Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: Rajat Sharma

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
adclicks.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 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
buyclicks.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.

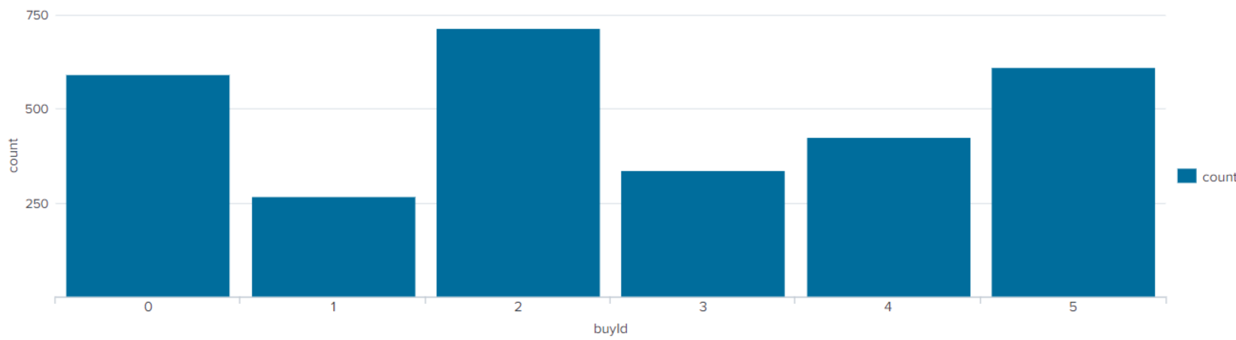
		nick: the nickname chosen by the user. twitter: the twitter handle of the use r. dob: the date of birth of the user. country: the two- letter country code where the user lives.
team.csv	This file contains a line for each team terminated in the game.	teamId: the id of the team name: the name of the team teamCreationTime: the timestamp when the team was created teamEndTime: the timestamp whe n the last member left the team strength: a measure of team stren gth, roughly corresponding to the s uccess of a team currentLevel: the current level of th e team
teamassignments.csv	A line is added to this file each ti me a user joins a team. A user c an be in at most a single team at a time.	timestamp: when the user joined th e team. team: the id of the team userId: the id of the user assignmentId: a unique id for this a ssignment
levelevents.csv	A line is added to this file each ti me a team starts or finishes a le vel in the game	timestamp: when the event occurr ed. eventId: a unique id for the event teamId: the id of the team teamLevel: the level started or co mpleted eventType: the type of event, eithe r start or end
usersession.csv	Each line in this file describes a user session, which denotes wh en a user starts and stops playing the game.	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 assignmen t 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 d uring this session. platformType: the type of platform of the user during this session.

gameclicks.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

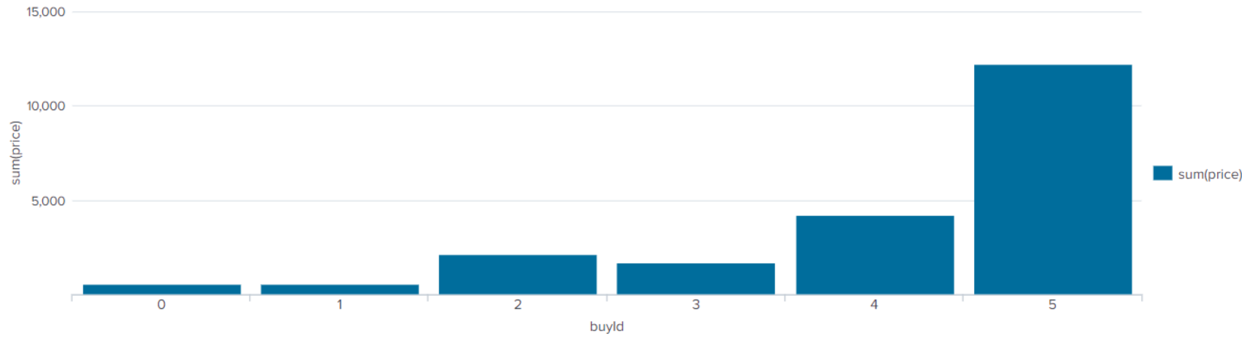
Aggregation

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:

