**Graph Analytics**

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

It’s a chat graph. It’s contain user create chat session, user joins a chat session and user leaves a chat session. It’s also contain user chats in a chat session, and the chat may mentions other user or responds to other user.

**Creation of the Graph Database for Chats**

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

1. **Write the schema of the 6 CSV files**

**chat\_create\_team\_chat.csv:** userid, teamid, TeamChatSessionID, timestamp

A line is added to this file when a player creates a new chat with their team.

**chat\_join\_team\_chat.csv:** userid, TeamChatSessionID, timestamp

Creates an edge labeled "Joins" from User to TeamChatSession. The columns are the User id,

TeamChatSession id and the timestamp of the Joins edge.

**chat\_leave\_team\_chat.csv:** userid, TeamChatSessionID, timestamp

Creates an edge labeled "Leaves" from User to TeamChatSession. The columns are the User id,

TeamChatSession id and the timestamp of the Leaves edge.

**chat\_item\_team\_chat.csv:** userid, TeamChatSessionID, chatitemid, timestamp

Creates nodes labeled ChatItems. Column 0 is User id, column 1 is the TeamChatSession id, column 2 is

the ChatItem id (i.e., the id property of the ChatItem node), column 3 is the timestamp for an edge

labeled "CreateChat". Also create an edge labeled "PartOf" from the ChatItem node to the

TeamChatSession node. This edge should also have a timeStamp property using the value from Column

3.

**chat\_mention\_team\_chat.csv:** ChatItem, userid, timeStamp

Creates an edge labeled "Mentioned". Column 0 is the id of the ChatItem, column 1 is the id of the User,

and column 2 is the timeStamp of the edge going from the chatItem to the User.

**chat\_respond\_team\_chat.csv:** chatid1, chatid2,timestamp

A line is added to this file when player with chatid2 responds to a chat post by another player with

chatid1.

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

The first line gives the path of the file.

Then create nodes and attributes through MERGE.

Finally, create edges (source node, destination node, and relative attribute) through MERGE.

LOAD CSV FROM "file:/Users/hahadsg/Downloads/tmp/z/big\_data\_capstone\_datasets\_and\_scripts/chat-data/chat\_create\_team\_chat.csv" AS row

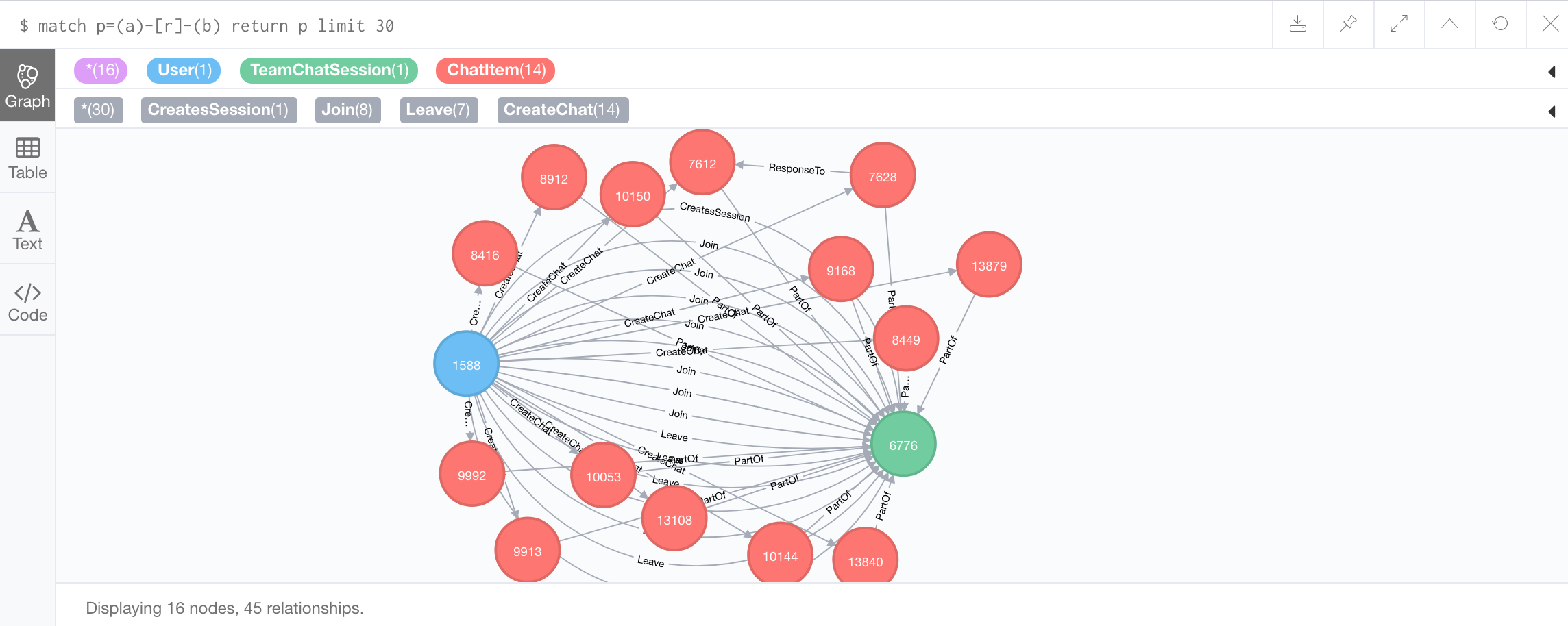
MERGE (u:User {id: toInt(row[0])}) MERGE (t:Team {id: toInt(row[1])})

