In [2]:

```
import os
import tqdm
import pickle
import random
import warnings
warnings.filterwarnings("ignore")
from datetime import timedelta

### Data Wrangling ###
import numpy as np
import pandas as pd

### Data Visualization ###
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import timedelta
```

In [67]:

```
data=pd.read_csv('Train.csv')
data.head(3).append(data.tail(3))
```

Out[67]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
0	2012-10-02 09:00:00	None	288.28	0.0	0.0	40	Clouds	scattered clouds	5545
1	2012-10-02 10:00:00	None	289.36	0.0	0.0	75	Clouds	broken clouds	4516
2	2012-10-02 11:00:00	None	289.58	0.0	0.0	90	Clouds	overcast clouds	4767
38560	2017-11-01 19:00:00	None	274.62	0.0	0.0	90	Mist	mist	3045
38561	2017-11-01 19:00:00	None	274.62	0.0	0.0	90	Rain	light rain	3045
38562	2017-11-01 20:00:00	None	274.75	0.0	0.0	90	Drizzle	light intensity drizzle	2704

DATA IS FROM 2012-10-02 TO 2017-11-01

#

Exploratory data analysis

In [4]:

```
import sweetviz as sv
my_report = sv.analyze(data)
my_report.show_notebook()
```

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38563 entries, 0 to 38562
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   date_time            38563 non-null  object 
 1   holiday              38563 non-null  object 
 2   temp                 38563 non-null  float64
 3   rain_1h              38563 non-null  float64
 4   snow_1h              38563 non-null  float64
 5   clouds_all           38563 non-null  int64  
 6   weather_main         38563 non-null  object 
 7   weather_description  38563 non-null  object 
 8   traffic_volume       38563 non-null  int64  
dtypes: float64(3), int64(2), object(4)
memory usage: 2.6+ MB
```

In [68]:

```
data['date_time']=pd.to_datetime(data['date_time'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38563 entries, 0 to 38562
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   date_time            38563 non-null  datetime64[ns]
 1   holiday              38563 non-null  object
 2   temp                 38563 non-null  float64
 3   rain_1h              38563 non-null  float64
 4   snow_1h              38563 non-null  float64
 5   clouds_all           38563 non-null  int64
 6   weather_main         38563 non-null  object
 7   weather_description  38563 non-null  object
 8   traffic_volume       38563 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(2), object(3)
memory usage: 2.6+ MB
```

In [69]:

```
print(data['date_time'].min(),data['date_time'].max())
```

```
2012-10-02 09:00:00 2017-11-01 20:00:00
```

we are having data in the span of 6 years

In [70]:

```
data.shape
```

Out[70]:

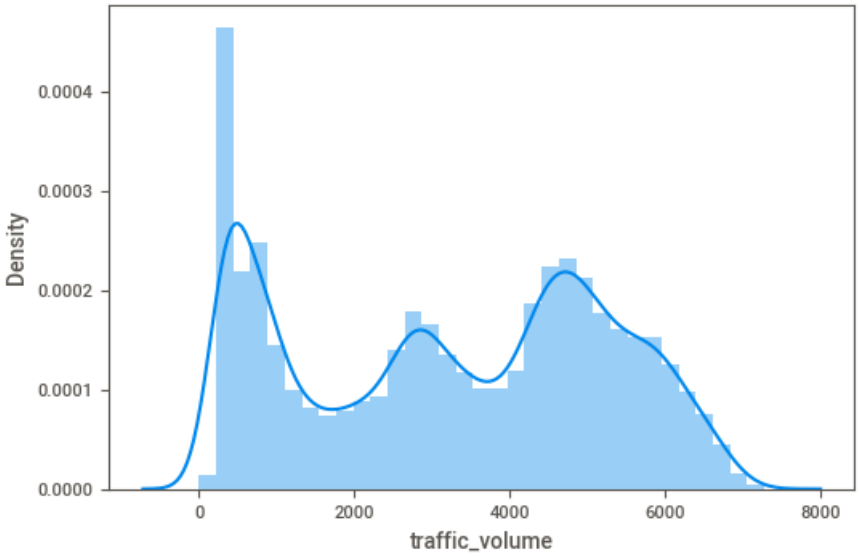
```
(38563, 9)
```

In [9]:

```
sns.distplot(data['traffic_volume'],kde=True)
```

Out[9]:

```
<AxesSubplot:xlabel='traffic_volume', ylabel='Density'>
```



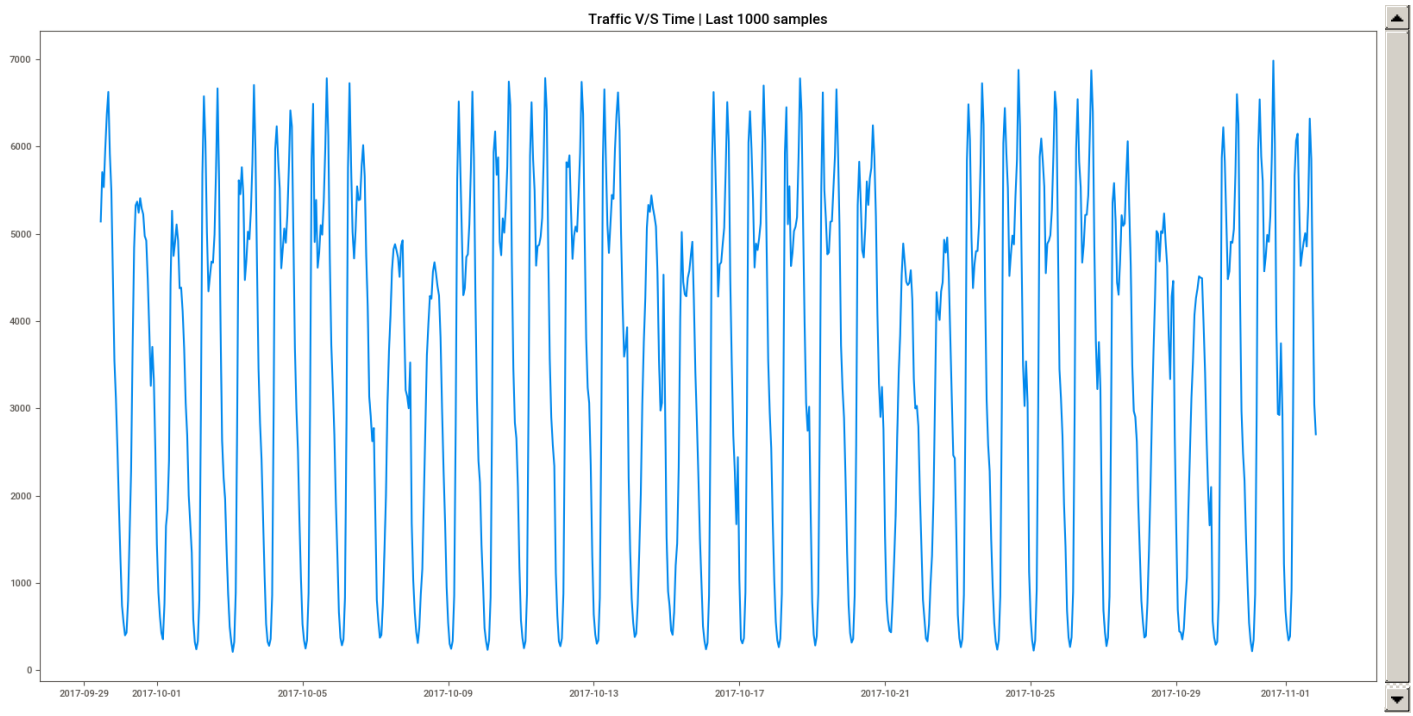
from the above dist plot we can observe that the data is not normally distributed, so we have to Sandardize it for training

In [10]:

```
import datetime
```

In [13]:

```
fig, ax = plt.subplots(figsize=(20,10))
last = 1000
sns.lineplot(x=data['date_time'].values[-last:],y=data['traffic_volume'].values[-last:])
plt.title(f'Traffic V/S Time | Last {last} samples')
plt.show()
```



from the Last few samples we can observe a that there are a lot of fluctuations in traffic volume with respect to time

In [12]:

```
data['holiday'].nunique()
```

Out[12]:

```
12
```

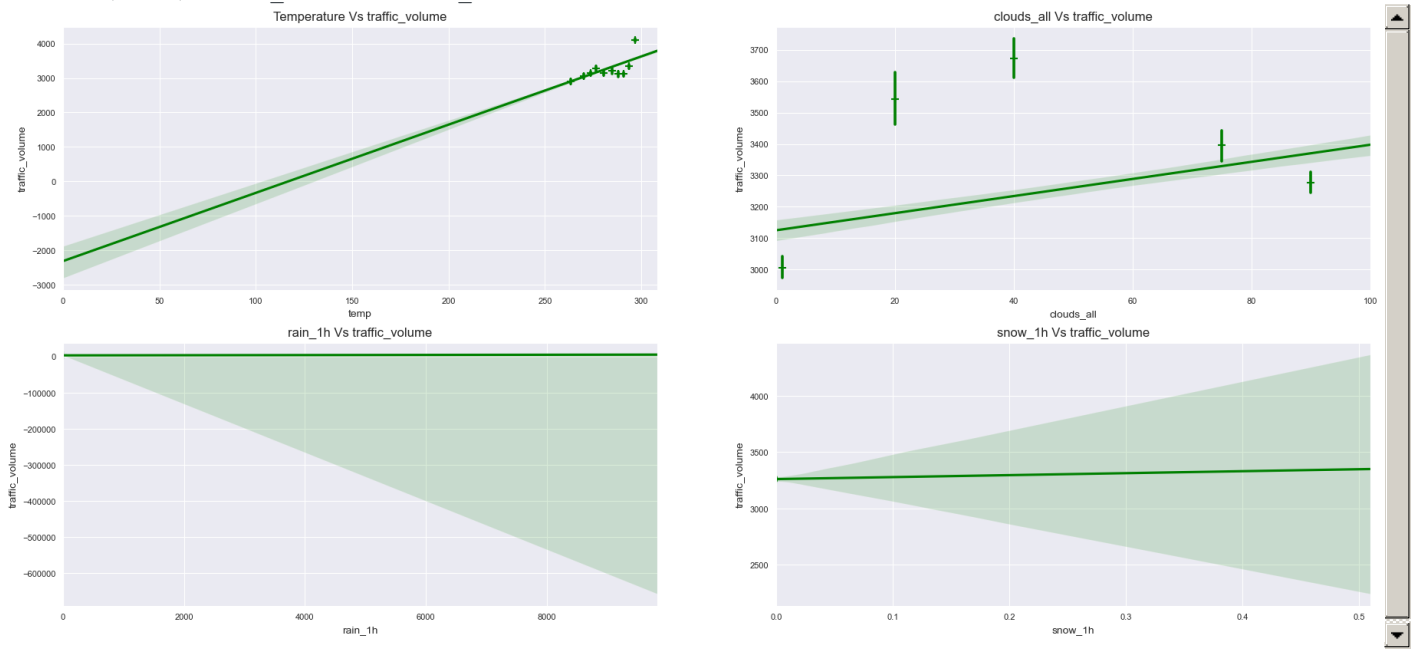
Lets plot some regression plots to understand how features are behaving with respect to traffic volume

In [14]:

```
sns.set_style("darkgrid")
fig,ax = plt.subplots(2,2,figsize = (22,10))
sns.regplot(x = 'temp',y = 'traffic_volume',data = data,marker = '+',color = 'g',x_bins = 10,ax=ax[0,0])
ax[0,0].set_title("Temperature Vs traffic_volume")
sns.regplot(x = 'clouds_all',y = 'traffic_volume',data = data,marker = '+',color = 'g',ax=ax[0,1],x_bins =
ax[0,1].set_title("clouds_all Vs traffic_volume")
sns.regplot(x = 'rain_1h',y = 'traffic_volume',data = data,marker = '+',color = 'g',ax=ax[1,0],x_bins = 20
ax[1,0].set_title("rain_1h Vs traffic_volume")
sns.regplot(x = 'snow_1h',y = 'traffic_volume',data = data,marker = '+',color = 'g',ax=ax[1,1],x_bins = 20
ax[1,1].set_title("snow_1h Vs traffic_volume")
```

Out[14]:

```
Text(0.5, 1.0, 'snow_1h Vs traffic_volume')
```



As we observe some strange behaviour of rain_1h and snow_1h, we can scatter plot them with respect to traffic volume to understand them more closely

In [15]:

```
data['snow_1h'].nunique()
```

Out[15]:

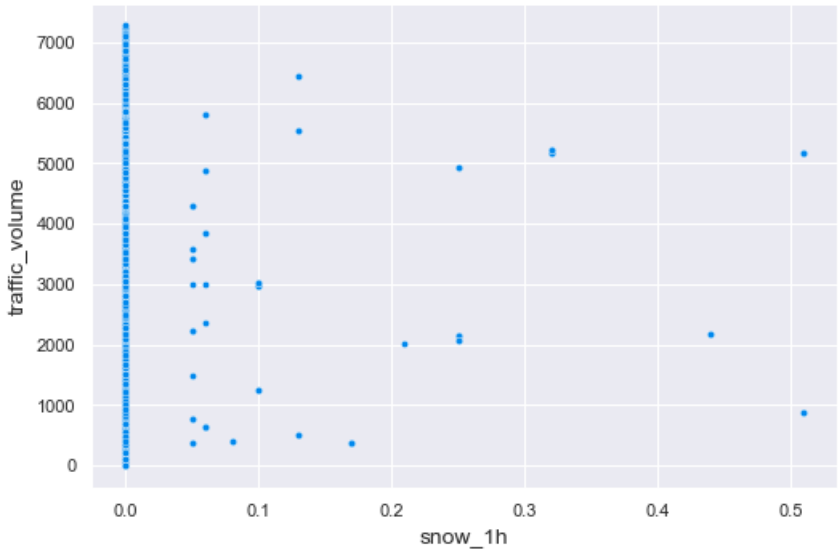
```
12
```

In [16]:

```
sns.scatterplot(data=data, x="snow_1h", y="traffic_volume")
```

Out[16]:

```
<AxesSubplot:xlabel='snow_1h', ylabel='traffic_volume'>
```

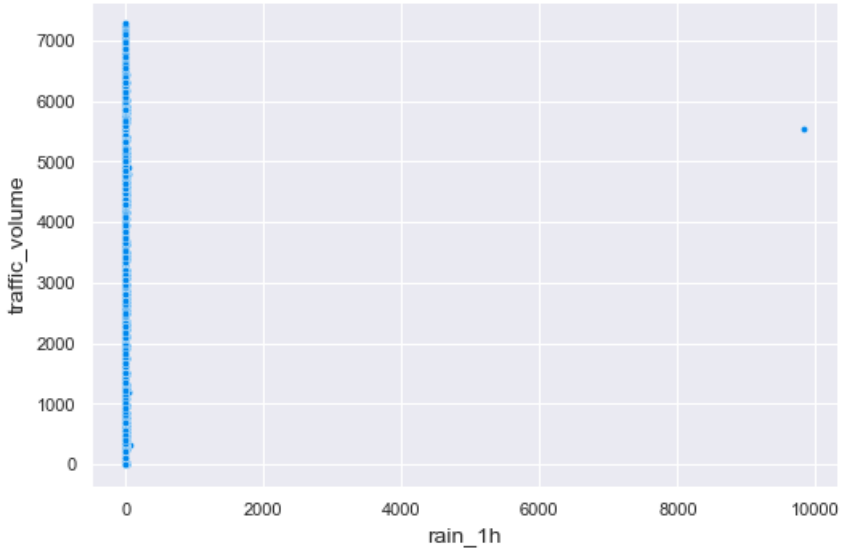


In [17]:

```
sns.scatterplot(data=data, x="rain_1h", y="traffic_volume")
```

Out[17]:

```
<AxesSubplot:xlabel='rain_1h', ylabel='traffic_volume'>
```



In [18]:

```
data.holiday.value_counts()
```

Out[18]:

```
None                          38515
Labor Day                         5
Columbus Day                      5
Christmas Day                     5
Thanksgiving Day                  5
New Years Day                     5
Independence Day                  4
Veterans Day                      4
Memorial Day                      4
State Fair                        4
Washingtons Birthday              4
Martin Luther King Jr Day         3
Name: holiday, dtype: int64
```

In [71]:

```
data[data['holiday']!='None']
```

Out[71]:

date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
126	2012-10-08	Columbus Day	273.080	0.00	0.0	20	Clouds	few clouds	455
1123	2012-11-12	Veterans Day	288.120	0.00	0.0	87	Clear	sky is clear	1000
1370	2012-11-22	Thanksgiving Day	278.540	0.00	0.0	20	Mist	mist	919
2360	2012-12-25	Christmas Day	264.400	0.00	0.0	90	Clouds	overcast clouds	803
2559	2013-01-01	New Years Day	263.490	0.00	0.0	58	Clouds	broken clouds	1439
3697	2013-02-18	Washingtons Birthday	258.960	0.00	0.0	20	Clouds	few clouds	556
6430	2013-05-27	Memorial Day	286.370	0.00	0.0	90	Clouds	overcast clouds	863
7414	2013-07-04	Independence Day	290.080	0.00	0.0	1	Clear	sky is clear	1060
8575	2013-08-22	State Fair	297.420	0.00	0.0	12	Clouds	few clouds	661
8742	2013-09-02	Labor Day	288.780	0.00	0.0	0	Clear	Sky is Clear	1041
9455	2013-10-14	Columbus Day	277.720	0.00	0.0	0	Clear	Sky is Clear	615
9768	2013-11-11	Veterans Day	275.440	0.00	0.0	64	Clouds	broken clouds	514
10181	2013-11-28	Thanksgiving Day	268.240	0.00	0.0	64	Clouds	broken clouds	929
10939	2013-12-25	Christmas Day	260.170	0.25	0.0	64	Rain	light rain	712
11132	2014-01-01	New Years Day	250.140	0.00	0.0	90	Clouds	overcast clouds	1395
11611	2014-01-20	Martin Luther King Jr Day	271.790	0.00	0.0	64	Clouds	broken clouds	480
12303	2014-02-17	Washingtons Birthday	267.440	0.00	0.0	90	Clouds	overcast clouds	583
14645	2014-05-26	Memorial Day	294.310	0.00	0.0	40	Clouds	scattered clouds	967
16204	2015-07-03	Independence Day	289.200	0.00	0.0	1	Clear	sky is clear	959
17750	2015-08-27	State Fair	287.970	0.00	0.0	1	Clear	sky is clear	605
18041	2015-09-07	Labor Day	295.020	0.00	0.0	90	Clouds	overcast clouds	973
18946	2015-10-12	Columbus Day	293.020	0.00	0.0	1	Clear	sky is clear	494
19410	2015-11-11	Veterans Day	283.350	0.00	0.0	75	Clouds	broken clouds	559
19675	2015-11-26	Thanksgiving Day	277.220	0.00	0.0	90	Mist	mist	833
19676	2015-11-26	Thanksgiving Day	277.220	0.00	0.0	90	Haze	haze	833
20185	2015-12-25	Christmas Day	269.090	0.00	0.0	90	Snow	light snow	894
20344	2016-01-01	New Years Day	265.940	0.00	0.0	90	Haze	haze	1513
20345	2016-01-01	New Years Day	265.940	0.00	0.0	90	Snow	light snow	1513
21294	2016-02-15	Washingtons Birthday	265.350	0.00	0.0	90	Mist	mist	785
23781	2016-05-30	Memorial Day	288.380	0.00	0.0	1	Mist	mist	1082
24654	2016-07-04	Independence Day	289.950	0.00	0.0	1	Clear	sky is clear	1115
26002	2016-08-25	State Fair	290.820	0.00	0.0	1	Clear	sky is clear	655
26319	2016-09-05	Labor Day	293.170	1.52	0.0	90	Rain	moderate rain	1064
26320	2016-09-05	Labor Day	293.170	1.52	0.0	90	Thunderstorm	proximity thunderstorm	1064
27224	2016-10-10	Columbus Day	282.341	0.00	0.0	0	Clear	Sky is Clear	484
27983	2016-11-11	Veterans Day	281.960	0.00	0.0	1	Clear	sky is clear	572
28399	2016-11-24	Thanksgiving Day	274.340	0.00	0.0	75	Mist	mist	763
29503	2016-12-26	Christmas Day	276.080	0.00	0.0	90	Rain	light rain	732
29504	2016-12-26	Christmas Day	276.080	0.00	0.0	90	Mist	mist	732
29674	2017-01-02	New Years Day	270.620	0.00	0.0	90	Snow	light snow	798
30080	2017-01-16	Martin Luther King Jr Day	266.080	0.00	0.0	1	Mist	mist	698
30081	2017-01-16	Martin Luther King Jr Day	266.080	0.00	0.0	1	Haze	haze	698
31145	2017-02-20	Washingtons Birthday	285.059	0.00	0.0	68	Clouds	broken clouds	629
34095	2017-05-29	Memorial Day	285.870	0.00	0.0	40	Clouds	scattered clouds	1538
35057	2017-07-04	Independence Day	293.410	0.00	0.0	1	Clear	sky is clear	1225

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
36551	2017-08-24	State Fair	289.690	0.00	0.0	90	Rain	light rain	657
36882	2017-09-04	Labor Day	295.540	0.00	0.0	1	Clear	sky is clear	1026
37882	2017-10-09	Columbus Day	284.620	0.00	0.0	1	Clear	sky is clear	549

In [72]:

```
isholiday=data.holiday!='None'
date=data.date_time[isholiday].dt.date.values
holiday=data.holiday[isholiday].values
mapping=dict(zip(date,holiday))
data['holiday']=data['date_time'].dt.date.map(mapping)
data.loc[data['holiday'].isna(),'holiday'] = 'Work Day'
data['holiday'].value_counts()
```

Out[72]:

```
Work Day                    37462
Labor Day                     114
Columbus Day                  112
Thanksgiving Day              111
New Years Day                 107
Memorial Day                  106
Christmas Day                 104
State Fair                     97
Independence Day               96
Veterans Day                   96
Washingtons Birthday           95
Martin Luther King Jr Day      63
Name: holiday, dtype: int64
```

In [73]:

```
data['holiday'].map(lambda x: 'Holiday' if x != 'Work Day' else 'Work Day').value_counts(normalize=True)
```

Out[73]:

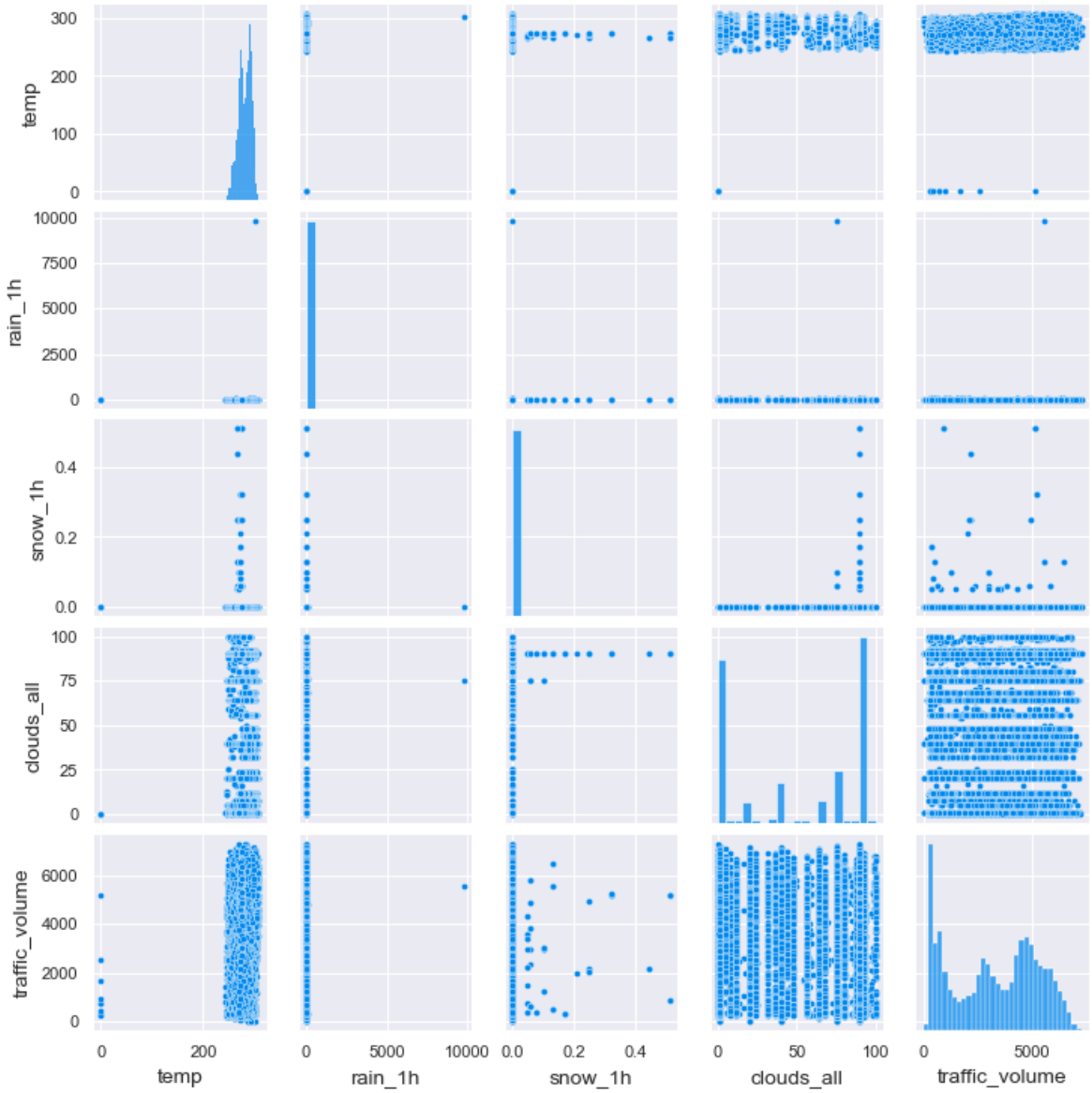
```
Work Day    0.971449
Holiday     0.028551
Name: holiday, dtype: float64
```

In [22]:

```
column = ['temp','rain_1h','snow_1h','clouds_all','traffic_volume']
sns.pairplot(data[column],height = 1.5)
```

Out[22]:

```
<seaborn.axisgrid.PairGrid at 0x1d2a4a61a30>
```



In [74]:

```
data.head()
```

Out[74]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918

In [24]:

```
data['clouds_all'].nunique()
```

Out[24]:

```
60
```

In [75]:

```
data['year'] = pd.DatetimeIndex(data['date_time']).year
data.head()
```

Out[75]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	2012
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	2012
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	2012

In [76]:

```
data['day'] = data.date_time.dt.day
data['month'] = data.date_time.dt.month_name()
data['weekday'] = data.date_time.dt.day_name()
data['hour'] = data.date_time.dt.hour
```

In [83]:

```
data.head()
```

Out[83]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	day	month	weekday
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	2	October	Tuesday
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012	2	October	Tuesday
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	2012	2	October	Tuesday
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	2012	2	October	Tuesday
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	2012	2	October	Tuesday

In [78]:

```
data['month'].unique()
```

Out[78]:

```
array(['October', 'November', 'December', 'January', 'February', 'March',
       'April', 'May', 'June', 'July', 'August', 'September'],
      dtype=object)
```

In [84]:

```
data['weather_main'].unique()
```

Out[84]:

```
array(['Clouds', 'Clear', 'Rain', 'Drizzle', 'Mist', 'Haze', 'Fog',
       'Thunderstorm', 'Snow', 'Squall', 'Smoke'], dtype=object)
```

In [85]:

```
data.shape
```

Out[85]:

```
(38563, 14)
```

In [86]:

```
data.head()
```

Out[86]:

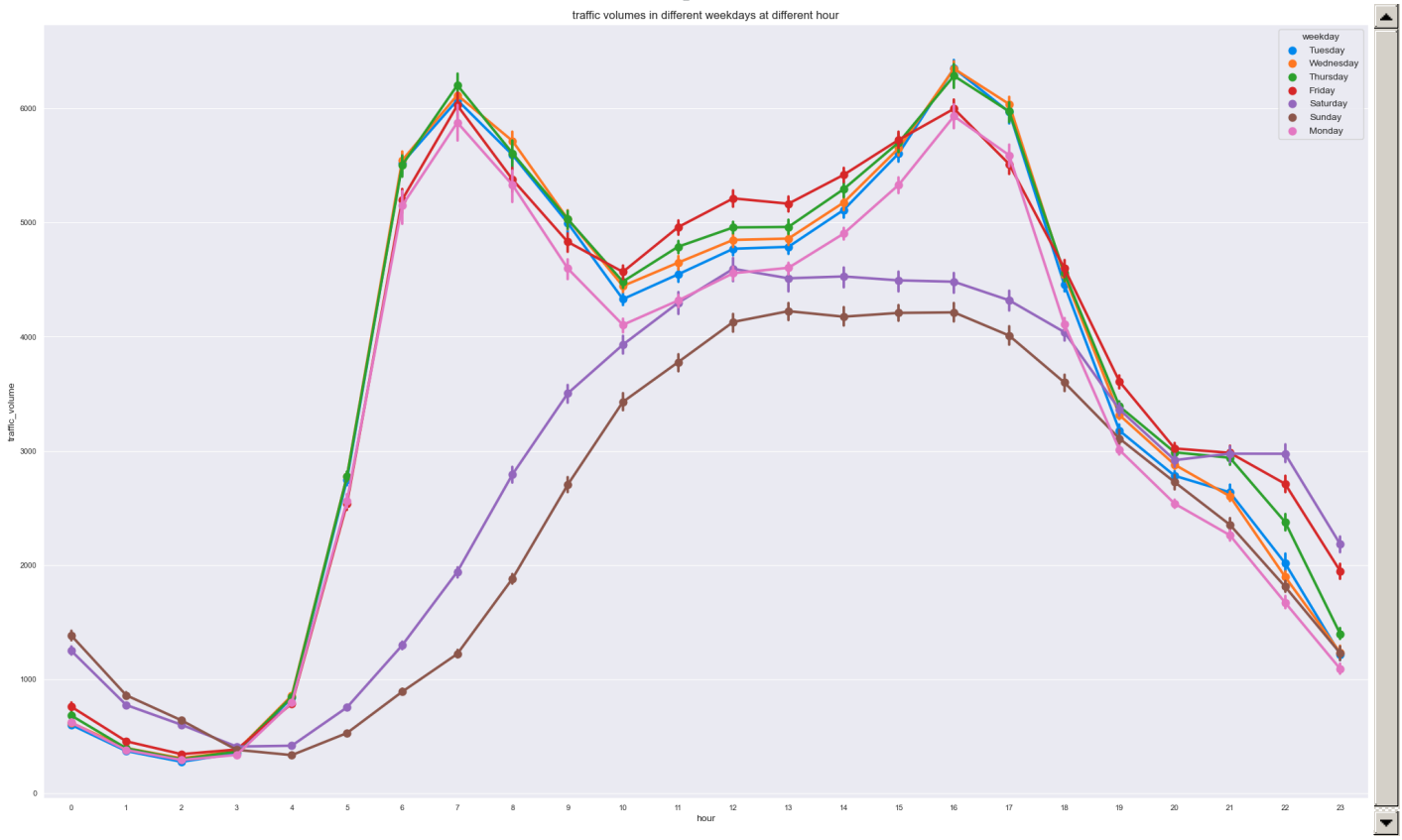
	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	day	month	weekday
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	2	October	Tuesday
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012	2	October	Tuesday
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	2012	2	October	Tuesday
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	2012	2	October	Tuesday
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	2012	2	October	Tuesday

In [90]:

```
plt.figure(figsize = (25,15))
sns.pointplot(x=  'hour', y = 'traffic_volume', hue = 'weekday' , data = data)
plt.title("traffic volumes in different weekdays at different hour")
```

Out[90]:

```
Text(0.5, 1.0, 'traffic volumes in different weekdays at different hour')
```



From Above plot we can observe that traffic volume During: monday,friday=Moderate : Tuesday,wednesday,thursday=High : Saturday,Sunday=Low

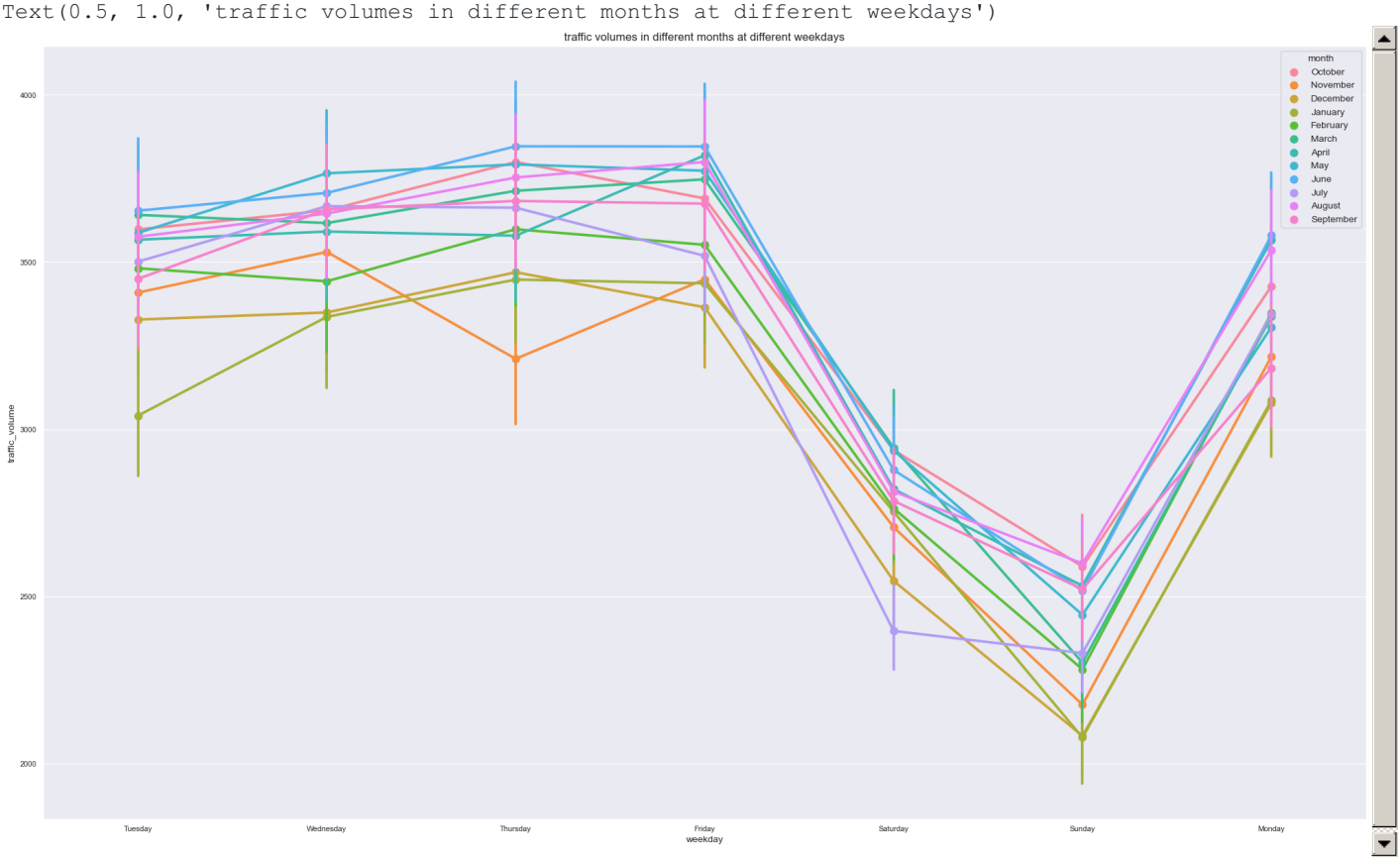
And the traffic volume is high during THE HOUR 5-8 AND 15-18

In [91]:

```
plt.figure(figsize = (25,15))
sns.pointplot(x=  'weekday', y = 'traffic_volume', hue = 'month' , data = data)
plt.title("traffic volumes in different months at different weekdays")
```

Out[91]:

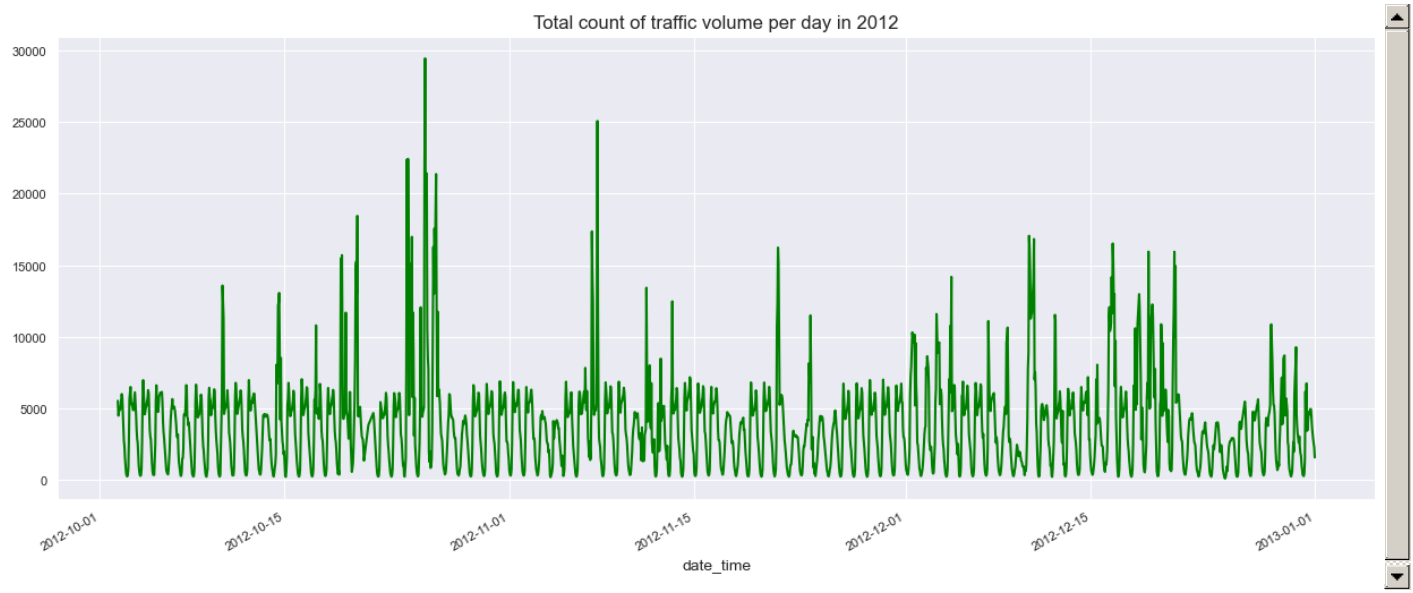
```
Text(0.5, 1.0, 'traffic volumes in different months at different weekdays')
```



FROM ABOVE PLOT WE CAN OBSERVE THAT IN EVERY MONTH DURING SATURDAY,SUNDAY THE TRAFFIC VOLUME IS VERY LOW.AND IN MONTHS JUNE,MAY,MARCH,SEPTEMBER,APRIL THE TRAFFIC VOLUME IS HIGH

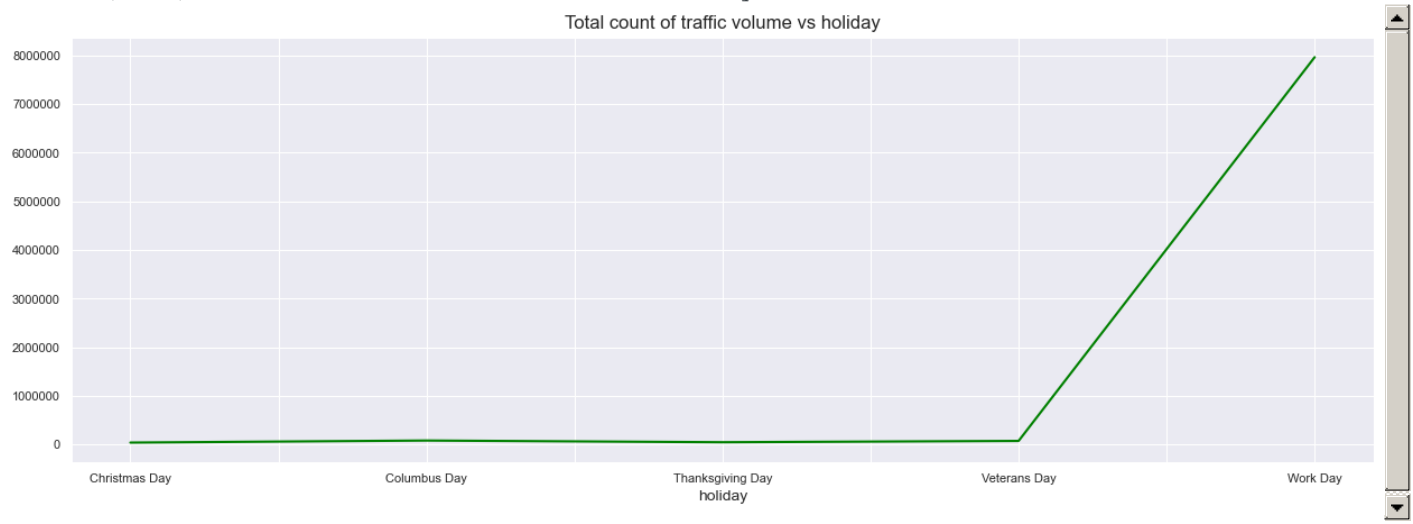
In [94]:

```
y_2012 = data.loc[data['year']==2012]
y_2012.groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,6),color = 'g')
plt.title("Total count of traffic volume per day in 2012")
plt.show()
y_2012.groupby('holiday')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,5),color ='g')
plt.title("Total count of traffic volume vs holiday")
```



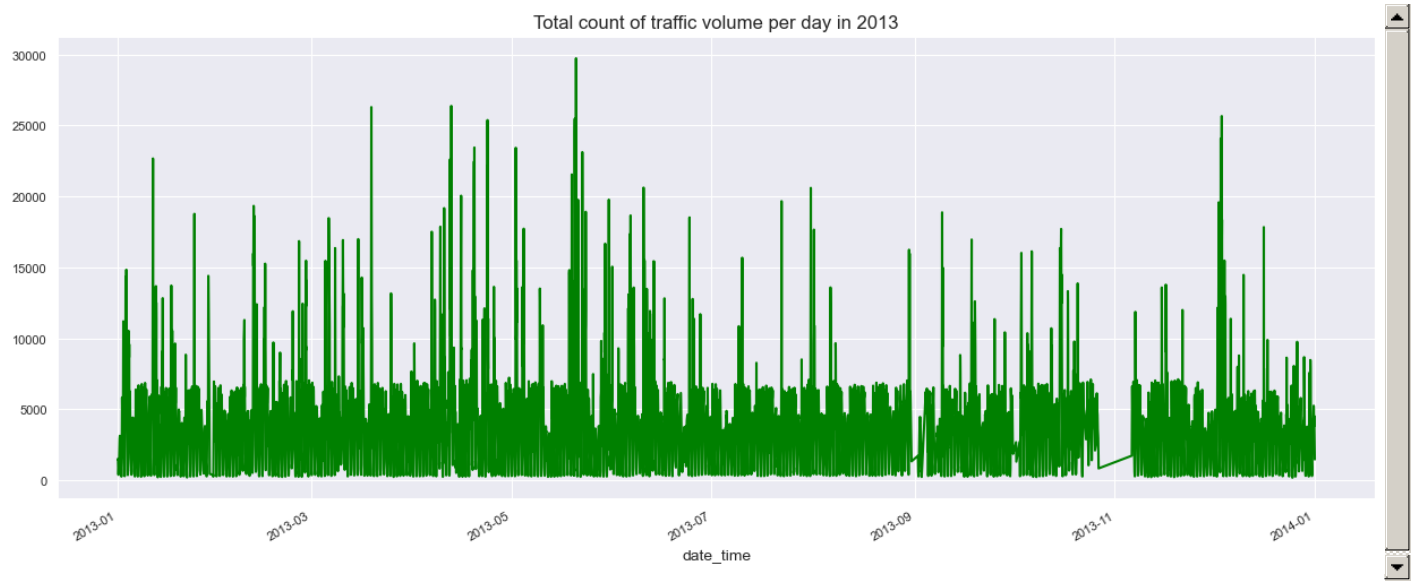
Out[94]:

```
Text(0.5, 1.0, 'Total count of traffic volume vs holiday')
```



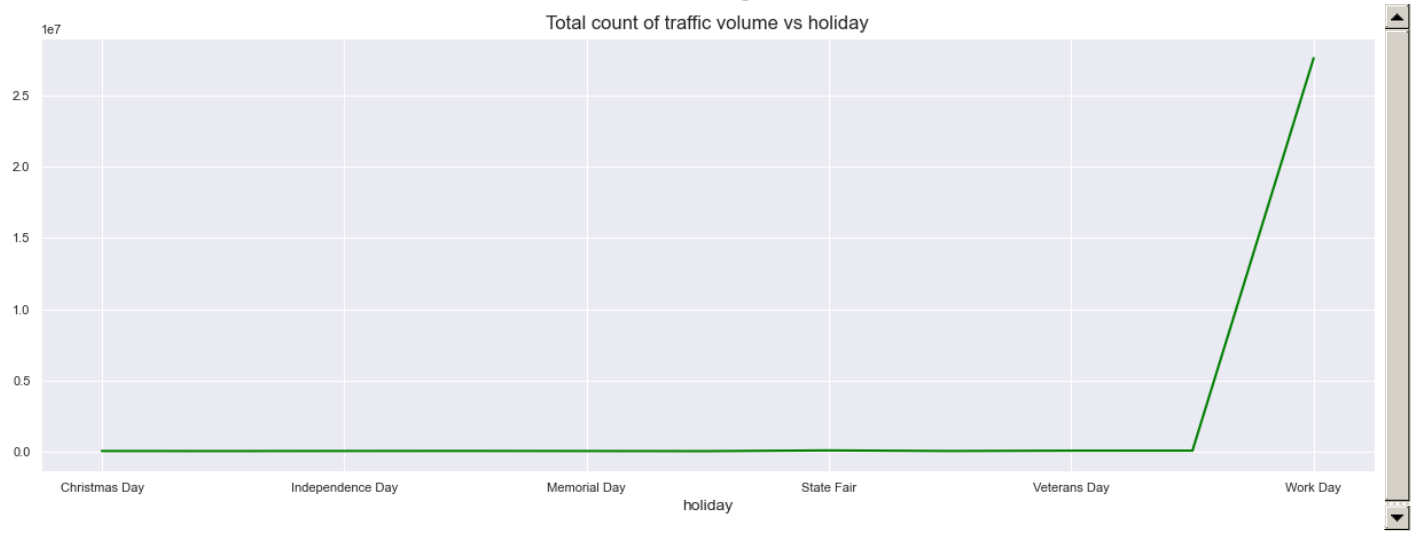
In [96]:

```
y_2013 = data.loc[data['year']==2013]
y_2013.groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,6),color = 'g')
plt.title("Total count of traffic volume per day in 2013")
plt.show()
y_2013.groupby('holiday')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,5),color ='g')
plt.title("Total count of traffic volume vs holiday")
```



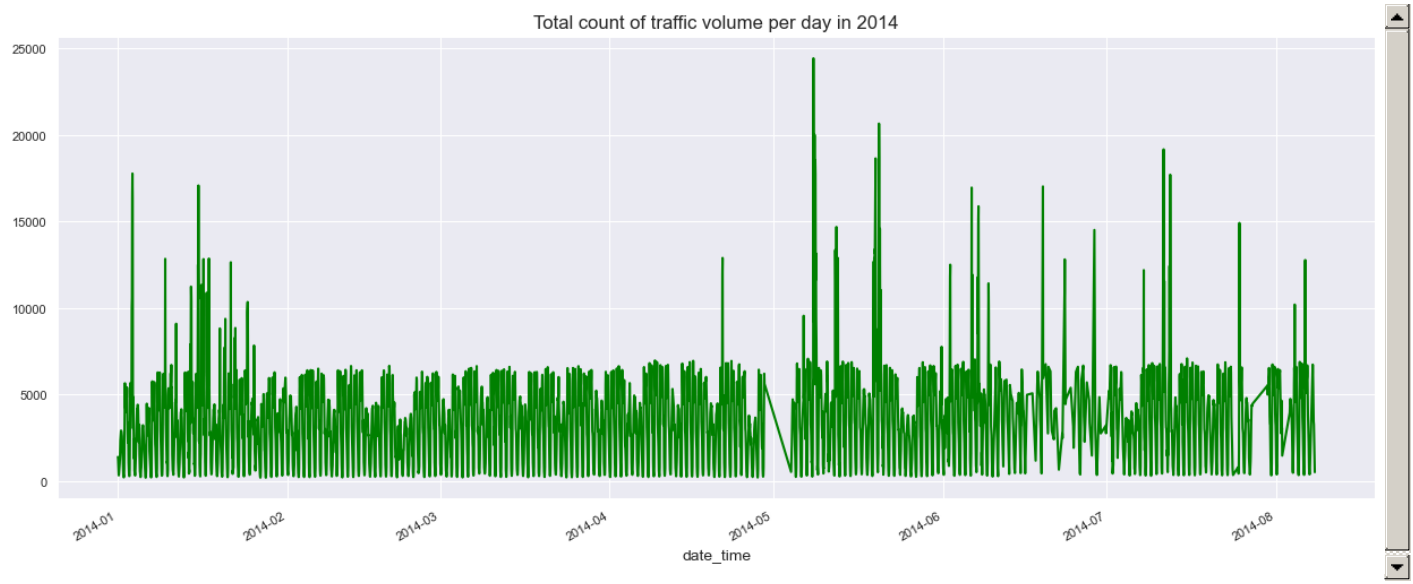
Out[96]:

```
Text(0.5, 1.0, 'Total count of traffic volume vs holiday')
```



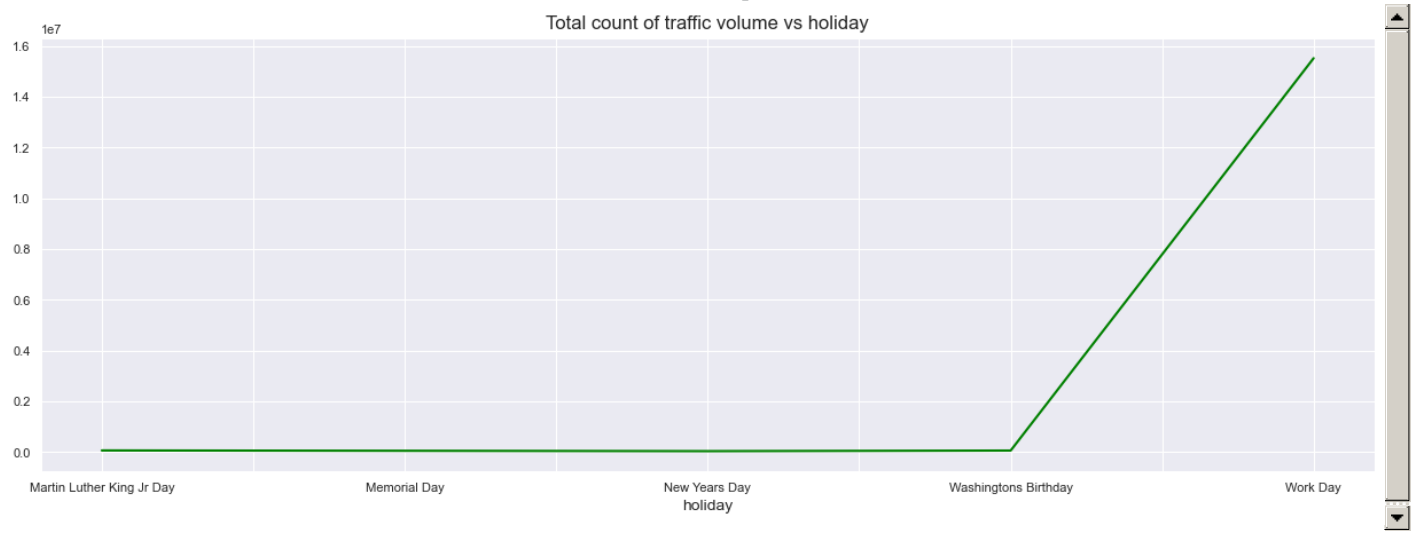
In [97]:

```
y_2014 = data.loc[data['year']==2014]
y_2014.groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,6),color = 'g')
plt.title("Total count of traffic volume per day in 2014")
plt.show()
y_2014.groupby('holiday')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,5),color ='g')
plt.title("Total count of traffic volume vs holiday")
```



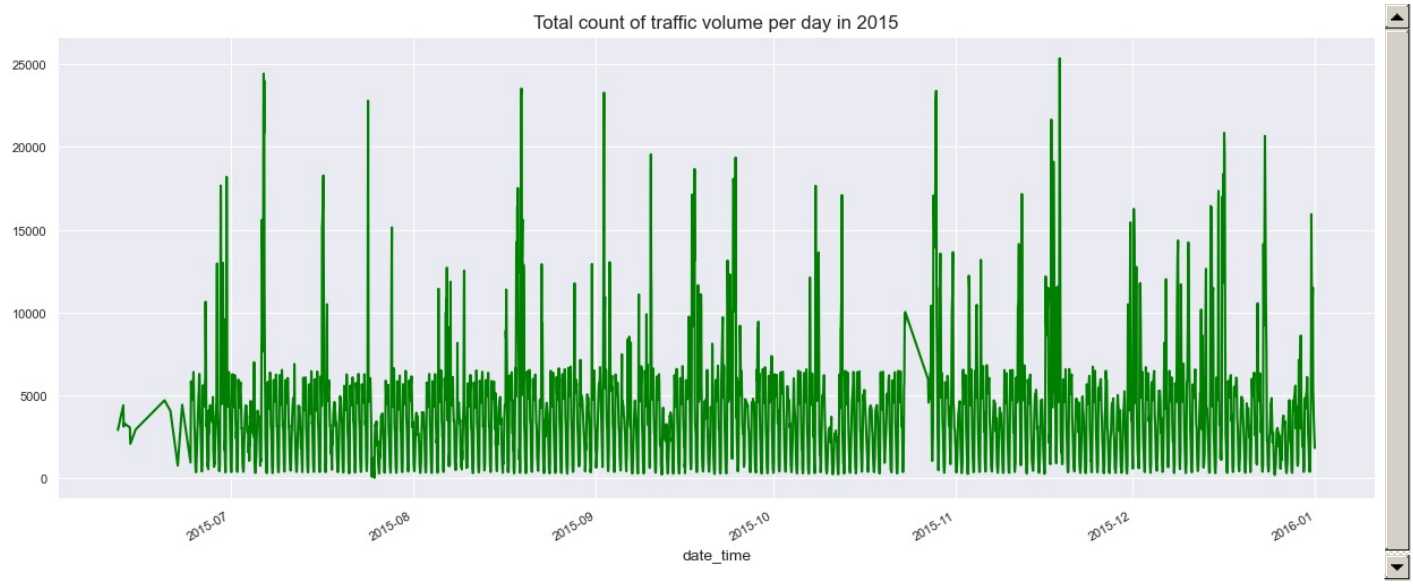
Out[97]:

```
Text(0.5, 1.0, 'Total count of traffic volume vs holiday')
```



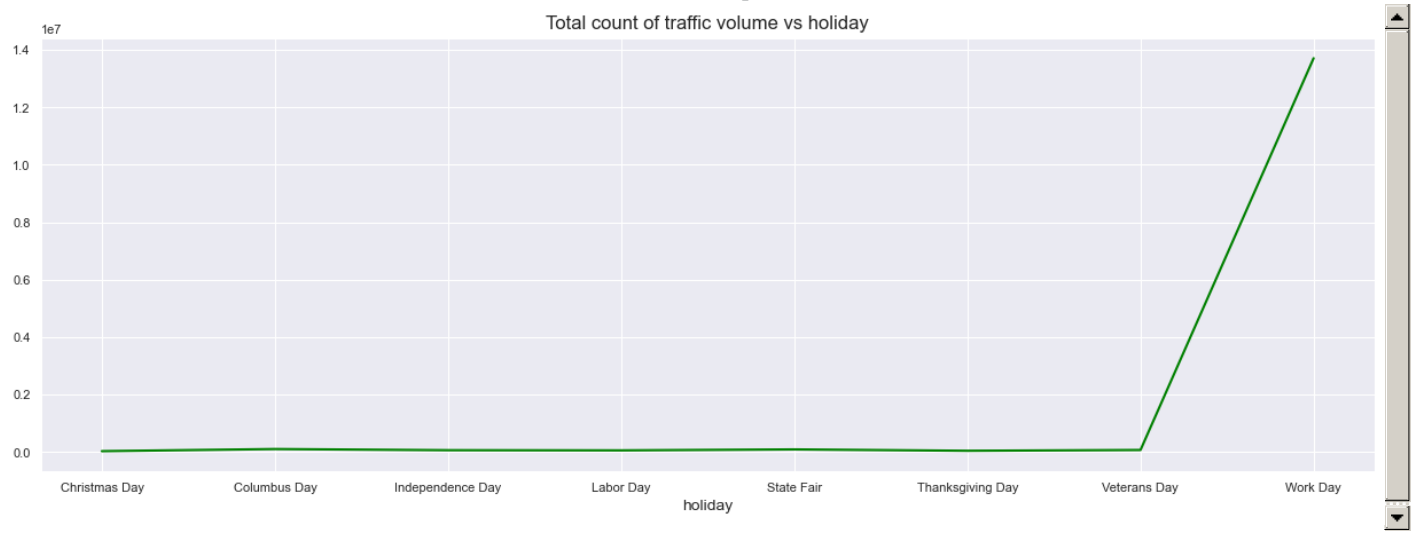
In [98]:

```
y_2015 = data.loc[data['year']==2015]
y_2015.groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,6),color = 'g')
plt.title("Total count of traffic volume per day in 2015")
plt.show()
y_2015.groupby('holiday')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,5),color ='g')
plt.title("Total count of traffic volume vs holiday")
```



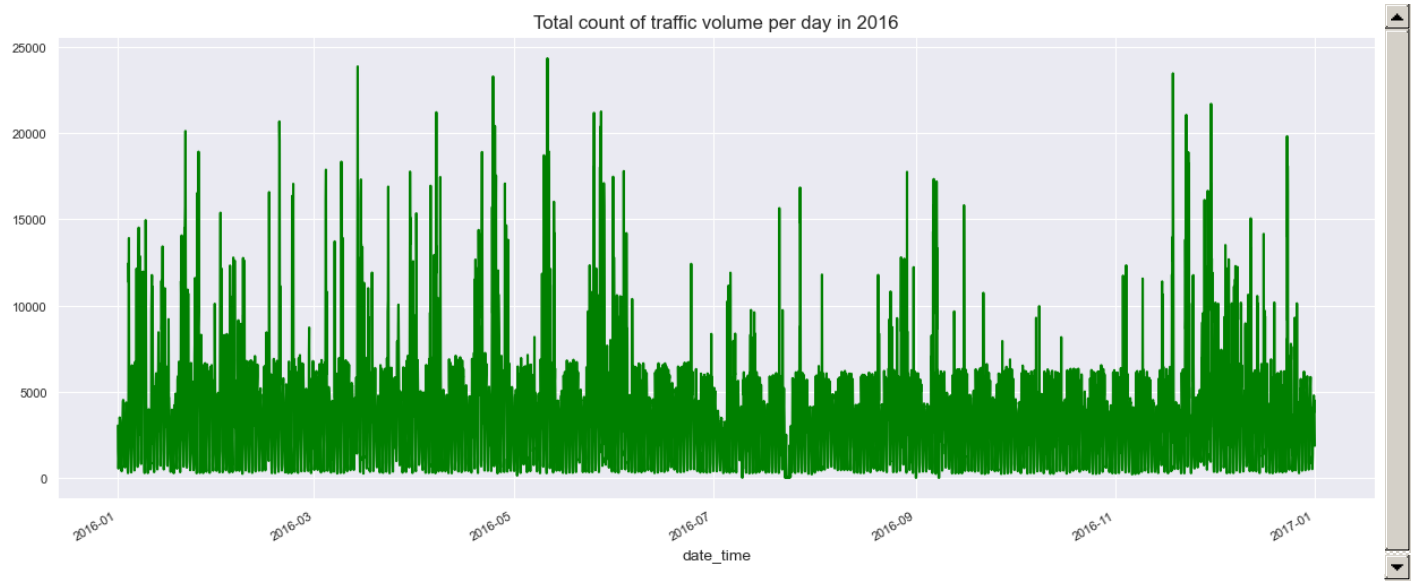
Out[98]:

```
Text(0.5, 1.0, 'Total count of traffic volume vs holiday')
```



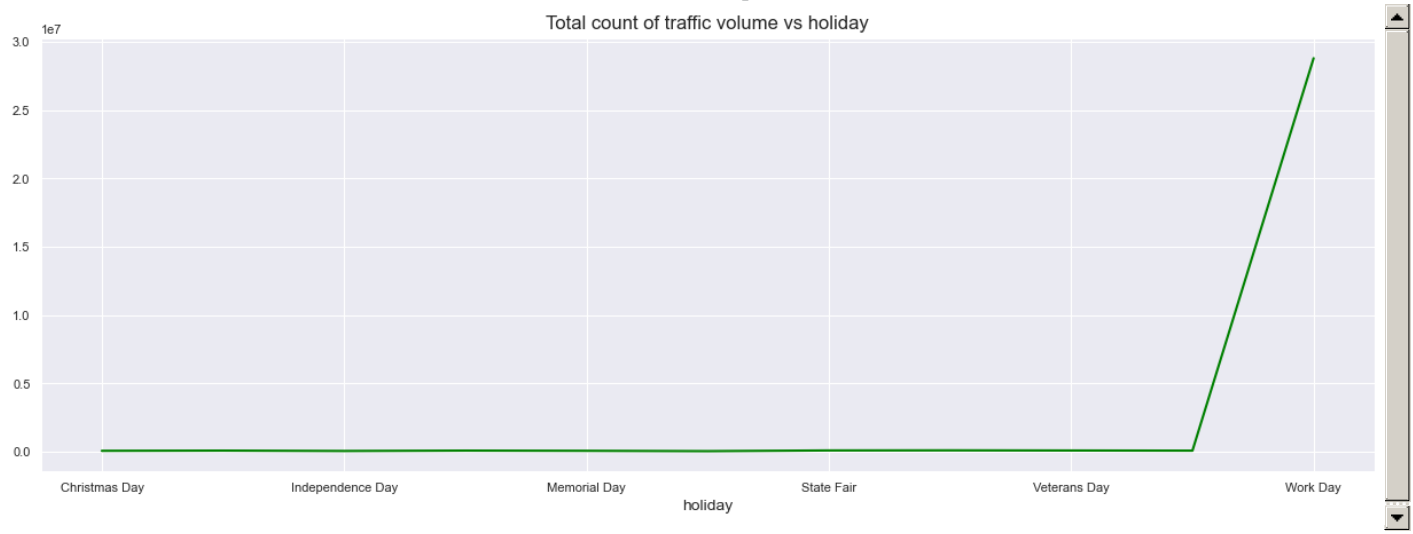
In [99]:

```
y_2016 = data.loc[data['year']==2016]
y_2016.groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,6),color = 'g')
plt.title("Total count of traffic volume per day in 2016")
plt.show()
y_2016.groupby('holiday')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,5),color ='g')
plt.title("Total count of traffic volume vs holiday")
```



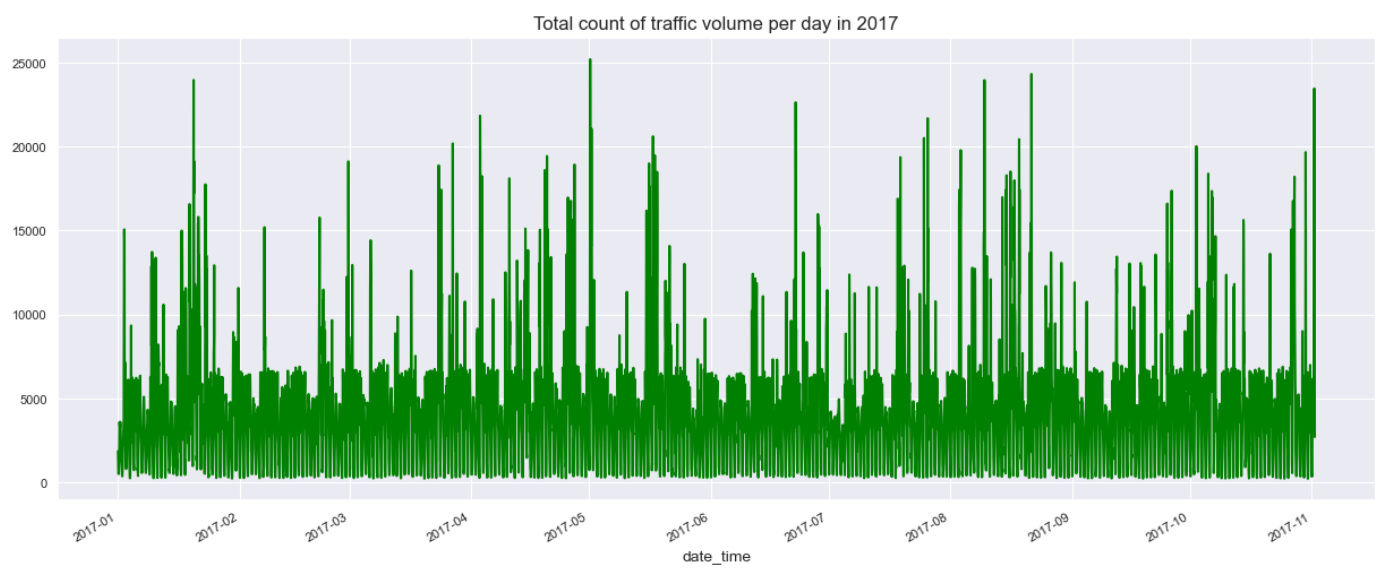
Out[99]:

```
Text(0.5, 1.0, 'Total count of traffic volume vs holiday')
```



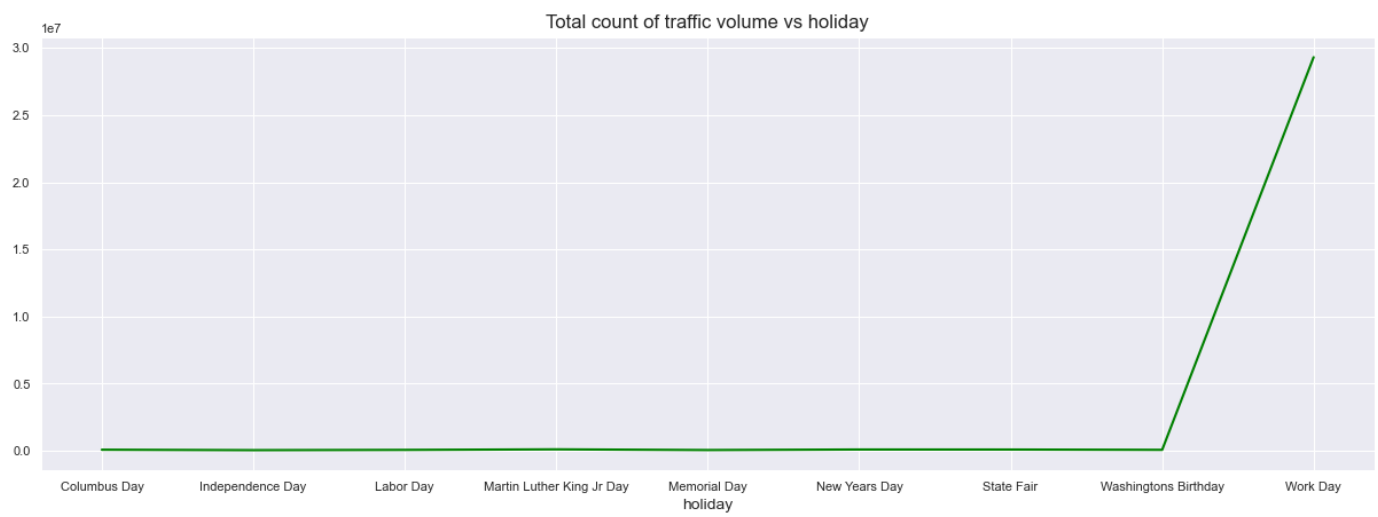
In [100]:

```
y_2017 = data.loc[data['year']==2017]
y_2017.groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,6),color = 'g')
plt.title("Total count of traffic volume per day in 2017")
plt.show()
y_2017.groupby('holiday')['traffic_volume'].sum().plot(kind = 'line',figsize = (15,5),color ='g')
plt.title("Total count of traffic volume vs holiday")
```



Out[100]:

```
Text(0.5, 1.0, 'Total count of traffic volume vs holiday')
```

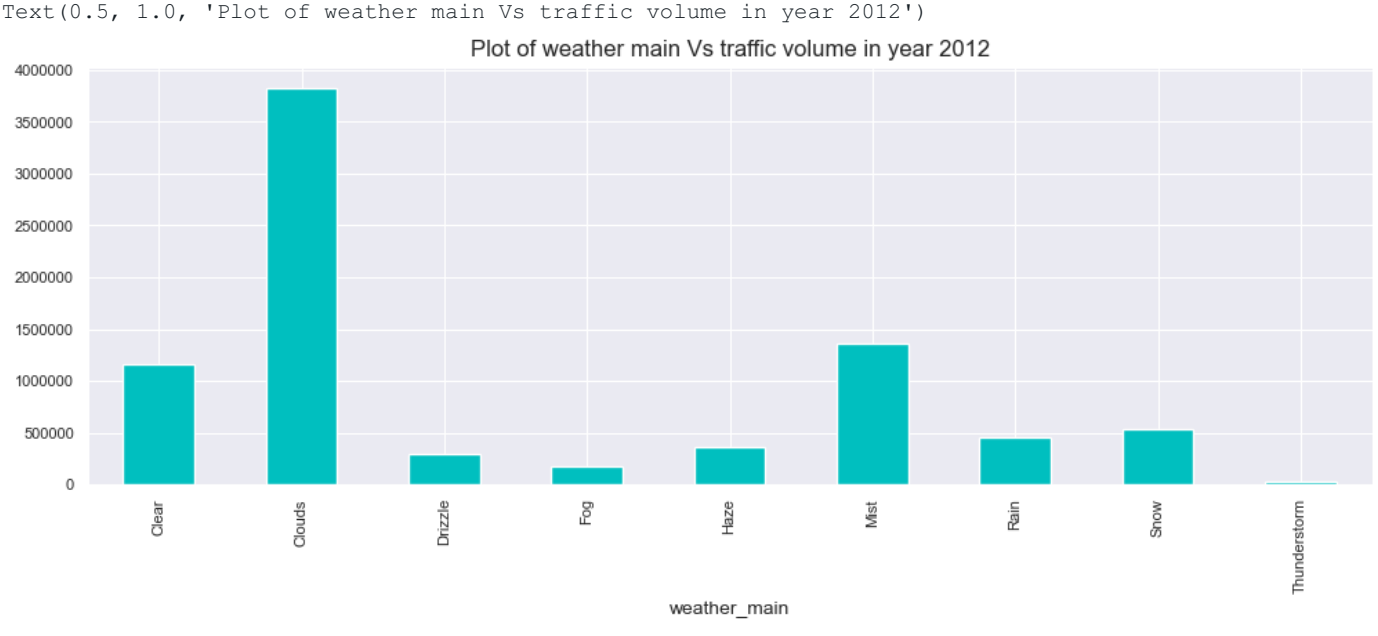


In [101]:

```
y_2012.groupby('weather_main')['traffic_volume'].sum().plot(kind = 'bar',color = 'c',figsize= (12,4))
plt.title("Plot of weather main Vs traffic volume in year 2012")
```

Out[101]:

```
Text(0.5, 1.0, 'Plot of weather main Vs traffic volume in year 2012')
```

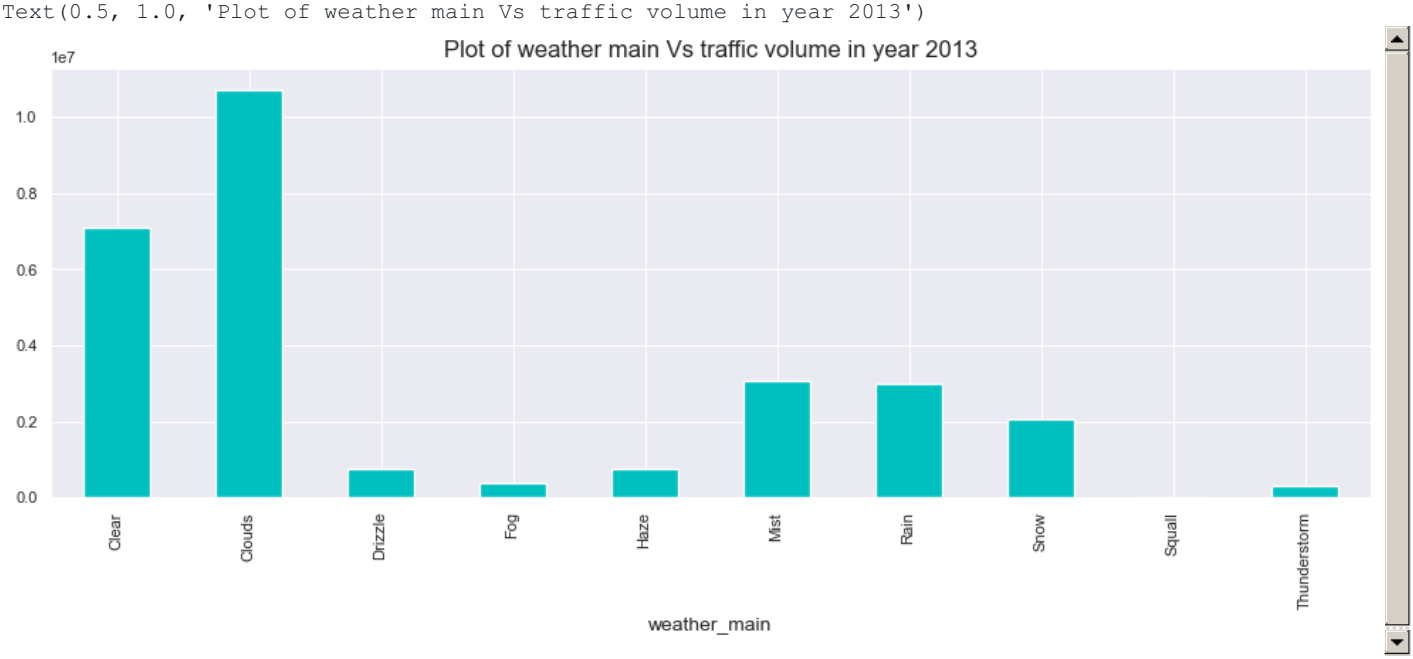


In [102]:

```
y_2013.groupby('weather_main')['traffic_volume'].sum().plot(kind = 'bar',color = 'c',figsize= (12,4))
plt.title("Plot of weather main Vs traffic volume in year 2013")
```

Out[102]:

```
Text(0.5, 1.0, 'Plot of weather main Vs traffic volume in year 2013')
```

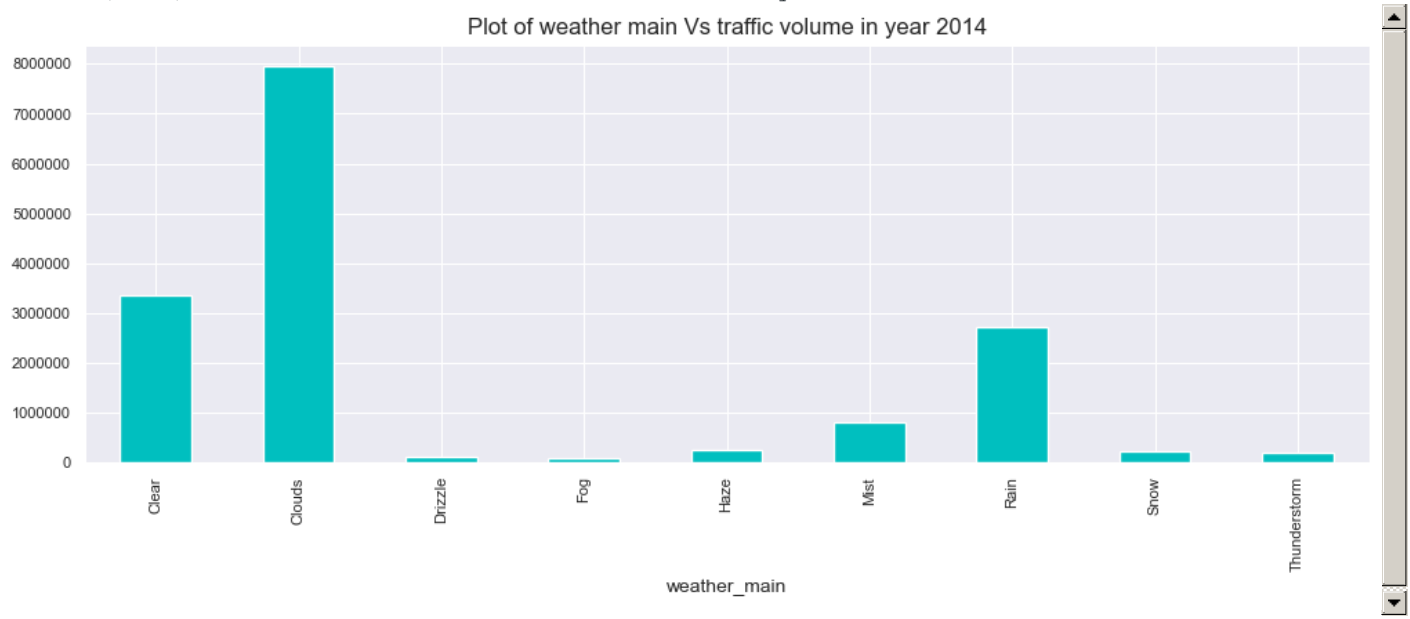


In [103]:

```
y_2014.groupby('weather_main')['traffic_volume'].sum().plot(kind = 'bar',color = 'c',figsize= (12,4))
plt.title("Plot of weather main Vs traffic volume in year 2014")
```

Out[103]:

```
Text(0.5, 1.0, 'Plot of weather main Vs traffic volume in year 2014')
```

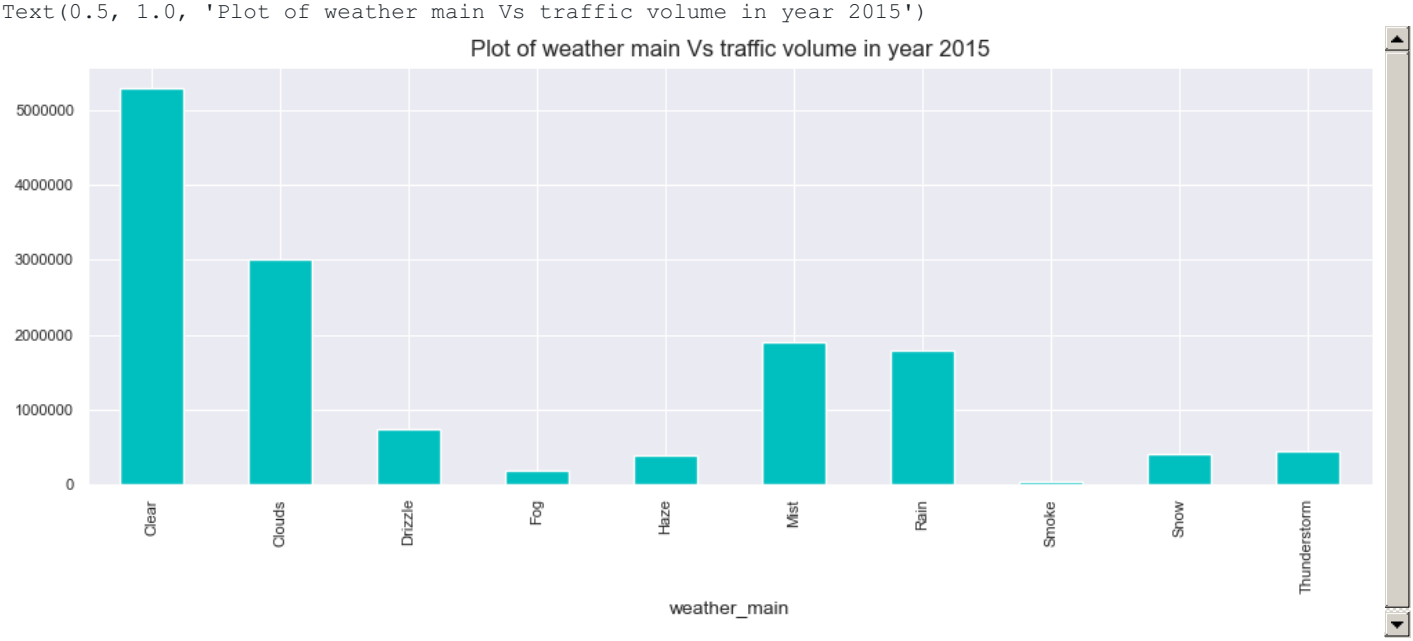


In [104]:

```
y_2015.groupby('weather_main')['traffic_volume'].sum().plot(kind = 'bar',color = 'c',figsize= (12,4))
plt.title("Plot of weather main Vs traffic volume in year 2015")
```

Out[104]:

```
Text(0.5, 1.0, 'Plot of weather main Vs traffic volume in year 2015')
```

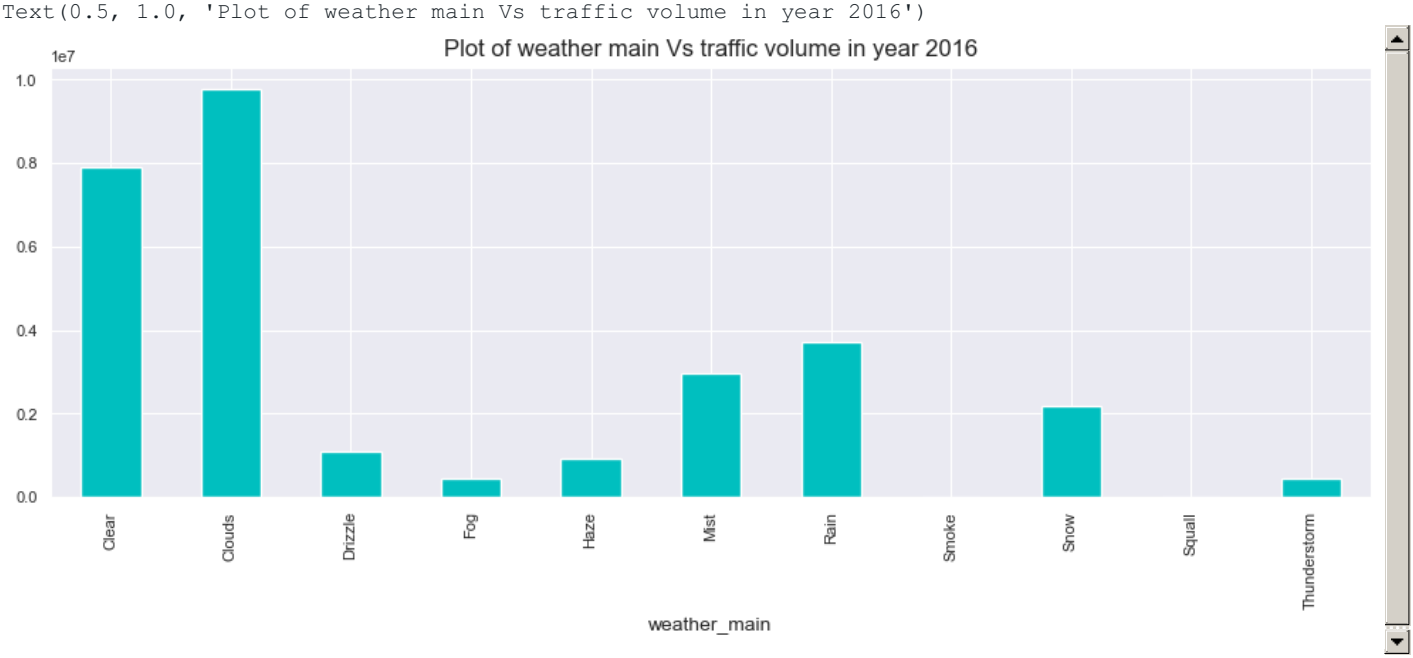


In [105]:

```
y_2016.groupby('weather_main')['traffic_volume'].sum().plot(kind = 'bar',color = 'c',figsize= (12,4))
plt.title("Plot of weather main Vs traffic volume in year 2016")
```

Out[105]:

```
Text(0.5, 1.0, 'Plot of weather main Vs traffic volume in year 2016')
```

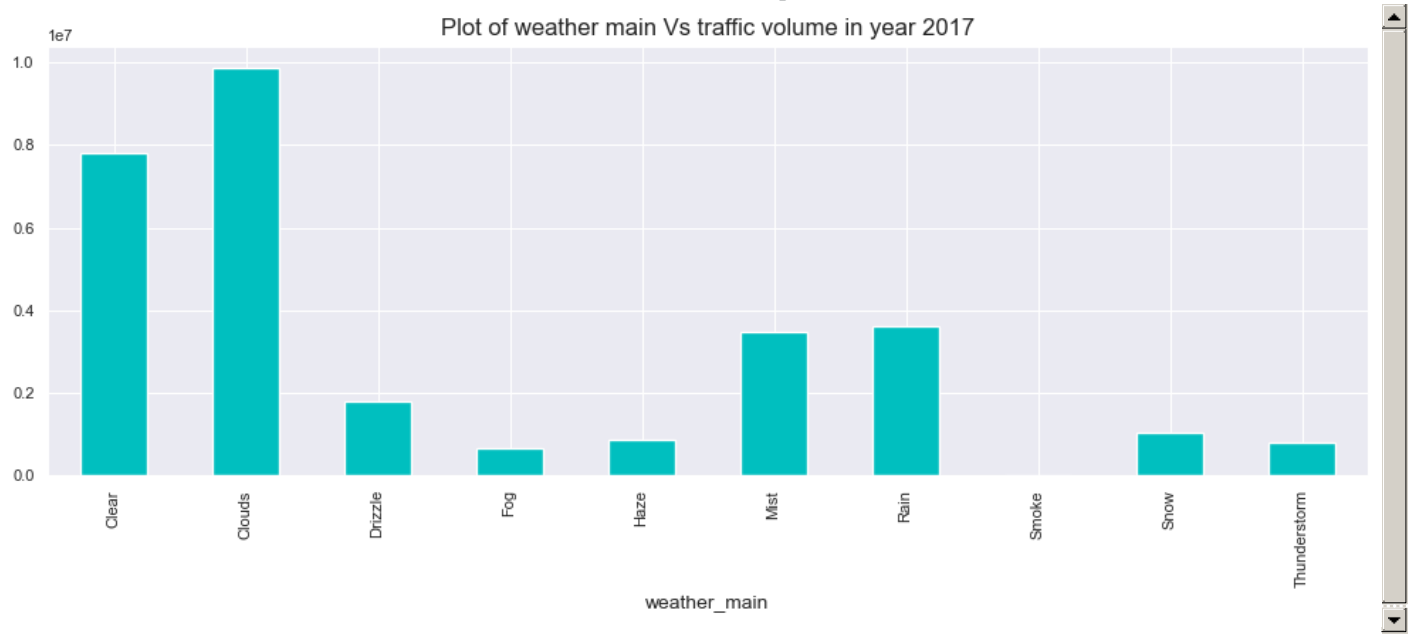


In [106]:

```
y_2017.groupby('weather_main')['traffic_volume'].sum().plot(kind = 'bar',color = 'c',figsize= (12,4))
plt.title("Plot of weather main Vs traffic volume in year 2017")
```

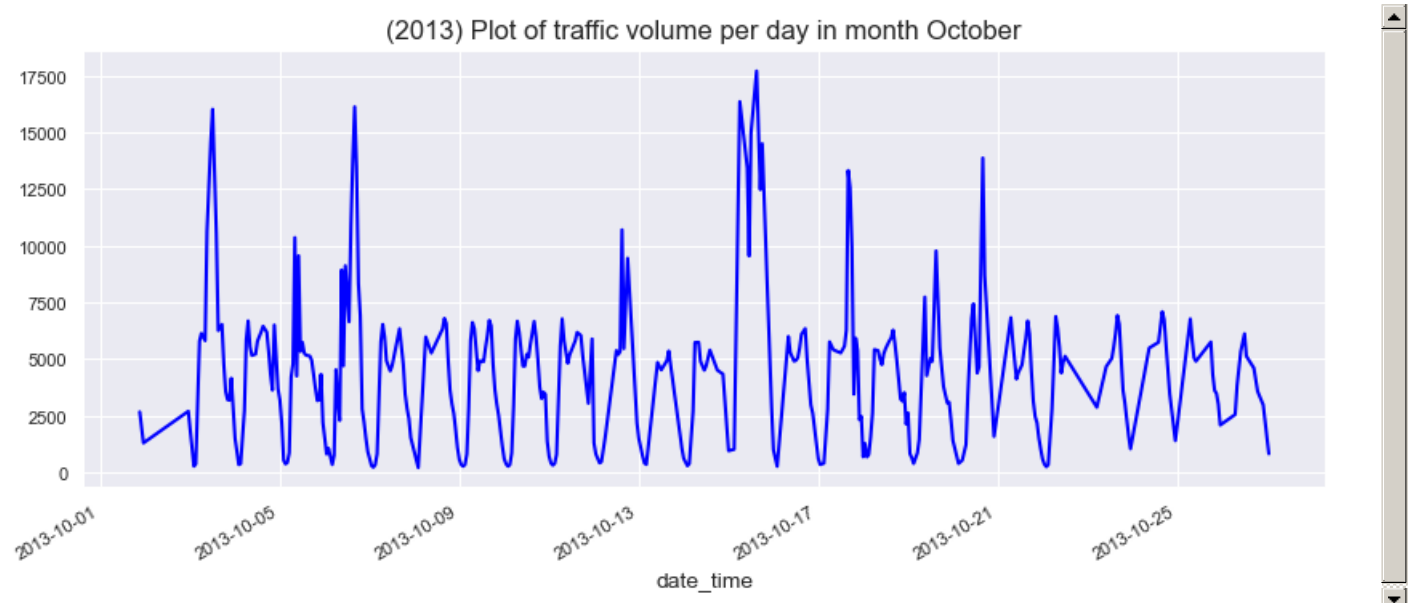
Out[106]:

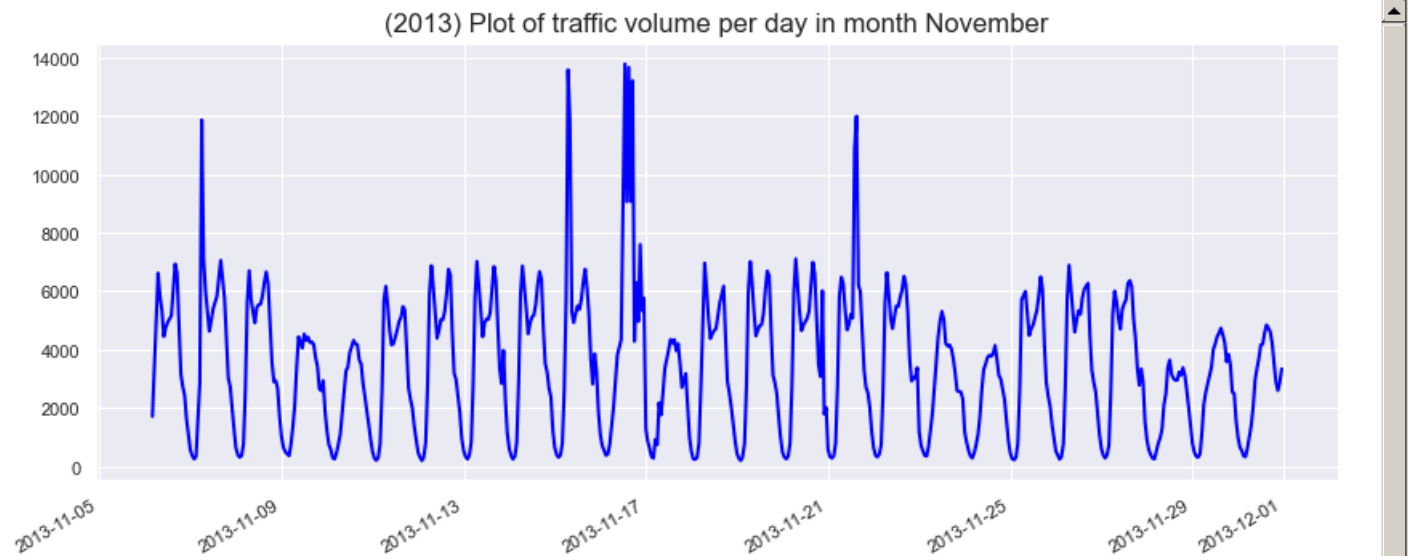
```
Text(0.5, 1.0, 'Plot of weather main Vs traffic volume in year 2017')
```

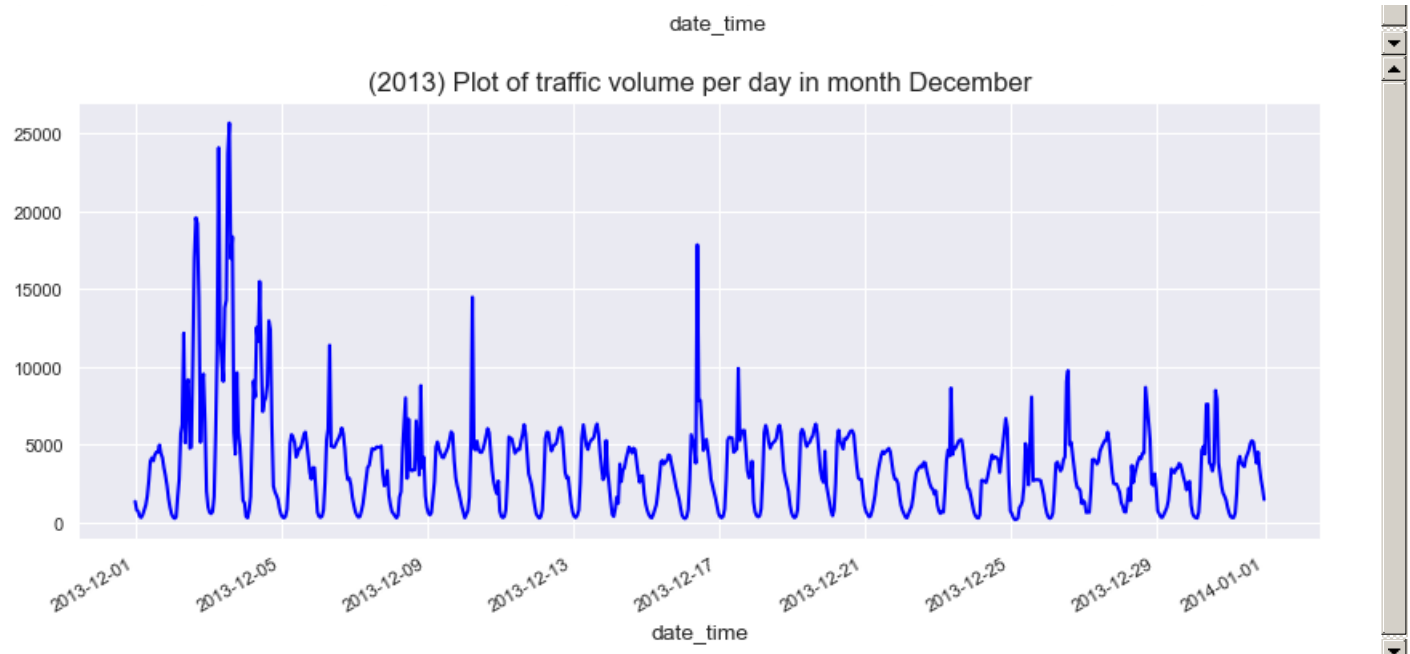


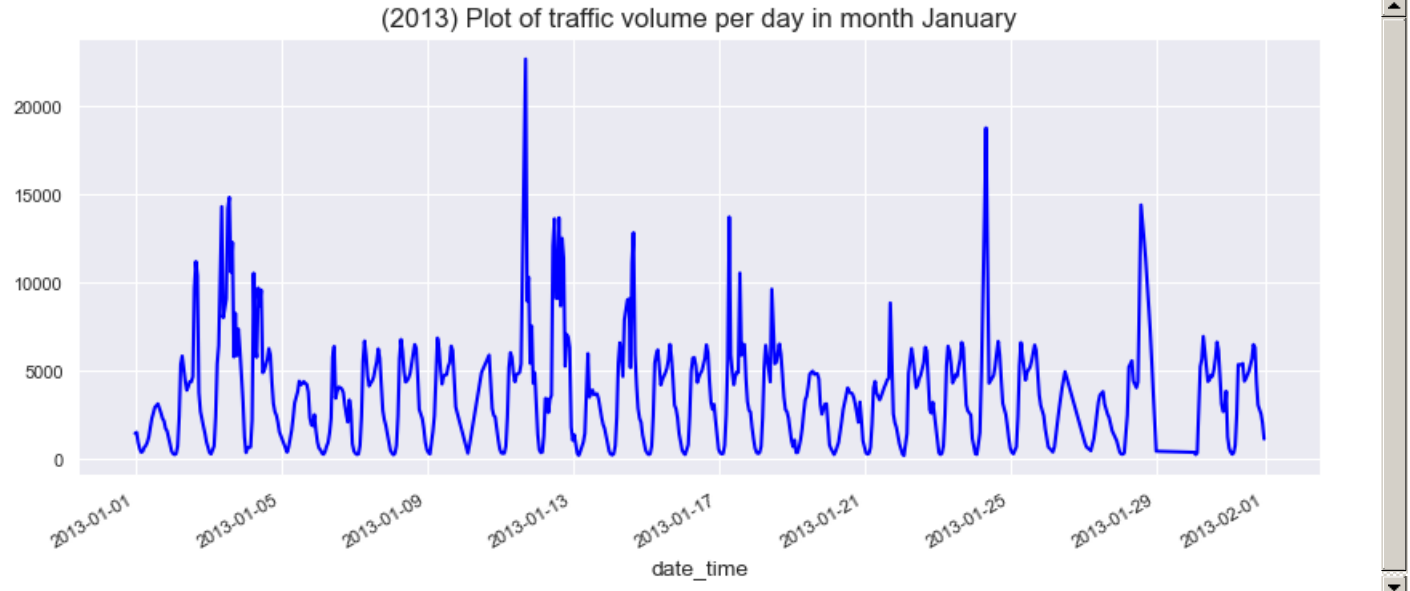
In [119]:

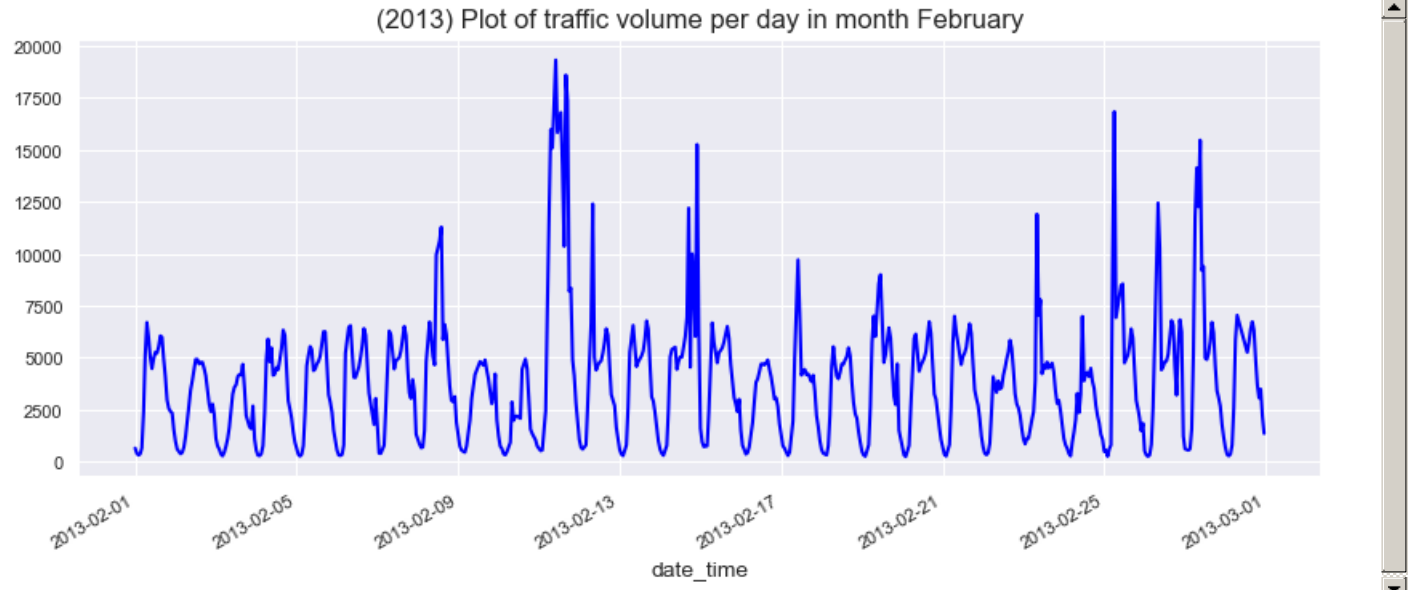
```
mnths=('October', 'November', 'December', 'January', 'February', 'March',
       'April', 'May', 'June', 'July', 'August', 'September')
for i in mnths:

        y_2013.loc[y_2013['month']==i].groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',fi
        plt.title("(2013) Plot of traffic volume per day in month "+str(i))
        plt.show()
```

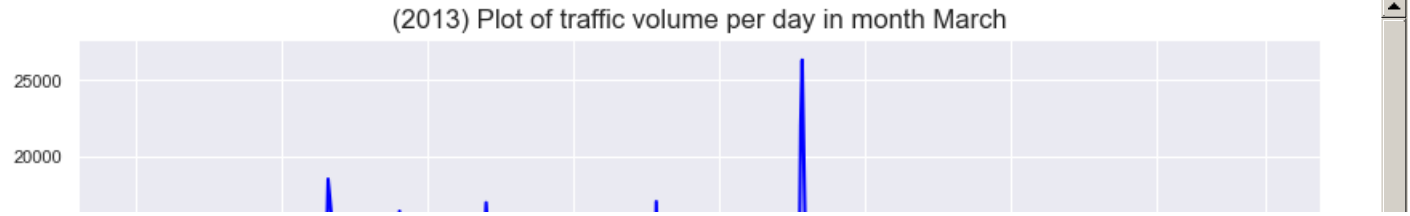


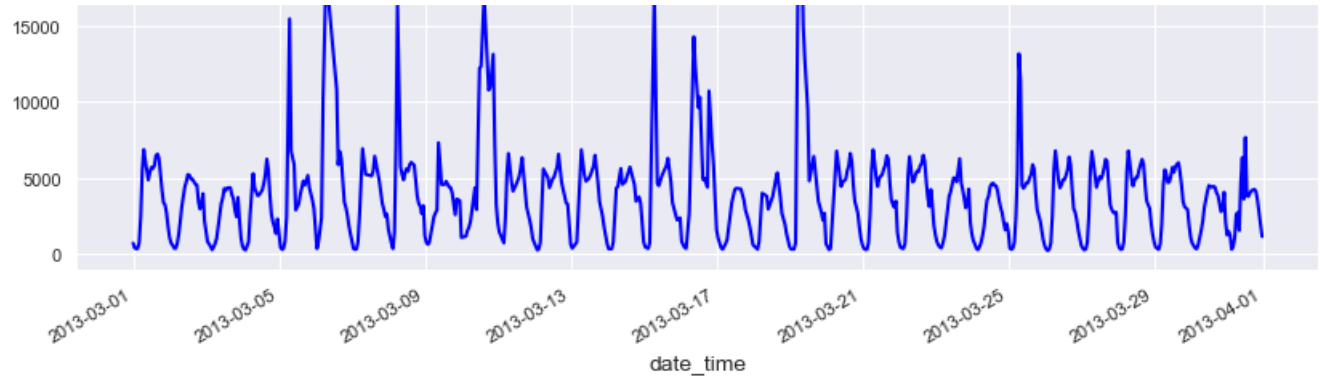



date_time
(2013) Plot of traffic volume per day in month December
25000
20000
15000
10000
5000
0
2013-12-01
2013-12-05
2013-12-09
2013-12-13
2013-12-17
2013-12-21
2013-12-25
2013-12-29
2014-01-01
date_time

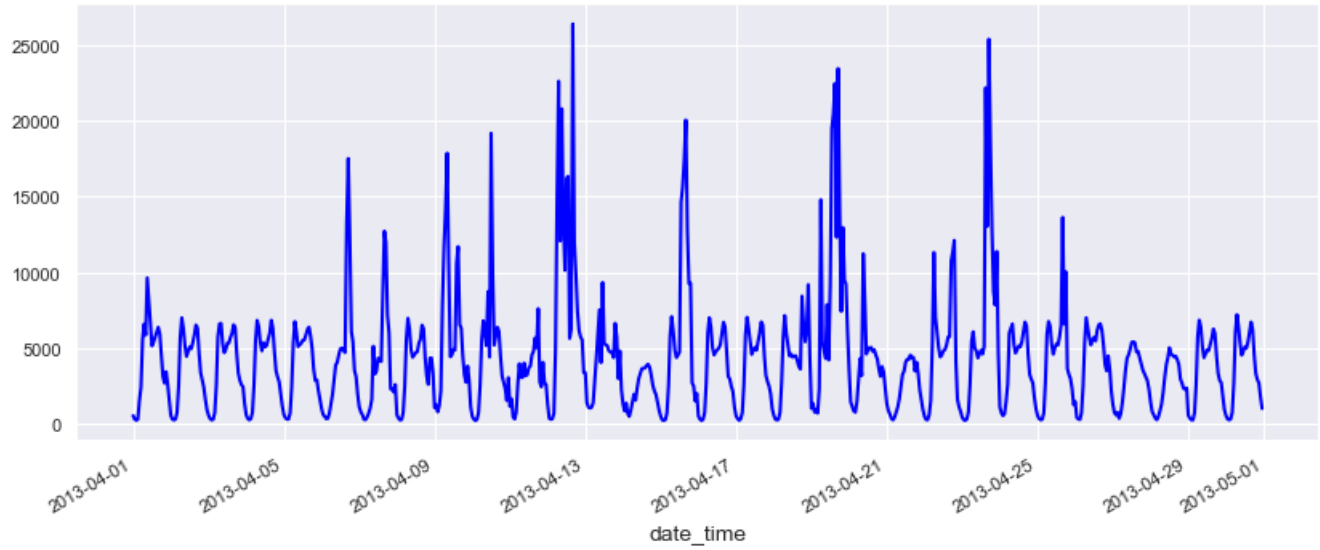

(2013) Plot of traffic volume per day in month January
20000
15000
10000
5000
0
2013-01-01
2013-01-05
2013-01-09
2013-01-13
2013-01-17
2013-01-21
2013-01-25
2013-01-29
2013-02-01
date_time


(2013) Plot of traffic volume per day in month February
20000
17500
15000
12500
10000
7500
5000
2500
0
2013-02-01
2013-02-05
2013-02-09
2013-02-13
2013-02-17
2013-02-21
2013-02-25
2013-03-01
date_time

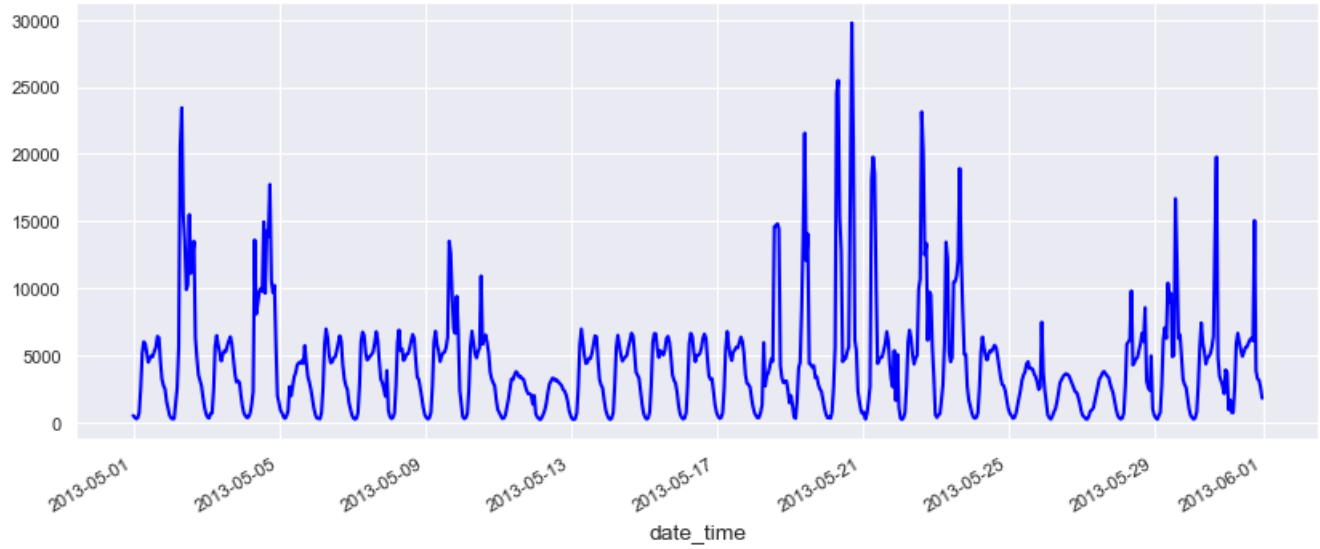

(2013) Plot of traffic volume per day in month March
25000
20000



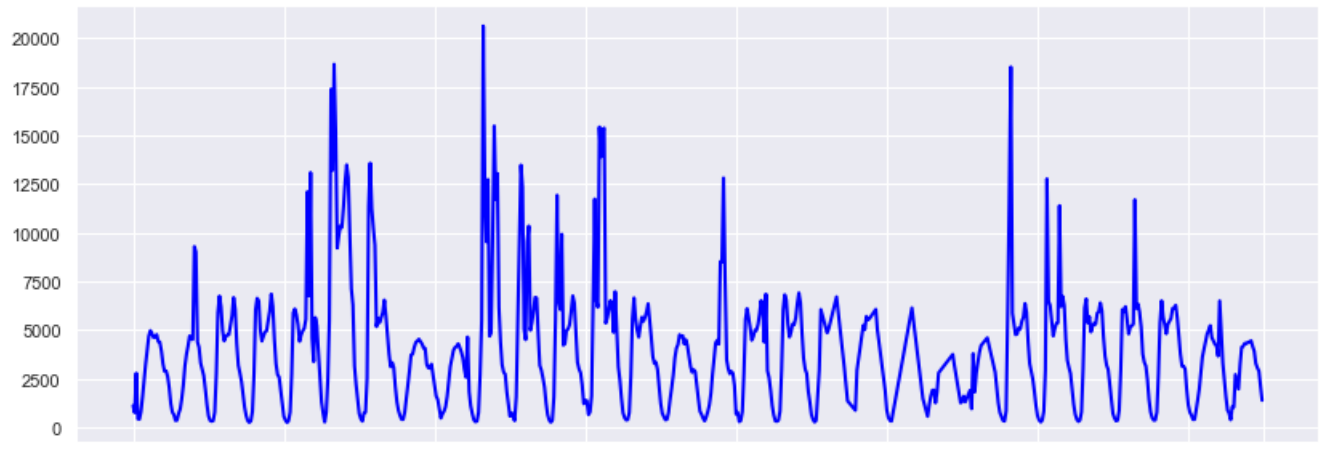
(2013) Plot of traffic volume per day in month April



(2013) Plot of traffic volume per day in month May



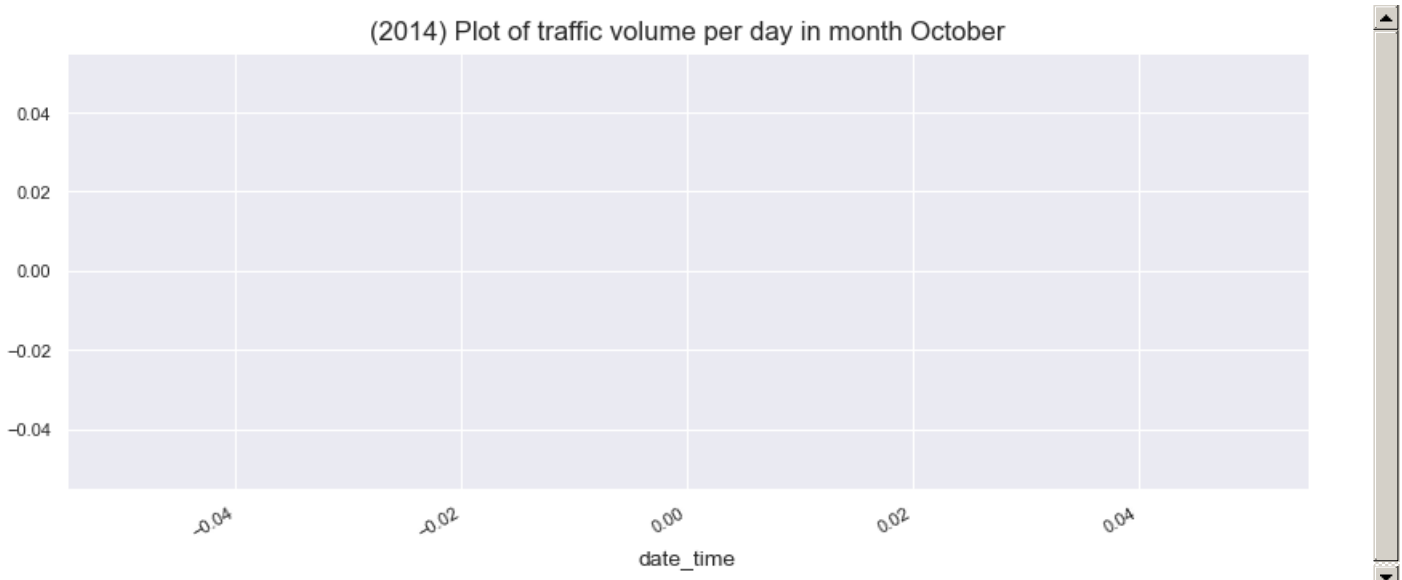
(2013) Plot of traffic volume per day in month June

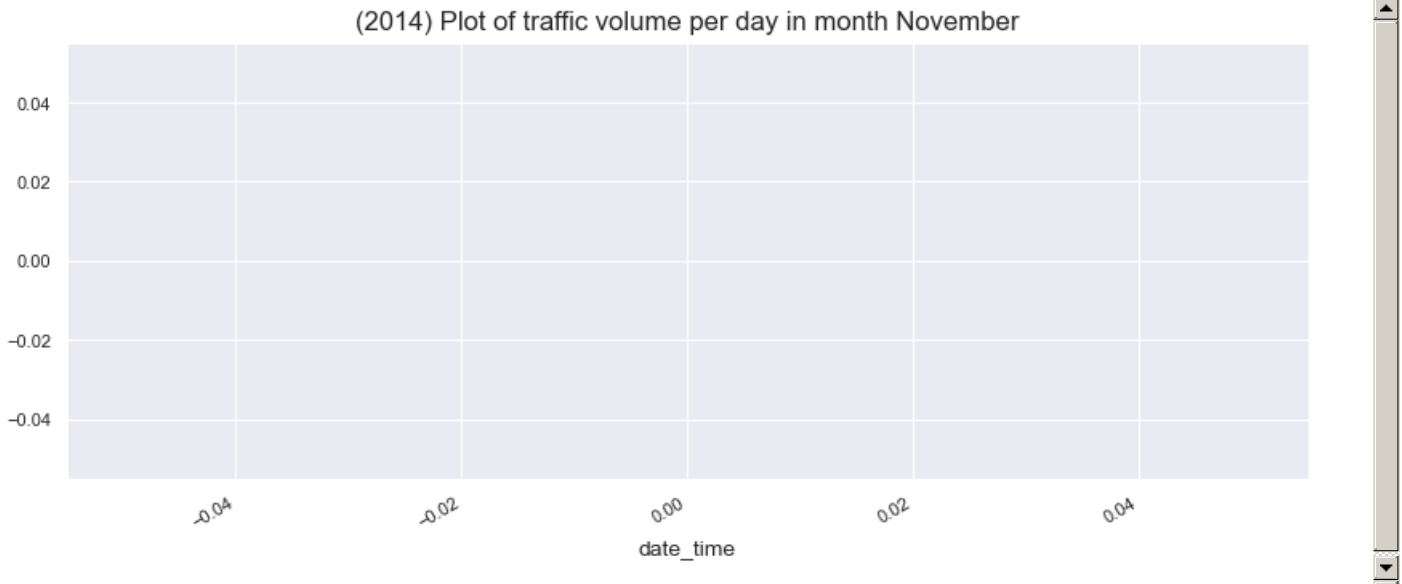


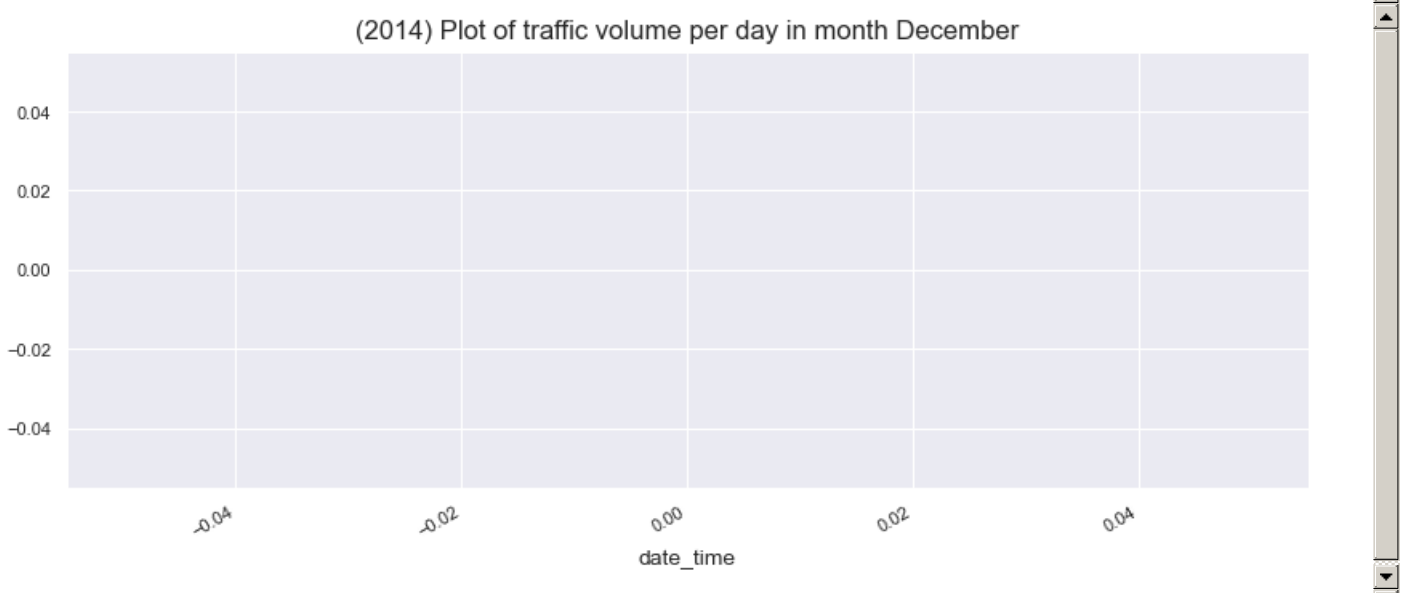
In [125]:

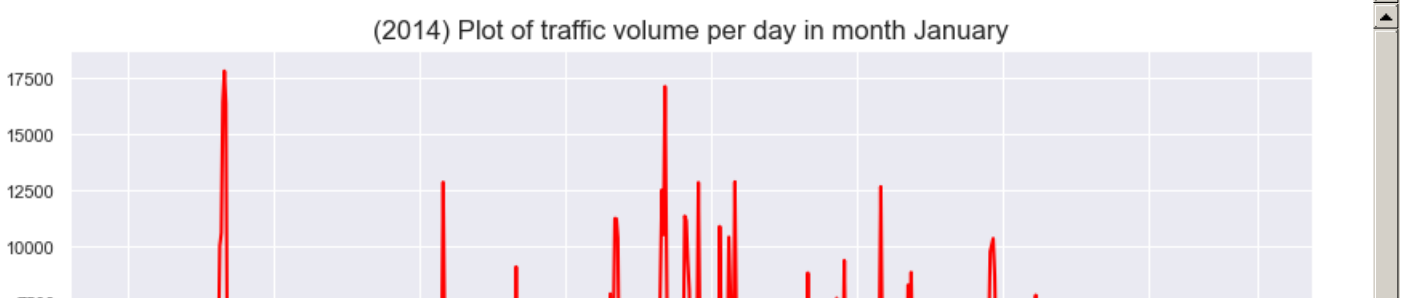
```
for i in mnths:

        y_2014.loc[y_2014['month']==i].groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',fi
        plt.title("(2014) Plot of traffic volume per day in month "+str(i))
```

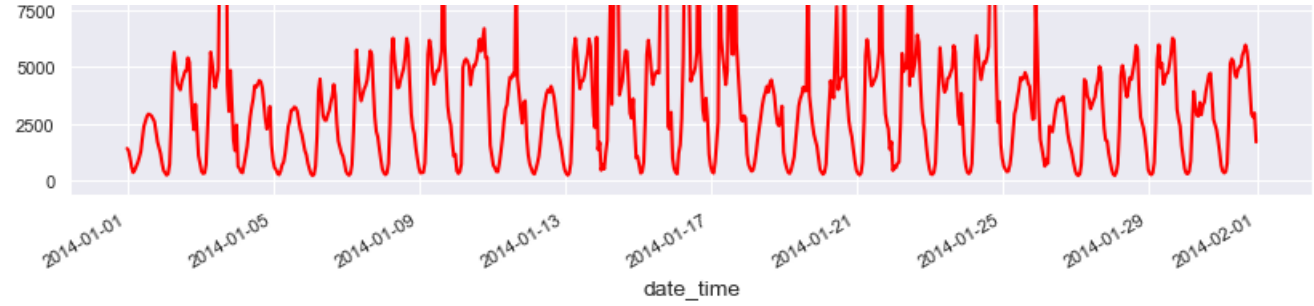
```
plt.show()
```


(2014) Plot of traffic volume per day in month October
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time

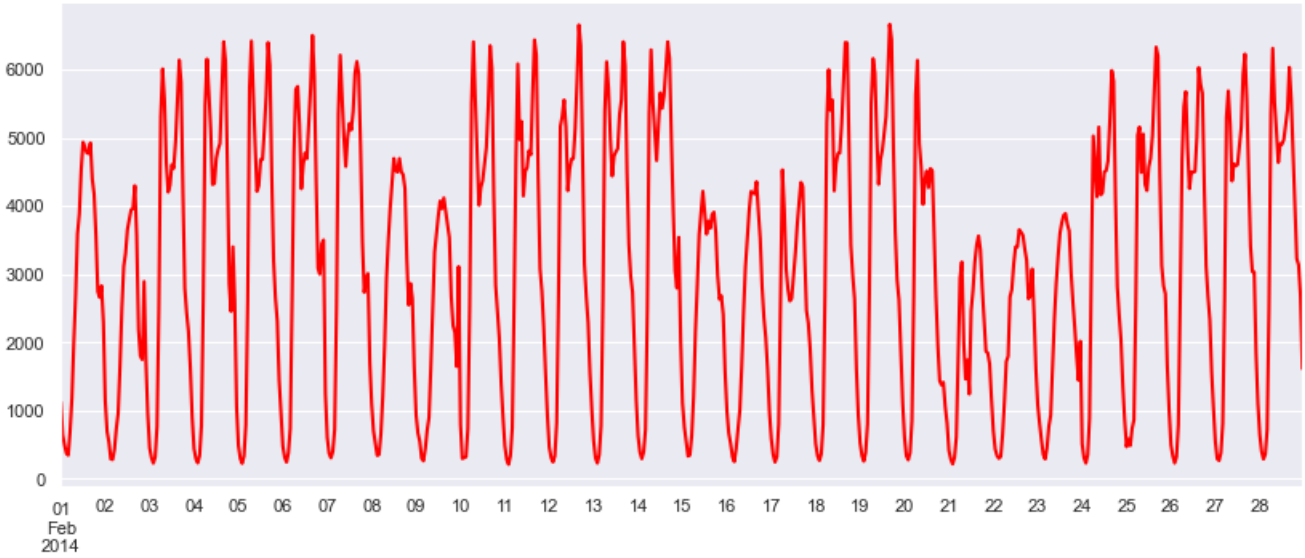

(2014) Plot of traffic volume per day in month November
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time


(2014) Plot of traffic volume per day in month December
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time

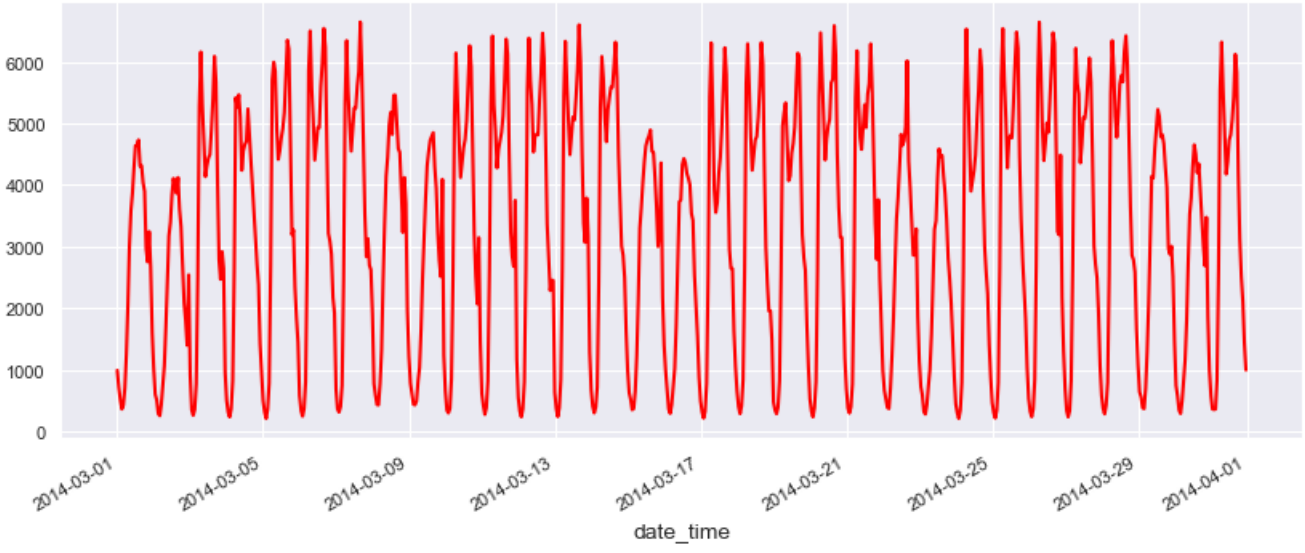

(2014) Plot of traffic volume per day in month January
17500
15000
12500
10000



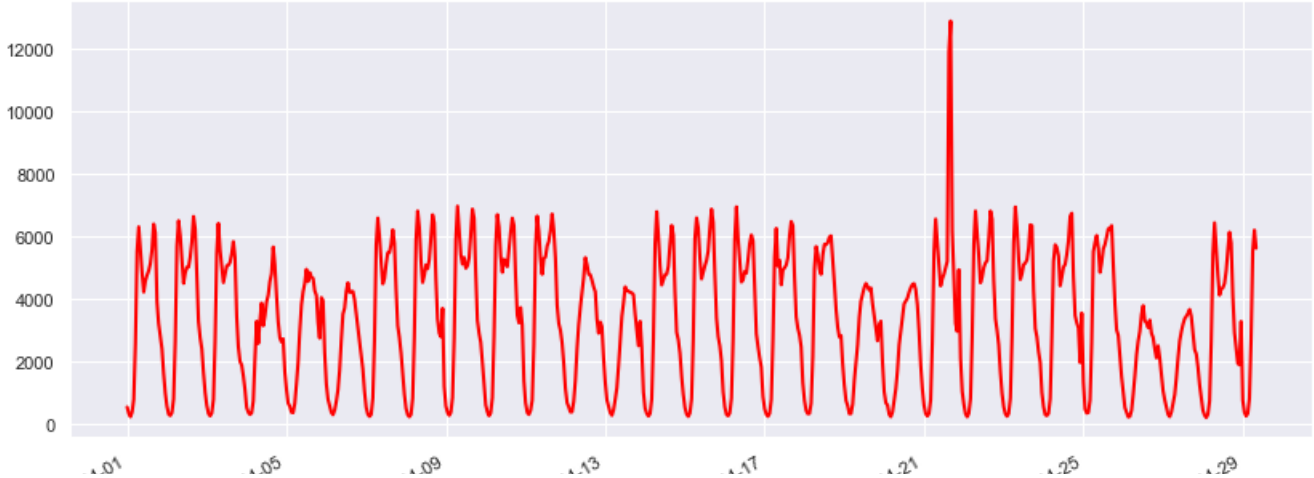
(2014) Plot of traffic volume per day in month February

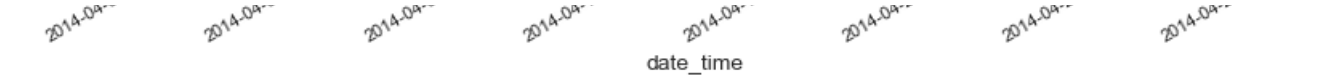


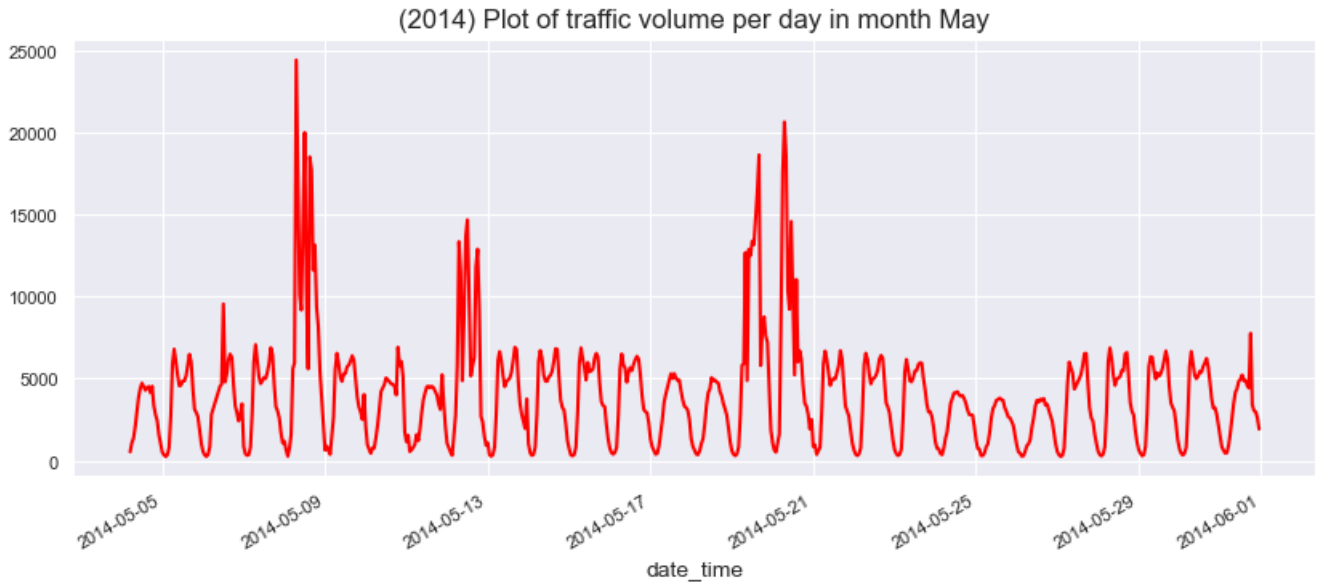
(2014) Plot of traffic volume per day in month March

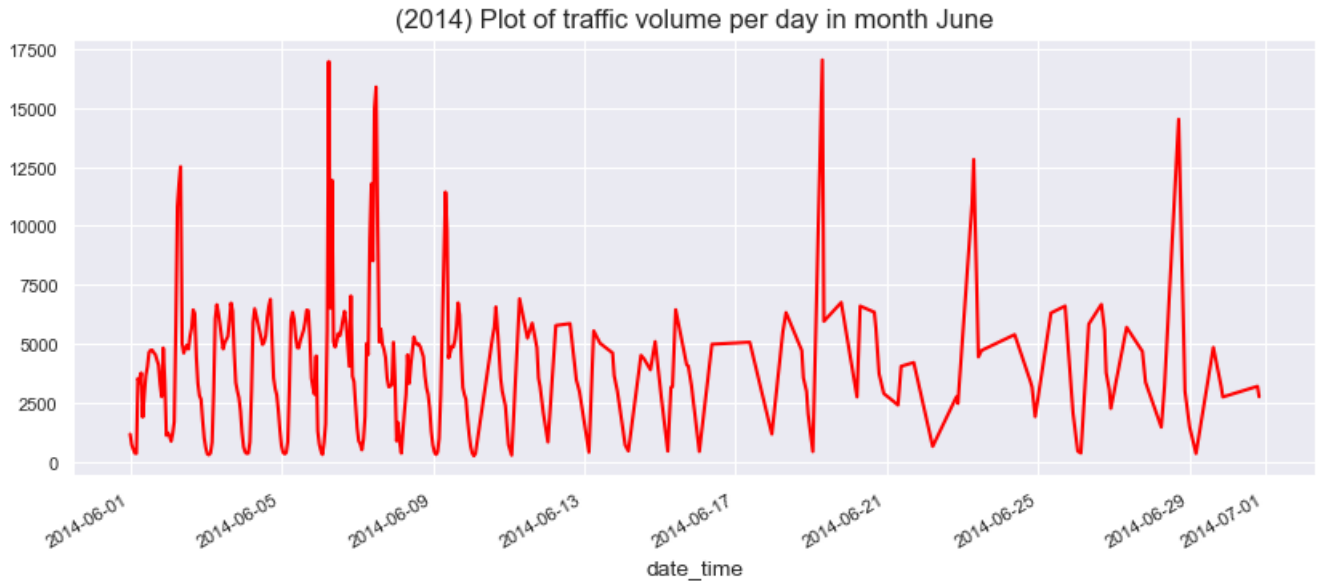


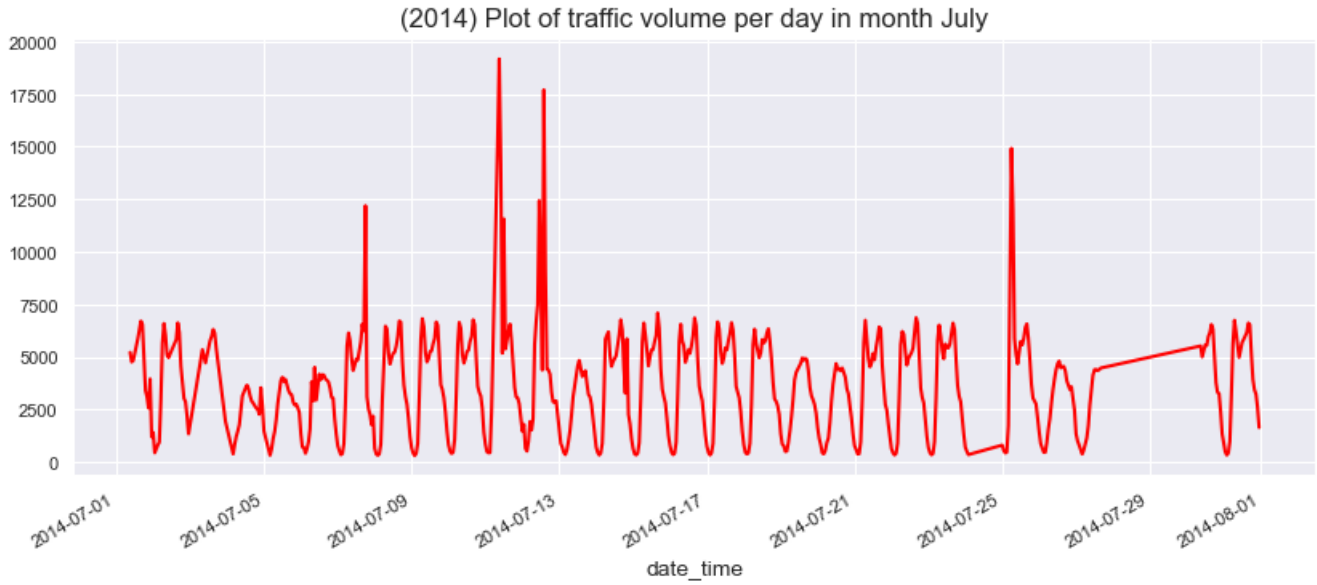
(2014) Plot of traffic volume per day in month April



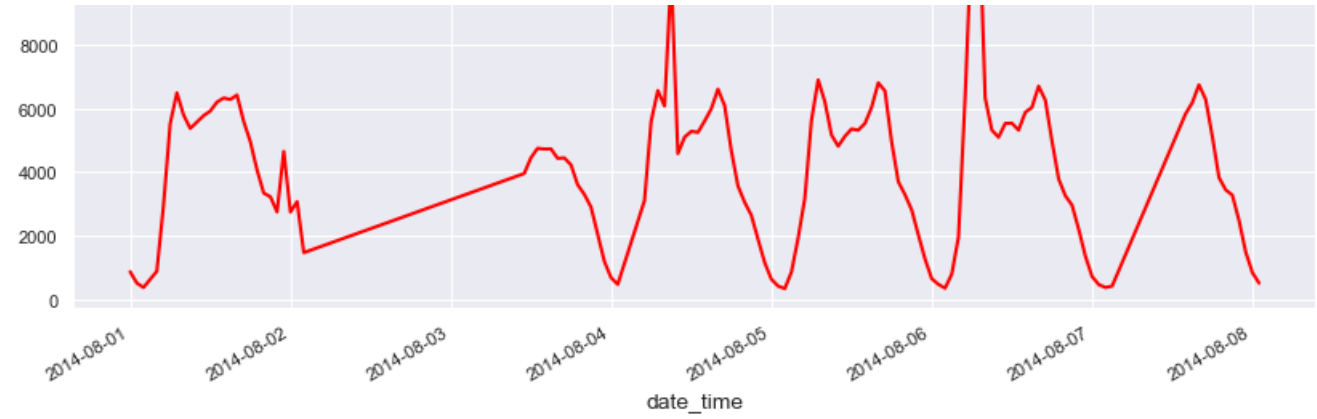

date_time

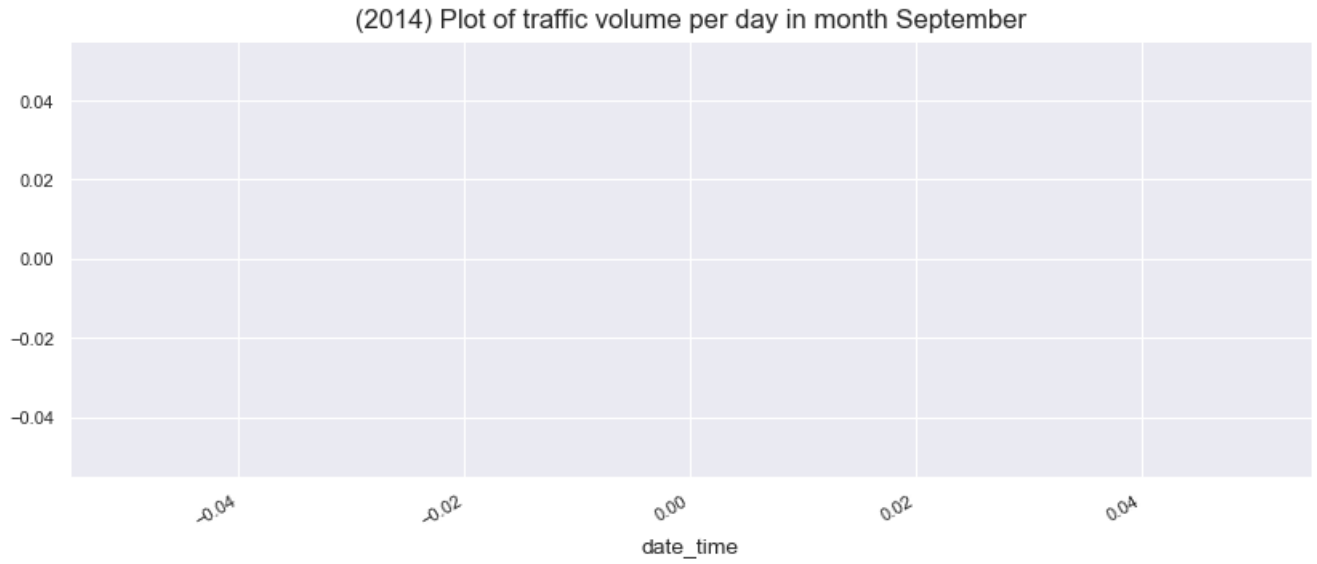

(2014) Plot of traffic volume per day in month May
25000
20000
15000
10000
5000
0
2014-05-05
2014-05-09
2014-05-13
2014-05-17
2014-05-21
2014-05-25
2014-05-29
2014-06-01
date_time


(2014) Plot of traffic volume per day in month June
17500
15000
12500
10000
7500
5000
2500
0
2014-06-01
2014-06-05
2014-06-09
2014-06-13
2014-06-17
2014-06-21
2014-06-25
2014-06-29
2014-07-01
date_time


(2014) Plot of traffic volume per day in month July
20000
17500
15000
12500
10000
7500
5000
2500
0
2014-07-01
2014-07-05
2014-07-09
2014-07-13
2014-07-17
2014-07-21
2014-07-25
2014-07-29
2014-08-01
date_time

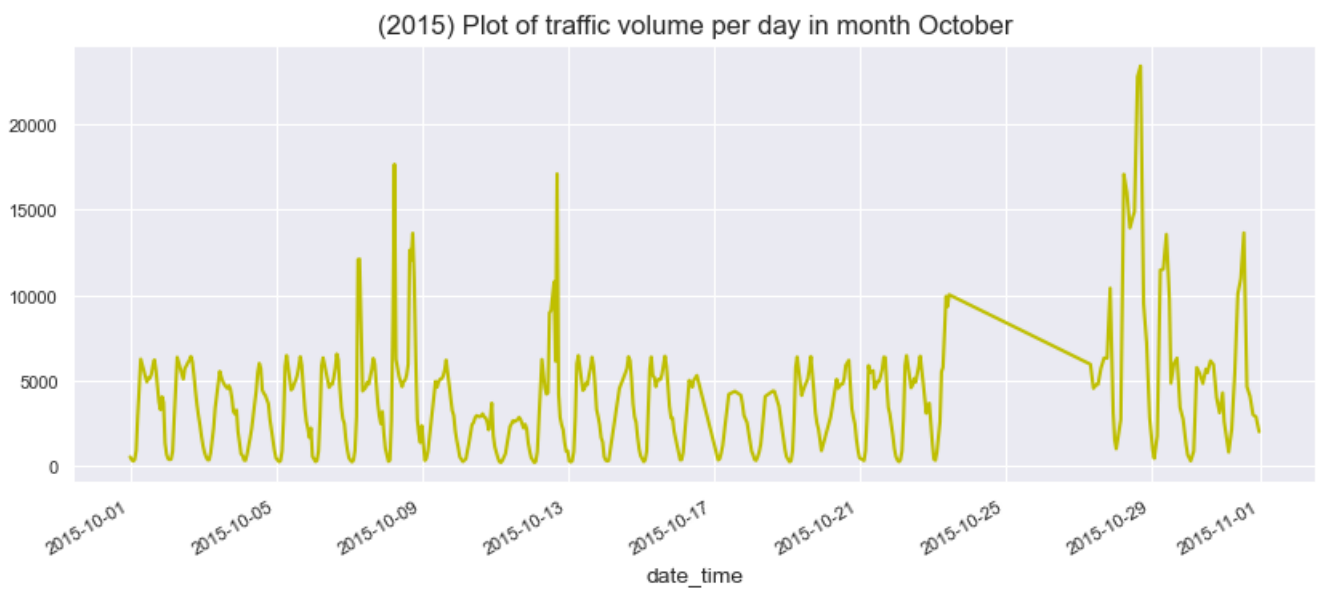

(2014) Plot of traffic volume per day in month August
12000
10000

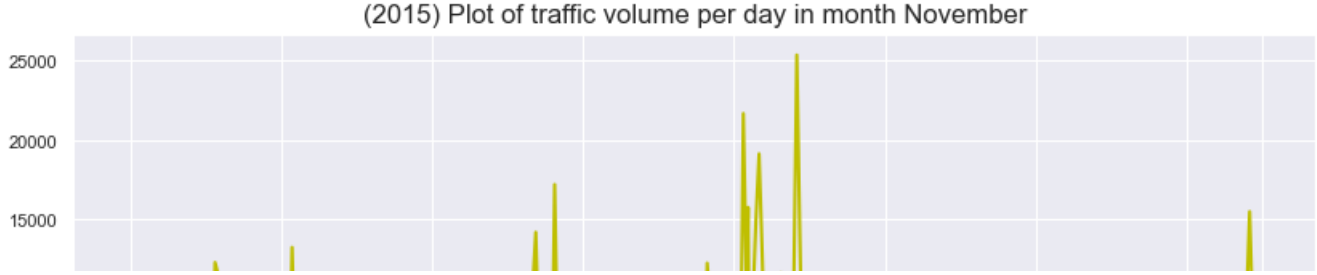


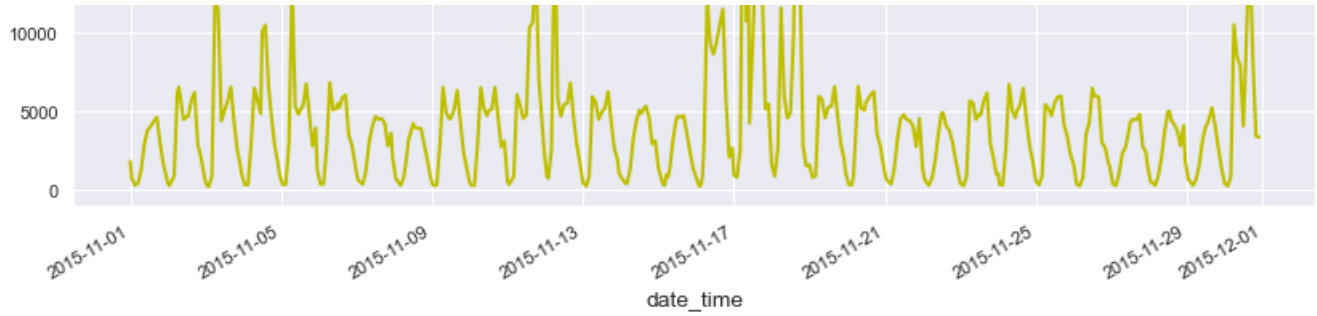


In [126]:

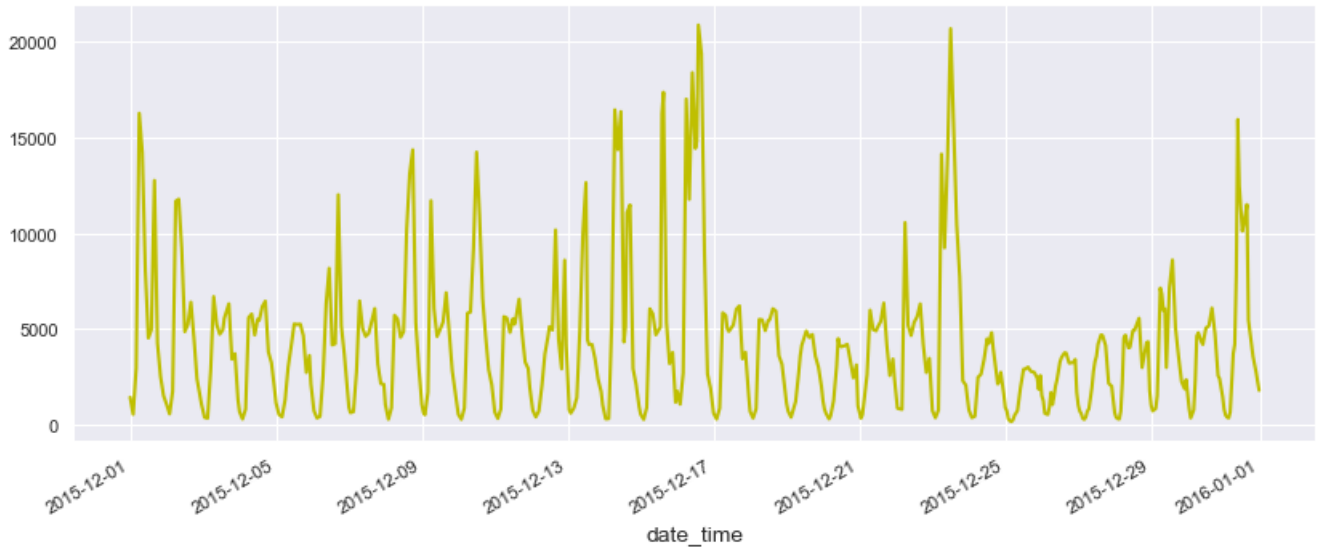
```
for i in mnths:

    y_2015.loc[y_2015['month']==i].groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',fi
    plt.title("(2015) Plot of traffic volume per day in month "+str(i))
    plt.show()
```



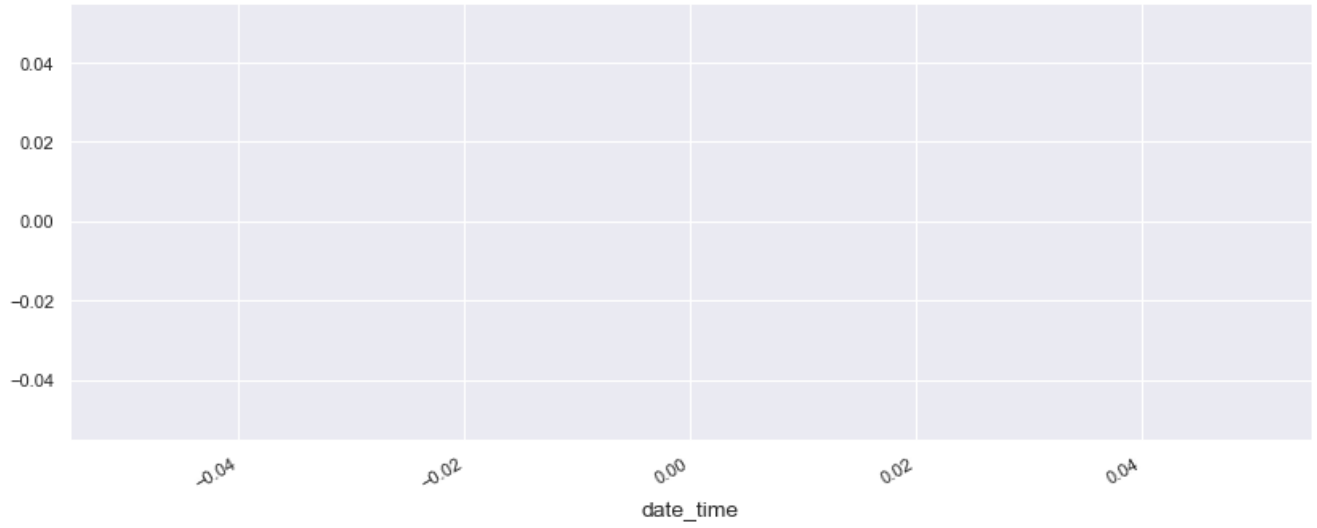




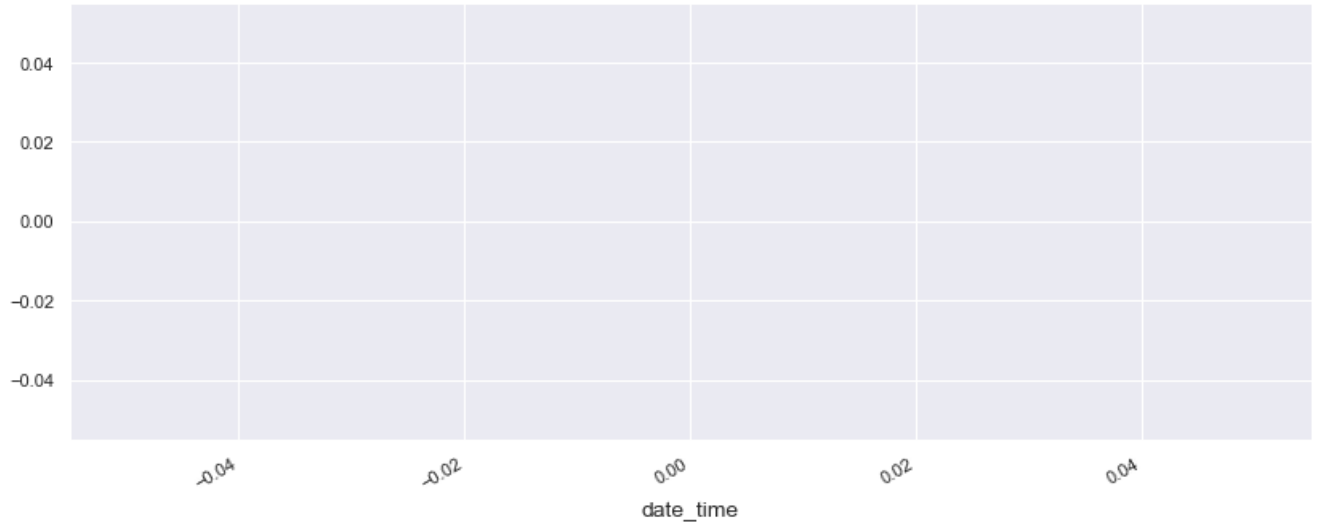
(2015) Plot of traffic volume per day in month December

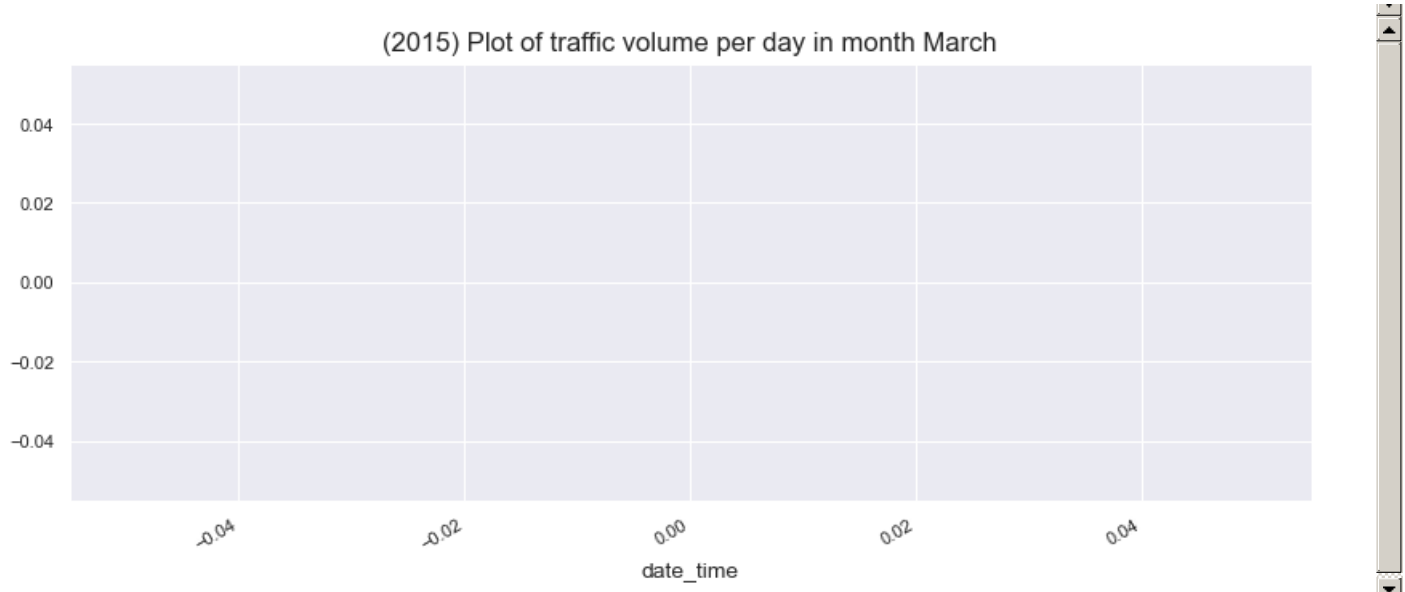


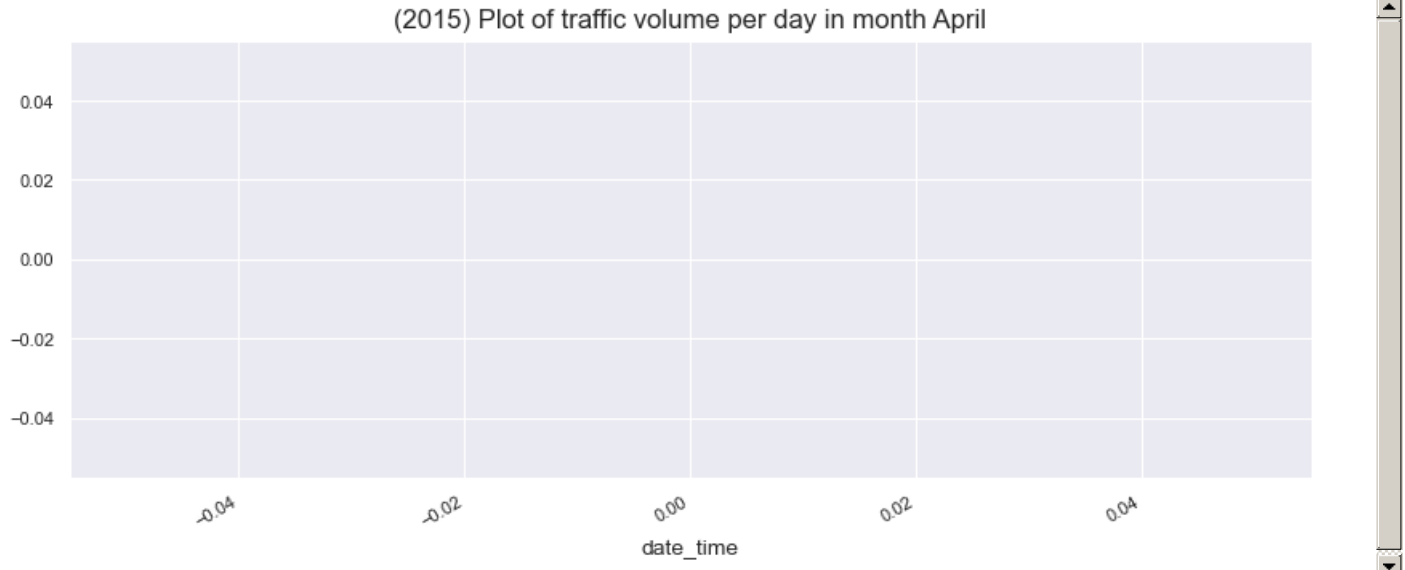
(2015) Plot of traffic volume per day in month January

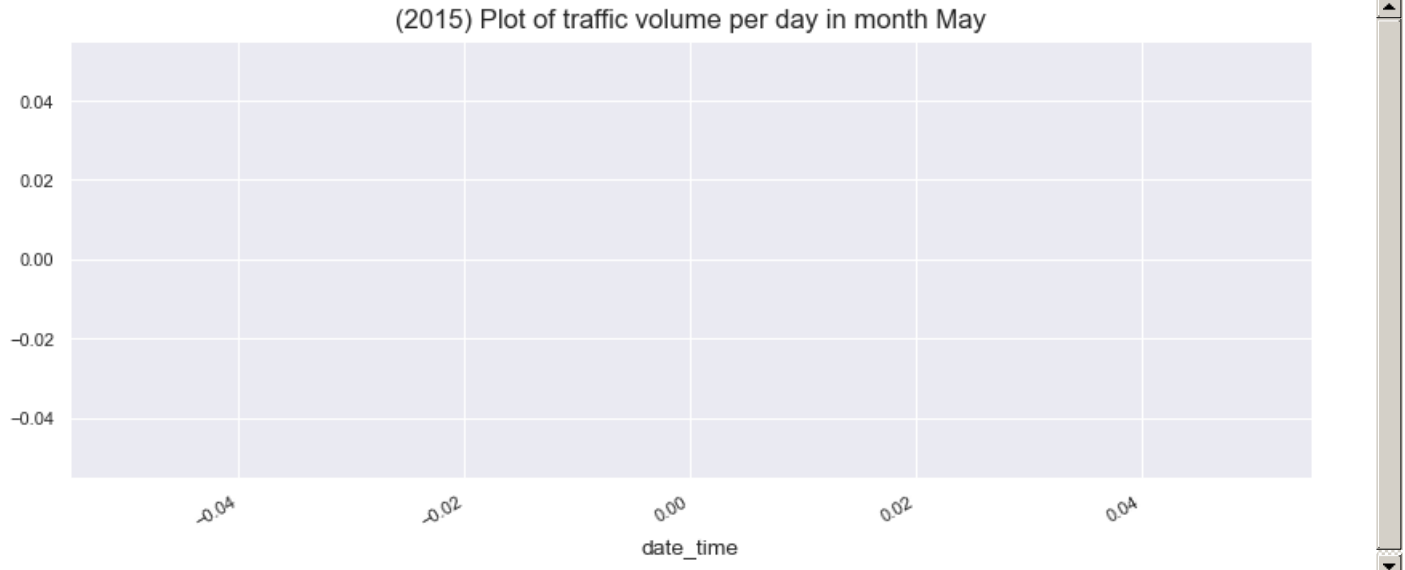


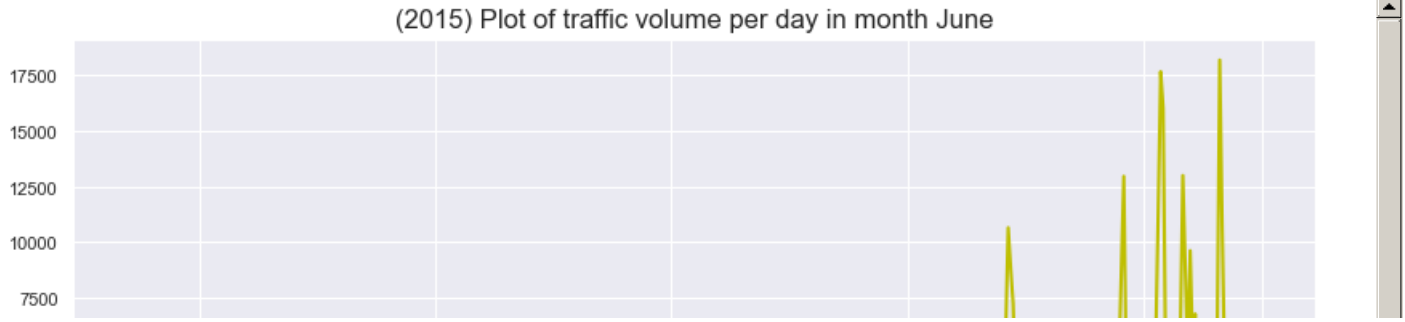
(2015) Plot of traffic volume per day in month February

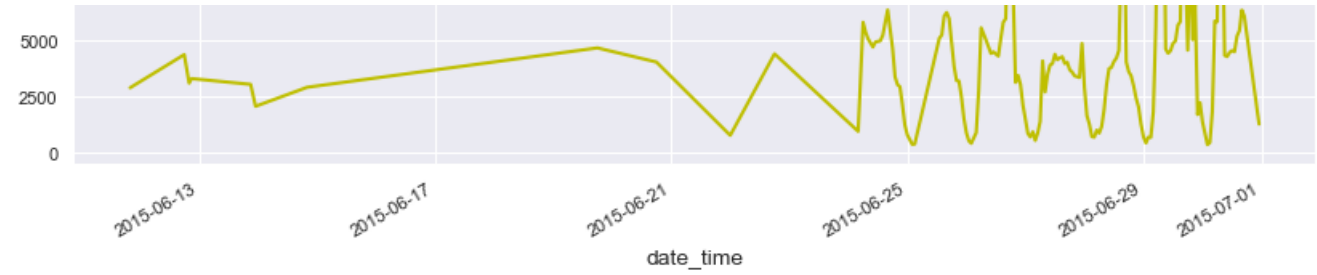



(2015) Plot of traffic volume per day in month March
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time

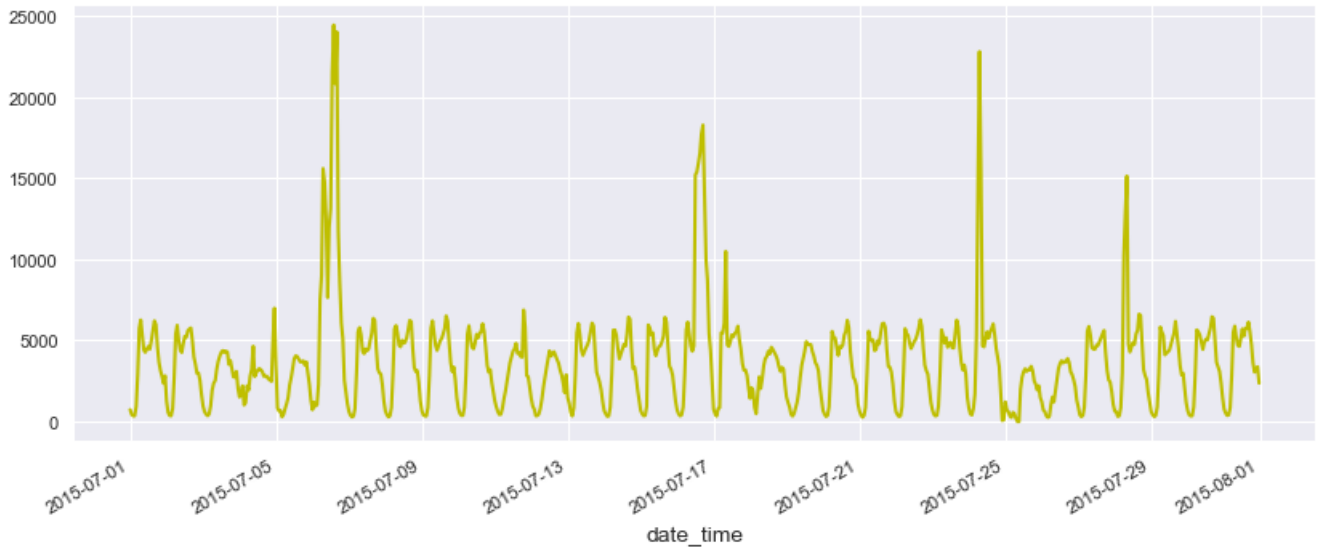

(2015) Plot of traffic volume per day in month April
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time


(2015) Plot of traffic volume per day in month May
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time

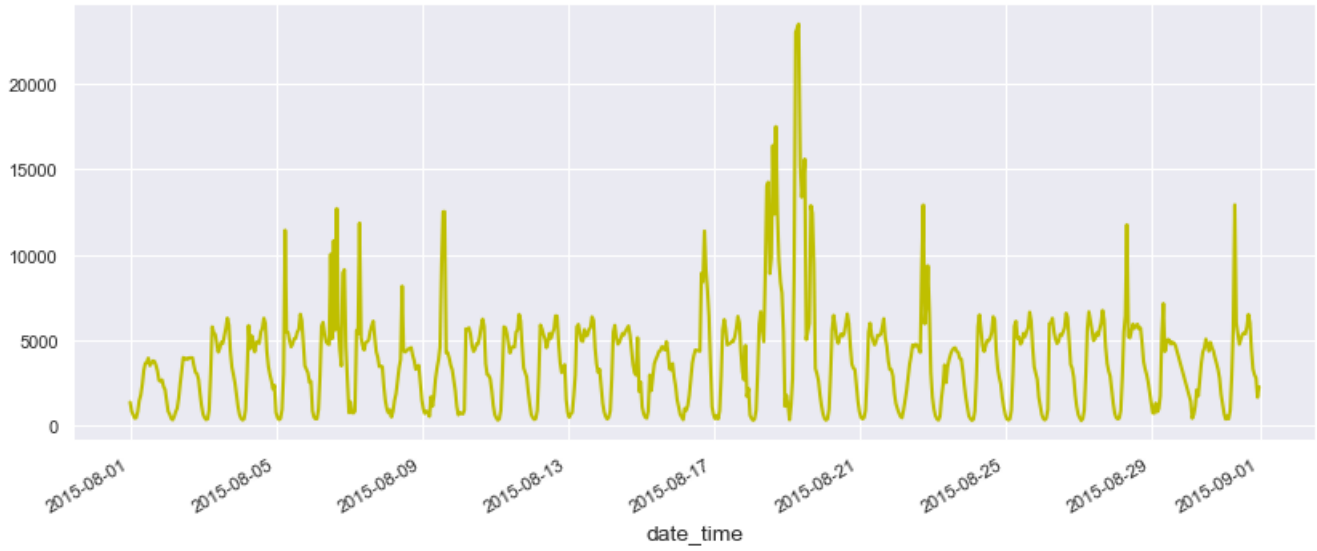

(2015) Plot of traffic volume per day in month June
17500
15000
12500
10000
7500



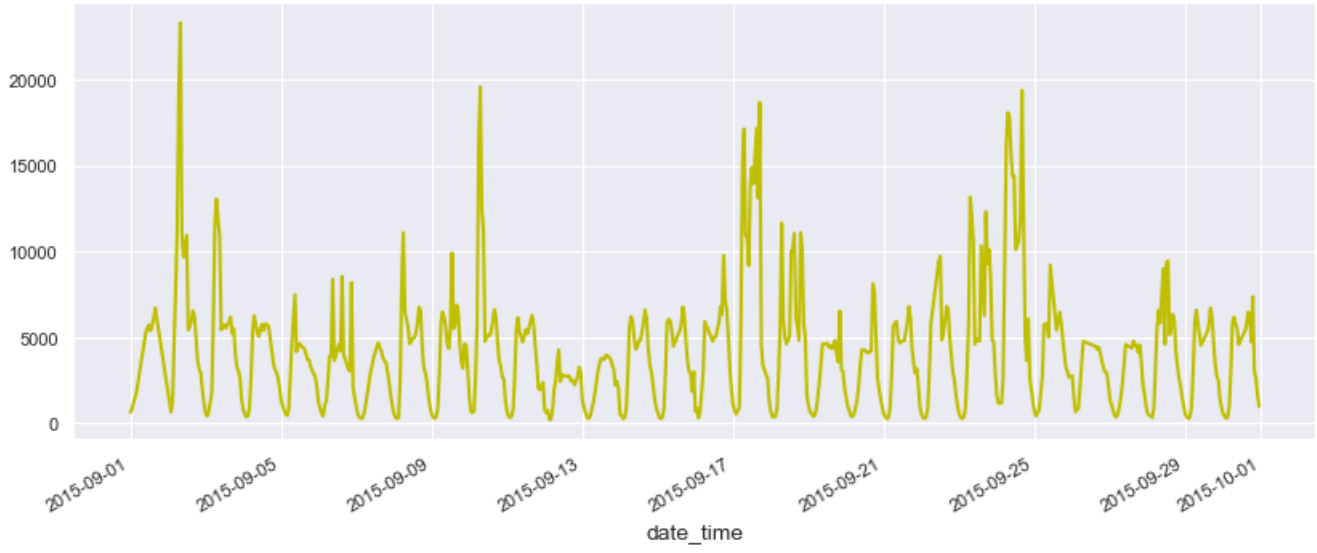
(2015) Plot of traffic volume per day in month July



(2015) Plot of traffic volume per day in month August

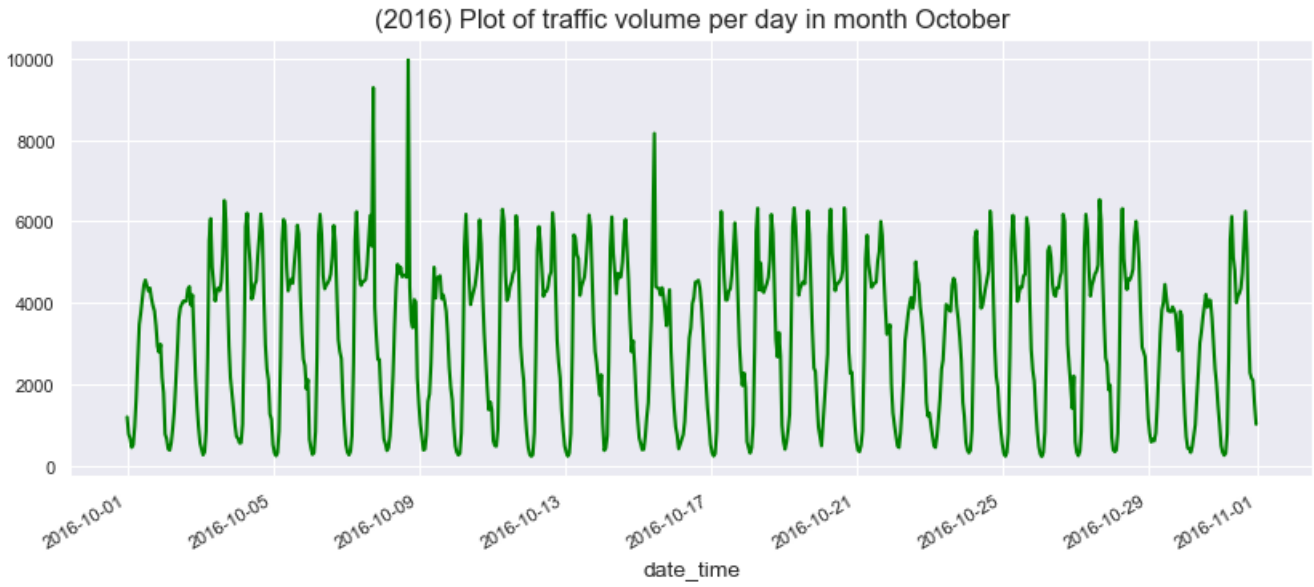


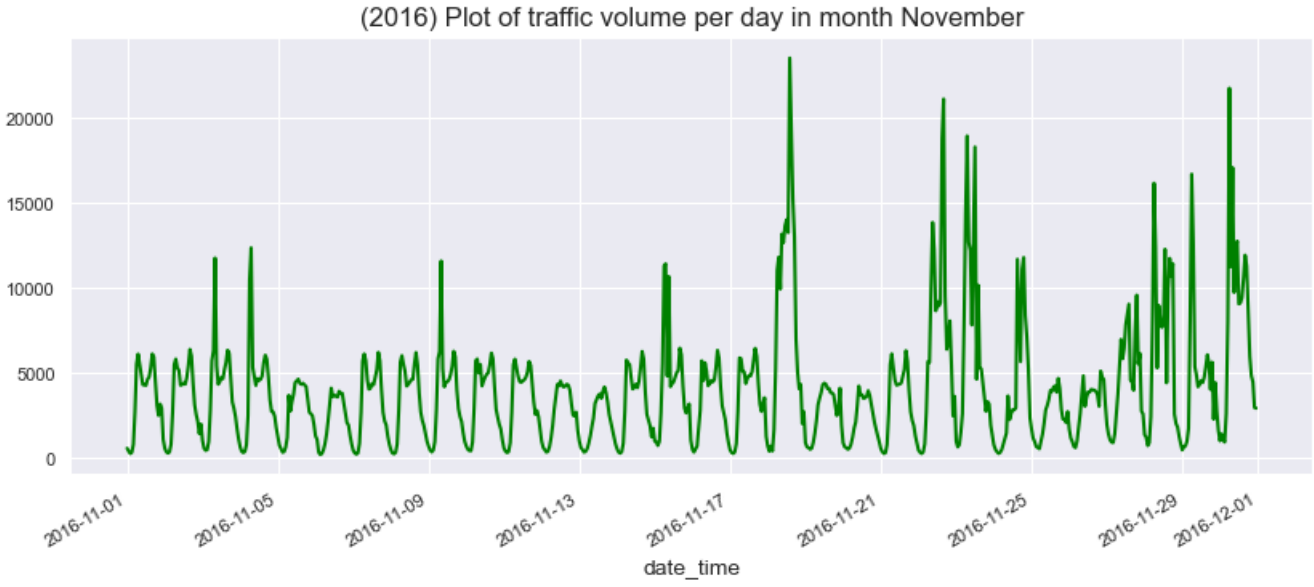
(2015) Plot of traffic volume per day in month September

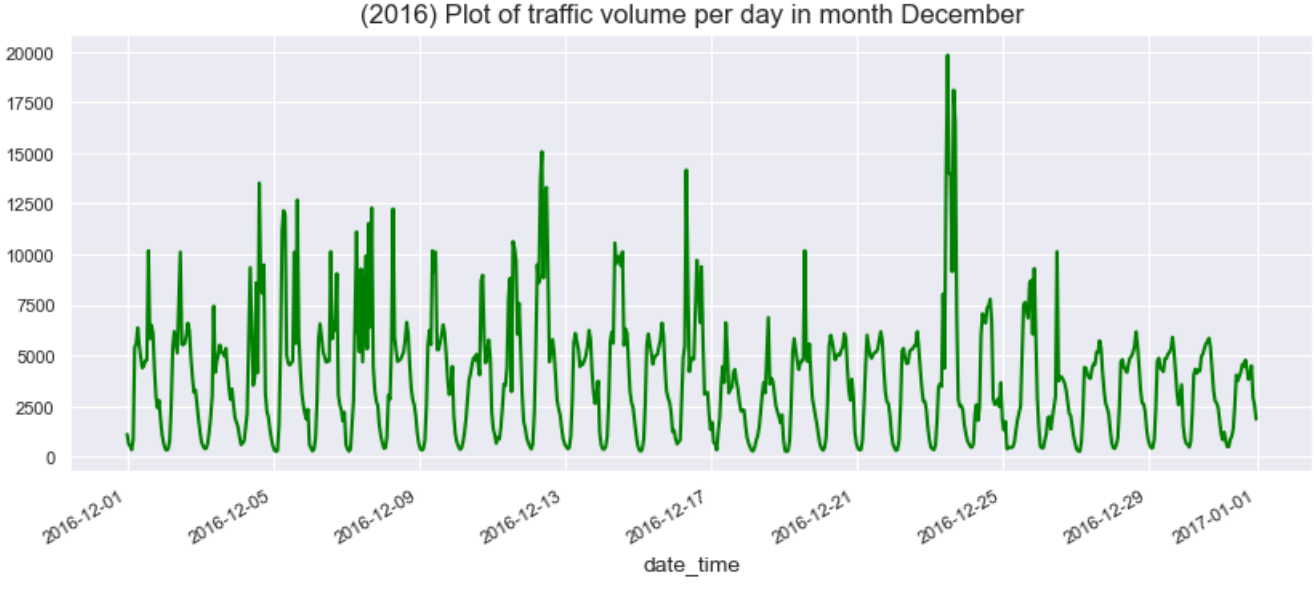


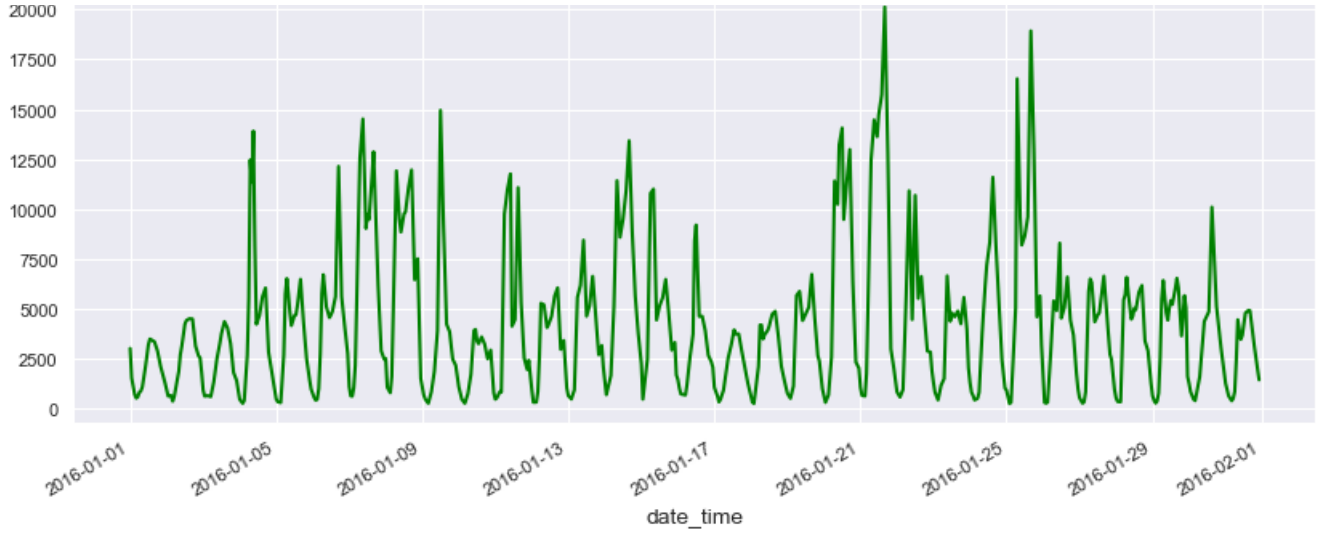
In [129]:

```
for i in mnths:

    y_2016.loc[y_2016['month']==i].groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',fi
    plt.title("(2016) Plot of traffic volume per day in month "+str(i))
    plt.show()
```

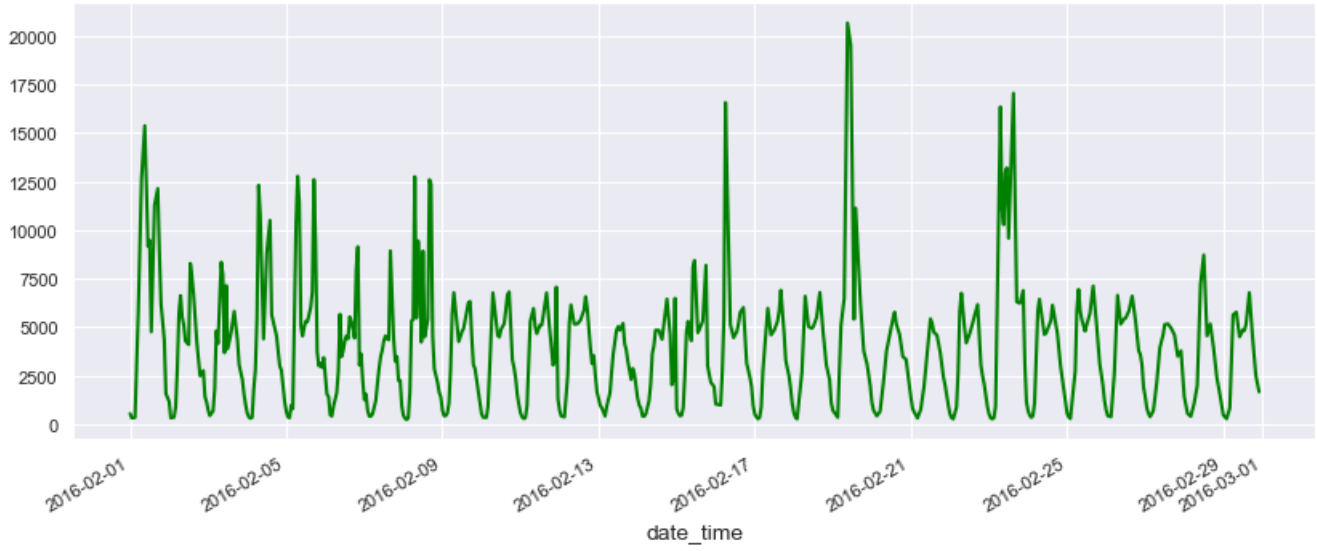




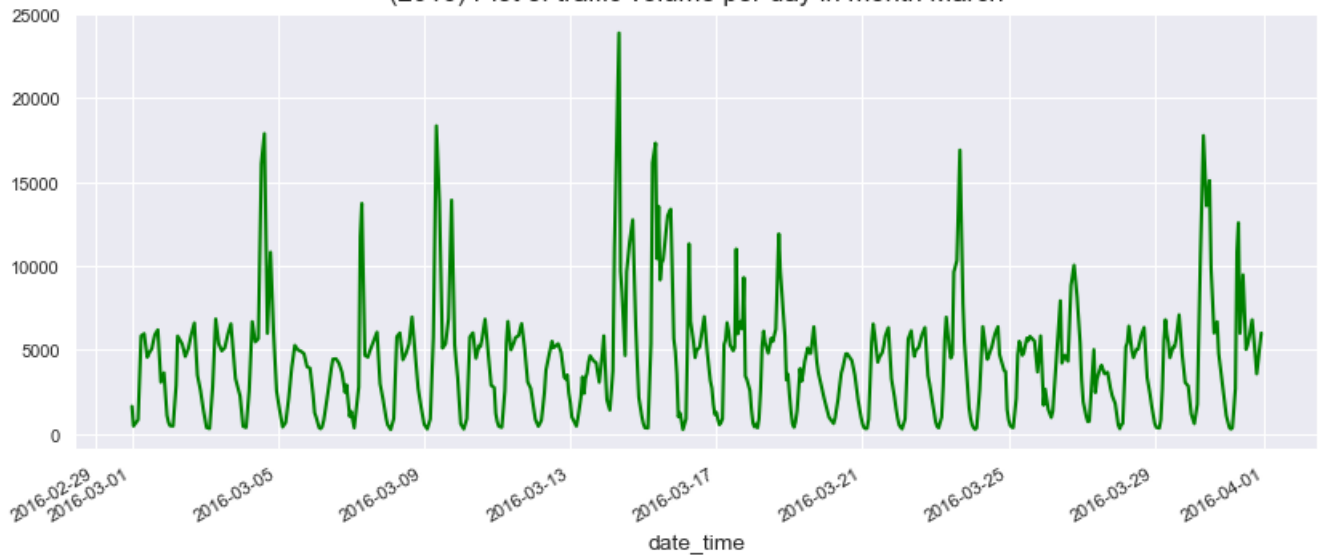




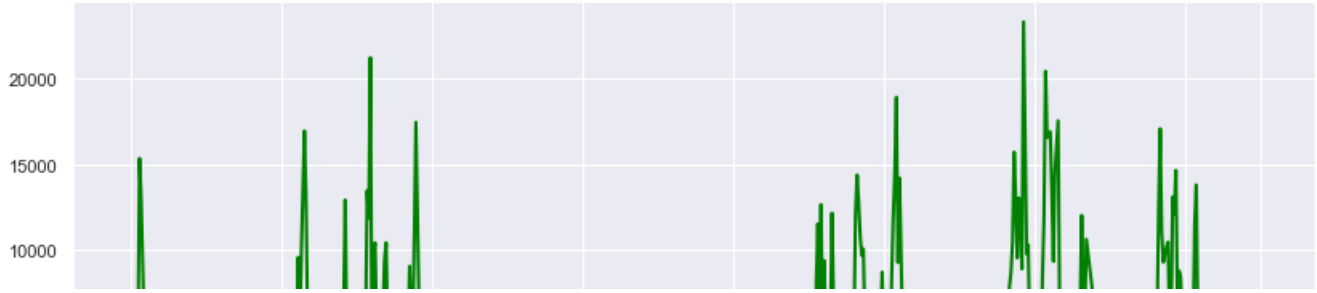
(2016) Plot of traffic volume per day in month February

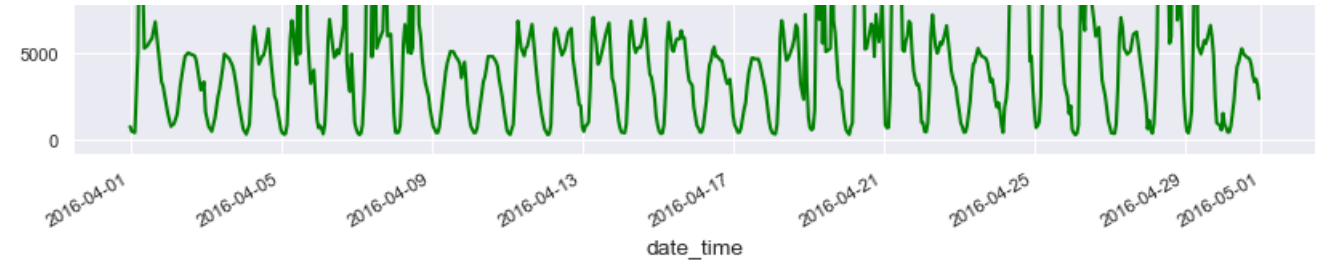


(2016) Plot of traffic volume per day in month March

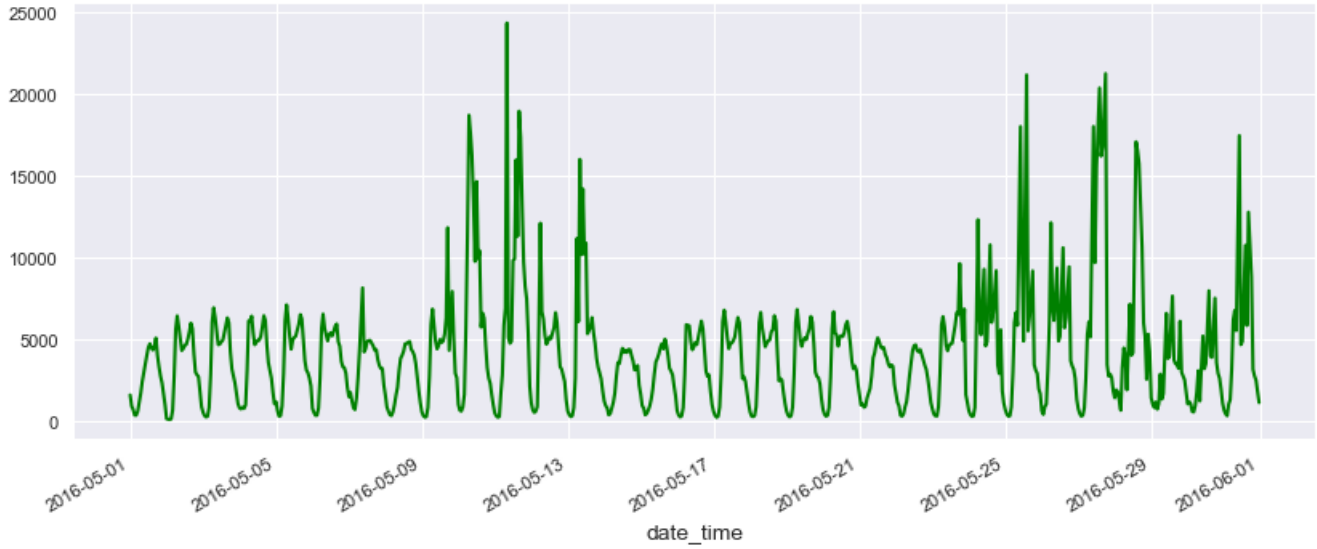


(2016) Plot of traffic volume per day in month April

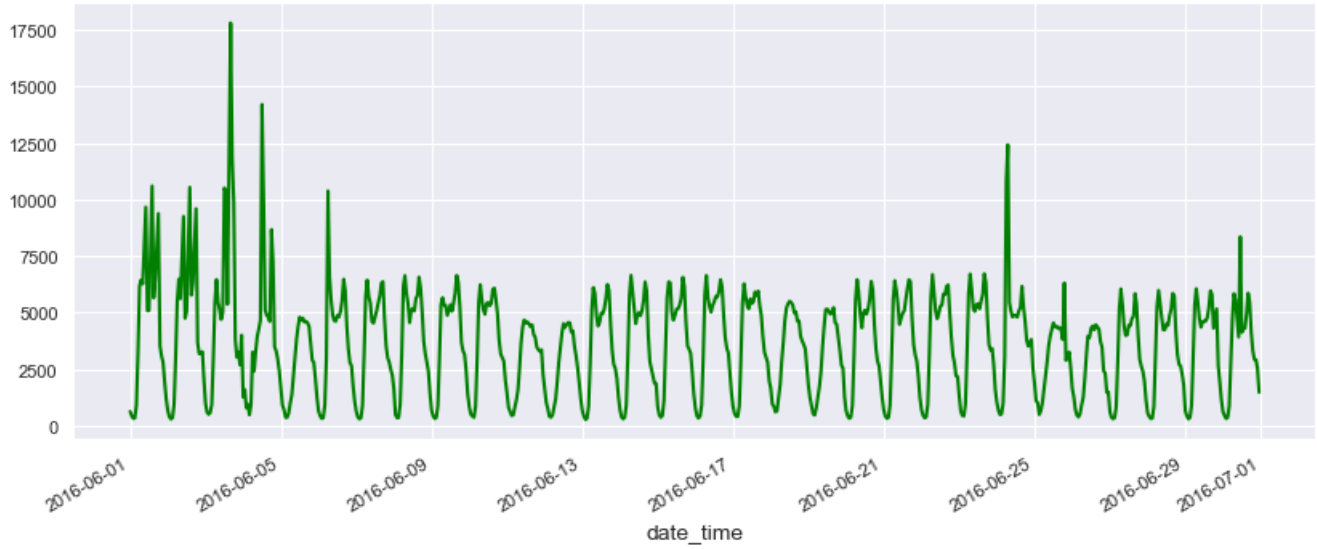




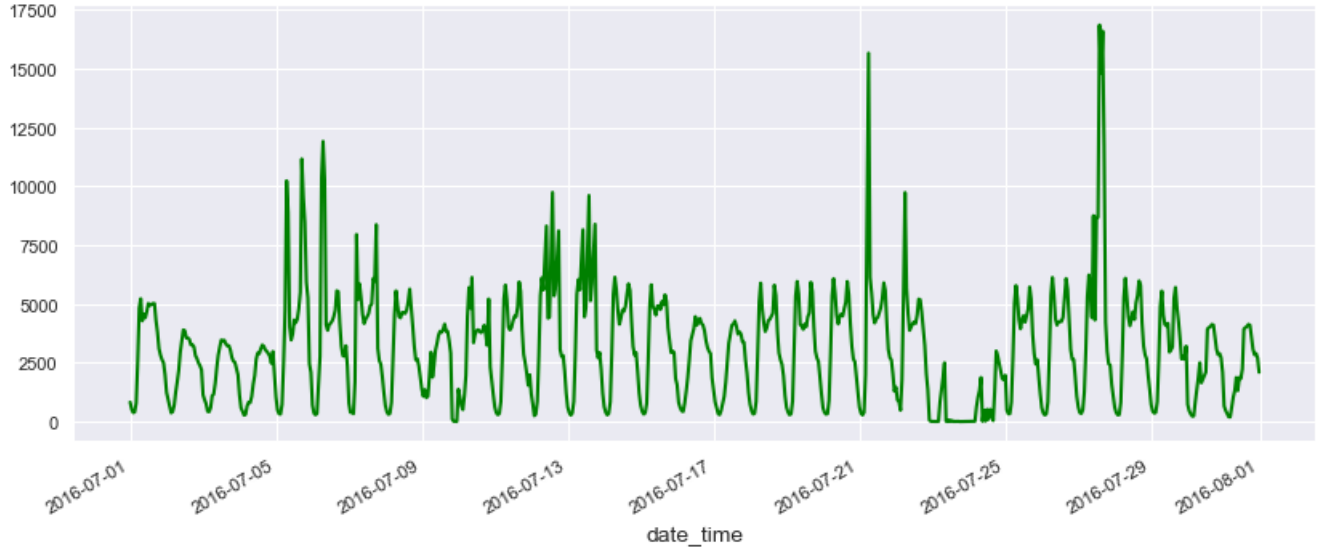
(2016) Plot of traffic volume per day in month May

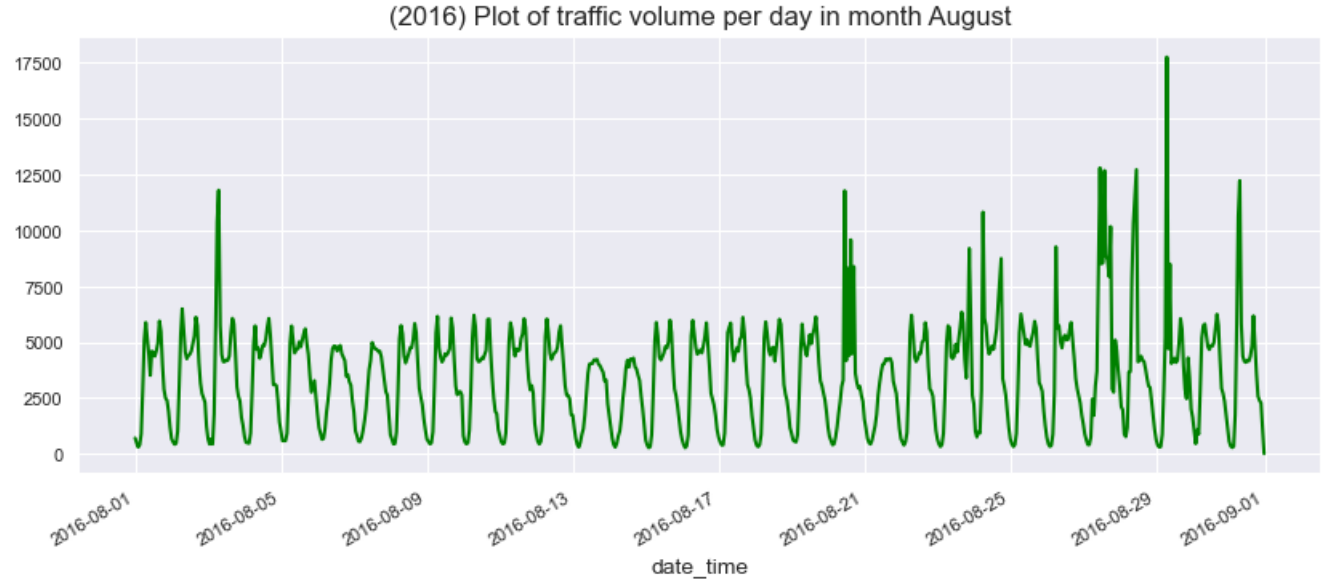


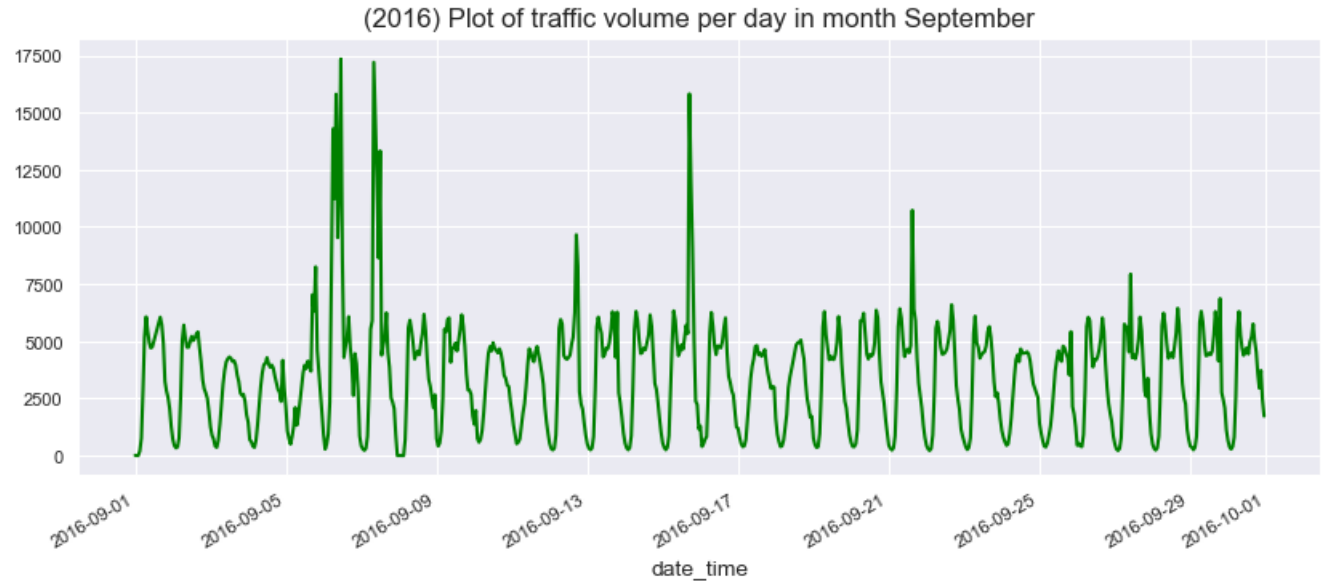
(2016) Plot of traffic volume per day in month June



(2016) Plot of traffic volume per day in month July

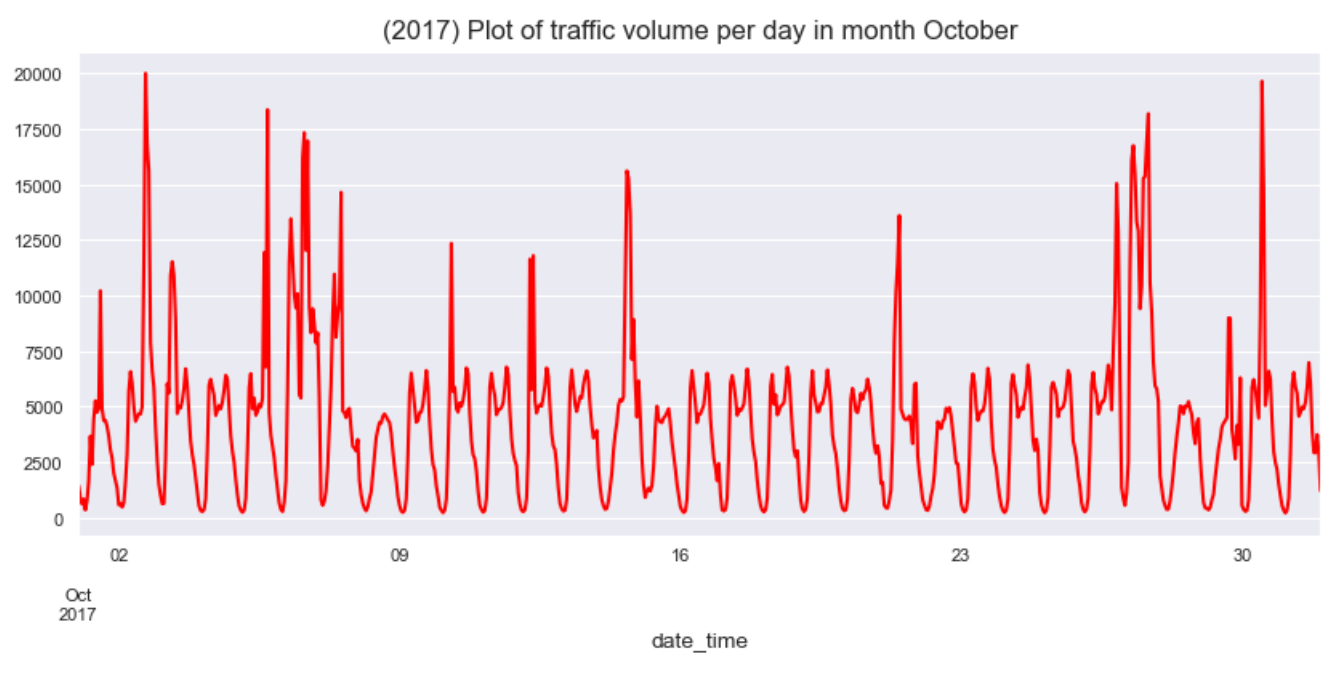


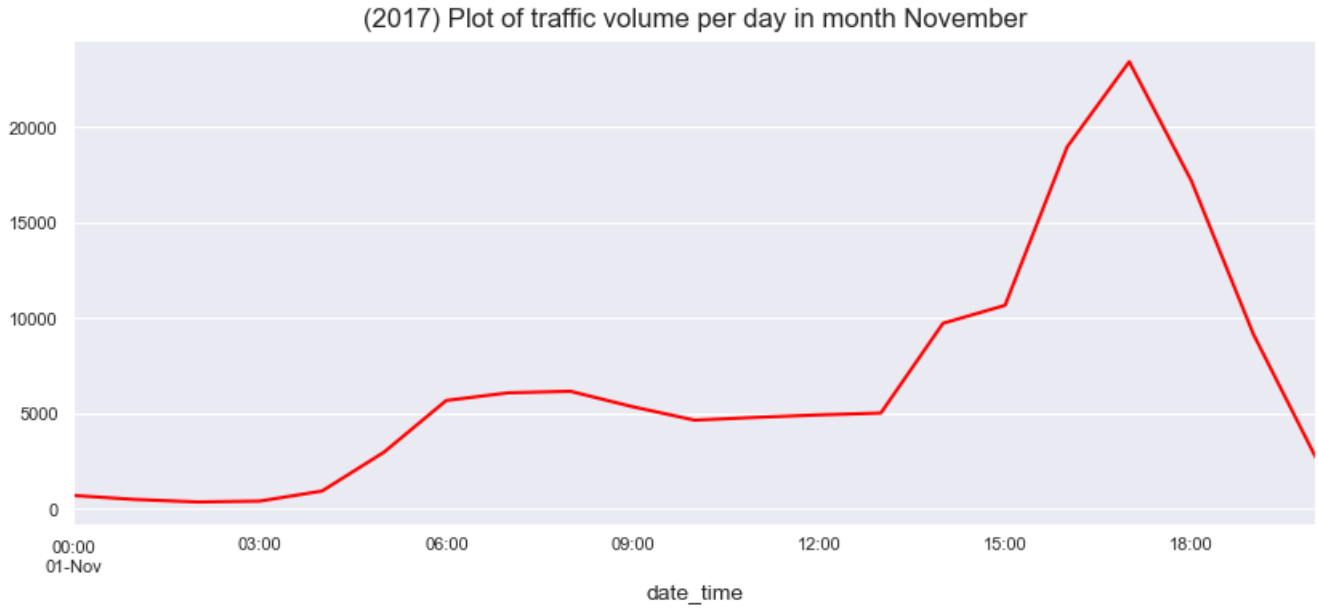


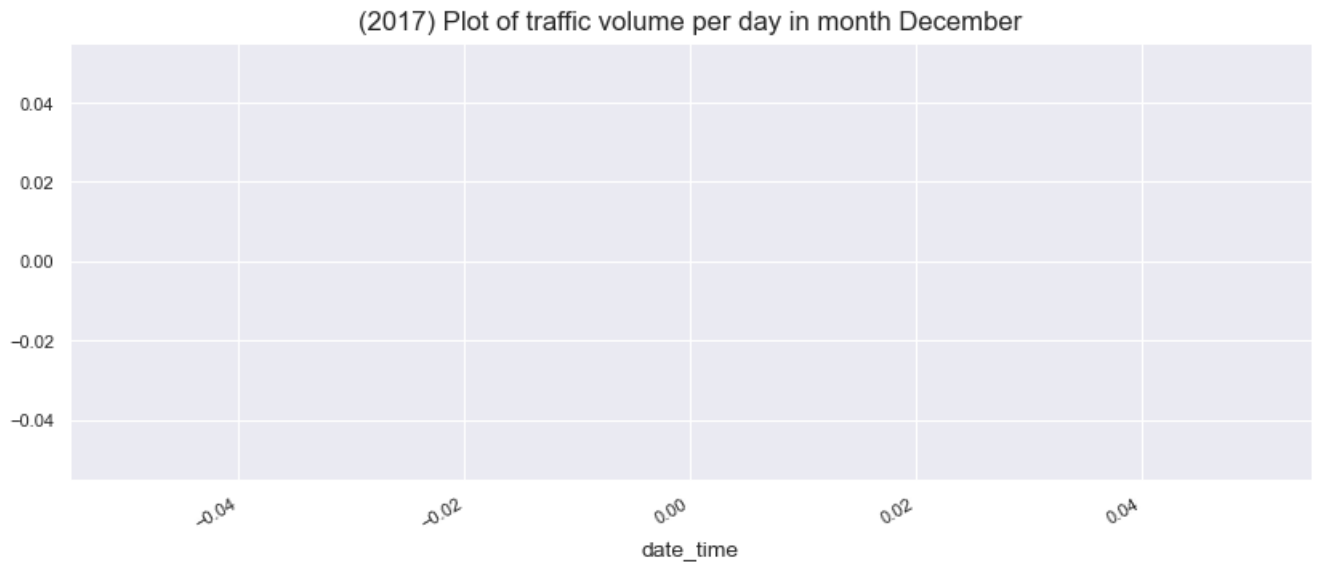


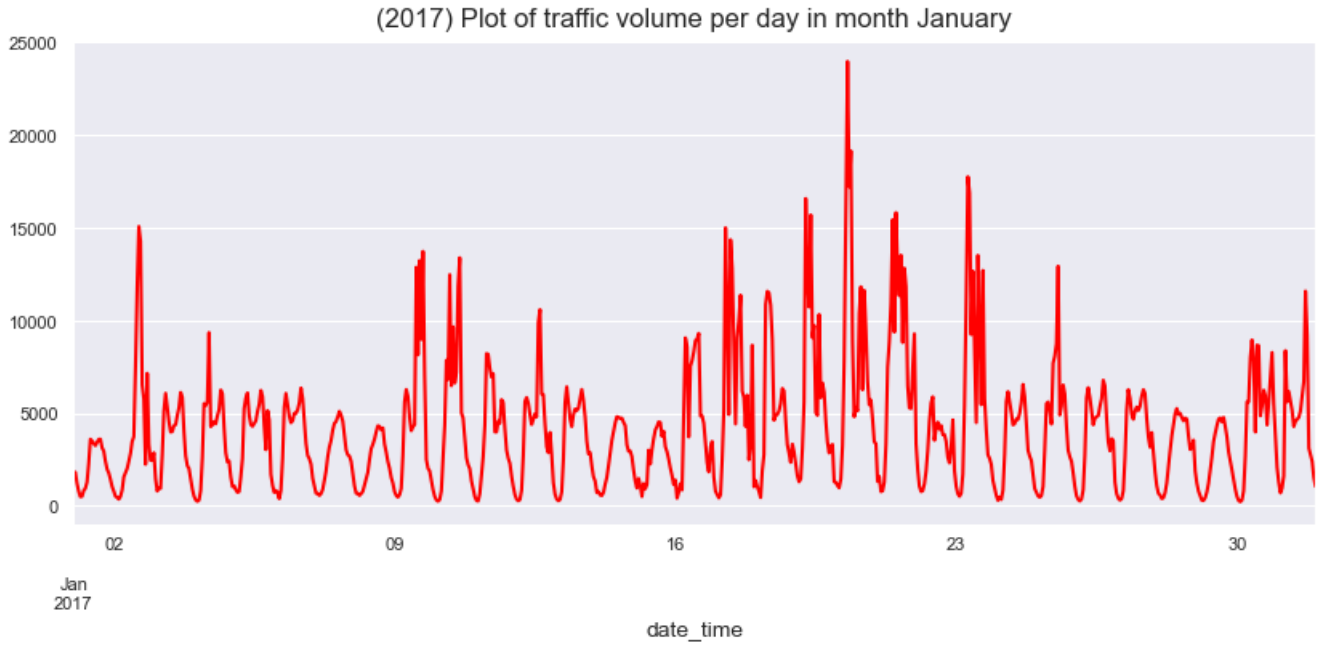
In [133]:

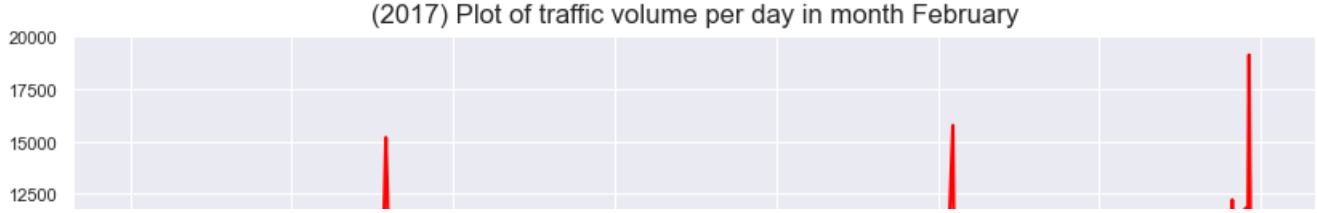
```
for i in mnths:

    y_2017.loc[y_2017['month']==i].groupby('date_time')['traffic_volume'].sum().plot(kind = 'line',fi
    plt.title("(2017) Plot of traffic volume per day in month "+str(i))
    plt.show()
```

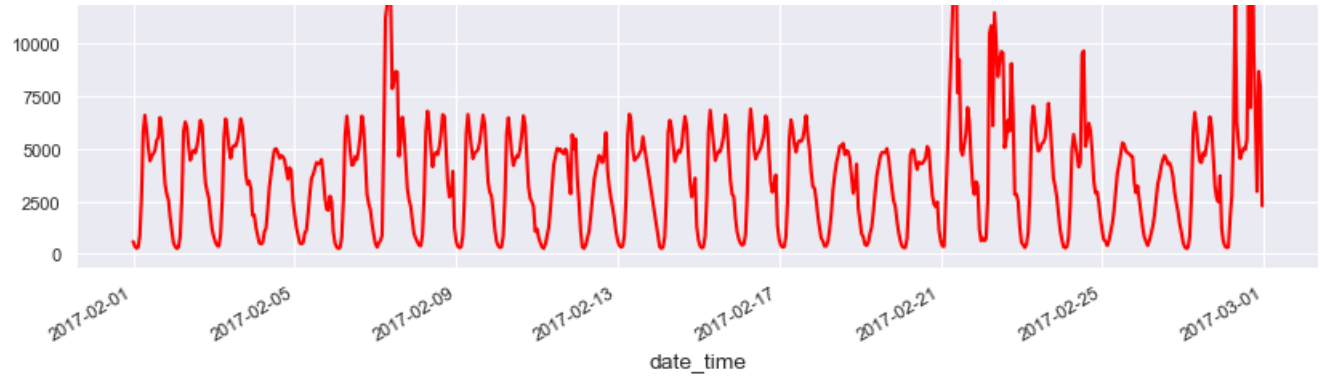



(2017) Plot of traffic volume per day in month November
20000
15000
10000
5000
0
00:00
01-Nov
03:00
06:00
09:00
12:00
15:00
18:00
date_time

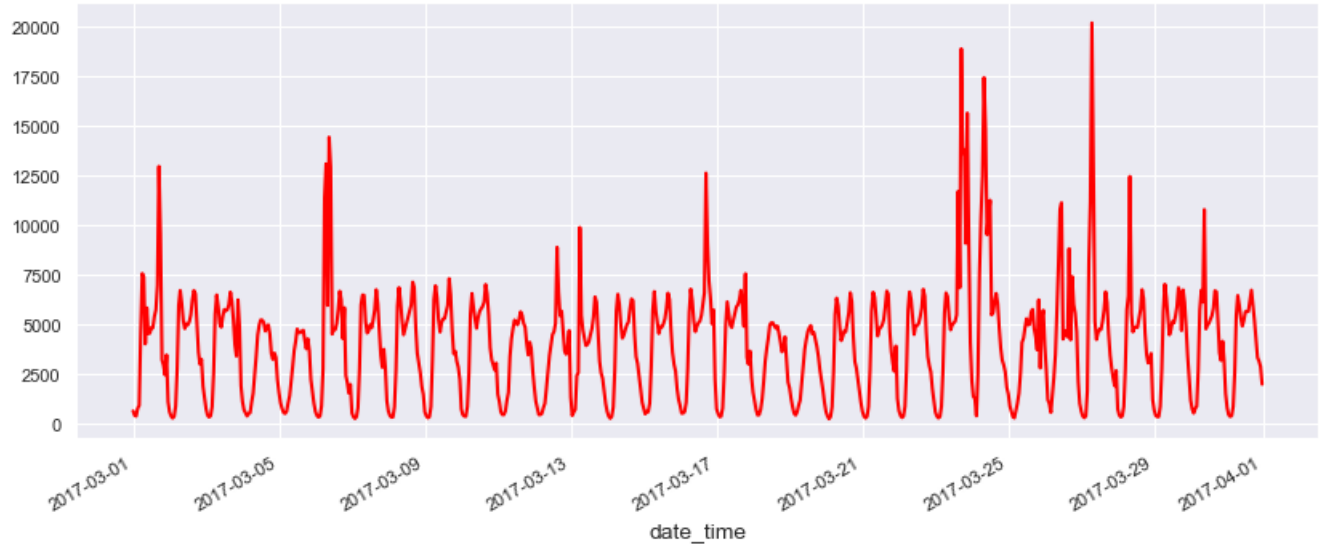

(2017) Plot of traffic volume per day in month December
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
date_time


(2017) Plot of traffic volume per day in month January
25000
20000
15000
10000
5000
0
02
09
16
23
30
Jan
2017
date_time

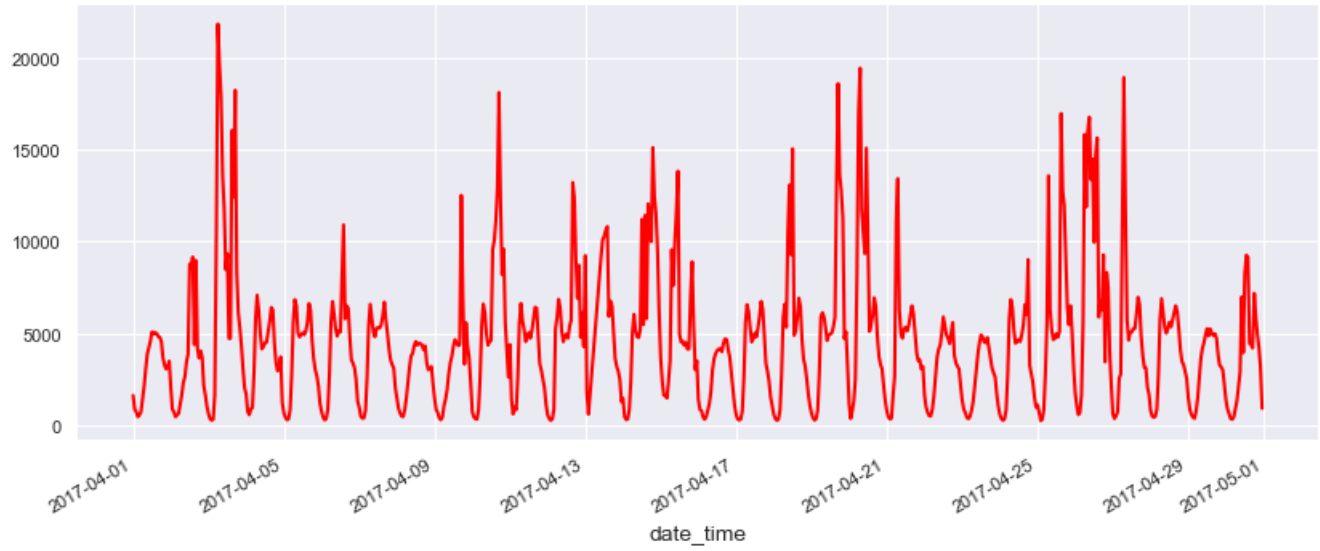

(2017) Plot of traffic volume per day in month February
20000
17500
15000
12500



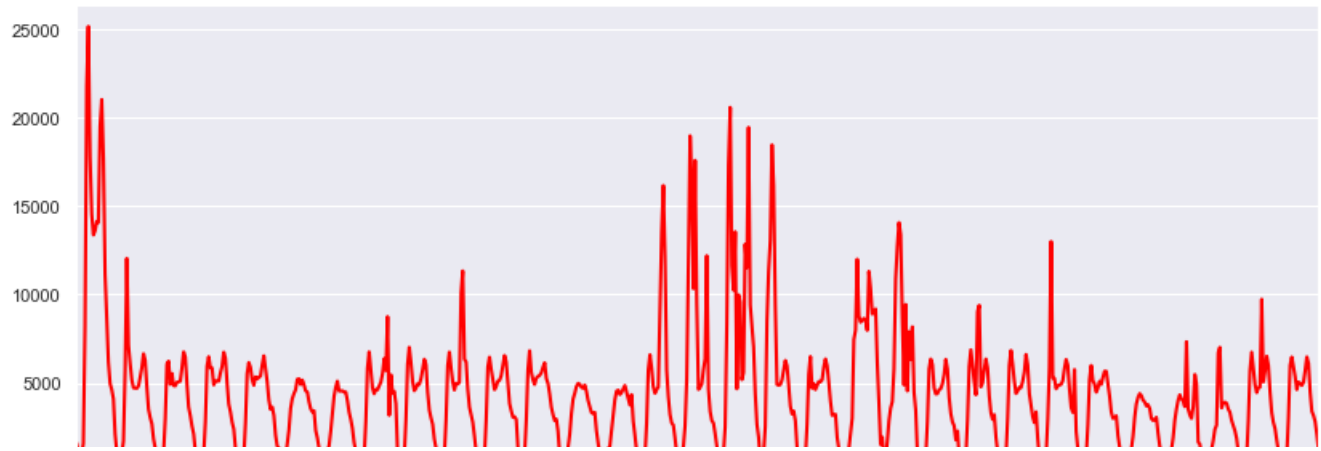
(2017) Plot of traffic volume per day in month March



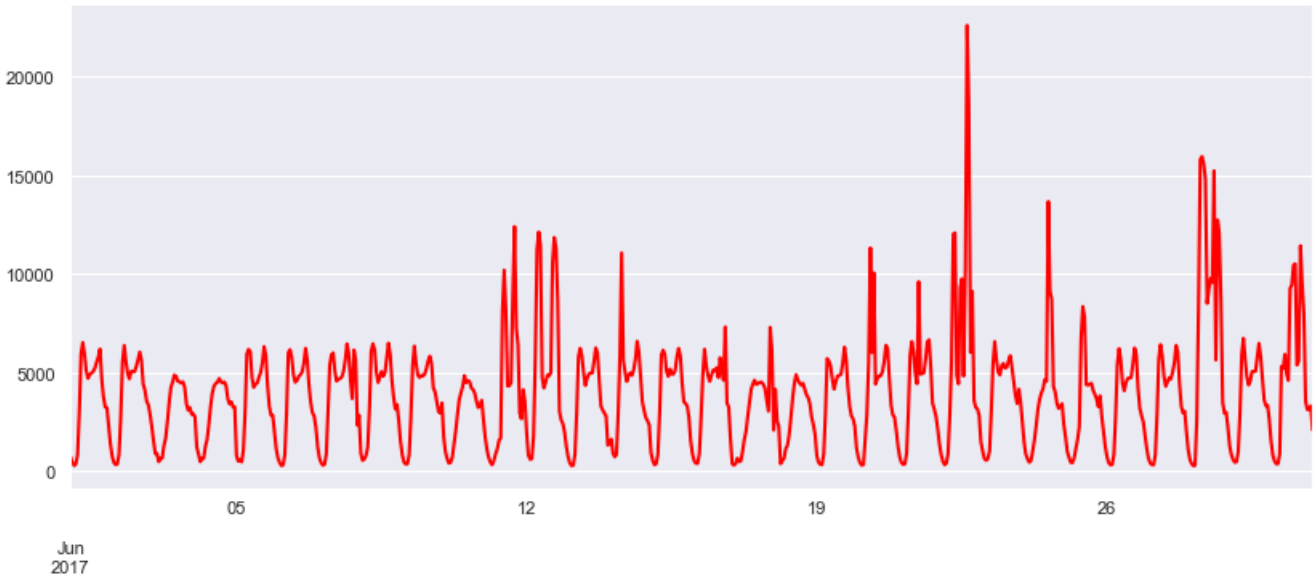
(2017) Plot of traffic volume per day in month April



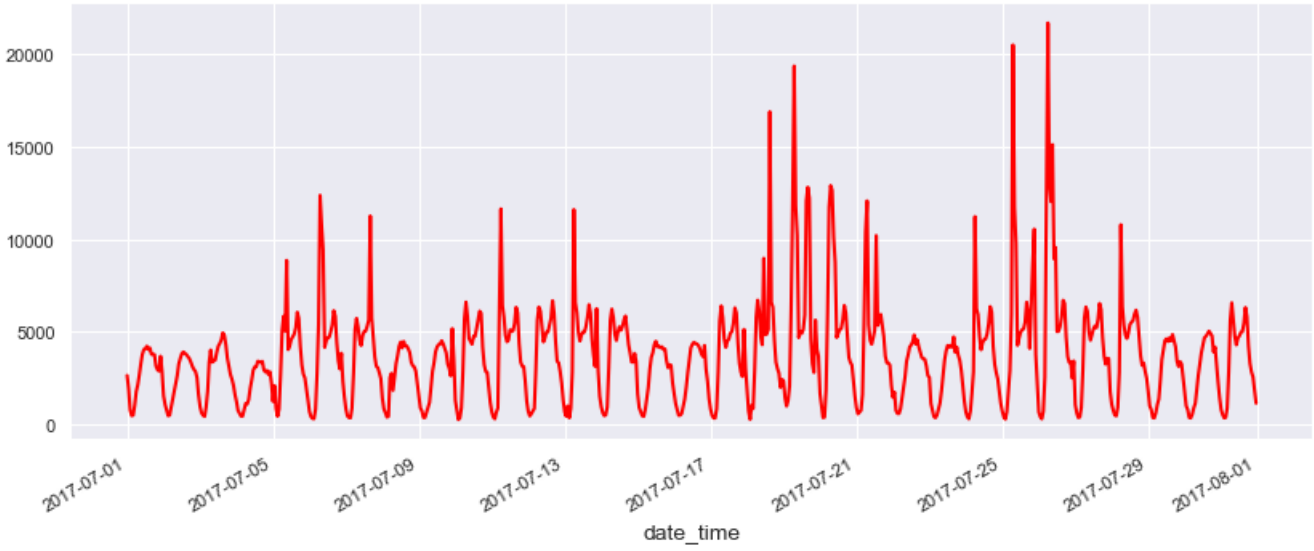
(2017) Plot of traffic volume per day in month May



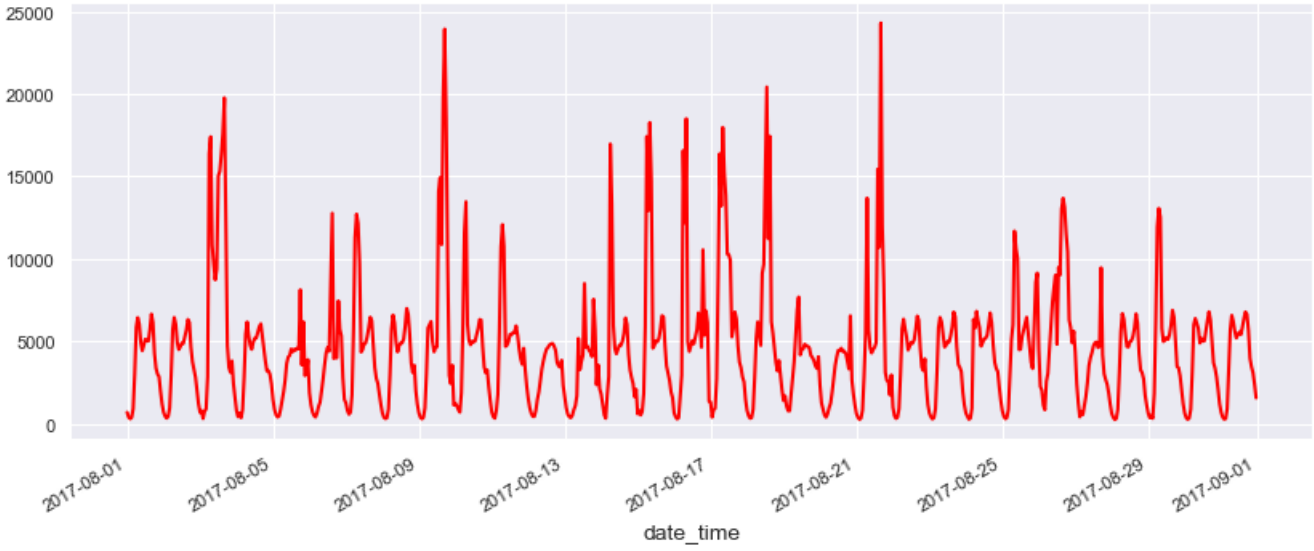
(2017) Plot of traffic volume per day in month June



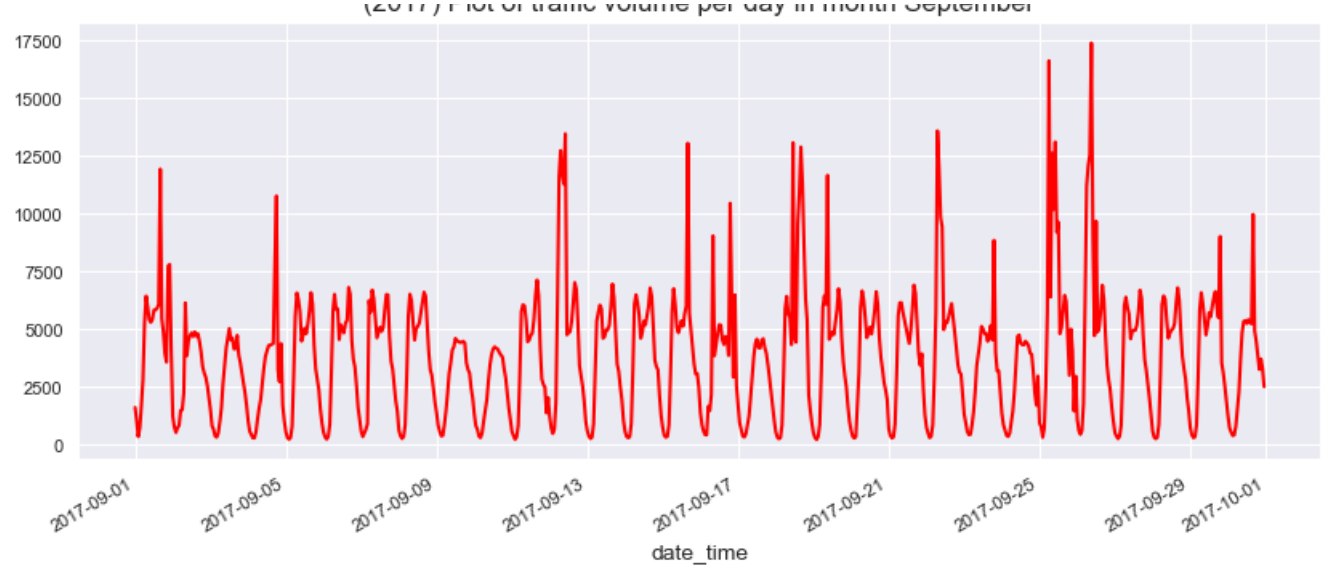
(2017) Plot of traffic volume per day in month July



(2017) Plot of traffic volume per day in month August



(2017) Plot of traffic volume per day in month September

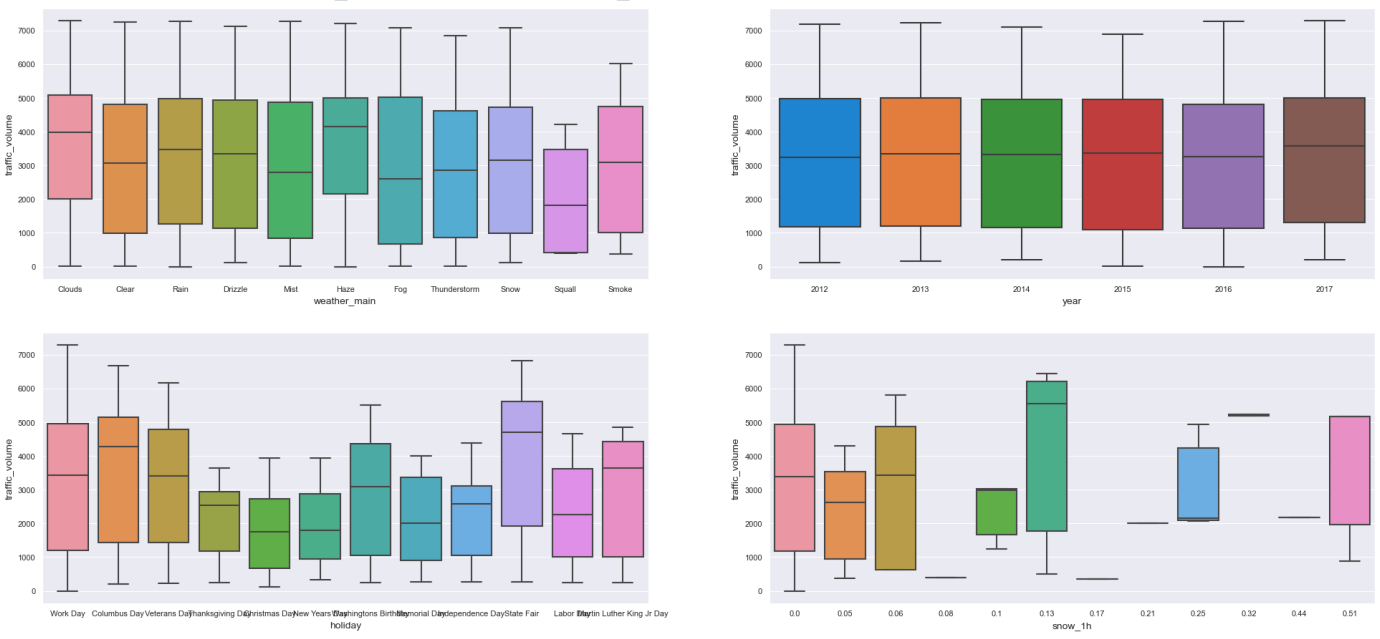


In [134]:

```
fig, axes = plt.subplots(2, 2, figsize=(24, 11))
sns.boxplot(ax=axes[0, 0], data=data, x='weather_main', y='traffic_volume')
sns.boxplot(ax=axes[0, 1], data=data, x='year', y='traffic_volume')
sns.boxplot(ax=axes[1,0], data=data, x='holiday', y='traffic_volume')
sns.boxplot(ax=axes[1, 1], data=data, x='snow_1h', y='traffic_volume')
```

Out[134]:

```
<AxesSubplot:xlabel='snow_1h', ylabel='traffic_volume'>
```

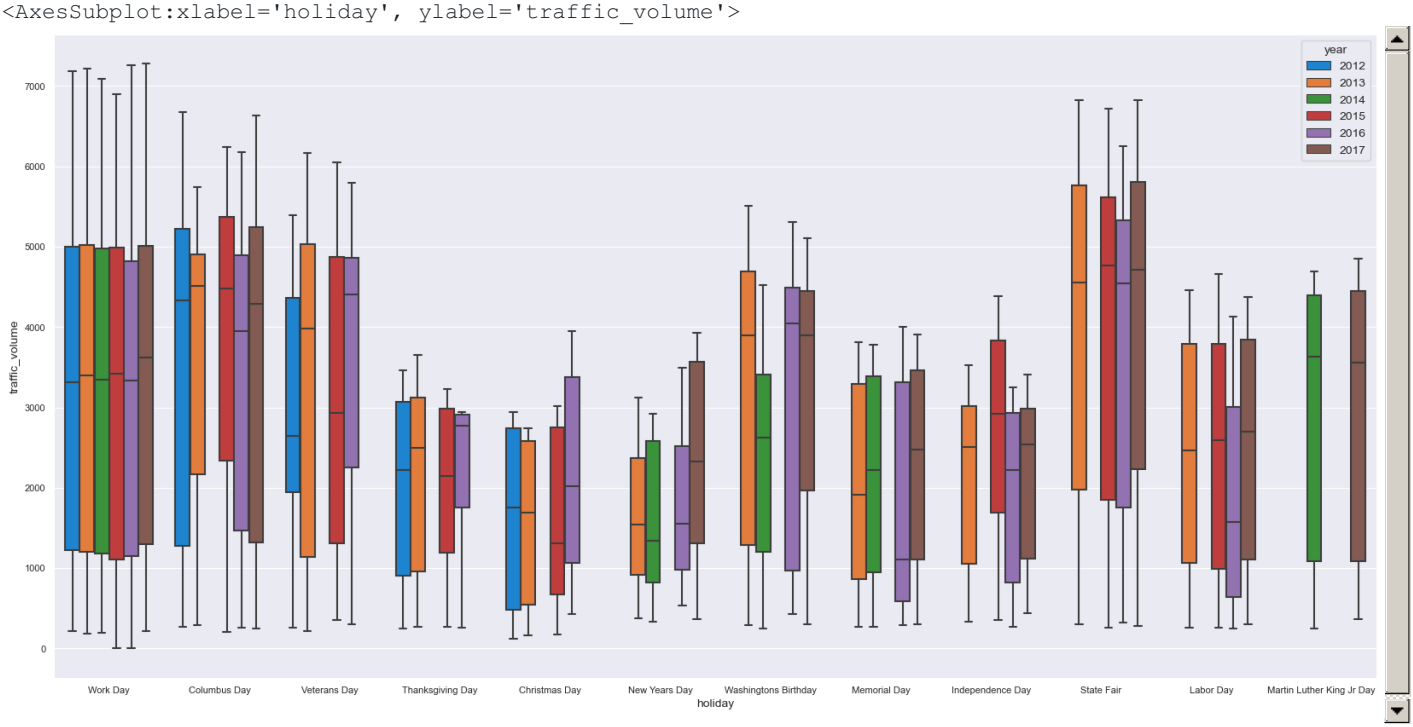


In [135]:

```
fig, ax = plt.subplots(figsize=(20,10))
sns.boxplot(data=data, x='holiday', y='traffic_volume',hue='year')
```

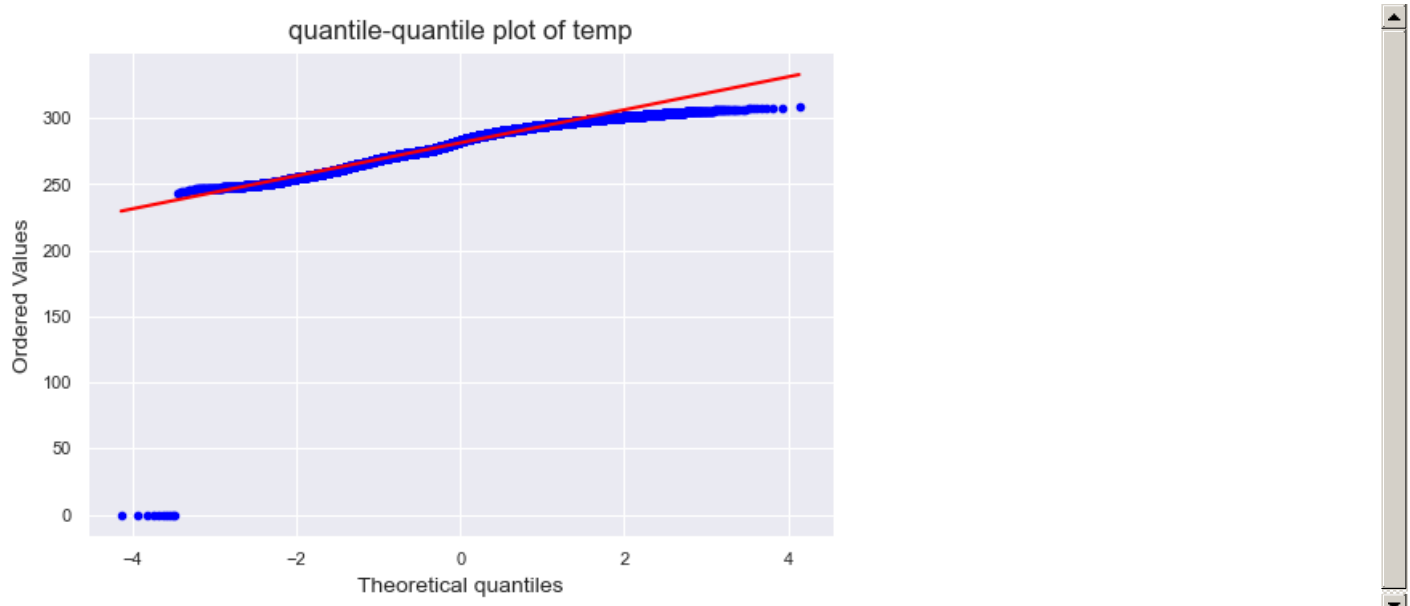
Out[135]:

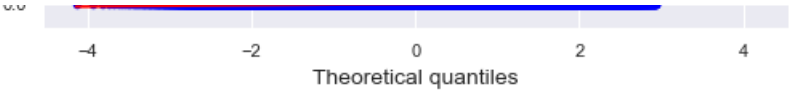
```
<AxesSubplot:xlabel='holiday', ylabel='traffic_volume'>
```

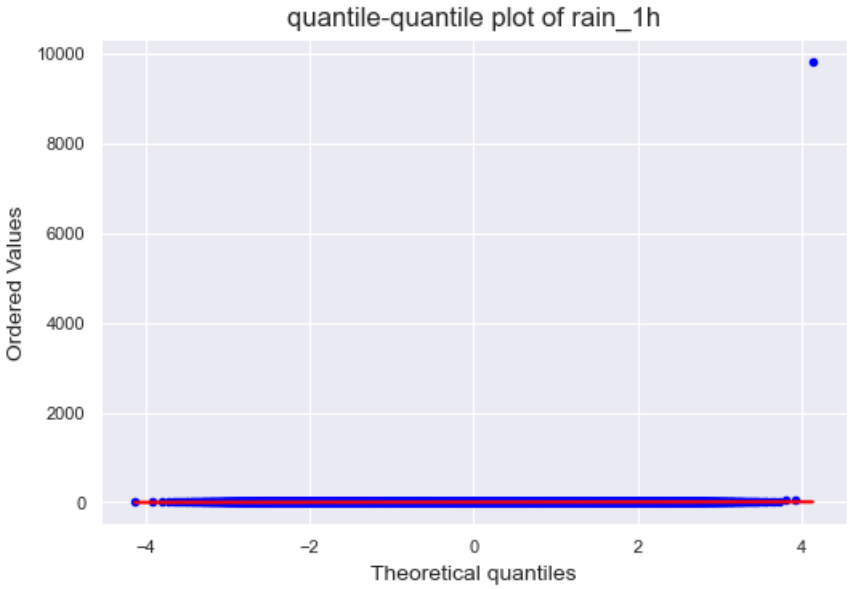


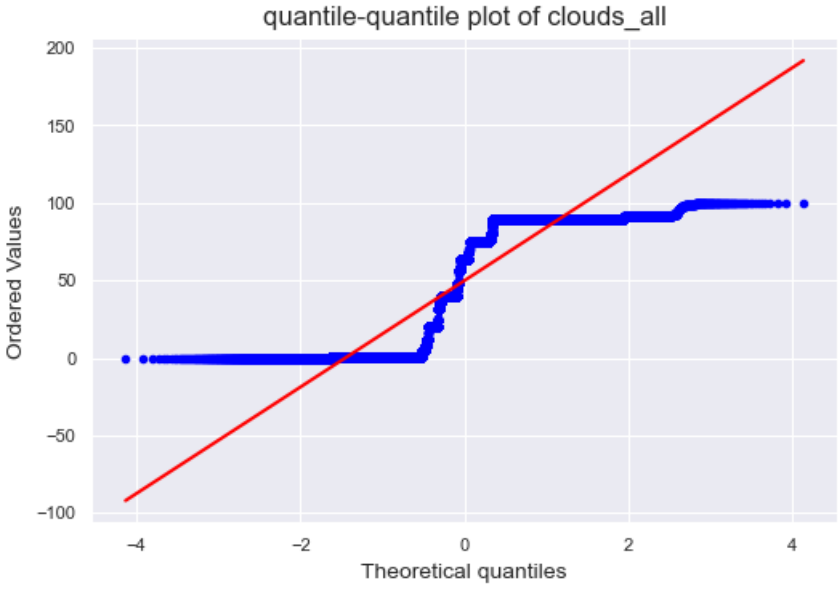
In [136]:

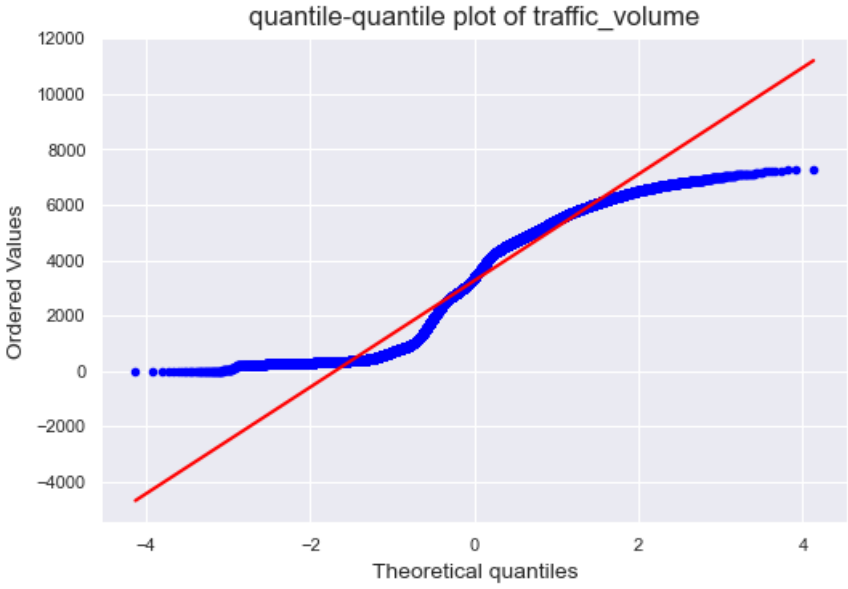
```
import scipy
cols = ['temp','snow_1h','rain_1h','clouds_all','traffic_volume']
for i in cols:
      scipy.stats.probplot(data[i], dist="norm", plot=plt)
      plt.title("quantile-quantile plot of "+i)
      plt.show()
```











In [137]:

```
import plotly.express as px
fig = px.sunburst(data, path=['year', 'weather_main'],title = "Plot of Season vs weather condition",color
fig.show()
```

In [138]:

```
# Replacing "0 K" to np.nan and Forward Filling values

data.loc[data.temp==0,'temp'] = np.nan

data['temp'].fillna(method='ffill',inplace=True)
```

In [139]:

```
def getDayTime(hour):

    if hour<4:
        return "Late Night"
    elif hour<8:
        return "Early Morning"
    elif hour<12:
        return "Morning"
    elif hour<16:
        return "Afternoon"
    elif hour<19:
        return "Evening"
    elif hour<24:
        return "Night"

data['day_time'] = data[['hour']].applymap(getDayTime)

data.head()
```

Out[139]:

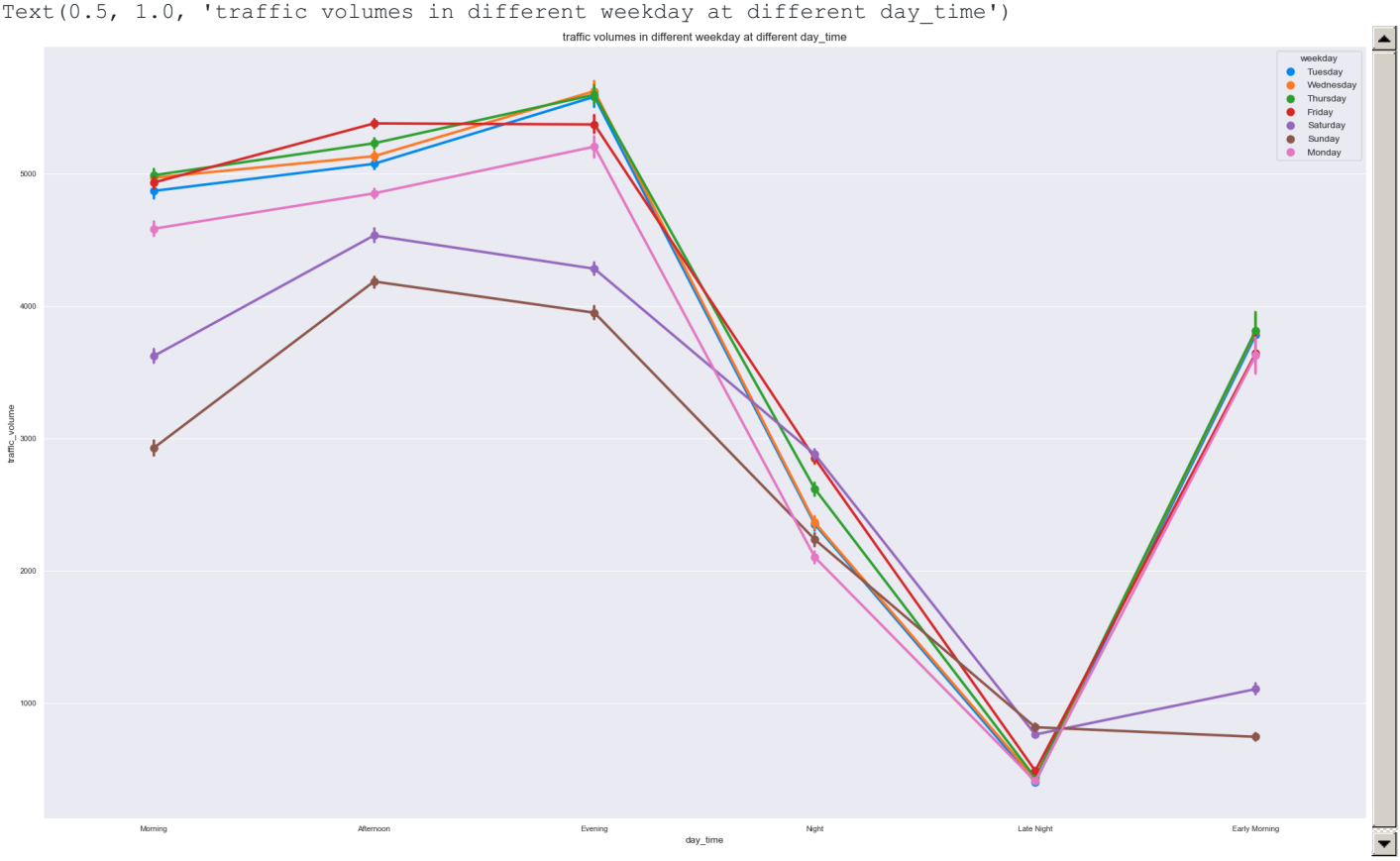
	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	day	month	weekday
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	2	October	Tuesday
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012	2	October	Tuesday
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	2012	2	October	Tuesday
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	2012	2	October	Tuesday
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	2012	2	October	Tuesday

In [140]:

```
plt.figure(figsize = (25,15))
sns.pointplot(x=  'day_time', y = 'traffic_volume', hue = 'weekday' , data = data)
plt.title("traffic volumes in different weekday at different day_time")
```

Out[140]:

```
Text(0.5, 1.0, 'traffic volumes in different weekday at different day_time')
```

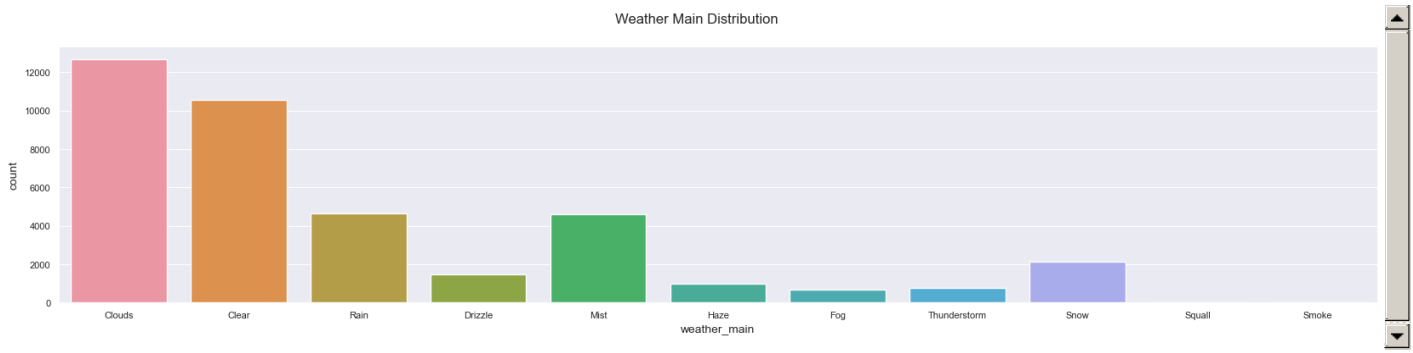


In [141]:

```
data.loc[data.rain_1h>2000,'rain_1h'] = np.nan
data['rain_1h'].fillna(method='ffill',inplace=True)
```

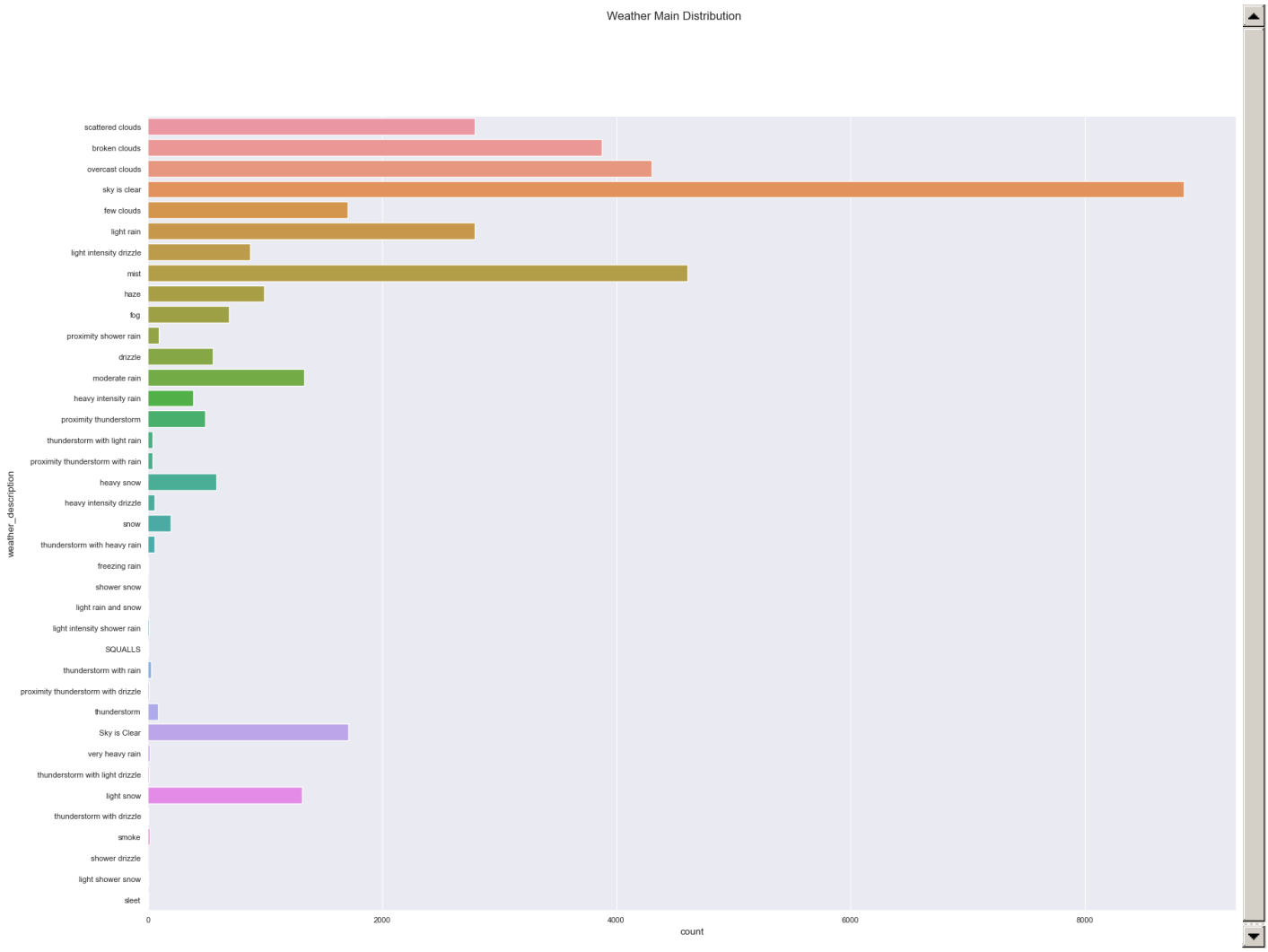
In [142]:

```
fig, ax = plt.subplots(figsize=(20,4))
fig.suptitle('Weather Main Distribution')

sns.countplot(x=data['weather_main'])

plt.show()
```



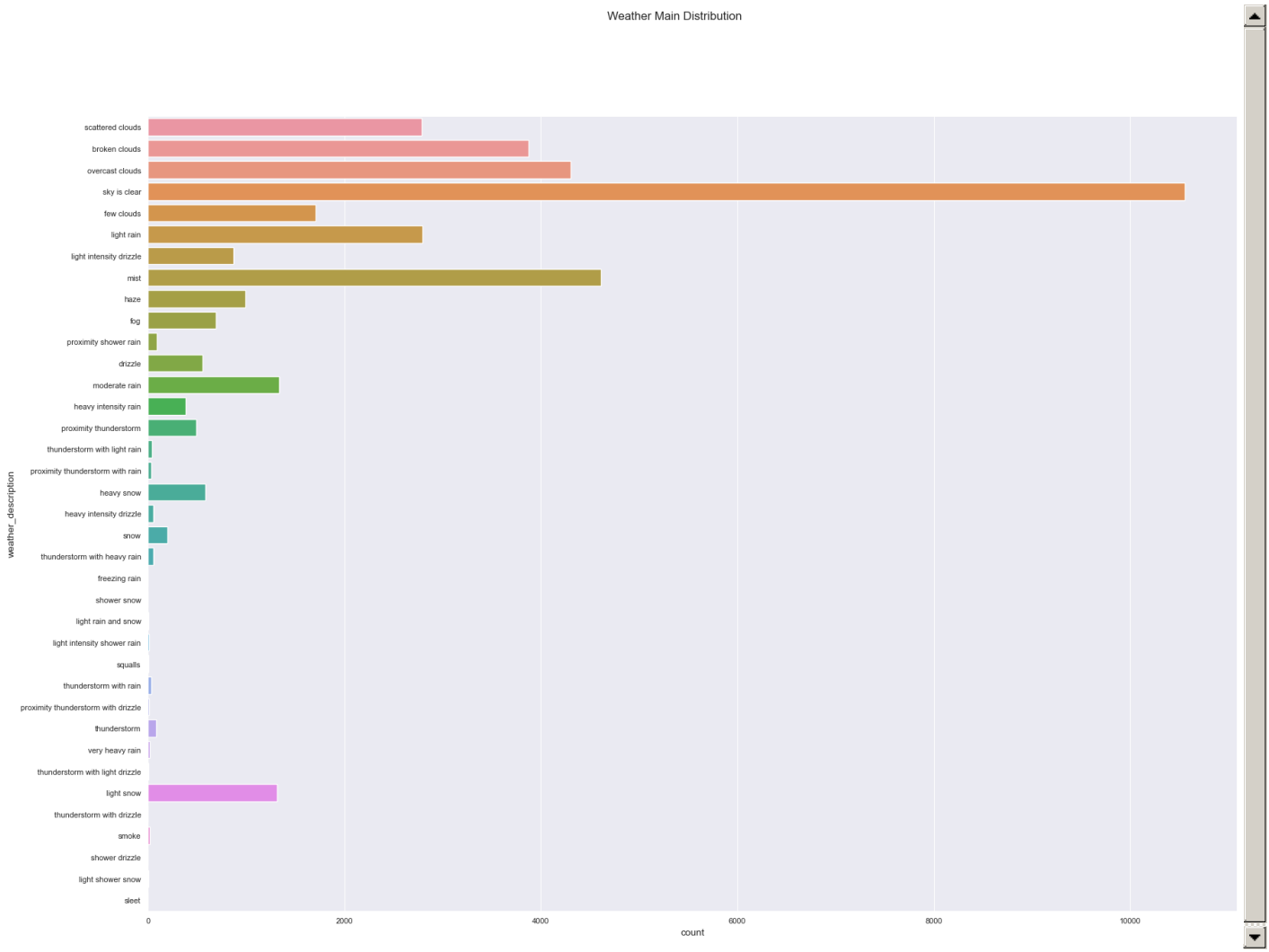
In [143]:

```
fig, ax = plt.subplots(figsize=(20,15))
fig.suptitle('Weather Main Distribution')

sns.countplot(y=data['weather_description'])

plt.show()
```



In [144]:

```
data['weather_description'] = data['weather_description'].map(lambda x: x.lower())

fig, ax = plt.subplots(figsize=(20,15))
fig.suptitle('Weather Main Distribution')

sns.countplot(y=data['weather_description'])

plt.show()
```



In [145]:

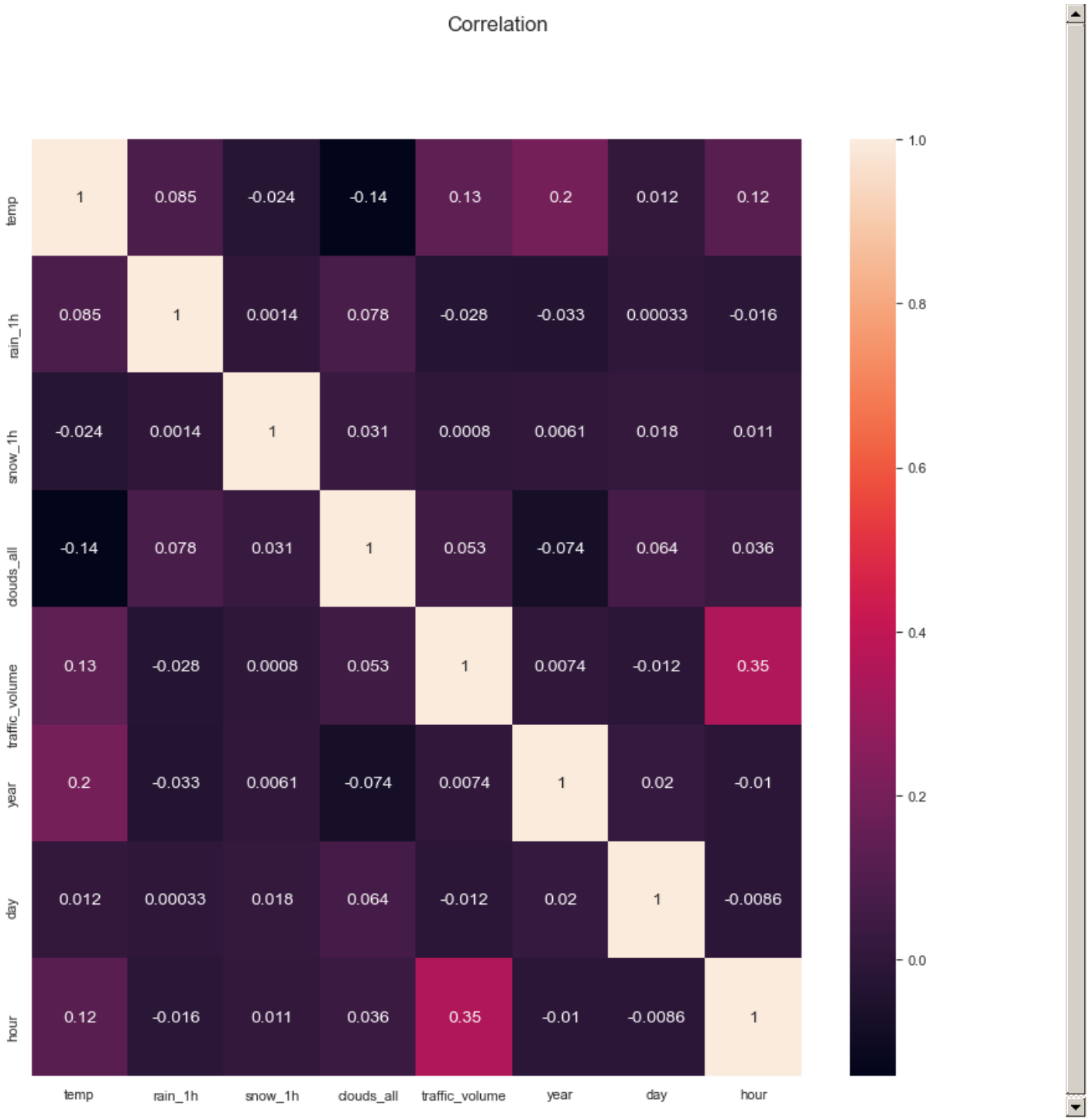
```
data.groupby(['weather_main','weather_description']).aggregate({'hour':'count','traffic_volume':'mean'}).
```

Out[145]:

	weather_main	weather_description	hour	traffic_volume
0	Clear	sky is clear	10560	3085.865909
1	Clouds	broken clouds	3879	3564.464037
2	Clouds	few clouds	1708	3619.433255
3	Clouds	overcast clouds	4302	3339.694561
4	Clouds	scattered clouds	2791	3875.658904
5	Drizzle	drizzle	554	3073.518051
6	Drizzle	heavy intensity drizzle	56	3206.375000
7	Drizzle	light intensity drizzle	871	3338.663605
8	Drizzle	shower drizzle	1	2010.000000
9	Fog	fog	693	2833.751804
10	Haze	haze	993	3574.350453
11	Mist	mist	4611	2951.615268
12	Rain	freezing rain	2	4314.000000
13	Rain	heavy intensity rain	387	3057.023256
14	Rain	light intensity shower rain	11	4351.545455
15	Rain	light rain	2795	3359.250089
16	Rain	moderate rain	1333	3171.570143
17	Rain	proximity shower rain	94	4501.202128
18	Rain	very heavy rain	18	2568.833333
19	Smoke	smoke	18	3103.722222
20	Snow	heavy snow	587	3085.862010
21	Snow	light rain and snow	6	3961.166667
22	Snow	light shower snow	4	4570.750000
23	Snow	light snow	1318	3045.698027
24	Snow	shower snow	1	5664.000000
25	Snow	sleet	2	3882.000000
26	Snow	snow	199	2808.035176
27	Squall	squalls	4	2061.750000
28	Thunderstorm	proximity thunderstorm	489	3005.149284
29	Thunderstorm	proximity thunderstorm with drizzle	12	3131.500000
30	Thunderstorm	proximity thunderstorm with rain	38	2507.026316
31	Thunderstorm	thunderstorm	88	2701.500000
32	Thunderstorm	thunderstorm with drizzle	2	2297.000000
33	Thunderstorm	thunderstorm with heavy rain	56	2603.857143
34	Thunderstorm	thunderstorm with light drizzle	8	2463.375000
35	Thunderstorm	thunderstorm with light rain	41	2673.926829
36	Thunderstorm	thunderstorm with rain	31	3456.322581

In [146]:

```
fig, ax = plt.subplots(figsize=(10,10))
fig.suptitle('Correlation')

sns.heatmap(data.corr(),annot=True)

plt.show()
```



In [147]:

```
train=data.reset_index(drop=True)
```

In [148]:

```
train.head()
```

Out[148]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	day	month	weekday
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	2	October	Tuesday
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012	2	October	Tuesday
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	2012	2	October	Tuesday
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	2012	2	October	Tuesday
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	2012	2	October	Tuesday

#

Preprocessing data

In [515]:

```
raw=pd.read_csv('Train.csv')
raw['date_time'] = pd.to_datetime(raw['date_time'])
train = raw.reset_index(drop=True)
train.head()
```

Out[515]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
0	2012-10-02 09:00:00	None	288.28	0.0	0.0	40	Clouds	scattered clouds	5545
1	2012-10-02 10:00:00	None	289.36	0.0	0.0	75	Clouds	broken clouds	4516
2	2012-10-02 11:00:00	None	289.58	0.0	0.0	90	Clouds	overcast clouds	4767
3	2012-10-02 12:00:00	None	290.13	0.0	0.0	90	Clouds	overcast clouds	5026
4	2012-10-02 13:00:00	None	291.14	0.0	0.0	75	Clouds	broken clouds	4918

In [516]:

```
def getLastDayValue(df, feature, n_days=1):

    for days in range(1,n_days+1):
        past_days = (df.date_time - timedelta(days=days)).values
        col = f'{feature}_{days}_days_before'
        past_days_in_df = df.loc[df.date_time.isin(past_days),'date_time']
        past_days_value = df.loc[df.date_time.isin(past_days),feature].values

        mapping = dict(zip((past_days_in_df+timedelta(days=days)).values, past_days_value))

        df[col] = df['date_time'].map(mapping)

    return df
```

In [517]:

```
def getNextDayValue(df, feature, n_days=1):

    for days in range(1,n_days+1):
        past_days = (df.date_time + timedelta(days=days)).values
        col = f'{feature}_{days}_days_after'
        past_days_in_df = df.loc[df.date_time.isin(past_days),'date_time']
        past_days_value = df.loc[df.date_time.isin(past_days),feature].values

        mapping = dict(zip((past_days_in_df-timedelta(days=days)).values, past_days_value))
```

```
        df[col] = df['date_time'].map(mapping)

    return df
```

In [518]:

```
def getLastHourValue(df, feature, n_hours=1):

    for hours in range(1,n_hours+1):
        past_hours = (df.date_time - timedelta(hours=hours)).values
        col = f'{feature}_{hours}_hours_before'
        past_hours_in_df = df.loc[df.date_time.isin(past_hours),'date_time']
        past_hours_value = df.loc[df.date_time.isin(past_hours),feature].values

        mapping = dict(zip((past_hours_in_df+timedelta(hours=hours)).values, past_hours_value))

        df[col] = df['date_time'].map(mapping)

    return df
```

In [519]:

```
### Holiday ###

def preprocessHoliday(df):

    """Marking All the samples collected on a Holiday as its respective Holiday.
    Because only one sample is marked as a Holiday for all Holidays in the raw data.
    For Ex: if there is a Holiday on 2012-02-08 then sample with date_time=2012-02-08 00:00:00 is marked

    isHoliday = df.holiday!='None'
    date = df.date_time[isHoliday].dt.date.values
    holiday = df.holiday[isHoliday].values

    mapping = dict(zip(date,holiday))

    df['holiday'] = df['date_time'].dt.date.map(mapping)
    df.loc[df['holiday'].isna(),'holiday'] = 'Work Day'

    return df
```

In [520]:

```
train = preprocessHoliday(train)
test=pd.read_csv('Test.csv')
test=test.reset_index(drop=True)
test['date_time'] = pd.to_datetime(test['date_time'])
test=preprocessHoliday(test)
```

In [521]:

```
train.head(3)
```

Out[521]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767

In [522]:

```
isHoliday = lambda x: 0 if x == 'Work Day' else 1
train['isHoliday'] = train['holiday'].map(isHoliday)
test['isHoliday'] = test['holiday'].map(isHoliday)
```

In [523]:

```
def getWasYesterdayAHoliday(df):

    past_days = (df.date_time - timedelta(days=1)).dt.date.values
    col = 'yesterdayHoliday'
    past_days_in_df = df.loc[df.date_time.dt.date.isin(past_days),'date_time']
    past_days_value = df.loc[df.date_time.dt.date.isin(past_days),'isHoliday'].values

    mapping = dict(zip((past_days_in_df+timedelta(days=1)).dt.date.values, past_days_value))

    df[col] = df['date_time'].dt.date.map(mapping)

    return df
```

In [524]:

```
train = getWasYesterdayAHoliday(train)
test = getWasYesterdayAHoliday(test)
```

In [525]:

```
def getIsTomorrowAHoliday(df):

    past_days = (df.date_time + timedelta(days=1)).dt.date.values
    col = 'tomorrowHoliday'
    past_days_in_df = df.loc[df.date_time.dt.date.isin(past_days),'date_time']
    past_days_value = df.loc[df.date_time.dt.date.isin(past_days),'isHoliday'].values

    mapping = dict(zip((past_days_in_df-timedelta(days=1)).dt.date.values, past_days_value))

    df[col] = df['date_time'].dt.date.map(mapping)

    return df

train = getIsTomorrowAHoliday(train)
test = getIsTomorrowAHoliday(test)

train.head()
```

Out[525]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	yesterdayHoliday	tom
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	0	NaN	
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	0	NaN	
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	0	NaN	
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	0	NaN	
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	0	NaN	

In [526]:

```
train.loc[train.temp==0,'temp'] = np.nan
train.temp.fillna(method='ffill',inplace=True)

### Last few day's temperature at this time? ###
train = getLastDayValue(train, feature = 'temp')
test = getLastDayValue(test, feature = 'temp')

### Last few hour's temperature? ###
train = getLastHourValue(train, feature = 'temp', n_hours=3)
test = getLastHourValue(test, feature = 'temp', n_hours=3)

train.head()
```

Out[526]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	yesterdayHoliday	tom
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	0	NaN	
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	0	NaN	
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	0	NaN	
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	0	NaN	
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	0	NaN	

In [527]:

```
train[train.rain_1h>1000]
```

Out[527]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	yesterdayHoliday
24872	2016-07-11 17:00:00	Work Day	302.11	9831.3	0.0	75	Rain	very heavy rain	5535	0	0.0

In [528]:

```
train[train.date_time.dt.date.isin(train.loc[train.rain_1h>1000,'date_time'].dt.date.values)]
```

Out[528]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	yesterdayHoliday
24855	2016-07-11 00:00:00	Work Day	296.22	0.00	0.0	90	Clouds	overcast clouds	621	0	0.0
24856	2016-07-11 01:00:00	Work Day	295.80	0.00	0.0	90	Clouds	overcast clouds	376	0	0.0
24857	2016-07-11 02:00:00	Work Day	295.62	0.00	0.0	90	Clouds	overcast clouds	291	0	0.0
24858	2016-07-11 03:00:00	Work Day	295.24	0.00	0.0	75	Clouds	broken clouds	342	0	0.0
24859	2016-07-11 04:00:00	Work Day	295.40	0.00	0.0	75	Clouds	broken clouds	848	0	0.0
24860	2016-07-11 05:00:00	Work Day	295.37	0.00	0.0	75	Clouds	broken clouds	2780	0	0.0
24861	2016-07-11 06:00:00	Work Day	295.05	0.00	0.0	90	Rain	light rain	5211	0	0.0
24862	2016-07-11 07:00:00	Work Day	294.99	0.00	0.0	75	Rain	light rain	5803	0	0.0
24863	2016-07-11 08:00:00	Work Day	294.83	0.00	0.0	90	Rain	light rain	5157	0	0.0
24864	2016-07-11 09:00:00	Work Day	294.79	0.00	0.0	90	Clouds	overcast clouds	4016	0	0.0
24865	2016-07-11 10:00:00	Work Day	295.13	0.00	0.0	90	Thunderstorm	proximity thunderstorm	3891	0	0.0

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	yesterdayHoliday
24866	2016-07-11 11:00:00	Work Day	296.45	0.00	0.0	90	Thunderstorm	proximity thunderstorm	3998	0	0.0
24867	2016-07-11 12:00:00	Work Day	297.73	0.00	0.0	90	Rain	light rain	4273	0	0.0
24868	2016-07-11 13:00:00	Work Day	299.15	0.00	0.0	75	Thunderstorm	proximity thunderstorm	4475	0	0.0
24869	2016-07-11 14:00:00	Work Day	300.18	0.00	0.0	90	Clouds	overcast clouds	4456	0	0.0
24870	2016-07-11 15:00:00	Work Day	300.81	0.00	0.0	75	Thunderstorm	proximity thunderstorm	4858	0	0.0
24871	2016-07-11 16:00:00	Work Day	301.48	0.00	0.0	75	Thunderstorm	proximity thunderstorm	5934	0	0.0
24872	2016-07-11 17:00:00	Work Day	302.11	9831.30	0.0	75	Rain	very heavy rain	5535	0	0.0
24873	2016-07-11 18:00:00	Work Day	302.54	0.00	0.0	75	Thunderstorm	proximity thunderstorm	3900	0	0.0
24874	2016-07-11 19:00:00	Work Day	302.39	0.00	0.0	75	Clouds	broken clouds	2856	0	0.0
24875	2016-07-11 20:00:00	Work Day	302.45	0.00	0.0	75	Thunderstorm	thunderstorm	2506	0	0.0
24876	2016-07-11 21:00:00	Work Day	301.70	0.00	0.0	75	Thunderstorm	thunderstorm with light rain	2062	0	0.0
24877	2016-07-11 22:00:00	Work Day	300.35	0.51	0.0	75	Rain	light rain	1544	0	0.0
24878	2016-07-11 23:00:00	Work Day	299.61	0.00	0.0	90	Thunderstorm	proximity thunderstorm	995	0	0.0
24879	2016-07-11 23:00:00	Work Day	299.61	0.00	0.0	90	Rain	light rain	995	0	0.0

In [529]:

```
train.loc[train.rain_1h>1000,'rain_1h'] = np.nan
train.rain_1h.fillna(method='ffill',inplace=True)

### Last few hour's rain? ###
train = getLastHourValue(train, feature = 'rain_1h', n_hours=3)
test = getLastHourValue(test, feature = 'rain_1h', n_hours=3)

### Last few hour's snow? ###
train = getLastHourValue(train, feature = 'snow_1h', n_hours=3)
test = getLastHourValue(test, feature = 'snow_1h', n_hours=3)

train.head()
```

Out[529]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	...	temp_1_days_befc
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	0	...	N
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	0	...	N
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	0	...	N
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	0	...	N
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	0	...	N

5 rows × 22 columns

In [530]:

```
train = getLastHourValue(train, feature = 'clouds_all', n_hours=3)
test = getLastHourValue(test, feature = 'clouds_all', n_hours=3)

train.head()
```

Out[530]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	...	temp_3_hours_bef
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	0	...	N
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	0	...	N
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	0	...	N
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	0	...	288
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	0	...	289

5 rows × 25 columns

In [531]:

```
def getDateTimeFeatures(df):
    df['day'] = df.date_time.dt.day
    df['month'] = df.date_time.dt.month
    df['year'] = df.date_time.dt.year
    df['weekday'] = df.date_time.dt.weekday  # Monday is denoted by 0 and Sunday is denoted by 6
    df['hour'] = df.date_time.dt.hour

    # Binning hour feature
    def getDayTime(hour):

        if hour<4:
            return "Late Night"
        elif hour<8:
            return "Early Morning"
        elif hour<12:
            return "Morning"
        elif hour<16:
            return "Afternoon"
        elif hour<19:
            return "Evening"
        elif hour<24:
```

```
            return "Night"

    df['day_time'] = df['hour'].map(getDayTime)

    df['is_weekend'] = df['weekday'].map(lambda x: 1 if x in ['Sunday','Saturday'] else 0)

    return df
train = getDateTimeFeatures(train)
test = getDateTimeFeatures(test)

train.head(5)
```

Out[531]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	...	clouds_all_1_hours
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	0	...	
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	0	...	
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	0	...	
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	0	...	
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	0	...	

5 rows × 32 columns

In [532]:

```
numeric = train.select_dtypes(exclude='object').copy()
for col in numeric.columns[numeric.isna().any(axis=0)]:

    train[f'unknown_{col}'] = 0
    train.loc[train[col].isna(),f'unknown_{col}'] = 1
    train.loc[train[col].isna(), col] = -1

    test[f'unknown_{col}'] = 0
    test.loc[test[col].isna(),f'unknown_{col}'] = 1
    test.loc[test[col].isna(), col] = -1
cate = train.select_dtypes(include='object').copy()
for col in cate.columns[cate.isna().any(axis=0)]:

    train[f'unknown_{col}'] = 0
    train.loc[train[col].isna(),f'unknown_{col}'] = 1
    train.loc[train[col].isna(), col] = 'Missing'

    test[f'unknown_{col}'] = 0
    test.loc[test[col].isna(),f'unknown_{col}'] = 1
    test.loc[test[col].isna(), col] = 'Missing'

train.head()
```

Out[532]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	isHoliday	...	unknown_temp_3_
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	0	...	
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	broken clouds	4516	0	...	
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	0	...	
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	0	...	
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	broken clouds	4918	0	...	

5 rows × 47 columns

In [533]:

```
X = train.drop(['traffic_volume'],axis=1).copy()
y = train['traffic_volume'].copy().astype(np.float32)
```

In [534]:

```
def rmse(y_true,y_pred):
    return np.sqrt(np.mean((y_true-y_pred)**2))

### Mean Absolute Error ###
def mae(y_true, y_pred):
    return np.mean(np.abs(y_true-y_pred))
```

In [535]:

```
X.shape
```

Out[535]:

```
(38563, 46)
```

In [536]:

```
test.shape
```

Out[536]:

```
(9641, 46)
```

In [537]:

```
X.drop(['weather_description'],axis=1,inplace=True)
```

In [538]:

```
test.drop(['weather_description'],axis=1,inplace=True)
```

In [539]:

```
print(X.shape)
print(test.shape)
```

```
(38563, 45)
(9641, 45)
```

In [540]:

```
X.head()
```

Out[540]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	isHoliday	yesterdayHoliday	tomorrowHoliday	...	unknown_temp_3_l
0	2012-10-02 09:00:00	Work Day	288.28	0.0	0.0	40	Clouds	0	-1.0	0.0	...	
1	2012-10-02 10:00:00	Work Day	289.36	0.0	0.0	75	Clouds	0	-1.0	0.0	...	
2	2012-10-02 11:00:00	Work Day	289.58	0.0	0.0	90	Clouds	0	-1.0	0.0	...	
3	2012-10-02 12:00:00	Work Day	290.13	0.0	0.0	90	Clouds	0	-1.0	0.0	...	
4	2012-10-02 13:00:00	Work Day	291.14	0.0	0.0	75	Clouds	0	-1.0	0.0	...	

5 rows × 45 columns

In [541]:

```
X.drop(['holiday'],axis=1,inplace=True)
```

In [542]:

```
test.drop(['holiday'],axis=1,inplace=True)
```

In [543]:

```
print(X.shape)
print(test.shape)
```

```
(38563, 44)
(9641, 44)
```

In [544]:

```
X.head()
```

Out[544]:

	date_time	temp	rain_1h	snow_1h	clouds_all	weather_main	isHoliday	yesterdayHoliday	tomorrowHoliday	temp_1_days_before	...	unkno
0	2012-10-02 09:00:00	288.28	0.0	0.0	40	Clouds	0	-1.0	0.0	-1.0	...	
1	2012-10-02 10:00:00	289.36	0.0	0.0	75	Clouds	0	-1.0	0.0	-1.0	...	
2	2012-10-02 11:00:00	289.58	0.0	0.0	90	Clouds	0	-1.0	0.0	-1.0	...	
3	2012-10-02 12:00:00	290.13	0.0	0.0	90	Clouds	0	-1.0	0.0	-1.0	...	
4	2012-10-02 13:00:00	291.14	0.0	0.0	75	Clouds	0	-1.0	0.0	-1.0	...	

5 rows × 44 columns

In [545]:

```
X['weather_main'].unique()
```

Out[545]:

```
array(['Clouds', 'Clear', 'Rain', 'Drizzle', 'Mist', 'Haze', 'Fog',
       'Thunderstorm', 'Snow', 'Squall', 'Smoke'], dtype=object)
```

In [546]:

```
pd.get_dummies(X['weather_main'], prefix='weather',dummy_na=True)
```

Out[546]:

	weather_Clear	weather_Clouds	weather_Drizzle	weather_Fog	weather_Haze	weather_Mist	weather_Rain	weather_Smoke	weather_Sno
0	0	1	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	
2	0	1	0	0	0	0	0	0	
3	0	1	0	0	0	0	0	0	
4	0	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
38558	0	0	0	0	0	0	0	0	
38559	0	0	1	0	0	0	0	0	
38560	0	0	0	0	0	1	0	0	
38561	0	0	0	0	0	0	1	0	
38562	0	0	1	0	0	0	0	0	

38563 rows × 12 columns

In [547]:

```
X_df = pd.concat([X,pd.get_dummies(X['weather_main'], prefix='weather',dummy_na=True)],axis=1).drop(['wea
```

In [ ]:

In [548]:

```
pd.get_dummies(test['weather_main'], prefix='weather',dummy_na=True)
```

Out[548]:

	weather_Clear	weather_Clouds	weather_Drizzle	weather_Fog	weather_Haze	weather_Mist	weather_Rain	weather_Smoke	weather_Snov
0	0	0	0	0	0	1	0	0	
1	0	0	0	0	0	0	1	0	
2	0	0	0	0	0	1	0	0	
3	0	0	1	0	0	0	0	0	
4	0	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	.
9636	0	1	0	0	0	0	0	0	
9637	0	1	0	0	0	0	0	0	
9638	0	0	0	0	0	0	0	0	
9639	0	1	0	0	0	0	0	0	
9640	0	1	0	0	0	0	0	0	

9641 rows × 11 columns

In [549]:

```
test_df = pd.concat([test,pd.get_dummies(test['weather_main'], prefix='weather',dummy_na=True)],axis=1).d
```

In [550]:

```
print(X_df.shape)
print(test_df.shape)
```

```
(38563, 55)
(9641, 54)
```

In [551]:

```
X_df.drop(['weather_Squall'],axis=1,inplace=True)
```

In [552]:

```
X_df.head()
```

Out[552]:

	date_time	temp	rain_1h	snow_1h	clouds_all	isHoliday	yesterdayHoliday	tomorrowHoliday	temp_1_days_before	temp_1_hours_before	..
0	2012-10-02 09:00:00	288.28	0.0	0.0	40	0	-1.0	0.0	-1.0	-1.00	..
1	2012-10-02 10:00:00	289.36	0.0	0.0	75	0	-1.0	0.0	-1.0	288.28	..
2	2012-10-02 11:00:00	289.58	0.0	0.0	90	0	-1.0	0.0	-1.0	289.36	..
3	2012-10-02 12:00:00	290.13	0.0	0.0	90	0	-1.0	0.0	-1.0	289.58	..
4	2012-10-02 13:00:00	291.14	0.0	0.0	75	0	-1.0	0.0	-1.0	290.13	..

5 rows × 54 columns

In [553]:

```
test.drop(['date_time'],axis=1,inplace=True)
X.drop(['date_time'],axis=1,inplace=True)
```

In [ ]:

In [554]:

```
X_df.shape
```

Out[554]:

```
(38563, 54)
```

In [555]:

```
pd.get_dummies(X_df['day_time'], prefix='day',dummy_na=True)
```

Out[555]:

	day_Afternoon	day_Early Morning	day_Evening	day_Late Night	day_Morning	day_Night	day_nan
0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0
2	0	0	0	0	1	0	0
3	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...
38558	0	0	1	0	0	0	0
38559	0	0	0	0	0	1	0
38560	0	0	0	0	0	1	0
38561	0	0	0	0	0	1	0
38562	0	0	0	0	0	1	0

38563 rows × 7 columns

In [556]:

```
X_df = pd.concat([X_df,pd.get_dummies(X_df['day_time'], prefix='day',dummy_na=True)],axis=1).drop(['day_t
```

In [557]:

```
X_df.shape
```

Out[557]:

```
(38563, 60)
```

In [558]:

```
pd.get_dummies(test_df['day_time'], prefix='day',dummy_na=True)
```

Out[558]:

	day_Afternoon	day_Early Morning	day_Evening	day_Late Night	day_Morning	day_Night	day_nan
0	0	0	0	0	0	1	0
1	0	0	0	0	0	1	0
2	0	0	0	0	0	1	0
3	0	0	0	0	0	1	0
4	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...
9636	0	0	0	0	0	1	0
9637	0	0	0	0	0	1	0
9638	0	0	0	0	0	1	0
9639	0	0	0	0	0	1	0
9640	0	0	0	0	0	1	0

9641 rows × 7 columns

In [559]:
```
test_df = pd.concat([test_df,pd.get_dummies(test_df['day_time'], prefix='day',dummy_na=True)],axis=1).dro
```

In [560]:
```
X_df.shape
```

Out[560]:
```
(38563, 60)
```

In [561]:
```
test_df.shape
```

Out[561]:
```
(9641, 60)
```

In [562]:
```
print(X_df.shape)
print(test_df.shape)
```

```
(38563, 60)
(9641, 60)
```

In [563]:
```
X_df.drop(['date_time'],axis=1,inplace=True)
test_df.drop(['date_time'],axis=1,inplace=True)
```

In [564]:
```
X_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38563 entries, 0 to 38562
Data columns (total 59 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   temp                                38563 non-null  float64
 1   rain_1h                             38563 non-null  float64
 2   snow_1h                             38563 non-null  float64
 3   clouds_all                          38563 non-null  int64
 4   isHoliday                           38563 non-null  int64
 5   yesterdayHoliday                    38563 non-null  float64
 6   tomorrowHoliday                     38563 non-null  float64
 7   temp_1_days_before                  38563 non-null  float64
 8   temp_1_hours_before                 38563 non-null  float64
 9   temp_2_hours_before                 38563 non-null  float64
 10  temp_3_hours_before                 38563 non-null  float64
 11  rain_1h_1_hours_before              38563 non-null  float64
 12  rain_1h_2_hours_before              38563 non-null  float64
 13  rain_1h_3_hours_before              38563 non-null  float64
 14  snow_1h_1_hours_before              38563 non-null  float64
 15  snow_1h_2_hours_before              38563 non-null  float64
 16  snow_1h_3_hours_before              38563 non-null  float64
 17  clouds_all_1_hours_before           38563 non-null  float64
 18  clouds_all_2_hours_before           38563 non-null  float64
 19  clouds_all_3_hours_before           38563 non-null  float64
 20  day                                 38563 non-null  int64
 21  month                               38563 non-null  int64
 22  year                                38563 non-null  int64
 23  weekday                             38563 non-null  int64
 24  hour                                38563 non-null  int64
 25  is_weekend                          38563 non-null  int64
 26  unknown_yesterdayHoliday            38563 non-null  int64
 27  unknown_tomorrowHoliday             38563 non-null  int64
 28  unknown_temp_1_days_before          38563 non-null  int64
 29  unknown_temp_1_hours_before         38563 non-null  int64
 30  unknown_temp_2_hours_before         38563 non-null  int64
 31  unknown_temp_3_hours_before         38563 non-null  int64
 32  unknown_rain_1h_1_hours_before      38563 non-null  int64
 33  unknown_rain_1h_2_hours_before      38563 non-null  int64
 34  unknown_rain_1h_3_hours_before      38563 non-null  int64
 35  unknown_snow_1h_1_hours_before      38563 non-null  int64
 36  unknown_snow_1h_2_hours_before      38563 non-null  int64
 37  unknown_snow_1h_3_hours_before      38563 non-null  int64
 38  unknown_clouds_all_1_hours_before   38563 non-null  int64
 39  unknown_clouds_all_2_hours_before   38563 non-null  int64
 40  unknown_clouds_all_3_hours_before   38563 non-null  int64
 41  weather_Clear                       38563 non-null  uint8
 42  weather_Clouds                      38563 non-null  uint8
 43  weather_Drizzle                     38563 non-null  uint8
 44  weather_Fog                         38563 non-null  uint8
 45  weather_Haze                        38563 non-null  uint8
 46  weather_Mist                        38563 non-null  uint8
 47  weather_Rain                        38563 non-null  uint8
 48  weather_Smoke                       38563 non-null  uint8
 49  weather_Snow                        38563 non-null  uint8
 50  weather_Thunderstorm                38563 non-null  uint8
 51  weather_nan                         38563 non-null  uint8
 52  day_Afternoon                       38563 non-null  uint8
 53  day_Early Morning                   38563 non-null  uint8
 54  day_Evening                         38563 non-null  uint8
 55  day_Late Night                      38563 non-null  uint8
 56  day_Morning                         38563 non-null  uint8
 57  day_Night                           38563 non-null  uint8
 58  day_nan                             38563 non-null  uint8
dtypes: float64(18), int64(23), uint8(18)
memory usage: 12.7 MB
```

In [565]:

```
from sklearn.ensemble import RandomForestRegressor
RF=RandomForestRegressor(bootstrap= True, ccp_alpha= 0.0, criterion= 'mse', max_depth= None, max_features
```

In [566]:

```
RF.fit(X_df,y)
```

Out[566]:

```
RandomForestRegressor(max_features=0.8, max_samples=0.8, min_samples_leaf=5,
                      min_samples_split=10, n_jobs=-1, random_state=42)
```

In [567]:

```
pr=RF.predict(test_df)
```

In [568]:

```
pr[0]
```

Out[568]:

```
2587.8154193143455
```

In [ ]:

```
import xgboost as xgb
regressor=xgb.XGBRegressor()
booster=['gbtree','gblinear']
base_score=[0.25,0.5,0.75,1]
n_estimators = [100, 500, 900, 1100, 1500]
max_depth = [2, 3, 5, 10, 15]
booster=['gbtree','gblinear']
learning_rate=[0.05,0.1,0.15,0.20]
min_child_weight=[1,2,3,4]

# Define the grid of hyperparameters to search
hyperparameter_grid = {
    'n_estimators': n_estimators,
    'max_depth':max_depth,
    'learning_rate':learning_rate,
    'min_child_weight':min_child_weight,
    'booster':booster,
    'base_score':base_score
    }
from sklearn.model_selection import RandomizedSearchCV
random_cv = RandomizedSearchCV(estimator=regressor,
            param_distributions=hyperparameter_grid,
            cv=5, n_iter=50,
            scoring = 'neg_mean_absolute_error',n_jobs = 4,
            verbose = 5,
            return_train_score = True,
            random_state=42)
random_cv.fit(X_df,y)
```

In [ ]:

```
random_cv.best_estimator_
```

In [ ]:

```
xgreg=xgb.XGBRegressor(base_score=0.25, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=10,
             min_child_weight=3, missing=np.nan, monotone_constraints='()',
             n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

In [384]:

```
xgreg.fit(X_df,y)
```

Out[384]:

```
XGBRegressor(base_score=0.25, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=10,
             min_child_weight=3, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

In [385]:

```
y_pred=xgreg.predict(test_df)
```

In [388]:

```
y_pred[0]
```

Out[388]:

```
2566.1528
```

In [390]:

```
tr=pd.DataFrame({
    'Values':pr
})
tr.to_csv('RF_out.csv',index=True)
```

In [391]:

```
from tensorflow import keras
import tensorflow as tf
```

```
import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
def build_model(hp):
    model = keras.Sequential()
    for i in range(hp.Int('num_layers', 2, 20)):
        model.add(layers.Dense(units=hp.Int('units_' + str(i),
                                            min_value=32,
                                            max_value=512,
                                            step=32),
                               activation='relu'))
    model.add(layers.Dense(1, activation='linear'))
    model.compile(
        optimizer=keras.optimizers.Adam(
            hp.Choice('learning_rate', [1e-2, 1e-3, 1e-4])),
        loss='mean_absolute_error',
        metrics=['mean_absolute_error'])
    return model

import os
tuner = RandomSearch(
    build_model,
    objective='val_mean_absolute_error',
    max_trials=5,
    executions_per_trial=3,
    directory=os.path.normpath('C:/'),
    project_name='traffic volume2')
```

In [392]:

```
tuner.search_space_summary()
```

```
Search space summary
Default search space size: 4
num_layers (Int)
{'default': None, 'conditions': [], 'min_value': 2, 'max_value': 20, 'step': 1, 'sampling': None}
units_0 (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step': 32, 'sampling': None}
units_1 (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step': 32, 'sampling': None}
learning_rate (Choice)
{'default': 0.01, 'conditions': [], 'values': [0.01, 0.001, 0.0001], 'ordered': True}
```

In [393]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_df, y, test_size=0.1, random_state=0)
```

In [395]:

```
tuner.search(X_train, y_train,
             epochs=40,
             validation_data=(X_test, y_test))
```

```
Trial 6 Complete [00h 06m 33s]
val_mean_absolute_error: 311.48248291015625

Best val_mean_absolute_error So Far: 311.48248291015625
Total elapsed time: 00h 28m 25s
INFO:tensorflow:Oracle triggered exit
```

In [396]:

```
tuner.results_summary()
```

```
Results summary
Results in C:\traffic volume2
Showing 10 best trials
Objective(name='val_mean_absolute_error', direction='min')
Trial summary
Hyperparameters:
num_layers: 8
units_0: 416
units_1: 192
learning_rate: 0.001
units_2: 64
units_3: 384
units_4: 256
units_5: 224
units_6: 96
units_7: 448
units_8: 128
units_9: 320
```

```
units_10: 320
units_11: 224
units_12: 480
units_13: 320
Score: 311.48248291015625
Trial summary
Hyperparameters:
num_layers: 4
units_0: 128
units_1: 192
learning_rate: 0.001
units_2: 32
units_3: 32
Score: 345.4068603515625
Trial summary
Hyperparameters:
num_layers: 5
units_0: 224
units_1: 288
learning_rate: 0.0001
units_2: 384
units_3: 192
units_4: 96
units_5: 320
units_6: 96
units_7: 224
Score: 534.2568969726562
Trial summary
Hyperparameters:
num_layers: 8
units_0: 416
units_1: 192
learning_rate: 0.01
units_2: 352
units_3: 512
units_4: 32
units_5: 32
units_6: 32
units_7: 32
Score: 552.9955647786459
Trial summary
Hyperparameters:
num_layers: 14
units_0: 352
units_1: 160
learning_rate: 0.01
units_2: 192
units_3: 224
units_4: 192
units_5: 416
units_6: 32
units_7: 288
units_8: 32
units_9: 32
units_10: 32
units_11: 32
units_12: 32
units_13: 32
Score: 627.6962890625
```

In [397]:

```
from tensorflow.keras import backend as K
def root_mean_squared_error(y_true, y_pred):
        return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

In [400]:

```
import tensorflow
from tensorflow import keras
from tensorflow.keras.layers import Dense
# Initialising the ANN
classifier = tensorflow.keras.Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(units = 128,kernel_initializer='he_uniform',activation='relu',input_dim = 59))

# Adding the second hidden layer
classifier.add(Dense(units = 416, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 192, kernel_initializer = 'he_uniform',activation='relu'))
```

```
classifier.add(Dense(units = 64, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 384, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 256, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 416, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 224, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 96, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 448, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 128, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 320, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 224, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 480, kernel_initializer = 'he_uniform',activation='relu'))
classifier.add(Dense(units = 320, kernel_initializer = 'he_uniform',activation='relu'))

# Adding the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'he_uniform', activation = 'linear'))

from tensorflow.keras.optimizers import Adam
opt=Adam(0.001)
# Compiling the ANN
classifier.compile(optimizer =opt, loss = root_mean_squared_error)
```

In [401]:

```
model_history=classifier.fit(X_df, y,validation_split=0.20, batch_size = 10, epochs = 20)
```

```
Epoch 1/20
3085/3085 [==============================] - 15s 5ms/step - loss: 2056.4169 - val_loss: 1473.4447
Epoch 2/20
3085/3085 [==============================] - 15s 5ms/step - loss: 1505.6940 - val_loss: 1013.7050
Epoch 3/20
3085/3085 [==============================] - 15s 5ms/step - loss: 1095.3258 - val_loss: 1401.8423
Epoch 4/20
3085/3085 [==============================] - 15s 5ms/step - loss: 1024.0867 - val_loss: 806.0040
Epoch 5/20
3085/3085 [==============================] - 15s 5ms/step - loss: 969.5723 - val_loss: 828.7491
Epoch 6/20
3085/3085 [==============================] - 17s 5ms/step - loss: 951.2166 - val_loss: 948.1044
Epoch 7/20
3085/3085 [==============================] - 16s 5ms/step - loss: 931.4168 - val_loss: 897.5654
Epoch 8/20
3085/3085 [==============================] - 15s 5ms/step - loss: 913.1956 - val_loss: 783.6357
Epoch 9/20
3085/3085 [==============================] - 15s 5ms/step - loss: 891.6628 - val_loss: 856.4600
Epoch 10/20
3085/3085 [==============================] - 15s 5ms/step - loss: 870.5150 - val_loss: 591.5922
Epoch 11/20
3085/3085 [==============================] - 15s 5ms/step - loss: 813.2217 - val_loss: 869.7451
Epoch 12/20
3085/3085 [==============================] - 15s 5ms/step - loss: 812.6589 - val_loss: 839.4857
Epoch 13/20
3085/3085 [==============================] - 15s 5ms/step - loss: 862.9013 - val_loss: 980.1609
Epoch 14/20
3085/3085 [==============================] - 15s 5ms/step - loss: 798.8660 - val_loss: 714.6978
Epoch 15/20
3085/3085 [==============================] - 15s 5ms/step - loss: 724.7006 - val_loss: 553.1680
Epoch 16/20
3085/3085 [==============================] - 15s 5ms/step - loss: 707.6517 - val_loss: 595.8585
Epoch 17/20
3085/3085 [==============================] - 15s 5ms/step - loss: 662.9933 - val_loss: 435.8539
Epoch 18/20
3085/3085 [==============================] - 15s 5ms/step - loss: 679.6038 - val_loss: 432.1163
Epoch 19/20
3085/3085 [==============================] - 15s 5ms/step - loss: 699.9949 - val_loss: 410.0677
Epoch 20/20
3085/3085 [==============================] - 15s 5ms/step - loss: 658.9631 - val_loss: 743.5485
```

In [403]:

```
nn=classifier.predict(test_df)
```

In [424]:

```
nn[0]
```

Out[424]:

```
2319.352783203125
```

In [412]:

```
sub6 = pd.DataFrame({

        "Values": nn
    })
```

```
sub6.to_csv('nn_out.csv', index=True)
```

#

LSTM

In [428]:

```
import tensorflow
from tensorflow import keras
from tensorflow.keras.layers import LSTM
```

In [443]:

```
from xgboost import XGBRegressor
```

In [509]:

```
ttt=pd.read_csv('Train.csv')
```

In [510]:

```
pp=ttt['traffic_volume']
```

In [511]:

```
pp.shape
```

Out[511]:

```
(38563,)
```

In [570]:

```
from sklearn.datasets import load_diabetes
from sklearn.linear_model import RidgeCV
from sklearn.svm import LinearSVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import StackingRegressor
X,ty = load_diabetes(return_X_y=True)
estimators = [
     ('lr', RidgeCV()),
     ('svr', LinearSVR(random_state=42),
     ('xgb',XGBRegressor()))

 ]
reg = StackingRegressor(
     estimators=estimators,
     final_estimator=RandomForestRegressor(n_estimators=10,
                                           random_state=42)
 )
reg.fit(X_df,y)
```

Out[570]:

```
StackingRegressor(estimators=[('lr', RidgeCV(alphas=array([ 0.1,  1. , 10. ]))),
                              ('svr', LinearSVR(random_state=42),
                               ('xgb',
                                XGBRegressor(base_score=None, booster=None,
                                             colsample_bylevel=None,
                                             colsample_bynode=None,
                                             colsample_bytree=None, gamma=None,
                                             gpu_id=None,
                                             importance_type='gain',
                                             interaction_constraints=None,
                                             learning_rate=None,
                                             max_delta_step=None,
                                             max_depth=None,
                                             min_child_weight=None, missing=nan,
                                             monotone_constraints=None,
                                             n_estimators=100, n_jobs=None,
                                             num_parallel_tree=None,
                                             random_state=None, reg_alpha=None,
                                             reg_lambda=None,
                                             scale_pos_weight=None,
                                             subsample=None, tree_method=None,
                                             validate_parameters=None,
                                             verbosity=None)))],
                  final_estimator=RandomForestRegressor(n_estimators=10,
                                                        random_state=42))
```

In [572]:

```
dd=reg.predict(test_df)
```

In [576]:

```
w1=0.2
w2=0.7 # high for Xg boost as it is hyperparameter tuned and as it focuses on both bias and variance whe
w3=0.1 #low for neural network as the loss is high
final_pred=((w1*pr)+(w2*y_pred)+(w3*nn))
```

In [577]:

```
final_pred[0]
```

Out[577]:

```
2545.805369019119
```

In [578]:

```
sub6 = pd.DataFrame({

        "Values": nn
    })
sub6.to_csv('output.csv', index=True)
```

In [ ]: