my-capstone-project-on-911-call

June 9, 2022

1 my-capstone-project-on-911-call

Use the "Run" button to execute the code.

[ ]:
```
!pip install jovian --upgrade --quiet
```

[ ]:
```
import jovian
```

[ ]:
```
# Execute this to save new versions of the notebook
jovian.commit(project="my-capstone-project-on-911-call")
```

```
<IPython.core.display.Javascript object>
```

911 Calls Capstone Project - Solutions 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) Just go along with this notebook and try to complete the instructions or answer the questions in bold using your Python and Data Science skills!

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

[3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

[4]:
```
ls
```

```
'02-911 Calls Data Capstone Project - Solutions.ipynb'
 911.csv
 my-capstone-project-on-911-call.ipynb
 work/
```

[7]:
```
df=pd.read_csv('911.csv')
```

[8]:
```
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
```

[131]:
```
df.head()
```

[131]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Month	Day of week	Date
0	40.297876	-75.581294	REINDEER CT & DEAD END; NEW HANOVER; Station …	19525.0	EMS: BACK PAINS/INJURY	2015-12-10 17:40:00	NEW HANOVER	REINDEER CT & DEAD END	1	EMS	17	12	Thu	2015-12-10
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP…	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:40:00	HATFIELD TOWNSHIP	BRIAR PATH & WHITEMARSH LN	1	EMS	17	12	Thu	2015-12-10
2	40.121182	-75.351975	HAWS AVE; NORRISTOWN; 2015-12-10 @ 14:39:21-St…	19401.0	Fire: GAS-ODOR/LEAK	2015-12-10 17:40:00	NORRISTOWN	HAWS AVE	1	Fire	17	12	Thu	2015-12-10
3	40.116153	-75.343513	AIRY ST & SWEDE ST; NORRISTOWN; Station 308A;…	19401.0	EMS: CARDIAC EMERGENCY	2015-12-10 17:40:01	NORRISTOWN	AIRY ST & SWEDE ST	1	EMS	17	12	Thu	2015-12-10
4	40.251492	-75.603350	CHERRYWOOD CT & DEAD END; LOWER POTTSGROVE; S…	NaN	EMS: DIZZINESS	2015-12-10 17:40:01	LOWER POTTSGROVE	CHERRYWOOD CT & DEAD END	1	EMS	17	12	Thu	2015-12-10

[18]:
```
df['zip'].value_counts().head()
```

[18]:
```
19401.0    6979
19464.0    6643
19403.0    4854
```

```
19446.0    4748
19406.0    3174
Name: zip, dtype: int64
```

[19]:
```
df['twp'].value_counts().head()
```

[19]:
```
LOWER MERION    8443
ABINGTON        5977
NORRISTOWN      5890
UPPER MERION    5227
CHELTENHAM      4575
Name: twp, dtype: int64
```

[26]:
```
df['title']
```

[26]:
```
0              EMS: BACK PAINS/INJURY
1             EMS: DIABETIC EMERGENCY
2                 Fire: GAS-ODOR/LEAK
3              EMS: CARDIAC EMERGENCY
4                      EMS: DIZZINESS
                      …
99487    Traffic: VEHICLE ACCIDENT -
99488    Traffic: VEHICLE ACCIDENT -
99489                EMS: FALL VICTIM
99490           EMS: NAUSEA/VOMITING
99491    Traffic: VEHICLE ACCIDENT -
Name: title, Length: 99492, dtype: object
```

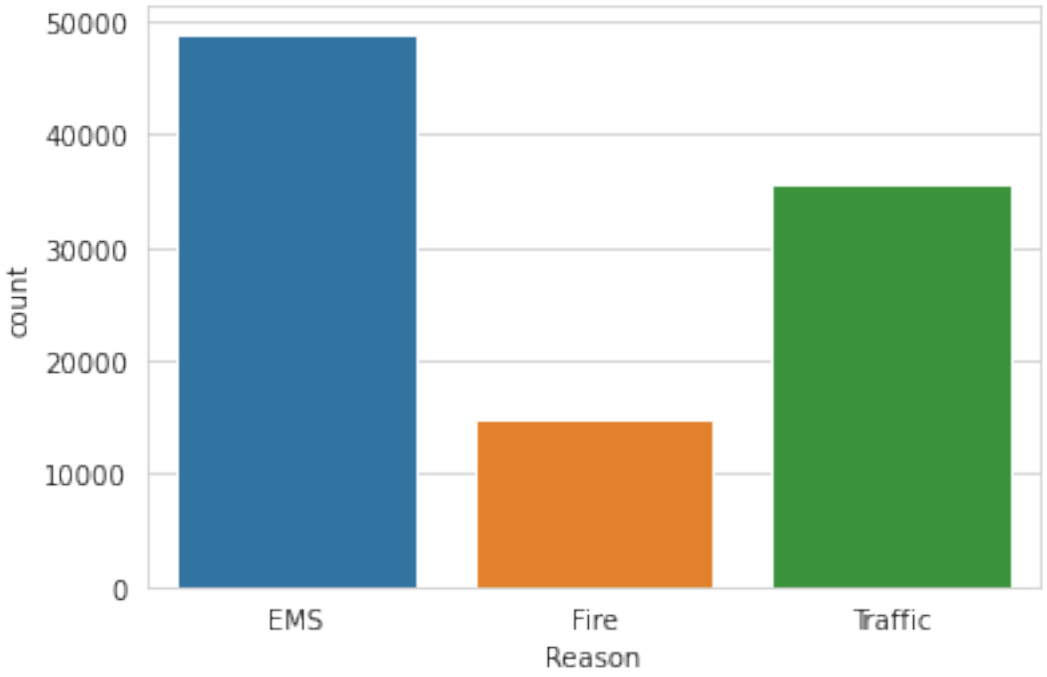
[27]:
```
df['Reason']=df['title'].apply(lambda title:title.split(':')[0])
```

[31]:
```
df['Reason'].value_counts()
```

[31]:
```
EMS        48877
Traffic    35695
Fire       14920
Name: Reason, dtype: int64
```

[33]:
```
sns.countplot(x='Reason',data=df)
```

[33]:
```
<AxesSubplot:xlabel='Reason', ylabel='count'>
```



[37]:
```
type(df['timeStamp'].iloc[0])
```

[37]:
```
str
```

[39]:
```
df['timeStamp']=pd.to_datetime(df['timeStamp'])
```

[140]:
```
type(df['timeStamp'].iloc[0])
```

[140]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

[144]:
```
time=df['timeStamp'].iloc[0]
time
```

[144]:
```
Timestamp('2015-12-10 17:40:00')
```

[157]:
```
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)
```

[59]:
```
df['Hour'].unique()
```

[59]:
```
array([17, 18, 19, 20, 21, 22, 23,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9,
       10, 11, 12, 13, 14, 15, 16])
```

[158]:
```
df['Month'].unique()
```

[158]:
```
array([12,  1,  2,  3,  4,  5,  6,  7,  8])
```

[64]:
```
df['Day of week'].unique()
```

[64]:
```
array([3, 4, 5, 6, 0, 1, 2])
```

[65]:
```
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
```

[66]:
```
df['Day of week']=df['Day of week'].map(dmap)
```

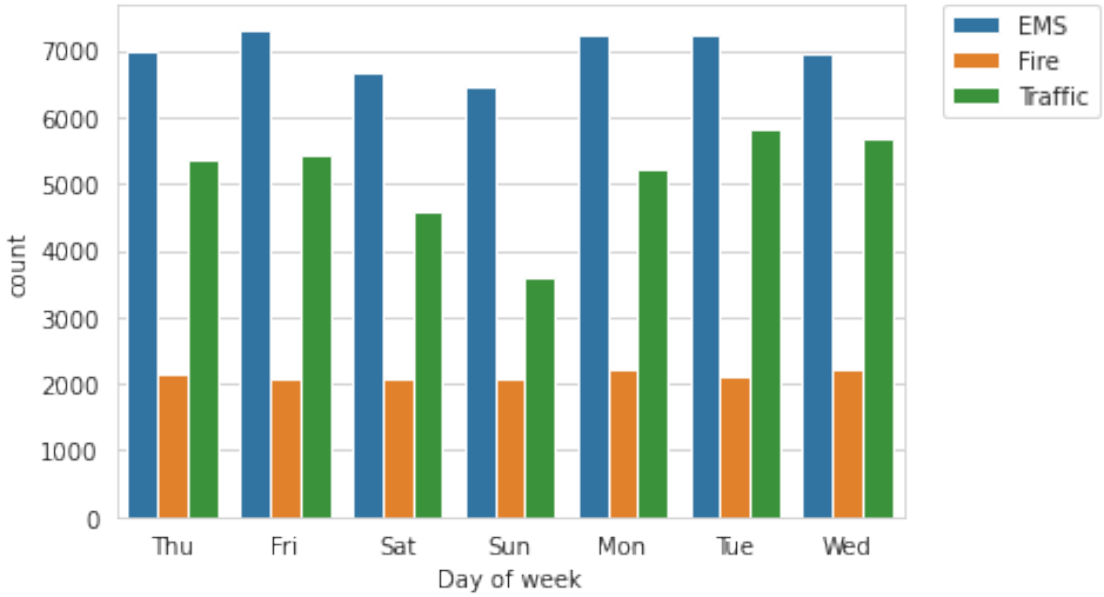
[86]:
```
df['Day of week'].unique()
```

[86]:
```
array(['Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Tue', 'Wed'], dtype=object)
```

[84]:
```
sns.countplot(x='Day of week',data=df,hue='Reason')

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

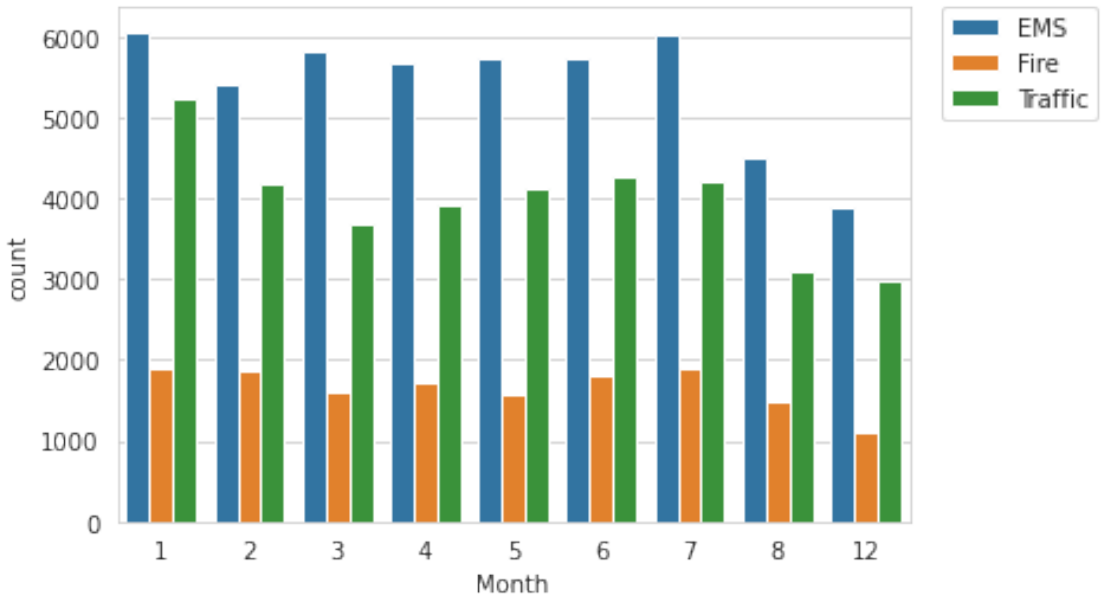
[84]:
```
<matplotlib.legend.Legend at 0x7f77f46b6df0>
```



[159]:
```
sns.countplot(x='Month',data=df,hue='Reason')

plt.legend(bbox_to_anchor=(1.04,1),loc=2,borderaxespad=0.)
```

[159]:
```
<matplotlib.legend.Legend at 0x7f77eb1701f0>
```



[160]:
```
byMonth=df.groupby('Month').count()
```

[161]:
```
byMonth
```

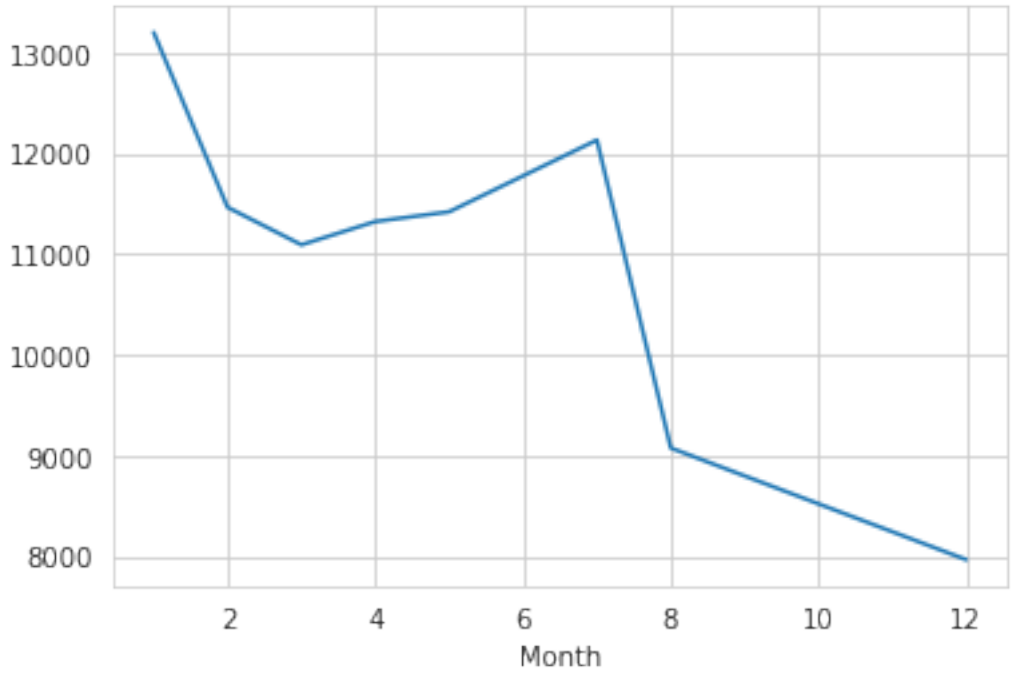
[161]:

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

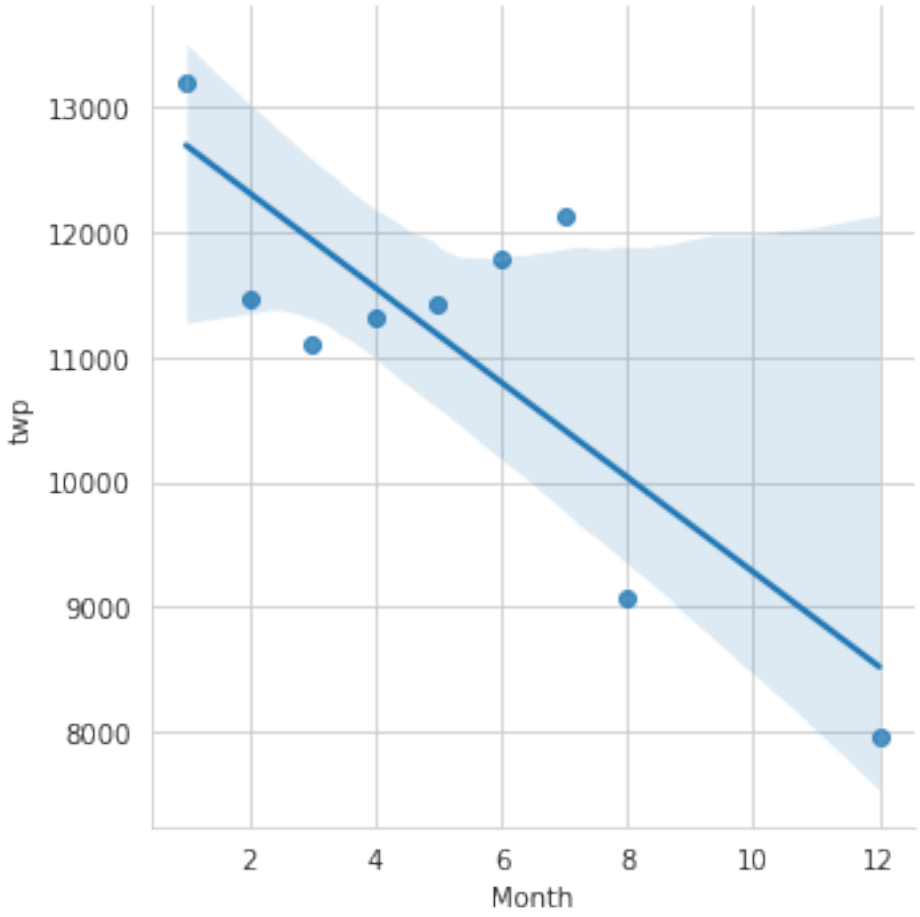
[162]:
```
byMonth['twp'].plot()
```

[162]: <AxesSubplot:xlabel='Month'>



[101]:
```
sns.lmplot(x='Month',y='twp',data=byMonth.reset_index())
```

[101]: <seaborn.axisgrid.FacetGrid at 0x7f77e389bd00>

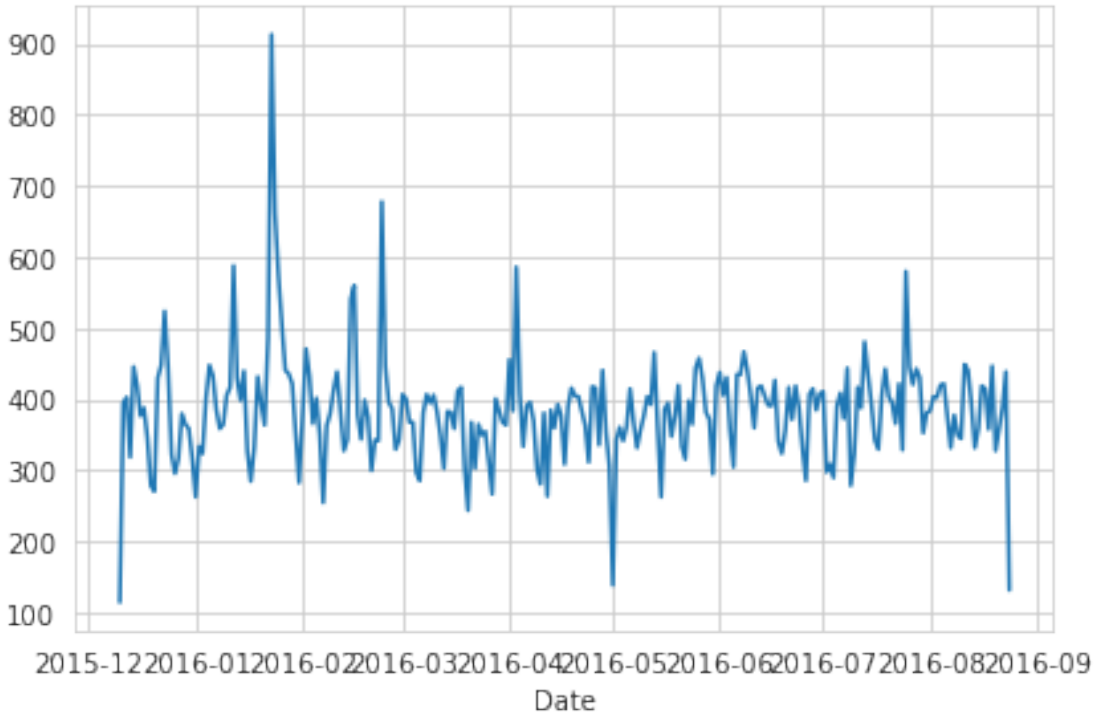


[118]:
```
df['Date']=df['timeStamp'].apply(lambda t:t.date())
```

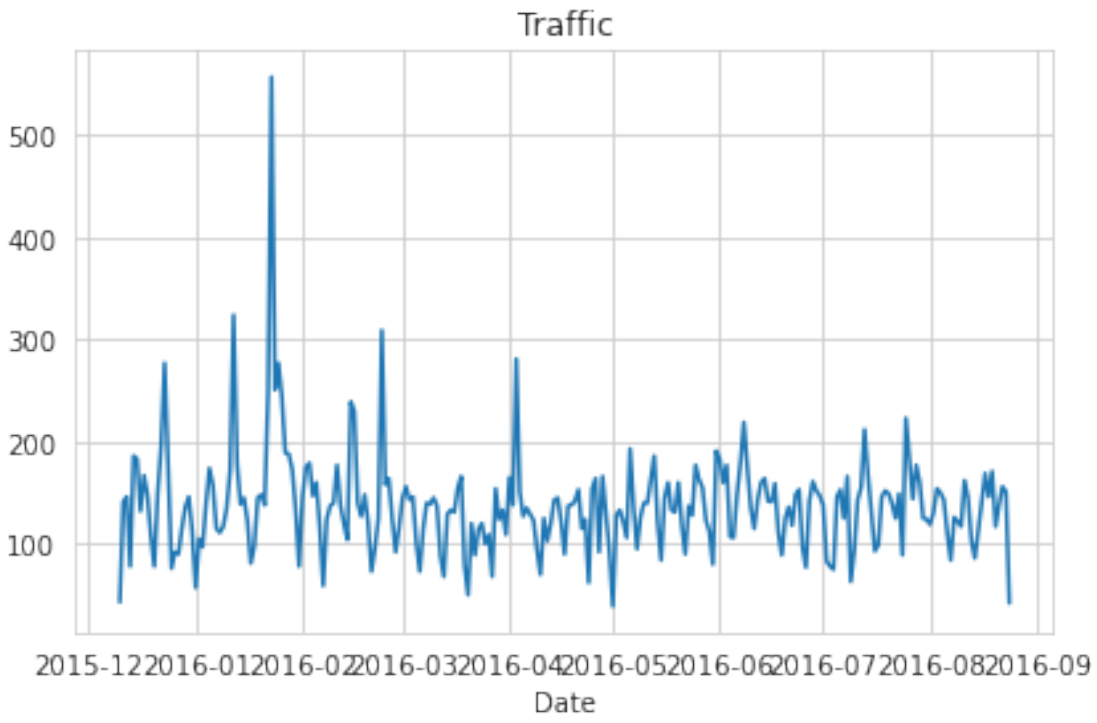
[119]:
```
df['Date'].head()
```

[119]:
```
0    2015-12-10
1    2015-12-10
2    2015-12-10
3    2015-12-10
4    2015-12-10
Name: Date, dtype: object
```

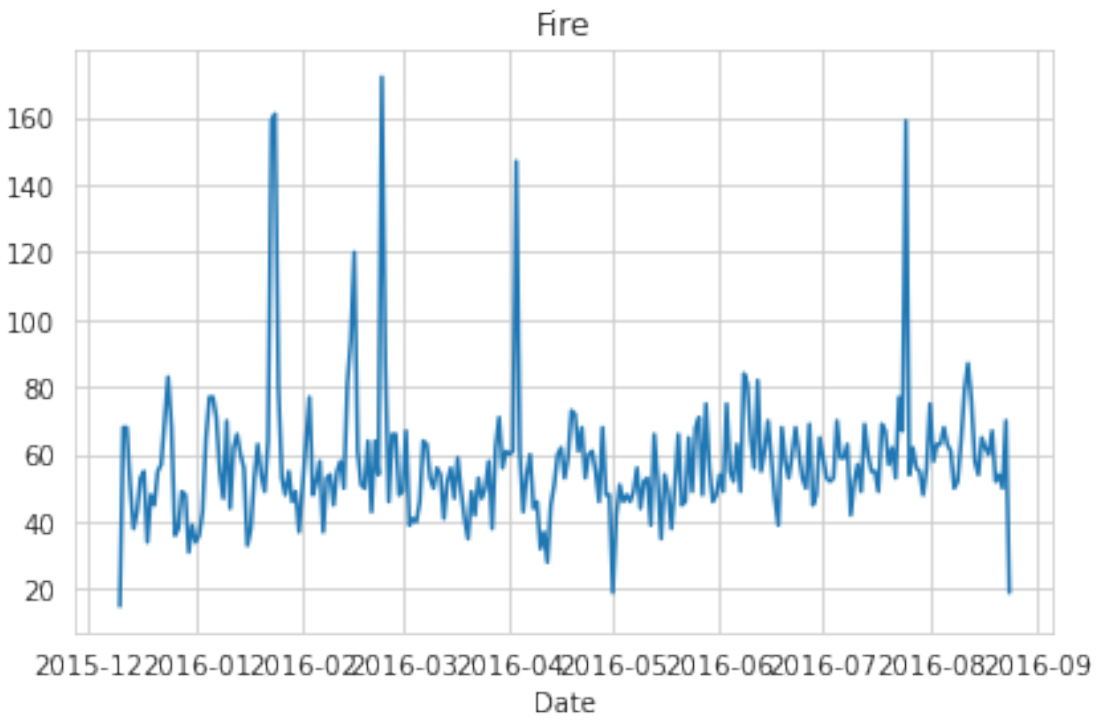
[120]:
```
df.groupby('Date').count()['twp'].plot()
plt.tight_layout()
```



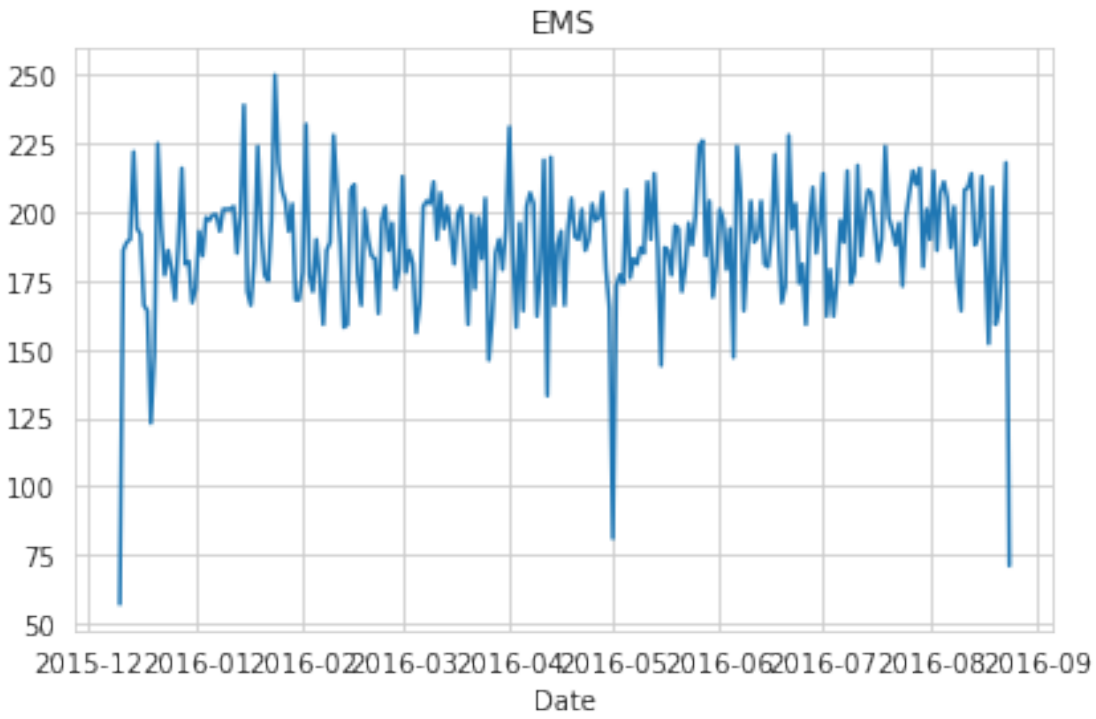
[149]:
```
df[df['Reason']=='Traffic'].groupby('Date').count()['twp'].plot()
plt.title('Traffic')
plt.tight_layout()
```



[150]:
```
df[df['Reason']=='Fire'].groupby('Date').count()['twp'].plot()
plt.title('Fire')
plt.tight_layout()
```



[163]:
```
df[df['Reason']=='EMS'].groupby('Date').count()['twp'].plot()
plt.title('EMS')
plt.tight_layout()
```


EMS
250
225
200
175
150
125
100
75
50
2015-12
2016-01
2016-02
2016-03
2016-04
2016-05
2016-06
2016-07
2016-08
2016-09
Date