Data Exploration

Data Set Overview

The table below lists each of the files available for analysis with a short description of what is found in each one.

File Name	Description	Fields
ad-clicks.csv	A line is added to this file when a player clicks on an advertisement in the Flamingo app.	timestamp: when the click occurred. txId: a unique id (within ad-clicks.log) for the click userSessionid: the id of the user session for the user who made the click teamid: the current team id of the user who made the click userid: the user id of the user who made the click adId: the id of the ad clicked on adCategory: the category/type of ad clicked on
buy-clicks.csv	A line is added to this file when a player makes an in-app purchase in the Flamingo app.	timestamp: when the purchase was made. txId: a unique id (within buy-clicks.log) for the purchase userSessionId: the id of the user session for the user who made the purchase

		team: the current team id of the user who made the purchase userId: the user id of the user who made the purchase buyId: the id of the item purchased price: the price of the item purchased
users.csv	This file contains a line for each user playing the game.	timestamp: when user first played the game. userId: the user id assigned to the user. nick: the nickname chosen by the user. twitter: the twitter handle of the user. dob: the date of birth of the user. country: the two-letter country code where the user lives.
team.csv	This file contains a line for each team terminated in the game.	teamId: the id of the team name: the name of the team teamCreationTime: the timestamp when the team was created teamEndTime: the timestamp when the last member left the team

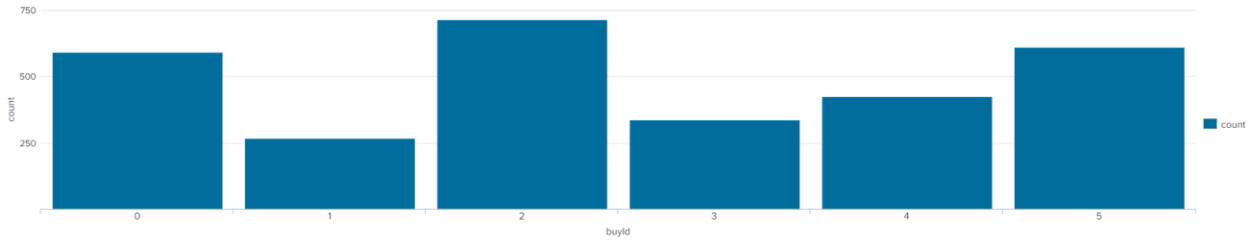
		strength: a measure of team strength, roughly corresponding to the success of a team currentLevel: the current level of the team
team-assignments.csv	A line is added to this file each time a user joins a team. A user can be in at most a single team at a time.	timestamp: when the user joined the team. team: the id of the team userId: the id of the user assignmentId: a unique id for this assignment
level-events.csv	A line is added to this file each time a team starts or finishes a level in the game	timestamp: when the event occurred. eventId: a unique id for the event teamId: the id of the team teamLevel: the level started or completed eventType: the type of event, either start or end
user-session.csv	Each line in this file describes a user session, which denotes when a user starts and stops playing the game. Additionally, when a team goes to the next level in the game, the session is ended for each user in the team and a new one started.	timestamp: a timestamp denoting when the event occurred. userSessionId: a unique id for the session.

		userId: the current user's ID. teamId: the current user's team. assignmentId: the team assignment id for the user to the team. sessionType: whether the event is the start or end of a session. teamLevel: the level of the team during this session. platformType: the type of platform of the user during this session.
game-clicks.csv	A line is added to this file each time a user performs a click in the game.	timestamp: when the click occurred. clickId: a unique id for the click. userId: the id of the user performing the click. userSessionId: the id of the session of the user when the click is performed. isHit: denotes if the click was on a flamingo (value is 1) or missed the flamingo (value is 0) teamId: the id of the team of the user teamLevel: the current level of the team of the user

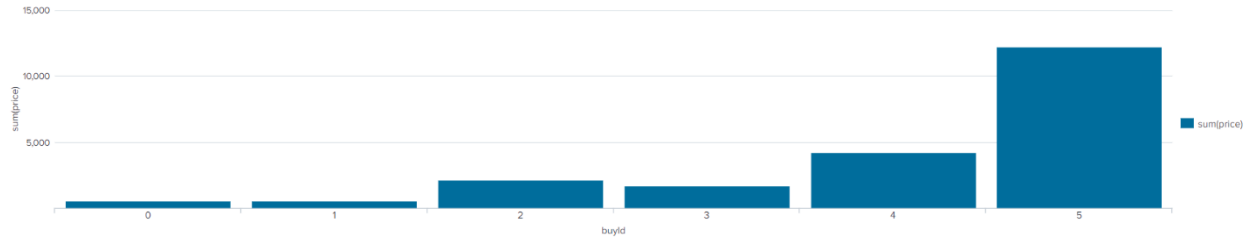
Aggregation

Amount spent buying items	21407.0
Number of unique items available to be purchased	6

