Questions:

1. ***Speed Limit Visualization***

   We divided speed from 20 to 110 km/hr with a step size of 10 km/hr, and each step is assigned with different color scheme.

```
style20 = {'fillColor': '#FAF9DF', 'color': '#FAF9DF'}
speed20_df = speed_gdf[(speed_gdf['SPEED']>=20) & (speed_gdf['SPEED']<35)]
folium.GeoJson(speed20_df['multiline'], style_function=lambda x:style20).add_to(smap)
```

   The dataFrame with different speed range will then be read by folium GeoJson and added to the Traffic map. After the speed limit was added to smap, template is used to create the legend of the speed and added to the smap.

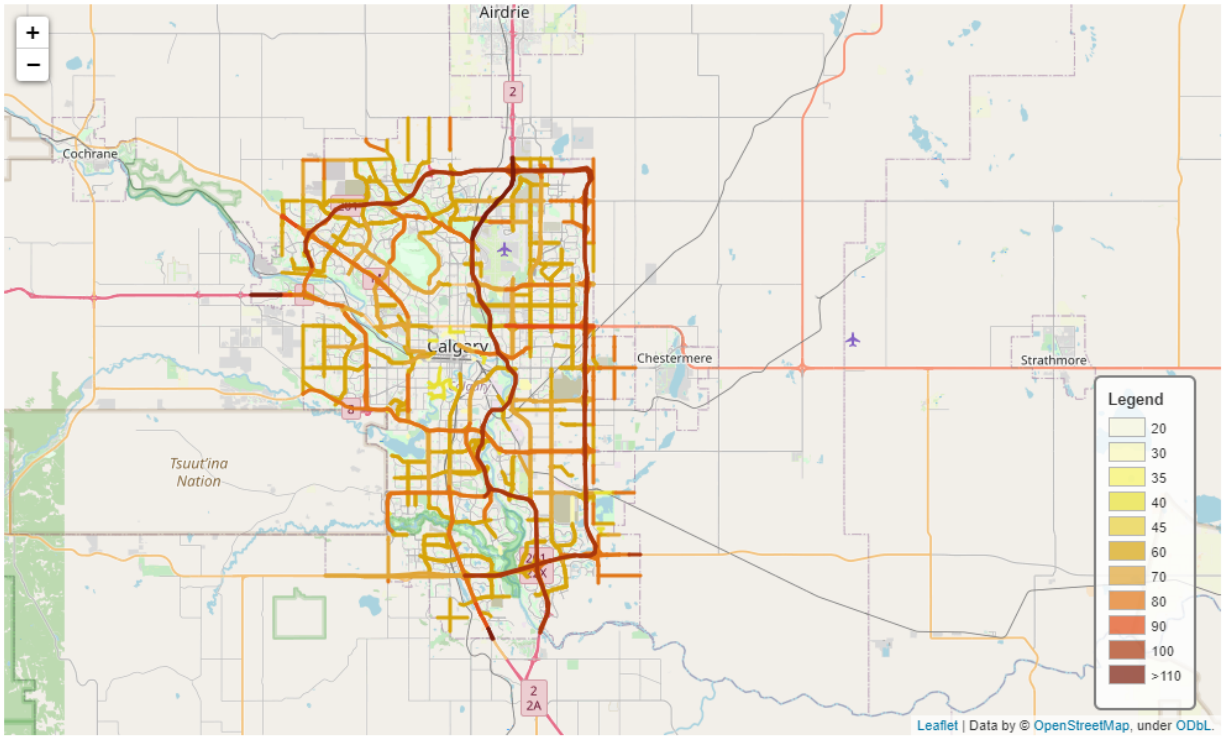
```
macro = MacroElement()
macro._template = Template(template)

smap.get_root().add_child(macro)

smap
```

```
smap.get_root().add_child(macro)

smap
```



2. ***Volume Heatmap Visualization***

```
def generate_traffic_volume_list(volume_df):
    volume_list=[]
    for i in range(volume_df.shape[0]):
        volume_coord=volume_df["multilinestring"].values[i]
        coor=get_coordinates(volume_coord)
        for j in range(len(coor)):
            coor[j].append(volume_df["VOLUME"].values[i])
            volume_list.append(coor[j])
    return volume_list
```

```
volume_df = pd.read_csv('Traffic_Volumes_for_2018.csv',
                        usecols=['YEAR', 'VOLUME', 'multilinestring'])
volume_df = volume_df[volume_df['YEAR']==2018].drop(columns=['YEAR'])

volume_list=generate_traffic_volume_list(volume_df)

traffic_volume_df=pd.DataFrame(volume_list,columns=['latitude','longitude','volume'])
traffic_volume_df.head(10)
```

the traffic volume csv contains columns such as Volume and multilinestring, meaning we have volume data for each corresponding road segment, however, in order to plot heatmap, we need to get volume data for each latitude, longitude coordinate:

- generate_traffic_volume_list is the function to create a volume_list with [**latitude, longitude, volume] data in each data set**
- traffic_volume_df used to convert volume_list into dataFrame with following output:

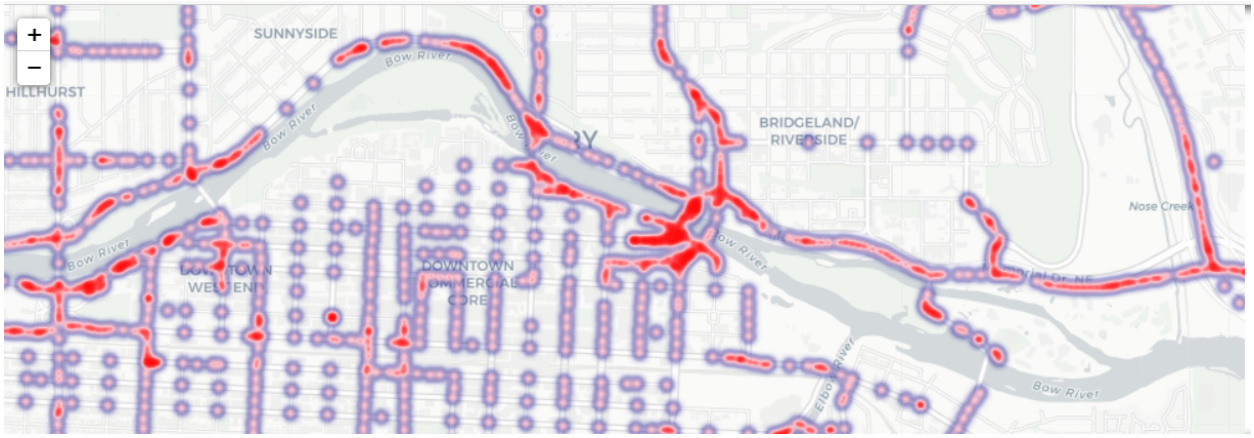
	latitude	longitude	volume
0	51.048319	-114.060367	22000
1	51.048250	-114.057908	22000
2	50.968634	-114.068768	5000
3	50.968648	-114.071465	5000
4	51.053232	-114.033722	5000
5	51.050882	-114.033709	5000
6	51.143370	-114.013188	2000
7	51.143370	-114.011856	2000
8	51.143370	-114.010524	2000
9	51.143370	-114.009192	2000

```
from folium import FeatureGroup, LayerControl, Map, Marker
from folium.plugins import HeatMap

hmap = Map(location=[51.03011, -114.08529], zoom_start = 8,tiles='cartodbpositron')
gradient = {.33: 'darkblue', .88: 'pink', 1: 'red'}
hm_wide = HeatMap(list(zip(traffic_volume_df.latitude.values, traffic_volume_df.longitude.values)),
                  min_opacity=0.1,
                  radius=5,
                  blur=5,
                  max_zoom=10,
                  gradient = gradient
                 )
hmap.add_child(hm_wide)
hmap
```

By zip latitude and longitude into list, and by adjusting opacity value – as small as possible, Radius—the size of displaying element, blur- clearness and gradient – color scheme for different range value.

Then finally added to the map, we get the following:



3. *Calculation Explanation*

(1) Speed Limit: we use the following code-

```
- (speed limit * road segment length) / total length of all segments
   in cell
```

To calculate the average speed limit, We loop through each multiline in the speed_df dataFrame and to find the intersection length of the multiline with corresponding grid and add them up to the total_length, and also add the product of speed and line length to weighted_total_speed, after the iteration, we can calculate the speed limit use the function defined

```
grid_df.assign(Speed_Limit=np.nan)
i=0
for polygon in grid_df['geometry']:
    weighted_total_speed=0
    total_length=0
    j=0
    for m in speed_df['multiline']:
        # convert string m to MULTILINESTRING object MultiLineString
        MultiLineString = shapely.wkt.loads(m)
        intersection=polygon.intersection(MultiLineString)
        speed=speed_df.iloc[j, 0]
        weighted_total_speed+=speed*intersection.length
        total_length+=intersection.length
        j+=1
    if total_length>0:
        grid_df.at[i, 'Speed_Limit']=math.trunc(weighted_total_speed/total_length)
    else:
        grid_df.at[i, 'Speed_Limit']=math.trunc(0)
    i+=1
```

(2) Average Traffic Volume: we use similar code to the Speed Limit calculation.
We use - (volume * road segment length)/ total length of all segments in the grid to get average volume data

```
grid_df.assign(Traffic_Volume=np.nan)
i=0
for polygon in grid_df['geometry']:
    weighted_total_volume=0
    total_length=0
    j=0
    for m in volume_df['multilinestring']:
        # convert string m to MULTILINESTRING object MultiLineString
        MultiLineString = shapely.wkt.loads(m)
        intersection=polygon.intersection(MultiLineString)
        volume=volume_df.iloc[j, 0]
        weighted_total_volume+=volume*intersection.length
        total_length+=intersection.length
        j+=1
    if total_length>0:
        grid_df.at[i, 'Traffic_Volume']=math.trunc(weighted_total_volume/total_length)
    else:
        grid_df.at[i, 'Traffic_Volume']=math.trunc(0)
    i+=1
```

(3) Traffic Signals/ Cameras/signs:

We use the similar approach to count the number of signals, cameras or signs, that is if the point is within a specific grid, we will increment the count for that grid and create a new column of specific count with corresponding categories into grid_df dataFrame

2.1.3 Total (not average) Number of Traffic Cameras

```
grid_df.assign(Traffic_Cameras=np.nan)
idx=0
for polygon in grid_df['geometry']:
    count=0
    for long, lat in zip(cameras_df['longitude'], cameras_df['latitude']):
        point = Point(long, lat)
        # check if points are inside of grid polygon
        if point.within(polygon):
            count+=1
    grid_df.at[idx, 'Traffic_Cameras'] = count
    idx+=1
```

2.1.4 Total Number of Traffic Signals

```
grid_df.assign(Traffic_Signals=np.nan)
idx=0
for polygon in grid_df['geometry']:
    count=0
    for long, lat in zip(signals_df['longitude'], signals_df['latitude']):
        point = Point(long, lat)
        if point.within(polygon):
            count+=1
    grid_df.at[idx, 'Traffic_Signals'] = count
    idx+=1
```

2.1.5 Total Number of Traffic Signs

```
grid_df.assign(Traffic_Signs=np.nan)
idx=0
for polygon in grid_df['geometry']:
    count=0
    for p in signs_df['POINT']:
        # convert string p to Point object point
        point = shapely.wkt.loads(p)
        if point.within(polygon):
            count+=1
    grid_df.at[idx, 'Traffic_Signs'] = count
```

