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	record when a player clicks the ad	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	record when a player makes an in-app purchase	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	Users 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	Teams 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	Record when an user joins a team, at most one 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

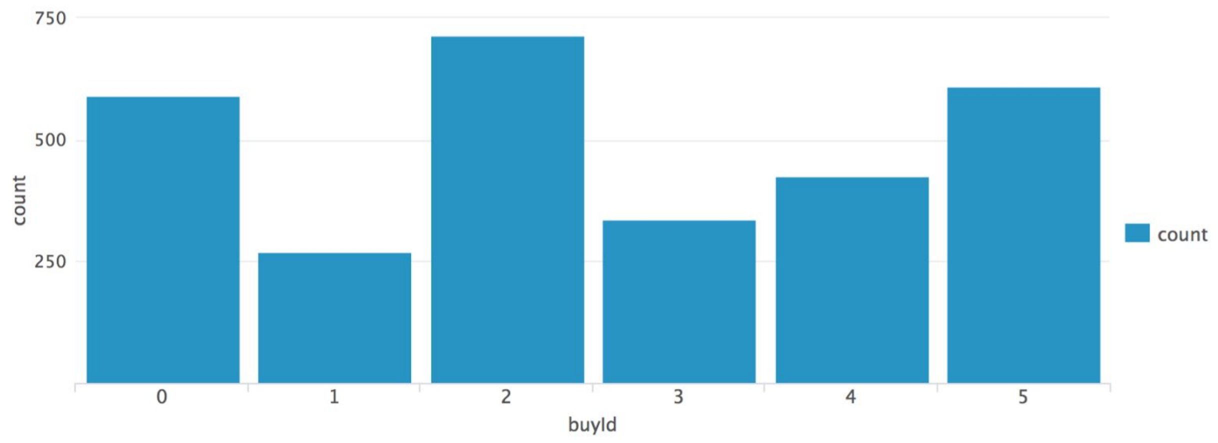
		assignment
level-events.csv	When a team starts or finishes a level	time: when the event occurred. eventid: a unique id for the event teamid: the id of the team level: the level started or completed eventType: the type of event, either start or end
user-session.csv	Record when a user starts and stops in the game. When a team goes to next level all the team user sessions end and new ones start.	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	Record when a user clicks 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
<Fill In>	<Fill in short phrase>	<Fill In: Name and describe all fields>

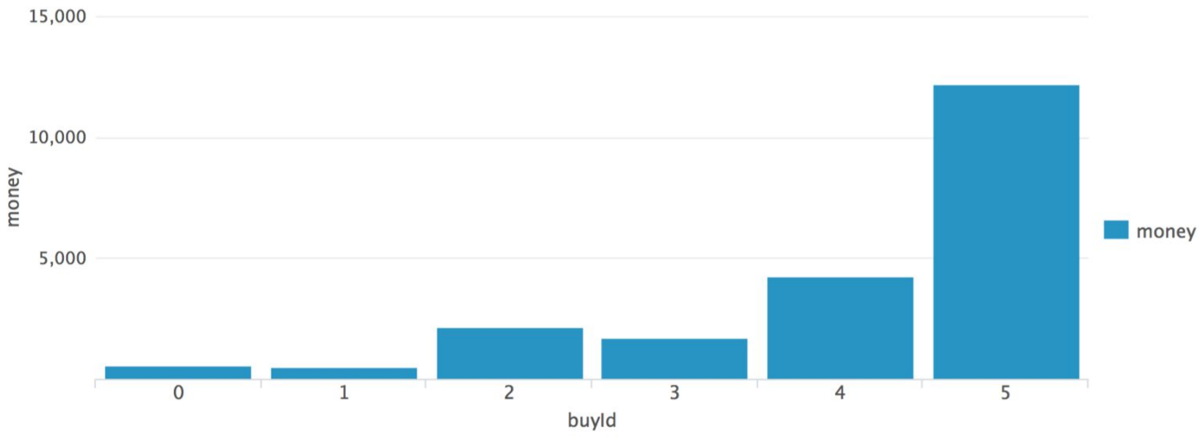
Aggregation

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

A histogram showing how many times each item is purchased:

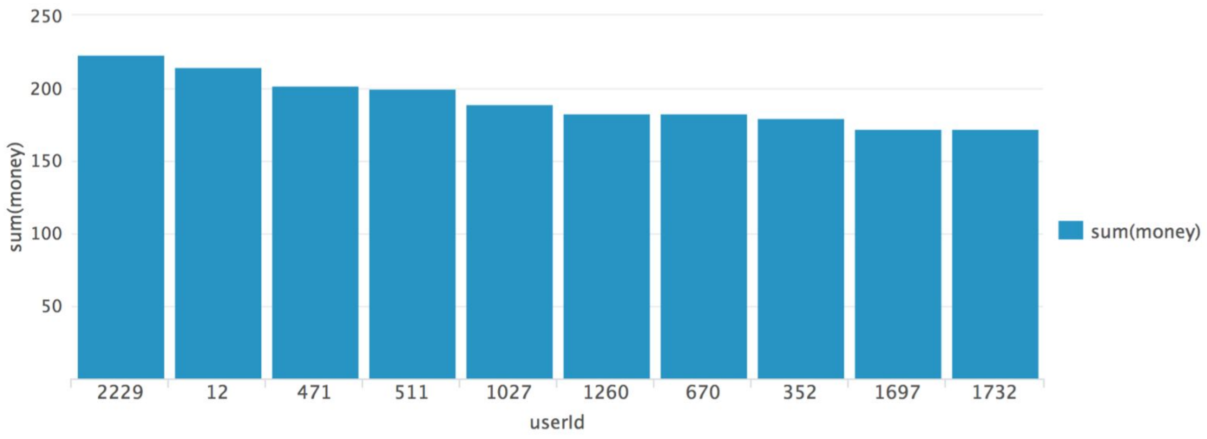


A histogram showing how much money was made from each item:



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	11.597%
2	12	iphone	13.0682%
3	471	iphone	14.5038%

Data Preparation

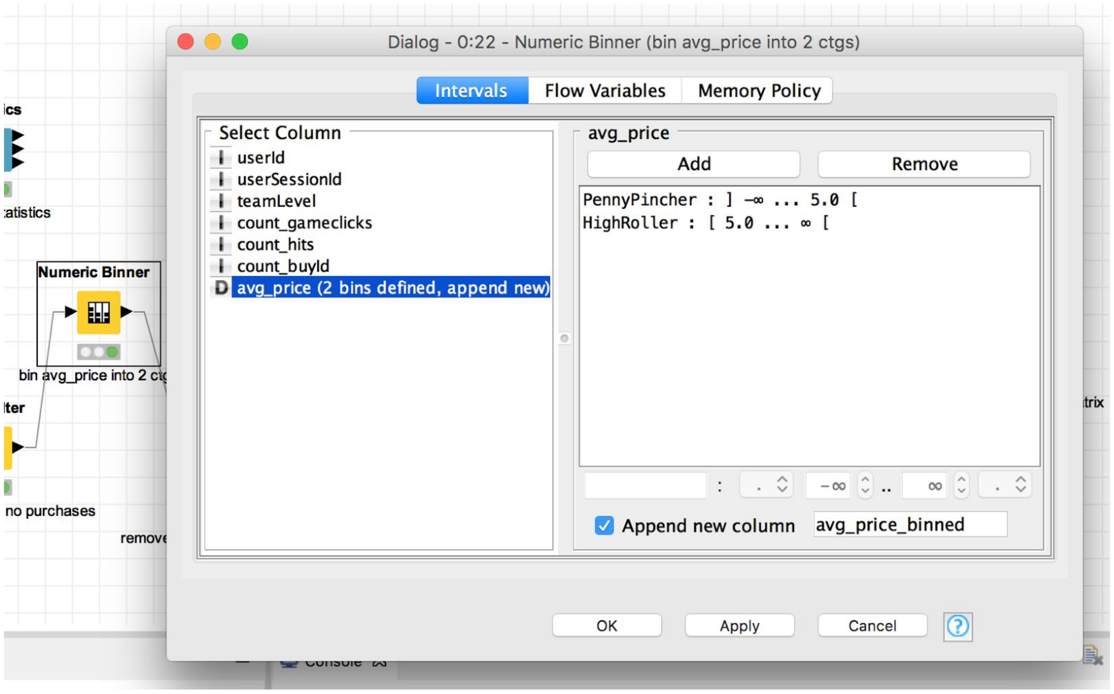
Analysis of combined_data.csv

Sample Selection

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

Attribute Creation

A new categorical attribute was created based on categorization of avg-price into 2 bins to enable analysis of players as broken HighRollers and PennyPinchers. A screenshot of the attribute follows:



Select Column
- userId
- userSessionId
- teamLevel
- count_gameclicks
- count_hits
- count_buyId
- avg_price (2 bins defined, append new)

avg_price

```
PennyPincher : ] −∞ ... 5.0 [
HighRoller : [ 5.0 ... ∞ [
```

☑ Append new column: avg_price_binned

New column named “avg_price_binned” is the new attribute where buyid > 5 belongs to “HighRollers” because the prices of them are over $5, while buyid <=5 belongs to “PennyPinchers” because the prices of those are not over $5.

The creation of this new categorical attribute was necessary because this is a classification problem, we should not use a continuous value field like avg-price.

