Technical Appendix
Catch the Pink Flamingo Analysis

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	When the user clicks on an advertisement in the Flamingo app, a line gets added to this file	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. id: the user id assigned to the user. nick: the nickname chosen by the user. twitter: the twitter account 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

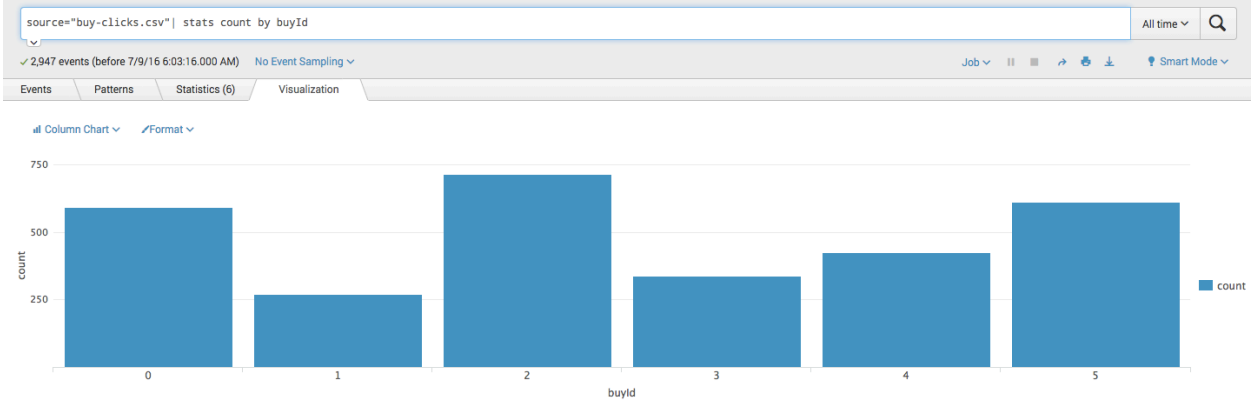
		currentLevel: the current level of the team
team-assignments.csv	A line is added to this file each time a user joins a team. A user can be in at most a single team at a time	time: when the user joined the team. team: the id of the team userid: the id of the user assignmentid: a unique id for this assignment
level-events.csv	A line is added to this file each time a team starts or finishes a level in the game	time: when the event occurred. eventid: a unique id for the event teamid: the id of the team level: 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.	time: 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

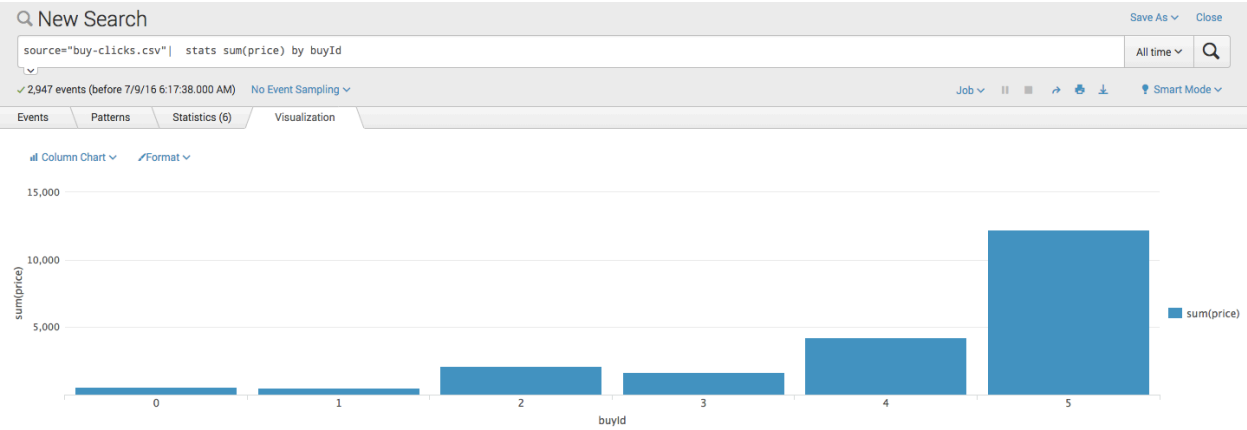
Amount spent buying items	21407
# Unique items available to be purchased	6

A histogram showing how many times each item is purchased:



buyId	count
0	592
1	269
2	714
3	337
4	425
5	610

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



buyId	sum(price)
0	592.0
1	538.0
2	2142.0
3	1685.0
4	4250.0
5	12200.0

Filtering

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

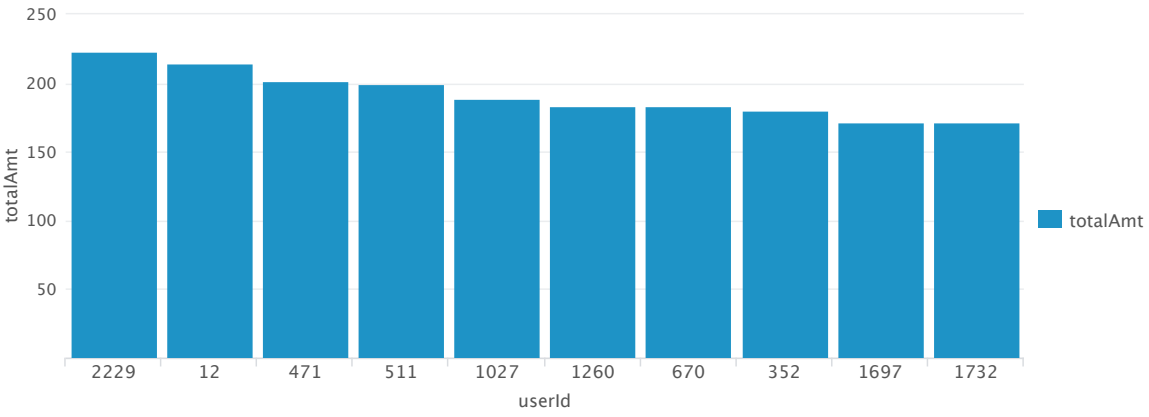
New Search

```
source="buy-clicks.csv"|  stats sum(price) as totalAmt by userId | sort 10 by totalAmt desc
```

All time

✓ 2,947 events (before 7/9/16 6:24:19.000 AM) No Event Sampling

Visualization



userId	totalAmt
2229	223.0
12	215.0
471	202.0
511	200.0
1027	189.0
1260	183.0
670	183.0
352	180.0
1697	172.0
1732	172.0