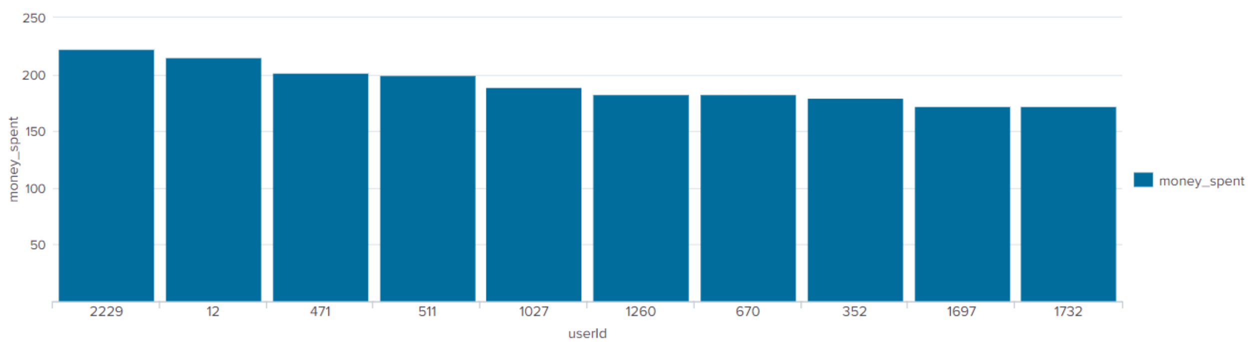


A histogram showing how much money was made from each item:



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	229	iphone	0.11596958174904944
2	12	iphone	0.13068181818181818
3	471	iphone	0.1450381679389313

