

# 911 Calls Data Capstone Project

March 23, 2019

## 1 A detail descriptive Analyses of 911 Calls Project in Python - A capstone Project - Phase 1

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)

Tools Used: Jupyter Notebook

## 2 Initial Setup of Libraries such as numpy, Pandas, Matplotlib and Seaborn

---

```
** 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
        %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 variable and see the column types and number of values associated with it \*\***

```
In [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
```

**\*\* Check the head of df \*\***

```
In [5]: df.head(50)
```

```
Out [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 POTTS GROVE; S...
5	40.253473	-75.283245	CANNON AVE & W 9TH ST; LANSDALE; Station 345;...
6	40.182111	-75.127795	LAUREL AVE & OAKDALE AVE; HORSHAM; Station 35...
7	40.217286	-75.405182	COLLEGEVILLE RD & LYWISKI RD; SKIPPACK; Stati...
8	40.289027	-75.399590	MAIN ST & OLD SUMNEYTOWN PIKE; LOWER SALFORD;...
9	40.102398	-75.291458	BLUEROUTE & RAMP I476 NB TO CHEMICAL RD; PLYM...
10	40.231990	-75.251891	RT202 PKWY & KNAPP RD; MONTGOMERY; 2015-12-10 ...
11	40.084161	-75.308386	BROOK RD & COLWELL LN; PLYMOUTH; 2015-12-10 @ ...
12	40.174131	-75.098491	BYBERRY AVE & S WARMINSTER RD; UPPER MORELAND;...
13	40.062974	-75.135914	OLD YORK RD & VALLEY RD; CHELTENHAM; 2015-12-1...
14	40.097222	-75.376195	SCHUYLKILL EXPY & CROTON RD UNDERPASS; UPPER M...
15	40.223778	-75.235399	STUMP RD & WITCHWOOD DR; MONTGOMERY; 2015-12-1...
16	40.243258	-75.286552	SUSQUEHANNA AVE & W MAIN ST; LANSDALE; Statio...
17	40.312181	-75.574260	CHARLOTTE ST & MILES RD; NEW HANOVER; Station...
18	40.114239	-75.338508	PENN ST & ARCH ST; NORRISTOWN; Station 308A; ...
19	40.209337	-75.135266	COUNTY LINE RD & WILLOW DR; HORSHAM; 2015-12-1...
20	40.114239	-75.338508	PENN ST & ARCH ST; NORRISTOWN; 2015-12-10 @ 17...
21	40.117948	-75.209848	CHURCH RD & REDCOAT DR; WHITEMARSH; 2015-12-10...
22	40.199006	-75.300058	LILAC CT & PRIMROSE DR; UPPER GWYNEDD; 2015-12...

23 40.143326 -75.422819 RT422 & PAWLINGS RD OVERPASS; LOWER PROVIDENC...  
 24 40.153268 -75.189558 SUMMIT AVE & RT309 UNDERPASS; UPPER DUBLIN; 20...  
 25 40.133037 -75.408463 SHANNONDELL DR & SHANNONDELL BLVD; LOWER PROV...  
 26 40.155283 -75.264230 PENLLYN BLUE BELL PIKE & VILLAGE CIR; WHITPAI...  
 27 40.028903 -75.351822 EDENTON PL & DURHAM DR; DELAWARE COUNTY; 2015-...  
 28 40.097222 -75.376195 SCHUYLKILL EXPY & WEADLEY RD OVERPASS; UPPER M...  
 29 40.209337 -75.135266 COUNTY LINE RD & WILLOW DR; HORSHAM; 2015-12-1...  
 30 40.097222 -75.376195 SCHUYLKILL EXPY & WEADLEY RD OVERPASS; UPPER M...  
 31 40.300736 -75.331973 CORNWALL TER & LIONS GATE CIR; FRANCONIA; Sta...  
 32 40.129398 -75.332213 PINE ST & W ROBERTS ST; NORRISTOWN; Station 3...  
 33 40.297876 -75.581294 ; NEW HANOVER; 2015-12-10 @ 18:20:28;  
 34 40.081260 -75.137025 ; CHELTENHAM; 2015-12-10 @ 18:20:28;  
 35 40.099362 -75.150035 E GLENSIDE AVE & S KESWICK AVE; CHELTENHAM; 20...  
 36 40.221227 -75.288737 MORRIS RD & MUHLENBURG DR; UPPER GWYNEDD; 2015...  
 37 40.073864 -75.316797 MOOREHEAD AVE & FRONT ST; WEST CONSHOHOCKEN; 2...  
 38 40.161732 -75.151055 DRESHER RD & WELSH RD; HORSHAM; 2015-12-10 @ 1...  
 39 40.066718 -75.307176 CONSHOHOCKEN STATE RD; WEST CONSHOHOCKEN; Sta...  
 40 40.065530 -75.307828 CONSHOHOCKEN STATE RD & MERION HILL LN; WEST C...  
 41 40.104206 -75.367665 HAMPTON RD & BELMONT RD; UPPER MERION; 2015-12...  
 42 40.122780 -75.267241 SPARANGO LN & MELISSA WAY; PLYMOUTH; Station ...  
 43 40.091055 -75.384365 ALLENDALE RD & WILLS BLVD; UPPER MERION; 2015-...  
 44 40.024967 -75.282905 ROSEMONT AVE & DEAD END; LOWER MERION; Statio...  
 45 40.125739 -75.339822 W WOOD ST & MARKLEY ST; NORRISTOWN; 2015-12-10...  
 46 40.224923 -75.528045 LINFIELD TRAPPE RD; LIMERICK; Station 324A; 2...  
 47 40.224923 -75.528045 LINFIELD TRAPPE RD; LIMERICK; 2015-12-10 @ 18:...  
 48 40.224923 -75.528045 AUTO PARK BLVD & LINFIELD TRAPPE RD; LIMERICK;...  
 49 40.230934 -75.522125 LINFIELD TRAPPE RD & RAMP N LEWIS RD TO RT422 ...

	zip	title	timeStamp \
0	19525.0	EMS: BACK PAINS/INJURY	2015-12-10 17:40:00
1	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:40:00
2	19401.0	Fire: GAS-ODOR/LEAK	2015-12-10 17:40:00
3	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 17:40:01
4	NaN	EMS: DIZZINESS	2015-12-10 17:40:01
5	19446.0	EMS: HEAD INJURY	2015-12-10 17:40:01
6	19044.0	EMS: NAUSEA/VOMITING	2015-12-10 17:40:01
7	19426.0	EMS: RESPIRATORY EMERGENCY	2015-12-10 17:40:01
8	19438.0	EMS: SYNCOPAL EPISODE	2015-12-10 17:40:01
9	19462.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:01
10	NaN	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:01
11	19428.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:02
12	19040.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:02
13	19027.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:02
14	NaN	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:02
15	18936.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:02
16	19446.0	EMS: RESPIRATORY EMERGENCY	2015-12-10 17:46:01
17	19525.0	EMS: DIZZINESS	2015-12-10 17:47:01
18	19401.0	EMS: VEHICLE ACCIDENT	2015-12-10 17:47:01

19	18974.0	Traffic: DISABLED VEHICLE -	2015-12-10 17:47:02
20	19401.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:47:02
21	19031.0	Traffic: DISABLED VEHICLE -	2015-12-10 17:57:02
22	19446.0	Fire: APPLIANCE FIRE	2015-12-10 18:02:01
23	NaN	Traffic: DISABLED VEHICLE -	2015-12-10 18:02:02
24	NaN	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:02:02
25	19403.0	EMS: GENERAL WEAKNESS	2015-12-10 18:06:25
26	19422.0	EMS: HEAD INJURY	2015-12-10 18:06:25
27	19085.0	Fire: CARBON MONOXIDE DETECTOR	2015-12-10 18:06:25
28	NaN	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:06:26
29	18974.0	Traffic: DISABLED VEHICLE -	2015-12-10 18:11:01
30	NaN	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:11:01
31	18964.0	EMS: RESPIRATORY EMERGENCY	2015-12-10 18:12:01
32	19401.0	EMS: UNKNOWN MEDICAL EMERGENCY	2015-12-10 18:22:00
33	19525.0	Traffic: DISABLED VEHICLE -	2015-12-10 18:22:01
34	NaN	Traffic: DISABLED VEHICLE -	2015-12-10 18:26:02
35	19038.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:26:02
36	NaN	Traffic: DISABLED VEHICLE -	2015-12-10 18:27:01
37	19428.0	Traffic: DISABLED VEHICLE -	2015-12-10 18:27:02
38	19044.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:27:02
39	NaN	EMS: VEHICLE ACCIDENT	2015-12-10 18:32:02
40	19428.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:32:02
41	19406.0	Fire: GAS-ODOR/LEAK	2015-12-10 18:37:01
42	19462.0	EMS: UNRESPONSIVE SUBJECT	2015-12-10 18:42:00
43	19406.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:42:01
44	NaN	EMS: CARDIAC EMERGENCY	2015-12-10 18:47:01
45	19401.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:51:01
46	19468.0	EMS: VEHICLE ACCIDENT	2015-12-10 18:52:00
47	19468.0	Fire: VEHICLE ACCIDENT	2015-12-10 18:52:00
48	19468.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:56:02
49	19468.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 18:56:02

	twp	addr	e
0	NEW HANOVER	REINDEER CT & DEAD END	1
1	HATFIELD TOWNSHIP	BRIAR PATH & WHITEMARSH LN	1
2	NORRISTOWN	HAWS AVE	1
3	NORRISTOWN	AIRY ST & SWEDE ST	1
4	LOWER POTTS GROVE	CHERRYWOOD CT & DEAD END	1
5	LANS DALE	CANNON AVE & W 9TH ST	1
6	HORSHAM	LAUREL AVE & OAKDALE AVE	1
7	SKIPPAK	COLLEGEVILLE RD & LYWISKI RD	1
8	LOWER SALFORD	MAIN ST & OLD SUMNEYTOWN PIKE	1
9	PLYMOUTH	BLUEROUTE & RAMP I476 NB TO CHEMICAL RD	1
10	MONTGOMERY	RT202 PKWY & KNAPP RD	1
11	PLYMOUTH	BROOK RD & COLWELL LN	1
12	UPPER MORELAND	BYBERRY AVE & S WARMINSTER RD	1
13	CHEL TENHAM	OLD YORK RD & VALLEY RD	1
14	UPPER MERION	SCHUYLKILL EXPY & CROTON RD UNDERPASS	1

15	MONTGOMERY	STUMP RD & WITCHWOOD DR	1
16	LANSDALE	SUSQUEHANNA AVE & W MAIN ST	1
17	NEW HANOVER	CHARLOTTE ST & MILES RD	1
18	NORRISTOWN	PENN ST & ARCH ST	1
19	HORSHAM	COUNTY LINE RD & WILLOW DR	1
20	NORRISTOWN	PENN ST & ARCH ST	1
21	WHITEMARSH	CHURCH RD & REDCOAT DR	1
22	UPPER GWYNEDD	LILAC CT & PRIMROSE DR	1
23	LOWER PROVIDENCE	RT422 & PAWLINGS RD OVERPASS	1
24	UPPER DUBLIN	SUMMIT AVE & RT309 UNDERPASS	1
25	LOWER PROVIDENCE	SHANNONDELL DR & SHANNONDELL BLVD	1
26	WHITPAIN	PENLLYN BLUE BELL PIKE & VILLAGE CIR	1
27	DELAWARE COUNTY	EDENTON PL & DURHAM DR	1
28	UPPER MERION	SCHUYLKILL EXPY & WEADLEY RD OVERPASS	1
29	HORSHAM	COUNTY LINE RD & WILLOW DR	1
30	UPPER MERION	SCHUYLKILL EXPY & WEADLEY RD OVERPASS	1
31	FRANCONIA	CORNWALL TER & LIONS GATE CIR	1
32	NORRISTOWN	PINE ST & W ROBERTS ST	1
33	NEW HANOVER		NaN
34	CHELTENHAM		NaN
35	CHELTENHAM	E GLENSIDE AVE & S KESWICK AVE	1
36	UPPER GWYNEDD	MORRIS RD & MUHLENBURG DR	1
37	WEST CONSHOHOCKEN	MOOREHEAD AVE & FRONT ST	1
38	HORSHAM	DRESHER RD & WELSH RD	1
39	WEST CONSHOHOCKEN	CONSHOHOCKEN STATE RD	1
40	WEST CONSHOHOCKEN	CONSHOHOCKEN STATE RD & MERION HILL LN	1
41	UPPER MERION	HAMPTON RD & BELMONT RD	1
42	PLYMOUTH	SPARANGO LN & MELISSA WAY	1
43	UPPER MERION	ALLENDALE RD & WILLS BLVD	1
44	LOWER MERION	ROSEMONT AVE & DEAD END	1
45	NORRISTOWN	W WOOD ST & MARKLEY ST	1
46	LIMERICK	LINFIELD TRAPPE RD	1
47	LIMERICK	LINFIELD TRAPPE RD	1
48	LIMERICK	AUTO PARK BLVD & LINFIELD TRAPPE RD	1
49	LIMERICK	LINFIELD TRAPPE RD & RAMP N LEWIS RD TO RT422 EB	1

## 2.1 Handling Basic Questions of descriptive analyses

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

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

```
Out[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? \*\***

```
In [7]: df['twp'].value_counts().head(5)
```

```
Out[7]: LOWER MERION      8443
        ABINGTON         5977
        NORRISTOWN       5890
        UPPER MERION     5227
        CHELTENHAM       4575
        Name: twp, dtype: int64
```

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

```
In [33]: c_reason=df['title'].nunique()
        c_reason
```

```
Out[33]: 110
```

## 2.2 Now, lets expand the hidden information by Creating new features

**\*\* In the titles column there are “Reasons/Departments” specified before the title code. These are EMS, Fire, and Traffic. We will use apply() function and use lambda expression to short our function and create a new feature named ‘Reasons’ for which a call was made to 911.\*\***

**For example, if the title column value is EMS: BACK PAINS/INJURY , the Reason column value would be EMS.**

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

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

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

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

**\*\* Now we will use seaborn, plotly to create a countplot of 911 calls by Reason. \*\***

```
In [39]: #plotly
        from plotly import __version__

        #cluflinks
        import cufflinks as cf

        #importing Iplot and other important libraries in plotly
        from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

        init_notebook_mode(connected=True)
        cf.go_offline()
        c_reason = df['Reason'].nunique()
        c_reason
```

IOPub data rate exceeded.  
The notebook server will temporarily stop sending output  
to the client in order to avoid crashing it.  
To change this limit, set the config variable  
`--NotebookApp.iopub\_data\_rate\_limit`.

Out[39]: 3

---

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

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

Out[12]: str

```
In [40]: ## Data Wrangling
```

**\*\* You should have seen that these timestamps are still strings. We will use pd.to\_datetime() to convert the column from strings to DateTime objects. We will do some data wrangling here \*\***

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

Out[13]: 17

**\*\* You can now grab specific attributes from a Datetime object by calling them and extracting Hour, Month and Day of week from the timestamp; We will use apply() to create 3 new columns called Hour, Month, and Day of Week. \*\***

```
In [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)
```

**\*\* Notice how the Day of Week is an integer 0-6. We will 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'}
```

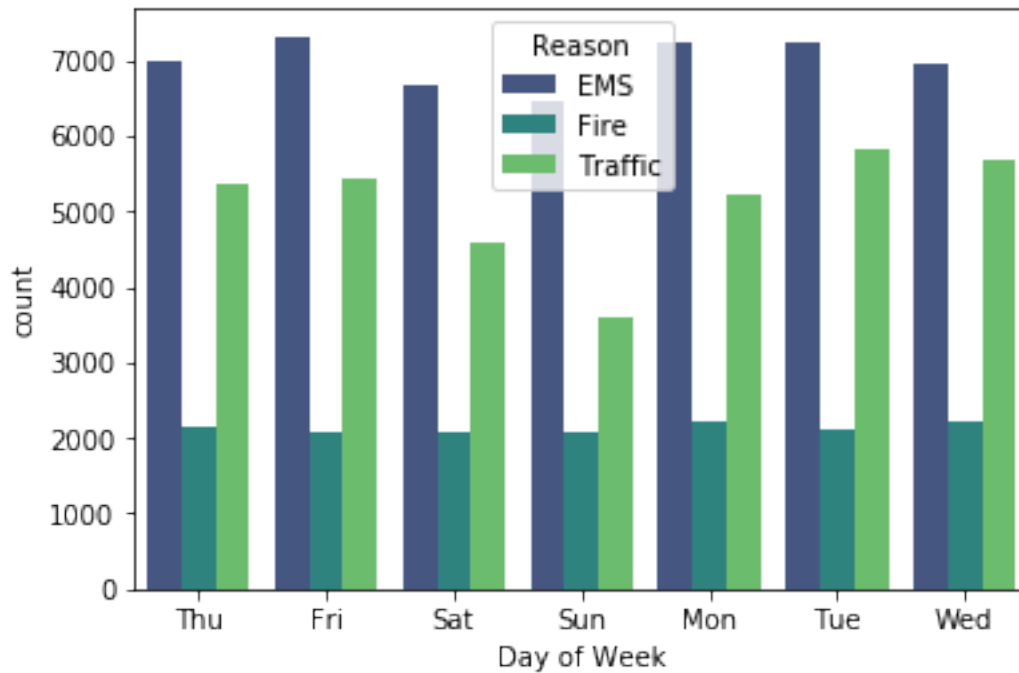
```
In [15]: dmap = {0: 'Mon', 1: 'Tue', 2: 'Wed', 3: 'Thu', 4: 'Fri', 5: 'Sat', 6: 'Sun'}
```

```
In [16]: df['Day of Week'] = df['Day of Week'].map(dmap)
```

**\*\* Now we will use seaborn to create a countplot of the Day of Week column with the hue based off of the Reason column. Here, we can see that EMS is the biggest reasons for most of the 911 calls, followed by Traffic and Fire\*\***

```
In [17]: sns.countplot(x='Day of Week',data=df,hue='Reason',palette='viridis')
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x2a907bb1048>
```

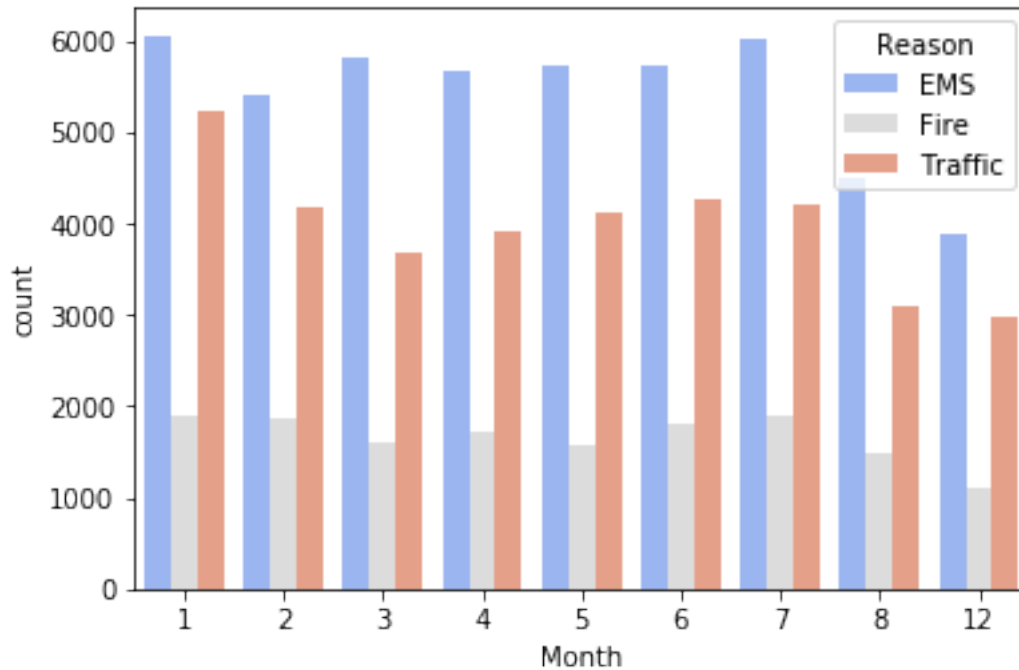


Now lets analyse reasons for the months

```
In [18]: sns.countplot(x='Month',data=df,hue='Reason',palette='coolwarm')
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x2a9098d1e48>
```





Did you notice something strange about the Plot?

