911 Calls Data Capstone Project

January 16, 2020

This project uses the 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)

1 Data and Setup

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

[2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
%matplotlib inline
```

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

[4]:
```
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
```

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

```
[5]:         lat        lng                                               desc  \
     0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station …
     1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP…
     2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St…
     3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;…
     4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S…

            zip                    title            timeStamp                twp  \
     0  19525.0   EMS: BACK PAINS/INJURY  2015-12-10 17:40:00        NEW HANOVER
     1  19446.0  EMS: DIABETIC EMERGENCY  2015-12-10 17:40:00  HATFIELD TOWNSHIP
     2  19401.0      Fire: GAS-ODOR/LEAK  2015-12-10 17:40:00         NORRISTOWN
     3  19401.0   EMS: CARDIAC EMERGENCY  2015-12-10 17:40:01         NORRISTOWN
     4      NaN           EMS: DIZZINESS  2015-12-10 17:40:01   LOWER POTTSGROVE

                              addr  e
     0      REINDEER CT & DEAD END  1
     1  BRIAR PATH & WHITEMARSH LN  1
     2                    HAWS AVE  1
     3          AIRY ST & SWEDE ST  1
     4    CHERRYWOOD CT & DEAD END  1
```

2 Basic Questions

** What are the top 5 zipcodes for 911 calls? **

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

```
[6]: 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? **

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

```
[7]: LOWER MERION    8443
     ABINGTON        5977
     NORRISTOWN      5890
     UPPER MERION    5227
```

```
CHELTENHAM      4575
Name: twp, dtype: int64
```

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

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

```
[8]: 110
```

3 Creating new features

** In the titles column there are "Reasons/Departments" specified before the title code. These are EMS, Fire, and Traffic. Create a new column called "Reason" that contains this string value.**

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

```
[9]:          lat        lng                                               desc  \
     0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station …
     1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP…
     2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St…
     3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;…
     4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S…

            zip                    title            timeStamp                twp  \
     0  19525.0   EMS: BACK PAINS/INJURY  2015-12-10 17:40:00        NEW HANOVER
     1  19446.0  EMS: DIABETIC EMERGENCY  2015-12-10 17:40:00  HATFIELD TOWNSHIP
     2  19401.0      Fire: GAS-ODOR/LEAK  2015-12-10 17:40:00         NORRISTOWN
     3  19401.0   EMS: CARDIAC EMERGENCY  2015-12-10 17:40:01         NORRISTOWN
     4      NaN           EMS: DIZZINESS  2015-12-10 17:40:01   LOWER POTTSGROVE

                              addr  e Reason
     0      REINDEER CT & DEAD END  1    EMS
     1  BRIAR PATH & WHITEMARSH LN  1    EMS
     2                    HAWS AVE  1   Fire
     3          AIRY ST & SWEDE ST  1    EMS
     4    CHERRYWOOD CT & DEAD END  1    EMS
```

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

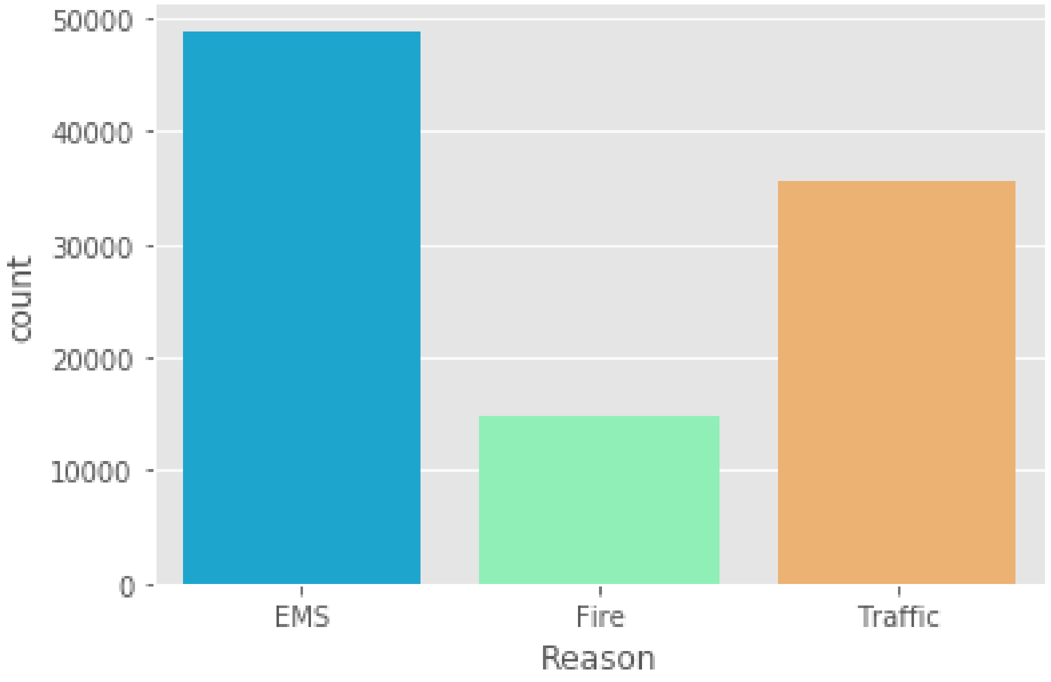
```
[10]: df['Reason'].value_counts().head()
```

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

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

[11]:
```
sns.countplot(df['Reason'], palette = 'rainbow')
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a23ec0110>
```



4 Time related analysis

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

[12]:
```
type(df['timeStamp'].iloc[0])
```

[12]:
```
str
```

** Convert the column from strings to DateTime objects. **

[13]:
```
df['timeStamp'] = pd.to_datetime(df['timeStamp'])
df['timeStamp'].iloc[0]
```

