Karthiga Thangavelu
V00925048
CSC 501

Content:

1. Dataset:

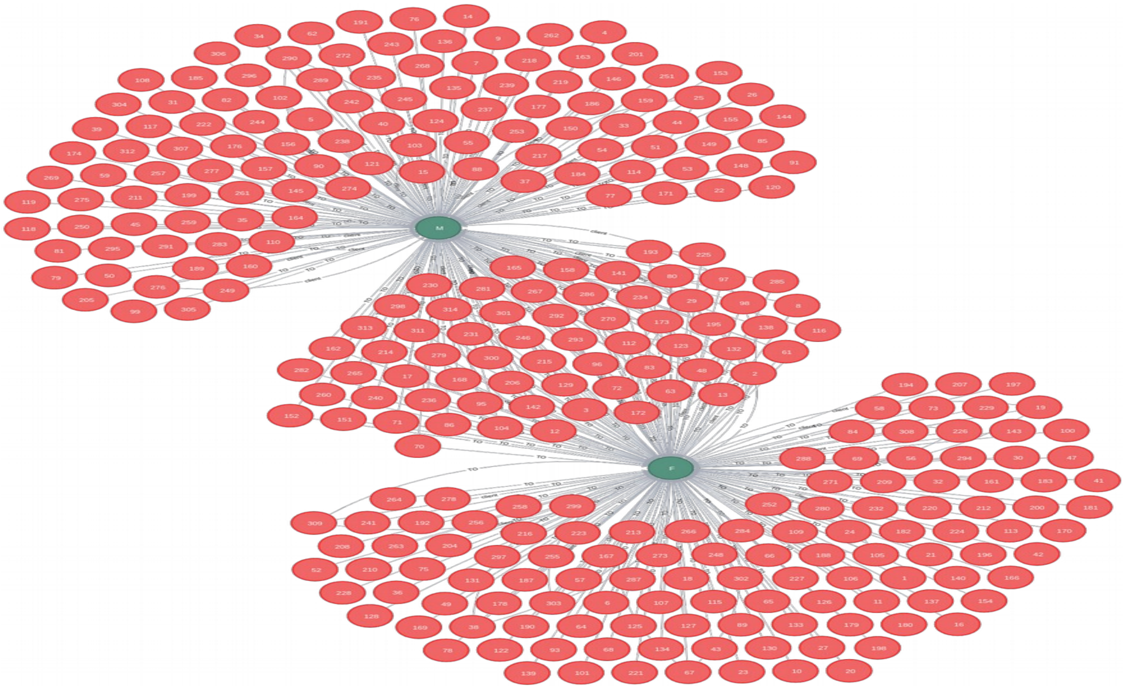
The given dataset for this assignment is the bank account details with important components account_id, client_id, a transaction of the customer in the bank. The dataset also includes loan details and card details along with the district table. The transaction table of the account which has 1056320 records. The relation between transaction table and account table is 1:N relationship where one account can have many number of transaction. In client table one account holder can have two client id one is the owner of the account and other is dispotent of the account. Dispotent are restricted for some operation in the with account like asking for loan.

For analyzing the I have used python, dataframe, and networkx. Fro visualization I have used neo4j(graph modeling) and Tableau.

2. Question 1: Analyzing dataset based on Gender:

2.1. Transaction type based on type of Characterization (Which gender did more transaction in each type)

Below is the graph which shows edges between gender and the accounts. Red node (account) and green mode (gender).



From the observation we can say that few account belong to both male and female. Some account has owner and disponent. In such cases the disponent are either male or female with different client id.

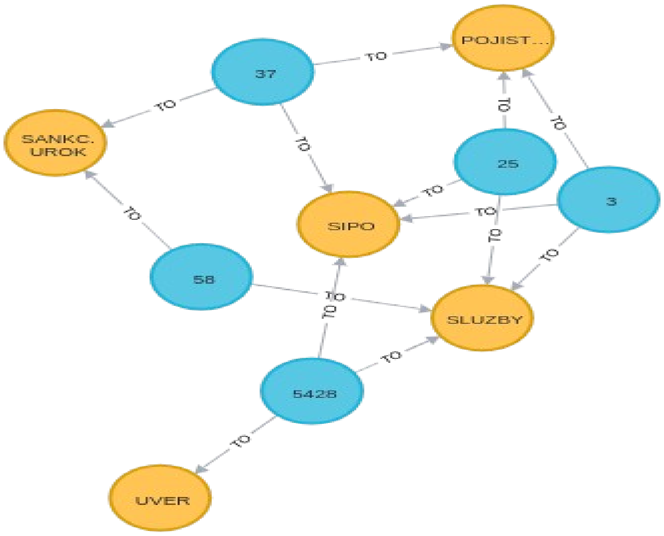
disp_id	client_id	account_id	type	birth_number	district_id	Gender
2	2	2	OWNER	450204	1	M
3	3	2	DISPONENT	406009	1	F

disp_id	client_id	account_id	type	birth_number	district_id	Gender
22	22	17	OWNER	696011	1	F
23	23	17	DISPONENT	730529	1	M

From this it is hard to say the transaction characterization of the gender. So let us assume that the transaction was done by owner and extract the data only for owner account. Now merging the disp table and trans table we can get the transaction done by each account. The null value in the column k_symbol in trans table is removed and only the record with values are taken. To find characterization of the transaction (household, loan payment, insurance payment, payment for statement) we have to consider only the amount withdrew from bank. So, here I have filtered only withdraw(VYDAJ) from trans table in column ‘type’.

Graph Modeling:

I have used entire extracted data from merging trans table and disp table with account_id for graph modeling to analyze the dataset and to find which gender transaction is higher for each characterization. For visualization (graph shown below), I used few accounts_id to show the data clearly. But for analyzing the dataset, I have used all the record extracted from tables. The given graph is a bipartite graph where several account did a transaction for same characterization. For instance, from observing the below graph we can say account_id 37, 25, and 3 has withdrew amount for paying their insurance and those account_id belong to male. This give an insight that the most of the insurance are paid by male. In this graph blue node is account_ids and yellow node is type of transaction (household, loan payment, insurance payment, etc)



graph 1.1

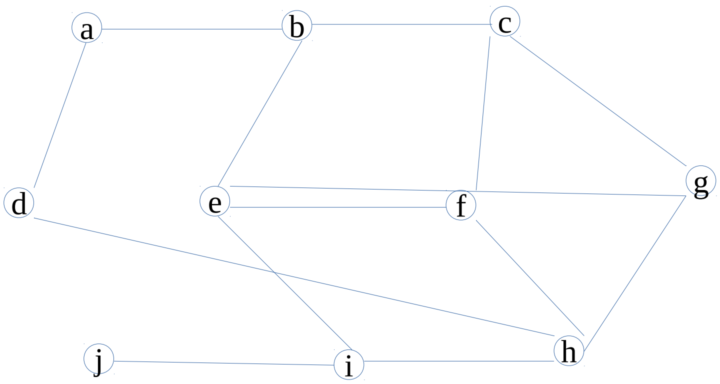
Considering source as account_id (node) and destination as k_symbol (node), we can say how many times the source node traversed to the destination node with the number of transaction. Now, for the below visualization I have taken only transaction type VYDAJ (withdrawal) for analysis from 1056320 records.

The graph 1.1 is **bipartite graph** which is defined as two disjoints and independent set, here in given graph it is accounts and characterization of transaction are two different sets. In the above graph I have filtered by male accounts which is an induced sub-graph.

For answering the 3 questions I have filtered male accounts since there are many number of records in male accounts.

Cost modeling:

Basic structural characteristic of graph are Degree Distribution, Average path length, and Clustering coefficient which together forms the small-world network. The graph can be physically represented for indicating edges between two vertex. The physical representation are adjacency list, adjacent matrix, and edge list.



The graph 1.2 is created from the same graph 1.1

Adjacency Matrix:

In adjacency matrix the index would be 1 when thee is an edge between two nodes. It is represented in matrix form.

$$\begin{array}{cccccc} & a & b & c & \dots & j \\ a & 0 & 1 & 0 & \dots & 0 \\ b & 1 & 0 & 1 & \dots & 0 \\ c & 0 & 1 & 0 & \dots & 0 \\ . & . & & & & \\ . & . & & & & \\ j & 0 & 0 & 0 & \dots & 0 \end{array}$$

This is the matrix representation for the given graph 1.1. The complexity of adjacency matrix is O(n^2).

Edge List:

Edge list is the list of edges between two nodes. The space complexity of edge list is O(E) since it contains only two or three edges. The representation of the edge list is,

[(3, 'POJISTNE'), (3, 'SLUZBY'), (3, 'SIPO'), (25, 'POJISTNE'), (25, 'SIPO'), (25, 'SLUZBY'), (37, 'SANKC. UROK'), (37, 'POJISTNE'), (37, 'SIPO'), (58, 'SANKC. UROK'), (58, 'SLUZBY'), (58, 'SANKC. UROK'), (5428, 'UVER'), (5428, 'SLUZBY'), (5428, 'SIPO')] for the given graph 1.2.

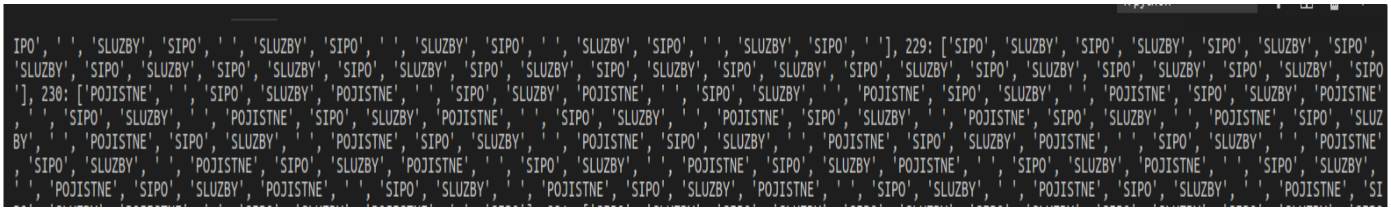
Adjacency List:

In adjacency list, the node traverse to the node connect with the edges. It stores only its neighboring nodes that are connected. The complexity of adjacency list is **O(E+V)**.

The drawback of adjacency matrix and edge list over adjacency matrix is that to find adjacent vertex we have to traverse through all the vertex. So the space complexity is high in the order of O(n^2) as it has to visit all the vertex. In case of edge list they are not efficient in space as adjacency list. Therefore here for cost analysis I have taken adjacency list. Here I have show adjacency list for given graph 1.1

{3: ['POJISTNE', 'SLUZBY', 'SIPO'], 25: ['POJISTNE', 'SIPO', 'SLUZBY'], 37: ['SANKC. UROK', 'POJISTNE', 'SIPO'], 58: ['SANKC. UROK', 'SLUZBY', 'SANKC. UROK'], 5428: ['UVER', 'SLUZBY', 'SIPO']}

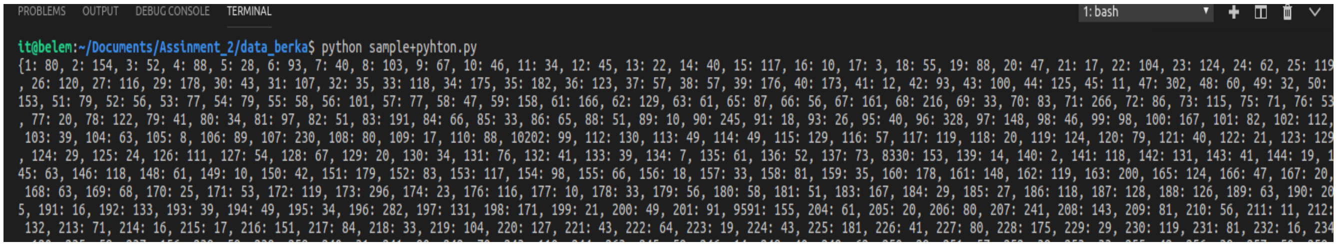
For the given graph between accounts and k_symbol the total record of dataset is 360988. The memory usage for this dataset is **44.1+ MB**. The time taken for running the adjacency list is **0.758 sec**. The adjacency list created for entire dataset is,



```
IPO', ' ', 'SLUZBY', 'SIPO', ' ', 'SLUZBY', 'SIPO', ' ', 'SLUZBY', 'SIPO', ' ', 'SLUZBY', 'SIPO', ' ', 'SLUZBY', 'SIPO', ' '], 229: ['SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO',
'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO', 'SLUZBY', 'SIPO
'], 230: ['POJISTNE', ' ', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE'
, ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZ
BY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE'
, 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY',
' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', 'POJISTNE', ' ', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SIPO', 'SLUZBY', ' ', 'POJISTNE', 'SI
```

Degree Distribution:

Degree Distribution is the number of edges belongs to a single node. Here I have shown the degree distribution from adjacency list. Below given is the degree distribution for entire dataset of 360988.



```
it@belem:~/Documents/Assinment_2/data_berka$ python sample+pyhton.py
{1: 80, 2: 154, 3: 52, 4: 88, 5: 28, 6: 93, 7: 40, 8: 103, 9: 67, 10: 46, 11: 34, 12: 45, 13: 22, 14: 40, 15: 117, 16: 10, 17: 3, 18: 55, 19: 88, 20: 47, 21: 17, 22: 104, 23: 124, 24: 62, 25: 119
, 26: 120, 27: 116, 29: 178, 30: 43, 31: 107, 32: 35, 33: 118, 34: 175, 35: 182, 36: 123, 37: 57, 38: 57, 39: 176, 40: 173, 41: 12, 42: 93, 43: 100, 44: 125, 45: 11, 47: 302, 48: 60, 49: 32, 50:
153, 51: 79, 52: 56, 53: 77, 54: 79, 55: 58, 56: 101, 57: 77, 58: 47, 59: 158, 61: 166, 62: 129, 63: 61, 65: 87, 66: 56, 67: 161, 68: 216, 69: 33, 70: 83, 71: 266, 72: 86, 73: 115, 75: 71, 76: 53
, 77: 20, 78: 122, 79: 41, 80: 34, 81: 97, 82: 51, 83: 191, 84: 66, 85: 33, 86: 65, 88: 51, 89: 10, 90: 245, 91: 18, 93: 26, 95: 40, 96: 328, 97: 148, 98: 46, 99: 98, 100: 167, 101: 82, 102: 112,
 103: 39, 104: 63, 105: 8, 106: 89, 107: 230, 108: 80, 109: 17, 110: 88, 10202: 99, 112: 130, 113: 49, 114: 49, 115: 129, 116: 57, 117: 119, 118: 20, 119: 124, 120: 79, 121: 40, 122: 21, 123: 129
, 124: 29, 125: 24, 126: 111, 127: 54, 128: 67, 129: 20, 130: 34, 131: 76, 132: 41, 133: 39, 134: 7, 135: 61, 136: 52, 137: 73, 8330: 153, 139: 14, 140: 2, 141: 118, 142: 131, 143: 41, 144: 19, 1
45: 63, 146: 118, 148: 61, 149: 10, 150: 42, 151: 179, 152: 83, 153: 117, 154: 98, 155: 66, 156: 18, 157: 33, 158: 81, 159: 35, 160: 178, 161: 148, 162: 119, 163: 200, 165: 124, 166: 47, 167: 20,
 168: 63, 169: 68, 170: 25, 171: 53, 172: 119, 173: 296, 174: 23, 176: 116, 177: 10, 178: 33, 179: 56, 180: 58, 181: 51, 183: 167, 184: 29, 185: 27, 186: 118, 187: 128, 188: 126, 189: 63, 190: 20
5, 191: 16, 192: 133, 193: 39, 194: 49, 195: 34, 196: 282, 197: 131, 198: 171, 199: 21, 200: 49, 201: 91, 9591: 155, 204: 61, 205: 20, 206: 80, 207: 241, 208: 143, 209: 81, 210: 56, 211: 11, 212:
 132, 213: 71, 214: 16, 215: 17, 216: 151, 217: 84, 218: 33, 219: 104, 220: 127, 221: 43, 222: 64, 223: 19, 224: 43, 225: 181, 226: 41, 227: 80, 228: 175, 229: 29, 230: 119, 231: 81, 232: 16, 234
```

Clustering Coefficient:

Clustering coefficient is the measure of transitive closure which is give as number of triangles divided by tirades. It is given as, CC = 3*number of triangles in graph / (3*number of triangles) + trades.

For calculating Clustering Coefficient for the graph between account and k_symbol I have used networkx function for clustering. The clustering coefficient for entire dataset is **2.432089**
.

APL (Average path length):

Shortest path length is the minimum path distance between source and the target node. Average path length is the average of all pairs of shortest path which is calculated as,

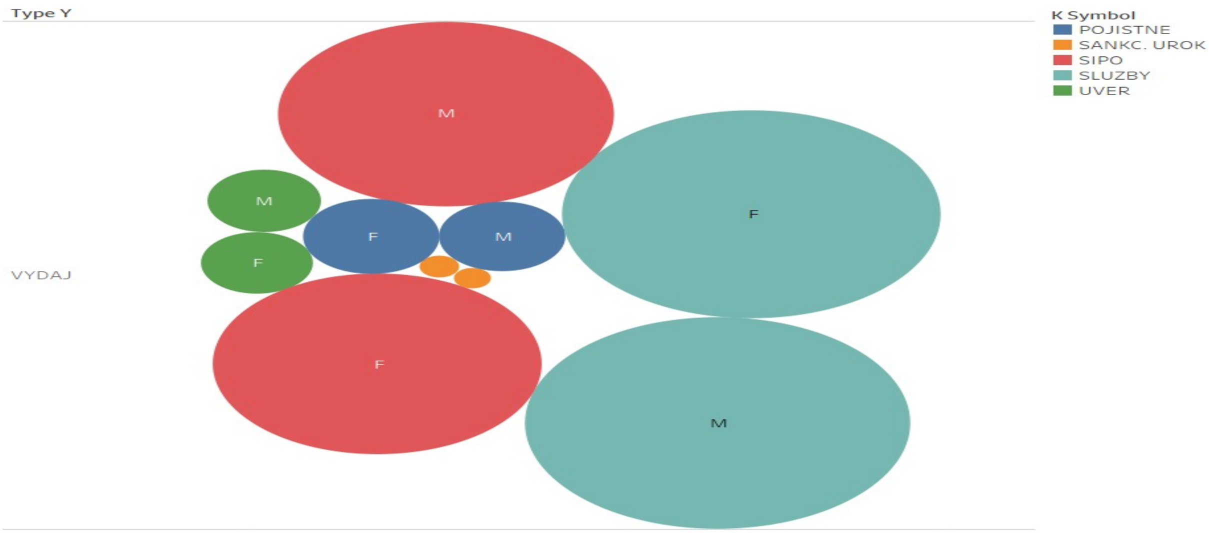
$APL = 1/(n(n-1) \sum \sum d(s,t)$, where s is the source node and t is the target node

For unweighted graph BFS (breath first search) algorithm is used for find shortest path and for weighted graph dijkstra’s algorithm is used to find shortest path.

Here, I have taken small graph shown in graph 1.1 to calculate average path length using breadth first search algorithm. In this few nodes doesn’t has any outwards edges and few node has more than three outward edges.

Insight:

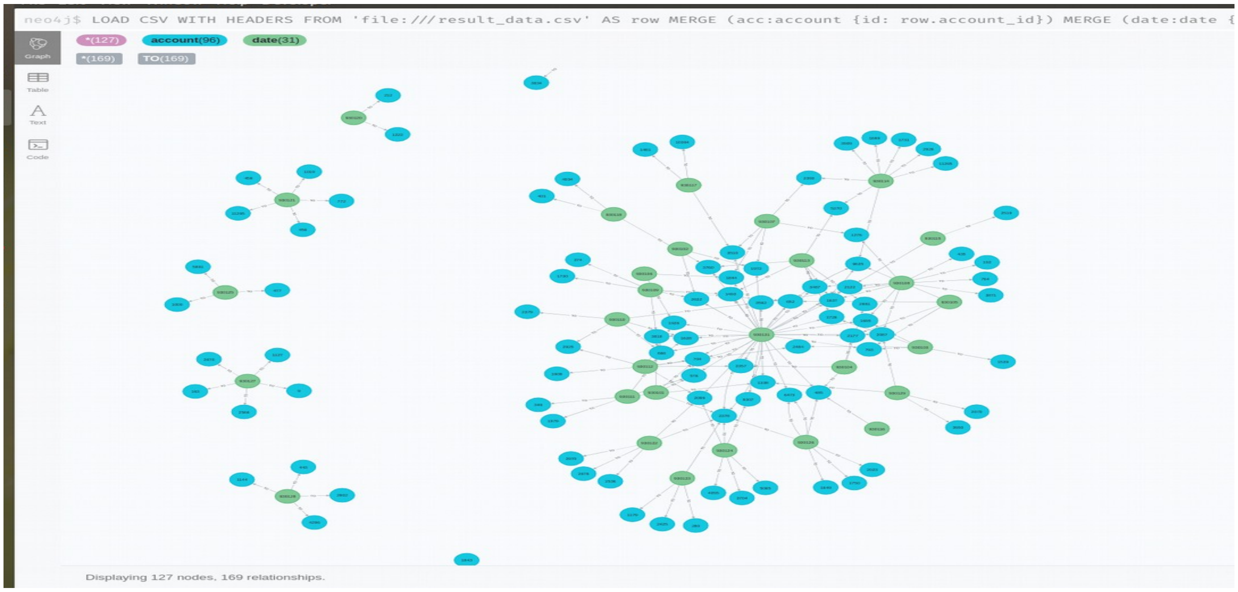
The insight from the observation of the visualization given below is that the household (SIPO) transactions are mostly made by male which is 60,279 transaction and the transaction made by female are 57,786 transactions. Whereas the insurance transaction are mostly made by female than man. For loan payment the male account transaction are higher than female.



2.2. Busiest week of transaction (when most of the two type (household and loan payment) transaction happened):

Temporal Analysis in graph modeling:

The second part of the question is to analyzes when most of the transaction are happening in a month whether it is month end or beginning of the month. The first graph shows the edges between date (green nodes) and account_id (blue nodes) which is bipartite graph. From observing the graph we can tell that two or more account did transaction on same date. For graph visualization neo4j has taken only less records.



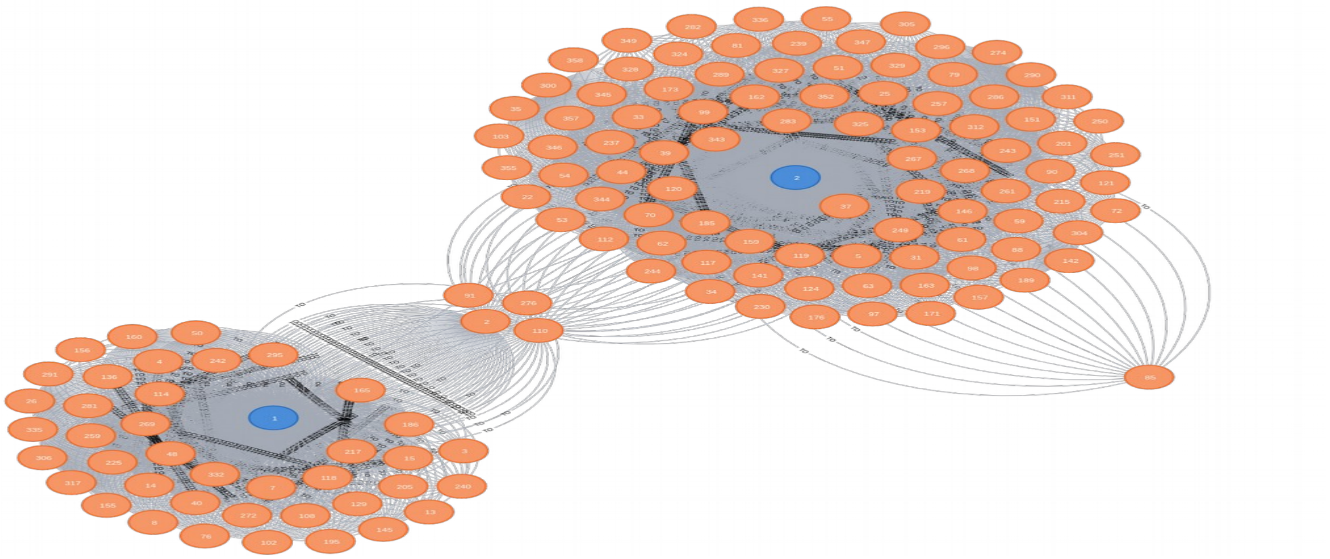
```
neo4j$ LOAD CSV WITH HEADERS FROM 'file:///result_data.csv' AS row MERGE (acc:account {id: row.account_id}) MERGE (date:date {
```

Displaying 127 nodes, 169 relationships.

Further analyzing the graph, I have taken the transaction done in each week. Here I have considered that male accounts who are the owner of the account doing the transaction . Further I have considered only household and loan payment by different accounts with mode of transaction withdrawal (YVDAJ).

Insight:

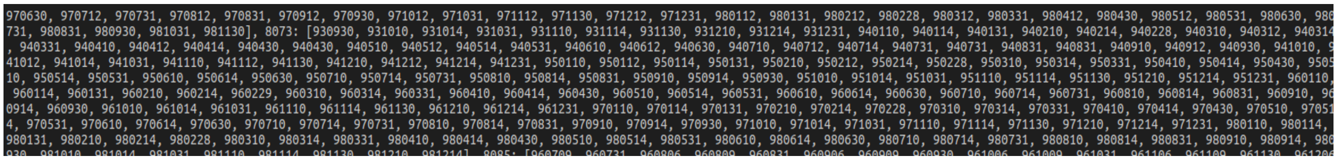
The insight of the graph is that first week of the month is more dense and more transaction happens for household and loan payment since most account holder receive their salary. The edges between these account nodes are the type of payment and first week of the month which connects the accounts.



Cost Analysis:

Adjacency list:

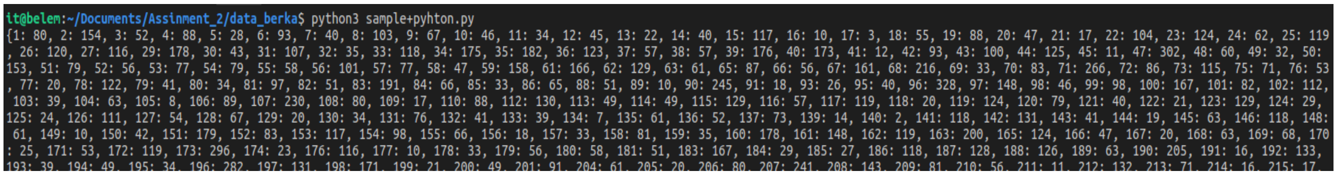
Below given is the adjacency list for nodes account and date of the transaction.



```
970630, 970712, 970731, 970812, 970831, 970912, 970930, 971012, 971031, 971112, 971130, 971212, 971231, 980112, 980131, 980212, 980228, 980312, 980331, 980412, 980430, 980512, 980531, 980630, 980
731, 980831, 980930, 981031, 981130], 8073: [930930, 931010, 931014, 931031, 931110, 931114, 931130, 931210, 931214, 931231, 940110, 940114, 940131, 940210, 940214, 940228, 940310, 940312, 940314
, 940331, 940410, 940412, 940414, 940430, 940430, 940510, 940512, 940514, 940531, 940610, 940612, 940630, 940710, 940712, 940714, 940731, 940731, 940831, 940831, 940910, 940912, 940930, 941010, 9
41012, 941014, 941031, 941110, 941112, 941130, 941210, 941212, 941214, 941231, 950110, 950112, 950114, 950131, 950210, 950212, 950214, 950228, 950310, 950314, 950331, 950410, 950414, 950430, 9505
10, 950514, 950531, 950610, 950614, 950630, 950710, 950714, 950731, 950810, 950814, 950831, 950910, 950914, 950930, 951010, 951014, 951031, 951110, 951114, 951130, 951210, 951214, 951231, 960110,
 960114, 960131, 960210, 960214, 960229, 960310, 960314, 960331, 960410, 960414, 960430, 960510, 960514, 960531, 960610, 960614, 960630, 960710, 960714, 960731, 960810, 960814, 960831, 960910, 96
0914, 960930, 961010, 961014, 961031, 961110, 961114, 961130, 961210, 961214, 961231, 970110, 970114, 970131, 970210, 970214, 970228, 970310, 970314, 970331, 970410, 970414, 970430, 970510, 97051
4, 970531, 970610, 970614, 970630, 970710, 970714, 970731, 970810, 970814, 970831, 970910, 970914, 970930, 971010, 971014, 971031, 971110, 971114, 971130, 971210, 971214, 971231, 980110, 980114,
980131, 980210, 980214, 980228, 980310, 980314, 980331, 980410, 980414, 980430, 980510, 980514, 980531, 980610, 980614, 980630, 980710, 980714, 980731, 980810, 980814, 980831, 980910, 980914, 980
930, 981010, 981014, 981031, 981110, 981114, 981130, 981210, 981214], 8085: [960709, 960731, 960806, 960809, 960831, 960906, 960909, 960930, 961006, 961009, 961031, 961106, 961109, 961130, 96120
```

Degree Distribution:

Below given image is the degree distribution from the adjacency list.



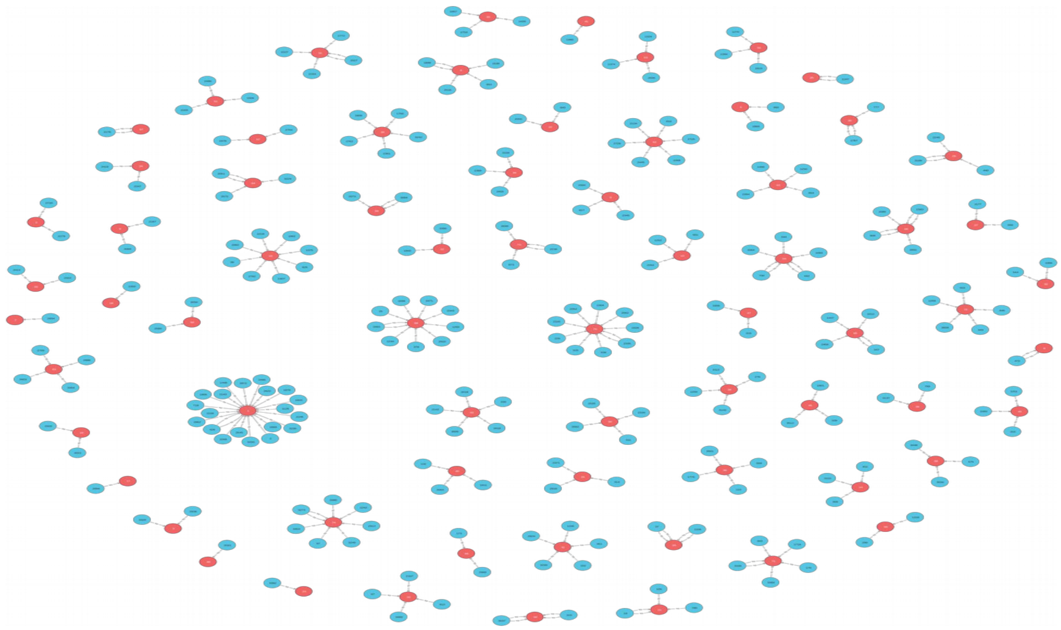
```
it@belem:~/Documents/Assinment_2/data_berka$ python3 sample+pyhton.py
{1: 80, 2: 154, 3: 52, 4: 88, 5: 28, 6: 93, 7: 40, 8: 103, 9: 67, 10: 46, 11: 34, 12: 45, 13: 22, 14: 40, 15: 117, 16: 10, 17: 3, 18: 55, 19: 88, 20: 47, 21: 17, 22: 104, 23: 124, 24: 62, 25: 119
, 26: 120, 27: 116, 29: 178, 30: 43, 31: 107, 32: 35, 33: 118, 34: 175, 35: 182, 36: 123, 37: 57, 38: 57, 39: 176, 40: 173, 41: 12, 42: 93, 43: 100, 44: 125, 45: 11, 47: 302, 48: 60, 49: 32, 50:
153, 51: 79, 52: 56, 53: 77, 54: 79, 55: 58, 56: 101, 57: 77, 58: 47, 59: 158, 61: 166, 62: 129, 63: 61, 65: 87, 66: 56, 67: 161, 68: 216, 69: 33, 70: 83, 71: 266, 72: 86, 73: 115, 75: 71, 76: 53
, 77: 20, 78: 122, 79: 41, 80: 34, 81: 97, 82: 51, 83: 191, 84: 66, 85: 33, 86: 65, 88: 51, 89: 10, 90: 245, 91: 18, 93: 26, 95: 40, 96: 328, 97: 148, 98: 46, 99: 98, 100: 167, 101: 82, 102: 112,
 103: 39, 104: 63, 105: 8, 106: 89, 107: 230, 108: 80, 109: 17, 110: 88, 112: 130, 113: 49, 114: 49, 115: 129, 116: 57, 117: 119, 118: 20, 119: 124, 120: 79, 121: 40, 122: 21, 123: 129, 124: 29,
125: 24, 126: 111, 127: 54, 128: 67, 129: 20, 130: 34, 131: 76, 132: 41, 133: 39, 134: 7, 135: 61, 136: 52, 137: 73, 139: 14, 140: 2, 141: 118, 142: 131, 143: 41, 144: 19, 145: 63, 146: 118, 148:
 61, 149: 10, 150: 42, 151: 179, 152: 83, 153: 117, 154: 98, 155: 66, 156: 18, 157: 33, 158: 81, 159: 35, 160: 178, 161: 148, 162: 119, 163: 200, 165: 124, 166: 47, 167: 20, 168: 63, 169: 68, 170
: 25, 171: 53, 172: 119, 173: 296, 174: 23, 176: 116, 177: 10, 178: 33, 179: 56, 180: 58, 181: 51, 183: 167, 184: 29, 185: 27, 186: 118, 187: 128, 188: 126, 189: 63, 190: 205, 191: 16, 192: 133,
193: 39, 194: 49, 195: 34, 196: 282, 197: 131, 198: 171, 199: 21, 200: 49, 201: 91, 204: 61, 205: 20, 206: 80, 207: 241, 208: 143, 209: 81, 210: 56, 211: 11, 212: 132, 213: 71, 214: 16, 215: 17,
```

The time taken for computing degree distribution and adjacency list is **0.658 sec**.

3. Question 2: Analyzing loan payment data

Which district people belongs to accounts who have taken loan and to say whether they are in debt and not payed the loan

The below graph shows the loan taken by the account (blue nodes) and which district (red nodes) they belongs to. From the graph we can observe that district 1 is the densely populated graph. The district 1 in the graph is Hl.m. Praha, Prague whose population is high comparing to other cities in district. From the observation it is known that for few records the client district id is not same as the bank district id. So, here I have taken only the account with bank district and client district. For graph visualization the neo4j has taken only less records.

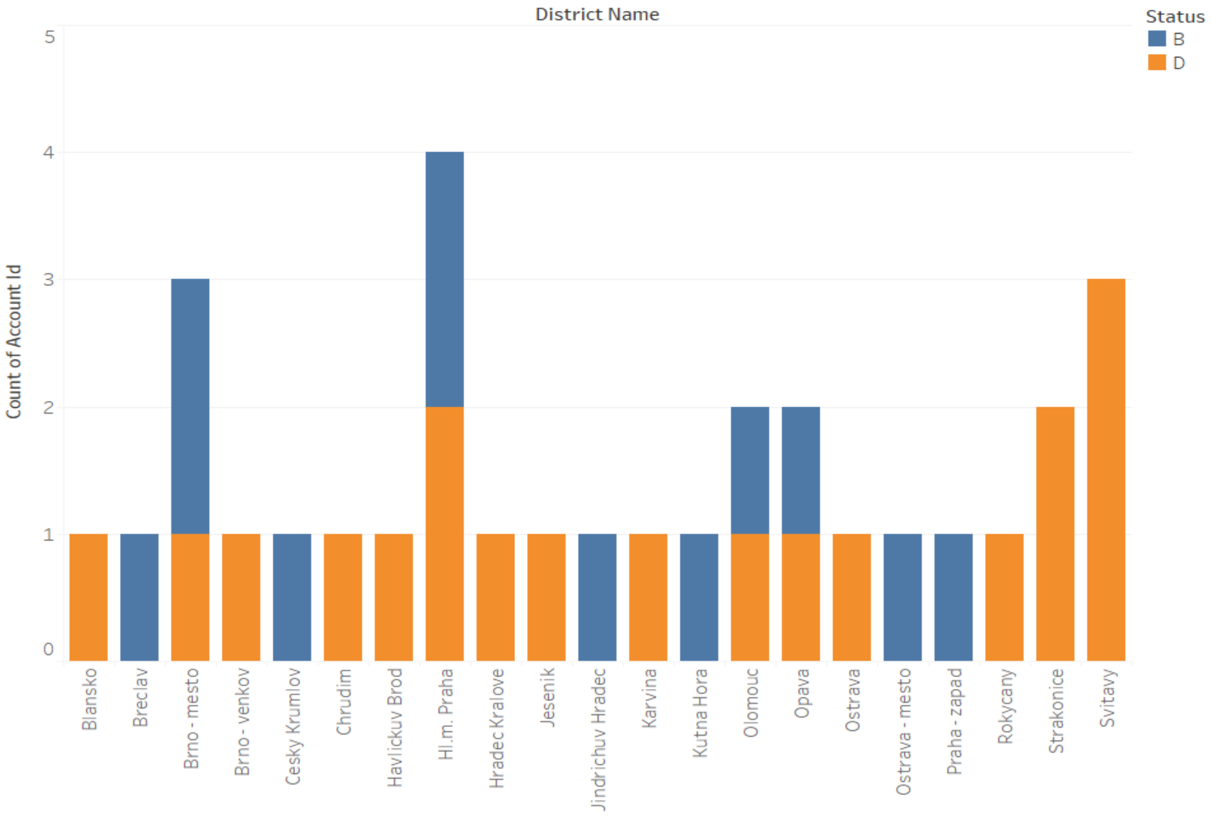


Spatial analysis:

For analyzing which district has more debt and not payed loan account, I have taken only the male account since the male account has more transaction for loan payment than female account. Since the loan table and loan order in order table are not having the same number of account who has taken loan I have joined loan table and order table using left join where order table has more number of loan account than loan table. So, here I have taken all the loan account details. After which I have filtered the account who has taken loan. From which further filtered the status of the account who has not payed loan and in debt for current running contract.

Insight:

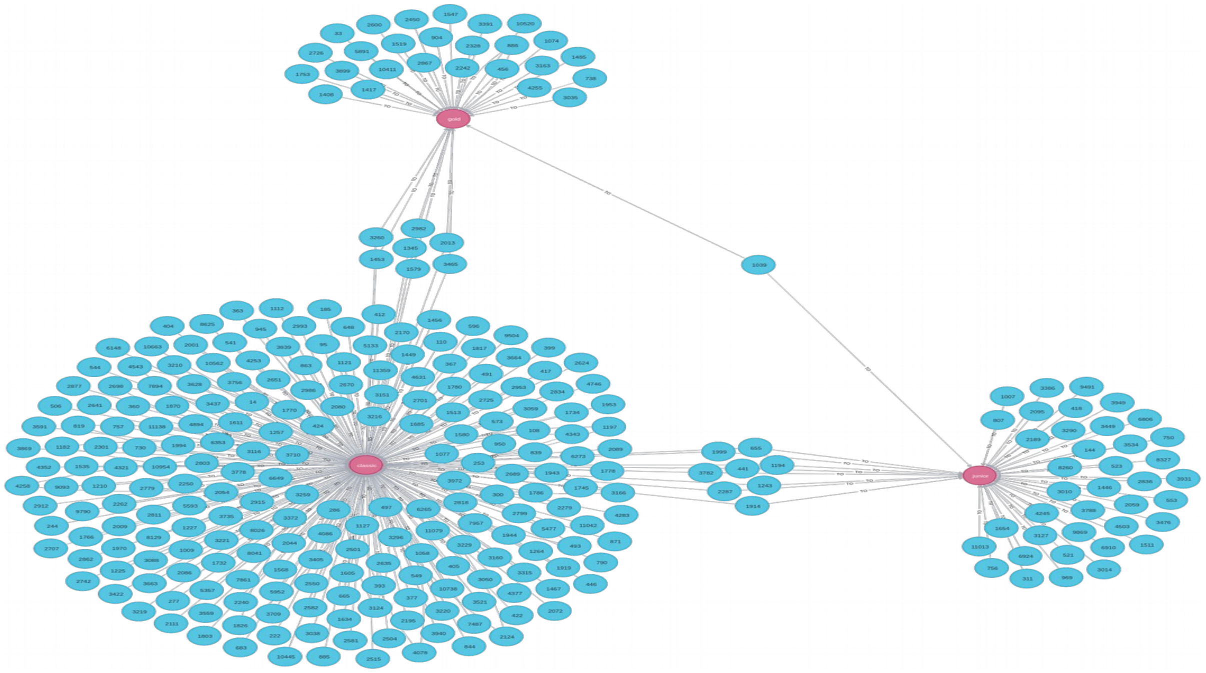
From below visualization we can say that there are more in debt and not paid loan account users in district **Hl.m. Praha.** The status B indicated contract finished but loan not payed and status D indicates it is current contract and they are still in debt. Comparing the highest not paying loan properly account with other data in district dataset we can see that Hl.m. Praha has highest population and their unemployment rate is less for year 95 and 96 comparing to other district. Since they have large population, the status D and B are high in that district.



4. **Question3 - Analyzing credit card data**

Suggesting up-gradation to the customer’s credit card based on their previous type of credit card or new credit card) :

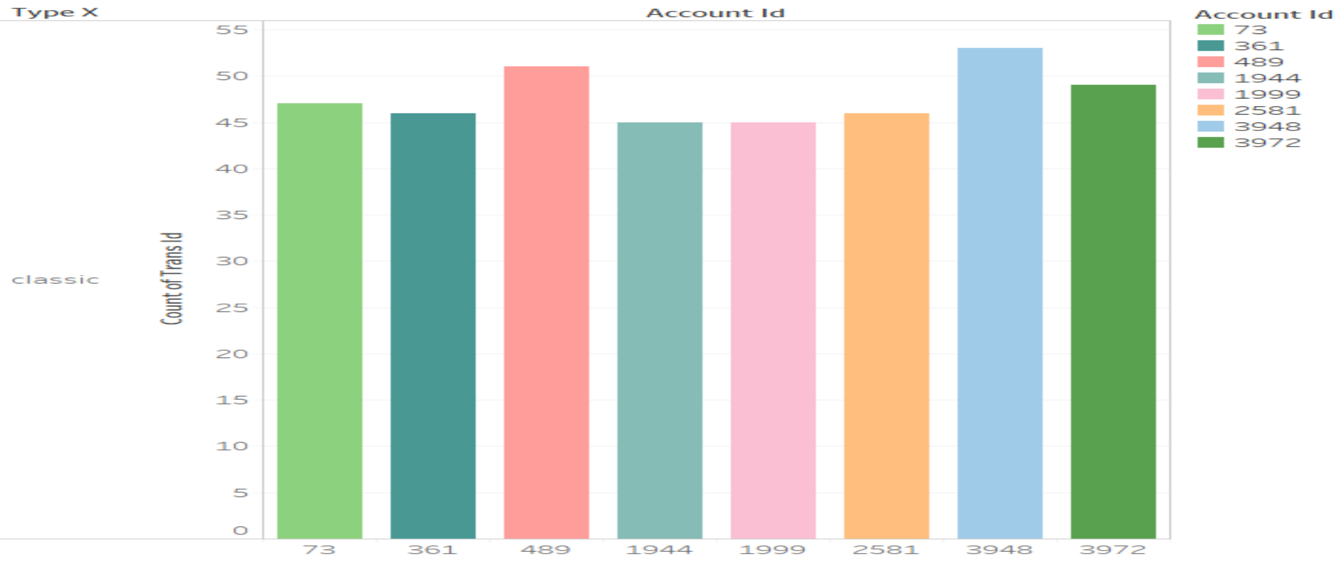
In the graph below I have shown the connections between credit card type (junior, classic, and gold). From the observation we can see that account_id holding classic credit card looks dense than gold and junior. There are also accounts that holds both classic and gold.



For suggesting the card I have taken only male account_ids from questions 1 and taken type classic. There are two parts for this question one is to suggest account holder to upgrade from classic to gold and other is to suggest account holder to get new credit card based on their mode and frequency of their transaction. For which I have taken card table and joined it with disp table. Then join the merged table with client table and extracted only male accounts who are in classic and has done credit card transaction so far. Filtering out those details, set the condition for upgrading the credit card from classic to gold as the number of transaction should be greater that 45. The account_id which satisfy this condition would be suggested for up-gradation.

Insight:

The visualization is show below for account whose credit card transaction so far is greater than 45. The insight from this visualization is that inspite of dense graph for the type classic, the eligible account are only 8 to upgrade to gold.

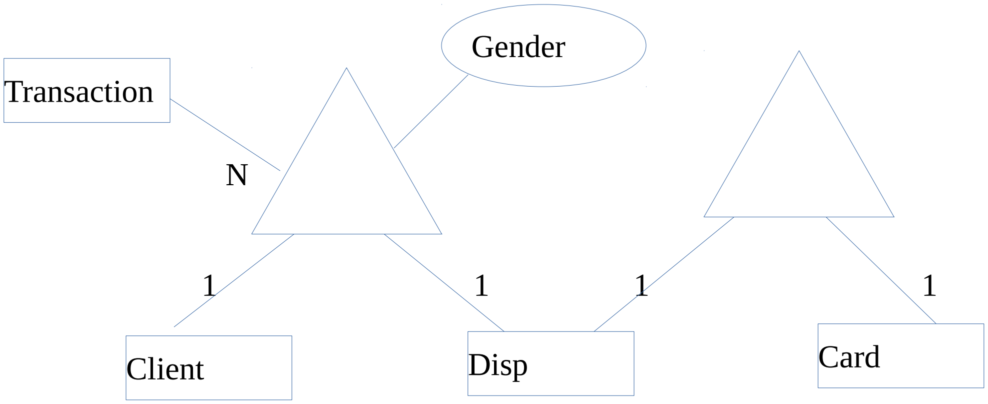


Relational Modeling:

The second part is to suggest new card for account who don’t have credit card based on their account operation. The suggestion is based on how frequently the account has been used, purpose, and mode of the transaction. Filtering out all the attribute and setting the condition for number of the transaction as 60 and greater than 60 the account which is suitable for suggestion can be taken out. Here I have joined the card detail and disp details using outer join since we have to filter the transaction who don’t have credit card. From the above visualization we can observe that these are the account can be suggested new card which are taken based on their frequency of their transaction.

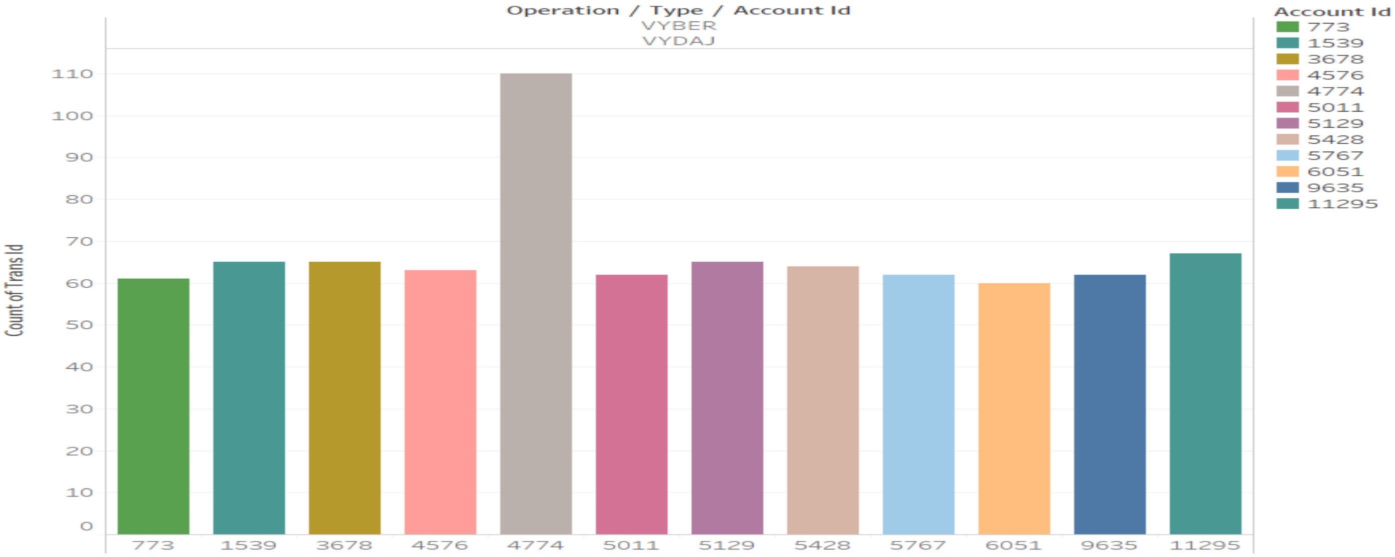
```
fre_trans = pd.merge(card_details, disp_details, how='outer', on='disp_id')
```

In outer join the data in table and and table b both are considered. Here the data in entire card details and disp details are considered. The hash join algorithm which supports the full outer joins. From the below representation the relationship between client and transaction table is 1:N, client and disp table is 1:1 and disp and card table is 1:1.



Insight:

From the visualization we can say that the transaction more than 60 are considered for suggesting new credit card based on their withdrawal of amount.


Operation / Type / Account Id
VYBER
VYDAJ
Count of Trans Id
Account Id