\*\* You should have noticed it was missing some Months, let's plot a simple line plot to get the missing months data. \*\*

\*\* Now create a groupby object called byMonth, where you group the DataFrame by the month column and use the count() method for aggregation. Use the head() method on this returned DataFrame. \*\*

```
In [19]: byMonth = df.groupby('Month').count()
         byMonth
```

```
Out[19]:
```

	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
6	11786	11786	11786	10212	11786	11786	11777	11732	11786
7	12137	12137	12137	10633	12137	12137	12133	12088	12137
8	9078	9078	9078	7832	9078	9078	9073	9025	9078
12	7969	7969	7969	6907	7969	7969	7963	7916	7969

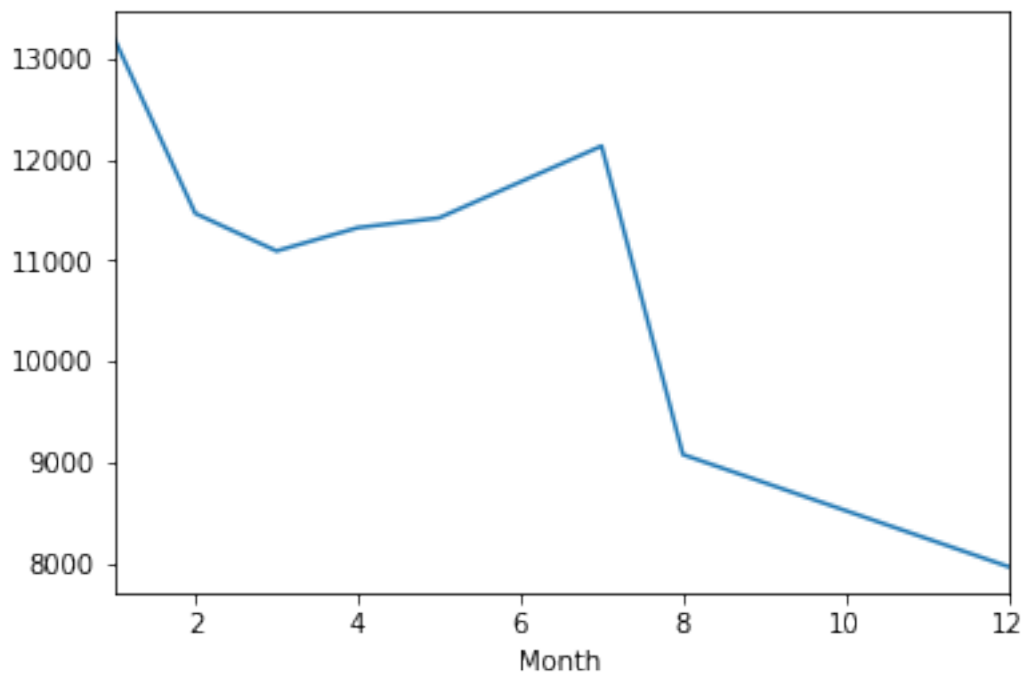
```
Reason Hour Day of Week
```

Month			
1	13205	13205	13205
2	11467	11467	11467
3	11101	11101	11101
4	11326	11326	11326
5	11423	11423	11423
6	11786	11786	11786
7	12137	12137	12137
8	9078	9078	9078
12	7969	7969	7969

**\*\* Now create a simple plot off of the dataframe indicating the count of calls per month here we can see the missing 9th and 10th month are also being plotted which shows a decreasing trend for September and October. Also, we can see that January and July notices increased number of 911 cases \*\***

```
In [20]: byMonth['twp'].plot()
```

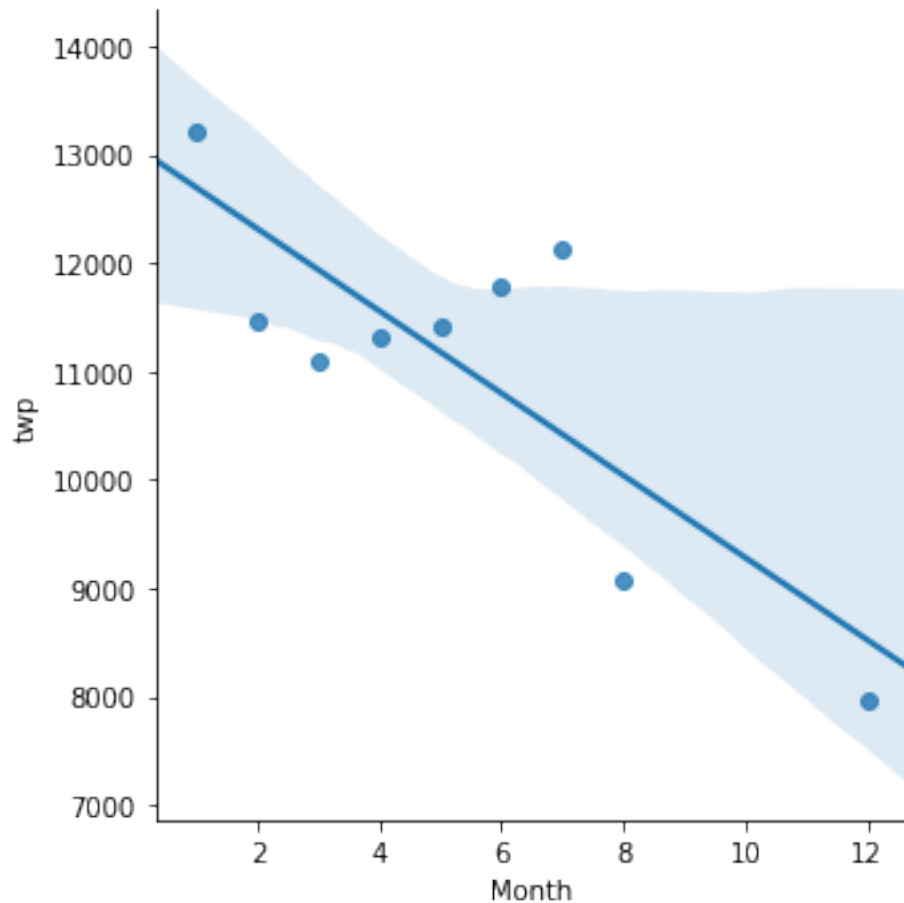
```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x2a909859ac8>
```