MERGE (c:TeamChatSession {id: toInt(row[2])})

MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c)

MERGE (c)-[:OwnedBy{timeStamp: row[3]}]->(t)

1. **Present a screenshot of some part of the graph you have generated. The graphs must include clearly visible examples of most node and edge types. Below are two acceptable examples. The first example is a rendered in the default Neo4j distribution, the second has had some nodes moved to expose the edges more clearly. Both include examples of most node and edge types.**



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

Report the results including the length of the conversation (path length) and how many unique users were part of the conversation chain. Describe your steps. Write the query that produces the correct answer.

Step1. Match all path using ResponseTo edge label

Step2. Get path\_length

Step3. Order by path\_length desc

Step4. Return and limit 1 show longest path

match p=(a)-[r:ResponseTo\*]->(b)

with p, length(p) as plen

order by plen desc

return p, plen

limit 1;

match p=(a)-[r:ResponseTo\*]->(b)

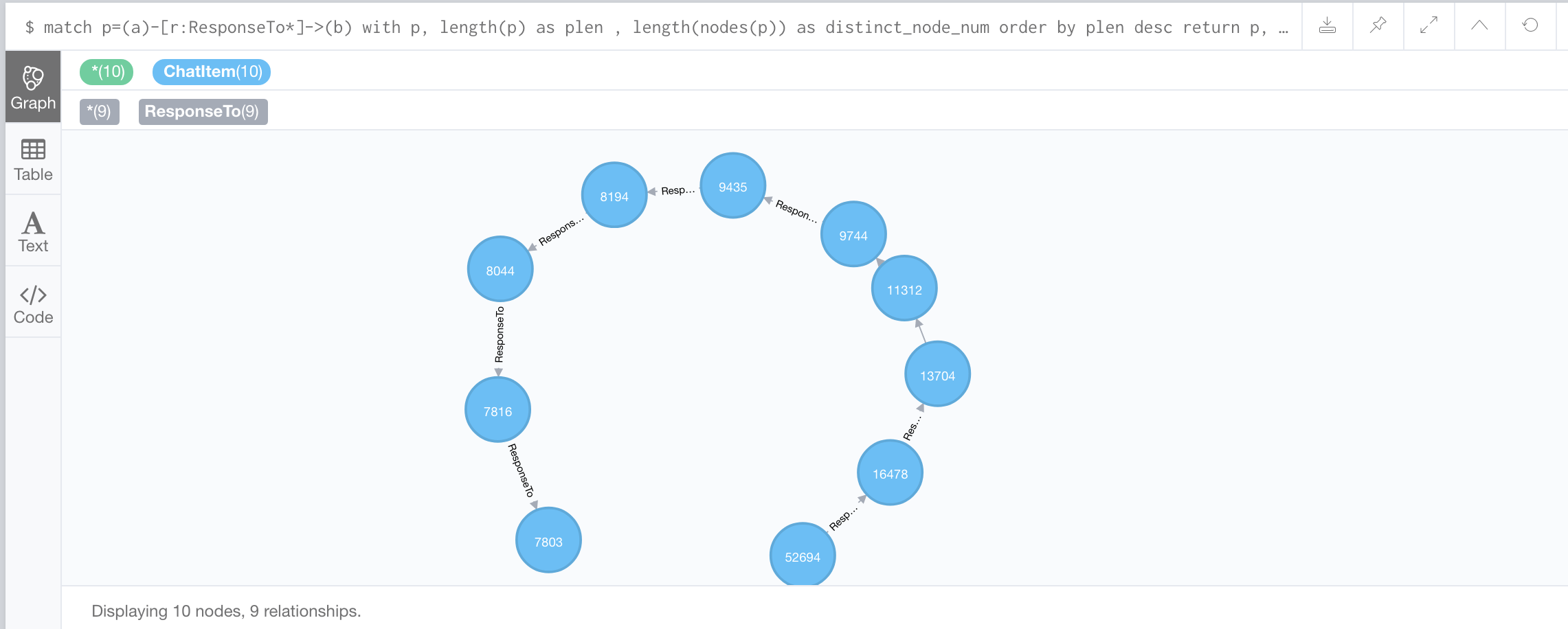
where length(p) = 9

with p

match (u:User)-[:CreateChat]->(i:ChatItem)

where i in nodes(p)

return count(distinct u);

****

The length of longest conversation is 9.

And 5 unique users were part of the longest conversation chain

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

Describe your steps from Question 2. In the process, create the following two tables. You only need to include the top 3 for each table. Identify and report whether any of the chattiest users were part of any of the chattiest teams.

**Chattiest Users**

|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 1087 | 109 |

Step1: Match all path using User node label and CreateChat edge label

Step2: count all user chat\_num

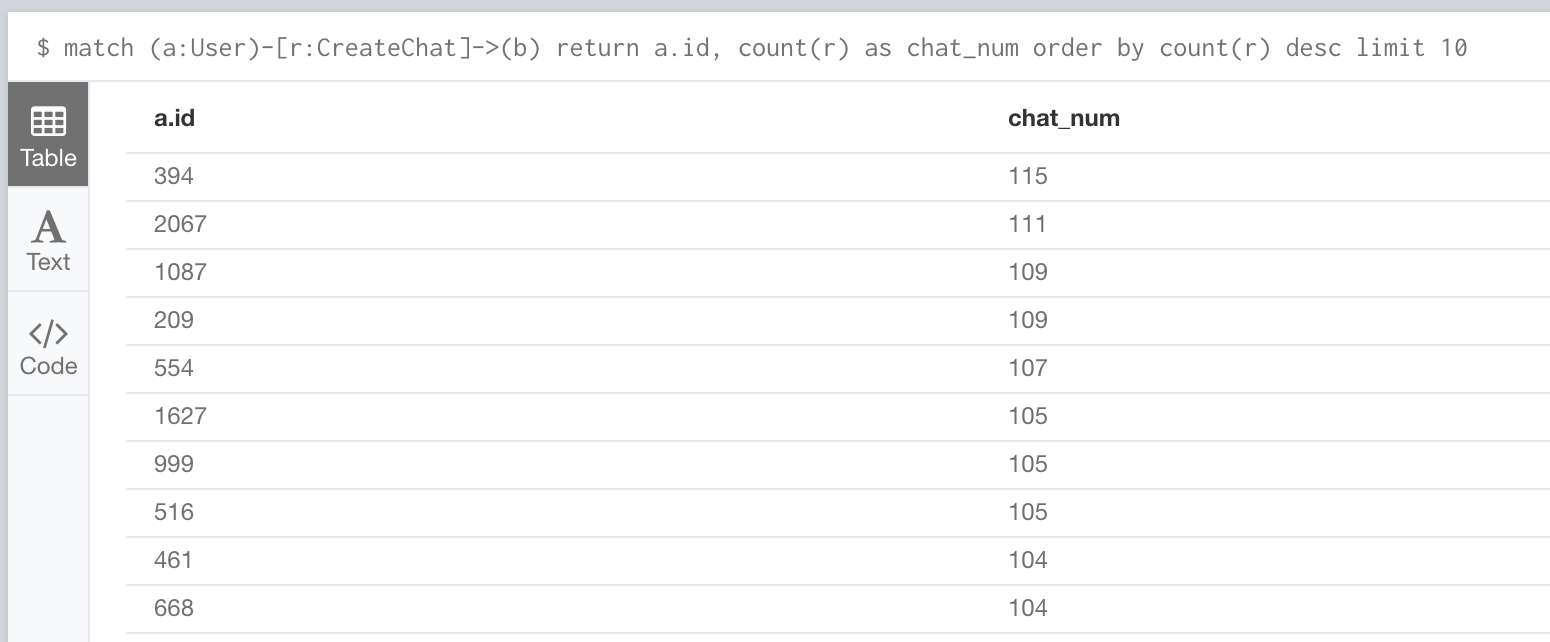
Step3: order by chat\_num desc and return

match (a:User)-[r:CreateChat]->(b)

return a.id, count(r) as chat\_num

order by count(r) desc

limit 10



**Chattiest Teams**

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

Step1. Match all path that satisfy (i:ChatItem)-[r1:PartOf]->(c:TeamChatSession)-[r2:OwnedBy]->(t)

