Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: Rita Philavanh

April 7, 2018

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	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

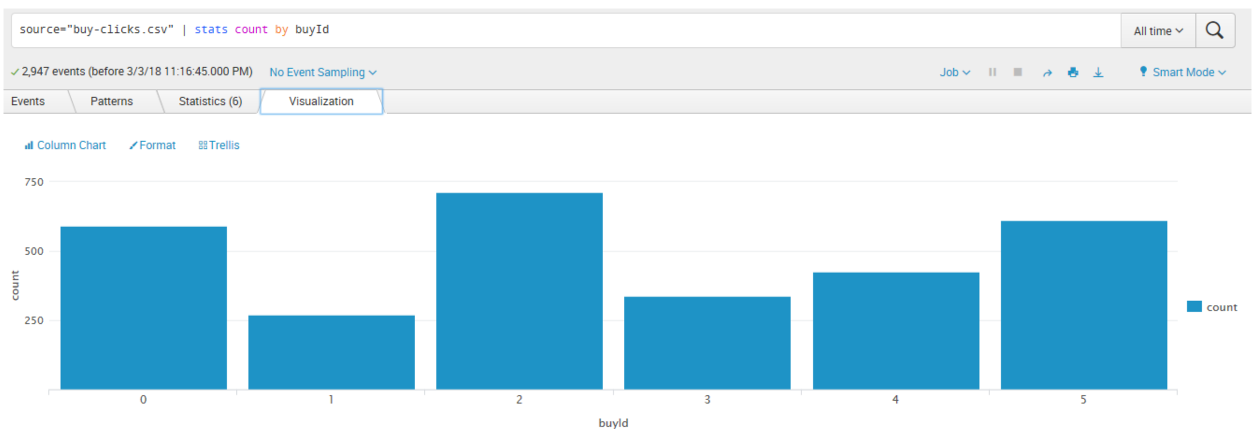
		assignmentId: a unique id for this assignment
level-events.csv	A line is added to this file each time a team starts or finishes a level in the game	timestamp: when the event occurred. eventId: a unique id for the event teamId: the id of the team teamLevel: the level started or completed eventType: the 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

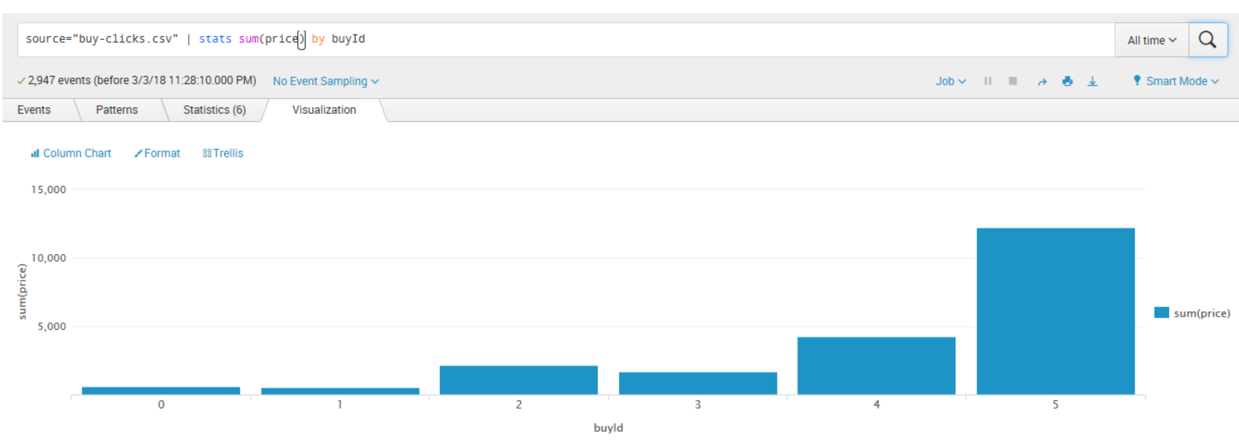
Aggregation

Amount spent buying items	21407.00
Number of unique items available to be purchased	6 items

A histogram showing how many times each item is purchased:

