**Comcast Telecom Consumer Complaints**

DESCRIPTION

Comcast is an American global telecommunication company. The firm has been

Providing terrible customer service. They continue to fall short despite

Repeated promises to improve. Only last month (October 2016) the authority

Fined them a $2.3 million, after receiving over 1000 consumer complaints.

The existing database will serve as a repository of public customer

Complaints filed against Comcast.

It will help to pin down what is wrong with Comcast's customer service.

Data Dictionary

 Ticket #: Ticket number assigned to each complaint

 Customer Complaint: Description of complaint

 Date: Date of complaint

 Time: Time of complaint

 Received Via: Mode of communication of the complaint

 City: Customer city

 State: Customer state

 Zip code: Customer zip

 Status: Status of complaint

 Filing on behalf of someone

**Analysis Task**

To perform these tasks, you can use any of the different Python libraries

such as NumPy, SciPy, Pandas, scikit-learn, matplotlib, and BeautifulSoup.

**Import data into Python environment.**

import os

# Function to Get the current

# working directory

def current\_path():

print("Current working directory before")

print(os.getcwd())

print()

current\_path()

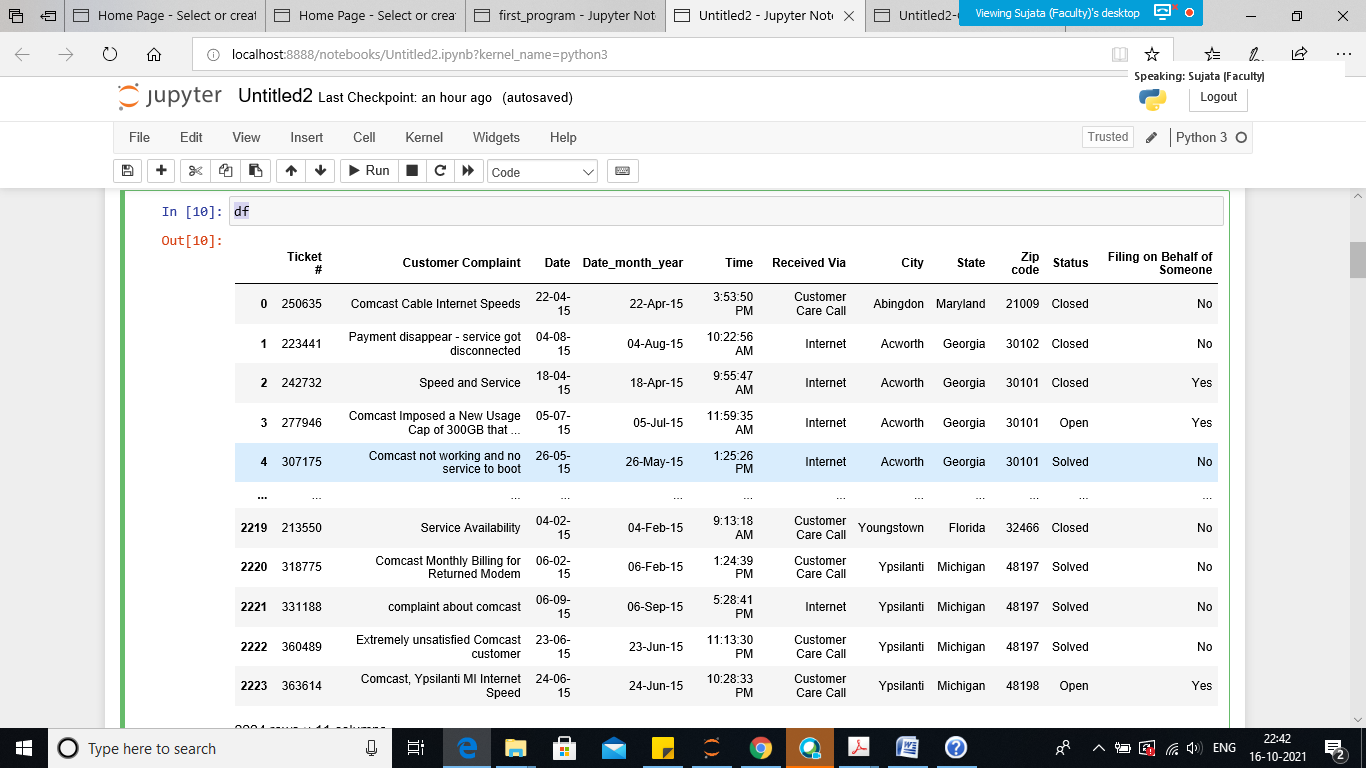
import numpy as np

import pandas as pd

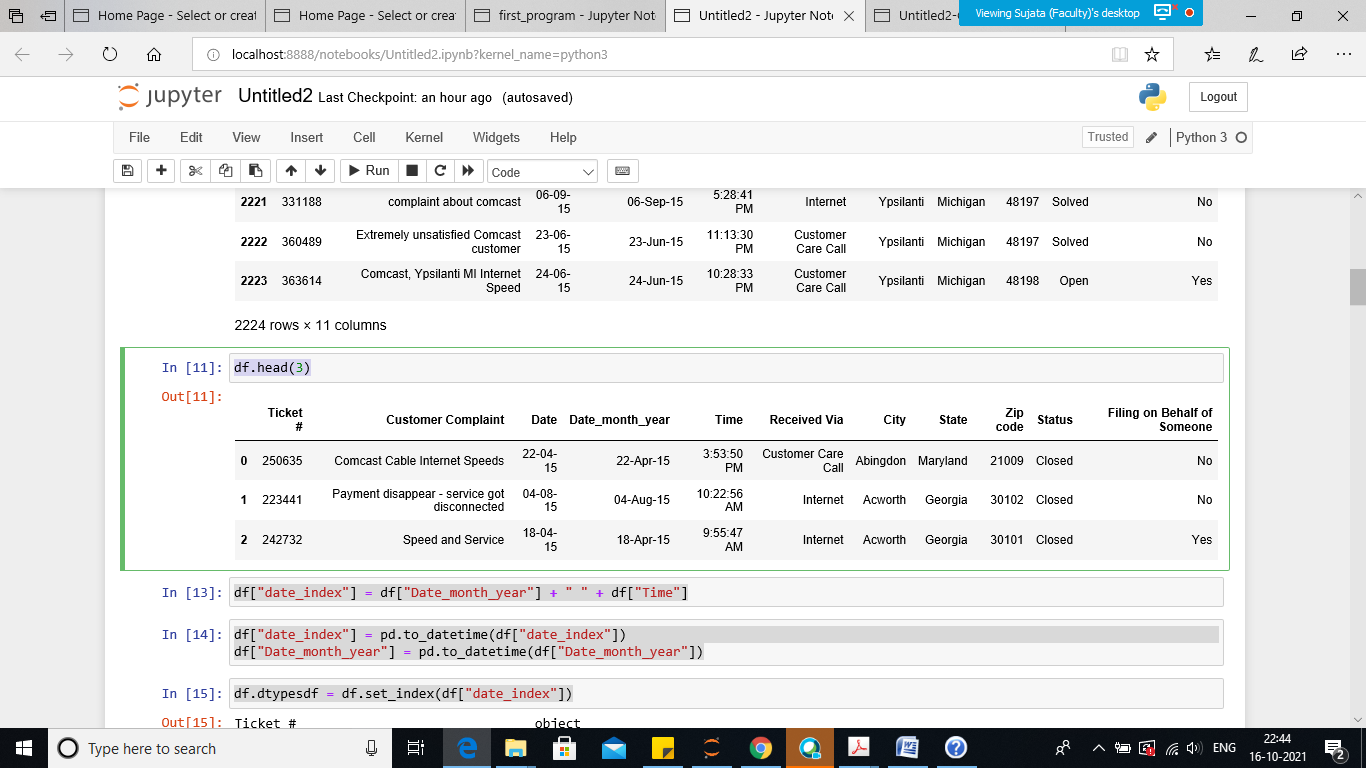
import matplotlib.pyplot as plt

df=pd.read\_csv('E:\\Data science\\Data Sceince with Python\\Project\\Comcast\_telecom\_complaints\_data.csv')

df



df.head(3)



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

Extracting just Month and Year separately from Pandas Date time column.

df["date\_index"] = df["Date\_month\_year"] + " " + df["Time"]