**\*\* Now see if you can use seaborn's lmplo() to create a linear fit on the number of calls per month and as our month is one of the index, we will reset it by using reset\_index() \*\***

```
In [21]: sns.lmplo(x='Month',y='twp',data=byMonth.reset_index())
```

```
Out[21]: <seaborn.axisgrid.FacetGrid at 0x2a909e6ca20>
```

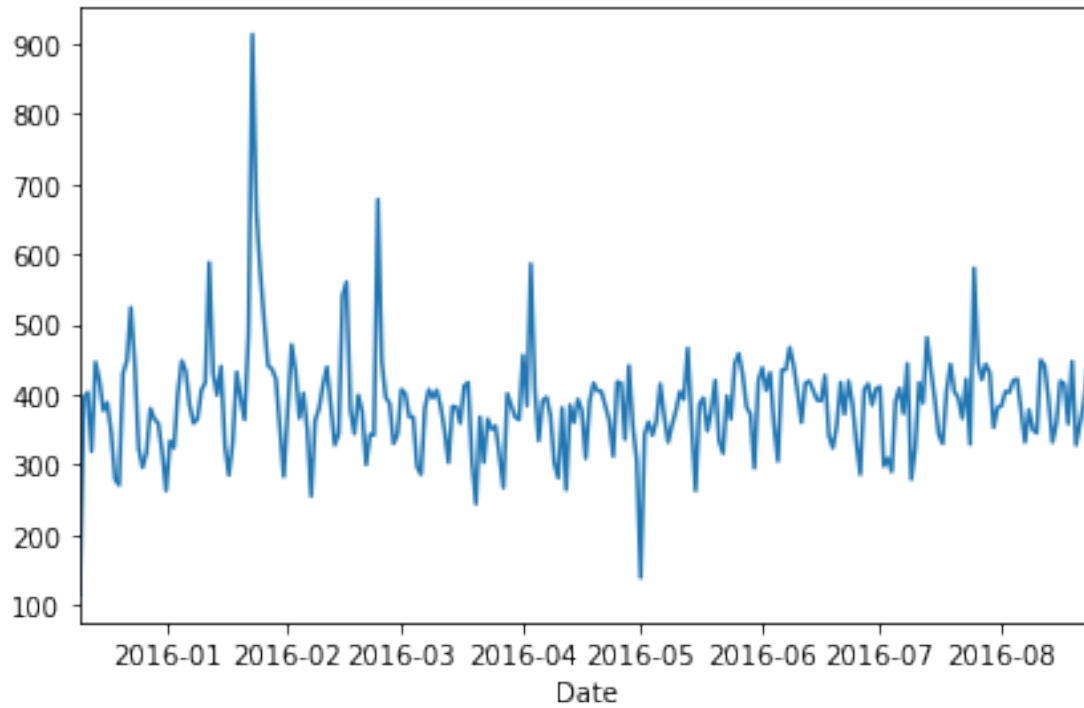


Create a new column called 'Date' that contains the date from the timeStamp column. You'll need to use apply along with the .date() method.

```
In [22]: df['Date']=df['timeStamp'].apply(lambda t: t.date())
```

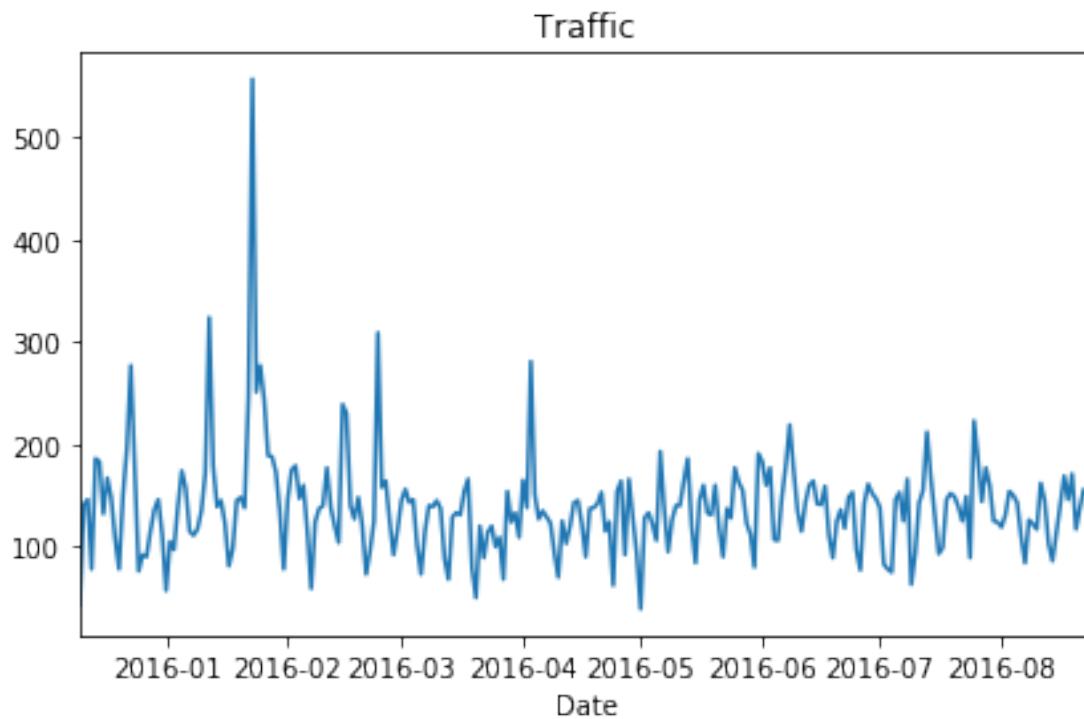
\*\* Now groupby this Date column with the count() aggregate and create a plot of counts of 911 calls.\*\*

```
In [23]: df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
```

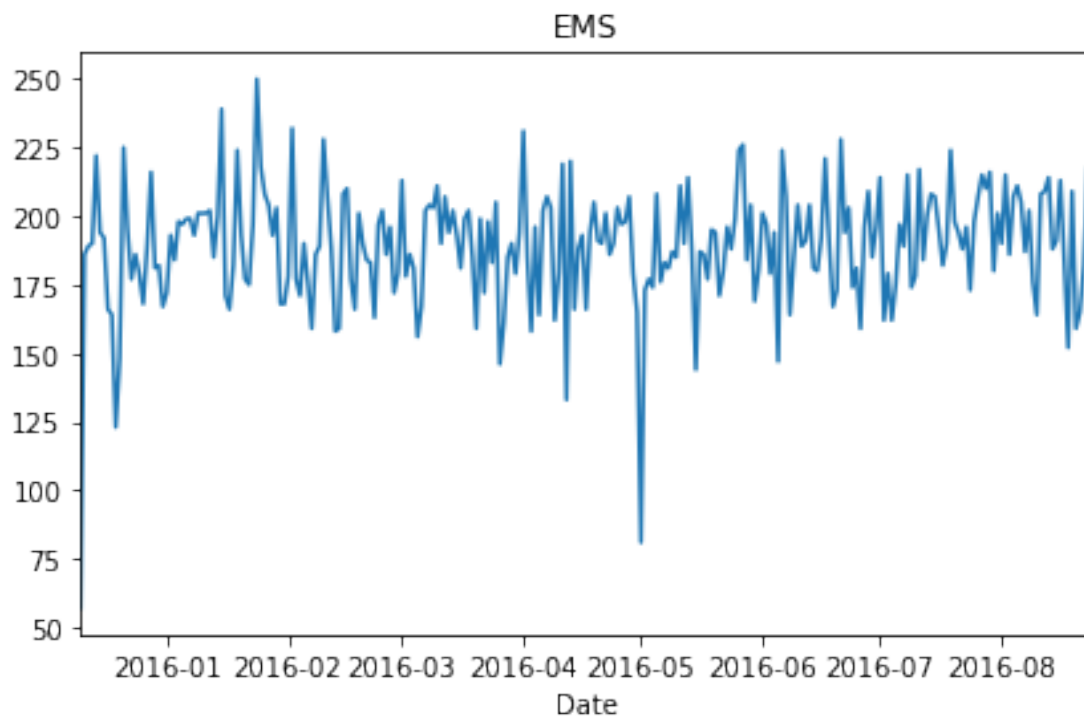


**\*\* Now lets recreate this plot but create 3 separate plots with each plot representing a Reason for the 911 call\*\***

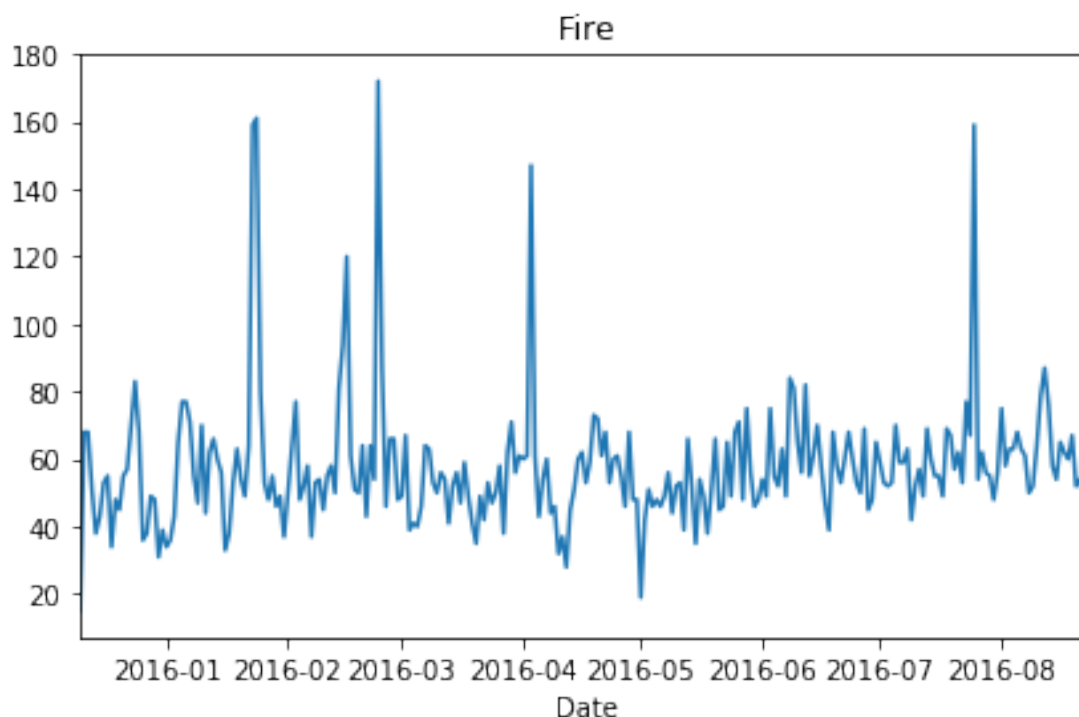
```
In [24]: df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()  
         plt.title('Traffic')  
         plt.tight_layout()
```



```
In [25]: df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()  
plt.title('EMS')  
plt.tight_layout()
```



```
In [26]: df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
plt.title('Fire')
plt.tight_layout()
```



**\*\* 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 becomes the Day of the Week using unstack()\*\***

```
In [27]: dayHour = df.groupby(by=['Day of Week', 'Hour']).count()['Reason'].unstack()
dayHour.head()
```

```
Out[27]: Hour      0      1      2      3      4      5      6      7      8      9  ...   14   15  \
Day of Week
Fri      275   235   191   175   201   194   372   598   742   752  ...   932   980
Mon      282   221   201   194   204   267   397   653   819   786  ...   869   913
Sat      375   301   263   260   224   231   257   391   459   640  ...   789   796
Sun      383   306   286   268   242   240   300   402   483   620  ...   684   691
Thu      278   202   233   159   182   203   362   570   777   828  ...   876   969

Hour      16      17      18      19      20      21      22      23
Day of Week
Fri      1039    980    820    696    667    559    514    474
```

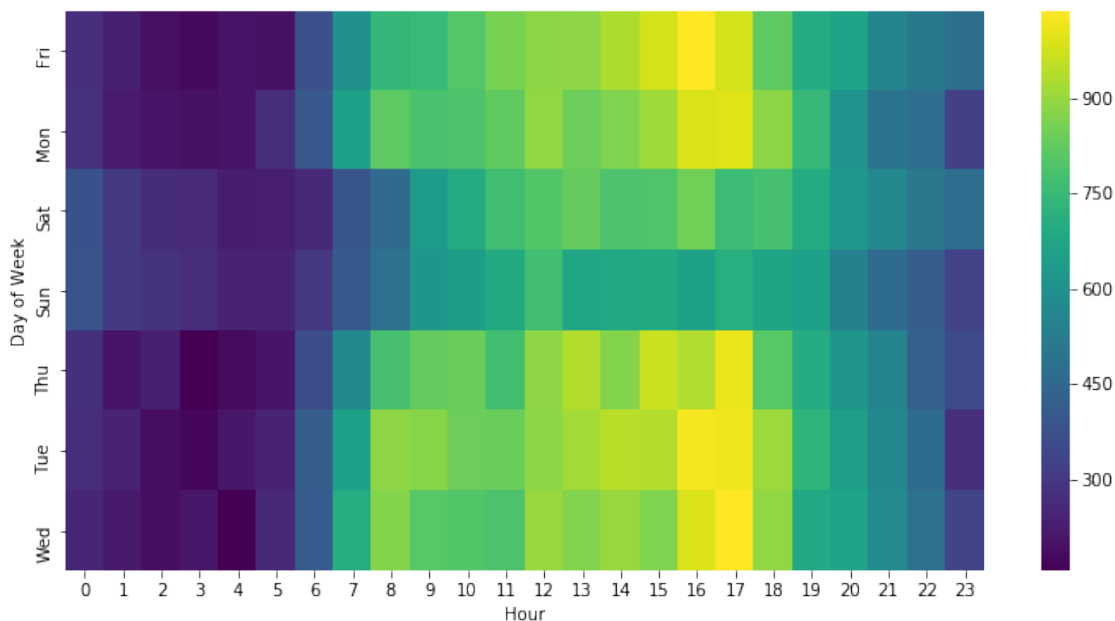
Mon	989	997	885	746	613	497	472	325
Sat	848	757	778	696	628	572	506	467
Sun	663	714	670	655	537	461	415	330
Thu	935	1013	810	698	617	553	424	354

[5 rows x 24 columns]

**\*\* Now create a HeatMap using this new DataFrame. Here, we can notice that in the afternoon and the hours between 12-5 notices increase cases of 911 calls which means in the start of the day the calls received are less and more frequent in the afternoon \*\***

```
In [28]: plt.figure(figsize=(12,6))
sns.heatmap(dayHour,cmap='viridis')
```

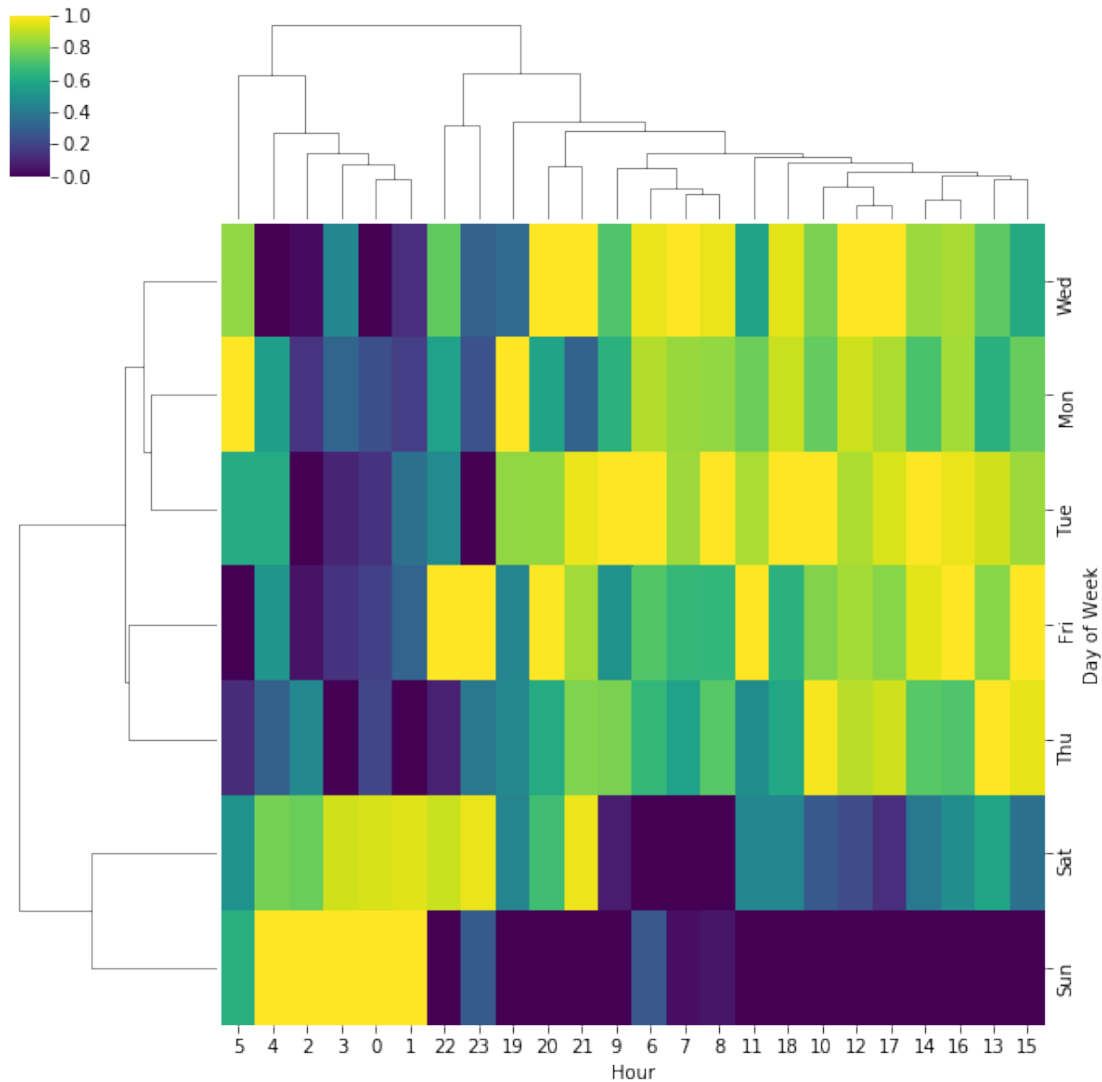
```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x2a907cf4160>
```



**\*\* Now we will create a clustermap using this DataFrame by normalising the number of calls on a standard scale of 0-1. We can notice that Thursday and Friday are combined together which shows and they have high scale of number of 911 calls whereas Monday and Wednesday are in group 2 and Saturday, Sunday are in group 3, receiving the least calls. Also, mostly on Sunday's maximum calls are received in the early morning from 12:00 am - 5:00 am \*\***

```
In [29]: sns.clustermap(dayHour,standard_scale=1,cmap='viridis')
```

```
Out[29]: <seaborn.matrix.ClusterGrid at 0x2a907cb6588>
```



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

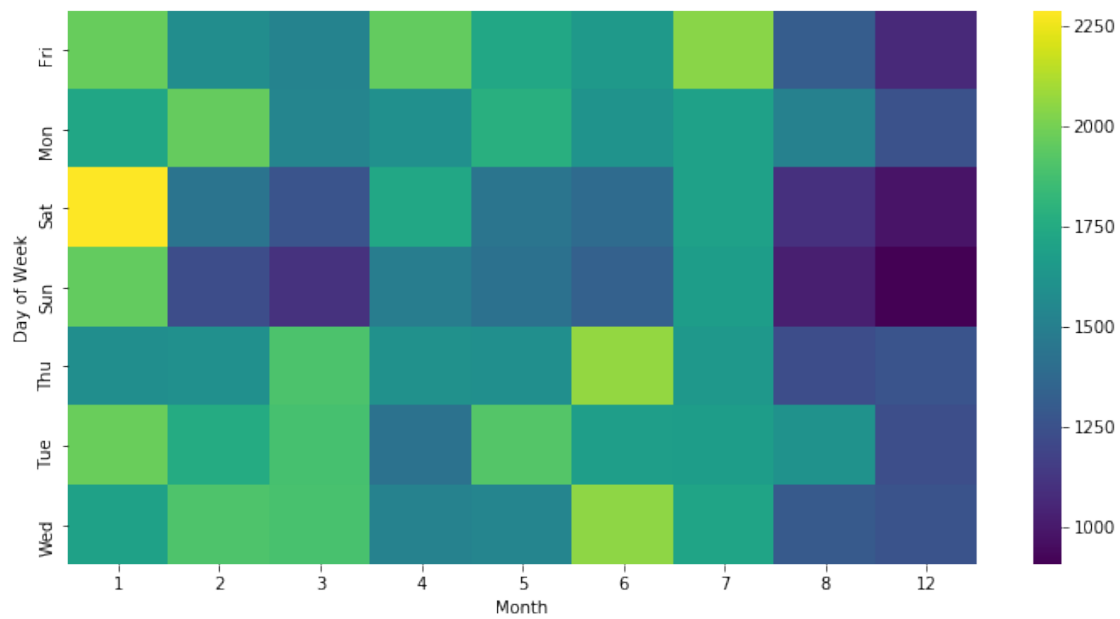
```
In [30]: dayMonth = df.groupby(by=['Day of Week', 'Month']).count()['Reason'].unstack()
          dayMonth.head()
```

```
Out[30]: 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 [31]: plt.figure(figsize=(12,6))
          sns.heatmap(dayMonth,cmap='viridis')
```

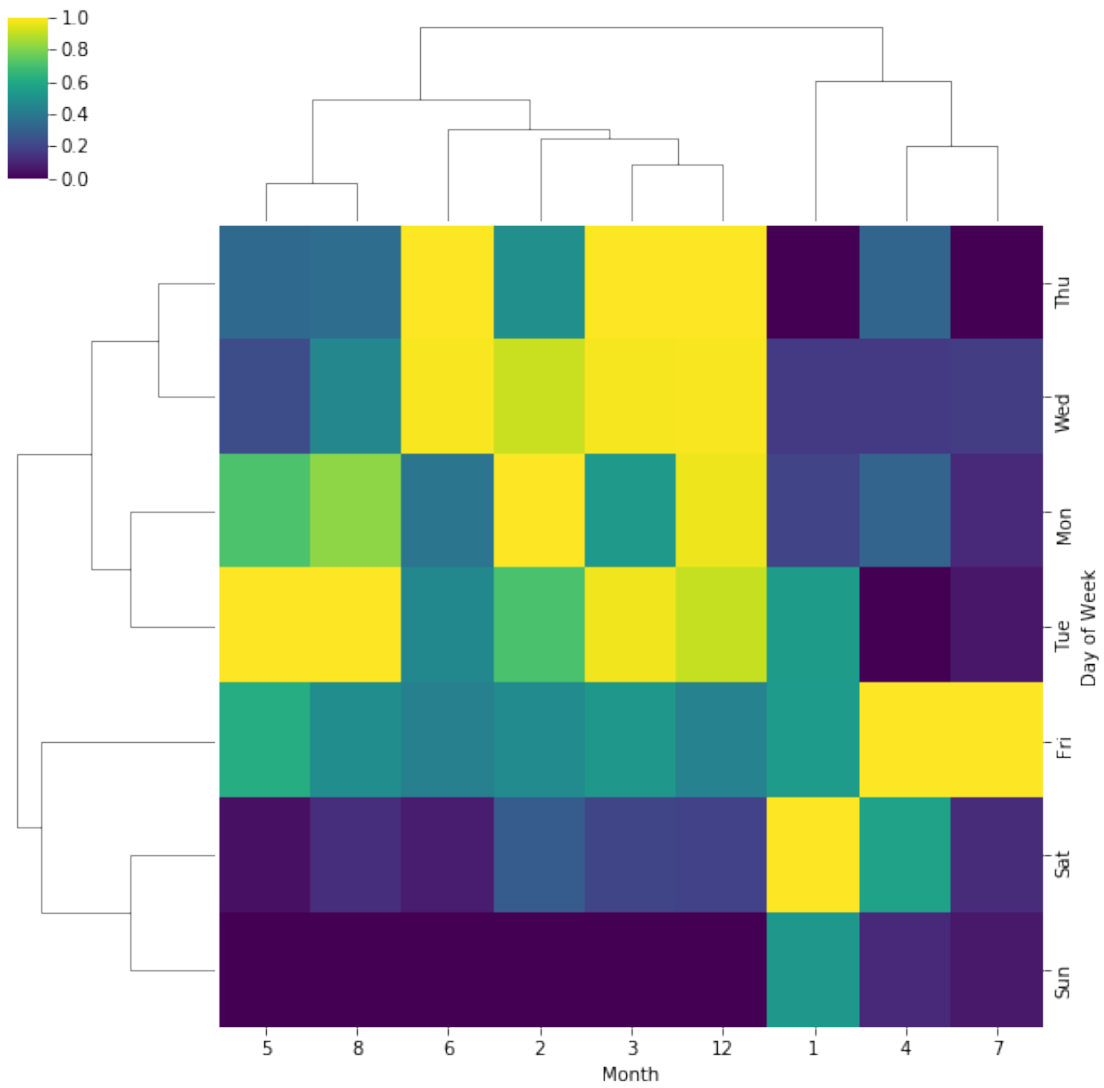


Out[31]: <matplotlib.axes.\_subplots.AxesSubplot at 0x2a90a085ba8>



In [46]: `sns.clustermap(dayMonth,cmap='viridis',standard_scale=1)`

Out[46]: <seaborn.matrix.ClusterGrid at 0x2a90a8bb978>



Thank You