911_Calls

September 1, 2020

1 Kaushal Rao - Analysis of Dataset Regarding 911 Calls

1.0.1 The dataset was downloaded from Kaggle (https://www.kaggle.com/mchirico/montcoalert), and it 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)

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set_style('whitegrid')
        %matplotlib inline
        # importing necessary libraries
```

```
In [2]: df = pd.read_csv('911.csv')
        df.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
```

```
In [3]: df.head()
```

```
Out[3]:          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
```

```
In [4]: df['zip'].value_counts().head()
        # top 5 zip codes of 911 calls
```

```
Out[4]: 19401.0    6979
        19464.0    6643
        19403.0    4854
        19446.0    4748
        19406.0    3174
        Name: zip, dtype: int64
```

```
In [5]: df['twp'].value_counts().head()
        # top 5 townships of 911 calls
```

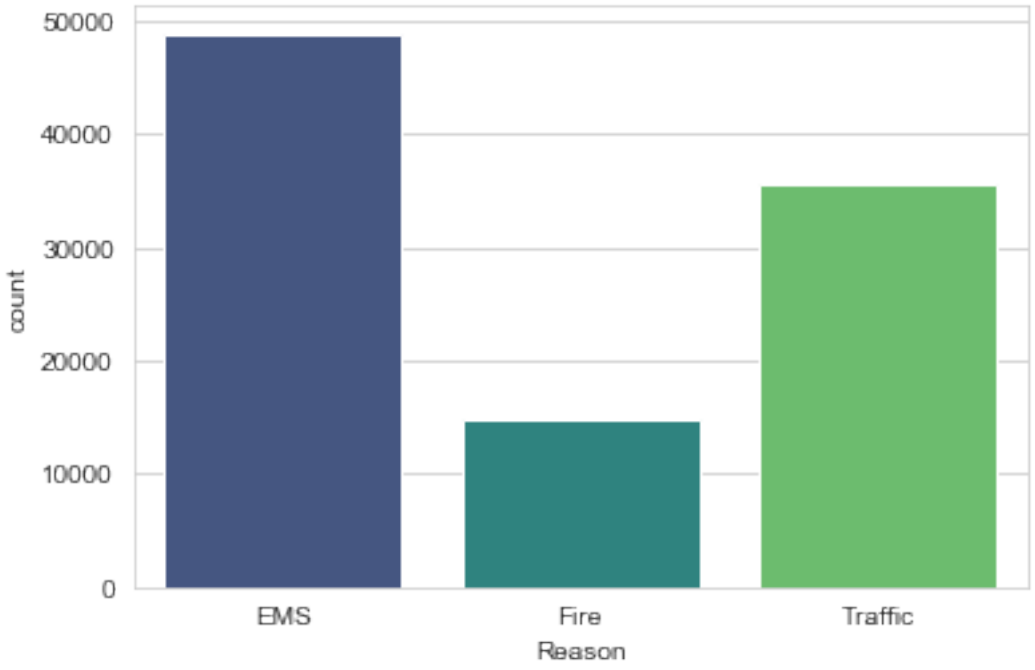
```
Out[5]: LOWER MERION    8443
        ABINGTON        5977
        NORRISTOWN      5890
        UPPER MERION    5227
        CHELTENHAM      4575
        Name: twp, dtype: int64
```

```
In [6]: df['title'].nunique()
        # 110 unique title codes from 911 calls
```

```
Out[6]: 110
```

1.0.2 Creating New Features

```
In [7]: df['Reason'] = df['title'].apply(lambda title: title.split(':')[0])
        # Creating new "Reason" column that contains the relevant 911 Department
        # Can be either EMS, Fire, and Traffic

In [8]: df['Reason'].value_counts()
        # Distribution of "Reason" column, EMS calls are the most common

Out[8]: EMS        48877
        Traffic    35695
        Fire       14920
        Name: Reason, dtype: int64

In [9]: sns.countplot(x='Reason',data=df,palette='viridis')
        # creating countplot to visualize this distribution

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2425dcf8>
```

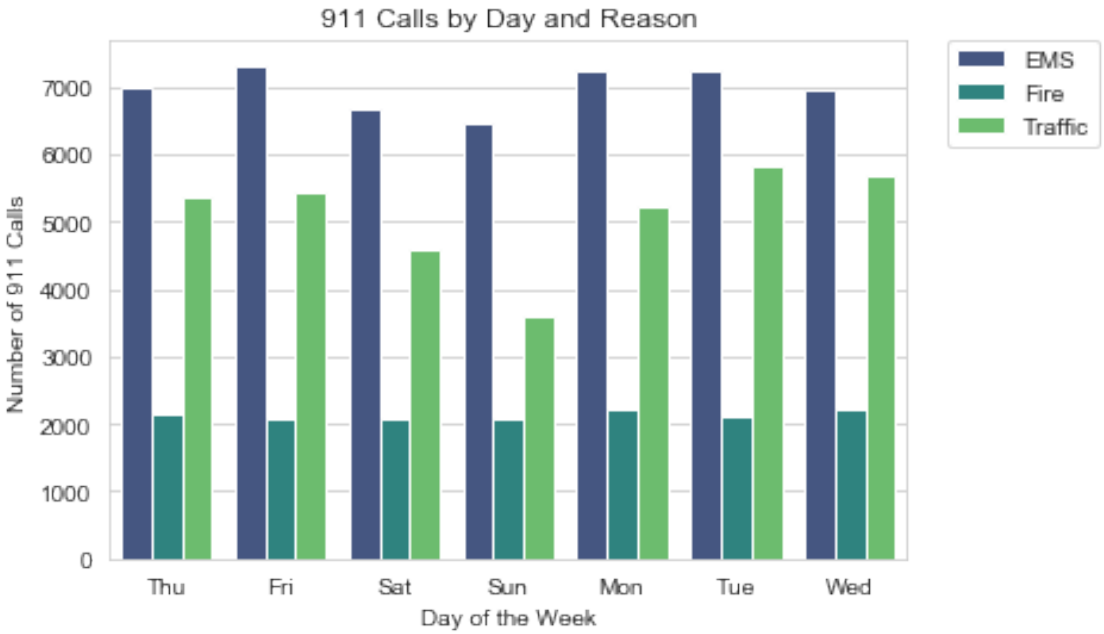



