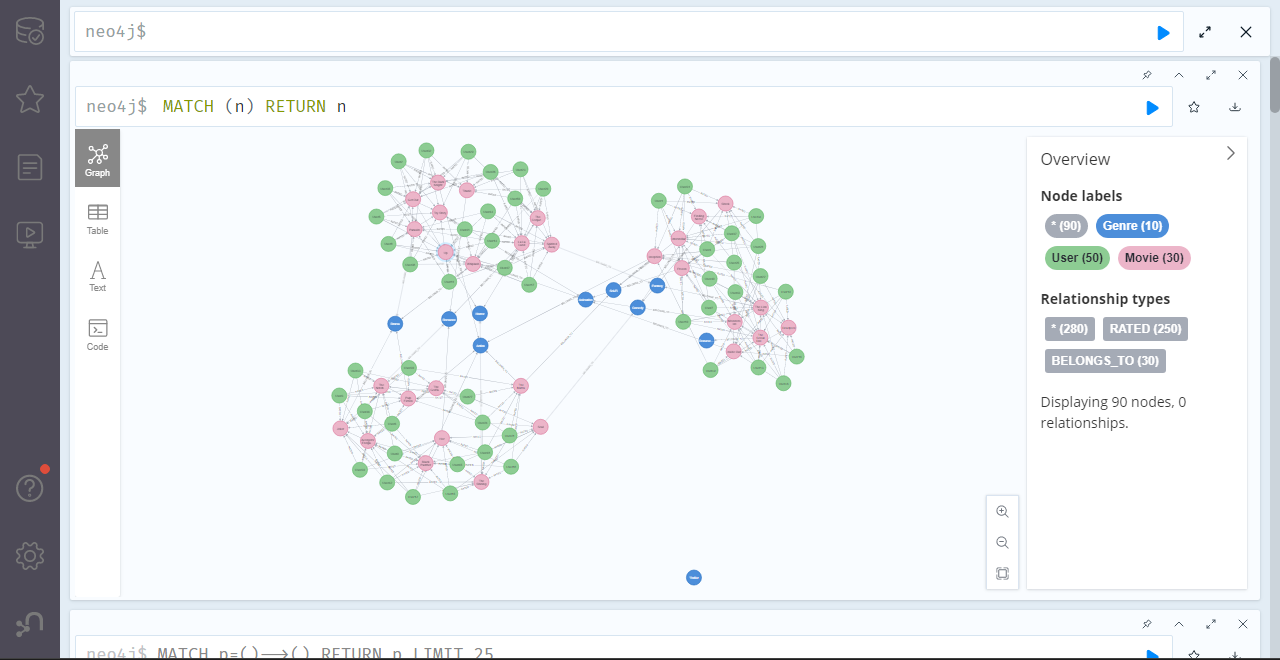
Kaziah Ofori

This graph database models the interactions between users, films, and genres in order to simulate a movie recommendation system. For analytics and query testing, this graph aims to replicate the composition and actions of an actual recommendation system. Using actual data from the MovieLens 100K dataset, which was made available by GroupLens Research, this graph database simulates a movie recommendation engine. To facilitate effective relationship inquiries, recommendation algorithms, and user behavior analysis, the data has been organized into a graph format. There are 100,000 user ratings for 1,682 films in the dataset. Every rating has a timestamp, user ID, movie ID, and rating (1–5). Genres and titles are included in movie metadata. In my graph I had three different node types, The user, which had a sample subset of 50 users and properties based on the users ID, age, gender, and occupation. The second ode type was movie, which had the properties movie id, title. and year and had a subset of 30-50 movies. The third node consisted of subsets derived from the genre flags in the dataset. Its only had one property called name, for example drama, comedy, romance. For the relationships, I first made a related relationship from user to movie which represents the user’s rating of the movie. Then, belongs to relationships type from movie to genre to show the genres each movie falls under.

