

Technical Appendix

Catch the Pink Flamingo Analysis

Produced by: Li Xiaowei

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 file about every advertisement click in the Flamingo app. One line is one advertisement click detail.	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 file about every in-app purchase in the Flamingo app. One line is one in-app purchase detail.	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

		<p>user who made the purchase</p> <p>userId: the user id of the user who made the purchase</p> <p>buyId: the id of the item purchased</p> <p>price: the price of the item purchased</p>
game-clicks.csv	A file about every click in the game from users. One line is one click detail.	<p>timestamp: when the click occurred.</p> <p>clickId: a unique id for the click.</p> <p>userId: the id of the user performing the click.</p> <p>userSessionId: the id of the session of the user when the click is performed.</p> <p>isHit: denotes if the click was on a flamingo (value is 1) or missed the flamingo (value is 0)</p> <p>teamId: the id of the team of the user</p> <p>teamLevel: the current level of the team of the user</p>
level-events.csv	A file about user starts and finishes a level. One line is either a team starts or finishes a level.	<p>timestamp: when the event occurred.</p> <p>eventId: a unique id for the event</p> <p>teamId: the id of the team</p> <p>teamLevel: the level started or completed</p> <p>eventType: the type of event, either start or end</p>
team.csv	A file about the team details when the team terminated the game. One line is one event when a team terminated in the game.	<p>teamId: the id of the team</p> <p>name: the name of the team</p> <p>teamCreationTime: the timestamp</p>

		<p>when the team was created</p> <p>teamEndTime: the timestamp when the last member left the team</p> <p>strength: a measure of team strength, roughly corresponding to the success of a team</p> <p>currentLevel: the current level of the team</p>
team-assignments.csv	A file about the events when a user joins a team. One line is one event.	<p>timestamp: when the user joined the team.</p> <p>team: the id of the team</p> <p>userId: the id of the user</p> <p>assignmentId: a unique id for this assignment</p>
users.csv	A file about the registered user details. One line is one user detail.	<p>timestamp: when user first played the game.</p> <p>userId: the user id assigned to the user.</p> <p>nick: the nickname chosen by the user.</p> <p>twitter: the twitter handle of the user.</p> <p>dob: the date of birth of the user.</p> <p>country: the two-letter country code where the user lives.</p>
user-session	A file about the session when a user starts or stops playing the game. One line is one session detail.	<p>timestamp: a timestamp denoting when the event occurred.</p> <p>userSessionId: a unique id for the session.</p> <p>userId: the current user's ID.</p> <p>teamId: the current user's team.</p>

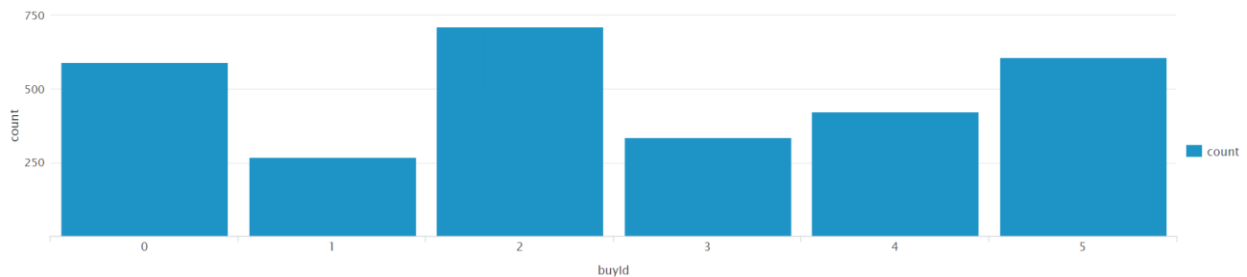
		<p>assignmentId: the team assignment id for the user to the team.</p> <p>sessionType: whether the event is the start or end of a session.</p> <p>teamLevel: the level of the team during this session.</p> <p>platformType: the type of platform of the user during this session.</p>

Aggregation

Amount spent buying items	21407
Number of unique items available to be purchased	6

