For this EDA project we will be analyzing some 911 call data from [Kaggle (https://www.kaggle.com/mchirico/montcoalert)](https://www.kaggle.com/mchirico/montcoalert) for our Machine Learning Project. 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

** Import numpy and pandas **

In [1]:
```
import numpy as np
import pandas as pd
```

** Import visualization libraries and set %matplotlib inline. **

In [2]:
```
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 [3]:
```
df = pd.read_csv('911.csv')
```

** Check the info() of the df **

In [4]:
```
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
```

Some of the rows like zip code, addr, etc have some null values

** Check the head of df **

In [5]:
```
df.describe()
```

Out[5]:

	lat	lng	zip	e
count	99492.000000	99492.000000	86637.000000	99492.0
mean	40.159526	-75.317464	19237.658298	1.0
std	0.094446	0.174826	345.344914	0.0
min	30.333596	-95.595595	17752.000000	1.0
25%	40.100423	-75.392104	19038.000000	1.0
50%	40.145223	-75.304667	19401.000000	1.0
75%	40.229008	-75.212513	19446.000000	1.0
max	41.167156	-74.995041	77316.000000	1.0

In [6]:
```
df.head(3)
```

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	REINE & 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	BRIAF WHITE
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

** Determining the top 5 zipcodes for 911 calls? **

In [7]:
```
df = df.drop('e', axis=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
```

We have dropped coloum 'e' becuase it didnt offer any advantage as the coloum only have the value 1 throught

We are trying to see where the number of callsare originating from

In [8]:
```
df.head(5)
```

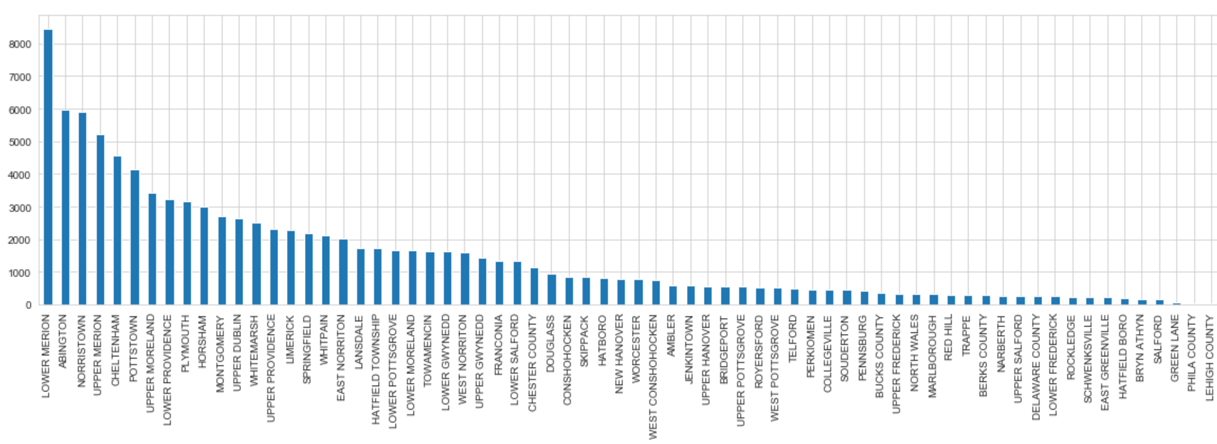
Out[8]:

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

** Determining the top townships (twp) for 911 calls? **

In [9]:
```
df['twp'].value_counts().head(5)
plt.subplots(figsize=(20,5))
df['twp'].value_counts().plot(kind='bar')
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x11f6bcf8c88>



We try to visualise the number of calls from each township

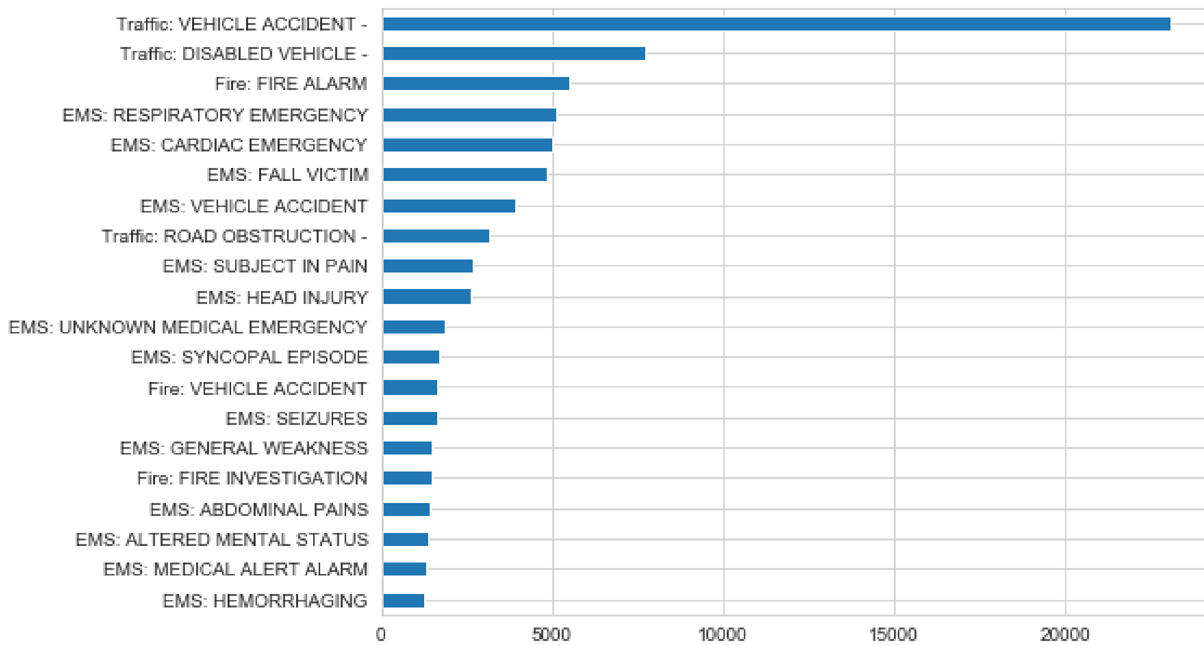
** Take a look at the 'title' column, how many unique title codes are there? **

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

Out[10]:
```
110
```

Creating new features

In [11]:
```
plt.subplots(figsize=(8,6))
df['title'].value_counts().sort_values(ascending=False).head(20).plot(kind='barh'
plt.gca().invert_yaxis()#creating a plot of how many unique title and reason for
```



We see the actual reason that people are calling 911

In [12]:
```
df['Reason'] = df['title'].apply(lambda title: title.split(':')[0])
```

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

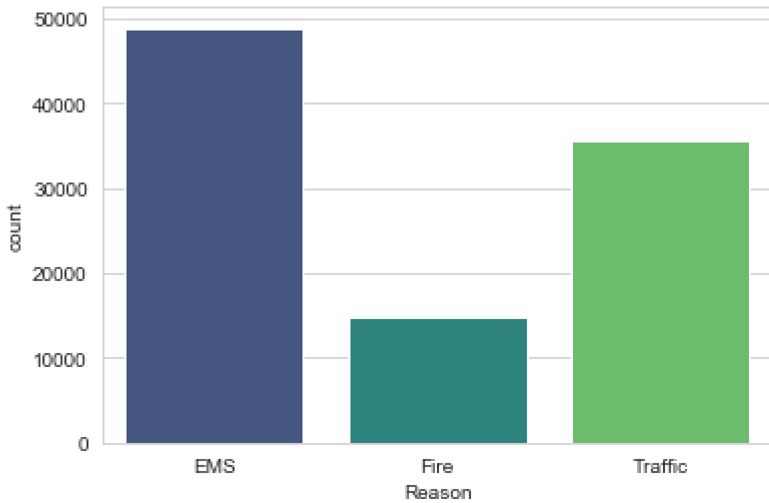
In [13]:
```
df['Reason'].value_counts()
```

Out[13]:
```
EMS        48877
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

** Now we are using seaborn to create a countplot of 911 calls by Reason. **

In [14]:
```
sns.countplot(x='Reason',data=df,palette='viridis')
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11f6f670e08>
```



We have clasified the total calls into 3 categaries mainly EMS, Fire and Traffic

---

** Now let us begin to focus on time information.**

In [15]:
```
#Here we have purified a subtype column a little bit more - replacing (+ with &)
df['type'], df['subtype'] = df['title'].str.split(': ', 1).str
df = df.drop('title', axis=1) #drop 'title' columns
df['subtype'] = df['subtype'].replace({'\+': '&', '\-': ''}, regex=True).map(lamb
```

```
C:\Users\aryan\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarni
ng: Columnar iteration over characters will be deprecated in future releases.
```

In [16]:
```
total = df['subtype'].value_counts().sort_values(ascending=False)
percent = (df['subtype'].value_counts()*100/df['subtype'].value_counts().sum()).s
subtype_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
subtype_data.head(10)#here we have analysed the main reason and count of 911 call
```

Out[16]:

	Total	Percent
VEHICLE ACCIDENT	28639	28.785229
DISABLED VEHICLE	7703	7.742331
FIRE ALARM	5510	5.538134
RESPIRATORY EMERGENCY	5112	5.138102
CARDIAC EMERGENCY	5012	5.037591
FALL VICTIM	4863	4.887830
ROAD OBSTRUCTION	3144	3.160053
SUBJECT IN PAIN	2687	2.700720
HEAD INJURY	2631	2.644434
UNKNOWN MEDICAL EMERGENCY	1874	1.883569

The total number of calls and the percentage out of the total for each category

In [17]:
```
type(df['timeStamp'].iloc[0])#to analyse timestamp here we have find the type of
```

Out[17]: str

In [18]:
```
df['timeStamp'] = pd.to_datetime(df['timeStamp'])# to convert the column from str
```

In [19]:
```
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)
df['year'] = df['timeStamp'].dt.year
df['hour'] = df['timeStamp'].dt.hour
df['minute'] = df['timeStamp'].dt.minute
df['weekday'] = df['timeStamp'].dt.day_name
#creating 3 new columns called Hour, Month, and Day of Week from DateTime objects
```

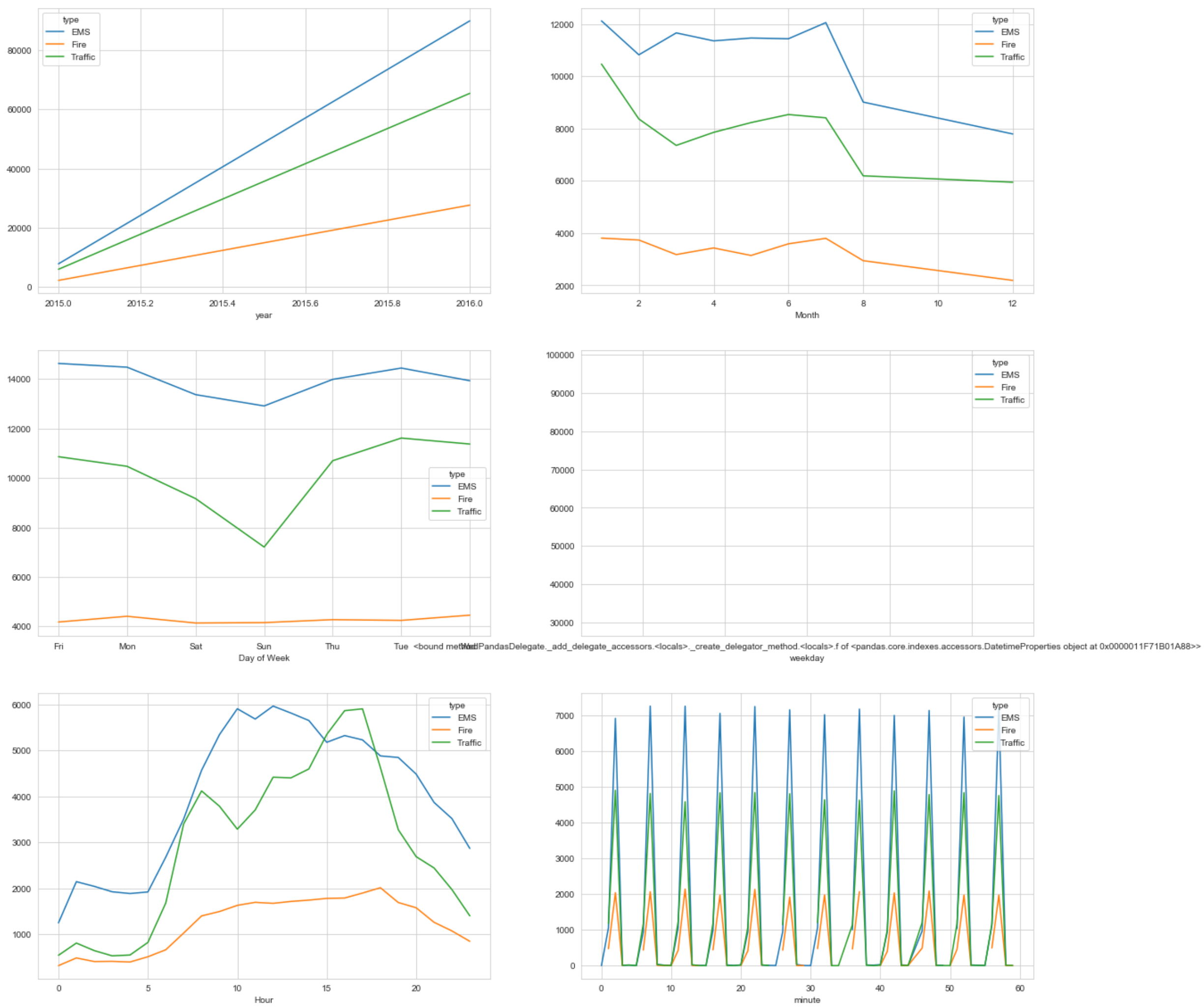
In [20]:
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}# using this dict
```

In [21]:
```
df['Day of Week'] = df['Day of Week'].map(dmap)#mapping the dictionary created ab
```

** Now we have used seaborn to create a countplot of the Day of Week column with the hue based off of the Reason column. **

In [22]:
```
fig,ax = plt.subplots(3, 2, figsize=(20, 20))
df[['type','year']].pivot_table(index=['year'], columns=['type'], aggfunc=np.cour
df[['type','Month']].pivot_table(index=['Month'], columns=['type'], aggfunc=np.cc
df[['type','Day of Week']].pivot_table(index=['Day of Week'], columns=['type'], a
df[['type','weekday']].pivot_table(index=['weekday'], columns=['type'], aggfunc=r
df[['type','Hour']].pivot_table(index=['Hour'], columns=['type'], aggfunc=np.cour
df[['type','minute']].pivot_table(index=['minute'], columns=['type'], aggfunc=np.
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11f71a1c7c8>
```

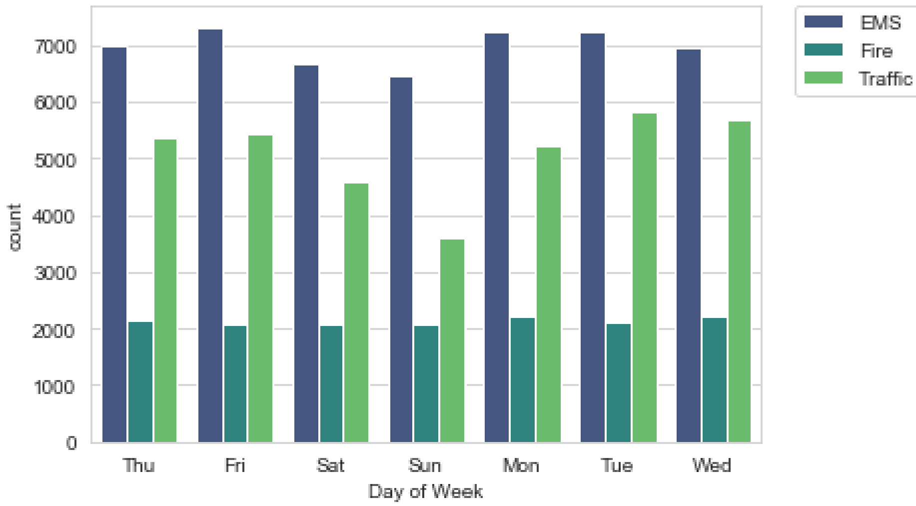


Trying to see when the most number of calls occur. We see that the highest number of calls generally occur in the month of Jan. The highest number of calls occuring in days are on fairly consistent but have higher traffic related calls on tuesday and wednesday. The highest number of calls were made between 3pm to 5 pm.

In [23]:
```
sns.countplot(x='Day of Week',data=df,hue='Reason',palette='viridis')

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

Out[23]: <matplotlib.legend.Legend at 0x11f718a46c8>



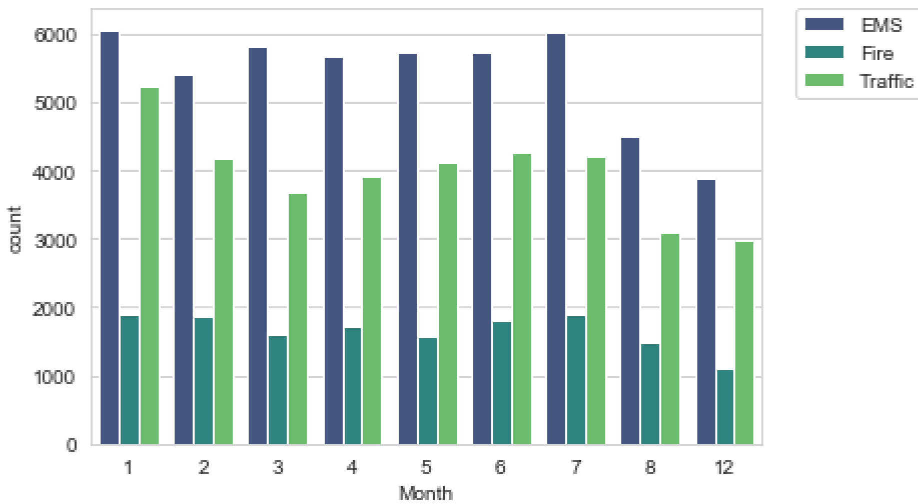
Fire is fairly consistent. EMS highest calls are on Friday while for traffic highest calls are on Tuesday

** Now doing the same for Month:**

In [24]:
```
sns.countplot(x='Month',data=df,hue='Reason',palette='viridis')

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

Out[24]:
```
<matplotlib.legend.Legend at 0x11f70c04208>
```



the highest month for the calls are jan and july

** We noticed some months being missing from the month column in the above plot **

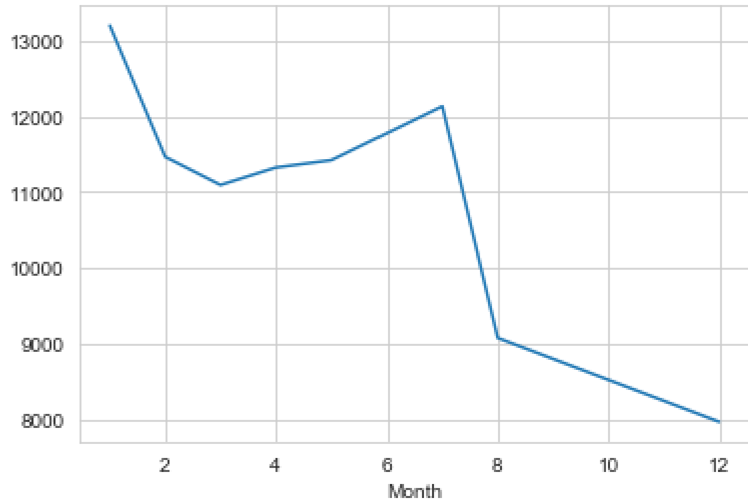
In [25]:
```
# It is missing some months! 9,10, and 11 are not there.
```

In [26]:
```
byMonth = df.groupby('Month').count()
```

** Now we have created a simple plot off of the dataframe indicating the count of calls per month for better analysis. **

In [27]:
```
byMonth['twp'].plot()
```

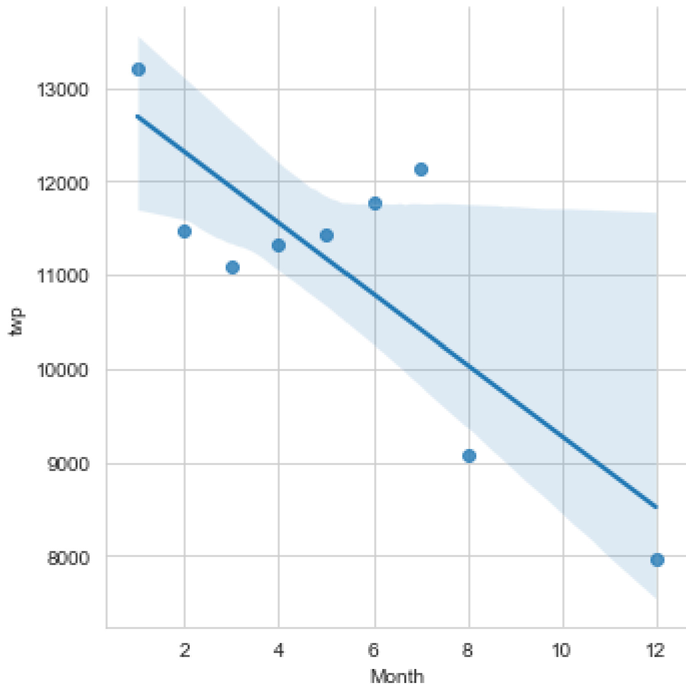
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x11f70b82108>



The highest number of calls in Jan and lowest in Dec, so we can assume that the people are going to vaction in dec and coming back in Jan and thus the calls are increasing

In [28]:
```
sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x11f70b39bc8>

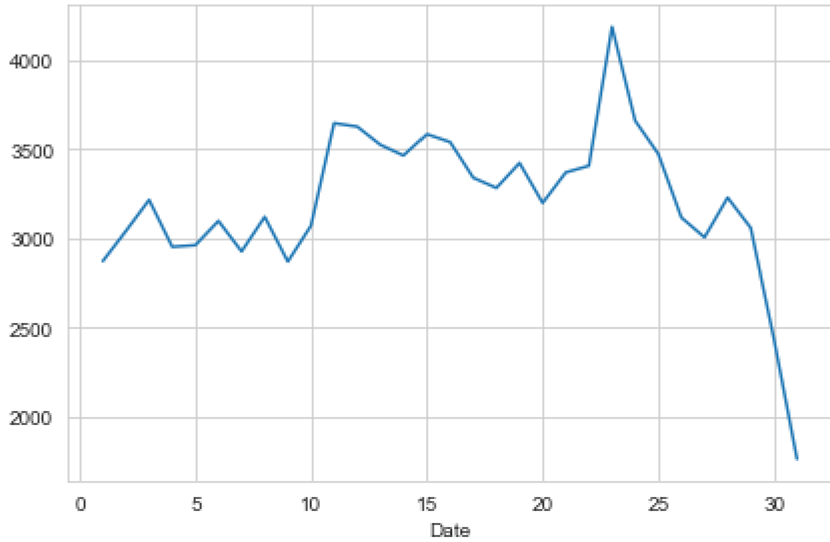


Create a new column called 'Date' that contains the date from the timeStamp column.

In [29]:
```
df['Date'] = df['timeStamp'].dt.day
```

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

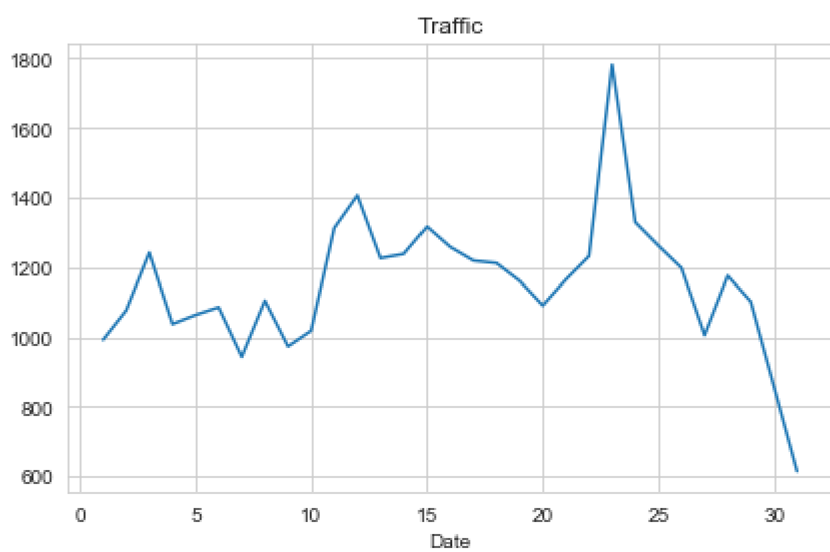
In [30]:
```
df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
```



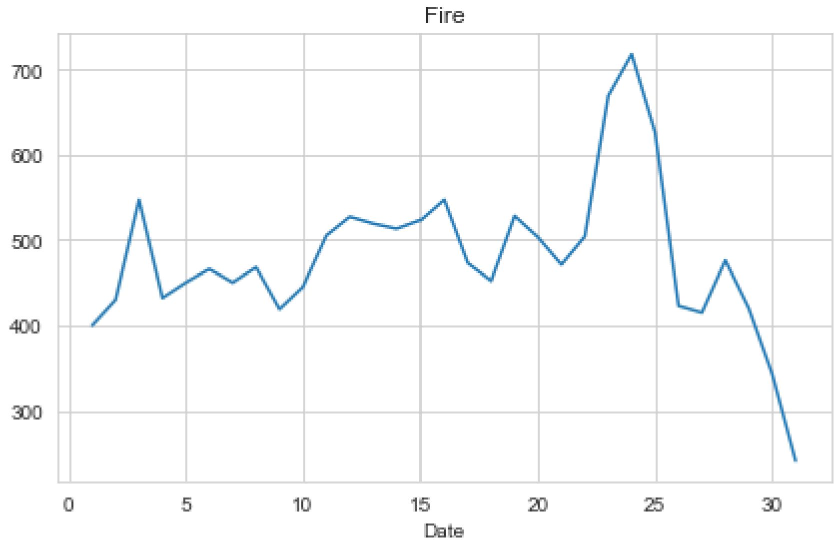
The highest number of calls generally occur on between 20th to 25th

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

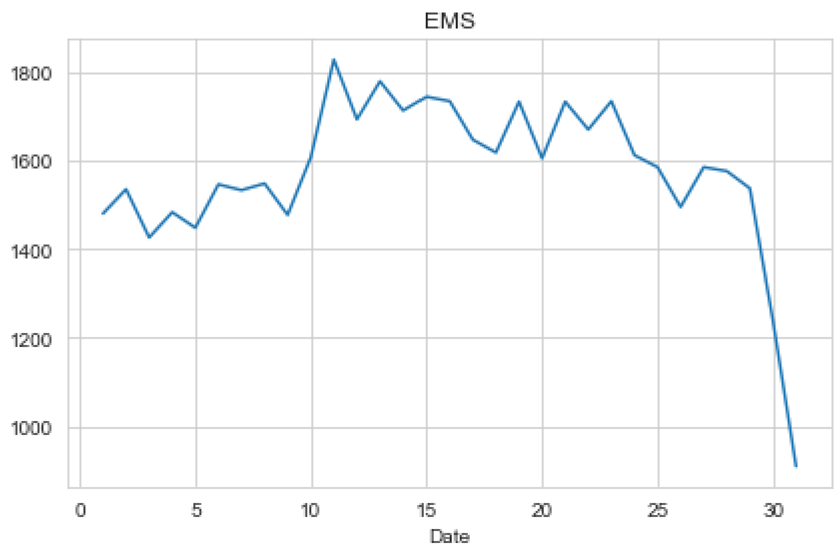
In [31]:
```
df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()
plt.title('Traffic')
plt.tight_layout()#plot for traffic reason for 911 call
```



In [32]:
```
df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
plt.title('Fire')
plt.tight_layout()#plot for Fire reason for 911 call
```



In [33]:
```
df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()
plt.title('EMS')
plt.tight_layout()#plot for EMS reason for 911 call
```



In [34]:
```
dayHour = df.groupby(by=['Day of Week','Hour']).count()['Reason'].unstack()
dayHour.head()#here we have restructure the dataframe so that the columns become
```

Out[34]:

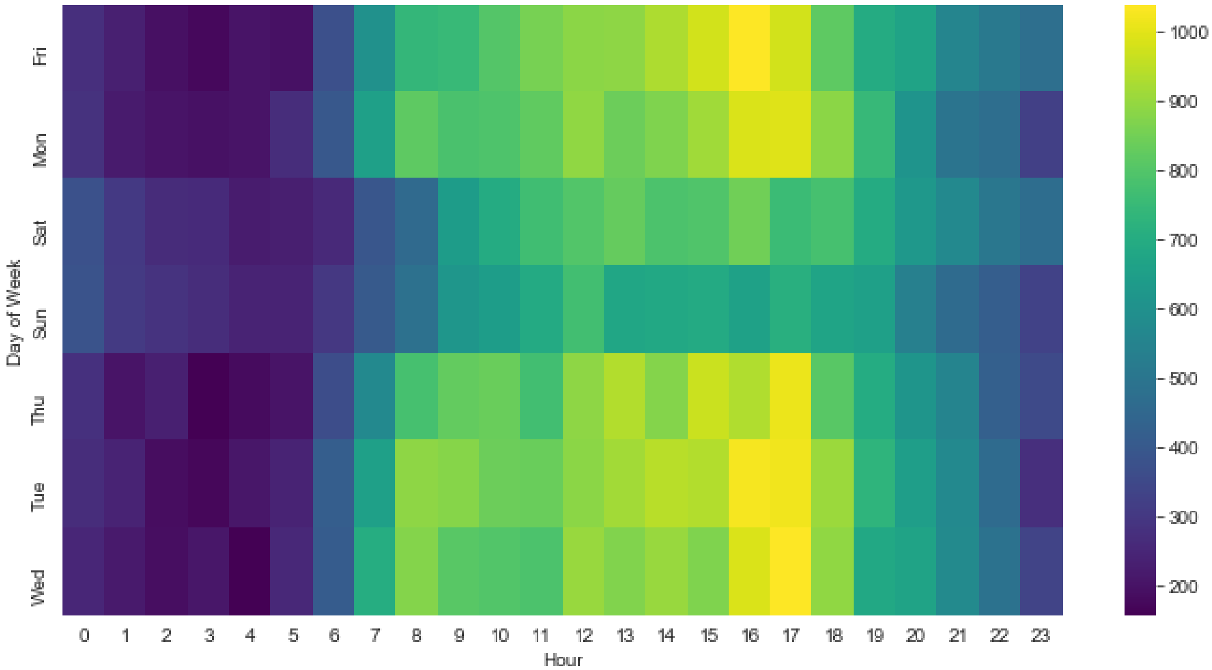
Hour Day of Week	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	
Fri	275	235	191	175	201	194	372	598	742	752	...	932	980	1039	980	820	696	6
Mon	282	221	201	194	204	267	397	653	819	786	...	869	913	989	997	885	746	6
Sat	375	301	263	260	224	231	257	391	459	640	...	789	796	848	757	778	696	6
Sun	383	306	286	268	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

5 rows × 24 columns

** Now create a HeatMap using this new DataFrame. **

In [35]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayHour,cmap='viridis')
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x11f709f13c8>

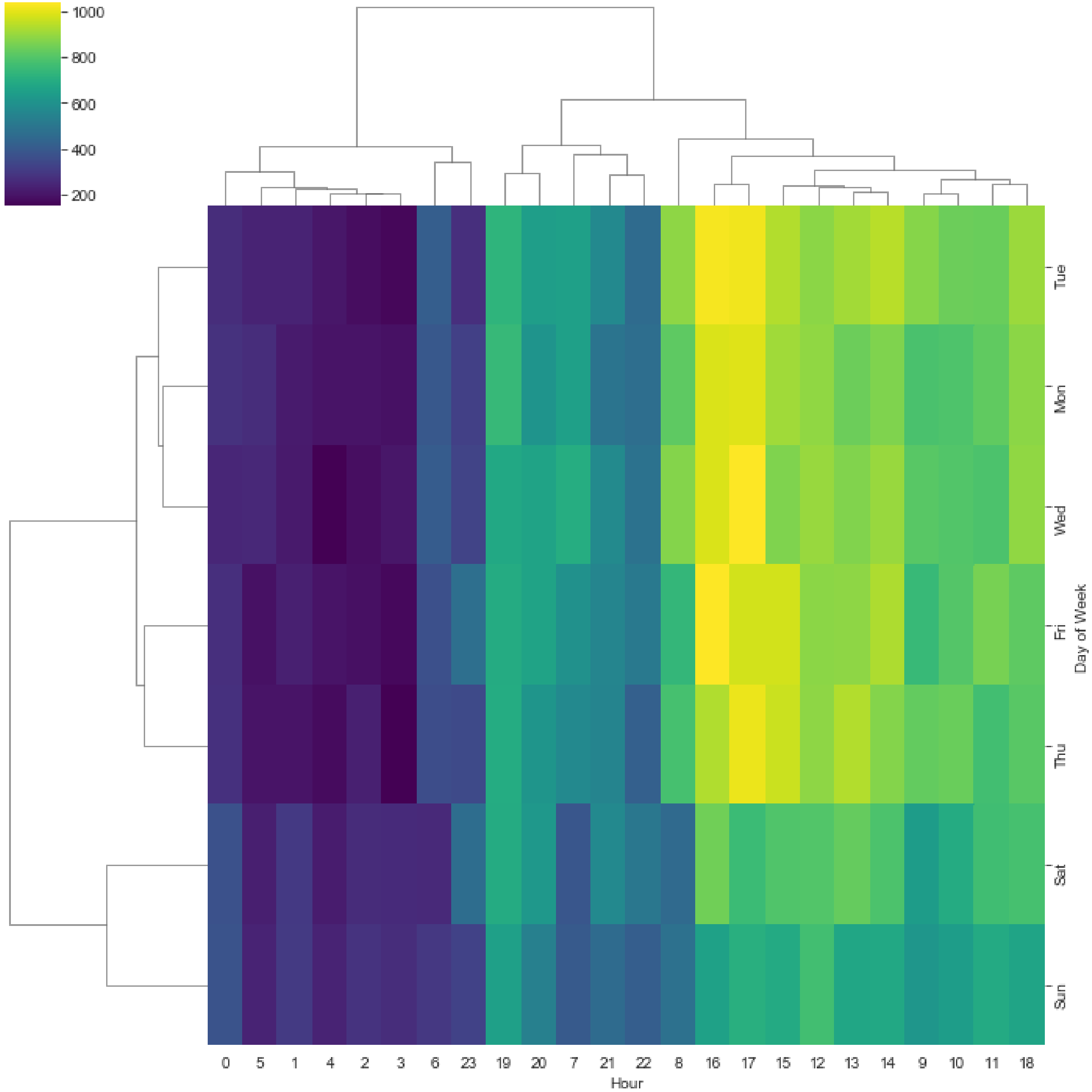


Creating a heatmap for better visualisation and seeing this we can conlude that genrally most number of calls occur on weekdays between 3 to 5pm.

** Now we have created a clustermap using this DataFrame. **

In [36]:
```
sns.clustermap(dayHour,cmap='viridis')
```

Out[36]: <seaborn.matrix.ClusterGrid at 0x11f70907108>



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

In [37]:
```
dayMonth = df.groupby(by=['Day of Week','Month']).count()['Reason'].unstack()
dayMonth.head()
```

Out[37]:

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

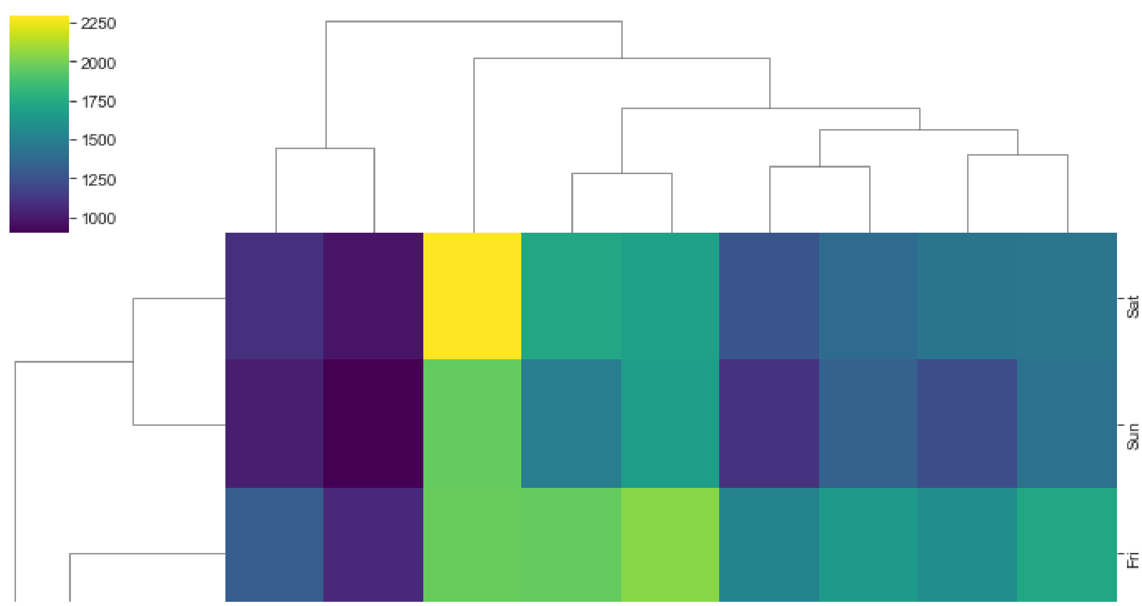
In [38]:
```
plt.figure(figsize=(12,6))
sns.heatmap(dayMonth,cmap='viridis')
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x11f707cdb48>

Creating a heatmap for month and days. We see that Jan has the highest number of calls and Dec has the lowest number of calls indicating that the people are going to vaction in dec and coming back in Jan

In [39]:
```
sns.clustermap(dayMonth,cmap='viridis')
```

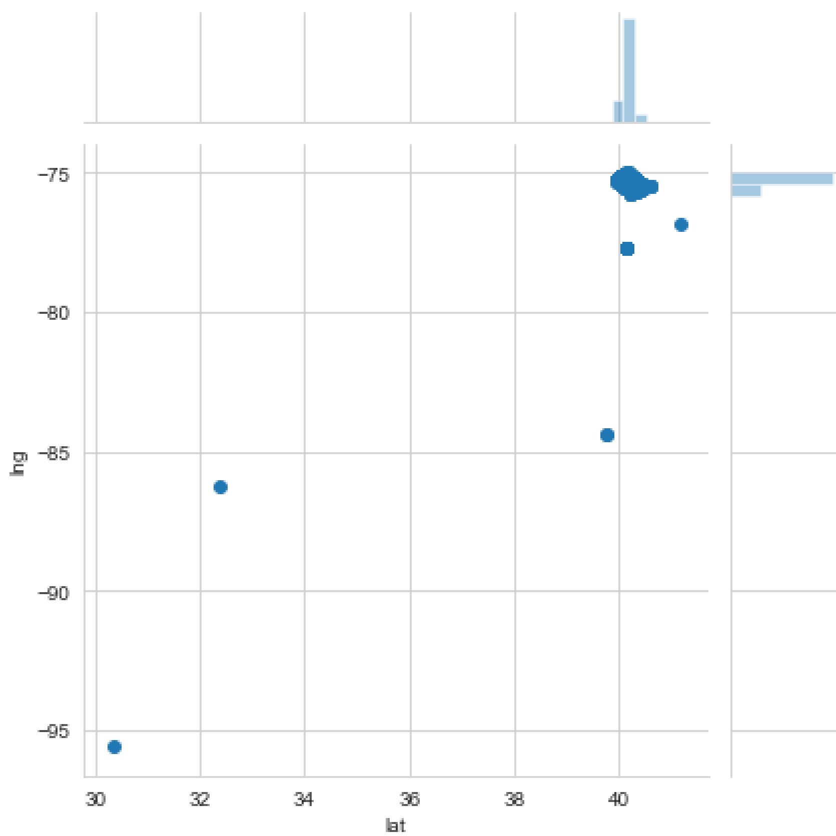
Out[39]: <seaborn.matrix.ClusterGrid at 0x11f70a20dc8>



Geographic Analysis (lattitude-longitude)

In [40]:
```
sns.jointplot(x='lat', y='lng', data=df, kind='scatter')#creating a joint plot oj
```

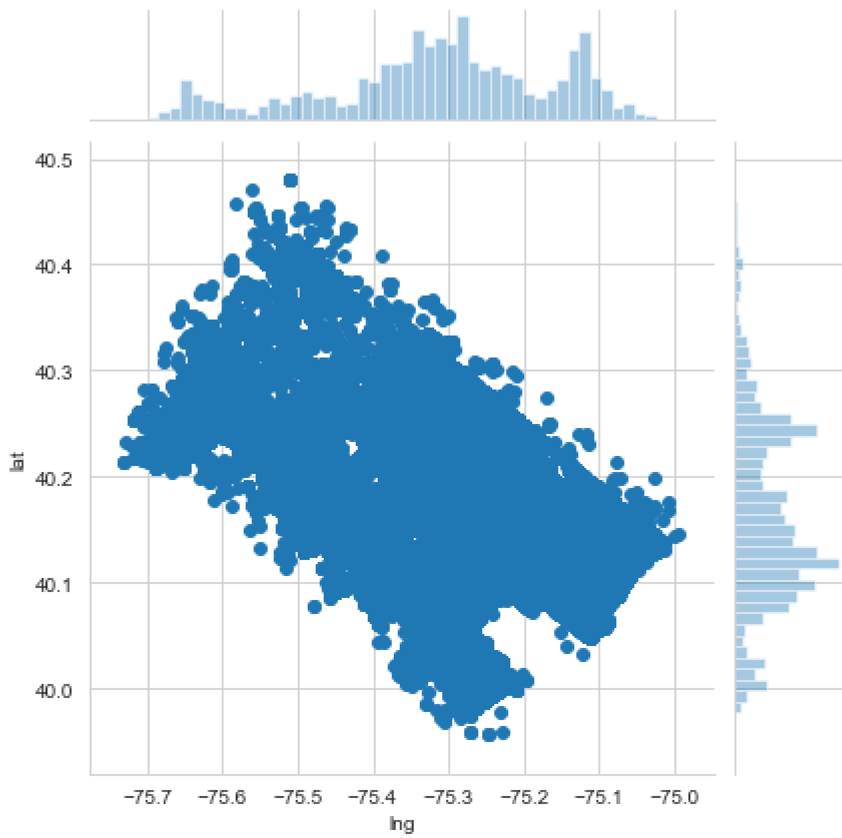
Out[40]: <seaborn.axisgrid.JointGrid at 0x11f70795dc8>



Doing visual analysis of the points from where the calls were originated from

In [41]:
```
# Removing outliers - SD of 4 and 10 as a limit of lat and lng respectively to co
df_geo=df[(np.abs(df["lat"]-df["lat"].mean())<=(4*df["lat"].std())) & (np.abs(df[
df_geo.reset_index().drop('index',axis=1,inplace=True)
sns.jointplot(data=df_geo,x='lng',y='lat',kind='scatter')
```

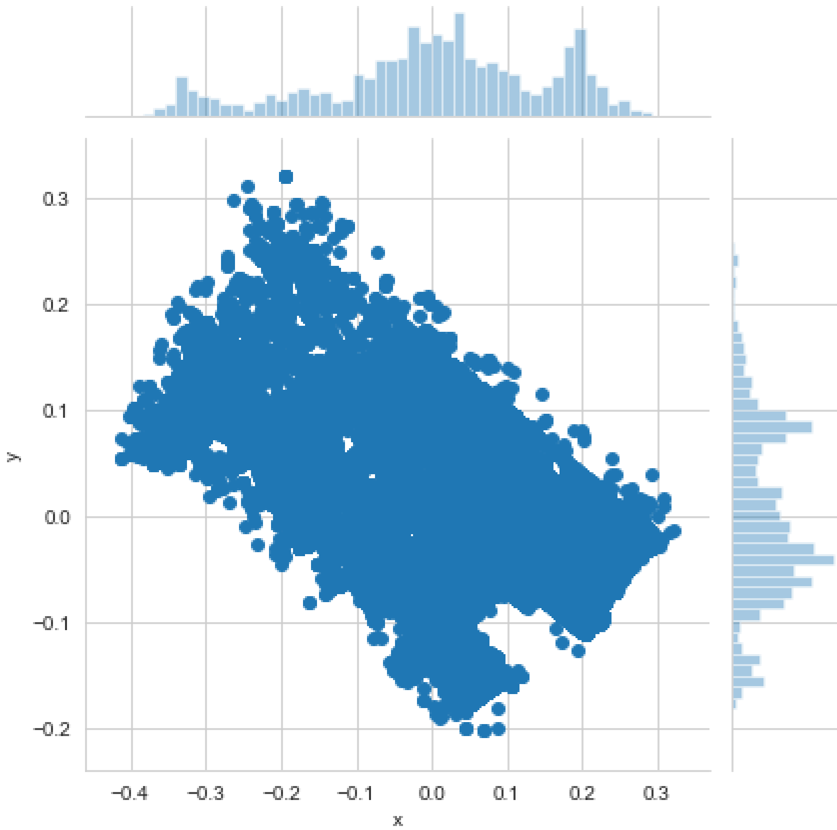
Out[41]: <seaborn.axisgrid.JointGrid at 0x11f707a2948>



Zooming out to the highest datapoints area by decreasing the scale.

In [42]:
```
#standardizing the column values of lat and long
pd.options.mode.chained_assignment = None #Remove Error Message
x_mean=df_geo['lng'].mean()
y_mean=df_geo['lat'].mean()
df_geo['x']=df_geo['lng'].map(lambda v:v-x_mean)
df_geo['y']=df_geo['lat'].map(lambda v:v-y_mean)
sns.jointplot(data=df_geo,x='x',y='y',kind='scatter')
```

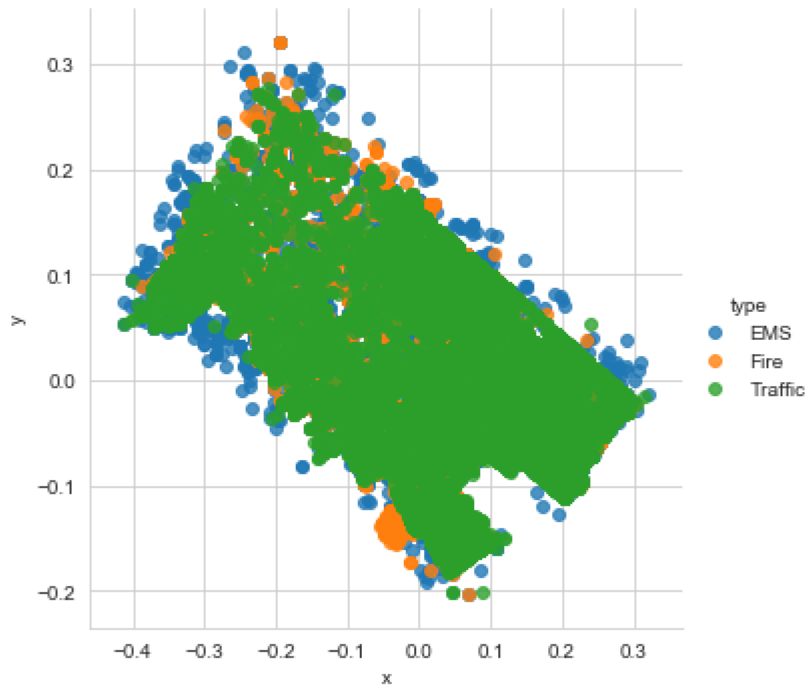
Out[42]: <seaborn.axisgrid.JointGrid at 0x11f70618208>



Standardising the data

In [44]:
```
sns.lmplot(x='x', y='y', hue='type', data=df_geo,fit_reg=False)
```

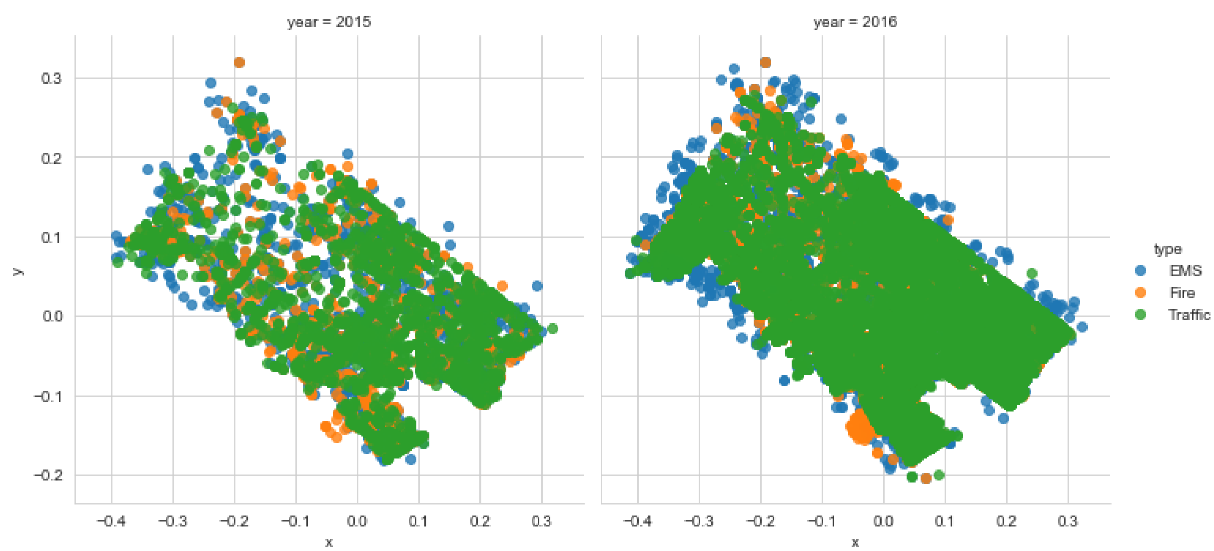
Out[44]: <seaborn.axisgrid.FacetGrid at 0x11f70354408>



Clasifing the categaries of data for the calls and where they originated from.

In [45]:
```
sns.lmplot(x='x', y='y', hue='type',col='year', data=df_geo,fit_reg=False)
```

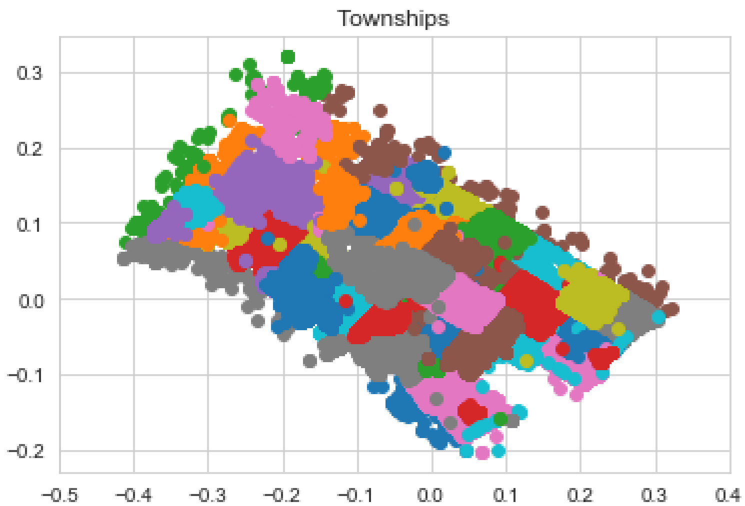
Out[45]: <seaborn.axisgrid.FacetGrid at 0x11f7031cd88>



Seeing the change of calls and location between the years of 2015 and 2016, we can see the number of calls hae increased drastically in the year 2016 compared to 2015, which is not a good trend and the city should increase their efforts if they want to stop this trend.

In [46]:
```
# Clustering lat-lng to map townships
group_town=df_geo.groupby('twp')
for name, group in group_town:
    plt.plot(group.x, group.y, marker='o', linestyle='', label=name)
plt.xlim(-0.5,0.4)
plt.title("Townships")
```

Out[46]:
```
Text(0.5, 1.0, 'Townships')
```



Classifing the number of calls with township and representing them.

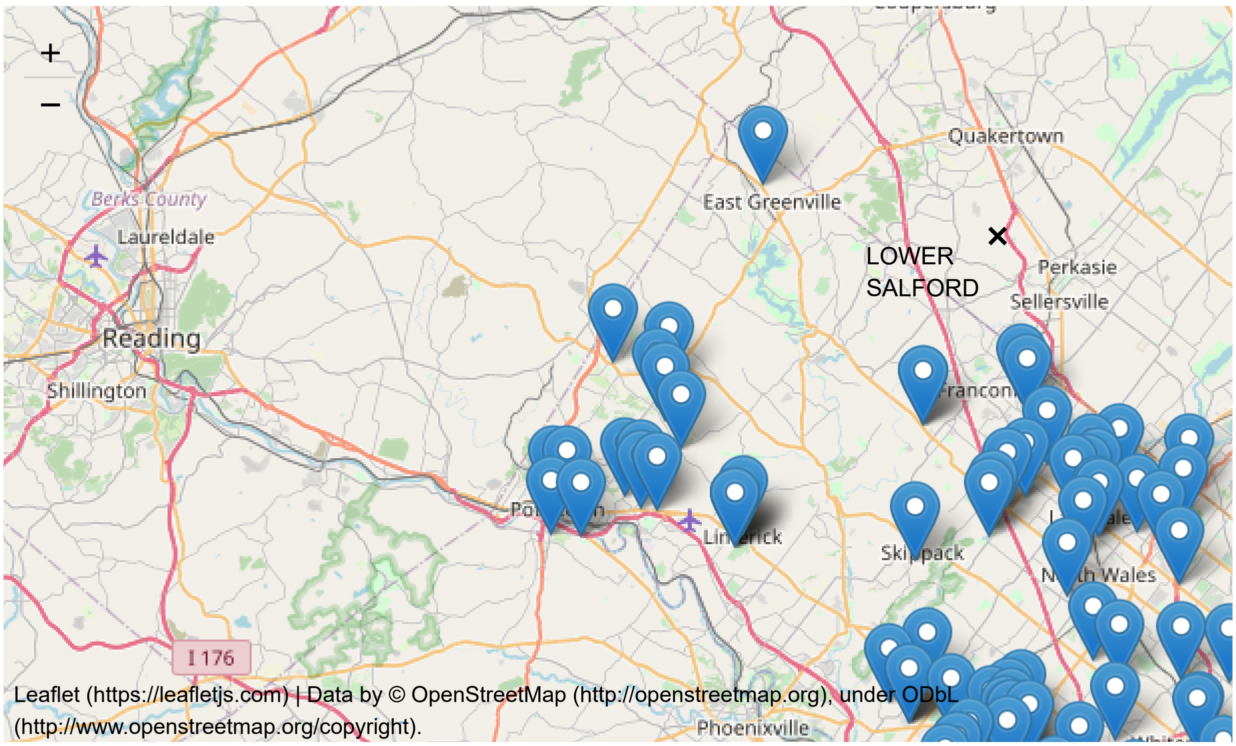
In [47]:
```
df['lat']=df['lat'].astype('float64')
df['lng']=df['lng'].astype('float64')
location = df['lat'].mean(), df['lng'].mean()

locationlist = df[['lat','lng']].values.tolist()
labels = df['twp'].values.tolist()

#Empty map
import folium
m = folium.Map(location=location, zoom_start=14)
#Accesing the latitude
for point in range(1,100):
    popup = folium.Popup(labels[point], parse_html=True)
    folium.Marker(locationlist[point], popup=popup).add_to(m)

m
```

Out[47]:



Representing the calls on an interactive map to help the authorities find the loctions quickly

Conclusion:

In this code, we have tried to narrow down the reason for people calling 911 and the time that they generally call 911. This will help the city to realise what they need to improve to reduce call.

They can also hire more staff around the days and times that tend to have more emergencies.

We have used folium as well to see a visual representation of the area where the calls are originated from.

We can see using our EDA that the calls have been increasing year to year and the most number of calls occur during the afternoon.

The highest number of calls generally tend to occur in on Saturdays in the Month of Jan while the highest calls are related to Emergency Medical Services.