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)

Target ¶

Countplot,Heatmap,Boxplot and Clustermap generation for visualization.

In [16]:

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

In [17]:

```
pwd
```

Out[17]:

```
'C:\\Users\\YOGESHWAR'
```

In [18]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

In [24]:

```
df = pd.read_csv(r"C:\Users\YOGESHWAR\Desktop\Data Science and Machine Learning\911.csv")
```

In [25]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 663522 entries, 0 to 663521
Data columns (total 9 columns):
lat          663522 non-null float64
lng          663522 non-null float64
desc         663522 non-null object
zip          583323 non-null float64
title        663522 non-null object
timeStamp    663522 non-null object
twp          663229 non-null object
addr         663522 non-null object
e            663522 non-null int64
dtypes: float64(3), int64(1), object(5)
memory usage: 45.6+ MB
```

In [26]:

```
df.head()
```

Out[26]:

	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 17:10:52	NEW HANOVER
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:29:21	HATFIELD TOWNSHIP
2	40.121182	-75.351975	HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...	19401.0	Fire: GAS-ODOR/LEAK	2015-12-10 14:39:21	NORRISTOWN
3	40.116153	-75.343513	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;...	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 16:47:36	NORRISTOWN
4	40.251492	-75.603350	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S...	NaN	EMS: DIZZINESS	2015-12-10 16:56:52	LOWER POTTSGROVE

In [27]:

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

Out[27]:

```
19401.0    45606
19464.0    43910
19403.0    34888
19446.0    32270
19406.0    22464
Name: zip, dtype: int64
```

In [28]:

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

Out[28]:

```
LOWER MERION    55490
ABINGTON        39947
NORRISTOWN      37633
UPPER MERION    36010
CHELTENHAM      30574
Name: twp, dtype: int64
```

In [31]:

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

Out[31]:

```
148
```

In [33]:

```
extract = lambda x: x.split(':')[0]

df = df.assign(Reason = df['title'].apply(extract))
```

In [34]:

```
df['Reason'].value_counts()
```

Out[34]:

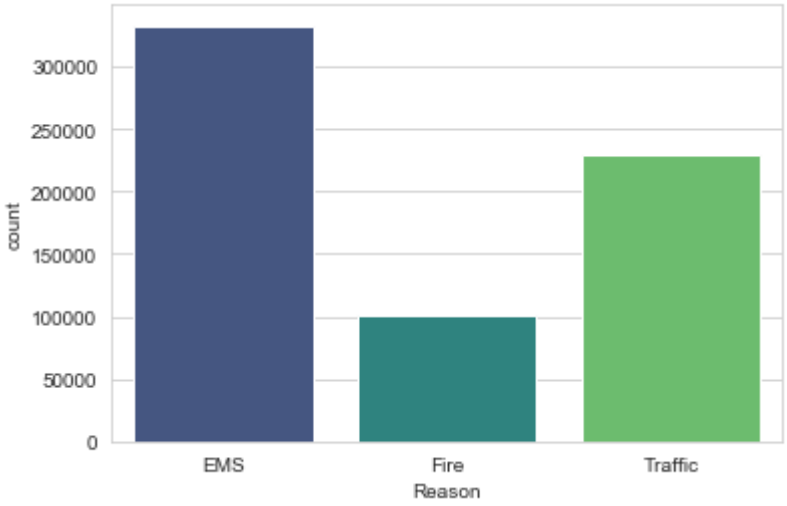
```
EMS        332692
Traffic    230208
Fire       100622
Name: Reason, dtype: int64
```

In [35]:

```
sns.countplot(x = 'Reason', data = df, palette='viridis')
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16317902438>
```



In [36]:

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

Out[36]:

```
str
```

In [37]:

```
df['timeStamp'] = pd.to_datetime(df['timeStamp'], format='%Y-%m-%d %H:%M:%S')
```

In [38]:

```
time = df['timeStamp'].iloc[0]
time.hour
```

Out[38]:

```
17
```

In [39]:

```
df['Hour'] = df['timeStamp'].apply(lambda x: x.hour)
df['Month'] = df['timeStamp'].apply(lambda x: x.month)
df['Day of Week'] = df['timeStamp'].apply(lambda x: x.dayofweek)

df.head()
```

Out[39]:

	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 17:10:52	NEW HANOVER
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:29:21	HATFIELD TOWNSHIP
2	40.121182	-75.351975	HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...	19401.0	Fire: GAS-ODOR/LEAK	2015-12-10 14:39:21	NORRISTOWN
3	40.116153	-75.343513	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;...	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 16:47:36	NORRISTOWN
4	40.251492	-75.603350	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S...	NaN	EMS: DIZZINESS	2015-12-10 16:56:52	LOWER POTTSGROVE

In [40]:

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

In [41]:

```
df['Day of Week'] = df['Day of Week'].apply(lambda x: dmap[x])

df.head()
```

Out[41]:

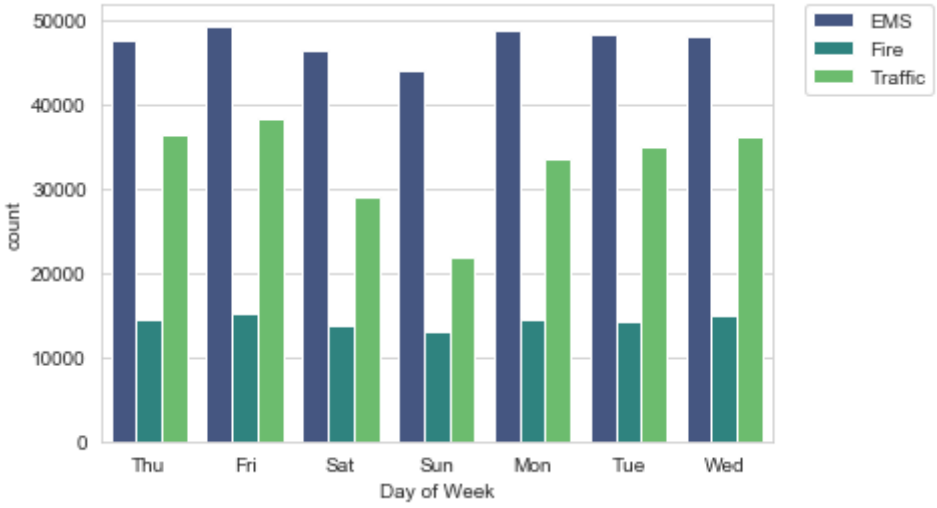
	lat	lng	desc	zip	title	timeStamp	twp
0	40.297876	-75.581294	REINDEER CT & DEAD END; NEW HANOVER; Station ...	19525.0	EMS: BACK PAINS/INJURY	2015-12-10 17:10:52	NEW HANOVER
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:29:21	HATFIELD TOWNSHIP
2	40.121182	-75.351975	HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St...	19401.0	Fire: GAS-ODOR/LEAK	2015-12-10 14:39:21	NORRISTOWN
3	40.116153	-75.343513	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;...	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 16:47:36	NORRISTOWN
4	40.251492	-75.603350	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S...	NaN	EMS: DIZZINESS	2015-12-10 16:56:52	LOWER POTTSGROVE

In [42]:

```
sns.countplot(x = 'Day of Week', data = df, palette='viridis', hue = 'Reason')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[42]:

```
<matplotlib.legend.Legend at 0x1631cf46358>
```

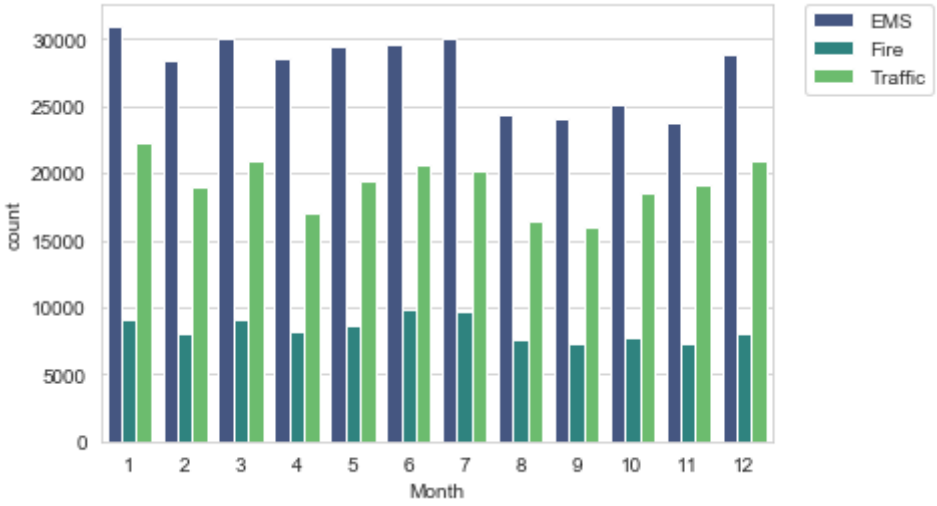


In [43]:

```
sns.countplot(x = 'Month', data = df, palette='viridis', hue = 'Reason')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[43]:

```
<matplotlib.legend.Legend at 0x1631cf0cac8>
```



In [44]:

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

byMonth.head()
```

Out[44]:

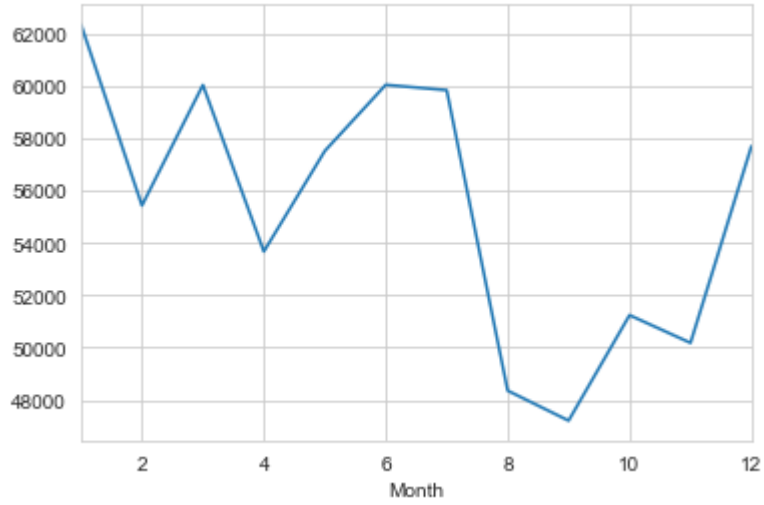
Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hou
1	62336	62336	62336	55294	62336	62336	62312	62336	62336	62336	62336
2	55427	55427	55427	48922	55427	55427	55405	55427	55427	55427	55427
3	60027	60027	60027	53252	60027	60027	60001	60027	60027	60027	60027
4	53671	53671	53671	47349	53671	53671	53655	53671	53671	53671	53671
5	57509	57509	57509	50354	57509	57509	57474	57509	57509	57509	57509

In [45]:

```
byMonth['lat'].plot()
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1631d0d3fd0>
```



In [46]:

```
byMonth = byMonth.reset_index()
byMonth.head()
```

Out[46]:

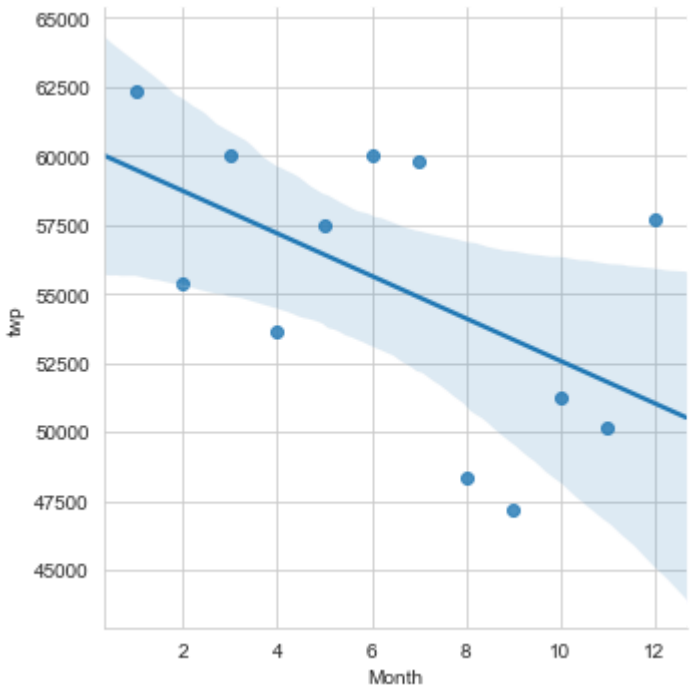
	Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	h
0	1	62336	62336	62336	55294	62336	62336	62312	62336	62336	62336	62
1	2	55427	55427	55427	48922	55427	55427	55405	55427	55427	55427	55
2	3	60027	60027	60027	53252	60027	60027	60001	60027	60027	60027	60
3	4	53671	53671	53671	47349	53671	53671	53655	53671	53671	53671	53
4	5	57509	57509	57509	50354	57509	57509	57474	57509	57509	57509	57

In [47]:

```
sns.lmplot(x = 'Month', y = 'twp', data = byMonth)
```

Out[47]:

```
<seaborn.axisgrid.FacetGrid at 0x163188ec048>
```

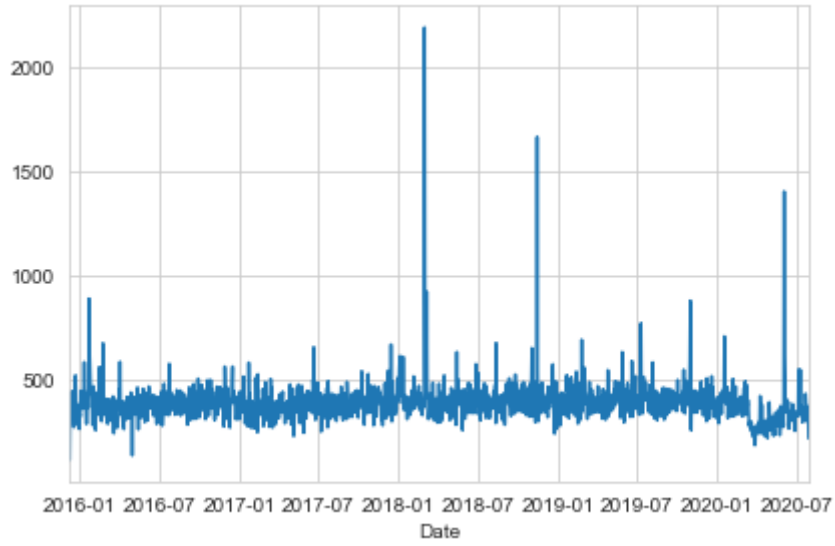


In [48]:

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

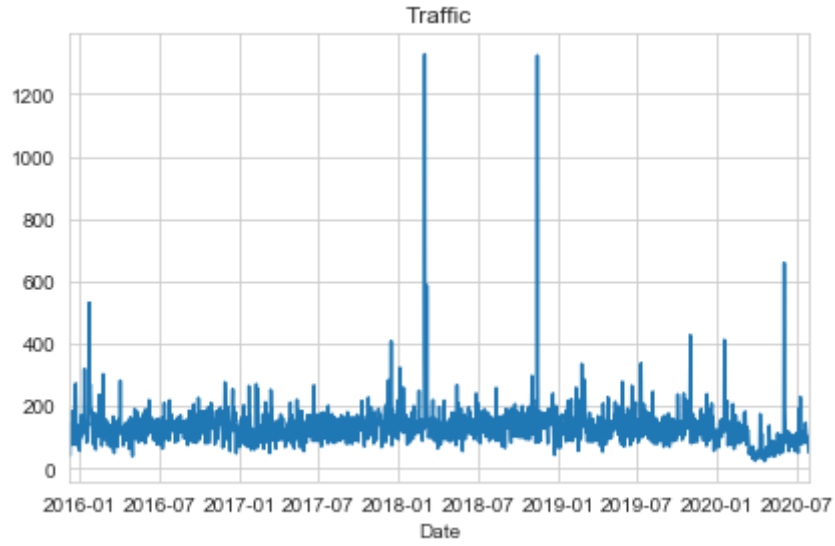
In [49]:

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



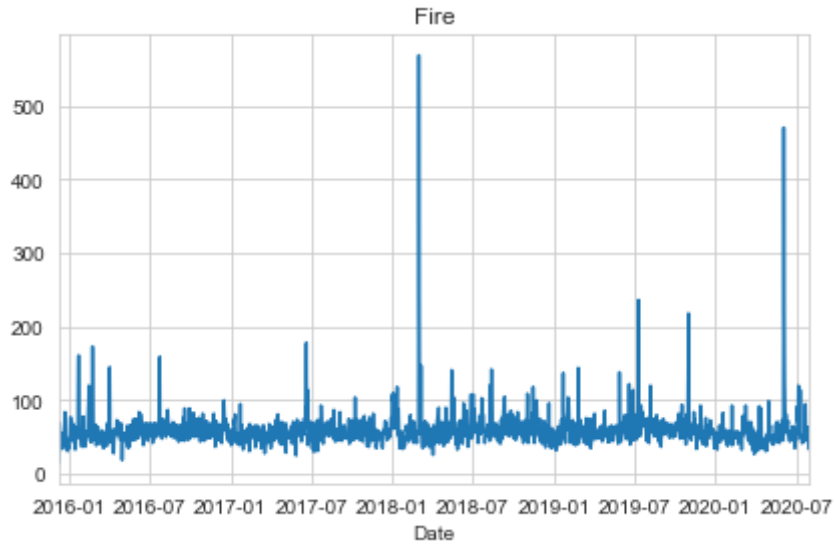
In [50]:

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



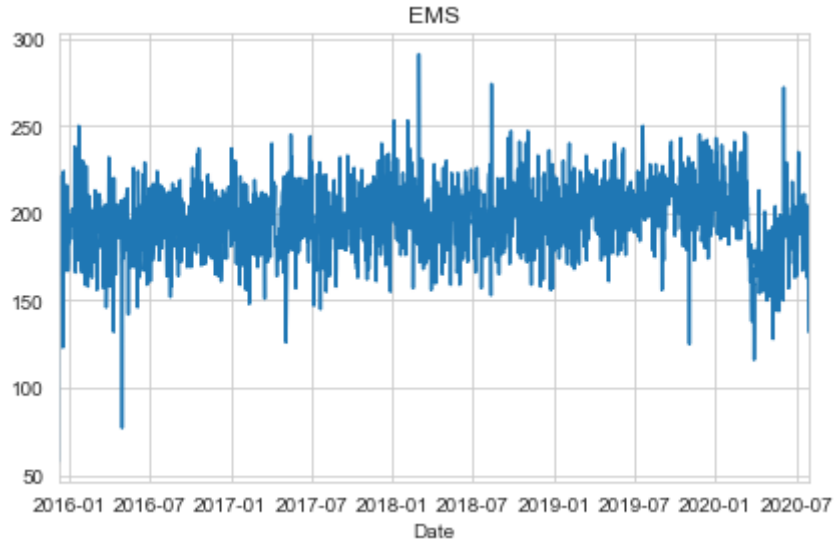
In [51]:

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



In [52]:

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



In [53]:

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

Out[53]:

Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16
Day of Week														
Fri	1983	1635	1449	1296	1339	1639	2670	4143	5018	5288	...	6394	7040	7065
Mon	1894	1571	1368	1272	1336	1844	2675	4430	5504	5724	...	5713	6289	6346
Sat	2447	2059	1883	1592	1451	1580	1880	2489	3457	4315	...	5421	5181	5211
Sun	2424	2135	1946	1614	1471	1488	1726	2408	3001	3728	...	4744	4475	4560
Thu	1731	1408	1426	1236	1293	1775	2816	4432	5297	5412	...	6079	6493	6375

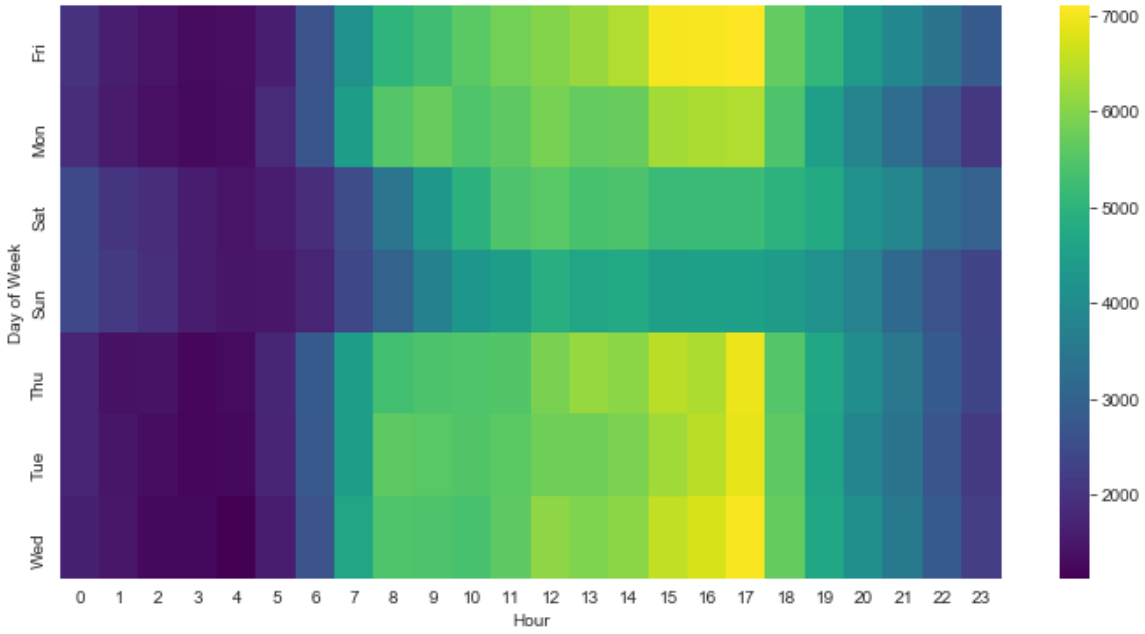
5 rows × 24 columns

In [54]:

```
plt.figure(figsize=(12,6))
sns.heatmap(dayHour,cmap='viridis')
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16318ae75f8>
```



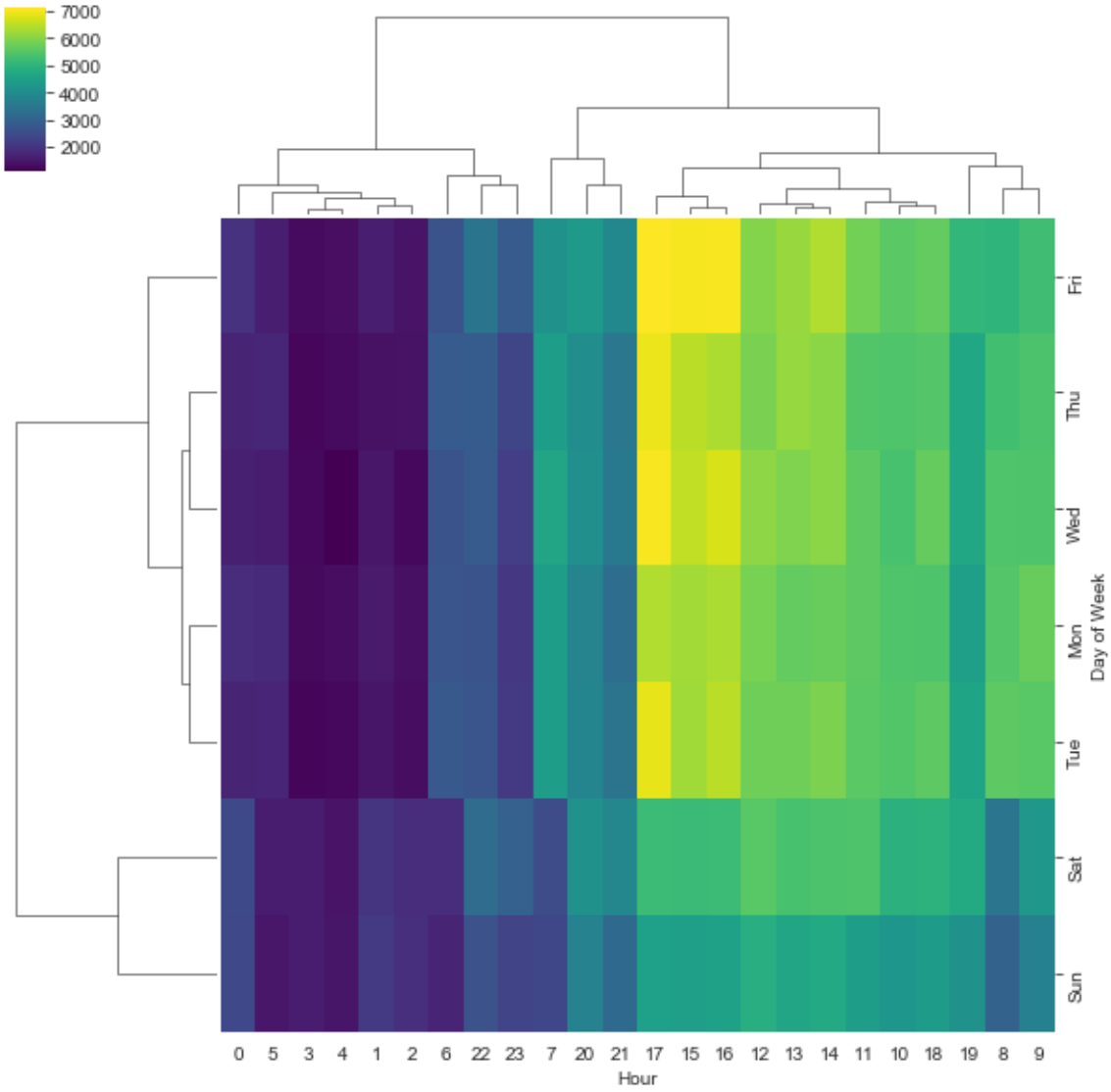
In [55]:

```
plt.figure(figsize=(12,6))
sns.clustermap(dayHour, cmap='viridis')
```

Out[55]:

```
<seaborn.matrix.ClusterGrid at 0x16318c12d68>

<Figure size 864x432 with 0 Axes>
```



In [56]:

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

Out[56]:

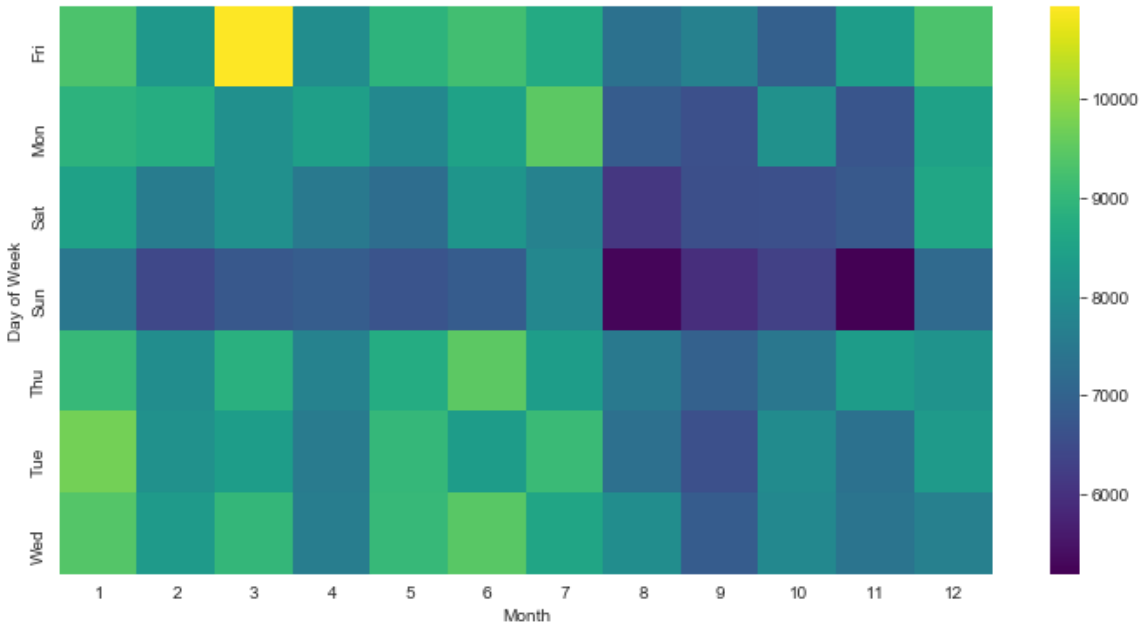
Month	1	2	3	4	5	6	7	8	9	10	11	12
Day of Week												
Fri	9309	8255	10941	7997	8904	9207	8681	7336	7694	6934	8379	9305
Mon	8896	8747	8060	8410	7881	8511	9499	6854	6598	8075	6722	8492
Sat	8475	7593	8050	7514	7223	8198	7748	6111	6566	6609	6773	8592
Sun	7478	6452	6766	6865	6694	6837	7859	5275	5956	6316	5196	7165
Thu	9055	7997	8849	7722	8740	9489	8378	7508	6954	7482	8358	8151

In [57]:

```
plt.figure(figsize=(12,6))
sns.heatmap(dayMonth,cmap='viridis')
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16318d1e2e8>
```



In [58]:

```
plt.figure(figsize=(12,6))
sns.clustermap(dayMonth, cmap='viridis')
```

Out[58]:

```
<seaborn.matrix.ClusterGrid at 0x1631d152dd8>

<Figure size 864x432 with 0 Axes>
```