Data Classification Analysis

Data Preparation

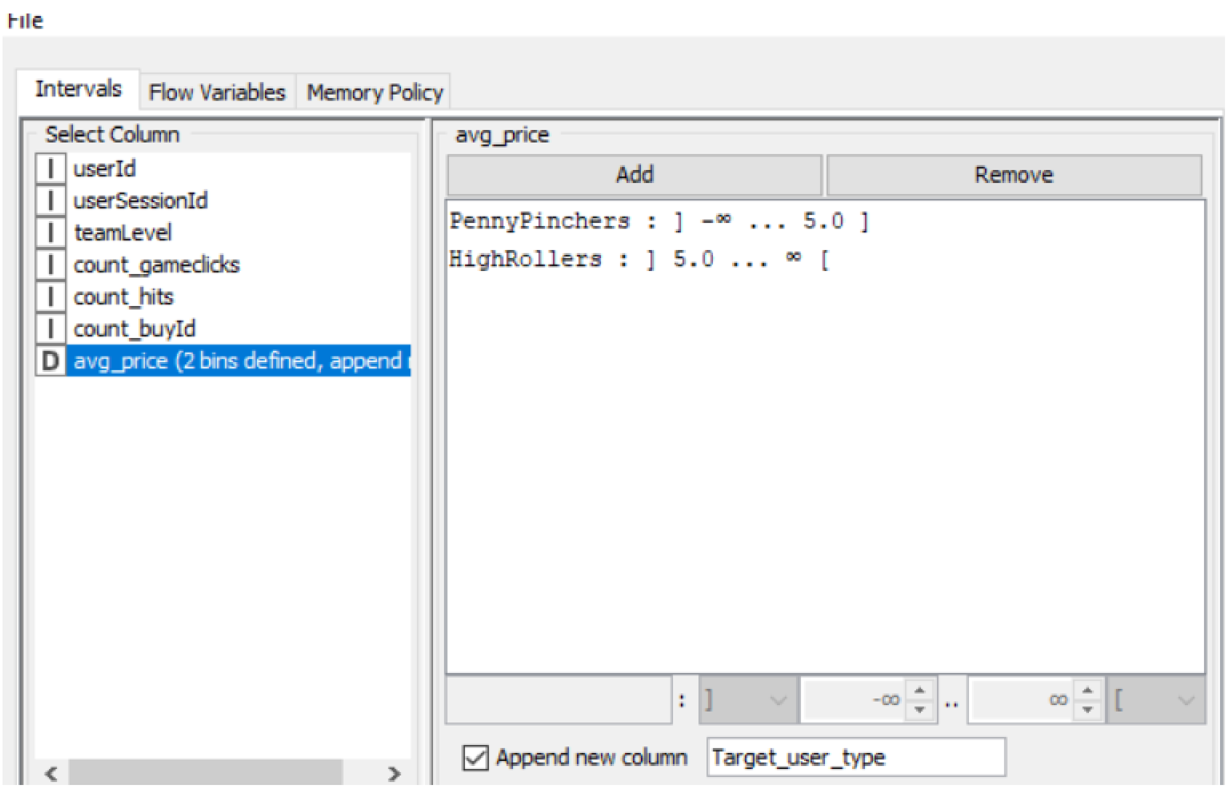
Analysis of combined_data.csv

<u>Sample Selection</u>

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

<u>Attribute Creation</u>

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	Target_user_type
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

Categorical attribute name: *Target_user_type*
Description: It is derived by binning avg_price attribute between HighRollers (buyers of items that cost more than $5.00) and PennyPinchers (buyers of items that cost $5.00 or less).

The new attribute will act as our target variable for training our classifier and segmenting the users between HighRollers and PennyPinchers. This classier is then used for predicting the type of new user (unseen data) in future.

Attribute Selection

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

Attribute	Rationale for Filtering
UserId	Id field
userSessionId	Id field
Avg_price	Numerical field from which target field is created

