Graph Analytics

Coursera Big Data Specialization Capstone Project, Week 4

Modeling Chat Data using a Graph Data Model

The graph model is a network based on chat interactions between users. A chat session can be initiated by a user, other users on the same team are able to join and leave the session. Interactions between users begins when a user create a post. It's possible for a user, mention another user. All relationship between entities are logged with a timestamp.

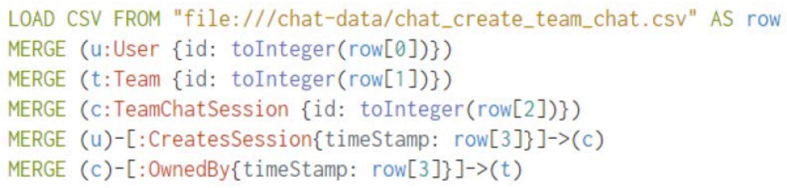
Creation of the Graph Database for Chats

Describe the steps you took for creating the graph database.

Write the schema of the 6 CSV files

chat_create_team_chat.csv	userID teamID teamChatSessionID timestamp
chat_join_team_chat.csv	userID teamChatSessionID timestamp
chat_leave_team_chat.csv	userID teamChatSessionID timestamp
chat_item_team_chat.csv	userID teamChatSessionID chatItemID timestamp
chat_mention_team_chat.csv	chatItemID userID timestamp
chat_respons_team_chat.csv	chatItemID_1 chatItemID_2 timestamp

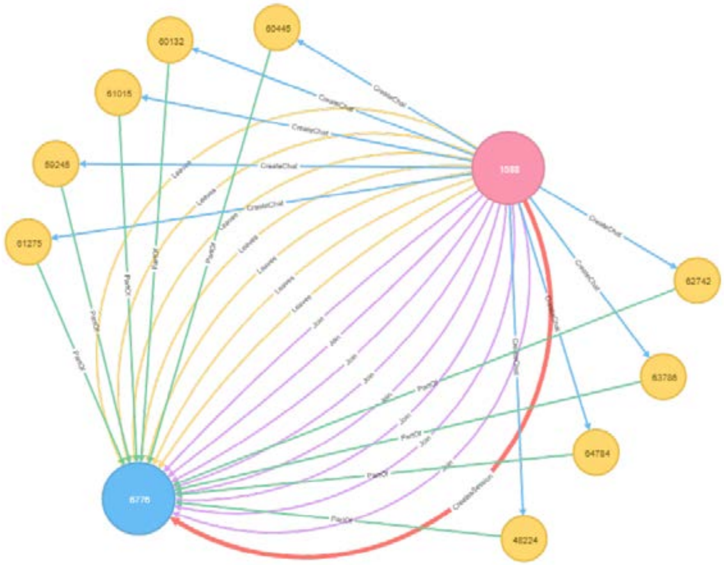
Explain the loading process and include a sample LOAD command



```
LOAD CSV FROM "file:///chat-data/chat_create_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])})
MERGE (t:Team {id: toInteger(row[1])})
MERGE (c:TeamChatSession {id: toInteger(row[2])})
MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c)
MERGE (c)-[:OwnedBy{timeStamp: row[3]}]->(t)
```

The first line load the csv from the specific location one row at a time. From the second line to fourth, create the nodes for User, Team, TeamChatSession with a specific column converted to integer, this field is used by the id attribute. The fifth and sixth lines create CreatesSession and OwnedBy edges and link the nodes previously created. The edges have a timestamp property filled by the fourth column of schema.

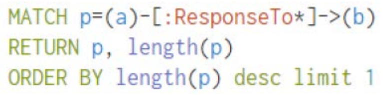
Present a screenshot of some part of the graph you have generated. The graphs must include clearly visible examples of most node and edge types.



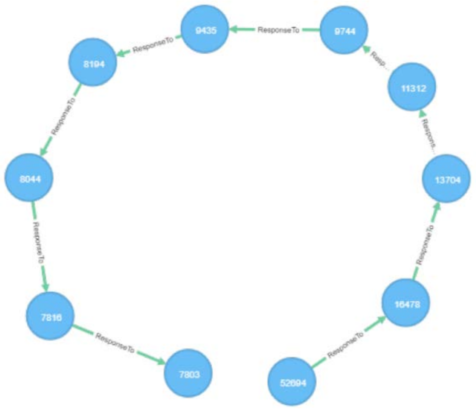
Finding the longest conversation chain and its participants

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.

How many cats are involved in it?

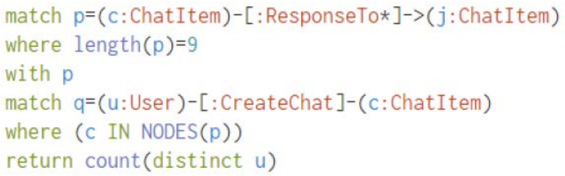


```
MATCH p=(a)-[:ResponseTo*]->(b)
RETURN p, length(p)
ORDER BY length(p) desc limit 1
```



The longest conversation chain in the chat data has path length 9, therefore 10 chats are involved in it.

How many users participated in this chain?



```
match p=(c:ChatItem)-[:ResponseTo*]->(j:ChatItem)
where length(p)=9
with p
match q=(u:User)-[:CreateChat]-(c:ChatItem)
where (c IN NODES(p))
return count(distinct u)
```

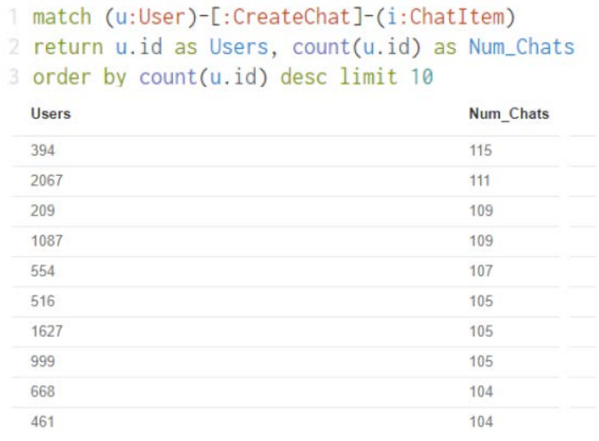
With 9 as longest path, count the number of distinct users who create ChatItem in this longest path. The query returns 5.

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

Determine the number of chats created by a user from the CreateChat edge



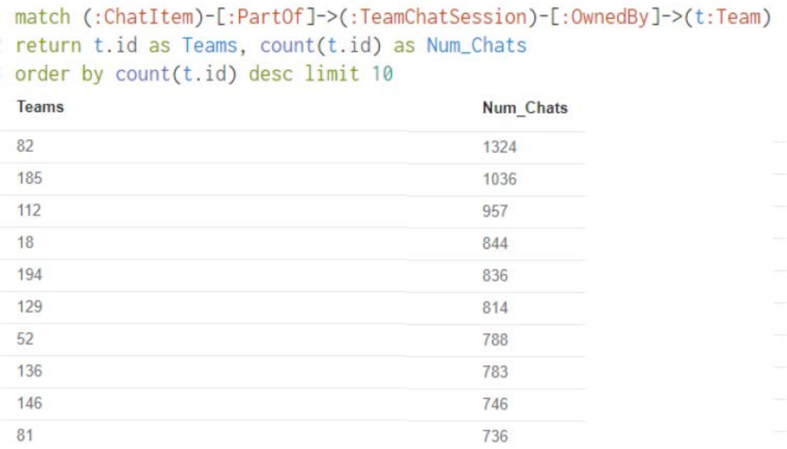
```
match (u:User)-[:CreateChat]-(i:ChatItem)
return u.id as Users, count(u.id) as Num_Chats
order by count(u.id) desc limit 10
```

Users	Num_Chats
394	115
2067	111
209	109
1087	109
554	107
516	105
1627	105
999	105
668	104
461	104

Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams

Match all ChatItem with a PartOd edge and connect them with a TeamChatSession node that have an OwnedBy edge connection them with any other node.

