Improving Revenue Generation on Catch the Pink Flamingo

by Sarsiz Chauhan

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	ERD TABLE	DESCRIPTION	FIELDS
ad-clicks.csv	AdClicks	A line is added to this file when a player clicks on an advertisement in the Flamingo app.	**timestamp** when the click occured unique id for the click (within ad-clicks.log) for the click id of user session for user who made click current team id of user who made the click user id of user who made the click id of the ad licked on category/type of ad clicked on

buy-clicks.csv	InAppPurchases	A line is added when a player makes an in-app purchase on Flamingo app	**timestamp** **txId** **userSessionid** **team** **userid** **buyId** **price**	when the click occured unique id for the click (within ad-clicks.log) id of user session for user who made click current team id of user who made the purchase user id of user who made the click id of the item purchased price of the item purchased
users.csv	User	File contains a line for each user playing the game.	**timestamp** **userId** **nick**	when the click occured user id of user who made the click nickname chosen by the user

			twitter handle of the user date of birth of the user 2-letter country code where the user lives
team.csv	Team	File contains a line for each team terminated in the game.	id of the team name of the team timestamp when team was created timestamp when last member of the team measure of team strength roughly corresponding to the success of a time current level of a team
team-assignments.csv	TeamAssignment	A line is added each time a user joins a team. A user can be in at most a single team at a time.	when the user joined the team id of the teamuser id of the user unique id for this assignment
level-events.csv	LevelEvent	A line is added each time a team starts or finishes a level in the game.	when the click occured unique id for the event id of the team level started or completed type of event (start or end)

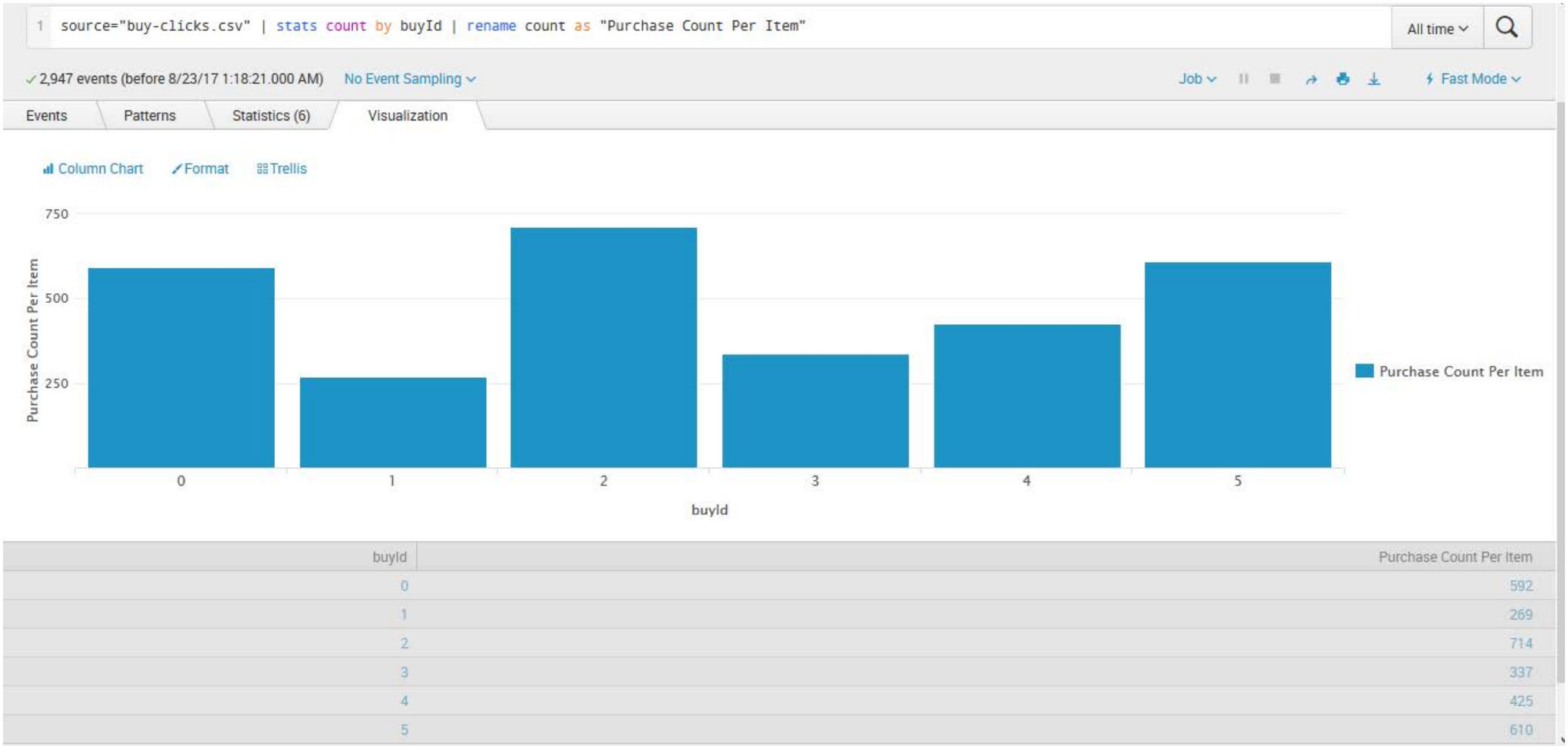
user.session.csv	User_Sessions	Each line describes a user session, which denotes when a user starts and stops playing the game. When a team goes to next game level, the session is ended for each user in the team and a new one is started.	when the click occured unique id for the session current user’s ID current user’s team team assignment id for the user to the team whether the event is the start or end of a session level of team during the session type of platform of the user during the session
game-clicks.csv	GameClicks	A line is added each time a user performs a click in the game.	when the click occured unique id for the click click user’s ID id of the session of user when click occurs if click hits flamingo (val=1) or missed (val=0) id of the team of the user id of the team of user level of team during the session

Aggregation

Amount spent buying items	source="buy-clicks.csv" | stats sum(price)	21407.0
Number of unique items available to be purchased	source="buy-clicks.csv" | stats dc(buyId)	6

Histograms

1. A histogram showing how many times each item is purchased:



buyId	Purchase Count Per Item
0	592
1	269
2	714
3	337
4	425
5	610

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

```
1 source="buy-clicks.csv" | stats sum(price) by buyId | rename sum(price) as "Total Revenue Per Item"
```

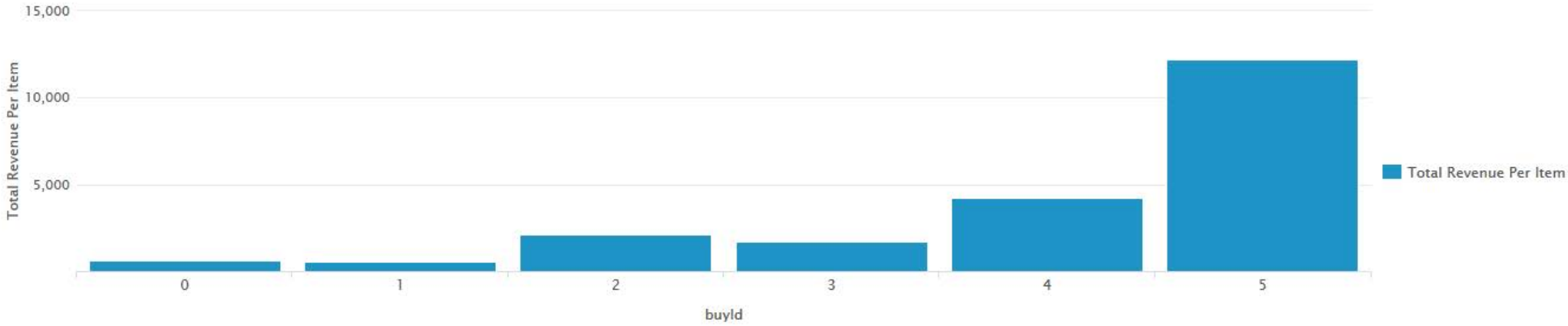
All time

✓ 2,947 events (before 8/23/17 1:25:16.000 AM) No Event Sampling

Job Fast Mode

Events Patterns Statistics (6) Visualization

Column Chart Format Trellis



buyId	Total Revenue Per Item
0	592.0
1	538.0
2	2142.0
3	1685.0
4	4250.0
5	12200.0

Filtering

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

```
source="buy-clicks.csv" | stats sum(price) by userId |sort sum(price) desc |rename sum(price) as "Total Money Spent" |head 10
```

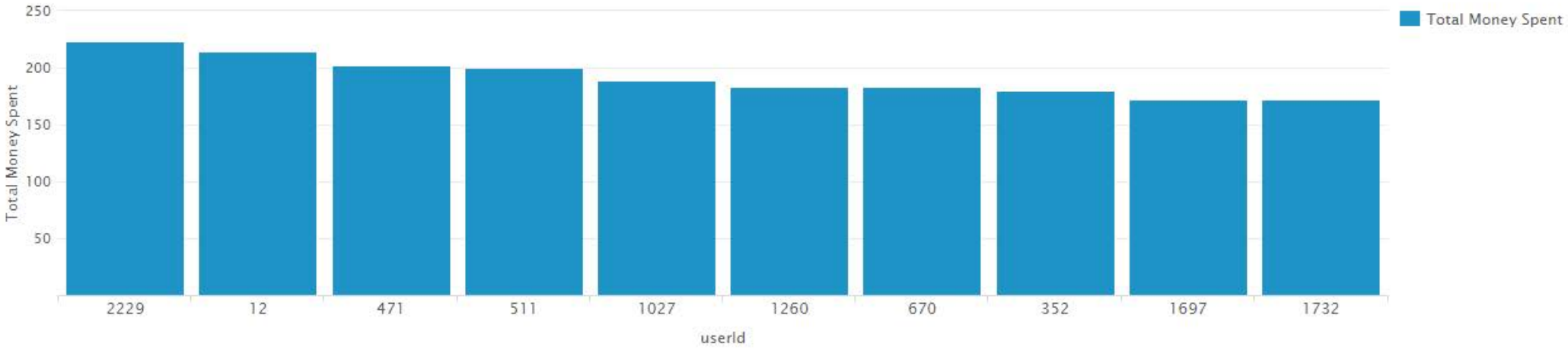
All time

✓ 2,947 events (before 6/23/17 8:40:48.000 PM) No Event Sampling

Job Smart Mode

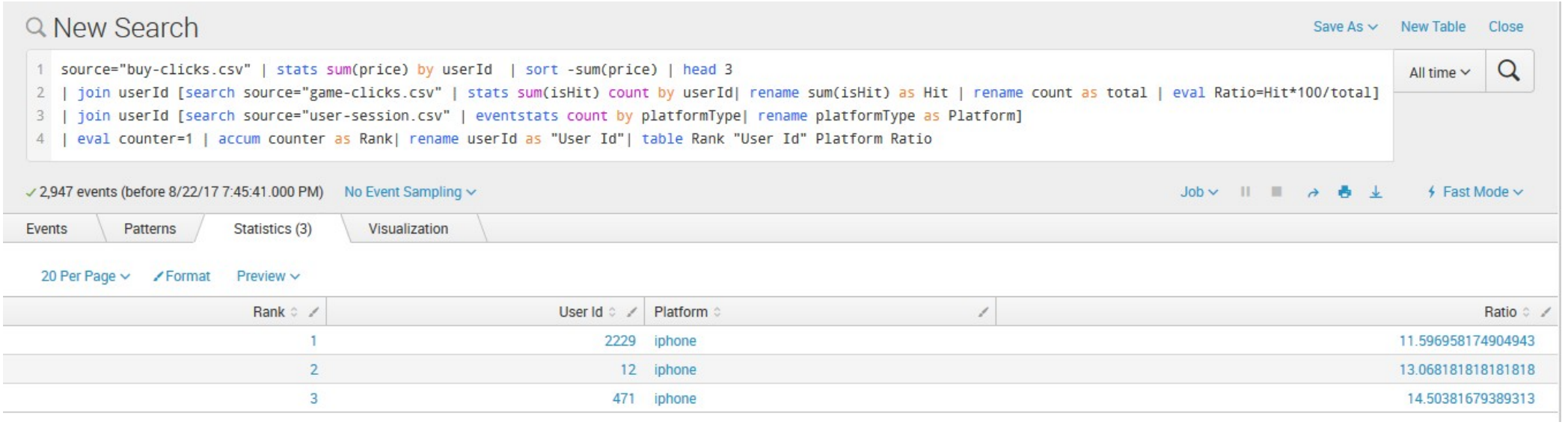
Events Patterns Statistics (10) Visualization

