911 Calls Capstone Project

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)

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 pandas as pd
import numpy as np
```

Import visualization libraries and set %matplotlib inline.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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()
```

Out[6]:

	lat	lng	desc	zip	title	timeStamp	twp

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	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	RE &
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:40:00	HATFIELD TOWNSHIP	BR W
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	
3	40.116153	-75.343513	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;...	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 17:40:01	NORRISTOWN	
4	40.251492	-75.603350	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S...	NaN	EMS: DIZZINESS	2015-12-10 17:40:01	LOWER POTTSGROVE	CHI

Basic Questions

What are the top 5 zipcodes for 911 calls?

In [8]:
```
df['zip'].value_counts().head()
```

Out[8]:
```
19401.0    6979
19464.0    6643
19403.0    4854
19446.0    4748
19406.0    3174
Name: zip, dtype: int64
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

What are the top 5 townships (twp) for 911 calls?

In [10]:
```
df['twp'].value_counts().head()
```

Out[10]:
```
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 [11]:
```
df['title'].nunique()
```

Out[11]:
```
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 [12]:
```
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 [13]:
```
df['Reason'].value_counts()
```

Out[13]:
```
EMS        48877
```

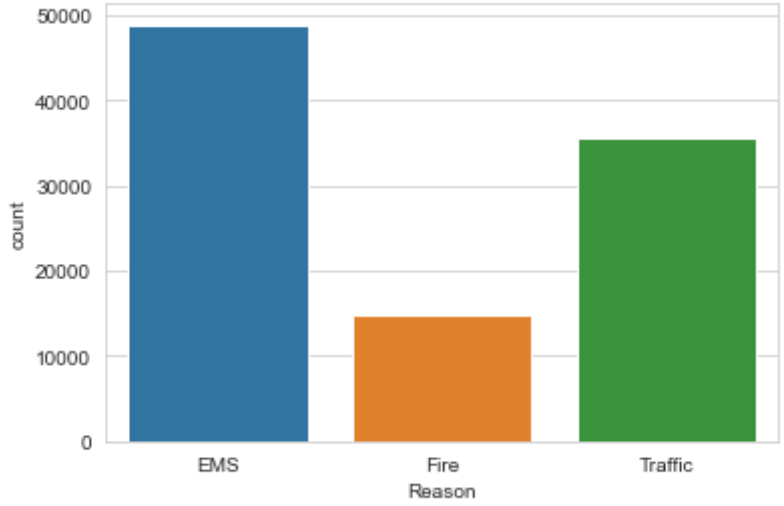
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

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

In [17]:
```
sns.countplot(x='Reason',data=df)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1c5ff370>
```



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

In [23]:
```
type(df['timeStamp'].loc[0])
```

Out[23]:
```
str
```

You should have seen that these timestamps are still strings. Use pd.to_datetime to

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

convert the column from strings to DateTime objects.

In [25]:
```
df['timeStamp']=pd.to_datetime(df['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, use .apply() to create 3 new columns called Hour, Month, and Day of Week. You will create these columns based off of the timeStamp column, reference the solutions if you get stuck on this step.

In [28]:
```
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 [30]:
```
dmap={0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
```

In [31]:
```
df['Day of Week'].map(dmap)
```

Out[31]:
```
0        Thu
1        Thu
2        Thu
3        Thu
4        Thu
       ...
```

