Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: <Arsanuos Essa Attia>

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	**timestamp**: when user first played

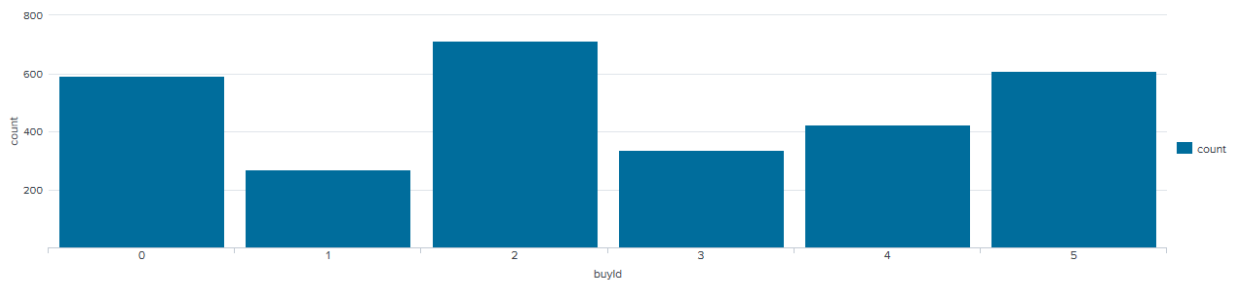
	user playing the game.	the game. **userId**: the user id assigned to the user. **nick**: the nickname chosen by the user. **twitter**: the twitter handle of the user. **dob**: the date of birth of the user. **country**: the two-letter country code where the user lives.
team.csv	This file contains a line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

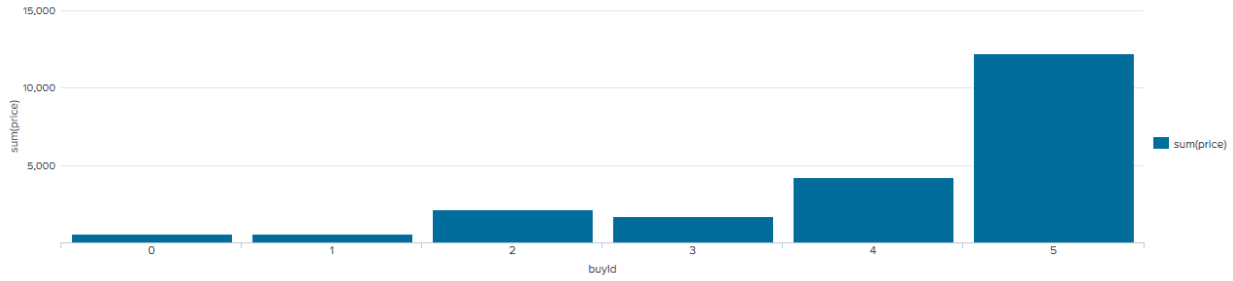
Amount spent buying items	21407.0
Number of unique items available to be purchased	6

A histogram showing how many times each item is purchased:



buyId	count
0	592
1	269
2	714
3	337
4	425
5	610

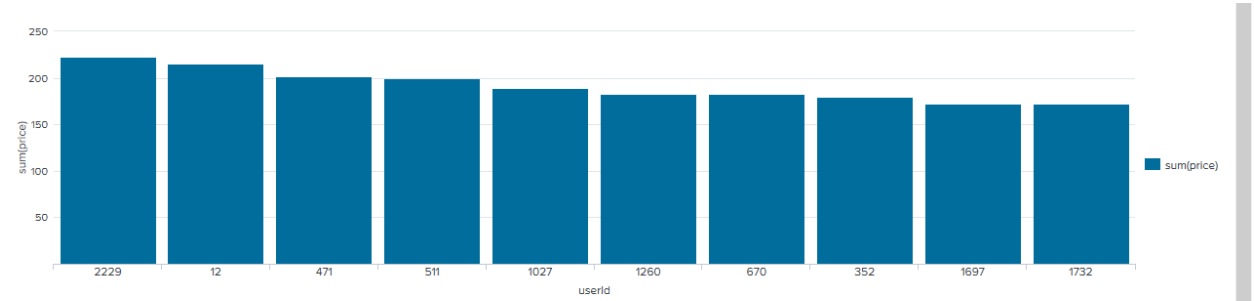
A histogram showing how much money was made from each item:



buyId	sum(price)
0	592.0
1	538.0
2	2142.0
3	1685.0
4	4250.0
5	12200.0

Filtering

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



userId	sum(price)
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

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	34500
2 - uneven	0.55	0.60	0.55	36289

Data Classification Analysis

Data Preparation

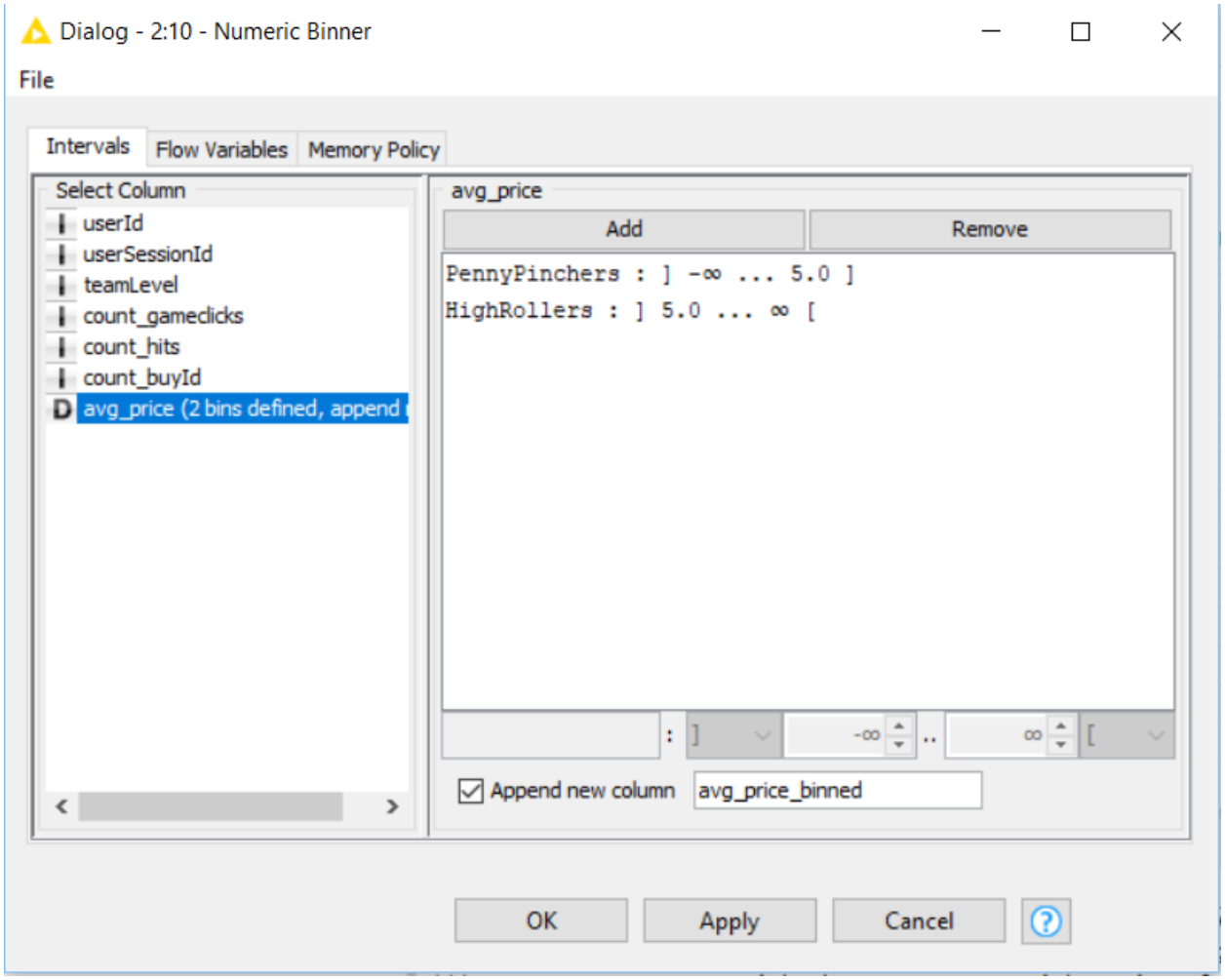
Analysis of combined_data.csv

Sample Selection

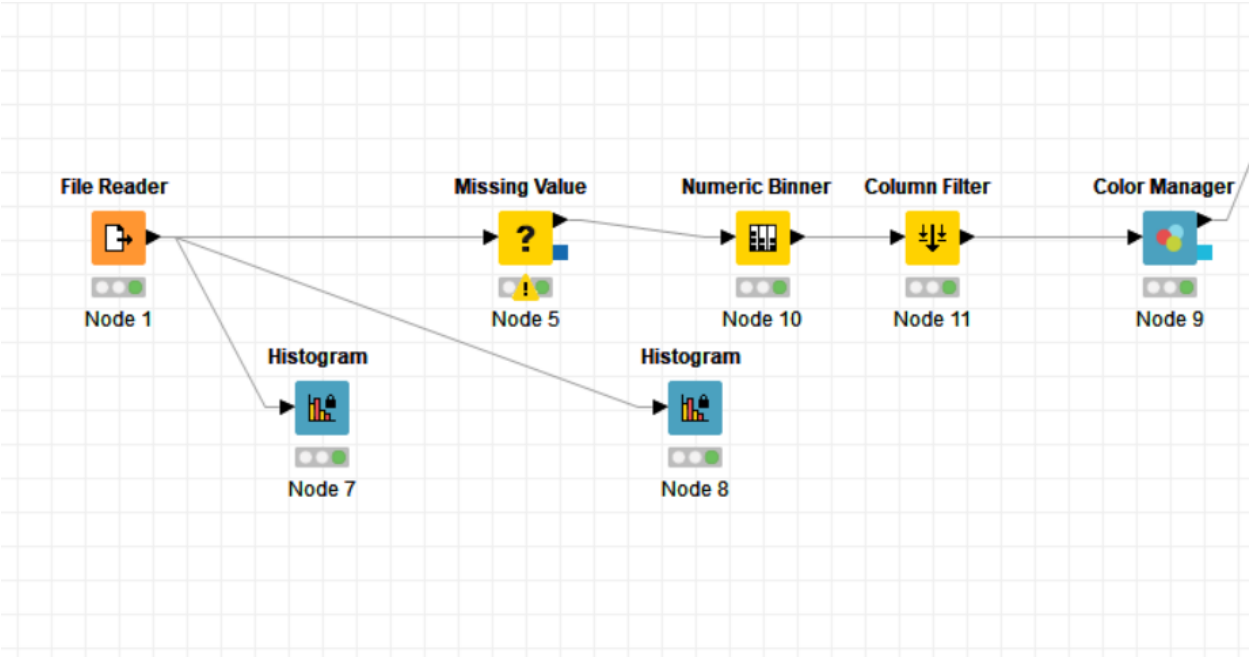
Item	Amount
# of Samples	4619
# of Samples with Purchases	1411

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:



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



Filter missing values of buyId by removing them. Binning every values in avg price less than or equal 5 in pennyPinchers bin and greater than 5 as highRollers.

Creating this new categorical variable to classify the use this variable to train and visualize selection of the decision tree.

Attribute Selection

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

Attribute	**Rationale for Filtering**
Avg price	As we don't need this attribute so far. We replaced it with the categorical variable.

Data Partitioning and Modeling

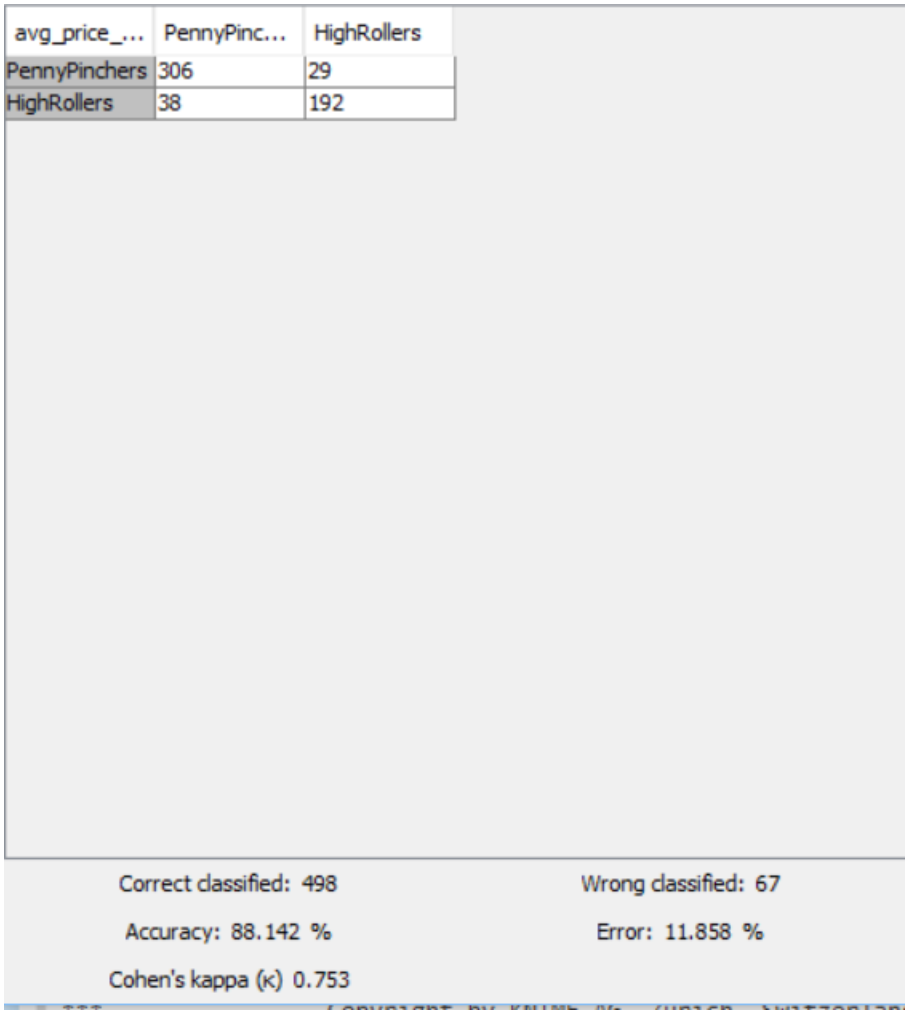
The data was partitioned into train and test datasets.
The train data set was used to create the decision tree model.
The trained model was then applied to the test dataset.
This is important because we want to know how our model works for example accuracy.

When partitioning the data using sampling, it is important to set the random seed because… get the same values in train set and test set as any one use this data set.

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

Evaluation

A screenshot of the confusion matrix can be seen below:

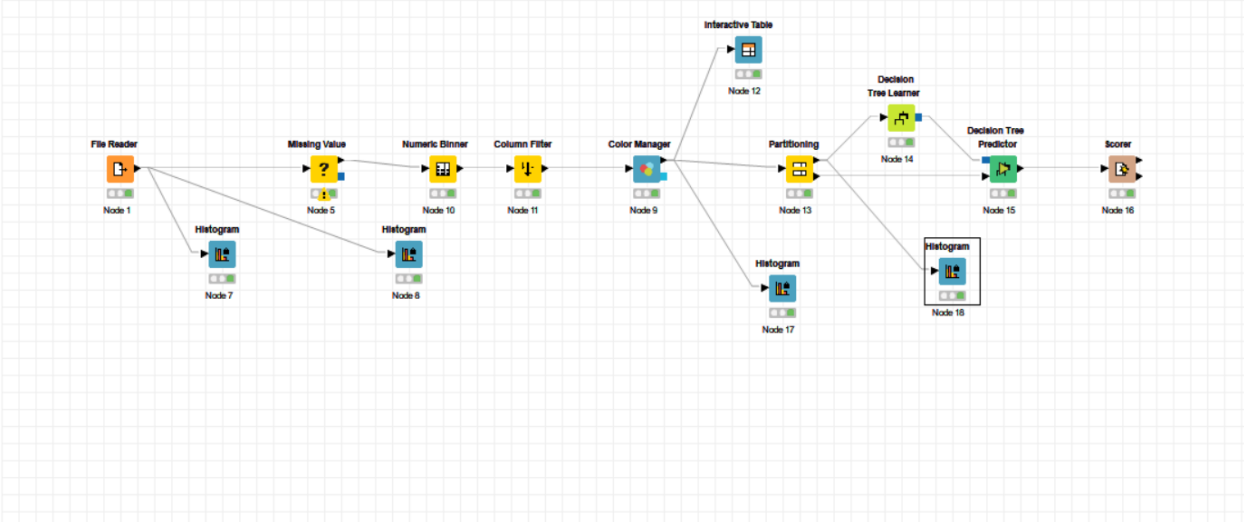


As seen in the screenshot above, the overall accuracy of the model is 88.142%

306 PenneyPincher were classified true.
192 HighRollers were classified true.
29 PenneyPincher were incorrectly classified as HighRollers.
38 HighRollers were incorrectly classified as penneyPincher.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?
According to the classification users that use Iphone tend to buy more than any kind of users (platform type).

Specific Recommendations to Increase Revenue
1. try to increase our users that uses Iphone buy updating the app in app store.
2. I think working on mac can have a great effect as they are the second source of income. Linux also have a very bad income so trying to find more solutions for these users.

Clustering Analysis

Attribute Selection

Attribute	Rationale for Selection
Ad-clicks per user	To distinguish the cluster with the number of clicks they made
Hit-clicks per user	To distinguish which cluster has more proficient users
Income per user	To distinguish which cluster has more income

Training Data Set Creation

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

userId	average	isHit	count
1	2.333333	96	396.0
8	10.600000	38	50.0
9	13.333333	62	222.0
10	1.100000	340	190.0
12	16.538462	92	598.0

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

# of clusters created: 4

Cluster Centers

Cluster #	Cluster Center
1	[-0.69125992, -0.37929573, -0.60651861]
2	[-0.28198212, -0.00380089, 1.34496615]
3	[-2.12494965e-01, 2.47162631e+00, 5.26368778e-04]
4	[1.21598491, -0.25880204, -0.35842573]

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that it containing users with min in all attributes (average income, number of hits, number of buy clicks .

Cluster 2 is different from the others in that it containing users with a better number of buy clicks and more income.

Cluster 3 is different from the others in that it containing users with the highest hit in games but on the other hand with the min in buyclicks and income which means that the more proficient were the users the less income.

Cluster 4 is different from the others in that it containing with the highest income but with a small buyclicks which means that the price of each item is critical in income and the type of the ad and its price is a great factor in income.

Recommended Actions

Action Recommended	Rationale for the action
higher prices can fit for cluster 4	They clicks less but buys more.
Ads can be appeared more for cluster 2	They clicks more on ads but buy less the thing that should be exploited in the way of increasing ads for this cluster.

Graph Analytics

Modeling Chat Data using a Graph Data Model

Representing chat data as database where user, chatSession, chatItems and teams is nodes while edges is the interaction based on each case which at general contains the timestamp of interaction.
More information described below.

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 Name	Description	Fields
chat_create_team_chat.csv	userid	the user id assigned to the user
	teamid	the id of the team
	TeamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the chat session created
chat_item_team_chat.csv	userid	the user id assigned to the user
	teamchatsessionid	a unique id for the chat session
	chatitemid	a unique id for the chat item
	timestamp	a timestamp denoting when the chat item created
chat_join_team_chat.csv	userid	the user id assigned to the user
	TeamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the user join in a chat session
chat_leave_team_chat.csv	userid	the user id assigned to the user
	teamchatsessionid	a unique id for the chat session
	timestamp	a timestamp denoting when the user leave a chat session
chat_mention_team_chat.csv	ChatItemId	the id of the ChatItem
	userid	the user id assigned to the user
	timeStamp	a timestamp denoting when the user mentioned by a chat item
chat_respond_team_chat.csv	chatid1	the id of the chat post 1
	chatid2	the id of the chat post 2
	timestamp	a timestamp denoting when the chat post 1 responds to the chat post 2

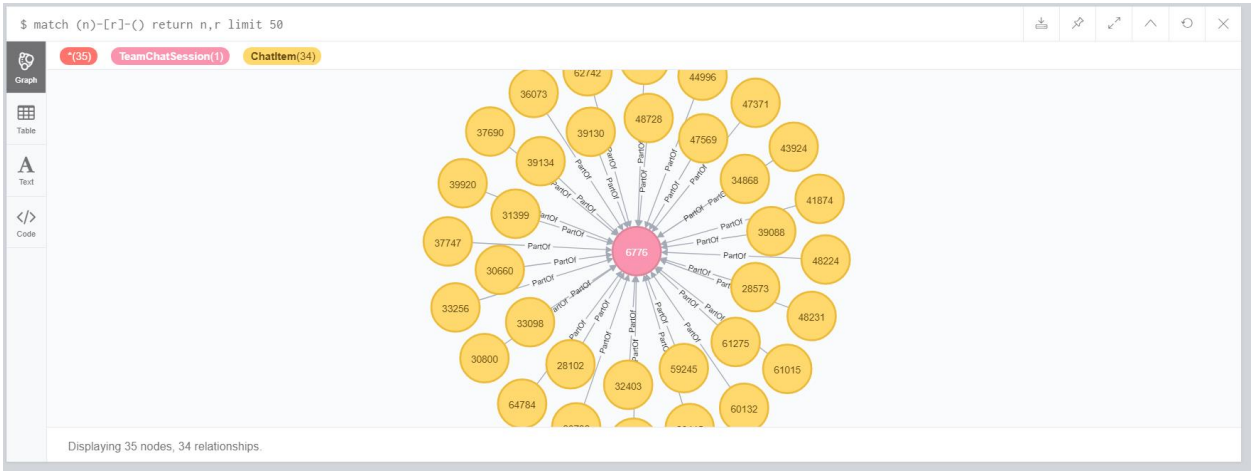
ii) Explain the loading process and include a sample LOAD command

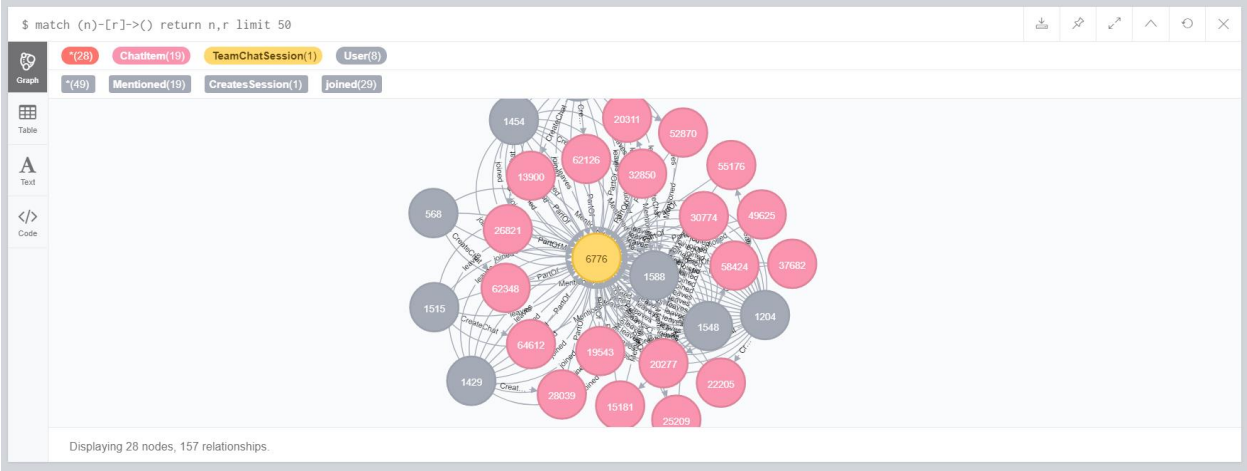
Copy data into the import folder inside the database folder the run queries like the following for each file

```
LOAD CSV FROM "file:/chat_join_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])})
MERGE (c:TeamChatSession {id: toInteger(row[1])})
MERGE (u)-[:joined{timeStamp: row[2]}]->(c)
```

Line1: loads data from chat_join_team_chat .csv file
Line2: creates node tagged as User and has attribute id from csv file data column 0
Line3: creates nodes tagged as TeamChatSession and has attribute id from csv file data column 1
Line4: links User node to TeamChatSession with relation called joined with timestamp attribute loaded from csv column 2

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

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.

- Finding the longest conversation chain

  match p = (i1)-[:ResponseTo*]->(i2)
  return length(p)
  order by length(p) desc limit 1
  Explanation:
  Get the longest path connect chatItem by getting all paths sort the in descending order the return the first (longest one).

- Unique users were part of this chain

  match p = (i1)-[:ResponseTo*]->(i2)
  where length(p) = 9
  with p
  match (u)-[:CreateChat]->(i)
  where i in nodes(p)
  return count(distinct u)

Explanation:
We know the longest conversation chain length the get the count of unique users.

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

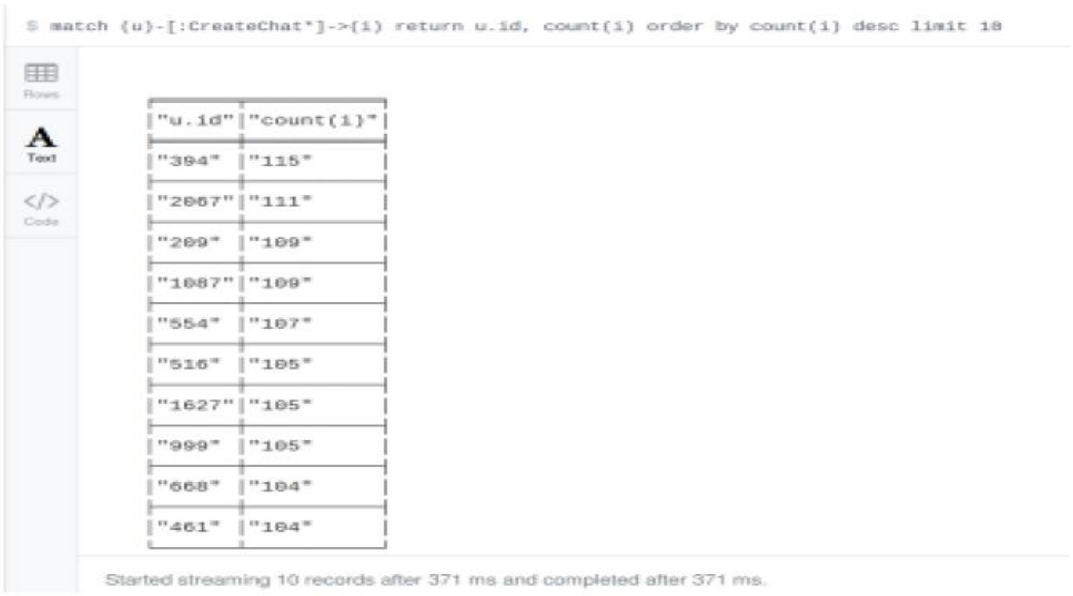
Describe your steps from Question 2. In the process, create the following two tables. You only need to include the top 3 for each table. Identify and report whether any of the chattiest users were part of any of the chattiest teams.

- **Query:**

  match (u)-[:CreateChat*]->(i)
  return u.id, count(i)
  order by count(i) desc limit 10

- **Explanation:**

  Match users that connected to chats using CreateChat relation then return user id and count of chats created by each user id in descending order limited by 10.



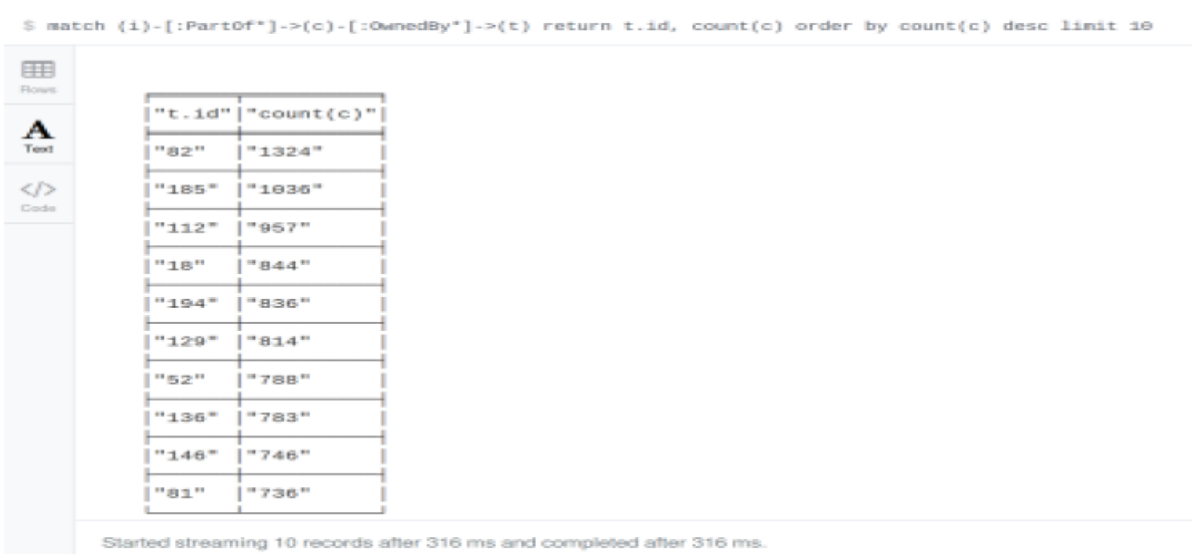
Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

- **Query:**

  match (i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)
  return t.id, count(c)
  order by count(c) desc limit 10
- **Explanation:**

  Match chatItem that is part of TeamChatSession then match TeamChatSession that is ownedBy t then return team id and count of TeamChatSession in descending order limited to 10.

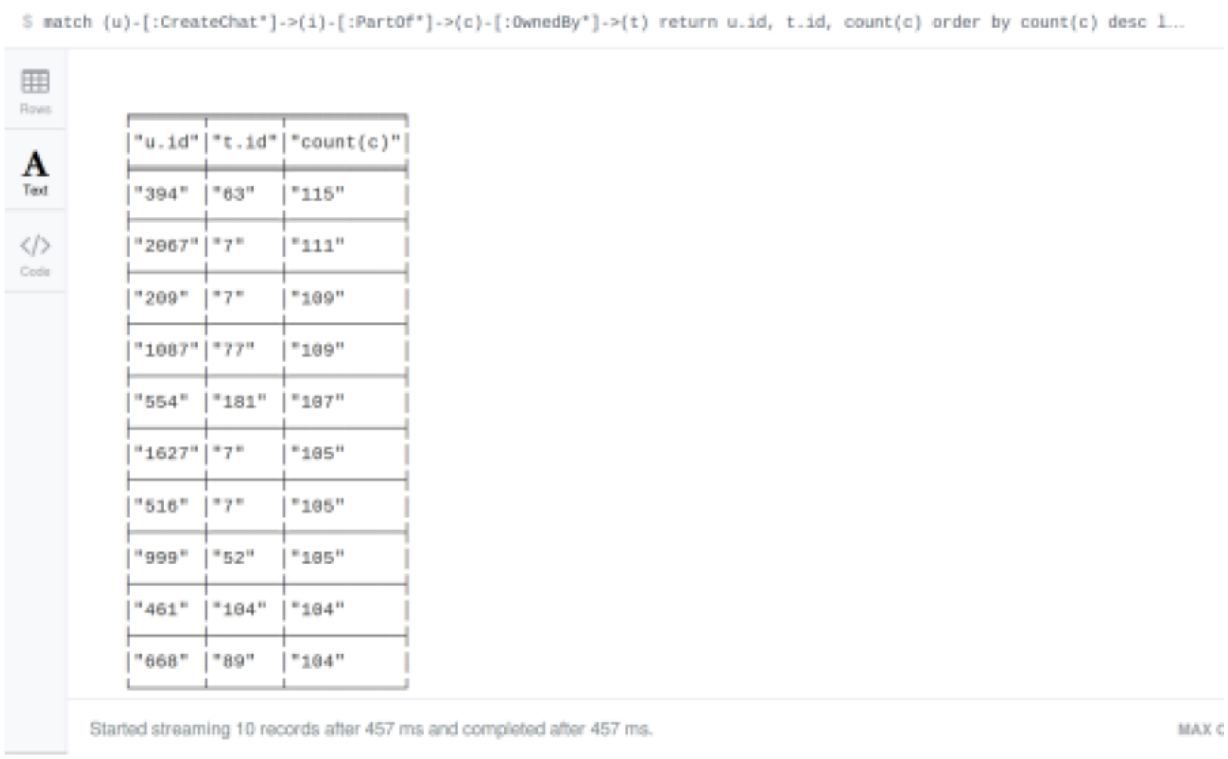


Chattiest Teams

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 following query
match (u)-[:CreateChat*]->(i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)
return u.id, t.id, count(c)
order by count(c) desc limit 10



Users	Team	Count
394	63	115
2067	7	111
209	7	109

User 999 in team 52 is part of top 10 but the rest aren't .

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.

a) We will construct the neighborhood of users. In this neighborhood, we will connect two users if

- One mentioned another user in a chat

- One created a chatItem in response to another user's chatItem
  - Queries:

```
match (u1:User)-[:CreateChat]->(i)-[:Mentioned]->(u2:User)
create (u1)-[:Deal]->(u2)

match (u1:User)-[:CreateChat]->(i1:ChatItem)-[:ResponseTo]-
(i2:ChatItem)
with u1, i1, i2
match (u2)-[:CreateChat]-(i2)
create (u1)-[:Deal]->(u2)
```

b) The above scheme will create an undesirable side effect if a user has responded to her own chatItem, because it will create a self-loop between two users. So, after the first two steps we need to eliminate all self-loops involving the edge "Deal".

```
match (u1)-[r:Deal]->(u1) delete r
remove them by the previous query
```

- The following query will return the number of edges

```
match (u1:User)-[r1:Deal]->(u2:User)
where u1.id <> u2.id with u1, collect(u2.id) as neighbors, count(distinct(u2)) as
neighborAmount
match (u3:User)-[r2:Deal]->(u4:User)
where (u3.id in neighbors) AND (u4.id in neighbors) AND (u3.id <> u4.id)
return u3.id, u4.id, count(r2)
```

- If one member has multiple edges with another member we need to count it as 1 because we care only if the edge exists or not.

```
match (u1:User)-[r1:Deal]->(u2:User)
where u1.id <> u2.id with u1, collect(u2.id) as neighbors, count(distinct(u2)) as
neighborAmount
match (u3:User)-[r2:Deal]->(u4:User)
where (u3.id in neighbors) AND (u4.id in neighbors) AND (u3.id <> u4.id)
return u3.id, u4.id, count(r2), case
when count(r2) > 0
then 1
else 0
end as value
```

c) The rest is to get the coefficient:

```
match (u1:User)-[r1:Deal]->(u2:User)
where u1.id <> u2.id with u1, collect(u2.id) as neighbors, count(distinct(u2)) as
neighborAmount
match (u3:User)-[r2:Deal]->(u4:User)
```

```
where (u3.id in neighbors) AND (u4.id in neighbors) AND (u3.id <> u4.id) with u1, u3, u4,
neighborAmount,
case when (u3)-->(u4)
then 1
else 0
end as value
return u1, (sum(value)/(neighborAmount*(neighborAmount-1))) as coeff
order by coeff desc limit 3
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
209	.9524
554	.9048
1087	.8