Craig Boger
01/2020
CAP4770

QUESTION 1

Question 1

In [33]:
```
#Get data and group.

#Practice creating hour_grouped using lambda function.
#Grab all rows from original log with specific date.
q1_may1_df = log_df[log_df['Date'] == '01/May/1998']
#Construct datetime column.  Raster over all rows and add datetime.
#.apply will apply a function to all values in a pandas series.
#axis = 1 to apply "row-wise" https://stackoverflow.com/questions/22149584/what-does-axis-in-pandas-mean
q1_may1_df.loc[:,('DateTime')] = pd.to_datetime(q1_may1_df.apply(lambda row: row['Date'] + ' ' + row['Time'], axis = 1))
#group by the hour of the row
q1_hour_grouped = q1_may1_df.groupby(lambda x: q1_may1_df['DateTime'][x].hour)
```

In [34]:
```
#Output groupig size to check with earlier in the lab.
q1_hour_grouped.size()
```

Out[34]:
```
0      6569
1      6103
2      6072
3      6625
4      6019
5      4733
6      4995
7      5094
8      6460
9      7892
10     7465
11     7893
12    10127
13    10225
14    12040
15    12256
16    13367
17    11494
18    11515
19    10386
20     9363
21     6610
dtype: int64
```

In [36]:
```
#Narrow down number of unique users and output lists.
uniqueUsers = q1_hour_grouped['ClientID'].nunique()
uniqueUsers
```

Out[36]:
```
0      830
1      812
2      772
3      815
4      789
5      657
6      716
7      754
8      871
9      993
10     942
11     998
12    1165
13    1256
14    1416
15    1397
16    1501
17    1347
18    1308
19    1212
20    1161
21     925
Name: ClientID, dtype: int64
```

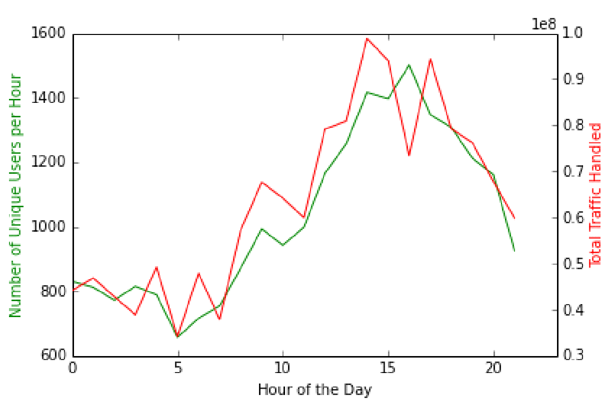
Craig Boger
01/2020
CAP4770

In [37]:
```
#Calculate the size of the traffic on the site every hour.
q1_hour_grouped_traffic = q1_hour_grouped['Size'].sum()
q1_hour_grouped_traffic
```

Out[37]:
```
0     44166352.0
1     46857868.0
2     42803283.0
3     38868040.0
4     49190470.0
5     34184105.0
6     47877742.0
7     37838488.0
8     57224306.0
9     67645841.0
10    64193518.0
11    59961757.0
12    79150391.0
13    80907946.0
14    98825640.0
15    94044070.0
16    73413868.0
17    94389754.0
18    79264404.0
19    76209823.0
20    67784666.0
21    59834046.0
Name: Size, dtype: float64
```

In [38]:
```
#Generating double line plot with x-axis of hour, y1 axis of traffic, and y2 axis of unique users.
#use twinx to create another y-axis sharing the same x-axis.  subplots returns a figure object and a tuple.
fig, q1_axis1 = plt.subplots()
q1_axis2 = q1_axis1.twinx()
x = q1_hour_grouped.size().index

q1_axis1.plot(x, uniqueUsers, 'g-')
q1_axis2.plot(x, q1_hour_grouped_traffic, 'r-')

q1_axis1.set_xlabel('Hour of the Day')
q1_axis1.set_ylabel('Number of Unique Users per Hour', color='g')
q1_axis2.set_ylabel('Total Traffic Handled', color='r')

#Adjust x-axis to be from 0-23 hours (24 total hours in the day.)
q1_axis1.set_xlim(0,23)
```

Out[38]: (0, 23)



There seems to be a correlations between number of unique users in the hour and the amount of traffic handled during that hour. As the number of unique users grows, so does the amount of traffic. As the number of unique users falls, so does the amount of traffic handled.

