911 Calls Capstone Project - Saurabh Gadre

For this capstone project we will be analyzing 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)

Data and Setup

Import numpy and pandas

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

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

Read in the csv file as a dataframe called df

In [4]:
```
df = pd.read_csv('911.csv')
```

Check the info() of the df

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99492 entries, 0 to 99491
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   lat        99492 non-null  float64
 1   lng        99492 non-null  float64
 2   desc       99492 non-null  object 
 3   zip        86637 non-null  float64
 4   title      99492 non-null  object 
 5   timeStamp  99492 non-null  object 
 6   twp        99449 non-null  object 
 7   addr       98973 non-null  object 
 8   e          99492 non-null  int64  
dtypes: float64(3), int64(1), object(5)
memory usage: 6.8+ MB
```

Check the head of df

In [6]:
```
df.head(3)
```

Out[6]:

	lat	lng	desc	zip	title	timeStamp	twp	ad
0	40.297876	-75.581294	REINDEER CT & DEAD END; NEW HANOVER; Station ...	19525.0	EMS: BACK PAINS/INJURY	2015-12-10 17:40:00	NEW HANOVER	REINDEER & DEAD EN
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:40:00	HATFIELD TOWNSHIP	BRIAR PA WHITEMAR
2	40.121182	-75.351975	HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...	19401.0	Fire: GAS-ODOR/LEAK	2015-12-10 17:40:00	NORRISTOWN	HAWS A

Basic Analysis

Top 5 zipcodes for 911 calls

In [7]:
```
df['zip'].value_counts().head(5)
```

Out[7]:
```
19401.0    6979
19464.0    6643
19403.0    4854
19446.0    4748
19406.0    3174
Name: zip, dtype: int64
```

Top 5 townships (twp) for 911 calls

In [30]:
```
df['twp'].value_counts().head(5)
```

Out[30]:
```
LOWER MERION    8443
ABINGTON        5977
NORRISTOWN      5890
UPPER MERION    5227
CHELTENHAM      4575
Name: twp, dtype: int64
```

Number of Unique Title Codes

In [8]:
```
df['title'].nunique()
```

Out[8]:
```
110
```

Creating new features

Most common reason for a 911 call, using .apply() function.

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

In [10]:
```
df['Reason'] = df['title'].apply(lambda title: title.split(':')[0])
```

Most common Reason for a 911 call

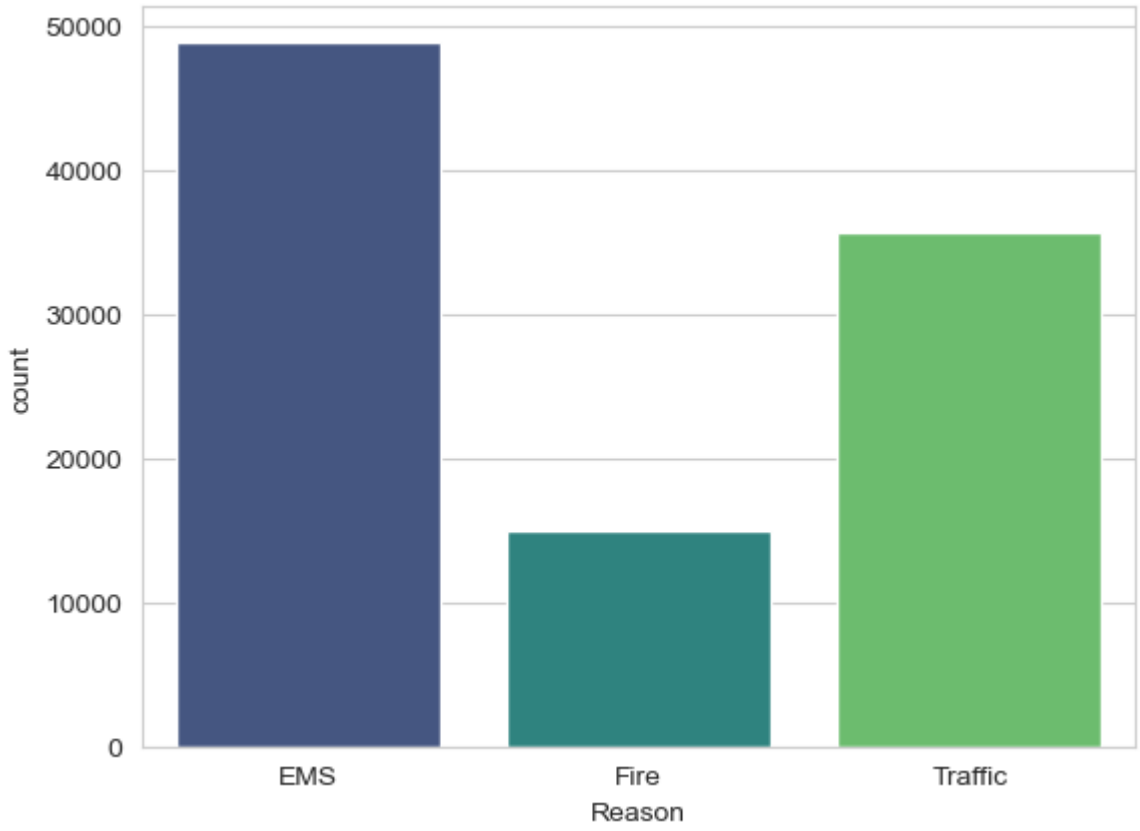
In [11]:
```
df['Reason'].value_counts()
```

Out[11]:
```
EMS        48877
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

Seaborn - Countplot of 911 calls by Reason.

In [12]:
```
sns.countplot(x='Reason',data=df,palette='viridis')
```

Out[12]:
```
<AxesSubplot:xlabel='Reason', ylabel='count'>
```



Data type of the objects in the timeStamp column?

In [13]:
```
type(df['timeStamp'].iloc[0])
```

Out[13]:
```
str
```

** Convert timestamps to datetime

In [15]:
```
df['timeStamp'] = pd.to_datetime(df['timeStamp'])
```

** Use .apply() to create 3 new columns called Hour, Month, and Day of Week.

In [16]:
```
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)
```

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 [17]:
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
```

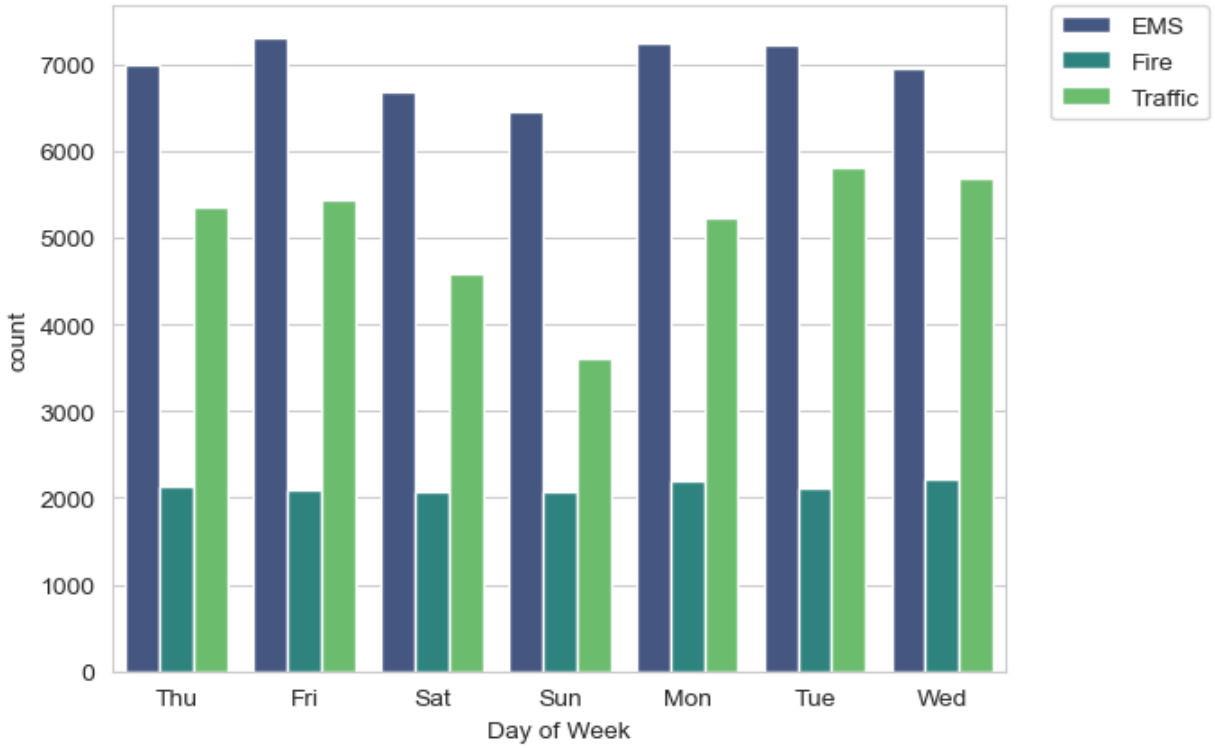
In [18]:
```
df['Day of Week'] = df['Day of Week'].map(dmap)
```

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

In [19]:
```
sns.countplot(x='Day of Week',data=df,hue='Reason',palette='viridis')

# To relocate the legend
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[19]:
```
<matplotlib.legend.Legend at 0x225802fb040>
```

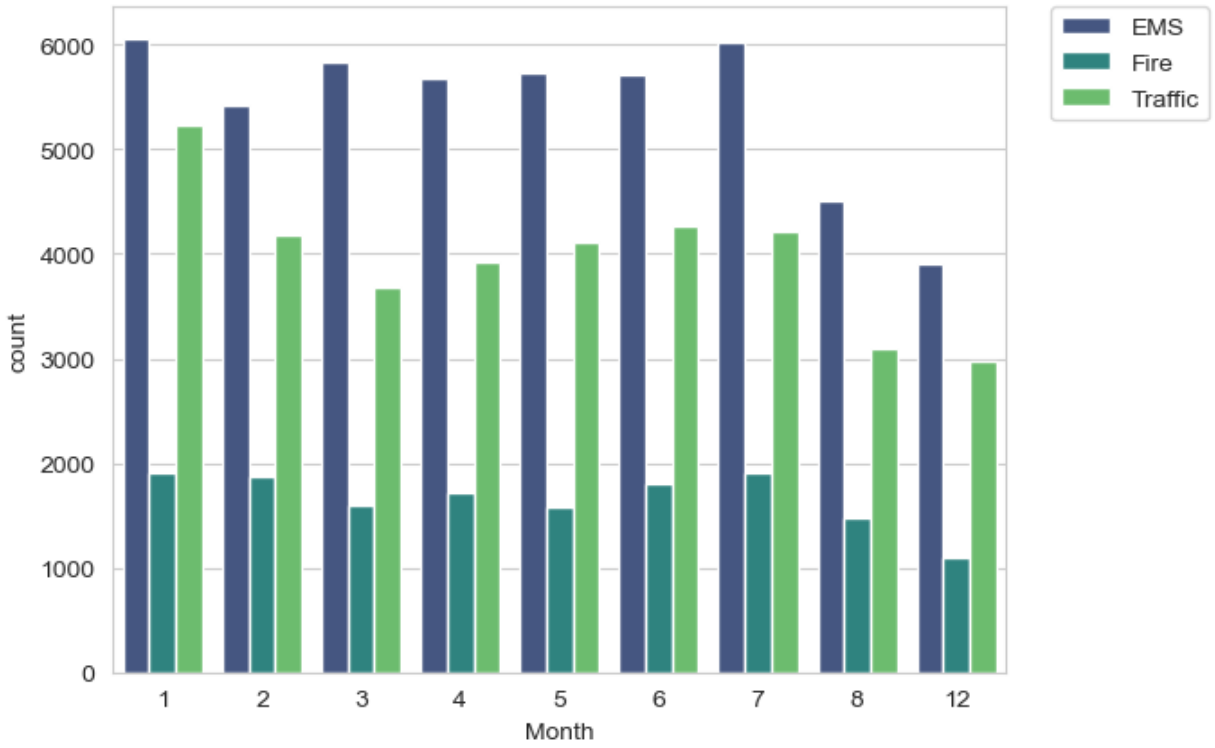


Same for Month:

In [20]:
```
sns.countplot(x='Month',data=df,hue='Reason',palette='viridis')

# To relocate the legend
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[20]:
```
<matplotlib.legend.Legend at 0x22580d312e0>
```



In [42]:
```
# It is missing some months! 9,10, and 11 are not there.
```

It was missing some Months, so I'll make a simple line plot that fills in the missing months.

** Gropuby object called byMonth, where I group the DataFrame by the month column and use the count() method for aggregation.

In [21]:
```
byMonth = df.groupby('Month').count()
byMonth.head()
```

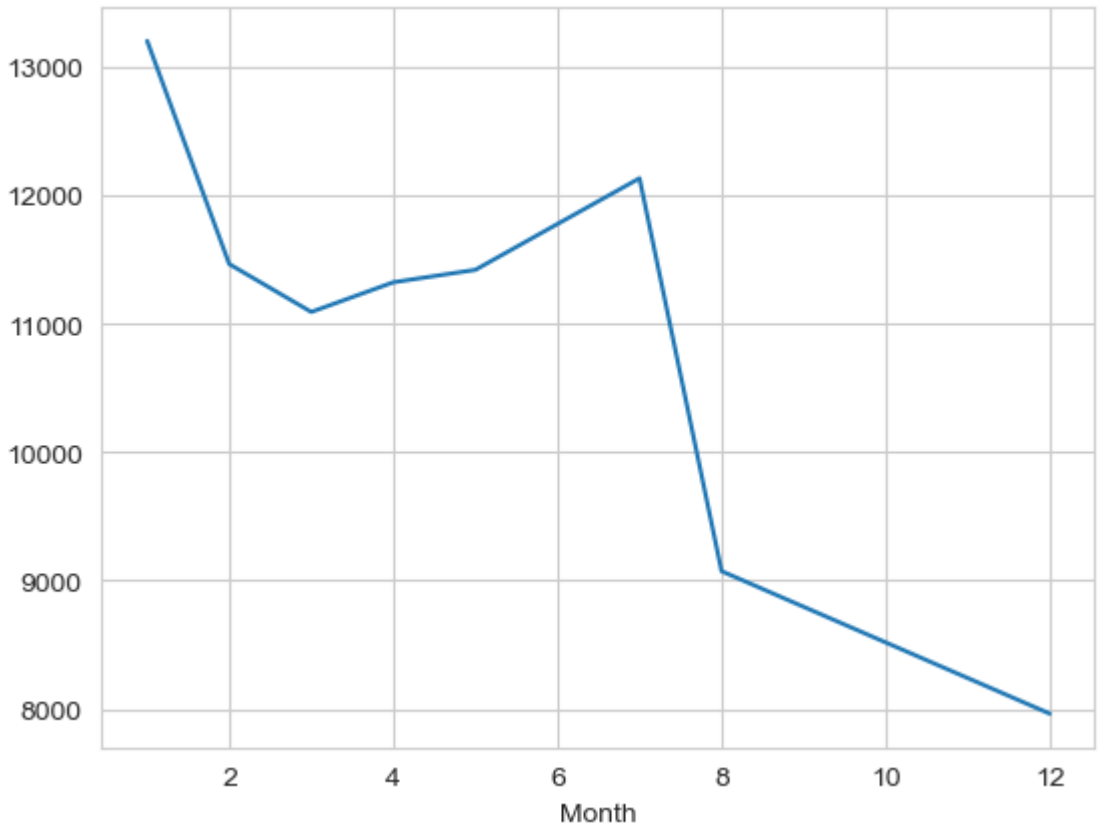
Out[21]:

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

Simple plot off of the dataframe indicating the count of calls per month.

In [22]:
```
# Could be any column
byMonth['twp'].plot()
```

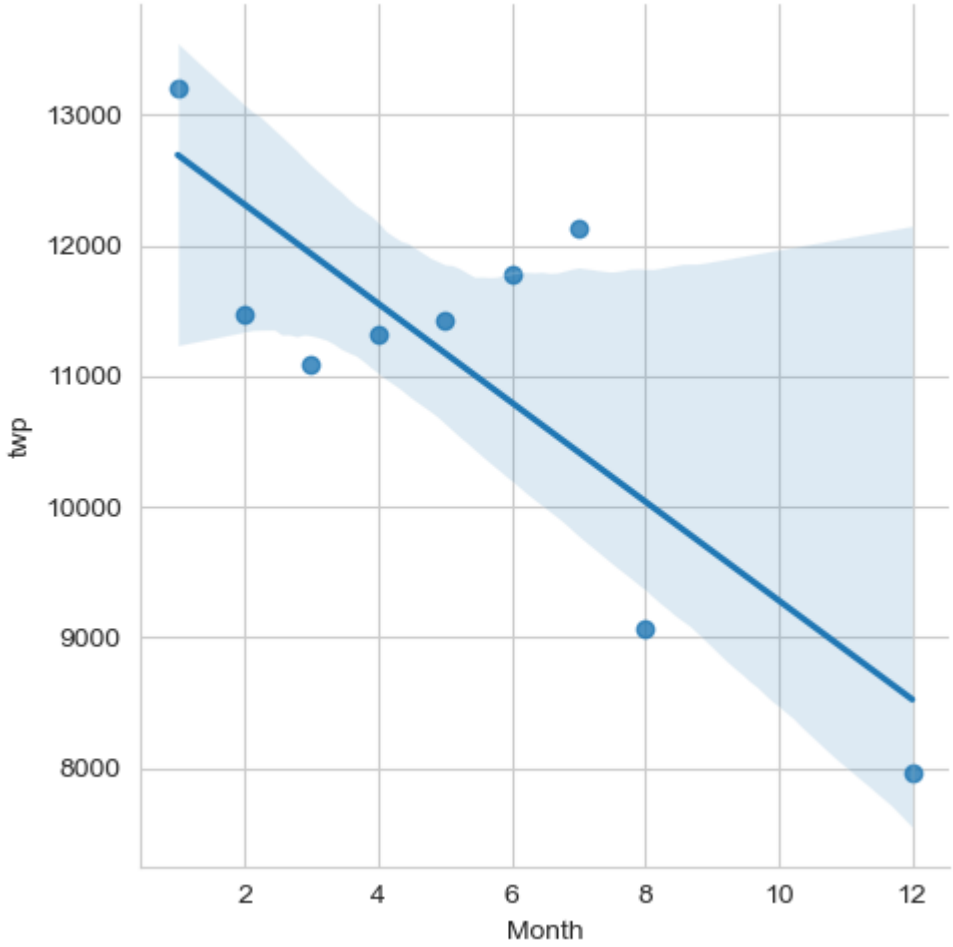
Out[22]: <AxesSubplot:xlabel='Month'>



** Seaborn's lmplot() to create a linear fit on the number of calls per month.

In [23]:
```
sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x22580db8910>

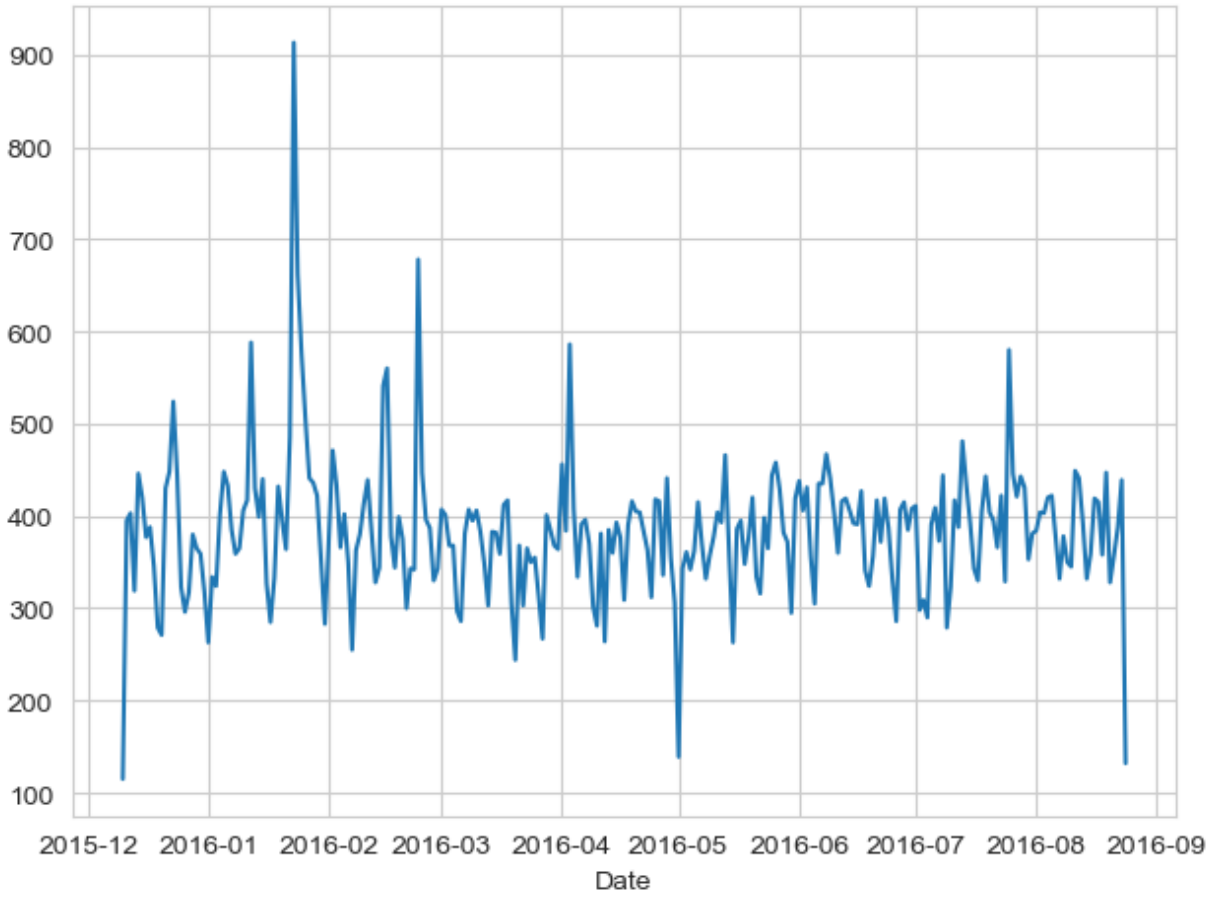


**A new column called 'Date' that contains the date from the timeStamp column.

In [24]:
```
df['Date']=df['timeStamp'].apply(lambda t: t.date())
```

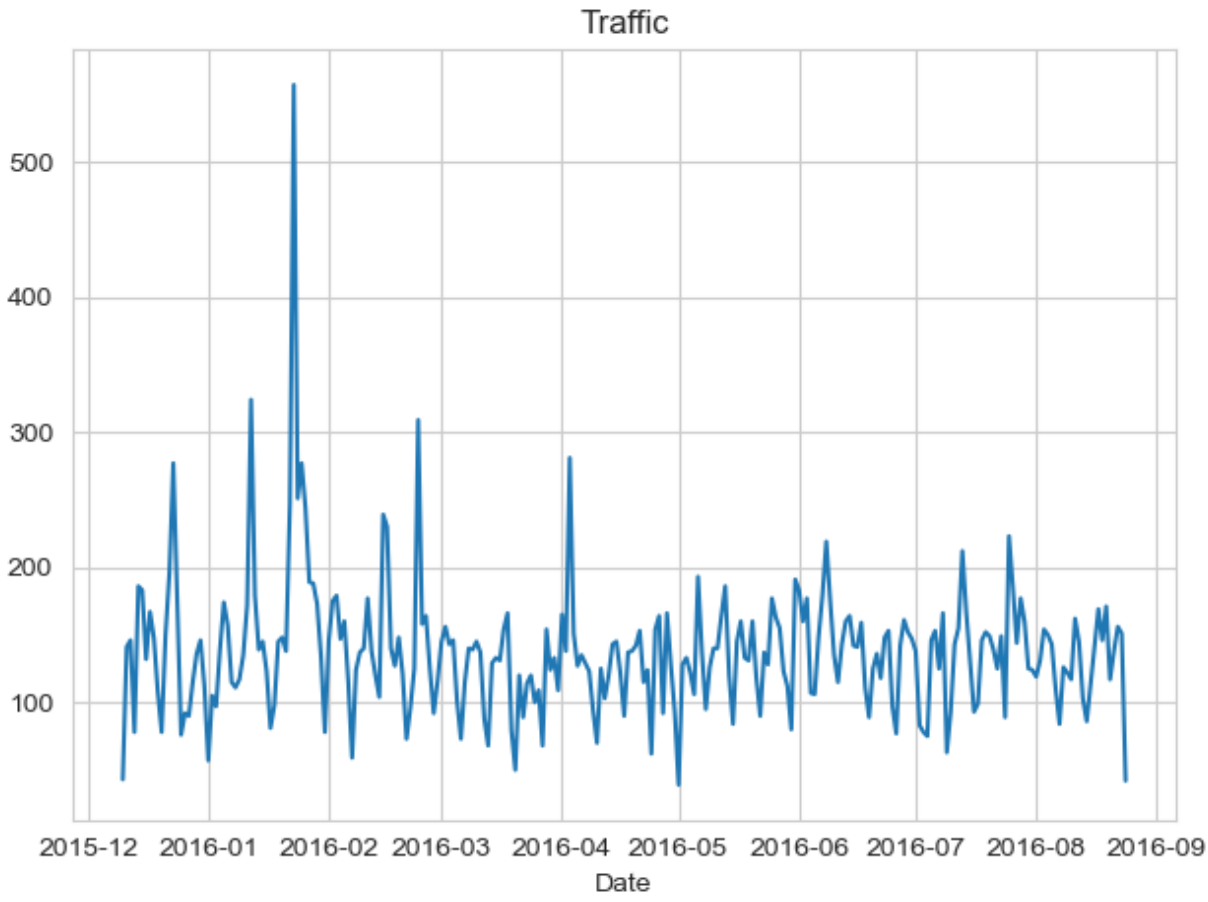
Groupby this Date column with the count() aggregate and create a plot of counts of 911 calls.

In [25]:
```
df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
```

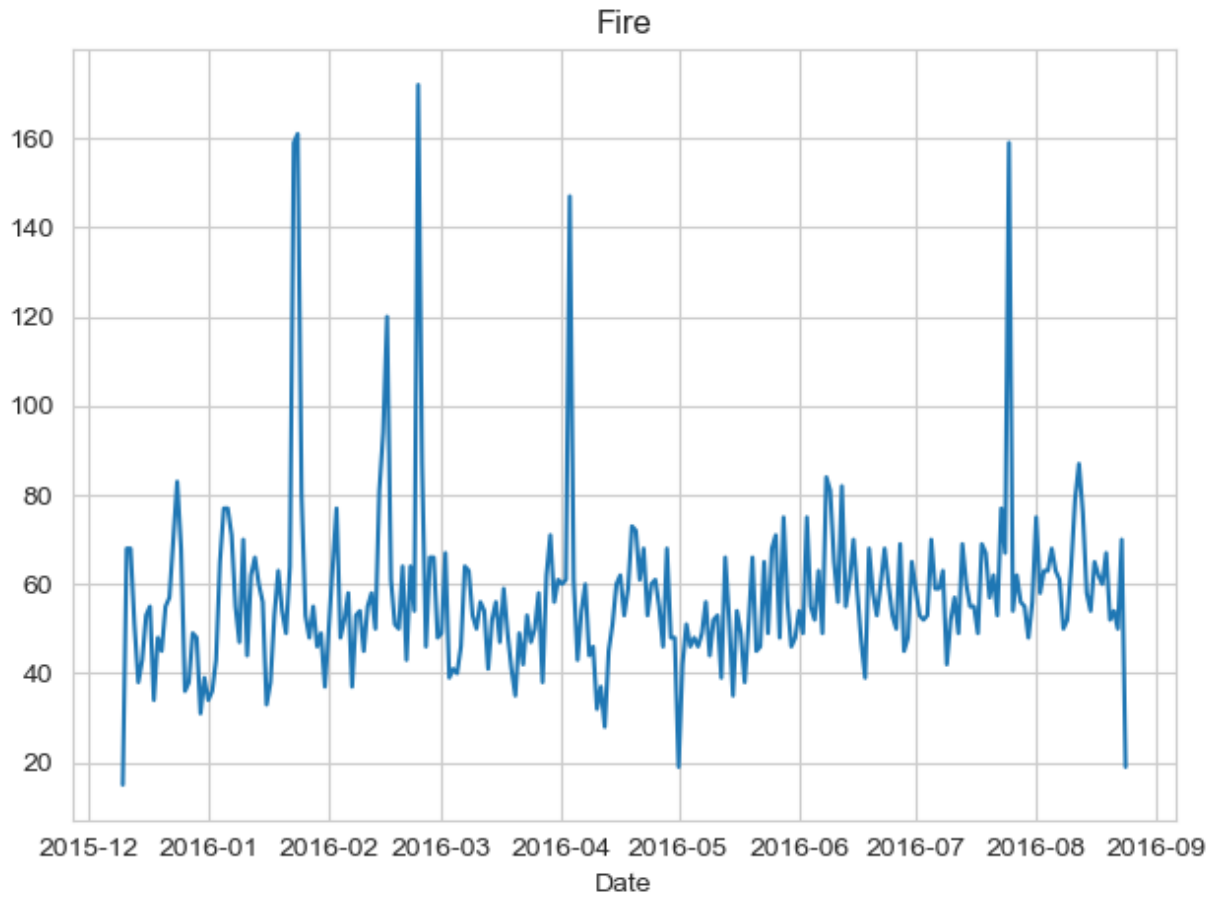


** Recreate this plot with 3 separate plots representing a Reason for the call.

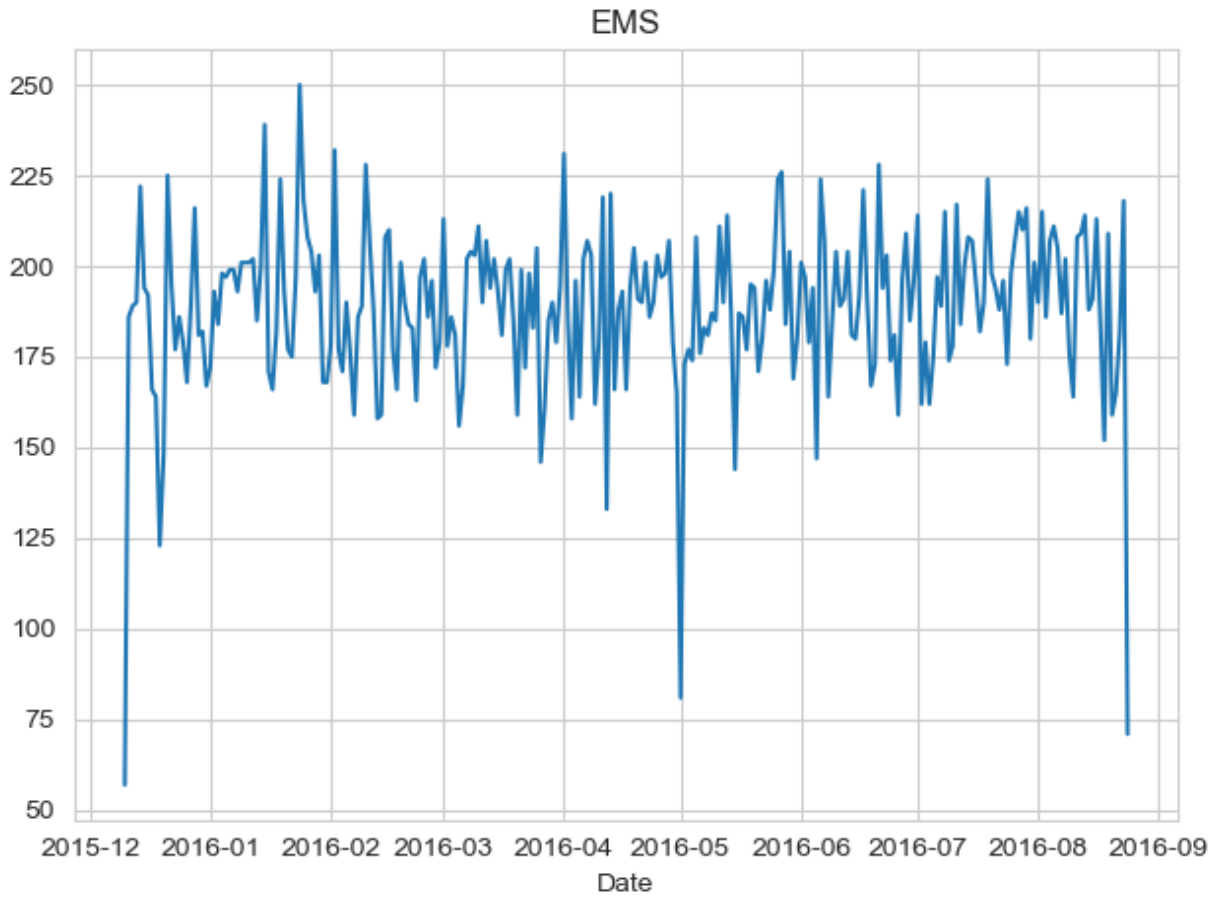
In [26]:
```
df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()
plt.title('Traffic')
plt.tight_layout()
```



In [27]:
```
df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
plt.title('Fire')
plt.tight_layout()
```



In [28]:
```
df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()
plt.title('EMS')
plt.tight_layout()
```



** Heatmaps with seaborn and our data.

In [29]:
```
dayHour = df.groupby(by=['Day of Week','Hour']).count()['Reason'].unstack()
dayHour.head()
```

Out[29]:

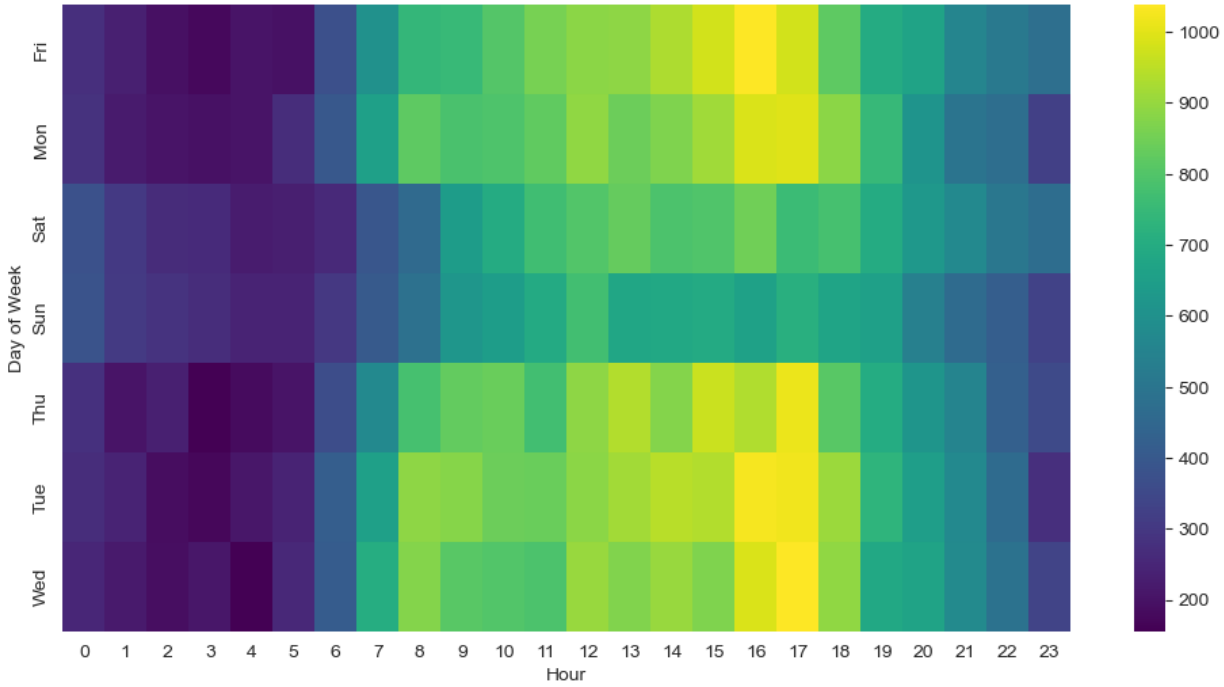
Day of Week \ Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20
Fri	275	235	191	175	201	194	372	598	742	752	...	932	980	1039	980	820	696	667
Mon	282	221	201	194	204	267	397	653	819	786	...	869	913	989	997	885	746	613
Sat	375	301	263	260	224	231	257	391	459	640	...	789	796	848	757	778	696	628
Sun	383	306	286	268	242	240	300	402	483	620	...	684	691	663	714	670	655	537
Thu	278	202	233	159	182	203	362	570	777	828	...	876	969	935	1013	810	698	617

5 rows × 24 columns

HeatMap using this new DataFrame.

In [30]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayHour,cmap='viridis')
```

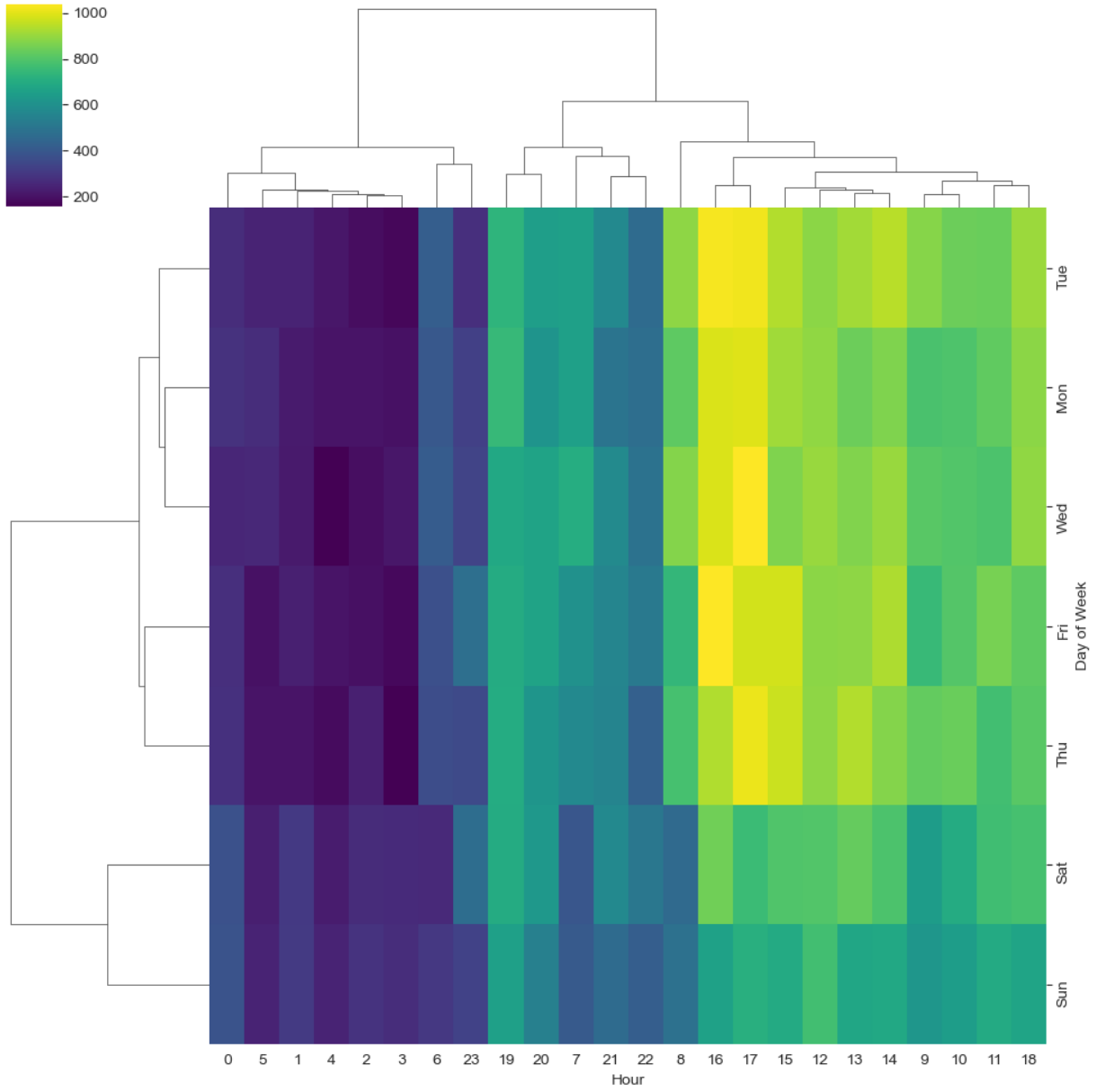
Out[30]:
```
<AxesSubplot:xlabel='Hour', ylabel='Day of Week'>
```



Clustermap using this DataFrame.

In [31]:
```
sns.clustermap(dayHour,cmap='viridis')
```

Out[31]:
```
<seaborn.matrix.ClusterGrid at 0x22580302eb0>
```



Same plots and operations, for a DataFrame that shows the Month as the column.

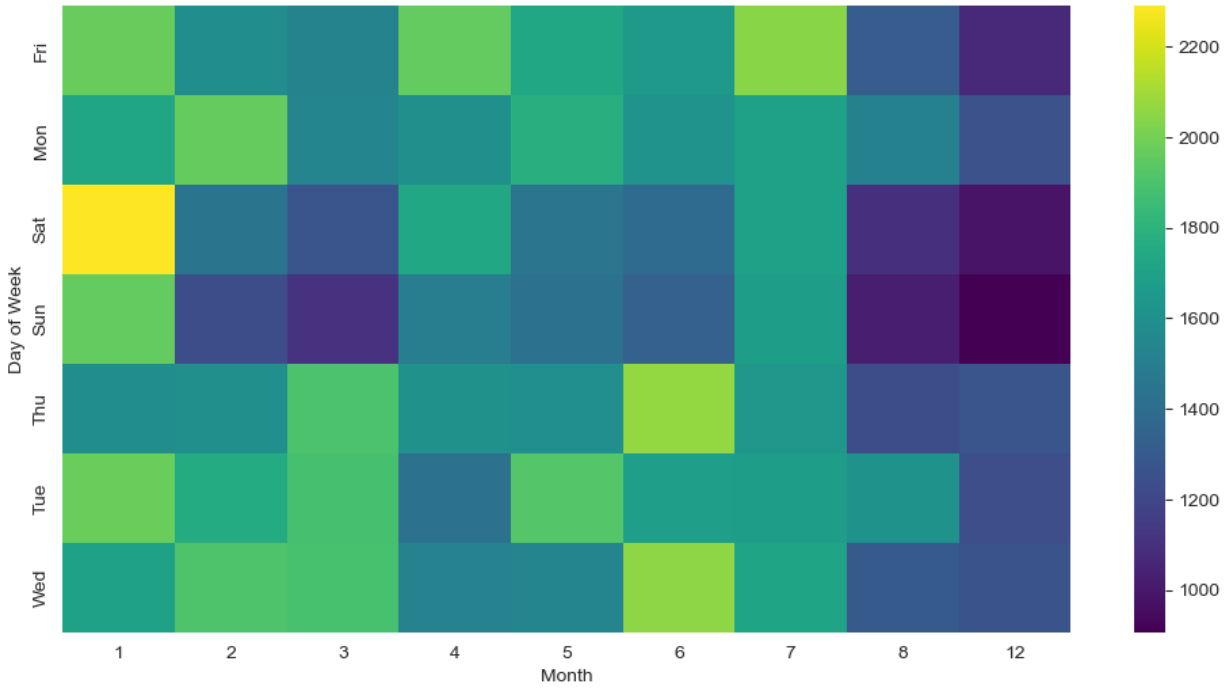
In [32]:
```
dayMonth = df.groupby(by=['Day of Week','Month']).count()['Reason'].unstack()
dayMonth.head()
```

Out[32]:

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

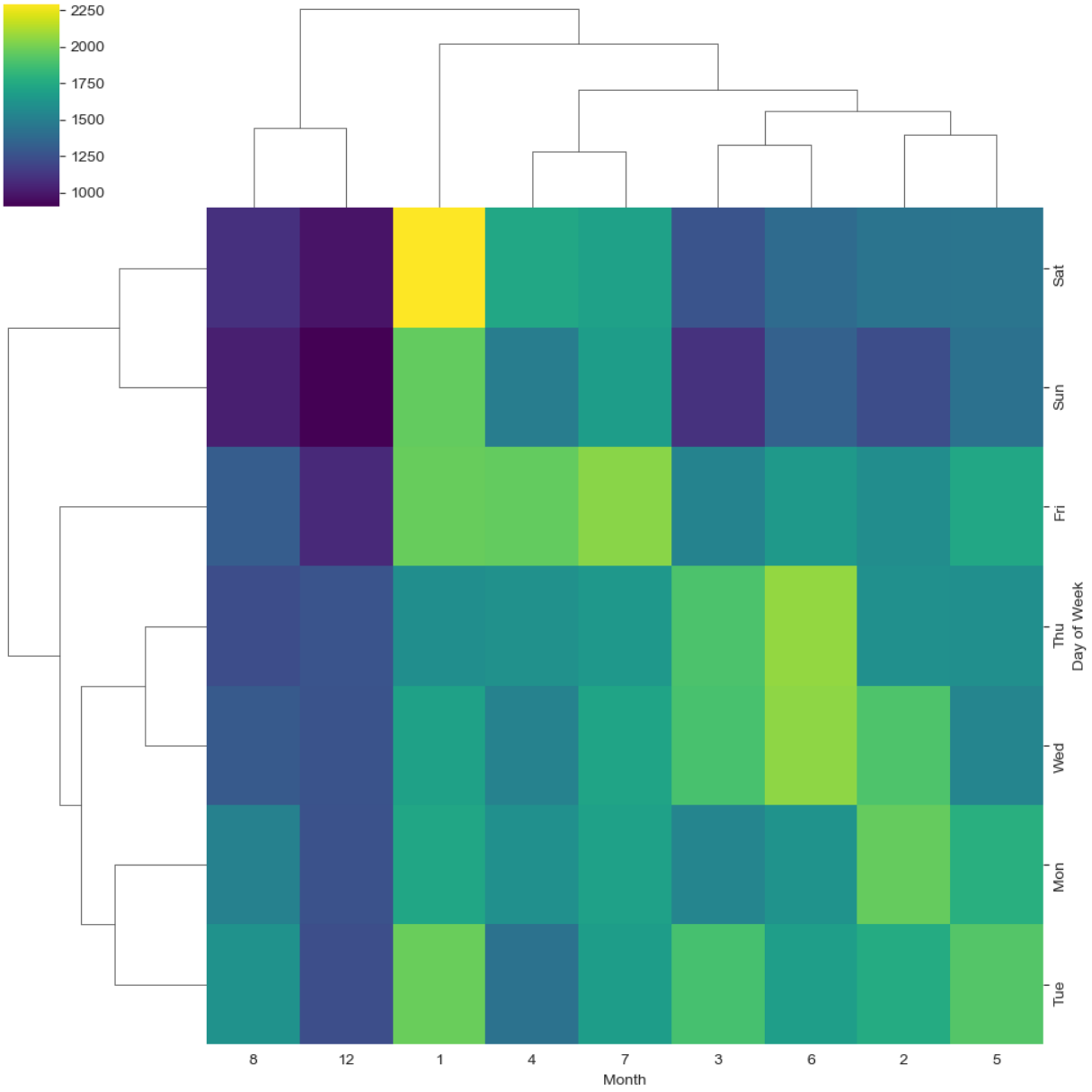
In [33]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayMonth,cmap='viridis')
```

Out[33]: <AxesSubplot:xlabel='Month', ylabel='Day of Week'>



In [34]:
```
sns.clustermap(dayMonth,cmap='viridis')
```

Out[34]: <seaborn.matrix.ClusterGrid at 0x2258031d5b0>


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