Movie recommendations using neo4j

Vikrham R A(17PD39)
Aakaash Babu P(17PD01)

Objective:

Design a movie recommendation system using neo4j database.

Dataset:

Movielens dataset containing the following:

- **Users data** (contains users along with the given ratings).
- **Movies data** (contains details about the movies).
- **Genres data** (collection of genres are available here).

The above datasets are represented as nodes of the graph database.

The above nodes can be visualised as a graph by defining relationship between them.

Relationships:

- Users_movies relation.
- Movies_genres relation.
- Users_genres relation.
- Movies_similarity (using cosine similarity).

Customize data for defining relationship:

We create 3 dfs (mov_tag_df, mov_genres_df, mov_rating_df) and calculate 3 cosine similarities for each of them.

The aggregation of all 3 cos_similarities is calculated as movies_similarity = (mov_tag_df*0.5+mov_genres_df*0.25+mov_rating_df*0.25). The calculated cos_similarity is shown.

	1	2	3	4	5	6	7	8	9	10	...	131241
movieId												
1	1.000000	0.829784	0.680520	0.640024	0.695793	0.561475	0.653804	0.766840	0.519063	0.631729	...	0.241819
2	0.829784	1.000000	0.593250	0.568572	0.589406	0.510859	0.568324	0.840497	0.569749	0.661595	...	0.150295
3	0.680520	0.593250	1.000000	0.833609	0.857064	0.570108	0.888614	0.610713	0.558890	0.587168	...	0.399088
4	0.640024	0.568572	0.833609	1.000000	0.767326	0.554754	0.863996	0.623618	0.540909	0.549538	...	0.368420
5	0.695793	0.589406	0.857064	0.767326	1.000000	0.511446	0.817490	0.584793	0.538056	0.548123	...	0.324114

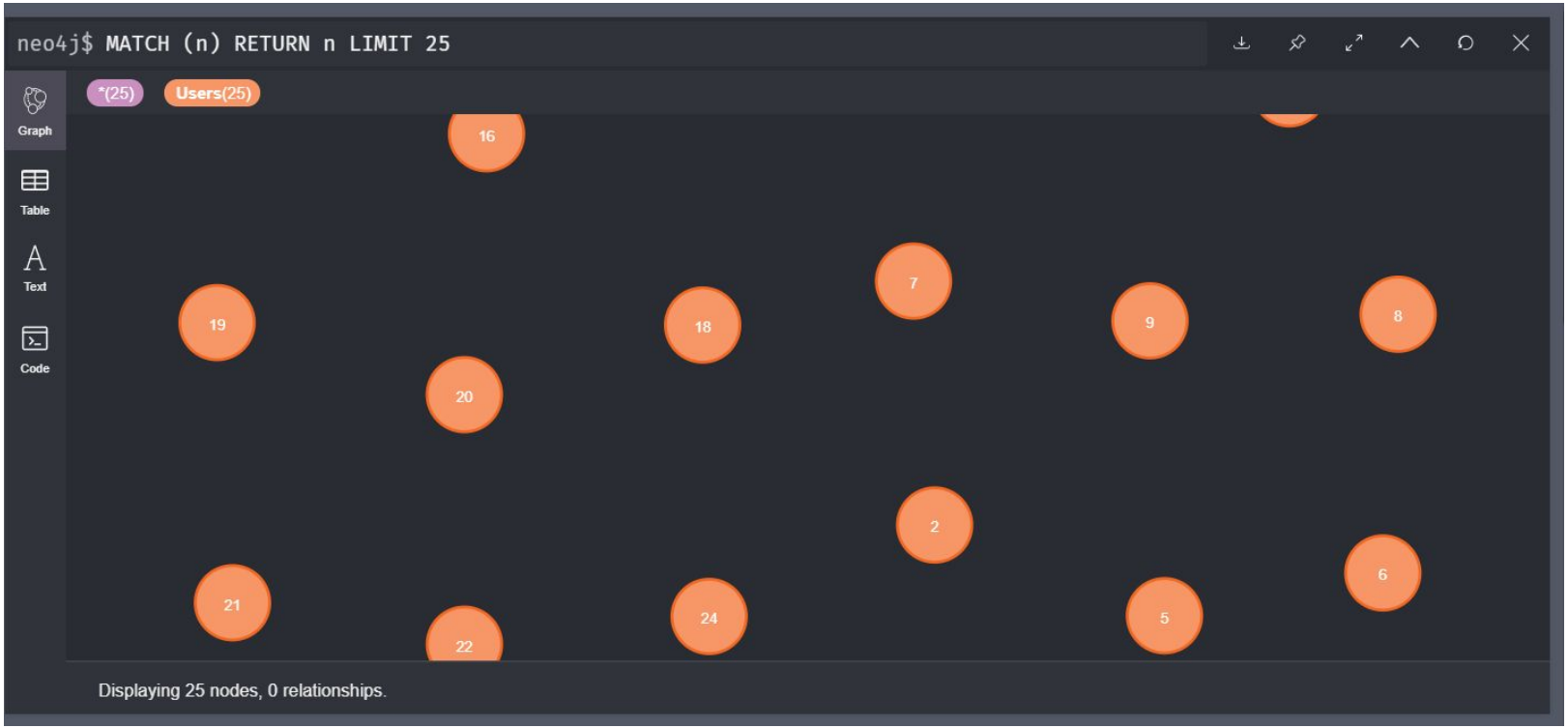
5 rows × 27278 columns

The nodes' data and relationships' data are exported as .csv files and loaded onto the neo4j database.

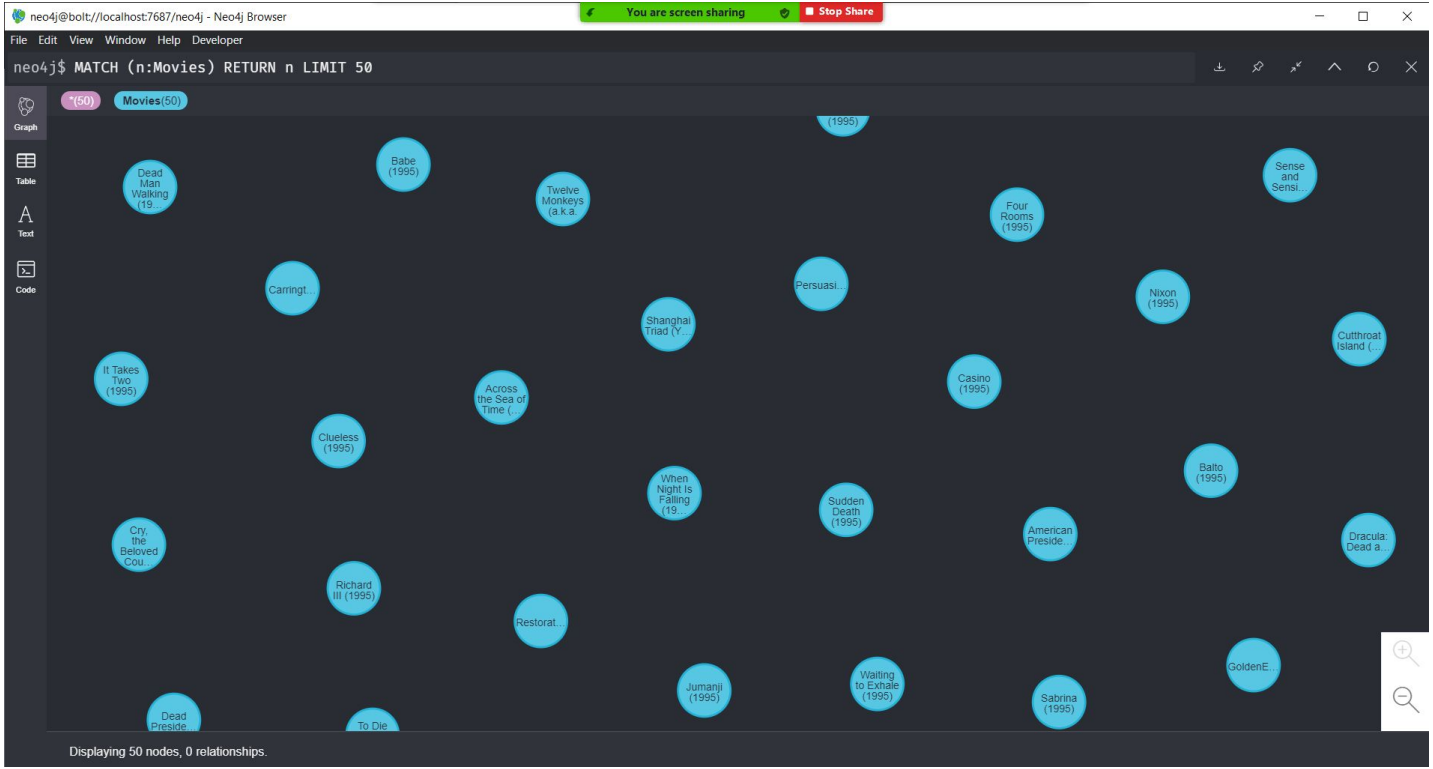
Importing nodes' data to neo4j:

