OLWETHU MATIWANE

COMCAST TELECOM COMPLAINT PROJECT 1

In [15]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import datetime
import calendar
from time import strptime
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [16]:
```
com_data = pd.read_csv('Comcast_telecom_complaints_data.csv')
```

In [3]:
```
com_data.head(5)
```

Out[3]:

	Ticket	Customer Complaint	Date	Date_month_year	Time	Received Via	City	State	Zip code	Status	Filing on Behalf of Someone
0	250635	Comcast Cable Internet Speeds	22-04-15	22-Apr-15	3:53:50 PM	Customer Care Call	Abingdon	Maryland	21009	Closed	No
1	223441	Payment disappear - service got disconnected	04-08-15	04-Aug-15	10:22:56 AM	Internet	Acworth	Georgia	30102	Closed	No
2	242732	Speed and Service	18-04-15	18-Apr-15	9:55:47 AM	Internet	Acworth	Georgia	30101	Closed	Yes
3	277946	Comcast Imposed a New Usage Cap of 300GB that ...	05-07-15	05-Jul-15	11:59:35 AM	Internet	Acworth	Georgia	30101	Open	Yes
4	307175	Comcast not working and no service to boot	26-05-15	26-May-15	1:25:26 PM	Internet	Acworth	Georgia	30101	Solved	No

In [4]:
```
com_data = pd.DataFrame(com_data)
```

In [5]:
```
com_data.isna().sum()
```

Out[5]:
```
Ticket                        0
Customer Complaint            0
Date                          0
Date_month_year               0
Time                          0
Received Via                  0
City                          0
State                         0
Zip code                      0
Status                        0
Filing on Behalf of Someone   0
dtype: int64
```

In [6]:
```
com_data.info()
```

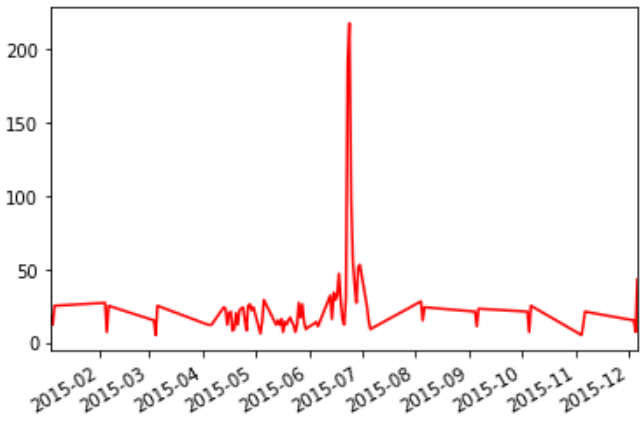
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2224 entries, 0 to 2223
Data columns (total 11 columns):
Ticket                          2224 non-null object
Customer Complaint              2224 non-null object
Date                            2224 non-null object
Date_month_year                 2224 non-null object
Time                            2224 non-null object
Received Via                    2224 non-null object
City                            2224 non-null object
State                           2224 non-null object
Zip code                        2224 non-null int64
Status                          2224 non-null object
Filing on Behalf of Someone     2224 non-null object
dtypes: int64(1), object(10)
memory usage: 191.2+ KB
```

In [7]:
```
# parsing the string to datetime
com_data['new_parsed_date'] = pd.to_datetime(com_data['Date_month_year'])
```

In [8]:
```
# daily trend
print(com_data['new_parsed_date'].value_counts().head())
com_data['new_parsed_date'].value_counts().plot(stacked = True, color = 'r')
```

```
2015-06-24    218
2015-06-23    190
2015-06-25     98
2015-06-26     55
2015-06-30     53
Name: new_parsed_date, dtype: int64
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c83cd406d8>
```



In [9]:
```
com_data['year'] = pd.DatetimeIndex(com_data['new_parsed_date']).year
com_data['month'] = pd.DatetimeIndex(com_data['new_parsed_date']).month
com_data['month'] = com_data['month'].apply(lambda x: calendar.month_abbr[x])
```

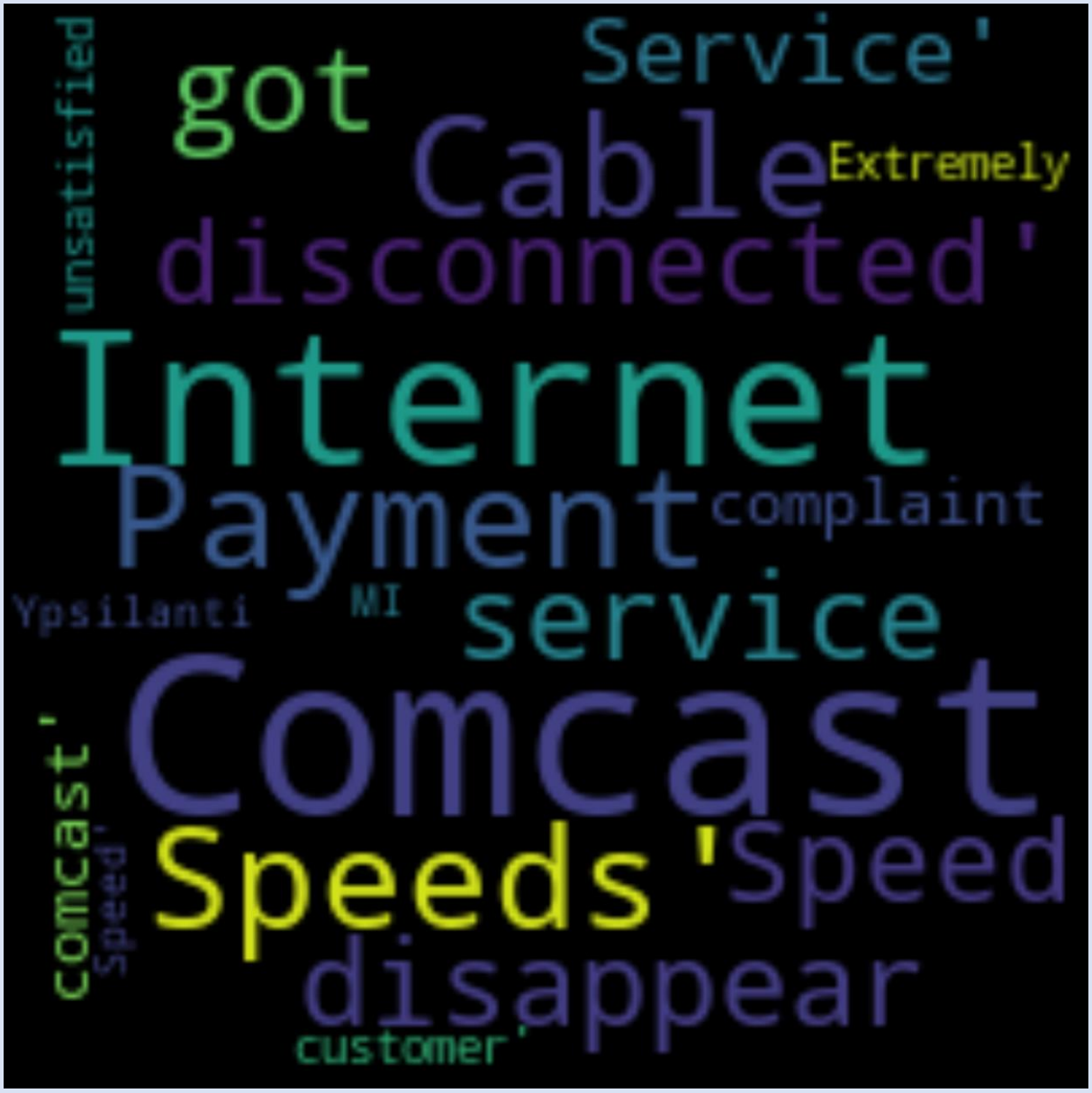
In [10]:
```
com_data['month_year'] = com_data['month'] + com_data['year'].astype(str)
```

In [46]:
```
import re # We clean text using regex
from collections import defaultdict # For accumlating values
from nltk.corpus import stopwords # To remove stopwords
from gensim import corpora # To create corpus and dictionary for the LDA model
from gensim.models import LdaModel # To use the LDA model
import pyLDAvis.gensim # To visualise LDA model effectively
import pandas as pd
import nltk
import string
from nltk import pos_tag
from nltk.corpus import stopwords
stopwords = set(stopwords.words('english'))
```

In [32]:
```
# viewing the wordcloud for Customer complaint
from wordcloud import WordCloud, STOPWORDS