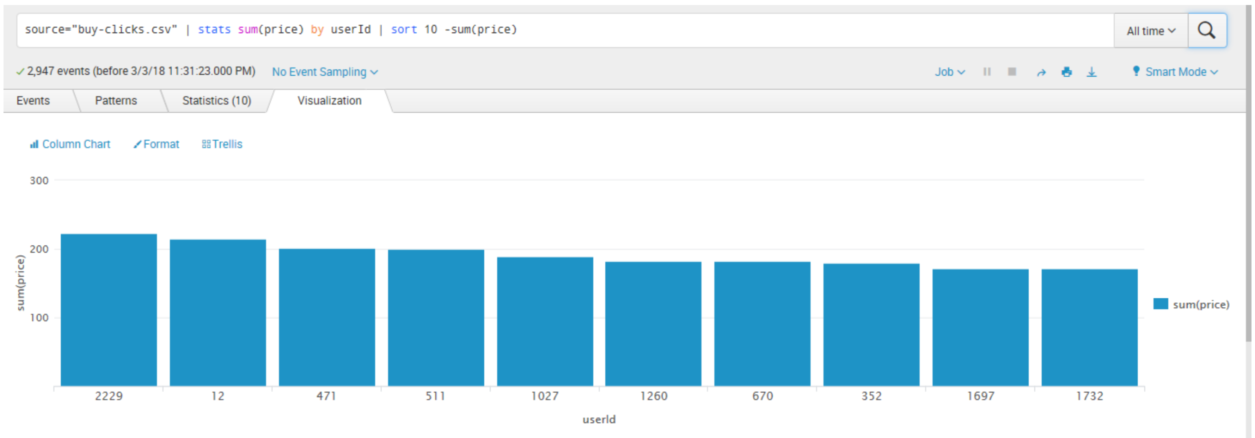


A histogram showing how much money was made from each item:



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	0.11596958174904944 = 11.6%
2	12	iphone	0.13068181818181818 = 13%
3	471	iphone	0.1450381679389313 = 14.5%

Data Preparation

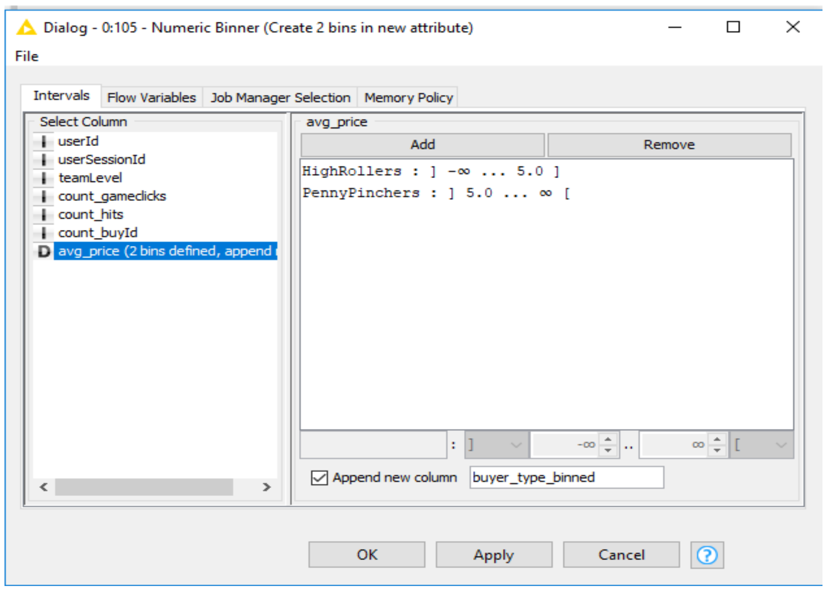
Analysis of combined_data.csv

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:



Description: buyer_type_binned is an attribute that categorizes the user into 2 bins based on it's avg_price value.

The creation of this new categorical attribute was necessary because we need to categorize users by the price of items they bought. HighRollers are users who bought items costing more than $5. PennyPinchers are users who bought items costing $5 or less.