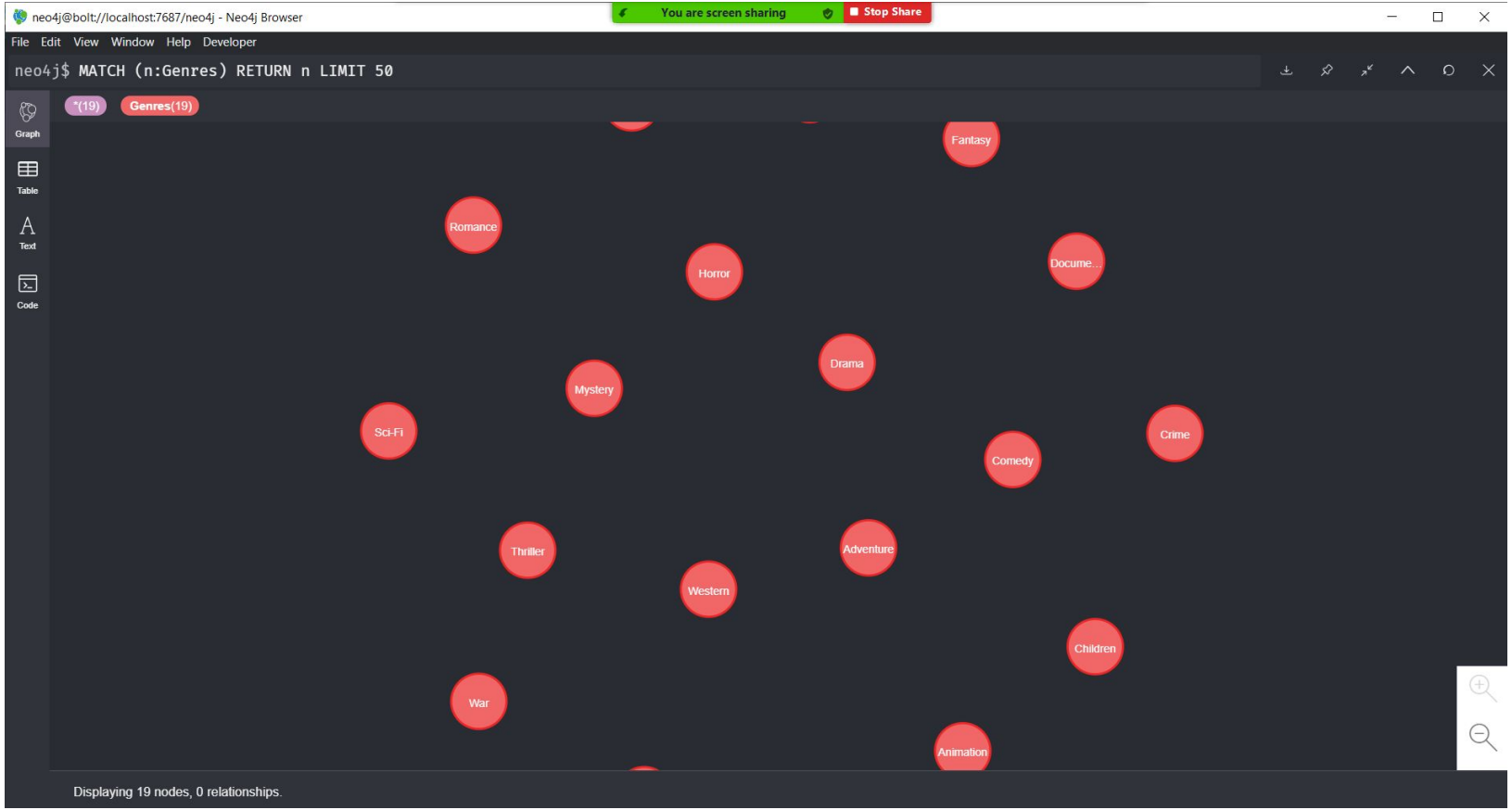
```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///users.csv" AS row
FIELDTERMINATOR '|'
CREATE (:Users {userId: row.userId});
```

Executing the above cypher in neo4j, the nodes will be loaded to define relationships.


neo4j$ MATCH (n) RETURN n LIMIT 25
Graph
Table
Text
Code
*(25)
Users(25)
16
19
18
7
9
8
20
2
6
21
22
24
5
Displaying 25 nodes, 0 relationships.

Executing the same cypher queries for creating the remaining nodes, we obtain the following.

