A1.

In [900]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pl
from scipy.stats import linregress
import seaborn as sns
%matplotlib inline
from pylab import rcParams
rcParams['figure.figsize'] = 10, 8
```

In [901]:

```
birth = pd.read_csv("Births.csv")
birth.head()
```

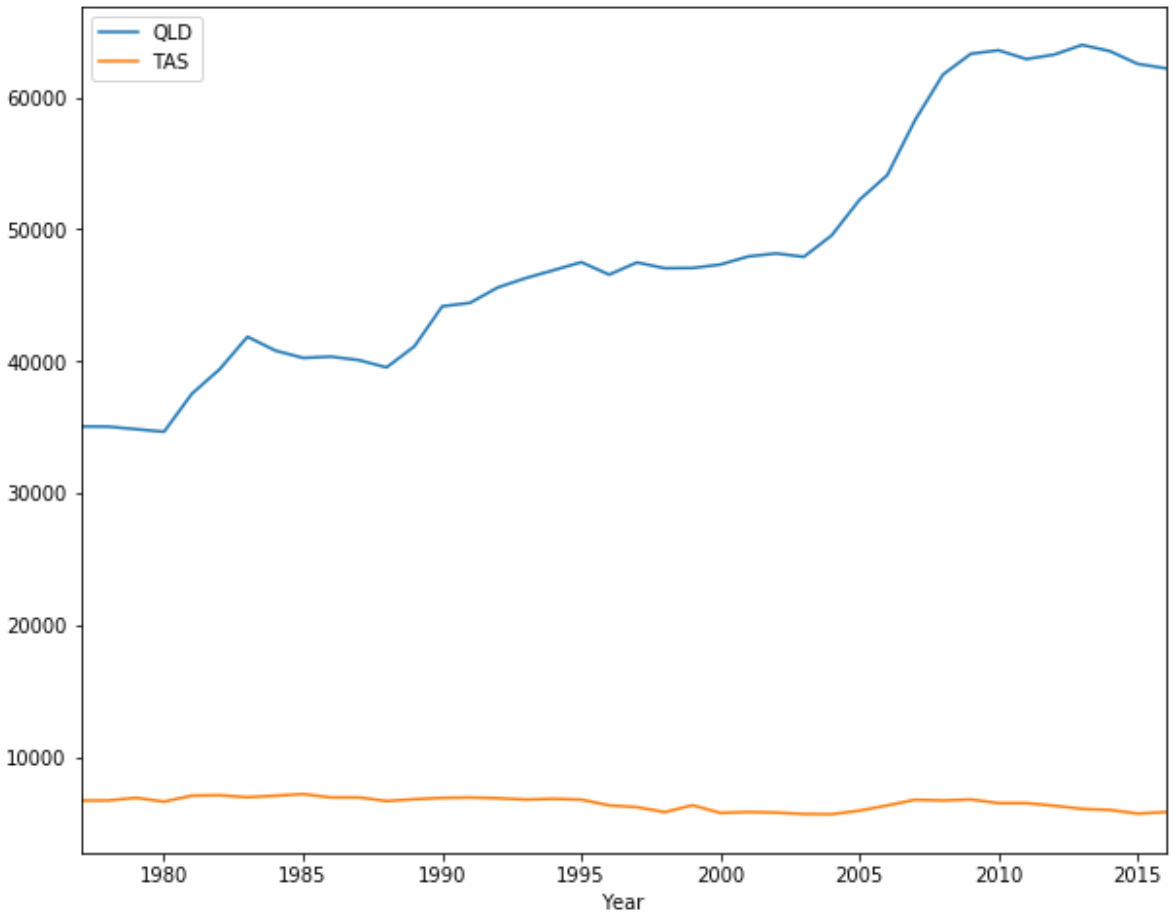
Out[901]:

	Year	NSW	ACT	QLD	SA	WA	TAS	VIC	NT
0	1977	78173	4381	35069	19086	20981	6739	59602	2923
1	1978	78190	4342	35054	18964	21094	6751	59364	2600
2	1979	77669	4217	34858	18403	20523	6947	58006	2747
3	1980	78859	4181	34666	18317	20354	6660	57768	2859
4	1981	80980	4193	37545	18960	21277	7112	58104	2749

A1.1.a

In [902]:

```
ax = pl.gca()
birth.plot(x='Year',kind='line',y='QLD',legend=True,ax=ax)
birth.plot(x='Year',kind='line',y='TAS',legend=True, ax=ax)
pl.show()
```



In the above graph we observe that as the years are increasing the number of births for QueensLand is gradually increasing while birth rate of Tasmania remains nearly constant as compared to Queensland

gradually increasing while birth rate of Tasmania remains nearly constant as compared to Queensland

A1.1.b

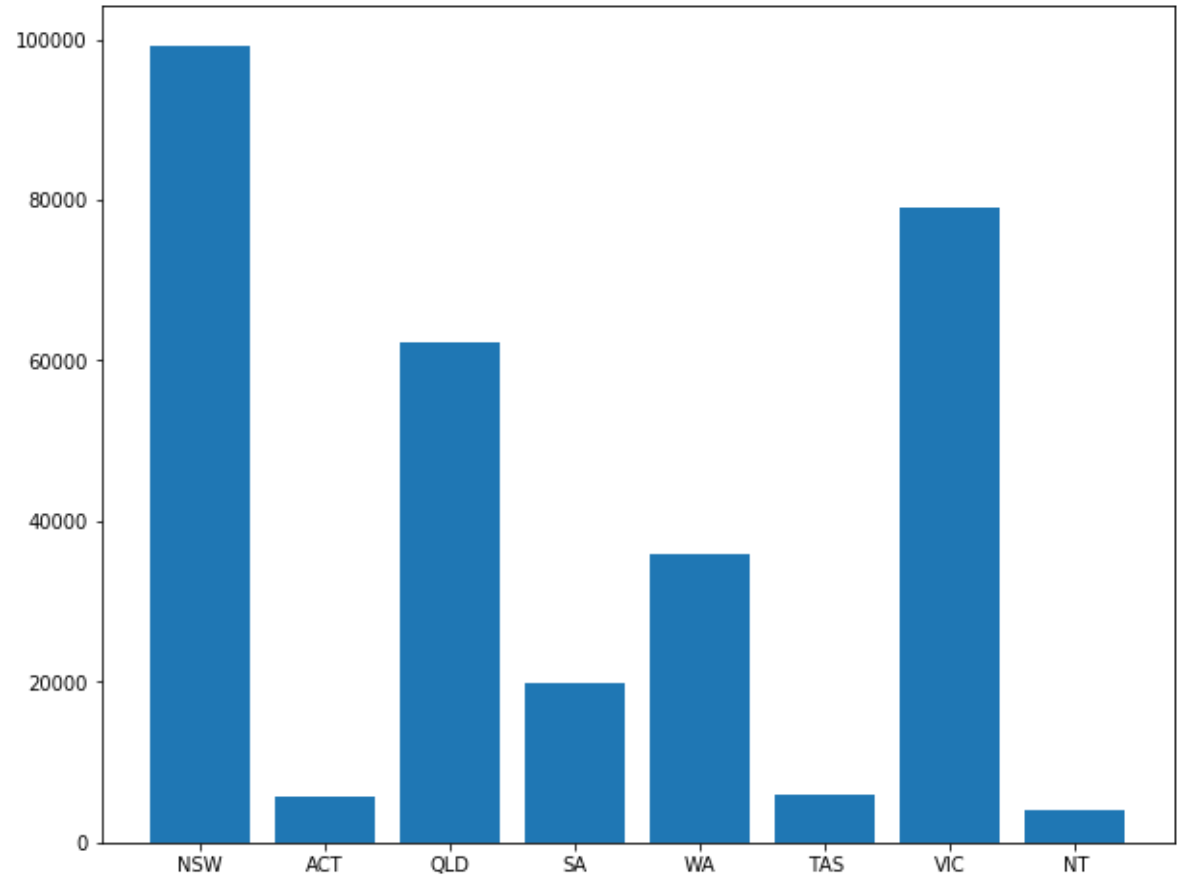
In [903]:

```
birth_1 = pd.DataFrame(birth)
result = birth_1.loc[birth_1['Year'] ==2016]
new_birth=result.drop(columns='Year')
print("Given Dataframe in year 2016:\n", new_birth.head())
```

```
Given Dataframe in year 2016:
       NSW   ACT    QLD     SA     WA   TAS    VIC    NT
39  99260  5703  62190  19917  35875  5869  78953  3928
```

In [904]:

```
#Resetting the index, for creation of bar chart
new_birth.reset_index(drop=True, inplace=True)
col = new_birth.columns[0:]
row = new_birth.values[0]
pl.bar(col,row)
pl.show()
```



The above bar chart shows that the birth rate of different states of Australia in 2016, where we could clearly see that birth rate of NSW is highest and North Territory be the lowest as compared to other states. While ACT and TAS are nearly same in its birth count

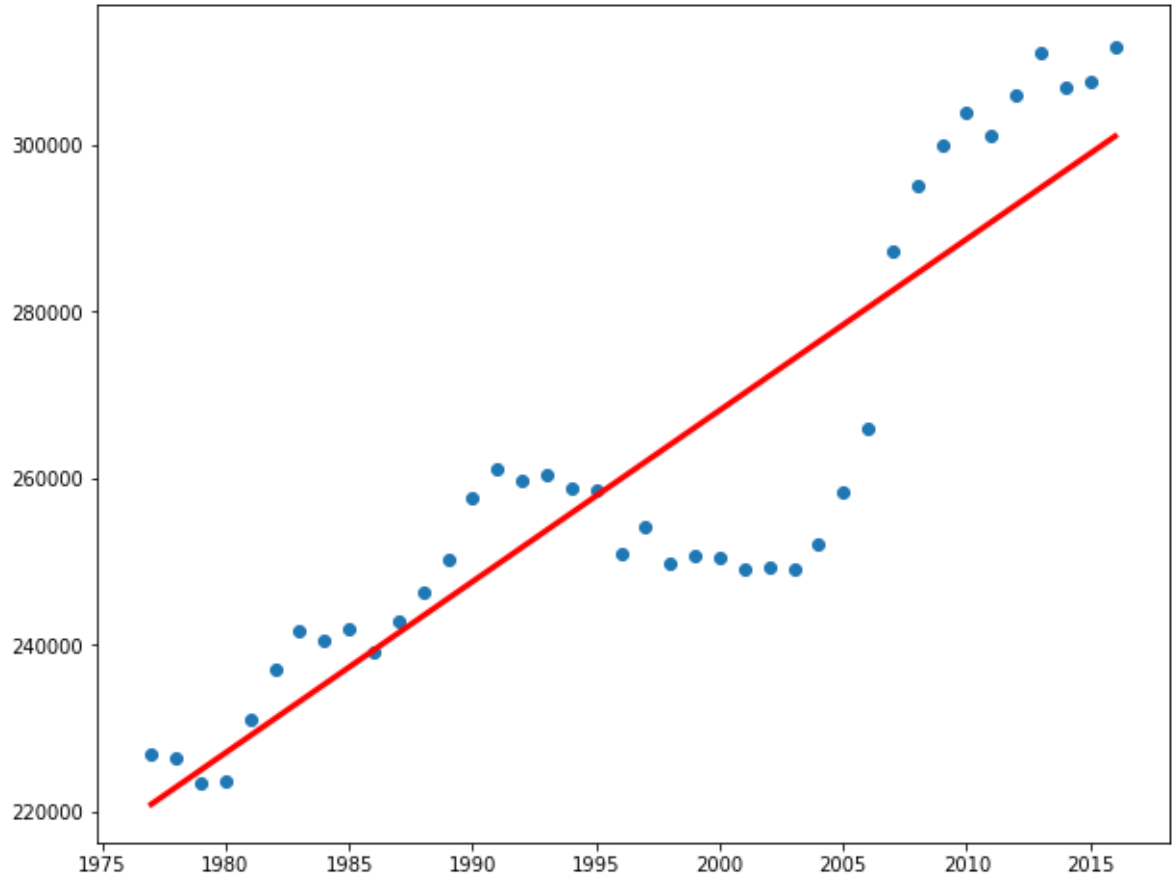
A.1.2.a

In [905]:

```
new_birth = birth_1.set_index('Year')
new_birth['Total']=new_birth.sum(axis=1)
new_birth = new_birth.reset_index(drop=False)
```

In [906]:

```
s, i, r, p, e = linregress(new_birth['Year'],new_birth['Total'])
line = [s*x + i for x in new_birth['Year']]
pl.plot(new_birth['Year'],line,'r-',linewidth=3)
pl.scatter(new_birth['Year'], new_birth['Total'])
pl.show()
```



A.1.2.b

No, as our scatter plot is not fitted on linear plot as per the graph. From year 1995 to 2017 we could see that the birth rate has a huge deviation.

A.1.2.c

In [907]:

```
#For predicting the model for year 2050 and 2100, implementing linear model
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
labelx=new_birth[['Year']]
labely=new_birth[['Total']]
reg.fit(labelx,labely)
print("Predicted values of 2050 and 2100")
reg.predict([[2050],[2100]])
```

```
Predicted values of 2050 and 2100
```

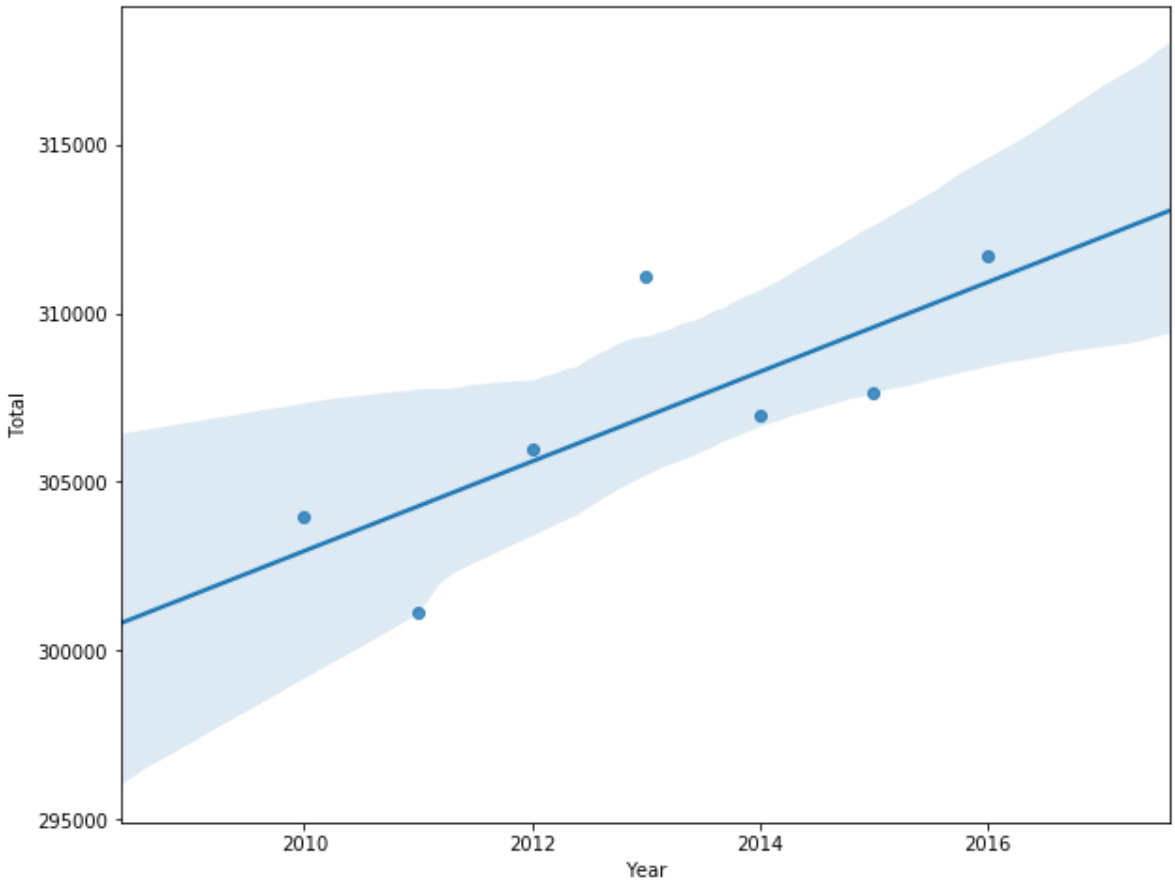
Out[907]:

```
array([[370945.74399625],
       [473754.24305816]])
```

A.1.2.d

In [908]:

```
new_birth=new_birth[-7:]
lm = sns.regplot(x='Year',y='Total', data = new_birth)
```



In [909]:

```
#Predicting the model for 2050 and 2100
lab_x=new_birth[['Year']]
lab_y=new_birth[['Total']]
reg.fit(lab_x,lab_y)
reg.predict([[2050],[2100]])
print("Predicted value for 2050 and 2100 as per the linear regression prediction")
reg.predict([[2050],[2100]])
```

```
Predicted value for 2050 and 2100 as per the linear regression prediction
```

Out[909]:

```
array([[355966.        ],
       [422230.28571429]])
```

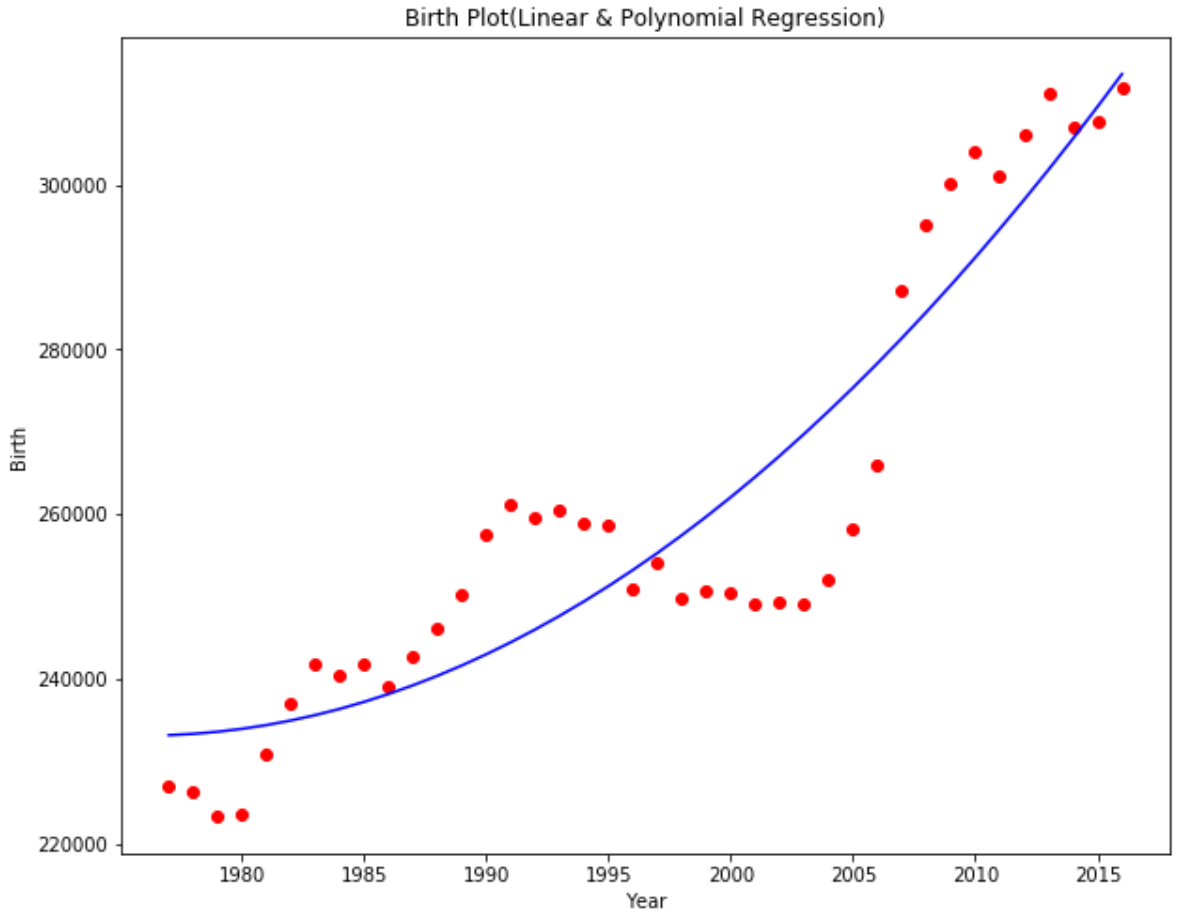
A.1.2.d

In the above graph we could see the 2-D view the linear regression graph. As per the model comparision I believe the first model was the better, as we had a large set of dataframe for that model as compared to the other dataframe where we have calculated the data from 2010,which causes prediction to varies.

A.1.2.e

In [910]:

```
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
X_poly = poly_reg.fit_transform(labelx)
pol_reg = LinearRegression()
pol_reg.fit(X_poly, labely)

# Visualizing the Polymonial Regression results
def viz_polymonial():
    pl.scatter(labelx, labely, color='red')
    pl.plot(labelx, pol_reg.predict(poly_reg.fit_transform(labelx)), color='blue')
    pl.title('Birth Plot(Linear & Polynomial Regression)')
    pl.xlabel('Year')
    pl.ylabel('Birth')
    pl.show()
    return
viz_polymonial()
```



In [911]:

```
# Predicting a new result with Polymonial Regression
x = pol_reg.predict(poly_reg.fit_transform([[2050]]))
y = pol_reg.predict(poly_reg.fit_transform([[2100]]))
print(x)
print(y)
```

```
[[507485.46080512]]
[[1003052.03131562]]
```

For this case we are following polynomial regression as it is cleary visible from the plot that the quadratic curve fits the data better than the stright line i.e linear Here we have used degree=2, which prevents over-fitting, we can add more training samples so that the algorithm doesn't learn the noise in the system and can become more generalized.

A1.3.

In [912]:

```
#Reading the TFR rate csv
fer=pd.read_csv('TFR.csv')
fer.head()
```

Out[912]:

	Year	NSW	VIC	QLD	SA	WA	TAS	NT	ACT
0	1971	2.806	2.858	3.025	2.629	3.047	2.903	4.029	2.908
1	1972	2.653	2.634	2.854	2.423	2.673	2.695	3.477	2.728
2	1973	2.385	2.393	2.637	2.192	2.391	2.461	3.291	2.493
3	1974	2.303	2.298	2.500	2.113	2.263	2.422	2.988	2.444
4	1975	2.125	2.099	2.316	1.999	2.182	2.232	2.846	2.137

Task A1.3.a

In [913]:

```
minimum=fer['QLD'].min()
min_yr=fer.query('QLD == 1.8')
o=min_yr[['Year','QLD','NT']]
print("Given Dataframe gives min value for TFR in QLD:", minimum)
print("\n")
print("Given Dataframe gives year and TR recorded in QLD and NT:\n",o)
```

```
Given Dataframe gives min value for TFR in QLD: 1.8


Given Dataframe gives year and TR recorded in QLD and NT:
     Year  QLD     NT
28  1999  1.8  2.123
```

A.1.4

A.1.4.a

In [914]:

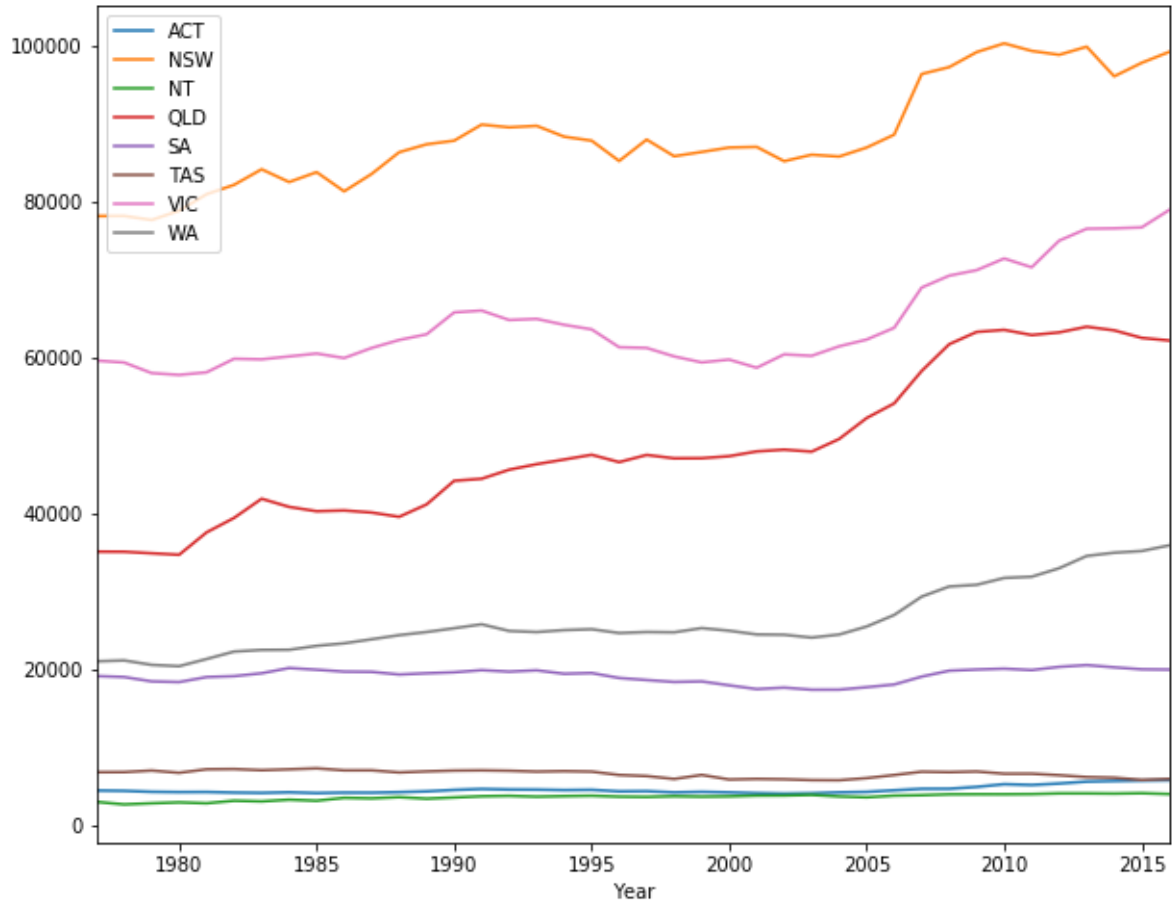
```
birth_2=pd.read_csv("Births.csv")
birth_3= birth_2.set_index('Year')
death_2=pd.read_csv("Deaths.csv")
death_3= death_2.set_index('Year')
natural_growth=birth_3.subtract(death_3)
natural_growth.head()
```

Out[914]:

Year	ACT	NSW	NT	QLD	SA	TAS	VIC	WA
1977	3595	36098	2208	18078	9371	3512	29535	13067
1978	3560	38069	1988	18701	9196	3452	30021	13313
1979	3458	37694	2161	18231	8655	3629	28483	12744
1980	3426	39060	2271	18892	8781	3461	28843	12276
1981	3398	41001	2078	20350	9154	3618	28960	12932

In [915]:

```
ax1 = pl.gca()
birth.plot(x='Year',kind='line',y='ACT', legend=True, ax=ax1)
birth.plot(x='Year',kind='line',y='NSW', legend=True, ax=ax1)
birth.plot(x='Year',kind='line',y='NT',legend=True, ax=ax1)
birth.plot(x='Year',kind='line',y='QLD', legend=True, ax=ax1)
birth.plot(x='Year',kind='line',y='SA',legend=True,ax=ax1)
birth.plot(x='Year',kind='line',y='TAS', legend=True, ax=ax1)
birth.plot(x='Year',kind='line',y='VIC', legend=True, ax=ax1)
birth.plot(x='Year',kind='line',y='WA', legend=True, ax=ax1)
pl.show()
```



From the above graph we could see that New South Whales polulation growth is volatile as compared to rest.Altough we can also observe, VIC and QLD is growth rate is increasing.But we could also see a drop in growth rate in year 2014 for QLD and NSW. For states like WA, there was no major changes in the growth rate from 1977 to 2008, but after that we could see the sudden growth in the dataframe. While ACT, TAS and NT has the growth rate in same level. Hence, from all the graph we could see that except ACT, TAS and NT every state has increasing growth rate which can be because of the end of war period and increase in immigrants in the Australia.

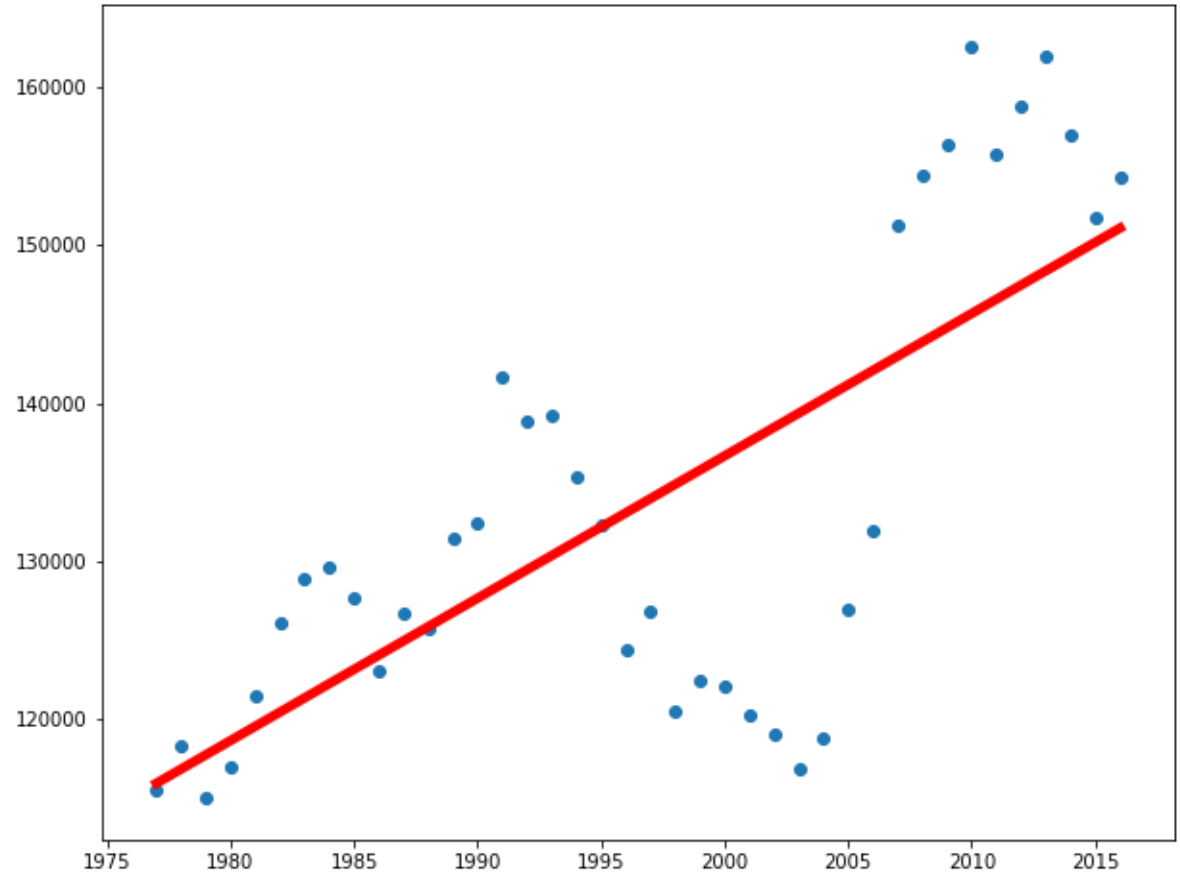
In [916]:

```
#Creating the total population column for the plotting
natural_growth['Total_Pop']=natural_growth.sum(axis=1)
natural_growth_1 = natural_growth.reset_index(drop=False)
print(natural_growth_1.head())
```

```
   Year   ACT    NSW    NT    QLD    SA   TAS    VIC     WA  Total_Pop
0  1977  3595  36098  2208  18078  9371  3512  29535  13067     115464
1  1978  3560  38069  1988  18701  9196  3452  30021  13313     118300
2  1979  3458  37694  2161  18231  8655  3629  28483  12744     115055
3  1980  3426  39060  2271  18892  8781  3461  28843  12276     117010
4  1981  3398  41001  2078  20350  9154  3618  28960  12932     121491
```

In [917]:

```
#Plotting the linear regression
s, i, r, p, e = linregress(natural_growth_1['Year'],natural_growth_1['Total_Pop'])
stright = [s*x + i for x in natural_growth_1['Year']]
pl.plot(natural_growth_1['Year'],stright,'r-', linewidth=5)
pl.scatter(natural_growth_1['Year'], natural_growth_1['Total_Pop'])
pl.show()
```



As per the linear regression, the growth rate should increase ideally. But we could analyse that by the year 1997 to 2007 there was a dip in growth rate of the population. While after 2007 the graph showed a positive slope and comes under the linear regression.

Task:A2

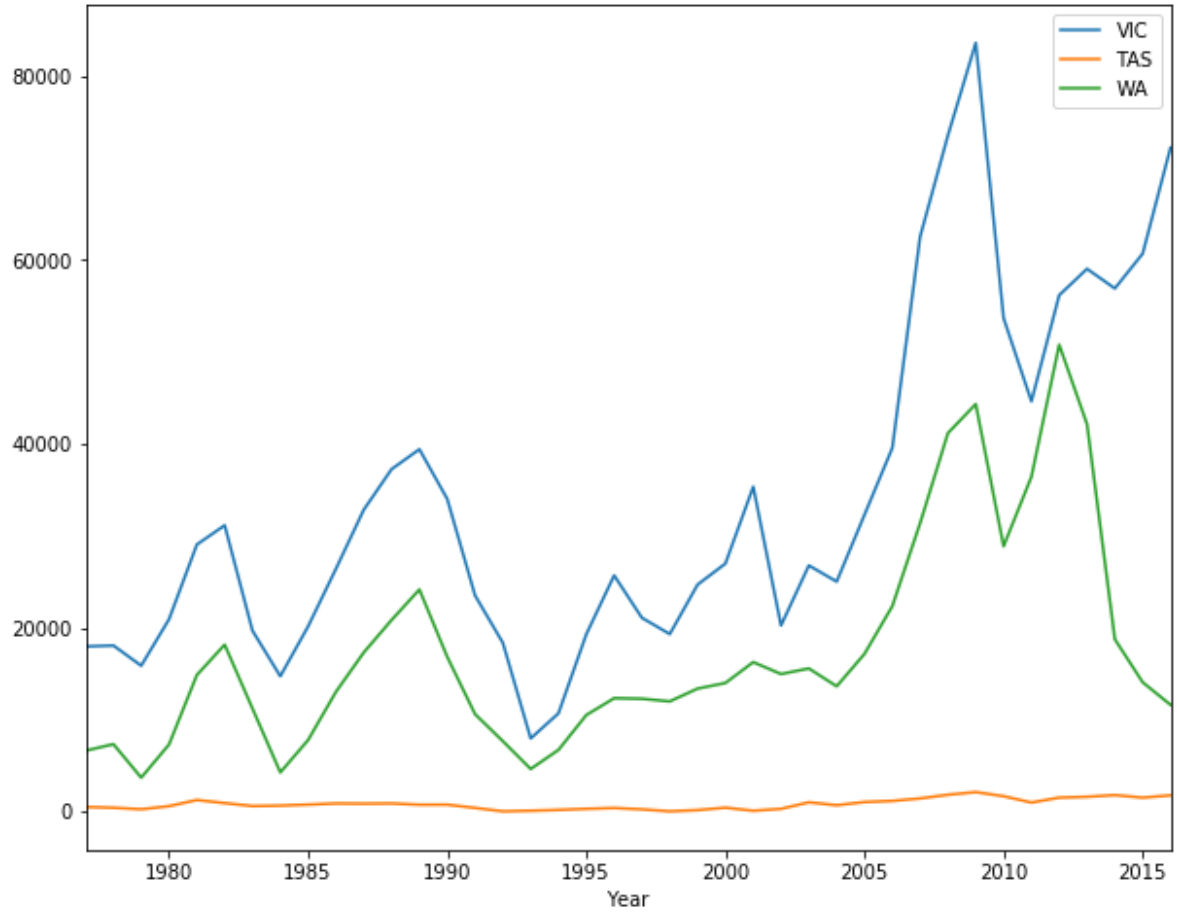
In [918]:

```
nom=pd.read_csv("NOM.csv")
nom_set= nom.set_index('Year')
print(nom_set.head())
```

```
        NSW    VIC    QLD    SA     WA    TAS    NT   ACT
Year
1977  25236  17969   4012  2874   6631   506   408   261
1978  25825  18068   6857  2638   7340   428  1428   131
1979  28086  15874   6489   541   3698   263   442  -256
1980  33499  20932   9480  3325   7309   599   361   436
1981  47291  29052  18168  6655  14866  1268  1043   832
```

In [919]:

```
    ax2 = pl.gca()
    nom.plot(kind='line',x='Year',y='VIC',legend=True,ax=ax2)
    nom.plot(kind='line',x='Year',y='TAS', legend=True, ax=ax2)
    nom.plot(kind='line',x='Year',y='WA', legend=True, ax=ax2)
    pl.show()
```

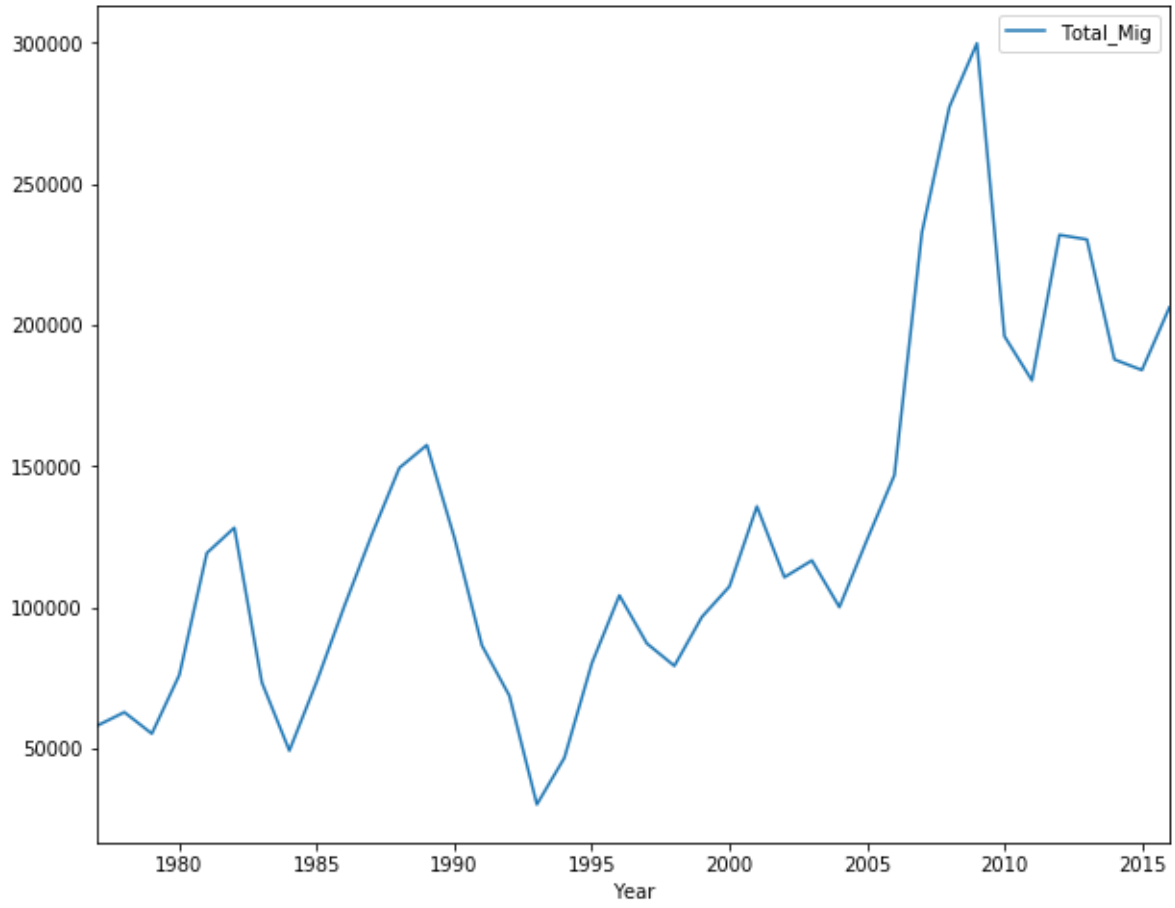


As per the above plot, we could observe the NOM rate of Victoria is highest in 2008 while there was a sudden dip till 2012. While the similar trend observed with lower initial NOM rate for WA, which was highest in 2013 with a sudden dip till 2016. Lastly for Tasmania, the growth rate was constant through out the period of 1977 till 2016.

A2.1.b

In [920]:

```
nom_set['Total_Mig']=nom_set.sum(axis=1)
nom_reset=nom_set.reset_index()
ax2 = pl.gca()
nom_reset.plot(x='Year',kind='line',y='Total_Mig',legend=True,ax=ax2)
pl.show()
```



The net overseas migration captures the difference between permanent departures and additions, as well as long-term arrivals & departures. While,for few years 2007-08 to 2008-09 when temporary long-term migration rate had grown. Plus there where many temporary arrivals who has managed to get permanent visas on-shore (or because, as New Zealand citizens, they didn't need such visas). This phenomenon also helps explain periods such as the late 1980s, and most years after 2006-07 when we analyse a strange trend in the graph i.e the net figures are higher.

A2.2

In [921]:

```
Over=pd.read_csv("NOM.csv")
Over= Over.set_index('Year')
Inter=pd.read_csv("NIM.csv")
Inter= Inter.set_index('Year')
Inter_final=Inter.rename(columns=
                         {"NSW":"NSW_NIM","VIC":"VIC_NIM","QLD":"QLD_NIM","SA":"SA_NIM","WA":"
Over_final=Over.rename(columns=
                         {"NSW":"NSW_NOM","VIC":"VIC_NOM","QLD":"QLD_NOM","SA":"SA_NOM","WA":"
con = pd.concat([NOM_final,NIM_final],axis=1)
print('On combining the data for both NIM and NOM into a single dataframe')
con.head()
```

```
On combining the data for both NIM and NOM into a single dataframe
```

Out[921]:

Year	NSW_NOM	VIC_NOM	QLD_NOM	SA_NOM	WA_NOM	TAS_NOM	NT_NOM	ACT_NOM	1
1977	25236	17969	4012	2874	6631	506	408	261	
1978	25825	18068	6857	2638	7340	428	1428	131	
1979	28086	15874	6489	541	3698	263	442	-256	
1980	33499	20932	9480	3325	7309	599	361	436	
1981	47291	29052	18168	6655	14866	1268	1043	832	

In [922]:

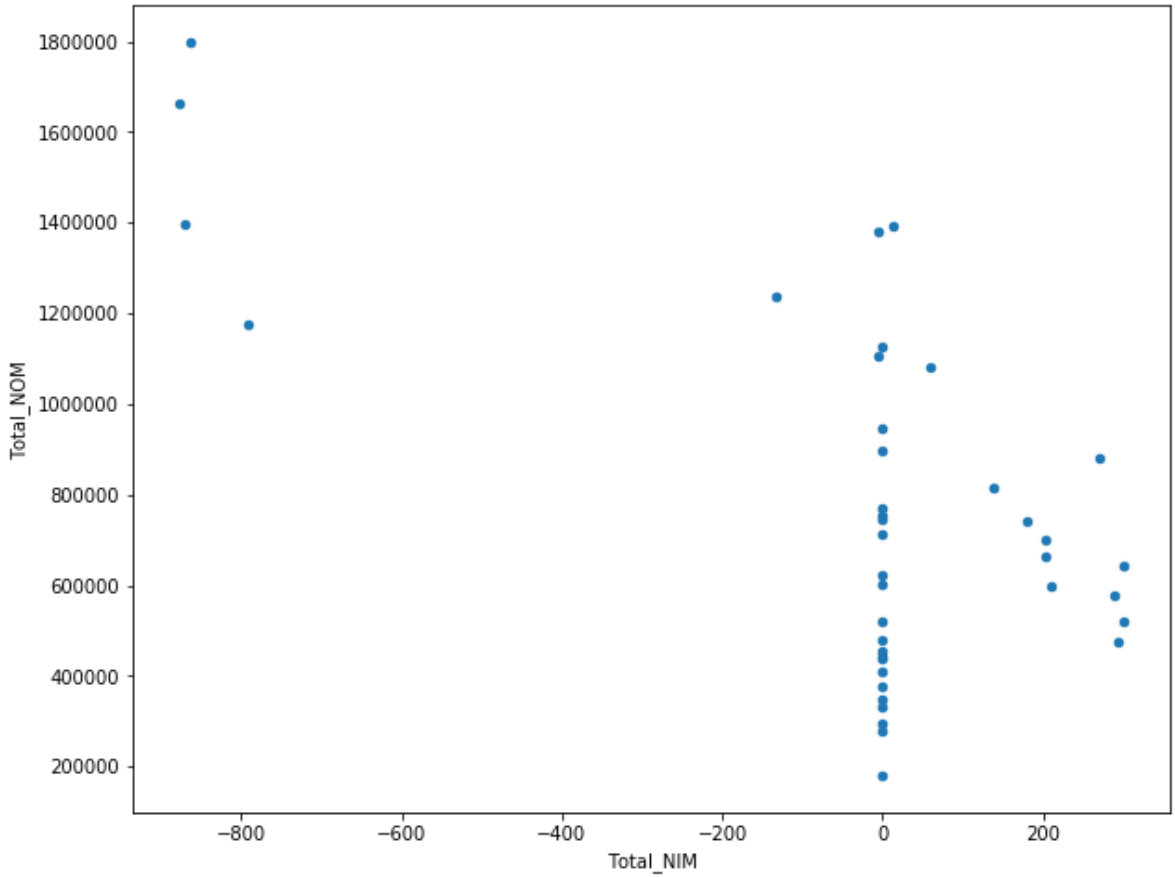
```
#First and Last year of dataframe
nom = con.reset_index(drop=False)
sort_df=nom.sort_values(by=['Year'])
first=sort_df["Year"].iloc[0]
last=sort_df["Year"].iloc[-1]
print("First Year of dataframe:",first)
print("Last Year of dataframe:",last)
```

```
First Year of dataframe: 1977
Last Year of dataframe: 2016
```

A2.2.b

In [923]:

```
Nom=NOM_final
Nim=NIM_final

Nom['Total_NOM']=Nom.sum(axis=1)

Nim['Total_NIM']=Nim.sum(axis=1)

Total_Nom = Nom.reset_index(drop=False)

Total_Nim = Nim.reset_index(drop=False)

concat = pd.concat([Total_Nom,Total_Nim],axis=1)

conct=concat[['Total_NIM','Total_NOM']].copy()


scatter_conct = conct.plot.scatter(x="Total_NIM", y="Total_NOM")
```

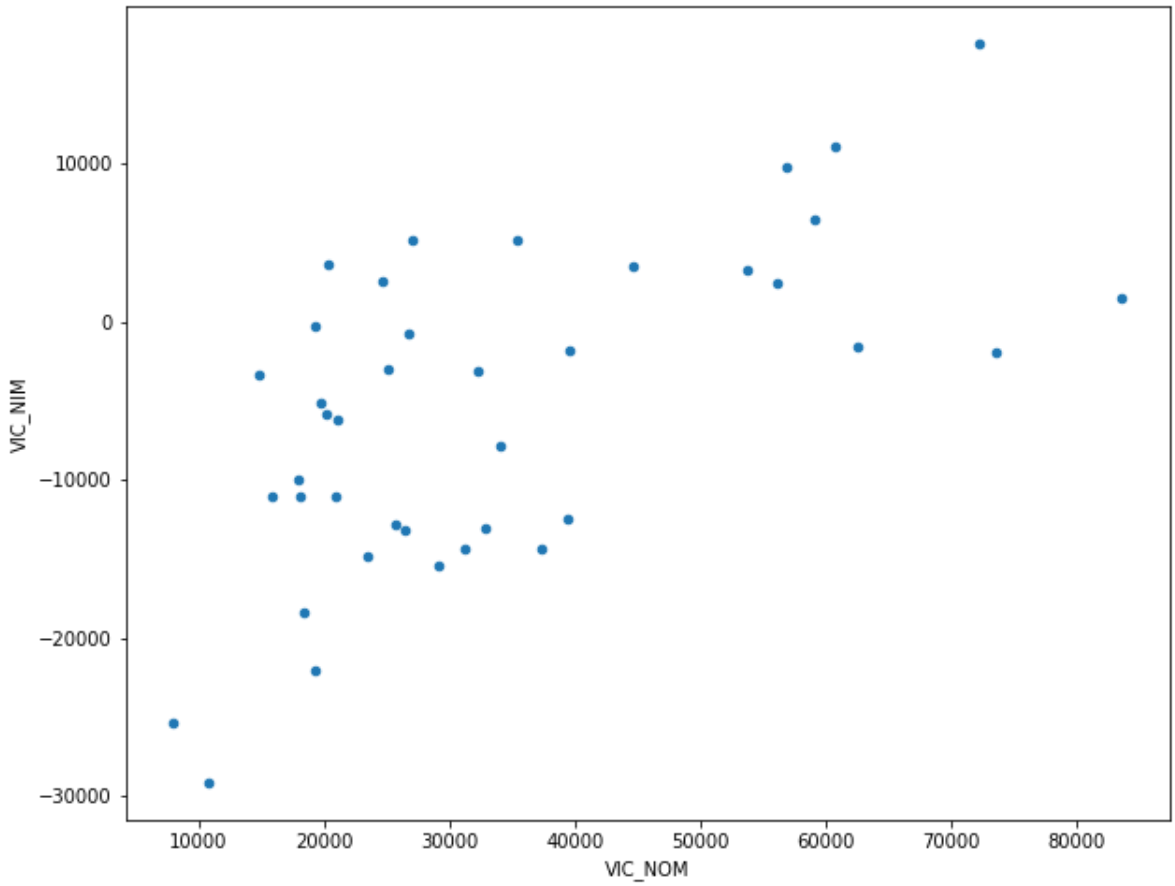


Scatterplots are typically used to determine if there is a relationship between the two variables From the above scatter plot, we can deduce that there is no correlation between two variables.

A.2.2.c

In [924]:

```
scatter_VIC = con.plot.scatter(x="VIC_NOM", y="VIC_NIM")
```



Both variables move in the same direction. In other words, as one variable increases, the other variable also increases. As one variable decreases, the other variable also decreases. I.e., VIC_NIM and VIC_NOM are positively correlated.

A.2.2.d

In [925]:

```
cont_reset = con.reset_index(drop=False)
cont_reset=cont_reset[['Year','NSW_NIM','QLD_NIM']]
cont_reset
ax = pl.gca()
cont_reset.plot(x='Year',kind='line',y='NSW_NIM',legend=True,ax=ax)
cont_reset.plot(x='Year',kind='line',y='QLD_NIM', legend=True, ax=ax)
pl.show()
```

From the above graph we can see that the hieght of the peak of QLD is nearly similar to the dip of QLD through which we can analyse that there might be a possibility that the number of people migarating from NSW to QLD.

A 3

In [926]:

```
Birth= birth.set_index('Year')
Death=pd.read_csv("Deaths.csv")
Death= Death.set_index('Year')
Fertility= fer.set_index('Year')
Interstate=pd.read_csv("NIM.csv")
Interstate= Interstate.set_index('Year')
Overstate=pd.read_csv("NOM.csv")
Overstate= Overstate.set_index('Year')
growth = Birth-Death+Interstate+Overstate
growth['Total_Population_Growth'] = growth.sum(axis=1)
Overstate['Total_NOM'] = Overstate.sum(axis=1)
Interstate['Total_NIM'] = Interstate.sum(axis=1)
Nom_final=Interstate.rename(columns={"NSW":"NSW_No","VIC":"VIC_No","QLD":"QLD_No","SA":"SA_
Nim_final=Interstate.rename(columns={"NSW":"NSW_Ni","VIC":"VIC_Ni","QLD":"QLD_Ni","SA":"SA_
B_final=Birth.rename(columns={"NSW":"NSW_B","VIC":"VIC_B","QLD":"QLD_B","SA":"SA_B","WA":"W
D_final=Death.rename(columns={"NSW":"NSW_D","VIC":"VIC_D","QLD":"QLD_D","SA":"SA_D","WA":"W
T_final=Tfr.rename(columns={"NSW":"NSW_T","VIC":"VIC_T","QLD":"QLD_T","SA":"SA_T","WA":"WA_
total_growth= pd.concat([Nom_final,Nim_final,B_final,D_final,T_final],axis=1)
total_growth['Total_Population_Growth']=growth['Total_Population_Growth']
total_growth['Total_NOM']=Overstate['Total_NOM']
total_growth['Total_NIM']=Interstate['Total_NIM']
new_total= total_growth.dropna(how='any')
print('After combining the total dataframe into a single table')
new_total.head()
```