Link to dataset: [Movie Lens](https://grouplens.org/datasets/movielens/100k/)

Match and Return



1. Find all users that rated “The Matrix”

MATCH (u:User)-[:RATED]->(m:Movie {title: "The Matrix"})

RETURN u.name;



1. List all movies rated higher than 4.

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

WHERE r.rating > 4

RETURN m.title, r.rating, u.name;



1. List all genres for the movie Interception.

MATCH (m:Movie {title: "Inception"})-[:BELONGS\_TO]->(g:Genre)

RETURN g.name;



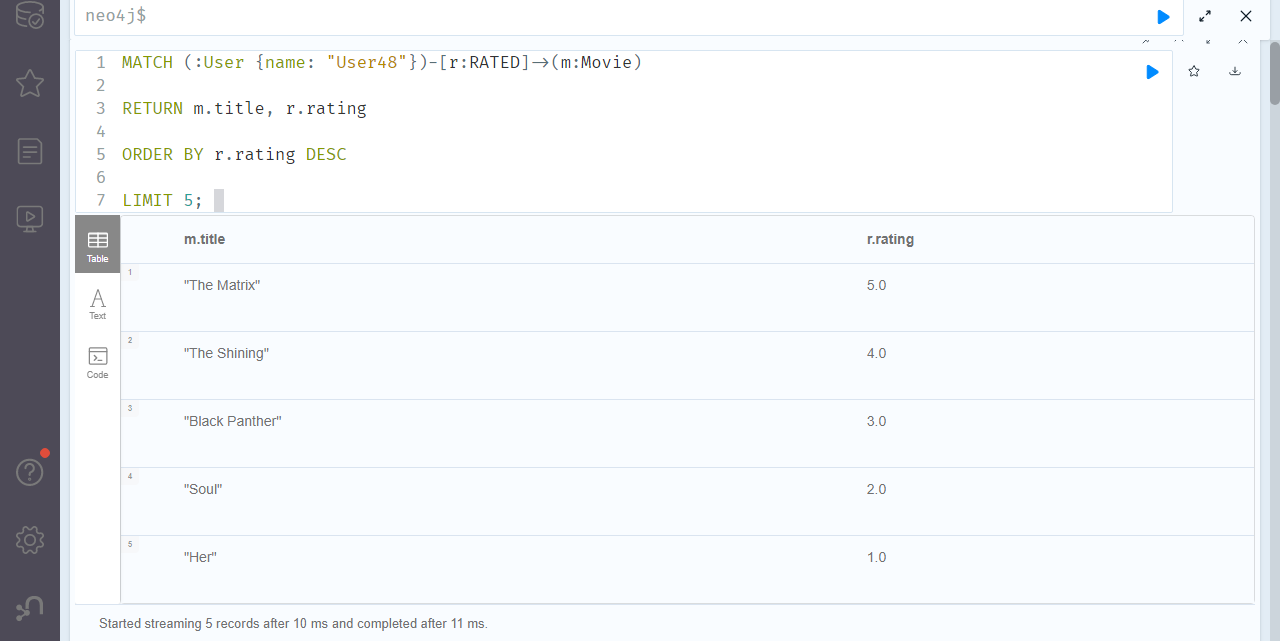
1. Find the top 5 movies rated by user 48.

MATCH (:User {name: "User48"})-[r:RATED]->(m:Movie)

RETURN m.title, r.rating

ORDER BY r.rating DESC

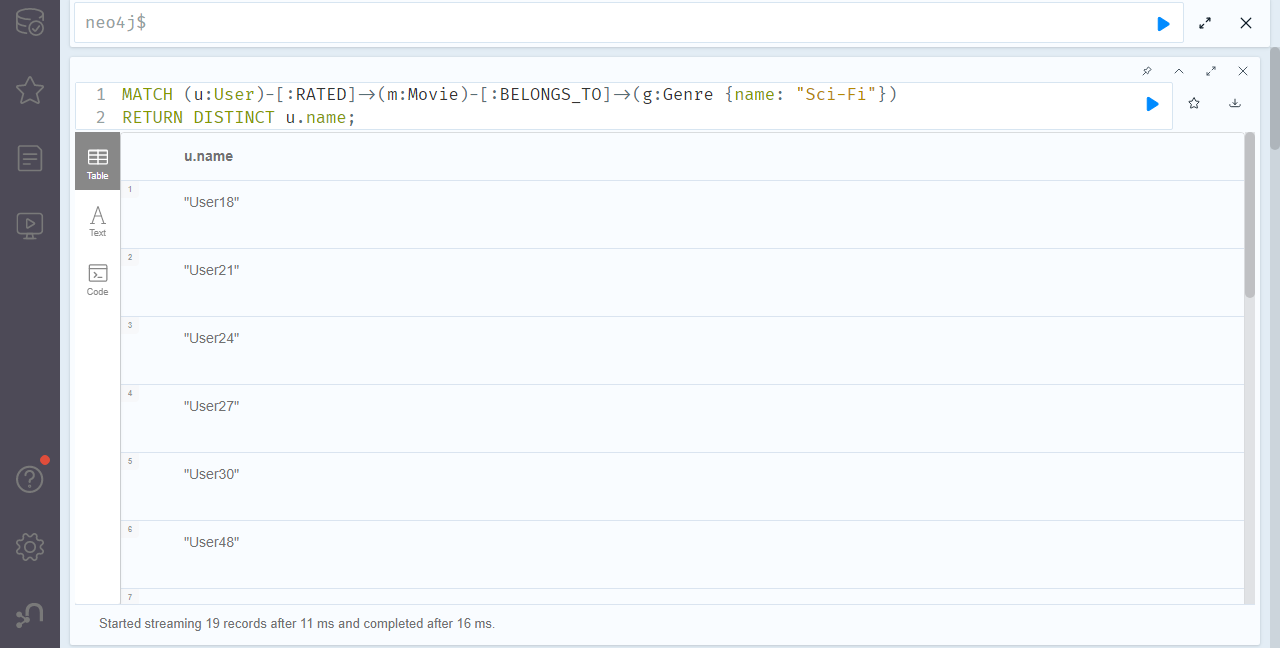
LIMIT 5;



