Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: Luisa López Vázquez

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. 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 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

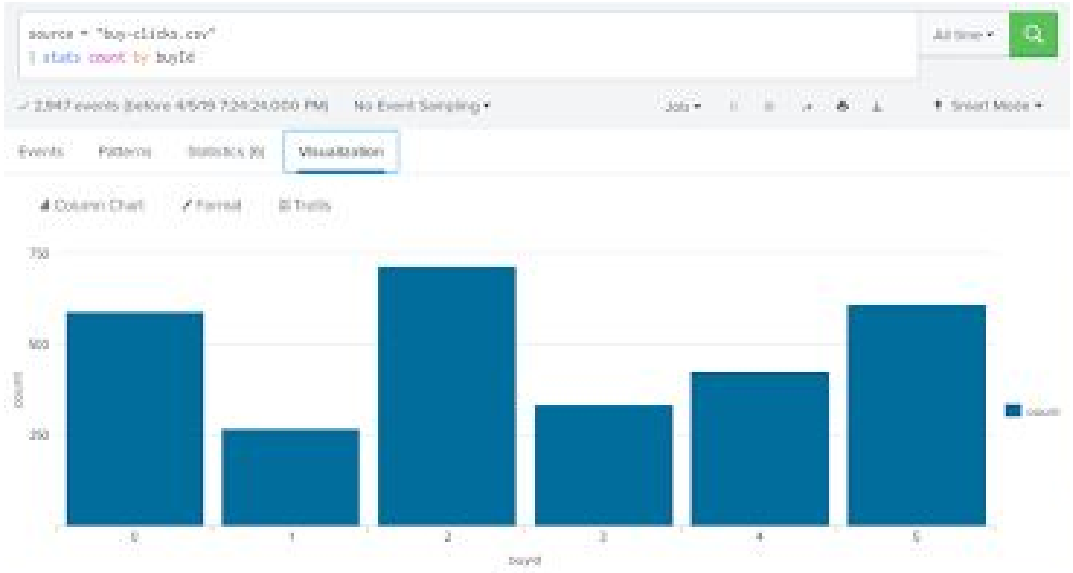
team-assignments.csv	A line is added to this file each time a user joins a team. A user can be in at most a single team at a time.timestamp: when the user joined the team. team: the id of the team userId: the id of the user assignmentId: a unique id for this assignment	timestamp: when the user joined the team. team: the id of the team userId: the id of the user assignmentId: a unique id for this assignment
LevelEvent.csv	A line is added to this file each time a team starts or finishes a level in the game	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: 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.	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

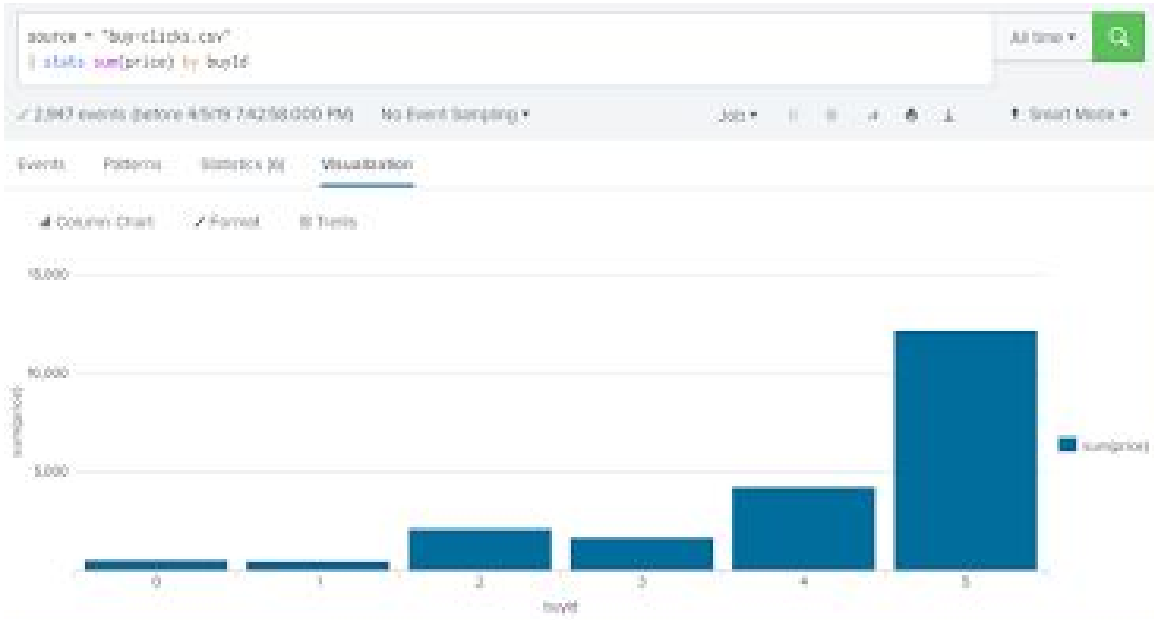
Amount spent buying items	21407.0
Number of unique items available to be purchased	buyId count 0 592 1 269 2 714 3 337 4 425 5 610

A histogram showing how many times each item is purchased:

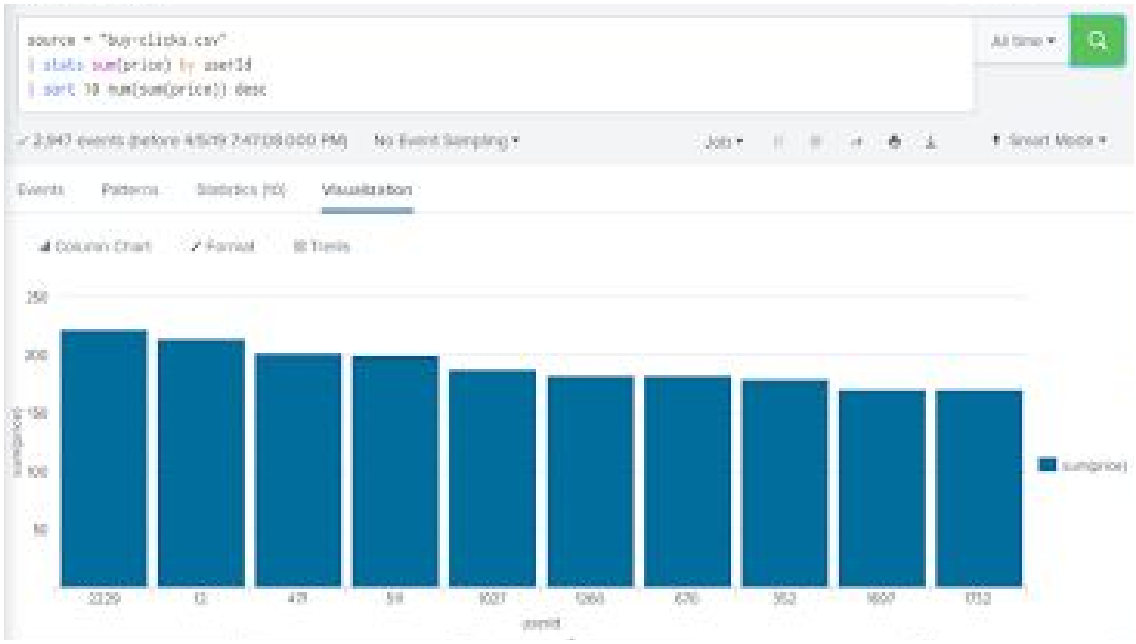


A histogram showing how much money was made from each item:

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	0.11596958174904944
2	12	iphone	0.13068181818181818
3	471	iphone	0.1450381679389313

Data Classification Analysis

Data Preparation

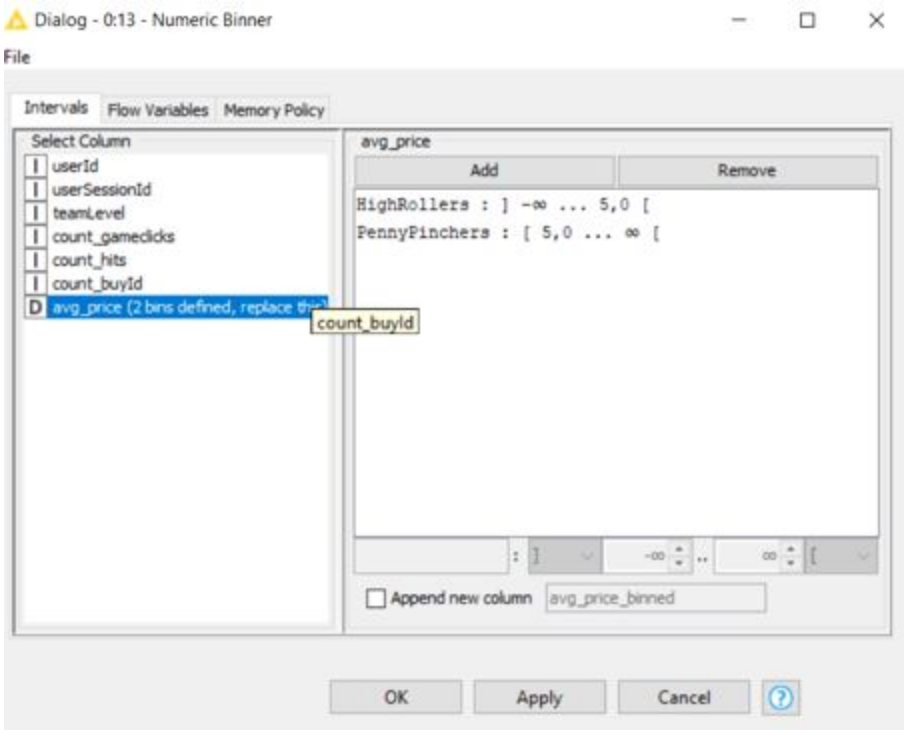
Analysis of combined_data.csv

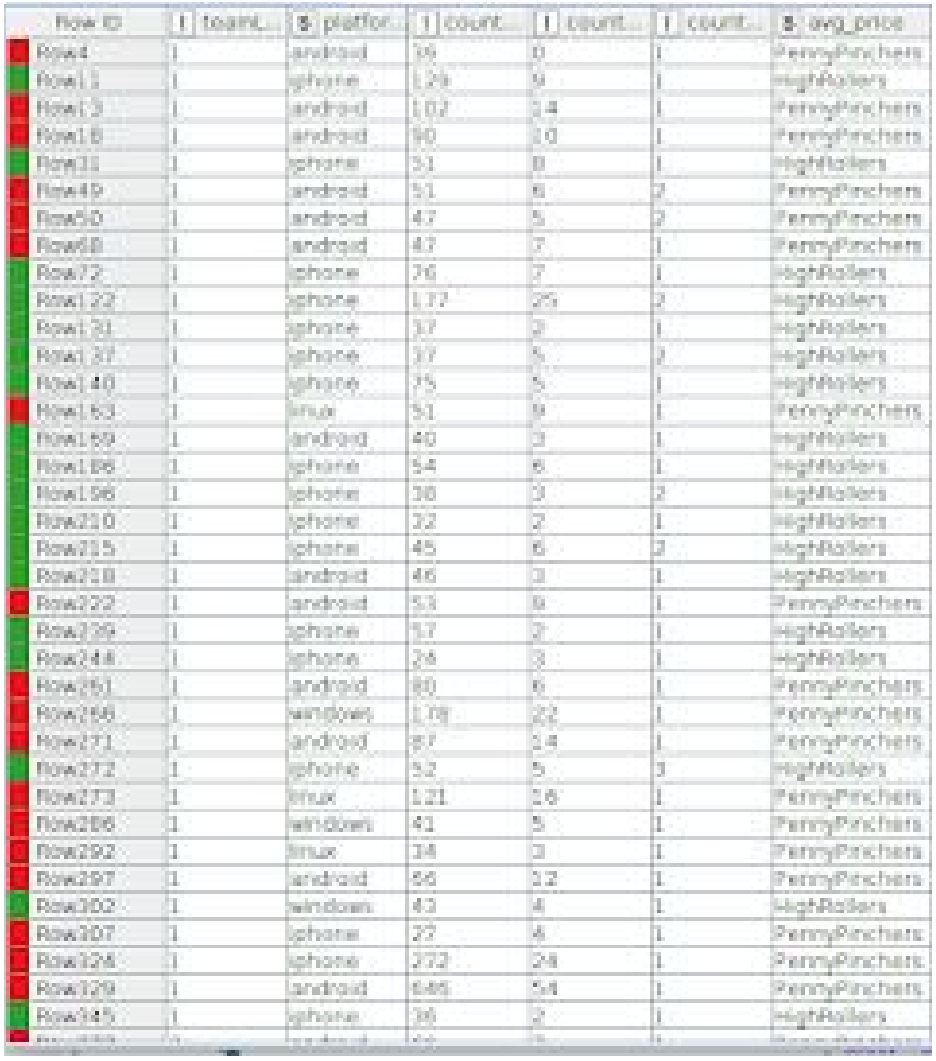
Sample Selection

Item	Amount
# of Samples	4619
# of Samples with Purchases	1411

Attribute Creation

A new categorial attribute was created to enable analysis of players as broken into 2 categories (HighRollers and PennyPinchers). A screenshot of the attribute follows:





Row ID	teamI...	platfor...	count...	count...	count...	avg_price
Row4	1	android	36	0	1	PennyPinchers
Row11	1	iphone	129	9	1	HighRollers
Row13	1	android	102	14	1	PennyPinchers
Row16	1	android	90	10	1	PennyPinchers
Row31	1	iphone	51	8	1	HighRollers
Row49	1	android	51	6	2	PennyPinchers
Row50	1	android	47	5	2	PennyPinchers
Row68	1	android	47	7	1	PennyPinchers
Row72	1	iphone	76	7	1	HighRollers
Row122	1	iphone	177	25	2	HighRollers
Row131	1	iphone	37	2	1	HighRollers
Row137	1	iphone	37	5	2	HighRollers
Row140	1	iphone	75	5	1	HighRollers
Row153	1	linux	51	9	1	PennyPinchers
Row159	1	android	40	3	1	HighRollers
Row186	1	iphone	54	6	1	HighRollers
Row196	1	iphone	38	3	2	HighRollers
Row210	1	iphone	32	2	1	HighRollers
Row215	1	iphone	45	6	2	HighRollers
Row218	1	android	46	3	1	HighRollers
Row222	1	android	53	9	1	PennyPinchers
Row229	1	iphone	57	2	1	HighRollers
Row244	1	iphone	28	3	1	HighRollers
Row251	1	android	80	8	1	PennyPinchers
Row258	1	windows	178	22	1	PennyPinchers
Row271	1	android	87	14	1	PennyPinchers
Row272	1	iphone	52	5	3	HighRollers
Row273	1	linux	121	16	1	PennyPinchers
Row286	1	windows	41	5	1	PennyPinchers
Row292	1	linux	34	3	1	PennyPinchers
Row297	1	android	66	12	1	PennyPinchers
Row302	1	windows	43	4	1	HighRollers
Row307	1	iphone	22	4	1	PennyPinchers
Row326	1	iphone	272	28	1	PennyPinchers
Row329	1	android	646	54	1	PennyPinchers
Row345	1	iphone	36	2	1	HighRollers

A new attribute was created basing on avg_price variable. Applying the 5$ criteria, we distinguished two categories of samples: HighRollers and PennyPinchers.

The creation of this new categorical attribute was necessary because:
We need to define a binary target variable to determine what event we want to model. It is neccessary for training the model according to the event.

Attribute Selection

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

Attribute	**Rationale for Filtering**

userId	It is not relevant because it does not provide information for predictio.
avg_price	It has already been taken into account
userSessionId	It is not relevant for predicting if users are likely to purchase big-tiket items.
	<Optional Fill in 1-3 sentences>

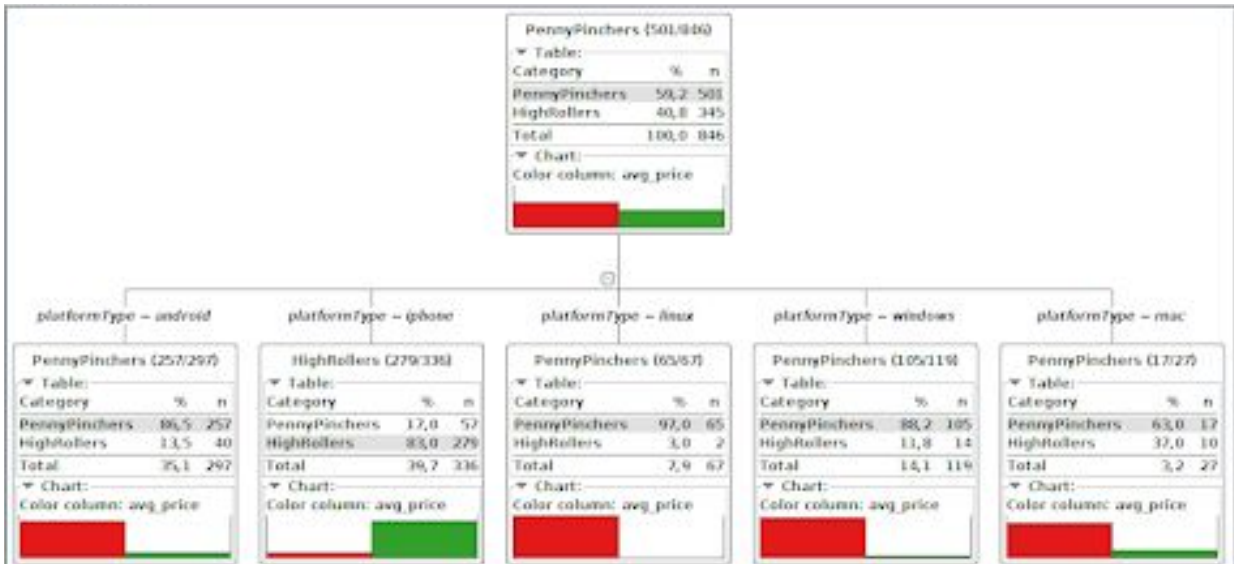
Data Partitioning and Modeling

The data was partitioned into train and test datasets.
The train data set(60%) was used to create the decision tree model.
The trained model was then applied to the test dataset.

This is important because it's a good practice to avoid overfitting,

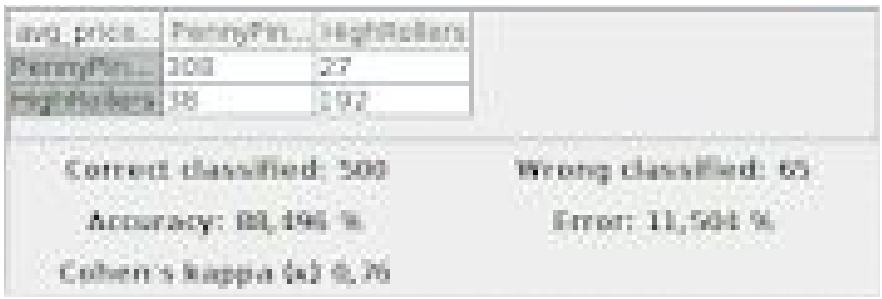
When partitioning the data using sampling, it is important to set the random seed because it's important to maintain the reproducibility.

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



Evaluation

A screenshot of the confusion matrix can be seen below:

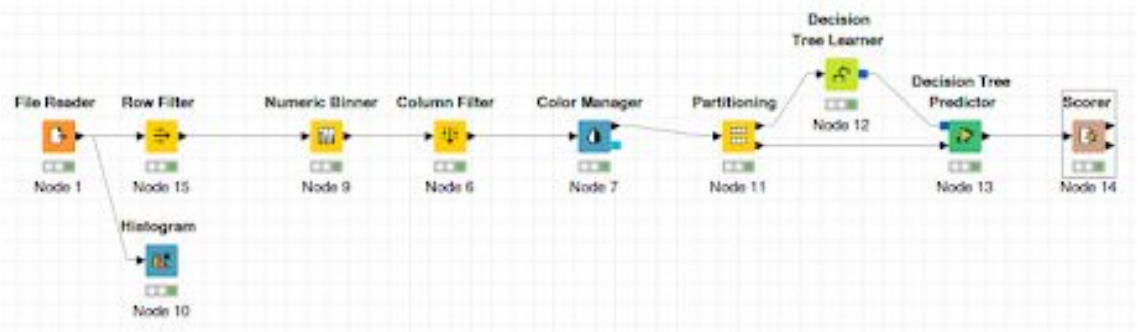


As seen in the screenshot above, the overall accuracy of the model is 87.08%

246 were classified PenneyPicher correctly. True positive
29 were classified PenneyPicher incorrectly. False positive
38 were classified HighRollers incorrectly. False positive
192 were classified HighRollers correctly. True positives

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller?

According to the classification tree, iphone users tend to spend more on their purchases when comparing with users that use other platforms.

HighRollers are notoriously associated with the Iphone plataform, given the fact that over 90% of the samples used to train the decision tree are from Iphone users. On the other hand windows has the highest presence of PennyPichers but is around 80% and the difference with the rest of the plataforms remains insignificant. HighRollers is almost sinonim of iPhone users in this data.

Specific Recommendations to Increase Revenue
1. Focus development and marketing initiatives on iphone users
2. Give preference in terms of new developments or marketing to mobile users.

Clustering Analysis

Attribute Selection

Attribute	**Rationale for Selection**
revenue	Represents the amount of money spent by a user in the game

gameClickSum	Sum of clicks on the game can be understood as engagement with the game
adClickSum	Sum of clicks on ads can tanslate on the quantifying of chance of purchase

Training Data Set Creation

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

```
In [34]: cluster_df = combined[['adClickSum', 'gameClickSum','revenue']]
         cluster_df.head(5)
```

Out[34]:

	adClickSum	gameClickSum	revenue
0	44	716	21.0
1	10	360	53.0
2	37	508	80.0
3	19	3107	11.0
4	46	704	215.0

```
In [35]: cluster_df.shape
Out[35]: (543, 3)
```

Dimensions of the training data set (rows x columns) : 543 X 3

# of clusters created: 3

Cluster Centers

Cluster #	**Cluster Center:** a*dClickSum, gameClickSum and revenue*
0	24.99 , 357.96 , 35.07
1	36.44 , 926.12 , 46.97
2	32.36 , 2310.64 , 39.42

Cluster 0: Customers on this cluster are not heavy players but still generate some revenue
Cluster 1: Customers of this cluster tend to be very cash convertible with ads even if they play a moderate amount, generating a high amount of revenue

Cluster 2: Customers that are very active and generate a moderate amount of revenue

Recommended Actions

Action Recommended	**Rationale for the action**
Promote / stimulate low cost and promotional offers in order to stimulate purchases from user of Cluster 2 customers	As we know, customers from cluster 2 are very active and generate a moderate amount of revenue. If we could stimulate them to buy a bit more, we could take advantage of their habit. To stimulate this consumption the idea is to have low cost / promotional offers, on this way appealing for the kind of purchase that better fit this profile
Promote / stimulate premium and featured offers in order to stimulate purchases from user of Cluster 1 customers.	Cluster 1: As we know, customers from cluster are very cash convertible with ads even if they play a moderate amount, generating a high amount of revenue. These guys like to spend their money, even if they aren't playing all the time. We should target premium features and personalize adds so that they are aware and can spend their cash.

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

A graph is used to represent the chat data model because its composed of several entities that relationships among them, for example: When one User creates a TeamChatSession, it is then owned by team. Users can join and leave the TeamChatSession. In TeamChatSession, users can

create ChatItem that is part of TeamChatSession. ChatItem could also be mentioned by Users. And User could respond to User as well. All the relationships are recorded with timestamp.

- Vertices (Entities)
  - User
  - Team
  - TeamChatSession
  - ChatItem
- Edges (Relationships)
  - User creates TeamChatSession with timestamp
  - Team owns TeamChatSession with timestamp
  - User joins TeamChatSession with timestamp
  - User leaves TeamChatSession with timestamp
  - User creates ChatItem with timestamp
  - ChatItem is part of TeamChatSession with timestamp
  - ChatItem is mentioned by User with timestamp
  - ChatItem responses to ChatItem with timestamp

Creation of the Graph Database for Chats

i)

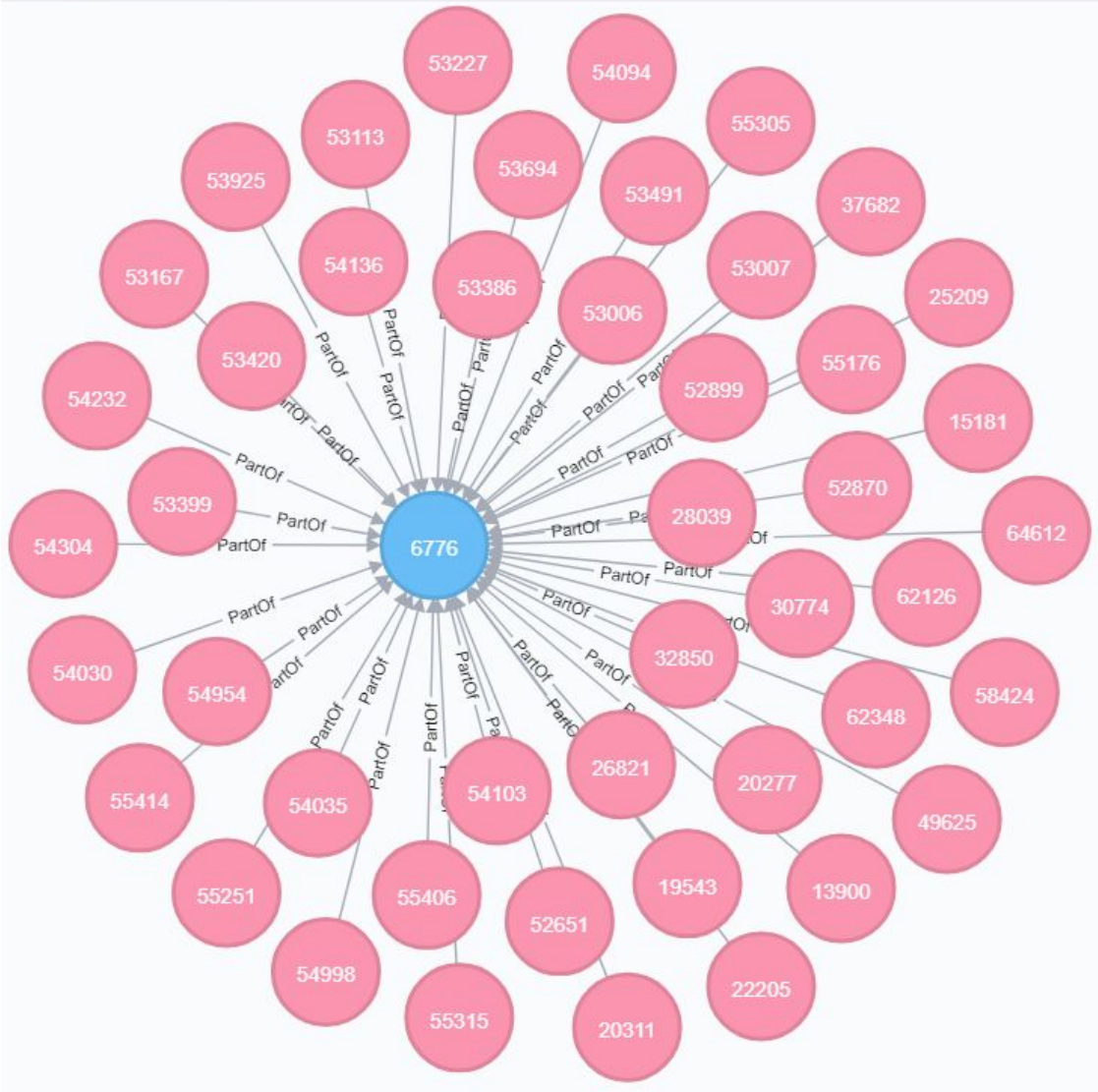
File Name	Fields	Description
chat_create_team_chat.csv	userID	the user id assigned to the user
	teamID	the id of the team
	teamChatSessionID	a unique id for the chat session

	timestamp	a timestamp denoting when the chat session created
chat_item_team_chat.csv	userID	the user id assigned to the user
	teamChatSessionID	a unique id for the chat session
	chatItemID	a unique id for the chat item
	timestamp	a timestamp denoting when the chat item created
chat_join_team_chat.csv	userID	the user id assigned to the user
	teamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the user join in a chat session
chat_leave_team_chat.csv	userID	the user id assigned to the user
	teamChatSessionID	a unique id for the chat session
	timestamp	a timestamp denoting when the user leave a chat session
chat_mention_team_chat.csv	chatItemID	the id of the ChatItem
	userID	the user id assigned to the user
	timestamp	a timestamp denoting when the user mentioned by a chat item
chat_respond_team_chat.csv	chatID1	the id of the chat post 1
	chatID2	the id of the chat post 2
	timestamp	a timestamp denoting when the chat post 1 responds to the chat post 2

```
# Clear database
MATCH (n)
OPTIONAL MATCH (n)-[r]-()
DELETE n,r

# Create the constraint primary key
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 chat_create_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/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)

# Load chat_join_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/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)

# Load chat_leave_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_leave_team_chat.csv" AS row MERGE (u:User {id: toInt(row[0])}) MERGE (c:TeamChatSession {id: toInt(row[1])}) MERGE (u)-[:Leave{timeStamp: row[2]}]->(c)

# Load chat_item_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/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)

# Load chat_mention_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/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 chat_respond_team_chat.csv
LOAD CSV FROM "file:///Users/iBowen/Desktop/chat-data/chat_respond_team_chat.csv" AS row MERGE (i:ChatItem {id: toInt(row[0])}) MERGE (j:ChatItem {id: toInt(row[1])}) MERGE (i)-[:ResponseTo {timeStamp: row[2]}]->(j)
```

A screenshot of the graph generated with clearly visible examples of most node and edge types.



Finding the longest conversation chain and its participants

The length of the conversation is 9 and the number of unique users that were part of the conversation chain, is 5. The steps taken consist on taking the length of the conversation first match statement and subsequently taking the nodes that went from users to the element of this chain. The query that produces the correct answer:

- Finding the longest conversation chain

```
match p = (i1)-[:ResponseTo*]->(i2)

return length(p)

order by length(p) desc limit 1
```

- Answer:

```
$ match p = (i1)-[:ResponseTo*]->(i2) return length(p) order by length(p) desc limit 1
```

Table

length(p)
9

- Unique users were part of this chain

```
match p = (i1)-[:ResponseTo*]->(i2)
where length(p) = 9
with p
match (u)-[:CreateChat]->(i)
where i in nodes(p)
return count(distinct u)
```

```
$ match p = (i1)-[:ResponseTo*]->(i2) where length(p) = 9 with p match (u)-[:Creat
```

Table

count(distinct u)
5

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.

The process of finding the chattiest users involves finding the origins of the edge create chat counting the Id's in descending order and looking at the first ten elements. To find the chattiest team a longer chain needed to be used having the form:

(:ChatItem)-[r:PartOf]->(:TeamChatSession)-[k:OwnedBy]->(n)

- Query

```
match (u)-[:CreateChat*]->(i)
```

return u.id, count(i)

order by count(i) desc limit 10

- Result

```
$ match (u)-[:CreateChat*]->(i) return u.id, count(i) order by count(i) desc limit 10
```

Table | Text | Code

u.id	count(i)
394	115
2067	111
1087	109
209	109
554	107
999	105
516	105
1627	105
461	104
668	104

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

- Query

match (i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)

return t.id, count(c)

order by count(c) desc limit 10

- Result

```
$ match (i)-[:PartOf*]->(c)-[:OwnedBy*]->(t) return t.id, count(c) order by count(c) desc limit 10
```

Table | Text | Code

t.id	count(c)
82	1324
185	1036
112	957
18	844
194	836
129	814
52	788
136	783
146	746
81	736

Chattiest Teams

Teams	Number of Chats
82	1324
185	1036
112	957

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

- Query

match (u)-[:CreateChat*]->(i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)

return u.id, t.id, count(c)

order by count(c) desc limit 10

- Result

```
$ match (u)-[:CreateChat*]->(i)-[:PartOf*]->(c)-[:OwnedBy*]->(t) return u.id, t.id, count(c)
```

Table | Text | Code

u.id	t.id	count(c)
394	63	115
2067	7	111
209	7	109
1087	77	109
554	181	107
516	7	105
1627	7	105
999	52	105
461	104	104
668	89	104

- Explanation

The user 999, which is in the team 52 is part of the top 10 chattiest teams, but other 9 users are not part of the top 10 chattiest teams. This demonstrates that most of the chattiest users are not in the 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.

- Constructing neighborhood of users using the following criteria: one mentioned another in a chat and one created a chatItem in response to another
    - Query:

match (u1:User)-[:CreateChat]->(i)-[:Mentioned]- >(u2:User) create (u1)-[:Deal]->(u2)

match (u1:User)-[:CreateChat]->(i1:ChatItem)- [:ResponseTo]- (i2:ChatItem) with u1, i1, i2 match (u2)-[:CreateChat]-(i2) create (u1)-[:Deal]->(u2)

- Removing self-loop
    - Query:

match (u1)-[r:Deal]->(u1) delete r

        - Removing self-loop from first query

match (u1:User)-[r1:Deal]->(u2:User) where u1.id <> u2.id with u1, collect(u2.id) as neighbors, count(distinct(u2)) as neighborAmount match (u3:User)-[r2:Deal]->(u4:User) where (u3.id in neighbors) AND (u4.id in neighbors) AND (u3.id <> u4.id) return u3.id, u4.id, count(r2)

- Removing duplicated edges count, counting unique entries
    - Query:

match (u1:User)-[r1:Deal]->(u2:User) where u1.id <> u2.id with u1, collect(u2.id) as neighbors, count(distinct(u2)) as neighborAmount match (u3:User)-[r2:Deal]->(u4:User) where (u3.id in neighbors) AND (u4.id in neighbors) AND (u3.id <> u4.id) return u3.id, u4.id, count(r2), case when count(r2) > 0 then 1 else 0 end as value

- Getting coefficient

match (u1:User)-[r1:Deal]->(u2:User) where u1.id <> u2.id with u1, collect(u2.id) as neighbors, count(distinct(u2)) as neighborAmount match (u3:User)-[r2:Deal]->(u4:User) where (u3.id in neighbors) AND (u4.id in neighbors) AND (u3.id <> u4.id) with u1, u3, u4, neighborAmount, case when (u3)-->(u4) then 1 else 0 end as value return u1, (sum(value)/(neighborAmount*(neighborAmount-1))) as coeff order by coeff desc limit 3

User ID	Coefficient
209	0,95
554	0,9
1087	0,8

Recommended Actions

- Android and Windows platform users are a representative part of the total of users so it makes sense invest on marketing and development to improve the turnover of sales on this group.
- Due to the high spending of the Iphone users, it worths invest on marketing to bring more users of this platform.
- Provide low rate fix pay for small advantages to promote low level spending groups.