# PSG COLLEGE OF TECHNOLOGY, COIMBATORE 641004 Department of Computer Science and Engineering 19Z002 - SOCIAL AND ECONOMIC NETWORK ANALYSIS



# PROJECT REPORT

Topic: Merging multi modal customer service tickets

# **Team members**

19z209 - Deepika S

19z212 - G Solaisneha

19z222 - Kaniska Varsini P D

19z231 - Pavithra L

19z233 - Pranitha V S

19z252 - Swetha G

# **Table of Contents**

1.	Problem statement.	2
2.	Dataset description.	.2
3.	Tools used.	2
4.	Challenges faced	3
5.	Contribution of team members.	3
6.	Annexure I : Code	4
7.	Annexure II : Snapshots of the outputs	6
8.	References	9

### **Problem Statement**

Solving customer issues is now a part of any organization offering its products or services. Customers reach out to companies by opening a ticket with their email or phone number and describing their issue. They contact the customer service via multiple channels. Oftentimes, customers reach out to customer service with the same issue, but under different email ids, phone numbers and accounts. Merging such tickets makes it efficient to address them correctly and in least amount of time. Our problem statement is to merge relevant tickets of a user to efficiently handle customer issues.

### **Analysis**

- 1. Merging tickets with the same issues to avoid addressing duplicate issues with the help of orderID, email, phoneNo. (A user may have more than one email, phoneNo and orders)
- 2. Analysis on how many times a user has contacted customer service for issue and find users who have raised higher number of issues
- 3. Analysis on the number of unique orders a user has made to find the super users and prioritize resolving their issues.

# **Dataset Description**

Contacts is the dataset used. It contains 5,00,000 instances, with the following five features:

- Id uniquely identifies a ticket
- Email email id with which the ticket was initiated
- Phone phone number with which the ticket was initiated
- Contacts Number of times user reached out in the particular ticket
- OrderId uniquely identifies an transaction

Dataset link: https://www.kaggle.com/datasets/eugenevoon/multichannel-contacts-problem

### **Tools used**

### Gephi:

Gephi is a visualization application developed in the Java language. It is mainly used for visualizing, manipulating, and exploring networks. The user interacts with the representation, manipulates the structures, shapes, and colors to reveal hidden patterns.

### **Python Packages used:**

### 1. Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

### 2. Networkx

NetworkX is a package for the Python programming language that's used to create, manipulate, and study the structure, dynamics, and functions of complex graph networks

# **Challenges faced:**

- As the dataset is huge(50 Mb file with 5 lakh instances), it was difficult to handle the data
- Importing the dataset and running visualization algorithms on Gephi was difficult given the size of the dataset
- Extracting the required nodes from the constructed graph was challenging

# **Contributions:**

NAME	ROLLNO	CONTRIBUTION
Deepika S	19z209	Graph creation
G Solai Sneha	19z212	Analysis on how many times a user has contacted customer service for issue and find users who have raised higher number of issues
Kaniska Varshini PD	19z222	Node extraction from connected components
Pavithra L	19z231	Merging tickets with the same issues to avoid addressing duplicate issues with the help of orderID, email, phoneNo
Pranitha V S	19z233	Dataset and Gephi visualization

Swetha G	19z252	Analysis on the number of unique orders a user has made to find the super users and prioritize resolving their issues.
----------	--------	--

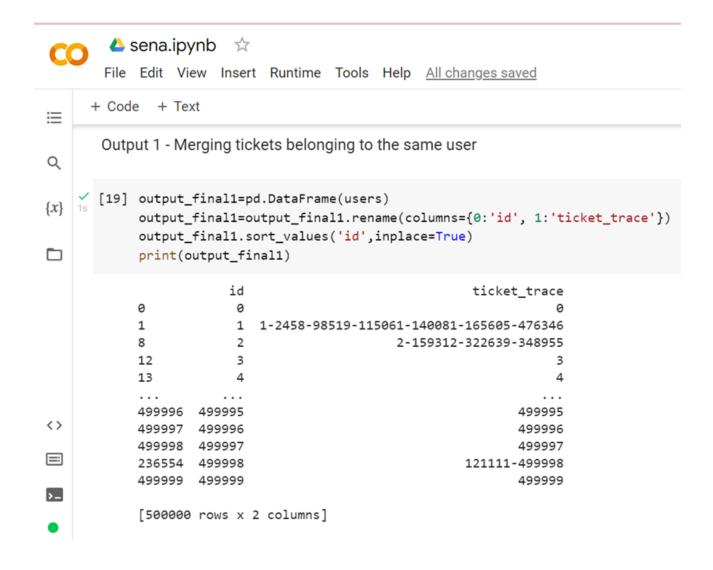
### **Annexure I: Code**

```
import pandas as pd
import networkx as nx
#Reading data
df=pd.read json('/contacts.json')
df.Email=df.Email.apply(lambda x: "Email "+x if x !=" else ")
df.Phone=df.Phone.apply(lambda x: "Phone "+x if x !=" else ")
df.OrderId=df.OrderId.apply(lambda x: "OrderId "+x if x !=" else ")
#Create graph G with edges id -> email, id -> phone, id -> orderId
nodes=[]
for ,Id, , ,Contacts, in df.itertuples():
  nodes.append((Id,{"Contacts": Contacts}))
G=nx.Graph()
G.add nodes from(nodes)
G.add edges from(df[df.Email!="][['Id', 'Email']].to records(index=False))
G.add_edges_from(df[df.Phone != "][['Id', 'Phone']].to_records(index=False))
G.add_edges_from(df[df.OrderId != "][['Id', 'OrderId']].to_records(index=False))
print(G.number of nodes(), G.number of edges())
#Finding connected components in Graph G
conn comp=list(nx.connected components(G))
#Analysis
users=[]
contact sum=[]
unique orders=[]
# for each connected component we appended the ticket id within
for each connected component in conn comp:
  id list=[]
```

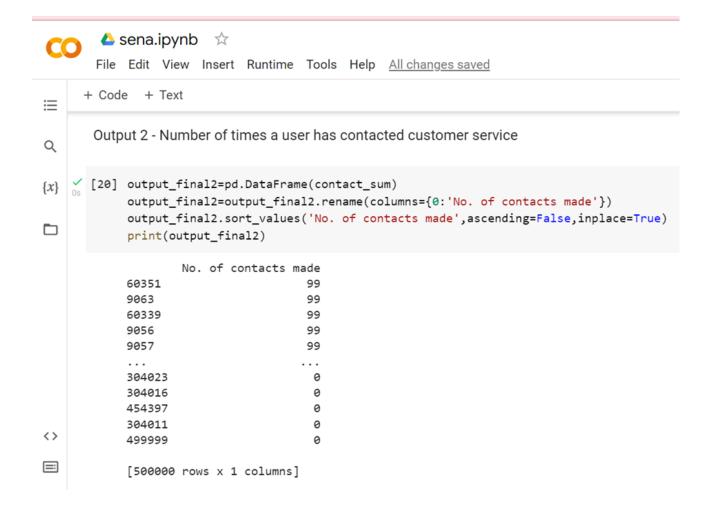
```
order list=[]
  for each node in each connected component:
    # check if the node is a number, append to the id list
     if str(each node).isnumeric():
       id list.append(each node)
#check if the node is an OrderId, append to the order list
    if str(each node).startswith('OrderId'):
       order list.append(each node)
  # sum of contacts that a node has made
  sum of contacts=0
  for order id in id list:
     sum of contacts += G.nodes[order id]['Contacts']
  output str1='-'.join([str(each node) for each node in sorted(id list)])
  output str2=','.join([str(each node) for each node in sorted(order list)])
  for id in id list:
     users.append([id, output str1])
    contact sum.append(sum of contacts)
  for order id in id list:
     unique orders.append([order id,output str2,len(order list)])
#Output 1 - Merging tickets belonging to the same user
output final1=pd.DataFrame(users)
output final1=output final1.rename(columns={0:'id', 1:'ticket trace'})
output final1.sort values('id',inplace=True)
print(output final1)
#Output 2 - Number of times a user has contacted customer service
output final2=pd.DataFrame(contact sum)
output final2=output final2.rename(columns={0:'No. of contacts made'})
output final2.sort values('No. of contacts made',ascending=False,inplace=True)
print(output final2)
#Output 3 - Number of unique orders a user has made
output final3=pd.DataFrame(unique orders)
output final3=output final3.rename(columns={0: 'id', 1:'Unique Orders', 2:'No. of Orders'})
output final3.sort values('No. of Orders', ascending=False, inplace=True)
print(output final3)
```

# **Annexure II - Snapshots of the Output**

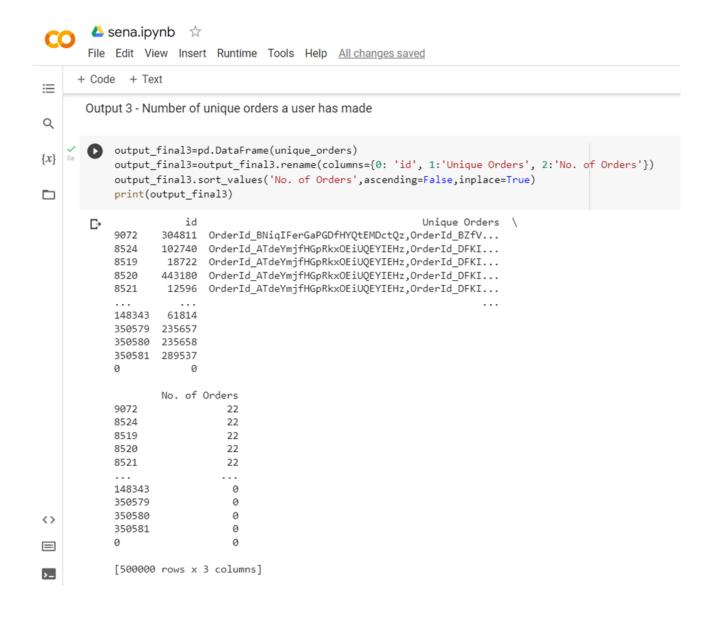
Output 1: Merging tickets belonging to the same user



Output 2 - Number of times a user has contacted customer service



# Output 3 - Number of unique orders a user has made



# References

- 1. Dataset link: <a href="https://www.kaggle.com/datasets/eugenevoon/multichannel-contacts-problem">https://www.kaggle.com/datasets/eugenevoon/multichannel-contacts-problem</a>
- 2. Problem statement: <a href="https://www.kaggle.com/competitions/scl-2021-da">https://www.kaggle.com/competitions/scl-2021-da</a>
- 3. https://pandas.pydata.org/docs/
- 4. <a href="https://networkx.org/documentation/stable/reference/index.html">https://networkx.org/documentation/stable/reference/index.html</a>
- 5. <a href="https://gephi.org/users/">https://gephi.org/users/</a>
- 6. <a href="https://www.geeksforgeeks.org/networkx-python-software-package-study-complex-netwo">https://www.geeksforgeeks.org/networkx-python-software-package-study-complex-netwo</a> <a href="https://www.geeksforgeeks.org/networkx-python-software-package-study-complex-netwo">rks/</a>
- 7. <a href="https://towardsdatascience.com/social-network-analysis-from-theory-to-applications-with-python-d12e9a34c2c7">https://towardsdatascience.com/social-network-analysis-from-theory-to-applications-with-python-d12e9a34c2c7</a>
- 8. https://www.cl.cam.ac.uk/teaching/1415/L109/1109-tutorial 2015.pdf
- 9. <a href="https://www.askpython.com/python/examples/network-graphs-from-pandas-dataframe">https://www.askpython.com/python/examples/network-graphs-from-pandas-dataframe</a>
- 10. <a href="https://www.python-graph-gallery.com/320-basic-network-from-pandas-data-frame/">https://www.python-graph-gallery.com/320-basic-network-from-pandas-data-frame/</a>