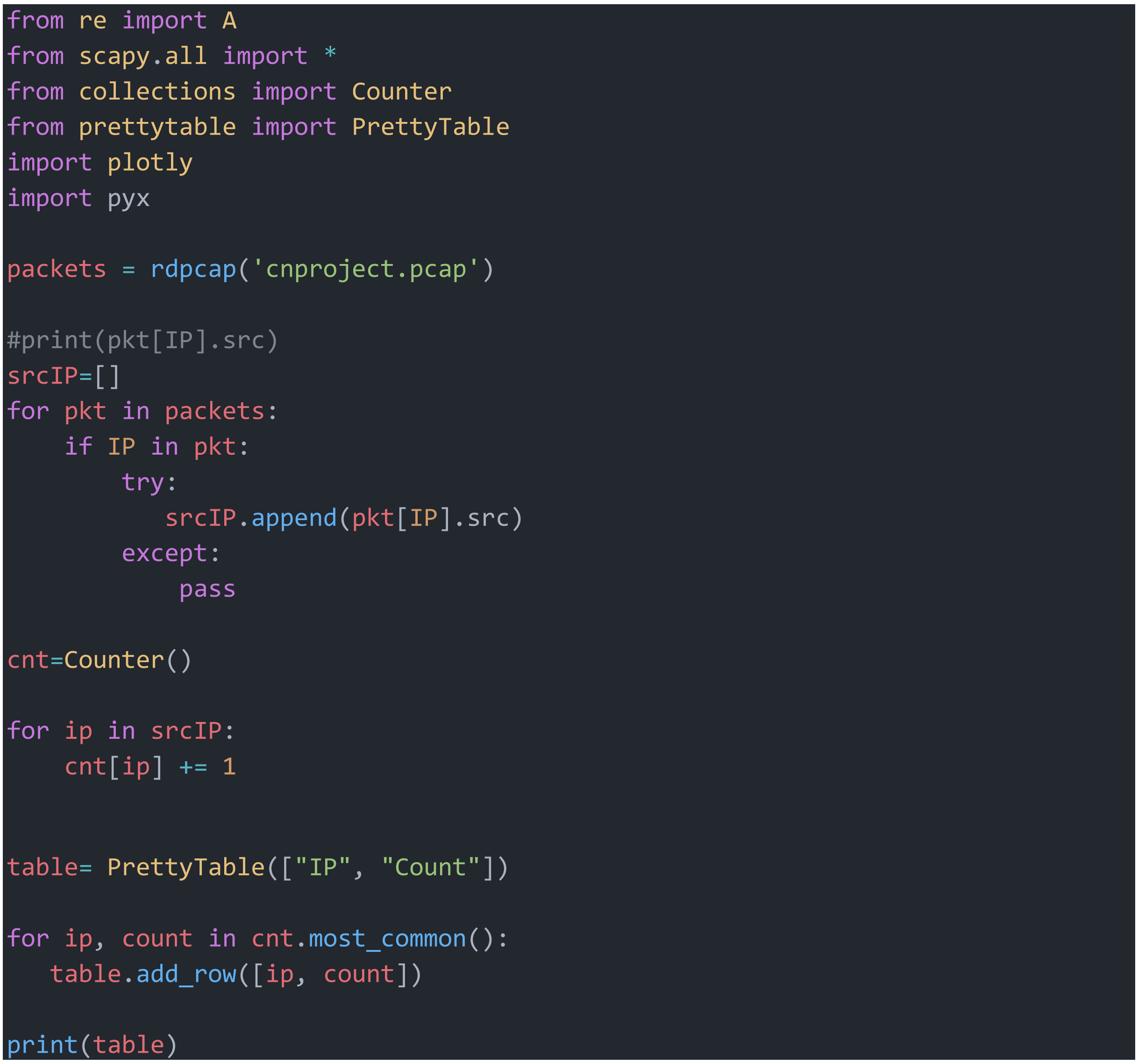
ANALYTICS :

In this part of the project , we have used NYKAA website as our source for the packets where we analyse them and use them to show at what time which port and source destinations have been connected with. Here we have used the dataset of cnproject.pcap. This file consists of the packets which we collected in wireshark when we were browsing through that website.