[13]:
```
Timestamp('2015-12-10 17:40:00')
```

[14]:
```
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.head()
```

```
[14]:          lat        lng                                                desc  \
      0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station …
      1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP…
      2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St…
      3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;…
      4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S…

             zip                    title           timeStamp                twp  \
      0  19525.0   EMS: BACK PAINS/INJURY 2015-12-10 17:40:00        NEW HANOVER
      1  19446.0  EMS: DIABETIC EMERGENCY 2015-12-10 17:40:00  HATFIELD TOWNSHIP
      2  19401.0      Fire: GAS-ODOR/LEAK 2015-12-10 17:40:00         NORRISTOWN
      3  19401.0   EMS: CARDIAC EMERGENCY 2015-12-10 17:40:01         NORRISTOWN
      4      NaN           EMS: DIZZINESS 2015-12-10 17:40:01   LOWER POTTSGROVE

                              addr  e Reason  Hour  Month  Day of Week
      0      REINDEER CT & DEAD END  1    EMS    17     12            3
      1  BRIAR PATH & WHITEMARSH LN  1    EMS    17     12            3
      2                    HAWS AVE  1   Fire    17     12            3
      3          AIRY ST & SWEDE ST  1    EMS    17     12            3
      4    CHERRYWOOD CT & DEAD END  1    EMS    17     12            3
```

** 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'}

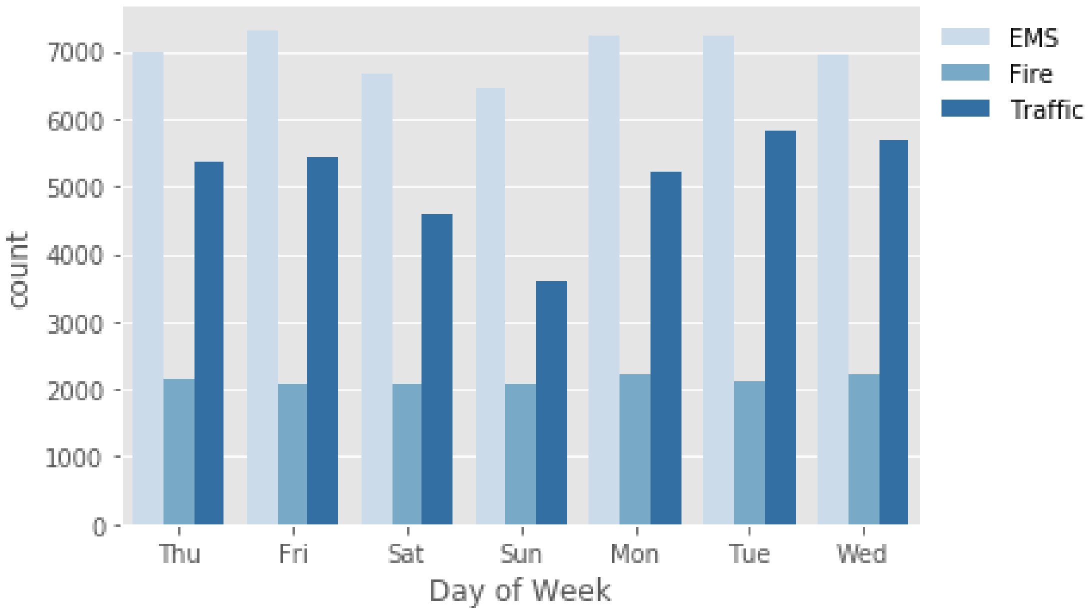
```
[15]: dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
      df['Day of Week'] = df['Day of Week'].map(dmap)
      df['Day of Week'].head()
```

```
[15]: 0    Thu
      1    Thu
      2    Thu
      3    Thu
      4    Thu
      Name: Day of Week, dtype: object
```

** Create a countplot of the Day of Week column with the hue based off of the Reason column. **

```
[16]: sns.countplot(df['Day of Week'], hue=df['Reason'], palette='Blues')
      plt.legend(loc='upper left', bbox_to_anchor=(1,1), frameon=False)
```

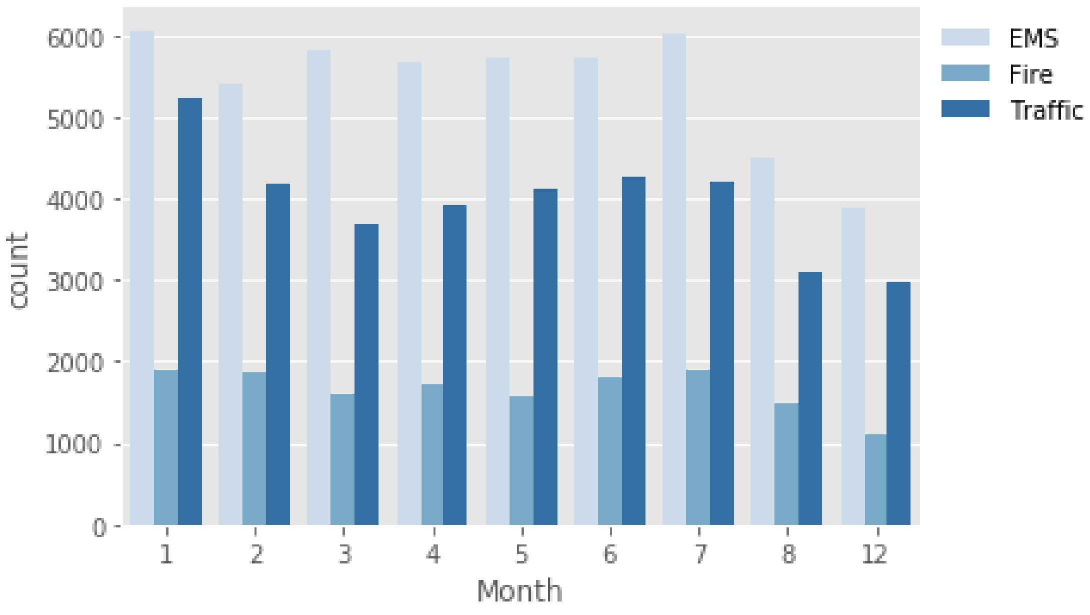
```
[16]: <matplotlib.legend.Legend at 0x1a23e932d0>
```



** Now do the same for Month:**

```
[17]: sns.countplot(df['Month'], hue=df['Reason'], palette='Blues')
      plt.legend(loc='upper left', bbox_to_anchor=(1,1), frameon=False)
```

```
[17]: <matplotlib.legend.Legend at 0x1a265bf4d0>
```



** Noticed there were some missing Months. Will plot the information in another way. **

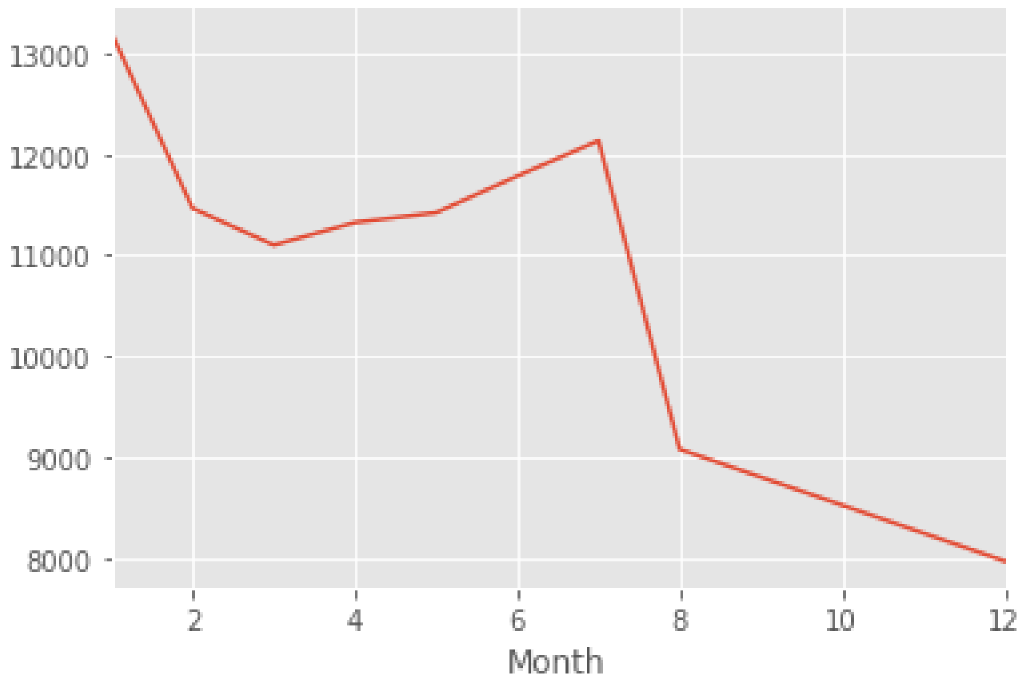
```
[18]: byMonth = df.groupby('Month').count()
      byMonth.head()
```

```
[18]:          lat    lng   desc    zip  title  timeStamp    twp   addr      e  \
      Month
      1      13205  13205  13205  11527  13205      13205  13203  13096  13205
      2      11467  11467  11467   9930  11467      11467  11465  11396  11467
      3      11101  11101  11101   9755  11101      11101  11092  11059  11101
      4      11326  11326  11326   9895  11326      11326  11323  11283  11326
      5      11423  11423  11423   9946  11423      11423  11420  11378  11423

             Reason   Hour  Day of Week
      Month
      1       13205  13205        13205
      2       11467  11467        11467
      3       11101  11101        11101
      4       11326  11326        11326
      5       11423  11423        11423
```

** Create a simple plot off of the dataframe indicating the count of calls per month. **

```
[19]: byMonth['title'].plot.line()
```

```
[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1a25115650>
```

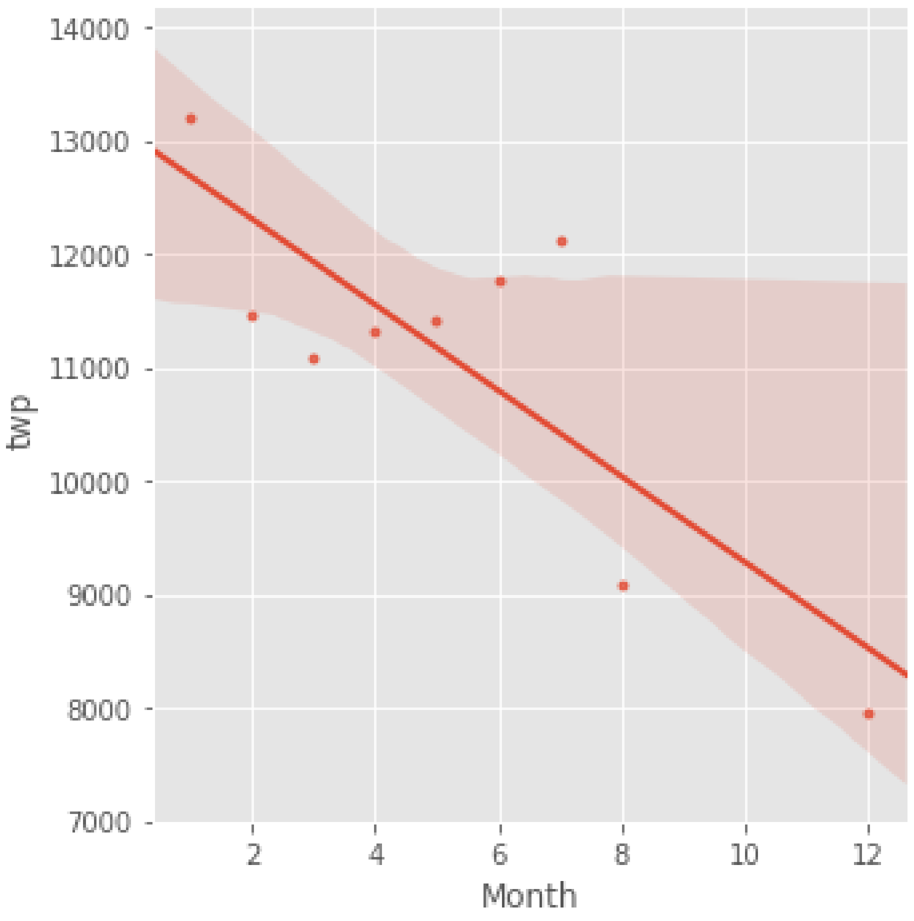


** Create a linear fit on the number of calls per month. **

```
[20]: byMonth.reset_index(inplace=True)
```

```
[21]: sns.lmplot(x='Month', y='twp', data=byMonth, markers='.')
```

```
[21]: <seaborn.axisgrid.FacetGrid at 0x1a2515a990>
```



** Create a new column called 'Date'. **

```
[22]: df['Date'] = df['timeStamp'].apply(lambda time: time.date())
      df.head()
```

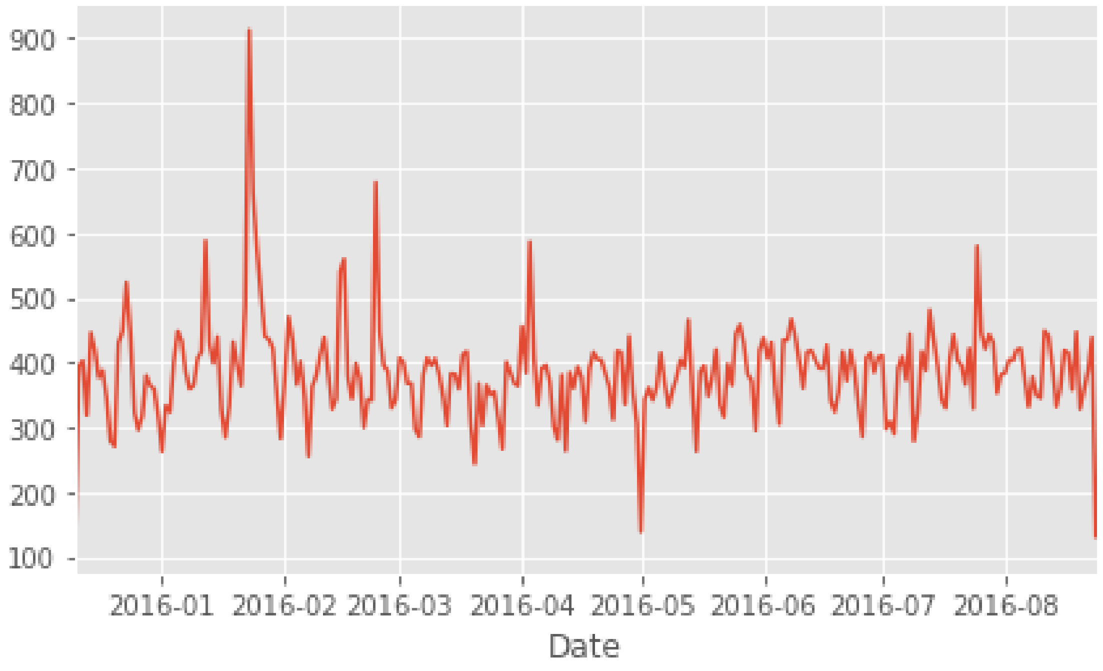
```
[22]:         lat        lng                                               desc  \
      0  40.297876 -75.581294  REINDEER CT & DEAD END;  NEW HANOVER; Station …
      1  40.258061 -75.264680  BRIAR PATH & WHITEMARSH LN;  HATFIELD TOWNSHIP…
      2  40.121182 -75.351975  HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St…
```

```
3  40.116153 -75.343513  AIRY ST & SWEDE ST;  NORRISTOWN; Station 308A;…
4  40.251492 -75.603350  CHERRYWOOD CT & DEAD END;  LOWER POTTSGROVE; S…

       zip                    title           timeStamp                twp  \
0  19525.0   EMS: BACK PAINS/INJURY 2015-12-10 17:40:00        NEW HANOVER
1  19446.0  EMS: DIABETIC EMERGENCY 2015-12-10 17:40:00  HATFIELD TOWNSHIP
2  19401.0      Fire: GAS-ODOR/LEAK 2015-12-10 17:40:00         NORRISTOWN
3  19401.0   EMS: CARDIAC EMERGENCY 2015-12-10 17:40:01         NORRISTOWN
4      NaN           EMS: DIZZINESS 2015-12-10 17:40:01   LOWER POTTSGROVE

                        addr  e Reason  Hour  Month Day of Week        Date
0      REINDEER CT & DEAD END  1    EMS    17     12         Thu  2015-12-10
1  BRIAR PATH & WHITEMARSH LN  1    EMS    17     12         Thu  2015-12-10
2                    HAWS AVE  1   Fire    17     12         Thu  2015-12-10
3          AIRY ST & SWEDE ST  1    EMS    17     12         Thu  2015-12-10
4    CHERRYWOOD CT & DEAD END  1    EMS    17     12         Thu  2015-12-10
```

** Groupby this Date column and create a plot of counts of 911 calls.**

[23]:
```
byDate = df.groupby('Date').count()
byDate['twp'].plot.line(figsize=(7,4))
```

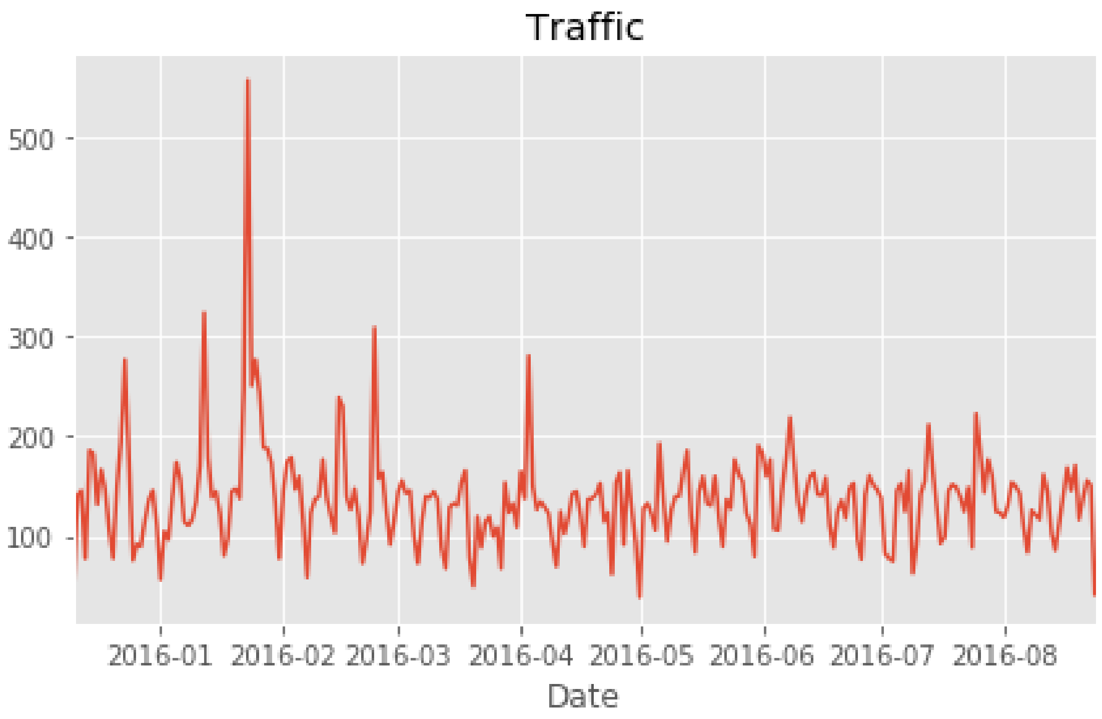
[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x110ff52d0>
```



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

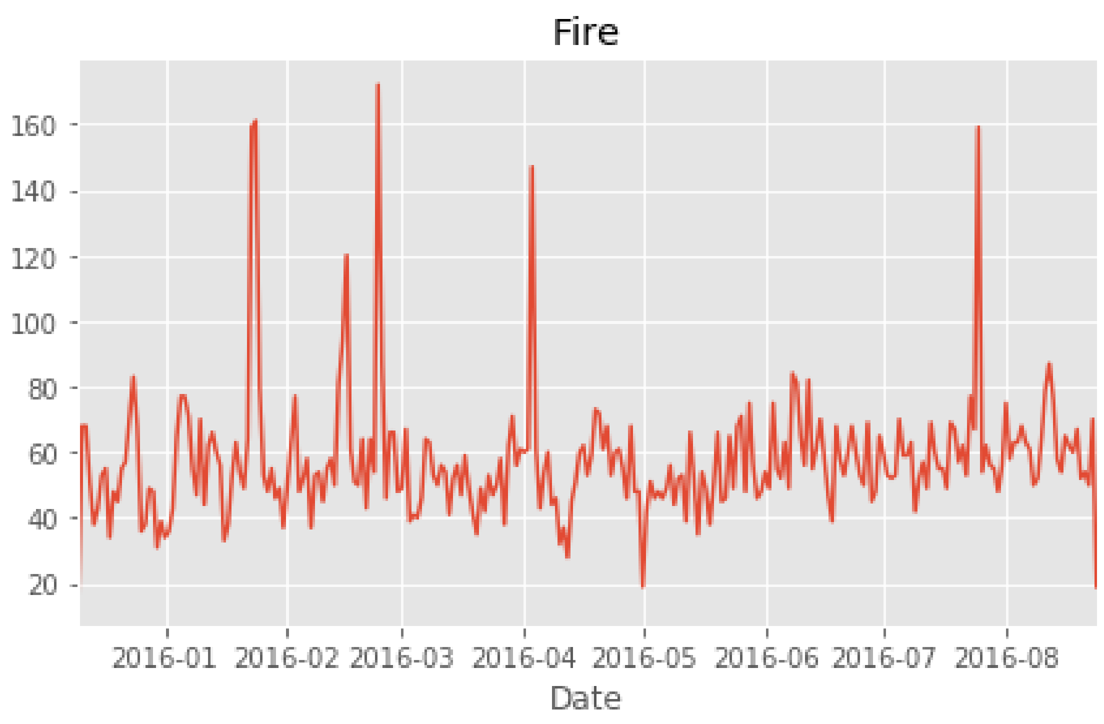
[24]:
```
byDate_T = df[df['Reason'] == 'Traffic'].groupby('Date').count()
byDate_T['twp'].plot.line(title='Traffic', figsize=(7,4))
```

[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a28b5c990>
```