Attribute Selection

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

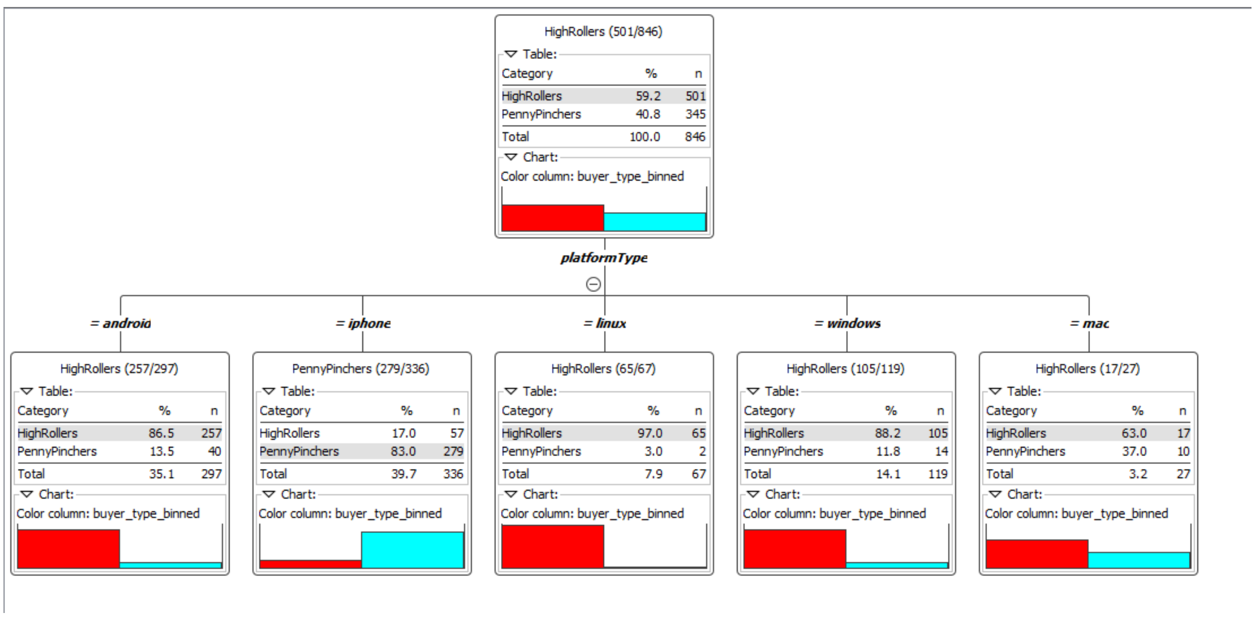
Attribute	Rationale for Filtering
userId	The id attribute is not relevant to classification analysis
userSessionid	The id attribute is not relevant to classification analysis
avg_price	This is the target of what is being predicted in the classification (buyer_type_bin), so we don't want to include it in the 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 evaluate the model on data it has not seen. So we train the model on a train set, and then evaluate it on a test set that the model did not see. This allows the accuracy results to be less bias.

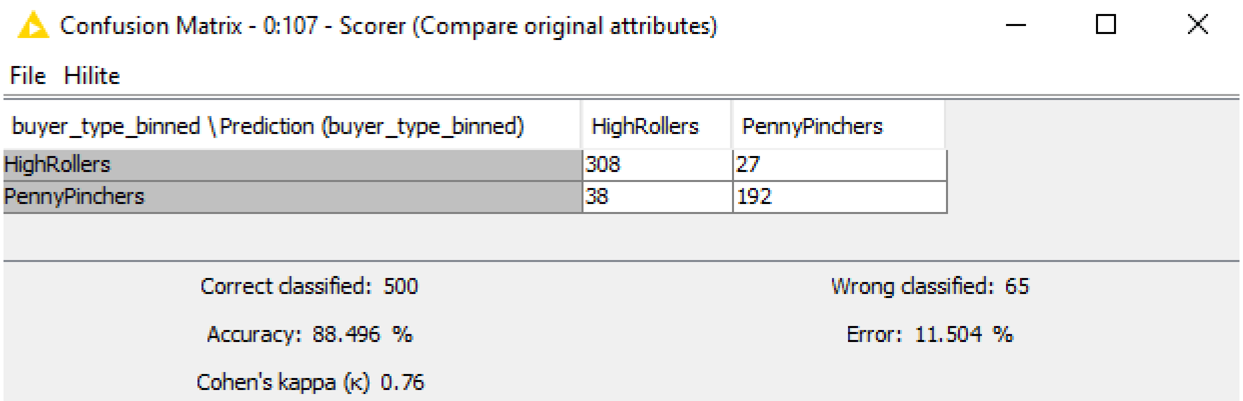
When partitioning the data using sampling, it is important to set the random seed because we want to be able to replicate the classification results we get. This allows us to tune other parameters without worrying if results could fluctuate based on the random sampling. So setting a random seed gives us the same random sampling each time

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



Evaluation

A screenshot of the confusion matrix can be seen below:



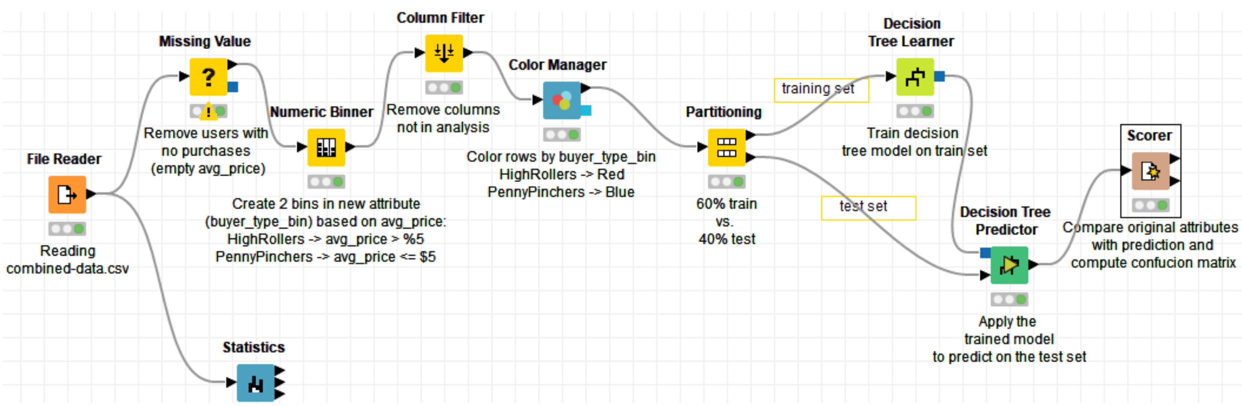
buyer_type_binned \ Prediction (buyer_type_binned)	HighRollers	PennyPinchers
HighRollers	308	27
PennyPinchers	38	192

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

308 samples that are suppose to be HighRollers have been **correctly** predicted as HighRollers.
192 samples that are suppose to be PennyPinchers have been **correctly** predicted as PennyPinchers.
27 samples that are suppose to be HighRollers have been **incorrectly** predicted as PennyPinchers.
38 samples that are suppose to be PennyPinchers have been **incorrectly** predicted as HighRollers.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?
From the decision tree results, users with iPhone Platform Types appear to be mostly PennyPinchers (at 83%). Users with other Platform Types (android, linix, windows,...) appear to be mostly HighRollers (~87.1%). So a user's Platform Type is an attribute that may be able to classify users as HighRoller vs PennyPincher.

Specific Recommendations to Increase Revenue
1. Focus marketing to target users on non iPhone platforms
2. Do a case study of iPhone users to determine when they DO spend money, and target that

Clustering Analysis

Attribute Selection

Attribute	Rationale for Selection
count_gameclicks	This attribute can provide information about how active the user is
count_hits	The attribute can provide information about a users skills
avg_price	The attribute, combined with the previous 2, can provide information if active or skilled users spend differently

Training Data Set Creation

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

```
df.show(5)
```

```
+----------------+----------+---------+
|count_gameclicks|count_hits|avg_price|
+----------------+----------+---------+
|              39|         0|      1.0|
|             129|         9|     10.0|
|             102|        14|      5.0|
|              39|         4|      3.0|
|              90|        10|      3.0|
+----------------+----------+---------+
only showing top 5 rows
```

Dimensions of the training data set (rows x columns) : 1411x3

# of clusters created: 2

Cluster Centers

```
centers=model.clusterCenters()
centers
```

```
[array([-0.31921137, -0.3124533 ,  0.02295286]),
 array([ 2.08938349,  2.04514884, -0.1502369 ])]
```

Cluster #	Cluster Center
1	[-0.31921137, -0.3124533, 0.02295286]
2	[2.08938349, 2.04514884, -0.1502369]

These clusters can be differentiated from each other as follows:

First number (field1) in each array refers to scaled version of the number of game clicks, the second number (field2) is the scaled version of the number of hits, and the third number (field3) is the scaled version of the average price spent per user.