1. Find people who like the genre Sci-Fi.

MATCH (u:User)-[:RATED]->(m:Movie)-[:BELONGS\_TO]->(g:Genre {name: "Sci-Fi"})

RETURN DISTINCT u.name;



1. Find the genres most associated with high ratings.

MATCH (u:User)-[r:RATED]->(m:Movie)-[:BELONGS\_TO]->(g:Genre)

WHERE r.rating > 4.0

RETURN g.name, COUNT(\*) AS freq

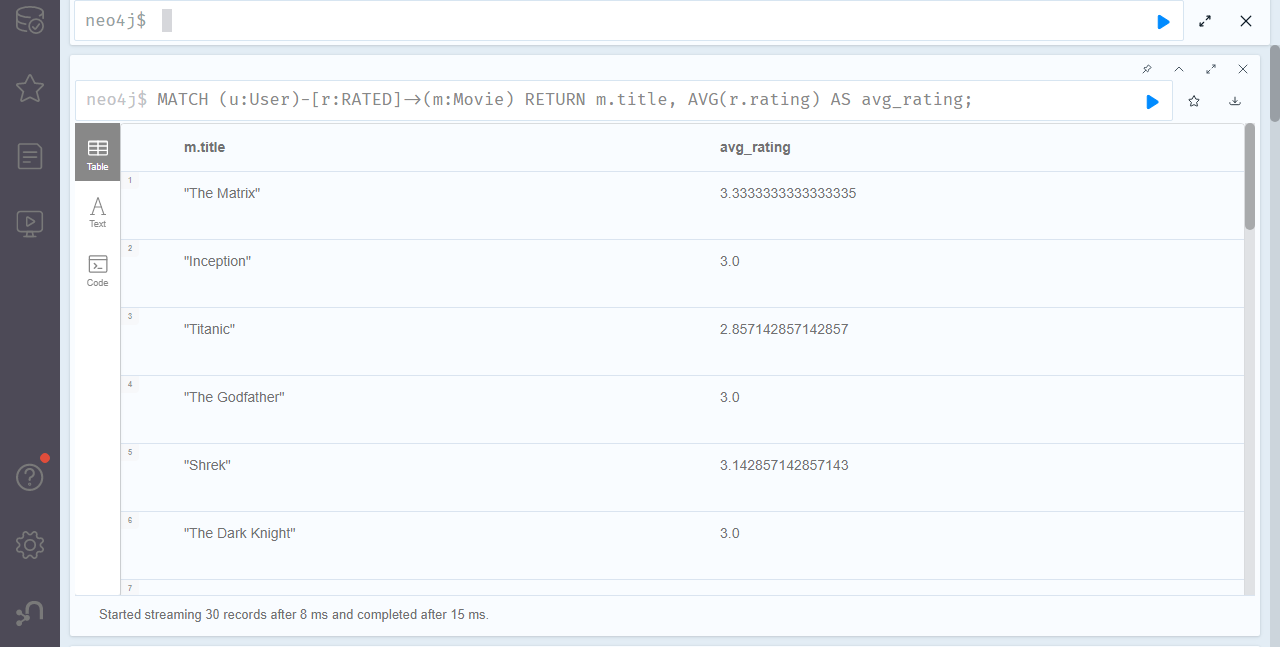
ORDER BY freq DESC;



1. Find average rating per movie.

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

RETURN m.title, AVG(r.rating) AS avg\_rating;



1. List movies that belong to multiple genres

MATCH (m:Movie)-[:BELONGS\_TO]->(g:Genre)

WITH m, COUNT(g) AS genre\_count

WHERE genre\_count > 1

RETURN m.title, genre\_count;



9.Find movies similar by genre

MATCH (m1:Movie {title: "The Matrix"})-[:BELONGS\_TO]->(g:Genre)<-[:BELONGS\_TO]-(m2:Movie)

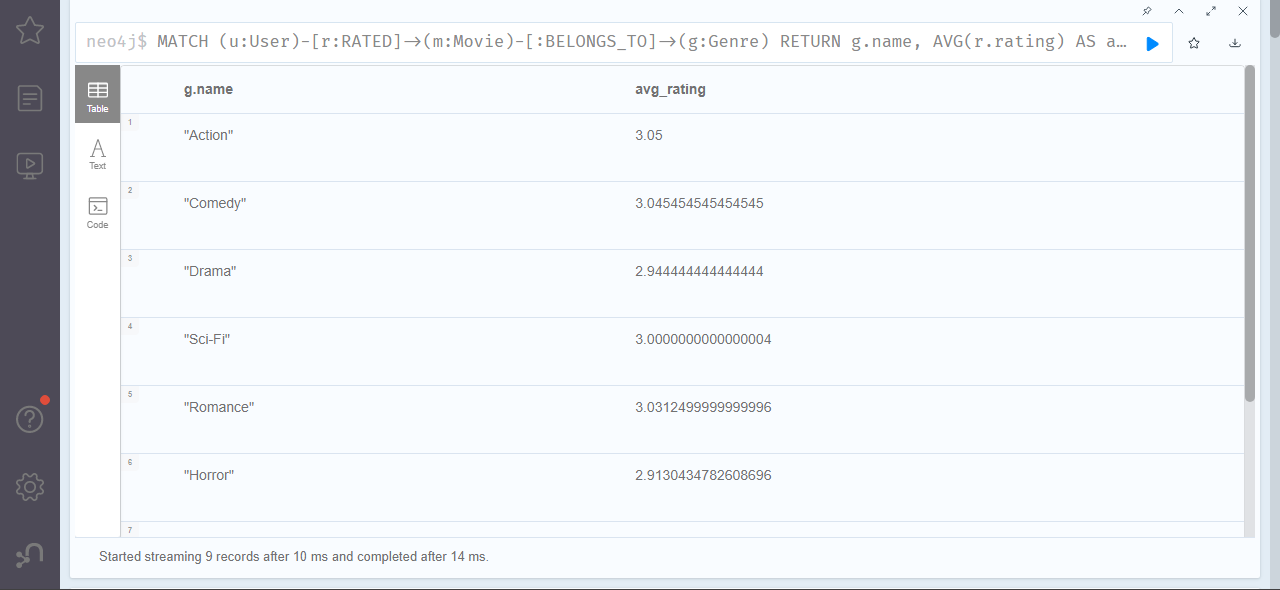
WHERE m1 <> m2

RETURN DISTINCT m2.title;

10. Find average ratings by each genre.

MATCH (u:User)-[r:RATED]->(m:Movie)-[:BELONGS\_TO]->(g:Genre)

RETURN g.name, AVG(r.rating) AS avg\_rating;



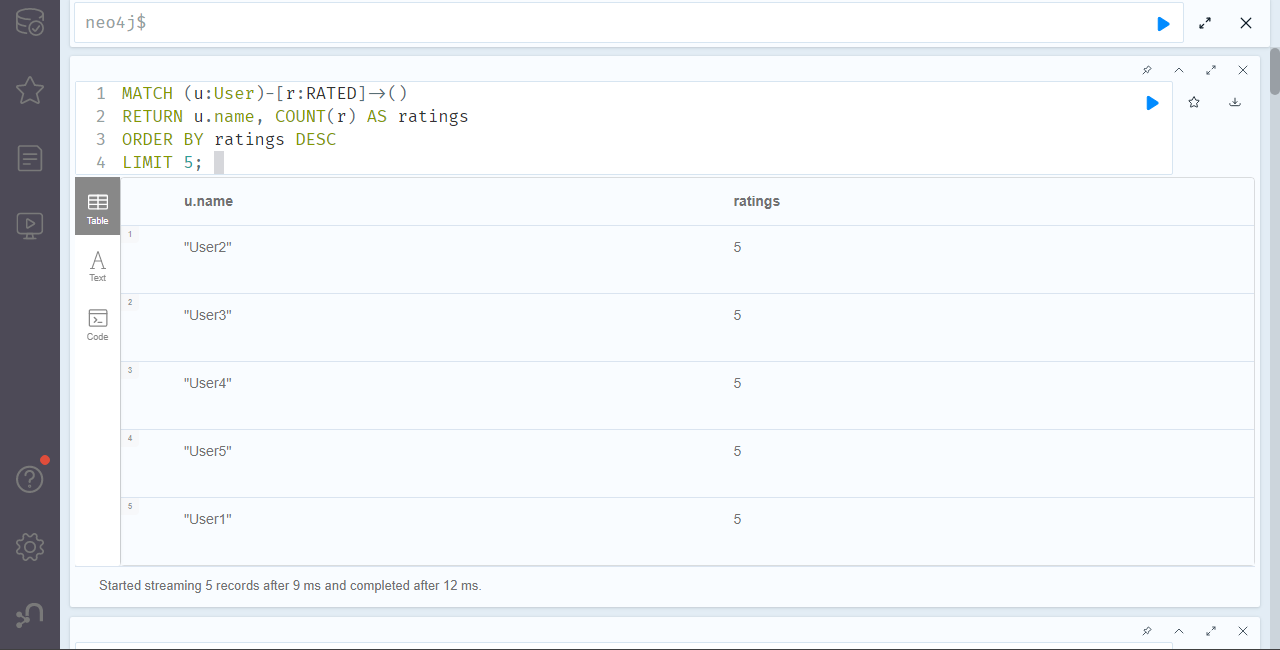
11.Find most active users by rating count

MATCH (u:User)-[r:RATED]->()

RETURN u.name, COUNT(r) AS ratings

ORDER BY ratings DESC

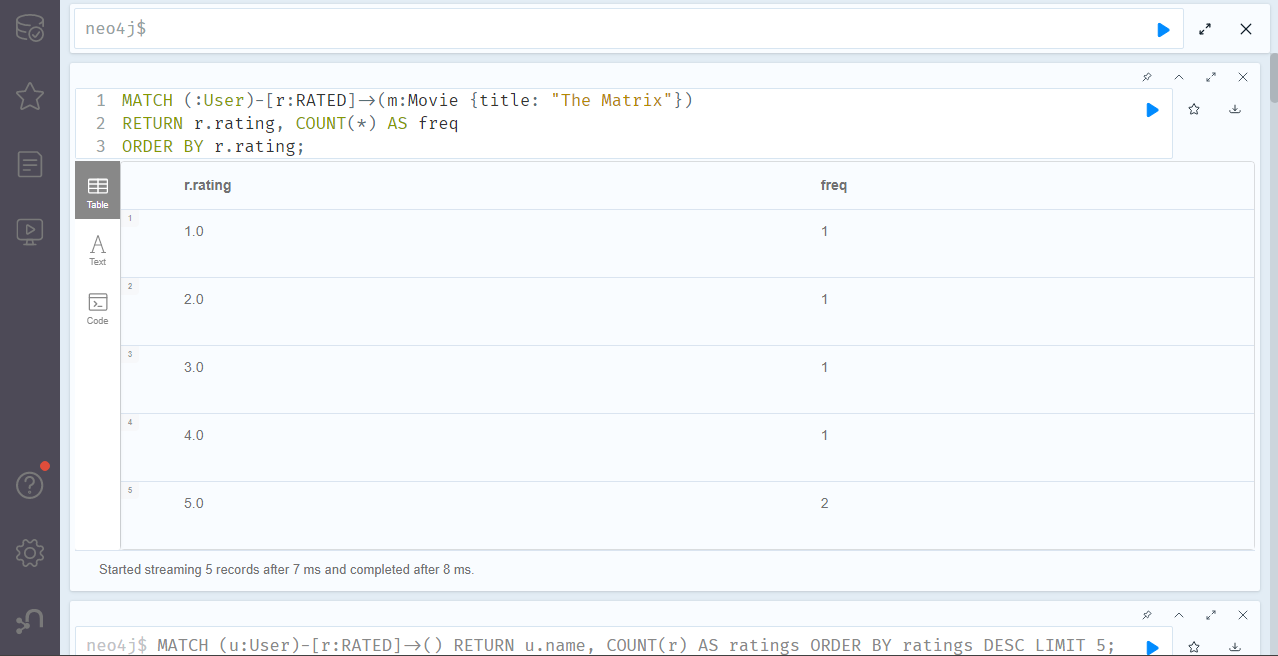
LIMIT 5;

12. Find the rating distribution for The Matrix

MATCH (:User)-[r:RATED]->(m:Movie {title: "The Matrix"})

RETURN r.rating, COUNT(\*) AS freq

ORDER BY r.rating;



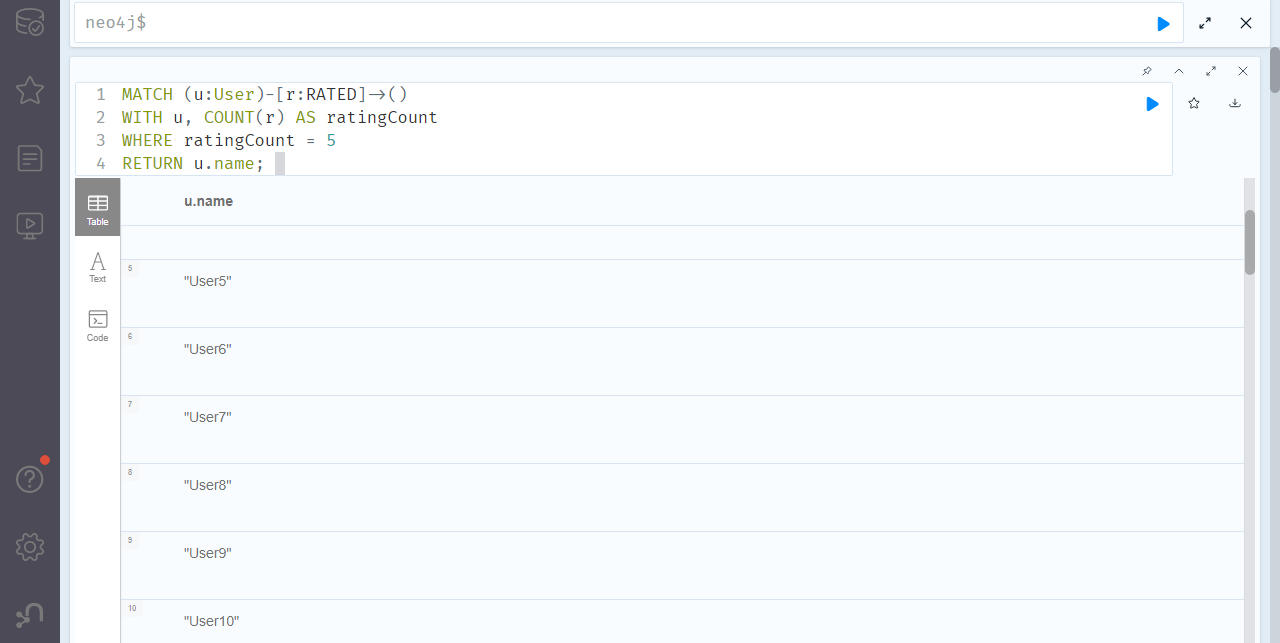
13.Find users that have rated 5 movies.

MATCH (u:User)-[r:RATED]->()

WITH u, COUNT(r) AS ratingCount

WHERE ratingCount = 5

RETURN u.name;



14.Find the most common genre.

MATCH (:Movie)-[:BELONGS\_TO]->(g:Genre)

RETURN g.name, COUNT(\*) AS movie\_count

ORDER BY movie\_count DESC

LIMIT 1;



15.Delete a rating

MATCH (u:User {name: "Bob"})-[r:RATED]->(m:Movie {title: "Titanic"})

DELETE r



16.Find users who have rated similar movies to User 5.

MATCH (a:User {name: "User5"})-[:RATED]->(m:Movie)<-[:RATED]-(u:User)

WHERE a <> u

RETURN u.name, COUNT(m) AS common\_movies

ORDER BY common\_movies DESC;



17. Find users who have only rated action movies.

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

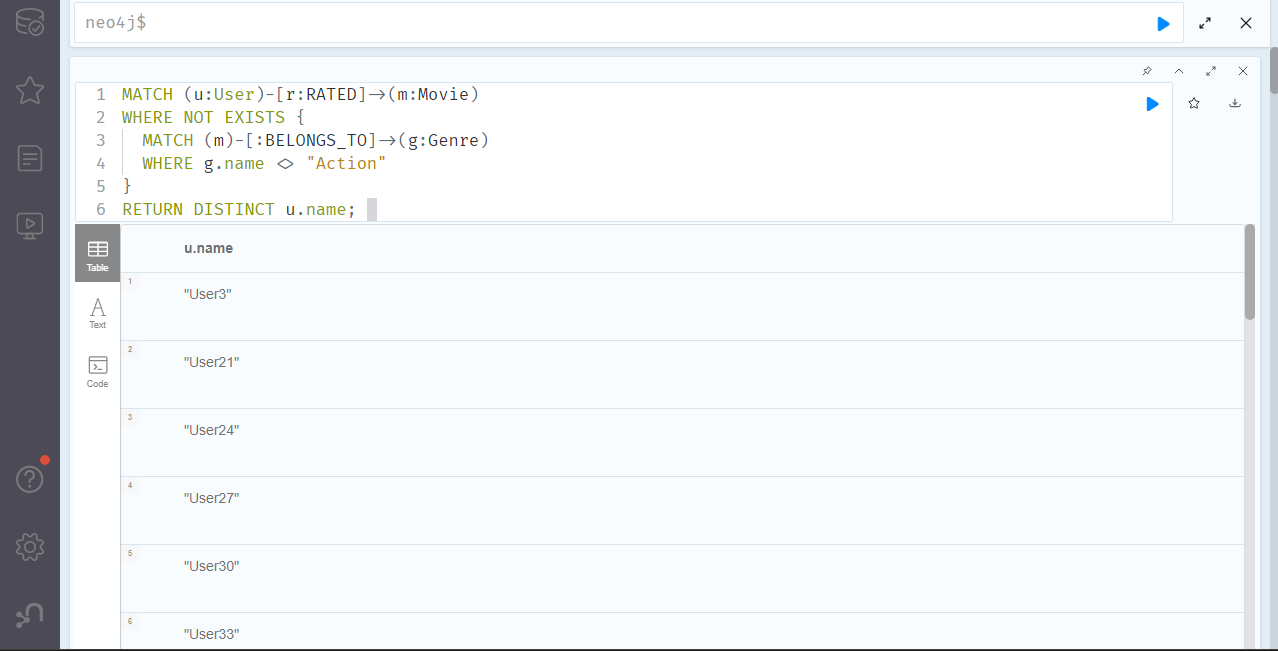
WHERE NOT EXISTS {

MATCH (m)-[:BELONGS\_TO]->(g:Genre)

WHERE g.name <> "Action"

}

RETURN DISTINCT u.name;



18. Find users who have rated only rated romance movies.

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

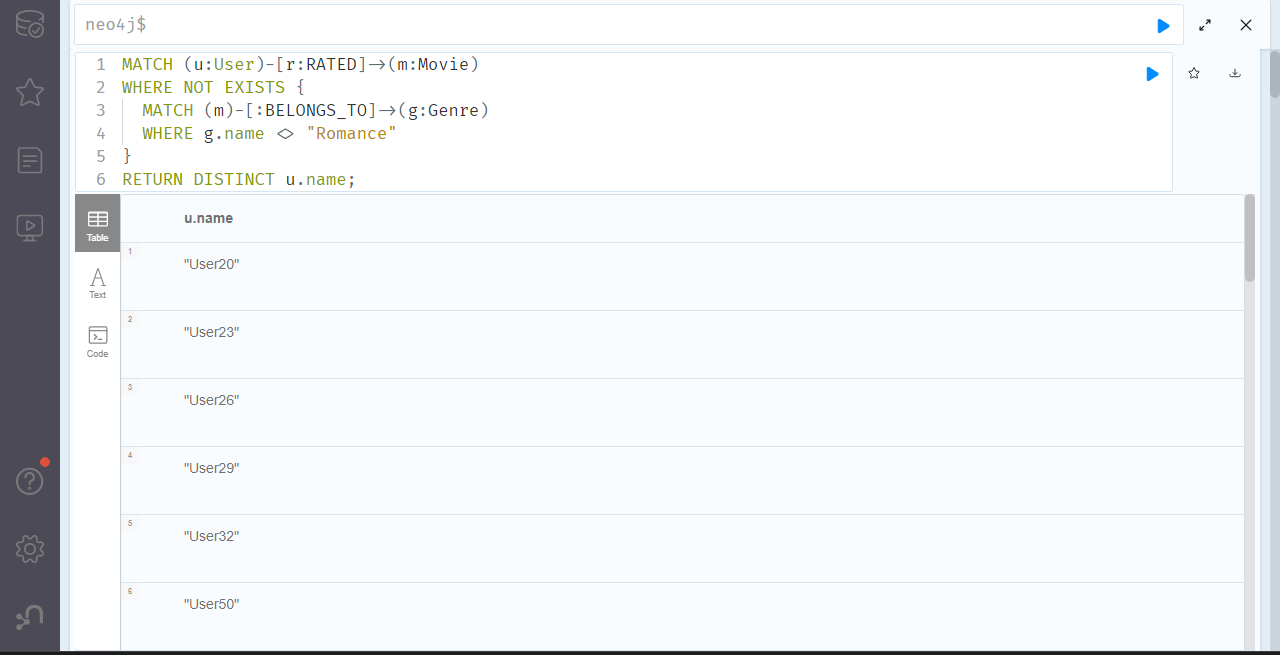
WHERE NOT EXISTS {

MATCH (m)-[:BELONGS\_TO]->(g:Genre)

WHERE g.name <> "Romance"

}

RETURN DISTINCT u.name;



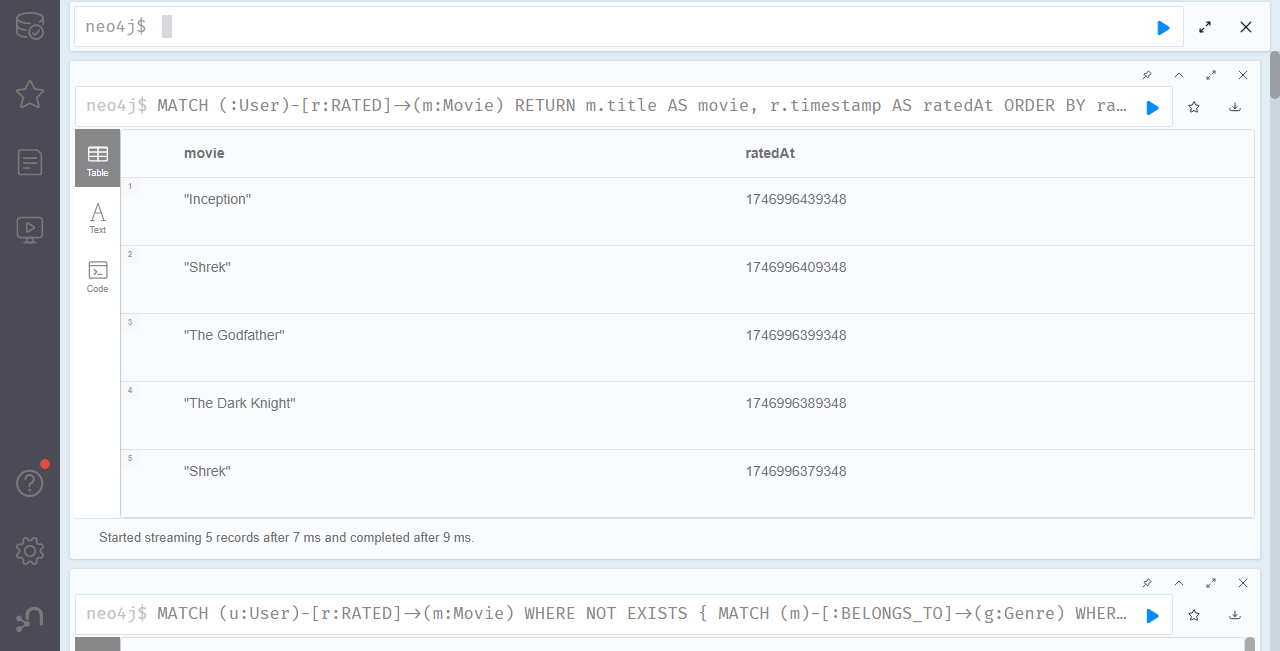
19. Find the most recently rated movie.

MATCH (:User)-[r:RATED]->(m:Movie)

RETURN m.title AS movie, r.timestamp AS ratedAt

ORDER BY ratedAt DESC

LIMIT 5;



20. Find the top 5 most rated movies.

MATCH (:User)-[r:RATED]->(m:Movie)

RETURN m.title AS movie, COUNT(r) AS numRatings

ORDER BY numRatings DESC

LIMIT 5;