A histogram showing how many times each item is purchased:

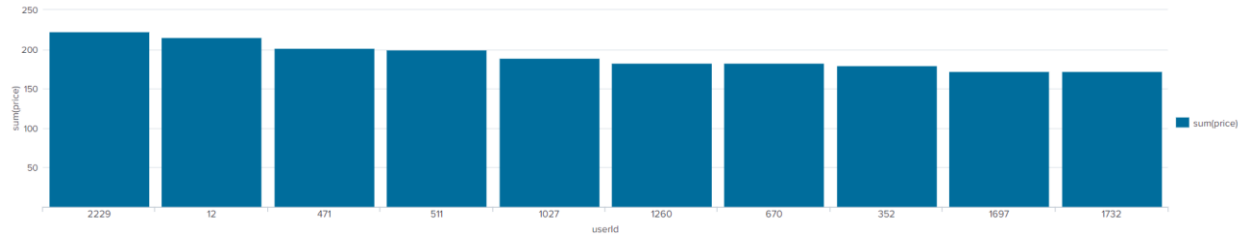


A histogram showing how much money was made from each item:



Filtering

A histogram showing total amount of money spent by the top ten users (ranked by how much money they spent).



The following table shows the user id, platform, and hit-ratio percentage for the top three buying users:

Rank	**User Id**	**Platform**	**Hit-Ratio (%)**

1	2229	Iphone	11.59
2	12	Iphone	13.06
3	471	Iphone	14.5

Data Preparation

Analysis of combined_data.csv

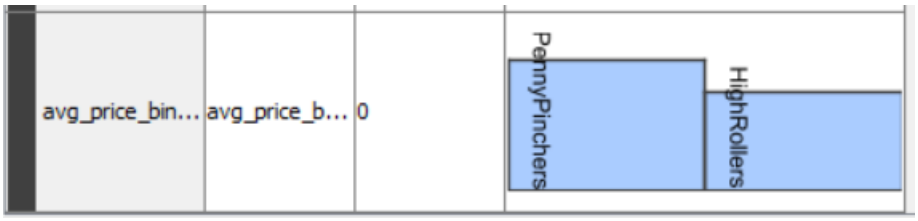
Sample Selection

Item	**Amount**
# of Samples	4619
# of Samples with Purchases	1411

Attribute Creation

A new categorical attribute was created to enable analysis of players as broken into 2 categories (HighRollers and PennyPinchers). A screenshot of the attribute follows:

Row ID	userId	userSe...	teamLevel	platfor...	count_...	count_...	count_...	avg_price	avg_pri...
Row4	937	5652	1	android	39	0	1	1	PennyPinchers
Row11	1623	5659	1	iphone	129	9	1	10	HighRollers
Row13	83	5661	1	android	102	14	1	5	PennyPinchers
Row17	121	5665	1	android	39	4	1	3	PennyPinchers
Row18	462	5666	1	android	90	10	1	3	PennyPinchers
Row31	819	5679	1	iphone	51	8	1	20	HighRollers
Row49	2199	5697	1	android	51	6	2	2.5	PennyPinchers
Row50	1143	5698	1	android	47	5	2	2	PennyPinchers
Row58	1652	5706	1	android	46	7	1	1	PennyPinchers
Row61	2222	5709	1	iphone	41	6	1	20	HighRollers
Row68	374	5716	1	android	47	7	1	3	PennyPinchers
Row72	1535	5720	1	iphone	76	7	1	20	HighRollers
Row73	21	5721	1	android	52	2	1	3	PennyPinchers
Row101	2379	5749	1	android	62	9	1	3	PennyPinchers
Row122	1807	5770	1	iphone	177	25	2	7.5	HighRollers
Row127	868	5775	1	iphone	54	5	1	10	HighRollers
Row129	1567	5777	1	android	27	4	2	4	PennyPinchers
Row131	221	5779	1	iphone	37	2	1	20	HighRollers
Row135	2306	5783	1	android	67	5	1	1	PennyPinchers
Row137	1065	5785	1	iphone	37	5	2	11.5	HighRollers
Row140	827	5788	1	iphone	75	5	1	20	HighRollers
Row150	1304	5798	1	mac	71	9	2	11.5	HighRollers
Row158	1264	5806	1	linux	81	12	1	5	PennyPinchers
Row159	1026	5807	1	iphone	52	10	1	20	HighRollers



HighRollers (buyers of items that cost more than $5.00) and PennyPinchers (buyers of items that cost $5.00 or less) were categorized as mentioned.

The creation of this new categorical attribute was necessary because this is what we need to predict for the users that whether they are likely to purchase big-ticket items while playing Catch the Pink Flamingo or not. So this is out target variable.

Attribute Selection

The following attributes were filtered from the dataset for the following reasons:

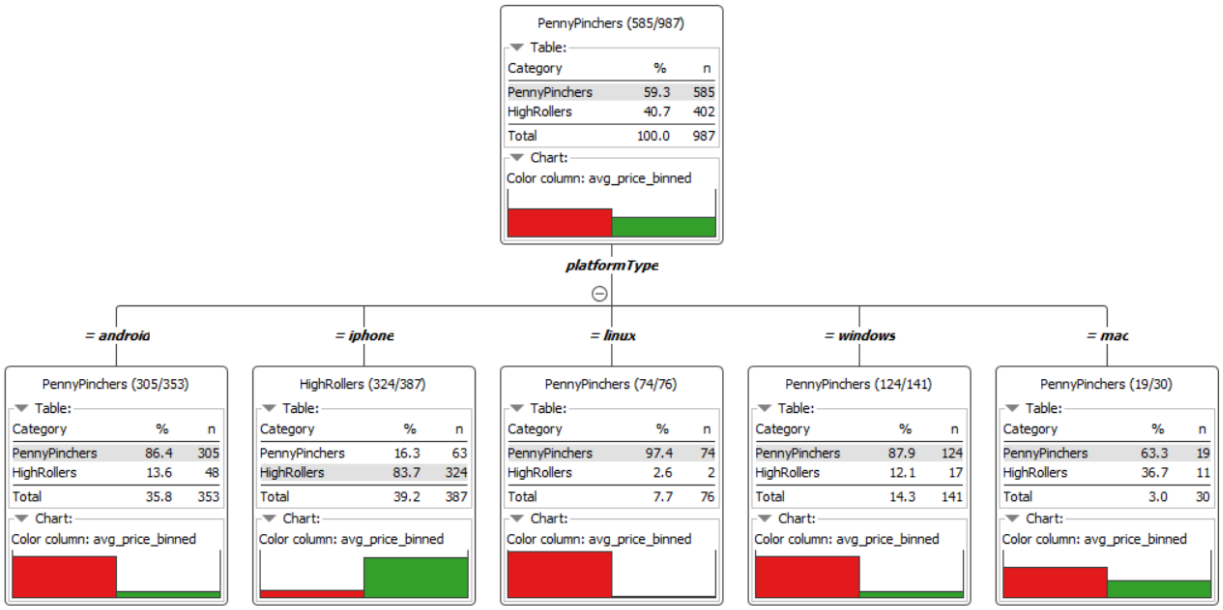
Attribute	**Rationale for Filtering**
UserId	Being an Id, it is just a unique attribute of a user i.e. it doesn't drives the decision of purchasing.
UserSessionId	Being an Id, it is just a unique attribute of a user i.e. it doesn't drives the decision of purchasing.
PlatformType	Platform Type doesn't contribute to the decision making.
countBuyId	Being an Id, it is just a unique attribute of a user i.e. it doesn't drives the decision of purchasing.

Data Partitioning and Modeling

The data was partitioned into train and test datasets.
The 60% data set was used to create the decision tree model.
The trained model was then applied to the 40% test dataset.
This is important to evaluate the learned network to evaluate its performance by testing on the test dataset.

When partitioning the data using sampling, it is important to set the random seed because it is necessary to avoid imbalance between the training and testing set.

A screenshot of the resulting decision tree can be seen below:



Evaluation

A screenshot of the confusion matrix can be seen below:

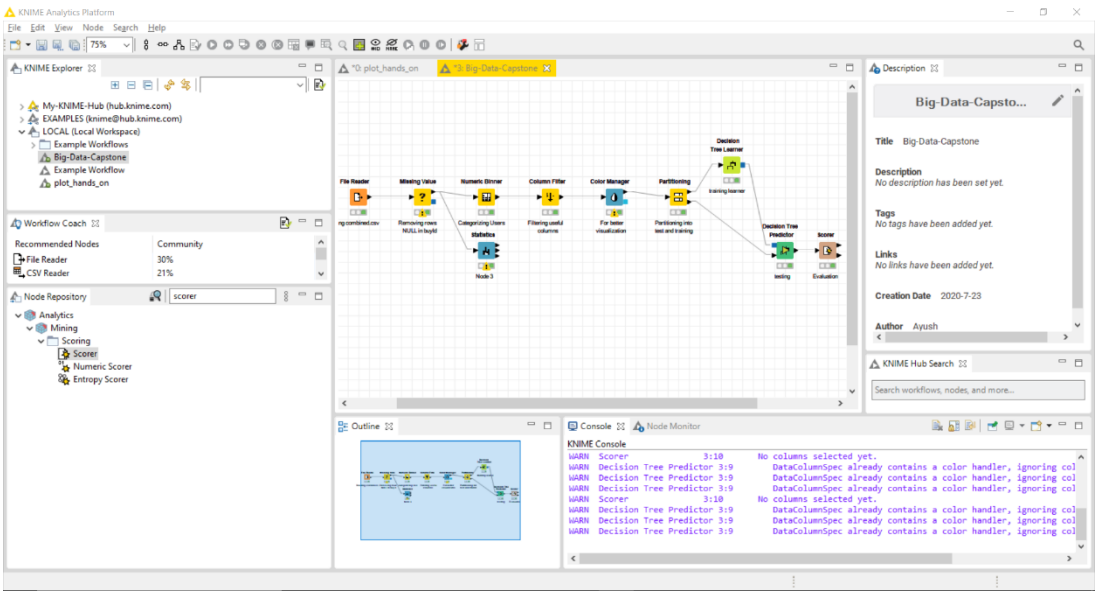
avg_price_...	PennyPinc...	HighRollers
PennyPinchers	308	27
HighRollers	38	192

As seen in the screenshot above, the overall accuracy of the model is 88.496%

It shows that (308+192) values are Correctly predicted, but 38 values were wrongly classified as HighRollers and 27 values are wrongly classified as PennyPinchers.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

Specific Recommendations to Increase Revenue
1. Should focus on users with Iphone as their platform as evident from the decision tree.
2. Mac users also have some potential of being HighRollers

Attribute Selection

features_used = ["totalAdClicks",”totalGameClicks”,”revenue”]

Attribute	Rationale for Selection

totalAdClicks	This could be used in determining the company's policy to determine which users to target
totalGameclicks	Could be a useful feature for grouping the users based on their performances and giving goodies to those users.
revenue	Is a useful feature in terms of determining the revenue of the users and thus targeting those users more.

Training Data Set Creation

The training data set used for this analysis is shown below (first 5 lines):

	totalAdClicks	totalGameClicks	revenue
0	44	716	21.0
1	10	380	53.0
2	37	508	80.0
3	19	3107	11.0
4	46	704	215.0

Dimensions of the final data set: (543,3)
# of clusters created: 3

Cluster Centers

The code used in creating cluster centers is given below:

Km_model = KMeans.train(parsed_data,3,maxIterations = 10,runs = 10, initializationMode = "random")
Centers = km_model.centers

Cluster centers formed are given in the table below

Cluster #	Center(totalAdClicks,totalGameClicks,revenue)
1	36.44134078, 926.11731844, 46.96648045
2	32.35555556, 2310.64444444, 39.42222222
3	24.98746082, 357.95924765, 35.06583072

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that users with the highest revenue and adclick are not the ones who are playing the most but an intermediate result in game clicks.

Cluster 2 is different from the others in that the users who play the less also produces the less ad revenue and click count

Cluster 3 is different from the others in that the users who play the most are not the ones who produce the most revenue, the revenue in the middle along with the ad click count

Below you can see the summary of the train data set:

	totalAdClicks	totalGameClicks	revenue
0	44	716	21.0
1	10	380	53.0
2	37	508	80.0
3	19	3107	11.0
4	46	704	215.0

```
print(km_model.centers)
```

[array([36.44134078, 926.11731844, 46.96648045]), array([24.98746082, 357.95924765, 35.06583072]), array([32.35555556, 2310.64444444, 39.42222222])]

Recommended Actions

Action Recommended	**Rationale for the action**
Increase ads to users who play a lot	It was seen that users who play a lot are also the users who spend less and click less on ads. If we increase ads to users who play a lot, it will promote these users to spend more and therefore increase the revenue
Show higher price ads to users who spend more	If we show higher price ads to users who spend more, we can increase the revenue faster. The users who spend the more also do not play too much, thus by showing them the more valuable ads first, we can increase the revenue faster

Graph Analytics

Modeling Chat Data using a Graph Data Model

We have used a Graph Data Model to illustrate the chatting among users with chat data. A user in a team can create a chat session and then create chat item in the chat session. User can also be mentioned in a chat item and a chat item can also be responded by another chat item. A User can also joins and leaves a chat session.

Creation of the Graph Database for Chats

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

i) Write the schema of the 6 CSV files

FILES	SCHEMA(COLUMNS)
chat_create_team_chat.csv	Userid : The unique ID of the user Teamid : The unique ID of the team TeamChatSessionID : The unique ID of a session Timestamp : denoting when the session was created
chat_item_team_chat.csv	userid, : The unique ID of the user teamchatsessionid : The unique ID of the chat session chatitemid : The unique ID of the chat item timestamp : denoting when the chat item was created
chat_join_team_chat.csv	Userid : The unique ID of the user TeamChatSessionID : The unique ID of the chat session Teamstamp : denoting when a user joined a chat session
chat_leave_team_chat.csv	Userid : The unique ID of the user Teamchatsessionid : The unique ID of the chat session Timestamp : denoting when a user leaves a chat session
chat_mention_team_chat.csv	ChatItem : The unique ID of the chat item Userid : The unique ID of the user timeStamp : denoting when the user was mentioned by the chat item
chat_respond_team_chat.csv	chatid1 : The unique ID of the chat item1 chatid2 : The unique ID of the chat item2 timestamp : denoting when chat item2 was responded to chat item1

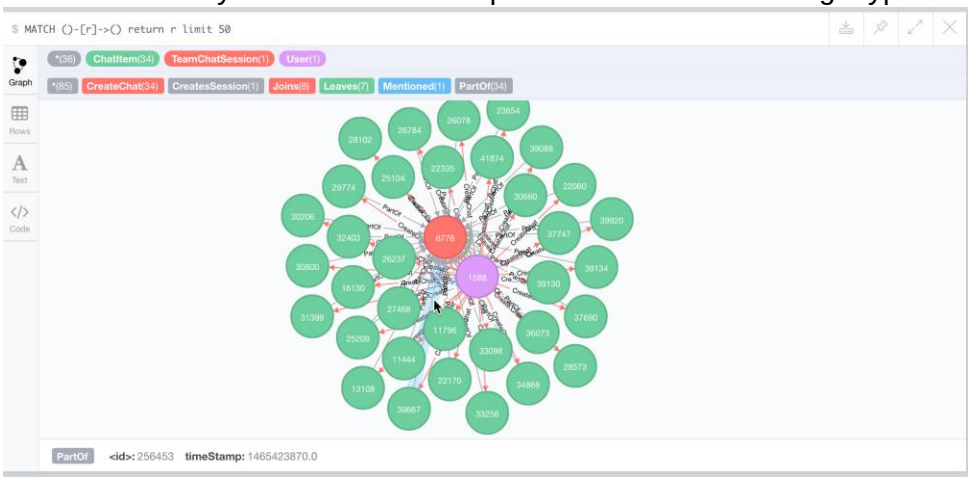
ii) Explain the loading process and include a sample LOAD command

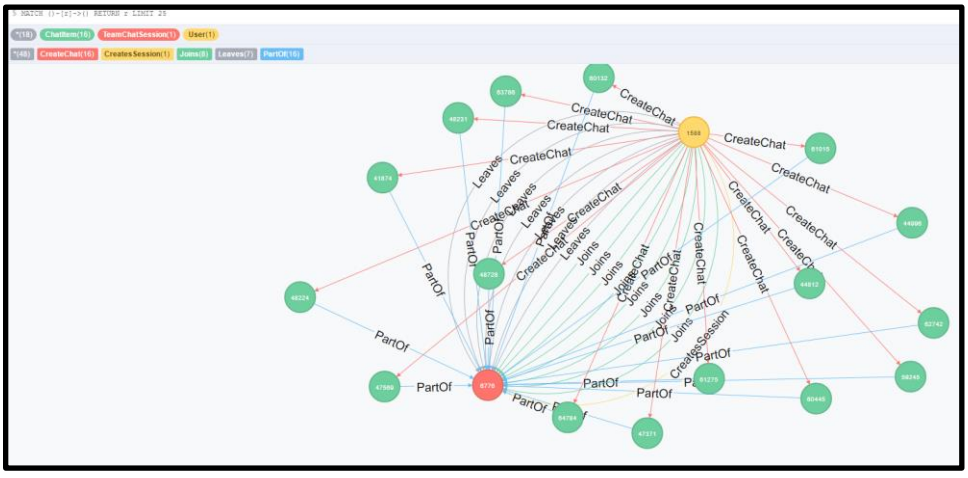
// loading process

```
LOAD CSV FROM "file:///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)
;
```