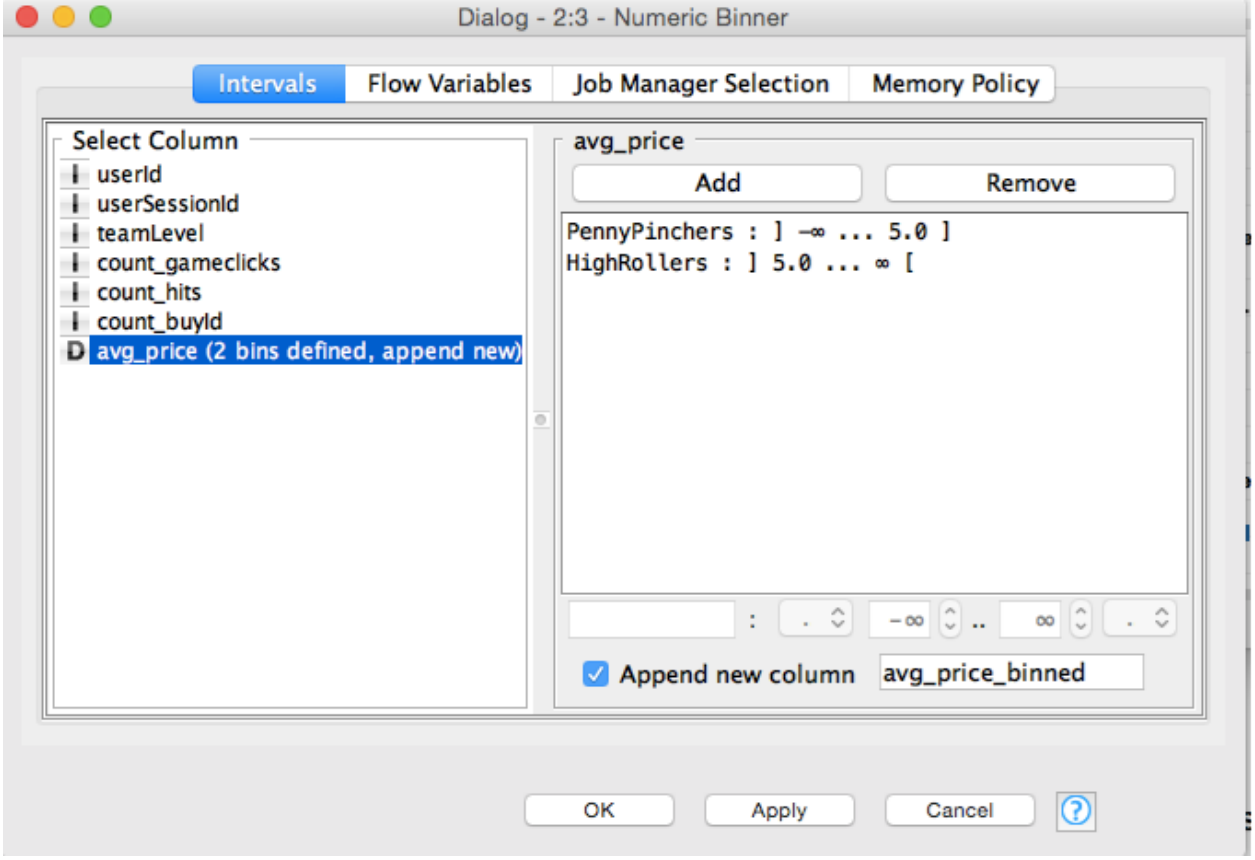
The following table shows the user id, platform, and hit-ratio percentage for the top three buying users:

Rank	User Id	Platform	Hit-Ratio (%)
1	2229	iphone	11.59
2	12	iphone	13.06
3	471	iphone	14.50

Data Preparation Analysis of combined_data.csv Sample Selection

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:

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



The buyers of big ticket items with price more than $5.00 have been classified as HighRollers. The buyers of inexpensive items which costs $5.00 or less are classified as PennyPinchers.

The creation of this new categorical attribute was necessary as the available data has to be analyzed to classify users as buyers of bigOticket items (“HighRollers”) vs. buyers of inexpensive items (“PennyPinchers”)

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

Attribute	**Rationale for Filtering**

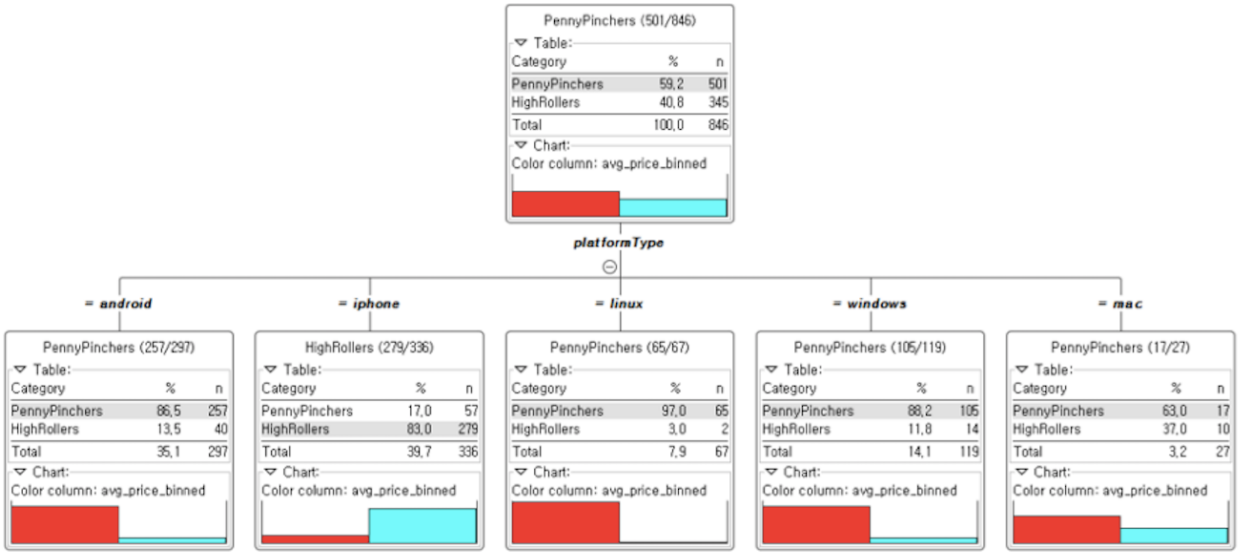
userId	It is not required for the classification

Data Partitioning and Modeling

The data was partitioned into train and test datasets. The train data set was used to create the decision tree model. The trained model was then applied to the test dataset. This is important because the model has to be tested on the data that was not used to train.

When partitioning the data using sampling, it is important to set the random seed because it ensures that the same partition is got every time while executing this node.

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



Evaluation

A screenshot of the confusion matrix can be seen below:

File Hilite

buyerType \ Prediction (buyerType)	PennyPinc...	HighRoller
PennyPincher	308	27
HighRoller	38	192

Correct classified: 500 Wrong classified: 65

Accuracy: 88.496 % Error: 11.504 %

Cohen's kappa (κ)

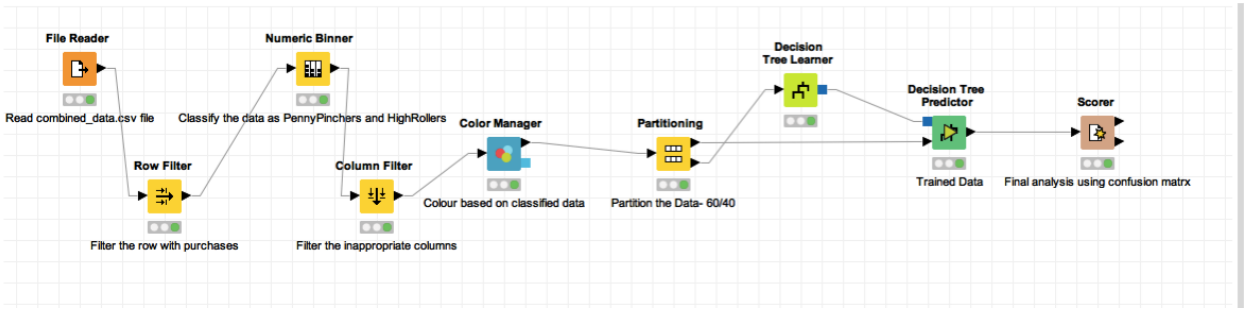
As seen in the screenshot above, the overall accuracy of the model is 88.496 %

308 users are correctly classified as PennyPinchers 27 users are wrongly classified as HighRollers when they are PennyPinchers 192 users are correctly classified as HighRollers

38 users are wrongly classified as PennyPinchers when they are HighRollers

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher? Many users having IPhone are HighRollers and most of the users of other platform types such as Windows, Mac or Linux are PennyPinchers

Specific Recommendations to Increase Revenue
1. Advertisements for app purchases can be increased for users of Windows, L Mac
2. Some flashy discounts on the price of apps can be given for the users

Clustering Analysis:

Attribute Selection

Attribute	Rationale for Selection
gameclick	Number of times a user makes a click in the game. Captures how effectively a user plays a game
adcount	No of ads clicked by user. Givers users inclination to click on an ad
buycount	Number of times a user buys an app. Captures the purchase behavior of a user

Cluster Centers

Cluster #	Cluster Center
1	array([2310.64444444, 32.35555556, 5.53333333])
2	array([357.95924765, 24.98746082, 4.81191223])
3	array([926.11731844, 36.44134078, 6.47486034])

These clusters can be differentiated from each other as follows:

In the first cluster, the user clicks more often on games, which is ~6.4 times compared to the second cluster, clicks ~1.3 and ~1.1 more times on ads and buy

From second cluster, it can be seen that when users click on game less often, they also click on ads and buy for less number of times

Training Data Set Creation

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

```
>>> trainingDF.head(n=5)
   totalGameClicks  totalAdClicks  totalBuyClicks
0              716             44               9
1              380             10               5
2              508             37               6
3             3107             19              10
4              704             46              13
```

Dimensions of the training data set (rows x columns): (543, 3)

# of clusters created: 3

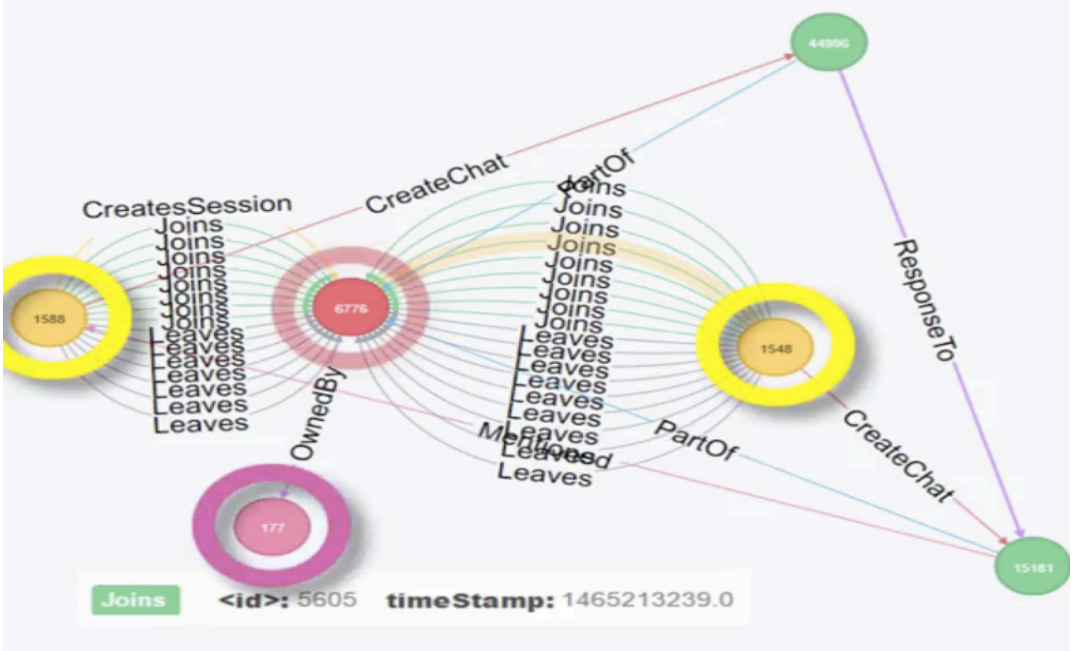
Recommended Actions

Action Recommended	**Rationale for the action**
Increasing number of ads for players with more game clicks	Players who have more number of game clicks, are inclined to click on ads more often, as seen in the clustering model. This may increase the number of apps that is being bought.
Increase the cost of app for players with more game and ad clicks	These players buy apps more often and increasing the cost will help Elegence Inc. increase their revenue

Graph Analytics

Modeling Chat Data using a Graph Data Model

The below picture shows the graph for chat data,



The chat data graph has four entity types such as user, team, team chat session and chat item. These entities are represented as nodes in graph. The users and the teams have their corresponding ID's. When a user creates a new chat session for a new team, team chat sessions are created. The edges represent the different actions by the user. The user can create a session for the team, create a new chat in the session, may join or leave the chat session, mention another user and respond to a previous chat item. Each action has a timestamp.

Creation of the Graph Database for Chats

Schema of Graph Database for Chats:

File: chat_create_team_chat.csv
ERD table:
chat_create_team_chat
userId
teamId
teamchatsessionid

timestamp

File: chat_item_team_chat.csv
ERD table:
chat_item_team_chat
userId
teamchatsessionid
chatitemid
timestamp

File: chat_join_team_chat.csv
ERD table:
chat_join_team_chat
userid,
teamchatsessionid
teamstamp

File: chat_leave_team_chat.csv
ERD table:
chat_leave_team_chat
userid
teamchatsessionid
teamstamp

File: chat_mention_team_chat.csv
ERD table:
chat_mention_team_chat
chatItem
userid
timeStamp

File: chat_respond_team_chat.csv
ERD table:
chat_respond_team_chat
chatid1
chatid2
timestamp

Loading CSV Files:

The CSV data files which has the chat data is loaded into Neo4j. The following are some of the scripts used for loading.

1) **Loading chat_create_team_chat.csv**

```
LOAD CSV FROM "file:/path/to/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)
```

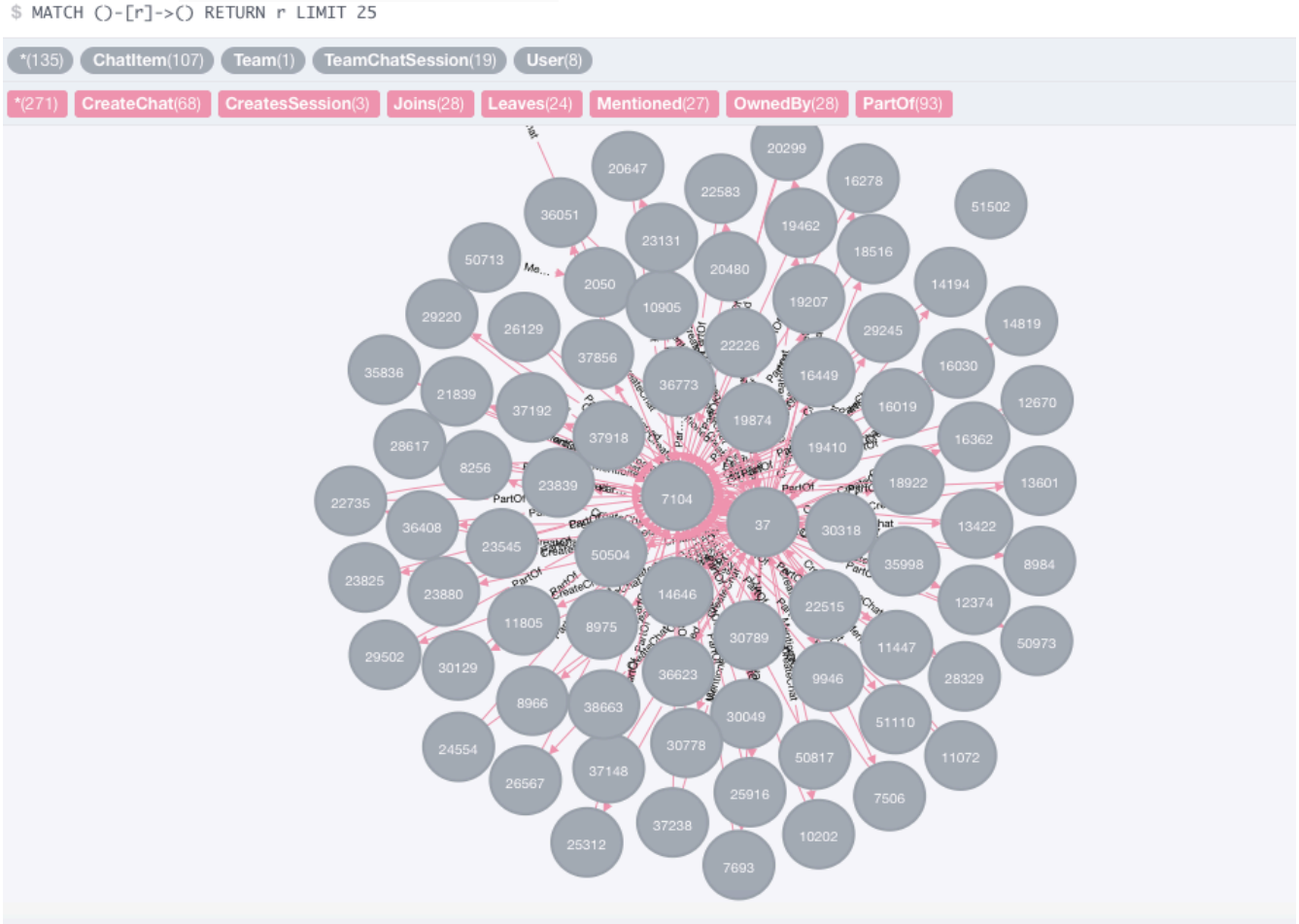
2) Loading chat_mention_team_chat.csv

```
LOAD CSV FROM "file:/path/to/chat_respond_team_chat.csv" AS row
MERGE (u:ChatItemId1 {id: toInt(row[0])})
MERGE (t:ChatItemId2 {id: toInt(row[1])})
MERGE (u)-[:ResponseTo {timeStamp:row[2]}]->(t)
```

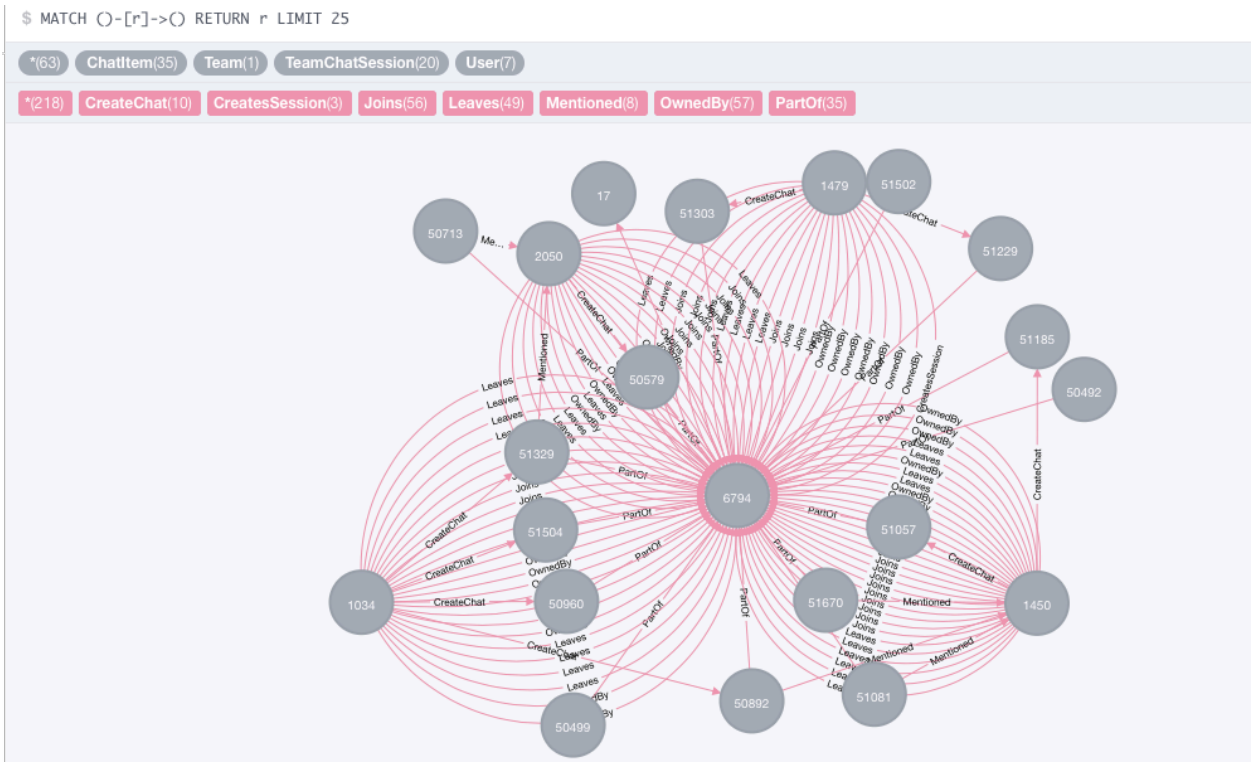
Graph Generated for Chat Data:

The following images shows the graph that has been generated for chat data in Neo4j.

1)



2) A closer image of the nodes and edges:



Longest Conversation Chain and Its Participants:

The longest conversation chain is found using the following query,

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

The length of the conversation chain is 9.

The number of unique users who part of the conversation chain is 5. This can be found by using the following query,
match p=()-[:ResponseTo]->()
where length(p)=9
with p
match (u:User)-[:CreateChat]-(i)
where i in nodes(p) return count(distinct u)

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

Chattiest Users:

The relationship between the top ten chattiest users can be found using the below query,

```
match (u:User)-[c:CreateChat]-()
return u.id as User, count(c) as NumberOfUserChats
order by NumberOfUserChats desc limit 10
```

The top three chattiest users are as follows,

Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams:

The top ten chattiest teams can be found using the following query,

```
match (i:ChatItem)-[r1:PartOf]->(c:TeamChatSession)-[r2:OwnedBy]->(t:Team)
return t.id,
count(t) as NumberOfChats
order by NumberOfChats desc limit 10
```

The top three chattiest teams are as follows,

Team	Number of Chats
82	1324
185	1036
112	957

The following query tells whether or not any of the chattiest users are part of any chattiest teams.

```
match (u:User)-[c:CreateChat]->(s:TeamChatSession)-[o:OwnedBy]->(t:Team)
where u.id in [394, 2067, 209, 1087, 554, 516, 1627, 999, 668, 461] return distinct u.id, t.id
```

There is only one user ‘999’ who is in one of the chattiest teams.

Active Users of the Group:

To find the active users of the group, we have to first compute how “dense” the neighborhood of a node is, in the context of chat that translates to how mutually interactive a certain group of users are. If we can identify these highly interactive neighborhoods, we can potentially target some members of the neighborhood for direct advertising. This can be done with the following series of steps,

1) We have to contrast 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 users chatItem.

The query for the first condition, ’One mentioned another user in a chat’ is as follows,

```
match (u1:User)-[r:CreateChat]-(t:TeamChatSession)-[p:PartOf]-(c:ChatItem)-[m:Mentioned]-
>(u2:User)
create (u1)-[i:InteractsWith]->(u2)
```

The query for the second condition, ‘One created a chatItem in response to another users chatItem’ is as follows,

```
match (u1:User)-[r1:CreateChat]-(c1:TeamChatSession)-[p1:PartOf]-(i1:ChatItem)-
[r:ResponseTo]- (i2:ChatItem)-[p2:PartOf]-(c2:TeamChatSession)-[r2:CreateChat]-(u2:User)
create (u1)-[i:InteractsWith]->(u2)
```

2) 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, we have to eliminate all self-loops involving the edge “Interacts with”. This can be achieved using the following query,

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

3) Given this new edge type, we will have to create a scoring mechanism to find users having dense neighborhoods. The score should range from 0 (a disconnected user) to 1 (a user in a

clique – where every node is connected to every other node). One such scoring scheme is called a “clustering coefficient” defined as follows. If the number of neighbors of node is 5, then the clustering coefficient of the node is the ratio between the number of edges amongst these 5 neighbors(not counting the given node) and 5*4 (all the pairwise edges that could possibly exist). Thus the denominator is k * (k-1) if the number of neighbors of the node is k.

The clustering coefficients of the top three chattiest users has to be calculated.

4) To do this computation, we have to,

- Get the list of neighbors and
- The number of neighbors of a node based on the “InteractsWith” edge.

```
match (u1:User)-[:InteractsWith]-(u2:User) where u1.id in [209,2067,394]
with u1, count(distinct u2) as degree set u1.deg = degree
return u1.id, u1. deg
```

5) The clustering coefficient is calculated using the following query,

```
match (u1:User)-[i:InteractsWith]-(u2:User)
where u1.id in [394,209,2067]
with u1 as u1node, collect(distinct u2.id) as neighbors
match (a)-[r:InteractsWith]-(b)
where a.id in neighbors and b.id in neighbors
return u1node.id, toFloat(count(distinct a.id + ' ' + b.id))*1000/(u1node.deg*(u1node.deg-
1))/1000 as coefficient order by coefficient desc
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	1.2
209	1.1666666666666667
2067	1.0178571428571428