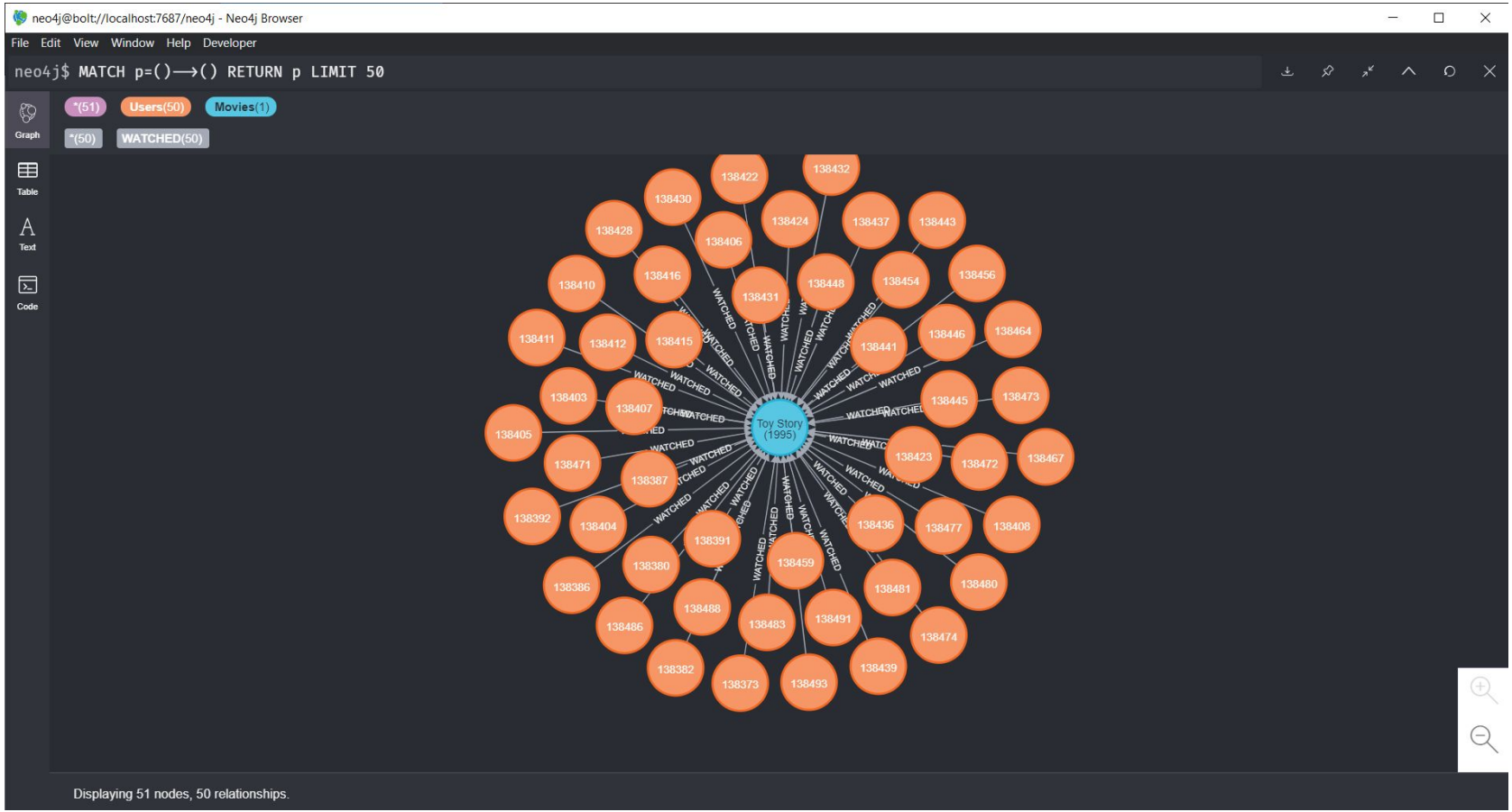



neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
You are screen sharing
Stop Share
File Edit View Window Help Developer
neo4j$ MATCH (n:Genres) RETURN n LIMIT 50
Graph
Table
Text
Code
*(19)
Genres(19)
Fantasy
Romance
Horror
Docume...
Drama
Mystery
Sci-Fi
Crime
Comedy
Thriller
Adventure
Western
Children
War
Animation
Displaying 19 nodes, 0 relationships.

