**Graph Analytics**

**Modeling Chat Data using a Graph Data Model**

Chat data (interaction between users of the Flamingo game) is modelled into a network-based graph model.

A user can create, join or leave a team chat session with their team, who owns the chat session. User create a chat item within the chat session to mention or respond to the attendees (other users attending the team chat session).

**Creation of the Graph Database for Chats**

Describe the steps you took for creating the graph database. As part of these steps

1. Schema of the 6 CSV files

|  |  |
| --- | --- |
| **File Name** | **Schema** |
| chat\_create\_team\_chat.csv | userid, teamid, TeamChatSessionID, timestamp |
| chat\_item\_team\_chat.csv | userid, teamchatsessionid, chatitemid, timestamp |
| chat\_join\_team\_chat.csv | userid, TeamChatSessionID, teamstamp |
| chat\_leave\_team\_chat.csv | userid, teamchatsessionid, timestamp |
| chat\_mention\_team\_chat.csv | ChatItem, userid, timeStamp |
| chat\_respond\_team\_chat.csv | chatid1, chatid2,timestamp |

1. Explain the loading process and include a sample LOAD command

Steps :

* Open Neo4j and create or open a database
* Place the file in the import folder of your respective database of Neo4j application
* Load the CSV data files into Neo4J using below query

Example query:

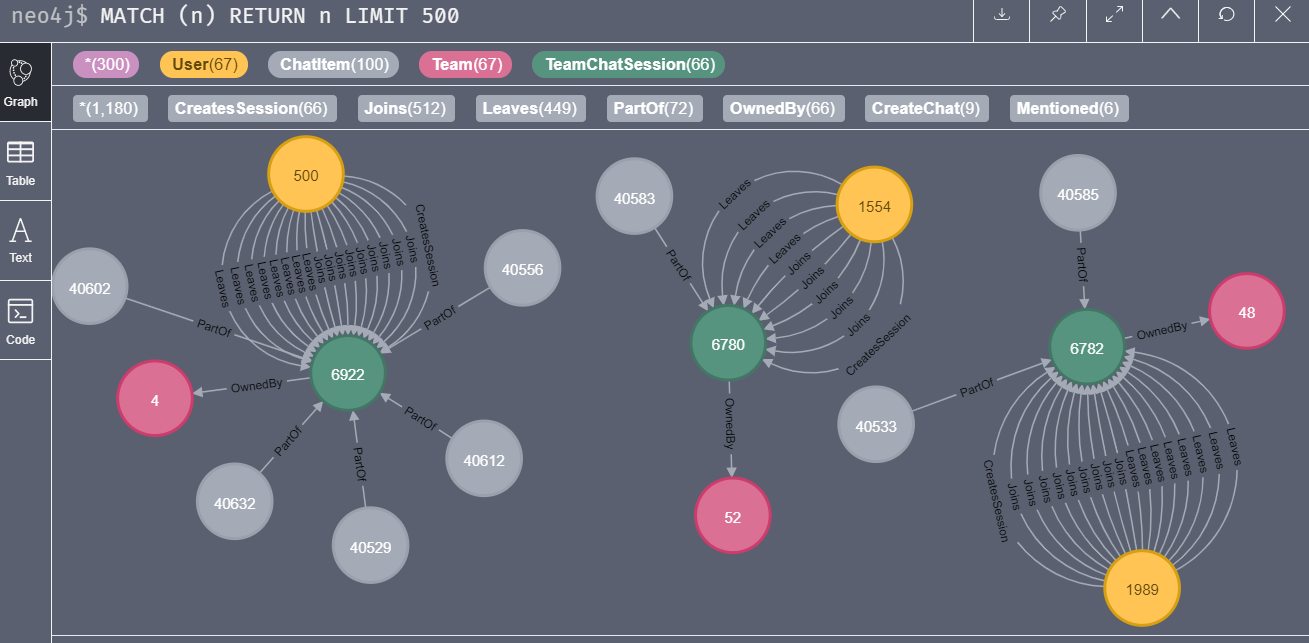
*LOAD CSV FROM "file:///chat\_join\_team\_chat.csv" AS row*

*MERGE (u:User {id: toInteger(row[0])})*

*MERGE (c:TeamChatSession {id: toInteger(row[1])})*

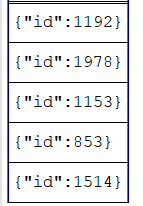
*MERGE (u)-[:Joins{timeStamp: row[2]}]->(c)*

1. Screenshot of graph generated.



**Finding the longest conversation chain and its participants**

* Path length of longest conversation is 9.
* Unique users involved in the conversation are below



* Query:

*match p = (m)-[:ResponseTo\*]->(n)*

*where length(p) = 9*

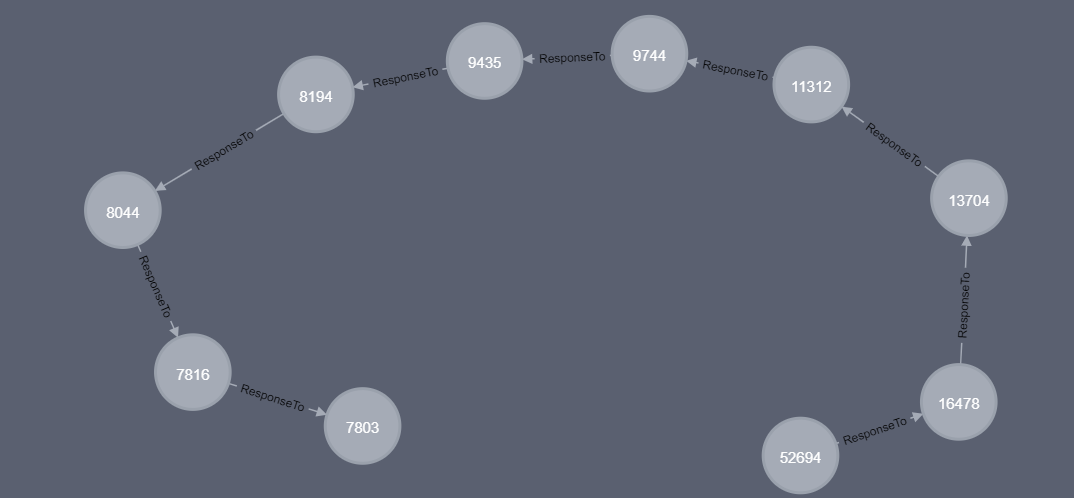
*with p*

*match (i)-[:CreateChat]->(j)*

*where j in nodes(p)*

*return distinct(i)*

* Path Screenshot



**Analyzing the relationship between top 10 chattiest users and top 10 chattiest teams**

* Below queries are run in neo4j to identify chattiest users and teams
* ***Chattiest Users***

*match (m)-[:CreateChat]->(n)*

*return m.id, count(n)*

*order by count(n) desc limit 10*

* ***Chattiest team***

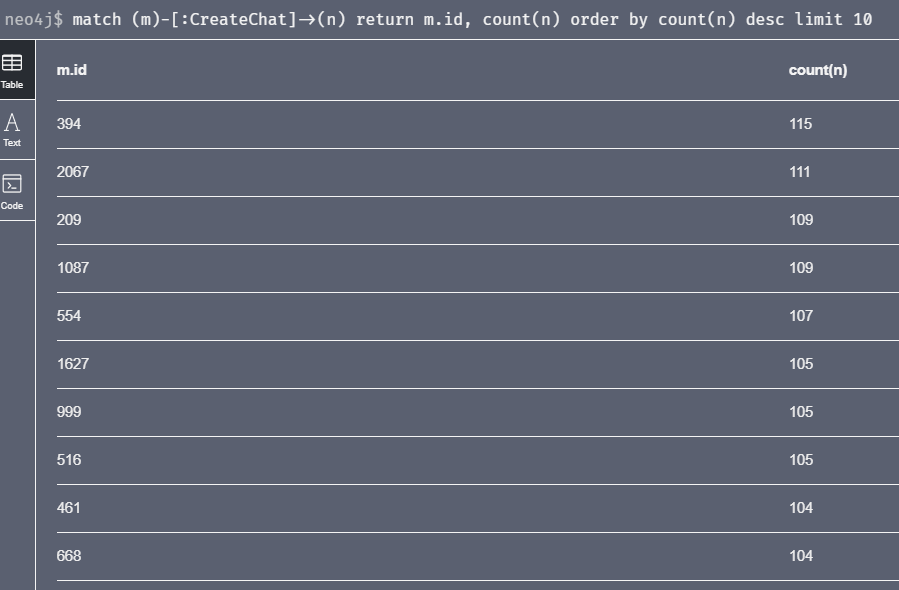
*match (m:ChatItem)-[:PartOf]->(:TeamChatSession)-[:OwnedBy]->(n)*

*return n.id, count(n)*

*order by count(n) desc limit 10*

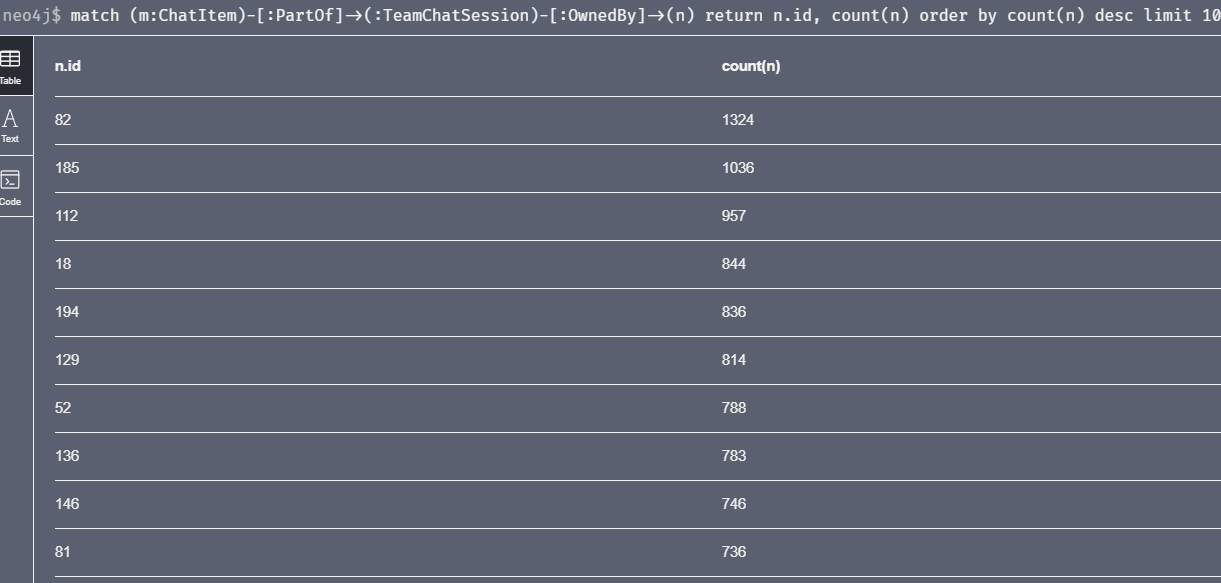
* ***Results are below***

**Chattiest Users**



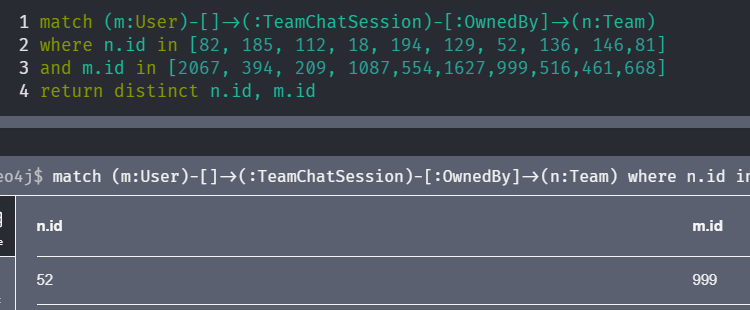
|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 209 | 109 |

**Chattiest Teams**



|  |  |
| --- | --- |
| **Teams** | **Number of Chats** |
| 82 | 1324 |
| 185 | 1036 |
| 112 | 957 |

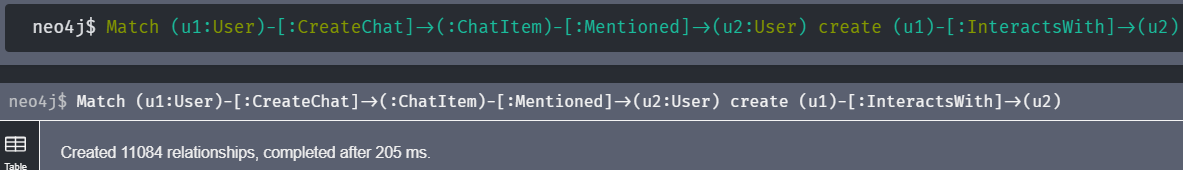
* **Only one chattiest user, 999 is part of the one of the chattiest team, 52**

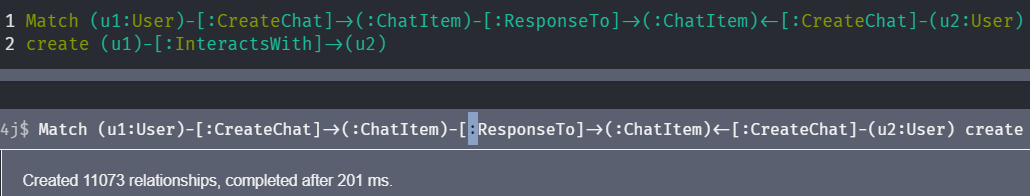


**How Active Are Groups of Users?**

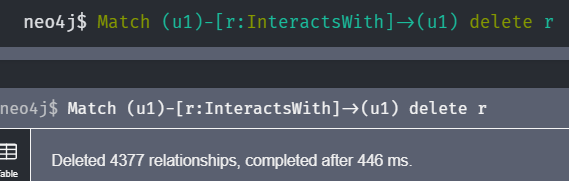
Describe your steps for performing this analysis. Be as clear, concise, and as brief as possible. Finally, report the top 3 most active users in the table below.

**Step 1:** construct the neighbourhood of users, below are the queries executed in neo4j

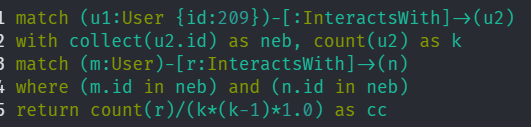




**Step 2:** Remove undesirable self-loop, using below query in neo4j.



**Step 3:** Calculate Cluster coefficient using below query in neo4j:



**Most Active Users (based on Cluster Coefficients)**

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 394 | 0.92 |
| 1087 | 0.8 |
| 209 | 0.95 |