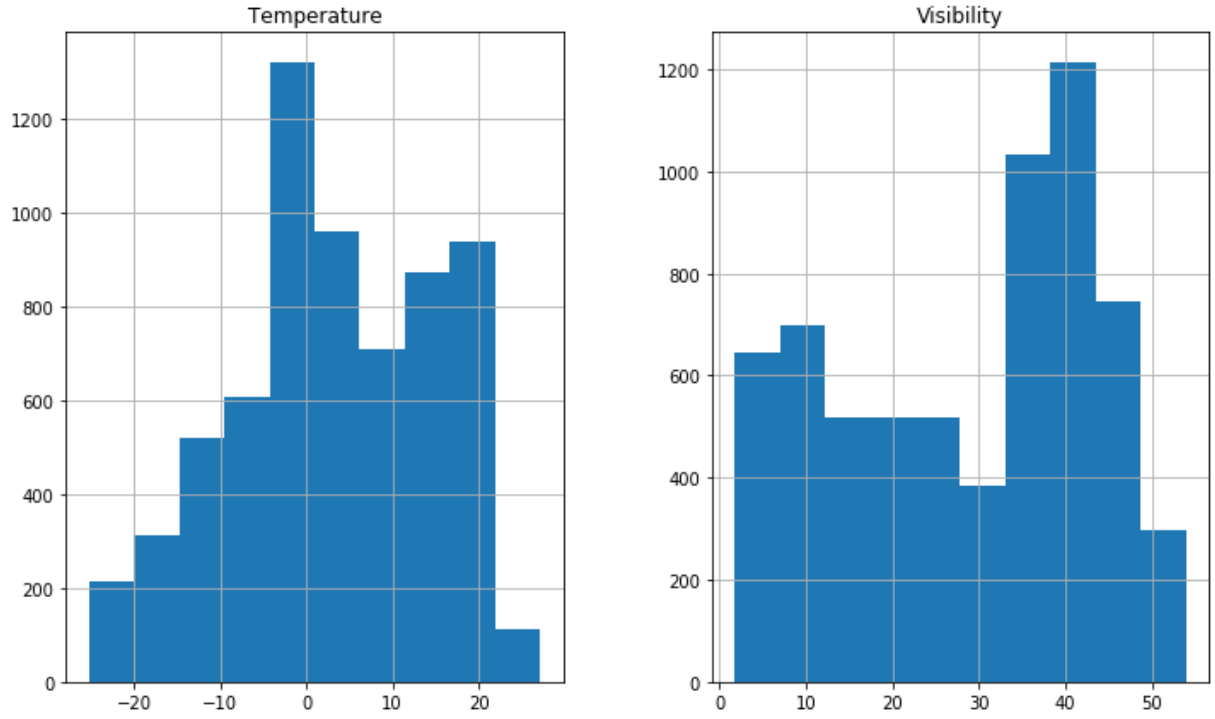
Correlation Analysis between features and Traffic Accidents:

a. Daily Temperature and Visibility Influence on Traffic Accident:

The histogram of Accident counts vs Temperature shows no relations between accident counts and temperature. Similarly, Accident counts vs Visibility scatter plot also indicates no relation between those two parameters.

However, by observing the distribution pattern, accident count peaked at temperature range (-5, 0), which may indicate that when road just start to freeze, slippery road

condition will create more traffic accidents than any other temperature range. Similarly, when the visibility range is below 50 (38-43), the number of accident count may be also highest.



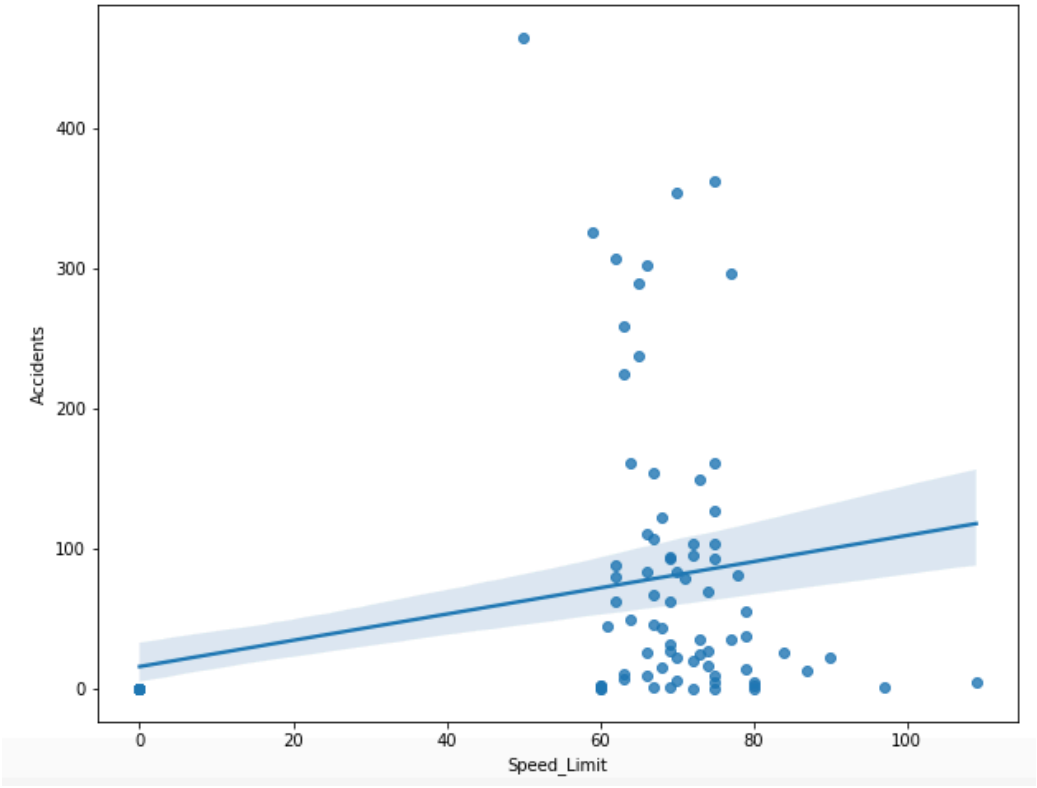
To draw these histograms, we have generated this below code-

```
fig, ax = plt.subplots(1, 2, figsize=(12,7))
# sns.scatterplot(x="Temperature", y="Accidents", ax=ax[0], data=incident_df)
# sns.scatterplot(x="Visibility", y="Accidents", ax=ax[1], data=incident_df)
incident_df.hist("Temperature", ax=ax[0])
incident_df.hist("Visibility", ax=ax[1])
plt.show()
```

b. Other Features Influence on Traffic Accident:

i. Correlation between Speed Limit and Traffic Accident:

We have study from the below graph that there is somewhat positive relation between number of Accidents and Speed Limit. However, most of the accidents can be seen within the speed limit range 60-80, which may indicates Expressways and Freeways within the city.



To visualize this correlation, we have used the following code-

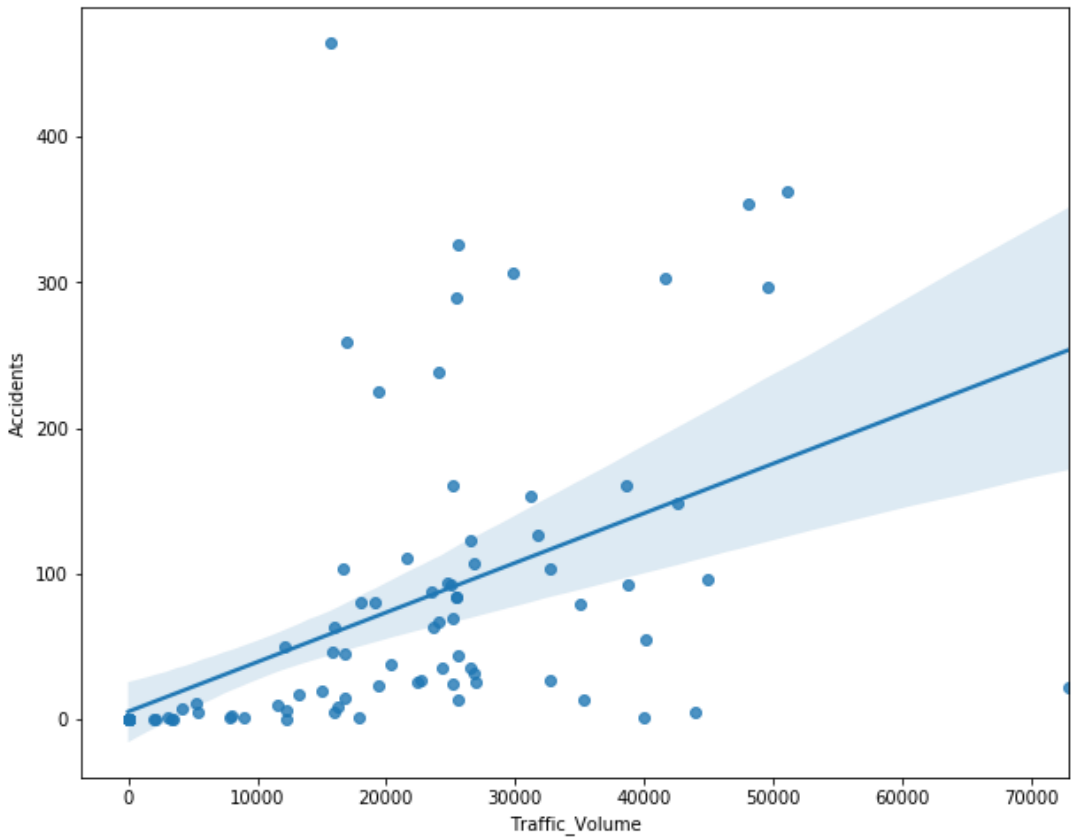
```
fig,ax=plt.subplots(figsize=(10,8))
sns.regplot(x='Speed_Limit',y='Accidents', ax=ax, data=grid_df, order=1)
```

ii. Correlation between Traffic Volume and Traffic Accident:

The below linear regression plot of Accidents Counts Vs Traffic Volume indicates a strong positive correlation between them. This pattern is within anticipation statistically, as locations with higher traffic volume has the larger sample pool so are prone to have more traffic accident. The quantified correlation will be studied at the end of this chapter

The generated code for this part is-

```
fig,ax=plt.subplots(figsize=(10,8))
sns.regplot(x='Traffic_Volume',y='Accidents', ax=ax, data=grid_df, order=1) # pointplot x vs y
```

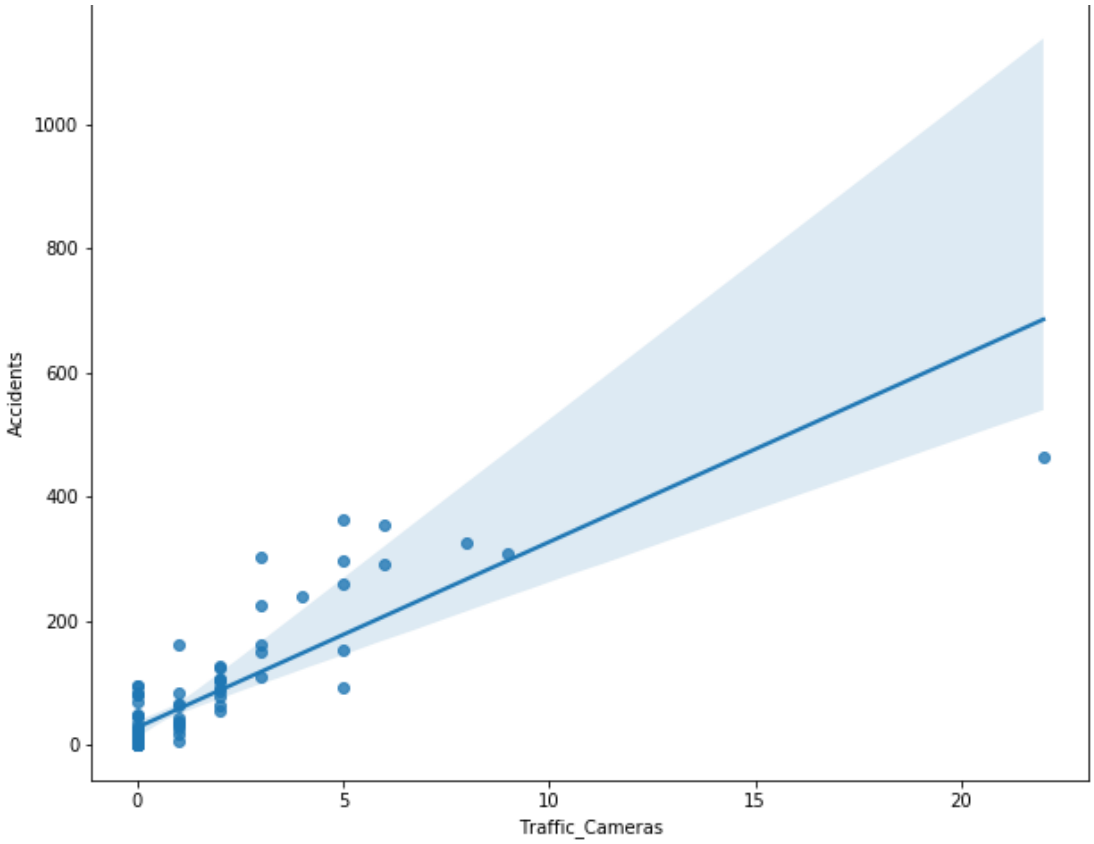


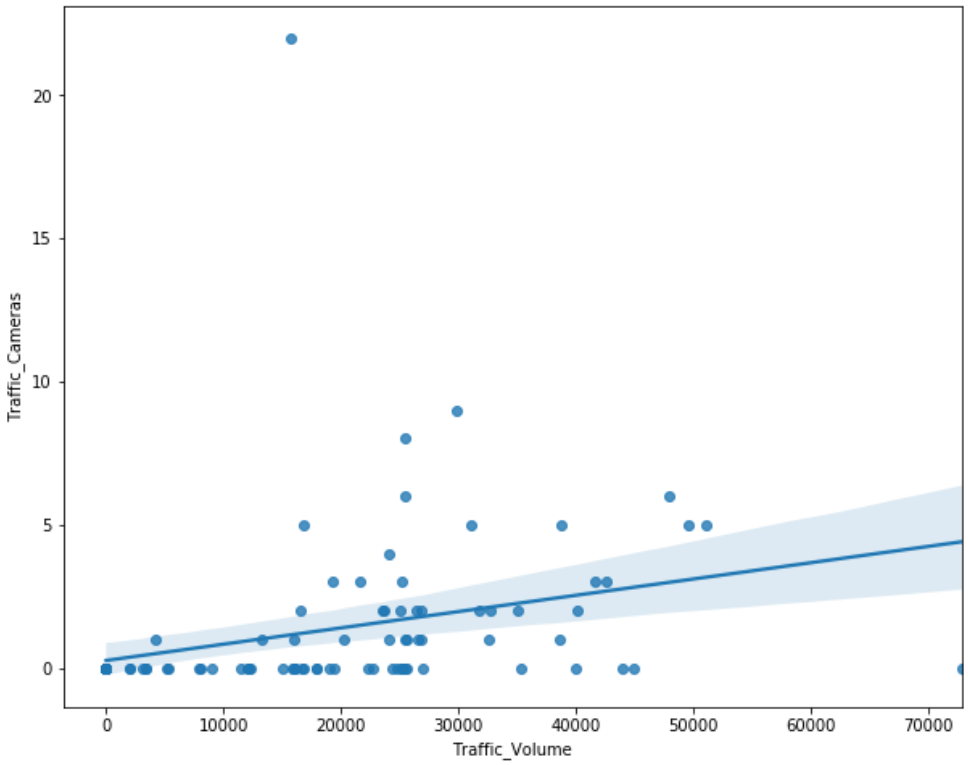
iii. Correlation between Traffic Cameras Counts and Traffic Accident:

The regression plot of Traffic Accident counts vs Traffic Cameras counts indicates a positive correlation between them. This effect may also because of the traffic volume, since the Traffic Cameras count vs Traffic Volume plot indicates a positive correlation between them. [The quantified correlation will be studied at the end of this chapter.]

To visualize this correlation, we have generated the similar code like the previous one.

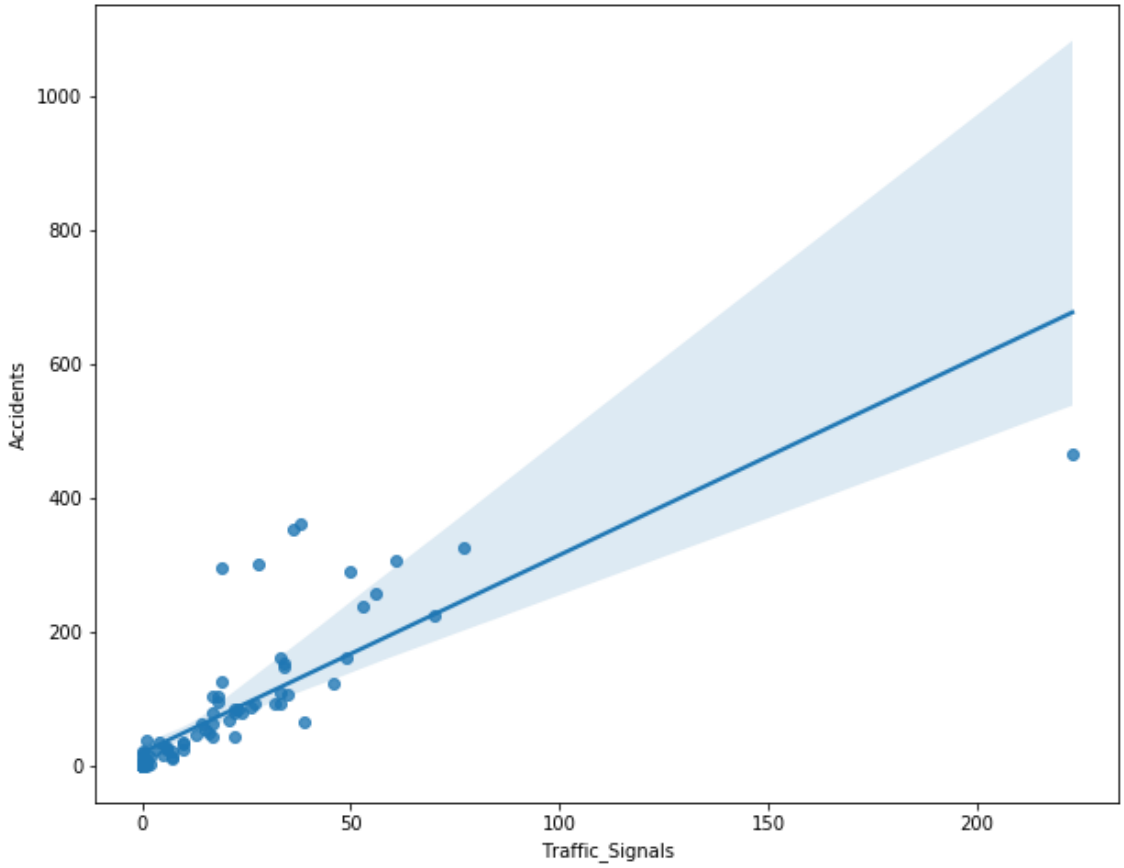
```
fig,ax=plt.subplots(figsize=(10,8))
sns.regplot(x="Traffic_Volume", y="Traffic_Cameras", ax=ax, data=grid_df)
plt.show()
```

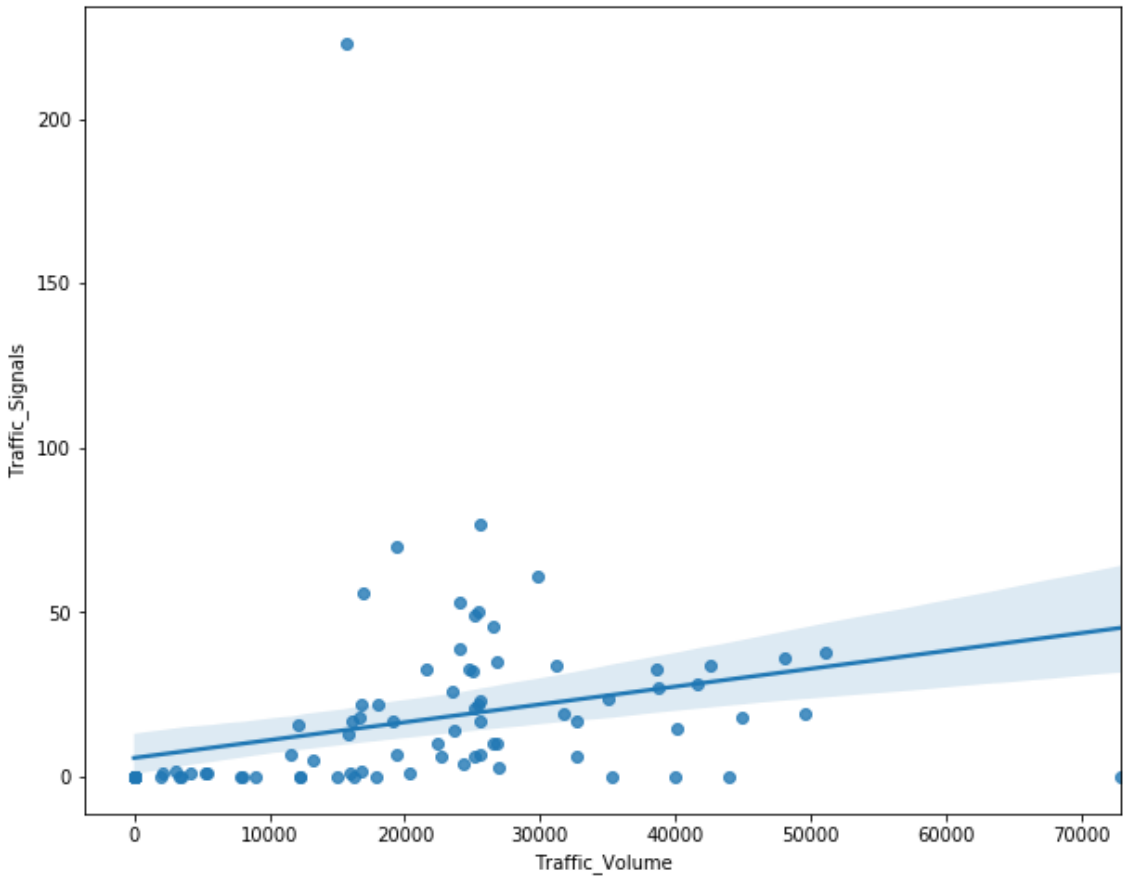

Accidents
1000
800
600
400
200
0
0
5
10
15
20
Traffic_Cameras


Traffic_Cameras
20
15
10
5
0
0
10000
20000
30000
40000
50000
60000
70000
Traffic_Volume

iv. Correlation between Traffic Signal Counts and Traffic Accident:

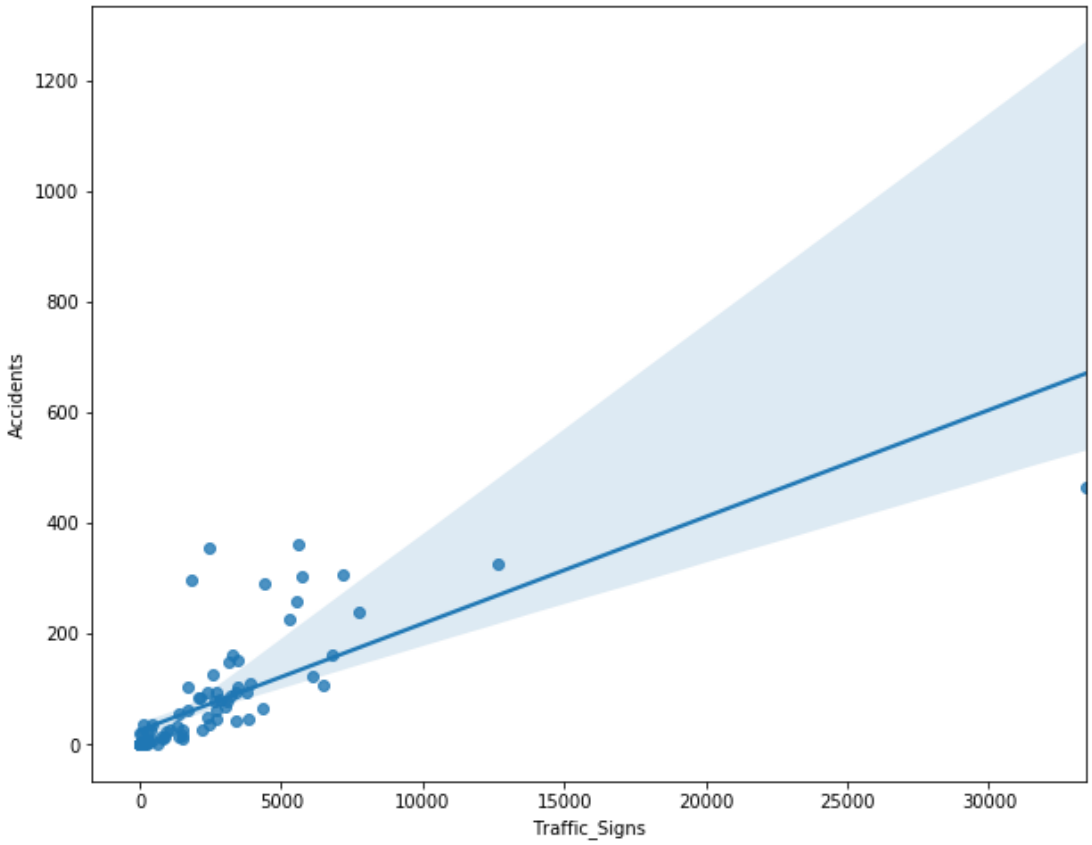
The regression plot of Traffic Accident counts vs Traffic Signal counts indicates a positive correlation between them. Like Traffic Cameras, this correlation may also because of the traffic volume, since the Traffic Signal count vs Traffic Volume plot indicates a positive correlation between them. [The quantified correlation will be studied at the end of this chapter.]

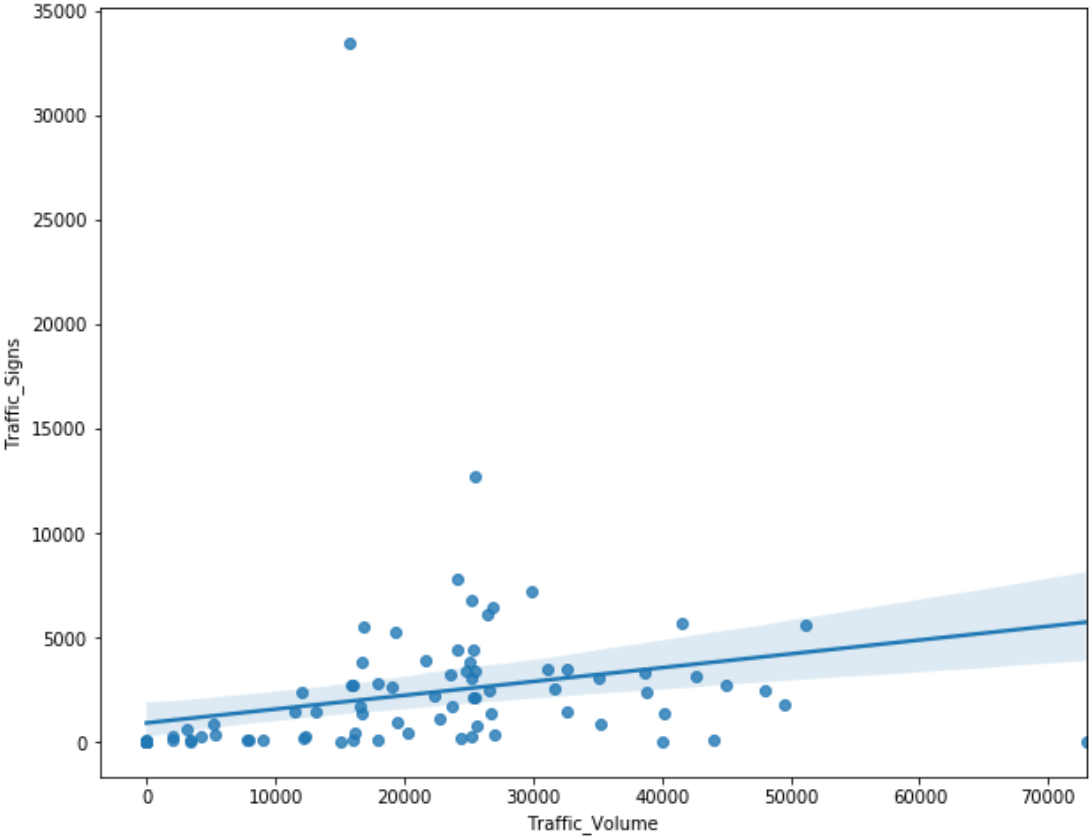




v. Correlation between Traffic Sign Counts and Traffic Accident:

The regression plot of Traffic Accident counts vs Traffic Sign counts indicates a positive correlation between them. Like Traffic Signals, this correlation may also because of the traffic volume, since the Traffic Sign count vs Traffic Volume plot indicates a positive correlation between them. [The quantified correlation will be studied at the end of this chapter.]


Accidents
Traffic_Signs
0
200
400
600
800
1000
1200
0
5000
10000
15000
20000
25000
30000


Traffic_Signs
Traffic_Volume
0
5000
10000
15000
20000
25000
30000
35000
0
10000
20000
30000
40000
50000
60000
70000

c. Quantitative Study of Factors Contribute to Traffic Accidents:

Three quantitative correlation study (Pearson, Kendall, Spearman) have been performed in this section.

In all three studies, we have observed positive correlations between any or the features and accident counts except Sppeed_Limit. Among those features, Traffic Sign counts and Traffic Signal counts are showing the strongest correlation to accidents.

i. Pearson Correlation Study: The generated code and output is-

```
parameters=['Accidents', 'Speed_Limit','Traffic_Volume', 'Traffic_Cameras', 'Traffic_Signals', 'Traffic_Signs']
grid_df[parameters].corr(method='pearson')
```

	Accidents	Speed_Limit	Traffic_Volume	Traffic_Cameras	Traffic_Signals	Traffic_Signs
Accidents	1.000000	0.299831	0.528462	0.846082	0.818475	0.757298
Speed_Limit	0.299831	1.000000	0.707985	0.168168	0.213830	0.196862
Traffic_Volume	0.528462	0.707985	1.000000	0.311684	0.302962	0.261465
Traffic_Cameras	0.846082	0.168168	0.311684	1.000000	0.919970	0.900012
Traffic_Signals	0.818475	0.213830	0.302962	0.919970	1.000000	0.972463
Traffic_Signs	0.757298	0.196862	0.261465	0.900012	0.972463	1.000000

ii. Kendall Correlation Study:

```
grid_df[parameters].corr(method='kendall')
```

	Accidents	Speed_Limit	Traffic_Volume	Traffic_Cameras	Traffic_Signals	Traffic_Signs
Accidents	1.000000	0.282082	0.633239	0.701243	0.835438	0.788238
Speed_Limit	0.282082	1.000000	0.555923	0.138387	0.169990	0.201583
Traffic_Volume	0.633239	0.555923	1.000000	0.473617	0.528490	0.537230
Traffic_Cameras	0.701243	0.138387	0.473617	1.000000	0.695502	0.611603
Traffic_Signals	0.835438	0.169990	0.528490	0.695502	1.000000	0.825797
Traffic_Signs	0.788238	0.201583	0.537230	0.611603	0.825797	1.000000

iii. Spearman Correlation Study:

```
grid_df[parameters].corr(method='spearman')
```

	Accidents	Speed_Limit	Traffic_Volume	Traffic_Cameras	Traffic_Signals	Traffic_Signs
Accidents	1.000000	0.439067	0.803756	0.800618	0.940578	0.929010
Speed_Limit	0.439067	1.000000	0.712159	0.193739	0.277248	0.360214
Traffic_Volume	0.803756	0.712159	1.000000	0.590118	0.679468	0.714335
Traffic_Cameras	0.800618	0.193739	0.590118	1.000000	0.792520	0.743034
Traffic_Signals	0.940578	0.277248	0.679468	0.792520	1.000000	0.943018
Traffic_Signs	0.929010	0.360214	0.714335	0.743034	0.943018	1.000000

d. Weather condition:

As we have not found any correlations between weather conditions (Temperature and Visibility) and accident counts, it may be helpful to confirm this conclusion by using quantitative tools as well.

As expected, all results from three methods are showing very weak correlations between weather conditions and traffic accident counts. However, on a side note, we can observe a somewhat positive correlation between temperature and visibility, which is also true by common sense. All the findings are as follows-

i. Pearson Correlation Study:

```
parameters=['Accidents', 'Temperature', 'Visibility']
incident_df[parameters].corr(method='pearson')
```

	Accidents	Temperature	Visibility
Accidents	1.000000	0.044856	0.037522
Temperature	0.044856	1.000000	0.229565
Visibility	0.037522	0.229565	1.000000

Ii. Kendall Correlation Study:

```
incident_df[parameters].corr(method='kendall')
```

	Accidents	Temperature	Visibility
Accidents	1.000000	0.028576	0.024483
Temperature	0.028576	1.000000	0.157612
Visibility	0.024483	0.157612	1.000000

iii. Spearman Correlation Study:

```
incident_df[parameters].corr(method='spearman')
```

	Accidents	Temperature	Visibility
Accidents	1.000000	0.042133	0.036063
Temperature	0.042133	1.000000	0.230376
Visibility	0.036063	0.230376	1.000000

4. Conclusion

From all the plots and quantitative calculations, we can see that Traffic_Signals and Traffic_Signs had a very strong positive affect on number of accidents in Calgary city in 2018. However, having moderate positive correlation with Traffic_Volume and Traffic_Cameras, the number of accidents was independ of the Speed_Limit. Though Traffic_Signal, Traffic_Cameras and Traffic_Sign had a dependency on number of Traffic_Volume of Calgary city in 2018.