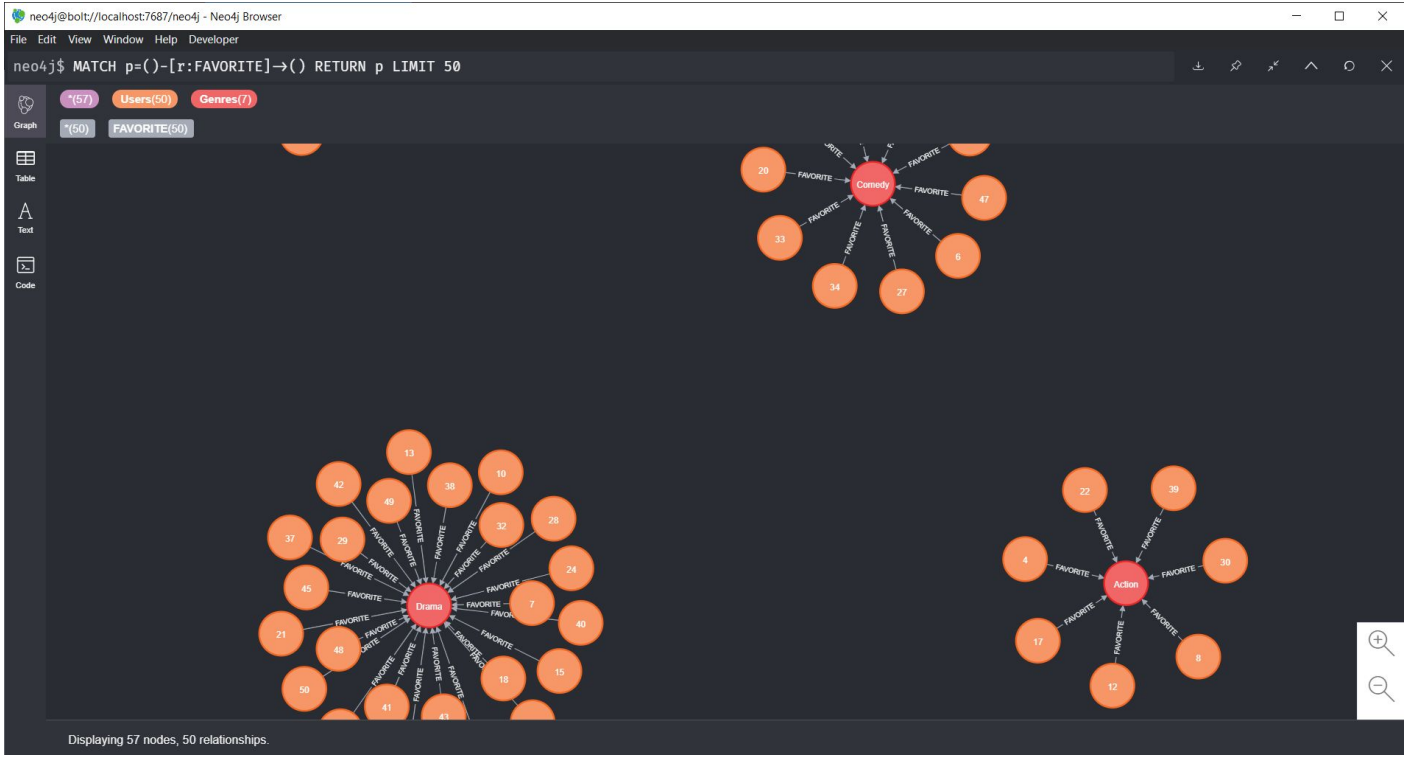
Importing relationships' data to neo4j:

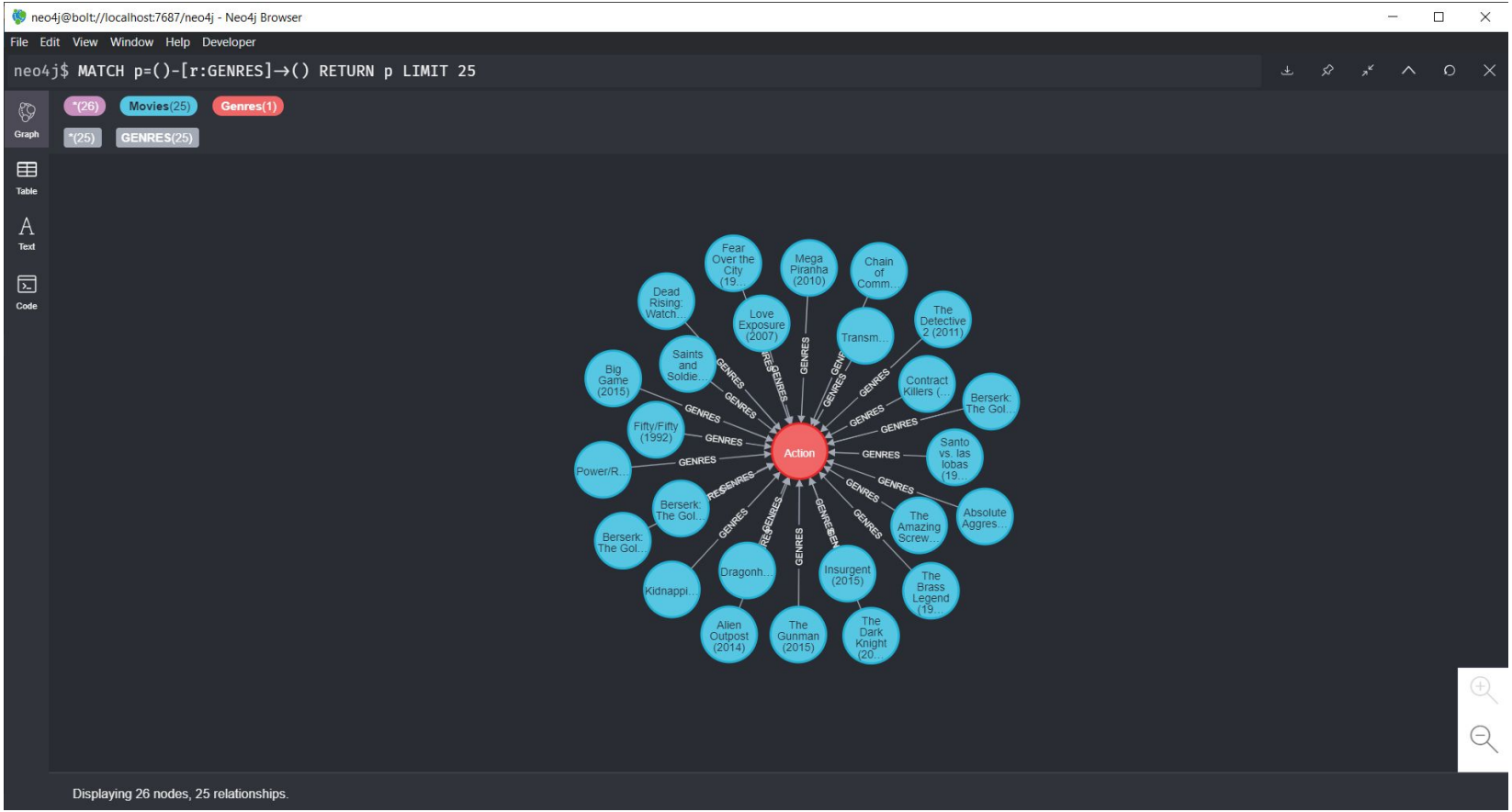
```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///users_movies.csv" AS row
FIELDTERMINATOR '|'
MATCH (user:Users {userId: row.userId})
MATCH (movie:Movies {movieId: row.movieId})
MERGE (user)-[:WATCHED {rating: row.rating}]->(movie);
```

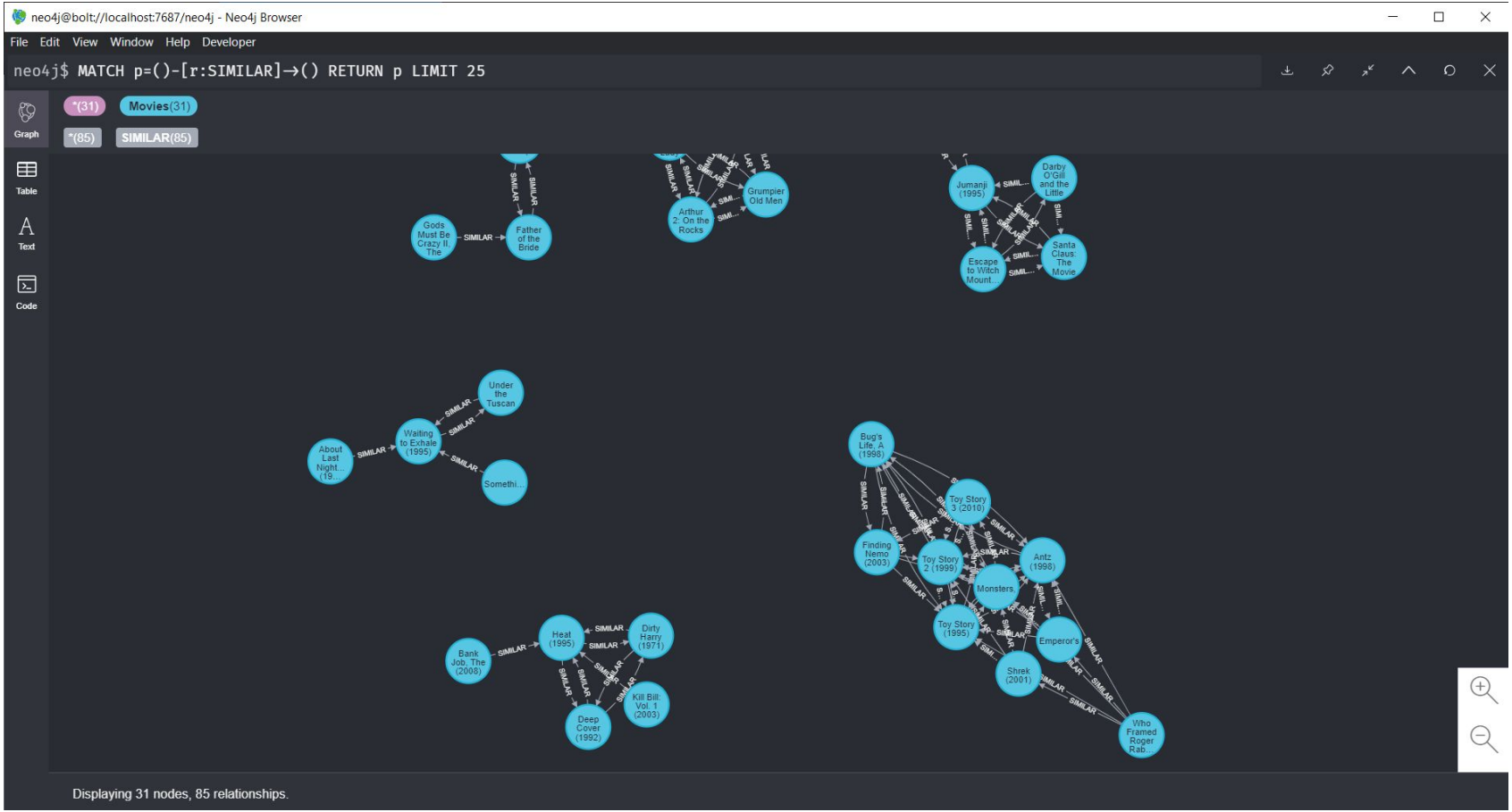
Executing the above cypher in neo4j, the relationship will be built between users' nodes and movies' nodes.


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ MATCH p=()⟶() RETURN p LIMIT 50
Graph
Table
Text
Code
*(51)
Users(50)
Movies(1)
*(50)
WATCHED(50)
Toy Story (1995)
138422
138432
138430
138424
138437
138443
138428
138406
138416
138448
138454
138456
138410
138431
138411
138412
138415
138441
138446
138464
138403
138407
138445
138473
138405
138423
138472
138467
138471
138387
138392
138404
138436
138477
138408
138391
138459
138380
138481
138480
138386
138488
138483
138491
138474
138486
138382
138373
138493
138439
Displaying 51 nodes, 50 relationships.

Similarly, for other relationships (users_genres, movies_genres and movies similarity)




neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ MATCH p=()-[r:GENRES]→() RETURN p LIMIT 25
Graph
Table
Text
Code
*(26)
Movies(25)
Genres(1)
*(25)
GENRES(25)
Action
Fear Over the City (19...
Mega Piranha (2010)
Chain of Comm...
Dead Rising: Watch...
Love Exposure (2007)
Transm...
The Detective 2 (2011)
Saints and Soldie...
Big Game (2015)
Contract Killers (...
Berserk: The Gol...
Fifty/Fifty (1992)
Santo vs. las lobas (19...
Power/R...
Absolute Aggres...
The Amazing Screw...
Berserk: The Gol...
Berserk: The Gol...
Dragonh...
Insurgent (2015)
The Brass Legend (19...
Kidnappi...
Alien Outpost (2014)
The Gunman (2015)
The Dark Knight (20...
GENRES
Displaying 26 nodes, 25 relationships.

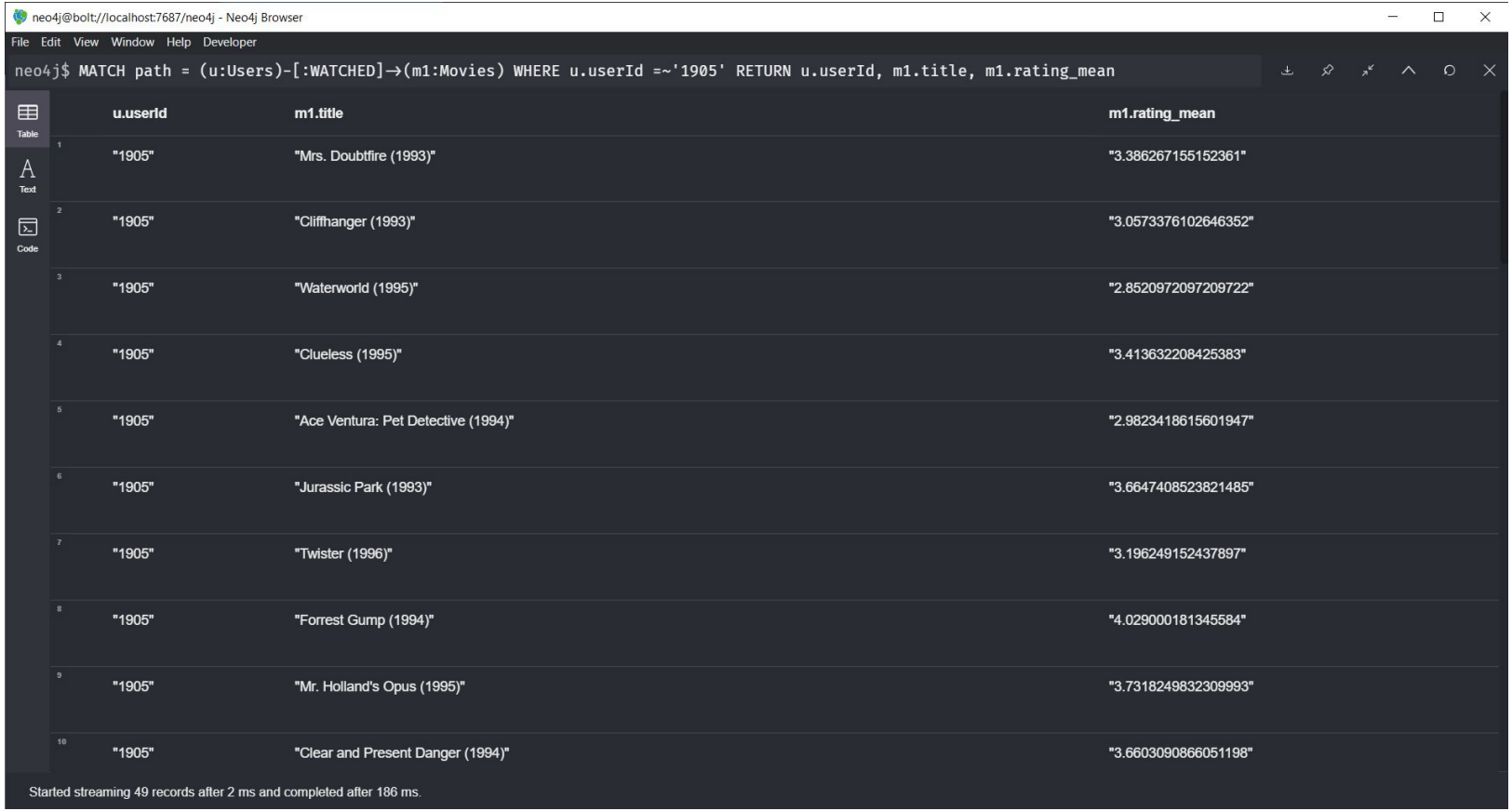

neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ MATCH p=()-[r:SIMILAR]→() RETURN p LIMIT 25
Graph
Table
Text
Code
*(31)
Movies(31)
*(85)
SIMILAR(85)
Gods Must Be Crazy II, The
Father of the Bride
Grumpier Old Men
Arthur 2: On the Rocks
Jumanji (1995)
Darby O'Gill and the Little
Escape to Witch Mount...
Santa Claus: The Movie
About Last Night... (19...
Waiting to Exhale (1995)
Under the Tuscan
Somethi...
Bug's Life, A (1998)
Toy Story 3 (2010)
Finding Nemo (2003)
Toy Story 2 (1999)
Monsters
Ants (1998)
Toy Story (1995)
Emperor's
Shrek (2001)
Who Framed Roger Rab...
Bank Job, The (2008)
Heat (1995)
Dirty Harry (1971)
Deep Cover (1992)
Kill Bill: Vol. 1 (2003)
Displaying 31 nodes, 85 relationships.

Query for recommendation:

Let’s review a user and check movies which are watched by him.

```
MATCH path = (u:Users)-[:WATCHED]->(m1:Movies)
WHERE u.userId =~'1905'
RETURN u.userId, m1.title, m1.rating_mean
```

The above cypher query shows movies watched by a user. The output is as follows.


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ MATCH path = (u:Users)-[:WATCHED]→(m1:Movies) WHERE u.userId =~'1905' RETURN u.userId, m1.title, m1.rating_mean
Table
Text
Code
u.userId
m1.title
m1.rating_mean
1 "1905" "Mrs. Doubtfire (1993)" "3.386267155152361"
2 "1905" "Cliffhanger (1993)" "3.0573376102646352"
3 "1905" "Waterworld (1995)" "2.8520972097209722"
4 "1905" "Clueless (1995)" "3.413632208425383"
5 "1905" "Ace Ventura: Pet Detective (1994)" "2.9823418615601947"
6 "1905" "Jurassic Park (1993)" "3.6647408523821485"
7 "1905" "Twister (1996)" "3.196249152437897"
8 "1905" "Forrest Gump (1994)" "4.029000181345584"
9 "1905" "Mr. Holland's Opus (1995)" "3.7318249832309993"
10 "1905" "Clear and Present Danger (1994)" "3.6603090866051198"
Started streaming 49 records after 2 ms and completed after 186 ms.

Filter movies which are already watched then sort them based on ratings and get 5 of them.

```
MATCH (u1:Users)-[:WATCHED]->(m3:Movies)
WHERE u1.userId =~'1905'
WITH [i in m3.movieId | i] as movies
MATCH path = (u:Users)-[:WATCHED]->(m1:Movies)-[s:SIMILAR]->
(m2:Movies),
(m2)-[:GENRES]->(g:Genres),
(u)-[:FAVORITE]->(g)
WHERE u.userId =~'1905' and not m2.movieId in movies
RETURN distinct u.userId as userId, g.genres as genres,
m2.title as title, m2.rating_mean as rating
ORDER BY m2.rating_mean descending
LIMIT 5
```


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ MATCH (u1:Users)-[:WATCHED]→(m3:Movies) WHERE u1.userId =~'1905' WITH [i in m3.movieId | i] as movies MATCH path = (u:Users)-…
Table
Text
Code
userId genres title rating
1 "1905" "Comedy" "Life Is Beautiful (La Vita è bella) (1997)" "4.175837188808107"
2 "1905" "Comedy" "Pulp Fiction (1994)" "4.174231169217055"
3 "1905" "Comedy" "Fargo (1996)" "4.112359031244223"
4 "1905" "Comedy" "Mister Roberts (1955)" "4.052339413164155"
5 "1905" "Comedy" "Snatch (2000)" "4.042146790032355"
Started streaming 5 records after 2 ms and completed after 34316 ms.

Thank you!

GitHub link: https://github.com/VikrhamRA/Movie-Recommendation