911 Calls Capstone Project

Data and Setup

** Import numpy and pandas **

In [2]:
```
import pandas as pd
import seaborn as sns
```

In [3]:
```
%matplotlib inline
```

In [4]:
```
df=pd.read_csv('/Users/LuckyDog/Downloads/Bootcamp-master/10-Data-Capst
```

In [5]:
```
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
```

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

Out[6]:

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

** Top 5 zipcodes for 911 calls? **

In [7]:
```
1 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
```

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

In [8]:
```
1 df['twp'].value_counts().head(5)
```

Out[8]:
```
LOWER MERION    8443
ABINGTON        5977
NORRISTOWN      5890
UPPER MERION    5227
CHELTENHAM      4575
Name: twp, dtype: int64
```

** How many unique title codes are there? **

In [9]:
```
1 df['title'].value_counts()
```

Out[9]:
```
Traffic: VEHICLE ACCIDENT -        23066
Traffic: DISABLED VEHICLE -         7702
Fire: FIRE ALARM                    5496
EMS: RESPIRATORY EMERGENCY          5112
EMS: CARDIAC EMERGENCY              5012
                                   ...
Fire: UNKNOWN MEDICAL EMERGENCY        1
EMS: BOMB DEVICE FOUND                 1
EMS: PLANE CRASH                       1
Fire: UNCONSCIOUS SUBJECT              1
EMS: DISABLED VEHICLE                  1
Name: title, Length: 110, dtype: int64
```

Creating new features

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

In [11]:
```
1 df.head()
```

Out[11]:

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

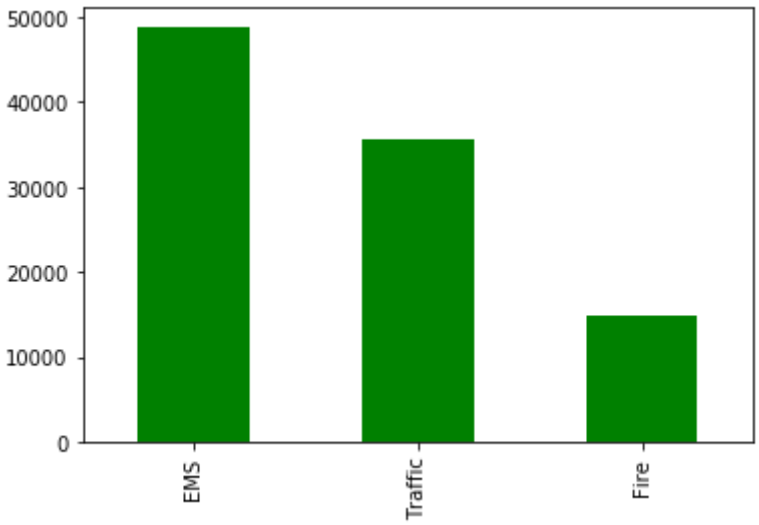
** What is the most common Reason for a 911 call based off of this new column? **

In [12]:
```
1 df['reason'].value_counts().head()
```

Out[12]:
```
EMS        48877
Traffic    35695
Fire       14920
Name: reason, dtype: int64
```

** Create a countplot of 911 calls by Reason. **

In [13]:
```
1 data=df['reason'].value_counts().plot(kind='bar', color='green')
```



*What is the data type of the objects in the timeStamp column? *

In [14]:
```
1 type('reason')
```

Out[14]:
```
str
```

In [109]:
```
1 df['timeStamp'] = pd.to_datetime(df['timeStamp'])
```

In [110]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99492 entries, 0 to 99491
Data columns (total 20 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 datetime64[ns]
twp            99449 non-null object
addr           98973 non-null object
e              99492 non-null int64
reason         99492 non-null object
hour           99492 non-null int64
month          99492 non-null int64
day            99492 non-null int64
year           99492 non-null int64
Day of Week    99492 non-null object
Hour           99492 non-null int64
Month          99492 non-null int64
Year           99492 non-null int64
Date           99492 non-null int64
date           99492 non-null object
dtypes: datetime64[ns](1), float64(3), int64(9), object(7)
memory usage: 15.2+ MB
```

Creating columns based off of the timeStamp column

In [111]:
```

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 [112]:
```
df.head()
```

Out[112]:

	desc	zip	title	timeStamp	twp	addr	e	reason	hour	ı
	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	EMS	17	
	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:40:00	HATFIELD TOWNSHIP	BRIAR PATH & WHITEMARSH LN	1	EMS	17	
	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	Fire	17	
	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;...	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 17:40:01	NORRISTOWN	AIRY ST & SWEDE ST	1	EMS	17	
	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S...	NaN	EMS: DIZZINESS	2015-12-10 17:40:01	LOWER POTTSGROVE	CHERRYWOOD CT & DEAD END	1	EMS	17	

Using the .map() with this dictionary to map the actual string names to the day of the wee

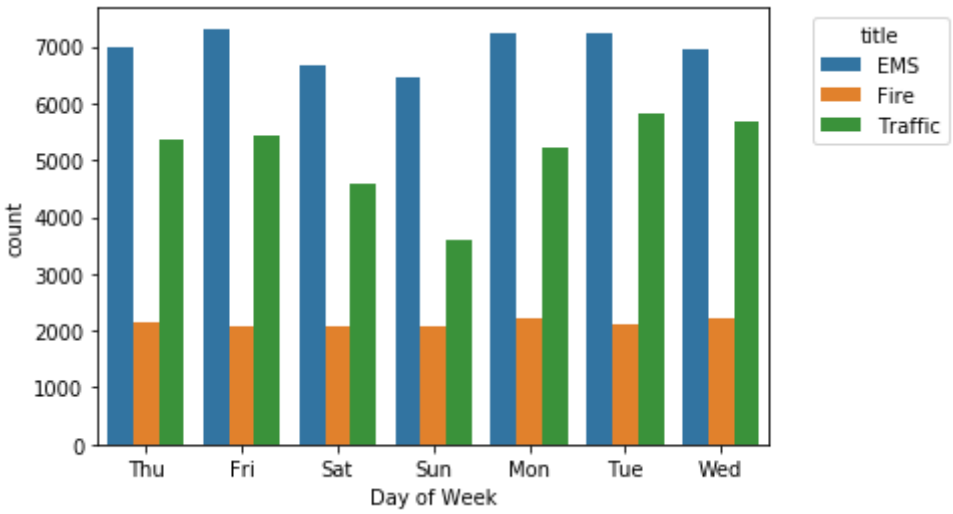
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
```

In [113]:
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
df['Day of Week'] = df['Day of Week'].map(dmap)
```

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

In [114]:
```
sns.countplot(x='Day of Week', data=df,hue='reason')
plt.legend(title='title', bbox_to_anchor=(1.05, 1), loc='upper left')

```

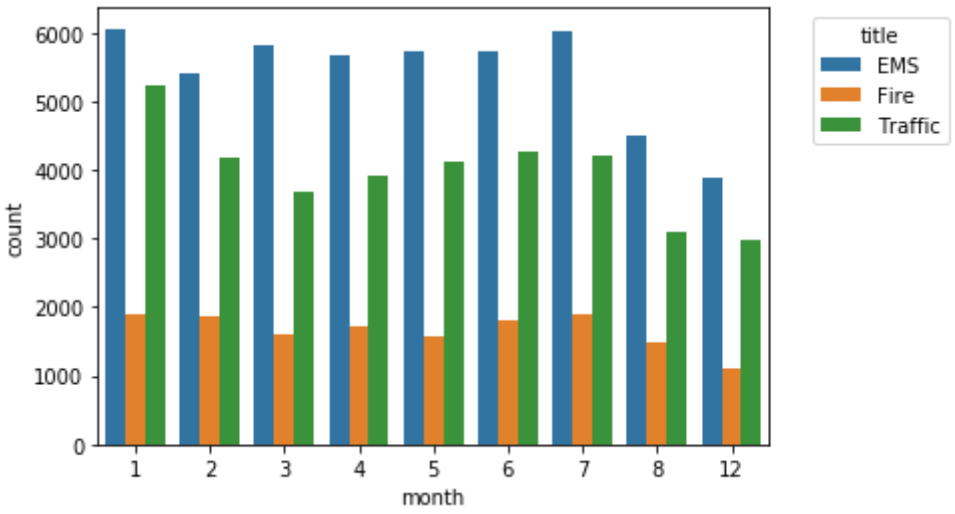
Out[114]: <matplotlib.legend.Legend at 0x7f9de3053c50>



Now do the same for Month:

In [115]:
```
sns.countplot(x='month', data=df, hue='reason' )
plt.legend( title='title', bbox_to_anchor=(1.05, 1), loc='upper left')

```

Out[115]: <matplotlib.legend.Legend at 0x7f9de3db5890>



Missing some Months

In [116]:
```
data = df.groupby('month').count()
data.head(5)
```

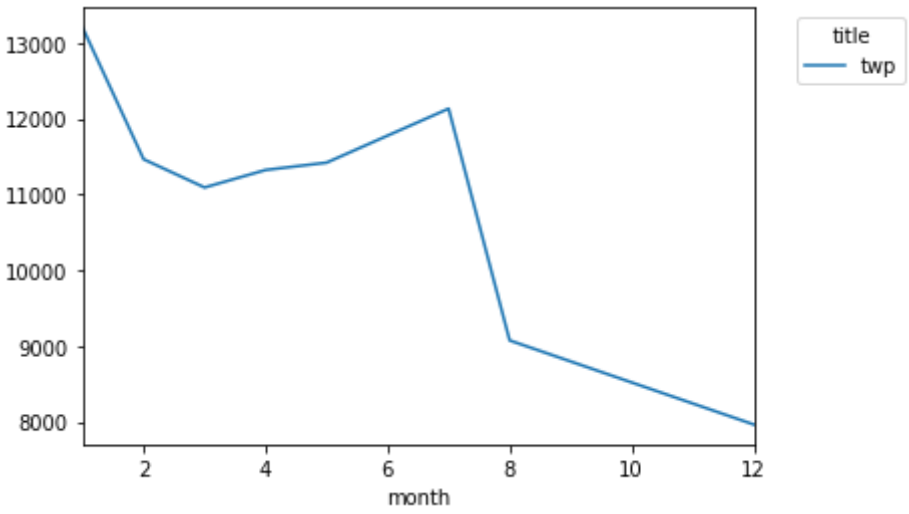
Out[116]:

month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	reason	hour	da
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

Count of calls per month

In [117]:
```
1 data['twp'].plot()
2 plt.legend( title='title', bbox_to_anchor=(1.05, 1), loc='upper left')
3
```

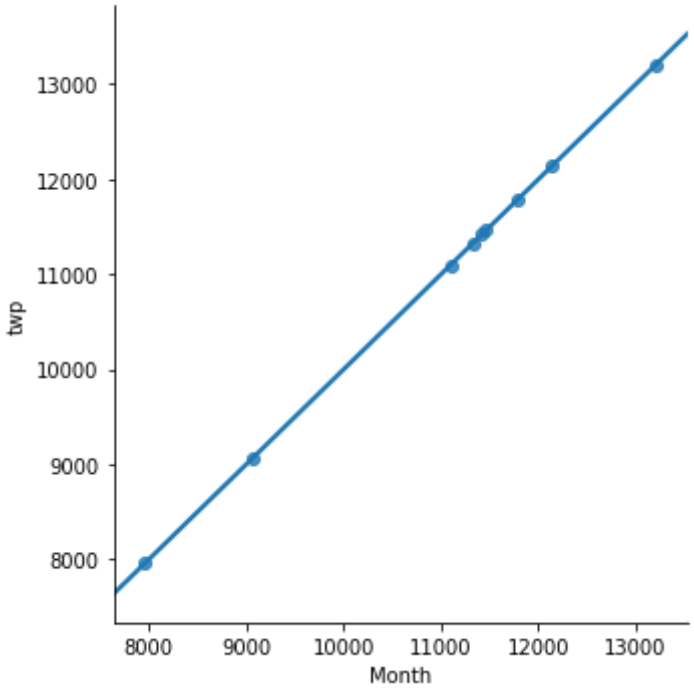
Out[117]: <matplotlib.legend.Legend at 0x7f9de288f5d0>



Linear fit on the number of calls per month

In [118]:
```
1 sns.lmplot(x='Month',y='twp',data=data.reset_index())
```

Out[118]: <seaborn.axisgrid.FacetGrid at 0x7f9de286d210>

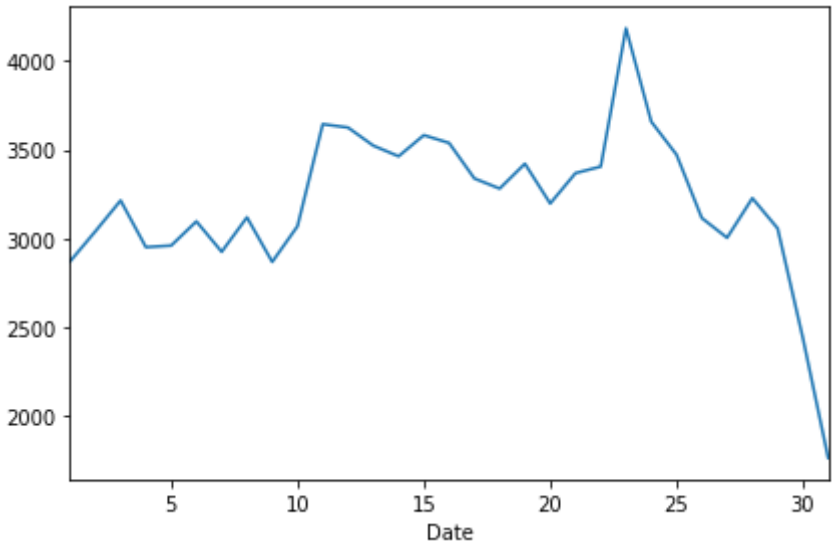


Creating a new column called 'Date' that contains the date from the timeStamp column

In [119]:
```
1 df['date']=df['timeStamp'].apply(lambda t: t.date())
```

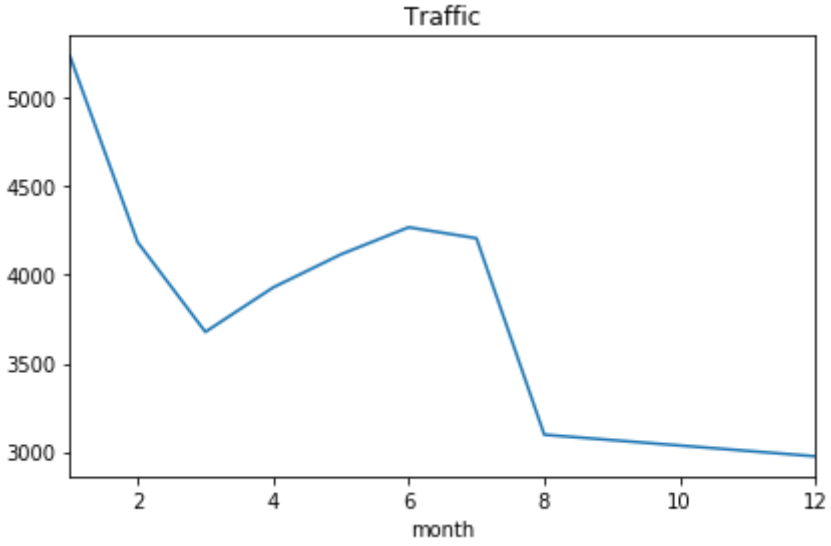
Groupby the Data column

In [120]:
```
df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
```

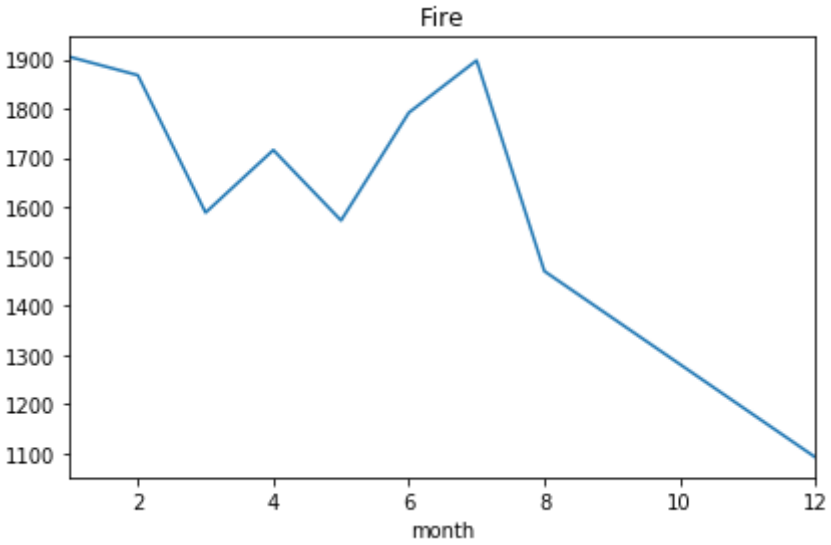


3 separate plots with each plot representing a Reason for the 911 call

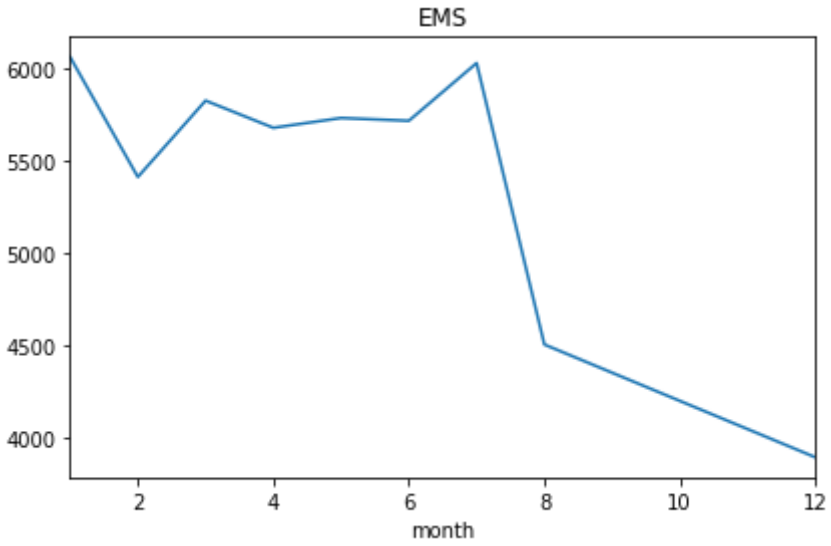
In [123]:
```
df[df['reason']=='Traffic'].groupby('month').count()['twp'].plot()
plt.title('Traffic')
plt.tight_layout()
```



In [122]:
```
df[df['reason']=='Fire'].groupby('month').count()['twp'].plot()
plt.title('Fire')
plt.tight_layout()
```



In [125]:
```
df[df['reason']=='EMS'].groupby('month').count()['twp'].plot()
plt.title('EMS')
plt.tight_layout()
```



**HeatMap Analyses

In [138]:
```
data = df.groupby(by=['Day of Week','hour']).count()['reason'].unstack(
data
```

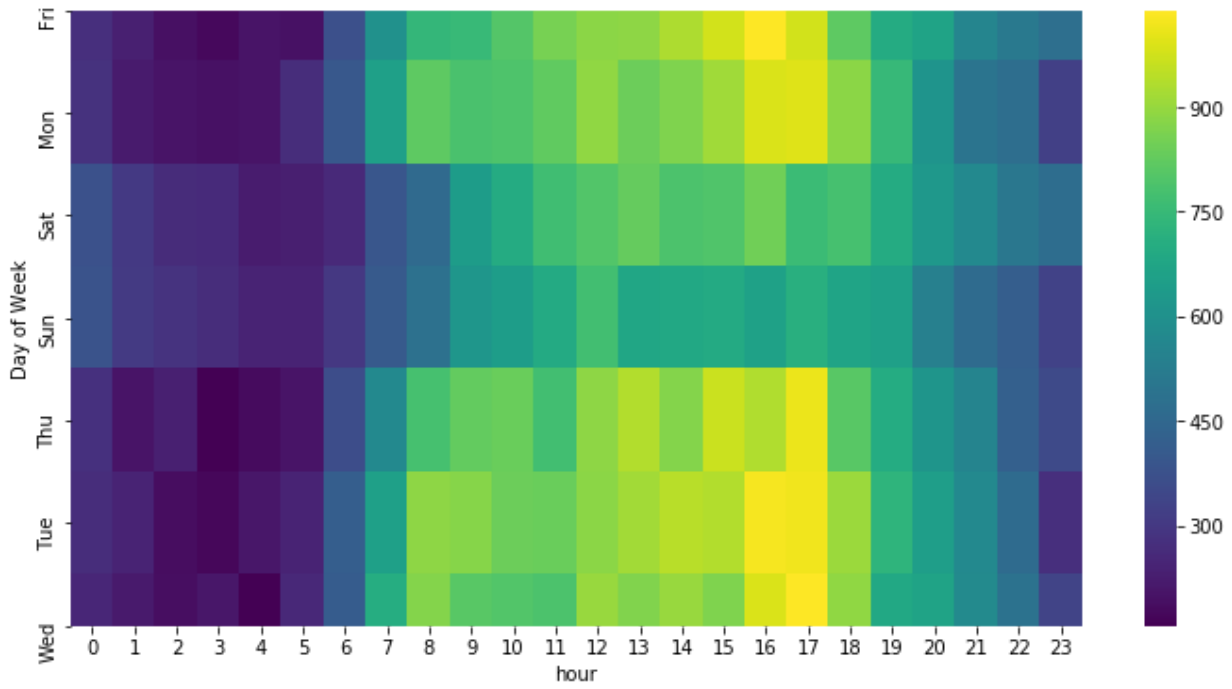
Out[138]:

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	6(
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	6
Sun	383	306	286	268	242	240	300	402	483	620	...	684	691	663	714	670	655	5
Thu	278	202	233	159	182	203	362	570	777	828	...	876	969	935	1013	810	698	6
Tue	269	240	186	170	209	239	415	655	889	880	...	943	938	1026	1019	905	731	6
Wed	250	216	189	209	156	255	410	701	875	808	...	904	867	990	1037	894	686	6(

7 rows × 24 columns

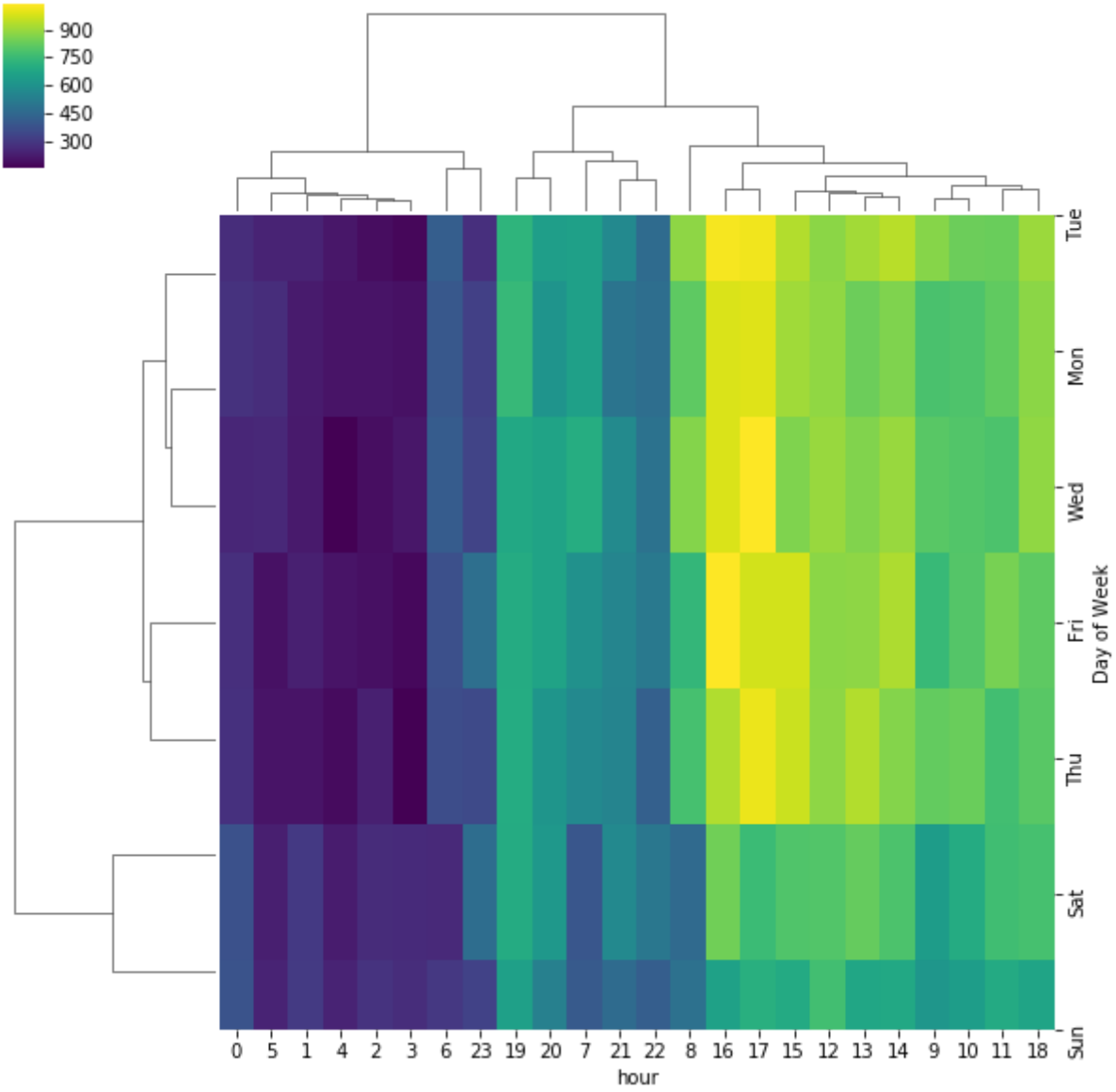
In [146]:
```
plt.figure(figsize=(12,6))
sns.heatmap(data=data,cmap='viridis')


```

Out[146]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9dcc562d10>



In [148]:
```
1 sns.clustermap(data=data, cmap='viridis')
```

Out[148]: <seaborn.matrix.ClusterGrid at 0x7f9dccbdaad0>



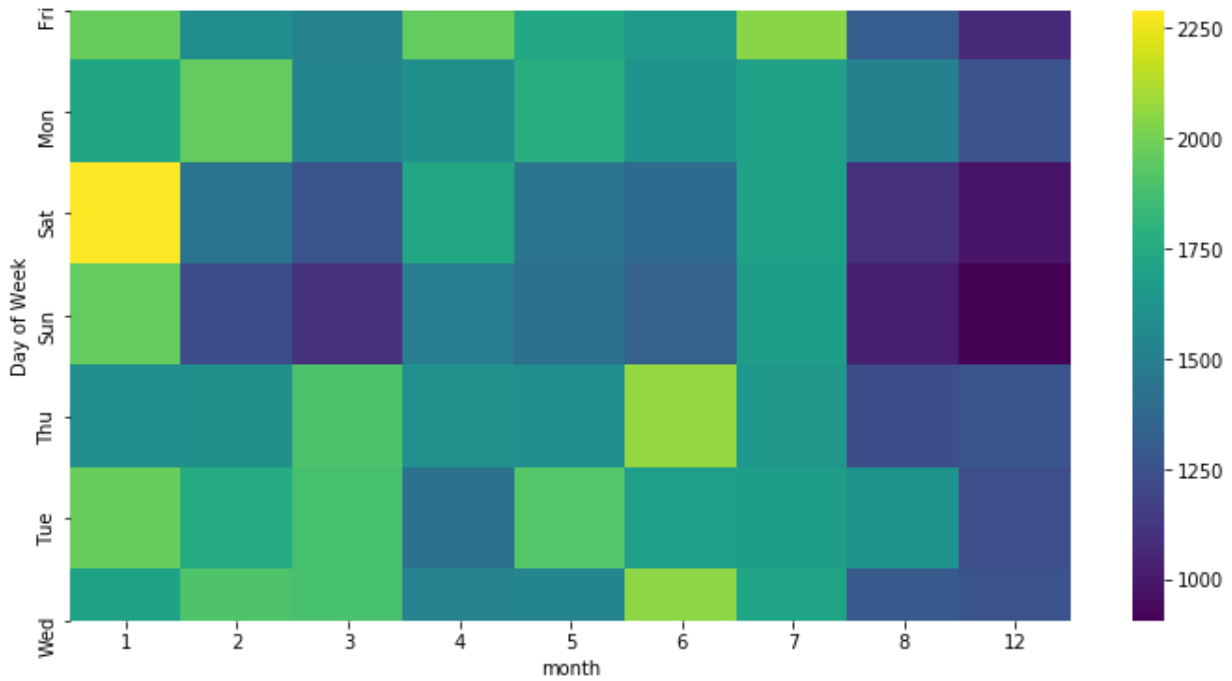
In [150]:
```
data2 = df.groupby(by=['Day of Week','month']).count()['reason'].unstac
data2
```

Out[150]:

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
Tue	1973	1753	1884	1430	1918	1676	1670	1612	1234
Wed	1700	1903	1889	1517	1538	2058	1717	1295	1262

In [153]:
```
plt.figure(figsize=(12,6))
sns.heatmap(data=data2, cmap='viridis')
```

Out[153]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9dcc410f50>



In [155]:
```
1 sns.clustermap(data=data2, cmap='viridis')
```

Out[155]: <seaborn.matrix.ClusterGrid at 0x7f9dce6db810>