Alexandra Mejia Vielman

11 May 2025

Music Streaming Graph Database

-My graph database models a music streaming platform, showing how users interact with artists and songs. It includes data on users, the artists they follow, and the songs they listen to. By using a graph database, we can easily track connections and patterns between users, songs, and artists—giving us a clear picture of how music is experienced on the platform. We can answer questions like, for example: Which songs are most listened to? Which artists have the most followers? What genres are most popular among users from different countries?

Node Labels:

- User – listeners on the platform
- Artist – musicians who create songs
- Song – individual tracks

Relationship Types:

- LISTENED_TO (User -->Song)
- FOLLOWS (User -->Artist)
- PERFORMS (Artist -->Song)

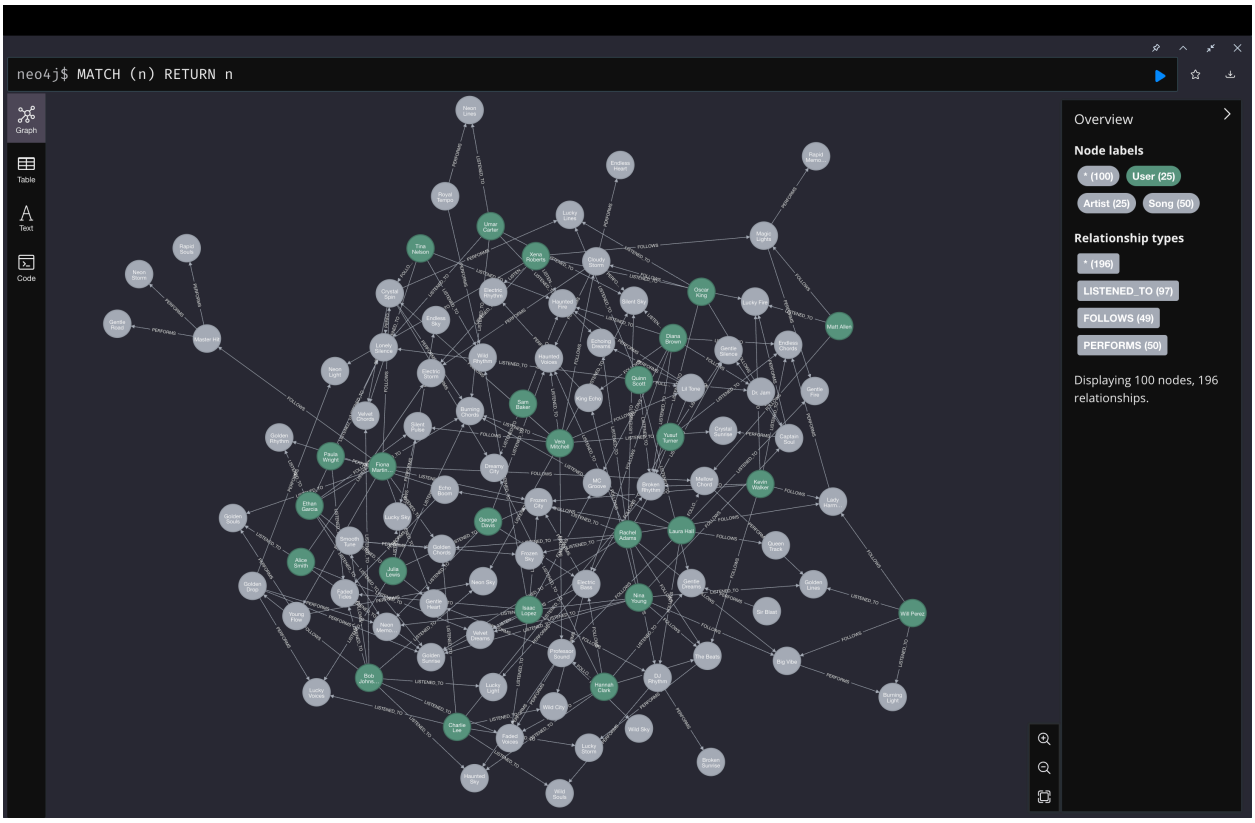
Properties (attributes):

- User: username, age, country
- Artist: name, genre
- Song: title, release_year, duration

-- Cypher Code –

```
LOAD CSV WITH HEADERS FROM 'file:///users.csv' AS row
RETURN row LIMIT 5;

LOAD CSV WITH HEADERS FROM 'file:///users.csv' AS row
WITH row
WHERE row.username IS NOT NULL AND row.username <> ''
MERGE (u:User {username: row.username})
SET u.name = row.name,
   u.age = toInteger(row.age),
   u.country = row.country;

LOAD CSV WITH HEADERS FROM 'file:///artists.csv' AS row
WITH row
WHERE row.name IS NOT NULL AND row.name <> ''
MERGE (a:Artist {name: row.name})
SET a.genre = row.genre;

LOAD CSV WITH HEADERS FROM 'file:///songs.csv' AS row
WITH row
WHERE row.title IS NOT NULL AND row.title <> ''
MERGE (s:Song {title: row.title})
SET s.release_year = toInteger(row.release_year),
   s.duration = toInteger(row.duration);


LOAD CSV WITH HEADERS FROM 'file:///performs.csv' AS row
```

