Technical Appendix

Catch the Pink Flamingo Analysis

Produced by: Ku Wei Chiet

Contents

Part 1: Acquiring, Exploring and Preparing the Data

1.1 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	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	timestamp: when user first played the

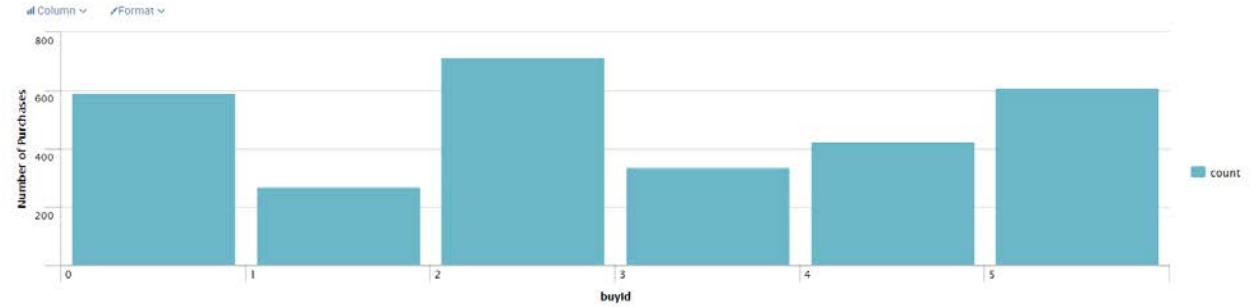
	user playing the game.	game. userId: the user id assigned to the user. nick: the nickname chosen by the user. twitter: the twitter handle of the user. dob: the date of birth of the user. country: the two-letter country code where the user lives.
teams.csv	This file contains a line for each team terminated 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
teamassignments.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

1.2 Aggregation

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

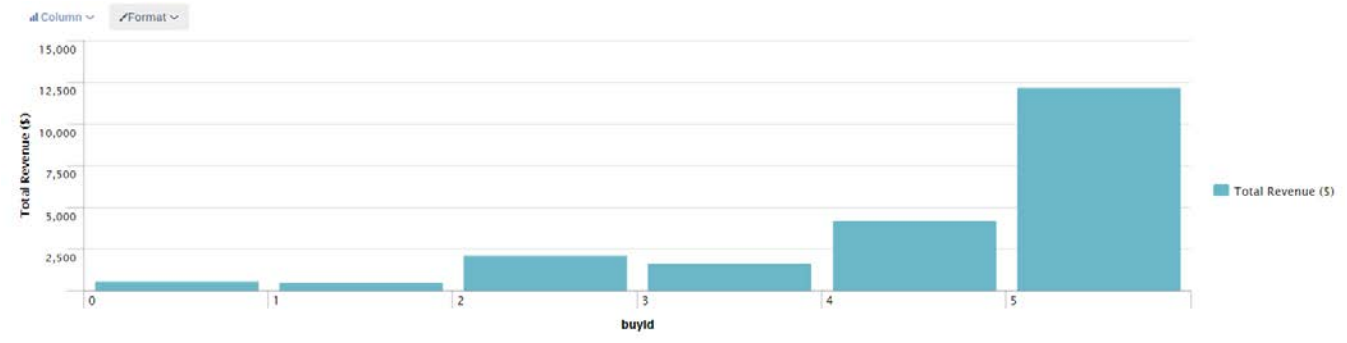
A histogram showing how many times each item is purchased:



buyId	count
0	592
1	269
2	714
3	537
4	425
5	610

Query: *source="buy-clicks.csv" | stats count by buyId | rename count as "Number of Purchases"*

