Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: Bill Ostaski

Acquiring, Exploring and Preparing the Data

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	Ad Clicks Information 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	In-App Purchase Information 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	User Information 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 handle of the user **dob**: the date of birth of the user **country**: the two-letter country code where the user lives
team.csv	Team Information 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 **currentLevel**: the current level at which the team is playing
team-assignments.csv	Team Users Information 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	Team Level Information 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	User Session information	**userSessionid**: a unique id for the

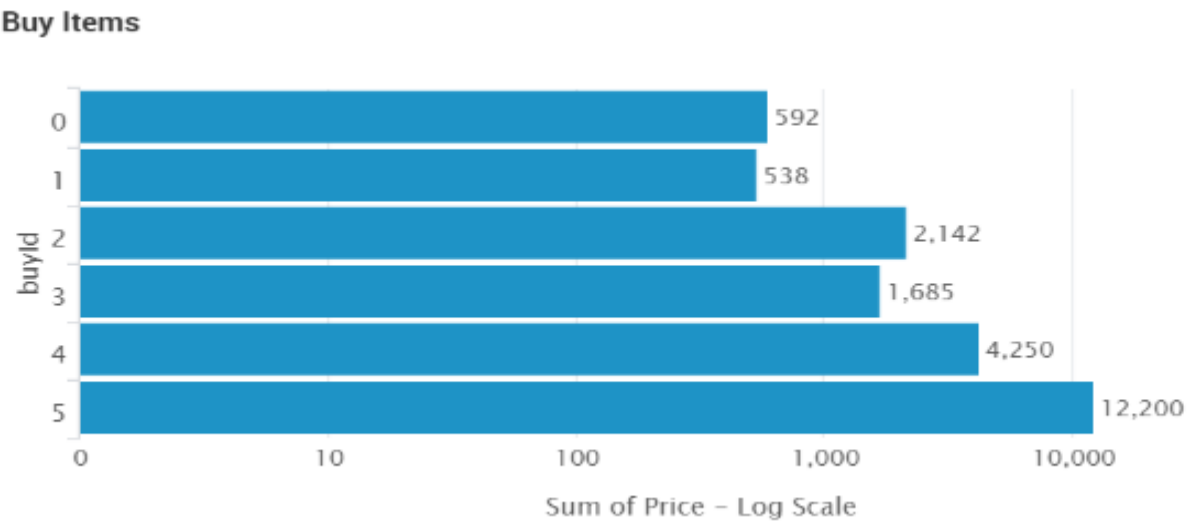
	Each line in this file describes a user session, which denotes when a user starts and stops playing the game. Additionally, when a team goes to the next level in the game, the session is ended for each user in the team and a new one started.	session **assignmentid**: the team assignment id for the user to the team **startTimeStamp**: a timestamp denoting when the session started **endTimeStamp**: a timestamp denoting when the session ended **team_level**: the level of the team during this session **platformType**: the type of platform of the user during this session
game-clicks.csv	User Click Information 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

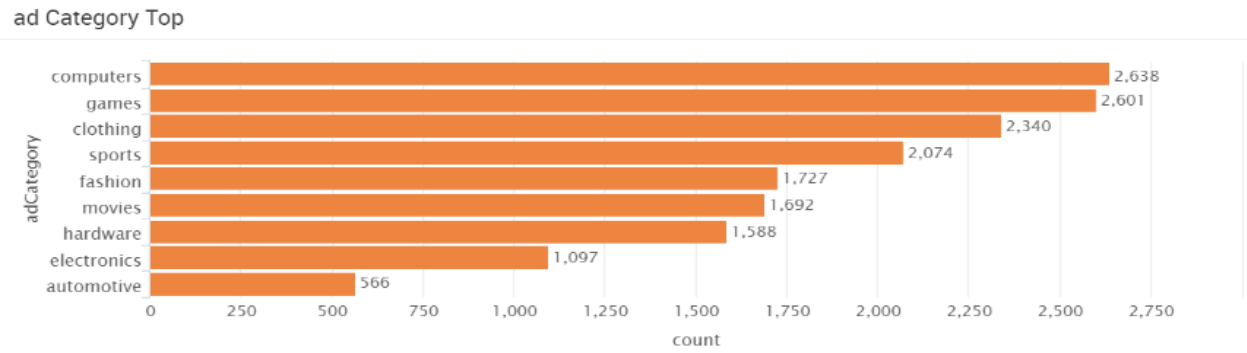
A histogram showing how many times each item is purchased:

A histogram showing how much money was made from each item:
The following Bar diagram shows the Revenues by Buy Items in a log scale.



