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	Data on in-app purchases made by Flamingo app players. 1 line for each 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 (equals teamid found in other flies) **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 which plays/played 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**: reflects how well a team is playing as a whole **currentLevel**: unknown quantity – ideally would represent the current level of the team, but is set to 1 for every team inconsistently with level-events.csv
team-assignments.csv	Data on when a user joins a team. 1 line for each time a user joins. A user can be in at most 1 team at a time.	**timestamp**: when the user joined the team. **team**: the id of the team (equals teamid found in other flies) **userId**: the id of the user **assignmentId**: a unique id for this assignment
level-events.csv	Data on when a team starts or finishes a level in the game. 1 line for each time a team starts or finishes.	**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:** when start/end of session occurred **userSessionid**: a unique id for the session **userId**: the id of the user **teamId**: the id of the team **assignmentId**: the team assignment id for the user to the team **sessionType**: denotes whether it is the start or end of session **teamLevel**: the level of the team during this session **platformType**: the type of platform of the user during this session

game-clicks.csv	Data from each time a user performs a click in the game. 1 line per click.	**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 **teamLevel**: the current level of the team during this click

Aggregation

<u>Notes</u>:

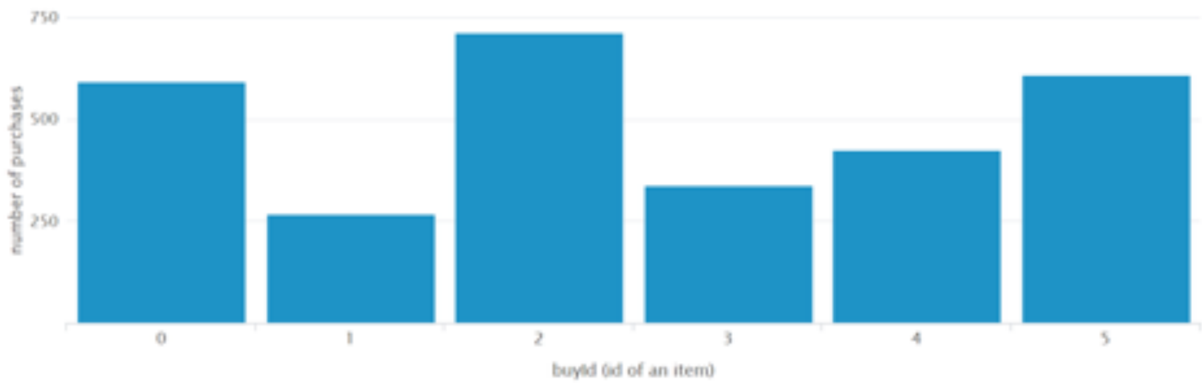
- *Worked with buy-clicks.csv*
- *Amount spent buying items is the total of all “price” values*
- *# Unique items available is the number of unique “buyid” values*

Amount spent buying items	21407
# Unique items available to be purchased	6

A histogram showing how many times each item is purchased:

<u>Notes</u>:

- *Worked with buy-clicks.csv*
- *Count number of occurrences of each “buyid” value to get how many times each item was purchased*

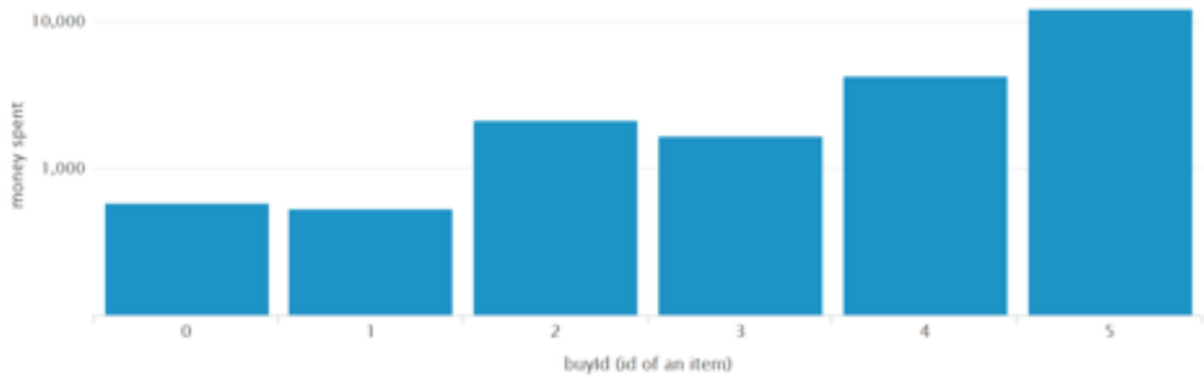


A histogram showing how much money was made from each item:

<u>Notes</u>:

- *Worked with buy-clicks.csv*

- *Sum of all “price” values by “buyid”*

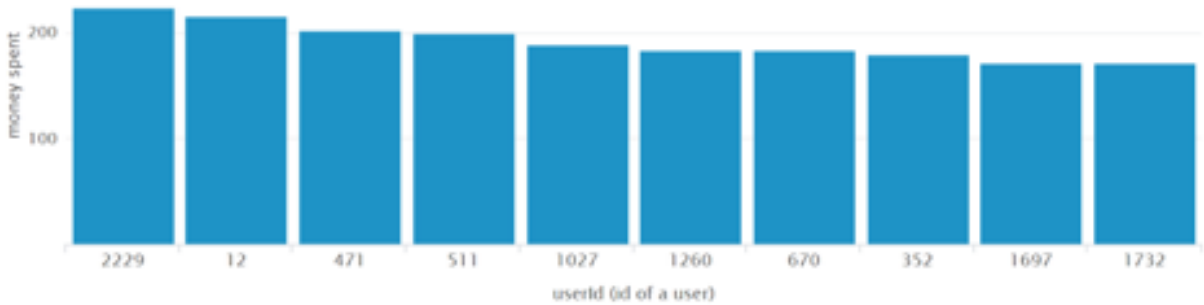


Filtering

A histogram showing total amount of money spent by the top ten users (ranked by how much money they spent).

<u>*Notes*</u>*:*

- *Worked with buy-clicks.csv*
- *Sum of all “price” values by “userid” and sort in descending order*



The following table shows the user id, platform, and hit-ratio percentage for the top three buying users:

<u>*Notes*</u>*:*

- *Get platform from “platformType” in user-session.csv for each userid*
- *Get hit ratio from sum and length of “isHit” values in game-clicks.csv for each userid*

Rank	**User Id**	**Platform**	**Hit-Ratio (%)**
1	2229	iphone	11.59
2	12	iphone	13
3	471	iphone	14.5

Data Preparation

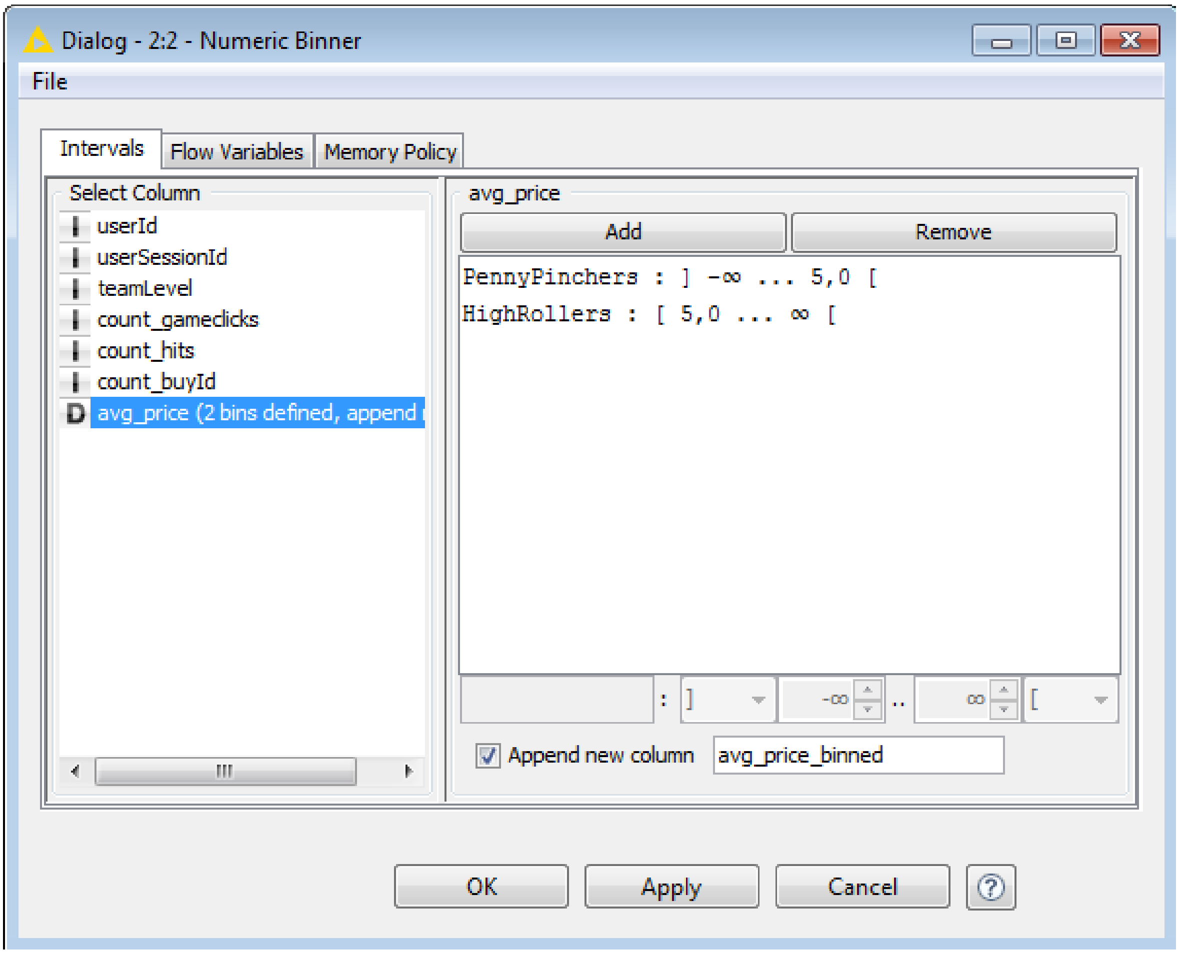
Analysis of combined_data.csv

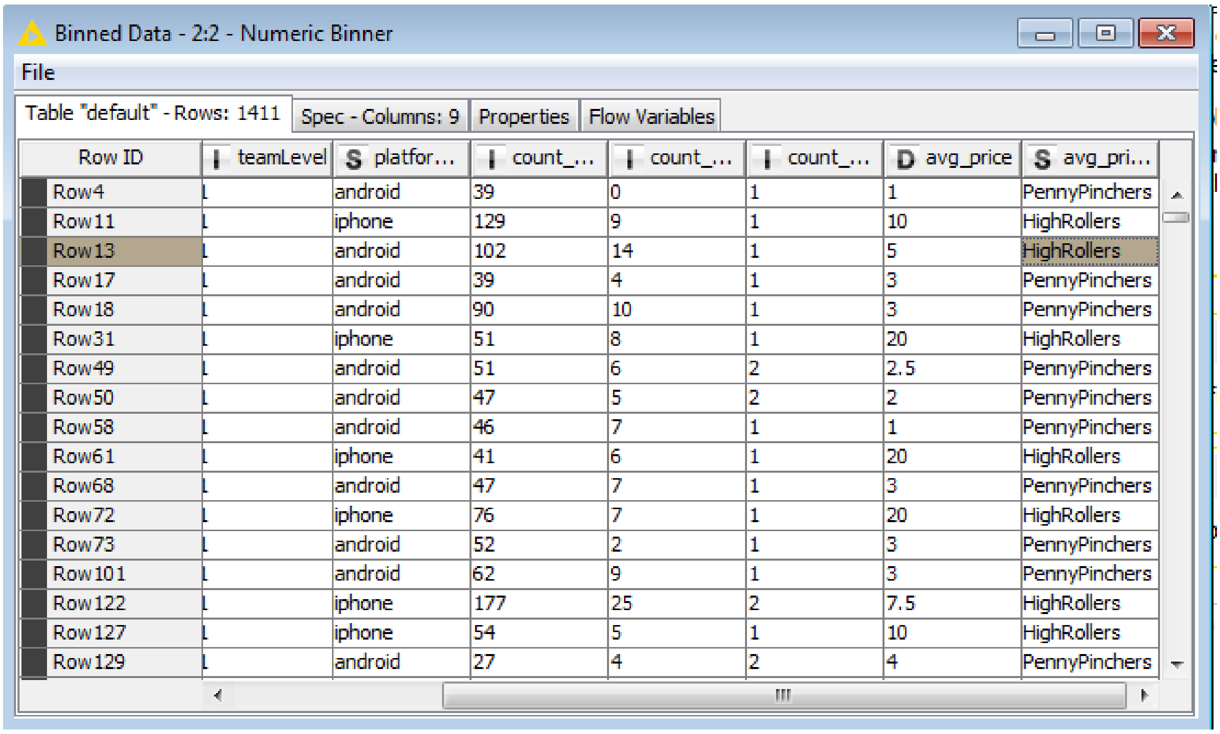
Sample Selection

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 - 2:2 - Numeric Binner

File

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

Row ID	teamLevel	platfor...	count_...	count_...	count_...	avg_price	avg_pri...
Row4	1	android	39	0	1	1	PennyPinchers
Row11	1	iphone	129	9	1	10	HighRollers
Row13	1	android	102	14	1	5	HighRollers
Row17	1	android	39	4	1	3	PennyPinchers
Row18	1	android	90	10	1	3	PennyPinchers
Row31	1	iphone	51	8	1	20	HighRollers
Row49	1	android	51	6	2	2.5	PennyPinchers
Row50	1	android	47	5	2	2	PennyPinchers
Row58	1	android	46	7	1	1	PennyPinchers
Row61	1	iphone	41	6	1	20	HighRollers
Row68	1	android	47	7	1	3	PennyPinchers
Row72	1	iphone	76	7	1	20	HighRollers
Row73	1	android	52	2	1	3	PennyPinchers
Row101	1	android	62	9	1	3	PennyPinchers
Row122	1	iphone	177	25	2	7.5	HighRollers
Row127	1	iphone	54	5	1	10	HighRollers
Row129	1	android	27	4	2	4	PennyPinchers

By use of a numeric binner operator two bins are created by evaluating the avg_price column. Less than 5 evaluate to PennyPichers. Greather than or equal to 5 evaluate to HighRollers.

The creation of this new categorical attribute was necessary because we need to create a model that classify/predict

<u>Attribute Selection</u>

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

Attribute	**Rationale for Filtering**
avg_price	It doesn't make sense to keep it, since we want to predict a category derived from this attribute
userId	This is an identity key with no value attached
userSessionId	This is an identity key with no value attached

Attribute Selection

Attribute	Rationale for Selection
# of ad clicks	This number can be used to detect the actual number of clicks on ads per each user.
Ratio BuyClicks/adClicks	This ration can give some insight on the correlation between number of ad clicks and the actual purchases he or she makes.
Ratio Price/BuyClicks	Can give some hints on the propensity of buying more expensive objects or cheaper ones. It can possibly show that those who spend more buy cheaper objects more often or otherwise.

Graph Analytics

Modeling Chat Data using a Graph Data Model

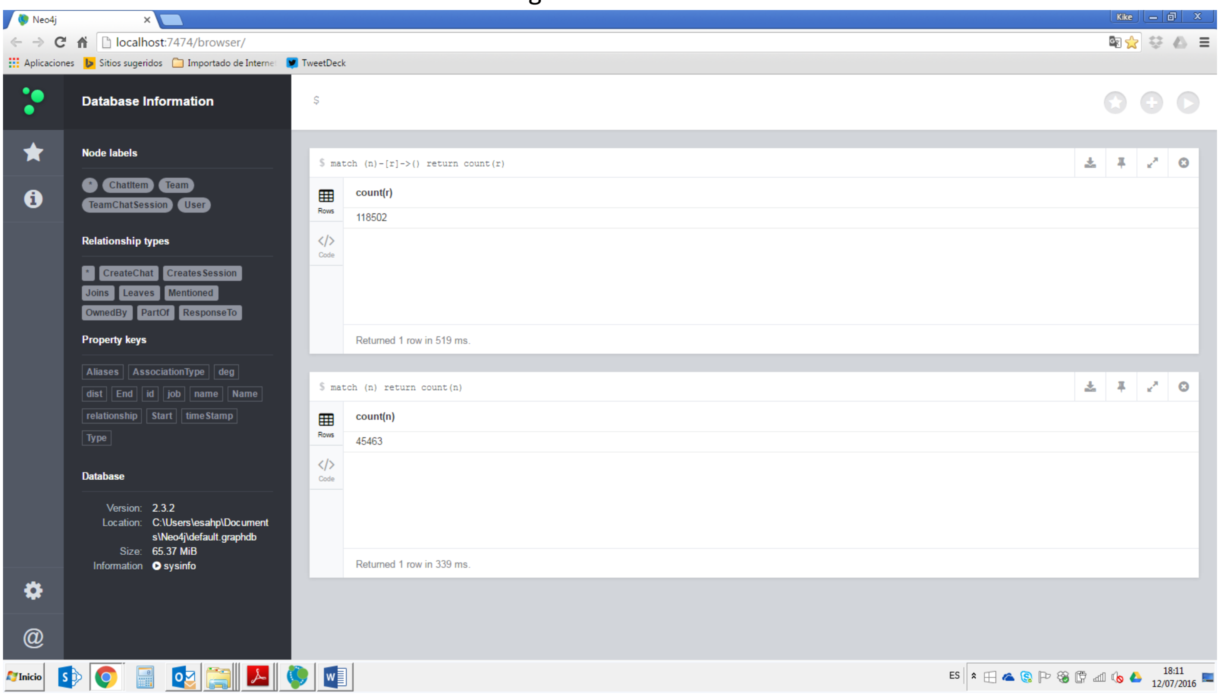
We have 4 types of nodes:

- User nodes, which represent the different players.
- Team nodes, which represent the different teams to which the players belong.
- ChatItem nodes, which represent the different chats that take place within the teams.
- TeamChatSession nodes, which represent the different sessions of the chats.

We have 8 types of edges (relationships) between the nodes:

- CreatesSession edges, which show the moment when a User created a TeamChatSession.
- OwnedBy edges, which show the moment when a TeamChatSession was launched within a team.
- Joins edges, which show the moment when a User joined a TeamChatSession.
- Leaves edges, which show the moment when a User left a TeamChatSession.
- PartOf edges, which show the moment when a ChatItem was started within a TeamChatSession.
- CreateChat edges, which show the moment when a User created a TeamChatSession.
- Mentioned edges, which show the moment when a ChatItem was mentioned by a User.
- ResponseTo edges, which show the moment when a ChatItem was sent in response to anocher ChatItem.

We have in total 45463 nodes and 118502 edges:



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 (ERD table: chat_create_team_chat)	A line is added to this file when a player creates a new chat with their team.	userid teamid timestamp TeamChatSessionid
chat_item_team_chat.csv (ERD table: chat_item_team_chat)	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.	userid TeamChatSessionid ChatItemid timestamp
chat_join_team_chat.csv (ERD table: chat_join_team_chat)	Creates an edge labeled "Joins" from User to TeamChatSession. The columns are the User id, TeamChatSession id and the timestamp of the Joins edge.	userid TeamChatSessionid timestamp
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.	userid TeamChatSessionid timestamp
chat_mention_team_chat.csv (ERD table: chat_mention_team_chat)	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.	ChatItemid userid timestamp
chat_respond_team_chat.csv (ERD table: chat_respond_team_chat)	A line is added to this file when a player responds to a chat post.	ChatItemid1 ChatItemid2

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

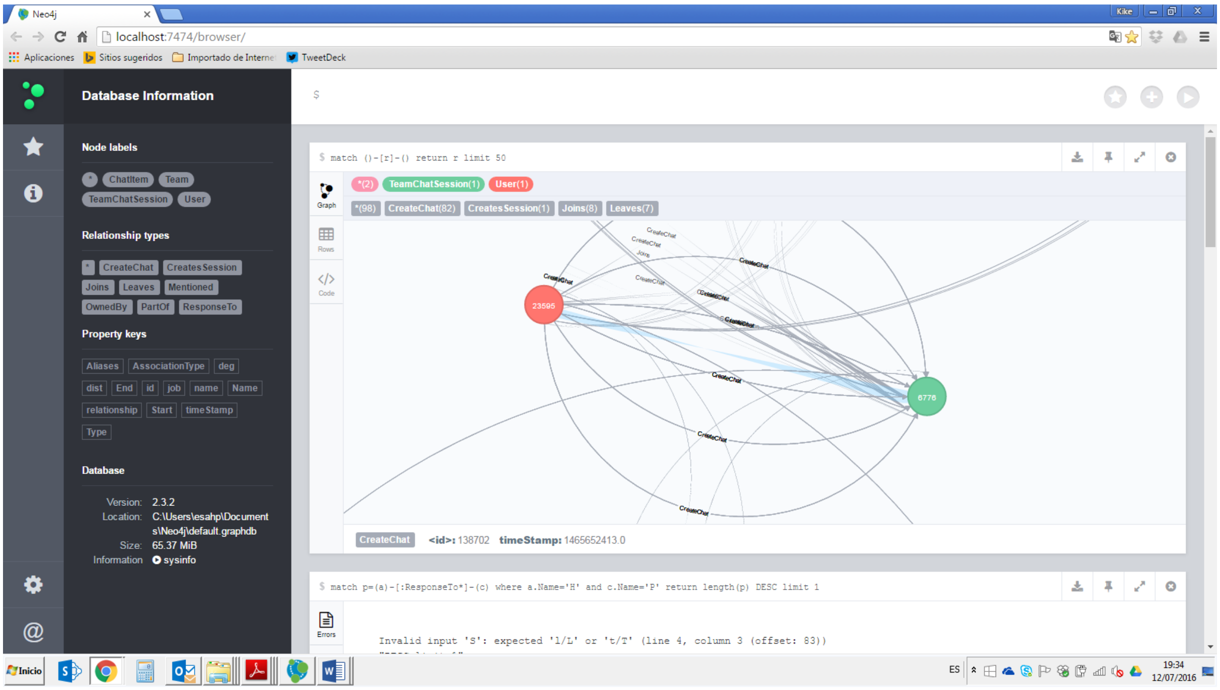
The first line gives the path of the file and reads it one row at a time. Then we create the different nodes by referring to the corresponding columns and converting them to integers. Finally, we create the edges by defining the origin and destiny nodes and labeling the relationship.

```
LOAD CSV FROM
"file:///D:/Formacion/BigData/_Coursera_BigDataSpecialization/06_CapstoneProject/_big_data_capsto
ne_datasets_and_scripts/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)

LOAD CSV FROM
"file:///D:/Formacion/BigData/_Coursera_BigDataSpecialization/06_CapstoneProject/_big_data_capsto
ne_datasets_and_scripts/chat_join_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (u)-[:Joins{timeStamp: row[2]}]->(c);

LOAD CSV FROM
"file:///D:/Formacion/BigData/_Coursera_BigDataSpecialization/06_CapstoneProject/_big_data_capsto
ne_datasets_and_scripts/chat_leave_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (u)-[:Leaves{timeStamp: row[2]}]->(c);

LOAD CSV FROM
"file:///D:/Formacion/BigData/_Coursera_BigDataSpecialization/06_CapstoneProject/_big_data_capsto
ne_datasets_and_scripts/chat_item_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (i:ChatItem {id: toInt(row[2])})
MERGE (i)-[:PartOf{timeStamp: row[3]}]->(c)
MERGE (u)-[:CreateChat{timeStamp: row[3]}]->(c);

LOAD CSV FROM
"file:///D:/Formacion/BigData/_Coursera_BigDataSpecialization/06_CapstoneProject/_big_data_capsto
ne_datasets_and_scripts/chat_mention_team_chat.csv" AS row
MERGE (i:ChatItem {id: toInt(row[0])})
MERGE (u:User {id: toInt(row[1])})
MERGE (i)-[:Mentioned{timeStamp: row[2]}]->(u);
```

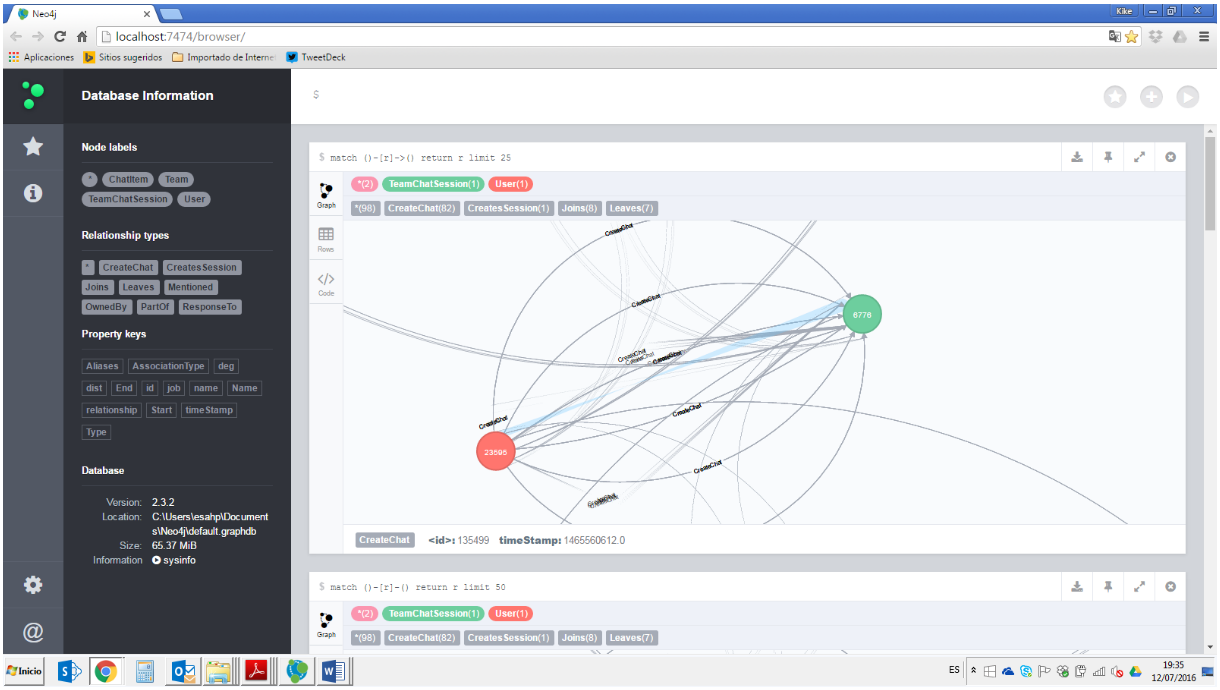
```
LOAD CSV FROM
"file:///D:/Formacion/BigData/_Coursera_BigDataSpecialization/06_CapstoneProject/_big_data_capstone_datasets_and_scripts/chat_respond_team_chat.csv" AS row
MERGE (i1:ChatItem {id: toInt(row[0])})
MERGE (i2:ChatItem {id: toInt(row[1])})
MERGE (i1)-[:ResponseTo{timeStamp: row[2]}]->(i2);
```

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.

50 relationships without direction



25 relationships without direction



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.

Looking at the “RespondTo” edge label among all nodes, I get the path lengths and sort out only the biggest result:

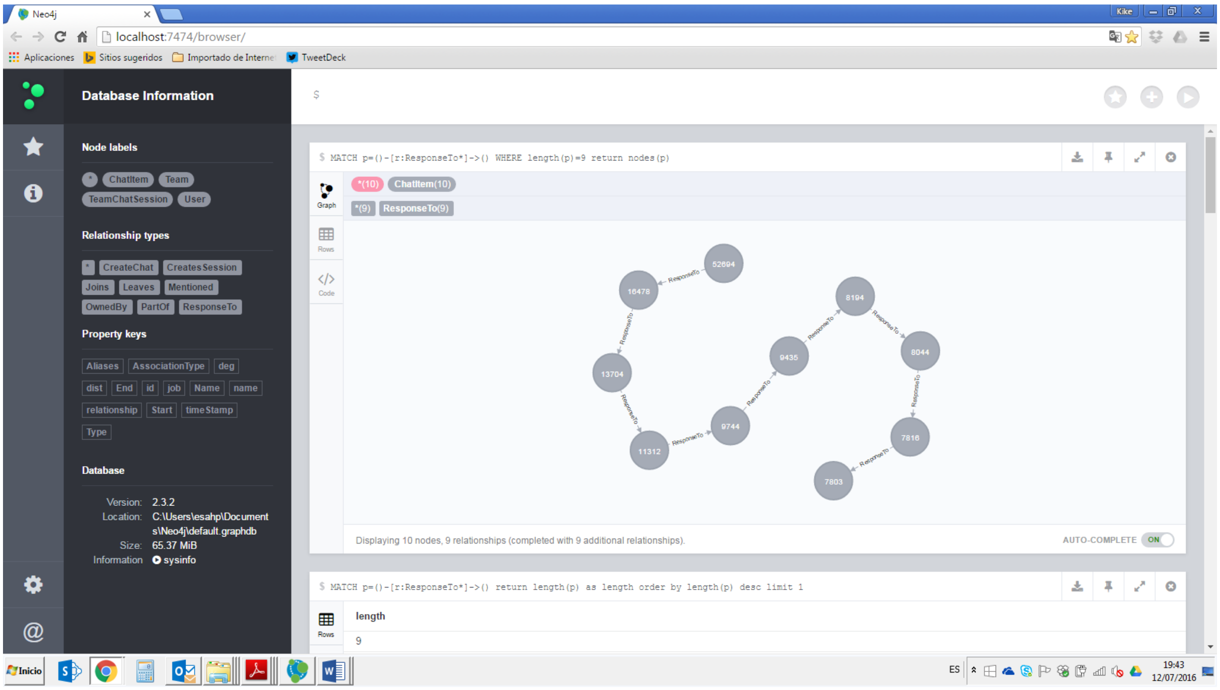
MATCH p=()-[r:Responds*]->()

return length(p) as length

order by length(p) desc

limit 1;

The longest path has 9 edges, and involves 10 ChatItems.



These 10 ChatItems corresponded to 5 unique users:

Userid	# Chat Items
1192	5
853	2
1978	1
1514	1
1153	1

This is useful for Eglence Inc so that they can identify the most active players.

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

MATCH (u)-[r:CreateChat]->()
RETURN u.id, count(r) AS connections
ORDER BY connections DESC LIMIT 10

Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams

MATCH ()-[r:PartOf]->(m)
with m
MATCH (m)-[r:OwnedBy]->(n)
RETURN n.id,count(r) AS counts
ORDER BY counts DESC LIMIT 10

Teams	Number of Chats
82	1324
185	1036
112	957

Finally, present your answer.

match p=(u:User)-[:Joins]-(c:TeamChatSession)-[:OwnedBy]-(t:Team) where t.id in [82,185,112,18,194,129,52,136,146,81] and u.id in [394,2067,209,1087,554,516,1627,999,668,461] return p

There is one chattest user id 999 who was part of one chattest team id 52

Question 3:

How Active are Groups of Users?

Most Active Users (based on Cluster Coefficients)

UserID	Coefficient
209	0.95
554	0.90
1078	0.80

In this question, we will compute an estimate of how “dense” the neighborhood of a node is. In the context of chat that translates to how mutually interactive a certain group of users are. If we can identify these highly interactive neighborhoods, we can potentially target some members of the neighborhood for direct advertising. We will do this in a series of steps.

(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

The way to make this connection will be to create a new edge called, for example, "InteractsWith" between users to satisfy either of the two conditions. So we will write a query to create these edges for each condition. For the first condition, this query would have the following structure:

Match (u1:User)-[:CreateChat]->(i:ChatItem)-[:Mentioned]->(u2:User) create (u1)-[:InteractsWith]->(u2)

Use the same logic to create the query statement for the second condition. This query will also have the form

Match (u1:User)-[:CreateChat]->(i1:ChatItem)-[:ResponseTo]-(i2:ChatItem)<-[:CreateChat]-(u2:User) create (u1)-[:InteractsWith]->(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 "Interacts with". This will take the form:

Match (u1)-[r:InteractsWith]->(u1) delete r

(c) Given this new edge type, we will have to create a scoring mechanism to find users having dense neighborhoods. The score should range from 0 (a disconnected user) to 1 (a user in a clique – where every node is connected to every other node). One such scoring scheme is called a "clustering coefficient" defined as follows. If the number of neighbors of node is 5, then the clustering coefficient of the node is the ratio between the number of edges amongst these 5 neighbors(not counting the given node) and 5*4 (all the pairwise edges that could possibly exist). Thus the denominator is k * (k-1) if the number of neighbors of the node is k.

Your task in this question is to find the clustering coefficients of the chattiest users (you know their ids) from Q2.

(d) To do this computation, we need to

· get the list of neighbors and

· the number of neighbors of a node based on the "InteractsWith" edge.

For each of these neighbors, we need to find

· The number of edges it has with the other members on the same list.

· 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

We then need to add the edges we get for each member and divide this by k*(K-1)