911 Calls Data Capstone Project

November 28, 2017

1 911 Calls Project

I analyzed some 911 call data from Kaggle. The data contains the following fields:

- lat : String variable, Latitude
- lng: String variable, Longitude
- desc: String variable, Description of the Emergency Call
- zip: String variable, Zipcode
- title: String variable, Title
- timeStamp: String variable, YYYY-MM-DD HH:MM:SS
- twp: String variable, Township
- addr: String variable, Address
- e: String variable, Dummy variable (always 1)

1.1 Data and Setup

** Import numpy and pandas **

```
In [2]: import numpy as np
        import pandas as pd
```

** Import visualization libraries **

```
In [8]: import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline
```

```
In [9]: df1 = pd.read_csv('911.csv')
```

```
In [10]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99492 entries, 0 to 99491
Data columns (total 9 columns):
lat          99492 non-null float64
```

```
lng          99492 non-null float64
desc         99492 non-null object
zip          86637 non-null float64
title        99492 non-null object
timeStamp    99492 non-null object
twp          99449 non-null object
addr         98973 non-null object
e            99492 non-null int64
dtypes: float64(3), int64(1), object(5)
memory usage: 6.8+ MB
```

** The head of df **

```
In [7]: df1.head()
```

```
Out[7]:          lat        lng                                                desc  \
        0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station ...
        1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP...
        2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...
        3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;...
        4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S...

               zip                    title            timeStamp                twp  \
        0  19525.0   EMS: BACK PAINS/INJURY  2015-12-10 17:40:00        NEW HANOVER
        1  19446.0  EMS: DIABETIC EMERGENCY  2015-12-10 17:40:00  HATFIELD TOWNSHIP
        2  19401.0      Fire: GAS-ODOR/LEAK  2015-12-10 17:40:00         NORRISTOWN
        3  19401.0   EMS: CARDIAC EMERGENCY  2015-12-10 17:40:01         NORRISTOWN
        4      NaN           EMS: DIZZINESS  2015-12-10 17:40:01   LOWER POTTSGROVE

                              addr  e
        0      REINDEER CT & DEAD END  1
        1  BRIAR PATH & WHITEMARSH LN  1
        2                    HAWS AVE  1
        3          AIRY ST & SWEDE ST  1
        4    CHERRYWOOD CT & DEAD END  1
```

** The top 5 zipcodes for 911 calls. **

```
In [12]: df1['zip'].value_counts().head(5)
```

```
Out[12]: 19401.0    6979
         19464.0    6643
         19403.0    4854
         19446.0    4748
         19406.0    3174
         Name: zip, dtype: int64
```

** The 5 townships (twp) for 911 calls. **

```
In [17]: df1['twp'].value_counts().head(5)
```

```
Out[17]: LOWER MERION    8443
         ABINGTON        5977
         NORRISTOWN      5890
         UPPER MERION    5227
         CHELTENHAM      4575
         Name: twp, dtype: int64
```

** unique title codes **

```
In [19]: df1['title'].nunique()
```

```
Out[19]: 110
```

1.2 Creating new features

** In the titles column there are "Reasons/Departments" specified before the title code. These are EMS, Fire, and Traffic. I used .apply() with a custom lambda expression to create a new column called "Reason" that contains this string value.**

For example, if the title column value is EMS: BACK PAINS/INJURY , the Reason column value would be EMS.

```
In [20]: x = df1['title'].iloc[0]
         x.split(':')[0]
         df1['Reason'] = df1['title'].apply(lambda title: title.split(':')[0])
         df1['Reason']
```

```
Out[20]: 0            EMS
         1            EMS
         2           Fire
         3            EMS
         4            EMS
         5            EMS
         6            EMS
         7            EMS
         8            EMS
         9        Traffic
         10       Traffic
         11       Traffic
         12       Traffic
         13       Traffic
         14       Traffic
         15       Traffic
         16           EMS
         17           EMS
         18           EMS
         19       Traffic
         20       Traffic
```

```
21       Traffic
22          Fire
23       Traffic
24       Traffic
25           EMS
26           EMS
27          Fire
28       Traffic
29       Traffic
          ...
99462        EMS
99463        EMS
99464        EMS
99465        EMS
99466        EMS
99467        EMS
99468       Fire
99469       Fire
99470        EMS
99471        EMS
99472        EMS
99473        EMS
99474        EMS
99475    Traffic
99476        EMS
99477        EMS
99478    Traffic
99479        EMS
99480        EMS
99481        EMS
99482    Traffic
99483        EMS
99484       Fire
99485    Traffic
99486    Traffic
99487    Traffic
99488    Traffic
99489        EMS
99490        EMS
99491    Traffic
Name: Reason, Length: 99492, dtype: object
```

** What is the most common Reason for a 911 call based off of this new column? **

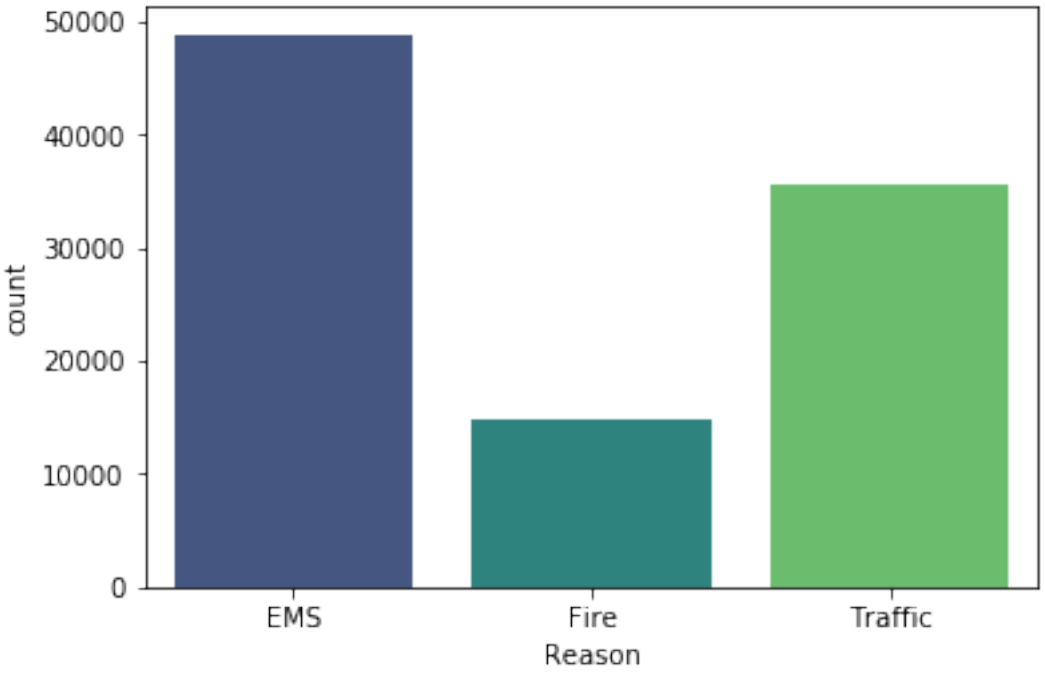
```
In [22]: df1['Reason'].value_counts().head(1)
```

```
Out[22]: EMS    48877
         Name: Reason, dtype: int64
```

** Used seaborn to create a countplot of 911 calls by Reason. **

```
In [24]: sns.countplot(x='Reason',data=df1,palette='viridis')
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1c1d4750>
```




```
In [26]: df1.info()
         type(df1['timeStamp'].iloc[0])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99492 entries, 0 to 99491
Data columns (total 10 columns):
lat          99492 non-null float64
lng          99492 non-null float64
desc         99492 non-null object
zip          86637 non-null float64
title        99492 non-null object
timeStamp    99492 non-null object
twp          99449 non-null object
addr         98973 non-null object
e            99492 non-null int64
Reason       99492 non-null object
dtypes: float64(3), int64(1), object(6)
memory usage: 7.6+ MB
```

```
Out[26]: str
```

** I used pd.to_datetime to convert the column from strings to DateTime objects. **

```
In [29]: df1['timeStamp'] = pd.to_datetime(df1['timeStamp'])
         type(df1['timeStamp'].iloc[0])

Out[29]: pandas._libs.tslib.Timestamp
```

** You can now grab specific attributes from a Datetime object by calling them. For example:**

```
time = df['timeStamp'].iloc[0]
time.hour
```

You can use Jupyter's tab method to explore the various attributes you can call. Now that the timestamp column are actually DateTime objects, where .apply() was used to create 3 new columns called Hour, Month, and Day of Week.

```
In [30]: time = df1['timeStamp'].iloc[0]
         time.hour

Out[30]: 17

In [33]: df1['Hour'] = df1['timeStamp'].apply(lambda time: time.hour)

In [34]: df1['Month'] = df1['timeStamp'].apply(lambda time: time.month)
         df1['Day of Week'] = df1['timeStamp'].apply(lambda time: time.dayofweek)

In [35]: df1.head()

Out[35]:          lat        lng                                                desc  \
         0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station ...
         1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP...
         2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...
         3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;...
         4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S...

                zip                    title            timeStamp                twp  \
         0  19525.0   EMS: BACK PAINS/INJURY  2015-12-10 17:40:00        NEW HANOVER
         1  19446.0  EMS: DIABETIC EMERGENCY  2015-12-10 17:40:00  HATFIELD TOWNSHIP
         2  19401.0      Fire: GAS-ODOR/LEAK  2015-12-10 17:40:00         NORRISTOWN
         3  19401.0   EMS: CARDIAC EMERGENCY  2015-12-10 17:40:01         NORRISTOWN
         4      NaN           EMS: DIZZINESS  2015-12-10 17:40:01   LOWER POTTSGROVE

                                 addr  e Reason  Hour  Month  Day of Week
         0      REINDEER CT & DEAD END  1    EMS    17     12            3
         1  BRIAR PATH & WHITEMARSH LN  1    EMS    17     12            3
         2                    HAWS AVE  1   Fire    17     12            3
         3          AIRY ST & SWEDE ST  1    EMS    17     12            3
         4    CHERRYWOOD CT & DEAD END  1    EMS    17     12            3
```

** Notice how the Day of Week is an integer 0-6. Use the .map() with this dictionary to map the actual string names to the day of the week: **

```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
```

```
In [36]: dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
```

```
In [37]: df1['Day of Week'] = df1['Day of Week'].map(dmap)
```

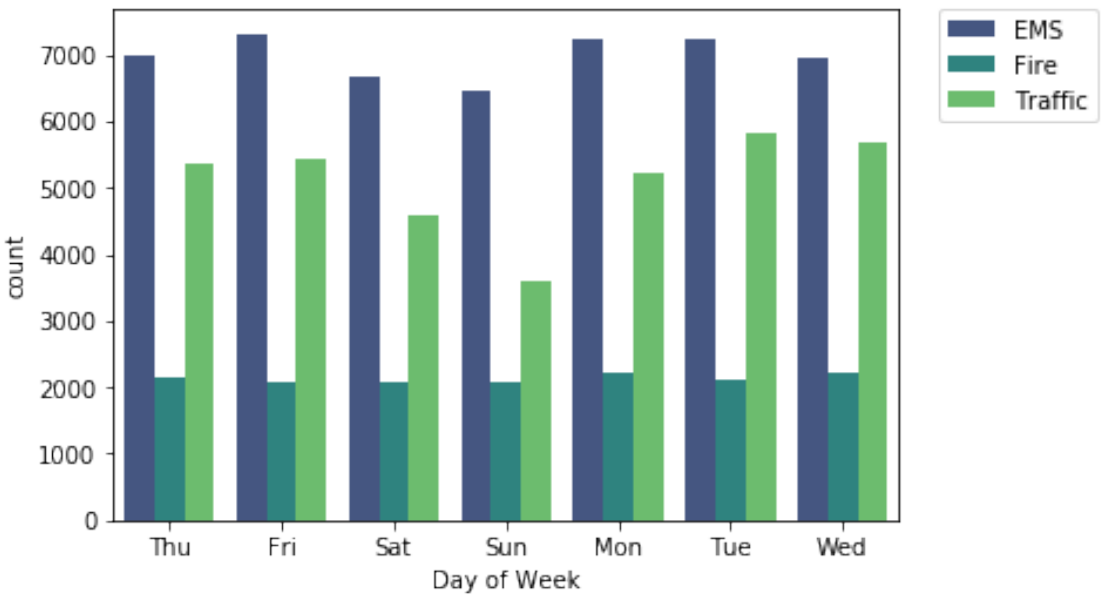
```
In [38]: df1.head()
```

```
Out[38]:          lat        lng                                              desc  \
         0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station ...
         1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP...
         2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...
         3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;...
         4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S...

                zip                    title           timeStamp                twp  \
         0  19525.0   EMS: BACK PAINS/INJURY 2015-12-10 17:40:00        NEW HANOVER
         1  19446.0  EMS: DIABETIC EMERGENCY 2015-12-10 17:40:00  HATFIELD TOWNSHIP
         2  19401.0      Fire: GAS-ODOR/LEAK 2015-12-10 17:40:00         NORRISTOWN
         3  19401.0   EMS: CARDIAC EMERGENCY 2015-12-10 17:40:01         NORRISTOWN
         4      NaN           EMS: DIZZINESS 2015-12-10 17:40:01   LOWER POTTSGROVE

                                 addr  e Reason  Hour  Month Day of Week
         0      REINDEER CT & DEAD END  1    EMS    17     12         Thu
         1  BRIAR PATH & WHITEMARSH LN  1    EMS    17     12         Thu
         2                   HAWS AVE  1   Fire    17     12         Thu
         3          AIRY ST & SWEDE ST  1    EMS    17     12         Thu
         4    CHERRYWOOD CT & DEAD END  1    EMS    17     12         Thu
```

** Seaborn: to create a countplot of the Day of Week column with the hue based off of the Reason column. **

```
In [42]: sns.countplot(x='Day of Week', data=df1,hue='Reason',palette='viridis')
         plt.legend(bbox_to_anchor=(1.05,1),loc=2,borderaxespad=0.)
```

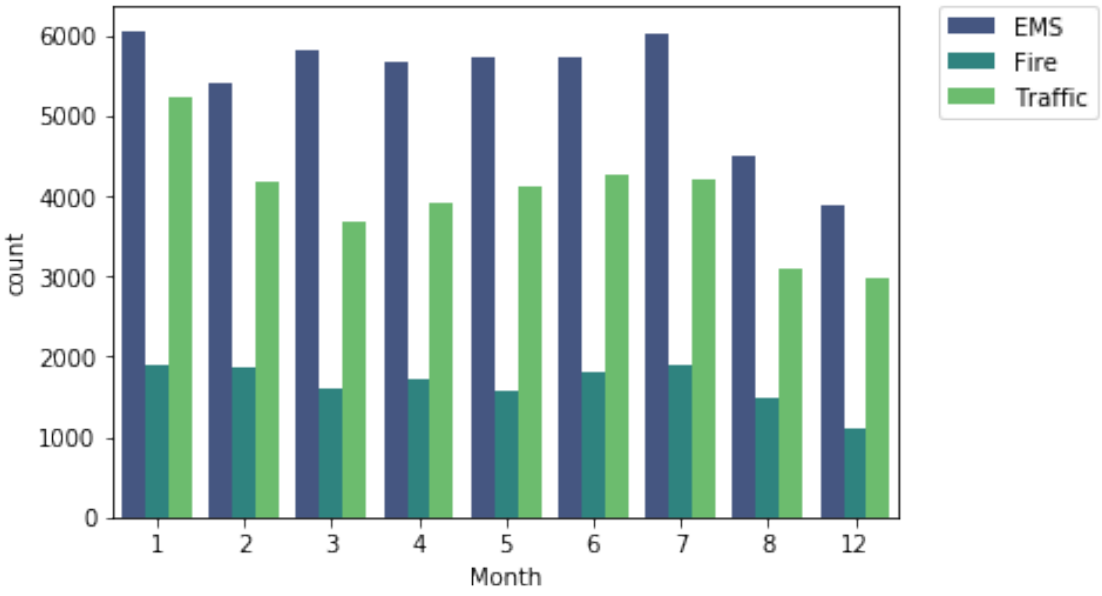
```
Out[42]: <matplotlib.legend.Legend at 0x1a1a0e94d0>
```



Now for Month:

```
In [43]: sns.countplot(x='Month', data=df1,hue='Reason',palette='viridis')
         plt.legend(bbox_to_anchor=(1.05,1),loc=2,borderaxespad=0.)
```

```
Out[43]: <matplotlib.legend.Legend at 0x1a1a012710>
```



Did you notice something strange about the Plot?

** You should have noticed it was missing some Months... **

** Group the DataFrame by the month column and used the count() method for aggregation. Used the head() method on this returned DataFrame. **

```
In [46]: byMonth = df1.groupby('Month').count()
         byMonth.head()
```

Out[46]:

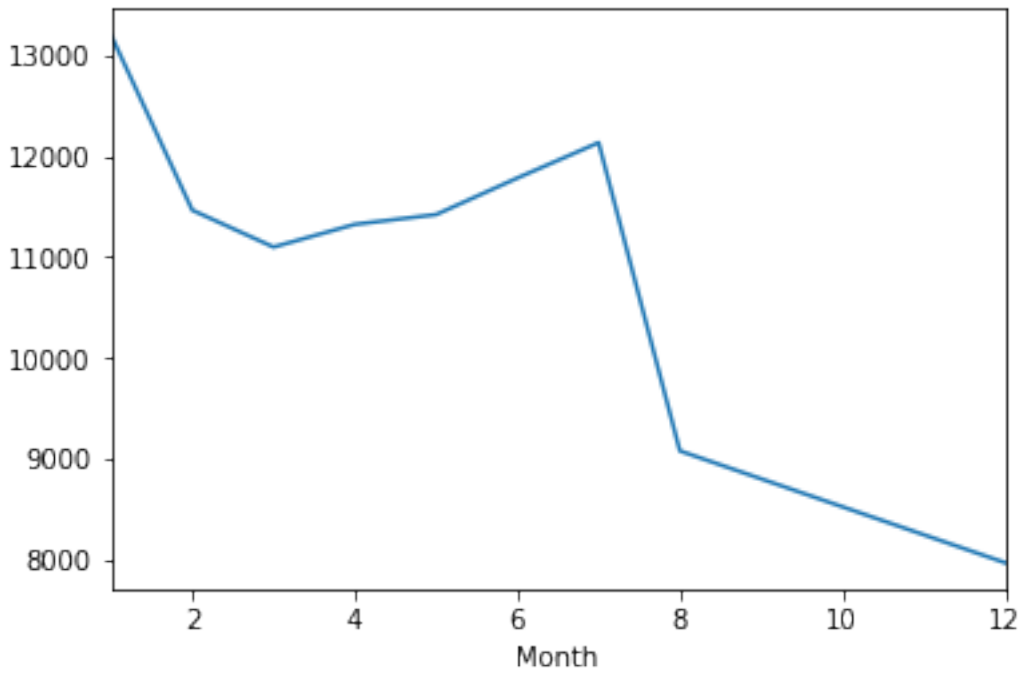
Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e
1	13205	13205	13205	11527	13205	13205	13203	13096	13205
2	11467	11467	11467	9930	11467	11467	11465	11396	11467
3	11101	11101	11101	9755	11101	11101	11092	11059	11101
4	11326	11326	11326	9895	11326	11326	11323	11283	11326
5	11423	11423	11423	9946	11423	11423	11420	11378	11423

Month	Reason	Hour	Day of Week
1	13205	13205	13205
2	11467	11467	11467
3	11101	11101	11101
4	11326	11326	11326
5	11423	11423	11423

** Created a simple plot off of the dataframe indicating the count of calls per month. **

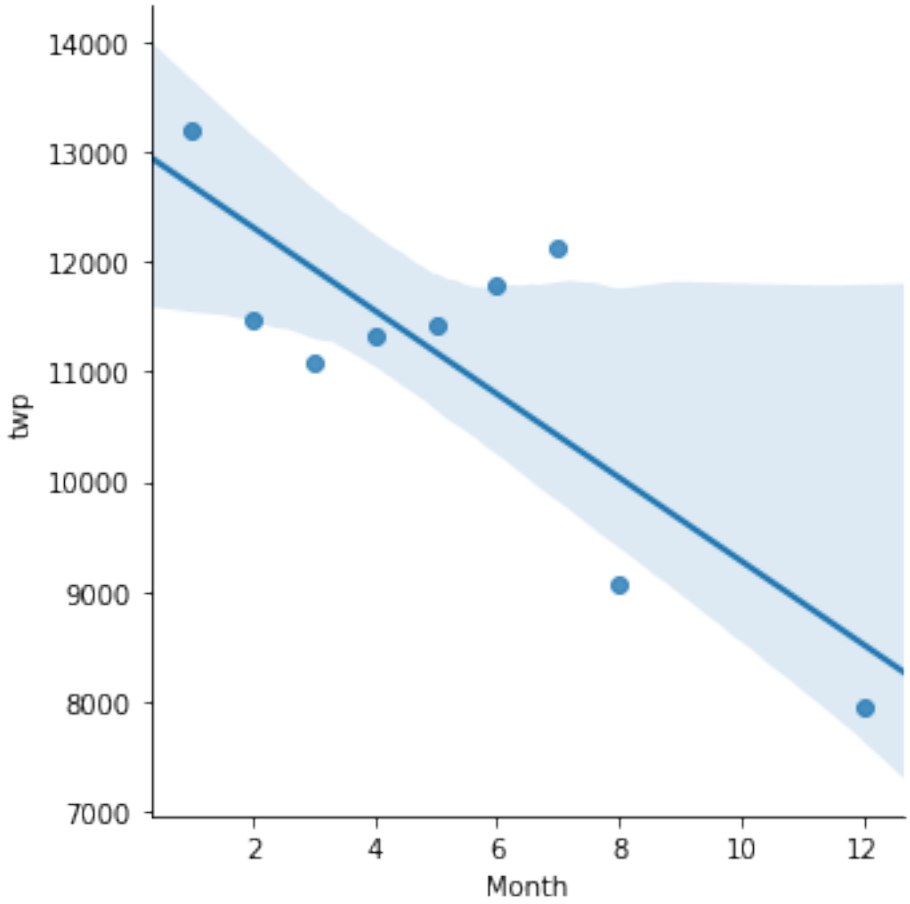
```
In [47]: byMonth['lat'].plot()
```

```
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1a19ef3d90>
```



** Used seaborn's lmplot() to create a linear fit on the number of calls per month. Using reset. **

```
In [49]: sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())

Out[49]: <seaborn.axisgrid.FacetGrid at 0x1a176e2290>
```



Created a new column called 'Date' that contains the date from the timeStamp column.

```
In [63]: t= df1['timeStamp'].iloc[0]
         df1['Date'] = df1['timeStamp'].apply(lambda t:t.date())
         t.date()
         df1.head()

Out[63]:          lat        lng                                                desc  \
         0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station ...
         1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP...
         2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...
```

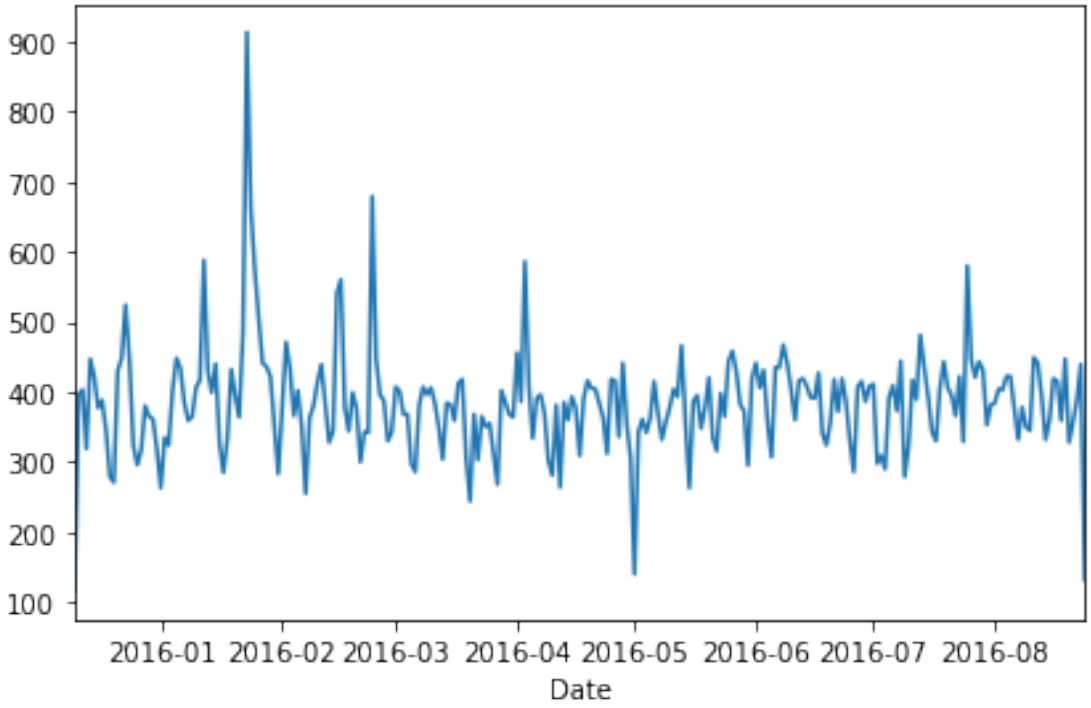
```
3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;...
4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S...

       zip                    title           timeStamp                twp  \
0  19525.0   EMS: BACK PAINS/INJURY 2015-12-10 17:40:00        NEW HANOVER
1  19446.0  EMS: DIABETIC EMERGENCY 2015-12-10 17:40:00  HATFIELD TOWNSHIP
2  19401.0     Fire: GAS-ODOR/LEAK 2015-12-10 17:40:00         NORRISTOWN
3  19401.0   EMS: CARDIAC EMERGENCY 2015-12-10 17:40:01         NORRISTOWN
4      NaN           EMS: DIZZINESS 2015-12-10 17:40:01   LOWER POTTSGROVE

                         addr  e Reason  Hour  Month Day of Week        Date
0      REINDEER CT & DEAD END  1    EMS    17     12         Thu  2015-12-10
1  BRIAR PATH & WHITEMARSH LN  1    EMS    17     12         Thu  2015-12-10
2                   HAWS AVE  1   Fire    17     12         Thu  2015-12-10
3          AIRY ST & SWEDE ST  1    EMS    17     12         Thu  2015-12-10
4    CHERRYWOOD CT & DEAD END  1    EMS    17     12         Thu  2015-12-10
```

** Now groupby this Date column with the count() aggregate and create a plot of counts of 911 calls.**

```
In [64]: df1.groupby('Date').count().head()
```

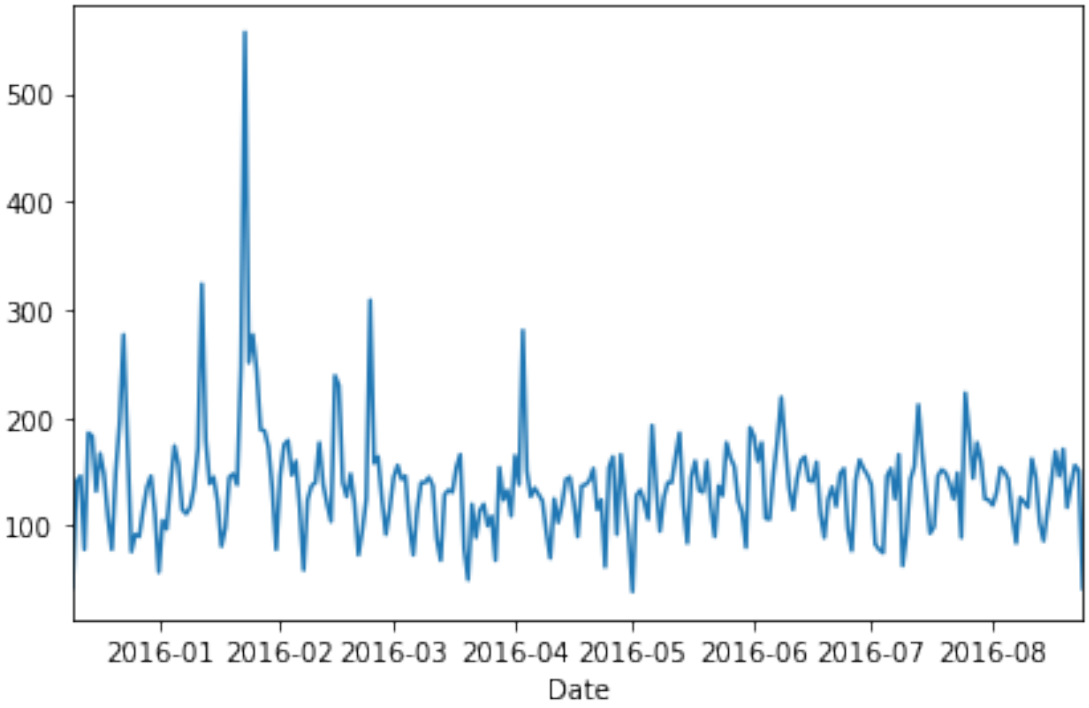
```
Out[64]:             lat  lng  desc  zip  title  timeStamp  twp  addr    e  Reason  \
         Date
         2015-12-10  115  115   115  100    115        115  115   113  115     115
         2015-12-11  396  396   396  333    396        396  395   391  396     396
         2015-12-12  403  403   403  333    403        403  403   401  403     403
         2015-12-13  319  319   319  280    319        319  319   317  319     319
         2015-12-14  447  447   447  387    447        447  446   445  447     447

                     Hour  Month  Day of Week
         Date
         2015-12-10   115    115          115
         2015-12-11   396    396          396
         2015-12-12   403    403          403
         2015-12-13   319    319          319
         2015-12-14   447    447          447
```

```
In [66]: df1.groupby('Date').count()['lat'].plot()
         plt.tight_layout()
```

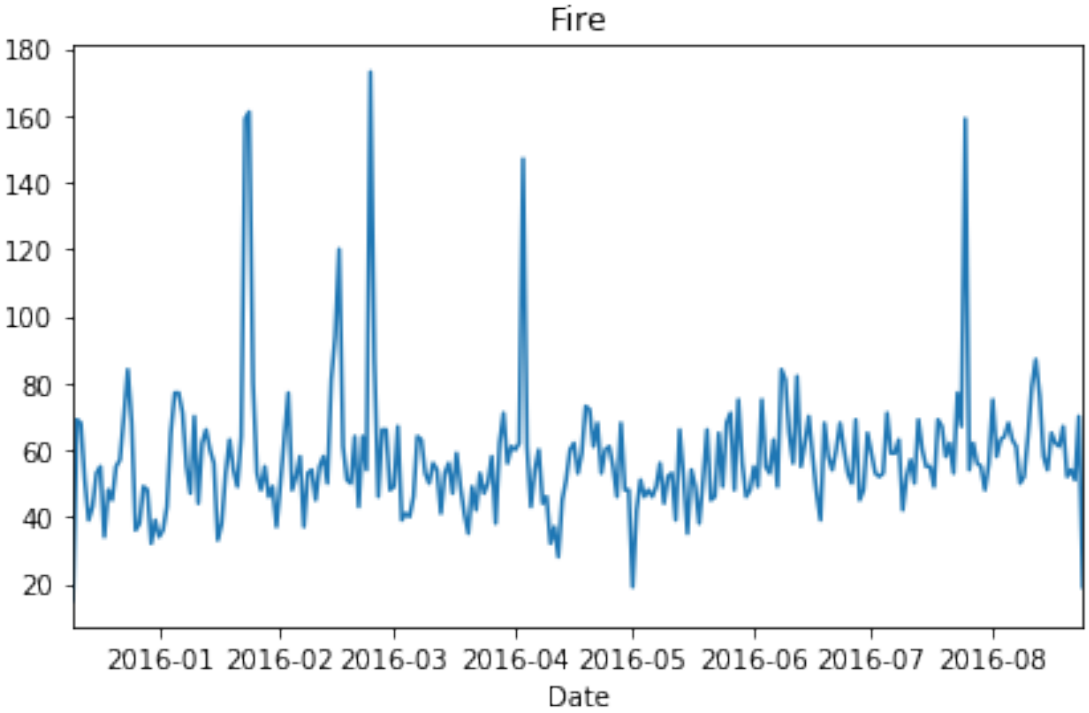


** Recreated this plot but create 3 separate plots with each plot representing a Reason for the 911 call**

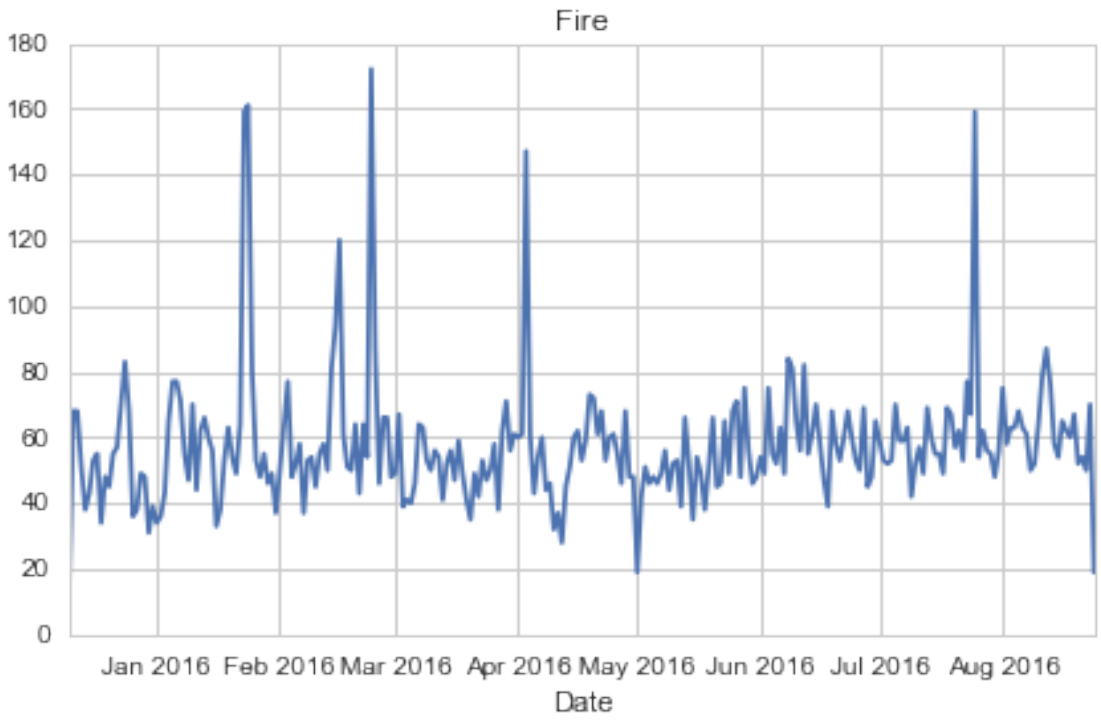
```
In [67]: df1[df1['Reason']=='Traffic'].groupby('Date').count()['lat'].plot()
         plt.title('Traffic')
         plt.tight_layout()
```



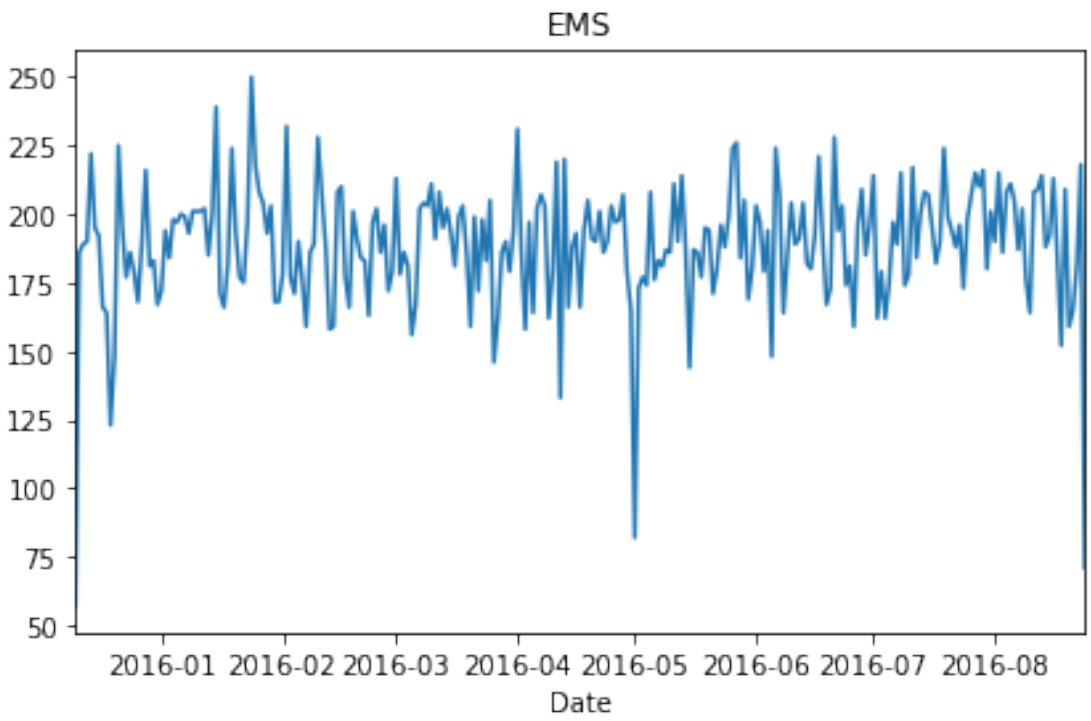
In [68]:
```
df1[df1['Reason']=='Fire'].groupby('Date').count()['lat'].plot()
plt.title('Fire')
plt.tight_layout()
```



In [201]:



In [69]:
```
df1[df1['Reason']=='EMS'].groupby('Date').count()['lat'].plot()
plt.title('EMS')
plt.tight_layout()
```

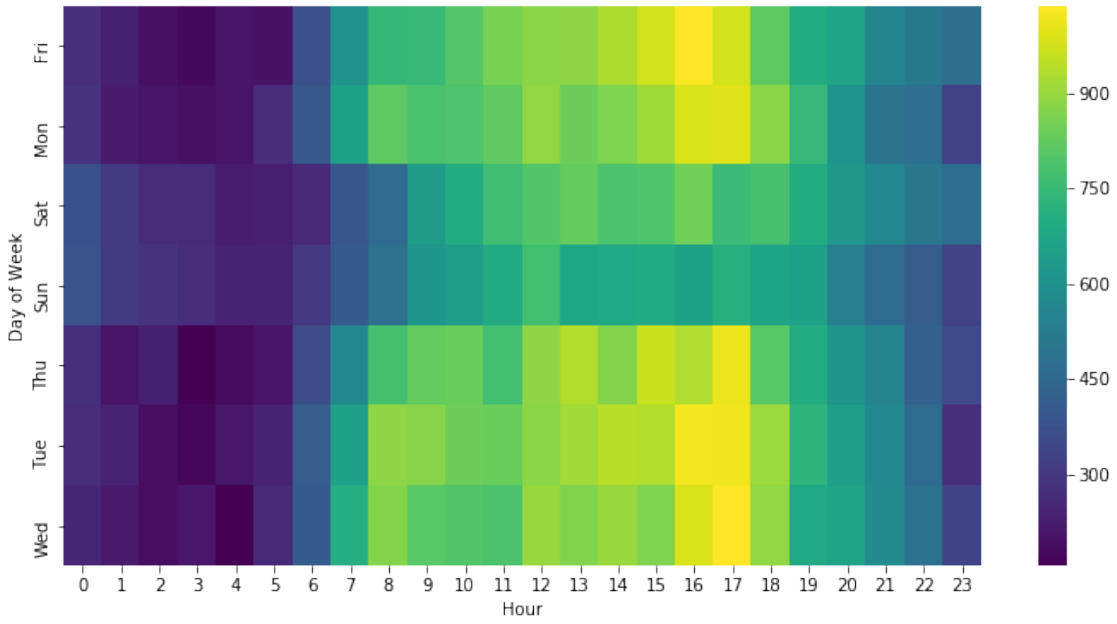


** Now let's move on to creating heatmaps with seaborn and our data. We'll first need to restructure the dataframe so that the columns become the Hours and the Index becomes the Day of the Week. I tried to combine groupby with an unstack method.**

```
In [73]: dayHour = df1.groupby(by=['Day of Week', 'Hour']).count()['Reason'].unstack()
```

** HeatMap using this new DataFrame. **

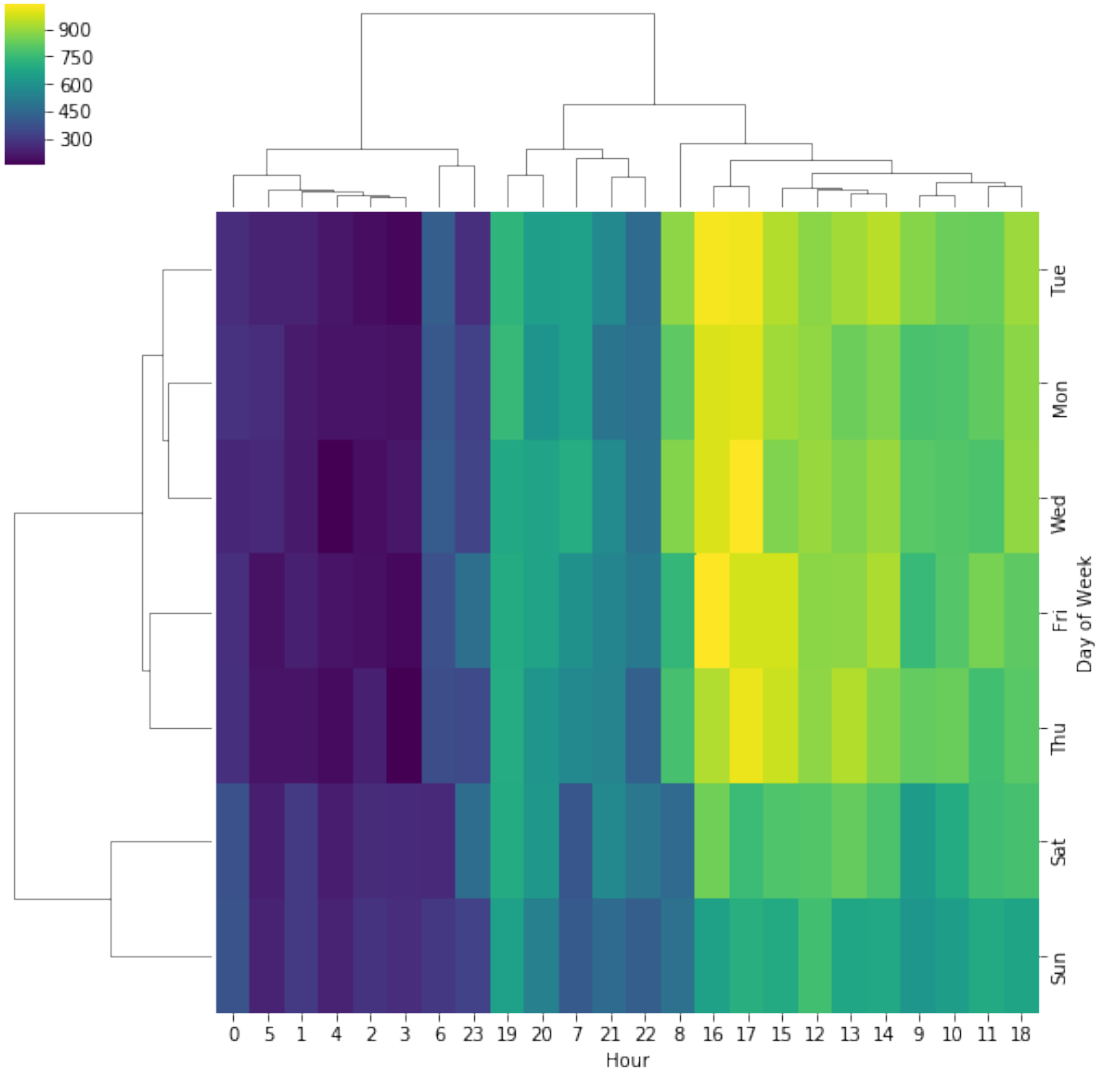
```
In [77]: plt.figure(figsize=(12,6))
         sns.heatmap(dayHour,cmap='viridis')

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x1a24b47a10>
```



** Clustermap using this DataFrame. **

```
In [78]: sns.clustermap(dayHour, cmap='viridis')
```

```
Out[78]: <seaborn.matrix.ClusterGrid at 0x1a25937510>
```



** Now repeat these same plots and operations, for a DataFrame that shows the Month as the column. **

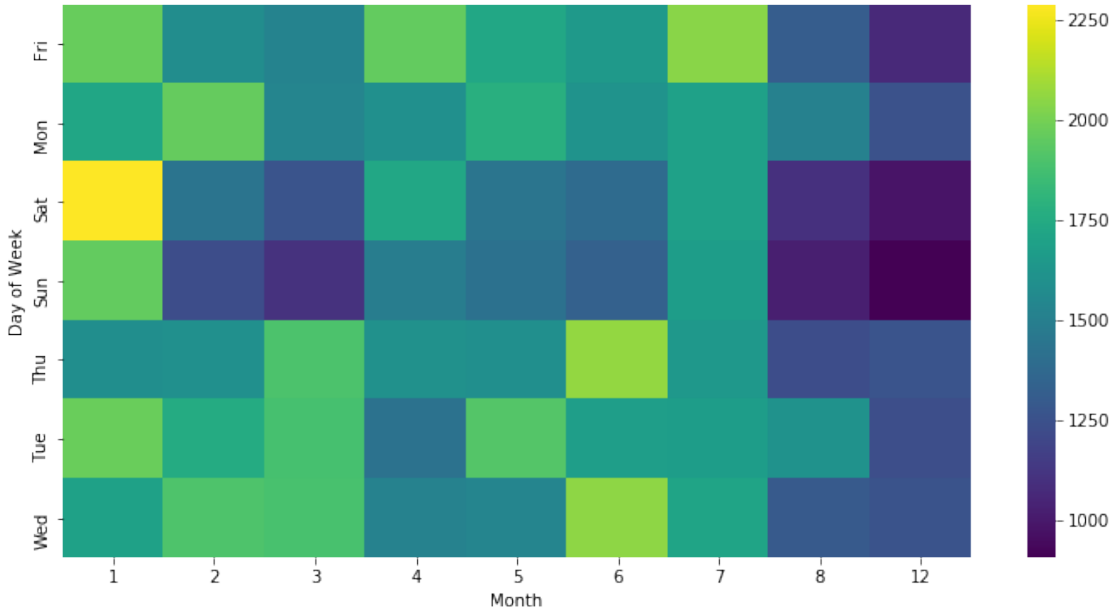
```
In [80]: dayMonth = df1.groupby(by=['Day of Week', 'Month']).count()['Reason'].unstack()
         dayMonth.head()
```

Out[80]:

Month	1	2	3	4	5	6	7	8	12
Day of Week									
Fri	1970	1581	1525	1958	1730	1649	2045	1310	1065
Mon	1727	1964	1535	1598	1779	1617	1692	1511	1257
Sat	2291	1441	1266	1734	1444	1388	1695	1099	978
Sun	1960	1229	1102	1488	1424	1333	1672	1021	907
Thu	1584	1596	1900	1601	1590	2065	1646	1230	1266

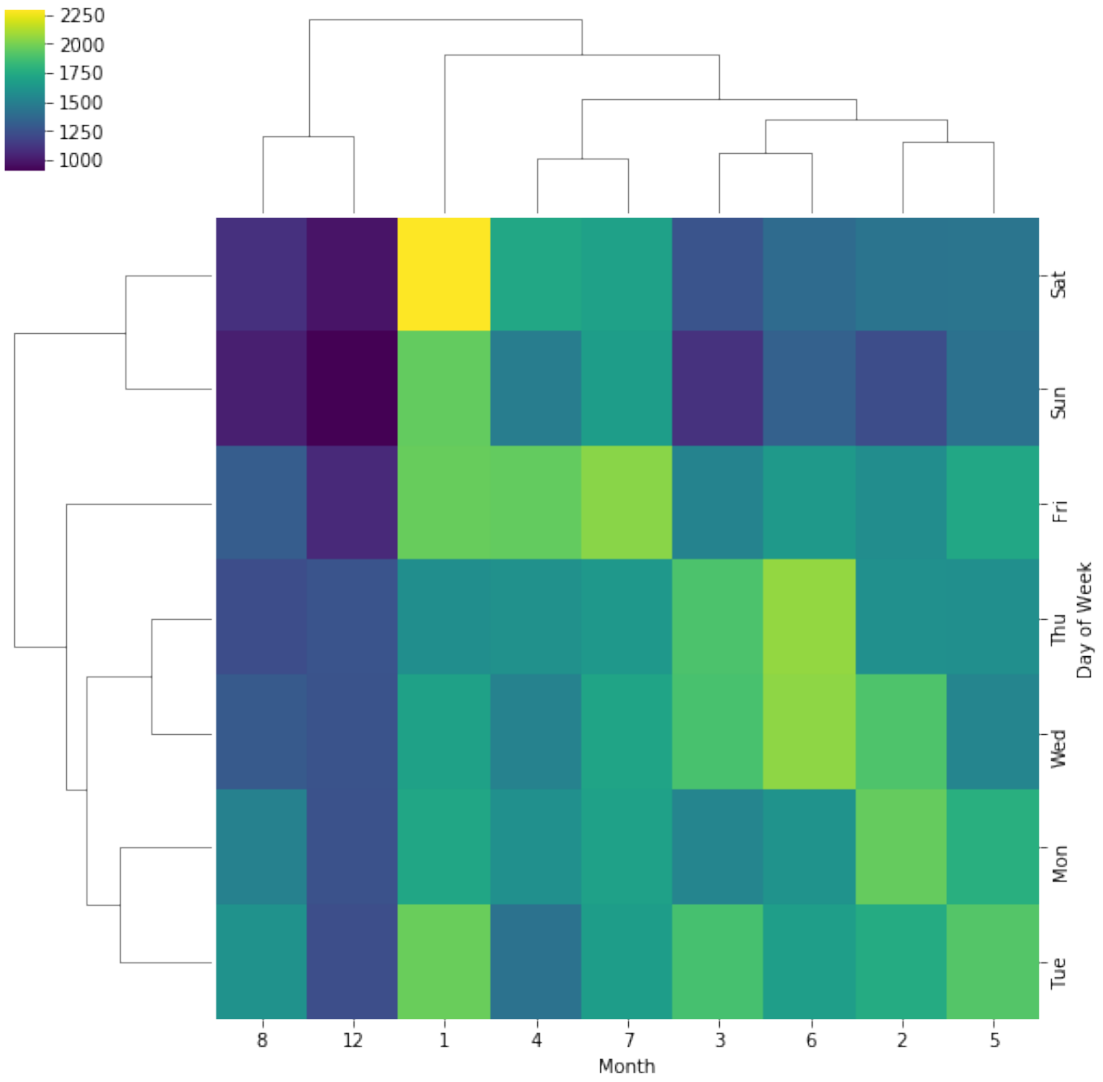
```
In [81]: plt.figure(figsize=(12,6))
         sns.heatmap(dayMonth, cmap='viridis')
```

Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x1a195c2c10>



In [82]: sns.clustermap(dayMonth,cmap='viridis')

Out[82]: <seaborn.matrix.ClusterGrid at 0x1a1ba448d0>


2250
2000
1750
1500
1250
1000
Sat
Sun
Fri
Thu
Wed
Mon
Tue
Day of Week
8
12
1
4
7
3
6
2
5
Month