In [179…
```
#..Import libraries
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

In [180…
```
##...Import the dataset
data = pd.read_csv("Comcast_telecom_complaints_data.csv",index_col=0,parse_dates=[2,3])
```

In [181…
```
data
```

Out[181]:

Ticket #	Customer Complaint	Date	Date_month_year	Time	Received Via	City	State	Zip code	Status	Filing on Behalf of Someone
250635	Comcast Cable Internet Speeds	2015-04-22	2015-04-22	3:53:50 PM	Customer Care Call	Abingdon	Maryland	21009	Closed	No
223441	Payment disappear - service got disconnected	2015-04-08	2015-08-04	10:22:56 AM	Internet	Acworth	Georgia	30102	Closed	No
242732	Speed and Service	2015-04-18	2015-04-18	9:55:47 AM	Internet	Acworth	Georgia	30101	Closed	Yes
277946	Comcast Imposed a New Usage Cap of 300GB that ...	2015-05-07	2015-07-05	11:59:35 AM	Internet	Acworth	Georgia	30101	Open	Yes
307175	Comcast not working and no service to boot	2015-05-26	2015-05-26	1:25:26 PM	Internet	Acworth	Georgia	30101	Solved	No
...	...	...	...	...	...	...	...	...	...	...
213550	Service Availability	2015-04-02	2015-02-04	9:13:18 AM	Customer Care Call	Youngstown	Florida	32466	Closed	No
318775	Comcast Monthly Billing for Returned Modem	2015-06-02	2015-02-06	1:24:39 PM	Customer Care Call	Ypsilanti	Michigan	48197	Solved	No
331188	complaint about comcast	2015-06-09	2015-09-06	5:28:41 PM	Internet	Ypsilanti	Michigan	48197	Solved	No
360489	Extremely unsatisfied Comcast customer	2015-06-23	2015-06-23	11:13:30 PM	Customer Care Call	Ypsilanti	Michigan	48197	Solved	No
363614	Comcast, Ypsilanti MI Internet Speed	2015-06-24	2015-06-24	10:28:33 PM	Customer Care Call	Ypsilanti	Michigan	48198	Open	Yes

2224 rows × 10 columns

Explore the data

In [182…
```
#check  the shape of data
data.shape
```

Out[182]:
```
(2224, 10)
```

In [183…
```
# Check the info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2224 entries, 250635 to 363614
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Customer Complaint           2224 non-null   object
 1   Date                         2224 non-null   datetime64[ns]
 2   Date_month_year              2224 non-null   datetime64[ns]
 3   Time                         2224 non-null   object
 4   Received Via                 2224 non-null   object
 5   City                         2224 non-null   object
 6   State                        2224 non-null   object
 7   Zip code                     2224 non-null   int64
 8   Status                       2224 non-null   object
 9   Filing on Behalf of Someone  2224 non-null   object
dtypes: datetime64[ns](2), int64(1), object(7)
memory usage: 191.1+ KB
```

In [184…
```
#check null values
data.isnull().sum()
```

Out[184]:
```
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
```

Provide the trend chart for the number of complaints at monthly and daily granularity levels.

In [185…
```
#Date of the month Table Trend Chart..monthly datewise
```

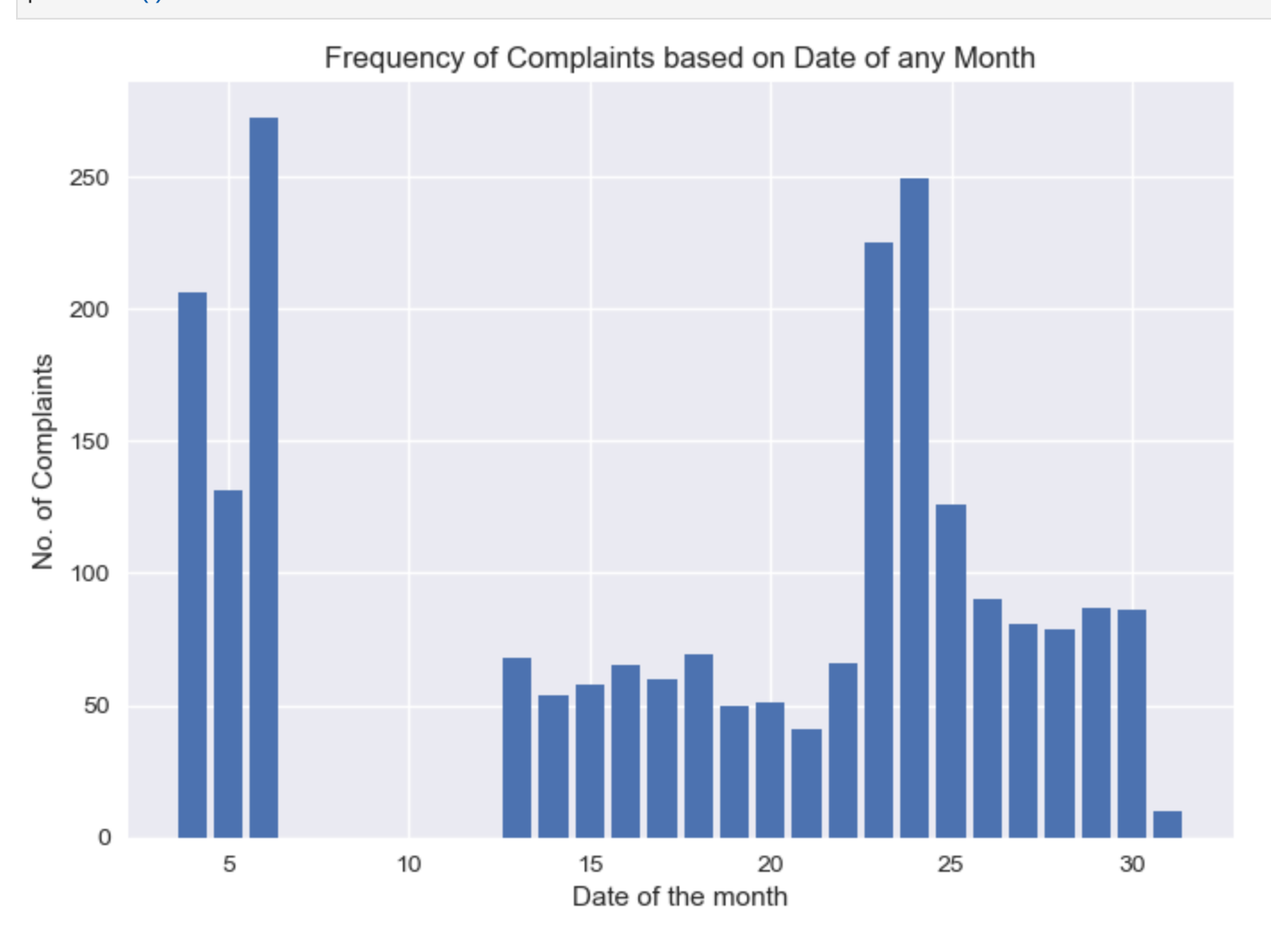
In [186…
```
#importing required date time libraries

