Dhir Thacker | 17070122019 | CSE-1

SD lab assignment 1

Problem statement : Write python code that loads any dataset and plot the graph

Code screenshots:

Dhir Thacker

17070122019

C1

In [4]:
```
import numpy as np
import pandas as pd
```

In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [6]:
```
df = pd.read_csv('911.csv')
```

In [7]:
```
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
```

Showing the '911.csv' dataset

```
In [9]: df.head(10)
```

Out[9]:

	lat	lng	desc	zip	title	timeStamp	twp	addr	e
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
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
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
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
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
5	40.253473	-75.283245	CANNON AVE & W 9TH ST; LANSDALE; Station 345;...	19446.0	EMS: HEAD INJURY	2015-12-10 17:40:01	LANSDALE	CANNON AVE & W 9TH ST	1
6	40.182111	-75.127795	LAUREL AVE & OAKDALE AVE; HORSHAM; Station 35...	19044.0	EMS: NAUSEA/VOMITING	2015-12-10 17:40:01	HORSHAM	LAUREL AVE & OAKDALE AVE	1
7	40.217286	-75.405182	COLLEGEVILLE RD & LYWISKI RD; SKIPPACK; Stati...	19426.0	EMS: RESPIRATORY EMERGENCY	2015-12-10 17:40:01	SKIPPACK	COLLEGEVILLE RD & LYWISKI RD	1
8	40.289027	-75.399590	MAIN ST & OLD SUMNEYTOWN PIKE; LOWER SALFORD;...	19438.0	EMS: SYNCOPAL EPISODE	2015-12-10 17:40:01	LOWER SALFORD	MAIN ST & OLD SUMNEYTOWN PIKE	1
9	40.102398	-75.291458	BLUEROUTE & RAMP I476 NB TO CHEMICAL RD; PLYM...	19462.0	Traffic: VEHICLE ACCIDENT -	2015-12-10 17:40:01	PLYMOUTH	BLUEROUTE & RAMP I476 NB TO CHEMICAL RD	1

```
In [10]: zip = df['zip'].value_counts().head(5)
         zip
```

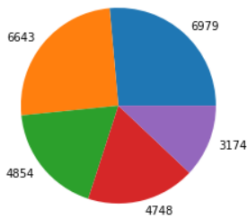
Here, I have created a pieplot for the zip column of the dataset. I wanted to analyze cases based on the zipcode.

```
In [10]: zip = df['zip'].value_counts().head(5)
         zip
```

```
Out[10]: 19401.0    6979
         19464.0    6643
         19403.0    4854
         19446.0    4748
         19406.0    3174
         Name: zip, dtype: int64
```

```
In [11]: plt.pie(x=zip, labels=zip)
```

```
Out[11]: ([<matplotlib.patches.Wedge at 0x291dcea9d00>,
           <matplotlib.patches.Wedge at 0x291dce9e220>,
           <matplotlib.patches.Wedge at 0x291dce9e6a0>,
           <matplotlib.patches.Wedge at 0x291dce9eb20>,
           <matplotlib.patches.Wedge at 0x291dce9efa0>],
          [Text(0.7419070087986036, 0.8121416072924159, '6979'),
           Text(-0.8484451578924782, 0.7001005742383075, '6643'),
           Text(-0.8564701100581407, -0.6902600601056071, '4854'),
           Text(0.2724380343627948, -1.0657286321726263, '4748'),
           Text(1.0224532159188744, -0.40569621794780364, '3174')])
```



Here, I am counting the number cases in a township and displaying the top 5 values and also counting the unique titles in the dataset.

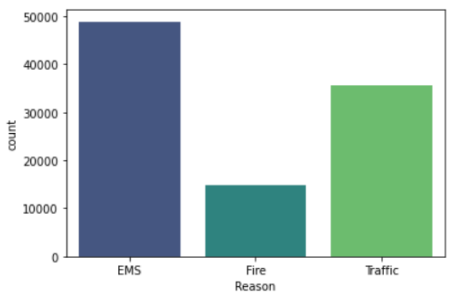
```
In [12]: df['twp'].value_counts().head(5)
Out[12]: LOWER MERION    8443
         ABINGTON        5977
         NORRISTOWN      5890
         UPPER MERION    5227
         CHELTENHAM      4575
         Name: twp, dtype: int64

In [13]: df['title'].nunique()
Out[13]: 110
```

Here, I am creating a new column Reason, and I'm storing the main reason for the 911 calls from the title column and storing them into the Reason column.

```
In [14]: df['Reason'] = df['title'].apply(lambda title: title.split(':')[0])

In [15]: df['Reason']
Out[15]: 0            EMS
         1            EMS
         2           Fire
         3            EMS
         4            EMS
                     ...
         99487    Traffic
         99488    Traffic
         99489        EMS
         99490        EMS
         99491    Traffic
         Name: Reason, Length: 99492, dtype: object

In [16]: y = df['Reason'].value_counts()
         y
Out[16]: EMS        48877
         Traffic    35695
         Fire       14920
         Name: Reason, dtype: int64
```

Here, I'm making a countplot based on the Reason column

```
In [17]: sns.countplot(x = 'Reason', data = df, palette = 'viridis')
Out[17]: <AxesSubplot:xlabel='Reason', ylabel='count'>
```



Here, I'm refactoring the timestamp column and converting it into a timestamp format and then accessing it I'm making three new columns into the dataset.

Reason

```
In [18]: type(df['timeStamp'].iloc[0])
Out[18]: str

In [19]: df['timeStamp'] = pd.to_datetime(df['timeStamp'])
         type(df['timeStamp'].iloc[0])
Out[19]: pandas._libs.tslibs.timestamps.Timestamp

In [20]: time = df['timeStamp'].iloc[0]
         time.hour
Out[20]: 17

In [21]: time.month
Out[21]: 12

In [22]: df['Hour'] = df['timeStamp'].apply(lambda time:time.hour)

In [23]: df['Month'] = df['timeStamp'].apply(lambda time:time.month)
         df['Day of week'] = df['timeStamp'].apply(lambda time:time.dayofweek)
```

I'm writing some code here to properly allocate the days of the week

```
In [24]: df.head()
Out[24]:
```

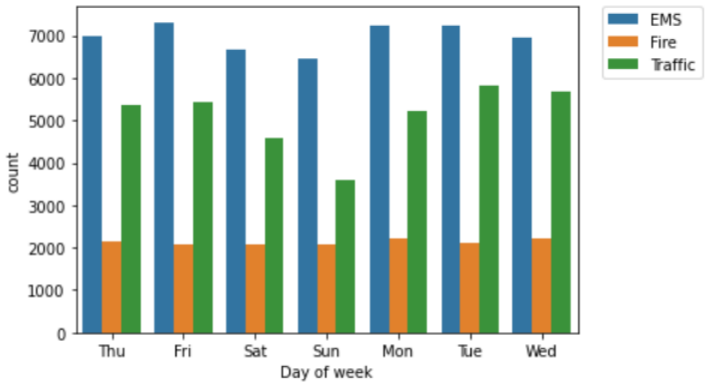
	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Month	Day of week
0	40.297876	-75.581294	REINDEER CT & DEAD END; NEW HANOVER; Station ...	19525.0	EMS: BACK PAINS/INJURY	2015-12-10 17:40:00	NEW HANOVER	REINDEER CT & DEAD END	1	EMS	17	12	3
1	40.258061	-75.264680	BRIAR PATH & WHITEMARSH LN; HATFIELD TOWNSHIP...	19446.0	EMS: DIABETIC EMERGENCY	2015-12-10 17:40:00	HATFIELD TOWNSHIP	BRIAR PATH & WHITEMARSH LN	1	EMS	17	12	3
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	3
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	3
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	3

```
In [25]: dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}

In [26]: df['Day of week'] = df['Day of week'].map(dmap)

In [27]: df.head()
Out[27]:
```

	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

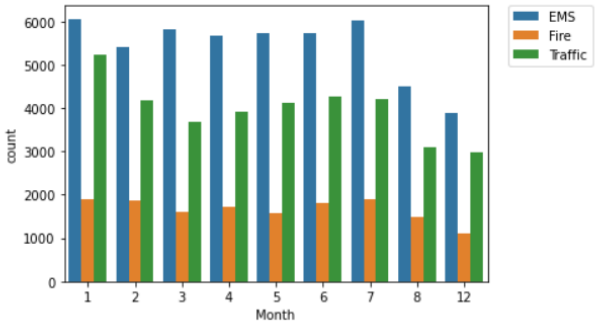
Here, I'm making a countplot based on the days of the week keeping Reason as hue, to understand the frequency of calls made on a given day of the week.

```
In [28]: sns.countplot(x = 'Day of week', data = df, hue = 'Reason' )
         plt.legend(bbox_to_anchor = (1.05,1), loc = 2, borderaxespad = 0.)
Out[28]: <matplotlib.legend.Legend at 0x291d63ce4c0>
```



Here, I'm making a countplot based on the month keeping Reason as hue to understand the frequency of calls made on a given month of the year.

```
In [29]: sns.countplot(x = 'Month', data = df, hue = 'Reason' )
         plt.legend(bbox_to_anchor = (1.05,1), loc = 2, borderaxespad = 0.)
Out[29]: <matplotlib.legend.Legend at 0x291dc4f6400>
```



Here, I'm grouping by months and plotting the graph to get an idea on the frequency of calls made in every month throughout the year.

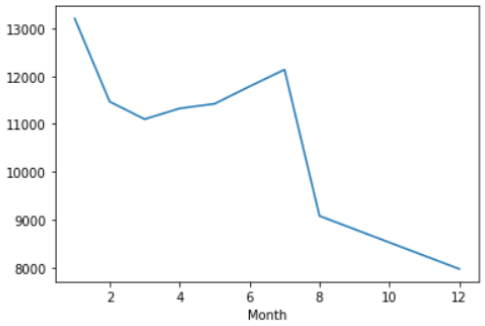
```
In [30]: byMonth = df.groupby('Month').count()
         byMonth.head()
```

Out[30]:

Month	lat	lng	desc	zip	title	timeStamp	twp	addr	e	Reason	Hour	Day of week
1	13205	13205	13205	11527	13205	13205	13203	13096	13205	13205	13205	13205
2	11467	11467	11467	9930	11467	11467	11465	11396	11467	11467	11467	11467
3	11101	11101	11101	9755	11101	11101	11092	11059	11101	11101	11101	11101
4	11326	11326	11326	9895	11326	11326	11323	11283	11326	11326	11326	11326
5	11423	11423	11423	9946	11423	11423	11420	11378	11423	11423	11423	11423

```
In [31]: byMonth['lat'].plot()
```

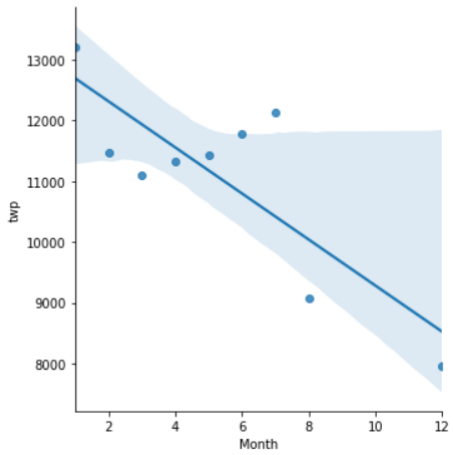
Out[31]: <AxesSubplot:xlabel='Month'>



Here, I'm making a lineplot between the month and the township to basically get an understanding on the total number of calls made from a township in a given month.


```
In [33]: sns.lmplot(x = 'Month', y = 'twp', data = byMonth.reset_index())
```

Out[33]: <seaborn.axisgrid.FacetGrid at 0x291dc656790>



Here, I'm making a date column from the timestamp column so that I can easily access the date separately to make plots.

In [34]:
```
t = df['timeStamp'].iloc[0]
```

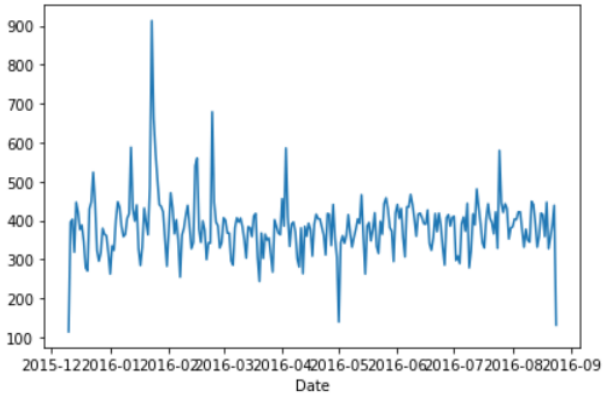
In [35]:
```
df['Date'] = df['timeStamp'].apply(lambda t : t.date())
df.head()
```

Out[35]:

	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

Here, I'm making a plot to get the understanding on the frequency of calls made during all days of the months from December 2015 to September 2016.

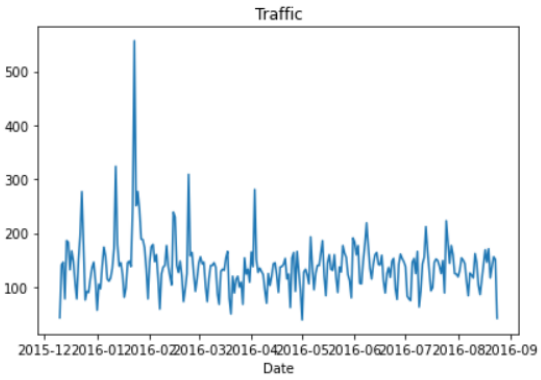
In [36]:
```
df.groupby('Date').count()['lat'].plot()
plt.tight_layout()
```



Here, I'm making a plot for the reason Traffic to get the understanding on the frequency of calls made during all days of the months from December 2015 to September 2016 related to traffic issues.

```
In [37]: df[df['Reason'] == 'Traffic'].groupby('Date').count()['lat'].plot()
         plt.tight_layout()
         plt.title('Traffic')
```

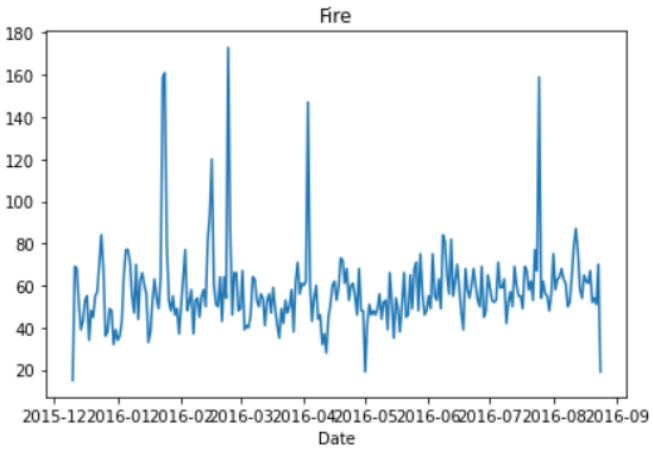
```
Out[37]: Text(0.5, 1.0, 'Traffic')
```



Here, I'm making a plot based on the reason Fire to get the understanding on the frequency of calls made during all days of the months from December 2015 to September 2016 related to fire issues.

```
In [38]: df[df['Reason'] == 'Fire'].groupby('Date').count()['lat'].plot()
         plt.tight_layout()
         plt.title('Fire')
```

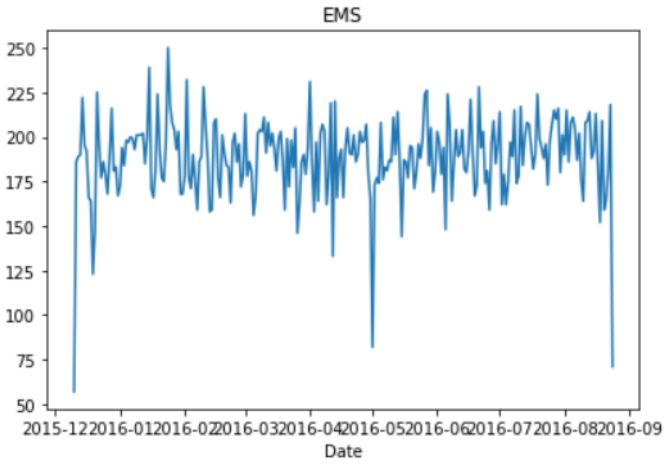
```
Out[38]: Text(0.5, 1.0, 'Fire')
```



Here, I'm making a plot based on the reason EMS to get the understanding on the frequency of calls made during all days of the months from December 2015 to September 2016 related to EMS issues.

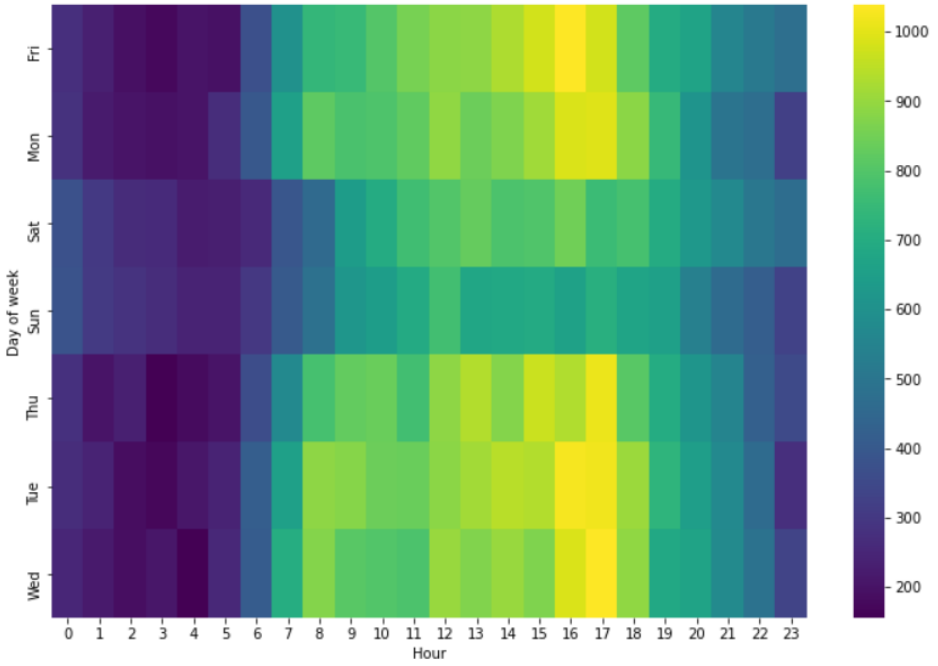
```
In [39]: df[df['Reason'] == 'EMS'].groupby('Date').count()['lat'].plot()
         plt.tight_layout()
         plt.title('EMS')
```

```
Out[39]: Text(0.5, 1.0, 'EMS')
```



Here, I'm making a heatmap based on hours and days of the week to get an understanding on the frequency of the calls made during a given day of the week and also understand the data regarding the number of calls made for a given hour of the day.

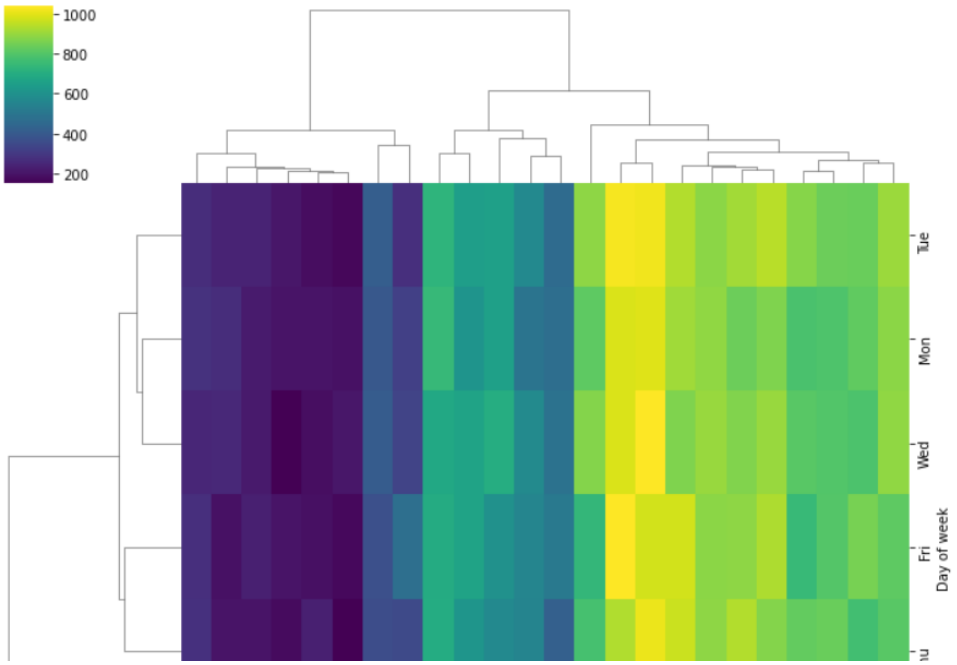
```
In [40]: dayhour = df.groupby(by=['Day of week','Hour']).count()['Reason'].unstack()

In [41]: plt.figure(figsize=(12,8))
         sns.heatmap(dayhour, cmap = 'viridis')

Out[41]: <AxesSubplot:xlabel='Hour', ylabel='Day of week'>
```

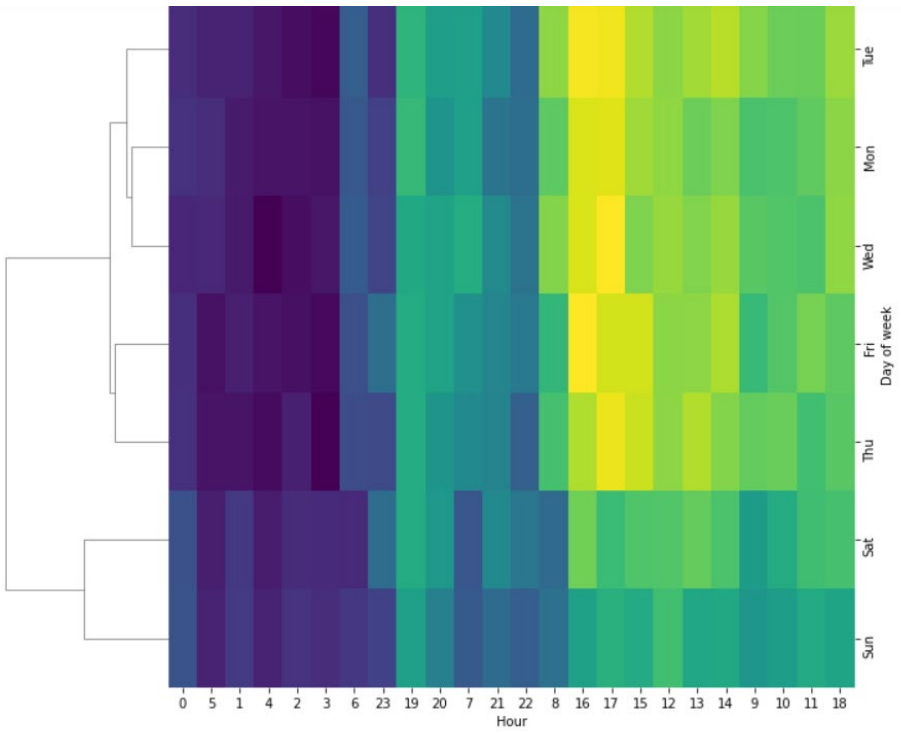


Here, I'm making a clustermap based on dayhour and Days of the week, to understand frequency of calls made in each of the 24 hours of the day for each of the seven days of the week.

In [42]:
```
sns.clustermap(dayhour, cmap = 'viridis')
```

Out[42]: <seaborn.matrix.ClusterGrid at 0x291dcd4d310>





```
In [43]: daymonth = df.groupby(by=['Day of week','Month']).count()['Reason'].unstack()
         daymonth.head()
```

Out[43]:

Day of week	Month 1	2	3	4	5	6	7	8	12
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

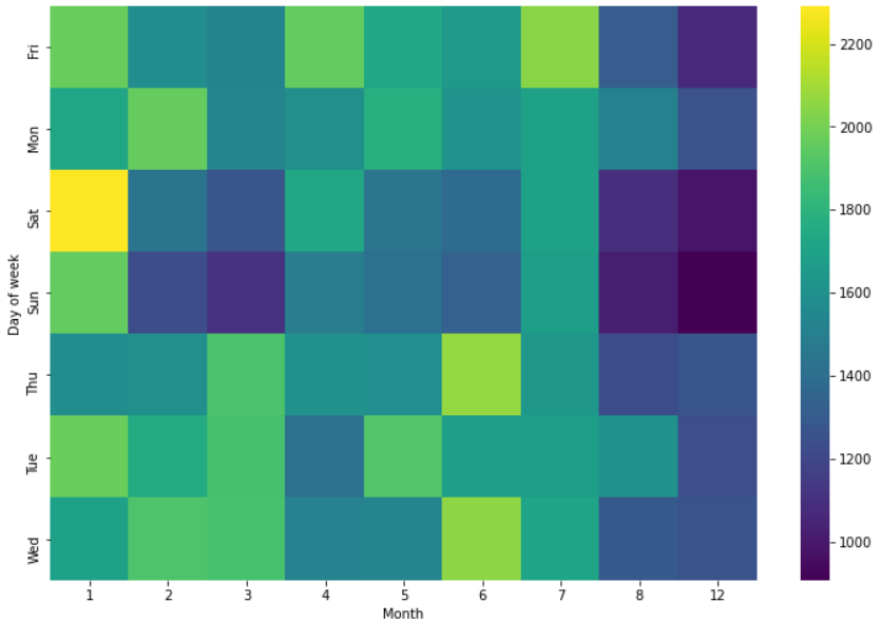
```
In [44]: plt.figure(figsize=(12,8))
         sns.heatmap(daymonth, cmap = 'viridis')
```

Out[44]: <AxesSubplot:xlabel='Month', ylabel='Day of week'>

Here, I'm making a heatmap based on the daymonth and the days of the week, so, I can understand the number of calls made on all the 7 days of the week for all the 12 months throughout the year.


```
In [44]: plt.figure(figsize=(12,8))
         sns.heatmap(daymonth, cmap = 'viridis')
```

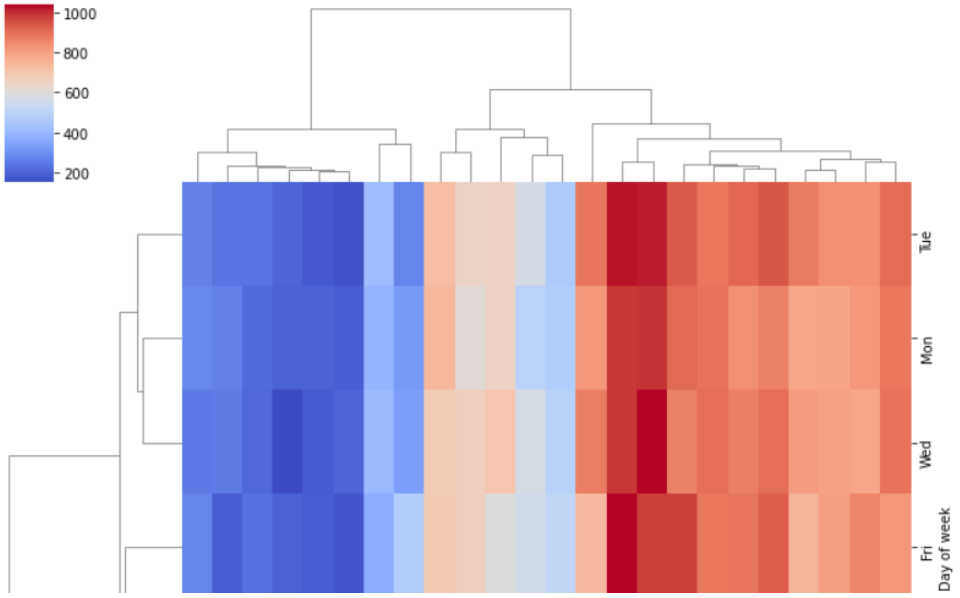
Out[44]: <AxesSubplot:xlabel='Month', ylabel='Day of week'>

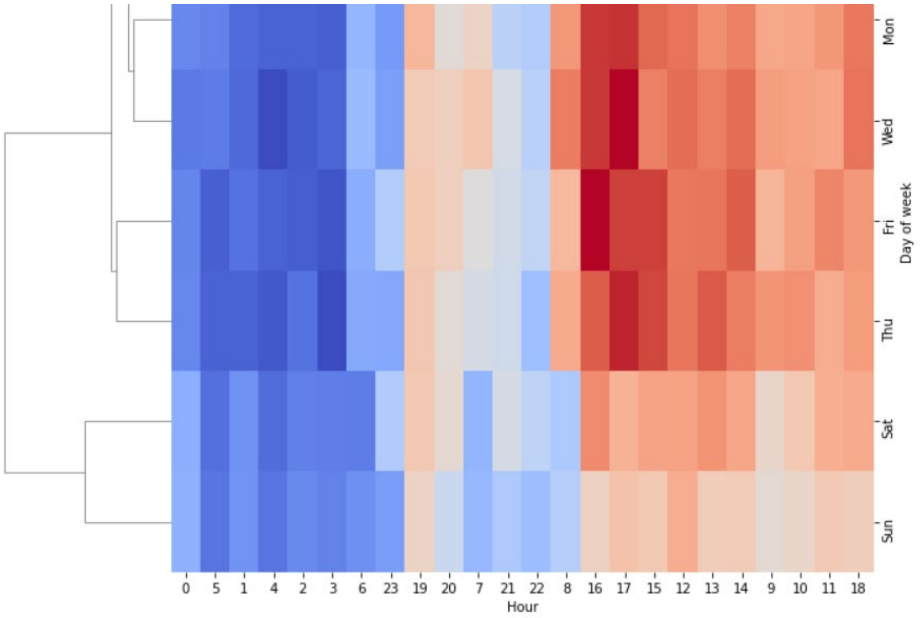


Here, I'm making a clustermap based on the dayhour and days of the week to get an idea about what hour of the day is the most occuring to make these calls and this data is plotted for all the seven days of the week.

```
In [45]: plt.figure(figsize=(12,8))
         sns.clustermap(dayhour, cmap = 'coolwarm')
```

```
Out[45]: <seaborn.matrix.ClusterGrid at 0x291dcf5af40>

         <Figure size 864x576 with 0 Axes>
```




Mon
Wed
Fri
Thu
Sat
Sun
Day of week
0 5 1 4 2 3 6 23 19 20 7 21 22 8 16 17 15 12 13 14 9 10 11 18
Hour