**Catch The Pink Flamingo Analysis**

**Technical Appendix**

**Produced by Eren Berkay Ünlü**



**Data Exploration with Splunk**

**Data Set Overview**

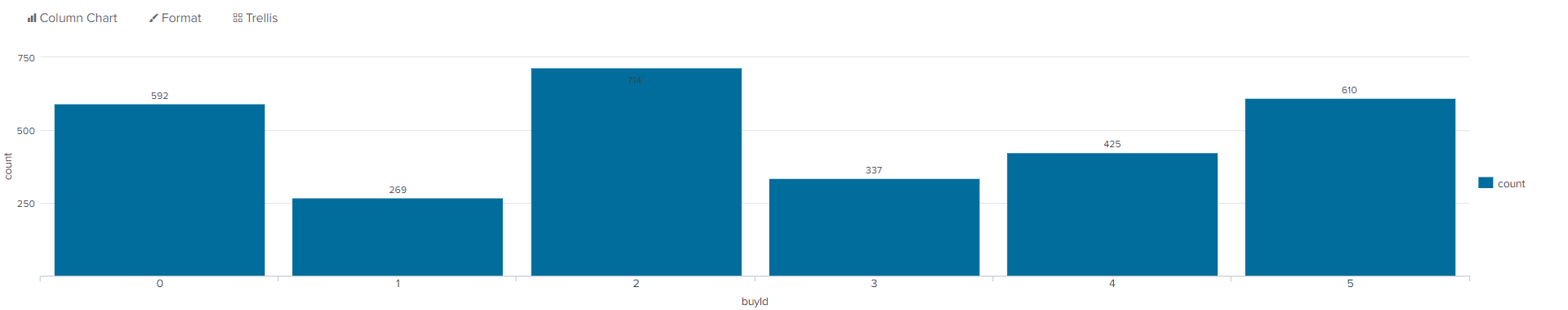
The table below lists each of the files available for analysis with a short description of what is found in each one.

|  |  |  |
| --- | --- | --- |
| **File Name** | **Description** | **Fields** |
| ad-click.csv | Database of each click a user performs on an advertisement | timestamp: time when the advertisement click performed  userId: id of the user that performed the click  userSessionId: id of the session for the user that performed the click  txId: id for the click  adId: id of the advertisement that is clicked  adCategory: category of the advertisement that is clicked |
| buy-clicks.csv | Database of each click a user performs to purchase | timestamp: time when the purchase made  userId: id of the user that made the purchase  userSessionId: id of the session for the user that made the purchase  txId: id for the purchase  team: id of the team for the user that made the purchase  buyId: id of the item that is purchased  price: price of the item that is purchased |
| game-clicks.csv | Database of each click a user performs in game | timestamp: time when the in game click performed  userId: id of the user that performed the click  userSessionId: id of the session for the user that performed the click  clickId: id for the in game click  teamId: id of the team for the user that performed the click  teamLevel: level of the the team of the user  isHit: denotes whether the click was on a flamingo or not |
| level-events.csv | Database that holds each level event for a team | timestamp: time when the event occured  eventId: id of the event  eventType: type of the event  teamId: id of the team  teamLevel: level of the team |
| team-assignments.csv | Database that holds a record each time a user joins a team | timestamp: time when the user joined the team  team: id of the team  userId: id of the user  assignmentId: id for the assignment |
| team.csv | Database of each team in the game | teamId: id of the team  name: name of the team  teamCreationTime: time when the team was created  teamEndTime: time when the team drop to zero members  strength: a measure for the success of the team  currentLevel: current level of the team |
| user-session.csv | Database that holds each session a user is in | timestamp: time when the session is created or finished  userId: id of the current user  userSessionId: if of the session  teamId: id of the team  assignmentId: id for the assignment  sessionType: whether the session is a start session or an end session  teamLevel: level of the the team  platformType: type of the platform for user during the session |
| users.csv | Database of the users | timestamp: time when the user is created  userId: id of the user  nick: nickname of the user  twitter: twitter handler for the user  dob: date of birth of the user  country: two letter country code where the user lives |

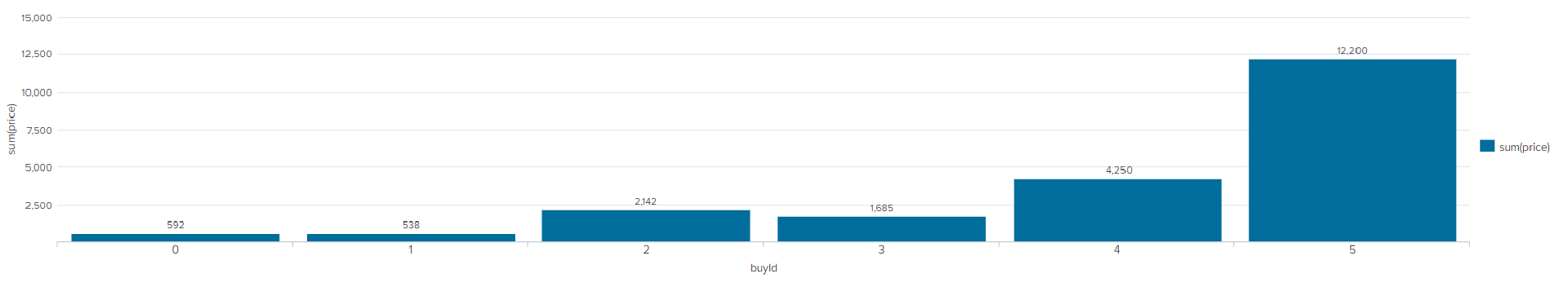
**Aggregation**

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

A histogram showing how many times each item is purchased:

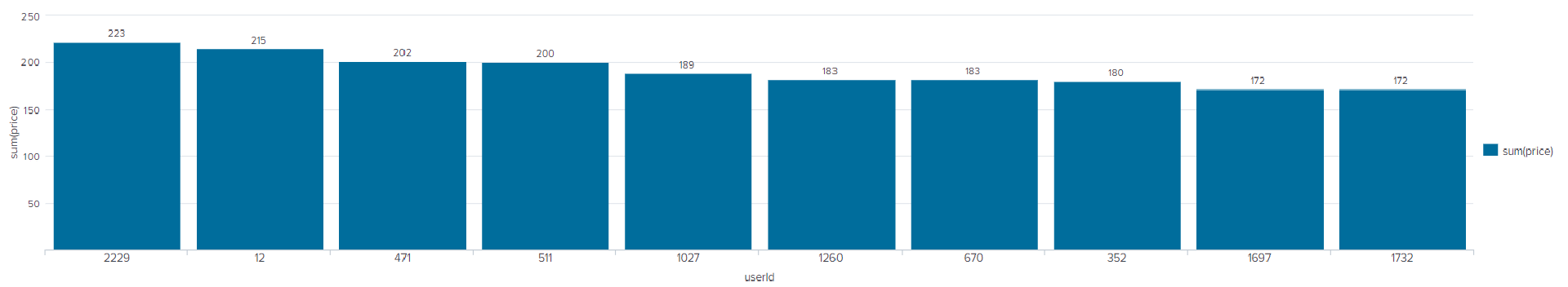


A histogram showing how much money was made from each item:



**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 | 13.068 |
| 3 | 471 | iPhone | 14.504 |

# https://forum-cdn.knime.com/uploads/default/original/2X/f/f05de1913205c68b8a92ff0eb7a1c9e58b3ea9eb.png

# Classification with KNIME

**Data Preparation**

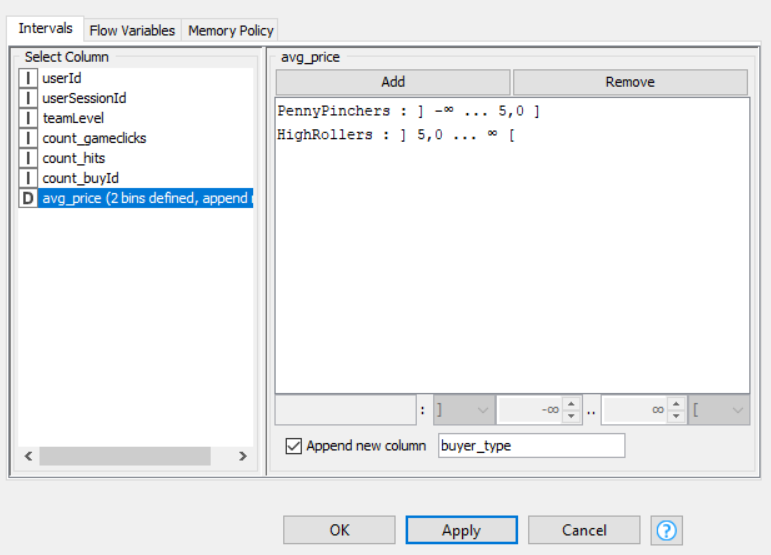
Analysis of combined\_data.csv

**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:



A new categorical attribute “buyer\_type” is created. We defined two categories namely HighRollers (buyers of items that cost more than $5.00) and PennyPinchers (buyers of items that cost less than $5.00).

The creation of this new categorical attribute was necessary because it simplifies the classification of users and it is the base that we are going to use to build our decision tree.

**Attribute Selection**

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

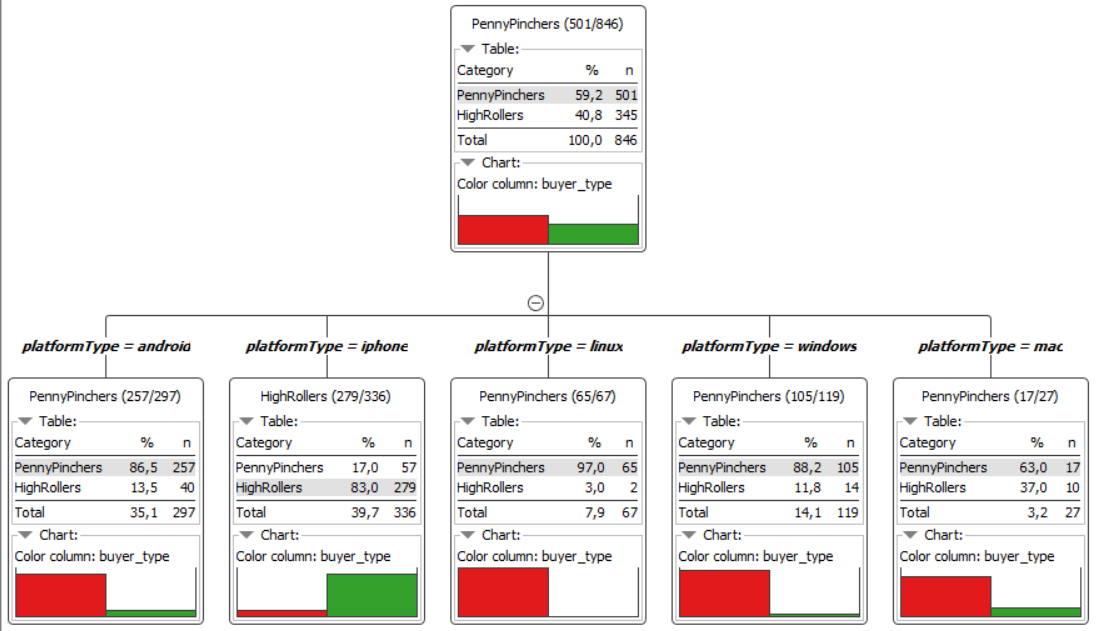
|  |  |
| --- | --- |
| **Attribute** | **Rationale for Filtering** |
| userId | It is filtered since this attribute has no effect on what we are looking for. |
| userSessionId | It is filtered since this attribute has no effect on what we are looking for. |
| Avg\_price | It is filtered since a new categorical attribute “buyer\_type” is created which made this attributte not necessary anymore. |

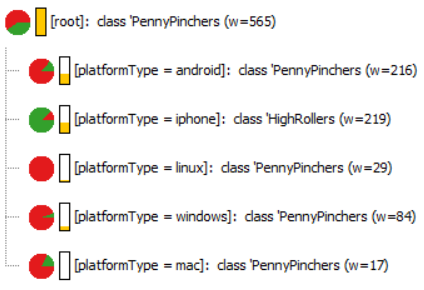
**Data Partitioning and Modeling**

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 train data set is used for creating the decision tree, test data allows us to see the accuracy of the model.

When partitioning the data using sampling, it is important to set the random seed because we can get the same data every time we execute the model.

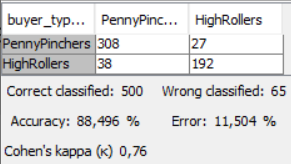
A screenshot of the resulting decision tree can be seen below:





**Evaluation**

A screenshot of the confusion matrix can be seen below:

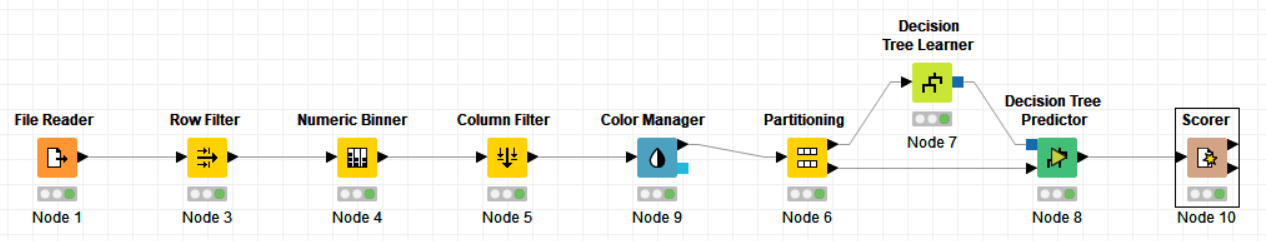


As seen in the screenshot above, the overall accuracy of the model is **88,496 %**

For “buyer\_type” PennyPinchers our model classified correctly 308 and incorrectly 27 of them. For “buyer\_type” HighRollers our model classified correctly 192 and incorrectly 38 of them.

**Analysis Conclusions**

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

Users who are HighRollers seem to use iPhone. On the other hand, in toher platforms most of the users are PennyPinchers.

|  |
| --- |
| **Specific Recommendations to Increase Revenue** |
| 1. Target suggestions and promotions especially for iOS users. |
| 2. A lot of cheap offers for other platforms so that they purchass more and more. |



# Clustering Analysis with Spark MLlib

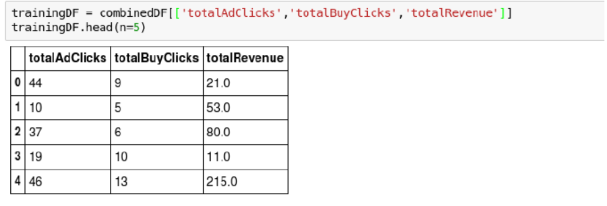
**Attribute Selection**

features\_used = ["revenue", “totalAdClicks”]

|  |  |
| --- | --- |
| **Attribute** | **Rationale for Selection** |
| totalRevenue | how much the users spend in the game will give us a monetary value of that user |
| totalBuyClicks | how often the users click to buy will show the engagement |
| totalAdClicks | how often the users click on the ad will show how likely they will spend money |

**Training Data Set Creation**

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

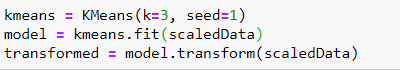


Dimensions of the final data set: 543 rows x 3 columns

# of clusters created: 3

**Cluster Centers**

The code used in creating cluster centers is given below:



Cluster centers formed are given in the table below:

|  |  |
| --- | --- |
| **Cluster #** | **Center** |
| 1 | 36.44134078, 926.11731844, 46.96648045 |
| 2 | 24.98746082, 357.95924765, 35.06583072 |
| 3 | 32.35555556, 2310.64444444, 39.42222222 |

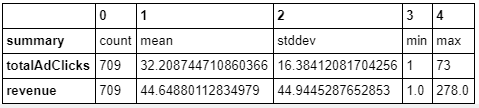
These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that users with the highest revenue and adclick are not the ones who are playing the most but an intermediate result in game clicks.

Cluster 2 is different from the others in that the users who play the less also produces the less ad revenue and click count

Cluster 3 is different from the others in that the users who play the most are not the ones who produce the most revenue, the revenue in the middle along with the ad click count

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



print(km\_model.centers)

[array([ 36.44134078, 926.11731844, 46.96648045]), array([ 24.98746082, 357.95924765, 35.06583072]), array([ 32.35555556, 2310.64444444, 39.42222222])]

**Recommended Actions**

|  |  |
| --- | --- |
| **Action Recommended** | **Rationale for the action** |
| Increase ads 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 ads. If we increase ads to users who play a lot, it will promote these users to spend more and therefore increase the revenue |
| Show higher price ads to users who spend more | If we show higher price ads to users who spend more, we can increase the revenue faster.The users who spend the more also do not play too much, thus by showing them the more valuable ads first, we can increase the revenue faster |



**Graph Analytics with Neo4j**

**Modeling Chat Data using a Graph Data Model**

The graph model is a network based on the chat interactions between users. A user in a team can create a chat session and then create chat item in that session. It is possible for a user to mention another user. Edges signify timestamp for the chat items.

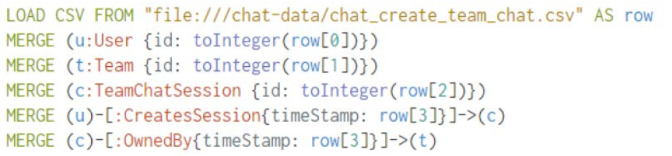
**Creation of the Graph Database for Chats**

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

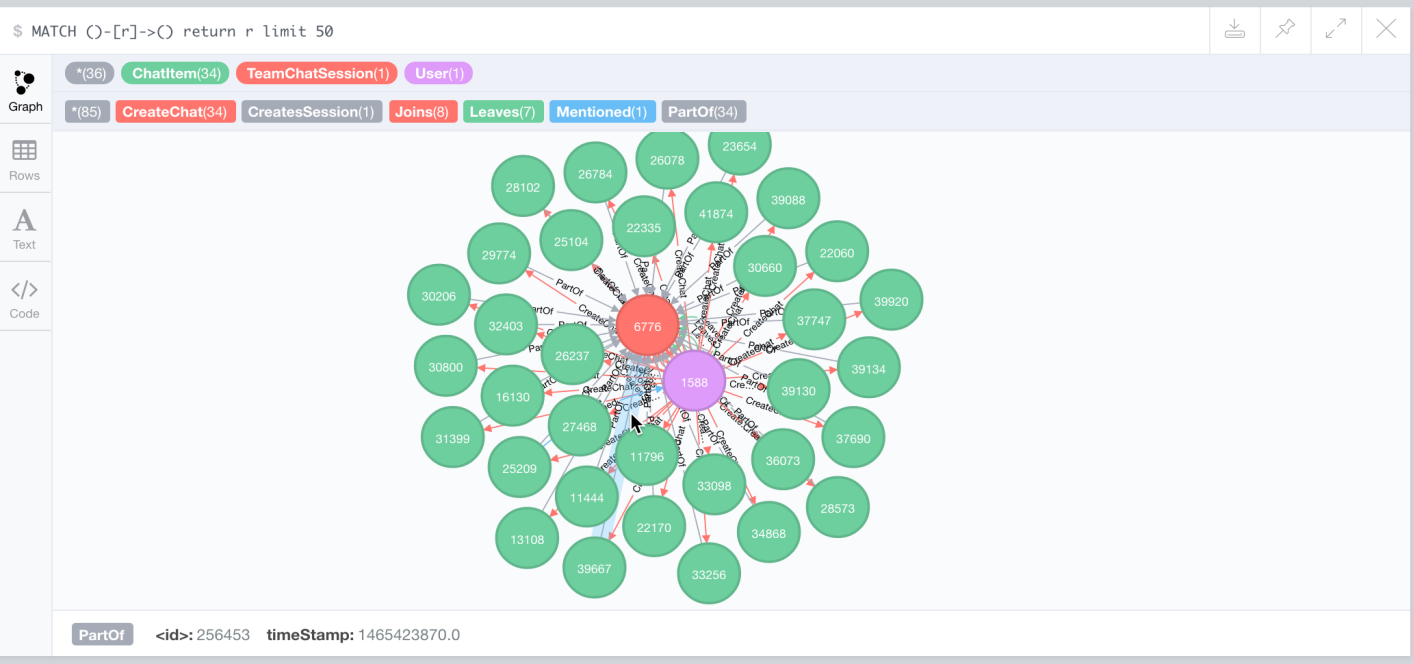
1. The schema of the 6 CSV files

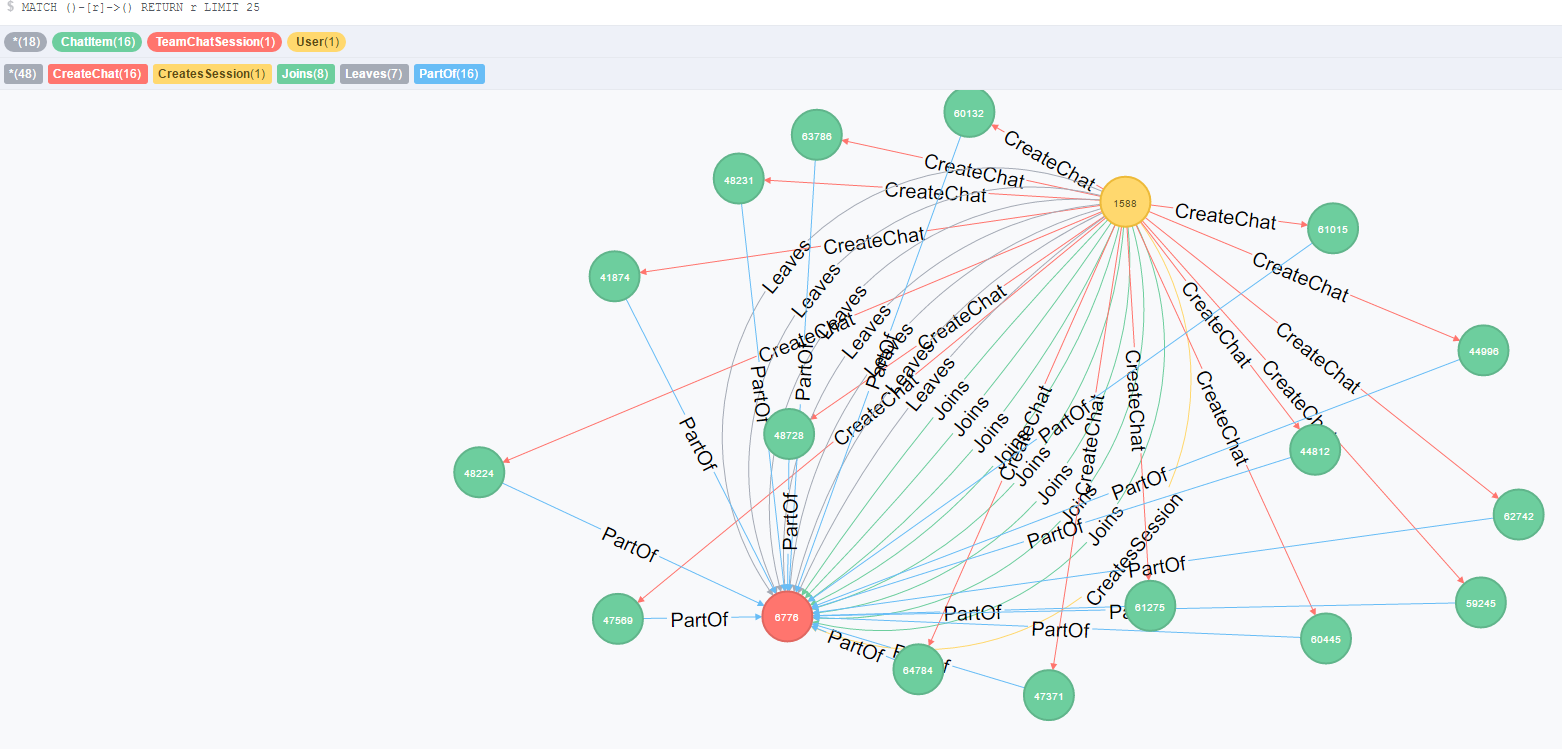
|  |  |
| --- | --- |
| **File name** | **Fields** |
| **chat\_create\_team\_chat.csv** | userID  teamID  teamChatSessionID  timestamp |
| **chat\_join\_team\_chat.csv** | userID  teamChatSessionID  timestamp |
| **chat\_leave\_team\_chat.csv** | userID  teamChatSessionID  timestamp |
| **chat\_item\_team\_chat.csv** | userID  teamChatSessionID  chatItemID  timestamp |
| **chat\_mention\_team\_chat.csv** | userID  chatItemID  timestamp |
| **chat\_respond\_team\_chat.csv** | chatItemID\_1  chatItemID\_2  timestamp |

1. The loading process and a sample LOAD command



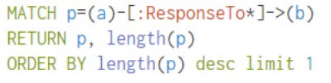
1. A screenshot of some part of the graph you have generated. The graphs include clearly visible examples of most node and edge types.





**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.



The longest conversation chain in the chat data has a path lenght of 9 and there are 10 chats in this path.

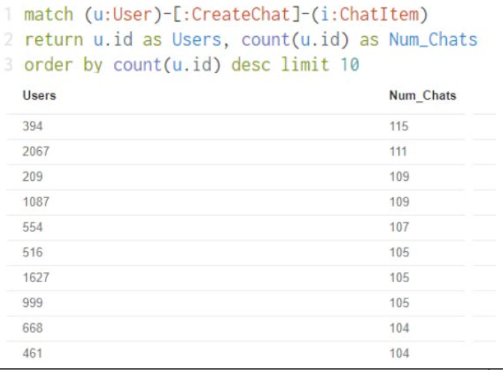


Count the number of distinct users in the longest path. The result is 5.

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

**Chattiest Users**

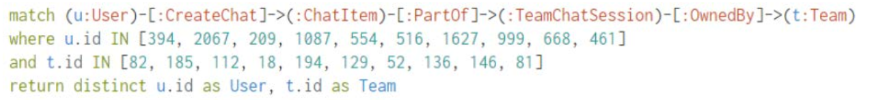
|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 209 | 109 |



**Chattiest Teams**

|  |  |
| --- | --- |
| **Teams** | **Number of Chats** |
| 82 | 1324 |
| 185 | 1036 |
| 112 | 957 |





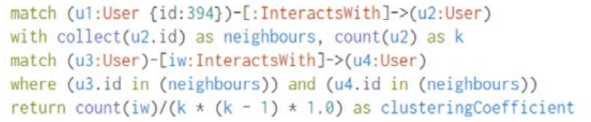
This query is used to investigate if the most chattiest users are part of any of the chattiest teams. It returns one result as the user with the userID 999 is part of the team with teamID 52.

**How Active Are Groups of Users?**

C:\Users\monster\Desktop\coursera\BD Capstone\g1.PNG

C:\Users\monster\Desktop\coursera\BD Capstone\g2.PNG

C:\Users\monster\Desktop\coursera\BD Capstone\g3.PNG



We first connected the mentioned users. Then connected the user responses with the chat creator. Eliminated all self interactions. Finally calculated the cluster coefficients.

**Most Active Users (based on Cluster Coefficients)**

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 394 | 0.9167 |
| 2067 | 0.7679 |
| 209 | 0.9524 |