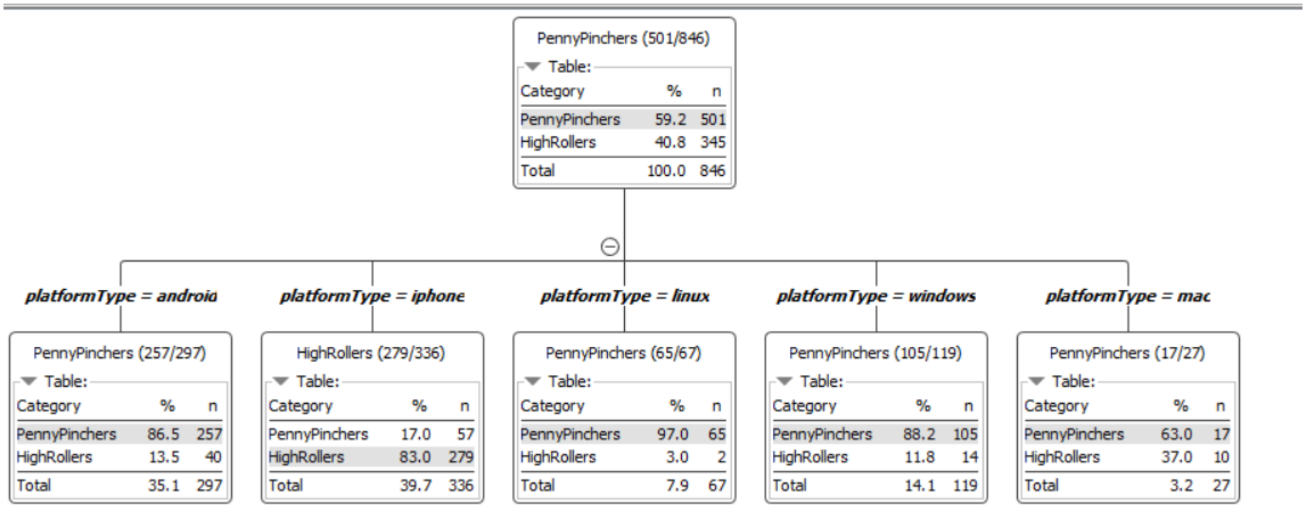
Data Partitioning and Modeling

The data was partitioned into train and test datasets.
The Training data set (846 rows) was used to create the decision tree model.
The trained model was then applied to the test dataset (565 rows).
This is important because to validate our model.

When partitioning the data using sampling, it is important to set the random seed because…
So that reproduceable results can be produced, other result may vary after every execution.

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

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



Evaluation

A screenshot of the confusion matrix can be seen below:

Prediction (Target_user_type) \ Target_user_type	PennyPinchers	HighRollers
PennyPinchers	308	38
HighRollers	27	192

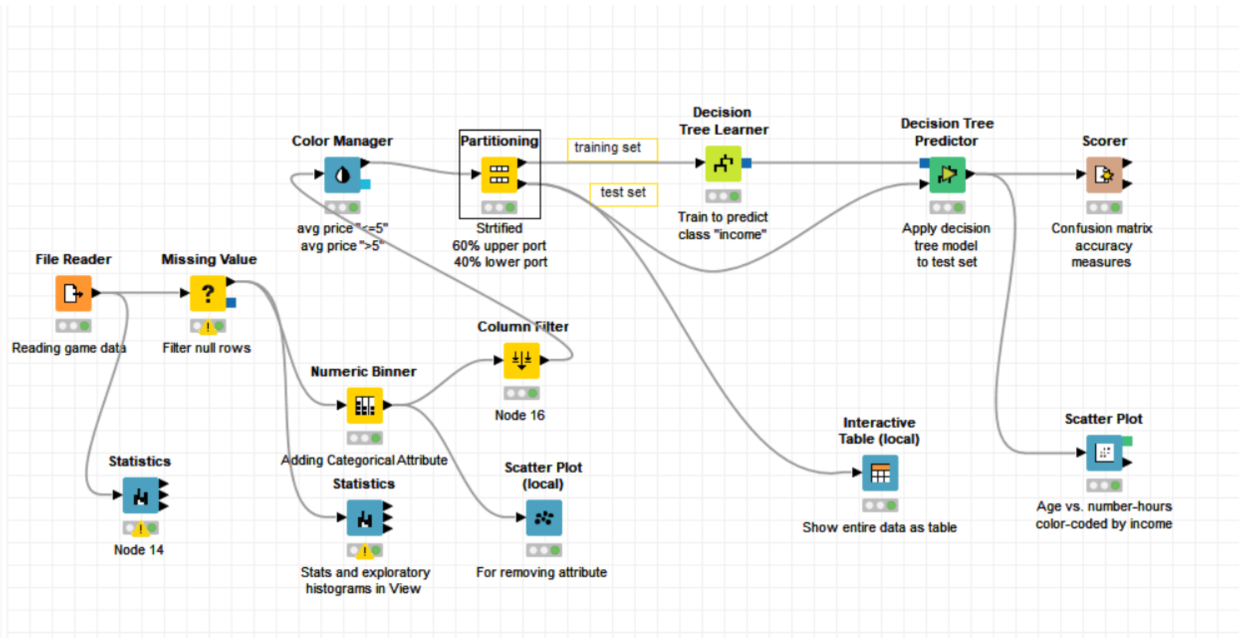
As seen in the screenshot above, the overall accuracy of the model is 88.5

308 were classified PenneyPicher correctly. True positive
38 were classified PenneyPicher incorrectly. False positive
27 were classified HighRollers incorrectly. False positive
192 were classified HighRollers correctly. True positives

Analysis Conclusions

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

According to the classification tree, iphone users tend to spend more on their purchases when comparing with users that use other platforms.

HighRollers are notoriously associated with the Iphone plataform, given the fact that over 90% of the samples used to train the decision tree are from Iphone users. On the other hand windows has the highest presence of PennyPichers but is around 80% and the difference with the rest of the plataforms remains insignificant. HighRollers is almost sinonim of iPhone users in this data.

Specific Recommendations to Increase Revenue
1. Most of the Iphone (86%) and of Mac(33%) user are Highroller, Focus development and marketing initiatives on iphone users
2. 35% of total users are android user. As android user hold majority of market, Give preference in terms of new developments or marketing to mobile users.

Clustering Analysis
Attribute Selection

Attribute	Rationale for Selection
revenue	Represents the amount of money spent by a user in the game
gameClickSum	Sum of clicks on the game can be understood as engagement with the game
adClickSum	Sum of clicks on ads can tanslate on the quantifying of chance of purchase
isHitSum	Total click accuracy of user. More hit means more **skill** in game. To identify effect of skill on profit the game.

Training Data Set Creation

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

```
mergeddf[featuresUsed].head(5)
```

	adClicks	gameClicks	isHit	revenue
0	25	465	62	21.0
1	9	355	37	53.0
2	19	312	33	80.0
3	16	2275	240	11.0
4	36	528	70	215.0

Dimensions of the final data set: 542x4

# of clusters created: 3

Cluster Centers

Cluster #	Center **[**adClicks, gameClicks, isHit, revenue**]**
1	[0.84748799, 0.15359459, 0.19433942, 0.52858763]

2	[-0.75304428, -0.51330678, -0.53621493, -0.45200352]
3	[0.35883284, 2.95405321, 2.86936817, 0.08827273]

Cluster 0: Customers on this cluster generate **highest revenue** and are **not the ones who have most game engagement but an intermediate (less Skilled) result** in game clicks.
Cluster 1: Customers of this cluster tend to **play the less** also **produces the less ad revenue**
Cluster 2: Customers that are **very active** and **generate a moderate amount of revenue**

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
Show more ads to user who have intermediate result (less skilled user)	User with intermediate result buy more

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

A graph is used to represent the chat data model because its composed of several entities that relationships among them, for example: When one User creates a TeamChatSession, it is then owned by team. Users can join and leave the TeamChatSession. In TeamChatSession, users can create ChatItem that is part of TeamChatSession. ChatItem could also be mentioned by Users. And User could respond to User as well. All the relationships are recorded with timestamp.

- Vertices (Entities)

- User
- Team
- TeamChatSession
- ChatItem

- Edges (Relationships)
  - User creates TeamChatSession with timestamp
  - Team owns TeamChatSession with timestamp
  - User joins TeamChatSession with timestamp
  - User leaves TeamChatSession with timestamp
  - User creates ChatItem with timestamp
  - ChatItem is part of TeamChatSession with timestamp
  - ChatItem is mentioned by User with timestamp
  - ChatItem responses to ChatItem with timestamp

Creation of the Graph Database for Chats

i)

File Name	Fields	Description
chat_create_team_chat.csv	userID	the user id assigned to the user
	teamID	the id of the team
	teamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the chat session created
chat_item_team_chat.csv	userID	the user id assigned to the user
	teamChatSessionID	a unique id for the chat session
	chatItemID	a unique id for the chat item
	timestamp	a timestamp denoting when the chat item created
chat_join_team_chat.csv	userID	the user id assigned to the user

	teamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the user join in a chat session
chat_leave_team_chat.csv	userID	the user id assigned to the user
	teamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the user leave a chat session
chat_mention_team_chat.csv	chatItemID	the id of the ChatItem
	userID	the user id assigned to the user
	timestamp	a timestamp denoting when the user mentioned by a chat item
chat_respond_team_chat.csv	chatID1	the id of the chat post 1
	chatID2	the id of the chat post 2
	timestamp	a timestamp denoting when the chat post 1 responds to the chat post 2

```
 # Clear database
MATCH (n)
OPTIONAL MATCH (n)-[r]-()
DELETE n,r

 # Create the constraint primary key
CREATE CONSTRAINT ON (u:User) ASSERT u.id IS UNIQUE;
CREATE CONSTRAINT ON (t:Team) ASSERT t.id IS UNIQUE;
CREATE CONSTRAINT ON (c:TeamChatSession) ASSERT c.id IS UNIQUE;
CREATE CONSTRAINT ON (i:ChatItem) ASSERT i.id IS UNIQUE;

# Load chat_create_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/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)
```

Load chat_join_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_join_team_chat.csv" AS row MERGE (u:User {id: toInt(row[0])}) MERGE (c:TeamChatSession {id: toInt(row[1])}) MERGE (u)-[:Join{timeStamp: row[2]}]->(c)

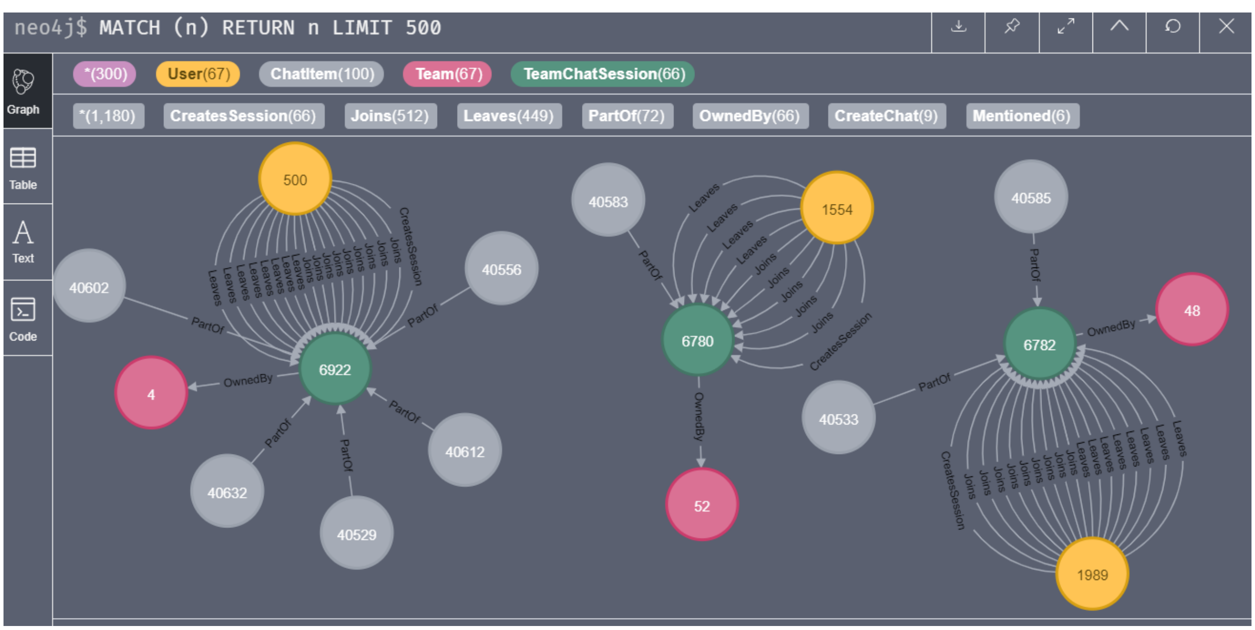
Load chat_leave_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_leave_team_chat.csv" AS row MERGE (u:User {id: toInt(row[0])}) MERGE (c:TeamChatSession {id: toInt(row[1])}) MERGE (u)-[:Leave{timeStamp: row[2]}]->(c)

Load chat_item_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_item_team_chat.csv" AS row MERGE (u:User {id: toInt(row[0])}) MERGE (c:TeamChatSession {id: toInt(row[1])}) MERGE (i:ChatItem {id: toInt(row[2])}) MERGE (u)-[:CreateChat{timeStamp: row[3]}]->(i) MERGE (i)-[:PartOf{timeStamp: row[3]}]->(c)

Load chat_mention_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_mention_team_chat.csv" AS row MERGE (i:ChatItem {id: toInt(row[0])}) MERGE (u:User {id: toInt(row[1])}) MERGE (i)-[:Mentioned {timeStamp: row[2]}]->(u)

Load chat_respond_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_respond_team_chat.csv" AS row MERGE (i:ChatItem {id: toInt(row[0])}) MERGE (j:ChatItem {id: toInt(row[1])}) MERGE (i)-[:ResponseTo {timeStamp: row[2]}]->(j)

A screenshot of the graph generated with clearly visible examples of most node and edge types.



Finding the longest conversation chain and its participants

The length of the conversation is 9 and the number of unique users that were part of the conversation chain, is 5.

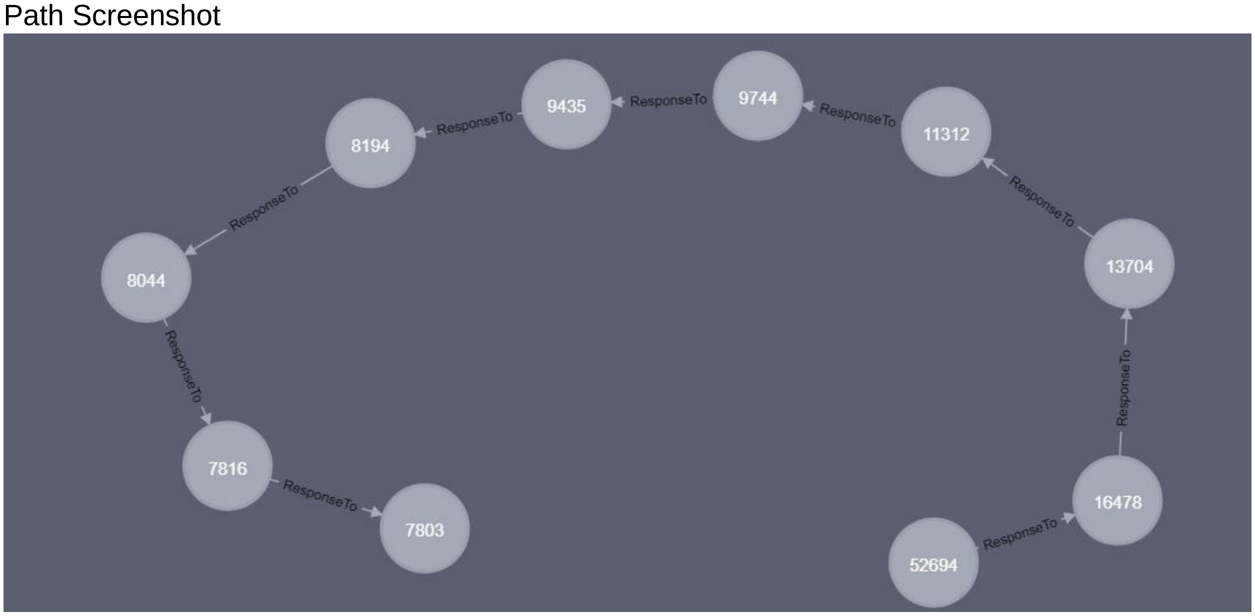
- Path length of longest conversation is 9.
- Unique users involved in the conversation are below

{"id":1192}
{"id":1978}
{"id":1153}
{"id":853}
{"id":1514}

- 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

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.

The process of finding the chattiest users involves finding the origins of the edge create chat counting the Id's in descending order and looking at the first ten elements. To find the chattiest team a longer chain needed to be used having the form:

(:ChatItem)-[r:PartOf]->(:TeamChatSession)-[k:OwnedBy]->(n)

- *Chattiest Users*
  match (m)-[:CreateChat]->(n)
  return m.id, count(n)
  order by count(n) desc limit 10
- Result

```
neo4j$ match (m)-[:CreateChat]→(n) return m.id, count(n) order by count(n) desc limit 10
```

Table Text Code

m.id	count(n)
394	115
2067	111
209	109
1087	109
554	107
1627	105
999	105
516	105
461	104
668	104

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

- *Chattiest team*
  match (m:ChatItem)-[:PartOf]->(:TeamChatSession)-[:OwnedBy]->(n)
  return n.id, count(n)
  order by count(n) desc limit 10

- *Result*

```
neo4j$ match (m)-[:CreateChat]→(n) return m.id, count(n) order by count(n) desc limit 10
```

m.id	count(n)
394	115
2067	111
209	109
1087	109
554	107
1627	105
999	105
516	105
461	104
668	104

Chattiest Teams

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.

- Query

  match (m:User)-[]->(:TeamChatSession)-[:OwnedBy]->(n:Team)

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

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

  return n.id, m.idResult

- *Result*

```
1 match (m:User)-[]→(:TeamChatSession)-[:OwnedBy]→(n:Team)
2 where n.id in [82, 185, 112, 18, 194, 129, 52, 136, 146,81]
3 and m.id in [2067, 394, 209, 1087,554,1627,999,516,461,668]
4 return distinct n.id, m.id
```

```
eo4j$ match (m:User)-[]→(:TeamChatSession)-[:OwnedBy]→(n:Team) where n.id in
```

n.id	m.id
52	999

- **Explanation**

The user 999, which is in the team 52 is part of the top 10 chattiest teams, but other 9 users are not part of the top 10 chattiest teams. This demonstrates that most of the chattiest users are not in the chattiest teams.

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.

- Constructing neighborhood of users using the following criteria: one mentioned another in a chat and one created a chatItem in response to another
    - Query:

```
neo4j$ Match (u1:User)-[:CreateChat]→(:ChatItem)-[:Mentioned]→(u2:User) create (u1)-[:InteractsWith]→(u2)
```

```
neo4j$ Match (u1:User)-[:CreateChat]→(:ChatItem)-[:Mentioned]→(u2:User) create (u1)-[:InteractsWith]→(u2)
Created 11084 relationships, completed after 205 ms.
```

```
1 Match (u1:User)-[:CreateChat]→(:ChatItem)-[:ResponseTo]→(:ChatItem)←[:CreateChat]-(u2:User)
2 create (u1)-[:InteractsWith]→(u2)
```

```
4j$ Match (u1:User)-[:CreateChat]→(:ChatItem)-[:ResponseTo]→(:ChatItem)←[:CreateChat]-(u2:User) create
Created 11073 relationships, completed after 201 ms.
```

- Removing self-loop
    - Query:

```
neo4j$ Match (u1)-[r:InteractsWith]→(u1) delete r
```

```
neo4j$ Match (u1)-[r:InteractsWith]→(u1) delete r
Deleted 4377 relationships, completed after 446 ms.
```

- Getting coefficient

```
match (u1:User {id:209})-[:InteractsWith]->(u2)
with collect(u2.id) as neb, count(u2) as k
match (m:User)-[r:InteractsWith]->(n)
where (m.id in neb) and (n.id in neb)
return count(r)/(k*(k-1)*1.0) as cc
```

User ID	Coefficient
209	0.95
554	0.9
1087	0.8

Recommended Actions

1. I-phone users tend to spend more on their purchases. Focus development and marketing initiatives on i-phone users.
2. Show very cash convertible ads to user with intermediate results.
3. Need to create low cost/promotional offers for highly active users