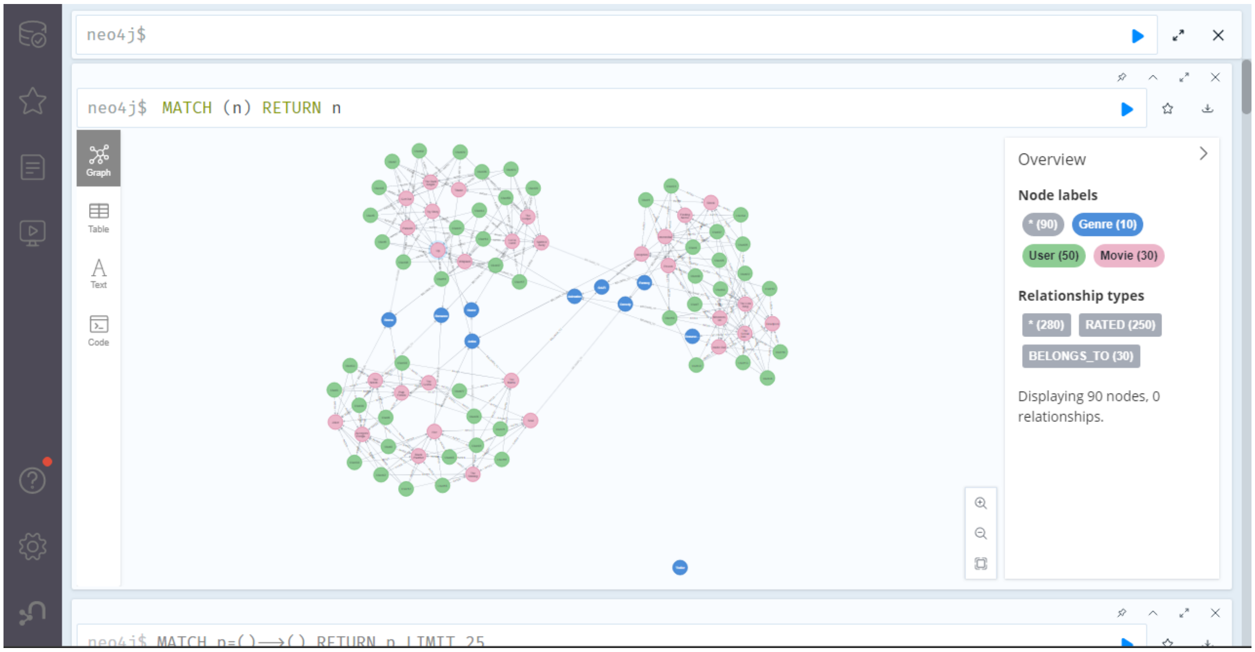
Kaziah Ofori

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

Match and Return



```
neo4j$
```

```
neo4j$ MATCH (n) RETURN n
```

Overview

Node labels

*(90) Genre (10) User (50) Movie (30)

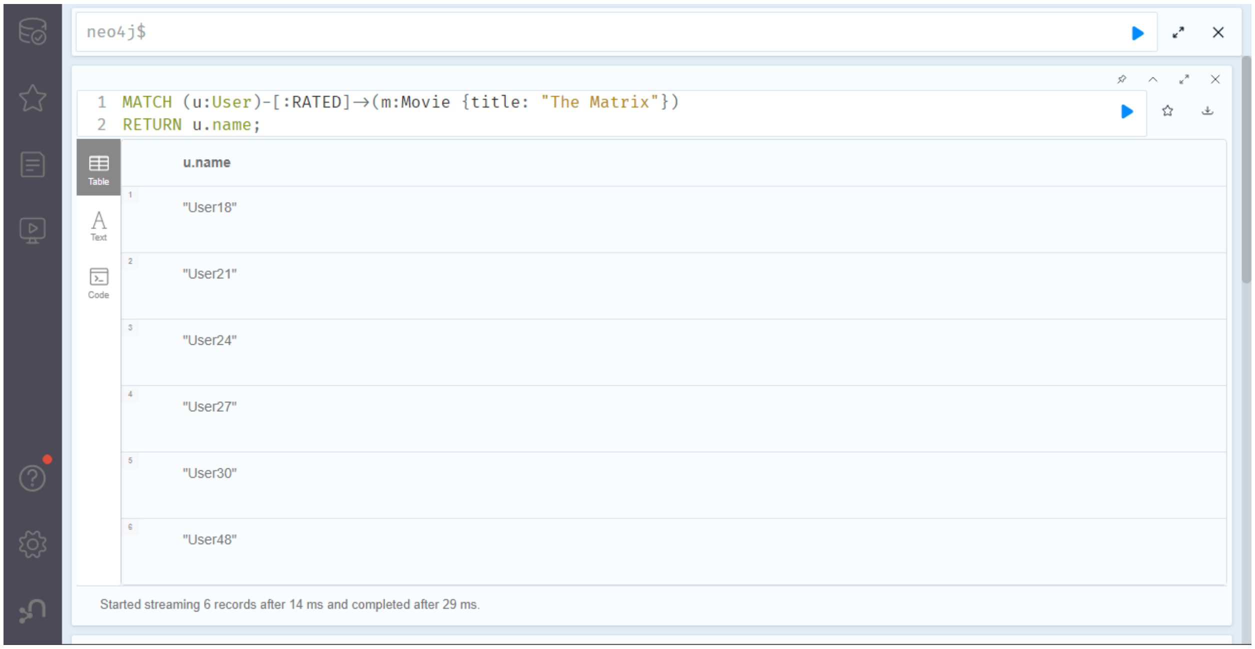
Relationship types

*(280) RATED (250) BELONGS_TO (30)

Displaying 90 nodes, 0 relationships.

```
neo4j$ MATCH p=()-->() RETURN p LIMIT 25
```

1. Find all users that rated “The Matrix”
   MATCH (u:User)-[:RATED]->(m:Movie {title: "The Matrix"})
   RETURN u.name;



```
neo4j$
```

```
MATCH (u:User)-[:RATED]→(m:Movie {title: "The Matrix"})
RETURN u.name;
```

	u.name
1	"User18"
2	"User21"
3	"User24"
4	"User27"
5	"User30"
6	"User48"

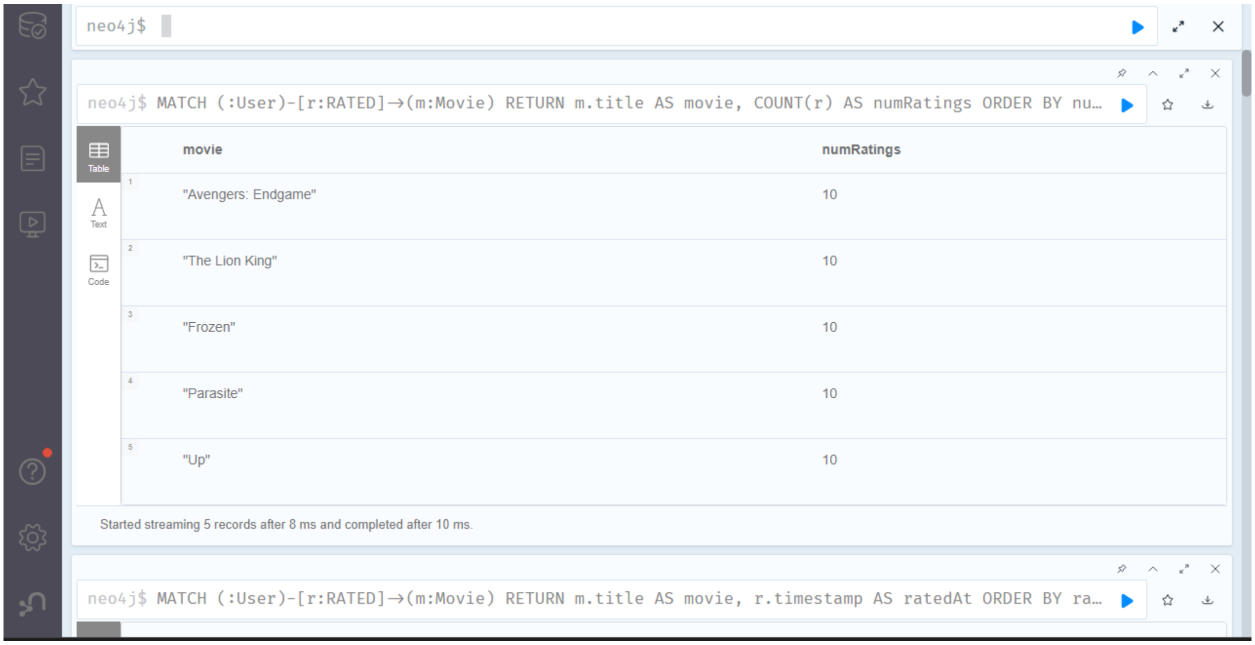
Started streaming 6 records after 14 ms and completed after 29 ms.

2. 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;



```
neo4j$
```

```
neo4j$ MATCH (:User)-[r:RATED]→(m:Movie) RETURN m.title AS movie, COUNT(r) AS numRatings ORDER BY nu…
```

	movie	numRatings
1	"Avengers: Endgame"	10
2	"The Lion King"	10
3	"Frozen"	10
4	"Parasite"	10
5	"Up"	10

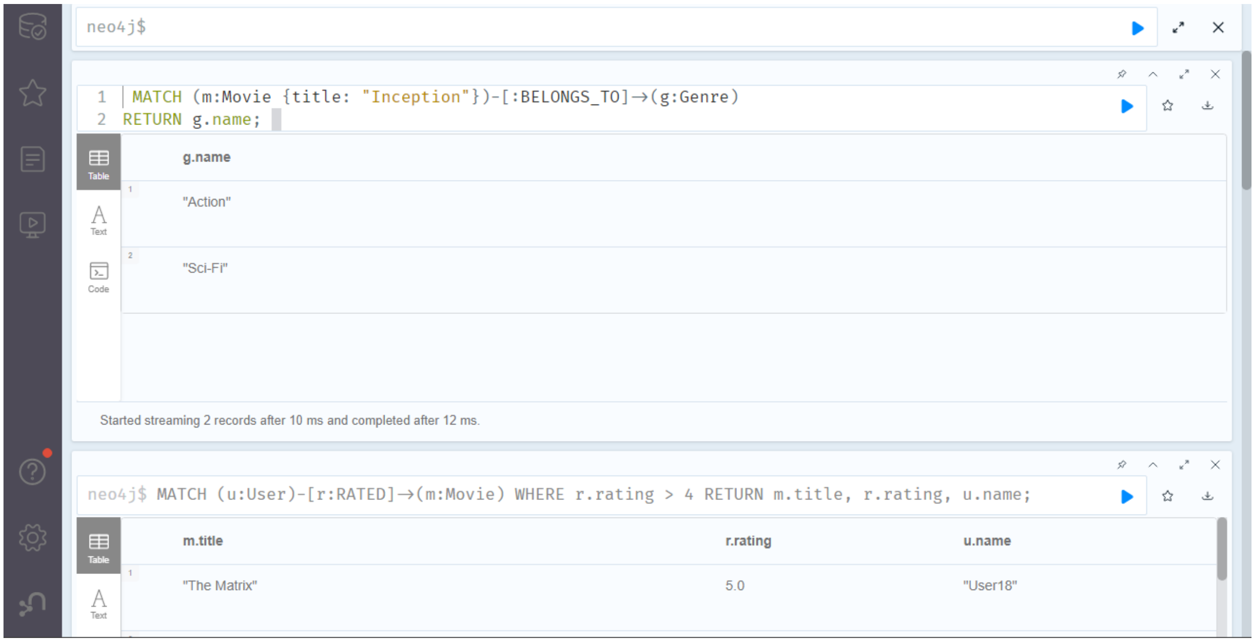
Started streaming 5 records after 8 ms and completed after 10 ms.

```
neo4j$ MATCH (:User)-[r:RATED]→(m:Movie) RETURN m.title AS movie, r.timestamp AS ratedAt ORDER BY ra…
```

3. List all genres for the movie Interception.

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

   RETURN g.name;



```
neo4j$
```

```
MATCH (m:Movie {title: "Inception"})-[:BELONGS_TO]→(g:Genre)
RETURN g.name;
```

	g.name
1	"Action"
2	"Sci-Fi"

Started streaming 2 records after 10 ms and completed after 12 ms.

```
neo4j$ MATCH (u:User)-[r:RATED]→(m:Movie) WHERE r.rating > 4 RETURN m.title, r.rating, u.name;
```

	m.title	r.rating	u.name
1	"The Matrix"	5.0	"User18"

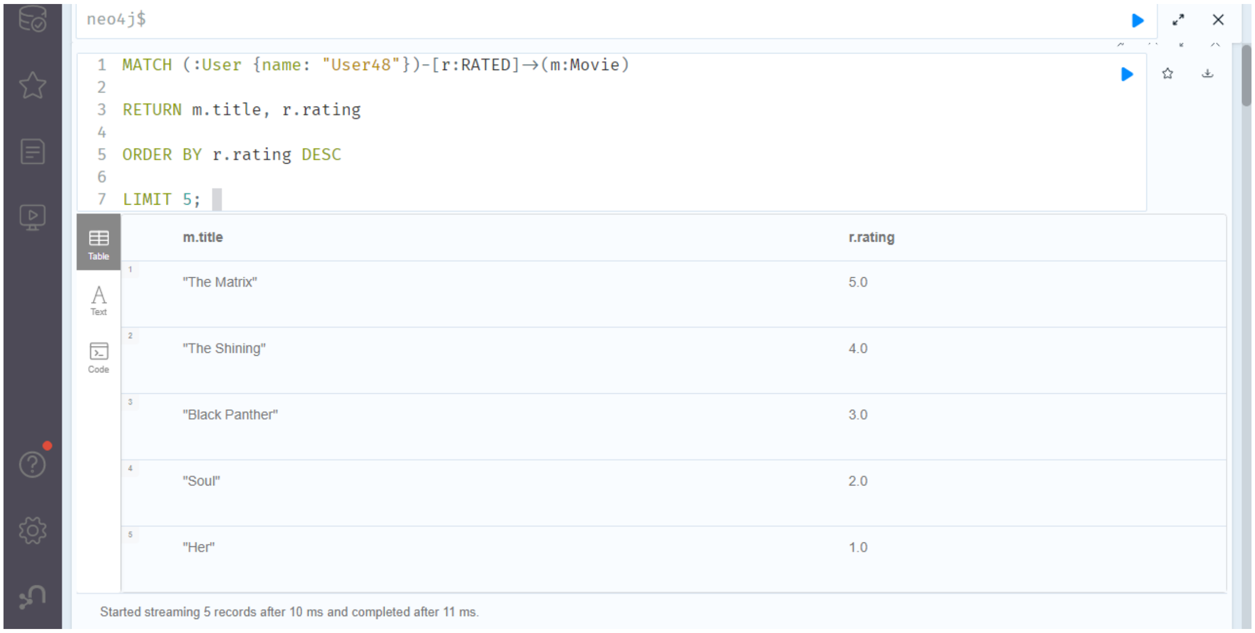
4. Find the top 5 movies rated by user 48.

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

RETURN m.title, r.rating

ORDER BY r.rating DESC

LIMIT 5;



```
neo4j$
```

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

RETURN m.title, r.rating

ORDER BY r.rating DESC

LIMIT 5;
```

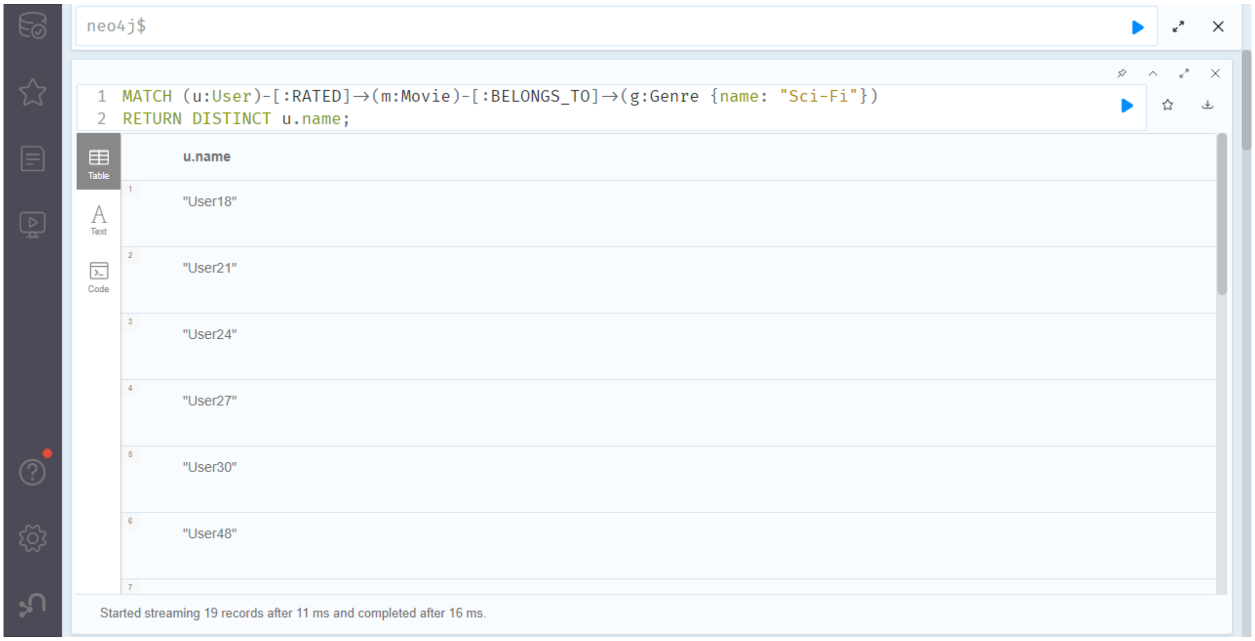
	m.title	r.rating
1	"The Matrix"	5.0
2	"The Shining"	4.0
3	"Black Panther"	3.0
4	"Soul"	2.0
5	"Her"	1.0

Started streaming 5 records after 10 ms and completed after 11 ms.

5. 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;



```
MATCH (u:User)-[:RATED]→(m:Movie)-[:BELONGS_TO]→(g:Genre {name: "Sci-Fi"})
RETURN DISTINCT u.name;
```

	u.name
1	"User18"
2	"User21"
3	"User24"
4	"User27"
5	"User30"
6	"User48"
7	

Started streaming 19 records after 11 ms and completed after 16 ms.

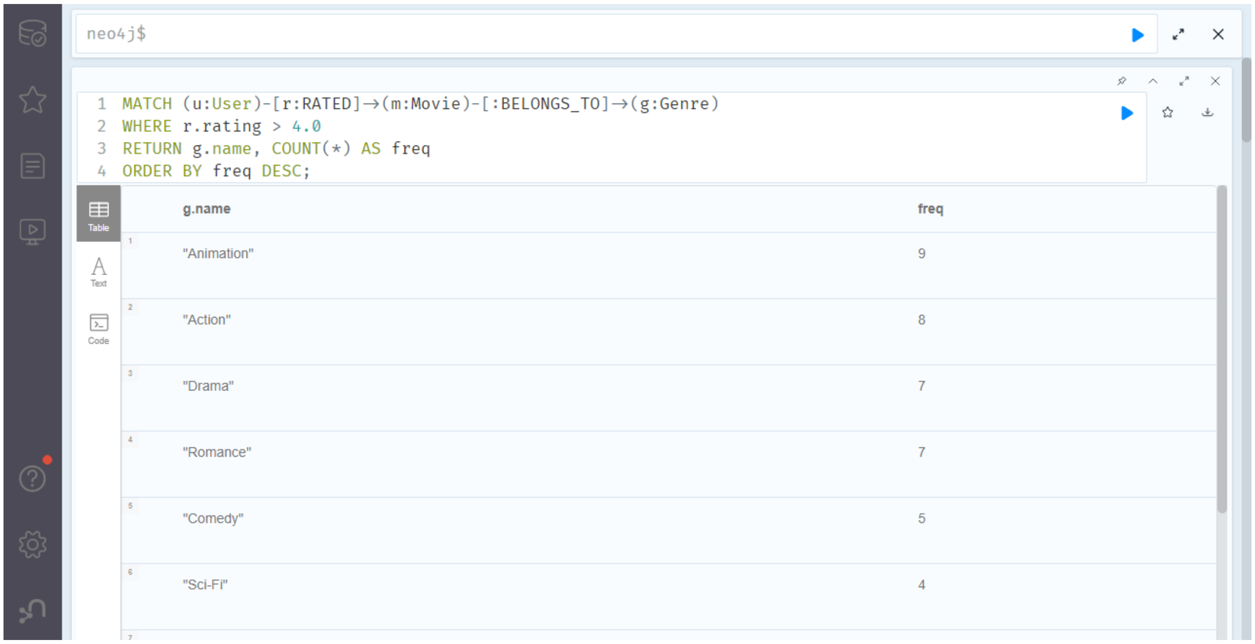
6. Find the genres most associated with high ratings.

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;



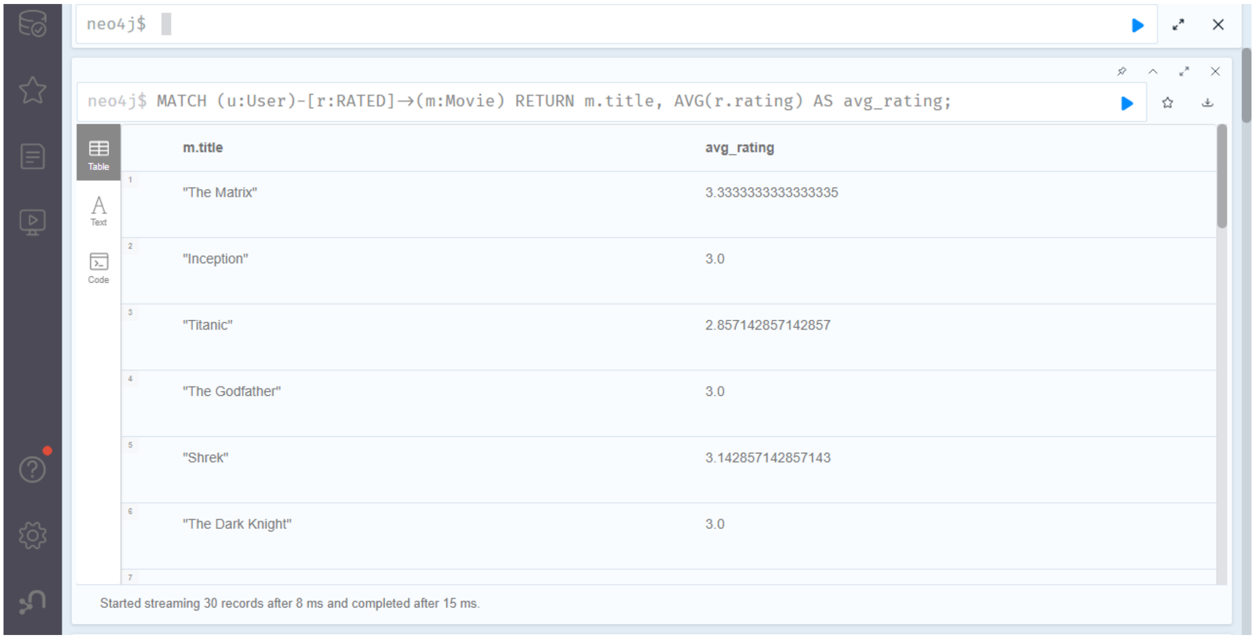
```
MATCH (u:User)-[r:RATED]->(m:Movie)-[:BELONGS_TO]->(g:Genre)
WHERE r.rating > 4.0
RETURN g.name, COUNT(*) AS freq
ORDER BY freq DESC;
```

g.name	freq
"Animation"	9
"Action"	8
"Drama"	7
"Romance"	7
"Comedy"	5
"Sci-Fi"	4

7. Find average rating per movie.

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

RETURN m.title, AVG(r.rating) AS avg_rating;



```
neo4j$ MATCH (u:User)-[r:RATED]->(m:Movie) RETURN m.title, AVG(r.rating) AS avg_rating;
```

m.title	avg_rating
"The Matrix"	3.3333333333333335
"Inception"	3.0
"Titanic"	2.857142857142857
"The Godfather"	3.0
"Shrek"	3.142857142857143
"The Dark Knight"	3.0

Started streaming 30 records after 8 ms and completed after 15 ms.

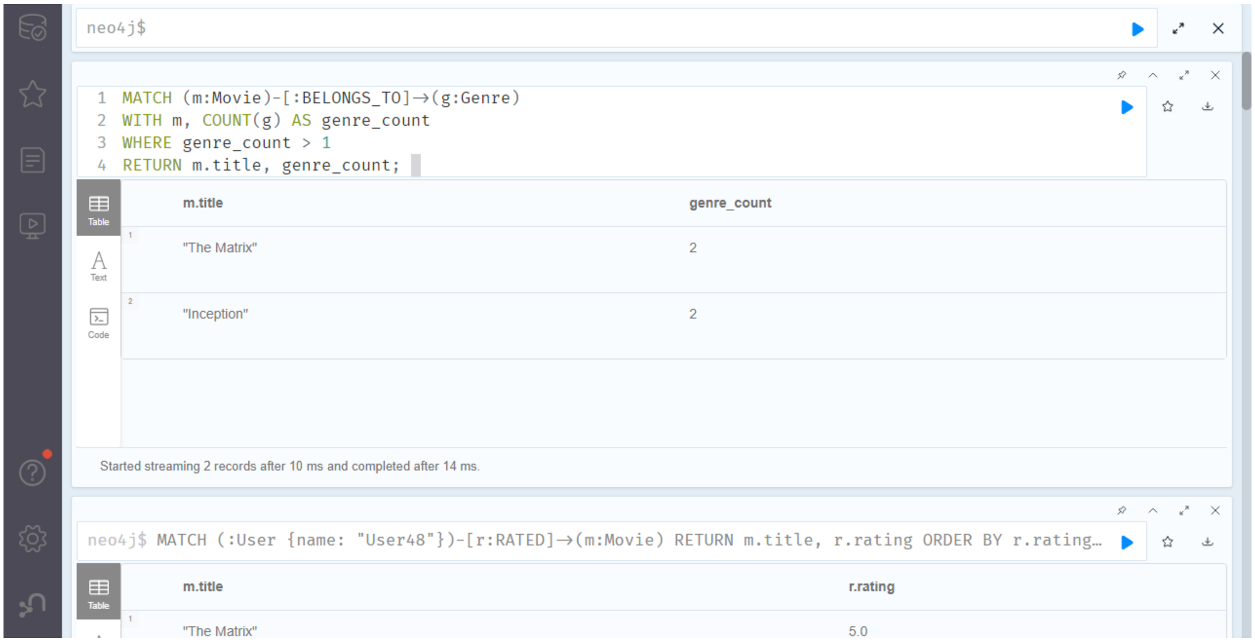
8. List movies that belong to multiple genres

```
MATCH (m:Movie)-[:BELONGS_TO]->(g:Genre)

WITH m, COUNT(g) AS genre_count

WHERE genre_count > 1

RETURN m.title, genre_count;
```



```
neo4j$
```

```
MATCH (m:Movie)-[:BELONGS_TO]→(g:Genre)
WITH m, COUNT(g) AS genre_count
WHERE genre_count > 1
RETURN m.title, genre_count;
```

m.title	genre_count
"The Matrix"	2
"Inception"	2

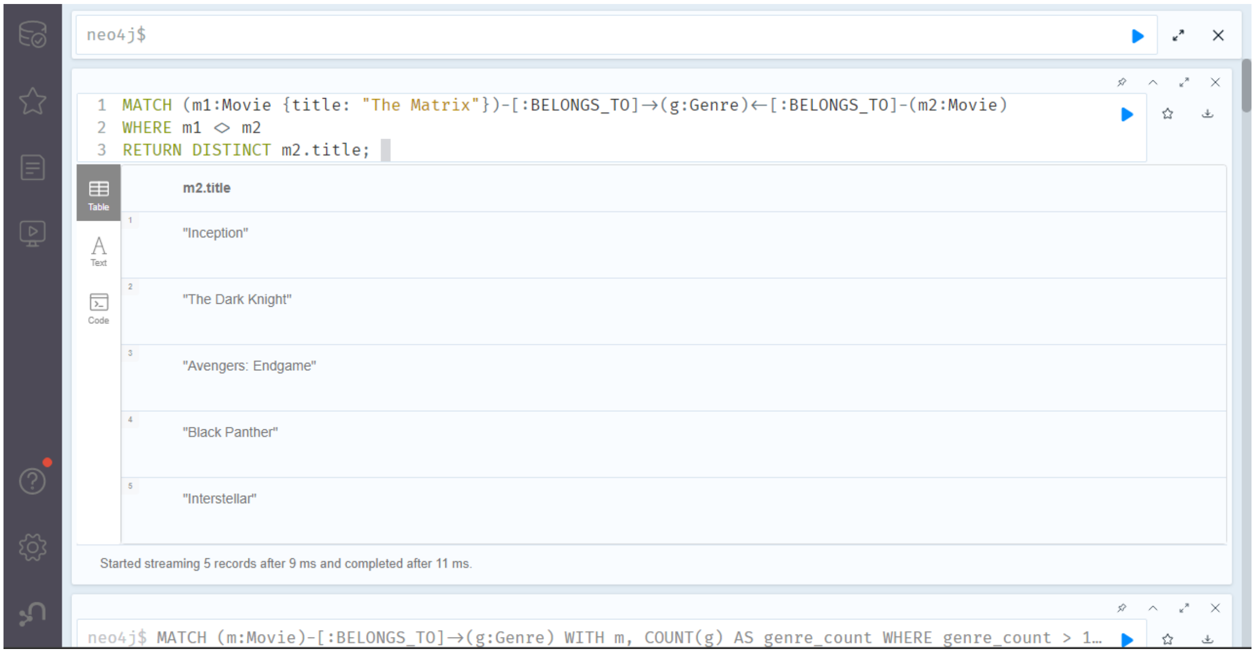
Started streaming 2 records after 10 ms and completed after 14 ms.

```
neo4j$ MATCH (:User {name: "User48"})-[r:RATED]→(m:Movie) RETURN m.title, r.rating ORDER BY r.rating…
```

m.title	r.rating
"The Matrix"	5.0

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;



```
neo4j$
```

```
MATCH (m1:Movie {title: "The Matrix"})-[:BELONGS_TO]→(g:Genre)←[:BELONGS_TO]-(m2:Movie)
WHERE m1 <> m2
RETURN DISTINCT m2.title;
```

m2.title
"Inception"
"The Dark Knight"
"Avengers: Endgame"
"Black Panther"
"Interstellar"

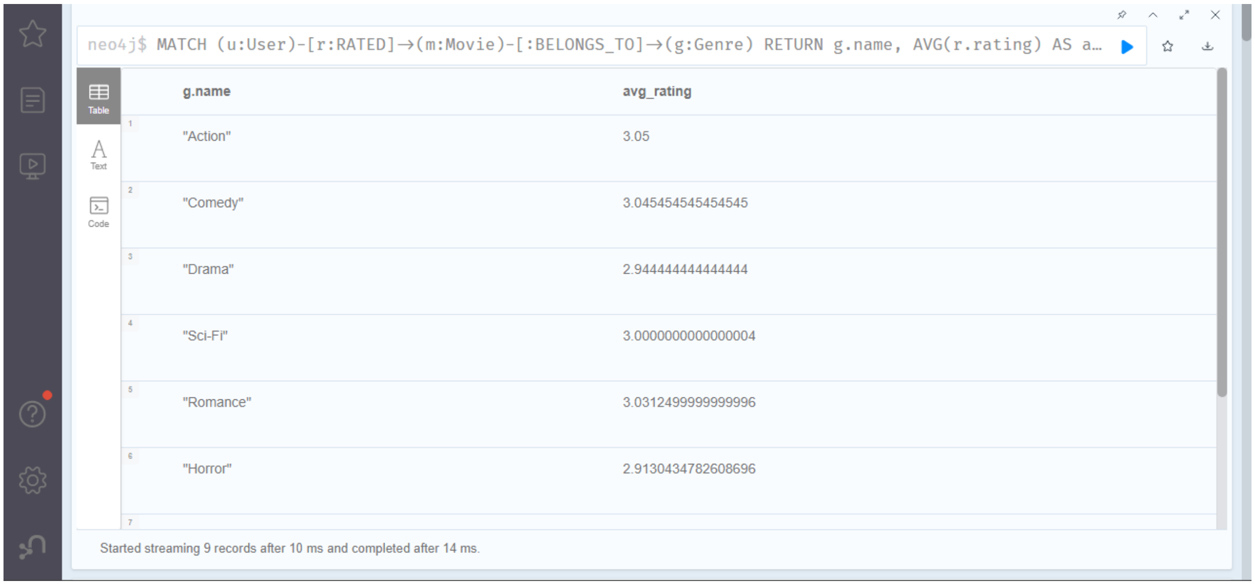
Started streaming 5 records after 9 ms and completed after 11 ms.

```
neo4j$ MATCH (m:Movie)-[:BELONGS_TO]→(g:Genre) WITH m, COUNT(g) AS genre_count WHERE genre_count > 1…
```

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;



```
neo4j$ MATCH (u:User)-[r:RATED]→(m:Movie)-[:BELONGS_TO]→(g:Genre) RETURN g.name, AVG(r.rating) AS a…
```

	g.name	avg_rating
1	"Action"	3.05
2	"Comedy"	3.045454545454545
3	"Drama"	2.944444444444444
4	"Sci-Fi"	3.0000000000000004
5	"Romance"	3.0312499999999996
6	"Horror"	2.9130434782608696
7		

Started streaming 9 records after 10 ms and completed after 14 ms.

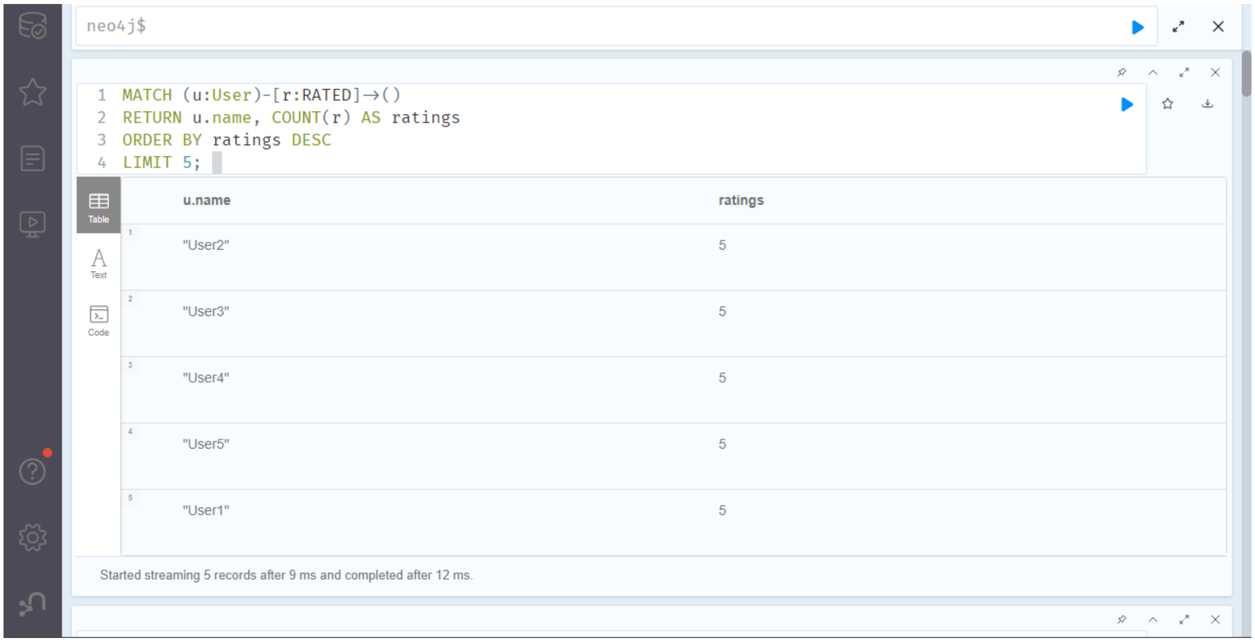
11.Find most active users by rating count

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

RETURN u.name, COUNT(r) AS ratings

ORDER BY ratings DESC

LIMIT 5;



```
MATCH (u:User)-[r:RATED]→()
RETURN u.name, COUNT(r) AS ratings
ORDER BY ratings DESC
LIMIT 5;
```

u.name	ratings
"User2"	5
"User3"	5
"User4"	5
"User5"	5
"User1"	5

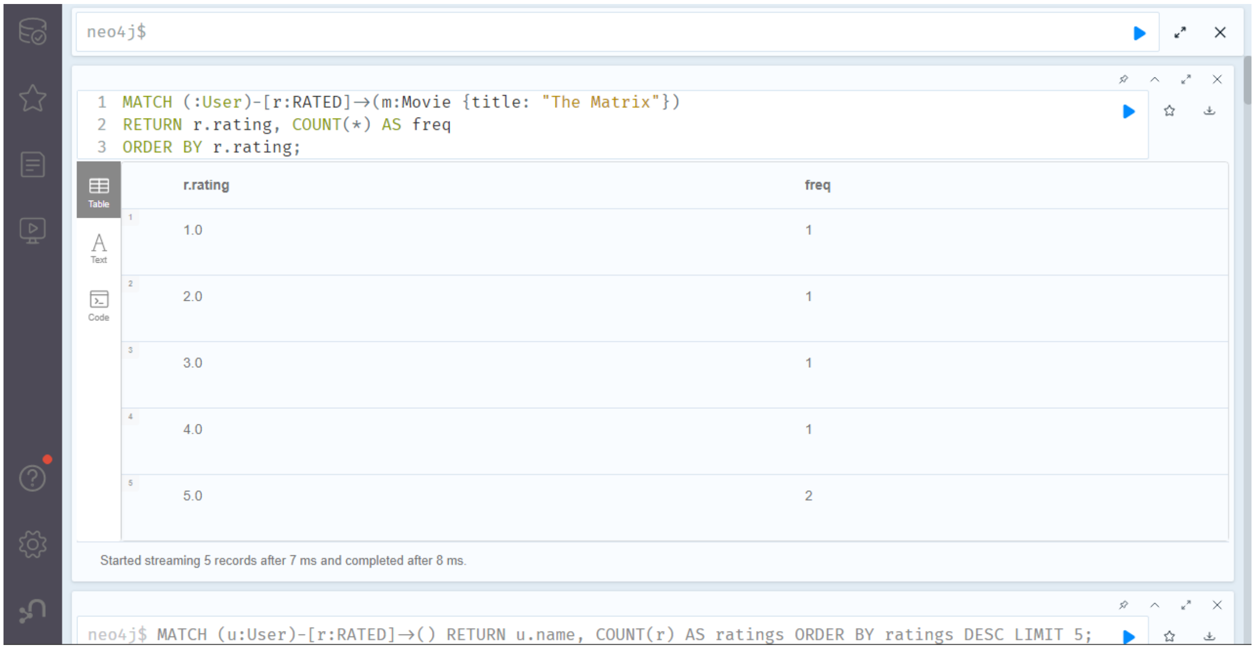
Started streaming 5 records after 9 ms and completed after 12 ms.

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;



```
MATCH (:User)-[r:RATED]→(m:Movie {title: "The Matrix"})
RETURN r.rating, COUNT(*) AS freq
ORDER BY r.rating;
```

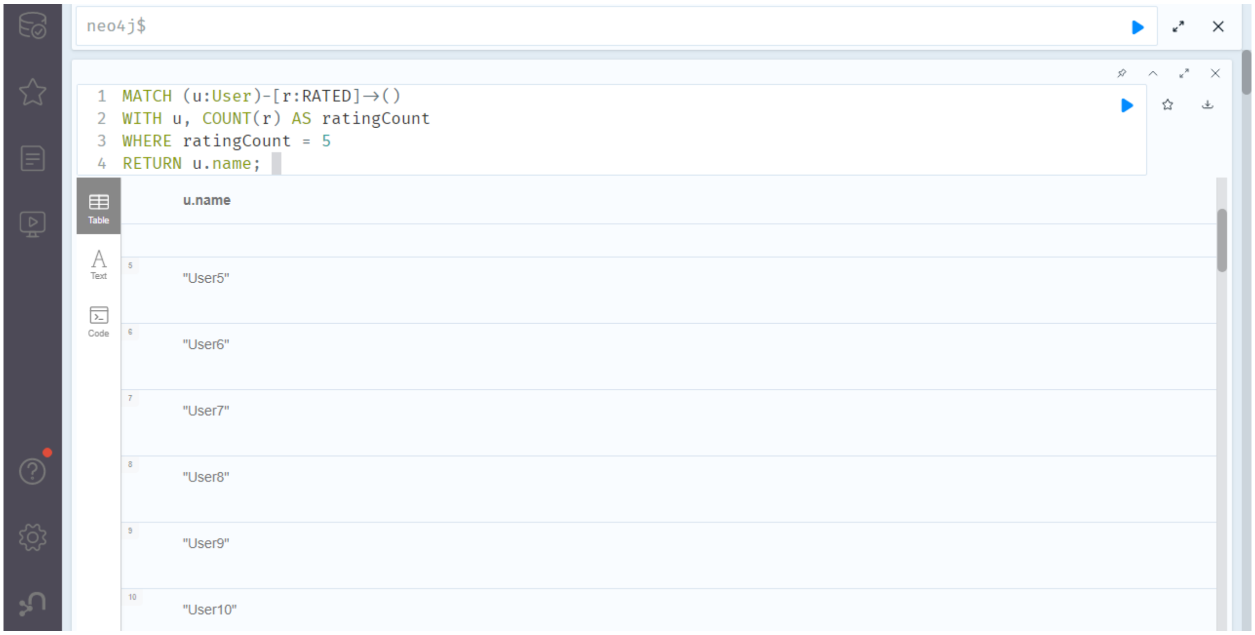
r.rating	freq
1.0	1
2.0	1
3.0	1
4.0	1
5.0	2

Started streaming 5 records after 7 ms and completed after 8 ms.

```
neo4j$ MATCH (u:User)-[r:RATED]→() RETURN u.name, COUNT(r) AS ratings ORDER BY ratings DESC LIMIT 5;
```

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;
```

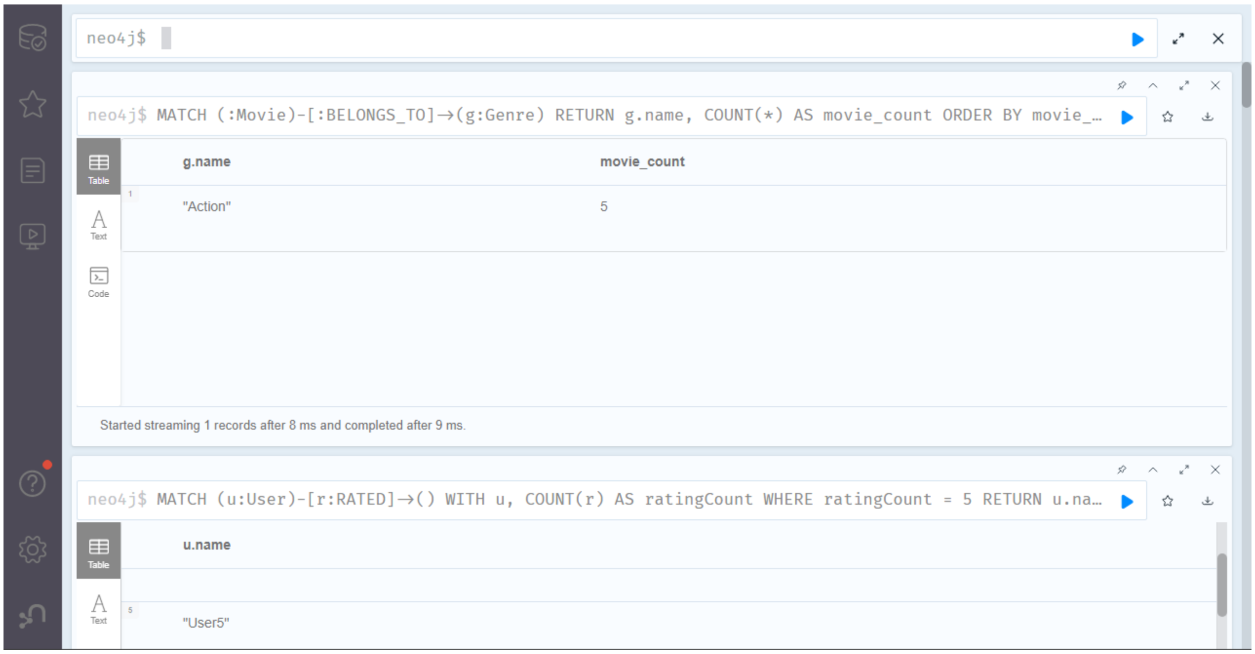


```
MATCH (u:User)-[r:RATED]→()
WITH u, COUNT(r) AS ratingCount
WHERE ratingCount = 5
RETURN u.name;
```

	u.name
5	"User5"
6	"User6"
7	"User7"
8	"User8"
9	"User9"
10	"User10"

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;
```



```
neo4j$
```

```
neo4j$ MATCH (:Movie)-[:BELONGS_TO]→(g:Genre) RETURN g.name, COUNT(*) AS movie_count ORDER BY movie_…
```

g.name	movie_count
"Action"	5

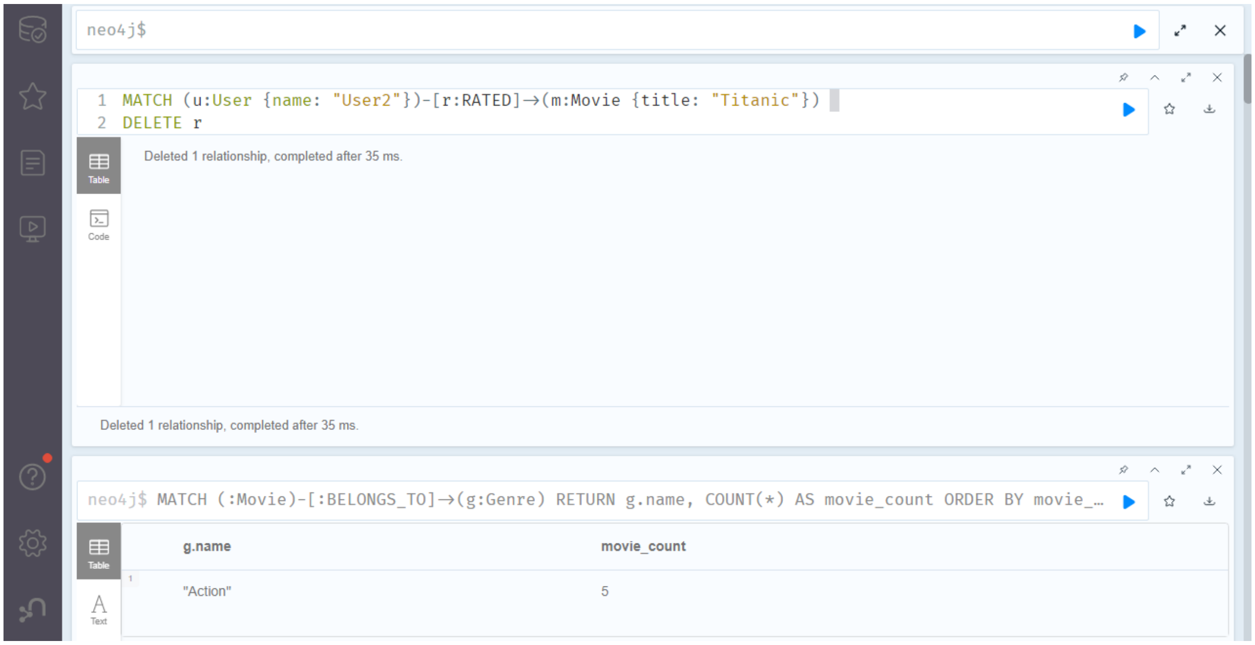
Started streaming 1 records after 8 ms and completed after 9 ms.

```
neo4j$ MATCH (u:User)-[r:RATED]→() WITH u, COUNT(r) AS ratingCount WHERE ratingCount = 5 RETURN u.na…
```

u.name
"User5"

15.Delete a rating

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



```
neo4j$
```

```
MATCH (u:User {name: "User2"})-[r:RATED]→(m:Movie {title: "Titanic"})
DELETE r
```

Deleted 1 relationship, completed after 35 ms.

Deleted 1 relationship, completed after 35 ms.

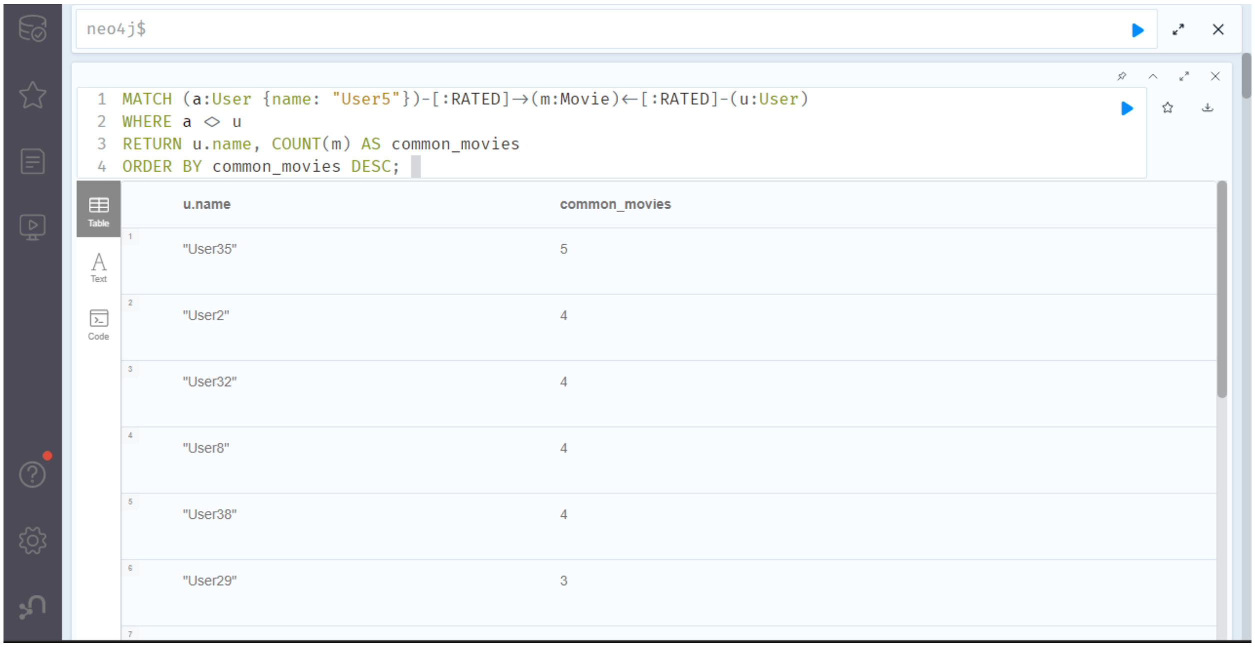
```
neo4j$ MATCH (:Movie)-[:BELONGS_TO]→(g:Genre) RETURN g.name, COUNT(*) AS movie_count ORDER BY movie_…
```

g.name	movie_count
"Action"	5

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;

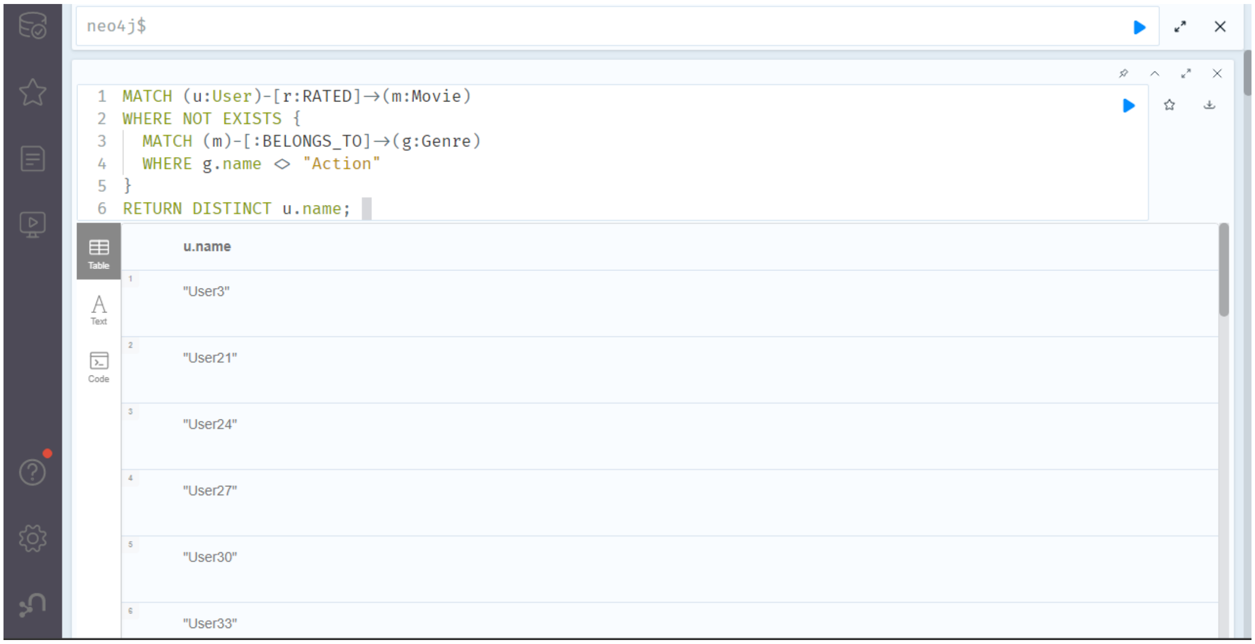


```
MATCH (a:User {name: "User5"})-[:RATED]→(m:Movie)←[:RATED]-(u:User)
WHERE a <> u
RETURN u.name, COUNT(m) AS common_movies
ORDER BY common_movies DESC;
```

u.name	common_movies
"User35"	5
"User2"	4
"User32"	4
"User8"	4
"User38"	4
"User29"	3

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;
```

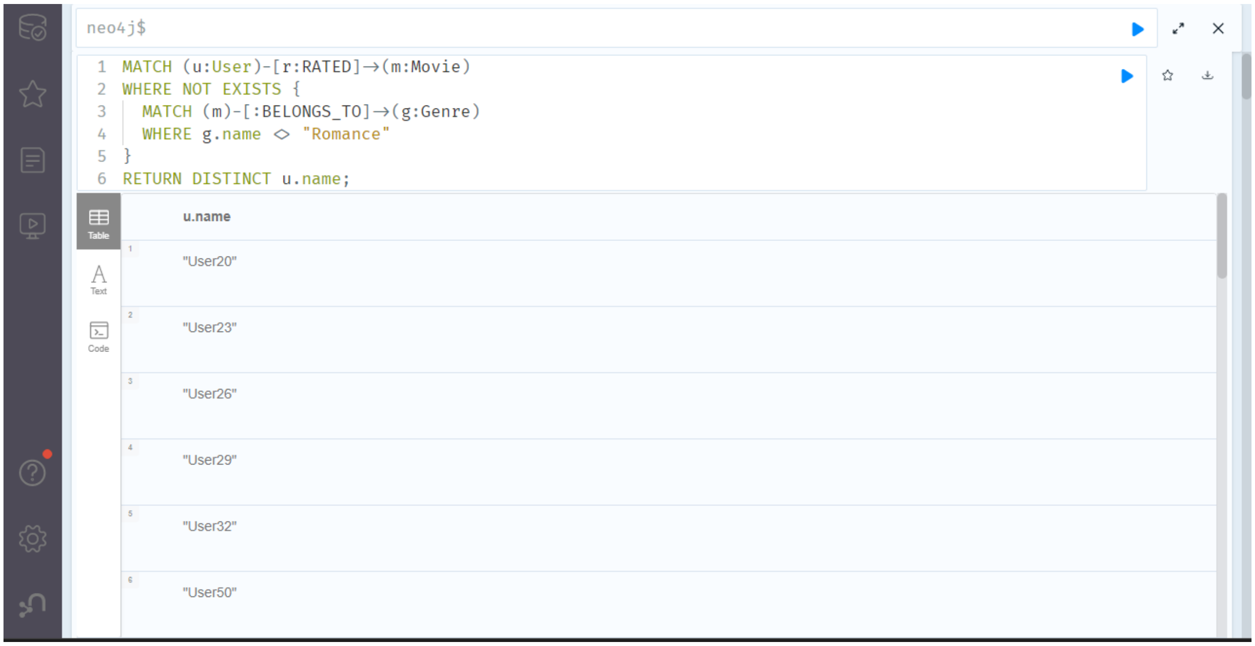


```
MATCH (u:User)-[r:RATED]→(m:Movie)
WHERE NOT EXISTS {
  MATCH (m)-[:BELONGS_TO]→(g:Genre)
  WHERE g.name <> "Action"
}
RETURN DISTINCT u.name;
```

	u.name
1	"User3"
2	"User21"
3	"User24"
4	"User27"
5	"User30"
6	"User33"

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;
```



```
MATCH (u:User)-[r:RATED]→(m:Movie)
WHERE NOT EXISTS {
  MATCH (m)-[:BELONGS_TO]→(g:Genre)
  WHERE g.name <> "Romance"
}
RETURN DISTINCT u.name;
```

u.name
"User20"
"User23"
"User26"
"User29"
"User32"
"User50"

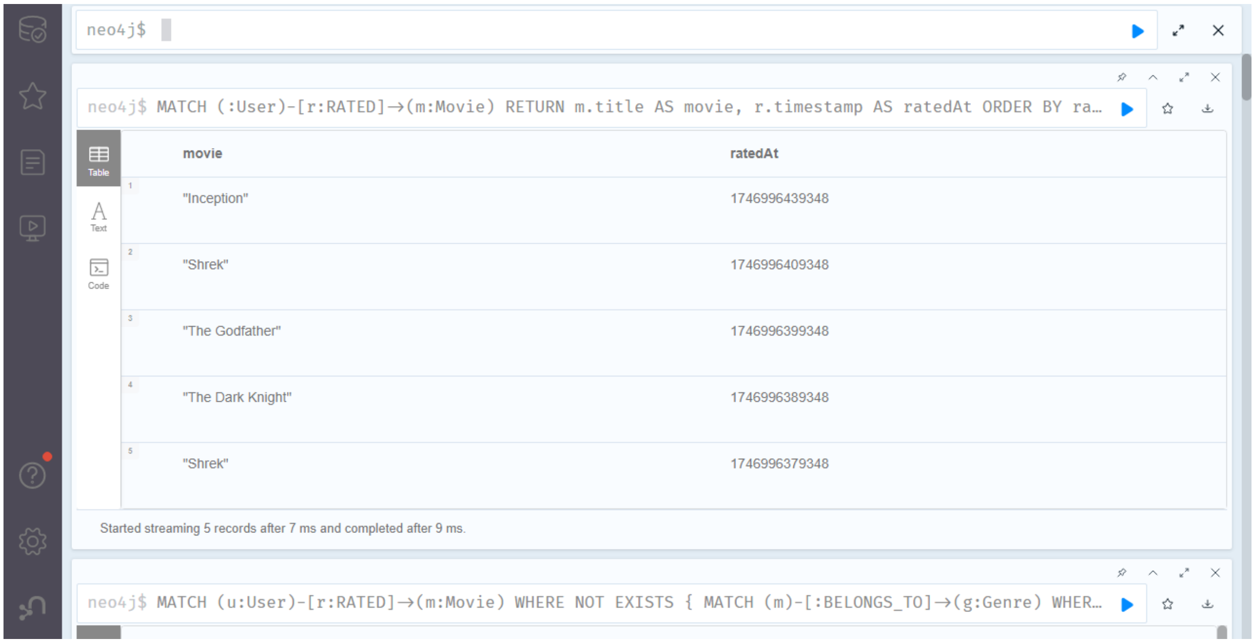
19. Find the most recently rated movie.

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

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

ORDER BY ratedAt DESC

LIMIT 5;



```
neo4j$ MATCH (:User)-[r:RATED]→(m:Movie) RETURN m.title AS movie, r.timestamp AS ratedAt ORDER BY ra…
```

movie	ratedAt
"Inception"	1746996439348
"Shrek"	1746996409348
"The Godfather"	1746996399348
"The Dark Knight"	1746996389348
"Shrek"	1746996379348

Started streaming 5 records after 7 ms and completed after 9 ms.

```
neo4j$ MATCH (u:User)-[r:RATED]→(m:Movie) WHERE NOT EXISTS { MATCH (m)-[:BELONGS_TO]→(g:Genre) WHER…
```

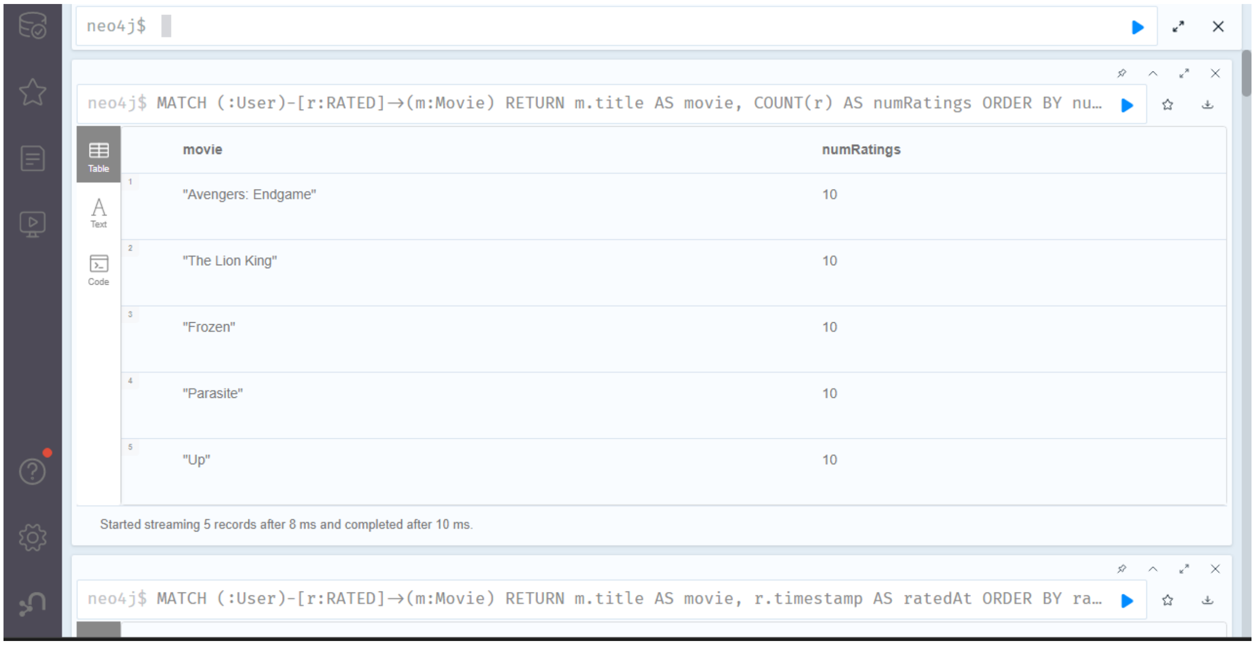
20. Find the top 5 most rated movies.

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

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

ORDER BY numRatings DESC

LIMIT 5;
```



```
neo4j$
```

```
neo4j$ MATCH (:User)-[r:RATED]→(m:Movie) RETURN m.title AS movie, COUNT(r) AS numRatings ORDER BY nu…
```

	movie	numRatings
1	"Avengers: Endgame"	10
2	"The Lion King"	10
3	"Frozen"	10
4	"Parasite"	10
5	"Up"	10

```
Started streaming 5 records after 8 ms and completed after 10 ms.
```

```
neo4j$ MATCH (:User)-[r:RATED]→(m:Movie) RETURN m.title AS movie, r.timestamp AS ratedAt ORDER BY ra…
```