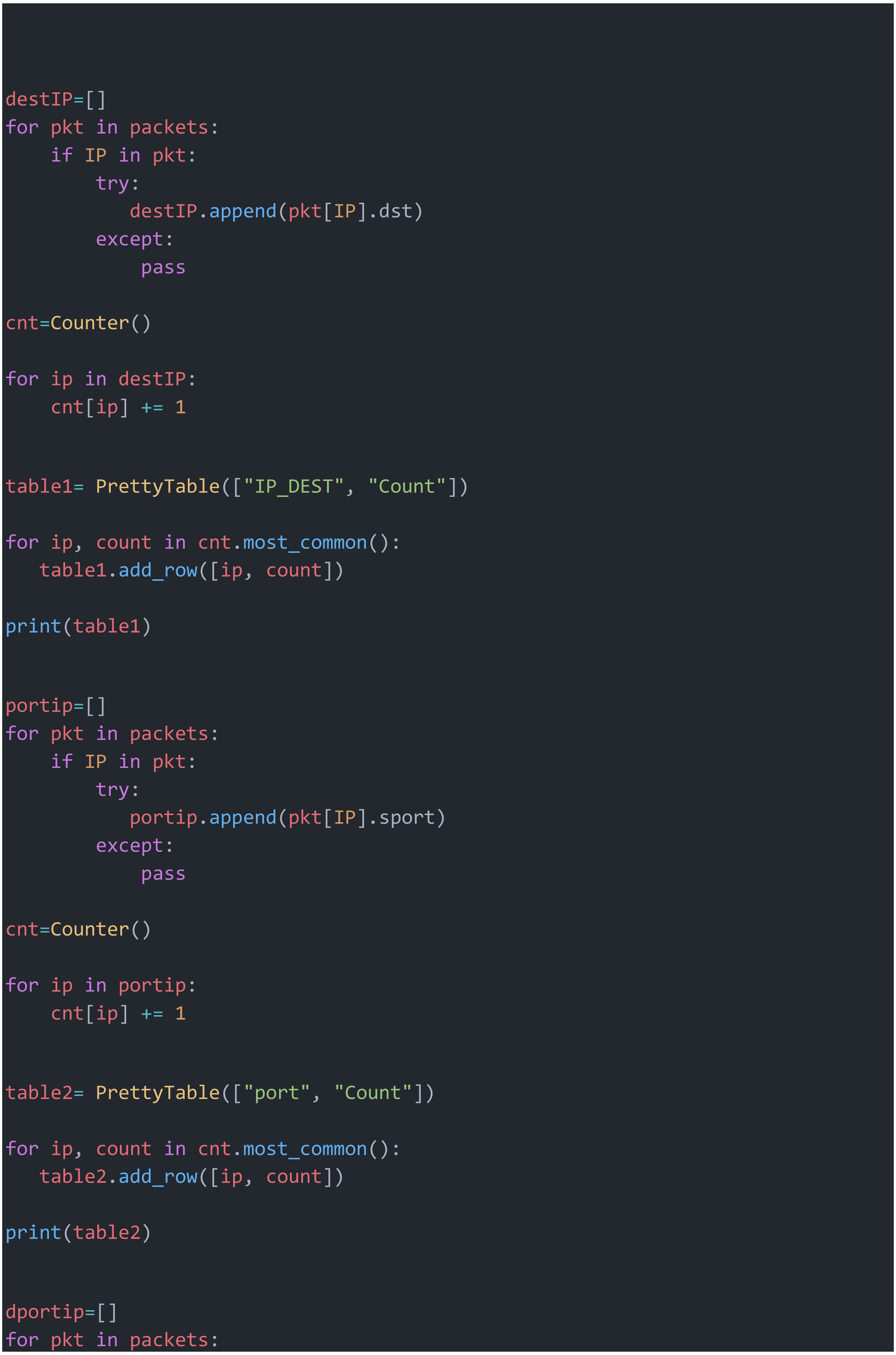
NYKAA ip address : 104.18.1.122

CODE:

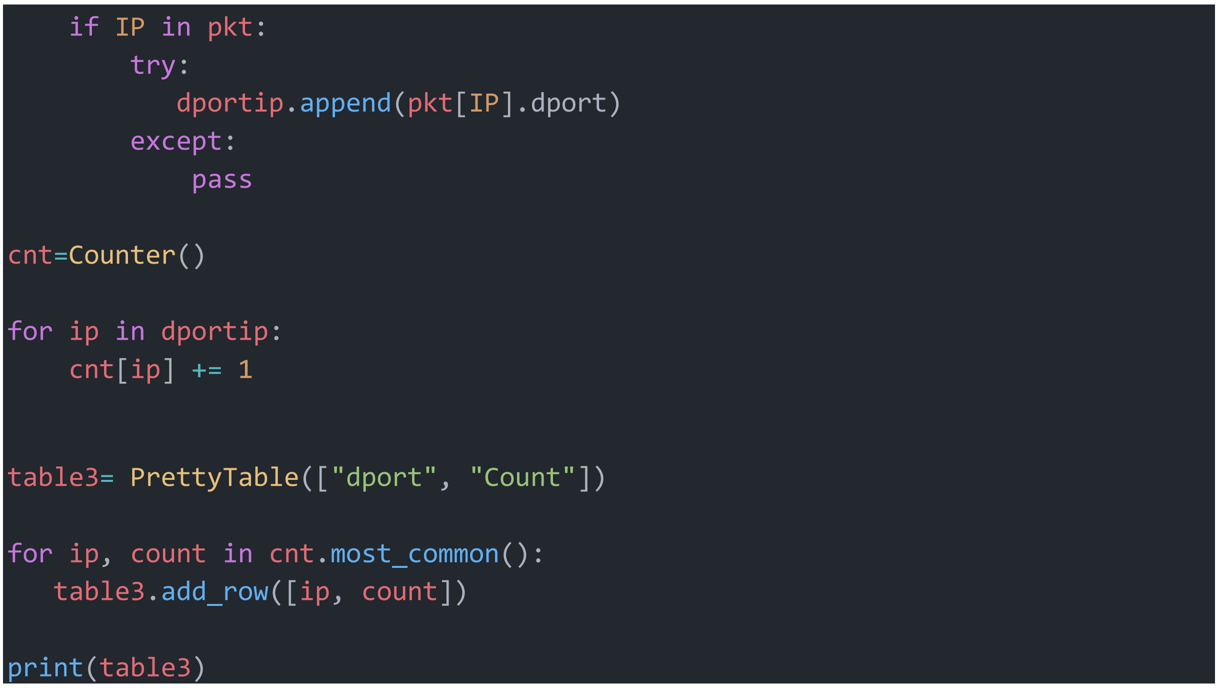
1) In this code we are using scapy module as well as pyx , prettytable and plotly module to plot our packets which we have captured. Here it checks the count of how many times we have visited and made a connection with. Three tables, one with source ip address, other with destination ip address, and another with the port number and its count.



```
from re import A
from scapy.all import *
from collections import Counter
from prettytable import PrettyTable
import plotly
import pyx

packets = rdpcap('cnproject.pcap')

#print(pkt[IP].src)
srcIP=[]
for pkt in packets:
    if IP in pkt:
        try:
           srcIP.append(pkt[IP].src)
        except:
            pass

cnt=Counter()

for ip in srcIP:
    cnt[ip] += 1


table= PrettyTable(["IP", "Count"])

for ip, count in cnt.most_common():
   table.add_row([ip, count])

print(table)
```

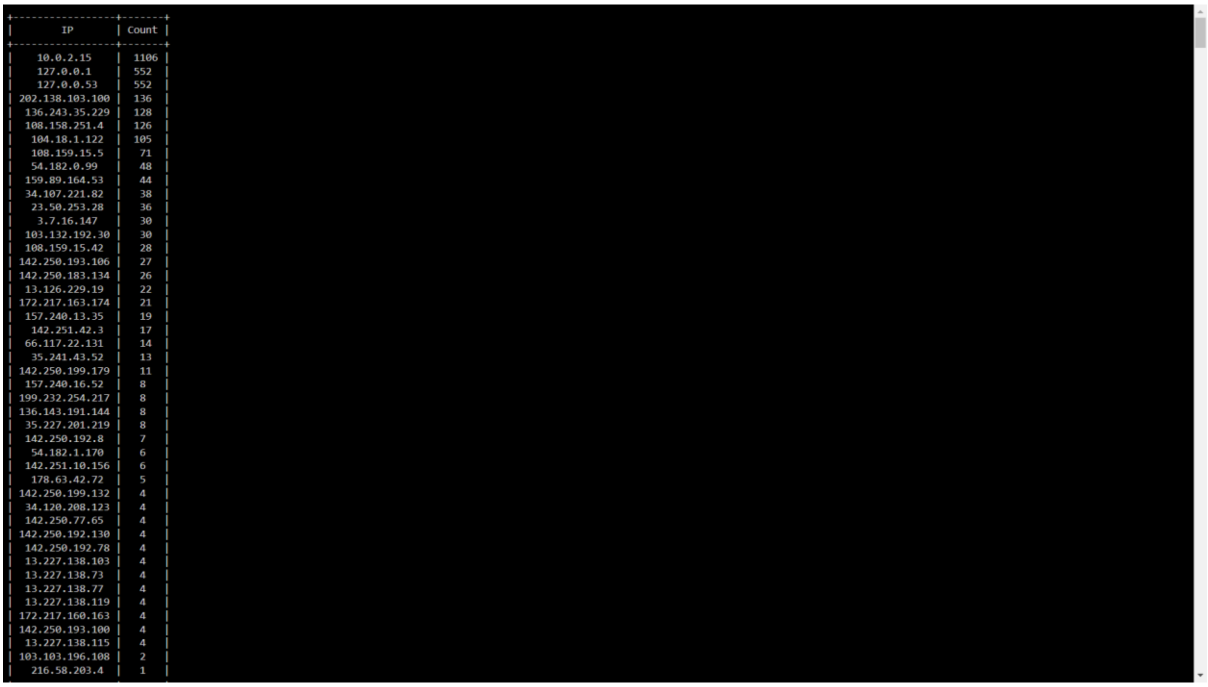


```
destIP=[]
for pkt in packets:
    if IP in pkt:
        try:
           destIP.append(pkt[IP].dst)
        except:
            pass

cnt=Counter()

for ip in destIP:
    cnt[ip] += 1


table1= PrettyTable(["IP_DEST", "Count"])

for ip, count in cnt.most_common():
   table1.add_row([ip, count])

print(table1)


portip=[]
for pkt in packets:
    if IP in pkt:
        try:
           portip.append(pkt[IP].sport)
        except:
            pass

cnt=Counter()

for ip in portip:
    cnt[ip] += 1


table2= PrettyTable(["port", "Count"])

for ip, count in cnt.most_common():
   table2.add_row([ip, count])

print(table2)


dportip=[]
for pkt in packets:
```

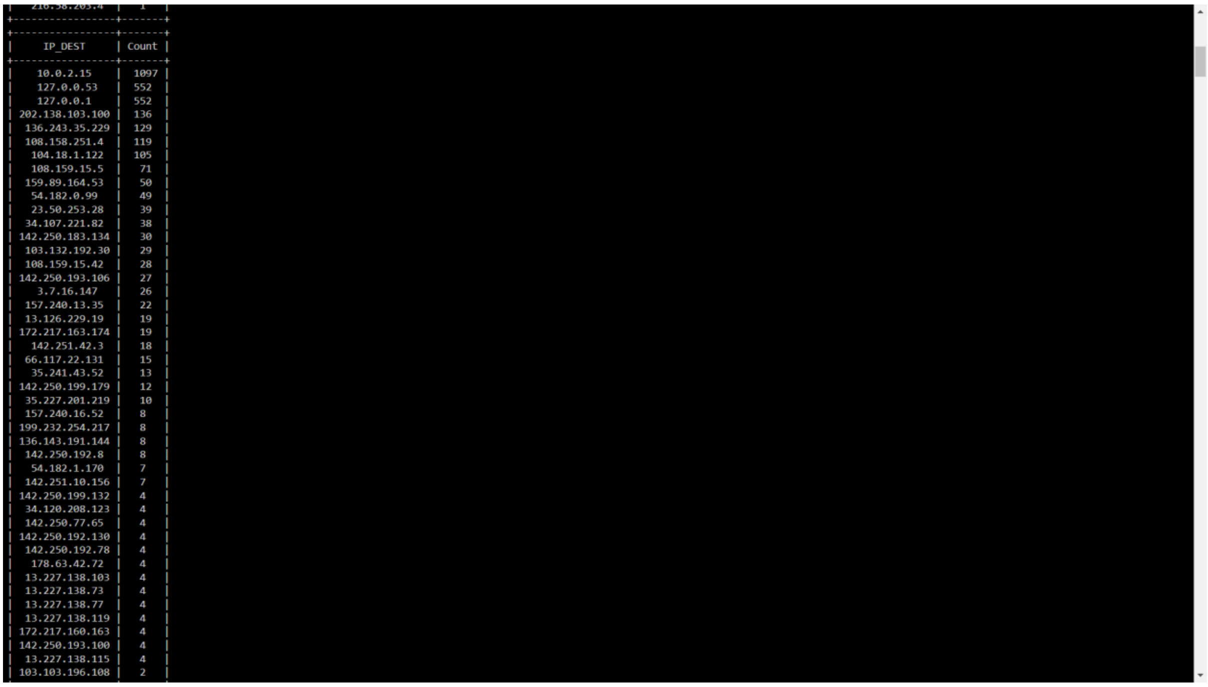


```
    if IP in pkt:
        try:
           dportip.append(pkt[IP].dport)
        except:
            pass

cnt=Counter()

for ip in dportip:
    cnt[ip] += 1


table3= PrettyTable(["dport", "Count"])

for ip, count in cnt.most_common():
   table3.add_row([ip, count])

print(table3)
```

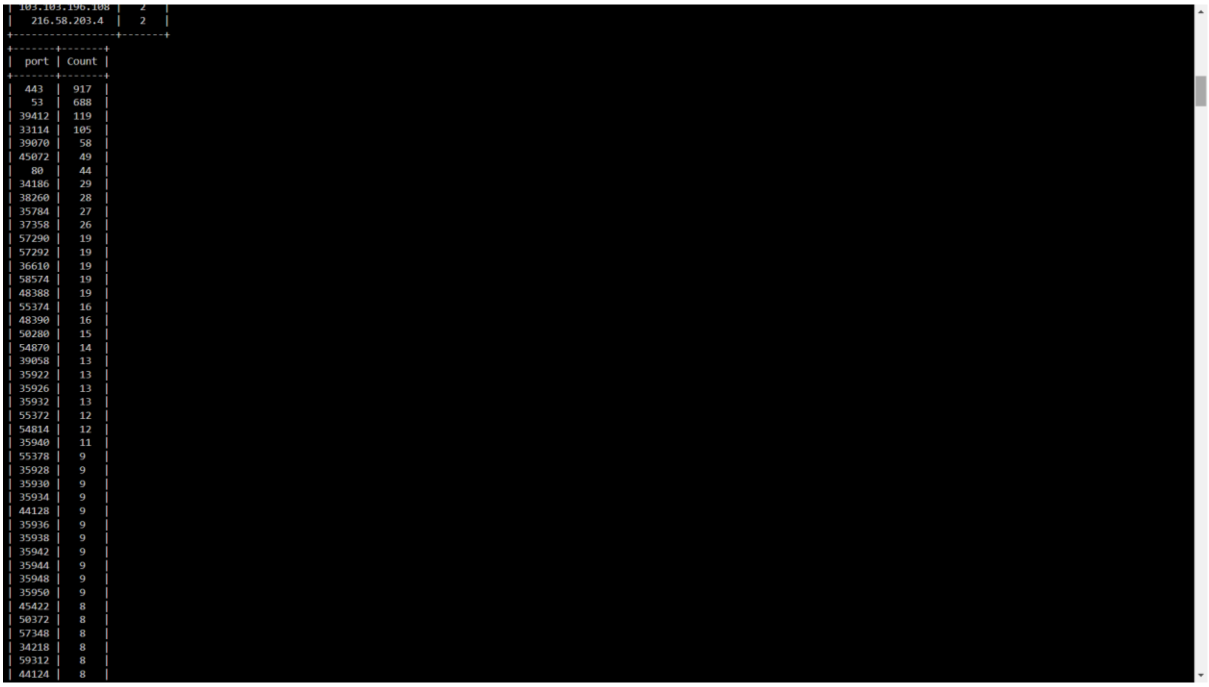
OUTPUT:



```
+-----------------+-------+
|        IP       | Count |
+-----------------+-------+
|    10.0.2.15    |  1106 |
|    127.0.0.1    |  552  |
|    127.0.0.53   |  552  |
| 202.138.103.100 |  136  |
|  136.243.35.229 |  128  |
|  108.158.251.4  |  126  |
|   104.18.1.122  |  105  |
|   108.159.15.5  |   71  |
|   54.182.0.99   |   48  |
|  159.89.164.53  |   44  |
|  34.107.221.82  |   38  |
|   23.50.253.28  |   36  |
|   3.7.16.147    |   30  |
|  103.132.192.30 |   30  |
|  108.159.15.42  |   28  |
| 142.250.193.106 |   27  |
| 142.250.183.134 |   26  |
|  13.126.229.19  |   22  |
| 172.217.163.174 |   21  |
|  157.240.13.35  |   19  |
|   142.251.42.3  |   17  |
|  66.117.22.131  |   14  |
|   35.241.43.52  |   13  |
| 142.250.199.179 |   11  |
|  157.240.16.52  |   8   |
| 199.232.254.217 |   8   |
| 136.143.191.144 |   8   |
|  35.227.201.219 |   8   |
|  142.250.192.8  |   7   |
|   54.182.1.170  |   6   |
|  142.251.10.156 |   6   |
|   178.63.42.72  |   5   |
| 142.250.199.132 |   4   |
|  34.120.208.123 |   4   |
|  142.250.77.65  |   4   |
| 142.250.192.130 |   4   |
|  142.250.192.78 |   4   |
|  13.227.138.103 |   4   |
|  13.227.138.73  |   4   |
|  13.227.138.77  |   4   |
|  13.227.138.119 |   4   |
| 172.217.160.163 |   4   |
| 142.250.193.100 |   4   |
|  13.227.138.115 |   4   |
| 103.103.196.108 |   2   |
|   216.58.203.4  |   1   |
```