```
In [10]: df['timeStamp'] = pd.to_datetime(df['timeStamp'])
         # converting "timeStamp" elements from string objects to datetime objects

In [11]: df['Hour'] = df['timeStamp'].apply(lambda time: time.hour)
         df['Month'] = df['timeStamp'].apply(lambda time: time.month)
         df['Day of Week'] = df['timeStamp'].apply(lambda time: time.dayofweek)
         # creating new columns for each of the time elements
```

In [12]:
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
df['Day of Week'] = df['Day of Week'].map(dmap)
# mapping integers to the actual day of the week
```

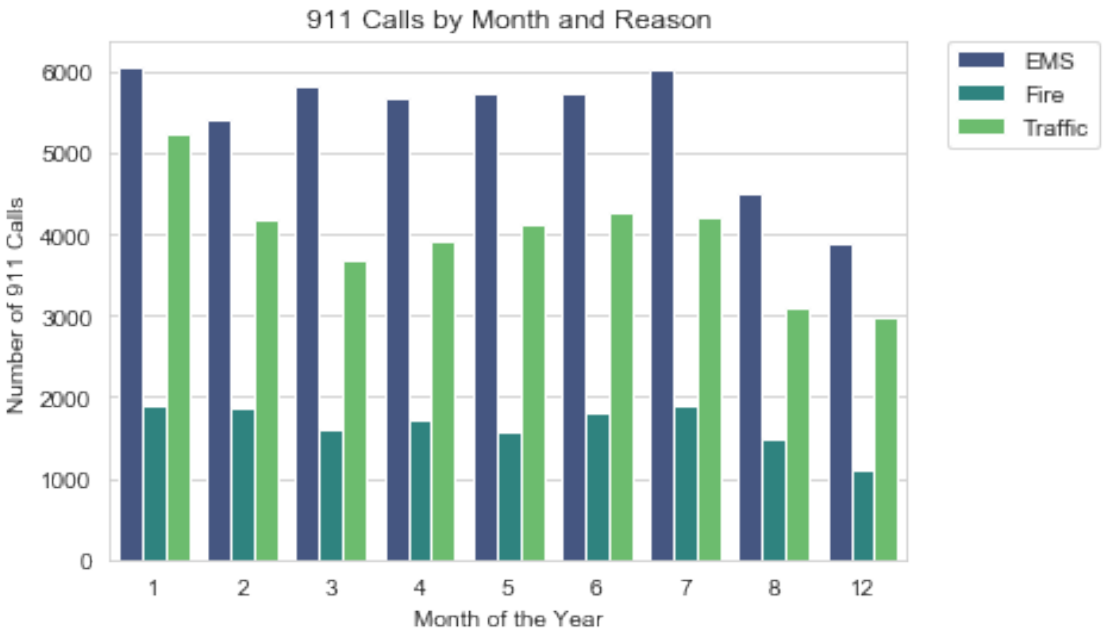
In [13]:
```
sns.countplot(x='Day of Week',data=df,hue='Reason',palette='viridis')
# visualizing number of 911 calls by day, and split by reason

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
# to relocate the legend
plt.title("911 Calls by Day and Reason")
plt.xlabel("Day of the Week")
plt.ylabel("Number of 911 Calls")

# seems like # of fire-related calls is the same every day
```

Out[13]:
```
Text(0, 0.5, 'Number of 911 Calls')
```



In [14]:
```
sns.countplot(x='Month',data=df,hue='Reason',palette='viridis')
# visualizing number of 911 calls by month, and split by reason

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
# to relocate the legend
plt.title("911 Calls by Month and Reason")
plt.xlabel("Month of the Year")
plt.ylabel("Number of 911 Calls")

# seems like # of EMS/traffic related 911 calls decreases through the year
# months 9,10, and 11 do not show up - let's explore this further
```

Out[14]: Text(0, 0.5, 'Number of 911 Calls')



In [15]:
```
byMonth = df.groupby('Month').count()
byMonth
```

Out[15]:

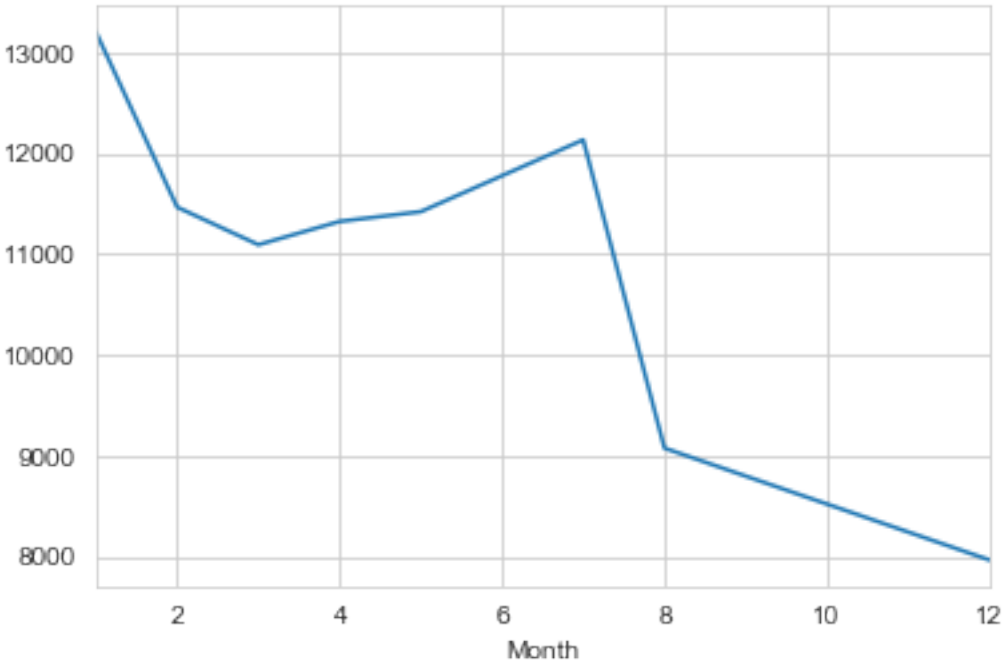
Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e
1	13205	13205	13205	11527	13205	13205	13203	13096	13205
2	11467	11467	11467	9930	11467	11467	11465	11396	11467
3	11101	11101	11101	9755	11101	11101	11092	11059	11101
4	11326	11326	11326	9895	11326	11326	11323	11283	11326
5	11423	11423	11423	9946	11423	11423	11420	11378	11423
6	11786	11786	11786	10212	11786	11786	11777	11732	11786
7	12137	12137	12137	10633	12137	12137	12133	12088	12137
8	9078	9078	9078	7832	9078	9078	9073	9025	9078
12	7969	7969	7969	6907	7969	7969	7963	7916	7969

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
6	11786	11786	11786
7	12137	12137	12137

```
8        9078   9078          9078
12       7969   7969          7969
```

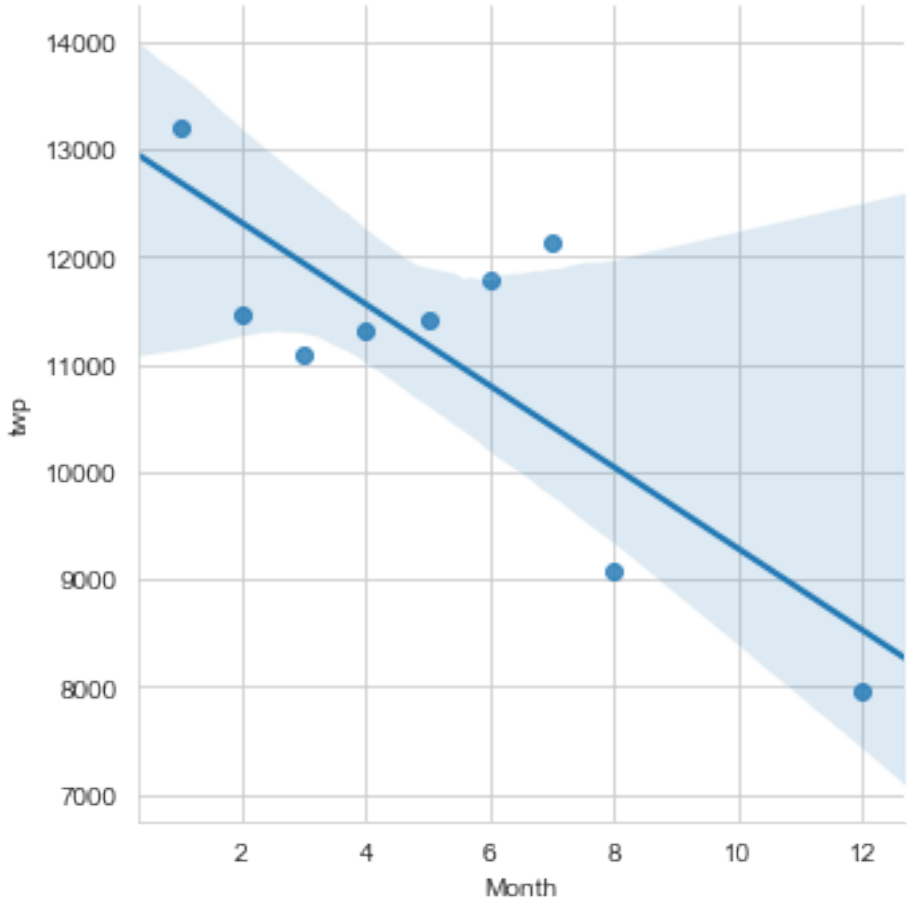
In [16]:
```
byMonth['twp'].plot()
# plotting line graph of count of 911 calls by month
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2699ee48>
```

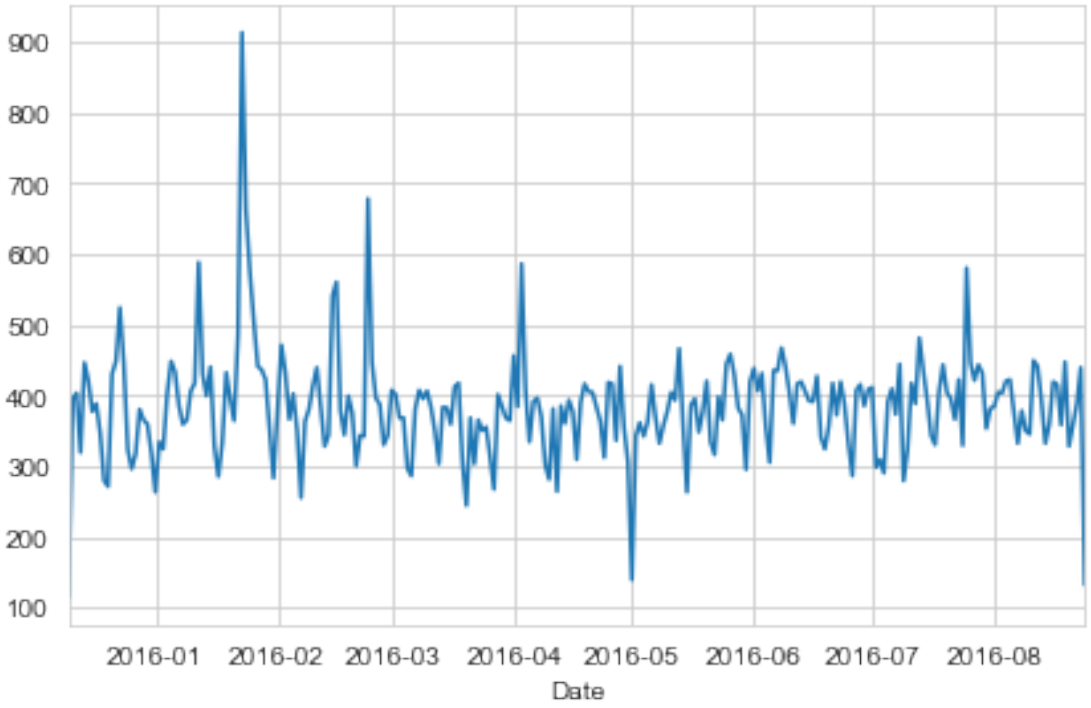


In [17]:
```
sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())
# plotting linear fit on # of calls by month
```

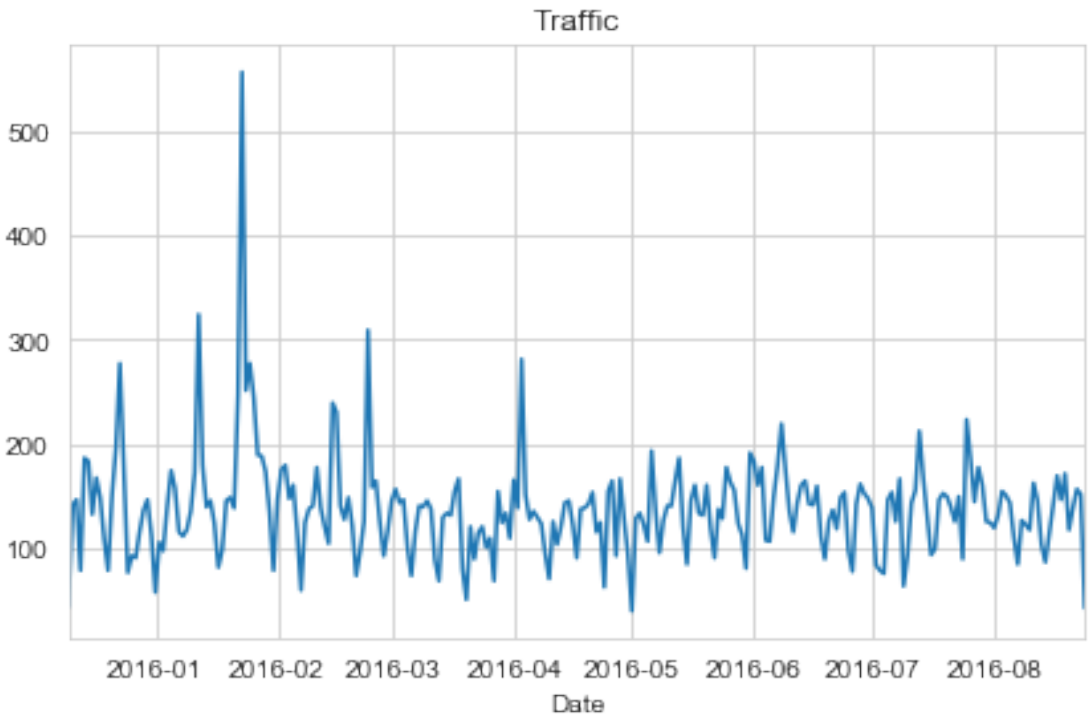
Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x1a254fe128>
```



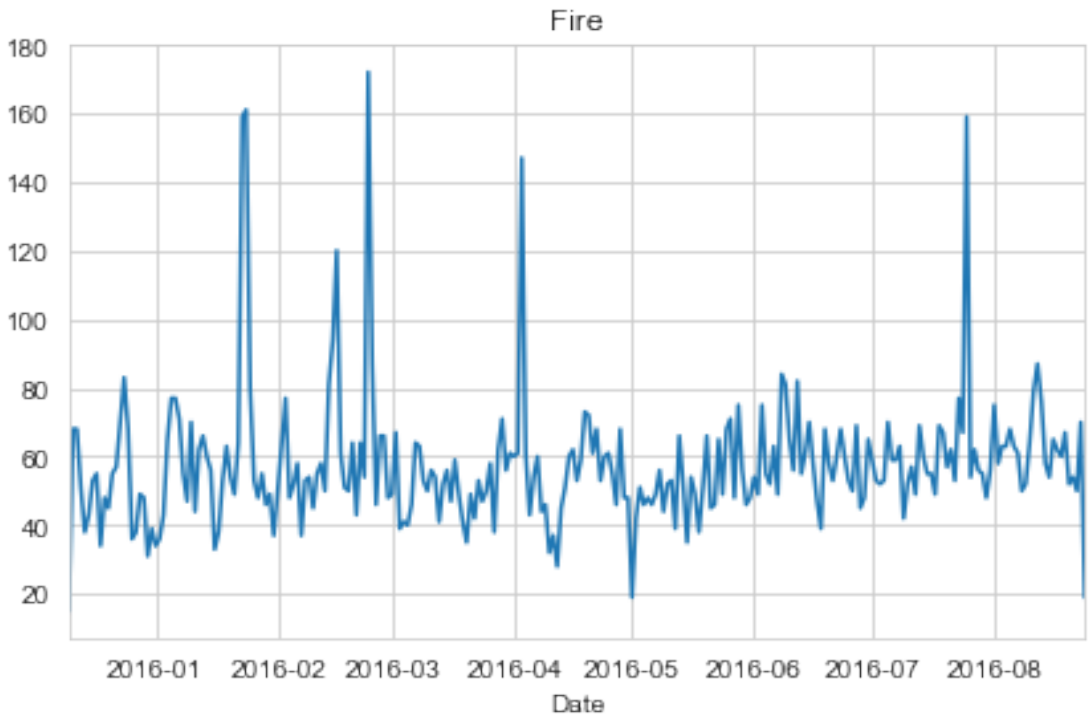
In [18]:
```
df['Date']=df['timeStamp'].apply(lambda t: t.date())
# creating new column that contains date from timestamp column

df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
# create a plot of 911 call counts after grouping by date
```



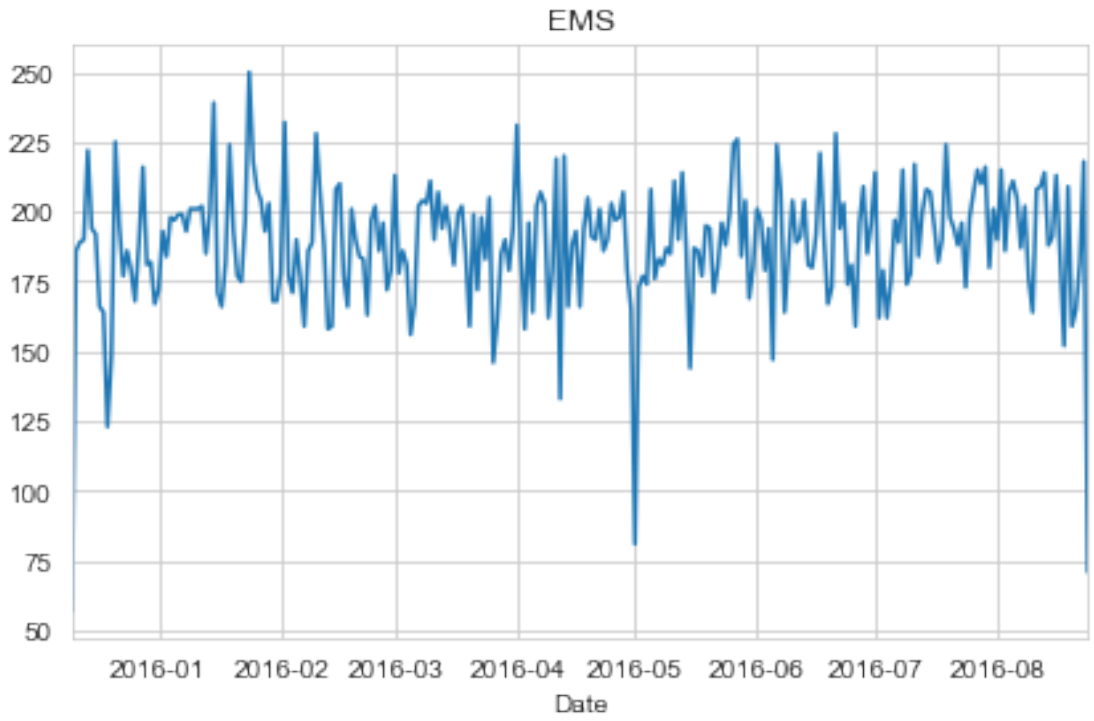
In [19]:
```
df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()
plt.title('Traffic')
plt.tight_layout()
# Traffic related calls
```




```
In [20]: df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
         plt.title('Fire')
         plt.tight_layout()
         # fire related calls
```




```
In [21]: df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()
         plt.title('EMS')
         plt.tight_layout()
         # EMS related calls
```



1.0.3 Creating Heatmaps

In [22]:
```
dayHour = df.groupby(by=['Day of Week','Hour']).count()['Reason'].unstack()
dayHour.head()
# create new dataframe so that columns become hours and index becomes day of the week
```

Out[22]:

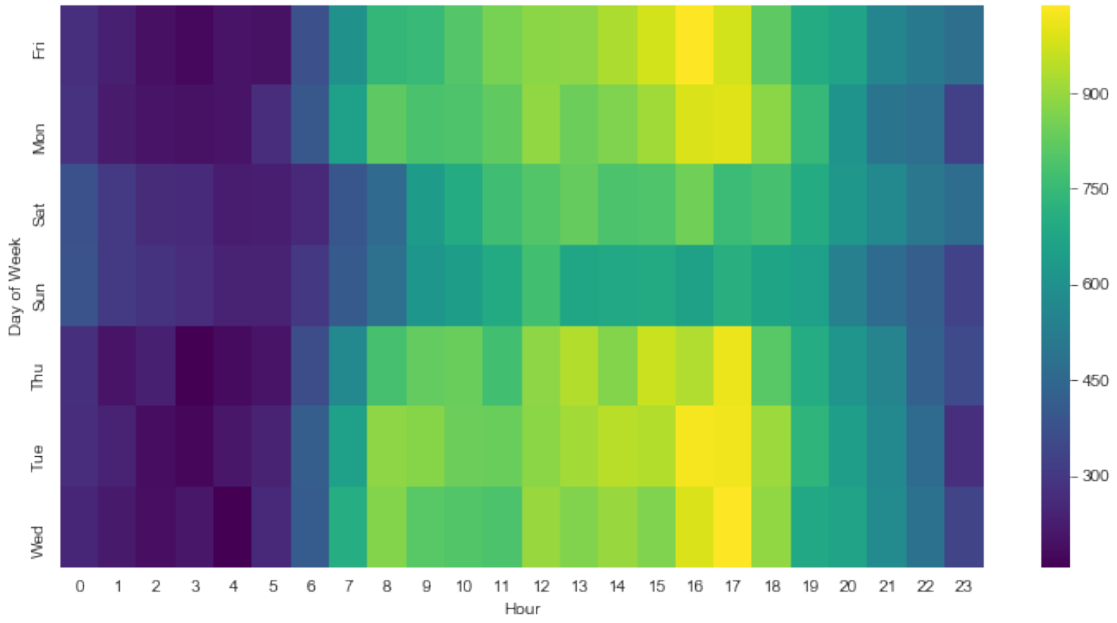
Hour	0	1	2	3	4	5	6	7	8	9	...	14	15
Day of Week											...		
Fri	275	235	191	175	201	194	372	598	742	752	...	932	980
Mon	282	221	201	194	204	267	397	653	819	786	...	869	913
Sat	375	301	263	260	224	231	257	391	459	640	...	789	796
Sun	383	306	286	268	242	240	300	402	483	620	...	684	691
Thu	278	202	233	159	182	203	362	570	777	828	...	876	969

Hour	16	17	18	19	20	21	22	23
Day of Week								
Fri	1039	980	820	696	667	559	514	474
Mon	989	997	885	746	613	497	472	325
Sat	848	757	778	696	628	572	506	467
Sun	663	714	670	655	537	461	415	330
Thu	935	1013	810	698	617	553	424	354

[5 rows x 24 columns]

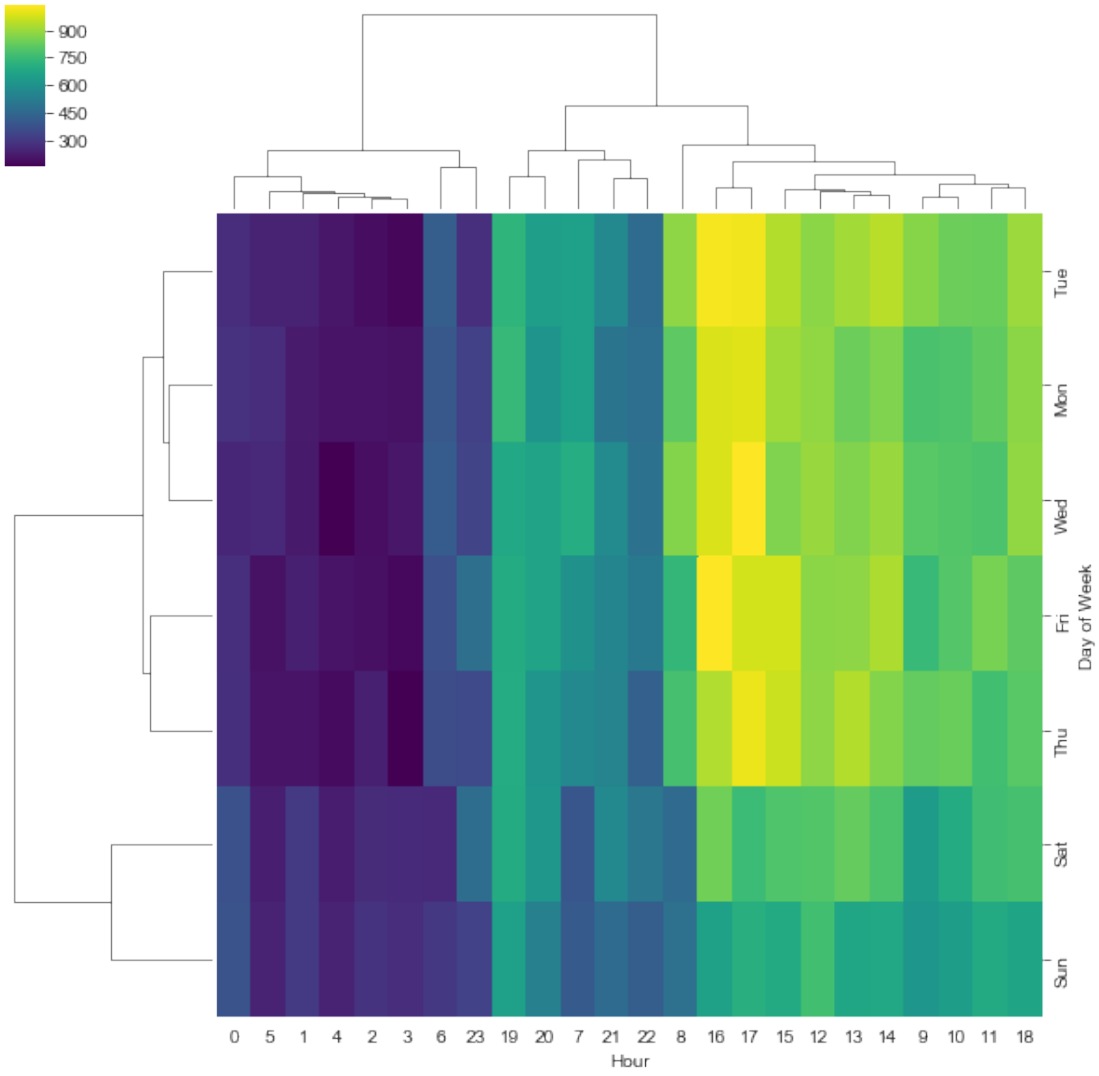
In [23]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayHour,cmap='viridis')
# more 911 calls occur generally in the afternoon/evenings
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26132320>
```



In [24]:
```
sns.clustermap(dayHour,cmap='viridis')
# creating clustermap
```

Out[24]:
```
<seaborn.matrix.ClusterGrid at 0x1a25d99320>
```



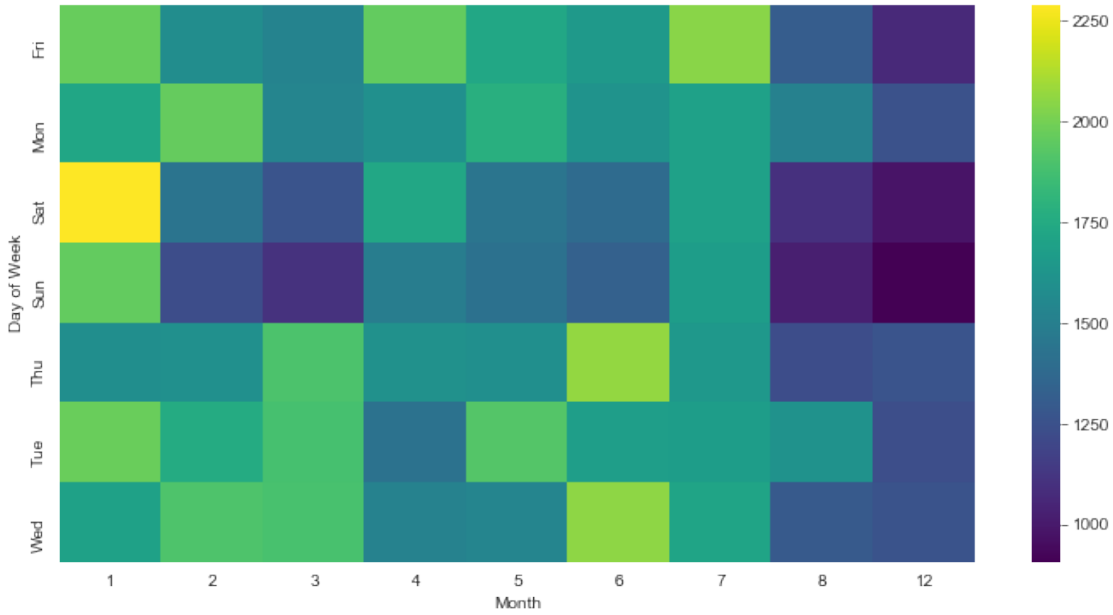
In [25]:
```
dayMonth = df.groupby(by=['Day of Week','Month']).count()['Reason'].unstack()
dayMonth
# create new dataframe so that columns become months and index becomes day of the week
```

Out[25]:

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
Tue	1973	1753	1884	1430	1918	1676	1670	1612	1234
Wed	1700	1903	1889	1517	1538	2058	1717	1295	1262

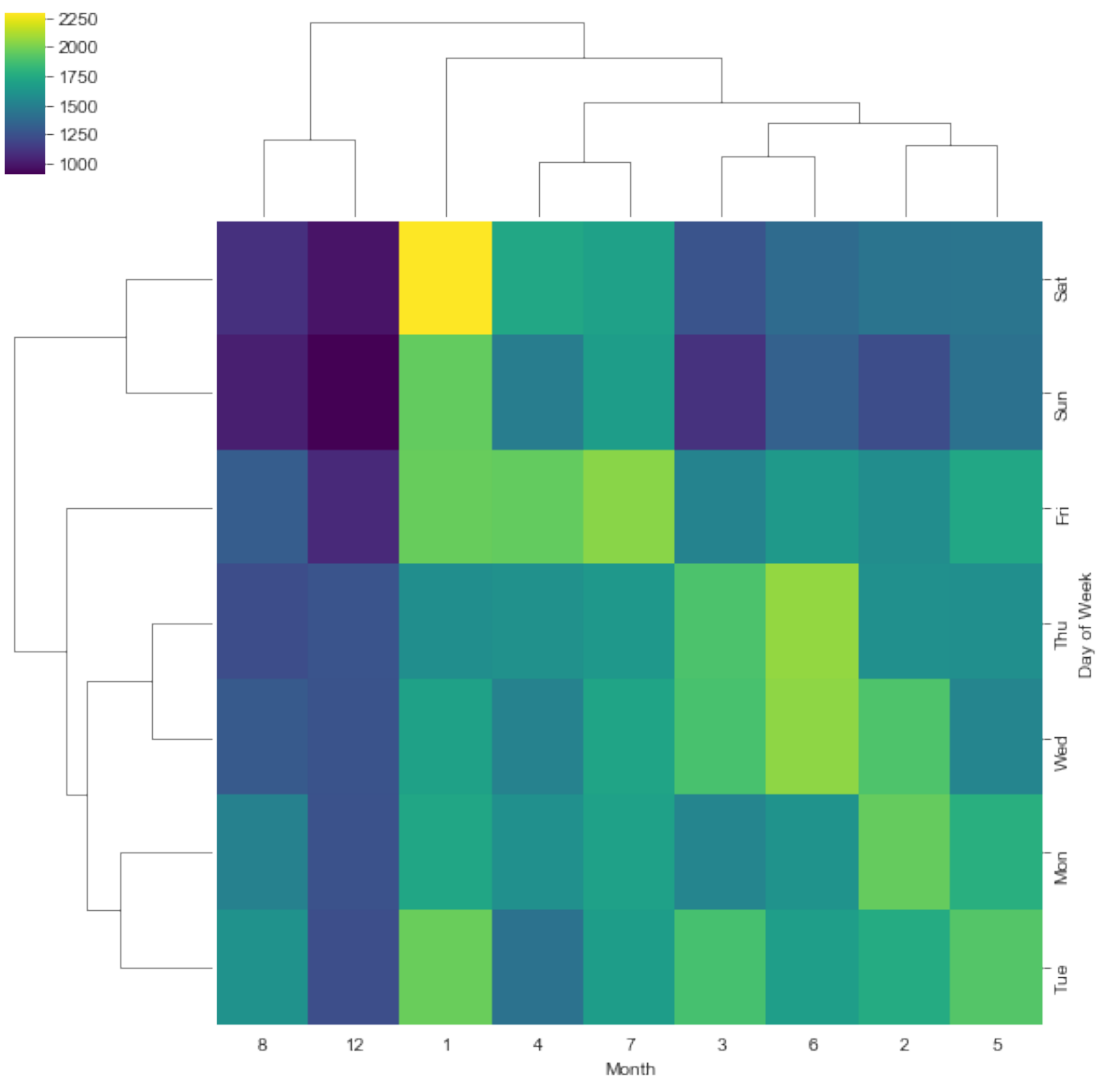
In [26]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayMonth,cmap='viridis')
# no clear patterns
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1a25e4d2b0>



In [27]:
```
sns.clustermap(dayMonth,cmap='viridis')
# clustermap
```

Out[27]: <seaborn.matrix.ClusterGrid at 0x1a26202518>


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