Practices work on Seaborn*Numpy* Matplotilb_Visualization, Data is from Kaggle and the work through is from Pierian Data

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)

In [23]:

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

In [24]:

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

In [25]:

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

In [26]:

```
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 [27]:

```
df.head(6)
```

Out[27]:

	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
5	40.253473	-75.283245	CANNON AVE & W 9TH ST; LANSDALE; Station 345;...	19446.0	EMS: HEAD INJURY	2015-12-10 17:40:01	LANSDALE	CANNON AVE & W 9TH ST	1

In [28]:

```
df.tail(4)
```

Out[28]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e
99488	40.006974	-75.289080	LANCASTER AVE & RITTENHOUSE PL; LOWER MERION; ...	19003.0	Traffic: VEHICLE ACCIDENT -	2016-08-24 11:07:02	LOWER MERION	LANCASTER AVE & RITTENHOUSE PL	1
99489	40.115429	-75.334679	CHESTNUT ST & WALNUT ST; NORRISTOWN; Station ...	19401.0	EMS: FALL VICTIM	2016-08-24 11:12:00	NORRISTOWN	CHESTNUT ST & WALNUT ST	1
99490	40.186431	-75.192555	WELSH RD & WEBSTER LN; HORSHAM; Station 352; ...	19002.0	EMS: NAUSEA/VOMITING	2016-08-24 11:17:01	HORSHAM	WELSH RD & WEBSTER LN	1
99491	40.207055	-75.317952	MORRIS RD & S BROAD ST; UPPER GWYNEDD; 2016-08...	19446.0	Traffic: VEHICLE ACCIDENT -	2016-08-24 11:17:02	UPPER GWYNEDD	MORRIS RD & S BROAD ST	1

In [29]:

```
#What are the top 5 zipcodes for 911 calls
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
```

In [30]:

```
#What are the top 5 townships (twp) for 911 calls?
df["twp"].value_counts().head(5)
```

Out[30]:

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

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])
```

In [33]:

```
df.head(5)
```

Out[33]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason
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	EMS
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	EMS
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	Fire
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	EMS
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	EMS

In [34]:

```
# What is the most common Reason for a 911 call based off of this new column?
df["Reason"].value_counts()
```

Out[34]:

```
EMS        48877
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

In [35]:

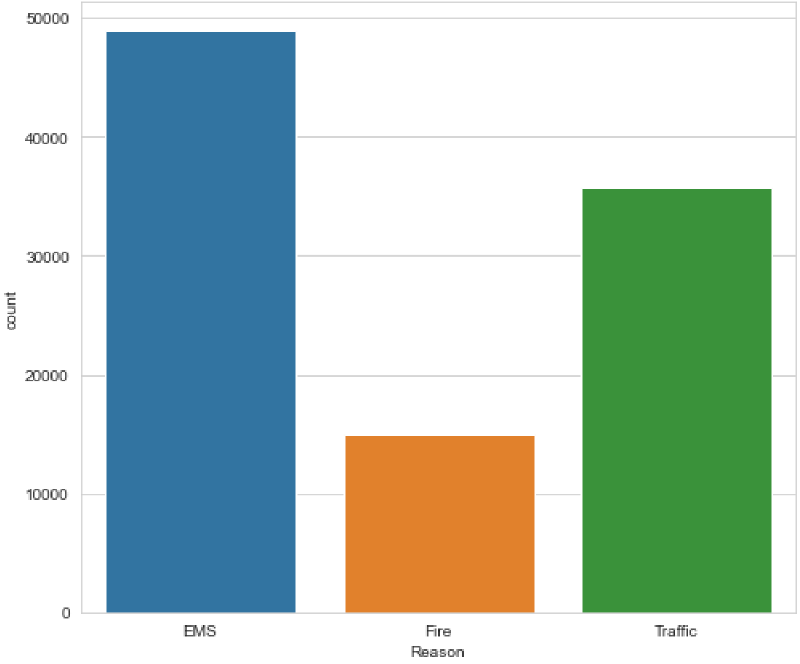
```
# Now use seaborn to create a countplot of 911 calls by Reason.

fig=plt.figure(figsize=(8,7))

sns.countplot(x="Reason",data=df)
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d700624988>
```



In [36]:

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



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

Out[36]:

```
str
```

In [37]:

```
#You should have seen that these timestamps are still strings. Use pd.to_datetime to convert the column from strings to DateTime objects.