Loading is done from all the six csv files and the nodes and the edges are labelled accordingly, including their attributes. Note that the files are loaded without the headers.

iii) 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.

1) The longest conversation chain is traced via ChatItem nodes by the edge RespondTo and finding the longest one.

Query -> match p = (i1)-[:ResponseTo*]->(i2)
return length(p)
order by length(p) desc limit 1

Result : 10 chats

2) We are querying the path with the path length as calculated by the above query and calculating the distict users which were a part of it.

Query -> match p = (i1)-[:ResponseTo*]->(i2)
where length(p) = 9
with p
match (u)-[:CreateChat]->(i)
where i in nodes(p)
return count(distinct u)

Result : 5 users

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.

Ware showing those users who created maximum number of chat items.

Query -> match (u)-[:CreateChat*]->(i)
return u.id, count(i)
order by count(i) desc limit 10

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams

We are calculating the teams with maximum number of chat sessions.

Query -> match (i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)
return t.id, count(c)
order by count(c) desc limit 10

Teams	Number of Chats
82	1324
185	1036

112	957

Combining the above two queries to see whether the most chat users are part of the most chat teams or not.

Query -> match (u)-[:CreateChat*]->(i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)
return u.id, t.id, count(c)
order by count(c) desc limit 10

It showed that the user 999 which is in team 52 is part of the top 10 chattiest team but the other 9 are not.

How Active Are Groups of Users?

In this question we will compute an estimate of how "dense" the neighborhood of a node is.

- We will construct the neighborhood of users. We will connect two users if :
  - One mentioned another in the chat item :
    Match (u1:User)-[:CreateChat]->(i)-[:Mentioned]->(u2:User)
    Create (u1)-[:InteractsWith]->(u2)

  - One created a chat item in response to another :
    Match (u1:User)-[:CreateChat]->(i)-[:ResponseTo]-(i2:ChatItem] with u1,i1,i2 match u2-[:CreateChat]-(i2)
    Create (u1)-[:InteractsWith]->(u2)

The above scheme will create an undesirable side effect if a user has responded to their 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 .
Match (u1)-[r:InteractsWith]->(u1) delete r

For each of these neighbours, we need to find

- The number of edges it has with the other members on the same list
  match (u1:User)-[r1:InteractsWith]->(u2:User) where u1.id<>u2.id with u1, collect(u2.id) as neighbours, count(distinct(u2)) as neighbourAmount match (u3:User)-[r2:InteractsWith]->(u4:User) where (u3.id in neighbours) AND (ur.id in neighbours) AND (u3.id <> u4.id) return u3.id,u4.id,count(r2)

- If one member is connected multiple times we will count it as one.
  match (u1:User)-[r1:InteractsWith]->(u2:User) where u1.id<>u2.id with u1, collect(u2.id) as neighbours, count(distinct(u2)) as neighbourAmount match (u3:User)-[r2:InteractsWith]->(u4:User) where (u3.id in neighbours) AND (ur.id in neighbours) AND (u3.id <> u4.id) return u3.id, u4.id, count(r2), case when count(r2)>0 then 1 else 0 end as value

- Last step, combining the steps altogether

```
match (u1:User)-[r1:InteractsWith]->(u2:User) where u1.id<>u2.id with u1, collect(u2.id) as neighbours, count(distinct(u2)) as neighbourAmount match (u3:User)-[r2:InteractsWith]->(u4:User) where (u3.id in neighbours) AND (ur.id in neighbours) AND (u3.id <> u4.id)  with u1,u3,u4, neighbourAmount, case when (u3)-->(u4) then 1 else 0 end as value return u1, sum(value)*1.0/(neighbourAmount*(neighbourAmount-1)) as coeff order by coeff desc limit 10
```

User ID	Coefficient
209	0.9523809523809523
554	0.9047619047619048
1087	0.8