**Technical Appendix**

**Catch the Pink Flamingo Analysis**

**Produced by: Satyanarayana Ponnuru**

Acquiring, Exploring and Preparing the Data

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 | 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.  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 | This file contains a line for each team last 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. | 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 | 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 | 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. | 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 |
|  |  |  |

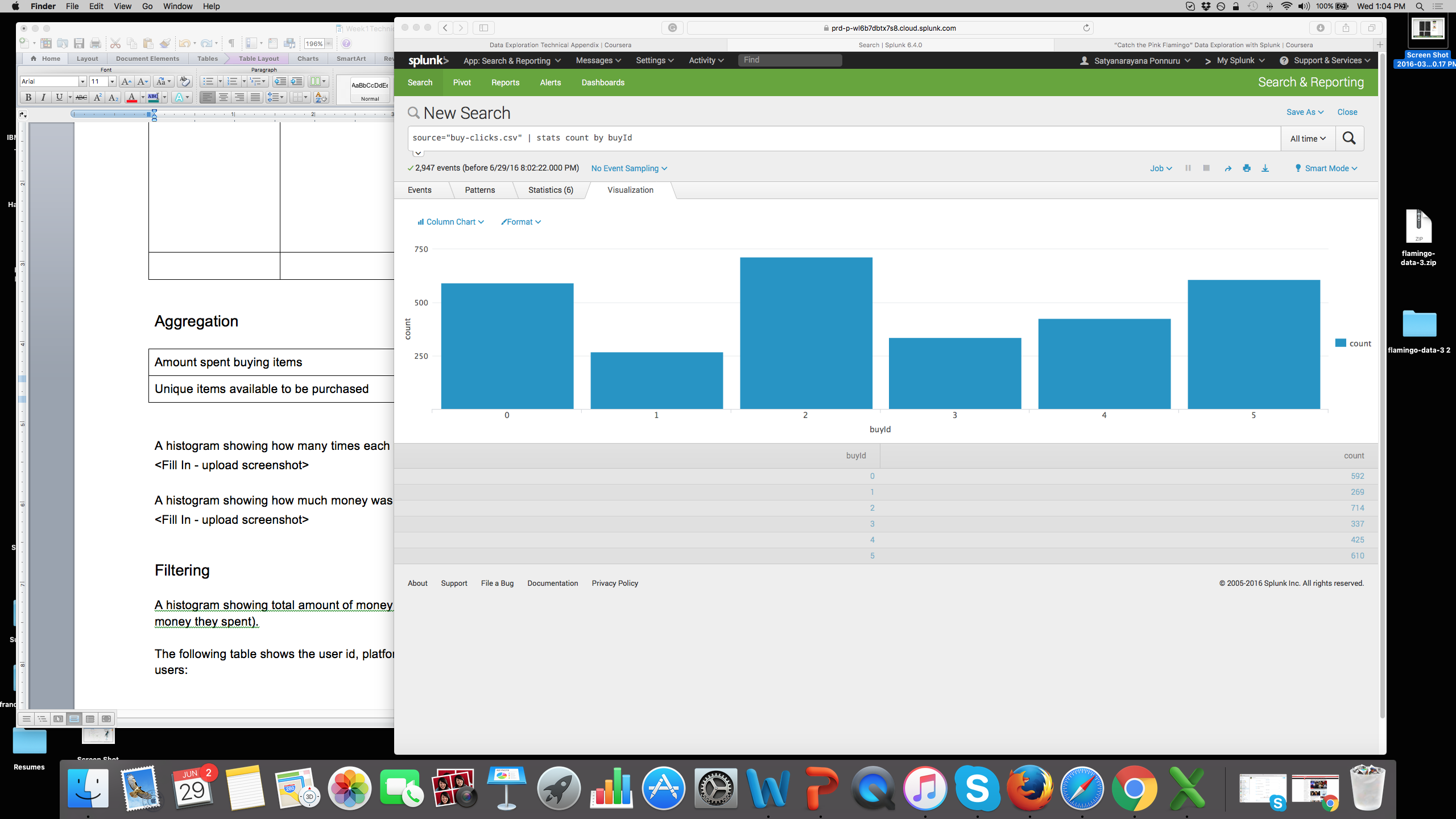
Aggregation

|  |  |
| --- | --- |
| Amount spent buying items | 21407.0 |
| Unique items available to be purchased | 6 |

source="buy-clicks.csv" | stats sum(price)