Cluster 1 is different from the others in that… the users have both low game clicks and hits, but slightly higher money spent. This indicates less active users aren't focused on the gameplay, but may have spend a bit more.

Cluster 2 is different from the others in that… the users have both high game clicks and hits, but slightly lower money spent. This indicates active users are focused on mastering the gameplay, but may spend slightly less.

Recommended Actions

Action Recommended	Rationale for the action
Add price discounts for top users	In one cluster, players who had many gameclicks and hits spend slightly less. To encourage them to spend more, can offer discounts in price for top users.
Add ability to unlock special game features if a purchase is made	Since players who are active and focused on mastering the gameplay spend less, can include unlocking of special features in the game if a purchase is made. This will encourage those players who are active in game to make a purchase in order to unlock that feature

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

Chatting activities of active users can be captured by a graph data model. Users, Teams, TeamChatSession, and ChatItems are all entities that can be represented as nodes on the graph. Interactions between these nodes are represented by edges, which can have their own properties (such as timestamps, labels). For example, when a user creates a new chat with their team, a line (edge) is added the between the User node and the TeamChatSession node along with a timestamp. This also happens when a user joins or leaves a team, however, it will get a different edge label - like “Leaves” for users leaving a TeamChatSession).

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

CSV File	**Description**	**Columns**
chat_create_team_chat.csv	A line is added to this file when a player creates a new chat with their team.	userid, teamid, TeamChatSessionID, timestamp
chat_item_team_chat.csv	Creates nodes labeled ChatItems. Also create an edge labeled "PartOf" from the ChatItem node to the TeamChatSession node. This edge should also have the same timeStamp property.	userid, teamchatsessionid, chatitemid, timestamp
chat_join_team_chat.csv	Creates an edge labeled "Joins" from User to TeamChatSession. The columns are the User id, TeamChatSession id and the timestamp of the Joins edge.	userid, TeamChatSessionID, teamstamp
chat_leave_team_chat.csv	Creates an edge labeled "Leaves" from User to TeamChatSession. The columns are the User id, TeamChatSession id and the timestamp of the Leaves edge.	userid, teamchatsessionid, timestamp
chat_mention_team_chat.csv	Creates an edge labeled "Mentioned". Column 0 is the id of the ChatItem, column 1 is the id of the User, and column 2 is the timeStamp of the edge going from the chatItem to the User.	ChatItem, userid, timeStamp
chat_respond_team_chat.csv	A line is added to this file when player with chatid2 responds to a chat post by another player with chatid1.	chatid1, chatid2,timestamp

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

To load a dataset from a csv, the path to the file has to be defined first. This will read the csv file one row at a time. In the command, each column has to be defined as a node or edge (with optional properties). The 2 nodes that an edge connects to should also be defined. As

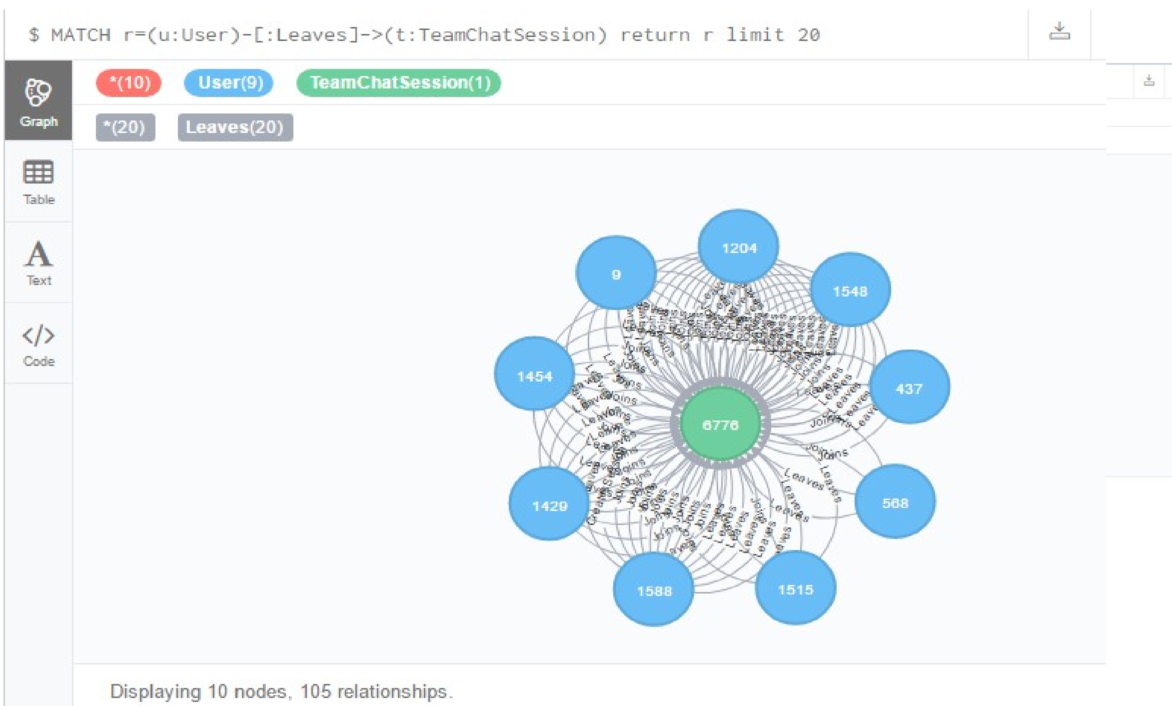
an example, take the chat_create_team_chat.csv file which contains these 4 columns:
`column 0: userid, column 1: teamid, column 2: TeamChatSessionID, column3: timestamp`

The LOAD command for this csv will look like this:

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

So this command defines 3 nodes:
A "User (u)" node having an "id" property taken from column 0 of the csv
A "Team (t)" node having an "id" property taken from column 1 of the csv
A "TeamChatSession (c)" node having an "id" property taken from column 2 of the csv
A "CreatesSession" edge with a "timeStamp" property taken from column 3, and connecting node (u) and (t)
A "OwnedBy" edge with a "timeStamp" property taken from column 3, and connecting node (c) and (t)

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

Longest length conversation (path length): **9**

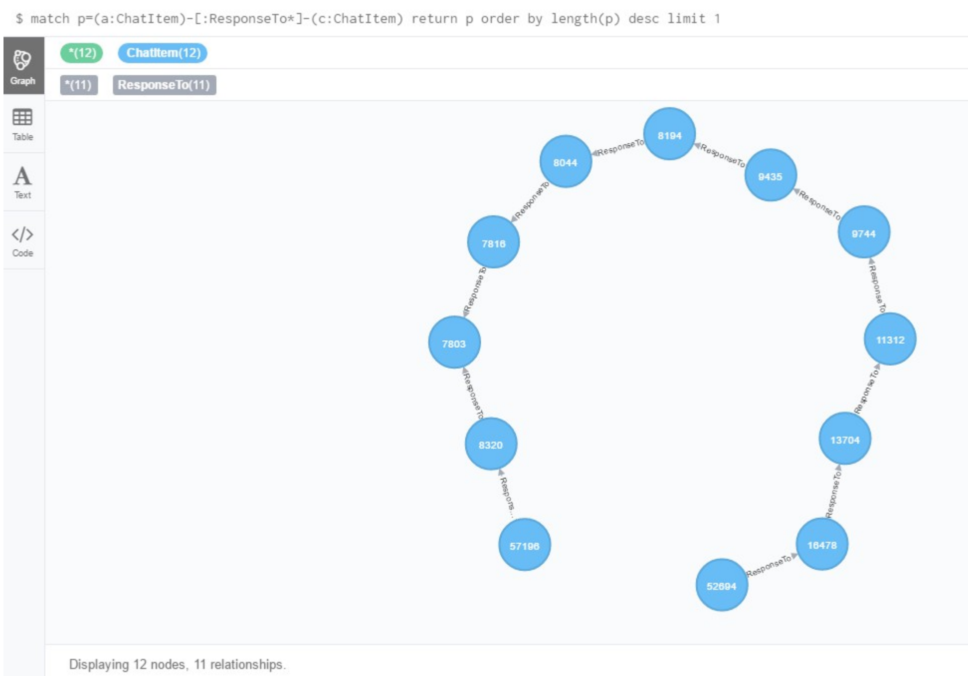
Command used:

```
match p=(a:ChatItem)-[:ResponseTo*]-(c:ChatItem)
return p order by length(p) desc limit 1
```

How many unique users were part of the conversation chain: **5**

Command used:

```
match p=(a:ChatItem)-[:ResponseTo*]-(c:ChatItem) where length(p) =9
with p
match (u:User)-[:CreatesChat*]-(c:ChatItem)
return count(distinct u)
```



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

To find chattiest users:

```
match (u:User)-[:CreatesChat*]->(i:ChatItem) return u, count(u) order by count(u) desc limit 10
```

To find chattiest teams:

```
match (i:ChatItem)-[:PartOf*]->(c:TeamChatSession)-[:OwnedBy*]-(t:Team) return t, count(t) order by
count(t) desc limit 10
```

To find which top 10 chattiest users are in the top 10 chattiest teams:

```
match (u:User)-[:Joins*]->(c:TeamChatSession)-[:OwnedBy*]-(t:Team) where (u.id=394 or u.id=2067 or
u.id=209 or u.id=1087 or u.id=554 or u.id=516 or u.id=1627 or u.id=999 or u.id=461 or u.id=668) and
(t.id = 82 or t.id=185 or t.id=112 or t.id = 18 or t.id=194 or t.id=129 or t.id = 52 or t.id=136 or
t.id=146 or t.id=81)  return u,t
```

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams

Teams	Number of Chats
82	1324
185	1036
112	957

Identify and report whether any of the chattiest users were part of any of the chattiest teams.
Top 10 chatty User id **999** is in Top 10 chatty Team **52**

This information can be useful to identify whether engaged users in a team behave differently.

How Active Are Groups of Users?

Describe your steps for performing this analysis. Be as clear, concise, and as brief as possible.
To create relationship where one user mentioned another user in a chat:

```
match (u1:User)-[:CreatesChat*]->(i:ChatItem)-[:Mentioned]->(u2:User) with distinct u1, u2
create (u1)-[:InteractsWith]->(u2)
```

To create relationship where one user created a chatItem in response to another user's chatItem:

```
match (u1:User)-[:CreatesChat]->(i1:ChatItem)<-[:ResponseTo]-(i2:ChatItem)<-[:CreatesChat*]->(u2:User)
with distinct u1, u2
create (u1)-[:InteractsWith]->(u2)
```

To find neighbors (k) for top 3 chatty Users:

```
match (u1:User {id:<id>})-[:InteractsWith]-(u2:User)with collect(distinct u2.id) as neighbors
return neighbors
```

id= 394: k=3 [1997,2011,1012]
id= 2067: k=8 [1265,1672,1627,697,516,2096,63,209]
id=209: k=7 [63,1672,1265,1627,516,2067,2096]

To find edges, run command on between each neighbors id:

```
match (u1:User {id:<id>})-[r]-(u2:User) where u2.id in [neighbors]
return count(r)
```

id=394: e=10
id= 2067: e=60
id=209: k=12 60

Finally, report the top 3 most active users in the table below.

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	10/(3*2) = 1.67
2067	60/(8*7) = 1.07
209	60/(7*6) = 1.43

Recommended Actions

There are a couple recommendations/actions I would like to make to Eglence, Inc. to improve their business and increase revenue.

1. From the initial data exploration, considering item 2 is most purchased, yet made little money, raising the price of item 2 could lead to increased profits.
2. From the classification analysis, users with iPhones tend to spend less money. More marketing geared towards iPhone users can capture revenue from this untapped market.
3. From the cluster analysis, top engaged players spend slightly less. To encourage them to spend more (while also encouraging engagement in the gameplay), discounts can be offered to players with the top scores.