Filtering

A histogram showing how many times each category of advertisement was clicked-on:



The following table shows the total amount of ad-click revenue for a set of specific values based on the advertisement category. All non-listed categories generate .25 revenue.

Scenario #	Electronics	Fashion	Automotive	Total Revenue
1 - even	0.50	0.50	0.50	3
2 - uneven	0.55	0.60	0.55	3,2

Data Classification Analysis

Data Preparation

Analysis of combined_data.csv

Sample Selection

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

Attribute Creation

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

Binned Data - 0:11 - Numeric Binner(Categorize HighRollers)

File

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

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
Row163	649	5811	1	linux	51	9	1	1	PennyPinchers
Row169	1958	5817	1	android	40	3	1	20	HighRollers
Row172	1300	5820	1	android	58	1	2	3	PennyPinchers
Row186	178	5834	1	iphone	54	6	1	20	HighRollers
Row196	670	5844	1	iphone	38	3	2	20	HighRollers
Row207	208	5855	1	iphone	32	3	1	20	HighRollers
Row210	157	5858	1	iphone	32	2	1	10	HighRollers
Row212	2221	5860	1	iphone	191	18	2	11.5	HighRollers
Row215	471	5863	1	iphone	45	6	2	15	HighRollers
Row218	1234	5866	1	android	46	3	1	10	HighRollers
Row222	371	5870	1	android	53	9	1	3	PennyPinchers
Row232	2146	5880	1	linux	46	7	1	2	PennyPinchers
Row239	935	5887	1	iphone	57	2	1	10	HighRollers
Row241	165	5889	1	iphone	49	3	1	5	PennyPinchers
Row244	1538	5892	1	iphone	24	3	1	20	HighRollers
Row245	1544	5893	1	iphone	36	6	2	20	HighRollers
Row261	2218	5909	1	android	80	6	1	3	PennyPinchers
Row262	1162	5910	1	windows	192	16	1	2	PennyPinchers
Row266	1821	5914	1	windows	178	22	1	1	PennyPinchers
Row271	2133	5919	1	android	87	14	1	3	PennyPinchers
Row272	1027	5920	1	iphone	52	5	3	15	HighRollers
Row273	518	5921	1	linux	121	16	1	1	PennyPinchers
Row282	2029	5930	1	iphone	89	7	1	10	HighRollers
Row286	2384	5934	1	windows	41	5	1	1	PennyPinchers
Row290	1155	5938	1	iphone	71	16	1	20	HighRollers
Row292	564	5940	1	linux	34	3	1	1	PennyPinchers
Row293	97	5941	1	android	74	10	1	1	PennyPinchers
Row297	253	5945	1	android	66	12	1	3	PennyPinchers
Row302	934	5950	1	windows	43	4	1	10	HighRollers
Row307	2009	5955	1	iphone	27	4	1	5	PennyPinchers
Row324	1815	6088	1	iphone	272	24	1	3	PennyPinchers
Row325	864	6173	1	android	79	9	1	3	PennyPinchers

The avg_price_binned attribute was created by binning on the avg_price attribute and distinguishes between “HighRollers” and “PennyPinchers.” HighRollers are buyers of items that cost more than $5.00. PennyPinchers are buyers of items that cost $5.00 or less.

The creation of this new categorical attribute was necessary because we want to separate players into two categories based on their purchasing behavior.

Attribute Selection

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

Attribute	**Rationale for Filtering**
userId	We are not analyzing individual users, so UserId is not relevant to this analysis.
userSessionId	We are not analyzing individual user sessions, so UserSessionId is not relevant to this analysis.
teamLevel	We are not analyzing team levels, so TeamLevel is not relevant to this analysis.
count_gameclicks	We are not analyzing game clicks, so count_gameclicks is not relevant to this analysis.
count_buyId	We are not analyzing buys, so count_buyId is not relevant to this analysis.

