Graph Theory and its Applications

Assignment 4

Movie Recommender System

Abhishek Patil	PES1UG20CS010	'A' Section
Amogh S Vithapnavar	PES1UG20CS035	'A' Section

About the Dataset :

For building the recommender system, we use three main datasets

- **'movies.csv'**
- **'ratings.csv'**
- **'genome scores.csv'**

provided by GroupLens.

We further break and perform some operations on these 3 datasets to get a total of 7 datasets

- 3 of them are for creating nodes

a. Users
b. Movies
c. Genres

- 4 are for relationships

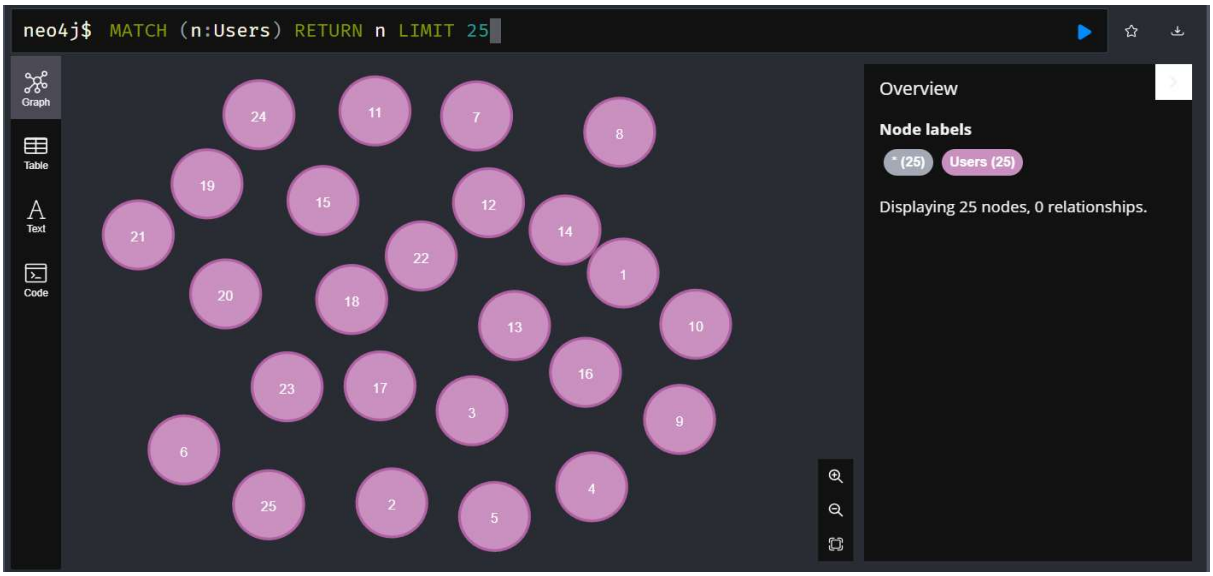
a. WATCHED
b. SIMILAR
c. GENRE
d. FAVORITE

Step 1. Import the Datasets into Neo4j Graph

A. Creating Nodes

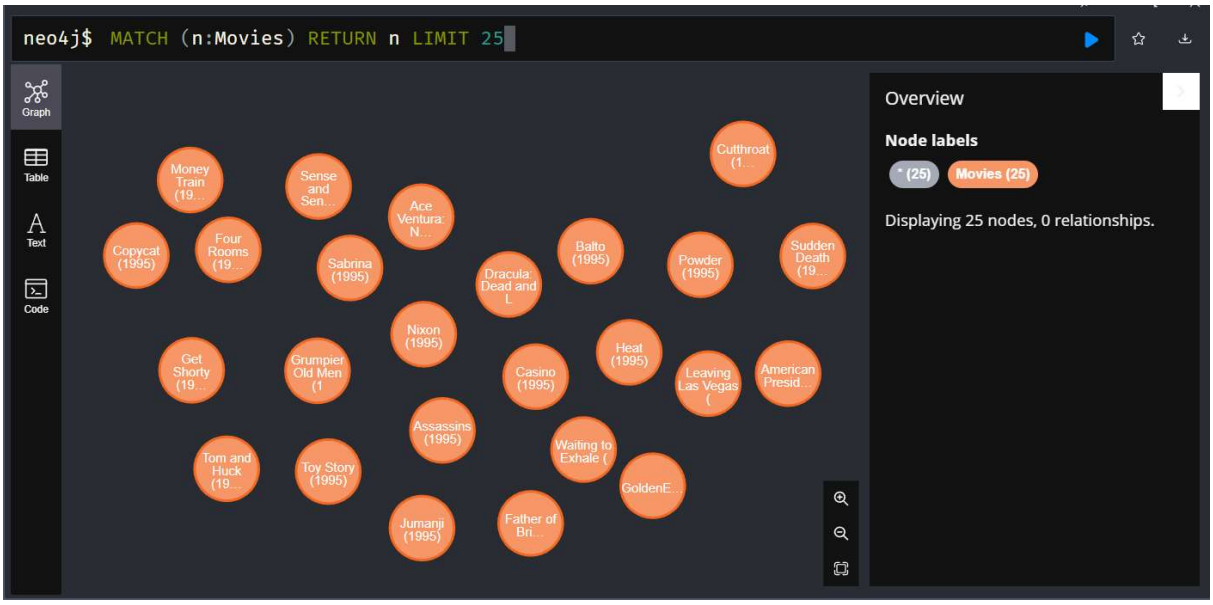
Users :