Step2. Count all team chat\_num

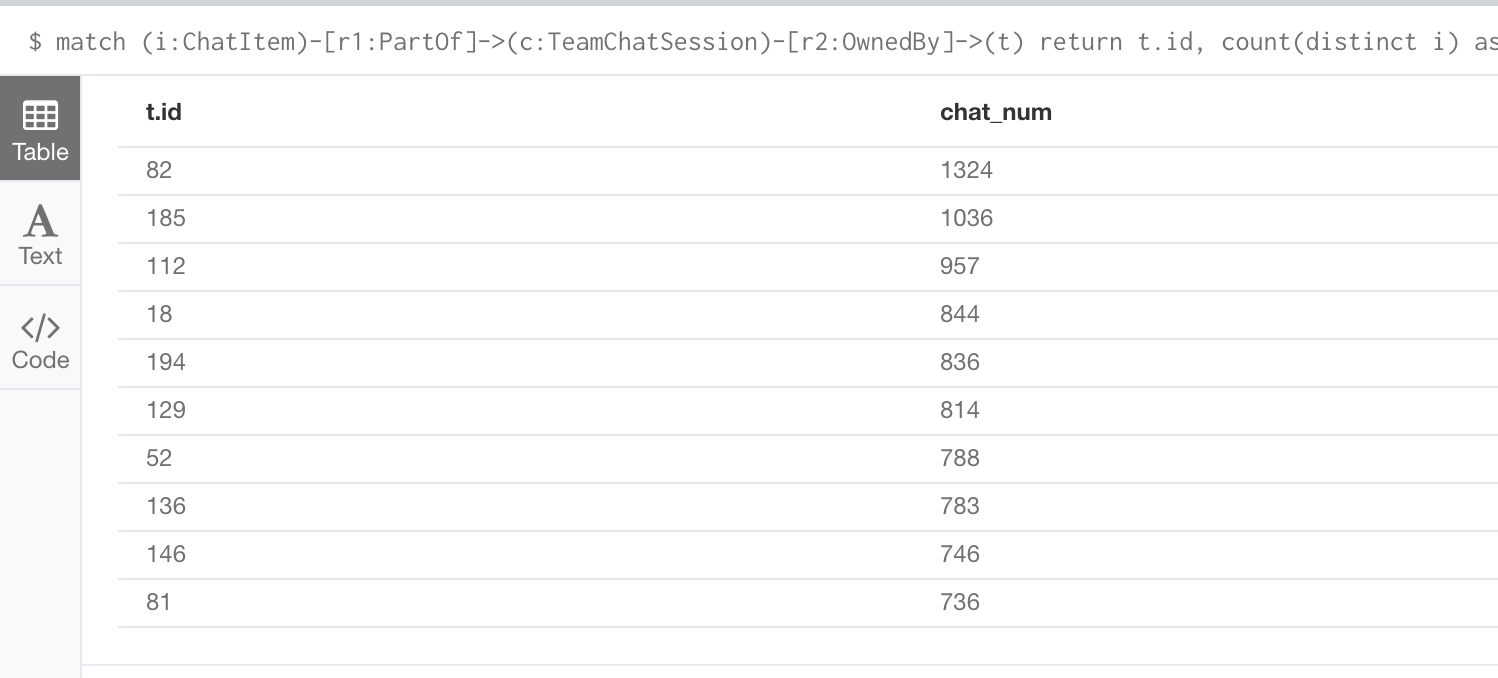
Step3. Order by chat\_num and return

match (i:ChatItem)-[r1:PartOf]->(c:TeamChatSession)-[r2:OwnedBy]->(t)

return t.id, count(distinct i) as chat\_num

order by count(distinct i) desc

limit 10



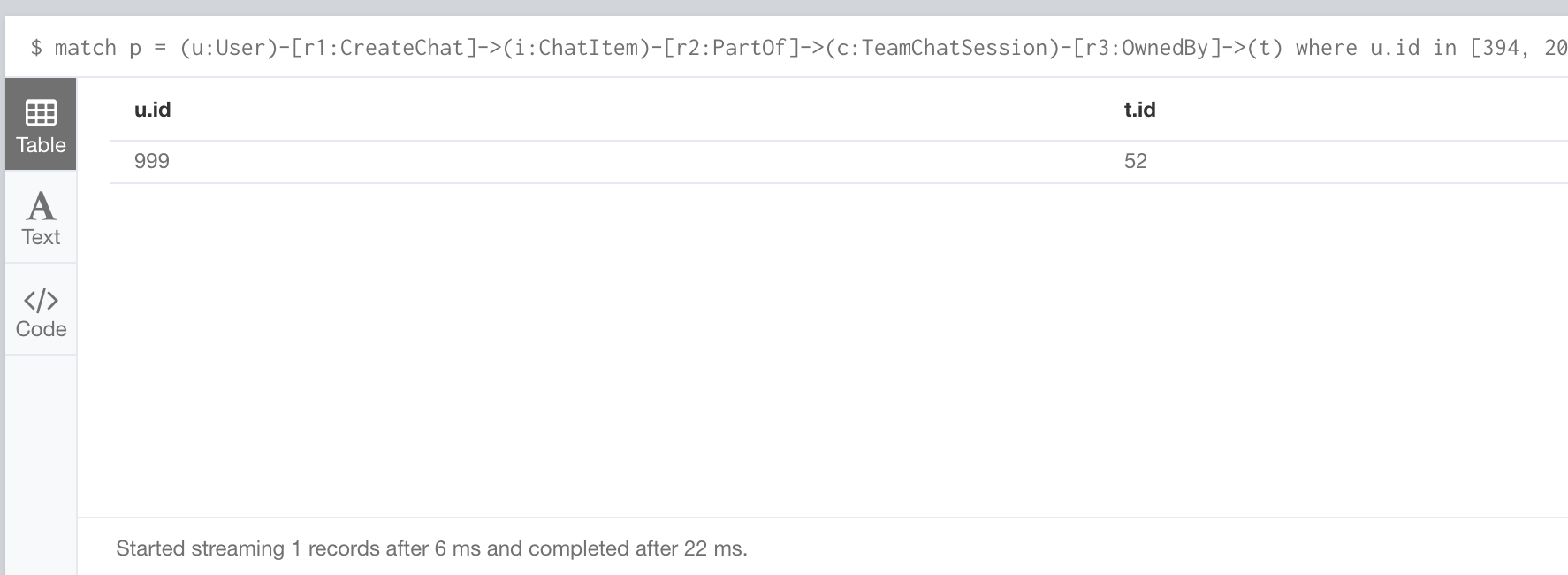
Finally, present your answer, i.e. whether or not any of the chattiest users are part of any of the chattiest teams.

**The User 999 in Team 52.**

match p = (u:User)-[r1:CreateChat]->(i:ChatItem)-[r2:PartOf]->(c:TeamChatSession)-[r3:OwnedBy]->(t)

where u.id in [394, 2067, 209, 1087, 554, 1627, 999, 516, 461, 668]

and t.id in [82, 185, 112, 18, 194, 129, 52, 136, 146, 81]

return distinct u.id, t.id

**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.

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

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 209 | 0.9523809523809523 |
| 554 | 0.9047619047619048 |
| 1087 | 0.8 |

**Step1**. We will construct the neighborhood of users. In this neighborhood, we will connect two users if

· One mentioned another user in a chat

· One created a chatItem in response to another user’s chatItem

The way to make this connection will be to create a new edge called, “InteractsWith”

between users to satisfy either of the two conditions.

Query: match p = (u1:User)-[:CreateChat]->(i:ChatItem)-[:Mentioned]->(u2:User)

create (u1)-[:InteractsWith]->(u2);

match p = (u1:User)-[:CreateChat]->(i1:ChatItem)-[:ResponseTo]->(i2:ChatItem)<-[:CreateChat]-(u2:User)

create (u1)-[:InteractsWith]->(u2);

**Step2.** The above scheme will create an undesirable side effect if a user has responded to her own

chatItem, because it will create a self loop between two users. So after the first two steps we need to

eliminate all self loops involving the edge “Interacts with”. This will take the form:

match (u1)-[r:InteractsWith]->(u1) delete r

**Step3.** Calcute clustering coefficients. Query:

match (u1:User)-[:InteractsWith]-(u2:User)

where u1.id in [394, 2067, 209, 1087, 554, 1627, 999, 516, 461, 668]

with u1, collect(distinct u2.id) as neighbors

match (a), (b)

where a.id in neighbors and b.id in neighbors

with u1, length(neighbors) \* (length(neighbors) - 1) as total\_num

, sum(case when (a)-[:InteractsWith]->(b) then 1 else 0 end) as link\_num

with u1, link\_num \* 1.0 / total\_num as cluster\_coef, link\_num, total\_num

order by cluster\_coef desc

return u1.id, cluster\_coef, link\_num, total\_num