```
After combining the total dataframe into a single table
```

Out[926]:

Year	NSW_No	VIC_No	QLD_No	SA_No	WA_No	TAS_No	NT_No	ACT_No	Total_NIM	NSW
1977	-9000.0	-10000.0	11000.0	0.0	5000.0	-1000.0	2000.0	2000.0	0.0	-90
1978	-2000.0	-11000.0	12000.0	-1500.0	1500.0	-1000.0	1500.0	500.0	0.0	-20
1979	1500.0	-11000.0	13000.0	-4000.0	1000.0	-500.0	500.0	-500.0	0.0	15
1980	-2000.0	-11000.0	17000.0	-4500.0	1500.0	-1000.0	500.0	-500.0	0.0	-20
1981	-14963.0	-15398.0	35054.0	-5109.0	2134.0	-1014.0	335.0	-1039.0	0.0	-149

5 rows × 44 columns

A3.1

In [927]:

```
from motionchart.motionchart import MotionChart
import pandas as pd
```

In [928]:

```
%%html
<style>
.output_wrapper, .output {
    height:auto !important;
    max-height:1000px;  /* your desired max-height here */
}
.output_scroll {
    box-shadow:none !important;
    webkit-box-shadow:none !important;
}
</style>
```

In [929]:

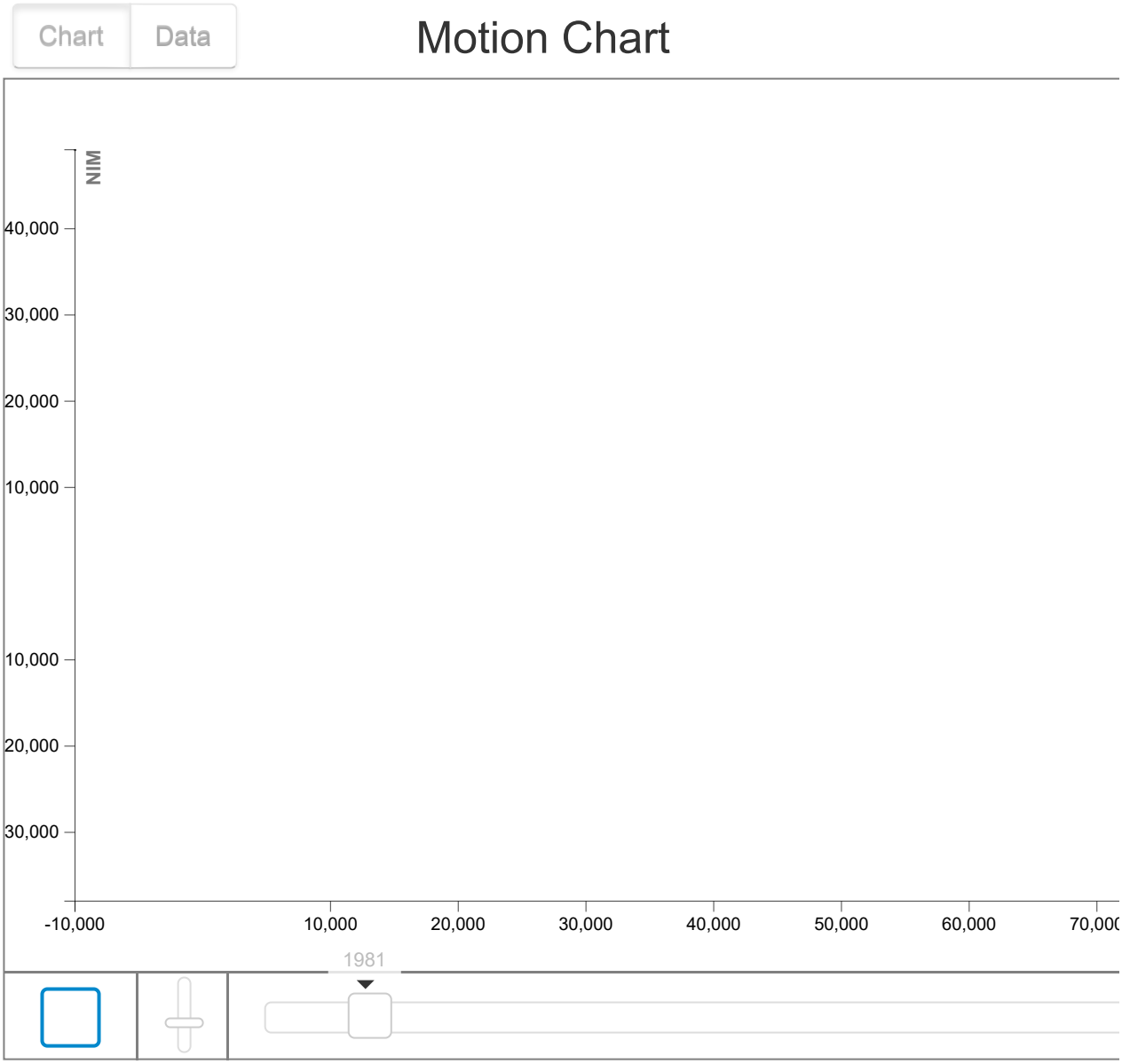
```
growth_new = Birth-Death+Nim_1+Nom_1
NIM= Nim_1.reset_index()
NIM_melted=pd.melt(NIM, id_vars =['Year'], value_vars =['ACT','NSW','NT','QLD','SA','TAS','
NOM= Nom_1.reset_index()
NOM_melted=pd.melt(NOM, id_vars =['Year'], value_vars =['ACT','NSW','NT','QLD','SA','TAS','
NOM_melted
growth_new= growth_new.reset_index()
melted_state=pd.melt(growth_new, id_vars =['Year'], value_vars =['ACT','NSW','NT','QLD','SA
melted_state.head()
melted_state['NOM']=NOM_melted['value']
melted_state['NIM']=NIM_melted['value']
melted_state.head()
```

Out[929]:

	Year	variable	value	NOM	NIM
0	1977	ACT	5856	261	2000
1	1978	ACT	4191	131	500
2	1979	ACT	2702	-256	-500
3	1980	ACT	3362	436	-500
4	1981	ACT	3191	832	-1039

In [930]:

```
mChart = MotionChart(df = melted_state, key='Year', x='NOM', y='NIM', xscale='linear', ysca
                     size='value', color='variable')

mChart.to_notebook()
```



A3.2.a

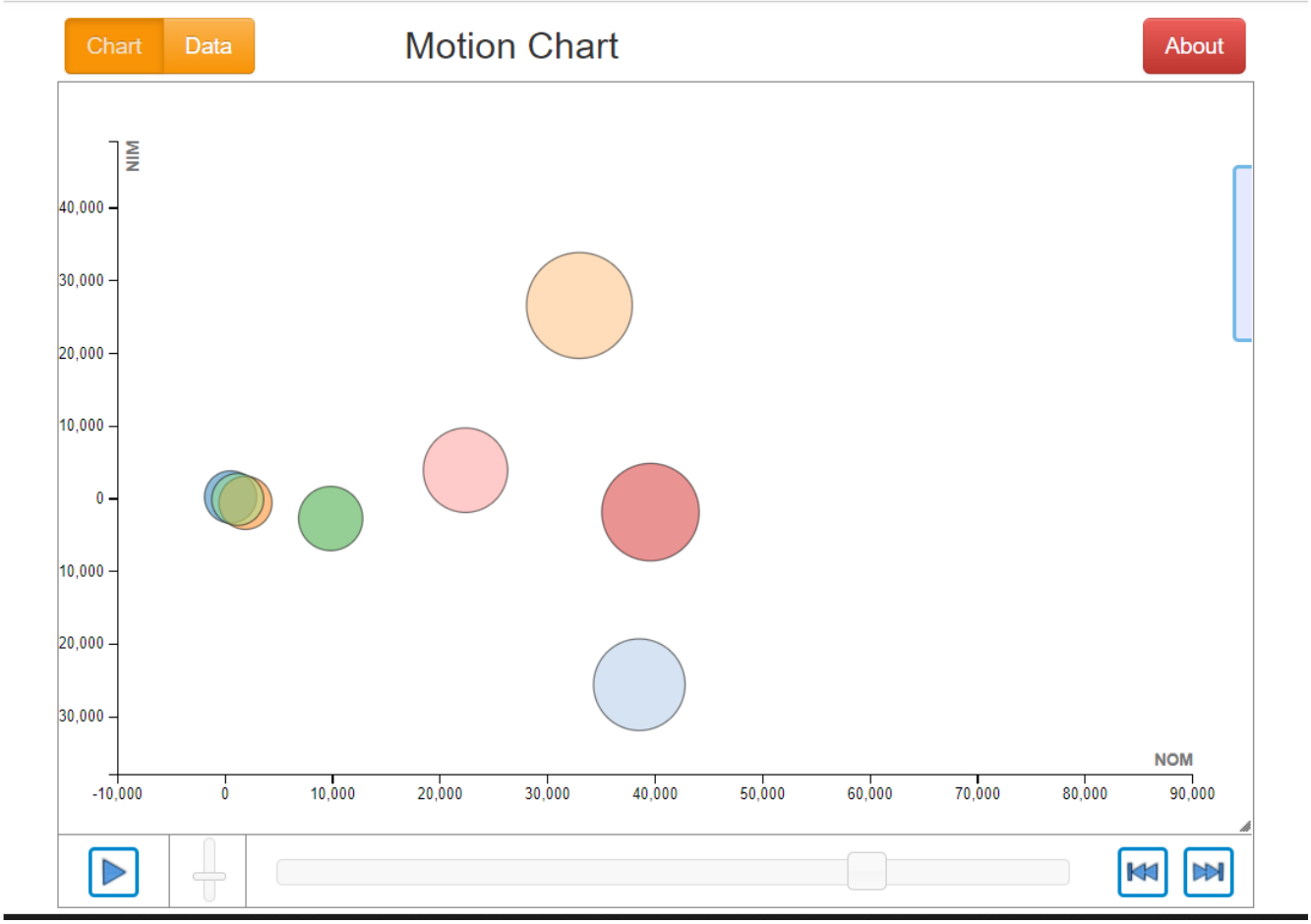
As per the motion chart we can see that there is no correlation identified between NIM and NOM as are not able to specify the linear relation between total NOM and NIM.

A3.2.b

From the year 2005 to 2006, we can observe that VIC has the highest NOM than NSW

A3.2.c

We can observe from the motion chart that the QueensLand has the highest NIM value among all.



Task B

In [931]:

```
Crime_read=pd.read_csv("Crime_Statistics_SA_2014_2019.csv")
crime=Crime_read
```

B1.1

In [932]:

```
offence=Crime_read.groupby(['Suburb - Incident','Reported Date']).sum()
# filter rows for offence greater than 15
offence_15 = offence[offence['Offence Count']>= 15]
offence_new = offence_15.groupby(['Suburb - Incident']).count()
offence_sum=offence_new.sum()
offence_sum
```

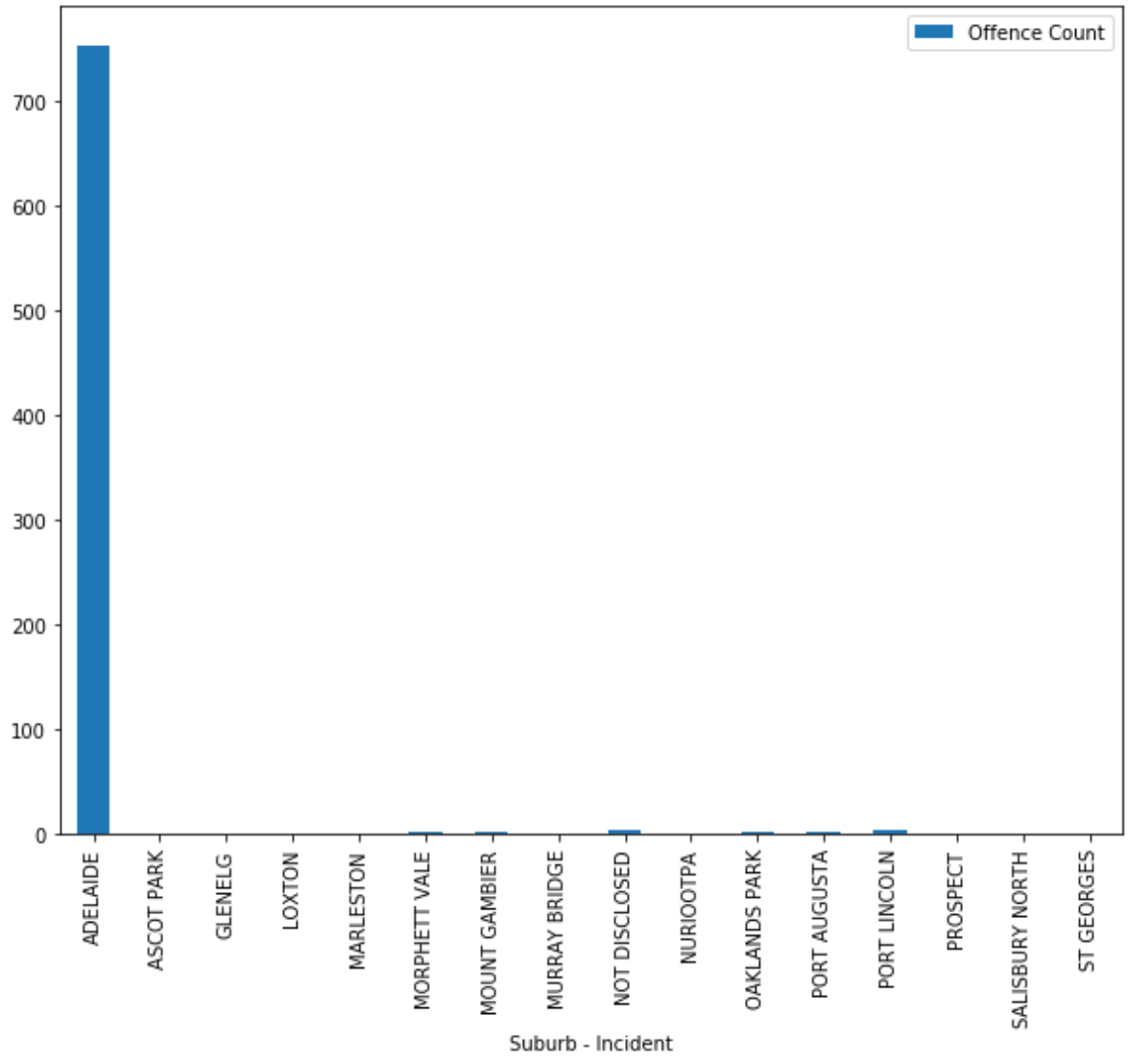
Out[932]:

```
Offence Count    919
dtype: int64
```

B1.2

In [933]:

```
#Creation of bar chart
off_more_15=Crime_read.groupby(['Suburb - Incident','Reported Date']).sum()
off_more_15 = offence[offence['Offence Count']> 15]
off_15= off_more_15.groupby(['Suburb - Incident']).count()
off_grp=pd.DataFrame(off_15)
ax = off_grp.plot.bar(rot=0)
pl.xticks(rotation = 90)
pl.show()
```



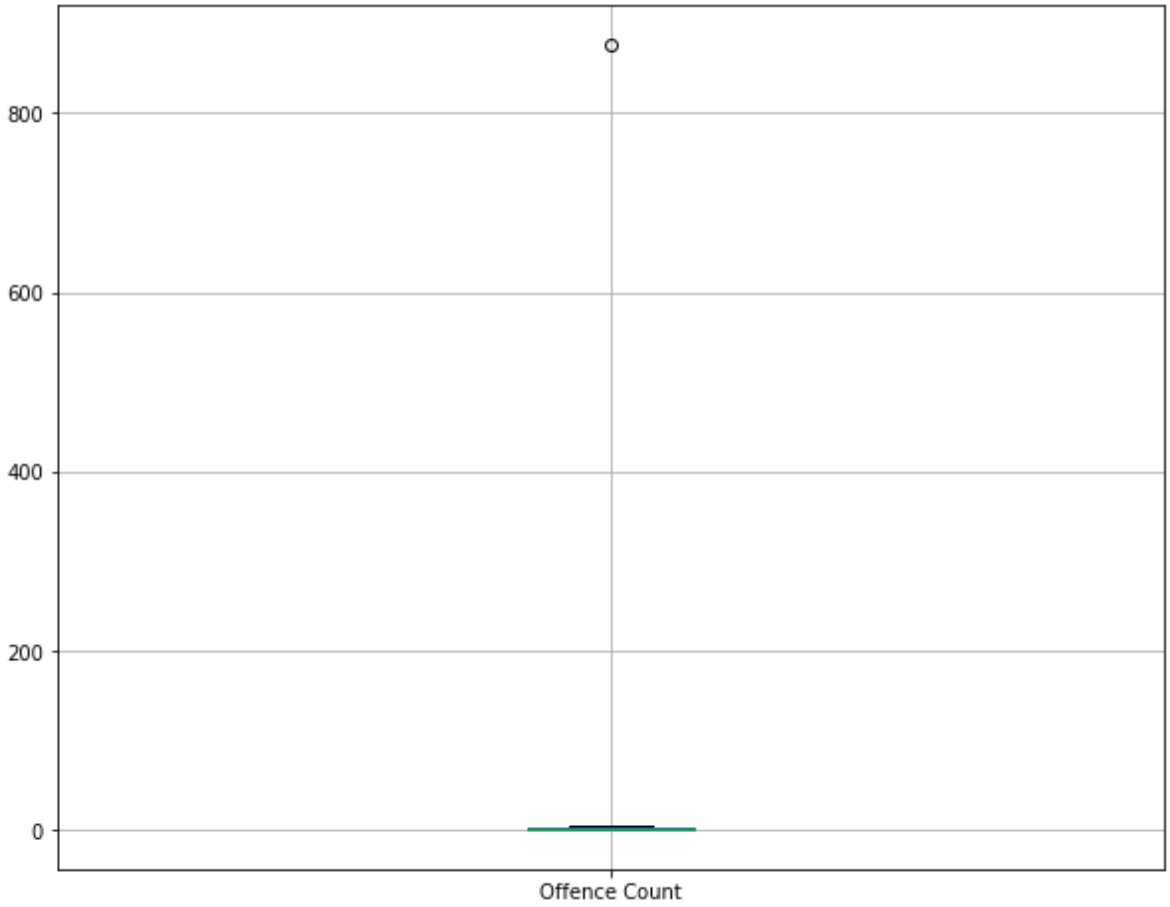
B1.3

In [934]:

```
#As per using the appropriate graph that could detect the outliers, we are plotting the gra
boxplot=offence_new.boxplot()
boxplot
```

Out[934]:

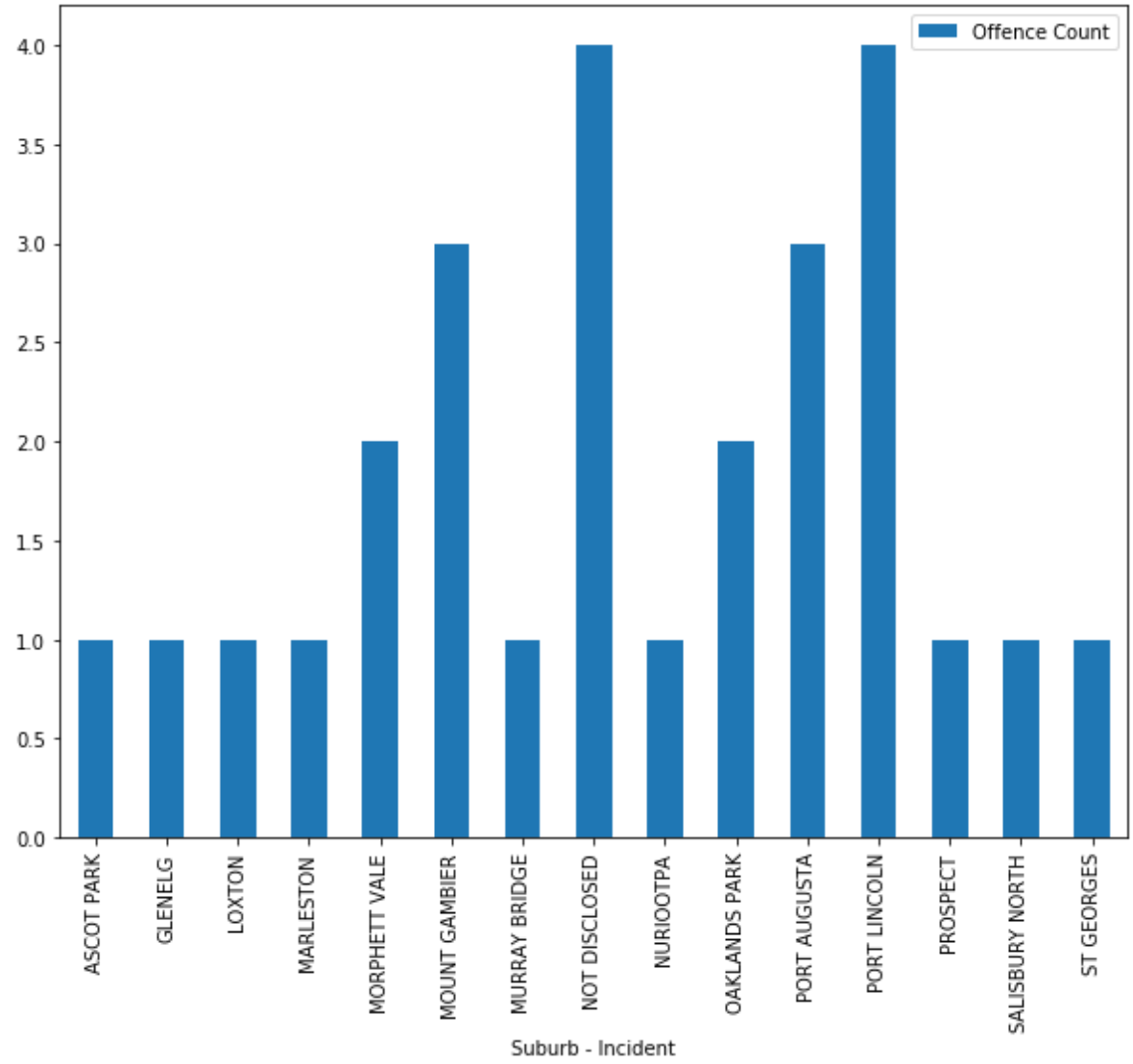
```
<matplotlib.axes._subplots.AxesSubplot at 0x23f03431da0>
```



After the visualization we could identify that Adelaide to be the Suburb with the extreme value. Hence, we are dropping the Adelaide from the dataframe and plotting it again in bar chart and box plot

In [935]:

```
#Dropping Adelaide and plotting bar graph and box plot
off_drop = off_gr.drop(['ADELAIDE'])
ax=off_drop.plot.bar(rot=0)
pl.xticks(rotation = 90)
pl.show()
```



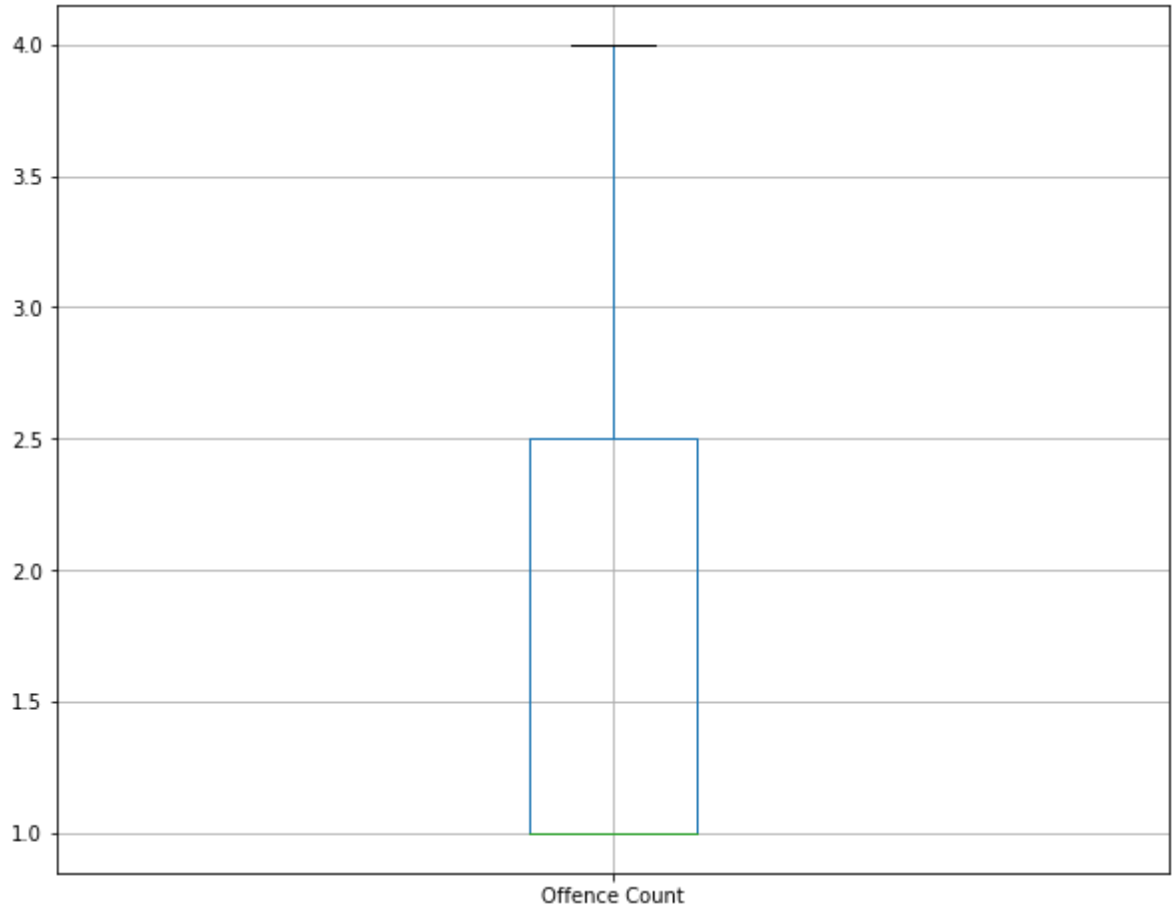
After dropping Adelaide,the interpretebility of the bar chart increases as well as analysing the data becomes easier

In [936]:

```
boxplot = off_drop.boxplot()
boxplot
```

Out[936]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f63d42da0>
```



For the box plot,we can see that for all the offense count the hihest frequecy is 17 as per the median value of the box plot.

B1.4

Compare the bar graphs in step 2 and 3. Which bar graph is easier to interpret? Why?

From the above two plotting we could identify plot 3 to be easily interpreted.As the outliar has the huge value, so after removing the outlier we could easily analyse the graph, which altogether makes it easy to interpret.

Type *Markdown* and LaTeX: α^2

B2

In [937]:

```
crime_new=Crime_read['Offence Level 3 Description'].unique().tolist()
repeated_val=Crime_read['Offence Level 3 Description']
repeat1=repeated_val[2]
repeat2=repeated_val[385278]
print("Redundant value>>>",repeat1,"and",repeat2)
```

```
Redundant value>>> Assault Police and Assault police
```

In [938]:

```
val_count=Crime_read['Offence Level 3 Description'].value_counts()
crime_new
one=val_count[21]
two=val_count[26]
print("Redundant value count for Assault Police and Assault police>>>",one,"and",two)
```

```
Redundant value count for Assault Police and Assault police>>> 2226 and 480
```

As per the above output, we could analyse that in the coulumn 'Offence Level 3 Description', there are two data entry for Assault Police; i.e 'Assault Police', 'Assault police' which is a repetetion.

Second error

Suburb incident and Postcode have null values. From below we are replacing na values from the dataframe.

In [939]:

```
new_total= tot.dropna(how='any')
```

We have corrected the values in the dataframe by replcaing entry values of 'Assault police' to 'Assault Police'.After correcting the error we cold see that both Assault Police and Assault police are now grouped together in single dataframe

In [940]:

```
#correcting the first error and replcaing the values
correct_crime=Crime_read.replace('Assault police', 'Assault Police')
Assault_count=correct_crime['Offence Level 3 Description'].value_counts()
print("After grouping the redundant value for Assault Police",Assault_count[17])
```

```
After grouping the redundant value for Assault Police 2706
```

In [941]:

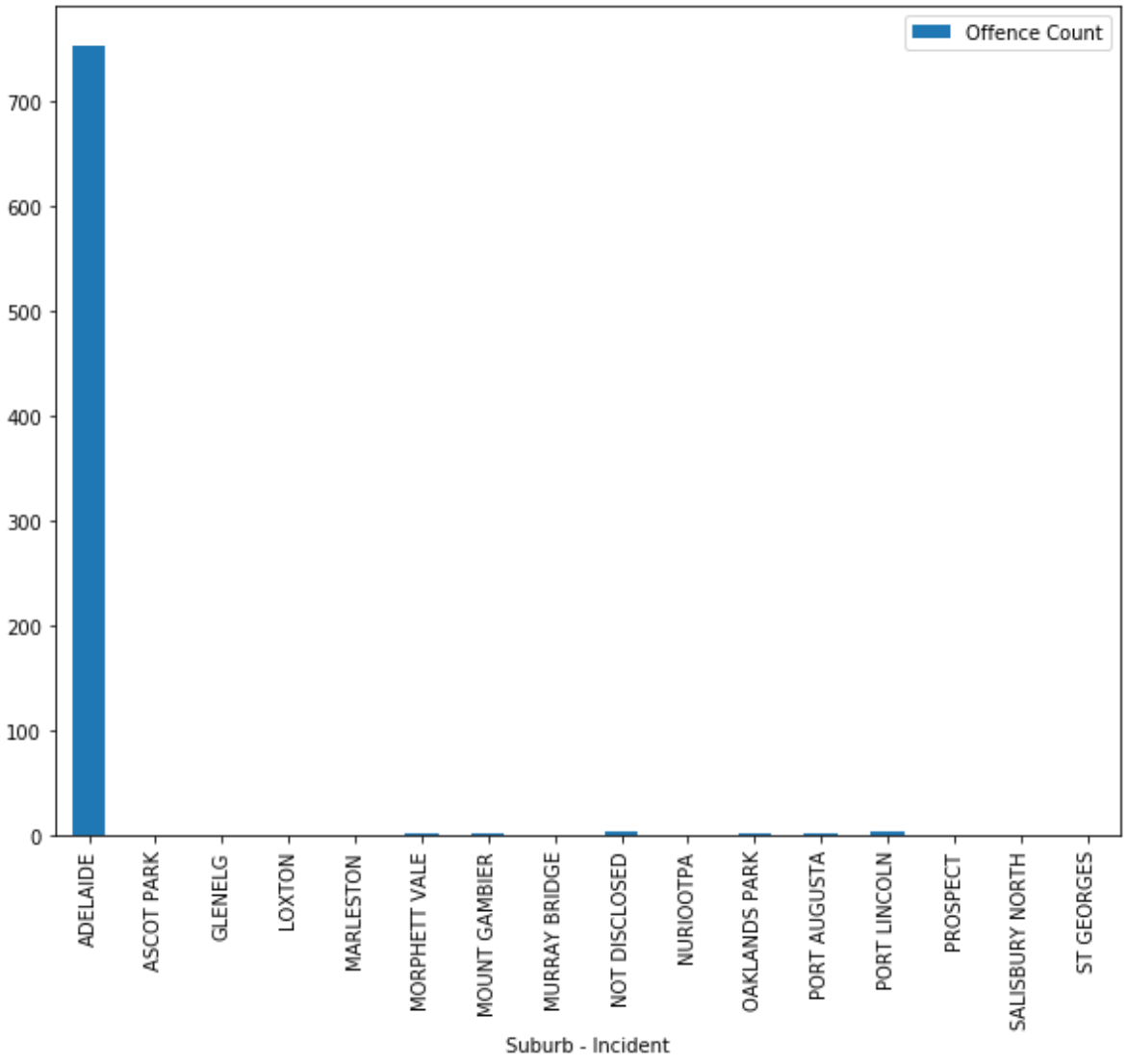
```
# Checking for B1.1
fixed=correct_crime.groupby(['Suburb - Incident','Reported Date']).sum()
# filter rows for offence greater than 15
fixed_15 = fixed[fixed['Offence Count']>= 15]
fixed_new = fixed_15.groupby(['Suburb - Incident']).count()
fixed_new.head()
```

Out[941]:

Suburb - Incident	Offence Count
ADELAIDE	877
ASCOT PARK	1
DAVOREN PARK	1
FINDON	1
GLENELG	1

In [942]:

```
#Creation of bar chart
off_fix=correct_crime.groupby(['Suburb - Incident','Reported Date']).sum()
fix_15 = off_fix[off_fix['Offence Count']> 15]
fix_15= fix_15.groupby(['Suburb - Incident']).count()
off_final=pd.DataFrame(fix_15)
ax = off_final.plot.bar(rot=0)
pl.xticks(rotation = 90)
pl.show()
```

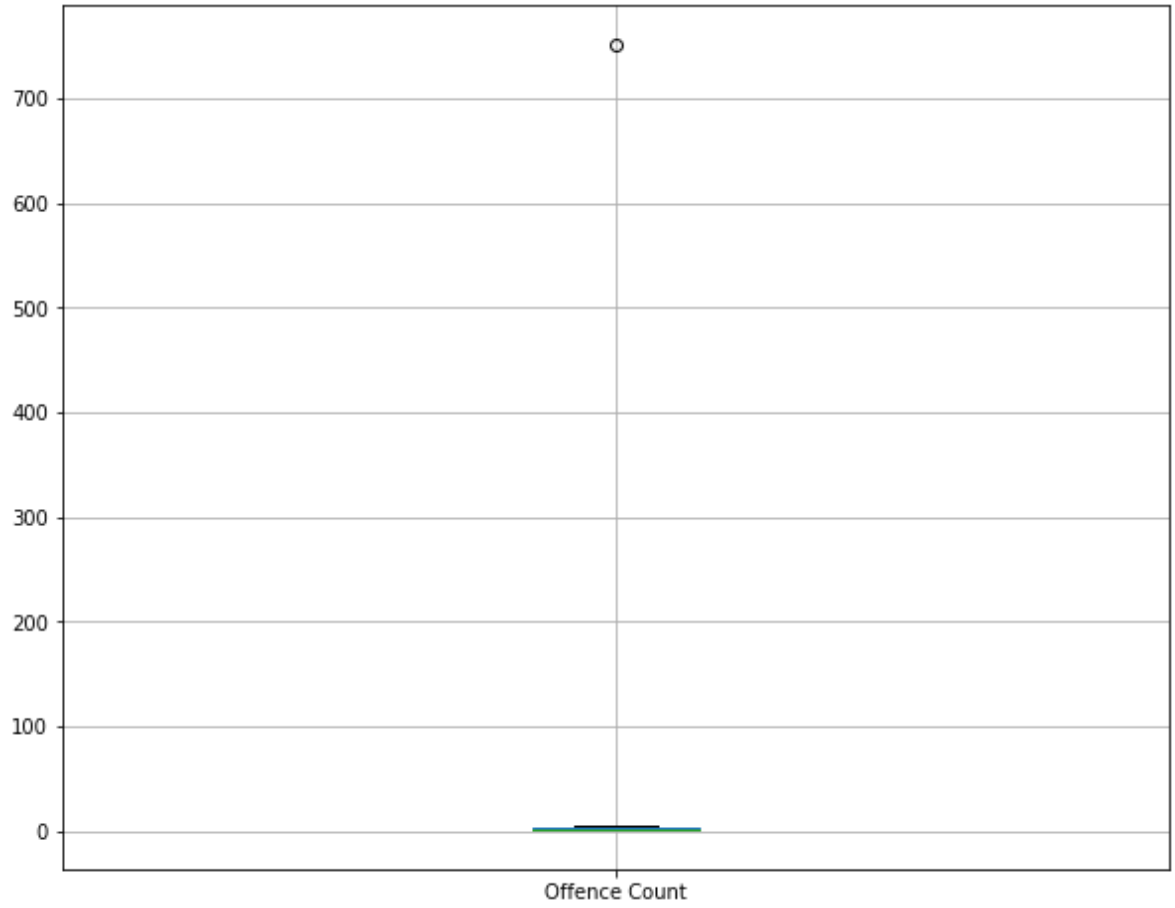


In [943]:

```
off_final.boxplot()
```

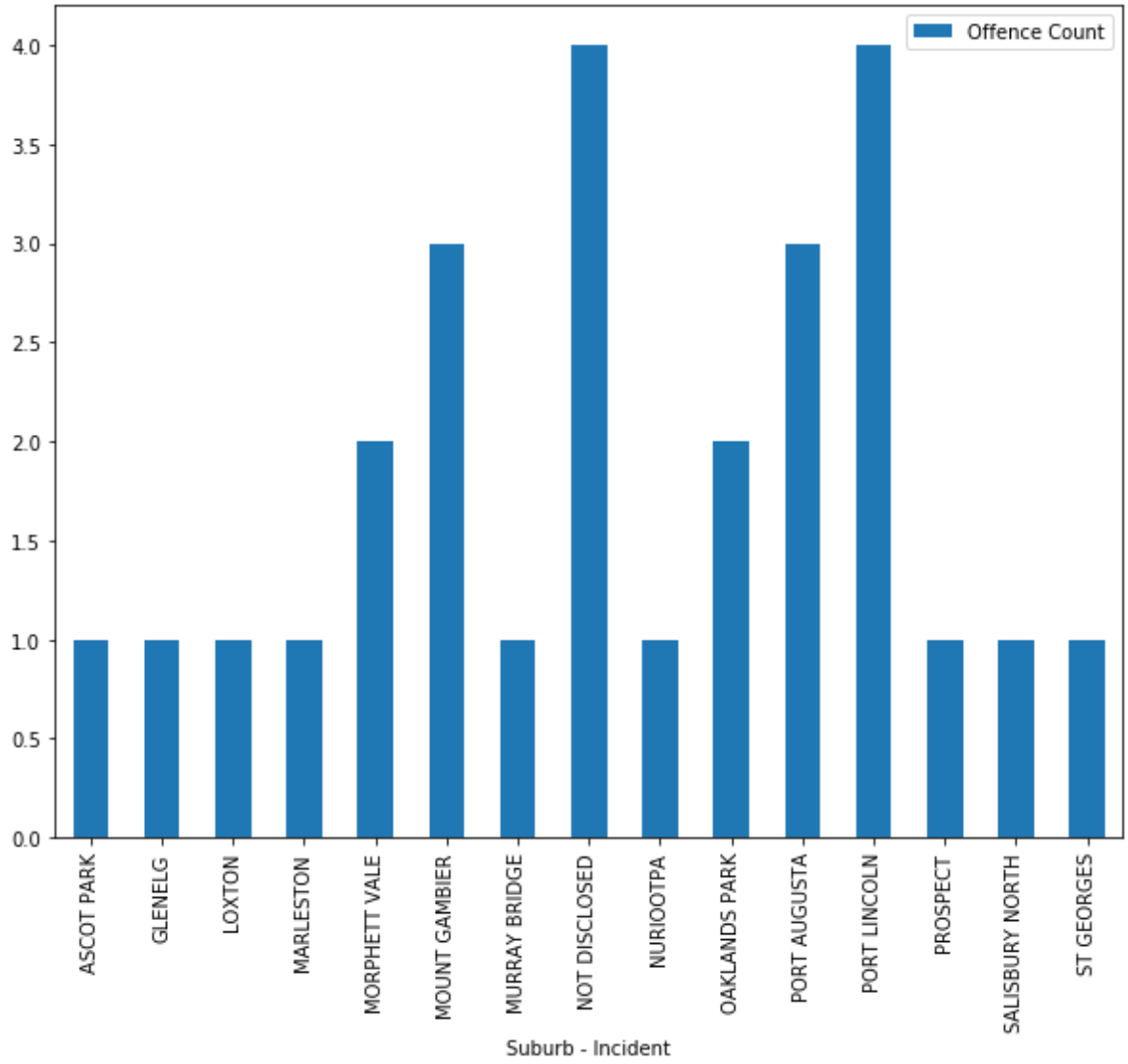
Out[943]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f63d20128>
```



In [944]:

```
#Dropping Adelaide and plotting bar graph and box plot
off = off_final.drop(['ADELAIDE'])
ax=off.plot.bar(rot=0)
pl.xticks(rotation = 90)
pl.show()
```

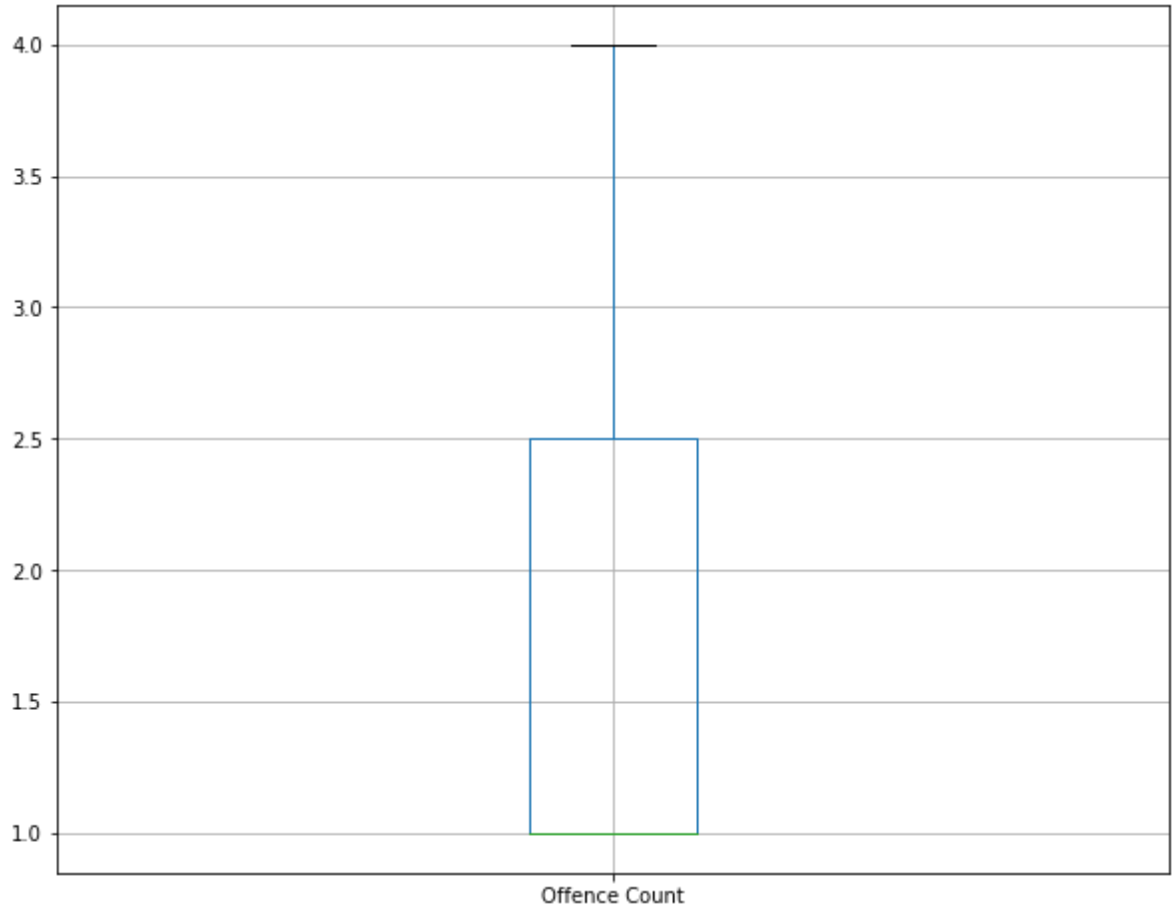


In [945]:

```
boxplot = off_drop.boxplot()
boxplot
```

Out[945]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f6bca5cf8>
```



As per repeating the above task of B1, we could analyse that none of the graph is been affected after fixing the errors.

Task C

In [946]:

```
import seaborn as sea
bos=pd.read_csv("Crime_Boston.csv",header=0,encoding = 'unicode_escape')
#basic information of boston crime dataframe
bos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 16 columns):
INCIDENT_NUMBER        299 non-null object
OFFENSE_CODE           299 non-null int64
OFFENSE_CODE_GROUP     299 non-null object
OFFENSE_DESCRIPTION    299 non-null object
DISTRICT               283 non-null object
REPORTING_AREA         299 non-null object
OCCURRED_ON_DATE       299 non-null object
YEAR                   299 non-null int64
MONTH                  299 non-null int64
DAY_OF_WEEK            299 non-null object
HOUR                   299 non-null int64
UCR_PART               299 non-null object
STREET                 284 non-null object
Lat                    292 non-null float64
Long                   292 non-null float64
Location               299 non-null object
dtypes: float64(2), int64(4), object(10)
memory usage: 37.5+ KB
```

In [947]:

```
#Displaying the first 5 rows of the dataset
bos.head()
```

Out[947]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DI
0	I182070945	619	Larceny	LARCENY ALL OTHERS	
1	I182070943	1402	Vandalism	VANDALISM	
2	I182070941	3410	Towed	TOWED MOTOR VEHICLE	
3	I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	
4	I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	

In [948]:

```
# Top 10 safe streets in boston

safe_zone = bos.groupby([bos['STREET'].fillna('NO STREET NAME')])['REPORTING_AREA'].aggrega
safe_zone
```

Out[948]:

	STREET	REPORTING_AREA
0	ADAMS ST	1
110	LEXINGTON AVE	1
111	LINCOLN ST	1
112	LYNDHURST ST	1
113	LYNNVILLE TER	1
114	MARLBOROUGH ST	1
115	MARTHA RD	1
116	MASCOT ST	1
118	METROPOLITAN AVE	1
119	MILK ST	1

As per the data interpretation,this shows the top 10 streets of crime occurance in Boston. We can see that lowest crime occurance is in around the rural areas or the outskirts of the city.

In [949]:

```
# Top 10 Offense types
off_type = bos.groupby([bos['OFFENSE_CODE_GROUP']])['STREET'].aggregate(np.size).reset_inde
off_type
```

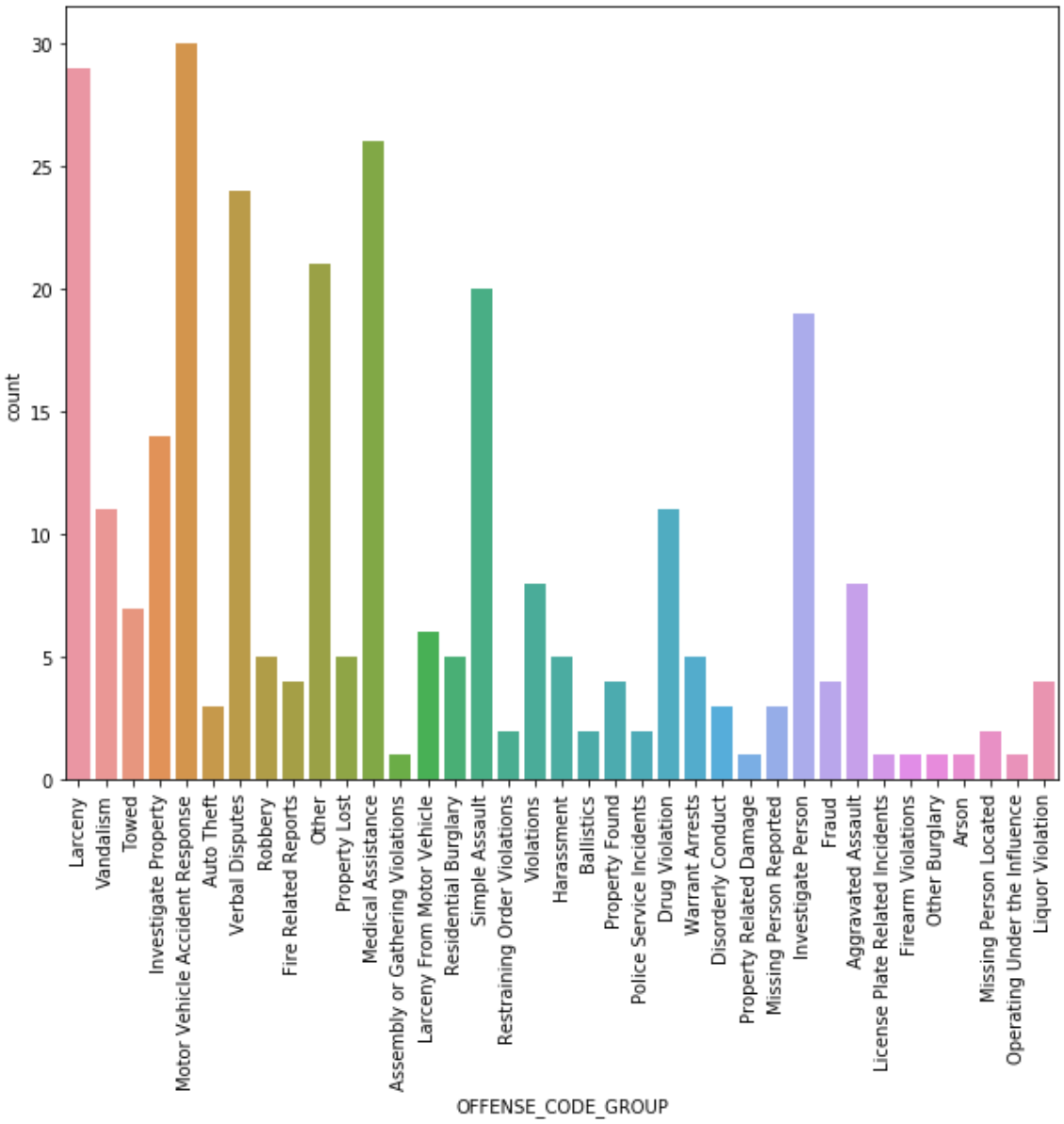
Out[949]:

	OFFENSE_CODE_GROUP	STREET
20	Motor Vehicle Accident Response	30
13	Larceny	29
17	Medical Assistance	26
34	Verbal Disputes	24
22	Other	21
31	Simple Assault	20
11	Investigate Person	19
12	Investigate Property	14
33	Vandalism	11
6	Drug Violation	11

As per the data interpretation,this shows the top 10 OFFENSE type of crime occurance in Boston. We can see that highest crime occurance group is Motor Vehicle Accident Response and Larceny. And from the studies we observed that the Boston Police has to majorly deal with such issues during the year of 2018 and 2017. Now lets plot a countplot on our data interpretation.

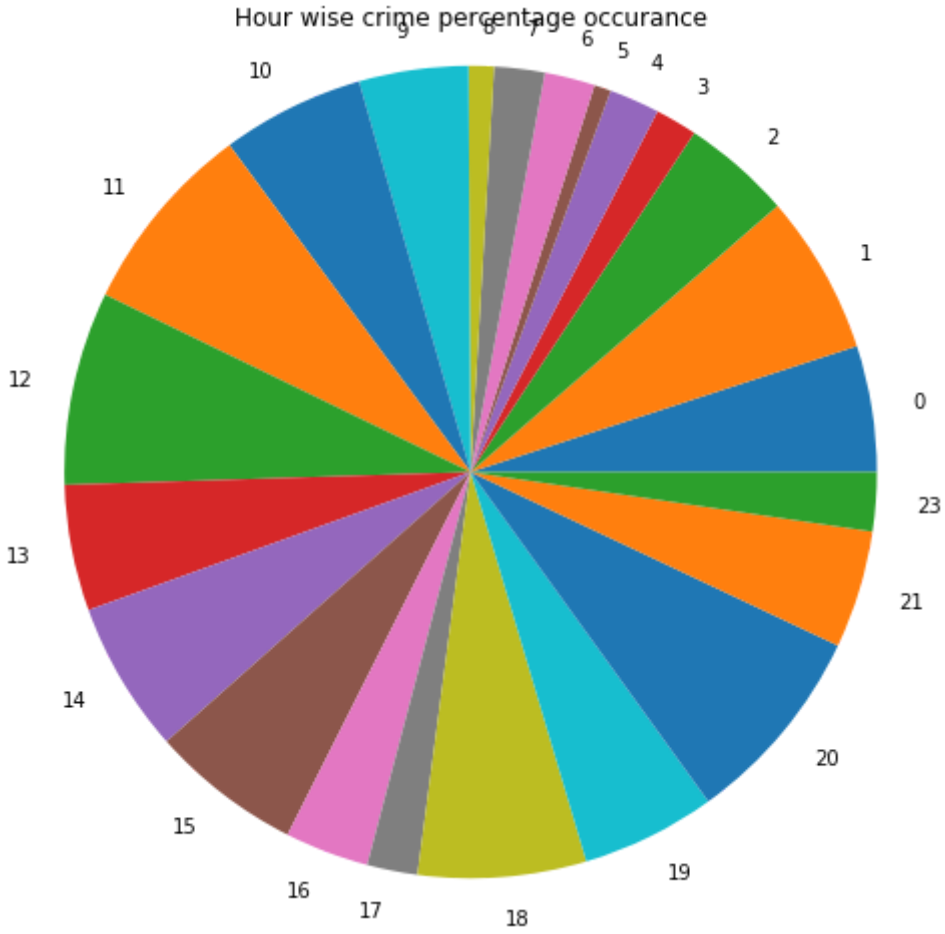
In [950]:

```
#Plotting the count of OFFENSE_CODE_GROUP when the crime is occuring the highest
plot= sea.countplot(x="OFFENSE_CODE_GROUP", data=bos)
pl.xticks(rotation = 90)
pl.show()
```



In [951]:

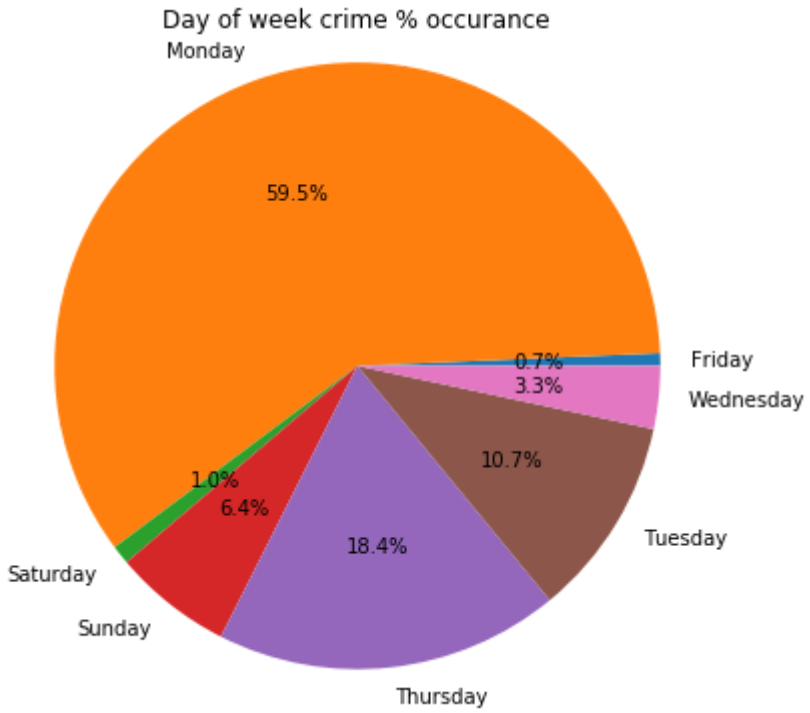
```
# Year wise percentage rate
year = bos['HOUR'].astype('category').cat.categories.tolist()
year_count = bos['HOUR'].value_counts()
sizes = [year_count[year] for year in year]
fig1, ax1 = pl.subplots(figsize=(8,8))
ax1.pie(sizes, labels=year)
ax1.axis('equal')
ax1.set_title("Hour wise crime percentage occurance")
pl.show()
```



Here we explore the possibility that seasonality affects the rate at which crime is committed per hour. From the above plot we could identify that as per the comparsion of crime rate with the hour we could see, that crime attempt were maximum during 21:00,13:00 or 14:00 as compared to the others. As per the studies published, shack shake to pedestrian safety(Vision Zero Boston) Boston seems to get better year after year. Vision Zero Boston is Cities commitment to focus the City's resources on proven strategies to eliminate fatal and serious traffic crashes in the City by 2030, that rate Boston as tops in the nation.

In [952]:

```
# Day of the week crime %
dayofwkcount = bos['DAY_OF_WEEK'].value_counts()
dayofwk = bos['DAY_OF_WEEK'].astype('category').cat.categories.tolist()
sizes = [dayofwkcount[dow] for dow in dayofwk]
fig1, ax1 = pl.subplots(figsize=(6,6))
ax1.pie(sizes, labels=dayofwk,  autopct='%1.1f%%',shadow=False)
ax1.axis('equal')
ax1.set_title("Day of week crime % occurance")
pl.show()
```



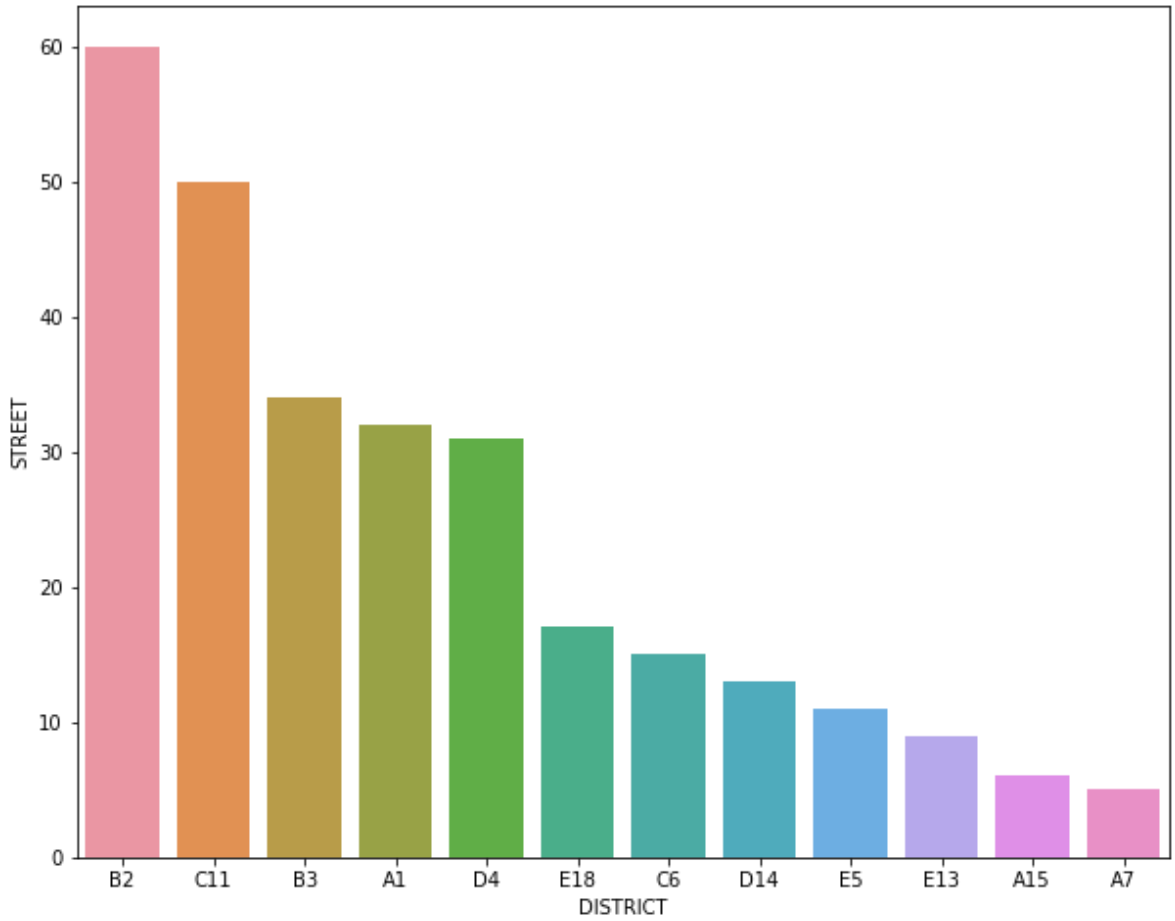
We could identify that highest percentage of crime takes place on monday as comapred to rest of the week.

In [953]:

```
# Count of crimes - reporting district wise

crimedistrict = bos.groupby([bos['DISTRICT']])['STREET'].aggregate(np.size).reset_index().s
sns.barplot(x="DISTRICT", y="STREET", data = crimedistrict)
```

Out[953]:

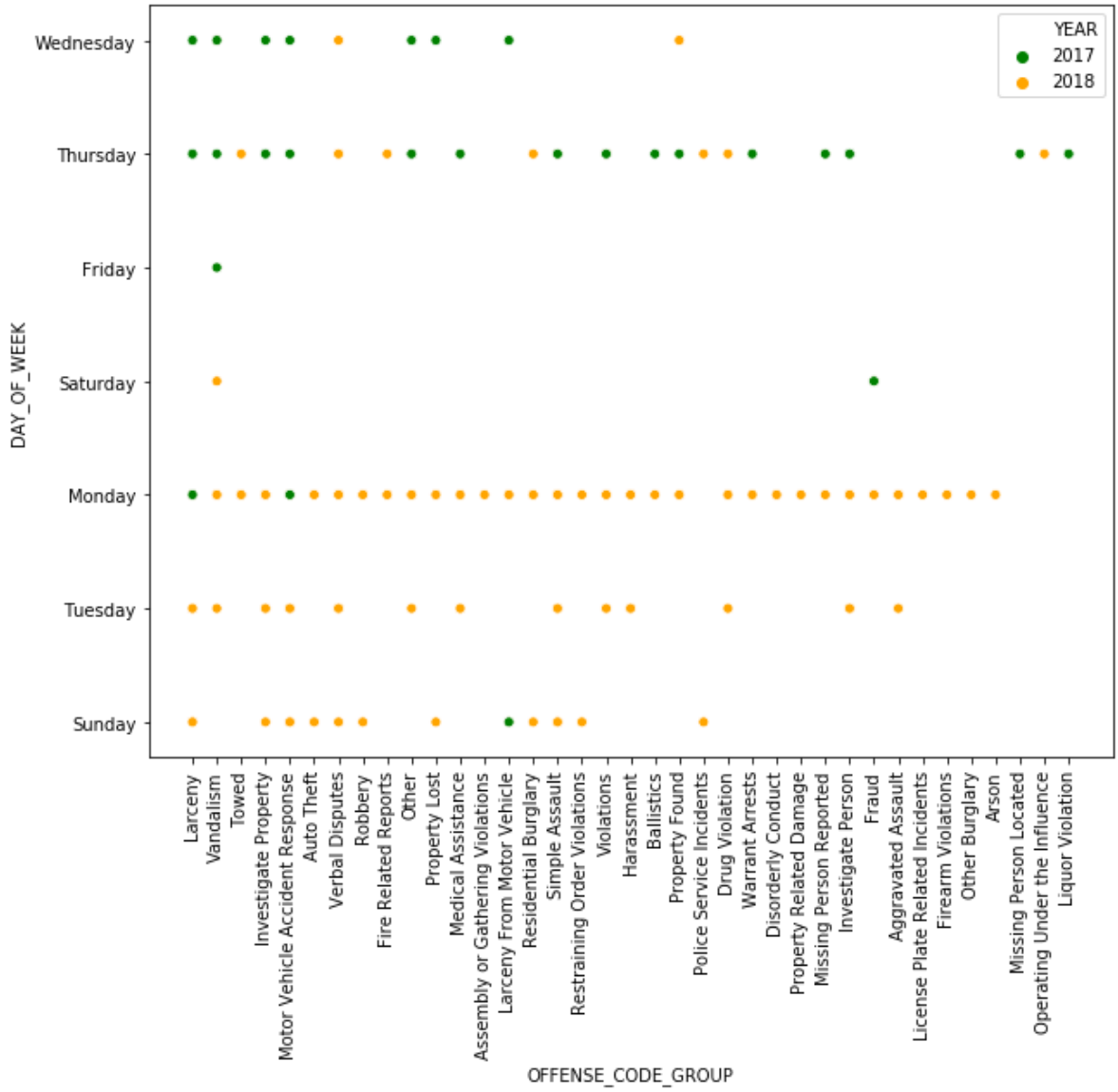
```
<matplotlib.axes._subplots.AxesSubplot at 0x23f71353668>
```



As per the different districts, we can see B2 as highly affected by the crime. As per the studies we came across B2 to be Boston Police District B-2 Roxbury, which comes under Washington St and as per the above analysis we identifies Washington St to be one to street where the offense count is higher.

In [954]:

```
plot=sea.scatterplot(x="OFFENSE_CODE_GROUP",y="DAY_OF_WEEK",hue="YEAR",palette=['green','or
pl.xticks(rotation = 90)
pl.show()
```



It can be visualized from the plot, that during year 2018 the occurance of different offence group is majorly on Mondays except Drug Violations and Police Service Incidents. While in year 2017, the occurance of crime days changed to Wednesday and Thrusday.Plus we can also see the occurance of crime has decreased as compared to 2018.

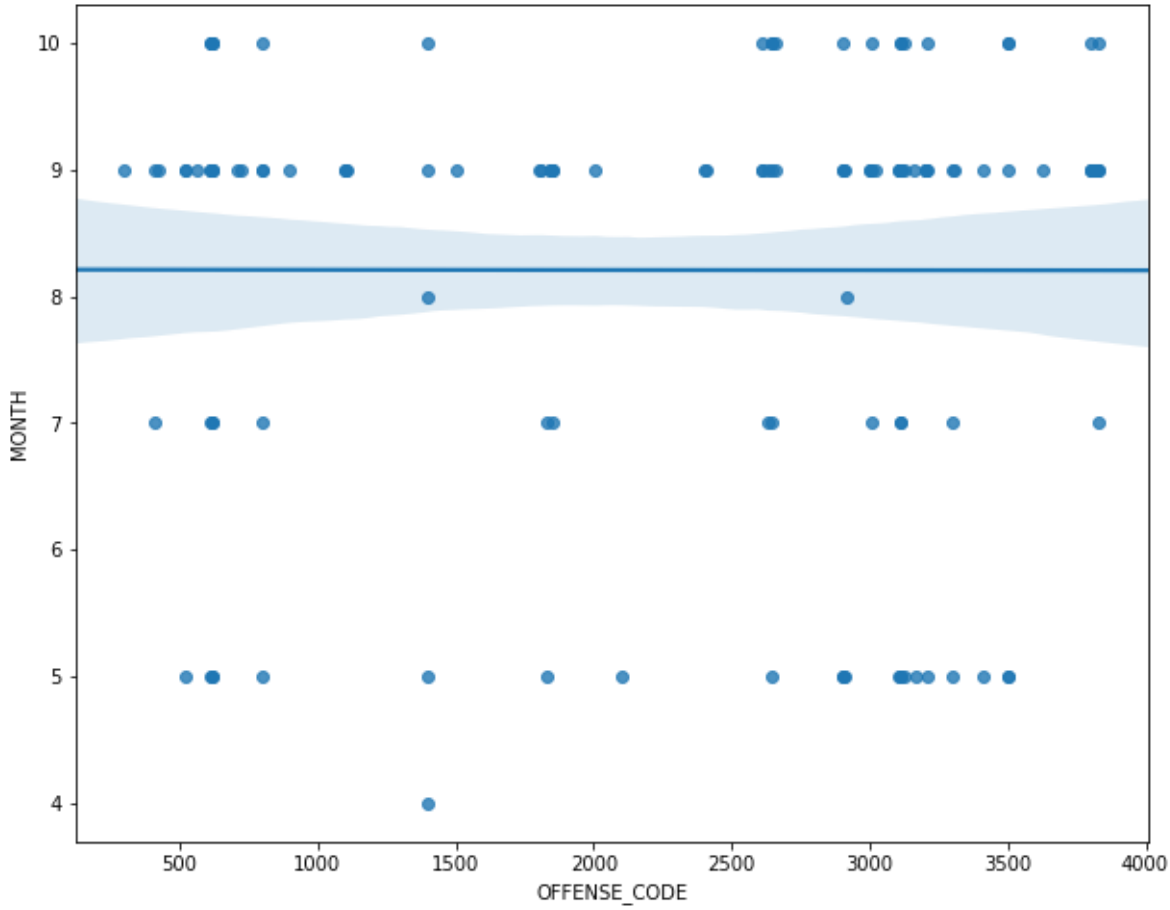
Let us draw a relationship between offence_code,month and hour

In [955]:

```
bostonPlot=bos.groupby(['OFFENSE_CODE','MONTH','YEAR']).sum()
bostonPlot.head()
b=bostonPlot.reset_index()
plot=sea.regplot(x="OFFENSE_CODE",y="MONTH",data=b)
labx=b[['OFFENSE_CODE']]
laby=b[['MONTH']]
reg.fit(labx,laby)
reg.predict([[1402],[619]])
```

Out[955]:

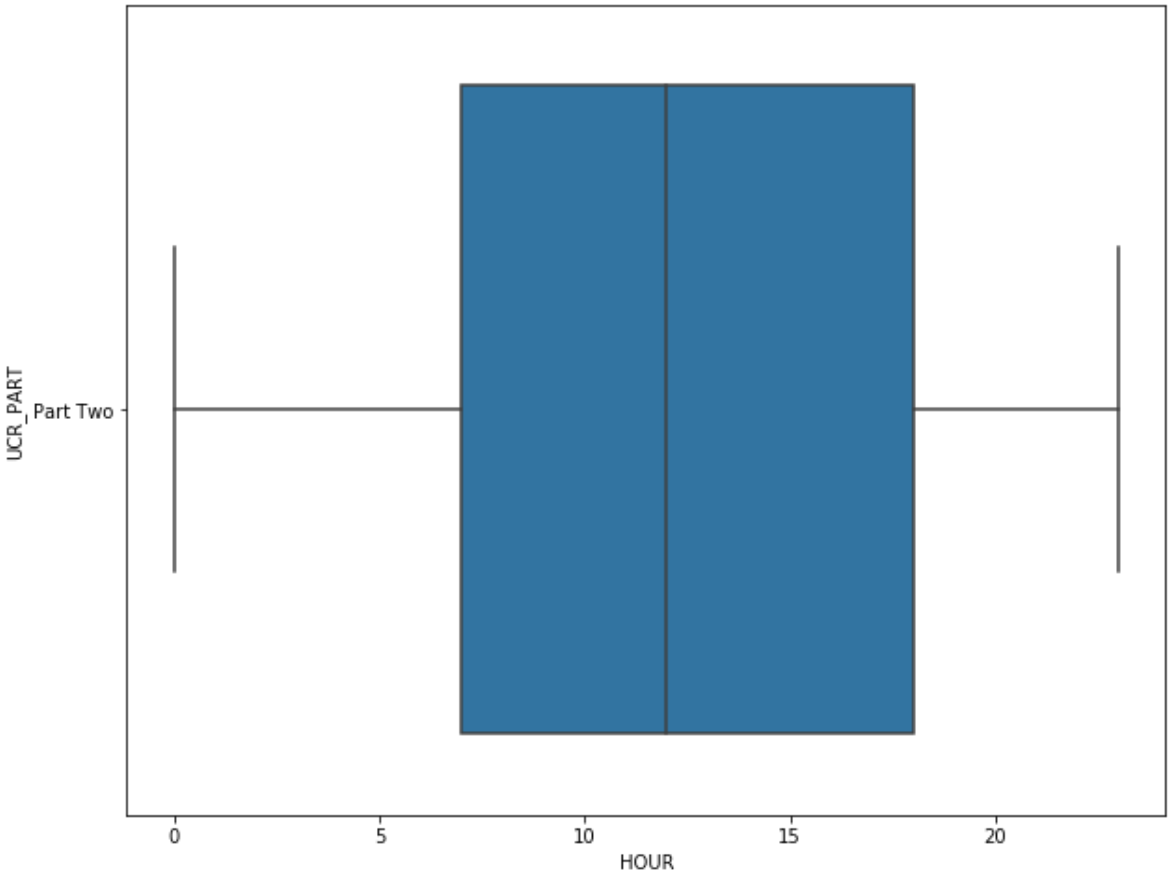
```
array([[8.21053687],
       [8.2116893 ]])
```



From the above graph we can deduce that Offense code 1402() and 619(), have the highest probablity of occuring in the month of August

In [956]:

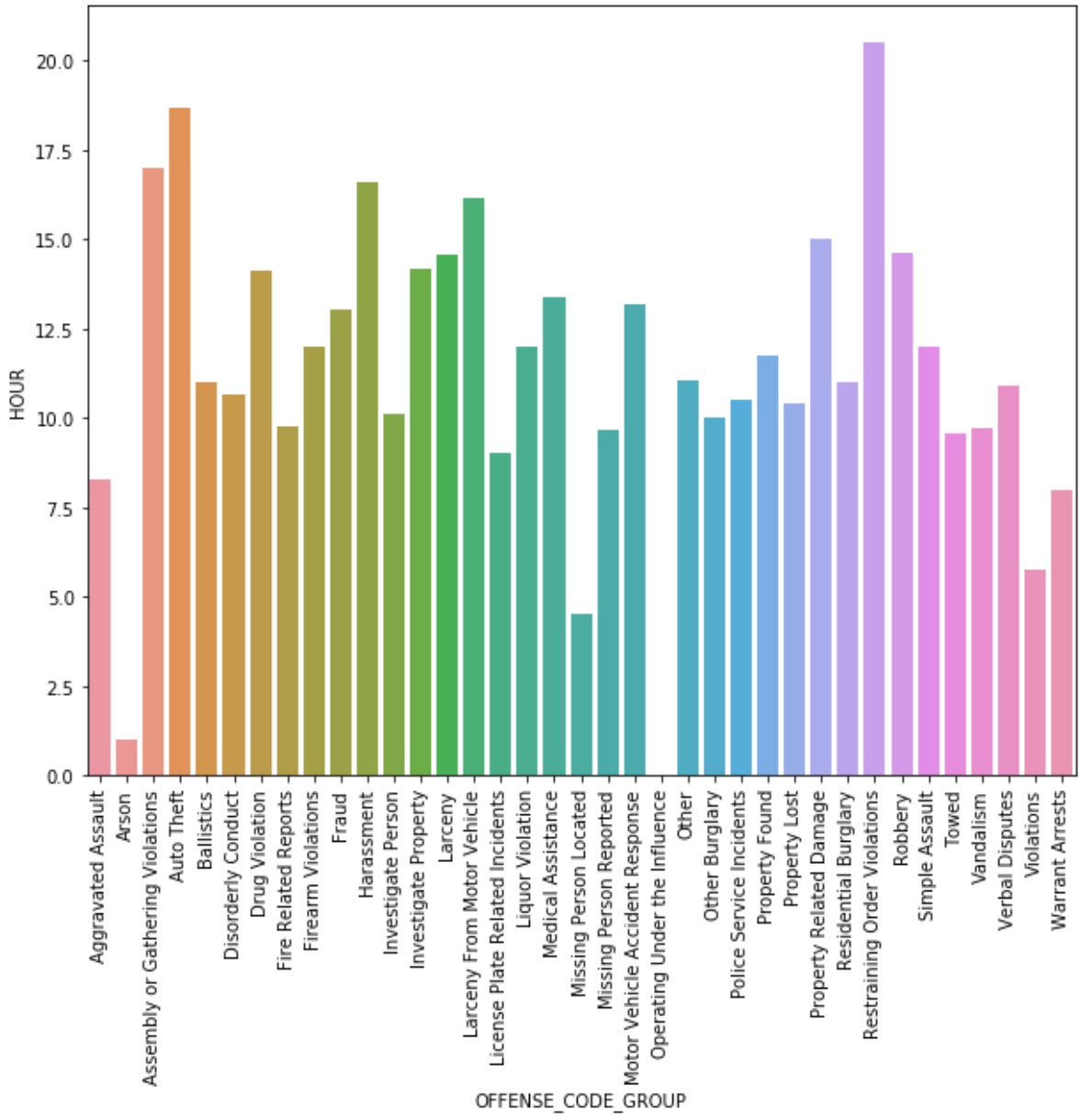
```
#Correlation between UCR_PART ="Part Two" and Hour
boss=(bos['UCR_PART']=='Part Two')
plot = sea.boxplot(y="UCR_PART",x="HOUR",data=bos[boss])
```



We can deduce from the above plot that duration of the attempting the crime of UCR_PART two which includes Forgery and counterfeiting,Vandalism, Fraud, Prostitution and commercialized vice as wel as Sex offenses that is in range of 6 to 18 hours. Which makes it high alert time in the areas where these crime occurs mostly.

In [957]:

```
# Calculating the mean chance of hour in relation to the Offrnce_code_group using the group
boston = bos.groupby(['OFFENSE_CODE_GROUP'])['HOUR'].mean().reset_index(name="HOUR")

# Plotting the data using bar graph
plot = sea.barplot(data=boston , x="OFFENSE_CODE_GROUP",y="HOUR")
pl.xticks(rotation = 90)
pl.show()
```

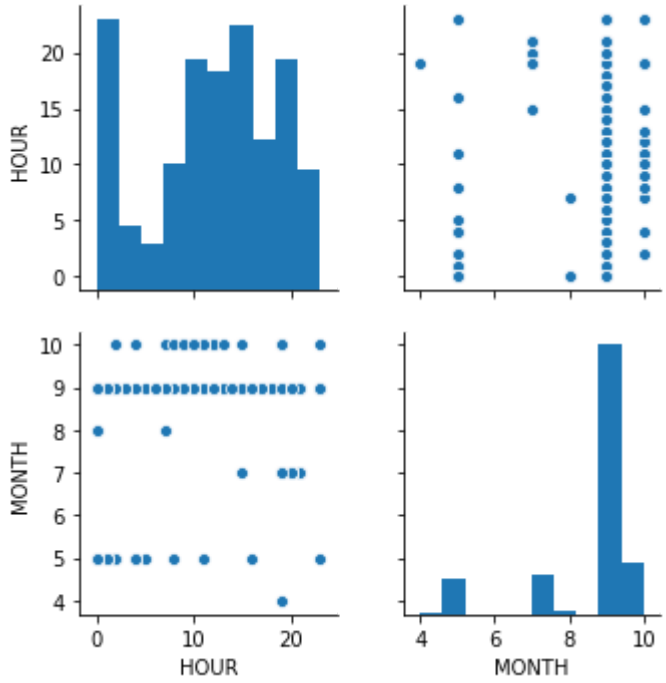


From the above graph, we could see the occurance of crime hours is average 11. While the offence group for

Restraining Order Violations, is the highest throughout the year 2017 to 2018. Through this we could visualize the major occurance and type of crime occured in Boston, which alerts the Boston Security for the worst.

In [958]:

```
#Showing interrelationship by distribution between the given parameters(In which month the
plot= sea.pairplot(bos,vars=['HOUR','MONTH'])
```

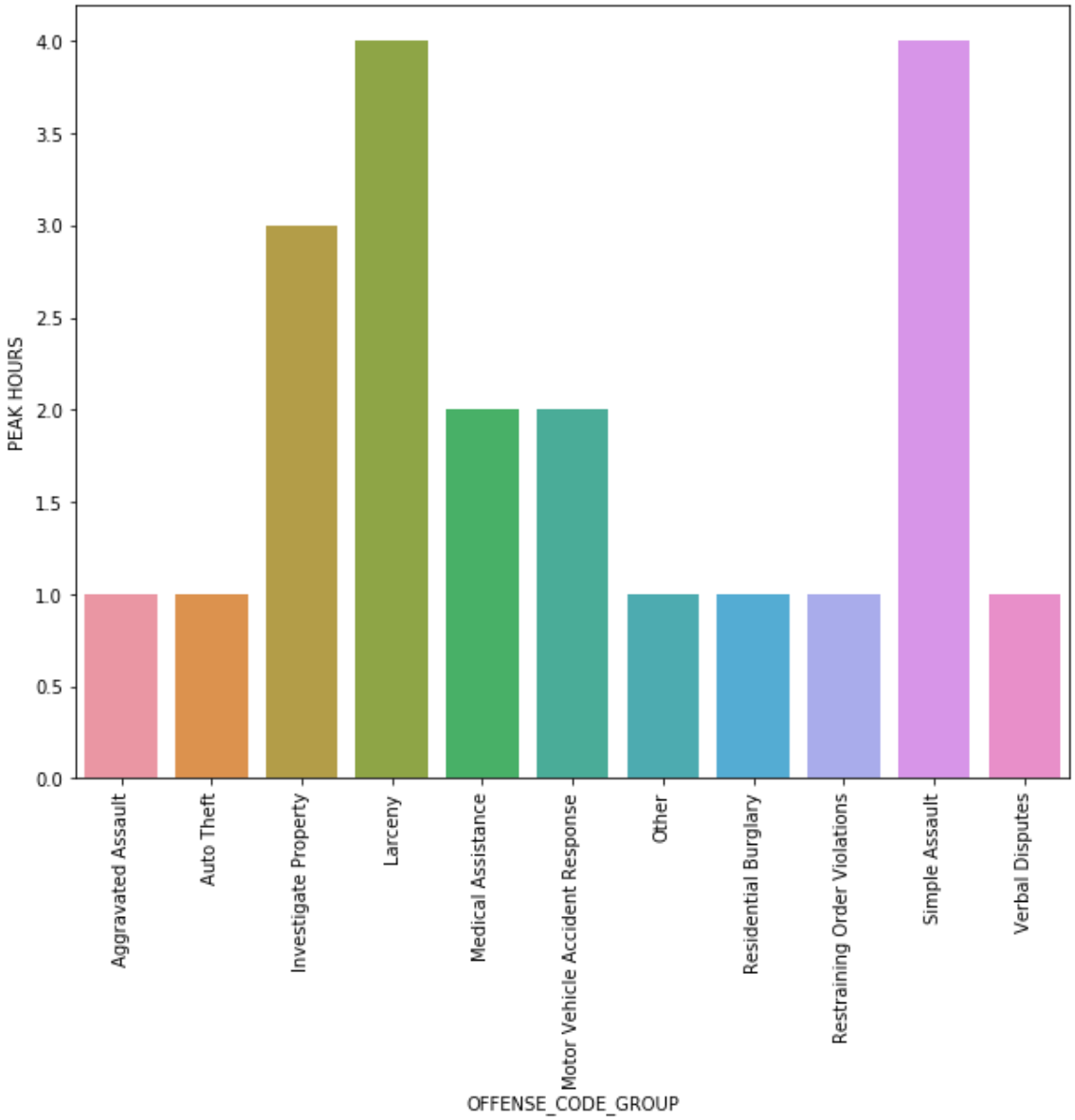


Here we could identify that there is no correlation between month and hour.

Higher chances of getting crime occurance

In [959]:

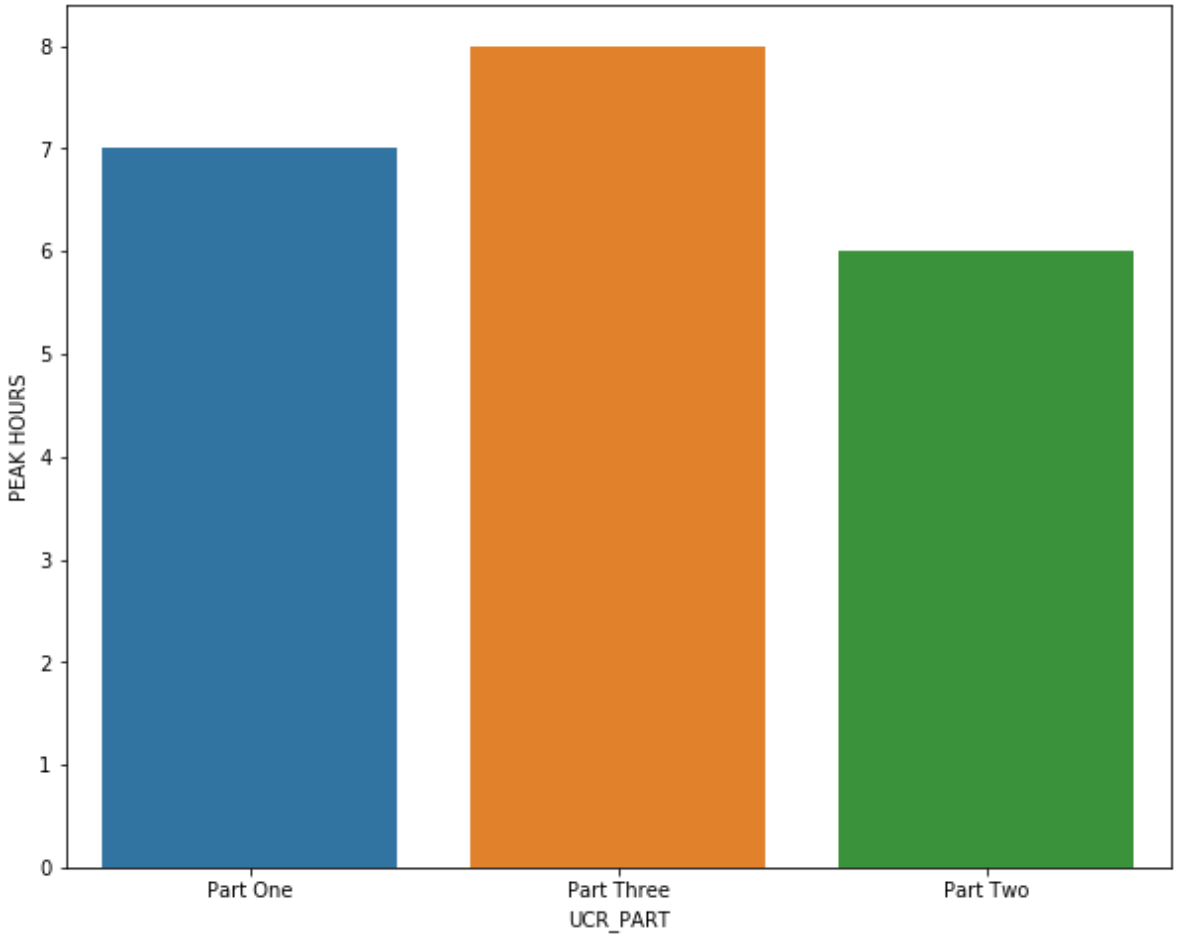
```
# Getting the crime chances of occurance during peak hours
crime = (bos['HOUR']>20)
# Relationship between students with higher admission chance and uni rating
chance=bos[crime].groupby(['OFFENSE_CODE_GROUP'])['HOUR'].size().reset_index(name="PEAK HOU
plot = sea.barplot(data=chance , x="OFFENSE_CODE_GROUP",y="PEAK HOURS")
pl.xticks(rotation = 90)
pl.show()
pl.figure(figsize=(7,7))
```



As per the above graphs we had identified Residential Burglary as the highest occurance crime group, but when we check for the peak hours we could clearly see that during peak hours i.e after 20 hrs. The active crime group that was Larceny and Simple assault.

In [960]:

```
chance=bos[crime].groupby(['UCR_PART'])['HOUR'].size().reset_index(name="PEAK HOURS")
plot = sea.barplot(data=chance , x="UCR_PART",y="PEAK HOURS")
```



From the above graph se can analyse that occurance UCR_Part_Three was the highest among the three UCR_PART and the crime commitment duration was till 8. For Part_One performed till 7 while Part two with least chances of occurance at 6.