Week-1

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	ERD table: AdClicks 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	ERD table: InAppPurchases 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

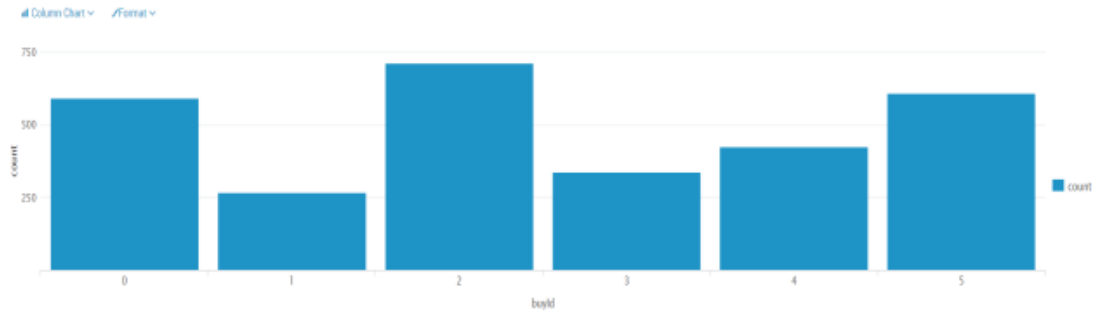
users.csv	ERD table: User This file contains a line for each user playing the game.	**timestamp**: when user first played the game. **id**: the user id assigned to the user. **nick**: the nickname chosen by the user. **twitter**: the twitter handle of the user. **dob**: the date of birth of the user. **country**: the two-letter country code where the user lives.
team.csv	ERD table: Team 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	ERD table: TeamAssignment 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.	**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	ERD table: LevelEvent A line is added to this file each time a team starts or finishes a level in the game	**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	ERD table: User_Sessions 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	ERD table: GameClicks A line is added to this file each time a user performs a click 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

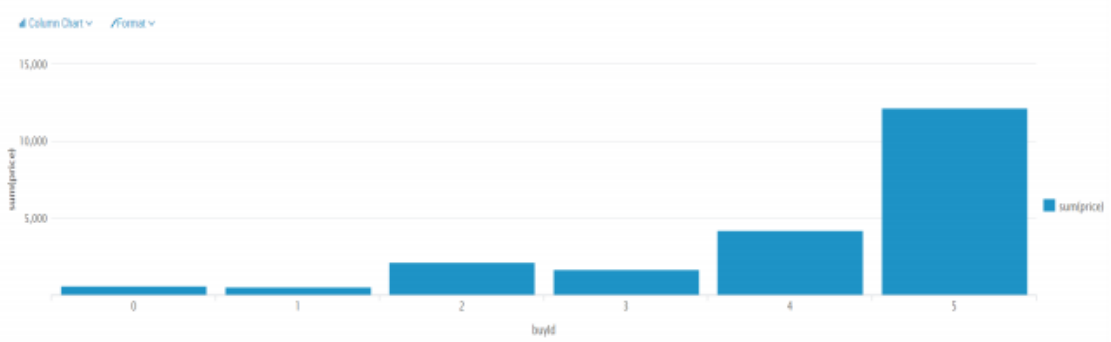
Aggregation

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:

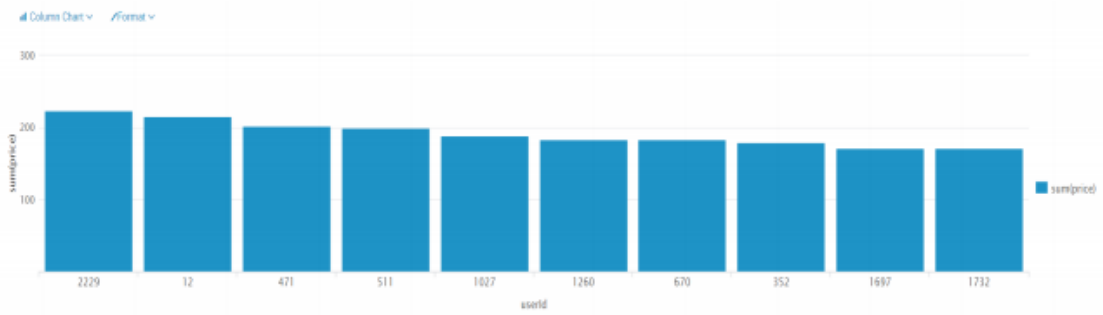


A histogram showing how much money was made from each item:



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

Week-2

Data Preparation

Analysis of combined_data.csv

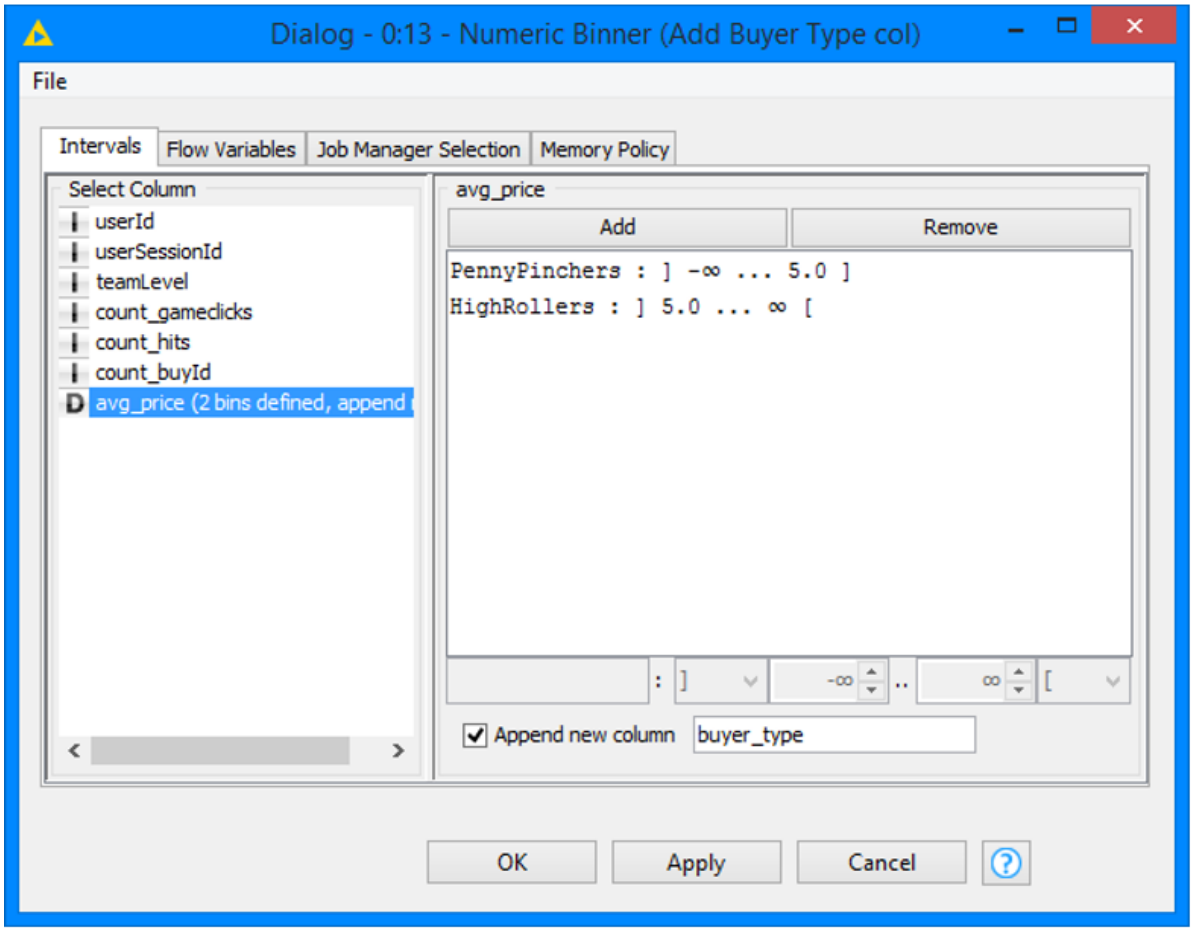
Sample Selection

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:

Data with categorized attribute is:



This "prediction" attribute is created using Rule Engine node with rule as follows:
avg_price > 5.0 => "HighRollers" //users paying more than $5.0

avg_price <= 5.0 => "PennyPinchers" //users paying less than or equal to $5.0

The creation of this new categorical attribute was necessary because our aim in this assignment is predicting which user is likely to purchase big-ticket items while playing Catch the Pink Flamingo. This information is valuable for Eglence since in-app purchases are a major source of revenue.

Attribute Selection

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

Attribute	**Rationale for Filtering**
userSessionId	This attribute just shows the session of the user. It has nothing to do with whether the user will be a high payer or not.
teamLevel	The level of the team does't help predict whether user is a high payer or not.
count_hits	No. of hits too, doesn't help in predicting user's paying capacity.

Data Partitioning and Modeling

The data was partitioned into **train and test** datasets.

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

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

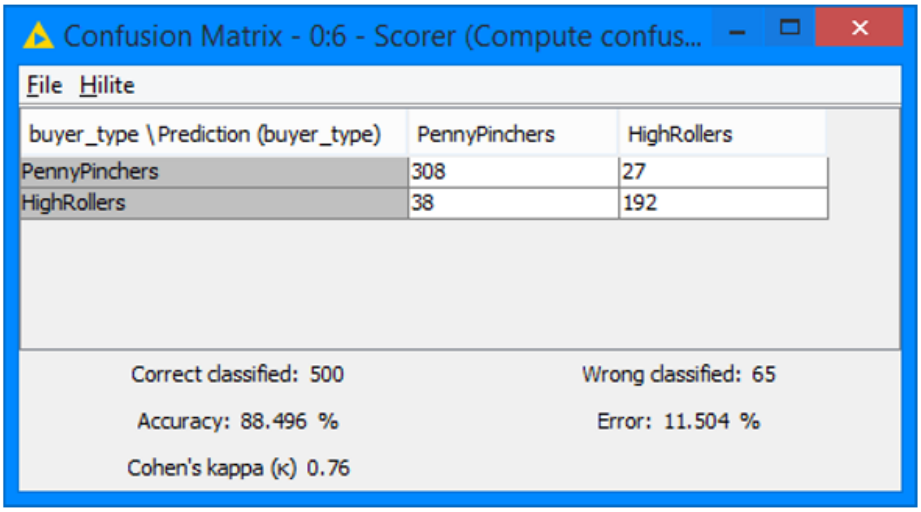
This is important because testing on whole dataset will be more cumbersome and less predictive than to train 60% of it and develop a model to predict and test the remaining 40% dataset.

When partitioning the data using sampling, it is important to set the random seed because it will ensure that we obtain the same partitions every time we run the model.

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

Evaluation

A screenshot of the confusion matrix can be seen below:

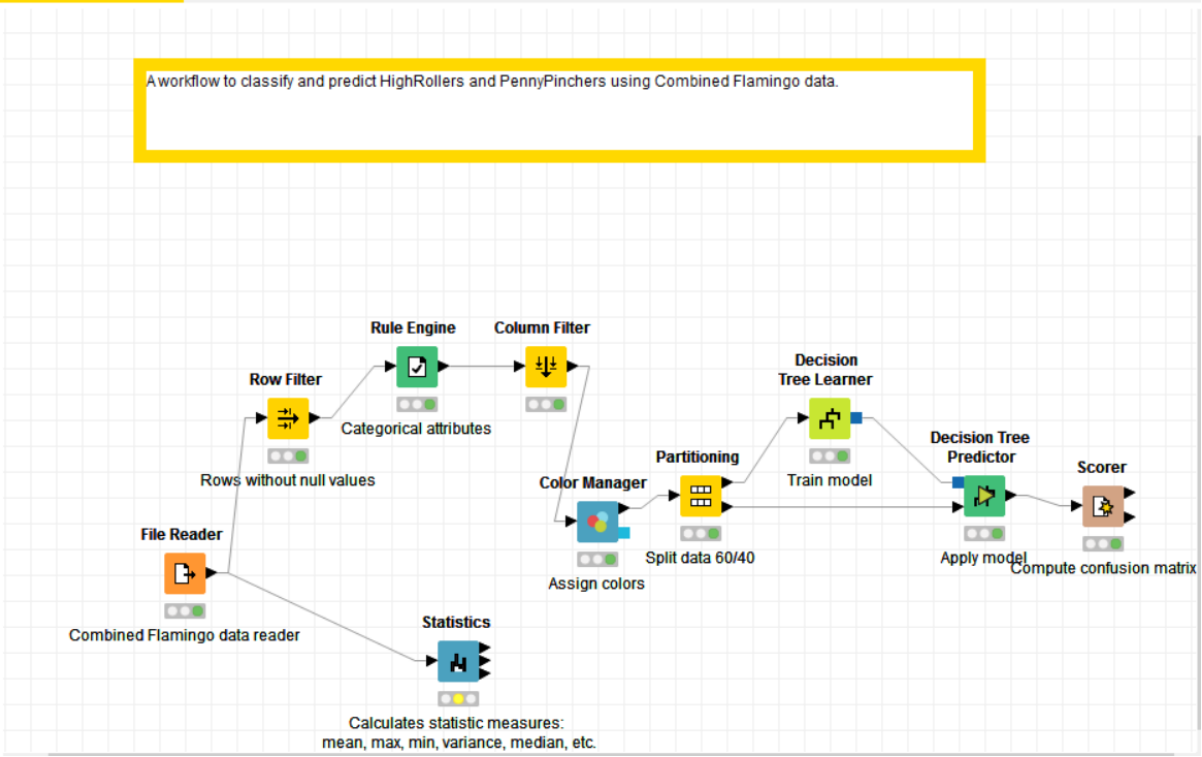


As seen in the screenshot above, the overall accuracy of the model is **88.496%**

- PennyPinchers/PennyPinchers: 308 actual Penny Pinchers were correctly predicted as Penny Pinchers.
- PennyPinchers/HighRollers: 27 PennyPinchers were incorrectly predicted as HighRollers.
- HighRollers /PennyPinchers: 38 HighRollers were incorrectly predicted as PennyPinchers.
- HighRollers / HighRollers: 192 actual HighRollers were correctly predicted as HighRollers

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?
The model predicts a whether a user is a HighRoller or a PennyPincher based on the in app purchases. The platform seems to strongly determine if a user is a High Roller. The majority of iPhone users are High Rollers whereas in other platforms, the majority of users are Penny Pinchers.

Specific Recommendations to Increase Revenue
1. Personalized suggestions to users for items to purchase (Especially to iPhone users as they will be willing to spend if suggestions are even more related to their liking).
2. Cheap combo offers for users of other platforms i.e., except iPhone users to attract PennyPinchers.

Week-3

Clustering using Spark:

Attribute Selection

Attribute	Rationale for Selection
Number of Purchases	The number of purchases by each user shows the purchasing record of the user and thus will help in predicting future purchasing power.
game clicks per hour by each user	Decides how active the player is.
Range of purchases	For learning the range of items a user purchases, whether he is confined to similar ones or tries different ones.

Cluster Centers

Cluster #	Cluster Center
1	(1.8950, 2.9843)
2	(2.53726, 6.13456)
3	(3.98475, 10.69453)

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that these users buy the least range of items and also the least no. of purchases (about 1.89 items).

Cluster 2 is different from the others in that they have average purchases of 2.53 items and make purchases spread out in more range.

<Optional Fill In> Cluster 3 is different from the others in that it includes users making maximum purchases and maximum range too.

Training Data Set Creation

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

```
   rangeOfPurchasedItems  numPurchases
0                      3             9
1                      3             5
2                      2             6
3                      2            10
4                      4            13
```

Dimensions of the training data set (rows x columns) : 546x2

# of clusters created: 3

Recommended Actions

Action Recommended	Rationale for the action
Customized ads for all users	The users having a lesser range of purchases should be recommended more similar items rather than of variety and those with huge range of purchases should be recommended a nice variety of items. This will increase purchases of both type of users.
Increase variety of products	As the range of purchasing items increases, the number of items purchased increases too. Therefore, increasing variety will increase range and in turn, increase purchases.
Consider team purchases	Consider the purchases made by a team as a whole and suggest combo packs of items according to their demands.

Week-4

Graph Analytics

Modelling Chat Data using a Graph Data Model

The interactions involved with chats occurring in the game are modelled using this Graph Data Model. The nodes are ChatItem (a chat text), Team (to which users belong), TeamChatSession (the session involving a user on the team) and User. Edges signify timestamps of a chat event. The actual chat text is not used in this model. Edges involving users include: CreatesSession, Joins, Leaves or CreateChat. TeamChatSession is OwnedBy a team. A ChatItem is part of a TeamChatSession and ResponseTo is an edge between two ChatItems. An additional edge created for analysis denotes InteractsWith between Users and does not have a timestamp.

Creation of the Graph Database for Chats

Describe the steps you took for creating the graph database. As part of these steps

i) **Schema for 6 files:**

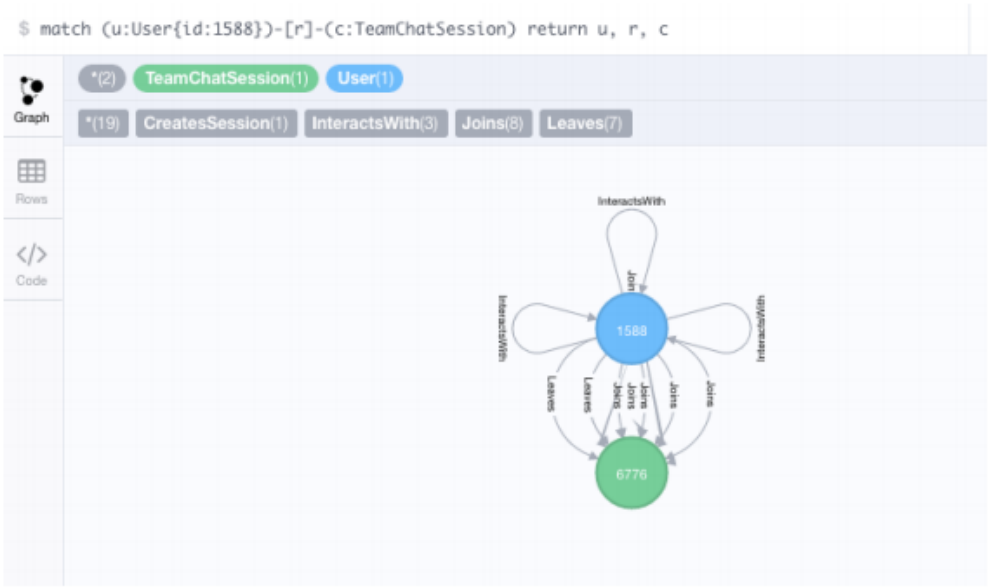
File Name	Description
chat_create_team_chat.csv	Userid: ID of user Teamid: ID of team TeamChatSessionID: Id of team chat session Timestamp: time stamp of team chat session
chat_item_team_chat.csv	Userid: ID of user TeamChatSessionID: Id of team chat session Chatitemid: ID of chat item created by user Timestamp: time stamp of created chat item
chat_join_team_chat.csv	Userid: ID of user TeamChatSessionID: Id of team chat session Timestamp: time stamp when user joins team chat session
chat_leave_team_chat.csv	Userid: ID of user TeamChatSessionID: Id of team chat session Timestamp: time stamp when user leaves team chat session
chat_mention_team_chat.csv	Userid: ID of user ChatItem: ID of chat item created by user timestamp: time stamps when user mentions chat item
chat_respond_team_chat.csv	chatid1: Chat ID of first chat chatid2: Chat ID of the second chat Timestamp: time stamp when chat was replied.

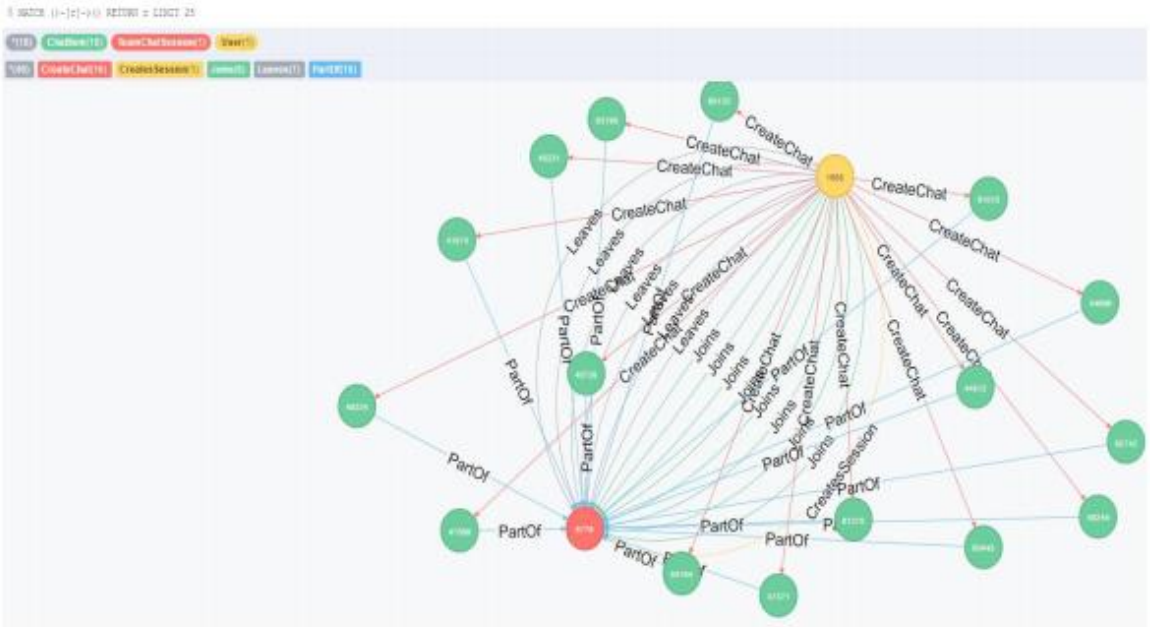
ii) **Loading the files**

Files are loaded into neo4j using LOAD CSV command without headers and columns are merged appropriately.
Sample load command: For **chat_create_team_chat .csv ,**

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

The query used to find the longest conversation path length was:

MATCH path=(chatItem1)-[:ResponseTo*]->(chatItem2) RETURN length(path) ORDER BY length(path) DESC LIMIT 1

Result: 10

In order to obtain the unique users, the following query was used:

MATCH path=(chatItem1)-[:ResponseTo*]->(chatItem2) with length(path) as ln,nodes(path) as pn ORDER BY ln DESC LIMIT 1 match p=(u)-[:CreateChat]->(c) where c in pn return distinct u.id

Result: 5 users

u.id
1192
1978
1153
853

u.id
1514

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

Chattiest Users

The query to obtain the chattiest users is:

Match(a)-[:CreateChat]->(c)

Return a.id, count(c)

Order by count(c) Desc limit 3

Users	Number of Chats
394	115
2067	111
209	109

user.id	numChats
394	115
2067	111
209	109
1087	109
554	107
1627	105
516	105
999	105
461	104
668	104

Chattiest Teams

Query to obtain chattiest teams is:

Match (chat)-[:PartOf*]->(teamSession)-[:OwnedBy*]->(team) return team, count(team) order by count(team) desc limit 10

Teams	Number of Chats
82	1324
185	1036
112	957

team.id	numChats
82	1324
185	1036
112	957
18	844
194	836
129	814
52	788
136	783
146	746
81	736

The chattiest user with **USER ID 999** was part of chattiest **team 52**.

How Active Are Groups of Users?

Most Active Users (based on Cluster Coefficients

For finding the active groups of users, ratio of existing edges between a user and his neighbours to possible edges that could exist between the user's neighbours is used.

Query used is:

Match p=(u)-[:CreateChat*]->(c) with u, count(u) as chat order by count(u) desc limit 10 match p=(u)-[:InteractsWith]-(b) with u, collect(distinct b.id) as neighbours match m,n where n.id in neighbours and m.id in neighbours with neighbours, u,n,m, case when (n)◊(m) then 1 else 0 end as value return u, sum(value), neighbours

User ID	Coefficient
209	0.9523
554	0.9047
1087	0.8
516	0.9523
394	1
999	0.8667
1627	0.7857
461	1
668	0.7
2067	0.7857