Column Chart Format



userId	Total Money Spent
2229	223.0
12	215.0
471	202.0
511	200.0
1027	189.0
1260	183.0
670	183.0
352	180.0
1697	172.0
1732	172.0

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

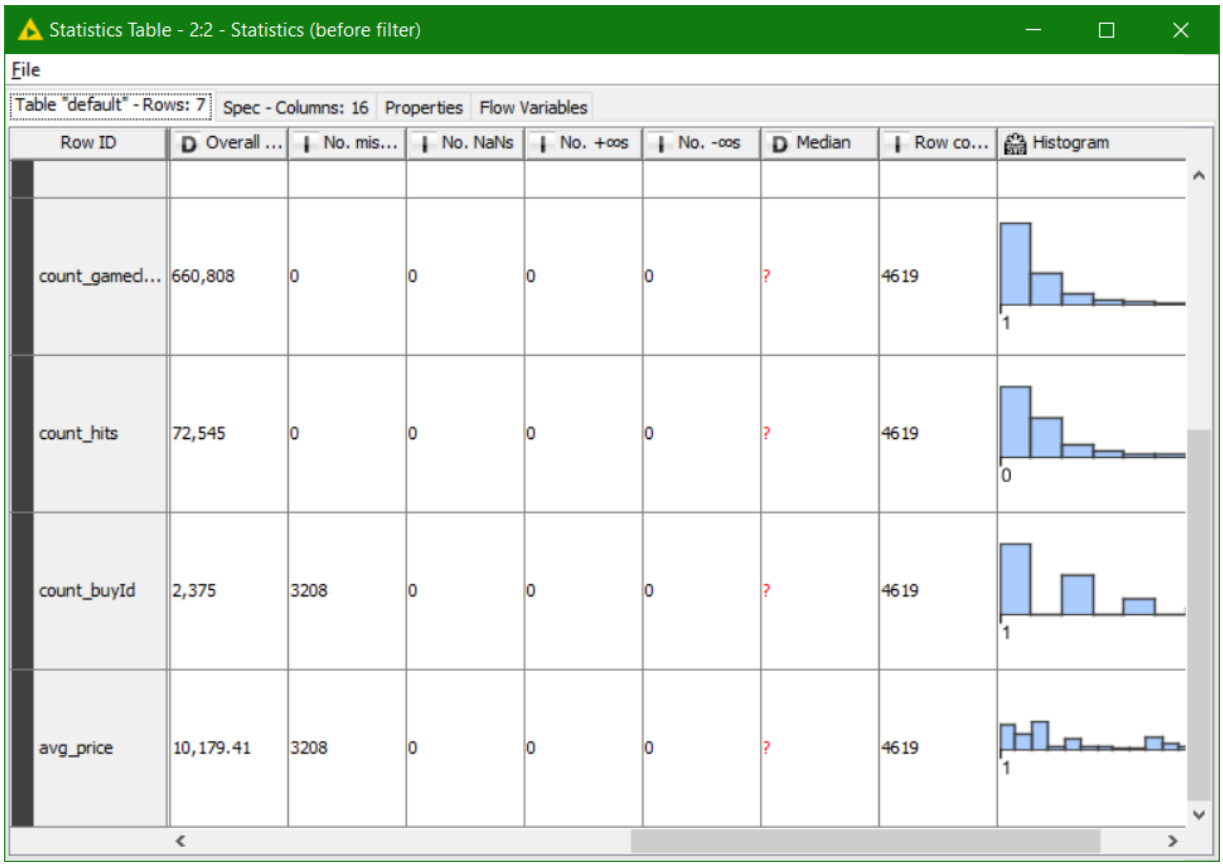


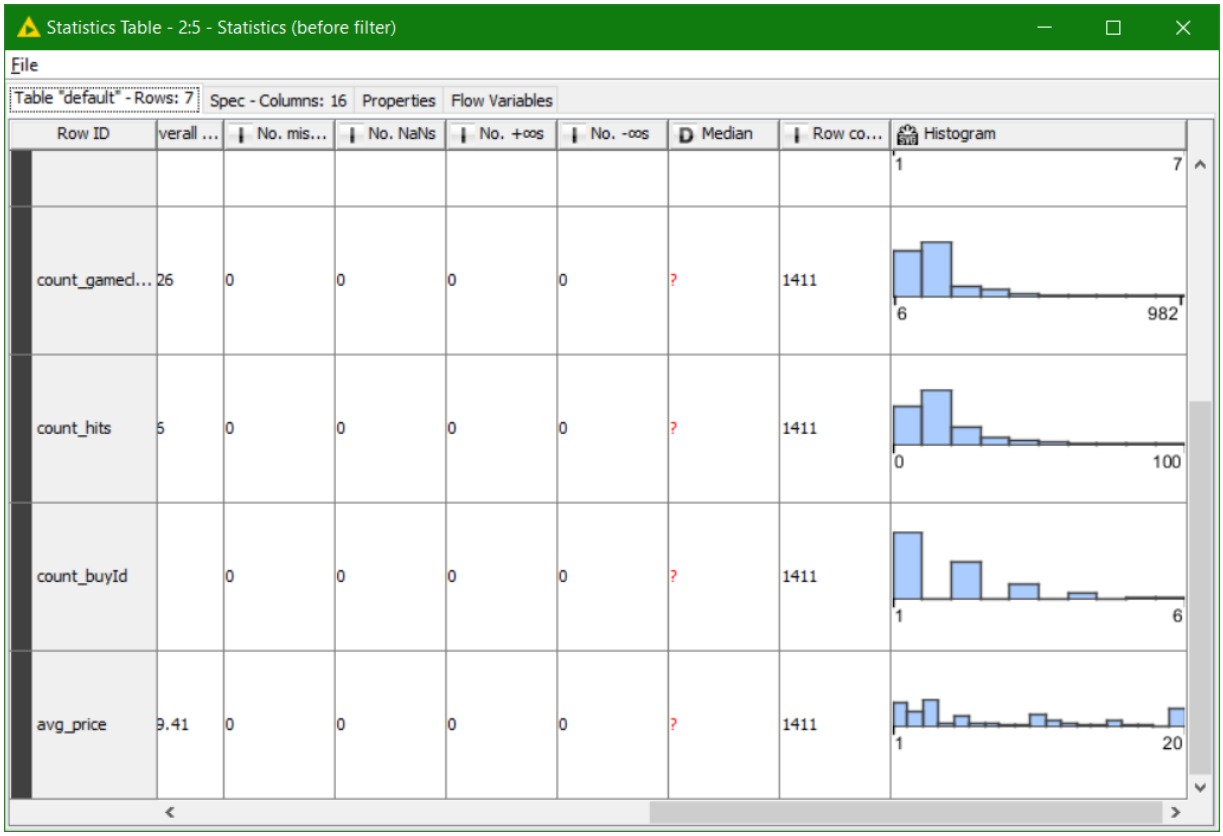
Data Preparation

Analysis of combined_data.csv

Sample Selection

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



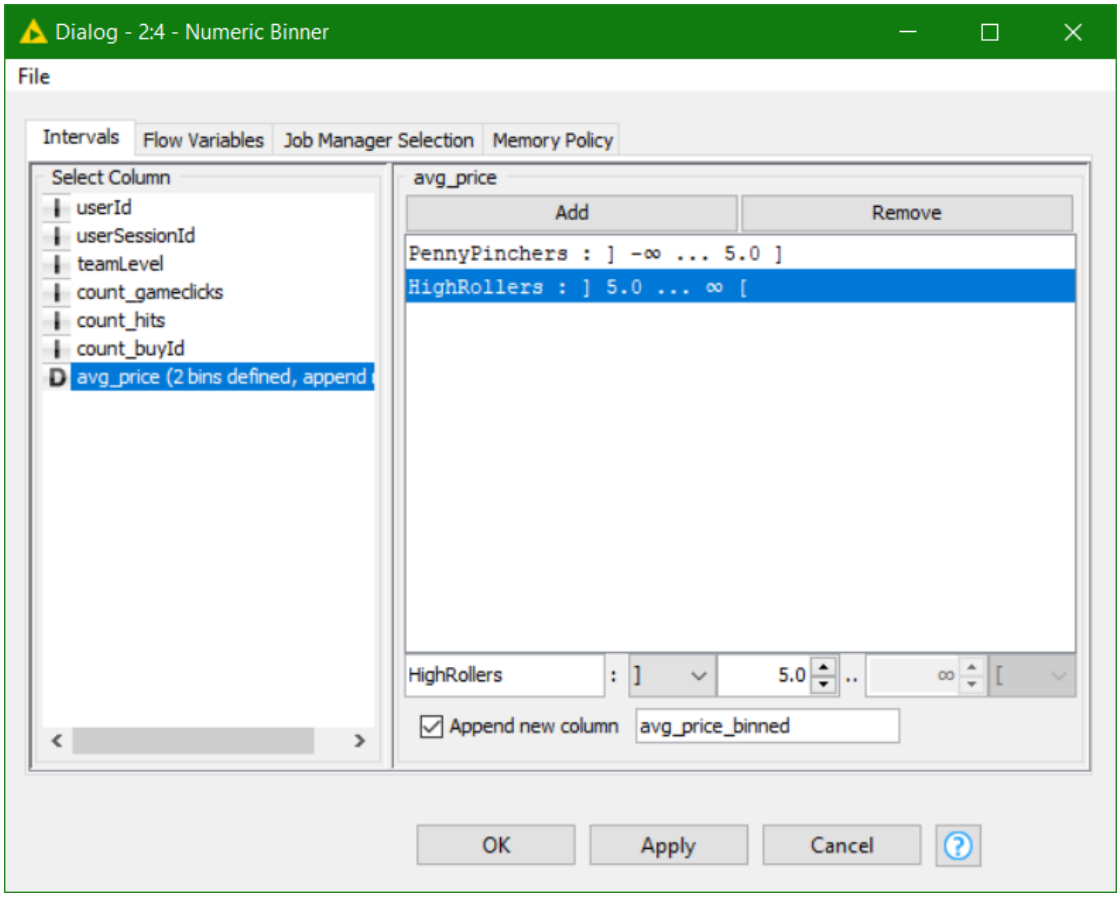


v

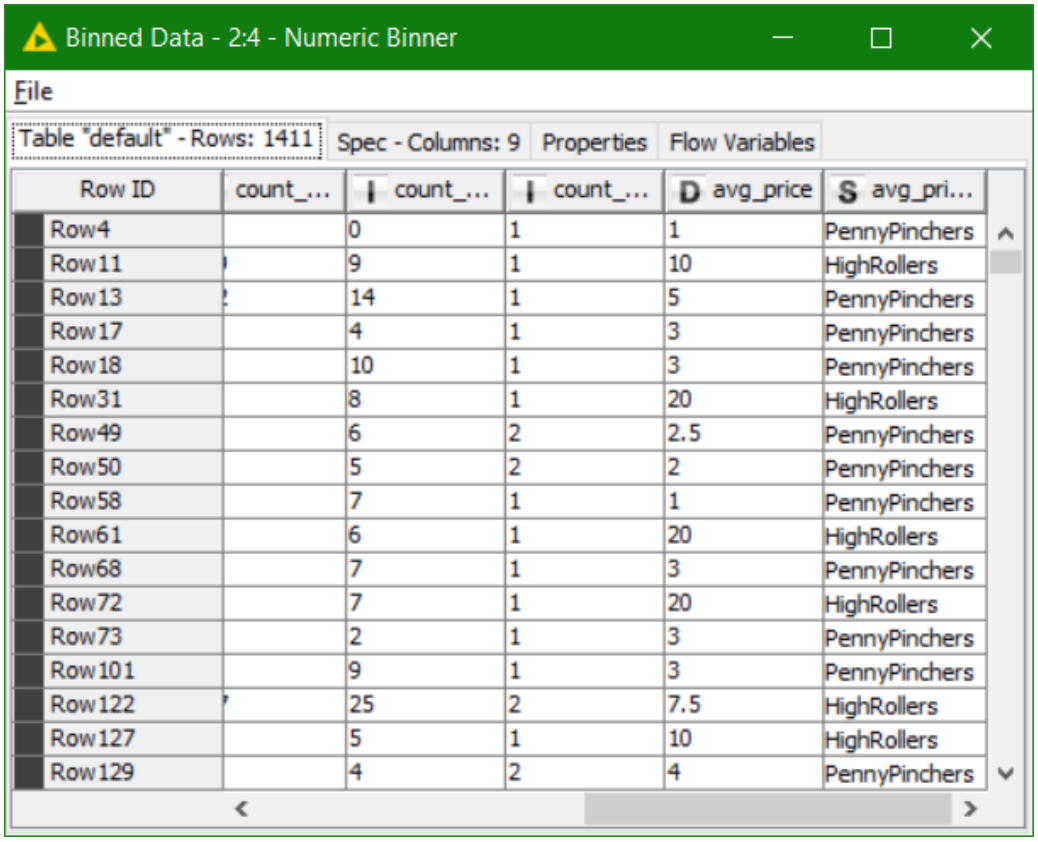
Here I have taken two snaps. First one is before applying filters (**4619 rows**) and the second one is after applying filters. Those rows which have NULL as a value are removed from the combined_data.csv file and then the statistics are observed similar to the first. After filtering we get **1411** rows and also the graph is quite similar to the previous one.

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 numeric avg_price variable was redefined as a category variable with 2 values: PennyPinchers and HighRollers. Penny Pinchers were those who bought items costing $5.00 or less, and HighRollers are those users who bought items costing above $5.00. The design is shown above, **where "]" is inclusive**, and **"[" is exclusive**. The new category variable is named **"avg_price_binned."**



Row ID	count_...	count_...	count_...	avg_price	avg_pri...
Row4		0	1	1	PennyPinchers
Row11		9	1	10	HighRollers
Row13		14	1	5	PennyPinchers
Row17		4	1	3	PennyPinchers
Row18		10	1	3	PennyPinchers
Row31		8	1	20	HighRollers
Row49		6	2	2.5	PennyPinchers
Row50		5	2	2	PennyPinchers
Row58		7	1	1	PennyPinchers
Row61		6	1	20	HighRollers
Row68		7	1	3	PennyPinchers
Row72		7	1	20	HighRollers
Row73		2	1	3	PennyPinchers
Row101		9	1	3	PennyPinchers
Row122		25	2	7.5	HighRollers
Row127		5	1	10	HighRollers
Row129		4	2	4	PennyPinchers

The creation of this new categorical attribute was necessary because we will be using a decision tree algo. to determine the attributes responsible for **highroller** and **a pennyPincher**. It will also serve as the reference for training and subsequently scoring the Decision Tree model.

Avg-price only can not be used for classification task with a continuous-value.

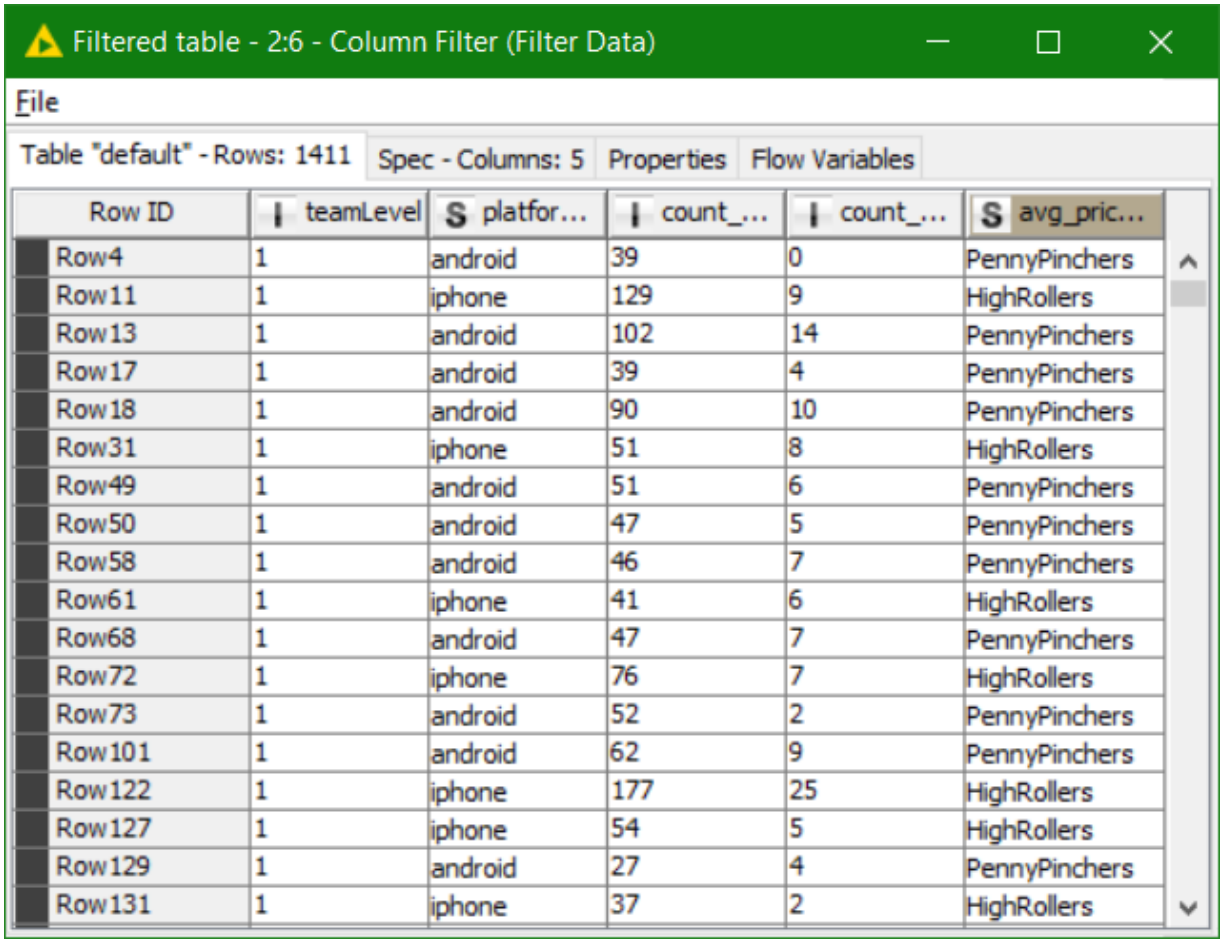
Attribute Selection

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

Attribute	**Rationale for Filtering**
userId	We are not interested in finding who exactly is a highRoller plus It doesn't make any sense on deciding whether a player is highRoller or a PennyPencher.
userSessionId	This attribute is used to identify the session and the session does

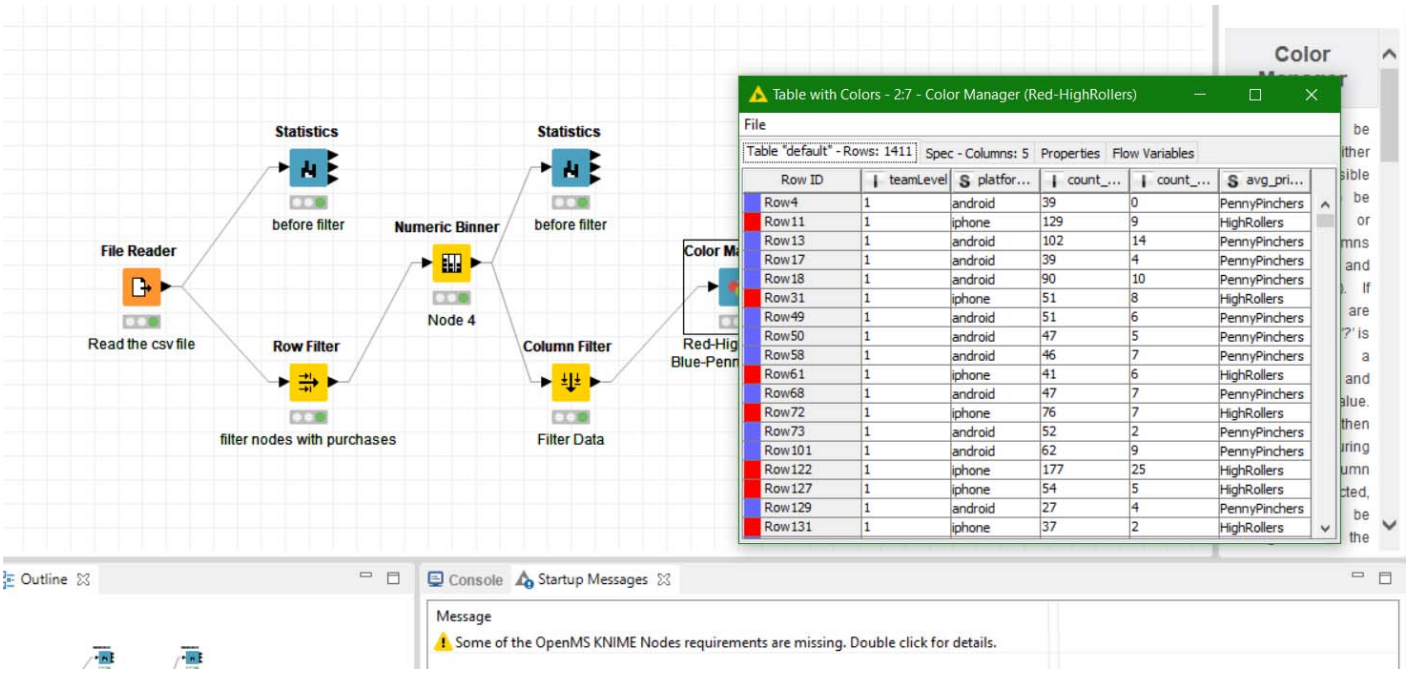
	not contribute to a highRoller or a PennyPincher
count_buyId	The number of items purchased doesn't define a highRoller
avg_price	The numeric variable was replaced by the category variable(Binned). So it cannot be included within the training and testing dataset as the decision tree will then be giving us 100% Accuracy which defeat the original objective of analysing the behaviour and predicting the results.

After applying column filter.
Now if you observe the last column, their you can find that it has specified the PennyPinchers and the highRollers



Row ID	teamLevel	platfor...	count_...	count_...	avg_pric...
Row4	1	android	39	0	PennyPinchers
Row11	1	iphone	129	9	HighRollers
Row13	1	android	102	14	PennyPinchers
Row17	1	android	39	4	PennyPinchers
Row18	1	android	90	10	PennyPinchers
Row31	1	iphone	51	8	HighRollers
Row49	1	android	51	6	PennyPinchers
Row50	1	android	47	5	PennyPinchers
Row58	1	android	46	7	PennyPinchers
Row61	1	iphone	41	6	HighRollers
Row68	1	android	47	7	PennyPinchers
Row72	1	iphone	76	7	HighRollers
Row73	1	android	52	2	PennyPinchers
Row101	1	android	62	9	PennyPinchers
Row122	1	iphone	177	25	HighRollers
Row127	1	iphone	54	5	HighRollers
Row129	1	android	27	4	PennyPinchers
Row131	1	iphone	37	2	HighRollers

The resulting table will be passed to the Color Manager node, where **High Rollers will be assigned a Red color** and **PennyPinchers a Blue.**



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.

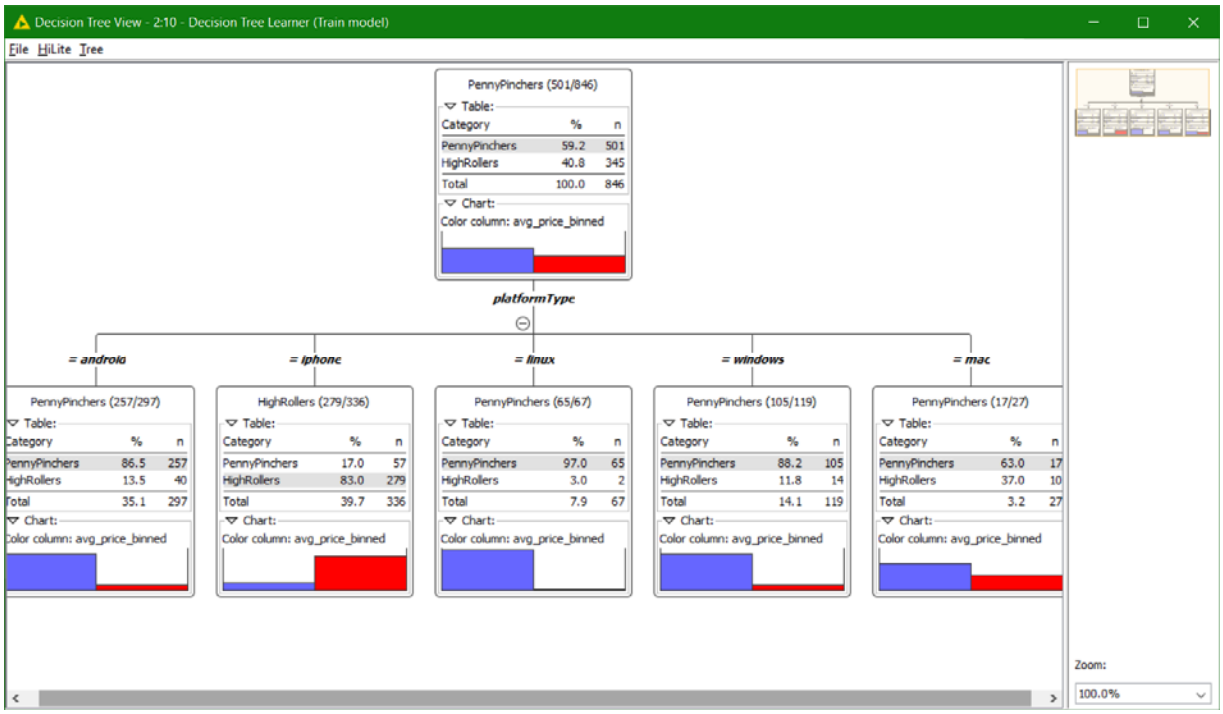
The trained model was then applied to the **Test** dataset.

This is important because the train data set consist of the records whose labels are already known and this will facilitate us to build the classification model(a decision tree) while the test data set would contain records with known labels, thus serving as an unbiased means of evaluating the performance of the trained model. Later on we will be comparing the results of the trained model(i.e accuracy, etc)

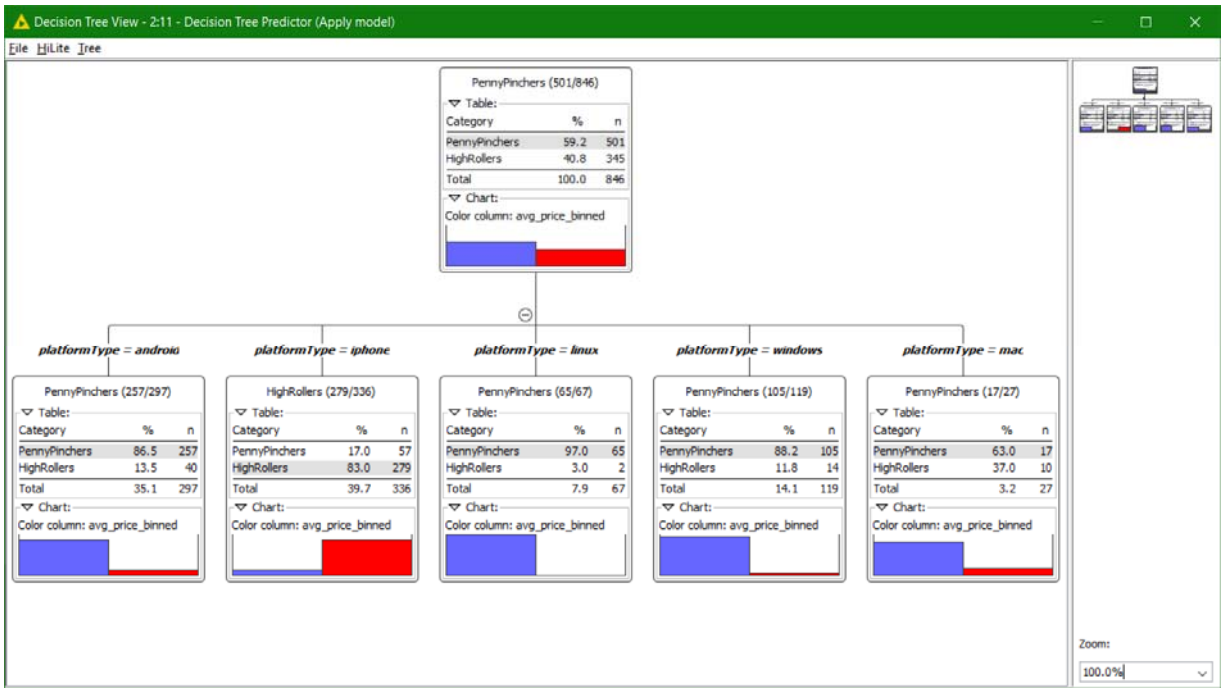
When partitioning the data using sampling, it is important to set the random seed because it ensures that I will get the same partitions every time I execute this node(i.e to replicate the same partitioned datasets for repeated executions of the training and scoring process). It is important to get reproducible results. Also, it is not set by default, so we will need to set it when we use this node.

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

Zoom-in the image to view more clearly

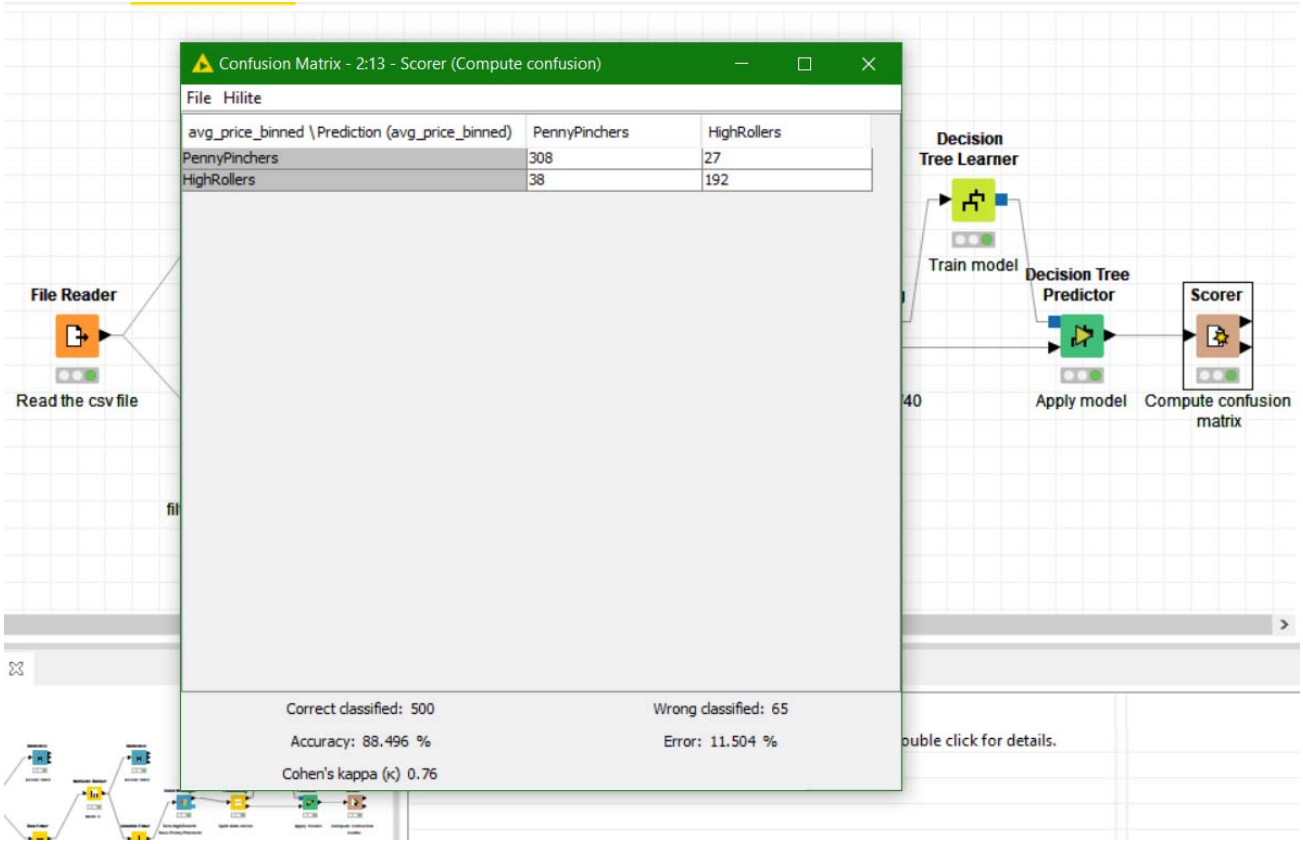


The following is the Decision Tree View of the Test Model



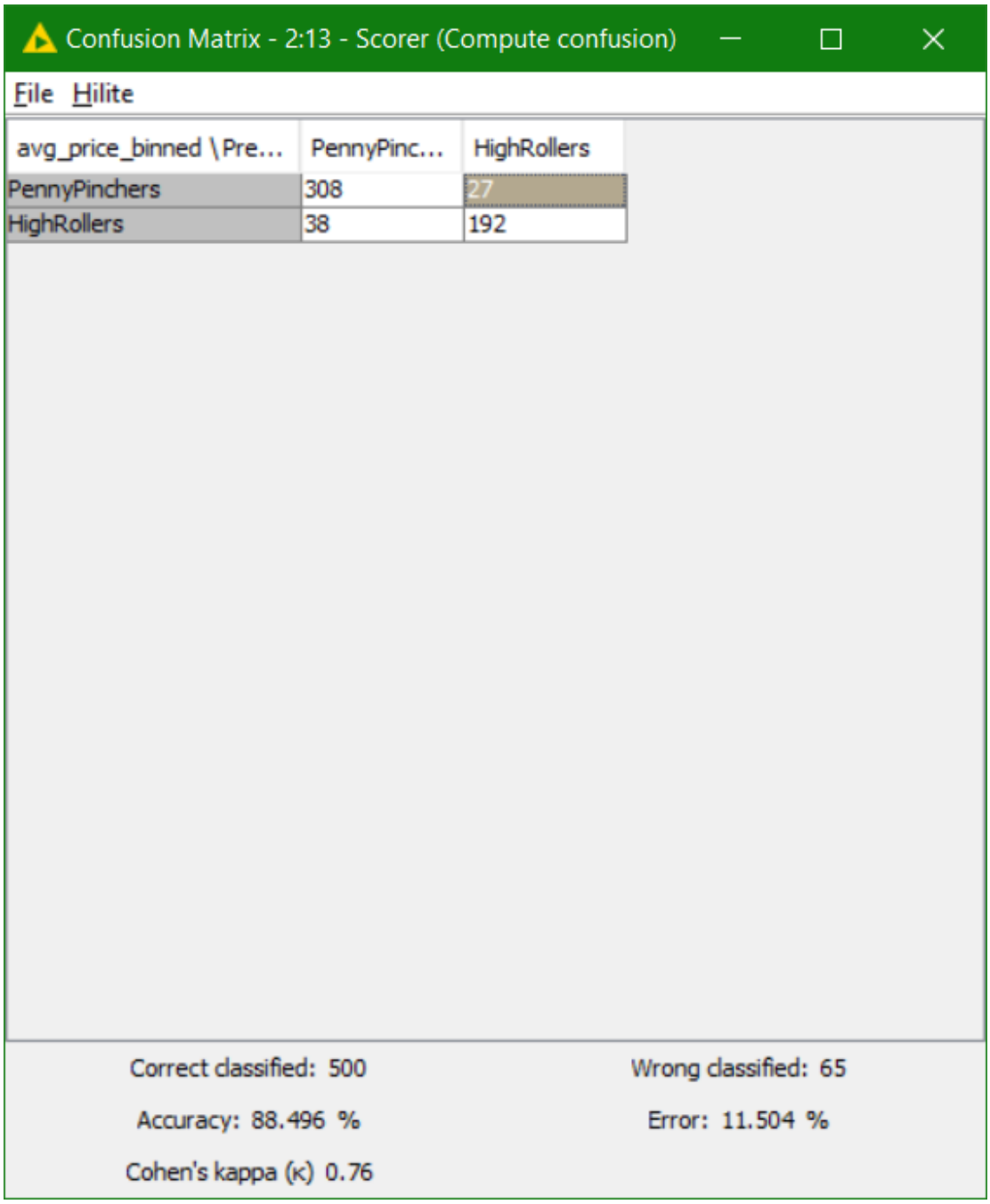
Evaluation

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%.

Confusion Matrix –



This shows that there are total **65** Wrong classified predictions (**38+27 = 65**)

Of the 335 Penny Pinchers in the test data set >>
308 (91.9%) of them were correctly predicted as Penny Pinchers by decision tree model.
27 (8.1%) of these Penny Pinchers were incorrectly predicted as High Rollers.

Of the 230 High Rollers in the test data set >>
192 (83.5%) of them were correctly predicted as High Rollers by decision tree model.
38 (16.5%) of these High Rollers were incorrectly predicted as Penny Pinchers.

Row 150 is highlighted. Wrong Prediction (Predicted – PennyPinchers, Actual – HighRollers)

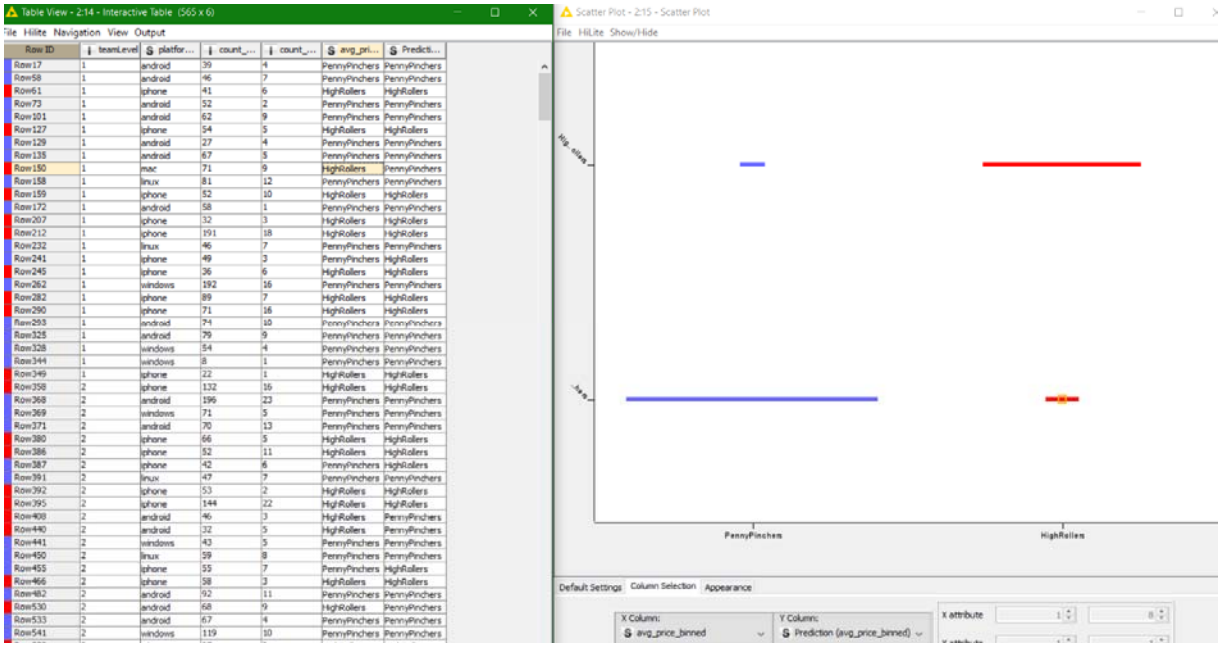


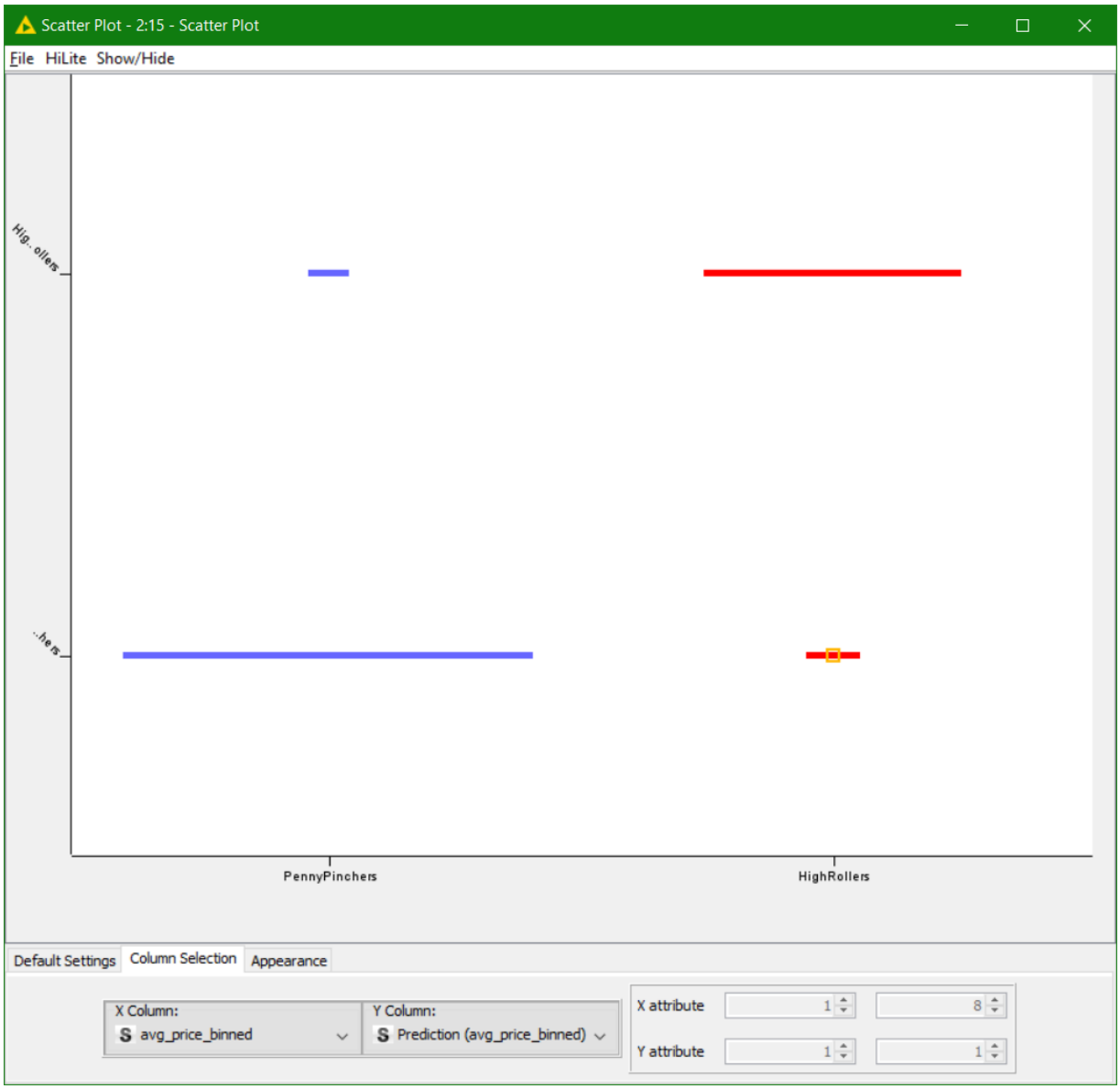
Table View - 2:14 - Interactive Table (565 x 6)

File Hilite Navigation View Output

Row ID	teamLevel	platfor...	count_...	count_...	avg_pri...	Predicti...
Row17	1	android	39	4	PennyPinchers	PennyPinchers
Row58	1	android	46	7	PennyPinchers	PennyPinchers
Row61	1	iphone	41	6	HighRollers	HighRollers
Row73	1	android	52	2	PennyPinchers	PennyPinchers
Row101	1	android	62	9	PennyPinchers	PennyPinchers
Row127	1	iphone	54	5	HighRollers	HighRollers
Row129	1	android	27	4	PennyPinchers	PennyPinchers
Row135	1	android	67	5	PennyPinchers	PennyPinchers
Row150	1	mac	71	9	HighRollers	PennyPinchers
Row158	1	linux	81	12	PennyPinchers	PennyPinchers
Row159	1	iphone	52	10	HighRollers	HighRollers
Row172	1	android	58	1	PennyPinchers	PennyPinchers
Row207	1	iphone	32	3	HighRollers	HighRollers
Row212	1	iphone	191	18	HighRollers	HighRollers
Row232	1	linux	46	7	PennyPinchers	PennyPinchers
Row241	1	iphone	49	3	PennyPinchers	HighRollers
Row245	1	iphone	36	6	HighRollers	HighRollers
Row262	1	windows	192	16	PennyPinchers	PennyPinchers
Row282	1	iphone	89	7	HighRollers	HighRollers
Row290	1	iphone	71	16	HighRollers	HighRollers
Row293	1	android	74	10	PennyPinchers	PennyPinchers
Row325	1	android	79	9	PennyPinchers	PennyPinchers
Row328	1	windows	54	4	PennyPinchers	PennyPinchers
Row344	1	windows	8	1	PennyPinchers	PennyPinchers
Row349	1	iphone	22	1	HighRollers	HighRollers
Row358	2	iphone	132	16	HighRollers	HighRollers
Row368	2	android	196	23	PennyPinchers	PennyPinchers

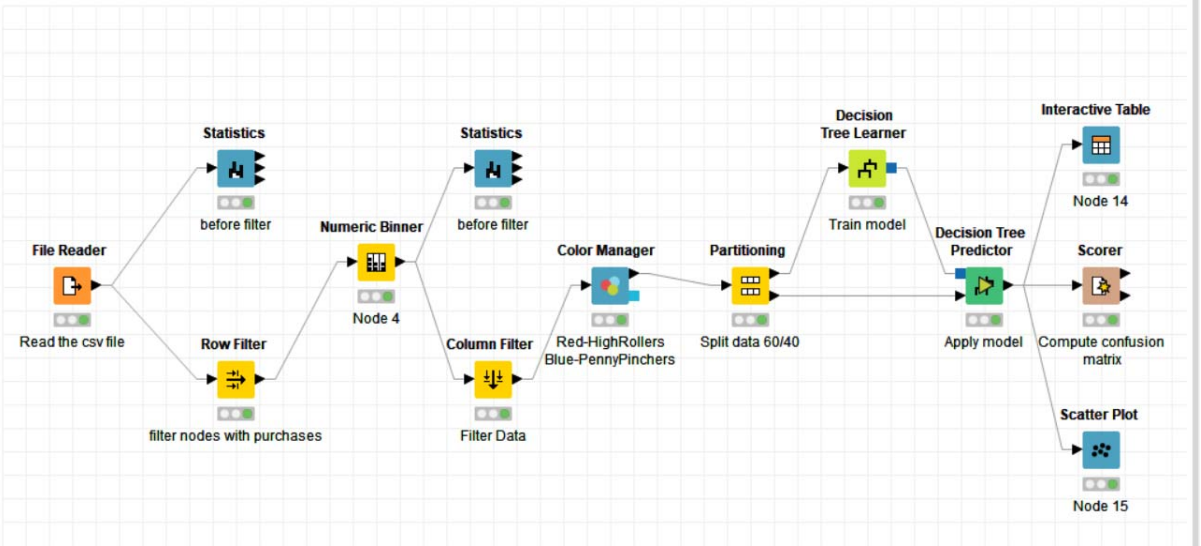
ROW 150 highlighted above is shown in the scatter plot below
The **wrong prediction are small horizontal blue line and the red line**. Both denote wrong Predictions.

The below graph means that if on both axis we have highRollers and Penny Pinchers then the prediction is good and since the success rate is 88.5% that's why we have long horizontal red and blue lines.


Scatter Plot - 2:15 - Scatter Plot
File HiLite Show/Hide
Hig.. ollers
..hers
PennyPinchers
HighRollers
Default Settings
Column Selection
Appearance
X Column:
avg_price_binned
Y Column:
Prediction (avg_price_binned)
X attribute
1
8
Y attribute
1
1

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

Based on the Decision Tree that was formulated, it can be concluded that players using iOS devices has been classified as HighRollers, while Players of other platforms have been classified as PennyPinchers.

Linux being the first in terms of PennyPinchers (97% users) and just 3% highrollers
Mac being the second in terms of HighRollers (37% users).
Android on third with 13.5% highRollers and 86.5% PennyPinchers.

Specific Recommendations to Increase Revenue
1. We should be Targeting the iOS users more than other platform users. A separate budget should be kept for promotion of the game on iOS platform so that the user base will increase and so does the revenue as iOS users being the HighRollers.
2. Encouraging them(iOS players) by providing rewards to write the good review of the game, so that other platform users can be influenced by the rating and reviews of the iOS platform. Thus increasing the Downloads and the user Base
3. Providing Special Discount to the Other platform users to increase the tendency of the user to make the purchase.
4. The most popular and frequently used item should be priced so that the users will be forced to make the purchase in order to play the game with ease. This method is not ideal as it will lead the players to uninstall the game. So this could be a way but not the best way

Attribute Selection

Attribute	Rationale for Selection
Average ad-click count	Similar to the in-app purchases, we record each ad-click by the user and using this we can further predict the tendency or chances that the user will click on the ads. With time some users will click ads more oftern than the others Thus it helps us to compare these. We will calculate the average ad-click count by counting all the ad-clicks for a user divide by the number of sessions played.
Average buy-click count	Whenever the user makes an in-app purchase it's recorded as a buy-click. So, as the game progress users make more purchases whilst making their way through the different levels in the game. We can calculate the avg-buy-click per level. From here we can see the trend if the users are making more or less purchases or anything else. Calculation : Count all the buy-clicks for a user dividing it by the number of sessions played.
Average Expenditure	We can study the average expenditure per level as there is a range of in-app game items for users to purchase. With this we can study the highrollers vs pennypinchers. Calculation : Count all the item-prices paid by a user then divide it by the number of sessions played.

Training Data Set Creation

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

avgPurchaseCount	avgAdClicks	avgExpPerSession
0	0	0
0.714285714	5.142857143	8.571428571
0	0	0
0	0	0
0	0	0
1.142857143	4.857142857	3.714285714
2	6.5	31.5
1	5	6.5
1.5	5.666666667	3.333333333
0	0	0

Dimensions of the training data set (rows x columns) : 1091 rows x 3 columns

# of clusters created: 3

Cluster Centers

Cluster #	Cluster Center
1	Average buy-clicks :: 0.02 Average Expenditure :: 0.03 Average ad-clicks :: 0.23 *Cluster size :: 589, 54% of dataset*
2	Average buy-clicks :: 0.94 Average Expenditure :: 4.30 Average ad-clicks :: 5.83 *Cluster size :: 390, 35.7% of dataset*
3	Average buy-clicks :: 1.58 Average Expenditure :: 20.19 Average ad-clicks :: 5.71 *Cluster size :: 112, 10.3% of dataset*

These clusters can be differentiated from each other as follows:

Cluster 1(freeloaders) is different from the others in that…
It has captured most of the users that essentially play the game without spending at all and typically clicks on 1 advertisement in every 4-5 games. They provide the basic behaviour of the users in the game

Cluster 2(pennypinchers) is different from the others in that…
It is characterised as users who makes near to 1 purchase every level. Their expenditure is around $4.303 within a level. They also display a tendency to click on nearly 6 advertisements per level. It is much higher than the freeloaders.

Cluster 3(highrollers) is different from the others in that…
1.58 purchases per level(68% more than the pennypinchers) and spending almost $20 per level (470% more than the pennypinchers). Also ad-click rate is just slightly lower by 2% than penny-pinchers, they also click close to 6 advertisements per level which is much higher than the freeloaders

Log file from cluster analysis :

Highlighted text indicates details of the model selected for evaluation.

[SAMPLE DATASET]
[array([0., 0., 0.]), array([0.71428571, 8.57142857, 5.14285714]), array([0., 0., 0.]), array([0., 0., 0.]), array([0., 0., 0.]),
array([1.14285714, 3.71428571, 4.85714286]), array([2. , 31.5, 6.5]), array([1. , 6.5, 5.]),
array([1.5 , 3.33333333, 5.66666667]), array([0., 0., 0.])]

[TRAIN DATASET SHAPE]
(1091, 3)

[K]=1
[CENTER(S)]
[array([0.50749203, 3.62997822, 2.79677448])]
[COST]=59925.05543342645
[CLUSTER SIZES]
dict_items([(0, 1091)])

[K]=2
[CENTER(S)]
[array([1.37703252, 17.03899811, 5.90320122]), array([0.35365747, 1.25772443, 2.24720167])]
[COST]=23212.28217540738
[CLUSTER SIZES]
dict_items([(0, 164), (1, 927)])

[K]=3
[CENTER(S)]
[array([0.93948718, 4.30493547, 5.8309768]), array([1.57476616, 20.18807802, 5.71264881]),
array([0.01850594, 0.03449349, 0.23324844])]
[COST]=12671.3123894749
[CLUSTER SIZES]
dict_items([(0, 390), (1, 112), (2, 589)])

[K]=4
[CENTER(S)]
[array([1.16061224, 11.48923622, 5.95703231]), array([0.01357597, 0.02816291, 0.15800116]),
array([1.84887218, 25.41132331, 5.46215539]), array([0.87687397, 2.79847867, 5.72487983])]
[COST]=7645.50590694292
[CLUSTER SIZES]
dict_items([(0, 140), (1, 577), (2, 57), (3, 317)])

[K]=5

[CENTER(S)]

[array([0.83032237, 2.41210121, 5.65188594]), array([1.14658163, 9.47171122, 6.06686224]),
array([0.01146922, 0.02439024, 0.14227642]), array([1.54826531, 19.3221017 , 5.38472789]),
array([2.28541667, 35.36238095, 6.1639881])]

[COST]=4983.973681127357

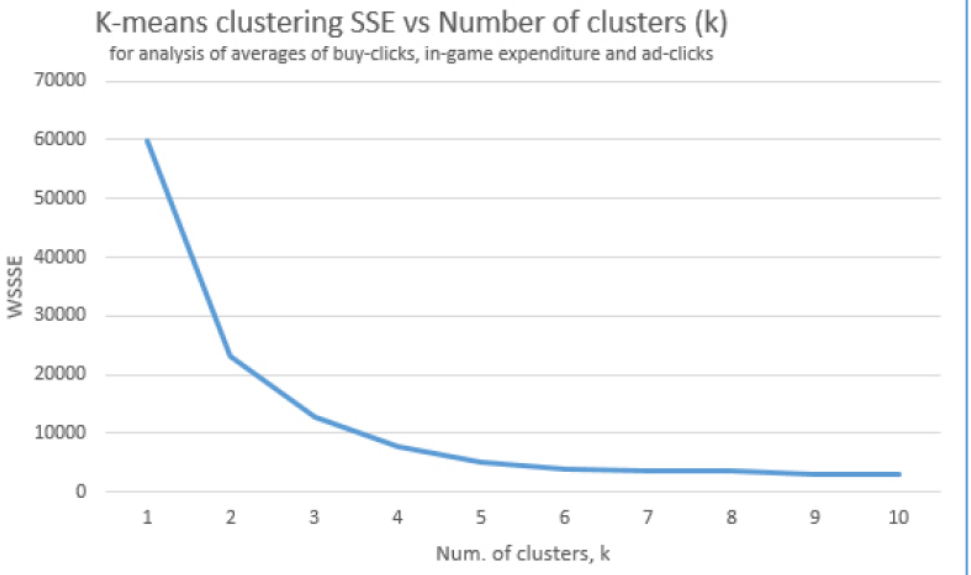
[CLUSTER SIZES]

dict_items([(0, 291), (1, 141), (2, 574), (3, 69), (4, 16)])

<Log truncated for brevity. k=6 onwards not included here>

Cost evaluation of cluster analysis

Num. of clusters, k	WSSSE
1	59925
2	23212
3	12671
4	7646
5	4984
6	3982
7	3610
8	3555
9	3166
10	3158



K=3 picked for evaluation of cluster analysis

Recommended Actions

Action Recommended	Rationale for the action
Provide multiple in-game purchases at time varying discounts	*Higher-rollers* make close to 1.5 purchases per level tend to spend much more than the *pennypinchers* (470%, i.e. almost 4 to 5 times more in the long run), thus a slight discount on bulk purchases can entice this group to continue purchasing the items. in the long term (perhaps maybe spend even more), or encourage the penny-pinchers to spend more. This would also lead to greater revenue generated from in-app purchases.
Assign high value ads to users who make at least 1 purchase per game.	The cluster analysis reveals that the users who make at least 1 ingame purchase per level have a tendency to click up close to 6 ads per level. This could optimise the revenue generated from ‘high-rate’ ads. The ‘common-rate’ ads can be assigned to the freeloaders.

Graph Analytics

Modeling Chat Data using a Graph Data Model

(Describe the graph model for chats in a few sentences. Try to be clear and complete.)

The Graph Analysis of chat data between "Catch the Pink Flamingo" users catch the interaction and the behaviour of them. Any user from a team is allowed to create a chat session and other users from the same team can join or leave the chat room. Chat-item is when a user creates or replies to a post. These posts are linearly linked to one another as 'Response' and all these posts are of the same team chat-session. The relationships between the entities of the graph model are marked with a timestamp and the entities are identified by a unique id value.

Graph model contains 45463 nodes and 118502 edges.

4 Node Types : User, Team, TeamChatSession, ChatItem

8 Edge Types : CreateChat, CreateSession, Join, Leaves, Mentioned, OwnedBy, PartOf and ResponseTo

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	Attributes(Description)
chat_create_team_chat.csv	User, Team, TeamChatSession → ID value timeStamp(CreatesSession), timeStamp(OwnedBy) →Date/Time of creation of chatSession & ownership assignment of chatSession to team
chat_join_team_chat.csv	User, TeamChatSession → ID value timestamp(Join) → Date/time of user joining team chat-session
chat_item_team_chat.csv	User, TeamChatSession, ChatItem → ID value timeStamp(CreateChat), timeStamp(Part of) → Date/Time of creation of chatSession,
chat_leave_team_chat.csv	User, TeamChatSession → ID value timeStamp (Leaves) → Date/Time of user leaving the chat group
chat_mention_team_chat.csv	chatItem, User → ID value timestamp(Mentioned) → Date/time of user mentioning a particular post
chat_respond_team_chat.csv	chatItem, chatItem →Post1 ID value, Post2 ID value TimeStamp(ResponseTo) → Date/Time of post1 created as a response to

	post2

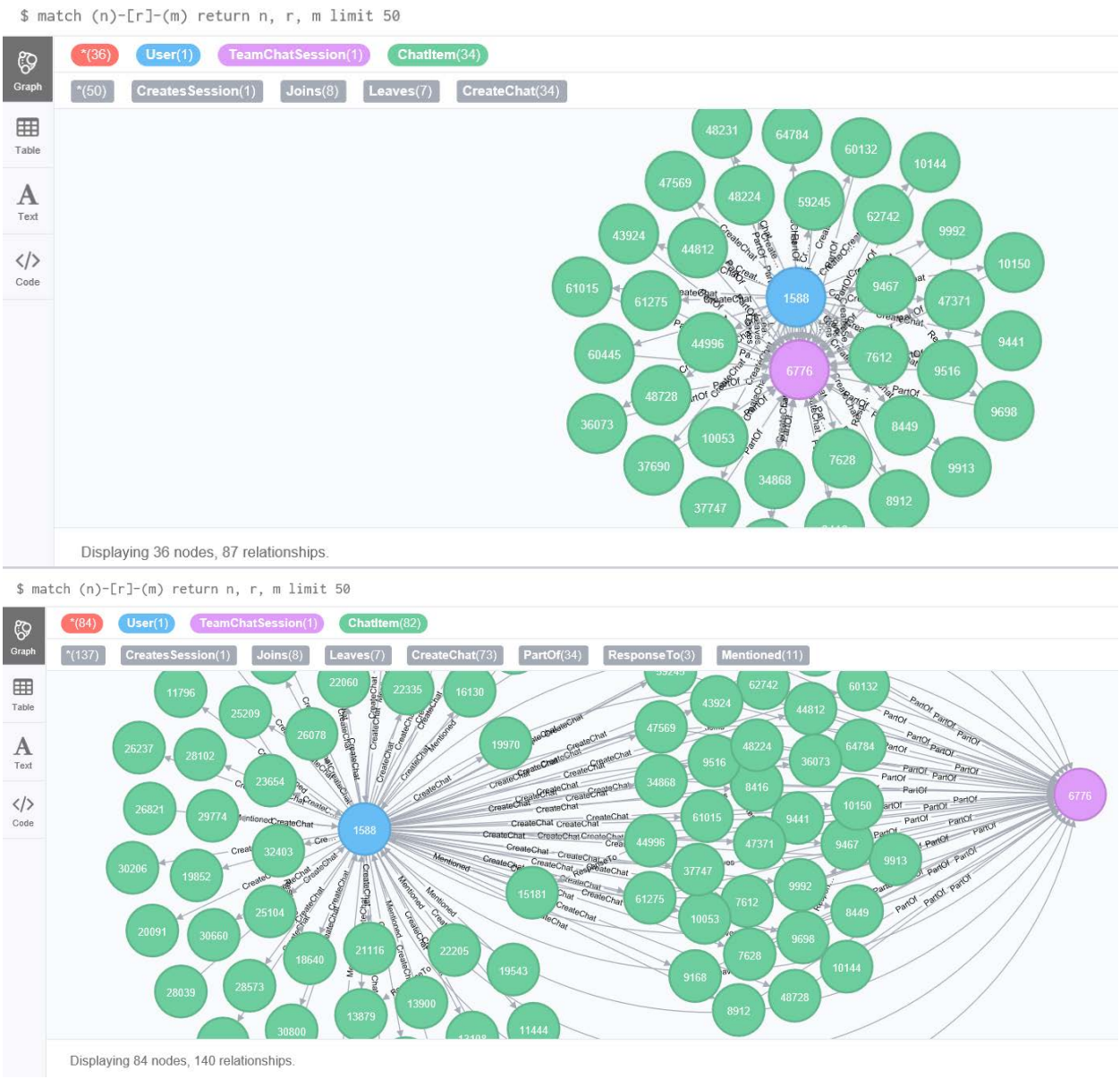
ii) Explain the loading process and include a sample LOAD command

When loading a CSV file, each row is parsed for extracting ID values to create nodes, and timestamp values to create edges. Example :

```
LOAD CSV FROM "file:///C:/Users/Sarsiz/Desktop/Capstone%20Project/Week4/chat-data/chat_item_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])})
MERGE (c:TeamChatSession {id: toInteger(row[1])})
MERGE (i:ChatItem {id: toInteger(row[2])})
MERGE (u)-[:CreateChat{timestamp: row[3]}]->(i)
MERGE (i)-[:PartOf{timestamp: row[3]}]->(c)
```

1st line loads the csv file from the specified location and parses each line as a 'row' variable.
2nd, 3rd and 4th lines create User, , TeamChatSession and ChatItem nodes, with respective ID values extracted from the 'row' tuple using index values in sequential notation.
4th and 5th lines create the CreateChat and part of edges with respective timestamp values extracted in similar manner, to link up the nodes

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.

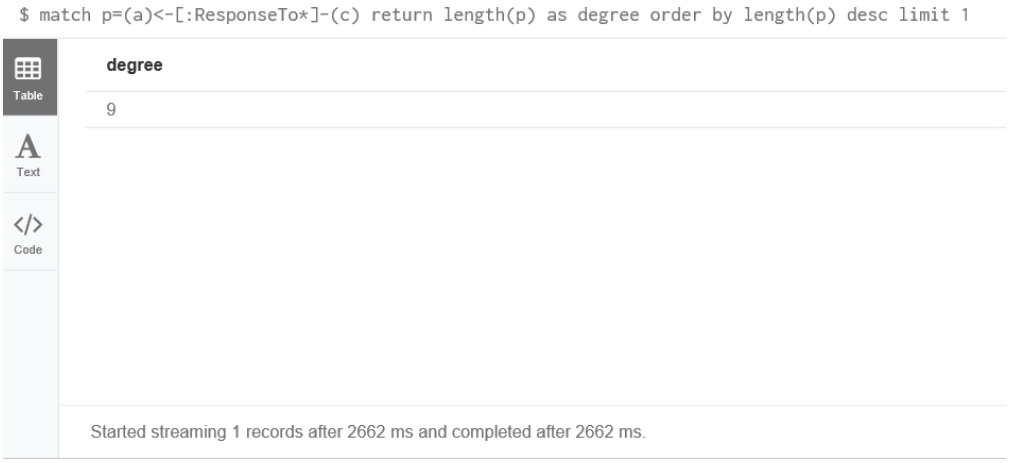


Finding the longest conversation chain and its participants

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.

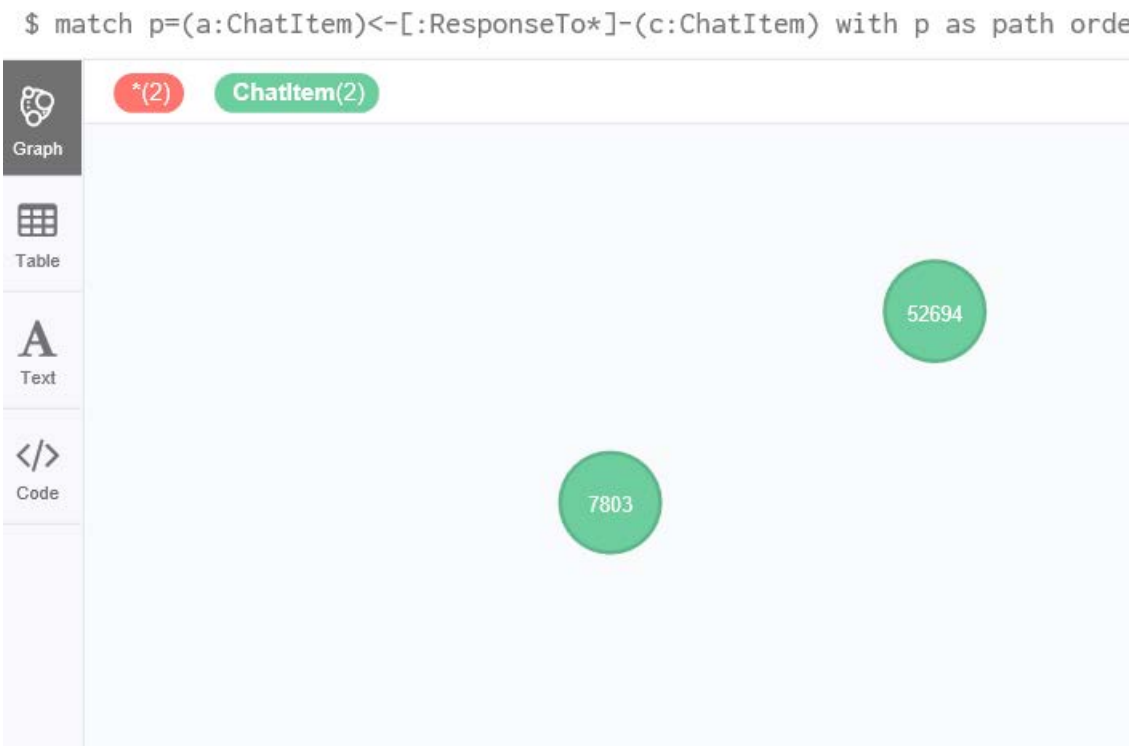
The relationship between ChatItems is “ResponseTo”. So we will use this to find the longest conversation chain. **The returned result is 9, i.e 10 chat item nodes**

```
match p=(a)<-[:ResponseTo*]-(c)
```
return length(p) as degree order by length(p) desc limit 1



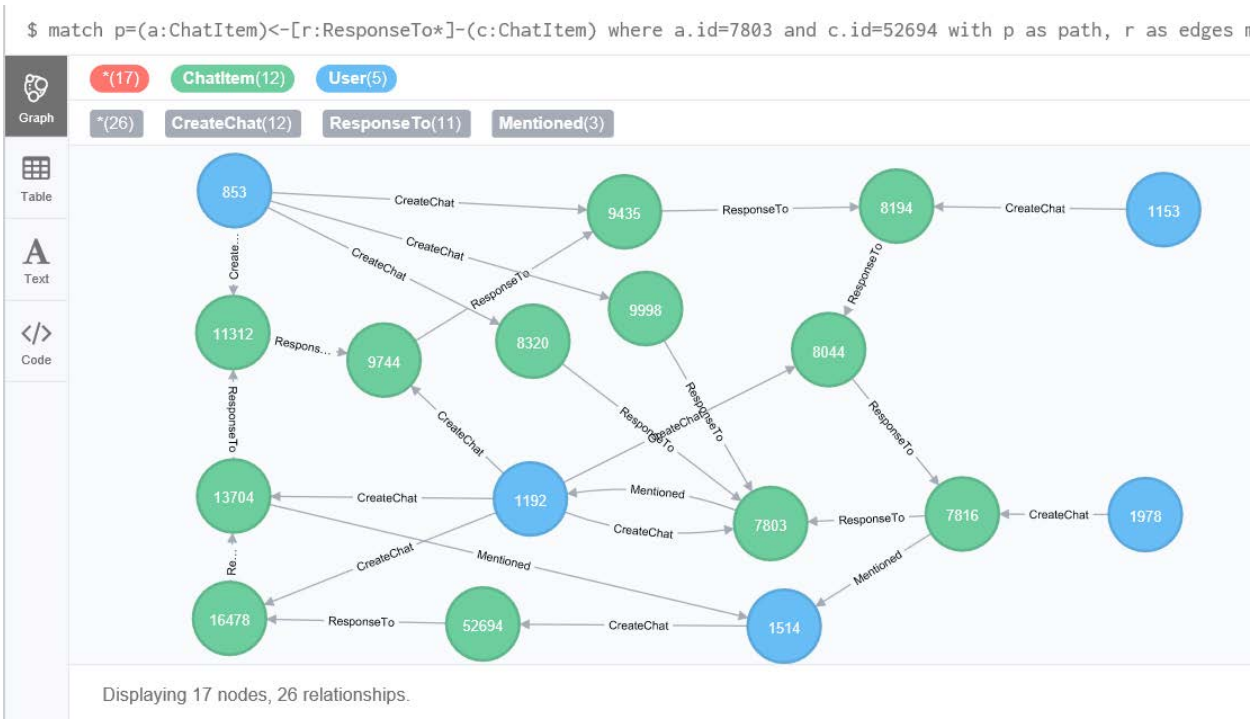
TO calculate the Number of Users Participated in the chain, I will find the first and the last node and then all the nodes between them.

To find the first and last node of the longest conversation chain. Below query finds the longest conversation chain and returns the first and last node of the chain.
match p=(a:ChatItem)<-[:ResponseTo]-(c:ChatItem)*
with p as path order by length(p) desc limit 1
return head(nodes(path)) as first, last(nodes(path)) as last
The first node has an ID of 7803 and the last node has and ID of 52694.



Used the IDs of the first and last node in the following query to find the number of users participated in this chain. Below query finds the conversation chain given the id of the first and last node of the chain, then looks for users who created each chat item.

match p=(a:ChatItem)<-[r:ResponseTo]-(c:ChatItem)*
where a.id=7803 and c.id=52694
with p as path, r as edges
match (u)-[r:CreateChat]-(n)
where n in nodes(path)
return u, r, n, edges



Number of users participating in this chain : 5

UserIDs of 5 users : 1153, 1978, 1514, 1192, 853

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.

Here I count the number of chat item created by each user, then sorts the result in descending order and return the top 10 results.
match (u:User)-[r:CreatesChat]->(:ChatItem)
return u.id as userID, count(r) as chatNum order by chatNum desc limit 10

```
$ match (u:User)-[r:CreateChat]->(:ChatItem) return u.id as userID, count(r) as chatNum order by chatNum desc limit 10
```

Table | Text | Code

userID	chatNum
394	115
2067	111
209	109
1087	109
554	107
516	105
1627	105
999	105
668	104
461	104

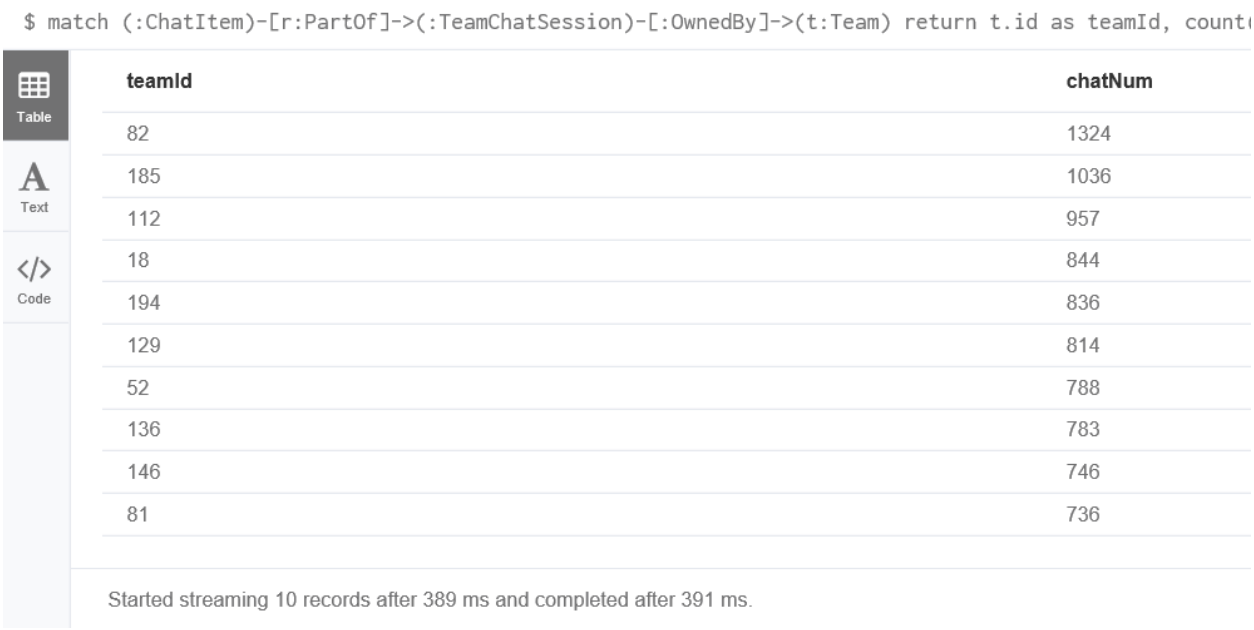
Started streaming 10 records after 828 ms and completed after 828 ms.

Chattiest Users

Users	Number of Chats
394(1st)	115
2067(2nd)	111
209(3rd)	109
1087(3rd)	109

To find the chattiest team, I applied the query :

match (:ChatItem)-[r:PartOf]->(:TeamChatSession)-[:OwnedBy]->(t:Team)
return t.id as teamId, count(r) as chatNum order by chatNum desc limit 10



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.

I find teamID of each top 10 chattiest users. Below query takes in a userID (replace XXXX by userID), and uses the Joins and OwedBy relationship to find the chat session a user have joined, then finds the team that owns the chat session

match (u:User {id:XXXX})-[:Joins]-(:TeamChatSession)-[:OwnedBy]->(t:Team)
return distinct u.id as userId, t.id as teamId

It turned out that only one of the top 10 chattiest users is from the top 10 chattiest Team. This user has a userID **of 999 (8th in top 10 chattiest user), and the team for this user has a teamID of 52.**

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.

First I will create relations between the users who interact with each other.

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

Above query is used to create relations for users who mentioned another user in chat.

Now, I will be creating relations for users who created a chatItem in response to another users' chatitem.

match (u1:User)-[:CreateChat]->(i1:ChatItem)-[:ResponseTo]->(i2:ChatItem)<-[:CreateChat]-(u2:User)
create (u1)-[:InteractsWith]->(u2)

I deleted any loop relations that connects a user to himself or herself

match (u)-[r:InteractsWith]->(u)
delete r

Now I find the clustering coefficient. Equation is *coef=numEdge/(k*(k-1)).* This will take a userID and it will find all direct neighbours and cal. k(no. of neighbours). Now I will calculate numEdge while examining the connectivity of every pair of neighbour nodes.

In the end numEdge is divided by *k*(k-1)* to get the clustering coefficient.

match (a:User {id:XXXX})-[:InteractsWith]-(c:User)
with a, collect(distinct c.id) as neighbors
match (n:User), (m:User)
where (n.id in neighbors) and (m.id in neighbors) and (n<>m)
with a, length(neighbors) as k, sum (case when (n)-[:InteractsWith]-(m) then 1 else 0 end) as numEdge
*return a.id as userId, 1.0*numEdge/(k*(k-1)) as coef*

Similarly by putting the different ids we can get the coefficients

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	1
461	1
209	0.9523809523809523
516	0.9523809523809523
554	0.904761904761904
999	0.866666666666666
1087	0.8
2067	0.785714285714285
1627	0.785714285714285
668	0.7