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

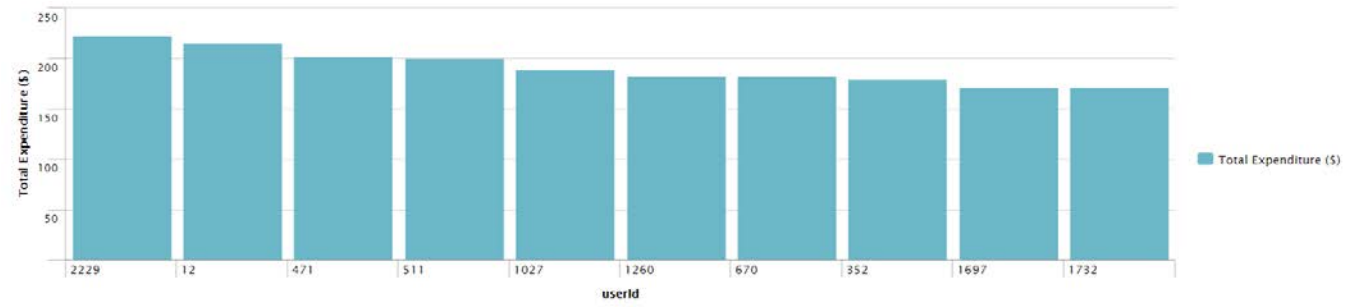


buyId	Total Revenue ($)
0	592.0
1	538.0
2	2142.0
3	1685.0
4	4250.0
5	12200.0

Query: *source="buy-clicks.csv" | stats sum(price) as "Total Revenue ($)" by buyId*

1.3 Filtering

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



userId	Total Expenditure ($)
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

Query: *source="buy-clicks.csv" | stats sum(price) as "Total Expenditure ($)" by userId | sort -num("Total Expenditure ($)") | head 10*

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	11.60%
2	12	iphone	13.07%
3	471	iphone	14.50%

Part 2: Data Classification Analysis

2.1 Data Preparation

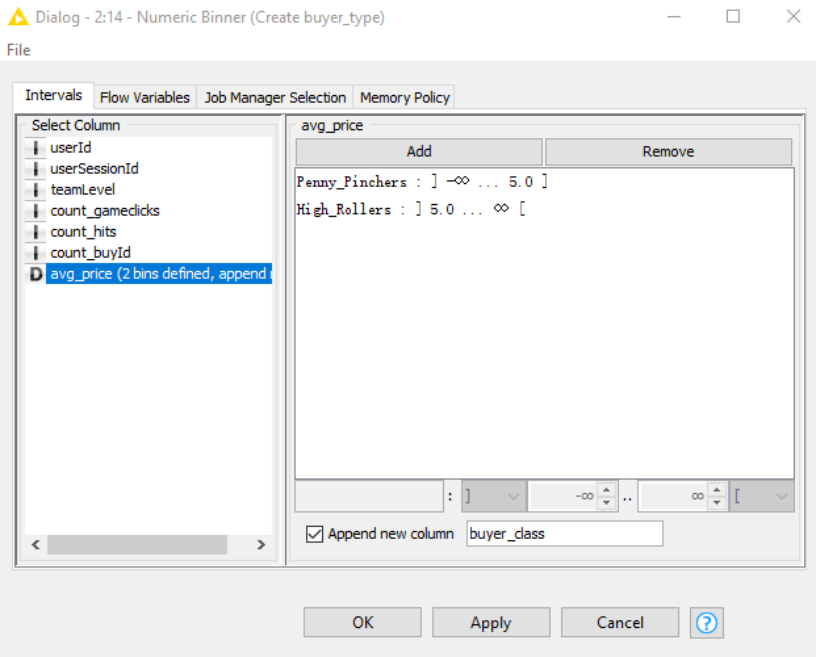
Analysis of combined_data.csv

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:



Configuring the Numeric Binner node

Binned Data - 2:14 - Numeric Binner (Create buyer_type)

File Hilite Navigation View

Table "default" - Rows: 1411 | Spec - Columns: 9 | Properties | Flow Variables

Row ID	userId	userSe...	teamLevel	platfor...	count_...	count_...	count_...	avg_price	buyer_class
Row4	937	5652	1	android	39	0	1	1	Penny_Pinchers
Row11	1623	5659	1	iphone	129	9	1	10	High_Rollers
Row13	83	5661	1	android	102	14	1	5	Penny_Pinchers
Row17	121	5665	1	android	39	4	1	3	Penny_Pinchers
Row18	462	5666	1	android	90	10	1	3	Penny_Pinchers
Row31	819	5679	1	iphone	51	8	1	20	High_Rollers
Row49	2199	5697	1	android	51	6	2	2.5	Penny_Pinchers
Row50	1143	5698	1	android	47	5	2	2	Penny_Pinchers
Row58	1652	5706	1	android	46	7	1	1	Penny_Pinchers
Row61	2222	5709	1	iphone	41	6	1	20	High_Rollers
Row68	374	5716	1	android	47	7	1	3	Penny_Pinchers
Row72	1535	5720	1	iphone	76	7	1	20	High_Rollers
Row73	21	5721	1	android	52	2	1	3	Penny_Pinchers
Row101	2379	5749	1	android	62	9	1	3	Penny_Pinchers
Row122	1807	5770	1	iphone	177	25	2	7.5	High_Rollers
Row127	868	5775	1	iphone	54	5	1	10	High_Rollers
Row129	1567	5777	1	android	27	4	2	4	Penny_Pinchers
Row131	221	5779	1	iphone	37	2	1	20	High_Rollers
Row135	2306	5783	1	android	67	5	1	1	Penny_Pinchers
Row137	1065	5785	1	iphone	37	5	2	11.5	High_Rollers
Row140	827	5788	1	iphone	75	5	1	20	High_Rollers
Row150	1304	5798	1	mac	71	9	2	11.5	High_Rollers
Row158	1264	5806	1	linux	81	12	1	5	Penny_Pinchers
Row159	1026	5807	1	iphone	52	10	1	20	High_Rollers
Row163	649	5811	1	linux	51	9	1	1	Penny_Pinchers

New created columns, buyer_class

By using Numeric Binner node, a new categorical attribute was crated based on ‘avg_price’ attribute. Samples with ‘avg_price’ more than $5 were classified as ‘HighRollers’, while others with $5 or less were classified as ‘PennyPinchers’.

The creation of this new categorical attribute was necessary because we are having a classification problem and the new categorical attribute would be served as the model answer for the training and validation of the Decision Tree model. Yes, the explanation discussed that because this is a classification task, one cannot use a continuous-value field like avg-price

Attribute Selection

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

Attribute	**Rationale for Filtering**
userId	This index is redundant because it has no logical bearing on deciding whether a player is a penny-pincher or a high-roller.
userSessionId	This index is redundant because it has no logical bearing on deciding whether a player is a penny-pincher or a high-roller.
avg_price	It is redundant now because the new categorical attribute ‘buyer_class’ would be used as the label for classification problem.

2.2 Data Partitioning and Modeling

The data was partitioned into train and test datasets.

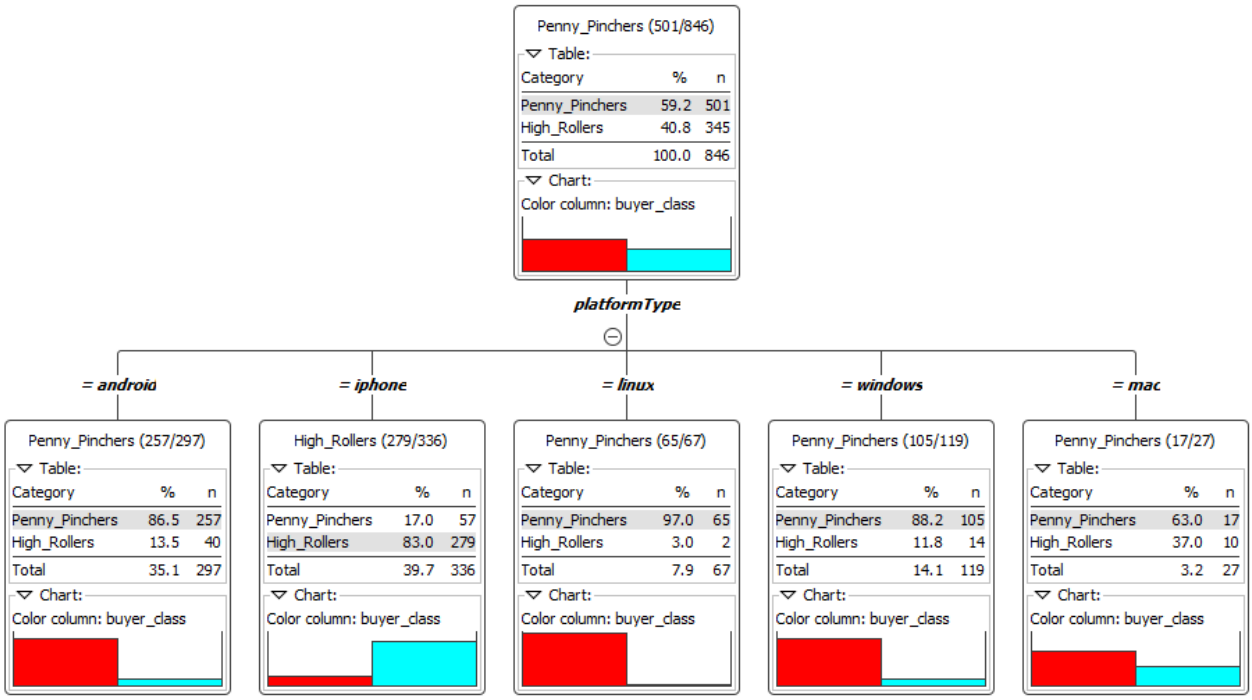
The training data set was used to create the decision tree model.

The trained model was then applied to the test dataset.

This is important because we could avoid overfitting by partitioning the data into training set and test set. If we use all the samples for training the model, the model might only perform well on the same set data but unable to achieve the same performance on unseen data. This would lead to overfitting. one should test one's model on a data set that was not used to train the model.

When partitioning the data using sampling, it is important to set the random seed because we would like to replicate the same partitioned datasets for multiple training and evaluation process. During this process, we would like the use the same partitioned datasets while evaluating the effect of different configurations of decision tree model on the classification performance.

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



2.3 Evaluation

A screenshot of the confusion matrix can be seen below:

buyer_class \ Prediction (buyer_class)	Penny_Pinchers	High_Rollers
Penny_Pinchers	308	27
High_Rollers	38	192

Correct classified: 500 | Wrong classified: 65

Accuracy: 88.496 % | Error: 11.504 %

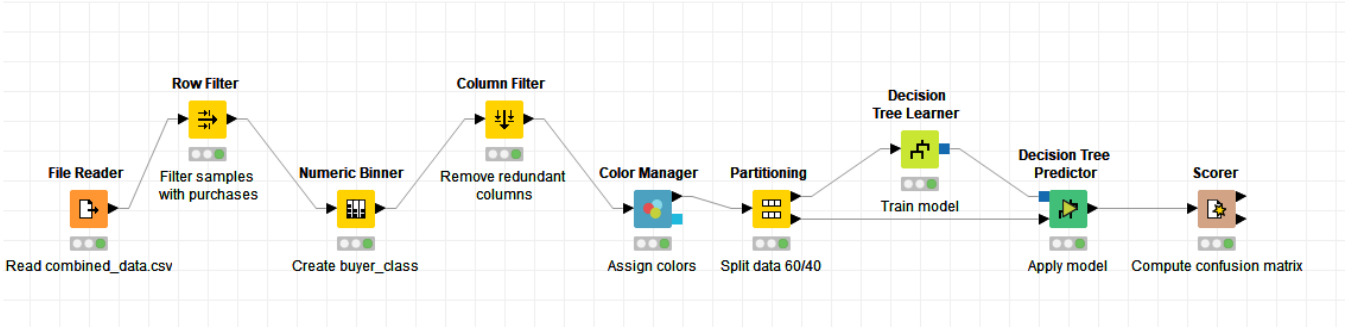
Cohen's kappa (κ) 0.76

As seen in the screenshot above, the overall accuracy of the model is 88.50%.

308 'Penny_Pincher' were correctly classified as 'Penny_Pincher', while the others 27 were incorrectly classified as 'High_Rollers'. On the other hand, 192 'High_Rollers' were correctly classified, while 38 were wrongly predicted as 'Penny_Pinchers'.

2.4 Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

- The platform used by the user could be used to determine whether the user is a HighRoller or a PennyPincher.
- For mobile platform, iPhone players are more likely to be HighRoller (83%), while Android players tend to be PennyPinchers (86.5%).

- Most of players on PC platform are generally PennyPinchers, but part of Mac users are HighRollers (37%).

Specific Recommendations to Increase Revenue
1. Eglence Inc. could consider promoting their game among iOS users as they are more likely to be HighRoller that would spend more than $5 on purchasing items.
2. They could also consider allocating more of their marketing budget to promote the game on the Mac platform because compared to Linux and Windows, there are more potential HighRoller players using Mac.

Part 3: Clustering Analysis

3.1 Attribute Selection

Attribute	Rationale for Selection
totalAdClicks	Total number of ad-clicks per user • This attribute is related to the advertisement revenue that brings profit to the company
totalBuyClicks	Total number of in-app purchase per user • This attribute is related to the revenue of the company generated from in-app purchase items.
totalRevenue	Total money spent on in-app purchase items per user

3.2 Training Data Set Creation

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

Create the training data set for clustering and show the first 5 rows

```
In [42]: trainingDF = combinedDF[['totalAdClicks','totalBuyClicks','totalRevenue']]
         trainingDF.head(n=5)
```

Out[42]:

	totalAdClicks	totalBuyClicks	totalRevenue
0	44	9	21.0
1	10	5	53.0
2	37	6	80.0
3	19	10	11.0
4	46	13	215.0

Show the dimension of the training data set

```
In [43]: trainingDF.shape
Out[43]: (543, 3)
```

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

# of clusters created: 3

3.3 Cluster Centers

Display the centers of three clusters formed

```
In [49]: print(my_kmmodel.centers)
[array([ 34.27777778,   6.45238095,  67.22222222]), array([  41.06666667,   10.28888889,  145.51111111]), array([ 26.2983
871,   4.4811828,  17.0672043])]
```

Cluster #	**Cluster Center** ['totalAdClicks', 'totalBuyClicks', 'totalRevenue']
1	[41.07, 10.29, 145.51]
2	[34.28, 6.45, 67.22]
3	[26.30, 4.48, 17.07]

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that the players in this cluster have the *highest* 'totalAdClicks', 'totalBuyClicks' and 'totalRevenue'.

Cluster 2 is different from the others in that the players in this cluster have the *second highest* 'totalAdClicks', 'totalBuyClicks' and 'totalRevenue'.

Cluster 3 is different from the others in that the players in this cluster have the *lowest* 'totalAdClicks', 'totalBuyClicks' and 'totalRevenue'.

3.4 Recommended Actions

Action Recommended	**Rationale for the action**
Increase the prices for advertisements Eglence Inc. shows to players belong to Cluster 1	• Players belong to Cluster 1 are frequent ad-clickers. • Increase the prices for advertisements targeting these players could potentially increase the company's revenue.
Charge players in Cluster 3 lower fees for the price of the in-app purchase items	• Players in Cluster 2 only purchase about 1.4 times more in-app items than players in Cluster 3, but the total amount of purchase is about 4 times more than players in Cluster 3. • This is perhaps due to players in Clusters 3 only purchase items with lower price. • Lowering the price of the in-app purchase items or giving

	discount to players in Cluster 3 could encourage them to spend more on more expensive items.

Part 4: Graph Analytics Analysis

4.1 Modeling Chat Data using a Graph Data Model

The graph model depicts the network based on chat interaction between 'Catch the Flamingo' players. A chat session can be initiated by a user and other users on the same team are free to join and leave the chat session afterwards. Interaction among users begins when a user created a post (chat item). Beside creating a new chat item in response to another user's chat item, a user can also mention another user in a chat item. All relationships between entities of this graph model are logged with a timestamp.

4.2 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

1) **chat_create_team_chat.csv**: userID, teamID, teamChatSessionID, timeStamp
2) **chat_join_team_chat.csv**: userID, teamChatSessionID, timeStamp
3) **chat_leave_team_chat.csv**: userID, teamChatSessionID, timeStamp
4) **chat_item_team_chat.csv**: userID, teamChatSessionID, chatItemID, timeStamp
5) **chat_mention_team_chat.csv**: chatItemID, userID, timeStamp
6) **chat_respond_team_chat.csv**: chatItemID_1, chatItemID_2, timeStamp

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

Consider the following load command:

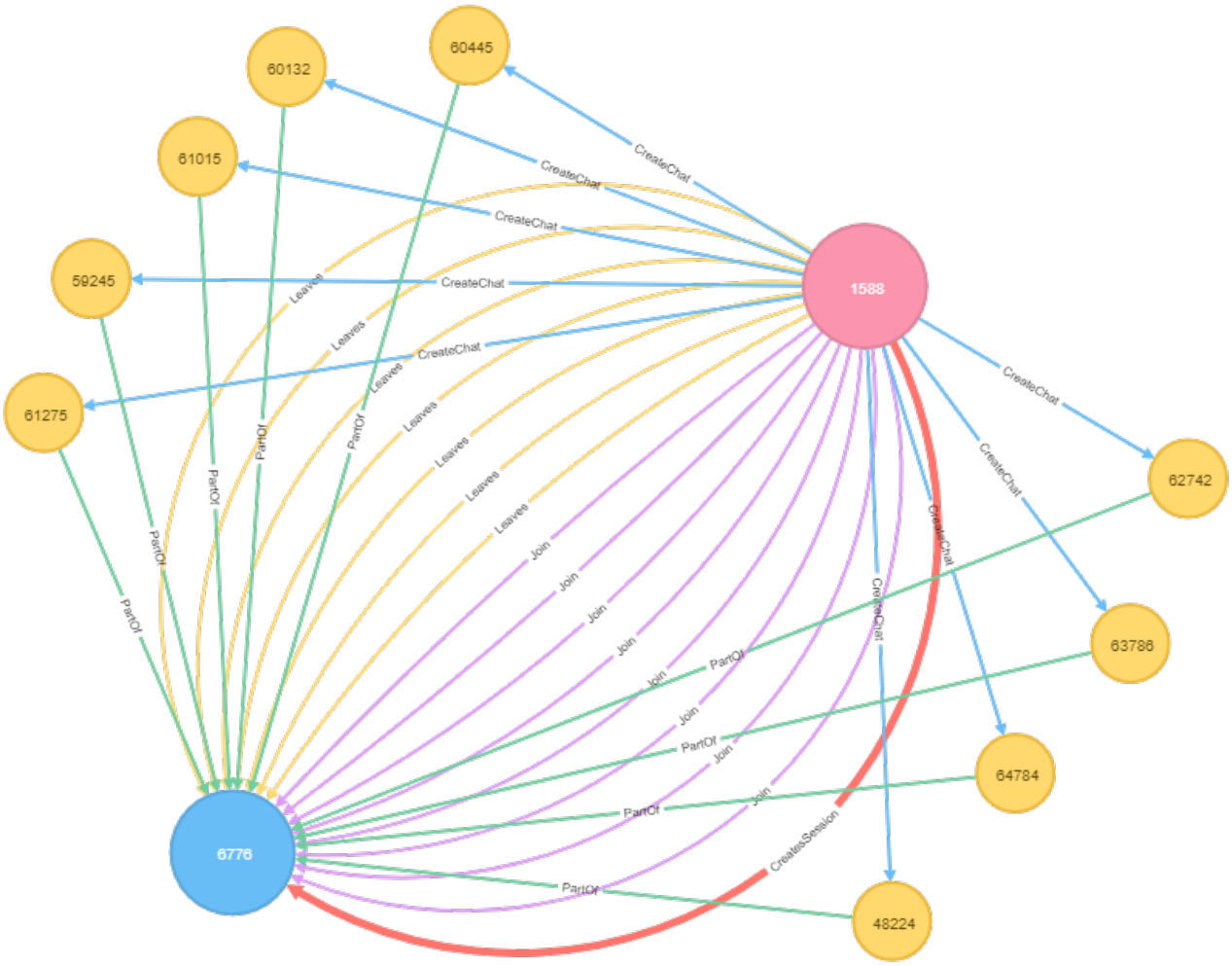
```
1 LOAD CSV FROM "file:///chat-data/chat_create_team_chat.csv" AS row
2 MERGE (u:User {id: toInteger(row[0])})
3 MERGE (t:Team {id: toInteger(row[1])})
4 MERGE (c:TeamChatSession {id: toInteger(row[2])})
5 MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c)
6 MERGE (c)-[:OwnedBy{timeStamp: row[3]}]->(t)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

Line 1: Loads the csv file from the specific location one row at a time

Line 2-4: Create User, Team and TeamChatSession nodes, with specific column value converted to an integer and used to populate the ID attribute.

Line 5-6: Create CreatesSession and OwnedBy edges by linking up the nodes created in previous lines. The edges have a property called timestamp which is filled by the content of specific column in the schema.

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.

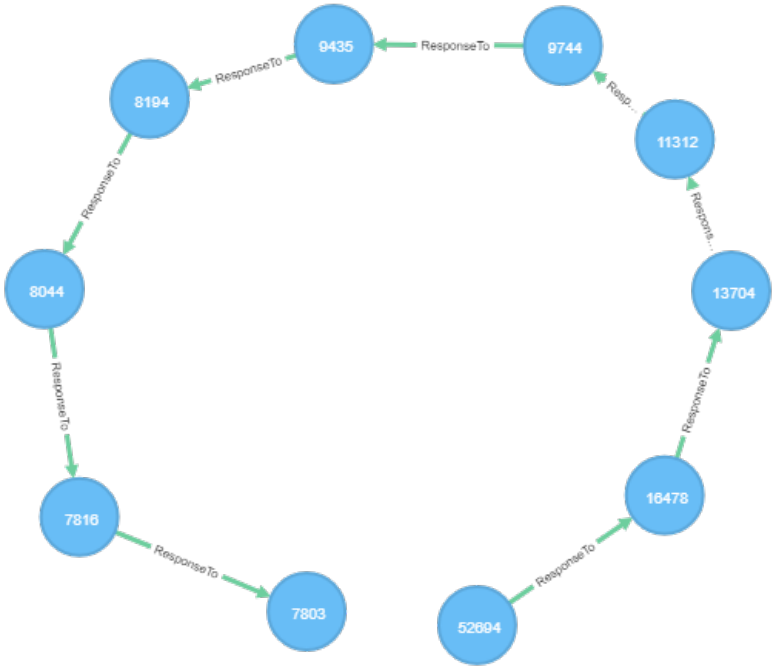


4.3 Finding the longest conversation chain and its participants

Find the longest conversation chain in the chat data using the "ResponseTo" edge label.

1) How many chats 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 of 9. Therefore 10 chats are involved in it.

2) How many users participated in this chain?
With 9 as the determined longest path, count the number distinct users who create ChatItem in this longest path:

```
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)
```

The answer returned by this query is 5.

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

1) Find the top 10 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

Chattiest Users

Users	Number of Chats
394	115
2067	111
209 or 1087	109

2) Find the top 10 chattiest teams

Match all ChatItems with a PartOf edge connecting them with a TeamChatSession node, with TeamChatSession nodes must have an OwnedBy edge connecting 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

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.

```
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
```

The above query uses the in information of the top 10 chattiest users and teams to investigate whether the top 10 chattiest users reside within the top 10 chattiest teams. The query returned only one result, where user ID 999 is one of the top 10 chattiest users and it also belongs to one of the top 10 chattiest teams, with team ID 52.

4.5 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.

1) Connect two users if one mentioned another user in a chat

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

2) Connect two users if one created a chatItem in response to another user's chatItem

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

3) Eliminate all self loops involving the edge "Interacts with"

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

4) Calculate the clustering coefficient of a node. The following query is an example of user ID 394:

```
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
```

Line 1: Get the of neighbors of the node (user ID 394) based on the "InteractsWith" edge
Line 2: k is the number of neighbors of the node
Line 3-4: For each of these neighbors, find the edges it has with the other members on the same list.
Line 5: Calculate the clustering coefficient by dividing the number of edges found in Line 3-4 by the number of neighbors found in Line 2.

5) Find the clustering coefficients of the top 3 chattiest users

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	0.9167
2067	0.7679
209	0.9524
1087	0.7667