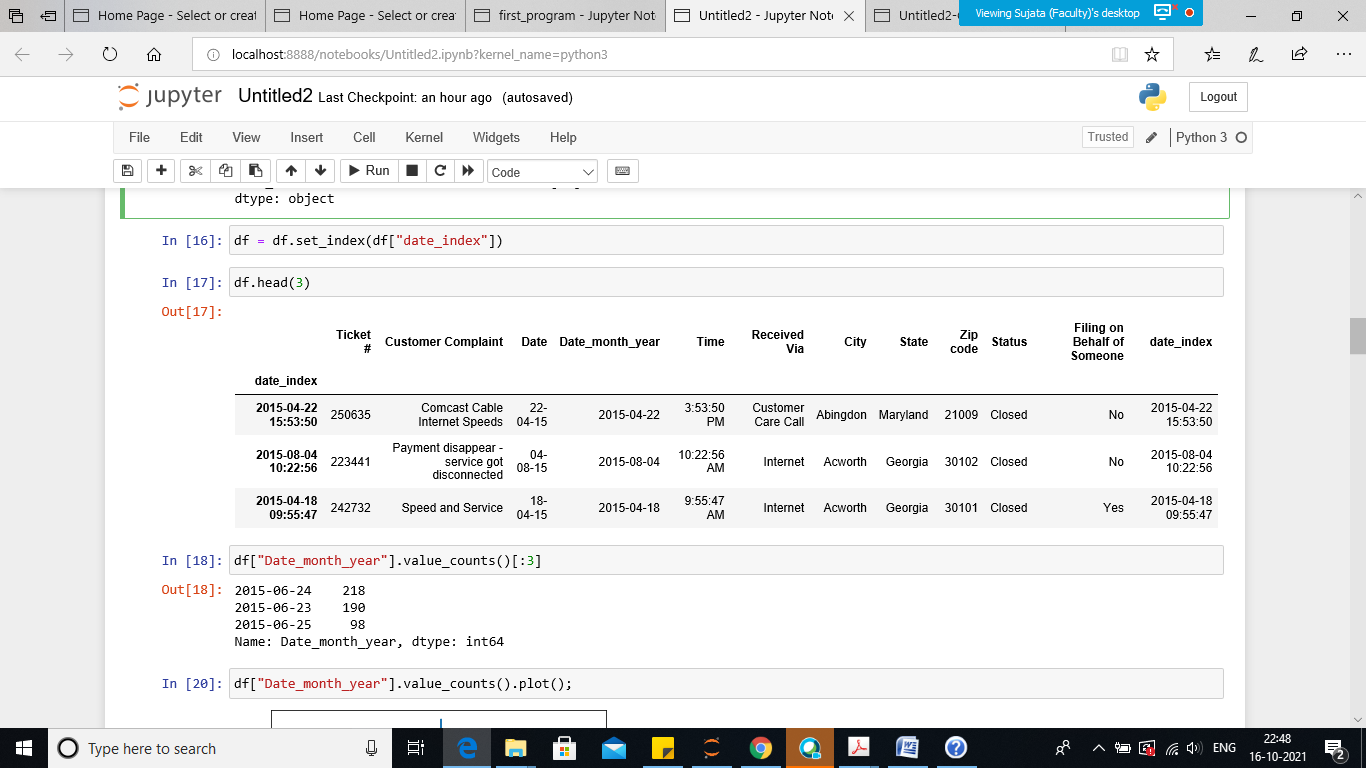
df["date\_index"] = pd.to\_datetime(df["date\_index"])

df["Date\_month\_year"] = pd.to\_datetime(df["Date\_month\_year"])

df.dtypesdf = df.set\_index(df["date\_index"])

Use the header function to view the first 3 records

df.head(3)



df["Date\_month\_year"].value\_counts()[:3]

Out[18]:

2015-06-24 218

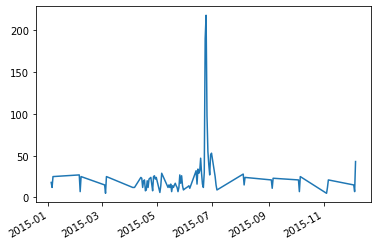
2015-06-23 190

2015-06-25 98

Name: Date\_month\_year, dtype: int64

Now using the plot method

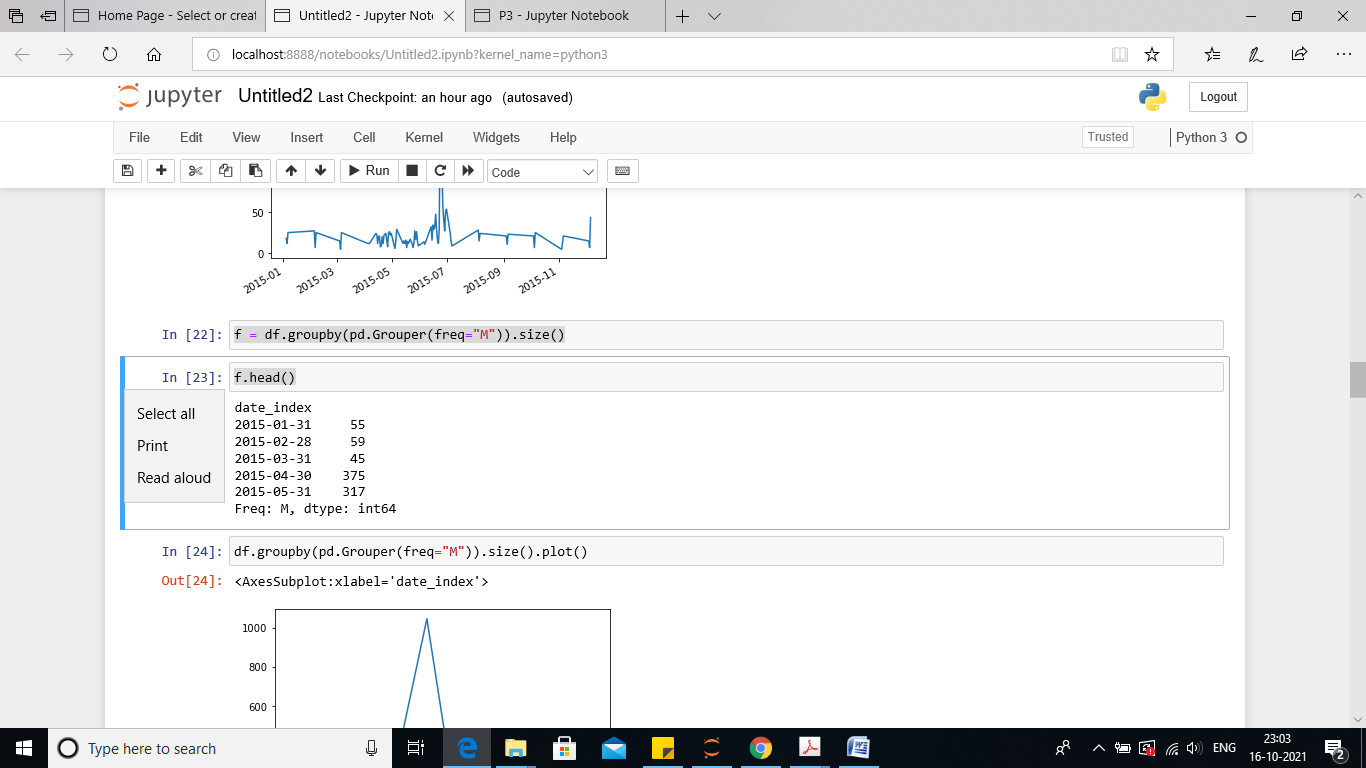
df["Date\_month\_year"].value\_counts().plot();



**Provide a table with the frequency of complaint types.**

f = df.groupby(pd.Grouper(freq="M")).size()

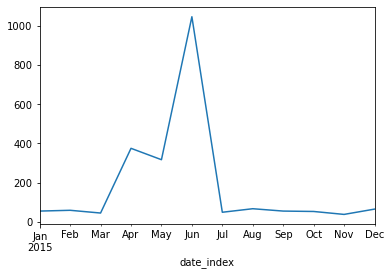
f.head()



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

- 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.

df.groupby(pd.Grouper(freq="M")).size().plot()



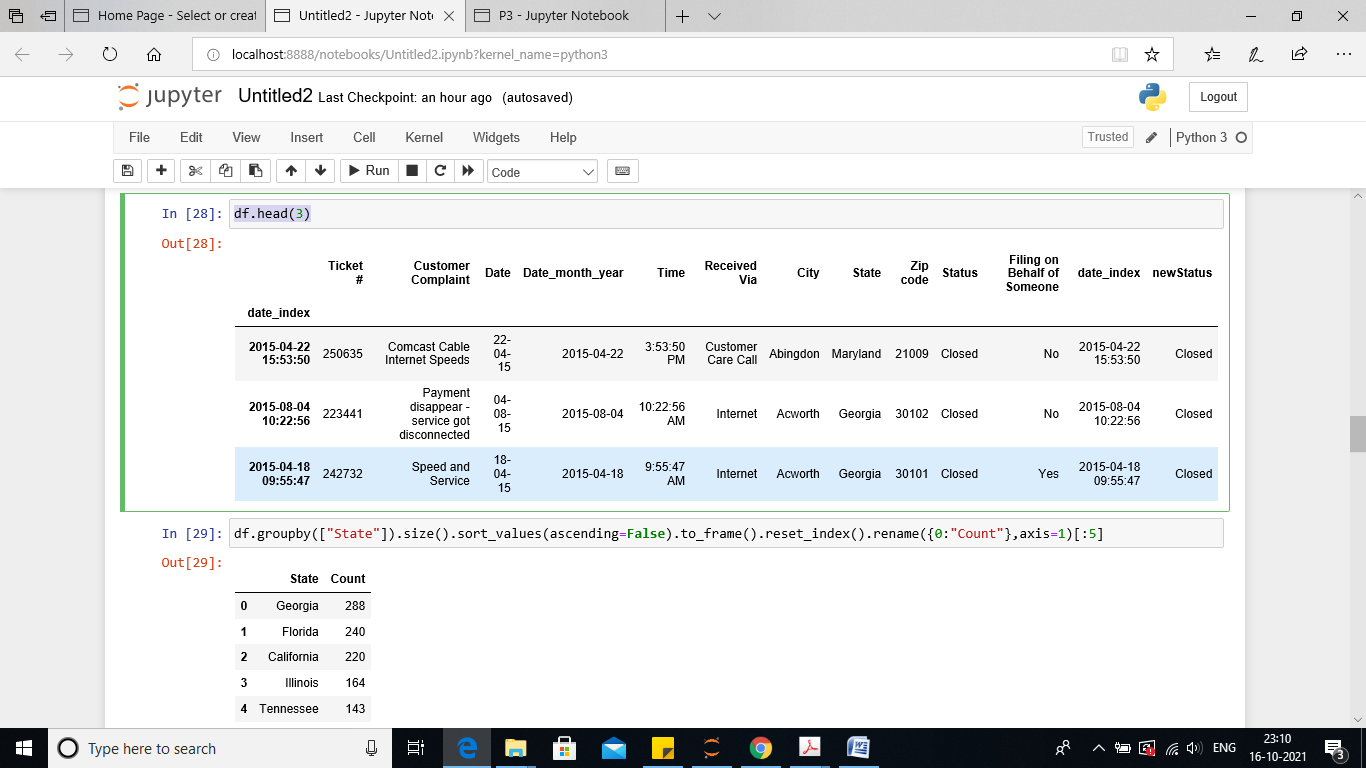
df.Status.unique()

out[25]:

array(['Closed', 'Open', 'Solved', 'Pending'], dtype=object)

df["newStatus"] = ["Open" if Status=="Open" or Status=="Pending" else "Closed" for Status in df["Status"]]

df.head(3)



**Provide state wise status of complaints in a stacked bar chart. Use the**

**categorized variable from Q3. Provide insights on:**

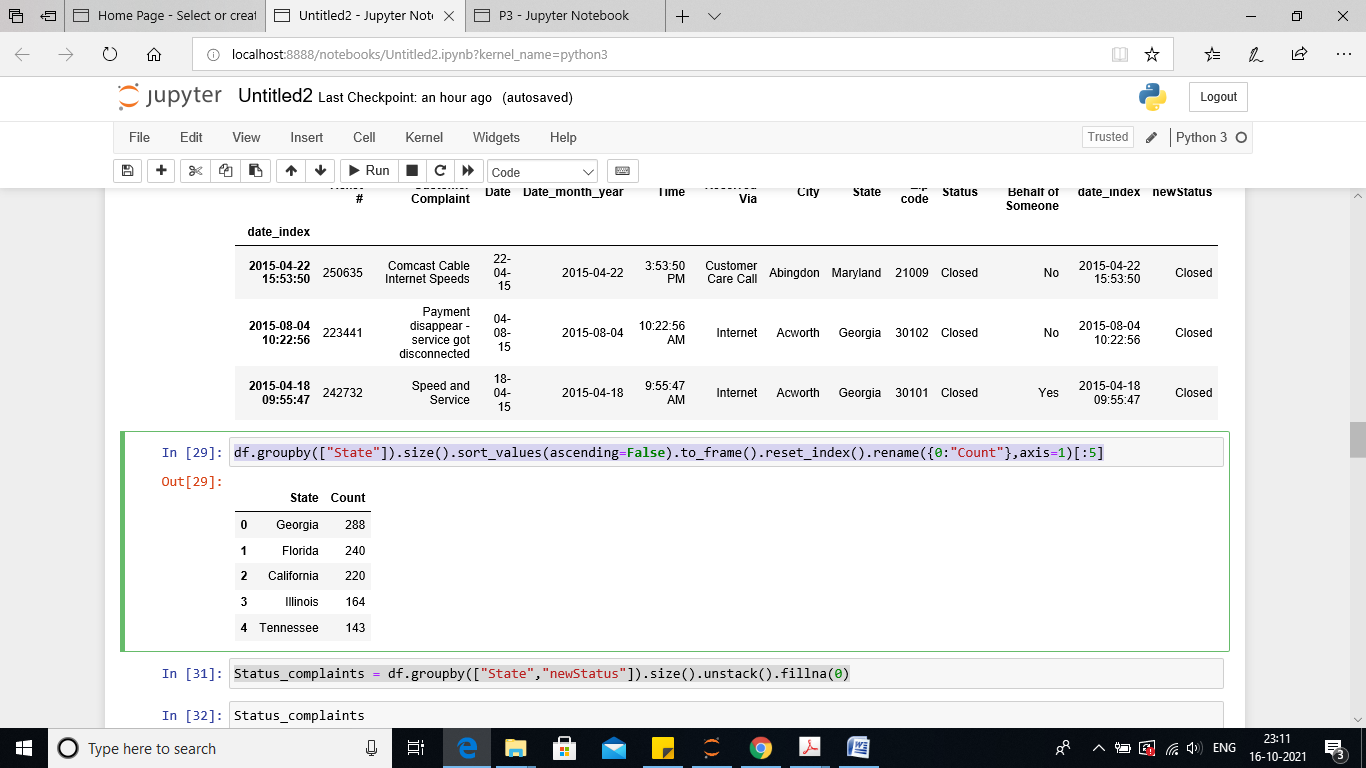
Which state has the maximum complaints:-

So here sorting the states based on the maximum count

First try to sort by State, followed by corresponding size of each and then

Fetch the indexing array from 0 = [:5]

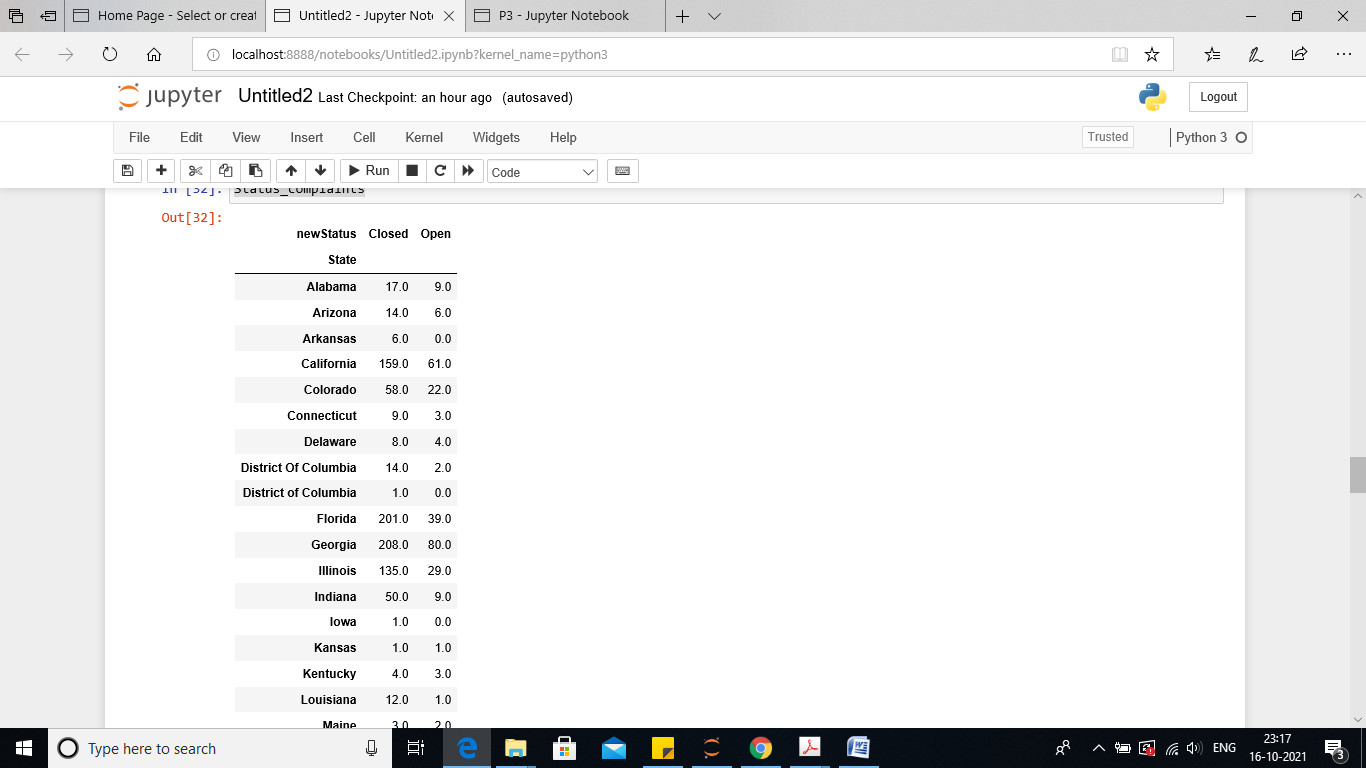
df.groupby(["State"]).size().sort\_values(ascending=False).to\_frame().reset\_index().rename({0:"Count"},axis=1)[:5]



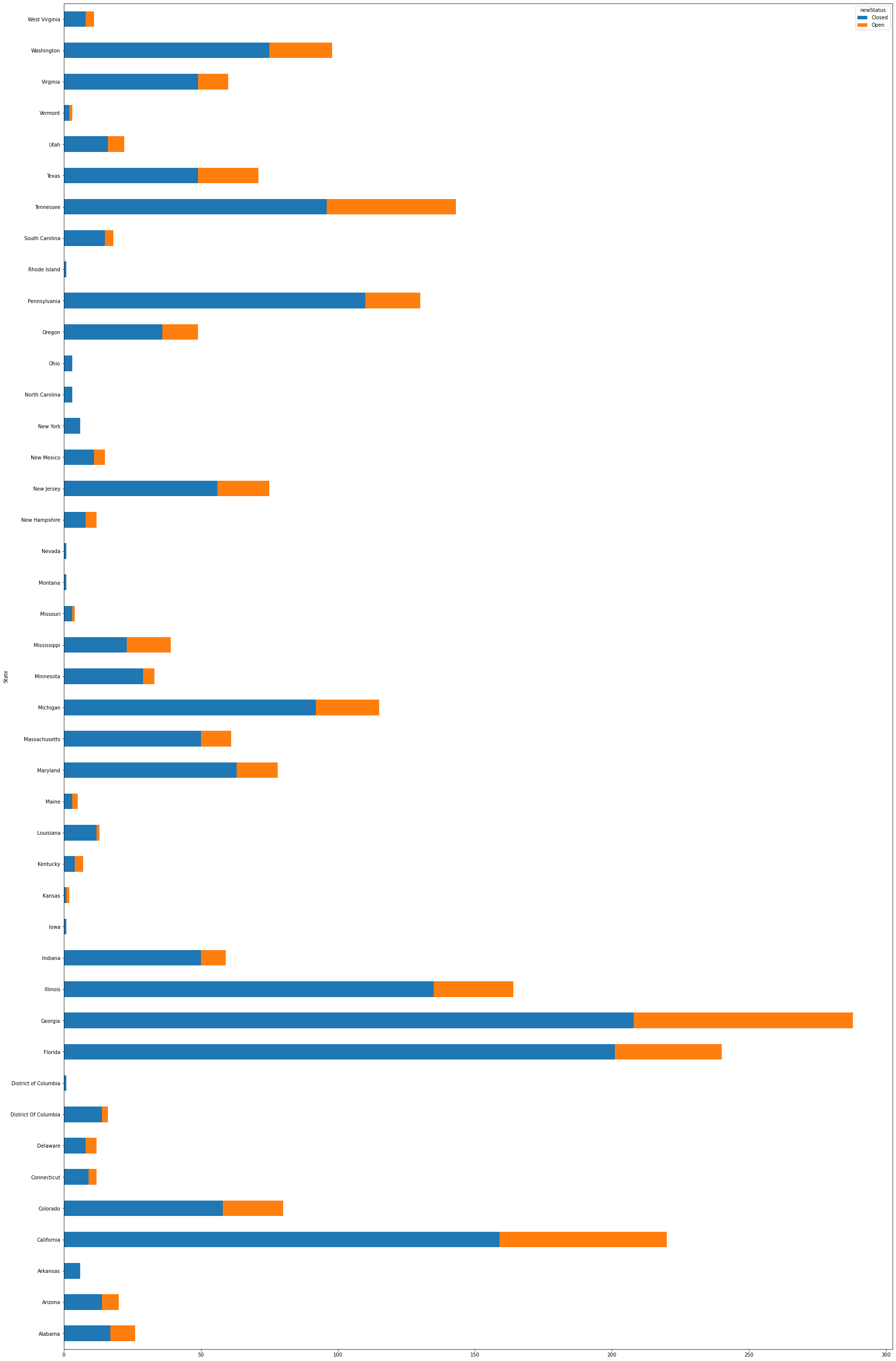
Which state has the highest percentage of unresolved complaints?

Status\_complaints = df.groupby(["State","newStatus"]).size().unstack().fillna(0)

Status\_complaints

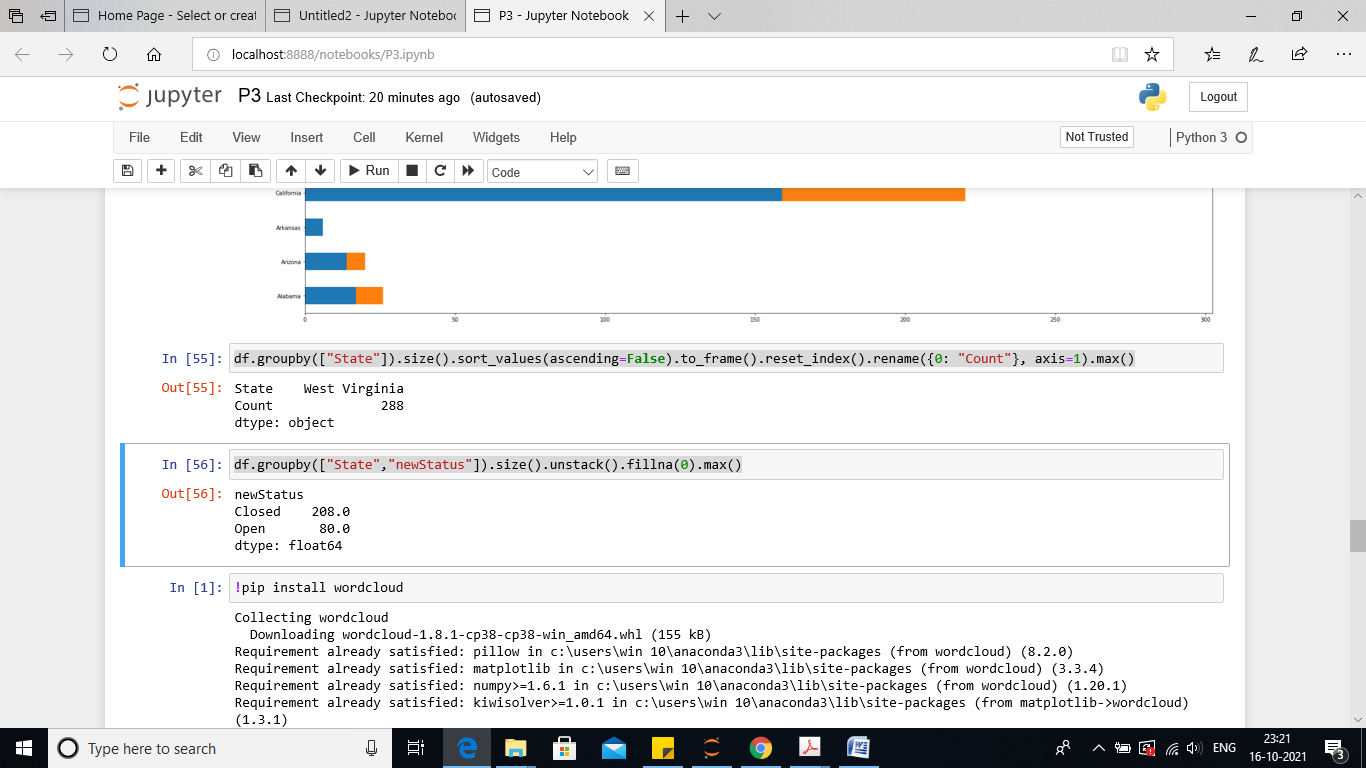


Status\_complaints.plot(kind="barh", figsize=(30,50), stacked=True)



df.groupby(["State"]).size().sort\_values(ascending=False).to\_frame().reset\_index().rename({0: "Count"}, axis=1).max()

df.groupby(["State","newStatus"]).size().unstack().fillna(0).max()



**Provide the percentage of complaints resolved till date, which were received through the Internet and customer care calls.**

def get\_simple\_topic\_percentage(topic):

"""

Returns a percentage of rows that this particular topic is found

in using simple string manipulation. Note: this can have overlaps,

for example if you have two topics, one 'Internet' and one 'Speed',

you will get duplicate findings if the customer has 'Internet Speed'

as their topic.

topic: the customer complaint category entered by the customer.

"""

return df[df['Customer Complaint'].str.contains(topic, case=False)].shape[0]/ len(df['Customer Complaint'])\*100

print('Comcast:', get\_simple\_topic\_percentage('comcast'))

print('Data cap:', get\_simple\_topic\_percentage('data'))

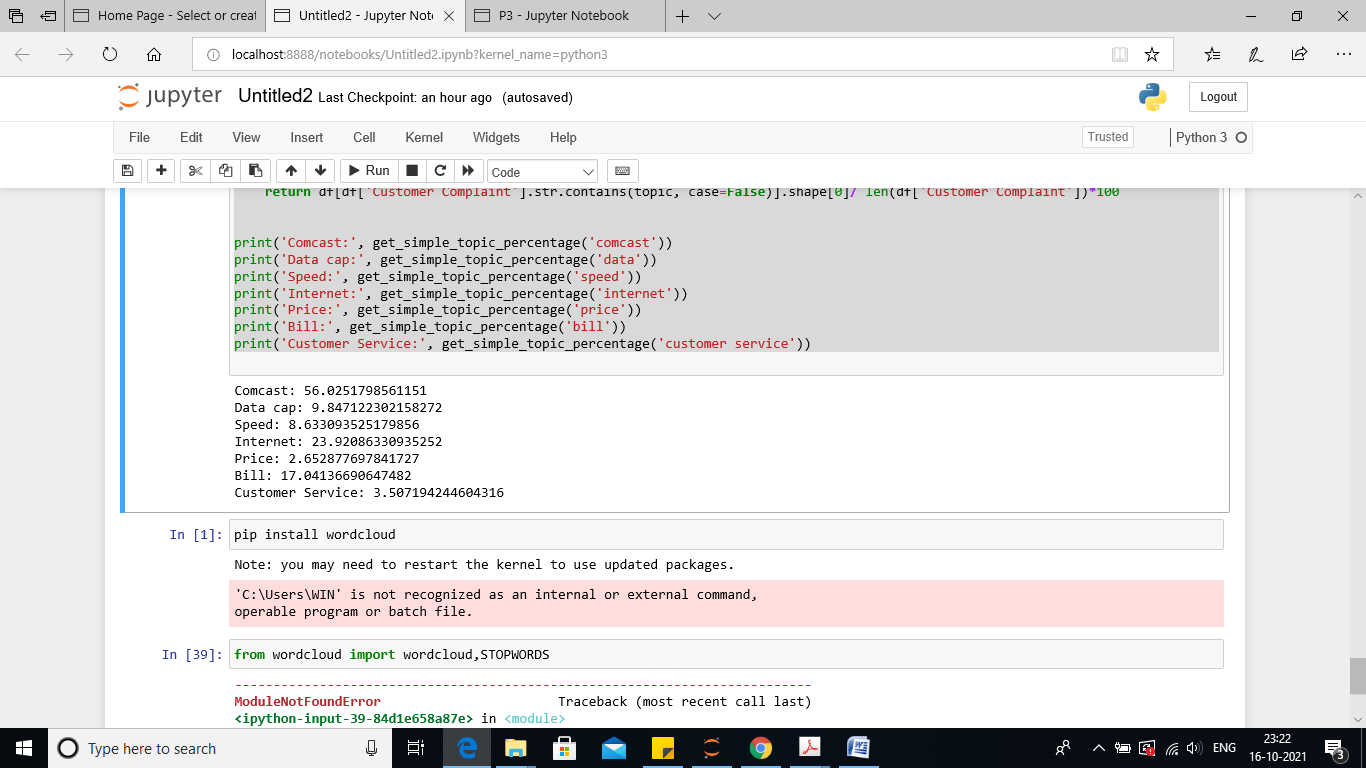
print('Speed:', get\_simple\_topic\_percentage('speed'))

print('Internet:', get\_simple\_topic\_percentage('internet'))

print('Price:', get\_simple\_topic\_percentage('price'))

print('Bill:', get\_simple\_topic\_percentage('bill'))

print('Customer Service:', get\_simple\_topic\_percentage('customer service'))



All defined functions will have return value, so here we are returning the value in % format by multiplying the same to 100.

Then printing the result as displayed in above screen shot.