text = com_data['Customer Complaint'].values
wordcloud = WordCloud(width = 200, height = 200, background_color = 'black', stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(figsize = (40, 30), facecolor = 'k', edgecolor = 'k')

plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



In [12]:
```
# load reviews
rev = com_data["Customer Complaint"].tolist()
```

In [16]:
```
# removing all punctuations
rev = [re.sub(r'[^\w\s]','',str(item)) for item in rev]
```

In [17]:
```
from collections import defaultdict
# removing stop-words
txt = [[word for word in document.lower().split() if word not in stopwords] for document in rev]

# taking out the less frequent words
freq = defaultdict(int)
for text in txt:
    for token in text:
        freq[token] += 1

txt = [[token for token in txt if freq[token] > 1] for text in txt]
```

In [18]:
```
#Machine can't understand words and documents as they are. So we split and vectorize them.
#Turning our text to a dictionary i.e. mapping between words and their integer ids.

#In this representation, each word is represented by one vector, where each vector element represents a question-answer pair.
#e.g. "How many times does the word 'Payment' appear in the coprus? Twice."

dic = corpora.Dictionary(txt)
print(dic)

corpus = [dic.doc2bow(text) for text in txt]
corpus
```

```
Dictionary(1498 unique tokens: ['cable', 'comcast', 'internet', 'speeds', 'disappear']...)
```

Out[18]:
```
[[(0, 1), (1, 1), (2, 1), (3, 1)],
 [(4, 1), (5, 1), (6, 1), (7, 1), (8, 1)],
 [(8, 1), (9, 1)],
 [(1, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1), (16, 1)],
 [(1, 1), (8, 1), (17, 1), (18, 1)],
 [(19, 1), (20, 1), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1)],
 [(8, 1), (21, 1), (26, 1), (27, 1), (28, 1)],
 [(1, 1), (8, 1), (29, 1), (30, 1), (31, 1), (32, 1)],
 [(1, 1), (33, 1), (34, 1)],
 [(1, 1), (35, 1), (36, 1), (37, 1), (38, 1)],
 [(5, 1), (8, 1), (39, 1), (40, 1)],
 [(41, 1), (42, 1), (43, 1), (44, 1), (45, 1), (46, 1)],
 [(1, 1),
  (2, 1),
  (47, 1),
  (48, 1),
  (49, 1),
```

In [22]:
```
#LDA MODEL
NUM_TOPICS = 9 #  This is a Assumption. You can vary this
ldamodel = LdaModel(corpus, num_topics = NUM_TOPICS, id2word=dic, passes=15)
#This might take some time based on num_topics and passes.
```

In [24]:
```
#Extracting Topics from the model
topics = ldamodel.show_topics()
for topic in topics:
    print(topic)
```

```
(0, '0.115*"comcast" + 0.062*"charges" + 0.028*"issue" + 0.023*"fraudulent" + 0.023*"monopolistic" +
0.022*"bill" + 0.022*"lack" + 0.020*"show" + 0.019*"fee" + 0.018*"credit"')
(1, '0.051*"comcast" + 0.033*"false" + 0.026*"switch" + 0.025*"price" + 0.022*"charging" + 0.021*"dec
eptive" + 0.020*"advertising" + 0.020*"phone" + 0.018*"months" + 0.017*"bait"')
(2, '0.120*"internet" + 0.094*"comcast" + 0.058*"billing" + 0.055*"speeds" + 0.052*"practices" + 0.04
6*"unfair" + 0.043*"slow" + 0.040*"throttling" + 0.028*"services" + 0.023*"pricing"')
(3, '0.236*"data" + 0.137*"comcast" + 0.097*"caps" + 0.093*"cap" + 0.045*"usage" + 0.015*"xfinity" +
0.012*"overage" + 0.010*"limit" + 0.009*"scam" + 0.008*"fees"')
(4, '0.156*"billing" + 0.145*"comcast" + 0.050*"issues" + 0.042*"service" + 0.023*"comcastxfinity" +
0.022*"bill" + 0.020*"refund" + 0.017*"back" + 0.015*"incorrect" + 0.015*"customer"')
(5, '0.079*"cable" + 0.042*"high" + 0.037*"internet" + 0.036*"monthly" + 0.036*"bill" + 0.033*"connec
tion" + 0.026*"service" + 0.025*"paying" + 0.023*"broadband" + 0.023*"prices"')
(6, '0.098*"service" + 0.091*"comcast" + 0.027*"customer" + 0.024*"charge" + 0.022*"terrible" + 0.020
*"help" + 0.019*"without" + 0.017*"failure" + 0.016*"account" + 0.011*"please"')
(7, '0.216*"comcast" + 0.167*"internet" + 0.124*"service" + 0.044*"speed" + 0.044*"complaint" + 0.028
*"xfinity" + 0.016*"poor" + 0.011*"customer" + 0.011*"problems" + 0.009*"business"')
(8, '0.046*"comcast" + 0.034*"issues" + 0.033*"pay" + 0.032*"get" + 0.031*"charged" + 0.023*"billed"
+ 0.023*"service" + 0.021*"services" + 0.019*"several" + 0.018*"low"')
```

In [29]:
```
#Above result looks too messy to understand easily. Let's print them in a better view using pandas dataframe.

word_dict = {};
for i in range(NUM_TOPICS):
    words = ldamodel.show_topic(i, topn = 20)
    word_dict['Topic # ' + '{:02d}'.format(i+1)] = [i[0] for i in words]
com_data.DataFrame(word_dict)
```

Out[17]:

	Topic # 01	Topic # 02	Topic # 03	Topic # 04	Topic # 05	Topic # 06	Topic # 07	Topic # 08	Topic # 09
0	cap	internet	service	service	service	comcast	billing	comcast	internet
1	comcast	speeds	comcast	comcast	comcast	data	comcast	bill	speed
2	data	comcast	customer	charges	internet	internet	complaint	without	service
3	cable	slow	fees	poor	connection	caps	issues	services	services
4	false	pricing	charged	help	charge	xfinity	practices	high	outage
5	price	service	day	lack	problems	throttling	unfair	service	phone
6	deceptive	paying	account	overage	terrible	usage	service	monthly	signal
7	charging	comcastxfinity	services	cramming	unreliable	contract	issue	billed	several
8	advertising	switch	poor	credit	months	business	monopolistic	refund	availability
9	back	connectivity	horrible	misleading	fee	service	bandwidth	modem	2
10	hbo	intermittent	unauthorized	get	quality	broadband	throttled	payment	loss
11	sales	promised	home	complaint	lied	monopoly	fraudulent	increased	years
12	go	bait	refusal	failure	equipment	email	regarding	prices	provider
13	xfinitycomcast	low	practice	please	installation	show	improper	said	provided
14	much	shitty	extremely	bad	complaints	12	isp	notice	claims
15	trade	mbs	still	provide	incorrect	limit	way	wont	prices
16	promotion	access	cable	days	year	300gb	higher	incorrect	misrepresentation
17	transfer	speed	agreement	fraudulent	excessive	plan	failing	pay	inability
18	request	scam	security	xfinity	didnt	ps4	unresolved	added	charged

In [30]:
```
# Create a new categorical variable with value as - Open and Closed.
# Open & Pending to be categorized as 'Open' and Closed & Solved to be categorized as 'Closed'

com_data['Status'].unique()
```

Out[30]:
```
array(['Closed', 'Open', 'Solved', 'Pending'], dtype=object)
```

In [31]:
```
com_data.assign(New_Status = "")
```

t	Customer Complaint	Date	Date_month_year	Time	Received Via	City	State	Zip code	Status	Filing on Behalf of Someone	new_parsed_date	year	month	month_year	New_Status
5	Comcast Cable Internet Speeds	22-04-15	22-Apr-15	3:53:50 PM	Customer Care Call	Abingdon	Maryland	21009	Closed	No	2015-04-22	2015	Apr	Apr2015	
1	Payment disappear - service got disconnected	04-08-15	04-Aug-15	10:22:56 AM	Internet	Acworth	Georgia	30102	Closed	No	2015-08-04	2015	Aug	Aug2015	
2	Speed and Service	18-04-15	18-Apr-15	9:55:47 AM	Internet	Acworth	Georgia	30101	Closed	Yes	2015-04-18	2015	Apr	Apr2015	
6	Comcast Imposed a New Usage Cap of 300GB that ...	05-07-15	05-Jul-15	11:59:35 AM	Internet	Acworth	Georgia	30101	Open	Yes	2015-07-05	2015	Jul	Jul2015	

In [32]:
```
# Open & Pending to be categorized as 'Open' and Closed & Solved to be categorized as 'Closed'

com_data['New_Status'] = ['Open' if (x == 'Open' or x =='Pending') else 'Closed' for x in com_data['Status']]
```

In [34]:
```
state_count = pd.core.frame.DataFrame({"count": com_data.groupby(['State']).size().sort_values(ascending=False)}).reset_index()
state_count.head(10)
```

Out[34]:

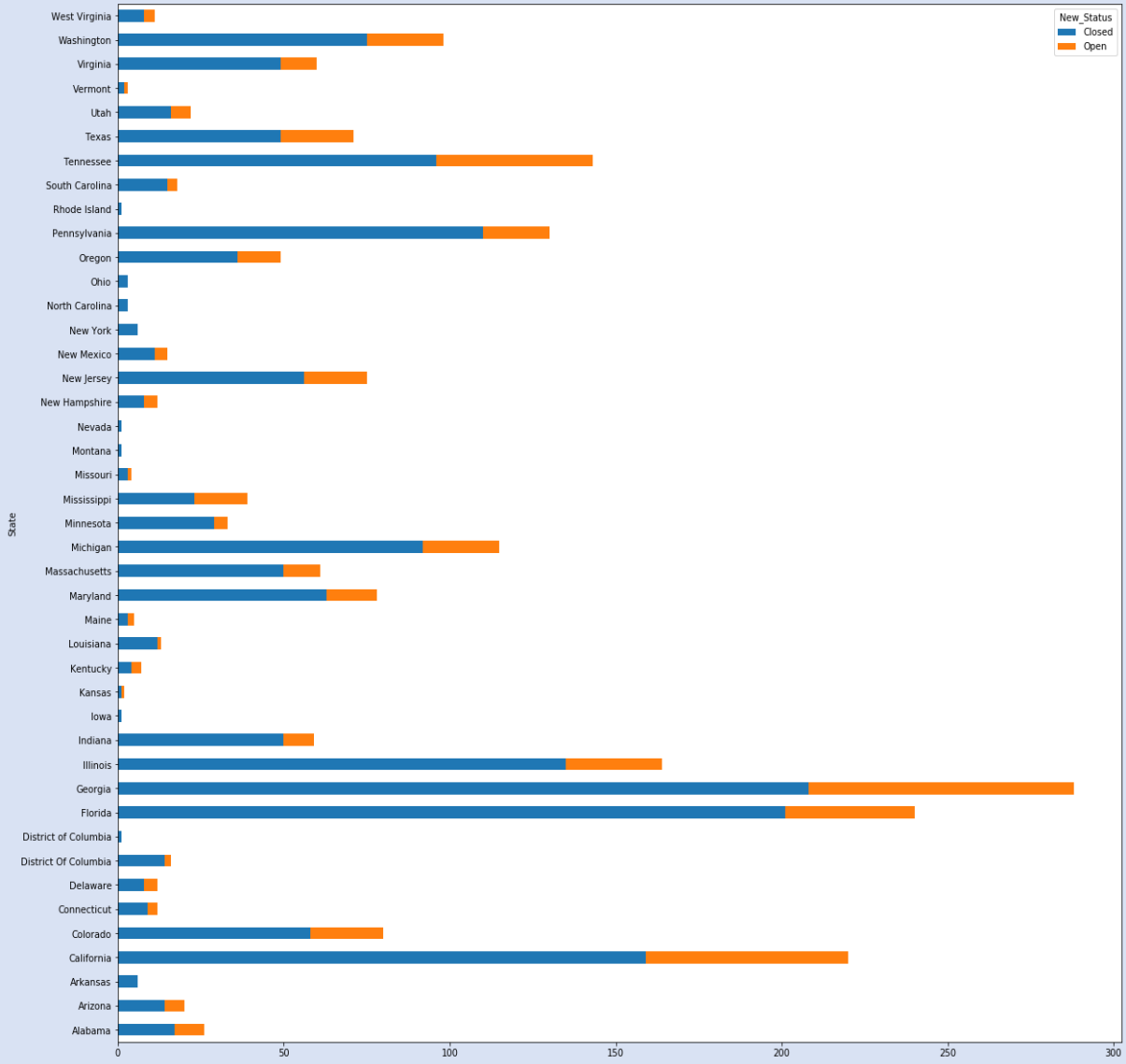
	State	count
0	Georgia	288
1	Florida	240
2	California	220
3	Illinois	164
4	Tennessee	143
5	Pennsylvania	130
6	Michigan	115
7	Washington	98
8	Colorado	80
9	Maryland	78

In [35]:
```
Status_com = com_data.groupby(['State', 'New_Status']).size().unstack()
Status_com.head()
```

Out[35]:

New_Status	Closed	Open
State		
Alabama	17.0	9.0
Arizona	14.0	6.0
Arkansas	6.0	NaN
California	159.0	61.0
Colorado	58.0	22.0

In [36]:
```
#State wise status of complaints in a stacked bar chart
com_data.groupby(['State', 'New_Status']).size().unstack().plot.barh(stacked=True,figsize=(20,20))
```




```
In [37]: Percentage_unresolved = [ Status_com['Open'] / (Status_com['Open']+Status_com['Closed'])]
         Percentage_unresolved

         # kansas has the highest nymber of unresolved cases
```

Out[37]:
```
[State
 Alabama                  0.346154
 Arizona                  0.300000
 Arkansas                      NaN
 California               0.277273
 Colorado                 0.275000
 Connecticut              0.250000
 Delaware                 0.333333
 District Of Columbia     0.125000
 District of Columbia          NaN
 Florida                  0.162500
 Georgia                  0.277778
 Illinois                 0.176829
 Indiana                  0.152542
 Iowa                          NaN
 Kansas                   0.500000
 Kentucky                 0.428571
 Louisiana                0.076923
 Maine                    0.400000
 Maryland                 0.192308
 Massachusetts            0.180328
 Michigan                 0.200000
 Minnesota                0.121212
 Mississippi              0.410256
 Missouri                 0.250000
 Montana                       NaN
 Nevada                        NaN
```

```
 New Hampshire            0.333333
 New Jersey               0.253333
 New Mexico               0.266667
 New York                      NaN
 North Carolina                NaN
 Ohio                          NaN
 Oregon                   0.265306
 Pennsylvania             0.153846
 Rhode Island                  NaN
 South Carolina           0.166667
 Tennessee                0.328671
 Texas                    0.309859
 Utah                     0.272727
 Vermont                  0.333333
 Virginia                 0.183333
 Washington               0.234694
 West Virginia            0.272727
 dtype: float64]
```

In [38]:
```
com_data['Received Via'].unique()

#notice that there are only two categories
```

Out[38]:
```
array(['Customer Care Call', 'Internet'], dtype=object)
```

In [39]:
```
num_per1 = com_data.groupby(['New_Status']).size()
num_per1
```

Out[39]:
```
New_Status
Closed    1707
Open       517
dtype: int64
```

In [40]:
```
num_per2 = com_data.groupby(['New_Status']).size().sum()
num_per2
```

Out[40]:
```
2224
```

In [41]:
```
percentage_resolved = num_per1[0]/num_per2*100
percentage_resolved.round()
```

Out[41]:
```
77.0
```

In [ ]:
```
# Resolution rate is 77% for the complaints received through Internet and customer care calls.
```