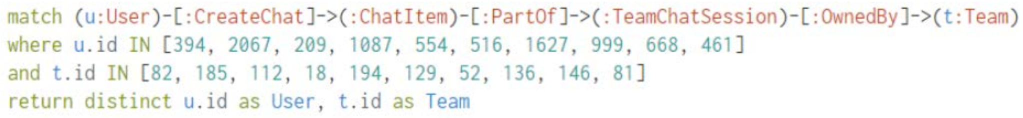


```
match (:ChatItem)-[:PartOf]->(:TeamChatSession)-[:OwnedBy]->(t:Team)
return t.id as Teams, count(t.id) as Num_Chats
order by count(t.id) desc limit 10
```

Teams	Num_Chats
82	1324
185	1036
112	957
18	844
194	836
129	814
52	788
136	783
146	746
81	736

Teams	Number of Chats
82	1324
185	1036
112	957

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



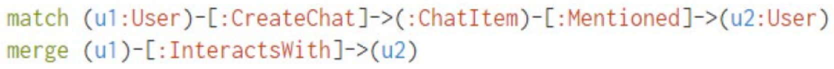
```
match (u:User)-[:CreateChat]->(:ChatItem)-[:PartOf]->(:TeamChatSession)-[:OwnedBy]->(t:Team)
where u.id IN [394, 2067, 209, 1087, 554, 516, 1627, 999, 668, 461]
and t.id IN [82, 185, 112, 18, 194, 129, 52, 136, 146, 81]
return distinct u.id as User, t.id as Team
```

This query is used to investigate if the most chattiest user are part of any chattiest team and it return one result, userID 999 is part of teamID 52.

How Active Are Groups of Users?

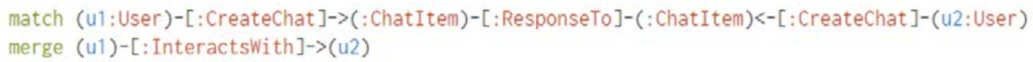
Describe your steps for performing this analysis. Be as clear, concise, and as brief as possible. Finally, report the top 3 most active users in the table below.

Connect mentioned users



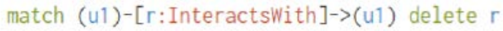
```
match (u1:User)-[:CreateChat]->(:ChatItem)-[:Mentioned]->(u2:User)
merge (u1)-[:InteractsWith]->(u2)
```

Connect users responses with the chat creator



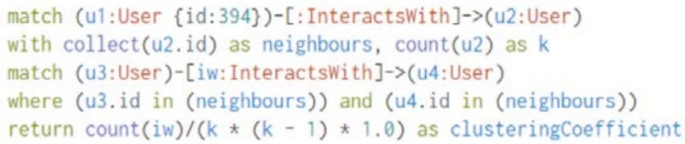
```
match (u1:User)-[:CreateChat]->(:ChatItem)-[:ResponseTo]-(:ChatItem)<-[:CreateChat]-(u2:User)
merge (u1)-[:InteractsWith]->(u2)
```

Eliminate all self interaction



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

<u>Calclulate the cluster coefficient.</u>



```
match (u1:User {id:394})-[:InteractsWith]->(u2:User)
with collect(u2.id) as neighbours, count(u2) as k
match (u3:User)-[iw:InteractsWith]->(u4:User)
where (u3.id in (neighbours)) and (u4.id in (neighbours))
return count(iw)/(k * (k - 1) * 1.0) as clusteringCoefficient
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	0.9167
2067	0.7679
209	0.9524