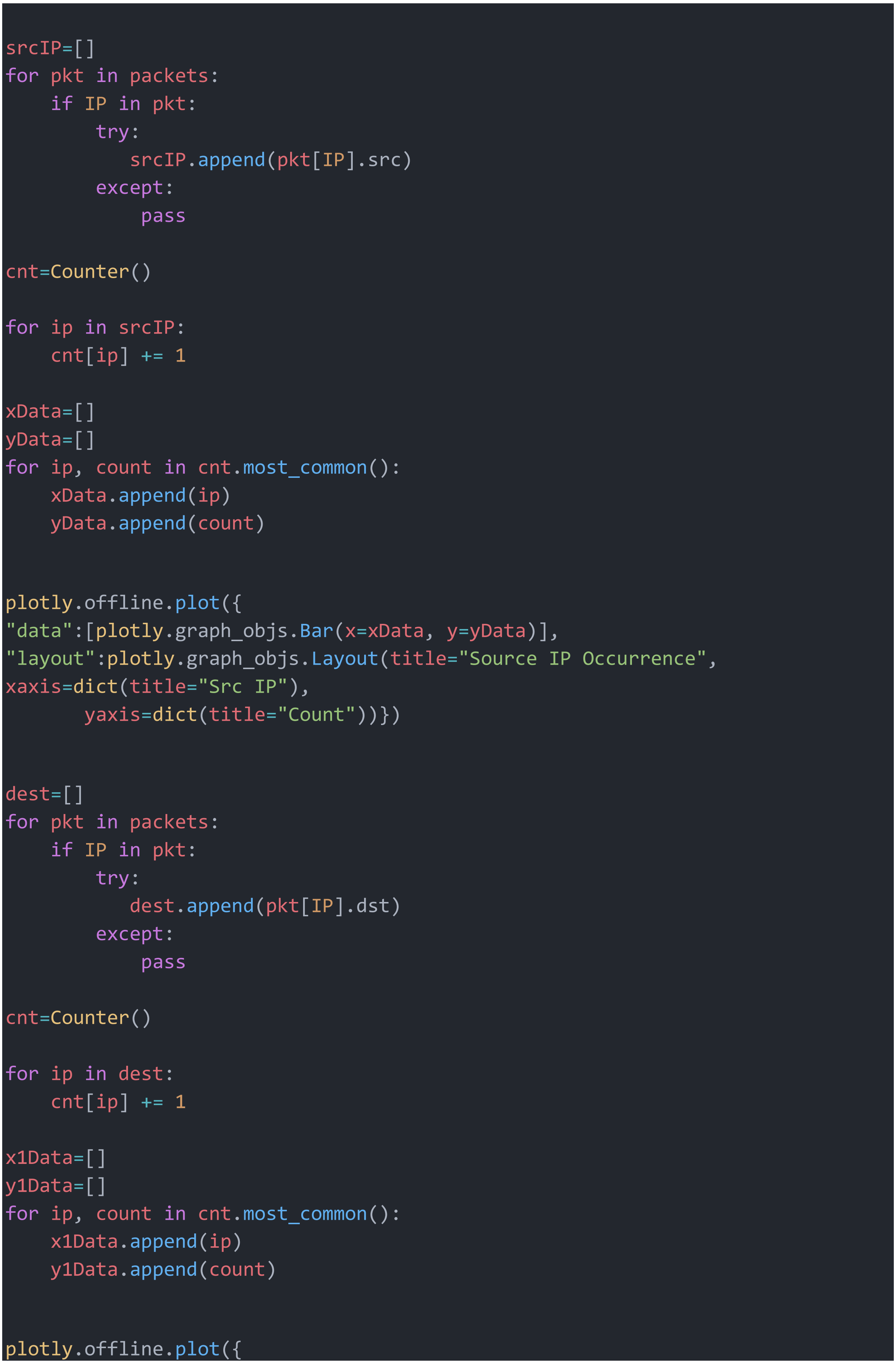
```
|    216.58.203.4   |   1   |
+-------------------+-------+
+-------------------+-------+
|      IP_DEST      | Count |
+-------------------+-------+
|     10.0.2.15     |  1097 |
|     127.0.0.53    |  552  |
|     127.0.0.1     |  552  |
|  202.138.103.100  |  136  |
|   136.243.35.229  |  129  |
|   108.158.251.4   |  119  |
|    104.18.1.122   |  105  |
|    108.159.15.5   |   71  |
|   159.89.164.53   |   50  |
|    54.182.0.99    |   49  |
|    23.50.253.28   |   39  |
|   34.107.221.82   |   38  |
|  142.250.183.134  |   30  |
|   103.132.192.30  |   29  |
|   108.159.15.42   |   28  |
|  142.250.193.106  |   27  |
|     3.7.16.147    |   26  |
|   157.240.13.35   |   22  |
|   13.126.229.19   |   19  |
|  172.217.163.174  |   19  |
|    142.251.42.3   |   18  |
|   66.117.22.131   |   15  |
|    35.241.43.52   |   13  |
|  142.250.199.179  |   12  |
|   35.227.201.219  |   10  |
|   157.240.16.52   |   8   |
|  199.232.254.217  |   8   |
|  136.143.191.144  |   8   |
|   142.250.192.8   |   8   |
|    54.182.1.170   |   7   |
|   142.251.10.156  |   7   |
|  142.250.199.132  |   4   |
|   34.120.208.123  |   4   |
|   142.250.77.65   |   4   |
|  142.250.192.130  |   4   |
|   142.250.192.78  |   4   |
|    178.63.42.72   |   4   |
|   13.227.138.103  |   4   |
|   13.227.138.73   |   4   |
|   13.227.138.77   |   4   |
|   13.227.138.119  |   4   |
|  172.217.160.163  |   4   |
|  142.250.193.100  |   4   |
|   13.227.138.115  |   4   |
|  103.103.196.108  |   2   |
```



```
|  103.103.196.108  |   2   |
|    216.58.203.4   |   2   |
+-------------------+-------+
+-------+-------+
|  port | Count |
+-------+-------+
|  443  |  917  |
|   53  |  688  |
| 39412 |  119  |
| 33114 |  105  |
| 39070 |   58  |
| 45072 |   49  |
|   80  |   44  |
| 34186 |   29  |
| 38260 |   28  |
| 35784 |   27  |
| 37358 |   26  |
| 57290 |   19  |
| 57292 |   19  |
| 36610 |   19  |
| 58574 |   19  |
| 48388 |   19  |
| 55374 |   16  |
| 48390 |   16  |
| 50280 |   15  |
| 54870 |   14  |
| 39058 |   13  |
| 35922 |   13  |
| 35926 |   13  |
| 35932 |   13  |
| 55372 |   12  |
| 54814 |   12  |
| 35940 |   11  |
| 55378 |   9   |
| 35928 |   9   |
| 35930 |   9   |
| 35934 |   9   |
| 44128 |   9   |
| 35936 |   9   |
| 35938 |   9   |
| 35942 |   9   |
| 35944 |   9   |
| 35948 |   9   |
| 35950 |   9   |
| 45422 |   8   |
| 50372 |   8   |
| 57348 |   8   |
| 34218 |   8   |
| 59312 |   8   |
| 44124 |   8   |
```

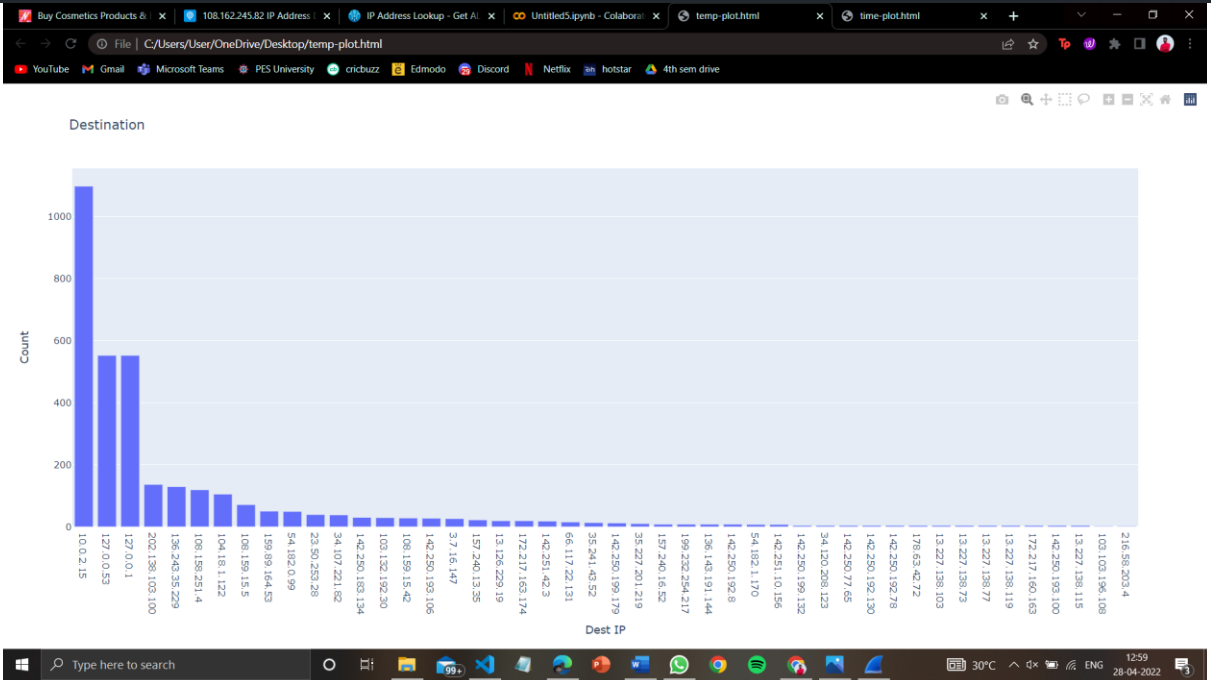
2) Here we have collected the information about the packets which we colleceted and used them to create a plot where we know how much and what peek destination address we reached to. This helps us to classify the information about the addresses.

```
from scapy.all import *
from collections import Counter
import plotly

packets = rdpcap('cnproject.pcap')
```



```
srcIP=[]
for pkt in packets:
    if IP in pkt:
        try:
           srcIP.append(pkt[IP].src)
        except:
            pass

cnt=Counter()

for ip in srcIP:
    cnt[ip] += 1

xData=[]
yData=[]
for ip, count in cnt.most_common():
    xData.append(ip)
    yData.append(count)


plotly.offline.plot({
"data":[plotly.graph_objs.Bar(x=xData, y=yData)],
"layout":plotly.graph_objs.Layout(title="Source IP Occurrence",
xaxis=dict(title="Src IP"),
       yaxis=dict(title="Count"))})


dest=[]
for pkt in packets:
    if IP in pkt:
        try:
           dest.append(pkt[IP].dst)
        except:
            pass

cnt=Counter()

for ip in dest:
    cnt[ip] += 1

x1Data=[]
y1Data=[]
for ip, count in cnt.most_common():
    x1Data.append(ip)
    y1Data.append(count)


plotly.offline.plot({
```

```
"data":[plotly.graph_objs.Bar(x=x1Data, y=y1Data)],
"layout":plotly.graph_objs.Layout(title="Destination",
xaxis=dict(title="Dest IP"),
       yaxis=dict(title="Count"))})
```

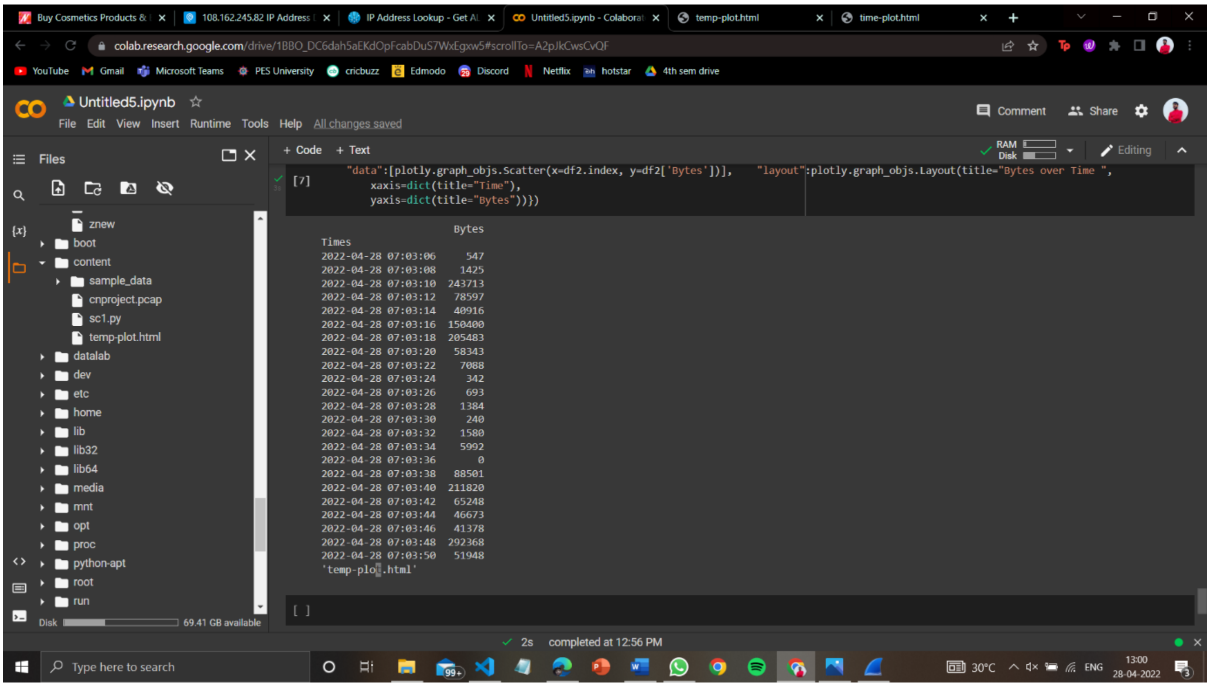


3) In the similar manner as 2, we have used the time plot, to know at what time we have implemented it, and how much it has covered in a certain period.

```
from scapy.all import *
import plotly
from datetime import datetime
import pandas as pd

packets = rdpcap('cnproject.pcap')#Lists to hold packet info
pktBytes=[]
pktTimes=[]#Read each packet and append to the lists.
for pkt in packets:
    if IP in pkt:
        try:
            pktBytes.append(pkt[IP].len)           #First we need to covert
Epoch time to a datetime
            pktTime=datetime.fromtimestamp(pkt.time)
            #Then convert to a format we like
            pktTimes.append(pktTime.strftime("%Y-%m-%d %H:%M:%S.%f"))

        except:
            pass

#This converts list to series
```

```
bytes = pd.Series(pktBytes).astype(int)

#Convert the timestamp list to a pd date_time
times = pd.to_datetime(pd.Series(pktTimes).astype(str),  errors='coerce')

#Create the dataframe
df  = pd.DataFrame({"Bytes": bytes, "Times":times})

#set the date from a range to an timestamp
df = df.set_index('Times')

#Create a new dataframe of 2 second sums to pass to plotly
df2=df.resample('2S').sum()
print(df2)

#Create the graph
plotly.offline.plot({
    "data":[plotly.graph_objs.Scatter(x=df2.index,
y=df2['Bytes'])],    "layout":plotly.graph_objs.Layout(title="Bytes over Time
",
        xaxis=dict(title="Time"),
        yaxis=dict(title="Bytes"))})
```

OUTPUT:

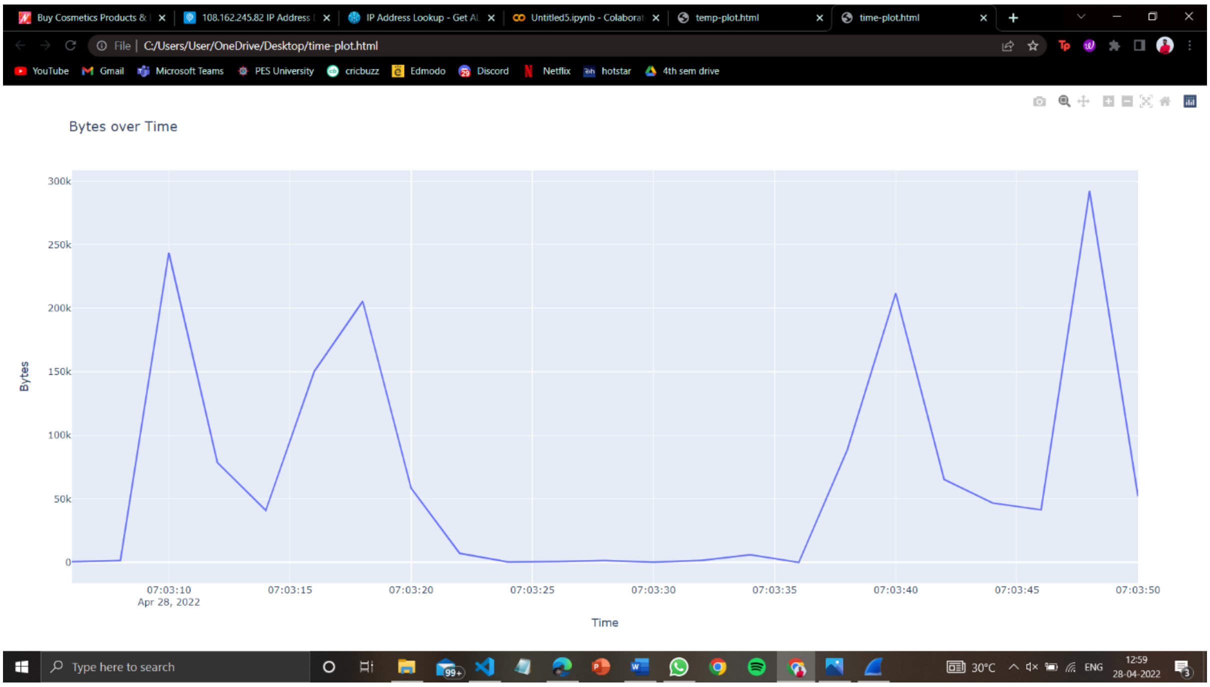


```
[7]        "data":[plotly.graph_objs.Scatter(x=df2.index, y=df2['Bytes'])],    "layout":plotly.graph_objs.Layout(title="Bytes over Time ",
              xaxis=dict(title="Time"),
              yaxis=dict(title="Bytes"))})
```

```
                     Bytes
Times
2022-04-28 07:03:06     547
2022-04-28 07:03:08    1425
2022-04-28 07:03:10  243713
2022-04-28 07:03:12   78597
2022-04-28 07:03:14   40916
2022-04-28 07:03:16  150400
2022-04-28 07:03:18  205483
2022-04-28 07:03:20   58343
2022-04-28 07:03:22    7088
2022-04-28 07:03:24     342
2022-04-28 07:03:26     693
2022-04-28 07:03:28    1384
2022-04-28 07:03:30     240
2022-04-28 07:03:32    1580
2022-04-28 07:03:34    5992
2022-04-28 07:03:36       0
2022-04-28 07:03:38   88501
2022-04-28 07:03:40  211820
2022-04-28 07:03:42   65248
2022-04-28 07:03:44   46673
2022-04-28 07:03:46   41378
2022-04-28 07:03:48  292368
2022-04-28 07:03:50   51948
'temp-plot.html'
```

```
[ ]
```

✓ 2s completed at 12:56 PM


Buy Cosmetics Products &
108.162.245.82 IP Address
IP Address Lookup - Get A
Untitled5.ipynb - Colabora
temp-plot.html
time-plot.html
File | C:/Users/User/OneDrive/Desktop/time-plot.html
YouTube
Gmail
Microsoft Teams
PES University
cricbuzz
Edmodo
Discord
Netflix
hotstar
4th sem drive
Bytes over Time
Bytes
300k
250k
200k
150k
100k
50k
0
07:03:10
Apr 28, 2022
07:03:15
07:03:20
07:03:25
07:03:30
07:03:35
07:03:40
07:03:45
07:03:50
Time
Type here to search
30°C
ENG
12:59
28-04-2022