import datetime
#from dateutil.parser import parse
#import pytz
```

In [187…
```
#Date of the month Table
data['Date of the Month'] = data['Date_month_year'].apply(lambda d: d.day)
monthlydate = data['Customer Complaint'].groupby(data['Date of the Month']).count().reset_index()
monthlydate
```

Out[187]:

	Date of the Month	Customer Complaint
0	4	206
1	5	131
2	6	272
3	13	68
4	14	54
5	15	58
6	16	65
7	17	60
8	18	69
9	19	50
10	20	51
11	21	41
12	22	66
13	23	225
14	24	249
15	25	126
16	26	90
17	27	81
18	28	79
19	29	87
20	30	86
21	31	10

In [188…
```
#Date of the month Table Trend Chart

#Highest number of complaints --date..6th
# No any complaint -- date..7 to 12th

plt.style.use('seaborn')
plt.bar(monthlydate["Date of the Month"],monthlydate["Customer Complaint"])
plt.xlabel("Date of the month")
plt.ylabel("No. of Complaints")
plt.title("Frequency of Complaints based on Date of any Month")
plt.show()
```



In [189…
```
# Trend Chart---month wise data
```

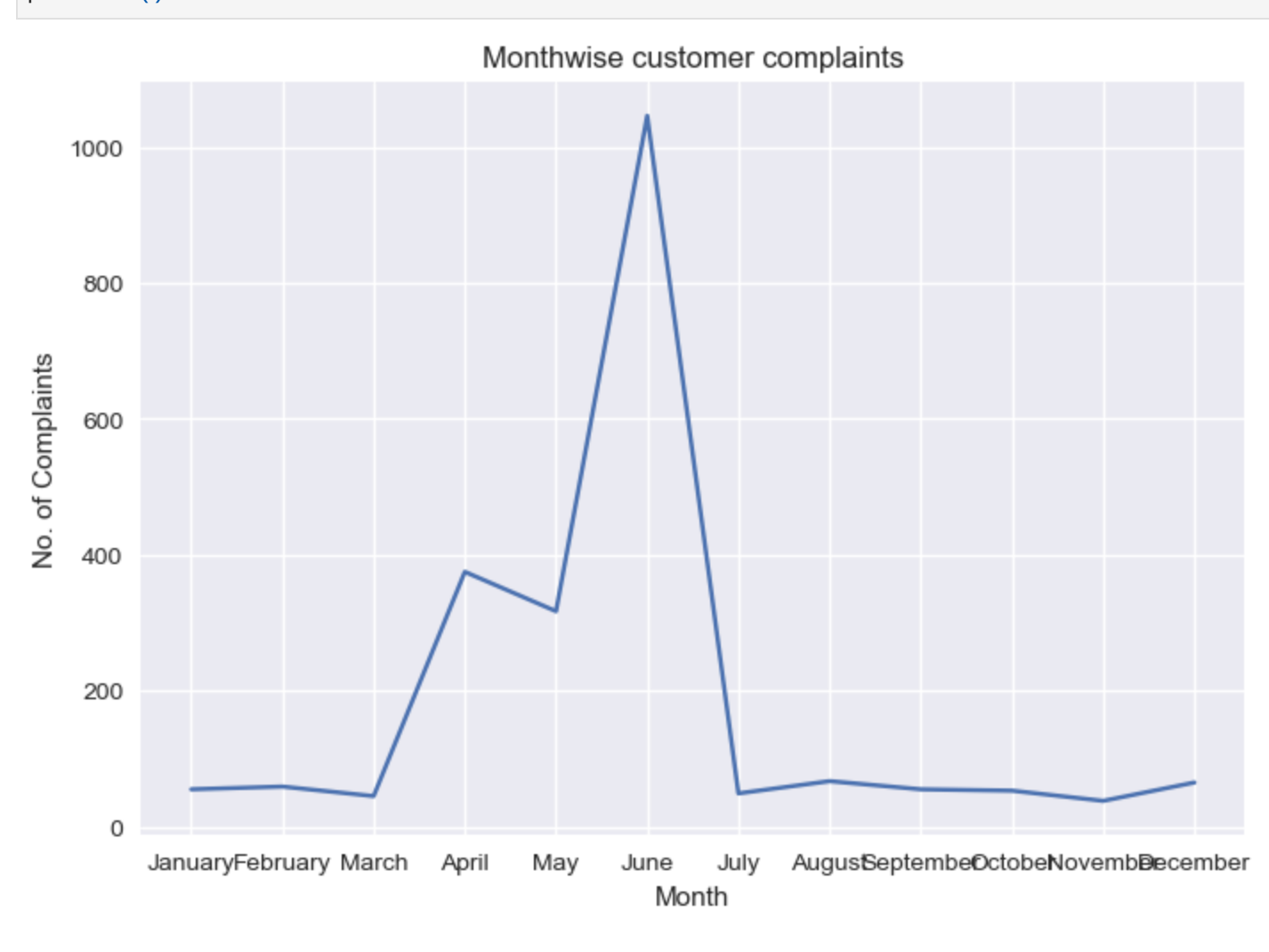
In [190…
```
#month wise data table
monthlywise = data.groupby(pd.Grouper(key='Date_month_year',freq='M')).count()['Customer Complaint'].reset_index()

monthlywise['month'] = monthlywise['Date_month_year'].dt.strftime('%B')

monthlywise.drop(['Date_month_year'], axis='columns', inplace=True)
```

In [191…
```
monthlywise
```

Out[191]:

	Customer Complaint	month
0	55	January
1	59	February
2	45	March
3	375	April
4	317	May
5	1046	June
6	49	July
7	67	August
8	55	September
9	53	October
10	38	November
11	65	December

In [192…
```
# Trend Chart---month wise data


#June-- highest complaints

plt.style.use('seaborn')
plt.plot(monthlywise['month'],monthlywise['Customer Complaint'])
plt.xlabel("Month")
plt.ylabel("No. of Complaints")
plt.title("Monthwise customer complaints")
plt.show()
```



In [193…
```
#Trend chart weekwise data
```

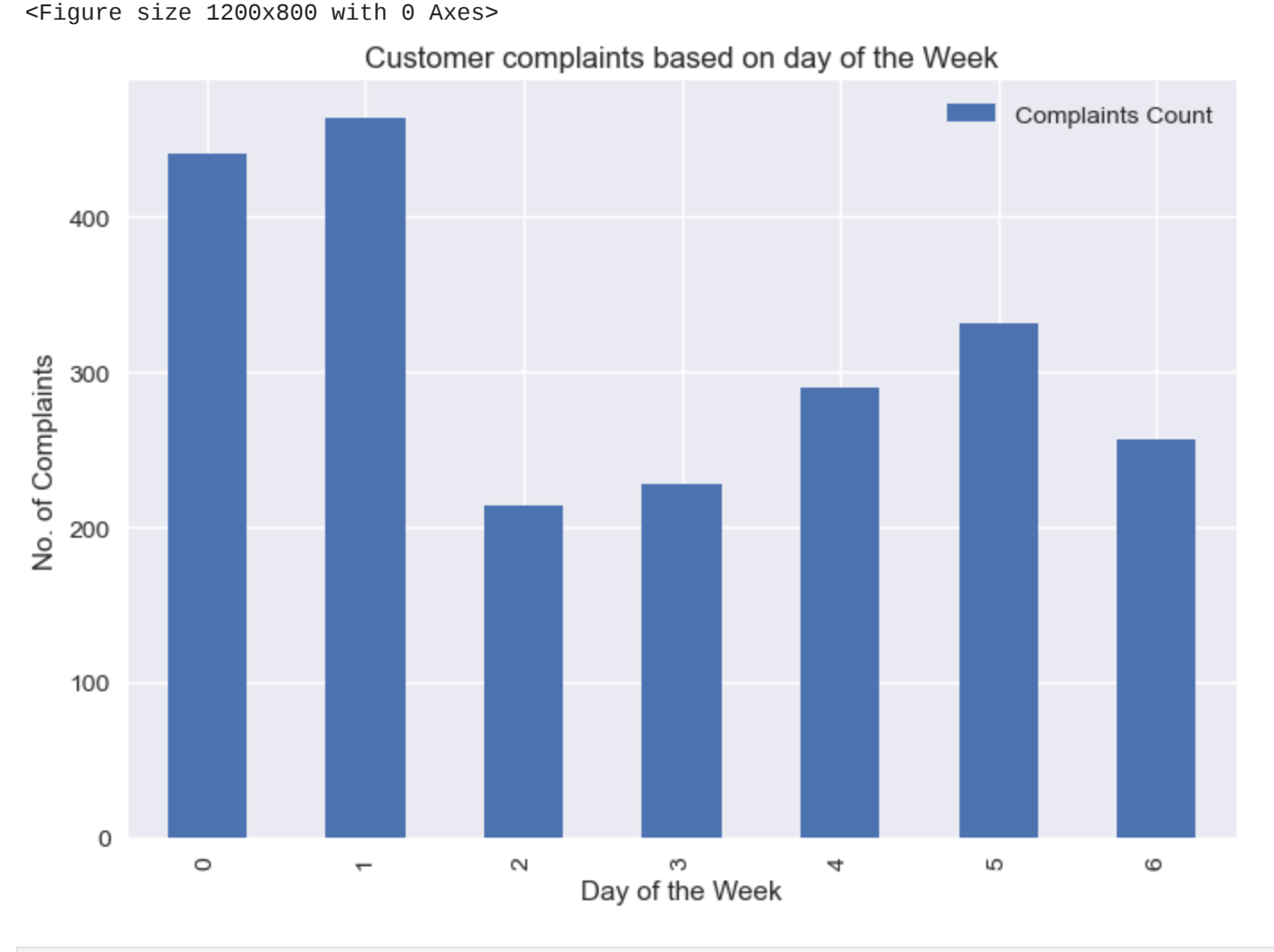
In [194…
```
#weekwise data table
data['day_of_week'] = data['Date_month_year'].dt.day_name() #Creating a Weekday Column
weekwise= data['day_of_week'].value_counts(sort=False).reset_index()
weekwise.columns = ['Day of Week', 'Complaints Count']
weekwise
```

Out[194]:

	Day of Week	Complaints Count
0	Wednesday	441
1	Tuesday	464
2	Saturday	214
3	Sunday	228
4	Friday	290
5	Thursday	331
6	Monday	256

In [195…
```
#Creating Trend Chart

# Tuesday, Wednesday--highest complaints &
#Saturday__lowest

plt.style.use('seaborn')
plt.figure(figsize=(12,8))
weekwise.plot(kind='bar')
plt.xlabel("Day of the Week")
plt.ylabel("No. of Complaints")
plt.title("Customer complaints based on day of the Week")
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [196…
```
#Trend chart of complaint---- everyday
```

In [197…
```
# Creating daily complaints table and arranging them in order
everyday = data.groupby(pd.Grouper(key='Date_month_year', freq='D')).count()['Customer Complaint'].reset_index()
everyday.sort_values(by ='Customer Complaint', ascending=False )
everyday
```

Out[197]:

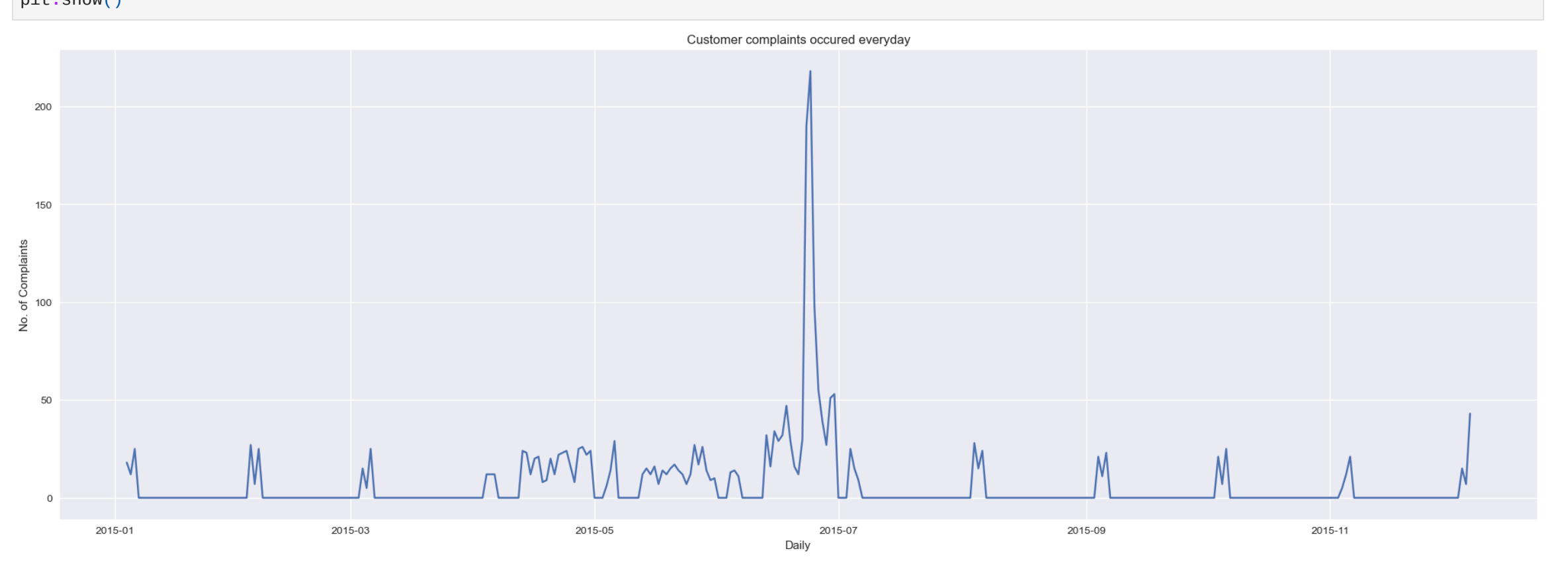
	Date_month_year	Customer Complaint
0	2015-01-04	18
1	2015-01-05	12
2	2015-01-06	25
3	2015-01-07	0
4	2015-01-08	0
...	...	...
332	2015-12-02	0
333	2015-12-03	0
334	2015-12-04	15
335	2015-12-05	7
336	2015-12-06	43

337 rows × 2 columns

In [198…
```
everyday.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 337 entries, 0 to 336
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date_month_year     337 non-null    datetime64[ns]
 1   Customer Complaint  337 non-null    int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 5.4 KB
```

In [199…
```
#Creating Trend Chart

plt.style.use('seaborn')
plt.figure(figsize=(25,8))
plt.plot(everyday['Date_month_year'],everyday['Customer Complaint'])
plt.xlabel("Daily")
plt.ylabel("No. of Complaints")
plt.title("Customer complaints occured everyday")
plt.show()
```



Provide a table with the frequency of complaint types

In [200…
```
data['Customer Complaint'].value_counts().to_frame().reset_index()
```

Out[200]:

	index	Customer Complaint
0	Comcast	83
1	Comcast Internet	18
2	Comcast Data Cap	17
3	comcast	13
4	Comcast Billing	11
...	...	...
1836	Improper Billing and non resolution of issues	1
1837	Deceptive trade	1
1838	intermittent internet	1
1839	Internet Speed on Wireless Connection	1
1840	Comcast, Ypsilanti MI Internet Speed	1

1841 rows × 2 columns

- Which complaint types are maximum i.e., around internet, network issues, or across any other domains.

In [201…
```
data['Customer Complaint'].value_counts().head(5)
```

Out[201]:
```
Comcast             83
Comcast Internet    18
Comcast Data Cap    17
comcast             13
Comcast Billing     11
Name: Customer Complaint, dtype: int64
```

In [202…
```
## - Create a new categorical variable with value as Open and Closed. Open & Pending is to be categorized as Open and
## Closed & Solved is to be categorized as Closed.
```

In [203…
```
#Creating a function which returns open for open & pending cases and returns closed for closed and solved cases
def new_status(col):
    if col == "Open":
        val = "Open"
    elif col == "Pending":
        val = "Open"
    elif col == "Solved":
        val = "Closed"
    else:
        val = "Closed"
    return val
```

In [204…
```
data['New Status'] = data['Status'].apply(new_status)
```

In [205…
```
n= data['New Status'].value_counts()
n
```

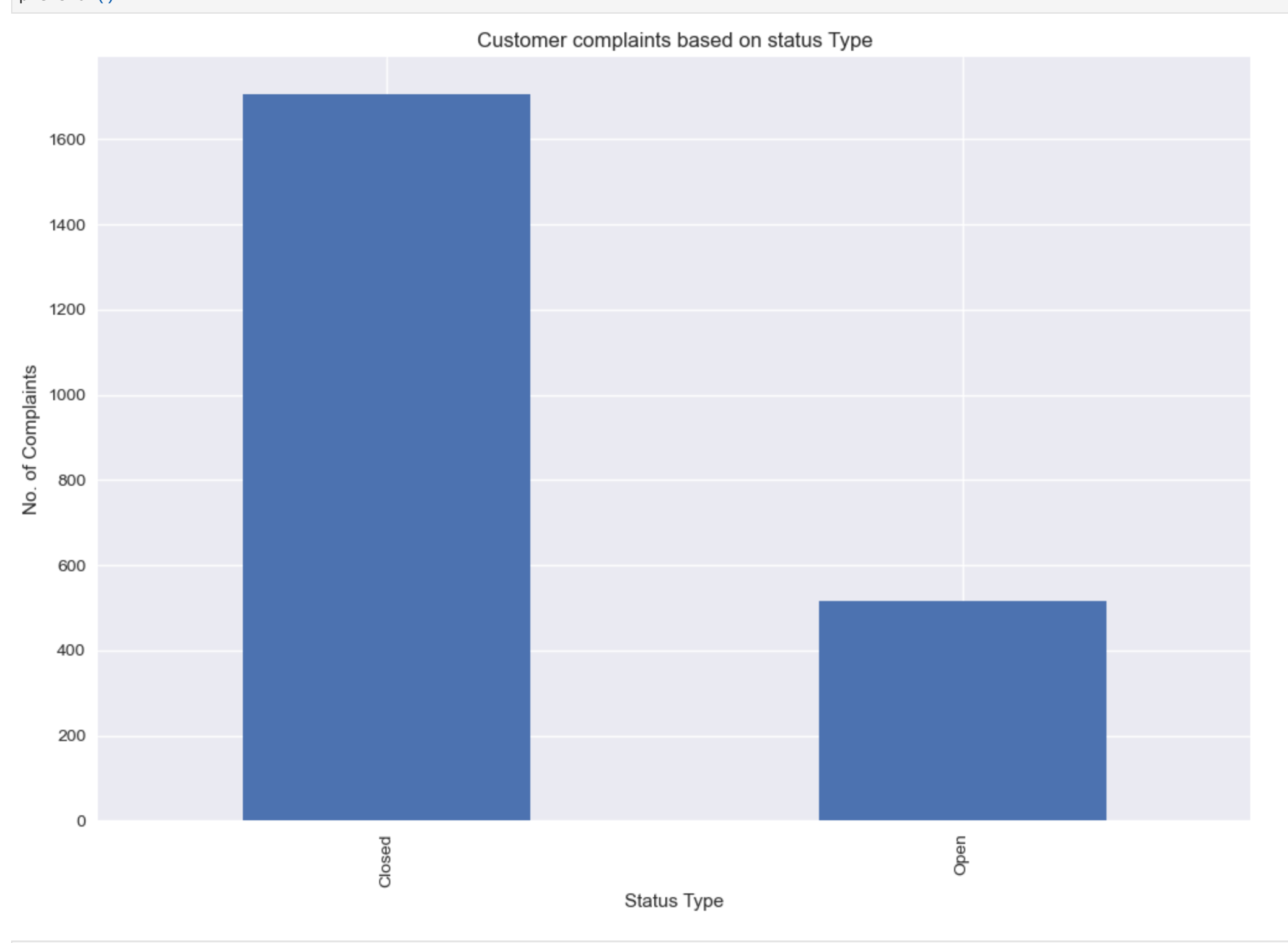
Out[205]:
```
Closed    1707
Open       517
Name: New Status, dtype: int64
```

In [206…
```
data['Status'].value_counts()
```

Out[206]:
```
Solved     973
Closed     734
Open       363
Pending    154
Name: Status, dtype: int64
```

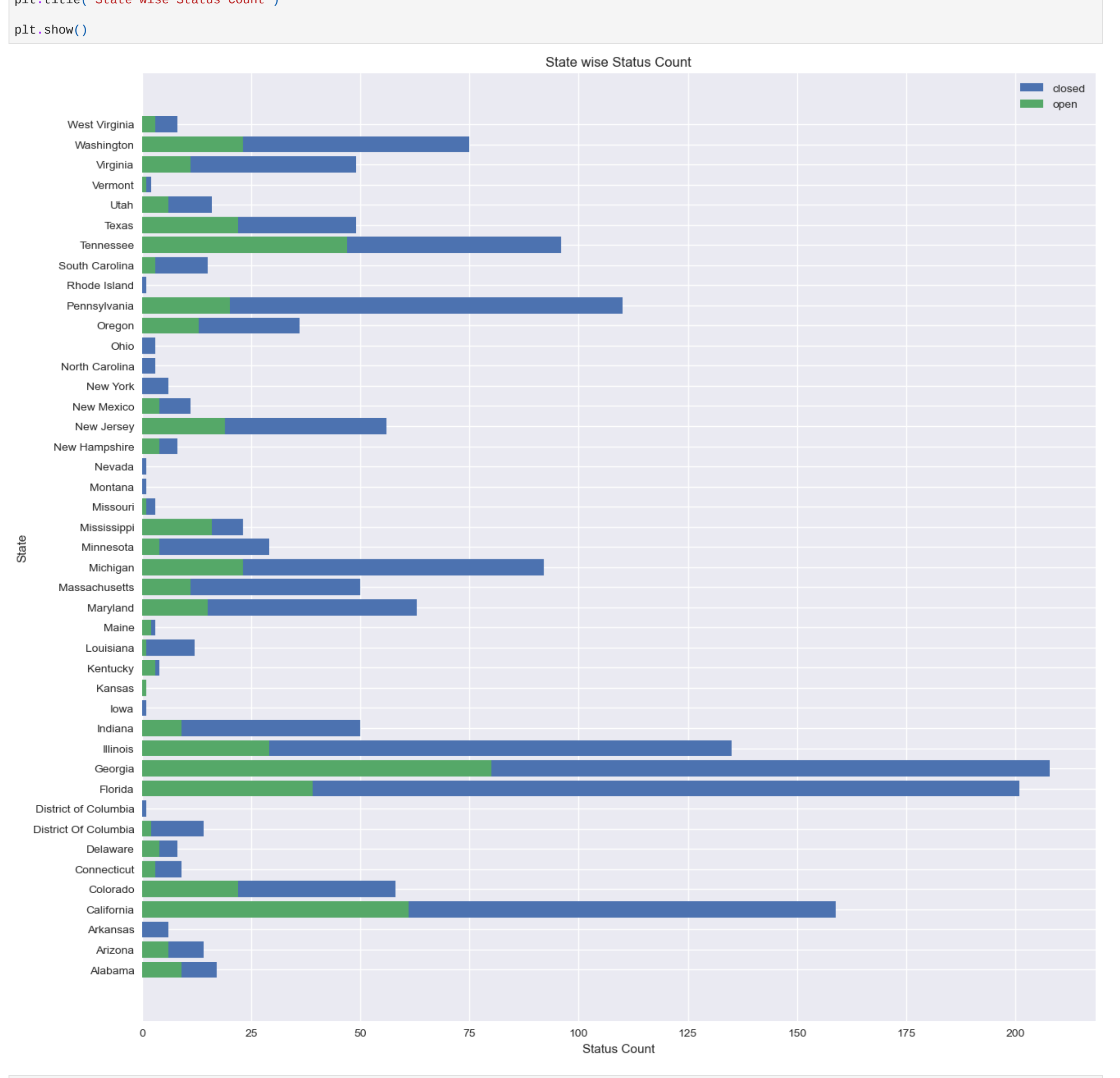
In [207…
```
### Sum of solved & closed = 973+734 = 1707, sum of Open & Pending = 363+154 = 517. Values are matching hence the answers are correct
```

In [208…
```
#Plotting the data
plt.style.use('seaborn')
plt.figure(figsize=(12,8))
n.plot(kind='bar')
plt.xlabel("Status Type")  # add X-axis label
plt.ylabel("No. of Complaints")  # add Y-axis label
plt.title("Customer complaints based on status Type")  # add title
plt.show()
```



In [209…
```
## - Create a new categorical variable with value as Open and Closed. Open & Pending is to be categorized as Open and
## Closed & Solved is to be categorized as Closed.

data['Status']=data['Status'].apply(lambda x: 'Open' if ((x=='Open') | (x=='Pending')) else 'Closed')
```

In [210…
```
## - Provide state wise status of complaints in a stacked bar chart. Use the categorized variable from Q3.
opn=data[data['Status']=='Open'].groupby(['State'])['Status'].count().to_frame().reset_index()
clos=data[data['Status']=='Closed'].groupby(['State'])['Status'].count().to_frame().reset_index()

fig=plt.figure(figsize=(15,15))
plt.barh(clos.State, clos.Status)
plt.barh(opn.State, opn.Status)
plt.ylabel("State")
plt.xlabel("Status Count")
plt.legend(["closed", "open"])
plt.title("State wise Status Count")

plt.show()
```

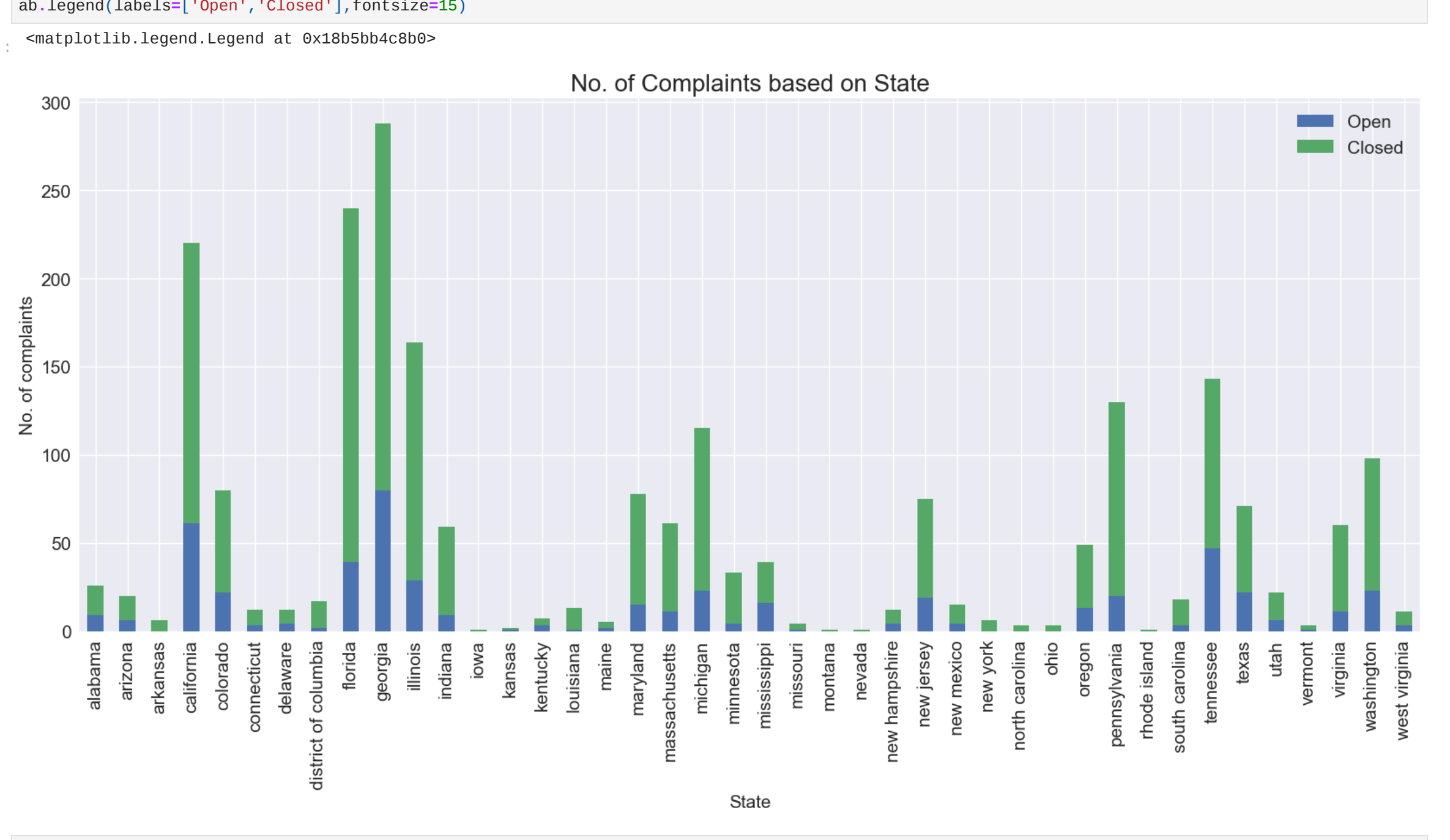


In [211…
```
## Which state has the maximum complaints

##   Georgia--highest Complaints

data.groupby("State")['Customer Complaint'].agg("count").sort_values(ascending=False).head(1)
```

Out[211]:
```
State
Georgia    288
Name: Customer Complaint, dtype: int64
```

In [212…
```
#Creating Stacked bar chart using New Status Created
data["State"]= data["State"].str.lower() #District of colombia is in different formats
state_chart = data.groupby(['State', 'New Status'])['State'].count().unstack('New Status').fillna(0)
ab = state_chart[['Open','Closed']].plot(kind='bar', stacked=True,rot=90,figsize=(20,8),fontsize=15)
ab.set_xlabel("State",fontsize=15)
ab.set_ylabel("No. of complaints",fontsize=15)
ab.set_title("No. of Complaints based on State",fontsize=20)
ab.legend(labels=['Open','Closed'],fontsize=15)
```

Out[212]:
```
<matplotlib.legend.Legend at 0x18b5bb4c8b0>
```



In [213…
```
## Which state has the highest percentage of unresolved complaints

State_Unsolved=data.loc[data['Status']=='Open',['State']].value_counts()
State_Unsolved.head(1)/State_Unsolved.sum()*100
```

Out[213]:
```
State
georgia    15.473888
dtype: float64
```

In [214…
```
# Provide the percentage of complaints resolved till date, which were received through the Internet and customer care calls.
```

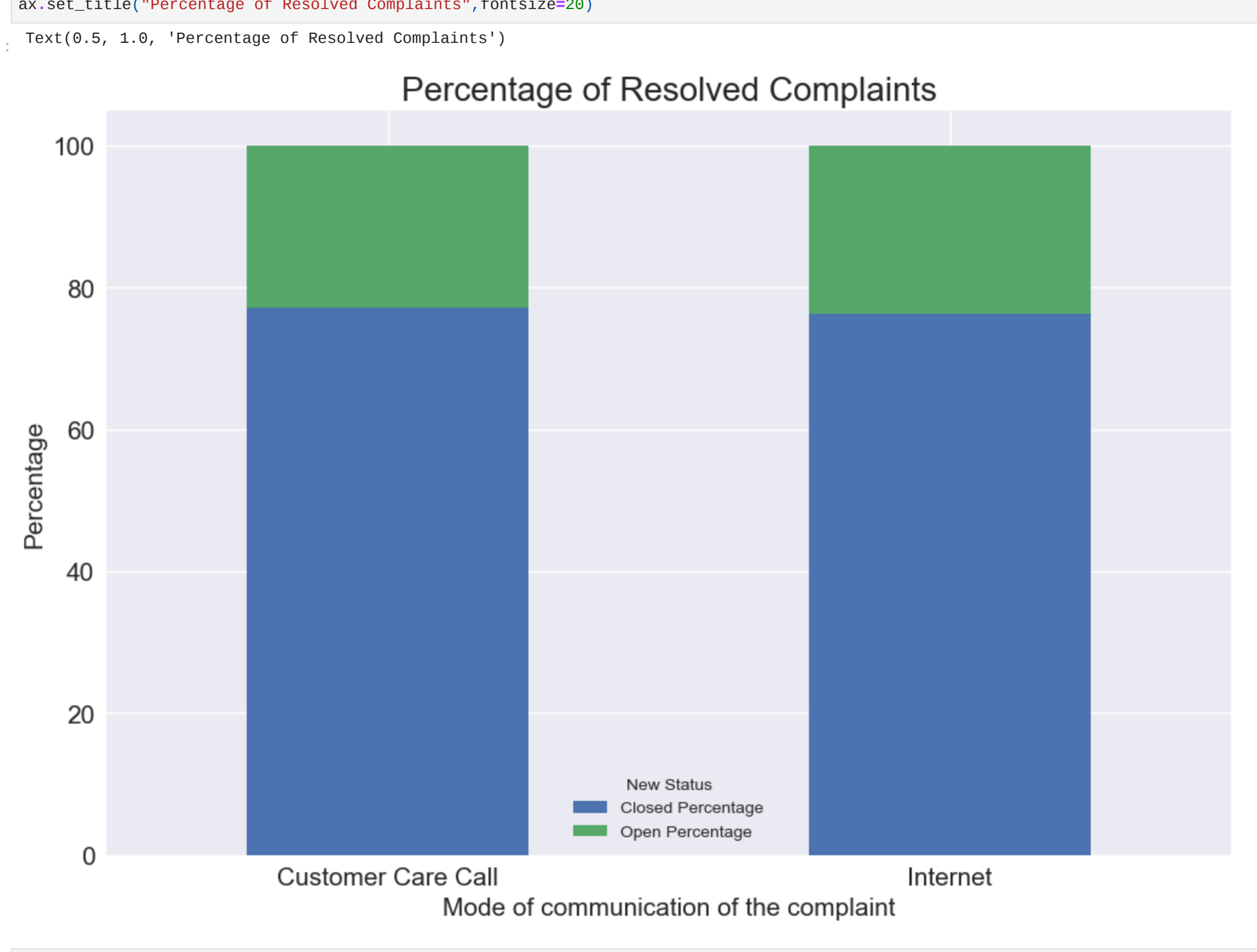
In [215…
```
#Creating Stacked bar chart table using New Status Created
state_chart = data.groupby(['Received Via', 'New Status'])['Received Via'].count().unstack('New Status').fillna(0)

#Creating Closed Percentage column
state_chart['Closed Percentage'] = 100*(state_chart['Closed'] / (state_chart['Closed']+state_chart['Open']))

#Creating Open Percentage
state_chart['Open Percentage'] = 100*(state_chart['Open'] / (state_chart['Closed']+state_chart['Open']))

#required table
state_chart[['Closed Percentage','Open Percentage']]
```

Out[215]:

New Status / Received Via	Closed Percentage	Open Percentage
Customer Care Call	77.211796	22.788204
Internet	76.289593	23.710407

In [216…
```
#Plotting the graph
ax = state_chart[['Closed Percentage','Open Percentage']].plot(kind='bar', stacked=True,rot=0,figsize=(12,8),fontsize=15)
ax.set_xlabel("Mode of communication of the complaint",fontsize=15)
ax.set_ylabel("Percentage",fontsize=15)
ax.set_title("Percentage of Resolved Complaints",fontsize=20)
```

Out[216]:
```
Text(0.5, 1.0, 'Percentage of Resolved Complaints')
```



In [217…
```
# Provide the percentage of complaints resolved till date, which were received through the Internet and customer care calls.
data[data['Status']=='Closed'].groupby('Status')['Received Via'].value_counts(normalize=True)*100
```

Out[217]:
```
Status  Received Via
Closed  Customer Care Call    50.615114
        Internet              49.384886
Name: Received Via, dtype: float64
```

In [218…
```
##Thank you!!!..
```