Data Partitioning and Modeling

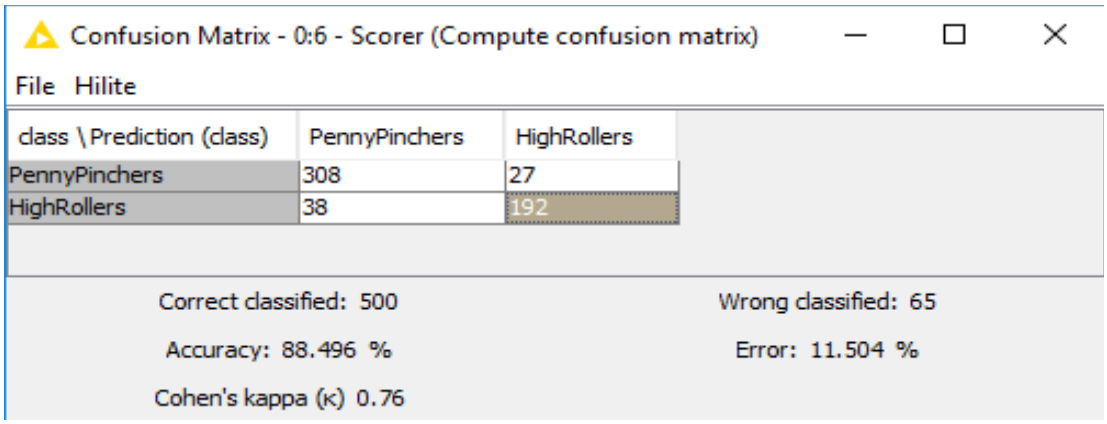
The data was partitioned into train and test datasets.
The train data set was used to create the decision tree model.
The trained model was then applied to the test dataset.
This is important because we want to test our model on data that was not used to train (i.e., create) it.

When partitioning the data using sampling, it is important to set the random seed was set because we want to get reproducible results.

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

Evaluation

A screenshot of the confusion matrix can be seen below:

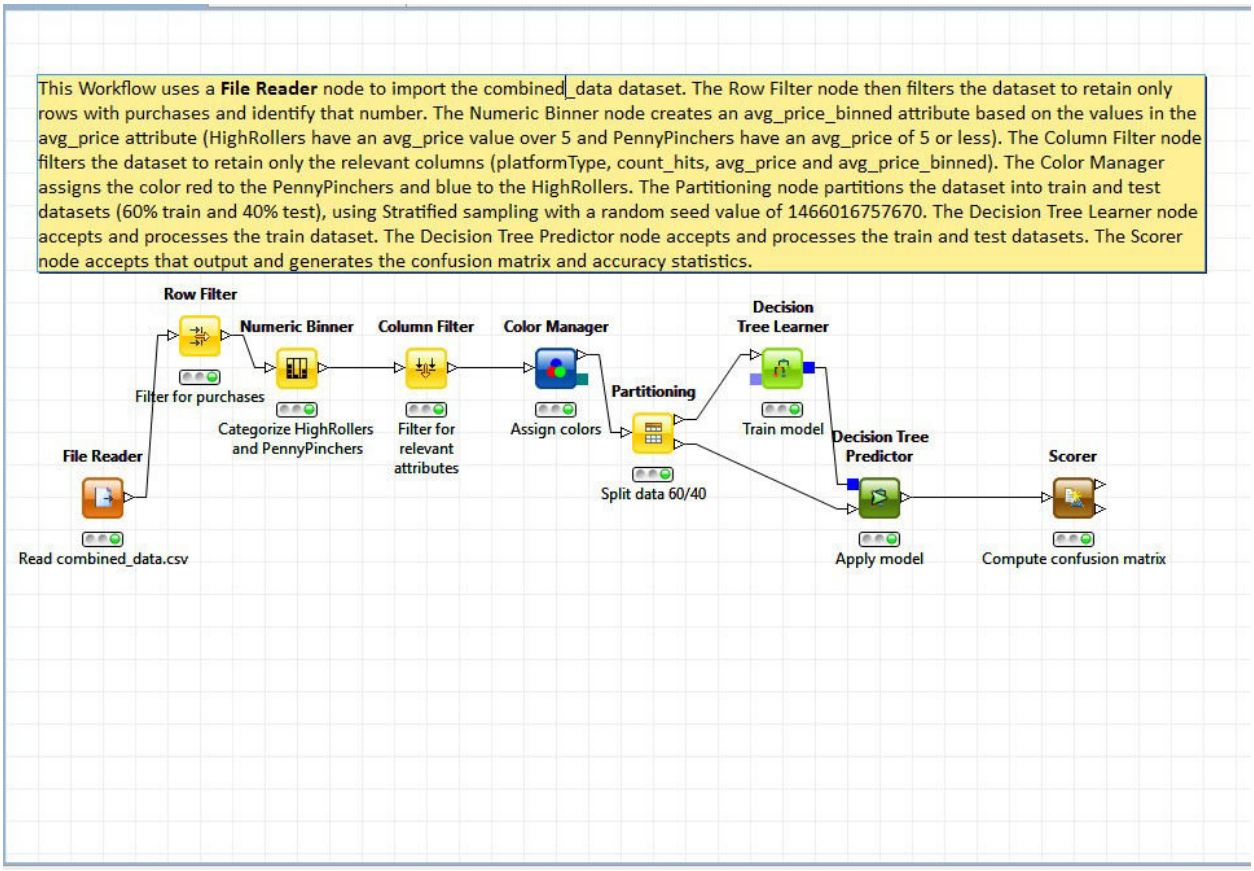


As seen in the screenshot above, the overall accuracy of the model is 88.496%.

308 PennyPinchers were correctly predicted as PennyPinchers.
27 PennyPinchers were incorrectly predicted as HighRollers.
192 HighRollers were correctly predicted as HighRollers.
38 HighRollers were incorrectly predicted as PennyPinchers.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller?

We defined a HighRoller as a user who buys items that cost more than $5.00 and a PennyPincher as a user who buys items that cost $5.00 or less.

Building on our observations from the Week1Technical Appendix, where we found the top 3 spenders use the iPhone platform, we can now confirm that the majority of HighRollers use the iPhone platform according to the Classified Data found in the Decision Tree Predictor.

Specific Recommendations to Increase Revenue
1. Given that the HighRollers typically use iPhones, it would be beneficial to offer more ads for iPhone-related products in general.
2. Assuming it is possible to generate targeted ads, offering ads for in-game purchases (e.g., binoculars) to players with low count_hits should increase revenue.

Clustering Analysis

Attribute Selection

Attribute	Rationale for Selection
adCount	This attribute is derived by adding a column to the adclicksDF generated by pd.read_csv('./ad-clicks.csv') and assigning a value of 1 to each row in order to count the number of ads each player has clicked on.
price	This attribute's values are parsed from buyclicksDF generated by pd.read_csv('./buy-clicks.csv') and is used to calculate the total revenue received from each player.
birthYear	This attribute is derived from the dob column values in usersDF generated by pd.read_csv('./users.csv') in order to determine the relative age of the players who have clicked on ads and made purchases.

Training Data Set Creation

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

	totalAdClicks	revenue		birthYear
0	44	21.0	1970-01-01	00:00:00.000001980
1	10	53.0	1970-01-01	00:00:00.000001994
2	37	80.0	1970-01-01	00:00:00.000001956
3	19	11.0	1970-01-01	00:00:00.000001997
4	46	215.0	1970-01-01	00:00:00.000001994

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

# of clusters created: 3

Cluster Centers

Cluster #	Cluster Center
1	array([41.39622642, 138.96226415, 1976.9245283])
2	array([25.68945869, 15.74643875, 1977.38746439])
3	array([34.09352518, 60.97122302, 1976.20863309])

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that it has the largest number of ad-clicks and revenue per user.

Cluster 2 is different from the others in that it has the lowest number of ad-clicks and revenue per user.

Cluster 3 is different from the others in that it has roughly the median number of ad-clicks and revenue per user. Cluster 3 also has a birthYear of 1977 and the other two has a birthYear of 1976, so this cluster centers on 39 year old players while the other clusters center on 40 year old players.

Recommended Actions

Action Recommended	**Rationale for the action**
Increase the prices for ads shown to frequent ad clickers.	Generally speaking, frequent ad clickers make more in-app purchases and therefore generate more revenue. Frequent ad clickers are more valuable to in-app retailers, so the price of the ads should reflect that value.
Charge higher fees for hosting the in-app purchase items shown to the higher revenue generating buyers.	Similarly, higher revenue generating buyers are more valuable to in-app retailers, so the hosting fees should reflect that value.
Focus ads and in-app purchase items that are more appealing to players in the 39-40 age group.	Overall, players in the 39-40 age group are the highest revenue generating buyers, so it would make sense to cater to this age group.

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

This graph model for chats is generated from 6 CSV files. It has nodes representing Users, Teams, TeamChatSessions and ChatItems. This graph also has relationships (edges) representing CreateSession, OwnedBy, Joins, Leaves, CreateChat, PartOf, Mentioned and ResponseTo.

Creation of the Graph Database for Chats

i) The schema for these 6 CSV files are:
chat_create_team_chat.csv: userId, teamId, teamChatSessionId and timestamp
chat_join_team_chat.csv: userId, TeamChatSessionId and timestamp
chat_leave_team_chat.csv: userId, TeamChatSessionId and timestamp
chat_item_team_chat.csv: userId, TeamChatSessionId, chatItemId and timestamp
chat_mention_team_chat.csv: chatItemId, userId and timestamp
chat_respond_team_chat.csv: response chatItemId, original chatItemId and timestamp

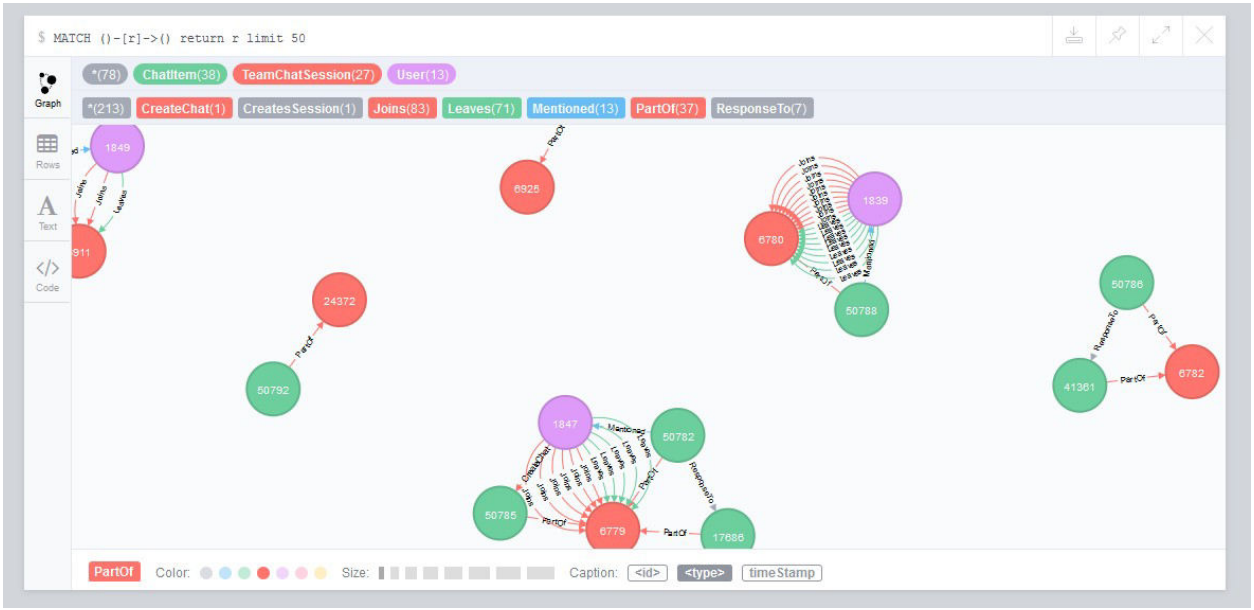
ii) The LOAD commands parse each CSV file and use MERGE to create a node or edge if it does not already exist. The MERGE command can be considered a combination of MATCH and CREATE. The LOAD command addressing chat_create_team_chat.csv MERGEs the userId, teamId and teamChatSessionId nodes and the CreatesSession and OwnedBy edges. The LOAD command addressing chat_join_team_chat.csv MERGEs the userId and TeamChatSessionId nodes and the Joins edges. The LOAD command addressing chat_leave_team_chat.csv MERGEs the userId and TeamChatSessionId nodes and the Leaves edges. The LOAD command addressing chat_item_team_chat.csv MERGEs the userId, TeamChatSessionId and chatItemId nodes and the CreateChat and PartOf edges. The LOAD command addressing chat_mention_team_chat.csv MERGEs the chatItemId and userId nodes and the Mentioned edges. The LOAD command addressing chat_respond_team_chat.csv MERGEs the response chatItemId and original chatItemId and the ResponseTo edges.

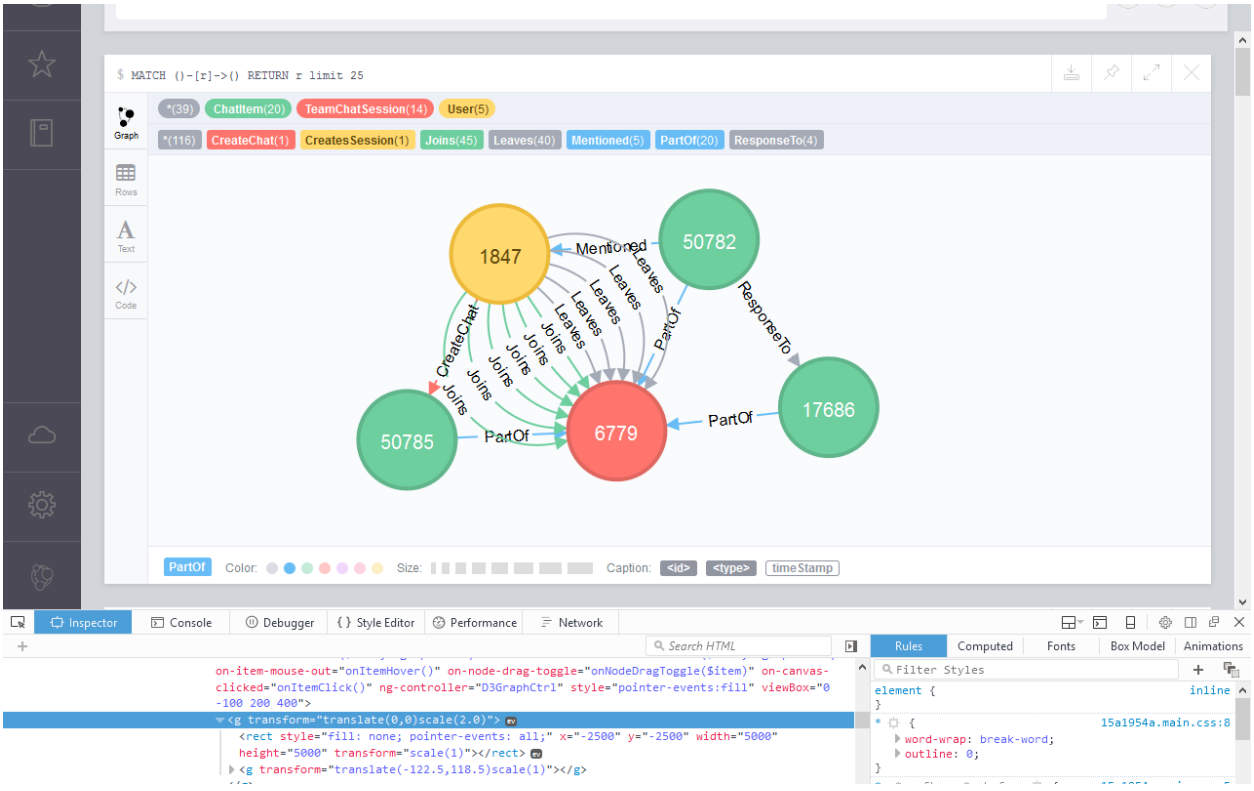
That last LOAD command is the most complex, as it also uses the WITH and AS keywords to distinguish between the response and original ChatItems without creating new nodes:

```
LOAD CSV FROM "file:///C:/chat-data/chat_respond_team_chat.csv" AS row
MERGE (i:ChatItem {id: toInt(row[0])})
WITH row, i AS response
MERGE (i:ChatItem {id: toInt(row[1])})
WITH row, response, i AS target
MERGE (response)-[:ResponseTo{timeStamp: row[2]}]->(target);
```

After executing these 6 LOAD statements, we have 45463 nodes and 118502 edges.

iii) The first graph displays examples of most node and edge types. The second graph displays the edges more clearly.





Finding the longest conversation chain and its participants

Part 1. We find the longest conversation chain in the chat data using the "ResponseTo" edge label, so the query is:

```
MATCH p=(a)-[:ResponseTo*]-(c)
RETURN length(p)
ORDER BY length(p) DESC LIMIT 1
```

This query returns:

```
length(p)
   11
```

The longest path length is 11.

Part 2. We then expand on the last query to find the number of unique users in the longest conversation chain using the CreateChat edge.

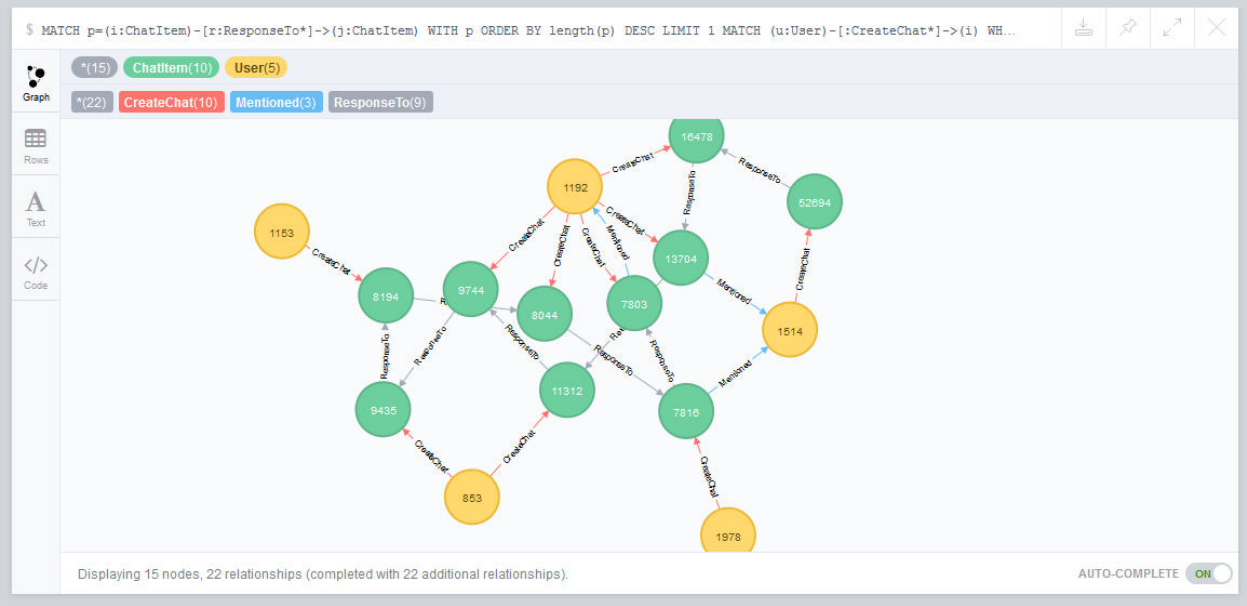
```
MATCH p=(i:ChatItem)-[r:ResponseTo*]->(j:ChatItem)
WITH p ORDER BY length(p) DESC LIMIT 1
MATCH (u:User)-[r:CreateChat*]->(i)
WHERE i IN nodes(p)
RETURN count(distinct u)
```

Returns count(distinct u): 5

We also want to count the number of items in the longest chat. The following query returns 10 ChatItems:

```
MATCH p=(i:ChatItem)-[r:ResponseTo*]->(j:ChatItem)
WITH p ORDER BY length(p) DESC LIMIT 1
MATCH (u:User)-[:CreateChat*]->(i)
WHERE i IN nodes(p)
RETURN i
```

We have 5 unique users and 10 ChatItems in the longest conversation chain. The following is the chart created.



This information is useful to Eglence Inc., as it identifies the players in the most active conversation. We will now address the most active players and teams. Eglence Inc. will likely want to address direct advertising to these players and groups since they may mention these products in future chats.

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

To find the chattiest users, we MATCH all Users with a CreateChat edge, return the Users' Ids, the count of the Users and sort them in descending order and limit the results to 10.

```
MATCH (u:User)-[:CreateChat]->(n:ChatItem)
RETURN u, count(n)
ORDER BY count(n) DESC LIMIT 10
```

Looking for the top 3 chattiest users reveals a tie for third place, with both UserID 209 and 1087 having count(n) values of 109, so we modified the query to LIMIT to 3 and got the following results.

Chattiest Users

Users	Number of Chats
394	115
2067	111
209 (and 1087)	109

To find the top 10 chattiest teams, we MATCH all ChatItems with a PartOf edge connecting them with a TeamChatSession node where the TeamChatSession nodes have an OwnedBy edge connecting them with any other node.

```
MATCH (t:Team)<-[r1:OwnedBy]-(c:TeamChatSession)
MATCH (u:User)-[r2:CreateChat]->(c:TeamChatSession)
RETURN t.id , c.id, COUNT(r2) AS InDegree
ORDER BY InDegree DESC
```

Chattiest Teams

Teams	Number of Chats
82	1324
185	1036
112	957

Though it is not readily apparent in the tables above, we did identify one user in the chattiest teams. UserID 999 is a member of TeamID 52.

How Active Are Groups of Users?

We compute an estimate of how “dense” the neighborhood of a node is. In the context of chats, that translates to how mutually interactive a certain group of users are.

First we construct the neighborhood of users. In this neighborhood, we will connect two users based on 2 conditions. The first condition is if one user mentioned another user in a chat:

```
MATCH (u1:User)--(i:ChatItem)-[:Mentioned]->(u2:User) CREATE (u1)-[r:InteractsWith]->(u2)
```

This MATCH statement creates 11084 InteractsWith relationships.

The second condition is if one user created a chatItem in response to another user’s chatItem:

```
MATCH (u1:User)--(i:ChatItem)-[:ResponseTo*]->(j:ChatItem)<--(u2:User) CREATE (u1)-[r:InteractsWith]->(u2)
```

This MATCH statement creates 11073 InteractsWith relationships.

We now have our neighborhood of users, complete with the InteractsWith relationships. However these InteractsWith relationships include users who have responded to their own ChatItems, so we need to eliminate all self-loops involving the InteractsWith relationships.

MATCH (u1)-[r:InteractsWith]->(u1) delete r

This MATCH statement deletes 4377 self-looping InteractsWith relationships.

We now create a scoring mechanism, called a clustering coefficient, to find users having dense neighborhoods. The score will range from 0 (a disconnected user) to 1 (a user in a clique – where every node is connected to every other node).

```
MATCH (d {id:<*USERID*>})-[:InteractsWith]-(b) WITH COLLECT(DISTINCT b.id) AS
neighbors, COUNT(DISTINCT b) AS degree, d.id AS id MATCH (n), (m) WHERE
(n.id IN (neighbors)) AND (m.id IN (neighbors)) RETURN id, SUM(CASE
WHEN (n)-[:InteractsWith]-(m) THEN 1 ELSE 0  END)*1.0/
(degree*(degree-1)) AS value
```

Where <*USERID*> is the UserID of the top 3 chattiest users.

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	1
2067	0,79
209	0,95

Recommended Actions

From our initial Data Exploration (see page 5), we identified the higher revenue generating product categories as Electronics, Fashion and Automotive. Presenting more ads for these categories may increase Eglence, Inc.'s total revenue.

Our Data Classification Analysis concluded with 2 recommendations (see page 10): 1) Given that the HighRollers typically use iPhones, it would be beneficial to offer more ads for iPhone-related products in general and 2) Assuming it is possible to generate targeted ads, offering ads for in-game purchases (e.g., binoculars) to players with low count_hits should increase revenue.

Our Clustering Analysis concluded with 3 recommendations (see page 12): 1) Increase the prices for ads shown to frequent ad clickers, 2) Charge higher fees for hosting the in-app purchase items shown to the higher revenue generating buyers and 3) Focus ads and in-app purchase items that are more appealing to players in the 39-40 age group.

From our Graph Analytics Analysis (see pages 16 and 17), we identified the Chattiest Users and Groups. Eglence, Inc. may want to consider focusing new ads to these users and groups since they will likely mention these ads and products in their chats.