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
users.csv	A line for every user in 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.
ad-clicks.csv	Creates an entry when an user clicks an ad	timestamp: when the click occurred. txId: a unique id (within adclicks.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	Creates an entry when an user makes an in-app-purchase	timestamp: when the purchase was made. txId: a unique id (within

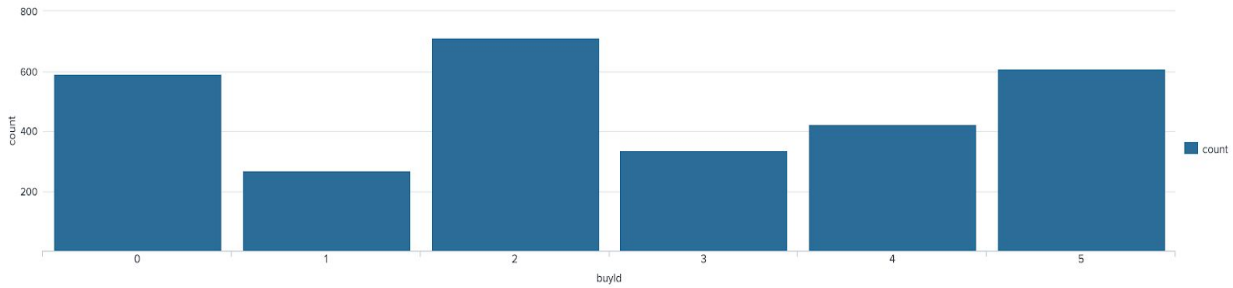
		buyclicks.log) for the purchase userSessionId: the id of the user session for the user who made the purchase team: the current team id of the user who made the purchase userId: the user id of the user who made the purchase buyId: the id of the item purchased price: the price of the item purchased
team.csv	A line for every team 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-assignmets.csv	Creates an entry when an user joins a team	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	Creates an entry when a team starts or finishes a level	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	Indicates when an user starts and stop playing in a session	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	Creates an entry for all the clicks made	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

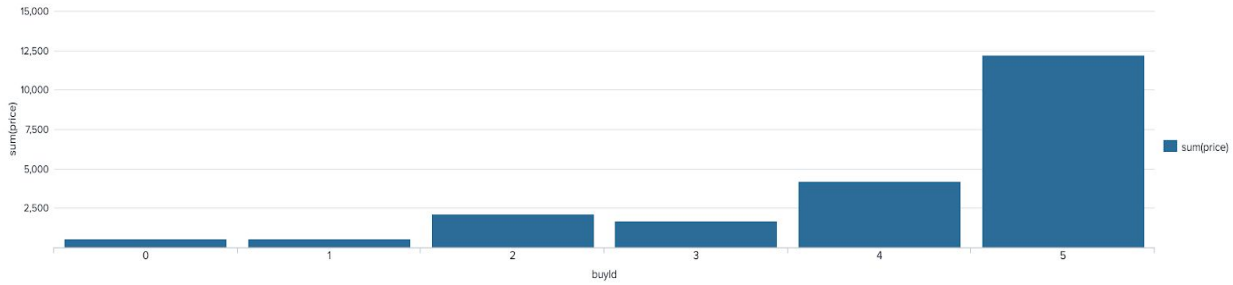
Aggregation

Amount spent buying items	$21,407
Number of unique items available to be purchased	6

A histogram showing how many times each item is purchased:

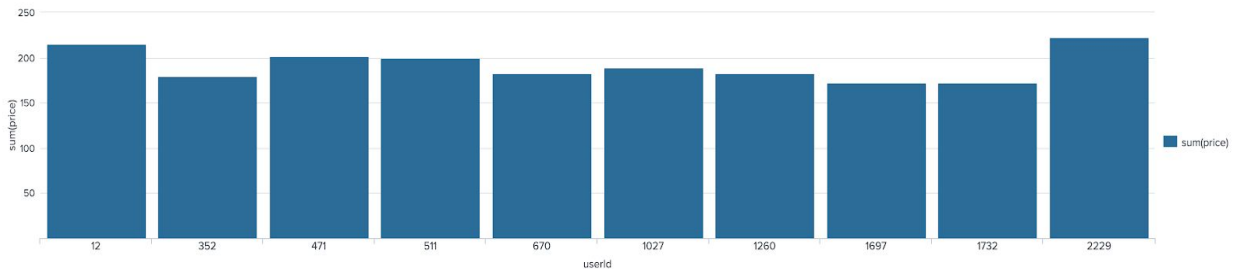


A histogram showing how much money was made from each item:



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	13.0%
2	12	iPhone	11.6%
3	471	iPhone	14.5%

Data Preparation

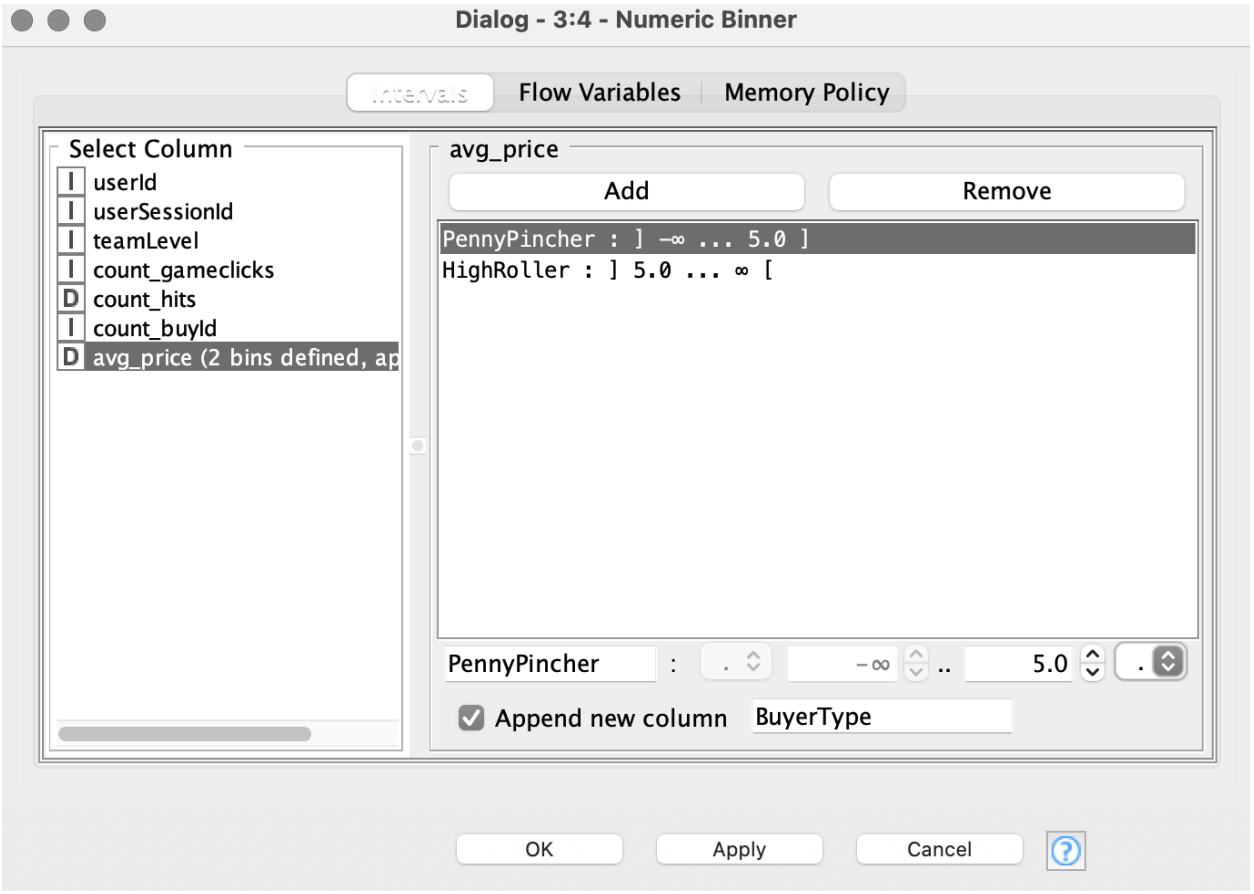
Analysis of combined_data.csv

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:



The attribute consists of two classes. PennyPinchers spend less or equal than $5 on average and HighRollers spend more than $5 on average.

The creation of this new categorical attribute was necessary because it will give us insight into the users' buying behaviors. This will be the category our model will try to predict.

Attribute Selection

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

Attribute	**Rationale for Filtering**
avg_price	This column was used to build the target column. Keeping would mean a data leakage
userId	Not influential in the model
sessionId	Not influential in the model

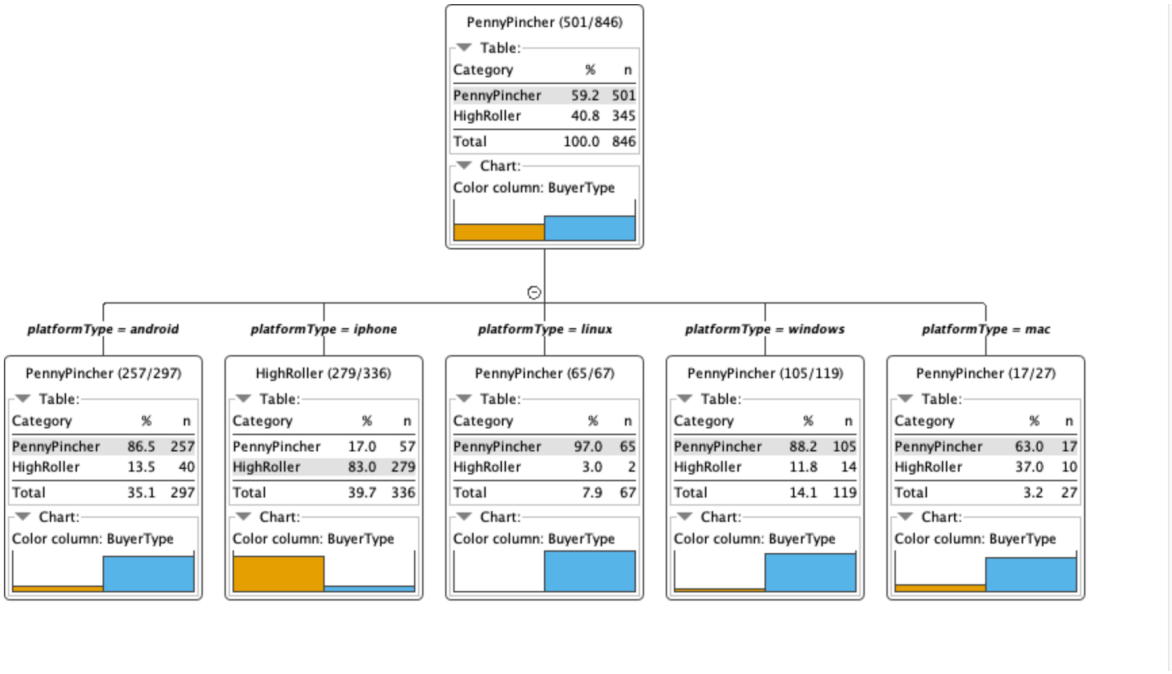
Data Partitioning and Modeling

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 that lets us evaluate how good the model is performing and lets us identify if the model is overfitted

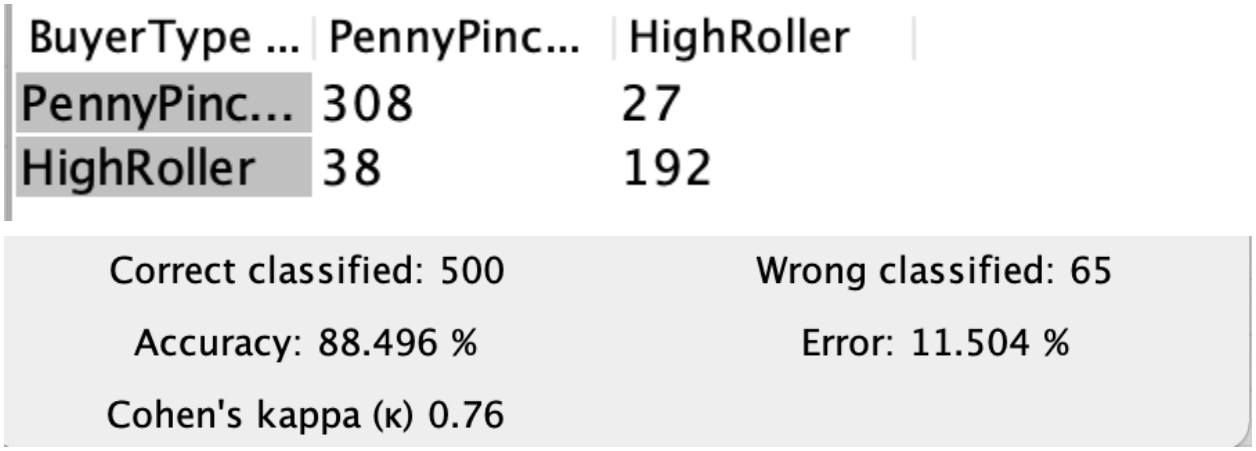
When partitioning the data using sampling, it is important to set the random seed because that leads to a standardized result.

