911 Calls- Data Analysis Project

May 19, 2021

1 911 Calls Capstone Project

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)

1.1 Data and Setup

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

Importing visualization libraries

[2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

Reading in the csv file

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

Checking the nature of data from dataframe

[6]:
```
my_USfile.head()
```

[6]:
```
         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
```

[7]:
```
my_USfile.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
```

1.2 Basic Questions

What are the top 5 zipcodes for 911 calls?

[10]:
```
my_USfile['zip'].value_counts().head(5)
```

[10]:
```
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?

[11]:
```
my_USfile['twp'].value_counts().head(5)
```

[11]:
```
LOWER MERION    8443
ABINGTON        5977
NORRISTOWN      5890
UPPER MERION    5227
CHELTENHAM      4575
Name: twp, dtype: int64
```

Taking a look at the 'title' column- how many unique title codes are there? –> for understanding the various complaints.

[14]:
```
my_USfile['title'].nunique()    # or , len(my_USfile['title'].unique())
```

[14]:
```
110
```

1.3 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() with a custom lambda expression to create a new column called "Reason" that contains this string value.

[15]:
```
my_USfile.head()
```

[15]:

	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

[4]:
```
my_USfile['Reason'] = my_USfile['title'].apply(lambda x: x.split(':')[0])
```

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

[5]:

	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

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

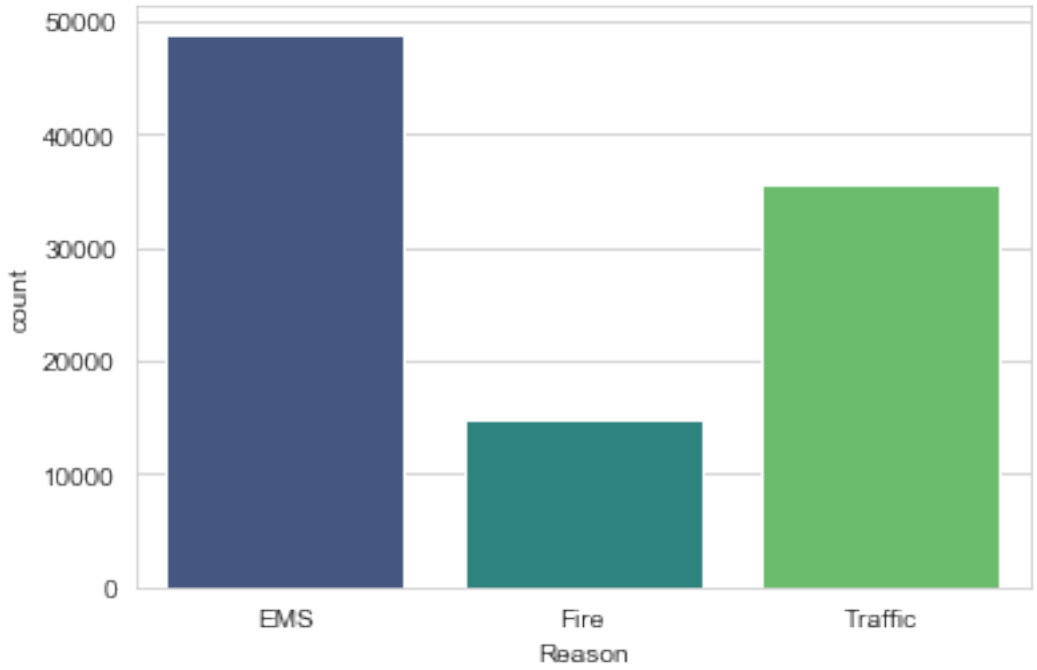
[22]:
```
my_USfile['Reason'].value_counts()
```

[22]:
```
EMS        48877
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

Now using the seaborn to create a countplot of 911 calls by Reason.

[24]:
```
sns.countplot(x='Reason', data=my_USfile, palette='viridis')
```

[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16caef0bfc8>
```



Now focusing on time information: What is the data type of the objects in the timeStamp column?

[27]:
```
type(my_USfile['timeStamp'].iloc[0])
```

[27]:
```
str
```

We will need to convert data type to date. We will use **pd.to_datetime()** for this.

[29]:
```
pd.to_datetime(my_USfile['timeStamp']).head()
```

[29]:
```
0   2015-12-10 17:40:00
1   2015-12-10 17:40:00
2   2015-12-10 17:40:00
3   2015-12-10 17:40:01
4   2015-12-10 17:40:01
Name: timeStamp, dtype: datetime64[ns]
```

[6]:
```
my_USfile['timeStamp'] = pd.to_datetime(my_USfile['timeStamp'])
```

We can now grab specific attributes from a Datetime object by calling them. For example:

```
time = df['timeStamp'].iloc[0]
time.hour
```

We will use .apply() to create 3 new columns called Hour, Month, and Day of Week.

[18]:
```
def avd_hour(myts):
    return myts.hour
my_USfile['Hour'] = my_USfile['timeStamp'].apply(avd_hour)
```

[19]:
```
def avd_month(myts):
    return myts.month
my_USfile['Month'] = my_USfile['timeStamp'].apply(avd_month)
```

[20]:
```
def avd_day(myts):
    return myts.dayofweek
my_USfile['Day of Week'] = my_USfile['timeStamp'].apply(avd_day)
```

OR: Using lambda

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)

[21]:
```
my_USfile.head()      # with new columns
```

[21]:

	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

[35]:
```
temp = my_USfile['timeStamp'][0]       # demo, how we to use methods on this␣
↪type of object
```

[33]:
```
temp.hour
```

[33]: 17

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'}
```

[22]:
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
my_USfile['Day of Week'] = my_USfile['Day of Week'].map(dmap)
```

[23]:
```
my_USfile.head()    # to show the changed column
```

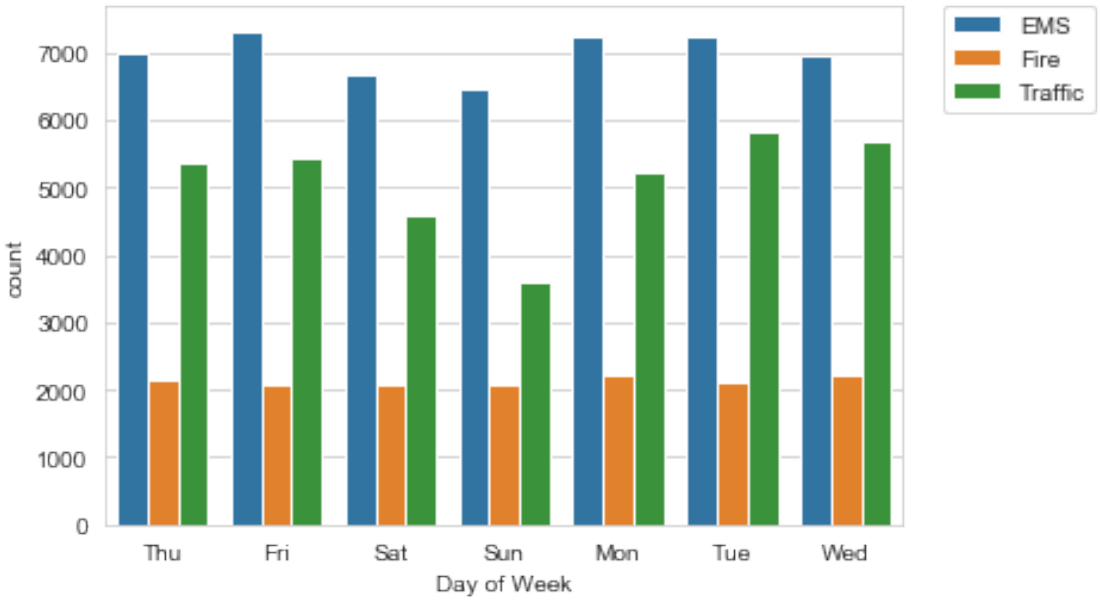
[23]:

	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

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

[51]:
```
sns.countplot(x='Day of Week', data = my_USfile, hue = 'Reason')
# To relocate the legend
plt.legend(bbox_to_anchor = (1.05, 1), loc=2, borderaxespad = 0)
```

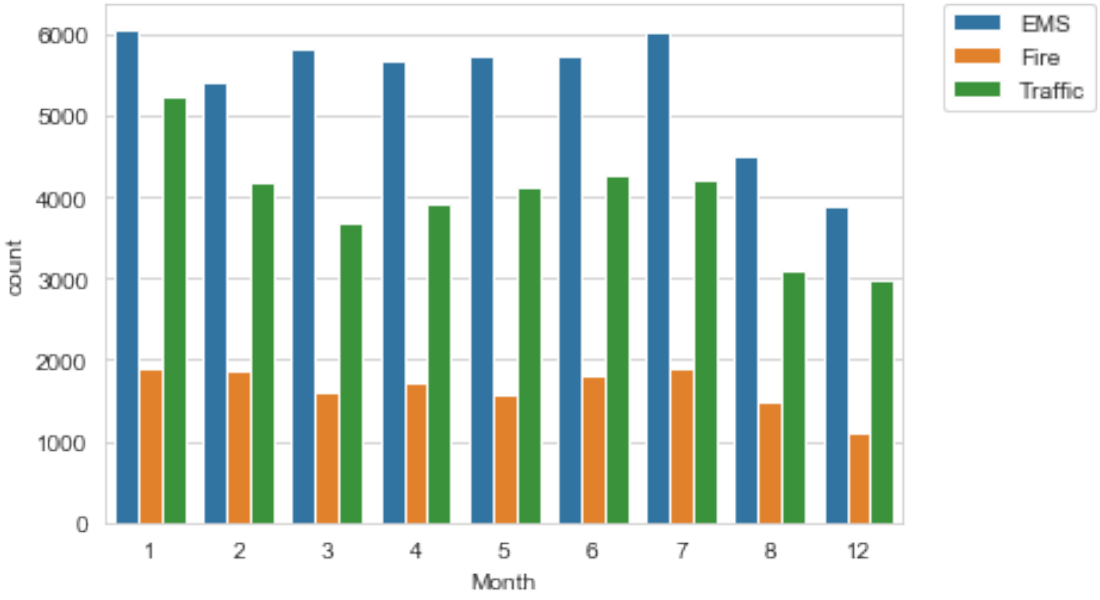
[51]: <matplotlib.legend.Legend at 0x16cb29a5d48>



Doing the same for Month:

[53]:
```
sns.countplot(x='Month', data = my_USfile, hue = 'Reason')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0)
```

[53]: <matplotlib.legend.Legend at 0x16cb32d5ac8>



We can notice something strange about the Plot - Some months are missing

Let's see if we can maybe fill in this information by plotting the information in another way, possibly a simple line plot that fills in the missing months.

Now we will create a gropuby object called byMonth, to get the count for every month.

[55]:
```
byMonth = my_USfile.groupby('Month')
```

[57]:
```
byMonth.count()
```

[57]:
```
          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
```

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

[59]:
```
type(byMonth)
```

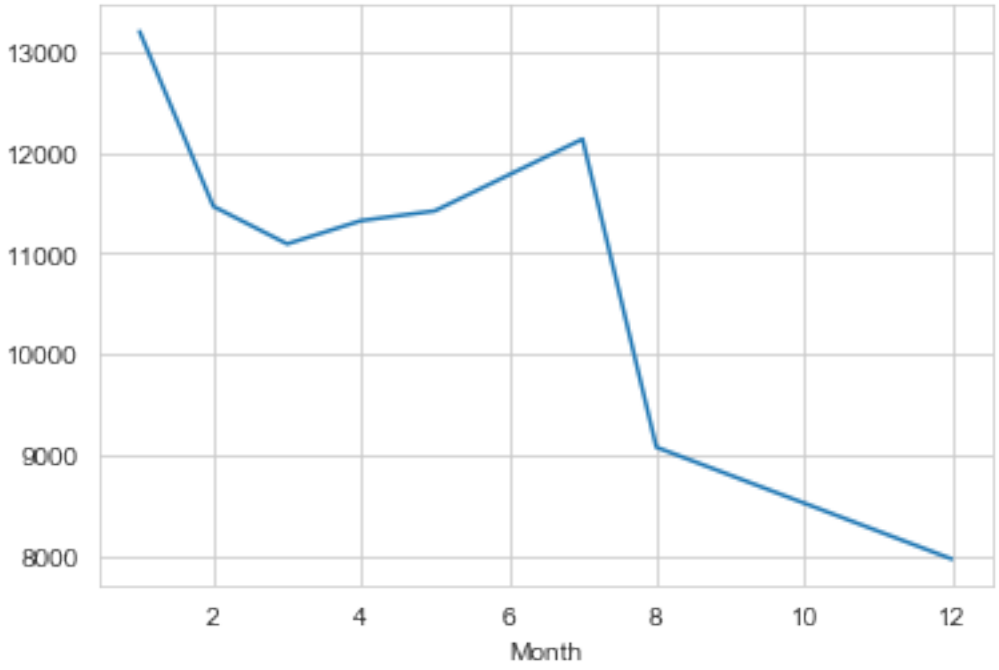
[59]:
```
pandas.core.groupby.generic.DataFrameGroupBy
```

The byMonth object is not in required form. We need to change it to a dataframe;
In solution, it is as below:

[24]:
```
byMonth = my_USfile.groupby('Month').count()      # this object is defined␣
↪after we add 'count() method'
```

[26]:
```
byMonth['twp'].plot()
```

[26]: <matplotlib.axes._subplots.AxesSubplot at 0x27a6194d908>



Question: why it is not showing '0' value for months 9, 10, 11 ?

One of the answers –> This is a line plot, so what it did is just connect the two dots with a line. We don't know exactly why the data is missing, so we did not impute the data here. - Plot just connects 8th and 12th months' values, and the missing values are assumed to be on that line

[29]:
```
byMonth
```

[29]:

Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e
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

Month	Reason	Hour	Day of Week
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 we will try to use seaborn's lmplot() to create a linear fit on the number of calls per month.

We need to reset the index, in order to be able to use the month column for any operations.

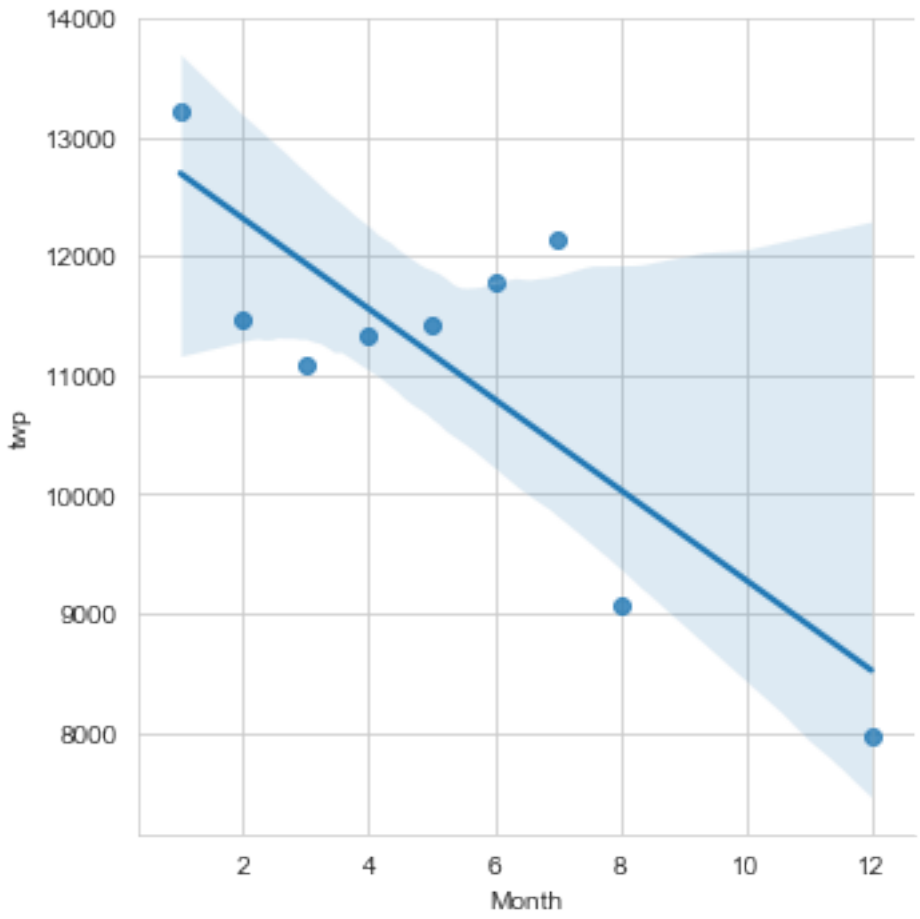
Lets do reset_Index below now

[28]:
```
byMonth.reset_index()
```

[28]:
```
    Month    lat    lng   desc    zip  title  timeStamp    twp   addr      e  \
0       1  13205  13205  13205  11527  13205      13205  13203  13096  13205
1       2  11467  11467  11467   9930  11467      11467  11465  11396  11467
2       3  11101  11101  11101   9755  11101      11101  11092  11059  11101
3       4  11326  11326  11326   9895  11326      11326  11323  11283  11326
4       5  11423  11423  11423   9946  11423      11423  11420  11378  11423
5       6  11786  11786  11786  10212  11786      11786  11777  11732  11786
6       7  12137  12137  12137  10633  12137      12137  12133  12088  12137
7       8   9078   9078   9078   7832   9078       9078   9073   9025   9078
8      12   7969   7969   7969   6907   7969       7969   7963   7916   7969

    Reason   Hour  Day of Week
0    13205  13205        13205
1    11467  11467        11467
2    11101  11101        11101
3    11326  11326        11326
4    11423  11423        11423
5    11786  11786        11786
6    12137  12137        12137
7     9078   9078         9078
8     7969   7969         7969
```

[74]:
```
sns.lmplot(x='Month', y= 'twp', data= byMonth.reset_index() )
```

[74]:
```
<seaborn.axisgrid.FacetGrid at 0x16cb33fd1c8>
```



2 ———- ———— ———— ———— ———- - - ———

———-

We will create a new column called 'Date' that contains the date from the timeStamp column.

[47]:
```
def take_date(myts):
    return myts.date()
my_USfile['Date'] = my_USfile['timeStamp'].apply(take_date)
```

[48]:
```
my_USfile.head()      # with a new column 'Date'
```

[48]:

	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
```

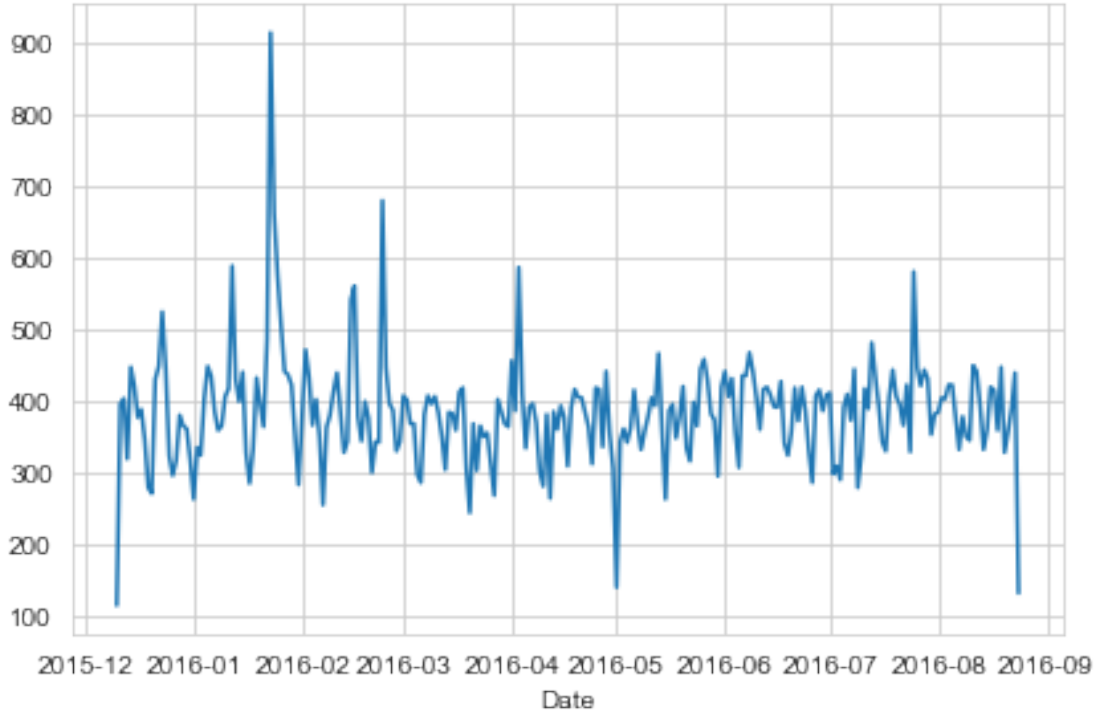
Now we will groupby this Date column with the count() aggregate and create a plot of counts of 911 calls.

[52]:
```
my_USfile.groupby('Date').count()
```

[52]:
```
             lat  lng  desc  zip  title  timeStamp  twp  addr    e  Reason  \
Date
2015-12-10   115  115   115  100    115        115  115   113  115     115
2015-12-11   396  396   396  333    396        396  395   391  396     396
2015-12-12   403  403   403  333    403        403  403   401  403     403
2015-12-13   319  319   319  280    319        319  319   317  319     319
2015-12-14   447  447   447  387    447        447  446   445  447     447
…            …    …     …    …      …          …    …     …    …       …
2016-08-20   328  328   328  279    328        328  328   327  328     328
2016-08-21   357  357   357  299    357        357  357   352  357     357
2016-08-22   389  389   389  336    389        389  388   384  389     389
2016-08-23   439  439   439  390    439        439  439   437  439     439
2016-08-24   132  132   132  106    132        132  132   132  132     132

             Hour  Month  Day of Week
Date
2015-12-10    115    115          115
2015-12-11    396    396          396
2015-12-12    403    403          403
2015-12-13    319    319          319
2015-12-14    447    447          447
…             …      …            …
2016-08-20    328    328          328
2016-08-21    357    357          357
2016-08-22    389    389          389
2016-08-23    439    439          439
```

```
2016-08-24    132     132           132

[259 rows x 13 columns]
```

[56]:
```
my_USfile.groupby('Date').count()['lat'].plot()          # any one column name␣
 ↪as an argument for plot()
plt.tight_layout()                        # to separate the labels on the axes
```

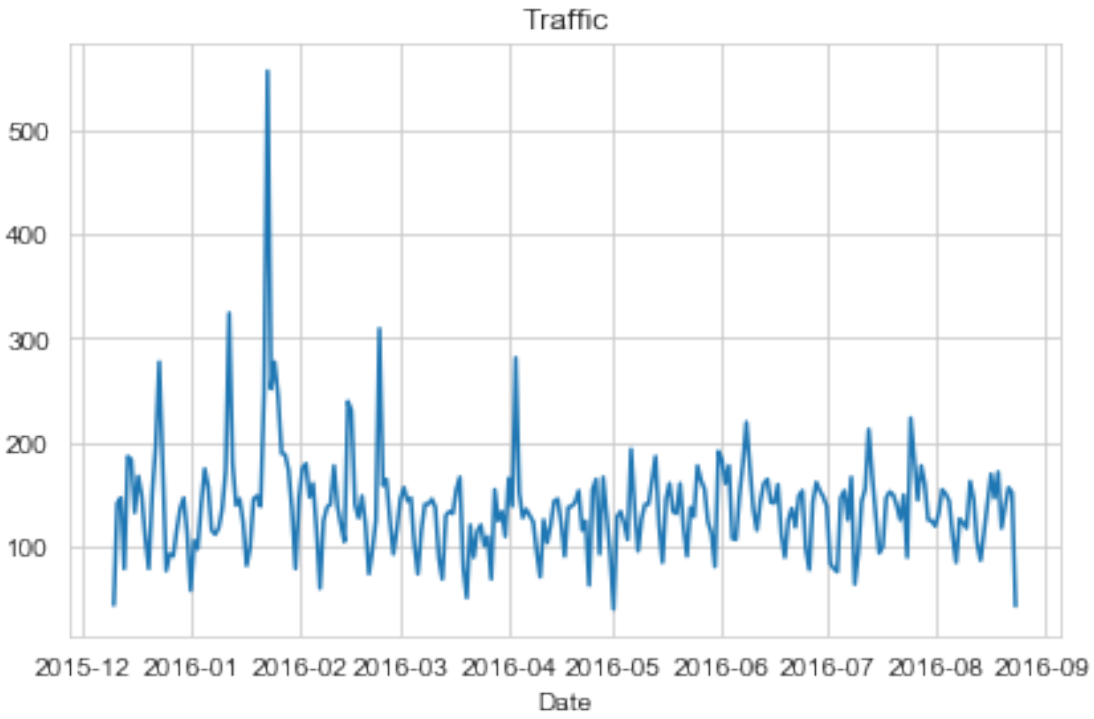


Now we will recreate this plot but 3 separate plots with each plot representing a Reason for the 911 call

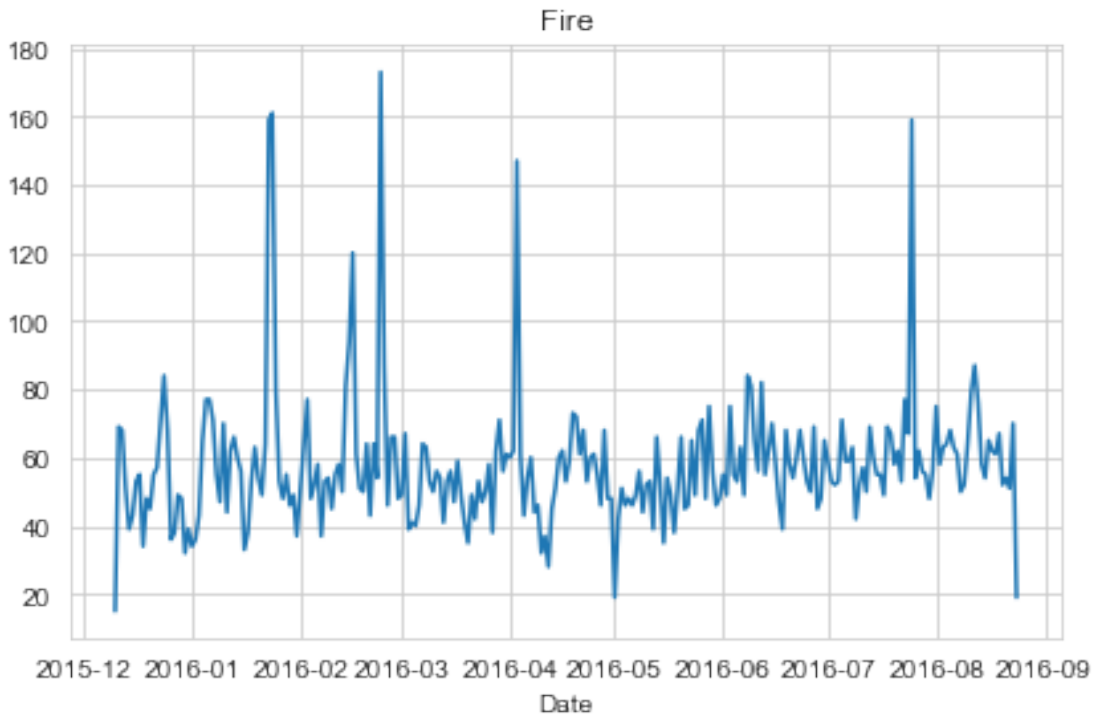
[65]:
```
my_USfile.groupby('Reason').count()
```

[65]:
```
            lat    lng   desc    zip  title  timeStamp    twp   addr      e  \
Reason
EMS       48877  48877  48877  44327  48877      48877  48853  48877  48877
Fire      14920  14920  14920  13012  14920      14920  14903  14900  14920
Traffic   35695  35695  35695  29298  35695      35695  35693  35196  35695

            Hour  Month  Day of Week   Date
Reason
EMS       48877  48877        48877  48877
Fire      14920  14920        14920  14920
Traffic   35695  35695        35695  35695
```

[80]:
```
my_USfile[my_USfile['Reason'] == 'Traffic'].groupby('Date').count()['lat'].
 ↪plot()
plt.title('Traffic')
plt.tight_layout()
```

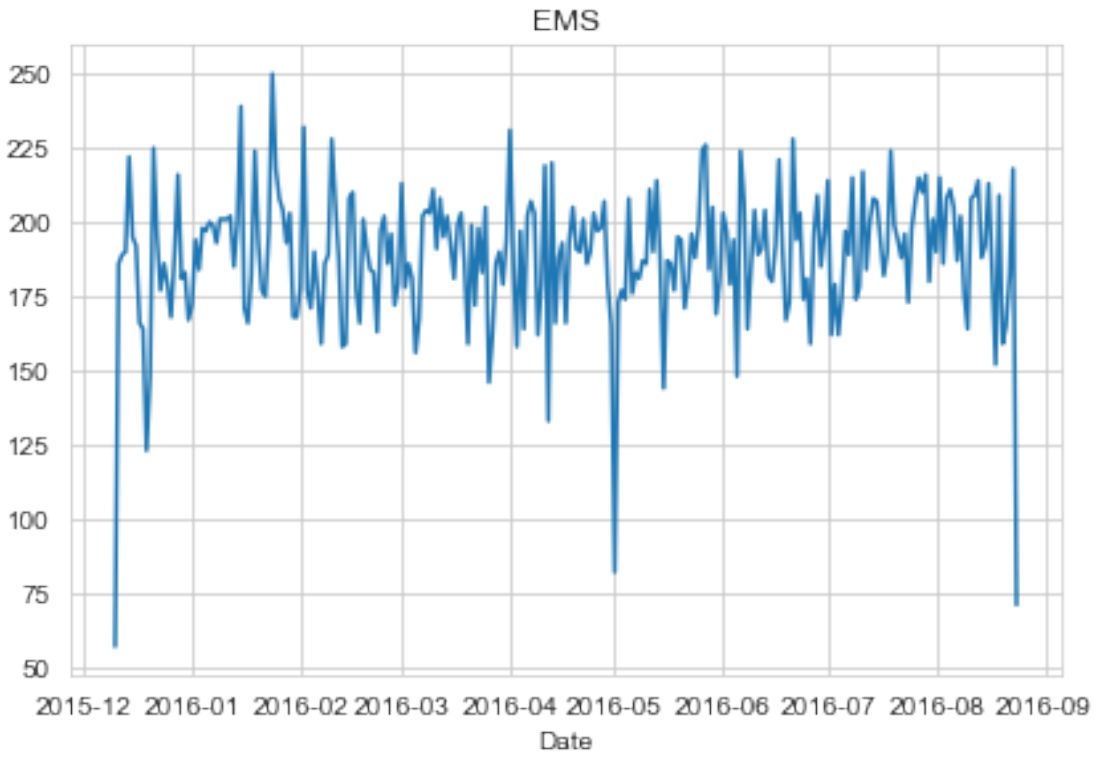


[82]:
```
my_USfile[my_USfile['Reason'] == 'Fire'].groupby('Date').count()['lat'].plot()
plt.title('Fire')
plt.tight_layout()
```



[77]:
```
my_USfile[my_USfile['Reason'] == 'EMS'].groupby('Date').count()['lat'].plot()
plt.tight_layout()
plt.title('EMS')
```

[77]: Text(0.5, 1, 'EMS')



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. We will do this using groupby along with unstack method.

[90]:
```
my_USfile.head()
```

[90]:

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

[96]:
```
my_USfile.groupby(by = ['Day of Week', 'Hour']).count().head(28)#['Reason'].
↪unstack()
```

[96]:

		lat	lng	desc	zip	title	timeStamp	twp	addr	e	\
Day of Week	Hour										
Fri	0	275	275	275	248	275	275	275	275	275	
	1	235	235	235	200	235	235	235	232	235	
	2	191	191	191	165	191	191	191	191	191	
	3	175	175	175	164	175	175	175	175	175	
	4	201	201	201	184	201	201	201	201	201	
	5	194	194	194	166	194	194	194	194	194	
	6	372	372	372	319	372	372	372	369	372	
	7	598	598	598	526	598	598	598	593	598	
	8	742	742	742	637	742	742	742	737	742	
	9	752	752	752	663	752	752	752	748	752	
	10	803	803	803	722	803	803	803	800	803	
	11	859	859	859	756	859	859	859	858	859	
	12	885	885	885	764	885	885	885	877	885	
	13	890	890	890	767	890	890	890	885	890	
	14	932	932	932	808	932	932	931	926	932	
	15	980	980	980	840	980	980	980	976	980	
	16	1039	1039	1039	897	1039	1039	1039	1038	1039	
	17	980	980	980	826	980	980	980	971	980	
	18	820	820	820	714	820	820	819	818	820	
	19	696	696	696	616	696	696	696	693	696	
	20	667	667	667	569	667	667	667	666	667	
	21	559	559	559	491	559	559	558	553	559	
	22	514	514	514	445	514	514	514	513	514	
	23	474	474	474	400	474	474	474	469	474	
Mon	0	282	282	282	243	282	282	282	282	282	
	1	221	221	221	198	221	221	220	221	221	
	2	201	201	201	183	201	201	201	201	201	
	3	194	194	194	173	194	194	194	194	194	

		Reason	Month	Date
Day of Week	Hour			
Fri	0	275	275	275
	1	235	235	235
	2	191	191	191
	3	175	175	175

```
                4         201    201    201
                5         194    194    194
                6         372    372    372
                7         598    598    598
                8         742    742    742
                9         752    752    752
                10        803    803    803
                11        859    859    859
                12        885    885    885
                13        890    890    890
                14        932    932    932
                15        980    980    980
                16       1039   1039   1039
                17        980    980    980
                18        820    820    820
                19        696    696    696
                20        667    667    667
                21        559    559    559
                22        514    514    514
                23        474    474    474
Mon             0         282    282    282
                1         221    221    221
                2         201    201    201
                3         194    194    194
```

2.0.1 This previous processing is very imp. It makes a multi-level index.

[101]:
```
my_USfile.groupby(by = ['Day of Week', 'Hour']).count()['Reason'].head(10)#␣
↪This 'reason' is just to select any one column values.
                                    # unstack()    -- this will be added in␣
↪next step
```

[101]:
```
Day of Week  Hour
Fri          0       275
             1       235
             2       191
             3       175
             4       201
             5       194
             6       372
             7       598
             8       742
             9       752
Name: Reason, dtype: int64
```

[100]:
```
my_USfile.groupby(by = ['Day of Week', 'Hour']).count()['Reason'].unstack()
```

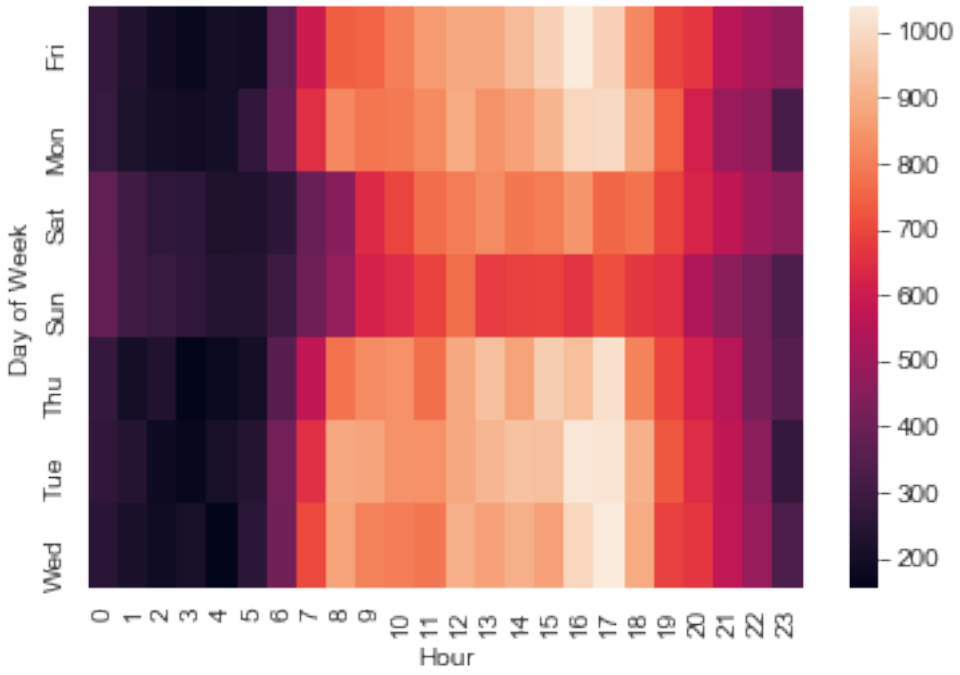
[100]:
```
Hour           0    1    2    3    4    5    6    7    8    9   …   14   15  \
Day of Week                                                      …
Fri          275  235  191  175  201  194  372  598  742  752  …  932  980
Mon          282  221  201  194  204  267  397  653  819  786  …  869  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
Tue          269  240  186  170  209  239  415  655  889  880  …  943  938
Wed          250  216  189  209  156  255  410  701  875  808  …  904  867

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
Tue          1026  1019  905  731  647  571  462  274
Wed           990  1037  894  686  668  575  490  335

[7 rows x 24 columns]
```

Now we will create a HeatMap using this new DataFrame.

[102]:
```
US_Heat = my_USfile.groupby(by = ['Day of Week', 'Hour']).count()['Reason'].
↪unstack()
```

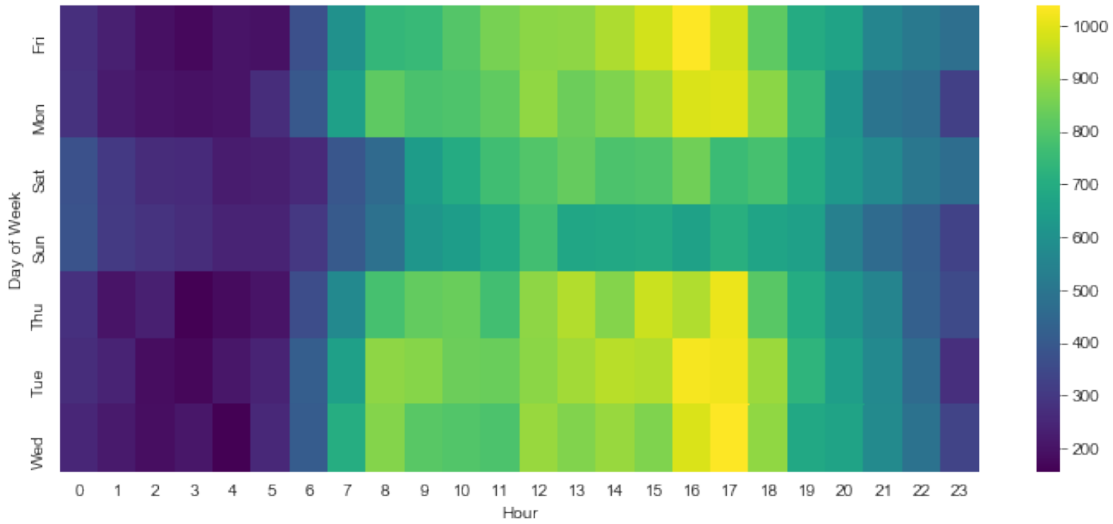
[110]:
```
sns.heatmap(US_Heat) # this default cmap is 'magma'
```

[110]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27a64016948>
```



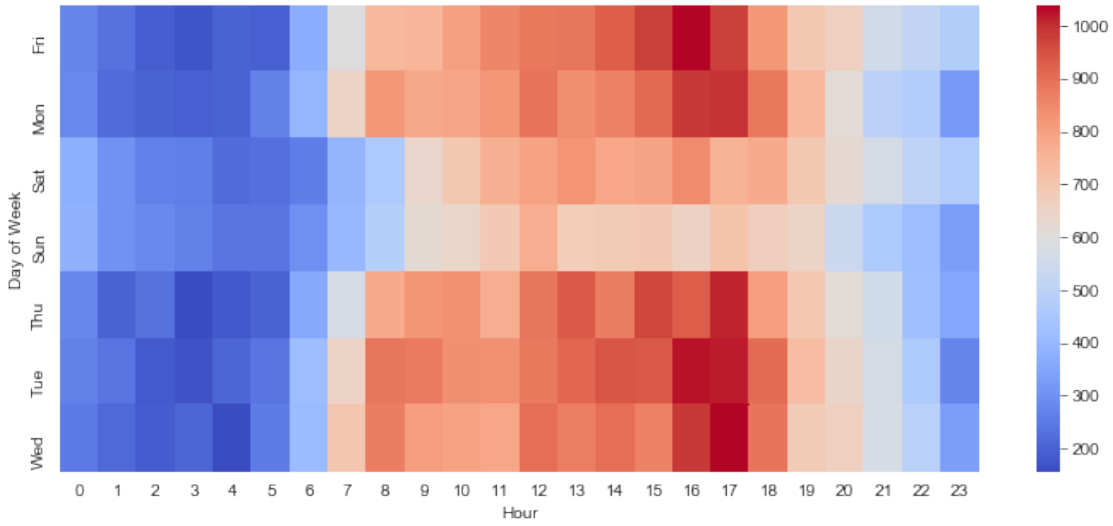
[109]:
```
plt.figure(figsize = (12,5))
sns.heatmap(US_Heat, cmap = 'viridis')
```

[109]: <matplotlib.axes._subplots.AxesSubplot at 0x27a64975d48>



[111]:
```
plt.figure(figsize = (12,5))
sns.heatmap(US_Heat, cmap = 'coolwarm')
```

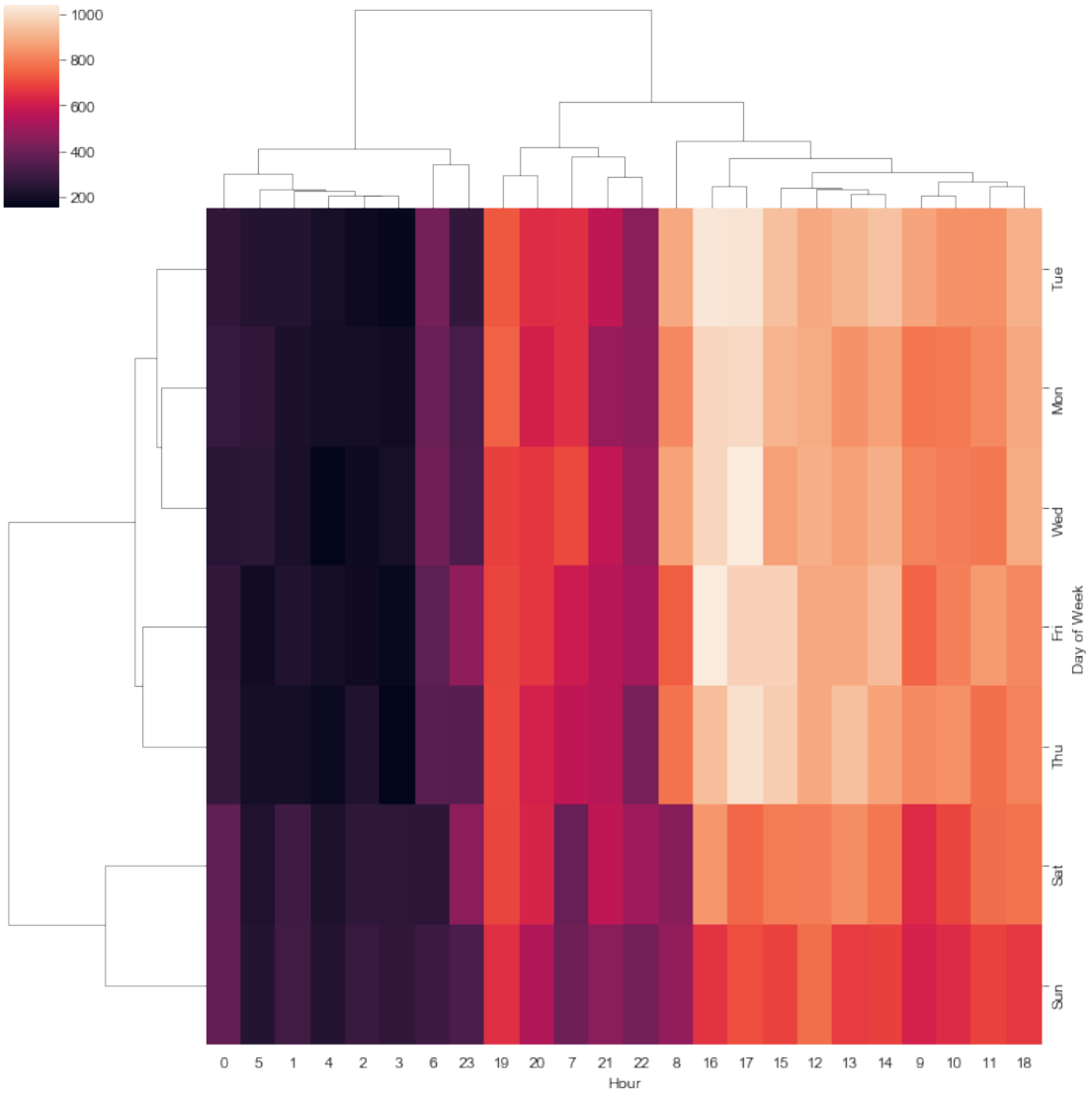
[111]: <matplotlib.axes._subplots.AxesSubplot at 0x27a63f5be08>



Now we will create a clustermap using this DataFrame.

[112]:
```
sns.clustermap(US_Heat)
```

[112]: <seaborn.matrix.ClusterGrid at 0x27a637d6e48>



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

[121]:
```
my_USfile.groupby(by=['Day of Week', 'Month']).count()['Reason']      # It has␣
↪only 63 entries, as info abt only 9 months is
                                                                      # there. So, 9 * 7(days)
```

[121]:
```
Day of Week  Month
Fri          1        1970
             2        1581
             3        1525
             4        1958
             5        1730
```

```
                    …
Wed          5        1538
             6        2058
             7        1717
             8        1295
             12       1262
Name: Reason, Length: 63, dtype: int64
```

[123]:
```
my_USfile.groupby(by=['Day of Week', 'Month']).count()['Reason'].unstack()
```

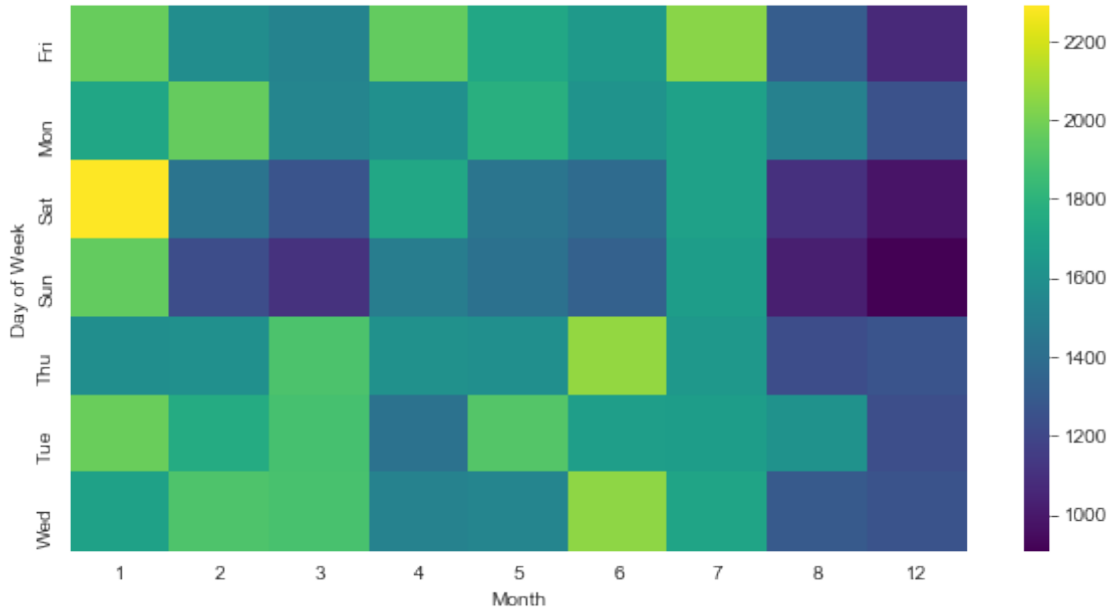
[123]:

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

[124]:
```
US_month = my_USfile.groupby(by=['Day of Week', 'Month']).count()['Reason'].
↪unstack()
```

[127]:
```
plt.figure(figsize = (10, 5))
sns.heatmap(US_month, cmap = 'viridis')
```

[127]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27a64940cc8>
```



[129]:
```
sns.clustermap(US_month,cmap = 'viridis')
```

[129]: <seaborn.matrix.ClusterGrid at 0x27a64fc0c48>