In [1]:

```python
import numpy as np
import pandas as pd
```

In [2]:

```
!wget -O "ahalytix_internship_hiring_ai_challenge-dataset.zip" "https://dockship-job-models.s3.ap-south-1.amazonaws.com/cdc11b6409c317a6afdf94fbeb1b727e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIDOPTEUZ2LEOQEGQ%2F20201216%2Fap-south-1%2Fs3%2Faws4_request&X-Amz-Date=20201216T133708Z&X-Amz-Expires=1800&X-Amz-Signature=5e78e73dd1231cce9b719e12618f2535c8ba719ff75929c4ada1432d681ddc8d&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3D%22ahalytix_internship_hiring_ai_challenge-dataset.zip%22"
```

```
--2020-12-16 13:37:40--  https://dockship-job-models.s3.ap-south-1.amazonaws.com/cdc11b6409c317a6afdf94fbeb1b727e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIDOPTEUZ2LEOQEGQ%2F20201216%2Fap-south-1%2Fs3%2Faws4_request&X-Amz-Date=20201216T133708Z&X-Amz-Expires=1800&X-Amz-Signature=5e78e73dd1231cce9b719e12618f2535c8ba719ff75929c4ada1432d681ddc8d&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3D%22ahalytix_internship_hiring_ai_challenge-dataset.zip%22
Resolving dockship-job-models.s3.ap-south-1.amazonaws.com (dockship-job-models.s3.ap-south-1.amazonaws.com)... 52.219.64.68
Connecting to dockship-job-models.s3.ap-south-1.amazonaws.com (dockship-job-models.s3.ap-south-1.amazonaws.com)|52.219.64.68|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 410779 (401K) [binary/octet-stream]
Saving to: ‘ahalytix_internship_hiring_ai_challenge-dataset.zip’

ahalytix_internship 100%[===================>] 401.15K   350KB/s    in 1.1s

2020-12-16 13:37:42 (350 KB/s) - ‘ahalytix_internship_hiring_ai_challenge-dataset.zip’ saved [410779/410779]
```

In [3]:

```
!unzip 'ahalytix_internship_hiring_ai_challenge-dataset.zip'
```

```
Archive:  ahalytix_internship_hiring_ai_challenge-dataset.zip
  inflating: sample_submission.csv
  inflating: TEST.csv
  inflating: TRAIN.csv
```

In [4]:

```python
data=pd.read_csv('TRAIN.csv')
```

In [5]:

```python
df=data.copy()
```

In [6]:

```python
df.head()
```

Out[6]:

	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 [7]:

```
df.tail()
```

Out[7]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume
38558	2017-11-01 18:00:00	None	274.35	0.0	0.0	90	Snow	light snow	4297
38559	2017-11-01 19:00:00	None	274.62	0.0	0.0	90	Drizzle	light intensity drizzle	3045
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

In [8]:

```
df.shape
```

Out[8]:

```
(38563, 9)
```

In [9]:

```
df.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 [10]:

```
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
temp	38563.0	281.351757	13.216927	0.0	272.858	282.75	291.54	308.24
rain_1h	38563.0	0.392733	50.075055	0.0	0.000	0.00	0.00	9831.30
snow_1h	38563.0	0.000278	0.009131	0.0	0.000	0.00	0.00	0.51
clouds_all	38563.0	49.920364	38.849106	0.0	1.000	64.00	90.00	100.00
traffic_volume	38563.0	3260.940409	1991.628329	0.0	1186.500	3378.00	4939.00	7280.00

Checking null values if any in our data

In [11]:

```
df.isna().sum()
```

Out[11]:

```
date_time              0
```

```
holiday                0
temp                   0
rain_1h                0
snow_1h                0
clouds_all             0
weather_main           0
weather_description    0
traffic_volume         0
dtype: int64
```

Extracting new features from date_time column of our data

In [12]:

```
df['date_time'].head()
```

Out[12]:

```
0    2012-10-02 09:00:00
1    2012-10-02 10:00:00
2    2012-10-02 11:00:00
3    2012-10-02 12:00:00
4    2012-10-02 13:00:00
Name: date_time, dtype: object
```

In [13]:

```
df['date_time']=pd.to_datetime(df['date_time'])
```

In [14]:

```
df['date_time'].head()
```

Out[14]:

```
0   2012-10-02 09:00:00
1   2012-10-02 10:00:00
2   2012-10-02 11:00:00
3   2012-10-02 12:00:00
4   2012-10-02 13:00:00
Name: date_time, dtype: datetime64[ns]
```

In [15]:

```
df['year'] = df['date_time'].dt.year
df['month'] = df['date_time'].dt.month
df['week'] = df['date_time'].dt.week
df['day'] = df['date_time'].dt.day
df['hour'] = df['date_time'].dt.hour
df['week_day'] = df['date_time'].dt.dayofweek
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: FutureWarning: Series.dt.
weekofyear and Series.dt.week have been deprecated.  Please use Series.dt.isocalendar().w
eek instead.
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [16]:

```
df.head()
```

Out[16]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	month
0	2012-10-02 09:00:00	None	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	10
1	2012-10-02 10:00:00	None	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012	10
	2012-10-										

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	month
2	2012-10-02 11:00:00	None	289.58	0.0	0.0	90	Clouds	overcast clouds	4767	2012	10
3	2012-10-02 12:00:00	None	290.13	0.0	0.0	90	Clouds	overcast clouds	5026	2012	10
4	2012-10-02 13:00:00	None	291.14	0.0	0.0	75	Clouds	broken clouds	4918	2012	10

In [ ]:

```

```

In [ ]:

```

```

EXPLORATORY DATA ANALYSIS (EDA)

In [17]:

```
# import library for data visualization

import matplotlib.pyplot as plt
import seaborn as sns
```

In [18]:

```
# convert holiday column into binary data
# 0 --> not holiday , 1 --> holiday

df['holiday']=df['holiday'].apply(lambda x: 0 if x=='None' else 1)
```

In [19]:

```
df.head(2)
```

Out[19]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	month
0	2012-10-02 09:00:00	0	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	10
1	2012-10-02 10:00:00	0	289.36	0.0	0.0	75	Clouds	broken clouds	4516	2012	10

holiday vs traffic_volume

In [20]:

```
df['holiday'].value_counts()
```

Out[20]:

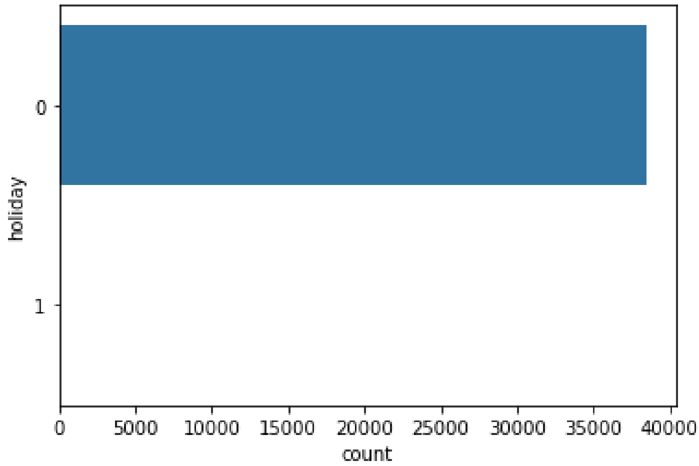
```
0    38515
1       48
Name: holiday, dtype: int64
```

In [21]:

```
sns.countplot(y='holiday',data=df)
```

Out[21]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef45e82d90>

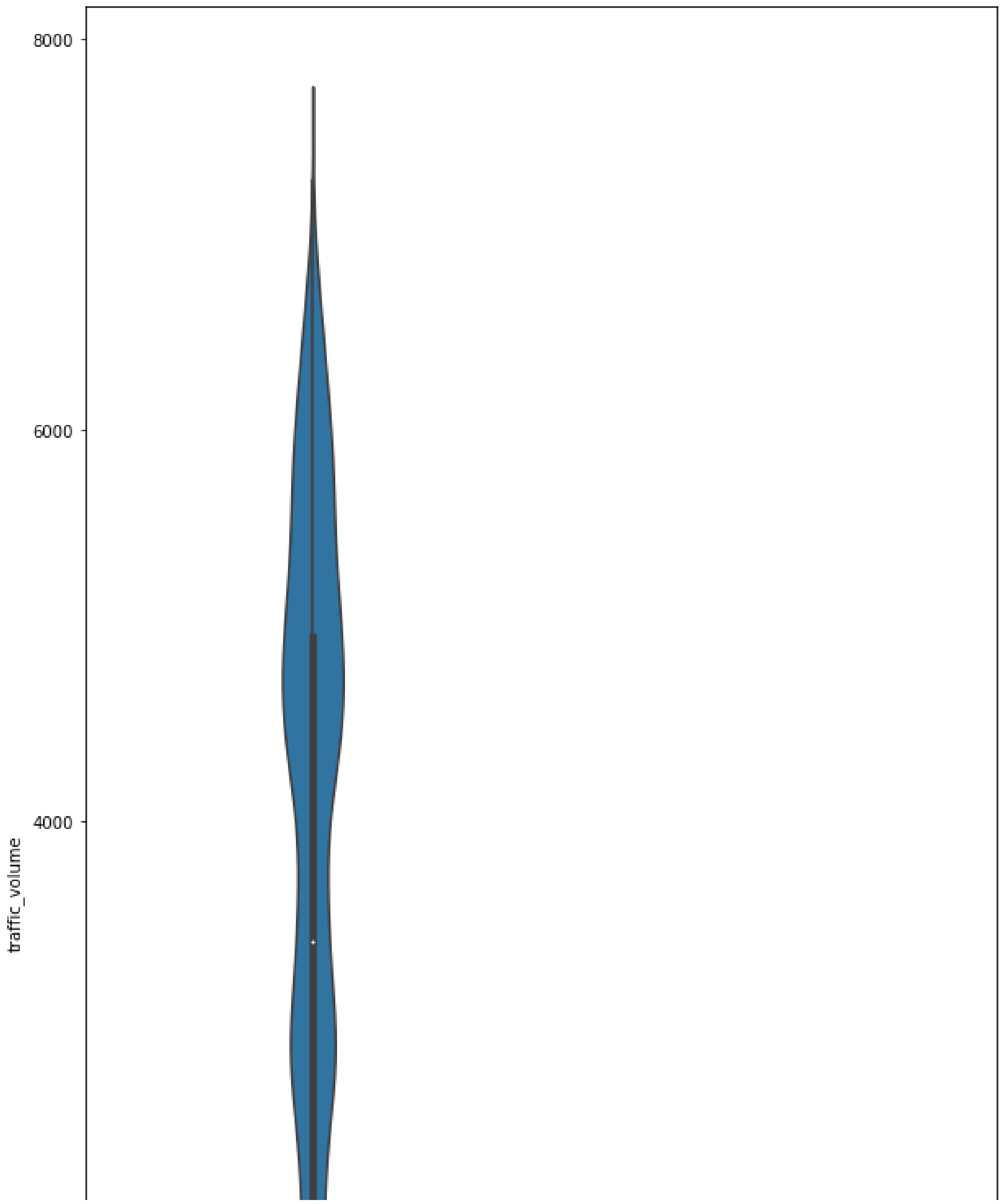


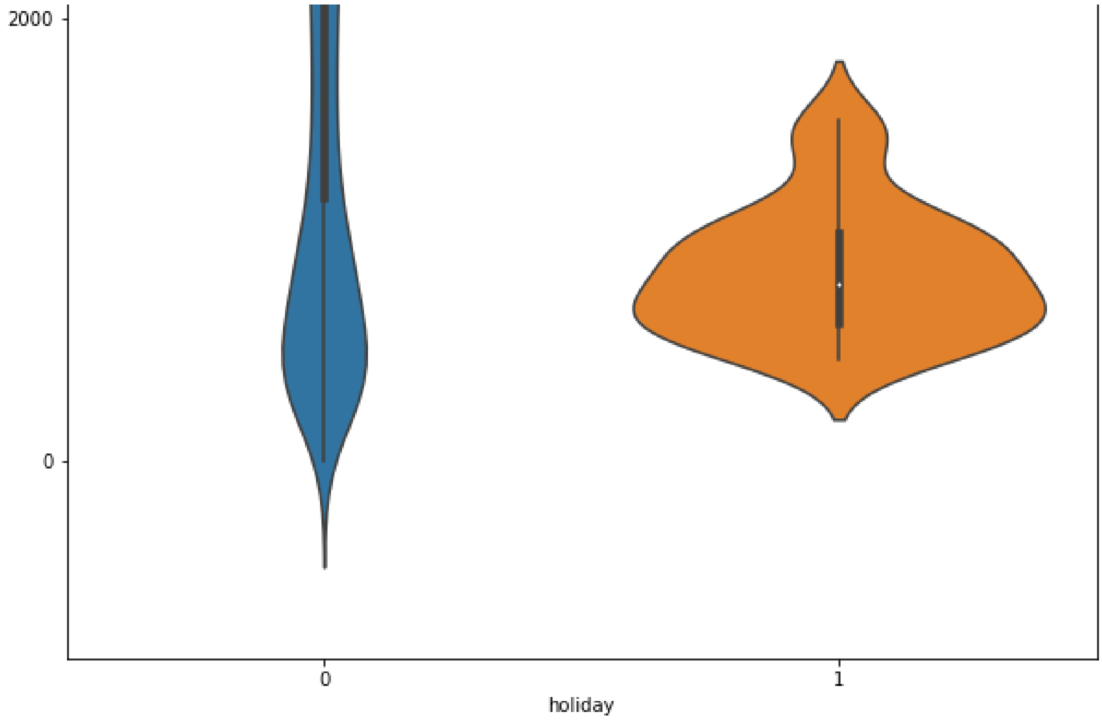
In [22]:

```
plt.figure(figsize=(10,20))
sns.violinplot(x='holiday',y='traffic_volume',data=df)
```

Out[22]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3e85bd10>





During Holidays the traffic Volume is less as comapred to normal days

---

temp vs traffic_volume

In [23]:

```
df['temp'].value_counts()
```

Out[23]:

```
274.150    107
276.793     78
275.150     68
274.080     59
287.150     54
          ...
250.470      1
248.890      1
249.570      1
246.640      1
259.210      1
Name: temp, Length: 5670, dtype: int64
```

In [24]:

```
min(df['temp']) , max(df['temp'])
```

Out[24]:

```
(0.0, 308.24)
```

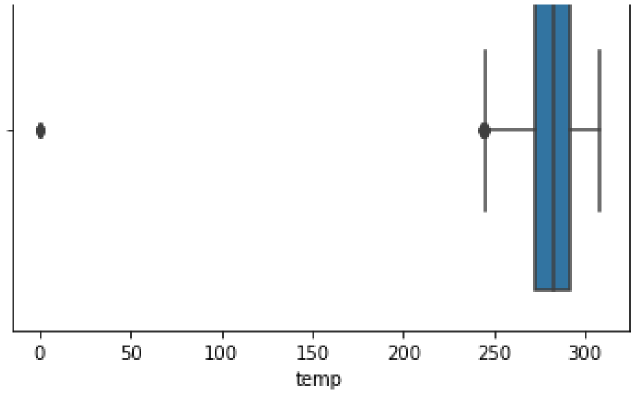
In [25]:

```
# temp column contains outliers

sns.boxplot(df['temp'])
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c2a6510>
```



In [26]:

```
# remove outlier in temp column

df=df[df['temp']>=200]
```

In [27]:

```
df.head()
```

Out[27]:

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

In [28]:

```
df.shape
```

Out[28]:

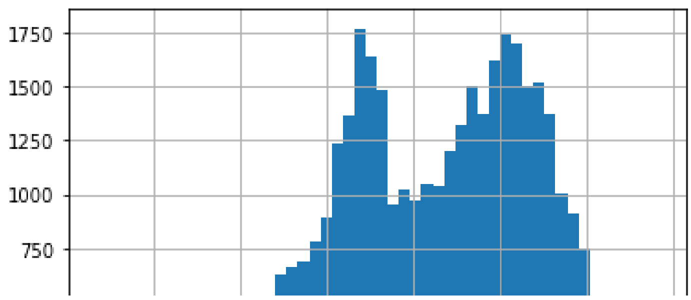
(38553, 15)

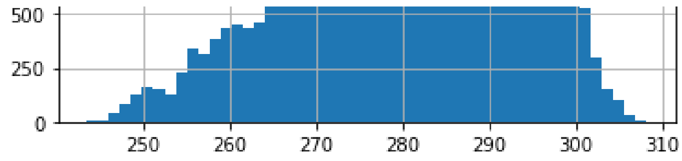
In [29]:

```
df['temp'].hist(bins=50)
```

Out[29]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c693510>



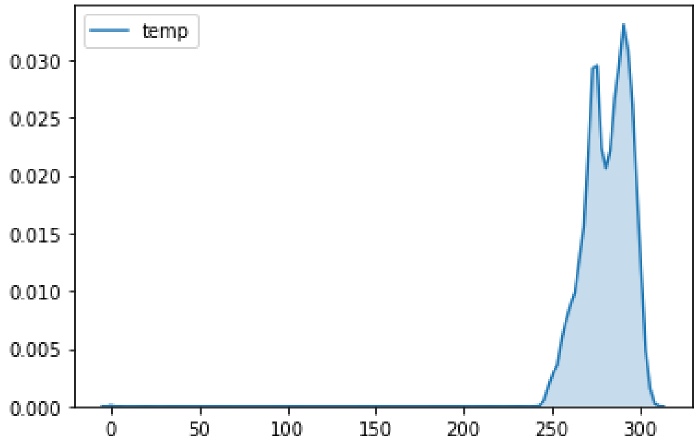


In [30]:

```
sns.kdeplot(data['temp'],shade=True)
```

Out[30]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c56a3d0>

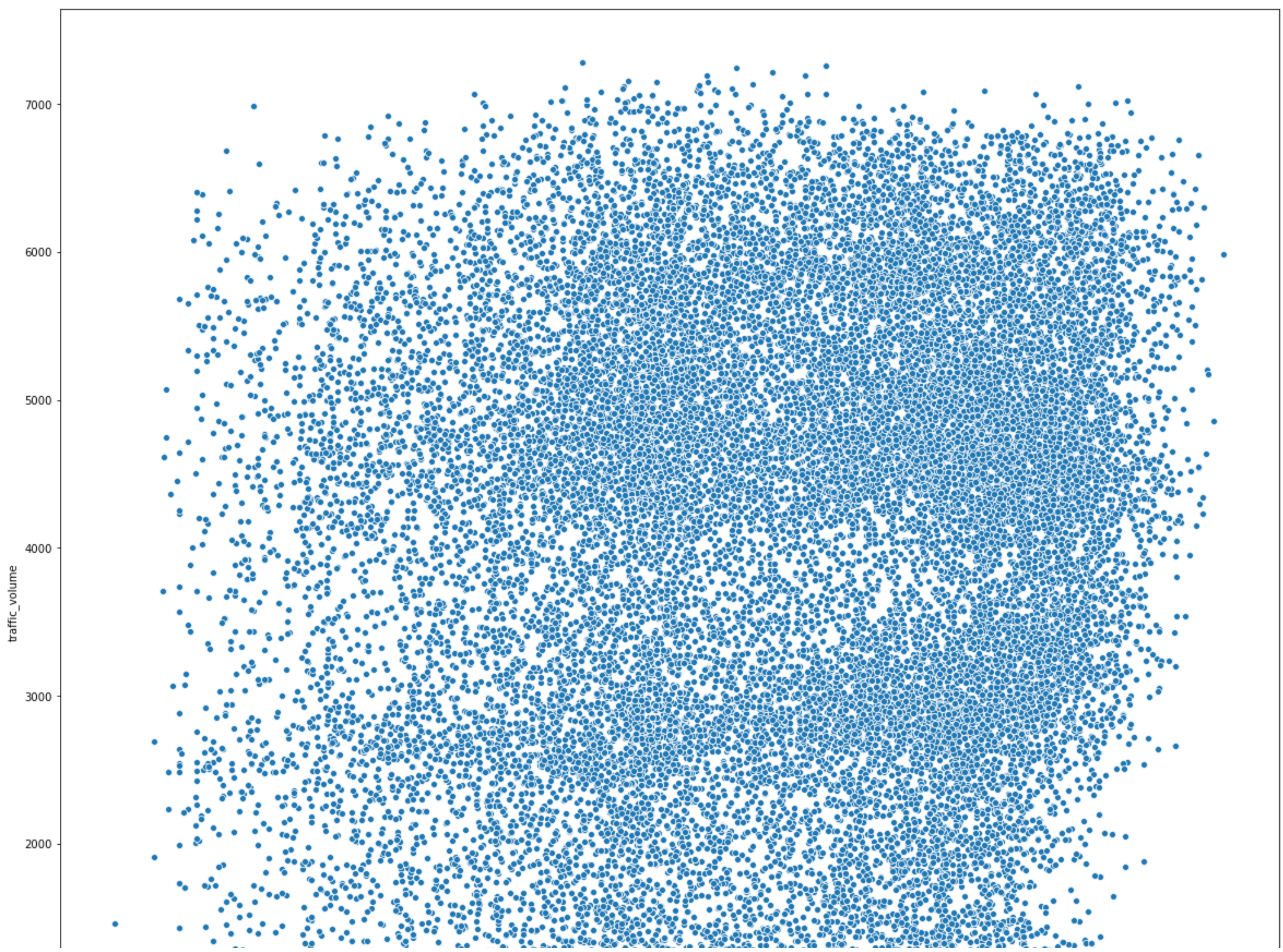


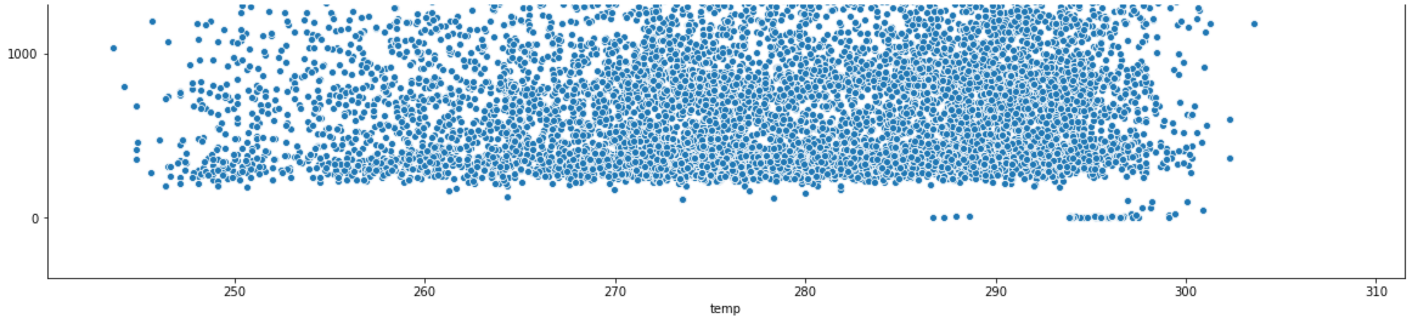
In [31]:

```
plt.figure(figsize=(20,20))
sns.scatterplot(x=df['temp'],y=df['traffic_volume'])
```

Out[31]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c646e10>





When temperature is low Traffic Volume is low as compared to favourable tempearature

---

rain_1h vs traffic_volume

In [32]:

```
df['rain_1h'].value_counts()
```

Out[32]:

```
0.00    35799
0.25      679
0.51      241
0.30      119
0.76       97
        ...
3.12        1
1.53        1
2.34        1
0.92        1
2.13        1
Name: rain_1h, Length: 353, dtype: int64
```

In [33]:

```
df['rain_1h'].sort_values()
```

Out[33]:

```
0           0.00
25716       0.00
25717       0.00
25718       0.00
25719       0.00
          ...
10807      28.70
25779      31.75
7179       44.45
8247       55.63
24872    9831.30
Name: rain_1h, Length: 38553, dtype: float64
```

In [34]:

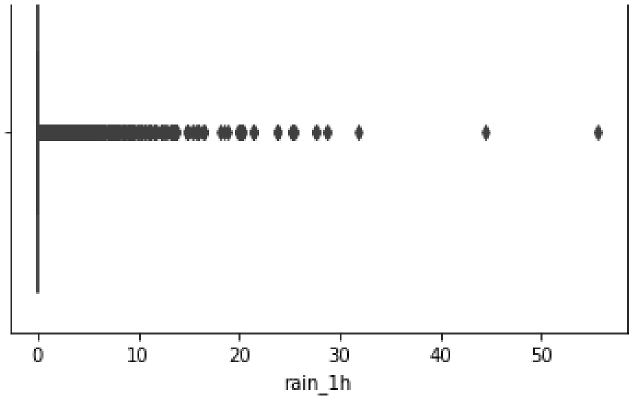
```
df=df[df['rain_1h']<60]
```

In [35]:

```
sns.boxplot(df['rain_1h'])
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c499350>
```

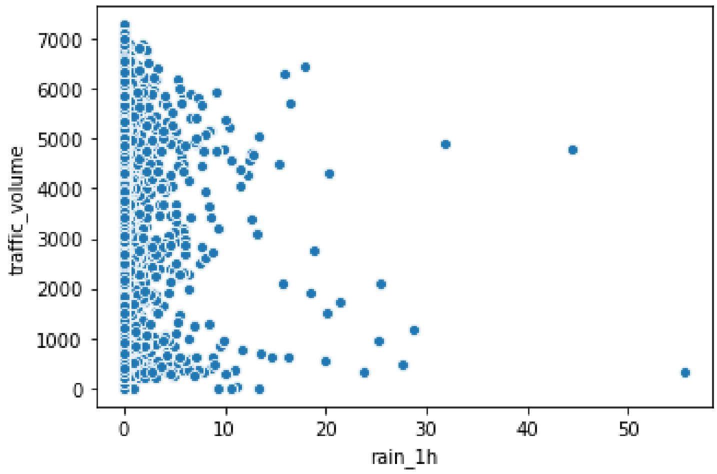


In [36]:

```
sns.scatterplot(df['rain_1h'], df['traffic_volume'])
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c399b90>
```



We observe that rain_1h is insignificant to traffic_volume

---

snow_1h vs traffic_volume

In [37]:

```
df['snow_1h'].value_counts()
```

Out[37]:

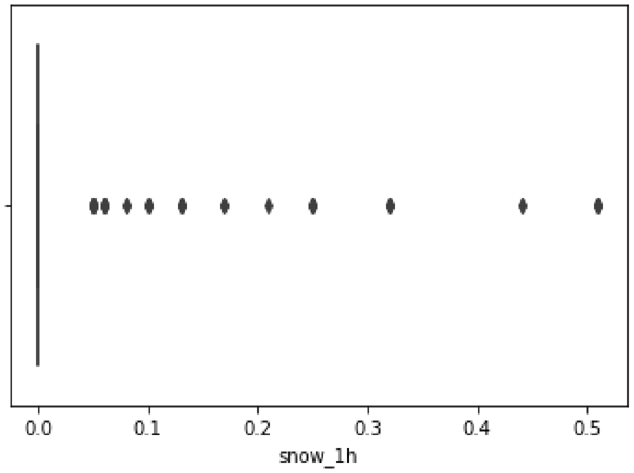
```
0.00    38489
0.05       14
0.06       12
0.10        6
0.13        6
0.51        6
0.25        6
0.32        5
0.17        3
0.08        2
0.44        2
0.21        1
Name: snow_1h, dtype: int64
```

In [38]:

```
sns.boxplot(df['snow_1h'])
```

Out[38]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c38dcd0>

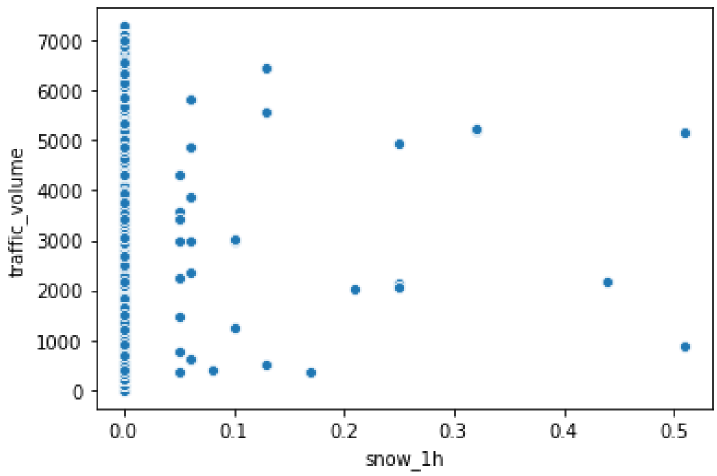


In [39]:

```
sns.scatterplot(df['snow_1h'], df['traffic_volume'])
```

Out[39]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c27ec50>



We observe that snow_1h is insignificant to traffic_volume

---

clouds_all vs traffic_volume

In [40]:

```
df['clouds_all'].value_counts()
```

Out[40]:

```
90      13037
1        9712
75       3795
40       3021
0        1964
64       1557
20       1538
92        772
5         506
12        270
8         265
32        185
24        182
88        181
80        180
```

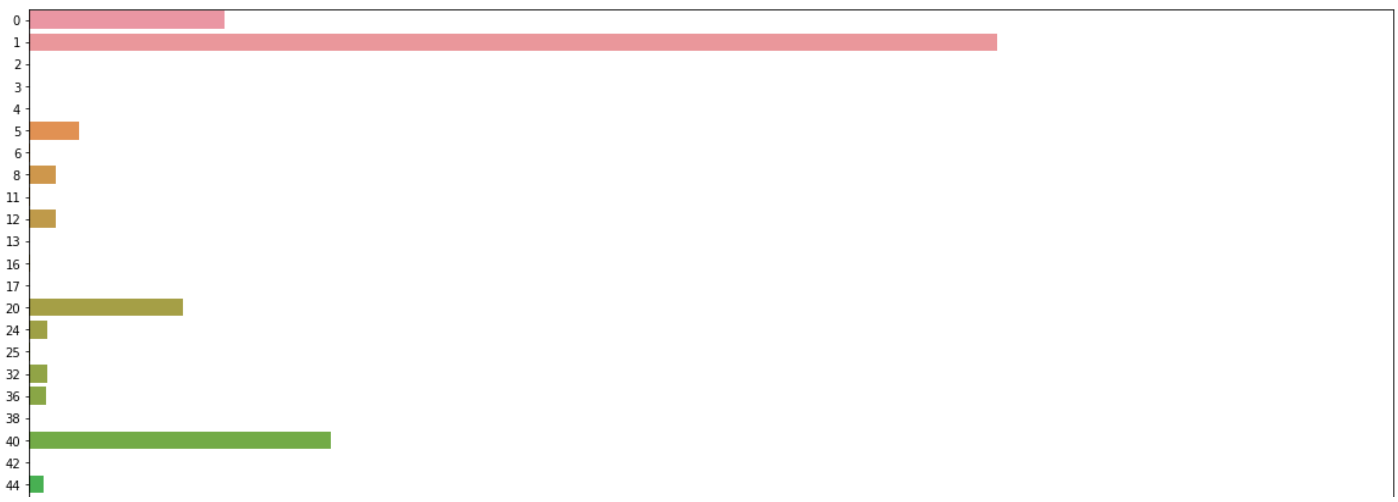
```
48        178
68        175
36        168
56        165
76        153
44        146
100        94
99         44
93         34
98         21
87         18
85         17
59         16
96         14
57         13
89         13
91         12
81         11
46          9
97          8
77          7
6           6
11          5
83          5
16          5
63          5
70          4
58          4
25          4
72          3
3           3
4           3
13          3
50          2
42          2
38          2
2           2
54          2
86          2
62          2
60          2
67          2
17          1
84          1
78          1
Name: clouds_all, dtype: int64
```

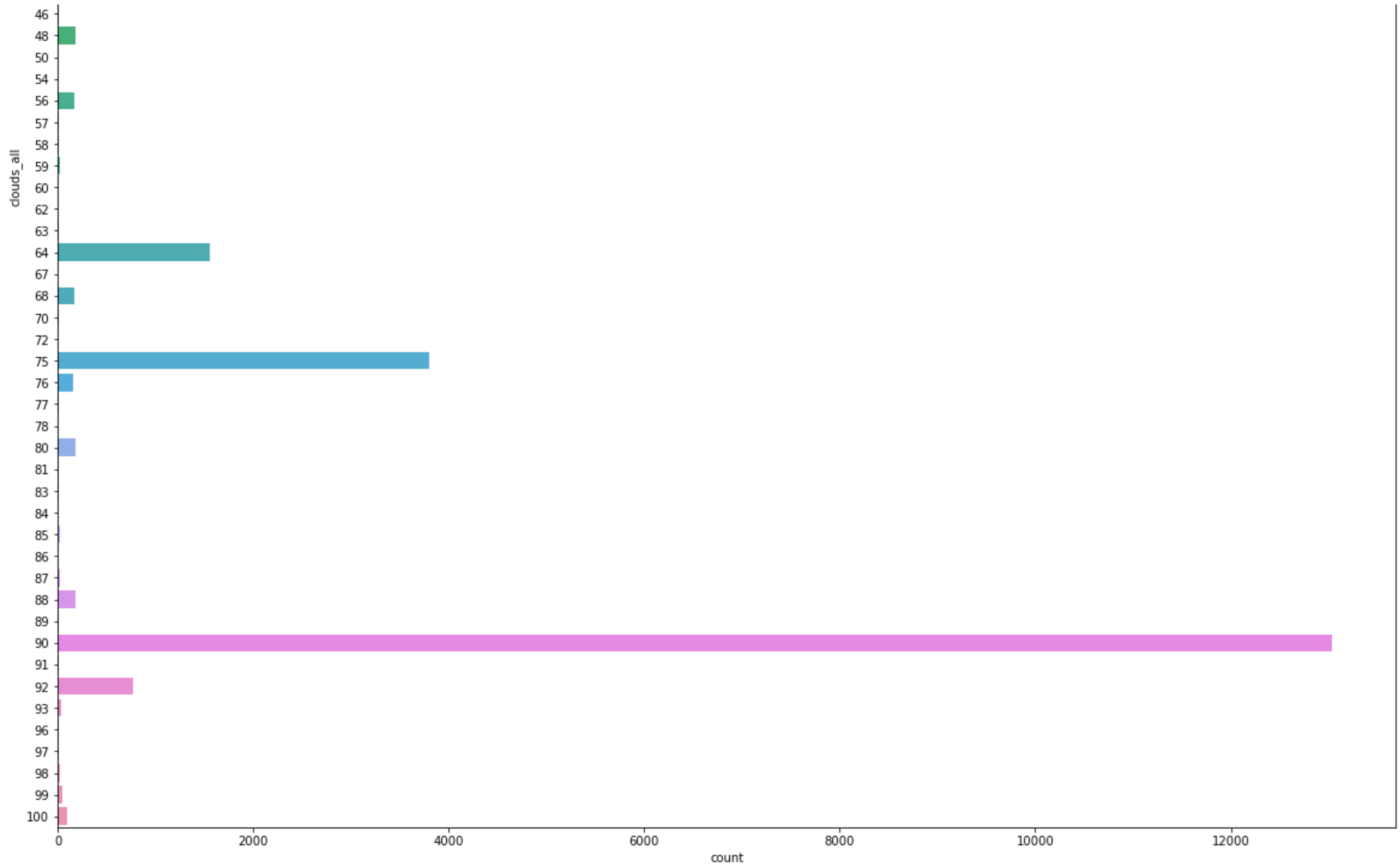
In [41]:

```python
plt.figure(figsize=(20,20))
sns.countplot(y='clouds_all',data=df)
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c28a090>
```



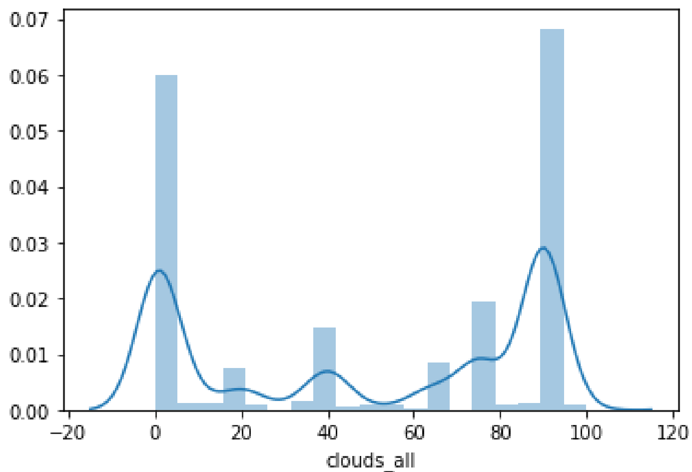


In [42]:

```
sns.distplot(df['clouds_all'])
```

Out[42]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c0bae90>

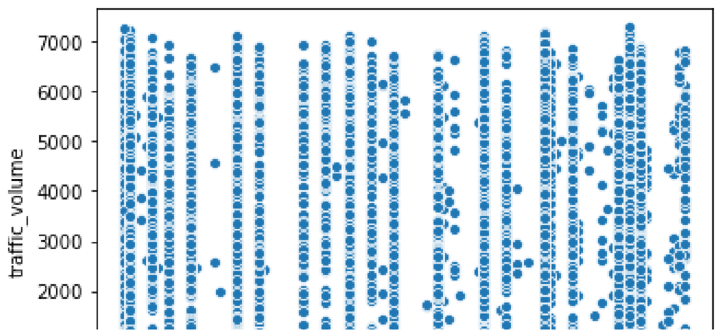


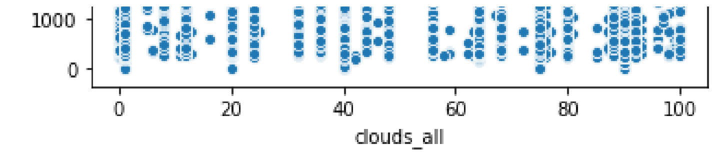
In [43]:

```
sns.scatterplot(df['clouds_all'] , df['traffic_volume'])
```

Out[43]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef374b6d10>



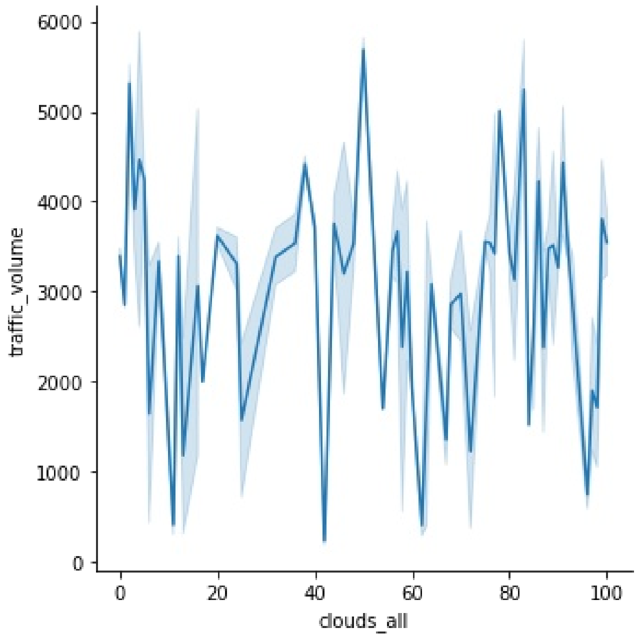


In [44]:

```
sns.relplot(x="clouds_all", y="traffic_volume", data=df, kind='line')
```

Out[44]:

<seaborn.axisgrid.FacetGrid at 0x7fef374b6ed0>



In [45]:

```
df.head(1)
```

Out[45]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	year	month
0	2012-10-02 09:00:00	0	288.28	0.0	0.0	40	Clouds	scattered clouds	5545	2012	10

In [46]:

```
df['weather_main'].value_counts()
```

Out[46]:

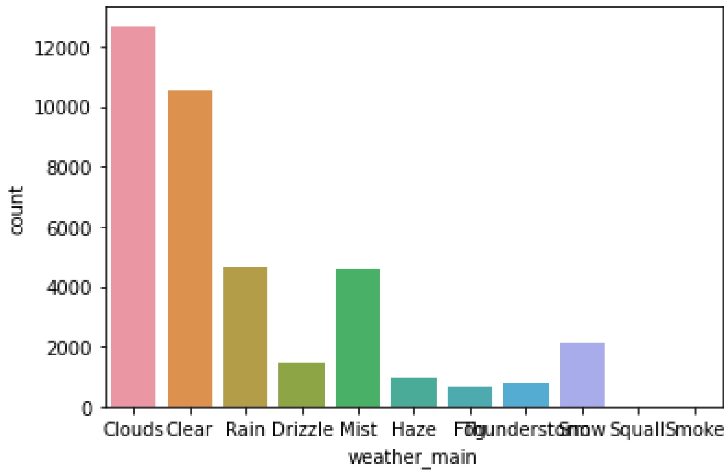
```
Clouds          12680
Clear           10550
Rain             4639
Mist             4611
Snow             2117
Drizzle          1482
Haze              993
Thunderstorm      765
Fog               693
Smoke              18
Squall              4
Name: weather_main, dtype: int64
```

In [47]:

```
sns.countplot(df['weather_main'])
```

Out[47]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3c386150>

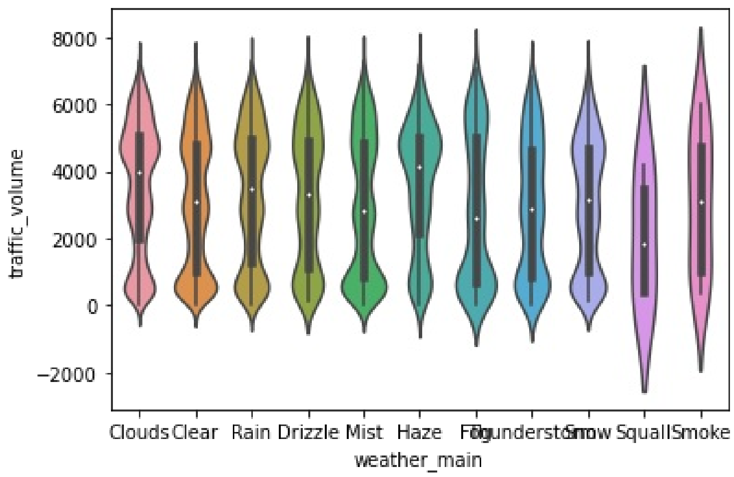


In [48]:

```
sns.violinplot(x='weather_main',y='traffic_volume',data=df)
```

Out[48]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef373f6050>

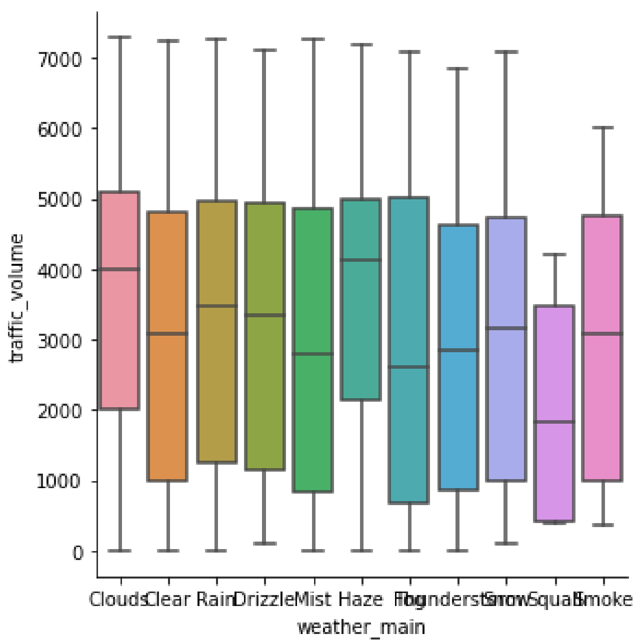


In [49]:

```
sns.catplot(x="weather_main", y="traffic_volume", kind="box", data=df)
```

Out[49]:

<seaborn.axisgrid.FacetGrid at 0x7fef372aa110>

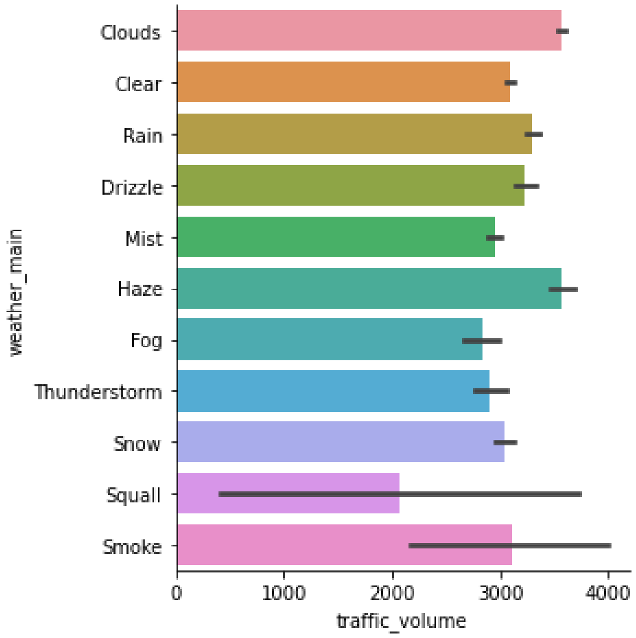


In [50]:

```
sns.catplot(y="weather_main", x="traffic_volume", kind="bar", data=df)
```

Out[50]:

```
<seaborn.axisgrid.FacetGrid at 0x7fef371e6790>
```



In [51]:

```
df['weather_description'].value_counts()
```

Out[51]:

```
sky is clear                           8838
mist                                   4611
overcast clouds                        4302
broken clouds                          3879
light rain                             2795
scattered clouds                       2791
Sky is Clear                           1712
few clouds                             1708
moderate rain                          1333
light snow                             1318
haze                                    993
light intensity drizzle                 871
fog                                     693
heavy snow                              587
drizzle                                 554
proximity thunderstorm                  489
heavy intensity rain                    387
snow                                    199
proximity shower rain                    94
thunderstorm                             88
thunderstorm with heavy rain             56
heavy intensity drizzle                  56
thunderstorm with light rain             41
proximity thunderstorm with rain         38
thunderstorm with rain                   31
smoke                                    18
very heavy rain                          17
proximity thunderstorm with drizzle      12
light intensity shower rain              11
thunderstorm with light drizzle           8
light rain and snow                       6
SQUALLS                                   4
light shower snow                         4
sleet                                     2
thunderstorm with drizzle                 2
freezing rain                             2
```

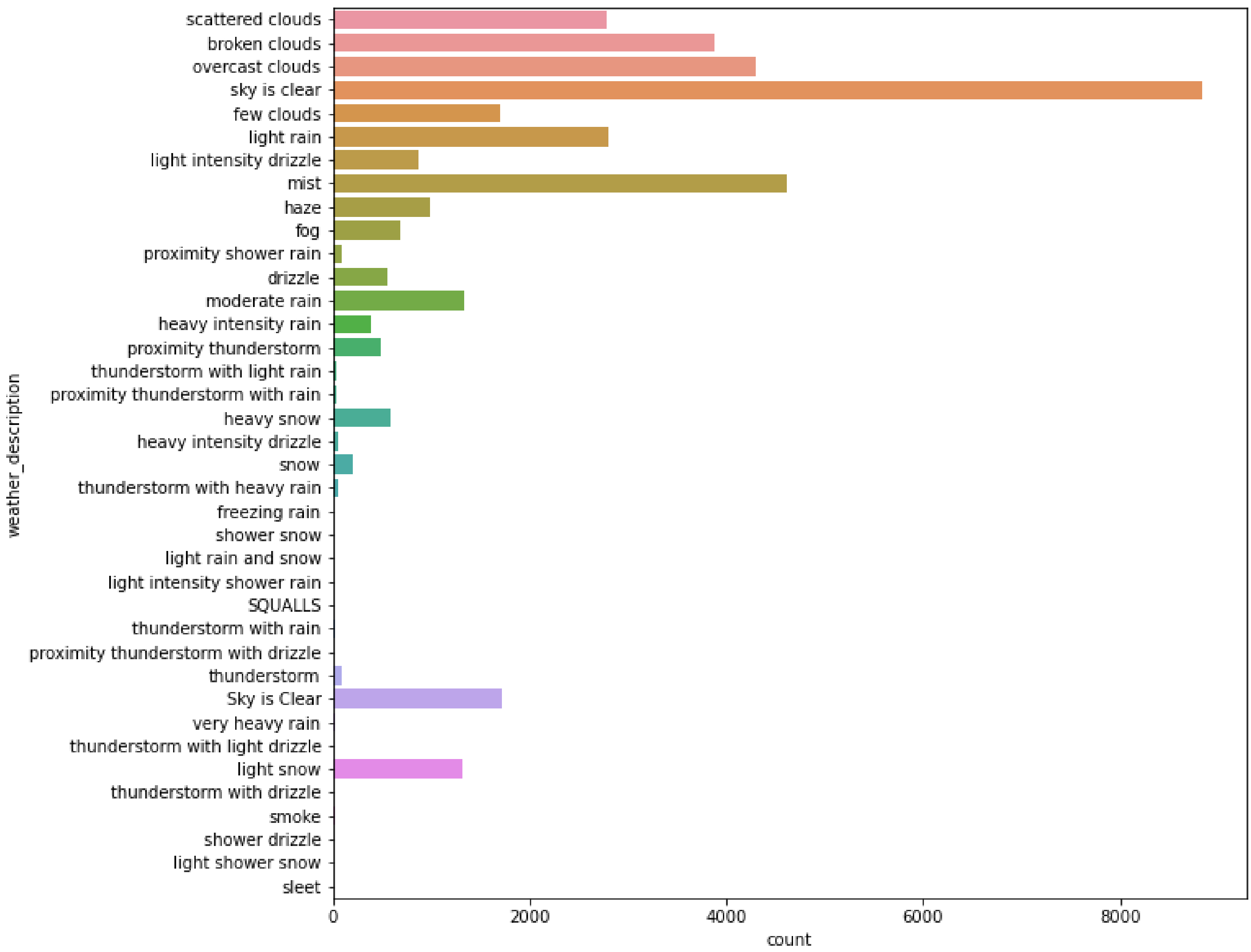
```
shower snow                           1
shower drizzle                        1
Name: weather_description, dtype: int64
```

In [52]:

```
plt.figure(figsize=(10,10))
sns.countplot(y=df['weather_description'])
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef3715ab50>
```

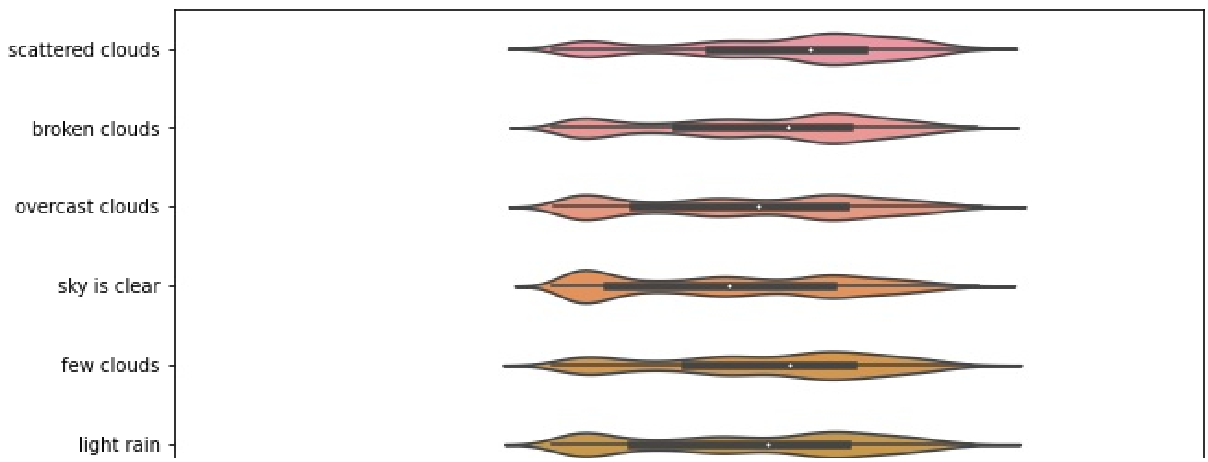


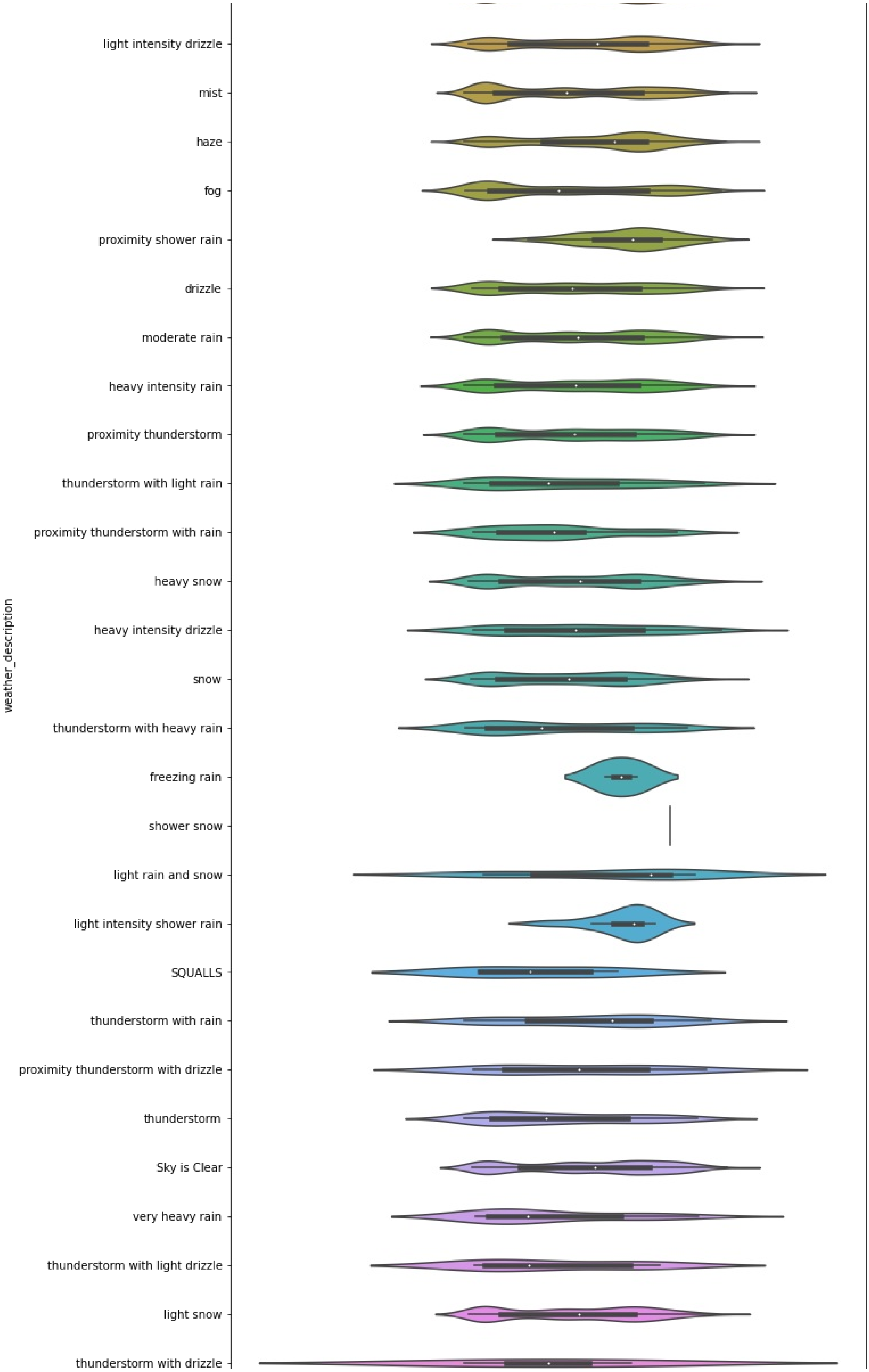
In [53]:

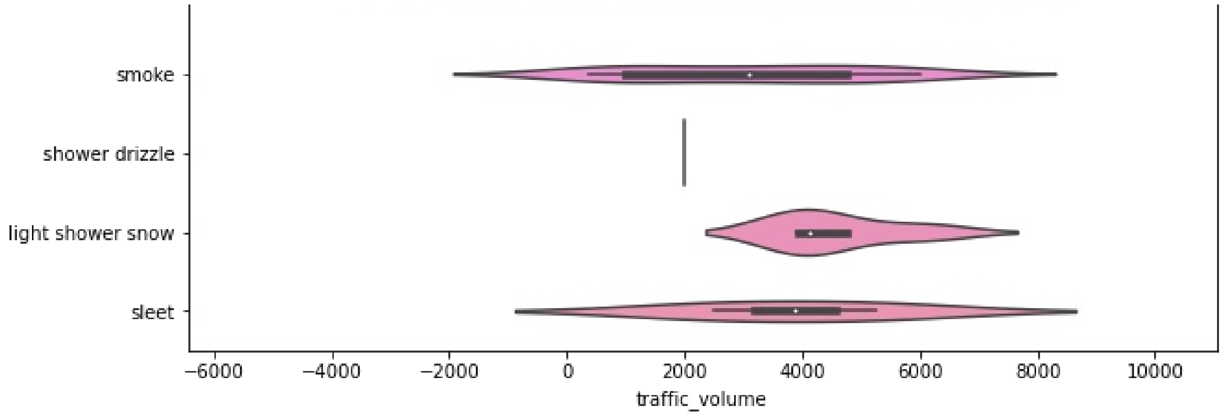
```
plt.figure(figsize=(10,30))
sns.violinplot(y='weather_description',x='traffic_volume',data=df)
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef3703fb90>
```




weather_description
light intensity drizzle
mist
haze
fog
proximity shower rain
drizzle
moderate rain
heavy intensity rain
proximity thunderstorm
thunderstorm with light rain
proximity thunderstorm with rain
heavy snow
heavy intensity drizzle
snow
thunderstorm with heavy rain
freezing rain
shower snow
light rain and snow
light intensity shower rain
SQUALLS
thunderstorm with rain
proximity thunderstorm with drizzle
thunderstorm
Sky is Clear
very heavy rain
thunderstorm with light drizzle
light snow
thunderstorm with drizzle

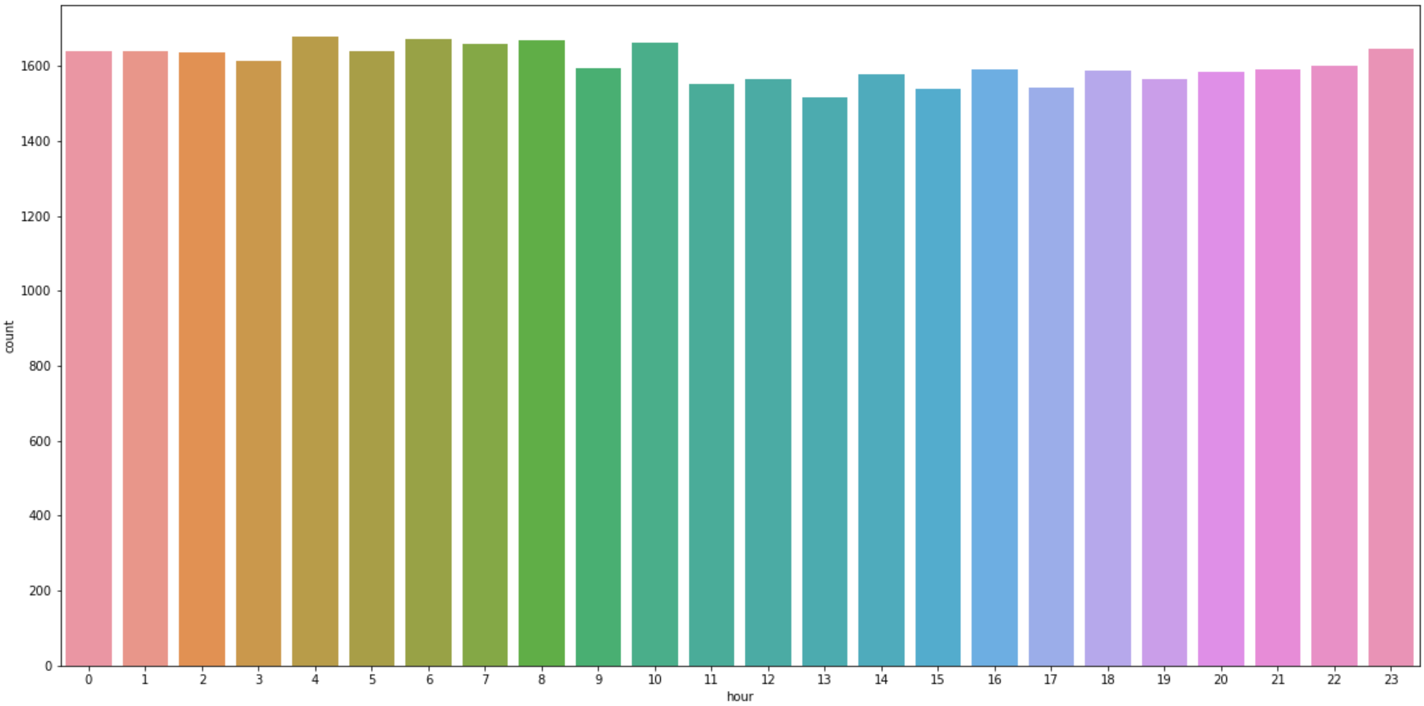


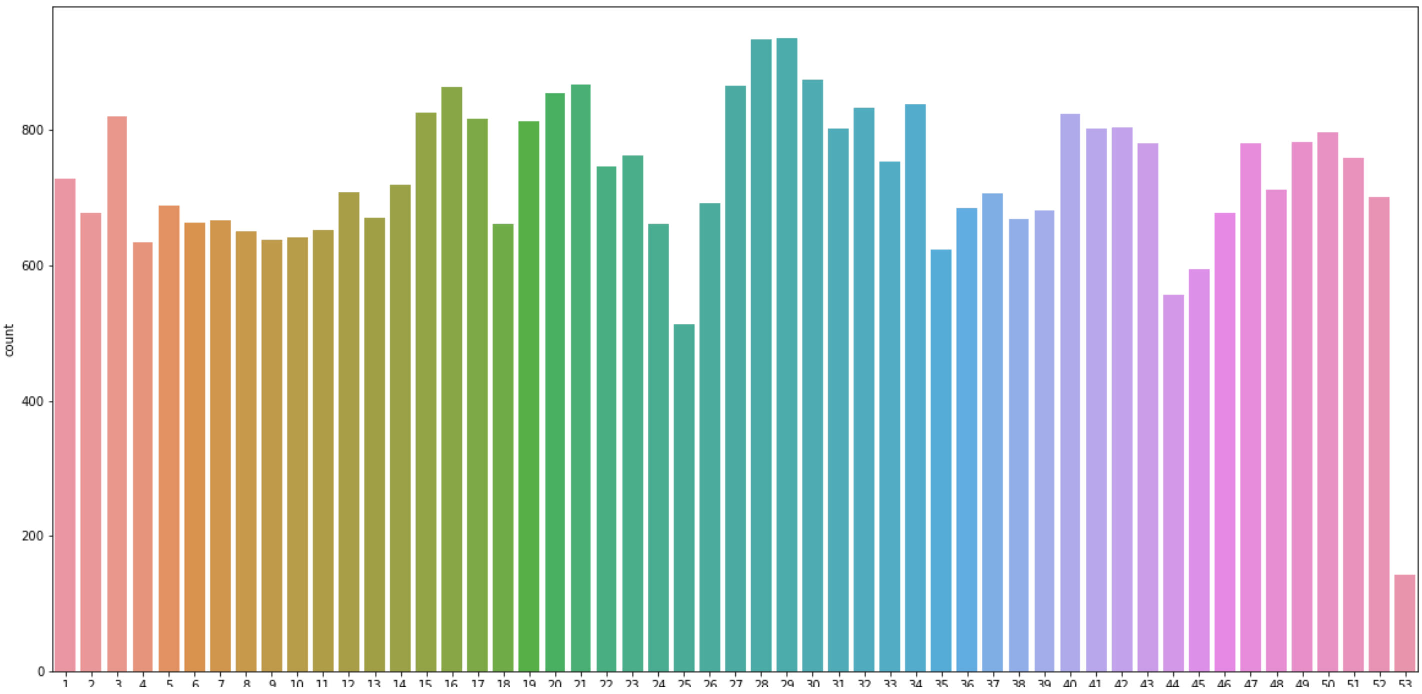
In [54]:

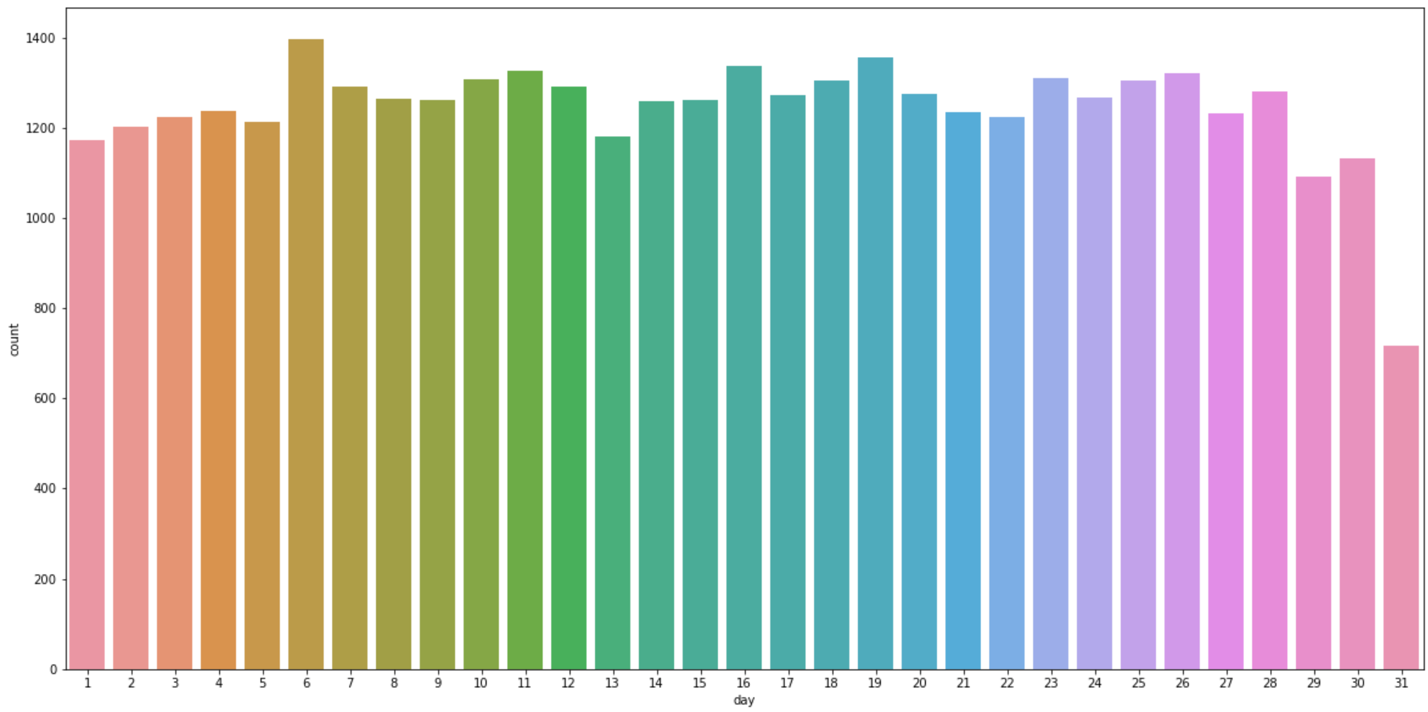
```
cols=['hour','week','day']
```

In [55]:

```
for i in cols:
    plt.figure(figsize=(20,10))
    sns.countplot(df[i])
    plt.show()
```





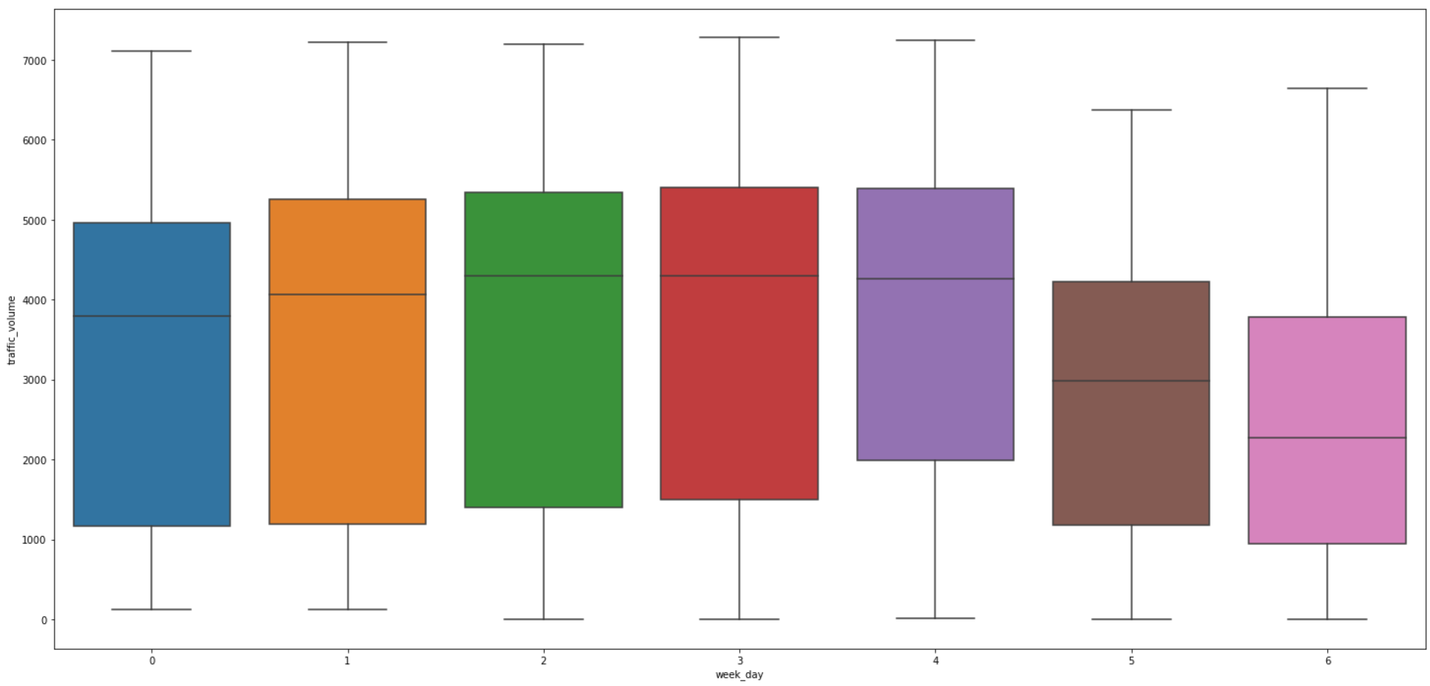


In [56]:

```
plt.figure(figsize=(25,12))
sns.boxplot(df['week_day'],df['traffic_volume'])
```

Out[56]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef37277950>

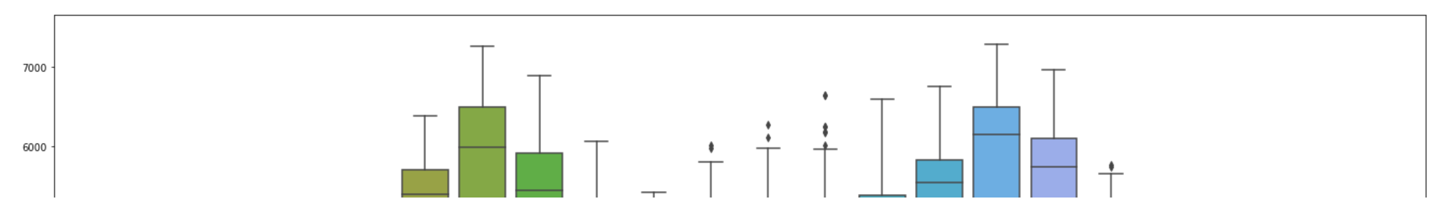


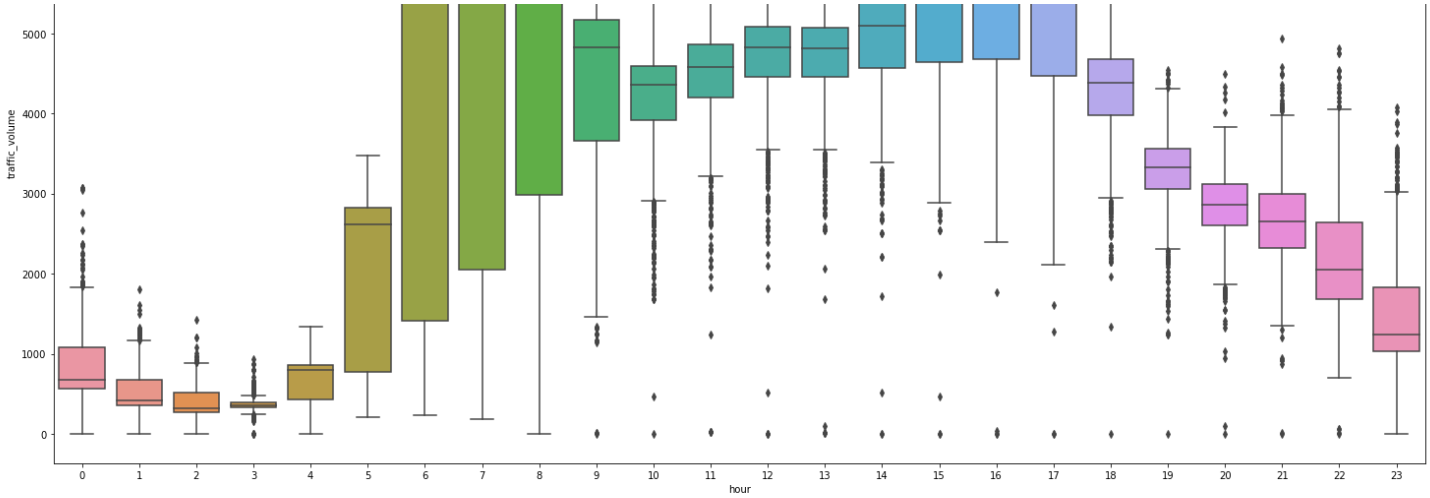
In [57]:

```
plt.figure(figsize=(25,12))
sns.boxplot(df['hour'],df['traffic_volume'])
```

Out[57]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fef34b785d0>





hour and week_day column seems to be highly significant in our data

In [ ]:

DATA CORRELATION

In [58]:

```
df=pd.get_dummies(df, columns=['weather_main','weather_description'])
```

In [59]:

```
df.head()
```

Out[59]:

	date_time	holiday	temp	rain_1h	snow_1h	clouds_all	traffic_volume	year	month	week	...	weather_description_sleet
0	2012-10-02 09:00:00	0	288.28	0.0	0.0	40	5545	2012	10	40	...	0
1	2012-10-02 10:00:00	0	289.36	0.0	0.0	75	4516	2012	10	40	...	0
2	2012-10-02 11:00:00	0	289.58	0.0	0.0	90	4767	2012	10	40	...	0
3	2012-10-02 12:00:00	0	290.13	0.0	0.0	90	5026	2012	10	40	...	0
4	2012-10-02 13:00:00	0	291.14	0.0	0.0	75	4918	2012	10	40	...	0

5 rows × 62 columns

In [60]:

```
df.shape
```

Out[60]:

(38552, 62)

In [61]:

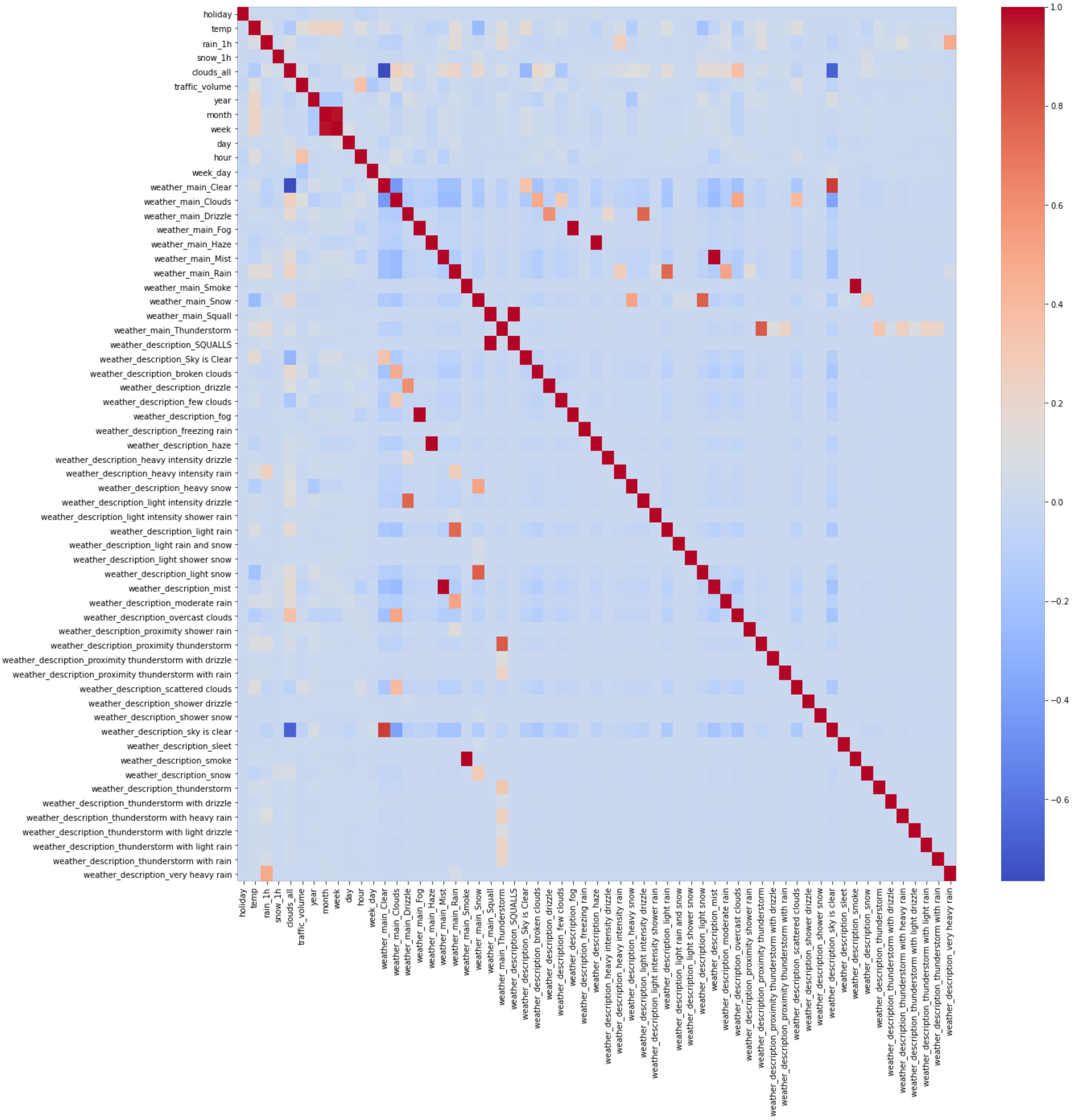
```python
# drop date_time column as it is insignificant
df.drop(['date_time'],axis=1,inplace=True)
```

In [62]:

```python
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), cmap='coolwarm')
```

Out[62]:

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7fef34837c50&gt;
```



In [63]:

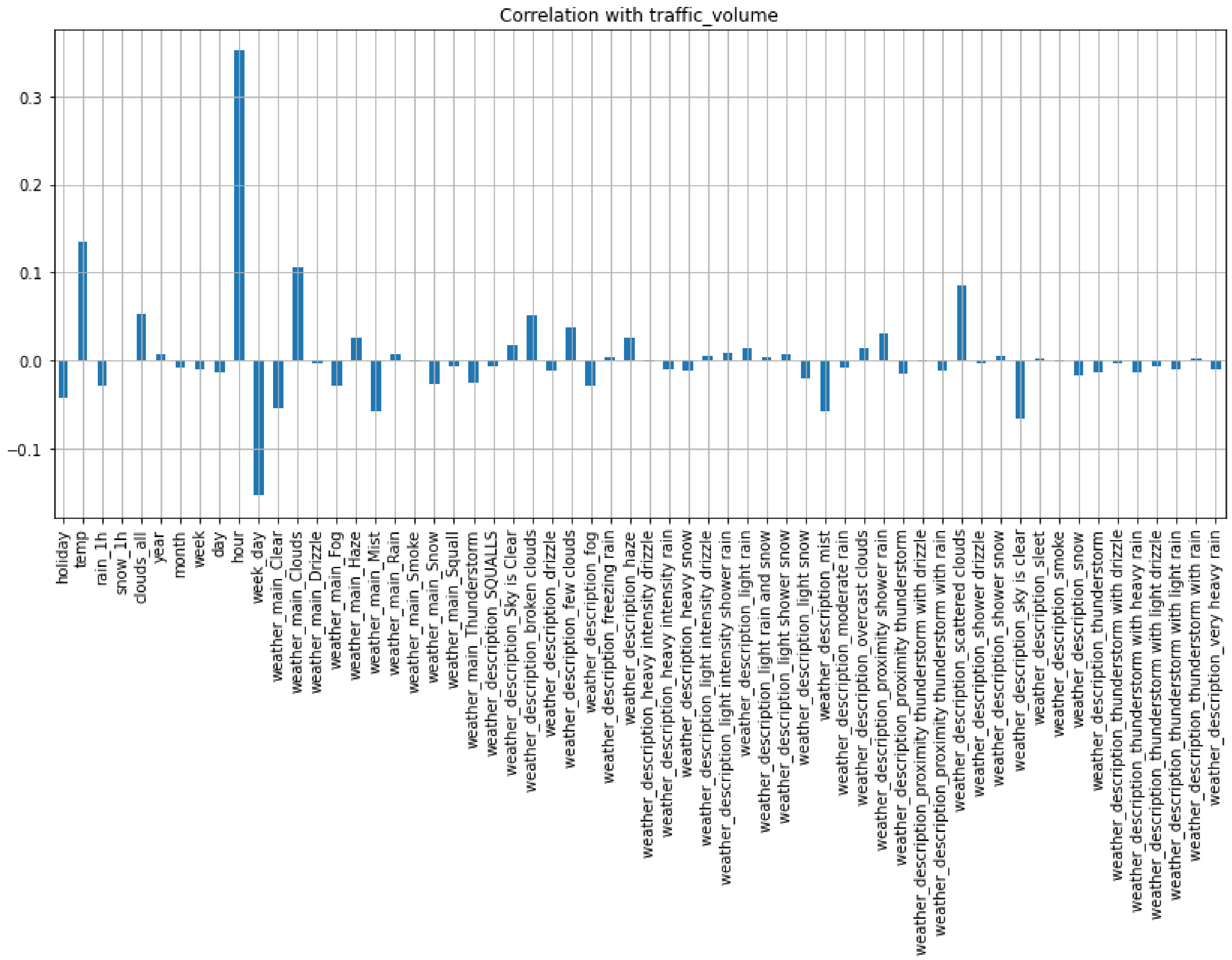
```python
# lets plot correlation plot in another form as heatmap is not clear to conclude feature importance

plt.figure(figsize=(14,6))
df.drop('traffic_volume', axis=1).corrwith(df.traffic_volume).plot(kind = 'bar', grid = True,title = "Correlation with traffic_volume")
```

Out[63]:

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7fef3461bc90&gt;
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe15401be90>
```



In [ ]:

```

```

TRAINING AND TESTING DATA USING SKLEARN

In [64]:

```
# import all required modules

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import GradientBoostingRegressor
from catboost import CatBoostRegressor
```

In [65]:

```
# arranging features of data on basis of its correlation with traffic_volume

cols=df[df.columns[0:]].corr()['traffic_volume'][:-1]
for i in range(len(cols)):
    if cols[i]<0:
        cols[i]*=(-1)
cols.sort_values(ascending=False,inplace=True)
```

LinearRegression

In [66]:

```
l=[]
for i in range(1,70):
    X=df[cols.index[1:i+1]]
    y=df['traffic_volume']
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=40)
    a=LinearRegression()
    a.fit(x_train,y_train)
    b=a.predict(x_test)
    b=mean_squared_error(y_test,b,squared=False)
    l.append(i)
    l.append(b)
```

In [67]:

```
count=0
print('For linear Regression  --- >>>  ')
for i in range(0,len(l),2):
    count+=1
    print('Selecting Top',count,'feature : RMSE ',l[i+1])
```

```
For linear Regression  --- >>>
Selecting Top 1 feature : RMSE  1863.8354702263086
Selecting Top 2 feature : RMSE  1844.5551234874235
Selecting Top 3 feature : RMSE  1834.1434941558252
Selecting Top 4 feature : RMSE  1827.025846939638
Selecting Top 5 feature : RMSE  1825.875219962565
Selecting Top 6 feature : RMSE  1825.0664996155135
Selecting Top 7 feature : RMSE  1824.9647215168668
Selecting Top 8 feature : RMSE  1824.9647215168668
Selecting Top 9 feature : RMSE  1825.1333415672136
Selecting Top 10 feature : RMSE  1823.1615275958723
Selecting Top 11 feature : RMSE  1822.764557732077
Selecting Top 12 feature : RMSE  1822.372654236608
Selecting Top 13 feature : RMSE  1820.4489542521042
Selecting Top 14 feature : RMSE  1820.1365046612295
Selecting Top 15 feature : RMSE  1820.0594470739948
Selecting Top 16 feature : RMSE  1820.0594470739948
Selecting Top 17 feature : RMSE  1819.1296967706014
Selecting Top 18 feature : RMSE  1819.2575216551734
Selecting Top 19 feature : RMSE  1818.5096414347856
Selecting Top 20 feature : RMSE  1818.5096414347856
Selecting Top 21 feature : RMSE  1818.240954465909
Selecting Top 22 feature : RMSE  1818.186721288551
Selecting Top 23 feature : RMSE  1818.186721288551
Selecting Top 24 feature : RMSE  1817.780993261377
Selecting Top 25 feature : RMSE  1817.7522348889916
Selecting Top 26 feature : RMSE  1817.7522348889916
Selecting Top 27 feature : RMSE  1817.2799409289216
Selecting Top 28 feature : RMSE  1817.2932739169974
Selecting Top 29 feature : RMSE  1817.2866311300475
Selecting Top 30 feature : RMSE  1817.0959175302248
Selecting Top 31 feature : RMSE  1816.9994350864897
Selecting Top 32 feature : RMSE  1817.2054844597583
Selecting Top 33 feature : RMSE  1817.071980514645
Selecting Top 34 feature : RMSE  1816.980610998821
Selecting Top 35 feature : RMSE  1816.3262157323782
Selecting Top 36 feature : RMSE  1816.1533908359886
Selecting Top 37 feature : RMSE  1816.353571475113
Selecting Top 38 feature : RMSE  1816.355432450814
Selecting Top 39 feature : RMSE  1816.1342380687424
Selecting Top 40 feature : RMSE  1816.1308742263461
Selecting Top 41 feature : RMSE  1816.1153558407304
Selecting Top 42 feature : RMSE  1816.090622046861
Selecting Top 43 feature : RMSE  1816.1262297825199
Selecting Top 44 feature : RMSE  1816.1091841869659
Selecting Top 45 feature : RMSE  1816.1091841869659
Selecting Top 46 feature : RMSE  1816.51546762545
Selecting Top 47 feature : RMSE  1816.6155237758526
Selecting Top 48 feature : RMSE  1816.6303365024462
```

```
Selecting Top 49 feature : RMSE  1816.6374026276105
Selecting Top 50 feature : RMSE  1816.622321442561
Selecting Top 51 feature : RMSE  1816.6907347818983
Selecting Top 52 feature : RMSE  1816.8498062033436
Selecting Top 53 feature : RMSE  1816.8165725098113
Selecting Top 54 feature : RMSE  1816.8165725098113
Selecting Top 55 feature : RMSE  1816.8165725098113
Selecting Top 56 feature : RMSE  1816.8165725098113
Selecting Top 57 feature : RMSE  1816.816572509811
Selecting Top 58 feature : RMSE  1816.8165725098113
Selecting Top 59 feature : RMSE  1816.8150639322241
Selecting Top 60 feature : RMSE  1816.8150639322241
Selecting Top 61 feature : RMSE  1816.8150639322241
Selecting Top 62 feature : RMSE  1816.8150639322241
Selecting Top 63 feature : RMSE  1816.8150639322241
Selecting Top 64 feature : RMSE  1816.8150639322241
Selecting Top 65 feature : RMSE  1816.8150639322241
Selecting Top 66 feature : RMSE  1816.8150639322241
Selecting Top 67 feature : RMSE  1816.8150639322241
Selecting Top 68 feature : RMSE  1816.8150639322241
Selecting Top 69 feature : RMSE  1816.8150639322241
```

KNeighborsRegressor

In [68]:

```
l=[]
for i in range(1,70):
    X=df[cols.index[1:i+1]]
    y=df['traffic_volume']
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=40)
    a=KNeighborsRegressor()
    a.fit(x_train,y_train)
    b=a.predict(x_test)
    b=mean_squared_error(y_test,b,squared=False)
    l.append(i)
    l.append(b)
```

In [69]:

```
count=0
print('For KNeighborsRegressor  --- >>>  ')
for i in range(0,len(l),2):
    count+=1
    print('Selecting Top',count,'feature : RMSE ',l[i+1])
```

```
For KNeighborsRegressor  --- >>>
Selecting Top 1 feature : RMSE  961.5581163166052
Selecting Top 2 feature : RMSE  535.8126537823771
Selecting Top 3 feature : RMSE  506.87745645319353
Selecting Top 4 feature : RMSE  515.7258862664064
Selecting Top 5 feature : RMSE  514.297522109108
Selecting Top 6 feature : RMSE  525.4181582282628
Selecting Top 7 feature : RMSE  530.7776820212129
Selecting Top 8 feature : RMSE  546.1450847686089
Selecting Top 9 feature : RMSE  550.1158653259888
Selecting Top 10 feature : RMSE  616.5547009351515
Selecting Top 11 feature : RMSE  617.4432958223443
Selecting Top 12 feature : RMSE  617.4074550661514
Selecting Top 13 feature : RMSE  616.9187592570665
Selecting Top 14 feature : RMSE  616.5214630764492
Selecting Top 15 feature : RMSE  616.1195210761219
Selecting Top 16 feature : RMSE  619.2910425965163
Selecting Top 17 feature : RMSE  623.6834794981223
Selecting Top 18 feature : RMSE  626.3303972509677
Selecting Top 19 feature : RMSE  625.65850953723
Selecting Top 20 feature : RMSE  626.6947938242398
Selecting Top 21 feature : RMSE  626.0211730040845
Selecting Top 22 feature : RMSE  626.7634397728308
Selecting Top 23 feature : RMSE  627.8946410583104
```

```
Selecting Top 24 feature : RMSE  627.9117331477111
Selecting Top 25 feature : RMSE  628.1843817472264
Selecting Top 26 feature : RMSE  631.8077485308472
Selecting Top 27 feature : RMSE  633.0061084718919
Selecting Top 28 feature : RMSE  633.2377553245799
Selecting Top 29 feature : RMSE  633.3444188906665
Selecting Top 30 feature : RMSE  748.6445276946206
Selecting Top 31 feature : RMSE  748.7538399891228
Selecting Top 32 feature : RMSE  749.551466186064
Selecting Top 33 feature : RMSE  749.5587854145365
Selecting Top 34 feature : RMSE  748.6960857837117
Selecting Top 35 feature : RMSE  895.2532218169606
Selecting Top 36 feature : RMSE  895.2339494170343
Selecting Top 37 feature : RMSE  895.237365599281
Selecting Top 38 feature : RMSE  897.3425044177555
Selecting Top 39 feature : RMSE  897.1042145335662
Selecting Top 40 feature : RMSE  900.1157911939647
Selecting Top 41 feature : RMSE  915.3084716179219
Selecting Top 42 feature : RMSE  915.3084716179219
Selecting Top 43 feature : RMSE  915.3084716179219
Selecting Top 44 feature : RMSE  915.3084716179219
Selecting Top 45 feature : RMSE  915.3713690949908
Selecting Top 46 feature : RMSE  915.5667088818035
Selecting Top 47 feature : RMSE  915.5809285710716
Selecting Top 48 feature : RMSE  915.5809285710716
Selecting Top 49 feature : RMSE  915.5809285710716
Selecting Top 50 feature : RMSE  915.5809285710716
Selecting Top 51 feature : RMSE  915.5809285710716
Selecting Top 52 feature : RMSE  917.2076646489493
Selecting Top 53 feature : RMSE  917.0760885487003
Selecting Top 54 feature : RMSE  917.0760885487003
Selecting Top 55 feature : RMSE  917.000200266221
Selecting Top 56 feature : RMSE  917.000200266221
Selecting Top 57 feature : RMSE  917.1644757146362
Selecting Top 58 feature : RMSE  917.1448653842181
Selecting Top 59 feature : RMSE  917.1448653842181
Selecting Top 60 feature : RMSE  917.1448653842181
Selecting Top 61 feature : RMSE  917.1448653842181
Selecting Top 62 feature : RMSE  917.1448653842181
Selecting Top 63 feature : RMSE  917.1448653842181
Selecting Top 64 feature : RMSE  917.1448653842181
Selecting Top 65 feature : RMSE  917.1448653842181
Selecting Top 66 feature : RMSE  917.1448653842181
Selecting Top 67 feature : RMSE  917.1448653842181
Selecting Top 68 feature : RMSE  917.1448653842181
Selecting Top 69 feature : RMSE  917.1448653842181
```

RandomForestRegressor

In [ ]:

```
l=[]
for i in range(1,70):
    X=df[cols.index[1:i+1]]
    y=df['traffic_volume']
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=40)
    a=RandomForestRegressor()
    a.fit(x_train,y_train)
    b=a.predict(x_test)
    b=mean_squared_error(y_test,b,squared=False)
    l.append(i)
    l.append(b)
```

In [ ]:

```
count=0
print('For RandomForestRegressor  --- >>>  ')
for i in range(0,len(l),2):
    count+=1
```

```
    print('Selecting Top',count,'feature : RMSE ',l[i+1])
```

GradientBoostingRegressor

In [ ]:

```
l=[]
for i in range(1,70):
    X=df[cols.index[1:i+1]]
    y=df['traffic_volume']
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=40)
    a=GradientBoostingRegressor()
    a.fit(x_train,y_train)
    b=a.predict(x_test)
    b=mean_squared_error(y_test,b,squared=False)
    l.append(i)
    l.append(b)
```

In [ ]:

```
count=0
print('For GradientBoostingRegressor  --- >>>  ')
for i in range(0,len(l),2):
    count+=1
    print('Selecting Top',count,'feature : RMSE ',l[i+1])
```

CatBoostRegressor

In [ ]:

```
l=[]
for i in range(1,60):
    X=df[cols.index[1:i+1]]
    y=df['traffic_volume']
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=40)
    a=CatBoostRegressor(iterations=300)
    a.fit(x_train,y_train)
    b=a.predict(x_test)
    b=mean_squared_error(y_test,b,squared=False)
    l.append(i)
    l.append(b)
```

In [ ]:

```
count=0
print('For CatBoostRegressor  --- >>>  ')
for i in range(0,len(l),2):
    count+=1
    print('Selecting Top',count,'feature : RMSE ',l[i+1])
```

In [ ]:

```

```

In [ ]:

```
cols.index[1:42]
```

We observe that when we select top 42 feature of data the CatBoostRegressor model is giving good rmse

In [ ]:

```
cols_req=cols.index[1:42]
```

In [ ]:

```
if 1:
    X=df[cols.index[1:42]]
    y=df['traffic_volume']
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=40)
    model=CatBoostRegressor(iterations=5000)
    model.fit(x_train,y_train)
    b=model.predict(x_test)
    for i in range(len(b)):
        b[i]=int(b[i])
        if b[i]<=0:
            b[i]=0

    c=mean_squared_error(y_test,b,squared=False)
    print(c)
```

In [ ]:

```

```

Prediction of Test data

In [ ]:

```

```

In [ ]:

```
test=pd.read_csv('TEST.csv')
```

In [ ]:

```
tes=test.copy()
```

In [ ]:

```
tes.head()
```

In [ ]:

```
tes['date_time']=pd.to_datetime(test['date_time'])
```

In [ ]:

```
tes['year'] = tes['date_time'].dt.year
tes['month'] = tes['date_time'].dt.month
tes['week_day'] = tes['date_time'].dt.week
tes['day'] = tes['date_time'].dt.day
tes['hour'] = tes['date_time'].dt.hour
tes['week'] = tes['date_time'].dt.dayofweek
```

In [ ]:

```
tes.head(2)
```

In [ ]:

```
tes.isna().sum()
```

In [ ]:

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

In [ ]:

```
tes=pd.get_dummies(tes, columns=['weather_main','weather_description'])
```

In [ ]:

```
tes.head()
```

In [ ]:

```
tes['holiday']=tes['holiday'].apply(lambda x: 0 if x=='None' else 1)
```

In [ ]:

```
tes=tes[cols_req]
```

In [ ]:

```
tes.head()
```

In [ ]:

```
pred=model.predict(tes)
```

In [ ]:

```
min(pred)
```

In [ ]:

```
# convert -ve predicted values to 0
for i in range(len(pred)):
        pred[i]=int(pred[i])
        if pred[i]<=0:
            pred[i]=0
```

In [ ]:

```
pred
```

In [ ]:

```
index=[i for i in range(9641)]
```

In [ ]:

```
sub=pd.DataFrame({'Index':index, 'Value':pred})
```

In [ ]:

```
sub
```

In [ ]:

```
# save and download csv file
from IPython.display import HTML
import base64
```

In [ ]:

```
def create_download_link( df, title = "Download CSV file", filename = "data.csv"):
    csv = df.to_csv(index=False)
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="_blank
">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)
```

In [ ]:

```
create_download_link(sub)
```

In [ ]:

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```