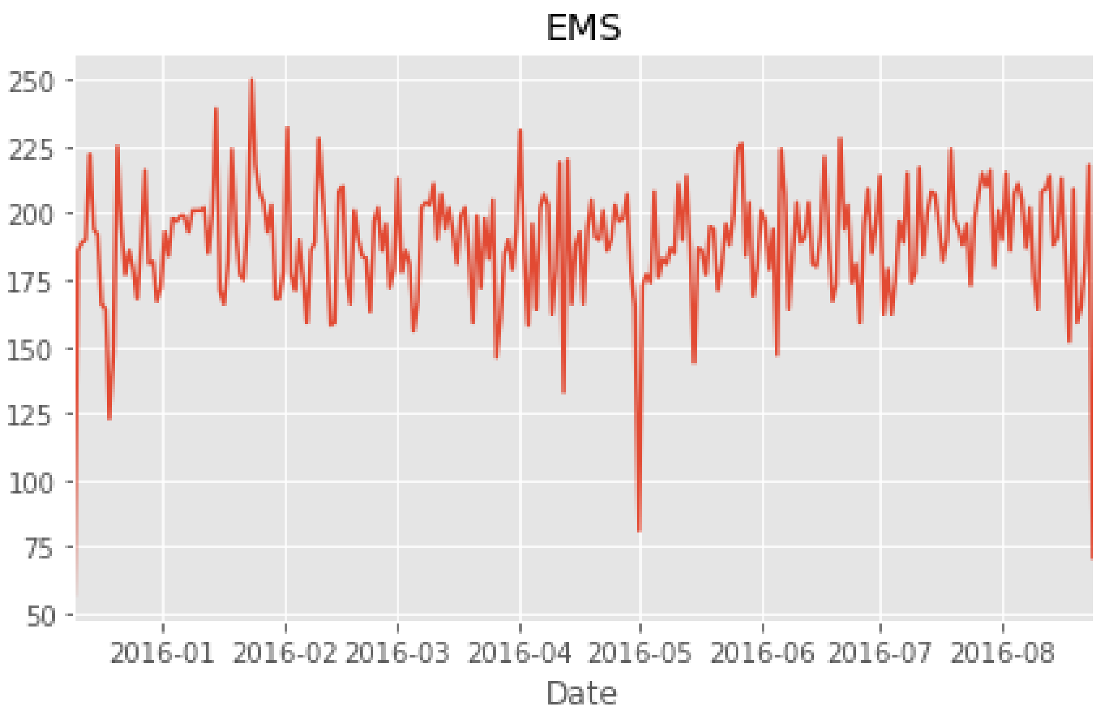
[25]:
```
byDate_F = df[df['Reason'] == 'Fire'].groupby('Date').count()
byDate_F['twp'].plot.line(title='Fire', figsize=(7,4))
```

[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2627e6d0>
```



```
[26]: byDate_E = df[df['Reason'] == 'EMS'].groupby('Date').count()
      byDate_E['twp'].plot.line(title='EMS', figsize=(7,4))
```

```
[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1a26e083d0>
```



5 Heatmap & Clustermap

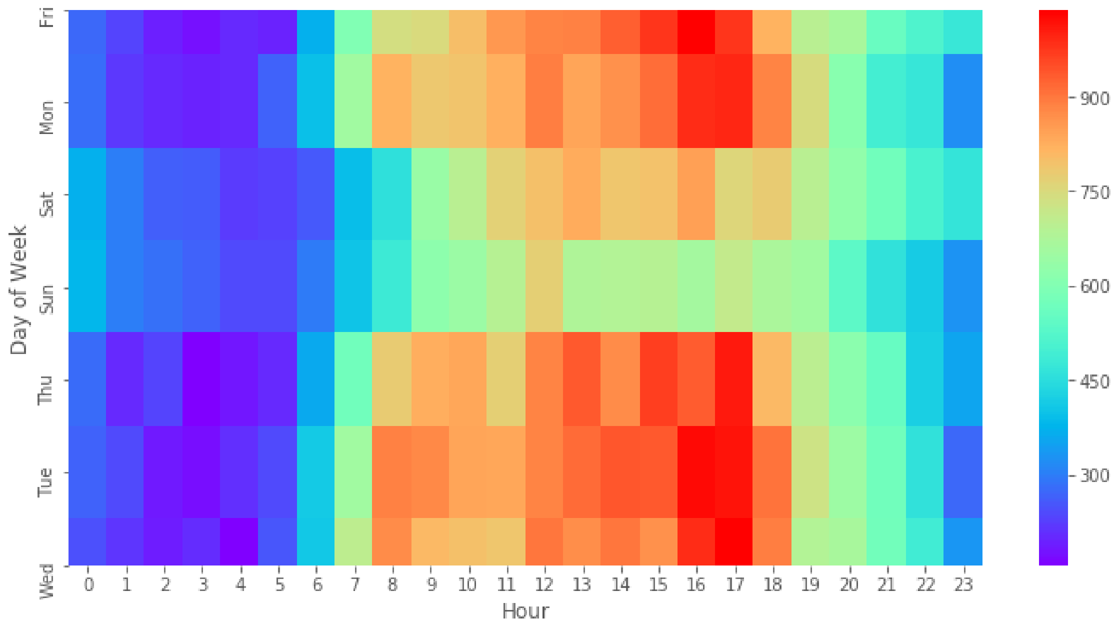
** Create heatmaps. First need to restructure the dataframe so that the columns become the Hours and the Index becomes the Day of the Week. **

[28]:
```
pivot = df.groupby(['Day of Week','Hour']).count()
pivot = pivot['twp'].unstack()
```

** Create a HeatMap using this new DataFrame. **

[29]:
```
plt.figure(figsize=(12,6))
sns.heatmap(pivot, cmap='rainbow')
```

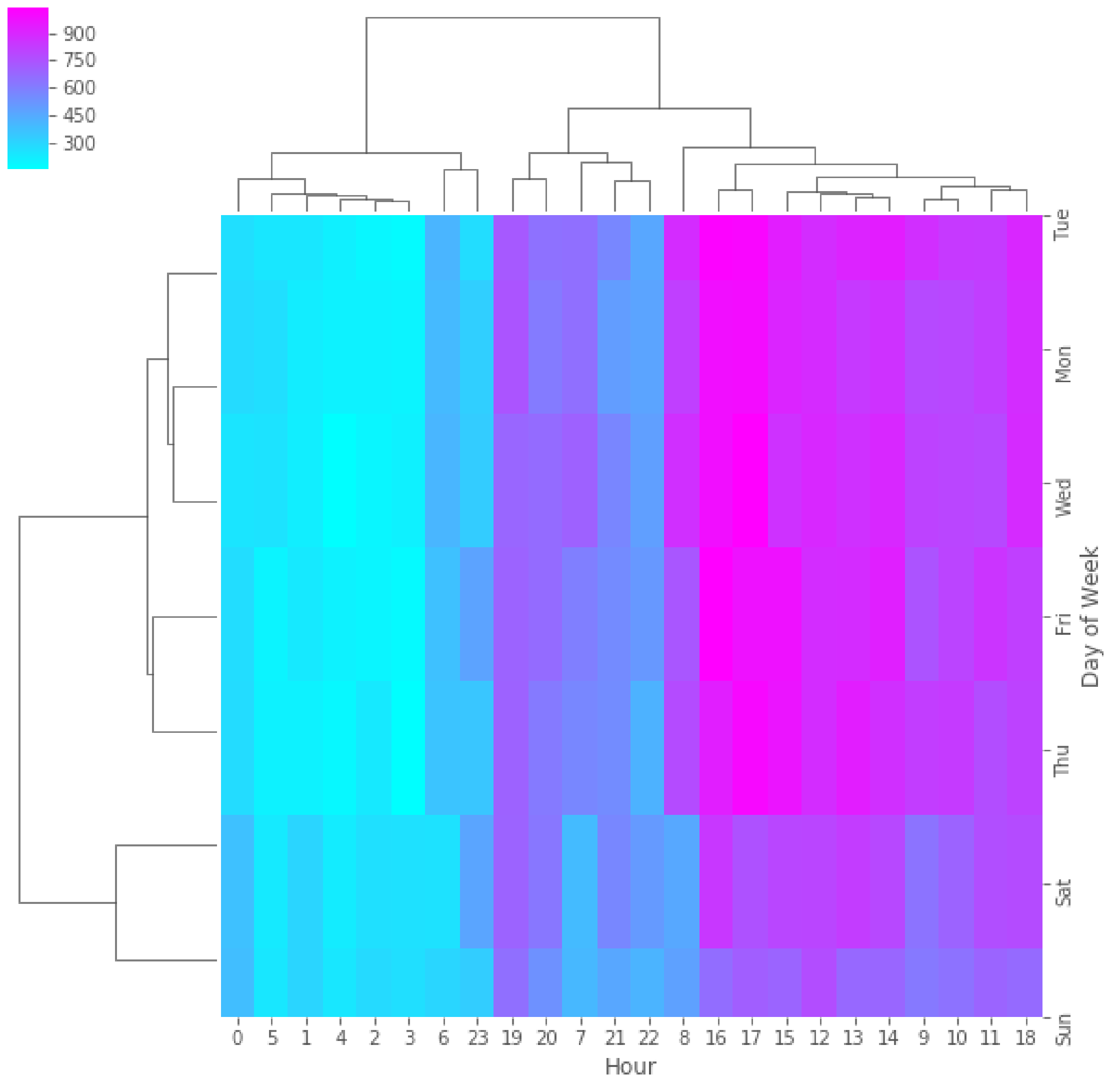
[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26e02dd0>
```



** Create a clustermap using this DataFrame. **

```
[30]: plt.figure(figsize=(10,6))
      sns.clustermap(pivot, cmap='cool')
```

```
[30]: <seaborn.matrix.ClusterGrid at 0x1a2662fc10>

      <Figure size 720x432 with 0 Axes>
```



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

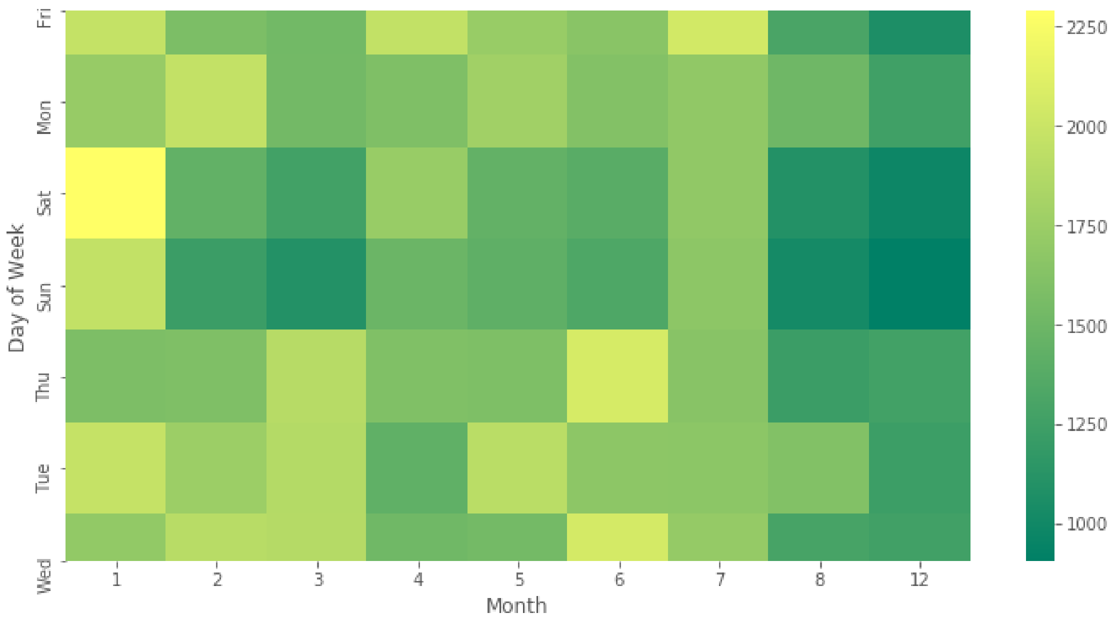
```
[31]: pivot3 = df.groupby(['Day of Week','Month']).count()
      pivot3 = pivot3['twp'].unstack()
      pivot3
```

```
[31]: Month           1     2     3     4     5     6     7     8     12
      Day of Week
      Fri          1970  1581  1523  1958  1730  1649  2045  1310  1064
      Mon          1727  1964  1533  1597  1779  1617  1692  1509  1256
      Sat          2290  1440  1264  1732  1444  1388  1695  1099   978
      Sun          1960  1229  1100  1488  1422  1331  1672  1021   907
      Thu          1584  1596  1900  1601  1590  2065  1646  1227  1265
```

```
Tue           1973  1753  1884  1430  1917  1673  1668  1612  1233
Wed           1699  1902  1888  1517  1538  2054  1715  1295  1260
```

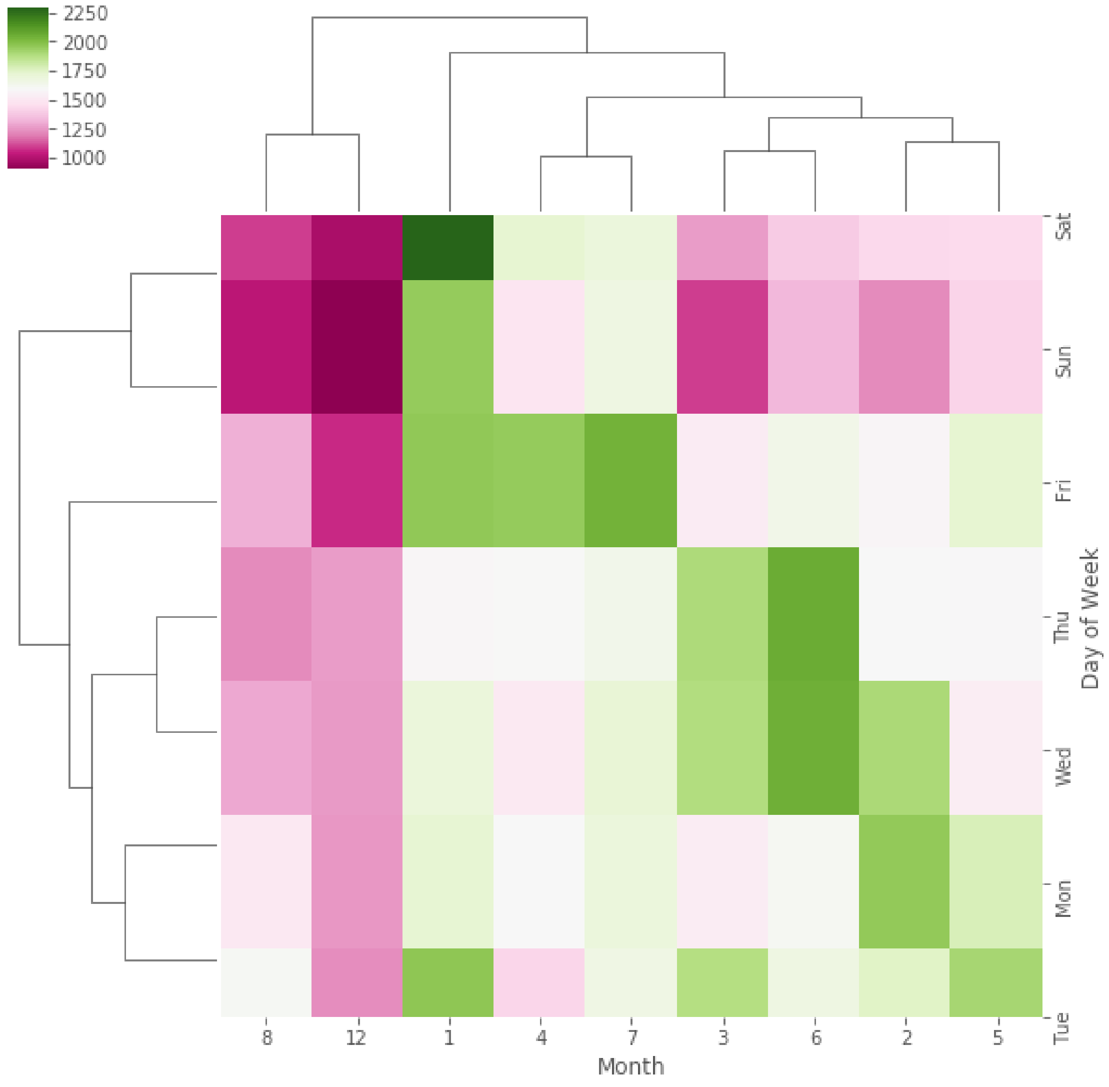
[32]:
```
plt.figure(figsize=(12,6))
sns.heatmap(pivot3, cmap='summer')
```

[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a264422d0>
```



[33]:
```
sns.clustermap(pivot3, cmap='PiYG')
```

[33]:
```
<seaborn.matrix.ClusterGrid at 0x1a26b92c10>
```



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