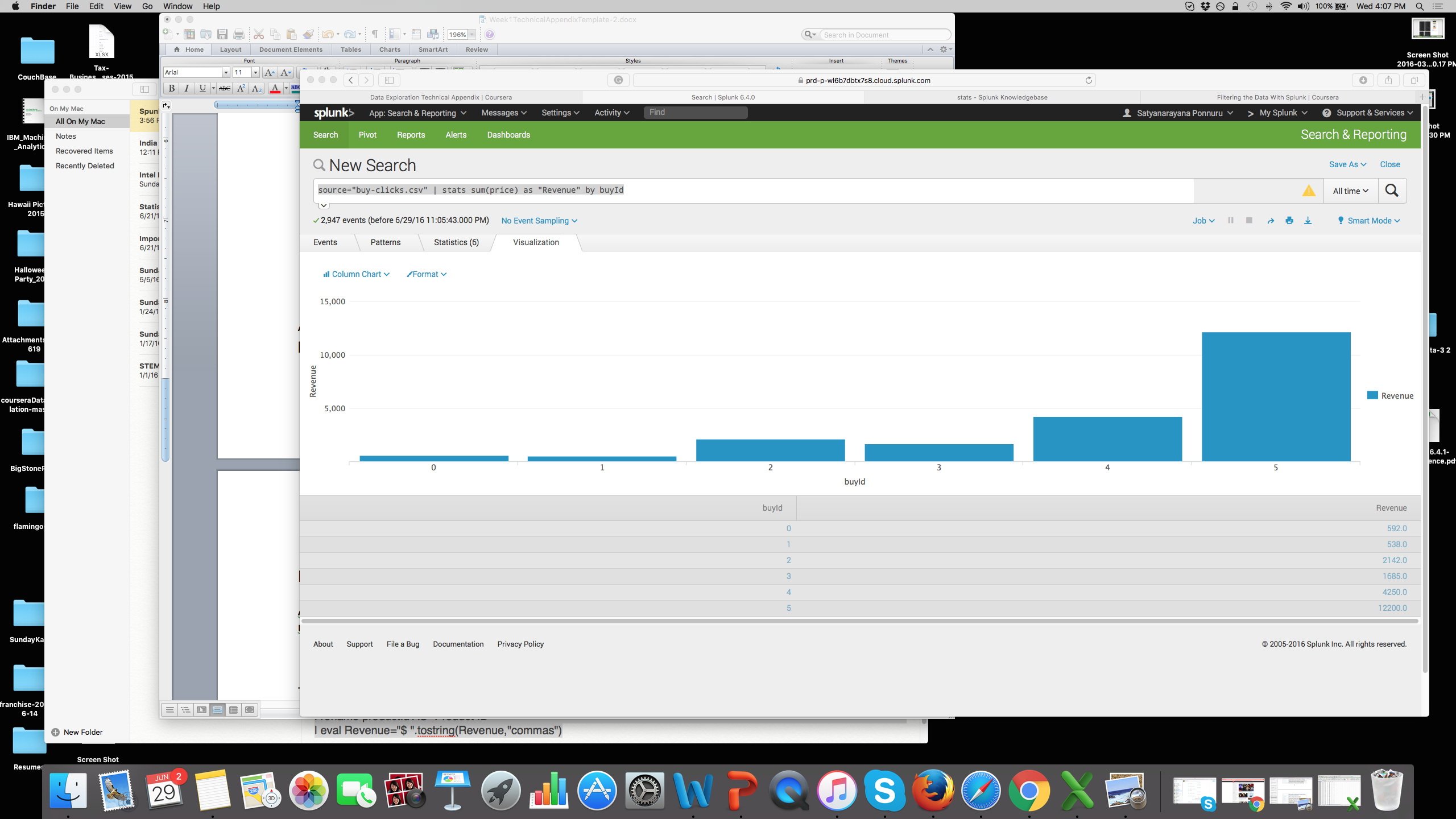
A histogram showing how many times each item is purchased:

[ Splunk Query - source="buy-clicks.csv" | stats count by buyId ]



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

[ Splunk Query - source="buy-clicks.csv" | stats sum(price) as "Revenue" by buyId ]

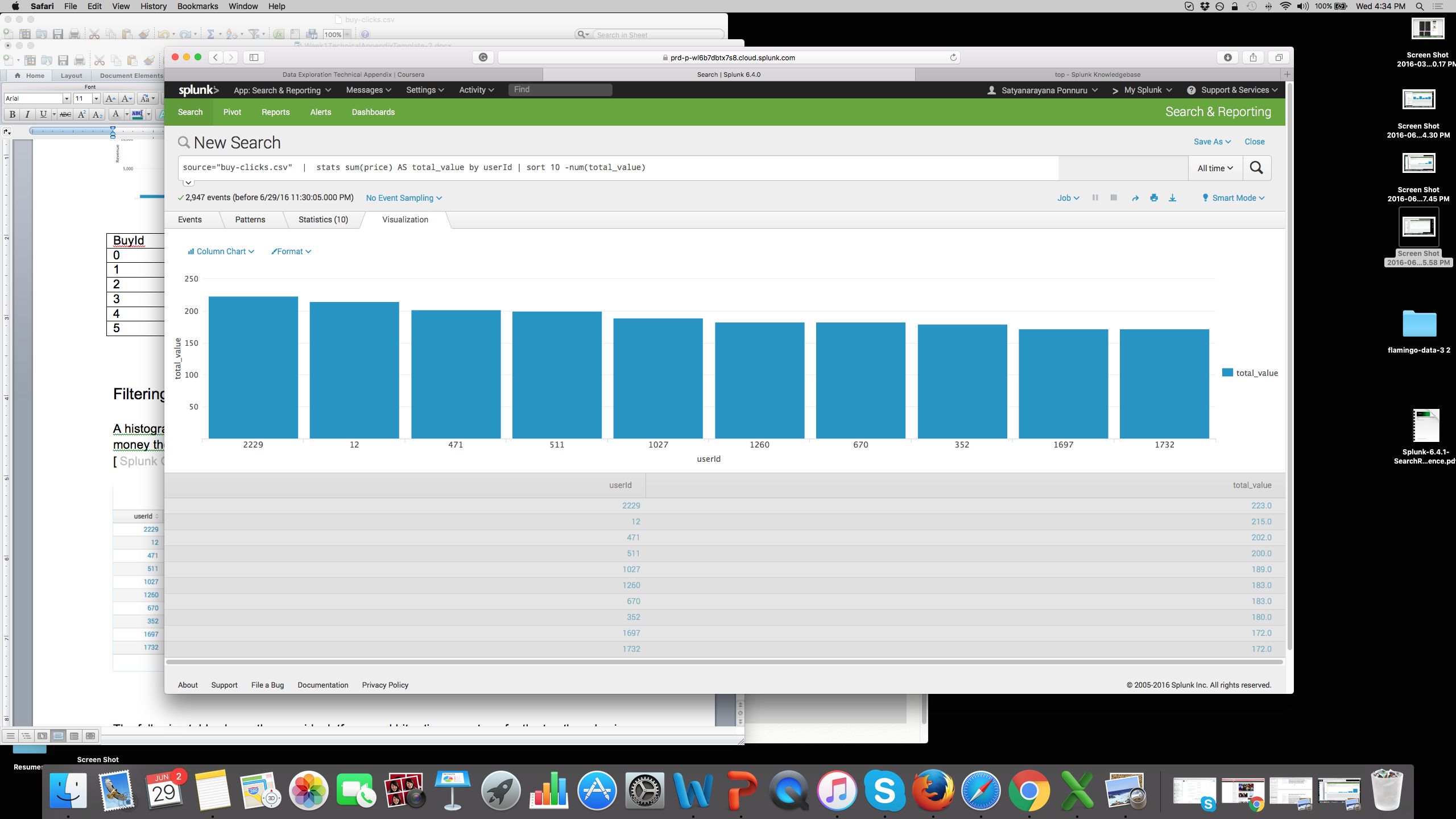


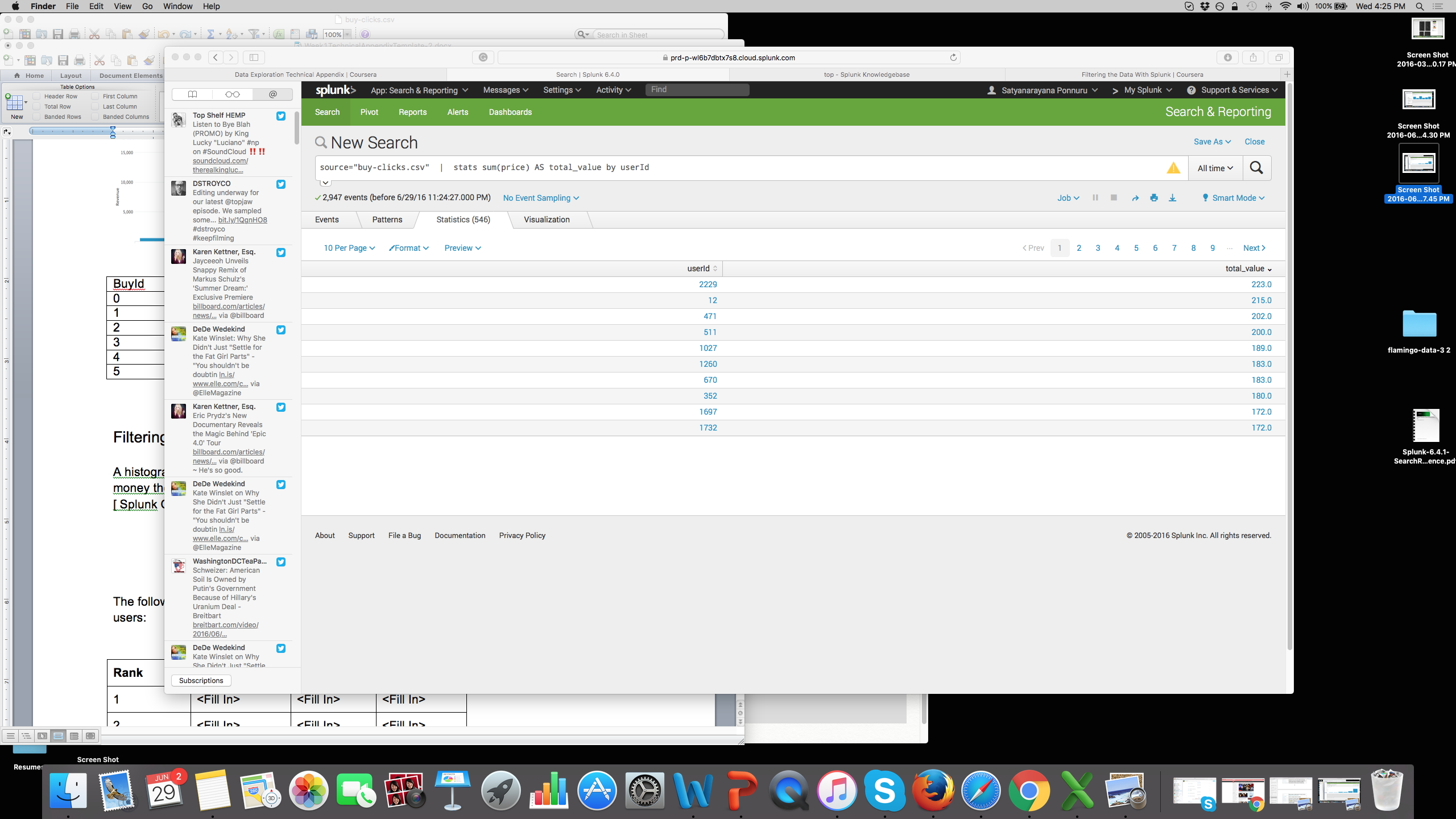
|  |  |
| --- | --- |
| BuyId | Revenue |
| 0 | 592 |
| 1 | 538 |
| 2 | 2142 |
| 3 | 1685 |
| 4 | 4250 |
| 5 | 12200 |

Filtering

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

[ Splunk Query - source="buy-clicks.csv" | stats sum(price) AS total\_value by userId | sort 10 –num(total\_value) ]





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

[ Splunk: source="game-clicks.csv" | stats count(isHit) as totalClicks, count(eval(isHit==1)) as hitClicks by userId | eval hitRatio = hitClicks/totalClicks | table userId, hitRatio |sort 3 -num(hitRatio) ]

|  |  |  |  |
| --- | --- | --- | --- |
| **Rank** | **User Id** | **Platform** | **Hit-Ratio (%)** |
| 1 | 2229 | Iphone | 0.1159 |
| 2 | 12 | Iphone | 0.1306 |
| 3 | 471 | Iphone | 0.1450 |

Data Classification Analysis

**Data Preparation**

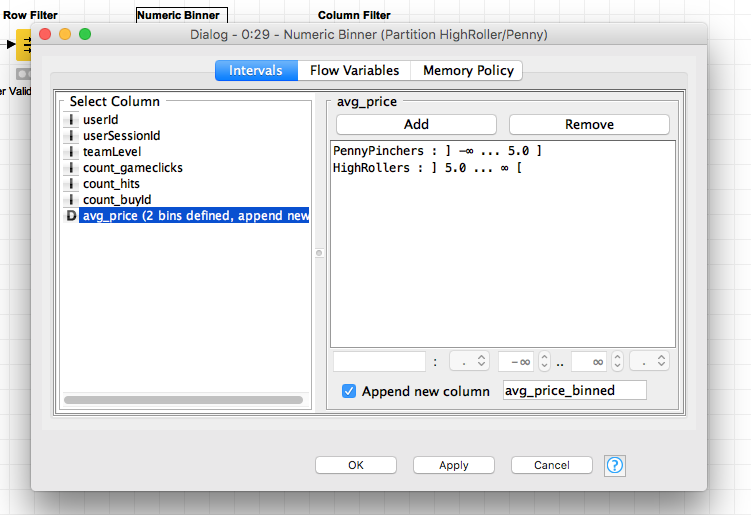
Analysis of combined\_data.csv

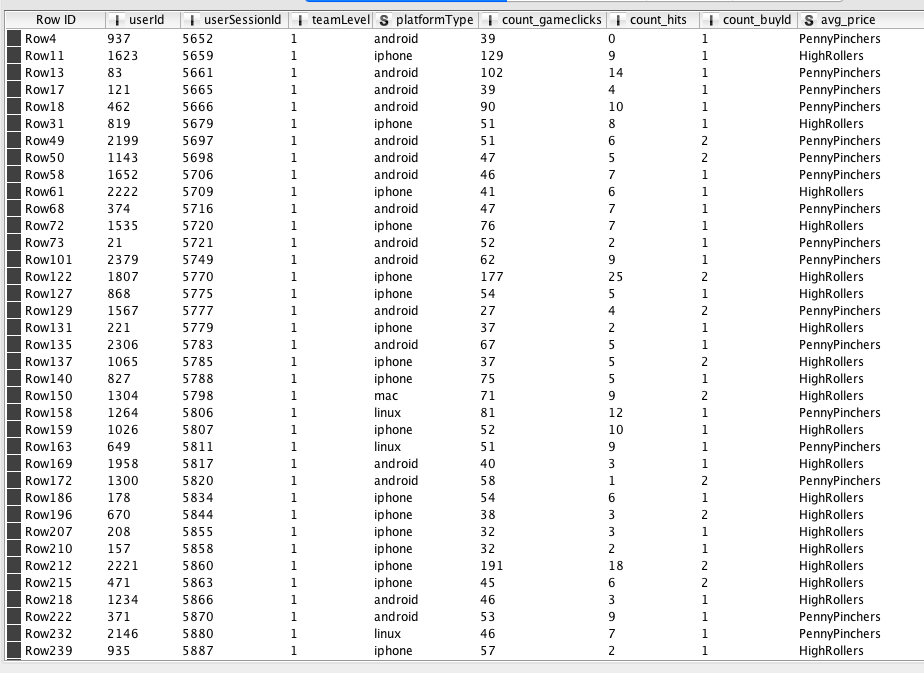
Sample Selection

|  |  |
| --- | --- |
| **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:





For predication, we need to have class variable to be in categorical values. We convert to categorical values. At the present the class variable is not in categorical value. So we convert to categorical using Numerical Binner node. The Numerical Binner node will take numeric value, and compare if the numeric is less or equal to 5, it will put PennyPinchers category. Otherwise it will change the numeric value to HighRoller category.

Creating the new categorical attribute was necessary because, it will help to predicate the right user who will be bringing revenue or a game enthusiastic/promoter.

Attribute Selection

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

|  |  |
| --- | --- |
| **Attribute** | **Rationale for Filtering** |
| UserId | To know gamer will spend, there is no valuable information in UserId attribute. |
| UserSessionId | Similarly no valuable information is there in the UserSessionId attribute. |
| Avg\_price | Created new class variable which has this information in categorical values |
|  |  |

**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.

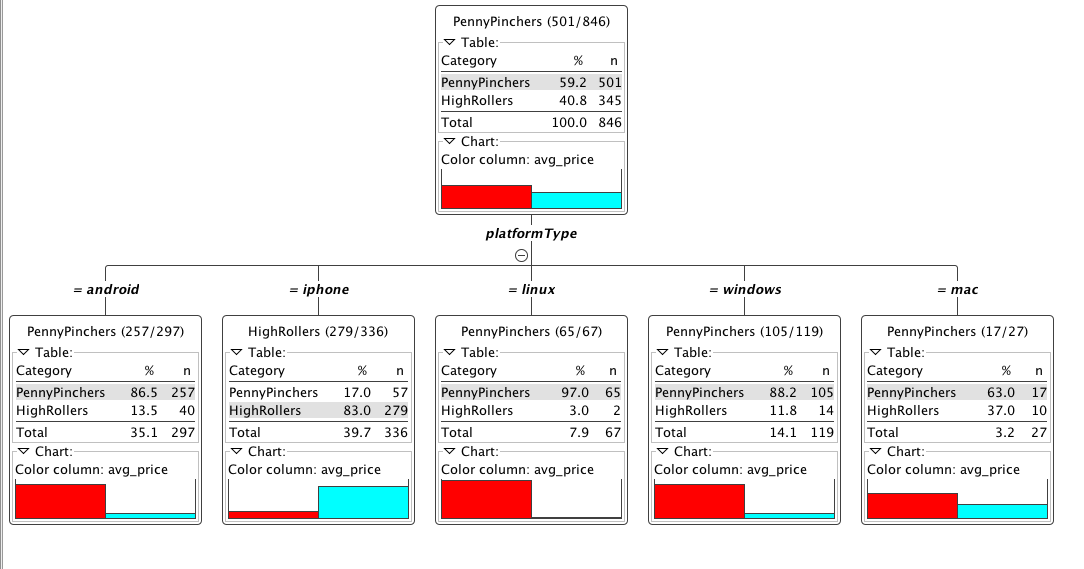
This is important because…

1. To test the accuracy of model.
2. It will help to pick the right attributes to build right model.

When partitioning the data using sampling, it is important to set the random seed because

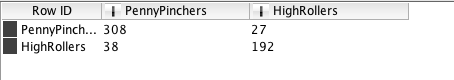
1. When building the model, we are keeping constant random seed for entropy calculations for different attribute selections. So, we can determine the accuracy and confusion matrix comparison.

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

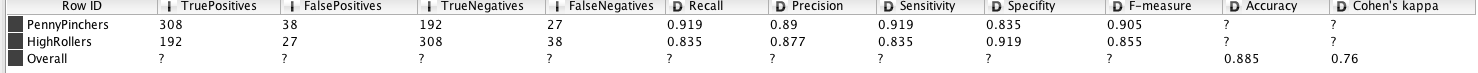


**Evaluation**

A screenshot of the confusion matrix can be seen below:



As seen in the screenshot above, the overall accuracy of the model:

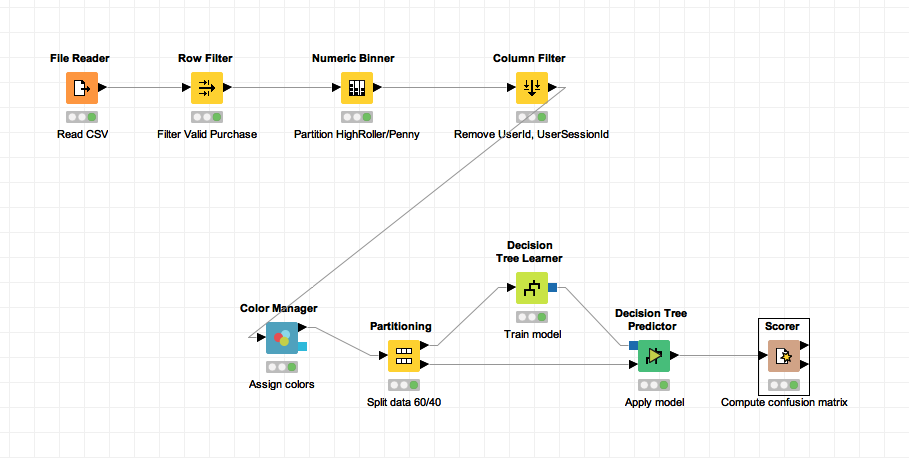


<Fill In: Write one sentence for each of the values of the confusion matrix indicating what has been correctly or incorrectly predicted.>

1. 308 times out of 335 PennyPincher was predicted correctly.
2. 27 times out of 335 PennyPincher was not predicted correctly.
3. 192 times out of 230, HighRollers was predicted correctly.
4. 38 times out of 230, HighRollers was not predicted correctly.

**Analysis Conclusions**

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

HighRollers will bring more than $5 revenue per user uses “iphone” platform where as PennyPincher revenue in his game life will less $5 dollars uses “android, linux, mac, window’s etc” platform.

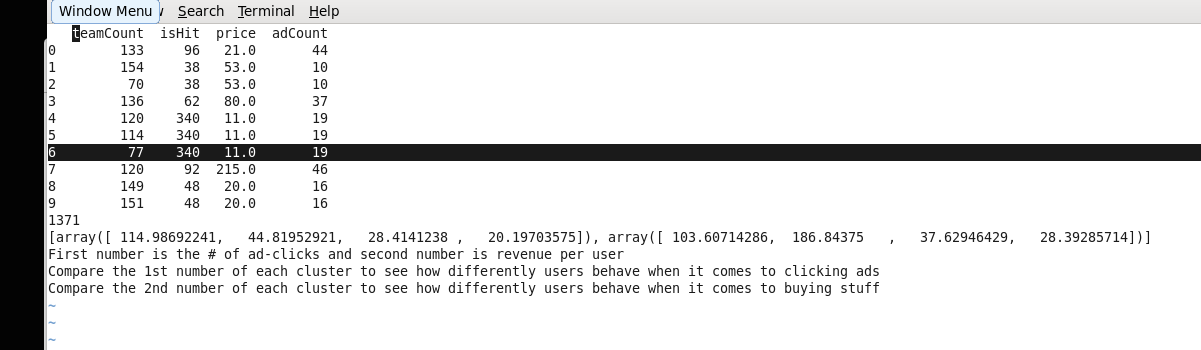
|  |
| --- |
| **Specific Recommendations to Increase Revenue** |
| 1. Make sure iphone platform works nicely without bugs, because this platform brings high revenue. |
| 2. In other platform (android, linux, mac, windows etc), promote the users by giving promotions to bring in new iphone users ☺. |

Clustering Analysis

**Attribute Selection**

|  |  |
| --- | --- |
| **Attribute** | **Rationale for Selection** |
| adcount | This attribute tells how many times player clicked on the ads. And in the analysis it can give insights how many times player will click on the ads in his game lifetime. |
| price | Attribute can tell us how much revenue player will generate in his game lifetime. |
| isHit | Attribute will give insight in his score and relationship with other attributes like adcount, price and team size. |
| team | Attribute will give insight how big his team is and any relationship in how his isHit score, how much he buys and how many ad’s he will click. |

**Training Data Set Creation**

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


Dimensions of the training data set (rows x columns) : 1371 X 4

# of clusters created: 2

**Cluster Centers**

|  |  |
| --- | --- |
| **Cluster #** | **Cluster Center** |
| 1 | 114.98, 44.82, 28.414, 20.197 (teamcount, isHit, price, adcount) |
| 2 | 103.61,186.84, 37.63, 28.39 (teamcount, isHit, price, adcount) |
|  |  |

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that…

The bigger the teamcount, the game clicks(isHit) is less per individual, the price revenue and ad clicks are less.

Cluster 2 is different from the others in that …

If the team count is less, the game clicks(isHit) is high and similarly price revenue and ad clicks are more.

**Recommended Actions**

|  |  |
| --- | --- |
| **Action Recommended** | **Rationale for the action** |
| Maintain ideal team size ~ 100 | From analysis, ideal team is around 100, the correct game clicks are high, and the revenue from purchase are high and ad clicks are high. |
| More than 115 team size, give recommendations to reduce the size, so the score will advance fast. | From analysis, if the team is bigger (more than 118 team), score, adcount and revenue are reducing.  From players point, we can increase the score, From company point, we can increase revenue and adcount if we recommend to reduce the teamsize. |
|  |  |
|  |  |

**Graph Analytics Analysis**

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

Graph model loaded from the script commands and CSV file.

Graph model consists of the following Nodes: User – each unique user, Team – each unique team, TeamChatSession – Unique Team Chat Session, ChatItem – Unique Chat Item added by a User.

Again in the graph model, these Nodes have relationships.

* (User)-[CreatesSession]->(TeamChatSession)-[OwnedBy]->(Team)
* (User)-[Joins]->(TeamChatSession)
* (User)-[Leaves]->(TeamChatSession)
* (User)-[CreateChat]->(ChatItem)-[PartOf]->(TeamChatSession)
* (ChatItem)-[Mentioned]->(User)
* (ChatItem)-[ResponseTo]->(ChatItem)

**Creation of the Graph Database for Chats**

1. Schema of the 6 CSV files

**File:** 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.

**Schema:** UserId, TeamId, TeamChatSessionId, TimeStamp

**File:** chat\_item\_team\_chat.csv

**ERD table:** chat\_item\_team\_chat

Creates nodes labeled ChatItem. Column 0 is User id, column 1 is the TeamChatSession id, column 2 is the ChatItem (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.

**Schema:** UserID, TeamChatSessionId, ChatItemID, TimeStamp

**File:** 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.

**Schema:** UserID, TeamChatSessionId, TimeStamp

**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.

**Schema:** UserId, TeamChatSessionId, TimeStamp

**File:** 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.

**Schema:** ChatItemId, UserId, TimeStamp

**File:** 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.

**Schema:** ChatItemId, ChatItemId, TimeStamp

1. Loading process and include a sample LOAD command

In the Neo4J system first execute the following instructions:

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 the CSV data files into Neo4J.

The following script loads the first file:

LOAD CSV FROM "file:/path/to/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)

The first line gives the path of the file. Use the directory path on your machine to replace the "path/to/" string in the LOAD CSV line. This command reads the chat\_create\_team\_chat.csv file one row at a time and creates User nodes. The 0th column value is converted to an integer and is used to populate the id attribute. Similarly the other nodes are created.

Line 4, MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c) creates an edge labeled "CreatesSession" between the user node u and the TeamChatSession node c. This edge has a property called timeStamp. This property is filled by the content of column 3 of the same row.

Similarly, the last line creates an edge labeled "OwnedBy" between the TeamChatSession node and the Team node.

**Logic to load the other files based on the following information:**

(i) chat\_join\_team\_chat.csv file creates an edge labeled "Joins" from User to TeamChatSession. The columns are the User id, TeamChatSession id and the timestamp of the Joins edge. For this graph, you would use a similar MERGE statement for User id and TeamChatSession as previously done (take care that column values are slightly different), and you would define your edges labelled "Join" with a similar MERGE statement using the timeStamp as the "OwnedBy" edges were defined previously.

(ii) chat\_leave\_team\_chat.csv file creates an edge labeled "Leaves" from User to TeamChatSession. The columns are the User id, TeamChatSession id and the timestamp of the Leaves edge. For this graph, you would use a similar MERGE statement for User id and TeamChatSession as previously done, and you would define your edges labelled "Leaves" with a similar MERGE statement using the timeStamp as the "Join" edges were defined previously.

(iii) chat\_item\_team\_chat.csv file 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. For this graph, you would use a similar MERGE statement for User id and TeamChatSession as previously done. You would define a third node type, "ChatItem" in a similar way as the Team node in (i) above, and you would define two sets of edges, one labelled "CreateChat" and the other labelled "PartOf" with similar MERGE statements using the timeStamp as the those which were defined previously in (i) above.

(iv) chat\_mention\_team\_chat.csv file 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. For this graph, you would use a similar MERGE statement for User and ChatItem as previously done (take care that column values are slightly different), and you would define your edge labelled "Mentioned" with a similar MERGE statement using the timeStamp as was defined previously.

(v) chat\_respond\_team\_chat.csv file creates an edge labeled "ResponseTo" from a ChatItem node to another ChatItem node. Column 0 has the ID of the first ChatItem node, column 1 has the ID of the second ChatItem node and column 2 has the timeStamp of the edge. For this graph, you would use two MERGE statements for each of the two ChatItem columns as previously done, and you would define your edge labelled "ResponseTo" with a similar MERGE statement using the timeStamp as was defined previously.

chat\_join\_team\_chat.csv (User, TeamChatSession, timeStamp)

LOAD CSV FROM "file:/chat\_join\_team\_chat.csv" AS row

MERGE (u:User {id: toInt(row[0])})

MERGE (c:TeamChatSession {id: toInt(row[1])})

MERGE (u)-[:Join{timeStamp: row[2]}]->(c)

chat\_leave\_team\_chat.csv

LOAD CSV FROM "file:/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)

chat\_item\_team\_chat.csv (User, TeamChatSession, ChatItem, timestamp)

LOAD CSV FROM "file:/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 (u)-[:CreateChat{timeStamp: row[3]}]->(i)

MERGE (i)-[:PartOf{timeStamp: row[3]}]->(c)

chat\_mention\_team\_chat.csv (ChatItem, User, timestamp)

LOAD CSV FROM "file:/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)

chat\_respond\_team\_chat.csv (ChatItem, ChatItem, timestamp)

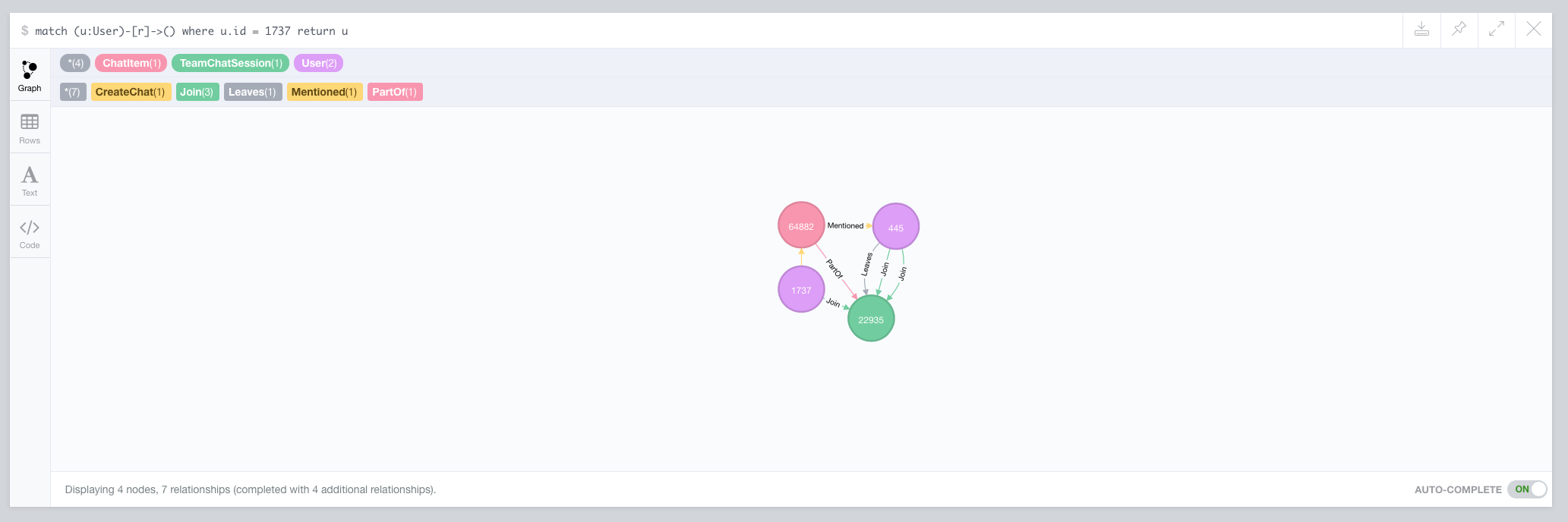
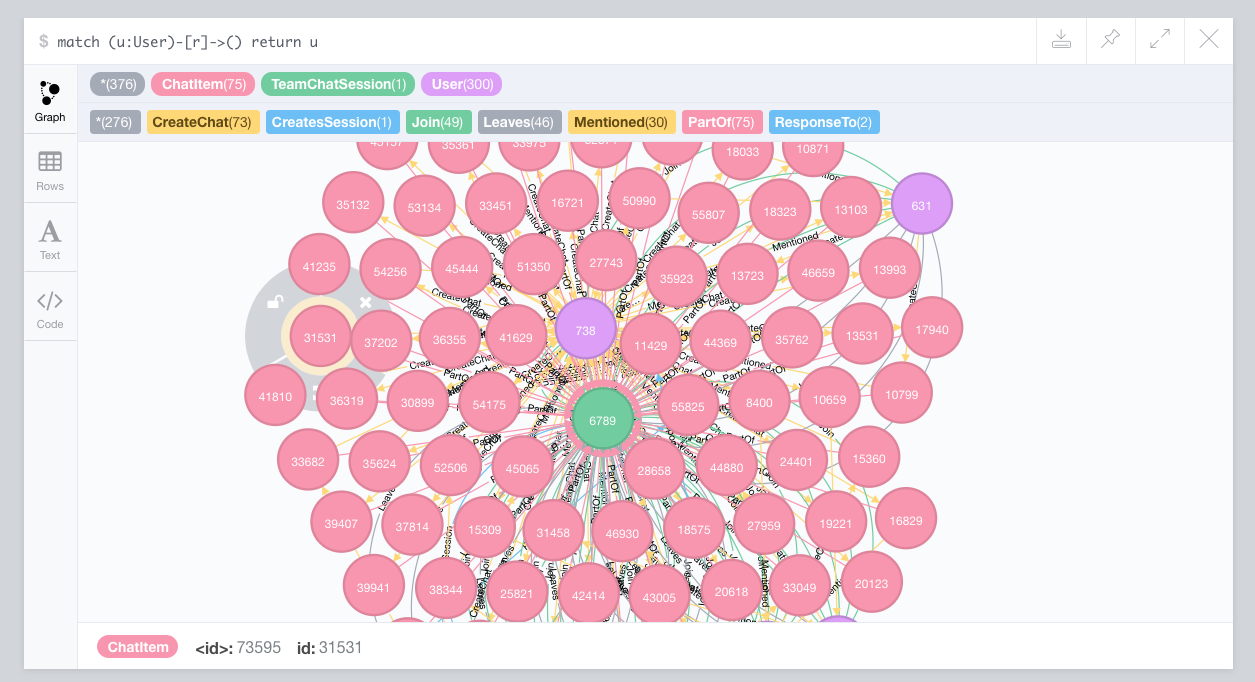
LOAD CSV FROM "file:/chat\_respond\_team\_chat.csv" AS row

MERGE (i:ChatItem {id: toInt(row[0])})

MERGE (y:ChatItem {id: toInt(row[1])})

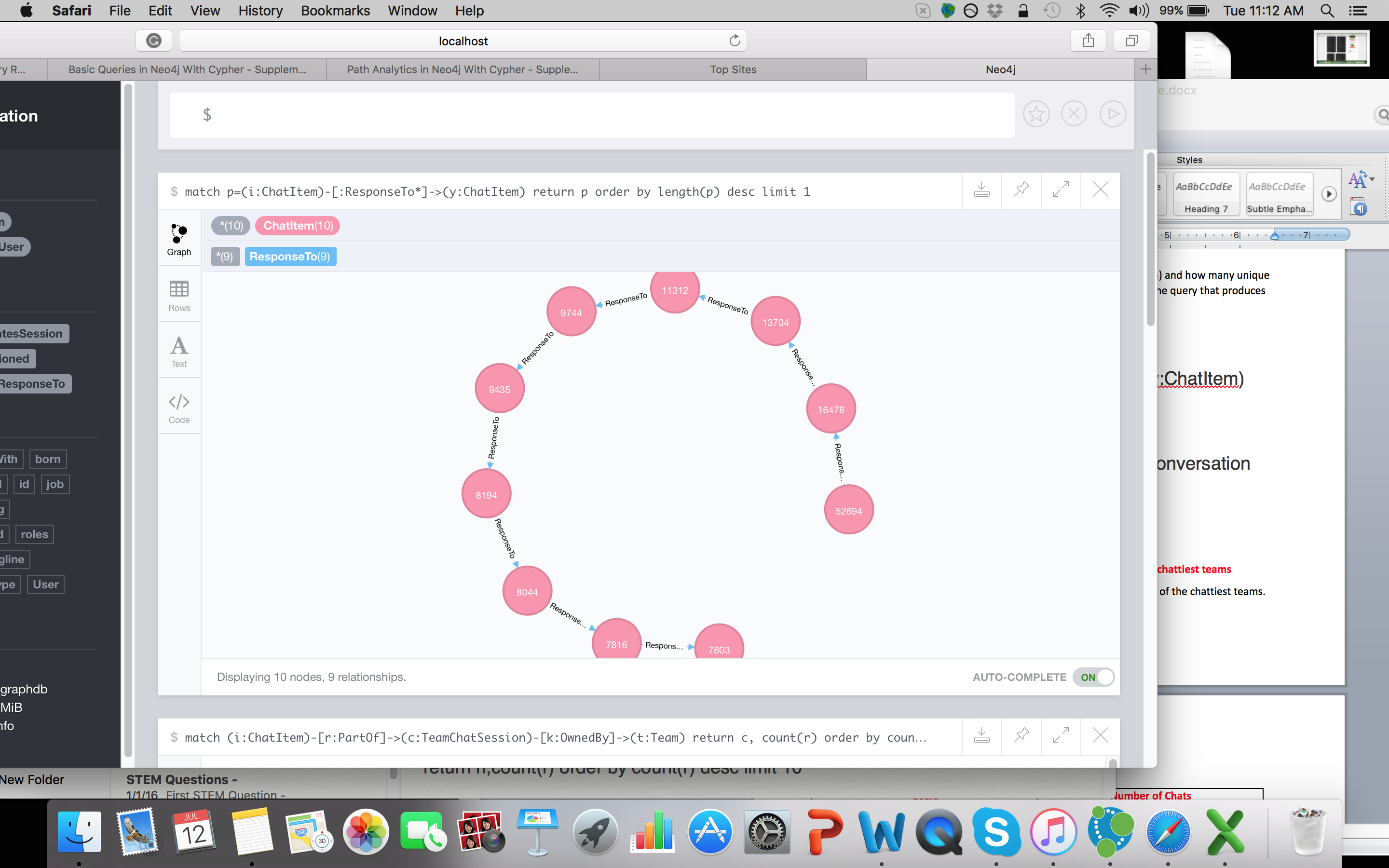
MERGE (i)-[:ResponseTo{timeStamp: row[2]}]->(y)

1. Screenshot of some part of the graph.



**Finding the longest conversation chain and its participants**

10



Query for Longest conversation Chain:

match p=(i:ChatItem)-[:ResponseTo\*]->(y:ChatItem)

return p order by length(p) desc limit 1

Steps –

Get all the nodes that are of chatItem – ResponseTo -> chatItem.

Measure the length of the relationships and put them in the descending order with highest value in the top.

Query for unique users were part of the conversation chain:

5 Unique users

Query –

match p=(i:ChatItem)-[:ResponseTo\*]->(y:ChatItem) where length(p)=9 with p match (u:User)-[r:CreateChat\*]->(i:ChatItem) where i.id in EXTRACT(n IN NODES(p)| n.id) return distinct u as u

Users - 1192, 1514, 853, 1153, 1978

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

**Chattiest Users**

|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 1087 | 109 |

// 10 chattiest Users Query

match (n:User)-[r:CreateChat]->()

return n,count(r) order by count(r) desc limit 10

**Chattiest Teams**

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

// 10 Chattiest Teams Query.

match (i:ChatItem)-[r:PartOf]->(c:TeamChatSession)-[k:OwnedBy]->(t:Team) return t, count(r) order by count(r) desc limit 10

Chattiest User in the Chattiest Team:

match p=(u:User)-[:Join]-(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 t, u

Chattiest Team Id: 52, and Chattiest User: 999

With the help of 10 chattiest Users, and 10 Chattiest Teams, we can find the chattiest user in the chattiest team. And these Users are talking to the whole team. They can spread the message quickly. So, Elegance can promote special messages to these users, so in turn these users can promote to the whole team.

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

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 209 | 0.9642857142857143 |
| 554 | 0.8571428571428571 |
| 1087 | 0.8333333333333333 |

**Recommended Actions**

Promote to bring in new iPhone players and make sure iPhone platform has good customer support.

Rationale –

Using data exploration, we see iphone users are the highest revenue generators. Unfortunately, their hit-ratio is 10%. Should provide some kind of support, so their scores go up and they stay long time span. In turn they will spend more on in-store purchase and advertisements.

And using classification analysis, we found one important attribute that leads to spending more revenue. That attribute is “platform” and the players using iPhone platform spends more money!!

So Eglence should bring in new iPhone players and make sure existing iPhone player’s has great support as highest priority.