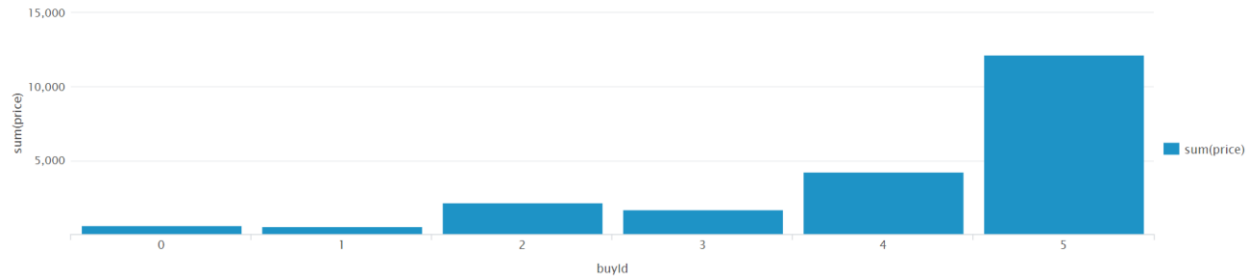
A histogram showing how many times each item is purchased:

source="buy-clicks.csv" | stats count by buyId



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

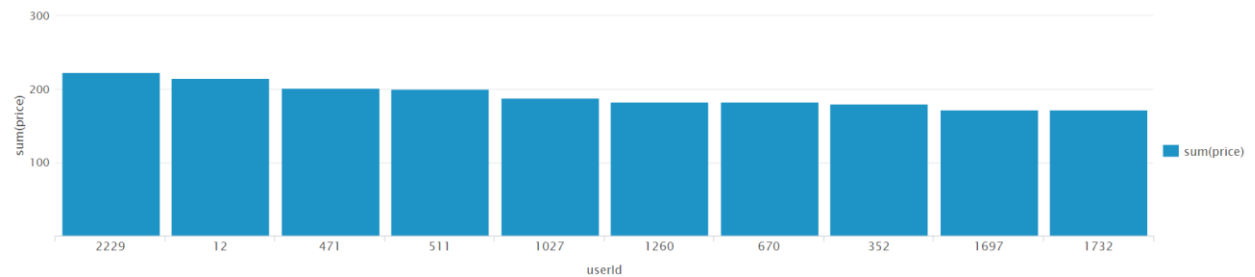
source="buy-clicks.csv" | stats sum(price) by buyId



Filtering

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

source="buy-clicks.csv" | stats sum(price) by userId | sort 10 -num(sum(price))



The following table shows the user id, platform, and hit-ratio percentage for the top three buying users:

source="game-clicks.csv" (userId=2229 OR userId=12 OR userId=471) | stats avg(isHit) by userId

Rank	User Id	Platform	Hit-Ratio (%)
1	2229	iPhone	11.6
2	12	iPhone	13.1
3	471	iPhone	14.5

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

The screenshot shows the KNIME 'Append Column' dialog. On the left, the 'Column List' contains various attributes, with 'avg_price' selected. The 'Function' dropdown is set to 'Numeric', and the 'Description' is 'Left <= right'. The 'Expression' field contains a list of rules, with the last two rules highlighted: '\$avg_price\$ > 5.0 => "HighRollers"' and '\$avg_price\$ <= 5.0 => "PennyPincher"'. The 'Append Column' field at the bottom is set to 'buyer'.

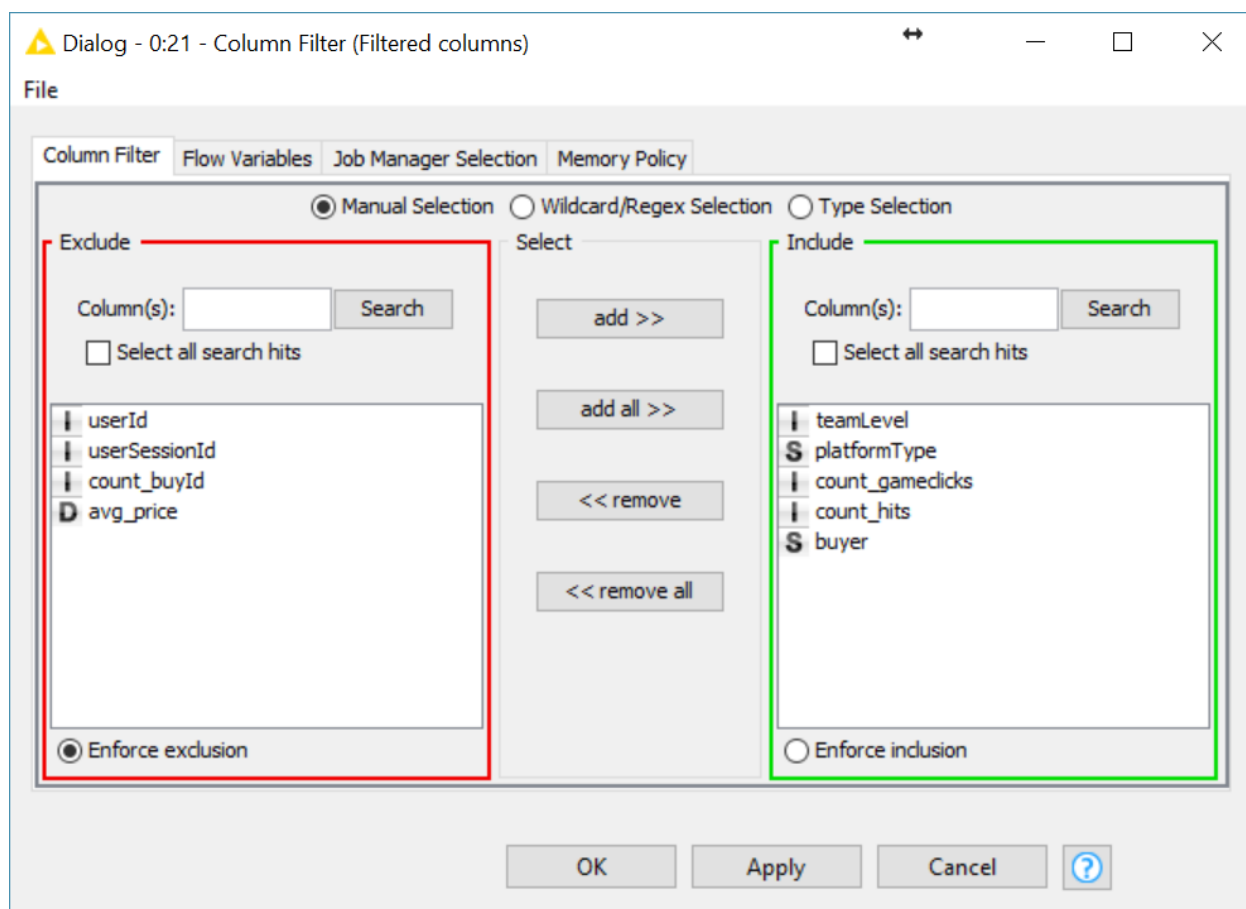
A new column/attribute was added based on the value of avg_price. The new attribute has value of either “HighRollers” or “PennyPincher”

The creation of this new categorical attribute was necessary because we will use this attribute to train and predict high-value players.

Attribute Selection

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

Attribute	Rationale for Filtering
userId	Random number of ID
userSessionId	Random number of user session
Count_buyId	The ID of the purchase item, note relevant to the classification.
Avg_price	We are interested in HighRollers or PennyPinchers.



Data Partitioning and Modeling

The data was partitioned into train and test datasets.

The 60% data set was used to create the decision tree model.

The trained model was then applied to the 40% dataset.

This is important because testing data can't be used for training.

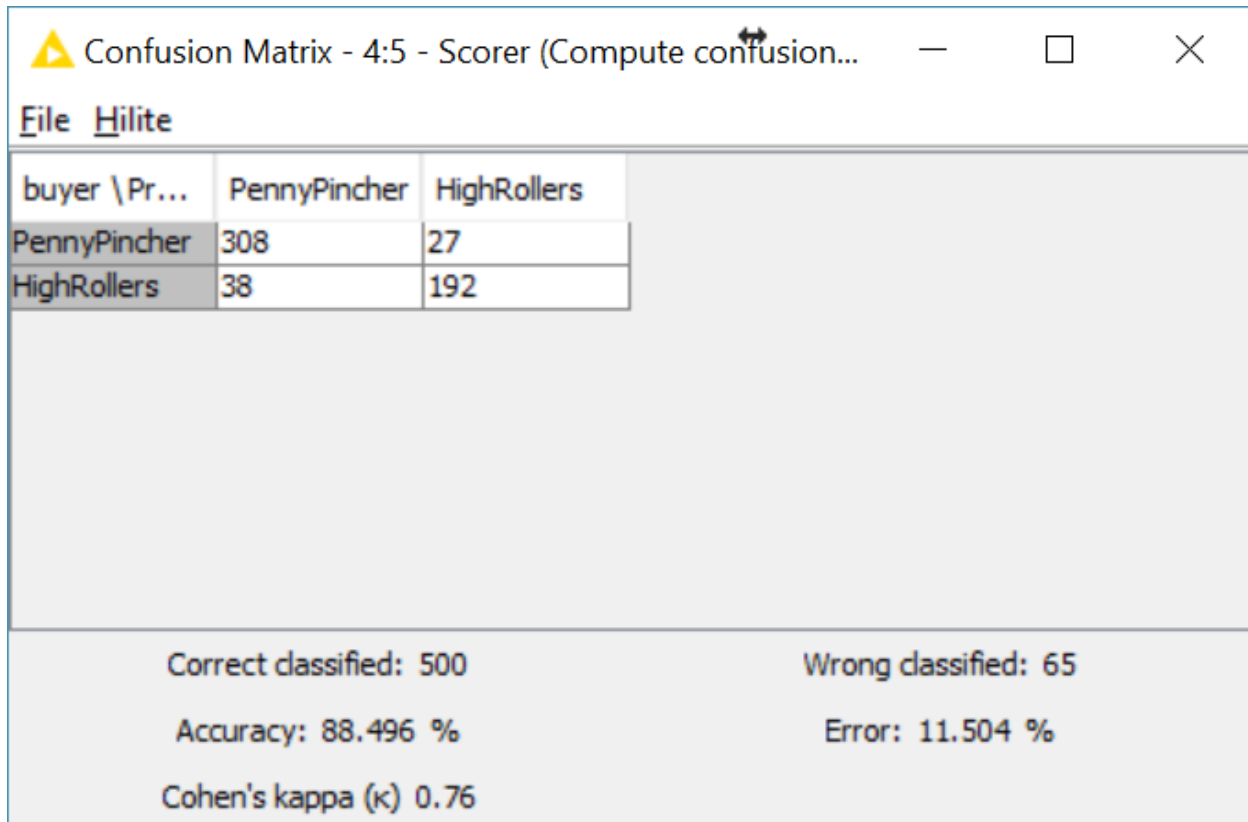
When partitioning the data using sampling, it is important to set the random seed because it'll create constant data partitioning result.

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



Evaluation

A screenshot of the confusion matrix can be seen below:



The screenshot shows a window titled "Confusion Matrix - 4:5 - Scorer (Compute confusion...)". It contains a table with the following data:

buyer \ Pr...	PennyPincher	HighRollers
PennyPincher	308	27
HighRollers	38	192

Below the table, the following statistics are displayed:

- Correct classified: 500
- Wrong classified: 65
- Accuracy: 88.496 %
- Error: 11.504 %
- Cohen's kappa (κ) 0.76

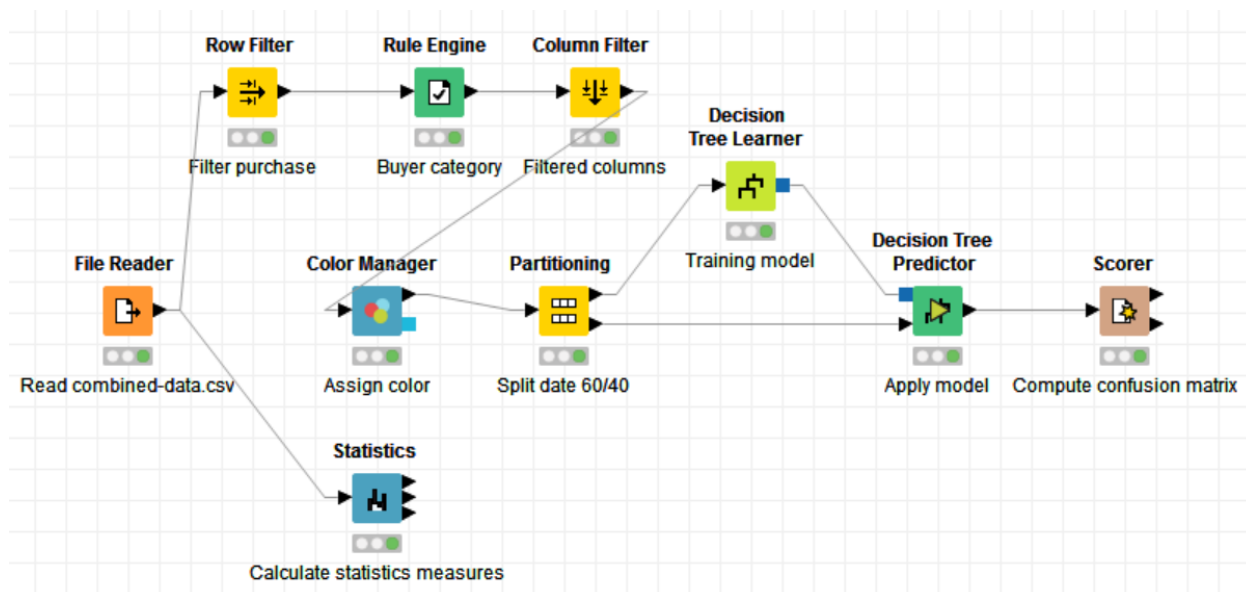
As seen in the screenshot above, the overall accuracy of the model is 88.5%.

308 PennyPinchers are correctly classified, while 27 PennyPinchers are wrongly classified as HighRollers.

192 HighRollers are correctly classified, while 38 HighRollers are wrongly classified as PennyPinchers.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

Based on the decision tree model, the platform type is used for the prediction. In another word, money talks. Without doubt, the average price of iphone is greater than android. And most of iphone players are HighRollers while most of Android players are PennPinchers.

Specific Recommendations to Increase Revenue	
1.	Produce iPhone only addon package as in-app purchases.
2.	Make some promotions for platforms other than iPhone to stipulate the purchase.

Clustering Analysis

Attribute Selection

Attribute	Rationale for Selection
Game-clicks.csv, isHit	I'd like to explore the relationship between the ratio of hitting a flamingo and money spent by the user. So this feature is a must.
Buy-clicks.csv, price	In order to analyse the mentioned relationship, price of the purchase is a must.
Game-clicks.csv, userId	Need this attribute to perform aggregation.
Buy-clicks.csv, userId	Need this attribute to perform aggregation.

Training Data Set Creation

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

```

ratio revenue
0 0.134078    21.0
1 0.100000    53.0
2 0.122047    80.0
3 0.109430    11.0
4 0.130682   215.0

```

Dimensions of the training data set (rows x columns) : 546 * 2

```

In [53]: trainingDF.shape
Out[53]: (546, 2)

```

of clusters created: 2. Here I set the number to 2.

Cluster Centers

Cluster #	Cluster Center (hitratio, revenue)
1	Array([0.12591902, 117.86206897])
2	Array([0.11197414, 24.29847495])

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that the flamingo hit ratio is higher and revenue generated is much more than cluster 2.

Cluster 2 is different from the others in that the flamingo hit ratio is slight lower but revenue generated is much less than cluster 1.

Despite the significant difference on revenue for the 2 clusters, the hit ratio is considered quite close to each other. Hence, any predication on this result may not lead to valuable decisions in real world.

Recommended Actions

Action Recommended	Rationale for the action
--------------------	--------------------------

Increase the in-app purchase price for users better at hitting flamingos	At least from the cluster center, we can tell higher hit ratio leads to significant revenue increase.
Increase the game difficulty for users better at hitting flamingos	Since they already have good records on spending money for in-app purchase. The difficulty increment should stipulate them to purchase more in-app items.

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

(Describe the graph model for chats in a few sentences. Try to be clear and complete.)

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

File	Attributes
chat_create_team_chat.csv	User, Team, TeamChTeamatSession, timeStamp(CreateSession), timeStamp(OwnedBy)
chat_join_team_chat.csv	User, TeamChatSession, timeStamp(Join)
chat_leave_team_chat.csv	User, TeamChatSession, timeStamp(Leaves)
chat_item_team_chat.csv	User, TeamChatSession, ChatItem, timestamp(CreateChat), timeStamp(PartOf)
chat_mention_team_chat.csv	ChatItem, User, timeStamp (Mentioned)
chat_respond_team_chat.csv	ChatItem, ChatItem, timestamp(ResponseTo)

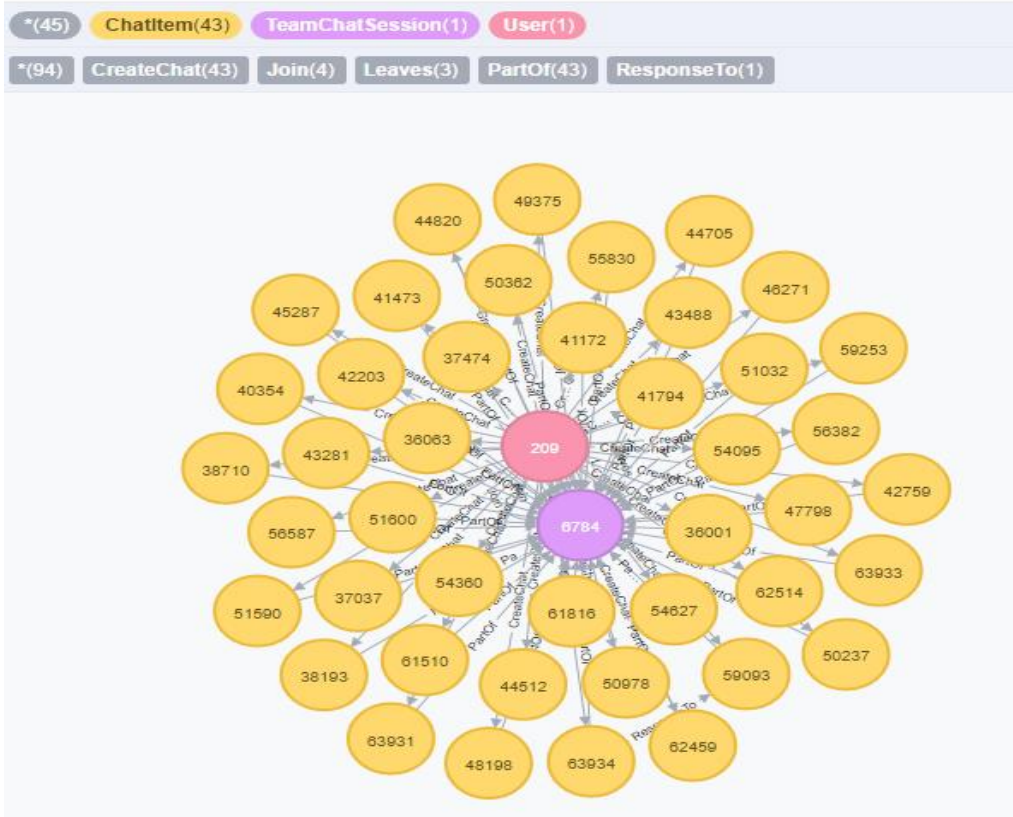
- ii) Explain the loading process and include a sample LOAD command
Below is the command of loading “chat_item_team_chat.csv” file. The first 4 lines read the file and create the 3 attributes User, TeamChatSession and ChatItem. The latter 2 lines create the relationship of “User CreatChat ChatItem” and “ChatItem PartOf TeamChatSession”.

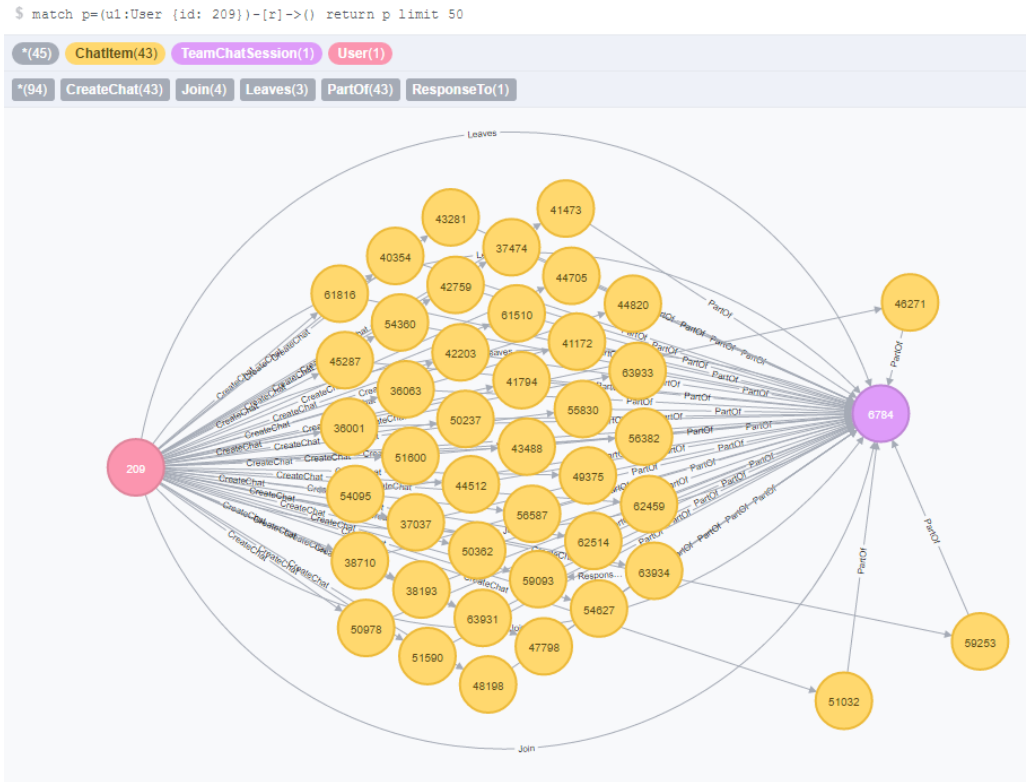
```
LOAD CSV FROM "file:/chat_item_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])})
MERGE (c:TeamChatSession {id: toInteger(row[1])})
MERGE (i:ChatItem {id: toInteger(row[2])})
MERGE (u)-[:CreateChat{timestamp: row[3]}]->(i)
MERGE (i)-[:PartOf{timestamp: row[3]}]->(c)
```

- 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.

```
$ match p=(u1:User {id: 209})-[r]->() return p limit 50
```





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.

The relationship between ChatItems is "ResponseTo", in order to find the longest conversation, need to use "ResponseTo*" to perform the query. Below is the code and returned result is 9, so 10 chats.

```
match p=(m:ChatItem)-[:ResponseTo*]->(n:ChatItem)
```

```
return length(p), p
```

```
order by length(p) desc limit 1
```

The second part is to find the participants. The returned result is 5.

```
match p=(m:ChatItem)-[:ResponseTo*]->(n:ChatItem)
```

```
where length(p)=9
```

```
with p as onepath
```

```
match q=(u:User)-[:CreateChat]-(e:ChatItem)
```

```
where (e IN NODES(onepath))
```

```
return count(distinct u)
```

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 number of chats created by a user can be determined by the “CreateChat” relationship. So below query does the job.

```
match p=(u:User)-[r>CreateChat]->()
```

```
return count(u), u.id
```

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

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

It’s a little troublesome to find the answer of chattiest team, but by combining the relationships of “PartOf” and “OwnedBy”, below query will work.

```
match q=(i:ChatItem)-[p:PartOf]->(c:TeamChatSession)-[o:OwnedBy]->(t:Team)
```

```
return count(q), t.id
```

```
order by count(q) 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.

By using the information of the top 10 chattiest users and teams. Below query returns 1 result, user id 999 is the only one from both top 10 chattiest users and teams, and team id is 52.

```
match q=(u:User)-[r>CreateChat]->(i:ChatItem)-[p:PartOf]->(c:TeamChatSession)-[o:OwnedBy]->(t:Team)
```

```
where t.id IN [82, 185, 112, 18, 194, 129, 52, 136, 146, 81] and u.id IN [394, 2067, 209, 1087, 554, 516, 1627, 999, 668, 461]
```


return distinct u.id, t.id

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.

Firstly, based on the guide to create the “InteractsWith” relationship and remove loops.

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

```
Create (u1)-[:InteractsWith]->(u2)
```

```
MATCH (u1:User)-[:CreateChat]->(i:ChatItem)-[:ResponseTo]->(o:ChatItem)-[:CreateChat]->(u2:User)
```

```
Create (u1)-[:InteractsWith]->(u2)
```

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

Then, figuring out the users who have the “InteractsWith” relationship with the chattiest user. Now we know the user IDs in the neighbourhood. The next task will be finding how much distinct edges in between them. Below is the example of user id 394.

```
match (u1:User {id: 394})-[:InteractsWith]->(u2:User) with collect(distinct u2.id) as neighbors
```

```
match p=(u3:User)-[:InteractsWith]->(u4:User) where (u3.id IN (neighbors + 394)) and (u4.id IN (neighbors + 394))
return count(distinct nodes(p))
```

And the returned result is 17. So coefficient is $17 / (5 * 4) = 0.85$. It works the same for the other two users.

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	0.85
2067	0.95
209	1.0

Recommended Actions

Finally, make recommendations to Eglence, Inc. and include examples of how your findings support them. Include this information in Slide 6 of your final presentation.

It can be easily seen that these neighbourhoods are quite dense. It can be a good idea to push them some advertisements about the in-app purchases that can help them bond their relationships.