Kashish Bhagat

In [1]:

```python
#Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
#Importing dataset from csv to data frame
df_traffic_data = pd.read_csv(r'C:\Users\welcome\Desktop\Final Year Projects\
```

In [3]:

```python
df_traffic_data.head()
```

Out[3]:

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

In [4]:

```python
df_traffic_data.shape
```

Out[4]:

```
(38563, 9)
```

In [5]:
```
df_traffic_data.dtypes
```

Out[5]:
```
date_time               object
holiday                 object
temp                   float64
rain_1h                float64
snow_1h                float64
clouds_all               int64
weather_main            object
weather_description     object
traffic_volume           int64
dtype: object
```

In [6]:
```
df_traffic_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
```

No null value is present in the data.

In [7]:
```
df_traffic_data.describe()
```

Out[7]:

	temp	rain_1h	snow_1h	clouds_all	traffic_volume
count	38563.000000	38563.000000	38563.000000	38563.000000	38563.000000
mean	281.351757	0.392733	0.000278	49.920364	3260.940409
std	13.216927	50.075055	0.009131	38.849106	1991.628329
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	272.858000	0.000000	0.000000	1.000000	1186.500000
50%	282.750000	0.000000	0.000000	64.000000	3378.000000
75%	291.540000	0.000000	0.000000	90.000000	4939.000000
max	308.240000	9831.300000	0.510000	100.000000	7280.000000

In [8]:

```
df_traffic_data.describe(include='object')
```

Out[8]:

	date_time	holiday	weather_main	weather_description
count	38563	38563	38563	38563
unique	32607	12	11	38
top	2013-05-19 10:00:00	None	Clouds	sky is clear
freq	6	38515	12680	8848

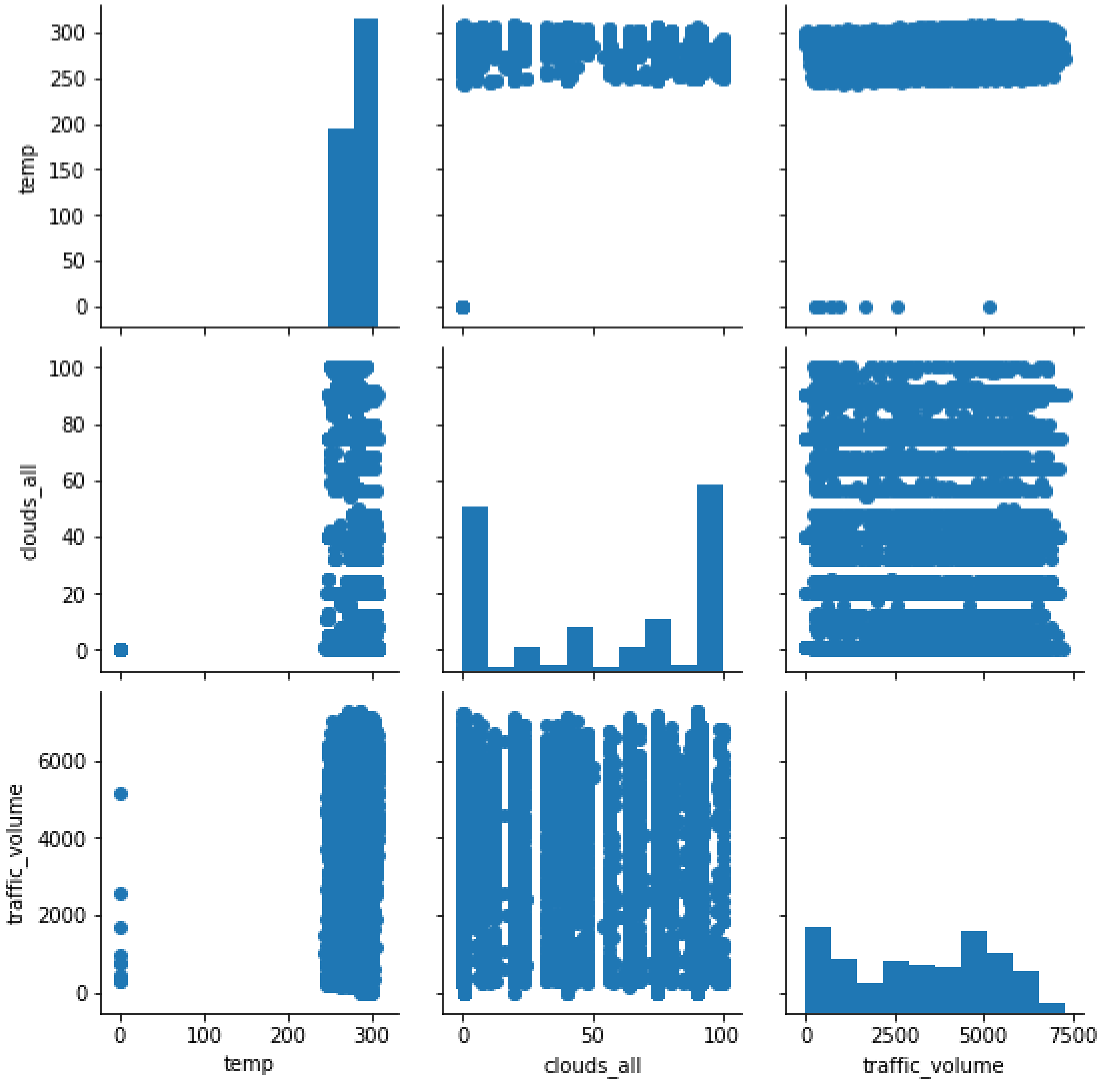
In [9]:

```
print("max date :" +df_traffic_data.date_time.max())
print("min date :" +df_traffic_data.date_time.min())
```

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

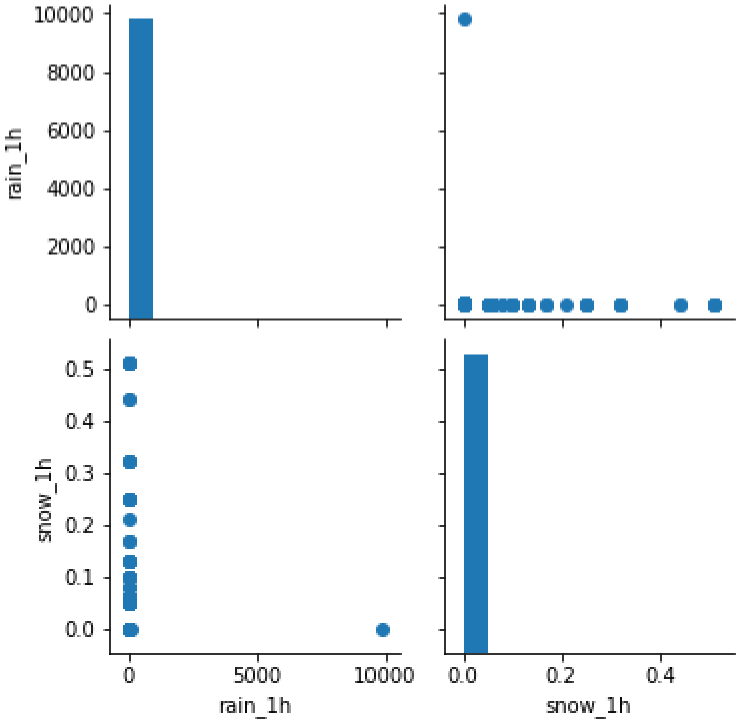
```
In [13]: g = sns.PairGrid(df[[ 'temp',  'clouds_all','traffic_volume']])
         g.map_diag(plt.hist)
         g.map_offdiag(plt.scatter)
```

```
Out[13]: <seaborn.axisgrid.PairGrid at 0x2aa68dc1f48>
```



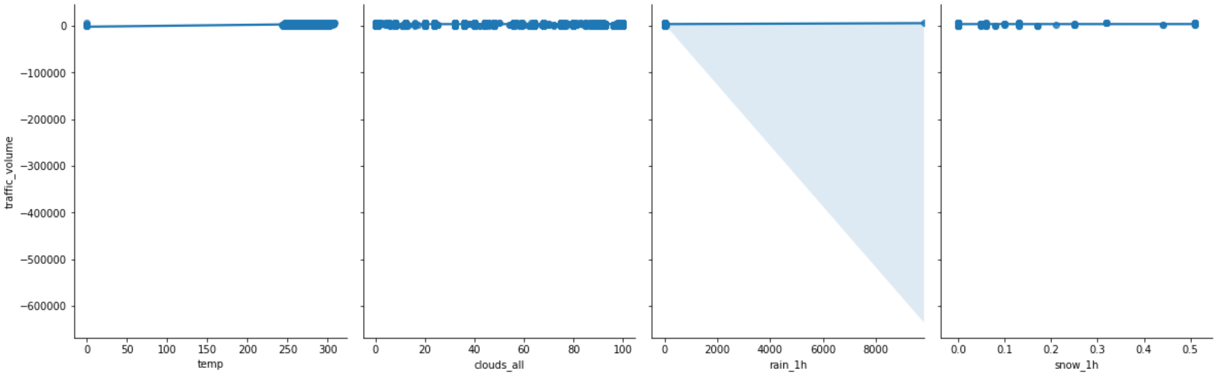
In [14]:
```
g = sns.PairGrid(df[[ 'rain_1h', 'snow_1h']])
g.map_diag(plt.hist)
g.map_offdiag(plt.scatter)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x2aa69ffc488>
```



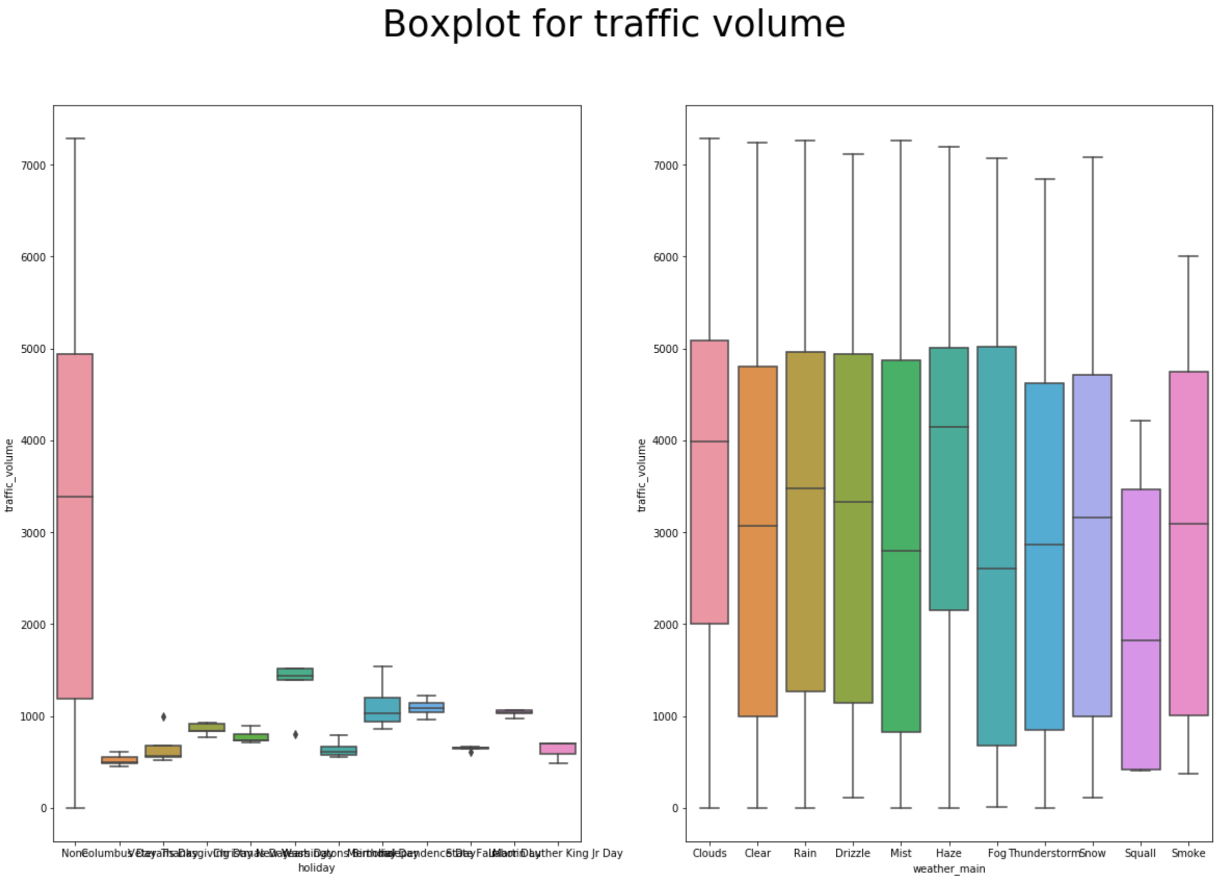
In [15]:

```
sns.pairplot(df, x_vars=['temp',  'clouds_all','rain_1h', 'snow_1h'], y_vars=
                 height=5, aspect=.8, kind="reg");
```



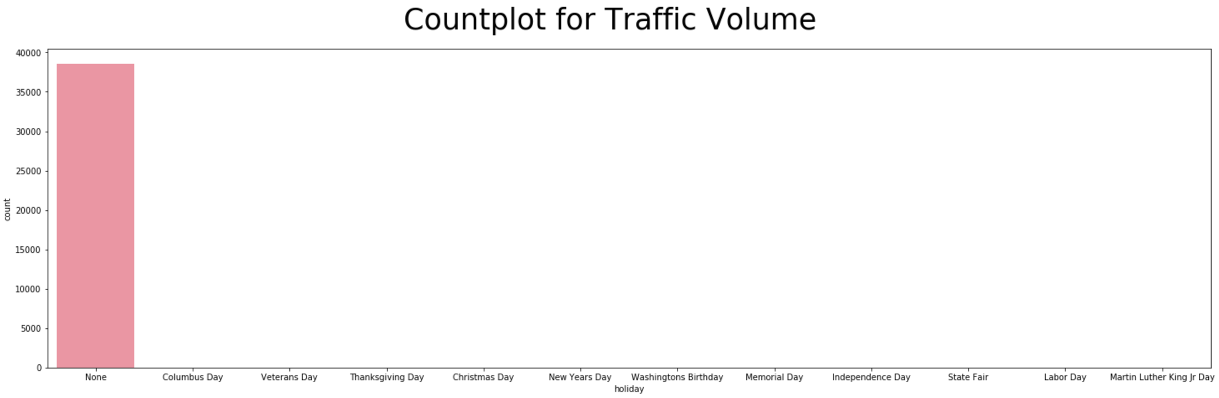
In [16]:
```
fig, (ax1, ax2) = plt.subplots(1,2,figsize=(20,13))

fig.suptitle("Boxplot for traffic volume", fontsize=35)

sns.boxplot(x="holiday", y="traffic_volume", data=df,ax=ax1)
sns.boxplot(x="weather_main", y="traffic_volume", data=df,ax=ax2)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aa6b69f648>
```



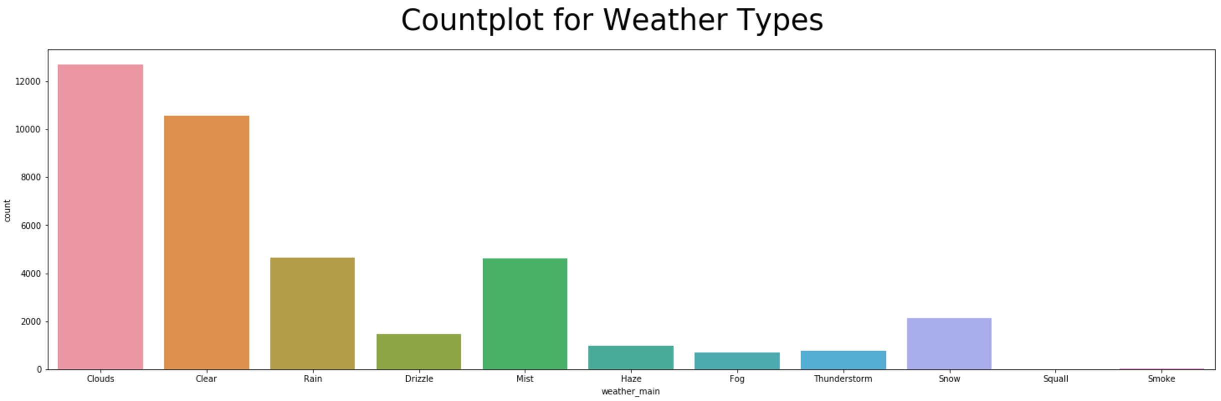
In [17]:
```
fig, (ax1) = plt.subplots(1,1,figsize=(25,7))

fig.suptitle("Countplot for Traffic Volume", fontsize=35)

sns.countplot(x="holiday", data=df,ax=ax1)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aa6c4df0c8>
```



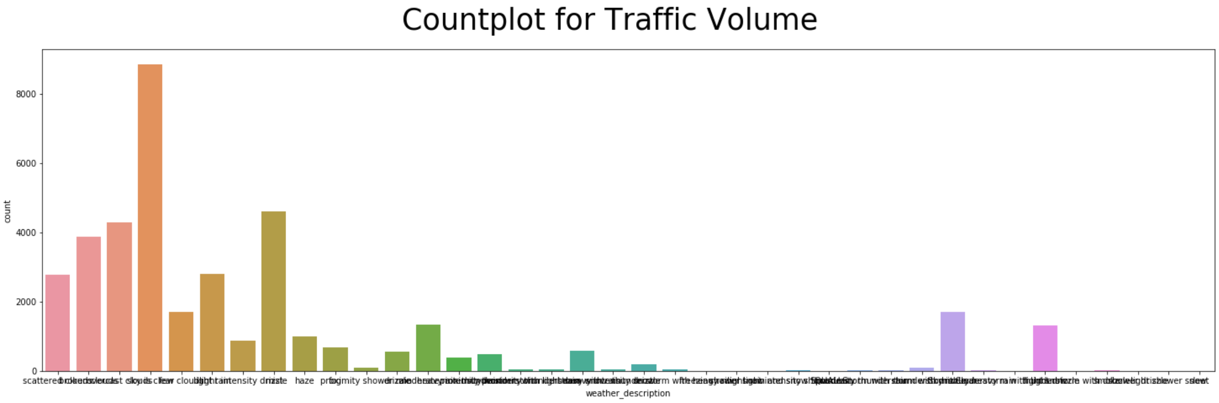
In [18]:
```
fig, (ax1) = plt.subplots(1,1,figsize=(25,7))

fig.suptitle("Countplot for Weather Types", fontsize=35)

sns.countplot(x="weather_main", data=df,ax=ax1)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aa6bfcaf48>
```



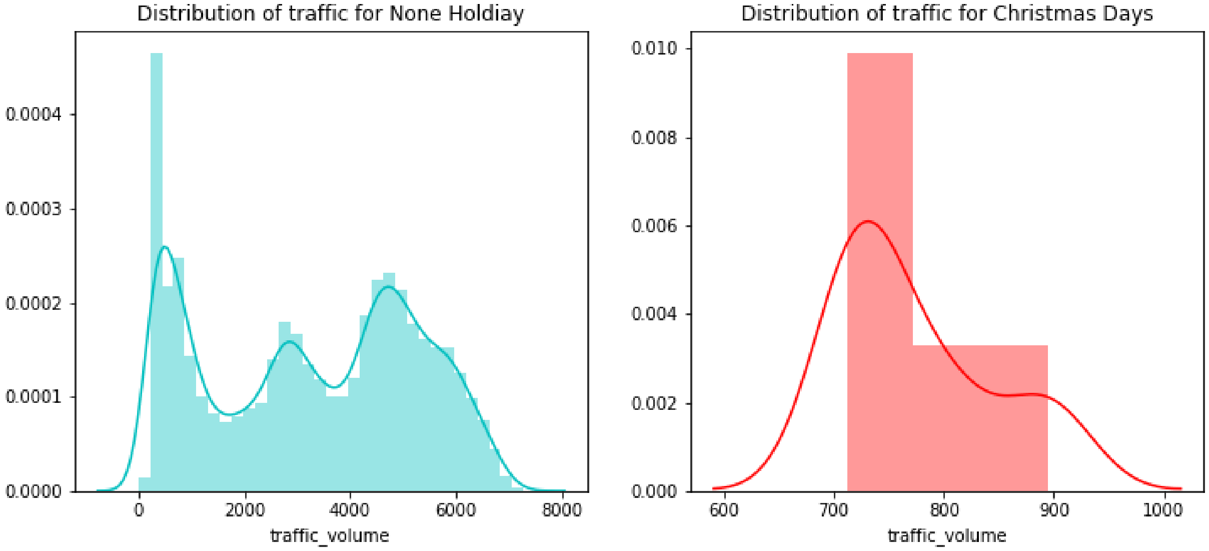
In [19]:
```
fig, (ax1) = plt.subplots(1,1,figsize=(25,7))

fig.suptitle("Countplot for Traffic Volume", fontsize=35)

sns.countplot(x="weather_description", data=df,ax=ax1)
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aa6c055588>
```



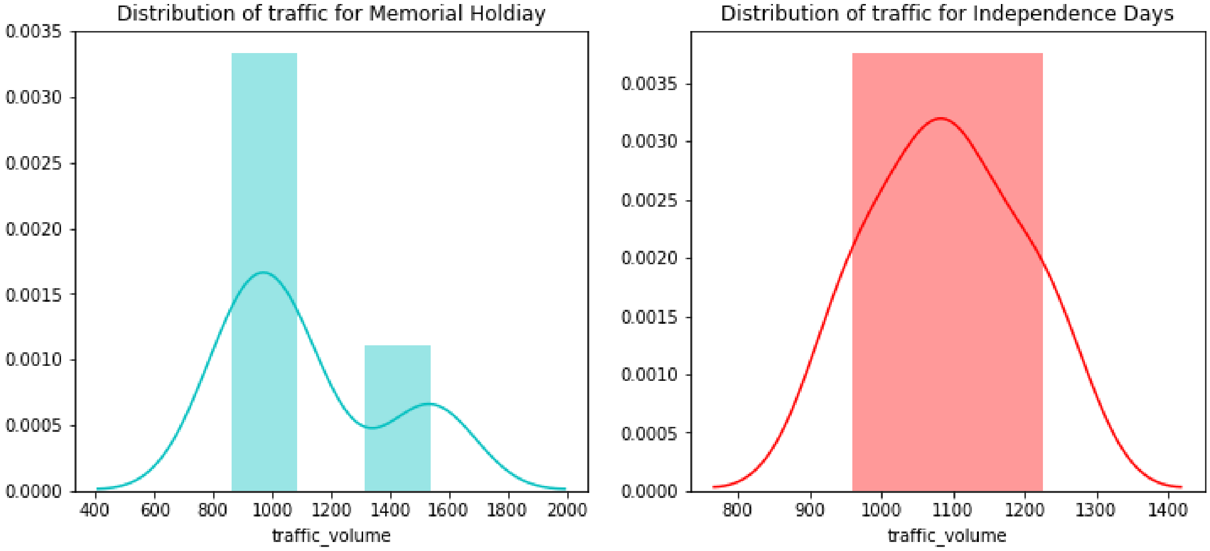
```
In [20]: f= plt.figure(figsize=(12,5))

         ax=f.add_subplot(121)
         sns.distplot(df[(df.holiday == "None")]["traffic_volume"],color='c',ax=ax)
         ax.set_title('Distribution of traffic for None Holdiay')



         ax=f.add_subplot(122)
         sns.distplot(df[(df.holiday == "Christmas Day")]['traffic_volume'],color='r',
         ax.set_title('Distribution of traffic for Christmas Days')
```

```
Out[20]: Text(0.5, 1.0, 'Distribution of traffic for Christmas Days')
```

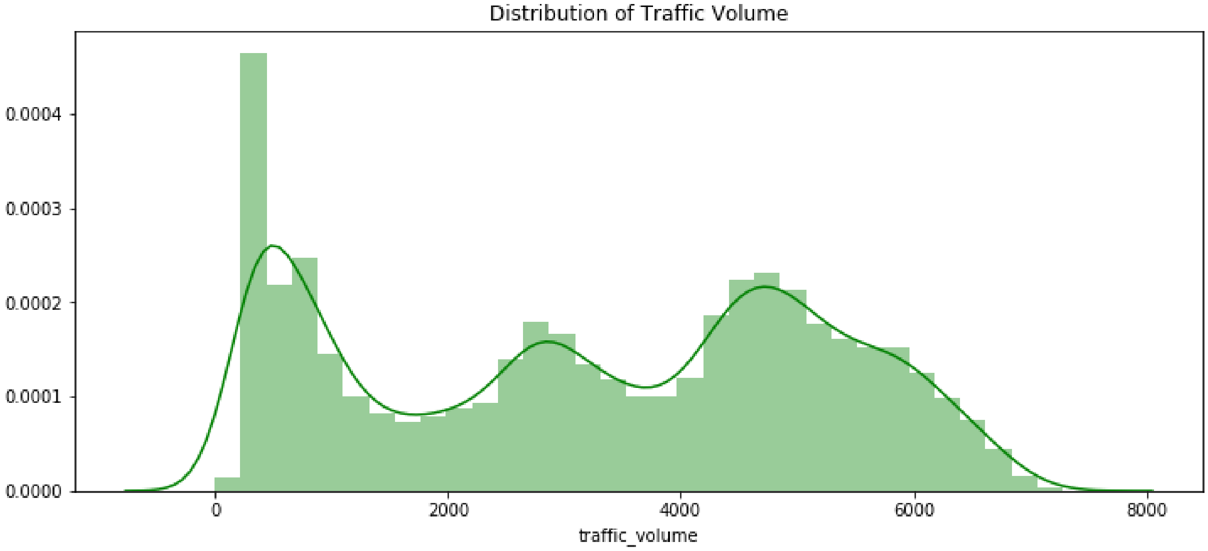


In [21]:
```
f= plt.figure(figsize=(12,5))

ax=f.add_subplot(121)
sns.distplot(df[(df.holiday == "Memorial Day")]["traffic_volume"],color='c',a
ax.set_title('Distribution of traffic for Memorial Holdiay')



ax=f.add_subplot(122)
sns.distplot(df[(df.holiday == "Independence Day")]['traffic_volume'],color='
ax.set_title('Distribution of traffic for Independence Days')
```

Out[21]:
```
Text(0.5, 1.0, 'Distribution of traffic for Independence Days')
```

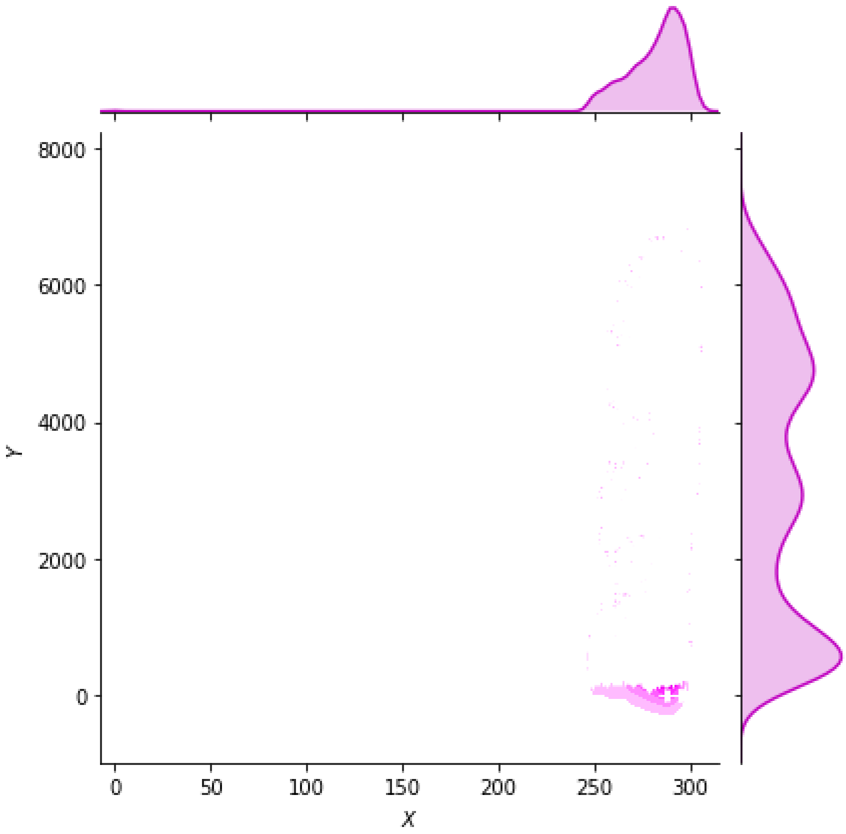


In [22]:
```
plt.figure(figsize=(12,5))
plt.title("Distribution of Traffic Volume")
ax = sns.distplot(df["traffic_volume"], color = 'g')
```



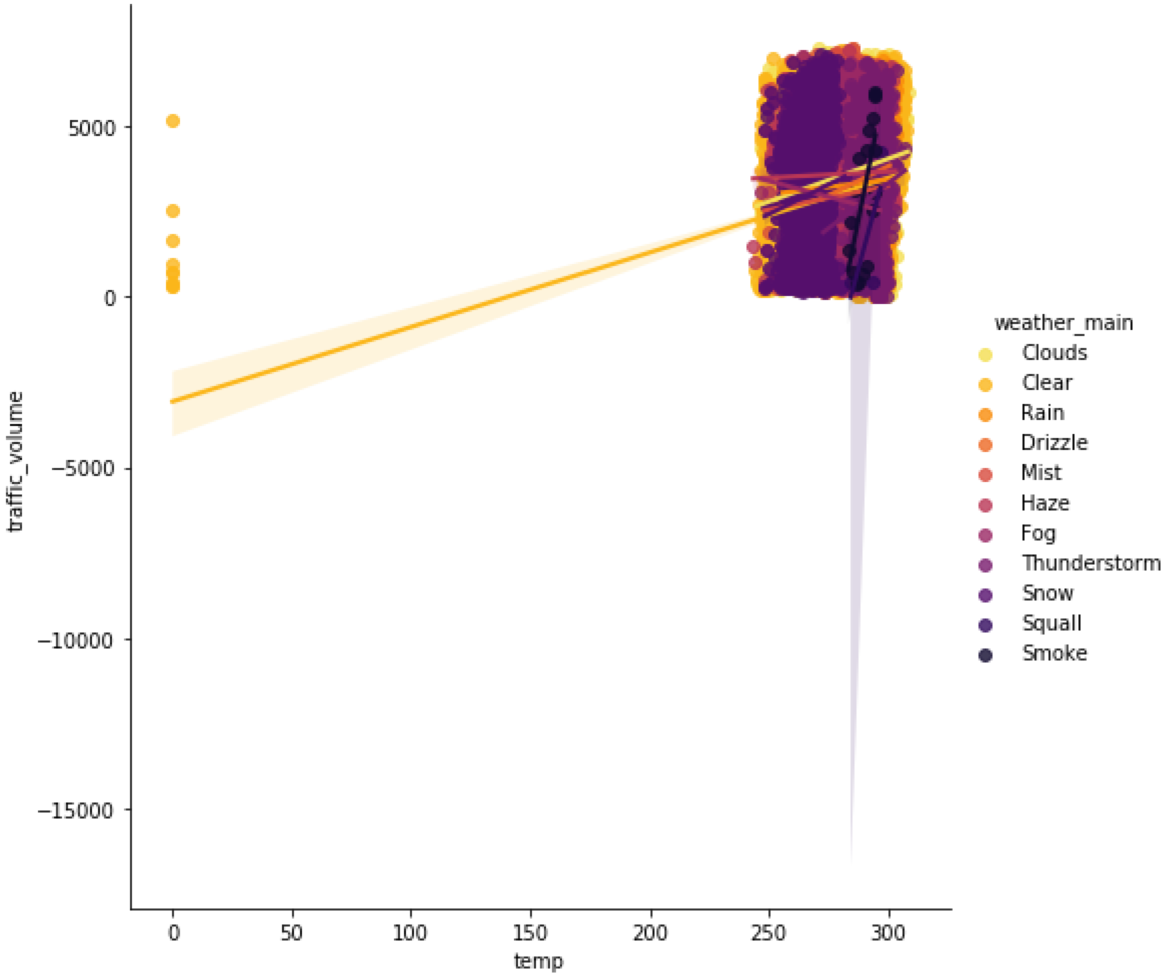
In [23]:
```
g = sns.jointplot(x="temp", y="traffic_volume", data = df[(df.weather_main ==
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("$X$", "$Y$")
ax.set_title('Distribution of charges and age for non-smokers')
```

Out[23]:
```
Text(0.5, 1, 'Distribution of charges and age for non-smokers')
```



In [24]:
```
sns.lmplot(x="temp", y="traffic_volume", hue="weather_main", data=df, palette
ax.set_title('Smokers and non-smokers')
```

Out[24]:
```
Text(0.5, 1, 'Smokers and non-smokers')
```

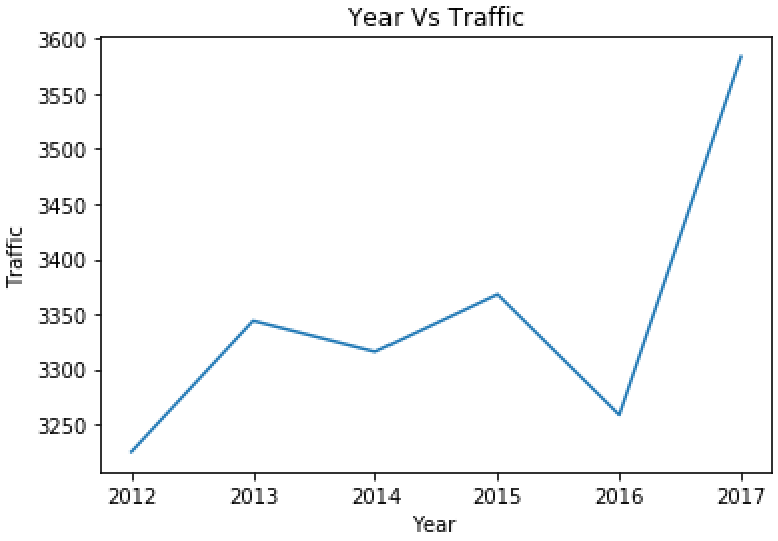


```
In [24]: #A very Clustered Data .....
```

```
In [25]: df['month']=pd.to_datetime(df['date_time']).dt.month
         df['day']=pd.to_datetime(df['date_time']).dt.day
         df['date']=pd.to_datetime(df['date_time']).dt.date
         df['year']=pd.to_datetime(df['date_time']).dt.year
         df['hour']=pd.to_datetime(df['date_time']).dt.hour
```

```
In [28]: df.groupby('year')['traffic_volume'].median().plot()
         plt.xlabel('Year ')
         plt.ylabel('Traffic')
         plt.title("Year Vs Traffic")
```

```
Out[28]: Text(0.5, 1.0, 'Year Vs Traffic')
```

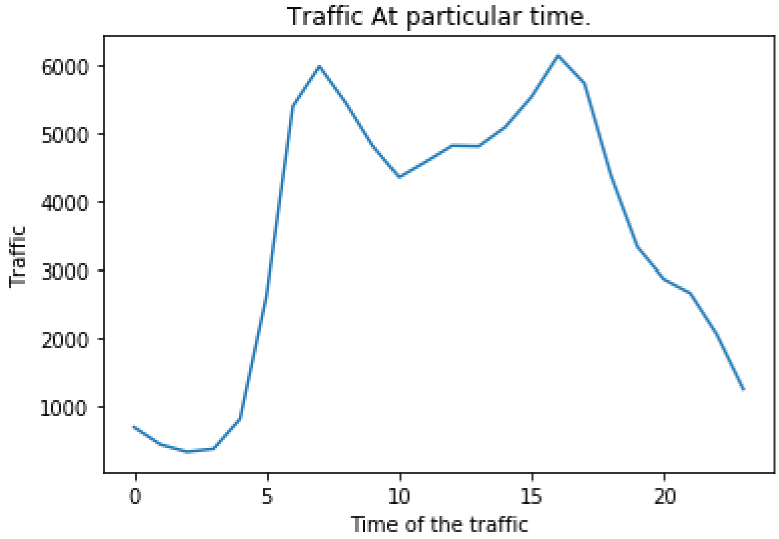


In [30]:

```
df.groupby('hour')['traffic_volume'].median().plot()
plt.xlabel('Time of the traffic')
plt.ylabel('Traffic')
plt.title("Traffic At particular time.")
```

Out[30]:

```
Text(0.5, 1.0, 'Traffic At particular time.')
```

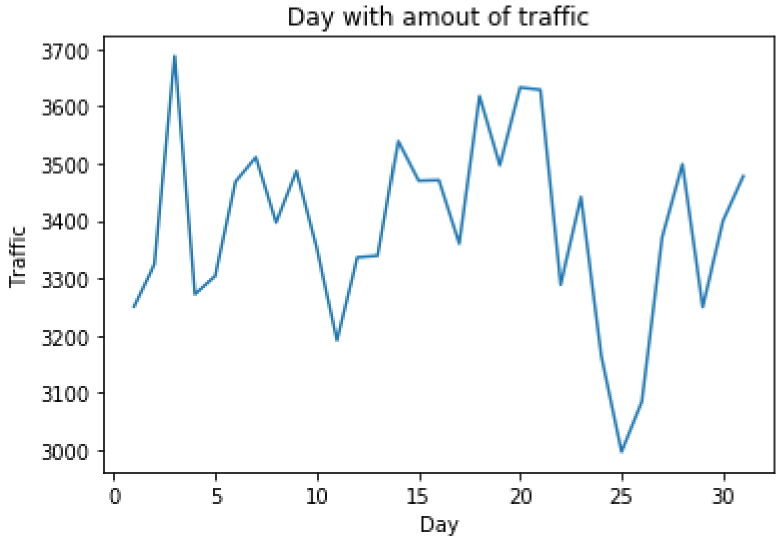


In [32]:

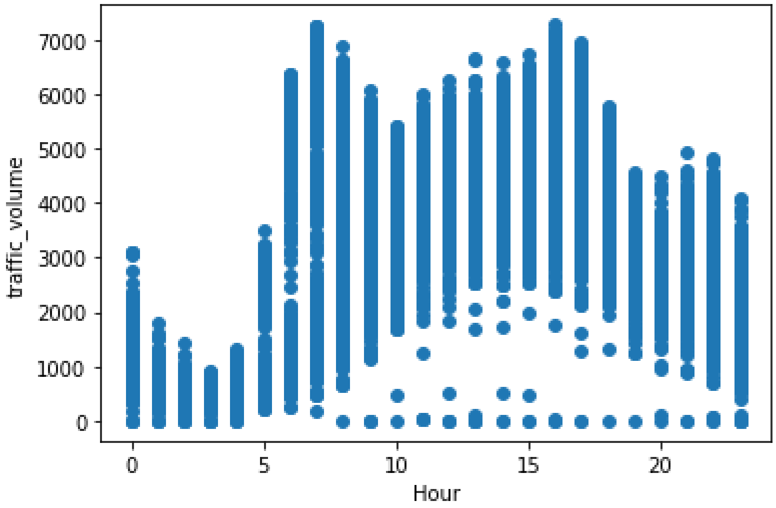
```
df.groupby('day')['traffic_volume'].median().plot()
plt.xlabel('Day')
plt.ylabel('Traffic')
plt.title("Day with amout of traffic")
```

Out[32]:

```
Text(0.5, 1.0, 'Day with amout of traffic')
```

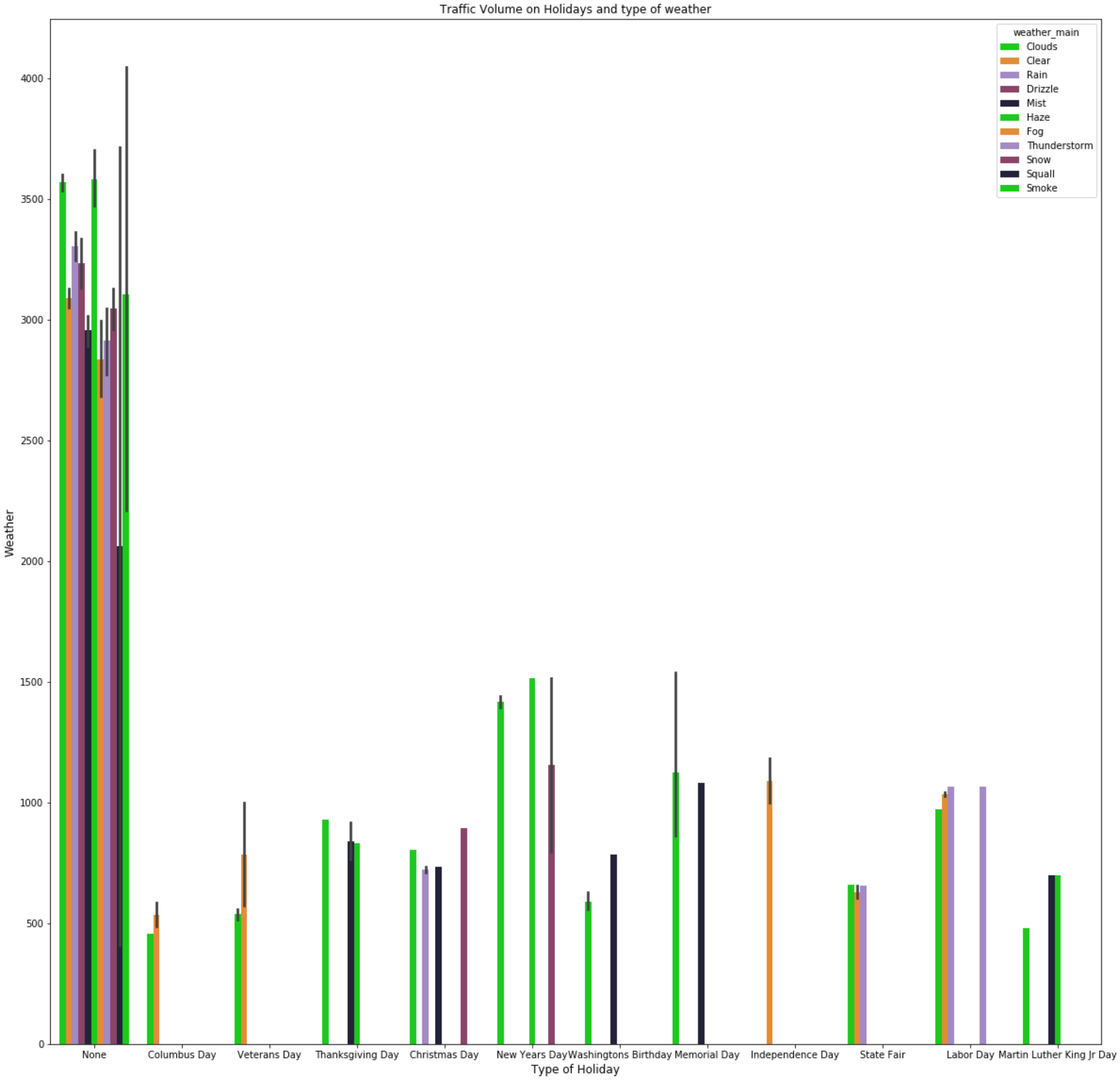


In [36]:
```
plt.scatter(df["hour"],df['traffic_volume'])
plt.xlabel("Hour")
plt.ylabel('traffic_volume')
plt.show()
```



In [46]:

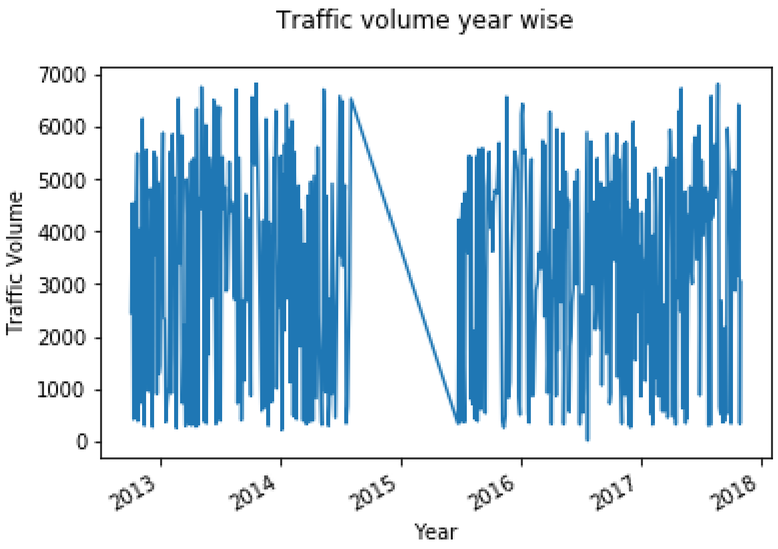
```
fig, ax = plt.subplots(figsize=(20,20))
colors = ["#00e600", "#ff8c1a","#a180cc","#963867","#1e1f3d"]
sns.barplot(x="holiday", y="traffic_volume",hue="weather_main", palette=color
ax.set_title("Traffic Volume on Holidays and type of weather",fontdict= {'siz
ax.xaxis.set_label_text("Type of Holiday",fontdict= {'size':12})
ax.yaxis.set_label_text("Weather",fontdict= {'size':12})
plt.show()
```



In [48]:
```
def file_len(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1

len_of_file = file_len(r'C:\Users\welcome\Desktop\Final Year Projects\Machine
print (len_of_file)

skipped = np.setdiff1d(np.arange(len_of_file), np.arange(0,len_of_file,80))
print (skipped)
```

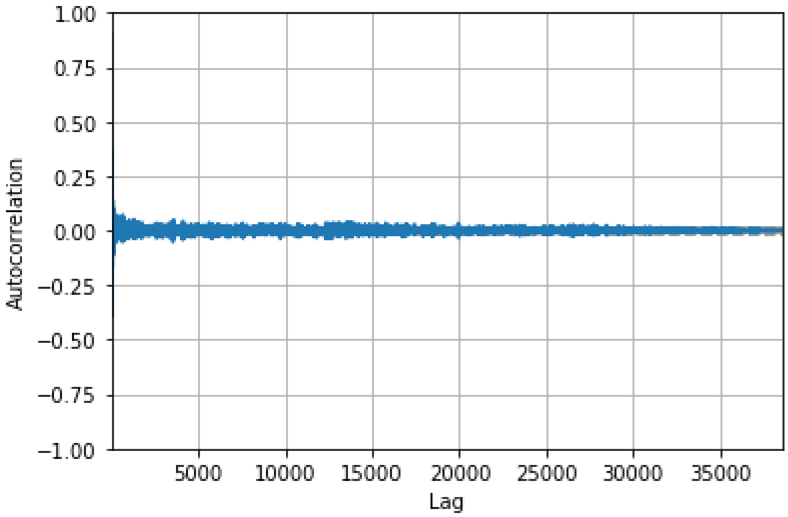
```
38564
[    1     2     3 ... 38561 38562 38563]
```

In [49]:
```
series = pd.read_csv(r'C:\Users\welcome\Desktop\Final Year Projects\Machine L
series.plot()
plt.suptitle('Traffic volume year wise')
plt.xlabel('Year')
plt.ylabel('Traffic Volume')
plt.show()
```



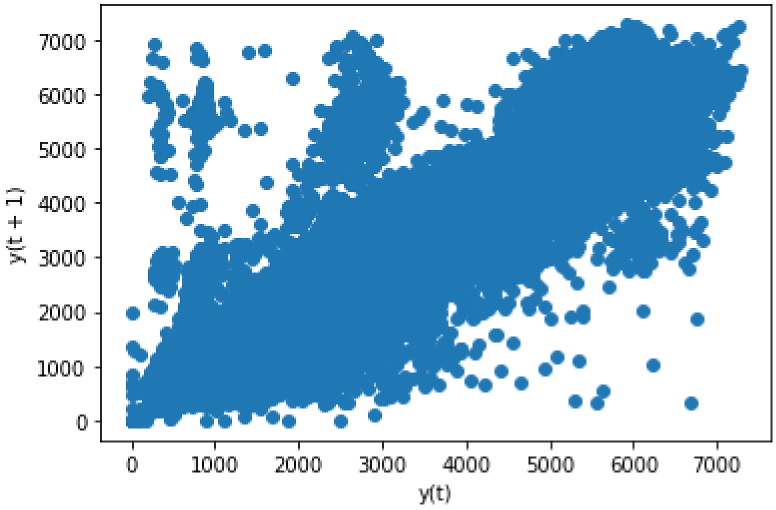
In [50]:

```
from pandas.plotting import autocorrelation_plot
import matplotlib.pyplot as plt
series = pd.read_csv(r'C:\Users\welcome\Desktop\Final Year Projects\Machine L
autocorrelation_plot(series)
plt.show()
```



In [51]:

```
from pandas.plotting import lag_plot
series = pd.read_csv(r'C:\Users\welcome\Desktop\Final Year Projects\Machine L
lag_plot(series)
plt.show()
```



In [52]:

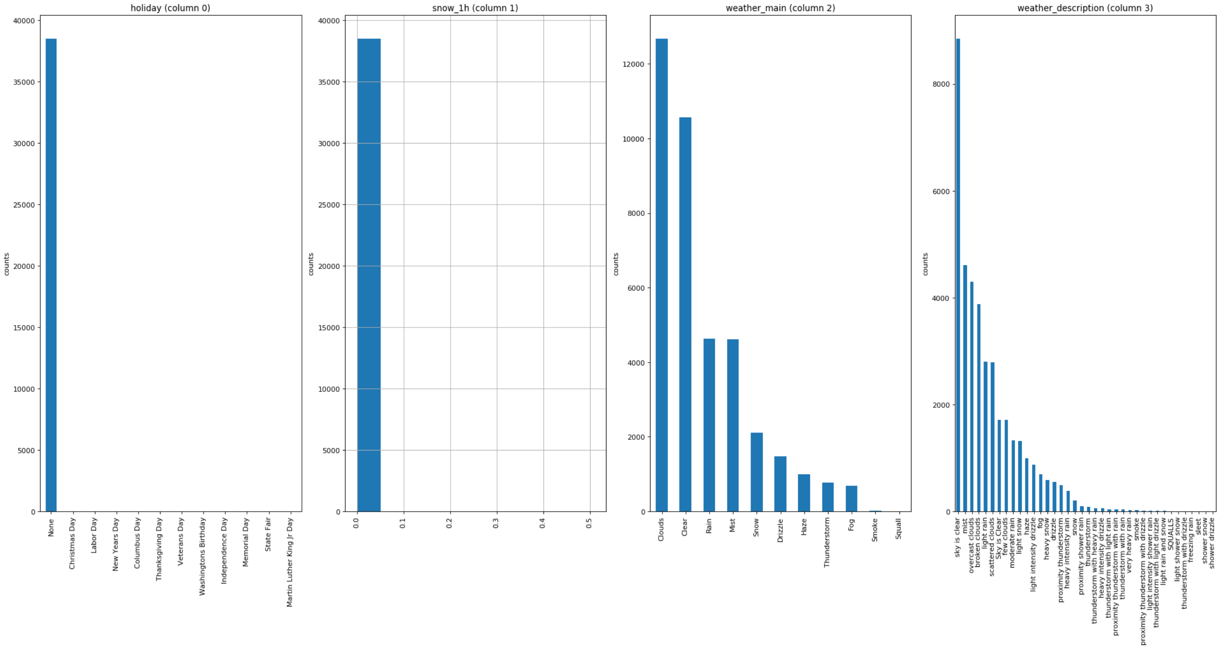
```
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] #
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi =
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

In [58]:

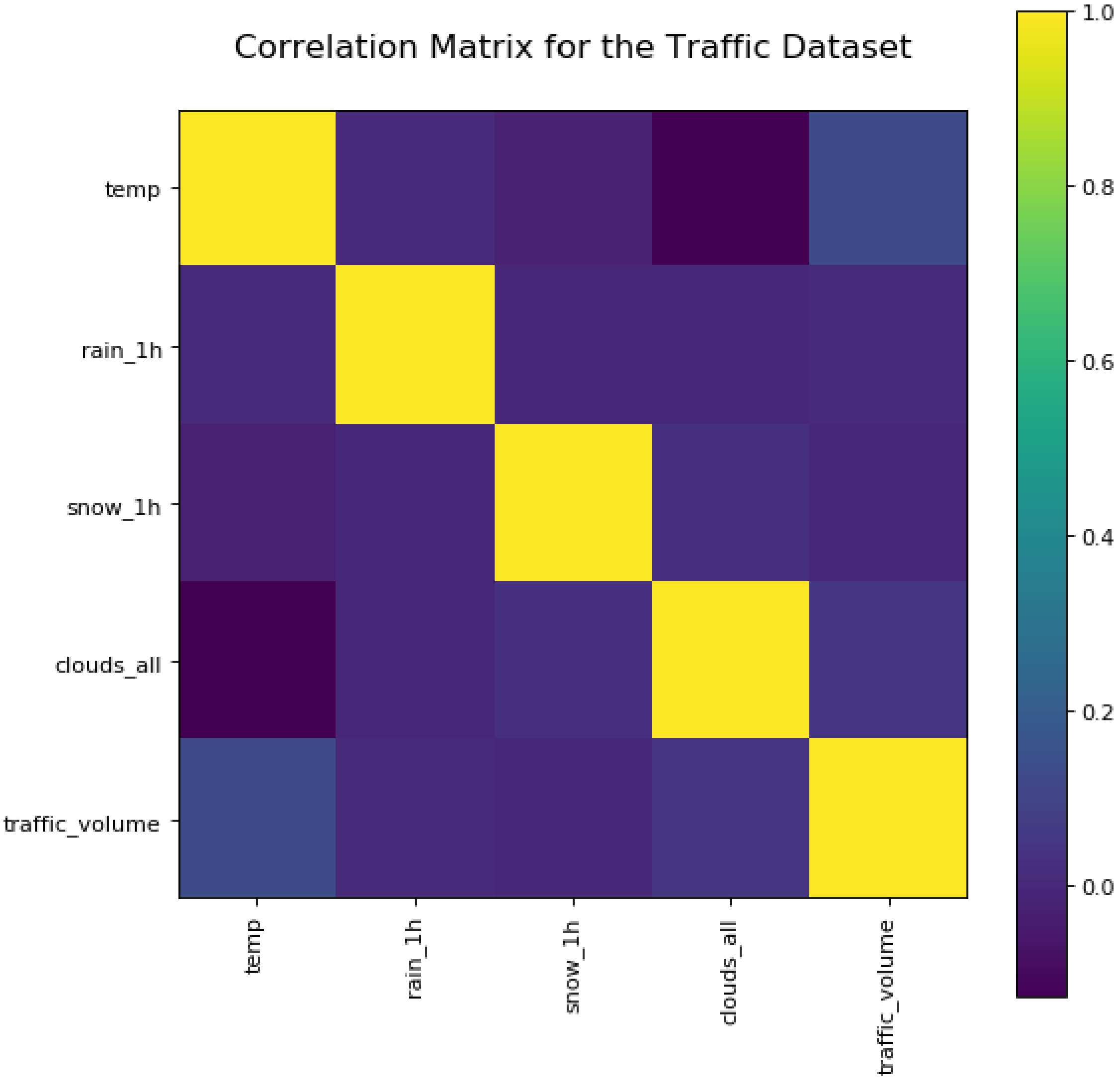
```
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
#     filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns wher
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor=
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for the Traffic Dataset', fontsize=15)
    plt.show()
```

In [54]:

```
plotPerColumnDistribution(df_traffic_data, 10, 5)
```



In [59]:
```
plotCorrelationMatrix(df_traffic_data, 8)
```



In [ ]:

data is collected over 6 years

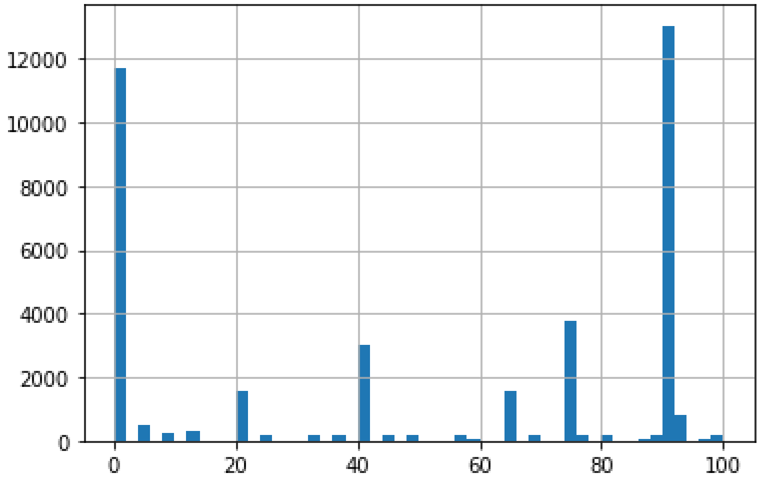
In [12]:
```
df=df_traffic_data.copy()
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['date_time', 'holiday', 'temp', 'rain_1h', 'snow_1h', 'clouds_all',
       'weather_main', 'weather_description', 'traffic_volume'],
      dtype='object')
```

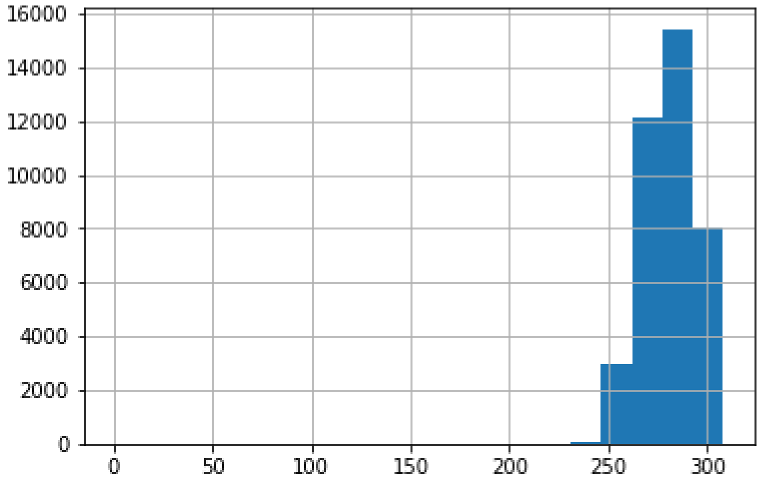
In [13]:
```
df.clouds_all.hist(bins=50)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1f48969c8>
```



In [15]:
```
df.temp.hist(bins=20)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1f5530708>
```

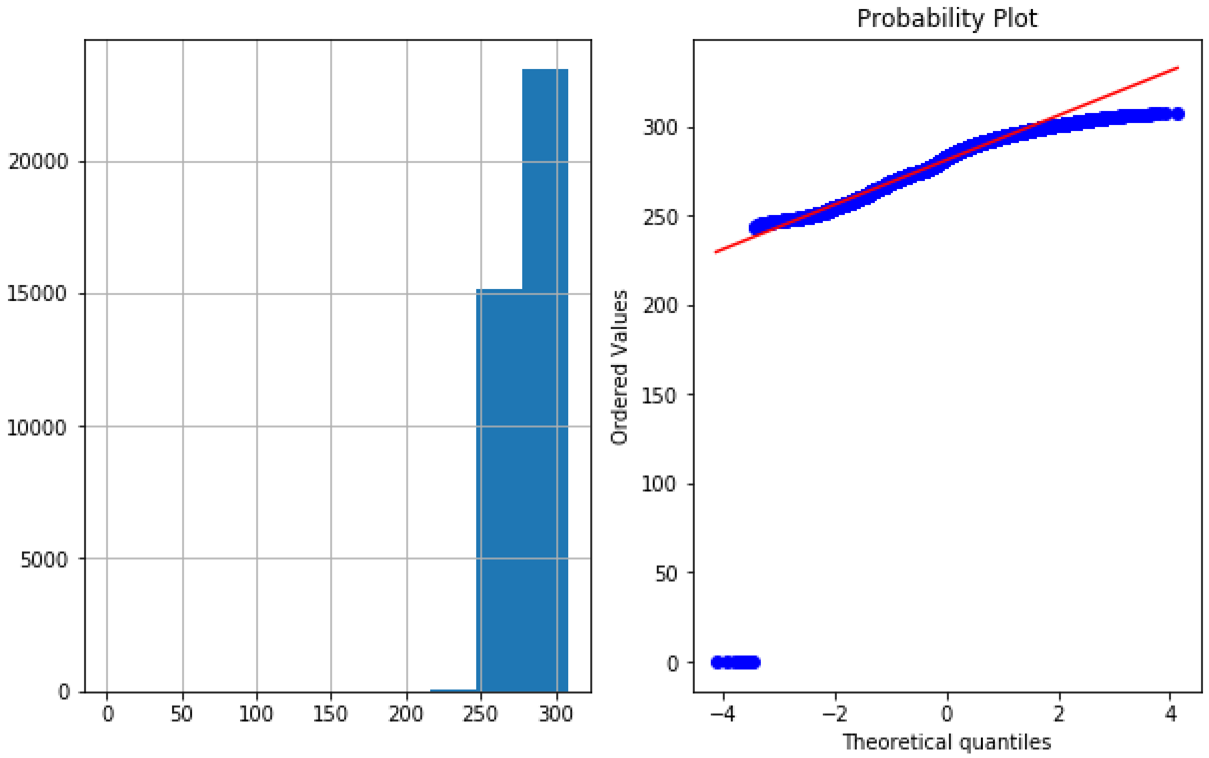


In [16]:
```
#Transforming The Data Into Good form...
```

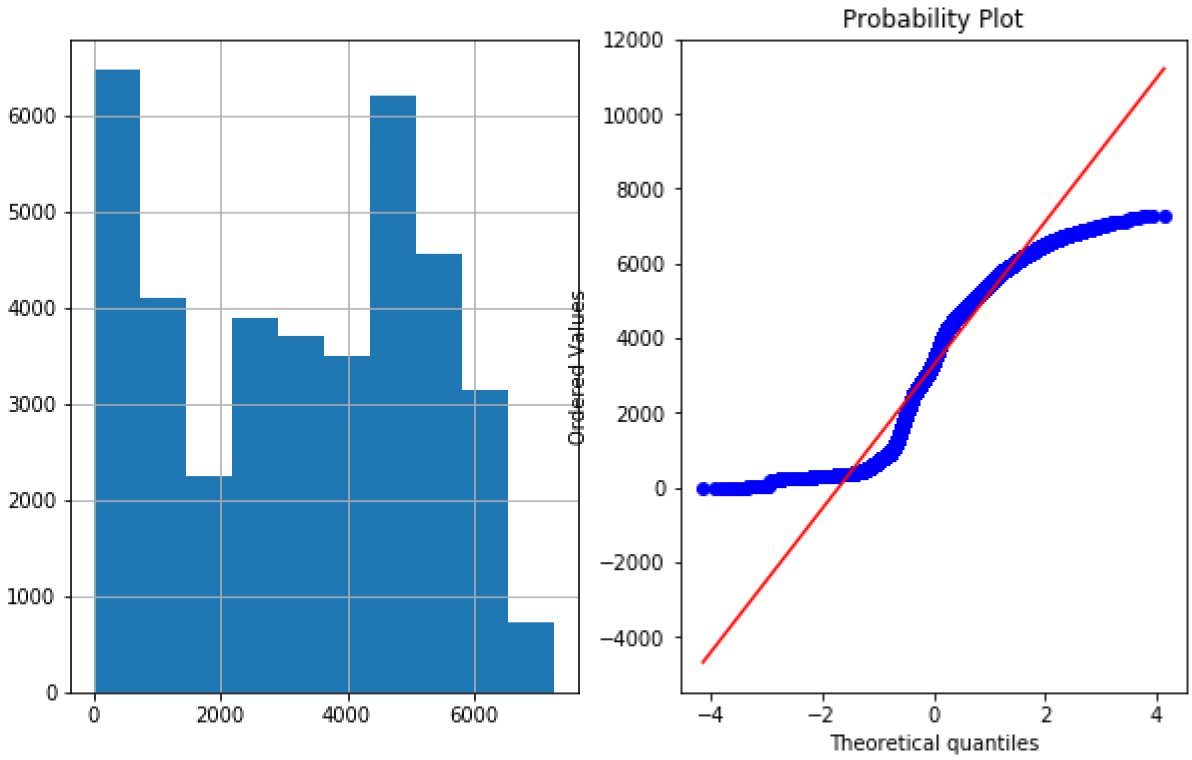
In [17]:
```
import scipy.stats as stat
from matplotlib import pylab
from pylab import *
```

In [18]:
```
def plot_data(df,feature):
    plt.figure(figsize=(10,6))
    plt.subplot(1,2,1) #1 row 2 columns
    df[feature].hist()
    plt.subplot(1,2,2) #1st row 2nd column 2nd index
    stat.probplot(df[feature],dist='norm',plot=pylab)
    plt.show()
```

In [19]:
```
plot_data(df,'temp')
```

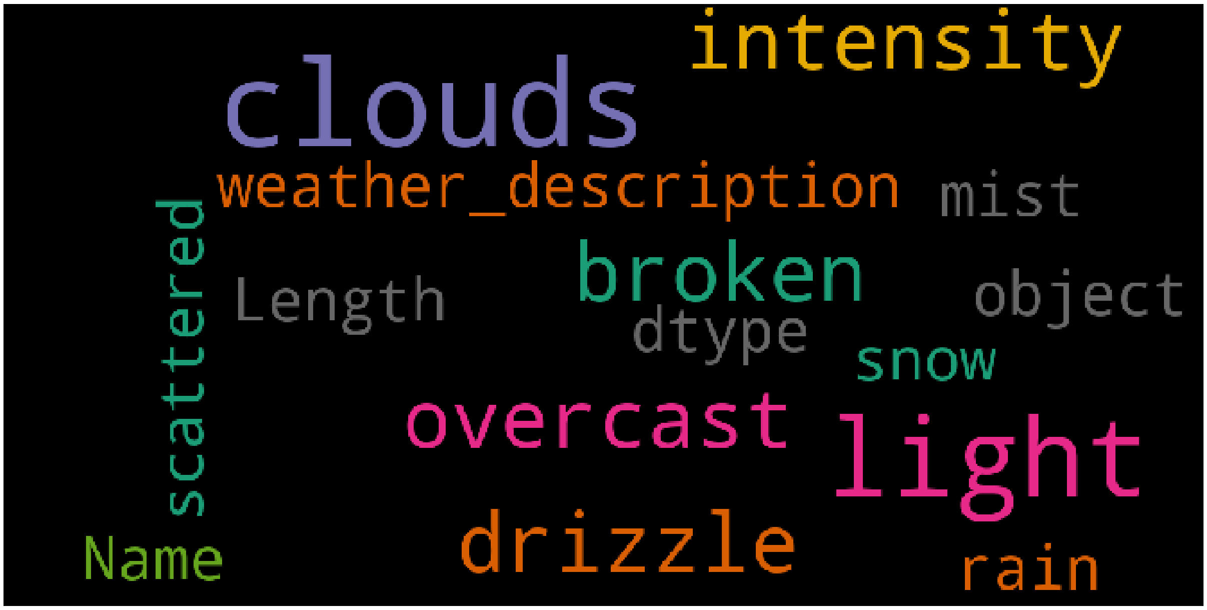


```
In [21]: plot_data(df,'traffic_volume')
```

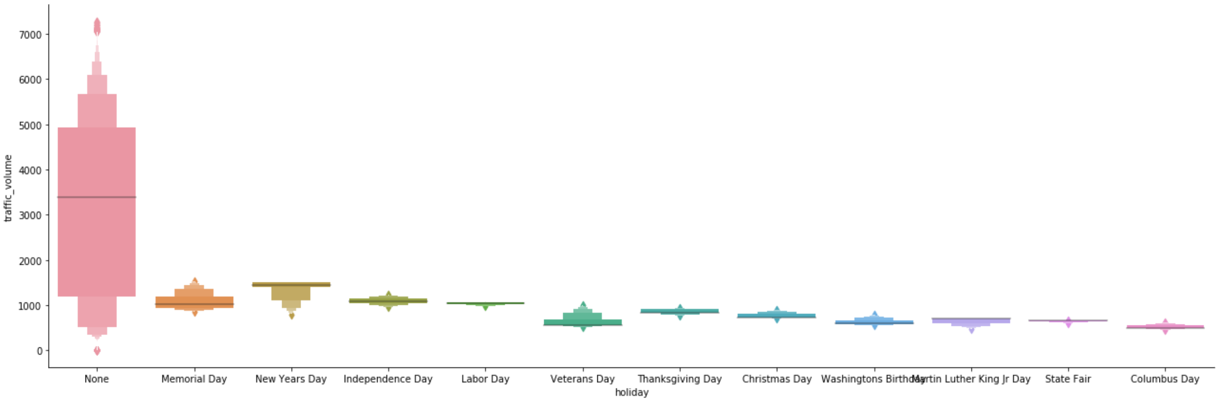


In [25]:

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        colormap='Dark2',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.imshow(wordcloud,interpolation='bilinear')
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(df['weather_description'])
```

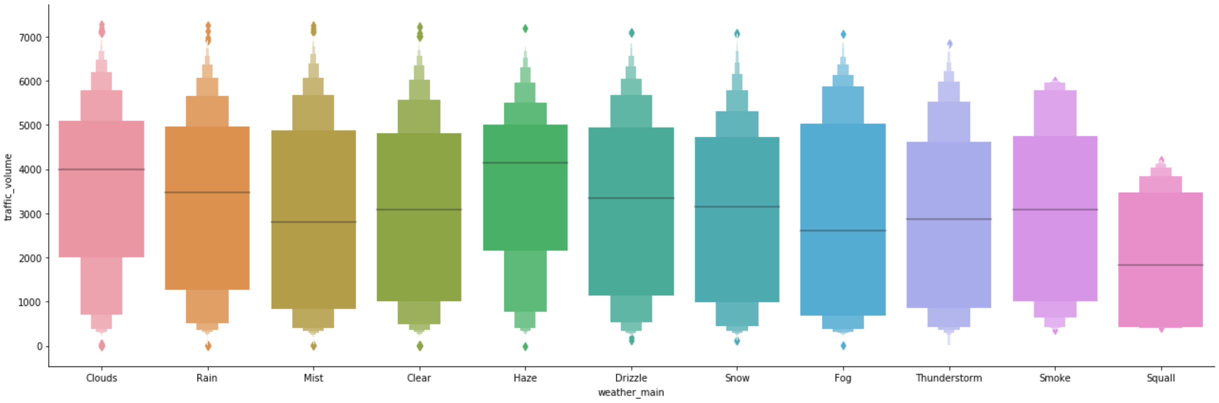


In [26]:
```
sns.catplot(y='traffic_volume',x='holiday',data=df.sort_values("traffic_volum
plt.show()
```



In [ ]:
```
#As usual the traffic is the most on normal days, after that we can consider 
```

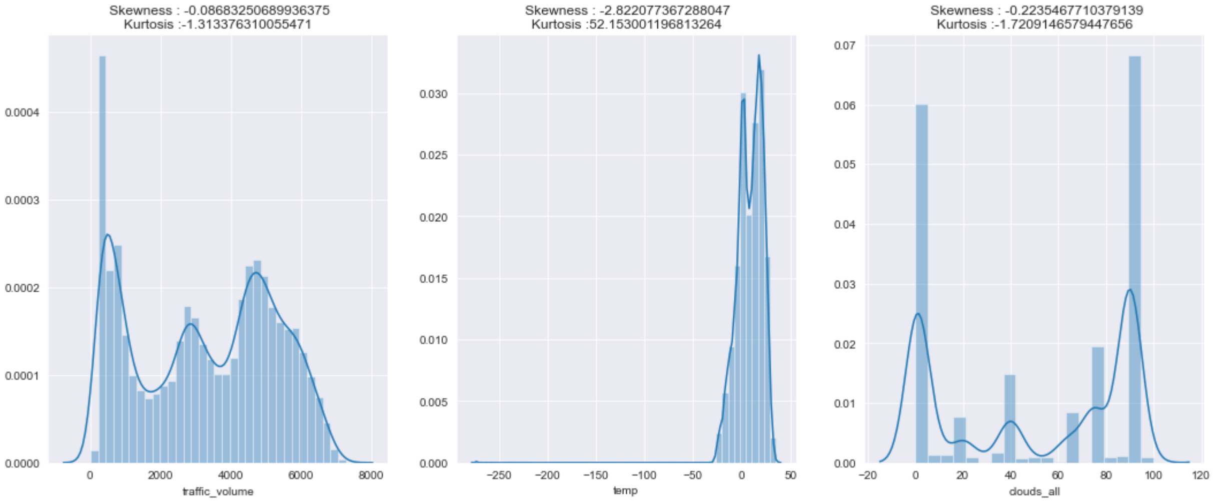
In [27]:
```
sns.catplot(y='traffic_volume',x='weather_main',data=df.sort_values("traffic_
plt.show()
```



Checking the skewnwss of data

In [78]:
```
def distribution_check(df,Cols):
    plt.figure(figsize=(18,15))
    fig = 1
    i = (len(Cols)//3)+1
    for col in Cols:
        sk = " Skewness : " + str(skew(df[col])) +"\nKurtosis :" + str(kurtos
        plt.subplot(i, 3, fig)
        sns.distplot(df[col]).set_title(sk)
        fig = fig+1
```

In [79]:
```
from scipy.stats import skew,kurtosis,zscore
distribution_check(df_traffic_data,['traffic_volume','temp','clouds_all'])
```

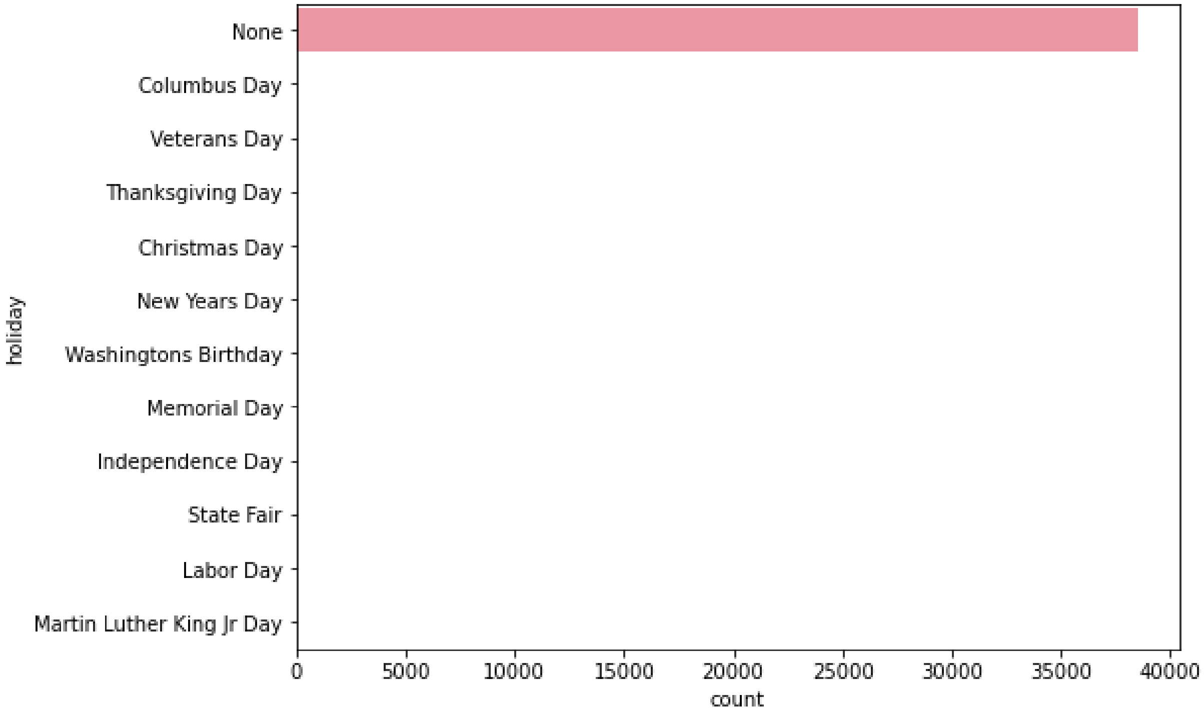

Skewness : -0.08683250689936375
Kurtosis :-1.313376310055471

Skewness : -2.822077367288047
Kurtosis :52.153001196813264

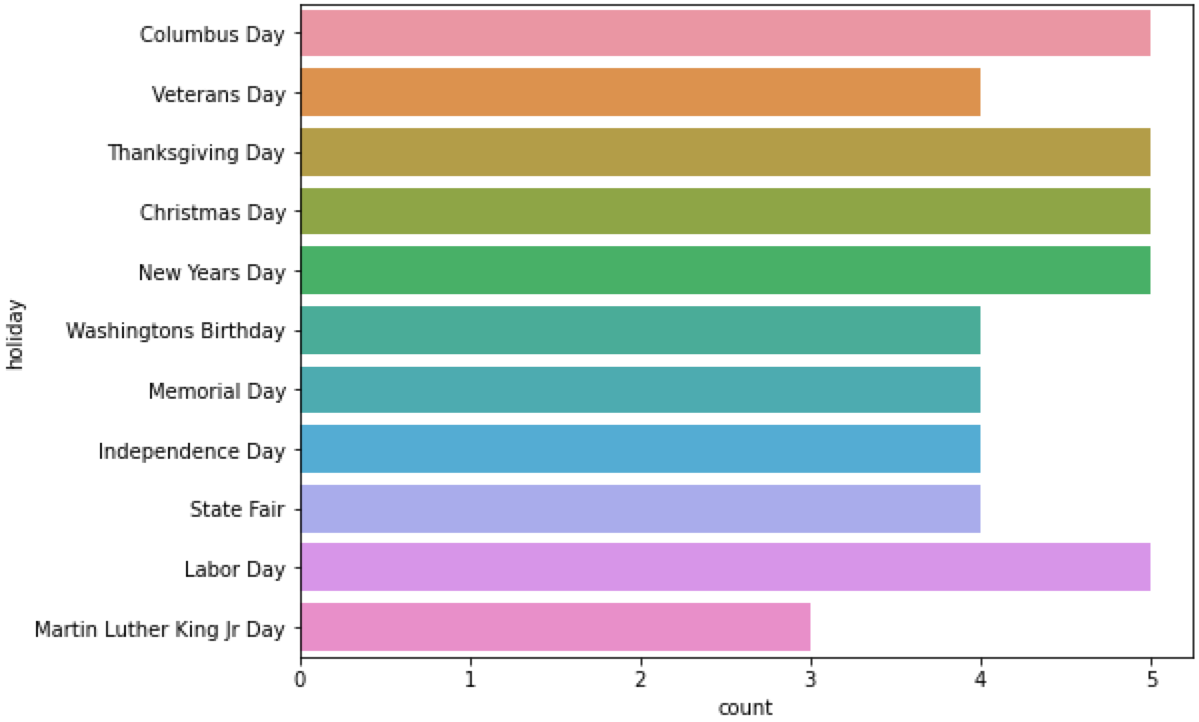
Skewness : -0.2235467710379139
Kurtosis :-1.7209146579447656

Univariate Analysis

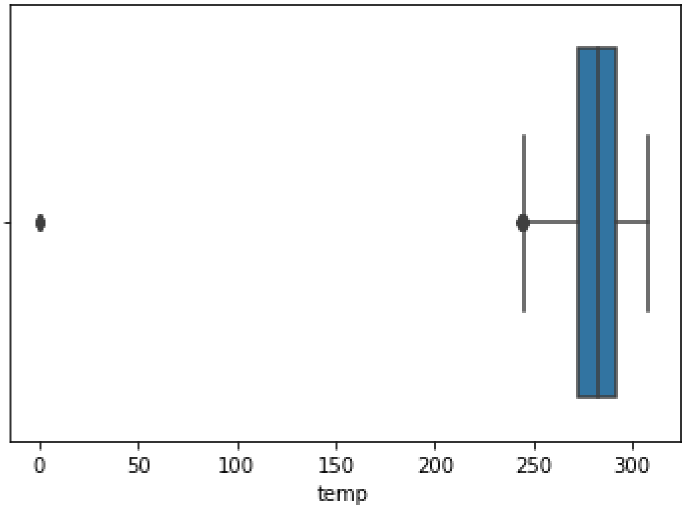
In [26]:
```
#Plotting frequency of each category in holiday column
plt.figure(figsize = (8,6))
sns.countplot(y='holiday', data = df_traffic_data)
plt.show()
```



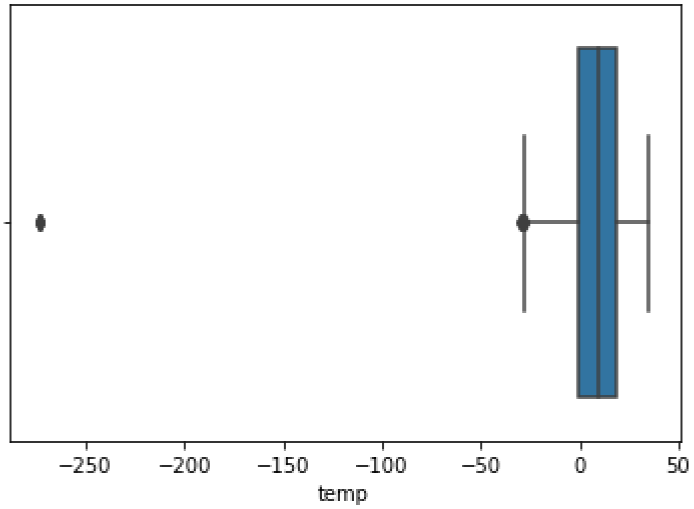
In [27]:
```
#'None' is far greater than the other days. Removing None data to visualize t
holidays = df_traffic_data.loc[df_traffic_data.holiday != 'None']
plt.figure(figsize=(8,6))
sns.countplot(y='holiday', data= holidays)
plt.show()
```



In [28]:
```
#plotting distribution of temperature variable
plt.figure(figsize=(6,4))
sns.boxplot('temp', data = df_traffic_data)
plt.show()
```

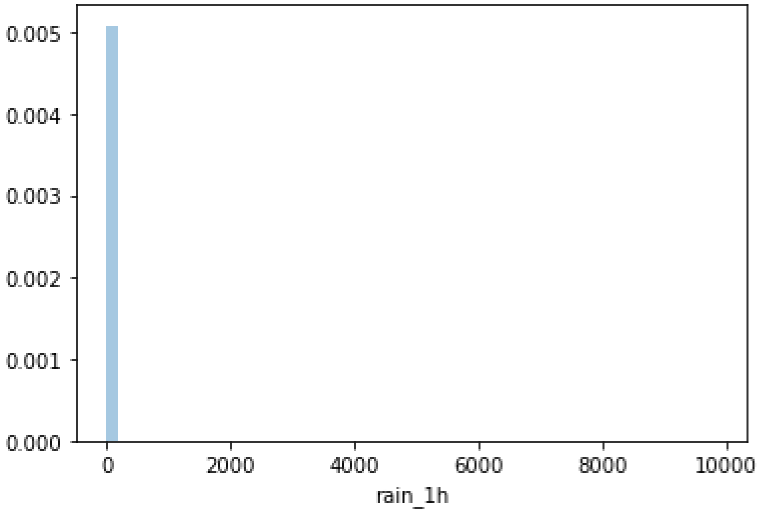


In [29]:
```
#Temperature is measured in Kelvin, changing to degree celsius to make it mor
#convert kelvin to celsius
#(0K - 273.15)
df_traffic_data['temp'] = (df_traffic_data['temp']-273.15)
plt.figure(figsize=(6,4))
sns.boxplot('temp', data = df_traffic_data)
plt.show()
```

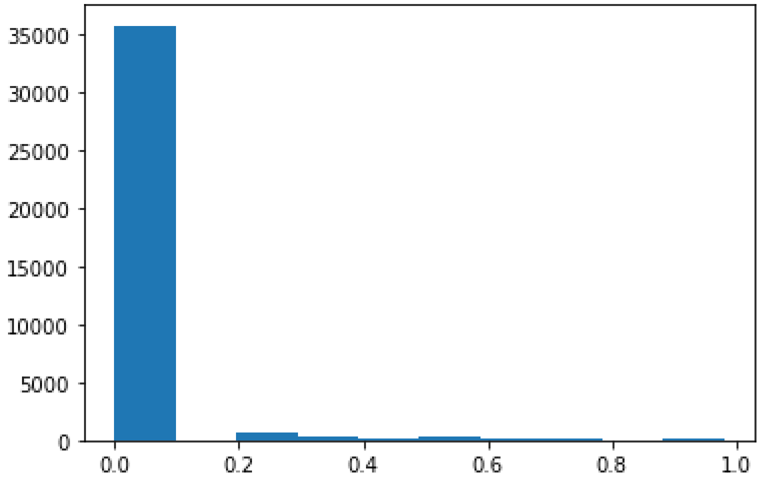


In [30]:
```
#There is one data point far away from the rest around -300 degrees celsius.
#Eliminating will be eliminated in the data cleaning phase.
```

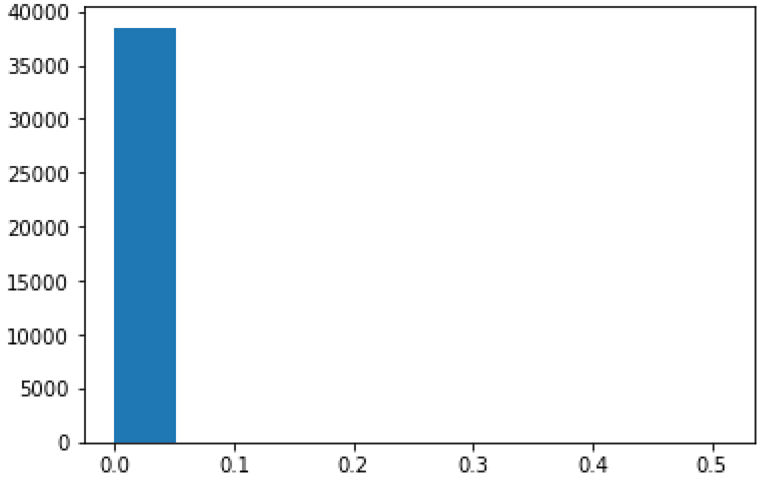
In [31]:
```
#Plotting rain variable
plt.figure(figsize=(6,4))
sns.distplot(df_traffic_data.rain_1h)
plt.show()
#From the distribution, it shows that the data is extremely skewed. Most of t
```



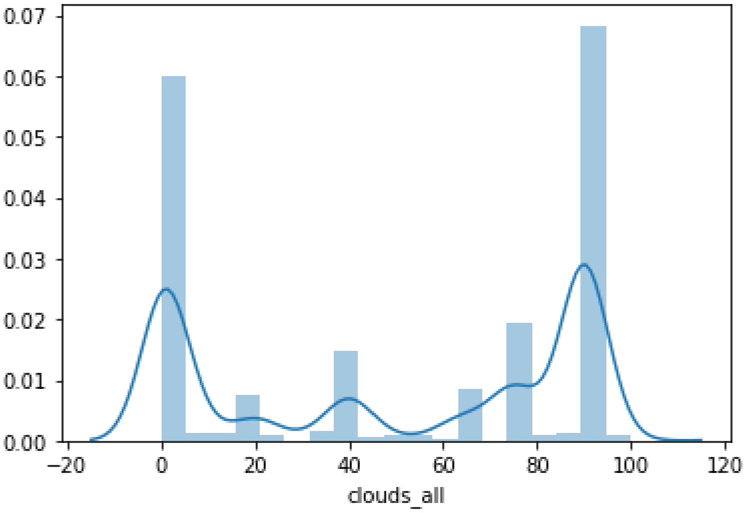
```
In [32]: #Plotting observations with values less than 1mm rain shows that more than 40
         plt.hist(df_traffic_data.rain_1h.loc[df_traffic_data.rain_1h<1])
         plt.show()
```



```
In [33]: #Plotting snow variable indicates that data is again skewed and most of the o
         plt.hist(df_traffic_data.snow_1h)
         plt.show()
```

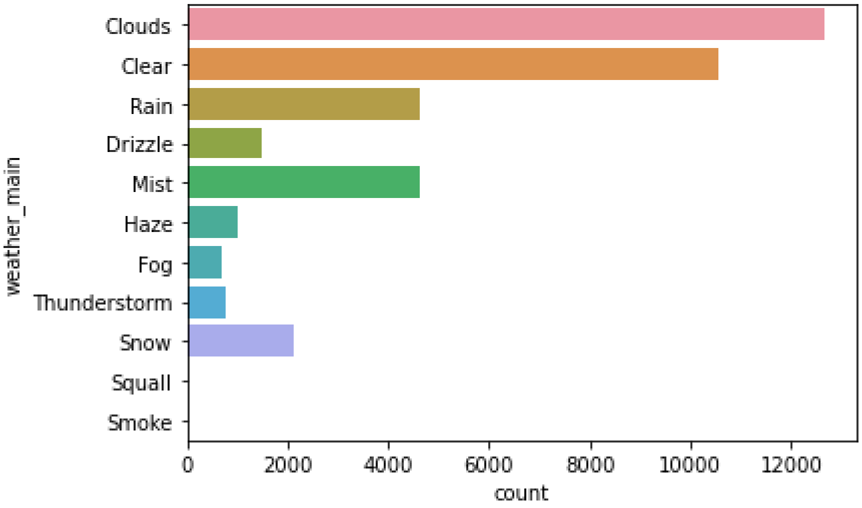


In [34]:
```
#clouds_all indicates the cloud coverage for the give day and hour
sns.distplot(df_traffic_data.clouds_all)
plt.show()
```

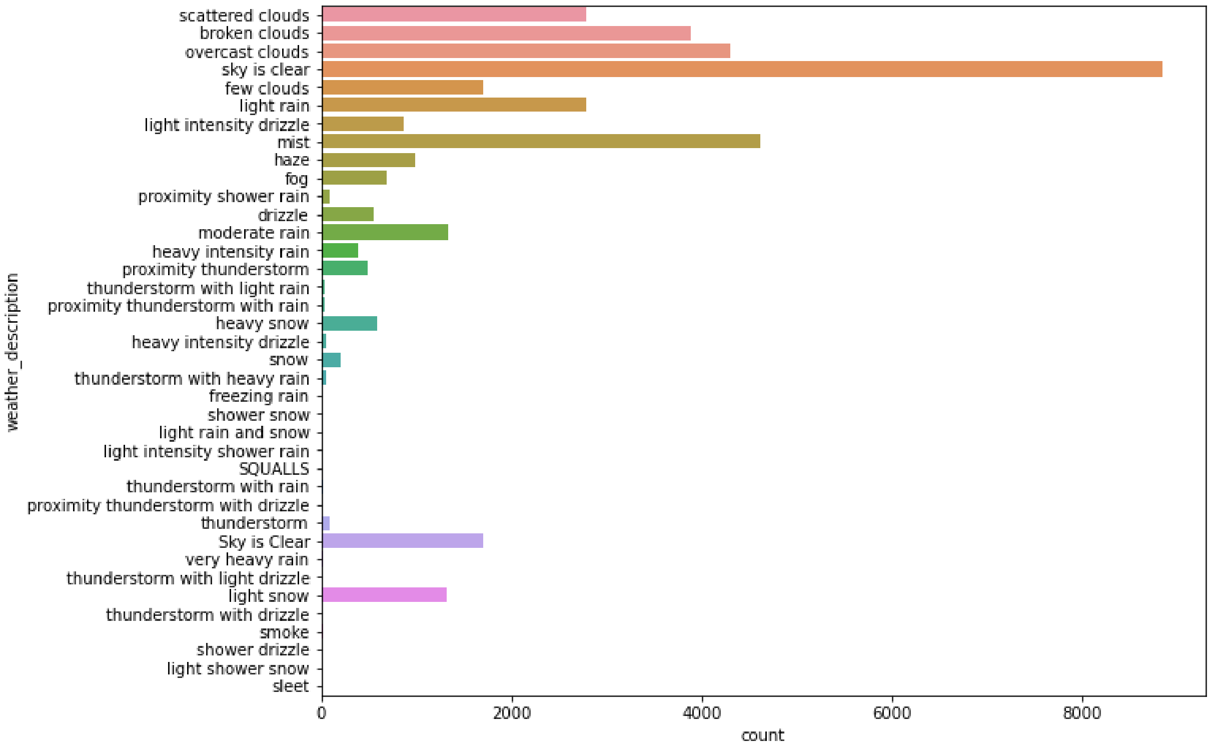


In [35]:
```
#exploring different categories in weather_main
sns.countplot(y='weather_main', data=df_traffic_data)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x134fab87280>
```



```
In [36]: #exploring different categories in weather_description
         plt.figure(figsize=(10,8))
         sns.countplot(y='weather_description', data=df_traffic_data)
         plt.show()
```

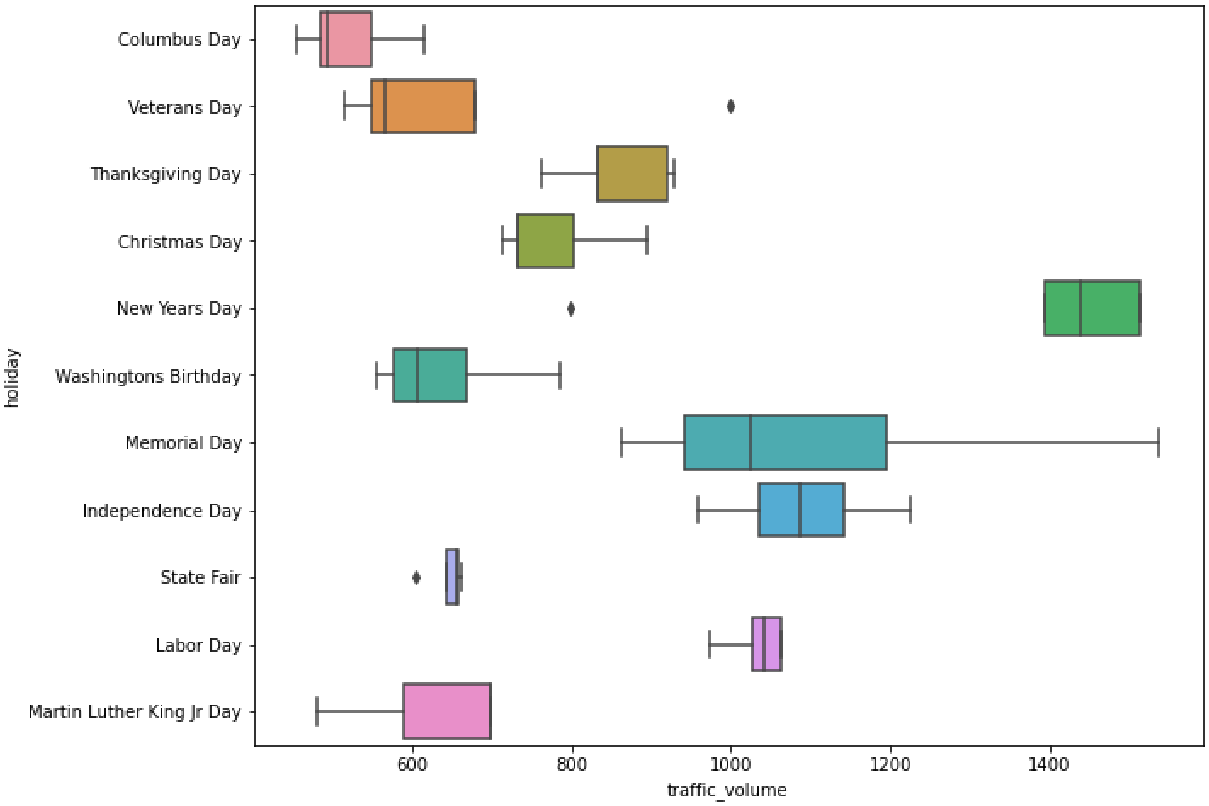


Bivariate Analysis

Exploring relationship between traffic and other features

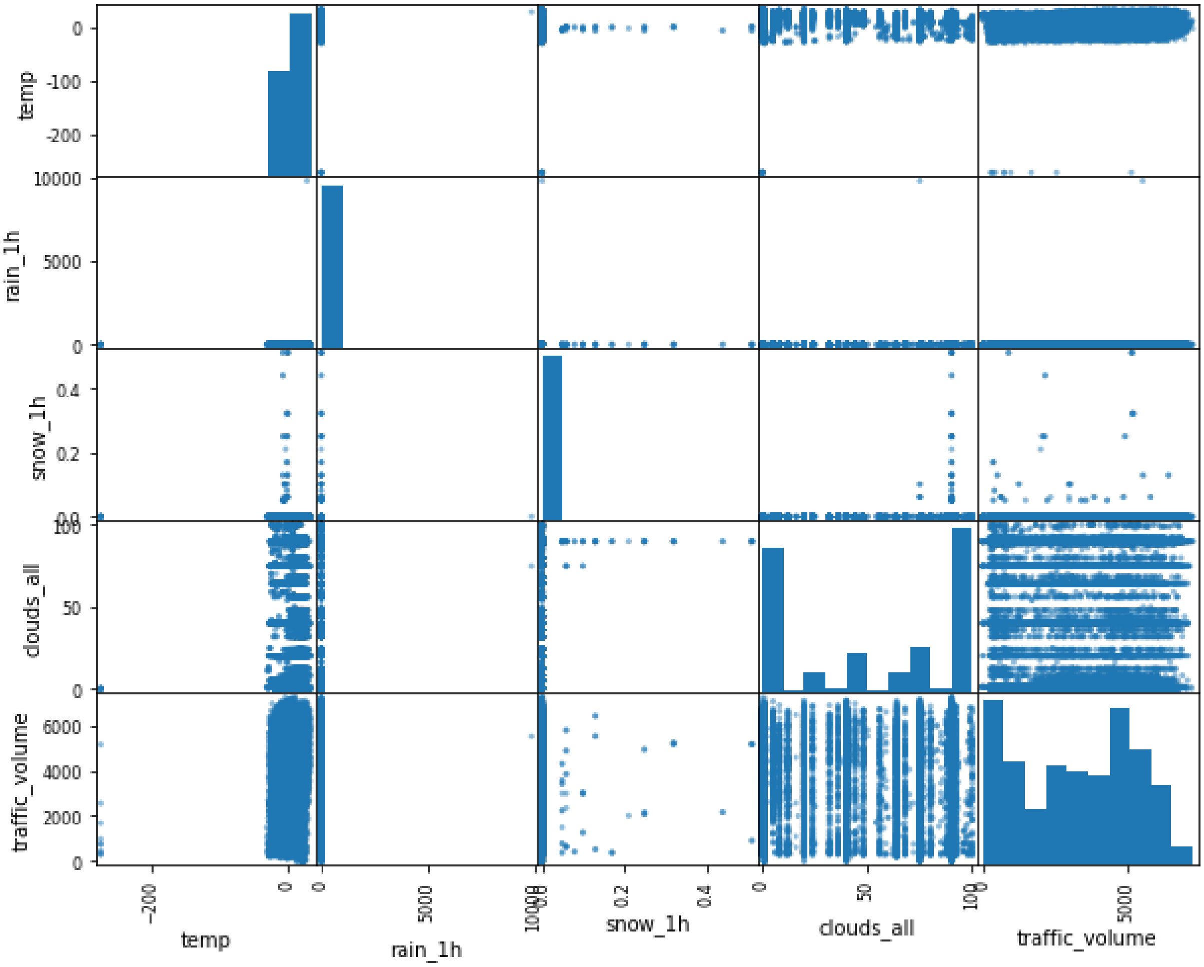
In [37]:

```
#Exploring traffic volume on holidays
plt.figure(figsize=(10,8))
sns.boxplot(y='holiday',x='traffic_volume', data = holidays)
plt.show()
```



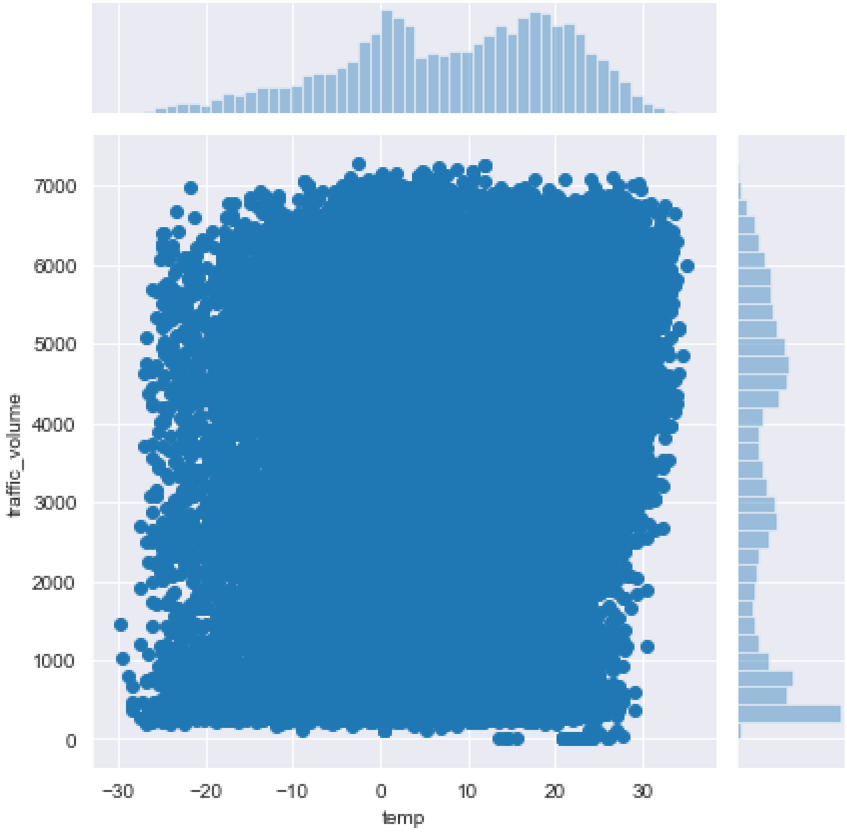
In [38]:

```
#Plotting relationship between temp, rain_1h, snow_1h, cloud_all.
num_vars = ['temp','rain_1h','snow_1h','clouds_all','traffic_volume']
from pandas.plotting import scatter_matrix
scatter_matrix(df_traffic_data[num_vars],figsize=(10,8))
plt.show()
```



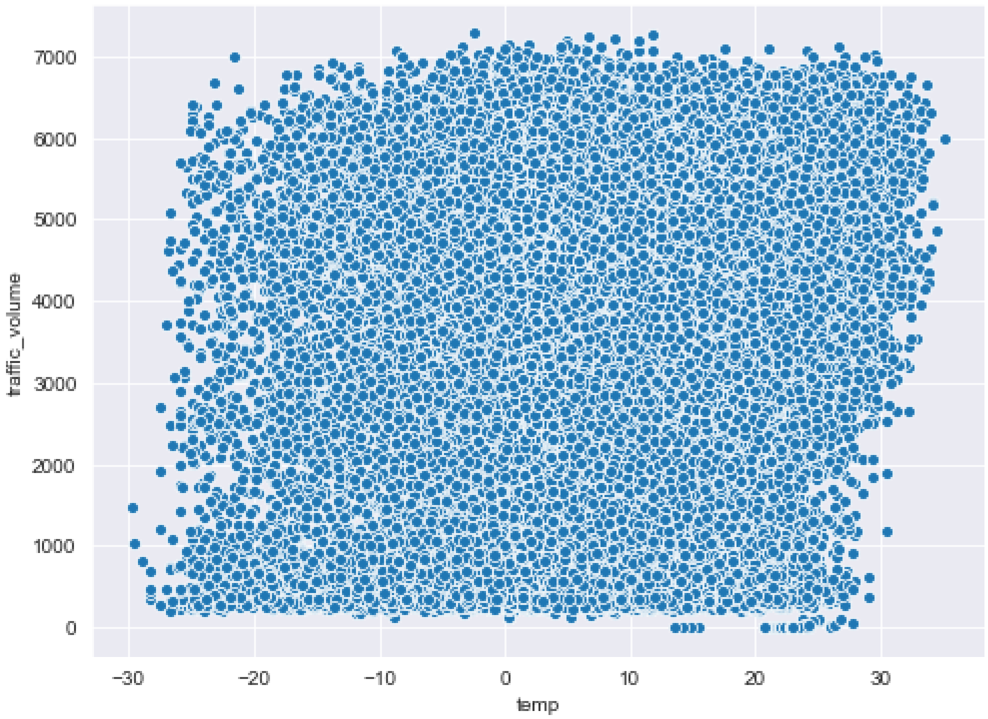
```
In [39]: #plotting temperature against traffic volume
         plt.figure(figsize=(10,8))
         sns.set_style('darkgrid')
         sns.jointplot(y='traffic_volume', x='temp', data = df_traffic_data.loc[df_tra
         plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```

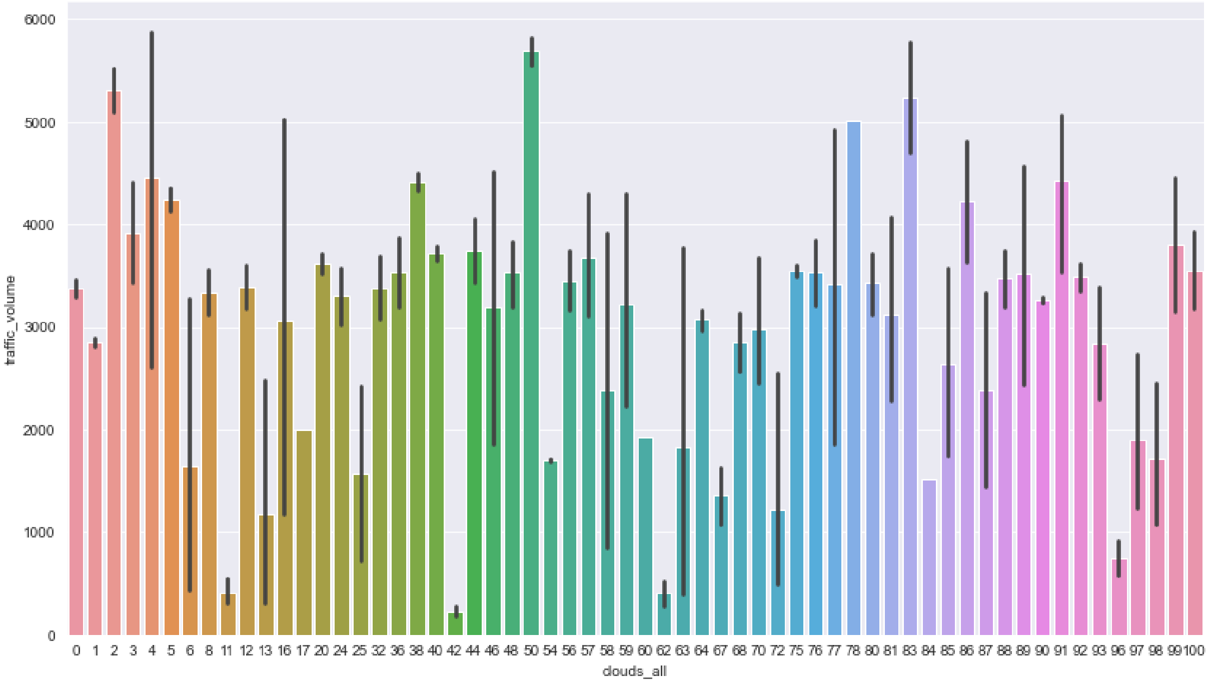


In [40]:
```
#scatterplot between traffic_volume and temp
plt.figure(figsize=(8,6))
sns.scatterplot(y='traffic_volume', x='temp', data = df_traffic_data.loc[df_t
```

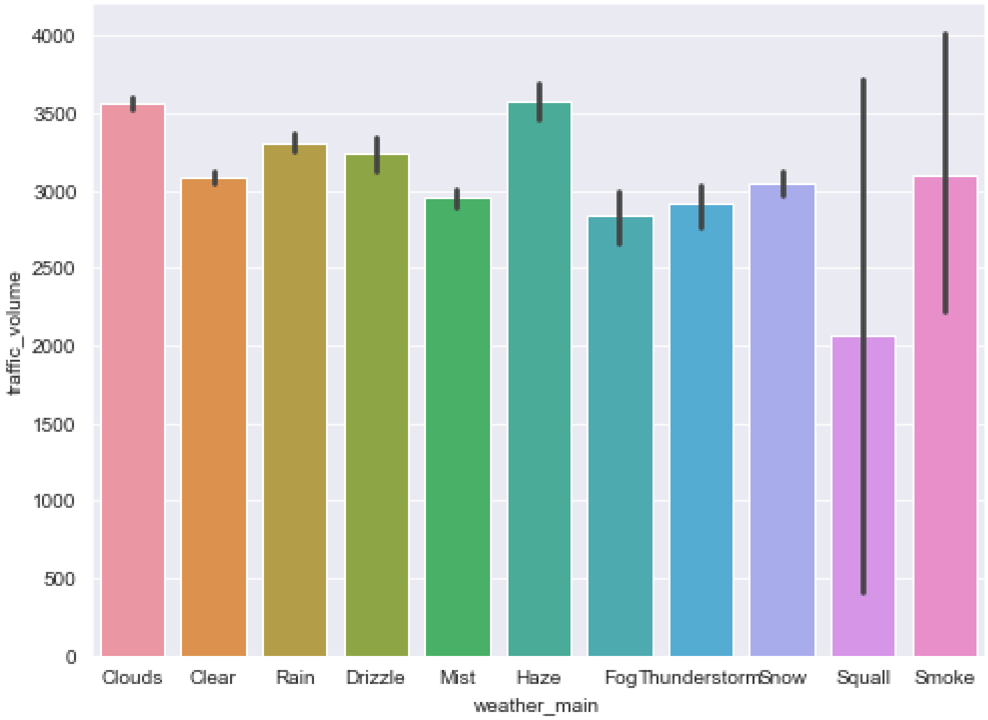
Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x134f96bc9a0>
```



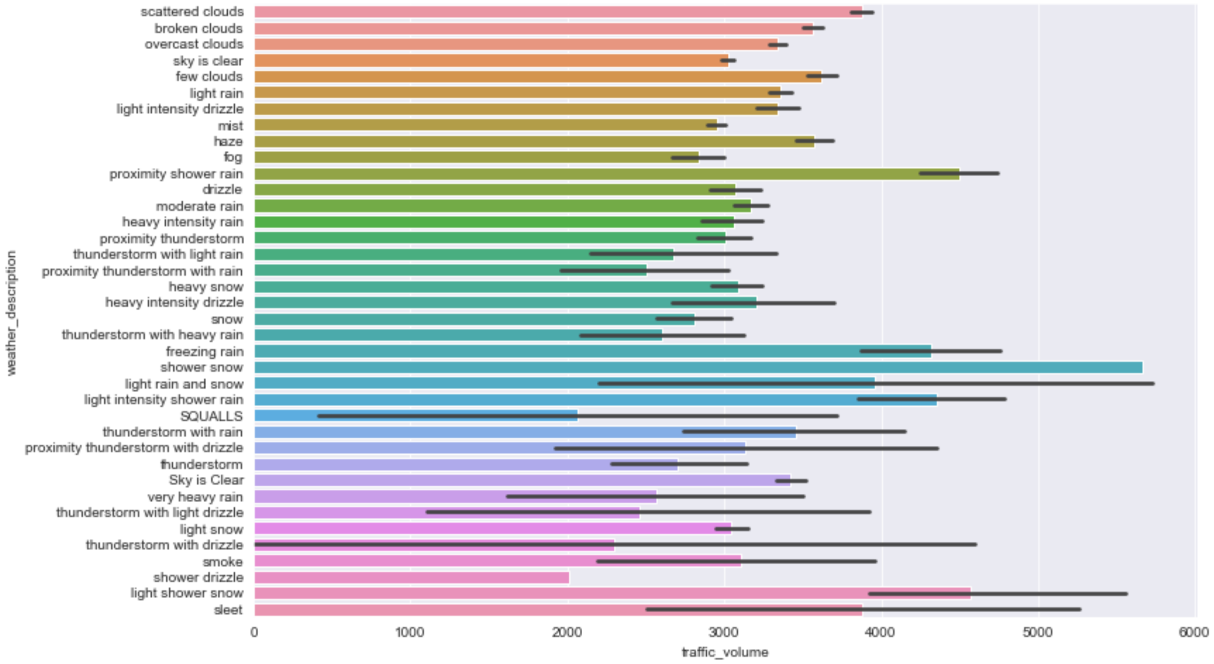
```
In [41]: #Plotting traffic volume over clouds_all
         plt.figure(figsize=(14,8))
         sns.barplot(x='clouds_all', y = 'traffic_volume', data = df_traffic_data)
         plt.show()
```



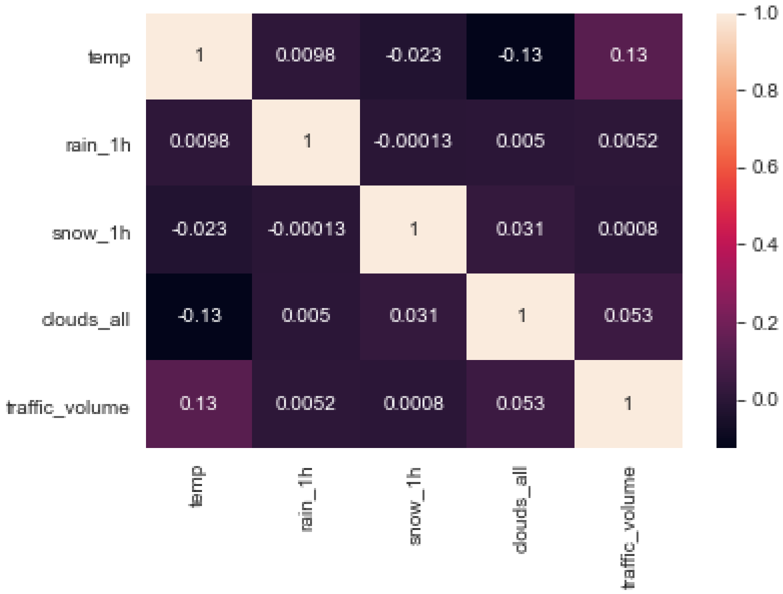
In [42]:
```
#Plotting weather_main over traffic volume
plt.figure(figsize=(8,6))
sns.barplot(x='weather_main', y = 'traffic_volume', data = df_traffic_data)
plt.show()
```



In [43]:
```
#Plotting weather_description over traffic volume
plt.figure(figsize=(12,8))
sns.barplot(y='weather_description', x = 'traffic_volume', data = df_traffic_
plt.show()
```



In [44]:
```
#correlation between different numeric variables. plot shows no strong correl
sns.heatmap(df_traffic_data.corr(), annot=True)
plt.show()
```



Feature engineering and Data cleaning

In [10]:
```
#copying data to new data frame
df_traffic_features = df_traffic_data.copy()
```

In [11]:
```
#Extracting features from date_time variable
df_traffic_features['date_time'] = pd.to_datetime(df_traffic_features.date_ti
df_traffic_features['weekday'] = df_traffic_features.date_time.dt.weekday
df_traffic_features['date'] = df_traffic_features.date_time.dt.date
df_traffic_features['hour'] = df_traffic_features.date_time.dt.hour
df_traffic_features['month'] = df_traffic_features.date_time.dt.month
df_traffic_features['year'] = df_traffic_features.date_time.dt.year
#Monday is 0 and Sunday is 6
```

In [12]:
```
df=df_traffic_features.copy()
```

In [13]:
```
df.head()
```

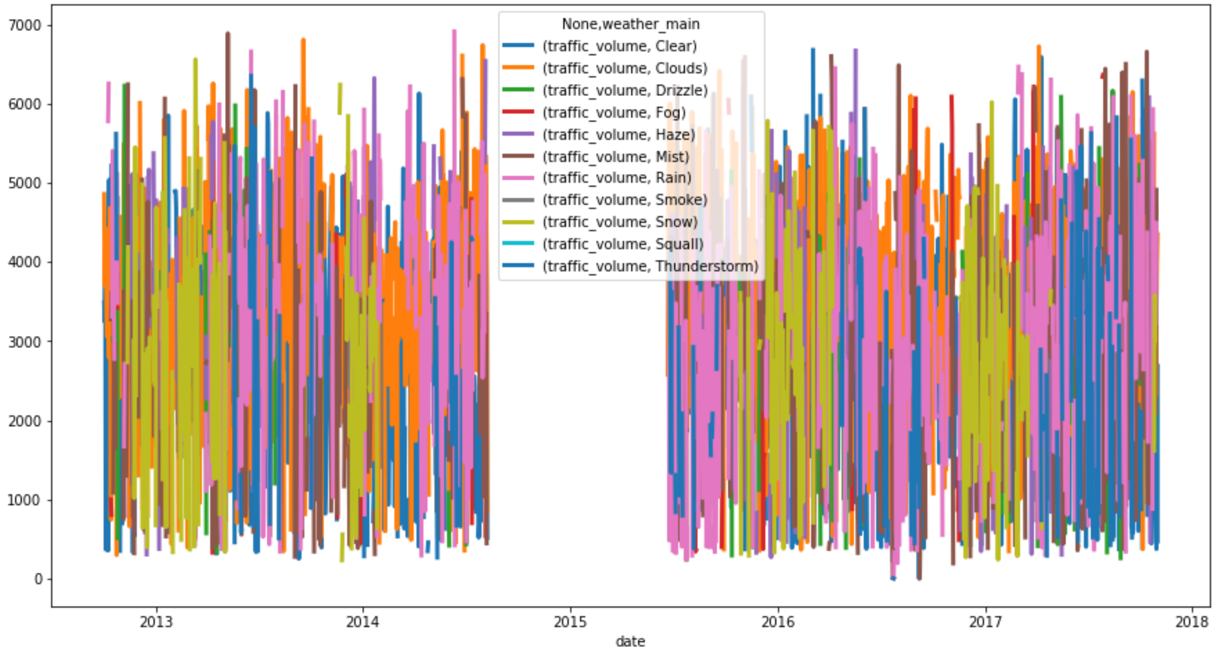
Out[13]:

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

In [14]:
```
df.weather_main.unique()
```

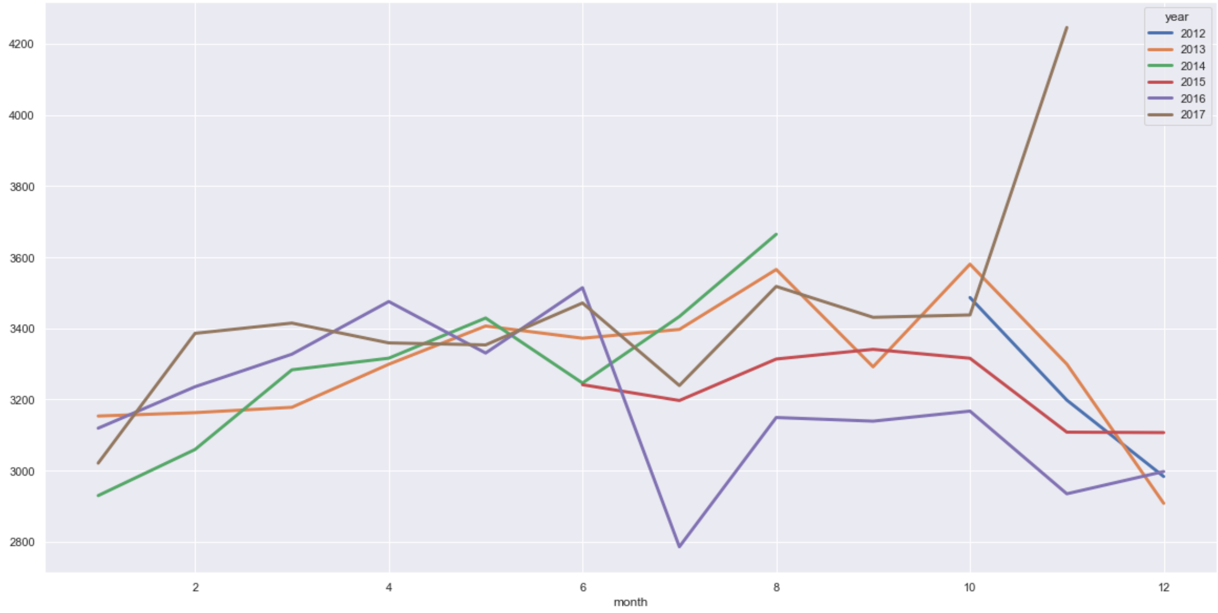
Out[14]:
```
array(['Clouds', 'Clear', 'Rain', 'Drizzle', 'Mist', 'Haze', 'Fog',
       'Thunderstorm', 'Snow', 'Squall', 'Smoke'], dtype=object)
```

```
In [15]: statelist = ['Clouds', 'Clear', 'Rain', 'Drizzle', 'Mist', 'Haze', 'Fog',
                'Thunderstorm', 'Snow', 'Squall', 'Smoke']
         stateseries = pd.DataFrame(df[(df['weather_main'].\
             isin(statelist))][['date','weather_main','traffic_volume']].\
             dropna().\
             groupby(['date', 'weather_main'])['weather_main','traffic_volume'].mean()
         stateseries.plot(figsize=(15,8), linewidth=3)
         plt.show()
```

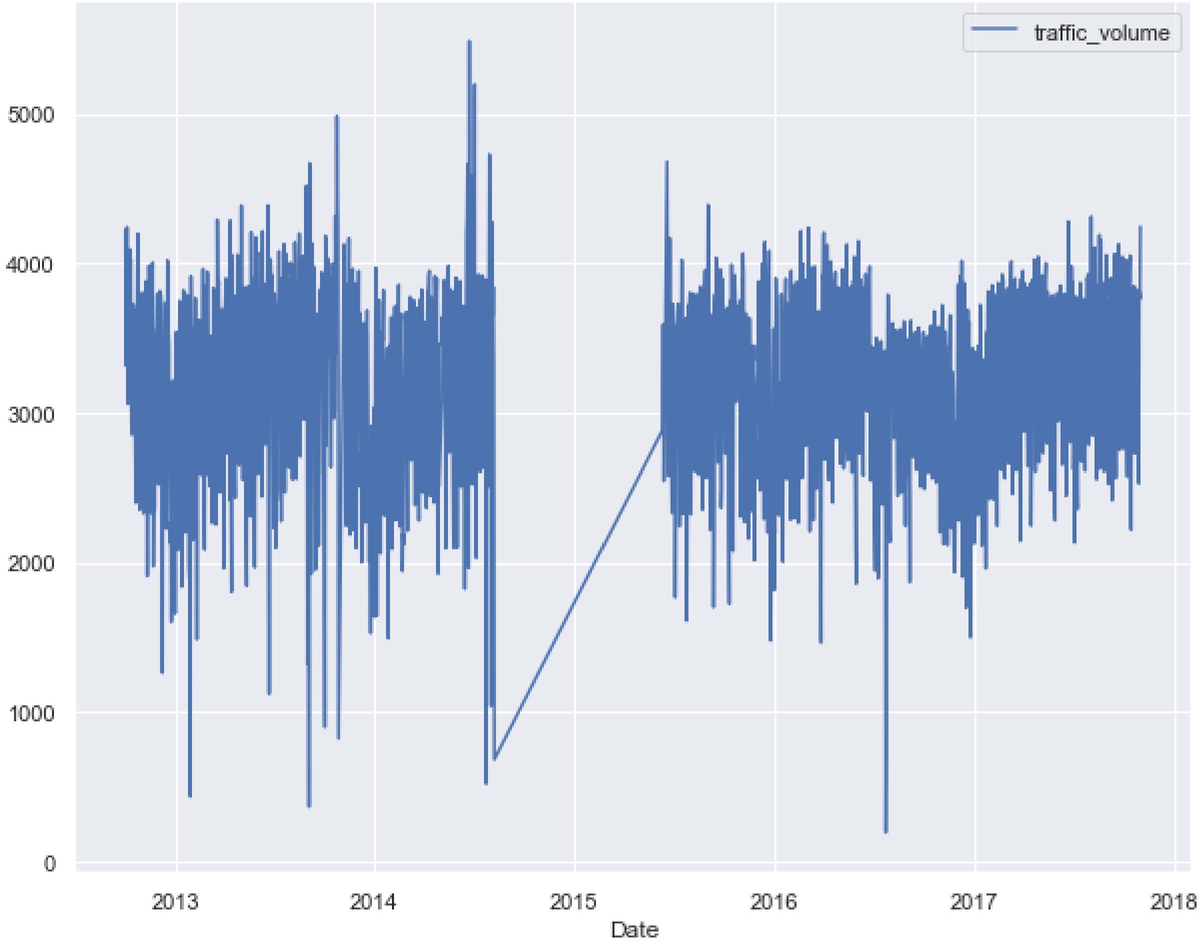


In [18]:

```
sns.set()
season = df
season['Date'] = df.date
season['Year'] =pd.to_datetime( df['Date']).dt.year
season['Month'] = pd.to_datetime(df['Date']).dt.month
spivot = pd.pivot_table(season, index='month', columns = 'year', values = 'tr
spivot.plot(figsize=(20,10), linewidth=3)
plt.show()
```

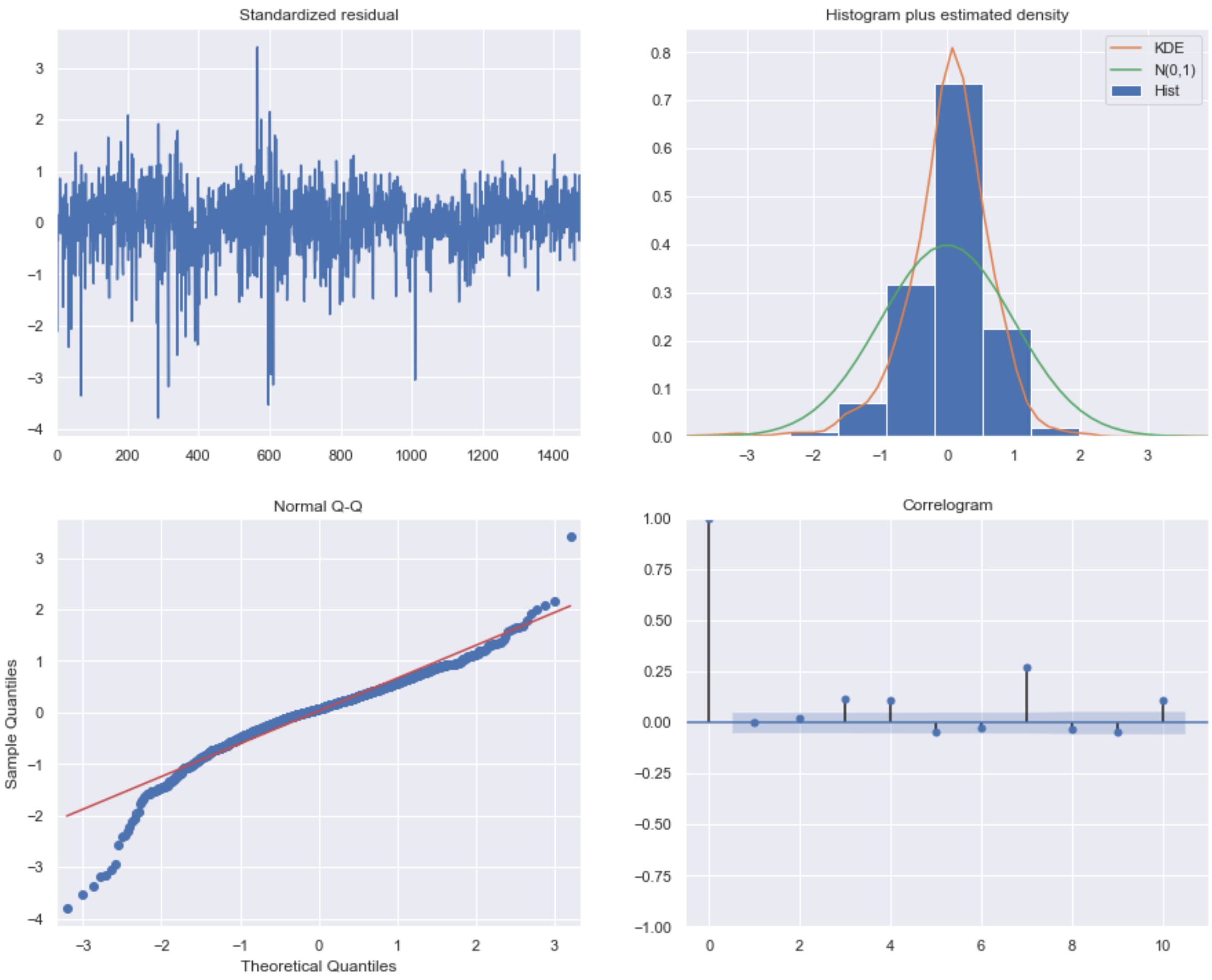


In [19]:
```
allhomes = df.groupby('Date')['Date','traffic_volume'].mean().dropna()
allhomes.plot(figsize=(10,8))
plt.show()
```



In [20]:

```
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
mod = sm.tsa.statespace.SARIMAX(allhomes,
                                order = (2, 0, 4),
                                seasonal_order = (3, 1, 2, 12),
                                enforce_stationarity = False,
                                enforce_invertibility = False)
results = mod.fit()
results.plot_diagnostics(figsize=(15,12))
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [47]:
```
df_traffic_features.head()
```

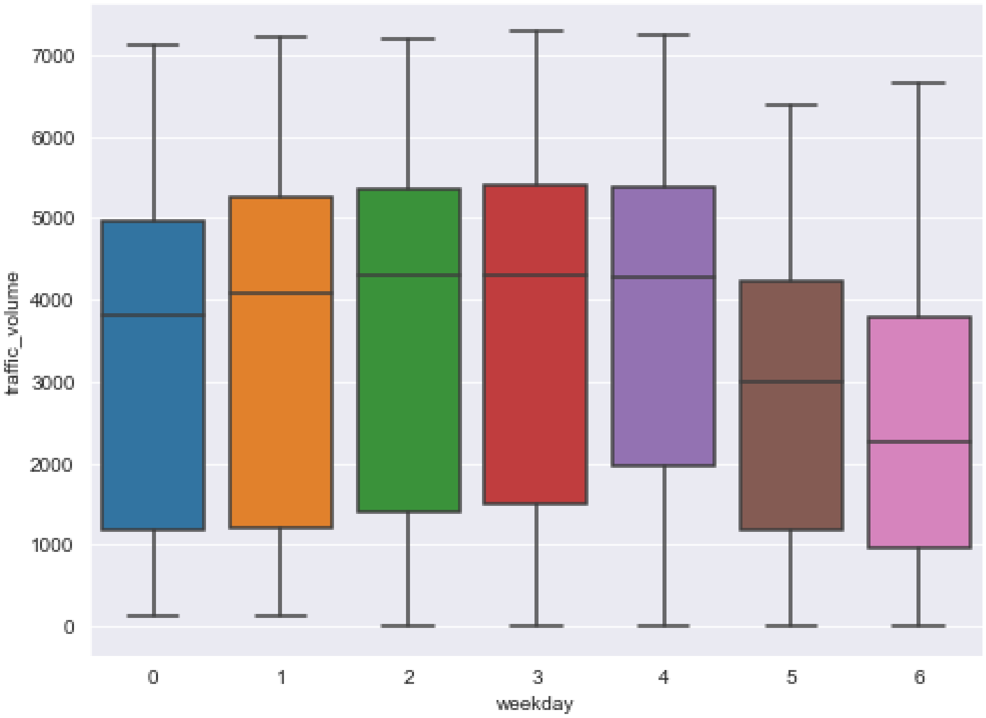
Out[47]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	tra
0	2012-10-02 09:00:00	None	15.13	0.0	0.0	40	Clouds	scattered clouds	
1	2012-10-02 10:00:00	None	16.21	0.0	0.0	75	Clouds	broken clouds	
2	2012-10-02 11:00:00	None	16.43	0.0	0.0	90	Clouds	overcast clouds	
3	2012-10-02 12:00:00	None	16.98	0.0	0.0	90	Clouds	overcast clouds	
4	2012-10-02 13:00:00	None	17.99	0.0	0.0	75	Clouds	broken clouds	

In [48]:
```
#categorizing hours to different time periods like morning, afternoon etc
def hour_modify(x):
    Early_Morning = [4,5,6,7]
    Morning = [8,9,10,11]
    Afternoon = [12,13,14,15]
    Evening = [16,17,18,19]
    Night = [20,21,22,23]
    Late_Night = [24,1,2,3]
    if x in Early_Morning:
        return 'Early_Morning'
    elif x in Morning:
        return 'Morning'
    elif x in Afternoon:
        return 'Afternoon'
    elif x in Evening:
        return 'Evening'
    elif x in Night:
        return 'Night'
    else:
        return 'Late_Night'

df_traffic_features['hour'] = df_traffic_features.hour.map(hour_modify)
```

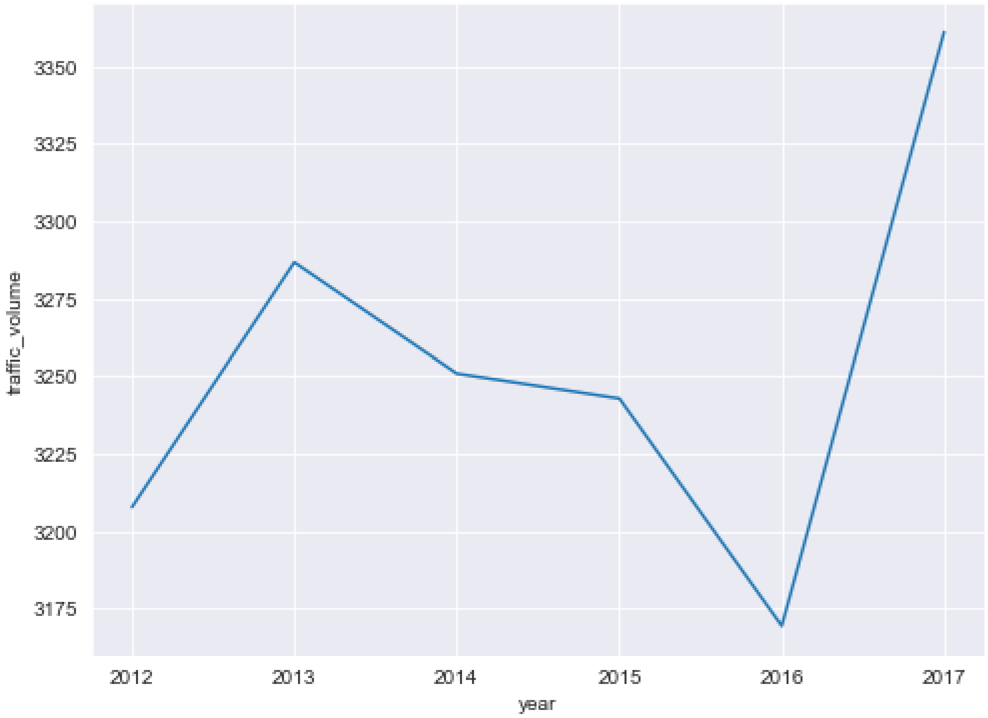
In [49]:

```
#Traffic volume plotted against weekday. Weekends show less traffic volume.
plt.figure(figsize=(8,6))
sns.boxplot(x='weekday', y='traffic_volume', data = df_traffic_features)
plt.show()
```



In [50]:

```
#aggreagating traffic volume over year and plotting

df_date_traffic = df_traffic_features.groupby('year').aggregate({'traffic_vol
plt.figure(figsize=(8,6))
sns.lineplot(x = df_date_traffic.index, y = df_date_traffic.traffic_volume, d
plt.show()
```



In [51]:

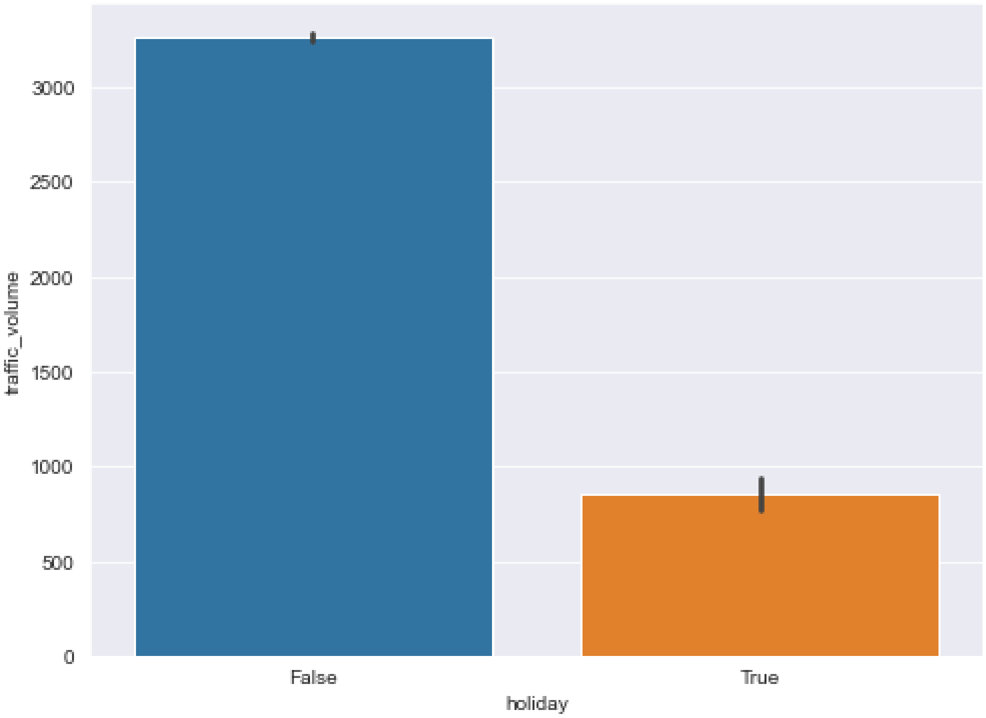
```
#Other holidays are very sparse compared to none holidays.
#Hence encoding the holidays as TRUE and none Holidays as FALSE

def modify_holiday(x):
    if x == 'None':
        return False
    else:
        return True
df_traffic_features['holiday'] = df_traffic_features['holiday'].map(modify_ho
```

In [52]:

```
#Outlier in temp which was detected earlier needs to be removed
df_traffic_features = df_traffic_features.loc[df_traffic_features.temp>-250]
```

In [53]:

```
#Traffic volume difference during holiday and non holiday
plt.figure(figsize=(8,6))
sns.barplot(x='holiday', y='traffic_volume', data = df_traffic_features)
plt.show()
```



In [54]:

```
#clouds, rain and snow distribution over different weather conditions
df_traffic_features.groupby('weather_description').aggregate({'traffic_volume
                                                             'clouds_all':'c
```

Out[54]:

	traffic_volume		clouds_all	rain_1h	snow_1h
	mean	size	count	mean	mean
weather_description					
SQUALLS	2061.750000	4	4	3.482500	0.000000
Sky is Clear	3420.036215	1712	1712	0.000000	0.000000
broken clouds	3564.464037	3879	3879	0.000000	0.000000
drizzle	3073.518051	554	554	0.116986	0.000000
few clouds	3619.433255	1708	1708	0.000000	0.000000
fog	2833.751804	693	693	0.071558	0.000823
freezing rain	4314.000000	2	2	0.000000	0.000000
haze	3574.350453	993	993	0.037100	0.000000
heavy intensity drizzle	3206.375000	56	56	0.101071	0.000000
heavy intensity rain	3057.023256	387	387	2.936770	0.000000
heavy snow	3085.862010	587	587	0.001295	0.000000
light intensity drizzle	3338.663605	871	871	0.170034	0.000000
light intensity shower rain	4351.545455	11	11	0.393636	0.000000
light rain	3359.250089	2795	2795	0.134544	0.000082
light rain and snow	3961.166667	6	6	0.211667	0.000000
light shower snow	4570.750000	4	4	0.000000	0.000000
light snow	3045.698027	1318	1318	0.049196	0.002269
mist	2951.615268	4611	4611	0.251527	0.000939
moderate rain	3171.570143	1333	1333	0.541028	0.000623
overcast clouds	3339.694561	4302	4302	0.000000	0.000000
proximity shower rain	4501.202128	94	94	0.080532	0.000000
proximity thunderstorm	3005.149284	489	489	1.322311	0.000000
proximity thunderstorm with drizzle	3131.500000	12	12	0.310000	0.000000
proximity thunderstorm with rain	2507.026316	38	38	0.586053	0.000000
scattered clouds	3875.658904	2791	2791	0.000000	0.000000
shower drizzle	2010.000000	1	1	0.000000	0.000000
shower snow	5664.000000	1	1	0.000000	0.000000
sky is clear	3023.134193	8838	8838	0.000000	0.000000
sleet	3882.000000	2	2	0.000000	0.000000
smoke	3103.722222	18	18	0.585556	0.000000

	traffic_volume		clouds_all	rain_1h	snow_1h
	mean	size	count	mean	mean
weather_description					
snow	2808.035176	199	199	0.028492	0.008894
thunderstorm	2701.500000	88	88	1.130341	0.000000
thunderstorm with drizzle	2297.000000	2	2	5.345000	0.000000
thunderstorm with heavy rain	2603.857143	56	56	3.294821	0.000000
thunderstorm with light drizzle	2463.375000	8	8	1.002500	0.000000
thunderstorm with light rain	2673.926829	41	41	0.867317	0.000000
thunderstorm with rain	3456.322581	31	31	2.028387	0.000000
very heavy rain	2568.833333	18	18	570.208333	0.000000

In [55]:
```python
df_traffic_features['weather_description'] = df_traffic_features['weather_des
```

In [56]:
```python
#The weather description mostly describes rain, snow, thunderstorms, fog, mis

#I will create following new columns:
#thunderstorm - True where weather description contains Thunderstorm else Fal
#fog - True where weather description contains fog else False
#mist - True where weather description contains mist else False
#haze - True where weather description contains haze else False
```

In [57]:
```python
#Any row containing "thunderstorm" is replaced by "thunderstorm"
df_traffic_features.loc[df_traffic_features['weather_description'].str.contai
```

In [58]:
```python
weather = ['thunderstorm','mist','fog','haze']
df_traffic_features.loc[np.logical_not(df_traffic_features['weather_descripti
```

In [59]:
```python
df_traffic_features.weather_description.value_counts()
```

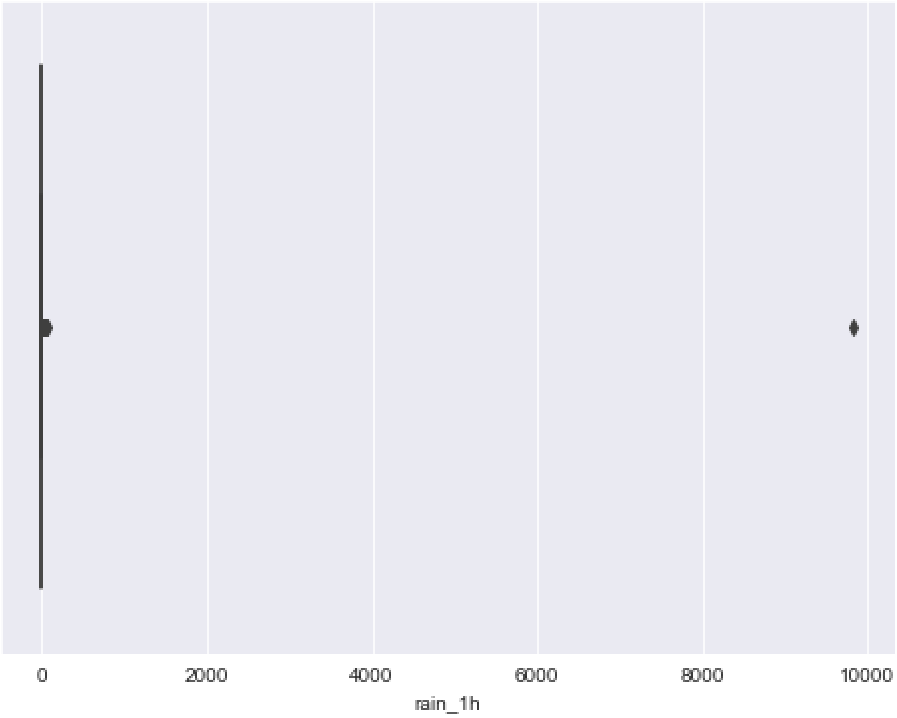
Out[59]:
```
other           31491
mist             4611
haze              993
thunderstorm      765
fog               693
Name: weather_description, dtype: int64
```

In [60]:
```python
#creating dummy variables for these newly created categories in weather descr
df_traffic_features = pd.get_dummies(columns=['weather_description'],data=df_
In [61]: df_traffic_features.rename(columns={'weather_description_fog':'fog', 'weather
                                   'weather_description_mist':'mist', 'weathe
df_traffic_features.drop(columns = ['weather_description_other', 'weather_mai
```

```
In [62]: df_traffic_features.columns
```

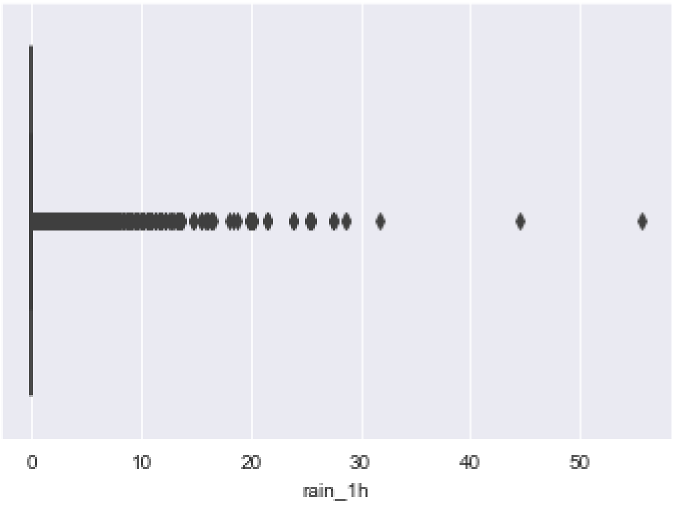
```
Out[62]: Index(['date_time', 'holiday', 'temp', 'rain_1h', 'snow_1h', 'clouds_all',
                'traffic_volume', 'weekday', 'date', 'hour', 'month', 'year', 'fog',
                'haze', 'mist', 'thunderstorm'],
               dtype='object')
```

```
In [63]: #Plotiing rain data shows one outlier data point. Lets remove it.
         plt.figure(figsize=(8,6))
         sns.boxplot('rain_1h',data = df_traffic_features)
         plt.show()
```



```
In [64]: sns.boxplot('rain_1h',data = df_traffic_features.loc[df_traffic_features.rain_
```

```
Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x134f94f1d90>
```



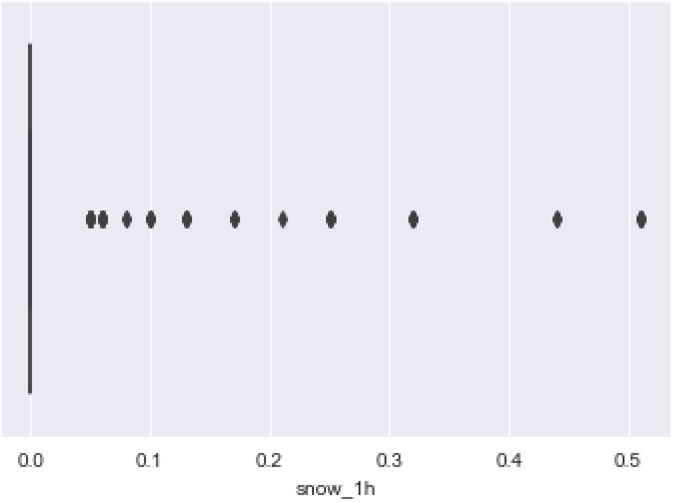
In [65]:
```
#Removing outlier in rain column and converting numeric data to categories
#rain value equal to 0.0 as no_rain
#rain value greater than 0.0 is cut into 3 quantiles

df_traffic_features = df_traffic_features.loc[df_traffic_features.rain_1h<200
df_traffic_features_temp = df_traffic_features.loc[df_traffic_features.rain_1
rain_q = pd.DataFrame(pd.qcut(df_traffic_features_temp['rain_1h'] ,q=3, label
df_traffic_cat = df_traffic_features.merge(rain_q,left_index=True, right_inde
df_traffic_cat['rain_1h_y'] = df_traffic_cat.rain_1h_y.cat.add_categories('no
df_traffic_cat['rain_1h_y'].fillna('no_rain', inplace = True) #no_rain is not

df_traffic_cat.drop(columns=['rain_1h_x'], inplace = True)
df_traffic_cat.rename(columns={'rain_1h_y':'rain_1h'}, inplace = True)
df_traffic_cat.head()
```

Out[65]:

	date_time	holiday	temp	snow_1h	clouds_all	traffic_volume	weekday	date	hour	mont
0	2012-10-02 09:00:00	False	15.13	0.0	40	5545	1	2012-10-02	Morning	1
1	2012-10-02 10:00:00	False	16.21	0.0	75	4516	1	2012-10-02	Morning	1
2	2012-10-02 11:00:00	False	16.43	0.0	90	4767	1	2012-10-02	Morning	1
3	2012-10-02 12:00:00	False	16.98	0.0	90	5026	1	2012-10-02	Afternoon	1
4	2012-10-02 13:00:00	False	17.99	0.0	75	4918	1	2012-10-02	Afternoon	1

In [66]:
```
#Plotiing snow data shows that it is extremely skewed as observed during univ
sns.boxplot('snow_1h',data = df_traffic_features)
```

Out[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x134fb35e3d0>
```



In [67]:
```
#only 63 observations have snow greater than 0.0, it can be encoded as no_sno
df_traffic_features.snow_1h[df_traffic_features.snow_1h>0].count()
#63 columns -> change to snow, no_snow
```

Out[67]:
```
63
```

In [68]:
```
def modify_snow1h(x):
    if x==0:
        return 'no_snow'
    else:
        return 'snow'

df_date_traffic['snow_1h'] = df_traffic_cat.snow_1h.map(modify_snow1h)
```

In [69]:
```
df_traffic_features.head()
```

Out[69]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	traffic_volume	weekday	date	ho
0	2012-10-02 09:00:00	False	15.13	0.0	0.0	40	5545	1	2012-10-02	Morni
1	2012-10-02 10:00:00	False	16.21	0.0	0.0	75	4516	1	2012-10-02	Morni
2	2012-10-02 11:00:00	False	16.43	0.0	0.0	90	4767	1	2012-10-02	Morni
3	2012-10-02 12:00:00	False	16.98	0.0	0.0	90	5026	1	2012-10-02	Afterno
4	2012-10-02 13:00:00	False	17.99	0.0	0.0	75	4918	1	2012-10-02	Afterno

In [70]:
```
#setting date as index
df_traffic_cat.set_index('date', inplace = True)
```

In [71]:
```
df_traffic_cat.columns
```

Out[71]:
```
Index(['date_time', 'holiday', 'temp', 'snow_1h', 'clouds_all',
       'traffic_volume', 'weekday', 'hour', 'month', 'year', 'fog', 'haze',
       'mist', 'thunderstorm', 'rain_1h'],
      dtype='object')
```