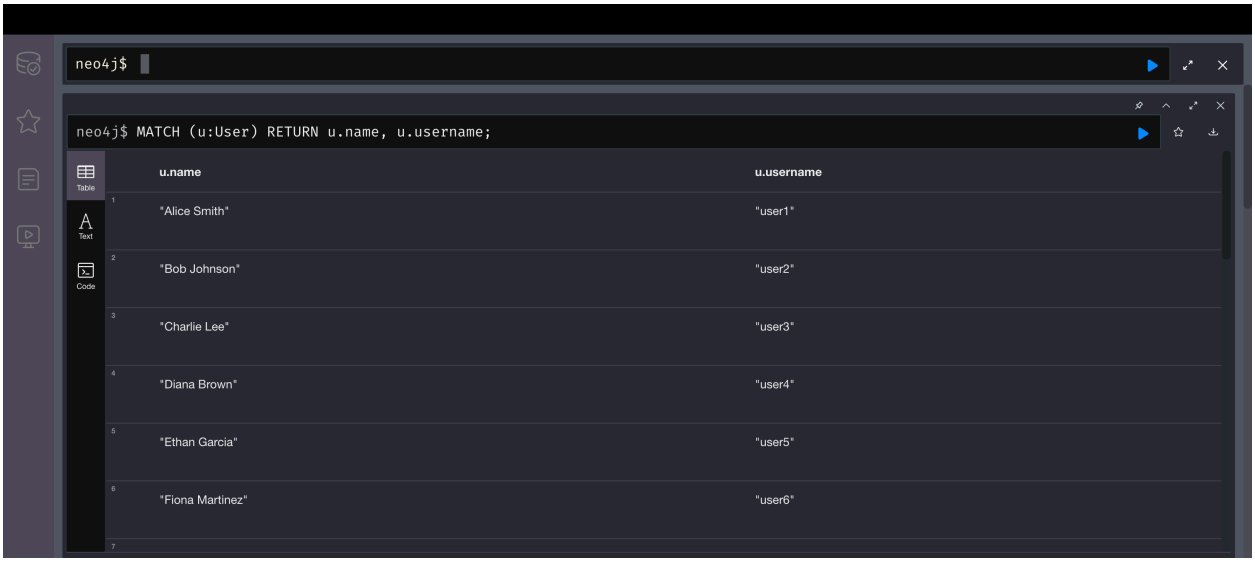
```
WITH row
WHERE row.artist IS NOT NULL AND row.song IS NOT NULL
MATCH (a:Artist {name: row.artist})
MATCH (s:Song {title: row.song})
MERGE (a)-[:PERFORMS]->(s);

LOAD CSV WITH HEADERS FROM 'file:///listens.csv' AS row
WITH row
WHERE row.user IS NOT NULL AND row.song IS NOT NULL
MATCH (u:User {username: row.user})
MATCH (s:Song {title: row.song})
MERGE (u)-[:LISTENED_TO]->(s);

LOAD CSV WITH HEADERS FROM 'file:///follows.csv' AS row
WITH row
WHERE row.user IS NOT NULL AND row.artist IS NOT NULL
MATCH (u:User {username: row.user})
MATCH (a:Artist {name: row.artist})
MERGE (u)-[:FOLLOWS]->(a);



MATCH (n) RETURN n
```



--20 queries—

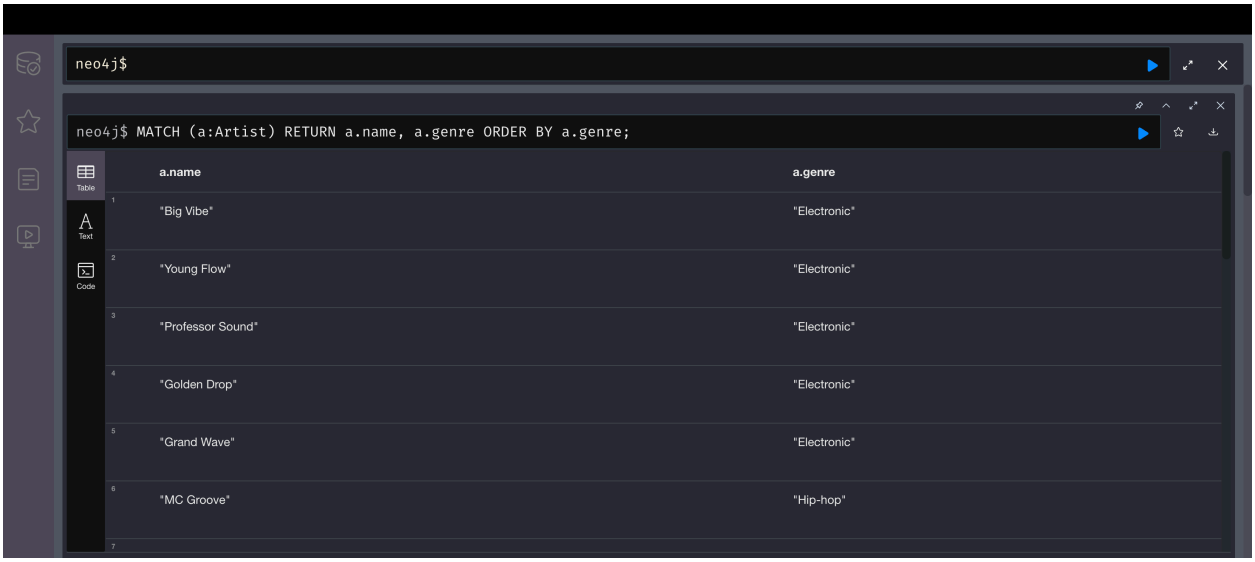
1. List all users

MATCH (u:User) RETURN u.name, u.username;



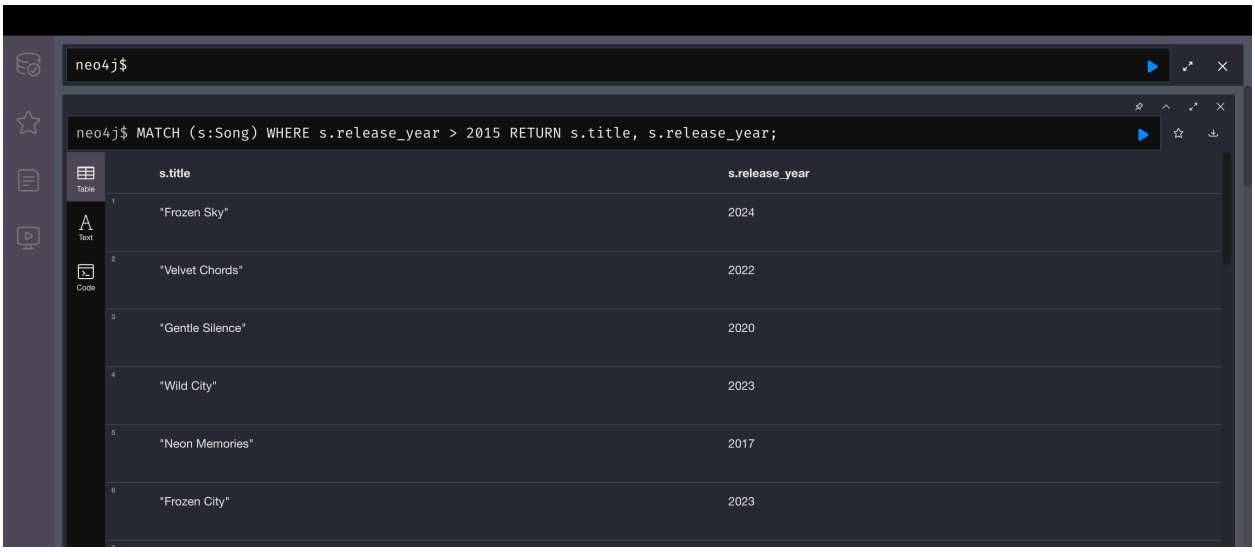
2. List all artists by genre

MATCH (a:Artist) RETURN a.name, a.genre ORDER BY a.genre;



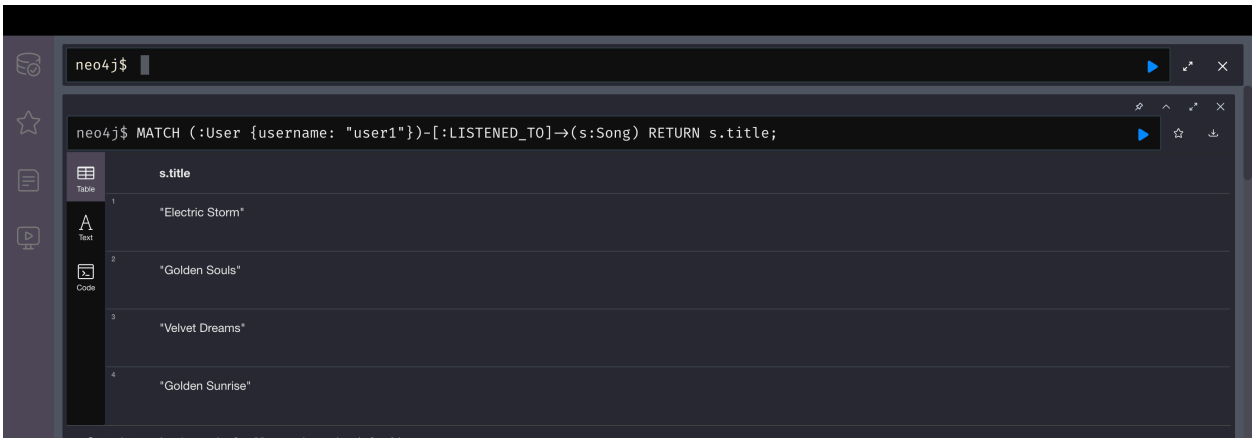
3. List all songs released after 2015

MATCH (s:Song) WHERE s.release_year > 2015 RETURN s.title, s.release_year;



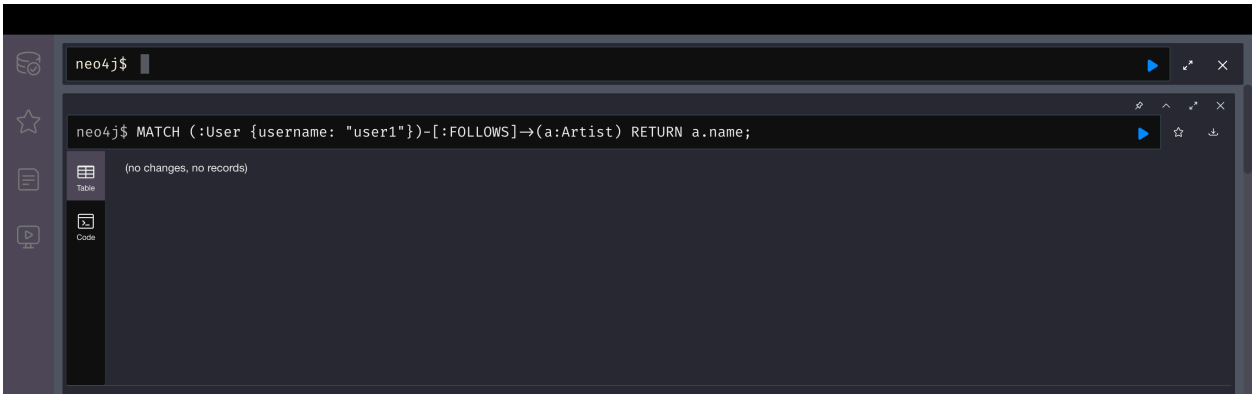
4. Which songs did user1 listen to?

MATCH (:User {username: "user1"})-[:LISTENED_TO]->(s:Song) RETURN s.title;



5. Which artists does user1 follow?

MATCH (:User {username: "user1"})-[:FOLLOWS]->(a:Artist) RETURN a.name;



6. Which songs were performed by 'DJ Rhythm'?

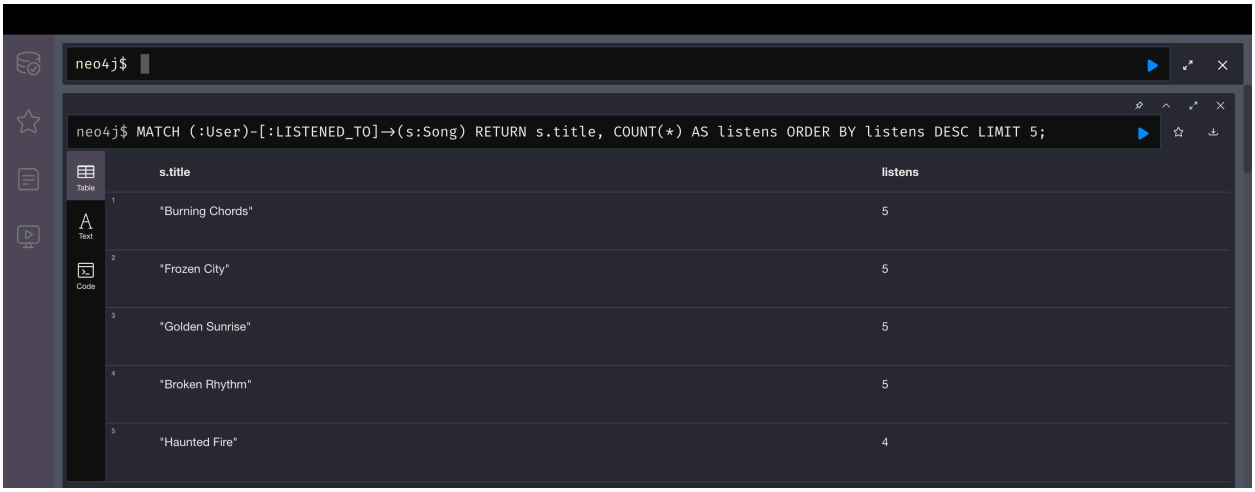
MATCH (:Artist {name: "DJ Rhythm"})-[:PERFORMS]->(s:Song) RETURN s.title;

7. Top 5 most listened-to songs

MATCH (:User)-[:LISTENED_TO]->(s:Song)

RETURN s.title, COUNT(*) AS listens

ORDER BY listens DESC LIMIT 5;

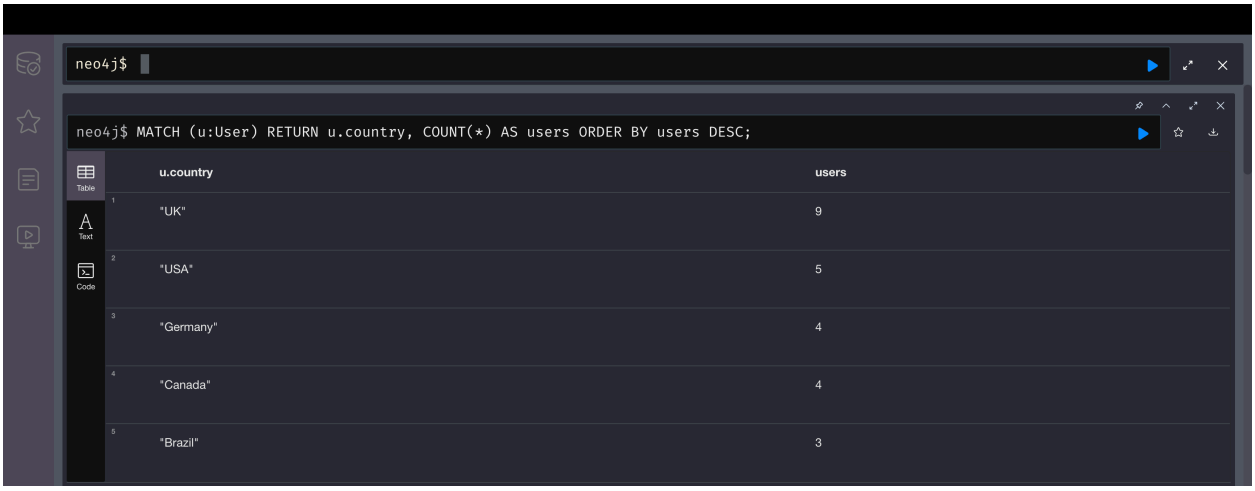


8. Number of users per country

MATCH (u:User)

RETURN u.country, COUNT(*) AS users

ORDER BY users DESC;

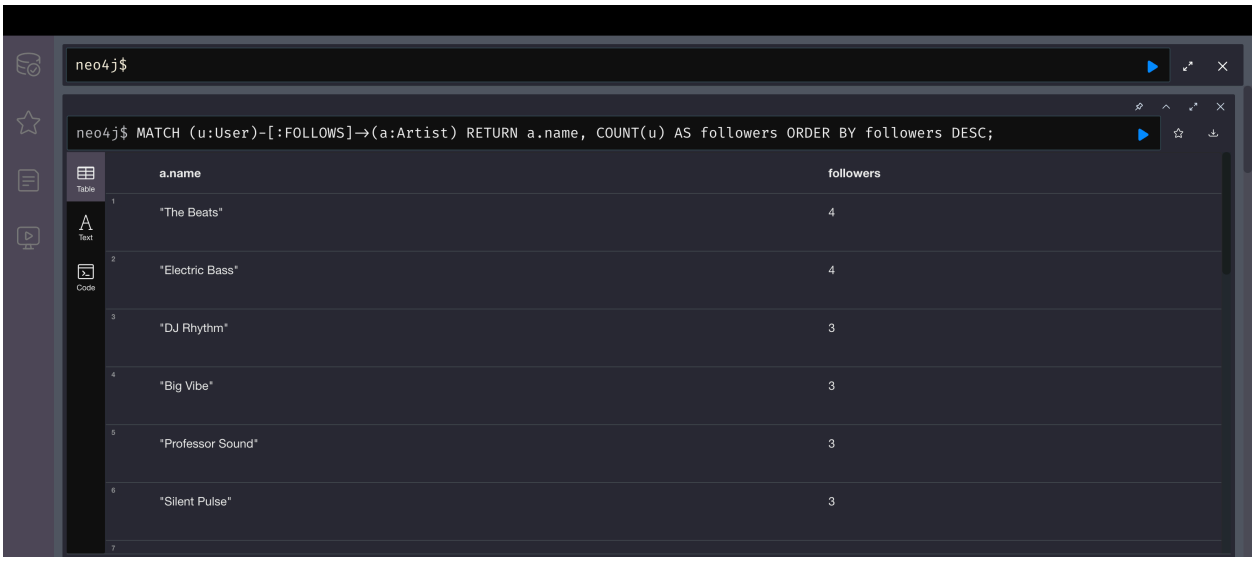


9. How many users follow each artist?

MATCH (u:User)-[:FOLLOWS]->(a:Artist)

RETURN a.name, COUNT(u) AS followers

ORDER BY followers DESC;

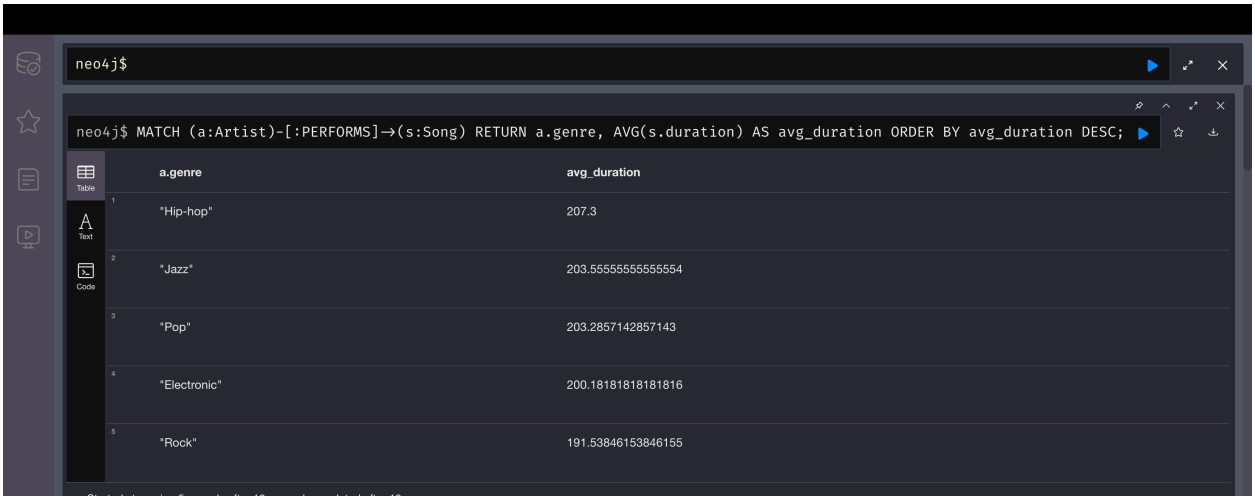


10. Average song duration per genre

MATCH (a:Artist)-[:PERFORMS]->(s:Song)

RETURN a.genre, AVG(s.duration) AS avg_duration

ORDER BY avg_duration DESC;



11. Users who listened to songs longer than 4 minutes

MATCH (u:User)-[:LISTENED_TO]->(s:Song)

WHERE s.duration > 240

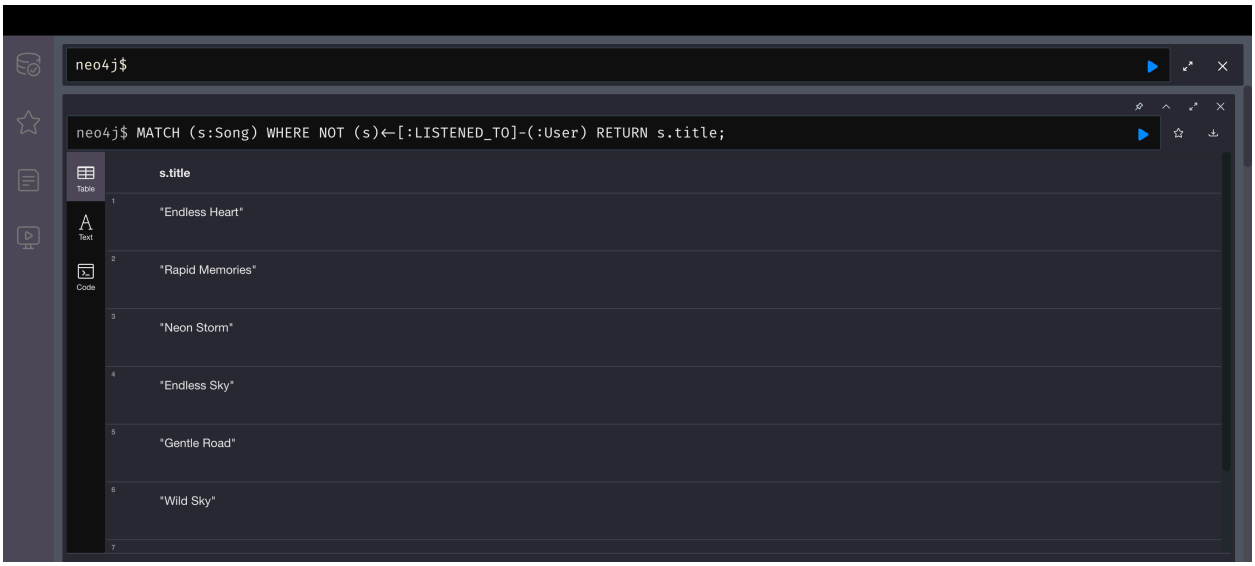
RETURN DISTINCT u.name, s.title;

12. Songs that no one has listened to

MATCH (s:Song)

WHERE NOT (s)<-[:LISTENED_TO]-(:User)

RETURN s.title;



13. Artists who performed more than 2 songs

MATCH (a:Artist)-[:PERFORMS]->(s:Song)

WITH a, COUNT(s) AS songs

WHERE songs > 2

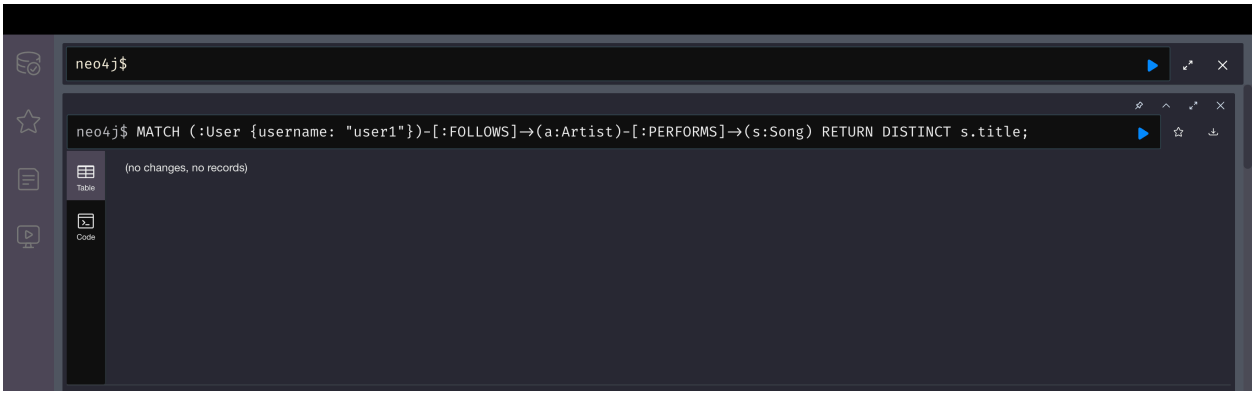
RETURN a.name, songs

ORDER BY songs DESC;

14. Songs by artists followed by user1

MATCH (:User {username: "user1"})-[:FOLLOWS]->(a:Artist)-[:PERFORMS]->(s:Song)

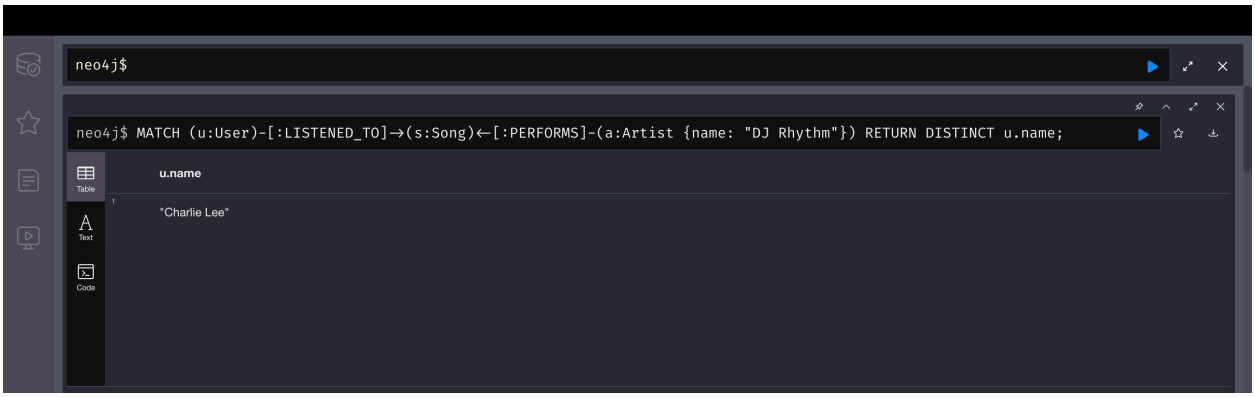
RETURN DISTINCT s.title;



15.Users who listened to songs by 'DJ Rhythm'

MATCH (u:User)-[:LISTENED_TO]->(s:Song)<-[:PERFORMS]-(a:Artist {name: "DJ Rhythm"})

RETURN DISTINCT u.name;

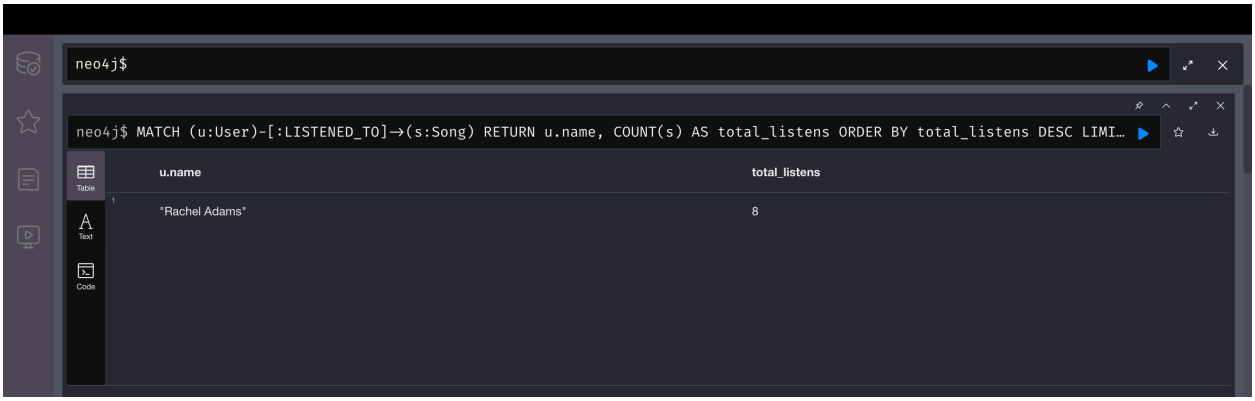


16. Most active listener (by number of listens)

MATCH (u:User)-[:LISTENED_TO]->(s:Song)

RETURN u.name, COUNT(s) AS total_listens

ORDER BY total_listens DESC LIMIT 1;

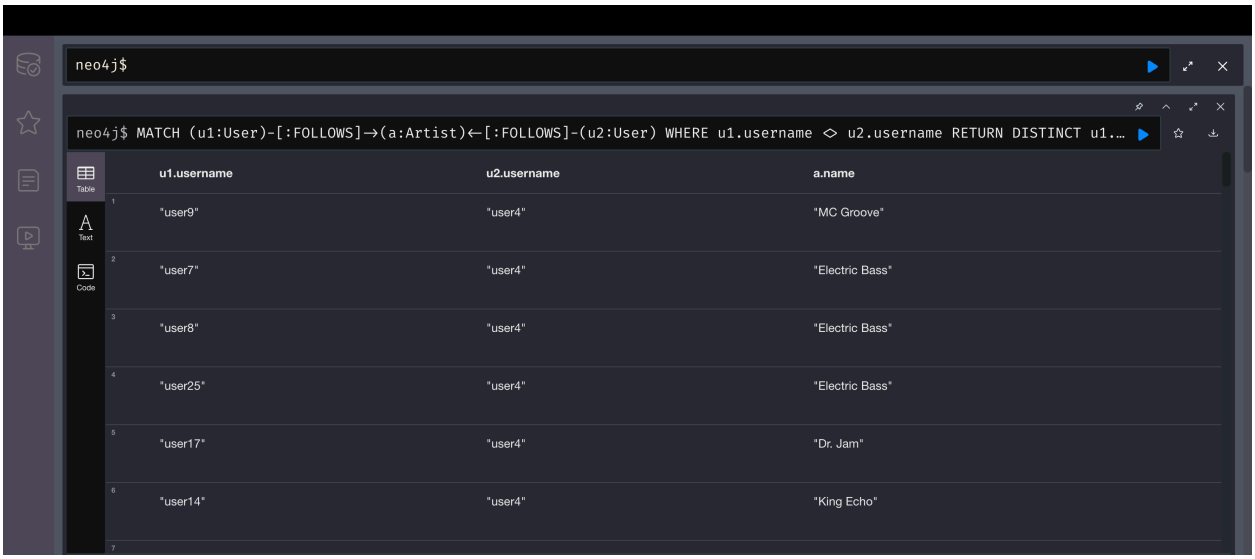


17.Users who follow the same artist

MATCH (u1:User)-[:FOLLOWS]->(a:Artist)<-[:FOLLOWS]-(u2:User)

WHERE u1.username <> u2.username

RETURN DISTINCT u1.username, u2.username, a.name;



18. Number of songs each user listened to

MATCH (u:User)-[:LISTENED_TO]->(s:Song)

RETURN u.username, COUNT(s) AS song_count

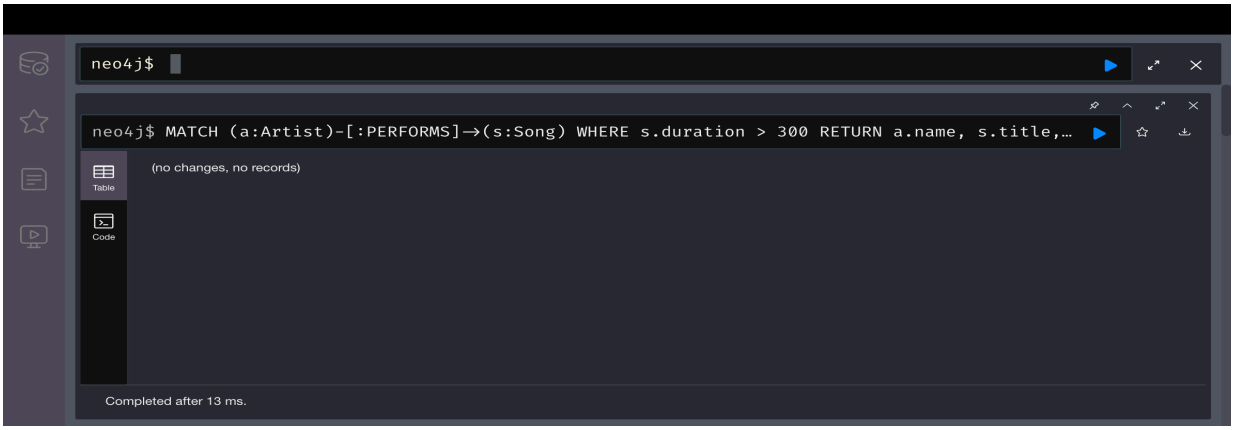
ORDER BY song_count DESC;

19. Artists that perform songs longer than 5 minutes

MATCH (a:Artist)-[:PERFORMS]->(s:Song)

WHERE s.duration > 300

RETURN a.name, s.title, s.duration;



20.Genres of songs listened to by users in Canada

MATCH (u:User {country: "Canada"})-[:LISTENED_TO]->(s:Song)<-[:PERFORMS]-(a:Artist)

RETURN DISTINCT a.genre;