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

```
df.head(3)
```

Out[38]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Month	Day of Week
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	EMS	17	12	3
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	EMS	17	12	3
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	Fire	17	12	3

In [39]:

```
# need to know or understand more on DayStamp and time
```

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 [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"].map(dmap)
```

In [42]:

```
df.head()
```

Out[42]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Month	Day of Week
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	EMS	17	12	Thu
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	EMS	17	12	Thu
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	Fire	17	12	Thu
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	EMS	17	12	Thu
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	EMS	17	12	Thu

In [43]:

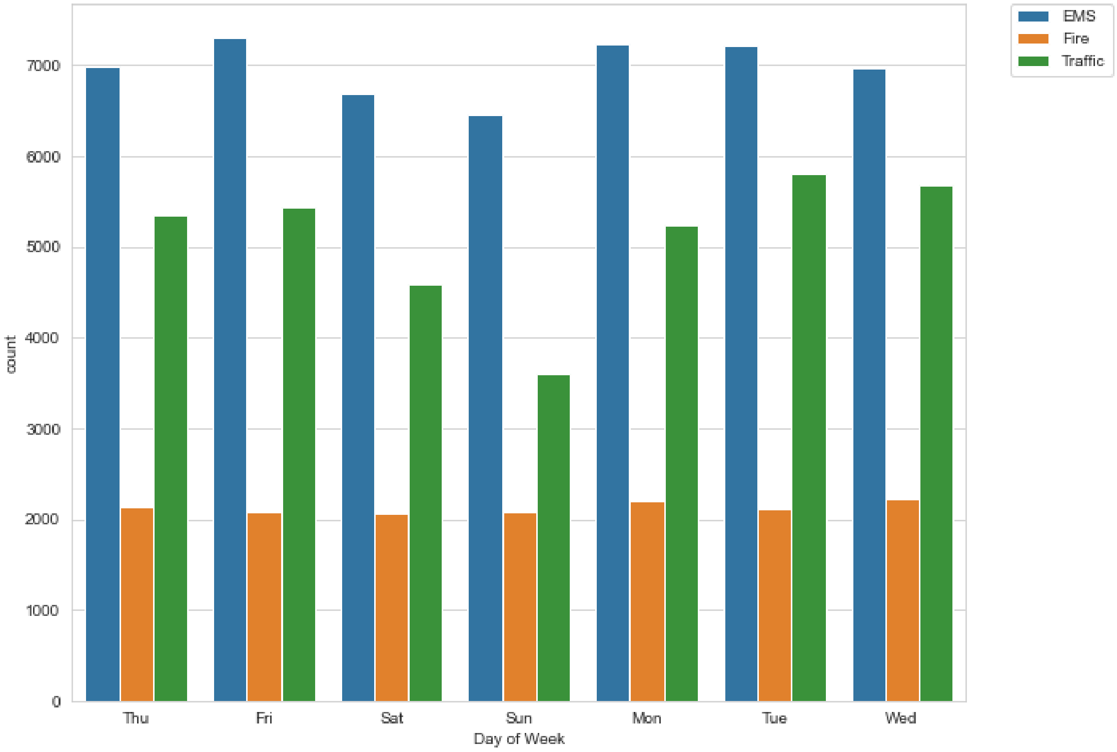
```
# Now use seaborn to create a countplot of the Day of Week column with the hue based off of the Reason column.

fig=plt.figure(figsize =(10,8))
sns.countplot(x="Day of Week", hue="Reason", data=df)

# To relocate the legend
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[43]:

```
<matplotlib.legend.Legend at 0x1d700896d88>
```



In [44]:

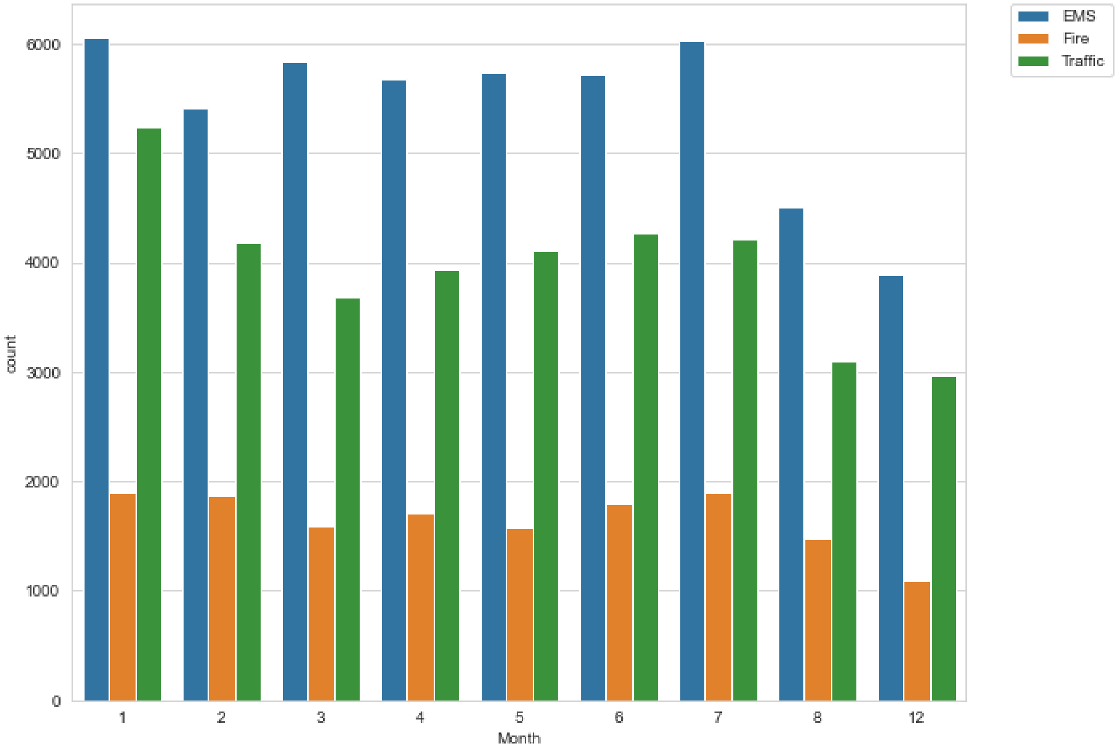
```
#Now do the same for Month:

fig=plt.figure(figsize =(10,8))
sns.countplot(x="Month", hue="Reason", data=df)

# To relocate the legend
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[44]:

```
<matplotlib.legend.Legend at 0x1d776c37288>
```



In [45]:

```
# the above countplot shows 3 missing month.
```

In [46]:

```
byMonth = df.groupby("Month").count()
byMonth.head(7)
```

Out[46]:

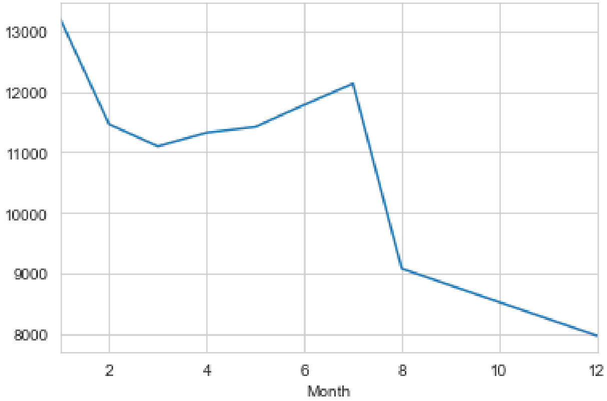
	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Day of Week
Month												
1	13205	13205	13205	11527	13205	13205	13203	13096	13205	13205	13205	13205
2	11467	11467	11467	9930	11467	11467	11465	11396	11467	11467	11467	11467
3	11101	11101	11101	9755	11101	11101	11092	11059	11101	11101	11101	11101
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
6	11786	11786	11786	10212	11786	11786	11777	11732	11786	11786	11786	11786
7	12137	12137	12137	10633	12137	12137	12133	12088	12137	12137	12137	12137

In [47]:

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

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d700be5588>
```



In [48]:

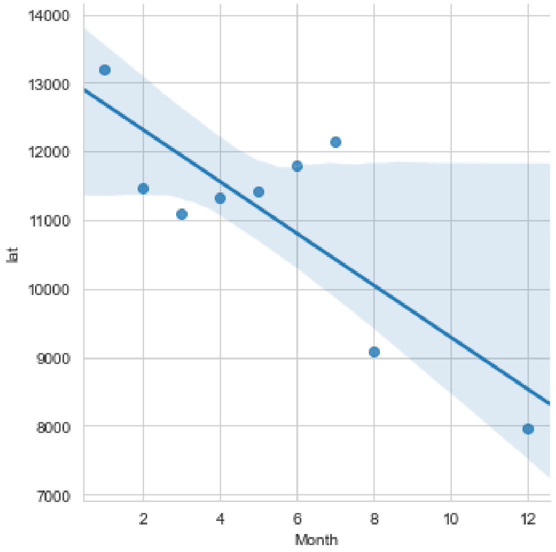
```
# Now create a simple plot off of the dataframe indicating the count of calls per month  also using a drop in from the missing months
```

In [49]:

```
sns.lmplot(x= "Month", y="lat", data= byMonth.reset_index() )
```

Out[49]:

```
<seaborn.axisgrid.FacetGrid at 0x1d700a5d708>
```



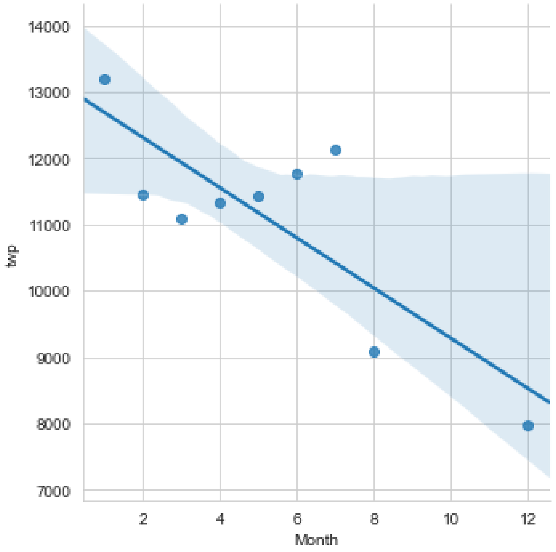
In [50]:

```
# creating   lmplot() to create a linear fit on the number of calls per month. Keep in mind to  reset the index to a column.

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

Out[50]:

```
<seaborn.axisgrid.FacetGrid at 0x1d700ae6ac8>
```



In [51]:

```
# 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 [52]:

```
df["Date"] = df ["timeStamp"].apply (lambda t:t.date())
```

In [53]:

```
df.head()
```

Out[53]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Month	Day of Week	Date
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	EMS	17	12	Thu	2015-12-10
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	EMS	17	12	Thu	2015-12-10
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	Fire	17	12	Thu	2015-12-10
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	EMS	17	12	Thu	2015-12-10
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	EMS	17	12	Thu	2015-12-10

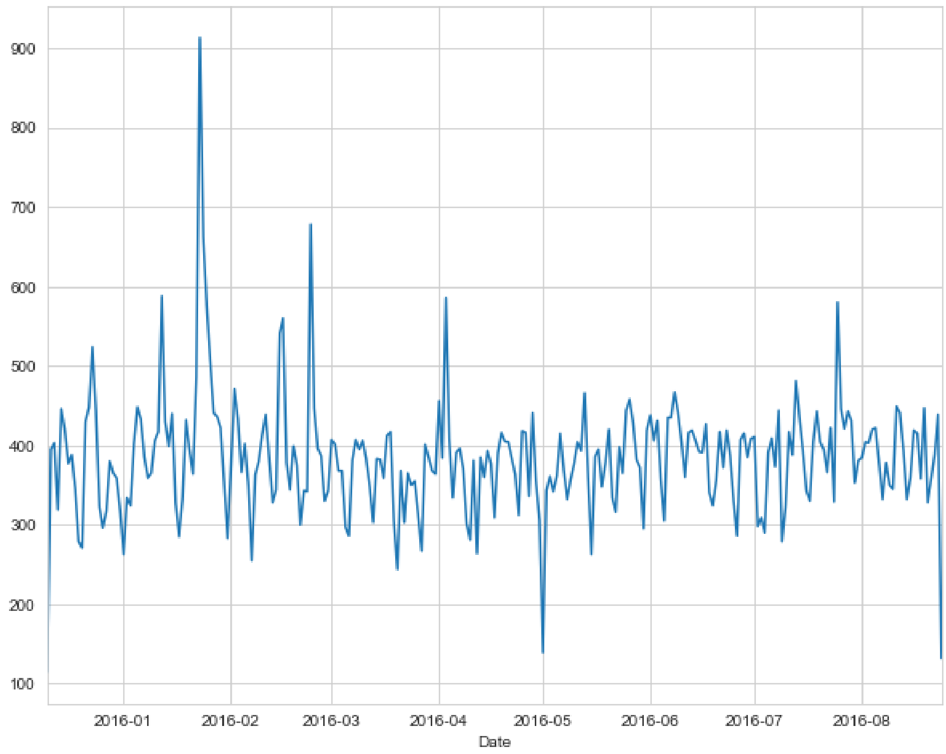
In [54]:

```
fig=plt.figure(figsize=(10,8))

df.groupby("Date").count()["twp"].plot()
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d700b48888>
```



In [55]:

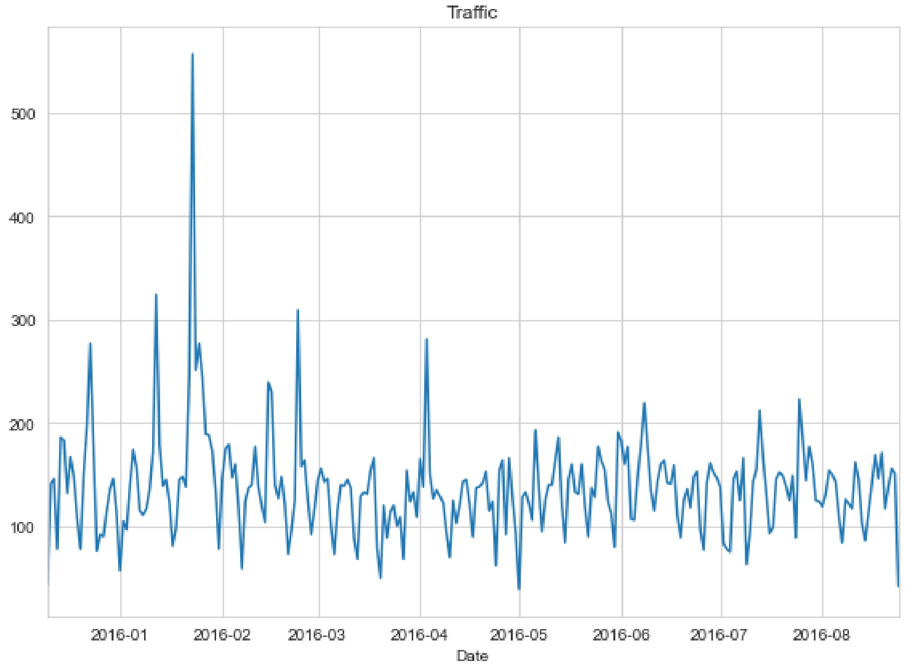
```
# creating 3 separte plot for Reason  each plot represntinig a Reason
```

In [56]:

```
fig=plt.figure(figsize=(8,6))

df[df["Reason"]=="Traffic"].groupby("Date").count()["twp"].plot()


plt.title("Traffic")
plt.tight_layout()
```



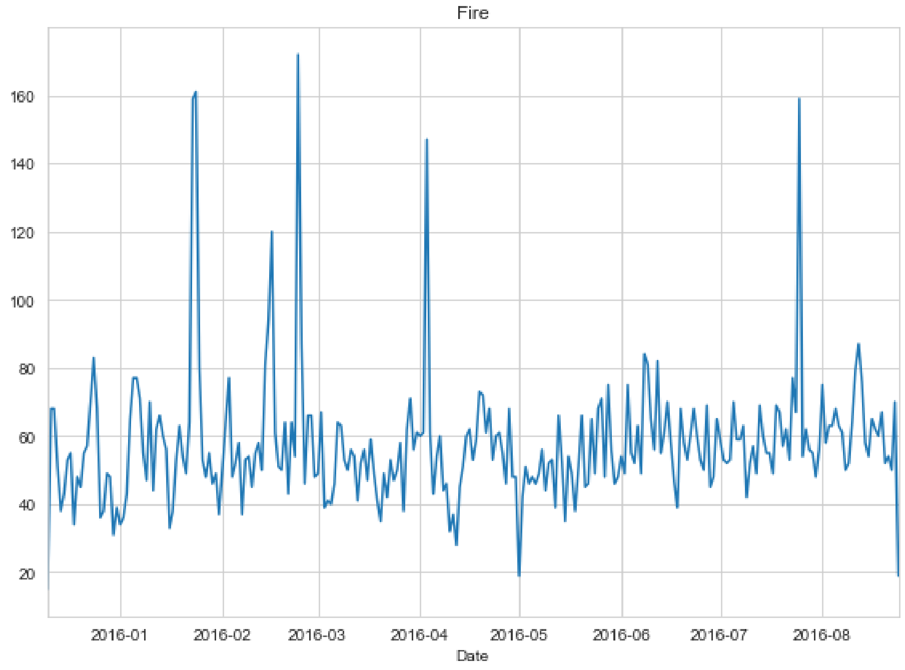
In [ ]:

In [57]:

```
fig=plt.figure(figsize=(8,6))

df[df["Reason"]=="Fire"].groupby("Date").count()["twp"].plot()


plt.title("Fire")
plt.tight_layout()
```

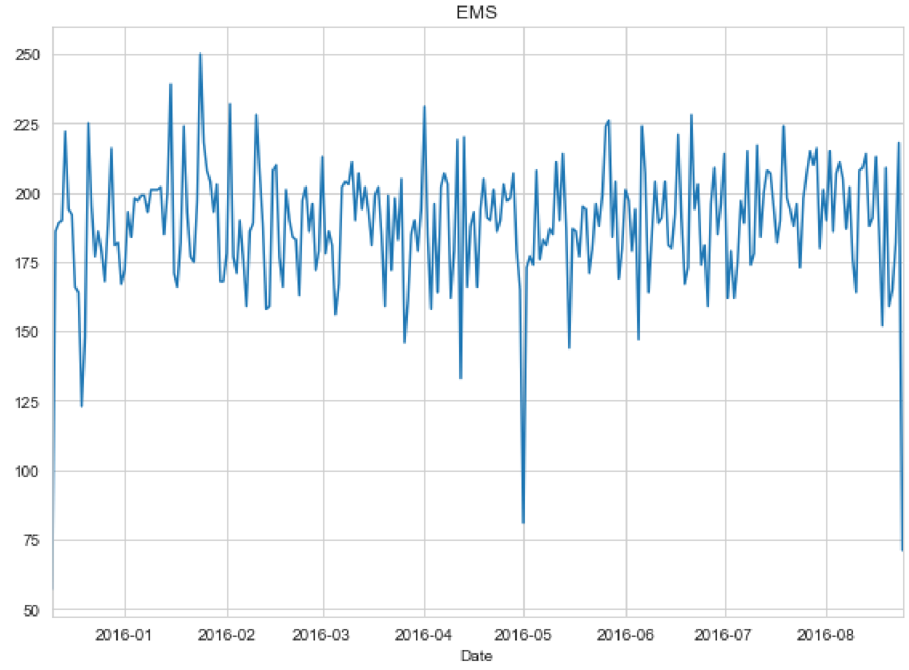


In [58]:

```
fig=plt.figure(figsize=(8,6))

df[df["Reason"]=="EMS"].groupby("Date").count()["twp"].plot()


plt.title("EMS")
plt.tight_layout()
```



This a note to use unstack method

In [59]:

```
# 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 becomeas 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.
```

In [60]:

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

In [61]:

```
dayHour.head(6)
```

Out[61]:

Hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Day of Week																					
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

6 rows × 24 columns

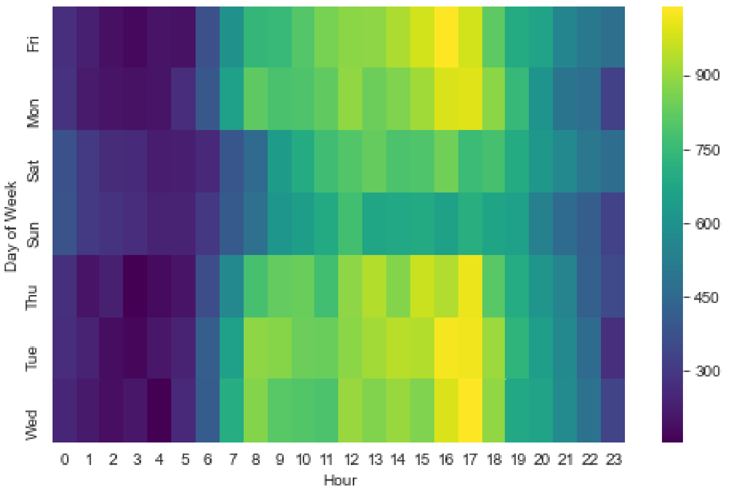
In [62]:

```
fig=plt.figure(figsize=(8,5))

sns.heatmap(dayHour,cmap= "viridis")
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d701b6dd48>
```



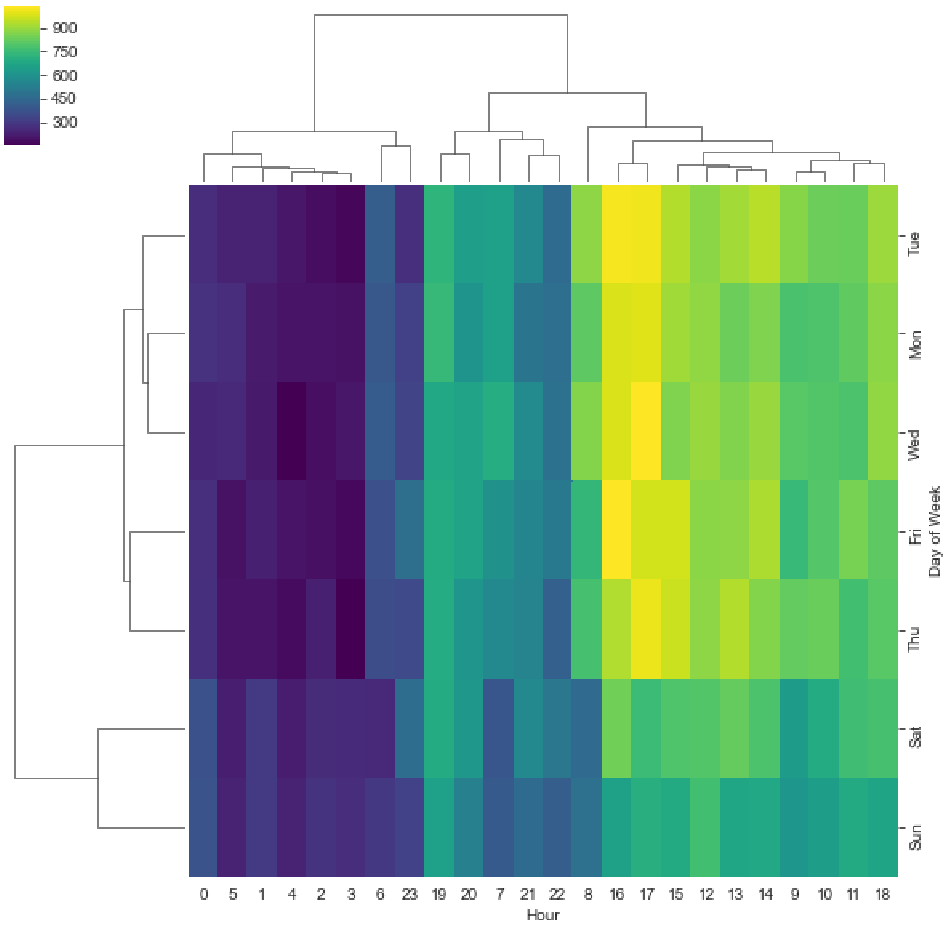
In [68]:

```
#Now create a clustermap using this DataFrame

sns.clustermap(dayHour, cmap="viridis")
```

Out[68]:

```
<seaborn.matrix.ClusterGrid at 0x1d77a95dd88>
```



In [77]:

```
#Now repeating the same plots and operations, for a DataFrame that shows the Month as the column

dayMonth = df.groupby(by=["Day of Week","Month"]).count()["Reason"].unstack()


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

In [78]:

```
dayHour.head()
```

Out[78]:

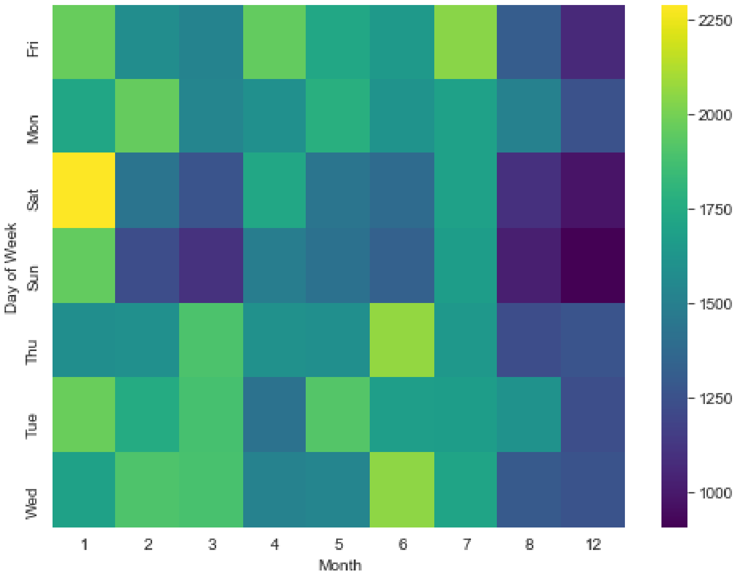
Month	1	2	3	4	5	6	7	8	12
Day of Week									
Fri	1970	1581	1525	1958	1730	1649	2045	1310	1065
Mon	1727	1964	1535	1598	1779	1617	1692	1511	1257
Sat	2291	1441	1266	1734	1444	1388	1695	1099	978
Sun	1960	1229	1102	1488	1424	1333	1672	1021	907
Thu	1584	1596	1900	1601	1590	2065	1646	1230	1266

In [85]:

```
fig=plt.figure(figsize=(8,6))
sns.heatmap(dayMonth,cmap= "viridis")
```

Out[85]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d705e39f88>
```

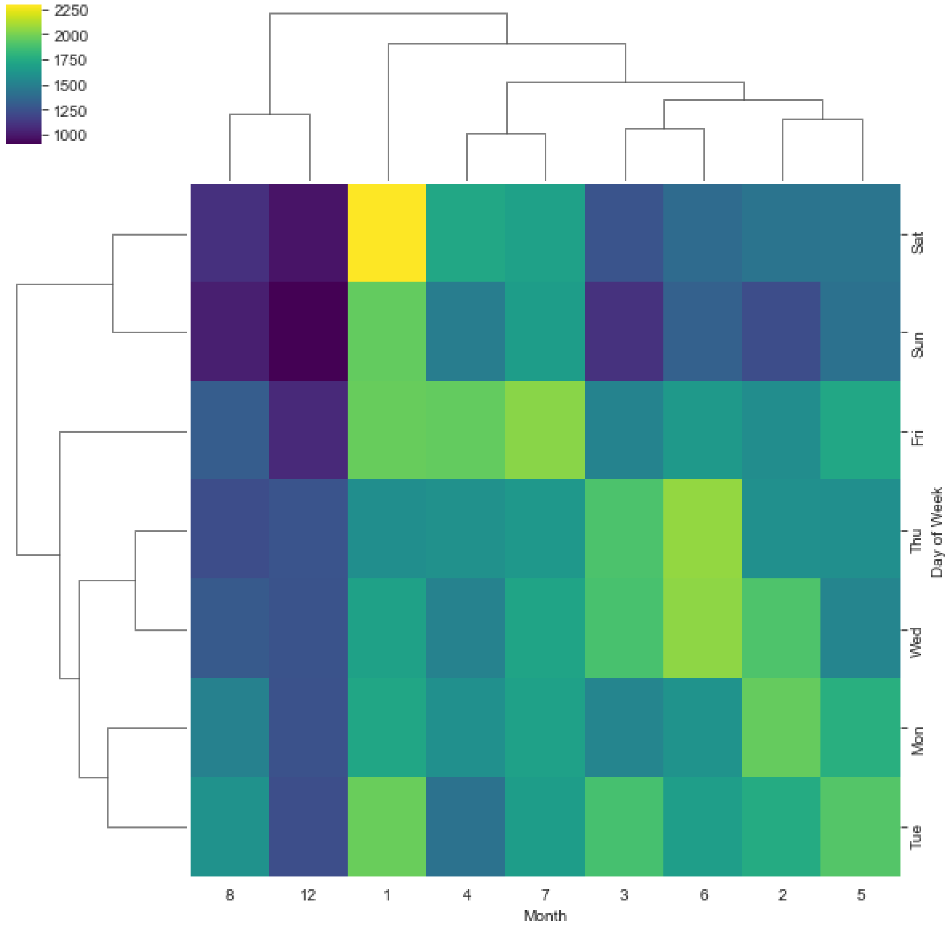


In [83]:

```
fig=plt.figure(figsize=(8,6))
sns.clustermap(dayMonth,cmap= "viridis")
```

Out[83]:

```
<seaborn.matrix.ClusterGrid at 0x1d705375c08>
<Figure size 576x432 with 0 Axes>
```


2250
2000
1750
1500
1250
1000
Sat
Sun
Fri
Thu
Wed
Mon
Tue
Day of Week
8
12
1
4
7
3
6
2
5
Month

In [ ]: