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 purchase was made tdId: 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
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

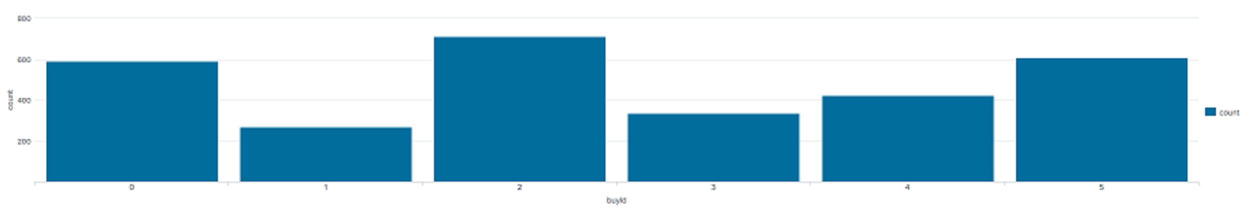
		when the team was created teamEndTime: the timestamp when the last member left the team strength: a measure of team strength, roughly corresponding to the success of a team currentLevel: the current level of the team
team-assignments.csv	A line is added to this file each time a user joins a team. A user can be in at most a single team at a time.	timestamp: when the user joined the team team: the id ot 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
<Fill In>	<Fill in short phrase>	<Fill In: Name and describe all fields>
<Fill In>	<Fill in short phrase>	<Fill In: Name and describe all fields>

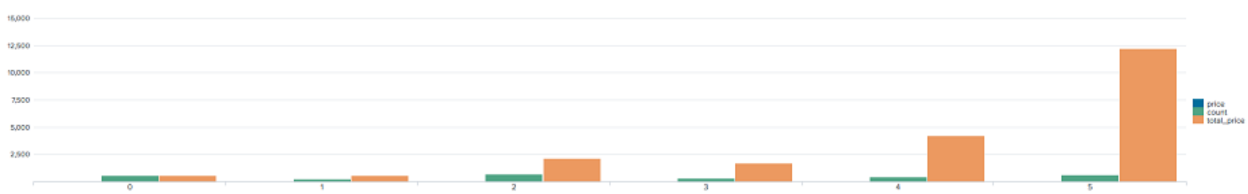
Aggregation

Amount spent buying items	592,538,2142,1685,4250,12200
Number of unique items available to be purchased	6

A histogram showing how many times each item is purchased:

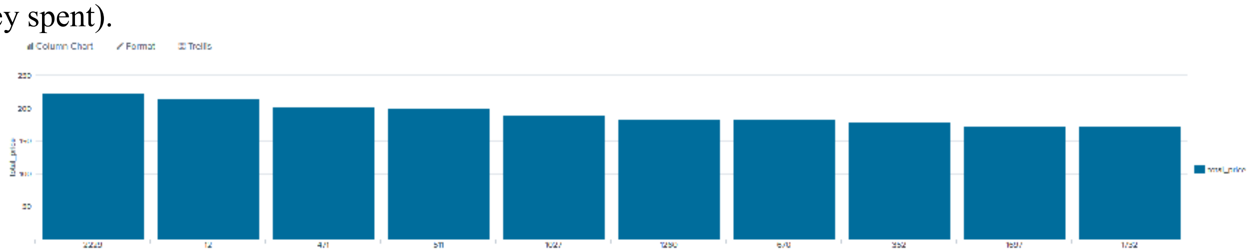


A histogram showing how much money was made from each item:



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	11.6
2	12	iphone	13.07
3	471	iphone	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:

Binned Data - 0:13 - Numeric Binner

File Hilite Navigation View

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

Row ID	I userId	I userSe...	I teamLevel	S platfor...	I count_...	I count_...	I count_...	D avg_price	S 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	HighRollers
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	HighRollers
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	HighRollers

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 avgprice.