Craig Boger
01/2020
CAP4770

QUESTION 2

Question 2

```
#Interpreting the question as we need to see a correlation between a randomly selected set of 100
#users (client IDs) and the times at which they visited the website.  This will be accomplished with
#a scatter plot whose x-axis goes from 1-100 (randomly selected id indeces) and whose y-axis goes from
#0-23 for the hours of the day.  The most popular hours will be where the plotted points cluster in the
#y-axis direction.

#DISREGARD ABOVE ASSUMPTION.  After speaking with TA, the unique client ids should be found,
# then the first 100 unique client ids are selected, no need to randomize.  Instead of selecting
# only 1 row from the client id group, all rows from that client id group are added to the final
# scatter plot.

#Group data by client ID.  Will then later use to randomly select entry from client ID group.
q2_clientID_grouped = log_df.groupby('ClientID')
```

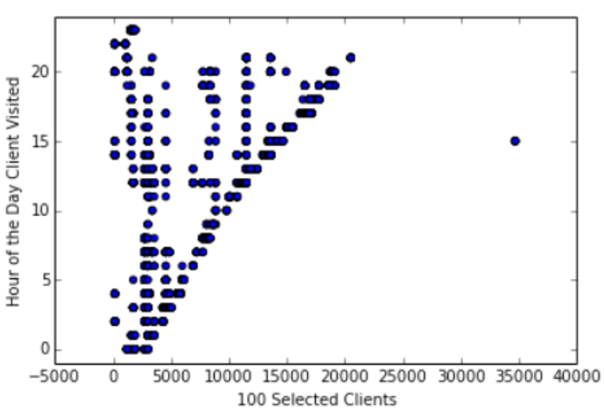
```
#Get arry of distinct client id's from original log df.
#Select first 100 client ID's from the first 100 in the set.
q2_distinctClients = log_df['ClientID'].unique()
q2_distinctClients = q2_distinctClients[0:100]
import numpy as np
#q2_distinctClients = np.random.choice(q2_distinctClients, size=100)
#q2_distinctClients.size
```

```
#Use client ID's to then find a client ID group
#then take all rows from that clientID group.
q2_selected_clients = pd.DataFrame()
for i in q2_distinctClients:
    q2_selected_clients = q2_selected_clients.append(q2_clientID_grouped.get_group(i))
```

```
#Create indices from 1 to 100 for scatter plot.
#q2_selected_clients['GraphIndices'] = q2_selected_clients.reset_index().index + 1
#q2_selected_clients = q2_selected_clients[['hour','GraphIndices']]
```

```
#Create scatter plot comparing 100 unique clients and the hours they visited the site.
q2_plot = q2_selected_clients.plot(kind='scatter',x='ClientID',y='hour')
q2_plot.set_xlabel('100 Selected Clients')
q2_plot.set_ylabel('Hour of the Day Client Visited')
#q2_plot.set_xlim(0,101)
q2_plot.set_ylim(-1,24)
```

```
(-1, 24)
```



QUESTION 3

Craig Boger
01/2020
CAP4770

Question 3

```
#Load csv file and generate datetime column and URL_Type column
q3_log_df = pd.read_csv("./wc_day91_1_log.csv",
                        names=['ClientID', 'Date', 'Time', 'URL', 'ResponseCode', 'Size'],
                        na_values=['-'])
q3_log_df.loc[:,('DateTime')] = pd.to_datetime(q3_log_df.apply(lambda row: row['Date'] + ' ' + row['Time'], axis=1))
q3_log_df['hour'] = q3_log_df.DateTime.dt.hour
q3_log_df['URL_type'] = q3_log_df['URL'].str.split(".", n = -1, expand = False).str.get(-1)
```

```
/home/datascience/.local/lib/python2.7/site-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (4
) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
q3_log_df
```

	ClientID	Date	Time	URL	ResponseCode	Size	DateTime	hour	URL_type
0	2743832	24/Jul/1998	22:00:01	/english/history/body.html	200	2909.0	1998-07-24 22:00:01	22	html
1	2572248	24/Jul/1998	22:00:01	/	200	4930.0	1998-07-24 22:00:01	22	/
2	31798	24/Jul/1998	22:00:02	/french/competition/maincomp.htm	200	12970.0	1998-07-24 22:00:02	22	htm
3	1848501	24/Jul/1998	22:00:02	/	200	4930.0	1998-07-24 22:00:02	22	/
4	248	24/Jul/1998	22:00:03	/images/home_intro.anim.gif	200	60349.0	1998-07-24 22:00:03	22	gif
5	2742956	24/Jul/1998	22:00:03	/french/history/images/history_hm_nav.gif	304	0.0	1998-07-24 22:00:03	22	gif
6	299067	24/Jul/1998	22:00:03	/english/images/news_btn_part_off.gif	304	NaN	1998-07-24 22:00:03	22	gif

```
#Filter out rows just for July 24th, 1998
q3_jul24_log_df = q3_log_df[q3_log_df['Date'] == '24/Jul/1998']
```

```
#Filter out rows just for July 25th, 1998
q3_jul25_log_df = q3_log_df[q3_log_df['Date'] == '25/Jul/1998']
```

QUESTION 1 REDONE WITH JULY 24TH DATA

```
QUESTION 1 REDONE WITH JULY 24TH DATA
```

```
#Narrow down number of unique users and output lists.
q3_jul24_hour_grouped = q3_jul24_log_df.groupby(lambda x: q3_jul24_log_df['DateTime'][x].hour)
q3_jul24_uniqueUsers = q3_jul24_hour_grouped['ClientID'].nunique()
q3_jul24_uniqueUsers
```

```
22    1205
23    1166
Name: ClientID, dtype: int64
```

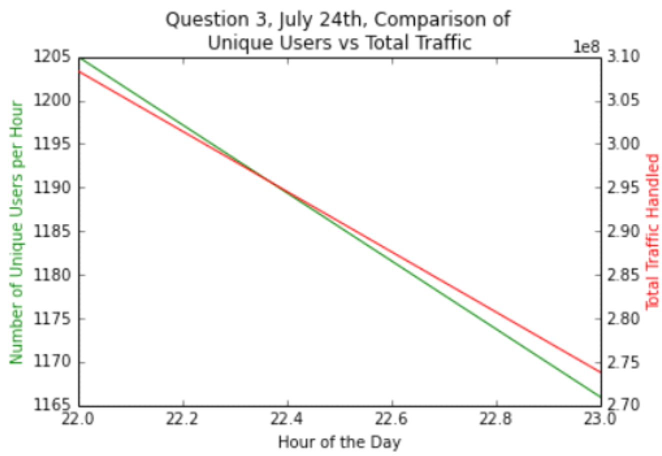
```
#Calculate the size of the traffic on the site every hour.
q3_jul24_hour_grouped_traffic = q3_jul24_hour_grouped['Size'].sum()
q3_jul24_hour_grouped_traffic
```

```
22    308367954.0
23    273831997.0
Name: Size, dtype: float64
```

Craig Boger
01/2020
CAP4770

```
#Generating double line plot with x-axis of hour, y1 axis of traffic, and y2 axis of unique users.
#use twinx to create another y-axis sharing the same x-axis.  subplots returns a figure object and a tuple.
%matplotlib inline
import matplotlib.pyplot as plt

fig, q3_jul24_axis1 = plt.subplots()
q3_jul24_axis2 = q3_jul24_axis1.twinx()
x = q3_jul24_hour_grouped.size().index

q3_jul24_axis1.plot(x, q3_jul24_uniqueUsers, 'g-')
q3_jul24_axis2.plot(x, q3_jul24_hour_grouped_traffic, 'r-')

q3_jul24_axis1.set_title('Question 3, July 24th, Comparison of \nUnique Users vs Total Traffic')
q3_jul24_axis1.set_xlabel('Hour of the Day')
q3_jul24_axis1.set_ylabel('Number of Unique Users per Hour', color='g')
q3_jul24_axis2.set_ylabel('Total Traffic Handled', color='r')

#Adjust x-axis to be from 0-23 hours (24 total hours in the day.)
q3_jul24_axis1.set_xlim(22,23)
```

```
(22, 23)
```



There seems to be a correlations between number of unique users in the hour and the amount of traffic handled during that hour.

Craig Boger
01/2020
CAP4770

QUESTION 1 REDONE WITH JULY 25TH DATA

QUESTION 1 REDONE WITH JULY 25TH DATA

```
#Narrow down number of unique users and output lists.
q3_jul25_hour_grouped = q3_jul25_log_df.groupby(lambda x: q3_jul25_log_df['DateTime'][x].hour)
q3_jul25_uniqueUsers = q3_jul25_hour_grouped['ClientID'].nunique()
q3_jul25_uniqueUsers
```

```
0     1083
1     1100
2     1152
3     1236
4     1163
5     1062
6     1007
7     1060
8     1027
9      950
10    1030
11     944
12    1076
13    1196
14    1246
15    1266
16    1274
17    1171
18    1097
19    1047
20    1032
21     919
Name: ClientID, dtype: int64
```

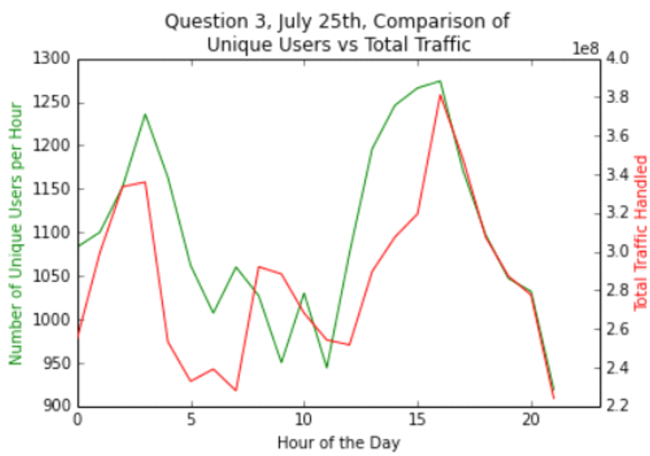
```
#Calculate the size of the traffic on the site every hour.
q3_jul25_hour_grouped_traffic = q3_jul25_hour_grouped['Size'].sum()
q3_jul25_hour_grouped_traffic
```

```
0     254887315.0
1     299315349.0
2     333559527.0
3     335976803.0
4     253267003.0
5     232853603.0
6     239199885.0
7     227961601.0
8     292132496.0
9     288425535.0
10    268152384.0
11    254261855.0
12    251748043.0
13    289745702.0
14    307478577.0
15    319555678.0
16    381169538.0
17    347984361.0
18    307499565.0
19    287315331.0
20    277512322.0
21    224041283.0
Name: Size, dtype: float64
```

Craig Boger
01/2020
CAP4770

```
#Generating double line plot with x-axis of hour, y1 axis of traffic, and y2 axis of unique users.
#use twinx to create another y-axis sharing the same x-axis.  subplots returns a figure object and a tuple.
%matplotlib inline
import matplotlib.pyplot as plt

fig, q3_jul25_axis1 = plt.subplots()
q3_jul25_axis2 = q3_jul25_axis1.twinx()
x = q3_jul25_hour_grouped.size().index

q3_jul25_axis1.plot(x, q3_jul25_uniqueUsers, 'g-')
q3_jul25_axis2.plot(x, q3_jul25_hour_grouped_traffic, 'r-')

q3_jul25_axis1.set_title('Question 3, July 25th, Comparison of \nUnique Users vs Total Traffic')
q3_jul25_axis1.set_xlabel('Hour of the Day')
q3_jul25_axis1.set_ylabel('Number of Unique Users per Hour', color='g')
q3_jul25_axis2.set_ylabel('Total Traffic Handled', color='r')

#Adjust x-axis to be from 0-23 hours (24 total hours in the day.)
q3_jul25_axis1.set_xlim(0,23)
```

```
(0, 23)
```




```
There seems to be a correlations between number of unique users in the hour and the amount of traffic handled during
that hour.
```

Craig Boger
01/2020
CAP4770

QUESTION 2 REDONE WITH JULY 24TH DATA

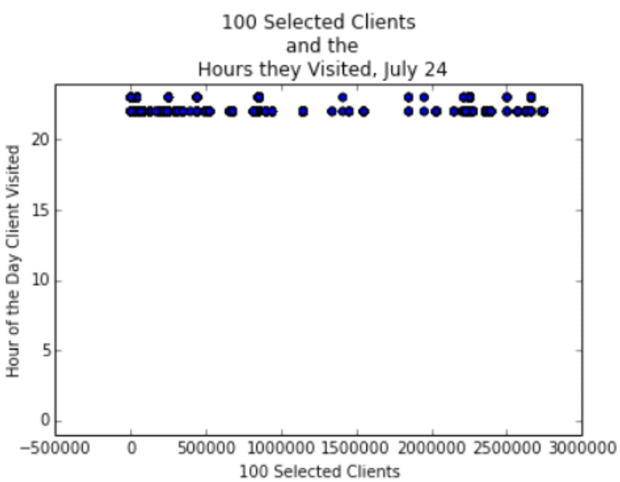
QUESTION 2 REDONE WITH JULY 24TH DATA

```
#Group data by client ID.  Will then later use to select first entry from client ID group.
q3_jul24_clientID_grouped = q3_jul24_log_df.groupby('ClientID')
#Get arry of distinct client id's from original log df.
#Select first 100 client ID's from the first 100 in the unique set.
q3_jul24_distinctClients = q3_jul24_log_df['ClientID'].unique()
q3_jul24_distinctClients = q3_jul24_distinctClients[0:100]
#Use client ID's to then find a client ID group
#then take first row from that clientID group.
q3_jul24_selected_clients = pd.DataFrame()
for i in q3_jul24_distinctClients:
    q3_jul24_selected_clients = q3_jul24_selected_clients.append(q3_jul24_clientID_grouped.get_group(i))

#q3_jul24_selected_clients

#Create scatter plot comparing 100 unique clients and the hours they visited the site.
q3_jul24_plot = q3_jul24_selected_clients.plot(kind='scatter',x='ClientID',y='hour')
q3_jul24_plot.set_xlabel('100 Selected Clients')
q3_jul24_plot.set_ylabel('Hour of the Day Client Visited')
q3_jul24_plot.set_ylim(-1,24)
q3_jul24_plot.set_title('100 Selected Clients\n and the\n Hours they Visited, July 24')
```

```
<matplotlib.text.Text at 0x7fa46e424290>
```



```
<matplotlib.text.Text at 0x7fa46e424290>
```

Craig Boger
01/2020
CAP4770

QUESTION 2 REDONE WITH JULY 25TH DATA

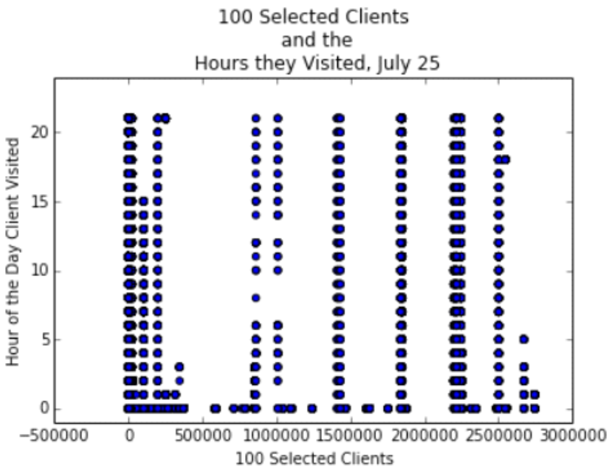
```
QUESTION 2 REDONE WITH JULY 25TH DATA
```

```
#Group data by client ID.  Will then later use to select first entry from client ID group.
q3_jul25_clientID_grouped = q3_jul25_log_df.groupby('ClientID')
#Get arry of distinct client id's from original log df.
#Select first 100 client ID's from the first 100 in the unique set.
q3_jul25_distinctClients = q3_jul25_log_df['ClientID'].unique()
q3_jul25_distinctClients = q3_jul25_distinctClients[0:100]
#Use client ID's to then find a client ID group
#then take first row from that clientID group.
q3_jul25_selected_clients = pd.DataFrame()
for i in q3_jul25_distinctClients:
    q3_jul25_selected_clients = q3_jul25_selected_clients.append(q3_jul25_clientID_grouped.get_group(i))

#q3_jul24_selected_clients

#Create scatter plot comparing 100 unique clients and the hours they visited the site.
q3_jul25_plot = q3_jul25_selected_clients.plot(kind='scatter',x='ClientID',y='hour')
q3_jul25_plot.set_xlabel('100 Selected Clients')
q3_jul25_plot.set_ylabel('Hour of the Day Client Visited')
q3_jul25_plot.set_ylim(-1,24)
q3_jul25_plot.set_title('100 Selected Clients\n and the\n Hours they Visited, July 25')
```

```
<matplotlib.text.Text at 0x7fa46da7d3d0>
```

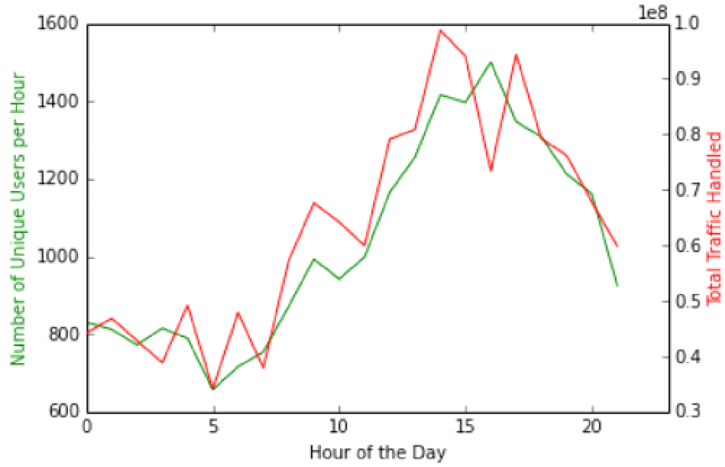


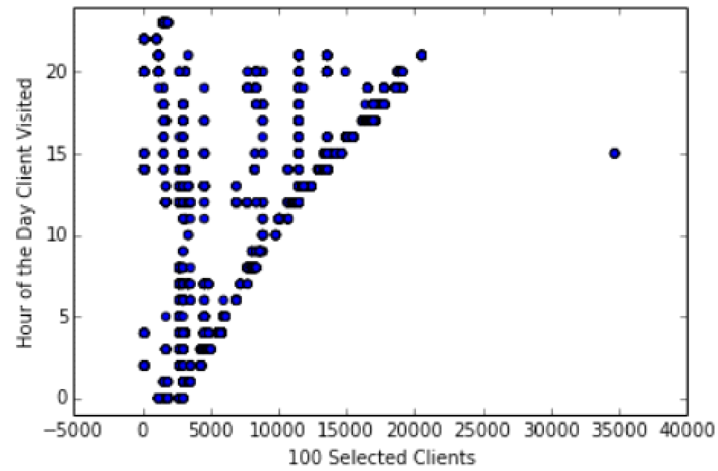
```
<matplotlib.text.Text at 0x7fa46da7d3d0>
```

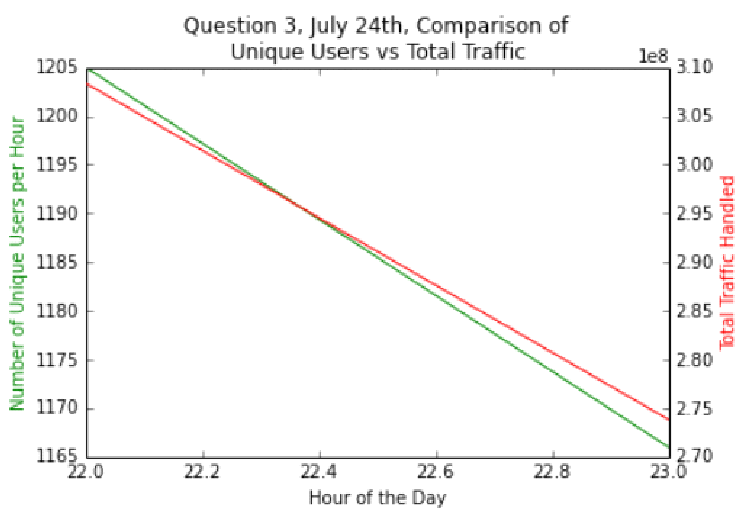



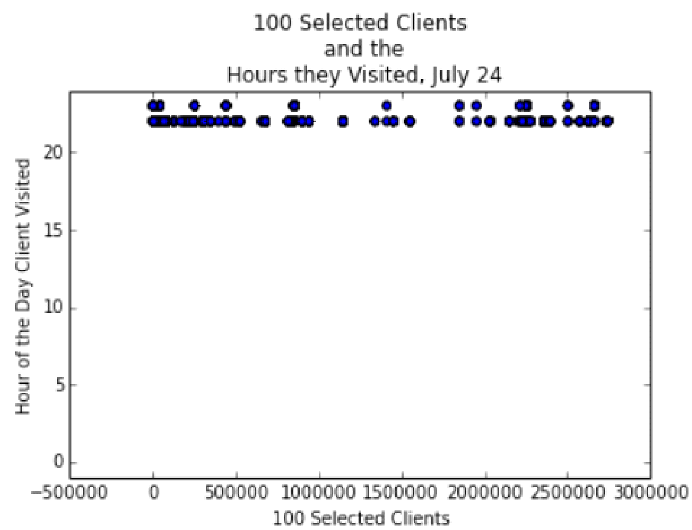
Craig Boger
01/2020
CAP4770

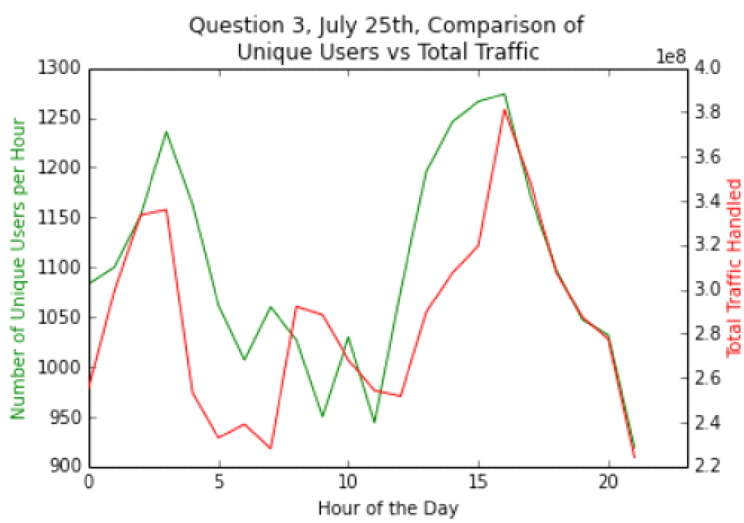
Plots Compiled from Q1, Q2, and Q3

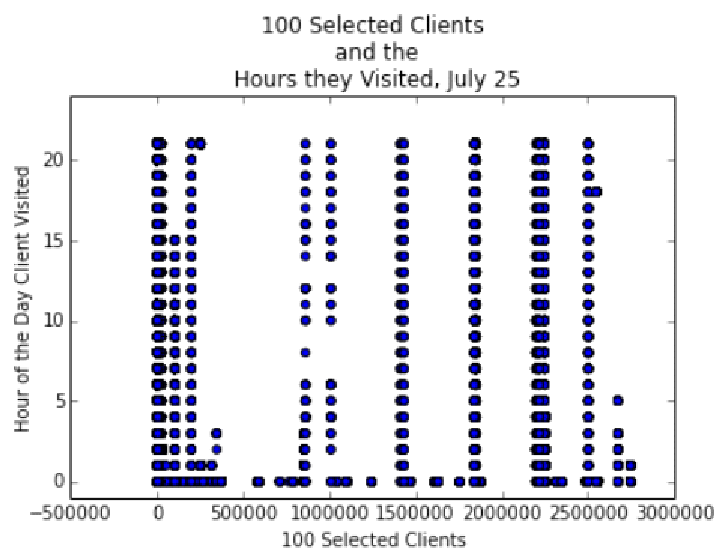












Craig Boger
01/2020
CAP4770

Conclusions from Plots

For the line plots of "number of unique users" vs "hour of the day", there was a clear connection in all cases between the number of unique visitors and the traffic the website handled. Going from May to July (question 1 to question 3), there was what appeared to be more sustained users on the website through hours 22 and 23 or July 24th and 0-21 of July 25th with a sharp drop during the 17th hour. It also appears in July that users were browsing more content that required more traffic. From May, a user level of 800 users would sustain a traffic level of about 0.45E8. The traffic level in July for only 100-150 more unique users increased 4.9 times from 0.45E8 to 2.2E8.

For the scatter plots, there seemed to be less users browsing repeatedly throughout the entire day. Some would visit between hours 0-23 consistently, but others repeatedly visited in narrower windows between hours 10-20 for example. The July data showed many more sustained users throughout the entire hours range than the May data. Many users were consistently visiting almost every hour of the day on July 25th.