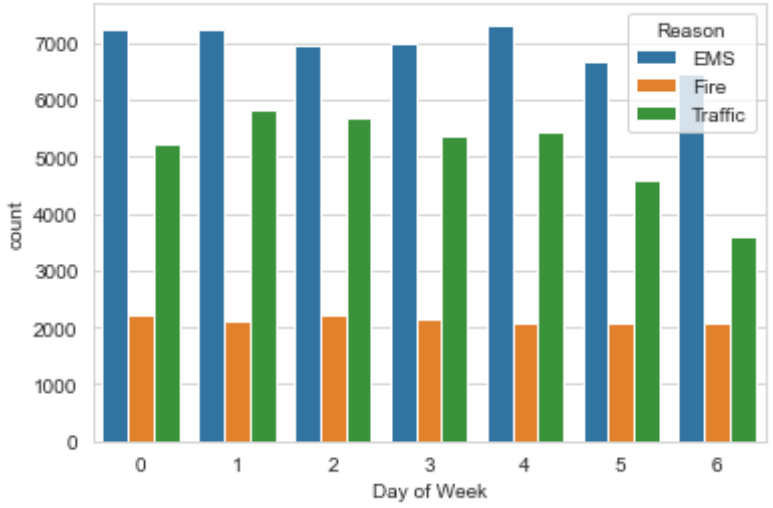
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
99487    Wed
99488    Wed
99489    Wed
99490    Wed
99491    Wed
Name: Day of Week, Length: 99492, dtype: object
```

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

In [32]:
```
sns.countplot(x=df['Day of Week'],data=df,hue=df['Reason'])
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1bddef10>
```

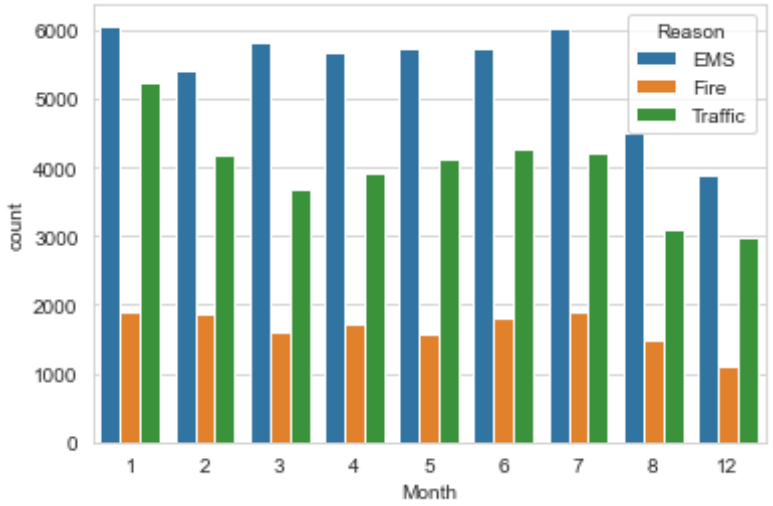


Now do the same for Month:

In [35]:
```
sns.countplot(x=df['Month'],data=df,hue=df['Reason'])
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1d1c2220>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Did you notice something strange about the Plot?

You should have noticed it was missing some Months, let's see if we can maybe fill in this information by plotting the information in another way, possibly a simple line plot that fills in the missing months, in order to do this, we'll need to do some work with pandas...

Now create a gropuby object called byMonth, where you group the DataFrame by the month column and use the count() method for aggregation. Use the head() method on this returned DataFrame.

In [36]:
```
bymonth=df.groupby('Month').count()
bymonth.head()
```

Out[36]:

Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	W
1	13205	13205	13205	11527	13205	13205	13203	13096	13205	13205	13205	13

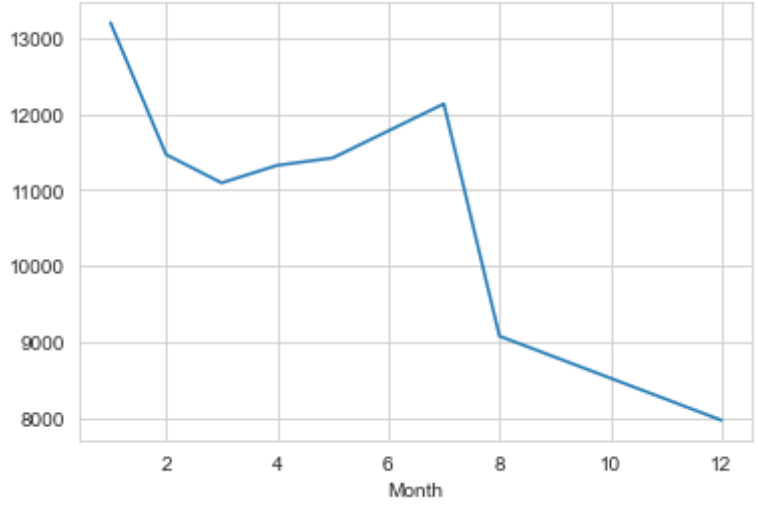
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	W
2	11467	11467	11467	9930	11467	11467	11465	11396	11467	11467	11467	11
3	11101	11101	11101	9755	11101	11101	11092	11059	11101	11101	11101	11
4	11326	11326	11326	9895	11326	11326	11323	11283	11326	11326	11326	11
5	11423	11423	11423	9946	11423	11423	11420	11378	11423	11423	11423	11

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

In [37]:
```
bymonth['twp'].plot()
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1cf68a90>
```

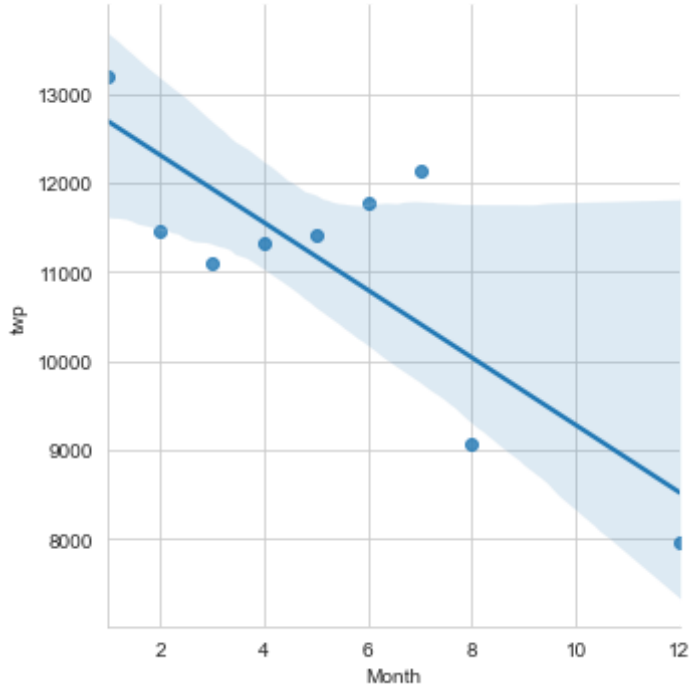


Now see if you can use seaborn's lmplot() to create a linear fit on the number of calls per month. Keep in mind you may need to reset the index to a column.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [38]:
```
sns.lmplot(x='Month',y='twp',data=bymonth.reset_index())
```

Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x1bd1cf97fa0>
```



Create a new column called 'Date' that contains the date from the timeStamp column. You'll need to use apply along with the .date() method.

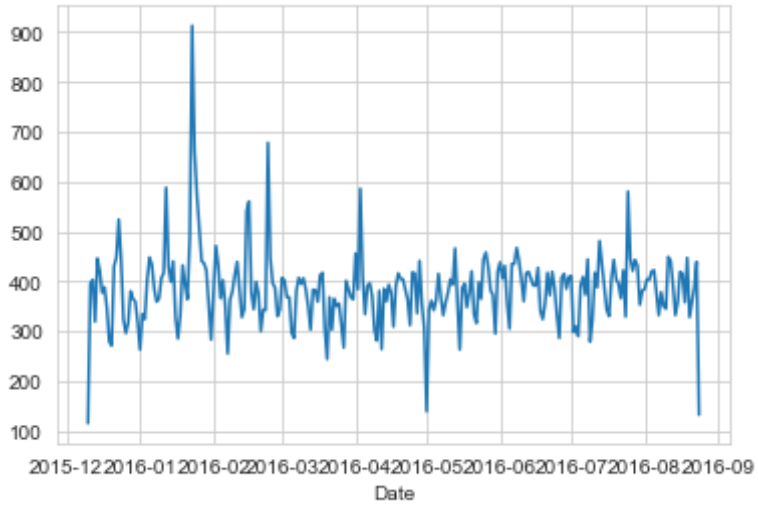
In [40]:
```
df['Date']=df['timeStamp'].apply(lambda time:time.date())
```

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

In [45]:
```
bydate=df.groupby('Date').count()
bydate['twp'].plot()
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1d0cd2e0>
```

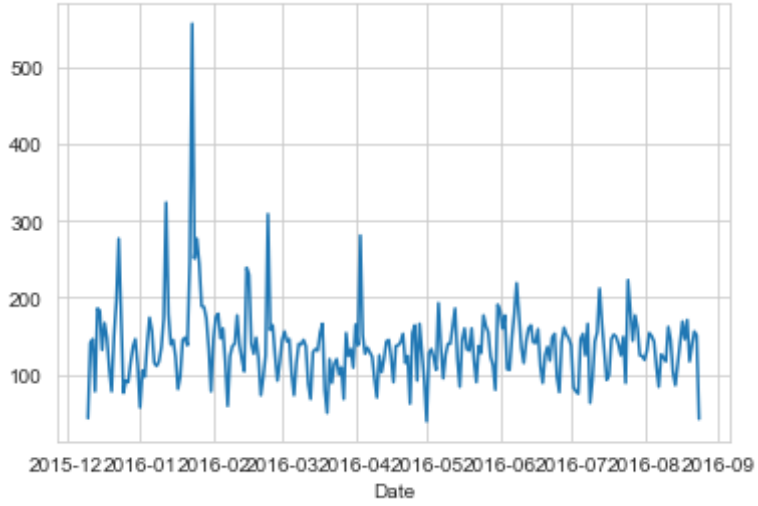
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



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

In [46]:
```
df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()
```

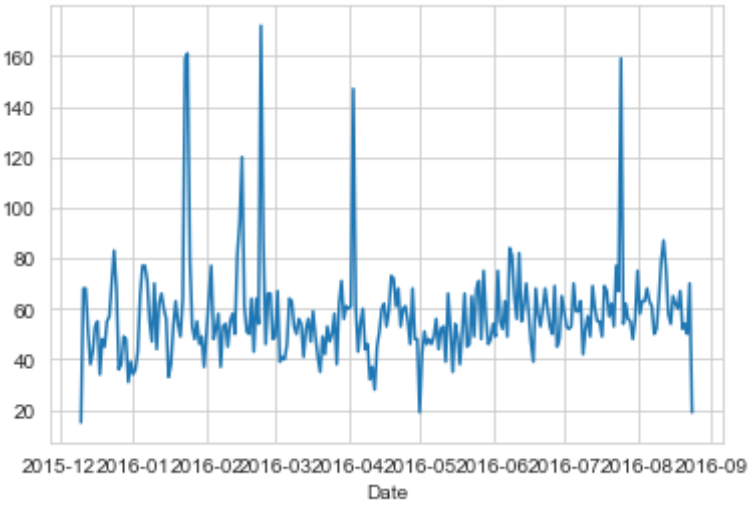
Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1d117130>
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [48]:
```
df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
```

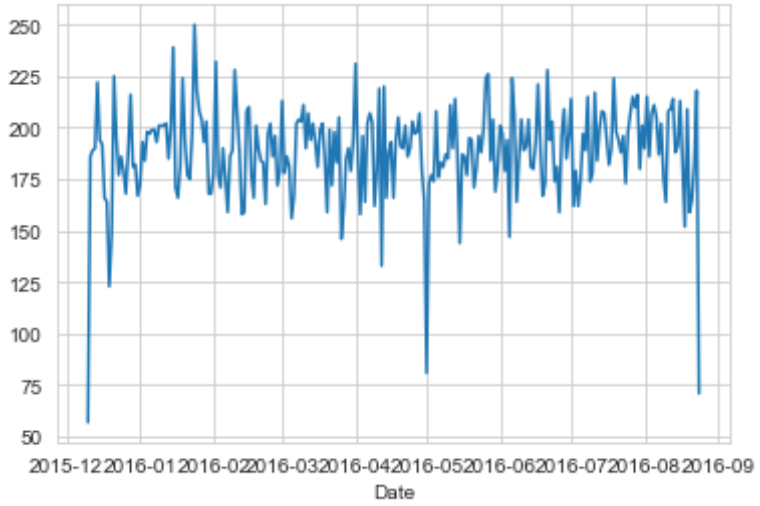
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1bd1d204040>



In [51]:
```
df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x1bd1d277af0>

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



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. There are lots of ways to do this, but I would recommend trying to combine groupby with an unstack method. Reference the solutions if you get stuck on this!

In [53]:
```
dayHour=df.groupby(by=['Day of Week','Hour']).count()['Reason'].unstack()
dayHour.head()
```

Out[53]:

Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19
Day of Week																	
0	282	221	201	194	204	267	397	653	819	786	...	869	913	989	997	885	746
1	269	240	186	170	209	239	415	655	889	880	...	943	938	1026	1019	905	731

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19
Day of Week																	
2	250	216	189	209	156	255	410	701	875	808	...	904	867	990	1037	894	686
3	278	202	233	159	182	203	362	570	777	828	...	876	969	935	1013	810	698
4	275	235	191	175	201	194	372	598	742	752	...	932	980	1039	980	820	696

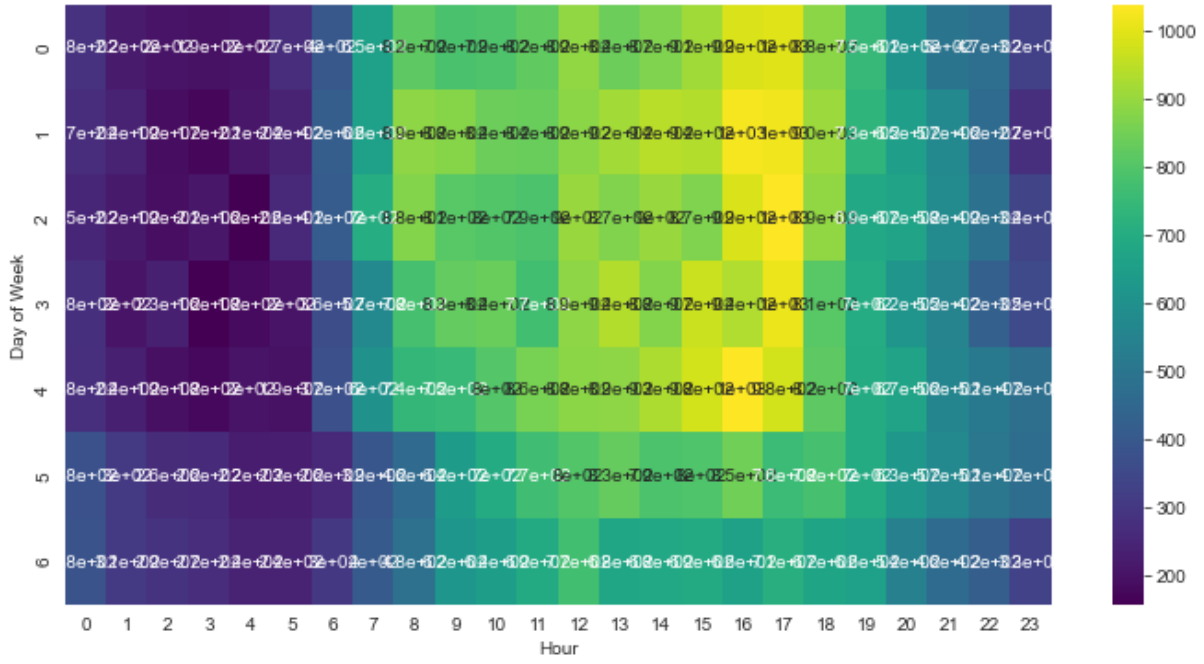
5 rows × 24 columns

Now create a HeatMap using this new DataFrame.

In [56]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayHour,annot=True,cmap='viridis')
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1da181c0>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

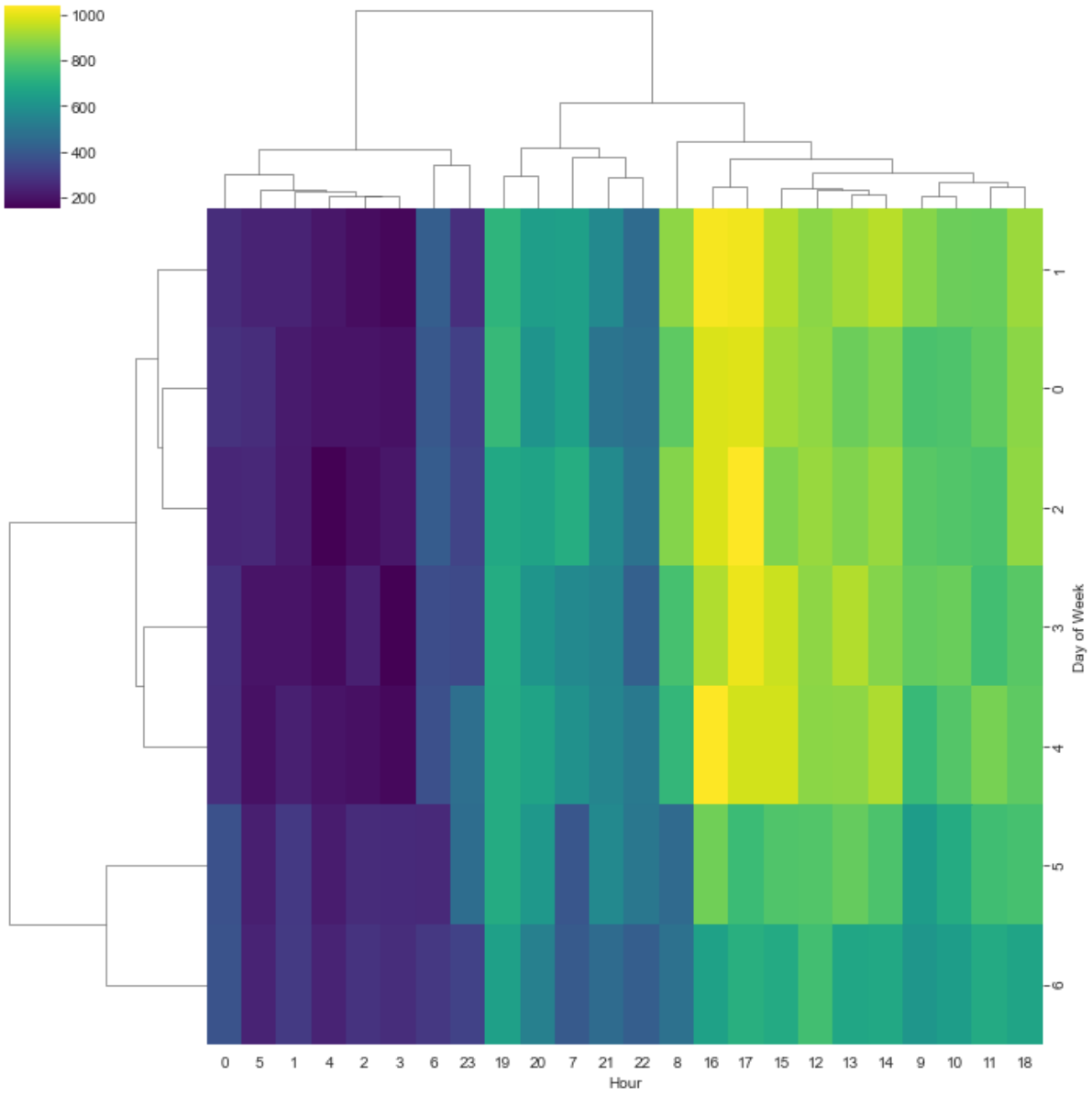


Now create a clustermap using this DataFrame.

In [57]:
```
sns.clustermap(dayHour,cmap='viridis',)
```

Out[57]:
```
<seaborn.matrix.ClusterGrid at 0x1bd1deb0ca0>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

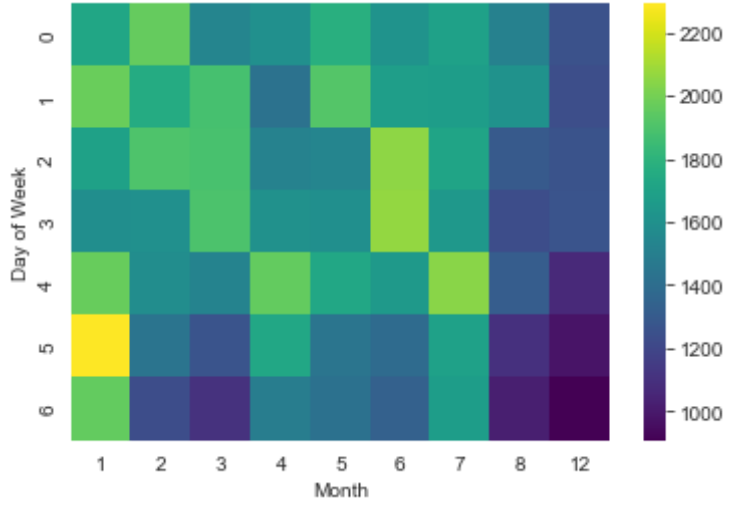
In [58]:
```
dayMonth=df.groupby(by=['Day of Week','Month']).count()['Reason'].unstack()
dayMonth.head()
```

Out[58]:

Month	1	2	3	4	5	6	7	8	12
Day of Week									
0	1727	1964	1535	1598	1779	1617	1692	1511	1257
1	1973	1753	1884	1430	1918	1676	1670	1612	1234
2	1700	1903	1889	1517	1538	2058	1717	1295	1262
3	1584	1596	1900	1601	1590	2065	1646	1230	1266
4	1970	1581	1525	1958	1730	1649	2045	1310	1065

In [59]:
```
sns.heatmap(dayMonth,cmap='viridis')
```

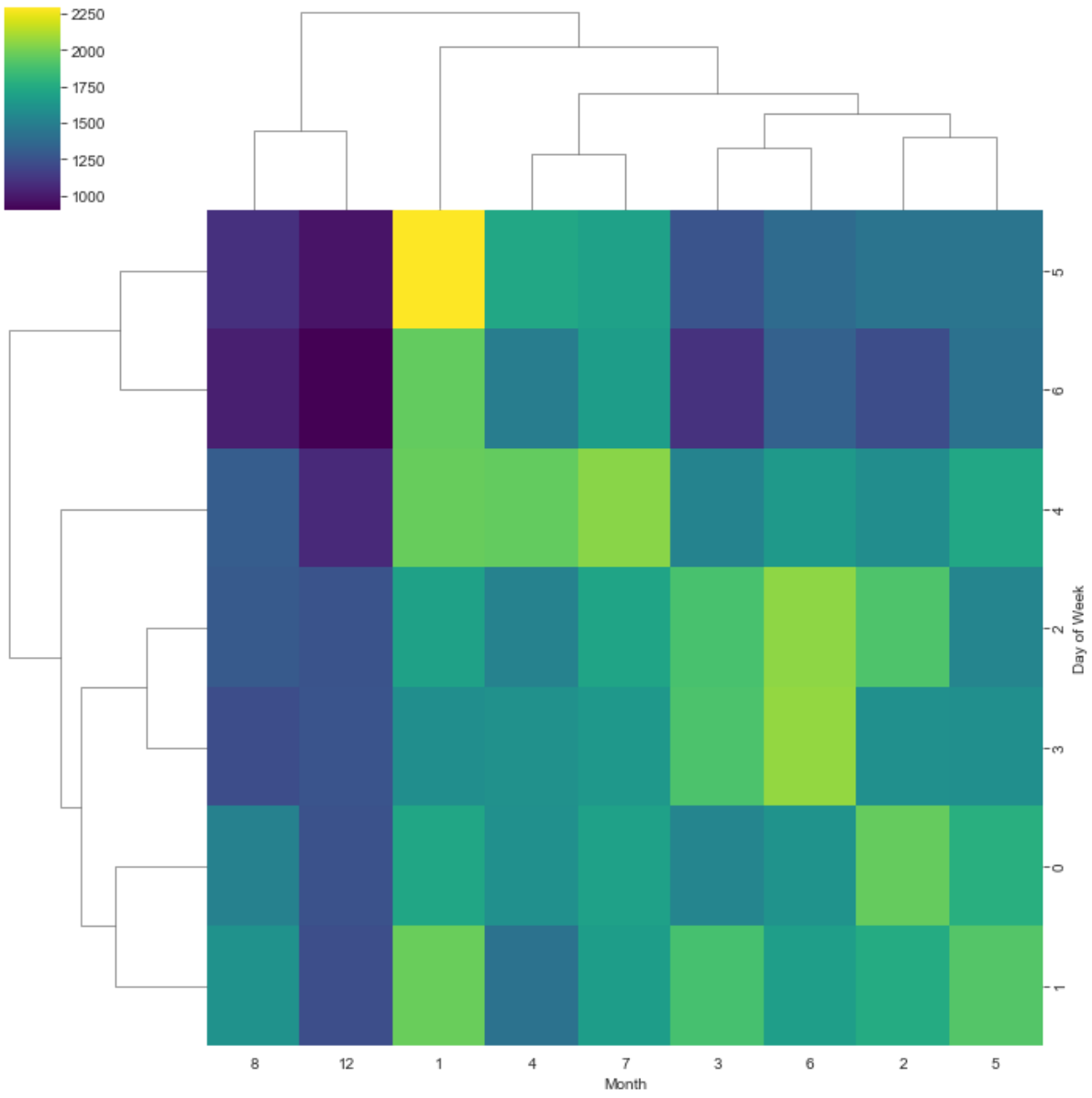
Out[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd1db700a0>
```



In [60]:
```
sns.clustermap(dayMonth,cmap='viridis')
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Out[60]: <seaborn.matrix.ClusterGrid at 0x1bd1db18340>



Continue exploring the Data however you see fit!

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Great Job!

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD