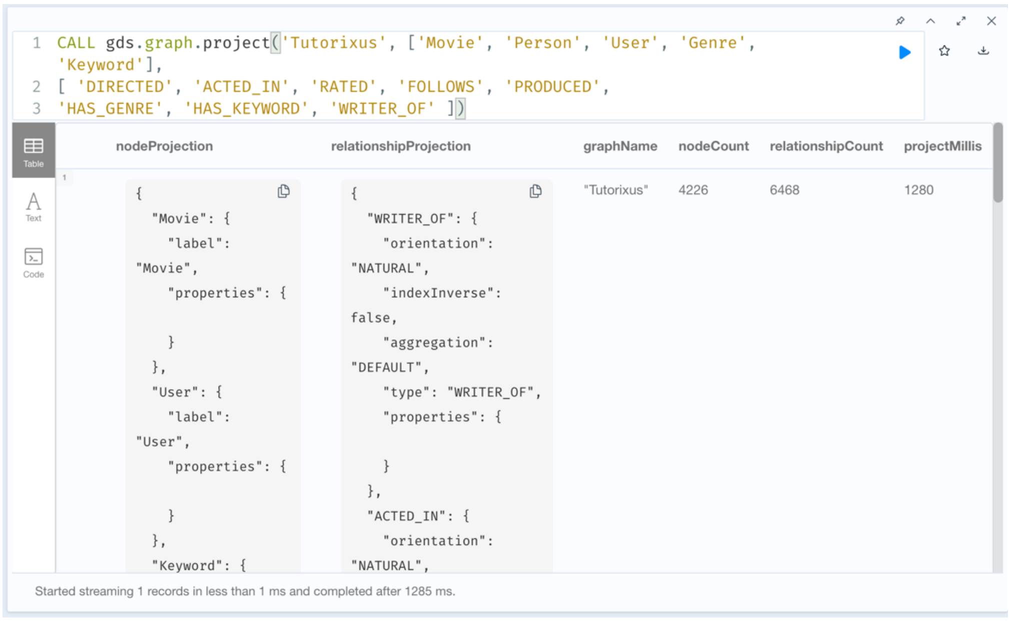
```
MATCH (aaron:User {username:'Aaron'})-[:FOLLOWS]→(u:User)-[r:RATED]→ (m:Movie)
RETURN m.title AS `MOVIES`, u.username AS `Followed user`, r.rating AS `rating`
```

```
╒═════════════════╤═════════════╤══════╕
│MOVIES           │Followed user│rating│
╞═════════════════╪═════════════╪══════╡
│"12 Angry Men"   │"James"      │3     │
├─────────────────┼─────────────┼──────┤
│"Inception"      │"Teng"       │4     │
├─────────────────┼─────────────┼──────┤
│"The Matrix"     │"Teng"       │5     │
├─────────────────┼─────────────┼──────┤
│"3 Days to Kill" │"Teng"       │4     │
├─────────────────┼─────────────┼──────┤
│"The Dark Knight"│"Teng"       │4     │
├─────────────────┼─────────────┼──────┤
│"The Godfather"  │"Teng"       │5     │
└─────────────────┴─────────────┴──────┘
```

Question 2
(a) The concept of betweenness centrality in the Tutorixus database is different from other contexts such as social networks. "Betweenness centrality measures the number of shortest paths that pass through a node". In social networks, nodes with high betweenness centrality scores imply a more bridging role between two communities. However, for the Tutorixus database, nodes with high betweenness centrality scores tend to imply commonality between clusters (Communities), e.g. Genre and keyword nodes tend to have high centrality scores, indicating common genres and keywords among different movies.

(b) This query creates a native projection that projects the graph from the Neo4j database to the GDS graph catalog.
The name of the graph: Tutorixus
The node being projected: ['Movie', 'Person', 'User', 'Genre', 'Keyword']
The Relationship being projected:
['DIRECTED', 'ACTED IN', 'RATED', 'FOLLOWS', 'PRODUCED', 'HAS GENRE', 'HAS KEYWORD', 'WRITER OF']

(c) Nodes Count:4226
Relationship Count: 6468



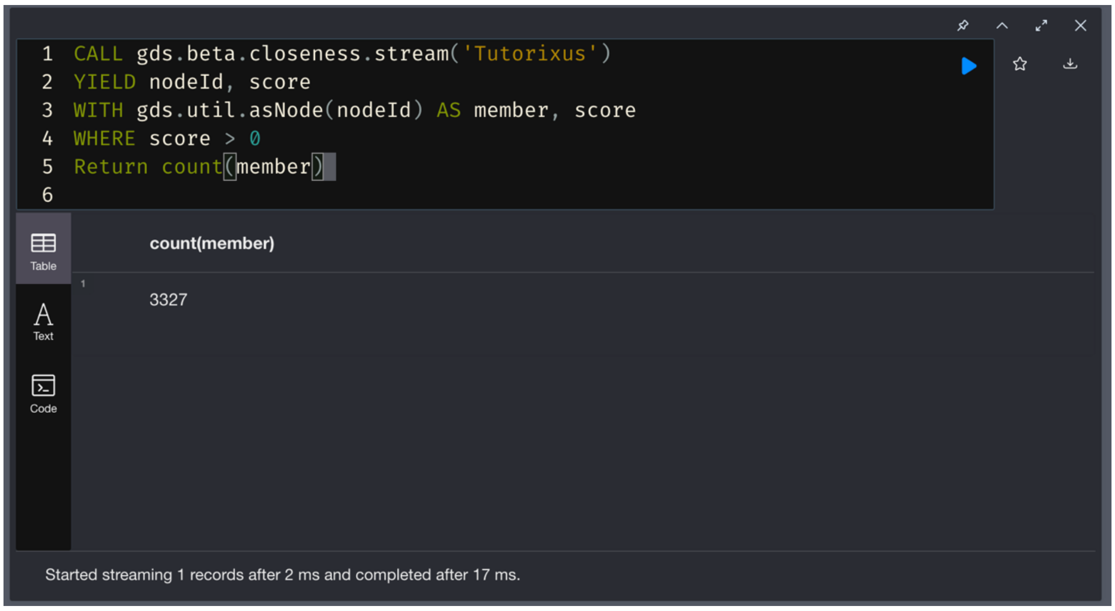
```
CALL gds.graph.project('Tutorixus', ['Movie', 'Person', 'User', 'Genre',
'Keyword'],
[ 'DIRECTED', 'ACTED_IN', 'RATED', 'FOLLOWS', 'PRODUCED',
'HAS_GENRE', 'HAS_KEYWORD', 'WRITER_OF' ])
```

nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	projectMillis
{ "Movie": { "label": "Movie", "properties": { } }, "User": { "label": "User", "properties": { } }, "Keyword": {	{ "WRITER_OF": { "orientation": "NATURAL", "indexInverse": false, "aggregation": "DEFAULT", "type": "WRITER_OF", "properties": { } }, "ACTED_IN": { "orientation": "NATURAL",	"Tutorixus"	4226	6468	1280

Started streaming 1 records in less than 1 ms and completed after 1285 ms.

(d) 3327
CALL gds.beta.closeness.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE score > 0
Return count(member)

Screenshots:



```
CALL gds.beta.closeness.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE score > 0
Return count(member)
```

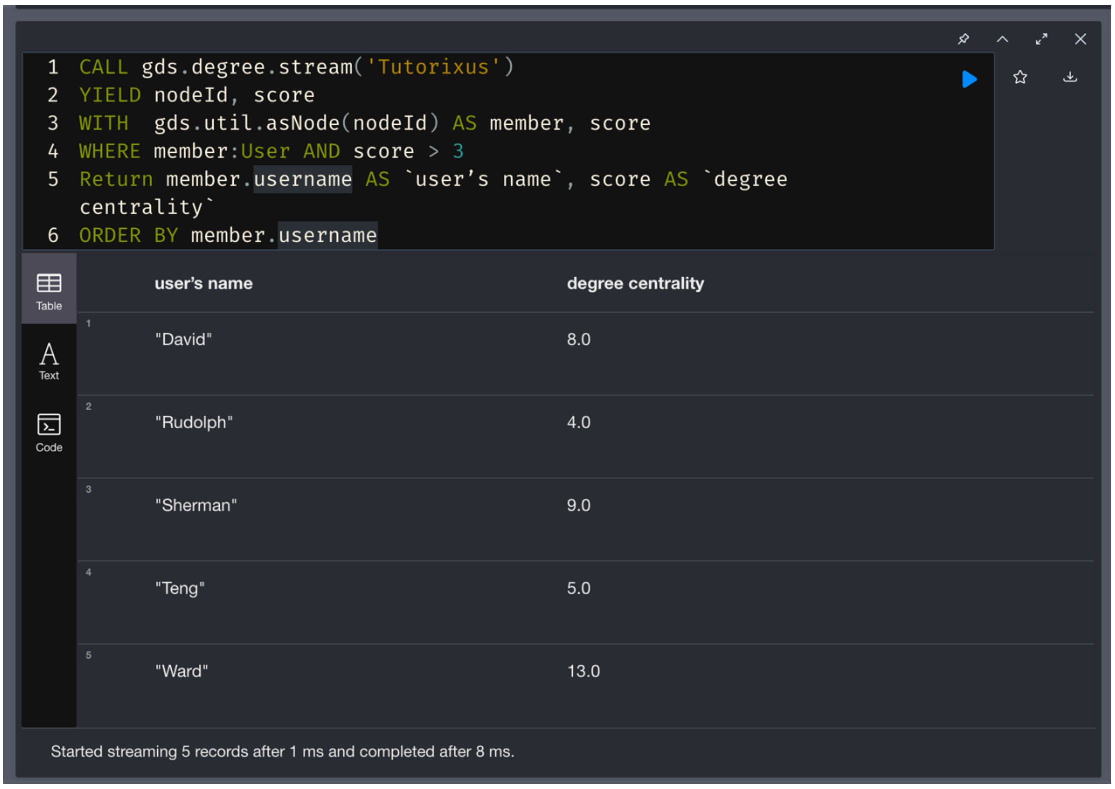
count(member)
3327

Started streaming 1 records after 2 ms and completed after 17 ms.

(e) CALL gds.degree.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE member:User AND score > 3

Return member.username AS `user’s name`, score AS `degree centrality`
ORDER BY member.username

Screenshots:



```
CALL gds.degree.stream('Tutorixus')
YIELD nodeId, score
WITH  gds.util.asNode(nodeId) AS member, score
WHERE member:User AND score > 3
Return member.username AS `user’s name`, score AS `degree
centrality`
ORDER BY member.username
```

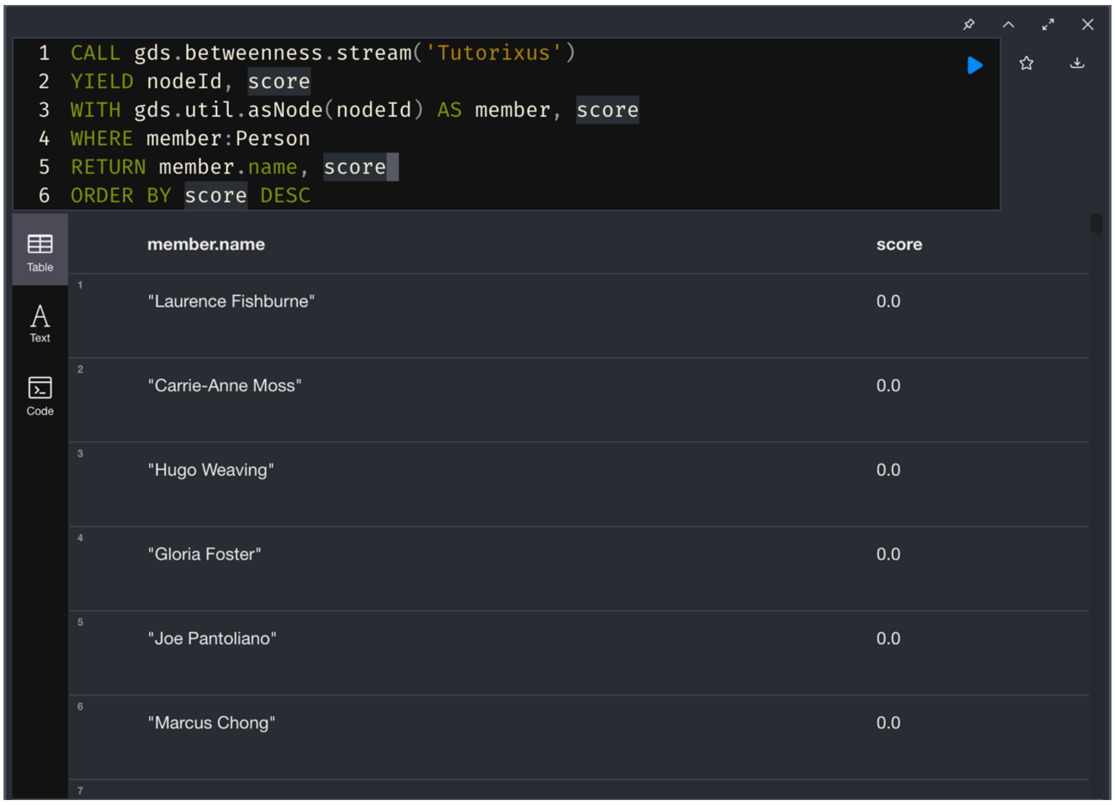
	user’s name	degree centrality
1	"David"	8.0
2	"Rudolph"	4.0
3	"Sherman"	9.0
4	"Teng"	5.0
5	"Ward"	13.0

Started streaming 5 records after 1 ms and completed after 8 ms.

Results Meaning: For the users node's degree centrality, a higher degree centrality means that this user has rated more movies.

(f) CALL gds.betweenness.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE member:Person
RETURN member.name, score
ORDER BY score DESC

Screenshot:



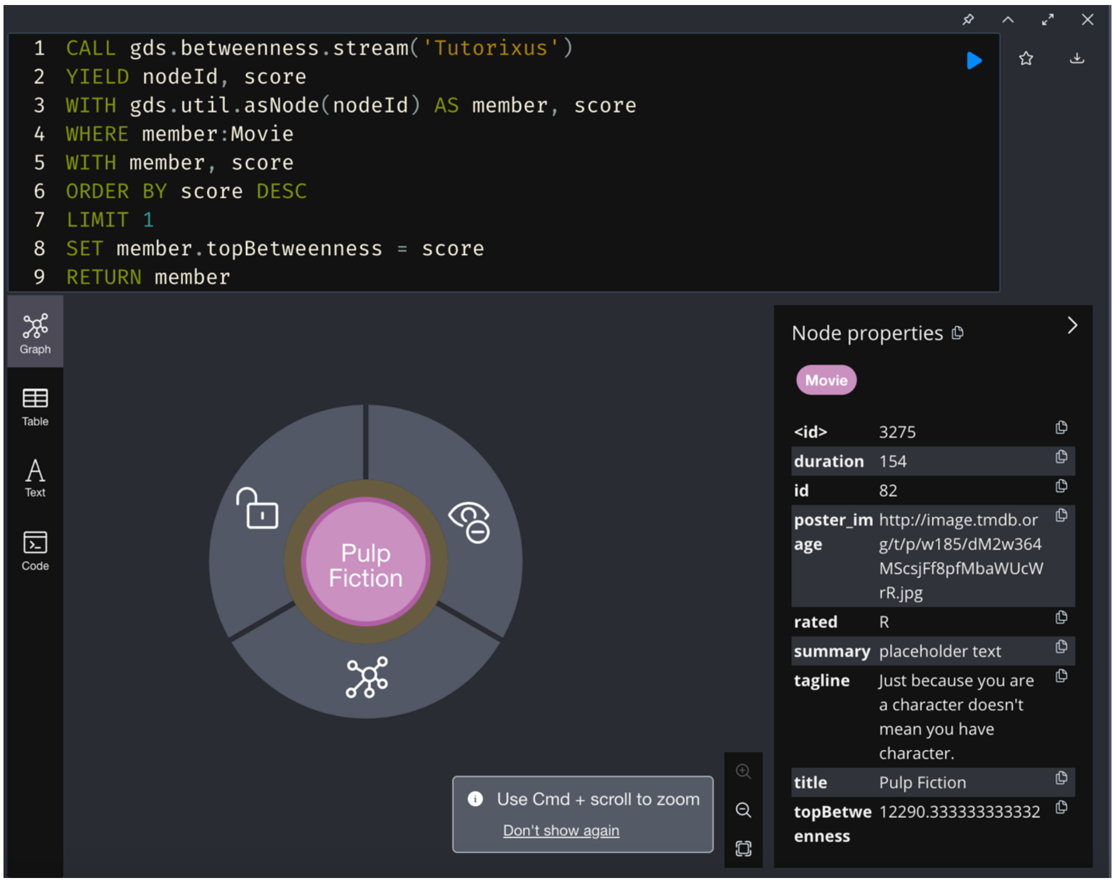
```
CALL gds.betweenness.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE member:Person
RETURN member.name, score
ORDER BY score DESC
```

	member.name	score
1	"Laurence Fishburne"	0.0
2	"Carrie-Anne Moss"	0.0
3	"Hugo Weaving"	0.0
4	"Gloria Foster"	0.0
5	"Joe Pantoliano"	0.0
6	"Marcus Chong"	0.0
7		

Results Meaning: From the results, the betweenness centrality of all persons is 0, which means that no shortest paths go through the Person node. This is easy to understand because Person nodes are used as the start of unidirectional paths, so no paths will "pass through" them.

(g) CALL gds.betweenness.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE member:Movie
WITH member, score
ORDER BY score DESC
LIMIT 1
SET member.topBetweenness = score
RETURN member

Screenshots:



```
CALL gds.betweenness.stream('Tutorixus')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS member, score
WHERE member:Movie
WITH member, score
ORDER BY score DESC
LIMIT 1
SET member.topBetweenness = score
RETURN member
```

Node properties

Movie

<id>	3275
duration	154
id	82
poster_image	http://image.tmdb.org/t/p/w185/dM2w364MScsjFf8pfMbaWUcWrR.jpg
rated	R
summary	placeholder text
tagline	Just because you are a character doesn't mean you have character.
title	Pulp Fiction
topBetweenness	12290.333333333332