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



Evaluation

A screenshot of the confusion matrix can be seen below:



BuyerType ...	PennyPinc...	HighRoller
PennyPinc...	308	27
HighRoller	38	192

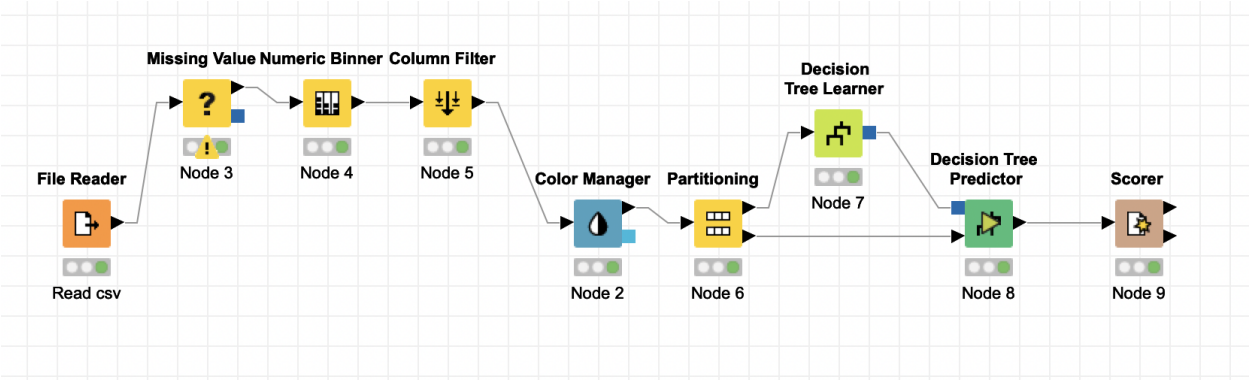
Correct classified: 500	Wrong classified: 65
Accuracy: 88.496 %	Error: 11.504 %
Cohen's kappa (κ) 0.76	

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

When the true BuyerType is PennyPincher, the model classified it correctly 308 times and incorrectly 38 times. When the true BuyerType is HighRoller, the model classified it correctly 192 times and incorrectly 27 times.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?
The most important feature is the PlatformType. iPhone users tend to be HighRollers while the users of the remaining platforms tend to be PennyPinchers.

Specific Recommendations to Increase Revenue
1. Focus iPhone advertisement in high-ticket products.
2. Focus Windows, Android, Linux, Mac advertisement in cheap transactions. Lowering the prices in sales may increase the amount of buyers

Attribute Selection

features_used = [‘totalAdClicks’, ‘revenue’, ‘BuyClicks’]

Attribute	**Rationale for Selection**
totalAdClicks	It shows the amount of times the user clicked an advertisement that translates into the company’s profit.
revenue	It shows the total revenue from the ads shown to the user
BuyClicks	It shows the amount of times the user clicked and made a purchase.

Training Data Set Creation

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

	totalAdClicks	totalBuyClicks	totalRevenue
0	44	9	21.0
1	10	5	53.0
2	37	6	80.0
3	19	10	11.0
4	46	13	215.0

Dimensions of the final data set: 543 x 3
# of clusters created: 3

Cluster Centers

The code used in creating cluster centers is given below:

```
kmeans = KMeans(k=3, seed=1)
model = kmeans.fit(scaledData)
transformed = model.transform(scaledData)
```

```
centers = model.clusterCenters()
centers
```

Cluster centers formed are given in the table below

Cluster #	Center
1	41.07, 10.29, 145.51
2	34.28, 6.45, 67.22
3	26,30 4.48, 17.07

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that users with the highest revenue and adclick are also the ones who make more buyclicks

Cluster 2 is different from the others in that the users who generate medium revenue also click in ads and buy in an average way

Cluster 3 is different from the others in that the users who make the least revenue are also the ones who click the least in ads and buy.

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

	totalAdClicks	totalBuyClicks	totalRevenue
0	44	9	21.0
1	10	5	53.0
2	37	6	80.0
3	19	10	11.0
4	46	13	215.0

Recommended Actions

Action Recommended	Rationale for the action
Increase the price of ads shown at members of the first cluster	It was seen that this group clicks and buys a lot. So increasing the price of third party ads targeting this group would increase the company's revenue.
Show 'Special Offers' to members of the third cluster	If we show special offers to the users who clicked the least, we may increase the amount of ads they click in and therefore the revenue.

Graph Analytics

Modeling Chat Data using a Graph Data Model

The graph consists of Users who create ChatItems that are part of TeamChatSessions. These TeamChatSessions can be crated by Users and owned by Teams. The ChatItems can mention Users or respond to other ChatItems. Users can join or leave TeamChatSessions during time. All these relations include 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) Write the schema of the 6 CSV files

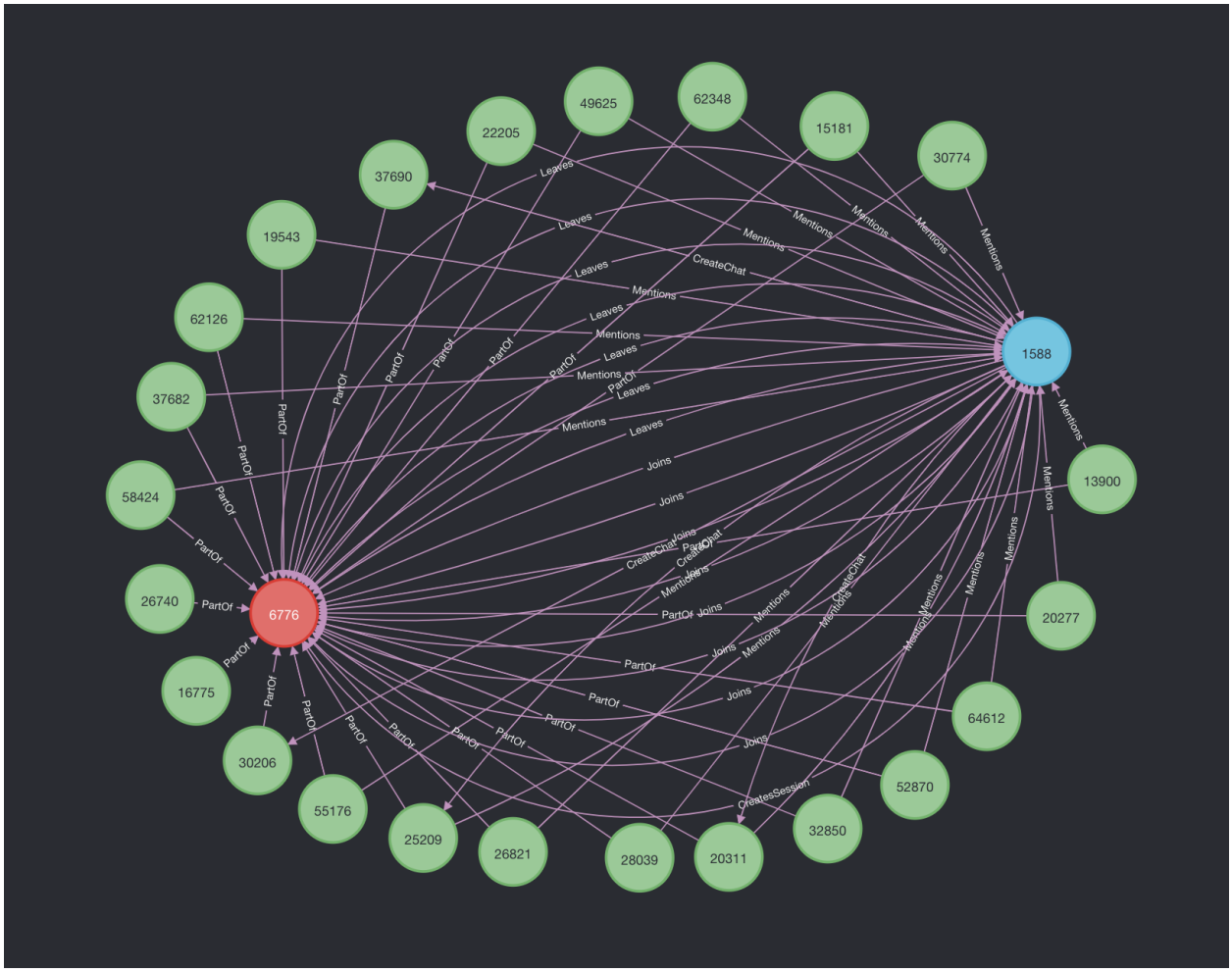
Dataset	Columns
chat_create_team_chat	userid teamid teamchatsessionid timestamp
chat_item_team_chat	userid teamchatsessionid chatitemid timestamp
chat_join_team_chat	userid teamchatsessionid timestamp
chat_leave_team_chat	userid teamchatsessionid timestamp
chat_mention_team_chat	chatitemid userid timestamp
chat_respond_team_chat	chatid1 chatid2 timestamp

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

```
1 LOAD CSV FROM "file:///chat_create_team_chat.csv" AS row
2 MERGE (u:User {id: toInteger(row[0])}) MERGE (t:Team {id: toInteger(row[1])})
3 MERGE (c:TeamChatSession {id: toInteger(row[2])})
4 MERGE (u)-[:CreatesSession{timeStamp: row[3]}]→(c)
5 MERGE (c)-[:OwnedBy{timeStamp: row[3]}]→(t)
```

The first line imports the csv archive and reads it row by row. The second line creates User and Team nodes and sets id attribute as an integer from an specific column of each row. The third line does the same process for TeamChatSession nodes. The fourth line creates an edge named CreateSession from Users to TeamChatSession nodes with a timestamp attribute. The fifth line does the same process for edges OwnedBy from TeamChatSession to Team

iii) **Present a screenshot of some part of the graph you have generated.**

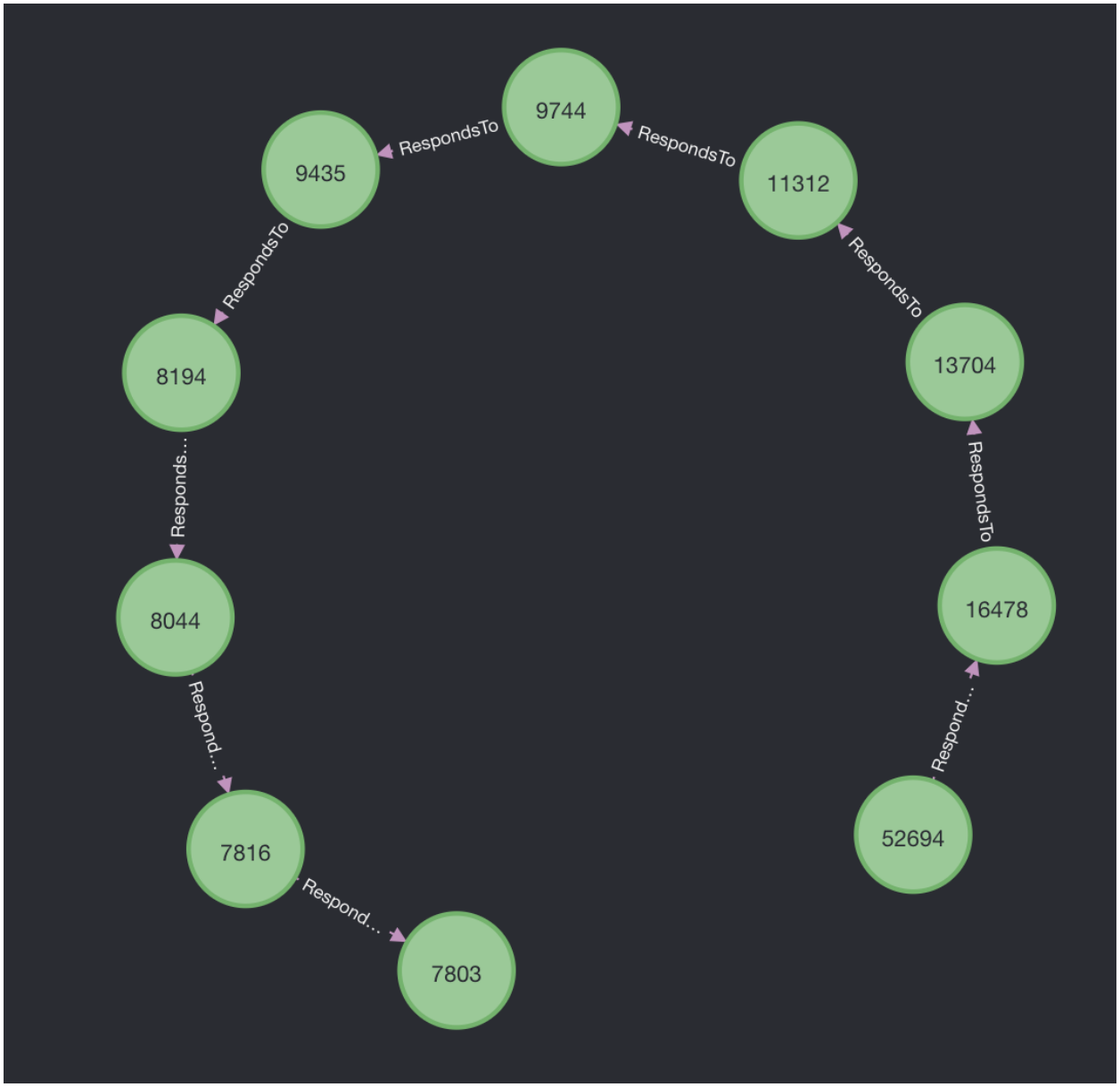


Finding the longest conversation chain and its participants

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

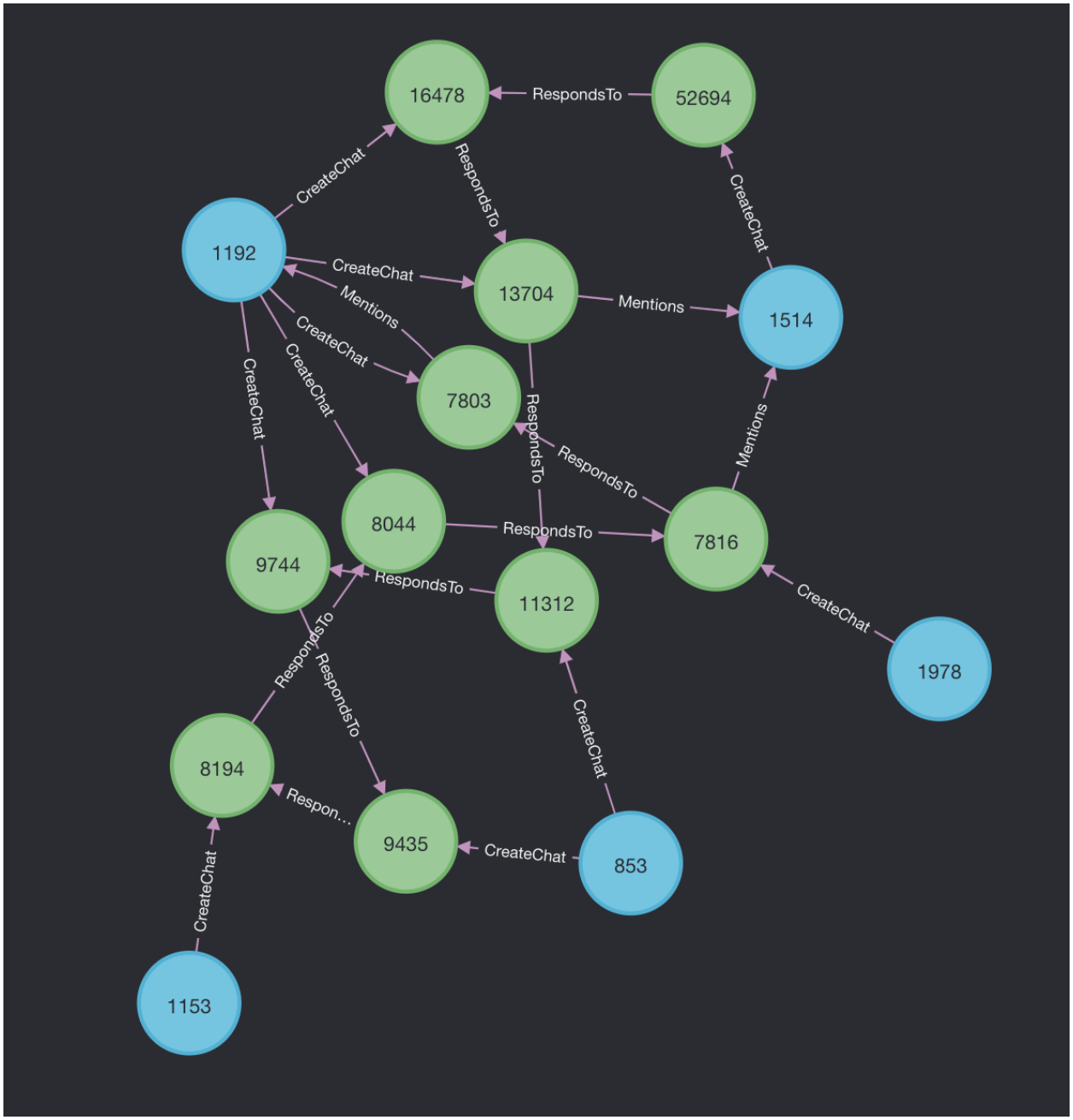
answer.

```
neo4j$ match p=()-[:RespondsTo*]→() return p, length(p) order by length(p) desc limit 1
```



The query results in a longest path of length 9, meaning the longest chat chain consists of 10 ChatItems.

```
neo4j$ match p=()-[:RespondsTo*9]→() with p match q=(u:User)-[:CreateChat]→(c:ChatItem) where (c in nodes(p))
       return count(distinct u)
```



The query return a total of 5 distinct users involved in the longest chat chain

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

```
neo4j$ match (u:User)-[r:CreateChat]→() return u.id, count(r) order by count(r) desc limit 10
```

	u.id	count(r)
1	394	115
2	2067	111
3	1087	109
4	209	109
5	554	107
6	1627	105
7	516	105
8	999	105
9	668	104
10	461	104

Users	Number of Chats
394	115
2067	111
1087	109

Chattiest Teams

```
neo4j$ match ()-[r:PartOf]→(c) return c.id, count(r) order by count(r) desc limit 10
```

Nueva pestaña

	c.id	count(r)
1	6792	1324
2	6783	1036
3	6925	957
4	6791	844
5	6974	836
6	6889	814
7	6780	788
8	6819	783
9	6850	746
10	6778	736

Teams	Number of Chats
6792	1324
6783	1036
6925	957

Finally, present your answer, i.e. whether or not any of the chattiest users are part of any of the chattiest teams.

```
neo4j$ match(u:User)⟶(:ChatItem)⟶(:TeamChatSession)⟶(t:Team) where u.id in
       [394,2067,209,1087,554,516,1627,999,668,461] and t.id in [82,185,112,28,194,129,52,136,146,81] return
       distinct u.id, t.id
```

	u.id	t.id
1	999	52

The query searches for the chattiest users that belong to the chattiest teams. It only shows a pair consisting of User 999 belonging to Team 52. It shows that in general the 10 chattiest Users do not belong to the 10 chattiest Teams.

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.

Most Active Users (based on Cluster Coefficients)

This query creates relations Interacts from an User who mention another User in a ChatItem

```
neo4j$ match (u1:User)←[:Mentions]-()←[:CreateChat]-(u2:User) merge (u1)←[:Interacts]-(u2)
```

This query creates relations Interacts from an User who responds to another User in a ChatITem

```
neo4j$ match (u1:User)-[:CreateChat]→()-[:RespondsTo]→()←[:CreateChat]-(u2:User) merge (u1)-[:Interacts]→(u2)
```

This query deletes all self loops for Interacts relation

```
neo4j$ match (u)-[r:Interacts]→(u) delete r
```

This query searches for a specific User's neighbours and evaluates how many connections are there between those neighbours. Then computes a coefficient relative to the max amount of possible relations.

```
neo4j$ match (u:User{id:394})-[:Interacts]→(n) with collect(n.id) as nb, count(n) as k match (n1)-[i:Interacts]→
       (n2) where (n1.id in nb) and (n2.id in nb) return count(i)/(k*(k-1)*1.0)
```

User ID	Coefficient
394	0.7500
2067	0.9333
1087	0.7333