DATA VISUALISATION & ANALYSIS PROJECT- 911 CALL

Analysing some 911 call data from Kaggle using python libraries like numpy, pandas, seaborn, matplotlib and scikit-learn. 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

- Importing numpy, pandas, visualisation libraries.

```
[2]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     sns.set_style('whitegrid')
     %matplotlib inline
```

- Reading csv file

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

- Information of the file

```
[12]: 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
```

- Sample dataset

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

[6]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e
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 CT & DEAD END	1
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 PATH & WHITEMARSH LN	1
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 AVE	1
3	40.116153	-75.343513	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;...	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 17:40:01	NORRISTOWN	AIRY ST & SWEDE ST	1
4	40.251492	-75.603350	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S...	NaN	EMS: DIZZINESS	2015-12-10 17:40:01	LOWER POTTSGROVE	CHERRYWOOD CT & DEAD END	1

Exploratory Data Analysis

- Top 5 zipcodes for 911 calls

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

```
[8]:

zip
19401.0    6979
19464.0    6643
19403.0    4854
19446.0    4748
19406.0    3174
Name: count, dtype: int64
```

- Top 5 townships for 911 calls

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

```
[10]:

twp
LOWER MERION    8443
ABINGTON        5977
NORRISTOWN      5890
UPPER MERION    5227
CHELTENHAM      4575
Name: count, dtype: int64
```

- Number of unique title codes

```
[12]:
df['title'].nunique()
```

```
[12]:
110
```

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

```
[14]:
df['Reason']=df['title'].apply(lambda x: x.split(':')[0])
```

- Most common Reason for 911 calls

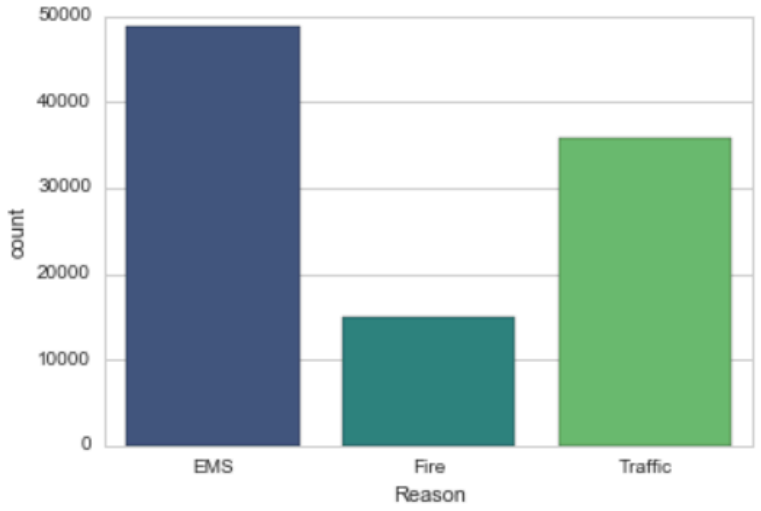
```
[16]:
df['Reason'].value_counts().head()
```

```
[16]:
Reason
EMS        48877
Traffic    35695
Fire       14920
Name: count, dtype: int64
```

- Countplot for 911 calls by Reason

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

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



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

  Creating these columns based off of the timeStamp column.

  [20]:

```
type(df['timeStamp'].iloc[0])
df['timeStamp']=pd.to_datetime(df['timeStamp'])
df['Hour']=df['timeStamp'].apply(lambda x:x.hour)
df['Month']=df['timeStamp'].apply(lambda x:x.month)
df['DayOfWeek']=df['timeStamp'].apply(lambda x:x.dayofweek)
```

  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.

  [22]:

```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
df['DayOfWeek']=df['DayOfWeek'].map(dmap)
```

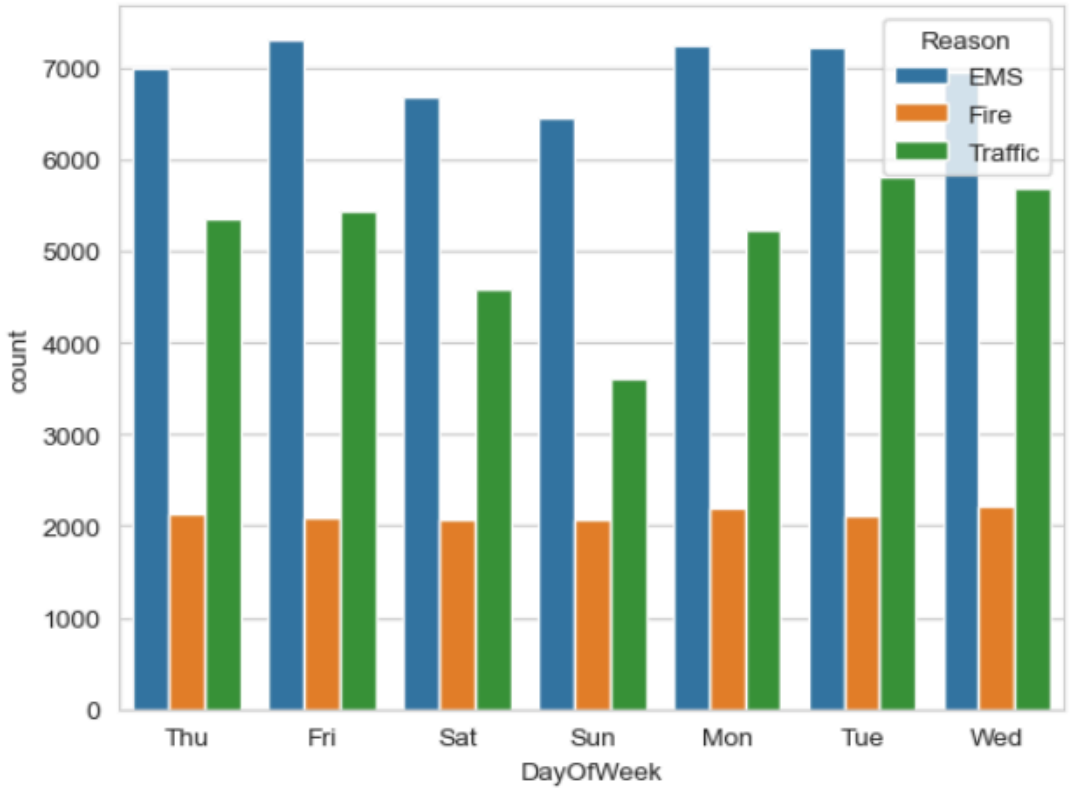
- Creating a countplot of the Day of Week column with the hue based off of the Reason column.

[24]:

```
sns.countplot(x=df['DayOfWeek'],data=df,hue=df['Reason'])
```

[24]:

```
<Axes: xlabel='DayOfWeek', ylabel='count'>
```



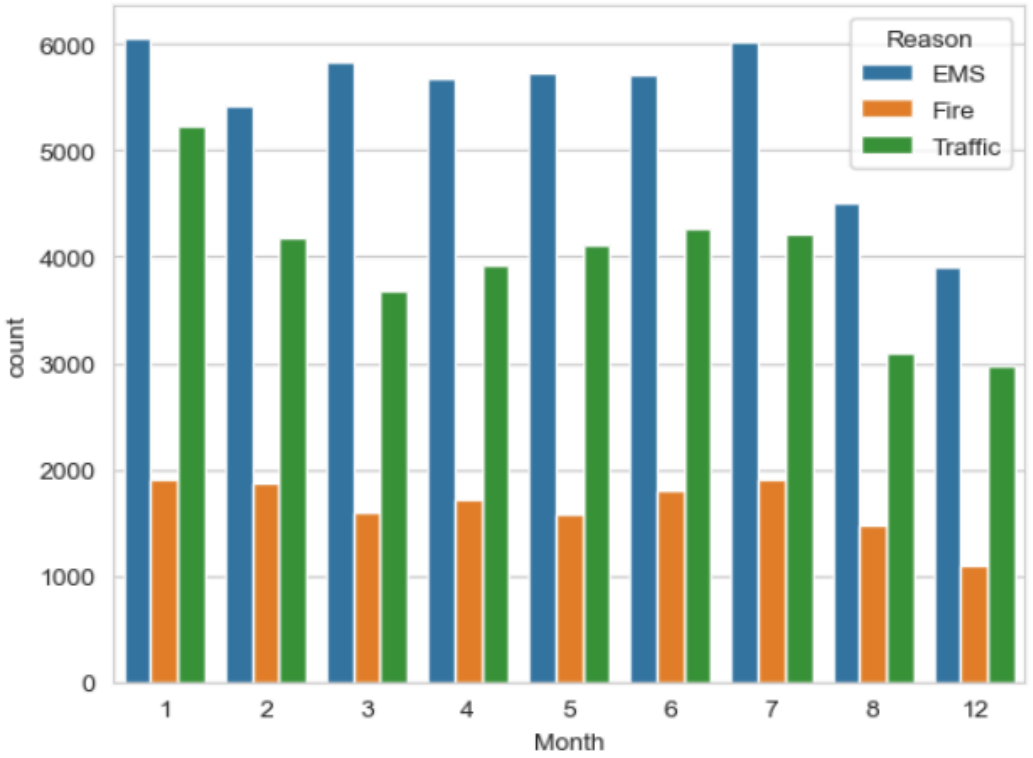
- Creating a countplot of the Month column with the hue based off of the Reason column.

[26]:

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

[26]:

```
<Axes: xlabel='Month', ylabel='count'>
```



- The above countplot is missing some Months, to fill in this information by plotting the information in another way.
  Creating a gropuby object called byMonth, where the DataFrame is grouped by the month column and using the count() method for aggregation. Using the head() method on this returned DataFrame.

[28]:

```
byMonth=df.groupby('Month').count()
byMonth.head()
```

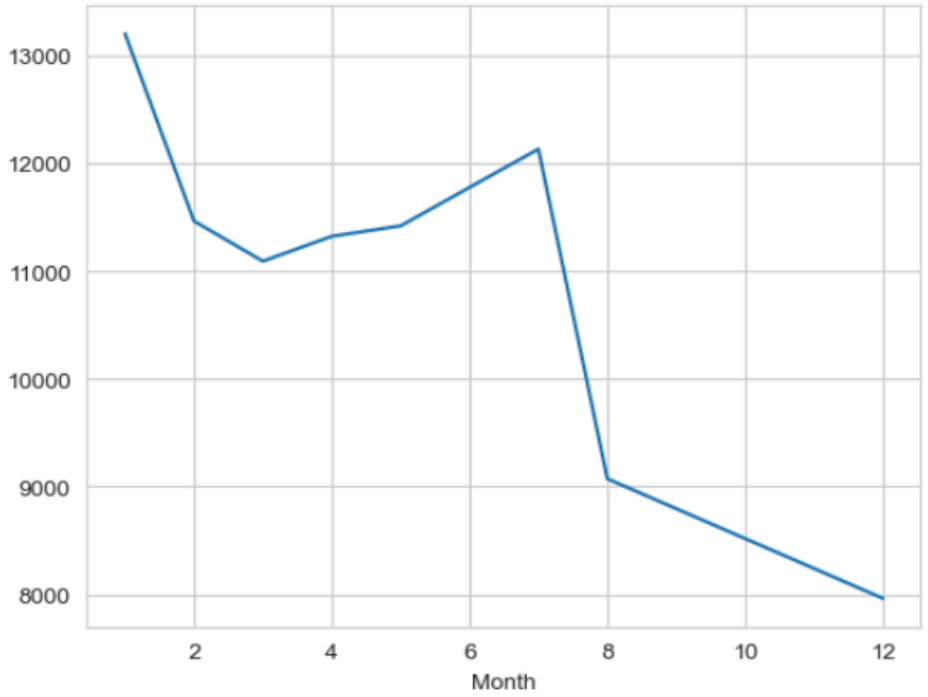
[28]:

Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	DayOfWeek
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

A simple plot off of the dataframe indicating the count of calls per month.

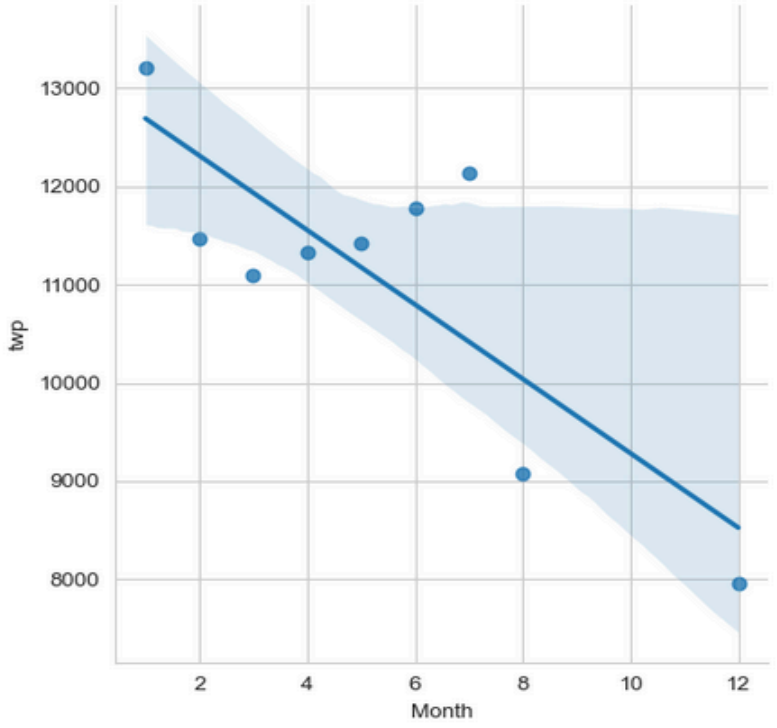
```
[30]:
byMonth['twp'].plot()
```

```
[30]:
<Axes: xlabel='Month'>
```



- Creating a linear fit on the number of calls per month.

```
[32]: sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())
[32]: <seaborn.axisgrid.FacetGrid at 0x1d1d684b2d0>
```



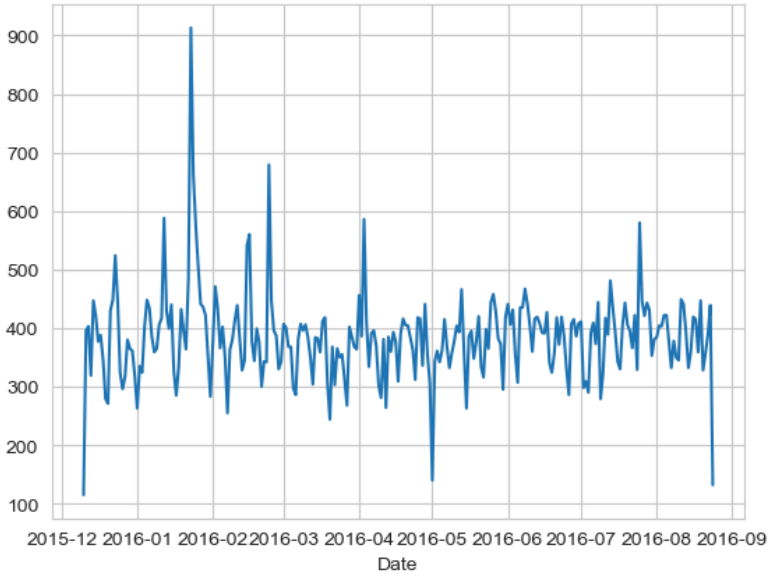
- Creating a new column called 'Date' that contains the date from the timeStamp column.

```
[34]:
df['Date']=df['timeStamp'].apply(lambda x:x.date())
```

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

```
[36]: df.groupby('Date').count()['lat'].plot()
```

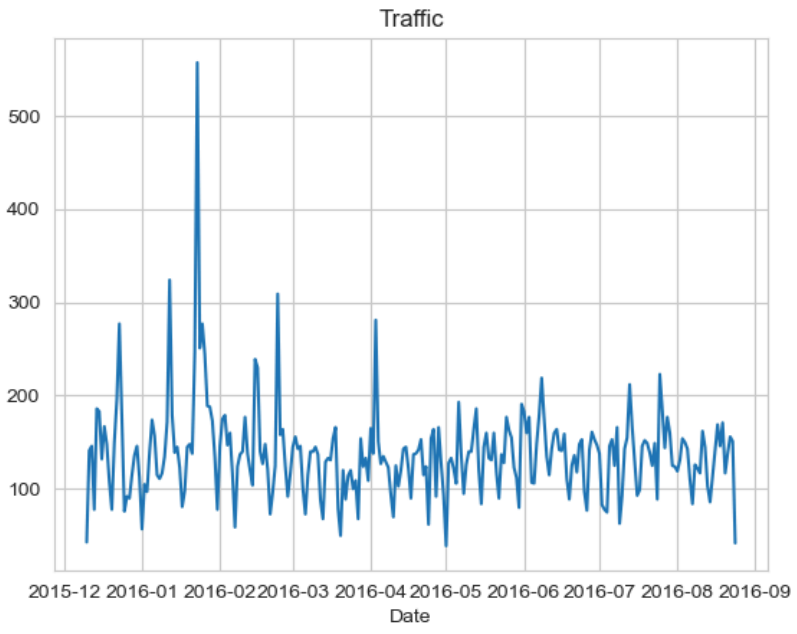
```
[36]: <Axes: xlabel='Date'>
```



- Recreating this plot but creating 3 separate plots with each plot representing a Reason for the 911 call.

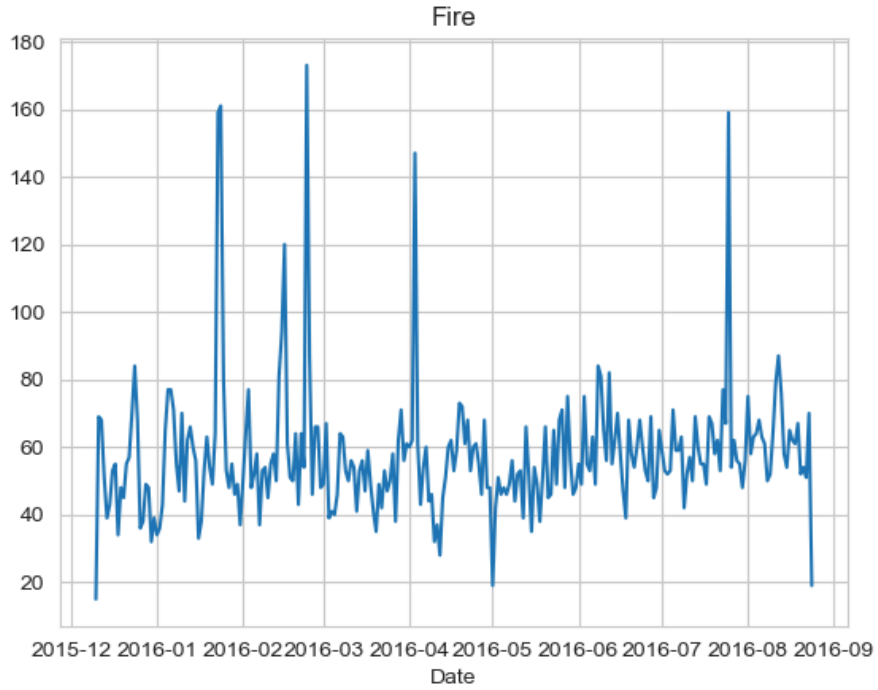
```
[38]: df[df['Reason']=='Traffic'].groupby('Date').count()['lat'].plot()
      plt.title('Traffic')
```

```
[38]: Text(0.5, 1.0, 'Traffic')
```



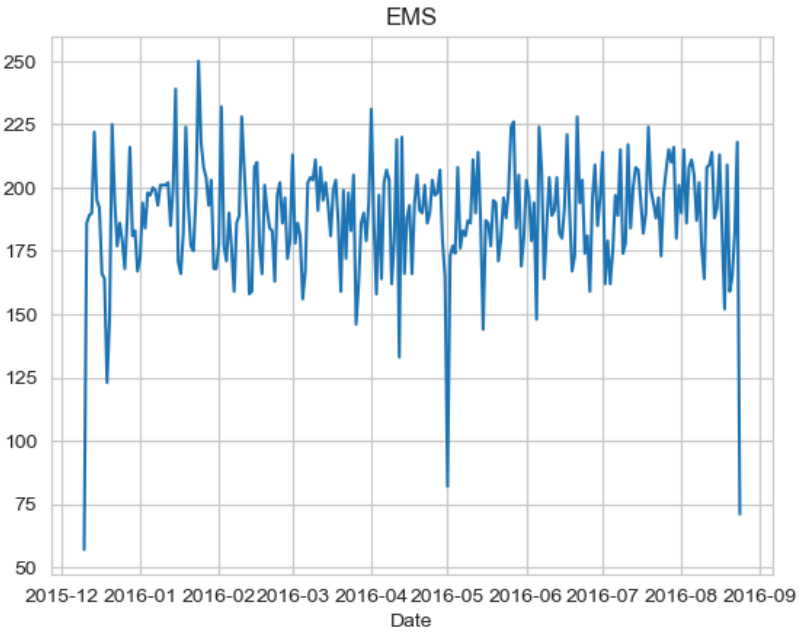
[40]:
```
df[df['Reason']=='Fire'].groupby('Date').count()['lat'].plot()
plt.title('Fire')
```

[40]:
```
Text(0.5, 1.0, 'Fire')
```



[42]:
```
df[df['Reason']=='EMS'].groupby('Date').count()['lat'].plot()
plt.title('EMS')
```

[42]:
```
Text(0.5, 1.0, 'EMS')
```



- Creating heatmaps with seaborn and our data.
  First I need to restructure the data frame so that the columns become the Hours and the Index becomes the Day of the Week.

```
[44]: df.groupby(by=['DayOfWeek','Hour']).count()['Reason'].unstack()
```

[44]:

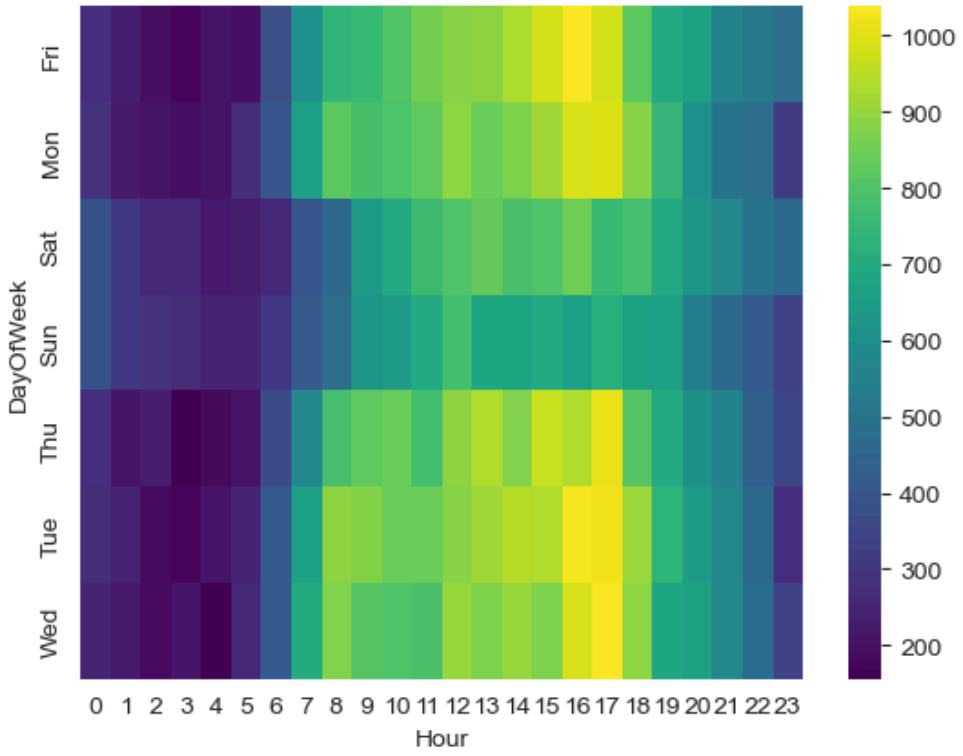
Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
DayOfWeek																					
Fri	275	235	191	175	201	194	372	598	742	752	...	932	980	1039	980	820	696	667	559	514	474
Mon	282	221	201	194	204	267	397	653	819	786	...	869	913	989	997	885	746	613	497	472	325
Sat	375	301	263	260	224	231	257	391	459	640	...	789	796	848	757	778	696	628	572	506	467
Sun	383	306	286	268	242	240	300	402	483	620	...	684	691	663	714	670	655	537	461	415	330
Thu	278	202	233	159	182	203	362	570	777	828	...	876	969	935	1013	810	698	617	553	424	354
Tue	269	240	186	170	209	239	415	655	889	880	...	943	938	1026	1019	905	731	647	571	462	274
Wed	250	216	189	209	156	255	410	701	875	808	...	904	867	990	1037	894	686	668	575	490	335

7 rows × 24 columns

- Creating a HeatMap using this new DataFrame.

```
[46]: dayHour=df.groupby(by=['DayOfWeek','Hour']).count()['Reason'].unstack()
      sns.heatmap(dayHour,cmap='viridis')
```

```
[46]: <Axes: xlabel='Hour', ylabel='DayOfWeek'>
```



- Creating a ClusterMap using this DataFrame.

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

```
[48]: <seaborn.matrix.ClusterGrid at 0x1d1d6e61f10>
```