Attribute Selection

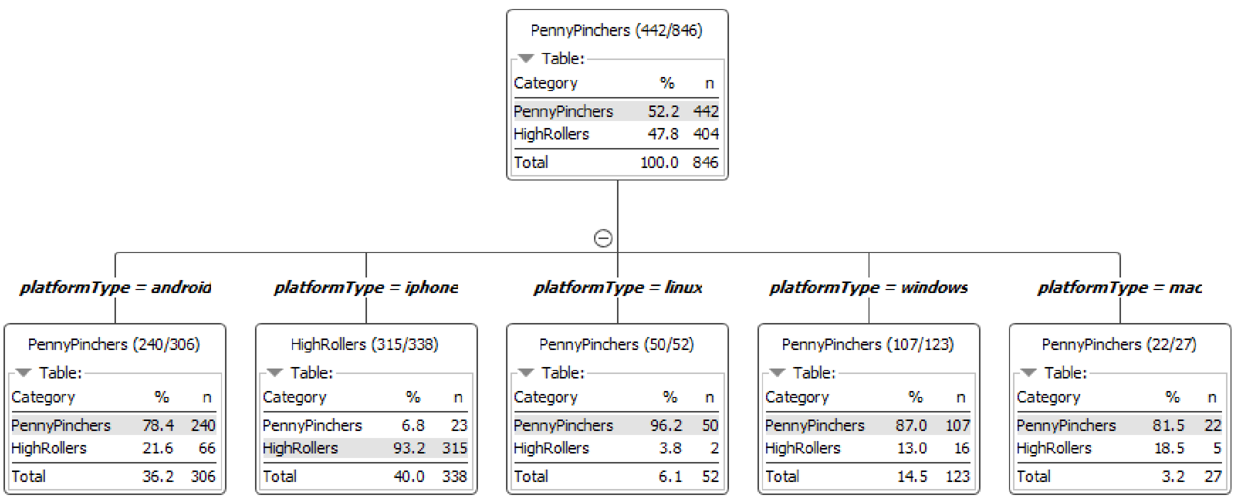
The following attributes were filtered from the dataset for the following reasons:

Attribute	**Rationale for Filtering**
avg_price	We don’t need the average price anymore since we have a new
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

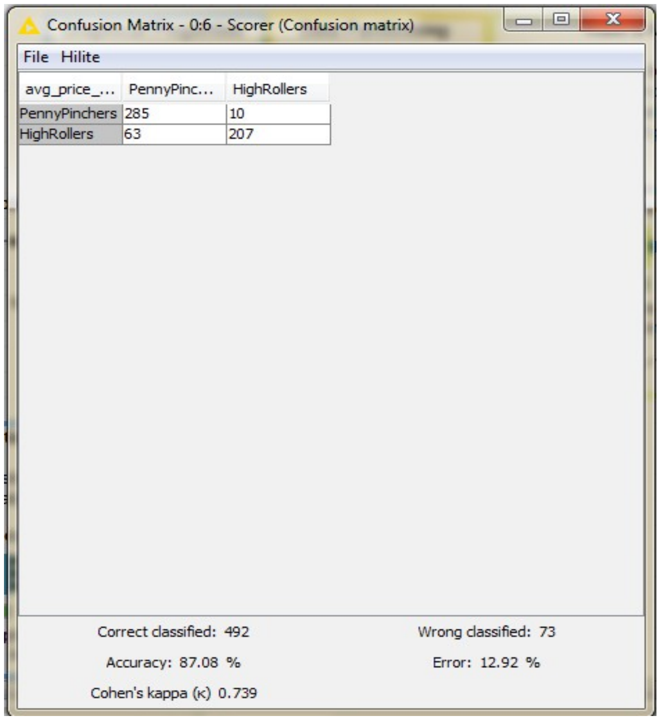
The data was partitioned into train and test datasets.
The <Fill In> data set was used to create the decision tree model.
The trained model was then applied to the test dataset.
This is important because when we do data analysis, we should test our model on a data set that was not used to train the model . 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:



Evaluation

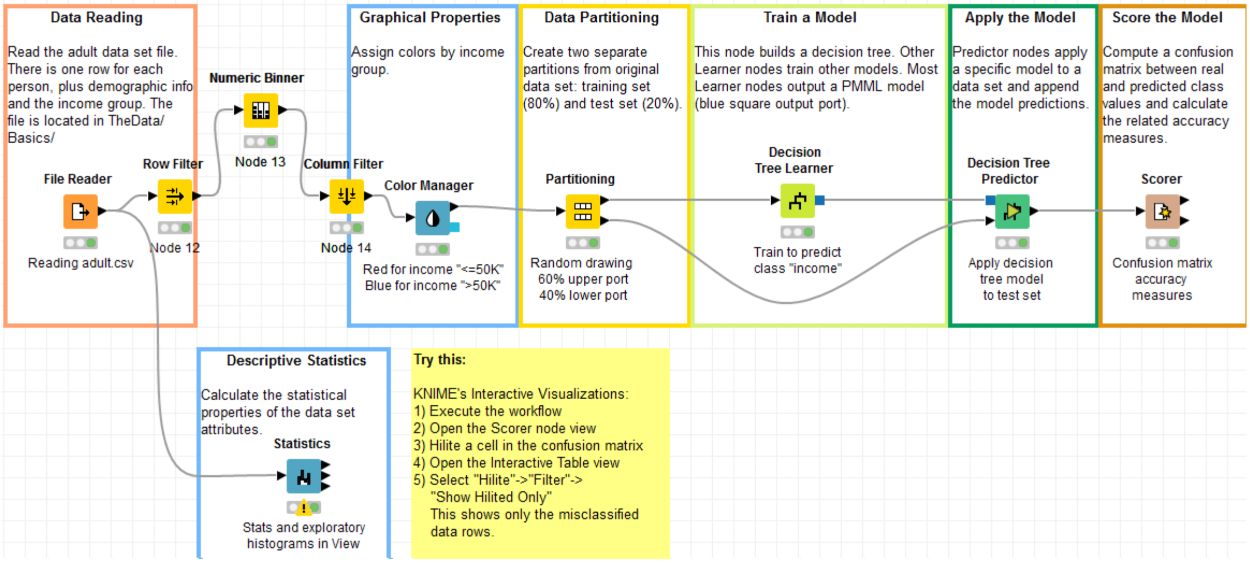
A screenshot of the confusion matrix can be seen below:



As seen in the screenshot above, the overall accuracy of the model is
207 HighRollers have been predicted correctly.
10 HighRollers have been predicted incorrectly.
285 PennyPinchers have been predicted correctly.
63 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 (93.2%) and other platformType users are PennyPinchers (6.8%).

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
Total game clicks	The total game clicks is the total number of times a user has clicked in the game. This is the result of adding the count of game clicks per user in the game_clicks dataset
Total add clicks	The add clicks is the total number of times a user has clicked adds during all the user's history in the game. This is the result of adding the add_clicks dataset per user
Revenue	The revenue is the sum of the average price of the add the user has clicked. This is the result of adding the price in the buy_clicks dataset per user.

Training Data Set Creation

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

- Read of the following files game-clicks.csv and buy-clicks.csv
- Feature Selection of the selected columns
    - user_purchases = buyclicks_df[['userId','price']]
    - user_hits = gameclicks_df[['userId','isHit'']]
- Sum the target columns per User Id
    - hits_per_user = user_hits.groupby('userId').sum()
    - revenue_per_user = user_purchases.groupby('userId').sum()
- Merge the datasets into one for the analysis
    - combined_df = hits_per_user.merge(revenue_per_user, on='userId')

Dimensions of the final data set:
cluster_df.shape
(546, 2)

# of clusters created: 3

Cluster Centers

Cluster #	Center
1	0.10815052, 1.97281387
2	-0.32965623, -0.36089697
3	2.27468978, -0.0926773

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that users produce average number of hits but made the most purchases.

Cluster 2 is different from the others in that the users who produce less hits and also made less purchases

Cluster 3 is different from the others in that the users who produce the most hits but made the less purchases

Below you can see the summary of the train data set:

Print the center of these two clusters:

In [38]:
```
centers = model.clusterCenters()
centers
```

Out[38]:
```
[array([-0.32965623, -0.36089697]),
 array([0.10815052, 1.97281387]),
 array([ 2.27468978, -0.0926773 ])]
```

Recommended Actions

Action Recommended	Rationale for the action
Increase adds to users who play a lot	It was seen that users who play a lot are also the users who spend less and click less on adds, this adds increase will promote this users to spend more and therefore increase the revenue
Show higher price add adds to users who spend more	The users who spend the more show also that they do not play too much so they usually play and always spend, thus, by showing them the more valuable adds first, we can increase the revenue faster

Graph Analytics

Modeling Chat Data using a Graph Data Model

A graph model is used to illustrate the interactions between users. A user(node) can interact(creating edges) with others by chatting with others in a session.

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: chat_create_team_chat.csv
A line is added to this file when a player creates a new chat with their team.
Example:
userid, teamid, TeamChatSessionID, timestamp
559,48,6288,14567

File: chat_item_team_chat.csv
Creates nodes labeled ChatItems. Column 0 is User id, column 1 is the TeamChatSession id, column 2 is the ChatItem id (i.e., the id property of the ChatItem node), column 3 is the timestamp for an edge labeled "CreateChat". Also create an edge labeled "PartOf" from the ChatItem node to the TeamChatSession node. This edge should also have a timeStamp property using the value from Column 3.
Example:
userid, teamchatsessionid, chatitemid, timestamp
1956,6299,6305,1464235803

File: 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.
Example:
userid, TeamChatSessionID, teamstamp
559,6288,12345

File: chat_leave_team_chat.csv
ERD table: chat_leave_team_chat
Creates an edge labeled "Leaves" from User to TeamChatSession. The columns are the User id, TeamChatSession id and the timestamp of the Leaves edge.
Example:
userid, teamchatsessionid, timestamp
1244,6821,1464241204.0

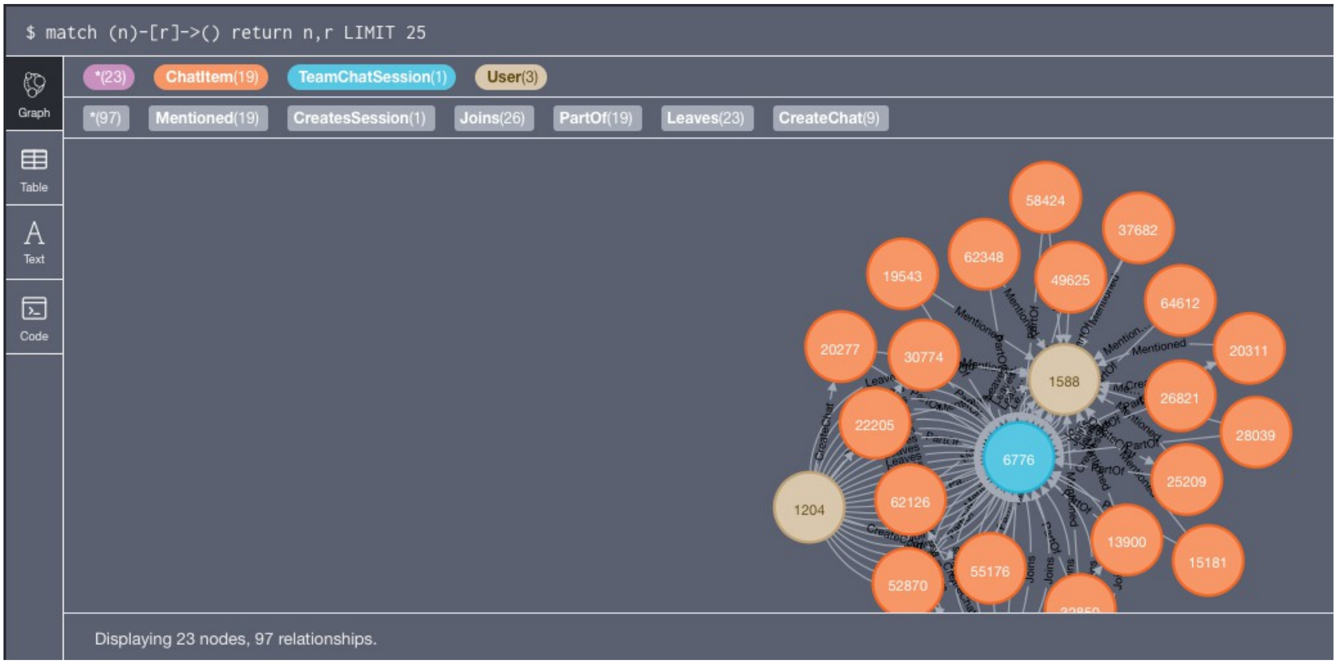
File: chat_mention_team_chat.csv
Creates an edge labeled "Mentioned". Column 0 is the id of the ChatItem, column 1 is the id of the User, and column 2 is the timeStamp of the edge going from the chatItem to the User.
Example:
ChatItem, userid, timeStamp
6349,2508

File: 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.
Example:
chatid1, chatid2,timestamp
6326,6305,21564

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

```
CREATE CONSTRAINT ON (u:User) ASSERT u.id IS UNIQUE;
CREATE CONSTRAINT ON (t:Team) ASSERT t.id IS UNIQUE;
CREATE CONSTRAINT ON (c:TeamChatSession) ASSERT c.id IS UNIQUE;
CREATE CONSTRAINT ON (i:ChatItem) ASSERT i.id IS UNIQUE;

LOAD CSV FROM "file:///chat-data/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)

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

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

LOAD CSV FROM "file:///chat-data/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)

LOAD CSV FROM "file:///chat-data/chat_mention_team_chat.csv" AS row
MERGE (i:ChatItem {id: toInteger(row[0])})
MERGE (u:User {id: toInteger(row[1])})
MERGE (i)-[:Mentioned{timeStamp: row[2]}]->(u)

LOAD CSV FROM "file:///chat-data/chat_respond_team_chat.csv" AS row
MERGE (i:ChatItem {id: toInteger(row[0])})
MERGE (w:ChatItem {id: toInteger(row[1])})
MERGE (i)-[:ResponseTo{timeStamp: row[2]}]->(w)
```

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.

a. There are 9 chats involved.

b. 5 users participated in this chain

```
$ match p = (u)-[:ResponseTo*]->(v) return length(p) order by length(p) desc limit 1
```

length(p)
9

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

count(distinct u)
5

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.

Chattiest Users
We count the number of edges/chats user created.
Chattiest Teams
We count the total number of chats the team made.
Only team 52 is part of the top 10 chattiest teams. Hence, most of the chattiest users are not in the chattiest teams.

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

u.id	count(i)
394	115
2067	111
1087	109
209	109
554	107
1627	105
999	105
516	105
461	104
668	104

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

t.id	count(c)
null	3850163
82	1324
185	1036
112	957
18	844
194	836
129	814
52	788
136	783
146	746

How Active Are Groups of Users?

Firstly, we create the InteractsWith relationship.
Second, we get the neighbours of each of the chattiest users
Finally, we query each user in the neighbourhood and sum up the results and calculate the coefficient by dividing the sum with the number of possible interactions.

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
209	0.95
394	1.00
2067	0.93