Attribute Selection

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

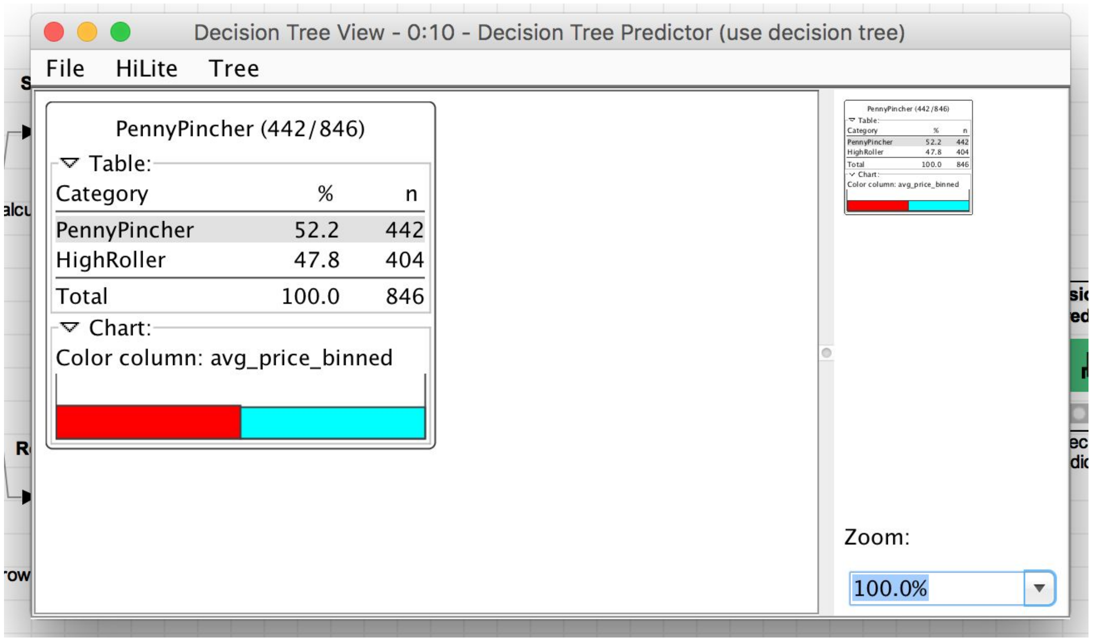
Attribute	**Rationale for Filtering**
avg_price	We don’t need the average price anymore since we have a new categorized attribute avg_price_binned based on this.
user_Id	Don’t need this since it’s just a computer generated number
user_Session_Id	Don’t need this since it’s just a computer generated number

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 when we do data analysis, we should test our model on a data set that was not used to train the model. After a model has been processed by using the training set, you test the model by making predictions against the test set.

When partitioning the data using sampling, it is important to set the random seed to make sure the partition is the same every time you run the program. That is needed when you need a reproducible result.

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


Decision Tree View - 0:10 - Decision Tree Predictor (use decision tree)
File
HiLite
Tree
Zoom:
100.0%

PennyPincher (442/846)

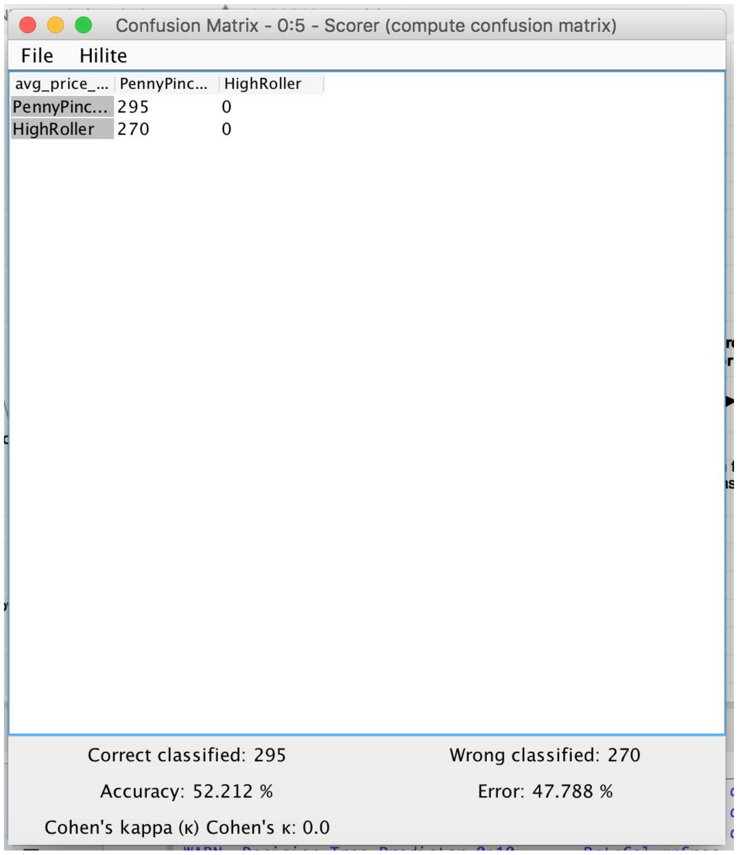
Table:

Category	%	n
PennyPincher	52.2	442
HighRoller	47.8	404
Total	100.0	846

Chart:
Color column: avg_price_binned

Evaluation

A screenshot of the confusion matrix can be seen below:



avg_price_...	PennyPinc...	HighRoller
PennyPinc...	295	0
HighRoller	270	0

Correct classified: 295 Wrong classified: 270

Accuracy: 52.212 % Error: 47.788 %

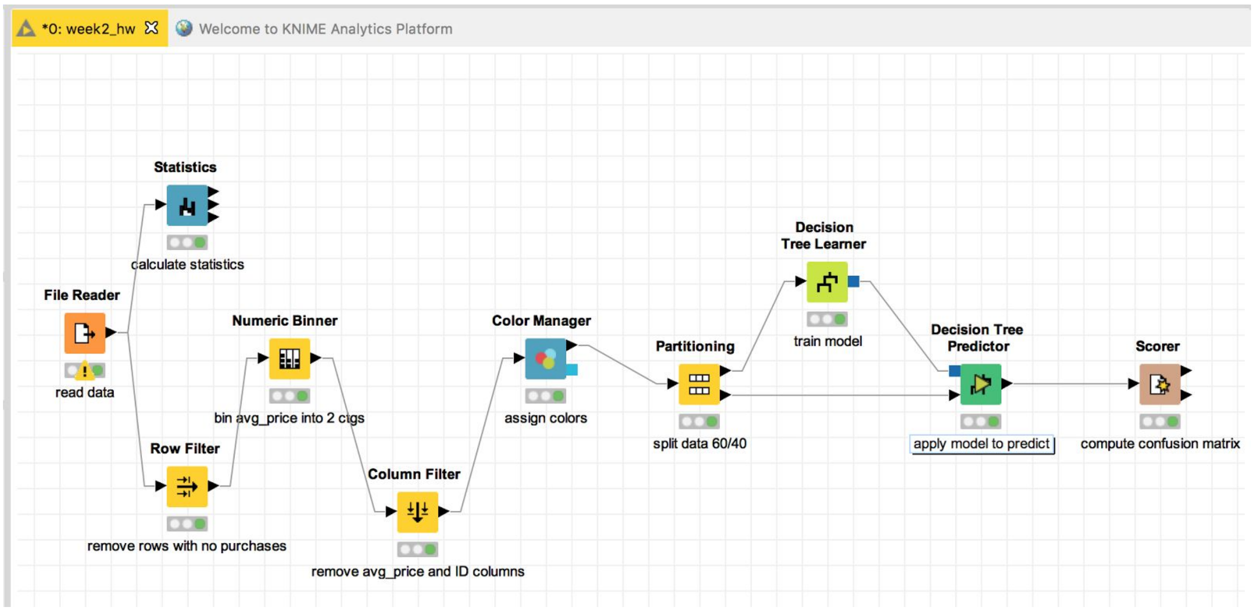
Cohen's kappa (κ) Cohen's κ: 0.0

As seen in the screenshot above, the overall accuracy of the model is 52.212%

0 HighRollers have been predicted correctly.
270 HighRollers have been predicted incorrectly.
295 PennyPinchers have been predicted correctly.
0 PennyPinchers have been predicted incorrectly.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

iPhone users are HighRollers and other platformType users are PennyPinchers.

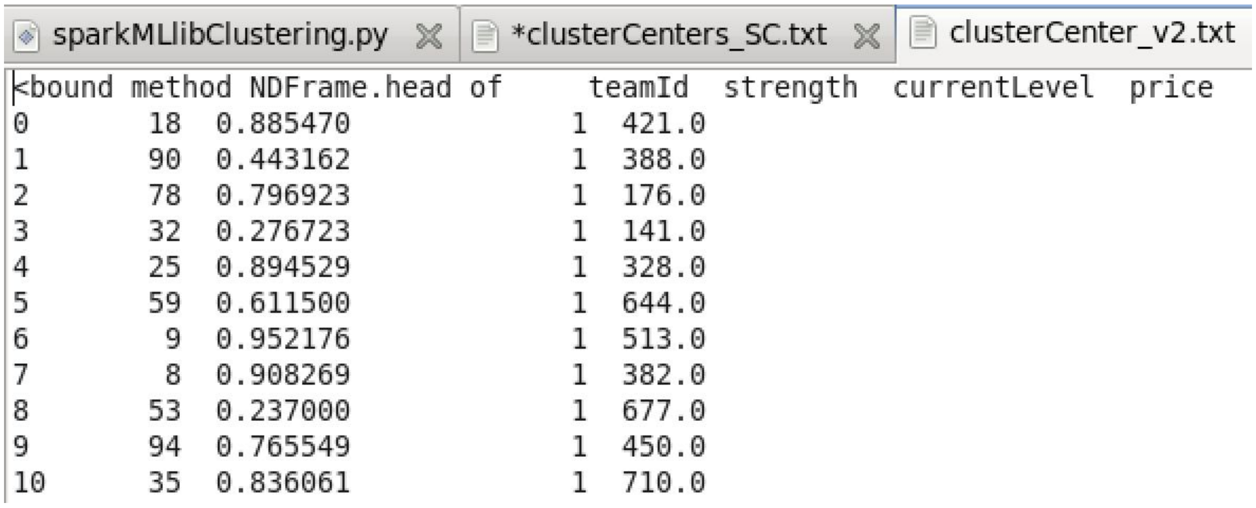
Specific Recommendations to Increase Revenue
1. Show more ads to iphone users
2. Increase ads price for iphone platform device

Attribute Selection

Attribute	**Rationale for Selection**
team strength	players with different strength might behave differently
revenue	money is the ultimate goal
team current level	similar to player strength and i want to see if there's any connection between team level and revenue

Training Data Set Creation

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



```
<bound method NDFrame.head of      teamId  strength  currentLevel  price
0       18  0.885470             1  421.0
1       90  0.443162             1  388.0
2       78  0.796923             1  176.0
3       32  0.276723             1  141.0
4       25  0.894529             1  328.0
5       59  0.611500             1  644.0
6        9  0.952176             1  513.0
7        8  0.908269             1  382.0
8       53  0.237000             1  677.0
9       94  0.765549             1  450.0
10      35  0.836061             1  710.0
```

Dimensions of the training data set (rows x columns) : 44 x 4

# of clusters created: 3

Cluster Centers

Cluster #	Cluster Center
1	3.71972897e-01, 1.00000000e+00, 7.01875000e+02
2	0.55483945, 1. , 177.05882353
3	0.54852125, 1. , 418.47368421

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that teams with less strength spend more money in in-app purchases.

Cluster 2 is different from the others in that teams with more strength spend less money in in-app purchases.

Cluster 3 is different from the others in that it's showing the separation right between cluster 1 and cluster 2.

Recommended Actions

Action Recommended	**Rationale for the action**
increase in-app ad fees at lower level games	teams with lower strength tend to spend more on purchases
reduce in-app ads at higher level games	teams with higher strenger playing in high-level games tend to spend less in buying stuff

<u>Graph Analytics</u>

Modeling Chat Data using a Graph Data Model

Use Neo4j graphics to model chat data and perform analysis of relationships between teams, users, and chats.

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

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

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.

Chat_create_team_chat.csv : when a user create a team chat session

Chat_join_team_chat.csv : when a user joins a team chat session

Chat_leave_team_chat.csv : when a user leaves a team chat session

Chat_item_team_chat.csv : when a new chat is created in a team chat session

Chat_mention_team_chat.csv : when a user mentions a specific chat item

Chat_respond_team_chat.csv : when a user responds a chat to another user

Sample LOAD script:

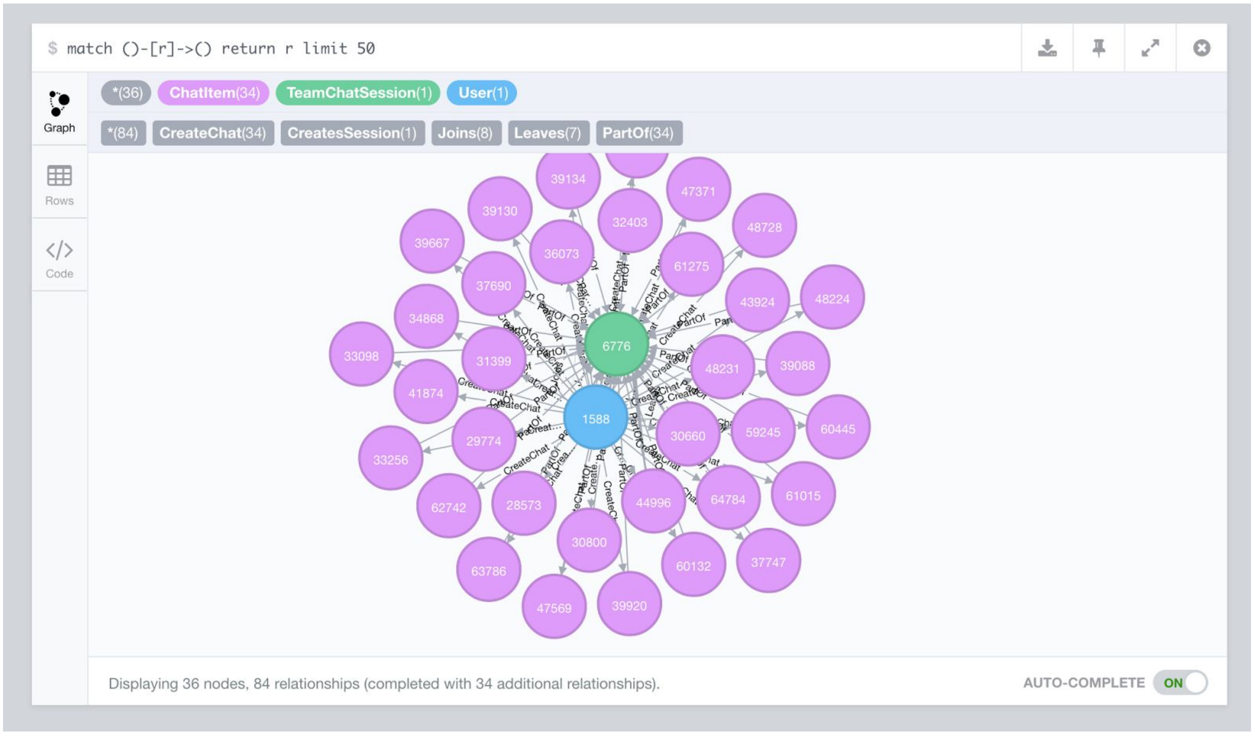
LOAD CSV FROM "file:///Users/EstherChang/Documents/R/Big_Data_Capstone/big_data_capstone_datasets_and_scripts/chat-data/chat_join_team_chat.csv" AS row

MERGE (u:User {id: toInt(row[0])})

MERGE (c:TeamChatSession {id: toInt(row[1])})

MERGE (u)-[:Joins{timeStamp: row[2]}]->(c)

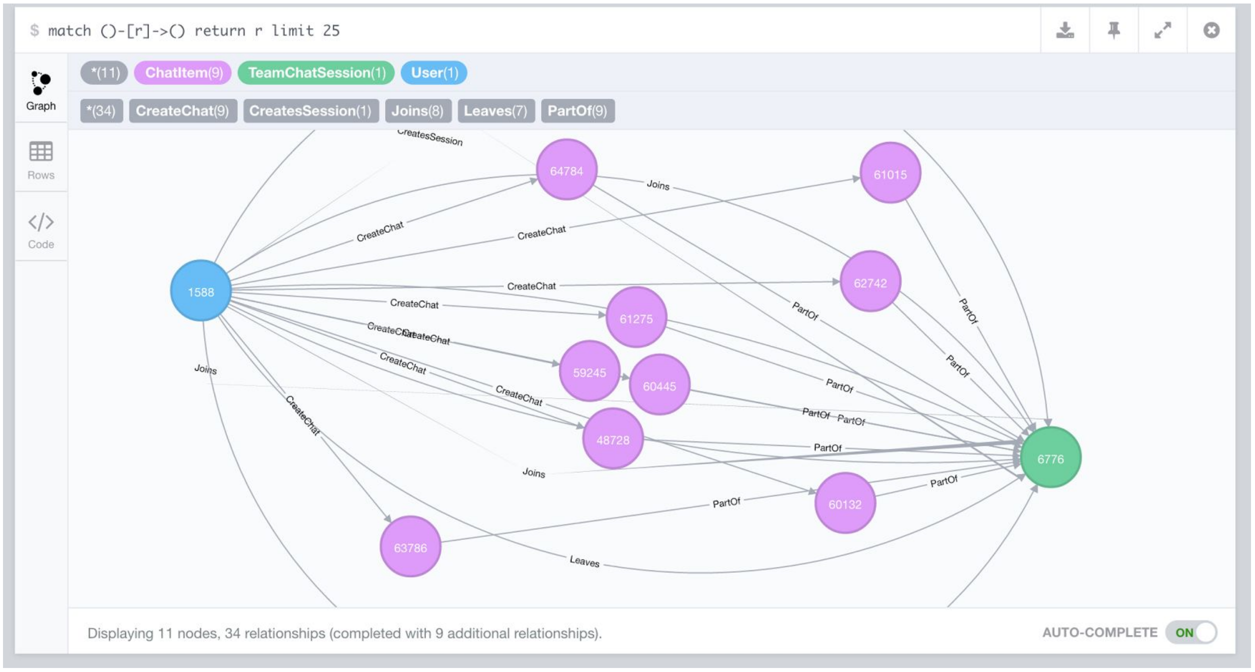
Screenshot:


```
$ match ()-[r]->() return r limit 50
```
*(36) ChatItem(34) TeamChatSession(1) User(1)

*(84) CreateChat(34) CreatesSession(1) Joins(8) Leaves(7) PartOf(34)

Displaying 36 nodes, 84 relationships (completed with 34 additional relationships).

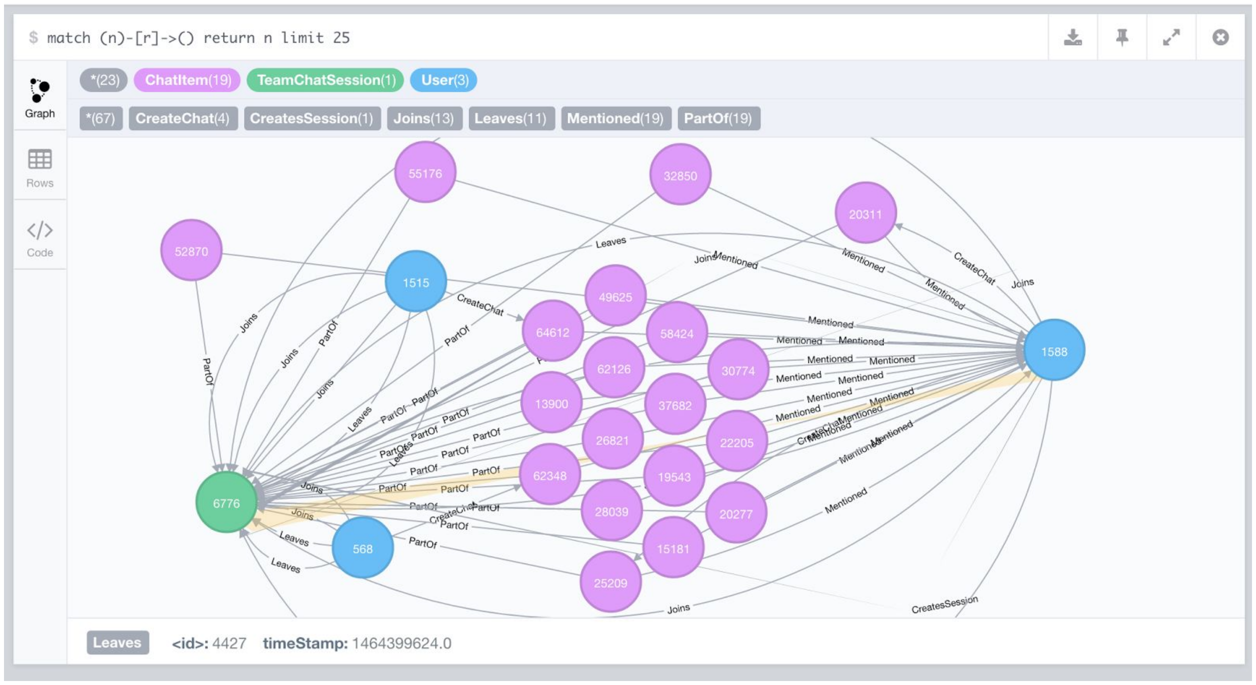
AUTO-COMPLETE ON


```
$ match ()-[r]->() return r limit 25
```
*(11) ChatItem(9) TeamChatSession(1) User(1)

*(34) CreateChat(9) CreatesSession(1) Joins(8) Leaves(7) PartOf(9)

Displaying 11 nodes, 34 relationships (completed with 9 additional relationships).

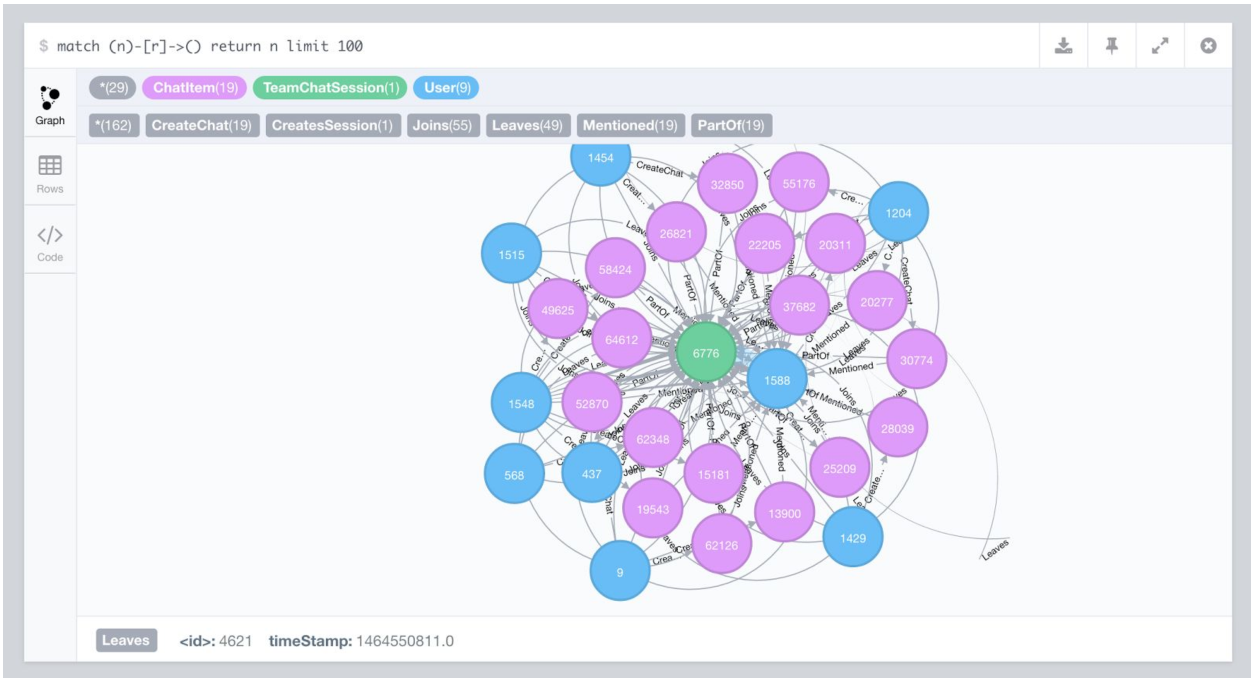
AUTO-COMPLETE ON



```
$ match (n)-[r]->() return n limit 25
```

*(23) ChatItem(19) TeamChatSession(1) User(3)
*(67) CreateChat(4) CreatesSession(1) Joins(13) Leaves(11) Mentioned(19) PartOf(19)
Graph
Rows
Code

```
Leaves   <id>: 4427   timeStamp: 1464399624.0
```



```
$ match (n)-[r]->() return n limit 100
```

*(29) ChatItem(19) TeamChatSession(1) User(9)
*(162) CreateChat(19) CreatesSession(1) Joins(55) Leaves(49) Mentioned(19) PartOf(19)
Graph
Rows
Code

```
Leaves   <id>: 4621   timeStamp: 1464550811.0
```

Finding the longest conversation chain and its participants

Report the results including the length of the conversation (path length) and how many unique users were part of the conversation chain. Describe your steps. Write the query that produces the correct answer.

Longest chain : 10 (directional)

Unique users : 5

STEPS:

```
match p=(i1:ChatItem)-[:ResponseTo*]->(i2:ChatItem)

return length(p)

ORDER BY length(p) DESC limit 1


match p=(a)-[:ResponseTo*]->(c) where length(p)=10

with p

match (i:ChatItem)<-[r:CreateChat]-(u:User)

where i in nodes(p)

return count(distinct u)
```

This kind of search may be relevant to Eglence Inc. business plan because Eglence Inc. can know what subject attracts more conversations and target it as a new business plan to increase revenue.

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.

STEPS:

```
match (u)-[:CreateChat*]->(i)

RETURN DISTINCT count(u.id), u.id

order by count(u.id) desc limit 10
```

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109
1087	109(there's a tie in the third place)

STEPS:

match (i:ChatItem)-[:PartOf]->(c:TeamChatSession)-[:OwnedBy]->(t)

return distinct count(t.id), t.id

order by count(t.id) desc limit 10

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.

Steps:

match (u:User)-[r:CreateChat]->(i:ChatItem)
with u.id as uid, order by count(i) desc limit 10

match (u:User)-[:Joins]->(c:TeamChatSession)-[:OwnedBy]->(t:Team)
where u.id=uid

return distinct u.id, t.id

Yes. There is one chattiest user (999) belongs to one chattiest team (52).

This kind of search may be relevant to Eglence Inc. business plan because there is a direct link between the chattiest user and his/her team being also the chattiest, Eglence could target the chattiest team with paid chatting stickers/emojis or something like that to gain revenue.

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.

Steps:

```
Match (u1:User)-[:CreateChat]->(i:ChatItem)-[:Mentioned]->(u2:User)

create (u1)-[:InteractsWith]->(u2)

Match (u1:User)-[:CreateChat]->(i:ChatItem)-[:ResponseTo]->(i2:ChatItem),
(u2:User)-[:CreateChat]->(i2:ChatItem)

create (u1)-[:InteractsWith]->(u2)

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

Finding coefficient Example:

```
MATCH (u1:User) WHERE u1.id = 394

MATCH (u1)-[:InteractsWith]-(n:User)

return count(distinct n)


MATCH (u1:User) WHERE u1.id = 394

MATCH (u1)-[:InteractsWith]-(n:User)

MATCH (n)-[:InteractsWith]->(o:User) where o <> u1

RETURN n.id,o.id, count(case when (n)-->(o) then 1

else 0

end) as value
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	0.75
2067	0.64
209	0.83

User 394 has k=4 neighbors, therefore k*(k-1) = 12. There are total of 9 edges in neighbors of neighbors.

User 2067 has k=8 neighbors, therefore k*(k-1) = 56. There are total of 36 edges in neighbors of neighbors.

User 209 has k=7 neighbors, therefore k*(k-1) = 42. There are total of 35 edges in neighbors of neighbors.