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



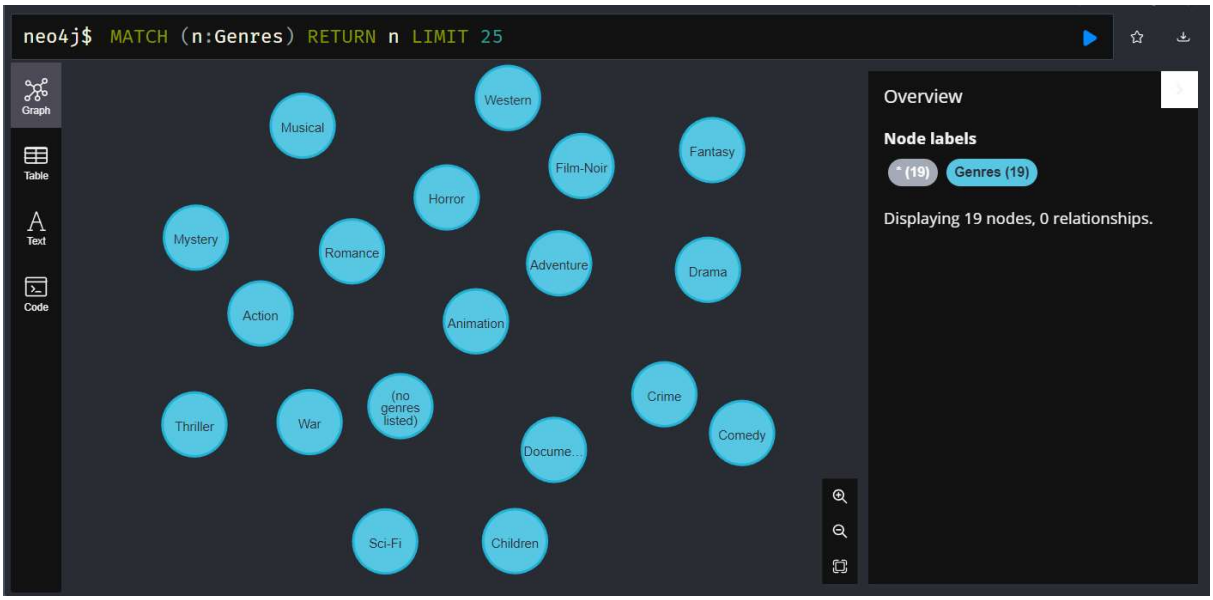
Movies:

```
LOAD CSV WITH HEADERS FROM "file:///movies.csv" AS row
FIELDTERMINATOR '|'
CREATE (:Movies {movieId: row.movieId, title: row.title, rating_mean:
row.rating_mean});
```



Genres:

```
LOAD CSV WITH HEADERS FROM "file:///genres.csv" AS row
FIELDTERMINATOR '|'
CREATE (:Genres {genres: row.genres});
```



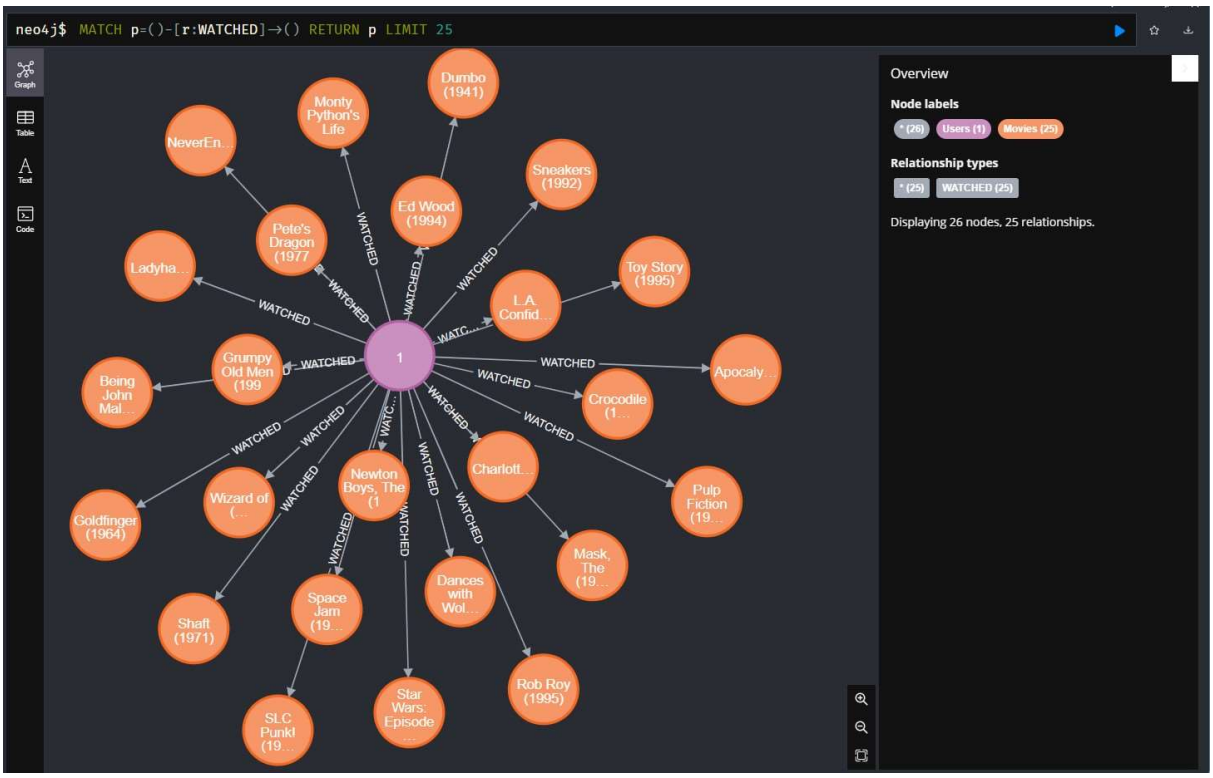
Create **INDEX** for nodes :

```
CREATE INDEX FOR (Users:userId) ON (Users.userId);
CREATE INDEX FOR (Movies:movieId) ON (Movies.movieId);
```

B. Relations

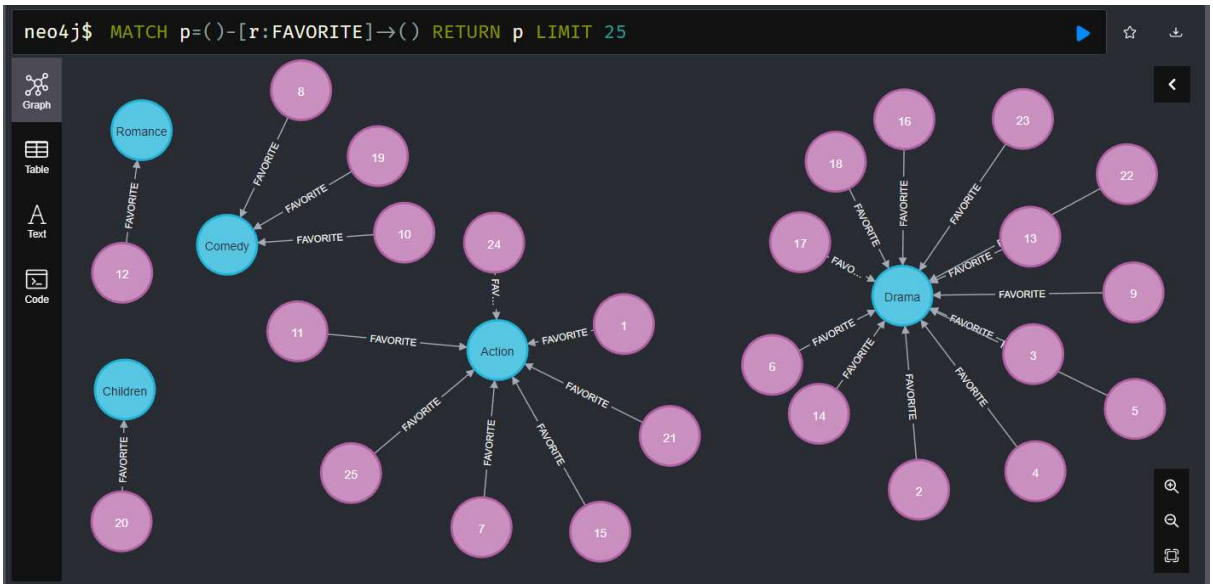
user-movie : **WATCHED**

```
CREATE INDEX FOR (Movies:movieId) ON (Movies.movieId);
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);
CREATE INDEX FOR (Movies:movieId) ON (Movies.movieId);
```



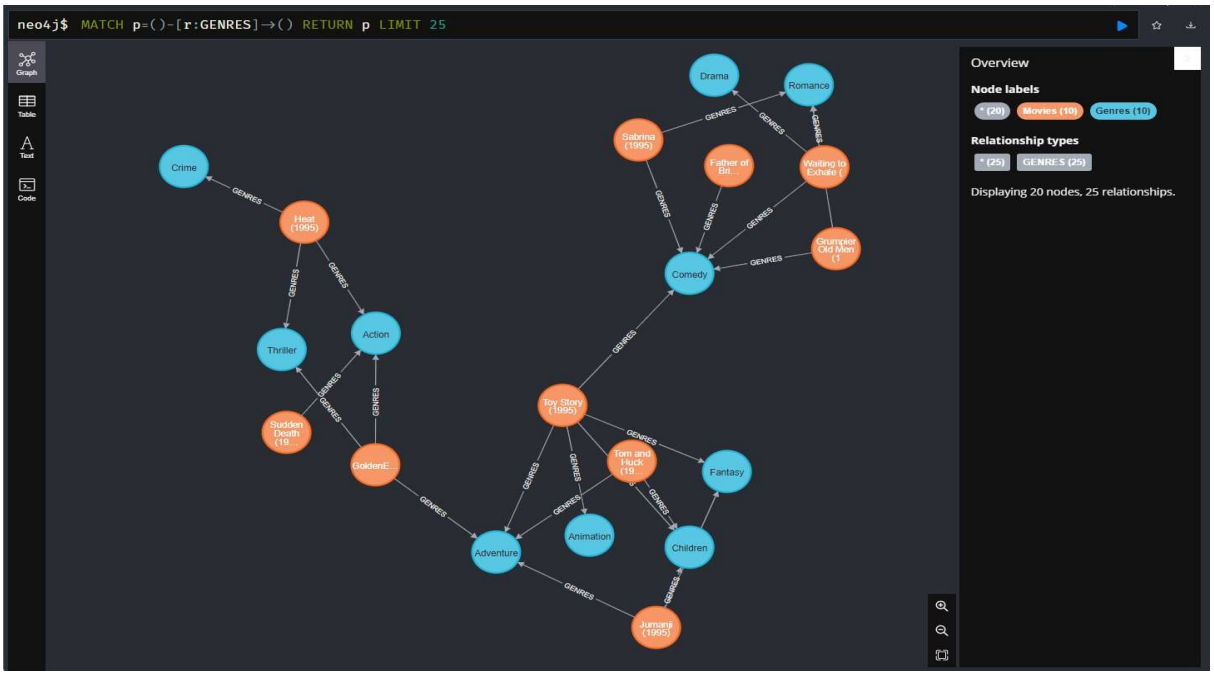
user-genre : **FAVORITE**

```
LOAD CSV WITH HEADERS FROM "file:///users_genres.csv" AS row
FIELDTERMINATOR '|'
MATCH (user:Users {userId: row.userId})
MATCH (genres:Genres {genres: row.genre})
MERGE (user)-[:FAVORITE]→(genres);
```



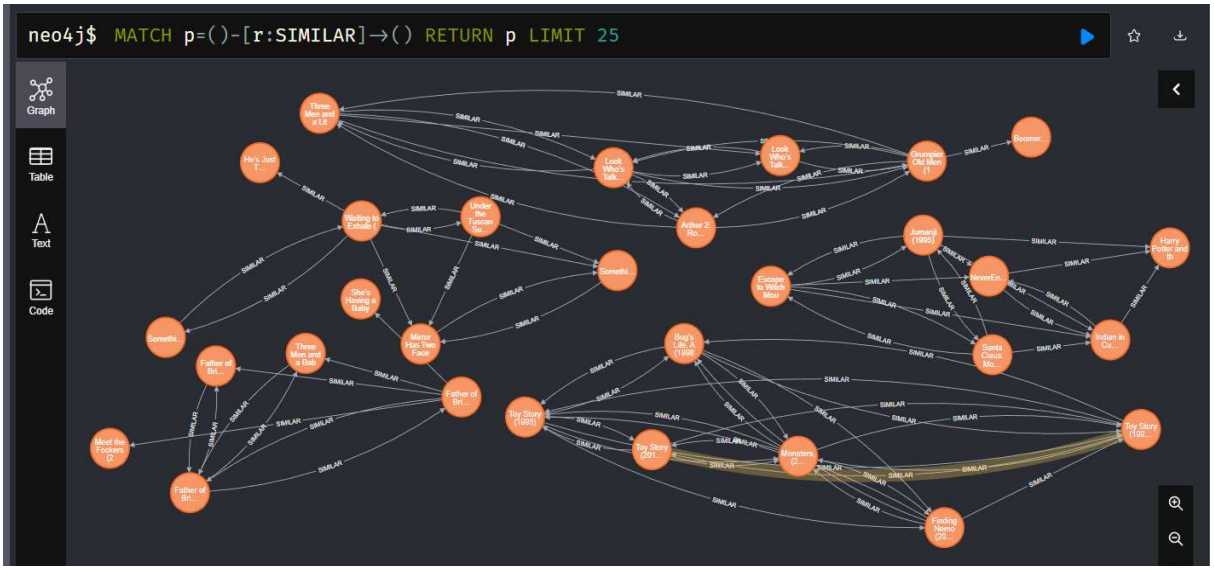
movie - genre : **GENRE**

```
CREATE INDEX FOR (Movies:movieId) ON (Movies.movieId);
LOAD CSV WITH HEADERS FROM "file:///movies_genres.csv" AS row
FIELDTERMINATOR '|'
MATCH (movie:Movies {movieId: row.movieId})
MATCH (genres:Genres {genres: row.genres})
MERGE (movie)-[:GENRES]→(genres);
```



movie-movie : **SIMILAR**

```
CREATE INDEX FOR (Movies:movieId) ON (Movies.movieId);
LOAD CSV WITH HEADERS FROM "file:///movies_similarity.csv" AS row
FIELDTERMINATOR '|'
MATCH (movie1:Movies {movieId: row.movieId})
MATCH (movie2:Movies {movieId: row.sim_moveId})
MERGE (movie1)-[:SIMILAR {relevance: row.relevance}]→(movie2);
```



Step 2. Basic Queries

Top 5 rated- movies watched by user **10** :

```
MATCH path = (u:Users)-[:WATCHED]→(m1:Movies)
WHERE u.userId =~'10'
RETURN u.userId as user, m1.title as movie, m1.rating_mean
as mean_rating order by mean_rating desc LIMIT 5
```

```
neo4j$ MATCH path = (u:Users)-[:WATCHED]→(m1:Movies) WHERE u.u…
```

	user	movie	mean_rating
1	"10"	"The Hundred-Foot Journey (2014)"	"4.5"
2	"10"	"Fight Club (1999)"	"4.272935779816514"
3	"10"	"Priceless (Hors de prix) (2006)"	"4.25"
4	"10"	"Casablanca (1942)"	"4.24"
5	"10"	"Dark Knight, The (2008)"	"4.238255033557047"

Favorite Genre of **first 5** users:

```
MATCH path = (u:Users)-[:FAVORITE]→(g:Genres)
RETURN u.userId as User, g.genres as Fav_Genre
order by toInteger(User) LIMIT 5
```

```
neo4j$ MATCH path = (u:Users)-[:FAVORITE]→(g:Genres) RETURN u.…
```

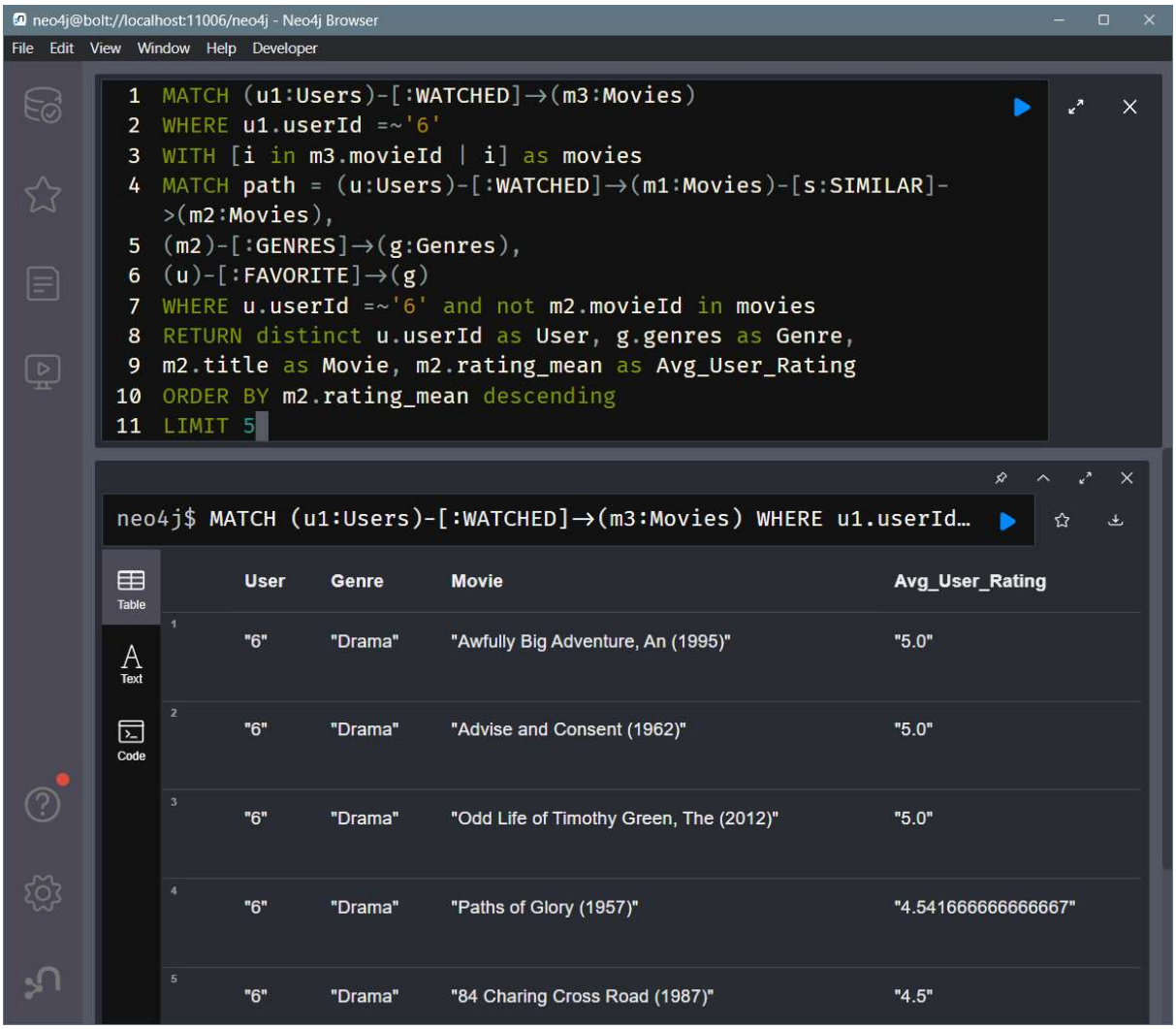
	User	Fav_Genre
1	"1"	"Action"
2	"2"	"Drama"
3	"3"	"Drama"
4	"4"	"Drama"
5	"5"	"Drama"

Step 3. Content Based Filtering:

Content-based filtering uses the category of the movies to recommend other movies which fall under the categories that the user likes, based on the ratings of their previous watched movies through movie ratings.

The following are the steps and the demonstration of Content based filtering to recommend movies to user **6**:

a. Get the list of movies **watched** by User 6 - **List A**
b. For List A, get the list of **similar** movies - **List B**
c. From List B, select the movies which fall under the **favorite** genre of the user
d. Order them by their **Avg ratings** given by the users and recommend top 5 among them.

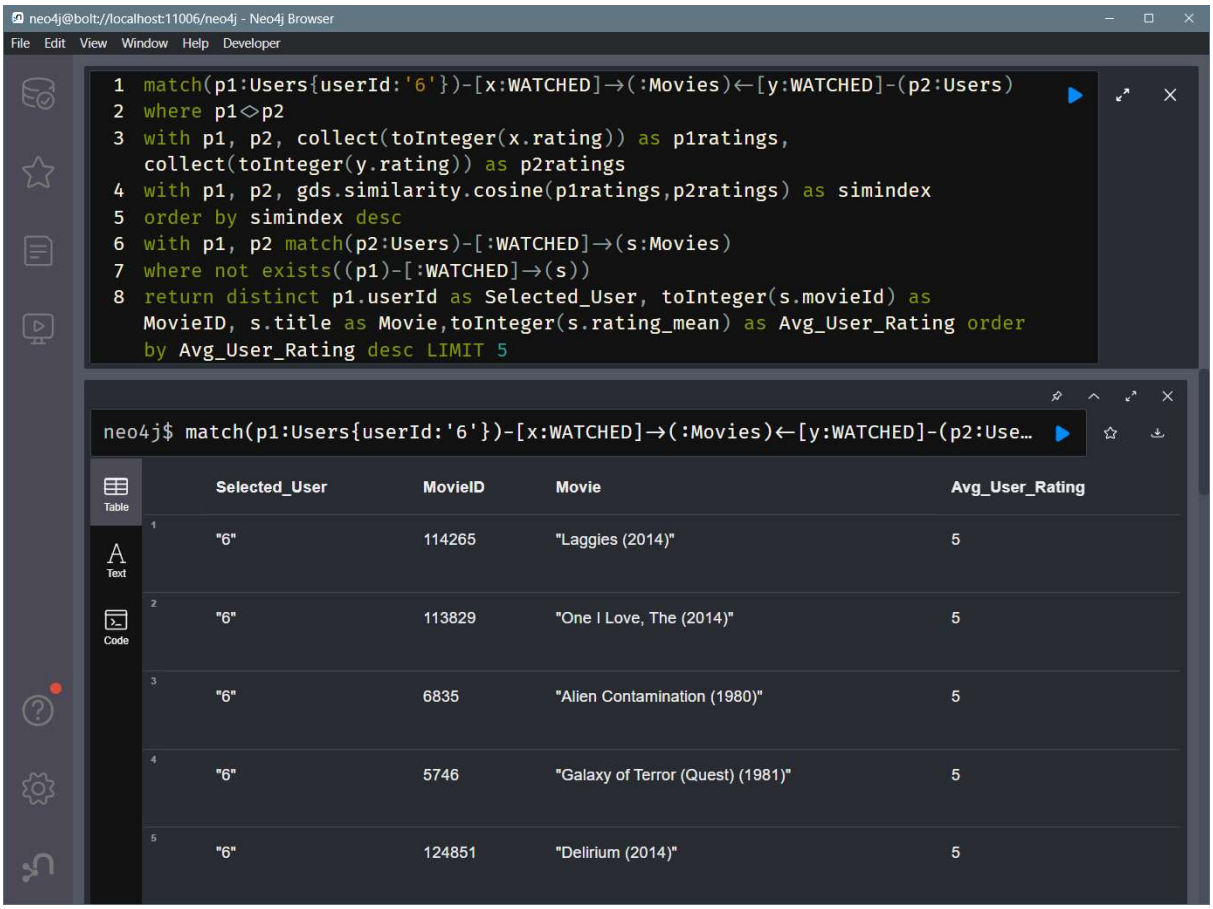


Step 4. Collaborative Filtering:

Collaborative Filtering uses **similarities** between users and movies simultaneously, to provide recommendations. This allows for serendipitous recommendations: i.e. collaborative filtering models can recommend a movie to user A based on the interests of a similar user B.

The following are the steps and the demonstration of Collaborative filtering to recommend movies to user **6**:

a. Get the list of users who watched the **same** movies as selected user
b. Find the **cosine similarity** between the rating of the selected user and the ratings of the other users for the movies they've watched and order them according to the **cosine similarity index**
c. Get the list of the movies these users have watched but **aren't watched** by the selected user and recommend the **top 5** according to the ratings.

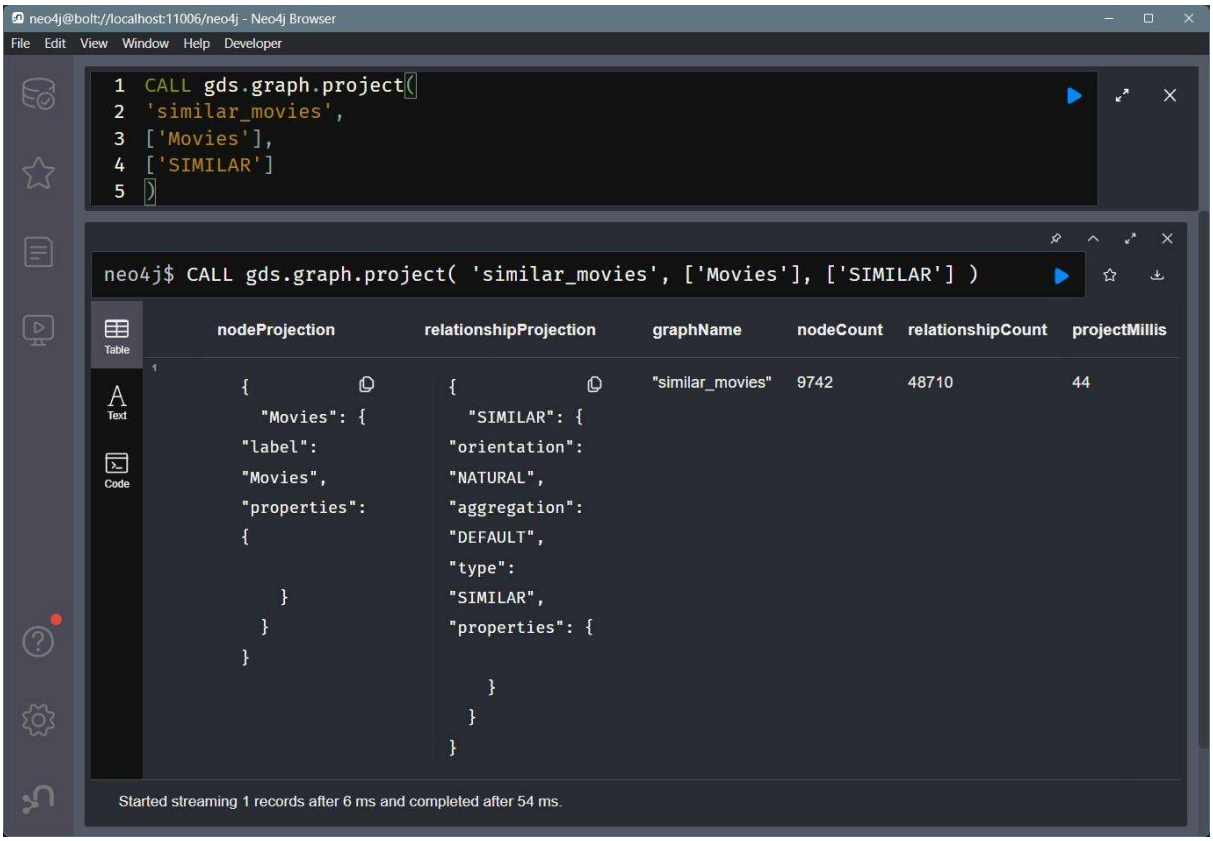


	Selected_User	MovieID	Movie	Avg_User_Rating
1	"6"	114265	"Laggies (2014)"	5
2	"6"	113829	"One I Love, The (2014)"	5
3	"6"	6835	"Alien Contamination (1980)"	5
4	"6"	5746	"Galaxy of Terror (Quest) (1981)"	5
5	"6"	124851	"Delirium (2014)"	5

Step 5. Betweenness Centrality:

Betweenness Centrality is a measure of centrality in a graph based on shortest paths. For each node, it is the **number of shortest paths** (between pairs of nodes) that pass through the node.

First create a **graph** of **similar** movies:



As seen in the above image, the nodeCount for the graph is **9742**.

We calculated the betweenness centrality measure for each node, then normalized the measures using the **normalization factor = (n-1)(n-2)**, here n = 9742 and fetched the **top 5** nodes with the highest betweenness centrality:

neo4j@bolt://localhost:11006/neo4j - Neo4j Browser

File Edit View Window Help Developer

```
CALL gds.betweenness.stream('similar_movies')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).movieId AS MovieID,
gds.util.asNode(nodeId).title AS Movie, score/((9742-1)*(9742-2)) as
betweenness_measure order by score desc LIMIT 5
```

```
neo4j$ CALL gds.betweenness.stream('similar_movies') YIELD nodeId, score RETURN g…
```

Table | Text | Code

	MovieID	Movie	betweenness_measure
1	"1821"	"Object of My Affection, The (1998)"	0.06160804423642283
2	"4722"	"All Over the Guy (2001)"	0.06148083703894664
3	"60408"	"Welcome to the Sticks (Bienvenue chez les Ch'tis) (2008)"	0.06127148976078798
4	"4381"	"Closet, The (Placard, Le) (2001)"	0.060856188128415756
5	"8033"	"How to Steal a Million (1966)"	0.057416716685083226

Started streaming 5 records after 1 ms and completed after 4788 ms.