911 Calls Capstone Project - Solutions

For this capstone project we will be analyzing some 911 call data from Kaggle (https://www.kaggle.com/mchirico/montcoalert). 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)

Just go along with this notebook and try to complete the instructions or answer the questions in bold using your Python and Data Science skills!

Data and Setup

** Import numpy and pandas **

In [1]:

```
import numpy as np
import pandas as pd
```

** Import visualization libraries and set %matplotlib inline. **

In [2]:

```
import numpy as np
import pandas as pd

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 [3]:

```
df = pd.read_csv('911.csv')
```

** Check the info() of the df **

In [4]:
```
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
```

** Check the head of df **

In [5]:
```
df.head(3)
```

Out[5]:

	lat	lng	desc	zip	title	timeStamp	twp	
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	REINDI & DEA
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 WHITEI
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	HAV

Basic Questions

** What are the top 5 zipcodes for 911 calls? **

In [29]:
```
df['zip'].value_counts().head(5)
```

Out[29]:
```
19401.0    6979
19464.0    6643
19403.0    4854
19446.0    4748
19406.0    3174
Name: zip, dtype: int64
```

** What are the 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
```

** Take a look at the 'title' column, how many unique title codes are there? **

In [31]:
```
df['title'].nunique()
```

Out[31]:
```
110
```

Creating new features

** In the titles column there are "Reasons/Departments" specified before the title code. These are EMS, Fire, and Traffic. Use .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 [32]:
```
df['Reason'] = df['title'].apply(lambda title: title.split(':')[0])
```

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

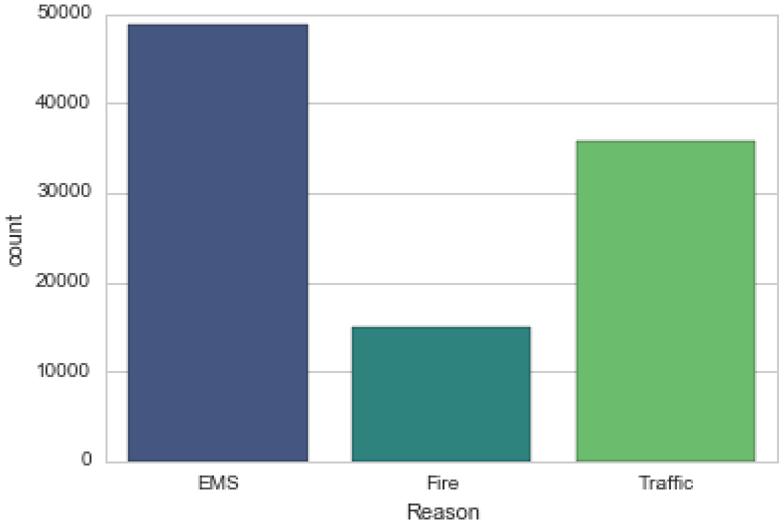
In [33]:
```
df['Reason'].value_counts()
```

Out[33]:
```
EMS        48877
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

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

```
In [34]: sns.countplot(x='Reason',data=df,palette='viridis')
```

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x121757b70>
```



** Now let us begin to focus on time information. What is the data type of the objects in the timeStamp column? **

```
In [35]: type(df['timeStamp'].iloc[0])
```

```
Out[35]: str
```

** You should have seen that these timestamps are still strings. Use [pd.to_datetime (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html) to convert the column from strings to DateTime objects. **

```
In [10]: df['timeStamp'] = pd.to_datetime(df['timeStamp'])
```

```
In [37]: 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)
```

In [38]:

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

In [39]:

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

** Now use seaborn to create a countplot of the Day of Week column with the hue based off of the Reason column. **

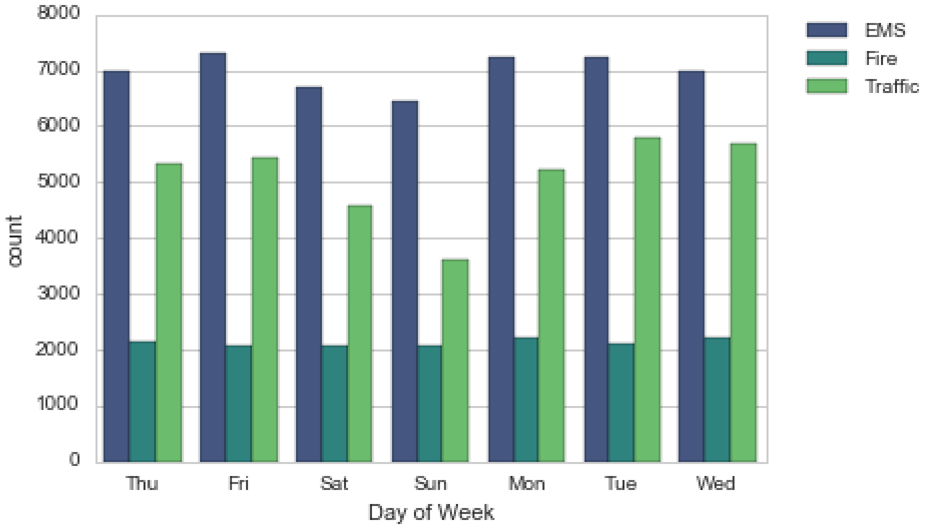
In [40]:

```
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[40]:

```
<matplotlib.legend.Legend at 0x121762710>
```

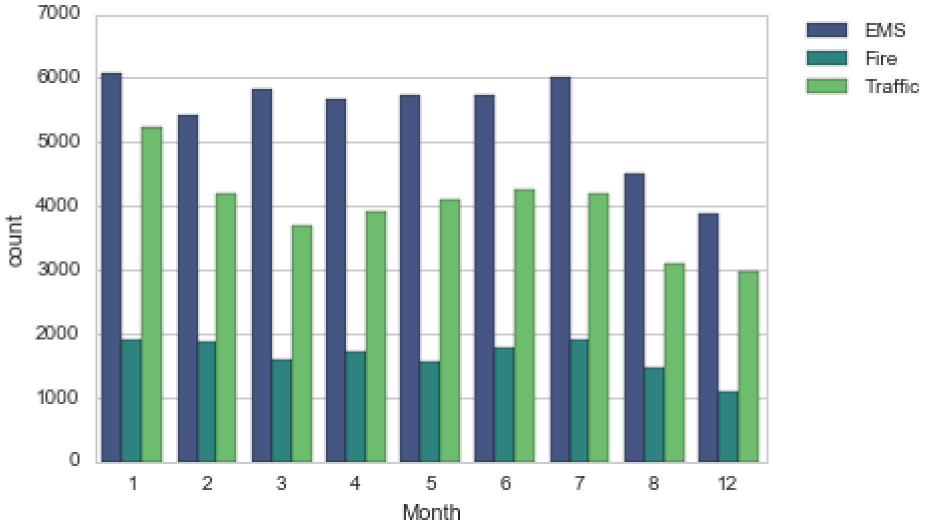


** Now do the same for Month:**

In [41]:
```
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[41]:
```
<matplotlib.legend.Legend at 0x11fa7ad68>
```



** Did you notice something strange about the Plot? **

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

In [43]:
```
byMonth = df.groupby('Month').count()
byMonth.head()
```

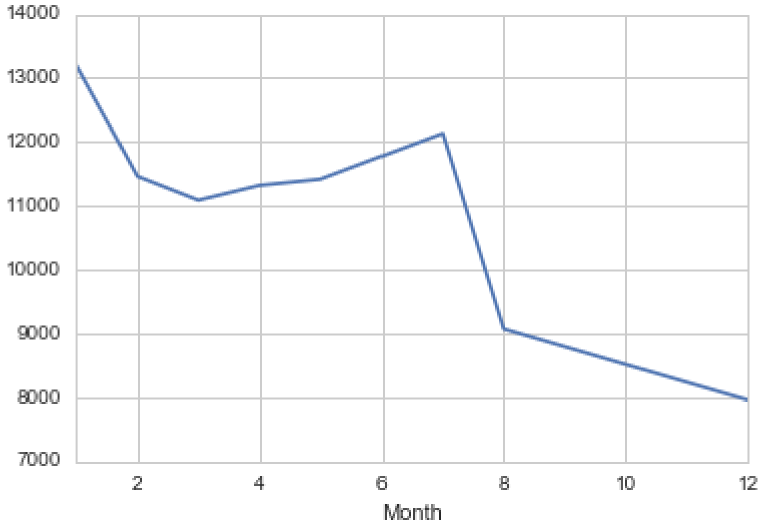
Out[43]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Da Wee
Month												
1	13205	13205	13205	11527	13205	13205	13203	13096	13205	13205	13205	1320
2	11467	11467	11467	9930	11467	11467	11465	11396	11467	11467	11467	1146
3	11101	11101	11101	9755	11101	11101	11092	11059	11101	11101	11101	1110
4	11326	11326	11326	9895	11326	11326	11323	11283	11326	11326	11326	1132
5	11423	11423	11423	9946	11423	11423	11420	11378	11423	11423	11423	1142

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

In [44]:
```
# Could be any column
byMonth['twp'].plot()
```

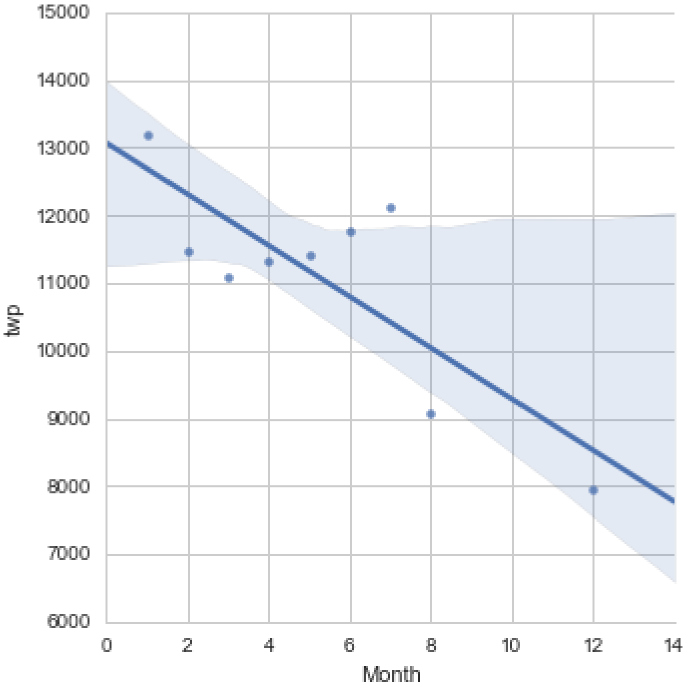
Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11fa06630>
```



** Now see if you can use seaborn's lmplot() to create a linear fit on the number of calls per month. Keep in mind you may need to reset the index to a column. **

In [45]:
```
sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())
```

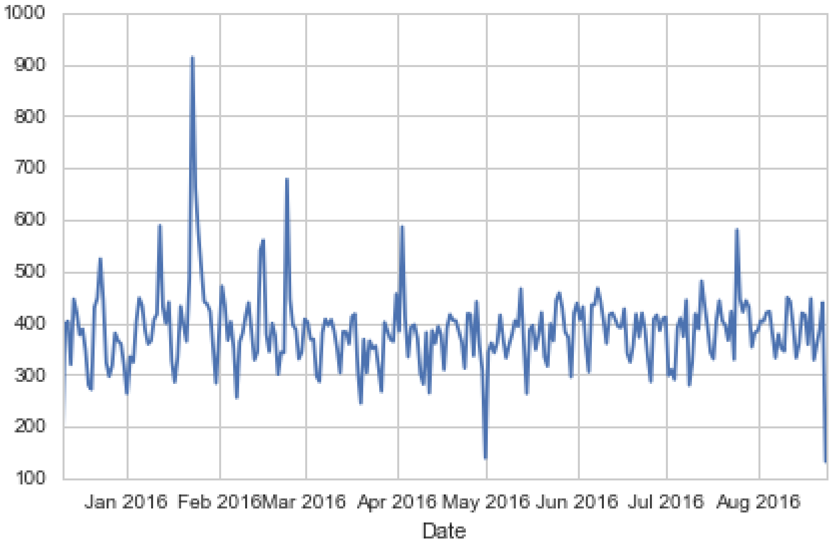
Out[45]:
```
<seaborn.axisgrid.FacetGrid at 0x11bf002b0>
```



In [46]:
```
df['Date']=df['timeStamp'].apply(lambda t: t.date())
```

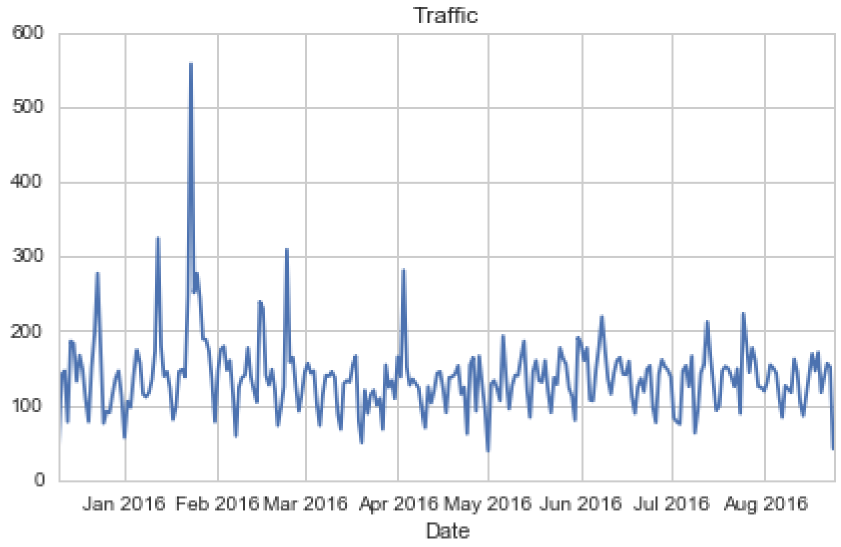
** Now groupby this Date column with the count() aggregate and create a plot of counts of 911 calls.**

In [47]:
```
df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
```

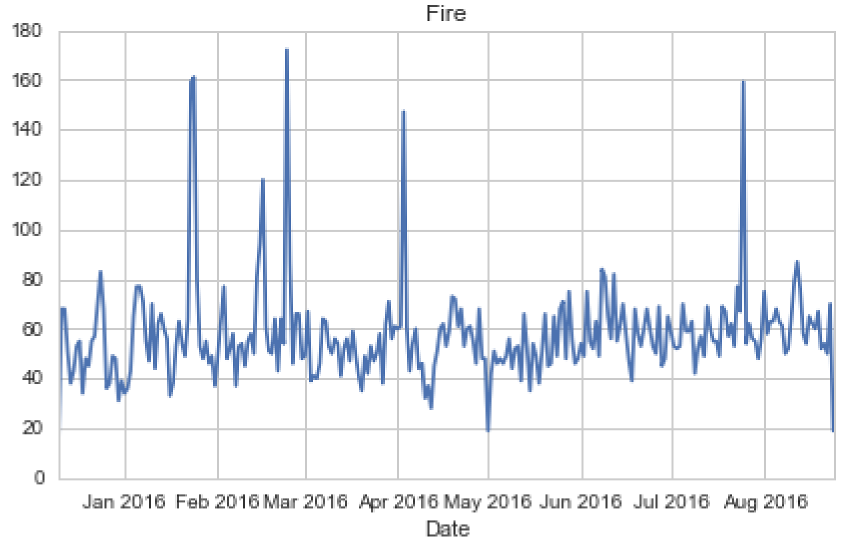


** Now recreate this plot but create 3 separate plots with each plot representing a Reason for the 911 call**

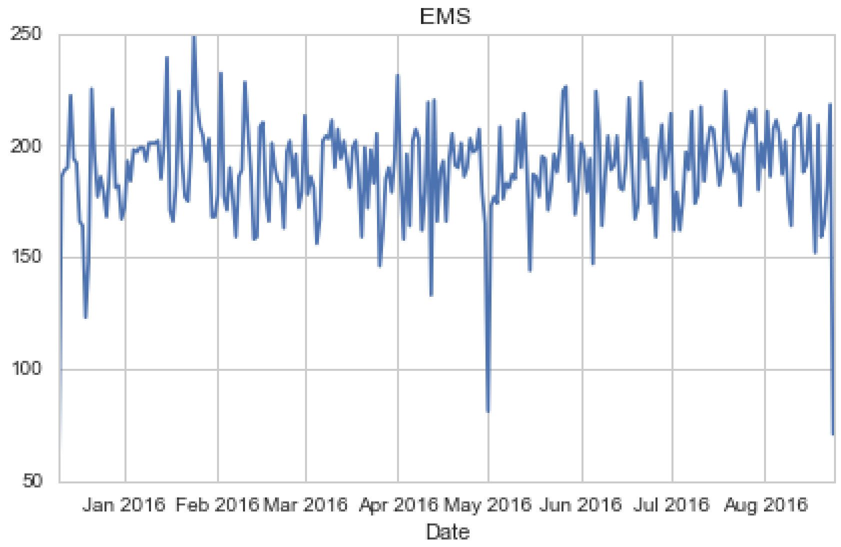
In [48]:
```
df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()
plt.title('Traffic')
plt.tight_layout()
```



In [49]:
```
df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
plt.title('Fire')
plt.tight_layout()
```



In [50]:
```
df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()
plt.title('EMS')
plt.tight_layout()
```



In [51]:
```
dayHour = df.groupby(by=['Day of Week','Hour']).count()['Reason'].unstack()
dayHour.head()
```

Out[51]:

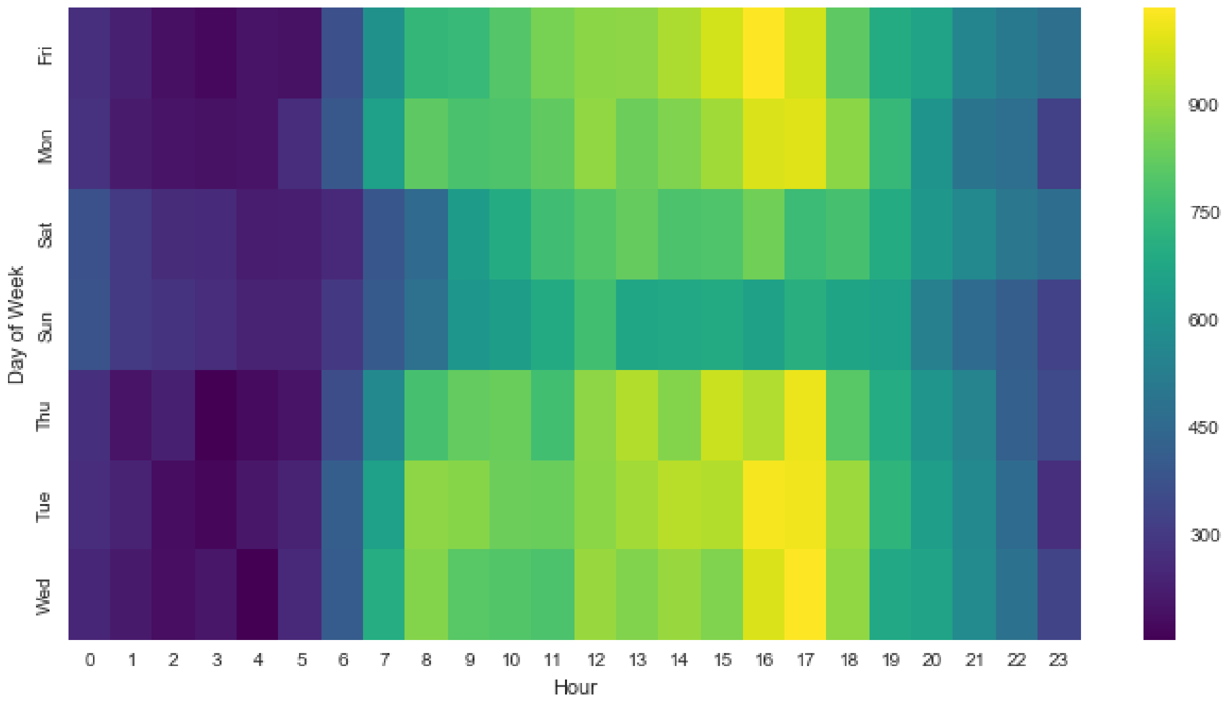
Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	2
Day of Week																		
Fri	275	235	191	175	201	194	372	598	742	752	...	932	980	1039	980	820	696	66
Mon	282	221	201	194	204	267	397	653	819	786	...	869	913	989	997	885	746	6
Sat	375	301	263	260	224	231	257	391	459	640	...	789	796	848	757	778	696	62
Sun	383	306	286	268	242	240	300	402	483	620	...	684	691	663	714	670	655	53
Thu	278	202	233	159	182	203	362	570	777	828	...	876	969	935	1013	810	698	6

5 rows × 24 columns

** Now create a HeatMap using this new DataFrame. **

In [52]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayHour,cmap='viridis')
```

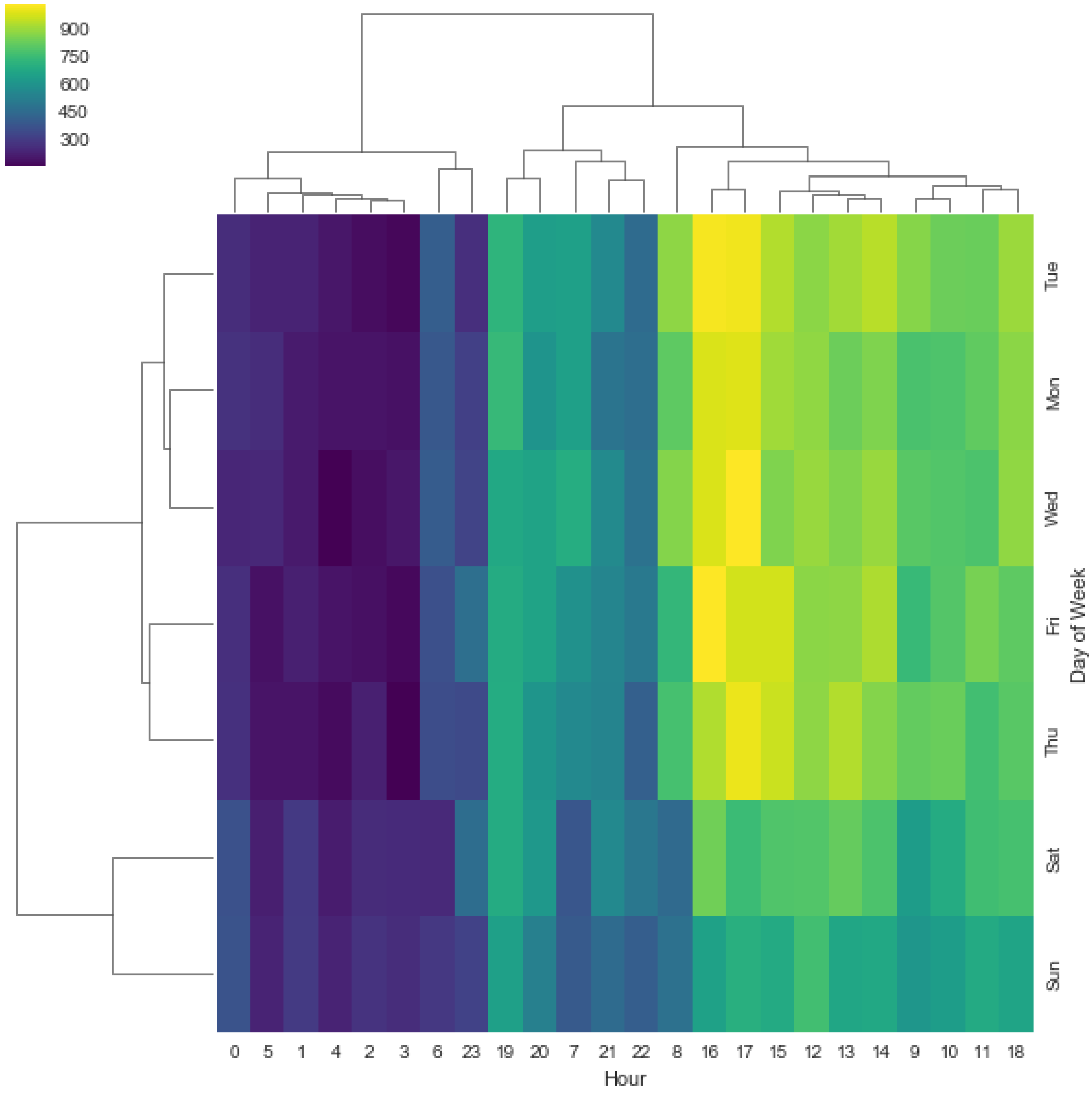
Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12305acf8>
```



** Now create a clustermap using this DataFrame. **

```
In [53]: sns.clustermap(dayHour,cmap='viridis')
```

```
Out[53]: <seaborn.matrix.ClusterGrid at 0x103276748>
```



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

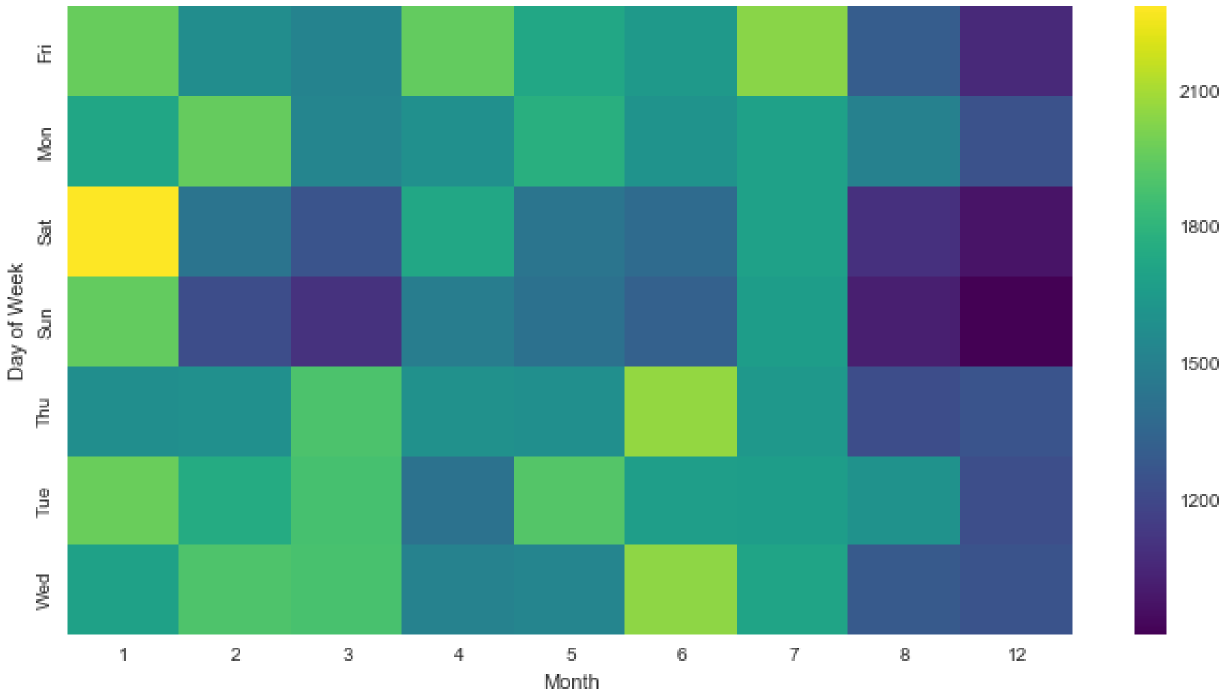
In [54]:
```
dayMonth = df.groupby(by=['Day of Week','Month']).count()['Reason'].unstack
dayMonth.head()
```

Out[54]:

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

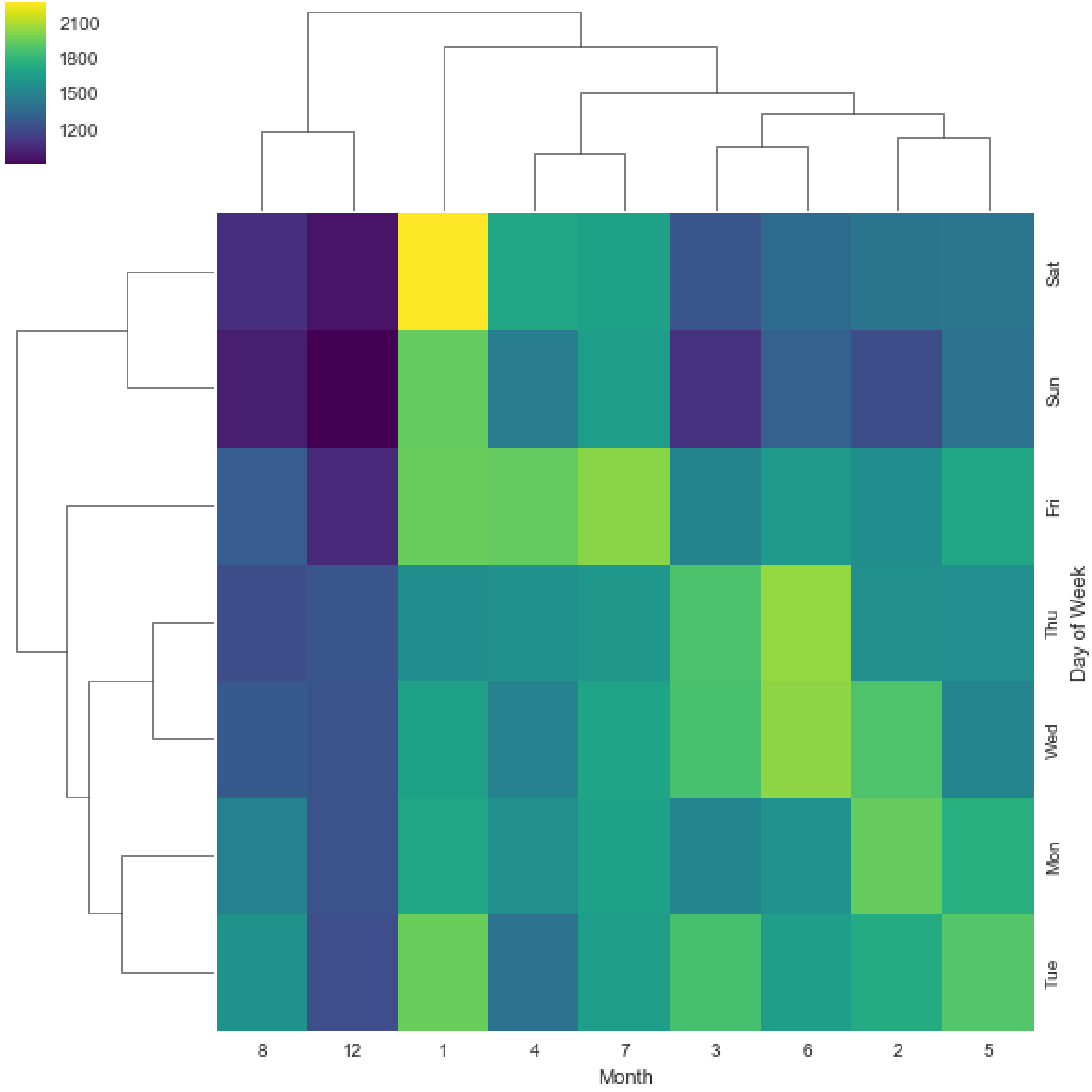
In [55]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayMonth,cmap='viridis')
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11bcabf98>
```



```
In [56]: sns.clustermap(dayMonth,cmap='viridis')
```

```
Out[56]: <seaborn.matrix.ClusterGrid at 0x120341e80>
```


Sat
Sun
Fri
Thu
Wed
Mon
Tue
Day of Week
Month